UNIVERSITY OF
EASTERN FINLAND

# p300 knockdown impairs dexamethasone-induced transcription of genes but not enhancers in A549 cells

Siiri Sutinen
Master´s Degree Programme in Biomedicine
University of Eastern Finland
Faculty of Health Sciences
School of Medicine
Institute of Biomedicine
20.05.2022

University of Eastern Finland, Faculty of Health Sciences, School of Medicine
Institute of Biomedicine
Master´s Degree Programme in Biomedicine
Sutinen, Siiri S. S.: p300 knockdown impairs dexamethasone-induced transcription of genes but not enhancers in A549 cells
Master of Science thesis; 40 pages, 2 appendices (88 pages)
Supervisors: docent Einari Niskanen, PhD, UEF; assistant professor Nihay Laham Karam, PhD, A. I. Virtanen institute; Bharadwaja Velidendla, MSc, UEF
20.05.2022

Keywords: glucocorticoid receptor, p300, coregulators, protein knockdown, auxin-inducible degron, CRISPR/Cas9

## Abstract

Coregulators like p300 and CBP are crucial for the control of transcription by transcription factors including the anti-inflammatory glucocorticoid receptor (GR) and proinflammatory NF-κB. The presence of coregulators in limiting amounts could also explain the long researched yet incompletely understood GR-mediated repression of NF-κB activity since competition for them would be required. To investigate the primary effects of p300 loss and its contribution to GR–NF-κB crosstalk, the rapid protein knockdown of p300 using the auxin-inducible degron 2 (AID2) system was attempted but unfortunately the clustered regularly interspaced short palindromic repeats (CRISPR)/CRISPR-associated 9 (Cas9)-induced integration of the *Oryza sativa* transport inhibitor response 1 F74G mutant (OsTIR1$^{F74G}$) AID2 system component was unsuccessful. Therefore, p300 knockdown was achieved with short interfering RNA (siRNA) and precision nuclear run-on sequencing (PRO-seq) was done to elucidate the changes in active transcription in A549 human lung adenocarcinoma cells treated with dexamethasone (dex), Kdo2 lipid A (KLA) which activates NF-κB, or both. siRNA-mediated p300 knockdown drastically attenuated gene expression in response to dex but enhancer transcription was essentially unchanged by p300 knockdown. The differentially transcribed genes were often not closest to the differentially transcribed enhancers. Furthermore, at the pathway level, many processes were spared from repression in the absence of p300. Together these results suggest that CBP, a close paralogue of p300, could replace p300 activity to some extent at enhancers but less so at genes, indicating that a certain level of both redundancy and nonredundancy exists between p300 and CBP.

# Introduction

Inflammation is a complex tissue response to infection, irritation, and injury (Gupte *et al*, 2013; Uhlenhaut *et al*, 2013). It is characterized by the dilation of blood vessels and increased vascular permeability at inflammatory foci leading to the leakage of plasma to the inflamed tissue (Cain & Cidlowski, 2017). Vascular permeability together with the increased expression of adhesion molecules on the blood vessel walls facilitates the extravasation of leukocytes such as monocytes and neutrophils to eliminate pathogens (Cain & Cidlowski, 2017; Yu *et al*, 2020). Innate immune system macrophages are essential for the detection of pathogen- or damage-associated molecular patterns via pattern recognition receptors such as Toll-like receptors (TLR) which recognize for example viral nucleic acids and lipopolysaccharide (LPS), a toxin found in the cell walls of gram-negative bacteria (Bekhbat *et al*, 2017; Escoter-Torres *et al*, 2019; Gupte *et al*, 2013; Hoppstädter & Ammit, 2019). Activated macrophages secrete cytokines and chemokines to activate and attract immune cells (Cain & Cidlowski, 2017; Gupte *et al*, 2013). Thus, central to the inflammatory reaction is the *de novo* transcriptional activation of proinflammatory genes including cytokines and chemokines mediated by the proinflammatory transcription factors (TF) nuclear factor kappa-light-chain-enhancer of activated B cells (NF-κB) and activator protein 1 (AP-1) (Bekhbat *et al*, 2017; Gupte *et al*, 2013).

NF-κB is a homo- or heterodimeric TF formed by combinations of the five members of the protein family: p65 (RelA), RelB, c-Rel, p50, and p52 (Oeckinghaus & Ghosh, 2009). All five share an N-terminal Rel-homology domain which is important for dimerization, DNA binding, and nuclear translocation, while only the first three have C-terminal transactivation domains (Bekhbat *et al*, 2017). The canonical NF-κB heterodimer is the most abundant p65/p50 (hereafter NF-κB) which is present in almost all cell types (Oeckinghaus & Ghosh, 2009). When inactive, NF-κB is bound via its Rel-homology domain to NF-κB inhibitor α (IκBα) which masks the nuclear localization signal (NLS) of p65 but not of p50, thus reducing the rate of constant shuttling of NF-κB between the nucleus and the cytoplasm (Fig. 1A) (Bekhbat *et al*, 2017; Oeckinghaus & Ghosh, 2009). NF-κB can be activated by a variety of inflammatory stimuli including LPS *via* TLR4, cytokines such as interleukin (IL) 1α, IL-1β and tumour necrosis factor α (TNF-α), and T- and B-cell antigen receptors (Fig. 1A) (Verstrepen *et al*, 2008). All these pathways converge on the activation of IκB kinase (IKK), which consists of catalytic homologous subunits IKKα and IKKβ and the regulatory subunit IKKγ (Fig. 1A) (Bekhbat *et al*, 2017; Oeckinghaus & Ghosh, 2009; Yu *et al*, 2020). IKKγ recruits TGF-β-activated

kinase 1 (TAK1) by recognizing its lysine 63 (K63)-linked ubiquitin chain functioning as a signalling scaffold, and TAK1 phosphorylates serine 177 (S177) and S181 of IKKβ (Fig. 1A) (Verstrepen *et al*, 2008; Wertz & Dixit, 2010). In addition to NF-κB, TAK1 can also activate mitogen-activated protein kinase (MAPK) signalling cascades which activate AP-1 (Fig. 1A) (Yu *et al*, 2020). IKKβ then phosphorylates IκBα at S32 and S36, causing its dissociation from NF-κB, thus eliminating the nuclear export signal of IκBα and allowing NF-κB to translocate to the nucleus with the help of the motor protein dynein along the microtubules (Fig. 1A) (Bekhbat *et al*, 2017; Nguyen *et al*, 2014). Subsequently, the K48-linked polyubiquitination of IκBα by an Skp1, Cullin and F-box (SCF) family E3 ubiquitin ligase triggers its degradation by the 26S proteasome (Fig. 1A) (Verstrepen *et al*, 2008). In the nucleus, NF-κB binds to κB response elements (κBRE) with the consensus sequence 5'-GGGRNNYYCC-3', thus inducing the expression of proinflammatory genes including cytokines, chemokines, adhesion molecules, complement cascade proteins, and inflammatory enzymes and receptors (Bekhbat *et al*, 2017; Oeckinghaus & Ghosh, 2009).

Much like NF-κB, AP-1 is a homo- or heterodimeric TF composed of proteins with the basic-leucine zipper DNA-binding domain from the Jun, Fos, ATF, and Mef families, of which the ubiquitous and transcriptionally active c-Fos/c-Jun (hereafter AP-1) is considered the canonical AP-1 (Atsaves *et al*, 2019; Smoak & Cidlowski, 2004). Cytokines, chemokines, and microbial pathogens activate MAPK cascades resulting in the activation of extracellular signal regulated kinases (ERK) and c-Jun N-terminal kinase (JNK) which induce the expression of c-Fos and c-Jun, respectively (Fig. 1A) (Mirzaei *et al*, 2020). The phosphorylation of c-Jun by JNK stimulates dimerization and thus facilitates AP-1 binding to TPA-response elements (TRE) with the consensus sequence 5'-TGASTCA-3' (Fig. 1A) (Atsaves *et al*, 2019; Nagesh *et al*, 2021; Smoak & Cidlowski, 2004).

However, dysregulated NF-κB activity is associated with the pathogenesis of chronic inflammation and autoimmunity (Yu *et al*, 2020). Moreover, NF-κB and AP-1 dysregulation is associated with cancer as they promote cancer proliferation, survival, and angiogenesis, and thus contribute to the inflammatory tumour microenvironment known to be one of the hallmarks of cancer (Gazon *et al*, 2017; Nguyen *et al*, 2014; Yu *et al*, 2020). Therefore, the repression of proinflammatory genes by glucocorticoids (GC) is essential to attenuate immune responses which, when overactivated, could be lethal, like for example the cytokine storm associated with coronavirus disease 2019 (Hu *et al*, 2021; Hudson *et al*, 2018). GCs are among the most widely prescribed anti-inflammatory drugs worldwide for the treatment of acute inflammation caused by microbial infection and

chronic inflammatory diseases such as asthma, allergies, rheumatoid arthritis, multiple sclerosis, eczema and psoriasis as oral, topical, and inhaled solutions (Hoppstädter & Ammit, 2019; Kadmiel & Cidlowski, 2013; Oh *et al*, 2017; Vandevyver *et al*, 2014). GCs can also be used for the treatment of some cancers or as adjuvant treatment for the alleviation of cancer therapy-related side effects (Escoter-Torres *et al*, 2019; Prekovic *et al*, 2021).

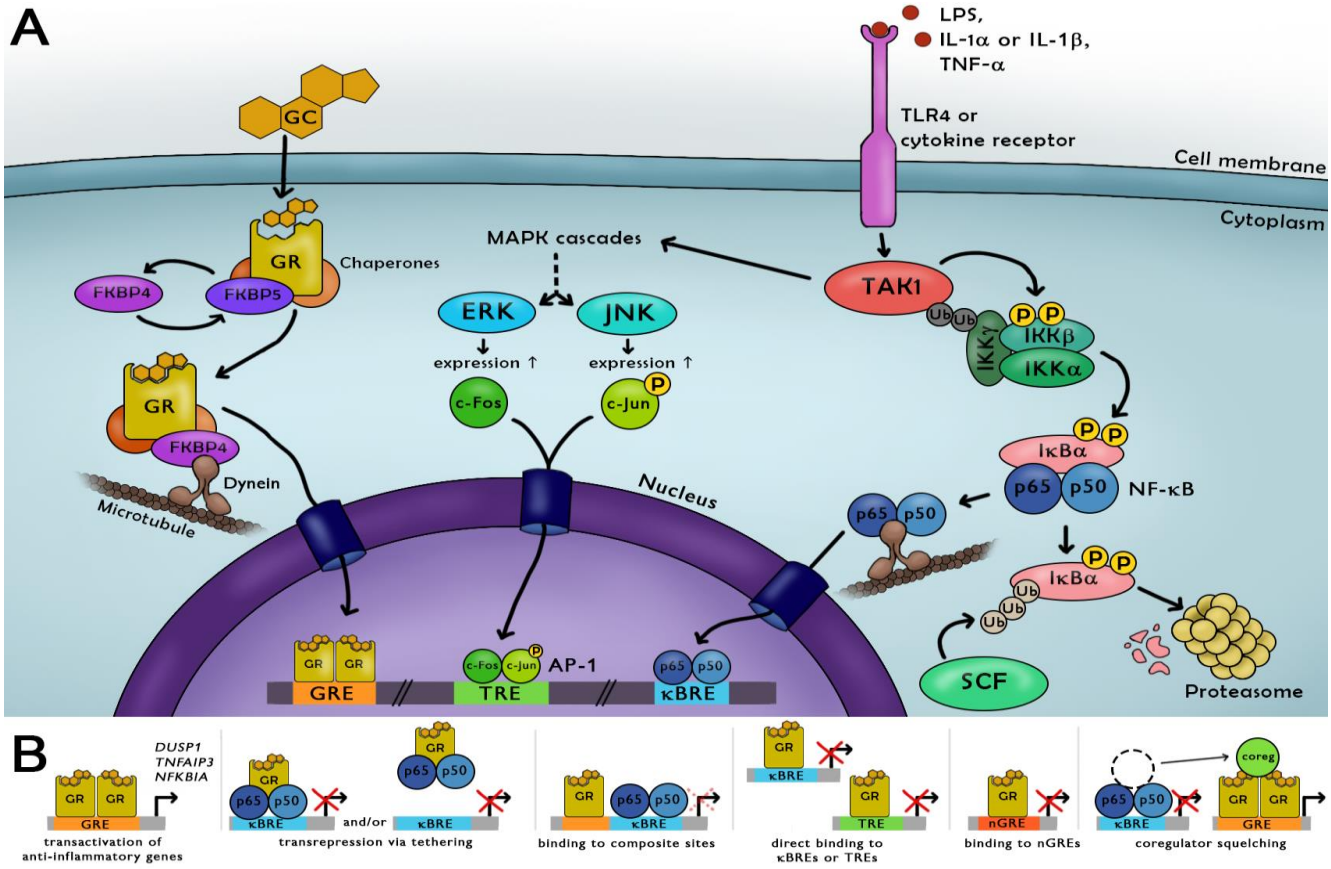



**Figure 1. GR, NF-κB, and AP-1 signalling pathways and various mechanisms of GR-mediated repression of NF-κB and AP-1 target genes. A)** GR (left), NF-κB (right), and AP-1 (middle) are activated by various signals and all three translocate to the nucleus to bind to their respective response elements. See text for details. P, phosphorylation; Ub (beige), K48-linked polyubiquitin; Ub (grey), K63-linked polyubiquitin. **B)** GR can repress inflammation by binding to GREs, directly to NF-κB or AP-1, to composite sites, to κBREs or TREs, to negative GREs (nGRE), or *via* coregulator (coreg) squelching. The outcome of transcription at composite sites depends on the context and can be either activating or repressing.

The endogenous GC in humans is the steroid hormone cortisol which is synthesized from cholesterol in the mitochondria of the zona fasciculata in the adrenal cortex and secreted in a diurnal and stress-responsive fashion under the control of the hypothalamus-pituitary-adrenal axis (Escoter-Torres *et al*, 2019; Kadmiel & Cidlowski, 2013). From systemic circulation, lipophilic GCs diffuse through the cell membrane and bind to their intracellular GC receptor (GR). The 777-amino acid GR encoded by exons 2-9 of the nine-exon gene *NR3C1* and ubiquitously expressed by nearly all nucleated vertebrate cells is a member of the nuclear hormone receptor

family of ligand-activated TFs (Greulich *et al*, 2016). Inactive GR resides predominantly in the cytoplasm as a monomer sequestered *via* its C-terminal ligand-binding domain in a multimeric chaperone complex which maintains GR in a conformation with high affinity for its ligand and prevents its degradation (Fig. 1A) (Cain & Cidlowski, 2017; Escoter-Torres *et al*, 2019; Vandevyver *et al*, 2014; Weikum *et al*, 2017a). Upon ligand binding, FK506 binding protein 5 (FKBP5) of the chaperone complex is exchanged for FKBP4 which associates with dynein to facilitate nuclear translocation of GR (Fig. 1A) (Vandevyver *et al*, 2012). In the nucleus, GR binds to GC response elements (GRE) on the same side of the DNA strand in a head-to-head conformation *via* contacts between the Proximal box in the first of two zinc fingers of the DNA-binding domain (DBD) and the major groove of DNA, and the DBDs of two GRs homodimerize (Fig. 1A) (Bekhbat *et al*, 2017; Syed *et al*, 2020; Vandevyver *et al*, 2014; Weikum *et al*, 2017a). Classical GREs are two inverted imperfect palindromic, hexameric half-sites separated by a 3-bp spacer with the consensus sequence 5'-AGAACANNNTGTTCT-3' where the first and third bases of each half site are the least conserved (Greulich *et al*, 2016; Kadmiel & Cidlowski, 2013). GR predominantly binds to intergenic or intronic enhancers, which are TF-binding non-coding regions distal to the promoters and transcription start sites (TSS) they regulate, and upregulates transcription using the N-terminal ligand-independent transcription activation function 1 (AF-1) or C-terminal ligand-dependent AF-2, both of which interact with coregulators (Borczyk *et al*, 2021; Escoter-Torres *et al*, 2019; McDowell *et al*, 2018; Sacta *et al*, 2018; Vandevyver *et al*, 2014).

GCs regulate up to 20% of the genome through GR, thus affecting numerous processes ranging from glucose metabolism and development to inflammation and cell proliferation. Due to these pleiotropic effects, the prolonged or excessive use of GCs causes detrimental side effects such as hyperglycemia, insulin resistance, hepatic steatosis, adipocyte hypertrophy leading to central obesity, fluid retention, muscle and skin atrophies, osteoporosis, and hypertension (Escoter-Torres *et al*, 2019; Weikum *et al*, 2017a). Therefore, the cell-specific activity of GR is tightly regulated on many levels including multiple isoforms with different activities arising from alternative splicing and alternative translation start sites, mRNA stability control, post-translational modifications, chromatin architecture and DNA methylation at GR binding sites and the coregulators recruited to them (Cain & Cidlowski, 2017; Greulich *et al*, 2016; Johnson *et al*, 2018; Syed *et al*, 2020; Vandevyver *et al*, 2014). Another problem with GC treatments is GC resistance, *i.e.* unresponsiveness to the therapeutic effects of GCs, which some patients may develop

(Dendoncker *et al*, 2019). Thus, CG resistance and these side effects motivate the search for new and improved approaches to the treatment of inflammation such as new drugs or drug targets.

NF-κB is similarly tightly regulated, including the opposition of NF-κB-mediated inflammation by GR (Bekhbat *et al*, 2017). In fact, GC production is triggered by cytokines such as TNF-α and IL-1, thus creating a feedback loop for the induction of NF-κB (Cain & Cidlowski, 2017). By binding to classical GREs, GR upregulates anti-inflammatory genes including *NFKBIA*, *DUSP1*, and *TNFAIP3* which code for IκBα, a phosphatase attenuating MAPK signalling, and a deubiquitinase inhibiting NF-κB activation, respectively (Fig. 1B) (Newton *et al*, 2017; Wertz & Dixit, 2010). This mechanism is referred to as transactivation. On the other hand, transrepression is a mechanism where GR tethers to, *i.e.* physically interacts with, NF-κB or AP-1, specifically with the p65 and c-Jun subunits, respectively, to inhibit their activity without binding to DNA (Fig. 1B) (Cain & Cidlowski, 2017; Greulich *et al*, 2016). Indeed, up to half of the GR cistrome has been found to be co-occupied by either p65 or c-Jun or both, and GR has been reported to gain a large number of binding sites with NF-κB and AP-1 motifs in a cotreatment of GC and LPS (Uhlenhaut *et al*, 2013). However, whether tethering displaces NF-κB from DNA or not is debated as both a global reduction and the persistence of NF-κB binding in the presence of GC and have been reported (Fig. 1B) (Oh *et al*, 2017; Sacta *et al*, 2018). Traditionally, the anti-inflammatory effects of GCs are thought to arise from the transrepression of cytokines whereas transactivation of metabolic genes leads to adverse side effects (Escoter-Torres *et al*, 2020; Lim *et al*, 2015). Though, this notion is overly simplified because it has been reported that whether a gene is up- or downregulated by GR cannot be predicted based on what TFs or motifs are present at a binding site and that GREs can mediate repression and tethered sites can mediate activation of target genes (Uhlenhaut *et al*, 2013). Additionally, GR and NF-κB can work synergistically in specific contexts, for example in the upregulation of *NFKBIA* and *TNFAIP3* or with a low GC dose (Cain & Cidlowski, 2017; Newton *et al*, 2017).

Other mechanisms of GR–NF-κB crosstalk include the binding of GR to composite sites, *i.e.* GR half-sites or full GRE motifs adjacent to binding sites for other TFs, where it can inhibit or enhance the activity of the partner TF (Fig. 1B) (Cain & Cidlowski, 2017; Weikum *et al*, 2017a). In fact, it has been found that GR binding to half-sites is more prevalent than to GREs and thus could facilitate tethering between DNA-bound TFs at these sites (Lim *et al*, 2015). Together with the discovery by Escoter-Torres *et al.* that DNA binding by GR is required for both up and downregulation of genes,

this suggests that these sites could significantly contribute to GR–NF-κB crosstalk and that tethering directly to NF-κB alone cannot explain the repression of proinflammatory genes by GR (Escoter-Torres *et al*, 2020). The latter suggestion is also supported by the finding that cotreatment to activate GR and NF-κB did not drastically increase the number of binding sites for either, which would be explained by tethering (Rao *et al*, 2011). Moreover, GR binding may not be limited to GREs and half-sites because a recent study reports that monomeric GR can bind directly, specifically, and independently of NF-κB presence to a highly conserved 5′-AATTY-3′ sequence not bound by NF-κB in the spacer of κBREs and repress genes associated with these κBREs (Fig. 1B) (Hudson *et al*, 2018). Similarly, monomeric GR has been found to directly bind to a 5′-TGASTC-3′ sequence resembling a GR half-site within TREs with similar affinity as to GREs through base-specific interactions even in the absence of AP-1 (Fig. 1B) (Weikum *et al*, 2017b). GR has also been proposed to bind to negative GREs with the consensus sequence 5′-CTCC$(N)_{0\text{-}2}$GGAGA-3′ as two separate monomers on opposite sides of the DNA strand in a head-to-tail or tail-to-tail fashion and recruit corepressors to inhibit transcription (Fig. 1B) (Bekhbat *et al*, 2017; Cain & Cidlowski, 2017; Surjit *et al*, 2011; Weikum *et al*, 2017a, 2017b). However, evidence for this *in vivo* is still lacking and many other studies report detecting little to no presence of these sites in the genome or their interaction with GR (Lim *et al*, 2015; Sasse *et al*, 2019; Syed *et al*, 2020; Uhlenhaut *et al*, 2013). Yet, despite decades of intensive research into the various aspects of GR–NF-κB crosstalk and multiple contradicting reports, these mechanisms remain incompletely understood.

Besides binding to DNA or other TFs, a broader concept known as coregulator squelching could apply to GR–NF-κB crosstalk. Coregulator squelching is defined as competition between TFs for a limited number of coregulators in a cell leading to the repression of one of these TFs (Fig. 1B) (Schmidt *et al*, 2016). Coregulators, which in contrast to TFs do not bind DNA sequence-specifically, can remodel chromatin, modify histones, or facilitate RNA Polymerase II (Pol II) recruitment and thus are crucial for the control of transcription by GR and NF-κB (Greulich *et al*, 2016). The ubiquitously expressed paralogous histone acetyltransferases and coregulators p300 and CREB-binding protein (CBP) (often denoted p300/CBP) are recruited by GR, NF-κB, and AP-1, making them prime targets for coregulator squelching (Escoter-Torres *et al*, 2019; Fonte *et al*, 2007; Mirzaei *et al*, 2020; Oeckinghaus & Ghosh, 2009). CBP and p300 are responsible for the deposition of activating acetylation marks on K27 of histone 3 (H3K27ac) at promoters and enhancers, which has been shown to be important for Pol II recruitment and, by extension, gene expression (Bedford & Brindle, 2012; Jin *et al*, 2011; Martire *et al*, 2020). While coregulator

squelching is controversial and remains to be formally proven, it is strongly supported by the recent discovery by Gillespie *et al.* that coactivators, including p300/CBP, are present in the nucleus in limiting amounts compared to TFs whereas corepressors are present in excess compared to TFs (Gillespie *et al*, 2020; Schmidt *et al*, 2016). Further support for this hypothesis is provided by findings that equally many genes are upregulated and downregulated by GR, reflecting the redistribution of coregulators from repressed to activated genes (Borczyk *et al*, 2021; McDowell *et al*, 2018), that GR interacted with p300 less after TNF treatment, suggesting that NF-κB sequestered p300, but p300 overexpression remedied this (Dendoncker *et al*, 2019), and that local GR occupancy is not required for the repression of transcription in BEAS-2B cells (Sasse *et al*, 2019). On the other hand, De Bosscher *et al.* report that repression of NF-κB or AP-1 by GCs was not explained by limiting amounts of CBP or p300 in HEK293T cells as the transcription of the *IL6* reporter gene was repressed by GC even with the overexpression of p300 or CBP (De Bosscher *et al*, 2000, 2001). Thus, the role of p300 in the control of transcription by GR and NF-κB could be relevant to GR–NF-κB crosstalk but is not fully known.

The role of a protein of interest (POI) in living cells or *in vivo* can be studied with the help of loss-of-function experiments where the expression of the POI is silenced by constitutive knockout or conditional knockdown (Natsume & Kanemaki, 2017; Saito & Kanemaki, 2021; Yesbolatova *et al*, 2020). The advantage of knockdown over knockout is its reversibility and the possibility to deplete proteins essential for survival (Natsume *et al*, 2016; Nishimura & Fukagawa, 2021). The conditional depletion of the POI can be controlled at the level of DNA or mRNA using for example conditional promoters or short interfering RNA (siRNA), respectively (Natsume & Kanemaki, 2017). However, the indirect depletion of the POI that these methods employ depends on the stability and natural turnover of the POI (Nishimura *et al*, 2009). This makes these methods slow, especially for long-lived proteins (Holland *et al*, 2012). For example, the commonly utilized siRNA typically requires 2-3 days to deplete the POI (Natsume & Kanemaki, 2017). Such gradual depletion is the major limitation of these methods as it could produce a different phenotype caused by adaptation to the loss of the POI compared to a faster depletion (Holland *et al*, 2012; Nishimura *et al*, 2009). Conversely, due to its rapidity, knockdown at the protein level is better compared to DNA or mRNA level methods because it allows the investigation of the immediate primary phenotype resulting as a direct consequence of the loss of the POI before the phenotype is complicated or compromised by secondary effects (Natsume & Kanemaki, 2017; Negishi *et al*, 2021; Saito & Kanemaki, 2021; Yesbolatova *et al*, 2020). Furthermore, protein knockdown is also rapidly

reversible as only *de novo* translation but not transcription is required for new protein to begin accumulating (Holland *et al*, 2012).

One method for targeted and timed conditional protein knockdown is the auxin-inducible degron (AID) system developed by Nishimura *et al.* which is based on the exogenous expression of the plant-specific transport inhibitor response 1 (TIR1) protein in non-plant cells (Adhikari *et al*, 2021). As the F-box subunit of the SCF E3 ubiquitin ligase, TIR1 complexes with the other conserved endogenous SCF pathway components (Fig. 2A) (Holland *et al*, 2012; Negishi *et al*, 2021). Upon binding of the natural phytohormone auxin indole-3-acetic acid (IAA) or its derivatives to the auxin receptor TIR1, the POI fused with a degron domain is recruited, and the degron domain functioning as the substrate is polyubiquitinated by the SCF complex and targeted for proteolysis by the 26S proteasome similarly to NF-κB (Fig. 2A) (Adhikari *et al*, 2021; Natsume & Kanemaki, 2017; Natsume *et al*, 2016; Nishimura *et al*, 2009; Yunusova *et al*, 2021). Unfortunately, the original AID system suffered from two drawbacks: basal degradation even in the absence of administered IAA most likely caused by auxin-like indole chemicals in the serum in cell culture medium, and the requirement of a high auxin dose for activation, which was found to cause slow cell growth and alterations in gene expression (Yesbolatova *et al*, 2019, 2020). The recently developed improved AID2 system overcomes these drawbacks by using *Oryza sativa* TIR1 F74G ($OsTIR1^{F74G}$) mutant with negligible basal degradation and a more potent and more lipophilic ligand 5-phenyl-IAA (5-Ph-IAA) with no side effects and possibly higher cell permeability instead of wild type (wt) OsTIR1 and IAA (Fig. 2A) (Negishi *et al*, 2021). For the degron tag, AID2 uses the 68-amino acid, 7-kDa mini-AID (mAID) tag which constitutes of amino acids 65-132 of the original 25-kDa tag derived from *Arapidopsis thaliana* (Natsume *et al*, 2016; Natsume & Kanemaki, 2017; Saito & Kanemaki, 2021). Compared to AID, AID2 was shown to be even quicker and to require less $OsTIR1^{F74G}$ for the efficient degradation of the POI due to higher expression of $OsTIR1^{F74G}$ than wtOsTIR1 which was possibly a target for basal self-degradation (Yesbolatova *et al*, 2020). The AID and AID2 systems can be utilized in yeast, *in vivo* in mice, *Caenorhabditis elegans* nematodes and *Drosophila melanogaster* fruit flies, and in a wide variety of both transformed and non-transformed cell lines from chicken, mouse, monkey, and human (Bence *et al*, 2017; Holland *et al*, 2012; Negishi *et al*, 2021; Nishimura *et al*, 2009; Venz *et al*, 2021; Yesbolatova *et al*, 2020). Furthermore, AID allows the degradation of cytoplasmic, nuclear, membrane-binding, and transmembrane proteins, including ones incorporated in macromolecular complexes (Holland *et al*, 2012; Venz *et al*, 2021). For example, AID has been successfully used for the degradation of two chromatin-binding proteins

with a normal half-life of up to 24 h, the chromatin architectural protein CCCTC-binding factor (CTCF) (Luan *et al*, 2021) and the pluripotency TF octamer-binding transcription factor 4 (OCT4) (Bates *et al*, 2021), suggesting that AID2 could be applied to successfully study the role of p300 in GR–NF-κB crosstalk.

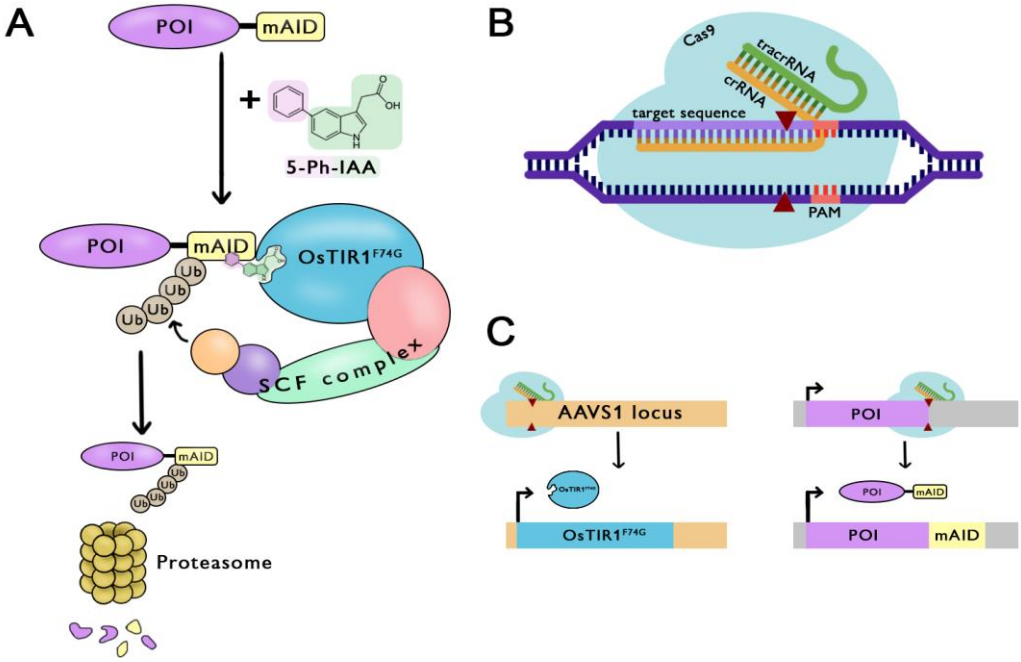



**Figure 2. AID2-mediated degradation of mAID-tagged POI (A), the structure of the CRISPR/Cas9 system (B), and CRISPR/Cas9-mediated integration of OsTIR1$^{F74G}$ and mAID tag. A)** In the presence of 5-Ph-IAA, the mAID-tagged POI is polyubiquitinated (Ub) by the SCF complex and degraded by the proteasome. **B)** The CRISPR/Cas9 system consisting of the Cas9 enzyme, crRNA, and tracrRNA induces a DSB (dark red triangles) at the target site. **C)** For a functional AID2 system, OsTIR1$^{F74G}$ and mAID must be integrated to the genome and expressed.

The prerequisite for a functional AID2 system is the stable expression of its two components: OsTIR1$^{F74G}$ and the chimeric fusion POI with mAID inserted either at the start codon or before the stop codon for in frame N- or C-terminal tagging, respectively (Fig. 2C) (Adhikari *et al*, 2021; Natsume & Kanemaki, 2017; Yunusova *et al*, 2021). To introduce these components, the clustered regularly interspaced short palindromic repeats (CRISPR)/CRISPR-associated 9 (Cas9) DNA editing method can be employed. CRISPR/Cas9 was derived from the natural bacterial adaptive immune defense mechanism against foreign genetic elements such as phage infection and plasmid transfer. The system consists of the RNA-guided Cas9 endonuclease which induces a sequence-specific double-strand break (DSB) using two distinct nuclease domains and a guide RNA (gRNA) which directs Cas9 to the 20-bp target DNA sequence *via* complementary base pairing (Fig. 2B). In the natural CRISPR/Cas9 system, the gRNA consists of CRISPR RNA (crRNA) which confers the sequence specificity and trans-activating crRNA (tracrRNA) which pairs with the crRNA and binds to Cas9 to form a ribonucleoprotein (RNP) complex (Fig. 2B). Alternatively, the crRNA and tracrRNA

can be fused to form a synthetic, chimeric single gRNA. CRISPR/Cas9 cuts the DNA 3 bp upstream of the protospacer adjacent motif (PAM) which must be flanking the 3' target site directly (Fig. 2B). The PAM sequence is 5'-NGG-3' for the most frequently used *Streptococcus pyogenes* Cas9. The DSB is subsequently repaired either by error-prone non-homologous end joining or high-fidelity homology-directed repair (HDR) using a typically exogenously introduced homologous repair template. Off-target DSBs can cause unwanted mutations at sites that have sequence homology with the target site and thus they should be avoided. By introducing a D10A or H840A mutation to one of the two nuclease domains, Cas9 can be converted to a nickase which induces only a single-strand break on the target strand complementary to the gRNA or the nontarget strand, respectively. A staggered DSB with overhangs can be created using a pair of appropriately spaced and oriented sense and antisense gRNAs targeting opposite DNA strands. This double nicking minimizes the frequency of unwanted off-target DSBs because a DSB can only occur if both gRNAs recognize a target sequence within a limited distance and because double nicks are preferentially repaired through HDR (Jiang & Doudna, 2017; Ran *et al*, 2013).

The AID2-mediated rapid depletion of p300 alone is not necessarily sufficient to gain insight into the effects of p300 knockdown on GR–NF-κB crosstalk. Rather, to investigate the immediate effects of p300 knockdown, precision nuclear run-on sequencing (PRO-seq) can be employed to investigate the level of nascent transcription, compared to RNA sequencing which does not differentiate nascent transcripts from the total transcriptome. Furthermore, the transcription at a TSS is bidirectional and produces short unstable non-coding enhancer RNAs (eRNA) from enhancers (Mahat *et al*, 2016; Sasse *et al*, 2019). These eRNAs can be used to identify active enhancers which have been shown to correlate with actively transcribed genes (Greulich *et al*, 2021). Unlike RNA sequencing, PRO-seq can detect eRNAs much more reliably as it quantitatively maps the genome-wide occupancy of active Pol II with strand specificity at a resolution of up to one bp compared to tens of bases for global run-on sequencing. This is achieved by isolating nuclei from cells on ice and washing away native nucleotides to halt active transcription without disrupting engaged Pol II. Transcription is then re-engaged in a controlled manner under run-on conditions by incubating the nuclei with biotin-labeled nucleoside triphosphates (NTP), which active Pol II incorporates into nascent transcripts during elongation, thus labeling the transcripts. Finally, biotin-labeled mRNA is affinity purified using streptavidin-coated magnetic beads and used to prepare libraries which are size selected to remove adaptor or primer dimers and sequenced (Mahat *et al*, 2016).

In this study, to examine the role of p300 in GR–NF-κB crosstalk, the integration of the first component for AID2 system setup, OsTIR1$^{F74G}$, into the so-called safe-harbour adeno-associated virus integration site 1 (*AAVS1*) locus of A549 cells was attempted using a Cas9 D10A nickase and the crRNA-tracrRNA setup (Fig. 2B, 2C) (Saito & Kanemaki, 2021). A549 is a human epithelial cell line from the adenocarcinoma subtype of non-small cell lung cancer (Luk *et al*, 2001; Rao *et al*, 2011). Besides macrophages, A549 cells are commonly used to study GR and NF-κB actions and thus have an abundance of data from previous studies available, making them an excellent model to be used here. After transfection of CRISPR/Cas9 *via* lipofection, single-cell colonies were screened by genomic PCR and Western blotting but unfortunately the integration was found to be unsuccessful. Therefore, p300 knockdown was achieved using siRNA, and PRO-seq was done to identify genes that are actively transcribed in response to treatment with the synthetic GC dexamethasone (dex), the synthetic structural component of LPS and TLR4-specific agonist Kdo2-lipid A (KLA), or both (dexKLA) (Sims *et al*, 2010; Weikum *et al*, 2017a). Surprisingly, KLA had virtually no effect on A549 cells even though A549 are known to respond to LPS, and this is most likely because TLR4 was found to not be expressed. Regardless, PRO-seq revealed that p300 knockdown repressed the expression of many genes but enhancer transcription was not impaired similarly. Though, not all genes or pathways were impaired by the absence of p300. Moreover, actively transcribed enhancers were not associated with the nearest actively transcribed genes. Therefore, this study proposes that the redundancy between p300 and CBP may be gene-specific and that different mechanisms may regulate gene and enhancer transcription.

## Materials and methods

### Cell culture

A549 cells (ATCC) were cultured in Ham's F-12K (Kaighn's) Medium (21127-022, Gibco) supplemented with 10% (vol/vol) fetal bovine serum (FBS) (10270-106, Gibco), 100 U/ml penicillin and 100 µg/ml streptomycin (5140122, Thermo Fisher) and incubated in 37 °C and 5% $CO_2$. For 24 h before hormone treatments, cells were cultured in F-12K medium supplemented with 2.5% charcoal stripped FBS. For PRO-seq experiments, cells were treated with 100 nM dex (02194561-CF, MP Biochemicals), 100 ng/ml KLA (SML2430, Sigma-Aldrich), both, or equivalent volumes of absolute ethanol (EtOH) and 1x phosphate-buffered saline (PBS) as control for 1 h as two replicates before nuclear extraction. Cos-1 monkey fibroblast cells (ATCC) were cultured in

Dulbecco's Modified Eagle Medium (41965-039, Gibco) supplemented with 10% (vol/vol) FBS, 25 U/ml penicillin and 25 µg/ml streptomycin and incubated in 37 °C and 5% $CO_2$.

## Plasmid and CRISPR/Cas9 transfections

For transfections, cells were seeded to 6-well plates with 300,000-350,000 cells/well. To induce genomic integration of OsTIR1$^{F74G}$, A549 cells were transfected with 2.5 µg of pMK381 (#140536, Addgene) containing genes for OsTIR1$^{F74G}$ and puromycin resistance using TransIT®-LT1 (MIR2305, MirusBio) according to the manufacturer's instructions using the 3:1 ratio of TransIT®-LT1:plasmid. The next day, the cells were forward transfected with RNP complexes consisting of Alt-R CRISPR-Cas9 crRNA (IDT), Alt-R CRISPR-Cas9 tracrRNA (1077024, IDT) and Alt-R S.p. Cas9 enzyme (1081063, IDT) according to the manufacturer's instructions. Briefly, gRNA oligos were prepared by mixing crRNA and tracrRNA in equimolar concentrations, after which equimolar concentrations of the oligos and Cas9 enzyme were mixed with Cas9 PLUS™ Reagent (100035636, Invitrogen) to form the RNP complex. The RNP complex was mixed with Lipofectamine™ CRISPRMAX™ (100035631, Invitrogen) and applied dropwise to the cells after a 20-min incubation. The CRISPR/Cas9 RNP was forward transfected because the pMK381 transfection was done on the previous day. It was tested whether it was optimal to transfect pMK381 or CRISPR/Cas9 RNP first (Supplementary Method 1), but it was found that the order did not make a difference (Supplementary Method 1, Fig. S1). The crRNA-tracrRNA setup is advantageous over single gRNA when targeting multiple sites, which is the case when using a Cas9 nickase, because target-specific design is needed only for the crRNAs while the tracrRNA is the same for all reactions. The following crRNAs were used: crRNA1, 5'-GACAGAAAAGCCCCATCCTT-3'; crRNA2, 5'-TGTCCCTAGTGGCCCCACTG-3'; crRNA3, 5'-CTAGTGGCCCCACTGTGGGG-3'; and crRNA4, 5'-ACTAGGAAGGAGGAGGCCTA-3'. The pairs used were crRNA 1+2, 1+3 and 4+2, with the first two for batch 1 and all three for batch 2. crRNA pairs were combined at the RNP complex formation or the transfection step for batch 1 and batch 2, respectively. For batch 2, in addition to circular pMK381, linear pMK381 was used as it showed promise in test transfections (Fig. S1). pMK381 was linearized by digesting with EcoRI (#FD0274, Fermentas) according to the manufacturer's instructions, the restriction enzyme was heat inactivated at 80 °C for 5 min and 2.5 µg of linearized pMK381 was transfected as described above, followed by either crRNA pair 1+2 or no CRISPR/Cas9 for random integration the next day. To generate a positive control for OsTIR1$^{F74G}$ Western blotting, Cos-1 cells were transfected with

pMK381 as described above and harvested by scraping in 1x PBS and centrifuging at 150 x g for 10 min.

## Generation and harvesting of single-cell colonies

Two days after transfection of the CRISPR/Cas9 system, the cells were diluted to approx. 150,000 cells/10-cm dish and antibiotic selection with puromycin (P8833, Sigma-Aldrich) was started initially with 1.5 µg/ml but after all cells seemed to be dying the concentration was dropped to 0.5 µg/ml determined to be sufficient by a kill curve test (Supplementary Method 2). Once single-cell colonies were visible by naked eye, they were picked with a pipette tip to a 96-well plate containing 20 µl of trypsin to disperse the cells and, after a short incubation, complete growth medium with puromycin was added. After the well was ≥70% confluent, the cells were trypsinized and half was split to a new 48 or 96-well copy plate. For genomic DNA (gDNA) extraction, cells at ≥70% confluence were harvested by scraping in QuickExtract™ DNA Extraction Solution 1.0 (#SS000035-D2, Biosearch). Next, a denaturation of 65 °C for 15 min, 68 °C for 15 min, 98 °C for 10 min was performed. The samples were stored at -20 °C until genomic PCR analysis.

## Genomic PCR

To assess OsTIR1$^{F74G}$ integration status, single-cell colonies were screened by running PCRs 1 and 2 (Table 1) using Phusion™ High-Fidelity DNA polymerase (F-530L, Thermo Scientific) in 20-µl reactions of 1x Phusion™ HF buffer (F-518, Thermo Scientific), 200 µM dNTP each, 0.5 µM forward and reverse primer each, 0.02 U/µl Phusion™ High-Fidelity DNA polymerase. 1:2 diluted colony gDNA was used as template as suggested by optimization of the template concentration in the presence of QuickExtract™ lysis reagent (Supplementary Method 3, Fig. S2A). The following PCR program was run: heat inactivation of QuickExtract™ at 98 °C for 10 min after which polymerase was added; 98 °C for 30 sec; 40 cycles of 98 °C for 10 sec, annealing for 30 seconds at the appropriate temperature for the primer pair (Table 1) and extension at 72 °C for the appropriate time for the primer pair (Table 1); and final extension at 72 °C for 5 min. The primer annealing temperatures were optimized in the presence of QuickExtract™ (Supplementary Method 3, Fig. S2B-D). All PCR products were prepared in Southern buffer (10% glycerol, 0.025% bromophenol blue) and electrophorized on 0.8% agarose gels with GeneRuler 1 kb DNA Ladder (SM0311, Thermo Scientific) with 10x SYBR® Green I (S7563, Invitrogen). A few single cell colonies were screened with PCR3 (Table 1) as described above except adding 3% DMSO to the PCR mix. For this wtA549 gDNA and pMK381 were used as negative and positive control, respectively, and

QuickExtract™ was added at a 1:2 dilution to mimic single-cell colonies. To analyze the sequence of these PCR products, they were digested with HpaII (1x Tango Buffer [BY5, Thermo Scientific], 0.5 U/µl HpaII [#ER0511, Thermo Scientific], 30% [vol/vol] PCR product) at 37 °C for 2 h and then electrophorized (100 V) on a 1.5% agarose gel with 1:2 diluted GeneRuler 100 bp Plus DNA Ladder (SM0321, Thermo Scientific) with 10x SYBR.

**Table 1. Primer pairs, sequences, annealing temperatures, and extension times for genomic PCR __ reactions 1, 2 and 3.**

| | Forward primer | | Reverse primer | | Annealing temperature | Extension time |
|---|---|---|---|---|---|---|
| | Name | Sequence (5′ → 3′) | Name | Sequence (5′ → 3′) | | |
| PCR1 | fw1 | GGTCCAGGCCAAGTAGGTG | rev1 | AGGTCCCTCGAAGAGGTTCA | 63 °C | 30 sec |
| PCR2 | fw1 | GGTCCAGGCCAAGTAGGTG | rev3 | CTCTAACGCTGCCGTCTCTC | 60 °C | 3 min |
| PCR3 | fw2 | CGCCGCTAGGTTTCCAAATG | rev2 | CAGGGACCCAGCATTCACTT | 65 °C | 30 sec |

## siRNA transfections

To test the effectiveness of the p300 siRNA, 100,000 cells/well of A549 cells were plated on a 12-well plate and transfected with 10 pmol of ON-TARGETplus Human EP300 siRNA SMARTPool (L-003486-00-0020, Dharmacon) targeting p300 (siEP300) or ON-Target Plus Non-targeting Control Pool (D-001810-10-20, Dharmacon) as control (siNON) using Lipofectamine® RNAiMAX (13778-150, Invitrogen) according to the manufacturer's instructions. The cells were harvested by scraping in 1x PBS after 1, 2, and 3 days of incubation with siRNAs and centrifuged at 150 x g on day 1 or at 300 x g on days 2 and 3 for 3 min at 4 °C. Based on this test, the siRNA was confirmed to work, and 2 days was selected as the optimum transfection time (Fig. S4). For the PRO-seq experiment, 2 or 3 million A549 cells for batch 1 and 2, respectively, were plated on a 10-cm dish and transfected with 145 pmol of the above siRNAs using RNAiMAX.

## Western blotting

For Western blotting, single-cell colonies were expanded to 6-well scale or Cos-1 cells were grown on 6-well plates. The harvested siRNA samples were resuspended in 30 µl of 1x SDS buffer (66 mM Tris-HCl, 13% glycerol, 2.1% SDS, 0.005% bromophenol blue) with 1x cOmplete™ Protease Inhibitor Cocktail (PIC) (54937500, Roche) and 2% β-mercaptoethanol and sonicated for 10 cycles of 10 sec on/10 sec off on the high setting (Bioruptor, Diagenode). The single-cell colony samples and Cos-1 cells were resuspended in 250 µl of 1x SDS buffer with 1x PIC and sonicated on max settings for 10 sec twice (UP50H, Hielscher Ultrasonics), after which 2% β-mercaptoethanol was added. Next,

all samples were boiled at 95 °C for 5 min. 10 µl of the samples and 5 µl of PageRuler™ Plus Prestained Protein Ladder (#26619, Thermo Scientific) were loaded to the acrylamide gels, which were 5% and 10% acrylamide gels for the siRNA samples and the single-cell colony samples, respectively, both with 4% acrylamide upper gels. The acrylamide gels were electrophorized at 160 V for 10 min and then 200 V for 30-40 min in SDS run buffer (25 mM Tris, 190 mM glycine, 3.5 mM SDS). The proteins were transferred to nitrocellulose membranes using the Mini Trans-Blot® (Bio-Rad) assembled according to the manufacturer's instructions by running at 250 mA for 1 h in transfer buffer (25 mM Tris, 0.192 M glycine, 20% methanol). To assess the protein loading, the membranes were stained with Ponceau S stain (P7170-1L, Sigma-Aldrich) for 1 min at room temperature (RT), rinsed with water to remove excess stain, imaged, and rinsed twice with wash buffer (1x TBS, 0.1% Tween). The membranes were incubated in blocking buffer (5% fat-free milk powder in wash buffer) for 1 h at RT with shaking and then with primary antibodies diluted in blocking buffer overnight at 4 °C with shaking. The primary antibodies used were 1:5000 anti-p300 (A300-358A, Bethyl) for the siRNA samples and 1:1000 anti-OsTIR1 (PD048, MBL) for the single-cell colony samples. After washes of 1x 15 min and 2x 5 min in wash buffer, the membranes were incubated with 1:10,000 diluted secondary antibody goat anti-rabbit IgG (H+L) horseradish peroxidase conjugate (G21234, Invitrogen) in wash buffer at RT for 45 minutes with shaking. After washes of 2x 10 min in wash buffer and 1x 5 min in 1x TBS, the detection was done using Pierce™ ECL Western Blotting Substrate (32106, Thermo Scientific) according to the manufacturer's instructions and imaged with Chemidoc (Bio-Rad).

## Nuclear extraction

After 2-day incubation with siRNA, cells were washed three times with ice-cold 1x PBS and scraped in Swelling buffer (10 mM Tris-HCl [pH 7.5], 2 mM $MgCl_2$, 3 mM $CaCl_2$) with 1 U/ml of SUPERase-In™ RNase Inhibitor (AM2694, Invitrogen) and RNasin® Plus RNase Inhibitor (N261A, Promega) each after a 5-min incubation on ice. The cells were centrifuged at 400 x g for 10 min at 4°C and resuspended in Resuspension buffer (10% glycerol in Swelling buffer, 2 U/ml of SUPERase-In™ and RNasin® Plus each) while combining two replicates to one. Vortex buffer (1% Igepal, 10% glycerol in Swelling buffer, 1 U/ml of SUPERase-In™ and RNasin® Plus each) was added dropwise while vortexing, the samples were incubated for 5 min on ice, and the nuclei were washed twice with Lysis buffer (0.5% Igepal, 10% glycerol in Swelling buffer, 1 U/ml of SUPERase-In™ and RNasin® Plus each) by centrifuging at 600 x g for 5 min at 4 °C. A sample for nuclei counting was taken and

diluted 1:2 in 2:5 Trypan blue (T8154, Sigma-Aldrich):Freezing buffer (40% glycerol, 5 mM $MgCl_2$, 0.1 mM EDTA, 50 mM Tris-HCl [pH 8.3], 1 U/ml of SUPERase-In™ and RNasin® Plus each). After centrifugation at 900 x g for 6 min at 4 °C, nuclei were resuspended with 5 million nuclei/100 µl of Freezing buffer. siRNA transfection batches 1 and 2 were extracted on different days.

## PRO-seq library preparation

A 1-biotin nuclear run-on reaction was done by preparing the mix (50% [vol/vol] extracted nuclei; 5 mM Tris-HCl [pH 8.0]; 2.5 mM $MgCl_2$; 0.5 mM dithiothreitol [DTT]; 150 mM KCl; 25 nM biotin-11-CTP [NU-831-BIOX, Jena Bioscience]; 0.25 nM CTP [NU-1011, Jena Bioscience]; 125 nM ATP, GTP and UTP each [NU-1010, NU-1012, NU-1013, respectively, Jena Bioscience]; 0.5% sarkosyl [L7414-10ML, Sigma Aldrich]; 0.1 U/µl SUPERase-In™ and RNasin® Plus each) and incubating for 5 min at 30 °C. RNA was extracted by adding 600 µl of Trizol LS (10296028, Ambion), incubating for 5 min at RT, adding 120 µl of chloroform and incubating 2-3 min at RT, after which the samples were centrifuged at 12,000 x g for 15 min at 4 °C and the colourless upper layer was extracted. To the extracted RNA, NaOAc, glycogen (#R0561, Fermentas) and isopropanol were added (0.3 M NaOAc, 50% [vol/vol] isopropanol, 20 µg glycogen), the mix was incubated for 30 min at RT and centrifuged at 14,000 x g for 15 min at 4 °C. The pellet was washed twice with 75% ethanol by vortexing and centrifuging at 7,500 x g for 5 min at 4 °C. The pellet was air-dried to remove remaining ethanol, suspended into DNase, RNase free water and incubated for 10 min at 60 °C for concentration measurements with NanoDrop One. RNA fragmentation was done using RNA Fragmentation Reagents (AM8740, Invitrogen) according to the manufacturer's instructions for 2-20 µg of RNA except incubating only for 13 min. Next, the samples were purified using Micro Bio-Spin P-30 Gel Columns (#7326250, Bio-Rad) according to the manufacturer's instructions. RNA was captured by combining equal volume of the samples with 30 µl/library of Dynabeads™ MyOne™ Streptavidin C1 (65001, Invitrogen) beads that were prewashed as described previously (Mahat *et al*, 2016) and incubating for 20 min at RT with rotation. The beads were washed twice with ice-cold high-salt wash buffer (50 mM Tris-HCl [pH 7.4], 2 M NaCl, 0.5% [vol/vol] Triton-X-100), twice with binding buffer (10 mM Tris-HCl [pH 7.4], 300 mM NaCl, 0.1% [vol/vol] Triton-X-100) and once with low-salt wash buffer (5 mM Tris-HCl [pH 7.4], 0.1% [vol/vol] Triton-X-100) for 1 min. Next, the RNA extraction was repeated as above except using 400 µl of Trizol LS, incubating for 3 min and then adding 100 µl of chloroform. The extracted RNA pellet was resuspended in 1x TE – 0.05% Tween, vortexed to dissolve and heated to 70 °C for 2 min. 3' repair and 5' kinasing were done by

incubating the samples with 20% (vol/vol) sample, 1x T4 Polynucleotide Kinase Buffer (B0201S, NEB), 0.7 U/µl T4 Polynucleotide Kinase (M0201S, NEB), 0.17 U/µl RNA 5′ Pyrophospohydrolase (M03563, NEB), 0.03% Tween, and 0.2 U/µl SUPERase-In™ and RNasin® Plus each at 37 °C for 1 h 40 min, then adding 1x T4 Polynucleotide Kinase Buffer, 0.7 U/µl T4 Polynucleotide Kinase, 2.3 mM ATP, and 0.03% Tween and incubating for an additional 45 min. Next, EDTA was added (final conc. 0.01%) and the samples were heated to 70 °C for 2 min, after which RNA capture with streptavidin beads and the following RNA extraction were repeated as above.

Library preparation was done using the NEBNext Small RNA Library Prep Set for Illumina kit (E7330, NEB) according to the manufacturer's instructions except using 1:5 diluted 3′ SR Adaptor, SR RT Primer and 5′ SR Adaptor. Amplification was done as described in the kit except using only 18.5 µl of sample and adjusting the volume of water accordingly and using 18 cycles for the amplification PCR program as was determined to be optimal for the samples (Supplementary Method 4, Fig. S3). For index primers, NEBNext Multiplex Oligos for Illumina Sets 1 and 2 (E7335 and E7500, NEB) were used. After PCR amplification, the samples were purified using the Monarch PCR & DNA Kit (T1030, NEB) with the 7:1 ratio of sample:binding buffer according to the manufacturer's instructions except eluting to 10 µl. Quality controls were run on the DNA 1000 chip (5067-1504, Agilent) using the 2100 Bioanalyzer (Agilent) according to the manufacturer's instructions before and after size selection to determine concentration and purity of the samples. Due to poor quality, siNONEtOH rep2 and siNONdexKLA rep2 samples were reamplified from cDNA with 19 and 21 cycles, respectively, and siNONEtOH was combined with the existing sample and siNONdexKLA was not. To remove the 128-bp adaptor dimers, left side size selection was done using the determined optimal 0.9x ratio of SPRIselect beads (B23317, Beckman Coulter) according to the manufacturer's instructions except incubating samples for 10 min instead of 1 min. The samples were combined to a pooled library in equimolar ratios (~21-26 ng/library) and sequenced twice to a depth of $400x10^6$ bases (~$33.3x10^6$/library) with a 75-bp read length at EMBL GeneCore on the NextSeq 500, and the reads were combined.

## Data analysis

For citations to all used software, R packages, and other resources, see Supplementary Table 1. Raw sequenced reads were trimmed with Trimmomatic using single-end mode, and Phred 33 quality scores to remove reads below 20 bp and bases from the beginnings and ends of reads if their quality was below 3. Adaptor sequences (AGATCGGAAGAG) were trimmed from the 3′ ends

of reads using the HOMER command trim with maximum allowed mismatches in adaptor sequence set to 1 and removing any sequences shorter than 20 bp after trimming. To decontaminate the reads of common background sequences, they were mapped against the iGenomes abundant sequences hg38 library (Illumina) using bwa mem with the default options. The remaining reads were mapped to the hg38 genome with bwa mem with default options. Raw reads were counted at the whole gene level from only the + strand from all replicates with the analyzeRepeats.pl command from HOMER using default RefSeq annotations for the hg38 genome. Because the samples had high levels of PCR duplicates ranging between 48-68% (Fig. S5), only one identical read per position was allowed to be counted, thus removing duplicates. Differential gene expression analysis was done from raw read counts with the getDiffExpression.pl command from HOMER using the default DESeq2 calculation program and performing all possible comparisons between treatments. The outputs from this and raw read counting were combined with addDataHeader.pl from HOMER. Enhancer transcription was detected using the dREG cloud computing service from merged technical (sequencing) and biological replicates aligned with the proseq2.0.bsh command using options for single-end counting for 5' sequenced reads and combined with the mergeBigWig.bsh command. The outputs were filtered to false discovery rate (FDR) (Benjamini-Hochberg) < 0.05, and blacklisted regions (Supplementary Table 1) were removed using intersectBed from bedtools. A consensus file with all enhancers detected in at least one dataset was created using bedops with a range of 100 bp to increase peaks by 100 bp in both directions. From this, promoter regions were excluded by removing reads overlapping with TSSs, which were found with annotatePeaks.pl from HOMER for hg38, leaving only true enhancers. Raw and normalized read counting for enhancers was done the same way as for the genes except counting reads from both strands, and the outputs were combined.

For details of data analysis steps in R, see Appendix 2. Supplementary Code. Genes were defined to be transcribed if the normalized reads (transcript per million, TPM) exceeded the max median of TPMs and differentially transcribed if they filled the cutoff criteria of FDR < 0.01 and $\log_2$ fold change (FC) >0.5 or <–0.5. For enhancers, no TPM or log FC cutoff was used, and the FDR cutoff was set at 0.05. Heatmaps were drawn using the ComplexHeatmap package, network maps using ggnetwork and network packages, Venn diagrams using the ggVennDiagram package, and all other plots using the ggplot2 package. Known motif analysis was done from up- and downregulated enhancers separately using the findMotifsGenome.pl command from HOMER with the size of the sequences given and using all transcribed enhancers as background. Of the

resulting motifs, similar ones were merged together to reduce redundancy using the universalmotif package and then the top 5 most significant motifs from each comparison were kept. Pathway analysis was done using the Metascape online tool with up- and downregulated genes and enhancers separately and the top 5 most significant pathways from each comparison were kept.

## Results

### OsTIR1$^{F74G}$ was not integrated to the *AAVS1* locus

The first step to establishing a functional AID2 system is the stable expression of OsTIR1$^{F74G}$ (Yesbolatova *et al*, 2020). To achieve this, the insertion of OsTIR1$^{F74G}$ and a puromycin resistance marker to the *AAVS1* locus of A549 cells was attempted using CRISPR/Cas9 (Fig. 3A). To generate single-cell colonies, transfected cells were diluted to achieve single-cell resolution, and cells were allowed to proliferate to under antibiotic selection initially with 1.5 µg/ml puromycin, which had been determined suitable in a higher density cell culture. When no cells appeared to be surviving this selection, the concentration was reduced to 0.5 µg/ml, which killed all cells in a puromycin kill curve test in a comparably low-density cell culture (data not shown), and single-cell colonies were successfully obtained.

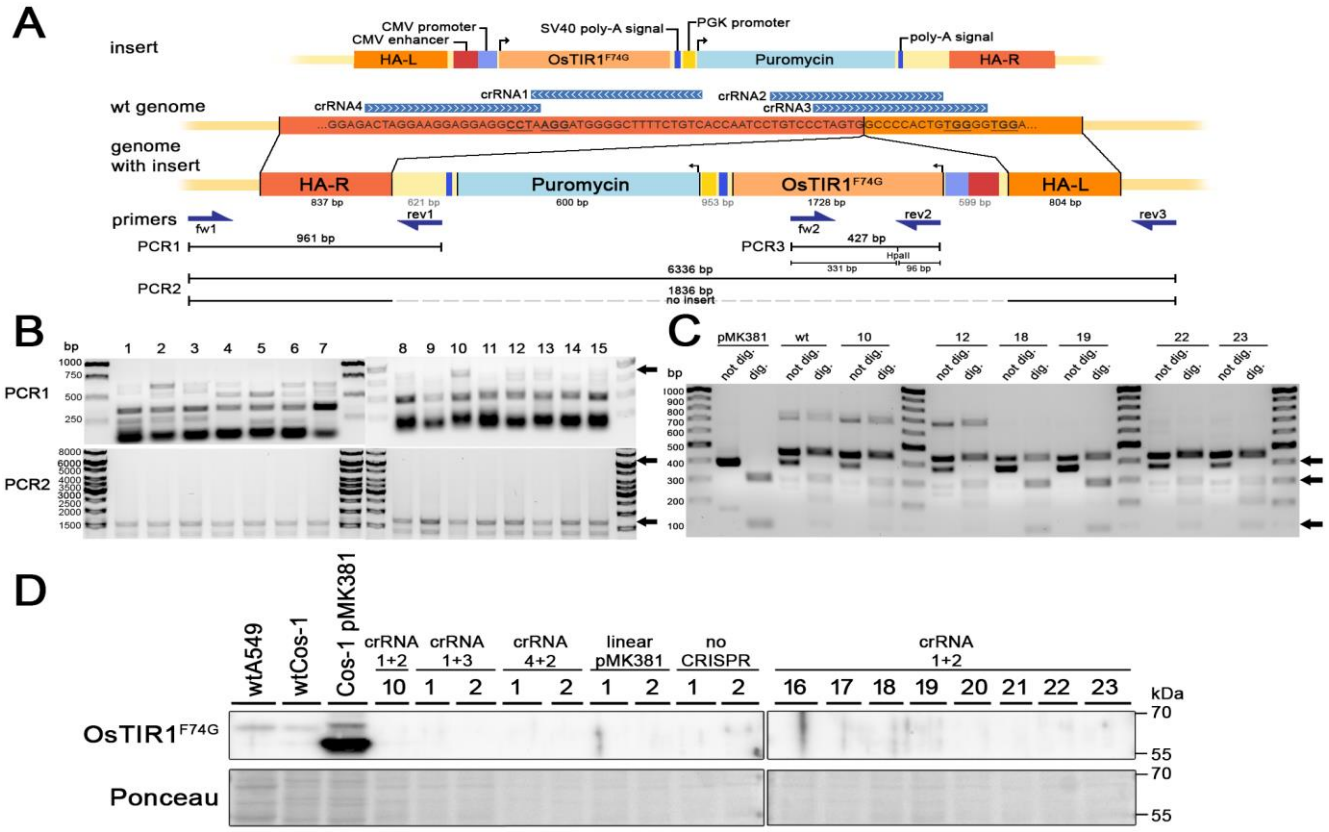



**Figure 3. OsTIR1$^{F74G}$ is not integrated into the *AAVS1* locus or expressed in A549 cells. A)** Schematic

representation of the insert with OsTIR1$^{F74G}$ and puromycin resistance flanked by the left (L) and right (R) homology arms (HA) and expressed from separate promoters with separate poly-A signals, the wt genome with the crRNA complementary sequences, and the genome with the insert as well as PCR primer binding sites and expected product sizes for screening. The arrows on crRNAs indicate if the crRNA binds to the sense (left) or antisense (right) strand. The PAM sequences for each crRNA are underlined. **B)** Screening of single-cell colonies with PCR1 (top) and PCR2 (bottom) revealed that no integration has occurred as no clear expected products were observed. 15 representative colonies generated with crRNA 1+2 are shown. **C)** Screening with PCR3 was inconclusive because the wt (negative control) produced the same size undigested (not dig.) fragment and digested fragments (dig.) as the positive control (pMK381). 6 representative single-cell colonies generated with crRNA 1+2 are shown. Arrows indicate expected product sizes. **D)** Western blotting with an anti-OsTIR1 antibody revealed that none of the representative single-cell colonies shown express OsTIR1$^{F74G}$ (64 kDa) and therefore no integration had occurred. Ponceau stain indicates protein loading which is approximately equal between samples. For uncropped versions, see Fig. S6-8.

After necessary optimization of the genomic PCR reactions as the QuickExtract™ lysis reagent used to harvest single-cell colonies was found to interfere with polymerase activity and alter the annealing temperatures of primers (Fig. S2), a total of 237 colonies were screened with PCR1 and PCR2 to determine OsTIR1$^{F74G}$ integration status (Fig. 3A). For PCR2, none of these colonies displayed the expected 6336-bp product indicating integration and, conversely, all displayed the 1836-bp product resulting from the wt genome, suggesting that no integration had occurred (Fig. 3B). As the longer 6336-bp product was not observed in any colonies screened and it is a rather long amplicon, it is possible its absence could be explained by failure of the PCR rather than the absence of the insert as it is a demanding PCR and QuickExtract™ greatly altered the reaction conditions. However, for PCR1, the only potential 961-bp products indicating integration were observed for colony number 10 and more faintly for colonies 8, 12, 13, and 15 of the representative colonies, while no other colonies displayed this product (Fig. 3B). Combining the results of these two screens, integration to the *AAVS1* locus was not likely.

To confirm that OsTIR1$^{F74G}$ had not integrated somewhere other than the expected locus or into the locus in an unexpected way, a sample of colonies, including colonies 10 and 12, was screened with PCR3 to amplify a sequence within the insert regardless of where or how it has integrated (Fig. 3A). The PCR products were digested with HpaII to analyze their sequence. Interestingly, the wt genome produced an unknown 427-bp fragment similarly to the positive control of pMK381 and, moreover, this fragment was digested identically to the positive control, producing 331-bp and 96-bp fragments (Fig. 3C). This made it difficult if not impossible to interpret these results reliably as a positive and negative result could not be distinguished from each other. The identity

of this unknown fragment from the wt genome could possibly be elucidated by sequencing in future experiments.

Since the results of PCR3 were inconclusive, the possibility of OsTIR1$^{F74G}$ integration in an unexpected manner not detectable by the PCRs still existed. To investigate if some colonies expressed OsTIR1$^{F74G}$, a total of 17 colonies from different transfections were screened by Western blotting with an anti-OsTIR1 antibody. At this point no positive control for the antibody existed and no A549 cells were in upkeep, so the closest alternative Cos-1 cells were used to generate a negative control and a positive control by transfection of pMK381. OsTIR1$^{F74G}$ was successfully detected in the positive control with the expected size of 64 kDa (Fig. 3D). Conversely, OsTIR1$^{F74G}$ was not detected in any of the colonies screened (Fig. 3D), indicating that it was not expressed by these colonies and thus most likely not integrated into their genome. In conclusion, these results showed that unfortunately the integration of OsTIR1$^{F74G}$ to the *AAVS1* locus was not successful, at least in the colonies screened. Nearly all the colonies screened happened to be generated with crRNAs 1+2 because those colonies were generated and picked first. Over 500 colonies generated using other crRNA pairs or the linearized repair template had been picked and remained unscreened but due to time constraints this was not pursued.

## p300 knockdown altered the expression of genes

Because the setup of the AID2 system for p300 knockdown at the protein level was unsuccessful, p300 knockdown was done at the mRNA level using the traditional method of siRNA. After transfection of siEP300 or siNON, the cells were treated with dex, KLA, both (dexKLA), or control (EtOH) for 1 h, nuclei were extracted, and PRO-seq was done. Comparisons were done for treatment effect within siNON (siNON vs siNON) and within siEP300 (siEP300 vs siEP300) conditions by comparing dex, KLA, and dexKLA to EtOH and KLA and dex to dexKLA, and for siRNA effect within all four treatments (siNON vs siEP300). A total of 1246 genes were differentially transcribed (DT) across all 14 comparisons (Fig. 4A, S9). The two replicates agreed reasonably well with each other as they formed clear clusters both in the heatmap (Fig. 4A) and the principal component analysis (PCA) (Fig. 4B), suggesting that there was no notable batch effect. Surprisingly, KLA had very little effect on the A549 cells as evidenced by there being very few or no DT genes in comparisons modeling the KLA effect (Fig. 4A, S9) and by KLA clustering with EtOH in the PCA (Fig. 4B). The explanation for this is most likely that TLR4 is barely expressed (Fig. S10) and thus the cells are not responsive to KLA. Interestingly, one of the 3 upregulated genes in siNONEtOH vs

siNONKLA, *CCL2*, was downregulated by dex (Fig. S11A), which would be expected to be seen more if the cells were responsive to KLA.

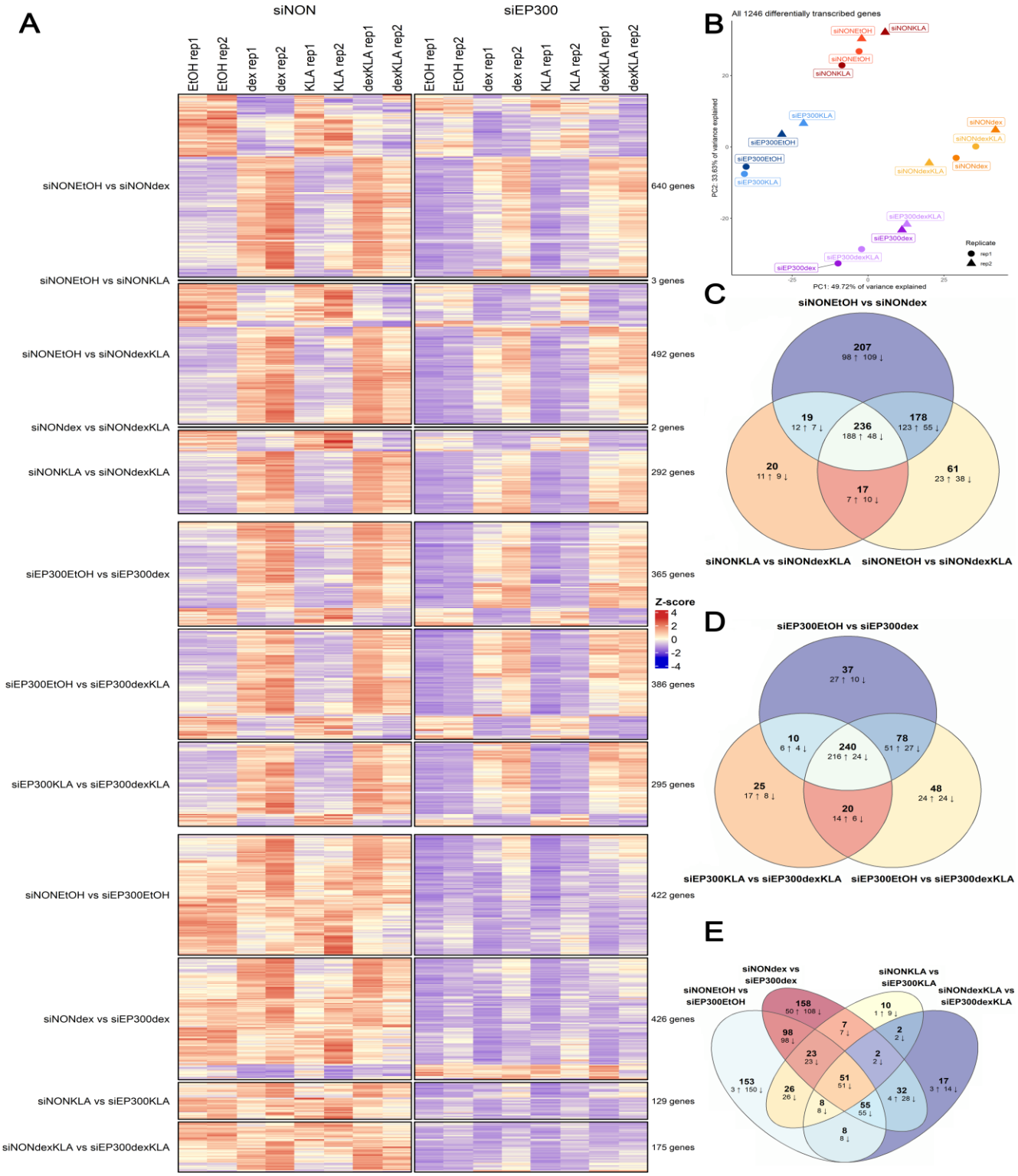



**Figure 4. Gene expression is repressed by p300 knockdown. A)** The heatmap with normalized TPMs of all DT genes for each comparison shows that in siNON vs siNON and siEP300 vs siEP300 comparisons dex upregulates a large set and downregulates a small set of genes, and that in the siNON vs siEP300 comparisons the DT genes are mainly downregulated, indicating that p300 knockdown impairs gene expression. The DT genes are less upregulated (lighter red) and more downregulated (darker blue) in the absence of p300 than in the presence of it. **B)** PCA plot of all 8 treatments from 2 replicates (rep) using all

DT genes shows that KLA had little effect on A549 cells because it clusters together with EtOH and similarly dexKLA clusters with dex, but dex and siEP300 had a clear effect as they form their own clusters. **C-E)** Venn diagrams of siNON vs siNON (C), siEP300 vs siEP300 (D), and siNON vs siEP300 (E) comparisons illustrate the overlap DT genes in different treatments and indicate that the dex effect is very similar between comparisons as many DT genes are shared. The bolded numbers indicate total counts and the smaller numbers indicate how many of these are up- (↑) and downregulated (↓).

Conversely, dex had a notable effect as it upregulated a majority of genes in the siNON vs siNON and siEP300 vs siEP300 comparisons modeling the dex effect (Fig. 4A) and of these a majority (67-93% and 87-91%, respectively) were shared between two or all three of the comparisons for dex effect (Fig. 4C, 4D). Furthermore, most of these genes, but not all of them, were also shared between the siNON vs siNON and siEP300 vs siEP300 comparisons (Fig. S11B, S11D, S11E). In the siEP300 vs siEP300 comparisons, dex was still able to upregulate a set of genes, whereas no DT genes in these comparisons would be expected if transcription was completely abrogated by p300 knockdown, though the upregulation of genes is less drastic and vice versa the downregulation is greater than in the siNON treatments (Fig. 4A). These results indicate that p300 knockdown somewhat altered the dex response while also largely maintaining it. On the other hand, in the siNON vs siEP300 comparisons, a significant portion of genes are downregulated compared to siNON already in the EtOH vs EtOH comparison (Fig. 4A) and over half of these genes were shared with one or more of the other siNON vs siEP300 comparisons (Fig. 4E) but not between the other siRNA comparisons to the same degree (Fig. S11B-D). In summary, these results indicate that p300 knockdown impaired gene expression but did not completely abolish it.

## Transcription of enhancers was largely unchanged by p300 knockdown

From the PRO-seq data, enhancers were detected using the dREG online tool, any regions overlapping with TSSs were removed, and the enhancers were associated to the nearest gene. A total of 663 enhancers were DT (Fig. 5A, S12) and of these the upregulated enhancers were enriched for GRE and GR half-site motifs (Fig. S13). Similarly to the DT genes, KLA had very little effect on the A549 cells as evidenced by the lack of DT enhancers for comparisons modeling the KLA effect for both the siNON and the siEP300 conditions (Fig. 5A, S15A). Moreover, the DT enhancers showed a similar response to dex as most enhancers were upregulated in the siNON vs siNON and siEP300 vs siEP300 comparisons and shared within these comparisons (Fig. 5A-C). Many DT enhancers were also shared between these two comparisons (Fig. S15B-D), suggesting that p300 knockdown alters the expression of some enhancers but not others. However, conversely to the DT genes, very few enhancers were DT despite the absence of p300 in the siNON

vs siEP300 comparisons (Fig. 5A, 5D) Moreover, a major downregulation of enhancers in these comparisons could not be observed like it was observed for DT genes and, instead, a larger fraction of enhancers was upregulated than downregulated in the siNONdex vs siEP300dex comparison (Fig. 5A, 5D). These results indicated that p300 knockdown did not affect which enhancers were actively transcribed, completely opposite to DT genes. This could hint at a compensatory mechanism to replace p300 activity in its absence, possibly by CBP, though CBP was not transcriptionally induced in p300 knockdown compared to siNON (Fig. S16). In conclusion, the observed lack of response to siEP300 by enhancers could be explained by redundancy between p300 and CBP or possibly some other coregulator.

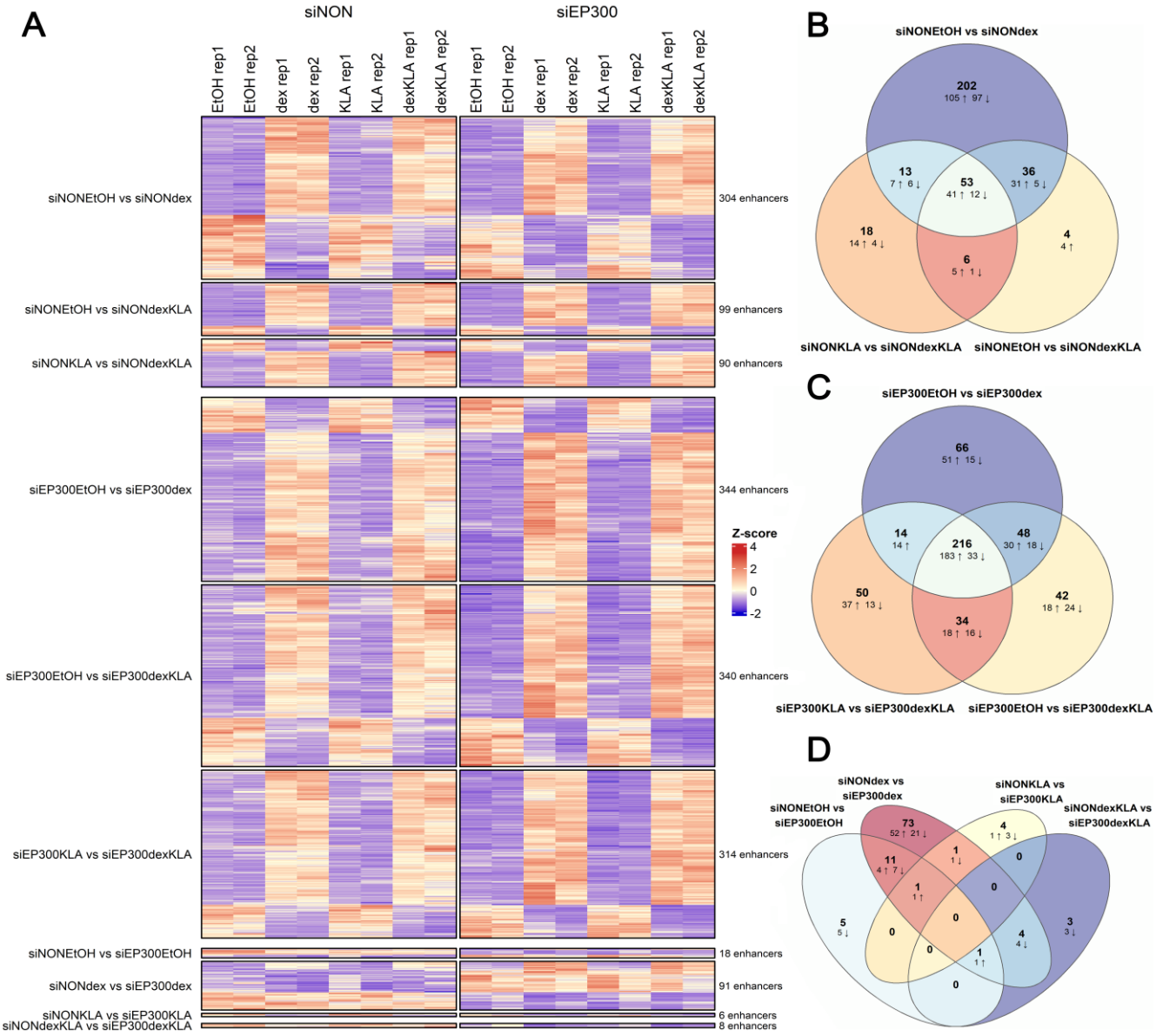



**Figure 5. Enhancer transcription is maintained in the absence of p300. A)** The heatmap with normalized TPMs of all DT enhancers for each comparison shows a similar pattern of upregulation by dex in siNON vs siNON and siEP300 vs siEP300 comparisons whereas very few enhancers are DT in the siNON vs siEP300 comparisons. Both up- and downregulation of enhancers is slightly more pronounced (darker red, darker blue) in the absence of p300 than in its presence. **B-D)** Venn diagrams of siNON vs siNON (B), siEP300 vs

siEP300 (C), and siNON vs siEP300 (D) comparisons indicate that, similarly to DT genes, a large portion are shared in the siNON vs siNON and siEP300 vs siEP300 comparisons, while for the siNON vs siE300 comparisons the same is not seen due to the low number of DT enhancers. The bolded numbers indicate total counts and the smaller numbers indicate how many of these are up- (↑) and downregulated (↓).

## Some genes lose upregulation in p300 knockdown but some do not

To further investigate the connections of DT genes between treatments and siRNA conditions, network maps were drawn for genes and genes associated with DT enhancers (Fig. 6). The DT genes (Fig. 6A) and genes associated with DT enhancers (Fig. 6C) within the siNON and siEP300 conditions formed two clusters, one for upregulated genes and one for downregulated genes with many shared genes within these clusters but none between them. In fact, two thirds of the DT genes were shared between any two comparisons (Fig. 6B) and a gene was DT at most in 10 comparisons (Fig. S17), indicating high connectivity of the DT genes, whereas, overall, the enhancers were less connected than DT genes as nearly half of them were unique to a comparison rather than shared (Fig. 6D) and the maximally shared DT enhancer found in 8 comparisons (Fig. S18). The genes downregulated by siEP300 landed between these two clusters and made connections to both, thus linking the two completely separated clusters (Fig. 6A), while due to the low number of DT enhancers in the siNON vs siEP300 comparisons, they made very few connections to the up and downregulated clusters (Fig. 6C), further confirming that the expression of enhancers was not greatly altered by siEP300. This observed linkage of the up- and downregulated clusters by the siNON vs siEP300 cluster is expected because of the massive downregulation of genes observed in the siNON vs siEP300 comparisons (Fig. 4A) and it further confirms that the expression of these genes was impaired by p300 knockdown. However, not all upregulated genes in the siNON vs siNON and siEP300 vs siEP300 comparisons are downregulated in the siNON vs siEP300 comparisons, suggesting that some genes are spared from p300 knockdown-mediated transcriptional repression. Moreover, interestingly a small set of genes, including coactivators *SERTAD3* and *BCL3* (GeneCards – the human gene database), in the siNON vs siEP300 comparisons gain upregulation in the absence of p300 while they are normally downregulated (Fig. S11B, S11D), which could result from the attempt by the cells to adapt to the p300 knockdown. In addition, a large set of genes are uniquely downregulated in the siNON vs siEP300 comparison, implying that their downregulation could be somewhat stochastic. To summarize, a portion of genes are downregulated by p300 knockdown while others evade this effect, indicating that this response may be gene-specific.

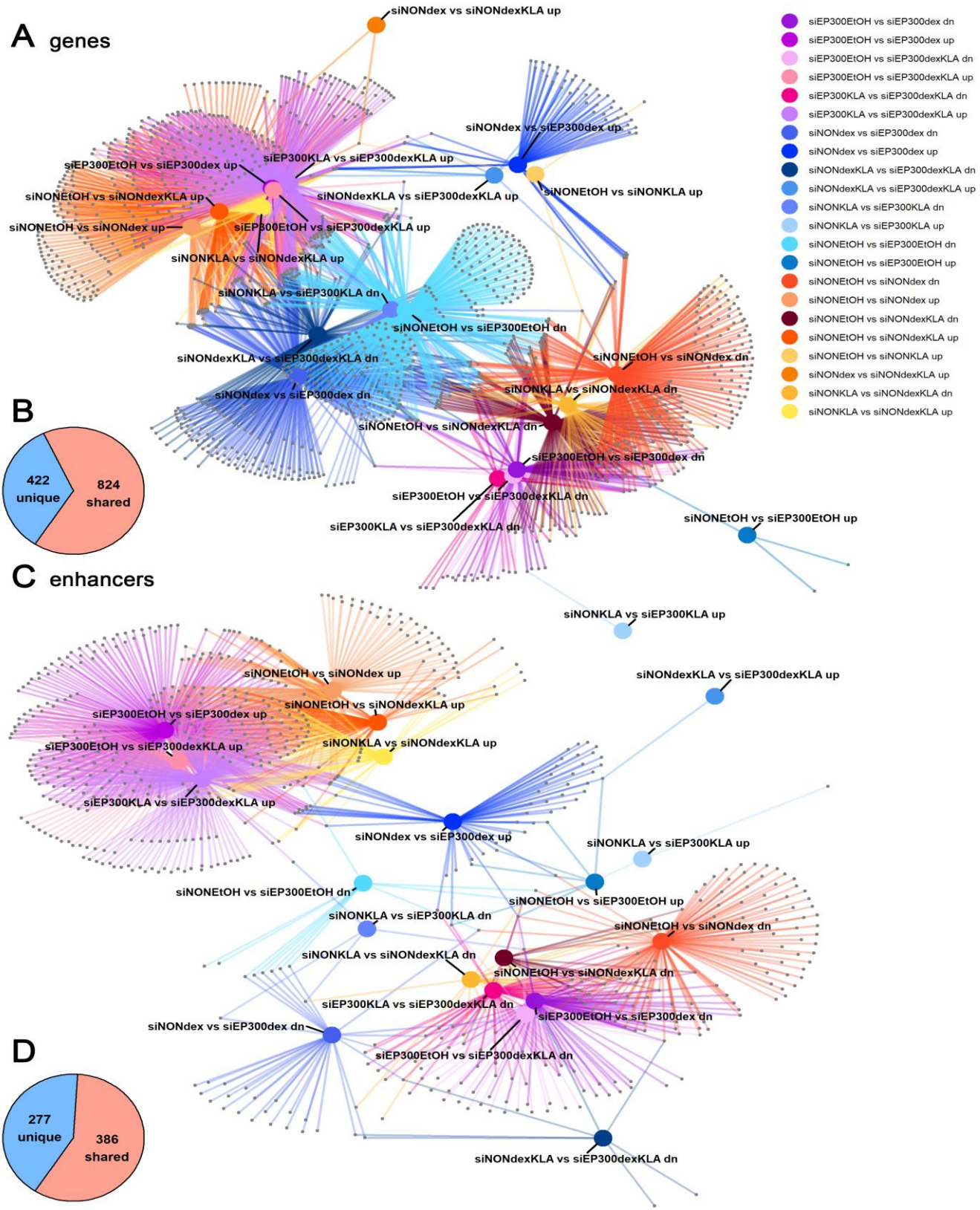



**Figure 6. Some genes lose upregulation in p300 knockdown, but enhancers do not. A and C)** Network maps of DT genes (A) and genes associated with DT enhancers (C) show how upregulated and downregulated genes and enhancers form their own clusters which are, more or less, connected by the downregulated genes and enhancers in the siNON vs siEP300 comparisons. The grey dots represent DT genes and genes associated with DT enhancers. **B and D)** Pie charts illustrate that 34% and 42% of all DT genes (B) and genes associated with DT enhancers (D), respectively, are unique to one comparison, indicating that enhancers are less shared between comparisons.

## DT enhancers were not associated with the DT genes

Interested by the observed completely opposite patterns of response to siEP300 by genes and enhancers, I set out to elucidate to what degree the DT genes and enhancers were shared with each other. Overall, the up and downregulated genes and enhancers formed four separate clusters which all had interactions between one another, the most connections being between the up- and downregulated genes, but these interactions were scarce and most of the genes or enhancers were uniquely up- or downregulated (Fig. 7, S19A). The DT enhancers agreed well with each other on whether the enhancer was up- or downregulated as only 6 (1%) were differentially regulated, compared to the genes with nearly 150 (12%) differentially regulated genes (Fig. S19B), which is expected based on the observation that the up- and downregulated clusters of enhancers had few interactions. Of the total 1246 and 663 genes and enhancers, respectively, only 49 total were shared between any of the comparisons (Fig. S19B). Though, interestingly, more than half (67%) of these 49 shared genes and enhancers were differentially regulated, *i.e.* upregulated as genes and downregulated as enhancers or vice versa (Fig. S19B), which together with the observed disconnect contradicts the notion that upregulated enhancers typically correlate with upregulated genes (Greulich *et al*, 2021).

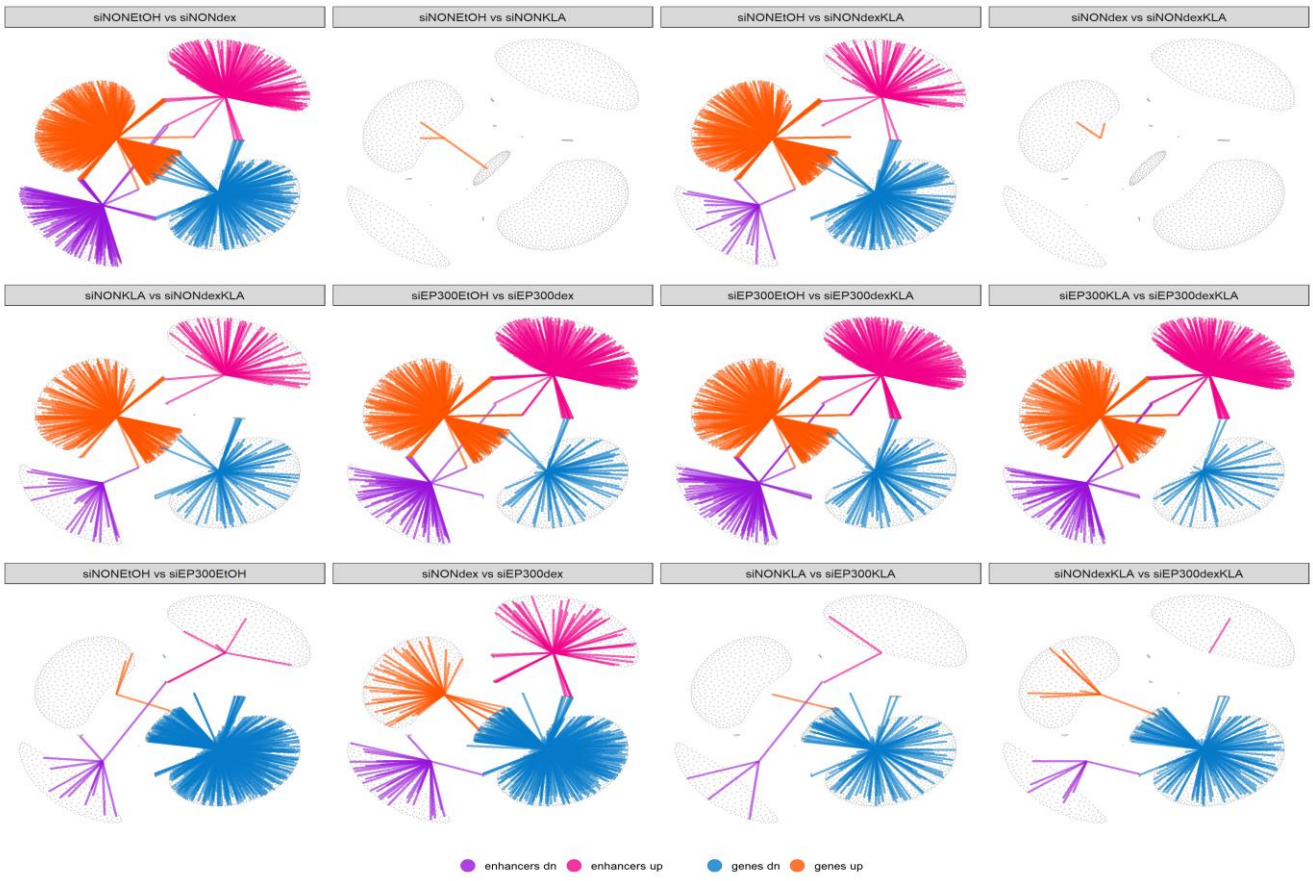



**Figure 7. DT enhancers were not associated with the DT genes.** Network maps of all DT genes and genes

associated with DT enhancers in each comparison reflect previously observed patterns of mostly upregulated genes in siNON vs siNON and siEP300 vs siEP300 comparisons and downregulated genes and few DT enhancers in siNON vs siEP300 comparisons. Very few connections between the enhancers and genes can be observed. Even though some genes or enhancers appear to be simultaneously up- and downregulated in a comparison, they are actually connected to different genes or enhancers. The grey dots represent DT genes and genes associated with DT enhancers.

## Activity of certain pathways is maintained at the cost of others in p300 knockdown

To investigate the broader processes affected by the dex and siEP300 treatments, pathway analysis was done for significantly up- and downregulated genes and genes associated with enhancers using Metascape, which includes pathways from multiple databases, and the top 5 most significant pathways from each comparison were reported (Supplementary Table 3). A total of 70 pathways, of which 62 and 37, were enriched to DT genes and enhancers, respectively. Of these, 29 pathways (40%) were shared between genes and enhancers, which is notably more than the 10% of shared DT genes or genes associated with DT enhancers (Fig. S13B), suggesting that while the genes regulated may have been different, they were ultimately involved in the same processes. Unsurprisingly, the glucocorticoid receptor pathway and nuclear receptors meta-pathway were very significantly enriched to both upregulated and downregulated genes but surprisingly not to enhancers in the comparisons modeling dex effect (Fig. 9). The downregulation was because these pathways include downregulated genes like *NR3C1*, *JUN* coding for c-Jun, *NFKB2* coding for p52, *CCL2*, and *PTGS2* coding for cyclo-oxygenase 2. Furthermore, pathways related to inflammation such as TNF, cytokine, interleukin, and NIK/NF-κB signalling were also enriched to DT genes and enhancers in siNON vs siNON and siEP300 vs siEP300 but not in siNON vs siEP300 comparisons (Fig. 9), suggesting that, overall, the expression of these pathways was not impaired by siEP300. On the other hand, multiple pathways related to cell adhesion, junctions, and morphology were upregulated in the siNON vs siNON and siEP300 vs siEP300 comparisons but downregulated in the siNON vs siEP300 comparisons (Fig. 9), which could indicate that the activity of these pathways was sacrificed in order to support the activity of other pathways. The observation that various pathways were differentially regulated even in the siNONEtOH vs siEP300EtOH comparison is not surprising as p300 is known to interact with more than 400 partners and thus participate in the regulation of pleiotropic processes (Martire *et al*, 2020). Moreover, in line with the earlier observation that most DT enhancers were upregulated, most of the pathways enriched to genes associated with DT enhancers were upregulated, such as ones related to cell death and apoptosis, various virus infections, hormone response, and pathways in

cancer, and much fewer pathways were downregulated for enhancers than for genes (Fig. 9). These pathways, though statistically significant (FDR < 0.05), were less significant than most pathways enriched to DT genes (Fig. 9), which may suggest that the DT enhancers did not regulate the nearest genes and thus, with an essentially random set of genes, pathway enrichment was not as significant. Taken together, these results suggest that some crucial pathways avoid downregulation in the absence of p300, hinting towards a compensatory mechanism.


**Figure 9. Some pathways for DT genes escape repression by p300 knockdown.** The x-axis represents the $\log_{10}$ FDRs so that upregulated pathways have positive values for the purposes of plotting but, in reality, the values are negative like for the downregulated pathways, so the most significant FDRs are at the outer edges of the plots. It is important to note that the FDR values are in completely different ranges for the genes and enhancers. The overlap (%) describes the percentage of how many genes in the pathway were found from the input list of DT genes or genes associated with DT enhancers.

## Discussion

In this study, I investigated the role of the coregulator p300 in the crosstalk between the anti-inflammatory GR and proinflammatory NF-κB in A549 lung adenocarcinoma cells because coregulator squelching has been suggested to be a mechanism of GR-mediated repression of inflammation (Schmidt *et al*, 2016). At first this was pursued through the rapid AID2-mediated

p300 knockdown at the protein level but unfortunately the setup of the AID2 system was unsuccessful and thus I resorted to siRNA-mediated p300 knockdown and performed PRO-seq to elucidate which genes actively respond to treatment with dex, KLA, or both. It was found that p300 knockdown attenuated gene expression while enhancer activity remained largely unaffected, but overall, on the level of pathways, multiple important functions were maintained despite p300 knockdown. In addition, it was observed that the genes associated with DT enhancers were not the same as the DT genes.

In the PRO-seq experiment, unfortunately and surprisingly, the KLA treatment elicited a meager response in the A549 cells, and this limited the investigation of the role of p300 to only GR activity instead of GR–NF-κB crosstalk. The most likely, if not definite, explanation for this could be that the A549 cells do not express TLR4 required for the detection of KLA as, indeed, PRO-seq showed only minute transcription of TLR4 across all samples. This was unexpected because A549 have been shown to express TLR4 and to be responsive to LPS (Yang & Ma, 2022), though surprisingly no studies report using KLA in A549 cells. Thus, it appears that the cells have lost the expression of TLR4 during passaging for an unknown reason, which could be further confirmed by *e.g.* Western blot. Though, the lack of response could also be attributed to the nature of PRO-seq only detecting nascent transcripts, making the time of measuring crucial as it is possible there was no target gene expression at the moment of measuring. Furthermore, Cheng *et al*. report an interesting finding regarding NF-κB dynamics that different activating stimuli cause differing temporal patterns of NF-κB nuclear localization (Cheng *et al*, 2021). For example, when stimulated with TNF, the nuclear localization of NF-κB oscillates so that at 1 h post-treatment it is not localized in the nucleus (Cheng *et al*, 2021). KLA could have a similar effect and thus no expression is detected at the 1-hour time point. To verify if this is the case, mRNA production from a wider time scale could be assessed with a bulk RNA method like RNA sequencing.

The observation that p300 impaired gene but not enhancer expression is interesting because it implies that p300 is simultaneously essential and nonessential. p300 and CBP are often considered to be functionally identical (Martire *et al*, 2020). However, a growing body of evidence suggests that p300/CBP have both redundant and nonredundant actions depending on the context. For example, mice homozygous for CBP or p300 knockout or double heterozygotes for p300/CBP knockout are embryonic lethal at approximately day 10 but they do survive up to that point (Kasper *et al*, 2006), which demonstrates that some but not all functions of p300 and CBP

can be replaced by the other. Many studies also report finding differing effects of p300 and CBP in various contexts such as T cell development *in vivo* in mice and in different cell types including between T cells and macrophages (Kasper *et al*, 2006) or between astrocytes and Schwann cells (Fonte *et al*, 2007). Furthermore, knockdown of p300 and CBP was reported to regulate distinct gene sets and pathways from each other in mouse embryonic stem cells (Martire *et al*, 2020). Therefore, it seems likely that CBP could compensate for p300 activity at enhancers and at some genes but not all of them. Though, it was observed that the transcription of CBP was not induced in response to p300 knockdown. However, a large portion of DT genes were uniquely downregulated in the absence of p300. Together these observations suggest a model where CBP is still present in a limiting amount and therefore can only assist in maintaining the expression of some genes in the absence of p300 while the genes it normally upregulates are downregulated as they lose CBP in a stochastic manner.

Indeed, it has been found that GR can repress genes by attenuating p300 recruitment by competing for the same tethering site on p65 with p300 and thus attenuating H3K27 acetylation which ultimately prevents *de novo* Pol II recruitment (Sacta *et al*, 2018). Moreover, Martire *et al.* determined p300 to be the major depositor of H3K27ac out of p300/CBP in mouse embryonic stem cells (Martire *et al*, 2020). Therefore, together these two findings could explain why gene expression was attenuated by p300 knockdown but not fully compensated for by CBP due to insufficient H3K27ac deposition required for Pol II recruitment. This hypothesis is further corroborated by the finding that p300 is required for the induction of *IL6* in MCF-7 cells (Hudson *et al*, 2018). Conversely, the heritable congenital developmental disorder Rubinstein-Taybi syndrome characterized by growth impairment and mental retardation and caused by mutations in p300 or CBP is milder when p300 is mutated instead of CBP, which suggests that p300 could be less crucial than CBP in some cell or tissue types (Bedford & Brindle, 2012; Borczyk *et al*, 2021; Lonard *et al*, 2007). This is somewhat in line with the observation that p300 did not completely impair transcription as dex was still able to elicit a response in the p300 knockdown condition and, at the pathway level, crucial functions such as immune regulation and hormone response were maintained even in the absence of p300. It is possible that the knockdown of p300 with siRNA may not have been complete, and other studies have suggested that while p300/CBP are limiting, they are not extremely so, and thus a low amount could still ensure critical functions (Kasper *et al*, 2006). Interestingly, Borczyk *et al.* found that active enhancer transcription was followed by the recruitment of p300, not preceded by it (Borczyk *et al*, 2021), which may suggest that enhancers

could be less dependent on p300 for activation and thus not be impacted by p300 knockdown. Overall, based on previous observations of differing actions of p300/CBP in different cell types and the current observation that some genes escape repression in p300 knockdown, it seems likely that whether p300 knockdown impairs gene expression may be specific to the cell type and gene.

This study found that there was little agreement between the DT genes and genes associated with DT enhancers, which contradicts the rather well accepted notion that actively transcribed enhancers are associated with actively expressed genes (Greulich *et al*, 2021). This disconnect could be because the DT enhancers were associated with the nearest gene while enhancers can be hundreds of kilobases away from the TSSs they regulate and so can interact over interfering genes (Sanyal *et al*, 2012). Indeed, it has been shown that only a small portion of enhancers are associated with the nearest gene (Sanyal *et al*, 2012). Thus, more reliable association of enhancers to genes would be desired and could be achieved through *e.g.* chromatin conformation capture (3C)-based methods (Sanyal *et al*, 2012). Additionally, it has been reported that the transcription of enhancers could persist even after the upregulation of gene expression settled (Borczyk *et al*, 2021), while entirely conversely, Huang *et al.* find that the transcription of *Ccl2* was significantly delayed compared to the transcription of its enhancer (Huang *et al*, 2021). Together, due to the nature of PRO-seq being essentially a snapshot of the active transcription, these could also explain the observed disconnectedness of DT genes and enhancers as different genes and enhancers were being actively transcribed. Though, PRO-seq is still a bulk method and as such the enhancer activation dynamics would have to be rather synchronized in the cell population.

The major limitation of the PRO-seq experiment is, of course, as mentioned earlier, that due to the slowness of siRNA-mediated knockdown, it is impossible to determine whether the observed changes in gene and enhancer expression are the direct effects of p300 knockdown or if they are caused by adaptation to the loss of p300. This is what originally motivated the use of AID2 for p300 knockdown and the AID2 approach is still a valid option to pursue with some further optimization. The OsTIR1$^{F74G}$ integration was targeted to the *AAVS1* locus because it is considered to provide sufficiently stable expression compared to random integration which may be subject to silencing depending on the locus (Yunusova *et al*, 2021). However, Yunusova *et al.* tested both targeted integration of OsTIR1 into the *AAVS1* locus and random integration and found no significant difference in the expression levels or POI degradation efficiency between the two (Yunusova *et al*, 2021). Since, for OsTIR1$^{F74G}$ integration, the locus does not matter like it does for mAID tagging of

the POI, random integration is a viable option and therefore it was attempted by transfecting only pMK381. In fact, it is possible that a positive colony was achieved with this method, but those colonies have simply not been screened yet.

One possible reason for the failure of the expected genomic integration using CRISPR/Cas9 could be the mutation of the target site or the insert caused by the recutting of DNA by CRISPR/Cas9 if the PAM site and gRNA target sequence are still present after the integration (Adhikari *et al*, 2021). However, in this study this is not possible as the target sequences of crRNA2 and crRNA3 should be disrupted by the integration and the remaining intact crRNA1 and crRNA4 target sequences would only produce a single-strand break. Instead, the failure of the OsTIR1$^{F74G}$ integration is most likely the result of the lowered puromycin concentration for selection combined with a low transfection, CRISPR/Cas9, or HDR efficiency. HDR efficiency can vary depending on cell type, integration locus, and repair template (Ran *et al*, 2013). The efficiency of the of crRNAs can also be variable if, for example, there are common sequence variants or mutations in the genome at the target site, which is entirely possible since A549 is a cancer cell line. Adhikari *et al.* suggest screening the CRISPR/Cas9 target site for an insertion or deletion to test for the CRISPR/Cas9-induced cut if a successful clone is not obtained (Adhikari *et al*, 2021), so this could be explored to determine if new crRNA design needs to be considered.

A low efficiency of transfection, cutting, or repair leads to very few colonies with successful integration, making their selection from unsuccessful clones essential. Although initially the lower puromycin concentration is sufficient to kill non-resistant cells, as the cells proliferate it is likely that the concentration can no longer kill the most resilient of the non-resistant cells because the mechanism through which puromycin inhibits translation involves its irreversible incorporation to the C-terminus of the nascent polypeptide by mimicking tRNA, and thus the puromycin is consumed from the media and selection pressure is lifted if puromycin is not replenished frequently enough (Aviner, 2020). Thus, the non-resistant colonies can overpower the resistant colonies, and the chance of picking a resistant colony from many non-resistant colonies is low. In fact, the higher puromycin concentration may have been suitable from the beginning and it only appeared that all cells were dying because very few were resistant. The solution would be to do antibiotic selection at only the higher puromycin concentration or, if the higher concentration truly is too high even for the resistant cells to survive in a low-density population, to initiate selection at the lower concentration and then increase the concentration once the cells have proliferated.

To improve the integration of OsTIR1$^{F74G}$, some different strategies could be considered. A method especially useful for achieving homozygous mAID tagging but which could also be applied for OsTIR1$^{F74G}$ integration is dual-antibiotic selection. This is done by using two homologous repair templates with a different antibiotic marker in each so that the cell clone will survive only if it has biallelic insertion with one copy of each marker gene (Yesbolatova *et al*, 2019). Indeed, Natsume *et al.* report achieving a >70% homozygous mAID insertion at the correct locus for resistant clones using dual selection (Natsume *et al*, 2016). Here, it is important to consider the ploidy of the cell line because tagging all POI alleles with mAID for homozygosity enables the study of the true loss of the POI as compared to heterozygous tagging (Saito & Kanemaki, 2021). Though, for OsTIR1$^{F74G}$, homozygosity is not strictly necessary but preferable for possibly higher expression. A549 is a hypotriploid cell line, and an unbalanced rearrangement of the chromosome arm 19q containing the *AAVS1* locus to chromosome 15 has been found, making A549 triploid for the *AAVS1* locus. Fortunately, however, A549 is diploid for chromosome 22 where *EP300* coding for p300 is located (Luk *et al*, 2001; Peng *et al*, 2010).

Alternatively, to ensure the expression of OsTIR1$^{F74G}$, a template could be used where the puromycin resistance is connected to OsTIR1$^{F74G}$ with either a self-cleaving P2A sequence or an internal ribosome entry site (IRES). The IRES is an RNA-sequence which recruits the ribosome and initiates translation independent of the 5' cap while the P2A sequence is skipped by the ribosome during translation, both thus allowing the translation of two separate polypeptides from the same mRNA (Lee *et al*, 2019; Saito & Kanemaki, 2021). This way, only colonies expressing both OsTIR1$^{F74G}$ and puromycin would survive the selection. These sequences could also be used to attach a fluorescent protein to OsTIR1$^{F74G}$ to significantly accelerate the screening process through the identification of a positive colony simply by fluorescence microscopy without PCR screening (Saito & Kanemaki, 2021). Furthermore, Nishimura and Fukagawa present a promising protocol in which both components of the AID system, OsTIR1 and AID-tagged POI, are inserted in the middle of the endogenous POI to disrupt it using a single homologous repair template and thus a single CRISPR/Cas9 transfection instead of two (Nishimura & Fukagawa, 2021). This significantly simplifies and accelerates the setup process and saves resources as single-cell colonies need to be grown, picked, and screened only once. The downside is that the inserted POI will lack natural epigenetic modifications normally present on the endogenous gene, possibly affecting expression, and may have to be inserted as cDNA thus losing any regulation arising from the introns. Combined with dual-antibiotic selection to ensure homozygous integration crucial for eliminating

all functional copies of the endogenous gene and the use of P2A or IRES, this method could produce cells with successful integration expressing both OsTIR1$^{F74G}$ and the mAID-tagged POI.

Coregulators have been proposed to be promising new drug targets, especially if combined with existing drug treatments, even though they are considered generally difficult to target due to many lacking a specific binding pocket (Lonard & O'Malley, 2012). While this could provide alternatives to GCs in the future, targeting p300 with a drug is likely to have many adverse side effects because it interacts with over 400 partners, making it one of most heavily connected hubs in the mammalian protein-protein interactome and an important integrator of signals (Kasper *et al*, 2006; Lonard & O'Malley, 2012; Martire *et al*, 2020). However, over 350 coregulators have been reported in the literature, and among these there are likely to be some that are more specific to the GR-NF-κB crosstalk than p300 (Lonard & O'Malley, 2012). For possible future translation of these findings to the clinic, it is of course important to validate these results in other cell types and eventually *in vivo*, with different GR ligands, and with different time courses of administration in relation to inflammatory stimulus since GC responses are very cell-, ligand-, and administration time-specific (Cain & Cidlowski, 2017; Kadmiel & Cidlowski, 2013; Oh *et al*, 2017).

In conclusion, the data from this study suggest that CBP, or possibly some other coactivator, could replace p300 action at enhancers to some extent but not completely at the level of gene transcription, corroborating findings from previous studies that p300 is an important coregulator for GC responses. The level of redundancy between p300 and CBP in the context of GR–NF-κB crosstalk would have to be further elucidated in future studies, preferably by AID2-mediated rapid knockdown of either p300 or CBP or both. Despite the setbacks of the AID2 system setup forcing the use of siRNA and the lack of KLA response in the A549 cells, the data generated from the PRO-seq experiment are certainly usable as comparisons for future data from AID2-mediated p300 knockdown and, with the large amount of information that genome-wide sequencing methods generate, the current data could be explored much further. This study shed some light on the complex mechanisms of gene regulation and deepened the understanding of p300 function, revealing differential regulation of genes and enhancers which is an interesting phenomenon to be addressed in further studies.

## Acknowledgements

I thank Einari Niskanen for help with planning the project and guidance throughout it; Nihay Laham Karam for valuable advice on CRISPR/Cas9 and genomic PCR; Bharadwaja Velidendla for

help with PRO-seq sample preparation, especially for performing the Bioanalyzer quality checks and the size selections, and for general support and guidance in the laboratory; Merja Räsänen for guidance through the Western blot process and for providing technical assistance in the laboratory; Eija Korhonen for providing technical assistance and for providing the cultured Cos-1 cells; Kaiser Manjur for showing how to pick single-cell colonies; Minna Kaikkonen-Määttä for providing the PRO-seq protocol used and advice regarding it; and UEF Bioinformatics Center for providing computational infrastructure for data analysis.

## References

Adhikari B, Narain A, Wolf E (2021) Generation of auxin inducible degron (AID) knock-in cell lines for targeted protein degradation in mammalian cells. *STAR Protoc* 2: 100949

Atsaves V, Leventaki V, Rassidakis GZ, Claret FX (2019) AP-1 Transcription Factors as Regulators of Immune Responses in Cancer. *Cancers (Basel)* 11: 1037

Aviner R (2020) The science of puromycin: From studies of ribosome function to applications in biotechnology. *Comput Struct Biotechnol J* 18: 1074–1083

Bates LE, Alves MRP, Silva JCR (2021) Auxin-degron system identifies immediate mechanisms of OCT4. *Stem Cell Reports* 16: 1818–1831

Bedford DC, Brindle PK (2012) Is histone acetylation the most important physiological function for CBP and p300? *Aging (Albany NY)* 4: 247–255

Bekhbat M, Rowson SA, Neigh GN (2017) Checks and balances: The glucocorticoid receptor and NFκB in good times and bad. *Front Neuroendocrinol* 46: 15–31

Bence M, Jankovics F, Lukácsovich T, Erdélyi M (2017) Combining the auxin-inducible degradation system with CRISPR/Cas9-based genome editing for the conditional depletion of endogenous Drosophila melanogaster proteins. *FEBS J* 284: 1056–1069

Borczyk M, Zieba M, Korostyński M, Piechota M (2021) Role of Non-Coding Regulatory Elements in the Control of GR-Dependent Gene Expression. *Int J Mol Sci* 22: 4258

De Bosscher K, Berghe WV, Haegeman G (2001) Glucocorticoid Repression of AP-1 Is Not Mediated by Competition for Nuclear Coactivators. *Mol Endocrinol* 15: 219–227

De Bosscher K, Berghe WV, Vermeulen L, Plaisance S, Boone E, Haegeman G (2000) Glucocorticoids repress NF-κB-driven genes by disturbing the interaction of p65 with the basal transcription machinery, irrespective of coactivator levels in the cell. *Proc Natl Acad Sci U S A* 97: 3919–3924

Cain DW, Cidlowski JA (2017) Immune regulation by glucocorticoids. *Nat Rev Immunol* 17: 233–247

Cheng QJ, Ohta S, Sheu KM, Spreafico R, Adelaja A, Taylor B, Hoffmann A (2021) NFκB Dynamics Determine the Stimulus-Specificity of Epigenomic Reprogramming in Macrophages. *Science (80- )* 372: 1349–1353

Dendoncker K, Timmermans S, Vandewalle J, Eggermont M, Lempiäinen J, Paakinaho V, Van Hamme E, Dewaele S, Vandevyver S, Ballegeer M, *et al* (2019) TNF-α inhibits glucocorticoid receptor-induced gene expression by reshaping the GR nuclear cofactor profile. *Proc Natl Acad Sci* 116: 12942–12951

Escoter-Torres L, Caratti G, Mechtidou A, Tuckermann J, Uhlenhaut NH, Vettorazzi S (2019) Fighting the fire: Mechanisms of inflammatory gene regulation by the glucocorticoid receptor. *Front Immunol* 10: 1859

Escoter-Torres L, Greulich F, Quagliarini F, Wierer M, Uhlenhaut NH (2020) Anti-inflammatory functions of the glucocorticoid receptor require DNA binding. *Nucleic Acids Res* 48: 8393–8407

Fonte C, Trousson A, Grenier J, Schumacher M, Massaad C (2007) Opposite effects of CBP and p300 in glucocorticoid signaling in astrocytes. *J Steroid Biochem Mol Biol* 104: 220–227

Gazon H, Barbeau B, Mesnard JM, Peloponese JM (2017) Hijacking of the AP-1 Signaling Pathway during Development of ATL. *Front Microbiol* 8: 2686

Gillespie MA, Palii CG, Sanchez-Taltavull D, Shannon P, Longabaugh WJR, Downes DJ, Sivaraman K, Espinoza HM, Hughes JR, Price ND, *et al* (2020) Absolute Quantification of Transcription Factors Reveals Principles of Gene Regulation in Erythropoiesis. *Mol Cell* 78: 960-974.e11

Greulich F, Bielefeld KA, Scheundel R, Mechtidou A, Strickland B, Uhlenhaut NH (2021) Enhancer RNA Expression in Response to Glucocorticoid Treatment in Murine Macrophages. *Cells* 11: 28

Greulich F, Hemmer MC, Rollins DA, Rogatsky I, Uhlenhaut NH (2016) There goes the neighborhood: Assembly of transcriptional complexes during the regulation of metabolism and inflammation by the Glucocorticoid Receptor. *Steroids* 114: 7–15

Gupte R, Muse GW, Chinenov Y, Adelman K, Rogatsky I (2013) Glucocorticoid receptor represses proinflammatory genes at distinct steps of the transcription cycle. *Proc Natl Acad Sci U S A* 110: 14616–14621

Holland AJ, Fachinetti D, Han JS, Cleveland DW (2012) Inducible, reversible system for the rapid and complete degradation of proteins in mammalian cells. *Proc Natl Acad Sci U S A* 109: E3350–E3357

Hoppstädter J, Ammit AJ (2019) Role of dual-specificity phosphatase 1 in glucocorticoid-driven antiinflammatory responses. *Front Immunol* 10: 1446

Hu B, Huang S, Yin L (2021) The cytokine storm and COVID-19. *J Med Virol* 93: 250–256

Huang Z, Liang N, Goñi S, Damdimopoulos A, Wang C, Ballaire R, Jager J, Niskanen H, Han H, Jakobsson T, *et al* (2021) The corepressors GPS2 and SMRT control enhancer and silencer remodeling via eRNA transcription during inflammatory activation of macrophages. *Mol Cell* 81: 953-968.e9

Hudson WH, Vera IMSD, Nwachukwu JC, Weikum ER, Herbst AG, Yang Q, Bain DL, Nettles KW, Kojetin DJ, Ortlund EA (2018) Cryptic glucocorticoid receptor-binding sites pervade genomic NF-κB response elements. *Nat Commun* 9: 1337

Jiang F, Doudna JA (2017) CRISPR–Cas9 Structures and Mechanisms. *Annu Rev Biophys* 46: 505–529

Jin Q, Yu LR, Wang L, Zhang Z, Kasper LH, Lee JE, Wang C, Brindle PK, Dent SYR, Ge K (2011) Distinct roles of GCN5/PCAF-mediated H3K9ac and CBP/p300-mediated H3K18/27ac in nuclear receptor transactivation. *EMBO J* 30: 249–262

Johnson TA, Chereji RV, Stavreva DA, Morris SA, Hager GL, Clark DJ (2018) Conventional and pioneer modes of glucocorticoid receptor interaction with enhancer chromatin in vivo. *Nucleic Acids*

*Res* 46: 203–214

Kadmiel M, Cidlowski JA (2013) Glucocorticoid receptor signaling in health and disease. *Trends Pharmacol Sci* 34: 518–530

Kasper LH, Fukuyama T, Biesen MA, Boussouar F, Tong C, de Pauw A, Murray PJ, van Deursen JMA, Brindle PK (2006) Conditional Knockout Mice Reveal Distinct Functions for the Global Transcriptional Coactivators CBP and p300 in T-Cell Development. *Mol Cell Biol* 26: 789–809

Lee S, Kim JA, Kim HD, Chung S, Kim K, Choe HK (2019) Real-Time Temporal Dynamics of Bicistronic Expression Mediated by Internal Ribosome Entry Site and 2A Cleaving Sequence. *Mol Cells* 42: 418–425

Lim HW, Uhlenhaut NH, Rauch A, Weiner J, Hübner S, Hübner N, Won KJ, Lazar MA, Tuckermann J, Steger DJ (2015) Genomic redistribution of GR monomers and dimers mediates transcriptional response to exogenous glucocorticoid in vivo. *Genome Res* 25: 836–844

Lonard DM, Lanz RB, O'Malley BW (2007) Nuclear Receptor Coregulators and Human Disease. *Endocr Rev* 28: 575–587

Lonard DM, O'Malley BW (2012) Nuclear receptor coregulators: modulators of pathology and therapeutic targets. *Nat Rev Endocrinol* 8: 598–604

Luan J, Xiang G, Gómez-García PA, Tome JM, Zhang Z, Vermunt MW, Zhang H, Huang A, Keller CA, Giardine BM, *et al* (2021) Distinct properties and functions of CTCF revealed by a rapidly inducible degron system. *Cell Rep* 34: 108783

Luk C, Tsao MS, Bayani J, Shepherd F, Squire JA (2001) Molecular cytogenetic analysis of non-small cell lung carcinoma by spectral karyotyping and comparative genomic hybridization. *Cancer Genet Cytogenet* 125: 87–99

Mahat DB, Kwak H, Booth GT, Jonkers IH, Danko CG, Patel RK, Waters CT, Munson K, Core LJ, Lis JT (2016) Base-Pair Resolution Genome-Wide Mapping Of Active RNA polymerasesusing Precision Nuclear Run-On (PRO-seq). *Nat Protoc* 11: 1455–1476

Martire S, Nguyen J, Sundaresan A, Banaszynski LA (2020) Differential contribution of p300 and CBP to regulatory element acetylation in mESCs. *BMC Mol Cell Biol* 21: 55

McDowell IC, Barrera A, D'Ippolito AM, Vockley CM, Hong LK, Leichter SM, Bartelt LC, Majoros WH, Song L, Safi A, *et al* (2018) Glucocorticoid receptor recruits to enhancers and drives activation by motif-directed binding. *Genome Res* 28: 1272–1284

Mirzaei H, Khodadad N, Karami C, Pirmoradi R, Khanizadeh S (2020) The AP-1 pathway; A key regulator of cellular transformation modulated by oncogenic viruses. *Rev Med Virol* 30: e2088

Nagesh R, Kumar KMK, Kumar MN, Patil RH, Sharma SC (2021) Regulation of Jun and Fos AP-1 transcription factors by JNK MAPKs signaling cascade in areca nut extract treated KB cells. *Biochem Biophys Reports* 27: 101090

Natsume T, Kanemaki MT (2017) Conditional Degrons for Controlling Protein Expression at the Protein Level. *Annu Rev Genet* 51: 83–102

Natsume T, Kiyomitsu T, Saga Y, Kanemaki MT (2016) Rapid Protein Depletion in Human Cells by Auxin-Inducible Degron Tagging with Short Homology Donors. *Cell Rep* 15: 210–218

Negishi T, Kitagawa S, Horii N, Tanaka Y, Haruta N, Sugimoto A, Sawa H, Hayashi K, Harata M, Kanemaki MT (2021) The auxin-inducible degron 2 (AID2) system enables controlled protein knockdown during embryogenesis and development in Caenorhabditis elegans. *Genetics*

iyab218

Newton R, Shah S, Altonsy MO, Gerber AN (2017) Glucocorticoid and cytokine crosstalk: Feedback, feedforward, and co-regulatory interactions determine repression or resistance. *J Biol Chem* 292: 7163–7172

Nguyen DP, Li J, Yadav SS, Tewari AK (2014) Recent insights into NF-κB signalling pathways and the link between inflammation and prostate cancer. *BJU Int* 114: 168–176

Nishimura K, Fukagawa T (2021) A simple method to generate super-sensitive aid (ssAID)-based conditional knockouts using crispr-based gene knockout in various vertebrate cell lines. *Bio-protocol* 11: e4092

Nishimura K, Fukagawa T, Takisawa H, Kakimoto T, Kanemaki M (2009) An auxin-based degron system for the rapid depletion of proteins in nonplant cells. *Nat Methods* 6: 917–922

Oeckinghaus A, Ghosh S (2009) The NF-κB Family of Transcription Factors and Its Regulation. *Cold Spring Harb Perspect Biol* 1: a000034

Oh KS, Patel H, Gottschalk RA, Lee WS, Baek S, Fraser IDC, Hager GL, Sung MH (2017) Anti-Inflammatory Chromatinscape Suggests Alternative Mechanisms of Glucocorticoid Receptor Action. *Immunity* 47: 298-309.e5

Peng KJ, Wang JH, Su WT, Wang XC, Yang FT, Nie WH (2010) Characterization of Two Human Lung Adenocarcinoma Cell Lines by Reciprocal Chromosome Painting. *Zool Res* 31: 113–121

Prekovic S, Schuurman K, Mayayo-Peralta I, Manjón AG, Buijs M, Yavuz S, Wellenstein MD, Barrera A, Monkhorst K, Huber A, *et al* (2021) Glucocorticoid receptor triggers a reversible drug-tolerant dormancy state with acquired therapeutic vulnerabilities in lung cancer. *Nat Commun* 12: 4360

Ran FA, Hsu PD, Wright J, Agarwala V, Scott DA, Zhang F (2013) Genome engineering using the CRISPR-Cas9 system. *Nat Protoc* 8: 2281–2308

Rao NAS, McCalman MT, Moulos P, Francoijs KJ, Chatziioannou A, Kolisis FN, Alexis MN, Mitsiou DJ, Stunnenberg HG (2011) Coactivation of GR and NFKB alters the repertoire of their binding sites and target genes. *Genome Res* 21: 1404–1416

Sacta MA, Tharmalingam B, Coppo M, Rollins DA, Deochand DK, Benjamin B, Yu L, Zhang B, Hu X, Li R, *et al* (2018) Gene-specific mechanisms direct glucocorticoid-receptor-driven repression of inflammatory response genes in macrophages. *Elife* 7: e34864

Saito Y, Kanemaki MT (2021) Targeted Protein Depletion Using the Auxin-Inducible Degron 2 (AID2) System. *Curr Protoc* 1: e219

Sanyal A, Lajoie BR, Jain G, Dekker J (2012) The long-range interaction landscape of gene promoters. *Nature* 489: 109–113

Sasse SK, Gruca M, Allen MA, Kadiyala V, Song T, Gally F, Gupta A, Pufall MA, Dowell RD, Gerber AN (2019) Nascent transcript analysis of glucocorticoid crosstalk with TNF defines primary and cooperative inflammatory repression. *Genome Res* 29: 1753–1765

Schmidt SF, Larsen BD, Loft A, Mandrup S (2016) Cofactor squelching: Artifact or fact? *BioEssays* 38: 618–626

Sims K, Haynes CA, Kelly S, Allegood JC, Wang E, Momin A, Leipelt M, Reichart D, Glass CK, Cameron Sullards M, *et al* (2010) Kdo2-lipid A, a TLR4-specific agonist, induces de Novo sphingolipid biosynthesis in RAW264.7 macrophages, which is essential for induction of autophagy. *J Biol*

*Chem* 285: 38568–38579

Smoak KA, Cidlowski JA (2004) Mechanisms of glucocorticoid receptor signaling during inflammation. *Mech Ageing Dev* 125: 697–706

Surjit M, Ganti KP, Mukherji A, Ye T, Hua G, Metzger D, Li M, Chambon P (2011) Widespread Negative Response Elements Mediate Direct Repression by Agonist- Liganded Glucocorticoid Receptor. *Cell* 145: 224–241

Syed AP, Greulich F, Ansari SA, Uhlenhaut NH (2020) Anti-inflammatory glucocorticoid action: genomic insights and emerging concepts. *Curr Opin Pharmacol* 53: 35–44

Uhlenhaut NH, Barish GD, Yu RT, Downes M, Karunasiri M, Liddle C, Schwalie P, Hübner N, Evans RM (2013) Insights into Negative Regulation by the Glucocorticoid Receptor from Genome-wide Profiling of Inflammatory Cistromes. *Mol Cell* 49: 158–171

Vandevyver S, Dejager L, Libert C (2012) On the Trail of the Glucocorticoid Receptor: Into the Nucleus and Back. *Traffic* 13: 364–374

Vandevyver S, Dejager L, Libert C (2014) Comprehensive Overview of the Structure and Regulation of the Glucocorticoid Receptor. *Endocr Rev* 35: 671–693

Venz R, Pekec T, Katic I, Ciosk R, Ewald CY (2021) End-of-life targeted degradation of DAF-2 insulin/IGF-1 receptor promotes longevity free from growth-related pathologies. *Elife* 10: e71335

Verstrepen L, Bekaert T, Chau TL, Tavernier J, Chariot A, Beyaert R (2008) TLR-4, IL-1R and TNF-R signaling to NF-kappaB: variations on a common theme. *Cell Mol life Sci* 65: 2964–2978

Weikum ER, Knuesel MT, Ortlund EA, Yamamoto KR (2017a) Glucocorticoid receptor control of transcription: precision and plasticity via allostery. *Nat Rev Mol Cell Biol* 18: 159–174

Weikum ER, De Vera IMS, Nwachukwu JC, Hudson WH, Nettles KW, Kojetin DJ, Ortlund EA (2017b) Tethering not required: the glucocorticoid receptor binds directly to activator protein-1 recognition motifs to repress inflammatory genes. *Nucleic Acids Res* 45: 8596–8608

Wertz IE, Dixit VM (2010) Signaling to NF-κB: Regulation by Ubiquitination. *Cold Spring Harb Perspect Biol* 2: a003350

Yang X, Ma L (2022) Post-treatment with propofol inhibits inflammatory response in LPS-induced alveolar type II epithelial cells. *Exp Ther Med* 23: 249

Yesbolatova A, Natsume T, Hayashi K ichiro, Kanemaki MT (2019) Generation of conditional auxin-inducible degron (AID) cells and tight control of degron-fused proteins using the degradation inhibitor auxinole. *Methods* 164–165: 73–80

Yesbolatova A, Saito Y, Kitamoto N, Makino-Itou H, Ajima R, Nakano R, Nakaoka H, Fukui K, Gamo K, Tominari Y, *et al* (2020) The auxin-inducible degron 2 technology provides sharp degradation control in yeast, mammalian cells, and mice. *Nat Commun* 11: 5701

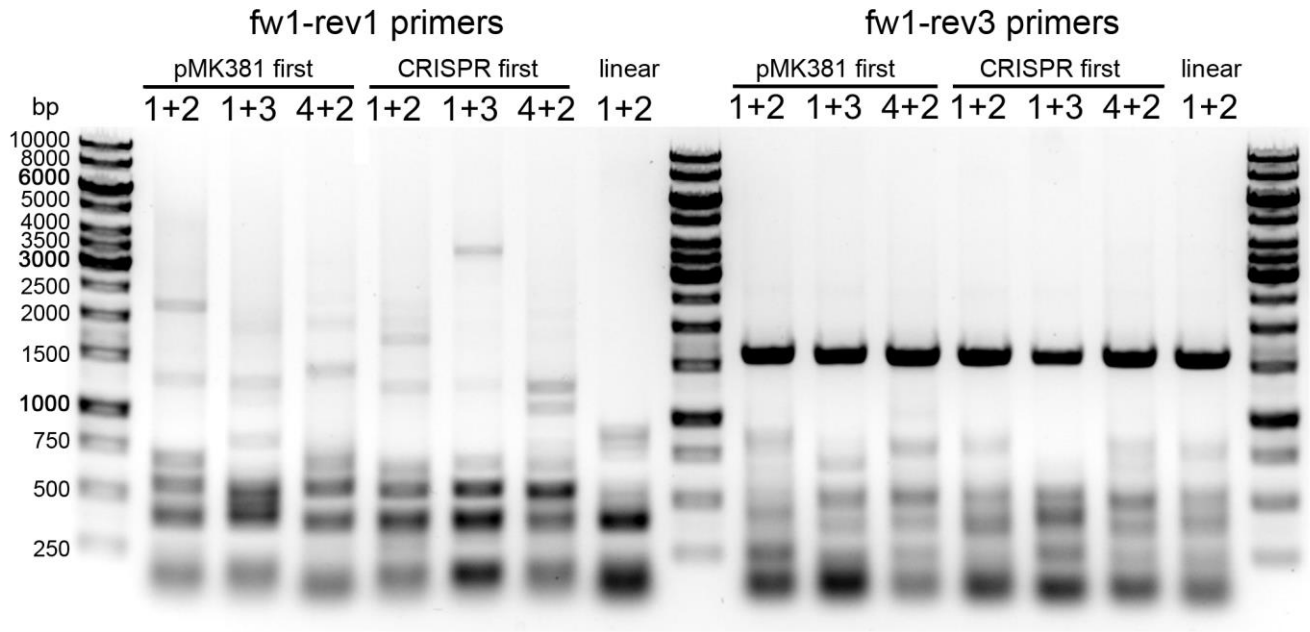
Yu H, Lin L, Zhang Z, Zhang H, Hu H (2020) Targeting NF-κB pathway for the therapy of diseases: mechanism and clinical study. *Signal Transduct Target Ther 2020 51* 5: 1–23

Yunusova A, Smirnov A, Shnaider T, Lukyanchikova V, Afonnikova S, Battulin N (2021) Evaluation of the OsTIR1 and AtAFB2 AID Systems for Genome Architectural Protein Degradation in Mammalian Cells. *Front Mol Biosci* 8: 757394

# Appendix 1. Supplementary Information

## Supplementary Method 1. Optimization of pMK381-CRISPR/Cas9 transfection order

For testing the order of plasmid and CRISPR/Cas9 transfections, A549 cells were plated with 300 000 cells/well in 6-well plates and transfected with either pMK381 or the CRISPR/Cas9 RNP complex first followed by the other the next day as described in Materials and methods section. For CRISPR/Cas9 transfections, all three crRNA pairs were used. Circular pMK381 was transfected with all three crRNA pairs and linearized pMK381 was transfected with crRNA 1+2. pMK381 was linearized by digesting with EcoRI (#FD0274, Fermentas) according to the manufacturer's instructions, extracted from a 0.5% UltraPure™ agarose (16500-100, Invitrogen) gel, and purified using the QIAEX® II Gel Extraction Kit (20021, QIAGEN) according to the manufacturer's instructions for 25 µg of a 6000-bp fragment except eluting by incubating at 50 °C for 10 minutes. After this, only 0.25 µg of linearized pMK381 was available and thus all of that was transfected followed by CRISPR/Cas9 RNP the next day. After 2 days from transfections, the cells were harvested by scraping in 1x PBS and gDNA was extracted using the DNeasy Blood & Tissue kit (69504, QIAGEN) according to the manufacturer's instructions. The samples were screened using PCR1 and PCR2 (Table 1) as described in the Materials and methods section except without the heat inactivation step and using the annealing temperature of 65 °C and extension time of 3 min for both. Based on the results (Fig. S1), it was concluded that the transfection order or the crRNA pair used did not make a difference for integration efficiency and therefore all crRNA pairs were used to generate colonies with pMK381 transfected first.




**Supplementary Figure 1. Transfection order of pMK381 and CRISPR/Cas9 does not affect OsTIR1$^{F74G}$ integration.** To elucidate if the transfection order of pMK381 and CRISPR/Cas9 affected integration and if one crRNA pair was more efficient than others, test transfections were done using all the different crRNA combinations and circular or linearized pMK381 by transfecting either pMK381 or CRISPR/Cas9 first. Pooled cells were then harvested and screened with fw1-rev1 and fw1-rev3 primers. As expected, all samples show the 1836-bp negative band (right). However, none show the positive 6336-bp fragment (right) nor the 941-bp fragment except for linear 1+2 (left), which shows a faint approximately 900-bp band and thus is most promising. Thus, the transfection order does not greatly improve the integration efficiency.

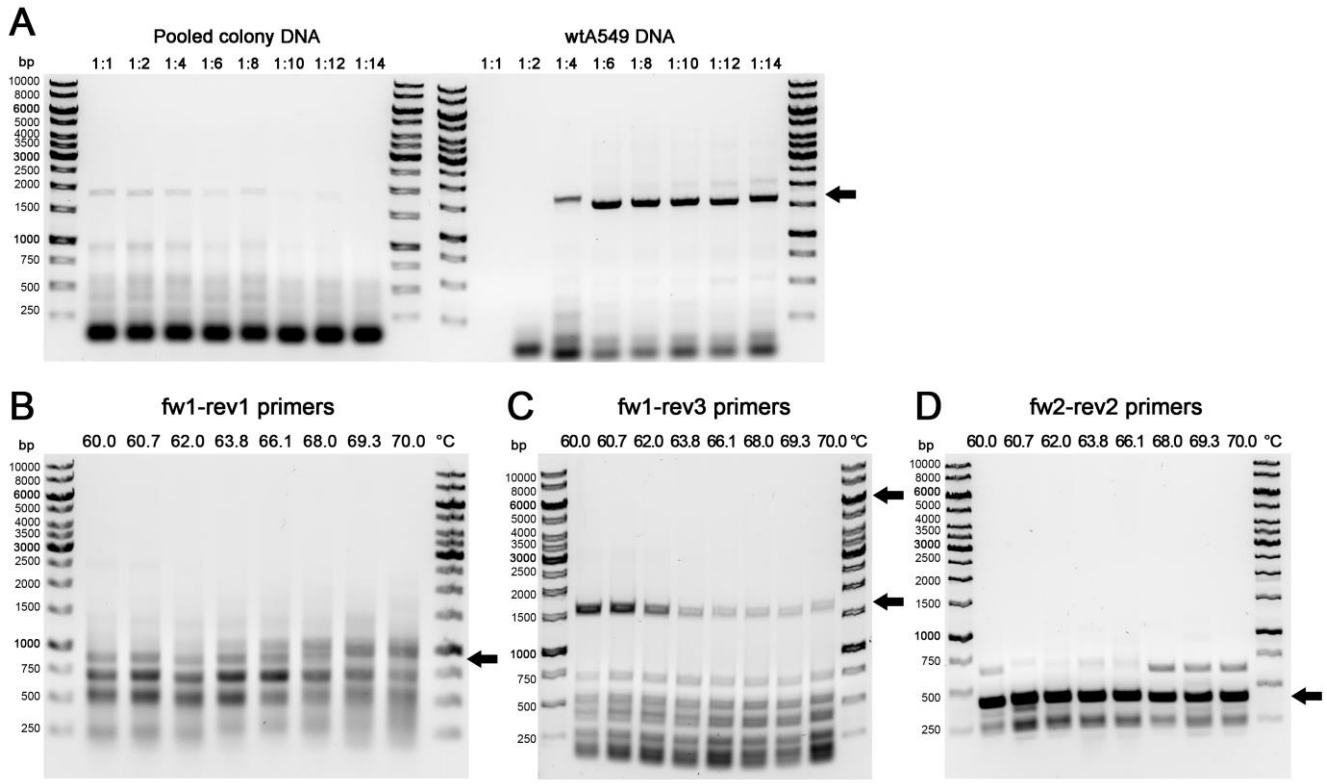
## Supplementary Method 2. Puromycin kill curve test

A549 cells were seeded in 6-well plates as duplicate with 25 500 cells/well to mimic the low concentration of pMK381-CRISPR/Cas9-transfected cells. At this concentration, single-cell resolution was achieved. Puromycin concentrations tested were 0.25 µg/ml, 0.5 µg/ml, 0.75 µg/ml, 1.0 µg/ml, 1.5 µg/ml, and control (0 µg/ml). After 5 days under selection, the control cells were growing normally and forming single-cell colonies while the 0.25 µg/ml concentration showed stunted growth with smaller colonies and concentrations of 0.5 µg/ml and higher had no live cells. This did not change during the total of 17 days cells spent under selection, except for that towards the end few cells were still alive in the 0.25 µg/ml concentration but alive nonetheless (data not shown). Based on this, the 0.5 µg/ml concentration was sufficient to kill all cells in a low-density culture.

## Supplementary Method 3. Optimization of template concentration and primer annealing temperatures in the presence of QuickExtract lysis reagent for genomic PCR

To optimize the amount of template for screening single cell colonies by PCR, a PCR using pooled colony DNA as a gradient of 1:1, 1:2, 1:4, 1:6, 1:8, 1:10, 1:12, and 1:14 dilutions or wtA549 gDNA (final conc. 2.5 ng/µl) with an equal volume of QuickExtract™ and the same dilution gradient as above as control was run using PCR1 as described in Materials and methods section except with extension for 1 min and 35 cycles. Based on this, 1:2 dilution was determined to be optimal for screening single-cell colonies (Fig. S2A).

Annealing temperatures for primer pairs (Table 1) in the presence of QuickExtract™ were optimized by running a 60-70 °C gradient PCR with the T100 thermal cycler (Bio-Rad) using 1:2 diluted pooled colony gDNA for PCR1 and PCR2 or pMK381 (final conc. 0.2 ng/µl) with added QuickExtract™ 1:2 compared to template volume for PCR3 as template. The PCR program was run with the extension times reported in Table 1 without the heat inactivation step. Based on the results, the optimal annealing temperatures were used thereafter (Fig. S2B-D).



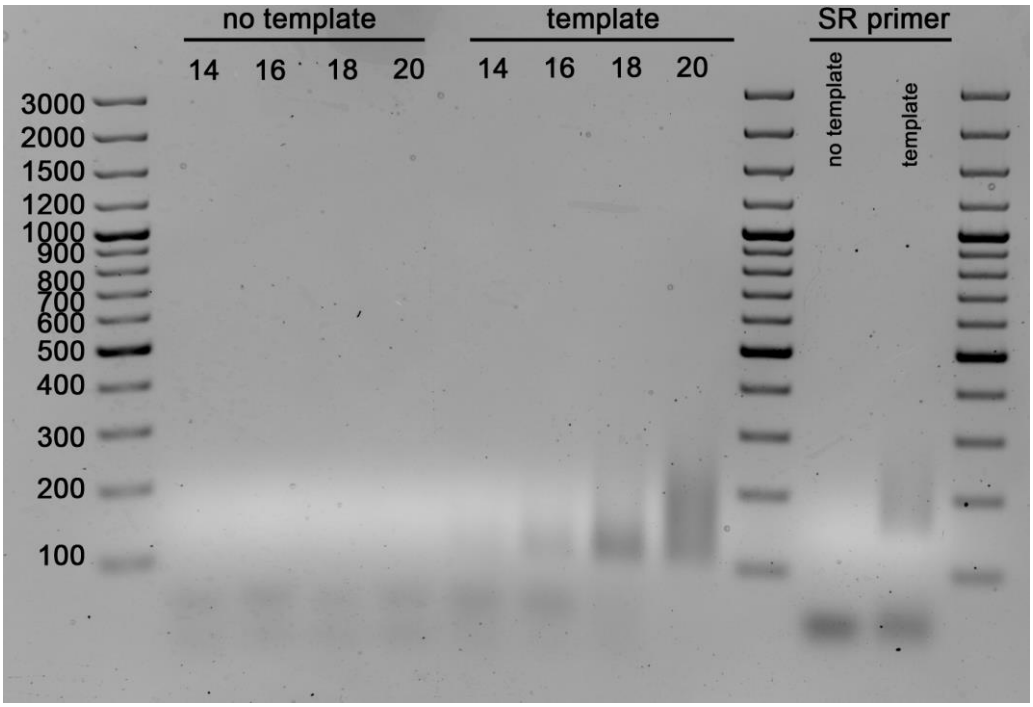

**Supplementary Figure 2. QuickExtract lysis reagent greatly alters polymerase activity. A)** To determine how QuickExtract affects polymerase activity, PCR was run with a dilution series of pooled DNA from single-cell colonies (left) or wtA549 DNA with equal dilution of QuickExtract (right). For wtA549 DNA, undiluted QuickExtract completely prevented amplification, a 1:2 dilution enabled amplification of primers but not the

1836-bp product, and from the 1:4 dilution onwards the concentration of QuickExtract was low enough to permit amplification of the product. Conversely, for pooled colony DNA, the 1836-bp product was amplified up to the 1:8 dilution, after which the concentration of DNA was too low for efficient amplification due to the lower concentration compared to wtA549 DNA. This indicates that when QuickExtract is used for lysis, its interfering components are exhausted, causing the reagent to lose its potency enough to permit amplification. Based on these results, the 1:2 dilution of harvested DNA was used for PCR thereafter. **B-D)** Annealing temperature optimizations for fw1-rev1 (*i.e.* PCR1) (B), fw1-rev3 (C) (*i.e.* PCR2), and fw2-rev2 (*i.e.* PCR3) (D) primers were run by gradient PCR to minimize background bands and maximize production of the desired product from the single-cell colony DNA that is less purified than wtA549 DNA. Fw1-rev1 primers produced clearer bands at lower temperatures but it is important to note that the expected 961-bp product can only be produced from a clone with successful integration and thus was not seen here, complicating the optimization of these primers as no positive control existed. However, it was estimated that 63 °C could be a reasonable annealing temperature as evidenced by the relatively sharpest bands at this temperature. Fw1-rev3 primers produced the expected fragment most efficiently at low temperatures, and the optimal annealing temperature was determined to be 60 °C with the darkest product band and faintest background bands. For fw2-rev2, all temperatures produced the expected product equally efficiently with only some more prominent background bands at higher temperatures, mainly due to the lower complexity of the used plasmid as template. Therefore, the suggested 65 °C annealing temperature was kept as no evidence suggested it should be changed. These temperatures differed greatly from the annealing temperatures suggested by the Thermo Scientific Tm calculator for Phusion polymerase, which were 65.1 °C, 65.0 °C, and 64.7 °C, for fw1-rev1, fw1-rev3, and fw2-rev2, respectively. The arrows indicate the expected product sizes.

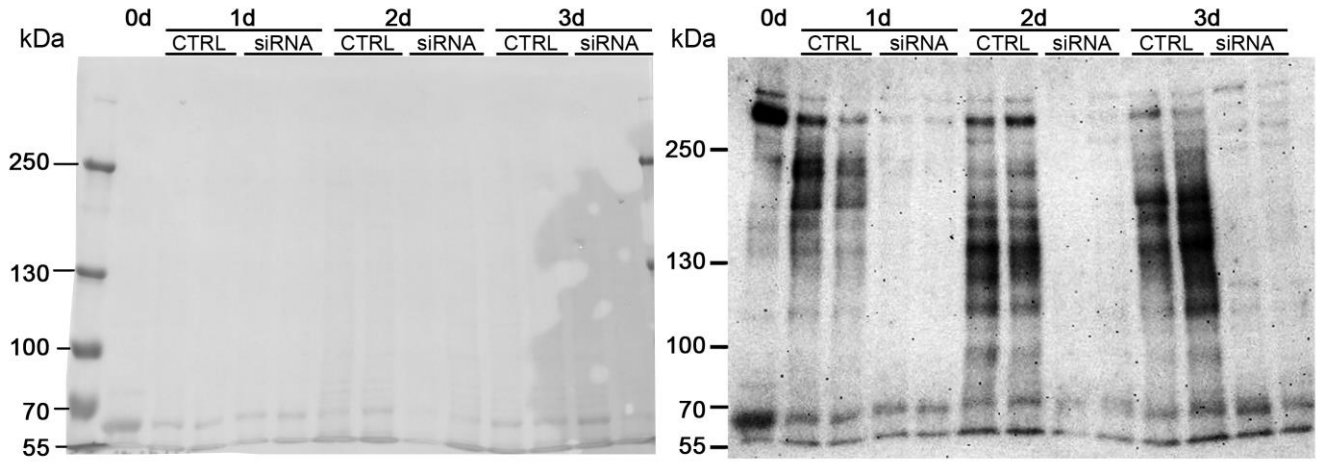
## Supplementary Method 4. Optimizing amplification cycle number for PRO-seq library prep

Before PCR amplification of the PRO-seq libraries, the number of amplification cycles was tested by running a test PCR using a pooled sample from all PRO-seq samples or no template as control to assess the production of primer dimers as an indicator for amplification efficiency. PCR samples were prepared as 10-µl reactions of 1X Phusion™ HF buffer, 300 µM dNTPs each, 187.5 nM forward and reverse primer each, 0.02 U/µlPhusion™ High-Fidelity DNA polymerase with 3 µl of pooled library as template in order to most closely mimic the conditions of the amplification PCR described in the NEBNext Small RNA Library Prep Set for Illumina kit. The primers used were forward 5'-CCGAGATCTACACGTTCAGAGTTCTACAGTCCGA-3' and reverse 5'-GTGACTGGAGTTCAGACGTGTGCTCTTCCGATCT-3' which were designed based on the primers included in the kit. Additionally, for one sample with and without template, the kit primers were used to see if there was a difference in efficiency between the primers in the kit and the ones designed based on them. The samples were run for 14, 16, 18, or 20 cycles using the following PCR program: 98 °C for 30 sec; 14-20 cycles of 98 °C degrees for 7 sec, 65 °C for 25 sec and 72 °C for 45 sec; 72 °C for 5 min. The sample with the kit primers was run for 16 cycles. To visualize, the

samples were run on a 2% agarose gel with 0.01% ethidium bromide. Based on this, 18 cycles were used for the amplification of the PRO-seq examples (Fig. S3).




**Supplementary Figure 3. Optimization of amplification cycle number for PRO-seq library preparation.** The optimal amplification cycle number is 18 cycles for these PRO-seq libraries because when amplified for 18 cycles there is less primer dimers (<100 bp) than with 16 cycles, more library product (100-200 bp) than with 16 cycles, and less overamplification than with 20 cycles. The primers that came with the kit (SR primer) produce primer dimers more efficiently than the primers designed for this experiment. As expected, only primer dimers are observed when no template is added.

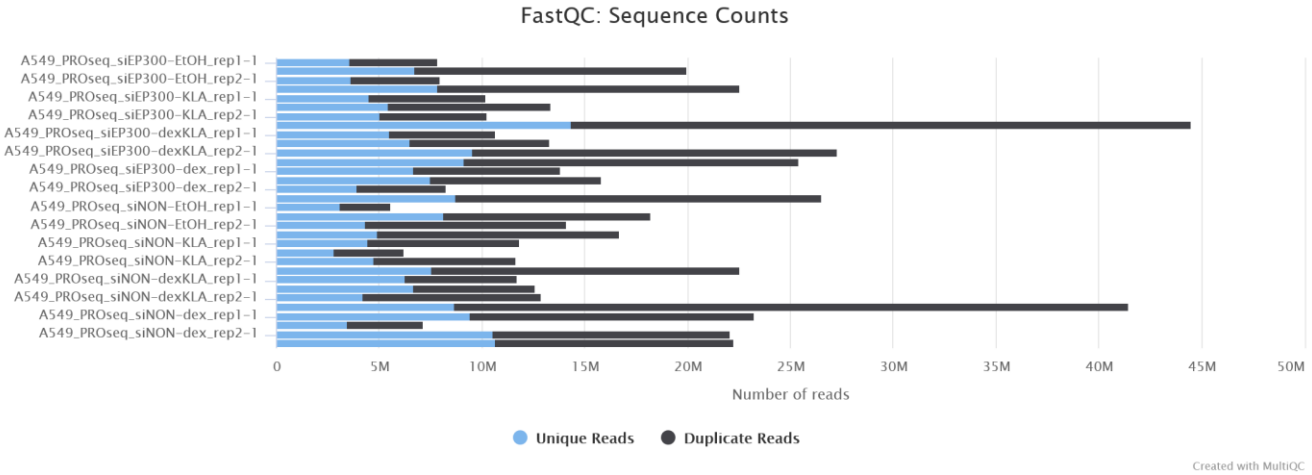



**Supplementary Figure 4. Validation of the effectiveness of siEP300 by Western blot.** A549 cells transfected with siEP300 (siRNA) or siNON (CTRL) for 1, 2, and 3 days show that the level of p300 (264 kDa) is markedly reduced compared to siNON at all three of these days but most greatly at day 2 where almost no p300 can be observed (right). The Ponceau stained membrane on the left shows approximately equal protein loading across all samples.

Supplementary Table 1. Used software, software versions, and citations.

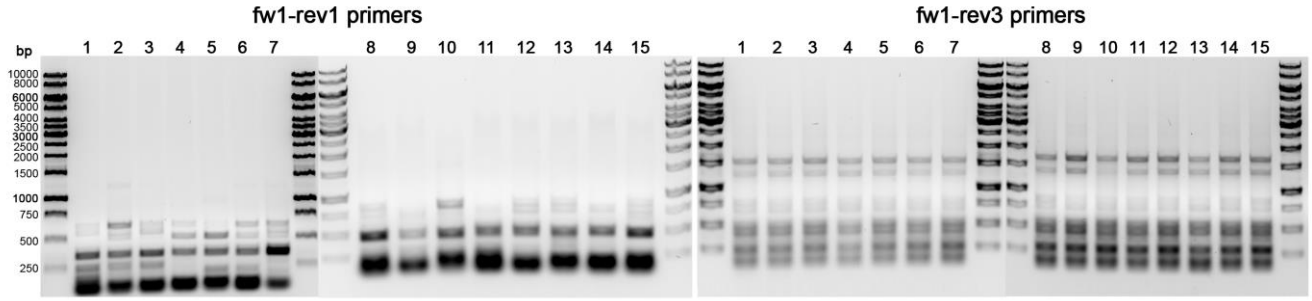
| Resource | Version | Citation |
|---|---|---|
| R | 4.1.3 | R Core Team (2022) R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria |
| RStudio | 2022.2.0.443 | RStudio Team (2022) RStudio: Integrated Development Environment for R. RStudio, PBC, Boston, MA |
| Command line modules | | |
| Trimmomatic | 0.36 | Bolger AM, Lohse M, Usadel B (2014) Trimmomatic: A flexible trimmer for Illumina Sequence Data. *Bioinformatics* 30: 2114–2120 |
| DESeq2 | part of HOMER | Love MI, Huber W, Anders S (2014) Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biol* 15: 550 |
| HOMER | 4.10 | Heinz S, Benner C, Spann N, Bertolino E, Lin YC, Laslo P, Cheng JX, Murre C, Singh H, Glass CK (2010) Simple Combinations of Lineage-Determining Transcription Factors Prime cis-Regulatory Elements Required for Macrophage and B Cell Identities. *Mol Cell* 38: 576-589 |
| bwa | 0.7.17 | Li H, Durbin R (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics* 25: 1754-1760 |
| proseq2.0 | 2.0 | Chu T, Wang Z, Chou SP, Danko CG (2019) Discovering Transcriptional Regulatory Elements From Run‐On and Sequencing Data Using the Web‐Based dREG Gateway. *Curr Protoc Bioinformatics* 66: e70. |
| BEDOPS | v2.4.39 | Neph S, Kuehn MS, Reynolds AP, Haugen E, Thurman RE, Johnson AK, Rynes E, Maurano MT, Vierstra J, Thomas S, Sandstrom R, Humbert R, Stamatoyannopoulos JA (2012) BEDOPS: high-performance genomic feature operations. *Bioinformatics* 28: 1919-1920 |
| bedtools | v2.29.2 | Quinlan AR, Hall IM (2010) BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics* 6: 841-842 |
| Online tools | | |
| dREG | - | Wang Z, Chu T, Choate LA, Danko CG (2019) Identification of regulatory elements from nascent transcription using dREG. *Genome Res* 29: 298-308 |
| Metascape | 3.5 | Zhou Y, Zhou B, Pache L, Chang M, Khodabakhshi AH, Tanaseichuk O, Benner C, Chanda SK(2019) Metascape provides a biologist-oriented resource for the analysis of systems-level datasets. *Nat Commun* 10: 1523 |
| R packages | | |
| ggplot2 | 3.3.5 | H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016. |

| Resource | Version | Citation |
|---|---|---|
| ggnetwork | 0.5.10 | François Briatte (2021) ggnetwork: Geometries to Plot Networks with 'ggplot2'. |
| network | 1.17.1 | Butts C (2015) network: Classes for Relational Data. The Statnet Project<br>Butts C (2008) network: a Package for Managing Relational Data in R. *J Stat Softw* 24: 1–36 |
| data.table | 1.14.2 | Dowle M, Srinivasan A (2021) data.table: Extension of `data.frame` |
| ComplexHeatmap | 2.10.0 | Gu Z, Eils R, Schlesner M (2016) Complex heatmaps reveal patterns and correlations in multidimensional genomic data. *Bioinformatics* 32: 2847-9 |
| ggrepel | 0.9.1 | Slowikowski K (2021) ggrepel: Automatically Position Non-Overlapping Text Labels with 'ggplot2' |
| circlize | 0.4.14 | Gu Z, Gu L, Eils R, Schlesner M, Brors B (2014) circlize implements and enhances circular visualization in R. *Bioinformatics* 30: 2811-2812 |
| gridExtra | 2.3 | Auguie B (2017) gridExtra: Miscellaneous Functions for "Grid" Graphics. |
| ggVennDiagram | 1.2.0 | Gao C (2021) ggVennDiagram: A 'ggplot2' Implement of Venn Diagram |
| RColorBrewer | 1.1-2 | Neuwirth E (2014) RColorBrewer: ColorBrewer Palettes |
| universalmotif | 1.12.4 | Tremblay BJ (2022) universalmotif: Import, Modify, and Export Motifs with R. R package |
| readxl | 1.4.0 | Wickham H, Bryan J (2022) readxl: Read Excel Files |
| **Other resources** | | |
| Blacklisted sequences | V1 | Amemiya HM, Kundaje A, Boyle AP (2019) The ENCODE Blacklist: Identification of Problematic Regions of the Genome. *Sci rep* 9: 9354 |

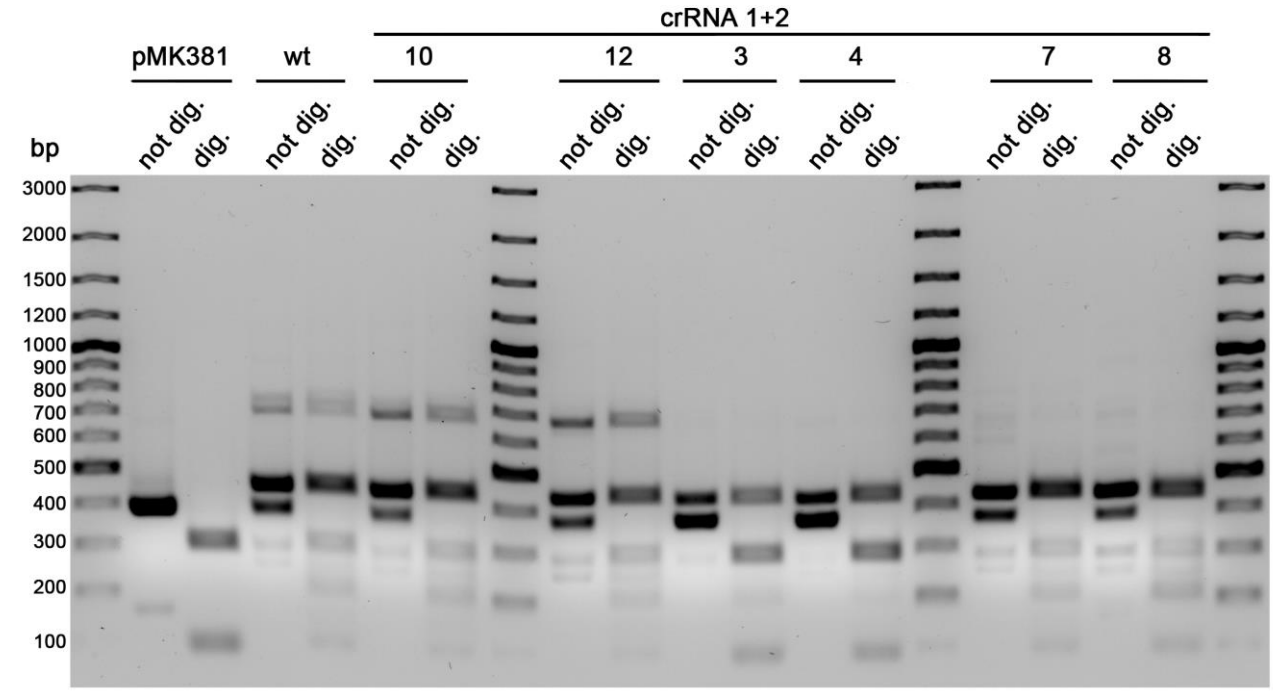



**Supplementary Figure 5. Sequencing depth and duplication level of trimmed PRO-seq reads.** The libraries were sequenced to a depth ranging between 5 million and 45 million reads. The trimmed sequences had

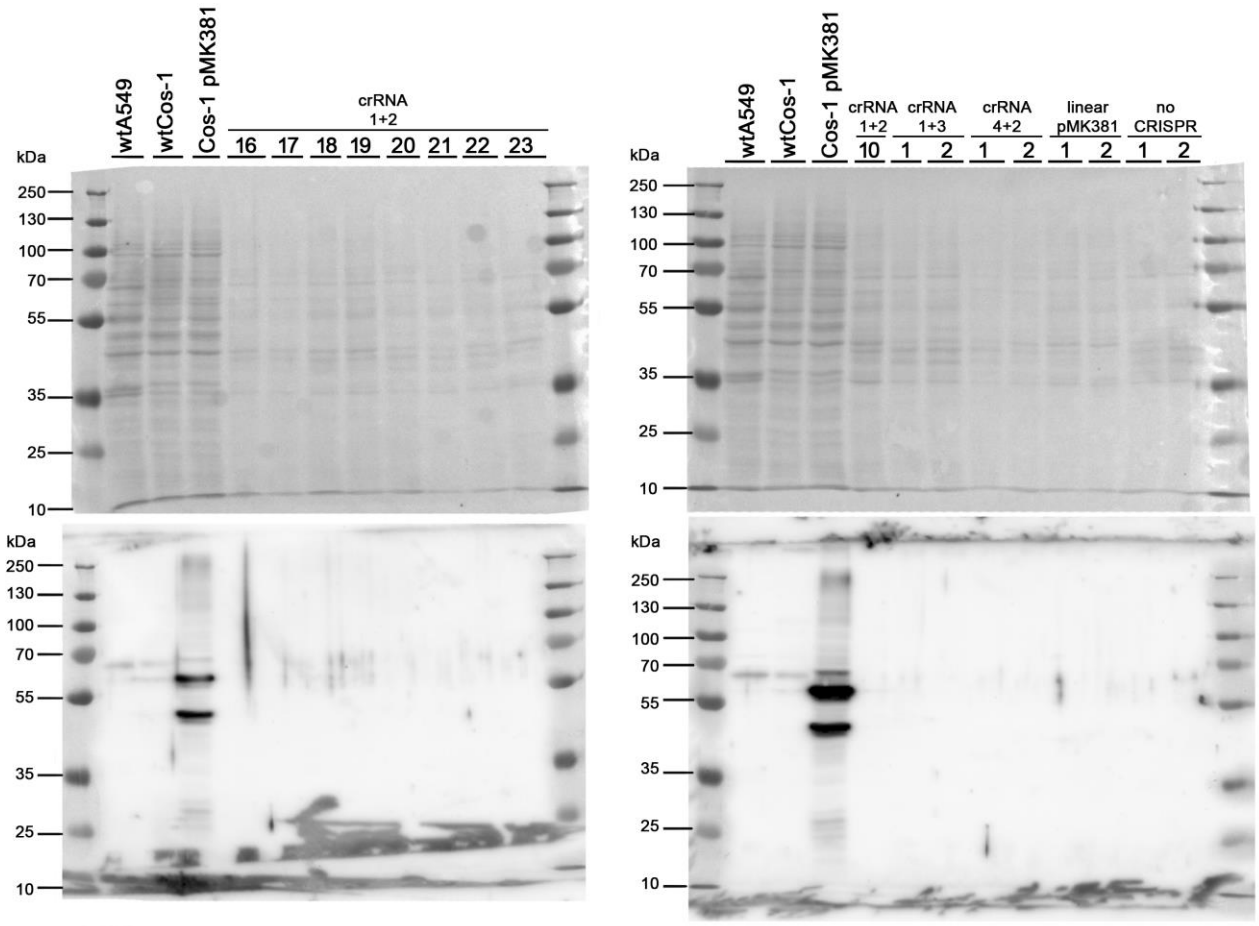
high levels of PCR duplicates (black) compared to unique reads (blue) as only approximately 3-15 million reads were unique.



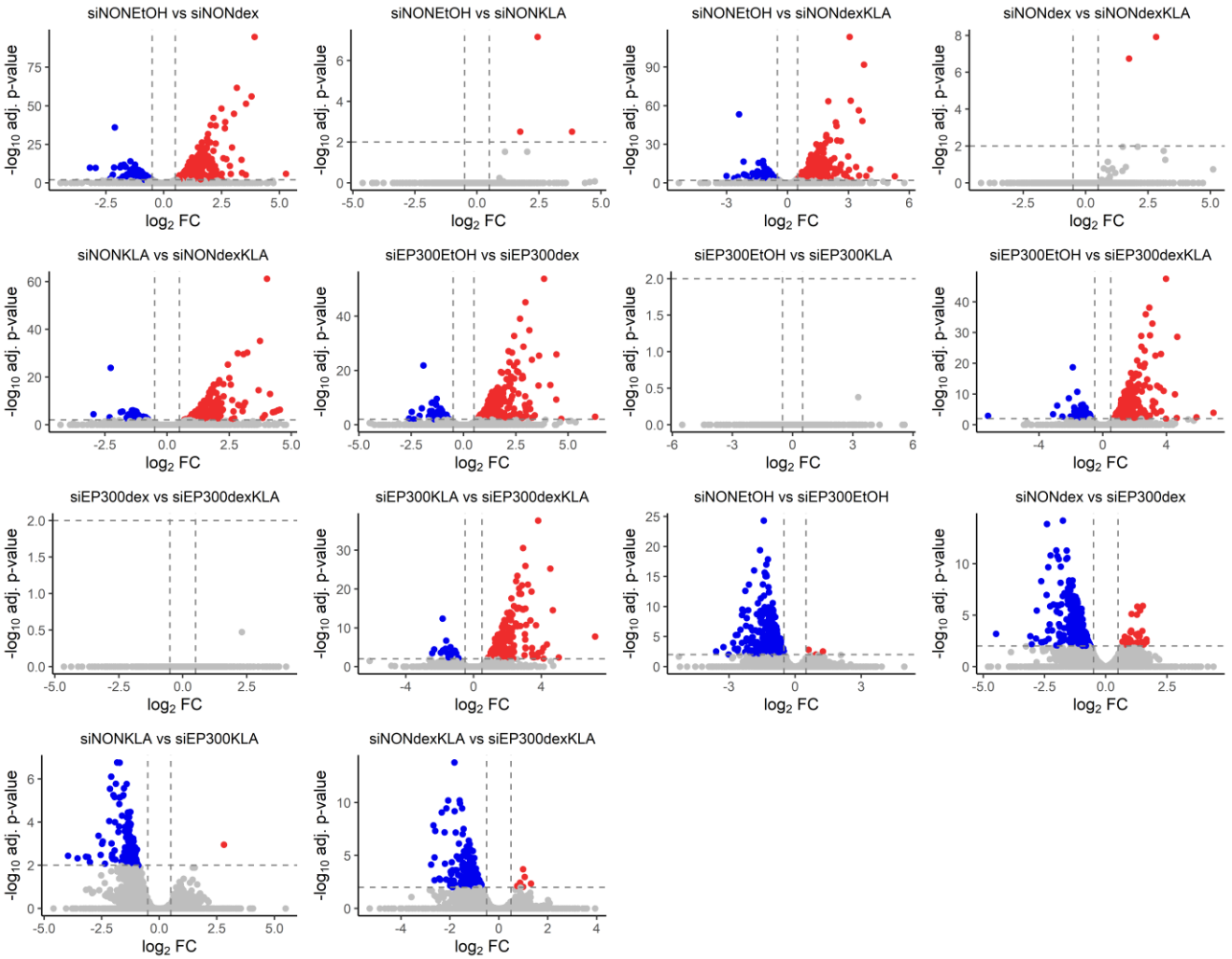

**Supplementary Figure 6. Uncropped versions of Fig. 3B.** Compared to Fig. 3B, the <1000 bp background bands of PCR2 (fw1-rev3 primers) can be seen.



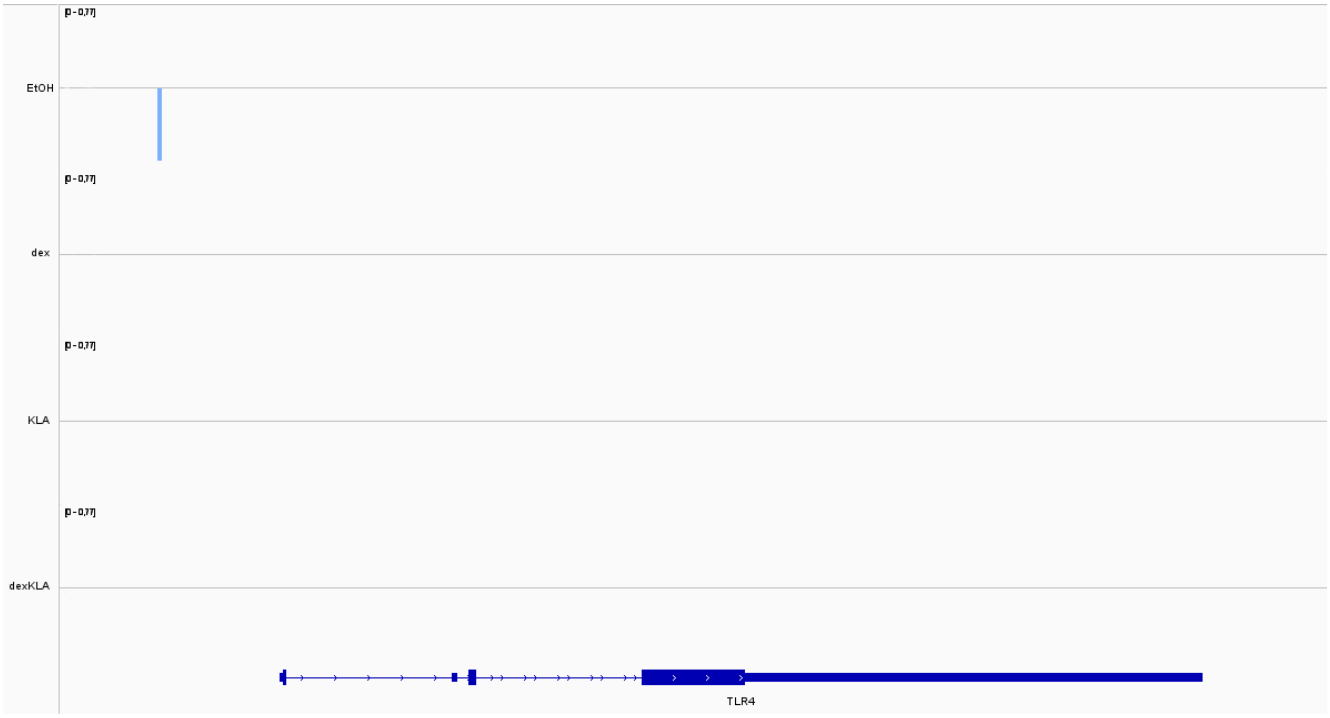

**Supplementary Figure 7. Uncropped version of Fig. 3C.** This figure shows that nothing essential was cropped out of Fig. 3C.



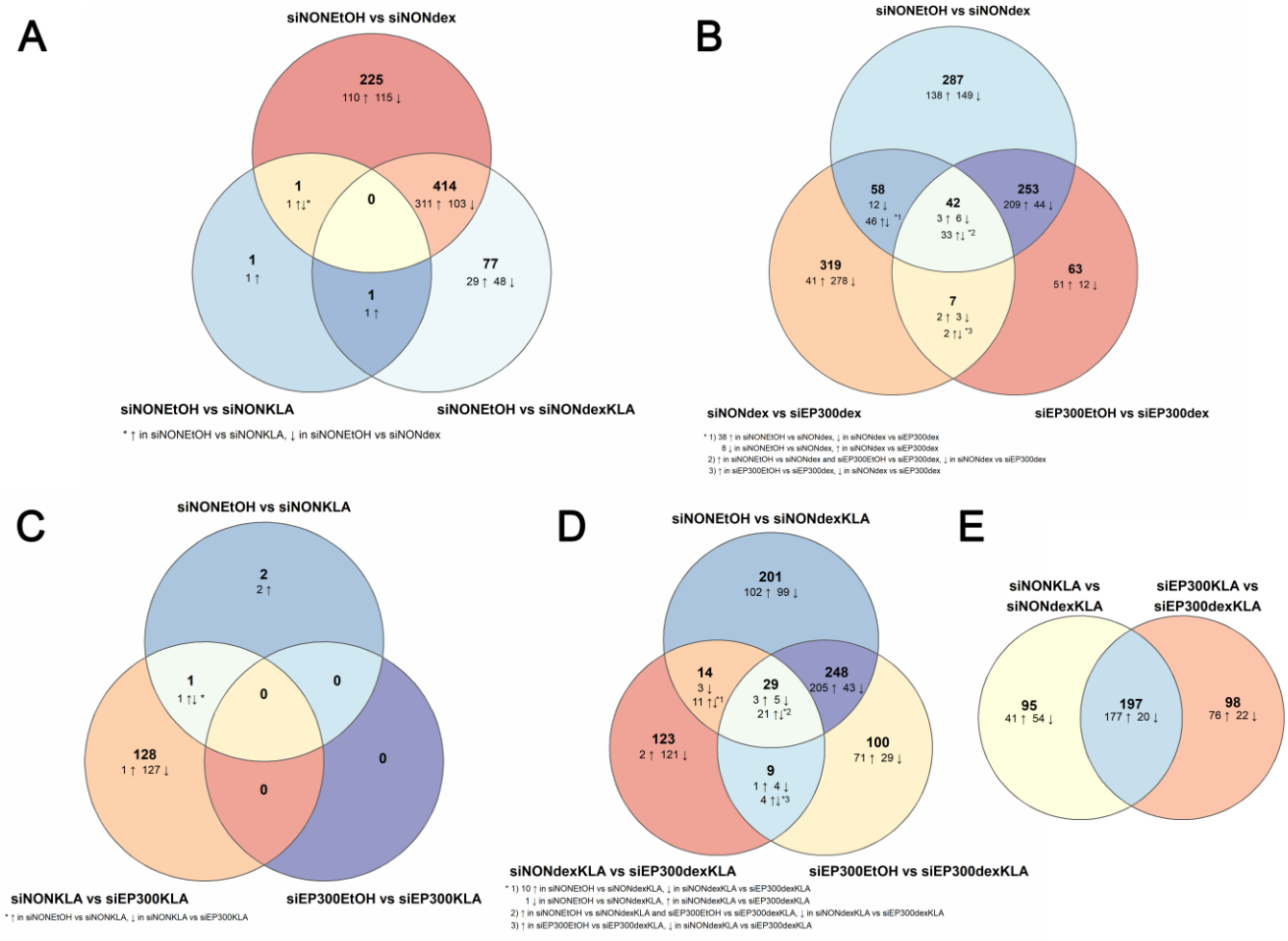

**Supplementary Figure 8. Uncropped version of Fig. 3D.** Ponceau stained membranes are on the top and membranes incubated with the OsTIR1 antibody are on the bottom. The controls of the membranes on the right were shown in Fig. 3D but the controls on the left membranes are essentially the same with both positive controls showing the ~64 kDa OsTIR1$^{F74G}$.



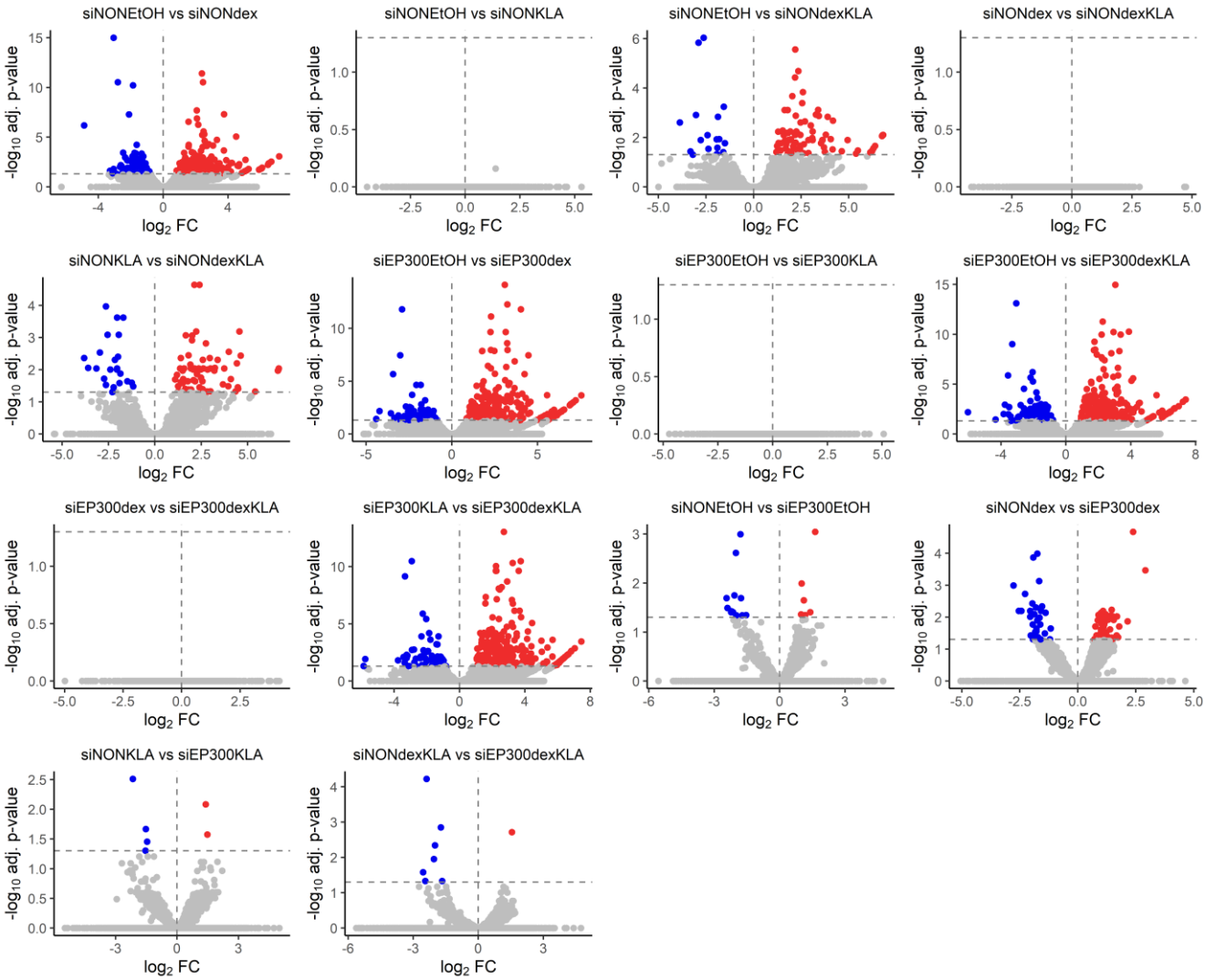

**Supplementary Figure 9. Volcano plots of genes for all comparisons.** Related to Fig. 4A. The comparisons modeling KLA effect (EtOH vs KLA, dex vs dexKLA) show very few or no DT genes, corroborating that the A549 cells did not respond to KLA, whereas the comparisons modeling dex effect (EtOH vs dex, EtOH vs dexKLA, KLA vs dexKLA) show a majority of upregulated genes with some downregulated genes. In the siNON vs siEP300 comparisons modeling siRNA effect, nearly all DT genes are downregulated. The genes not DT are shown as grey dots while the upregulated and downregulated genes are shown as red and blue dots, respectively. The dashed lines show the $\log_2$ FC cutoff (vertical) and the negative $\log_{10}$ adjusted p-value cutoff (horizontal).



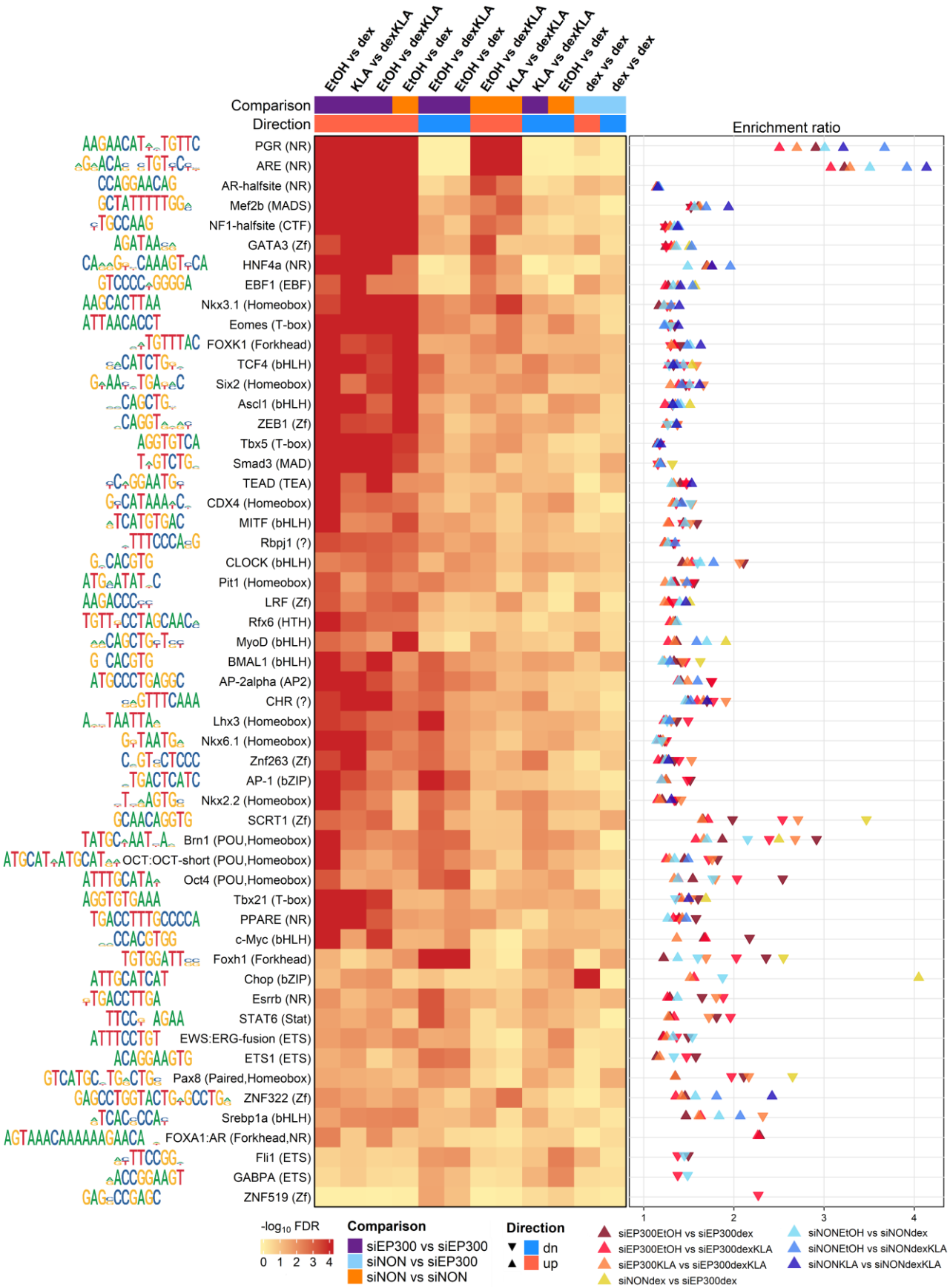

**Supplementary Figure 10. TLR4 is not transcribed in A549 cells.** The *TLR4* gene is shown at the bottom with tracks for PRO-seq signal from each treatment above it. The only observed very small peak (notice scale, same to both directions of the strand) on the negative strand is found in the siNONEtOH treatment (blue) but no other treatments show transcription of TLR4. siEP300 treatment signal would be shown in red.




**Supplementary Figure 11. Venn diagrams illustrate the overlap of DT genes in different comparisons.** Related to Fig. 4, especially Fig. 4C-E. **A)** Venn diagram of the overlap of DT genes in siNON vs siNON comparisons. This is otherwise the same figure as Fig. 4C except siNONKLA vs siNONdexKLA has been replaced by siNONEtOH vs siNONKLA. This shows that of the 3 upregulated genes in siNONEtOH vs siNONKLA, one is shared with siNONEtOH vs siNONdexKLA and one is downregulated in siNONEtOH vs siNONdex. **B-D)** Venn diagrams of the dex (B), KLA (C), and dexKLA (D) treatments across the different siRNA conditions show DT genes are mostly shared between the siNON vs siNON and siEP300 vs siEP300 comparisons and to a lesser degree with the siNON vs siEP300 comparison. **E)** In the KLA vs dexKLA comparison most DT genes are also shared between siNON vs siNON and siEP300 vs siEP300 comparisons similarly to EtOH vs dex and EtOH vs dexKLA. The bolded numbers indicate total counts, the smaller numbers indicate how many of these are up- (↑) and downregulated (↓).




**Supplementary Figure 12. Volcano plots of enhancers for all comparisons.** Related to Fig. 5A. Similarly to DT genes, the comparisons modeling KLA effect (EtOH vs KLA, dex vs dexKLA) show very few or no DT enhancers, which supports the conclusion that A549 did not respond to KLA, while the comparisons modeling dex effect (EtOH vs dex, EtOH vs dexKLA, KLA vs dexKLA) show a majority of upregulated enhancers with some downregulated enhancers and with more enhancers in the siEP300 vs siEP300 comparisons than the siNON vs siNON comparison. In the siNON vs siEP300 comparisons modeling siRNA effect, there are very few DT enhancers. The enhancers not DT are shown as grey dots while the upregulated and downregulated enhancers are shown as red and blue dots, respectively. The dashed lines show the $\log_2$ FC cutoff (vertical) and the negative $\log_{10}$ adjusted p-value cutoff (horizontal). Because there was no FC cutoff, the line is shown at 0.




**Supplementary Figure 13. AREs representing GREs are enriched to upregulated enhancers.** Known motif analysis was done from upregulated and downregulated enhancers using HOMER, motifs with similar consensus sequences were merged together to reduce redundancy and, of these, the motif with the lowest

FDR in any one comparison was kept as a representative motif (Supplementary Table 2). To further reduce insignificant motifs, the top 5 most significant motifs from within each comparison were selected, and the comparisons that did not have more than one significantly (FDR < 0.05) enriched motif were excluded. As expected, GREs, represented by the closely related androgen receptor (AR) response elements (ARE) and progesterone receptor (PGR) motifs were significantly enriched with high enrichment ratios to all upregulated enhancers but not to downregulated enhancers or to the siNON vs siEP300 comparisons. The AR half-site motif, potentially also representing GR half-sites, is also significantly enriched to the same enhancers with low enrichment over background, though this might be because the half-site consensus sequence could likely be recognized in the full palindromic ARE sequences, leading to a high enrichment in the background. AP-1 and NF-κB motifs are also enriched to both up- and downregulated siEP300 vs siEP300 enhancers or to upregulated enhancers in all three of the siEP300 vs siEP300 comparisons, respectively. The NF-κB motif was not among the top 5 most significant motifs in any comparison and thus is not seen in the figure but the enrichment was still significant. Many other motifs are also enriched to upregulated enhancers but very few are enriched exclusively to downregulated enhancers except for Fli1, GABPA, and ZNF519 motifs, most likely due to the low number of downregulated enhancers. The consensus sequences for the representative motifs are shown on the left. The enrichment ratios (right) calculated as the % of enhancers in the comparison with the motif divided by the % of background enhancers with the motif (enrichment ratio of 1 means no enrichment over background) are only plotted for comparisons where the motif was significantly (FDR < 0.05) enriched. See Fig. S14 for an individual version of the enrichment ratio plot.

**Supplementary Table 2. The representative known motifs from HOMER motif analysis shown in Fig. S13 and S14 and the motifs that were merged with them that they represent.**

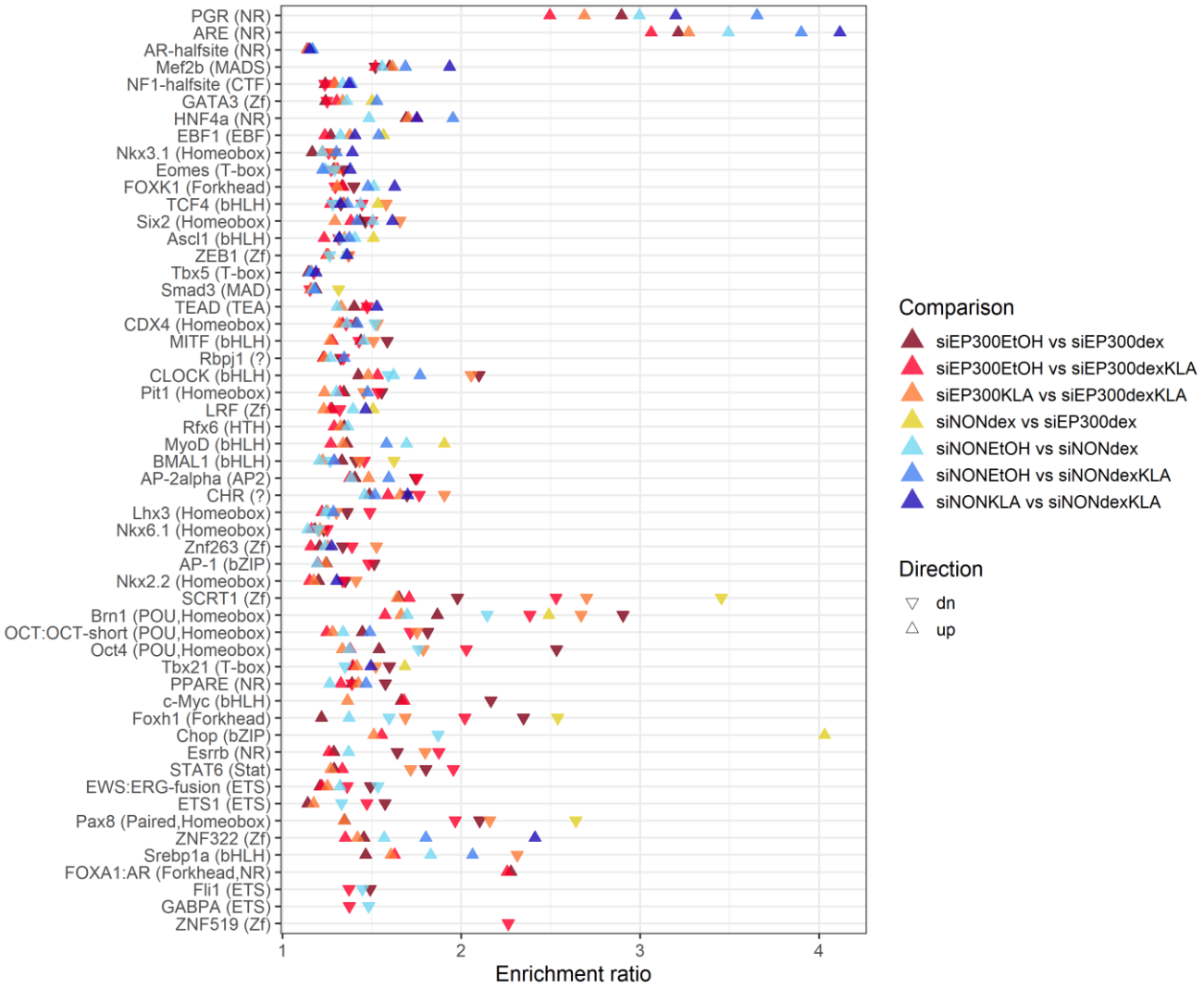
| Representative motif | Merged motifs |
|---|---|
| PGR(NR)/EndoStromal-PGR-ChIP-Seq(GSE69539)/Homer | PGR(NR)/EndoStromal-PGR-ChIP-Seq(GSE69539)/Homer |
| ARE(NR)/LNCAP-AR-ChIP-Seq(GSE27824)/Homer | GRE(NR),IR3/RAW264.7-GRE-ChIP-Seq(Unpublished)/Homer/<br>PR(NR)/T47D-PR-ChIP-Seq(GSE31130)/Homer/<br>ARE(NR)/LNCAP-AR-ChIP-Seq(GSE27824)/Homer/<br>GRE(NR),IR3/A549-GR-ChIP-Seq(GSE32465)/Homer |
| Nkx3.1(Homeobox)/LNCaP-Nkx3.1-ChIP-Seq(GSE28264)/Homer | Nkx3.1(Homeobox)/LNCaP-Nkx3.1-ChIP-Seq(GSE28264)/Homer/<br>Bapx1(Homeobox)/VertebralCol-Bapx1-ChIP Seq(GSE36672)/Homer/<br>Nkx2.5(Homeobox)/HL1-Nkx2.5.biotin-ChIP-Seq(GSE21529)/Homer |
| Mef2b(MADS)/HEK293-Mef2b.V5-ChIP-Seq(GSE67450)/Homer | Mef2d(MADS)/Retina-Mef2d-ChIP-Seq(GSE61391)/Homer/<br>Mef2b(MADS)/HEK293-Mef2b.V5-ChIP-Seq(GSE67450)/Homer/<br>Mef2c(MADS)/GM12878-Mef2c-ChIP-Seq(GSE32465)/Homer/<br>Mef2a(MADS)/HL1-Mef2a.biotin-ChIP-Seq(GSE21529)/Homer |
| ZNF322(Zf)/HEK293-ZNF322.GFP-ChIP-Seq(GSE58341)/Homer | ZNF322(Zf)/HEK293-ZNF322.GFP-ChIP-Seq(GSE58341)/Homer |

| Representative motif | Merged motifs |
|---|---|
| AR-halfsite(NR)/LNCaP-AR-ChIP-Seq(GSE27824)/Homer | AR-halfsite(NR)/LNCaP-AR-ChIP-Seq(GSE27824)/Homer |
| Eomes(T-box)/H9-Eomes-ChIP-Seq(GSE26097)/Homer | Eomes(T-box)/H9-Eomes-ChIP-Seq(GSE26097)/Homer |
| FOXK1(Forkhead)/HEK293-FOXK1-ChIP-Seq(GSE51673)/Homer | FOXK2(Forkhead)/U2OS-FOXK2-ChIP-Seq(E-MTAB-2204)/Homer/<br>FOXK1(Forkhead)/HEK293-FOXK1-ChIP-Seq(GSE51673)/Homer<br>Foxo3(Forkhead)/U2OS-Foxo3-ChIP-Seq(E-MTAB-2701)/Homer |
| NF1-halfsite(CTF)/LNCaP-NF1-ChIP-Seq(Unpublished)/Homer | NF1-halfsite(CTF)/LNCaP-NF1-ChIP-Seq(Unpublished)/Homer |
| Six2(Homeobox)/NephronProgenitor-Six2-ChIP-Seq(GSE39837)/Homer | Six4(Homeobox)/MCF7-SIX4-ChIP-Seq(Encode)/Homer/<br>Six2(Homeobox)/NephronProgenitor-Six2-ChIP-Seq(GSE39837)/Homer/<br>Six1(Homeobox)/Myoblast-Six1-ChIP-Chip(GSE20150)/Homer |
| Tbx5(T-box)/HL1-Tbx5.biotin-ChIP-Seq(GSE21529)/Homer | Tbx6(T-box)/ESC-Tbx6-ChIP-Seq(GSE93524)/Homer/<br>Tbx5(T-box)/HL1-Tbx5.biotin-ChIP-Seq(GSE21529)/Homer |
| LRF(Zf)/Erythroblasts-ZBTB7A-ChIP-Seq(GSE74977)/Homer | LRF(Zf)/Erythroblasts-ZBTB7A-ChIP-Seq(GSE74977)/Homer |
| Nkx2.2(Homeobox)/NPC-Nkx2.2-ChIP-Seq(GSE61673)/Homer | Nkx2.1(Homeobox)/LungAC-Nkx2.1-ChIP-Seq(GSE43252)/Homer/<br>Nkx2.2(Homeobox)/NPC-Nkx2.2-ChIP-Seq(GSE61673)/Homer |
| ZEB1(Zf)/PDAC-ZEB1-ChIP-Seq(GSE64557)/Homer | ZEB1(Zf)/PDAC-ZEB1-ChIP-Seq(GSE64557)/Homer |
| HNF4a(NR),DR1/HepG2-HNF4a-ChIP-Seq(GSE25021)/Homer | HNF4a(NR),DR1/HepG2-HNF4a-ChIP-Seq(GSE25021)/Homer/<br>RARa(NR)/K562-RARa-ChIP-Seq(Encode)/Homer |
| TEAD(TEA)/Fibroblast-PU.1-ChIP-Seq(Unpublished)/Homer | TEAD2(TEA)/Py2T-Tead2-ChIP-Seq(GSE55709)/Homer/<br>TEAD4(TEA)/Tropoblast-Tead4-ChIP-Seq(GSE37350)/Homer/<br>TEAD(TEA)/Fibroblast-PU.1-ChIP-Seq(Unpublished)/Homer |
| EBF1(EBF)/Near-E2A-ChIP-Seq(GSE21512)/Homer | EBF1(EBF)/Near-E2A-ChIP-Seq(GSE21512)/Homer |
| Tbx21(T-box)/GM12878-TBX21-ChIP-Seq(Encode)/Homer | Tbx21(T-box)/GM12878-TBX21-ChIP-Seq(Encode)/Homer/<br>Tbet(T-box)/CD8-Tbet-ChIP-Seq(GSE33802)/Homer |
| Znf263(Zf)/K562-Znf263-ChIP-Seq(GSE31477)/Homer | Znf263(Zf)/K562-Znf263-ChIP-Seq(GSE31477)/Homer |
| CHR(?)/Hela-CellCycle-Expression/Homer | CHR(?)/Hela-CellCycle-Expression/Homer |
| Ascl1(bHLH)/NeuralTubes-Ascl1-ChIP-Seq(GSE55840)/Homer | Tcf21(bHLH)/ArterySmoothMuscle-Tcf21-ChIP-Seq(GSE61369)/Homer/<br>Atoh1(bHLH)/Cerebellum-Atoh1-ChIP-Seq(GSE22111)/Homer/<br>E2A(bHLH)/proBcell-E2A-ChIP-Seq(GSE21978)/Homer/<br>SCL(bHLH)/HPC7-Scl-ChIP-Seq(GSE13511)/Homer/<br>Ascl1(bHLH)/NeuralTubes-Ascl1-ChIP-Seq(GSE55840)/Homer/ |

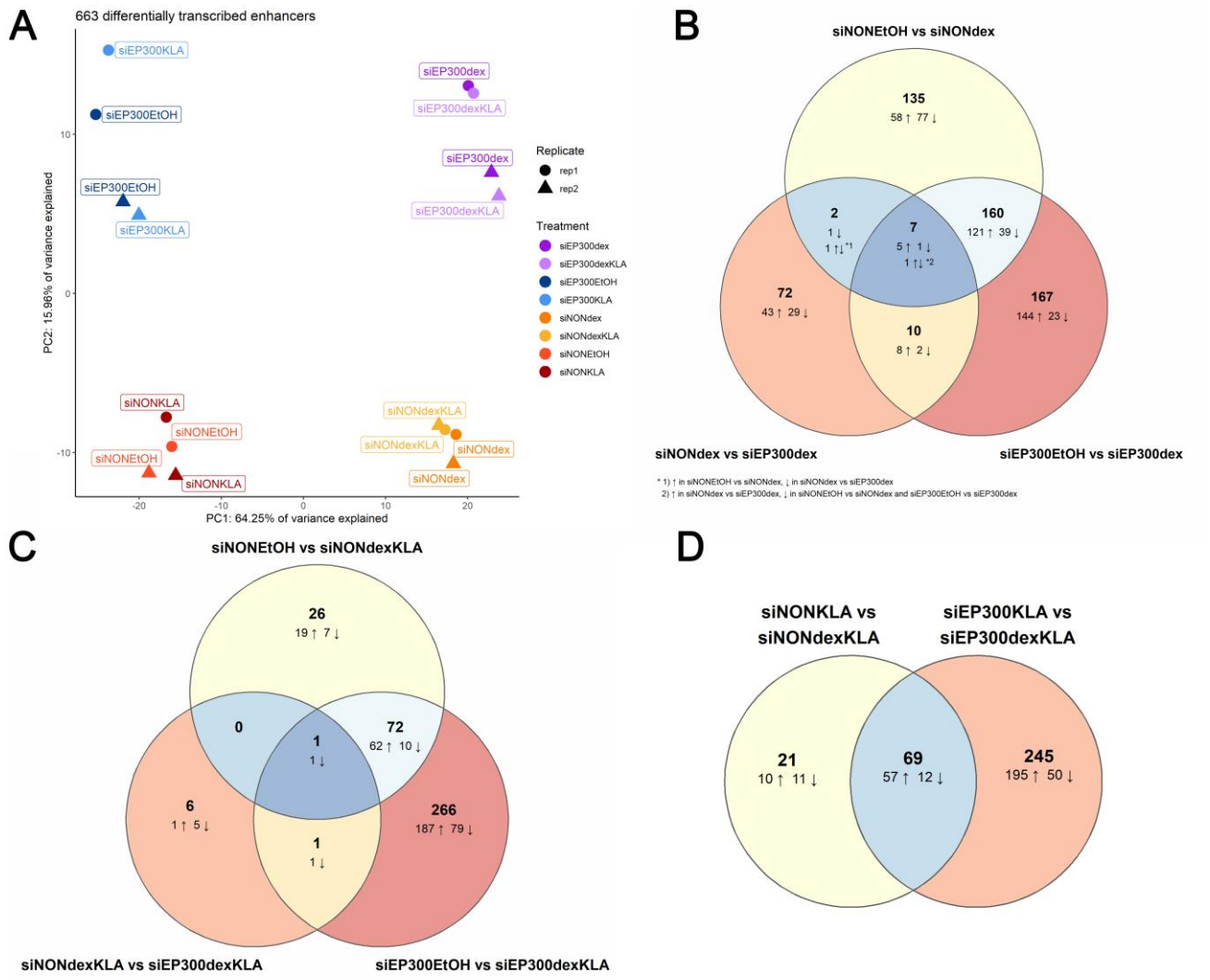
| Representative motif | Merged motifs |
|---|---|
| | HEB(bHLH)/mES-Heb-ChIP-Seq(GSE53233)/Homer<br>Ap4(bHLH)/AML-Tfap4-ChIP-Seq(GSE45738)/Homer |
| TCF4(bHLH)/SHSY5Y-TCF4-ChIP-Seq(GSE96915)/Homer | TCF4(bHLH)/SHSY5Y-TCF4-ChIP-Seq(GSE96915)/Homer<br>NeuroG2(bHLH)/Fibroblast-NeuroG2-ChIP-Seq(GSE75910)/Homer |
| BMAL1(bHLH)/Liver-Bmal1-ChIP-Seq(GSE39860)/Homer | E-box(bHLH)/Promoter/Homer/<br>USF1(bHLH)/GM12878-Usf1-ChIP-Seq(GSE32465)/Homer/<br>bHLHE40(bHLH)/HepG2-BHLHE40-ChIP-Seq(GSE31477)/Homer/<br>bHLHE41(bHLH)/proB-Bhlhe41-ChIP-Seq(GSE93764)/Homer/<br>BMAL1(bHLH)/Liver-Bmal1-ChIP-Seq(GSE39860)/Homer |
| Oct4(POU,Homeobox)/mES-Oct4-ChIP-Seq(GSE11431)/Homer | OCT4-SOX2-TCF-NANOG(POU,Homeobox,HMG)/mES-Oct4-ChIP-Seq(GSE11431)/Homer/<br>Oct2(POU,Homeobox)/Bcell-Oct2-ChIP-Seq(GSE21512)/Homer/<br>Oct4(POU,Homeobox)/mES-Oct4-ChIP-Seq(GSE11431)/Homer |
| AP-2alpha(AP2)/Hela-AP2alpha-ChIP-Seq(GSE31477)/Homer | AP-2alpha(AP2)/Hela-AP2alpha-ChIP-Seq(GSE31477)/Homer/<br>AP-2gamma(AP2)/MCF7-TFAP2C-ChIP-Seq(GSE21234)/Homer |
| AP-1(bZIP)/ThioMac-PU.1-ChIP-Seq(GSE21512)/Homer | Fra2(bZIP)/Striatum-Fra2-ChIP-Seq(GSE43429)/Homer/<br>Jun-AP1(bZIP)/K562-cJun-ChIP-Seq(GSE31477)/Homer/<br>AP-1(bZIP)/ThioMac-PU.1-ChIP-Seq(GSE21512)/Homer/<br>JunB(bZIP)/DendriticCells-Junb-ChIP-Seq(GSE36099)/Homer/<br>Atf3(bZIP)/GBM-ATF3-ChIP-Seq(GSE33912)/Homer/<br>BATF(bZIP)/Th17-BATF-ChIP-Seq(GSE39756)/Homer/<br>Fra1(bZIP)/BT549-Fra1-ChIP-Seq(GSE46166)/Homer/<br>Fosl2(bZIP)/3T3L1-Fosl2-ChIP-Seq(GSE56872)/Homer |
| Lhx3(Homeobox)/Neuron-Lhx3-ChIP-Seq(GSE31456)/Homer | Pax7(Paired,Homeobox),longest/Myoblast-Pax7-ChIP-Seq(GSE25064)/Homer/<br>Lhx3(Homeobox)/Neuron-Lhx3-ChIP-Seq(GSE31456)/Homer/<br>Lhx1(Homeobox)/EmbryoCarcinoma-Lhx1-ChIP-Seq(GSE70957)/Homer |
| PPARE(NR),DR1/3T3L1-Pparg-ChIP-Seq(GSE13511)/Homer | PPARE(NR),DR1/3T3L1-Pparg-ChIP-Seq(GSE13511)/Homer/<br>Nur77(NR)/K562-NR4A1-ChIP-Seq(GSE31363)/Homer |
| MyoD(bHLH)/Myotube-MyoD-ChIP-Seq(GSE21614)/Homer | MyoD(bHLH)/Myotube-MyoD-ChIP-Seq(GSE21614)/Homer/<br>Tcf12(bHLH)/GM12878-Tcf12-ChIP-Seq(GSE32465)/Homer |
| CLOCK(bHLH)/Liver-Clock-ChIP-Seq(GSE39860)/Homer | TFE3(bHLH)/MEF-TFE3-ChIP-Seq(GSE75757)/Homer/<br>CLOCK(bHLH)/Liver-Clock-ChIP-Seq(GSE39860)/Homer |
| Rfx6(HTH)/Min6b1-Rfx6.HA-ChIP-Seq(GSE62844)/Homer | Rfx6(HTH)/Min6b1-Rfx6.HA-ChIP-Seq(GSE62844)/Homer |
| Brn1(POU,Homeobox)/NPC-Brn1-ChIP-Seq(GSE35496)/Homer | Oct6(POU,Homeobox)/NPC-Pou3f1-ChIP-Seq(GSE35496)/Homer/<br>Brn1(POU,Homeobox)/NPC-Brn1-ChIP-Seq(GSE35496)/Homer |

| Representative motif | Merged motifs |
|---|---|
| SCRT1(Zf)/HEK293-SCRT1.eGFP-ChIP-Seq(Encode)/Homer | SCRT1(Zf)/HEK293-SCRT1.eGFP-ChIP-Seq(Encode)/Homer |
| Rbpj1(?)/Panc1-Rbpj1-ChIP-Seq(GSE47459)/Homer | ZNF143\|STAF(Zf)/CUTLL-ZNF143-ChIP-Seq(GSE29600)/Homer/<br>Rbpj1(?)/Panc1-Rbpj1-ChIP-Seq(GSE47459)/Homer |
| STAT6(Stat)/Macrophage-Stat6-ChIP-Seq(GSE38377)/Homer | STAT5(Stat)/mCD4+-Stat5-ChIP-Seq(GSE12346)/Homer/<br>STAT6(Stat)/Macrophage-Stat6-ChIP-Seq(GSE38377)/Homer |
| Smad3(MAD)/NPC-Smad3-ChIP-Seq(GSE36673)/Homer | Smad3(MAD)/NPC-Smad3-ChIP-Seq(GSE36673)/Homer<br>Smad2(MAD)/ES-SMAD2-ChIP-Seq(GSE29422)/Homer |
| GATA3(Zf)/iTreg-Gata3-ChIP-Seq(GSE20898)/Homer | GATA3(Zf),DR8/iTreg-Gata3-ChIP-Seq(GSE20898)/Homer/<br>Gata6(Zf)/HUG1N-GATA6-ChIP-Seq(GSE51936)/Homer/<br>Gata4(Zf)/Heart-Gata4-ChIP-Seq(GSE35151)/Homer/<br>GATA3(Zf)/iTreg-Gata3-ChIP-Seq(GSE20898)/Homer |
| Esrrb(NR)/mES-Esrrb-ChIP-Seq(GSE11431)/Homer | Esrrb(NR)/mES-Esrrb-ChIP-Seq(GSE11431)/Homer |
| Nkx6.1(Homeobox)/Islet-Nkx6.1-ChIP-Seq(GSE40975)/Homer | Nanog(Homeobox)/mES-Nanog-ChIP-Seq(GSE11724)/Homer/<br>Nkx6.1(Homeobox)/Islet-Nkx6.1-ChIP-Seq(GSE40975)/Homer |
| CDX4(Homeobox)/ZebrafishEmbryos-Cdx4.Myc-ChIP-Seq(GSE48254)/Homer | Hoxa11(Homeobox)/ChickenMSG-Hoxa11.Flag-ChIP-Seq(GSE86088)/Homer/<br>CDX4(Homeobox)/ZebrafishEmbryos-Cdx4.Myc-ChIP-Seq(GSE48254)/Homer/<br>Hoxd11(Homeobox)/ChickenMSG-Hoxd11.Flag-ChIP-Seq(GSE86088)/Homer |
| MITF(bHLH)/MastCells-MITF-ChIP-Seq(GSE48085)/Homer | MITF(bHLH)/MastCells-MITF-ChIP-Seq(GSE48085)/Homer |
| OCT:OCT-short(POU,Homeobox)/NPC-OCT6-ChIP-Seq(GSE43916)/Homer | OCT:OCT-short(POU,Homeobox)/NPC-OCT6-ChIP-Seq(GSE43916)/Homer/<br>Pit1+1bp(Homeobox)/GCrat-Pit1-ChIP-Seq(GSE58009)/Homer |
| Chop(bZIP)/MEF-Chop-ChIP-Seq(GSE35681)/Homer | CEBP:CEBP(bZIP)/MEF-Chop-ChIP-Seq(GSE35681)/Homer/<br>Chop(bZIP)/MEF-Chop-ChIP-Seq(GSE35681)/Homer/<br>Atf4(bZIP)/MEF-Atf4-ChIP-Seq(GSE35681)/Homer |
| Pit1(Homeobox)/GCrat-Pit1-ChIP-Seq(GSE58009)/Homer | OCT:OCT(POU,Homeobox)/NPC-OCT6-ChIP-Seq(GSE43916)/Homer/<br>Pit1(Homeobox)/GCrat-Pit1-ChIP-Seq(GSE58009)/Homer |
| Fli1(ETS)/CD8-FLI-ChIP-Seq(GSE20898)/Homer | ETV4(ETS)/HepG2-ETV4-ChIP-Seq(ENCODE)/Homer/<br>Elk1(ETS)/Hela-Elk1-ChIP-Seq(GSE31477)/Homer/<br>Elk4(ETS)/Hela-Elk4-ChIP-Seq(GSE31477)/Homer/<br>Fli1(ETS)/CD8-FLI-ChIP-Seq(GSE20898)/Homer |
| Srebp1a(bHLH)/HepG2-Srebp1a-ChIP-Seq(GSE31477)/Homer | Srebp2(bHLH)/HepG2-Srebp2-ChIP-Seq(GSE31477)/Homer/<br>Srebp1a(bHLH)/HepG2-Srebp1a-ChIP-Seq(GSE31477)/Homer |

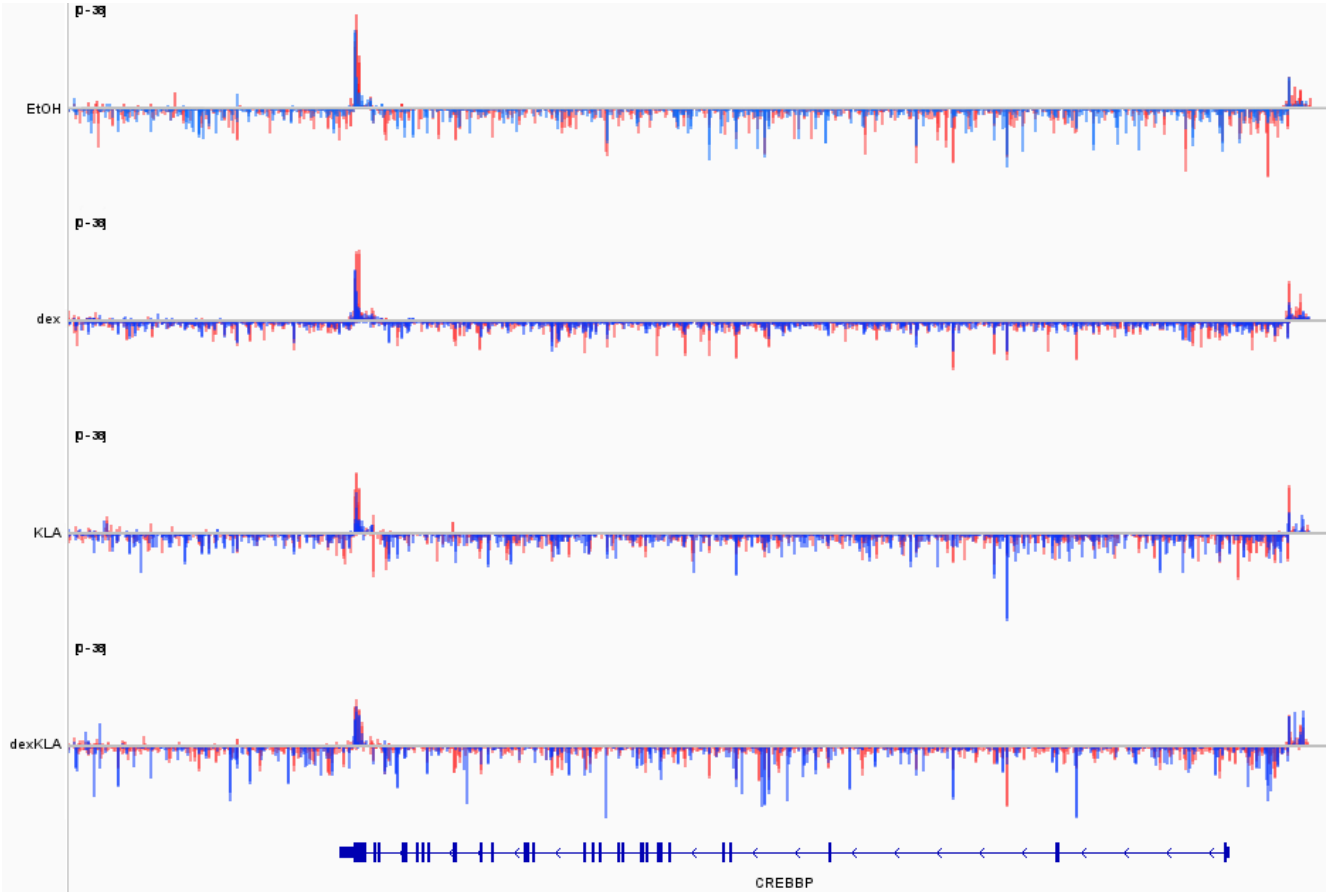
| Representative motif | Merged motifs |
|---|---|
| GABPA(ETS)/Jurkat-GABPa-ChIP-Seq(GSE17954)/Homer | ETS(ETS)/Promoter/Homer/<br>ETV1(ETS)/GIST48-ETV1-ChIP-Seq(GSE22441)/Homer/<br>GABPA(ETS)/Jurkat-GABPa-ChIP-Seq(GSE17954)/Homer/<br>ELF1(ETS)/Jurkat-ELF1-ChIP-Seq(SRA014231)/Homer |
| ETS1(ETS)/Jurkat-ETS1-ChIP-Seq(GSE17954)/Homer | ERG(ETS)/VCaP-ERG-ChIP-Seq(GSE14097)/Homer/<br>ETS1(ETS)/Jurkat-ETS1-ChIP-Seq(GSE17954)/Homer/<br>Etv2(ETS)/ES-ER71-ChIP-Seq(GSE59402)/Homer(0.967) |
| FOXA1:AR(Forkhead,NR)/LNCAP-AR-ChIP-Seq(GSE27824)/Homer | FOXA1:AR(Forkhead,NR)/LNCAP-AR-ChIP-Seq(GSE27824)/Homer |
| Pax8(Paired,Homeobox)/Thyroid-Pax8-ChIP-Seq(GSE26938)/Homer | PAX5(Paired,Homeobox)/GM12878-PAX5-ChIP-Seq(GSE32465)/Homer/<br>Pax8(Paired,Homeobox)/Thyroid-Pax8-ChIP-Seq(GSE26938)/Homer |
| EWS:ERG-fusion(ETS)/CADO_ES1-EWS:ERG-ChIP-Seq(SRA014231)/Homer | EWS:FLI1-fusion(ETS)/SK_N_MC-EWS:FLI1-ChIP-Seq(SRA014231)/Homer/<br>EWS:ERG-fusion(ETS)/CADO_ES1-EWS:ERG-ChIP-Seq(SRA014231)/Homer |
| Foxh1(Forkhead)/hESC-FOXH1-ChIP-Seq(GSE29422)/Homer | RUNX(Runt)/HPC7-Runx1-ChIP-Seq(GSE22178)/Homer/<br>Foxh1(Forkhead)/hESC-FOXH1-ChIP-Seq(GSE29422)/Homer |
| ZNF519(Zf)/HEK293-ZNF519.GFP-ChIP-Seq(GSE58341)/Homer | ZNF519(Zf)/HEK293-ZNF519.GFP-ChIP-Seq(GSE58341)/Homer |
| c-Myc(bHLH)/mES-cMyc-ChIP-Seq(GSE11431)/Homer | Max(bHLH)/K562-Max-ChIP-Seq(GSE31477)/Homer/<br>n-Myc(bHLH)/mES-nMyc-ChIP-Seq(GSE11431)/Homer/<br>c-Myc(bHLH)/mES-cMyc-ChIP-Seq(GSE11431)/Homer/<br>c-Myc(bHLH)/LNCAP-cMyc-ChIP-Seq(Unpublished)/Homer |



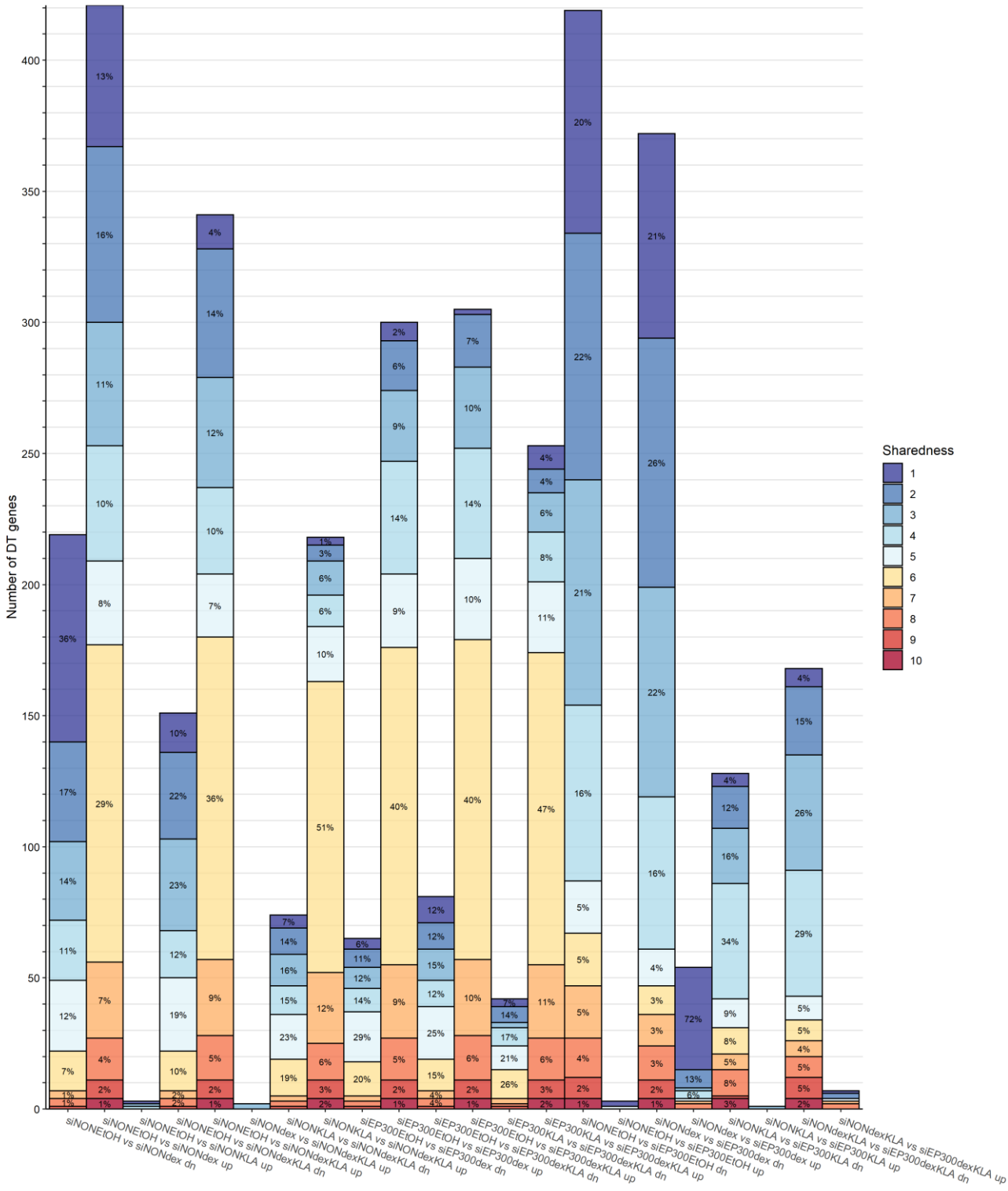

Supplementary Figure 14. The same enrichment ratio plot as shown beside the heatmap in Fig. S13 expanded slightly on the x-axis.




**Supplementary Figure 15. DT enhancers are mostly shared between siNON vs siNON and siEP300 vs siEP300 comparisons.** Related to Fig. 5, especially Fig. 5B-D. **A)** PCA plot of all DT enhancers shows a similar pattern as the PCA plot for DT genes (Fig. 4B) with KLA clustering with EtOH and dexKLA clustering with dex, indicating the lack of KLA response. **B-C)** Venn diagrams of DT enhancers show the overlap of enhancers in the three different siRNA conditions within dex (B) and dexKLA (C) treatments. Many enhancers are shared between the siNON vs siNON and siEP300 vs siEP300 comparisons while many are also unique to these treatments. Due to the low number of DT enhancers in the siNON vs siEP300 comparisons, these comparisons have little overlap with the others. **D)** Similarly to EtOH vs dex and EtOH vs dexKLA, many KLA vs dexKLA DT enhancers are shared between siNON vs siNON and siEP300 vs siEP300. The bolded numbers indicate total counts, the smaller numbers indicate how many of these are up- (↑) and downregulated (↓).

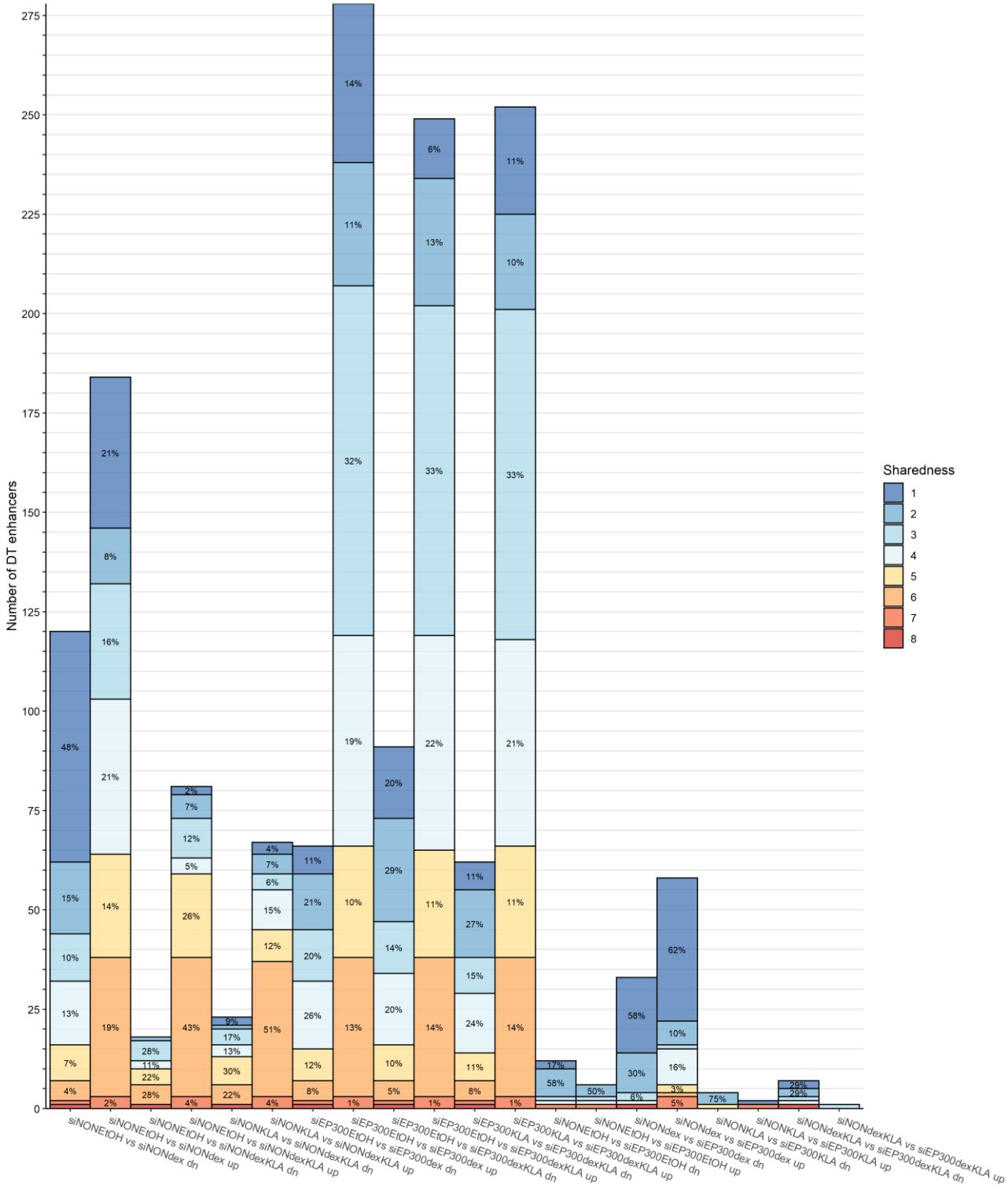



**Supplementary Figure 16. The expression of CBP is not transcriptionally activated in response to p300 knockdown.** At the bottom, *CREBBP* coding for CBP is shown and above it is shown the PRO-seq signal from each hormone treatment. Transcription from the + strand is shown as peaks going up and transcription from the – strand is shown as peaks going down. The blue peaks are from the siNON and the red peaks are from the siEP300 condition. The scale of the peaks is the same for both strands. *CREBBP* is transcribed both in the presence and absence of p300 and transcription is not significantly induced after p300 knockdown.



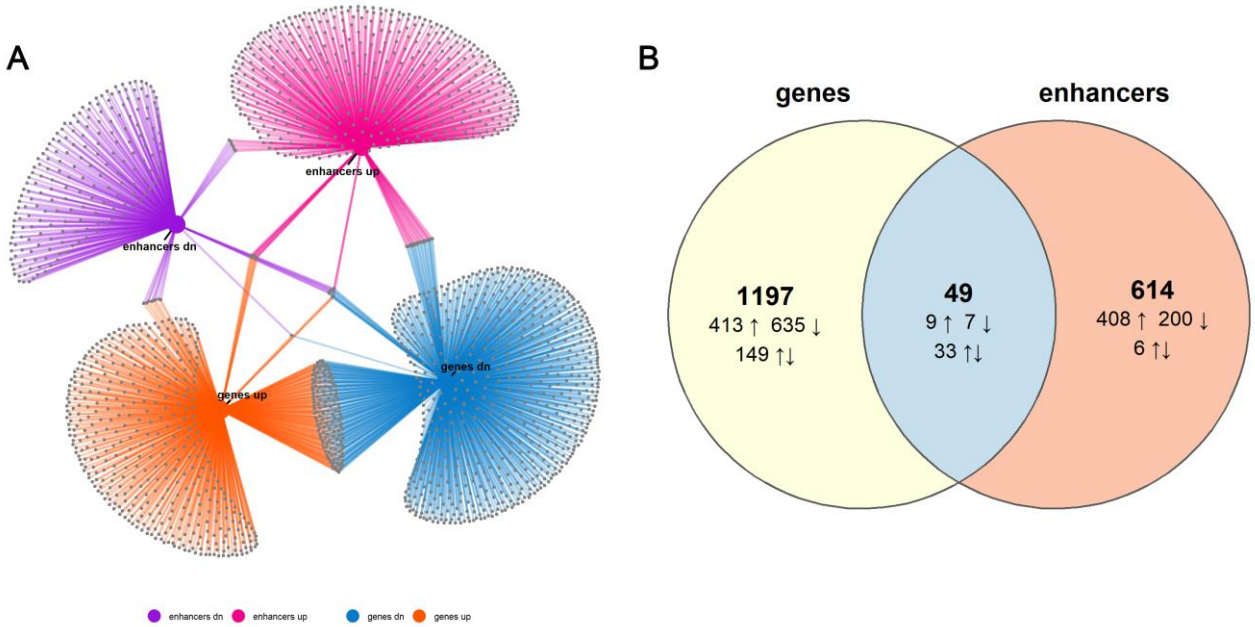

**Supplementary Figure 17. siNON vs siNON and siEP300 vs siEP300 upregulated genes are mostly shared in all 6 of these comparisons.** Related to Fig. 6A. The stacked barplot shows the number of DT genes on the y-axis for all comparisons (x-axis) with up- and downregulated genes separately. The colouring of the bars and the percentages shown on each colour indicate what portion of the DT genes in a treatment were shared in how many comparisons, i.e. a sharedness of 1 means the gene is unique to that comparison. If there were only one or two DT genes in a comparison shared to any degree, the percentage was not plotted to maintain readability. Approximately a third to a half of the upregulated genes in siNON vs siNON and

siEP300 vs siEP300 comparisons modeling the dex effect are shared in 6 comparisons, which are most likely these six comparisons, indicating the high similarity of these comparisons, while the downregulated genes were less shared as a minority of these genes had a sharedness of ≥5.




**Supplementary Figure 18. Upregulated enhancers are largely shared.** Related to Fig. 6C. The stacked barplot shows the number of DT enhancers on the y-axis for all comparisons (x-axis) with up- and downregulated enhancers separately. The colouring of the bars and the percentages shown on each colour indicate what portion of the DT enhancers in a treatment were shared in how many comparisons, i.e. a sharedness of 1

means the enhancer is unique to that comparison. If there was only one DT enhancer in a comparison shared to any degree, the percentage was not plotted to maintain readability. The earlier observations of there being very few DT enhancers in the siNON vs siEP300 comparisons and conversely many DT enhancers in the siEP300 vs siEP300 and siNONEtOH vs siNONdex comparisons can be confirmed here. Similarly to DT genes, the upregulated enhancers in the six siNON vs siNON and siEP300 vs siEP300 comparisons have similarly sized sets of DT enhancers with a sharedness of 6, suggesting that these could be shared in all six of these comparisons. Furthermore, of the three siEP300 vs siEP300 comparisons, the largest portion of upregulated enhancers have a sharedness of 3, which indicates a high similarity between these treatments. As for the DT genes, the downregulated enhancers are less shared than the upregulated ones.




**Supplementary Figure 19. DT enhancers do not regulate the DT genes.** A network map (A) and Venn diagram (B) of all DT genes and genes associated with DT enhancers in all the comparisons illustrate that the overlap between these two groups is very small and that the four categories are mostly separate except for the DT genes which have a notable proportion of genes that are both up- and downregulated. The grey dots in the network map represent DT genes and genes associated with DT enhancers. The bolded numbers in the Venn diagram indicate total counts, the smaller numbers indicate how many of these are up- (↑) and downregulated (↓).

**Supplementary Table 3. The 70 significantly enriched pathways in alphabetical order with the databases they are from and their unique identifiers in these databases.**

| Pathway description | Term ID | Database |
|---|---|---|
| actin cytoskeleton organization | GO:0030036 | GO Biological Processes |
| actin filament organization | GO:0007015 | GO Biological Processes |
| actin filament-based process | GO:0030029 | GO Biological Processes |
| blood circulation | GO:0008015 | GO Biological Processes |
| Burn wound healing | WP5055 | WikiPathways |
| CDC42 GTPase cycle | R-HSA-9013148 | Reactome Gene Sets |
| cell activation | GO:0001775 | GO Biological Processes |

| Pathway description | Term ID | Database |
|---|---|---|
| cell junction organization | GO:0034330 | GO Biological Processes |
| cell morphogenesis | GO:0000902 | GO Biological Processes |
| cell-cell adhesion | GO:0098609 | GO Biological Processes |
| cellular response to hormone stimulus | GO:0032870 | GO Biological Processes |
| cellular response to inorganic substance | GO:0071241 | GO Biological Processes |
| cellular response to metal ion | GO:0071248 | GO Biological Processes |
| cellular response to organic cyclic compound | GO:0071407 | GO Biological Processes |
| chemotaxis | GO:0006935 | GO Biological Processes |
| circulatory system process | GO:0003013 | GO Biological Processes |
| Cytokine Signaling in Immune system | R-HSA-1280215 | Reactome Gene Sets |
| DNA repair pathways, full network | WP4946 | WikiPathways |
| Ectoderm differentiation | WP2858 | WikiPathways |
| enzyme-linked receptor protein signaling pathway | GO:0007167 | GO Biological Processes |
| Extracellular matrix organization | R-HSA-1474244 | Reactome Gene Sets |
| Gastrin signaling pathway | WP4659 | WikiPathways |
| Glucocorticoid receptor pathway | WP2880 | WikiPathways |
| Herpes simplex virus 1 infection | hsa05168 | KEGG Pathway |
| Human T-cell leukemia virus 1 infection | hsa05166 | KEGG Pathway |
| IL-18 signaling pathway | WP4754 | WikiPathways |
| Kaposi sarcoma-associated herpesvirus infection | hsa05167 | KEGG Pathway |
| Mucin type O-glycan biosynthesis | hsa00512 | KEGG Pathway |
| negative regulation of osteoblast differentiation | GO:0045668 | GO Biological Processes |
| Network map of SARS-CoV-2 signaling pathway | WP5115 | WikiPathways |
| Neuronal System | R-HSA-112316 | Reactome Gene Sets |
| NIK/NF-kappaB signaling | GO:0038061 | GO Biological Processes |
| Nuclear receptors meta-pathway | WP2882 | WikiPathways |
| nucleus organization | GO:0006997 | GO Biological Processes |
| Osteoclast differentiation | hsa04380 | KEGG Pathway |
| Pathways in cancer | hsa05200 | KEGG Pathway |
| PID AVB3 OPN PATHWAY | M63 | Canonical Pathways |
| PID ECADHERIN STABILIZATION PATHWAY | M232 | Canonical Pathways |
| PID FRA PATHWAY | M65 | Canonical Pathways |
| PID INTEGRIN1 PATHWAY | M18 | Canonical Pathways |
| positive regulation of activin receptor signaling pathway | GO:0032927 | GO Biological Processes |
| positive regulation of apoptotic process | GO:0043065 | GO Biological Processes |
| positive regulation of cell adhesion | GO:0045785 | GO Biological Processes |
| positive regulation of cell death | GO:0010942 | GO Biological Processes |
| positive regulation of cell migration | GO:0030335 | GO Biological Processes |

| Pathway description | Term ID | Database |
|---|---|---|
| positive regulation of programmed cell death | GO:0043068 | GO Biological Processes |
| protein phosphorylation | GO:0006468 | GO Biological Processes |
| RANKL/RANK signaling pathway | WP2018 | WikiPathways |
| regulation of anatomical structure size | GO:0090066 | GO Biological Processes |
| regulation of cell adhesion | GO:0030155 | GO Biological Processes |
| regulation of endothelial cell apoptotic process | GO:2000351 | GO Biological Processes |
| regulation of epithelial cell apoptotic process | GO:1904035 | GO Biological Processes |
| regulation of ossification | GO:0030278 | GO Biological Processes |
| regulation of osteoblast differentiation | GO:0045667 | GO Biological Processes |
| regulation of pathway-restricted SMAD protein phosphorylation | GO:0060393 | GO Biological Processes |
| Regulation of signaling by NODAL | R-HSA-1433617 | Reactome Gene Sets |
| regulation of small GTPase mediated signal transduction | GO:0051056 | GO Biological Processes |
| reproductive structure development | GO:0048608 | GO Biological Processes |
| reproductive system development | GO:0061458 | GO Biological Processes |
| response to hormone | GO:0009725 | GO Biological Processes |
| response to xenobiotic stimulus | GO:0009410 | GO Biological Processes |
| Signaling by Interleukins | R-HSA-449147 | Reactome Gene Sets |
| skeletal system development | GO:0001501 | GO Biological Processes |
| Spinal cord injury | WP2431 | WikiPathways |
| supramolecular fiber organization | GO:0097435 | GO Biological Processes |
| taxis | GO:0042330 | GO Biological Processes |
| TNF signaling pathway | hsa04668 | KEGG Pathway |
| TNF-related weak inducer of apoptosis (TWEAK) signaling pathway | WP2036 | WikiPathways |
| tube morphogenesis | GO:0035239 | GO Biological Processes |
| vasculature development | GO:0001944 | GO Biological Processes |

# Appendix 2. Supplementary Code

```
################################################################################
## ANALYSIS OF DIFFERENTIALLY EXPRESSED GENES AND ENHANCERS FROM PRO-SEQ DATA ##
################################################################################

#under each title genes and enhancers are handled the same if not mentioned otherwise
#with the genes first and the genes second
#except for motif analysis which was only done from enhancers
#and pathway analysis and overlap of genes and enhancers which combined data from
#both genes and enhancers

#the steps for the genes are commented more extensively
#since the steps are the same for them and enhancers

######################
## Set up workspace ##
######################

#install necessary packages and any dependencies they may have
install.packages("ggplot2", dependencies = TRUE) #for drawing plots
install.packages("ggnetwork", dependencies = TRUE) #for drawing network maps
install.packages("network", dependencies = TRUE) #for handling data for network maps
install.packages("data.table", dependencies = TRUE) #for handling data
install.packages("circlize", dependencies = TRUE) #for creating colour gradients
install.packages("ggrepel", dependencies = TRUE) #for plotting non-overlapping labels
install.packages("gridExtra", dependencies = TRUE) #for plotting
install.packages("ggVennDiagram", dependencies = TRUE) #for drawing venn diagrams
install.packages("RColorBrewer", dependencies = TRUE) #for plot colour palettes
install.packages("readxl", dependencies = TRUE) #for reading in Excel files
#for some packages, they must be downloaded from BiocManager
#with :: the package to use the function from can be defined
if (!require("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
BiocManager::install("ComplexHeatmap") #for drawing heatmaps
BiocManager::install("universalmotif") #for merging and visualizing motifs


#load the packages
library(ggplot2)
library(ggnetwork)
library(network)
library(data.table)
library(ComplexHeatmap)
library(ggrepel)
library(circlize)
library(gridExtra)
library(ggVennDiagram)
library(RColorBrewer)
library(universalmotif)
library(readxl)

###############
## Functions ##
###############

#define a function for use later:
```

```
#use sep to separate items on one column x and return the n:th element
get_nth_of_list <- function(x, sep, n){
  list_all <- do.call(rbind, base::strsplit(as.character(x), split = sep))
  return(list_all[,n])
}

##################
## Read in data ##
##################

#######
##GENES

#load data and save it as a data.table
#the file has been placed in the same folder as the R project
#which is the workspace
dt <- fread("analysis.A549_PROseq_siNON-vs-siEP300_noalt.pc1.genes.plus.hg38.pos", fill = TRUE, sep
= "\t")

#clean up column names
#see what they are
colnames(dt)
#the first one includes information about the command used to generate the data
#which is unnecessary here
colnames(dt)[1] <- "Transcript"
#replace special characters and spaces between siRNA condition and hormone treatment
new_colnames <- colnames(dt)
new_colnames <- gsub(" ", "_", new_colnames)
new_colnames <- gsub("-", "", new_colnames)
new_colnames <- gsub("[.]", "", new_colnames)
new_colnames <- gsub("_[(].*", "", new_colnames)
new_colnames <- gsub("siNON_", "siNON", new_colnames)
new_colnames <- gsub("siEP300_", "siEP300", new_colnames)
#save them as the new column names
colnames(dt) <- new_colnames

#the raw read counts and differential gene expression results were merged together
#to form one table and this caused some rows to be duplicated, so
#remove duplicate columns
dt <- dt[,!109:115]

#the chr column has information about what chromosome the transcript is from
#define if the transcript is from a "normal" chromosome
#fifelse can be used to define a logical condition
#and then the function is provided with the value to output when the condition is fulfilled
#and finally the value to output when the condition is not fulfilled
#%in% can be used to seek if elements in the object before %in% are found in the object after %in%
# := is an operator used by data.table to add information to columns
#or new columns if one by the name given does not exist yet
dt[, is_normal_chr := fifelse(chr %in% c(paste0("chr", 1:23), c("chrX", "chrY")), 1, 0)]
table(dt$is_normal_chr)

#Annotation/Divergence column has all the different names associated to that transcript
#define unique gene symbols by keeping only the first one
head(dt$`Annotation/Divergence`)
#they are divided by | so use that to split and save as a new column
dt[, Gene := get_nth_of_list(dt$`Annotation/Divergence`, sep = "[|]", n = 1)]

```

```
#this is how it looks now:
colnames(dt)
dim(dt)


###########
##ENHANCERS

#load data
dt_enh <- fread("analysis.consensus_enhancers_range100.A549_dREG.BL.pc1.both.pos", fill = TRUE, sep
= "\t")

#clean up column names
colnames(dt_enh)
colnames(dt_enh)[1] <- "Transcript"
new_colnames <- colnames(dt_enh)
new_colnames <- gsub(" ", "_", new_colnames)
new_colnames <- gsub("-", "", new_colnames)
new_colnames <- gsub("[.]", "", new_colnames)
new_colnames <- gsub("_[(].*", "", new_colnames)
new_colnames <- gsub("siNON_", "siNON", new_colnames)
new_colnames <- gsub("siEP300_", "siEP300", new_colnames)
colnames(dt_enh) <- new_colnames

#remove duplicate columns
dt_enh <- dt_enh[,!109:115]

#define if transcript is from "normal" chromosome
dt_enh[, is_normal_chr := fifelse(chr %in% c(paste0("chr", 1:23), c("chrX", "chrY")), 1, 0)]
table(dt_enh$is_normal_chr)

#define unique symbols for genes associated with enhancers
head(dt_enh$`Annotation/Divergence`)
dt_enh[, Gene := get_nth_of_list(dt_enh$`Annotation/Divergence`, sep = "[|]", n = 1)]

#this is how it looks now:
colnames(dt_enh)
dim(dt_enh)



##########################################################
## Expression statistics and expressed genes/enhancers ##
##########################################################

#######
##GENES

#define tpm  rep columns containing the read counts
#all these columns contain the word "TPM" so that pattern can be used to get the columns with grep()
tpm_rep_columns <- grep("rep", grep("TPM",colnames(dt), value = TRUE), value = TRUE)
#see if there are any NAs
#in data.tables .SD is used to refer to the columns to perform the operation on
#and .SDcols defines the columns (possibly as an external object)
paste0("number of NAs in tpm columns: ", sum(is.na(as.matrix(dt[,.SD, .SDcols = tpm_rep_columns]))))

#calculate statistics using tpm rep columns
#maximums, means, medians, sums, and standard deviations (sd)
```

```
#and add these as new columns
dt[, max_tpm := matrixStats::rowMaxs(as.matrix(.SD)), .SDcols = tpm_rep_columns]
dt[, mean_tpm := matrixStats::rowMeans2(as.matrix(.SD)), .SDcols = tpm_rep_columns]
dt[, median_tpm := matrixStats::rowMedians(as.matrix(.SD)), .SDcols = tpm_rep_columns]
dt[, sum_tpm := matrixStats::rowSums2(as.matrix(.SD)), .SDcols = tpm_rep_columns]
dt[, sd_tpm := matrixStats::rowSds(as.matrix(.SD)), .SDcols = tpm_rep_columns]
dt[,lapply(.SD, sum), .SDcols = tpm_rep_columns]


#define cutoff for selecting expressed genes
#here, cutoff is the max median rather than a hard cutoff
#so that the cutoff is automatically adjusted to the data
tpm_cutoff <- max(dt[,lapply(.SD, median), .SDcols = tpm_rep_columns])
dt[, is_expressed := fifelse(max_tpm > tpm_cutoff, 1, 0)]
paste0("using maximum tpm cut-off of ", tpm_cutoff, " the number of unique expressed genes is: ",
sum(dt[!duplicated(Gene) == TRUE, is_expressed]))


#mark unique genes (is_unique)
#sort according to gene symbol and descending max_tpm
setorder(dt, Gene, -max_tpm)
#keep only the gene (transcript) that had highest max_tpm
#this is now the topmost one because of how the table was sorted by descending max_tpm
dt[, is_unique := fifelse(!duplicated(dt, by ="Gene"),1,0)]
paste0("number of unique gene symbols is: ", sum(dt$is_unique))
paste0("number of unique & expressed genes is: ", dim(dt[is_unique == 1 & is_expressed == 1,])[1])


###########
##ENHANCERS

#define tpm  rep columns
tpm_rep_columns <- grep("rep", grep("TPM",colnames(dt_enh), value = TRUE), value = TRUE)
paste0("number   of   NAs   in   tpm   columns:   ",   sum(is.na(as.matrix(dt_enh[,.SD,   .SDcols   =
tpm_rep_columns]))))

#calculate statistics using tpm rep columns
dt_enh[, max_tpm := matrixStats::rowMaxs(as.matrix(.SD)), .SDcols = tpm_rep_columns]
dt_enh[, mean_tpm := matrixStats::rowMeans2(as.matrix(.SD)), .SDcols = tpm_rep_columns]
dt_enh[, median_tpm := matrixStats::rowMedians(as.matrix(.SD)), .SDcols = tpm_rep_columns]
dt_enh[, sum_tpm := matrixStats::rowSums2(as.matrix(.SD)), .SDcols = tpm_rep_columns]
dt_enh[, sd_tpm := matrixStats::rowSds(as.matrix(.SD)), .SDcols = tpm_rep_columns]
dt_enh[,lapply(.SD, sum), .SDcols = tpm_rep_columns]


#define cutoff for selecting expressed enhancers
#here we do not want any cutoff as enhancers have already been filtered during their discovery from
the data
#so the cutoff is set to 0
tpm_cutoff <- 0
dt_enh[, is_expressed := fifelse(max_tpm > tpm_cutoff, 1, 0)]
paste0("using maximum tpm cut-off of ", tpm_cutoff, " the number of unique expressed enhancers is: ",
sum(dt_enh[!duplicated(Gene) == TRUE, is_expressed]))


#mark unique enhancers (is_unique)
setorder(dt_enh, Gene, -max_tpm)
dt_enh[, is_unique := fifelse(!duplicated(dt_enh, by ="Gene"),1,0)]
```

```
paste0("number of unique gene symbols is: ", sum(dt_enh$is_unique))
paste0("number of unique & expressed genes is: ", dim(dt_enh[is_unique == 1 & is_expressed == 1,])[1])

##########################################
## Differential gene/enhancer expression ##
##########################################

#technically differential gene/enhancer expression has already been done
#since the log fold change (logfc) values and p-values already exist in the data
#here the data is filtered based on the logfc and p-value cutoff
#and whether gene/enhancer is up- (up) or downregulated (dn) is defined
#based on whether logfc is positive or negative

#######
##GENES

#define differentially expressed (DE) genes
#set the cutoffs
pvalue_cutoff = 0.01
logfc_cutoff = 0.5

#list all the different comparisons
#not all of them are meaningful, we only need to filter those ones that are
grep("vs", grep("adj_pvalue", colnames(dt), value = TRUE), value = TRUE)

#filtering is done using fifelse and inputting the output into a new column
#1 for yes, 0 for no

#siNON vs siNON comparisons
dt[,  siNONEtOH_vs_siNONdex_up  :=  fifelse(siNONEtOH_vs_siNONdex_adj_pvalue  <  pvalue_cutoff  &
siNONEtOH_vs_siNONdex_Log2_Fold_Change > logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]
dt[,  siNONEtOH_vs_siNONdex_dn  :=  fifelse(siNONEtOH_vs_siNONdex_adj_pvalue  <  pvalue_cutoff  &
siNONEtOH_vs_siNONdex_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]
dt[,  siNONEtOH_vs_siNONKLA_up  :=  fifelse(siNONEtOH_vs_siNONKLA_adj_pvalue  <  pvalue_cutoff  &
siNONEtOH_vs_siNONKLA_Log2_Fold_Change > logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]
dt[,  siNONEtOH_vs_siNONKLA_dn  :=  fifelse(siNONEtOH_vs_siNONKLA_adj_pvalue  <  pvalue_cutoff  &
siNONEtOH_vs_siNONKLA_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]
dt[, siNONEtOH_vs_siNONdexKLA_up := fifelse(siNONEtOH_vs_siNONdexKLA_adj_pvalue < pvalue_cutoff &
siNONEtOH_vs_siNONdexKLA_Log2_Fold_Change > logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]
dt[, siNONEtOH_vs_siNONdexKLA_dn := fifelse(siNONEtOH_vs_siNONdexKLA_adj_pvalue < pvalue_cutoff &
siNONEtOH_vs_siNONdexKLA_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 & is_expressed == 1, 1,
0)]
dt[, siNONdex_vs_siNONdexKLA_up := fifelse(siNONdex_vs_siNONdexKLA_adj_pvalue < pvalue_cutoff &
siNONdex_vs_siNONdexKLA_Log2_Fold_Change > logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]
dt[, siNONdex_vs_siNONdexKLA_dn := fifelse(siNONdex_vs_siNONdexKLA_adj_pvalue < pvalue_cutoff &
siNONdex_vs_siNONdexKLA_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]
dt[, siNONKLA_vs_siNONdexKLA_up := fifelse(siNONKLA_vs_siNONdexKLA_adj_pvalue < pvalue_cutoff &
siNONKLA_vs_siNONdexKLA_Log2_Fold_Change > logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]
dt[, siNONKLA_vs_siNONdexKLA_dn := fifelse(siNONKLA_vs_siNONdexKLA_adj_pvalue < pvalue_cutoff &
siNONKLA_vs_siNONdexKLA_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]

#siEP300 vs siEP300 comparisons
dt[, siEP300EtOH_vs_siEP300dex_up := fifelse(siEP300EtOH_vs_siEP300dex_adj_pvalue < pvalue_cutoff &
siEP300EtOH_vs_siEP300dex_Log2_Fold_Change > logfc_cutoff & is_unique == 1 & is_expressed == 1, 1,
0)]
dt[, siEP300EtOH_vs_siEP300dex_dn := fifelse(siEP300EtOH_vs_siEP300dex_adj_pvalue < pvalue_cutoff &
siEP300EtOH_vs_siEP300dex_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 & is_expressed == 1, 1,
0)]
```

```
dt[, siEP300EtOH_vs_siEP300KLA_up := fifelse(siEP300EtOH_vs_siEP300KLA_adj_pvalue < pvalue_cutoff &
siEP300EtOH_vs_siEP300KLA_Log2_Fold_Change > logfc_cutoff & is_unique == 1 & is_expressed == 1, 1,
0)]
dt[, siEP300EtOH_vs_siEP300KLA_dn := fifelse(siEP300EtOH_vs_siEP300KLA_adj_pvalue < pvalue_cutoff &
siEP300EtOH_vs_siEP300KLA_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 & is_expressed == 1, 1,
0)]
dt[,    siEP300EtOH_vs_siEP300dexKLA_up    :=    fifelse(siEP300EtOH_vs_siEP300dexKLA_adj_pvalue    <
pvalue_cutoff & siEP300EtOH_vs_siEP300dexKLA_Log2_Fold_Change > logfc_cutoff & is_unique == 1 &
is_expressed == 1, 1, 0)]
dt[,    siEP300EtOH_vs_siEP300dexKLA_dn    :=    fifelse(siEP300EtOH_vs_siEP300dexKLA_adj_pvalue    <
pvalue_cutoff & siEP300EtOH_vs_siEP300dexKLA_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 &
is_expressed == 1, 1, 0)]
dt[, siEP300dex_vs_siEP300dexKLA_up := fifelse(siEP300dex_vs_siEP300dexKLA_adj_pvalue < pvalue_cutoff
& siEP300dex_vs_siEP300dexKLA_Log2_Fold_Change > logfc_cutoff & is_unique == 1 & is_expressed == 1,
1, 0)]
dt[, siEP300dex_vs_siEP300dexKLA_dn := fifelse(siEP300dex_vs_siEP300dexKLA_adj_pvalue < pvalue_cutoff
& siEP300dex_vs_siEP300dexKLA_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 & is_expressed == 1,
1, 0)]
dt[, siEP300KLA_vs_siEP300dexKLA_up := fifelse(siEP300KLA_vs_siEP300dexKLA_adj_pvalue < pvalue_cutoff
& siEP300KLA_vs_siEP300dexKLA_Log2_Fold_Change > logfc_cutoff & is_unique == 1 & is_expressed == 1,
1, 0)]
dt[, siEP300KLA_vs_siEP300dexKLA_dn := fifelse(siEP300KLA_vs_siEP300dexKLA_adj_pvalue < pvalue_cutoff
& siEP300KLA_vs_siEP300dexKLA_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 & is_expressed == 1,
1, 0)]

#siNON vs siEP300 comparisons
dt[, siNONEtOH_vs_siEP300EtOH_up := fifelse(siNONEtOH_vs_siEP300EtOH_adj_pvalue < pvalue_cutoff &
siNONEtOH_vs_siEP300EtOH_Log2_Fold_Change > logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]
dt[, siNONEtOH_vs_siEP300EtOH_dn := fifelse(siNONEtOH_vs_siEP300EtOH_adj_pvalue < pvalue_cutoff &
siNONEtOH_vs_siEP300EtOH_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 & is_expressed == 1, 1,
0)]
dt[, siNONdex_vs_siEP300dex_up := fifelse(siNONdex_vs_siEP300dex_adj_pvalue < pvalue_cutoff &
siNONdex_vs_siEP300dex_Log2_Fold_Change > logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]
dt[, siNONdex_vs_siEP300dex_dn := fifelse(siNONdex_vs_siEP300dex_adj_pvalue < pvalue_cutoff &
siNONdex_vs_siEP300dex_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]
dt[, siNONKLA_vs_siEP300KLA_up := fifelse(siNONKLA_vs_siEP300KLA_adj_pvalue < pvalue_cutoff &
siNONKLA_vs_siEP300KLA_Log2_Fold_Change > logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]
dt[, siNONKLA_vs_siEP300KLA_dn := fifelse(siNONKLA_vs_siEP300KLA_adj_pvalue < pvalue_cutoff &
siNONKLA_vs_siEP300KLA_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]
dt[,    siNONdexKLA_vs_siEP300dexKLA_up    :=    fifelse(siNONdexKLA_vs_siEP300dexKLA_adj_pvalue    <
pvalue_cutoff & siNONdexKLA_vs_siEP300dexKLA_Log2_Fold_Change > logfc_cutoff & is_unique == 1 &
is_expressed == 1, 1, 0)]
dt[,    siNONdexKLA_vs_siEP300dexKLA_dn    :=    fifelse(siNONdexKLA_vs_siEP300dexKLA_adj_pvalue    <
pvalue_cutoff & siNONdexKLA_vs_siEP300dexKLA_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 &
is_expressed == 1, 1, 0)]

#mark if gene is DE in any treatment (is_deg)
#this is the case if the sum of the rows added just above is greater than 0
#since 1 meant yes
cols_deg <- grep("vs", grep("_up|_dn", colnames(dt), value = TRUE), value = TRUE)
dt[, is_deg := fifelse(matrixStats::rowSums2(as.matrix(.SD)) > 0, 1, 0), .SDcols = cols_deg]
#mark if gene is up regulated in any treatment
dt[,  is_up  :=  fifelse(siNONEtOH_vs_siNONdex_up  ==  1  |  siNONEtOH_vs_siNONKLA_up  ==  1  |
siNONEtOH_vs_siNONdexKLA_up == 1 | siNONKLA_vs_siNONdexKLA_up == 1 | siNONdex_vs_siNONdexKLA_up == 1
|    siEP300EtOH_vs_siEP300dex_up    ==    1    |    siEP300EtOH_vs_siEP300KLA_up    ==    1    |
siEP300EtOH_vs_siEP300dexKLA_up    ==    1    |    siEP300KLA_vs_siEP300dexKLA_up    ==    1    |
siEP300dex_vs_siEP300dexKLA_up == 1 | siNONEtOH_vs_siEP300EtOH_up == 1 | siNONdex_vs_siEP300dex_up ==
1 | siNONKLA_vs_siEP300KLA_up == 1 | siNONdexKLA_vs_siEP300dexKLA_up == 1, 1, 0)]
```

```
#mark if gene is dn regulated in any treatment
dt[, is_dn := fifelse(siNONEtOH_vs_siNONdex_dn == 1 | siNONEtOH_vs_siNONKLA_dn == 1 |
siNONEtOH_vs_siNONdexKLA_dn == 1 | siNONKLA_vs_siNONdexKLA_dn == 1 | siNONdex_vs_siNONdexKLA_dn == 1
|    siEP300EtOH_vs_siEP300dex_dn    ==    1    |    siEP300EtOH_vs_siEP300KLA_dn    ==    1    |
siEP300EtOH_vs_siEP300dexKLA_dn    ==    1    |    siEP300KLA_vs_siEP300dexKLA_dn    ==    1    |
siEP300dex_vs_siEP300dexKLA_dn == 1 | siNONEtOH_vs_siEP300EtOH_dn == 1 | siNONdex_vs_siEP300dex_dn ==
1 | siNONKLA_vs_siEP300KLA_dn == 1 | siNONdexKLA_vs_siEP300dexKLA_dn == 1, 1, 0)]

paste0("number of treatments is: ", dim(dt[, .SD, .SDcols = grep("vs", grep("_up", names(dt), value
= TRUE))])[2])
paste0("number of differentially expressed unique genes is: ", sum(dt[is_unique == 1, is_deg]))

#we can look at the number of DE genes in each comparison
dt[is_unique == 1, apply(.SD, 2, sum), .SDcols = grep("vs", grep("_up|_dn", colnames(dt), value =
TRUE), value = TRUE)]
#and which genes are DE in each comparison separately
dt[siNONEtOH_vs_siNONdex_up == 1, Gene]
dt[siNONEtOH_vs_siNONdex_up == 1, Gene]
dt[siNONEtOH_vs_siNONKLA_up == 1, Gene]
dt[siNONEtOH_vs_siNONKLA_dn == 1, Gene]
dt[siNONEtOH_vs_siNONdexKLA_up == 1, Gene]
dt[siNONEtOH_vs_siNONdexKLA_dn == 1, Gene]
dt[siNONKLA_vs_siNONdexKLA_up == 1, Gene]
dt[siNONKLA_vs_siNONdexKLA_dn == 1, Gene]
dt[siNONdex_vs_siNONdexKLA_up == 1, Gene]
dt[siNONdex_vs_siNONdexKLA_dn == 1, Gene]
dt[siEP300EtOH_vs_siEP300dex_up == 1, Gene]
dt[siEP300EtOH_vs_siEP300dex_dn == 1, Gene]
dt[siEP300EtOH_vs_siEP300KLA_up == 1, Gene]
dt[siEP300EtOH_vs_siEP300KLA_dn == 1, Gene]
dt[siEP300EtOH_vs_siEP300dexKLA_up == 1, Gene]
dt[siEP300EtOH_vs_siEP300dexKLA_dn == 1, Gene]
dt[siEP300KLA_vs_siEP300dexKLA_up == 1, Gene]
dt[siEP300KLA_vs_siEP300dexKLA_dn == 1, Gene]
dt[siEP300dex_vs_siEP300dexKLA_up == 1, Gene]
dt[siEP300dex_vs_siEP300dexKLA_dn == 1, Gene]
dt[siNONEtOH_vs_siEP300EtOH_up == 1, Gene]
dt[siNONEtOH_vs_siEP300EtOH_dn == 1, Gene]
dt[siNONdex_vs_siEP300dex_up == 1, Gene]
dt[siNONdex_vs_siEP300dex_dn == 1, Gene]
dt[siNONKLA_vs_siEP300KLA_up == 1, Gene]
dt[siNONKLA_vs_siEP300KLA_dn == 1, Gene]
dt[siNONdexKLA_vs_siEP300dexKLA_up == 1, Gene]
dt[siNONdexKLA_vs_siEP300dexKLA_dn == 1, Gene]

#add columns to dt for each comparison that tells if the gene is either up and dn regulated
dt[, siNONEtOH_vs_siNONdex_reg := fifelse(siNONEtOH_vs_siNONdex_up == 1 | siNONEtOH_vs_siNONdex_dn ==
1, 1, 0)]
dt[, siNONEtOH_vs_siNONKLA_reg := fifelse(siNONEtOH_vs_siNONKLA_up == 1 | siNONEtOH_vs_siNONKLA_dn ==
1, 1, 0)]
dt[,    siNONEtOH_vs_siNONdexKLA_reg    :=    fifelse(siNONEtOH_vs_siNONdexKLA_up    ==    1    |
siNONEtOH_vs_siNONdexKLA_dn == 1, 1, 0)]
dt[,      siNONdex_vs_siNONdexKLA_reg      :=      fifelse(siNONdex_vs_siNONdexKLA_up      ==1      |
siNONdex_vs_siNONdexKLA_dn == 1, 1, 0)]
dt[,      siNONKLA_vs_siNONdexKLA_reg      :=      fifelse(siNONKLA_vs_siNONdexKLA_up      ==1      |
siNONKLA_vs_siNONdexKLA_dn == 1, 1, 0)]
dt[,    siEP300EtOH_vs_siEP300dex_reg    :=    fifelse(siEP300EtOH_vs_siEP300dex_up    ==1    |
siEP300EtOH_vs_siEP300dex_dn == 1, 1, 0)]
```

```
dt[,     siEP300EtOH_vs_siEP300KLA_reg     :=     fifelse(siEP300EtOH_vs_siEP300KLA_up     ==1     |
siEP300EtOH_vs_siEP300KLA_dn == 1, 1, 0)]
dt[,    siEP300EtOH_vs_siEP300dexKLA_reg    :=    fifelse(siEP300EtOH_vs_siEP300dexKLA_up    ==1    |
siEP300EtOH_vs_siEP300dexKLA_dn == 1, 1, 0)]
dt[,     siEP300dex_vs_siEP300dexKLA_reg    :=    fifelse(siEP300dex_vs_siEP300dexKLA_up    ==1    |
siEP300dex_vs_siEP300dexKLA_dn == 1, 1, 0)]
dt[,     siEP300KLA_vs_siEP300dexKLA_reg    :=    fifelse(siEP300KLA_vs_siEP300dexKLA_up    ==1    |
siEP300KLA_vs_siEP300dexKLA_dn == 1, 1, 0)]
dt[,     siNONEtOH_vs_siEP300EtOH_reg    :=    fifelse(siNONEtOH_vs_siEP300EtOH_up    ==    1    |
siNONEtOH_vs_siEP300EtOH_dn == 1, 1, 0)]
dt[, siNONdex_vs_siEP300dex_reg := fifelse(siNONdex_vs_siEP300dex_up == 1 | siNONdex_vs_siEP300dex_dn
== 1, 1, 0)]
dt[, siNONKLA_vs_siEP300KLA_reg := fifelse(siNONKLA_vs_siEP300KLA_up == 1 | siNONKLA_vs_siEP300KLA_dn
== 1, 1, 0)]
dt[,   siNONdexKLA_vs_siEP300dexKLA_reg   :=   fifelse(siNONdexKLA_vs_siEP300dexKLA_up   ==   1   |
siNONdexKLA_vs_siEP300dexKLA_dn == 1, 1, 0)]

############
##ENHANCERS

#set the cutoffs
#again, no logfc cutoff is desired
#technically the p-value cutoff is redundant since the enhancer data have already been filtered with
FDR<0.05
#but it has to be set nonetheless because otherwise it will use the cutoff of 0.01 from the genes
section above
pvalue_cutoff = 0.05
logfc_cutoff = 0

#list all the different comparisons
grep("vs", grep("adj_pvalue", colnames(dt_enh), value = TRUE), value = TRUE)

#siNON vs siNON comparisons
dt_enh[,  siNONEtOH_vs_siNONdex_up  :=  fifelse(siNONEtOH_vs_siNONdex_adj_pvalue  <  pvalue_cutoff  &
siNONEtOH_vs_siNONdex_Log2_Fold_Change > logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]
dt_enh[,  siNONEtOH_vs_siNONdex_dn  :=  fifelse(siNONEtOH_vs_siNONdex_adj_pvalue  <  pvalue_cutoff  &
siNONEtOH_vs_siNONdex_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]
dt_enh[,  siNONEtOH_vs_siNONKLA_up  :=  fifelse(siNONEtOH_vs_siNONKLA_adj_pvalue  <  pvalue_cutoff  &
siNONEtOH_vs_siNONKLA_Log2_Fold_Change > logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]
dt_enh[,  siNONEtOH_vs_siNONKLA_dn  :=  fifelse(siNONEtOH_vs_siNONKLA_adj_pvalue  <  pvalue_cutoff  &
siNONEtOH_vs_siNONKLA_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]
dt_enh[, siNONEtOH_vs_siNONdexKLA_up := fifelse(siNONEtOH_vs_siNONdexKLA_adj_pvalue < pvalue_cutoff
& siNONEtOH_vs_siNONdexKLA_Log2_Fold_Change > logfc_cutoff & is_unique == 1 & is_expressed == 1, 1,
0)]
dt_enh[, siNONEtOH_vs_siNONdexKLA_dn := fifelse(siNONEtOH_vs_siNONdexKLA_adj_pvalue < pvalue_cutoff
& siNONEtOH_vs_siNONdexKLA_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 & is_expressed == 1, 1,
0)]
dt_enh[, siNONdex_vs_siNONdexKLA_up := fifelse(siNONdex_vs_siNONdexKLA_adj_pvalue < pvalue_cutoff &
siNONdex_vs_siNONdexKLA_Log2_Fold_Change > logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]
dt_enh[, siNONdex_vs_siNONdexKLA_dn := fifelse(siNONdex_vs_siNONdexKLA_adj_pvalue < pvalue_cutoff &
siNONdex_vs_siNONdexKLA_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]
dt_enh[, siNONKLA_vs_siNONdexKLA_up := fifelse(siNONKLA_vs_siNONdexKLA_adj_pvalue < pvalue_cutoff &
siNONKLA_vs_siNONdexKLA_Log2_Fold_Change > logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]
dt_enh[, siNONKLA_vs_siNONdexKLA_dn := fifelse(siNONKLA_vs_siNONdexKLA_adj_pvalue < pvalue_cutoff &
siNONKLA_vs_siNONdexKLA_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]

#siEP300 vs siEP300 comparisons
```

```
dt_enh[, siEP300EtOH_vs_siEP300dex_up := fifelse(siEP300EtOH_vs_siEP300dex_adj_pvalue < pvalue_cutoff
& siEP300EtOH_vs_siEP300dex_Log2_Fold_Change > logfc_cutoff & is_unique == 1 & is_expressed == 1, 1,
0)]
dt_enh[, siEP300EtOH_vs_siEP300dex_dn := fifelse(siEP300EtOH_vs_siEP300dex_adj_pvalue < pvalue_cutoff
& siEP300EtOH_vs_siEP300dex_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 & is_expressed == 1, 1,
0)]
dt_enh[, siEP300EtOH_vs_siEP300KLA_up := fifelse(siEP300EtOH_vs_siEP300KLA_adj_pvalue < pvalue_cutoff
& siEP300EtOH_vs_siEP300KLA_Log2_Fold_Change > logfc_cutoff & is_unique == 1 & is_expressed == 1, 1,
0)]
dt_enh[, siEP300EtOH_vs_siEP300KLA_dn := fifelse(siEP300EtOH_vs_siEP300KLA_adj_pvalue < pvalue_cutoff
& siEP300EtOH_vs_siEP300KLA_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 & is_expressed == 1, 1,
0)]
dt_enh[,  siEP300EtOH_vs_siEP300dexKLA_up  :=  fifelse(siEP300EtOH_vs_siEP300dexKLA_adj_pvalue  <
pvalue_cutoff & siEP300EtOH_vs_siEP300dexKLA_Log2_Fold_Change > logfc_cutoff & is_unique == 1 &
is_expressed == 1, 1, 0)]
dt_enh[,  siEP300EtOH_vs_siEP300dexKLA_dn  :=  fifelse(siEP300EtOH_vs_siEP300dexKLA_adj_pvalue  <
pvalue_cutoff & siEP300EtOH_vs_siEP300dexKLA_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 &
is_expressed == 1, 1, 0)]
dt_enh[,  siEP300dex_vs_siEP300dexKLA_up  :=  fifelse(siEP300dex_vs_siEP300dexKLA_adj_pvalue  <
pvalue_cutoff & siEP300dex_vs_siEP300dexKLA_Log2_Fold_Change > logfc_cutoff & is_unique == 1 &
is_expressed == 1, 1, 0)]
dt_enh[,  siEP300dex_vs_siEP300dexKLA_dn  :=  fifelse(siEP300dex_vs_siEP300dexKLA_adj_pvalue  <
pvalue_cutoff & siEP300dex_vs_siEP300dexKLA_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 &
is_expressed == 1, 1, 0)]
dt_enh[,  siEP300KLA_vs_siEP300dexKLA_up  :=  fifelse(siEP300KLA_vs_siEP300dexKLA_adj_pvalue  <
pvalue_cutoff & siEP300KLA_vs_siEP300dexKLA_Log2_Fold_Change > logfc_cutoff & is_unique == 1 &
is_expressed == 1, 1, 0)]
dt_enh[,  siEP300KLA_vs_siEP300dexKLA_dn  :=  fifelse(siEP300KLA_vs_siEP300dexKLA_adj_pvalue  <
pvalue_cutoff & siEP300KLA_vs_siEP300dexKLA_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 &
is_expressed == 1, 1, 0)]

#siNON vs siEP300 comparisons
dt_enh[, siNONEtOH_vs_siEP300EtOH_up := fifelse(siNONEtOH_vs_siEP300EtOH_adj_pvalue < pvalue_cutoff
& siNONEtOH_vs_siEP300EtOH_Log2_Fold_Change > logfc_cutoff & is_unique == 1 & is_expressed == 1, 1,
0)]
dt_enh[, siNONEtOH_vs_siEP300EtOH_dn := fifelse(siNONEtOH_vs_siEP300EtOH_adj_pvalue < pvalue_cutoff
& siNONEtOH_vs_siEP300EtOH_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 & is_expressed == 1, 1,
0)]
dt_enh[, siNONdex_vs_siEP300dex_up := fifelse(siNONdex_vs_siEP300dex_adj_pvalue < pvalue_cutoff &
siNONdex_vs_siEP300dex_Log2_Fold_Change > logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]
dt_enh[, siNONdex_vs_siEP300dex_dn := fifelse(siNONdex_vs_siEP300dex_adj_pvalue < pvalue_cutoff &
siNONdex_vs_siEP300dex_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]
dt_enh[, siNONKLA_vs_siEP300KLA_up := fifelse(siNONKLA_vs_siEP300KLA_adj_pvalue < pvalue_cutoff &
siNONKLA_vs_siEP300KLA_Log2_Fold_Change > logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]
dt_enh[, siNONKLA_vs_siEP300KLA_dn := fifelse(siNONKLA_vs_siEP300KLA_adj_pvalue < pvalue_cutoff &
siNONKLA_vs_siEP300KLA_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 & is_expressed == 1, 1, 0)]
dt_enh[,  siNONdexKLA_vs_siEP300dexKLA_up  :=  fifelse(siNONdexKLA_vs_siEP300dexKLA_adj_pvalue  <
pvalue_cutoff & siNONdexKLA_vs_siEP300dexKLA_Log2_Fold_Change > logfc_cutoff & is_unique == 1 &
is_expressed == 1, 1, 0)]
dt_enh[,  siNONdexKLA_vs_siEP300dexKLA_dn  :=  fifelse(siNONdexKLA_vs_siEP300dexKLA_adj_pvalue  <
pvalue_cutoff & siNONdexKLA_vs_siEP300dexKLA_Log2_Fold_Change < -logfc_cutoff & is_unique == 1 &
is_expressed == 1, 1, 0)]

#mark if enhancer is DE, up or dn in any treatment (is_deg)
cols_deg <- grep("vs", grep("_up|_dn", colnames(dt_enh), value = TRUE), value = TRUE)
dt_enh[, is_deg := fifelse(matrixStats::rowSums2(as.matrix(.SD)) > 0, 1, 0), .SDcols = cols_deg]
dt_enh[,  is_up  :=  fifelse(siNONEtOH_vs_siNONdex_up  ==  1  |  siNONEtOH_vs_siNONKLA_up  ==  1  |
siNONEtOH_vs_siNONdexKLA_up == 1 | siNONKLA_vs_siNONdexKLA_up == 1 | siNONdex_vs_siNONdexKLA_up == 1
```

```
| siEP300EtOH_vs_siEP300dex_up == 1 | siEP300EtOH_vs_siEP300KLA_up == 1 |
siEP300EtOH_vs_siEP300dexKLA_up == 1 | siEP300KLA_vs_siEP300dexKLA_up == 1 |
siEP300dex_vs_siEP300dexKLA_up == 1 | siNONEtOH_vs_siEP300EtOH_up == 1 | siNONdex_vs_siEP300dex_up ==
1 | siNONKLA_vs_siEP300KLA_up == 1 | siNONdexKLA_vs_siEP300dexKLA_up == 1, 1, 0)]
dt_enh[, is_dn := fifelse(siNONEtOH_vs_siNONdex_dn == 1 | siNONEtOH_vs_siNONKLA_dn == 1 |
siNONEtOH_vs_siNONdexKLA_dn == 1 | siNONKLA_vs_siNONdexKLA_dn == 1 | siNONdex_vs_siNONdexKLA_dn == 1
| siEP300EtOH_vs_siEP300dex_dn == 1 | siEP300EtOH_vs_siEP300KLA_dn == 1 |
siEP300EtOH_vs_siEP300dexKLA_dn == 1 | siEP300KLA_vs_siEP300dexKLA_dn == 1 |
siEP300dex_vs_siEP300dexKLA_dn == 1 | siNONEtOH_vs_siEP300EtOH_dn == 1 | siNONdex_vs_siEP300dex_dn ==
1 | siNONKLA_vs_siEP300KLA_dn == 1 | siNONdexKLA_vs_siEP300dexKLA_dn == 1, 1, 0)]


paste0("number of treatments is: ", dim(dt_enh[, .SD, .SDcols = grep("vs", grep("_up", names(dt_enh),
value = TRUE))])[2])
paste0("number of differentially expressed unique enhancers is: ", sum(dt_enh[is_unique == 1,
is_deg]))

#we can look at the number of number of DE enhancers in each category
dt_enh[is_unique == 1, apply(.SD, 2, sum), .SDcols = grep("vs", grep("_up|_dn", colnames(dt_enh),
value = TRUE), value = TRUE)]
#if you want to see the genes associated with the DE enhancers, follow the same logic as above for
genes

#add columns to dt_enh for each comparison that tells if the enhancer is either up and dn regulated
dt_enh[, siNONEtOH_vs_siNONdex_reg := fifelse(siNONEtOH_vs_siNONdex_up == 1 |
siNONEtOH_vs_siNONdex_dn == 1, 1, 0)]
dt_enh[, siNONEtOH_vs_siNONKLA_reg := fifelse(siNONEtOH_vs_siNONKLA_up == 1 |
siNONEtOH_vs_siNONKLA_dn == 1, 1, 0)]
dt_enh[, siNONEtOH_vs_siNONdexKLA_reg := fifelse(siNONEtOH_vs_siNONdexKLA_up == 1 |
siNONEtOH_vs_siNONdexKLA_dn == 1, 1, 0)]
dt_enh[, siNONdex_vs_siNONdexKLA_reg := fifelse(siNONdex_vs_siNONdexKLA_up ==1 |
siNONdex_vs_siNONdexKLA_dn == 1, 1, 0)]
dt_enh[, siNONKLA_vs_siNONdexKLA_reg := fifelse(siNONKLA_vs_siNONdexKLA_up ==1 |
siNONKLA_vs_siNONdexKLA_dn == 1, 1, 0)]
dt_enh[, siEP300EtOH_vs_siEP300dex_reg := fifelse(siEP300EtOH_vs_siEP300dex_up ==1 |
siEP300EtOH_vs_siEP300dex_dn == 1, 1, 0)]
dt_enh[, siEP300EtOH_vs_siEP300KLA_reg := fifelse(siEP300EtOH_vs_siEP300KLA_up ==1 |
siEP300EtOH_vs_siEP300KLA_dn == 1, 1, 0)]
dt_enh[, siEP300EtOH_vs_siEP300dexKLA_reg := fifelse(siEP300EtOH_vs_siEP300dexKLA_up ==1 |
siEP300EtOH_vs_siEP300dexKLA_dn == 1, 1, 0)]
dt_enh[, siEP300dex_vs_siEP300dexKLA_reg := fifelse(siEP300dex_vs_siEP300dexKLA_up ==1 |
siEP300dex_vs_siEP300dexKLA_dn == 1, 1, 0)]
dt_enh[, siEP300KLA_vs_siEP300dexKLA_reg := fifelse(siEP300KLA_vs_siEP300dexKLA_up ==1 |
siEP300KLA_vs_siEP300dexKLA_dn == 1, 1, 0)]
dt_enh[, siNONEtOH_vs_siEP300EtOH_reg := fifelse(siNONEtOH_vs_siEP300EtOH_up == 1 |
siNONEtOH_vs_siEP300EtOH_dn == 1, 1, 0)]
dt_enh[, siNONdex_vs_siEP300dex_reg := fifelse(siNONdex_vs_siEP300dex_up == 1 |
siNONdex_vs_siEP300dex_dn == 1, 1, 0)]
dt_enh[, siNONKLA_vs_siEP300KLA_reg := fifelse(siNONKLA_vs_siEP300KLA_up == 1 |
siNONKLA_vs_siEP300KLA_dn == 1, 1, 0)]
dt_enh[, siNONdexKLA_vs_siEP300dexKLA_reg := fifelse(siNONdexKLA_vs_siEP300dexKLA_up == 1 |
siNONdexKLA_vs_siEP300dexKLA_dn == 1, 1, 0)]


##################################
## Principal component analysis ##
##################################

```

```
#######
##GENES
#using DE genes

#get tpm values of replicates based on expression
ma <- as.matrix(dt[is_deg == 1 & is_unique == 1, .SD, .SDcols = tpm_rep_columns])
#how many genes are included?
dim(ma)
#check for skewness of the data (should be approx. normally distributed)
hist(ma)
#log normalization for x+1 data to correct for skewness
ma <- log(as.matrix(ma[]+1))
hist(ma)

#make prcomp object for PCA
#options: center to zero, scale according to unit variance
ma.pca <- prcomp(t(ma), center = TRUE, scale. = TRUE)
plot(ma.pca, type = "l") # plot variances of principal components

#calculate the percentage of variance explained by PCs for further use (sd is square root of variance)
perc_var <- lapply(seq_along(ma.pca$sdev), function(i){
  #divide sd by sum of sds and multiply by 100 to get %, round to 2 digits
  perc <- round(ma.pca$sdev[i]^2 / sum((ma.pca$sdev)^2)*100, 2)
  return(perc)
}
)

#return PCs to data.table & reshape to long format
#i.e. add rows to identify what comparison data is from
#instead of having separate columns for each treatment
#long format data is extremely useful for ggplot2 visualizations
dt.pca <- as.data.table(ma.pca$x)
dt.pca[, sample := row.names(ma.pca$x)]
dt.pca[, group := get_nth_of_list(dt.pca$sample, sep = "_", n = 3)]
dt.pca[, replicate := get_nth_of_list(dt.pca$sample, sep = "_", n = 4)]

#plot PC1 vs PC2 using ggplot
#first define what the used data is (dt.pca)
#and then define with aes what should be mapped to the x and y axes and to other visual variables
#additional parameters are added with +
ggplot(dt.pca, aes(x = PC1, y = PC2, color = group, label = group, shape = replicate))+
  theme_classic()+
  #define colours as hex codes
  scale_color_manual(values                                                                    =
c("#9914db","#c77dff","#023e8a","#4895ef","#f77f00","#F7B22C","#fc4c27","#A10003")) +
  #set the labels for various parts
  labs(title = paste0("All ", dim(ma)[1], " differentially transcribed genes"),
       color = "Treatment", shape = "Replicate",
       x = paste0("PC1: ", perc_var[1],"% of variance explained"),
       y = paste0("PC2: ", perc_var[2],"% of variance explained"))+
  #annotate the data points
  #use ggrepel to force them to not overlap each other
  geom_label_repel(aes(label = group),
                   box.padding   = 0.35,
                   point.padding = 0.5,
                   max.overlaps = 100,
                   segment.color = "grey50") +
  geom_point(size = 5)
```

```
#use ggsave to save the most recently drawn plot as the desired file format and define dimensions of
the image
ggsave("genes_PCA.pdf", plot = last_plot(), device = "pdf", width = 8, height = 7, units =  "in")
ggsave("genes_PCA.tif", plot = last_plot(), device = "tiff", width = 8, height = 7, units =  "in")
ggsave("genes_PCA.png", plot = last_plot(), device = "png", width = 8, height = 7, units =  "in")


###########
##ENHANCERS
#using DT enhancers

#get tpm values of replicates based on expression
ma <- as.matrix(dt_enh[is_deg == 1 & is_unique == 1, .SD, .SDcols = tpm_rep_columns])
dim(ma)
hist(ma)
ma <- log(as.matrix(ma[]+1))
hist(ma)

#make prcomp object
ma.pca <- prcomp(t(ma), center = TRUE, scale. = TRUE)
plot(ma.pca, type = "l")

#calculate the percentage of variance explained by PCs for further use
perc_var <- lapply(seq_along(ma.pca$sdev), function(i){
  perc <- round(ma.pca$sdev[i]^2 / sum((ma.pca$sdev)^2)*100, 2)
  return(perc)
}
)

#return PCs to data.table & reshape to long format
dt_enh.pca <- as.data.table(ma.pca$x)
dt_enh.pca[, sample := row.names(ma.pca$x)]
dt_enh.pca[, group := get_nth_of_list(dt_enh.pca$sample, sep = "_", n = 3)]
dt_enh.pca[, replicate := get_nth_of_list(dt_enh.pca$sample, sep = "_", n = 4)]

# plot PC1 vs PC2
ggplot(dt_enh.pca, aes(x = PC1, y = PC2, color = group, label = group, shape = replicate))+
  theme_classic()+
  scale_color_manual(values                                                                                =
c("#9914db","#c77dff","#023e8a","#4895ef","#f77f00","#F7B22C","#fc4c27","#A10003")) +
  labs(title = paste0(dim(ma)[1], " differentially transcribed enhancers"),
       x = paste0("PC1: ", perc_var[1],"% of variance explained"),
       y = paste0("PC2: ", perc_var[2],"% of variance explained"),
       color = "Treatment", shape = "Replicate")+
  geom_label_repel(aes(label = group),
                   box.padding   = 0.35,
                   point.padding = 0.5,
                   max.overlaps = 100,
                   segment.color = 'grey50') +
  geom_point(size = 5)
ggsave("enhancers_PCA.pdf", plot = last_plot(), device = "pdf", width = 8, height = 7, units =  "in")
ggsave("enhancers_PCA.tif", plot = last_plot(), device = "tiff", width = 8, height = 7, units =  "in")
ggsave("enhancers_PCA.png", plot = last_plot(), device = "png", width = 8, height = 7, units =  "in")


###################################
## Heatmaps of DE genes/enhancers ##
###################################
```

```
693
694   #######
695   ##GENES
696
697   #decide which genes to plot in the heatmap (hm)
698   #here we take all of the up & dn regulated genes (_reg)
699   selected <- unique(grep("_reg", colnames(dt), value = TRUE))
700
701   #in the coming for-loop all rows that have DE genes in the comparison at hand
702   #are taken from the columns containing TPM values
703   #and added to the end of the objects at the end of each loop
704   #this means there are many duplicate DE genes for in the heatmap as a whole
705   #but in order to split it by comparison, it is done this way
706
707   #initiate objects for loop
708   #so that the objects can be cumulatively added to in the loop using rbind and append
709   #matrix needs to have the number of columns predefined
710   #the number of columns is as many as there are TPM columns
711   hm <- matrix(ncol = length(grep("_TPM", colnames(dt), value = TRUE)))
712   comparison <- vector()
713   number_annot <- list()
714
715   #for each selected comparison
716   for(i in 1:length(selected)){
717     #use the column name to fetch those TPM columns where the gene is DE in the comparison
718     #and add those to the end of the hm matrix
719     #the data.table is special in the way that it understands the column names as objects
720     #and thus calling them as character strings as they are in the selected vector does not work
721     #get() is used to get the object with the name matching to the character string given
722     hm <- rbind(hm, as.matrix(dt[(get(selected[i])) == 1, .SD, .SDcols = tpm_rep_columns]))
723     #create categories for splitting the heatmap
724     #in a vector repeat the comparison name as many times as there are rows with DE genes for that
725   comparison
726     comparison   <-   append(comparison,   rep(gsub("_",   "   ",   gsub("_reg",   "",   selected[i])),
727   nrow(dt[(get(selected[i])) == 1, .SD, .SDcols = tpm_rep_columns])))
728     #save the number of rows with DE genes to a list, these can be used to annotate the heatmap
729     number_annot[[i]] <- nrow(dt[(get(selected[i])) == 1, .SD, .SDcols = tpm_rep_columns])
730   }
731
732   #remove any NAs
733   hm <- na.omit(hm)
734   #use the comparison vector as the rownames of hm
735   rownames(hm) <- comparison
736
737   #remove any items that = 0
738   #so that the annotation is not offset
739   #because those with 0 will not be plotted in the heatmap but the annotation will
740   number_annot<- number_annot[! number_annot %in% 0]
741
742   #Z-score normalization
743   hm_scaled = t(apply(hm, 1, scale))
744   #add column names to normalized hm matrix and clean them up by taking only the treatment information
745   colnames(hm_scaled) <- get_nth_of_list(colnames(hm), sep = "_", n = 3)
746
747   #create a colour gradient for the heatmap
748   #set the breaks first and then the colours for those breaks
749   col_fun = colorRamp2(c(min(hm_scaled), 0, max(hm_scaled)), c("blue3", "lightyellow", "firebrick3"))
750
```

```
#create an empty annotation area for the heatmap where the numbers of DE genes can be annotated
HA <- rowAnnotation(annot = anno_empty(which = "row", border = FALSE, width = unit(15, "mm")))

#since the heatmap is not a ggplot2 object, it needs to be saved using the standard functions
#uncomment the other and rerun to save both pdf and png
#pdf(file = "genes_heatmap.png", width = 9.5, height = 9.5)
png(file = "genes_heatmap.png", width = 9.5, height = 17, units = "in", res = 600)
#plot the normalized TPM values as a heatmap
Heatmap(hm_scaled,
        #set the name for the legend
        name = "Z-score",
        #set the colouring
        col = col_fun,
        #add black border
        border_gp = gpar(col = "black", lwd = 1.5),
        use_raster = TRUE,
        #set column names to the top and define what they are
        #only take the hormone treatment and replicate number
        column_names_side = "top",
        column_labels = paste0(gsub("siEP300", "", gsub("siNON", "",
                                                        get_nth_of_list(colnames(hm), sep = "_", n =
3))), " ",
                                                        get_nth_of_list(colnames(hm), sep = "_", n =
4)),
        #set how to split rows into slices
        #use the column names for this and set the levels for the factor
        row_split = factor(rownames(hm_scaled), levels = unique(rownames(hm_scaled))),
        #same for columns except using the siRNA condition
        column_split = factor(c(rep("siNON", times = length(grep("siNON", colnames(hm_scaled)))),
                                rep("siEP300",         times         =         length(grep("siEP300",
colnames(hm_scaled)))),
                                levels = c("siNON", "siEP300")),
        #set row and column titles to be the levels defined above with %s
        column_title = "%s",
        row_title = "%s",
        #set graphical parameters for row titles
        row_title_gp = gpar(fontsize = 10),
        row_title_rot = 0,
        #add gaps between row slices so that larger gaps will be between siNON vs siNON,
        #siEP300 vs siEP300 and siNON vs siEP300 categories
        row_gap = unit(c(1, 1, 1, 1, 3, 1, 1, 3, 1, 1, 1), "mm"),
        #sort the rows automatically within slices to form clusters
        #but do not show dendrograms or row names
        cluster_rows = TRUE,
        show_row_dend = FALSE,
        show_row_names = FALSE,
        #do not cluster row or column slices or columns
        #so that they remain in the order they are in the data
        cluster_row_slices = FALSE,
        cluster_column_slices = FALSE,
        cluster_columns = FALSE,
        #add the empty space for the number annotations to the right of the heatmap
        right_annotation = HA
)

#add the annotations to the empty annotation using decorate_annotation
for(i in 1:length(number_annot)) {
  #place it at the corresponding slice
```

```
  decorate_annotation("annot", slice = i, {
    grid.text(paste0(number_annot[[i]], " genes"), gp = gpar(cex = 0.7), x = unit(0.2, "mm"), just =
"left")
  })}
#calling dev.off() finally saves the plot that was initiated with pdf/png
dev.off()


###########
##ENHANCERS

selected_enh <- unique(grep("_reg", colnames(dt_enh), value = TRUE))

hm_enh <- matrix(ncol = length(grep("_TPM", colnames(dt_enh), value = TRUE)))
comparison_enh <- vector()
number_annot_enh <- list()

for(i in 1:length(selected_enh)){
  hm_enh <- rbind(hm_enh, as.matrix(dt_enh[(get(selected[i])) == 1, .SD, .SDcols = tpm_rep_columns]))
  comparison_enh    <-    append(comparison_enh,    rep(gsub("_",    "    ",    selected_enh[i]),
nrow(dt_enh[(get(selected[i])) == 1, .SD, .SDcols = tpm_rep_columns])))
  number_annot_enh[[i]] <- nrow(dt_enh[(get(selected[i])) == 1, .SD, .SDcols = tpm_rep_columns])
}

hm_enh <- na.omit(hm_enh)
rownames(hm_enh) <- comparison_enh
number_annot_enh <- number_annot_enh[! number_annot_enh %in% 0]

#Z-score normalization
hm_enh_scaled = t(apply(hm_enh, 1, scale))
colnames(hm_enh_scaled) <- get_nth_of_list(colnames(hm_enh), sep = "_", n = 3)

col_fun_enh = colorRamp2(c(min(hm_enh_scaled), 0, max(hm_enh_scaled)), c("blue3", "lightyellow",
"firebrick3"))

HA_enh <- rowAnnotation(annot_enh = anno_empty(which = "row", border = FALSE, width = unit(20, "mm")))

#pdf(file = "enhancers_heatmap.pdf", width = 9.5, height = 12)
png(file = "enhancers_heatmap.png", width = 9.5, height = 12, units = "in", res = 600)
Heatmap(hm_enh_scaled,
        name = "Z-score",
        col = col_fun_enh,
        border_gp = gpar(col = "black", lwd = 1.5),
        row_gap = unit(c(1, 1, 3, 1, 1, 3, 1, 1, 2), "mm"),
        column_names_side = "top",
        column_labels       =       paste0(gsub("siEP300",       "",       gsub("siNON",       "",
get_nth_of_list(colnames(hm_enh), sep = "_", n = 3))), " ", get_nth_of_list(colnames(hm_enh), sep =
"_", n = 4)),
        row_split = factor(rownames(hm_enh_scaled), levels = unique(rownames(hm_enh_scaled))),
        column_split = factor(c(rep("siNON", times = length(grep("siNON", colnames(hm_enh_scaled)))),
                                rep("siEP300",         times         =         length(grep("siEP300",
colnames(hm_enh_scaled)))),
                              levels = c("siNON", "siEP300")),
        column_title = "%s",
        row_title_gp = gpar(fontsize = 10),
        row_title = "%s",
        row_title_rot = 0,
        show_row_dend = FALSE,
```

```
          cluster_rows = TRUE,
          show_row_names = FALSE,
          cluster_row_slices = FALSE,
          cluster_column_slices = FALSE,
          cluster_columns = FALSE,
          right_annotation = HA_enh
)

for(i in 1:length(number_annot_enh)) {
  decorate_annotation("annot_enh", slice = i, {
    grid.text(paste0(number_annot_enh[[i]], " enhancers"), gp = gpar(cex = 0.7), x = unit(0.2, "mm"),
just = "left")
  })}
dev.off()


###################
## Volcano plots ##
###################

#######
##GENES

#define the DE gene categories by column name
#remove "_reg" so that names can be more easily reformatted in the coming loop
deg_groups <- unique(gsub("_reg", "", grep("_reg", colnames(dt), value = TRUE)))

#define groups for volcano plots
vp_groups <- gsub("_Log2_Fold_Change","", grep("Log2_Fold_Change", names(dt), value = TRUE))
#not all comparisons in the log2 FC columns were relevant
#so the vp_groups needs to be filtered to keep only the relevant comparisons
#which were saved to deg_groups
vp_groups <- vp_groups[vp_groups %in% deg_groups]
#order vp_enh_groups according to deg_enh_groups
#so that when plotted the values are attributed to correct comparisons
vp_groups <- vp_groups[order(match(vp_groups, deg_groups))]

#reset the cutoffs to be the same as they were for the genes
#these are needed in the coming loop
pvalue_cutoff = 0.01
logfc_cutoff = 0.5

#in a loop make a list of volcano plots
vp <- lapply(seq_along(vp_groups), function(i){
  #save the comparison to be handled in iname
  iname <- vp_groups[[i]]
  #formulate the names of the columns with the required information using iname
  iname_vector <- c(paste0(rep(iname, 2), c("_Log2_Fold_Change", "_adj_pvalue")),
                    paste0(deg_groups[[i]], "_up"),
                    paste0(deg_groups[[i]], "_dn"))
  #use iname_vector to fetch the columns with unique and expressed genes
  vp_data <- dt[is_unique == 1 & is_expressed == 1, .SD, .SDcols = iname_vector]
  colnames(vp_data)[1:2] <- c("logfc", "adjpval")
  #define the direction of the DE gene
  #up in column 3 and dn in column 4
  #if up = 1 and dn = 0 -> 1 i.e. upregulated and if up = 0 and dn = 1 -> -1 i.e. downregulated
  vp_data[, deg_direction := factor((vp_data[,3] - vp_data[,4]), levels = c("-1", "0", "1"))]

```

```
  #make ggplot objects
  ggplot(vp_data, aes(x = logfc, y = -log10(adjpval)))+
    theme_classic()+
    theme(legend.position = "none", plot.title = element_text(hjust = 0.5, size = 10))+
    geom_point(aes(col = deg_direction))+
    #add lines at + and - log FC and at adj. p-value
    geom_vline(xintercept = c(logfc_cutoff, -logfc_cutoff), lty = 2, color = "gray50")+
    geom_hline(yintercept = -log10(pvalue_cutoff), lty = 2, color = "gray50")+
    #format axis and plot labels
    labs(title = gsub("_", " ",paste0(iname)),
         x = expression(paste("log"[2]," ", "FC", sep = "")),
         y = expression(paste("-log"[10]," ", "adj. p-value", sep = "")))+
    #select colouring for the points based on direction
    scale_color_manual(values=c("blue2", "gray", "firebrick2"), limits = c("-1", "0", "1"))
}
)

#show plots as grid using grid.arrange and do.call
pdf(file = "genes_volcano.pdf", width = 11.5, height = 9)
png(file = "genes_volcano.png", width = 11.5, height = 9, units = "in", res = 600)
do.call(grid.arrange, vp)
dev.off()


###########
##ENHANCERS

#define DE enhancer categories by name
deg_enh_groups <- unique(gsub("_reg", "", grep("_reg", colnames(dt_enh), value = TRUE)))

#define groups
vp_enh_groups <- gsub("_Log2_Fold_Change","", grep("Log2_Fold_Change", names(dt_enh), value = TRUE))
vp_enh_groups <- vp_enh_groups[vp_enh_groups %in% deg_enh_groups]
#order vp_enh_groups according to deg_enh_groups
vp_enh_groups <- vp_enh_groups[order(match(vp_enh_groups, deg_enh_groups))]

#reset the cutoffs to be the same as they were for the enhancers
pvalue_cutoff = 0.05
logfc_cutoff = 0

#make list of volcano plots
vp_enh <- lapply(seq_along(vp_enh_groups), function(i){
  iname <- vp_enh_groups[[i]]
  iname_vector <- c(paste0(rep(iname, 2), c("_Log2_Fold_Change", "_adj_pvalue")),
                    paste0(deg_enh_groups[[i]], "_up"),
                    paste0(deg_enh_groups[[i]], "_dn"))
  vp_enh_data <- dt_enh[is_unique == 1 & is_expressed == 1, .SD, .SDcols = iname_vector]
  colnames(vp_enh_data)[1:2] <- c("logfc", "adjpval")
  vp_enh_data[, deg_direction := vp_enh_data[,3] - vp_enh_data[,4]]
  vp_enh_data[, deg_direction := factor(deg_direction, levels = c("-1", "0", "1"))]

  #make ggplot objects
  ggplot(vp_enh_data, aes(x = logfc, y = -log10(adjpval)))+
    theme_classic()+
    theme(legend.position = "none", plot.title = element_text(hjust = 0.5, size = 10))+
    geom_point(aes(col = deg_direction))+
    geom_vline(xintercept = c(logfc_cutoff, -logfc_cutoff), lty = 2, color = "gray50")+
    geom_hline(yintercept = -log10(pvalue_cutoff), lty = 2, color = "gray50")+
```

```
    labs(title = gsub("_", " ",paste0(iname)),
         x = expression(paste("log"[2]," ", "FC", sep = "")),
         y = expression(paste("-log"[10]," ", "adj. p-value", sep = "")))+
    scale_color_manual(values=c("blue2", "gray", "firebrick2"), limits = c("-1", "0", "1"))
}
)

#show plots as grid
pdf(file = "enhancers_volcano.pdf", width = 11, height = 9)
png(file = "enhancers_volcano.png", width = 11, height = 9, units = "in", res = 600)
do.call(grid.arrange, vp_enh)
dev.off()


###################
## Venn diagrams ##
###################

#######
##GENES

#create a list of lists with each Venn diagram comparison to be plotted
#the Gene columns are taken and these will be used to see how many genes are shared between comparisons
toVenn <- list(list(siNONEtOH_vs_siNONdex = dt[siNONEtOH_vs_siNONdex_reg == 1, Gene],
siNONEtOH_vs_siNONKLA = dt[siNONEtOH_vs_siNONKLA_reg == 1, Gene], siNONEtOH_vs_siNONdexKLA =
dt[siNONEtOH_vs_siNONdexKLA_reg == 1, Gene]),
               list(siNONEtOH_vs_siEP300EtOH = dt[siNONEtOH_vs_siEP300EtOH_reg == 1, Gene],
siNONdex_vs_siEP300dex = dt[siNONdex_vs_siEP300dex_reg == 1, Gene], siNONKLA_vs_siEP300KLA =
dt[siNONKLA_vs_siEP300KLA_reg == 1, Gene], siNONdexKLA_vs_siEP300dexKLA =
dt[siNONdexKLA_vs_siEP300dexKLA_reg == 1, Gene]),
               list(siNONEtOH_vs_siNONdex = dt[siNONEtOH_vs_siNONdex_reg == 1, Gene],
siNONdex_vs_siEP300dex = dt[siNONdex_vs_siEP300dex_reg == 1, Gene], siEP300EtOH_vs_siEP300dex =
dt[siEP300EtOH_vs_siEP300dex_reg == 1, Gene]),
               list(siNONEtOH_vs_siNONKLA = dt[siNONEtOH_vs_siNONKLA_reg == 1, Gene],
siNONKLA_vs_siEP300KLA = dt[siNONKLA_vs_siEP300KLA_reg == 1, Gene], siEP300EtOH_vs_siEP300KLA =
dt[siEP300EtOH_vs_siEP300KLA_reg == 1, Gene]),
               list(siNONEtOH_vs_siNONdexKLA = dt[siNONEtOH_vs_siNONdexKLA_reg == 1, Gene],
siNONdexKLA_vs_siEP300dexKLA = dt[siNONdexKLA_vs_siEP300dexKLA_reg == 1, Gene],
siEP300EtOH_vs_siEP300dexKLA = dt[siEP300EtOH_vs_siEP300dexKLA_reg == 1, Gene]),
               list(siEP300EtOH_vs_siEP300dex = dt[siEP300EtOH_vs_siEP300dex_reg == 1, Gene],
siEP300EtOH_vs_siEP300KLA = dt[siEP300EtOH_vs_siEP300KLA_reg == 1, Gene],
siEP300EtOH_vs_siEP300dexKLA = dt[siEP300EtOH_vs_siEP300dexKLA_reg == 1, Gene]),
               list(siNONdex_vs_siNONdexKLA = dt[siNONdex_vs_siNONdexKLA_reg == 1, Gene],
siEP300dex_vs_siEP300dexKLA = dt[siEP300dex_vs_siEP300dexKLA_reg == 1, Gene]),
               list(siNONKLA_vs_siNONdexKLA = dt[siNONKLA_vs_siNONdexKLA_reg == 1, Gene],
siEP300KLA_vs_siEP300dexKLA = dt[siEP300KLA_vs_siEP300dexKLA_reg == 1, Gene]),
               list(siNONEtOH_vs_siNONdex = dt[siNONEtOH_vs_siNONdex_reg == 1, Gene],
siNONKLA_vs_siNONdexKLA = dt[siNONKLA_vs_siNONdexKLA_reg == 1, Gene], siNONEtOH_vs_siNONdexKLA =
dt[siNONEtOH_vs_siNONdexKLA_reg == 1, Gene]),
               list(siEP300EtOH_vs_siEP300dex = dt[siEP300EtOH_vs_siEP300dex_reg == 1, Gene],
siEP300KLA_vs_siEP300dexKLA = dt[siEP300KLA_vs_siEP300dexKLA_reg == 1, Gene],
siEP300EtOH_vs_siEP300dexKLA = dt[siEP300EtOH_vs_siEP300dexKLA_reg == 1, Gene])
)

#within the loop, process each comparison with ggVennDiagram commands and add columns with number of
up and downregulated genes for each area of the Venn diagram depending on how many comparisons are
plotted
for(i in 1:length(toVenn)){
```

```
    venndata <- process_data(Venn(toVenn[[i]]))

    #next the number of up and dn regulated genes are taken
    #and added as new columns to the resulting tibble
    #it is critical that the counts are input in the correct order
    if(length(toVenn[[i]]) == 2){
      #in the resulting tibbles, the counts are in the following order
      # 1, 2, 12
      #where 1 is the first , 2 is the second and 12 is their intersect
      #the fields in the resulting object can be accessed with @
      #and the columns of the tibble like data.frames with $
      venndata@region$up   <-   c(nrow(dt[(get(paste0(names(toVenn[[i]][1]),   "_reg"))   ==   1   &
  get(paste0(names(toVenn[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][1]), "_up")) == 1)]),
                                  nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    0    &
  get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][2]), "_up")) == 1)]),
                                  nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    1    &
  get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][1]), "_up")) == 1 &
  get(paste0(names(toVenn[[i]][2]), "_up")) == 1)])
      )
      venndata@region$dn   <-   c(nrow(dt[(get(paste0(names(toVenn[[i]][1]),   "_reg"))   ==   1   &
  get(paste0(names(toVenn[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][1]), "_dn")) == 1)]),
                                  nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    0    &
  get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][2]), "_dn")) == 1)]),
                                  nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    1    &
  get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][1]), "_dn")) == 1 &
  get(paste0(names(toVenn[[i]][2]), "_dn")) == 1)])
      )
    } else if(length(toVenn[[i]]) == 3){
      #with three items to compare, the order is this:
      # 1, 2, 3, 12, 13, 23, 123
      venndata@region$up   <-   c(nrow(dt[(get(paste0(names(toVenn[[i]][1]),   "_reg"))   ==   1   &
  get(paste0(names(toVenn[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 0 &
  get(paste0(names(toVenn[[i]][1]), "_up")) == 1)]),
                                  nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    0    &
  get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 0 &
  get(paste0(names(toVenn[[i]][2]), "_up")) == 1)]),
                                  nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    0    &
  get(paste0(names(toVenn[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 1 &
  get(paste0(names(toVenn[[i]][3]), "_up")) == 1)]),
                                  nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    1    &
  get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 0 &
  get(paste0(names(toVenn[[i]][1]), "_up")) == 1 & get(paste0(names(toVenn[[i]][2]), "_up"))== 1)]),
                                  nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    1    &
  get(paste0(names(toVenn[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 1 &
  get(paste0(names(toVenn[[i]][1]), "_up")) == 1 & get(paste0(names(toVenn[[i]][3]), "_up")) == 1)]),
                                  nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    0    &
  get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 1 &
  get(paste0(names(toVenn[[i]][2]), "_up")) == 1 & get(paste0(names(toVenn[[i]][3]), "_up")) == 1)]),
                                  nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    1    &
  get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 1 &
  get(paste0(names(toVenn[[i]][1]), "_up")) == 1 & get(paste0(names(toVenn[[i]][2]), "_up")) == 1 &
  get(paste0(names(toVenn[[i]][3]), "_up")) == 1)])
      )
      venndata@region$dn   <-   c(nrow(dt[(get(paste0(names(toVenn[[i]][1]),   "_reg"))   ==   1   &
  get(paste0(names(toVenn[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 0 &
  get(paste0(names(toVenn[[i]][1]), "_dn")) == 1)]),
```

```
                              nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    0    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 0 &
get(paste0(names(toVenn[[i]][2]), "_dn")) == 1)]),
                              nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    0    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 1 &
get(paste0(names(toVenn[[i]][3]), "_dn")) == 1)]),
                              nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    1    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 0 &
get(paste0(names(toVenn[[i]][1]), "_dn")) == 1 & get(paste0(names(toVenn[[i]][2]), "_dn"))== 1)]),
                              nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    1    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 1 &
get(paste0(names(toVenn[[i]][1]), "_dn")) == 1 & get(paste0(names(toVenn[[i]][3]), "_dn")) == 1)]),
                              nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    0    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 1 &
get(paste0(names(toVenn[[i]][2]), "_dn")) == 1 & get(paste0(names(toVenn[[i]][3]), "_dn")) == 1)]),
                              nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    1    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 1 &
get(paste0(names(toVenn[[i]][1]), "_dn")) == 1 & get(paste0(names(toVenn[[i]][2]), "_dn")) == 1 &
get(paste0(names(toVenn[[i]][3]), "_dn")) == 1)])
    )
  } else if(length(toVenn[[i]]) == 4){
    #with four items to compare, the order is this:
    # 1, 2, 3, 4, 12, 13, 14, 23, 24, 34, 123, 124, 134, 234, 1234
    venndata@region$up   <-   c(nrow(dt[(get(paste0(names(toVenn[[i]][1]),   "_reg"))   ==   1   &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 0 &
get(paste0(names(toVenn[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][1]), "_up")) == 1)]),
#1
                              nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    0    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 0 &
get(paste0(names(toVenn[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][2]), "_up")) == 1)]),
#2
                              nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    0    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 1 &
get(paste0(names(toVenn[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][3]), "_up")) == 1)]),
#3
                              nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    0    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 0 &
get(paste0(names(toVenn[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][4]), "_up")) == 1)]),
#4
                              nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    1    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 0 &
get(paste0(names(toVenn[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][1]), "_up")) == 1 &
get(paste0(names(toVenn[[i]][2]), "_up")) == 1)]),
                              nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    1    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 1 &
get(paste0(names(toVenn[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][1]), "_up")) == 1 &
get(paste0(names(toVenn[[i]][3]), "_up")) == 1)]),
                              nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    1    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 0 &
get(paste0(names(toVenn[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][1]), "_up")) == 1 &
get(paste0(names(toVenn[[i]][4]), "_up")) == 1)]),
                              nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    0    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 1 &
get(paste0(names(toVenn[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][2]), "_up")) == 1 &
get(paste0(names(toVenn[[i]][3]), "_up")) == 1)]),
                              nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    0    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 0 &
```

```
get(paste0(names(toVenn[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][2]), "_up")) == 1 &
get(paste0(names(toVenn[[i]][4]), "_up")) == 1)]),
                            nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    0    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 1 &
get(paste0(names(toVenn[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_up")) == 1 &
get(paste0(names(toVenn[[i]][4]), "_up")) == 1)]),
                            nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    1    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 1 &
get(paste0(names(toVenn[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][1]), "_up")) == 1 &
get(paste0(names(toVenn[[i]][2]), "_up")) == 1 & get(paste0(names(toVenn[[i]][3]), "_up")) == 1)]),
                            nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    1    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 0 &
get(paste0(names(toVenn[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][1]), "_up")) == 1 &
get(paste0(names(toVenn[[i]][2]), "_up")) == 1 & get(paste0(names(toVenn[[i]][4]), "_up")) == 1)]),
                            nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    1    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 1 &
get(paste0(names(toVenn[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][1]), "_up")) == 1 &
get(paste0(names(toVenn[[i]][3]), "_up")) == 1 & get(paste0(names(toVenn[[i]][4]), "_up")) == 1)]),
                            nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    0    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 1 &
get(paste0(names(toVenn[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][2]), "_up")) == 1 &
get(paste0(names(toVenn[[i]][3]), "_up")) == 1 & get(paste0(names(toVenn[[i]][4]), "_up")) == 1)]),
                            nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    1    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 1 &
get(paste0(names(toVenn[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][1]), "_up")) == 1 &
get(paste0(names(toVenn[[i]][2]), "_up")) == 1 & get(paste0(names(toVenn[[i]][3]), "_up")) == 1 &
get(paste0(names(toVenn[[i]][4]), "_up")) == 1)])
    )
    venndata@region$dn    <-    c(nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    1    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 0 &
get(paste0(names(toVenn[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][1]), "_dn")) == 1)]),
#1
                            nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    0    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 0 &
get(paste0(names(toVenn[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][2]), "_dn")) == 1)]),
#2
                            nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    0    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 1 &
get(paste0(names(toVenn[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][3]), "_dn")) == 1)]),
#3
                            nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    0    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 0 &
get(paste0(names(toVenn[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][4]), "_dn")) == 1)]),
#4
                            nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    1    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 0 &
get(paste0(names(toVenn[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][1]), "_dn")) == 1 &
get(paste0(names(toVenn[[i]][2]), "_dn")) == 1)]),
                            nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    1    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 1 &
get(paste0(names(toVenn[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][1]), "_dn")) == 1 &
get(paste0(names(toVenn[[i]][3]), "_dn")) == 1)]),
                            nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    1    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 0 &
get(paste0(names(toVenn[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][1]), "_dn")) == 1 &
get(paste0(names(toVenn[[i]][4]), "_dn")) == 1)]),
                            nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    0    &
get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 1 &
```

```
  get(paste0(names(toVenn[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][2]), "_dn")) == 1 &
  get(paste0(names(toVenn[[i]][3]), "_dn")) == 1)]),
                              nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    0    &
  get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 0 &
  get(paste0(names(toVenn[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][2]), "_dn")) == 1 &
  get(paste0(names(toVenn[[i]][4]), "_dn")) == 1)]),
                              nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    0    &
  get(paste0(names(toVenn[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 1 &
  get(paste0(names(toVenn[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_dn")) == 1 &
  get(paste0(names(toVenn[[i]][4]), "_dn")) == 1)]),
                              nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    1    &
  get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 1 &
  get(paste0(names(toVenn[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][1]), "_dn")) == 1 &
  get(paste0(names(toVenn[[i]][2]), "_dn")) == 1 & get(paste0(names(toVenn[[i]][3]), "_dn")) == 1)]),
                              nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    1    &
  get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 0 &
  get(paste0(names(toVenn[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][1]), "_dn")) == 1 &
  get(paste0(names(toVenn[[i]][2]), "_dn")) == 1 & get(paste0(names(toVenn[[i]][4]), "_dn")) == 1)]),
                              nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    1    &
  get(paste0(names(toVenn[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 1 &
  get(paste0(names(toVenn[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][1]), "_dn")) == 1 &
  get(paste0(names(toVenn[[i]][3]), "_dn")) == 1 & get(paste0(names(toVenn[[i]][4]), "_dn")) == 1)]),
                              nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    0    &
  get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 1 &
  get(paste0(names(toVenn[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][2]), "_dn")) == 1 &
  get(paste0(names(toVenn[[i]][3]), "_dn")) == 1 & get(paste0(names(toVenn[[i]][4]), "_dn")) == 1)]),
                              nrow(dt[(get(paste0(names(toVenn[[i]][1]),    "_reg"))    ==    1    &
  get(paste0(names(toVenn[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][3]), "_reg")) == 1 &
  get(paste0(names(toVenn[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn[[i]][1]), "_dn")) == 1 &
  get(paste0(names(toVenn[[i]][2]), "_dn")) == 1 & get(paste0(names(toVenn[[i]][3]), "_dn")) == 1 &
  get(paste0(names(toVenn[[i]][4]), "_dn")) == 1)])
      )
    }
    #define if the counts by processing Venn data match with manually added ones
    #if there were any genes that were upregulated in one but downregulated in the other
    #the automatic counts will not match
    #this column can be used to annotate the venn diagrams when plotting
    venndata@region$dif <- venndata@region$count - (venndata@region$up + venndata@region$dn)
    #save each separately
    assign(paste0("venndata", i), venndata)
  }

  #create a colour palette with 15 colours for a 4-comparison venn diagram
  adj_RdYlBu <- colorRampPalette(c("#313695", "#4575B4", "#74ADD1", "#ABD9E9", "#E0F3F8", "#FFFFBF",
  "#FEE090", "#FDAE61", "#F46D43", "#D73027", "#A50026" ), space = "rgb")(15)
  new_RdYlBu  <-  c("#A50026",  "#C82226",  "#D95039",  "#F46D43",    "#FDAE61",  "#FEE090",  "#FEEFA7",
  "#FFFFBF", "#BDDEE9", "#E0F3F8", "#A6DDF1", "#74ADD1", "#4575B4", "#6169B8", "#313695")
  #set index based on what order the plotting function plots the colours in
  index <- c(1, 7, 9, 3, 10, 12, 13, 14, 11, 6, 8, 2, 15, 4, 5)
  #order colours according to index
  ordered_new_RdYlBu <- new_RdYlBu[order(index)]

  #then plot each venndata
  #the plots require slightly different x- and y-axis nudges depending on the number of comparisons
  #and different annotations
  #so though lengthy it is easiest to do them one at a time

  #venndata1
```

```
#the venns are plotted as ggplot objects
ggplot()+
  #to draw the outline geom_sf is used
  geom_sf(aes(fill=name), data = venn_region(venndata1), show.legend = FALSE, alpha = 0.5) +
  #the counts are added as labels
  geom_sf_text(aes(label = count), fontface = "bold", data = venn_region(venndata1), size = 5, nudge_y
= 0.7) +
  #then the information about how many of those genes in the counts are up and dn is added
  #if either up or dn or both are 0, they are not plotted
  #sprintf is used to print an arrow pointing up ('\u2191') or down ('\u2193')
  geom_sf_text(aes(label = paste0(#"\n",
    fifelse(up != 0, paste0(up, " ", sprintf('\u2191')), paste0("")),
    fifelse(up != 0 & dn != 0, paste0("  "), paste0("")),
    fifelse(dn != 0, paste0(dn, " ", sprintf('\u2193')), paste0(""))
  )), data = venn_region(venndata1), nudge_y = 0.1) +
  #if there were genes shared and regulated in different directions, add those using the dif column
  geom_sf_text(aes(label = paste0(fifelse(dif > 0 , paste0(dif," ", sprintf('\u2191'),
sprintf('\u2193'), "*"), paste0("")))),
                data = venn_region(venndata1), nudge_y = 0.1) +
  #add titles for the comparisons plotted
  geom_sf_label(aes(label = gsub("_", " ", name)), data = venn_setlabel(venndata1), alpha = 0.5,
label.size = 0, fontface = "bold", size = 5, nudge_y = 0.05) +
  scale_x_continuous(expand = expansion(mult = .3)) +
  scale_fill_brewer(palette = "RdYlBu") +
  annotate(geom = "text", x = -1, y = -5.4, label = paste0("* ", sprintf('\u2191'), " in siNONEtOH vs
siNONKLA, ", sprintf('\u2193'), " in siNONEtOH vs siNONdex"), size = 4) +
  #do not plot any axes
  theme_void()
ggsave("genes_venn1.png", plot = last_plot(), device = "png", width = 8, height = 8, units =  "in")

#venndata2
ggplot()+
  geom_sf(aes(fill=name), data = venn_region(venndata2), show.legend = FALSE, alpha = 0.5) +
  geom_sf_text(aes(label = count), fontface = "bold", data = venn_region(venndata2), size = 5, nudge_y
= 0.02) +
  geom_sf_text(aes(label = paste0(#"\n",
    fifelse(up != 0, paste0(up, " ", sprintf('\u2191')), paste0("")),
    fifelse(up != 0 & dn != 0, paste0("  "), paste0("")),
    fifelse(dn != 0, paste0(dn, " ", sprintf('\u2193')), paste0(""))
  )), data = venn_region(venndata2), nudge_y = -0.01) +
  geom_sf_label(aes(label = gsub("vs ", "vs\n", gsub("_", " ", name))), data =
venn_setlabel(venndata2), alpha = 0.5, label.size = 0, fontface = "bold", size = 5, nudge_y = 0.03)
+
  scale_x_continuous(expand = expansion(mult = .3)) +
  scale_fill_manual(values = ordered_new_RdYlBu) +
  theme_void()
ggsave("genes_venn2.png", plot = last_plot(), device = "png", width = 9, height = 9, units =  "in")

#venndata3
#find out how many and which genes are regulated in different directions in the comparisons
#and use this information for annotation
nrow(dt[(siNONEtOH_vs_siNONdex_reg == 0 & siEP300EtOH_vs_siEP300dex_reg == 1 &
siNONdex_vs_siEP300dex_reg == 1 & siEP300EtOH_vs_siEP300dex_up == 1 & siNONdex_vs_siEP300dex_dn ==
1),])
dt[(siNONEtOH_vs_siNONdex_reg == 0 & siEP300EtOH_vs_siEP300dex_reg == 1 & siNONdex_vs_siEP300dex_reg
== 1 & siEP300EtOH_vs_siEP300dex_up == 1 & siNONdex_vs_siEP300dex_dn == 1), Gene]
```

```
nrow(dt[(siNONEtOH_vs_siNONdex_reg == 1 & siNONdex_vs_siEP300dex_reg == 1 &
siEP300EtOH_vs_siEP300dex_reg == 0 & siNONEtOH_vs_siNONdex_up == 1 & siNONdex_vs_siEP300dex_dn ==
1),])
dt[(siNONEtOH_vs_siNONdex_reg == 1 & siNONdex_vs_siEP300dex_reg == 1 & siEP300EtOH_vs_siEP300dex_reg
== 0 & siNONEtOH_vs_siNONdex_up == 1 & siNONdex_vs_siEP300dex_dn == 1), Gene]
nrow(dt[(siNONEtOH_vs_siNONdex_reg == 1 & siNONdex_vs_siEP300dex_reg == 1 &
siEP300EtOH_vs_siEP300dex_reg == 0 & siNONEtOH_vs_siNONdex_dn == 1 & siNONdex_vs_siEP300dex_up ==
1),])
dt[(siNONEtOH_vs_siNONdex_reg == 1 & siNONdex_vs_siEP300dex_reg == 1 & siEP300EtOH_vs_siEP300dex_reg
== 0 & siNONEtOH_vs_siNONdex_dn == 1 & siNONdex_vs_siEP300dex_up == 1), Gene]
nrow(dt[(siNONEtOH_vs_siNONdex_reg == 1 & siNONdex_vs_siEP300dex_reg == 1 &
siEP300EtOH_vs_siEP300dex_reg == 1 & siNONEtOH_vs_siNONdex_up == 1 & siNONdex_vs_siEP300dex_dn == 1
& siEP300EtOH_vs_siEP300dex_up == 1),])
dt[(siNONEtOH_vs_siNONdex_reg == 1 & siNONdex_vs_siEP300dex_reg == 1 & siEP300EtOH_vs_siEP300dex_reg
== 1 & siNONEtOH_vs_siNONdex_up == 1 & siNONdex_vs_siEP300dex_dn == 1 & siEP300EtOH_vs_siEP300dex_up
== 1), Gene]

ggplot()+
  geom_sf(aes(fill=name), data = venn_region(venndata3), show.legend = FALSE, alpha = 0.5) +
  geom_sf_text(aes(label = count), fontface = "bold", data = venn_region(venndata3), size = 5, nudge_y
= 0.5) +
  geom_sf_text(aes(label = paste0(#"\n",
    fifelse(up != 0, paste0(up, " ", sprintf('\u2191')), paste0("")),
    fifelse(up != 0 & dn != 0, paste0("  "), paste0("")),
    fifelse(dn != 0, paste0(dn, " ", sprintf('\u2193')), paste0(""))
  )), data = venn_region(venndata3), nudge_y = -0.01) +
  geom_sf_text(aes(label = paste0(fifelse(dif > 0 , paste0(dif," ", sprintf('\u2191'),
sprintf('\u2193')), paste0("")))),
                data = venn_region(venndata3), nudge_y = -0.55) +
  geom_sf_label(aes(label = gsub("_", " ", name)), data = venn_setlabel(venndata3), alpha = 0.5,
label.size = 0, fontface = "bold", size = 5, nudge_y = 0.03) +
  scale_x_continuous(expand = expansion(mult = .3)) +
  scale_y_continuous(expand = expansion(mult = .08)) +
  scale_fill_manual(values = adj_RdYlBu[c(13, 11, 9, 5, 1, 3, 7)]) +
  annotate(geom = "text", x = 0.2, y = 1.8, label = "*1", size = 3) +
  annotate(geom = "text", x = 2.6, y = 1.3, label = "*2", size = 3) +
  annotate(geom = "text", x = 2.5, y = -1.9, label = "*3", size = 3) +
  annotate(geom = "text", x = -2.3, y = -5.9, label = paste0("* 1) 38 ", sprintf('\u2191'), " in
siNONEtOH vs siNONdex, ", sprintf('\u2193'), " in siNONdex vs siEP300dex", "\n            8 ",
sprintf('\u2193'), " in siNONEtOH vs siNONdex, ", sprintf('\u2191'), " in siNONdex vs siEP300dex",
"\n\t\t\t                                                  2) ", sprintf('\u2191'), " in siNONEtOH
vs siNONdex and siEP300EtOH vs siEP300dex, ", sprintf('\u2193'), " in siNONdex vs siEP300dex",   "\n
3) ", sprintf('\u2191'), " in siEP300EtOH vs siEP300dex, ", sprintf('\u2193'), " in siNONdex vs
siEP300dex"), size = 3) +
  theme_void()
ggsave("genes_venn3.png", plot = last_plot(), device = "png", width = 9, height = 9, units =  "in")


#venndata4
nrow(dt[(siNONEtOH_vs_siNONKLA_reg == 1 & siNONKLA_vs_siEP300KLA_reg == 1 & siNONEtOH_vs_siNONKLA_up
== 1 & siNONKLA_vs_siEP300KLA_dn == 1)])
dt[(siNONEtOH_vs_siNONKLA_reg == 1 & siNONKLA_vs_siEP300KLA_reg == 1 & siNONEtOH_vs_siNONKLA_up == 1
& siNONKLA_vs_siEP300KLA_dn == 1), Gene]

ggplot()+
  geom_sf(aes(fill=name), data = venn_region(venndata4), show.legend = FALSE, alpha = 0.5) +
  geom_sf_text(aes(label = count), fontface = "bold", data = venn_region(venndata4), size = 5, nudge_y
= 0.5) +
```

```
  geom_sf_text(aes(label = paste0(#"\n",
    fifelse(up != 0, paste0(up, " ", sprintf('\u2191')), paste0("")),
    fifelse(up != 0 & dn != 0, paste0("  "), paste0("")),
    fifelse(dn != 0, paste0(dn, " ", sprintf('\u2193')), paste0(""))
  )), data = venn_region(venndata4), nudge_y = -0.01) +
  geom_sf_text(aes(label = paste0(fifelse(dif > 0 , paste0(dif," ", sprintf('\u2191'),
sprintf('\u2193'), " *"), paste0(""))),
              data = venn_region(venndata4), nudge_y = -0.1) +
  geom_sf_label(aes(label = gsub("_", " ", name)), data = venn_setlabel(venndata4), alpha = 0.5,
label.size = 0, fontface = "bold", size = 5, nudge_y = 0.03) +
  scale_x_continuous(expand = expansion(mult = .3)) +
  scale_fill_manual(values = adj_RdYlBu[c(1, 3, 5, 7, 9, 11, 13)]) +
  annotate(geom = "text", x = -2.6, y = -5.2, label = paste0("* ", sprintf('\u2191'), " in siNONEtOH
vs siNONKLA, ", sprintf('\u2193'), " in siNONKLA vs siEP300KLA"), size = 3) +
  theme_void()
ggsave("genes_venn4.png", plot = last_plot(), device = "png", width = 8, height = 8, units =  "in")


#venndata5
nrow(dt[(siNONEtOH_vs_siNONdexKLA_reg == 0 & siEP300EtOH_vs_siEP300dexKLA_reg == 1 &
siNONdexKLA_vs_siEP300dexKLA_reg == 1 & siEP300EtOH_vs_siEP300dexKLA_up == 1 &
siNONdexKLA_vs_siEP300dexKLA_dn == 1),]) #4
dt[(siNONEtOH_vs_siNONdexKLA_reg == 0 & siEP300EtOH_vs_siEP300dexKLA_reg == 1 &
siNONdexKLA_vs_siEP300dexKLA_reg == 1 & siEP300EtOH_vs_siEP300dexKLA_up == 1 &
siNONdexKLA_vs_siEP300dexKLA_dn == 1), Gene]
nrow(dt[(siNONEtOH_vs_siNONdexKLA_reg == 1 & siNONdexKLA_vs_siEP300dexKLA_reg == 1 &
siEP300EtOH_vs_siEP300dexKLA_reg == 0 & siNONEtOH_vs_siNONdexKLA_up == 1 &
siNONdexKLA_vs_siEP300dexKLA_dn == 1),])
dt[(siNONEtOH_vs_siNONdexKLA_reg == 1 & siNONdexKLA_vs_siEP300dexKLA_reg == 1 &
siEP300EtOH_vs_siEP300dexKLA_reg == 0 & siNONEtOH_vs_siNONdexKLA_up == 1 &
siNONdexKLA_vs_siEP300dexKLA_dn == 1), Gene]
nrow(dt[(siNONEtOH_vs_siNONdexKLA_reg == 1 & siNONdexKLA_vs_siEP300dexKLA_reg == 1 &
siEP300EtOH_vs_siEP300dexKLA_reg == 0 & siNONEtOH_vs_siNONdexKLA_dn == 1 &
siNONdexKLA_vs_siEP300dexKLA_up == 1),])
dt[(siNONEtOH_vs_siNONdexKLA_reg == 1 & siNONdexKLA_vs_siEP300dexKLA_reg == 1 &
siEP300EtOH_vs_siEP300dexKLA_reg == 0 & siNONEtOH_vs_siNONdexKLA_dn == 1 &
siNONdexKLA_vs_siEP300dexKLA_up == 1), Gene]
nrow(dt[(siNONEtOH_vs_siNONdexKLA_reg == 1 & siNONdexKLA_vs_siEP300dexKLA_reg == 1 &
siEP300EtOH_vs_siEP300dexKLA_reg == 1 & siNONEtOH_vs_siNONdexKLA_up == 1 &
siNONdexKLA_vs_siEP300dexKLA_dn == 1 & siEP300EtOH_vs_siEP300dexKLA_up == 1),]) #21
dt[(siNONEtOH_vs_siNONdexKLA_reg == 1 & siNONdexKLA_vs_siEP300dexKLA_reg == 1 &
siEP300EtOH_vs_siEP300dexKLA_reg == 1 & siNONEtOH_vs_siNONdexKLA_up == 1 &
siNONdexKLA_vs_siEP300dexKLA_dn == 1 & siEP300EtOH_vs_siEP300dexKLA_up == 1), Gene]

ggplot()+
  geom_sf(aes(fill=name), data = venn_region(venndata5), show.legend = FALSE, alpha = 0.5) +
  geom_sf_text(aes(label = count), fontface = "bold", data = venn_region(venndata5), size = 5, nudge_y
= 0.6) +
  geom_sf_text(aes(label = paste0(#"\n",
    fifelse(up != 0, paste0(up, " ", sprintf('\u2191')), paste0("")),
    fifelse(up != 0 & dn != 0, paste0("  "), paste0("")),
    fifelse(dn != 0, paste0(dn, " ", sprintf('\u2193')), paste0(""))
  )), data = venn_region(venndata5), nudge_y = 0.02) +
  geom_sf_text(aes(label = paste0(fifelse(dif > 0 , paste0(dif," ", sprintf('\u2191'),
sprintf('\u2193')), paste0(""))),
              data = venn_region(venndata5), nudge_y = -0.5) +
  geom_sf_label(aes(label = gsub("_", " ", name)), data = venn_setlabel(venndata5), alpha = 0.5,
label.size = 0, fontface = "bold", size = 5, nudge_y = 0.03) +
```

```
  scale_x_continuous(expand = expansion(mult = .5)) +
  scale_y_continuous(expand = expansion(mult = .1)) +
  scale_fill_manual(values = adj_RdYlBu[c(9, 13, 5, 3, 1, 11, 7)]) +
  annotate(geom = "text", x = 0.1, y = 1.8, label = "*1", size = 3) +
  annotate(geom = "text", x = 2.6, y = 1.35, label = "*2", size = 3) +
  annotate(geom = "text", x = 2.55, y = -1.9, label = "*3", size = 3) +
  annotate(geom = "text", x = -2.2, y = -5.9, label = paste0("* 1) 10 ", sprintf('\u2191'), " in
siNONEtOH vs siNONdexKLA, ", sprintf('\u2193'), " in siNONdexKLA vs siEP300dexKLA", "\n        1 ",
sprintf('\u2193'), " in siNONEtOH vs siNONdexKLA, ", sprintf('\u2191'), " in siNONdexKLA vs
siEP300dexKLA",  "\n\t\t\t                                                                    2) ",
sprintf('\u2191'), " in siNONEtOH vs siNONdexKLA and siEP300EtOH vs siEP300dexKLA, ",
sprintf('\u2193'), " in siNONdexKLA vs siEP300dexKLA",    "\n    3) ", sprintf('\u2191'), " in
siEP300EtOH vs siEP300dexKLA, ", sprintf('\u2193'), " in siNONdexKLA vs siEP300dexKLA"), size = 3) +
  theme_void()
ggsave("genes_venn5.png", plot = last_plot(), device = "png", width = 9, height = 9, units =  "in")

#venndata6
ggplot()+
  geom_sf(aes(fill=name), data = venn_region(venndata6), show.legend = FALSE, alpha = 0.5) +
  geom_sf_text(aes(label = count), fontface = "bold", data = venn_region(venndata6), size = 5, nudge_y
= 0.5) +
  geom_sf_text(aes(label = paste0(#"\n",
    fifelse(up != 0, paste0(up, " ", sprintf('\u2191')), paste0("")),
    fifelse(up != 0 & dn != 0, paste0("  "), paste0("")),
    fifelse(dn != 0, paste0(dn, " ", sprintf('\u2193')), paste0(""))
  )), data = venn_region(venndata6), nudge_y = -0.01) +
  geom_sf_text(aes(label  =  paste0(fifelse(dif  >  0  ,  paste0(dif,"  ",  sprintf('\u2191'),
sprintf('\u2193')), paste0("")))),
                data = venn_region(venndata6), nudge_y = -0.5) +
  geom_sf_label(aes(label = gsub("_", " ", name)), data = venn_setlabel(venndata6), alpha = 0.5,
label.size = 0, fontface = "bold", size = 5, nudge_y = 0.03) +
  scale_x_continuous(expand = expansion(mult = .3)) +
  scale_fill_brewer(palette = "RdYlBu") +
  theme_void()
ggsave("genes_venn6.png", plot = last_plot(), device = "png", width = 8, height = 8, units =  "in")

#venndata7
ggplot()+
  geom_sf(aes(fill=name), data = venn_region(venndata7), show.legend = FALSE, alpha = 0.5) +
  geom_sf_text(aes(label = count), fontface = "bold", data = venn_region(venndata7), size = 5, nudge_y
= 0.5) +
  geom_sf_text(aes(label = paste0(#"\n",
    fifelse(up != 0, paste0(up, " ", sprintf('\u2191')), paste0("")),
    fifelse(up != 0 & dn != 0, paste0("  "), paste0("")),
    fifelse(dn != 0, paste0(dn, " ", sprintf('\u2193')), paste0(""))
  )), data = venn_region(venndata7), nudge_y = -0.01) +
  geom_sf_text(aes(label  =  paste0(fifelse(dif  >  0  ,  paste0(dif,"  ",  sprintf('\u2191'),
sprintf('\u2193')), paste0("")))),
                data = venn_region(venndata7), nudge_y = -0.5) +
  geom_sf_label(aes(label  =  gsub("vs  ",  "vs\n",  gsub("_",  "  ",  name))),  data  =
venn_setlabel(venndata7), alpha = 0.5, label.size = 0, fontface = "bold", size = 5, nudge_y = 0.5) +
  scale_y_continuous(expand = expansion(mult = .3)) +
  scale_x_continuous(expand = expansion(mult = .3)) +
  scale_fill_brewer(palette = "RdYlBu") +
  theme_void()
ggsave("genes_venn7.png", plot = last_plot(), device = "png", width = 7, height = 5, units =  "in")

#venndata8
```

```
ggplot()+
  geom_sf(aes(fill=name), data = venn_region(venndata8), show.legend = FALSE, alpha = 0.5) +
  geom_sf_text(aes(label = count), fontface = "bold", data = venn_region(venndata8), size = 5, nudge_y
= 0.5) +
  geom_sf_text(aes(label = paste0(#"\n",
    fifelse(up != 0, paste0(up, " ", sprintf('\u2191')), paste0("")),
    fifelse(up != 0 & dn != 0, paste0("  "), paste0("")),
    fifelse(dn != 0, paste0(dn, " ", sprintf('\u2193')), paste0(""))
  )), data = venn_region(venndata8), nudge_y = -0.01) +
  geom_sf_text(aes(label = paste0(fifelse(dif > 0 , paste0(dif," ", sprintf('\u2191'),
sprintf('\u2193')), paste0(""))))),
                data = venn_region(venndata8), nudge_y = -0.5) +
  geom_sf_label(aes(label = gsub("vs ", "vs\n", gsub("_", " ", name))), data =
venn_setlabel(venndata8), alpha = 0.5, label.size = 0, fontface = "bold", size = 5, nudge_y = 0.5) +
  scale_x_continuous(expand = expansion(mult = .3)) +
  scale_y_continuous(expand = expansion(mult = .3)) +
  scale_fill_brewer(palette = "RdYlBu") +
  theme_void()
ggsave("genes_venn8.png", plot = last_plot(), device = "png", width = 7, height = 5, units =  "in")

#venndata9
ggplot()+
  geom_sf(aes(fill=name), data = venn_region(venndata9), show.legend = FALSE, alpha = 0.5) +
  geom_sf_text(aes(label = count), fontface = "bold", data = venn_region(venndata9), size = 5, nudge_y
= 0.5) +
  geom_sf_text(aes(label = paste0(#"\n",
    fifelse(up != 0, paste0(up, " ", sprintf('\u2191')), paste0("")),
    fifelse(up != 0 & dn != 0, paste0("  "), paste0("")),
    fifelse(dn != 0, paste0(dn, " ", sprintf('\u2193')), paste0(""))
  )), data = venn_region(venndata9), nudge_y = -0.01) +
  geom_sf_text(aes(label = paste0(fifelse(dif > 0 , paste0(dif," ", sprintf('\u2191'),
sprintf('\u2193'), " *"), paste0(""))))),
                data = venn_region(venndata9), nudge_y = -0.1) +
  geom_sf_label(aes(label = gsub("_", " ", name)), data = venn_setlabel(venndata9), alpha = 0.5,
label.size = 0, fontface = "bold", size = 5, nudge_y = 0.03) +
  scale_x_continuous(expand = expansion(mult = .3)) +
  scale_fill_manual(values = adj_RdYlBu[c(1, 3, 5, 7, 9, 11, 13)]) +
  theme_void()
ggsave("genes_venn9.png", plot = last_plot(), device = "png", width = 8, height = 8, units =  "in")

#venndata10
ggplot()+
  geom_sf(aes(fill=name), data = venn_region(venndata10), show.legend = FALSE, alpha = 0.5) +
  geom_sf_text(aes(label = count), fontface = "bold", data = venn_region(venndata10), size = 5,
nudge_y = 0.5) +
  geom_sf_text(aes(label = paste0(#"\n",
    fifelse(up != 0, paste0(up, " ", sprintf('\u2191')), paste0("")),
    fifelse(up != 0 & dn != 0, paste0("  "), paste0("")),
    fifelse(dn != 0, paste0(dn, " ", sprintf('\u2193')), paste0(""))
  )), data = venn_region(venndata10), nudge_y = -0.01) +
  geom_sf_text(aes(label = paste0(fifelse(dif > 0 , paste0(dif," ", sprintf('\u2191'),
sprintf('\u2193'), " *"), paste0(""))))),
                data = venn_region(venndata10), nudge_y = -0.1) +
  geom_sf_label(aes(label = gsub("_", " ", name)), data = venn_setlabel(venndata10), alpha = 0.5,
label.size = 0, fontface = "bold", size = 5, nudge_y = 0.03) +
  scale_x_continuous(expand = expansion(mult = .3)) +
  scale_fill_manual(values = adj_RdYlBu[c(1, 3, 5, 7, 9, 11, 13)]) +
  theme_void()
```

```
ggsave("genes_venn10.png", plot = last_plot(), device = "png", width = 8, height = 8, units =  "in")


###########
##ENHANCERS

toVenn_enh <- list(list(siNONEtOH_vs_siNONdex = dt_enh[siNONEtOH_vs_siNONdex_reg == 1, Gene],
siNONEtOH_vs_siNONKLA = dt_enh[siNONEtOH_vs_siNONKLA_reg == 1, Gene], siNONEtOH_vs_siNONdexKLA =
dt_enh[siNONEtOH_vs_siNONdexKLA_reg == 1, Gene]),
                   list(siNONEtOH_vs_siEP300EtOH = dt_enh[siNONEtOH_vs_siEP300EtOH_reg == 1, Gene],
siNONdex_vs_siEP300dex = dt_enh[siNONdex_vs_siEP300dex_reg == 1, Gene], siNONKLA_vs_siEP300KLA =
dt_enh[siNONKLA_vs_siEP300KLA_reg == 1, Gene], siNONdexKLA_vs_siEP300dexKLA =
dt_enh[siNONdexKLA_vs_siEP300dexKLA_reg == 1, Gene]),
                   list(siNONEtOH_vs_siNONdex = dt_enh[siNONEtOH_vs_siNONdex_reg == 1, Gene],
siNONdex_vs_siEP300dex = dt_enh[siNONdex_vs_siEP300dex_reg == 1, Gene], siEP300EtOH_vs_siEP300dex =
dt_enh[siEP300EtOH_vs_siEP300dex_reg == 1, Gene]),
                   list(siNONEtOH_vs_siNONKLA = dt_enh[siNONEtOH_vs_siNONKLA_reg == 1, Gene],
siNONKLA_vs_siEP300KLA = dt_enh[siNONKLA_vs_siEP300KLA_reg == 1, Gene], siEP300EtOH_vs_siEP300KLA =
dt_enh[siEP300EtOH_vs_siEP300KLA_reg == 1, Gene]),
                   list(siNONEtOH_vs_siNONdexKLA = dt_enh[siNONEtOH_vs_siNONdexKLA_reg == 1, Gene],
siNONdexKLA_vs_siEP300dexKLA = dt_enh[siNONdexKLA_vs_siEP300dexKLA_reg == 1, Gene],
siEP300EtOH_vs_siEP300dexKLA = dt_enh[siEP300EtOH_vs_siEP300dexKLA_reg == 1, Gene]),
                   list(siEP300EtOH_vs_siEP300dex = dt_enh[siEP300EtOH_vs_siEP300dex_reg == 1, Gene],
siEP300EtOH_vs_siEP300KLA = dt_enh[siEP300EtOH_vs_siEP300KLA_reg == 1, Gene],
siEP300EtOH_vs_siEP300dexKLA = dt_enh[siEP300EtOH_vs_siEP300dexKLA_reg == 1, Gene]),
                   list(siNONdex_vs_siNONdexKLA = dt_enh[siNONdex_vs_siNONdexKLA_reg == 1, Gene],
siEP300dex_vs_siEP300dexKLA = dt_enh[siEP300dex_vs_siEP300dexKLA_reg == 1, Gene]),
                   list(siNONKLA_vs_siNONdexKLA = dt_enh[siNONKLA_vs_siNONdexKLA_reg == 1, Gene],
siEP300KLA_vs_siEP300dexKLA = dt_enh[siEP300KLA_vs_siEP300dexKLA_reg == 1, Gene]),
                   list(siNONEtOH_vs_siNONdex = dt_enh[siNONEtOH_vs_siNONdex_reg == 1, Gene],
siNONKLA_vs_siNONdexKLA = dt_enh[siNONKLA_vs_siNONdexKLA_reg == 1, Gene], siNONEtOH_vs_siNONdexKLA =
dt_enh[siNONEtOH_vs_siNONdexKLA_reg == 1, Gene]),
                   list(siEP300EtOH_vs_siEP300dex = dt_enh[siEP300EtOH_vs_siEP300dex_reg == 1, Gene],
siEP300KLA_vs_siEP300dexKLA = dt_enh[siEP300KLA_vs_siEP300dexKLA_reg == 1, Gene],
siEP300EtOH_vs_siEP300dexKLA = dt_enh[siEP300EtOH_vs_siEP300dexKLA_reg == 1, Gene])
)


for(i in 1:length(toVenn_enh)){
  venndata <- process_data(Venn(toVenn_enh[[i]]))

  if(length(toVenn_enh[[i]]) == 2){
    venndata@region$up <- c(nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]), "_reg")) == 1 &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][1]), "_up"))
== 1)]),
                           nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]), "_reg")) == 0 &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][2]), "_up"))
== 1)]),
                           nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]), "_reg")) == 1 &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][1]), "_up"))
== 1 & get(paste0(names(toVenn_enh[[i]][2]), "_up")) == 1)])
    )
    venndata@region$dn <- c(nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]), "_reg")) == 1 &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][1]), "_dn"))
== 1)]),
                           nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]), "_reg")) == 0 &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][2]), "_dn"))
== 1)]),
```

```
                              nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  1  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][1]), "_dn"))
== 1 & get(paste0(names(toVenn_enh[[i]][2]), "_dn")) == 1)])
    )
  } else if(length(toVenn_enh[[i]]) == 3){
    # 1, 2, 3, 12, 13, 23, 123
    venndata@region$up  <-  c(nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  1  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 0 & get(paste0(names(toVenn_enh[[i]][1]), "_up")) == 1)]),
                              nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  0  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 0 & get(paste0(names(toVenn_enh[[i]][2]), "_up")) == 1)]),
                              nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  0  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 1 & get(paste0(names(toVenn_enh[[i]][3]), "_up")) == 1)]),
                              nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  1  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 0 & get(paste0(names(toVenn_enh[[i]][1]), "_up")) == 1 & get(paste0(names(toVenn_enh[[i]][2]),
"_up"))== 1)]),
                              nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  1  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 1 & get(paste0(names(toVenn_enh[[i]][1]), "_up")) == 1 & get(paste0(names(toVenn_enh[[i]][3]),
"_up")) == 1)]),
                              nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  0  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 1 & get(paste0(names(toVenn_enh[[i]][2]), "_up")) == 1 & get(paste0(names(toVenn_enh[[i]][3]),
"_up")) == 1)]),
                              nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  1  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 1 & get(paste0(names(toVenn_enh[[i]][1]), "_up")) == 1 & get(paste0(names(toVenn_enh[[i]][2]),
"_up")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_up")) == 1)])
    )
    venndata@region$dn  <-  c(nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  1  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 0 & get(paste0(names(toVenn_enh[[i]][1]), "_dn")) == 1)]),
                              nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  0  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 0 & get(paste0(names(toVenn_enh[[i]][2]), "_dn")) == 1)]),
                              nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  0  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 1 & get(paste0(names(toVenn_enh[[i]][3]), "_dn")) == 1)]),
                              nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  1  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 0 & get(paste0(names(toVenn_enh[[i]][1]), "_dn")) == 1 & get(paste0(names(toVenn_enh[[i]][2]),
"_dn"))== 1)]),
                              nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  1  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 1 & get(paste0(names(toVenn_enh[[i]][1]), "_dn")) == 1 & get(paste0(names(toVenn_enh[[i]][3]),
"_dn")) == 1)]),
                              nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  0  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 1 & get(paste0(names(toVenn_enh[[i]][2]), "_dn")) == 1 & get(paste0(names(toVenn_enh[[i]][3]),
"_dn")) == 1)]),
                              nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  1  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 1 & get(paste0(names(toVenn_enh[[i]][1]), "_dn")) == 1 & get(paste0(names(toVenn_enh[[i]][2]),
"_dn")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_dn")) == 1)])
    )
```

```
  } else if(length(toVenn_enh[[i]]) == 4){
    # 1, 2, 3, 4, 12, 13, 14, 23, 24, 34, 123, 124, 134, 234, 1234
    venndata@region$up  <-  c(nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  1  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 0 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][1]),
"_up")) == 1)]), #1
                          nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  0  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 0 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][2]),
"_up")) == 1)]), #2
                          nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  0  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 1 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][3]),
"_up")) == 1)]), #3
                          nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  0  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 0 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][4]),
"_up")) == 1)]), #4
                          nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  1  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 0 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][1]),
"_up")) == 1 & get(paste0(names(toVenn_enh[[i]][2]), "_up")) == 1)]),
                          nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  1  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 1 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][1]),
"_up")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_up")) == 1)]),
                          nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  1  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 0 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][1]),
"_up")) == 1 & get(paste0(names(toVenn_enh[[i]][4]), "_up")) == 1)]),
                          nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  0  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 1 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][2]),
"_up")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_up")) == 1)]),
                          nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  0  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 0 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][2]),
"_up")) == 1 & get(paste0(names(toVenn_enh[[i]][4]), "_up")) == 1)]),
                          nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  0  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 1 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]),
"_up")) == 1 & get(paste0(names(toVenn_enh[[i]][4]), "_up")) == 1)]),
                          nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  1  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 1 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][1]),
"_up"))     ==     1     &     get(paste0(names(toVenn_enh[[i]][2]),     "_up"))     ==     1     &
get(paste0(names(toVenn_enh[[i]][3]), "_up")) == 1)]),
                          nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  1  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 0 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][1]),
"_up"))     ==     1     &     get(paste0(names(toVenn_enh[[i]][2]),     "_up"))     ==     1     &
get(paste0(names(toVenn_enh[[i]][4]), "_up")) == 1)]),
                          nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  1  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 1 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][1]),
"_up"))     ==     1     &     get(paste0(names(toVenn_enh[[i]][3]),     "_up"))     ==     1     &
get(paste0(names(toVenn_enh[[i]][4]), "_up")) == 1)]),
```

```
                            nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  0  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 1 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][2]),
"_up"))     ==     1     &     get(paste0(names(toVenn_enh[[i]][3]),     "_up"))     ==     1     &
get(paste0(names(toVenn_enh[[i]][4]), "_up")) == 1)]),
                            nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  1  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 1 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][1]),
"_up"))     ==     1     &     get(paste0(names(toVenn_enh[[i]][2]),     "_up"))     ==     1     &
get(paste0(names(toVenn_enh[[i]][3]), "_up")) == 1 & get(paste0(names(toVenn_enh[[i]][4]), "_up")) ==
1)])
    )
    venndata@region$dn  <-  c(nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  1  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 0 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][1]),
"_dn")) == 1)]), #1
                            nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  0  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 0 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][2]),
"_dn")) == 1)]), #2
                            nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  0  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 1 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][3]),
"_dn")) == 1)]), #3
                            nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  0  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 0 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][4]),
"_dn")) == 1)]), #4
                            nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  1  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 0 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][1]),
"_dn")) == 1 & get(paste0(names(toVenn_enh[[i]][2]), "_dn")) == 1)]),
                            nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  1  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 1 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][1]),
"_dn")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_dn")) == 1)]),
                            nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  1  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 0 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][1]),
"_dn")) == 1 & get(paste0(names(toVenn_enh[[i]][4]), "_dn")) == 1)]),
                            nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  0  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 1 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][2]),
"_dn")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_dn")) == 1)]),
                            nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  0  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 0 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][2]),
"_dn")) == 1 & get(paste0(names(toVenn_enh[[i]][4]), "_dn")) == 1)]),
                            nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  0  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 1 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]),
"_dn")) == 1 & get(paste0(names(toVenn_enh[[i]][4]), "_dn")) == 1)]),
                            nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  1  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 1 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][1]),
"_dn"))     ==     1     &     get(paste0(names(toVenn_enh[[i]][2]),     "_dn"))     ==     1     &
get(paste0(names(toVenn_enh[[i]][3]), "_dn")) == 1)]),
```

```
                              nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  1  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 0 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][1]),
"_dn"))     ==     1     &     get(paste0(names(toVenn_enh[[i]][2]),     "_dn"))     ==     1     &
get(paste0(names(toVenn_enh[[i]][4]), "_dn")) == 1)]),
                              nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  1  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 0 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 1 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][1]),
"_dn"))     ==     1     &     get(paste0(names(toVenn_enh[[i]][3]),     "_dn"))     ==     1     &
get(paste0(names(toVenn_enh[[i]][4]), "_dn")) == 1)]),
                              nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  0  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 1 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][2]),
"_dn"))     ==     1     &     get(paste0(names(toVenn_enh[[i]][3]),     "_dn"))     ==     1     &
get(paste0(names(toVenn_enh[[i]][4]), "_dn")) == 1)]),
                              nrow(dt_enh[(get(paste0(names(toVenn_enh[[i]][1]),  "_reg"))  ==  1  &
get(paste0(names(toVenn_enh[[i]][2]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][3]), "_reg"))
== 1 & get(paste0(names(toVenn_enh[[i]][4]), "_reg")) == 1 & get(paste0(names(toVenn_enh[[i]][1]),
"_dn"))     ==     1     &     get(paste0(names(toVenn_enh[[i]][2]),     "_dn"))     ==     1     &
get(paste0(names(toVenn_enh[[i]][3]), "_dn")) == 1 & get(paste0(names(toVenn_enh[[i]][4]), "_dn")) ==
1)])
    )
  }

  venndata@region$dif <- venndata@region$count - (venndata@region$up + venndata@region$dn)
  assign(paste0("venndata_enh", i), venndata)
}

#venndata_enh1
ggplot()+
  geom_sf(aes(fill=name), data = venn_region(venndata_enh1), show.legend = FALSE, alpha = 0.5) +
  geom_sf_text(aes(label = count), fontface = "bold", data = venn_region(venndata_enh1), size = 5,
nudge_y = 0.7) +
  geom_sf_text(aes(label = paste0(#"\n",
    fifelse(up != 0, paste0(up, " ", sprintf('\u2191')), paste0("")),
    fifelse(up != 0 & dn != 0, paste0("  "), paste0("")),
    fifelse(dn != 0, paste0(dn, " ", sprintf('\u2193')), paste0(""))
  )), data = venn_region(venndata_enh1), nudge_y = 0.1) +
  geom_sf_text(aes(label  =  paste0(fifelse(dif  >  0  ,  paste0(dif,"  ",  sprintf('\u2191'),
sprintf('\u2193'), "*"), paste0(""))),
                data = venn_region(venndata_enh1), nudge_y = 0.1) +
  geom_sf_label(aes(label = gsub("_", " ", name)), data = venn_setlabel(venndata_enh1), alpha = 0.5,
label.size = 0, fontface = "bold", size = 5, nudge_y = 0.05) +
  scale_x_continuous(expand = expansion(mult = .3)) +
  scale_fill_brewer(palette = "RdYlBu") +
  theme_void()
ggsave("enhancers_venn1.png", plot = last_plot(), device = "png", width = 8, height = 8, units =
"in")

#venndata_enh2
ggplot()+
  geom_sf(aes(fill=name), data = venn_region(venndata_enh2), show.legend = FALSE, alpha = 0.5) +
  geom_sf_text(aes(label = count), fontface = "bold", data = venn_region(venndata_enh2), size = 5,
nudge_y = 0.025) +
  geom_sf_text(aes(label = paste0(#"\n",
    fifelse(up != 0, paste0(up, " ", sprintf('\u2191')), paste0("")),
    fifelse(up != 0 & dn != 0, paste0("  "), paste0("")),
    fifelse(dn != 0, paste0(dn, " ", sprintf('\u2193')), paste0(""))
```

```
  )), data = venn_region(venndata_enh2), nudge_y = -0.005) +
  geom_sf_text(aes(label = paste0(fifelse(dif > 0 , paste0(dif," ", sprintf('\u2191'),
sprintf('\u2193'), "*"), paste0(""))),
                data = venn_region(venndata_enh2), nudge_y = 0.1) +
  geom_sf_label(aes(label = gsub("vs", "vs\n", gsub("_", " ", name))), data =
venn_setlabel(venndata_enh2), alpha = 0.5, label.size = 0, fontface = "bold", size = 5, nudge_y =
0.03) +
  scale_x_continuous(expand = expansion(mult = .3)) +
  scale_fill_manual(values = ordered_new_RdYlBu) +
  theme_void()
ggsave("enhancers_venn2.png", plot = last_plot(), device = "png", width = 9, height = 9, units =
"in")



#venndata_enh3
nrow(dt_enh[(siNONEtOH_vs_siNONdex_reg == 1 & siEP300EtOH_vs_siEP300dex_reg == 0 &
siNONdex_vs_siEP300dex_reg == 1 & siNONdex_vs_siEP300dex_up == 1 & siNONEtOH_vs_siNONdex_dn == 1),])
nrow(dt_enh[(siNONEtOH_vs_siNONdex_reg == 1 & siNONdex_vs_siEP300dex_reg == 1 &
siEP300EtOH_vs_siEP300dex_reg == 1 & siNONEtOH_vs_siNONdex_dn == 1 & siNONdex_vs_siEP300dex_up == 1
& siEP300EtOH_vs_siEP300dex_dn == 1),])

ggplot()+
  geom_sf(aes(fill=name), data = venn_region(venndata_enh3), show.legend = FALSE, alpha = 0.5) +
  geom_sf_text(aes(label = count), fontface = "bold", data = venn_region(venndata_enh3), size = 5,
nudge_y = 0.7) +
  geom_sf_text(aes(label = paste0(#"\n",
    fifelse(up != 0, paste0(up, " ", sprintf('\u2191')), paste0("")),
    fifelse(up != 0 & dn != 0, paste0("  "), paste0("")),
    fifelse(dn != 0, paste0(dn, " ", sprintf('\u2193')), paste0(""))
  )), data = venn_region(venndata_enh3), nudge_y = 0.1) +
  geom_sf_text(aes(label = paste0(fifelse(dif > 0 , paste0(dif," ", sprintf('\u2191'),
sprintf('\u2193')), paste0(""))),
                data = venn_region(venndata_enh3), nudge_y = -0.4) +
  geom_sf_label(aes(label = gsub("_", " ", name)), data = venn_setlabel(venndata_enh3), alpha = 0.5,
label.size = 0, fontface = "bold", size = 5, nudge_y = 0.05) +
  scale_x_continuous(expand = expansion(mult = .3)) +
  scale_fill_brewer(palette = "RdYlBu") +
  annotate(geom = "text", x = 0, y = 1.9, label = "*1", size = 3) +
  annotate(geom = "text", x = 2.5, y = 1.4, label = "*2", size = 3) +
  annotate(geom = "text", x = -2.3, y = -5.6, label = paste0("* 1) ", sprintf('\u2191'), " in siNONEtOH
vs siNONdex, ", sprintf('\u2193'), " in siNONdex vs siEP300dex", "\n\t\t\t\t
2) ", sprintf('\u2191'), " in siNONdex vs siEP300dex, ", sprintf('\u2193'), " in siNONEtOH vs siNONdex
and siEP300EtOH vs siEP300dex"), size = 3) +
  theme_void()
ggsave("enhancers_venn3.png", plot = last_plot(), device = "png", width = 9, height = 9, units =
"in")

#venndata_enh4
ggplot()+
  geom_sf(aes(fill=name), data = venn_region(venndata_enh4), show.legend = FALSE, alpha = 0.5) +
  geom_sf_text(aes(label = count), fontface = "bold", data = venn_region(venndata_enh4), size = 5,
nudge_y = 0.7) +
  geom_sf_text(aes(label = paste0(#"\n",
    fifelse(up != 0, paste0(up, " ", sprintf('\u2191')), paste0("")),
    fifelse(up != 0 & dn != 0, paste0("  "), paste0("")),
    fifelse(dn != 0, paste0(dn, " ", sprintf('\u2193')), paste0(""))
  )), data = venn_region(venndata_enh4), nudge_y = 0.1) +
```

```
  geom_sf_text(aes(label = paste0(fifelse(dif > 0 , paste0(dif," ", sprintf('\u2191'),
sprintf('\u2193'), "*"), paste0(""))),
                data = venn_region(venndata_enh4), nudge_y = -0.3) +
  geom_sf_label(aes(label = gsub("_", " ", name)), data = venn_setlabel(venndata_enh4), alpha = 0.5,
label.size = 0, fontface = "bold", size = 5, nudge_y = 0.05) +
  scale_x_continuous(expand = expansion(mult = .3)) +
  scale_fill_brewer(palette = "RdYlBu") +
  theme_void()
ggsave("enhancers_venn4.png", plot = last_plot(), device = "png", width = 8, height = 8, units =
"in")


#venndata_enh5
ggplot()+
  geom_sf(aes(fill=name), data = venn_region(venndata_enh5), show.legend = FALSE, alpha = 0.5) +
  geom_sf_text(aes(label = count), fontface = "bold", data = venn_region(venndata_enh5), size = 5,
nudge_y = 0.7) +
  geom_sf_text(aes(label = paste0(#"\n",
    fifelse(up != 0, paste0(up, " ", sprintf('\u2191')), paste0("")),
    fifelse(up != 0 & dn != 0, paste0("  "), paste0("")),
    fifelse(dn != 0, paste0(dn, " ", sprintf('\u2193')), paste0(""))
  )), data = venn_region(venndata_enh5), nudge_y = 0.1) +
  geom_sf_text(aes(label = paste0(fifelse(dif > 0 , paste0(dif," ", sprintf('\u2191'),
sprintf('\u2193'), "*"), paste0(""))),
                data = venn_region(venndata_enh5), nudge_y = -0.3) +
  geom_sf_label(aes(label = gsub("_", " ", name)), data = venn_setlabel(venndata_enh5), alpha = 0.5,
label.size = 0, fontface = "bold", size = 5, nudge_y = 0.05) +
  scale_x_continuous(expand = expansion(mult = .3)) +
  scale_fill_brewer(palette = "RdYlBu") +
  theme_void()
ggsave("enhancers_venn5.png", plot = last_plot(), device = "png", width = 8, height = 8, units =
"in")

#venndata_enh6
ggplot()+
  geom_sf(aes(fill=name), data = venn_region(venndata_enh6), show.legend = FALSE, alpha = 0.5) +
  geom_sf_text(aes(label = count), fontface = "bold", data = venn_region(venndata_enh6), size = 5,
nudge_y = 0.7) +
  geom_sf_text(aes(label = paste0(#"\n",
    fifelse(up != 0, paste0(up, " ", sprintf('\u2191')), paste0("")),
    fifelse(up != 0 & dn != 0, paste0("  "), paste0("")),
    fifelse(dn != 0, paste0(dn, " ", sprintf('\u2193')), paste0(""))
  )), data = venn_region(venndata_enh6), nudge_y = 0.1) +
  geom_sf_text(aes(label = paste0(fifelse(dif > 0 , paste0(dif," ", sprintf('\u2191'),
sprintf('\u2193'), "*"), paste0(""))),
                data = venn_region(venndata_enh6), nudge_y = -0.3) +
  geom_sf_label(aes(label = gsub("_", " ", name)), data = venn_setlabel(venndata_enh6), alpha = 0.5,
label.size = 0, fontface = "bold", size = 5, nudge_y = 0.05) +
  scale_x_continuous(expand = expansion(mult = .3)) +
  scale_fill_brewer(palette = "RdYlBu") +
  theme_void()
ggsave("enhancers_venn6.png", plot = last_plot(), device = "png", width = 8, height = 8, units =
"in")

#venndata_enh7
ggplot()+
  geom_sf(aes(fill=name), data = venn_region(venndata_enh7), show.legend = FALSE, alpha = 0.5) +
```

```
  geom_sf_text(aes(label = count), fontface = "bold", data = venn_region(venndata_enh7), size = 5,
nudge_y = 0.7) +
  geom_sf_text(aes(label = paste0(#"\n",
    fifelse(up != 0, paste0(up, " ", sprintf('\u2191')), paste0("")),
    fifelse(up != 0 & dn != 0, paste0("  "), paste0("")),
    fifelse(dn != 0, paste0(dn, " ", sprintf('\u2193')), paste0(""))
  )), data = venn_region(venndata_enh7), nudge_y = 0.1) +
  geom_sf_text(aes(label = paste0(fifelse(dif > 0 , paste0(dif," ", sprintf('\u2191'),
sprintf('\u2193'), "*"), paste0("")))),
                data = venn_region(venndata_enh7), nudge_y = -0.3) +
  geom_sf_label(aes(label = gsub("vs ", "vs\n", gsub("_", " ", name))), data =
venn_setlabel(venndata_enh7), alpha = 0.5, label.size = 0, fontface = "bold", size = 5, nudge_y =
0.7) +
  scale_x_continuous(expand = expansion(mult = .3)) +
  scale_y_continuous(expand = expansion(mult = .3)) +
  scale_fill_brewer(palette = "RdYlBu") +
  theme_void()
ggsave("enhancers_venn7.png", plot = last_plot(), device = "png", width = 7, height = 5, units =
"in")

#venndata_enh8
ggplot()+
  geom_sf(aes(fill=name), data = venn_region(venndata_enh8), show.legend = FALSE, alpha = 0.5) +
  geom_sf_text(aes(label = count), fontface = "bold", data = venn_region(venndata_enh8), size = 5,
nudge_y = 0.7) +
  geom_sf_text(aes(label = paste0(#"\n",
    fifelse(up != 0, paste0(up, " ", sprintf('\u2191')), paste0("")),
    fifelse(up != 0 & dn != 0, paste0("  "), paste0("")),
    fifelse(dn != 0, paste0(dn, " ", sprintf('\u2193')), paste0(""))
  )), data = venn_region(venndata_enh8), nudge_y = 0.1) +
  geom_sf_text(aes(label = paste0(fifelse(dif > 0 , paste0(dif," ", sprintf('\u2191'),
sprintf('\u2193'), "*"), paste0("")))),
                data = venn_region(venndata_enh8), nudge_y = -0.3) +
  geom_sf_label(aes(label = gsub("vs ", "vs\n", gsub("_", " ", name))), data =
venn_setlabel(venndata_enh8), alpha = 0.5, label.size = 0, fontface = "bold", size = 5, nudge_y =
0.7) +
  scale_x_continuous(expand = expansion(mult = .3)) +
  scale_y_continuous(expand = expansion(mult = .3)) +
  scale_fill_brewer(palette = "RdYlBu") +
  theme_void()
ggsave("enhancers_venn8.png", plot = last_plot(), device = "png", width = 7, height = 5, units =
"in")


#venndata_enh9
ggplot()+
  geom_sf(aes(fill=name), data = venn_region(venndata_enh9), show.legend = FALSE, alpha = 0.5) +
  geom_sf_text(aes(label = count), fontface = "bold", data = venn_region(venndata_enh9), size = 5,
nudge_y = 0.5) +
  geom_sf_text(aes(label = paste0(#"\n",
    fifelse(up != 0, paste0(up, " ", sprintf('\u2191')), paste0("")),
    fifelse(up != 0 & dn != 0, paste0("  "), paste0("")),
    fifelse(dn != 0, paste0(dn, " ", sprintf('\u2193')), paste0(""))
  )), data = venn_region(venndata_enh9), nudge_y = -0.01) +
  geom_sf_text(aes(label = paste0(fifelse(dif > 0 , paste0(dif," ", sprintf('\u2191'),
sprintf('\u2193'), " *"), paste0("")))),
                data = venn_region(venndata_enh9), nudge_y = -0.1) +
```

```
  geom_sf_label(aes(label = gsub("_", " ", name)), data = venn_setlabel(venndata_enh9), alpha = 0.5,
label.size = 0, fontface = "bold", size = 5, nudge_y = 0.03) +
  scale_x_continuous(expand = expansion(mult = .3)) +
  scale_fill_manual(values = adj_RdYlBu[c(1, 3, 5, 7, 9, 11, 13)]) +
  theme_void()
ggsave("enhancers_venn9.png", plot = last_plot(), device = "png", width = 8, height = 8, units =
"in")

#venndata_enh10
ggplot()+
  geom_sf(aes(fill=name), data = venn_region(venndata_enh10), show.legend = FALSE, alpha = 0.5) +
  geom_sf_text(aes(label = count), fontface = "bold", data = venn_region(venndata_enh10), size = 5,
nudge_y = 0.5) +
  geom_sf_text(aes(label = paste0(#"\n",
    fifelse(up != 0, paste0(up, " ", sprintf('\u2191')), paste0("")),
    fifelse(up != 0 & dn != 0, paste0("  "), paste0("")),
    fifelse(dn != 0, paste0(dn, " ", sprintf('\u2193')), paste0(""))
  )), data = venn_region(venndata_enh10), nudge_y = -0.01) +
  geom_sf_text(aes(label  =  paste0(fifelse(dif  >  0  ,  paste0(dif,"  ",  sprintf('\u2191'),
sprintf('\u2193'), " *"), paste0(""))),
                data = venn_region(venndata_enh10), nudge_y = -0.1) +
  geom_sf_label(aes(label = gsub("_", " ", name)), data = venn_setlabel(venndata_enh10), alpha = 0.5,
label.size = 0, fontface = "bold", size = 5, nudge_y = 0.03) +
  scale_x_continuous(expand = expansion(mult = .3)) +
  scale_fill_manual(values = adj_RdYlBu[c(1, 3, 5, 7, 9, 11, 13)]) +
  theme_void()
ggsave("enhancers_venn10.png", plot = last_plot(), device = "png", width = 8, height = 8, units =
"in")


###################
## Network plots ##
###################

#######
##GENES

#take comparisons to be plotted
toNetwork <- grep("vs", grep("_up|_dn", colnames(dt), value = TRUE), value = TRUE)

#initiate data.frame for loop
df <- data.frame()

for(i in 1:length(toNetwork)){
  #collect genes that are DE in the comparison at hand
  genes <- dt[get(toNetwork[i]) == 1, Gene]
  #add the comprison name as many as there are DE genes in it
  treatment <- rep(gsub("_", " ", toNetwork[i]) , length(genes))
  #bind genes and treatment as columns and add rows to the end of the data.frame
  df <- rbind(df, cbind(treatment, genes))
}

#for determining node attributes for the network
#gather all genes and comparisons to a vector
#and turn it to a data.frame by adding columns with information on if the row is a gene or comparison
label <- append(unique(df$genes), unique(df$treatment))
node_col <- grepl("vs", label, fixed = TRUE)
node_size <- grepl("vs", label, fixed = TRUE)
```

```
node_label <- grepl("vs", label, fixed = TRUE)
nodes <- as.data.frame(cbind(label, node_col, node_size, node_label))
rownames(nodes) <- label

#replace genes (FALSE) and comparisons (TRUE) with the desired information
#mainly, label all genes "gene" (for colouring)
#set sizes to 1 or 5 (for point sizes in network)
#and replace all gene names with NAs (to not plot all gene names)
nodes$node_col <- replace(nodes$node_col, nodes$node_col == FALSE, "gene")
nodes[grep("up|dn", rownames(nodes), value = TRUE), "node_col"] <- nodes[grep("up|dn",
rownames(nodes), value = TRUE), "label"]
nodes$node_size <- replace(nodes$node_size, nodes$node_size == FALSE, as.integer(1))
nodes$node_size <- replace(nodes$node_size, nodes$node_size == TRUE, as.integer(5))
nodes$node_label <- replace(nodes$node_label, nodes$node_label == FALSE, NA)
nodes[grep("up|dn", rownames(nodes), value = TRUE), "node_label"] <- nodes[grep("up|dn",
rownames(nodes), value = TRUE), "label"]


#create the network
ntwrk <- network(df, directed = FALSE, multiple = TRUE)

#define the node attributes i.e. what groups you want the data to be plotted with
ntwrk %v% "category" <- nodes[ network.vertex.names(ntwrk), "node_col"]
ntwrk %v% "group" <- nodes[ network.vertex.names(ntwrk), "node_size"]
ntwrk %v% "label" <- nodes[ network.vertex.names(ntwrk), "node_label"]

#create the network plot data.frame that is used for plotting with ggnetwork()
#set the seed to always receive the same results
set.seed(3) #1 close
ntwrkplot <- ggnetwork(ntwrk, layout = "kamadakawai")

#so that labels are not plotted for every single node, replace all duplicated ones with NA
ntwrkplot$label <- ave(ntwrkplot$label, ntwrkplot$vertex.names, FUN = function(a) replace(a,
duplicated(a), NA_integer_))
#change group variable into integer so it can directly be plotted as the size
ntwrkplot$group <- as.integer(ntwrkplot$group)
#to have the comparisons plotted in the legend in the order desired, change the column into a factor
and define levels
ntwrk_levels <- c("siNONEtOH vs siNONdex dn", "siNONEtOH vs siNONdex up",
                  "siNONEtOH vs siNONKLA up",
                  "siNONEtOH vs siNONdexKLA dn", "siNONEtOH vs siNONdexKLA up",
                  "siNONdex vs siNONdexKLA up",
                  "siNONKLA vs siNONdexKLA dn", "siNONKLA vs siNONdexKLA up",
                  "siEP300EtOH vs siEP300dex dn","siEP300EtOH vs siEP300dex up",
                  "siEP300EtOH vs siEP300dexKLA dn", "siEP300EtOH vs siEP300dexKLA up",
                  "siEP300KLA vs siEP300dexKLA dn", "siEP300KLA vs siEP300dexKLA up",
                  "siNONEtOH vs siEP300EtOH dn", "siNONEtOH vs siEP300EtOH up",
                  "siNONdex vs siEP300dex dn", "siNONdex vs siEP300dex up",
                  "siNONKLA vs siEP300KLA dn", "siNONKLA vs siEP300KLA up",
                  "siNONdexKLA vs siEP300dexKLA dn", "siNONdexKLA vs siEP300dexKLA up", "gene")
ntwrkplot$category <- factor(ntwrkplot$category, levels = ntwrk_levels)

#the colours are mapped to the items in alphabetical order, not in the order of the levels set for
the factor
#manually select colours in the correct order
ntwrk_clr                                                                                    <-
c("#808080","#9914db","#bc00dd","#f7aef8","#ff8faa","#f20089","#c77dff","#4361ee","#0031f8","#f77f0
```

```
0","#023e8a","#4895ef","#54daff","#077bc9","#fc4c27","#f79d65","#720026","#ff5400","#ffcd61","#6684
ff","#a2d2ff","#fdb833","#ffe94e")
#create a table for colours to be matched to comparisons in the legend, which follows the factor
ordering
#colours are reorganized so that they will still match to what is plotted
#to have the gene not show up in the legend but be plotted as grey dots, make its colour white
ntwrk_cols  <-  as.data.frame(cbind(ntwrk_levels,  c("#fc4c27",  "#f79d65",  "#ffcd61",  "#720026",
"#ff5400", "#f77f00",
                                                    "#fdb833",   "#ffe94e",   "#9914db",   "#bc00dd",
"#f7aef8", "#ff8faa",
                                                    "#f20089",   "#c77dff",   "#54daff",   "#077bc9",
"#4361ee", "#0031f8",
                                                    "#6684ff",   "#a2d2ff",   "#023e8a",   "#4895ef",
"#FFFFFF")))
#change the gene to be an empty character string to not have it labeled in the legend
ntwrk_cols[ntwrk_cols$ntwrk_levels == "gene", 1] <- ""



#change the placing of siNONKLA vs siEP300KLA up:

#the original coordinates are 0.5620291 and 0, nudge them up and to the right
ntwrkplot[ntwrkplot$category == "siNONKLA vs siEP300KLA up","x"] <- 0.7220291
ntwrkplot[ntwrkplot$category == "siNONKLA vs siEP300KLA up","y"] <- 0.050696969
#the xend and yend columns also need to be changed to reflect the new values, but only where they
equal the original coordinates
#the second value is the one we want to replace
ntwrkplot[ntwrkplot$category == "siNONKLA vs siEP300KLA up", "xend"][2] <- 0.7220291
#same for y
ntwrkplot[ntwrkplot$category == "siNONKLA vs siEP300KLA up", "yend"][2] <- 0.050696969
#confirm
ntwrkplot[ntwrkplot$category == "siNONKLA vs siEP300KLA up",]
#looks good

#set the seed for labels to always be in same positions
set.seed(1)
ggplot(data = ntwrkplot,
       aes(x, y, xend = xend, yend = yend, color = category, size = group, label = label)) +
  geom_edges(size = 1, alpha = 0.4, show.legend = FALSE) + # draw edge layer
  geom_nodes() + # draw node layer
  geom_text_repel(seed = 1, max.overlaps = Inf, force = 15, force_pull = 0.001,
                  size = 3.5, colour = "black", fontface = "bold", point.size = 1,
                  box.padding = 0.75, point.padding = 1, segment.size = 0.80,
                  min.segment.length = 0,  segment.linetype = 1, segment.color = "black",
                  na.rm = TRUE) + # draw node label layer
  scale_size(range = c(1, 7)) +
  scale_color_manual(values = ntwrk_clr, labels = ntwrk_cols[,1]) +
  labs(color = "") +
  #to hide "gene" from the legend, override the aesthetics and and provide a different list of colours
  guides(size = "none", colour = guide_legend(override.aes = list(size=5, colour = ntwrk_cols[,2]),
ncol = 1)) +
  theme_blank() +
  theme(legend.position = "right", legend.key.size = unit(0.1, "cm"), legend.text = element_text(size
= 7))
ggsave("genes_network.pdf", plot = last_plot(), device = "pdf", width = 13, height = 9, units =  "in")
ggsave("genes_network.tif", plot = last_plot(), device = "tiff", width = 13, height = 9, units =
"in")
ggsave("genes_network.png", plot = last_plot(), device = "png", width = 13, height = 9, units =  "in")
```

```
2191
2192
2193
2194  ###########
2195  ##ENHANCERS
2196
2197  toNetwork_enh <- grep("vs", grep("_up|_dn", colnames(dt_enh), value = TRUE), value = TRUE)
2198
2199  df_enh <- data.frame()
2200
2201  for(i in 1:length(toNetwork_enh)){
2202    genes <- dt_enh[get(toNetwork_enh[i]) == 1, Gene]
2203    treatment <- rep(gsub("_", " ", toNetwork_enh[i]) , length(genes))
2204    df_enh <- rbind(df_enh, cbind(treatment, genes))
2205  }
2206
2207  label_enh <- append(unique(df_enh$genes), unique(df_enh$treatment))
2208  node_col_enh <- grepl("vs", label_enh, fixed = TRUE)
2209  node_size_enh <- grepl("vs", label_enh, fixed = TRUE)
2210  node_label_enh <- grepl("vs", label_enh, fixed = TRUE)
2211  nodes_enh <- as.data.frame(cbind(label_enh, node_col_enh, node_size_enh, node_label_enh))
2212  colnames(nodes_enh) <- c("label","node_col", "node_size", "node_label")
2213  rownames(nodes_enh) <- label_enh
2214
2215  nodes_enh$node_col <- replace(nodes_enh$node_col, nodes_enh$node_col == FALSE, "gene")
2216  nodes_enh[grep("up|dn", rownames(nodes_enh), value = TRUE), "node_col"] <- nodes_enh[grep("up|dn",
2217  rownames(nodes_enh), value = TRUE), "label"]
2218  nodes_enh$node_size <- replace(nodes_enh$node_size, nodes_enh$node_size == FALSE, 1)
2219  nodes_enh$node_size <- replace(nodes_enh$node_size, nodes_enh$node_size == TRUE, 5)
2220  nodes_enh$node_label <- replace(nodes_enh$node_label, nodes_enh$node_label == FALSE, NA)
2221  nodes_enh[grep("up|dn", rownames(nodes_enh), value = TRUE), "node_label"] <- nodes_enh[grep("up|dn",
2222  rownames(nodes_enh), value = TRUE), "label"]
2223
2224  ntwrk_enh <- network(df_enh, directed = FALSE)
2225
2226  ntwrk_enh %v% "category" <- nodes_enh[ network.vertex.names(ntwrk_enh), "node_col"]
2227  ntwrk_enh %v% "group" <- nodes_enh[ network.vertex.names(ntwrk_enh), "node_size"]
2228  ntwrk_enh %v% "label" <- nodes_enh[ network.vertex.names(ntwrk_enh), "node_label"]
2229
2230  ntwrk_levels_enh <- c("siNONEtOH vs siNONdex dn", "siNONEtOH vs siNONdex up",
2231                        "siNONEtOH vs siNONdexKLA dn", "siNONEtOH vs siNONdexKLA up",
2232                        "siNONKLA vs siNONdexKLA dn", "siNONKLA vs siNONdexKLA up",
2233                        "siEP300EtOH vs siEP300dex dn","siEP300EtOH vs siEP300dex up",
2234                        "siEP300EtOH vs siEP300dexKLA dn", "siEP300EtOH vs siEP300dexKLA up",
2235                        "siEP300KLA vs siEP300dexKLA dn", "siEP300KLA vs siEP300dexKLA up",
2236                        "siNONEtOH vs siEP300EtOH dn", "siNONEtOH vs siEP300EtOH up",
2237                        "siNONdex vs siEP300dex dn", "siNONdex vs siEP300dex up",
2238                        "siNONKLA vs siEP300KLA dn", "siNONKLA vs siEP300KLA up",
2239                        "siNONdexKLA vs siEP300dexKLA dn", "siNONdexKLA vs siEP300dexKLA up", "gene")
2240
2241  ntwrk_clr_enh                                                                                    <-
2242  c("#808080","#9914db","#bc00dd","#f7aef8","#ff8faa","#f20089","#c77dff","#4361ee","#0031f8","#023e8
2243  a","#4895ef","#54daff","#077bc9","#fc4c27","#f79d65","#720026","#ff5400","#6684ff","#a2d2ff","#fdb8
2244  33","#ffe94e")
2245  ntwrk_cols_enh <- as.data.frame(cbind(ntwrk_levels_enh, c("#fc4c27", "#f79d65", "#720026", "#ff5400",
2246                                                            "#fdb833",    "#ffe94e",    "#9914db",    "#bc00dd",
2247  "#f7aef8", "#ff8faa",
```

```
                                                    "#f20089",   "#c77dff",   "#54daff",   "#077bc9",
"#4361ee", "#0031f8",
                                                    "#6684ff",   "#a2d2ff",   "#023e8a",   "#4895ef",
"#FFFFFF")))
ntwrk_cols_enh[ntwrk_cols_enh$ntwrk_levels == "gene", 1] <- ""

set.seed(5)
ntwrkplot_enh <- ggnetwork(ntwrk_enh, layout = "kamadakawai")

ntwrkplot_enh$label  <-  ave(ntwrkplot_enh$label,  ntwrkplot_enh$vertex.names,  FUN  =  function(a)
replace(a, duplicated(a), NA_integer_))
ntwrkplot_enh$group <- as.integer(ntwrkplot_enh$group)
ntwrkplot_enh$category <- factor(ntwrkplot_enh$category, levels = ntwrk_levels_enh)


set.seed(5)
ggplot(data = ntwrkplot_enh,
       aes(x, y, xend = xend, yend = yend, color = category, size = group, label = label)) +
  geom_edges(size = 1, alpha = 0.3, show.legend = FALSE) + # draw edge layer
  geom_nodes() + # draw node layer
  scale_size(range = c(1, 7)) +
  geom_text_repel(seed = 1, size = 3.5, colour = "black", fontface = "bold", point.size = 1,
                  box.padding = 0.5, point.padding = 1, segment.size = 0.8,
                  min.segment.length = 0,  segment.linetype = 1, force = 30, force_pull = 0.001,
                  segment.color = "black", na.rm = TRUE) + # draw node label layer
  scale_color_manual(values = ntwrk_clr_enh, labels = ntwrk_cols_enh[,1]) +
  labs(color = "") +
  guides(size   =   "none",   colour   =   guide_legend(override.aes   =   list(size=5,   colour   =
ntwrk_cols_enh[,2]), ncol = 1)) +
  theme_blank() +
  theme(legend.position = "right", legend.key.size = unit(0.1, "cm"), legend.text = element_text(size
= 7))
ggsave("enhancers_network.pdf", plot = last_plot(), device = "pdf", width = 11, height = 9, units =
"in")
ggsave("enhancers_network.tif", plot = last_plot(), device = "tiff", width = 11, height = 9, units =
"in")
ggsave("enhancers_network.png", plot = last_plot(), device = "png", width = 13, height = 9, units =
"in")


####################
## Motif analysis ##
####################

#write BED files for analysis to use as input for HOMER
#columns 2-4 will form the first three and necessary columns of a BED file
#important to not print row or column names or quotes so that BED file is in correct format

toMotifAnalysis <- grep("vs", grep("_up|_dn", colnames(dt_enh), value = TRUE), value =TRUE)

for (i in 1:length(toMotifAnalysis)){
  #if there are differentially transcribed enhancers, write them into a table
  if(nrow(dt[get(toMotifAnalysis[i]) == 1] > 0)){
    write.table(dt[get(toMotifAnalysis[i])    ==    1,    2:4],    file    =    paste0("de_enhancers_",
toMotifAnalysis[i], ".bed"),
                sep = "\t", quote = FALSE, row.names = FALSE, col.names = FALSE)
  }
}
```

```
#also write a file of all the enhancers to use as background for the motif analysis
write.table(dt_enh[, 2:4], file = "background_enhancers.bed", sep ="\t", row.names = FALSE, quote =
FALSE, col.names = FALSE)


#read in analysis results from HOMER

#list all files and save to a list object using lapply and clean up the names
#motif result files are placed in the folder "motifs"
filenames <- list.files("motifs", pattern = "knownResults*")

motifs <- lapply(paste0("motifs/", filenames), read.csv, header = TRUE, sep = "\t")
names(motifs) <- filenames
names(motifs) <- gsub("knownResults_", "", names(motifs))
names(motifs) <- gsub("_enhancerbg.txt", "", names(motifs))

#fix column names and remove % symbols and duplicated rows and calculate enrichment ratio
#enrichment ratio is calculated as the % of target sequences with the motif divided by the % of
background sequences with the motif
#for enrichment ratio calculation, the columns with the % values have to be changed from string to
double
for (i in 1:length(motifs)){
  names(motifs[[i]]) <- gsub("[.]", "_", names(motifs[[i]]))
  colnames(motifs[[i]])[7] <- "Perc_of_Target_Sequences_with_Motif"
  colnames(motifs[[i]])[9] <- "Perc_of_Background_Sequences_with_Motif"
  motifs[[i]]$Perc_of_Target_Sequences_with_Motif         <-         as.double(gsub("%",         "",
as.character(motifs[[i]]$Perc_of_Target_Sequences_with_Motif)))
  motifs[[i]]$Perc_of_Background_Sequences_with_Motif        <-        as.double(gsub("%",        "",
as.character(motifs[[i]]$Perc_of_Background_Sequences_with_Motif)))
  motifs[[i]] <- motifs[[i]][!duplicated(motifs[[i]]),]
  motifs[[i]]$Enrichment_ratio       <-       motifs[[i]]$Perc_of_Target_Sequences_with_Motif       /
motifs[[i]]$Perc_of_Background_Sequences_with_Motif
}

#merge into one data.frame
#initiate by merging two of the first ones
motif_df <- merge.data.frame(motifs[[1]], motifs[[2]], by = c("Motif_Name", "Consensus"),
                             suffixes     =     c(paste0("_",     names(motifs)[1]),     paste0("_",
names(motifs)[2])))
#loop the rest
for (i in 3:length(motifs)){
  motif_df <- merge.data.frame(motif_df, motifs[[i]], by = c("Motif_Name", "Consensus"),
                               suffixes = c(paste0(""), paste0("_", names(motifs)[i])))
}
#this leaves the third merged data.frame (first data.frame of the loop) without column suffixes, add
them manually
colnames(motif_df)[c(19,20,21,23,25,26)]  <-  paste0(colnames(motif_df)[c(19,20,21,23,25,26)],  "_",
names(motifs)[3])

#for plotting with ggplot2 gather all into one long format data.frame
motif_all_df <- data.frame()

for(i in 1:length(motifs)){
  motifs[[i]]$comparison <- rep(gsub("_", " ", names(motifs)[i]), nrow(motifs[[i]]))
  #leave out the columns with absolute counts of enhancers
  #because these all have different column names and thus cannot be bound with rbind
  motif_all_df <- rbind(motif_all_df, motifs[[i]][, c(1:5,7,9:11)])
```

```
}


#to reduce redundancy, merge similar motifs together

#create motifs using the create_motif command from universalmotif package
#from the consensus sequences of motifs from the HOMER data
motifs_for_merging <- list()

for (i in 1:length(motif_df$Consensus)){
  motifs_for_merging[[i]] <- create_motif(motif_df$Consensus[i], name = motif_df$Motif_Name[i])
}

#then merge the similar motifs together
#by setting min.overlap to 20, which is very often longer than the motif
#no overhangs should be left
#so that motifs are not aligned only at the edges
merged_motifs <- merge_similar(motifs_for_merging, threshold = 0.80, min.overlap = 20)

#all motifs can now be viewed
view_motifs(merged_motifs)

#add smallest q-value column to motif_df
#for this column, always add the lowest q-value of any treatment on that row to the cell
for (i in 1:nrow(motif_df)){
  motif_df$Smallest_qval[i] <- min(motif_df[i, grep("q_value", colnames(motif_df), value = TRUE)])
}

#of the merged motifs, always keep the one that has the smallest q-value in any treatment
#this is done using nested for-loops
temp <- data.frame()
motifs_merged_keep <- data.frame(matrix(ncol = 2))

for (i in 1:length(merged_motifs)){
  for (j in 1:nrow(motif_df)){
    #if a motif name in motif_df is found in the merged_motifs, save that row to temp
    #the name spot of an universalmotif class object can be accessed with [""]
    if(grepl(motif_df$Motif_Name[j], merged_motifs[[i]]["name"], fixed = TRUE) == TRUE){
      temp <- rbind(temp, motif_df[j,])
    }
  }
  #order the table according to the smallest q-value column
  temp <- temp[order(temp$Smallest_qval),]
  #save the top one into motifs_merged_keep
  #into the 1st column place the name of the representative motif
  #in the second column place the name that shows the merged motifs
  motifs_merged_keep[i,1] <- temp$Motif_Name[1]
  motifs_merged_keep[i,2] <- merged_motifs[[i]]["name"]
  #reset
  temp <- data.frame()
}

#filter motif_df to include only the top motifs saved in motifs_merged_keep
motif_df_filtered <- motif_df[motif_df$Motif_Name %in% motifs_merged_keep[,1],]
#order both alphabetically and save to a new column information about which motifs have been merged
motif_df_filtered <- motif_df_filtered[order(motif_df_filtered$Motif_Name),]
motifs_merged_keep <- motifs_merged_keep[order(motifs_merged_keep[,1]),]
motif_df_filtered$Merged_motif <- motifs_merged_keep[,2]
```

```
#next keep only the top 5 motifs from each treatment
#initiate object for loop
top5 <- vector()

for (i in 1:length(motifs)){
  #order the data.frame according to the q-value column of one comparison at a time
  motif_df_filtered  <-  motif_df_filtered[order(motif_df_filtered[,  paste0("q_value__Benjamini__",
names(motifs)[i])]),]
  #and save the top 5 motifs
  top5 <- append(top5, motif_df_filtered$Motif_Name[1:5])
}
#see the unique motifs
unique(top5)

#keep only the selected motifs
motif_df_filtered_top5 <- motif_df_filtered[motif_df_filtered$Motif_Name %in% top5,]
#also in the long format df
motif_all_df_filtered_top5 <- motif_all_df[motif_all_df$Motif_Name %in% top5,]
#filter to keep only statistically significant ones
motif_all_df_filtered_top5_sig                                                                    <-
motif_all_df_filtered_top5[motif_all_df_filtered_top5$q_value__Benjamini_ < 0.05,]
#clean up motif names
motif_all_df_filtered_top5_sig$Motif_Name    <-    gsub("[(]",    "    (",    gsub("[)].*",    ")",
motif_all_df_filtered_top5_sig$Motif_Name))
#add a shorter version of comparison name
#and include information about direction
#these are used as parameters for plotting the enrichment ratios
motif_all_df_filtered_top5_sig$Treatment    <-    gsub("    dn",    "",    gsub("    up",    "",
motif_all_df_filtered_top5_sig$comparison))
motif_all_df_filtered_top5_sig$Direction                                                          <-
get_nth_of_list(motif_all_df_filtered_top5_sig$comparison, sep = " ", 4)

#drop columns with few significant motifs

toFilter <- grep("q_value", colnames(motif_df_filtered_top5), value = TRUE)

cols_to_keep <- vector()
#use j as an assisting index for cols_to_keep
#because using i would leave empty spaces in the vector when columns are excluded
j = 1
for(i in 1:length(toFilter)){
  if(length(which(motif_df_filtered_top5[, toFilter[i]] < 0.05)) > 1){
    cols_to_keep[j] <- gsub("q_value__Benjamini__", "", toFilter[i])
    j <- j + 1
  }
}

#use paste to get all of the columns to keep listed
motif_df_filtered_top5      <-      motif_df_filtered_top5[,      c("Motif_Name",      "Consensus",
grep(paste(cols_to_keep,   collapse="|"),   colnames(motif_df_filtered_top5),   value   =   TRUE),
"Smallest_qval", "Merged_motif")]

#review motifs by eye and manually merge similar ones not merged already
#for this you need to already initiate the heatmap!!
#because the roworderof the heatmap is needed to order the motifs
```

```
#so first, the heatmap
#just briefly here, it will be commented more closely later

#matrix for heatmap plotting

#save the q-value columns for each comparison and clean up the row and column names
motif_filtered_top5_matrix_qval <- as.matrix(motif_df_filtered_top5[, c(grep("q_value",
colnames(motif_df_filtered_top5)))])
rownames(motif_filtered_top5_matrix_qval) <- gsub("[(]", " (", gsub("[)].*", ")",
motif_df_filtered_top5$Motif_Name))
colnames(motif_filtered_top5_matrix_qval) <- gsub("_", " ", gsub(".*__", "",
colnames(motif_filtered_top5_matrix_qval)))

#change to neg log10 q-values
motif_filtered_top5_matrix_neg_log_qval <- -log10(motif_filtered_top5_matrix_qval)
#replace all Inf values with the max value found in the matrix
motif_filtered_top5_matrix_neg_log_qval[motif_filtered_top5_matrix_neg_log_qval == Inf] <-
max(motif_filtered_top5_matrix_neg_log_qval[motif_filtered_top5_matrix_neg_log_qval != Inf])

#make the colour gradient from min to max
col_fun_motif = colorRamp2(c(min(motif_filtered_top5_matrix_neg_log_qval),
max(motif_filtered_top5_matrix_neg_log_qval)), c("#FEEFA7", "#C82226"))

#make annotations
ratio_annot <- rowAnnotation("Enrichment ratio" = anno_points(seq(from = 1, to = 4.17, length.out =
nrow(motif_filtered_top5_matrix_neg_log_qval)), gp = gpar(col ="white"), axis_param = list(labels_rot
= 0)), border = TRUE, width = unit(9, "cm"), annotation_name_side = "top", annotation_name_rot = 0)
column_annot <- HeatmapAnnotation(Direction =
get_nth_of_list(colnames(motif_filtered_top5_matrix_neg_log_qval), sep = " ", 4), col =
list(Direction = c("up" = "tomato1", "dn" = "dodgerblue1")), annotation_name_side = "left")
#annotation_name_side = "left"
siRNA_annot <- HeatmapAnnotation(Comparison = gsub(" up", "", gsub(" dn", "", gsub("KLA", "",
gsub("dex", "", gsub("EtOH", "", colnames(motif_filtered_top5_matrix_neg_log_qval))))))),
col = list(Comparison = c("siEP300 vs siEP300" = "darkorchid4",
"siNON vs siEP300" = "lightskyblue", "siNON vs siNON" = "darkorange1")), annotation_name_side =
"left") #

motif_filtered_top5_matrix_neg_log_qval <-
motif_filtered_top5_matrix_neg_log_qval[order(rownames(motif_filtered_top5_matrix_neg_log_qval)),]

#for cleaning up column names
toSub <- c("up", "dn", "siNON", "siEP300")

#save heatmap to an object instead of plotting in order to get the roworder of motifs
hmap <- Heatmap(motif_filtered_top5_matrix_neg_log_qval,
                border_gp = gpar(col = "black", lwd = 1.5),
                heatmap_legend_param = list(title = expression(paste("-log"[10]," ", "FDR", sep =
""")), direction = "horizontal"),
                col = col_fun_motif,
                cluster_columns = TRUE,
                show_row_dend = FALSE,
                show_column_dend = FALSE,
                show_row_names = TRUE,
                row_names_side = "left",
                column_names_side = "top",
                column_names_rot = 55,
                column_labels = gsub(paste(toSub, collapse = "|"), "",
colnames(motif_filtered_top5_matrix_neg_log_qval)),
```

```
                  row_names_gp = gpar(fontsize = 10),
                  column_names_gp = gpar(fontsize = 10.5, fontface = "bold"),
                  right_annotation = ratio_annot,
                  top_annotation = c(siRNA_annot, column_annot)
)
#save it
roword <- row_order(hmap)

#and now the motif visualization

#merged motif consensus sequences
motifs_plotting <- list()
j=1
for (i in 1:length(merged_motifs)){
  #take those in merged_motifs that are found in motif_df_filtered_top5
  if(merged_motifs[[i]]["name"] %in% motif_df_filtered_top5$Merged_motif == TRUE){
    motifs_plotting[[j]] <- merged_motifs[[i]]
    #get index with which()
    ind <- which((motif_df_filtered$Merged_motif == merged_motifs[[i]]["name"]) == TRUE)
    #and use it to name the motif
    #which places the merged motif name with the representative motif name
    motifs_plotting[[j]]["name"] <- motif_df_filtered$Motif_Name[ind]
    j <- j + 1
  }

}

#get the positions and names
name <- vector()
pos <- vector()

for (i in 1:length(motifs_plotting)){
  pos[i] <- i
  name[i] <- motifs_plotting[[i]]["name"]
}
#and use them to order the motifs first before ordering with the row order to ensure correct order
namepos <- cbind(name, pos)
namepos <- namepos[order(namepos[,1]),]
help <- as.integer(namepos[,2])

motifs_plotting <- motifs_plotting[help]
motifs_plotting <- motifs_plotting[roword]

#and view them
view_motifs(motifs_plotting, min.overlap = 20, show.positions = FALSE, names.pos = "right", tryRC =
FALSE) +
  theme_void()

#original motif consensus sequences
motifs_merged_plotting <- list()
j=1
for (i in 1:length(motifs_for_merging)){
  if(motifs_for_merging[[i]]["name"] %in% motif_df_filtered_top5$Motif_Name == TRUE){
    motifs_merged_plotting[[j]] <- motifs_for_merging[[i]]
    j <- j + 1
  }
}

```

```
#order it the same as on the heatmap
motifs_merged_plotting <- motifs_merged_plotting[roword]
#and view
view_motifs(motifs_merged_plotting, min.overlap = 20, show.positions = FALSE, names.pos = "right",
tryRC = FALSE) +
  theme_void()


#based on this merge the following: TCF4 and NeuroG2, FOXK1 and Foxo3, Asd1 and Ap4, and Smad2 and
Smad3
#keep the one with the lowest q-value

#save them to a list
final_motif_filter <- list(c("TCF4", "NeuroG2"), c("FOXK1", "Foxo3"), c("Ascl1", "Ap4"), c("Smad2",
"Smad3"))

for (i in 1:length(final_motif_filter)){
  #save information about the q-values for both motifs to objects
  qval1           <-           motif_df_filtered_top5[motif_df_filtered_top5$Motif_Name           ==
grep(final_motif_filter[[i]][1], motif_df_filtered_top5$Motif_Name, value = TRUE), "Smallest_qval"]
  qval2           <-           motif_df_filtered_top5[motif_df_filtered_top5$Motif_Name           ==
grep(final_motif_filter[[i]][2], motif_df_filtered_top5$Motif_Name, value = TRUE), "Smallest_qval"]
  #if smallest q-value is the same, count the sum of all q-values and take the one with the smaller
sum
  if(qval1 == qval2){
    sum1         <-         sum(motif_df_filtered_top5[motif_df_filtered_top5$Motif_Name         ==
grep(final_motif_filter[[i]][1], motif_df_filtered_top5$Motif_Name, value = TRUE), grep("q_value",
colnames(motif_df_filtered_top5), value = TRUE)])
    sum2         <-         sum(motif_df_filtered_top5[motif_df_filtered_top5$Motif_Name         ==
grep(final_motif_filter[[i]][2], motif_df_filtered_top5$Motif_Name, value = TRUE), grep("q_value",
colnames(motif_df_filtered_top5), value = TRUE)])
    if(sum1<sum2){
      motif_df_filtered_top5    <-    motif_df_filtered_top5[motif_df_filtered_top5$Motif_Name    !=
grep(final_motif_filter[[i]][2], motif_df_filtered_top5$Motif_Name, value = TRUE),]
    } else if(sum1>sum2){
      motif_df_filtered_top5    <-    motif_df_filtered_top5[motif_df_filtered_top5$Motif_Name    !=
grep(final_motif_filter[[i]][1], motif_df_filtered_top5$Motif_Name, value = TRUE),]
    }
    #otherwise take the one with the lowest q-value
  } else if(qval1 < qval2){
    motif_df_filtered_top5     <-     motif_df_filtered_top5[motif_df_filtered_top5$Motif_Name     !=
grep(final_motif_filter[[i]][2], motif_df_filtered_top5$Motif_Name, value = TRUE),]
  } else if(qval1 > qval2){
    motif_df_filtered_top5     <-     motif_df_filtered_top5[motif_df_filtered_top5$Motif_Name     !=
grep(final_motif_filter[[i]][1], motif_df_filtered_top5$Motif_Name, value = TRUE),]
  }
}

#now draw the heatmap again

#matrix for heatmap plotting

#save the q-value columns for each comparison and clean up the row and column names
motif_filtered_top5_matrix_qval     <-     as.matrix(motif_df_filtered_top5[,     c(grep("q_value",
colnames(motif_df_filtered_top5)))])
rownames(motif_filtered_top5_matrix_qval)    <-    gsub("[(]",    "    (",    gsub("[)].*",    ")",
motif_df_filtered_top5$Motif_Name))
```

```
colnames(motif_filtered_top5_matrix_qval) <- gsub("_", " ", gsub(".*__", "", colnames(motif_filtered_top5_matrix_qval)))

#change to neg log10 q-values
motif_filtered_top5_matrix_neg_log_qval <- -log10(motif_filtered_top5_matrix_qval)
#replace all Inf values with the max value found in the matrix
motif_filtered_top5_matrix_neg_log_qval[motif_filtered_top5_matrix_neg_log_qval == Inf] <- max(motif_filtered_top5_matrix_neg_log_qval[motif_filtered_top5_matrix_neg_log_qval != Inf])

#make the colour gradient from min to max
col_fun_motif = colorRamp2(c(min(motif_filtered_top5_matrix_neg_log_qval), max(motif_filtered_top5_matrix_neg_log_qval)), c("#FEEFA7", "#C82226"))

#create annotations
#first an annotation for the space to plot the enrichment ratio in
#to give it a properly spaced axis, make it a point annotation
ratio_annot <- rowAnnotation("Enrichment ratio" = anno_points(seq(from = 1, to = 4.17, length.out = nrow(motif_filtered_top5_matrix_neg_log_qval)),
                                                              gp = gpar(col ="white"), axis_param = list(labels_rot = 0)), border = TRUE,
                             width = unit(9, "cm"), annotation_name_side = "top", annotation_name_rot = 0)
#then column annotations for the direction and the siRNA comparison
column_annot <- HeatmapAnnotation(Direction = get_nth_of_list(colnames(motif_filtered_top5_matrix_neg_log_qval), sep = " ", 4),
                                  col = list(Direction = c("up" = "tomato1", "dn" = "dodgerblue1")), annotation_name_side = "left")
siRNA_annot <- HeatmapAnnotation(Comparison = gsub(" up", "", gsub(" dn", "", gsub("KLA", "", gsub("dex", "", gsub("EtOH", "", colnames(motif_filtered_top5_matrix_neg_log_qval)))))),
                                 col = list(Comparison = c("siEP300 vs siEP300" = "darkorchid4", "siNON vs siEP300" = "lightskyblue", "siNON vs siNON" = "darkorange1")), annotation_name_side = "left") #

motif_filtered_top5_matrix_neg_log_qval <- motif_filtered_top5_matrix_neg_log_qval[order(rownames(motif_filtered_top5_matrix_neg_log_qval)),]

#for cleaning up column names
toSub <- c("up", "dn", "siNON", "siEP300")

hmap <- Heatmap(motif_filtered_top5_matrix_neg_log_qval,
                #give it an outline
                border_gp = gpar(col = "black", lwd = 1.5),
                #change the name of the legend
                #use expression() to get the 10 to be a subscript
                heatmap_legend_param = list(title = expression(paste("-log"[10],"  ", "FDR", sep = "")), direction = "horizontal"),
                col = col_fun_motif,
                cluster_columns = TRUE,
                show_row_dend = FALSE,
                show_column_dend = FALSE,
                show_row_names = TRUE,
                row_names_side = "left",
                column_names_side = "top",
                #rotate column names
                column_names_rot = 55,
                #use gsub() to remove all of the items saved to toSub from the column names
                column_labels = gsub(paste(toSub, collapse = "|"), "", colnames(motif_filtered_top5_matrix_neg_log_qval)),
```

```
                  row_names_gp = gpar(fontsize = 10),
                  column_names_gp = gpar(fontsize = 10.5, fontface = "bold"),
                  #set annotations
                  right_annotation = ratio_annot,
                  top_annotation = c(siRNA_annot, column_annot)
)

roword <- row_order(hmap)

#create the enrichment ratio plot to be placed in the row annotation of the heatmap using the long format data
plot <- ggplot(motif_all_df_filtered_top5_sig, aes(x = Enrichment_ratio, y = Motif_Name)) +
  #make the points a little transparent with alpha so that overlapping ones can be seen
  geom_point(aes(col = Treatment, fill = Treatment, shape = Direction), alpha = 0.8, size = 2.5) +
  #select triangles pointing up and down
  scale_shape_manual(values = c(25,24)) +
  #for those pchs selected, both a fill and colour (outline) have to be determined, make them the same colour
  scale_color_manual(values = c("#800016", "#FF002B", "#FD7936", "#E3CF24","#73D7F3", "#417FF2", "#1F0CB9")) +
  scale_fill_manual(values = c("#800016", "#FF002B", "#FD7936", "#E3CF24","#73D7F3", "#417FF2", "#1F0CB9")) +
  #the void theme will make it transparent
  theme_void(base_size = 14) +
  #order the y axis accoring to the row order on the heatmap
  scale_y_discrete(limits = rev(rownames(motif_filtered_top5_matrix_neg_log_qval)[roword])) +
  xlim(1, 4.15) +
  ylab(NULL) +
  xlab(NULL) +
  #remove all legend and axis labels so that the plot will be inserted correctly to the annotation
  guides(color = "none", fill = "none", shape = "none") +
  theme(axis.text.y = element_blank(), axis.text.x = element_blank(), axis.ticks = element_blank())

#start saving the plot
pdf(file = "motifs_heatmap.pdf", height = 14, width = 11)
png(file = "motifs_heatmap.png", height = 14, width = 11, units = "in", res = 600)
#draw the heatmap
draw(hmap, merge_legend = TRUE)

#use decorate annotation to add the enrichment ratio
decorate_annotation("Enrichment ratio", {
  #first add grid lines for the enrichment ratio part, vertical and horizontal
  grid.lines(x = unit(1, "native"), gp = gpar(col = "grey90"))
  grid.lines(x = unit(2, "native"), gp = gpar(col = "grey90"))
  grid.lines(x = unit(3, "native"), gp = gpar(col = "grey90"))
  grid.lines(x = unit(4, "native"), gp = gpar(col = "grey90"))
  for (i in 1:nrow(motif_filtered_top5_matrix_neg_log_qval)){
    grid.lines(y = unit(i, "native"), gp = gpar(col = "grey90"))
  }
  #into the current viewport (the row annotation), print the enrichment ratio plot
  vp = current.viewport()$name
  print(plot, vp = vp)
})
dev.off()
#the ordering of the rows in the heatmap may vary but overall the results are always the same

#also save separately the enrichment ratio plot with axes and legend
ggplot(motif_all_df_filtered_top5_sig, aes(x = Enrichment_ratio, y = Motif_Name)) +
```

```
  geom_point(aes(col = Treatment, fill = Treatment, shape = Direction), alpha = 0.8, size = 2.5) +
  scale_shape_manual(values = c(25,24), name = "Direction") +
  scale_color_manual(values = c("#800016", "#FF002B", "#FD7936", "#E3CF24","#73D7F3", "#417FF2",
"#1F0CB9"),
                     name = "Comparison") +
  scale_fill_manual(values = c("#800016", "#FF002B", "#FD7936", "#E3CF24","#73D7F3", "#417FF2",
"#1F0CB9"),
                    name = "Comparison") +
  theme_bw(base_size = 14) +
  scale_y_discrete(limits = rev(rownames(motif_filtered_top5_matrix_neg_log_qval)[roword])) +
  ylab(NULL) +
  xlab("Enrichment ratio") +
  guides(colour = guide_legend(override.aes = list(size=5, shape = 17),
         shape = guide_legend(override.aes = list(shape = c(25,24), fill = "black", colour =
"black")), alpha = 1))
ggsave("motifs_enrichment_ratio.pdf", plot = last_plot(), device = "pdf", width = 11, height = 9,
units = "in")
ggsave("motifs_enrichment_ratio.tif", plot = last_plot(), device = "tiff", width = 11, height = 9,
units = "in")
ggsave("motifs_enrichment_ratio.png", plot = last_plot(), device = "png", width = 11, height = 9,
units = "in")


#finally get the orig consensus sequences again to add to the figure
#orig motif consensus sequences
motifs_merged_plotting <- list()
j=1
for (i in 1:length(motifs_for_merging)){
  if(motifs_for_merging[[i]]["name"] %in% motif_df_filtered_top5$Motif_Name == TRUE){
    motifs_merged_plotting[[j]] <- motifs_for_merging[[i]]
    j <- j + 1
  }
}

#order same as on heatmap
motifs_merged_plotting <- motifs_merged_plotting[roword]

view_motifs(motifs_merged_plotting, min.overlap = 20, show.positions = FALSE, names.pos = "right",
tryRC = FALSE) +
  theme_void()
ggsave("motifs_consensus_sequences.png", plot = last_plot(), device = "png", width = 8, height = 18,
units = "in")

#write a table where the representative motifs are matched to the merged motifs
write.table(motif_df_filtered_top5[,      c("Motif_Name",      "Merged_motif")],      file      =
"merged_motifs_table.txt", row.names = FALSE)


################
## Sharedness ##
################

#######
##GENES

#piechart of how many genes are unique and how many are shared total

therows <- grep("_reg", colnames(dt), value = TRUE)
```

```
2827
2828  #take all columns with _reg
2829  piechart <- as.data.frame(dt[, ..therows])
2830  #count row sums
2831  #row sum = 1 means the gene is unique to a comparison
2832  #anything above that tells how many treatments it is DT in
2833  piechart$Sum <- rowSums(piechart)
2834
2835  #drop rows with sum = 0
2836  piechart <- piechart[piechart$Sum > 0,]
2837
2838  #count how many are unique and how many are shared
2839  #and save to data frame
2840  piechart_sums_genes <- data.frame(count = as.integer(c(nrow(piechart[piechart$Sum == 1,]),
2841                                                         nrow(piechart[piechart$Sum > 1,]))),
2842                                    comparison = c("unique", "shared"))
2843
2844  #plot pie chart
2845  #x = "" puts everything on the same bar and coord_polar changes it to a circle
2846  ggplot(data = piechart_sums_genes, aes(x = "", y = count, fill = comparison)) +
2847    geom_bar(stat = "identity", width = 1, colour = "black", alpha = 0.6) +
2848    coord_polar("y", start = 10) +
2849    geom_text(aes(label = paste0(count, "\n", comparison)), position = position_stack(vjust=0.5),
2850  fontface = "bold") +
2851    scale_fill_manual(values = c("tomato1", "dodgerblue1"), name = NULL) +
2852    guides(fill = "none") +
2853    theme_void()
2854  ggsave("genes_piechart.png", plot = last_plot(), device = "png", width = 2, height = 2)
2855
2856  #barplot by comparison
2857  #plot how many DT genes in each comparison
2858  #and how many of those are shared between how many other genes
2859
2860  #take all columns with _up or _dn and vs and count rowsums
2861  treatments <- grep("vs", grep("_up|_dn", colnames(dt), value = TRUE), value = TRUE)
2862  sharedness_genes <- as.data.frame(dt[, ..treatments])
2863  sharedness_genes$Sum <- rowSums(sharedness_genes)
2864
2865  #drop rows with sum = 0
2866  sharedness_genes <- sharedness_genes[sharedness_genes$Sum > 0,]
2867
2868  #change to long format data.frame
2869  sharedness_genes_df <- data.frame()
2870  for (i in 1:length(grep("_up|_dn", colnames(sharedness_genes)))){
2871    for (j in 1:max(sharedness_genes$Sum)){
2872      #count how many genes in each column are shared with 1,2,3,4 etc. others
2873      count <- nrow(sharedness_genes[sharedness_genes[,i] == 1 & sharedness_genes$Sum == j,])
2874      sharedness_genes_df <- rbind(sharedness_genes_df, cbind(treatments[i], j, count))
2875    }
2876  }
2877
2878  #drop those where all counts are 0
2879  for (i in 1:length(treatments)){
2880    if(nrow(sharedness_genes_df[sharedness_genes_df$V1 == treatments[i] & sharedness_genes_df$count ==
2881  0,]) == 10){
2882      sharedness_genes_df <- sharedness_genes_df[sharedness_genes_df$V1 != treatments[i],]
2883    }
2884  }
```

```
2885
2886  #clean up and set levels for factors
2887  sharedness_genes_df$count <- as.integer(sharedness_genes_df$count)
2888  sharedness_genes_df$j <- factor(sharedness_genes_df$j, levels = c("1", "2", "3", "4", "5", "6", "7",
2889  "8", "9", "10"))
2890  sharedness_genes_df$V1 <- gsub("_", " ", sharedness_genes_df$V1)
2891  sharedness_genes_df$V1 <- factor(sharedness_genes_df$V1, levels = c("siNONEtOH vs siNONdex dn",
2892  "siNONEtOH vs siNONdex up",
2893                                                                      "siNONEtOH vs siNONKLA dn",
2894  "siNONEtOH vs siNONKLA up",
2895                                                                      "siNONEtOH vs siNONdexKLA dn",
2896  "siNONEtOH vs siNONdexKLA up",
2897                                                                      "siNONdex vs siNONdexKLA dn",
2898  "siNONdex vs siNONdexKLA up",
2899                                                                      "siNONKLA vs siNONdexKLA dn",
2900  "siNONKLA vs siNONdexKLA up",
2901                                                                      "siEP300EtOH vs siEP300dex dn",
2902  "siEP300EtOH vs siEP300dex up",
2903                                                                      "siEP300EtOH vs siEP300KLA dn",
2904  "siEP300EtOH vs siEP300KLA up",
2905                                                                      "siEP300EtOH vs siEP300dexKLA
2906  dn", "siEP300EtOH vs siEP300dexKLA up",
2907                                                                      "siEP300dex vs siEP300dexKLA
2908  dn", "siEP300dex vs siEP300dexKLA up",
2909                                                                      "siEP300KLA vs siEP300dexKLA
2910  dn", "siEP300KLA vs siEP300dexKLA up",
2911                                                                      "siNONEtOH vs siEP300EtOH dn",
2912  "siNONEtOH vs siEP300EtOH up",
2913                                                                      "siNONdex vs siEP300dex dn",
2914  "siNONdex vs siEP300dex up",
2915                                                                      "siNONKLA vs siEP300KLA dn",
2916  "siNONKLA vs siEP300KLA up",
2917                                                                      "siNONdexKLA vs siEP300dexKLA
2918  dn", "siNONdexKLA vs siEP300dexKLA up"))
2919
2920  #count sums for genes in each comparison
2921  #each treatment has a sharedness ranging 1-10
2922
2923  i <- 1
2924  while (i <= nrow(sharedness_genes_df)) {
2925    sharedness_genes_df$sum[i:(i+9)] <- sum(sharedness_genes_df$count[i:(i+9)])
2926    i <- i+10
2927  }
2928
2929  ggplot(data = sharedness_genes_df, aes(x = V1, y = count, fill = j)) +
2930    geom_bar(stat = "identity", width = 1, colour = "black", alpha = 0.75) +
2931    scale_fill_brewer(palette = "RdYlBu", direction = -1, name = "Sharedness") +
2932    theme_classic() +
2933    scale_y_continuous(breaks = seq(from = 0, to = 450, by = 10), name = "Number of DE genes",
2934                       expand = c(0,0)) +
2935    xlab(NULL) +
2936    geom_text(aes(label = fifelse(round(count/sum, digits = 2)>0 & count > 2, paste0(round(count/sum,
2937  digits = 2)*100,"%"), "")),
2938              size = 2.5, position = position_stack(vjust = 0.5)) +
2939    theme(axis.text.x = element_text(angle = 340, hjust = 0),
2940          axis.text.y = element_text(colour = c("black", NA, NA, NA, NA)),
2941          plot.margin=unit(c(0.5,2,0.5,0.5), 'cm'), panel.grid.major.y = element_line(colour =
2942  "grey90"))
```

```
ggsave("genes_sharedness_absolute.png", plot= last_plot(), device = "png", width = 12, height = 14)


###########
##ENHANCERS

therows_enh <- grep("_reg", colnames(dt_enh), value = TRUE)

piechart_enh <- as.data.frame(dt_enh[, ..therows_enh])
piechart_enh$Sum <- rowSums(piechart_enh)

#drop rows with sum = 0
piechart_enh <- piechart_enh[piechart_enh$Sum > 0,]


piechart_sums_enhancers  <-  data.frame(count  =  as.integer(c(nrow(piechart_enh[piechart_enh$Sum  ==
1,]),
                                                               nrow(piechart_enh[piechart_enh$Sum      >
1,]))),
                                        comparison = c("unique", "shared"))

ggplot(data = piechart_sums_enhancers, aes(x = "", y = count, fill = comparison)) +
  geom_bar(stat = "identity", width = 1, colour = "black", alpha = 0.6) +
  coord_polar("y", start = 10) +
  geom_text(aes(label  =  paste0(count,  "\n",  comparison)),  position  =  position_stack(vjust=0.5),
fontface = "bold") +
  scale_fill_manual(values = c("tomato1", "dodgerblue1"), name = NULL) +
  guides(fill = "none") +
  theme_void()
ggsave("enhancers_piechart.png", plot = last_plot(), device = "png", width = 2, height = 2)

#barplot by comparison

sharedness_enhancers <- as.data.frame(dt_enh[, ..treatments])
sharedness_enhancers$Sum <- rowSums(sharedness_enhancers)

#drop rows with sum = 0
sharedness_enhancers <- sharedness_enhancers[sharedness_enhancers$Sum > 0,]

sharedness_enhancers_df <- data.frame()
for (i in 1:length(grep("_up|_dn", colnames(sharedness_enhancers)))){
  for (j in 1:max(sharedness_enhancers$Sum)){
    count <- nrow(sharedness_enhancers[sharedness_enhancers[,i] == 1 & sharedness_enhancers$Sum ==
j,])
    sharedness_enhancers_df <- rbind(sharedness_enhancers_df, cbind(treatments[i], j, count))
  }
}

#drop those where all counts are 0
for (i in 1:length(treatments)){
  if(nrow(sharedness_enhancers_df[sharedness_enhancers_df$V1        ==        treatments[i]        &
sharedness_enhancers_df$count == 0,]) == 8){
    sharedness_enhancers_df <- sharedness_enhancers_df[sharedness_enhancers_df$V1 != treatments[i],]
  }
}


sharedness_enhancers_df$count <- as.integer(sharedness_enhancers_df$count)
```

```
sharedness_enhancers_df$j <- factor(sharedness_enhancers_df$j, levels = c("1", "2", "3", "4", "5",
"6", "7", "8", "9", "10"))
sharedness_enhancers_df$V1 <- gsub("_", " ", sharedness_enhancers_df$V1)
sharedness_enhancers_df$V1 <- factor(sharedness_enhancers_df$V1, levels = c("siNONEtOH vs siNONdex
dn", "siNONEtOH vs siNONdex up",
                                                                             "siNONEtOH vs siNONKLA
dn", "siNONEtOH vs siNONKLA up",
                                                                             "siNONEtOH vs
siNONdexKLA dn", "siNONEtOH vs siNONdexKLA up",
                                                                             "siNONdex vs siNONdexKLA
dn", "siNONdex vs siNONdexKLA up",
                                                                             "siNONKLA vs siNONdexKLA
dn", "siNONKLA vs siNONdexKLA up",
                                                                             "siEP300EtOH vs
siEP300dex dn", "siEP300EtOH vs siEP300dex up",
                                                                             "siEP300EtOH vs
siEP300KLA dn", "siEP300EtOH vs siEP300KLA up",
                                                                             "siEP300EtOH vs
siEP300dexKLA dn", "siEP300EtOH vs siEP300dexKLA up",
                                                                             "siEP300dex vs
siEP300dexKLA dn", "siEP300dex vs siEP300dexKLA up",
                                                                             "siEP300KLA vs
siEP300dexKLA dn", "siEP300KLA vs siEP300dexKLA up",
                                                                             "siNONEtOH vs
siEP300EtOH dn", "siNONEtOH vs siEP300EtOH up",
                                                                             "siNONdex vs siEP300dex
dn", "siNONdex vs siEP300dex up",
                                                                             "siNONKLA vs siEP300KLA
dn", "siNONKLA vs siEP300KLA up",
                                                                             "siNONdexKLA vs
siEP300dexKLA dn", "siNONdexKLA vs siEP300dexKLA up"))


#count sums for enhancers in each comparison
#each treatment has a sharedness ranging 1-8

i <- 1
while (i <= nrow(sharedness_enhancers_df)) {
  sharedness_enhancers_df$sum[i:(i+7)] <- sum(sharedness_enhancers_df$count[i:(i+7)])
  i <- i+8
}

ggplot(data = sharedness_enhancers_df, aes(x = V1, y = count, fill = j)) +
  theme_classic() +
  geom_bar(stat = "identity", width = 1, colour = "black", alpha = 0.75) +
  scale_fill_brewer(palette = "RdYlBu", direction = -1, name = "Sharedness") +
  scale_y_continuous(breaks = seq(from = 0, to = 450, by = 5), name = "Number of DE enhancers",
                     expand = c(0,0)) +
  geom_text(aes(label = fifelse(round(count/sum, digits = 2)>0 & count > 1, paste0(round(count/sum,
digits = 2)*100,"%"), "")), size = 2.5, position = position_stack(vjust = 0.5)) +
  xlab(NULL) +
  theme(axis.text.x = element_text(angle = 340, hjust = 0),
        axis.text.y = element_text(colour = c("black", NA, NA, NA, NA)),
        plot.margin=unit(c(0.5,2,0.5,0.5),  'cm'),  panel.grid.major.y  =  element_line(colour  =
"grey90"))
ggsave("enhancers_sharedness_absolute.png", plot= last_plot(), device = "png", width = 12, height =
14)
```

```
3059
3060
3061    ###################################
3062    ## Overlap of genes and enhancers ##
3063    ###################################
3064
3065    ##############
3066    ##Venn diagram
3067
3068    #take those genes/enhancers that are unique and differentially transcribed and have gene names
3069    genes_df <- as.data.frame(dt[is_unique == 1 & is_deg == 1 & Gene != "0.000", c("Gene", "is_deg",
3070    "is_up", "is_dn")])
3071    enhancers_df <- as.data.frame(dt_enh[is_unique == 1 & is_deg == 1 & Gene != "0.000", c("Gene",
3072    "is_deg", "is_up", "is_dn")])
3073
3074    #combine to one data.frame
3075    df_comb <- merge.data.frame(genes_df, enhancers_df, by = "Gene", all = TRUE, suffixes = c("_genes",
3076    "_enhancers"))
3077    #replace NAs with 0
3078    df_comb[is.na(df_comb)] <- 0
3079
3080
3081    #create a list of lists with each Venn diagram comparison to be plotted
3082    toVenn   <-   list(genes   =   df_comb[df_comb$is_deg_genes   ==   1,   "Gene"],   enhancers   =
3083    df_comb[df_comb$is_deg_enhancers == 1, "Gene"])
3084
3085    #process data as above
3086    venndata_gene_enh <- process_data(Venn(toVenn))
3087    venndata_gene_enh@region$up <- c(nrow(df_comb[df_comb$is_up_genes == 1 & df_comb$is_dn_genes == 0 &
3088    df_comb$is_deg_enhancers == 0,]),
3089                                     nrow(df_comb[df_comb$is_up_enhancers == 1 & df_comb$is_dn_enhancers
3090    == 0 & df_comb$is_deg_genes == 0,]),
3091                                     nrow(df_comb[df_comb$is_up_enhancers == 1 & df_comb$is_dn_genes ==
3092    0 & df_comb$is_dn_enhancers == 0 & df_comb$is_up_genes == 1,]))
3093    venndata_gene_enh@region$dn <- c(nrow(df_comb[df_comb$is_dn_genes == 1 & df_comb$is_up_genes == 0 &
3094    df_comb$is_deg_enhancers == 0,]),
3095                                     nrow(df_comb[df_comb$is_dn_enhancers == 1 & df_comb$is_up_enhancers
3096    == 0 & df_comb$is_deg_genes == 0,]),
3097                                     nrow(df_comb[df_comb$is_dn_enhancers == 1 & df_comb$is_up_genes ==
3098    0 & df_comb$is_up_enhancers == 0 & df_comb$is_dn_genes == 1,]))
3099
3100    #define if the counts by processing Venn data match with manually added ones
3101    venndata_gene_enh@region$dif  <-  venndata_gene_enh@region$count  -  (venndata_gene_enh@region$up  +
3102    venndata_gene_enh@region$dn)
3103
3104    ggplot()+
3105      geom_sf(aes(fill=name), data = venn_region(venndata_gene_enh), show.legend = FALSE, alpha = 0.5) +
3106      geom_sf_text(aes(label = count), fontface = "bold", data = venn_region(venndata_gene_enh), size =
3107    5, nudge_y = 0.4) +
3108      geom_sf_text(aes(label = paste0(#"\n",
3109        fifelse(up != 0, paste0(up, " ", sprintf('\u2191')), paste0("")),
3110        fifelse(up != 0 & dn != 0, paste0("  "), paste0("")),
3111        fifelse(dn != 0, paste0(dn, " ", sprintf('\u2193')), paste0(""))
3112      )), data = venn_region(venndata_gene_enh), nudge_y = -0.2) +
3113      geom_sf_label(aes(label = gsub("_", " ", name)), data = venn_setlabel(venndata_gene_enh), alpha =
3114    0.5, label.size = 0, fontface = "bold", size = 5, nudge_y = 0.3) +
3115      geom_sf_text(aes(label   =   paste0(fifelse(dif   >   0   ,   paste0(dif,"   ",   sprintf('\u2191'),
3116    sprintf('\u2193')), paste0("")))),
```

```
                data = venn_region(venndata_gene_enh), nudge_y = -0.8) +
  scale_x_continuous(expand = expansion(mult = .3)) +
  scale_fill_brewer(palette = "RdYlBu") +
  theme_void()
ggsave("gene_enhancer_overlap_venn.png", plot = last_plot(), device = "png", width = 7, height = 6,
units = "in")


###############
##Network plots

#done the same way as above for genes and enhancers
#except using all up- and downregulated genes and enhancers

df_int <- data.frame(rbind(cbind(rep("genes up", length(dt[is_up == 1, Gene])), dt[is_up == 1, Gene]),
                           cbind(rep("genes dn", length(dt[is_dn == 1, Gene])), dt[is_dn == 1, Gene]),
                           cbind(rep("enhancers up", length(dt_enh[is_up == 1, Gene])), dt_enh[is_up
== 1, Gene]),
                           cbind(rep("enhancers dn", length(dt_enh[is_dn == 1, Gene])), dt_enh[is_dn
== 1, Gene]))
)
colnames(df_int) <- c("association", "gene")


nodes_int <- data.frame(label = append(unique(df_int$gene), unique(df_int$association)), row.names =
append(unique(df_int$gene), unique(df_int$association)))
nodes_int$label <- NA
nodes_int[grep("up|dn",  rownames(nodes_int),  value  =  TRUE),  "label"]  <-  grep("up|dn",
rownames(nodes_int), value = TRUE)
nodes_int$col <- "gene"
nodes_int[grep("up|dn",  rownames(nodes_int),  value  =  TRUE),  "col"]  <-  grep("up|dn",
rownames(nodes_int), value = TRUE)
nodes_int$size <- 1
nodes_int[grep("up|dn", rownames(nodes_int), value = TRUE), "size"] <- 5

ntwrk_int <- network(df_int, directed = FALSE)

ntwrk_int %v% "category" <- nodes_int[ network.vertex.names(ntwrk_int), "col"]
ntwrk_int %v% "group" <- nodes_int[ network.vertex.names(ntwrk_int), "size"]
ntwrk_int %v% "label" <- nodes_int[ network.vertex.names(ntwrk_int), "label"]

set.seed(1) #
ntwrkplot_int <- ggnetwork(ntwrk_int, layout = "kamadakawai")

ntwrkplot_int$label  <-  ave(ntwrkplot_int$label,  ntwrkplot_int$vertex.names,  FUN  =  function(a)
replace(a, duplicated(a), NA_integer_))
ntwrkplot_int$group <- as.integer(ntwrkplot_int$group)

set.seed(1)
ggplot(data = ntwrkplot_int,
       aes(x, y, xend = xend, yend = yend, color = category, size = group, label = label)) +
  geom_edges(size = 1, alpha = 0.3, show.legend = FALSE) + # draw edge layer
  geom_nodes() + # draw node layer
  geom_text_repel(seed = 1, size = 3.5, colour = "black", fontface = "bold", point.size = 1,
                  box.padding = 0.5, point.padding = 1, segment.size = 0.8,
                  min.segment.length = 0,  segment.linetype = 1, force = 30, force_pull = 0.001,
                  segment.color = "black", na.rm = TRUE) + # draw node label layer
  scale_size(range = c(1, 7)) +
```

```
  scale_color_manual(values = c("#9914db","#f20089","#808080","#077bc9","#ff5400"), labels =
c("enhancers dn", "enhancers up", "", "genes dn", "genes up")) +
  labs(color = "") +
  guides(size = "none", colour = guide_legend(override.aes = list(size=5, colour =
c("#9914db","#f20089","#FFFFFF","#077bc9","#ff5400")))) +
  theme_blank() +
  theme(legend.position = "bottom", legend.key.size = unit(0.1, "cm"), legend.text = element_text(size
= 8))
ggsave("gene_enhancer_overlap_network.png", plot = last_plot(), device = "png", width = 8, height =
8, units = "in")


##Enhancers and genes by treatment
#done the same as previously except taking all up- and downregulated genes and enhancers to one
data.frame

toNetwork_all <- grep("_reg", colnames(dt), value = TRUE)

df_all <- data.frame()
#genes up
for(i in 1:length(toNetwork_all)){
  genes <- dt[get(gsub("reg", "up", toNetwork_all[i])) == 1, Gene]
  identity <- rep("genes_up", length(genes))
  treatment <- rep(gsub("_reg", "", toNetwork_all[i]), length(genes))

  df_all <- rbind(df_all, cbind(genes, identity, treatment))
}
#genes dn
for(i in 1:length(toNetwork_all)){
  genes <- dt[get(gsub("reg", "dn", toNetwork_all[i])) == 1, Gene]
  identity <- rep("genes_dn", length(genes))
  treatment <- rep(gsub("_reg", "", toNetwork_all[i]), length(genes))

  df_all <- rbind(df_all, cbind(genes, identity, treatment))
}
#enhancers up
for(i in 1:length(toNetwork_all)){
  genes <- dt_enh[get(gsub("reg", "up", toNetwork_all[i])) == 1, Gene]
  identity <- rep("enhancers_up", length(genes))
  treatment <- rep(gsub("_reg", "", toNetwork_all[i]), length(genes))

  df_all <- rbind(df_all, cbind(genes, identity, treatment))
}
#enhancers dn
for(i in 1:length(toNetwork_all)){
  genes <- dt_enh[get(gsub("reg", "dn", toNetwork_all[i])) == 1, Gene]
  identity <- rep("enhancers_dn", length(genes))
  treatment <- rep(gsub("_reg", "", toNetwork_all[i]), length(genes))

  df_all <- rbind(df_all, cbind(genes, identity, treatment))
}

nodes_all <- data.frame(label = append(unique(df_all$genes), unique(df_all$identity)),
                        row.names = append(unique(df_all$genes), unique(df_all$identity)))
nodes_all$col <- "gene"
nodes_all[grep("up|dn", rownames(nodes_all), value = TRUE), "col"] <- grep("up|dn",
rownames(nodes_all), value = TRUE)

```

```
3233
3234
3235  ntwrk_all <- network(df_all[,c(2,1)], directed = FALSE, multiple = TRUE)
3236  ntwrk_all %v% "category" <- nodes_all[ network.vertex.names(ntwrk_all), "col"]
3237  #to be able to plot different edges for different plots, set treatment as edge attribute
3238  set.edge.attribute(ntwrk_all, "treatment", df_all[,3])
3239
3240  #to the ggnetwork function, by = is added to set the edge attribute
3241  #this is used to show the connections in each comparison separately
3242  set.seed(1)
3243  ntwrkplot_all <- ggnetwork(ntwrk_all, by = "treatment", layout = "kamadakawai")
3244  ntwrkplot_all$treatment <- gsub("_", " ", ntwrkplot_all$treatment)
3245
3246
3247  ntwrkplot_all$treatment <- factor(ntwrkplot_all$treatment,
3248                                    levels = c("siNONEtOH vs siNONdex", "siNONEtOH vs siNONKLA",
3249                                               "siNONEtOH vs siNONdexKLA", "siNONdex vs siNONdexKLA",
3250                                               "siNONKLA vs siNONdexKLA", "siEP300EtOH vs siEP300dex",
3251                                               "siEP300EtOH    vs    siEP300KLA",    "siEP300EtOH    vs
3252  siEP300dexKLA",
3253                                               "siEP300KLA    vs    siEP300dexKLA",    "siNONEtOH    vs
3254  siEP300EtOH",
3255                                               "siNONdex vs siEP300dex", "siNONKLA vs siEP300KLA",
3256                                               "siNONdexKLA vs siEP300dexKLA"))
3257
3258  set.seed(1)
3259  ggplot(data = ntwrkplot_all,
3260         aes(x, y, xend = xend, yend = yend, color = category)) +
3261
3262    geom_nodes(size = 0.1, alpha = 0.3) + # draw node layer
3263    geom_edges(size = 1, alpha = 0.6, show.legend = FALSE) + # draw edge layer
3264    labs(color = "") +
3265    scale_color_manual(values = c("#9914db","#f20089","#808080","#077bc9","#ff5400"),
3266                       labels = c("enhancers dn", "enhancers up", "", "genes dn", "genes up")) +
3267    theme_blank() +
3268    theme(legend.position = "bottom", legend.key.size = unit(0.1, "cm"), legend.text = element_text(size
3269  = 9)) +
3270    guides(colour = guide_legend(override.aes = list(alpha = 0.8, size=5,
3271                                                     colour                                                =
3272  c("#9914db","#f20089","#FFFFFF","#077bc9","#ff5400")))) +
3273    #separate by the treatment
3274    facet_wrap(~treatment, ncol = 4)
3275  ggsave("network_all.pdf", plot = last_plot(), device = "pdf", width = 14, height = 11, units =  "in")
3276  ggsave("network_all.tif", plot = last_plot(), device = "tiff", width = 14, height = 11, units =  "in")
3277  ggsave("network_all.png", plot = last_plot(), device = "png", width = 14, height = 11, units =  "in")
3278
3279
3280
3281  ######################
3282  ## Pathway analysis ##
3283  ######################
3284
3285  #generate files to input to Metascape
3286  toPathwayAnalysis <- grep("vs", grep("_up|_dn", colnames(dt), value = TRUE), value =TRUE)
3287
3288  #genes
3289  for (i in 1:length(toPathwayAnalysis)){
3290    #if there are differentially expressed genes, write them into a table
```

```
  if(nrow(dt[get(toPathwayAnalysis[i]) == 1] > 0)){
    write.table(dt[get(toPathwayAnalysis[i]) == 1, Gene], file = paste0("genes_",
toPathwayAnalysis[i], ".txt"), quote = FALSE, row.names = FALSE)
  }
}

#enhancers
for (i in 1:length(toPathwayAnalysis)){
  if(nrow(dt_enh[get(toPathwayAnalysis[i]) == 1] > 0)){
    write.table(dt_enh[get(toPathwayAnalysis[i]) == 1, Gene], file = paste0("enhancers_",
toPathwayAnalysis[i], ".txt"), quote = FALSE, row.names = FALSE)
  }
}

#read in data from excel table outputs from Metascape

#list all files and save to a variable similarly to motifs from a folder called pathway_results
pathways_genes_files <- list.files("pathway_results", pattern = "genes*")
pathways_enhancers_files <- list.files("pathway_results", pattern = "enhancers*")

pathways_genes <- lapply(paste0("pathway_results/", pathways_genes_files), read_excel, sheet =
"Enrichment")
names(pathways_genes) <- gsub("pathways_genes_", "", gsub(".xlsx", "", pathways_genes_files))

pathways_enhancers <- lapply(paste0("pathway_results/", pathways_enhancers_files), read_excel, sheet
= "Enrichment")
names(pathways_enhancers) <- gsub("pathways_enhancers_", "", gsub(".xlsx", "",
pathways_enhancers_files))


#initiate objects for loop
top_pathways_genes <- list()
genes_pathways <- vector()

#rows with Summary are duplicates, so remove those
#to reduce the number of pathways:
#first filter log_qvalue > 0.05 out
#then sort by Log(q-value) and take top 5 rows
for (i in 1:length(pathways_genes)){
  top_pathways_genes[[i]] <- as.data.frame(pathways_genes[[i]][-grep("Summary",
pathways_genes[[i]]$GroupID),])
  colnames(top_pathways_genes[[i]]) <- gsub("Log[(]q-value)", "Log_qvalue",
colnames(top_pathways_genes[[i]]))
  top_pathways_genes[[i]] <- top_pathways_genes[[i]][top_pathways_genes[[i]]$Log_qvalue <
log10(0.05),]
  top_pathways_genes[[i]] <- top_pathways_genes[[i]][order(top_pathways_genes[[i]]$Log_qvalue),]
  genes_pathways <- append(genes_pathways, top_pathways_genes[[i]][1:5, "Description"])
}

names(top_pathways_genes) <- names(pathways_genes)
#remove NAs
genes_pathways <- genes_pathways[!is.na(genes_pathways)]

#same for enhancers
top_pathways_enhancers <- list()
enhancers_pathways <- vector()

for (i in 1:length(pathways_enhancers)){
```

```
  top_pathways_enhancers[[i]]       <-       as.data.frame(pathways_enhancers[[i]][-grep("Summary",
pathways_enhancers[[i]]$GroupID),])
  colnames(top_pathways_enhancers[[i]])       <-       gsub("Log[(]q-value)",       "Log_qvalue",
colnames(top_pathways_enhancers[[i]]))
  top_pathways_enhancers[[i]] <- top_pathways_enhancers[[i]][top_pathways_enhancers[[i]]$Log_qvalue
< log10(0.05),]
  top_pathways_enhancers[[i]]                                                                    <-
top_pathways_enhancers[[i]][order(top_pathways_enhancers[[i]]$Log_qvalue),]
  enhancers_pathways <- append(enhancers_pathways, top_pathways_enhancers[[i]][1:5, "Description"])
}
names(top_pathways_enhancers) <- names(pathways_enhancers)
enhancers_pathways <- enhancers_pathways[!is.na(enhancers_pathways)]

#gather all pathways together
all_pathways <- unique(append(genes_pathways, enhancers_pathways))

#collect all comparisons from the list to a long format data.frame for plotting
top_pathways_genes_all <- data.frame()

for (i in 1:length(top_pathways_genes)){
  #add identifying information
  top_pathways_genes[[i]]$Comparison   <-   rep(gsub("_",   "   ",   names(top_pathways_genes)[i]),
nrow(top_pathways_genes[[i]]))
  top_pathways_genes[[i]]$Direction <- rep(get_nth_of_list(names(top_pathways_genes)[i], sep = "_",
4), nrow(top_pathways_genes[[i]]))
  top_pathways_genes[[i]]$Treatment     <-     gsub("     up",     "",     gsub("     dn",     "",
top_pathways_genes[[i]]$Comparison))
  top_pathways_genes_all <- rbind(top_pathways_genes_all, top_pathways_genes[[i]])
}
#change log10(q-values) for upregulated gene lists to positive
top_pathways_genes_all[grep("up",    top_pathways_genes_all$Direction),    "Log_qvalue"]    <-    -
1*top_pathways_genes_all[grep("up", top_pathways_genes_all$Direction), "Log_qvalue"]
#get InTerm_InList column from chr to numeric values
top_pathways_genes_all$InTerm_InList                                                             <-
as.integer(get_nth_of_list(top_pathways_genes_all$InTerm_InList,     sep     =     "/",     1))     /
as.integer(get_nth_of_list(top_pathways_genes_all$InTerm_InList, sep = "/", 2))

top_pathways_genes_all$Treatment <- factor(top_pathways_genes_all$Treatment, levels = c("siNONEtOH vs
siNONdex", "siNONEtOH vs siNONKLA", "siNONEtOH vs siNONdexKLA", "siNONdex vs siNONdexKLA", "siNONKLA
vs siNONdexKLA",

"siEP300EtOH vs siEP300dex", "siEP300EtOH vs siEP300KLA", "siEP300EtOH vs siEP300dexKLA", "siEP300dex
vs siEP300dexKLA", "siEP300KLA vs siEP300dexKLA",
                                                                                                 "siNONEtOH
vs siEP300EtOH", "siNONdex vs siEP300dex", "siNONKLA vs siEP300KLA", "siNONdexKLA vs siEP300dexKLA"))
#filter to keep only the determined top 5 significant pathways for each comparison
top_pathways_genes_all     <-     top_pathways_genes_all[top_pathways_genes_all$Description     %in%
all_pathways,]

#same for enhancers
top_pathways_enhancers_all <- data.frame()

for (i in 1:length(top_pathways_enhancers)){
  top_pathways_enhancers[[i]]$Comparison <- rep(gsub("_", " ", names(top_pathways_enhancers)[i]),
nrow(top_pathways_enhancers[[i]]))
  top_pathways_enhancers[[i]]$Direction <- rep(get_nth_of_list(names(top_pathways_enhancers)[i], sep
= "_", 4), nrow(top_pathways_enhancers[[i]]))
```

```
  top_pathways_enhancers[[i]]$Treatment <- gsub(" up", "", gsub(" dn", "",
top_pathways_enhancers[[i]]$Comparison))
  top_pathways_enhancers_all <- rbind(top_pathways_enhancers_all, top_pathways_enhancers[[i]])
}
#change log10(q-values) for upregulated gene lists to positive
top_pathways_enhancers_all[grep("up", top_pathways_enhancers_all$Direction), "Log_qvalue"] <- -
1*top_pathways_enhancers_all[grep("up", top_pathways_enhancers_all$Direction), "Log_qvalue"]
#get InTerm_InList column from chr to numeric values
top_pathways_enhancers_all$InTerm_InList <-
as.integer(get_nth_of_list(top_pathways_enhancers_all$InTerm_InList, sep = "/", 1)) /
as.integer(get_nth_of_list(top_pathways_enhancers_all$InTerm_InList, sep = "/", 2))

top_pathways_enhancers_all$Treatment <- factor(top_pathways_enhancers_all$Treatment, levels =
c("siNONEtOH vs siNONdex", "siNONEtOH vs siNONKLA", "siNONEtOH vs siNONdexKLA", "siNONdex vs
siNONdexKLA", "siNONKLA vs siNONdexKLA",

"siEP300EtOH vs siEP300dex", "siEP300EtOH vs siEP300KLA", "siEP300EtOH vs siEP300dexKLA", "siEP300dex
vs siEP300dexKLA", "siEP300KLA vs siEP300dexKLA",

"siNONEtOH vs siEP300EtOH", "siNONdex vs siEP300dex", "siNONKLA vs siEP300KLA", "siNONdexKLA vs
siEP300dexKLA"))
#filter
top_pathways_enhancers_all <- top_pathways_enhancers_all[top_pathways_enhancers_all$Description %in%
all_pathways,]

#how many pathways
length(unique(top_pathways_genes_all$Description))
length(unique(top_pathways_enhancers_all$Description))

#how many pathways shared between genes and enhancers
table(unique(top_pathways_genes_all$Description) %in%
unique(top_pathways_enhancers_all$Description))
table(unique(top_pathways_enhancers_all$Description) %in%
unique(top_pathways_genes_all$Description))


#combine genes and enhancers
#first determine two new identifiers
top_pathways_genes_all$Identity <- "gene"
top_pathways_enhancers_all$Identity <- "enhancer"
top_pathways_genes_all$Group <- paste0(top_pathways_genes_all$Identity, " ",
top_pathways_genes_all$Direction)
top_pathways_enhancers_all$Group <- paste0(top_pathways_enhancers_all$Identity, " ",
top_pathways_enhancers_all$Direction)
top_pathways_all <- rbind(top_pathways_genes_all, top_pathways_enhancers_all)
top_pathways_all$Group <- factor(top_pathways_all$Group, levels = c("gene dn", "gene up", "enhancer
dn", "enhancer up"))

#how many pathways total?
length(unique(top_pathways_all$Description))

#to get the labels correctly for the facet slices, save them to a vector
#where the names are the variable the faceting is done by
Group_labs <- c("Gene:\ndownregulated", "Gene:\nupregulated", "Enhancer:\ndownregulated",
"Enhancer:\nupregulated")
names(Group_labs) <- c("gene dn", "gene up", "enhancer dn", "enhancer up")

#select colours
```

```
pathways_cols <- c("#fc4c27", "#ffcd61", "#720026", "#ffe94e", "#9914db", "#f7aef8", "#c77dff",
"#54daff", "#4361ee", "#a2d2ff", "#023e8a")

#plot
#order y-axis (pathways) according to the q-values plotted
ggplot(top_pathways_all, aes(x = Log_qvalue, y = reorder(Description, Log_qvalue))) +
  geom_point(aes(col = Treatment, size = InTerm_InList), alpha = 0.8) +
  #set the sizes and breaks of the points
  #for labels, multiply by 100 to get %s
  scale_size(range = c(2, 6),
             breaks = c(min(top_pathways_genes_all$InTerm_InList), 0.05, 0.1, 0.2, 0.3,
                        max(top_pathways_genes_all$InTerm_InList)),
             labels = round(c(min(top_pathways_genes_all$InTerm_InList), 0.05, 0.1, 0.2, 0.3,
                              max(top_pathways_genes_all$InTerm_InList)), 2) * 100,
             name = "Overlap (%)") +
  scale_color_manual(values = pathways_cols, name = "Comparison") +
  theme_bw(base_size = 14) +
  ylab("Pathways") +
  xlab(expression(paste("log"[10]," ", "FDR", sep = ""))) +
  guides(size = guide_legend(order =1), colour = guide_legend(override.aes = list(size=5), alpha =
0.8, order = 2)) + #, shape = 15
  #split by gene/enhancer and up/downregulation
  #scales = "free_x" allows the x-axis to scale independently of the other slices
  facet_grid(cols = vars(Group), scales = "free_x", labeller = labeller(Group = Group_labs)) +
  theme(strip.text.x = element_text(face = "bold"),
        panel.spacing = unit(c(0.5,1,0.5), "lines"),
        axis.title.y = element_text(angle = 0, face = "bold", margin = margin(r = -50), vjust = 1.03))
ggsave("pathways_all.pdf", plot = last_plot(), device = "pdf", width = 11, height = 9, units =  "in")
ggsave("pathways_all.tif", plot = last_plot(), device = "tiff", width = 11, height = 9, units =  "in")
ggsave("pathways_all.png", plot = last_plot(), device = "png", width = 17, height = 14, units =  "in")

#write a table for which pathway terms are from which databases and their identifiers
write.table(top_pathways_all[!duplicated(top_pathways_all$Description),   c("Description",   "Term",
"Category")],
            file = "pathway_ids_and_databases.txt", sep= "\t", row.names = FALSE, quote = FALSE)
```