Universitat Politècnica de València
Escola Tècnica Superior D'Enginyeria Agronòmica i del Medi Natural

Role of chromatin modifying enzymes and histone acetylation in the regulation of the yeast metabolic cycle

Biotechnology Bachelor's Thesis
(Trabajo Fin de Grado en Biotecnología)
Academic year 2018-2019.

AUTHOR: Anastasiya Onofriychuk
TUTOR: José Javier Forment Millet
CO-TUTOR: Sonia Tarazona Campos
EXTERNAL TUTOR: Salvador Casaní Galdón

Valencia, July 2019

Title	— Role of chromatin modifying enzymes and histone acetylation in the regulation of the yeast metabolic cycle
Author	— Anastasiya Onofriychuk
Academic tutor	— José Javier Forment Millet
Co-tutor	— Sonia Tarazona Campos
External tutor	— Salvador Casaní Galdón
License	— Creative Commons, Non-Commercial (NC) and Non-Derivative Works (ND)
Location and date	— Valencia, July 2019

ABSTRACT

The yeast metabolic cycle (YMC) is a phenomenon that occurs in the baking yeast *Saccharomyces cerevisiae*. The YMC is produced at low glucose levels with a continuous feeding of the culture, and it is characterized by a cyclic expression of genes, together with cycles of oxygen consumption that repeat every 4–5 hours. These cycles feature three clearly differentiated phases, with sets of genes with characteristic functionalities expressing in each one of them.

The first one is the oxidative phase, in which genes related to ribosome and amino acid synthesis are upregulated. The reductive/building phase is characterized by peaks in genes related to mitochondria and cell cycle. The oxygen consumption diminishes in this phase. In the reductive/charging phase, there is an increase in functionalities related to non-respiratory metabolism, fatty acids consumption and storage of carbohydrates.

The characterization of the YMC has been performed by metabolic state studies, chromatin state studies and studies of gene expression, among others. For this study, public data of ChIP-Seq of three chromatin modifying enzymes, Esa1, Gcn5 and Set1 (two acetyltransferases and one methyltransferase, respectively), were used. Data of histone modifications have also been integrated, focusing on the modifications H3K18ac and H3K9ac, because of their relevance in the regulation of gene expression.

With this study, it was intended to understand the relationship between the chromatin modifying enzymes and the histone modifications H3K18ac and H3K9ac, and to study their impact on the gene expression and their importance in the YMC. To do this, ChIP-Seq data of Esa1, Gcn5 and Set were processed, and a study of the regions to which their attachment changed with time was performed, and also a multi-omics integration with histone acetylation data. These results can help to infer the possible regulation networks in which the chromatin plays an important role in the regulation of the gene expression during the YMC.

Key words: Yeast Metabolic Cycle; multi-omics integration; epigenetics; gene regulation; ChIP-Seq; histone modification

RESUMEN

El ciclo metabólico de la levadura (YMC) es un fenómeno que ocurre en la levadura *Saccharomyces cerevisiae*. El YMC se produce con bajos niveles de glucosa en una alimentación continua, y se caracteriza por una expresión cíclica de genes, junto con ciclos de consumo de oxígeno que se repiten cada 4-5 horas. Estos ciclos tienen tres fases claramente diferenciadas, y en cada una de ellas se expresa un grupo de genes que caracteriza las funciones predominantes en cada fase.

La primera fase es la oxidativa, en la que la se ven sobreexpresados genes relacionados con la síntesis de ribosomas y aminoácidos. La fase reductiva constructiva está caracterizada por picos de genes relacionados con las mitocondrias y el ciclo celular. El consumo de oxígeno disminuye en esta fase. En la fase reductiva de carga hay un aumento de funcionalidades relacionadas con el metabolismo no respiratorio, consumo de ácidos grasos y almacenamiento de carbohidratos.

La caracterización del YMC se ha hecho a través de estudios de su estado metabólico, de la cromatina o de la expresión génica, entre otros. Para este trabajo, se utilizaron datos públicos de ChIP-Seq de los modificadores de cromatina Esa1, Gcn5 y Set1 (dos acetiltransferasas y una metiltranferasa, respectivamente). También se integran datos de modificaciones de histonas, centrándose en las modificaciones H3K18ac y H3K9ac, por ser las más relevantes en la regulación de la expresión génica.

Con este trabajo, se propuso entender la relación entre los modificadores de cromatina y la modificación de las histonas H3K9 y H3K18, así como estudiar su impacto en la expresión génica y su importancia en el marco del YMC. Para ello, se procesaron los datos de ChIP-Seq de Esa1, Gcn5 y Set1, y se hizo un estudio de las regiones en las que su unión al genoma cambia a lo largo del tiempo, así como una integración multiómica con los datos de modificaciones de histonas. Estos resultados pueden ayudar a inferir las posibles redes de regulación en las que la cromatina adquiere un papel importante en la regulación de la expresión génica durante el YMC.

Palabras clave: Ciclo metabólico de la levadura; integración multiómica; epigenética; regulación génica; ChIP-Seq; modificación de histonas

ACKNOWLEDGEMENTS

Gracias a mi madre y a mi padre, Violetta y Vadym, y a mi hermana, Sofia, por darme tantas cosas y estar siempre conmigo. No habría llegado hasta aquí sin vosotros.

Gracias a Jorge, mi pareja, por darme apoyo siempre que lo necesito, por no dejarme caer, y por ser optimista por mí también cuando hacía falta.

Gracias a Salva, por guiarme durante la elaboración de este trabajo, tener paciencia conmigo y responder a todas las preguntas, por sencillas que fueran.

Gracias a Sonia, por permitirme formar parte del equipo de Genómica de la Expresión Génica, y haberme dado la oportunidad de aprender tantas cosas.

Y gracias a Javier, por enseñarme bioinformática y por mostrar que, con paciencia, de arriba abajo y de derecha a izquierda, cualquier código se puede leer y entender.

GENERAL INDEX

FIGURE INDEX

TABLE INDEX

Abbreviations

Acetyl-CoA – Acetyl coenzyme A

ATAC-Seq – Assay for Transposase-Accessible Chromatin using sequencing

ATP – Adenosine triphosphate

bp – Base pair

ChIP-chip – Chromatin immunoprecipitation with DNA microarray chips

ChIP-Seq – Chromatin immunoprecipitation followed by sequencing

DEG – Differentially expressed gene

DNA – Deoxyribonucleic acid

GEO – Gene Expression Omnibus

GLM – Generalized linear models

GO – Gene Ontology

HOC – High oxygen consumption

LOC – Low oxygen consumption

MACS – Model-based Analysis for ChIP-Seq

MORE – Multi-Omics REgulation

NADP(H) - Nicotinamide adenine dinucleotide phosphate

OX – Oxidative

PE – Paired end

PC – Principal component

PCA – Principal component analysis

RB – Reductive/Building

RC – Reductive/Charging

RNA – Ribonucleic acid

RNA-Seq – RNA sequencing

RPKM – Reads per kilo base per million mapped reads

SAGA – Spt-Ada-Gcn5-acetyltransferase

SE – Single end

SRA – Sequence Read Archive

TFIID – Transcription factor II D

TSS – Transcriptional start site

YMC – Yeast metabolic cycle

Introduction

Yeast metabolic cycle

Along the study of the living beings, it has been clearly observed that the organisms, as complex as they are, do not just function in a random manner: very often, they follow biological clocks, rhythms with underlying molecular mechanisms that coordinate them with the external conditions they are exposed to. Several specific cycles have been described for different organisms. The most known example are the circadian rhythms, which allow the organisms to adapt to ~24 hour periods and coordinate their internal functions with the environment during the whole day (Jagannath, Taylor, Wakaf, Vasudevan, & Foster, 2017; Tu & McKnight, 2006).

In 2005, the yeast metabolic cycle (YMC) was described in *Saccharomyces cerevisiae* (Tu, Kudlicki, Rowicka, & McKnight, 2005). The YMC is a 4-5 hours long ultradian cycle (meaning that it is completed within 24 hours or less) that occurs in conditions of very low, but continuous, glucose supply. It is characterized by the oscillation in oxygen consumption of the yeast culture (presenting the so called low oxygen consumption (LOC) and high oxygen consumption (HOC) periods), and the cyclic expression of nearly 50% of yeast genes (Tu, Kudlicki, Rowicka, & McKnight, 2005; Tu, et al., 2007). The cycle is very robust, being able to oscillate for around 100 cycles (Rowicka, Kudlicki, Tu, & Otwinowski, 2007).

In general, the YMC is divided in three different phases, each one characterized by a particular set of differentially expressed genes with distinctive functionalities (**Figure 1**):

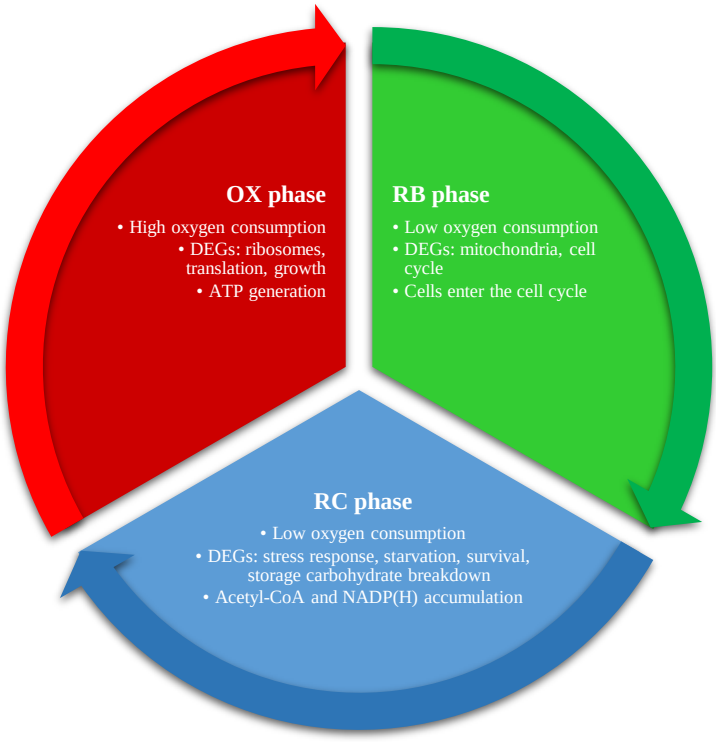

***Figure 1**. Summary of the three phases of the yeast metabolic cycle and their main characteristics: Oxidative (OX) phase, in red; Reductive/Building (RB) phase, in green; and Reductive/Charging (RC) phase, in blue.*

-The Oxidative phase, or OX phase, is characterized by a quick consumption of oxygen and the expression of genes related to growth, translation, ribosomes and amino acid biosynthesis (especially if dependent on NADP(H)). The accumulated acetyl-CoA is used, and ATP is

generated (Kuang, et al., 2014; Rao & Pellegrini, 2011; Tu, et al., 2007). It is followed by the Reductive/Building phase.

-The Reductive/Building phase, or RB phase, is characterized by high oxygen consumption (with a diminishment at its end), and the expression of genes related to mitochondria and the cell cycle (Kuang, et al., 2014; Rao & Pellegrini, 2011). It is followed by the Reductive/Charging phase.

-And the Reductive/Charging phase, or RC phase, characterized by little oxygen consumption, the expression of genes related to stress response, starvation response and survival, and the accumulation of acetyl-CoA (Kuang, et al., 2014; Rao & Pellegrini, 2011; Tu, et al., 2007). It is followed by the Oxidative phase again, closing the cycle.

In the OX phase, the oxygen levels in the yeast cell cultures drop, due to its high consumption rate. The oxidative metabolism is prevalent. The cell uses acetyl-CoA, accumulated in the previous phase, in order to produce energy in form of ATP molecules, probably preparing for the cell division in the next phase. NADP(H) also peaks at the beginning of this phase; this is associated with its role as precursor of reductive compounds, which would be useful to protect the cell, and especially its DNA, from oxidative damage. Genes related to translation and ribosome biogenesis are differentially expressed. Several amino acid synthesis processes are also enriched in this phase, mainly those dependent on NADP(H) (Kuang, et al., 2014; Rao & Pellegrini, 2011; Tu, et al., 2007).

At the beginning of the RB phase, the oxygen consumption rate is still high. Genes related to mitochondria and their biogenesis are more expressed. It is in this phase when nearly 50% of the cells enter the cell cycle; it is though that the remaining cells do not behave the same way as they come from the immediately previous cell division, so have still not grown enough to divide again (Burnetti, Aydin, & Buchler, 2016; Rowicka, Kudlicki, Tu, & Otwinowski, 2007). Consequently, the expressed genes are enriched in those related to the cell cycle. At the end of the phase, the oxygen consumption abruptly ceases: the metabolism is not oxidative anymore, probably to protect the DNA during the cell division (other studies, however, suggest that the DNA replication is connected to the catabolism of storage carbohydrates, and that it is not always separated from the HOC phase (Burnetti, Aydin, & Buchler, 2016)). Additionally, ethanol and acetate levels rise in early stage of this phase (Kuang, et al., 2014; Rao & Pellegrini, 2011; Tu, et al., 2007).

In the RC phase, oxygen levels remain low, as in the RB phase. Acetyl-CoA starts to accumulate, and so does NADP(H); both compounds peak at the end of the phase. Several enzymes from the pentose phosphate pathway, related to NADP(H) generation, are upregulated. Many amino acid synthesis processes are relevant, generally involving those not dependent on NADP(H). Many differentially expressed genes are involved in glycolysis, ethanol usage, fatty acid oxidation, breakdown of storage carbohydrates and protein degradation. Carnitine is abundant (Rao & Pellegrini, 2011; Tu, et al., 2007). Also, many genes in RC are enriched in functions as response to stress, response to starvation and survival (Kuang, et al., 2014).

Although the oscillation among the phases is precise, they are unevenly distributed in time; normally, the LOC period lasts longer than the HOC one. However, the exact timing of the cycle phases varies from one experiment to another, depending on the yeast strain or the dilution rate of glucose (Burnetti, Aydin, & Buchler, 2016).

In the last years, the cycle has been studied at many different levels, and the relation of many elements with the regulation of the YMC has been discovered. For instance, it has been seen that Acetyl-CoA levels are very important for the cycle: it is its high levels in the cells what triggers the transition from the RC to the OX phase, and its accumulation is closely linked to histone acetylation changes (Cai, Sutter, Li, & Tu, 2011). Different acetyltransferases, as Esa1 and Gcn5 (Cai, Sutter, Li, & Tu, 2011; Kuang, et al., 2014), histone modifications (Kuang, et al., 2014; Sánchez Gaya, et al., 2018), and also transcription factors (Rao & Pellegrini, 2011; Sánchez Gaya,

et al., 2018), have been implicated with the cycle, too. But, despite the different findings, the exact mechanism of YMC regulation still stays unclear.

It seems very likely that the yeast metabolic cycle is regulated at many different levels in the cells, and that the precisely orchestrated oscillation is the result of the combination of several molecular mechanisms. Because of this variability among the cycle regulators, and the variety of transcribed genes and triggered functionalities, a wide, multi-omics approach seems suitable for the study of the YMC.

Omics sciences

The high-throughput technologies have rapidly evolved in the last years. From Sanger's sequencing to the latest NGS techniques, the last decades supposed a great advance in this field, both in its capabilities and its price; and the informatic tools, absolutely required for the data processing, are being developed in parallel with it (Manzoni, et al., 2016).

The capability of obtaining huge amounts of information in an affordable manner is translated in the availability of lots of data at a global scale. These large amounts of data create the need of changing the approach by which these data are studied. Now that not only some genes are sequenced at the same time, but the whole genome, or not only one metabolite is analysed, but the whole metabolome, the so-called omics sciences are becoming crucial for the study of an organism as a whole.

While the omics sciences focus on molecules of only one biological level, whether they are DNA (genomics), RNA (transcriptomics), metabolites (metabolomic), proteins (proteomics), epigenetic modifications (epigenomics), etc., their study tries to encompass the entirety of these elements (Schneider & Orchard, 2011). This way, the omics data sets are very large, and offer a global view of the analysed sample.

In this study, data regarding transcripts, histone acetylation sites and chromatin modifying enzyme interaction sites are used, combining three different omics sciences: transcriptomics, epigenomics and cistromics.

Transcriptomics

Transcriptomics is the omics science that studies the transcriptome of an organism, in other words, all the RNA transcripts that it harbours, whether they are coding or non-coding. The most obvious applications of transcriptomics are the detection and quantification of the genes that are expressed in a sample or the discovery of differentially expressed genes (DEGs) in certain conditions. Other usages include the study of alternative splicing and quantitative assessment of genotype influence on gene expression (Manzoni, et al., 2016).

The techniques that are usually used in transcriptomics are RNA-microarrays and RNA-Seq. While the former is cheaper and well adjusted to already known organisms, the latter allows the discovery of unknown transcripts and variants as it is not based on a predesigned set of sequences and does not require *a priori* knowledge, despite being more difficult to process afterwards (Manzoni, et al., 2016).

Epigenomics

The epigenome, the complement of the different epigenetic modifications that occur in an organism, is the object of study of epigenomics. This field encompasses all the different reversible modifications that a genome can suffer without changing the DNA sequence, and which influence the gene expression, as can be the DNA methylation or the histone modifications (as acetylation, methylation, phosphorylation, ubiquitination, etc.) (Kouzarides, 2007; nature.com, n.d.).

Just as the transcriptome, the epigenome of a cell is dynamic, meaning that the epigenetic modifications change depending on many different factor, including developmental state and environment (Maunakea, Chepelev, & Zhao, 2010). To study the epigenome, experiments that

can detect these modifications are chosen, as can be DNA methylation microarrays, ChIP-chip, ChIP-Seq, ATAC-Seq assays among others.

Cistromics

It is defined as "cistrome" the set of cis-acting targets of a trans-acting factor on a genome-wide scale, referring to the different non-coding sites across the genome where, for example, a transcription factor, an enzyme or another protein interacts with and binds the DNA (Griffiths, Miller, Suzuki, Lewontin, & Gelbart, 2000; Liu, et al., 2011).

Cistromics is considered the omics science that studies the cistrome. Its research often uses assays as ChIP-chip, ChIP-Seq, DNase-Seq or ATAC-Seq among others (Zheng, et al., 2019). This omics, however, is usually combined with other studies, to obtain a wider picture of the studied regulatory elements and interactions (Jiang & Mortazavi, 2018; Liu, et al., 2011).

ChIP-Seq: a tool for omics sciences

Genome-wide studies are crucial for the understanding of the regulation of gene expression. Since its first usage in 2007, chromatin immunoprecipitation (ChIP) followed by sequencing (Seq), or ChIP-Seq, has shown to be a very efficient technique to perform genome-wide research, allowing the detection of a great amount of DNA-regulator interaction sites along the whole organism's genome in a single experiment (Park, 2009). This makes ChIP-Seq a very useful technique in omics sciences like epigenomics and cistromics.

As its own name suggests, ChIP-Seq technique consists in an immunoprecipitation of fragments of chromatin (linked to a protein) and a massive sequencing of these fragments. This way, the sample's DNA is filtered, enriched in the sequences that had a certain interaction.

To perform a ChIP-Seq, first of all, the interaction of interest has to be defined. It may be a protein, a chromatin modifying enzyme, a transcription factor (TF), etc., or a histone with a specific modification on it; this differentiates the non-histone and histone experiments (Park, 2009). Once chosen, formaldehyde application to the cell culture causes the crosslinking of the interacting proteins with the DNA. The chromatin is then fragmented by sonification, and purified: first, with immunoprecipitation, by using antibodies specific for the studied interaction, enriching the sample; and then, without it, so only DNA fragments remain (Nakato & Shirahige, 2017).

After the enriched and purified DNA is obtained, it is sequenced; the huge amount of sequences resulting from one experiment makes essential the use of high-throughput technologies. The reads are then aligned to the reference genome of the studied organism, and a peak calling procedure is performed to identify the peaks, regions of the genome that are enriched in these reads (Nakato & Shirahige, 2017). Further processing of these peaks makes it possible to infer regions of the genome that are regulated by the studied molecular elements, perform functional enrichments, integrate them with gene expression, try to discover motifs… (Park, 2009)

The ChIP-Seq technique presents several advantages. In contrast to its predecessor, the ChIP-chip assay, the detected sequences are not limited to a predefined set, repetitive regions can be covered, and smaller quantities of DNA sample are needed. The data coverage and resolution are also higher (Park, 2009). As hundreds of ChIP samples can be sequenced at the same time, data from different regulatory elements can be combined, integrating them in a multi-omics manner (Nakato & Shirahige, 2017).

But, despite the versatility that this assay offers, ChIP-Seq also has to face different challenges when executed, and many factors have to be taken into account in order to achieve a good result. On the one hand, the experimental protocol is crucial. The design must establish an adequate depth of sequencing to detect all the significant peaks, and consider only those with a minimum fold enrichment; a control experiment must be prepared, in order to contrast the found peaks with those that appear in a non-enriched sample and to differentiate noise from signal later. The enrichment highly depends on the immunoprecipitation step, as the antibodies' quality,

specificity, and possible cross-reactivity are determinant for the correct detection of interaction sites (Nakato & Shirahige, 2017; Park, 2009; Thomas, Thomas, Holloway, & Pollard, 2017). Limitations as the capability of the technique of detecting only one histone modification at each time, even if there are multiple modifications due to ChIP particularities (Kouzarides, 2007), or the elevated price of its performance, also have to be considered.

On the other hand, the posterior computation analysis of the data also has to be well adjusted. The large amount of managed data requires specific software for its management. Sequenced reads' quality has to be evaluated, and those with insufficient quality must be discarded, in order to avoid the accumulation of error in the next steps of the analysis. Aligners have to be suitable for short reads, and allow some mismatches that could appear due to differences with the reference genome or sequencing errors. The peak calling software must identify the peaks according to the nature of the studied interaction, considering its corresponding shifting length on the genome and whether the enriched regions appear as sharp, broad or mixed peaks (Nakato & Shirahige, 2017; Park, 2009). And, of course, to further associate the peaks with other elements (genes, transcription factors, binding motifs, functional annotation, etc.), specific software, together with specialized databases of biological information, must be available for the research.

Multi-omics integration

When the concept of integration is used in the context of bioinformatics, it refers to the process in which different kinds of biological data, coming from different omics experiments, are combined in order to obtain a global view of the studied subject, by comparing, contrasting and connecting the available information. This way, the study becomes a multi-omics analysis (Manzoni, et al., 2016).

A multi-omics approach is much more realistic than an individual omics study, as every organism is actually a huge amount of very different biomolecules interacting together and influencing on each other is a coordinated manner, So, integrating all kind of information to understand the functioning of an organism seems reasonable. However, this approach implies huge challenges at the analysis and computation levels (Misra, Langefeld, Olivier, & Cox, 2018).

Several methods are used for the data integration, including exploratory methods based on dimensional reduction (as principal component analysis, or PCA), clustering, networks learning, and regression models (Zeng & Lumley, 2018). The existence of tools that are able to perform these statistical analyses on biological data is very important to understand the molecular networks that regulate gene expression.

Another essential element to make possible the omics and multi-omics approaches is the existence of tools, public repositories and databases that can store, provide and share biological information from previous studies (Schneider & Orchard, 2011). The availability of data to all the scientific community is key for data integration, and by extent, for the improvement of the knowledge that we have about the living beings. Ensembl (Zerbino, et al., 2018), BioMart (Durinck, et al., 2005) or Gene Expression Omnibus (Edgar, Domrachev, & Lash, 2002) are some examples of this idea.

Gene expression and histone modification in yeast

The process by which the genetic information of a cell flows from DNA to RNA, and then results in the synthesis of a polypeptide chain, following the main dogma of molecular biology, is called gene expression (Watson, et al., 2014). This process takes several steps until it yields a functional protein: transcription of DNA into RNA, previous RNA processing (like splicing or RNA editing) when needed, translation of the RNA into a polypeptide chain, and even posterior modifications of the polypeptide. The complete process varies between the prokaryotic and eukaryotic cells, among the different organisms, and even among the different proteins of the same cell.

Gene expression can be seen as a complex assembly line that the cell executes at a molecular level. And, to execute it correctly and synthesize the adequate product, several regulation mechanisms control the whole process at different levels: changes in chromatin, regulation of transcription, RNA splicing and processing, RNA transport, RNA degradation, translational modification, protein modifications (Klug, Cummings, Spencer, & Palladino, 2012)…

Chromatin modification is the first step of genetic expression in eukaryotic cells. The genes that are about to be expressed must be made accessible for the transcriptional machinery of the cell, and this is achieved by altering the association of the DNA with other chromatin components. The alterations that may take place include DNA methylation, chromatin remodelling by histone repositioning and histone modification (Klug, Cummings, Spencer, & Palladino, 2012).

In the baking yeast *Saccharomyces cerevisiae*, each one of the four core histones that form the nucleosome (H2A, H2B, H3 and H4) is encoded by two genes. Their transcription is controlled by several regulators (Hir, Hpc, Spt10, Asf1, Swi4…), and different post-translational mechanisms influence their levels in the organism (Rando & Winston, 2012).

The nucleosomes in *S. cerevisiae* distribute differently through the chromatin depending on the type genes. In the case of growth genes, the region that harbours transcription factor binding sites and TATA-less promoters is very depleted in nucleosomes, and it is flanked by two nucleosomes that are very robustly positioned, while others, upstream, very from cell to cell. In stress genes, the region dedicated to the regulatory sequences is occluded by the nucleosomes, and the interaction is possible when a change in their positioning occurs (Rando & Winston, 2012) .

In both cases, the nucleosomes may influence the gene expression not only with their position in chromatin, but also by the way the histones are modified. The modifications that can occur on histones include acetylation, methylation, phosphorylation, ubiquitination or sumoylation, among many other. And, in most of them, different chromatin modifying enzymes are involved (Kouzarides, 2007).

Acetylation is considered one of the main histone modifications. In consists in the addition of an acetyl group to the lysine residues of the histones. In general, this modification is associated with the activation of the transcription (Kouzarides, 2007). Two important chromatin modifying enzymes that are responsible for histone acetylation are the Gcn5 acetyltransferase, part of the SAGA HAT complex, and Esa1 acetyltransferase, part of the NuA4 HAT complex (Rando & Winston, 2012).

Gcn5 has been associated to the acetylation of H3 and H2B histones, while Esa1 has been mainly related to the H4 histone, but also to H2A and H2B (Kouzarides, 2007; Rando & Winston, 2012). It is thought that both enzymes act in a redundant manner, affecting similar genes and functions in the cell (Rando & Winston, 2012).

Histone methylation (addition of a methyl group to a lysine or an arginine residue) is also considered an important modification. The methyltransferases that perform this modification are much more specific to the modified residues when compared to the acetyltransferases (Kouzarides, 2007). One of the many histone methyltransferases present in yeast is Set1, part of the COMPASS complex, associated to the methylation of the H3K4 residue (Rando & Winston, 2012).

Objective

The intention of this Bachelor's thesis was to uncover the relationship between chromatin modifying enzymes, histone acetylation and the transcription factors that might be involved in the regulation of the yeast metabolic cycle in *Saccharomyces cerevisiae*. To do so, public data of one of the widest omics YMC studies up to date (Kuang, et al., 2014) were analysed in a bioinformatic manner.

Two main objectives were pursued in this study:

- The processing of ChIP-Seq data of three chromatin modifying enzymes -Esa1, Gcn5 and Set1- obtained from different stages of the YMC, in order to detect their interaction sites during the cycle in a genome-wide manner.
- The multi-omics integration of the ChIP-Seq data of the three enzymes with ChIP-Seq data related to the acetylation of two histone residues that have been seen to be important in the regulation of the YMC, H3K9 and H3k18.

Materials and methods

Data origin

In order to perform this study, public RNA-Seq data and ChIP-Seq data (regarding the histone acetylations H3K9 and H3K18, and the chromatin modifying enzymes Esa1, Gcn5 and Set1) from yeast cultures undergoing the YMC were used. A ChIP-Seq of histone H3 was also used as a control sample for the peak calling procedure. All the data had been obtained at different time points of the yeast metabolic cycle (YMC).

All these omics datasets were obtained by Kuang et al. (2014) (RNA-Seq, ChIP-Seq of H3K9, H3, Esa1, Gcn5 and Set1) and by Dr. Jane Mellor's laboratory (ChIP-Seq of H3K18) (Sánchez Gaya, et al., 2018). Additionally, RNA-Seq data, and ChIP-Seq data for histone modifications H3K9 and H3K18 had undergone a previous processing (Sánchez Gaya, et al., 2018), and were provided directly as processed data. In this work, ChIP-seq sequencing data for chromatin modifying enzymes and histone H3 were processed, as will be described later in this section.

As described in more detail in Kuang et al. work, Illumina HiSeq 2000 was used to perform the RNA-Seq (single-end reads, 50 bp long), and Illumina HiSeq 2000 ChIP, Illumina Genome Analyzer ChIP, and AB SOLiD System were all used to perform the ChIP-Seq (single-end reads, read lengths of 35-51 bp).

General plan

In order to better understand the study that was performed on the obtained ChIP-Seq data, it was divided in four major parts. All the main steps can be seen in **Figure 2**.

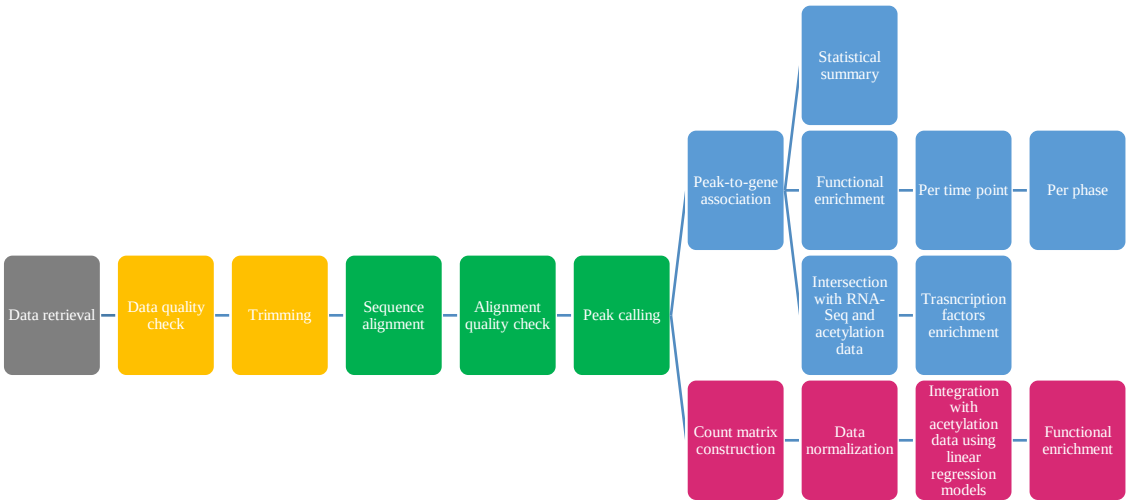

***Figure 2**. Scheme of the bioinformatic pipeline followed in this study. Each colour represents a step in the process: grey for the data retrieval, yellow for the first data preprocessing, green for the peak calling step (including the sequence alignment and its quality check), blue for the peak-to-gene association, and the related procedures of statistics summary, functional enrichment, or intersection with acetylation, and finally magenta for the statistical integration with acetylation data and posterior functional enrichment.*

-Firstly, the raw fastq files were preprocessed: their quality was checked and improved with trimming. This way, the data was prepared to be used in the following steps.

-Next, a peak calling was performed, including a previous sequence alignment to the yeast genome and its quality control.

-After that, the obtained peaks were associated to yeast genes by proximity. A functional enrichment on these genes allowed a general, qualitative, view of the sequencing data of the enzymes, showing the representative functions at each time. Additionally, these genes were intersected with RNA-Seq and acetylation ChIP-Seq data, and then enriched in transcription factors.

-Finally, in order to associate the ChIP-Seq data of the enzymes and the ChIP-Seq data of the acetylations, linear regression models were made for each gene. To do so, count matrixes for each ChIP-Seq data set were generated. This involved a more quantitative approach, yielding a set of genes with a significant association coefficient. These genes were functionally enriched, too.

All these different procedures are part of a bioinformatic pipeline, being the output of one step the input of the next one. Note that, as it is reflected in the **Figure 2**, both the third and the fourth sections of the study part from the same point (the peak calling output), so could have been performed in any order.

In the following sections, all the steps of the study will be explained in detail.

Data retrieval

The analysed ChIP-Seq data of the three histone modifying enzymes and the H3 histone (Kuang, et al., 2014) were obtained from the gene expression data repository GEO (Edgar, Domrachev, & Lash, 2002), with the following accession number: GSE52339.

The retrieval of the files was performed by using `fastq-dump`, a tool from the SRA Toolkit (SRA Toolkit Development Team, n.d.). This toolkit allows connecting to and using data from SRA, an international public resource that archives next-generation sequencing data (Leinonen, Sugawara, Shumway, & International Nucleotide Sequence Database Collaboration, 2011). Particularly, `fastq-dump` allows converting the SRA data into fastq format files, which are saved in user's working directory by default.

This way, 58 fastq files were obtained: 14 for each histone modifying enzyme (1 per each one of the 14 time points), and 16 (1 per each time point) for the histone H3, used as control.

Acetylation ChIP-Seq data was originally retrieved from Kuang et al. (2014), but was previously processed by Sánchez Gaya et al. (2018); H3K18 acetylation ChIP-Seq data was provided by Dr. Jane Mellor, and was also previously processed by Sánchez Gaya et al. (2018). Because of that, these data were provided directly in the form of normalized count matrixes. The raw reads of H3K18 ChIP-Seq, however, can be retrieved from GEO, with the accession number GSE118889.

The set of differentially expressed genes in each phase of the YMC was obtained from the RNA-Seq data of Kuang et al. (2014), and was also processed by Sánchez Gaya et al. (2018). The original files can also be accessed from GEO, with the same accession number as the ChIP-Seq of the histone modifying enzymes.

Chromatin modifying enzymes' ChIP-Seq data preprocessing

In order to use the retrieved ChIP-Seq data, the raw data files had to be previously processed. This included a quality check followed by a trimming step.

Quality analysis

First of all, each fastq file's quality was analysed with FastQC (Andrews, 2010).

FastQC is a tool that provides different controls when given high throughput sequencing data. These controls help determining whether the considered reads are fit for further bioinformatic procedures (as could be the sequence alignment). The detection of different problems and biases at the early stages of the pipeline is very important, as it helps avoiding error accumulation in the further downstream analysis.

The output of this quality analysis is a report in html format, in which the results are conveniently shown in a visual manner. The report includes basic statistics of the sequencing; charts with per base sequence quality, per tile sequence quality, per sequence quality scores, per base sequence content, per sequence GC content, per base N content, sequence length distribution, sequence duplication levels and adapter content; and a list of overrepresented sequences.

In case of an analysis in which the reads were to show a poor quality, due to, for example, the presence of adapter sequences in the reads, trimming the sequencing reads to remove such adapter sequences would be necessary.

Trimming

Next, fastq files underwent trimming, a procedure that consists in removing the adapter sequences and the low quality base pairs from the provided reads, improving their quality and facilitating their correct alignment. The trimming is especially important when reads are of a short length, since in these cases the adapter contamination is higher.

The trimming was performed with Trimmomatic (Bolger, Lohse, & Usadel, 2014), version 0.36. It is a preprocessing tool for next-generation sequencing data, being capable of working with both single end and paired end reads, and allowing a significant improvement of the sequencing data quality by applying different processing steps to the reads.

The parameters used in Trimmomatic were:

```
java -Xmx512M -jar $<Trimmomatic route>/trimmomatic-0.36.jar SE <fastq file route> <Route for trimmed file to be saved> ILLUMINACLIP:<Route of fasta file with adapter sequences>:2:30:10 LEADING:25 TRAILING:25 SLIDINGWINDOW:5:25 MINLEN:25
```

With the parameter `ILLUMINACLIP`, the adapter sequences indicated by the fasta file were cut, with the maximum mismatch allowed of 2 bases, the accuracy of the match between two reads (in case of palindrome alignment) of 30, and the accuracy of a match between any adapter sequence against a read of 10. `LEADING` and `TRAILING` cut bases off the start and the end of a read, respectively, if the threshold quality they had was below the selected value (25 in this case). The `SLIDINGWINDOW` parameter made Trimmomatic scan a read from the 5' end and cut it once the average quality within a window (5 bases) was lower than a threshold (25). Finally, `MINLEN` indicated the minimum length that a read should have (25 bases).

The single end reads mode (`SE`) was chosen for the trimming, as this was the nature of the analysed data.

The fasta files containing different adapter sequences were constructed specifically for each enzyme's ChIP-Seq dataset, and included several adapter sequences that appeared as overrepresented in the FastQC analysis.

Once the trimming was finished, another quality check with FastQC was performed on each file. Once the reads displayed an adequate condition, it was possible to continue with the ChIP-Seq analysis and proceed with the sequence alignment.

Peak calling

The peak calling is an essential step in the analysis of ChIP-Seq data. It consists in the detection of genomic regions enriched in aligned reads (the so called "peaks") and the testing of their statistical significance (Thomas, Thomas, Holloway, & Pollard, 2017). Therefore, this method allows finding the genomic sites where protein-DNA interactions have occurred with more confidence.

The performance of a peak calling requires a previous alignment step, where the reads are mapped to a reference genome, to localize the potential interaction regions. Additionally, it requires a control or input sample, in order to compare the detected peaks with reads from a sample that was not enriched in protein-DNA interactions and calculate noise to signal ratio.

Sequence alignment

The processed reads were aligned to the reference genome of *Saccharomyces cerevisiae*.

To do so, firstly, the fasta file of the reference genome was retrieved from the Ensembl genome database (Zerbino, et al., 2018), release 91, corresponding to the same genome version that was used by Sánchez Gaya et al. (2018).

Once obtained, the file was used to create the index of the genome using Bowtie 2 (Langmead & Salzberg, 2012), which is a sequencing reads aligner aimed at relatively short reads and long genomes. The indexing tool `bowtie2-build` gave, as a result, several bt2 files that were used to make the alignment of ChIP-Seq reads.

The alignment itself was also performed by using Bowtie 2, with the aligner's default options. It aligned the previously trimmed reads to the indexed yeast genome, generating one mapped sam file per each fastq file.

Mapping quality

Once the alignment of all the ChIP-Seq reads was finished, its quality was checked with Qualimap 2 (Okonechnikov, Conesa, & García-Alcalde, 2016), a tool designed to evaluate the quality of alignment sequencing data of many types of experiments, including ChIP-Seq. The tool generates an html report, which includes a statistical summary of the alignment and different charts regarding the coverage, GC content, mapping quality, etc.

Among the different modes Qualimap features, *Multi-sample BAM QC* was chosen, as it allows analysing the quality of multiple alignment files at once, in a combined way.

However, in order to use Qualimap, all the sam files had to be previously converted into bam format. For this purpose, SAMtools (Li, et al., 2009), a toolkit that offers different utilities for the processing of read alignments, was used. The conversion (and required assortment and indexing) was performed with several tools (`view`, `sort` and `index`), using the following commands and parameters:

```
samtools view -bS <sam file> > <bam file>

samtools sort <bam file> -o <sorted bam file>

samtools index <sorted bam file>
```

Once all the sam files were converted into bam files, *Multi-sample BAM QC* was run, for the 14 samples of each enzyme ChIP-Seq together, with the following parameters:

```
qualimap multi-bamqc -c -d <List of directories of the files to analyse> -r
-outdir <Output directory for Qualimap 2 report>
```

As one of the required parameters for *Multi-sample BAM QC* analysis is a list with all the directories to the bam files, three txt files (one per each dataset) were previously created, with two

tab-separated columns indicating the name of the sample (the first one) and the route to the corresponding file (the second one).

Control files merging

All the previous procedures were applied equally to the ChIP-Seq data of the three enzymes, and to the ChIP-Seq data of histone H3. However, when it came to the peak calling, a problem with these data was encountered: enzymes' ChIP-Seq consisted of 14 files for each enzyme, as the samples were collected at 14 different time points, while in case of histone H3, the number of files was 16, as the samples were taken at 16 time points, without exact temporal correspondence to the other ones.

Because of that, in order to use H3 ChIP-Seq as a control for the peak calling, the corresponding 16 sam files generated from the sequence alignment were merged into one. This was done considering that H3 data are not supposed to vary significantly with time, so a common file could be used as control for the peak calling of all 14 time points.

The merging of the 16 sam files was performed with the `merge` tool, part of the SAMtools toolkit (Li, et al., 2009), that combines all the input data into one single file.

Peak calling

After the quality control of the alignment, the peak calling was performed on the sam files. MACS2's `peakcall` was the tool used to do it (Zhang, et al., 2008), with the following parameters:

```
macs2 callpeak -t <Route to aligned sam file> -c <Route to control file> -f
SAM -g 1.2e7 -p 0.01 -n <Route for output files> -B --nomodel --extsize 147
```

For the peak calling, a p-value (`-p`) of 0.01 was chosen, as in other studies that used this software for ChIP-Seq peak calling (Law & Finger, 2017). The genome size (`-g`) was set to $1.2 \cdot 10^7$ base pairs, to correspond to the size of *S. cerevisiae* genome (Goffeau, et al., 1996). `-B` parameter allowed generating bedGraph files.

As initially MACS2 failed to construct a model of the shift size of the ChIP-Seq tags, a sequence length to which extend the reads in 5'->3' direction had to be defined (`--extsize`). A length of 147 bases was chosen (as the length of the core DNA on a nucleosome (Watson, et al., 2014)) for this purpose, as has been done in other studies with a similar approach (Law & Finger, 2017; Wang, et al., 2016). As the shift size was defined manually, the parameter `--nomodel` had to be used.

As mentioned previously, the merged file of the H3 histone alignments was used as a control for all the peak calling procedures.

This gave as an output narrowPeak, bed, xls and bdg files, one of each per each processed sam file.

Peak-to-gene association

Once the peak calling was finished and the peak files were generated, the bed files from this process were passed to the tool RGmatch (Furió Tarí, Conesa, & Tarazona, 2016). RGmatch is a Python based software able to associate given genomic regions to their closest genes, transcripts or exons, and with the regions within which they fall, as could be the TSS, the TTS, the promoter sites, etc. The associations are reported in the form of a txt file.

As a requirement to do the association of genomic regions (ChIP-Seq peaks, in this case) and features, RGmatch needed a gft file of *S. cerevisiae*, the annotation of the organism's genome. This annotation was retrieved from Ensembl (Zerbino, et al., 2018), considering the release 91, the same as for the fasta file of the reference genome.

Once the gtf file was obtained, each bed file obtained in the peak calling was processed with the tool, using the following parameters:

```
python rgmatch.py -g <Route to gtf file> -b <Route to bed file> -o <Route for output txt> -r 'gene' -q 3
```

As it is indicated, `-r` was set as `gene`, in order to report the associations at the gene level. The maximum distance to report an association (`-q`) was set at 3 kilobases.

Analysis of RGmatch results with R

The tool RGmatch gave, as output, a txt file for each sample, indicating the peak and the genetic feature to which it was associated. Given the 42 files, this information was processed with the programming language R (R Core Team, 2018), through the RStudio software (RStudio Team, 2016).

The analysis pipeline included a statistical summary of the RGmatch results, the functional enrichment analysis of the associated genes, the intersection of these genes with the differentially expressed genes of RNA-Seq and H3K9 and H3K18 acetylations' ChIP-Seq, and a transcriptional factor enrichment.

Statistical summary

Some summary statistics were computed from the RGmatch results: numbers of detected and associated peaks, number of associations, and numbers and percentages of peaks associated to concrete genic features. To visually represent this information, the data was also represented, using the ggplot2 package (Wickham, 2016), as a density plot.

Functional enrichment analysis

Genes with an association to a peak where used to study the enrichment of GO terms related to them (Ashburner, et al., 2000; The Gene Ontology Consortium, 2019). The enrichment was performed at a time point level and at YMC phase level, separately for each chromatin modifying enzyme. It was done against the rest of genes of the organism.

However, in order to perform the functional enrichment, the detected genes were previously filtered. Only genes whose peaks were related to a transcription start site (TSS) or to a promoter were considered, as these are the ones most likely related to transcriptional regulation.

Functional enrichment at a time point level

The filtered genes of each chromatin modifying enzyme's sample (14 samples per enzyme, corresponding to the 14 time points) were enriched separately.

The GO term data for *S. cerevisiae*, as well as the Ensembl gene identification, was retrieved from Ensembl (Zerbino, et al., 2018), with the help of the biomaRt package (Durinck, et al., 2005).

The enrichment analysis was performed by applying the Fisher's Exact Test. This statistical significance test determines whether two categorical variables are independent, considering the independency as the null hypothesis. It assigns an according p-value (a probability of the null hypothesis to be true) to the analysed data (McDonald, 2014).

In this case, the two variables considered were the annotation (or not) of a GO term to a gene, and the belonging or not of a gene to the set of selected genes. Thus, each sample's set of genes was compared to the rest of genes of the whole organism.

Once the enriched GO terms were obtained, those which resulted significant (with a p-value lower that 0.05, meaning than the probability of the enrichment being due to chance is less than 5%) were taken into account.

Functional enrichment at YMC phase level

To perform the functional enrichment of the genes associated to each enzyme per each one of the YMC phases (OX phase, RB phase and RC phase), the different time points at which the samples were taken had to be considered.

According to the timing of the sampling of the ChIP-Seq data (**Figure 3**), samples 1, 10, 11, 12, 13 and 14 belonged to the RC phase; samples 3 and 4, to the OX phase; and samples 7 and 8, to the RB phase. However, due to their close location to the phase limits, it was not clear to which YMC phase attribute the remaining samples (2, 5, 6 and 9). So, in order to assign them to a concrete phase, their most significant enriched GO terms were compared to those that presented the other samples, and the known characteristics of each phase (Kuang, et al., 2014; Rao & Pellegrini, 2011; Tu, et al., 2007). This way, they were included in the phase which was the most similar to them in terms of GO terms.

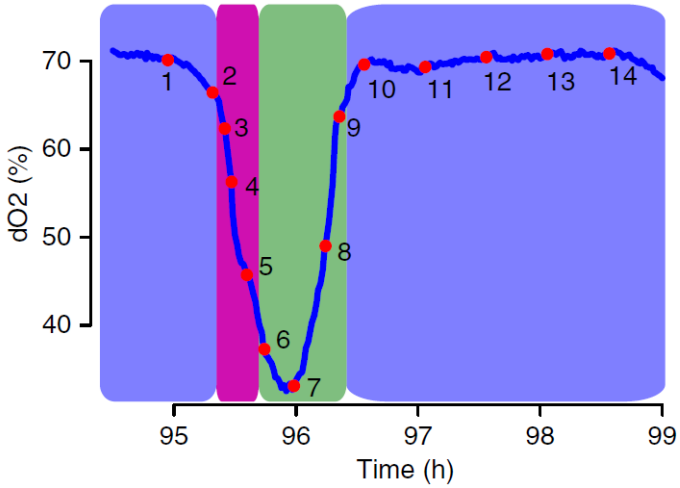

***Figure 3**. Distribution of the 14 samples of enzymes' ChIP-Seq along one YMC cycle, and the corresponding levels of dissolved oxygen (Kuang, et al., 2014).*

After the YMC phase was defined for the conflictive samples, all the filtered genes detected at each time point were used to construct a gene matrix. This matrix comprised all of the gene identification names, and indicated whether a gene had an association or not at a concrete time for each analysed enzyme.

Using this matrix, all the genes that appeared in more than a half of the samples that belonged to the same phase were filtered, and were considered as representative of that phase. Once each YMC phase was assigned its characteristic set of genes, these sets of genes underwent a GO term enrichment analysis against the rest of genes in the organism, by applying again a Fisher's Exact Test, as described previously, with a m

The obtained GO terms were filtered by their p-value (threshold of 0.05) and only significant ones were considered.

Intersection with RNA-Seq and acetylation ChIP-Seq data

Next, the filtered genes obtained from the RGmatch association were intersected with the RNA-Seq and acetylation ChIP-Seq data, already processed by Sánchez Gaya et al. (2018). This was done in order to detect all the genes that are shared among the enzymes and histone acetylations and are relevant to the YMC. RNA-Seq data consisted of a list of DEGs in yeast during the YMC; each gene was associated to a unique phase of the cycle. On the other hand, the acetylation ChIP-Seq data was provided as a matrix (resulting from the association of histone acetylations data with RNA-Seq datacusing MORE linear regression by Sánchez Gaya et al. (2018)) with all the genes associated to any of the two acetylations (H3K9 and H3K18), and a correlation coefficient for the gene-acetylation relation.

First, the genes associated to each chromatin modifying enzyme were intersected with the DEGs that correspond to each phase of the YMC. The DEGs of each enzyme were also intersected among them, for each phase separately, and then functionally enriched against the rest of the genes of the organism.

This association to the DEGs of each phase was also performed for the genes associated to each one of the acetylations; in the case of the acetylation data, only genes with a positive correlation coefficient were considered.

After that, the associated and differentially expressed genes of each enzyme were intersected with the associated and differentially expressed genes of the two histone acetylations, separately for each YMC phase. These intersections were made for each histone acetylation and enzymes combination, and for both acetylation gene sets and each enzyme together. Additionally, the latter were intersected among themselves, for each YMC phase.

Transcription factors enrichment

The gene subsets obtained from the described intersections were associated to different transcription factors that are related to them. Concretely, the subsets that integrated DEGs, one enzyme ChIP-Seq and two acetylation ChIP-Seq per each phase were considered for this association.

The data related to the yeast transcription factors and their associations to yeast genes was retrieved from Yeastract (Teixeira, et al., 2018), a database of transcriptional regulators of *S. cerevisiae*. These data were used to construct an annotation matrix that would be used in the following enrichment.

With the Yeastract data and the integrated gene subsets, Fisher's exact test was applied to these genes to obtain the transcription factors that are enriched among the different existent TF-gene relations. The enrichment was performed per enzyme and per phase separately, against all the DEGs in a particular phase.

Afterwards, the significant TFs (with a p-value < 0.05) were compared among the different phases and enzymes, and the common TFs were defined and represented with Venn diagrams.

Statistical integration of enzymes' and acetylations' ChIP-Seq data

In addition to the RGmatch data analysis, another integration procedure was applied. This time, however, a more quantitative approach was chosen: ChIP-Seq data of the two acetylations was related to data of two of the enzymes (Esa1 and Gcn5), by means of regression. Set1 was not considered in this part of the study, as the effect of the methylation was not expected to be as relevant as the acetylation effect.

To perform the omics data integration, read count matrixes of ChIP-Seq data had to be constructed and properly arranged. Also, a peak-to-gene association matrix had to be created for all the enzyme ChIP-Seq data.

Read count matrixes construction

A preliminary step for building of the statistical models was the obtention of read count matrixes for each set of omics data. While the H3K9 ChIP-Seq and H3K18 ChIP-Seq data were already processed (Sánchez Gaya, et al., 2018) and provided as normalized matrixes, the ones for Esa1 and Gcn5 ChIP-Seq had to be constructed.

In order create the matrixes, the different yeast genomic regions that featured peaks during the peak calling procedure had to be defined. This was achieved with the `makeGRangesFromDataFrame` tool, part of the GenomicRanges package for R (Lawrence, et al., 2013), which generated a saf file for each set of narrowPeak files obtained from MACS2 (one per each enzyme). These files contained consensus genomic regions of all the time point, considering close peaks as aligned reads enrichments of the same sites. The narrowPeak files' data had to be previously combined together in a data frame.

To count the ChIP-Seq reads to the defined genomic regions, the Subread package (Liao, Smyth, & Shi, 2019), and concretely, the `featureCounts` tool, was used. The read counts were obtained by the software from the set of alignment (sam) files of an enzyme and the correspondent saf file. These data were then arranged into a matrix, creating a read count matrix for each chromatin modifying enzyme's ChIP-Seq.

Data normalization and correction

After the construction of the read count matrixes for Esa1 and Gcn5 ChIP-Seq, their count values had to be corrected to remove potential technical biases.

For this purpose, Esa1 and Gcn5 read counts were normalized by RPKM (with `rpkm`, part of the NOISeq package (Tarazona, et al., 2015; Tarazona, García Alcalde, Dopazo, Ferrer, & Conesa, 2011)). This normalization methods corrects the difference in sequencing depth (total amount of reads) of each sample, and also the different length of the considered region, since longer peaks tend to accumulate more reads, which does not imply a stronger signal, but it is an unwanted effect of the sequencing technique. A logarithmic transformation was applied to the values, to obtain a more symmetrical distribution of peak counts which worked better in statistical models, and then they were centred, following the same idea as in the acetylations' read counts from Sánchez Gaya et al. (2018) study.

To explore the read count data of the enzymes and track their correct transformation, principal component analysis (PCA) was applied to the matrixes, before and after the correction. The PCA is a type of statistical analysis that diminishes the dimensionality of data, by detecting the directions (principal components, or PC) in which the variation of the data is maximal. These directions are a linear combination of the original variables of the samples; the different PCs are uncorrelated among them. This way, a sample can be represented with relatively few numbers, and plotting the PCs of the samples results in an easy way to identify similarities and clusters among the dataset. This analysis is frequently used for genome-wide expression studies (Ringnér, 2008).

Time point arrangement

To build the linear regression models, the read count matrixes for Esa1 and Gcn5 had to be made comparable to the already processed H3K9 and H3K18 count matrixes. This means that observations had to match between the predictor and the response omics in order to relate both of them. Direct correlation was not possible, as the enzymes samples' values corresponded to 14 time points, while the acetylations' values were made for 15 time points (16 originally (Kuang, et al., 2014), but time points 13 and 14 were averaged into one in Sánchez Gaya et al. (2018) study, and were provided as such), without exact time equivalence among them (**Figure 4**).

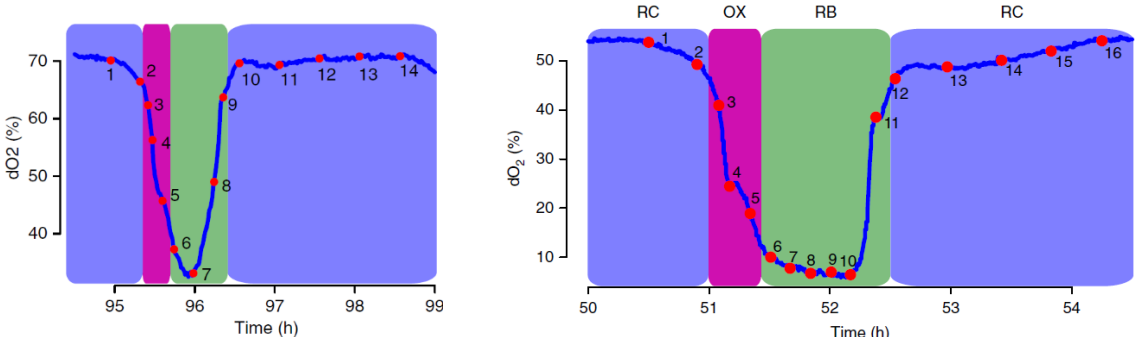

***Figure 4**. Sampling time points during the YMC of the chromatin modifying enzymes ChIP-Seq (on the left) and of the histone acetylations ChIP-Seq (on the right), and the corresponding levels of dissolved oxygen measured at each time (Kuang, et al., 2014).*

To make the data comparable, some time points' values were averaged into one (11 and 12 from the enzyme ChIP-Seq matrixes), and some of them (7 and 9 from the acetylation ChIP-Seq matrixes) were excluded from the study. In the end, 13 time points were considered for enzyme and acetylation read counts data, as it is reflected in **Table 1**.

***Table 1**. Correspondence of the new time points considered for the linear regression models and the original sampling time pints of histone acetylation and enzyme ChIP-Seq experiments. The dash ("-") between two values indicates the time point which values were averaged; the brackets ("( )") indicate the points which values had been already averaged by Sánchez Gaya et al. (2018).*

Considered time point	1	2	3	4	5	6	7	8	9	10	11	12	13
Acetylation ChIP-Seq time point	1	2	3	4	5	6	8	10	11	12	(13-14)	15	16
Enzyme ChIP-Seq time point	1	2	3	4	5	6	7	8	9	10	11-12	13	14

Integration of acetylation and enzyme ChIP-Seq data by regression models

In statistics, regression models are a way of estimating the relationship that exists between a variable of interest (response variable) and other variables (predictors or explanatory variables). Regression models provide a mathematical equation that relates the response and the predictors. The statistical significance of the coefficients of the predictors in such model will indicate whether the predictor has an important effect on the response. In the field of omics integration, the regression equations are used to explain the relations and interactions among different analysed elements of a biological system (Zeng & Lumley, 2018).

To construct regression models and evaluate the correlation of the acetylation of a gene with the attachment of a chromatin modifying enzyme to that gene, MORE software was used (Tarazona, Tomás Riquelme, Martínez Mira, Clemente Císcar, & Conesa, 2018). This tool allows the modelling and integration of different omics data in order to detect the potential regulators of the system, by applying generalized linear models (GLM).

To use MORE's `GetGLM` function (the one that constructs the regression models), an association list was made, which related each peak of the enzymes' ChIP-Seq data to a gene from the histone acetylations' ChIP-Seq data. The peak-to-gene correspondence was previously obtained with RGmatch (Furió Tarí, Conesa, & Tarazona, 2016) and the genomic region (saf) files of each enzyme, using the same parameters as described above (see **Analysis of RGmatch results with R**). This relationship postulates the potential regulators for which the GLM should be tested, hence these potential regulators will be the explanatory variables in the regression model.

In the `GetGLM` function, several input data sets had to be given: the values for the response variables for the models (`GeneExpression`), which in this case was the read count matrix of one acetylation ChIP-Seq; the list of peak-to-gene associations (`associations`) so that the potential regulators (explanatory variables) are identified for each possible response variable (gene); and the read count matrixes of both chromatin modifying enzymes' ChIP-Seq, which are the values of the potential regulators (`data.omics`). The count matrixes were previously filtered, and contained only the genes that were common between each histone acetylation and enzyme. Since the histone acetylation data had been log-transformed, a normal distribution for these data was assumed, so a particular case of GLM was applied: the linear regression model. Other important input parameters were related to the variable selection procedures implemented in MORE: ElasticNet and stepwise regression, which are applied in order to identify the most relevant regulators.

The MORE method was applied twice, for H3K9 and HK18 data separately. Lists with all the genes and regulators with a significant effect on them were obtained, for each histone acetylation and chromatin modifying enzyme separately. The genes were also divided depending on the sign of the correlation with the acetylation values.

The obtained gene sets were intersected with RNA-Seq data, in order to separate them according to their YMC phase. Then, they were enriched in their GO terms, against the yeast DEGs of each phase of the YMC.

Results and discussion

Chromatin modifying enzymes' ChIP-Seq data preprocessing

Quality check and trimming of ChIP-Seq reads

At the beginning of the study, the raw data files were analysed for their quality with FastQC (Andrews, 2010). Most of the fastq files of each enzyme's ChIP-Seq did not show a satisfactory quality, especially because many adapter sequences were found among the reads (**Figure 5**), and because of an important bias in the per base sequence content. In order to solve this problem, the reads were trimmed with Trimmomatic (Bolger, Lohse, & Usadel, 2014). Indeed, after the trimming, the quality (that was analysed once again) of the reads improved considerably **(Figure 6**), making it possible to proceed to the next step of the study: sequence alignment.

Overrepresented sequences

Sequence	Count	Percentage	Possible Source
GATCGGAAGAGCACACGTCTGAACTCCAGTCACTAGCTTATCTCGTATGC	1584040	24.064351068140045	TruSeq Adapter, Index 10 (100% over 50bp)
AATGATACGGCGACCACCGAGATCTACACTCTTTCCCTACACGACGCTCT	201334	3.058617243221704	Illumina Single End PCR Primer 1 (100% over 50bp)
AAGCAGAAGACGGCATACGAGATAAGCTAGTGACTGGAGTTCAGACGTGT	141336	2.147142194999269	TruSeq Adapter, Index 10 (100% over 50bp)
AGATCGGAAGAGCACACGTCTGAACTCCAGTCACTAGCTTATCTCGTATG	130408	1.9811266723656011	TruSeq Adapter, Index 10 (100% over 49bp)

Figure 5. *Example of some overrepresented adapter sequences, present in the fastq file of Gcn5 ChIP-Seq at time 14.*

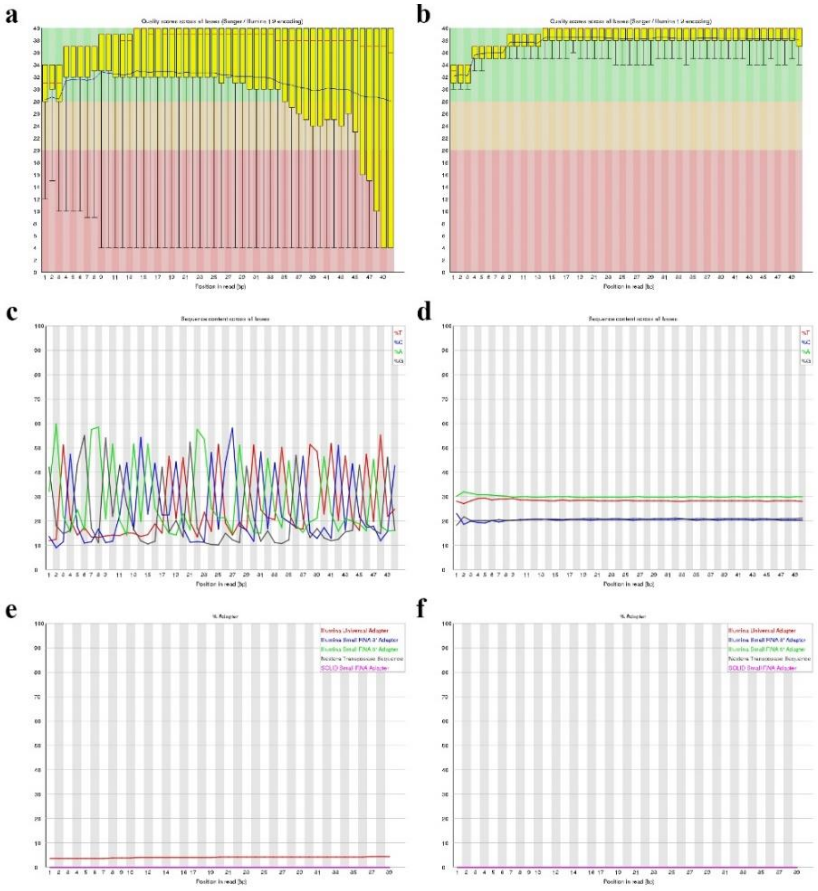

Figure 6. *Comparison of the quality of the fastq file of Gcn5 ChIP-Seq at time 14 before (*left*) and after (*right*) the trimming. Charts correspond to:* ***a*** *and* ***b****, per base quality content (quality score* vs. *position in the reads, in bp);* ***c*** *and* ***d****, per base sequence content (sequence content* vs. *position in the reads, in bp); and* ***e*** *and* ***f****, adapter content (% of adapter sequences score* vs. *position in the reads, in bp).*

Peak calling

Sequence alignment and quality check

For the sequence alignment, the trimmed reads for each enzyme were mapped onto the reference genome of *S. cerevisiae* (release 91) with Bowtie 2 software (Langmead & Salzberg, 2012). As a result, 42 sam files were obtained. Additionally, 16 sam files for the H3 histone were obtained, to be further used as the control sample in the peak calling, as the original study (Kuang *et al.*, 2014) did not provide a control sample with the same time scale (14 time points).

The quality of the mapping was checked with Qualimap 2 (Okonechnikov, Conesa, & García-Alcalde, 2016). As its *Multi-sample BAM QC* analysis was used, all 14 samples of each enzyme were processed together.

A visual representation of the quality of the mapping is shown in **Figure 7**. It can be seen that, in the alignment of Set1 reads, one sample's behaviour (corresponding to the 1st time point) differs from the rest of the samples: it presents a higher coverage, and a different GC content distribution. This is due to the fact that the quantity of available reads in this concrete sample was also much higher; this can be explained with a possible experimental error when the library was prepared for the sequencing.

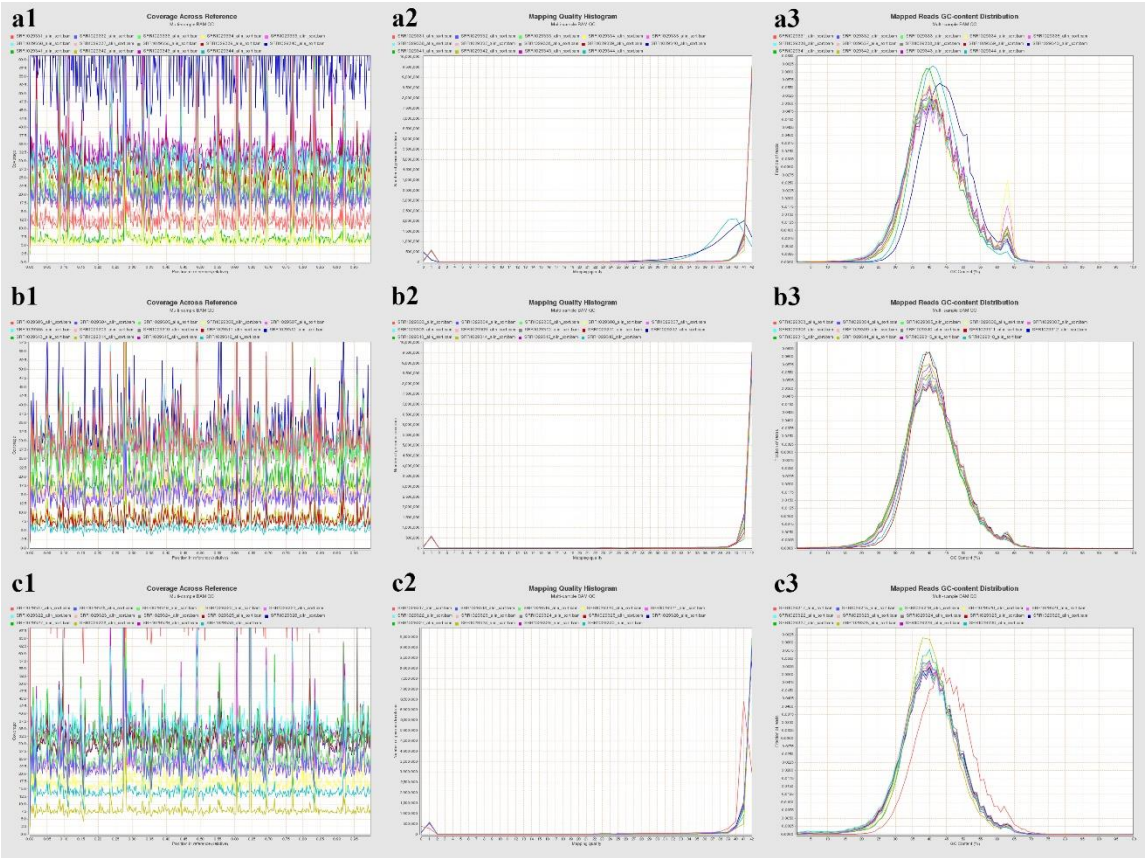

Figure 7*. Coverage across the reference genome (coverage, in X fold,* vs. *the relative position in the reference genome, in 0 to 1 range), mapping quality histograms (number of genomic locations* vs. *mapping quality, in Phred value) and GC content distribution (fraction of reads* vs. *GC content, in %) of mapped reads (from left to right) of the sequence alignments, obtained with Qualimap 2. Each group of three charts corresponds, from top to bottom, to Esa1 (a1, a2 and a3), Gcn5(b1, b2 and b3) and Set1 (c1, c2 and c3) ChIP-Seq reads' alignments.*

Table 2 shows summary statistics of the alignment. It can be observed that, in general, a high percentage of the reads was mapped (around 82%). The GC content of the reads corresponds to

an expected value as for the studied organism, which is be comprised between 39-42% (Bradnam, Seoighe, Sharp, & Wolfe, 1999). The values of coverage are proper as for a ChIP-Seq experiment (genohub.com, n.d.).

***Table 2**. Summary statistics of the alignment quality of Esa1, Gcn5 and Set1 ChIP-Seq reads, with the sum of the reads from the 14 samples of each enzyme, and the mean coverage, GC content and quality of their mapping.*

	Esa1	Gcn5	Set1
Total number of reads	107,221,823	102,538,663	147,325,458
Total number of mapped reads	88,825,853	84,041,836	121,546,944
% of mapped reads	82.84	81.96	82.50
Mean samples coverage (X)	28.53	24.33	35.17
Mean samples GC-content (%)	41.75	40.51	40.15
Mean samples mapping quality (Phred)	37.14	37.81	37.74

Peak calling

Next, the peak calling was performed. MACS2 tool (Zhang, et al., 2008) allowed to execute this task, detecting peaks of reads in each sample compared to the normal H3 histone (previously fusing the 16 sam files into one with SAMtools (Li, et al., 2009)).

The analysis gave an average quantity of ~2352 detected peaks, for each time point, for Esa1; ~1971 for Gcn5; and ~1764 for Set1. However, time 1 sample of Set1 dataset presented much more peaks than the rest (3755, in contrast to the 1192-1986 range of the remaining samples); this is explained by the fact that this sample had much more reads, as mentioned in the alignment analysis, were it showed a higher coverage across the reference genome,.

Peak-to-gene association

Statistical summary

The obtained peaks were associated to the different yeast genes by proximity, by using the RGmatch tool (Furió Tarí, Conesa, & Tarazona, 2016), in order to further analyse the genes potentially affected by the histone modifiers. This supposed the generation of 14 txt files per each enzyme, which indicated in detail the relation of each peak with a gene for each time point, to which genetic feature it is associated, its distance to the TSS, etc. .

The obtained data were processed in R Studio (RStudio Team, 2016), in order to statistically analyse the associations. The summary statistics of the peak-gene associations corresponding to each enzyme's ChIP-Seq can be seen in **Table 3**.

***Table 3**. Statistical summary of the peak-to-gene association for each chromatin modifying enzyme. The shown values are an average among the 14 time points.*

	Esa1	Gcn5	Set1
Detected peaks	2352.429	1971.000	1764.000
Associated peaks	2349.714	1968.929	1760.857
Number of associations	3782.143	2976.429	3114.214
% of peaks with an association to a gene	99.882	99.892	99.819
% of peaks with more than one association	55.092	46.111	70.710

% of associated peaks with association to the TSS, promoter, first exon or gene body	85.357	81.435	88.805

It was observed that most of the peaks were associated with one of these elements: the TSS, the promoter, the first exon or the gene body. The first exon and the promoter comprise an especially significant number of peaks (**Figure 8**). The high association to a regulatory element such as a promoter suggests an important role of these enzymes in the regulation of the correspondent genes. The association to the TSS, another cis-regulatory element, is much lower, probably because these sequences are shorter, which would difficult the direct association to them.

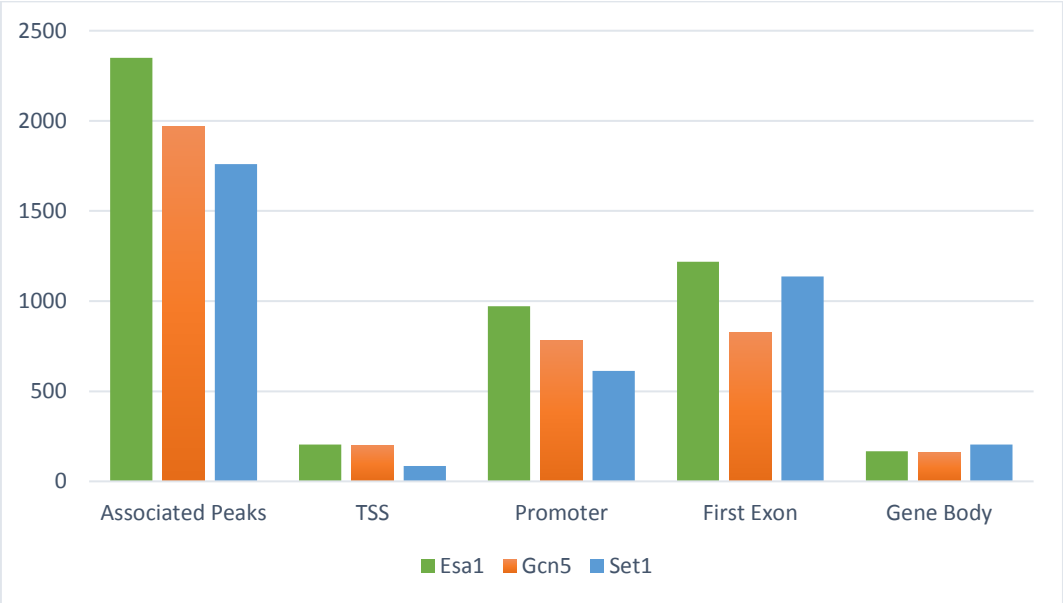

***Figure 8**. Total associated peaks to Esa1, Gcn5 and Set1, and the peaks associated to different genomic features (TSS, promoter, first exon and gene body).*

An example of a more general view of the distribution of the peaks is shown in **Figure 9**. All samples, for all the enzymes, featured a similar distribution. As expected, features like promoter or first exon are more frequent around the TSS, while gene body or upstream elements are more frequent further.

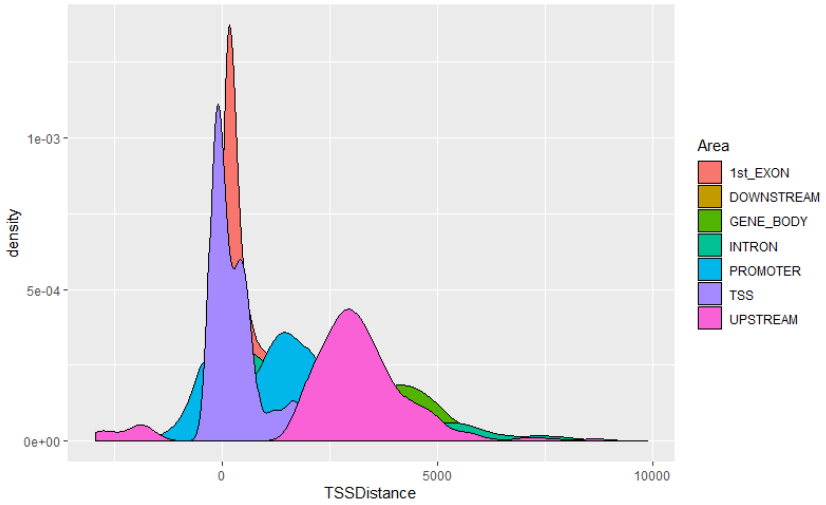

***Figure 9**. Density plot of Esa1's ChIP-Seq at time 1, showing the distribution of the detected peaks over the distance to the TSS, classified by the different genetic features that they are associated to.*

Due to the interest of this study in the significance of the chromatin modifying enzymes in the regulation of gene expression during the YMC, only the genes with an association to their regulatory elements (TSS or promoter) were considered in the further steps of the analysis, except the ones resulting from the usage of a read count matrix.

Assignation of time points to each YMC phase

After the statistical analysis, the different samples were enriched in their GO terms (The Gene Ontology Consortium, 2019), by applying the Fisher's Exact Test to each set of genes. The test was performed for each time point separately. The obtained, enriched GO terms in each sample were then functionally compared.

The similarities between the individual significative GO terms of each time point in each enzyme allowed to assign the previously mentioned conflictive samples to a proper YMC phase; the assignment, however, differs from the one established in Kuang et al. (2014) study. This way, the three phases of the cycle were further considered as follows:

-OX phase: samples 3, 4 and 5.

-RB phase: samples 6, 7 and 8.

-RC phase: samples 1, 2, 9, 10, 11, 12, 13 and 14.

As the representative GO terms in the samples of the same phase are very similar, the description of the functional enrichment will be performed directly per each phase and enzyme jointly in the next section.

Functional enrichment of the YMC phases

To perform a functional enrichment on the sets of genes of the different phases, only the genes that appeared in more than a half of the corresponding samples (at least in 2 samples in the case of OX and RB phases, and at least in 5 samples in case of RC phase) were selected as representative. For this purpose, a time matrix with all the detected genes (filtered by TSS or promoter association) and their presence or absence at a certain timepoint was constructed for each enzyme. This way, not all the initially detected genes were considered for the enrichment (**Table 4**).

***Table 4**. Genes associated to peaks with RGmatch by their promoter or TSS, and genes considered representative for each enzyme and phase of the YMC. Note that some genes were considered as representative in more than one phase.*

	Esa1	**Gcn5**	**Set1**
Associated genes	2411	2287	2171
Representative genes in OX phase	904	921	574
Representative genes in RB phase	955	811	455
Representative genes in RC phase	959	721	419

As the **Table 4** reflects, less than 40% of the total associated genes were considered in each phase, in the case of Esa1, and 31-40% in case of Gcn5; in comparison, Set1 values are lower: only ~19-26% of the genes were assigned to each phase.

Apparently, the Set1 methyltransferase seems to be involved in the regulation of much less genes than the other two enzymes. It cannot be determined, however, whether this is due to a particularity of the Set1 enzyme or, by contrary, histone methylation is not as much involved in the regulation of the YMC as the acetylation, which has been already observed in other studies (Sánchez Gaya, et al., 2018).

Regarding the acetyltransferases, Esa1 is associated to a similar amount of genes in each phase, but displays less genes for the OX phase, in contrast to the other two enzymes. On the other hand, Gcn5 presents noticeably lesser quantities of associated genes for the RB and RC phases. This can be explained by the "preference" the Gcn5 acetyltransferase may have for the genes corresponding the OX phase, implying its importance in the regulation of the oxidative phase, as has been already noticed in other studies (Cai, Sutter, Li, & Tu, 2011).

As for the functional enrichment, the acetyltransferases Esa1 and Gcn5 displayed very similar results, while the methyltransferase Set1's genes seemed to be enriched in other functionalities.

Curiously, the most significative GO terms for the two acetyltransferases (Esa1 and Gcn5) were very similar among both them and the cycle's phases: *Cytoplasmic translation*, *Structural constituent of ribosome*, *rRNA export from nucleus*, *Fructose transmembrane transporter activity*, *Mannose transmembrane transporter activity*, *L-phenylalanine transmembrane transporter activity*, *Glucose transmembrane transporter activity* or *Gluconeogenesis* are some examples.

Many of these GO terms seem related to the OX phase of the YMC (the ones related to translation, sugar transport or ribosomes), but are not usually representative of the RB and RC phases. A possible explanation to this observation is that Esa1 and Gcn5 have a similar, preferent attachment to regulation sites of typical OX phase genes, just as described for the latter enzyme (Cai, Sutter, Li, & Tu, 2011). In addition, the genes related to ribosomal RNA (rDNA) are found in large clusters in yeast (Saka, Takahashi, Sasaki, & Kobayashi, 2016), which increases the probability of association of a peak to one of them. However, this does not exclude their interaction with other genomic regions that are relevant for the other two phases.

On the other hand, Set1 displayed different GO terms for its associated genes than the acetyltransferases. Still, some terms appeared in all phases, like the *Nuclear nucleosome*, *Glutaminase activity*, *IMP cyclohydrolase activity*, *Pyridoxine metabolic process* or *Negative regulation of transcription, DNA-templated*, and several were shared by two phases.

In the Ox phase, the most significant GO terms that appeared are *L-iditol 2-dehydrogenase activity*, *Methylated histone binding*, *Asparagine-tRNA ligase activity*, *Biotin biosynthetic process* and *Methylated histone binding*, among others. In the RB phase, the obtained GO terms were related to histone modifications (*Histone methylation*, *Histone deacetylation*), thiamine (*Thiamine biosynthetic process*), *Asi complex* or TFs (*Transcription factor catabolic process*). The RC phase featured several GO terms related to response to stress (*Positive regulation of transcription from RNA polymerase II promoter in response to freezing, […] to cold, […] to nitrosative stress, […] to hydrogen peroxide*, *Response to starvation*…), epigenetic modification (*Methylation*, *Histone acetylation*), transport of different substances (*Sorbitol transport*, *Mannitol transport*, *Spermidine transport*, *Oligopeptide transport*…) or thiamine (*Thiamine biosynthetic process*) can be found.

Set1's functional enrichment yields terms that are, just as in the case of acetyltransferases, not so representative of the YMC. The GO terms, however, seem related to its own function, methylation, in all three cycle phases, and to the regulation of transcription (especially in stress conditions, typical of the RC phase).

Nevertheless, the results of this enrichment must be considered carefully. The genes were associated to each chromatin modifying enzyme by the proximity of the gene TSS or promoter to the peak, but this is no guarantee that the attachment is actually affecting all the genes, as the detected interaction may not necessarily imply a chromatin modification, and therefore, no regulation would be occurring at these sites. Furthermore, the associated genes may even not be expressed. Because of this, these results must be contrasted with other data, and considered, as for now, as genes that are *potentially* regulated by Esa1, Gcn5 or Set1.

Intersection with acetylation and gene expression data

After the functional enrichment, the set associated and filtered genes form each enzyme's ChIP-Seq were combined with the data of the other three omics experiments: RNA-Seq, H3K9 ChIP-Seq and H3K18 ChIP-Seq. Note that the genes considered this time were taken all together, without considering their representation in a YMC phase.

First, the associated genes from the enzymes' experiments (see **Table 4**) were intersected with the sets of differentially expressed genes (DEGs) that were obtained by the RNA-Seq experiment (Table 5). The intersection was made for each subset of DEGs separately, corresponding to each cycle phase **Table 6** shows the results of these intersection, as well as the total numbers of DEGs during the YMC.

***Table 5**. Differentially expressed genes (DEGs) during the YMC, in each phase, obtained from the RNA-Seq experiment.*

	Total	**OX phase**	**RB phase**	**RC phase**
DEGs	2552	1428	426	698

***Table 6**. Genes resulting from the intersection of the RNA-Seq data (differentially expressed genes) and the three enzymes' ChIP-Seq data (genes associated to an enzyme by the promotor or the TSS).*

	Esa1	**Gcn5**	**Set1**
Associated genes	2411	2287	2171
DEGs in OX	447	452	406
DEGs in RB	135	121	133
DEGs in RC	328	346	284

When the enzymes' associated DEGs were intersected among themselves, it was seen that most of the genes were common among the three chromatin modifiers (**Figure 10**).

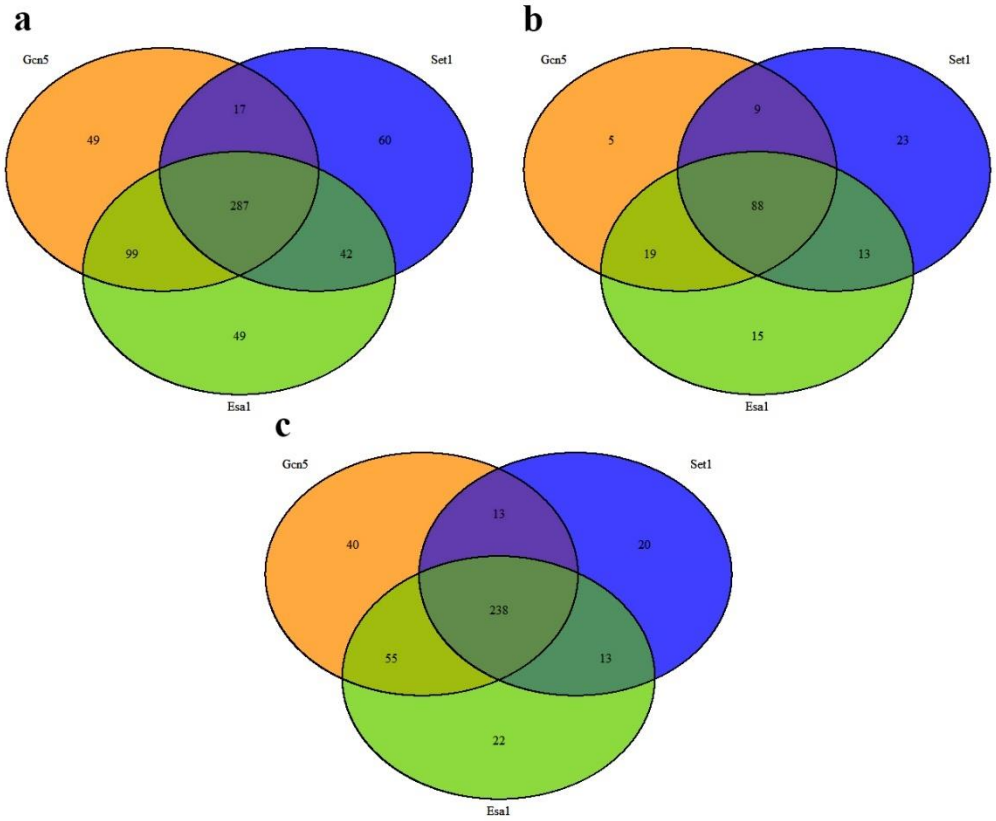

***Figure 10**. Intersection of the differentially expressed genes associated to each chromatin modifying enzyme in the OX (**a**), RB (**b**) and RC (**c**) phases of the cycle.*

The functional enrichment of the common DEGs of the three enzymes resulted in expected GO terms for each YMC phase. The common OX phase genes were enriched in GO terms related to the ribosomes and their assembly (*Ribosome*, *Cytosolic large ribosomal subunit*, *Cytosolic small ribosomal subunit*, *Ribosomal large subunit assembly*, *Ribosome biogenesis*…), translation (*Cytoplasmic translation*, *Translation*, *Maintenance of translational fidelity*…), RNA (*RNA binding*, *mRNA binding*, *pre-mRNA 5'-splice site binding*…) and biosynthesis of lysine (*Lysine biosynthetic process via aminoadipic acid*), among the most significant. RB phase genes featured as the most significant the terms related to the mitochondria (*Mitochondrial large ribosomal subunit*, *Mitochondrial translation*, *Mitochondrial electron transport, ubiquinol to cytochrome c*…), respiration (*Aerobic respiration*) and sporulation (*Sexual sporulation resulting in formation of a cellular spore*). In the RC phase, the common genes were enriched in GO terms regarding different metabolic processes (*Fatty acid metabolic process*, *Carbohydrate metabolic process*, *Fatty acid beta-oxidation*, *Carnitine metabolic process*…), glutathione (*Glutathione metabolic process*, *Glutathione transferase activity*…), peroxisomes (*Peroxisome*), stress responses (*Cellular response to oxidative stress*, *Cellular response to desiccation*…).

The facts that most of the associated DEGs of these subsets are common for the three enzymes, and that the enriched GO terms are representative of each phase (ribosomes and translation related terms in OX phase, mitochondria terms in RB phase, and fatty acid and carbohydrate metabolism and stress responses in RC phase) according to bibliography (Kuang, et al., 2014; Rao & Pellegrini, 2011; Tu, et al., 2007), suggest that all three enzymes could regulate very similar subsets of genes, probably the characteristic ones of each cycle's phase.

This might seem contrary to the functional enrichment analysis results that were discussed in the previous section. However, as was already stressed, the genes considered previously were taken into account only because of proximal association to a gene TSS or promoter. Now, these genes have been filtered by their expression: only those that are differentially expressed, and contribute to the metabolism of the cell during the YMC, are considered.

The subsets of genes coming from the two acetylation ChIP-Seq experiments were also intersected with the DEGs of each YMC phase coming from the RNA-Seq (**Table 7**). As the former genes had either a positive or a negative correlation with the corresponding acetylation, only those with a positive correlation coefficient were considered.

***Table 7**. Differentially expressed genes (DEGs) of each one of the analysed histone acetylations (H3K9 and H3K18) with a positive correlation, divided according the three YMC phases.*

	H3K9	H3K18
Genes with positive correlation	780	815
DEGs in OX	531	452
DEGs in RB	124	103
DEGs in RC	125	260

Next, the DEGs from the enzymes' ChIP-Seq were intersected with the DEGs from the acetylations' ChIP-Seq, differentiating the YMC phases. The intersections were made for each enzyme and both acetylations individually (**Table 8**) and together (

Table 9).

***Table 8**. Common genes from intersection of three experiments (RNA-Seq, a histone acetylation ChIP-Seq and a chromatin modifying enzyme ChIP-Seq), por each enzyme, histone acetylation and YMC phase.*

	H3K9			H3K18		
	Esa1	**Gcn5**	**Set1**	**Esa1**	**Gcn5**	**Set1**
Common genes in OX phase	156	167	122	154	159	115
Common genes in RB phase	38	32	32	36	35	44
Common genes in RC phase	71	71	52	123	135	106

***Table 9**. Common genes resulting from the intersection of four experiments (RNA-Seq, 2 acetylation ChIP-Seq and a chromatin modifying enzyme ChIP-Seq), por each enzyme and YMC phase.*

	Esa1	Gcn5	Set1
Common genes in OX phase	67	72	46
Common genes in RB phase	7	5	6
Common genes in RC phase	25	25	20

As expected, OX phase featured the largest number of common genes, as the initial gene set sizes were also larger (see **Figure 11**). Nevertheless, despite having associated a similar number of DEGs in OX phase in comparison to other enzymes, Set1 showed noticeably less genes in common with H3K9 and H3K18, probably because it is most likely for a histone acetylation to have commonly regulated genes with an acetyltransferase (like Esa1 and Gcn5) than with a methyltransferase (like Set1).

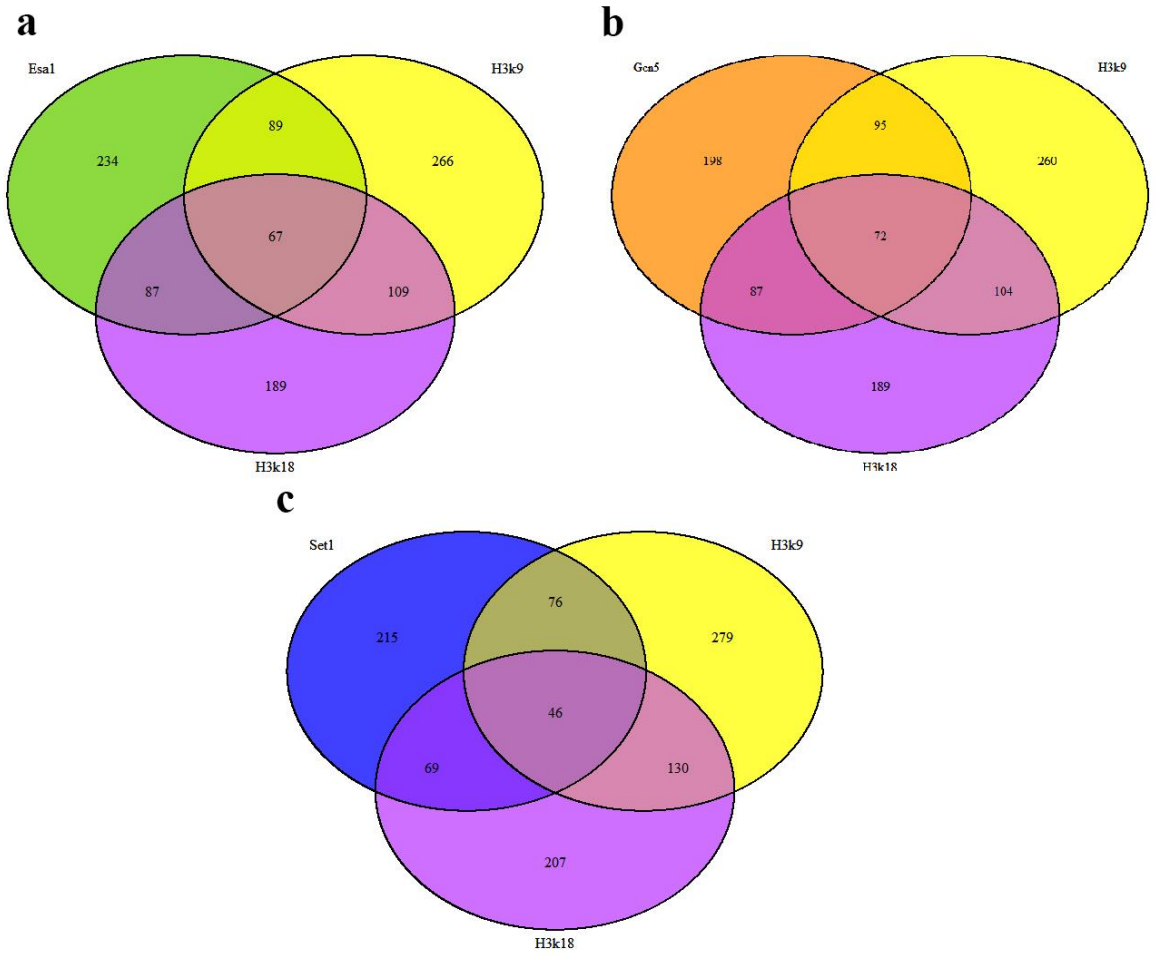

Figure 11. *Intersections of DEGs of the enzymes' and of acetylations' ChIP-Seq experiments, in OX phase.*

Interestingly, in the OX phase, the DEGs related to the three chromatin modifying enzymes display more common genes with H3K9 histone acetylation, while in the RB and RC phases, it was observed that there are usually more common genes with H3K18 acetylation. This could suggest a slight predisposition to certain acetylation sites of the histone modifiers depending on the YMC phase.

Finally, the common genes of each phase resulting from the intersection of four experiments (RNA-Seq, H3K9 ChIP-Seq, H3K18 ChIP-Seq and one of the enzymes' ChIP-seq) were intersected among the three chromatin modifying enzymes (**Figure 12**).

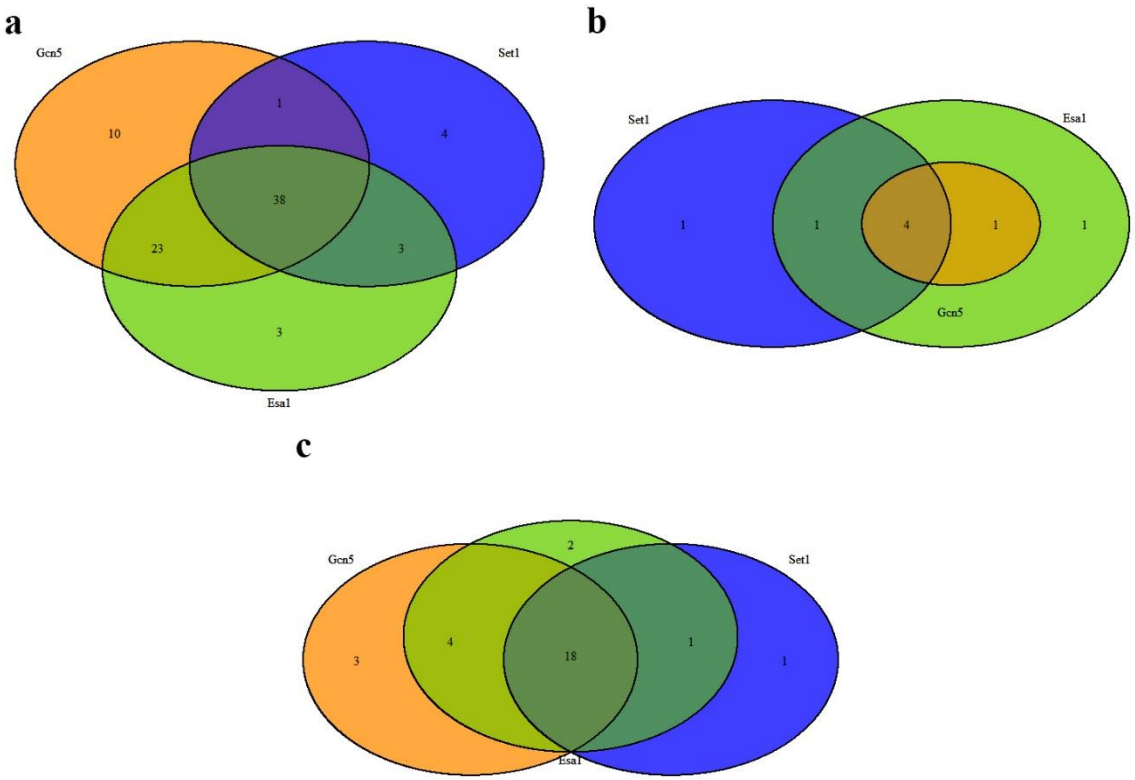

***Figure 12**. Intersections of the common genes of the four gene expression experiments (RNA-Seq, H3K9 ChIP-Seq, H3K18 ChIP-Seq and one of the enzymes' ChIP-seq) among the different chromatin modifying enzymes, in OX phase (**a**), RB phase (**b**) and RC phase (**c**).*

As **Figure 12** reflects, many of the common genes for all four gene expression experiments are shared by all three chromatin modifying enzymes, although Esa1 and Gcn5 have more elements in common that with Set1, probably because of the similar functions that they perform in the cell. As previously, OX phase features the largest number of common genes, while RB phase shows very few of them.

Enrichment with transcription factors

The common associated differentially expressed genes among each enzyme and the two histone acetylations of each YMC phase were selected to perform a transcriptional factor (TF) enrichment. The TF enrichment was performed against the set of DEGs of the corresponding phases of the cycle. This way, the transcription factors that appeared as related to the genes in significative quantities were obtained for each enzyme-acetylations DEGs set and for each YMC phase. The numerical results of the enrichments can be seen in **Table 10**.

***Table 10**. Significantly enriched TFs that associate to the common DEGs of the enzyme-acetylations intersections, according to the associated chromatin modifying enzyme and phase.*

	Esa1	**Gcn5**	**Set1**
TFs in OX	88	82	74
TFs in RB	9	7	8
TFs in RC	30	42	4

As it was expected, much more TFs are associated to the OX phase, as this phase has more genes associated to it. However, it stands out that the TFs associated to Set1 gene subset in the RC phase are very few in comparison to Esa1 and Gcn5.

It has also been observed that, in each phase, Esa1 and Gcn5 share many associated TFs (as they initially shared many associated genes), while Set1 contains different ones (**Figure 13**). The Set1 associated TFs in RC phase do not coincide with any TF of the acetyltransferases in the same phase.

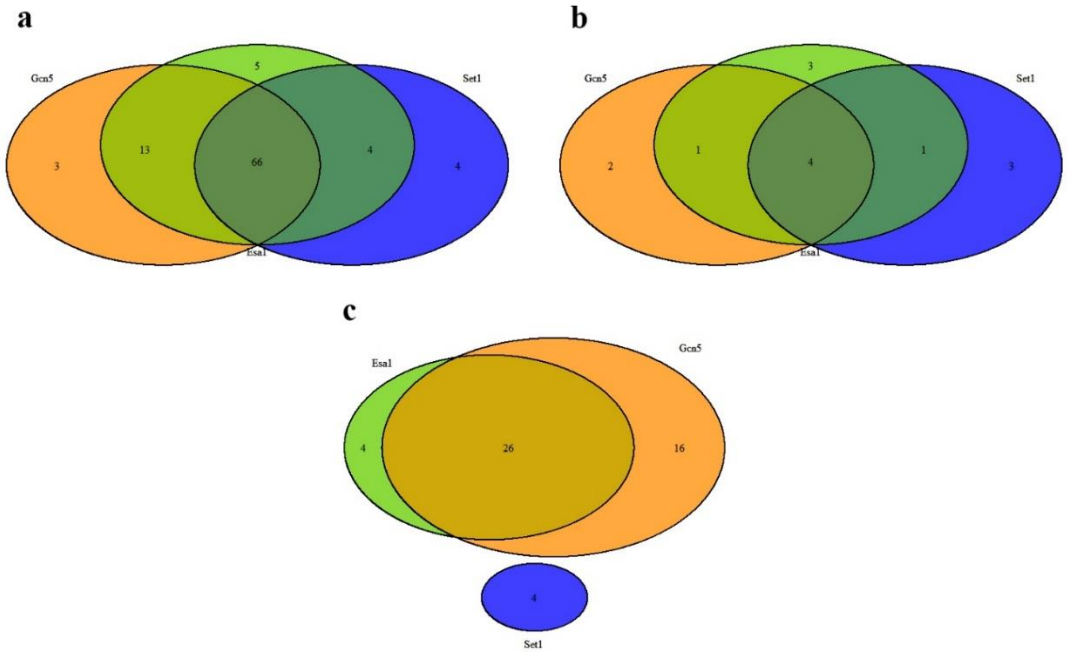

***Figure 13**. Common enriched transcription factors among all the DEG-enzyme-2 acetylations gene subsets, in the OX phase (**a**), RB phase (**b**) and RC phase (**c**) of the YMC.*

Some noticeable transcription factors that were found among the common ones, in OX phase, include IFH1, FHL1 and SFP1, all three involved in the transcription of ribosomal protein genes (and the last one, also with mitotic cell cycle); HFI1 and SPT20, related to the SAGA complex; factors related to SWI/SNF chromatin remodelling complexes, like TAF14 (also involved in involved in RNA polymerase II transcription initiation) and SWI5 (also related to genes expressed at the M/G1 phase boundary of the cell cycle); GCN4 and ARO80, related to amino acid biosynthetic genes and aromatic amino acid catabolic genes, respectively; and PIP2 (binds promoters of genes involved in beta-oxidation of fatty acids, peroxisome organization and biogenesis, activating transcription in the presence of oleate), in OX phase, and OPI1 (involved in telomere maintenance and mitochondrial metabolism) in RB phase (Cherry, et al., 2012).

Common TFs between Esa1 and Gcn5 in RC phase include GCN4, HSF1 and SWI5, which already appeared in OX phase. In case of Set1, the TFs that draw attention in the RC phase are NNF2 (interacts with a subunit of RNA polymerases I, II, and III) and TAF14 (Cherry, et al., 2012).

The fact that these transcription factors are involved in several processes that are characteristic of YMC, especially those of the OX, suggests that they may have an important role in the regulation of the cycle. This is supported by other studies, were many of these TFs (SFP1, HFI1, PIP2, GCN4, ARO80) were also identified as significant during the YMC (Rao & Pellegrini, 2011; Sánchez Gaya, et al., 2018).

However, the limitations of this analysis must not be forgotten: this enrichment is based on an association of transcription factors to certain genes according the bibliography, but it is not a proof

of their regulation of the YMC. Instead, these TFs are potential targets of being regulators of the cycle, and should be considered in further, more specific studies of their involvement in the cycle.

Statistical integration of enzymes' and acetylations' ChIP-Seq data

Read count matrixes construction and normalization

To study the interaction of the chromatin modifying enzymes in a more quantitative manner, read count matrixes were constructed from the aligned reads data, considering a consensus set of peaks for the 14 samples of each enzyme. This way, the matrixes related the quantity of reads at each consensus region with the time point of the YMC at which the interaction occurred.

The count matrixes, as well as the integration by linear regression (see **Statistical integration of enzymes' and acetylations' ChIP-Seq data**), were performed for Esa1 and Gcn5 enzymes, but not for Set1. The reason behind this decision is the function: while Gcn5 and Esa1 are acetyltransferases, Set1 is a methyltransferase, and the posterior multi-omics integration will be performed with histone acetylation data. Additionally, it has been seen in other studies (Sánchez Gaya, et al., 2018) that the impact of methylation it not as strong as the impact of acetylation. Due to these reasons, the decision of analysing only the enzymes related to the acetylation was made.

Once the read count matrixes were constructed, the data they contained were made comparable with the values from histone acetylation ChIP-Seq read counts. With this aim, the processing described in Sánchez Gaya et al. (2018) was performed: first, the read count values were normalized to RPKM; then, a logarithmic transformation was applied; and finally, the data were centred.

To confirm that the applied correction was done properly, PCA were performed as an exploratory analysis, as shown in **Figure 14**. It can be seen that PC1 and PC2 cluster the reads properly, according to their phases, before and after the correction.

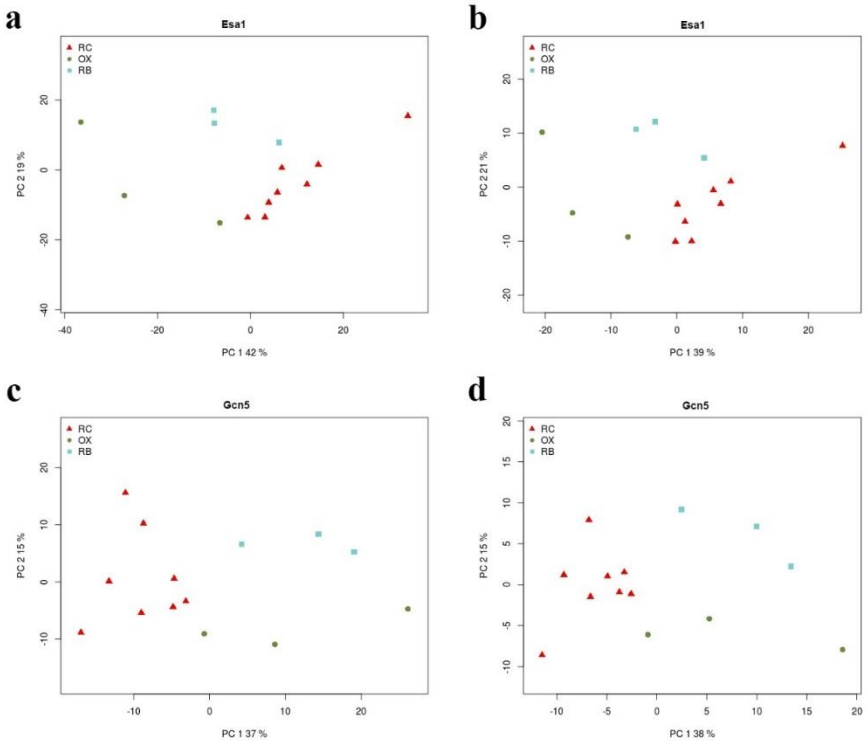

***Figure 14**. Principal component analysis (PCA) of the read counts of Esa1 (**a** and **b**) and Gcn5 (**c** and **d**) samples, before (left) and after (right) the data correction. PC1 and PC2 are considered. The samples*

are identified according to their phase: green circles for OX phase, blue squares for RB phase, and red triangles for RC phase.

The belonging of each time point to a certain phase was established following the previously made enrichment analysis (see Assignation of time points to each YMC phase). It can be considered that the assignment of the samples was accurate, as the ones whose phase was uncertain cluster as predicted.

Linear regression models

The data of ChIP-Seq of Esa1, Gcn5, H3K9 and H3K18 were integrated by means of linear regression models, using MORE software (Tarazona, Tomás Riquelme, Martínez Mira, Clemente Císcar, & Conesa, 2018).

With MORE, acetylation CHIP-Seq data were related to the data of chromatin modifying enzyme ChIP-Seq. By constructing a linear regression model for each gene that is potentially regulated by both elements (one enzyme and one acetylation), only those genes with a significative correlation, whether it is positive or negative, are returned as a result.

By integrating the read count matrixes of the two acetyltransferases with the count matrixes of each of the histone acetylations, several subsets of genes, with a significant correlation between the acetylation and the enzyme, were obtained. The results of this integration can be seen in **Table 11**.

***Table 11**. Genes associated to the histone acetylations H3K9 and H3K18 that were identified as potentially regulated by Esa1 or Gcn5 by MORE software. Genes with a positive and a negative correlation to the regulators were both included. Note that, for some genes, both positive and negative correlation was found for the same regulator.*

	H3K9			**H3K18**		
	Total	**Positive correlation**	**Negative correlation**	**Total**	**Positive correlation**	**Negative correlation**
Regulated by Esa1	224	166	70	266	192	92
Regulated by Gcn5	281	247	42	330	289	51

Overall, Gcn5 was considered a potential regulator of more genes than Esa1, for both histone acetylations. This coincides with the fact that Gcn5 is related to the acetylation of H3 histone, while Esa1 has been associated with H4 histone (Rando & Winston, 2012), so it is natural to see more genes related to Gcn5 if two acetylations of H3 histone are evaluated.

Also, both Esa1 and Gcn5 potentially regulate more genes associated to the H3K18. This is due to the quantity of genes originally associated to H3K18, which was superior to the quantity associated to H3K9 (see **Table 5**).

When the results from **Table 11** are compared to the ones shown in **Table 8**, is can be seen that the quantity of genes that could be potentially regulated by the two acetyltransferases is lower according the results of MORE. As the software uses read counts data, and therefore considers the temporal oscillation of each variable, many genes are filtered out when the linear regression models are constructed because of not presenting a real variation with time.

Regarding the correlation of the regulators with the histone acetylations, most of the genes regulated by each enzyme have a positive correlation. The genes featuring a negative correlation are superior for the Esa1 acetyltransferase, both in quantity and in proportion to the genes with a positive correlation. An example can be seen in **Figure 15**.

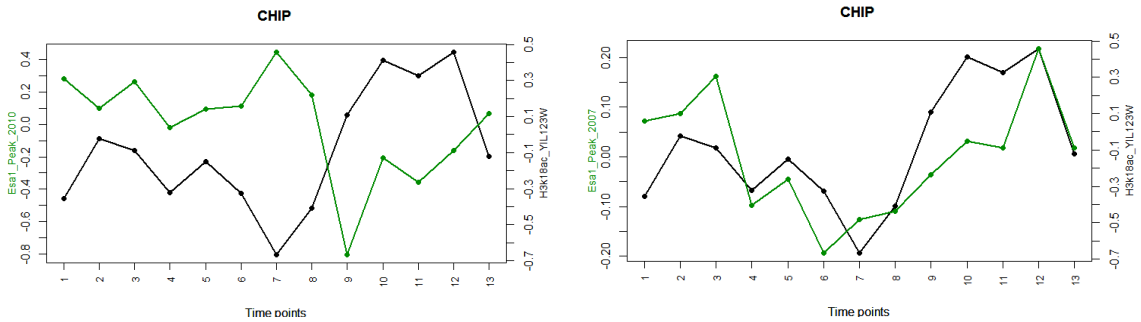

Figure 15. Example of a negative (left) and a positive (right) correlation of Esa1 acetyltransferase and H3K18 enzyme, regarding the gene YIL123W.

By intersecting them with RNA-Seq data, each subset of genes regulated by any acetylation-enzyme combination was divided according to the YMC phase in which the genes expressed in a differential manner (**Table 12**).

***Table 12**. Genes associated to the histone acetylations H3K9 and H3K18 that were identified as potentially regulated by Esa1 or Gcn5 by MORE software, divided by their YMC phase.*

	H3K9			**H3K18**		
	OX phase	**RB phase**	**RC phase**	**OX phase**	**RB phase**	**RC phase**
Regulated by Esa1	122	23	79	150	23	93
Regulated by Gcn5	192	19	70	194	21	115

The functional enrichment analysis of these genes showed similar results for all OX phase gene subsets. The featured terms were typical for the phase, related to growth genes, translation, ribosomes or RNA binding (*Structural constituent of ribosome*, *Cytoplasmic translation*, *RNA binding*, *rRNA export from nucleus*…).

In the RB phase, however, the enriched GO terms were different for the analysed subsets, probably because the samples on which the enrichment was performed were small, in comparison to those of OX or RC phases. Moreover, most of the enriched GO terms displayed a low significance. Esa1 subsets were enriched in terms that are related to the cell cycle, like *Chromatin assembly or disassembly*, *Replication fork protection* or *Nucleosome*; additionally, Esa1-H3K18 genes were also enriched in *Filamentous growth* and *Chromatin silencing*. Gcn5-H3K9 genes were functionally enriched in *Cell wall organization* and *Mitochondrial small ribosomal subunit*, and Gcn5-H3K18 genes had no significatively enriched GO terms.

The GO terms enriched in the RC phase were related to the characteristic functionalities of the phase, yet were not as defined as in the OX phase enrichment. Three of the enriched gene subsets (Esa1-H3K9, Esa1-H3K18 and Gcn5-H3K18) displayed GO terms related to fatty acids, which seem related to their metabolism (*Fatty acid beta-oxidation*, *Fatty-acyl-CoA transport*, *Fatty acid metabolic process*…); other three subsets (Esa1-H3K18, Gcn5-H3K9 and Gcn5-H3K18) were enriched in terms related to stress conditions (*Trehalose metabolism in response to stress*, *Cellular response to heat*, *Cellular response to desiccation*). Gcn5-H3K18 also featured terms related to malate (*Malate metabolic process*, *Malate dehydrogenase activity*) and carbohydrate metabolism (*Carbohydrate metabolic process*).

Conclusion

The exact mechanism of regulation of the different cellular processes that comprise the yeast metabolic cycle still remains a mystery. However, even the smallest insights into the elements that are related to this phenomenon take us closer to the answer.

The development of this genome-wide study allowed a general view of the role of three yeast chromatin modifying enzymes -Esa1, Gcn5 and Set1- in the YMC, and outlined their possible relations and interactions with differentially expressed genes, histone acetylation sites and transcription factor.

The observed results suggest that the two acetyltransferases, Esa1, and Gcn5, have a bigger impact on the YMC than the Set1 methyltransferase; and, as it is logical, they seem much more related to histone acetylations. The different putatively regulated genes that are shared by several elements, whether they are enzymes or acetylations, display the functionalities that are characteristic of the three phases of the YMC, enforcing the idea that the cycle is controlled by different regulatory mechanisms in a coordinated manner.

The performed study, however, had several limitations. Some of them were related to the raw data that were analysed: their quality was not ideal, and the uneven sampling might have caused an underrepresentation of some genes. The *a priori* knowledge of the bioinformatic strategies and specialized software that had to be applied in order to process these data was not always sufficient, so had to be improved alongside the analysis execution. Additionally, a new programming language, R, had to be learnt to efficiently work with the obtained results of ChIP-Seq data.

But one of the most important challenges is the studied matter itself. The YMC is a complex phenomenon that is characterized by many different processes at many different biological levels. This complicates its study and data analysis, even with a multi-omics approach, because of all the information that has to be computationally integrated and then understood at the level of a biological system.

This way, more research is required to fully understand how the YMC actually works in a yeast cell. New studies could focus on the genes potentially regulated by acetyltransferases and histone acetylations together, in search of patterns or conditions that cause the transcription of very concrete genes in each YMC phase. The individual contribution of the transcription factors to the cycle could also be assessed, both to prove their significance as potential regulators and to discover how exactly they come to regulate specific genes. And of course, multi-omics studies that integrate new trans-regulatory elements would help to better visualize the functioning of the cycle and to construct a regulatory network among all the elements implied in the YMC.

On a more personal level, the performance of this Bachelor's thesis was a great chance to participate in a real research process, and not only to widen my knowledge of the yeast metabolic cycle and its genetic and metabolic basis, but also to learn much more about bioinformatics and its application in biotechnology. This, of course, demanded considerable effort, as many of the used strategies in this study, as well as the R language management, required more practice and a deeper understanding that the one I initially had, despite having been explained or mentioned in class. However, I am satisfied with all I learnt while performing this study.

The bioinformatics basis I obtained thanks to the work in the Genomics of Gene Expression group will surely be useful to study and understand different biological issues in the future, as I intend to keep working in this field and continue to study the many questions that still remain unresolved in biology.

Bibliography

Andrews, S. (2010). *FastQC*. Retrieved June 20th, 2019, from http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., . . . Sherlock, G. (2000). Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature genetics, 25*(1), 25-29. doi:10.1038/75556

Bolger, A. M., Lohse, M., & Usadel, B. (2014). Trimmomatic: a flexible trimmer for Illumina sequence data. *Bioinformatics (Oxford, England), 30*(15), 2114-2120. doi:10.1093/bioinformatics/btu170

Bradnam, K. R., Seoighe, C., Sharp, P. M., & Wolfe, K. H. (1999). G+C content variation along and among Saccharomyces cerevisiae chromosomes. *Molecular Biology and Evolution, 16*(5), 666-675. doi:10.1093/oxfordjournals.molbev.a026

Burnetti, A. J., Aydin, M., & Buchler, N. E. (2016). Cell cycle Start is coupled to entry into the yeast metabolic cycle across diverse strains and growth rates. *Molecular biology of the cell, 27*(1), 64-74. doi:10.1091/mbc.E15-07-0454

Cai, L., Sutter, B. M., Li, B., & Tu, B. P. (2011). Acetyl-CoA Induces Cell Growth and Proliferation by Promoting the Acetylation of Histones at Growth Genes. *Molecular Cell, 42*(4), 426-437. doi:10.1016/j.molcel.2011.05.004

Cherry, J. M., Hong, E. L., Amundsen, C., Balakrishnan, R., Binkley, G., Chan, E. T., . . . Wong, E. D. (2012). Saccharomyces Genome Database: the genomics resource of budding yeast. *Nucleic acids research, 40*(Database issue), D700-D705. doi:10.1093/nar/gkr1029

Durinck, S., Moreau, Y., Kasprzyk, A., Davis, S., De Moor, B., Brazma, A., & Huber, W. (2005). BioMart and Bioconductor: a powerful link between biological databases and microarray data analysis. *Bioinformatics (Oxford, England), 21*(16), 3439-3440. doi:10.1093/bioinformatics/bti525

Edgar, R., Domrachev, M., & Lash, A. E. (2002). Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Research, 30*(1), 207-210. doi:10.1093/nar/30.1.207

Furió Tarí, P., Conesa, A., & Tarazona, S. (2016). RGmatch: matching genomic regions to proximal genes in omics data integration. *BMC bioinformatics, 17*(Suppl 15), 427. doi:10.1186/s12859-016-1293-1

genohub.com. (n.d.). *Recommended Coverage and Read Depth for NGS Applications*. Retrieved 07 1, 2019, from Genohub: https://genohub.com/recommended-sequencing-coverage-by-application/

Goffeau, A., Barrell, B. G., Bussey, H., Davis, R. W., Dujon, B., Feldmann, H., . . . Oliver, S. G. (1996). Life with 6000 genes. *Science, 274*(5287), 546, 563-567. doi:10.1126/science.274.5287.546

Griffiths, A. J., Miller, J. H., Suzuki, D. T., Lewontin, R. C., & Gelbart, W. M. (2000). Transcription: an overview of gene regulation in eukaryotes. In W. H. Freeman (Ed.), *An Introduction to Genetic Analysis* (7th ed.). New York. Retrieved from https://www.ncbi.nlm.nih.gov/books/NBK21780/

Jagannath, A., Taylor, L., Wakaf, Z., Vasudevan, S. R., & Foster, R. G. (2017). The genetics of circadian rhythms, sleep and health. *Human Molecular Genetics, 26*(R2), R128-R138. doi:10.1093/hmg/ddx240

Jiang, S., & Mortazavi, A. (2018). Integrating ChIP-seq with other functional genomics data. *Briefings in functional genomics, 17*(2), 104-115. doi:10.1093/bfgp/ely002

Klug, W. S., Cummings, M. R., Spencer, C. A., & Palladino, M. A. (2012). *Concepts of Genetics* (10th ed.). San Francisco, California, USA: Pearson.

Kouzarides, T. (2007). Chromatin modifications and their function. *Cell, 128*(4), 693-705. doi:10.1016/j.cell.2007.02.005

Kuang, Z., Cai, L., Zhang, X., Ji, H., Tu, B. P., & Boeke, J. D. (2014). High temporal-resolution view of transcription and chromatin states across distinct metabolic states in budding yeast. *Nature Structural & Molecular Biology, 21*(10), 854-863. doi:10.1038/nsmb.2881

Langmead, B., & Salzberg, S. L. (2012). Fast gapped-read alignment with Bowtie 2. *Nature methods, 9*(4), 357-359. doi:10.1038/nmeth.1923

Law, M. J., & Finger, M. A. (2017). The Saccharomyces cerevisiae Cdk8 Mediator Represses AQY1 Transcription by Inhibiting Set1p-Dependent Histone Methylation. *G3 (Bethesda, Md.), 7*(3), 1001-1010. doi:10.1534/g3.117.039586

Lawrence, M., Huber, W., Pagès, H., Aboyoun, P., Carlson, M., Gentleman, R., . . . Carey, V. (2013). Software for Computing and Annotating Genomic Ranges. *PLoS Computational Biology, 9*(8). doi:10.1371/journal.pcbi.1003118

Leinonen, R., Sugawara, H., Shumway, M., & International Nucleotide Sequence Database Collaboration. (2011). The Sequence Read Archive. *Nucleic acids research, 39*(Database issue), D19-D21. doi:doi:10.1093/nar/gkq1019

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., . . . 1000 Genome Project Data Processing Subgroup. (2009). The Sequence Alignment/Map format and SAMtools. *Bioinformatics (Oxford, England), 25*(16), 2078-2079. doi:10.1093/bioinformatics/btp352

Liao, Y., Smyth, G. K., & Shi, W. (2019). The R package Rsubread is easier, faster, cheaper and better for alignment and quantification of RNA sequencing reads. *Nucleic Acids Research, 47*(8), e47. doi:10.1093/nar/gkz114

Liu, T., Ortiz, J. A., Taing, L., Meyer, C. A., Lee, B., Zhang, Y., . . . Liu, X. S. (2011). Cistrome: an integrative platform for transcriptional regulation studies. *Genome biology, 12*(8), R83. doi:10.1186/gb-2011-12-8-r83

Manzoni, C., Kia, D. A., Vandrovcova, J., Hardy, J., Wood, N. W., Lewis, P. A., & Ferrari, R. (2016). Genome, transcriptome and proteome: the rise of omics data and their integration in biomedical sciences. *Briefings in bioinformatics, 19*(2), 286-302. doi:10.1093/bib/bbw114

Maunakea, A. K., Chepelev, I., & Zhao, K. (2010). Epigenome Mapping in Normal and Disease States. (B. Bruneau, Ed.) *Circulation research, 107*(3), 327-339. doi:10.1161/CIRCRESAHA.110.222463

McDonald, J. (2014). *Handbook of Biological Statistics* (3rd ed.). Baltimore, Maryland, USA: Sparky House Publishing.

Misra, B. B., Langefeld, C. D., Olivier, M., & Cox, L. A. (2018). Integrated omics: tools, advances and future approaches. *Journal of Molecular Endocrinology, 62*(1), R21-R45. doi:10.1530/JME-18-0055

Nakato, R., & Shirahige, K. (2017). Recent advances in ChIP-seq analysis: from quality management to whole-genome annotation. *Briefings in bioinformatics, 18*(2), 279-290. doi:10.1093/bib/bbw023

nature.com. (n.d.). *Epigenomics*. Retrieved 07 01, 2019, from nature.com: https://www.nature.com/subjects/epigenomics

Okonechnikov, K., Conesa, A., & García-Alcalde, F. (2016). Qualimap 2: advanced multi-sample quality control for high-throughput sequencing data. *Bioinformatics (Oxford, England), 32*(2), 292-294. doi:10.1093/bioinformatics/btv566

Park, P. J. (2009). ChIP-seq: advantages and challenges of a maturing technology. *Nature Reviews Genetics, 10*(10), 669-680. doi:10.1038/nrg2641

R Core Team. (2018). R: A language and environment for statistical computing. Vienna, Austria: Foundation for Statistical Computing. Retrieved from https://www.R-project.org/

Rando, O. J., & Winston, F. (2012). Chromatin and transcription in yeast. *Genetics, 190*(2), 351-387. doi:10.1534/genetics.111.132266

Rao, A. R., & Pellegrini, M. (2011). Regulation of the yeast metabolic cycle by transcription factors with periodic activities. *BMC Systems Biology, 5*(160). doi:10.1186/1752-0509-5-160

Ringnér, M. (2008). What is principal component analysis? *Nature Biotechnology, 26*(3), 303-304. doi:10.1038/nbt0308-303

Rowicka, M., Kudlicki, A., Tu, B. P., & Otwinowski, Z. (2007). High-resolution timing of cell cycle-regulated gene expression. *Proceedings of the National Academy of Sciences, 104*(43), 16892-16897. doi:10.1073/pnas.0706022104

RStudio Team. (2016). RStudio: Integrated Development for R. Boston, MA: RStudio, Inc. Retrieved from http://www.rstudio.com/

Saka, K., Takahashi, A., Sasaki, M., & Kobayashi, T. (2016). More than 10% of yeast genes are related to genome stability and influence cellular senescence via rDNA maintenance. *Nucleic acids research, 44*(9), 4211-4221. doi:10.1093/nar/gkw110

Sánchez Gaya, V., Casaní Galdón, S., Ugidos, M., Kuang, Z., Mellor, J., Conesa, A., & Tarazona, S. (2018). Elucidating the Role of Chromatin State and Transcription Factors on the Regulation of the Yeast Metabolic Cycle: A Multi-Omic Integrative Approach. *Frontiers in Genetics, 9*, 578. doi:10.3389/fgene.2018.00578

Schneider, M. V., & Orchard, S. (2011). Omics Technologies, Data and Bioinformatics Principles. In B. Mayer (Ed.), *Bioinformatics for Omics Data* (pp. 3-30). Humana Press. doi:10.1007/978-1-61779-027-0_1

SRA Toolkit Development Team. (n.d.). Retrieved June 20th, 2019, from SRA-Tools: http://ncbi.github.io/sra-tools/

Tarazona, S., Furio-Tari, P., Turra, D., Pietro, A. D., Nueda, M. J., Ferrer, A., & Conesa, A. (2015). Data quality aware analysis of differential expression in RNA-seq with NOISeq R/Bioc package. *Nucleic Acids Research, 43*(21), e140. doi:10.1093/nar/gkv711

Tarazona, S., García Alcalde, F., Dopazo, J., Ferrer, A., & Conesa, A. (2011). Differential expression in RNA-seq: a matter of depth. *Genome Research, 21*(12), 2213–2223. doi:10.1101/gr.124321.111

Tarazona, S., Tomás Riquelme, B., Martínez Mira, C., Clemente Císcar, M., & Conesa, A. (2018). MORE: Multi-Omics REgulation by regression models. R package version 0.1.0.

Teixeira, M. C., Monteiro, P. T., Palma, M., Costa, C., Godinho, C. P., Pais, P., . . . Sá-Correia, I. (2018). YEASTRACT: an upgraded database for the analysis of transcription

regulatory networks in Saccharomyces cerevisiae. *Nucleic Acids Research, 46*(D1), D348-D353. doi:10.1093/nar/gkx842

The Gene Ontology Consortium. (2019). The Gene Ontology Resource: 20 years and still GOing strong. *Nucleic Acids Research, 47*(D1), D330-D338. doi:10.1093/nar/gky1055

Thomas, R., Thomas, S., Holloway, A. K., & Pollard, K. S. (2017). Features that define the best ChIP-seq peak calling algorithms. *Briefings in bioinformatics, 18*(3), 441-450. doi:10.1093/bib/bbw035

Tu, B. P., & McKnight, S. L. (2006). Metabolic cycles as an underlying basis of biological oscillations. *Nature Reviews Molecular Cell Biology, 7*(9), 696-701. doi:10.1038/nrm1980

Tu, B. P., Kudlicki, A., Rowicka, M., & McKnight, S. L. (2005). Logic of the Yeast Metabolic Cycle: Temporal Compartmentalization of Cellular Processes. *Science, 310*(5751), 1152-1158. doi:10.1126/science.1120499

Tu, B. P., Mohler, R. E., Liu, J. C., Dombek, K. M., Young, E. T., Synovec, R. E., & McKnight, S. L. (2007). Cyclic changes in metabolic state during the life of a yeast cell. *Proceedings of the National Academy of Sciences, 104*(43), 16886-16891. doi:10.1073/pnas.0708365104

Wang, S., Zang, C., Xiao, T., Fan, J., Mei, S., Qin, Q., . . . Liu, X. S. (2016). Modeling cis-regulation with a compendium of genome-wide histone H3K27ac profiles [Supplementary methods]. *Genome research, 26*(10), 1417-1429. doi:10.1101/gr.201574.115

Watson, J. D., Baker, T. A., Bell, S. P., Gann, A., Levine, M., & Losick, R. M. (2014). *Molecular Biology of the Gene* (7th ed.). Pearson.

Wickham, H. (2016). ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York. Retrieved from https://ggplot2.tidyverse.org

Zeng, I., & Lumley, T. (2018). Review of Statistical Learning Methods in Integrated Omics Studies (An Integrated Information Science). *Bioinformatics and biology insights, 12*. doi:10.1177/1177932218759292

Zerbino, D. R., Achuthan, P., Akanni, W., Amode, M. R., Barrell, D., Bhai, J., . . . Flicek, P. (2018). Ensembl 2018. *Nucleic acids research, 46*(D1), D754-D761. doi:10.1093/nar/gkx1098

Zhang, Y., Liu, T., Meyer, C. A., Eeckhoute, J., Johnson, D. S., Bernstein, B. E., . . . Liu, X. S. (2008). Model-based Analysis of ChIP-Seq (MACS). *Genome Biology, 9*(9), R137. doi:10.1186/gb-2008-9-9-r137

Zheng, R., Wan, C., Mei, S., Qin, Q., Wu, Q., Sun, H., . . . Liu, X. S. (2019). Cistrome Data Browser: expanded datasets and new tools for gene regulatory analysis. *Nucleic acids research, 47*(D1), D729-D735. doi:10.1093/nar/gky1094

Annexes

Scripts

Analysis of RGmatch results and functional enrichment, in Esa1

```
##### Generar bucle total para todos los archivos, con un report final #####


### Crear bucle ###


## 0. Elegir el directorio de trabajo (donde esten todos los archivos del analisis de RGmatch de la proteina)
## e indicar donde estan los archivos bed


setwd("/home/biouser/Desktop/FilteredGenes/Reports/Esa1/") # DirecciÃ³n en la que guardar report de este analisis
rutargmatch <- "/home/biouser/Desktop/Cluster/YMC/denovo/RGmatch/Esa1" # Indicar ruta de reports del RGmatch para la proteina
rutabeds <- "/home/biouser/Desktop/Cluster/YMC/denovo/RGmatch/beds/Esa1" #Indicar ruta de los archivos bed
modificador <- "Esa1" # Modificador analizado


'%ni%' <- Negate('%in%') # Funcion para el "not in"


## 0.1 Obtener la lista de archivos .txt del RGmatch para el modificador de histona deseado


archivos <- list.files(rutargmatch)


## 0.2 Hacer bucle para procesar cada archivo de RGmatch


library(rlist) # Paquete para trabajar con listas


i <- 1 # Establecer contador a 1, para que empiece por el archivo 1


totalGOlist <- list() # Lista con totos los GO enriquecidos para cada tiempo
totalGenelist <- list() # Lista con todos los genes asociados para cada tiempo
totalFiltGenelist <- list()
```

```
for (archivo in archivos)
{
  nombre <- archivo
  carpeta <- rutargmatch
  ruta <- paste(carpeta,"/",nombre, sep = "")



### Sacar estadisticas del RGmatch ###



## 1. Cargar los datos del .txt de RGmatch a R

rgmatch_genes = read.table(ruta, header=T, as.is=T)



## 2. Determinar el porcentaje de picos con al menos 1 gen asociado

# Cantidad de picos (name es la columna de picos), sin repetir (porque unique) que se asocian a un gen
length(unique(rgmatch_genes$name)) #picos con 1 gen asociado
picasos <- length(unique(rgmatch_genes$name))

# Cantidad de asociaciones de picos con genes totales
length(rgmatch_genes$name) # total de asociaciones
asotot <- as.character(length(rgmatch_genes$name))

# Lista de genes asociados a picos, sin repetir genes
genes <- unique(rgmatch_genes$Gene)

totalGenelist <- list.append(totalGenelist, genes) # AÃ±adir el conjunto de genes de este tiempo a lista externa

#Cargar los datos de los picos

beds <- list.files(rutabeds) # Obtener nombres de archivos bed
```

```
picos <- paste(rutabeds,"/",beds[i], sep = "") # Crear nombre de ruta completo
peaks = read.table(picos, header=F)
nrow(peaks)
macpic <- nrow(peaks)

#Determinar el porcentaje de picos, sin repetir, que tengan un gen asociado (al menos 1)
associatedPeaks_perc = length(unique(rgmatch_genes$name))/nrow(peaks)*100


## 4. Hay picos con mas de una asociacion? Cuantos hay?  La asociaciÃ³n es en el mismo gen?

# Hacer conjunto de picos con +1 asociacion (duplicated),
# usando solo el nombre del pico (indicando columna "name"), sin repetir los picos (unique)
multAssocRegions = unique(rgmatch_genes[duplicated(rgmatch_genes$name),"name"] )

#Cantidad de picos con +1 asociaciÃ³n
length(multAssocRegions)

#Porcentaje de picos con +1 asociacion frente a todos los picos con asociacion
percMultiAssocRegions = length(multAssocRegions)/picasos*100

# Porcentaje de picos Ãºnicos asociados a TSS, Promotor, 1r Exon o Cuerpo del Gen
intpeaks <- unique(rgmatch_genes[rgmatch_genes$Area%in%c('TSS', 'PROMOTER', 'GENE_BODY', '1st_EXON'),]$name)
length(intpeaks)
portotal <- length(intpeaks)/picasos*100


############################################################
####### GENES A COGER PARA EL ENRICHMENT ANALYSIS #######
############################################################

filtergenes <- unique(rgmatch_genes[rgmatch_genes$Area%in%c('TSS', 'PROMOTER'),]$Gene)

totalFiltGenelist <- list.append(totalFiltGenelist, filtergenes)
```

```
############################################################
############################################################


# Porcentaje de picos Ãºnicos asociados a TSS,
tsspeaks <- unique(rgmatch_genes[rgmatch_genes$Area%in%c('TSS'),]$name)
length(tsspeaks)
portss <- length(tsspeaks)/picasos*100

# Porcentaje de picos Ãºnicos asociados a PROMOTOR
prompeaks <- unique(rgmatch_genes[rgmatch_genes$Area%in%c('PROMOTER'),]$name)
length(prompeaks)
porprom <- length(prompeaks)/picasos*100

# Porcentaje de picos Ãºnicos asociados a 1st Exon
expeaks <- unique(rgmatch_genes[rgmatch_genes$Area%in%c('1st_EXON'),]$name)
length(expeaks)
porex <- length(expeaks)/picasos*100

# Porcentaje de picos Ãºnicos asociados a GENE_BODY
bodpeaks <- unique(rgmatch_genes[rgmatch_genes$Area%in%c('GENE_BODY'),]$name)
length(bodpeaks)
porbod <- length(bodpeaks)/picasos*100

# Crear conjunto de picos, filtrando por el nombre del pico (elegir columna "name),
# siempre que se encuentre entre los multiasociados (lista de multiasociados)
# Se obtiene lista de multiasociados con todas las asociaciones
multAssoc = rgmatch_genes[rgmatch_genes$name%in%multAssocRegions,]


## 5. Graficar la distribucion de los eventos a traves de las regiones ("TSS","PROMOTER",
"1st EXON",...)
# en un pie chart


# Cargar paquete para hacer grÃ¡ficas
library(ggplot2)
```

```
# Hacer un barplot (grÃ¡fico de barras), dividiendo en tipos de region (Area)
ggplot(rgmatch_genes, aes(x=Area, fill=Area), color="black") +
  geom_bar()

# Hacer un piechart (grÃ¡fico circular o grÃ¡fico tarta)
df = as.data.frame(table(rgmatch_genes$Area))
slices <- df$Freq
lbls <-df$Var1
pct <- round(slices/sum(slices)*100)
lbls <- paste(lbls, pct) # add percents to labels
lbls <- paste(lbls,"%",sep="") # ad % to labels
pie(slices,labels = lbls, pch=10 )
```

6. Graficar la distribucion de las distancias de los picos al TSS. Donde estan enriquecidos los binding sites?

```
ggplot(rgmatch_genes, aes(x=TSSDistance, fill=Area)) +
  geom_density()

ggplot(rgmatch_genes[which(rgmatch_genes$TSSDistance<10000),], aes(x=TSSDistance, fill=Area)) +
  geom_density()

ggplot(rgmatch_genes[which(rgmatch_genes$TSSDistance<10000),], aes(x=TSSDistance, fill=Area))+
  geom_histogram(bins=200)

ggplot(rgmatch_genes[which(rgmatch_genes$Area%in%c("PROMOTER", "TSS","UPSTREAM")),], aes(x=TSSDistance, fill=Area)) +
  geom_density()

ggplot(rgmatch_genes[which(rgmatch_genes$Area%in%c("TTS", "INTRON","DOWNSTREAM", "GENE_BODY") & rgmatch_genes$TSSDistance<500000),], aes(x=TSSDistance, fill=Area)) +
  geom_density()
```

```
#######################
### Enriquecimiento ###

## 1. Cargar el paquete biomaRt, con la info de Ensembl

library(biomaRt)

## 2. Cargar los datos de la levadura desde Ensembl

biomartYeast = useMart(biomart = "ensembl", dataset="scerevisiae_gene_ensembl")

# Ver los atributos que hay

atributos = listAttributes(biomartYeast)
atributos[grep("GO", atributos$description, ignore.case = TRUE),]



## 3. Enrichment functions ORA (Over Representation Analysis) > Funciones para hacer enriquecimiento

EnrichALLterms = function (test, notTest, annotation,
                           p.adjust.method = "fdr") {

  annot2test = unique(annotation[,2])

  resultat = t(sapply(annot2test, Enrich1term, test = test, notTest = notTest, annotation = annotation))

  return (data.frame(resultat[,-6],
                     "pval" = as.numeric(resultat[,"pval"]),
                     "adjPval" = p.adjust(as.numeric(resultat[,"pval"]), method = p.adjust.method),
                     stringsAsFactors = F))

}

Enrich1term = function (term, test, notTest, annotation) {
```

```
  annotTest = length(intersect(test, annotation[annotation[,2] == term,1]))

  if ((annotTest) > 0) {
    annotNOTtest = length(intersect(notTest, annotation[annotation[,2] == term,1]))
    mytest = matrix(c(annotTest, length(test)-annotTest, annotNOTtest, length(notTest)-
annotNOTtest), ncol = 2)
    resultat = c(term, annotTest, length(test), annotNOTtest, length(notTest),
                 #TEST DE FISHER
                 fisher.test(mytest, alternative = "greater")$p.value)
    names(resultat) = c("term", "annotTest", "test", "annotNotTest", "notTest", "pval")
  } else {
    resultat = c(term, 0, 0, 0, 0, 100)
    names(resultat) = c("term", "annotTest", "test", "annotNotTest", "notTest", "pval")
  }

  return(resultat)

}


## 4. Over Representation Analysis (ORA)

#Obtener nombres de los genes asociados en la muestra

test <- filtergenes
length(test) # 2466 genes asociados


#Obtener nombres de todos los genes de la levadura

notTest <-  getBM(attributes = c("ensembl_gene_id"),
                  mart=biomartYeast)
notTest <- as.vector(t(notTest))
length(notTest) # 7036 genes totales en la levadura
```

```
#Resta entre los genes totales en la levadura y los genes de la muestra
notTest <- setdiff(notTest, test)
length(notTest) # 4662 genes



#Crear anotacion de los genes que hay en el organismo (gen asociado y descripciÃ³n del GO term (name_1006) asociado)

annotation = getBM(attributes = c("ensembl_gene_id","name_1006"),
                   mart=biomartYeast)



#Hacer analisis de enriquecimiento

myEnrichResults = EnrichALLterms(test = test,
                                 notTest = notTest,
                                 # De la anotaciÃ³n de genes de la muestra, coger solo la columna 1 (nombre del gen)
                                 # y 3 (nombre del termino GO)
                                 annotation = annotation,
                                 p.adjust.method = "fdr")

## 5. Seleccionar GO terms enriquecidos en la muestra

GOenriq <- myEnrichResults[myEnrichResults$pval<0.05, c('term','pval')] # Todos lo GO enriquecidos
GOenriqSel <- GOenriq[order(GOenriq$pval),] # Ordenados de menor a mayor p value
GOenriq25 <- GOenriqSel[2:26,]

GOenriqAdd <- GOenriq[,'term']
totalGOlist <-list.append(totalGOlist, GOenriqAdd) #Lista con conjutnos de GO enriquecidos en cada tiempo



### Crear un archivo Report ###

## 1. Crear datos con las variables con texto
```

```
archi_nombre <- substr(archivo, 1, 10)
tiempo <- as.character(i)

# Nombre del archivo de informe

report_name <- paste("Enrich_rep-",archi_nombre,"-Time_",tiempo,"-",modificador,".txt")

# Datos generales del analisis

titulo <- "### Informe del analisis de enriquecimiento ###"
archivo_analizado <- paste("Archivo analizado: ",
                           archivo,
                           sep = "")
bed_analizado <- paste("Archivo .bed analizado: ",
                       beds[i],
                       sep = "")
info_archivo <- paste(archi_nombre,
                      ", muestra de ",
                      modificador,
                      " en el tiempo ",
                      tiempo,
                      sep = "")

# Datos estadisticos generales

intro1 <- "# Datos estadisticos generales"
est1 <- paste("Para el modificador de histonas ",
              modificador,
              ", a tiempo ",
              tiempo,
              ", se han detectado ",
              as.character(macpic),
              " picos con MACS2.",
              sep = "")
est2 <- paste("A su vez, RGmatch ha detectado asociaciones con genes para ",
              as.character(picasos),
              " picos, con un total de ",
```

```
            asotot,
            " asociaciones.",
            sep = "")
  est3 <- paste("El ",
            as.character(round(associatedPeaks_perc,3)),
            "% de picos tiene, por lo tanto, asociacion a, al menos, un gen.",
            sep = "")
  est4 <- paste("El ",
            as.character(round(percMultiAssocRegions,3)),
            "% de picos (",
            as.character(length(multAssocRegions)),
            " picos) tiene mÃ¡s de una asociacion con respecto a la totalidad de picos
asociados.",
            sep = "")

  # Datos estadisticos sobre regiones especificas

  intro2 <- "# Asociaciones a regiones especificas"
  aso1 <- paste("De todos los picos asociados (",
            as.character(picasos),
            "), el ",
            as.character(round(portotal,3)),
            "% (",
            as.character(length(intpeaks)),
            ") de picos estÃ¡ asociado al TSS, promotor, primer exon o cuerpo de un gen (al
menos a uno de estos).",
            sep = "")
  aso2 <- paste("El ",
            as.character(round(portss, 3)),
            "% (",
            as.character(length(tsspeaks)),
            ") de picos esta asociado al TSS.",
            sep = "")
  aso3 <- paste("El ",
             as.character(round(porprom, 3)),
             "% (",
             as.character(length(prompeaks)),
```

```
            ") de picos esta asociado al promotor.",
            sep = "")
  aso4 <- paste("El ",
            as.character(round(porex, 3)),
            "% (",
            as.character(length(expeaks)),
            ") de picos esta asociado al primer exon.",
            sep = "")
  aso5 <- paste("El ",
            as.character(round(porbod, 3)),
            "% (",
            as.character(length(bodpeaks)),
            ") de picos esta asociado al cuerpo de un gen.",
            sep = "")

  # Datos del enriquecimiento

  intro3 <- "# Enriquecimiento"

  listaGO <- ""

  for (elemento in GOenriq25$term)
  {
   listaGO <- paste(listaGO,elemento,'\n')
  }

  enriq1 <- paste("Entre los diferentes tÃ©rminos GO enriquecidos en esta muestra, se pueden
encontrar:",
            "\n",
            "\n",
            listaGO,
            sep = "")

  ## 2. Montar todo en un archivo de texto

  report <- file(report_name)
  writeLines(c(titulo,
```

```
            "",
            "",
            archivo_analizado,
            bed_analizado,
            info_archivo,
            "",
            "",
            intro1,
            "",
            est1,
            est2,
            "",
            est3,
            est4,
            "",
            "",
            intro2,
            "",
            aso1,
            "",
            aso2,
            aso3,
            aso4,
            aso5,
            "",
            "",
            intro3,
            "",
            enriq1),
          report)
close(report)

i <- i + 1

}

### Analisis global del enriquecimiento por cada tiempo y fase ###
```

```
## 1. Sacar una tabla con genes por cada tiempo

todosGenesFilt <- c()

for (tiemp in totalFiltGenelist)
{
  todosGenesFilt <- c(todosGenesFilt, tiemp) # Hacer listado con todos los genes asociados en todas las muestras
}

todosGenesFilt <- unique(todosGenesFilt) # Coger los genes sin repetir
todosGenesFilt <- sort(todosGenesFilt) # Ordenar los genes

tiemposMuestra <- c(1:14)
names(totalGenelist) <- tiemposMuestra
names(totalFiltGenelist) <- tiemposMuestra

for (tiem in tiemposMuestra)
{
  if (tiem == 1){
    geneMatrix <- matrix(todosGenesFilt%in%totalFiltGenelist[[tiem]])
  }else if (tiem < tail(tiemposMuestra, n = 1)){
    geneMatrix <- cbind(geneMatrix, todosGenesFilt%in%totalFiltGenelist[[tiem]])
  }else if (tiem == tail(tiemposMuestra, n = 1)){
    geneMatrix <- cbind(geneMatrix, todosGenesFilt%in%totalFiltGenelist[[tiem]])
    rownames(geneMatrix) <- todosGenesFilt
    colnames(geneMatrix) <- tiemposMuestra
  }
}



## 2. Enriquecimiento por cada tiempo



todosEnriq <- c()
```

```
for (tiemp in totalGOlist)
{
  todosEnriq <- c(todosEnriq, tiemp)
}

todosEnriq <- unique(todosEnriq)
todosEnriq <- sort(todosEnriq)

tiemposMuestra <- c(1:14)
names(totalGOlist) <- tiemposMuestra



for (tiem in tiemposMuestra)
{
  if (tiem == 1){
    goMatrix <- matrix(todosEnriq%in%totalGOlist[[tiem]])
  }else if (tiem < tail(tiemposMuestra, n = 1)){
    goMatrix <- cbind(goMatrix, todosEnriq%in%totalGOlist[[tiem]])
  }else if (tiem == tail(tiemposMuestra, n = 1)){
    goMatrix <- cbind(goMatrix, todosEnriq%in%totalGOlist[[tiem]])
    rownames(goMatrix) <- todosEnriq
    colnames(goMatrix) <- tiemposMuestra
  }
}



## 3. Enriquecimiento diferencial

j <- 1
f1 <- 1
f2 <- 1
f3 <- 1

for (func in todosEnriq){
  # goEqMatrix <- Matriz donde todos los GO son verdaderos o falsos
  # goDifMatrix <- Matriz donde los GO no aparecen en otods los tiempos
```

```
    # go1DifMatrix <- Matriz con todos los GO que aparecen en todos los tiempos menos 1, o
en un unico tiempo
    go0 <- goMatrix[j,(1:length(goMatrix[1,]))]
    go1 <- sum(goMatrix[j,])
    if (go1 == length(goMatrix[1,]) | go1 == 0){
      if (f1 == 1){
        goEqMatrix <- matrix(goMatrix[j,(1:length(goMatrix[1,]))], ncol = length(go0))
        ggg1 <- c(rownames(goMatrix)[j])
        f1 <- f1 + 1
      }else{
        goEqMatrix <- rbind(goEqMatrix, goMatrix[j,(1:length(goMatrix[1,]))])
        ggg1 <- c(ggg1, rownames(goMatrix)[j])
      }
    }else{
      if (f2 == 1){
        goDifMatrix <- matrix(goMatrix[j,(1:length(goMatrix[1,]))], ncol = length(go0))
        ggg2 <- c(rownames(goMatrix)[j])
        f2 <- f2 + 1
        if ((go1 == 1 | go1 == (length(go0) -1)) & f3 == 1){
          go1DifMatrix <- matrix(goMatrix[j,(1:length(goMatrix[1,]))], ncol = length(go0))
          ggg3 <- c(rownames(goMatrix)[j])
          f3 <- f3 + 1
        }
      }else{
        goDifMatrix <- rbind(goDifMatrix, goMatrix[j,(1:length(goMatrix[1,]))])
        ggg2 <- c(ggg2, rownames(goMatrix)[j])
        if ((go1 == 1 | go1 == (length(go0) -1)) & f3 == 1){
          go1DifMatrix <- matrix(goMatrix[j,(1:length(goMatrix[1,]))], ncol = length(go0))
          ggg3 <- c(rownames(goMatrix)[j])
          f3 <- f3 + 1
        }else if ((go1 == 1 | go1 == (length(go0) -1)) & f3 != 1){
          go1DifMatrix <- rbind(go1DifMatrix, goMatrix[j,(1:length(goMatrix[1,]))])
          ggg3 <- c(ggg3, rownames(goMatrix)[j])
        }
      }
    }
    j <- j + 1
```

```
}

rownames(goEqMatrix) <- ggg1
rownames(goDifMatrix) <- ggg2
rownames(go1DifMatrix) <- ggg3


## 4. Enriquecimiento diferencial por fases

totalGOPhaseList <- list()

# 4.1 Fase RC (Reductive/Charging)

# Puntos seguros: 1, 10-14.
# Puntos en duda: 2 y 9

# Genes representativos en puntos seguros

tablaRC <- geneMatrix[, c('1','2','9','10','11','12','13','14')]
k1 <- 1
repreRC <- c()

for (gene in todosGenesFilt){
  valtrue <- sum(tablaRC[k1,])
  if (valtrue >= 5){
    repreRC <- c(repreRC, rownames(tablaRC)[k1])
  }
  k1 <- k1 + 1
}

# ORA en puntos seguros

#Obtener nombres de los genes asociados en la fase RC

testRC <- repreRC
length(repreRC)
```

```
#Obtener nombres de todos los genes de la levadura

notTestRC <-  getBM(attributes = c("ensembl_gene_id"),
                    mart=biomartYeast)
notTestRC <- as.vector(t(notTestRC))
length(notTestRC) # 7036 genes totales en la levadura



#Resta entre los genes totales en la levadura y los genes de la fase RC
notTestRC <- setdiff(notTestRC, testRC)
length(notTestRC)



#Crear anotacion de los genes que hay en el organismo (gen asociado y descripciÃ³n del GO term (name_1006) asociado)

annotation = getBM(attributes = c("ensembl_gene_id","name_1006"),
                   mart=biomartYeast)



#Hacer analisis de enriquecimiento de la fase RC con puntos seguros

enriquecimientoRC = EnrichALLterms(test = testRC,
                                   notTest = notTestRC,
                                   annotation = annotation,
                                   p.adjust.method = "fdr")

#Seleccionar GO terms enriquecidos en la muestra

GOenriqRC <- enriquecimientoRC[enriquecimientoRC$pval<0.05, c('term','pval')] # Todos lo GO enriquecidos
GOenriqRC <- GOenriqRC[order(GOenriqRC$pval),] # Ordenados de menor a mayor p value



GoenriqRCadd <- GOenriqRC[,'term']
totalGOPhaseList <-list.append(totalGOPhaseList, 'RC' = GoenriqRCadd)
```

```
# 4.2 Fase OX (Oxidative)

# Puntos seguros: 3, 4
# Puntos en duda: 2, 5, 6

# Genes representativos en puntos seguros

tablaOX <- geneMatrix[, c('3','4','5')]
k2 <- 1
repreOX <- c()

for (gene in todosGenesFilt){
  valtrue <- sum(tablaOX[k2,])
  if (valtrue >= 2){
    repreOX <- c(repreOX, rownames(tablaOX)[k2])
  }
  k2 <- k2 + 1
}

# ORA en puntos seguros

#Obtener nombres de los genes asociados en la fase OX

testOX <- repreOX
length(repreOX)



#Obtener nombres de todos los genes de la levadura

notTestOX <-  getBM(attributes = c("ensembl_gene_id"),
                    mart=biomartYeast)
notTestOX <- as.vector(t(notTestOX))
length(notTestOX) # 7036 genes totales en la levadura



#Resta entre los genes totales en la levadura y los genes de la fase RC
```

```
notTestOX <- setdiff(notTestOX, testOX)
length(notTestOX)



#Crear anotacion de los genes que hay en el organismo (gen asociado y descripciÃ³n del GO term (name_1006) asociado)

#annotation = getBM(attributes = c("ensembl_gene_id","name_1006"),
#                   mart=biomartYeast)



#Hacer analisis de enriquecimiento de la fase RC con puntos seguros

enriquecimientoOX = EnrichALLterms(test = testOX,
                                   notTest = notTestOX,
                                   annotation = annotation,
                                   p.adjust.method = "fdr")

#Seleccionar GO terms enriquecidos en la muestra

GOenriqOX <- enriquecimientoOX[enriquecimientoOX$pval<0.05, c('term','pval')] # Todos lo GO enriquecidos
GOenriqOX <- GOenriqOX[order(GOenriqOX$pval),] # Ordenados de menor a mayor p value

GoenriqOXadd <- GOenriqOX[,'term']
totalGOPhaseList <-list.append(totalGOPhaseList, 'OX' = GoenriqOXadd)



# 4.3 Fase RB (Reductive/Building)

# Puntos seguros: 7, 8
# Puntos en duda: 6, 9
# 9 como punto definitivo (4 coincidencias con GO unicos de RC, 3 con OX, y 12 con RB)

# Genes representativos en puntos seguros

tablaRB <- geneMatrix[, c('6','7','8')]
```

```
k3 <- 1
repreRB <- c()

for (gene in todosGenesFilt){
  valtrue <- sum(tablaRB[k3,])
  if (valtrue >= 2){
    repreRB <- c(repreRB, rownames(tablaRB)[k3])
  }
  k3 <- k3 + 1
}

# ORA en puntos seguros

#Obtener nombres de los genes asociados en la fase OX

testRB <- repreRB
length(repreRB)



#Obtener nombres de todos los genes de la levadura

notTestRB <-  getBM(attributes = c("ensembl_gene_id"),
                    mart=biomartYeast)
notTestRB <- as.vector(t(notTestRB))
length(notTestRB) # 7036 genes totales en la levadura



#Resta entre los genes totales en la levadura y los genes de la fase RC
notTestRB <- setdiff(notTestRB, testRB)
length(notTestRB)



#Crear anotacion de los genes que hay en el organismo (gen asociado y descripciÃ³n del GO
term (name_1006) asociado)

#annotation = getBM(attributes = c("ensembl_gene_id","name_1006"),
#                   mart=biomartYeast)
```

```
#Hacer analisis de enriquecimiento de la fase RC con puntos seguros

enriquecimientoRB = EnrichALLterms(test = testRB,
                                   notTest = notTestRB,
                                   annotation = annotation,
                                   p.adjust.method = "fdr")

#Seleccionar GO terms enriquecidos en la muestra

GOenriqRB <- enriquecimientoRB[enriquecimientoRB$pval<0.05, c('term','pval')] # Todos lo GO enriquecidos
GOenriqRB <- GOenriqRB[order(GOenriqRB$pval),] # Ordenados de menor a mayor p value

GoenriqRBadd <- GOenriqRB[,'term']
totalGOPhaseList <-list.append(totalGOPhaseList, 'RB' = GoenriqRBadd)



# 4.4 Puntos en duda

# Punto 2

GOenriqP2 <- totalGOlist[[2]]

# Punto 5

GOenriqP5 <- totalGOlist[[5]]

# Punto 6

GOenriqP6 <- totalGOlist[[6]]

# Punto 9

GOenriqP9 <- totalGOlist[[9]]
```

```
## 5. Ver funciones Ãºnicas en cada fase

# Funciones Ãºnicas en RC

unicoRC <- c()

for (element in totalGOPhaseList[['RC']]){
  if (element%in%totalGOPhaseList[['RB']]){
    next
  }else if (element%in%totalGOPhaseList[['OX']]){
    next
  }else{
    unicoRC <- c(unicoRC, element)
  }
}

# Funciones Ãºnicas en OX

unicoOX <- c()

for (element in totalGOPhaseList[['OX']]){
  if (element%in%totalGOPhaseList[['RB']]){
    next
  }else if (element%in%totalGOPhaseList[['RC']]){
    next
  }else{
    unicoOX <- c(unicoOX, element)
  }
}

# Funciones Ãºnicas en RB

unicoRB <- c()

for (element in totalGOPhaseList[['RB']]){
  if (element%in%totalGOPhaseList[['OX']]){
```

```
    next
  }else if (element%in%totalGOPhaseList[['RC']]){
    next
  }else{
    unicoRB <- c(unicoRB, element)
  }
}


# Funciones coincidentes con las Ãºnicas de cada fase, para el punto 2

p2enRC <- c()
p2enRB <- c()
p2enOX <- c()


for (element in GOenriqP2){
  if (element%in%unicoRB){
    p2enRB <- c(p2enRB, element)
  }
  if (element%in%unicoRC){
    p2enRC <- c(p2enRC, element)
  }
  if (element%in%unicoOX){
    p2enOX <- c(p2enOX, element)
  }
}


# Funciones coincidentes con las Ãºnicas de cada fase, para el punto 5

p5enRC <- c()
p5enRB <- c()
p5enOX <- c()


for (element in GOenriqP5){
```

```
  if (element%in%unicoRB){
    p5enRB <- c(p5enRB, element)
  }
  if (element%in%unicoRC){
    p5enRC <- c(p5enRC, element)
  }
  if (element%in%unicoOX){
    p5enOX <- c(p5enOX, element)
  }
}


# Funciones coincidentes con las Ãºnicas de cada fase, para el punto 6

p6enRC <- c()
p6enRB <- c()
p6enOX <- c()


for (element in GOenriqP6){
  if (element%in%unicoRB){
    p6enRB <- c(p6enRB, element)
  }
  if (element%in%unicoRC){
    p6enRC <- c(p6enRC, element)
  }
  if (element%in%unicoOX){
    p6enOX <- c(p6enOX, element)
  }
}


# Funciones coincidentes con las Ãºnicas de cada fase, para el punto 9

p9enRC <- c()
p9enRB <- c()
p9enOX <- c()
```

```
for (element in GOenriqP9){
  if (element%in%unicoRB){
    p9enRB <- c(p9enRB, element)
  }
  if (element%in%unicoRC){
    p9enRC <- c(p9enRC, element)
  }
  if (element%in%unicoOX){
    p9enOX <- c(p9enOX, element)
  }
}

# Funciones coincidentes con todas de cada fase, para el punto 9
# Hecho sin considerar 9 como parte de RB... Coincide, teniendo el punto 126 GO enriquecidos,
# con 62 de RC, 54 de OX, y 70 de RB
# Siendo de estos Ãºnicos de la fase: 4 de RC, 3 de OX, y 12 de RB)

p9entodoRC <- c()
p9entodoRB <- c()
p9entodoOX <- c()
unicoP9 <- c()

length(GOenriqP9)

for (element in GOenriqP9){
  if (element%in%totalGOPhaseList[['RB']]){
    p9entodoRB <- c(p9entodoRB, element)
  }
  if (element%in%totalGOPhaseList[['RC']]){
    p9entodoRC <- c(p9entodoRC, element)
  }
  if (element%in%totalGOPhaseList[['OX']]){
    p9entodoOX <- c(p9entodoOX, element)
  }else{
```

```
   unicoP9 <- c(unicoP9, element)
  }
}
```

```
# Funciones coincidentes con todas de cada fase, para el punto 2

p2entodoRC <- c()
p2entodoRB <- c()
p2entodoOX <- c()
unicoP2 <- c()

for (element in GOenriqP2){
  if (element%in%totalGOPhaseList[['RB']]){
    p2entodoRB <- c(p2entodoRB, element)
  }
  if (element%in%totalGOPhaseList[['RC']]){
    p2entodoRC <- c(p2entodoRC, element)
  }
  if (element%in%totalGOPhaseList[['OX']]){
    p2entodoOX <- c(p2entodoOX, element)
  }else{
    unicoP2 <- c(unicoP2, element)
  }
}

# GO enriquecidos unicas del punto 2, respecto a todos los otros puntos

zzunicoP2 <- c()
zzzP2 <- c()

for (element in GOenriqP2){
  z <- 0
  for (element1 in totalGOlist){
    z <- z + 1
```

```
   if (z == 2){
     next
   }
   for (element2 in element1){
     if (element == element2){
       zzzP2 <- c(zzzP2, element)
     }else{
       next
     }
   }
  }
  if (element%ni%zzzP2){
   zzunicoP2 <- c(zzunicoP2, element)
  }
}

# Funciones coincidentes con todas de cada fase, para el punto 5

p5entodoRC <- c()
p5entodoRB <- c()
p5entodoOX <- c()
unicoP5 <- c()

for (element in GOenriqP5){
  if (element%in%totalGOPhaseList[['RB']]){
    p5entodoRB <- c(p5entodoRB, element)
  }
  if (element%in%totalGOPhaseList[['RC']]){
    p5entodoRC <- c(p5entodoRC, element)
  }
  if (element%in%totalGOPhaseList[['OX']]){
    p5entodoOX <- c(p5entodoOX, element)
  }else{
    unicoP5 <- c(unicoP5, element)
  }
}
```

```
# GO enriquecidos unicas del punto 5, respecto a todos los otros puntos

zzunicoP5 <- c()
zzzP5 <- c()

for (element in GOenriqP5){
 z <- 0
 for (element1 in totalGOlist){
  z <- z + 1
  if (z == 5){
    next
  }
  for (element2 in element1){
    if (element == element2){
      zzzP5 <- c(zzzP5, element)
    }else{
      next
    }
  }
 }
 if (element%ni%zzzP5){
  zzunicoP5 <- c(zzunicoP5, element)
 }
}


# Funciones coincidentes con todas de cada fase, para el punto 6

p6entodoRC <- c()
p6entodoRB <- c()
p6entodoOX <- c()
unicoP6 <- c()

for (element in GOenriqP6){
 if (element%in%totalGOPhaseList[['RB']]){
  p6entodoRB <- c(p6entodoRB, element)
 }
```

```
  if (element%in%totalGOPhaseList[['RC']]){
    p6entodoRC <- c(p6entodoRC, element)
  }
  if (element%in%totalGOPhaseList[['OX']]){
    p6entodoOX <- c(p6entodoOX, element)
  }else{
    unicoP6 <- c(unicoP6, element)
  }
}


# GO enriquecidos unicas del punto 6, respecto a todos los otros puntos

zzunicoP6 <- c()
zzzP6 <- c()

for (element in GOenriqP6){
  z <- 0
  for (element1 in totalGOlist){
    z <- z + 1
    if (z == 6){
      next
    }
    for (element2 in element1){
      if (element == element2){
        zzzP6 <- c(zzzP6, element)
      }else{
        next
      }
    }
  }
  if (element%ni%zzzP6){
    zzunicoP6 <- c(zzunicoP6, element)
  }
}

# GO enriquecidos unicas del punto 9, respecto a todos los otros puntos
```

```
zzunicoP9 <- c()
zzzP9 <- c()

for (element in GOenriqP9){
 z <- 0
 for (element1 in totalGOlist){
  z <- z + 1
  if (z == 9){
   next
  }
  for (element2 in element1){
   if (element == element2){
    zzzP9 <- c(zzzP9, element)
   }else{
    next
   }
  }
 }
 if (element%ni%zzzP9){
  zzunicoP9 <- c(zzunicoP9, element)
 }
}
#########################
# Guardar el workspace

save.image(file = "workspace-esa1enrichFilt.RData")
```

Enrichment in transcription factors

```
### Definir directorio de trabajo donde estÃ© la lista Yeastract

setwd("/home/biouser/Desktop/FilteredGenes/")


### Cargar lista de genes de levadura con BiomaRt
```

```
library(biomaRt)

ensembl = useMart(host = "jul2018.archive.ensembl.org", biomart = "ENSEMBL_MART_ENSEMBL", dataset = "scerevisiae_gene_ensembl")
#ensembl = useDataset("scerevisiae_gene_ensembl", mart = ensembl)
gene.to.ensembl <- getBM(attributes = c("external_gene_name", "ensembl_gene_id", "description"), mart = ensembl)
gene.to.ensembl <- gene.to.ensembl[!gene.to.ensembl$external_gene_name==",]
rownames(gene.to.ensembl) <- gene.to.ensembl$external_gene_name

save(gene.to.ensembl, file = "gene.to.ensembl.RData")



### RelaciÃ³n TF con gen (con IDs de gen)

TF2gene <- read.table('Yeastract_TF_Gene.tsv')[[1]]

TF2gene.matrix <- matrix(nrow = length(TF2gene), ncol = 2) # Matriz con ID de TF en columna 1, e ID del gen en columna 2
colnames(TF2gene.matrix) <- c('TF', 'gene')

for (n in 1:length(TF2gene)){
  TF2gene.matrix[n, 1] <- gene.to.ensembl[strsplit(as.character(TF2gene[n]), ';')[[1]][1],2]
  TF2gene.matrix[n, 2] <- gene.to.ensembl[strsplit(as.character(TF2gene[n]), ';')[[1]][2],2]
}
rownames(TF2gene.matrix) <- TF2gene.matrix[,'TF']



### Cargar paquetes para trabajar

library(rlist) # Paquete para trabajar con listas

library(data.table) # Para tablas

library(prob) # Para comprobaciones



### Crear lista de TF para cada genes de cada modificador con modificaciones y fase
```

```
## Gcn5 en OX

TF.porgen.Gcn5.OX <- list()
gene.names <- c()

for (gen in overlaps.mods.YMC.filt[["Genes coincidentes entre ambas histonas y un modificador por fases"]][["overlap.ambos.mods.Gcn5.fases"]][["overlap.mods.Gcn5.OX"]]){
  prob1 <- isTRUE(length(TF2gene.matrix[TF2gene.matrix[,2]%in%c(gen), 2]) >= 1)
  if (prob1 == TRUE){
    TF.de.gen <- c()
    objeto <-  TF2gene.matrix[TF2gene.matrix[,2]%in%c(gen), 1]
    for (elemento in objeto){
      prob2 <- is.na(elemento)
      if (prob2 == FALSE){
        TF.de.gen <- c(TF.de.gen, elemento)
      }
    }
    TF.porgen.Gcn5.OX <- list.append(TF.porgen.Gcn5.OX, TF.de.gen)
    gene.names <- c(gene.names, gen)
  }
}

names(TF.porgen.Gcn5.OX) <- gene.names


## Gcn5 en RB

TF.porgen.Gcn5.RB <- list()
gene.names <- c()

for (gen in overlaps.mods.YMC.filt[["Genes coincidentes entre ambas histonas y un modificador por fases"]][["overlap.ambos.mods.Gcn5.fases"]][["overlap.mods.Gcn5.RB"]]){
  prob1 <- isTRUE(length(TF2gene.matrix[TF2gene.matrix[,2]%in%c(gen), 2]) >= 1)
  if (prob1 == TRUE){
    TF.de.gen <- c()
    objeto <-  TF2gene.matrix[TF2gene.matrix[,2]%in%c(gen), 1]
```

```
    for (elemento in objeto){
      prob2 <- is.na(elemento)
      if (prob2 == FALSE){
        TF.de.gen <- c(TF.de.gen, elemento)
      }
    }
    TF.porgen.Gcn5.RB <- list.append(TF.porgen.Gcn5.RB, TF.de.gen)
    gene.names <- c(gene.names, gen)
  }
}

names(TF.porgen.Gcn5.RB) <- gene.names

## Gcn5 en RC

TF.porgen.Gcn5.RC <- list()
gene.names <- c()

for (gen in overlaps.mods.YMC.filt[["Genes coincidentes entre ambas histonas y un
modificador por fases"]][["overlap.ambos.mods.Gcn5.fases"]][["overlap.mods.Gcn5.RC"]]){
  prob1 <- isTRUE(length(TF2gene.matrix[TF2gene.matrix[,2]%in%c(gen), 2]) >= 1)
  if (prob1 == TRUE){
    TF.de.gen <- c()
    objeto <-  TF2gene.matrix[TF2gene.matrix[,2]%in%c(gen), 1]
    for (elemento in objeto){
      prob2 <- is.na(elemento)
      if (prob2 == FALSE){
        TF.de.gen <- c(TF.de.gen, elemento)
      }
    }
    TF.porgen.Gcn5.RC <- list.append(TF.porgen.Gcn5.RC, TF.de.gen)
    gene.names <- c(gene.names, gen)
  }
}

names(TF.porgen.Gcn5.RC) <- gene.names
```

```
## Esa1 en OX

TF.porgen.Esa1.OX <- list()
gene.names <- c()

for (gen in overlaps.mods.YMC.filt[["Genes coincidentes entre ambas histonas y un modificador por fases"]][["overlap.ambos.mods.Esa1.fases"]][["overlap.mods.Esa1.OX"]]){
  prob1 <- isTRUE(length(TF2gene.matrix[TF2gene.matrix[,2]%in%c(gen), 2]) >= 1)
  if (prob1 == TRUE){
    TF.de.gen <- c()
    objeto <-  TF2gene.matrix[TF2gene.matrix[,2]%in%c(gen), 1]
    for (elemento in objeto){
      prob2 <- is.na(elemento)
      if (prob2 == FALSE){
        TF.de.gen <- c(TF.de.gen, elemento)
      }
    }
    TF.porgen.Esa1.OX <- list.append(TF.porgen.Esa1.OX, TF.de.gen)
    gene.names <- c(gene.names, gen)
  }
}

names(TF.porgen.Esa1.OX) <- gene.names


## Esa1 en RB

TF.porgen.Esa1.RB <- list()
gene.names <- c()

for (gen in overlaps.mods.YMC.filt[["Genes coincidentes entre ambas histonas y un modificador por fases"]][["overlap.ambos.mods.Esa1.fases"]][["overlap.mods.Esa1.RB"]]){
  prob1 <- isTRUE(length(TF2gene.matrix[TF2gene.matrix[,2]%in%c(gen), 2]) >= 1)
  if (prob1 == TRUE){
    TF.de.gen <- c()
    objeto <-  TF2gene.matrix[TF2gene.matrix[,2]%in%c(gen), 1]
    for (elemento in objeto){
```

```
      prob2 <- is.na(elemento)
      if (prob2 == FALSE){
        TF.de.gen <- c(TF.de.gen, elemento)
      }
    }
    TF.porgen.Esa1.RB <- list.append(TF.porgen.Esa1.RB, TF.de.gen)
    gene.names <- c(gene.names, gen)
  }
}

names(TF.porgen.Esa1.RB) <- gene.names

## Esa1 en RC

TF.porgen.Esa1.RC <- list()
gene.names <- c()

for (gen in overlaps.mods.YMC.filt[["Genes coincidentes entre ambas histonas y un
modificador por fases"]][["overlap.ambos.mods.Esa1.fases"]][["overlap.mods.Esa1.RC"]]){
  prob1 <- isTRUE(length(TF2gene.matrix[TF2gene.matrix[,2]%in%c(gen), 2]) >= 1)
  if (prob1 == TRUE){
    TF.de.gen <- c()
    objeto <-  TF2gene.matrix[TF2gene.matrix[,2]%in%c(gen), 1]
    for (elemento in objeto){
      prob2 <- is.na(elemento)
      if (prob2 == FALSE){
        TF.de.gen <- c(TF.de.gen, elemento)
      }
    }
    TF.porgen.Esa1.RC <- list.append(TF.porgen.Esa1.RC, TF.de.gen)
    gene.names <- c(gene.names, gen)
  }
}

names(TF.porgen.Esa1.RC) <- gene.names

## Set1 en OX
```

```
TF.porgen.Set1.OX <- list()
gene.names <- c()

for (gen in overlaps.mods.YMC.filt[["Genes coincidentes entre ambas histonas y un modificador por fases"]][["overlap.ambos.mods.Set1.fases"]][["overlap.mods.Set1.OX"]]){
  prob1 <- isTRUE(length(TF2gene.matrix[TF2gene.matrix[,2]%in%c(gen), 2]) >= 1)
  if (prob1 == TRUE){
    TF.de.gen <- c()
    objeto <-  TF2gene.matrix[TF2gene.matrix[,2]%in%c(gen), 1]
    for (elemento in objeto){
      prob2 <- is.na(elemento)
      if (prob2 == FALSE){
        TF.de.gen <- c(TF.de.gen, elemento)
      }
    }
    TF.porgen.Set1.OX <- list.append(TF.porgen.Set1.OX, TF.de.gen)
    gene.names <- c(gene.names, gen)
  }
}

names(TF.porgen.Set1.OX) <- gene.names



## Set1 en RB

TF.porgen.Set1.RB <- list()
gene.names <- c()

for (gen in overlaps.mods.YMC.filt[["Genes coincidentes entre ambas histonas y un modificador por fases"]][["overlap.ambos.mods.Set1.fases"]][["overlap.mods.Set1.RB"]]){
  prob1 <- isTRUE(length(TF2gene.matrix[TF2gene.matrix[,2]%in%c(gen), 2]) >= 1)
  if (prob1 == TRUE){
    TF.de.gen <- c()
    objeto <-  TF2gene.matrix[TF2gene.matrix[,2]%in%c(gen), 1]
    for (elemento in objeto){
      prob2 <- is.na(elemento)
```

```
      if (prob2 == FALSE){
        TF.de.gen <- c(TF.de.gen, elemento)
      }
    }
    TF.porgen.Set1.RB <- list.append(TF.porgen.Set1.RB, TF.de.gen)
    gene.names <- c(gene.names, gen)
  }
}

names(TF.porgen.Set1.RB) <- gene.names

## Set1 en RC

TF.porgen.Set1.RC <- list()
gene.names <- c()

for (gen in overlaps.mods.YMC.filt[["Genes coincidentes entre ambas histonas y un
modificador por fases"]][["overlap.ambos.mods.Set1.fases"]][["overlap.mods.Set1.RC"]]){
  prob1 <- isTRUE(length(TF2gene.matrix[TF2gene.matrix[,2]%in%c(gen), 2]) >= 1)
  if (prob1 == TRUE){
    TF.de.gen <- c()
    objeto <-  TF2gene.matrix[TF2gene.matrix[,2]%in%c(gen), 1]
    for (elemento in objeto){
      prob2 <- is.na(elemento)
      if (prob2 == FALSE){
        TF.de.gen <- c(TF.de.gen, elemento)
      }
    }
    TF.porgen.Set1.RC <- list.append(TF.porgen.Set1.RC, TF.de.gen)
    gene.names <- c(gene.names, gen)
  }
}

names(TF.porgen.Set1.RC) <- gene.names


#### Hacer lista con todos los TF de la muestra
```

```
TodosTFMuestra <- c()
TF.Esa1.OX <- c()
TF.Esa1.RB <- c()
TF.Esa1.RC <- c()
TF.Gcn5.OX <- c()
TF.Gcn5.RB <- c()
TF.Gcn5.RC <- c()
TF.Set1.OX <- c()
TF.Set1.RB <- c()
TF.Set1.RC <- c()

TF.muestra.Mod.Fase.porgen <- list("TF.porgen.Esa1.OX" = TF.porgen.Esa1.OX,
                        "TF.porgen.Esa1.RB" = TF.porgen.Esa1.RB,
                        "TF.porgen.Esa1.RC" = TF.porgen.Esa1.RC,
                        "TF.porgen.Gcn5.OX" = TF.porgen.Gcn5.OX,
                        "TF.porgen.Gcn5.RB" = TF.porgen.Gcn5.RB,
                        "TF.porgen.Gcn5.RC" = TF.porgen.Gcn5.RC,
                        "TF.porgen.Set1.OX" = TF.porgen.Set1.OX,
                        "TF.porgen.Set1.RB" = TF.porgen.Set1.RB,
                        "TF.porgen.Set1.RC" = TF.porgen.Set1.RC)



for (list in TF.muestra.Mod.Fase.porgen){
  for (element in list){
    for (elem in element){
      TodosTFMuestra <- c(TodosTFMuestra, elem)
    }
  }
}

TodosTFMuestra <- sort(TodosTFMuestra)
TodosTFMuestra <- unique(TodosTFMuestra)

## Y listas de TF para cada fase & enzima
```

```
# Esa1

for (list in TF.porgen.Esa1.OX){
  for (element in list){
    TF.Esa1.OX <- c(TF.Esa1.OX, element)
  }
}

TF.Esa1.OX <- sort(TF.Esa1.OX)
TF.Esa1.OX <- unique(TF.Esa1.OX)


for (list in TF.porgen.Esa1.RB){
  for (element in list){
    TF.Esa1.RB <- c(TF.Esa1.RB, element)
  }
}

TF.Esa1.RB <- sort(TF.Esa1.RB)
TF.Esa1.RB <- unique(TF.Esa1.RB)


for (list in TF.porgen.Esa1.RC){
  for (element in list){
    TF.Esa1.RC <- c(TF.Esa1.RC, element)
  }
}

TF.Esa1.RC <- sort(TF.Esa1.RC)
TF.Esa1.RC <- unique(TF.Esa1.RC)

# Gcn5

for (list in TF.porgen.Gcn5.OX){
  for (element in list){
    TF.Gcn5.OX <- c(TF.Gcn5.OX, element)
  }
```

```
}

TF.Gcn5.OX <- sort(TF.Gcn5.OX)
TF.Gcn5.OX <- unique(TF.Gcn5.OX)


for (list in TF.porgen.Gcn5.RB){
  for (element in list){
    TF.Gcn5.RB <- c(TF.Gcn5.RB, element)
  }
}

TF.Gcn5.RB <- sort(TF.Gcn5.RB)
TF.Gcn5.RB <- unique(TF.Gcn5.RB)


for (list in TF.porgen.Gcn5.RC){
  for (element in list){
    TF.Gcn5.RC <- c(TF.Gcn5.RC, element)
  }
}

TF.Gcn5.RC <- sort(TF.Gcn5.RC)
TF.Gcn5.RC <- unique(TF.Gcn5.RC)



# Set1

for (list in TF.porgen.Set1.OX){
  for (element in list){
    TF.Set1.OX <- c(TF.Set1.OX, element)
  }
}

TF.Set1.OX <- sort(TF.Set1.OX)
TF.Set1.OX <- unique(TF.Set1.OX)
```

```
for (list in TF.porgen.Set1.RB){
  for (element in list){
    TF.Set1.RB <- c(TF.Set1.RB, element)
  }
}

TF.Set1.RB <- sort(TF.Set1.RB)
TF.Set1.RB <- unique(TF.Set1.RB)


for (list in TF.porgen.Set1.RC){
  for (element in list){
    TF.Set1.RC <- c(TF.Set1.RC, element)
  }
}

TF.Set1.RC <- sort(TF.Set1.RC)
TF.Set1.RC <- unique(TF.Set1.RC)


TF.muestra.Mod.Fase <- list("TF.Esa1.OX" = TF.Esa1.OX,
                            "TF.Esa1.RB" = TF.Esa1.RB,
                            "TF.Esa1.RC" = TF.Esa1.RC,
                            "TF.Gcn5.OX" = TF.Gcn5.OX,
                            "TF.Gcn5.RB" = TF.Gcn5.RB,
                            "TF.Gcn5.RC" = TF.Gcn5.RC,
                            "TF.Set1.OX" = TF.Set1.OX,
                            "TF.Set1.RB" = TF.Set1.RB,
                            "TF.Set1.RC" = TF.Set1.RC)

# Guardar

save(TF.muestra.Mod.Fase.porgen, file = "TF.muestra.Mod.Fase.porgen.filt.RData")
save(TF.muestra.Mod.Fase, file = "TF.muestra.Mod.Fase.filt.RData")
```

```
#### Funciones para hacer el test de Fisher

# Para enriquecimiento

EnrichALLterms = function (test, notTest, annotation, extGenName, sample, phase,
p.adjust.method = "fdr") {


  cuasiannot2test = unique(annotation[,1])
  annot2test = c()
  for (element in cuasiannot2test){
    ques <- is.na(element)
    if (ques == F){
      annot2test = c(annot2test, element)
    }
  }

  resultat = t(sapply(annot2test, Enrich1term, test = test, notTest = notTest, annotation =
annotation, extGenName = extGenName, sample = sample, phase = phase))
  return (data.frame(resultat[,1:7],
                     "pval" = as.numeric(resultat[,"pval"]),
                     "adjPval" = p.adjust(as.numeric(resultat[,"pval"]), method = p.adjust.method),
                     "Sample" = resultat[,"Sample"],
                     "Phase" = resultat[,"Phase"],
                     stringsAsFactors = F))

}

Enrich1term = function (term, test, notTest, annotation, extGenName, sample, phase) {


  annotTest = length(intersect(test, annotation[annotation[,1] == term,2]))

  if ((annotTest) > 0) {
    annotNOTtest = length(intersect(notTest, annotation[annotation[,1] == term,2]))
    extName = extGenName[extGenName[,2] == term, 1]
```

```
    descrip = extGenName[extGenName[,2] == term, 3]
    mytest = matrix(c(annotTest, length(test)-annotTest, annotNOTtest, length(notTest)-
annotNOTtest), ncol = 2)
    resultat = c(term, extName, descrip, annotTest, length(test), annotNOTtest, length(notTest),
            fisher.test(mytest, alternative = "greater")$p.value, sample, phase)
    names(resultat) = c("Gene", "extName", "Description", "annotTest", "test", "annotNotTest",
"notTest", "pval", "Sample", "Phase")
  } else {
    extName = extGenName[extGenName[,2] == term, 1]
    descrip = extGenName[extGenName[,2] == term, 3]
    resultat = c(term, extName, descrip, 0, 0, 0, 0, 100, sample, phase)
    names(resultat) = c("Gene", "extName", "Description", "annotTest", "test", "annotNotTest",
"notTest", "pval", "Sample", "Phase")
  }

  return(resultat)

}

#### Hacer test de Fisher para todos los TFs relevantes

### Anotacion (lista de todos los genes (IDs de Ensembl) con sus TFs relacionados)

annotation <- as.data.frame(TF2gene.matrix)

### External Gene Names

extGenName <- gene.to.ensembl

### Esa1

## OX
```

```
  test <- overlaps.mods.YMC.filt[["Genes coincidentes entre ambas histonas y un modificador por fases"]][["overlap.ambos.mods.Esa1.fases"]][["overlap.mods.Esa1.OX"]]

  cuasiNotTest <- genes.YMC.Cluster[["Fase OX"]]

  cuasi <- intersect(test, cuasiNotTest)

  notTest <- setdiff(cuasiNotTest, cuasi)

  TF.enriq.Esa1.OX <- EnrichALLterms(test = test,
                                     notTest = notTest,
                                     annotation = annotation,
                                     extGenName = extGenName,
                                     sample = "Esa1",
                                     phase = "OX")

  ## RB

  test <- overlaps.mods.YMC.filt[["Genes coincidentes entre ambas histonas y un modificador por fases"]][["overlap.ambos.mods.Esa1.fases"]][["overlap.mods.Esa1.RB"]]

  cuasiNotTest <- genes.YMC.Cluster[["Fase RB"]]

  cuasi <- intersect(test, cuasiNotTest)

  notTest <- setdiff(cuasiNotTest, cuasi)

  TF.enriq.Esa1.RB <- EnrichALLterms(test = test,
                                     notTest = notTest,
                                     annotation = annotation,
                                     extGenName = extGenName,
                                     sample = "Esa1",
                                     phase = "RB")

  ## RC

  test <- overlaps.mods.YMC.filt[["Genes coincidentes entre ambas histonas y un modificador por fases"]][["overlap.ambos.mods.Esa1.fases"]][["overlap.mods.Esa1.RC"]]
```

```
cuasiNotTest <- genes.YMC.Cluster[["Fase RC"]]

cuasi <- intersect(test, cuasiNotTest)

notTest <- setdiff(cuasiNotTest, cuasi)

TF.enriq.Esa1.RC <- EnrichALLterms(test = test,
                                   notTest = notTest,
                                   annotation = annotation,
                                   extGenName = extGenName,
                                   sample = "Esa1",
                                   phase = "RC")

### Gcn5

## OX

test <- overlaps.mods.YMC.filt[["Genes coincidentes entre ambas histonas y un modificador
por fases"]][["overlap.ambos.mods.Gcn5.fases"]][["overlap.mods.Gcn5.OX"]]

cuasiNotTest <- genes.YMC.Cluster[["Fase OX"]]

cuasi <- intersect(test, cuasiNotTest)

notTest <- setdiff(cuasiNotTest, cuasi)

TF.enriq.Gcn5.OX <- EnrichALLterms(test = test,
                                   notTest = notTest,
                                   annotation = annotation,
                                   extGenName = extGenName,
                                   sample = "Gcn5",
                                   phase = "OX")

## RB
```

```
test <- overlaps.mods.YMC.filt[["Genes coincidentes entre ambas histonas y un modificador por fases"]][["overlap.ambos.mods.Gcn5.fases"]][["overlap.mods.Gcn5.RB"]]

cuasiNotTest <- genes.YMC.Cluster[["Fase RB"]]

cuasi <- intersect(test, cuasiNotTest)

notTest <- setdiff(cuasiNotTest, cuasi)

TF.enriq.Gcn5.RB <- EnrichALLterms(test = test,
                                   notTest = notTest,
                                   annotation = annotation,
                                   extGenName = extGenName,
                                   sample = "Gcn5",
                                   phase = "RB")

## RC

test <- overlaps.mods.YMC.filt[["Genes coincidentes entre ambas histonas y un modificador por fases"]][["overlap.ambos.mods.Gcn5.fases"]][["overlap.mods.Gcn5.RC"]]

cuasiNotTest <- genes.YMC.Cluster[["Fase RC"]]

cuasi <- intersect(test, cuasiNotTest)

notTest <- setdiff(cuasiNotTest, cuasi)

TF.enriq.Gcn5.RC <- EnrichALLterms(test = test,
                                   notTest = notTest,
                                   annotation = annotation,
                                   extGenName = extGenName,
                                   sample = "Gcn5",
                                   phase = "RC")

### Set1

## OX
```

```
test <- overlaps.mods.YMC.filt[["Genes coincidentes entre ambas histonas y un modificador por fases"]][["overlap.ambos.mods.Set1.fases"]][["overlap.mods.Set1.OX"]]

cuasiNotTest <- genes.YMC.Cluster[["Fase OX"]]

cuasi <- intersect(test, cuasiNotTest)

notTest <- setdiff(cuasiNotTest, cuasi)

TF.enriq.Set1.OX <- EnrichALLterms(test = test,
                                   notTest = notTest,
                                   annotation = annotation,
                                   extGenName = extGenName,
                                   sample = "Set1",
                                   phase = "OX")

## RB

test <- overlaps.mods.YMC.filt[["Genes coincidentes entre ambas histonas y un modificador por fases"]][["overlap.ambos.mods.Set1.fases"]][["overlap.mods.Set1.RB"]]

cuasiNotTest <- genes.YMC.Cluster[["Fase RB"]]

cuasi <- intersect(test, cuasiNotTest)

notTest <- setdiff(cuasiNotTest, cuasi)

TF.enriq.Set1.RB <- EnrichALLterms(test = test,
                                   notTest = notTest,
                                   annotation = annotation,
                                   extGenName = extGenName,
                                   sample = "Set1",
                                   phase = "RB")

## RC
```

```
test <- overlaps.mods.YMC.filt.filt[["Genes coincidentes entre ambas histonas y un modificador por fases"]][["overlap.ambos.mods.Set1.fases"]][["overlap.mods.Set1.RC"]]

cuasiNotTest <- genes.YMC.Cluster[["Fase RC"]]

cuasi <- intersect(test, cuasiNotTest)

notTest <- setdiff(cuasiNotTest, cuasi)

TF.enriq.Set1.RC <- EnrichALLterms(test = test,
                                   notTest = notTest,
                                   annotation = annotation,
                                   extGenName = extGenName,
                                   sample = "Set1",
                                   phase = "RC")



#### Guardar los resultados de enriquecimiento de TFs

Tf.enriquecidos.Mod.Fase.clu <- list("TF.enriq.Esa1.OX" = TF.enriq.Esa1.OX,
                                     "TF.enriq.Esa1.RB" = TF.enriq.Esa1.RB,
                                     "TF.enriq.Esa1.RC" = TF.enriq.Esa1.RC,
                                     "TF.enriq.Gcn5.OX" = TF.enriq.Gcn5.OX,
                                     "TF.enriq.Gcn5.RB" = TF.enriq.Gcn5.RB,
                                     "TF.enriq.Gcn5.RC" = TF.enriq.Gcn5.RC,
                                     "TF.enriq.Set1.OX" = TF.enriq.Set1.OX,
                                     "TF.enriq.Set1.RB" = TF.enriq.Set1.RB,
                                     "TF.enriq.Set1.RC" = TF.enriq.Set1.RC)

save(Tf.enriquecidos.Mod.Fase.clu, file = "TF.enriquecidos.Mod.Fase.clu.filt.RData")



#### Pasar los datos a Excel

### Montar data frame
### Debe contener los 10 TF mas significativos de cada conjunto
```

```
Top.TFs <- data.frame()

for (conjunto in Tf.enriquecidos.Mod.Fase.clu){

  a1 <- conjunto[order(conjunto$pval),]
  a2 <- a1[a1$pval<0.05,]
  Top10.TFs <- rbind(Top10.TFs, a2)
}

### Guardar Data Frame como excel

library("xlsx")

write.xlsx(Top.TFs, file = "TopTfsFiltVsCluster.xlsx", sheetName = "Top90TF",
          col.names = TRUE, row.names = TRUE, append = FALSE)

#### Hacer Venn Diagram con TF en los tres conjuntos por fase

library(VennDiagram)

VennDiag3Df = function(df1, #data frame 1
                       nam1, # nombre de conjunto 1
                       df2, # data frame 2
                       nam2, # nombre de conjunto 2
                       df3, #data frame 3
                       nam3, # nombre de conjunto 3
                       colnum) # numero de columna del data frame para extraer e interseccionar
datos

  {

  set1 = c()
  set2 = c()
  set3 = c()

  intersect12 = c()
  intersect23 = c()
```

```
intersect13 = c()

intersect123 = c()

for (gene in df1[, colnum]){
  set1 <- c(set1, gene)
}

for (gene in df2[, colnum]){
  set2 <- c(set2, gene)
}

for (gene in df3[, colnum]){
  set3 <- c(set3, gene)
}

intersect23 <- intersect(set2, set3)

intersect13 <- intersect(set1, set3)

intersect12 <- intersect(set1, set2)

intersect123 <- intersect(set1, intersect(set2, set3))


diag = draw.triple.venn(area1 = length(set1),
                        area2 = length(set2),
                        area3 = length(set3),
                        n12 = length(intersect12),
                        n23 = length(intersect23),
                        n13 = length(intersect13),
                        n123 = length(intersect123),
                        category = c(nam1, nam2, nam3),
                        fill = c("darkorange", "blue1", "chartreuse3"))

return(diag)
```

```
  }

  Intersect3Df = function(df1, #data frame 1
                          nam1, # nombre de conjunto 1
                          df2, # data frame 2
                          nam2, # nombre de conjunto 2
                          df3, #data frame 3
                          nam3, # nombre de conjunto 3
                          colnum) # numero de columna del data frame para extraer e interseccionar
datos

  {

    set1 = c()
    set2 = c()
    set3 = c()

    intersect12 = c()
    intersect23 = c()
    intersect13 = c()

    intersect123 = c()

    for (gene in df1[, colnum]){
      set1 <- c(set1, gene)
    }

    for (gene in df2[, colnum]){
      set2 <- c(set2, gene)
    }

    for (gene in df3[, colnum]){
      set3 <- c(set3, gene)
    }

    intersect23 <- intersect(set2, set3)
```

```
  intersect13 <- intersect(set1, set3)

  intersect12 <- intersect(set1, set2)

  intersect123 <- intersect(set1, intersect(set2, set3))

  return(intersect123)

}

### OX

TF.enriq.Esa1.OX.signif <- Tf.enriquecidos.Mod.Fase.clu[["TF.enriq.Esa1.OX"]]
TF.enriq.Esa1.OX.signif <- TF.enriq.Esa1.OX.signif[order(TF.enriq.Esa1.OX.signif$pval),]
TF.enriq.Esa1.OX.signif <- TF.enriq.Esa1.OX.signif[TF.enriq.Esa1.OX.signif$pval<0.05,]

TF.enriq.Gcn5.OX.signif <- Tf.enriquecidos.Mod.Fase.clu[["TF.enriq.Gcn5.OX"]]
TF.enriq.Gcn5.OX.signif <- TF.enriq.Gcn5.OX.signif[order(TF.enriq.Gcn5.OX.signif$pval),]
TF.enriq.Gcn5.OX.signif <- TF.enriq.Gcn5.OX.signif[TF.enriq.Gcn5.OX.signif$pval<0.05,]

TF.enriq.Set1.OX.signif <- Tf.enriquecidos.Mod.Fase.clu[["TF.enriq.Set1.OX"]]
TF.enriq.Set1.OX.signif <- TF.enriq.Set1.OX.signif[order(TF.enriq.Set1.OX.signif$pval),]
TF.enriq.Set1.OX.signif <- TF.enriq.Set1.OX.signif[TF.enriq.Set1.OX.signif$pval<0.05,]

VennDiag3Df(df3 = TF.enriq.Esa1.OX.signif,
            nam3 = "Esa1",
            df1 = TF.enriq.Gcn5.OX.signif,
            nam1 = "Gcn5",
            df2 = TF.enriq.Set1.OX.signif,
            nam2 = "Set1",
            colnum = 1)

TF.enriq.OX.comunes  <- Intersect3Df(df3 = TF.enriq.Esa1.OX.signif,
            nam3 = "Esa1-H3k18-H3k9",
            df1 = TF.enriq.Gcn5.OX.signif,
            nam1 = "Gcn5-H3k18-H3k9",
            df2 = TF.enriq.Set1.OX.signif,
```

```
                nam2 = "Set1-H3k18-H3k9",
                colnum = 1)


  aaa1 <- Tf.enriquecidos.Mod.Fase.clu[["TF.enriq.Esa1.OX"]]

  TF.enriq.OX.comunes.df <- aaa1[aaa1$Gene%in%TF.enriq.OX.comunes, 1:3]

  write.xlsx(TF.enriq.OX.comunes.df, file = "TF.enriq.OX.comunes.df.FILT.xlsx", sheetName
= "TF.enriq.OX.comunes.df",
          col.names = TRUE, row.names = FALSE, append = FALSE)

  ### RB

  TF.enriq.Esa1.RB.signif <- Tf.enriquecidos.Mod.Fase.clu[["TF.enriq.Esa1.RB"]]
  TF.enriq.Esa1.RB.signif <- TF.enriq.Esa1.RB.signif[order(TF.enriq.Esa1.RB.signif$pval),]
  TF.enriq.Esa1.RB.signif <- TF.enriq.Esa1.RB.signif[TF.enriq.Esa1.RB.signif$pval<0.05,]

  TF.enriq.Gcn5.RB.signif <- Tf.enriquecidos.Mod.Fase.clu[["TF.enriq.Gcn5.RB"]]
  TF.enriq.Gcn5.RB.signif <- TF.enriq.Gcn5.RB.signif[order(TF.enriq.Gcn5.RB.signif$pval),]
  TF.enriq.Gcn5.RB.signif <- TF.enriq.Gcn5.RB.signif[TF.enriq.Gcn5.RB.signif$pval<0.05,]

  TF.enriq.Set1.RB.signif <- Tf.enriquecidos.Mod.Fase.clu[["TF.enriq.Set1.RB"]]
  TF.enriq.Set1.RB.signif <- TF.enriq.Set1.RB.signif[order(TF.enriq.Set1.RB.signif$pval),]
  TF.enriq.Set1.RB.signif <- TF.enriq.Set1.RB.signif[TF.enriq.Set1.RB.signif$pval<0.05,]


  VennDiag3Df(df3 = TF.enriq.Esa1.RB.signif,
          nam3 = "Esa1",
          df1 = TF.enriq.Gcn5.RB.signif,
          nam1 = "Gcn5",
          df2 = TF.enriq.Set1.RB.signif,
          nam2 = "Set1",
          colnum = 1)

  TF.enriq.RB.comunes  <- Intersect3Df(df3 = TF.enriq.Esa1.RB.signif,
                          nam3 = "Esa1-H3k18-H3k9",
```

```
                df1 = TF.enriq.Gcn5.RB.signif,
                nam1 = "Gcn5-H3k18-H3k9",
                df2 = TF.enriq.Set1.RB.signif,
                nam2 = "Set1-H3k18-H3k9",
                colnum = 1)

aaa2 <- Tf.enriquecidos.Mod.Fase.clu[["TF.enriq.Esa1.RB"]]

TF.enriq.RB.comunes.df <- aaa2[aaa2$Gene%in%TF.enriq.RB.comunes, 1:3]

write.xlsx(TF.enriq.RB.comunes.df, file = "TF.enriq.RB.comunes.df.FILT.xlsx", sheetName
= "TF.enriq.RB.comunes.df",
        col.names = TRUE, row.names = FALSE, append = FALSE)


### RC

TF.enriq.Esa1.RC.signif <- Tf.enriquecidos.Mod.Fase.clu[["TF.enriq.Esa1.RC"]]
TF.enriq.Esa1.RC.signif <- TF.enriq.Esa1.RC.signif[order(TF.enriq.Esa1.RC.signif$pval),]
TF.enriq.Esa1.RC.signif <- TF.enriq.Esa1.RC.signif[TF.enriq.Esa1.RC.signif$pval<0.05,]

TF.enriq.Gcn5.RC.signif <- Tf.enriquecidos.Mod.Fase.clu[["TF.enriq.Gcn5.RC"]]
TF.enriq.Gcn5.RC.signif <- TF.enriq.Gcn5.RC.signif[order(TF.enriq.Gcn5.RC.signif$pval),]
TF.enriq.Gcn5.RC.signif <- TF.enriq.Gcn5.RC.signif[TF.enriq.Gcn5.RC.signif$pval<0.05,]

TF.enriq.Set1.RC.signif <- Tf.enriquecidos.Mod.Fase.clu[["TF.enriq.Set1.RC"]]
TF.enriq.Set1.RC.signif <- TF.enriq.Set1.RC.signif[order(TF.enriq.Set1.RC.signif$pval),]
TF.enriq.Set1.RC.signif <- TF.enriq.Set1.RC.signif[TF.enriq.Set1.RC.signif$pval<0.05,]


VennDiag3Df(df3 = TF.enriq.Esa1.RC.signif,
        nam3 = "Esa1",
        df1 = TF.enriq.Gcn5.RC.signif,
        nam1 = "Gcn5",
        df2 = TF.enriq.Set1.RC.signif,
        nam2 = "Set1",
        colnum = 1)
```

```
TF.enriq.RC.comunes  <- Intersect3Df(df3 = TF.enriq.Esa1.RC.signif,
                                     nam3 = "Esa1-H3k18-H3k9",
                                     df1 = TF.enriq.Gcn5.RC.signif,
                                     nam1 = "Gcn5-H3k18-H3k9",
                                     df2 = TF.enriq.Set1.RC.signif,
                                     nam2 = "Set1-H3k18-H3k9",
                                     colnum = 1)

aaa3 <- Tf.enriquecidos.Mod.Fase.clu[["TF.enriq.Esa1.RC"]]

TF.enriq.RC.comunes.df <- aaa3[aaa3$Gene%in%TF.enriq.RC.comunes, 1:3]

write.xlsx(TF.enriq.RC.comunes.df, file = "TF.enriq.RC.comunes.df.FILT.xlsx", sheetName
= "TF.enriq.RC.comunes.df",
          col.names = TRUE, row.names = FALSE, append = FALSE)


### Guardar datos

TF.enriq.comunes.fases.clu <- list("TF.enriq.OX.comunes" = TF.enriq.OX.comunes,
                                   "TF.enriq.RB.comunes" = TF.enriq.RB.comunes,
                                   "TF.enriq.RC.comunes" = TF.enriq.RB.comunes)

save(TF.enriq.comunes.fases.clu, file = "TF.enriq.comunes.fases.clu.filt.RData")


#### Hacer Venn diagram entre los genes coincidentes de las tres fases


VennDiag3Df(df1 = TF.enriq.OX.comunes.df,
          nam1 = "OX",
          df2 = TF.enriq.RB.comunes.df,
          nam2 = "RB",
          df3 = TF.enriq.RC.comunes.df,
          nam3 = "RC",
          colnum = 1)
```

```
TF.enriq.comunes.todasFases  <- Intersect3Df(df1 = TF.enriq.OX.comunes.df,
                              nam1 = "OX",
                              df2 = TF.enriq.RB.comunes.df,
                              nam2 = "RB",
                              df3 = TF.enriq.RC.comunes.df,
                              nam3 = "RC",
                              colnum = 1)



  ######################
  ## Sacar TFs coincidentes entre
  ## Esa1 y Gcn5 solo
  ######################

  ### Funciones para Pairwise Venn desde data frame

  VennDiag2Df = function(df1, #data frame 1
                         nam1, # nombre de conjunto 1
                         df2, # data frame 2
                         nam2, # nombre de conjunto 2
                         colnum) # numero de columna del data frame para extraer e interseccionar
datos

  {

    set1 = c()
    set2 = c()

    intersect12 = c()

    for (gene in df1[, colnum]){
      set1 <- c(set1, gene)
    }

    for (gene in df2[, colnum]){
      set2 <- c(set2, gene)
```

```
  }

  intersect12 <- intersect(set1, set2)

  diag = draw.pairwise.venn(area1 = length(set1),
                    area2 = length(set2),
                    cross.area = length(intersect12),
                    category = c(nam1, nam2),
                    fill = c("red", "green"))

  return(diag)

 }

 Intersect2Df = function(df1, #data frame 1
                  df2, # data frame 2
                  colnum) # numero de columna del data frame para extraer e interseccionar
datos

 {

  set1 = c()
  set2 = c()

  intersect12 = c()

  for (gene in df1[, colnum]){
   set1 <- c(set1, gene)
  }

  for (gene in df2[, colnum]){
   set2 <- c(set2, gene)
  }

  intersect12 <- intersect(set1, set2)

  return(intersect12)
```

```
}


# Fase OX

area1 <- TF.enriq.Gcn5.OX.signif
area2 <- TF.enriq.Esa1.OX.signif

VennDiag2Df(df1 = area1,
        nam1 = "Gcn5",
        df2 = area2,
        nam2 = "Esa1",
        colnum = 1)

com.tf.ox.gcn5.esa1.acet <- Intersect2Df(df1 = area1,
                                df2 = area2,
                                colnum = 1)



# Fase RB

area1 <- TF.enriq.Gcn5.RB.signif
area2 <- TF.enriq.Esa1.RB.signif

VennDiag2Df(df1 = area1,
        nam1 = "Gcn5",
        df2 = area2,
        nam2 = "Esa1",
        colnum = 1)

com.tf.rb.gcn5.esa1.acet <- Intersect2Df(df1 = area1,
                                df2 = area2,
                                colnum = 1)

# Fase RC
```

```
area1 <- TF.enriq.Gcn5.RC.signif
area2 <- TF.enriq.Esa1.RC.signif

VennDiag2Df(df1 = area1,
        nam1 = "Gcn5",
        df2 = area2,
        nam2 = "Esa1",
        colnum = 1)

com.tf.rc.gcn5.esa1.acet <- Intersect2Df(df1 = area1,
                                df2 = area2,
                                colnum = 1)


### Coincidencia entre fases

com.tf.todasfases.gcn5.esa1.acet <- intersect(com.tf.ox.gcn5.esa1.acet,
intersect(com.tf.rb.gcn5.esa1.acet, com.tf.rc.gcn5.esa1.acet))

draw.triple.venn(area1 = length(com.tf.ox.gcn5.esa1.acet),
            area2 = length(com.tf.rb.gcn5.esa1.acet),
            area3 = length(com.tf.rc.gcn5.esa1.acet),
            n12 = length(intersect(com.tf.ox.gcn5.esa1.acet, com.tf.rb.gcn5.esa1.acet)),
            n23 = length(intersect(com.tf.rb.gcn5.esa1.acet, com.tf.rc.gcn5.esa1.acet)),
            n13 = length(intersect(com.tf.ox.gcn5.esa1.acet, com.tf.rc.gcn5.esa1.acet)),
            n123 = length(intersect(com.tf.ox.gcn5.esa1.acet,
intersect(com.tf.rb.gcn5.esa1.acet, com.tf.rc.gcn5.esa1.acet))),
            category = c("OX", "RB", "RC"),
            fill = c("red", "green", "blue"))

com.tf.oxrc.gcn5.esa1.acet <- intersect(com.tf.ox.gcn5.esa1.acet, com.tf.rc.gcn5.esa1.acet)

Tf.com.oxrc.Gcn5.Esa1.2Acet.df <- Tf.enriquecidos.Mod.Fase.clu[["TF.enriq.Esa1.RC"]]
Tf.com.oxrc.Gcn5.Esa1.2Acet.df <-
Tf.com.oxrc.Gcn5.Esa1.2Acet.df[Tf.com.oxrc.Gcn5.Esa1.2Acet.df$Gene%in%c(com.tf.oxrc.gc
n5.esa1.acet),]
write.xlsx(Tf.com.oxrc.Gcn5.Esa1.2Acet.df, file = "Tf.com.oxrc.Gcn5.Esa1.2Acet.xlsx",
sheetName = "Tf.com.oxrc.Gcn5.Esa1.2Acet",
```

```
col.names = TRUE, row.names = FALSE, append = FALSE)
```

Read count matrixes construction

```
### Colapsar todos los picos cercanos de una regiÃ³n

## Cargar paquete de colapso

library(GenomicRanges)

## Esa1

# Indicar carpeta con los narrowPeak files (resultado del peak calling)

setwd("/home/biouser/Desktop/Cluster/YMC/denovo/PeakCall/Esa1/")

# Hacer data frame con los archivos de los pcios

peak.files <- list.files(pattern = "narrowPeak")
my_df <- data.frame()

for (file in peak.files){
  print(file)
  a <- read.table(file)
  colnames(a) <- c("chr", "start", "end", "peak", "score", "strand", "signalValue", "pValue",
"qValue")
  my_df <- rbind.data.frame(my_df, a)
}

# "Fusionar" regiones con picos en posiciones unicas

my_gr <- makeGRangesFromDataFrame(my_df, keep.extra.columns = T)
reduced.gr <- reduce(my_gr)

reduced.gr <- data.frame(reduced.gr)
myPeak <- paste("Peak", rownames(reduced.gr), sep = "_")
saf <- cbind(myPeak, reduced.gr[,1:3], reduced.gr[,5])
```

```
colnames(saf) <- c("peakID", "chr", "start", "end", "strand")

esa1saf <- saf

# Guardar archivo .saf con los picos fusionados

setwd("/home/biouser/Desktop/")
write.table(saf, file="Esa1_PicosFus_YMC.saf", quote = F, sep = '\t', row.names = F,
col.names = F)



## Gcn5

# Indicar carpeta con los narrowPeak files (resultado del peak calling)

setwd("/home/biouser/Desktop/Cluster/YMC/denovo/PeakCall/Gnc5/")

# Hacer data frame con los archivos de los pcios

peak.files <- list.files(pattern = "narrowPeak")
my_df <- data.frame()

for (file in peak.files){
  print(file)
  a <- read.table(file)
  colnames(a) <- c("chr", "start", "end", "peak", "score", "strand", "signalValue", "pValue",
"qValue")
  my_df <- rbind.data.frame(my_df, a)
}

# "Fusionar" regiones con picos en posiciones unicas

my_gr <- makeGRangesFromDataFrame(my_df, keep.extra.columns = T)
reduced.gr <- reduce(my_gr)

reduced.gr <- data.frame(reduced.gr)
myPeak <- paste("Peak", rownames(reduced.gr), sep = "_")
```

```
saf <- cbind(myPeak, reduced.gr[,1:3], reduced.gr[,5])
colnames(saf) <- c("peakID", "chr", "start", "end", "strand")

gcn5saf <- saf

# Guardar archivo .saf con los picos fusionados

setwd("/home/biouser/Desktop/")
write.table(saf, file="Gcn5_PicosFus_YMC.saf", quote = F, sep = '\t', row.names = F,
col.names = F)



## Set1

# Indicar carpeta con los narrowPeak files (resultado del peak calling)

setwd("/home/biouser/Desktop/Cluster/YMC/denovo/PeakCall/Set1/")

# Hacer data frame con los archivos de los pcios

peak.files <- list.files(pattern = "narrowPeak")
my_df <- data.frame()

for (file in peak.files){
  print(file)
  a <- read.table(file)
  colnames(a) <- c("chr", "start", "end", "peak", "score", "strand", "signalValue", "pValue",
"qValue")
  my_df <- rbind.data.frame(my_df, a)
}

# "Fusionar" regiones con picos en posiciones unicas

my_gr <- makeGRangesFromDataFrame(my_df, keep.extra.columns = T)
reduced.gr <- reduce(my_gr)

reduced.gr <- data.frame(reduced.gr)
```

```
myPeak <- paste("Peak", rownames(reduced.gr), sep = "_")
saf <- cbind(myPeak, reduced.gr[,1:3], reduced.gr[,5])
colnames(saf) <- c("peakID", "chr", "start", "end", "strand")

set1saf <- saf

# Guardar archivo .saf con los picos fusionados (SAF: Simplified Annotation Format)

setwd("/home/biouser/Desktop/")
write.table(saf, file="Set1_PicosFus_YMC.saf", quote = F, sep = '\t', row.names = F,
col.names = F)



saf.data <-list("esa1saf" = esa1saf,
                "gcn5saf" = gcn5saf,
                "set1saf" = set1saf)

save(saf.data, file = "/home/biouser/Desktop/MatrizConteo/saf.data.RData")



### Generar matriz de conteos



# Cargar paquete

library(Rsubread)

## Esa1

# Definir working directory donde esten los archivos

setwd("/home/biouser/Desktop/Cluster/YMC/denovo/Alineados/Esa1/")

# Crear variable con todos los nombres de archivos BAM o SAM

sam.files <- list.files(pattern = "sam")
```

```
# Usar la funcion featureCounts

Esa1counts <-featureCounts(files = sam.files, annot.ext =
"/home/biouser/Desktop/15.05/Esa1_PicosFus_YMC.saf")

# Guardar la matriz de conteos

save(Esa1counts, file = "/home/biouser/Desktop/Esa1counts.RData")



## Gcn5

# Definir working directory donde esten los archivos

setwd("/home/biouser/Desktop/CLUSTER/YMC/denovo/Alineados/Gnc5/")

# Crear variable con todos los nombres de archivos BAM o SAM

sam.files <- list.files(pattern = "sam")

# Usar la funcion featureCounts

Gcn5counts <- featureCounts(files = sam.files, annot.ext =
"/home/biouser/Desktop/15.05/Gcn5_PicosFus_YMC.saf")

# Guardar la matriz de conteos

save(Gcn5counts, file = "/home/biouser/Desktop/Gcn5counts.RData")

## Set1

# Definir working directory donde esten los archivos

setwd("/home/biouser/Desktop/CLUSTER/YMC/denovo/Alineados/Set1/")

# Crear variable con todos los nombres de archivos BAM o SAM
```

```
sam.files <- list.files(pattern = "sam")

# Usar la funcion featureCounts

Set1counts <- featureCounts(files = sam.files, annot.ext =
"/home/biouser/Desktop/15.05/Set1_PicosFus_YMC.saf")

# Guardar la matriz de conteos

save(Set1counts, file = "/home/biouser/Desktop/Set1counts.RData")



### Guardar matrices de conteos en formato matriz

# Esa1

Esa1CountMatrix <- as.matrix(Esa1counts[["counts"]])

colnam <- c(1:14)

colnames(Esa1CountMatrix) <- colnam



# Gcn5

Gcn5CountMatrix <- as.matrix(Gcn5counts[["counts"]])

colnam <- c(1:14)

colnames(Gcn5CountMatrix) <- colnam

# Set1

Set1CountMatrix <- as.matrix(Set1counts[["counts"]])

colnam <- c(1:14)
```

```
colnames(Set1CountMatrix) <- colnam


# Guardar

count.matrix.mods <- list("Esa1CountMatrix" = Esa1CountMatrix,
                          "Gcn5CountMatrix" = Gcn5CountMatrix,
                          "Set1CountMatrix" = Set1CountMatrix)

save(count.matrix.mods, file = "/home/biouser/Desktop/count.matrix.mods.RData")



##### Hacer PCA con los conteos y picos

# Usar NoiSeq (paquete de R, bioconductor)

library(NOISeq)

# Esa1

# Obtener los datos

my.data.1 <- count.matrix.mods[["Esa1CountMatrix"]]
my.data.1 <- rpkm(my.data.1)
my.factors.1 <- data.frame(TimePoint = c("1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11",
"12", "13", "14"),
                           Phase = c(rep("RC",2), rep("OX",3), rep("RB",3), rep("RC",6)))

my.extra.info.1 <- saf.data[["esa1saf"]]
my.extra.info.1 <- my.extra.info.1[, c(2,3,4)]
rownames(my.extra.info.1) <- saf.data[["esa1saf"]][, 1]

data.noiseq.esa1 <- readData(data = my.data.1, chromosome = my.extra.info.1, factors =
my.factors.1)
```

```
myPCA = dat(data.noiseq.esa1, type = "PCA", norm = T, logtransf = F)
explo.plot(myPCA, samples = c(1,2), plottype = "loadings", factor = "TimePoint")
explo.plot(myPCA, samples = c(1,3), plottype = "loadings", factor = "TimePoint")
# PC1 separa RC de OX, y PC3 separa OX y RB
explo.plot(myPCA, samples = c(1,2), plottype = "scores", factor = "Phase")
explo.plot(myPCA, samples = c(1,3), plottype = "scores", factor = "Phase")
explo.plot(myPCA, samples = c(1,2), plottype = "scores", factor = "TimePoint")
explo.plot(myPCA, samples = c(1,3), plottype = "scores", factor = "TimePoint")



# Gcn5


my.data.2 <- count.matrix.mods[["Gcn5CountMatrix"]]
my.data.2 <- rpkm(my.data.2)
my.factors.2 <- data.frame(TimePoint = c("1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11",
"12", "13", "14"),
                           Phase = c(rep("RC",2), rep("OX",3), rep("RB",3), rep("RC",6)))


my.extra.info.2 <- saf.data[["gcn5saf"]]
my.extra.info.2 <- my.extra.info.2[, c(2,3,4)]
rownames(my.extra.info.2) <- saf.data[["gcn5saf"]][, 1]


data.noiseq.gcn5 <- readData(data = my.data.2, chromosome = my.extra.info.2, factors =
my.factors.2)


myPCA = dat(data.noiseq.gcn5, type = "PCA", norm = T, logtransf = F)


explo.plot(myPCA, samples = c(1,2), plottype = "loadings", factor = "TimePoint")
explo.plot(myPCA, samples = c(1,5), plottype = "loadings", factor = "TimePoint")
explo.plot(myPCA, samples = c(2,5), plottype = "loadings", factor = "TimePoint")


explo.plot(myPCA, samples = c(1,2), plottype = "scores", factor = "Phase")
explo.plot(myPCA, samples = c(1,5), plottype = "scores", factor = "Phase")
explo.plot(myPCA, samples = c(2,5), plottype = "scores", factor = "Phase")
explo.plot(myPCA, samples = c(1,2), plottype = "scores", factor = "TimePoint")
explo.plot(myPCA, samples = c(1,5), plottype = "scores", factor = "TimePoint")
explo.plot(myPCA, samples = c(2,5), plottype = "scores", factor = "TimePoint")
```

```
# PC2 y PC5 separan OX y RB
# PC1 separa RC de OX y RB. PC2 separa OX y RB



# Set1

my.data.3 <- count.matrix.mods[["Set1CountMatrix"]]
my.data.3 <- rpkm(my.data.3)
my.factors.3 <- data.frame(TimePoint = c("1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11",
"12", "13", "14"),
                           Phase = c(rep("RC",2), rep("OX",3), rep("RB",3), rep("RC",6)))

my.extra.info.3 <- saf.data[["set1saf"]]
my.extra.info.3 <- my.extra.info.3[, c(2,3,4)]
rownames(my.extra.info.3) <- saf.data[["set1saf"]][, 1]

data.noiseq.set1 <- readData(data = my.data.3, chromosome = my.extra.info.3, factors =
my.factors.3)

myPCA = dat(data.noiseq.set1, type = "PCA", norm = T, logtransf = F)
# pca.set1.scores.pc1.pc2

explo.plot(myPCA, samples = c(1,2), plottype = "loadings", factor = "TimePoint")
explo.plot(myPCA, samples = c(1,5), plottype = "loadings", factor = "TimePoint")
explo.plot(myPCA, samples = c(2,5), plottype = "loadings", factor = "TimePoint")

explo.plot(myPCA, samples = c(1,2), plottype = "scores", factor = "Phase")
explo.plot(myPCA, samples = c(1,2), plottype = "scores", factor = "TimePoint")
# RC no se explica por ningÃºn PC
explo.plot(myPCA, samples = c(2,5), plottype = "scores", factor = "Phase")
explo.plot(myPCA, samples = c(2,5), plottype = "scores", factor = "TimePoint")
# PC5 separa OX (mezcla con RC) y RB. 2 separa RC-RB de OX
explo.plot(myPCA, samples = c(1,5), plottype = "scores", factor = "Phase")
explo.plot(myPCA, samples = c(1,5), plottype = "scores", factor = "TimePoint")

# Guardar datos noiseq
```

```
ymc.enzymes.noiseq.data <- list("data.noiseq.esa1" = data.noiseq.esa1,
                                "data.noiseq.gcn5" = data.noiseq.gcn5,
                                "data.noiseq.set1" = data.noiseq.set1)

save(ymc.enzymes.noiseq.data, file =
"/home/biouser/Desktop/MatrizConteo/ymc.enzymes.noiseq.data.RData")
```

Read count matrixes normalization

```
# Cargar paquete para normalizar

library(Rsubread)


# Cargar matriz de conteos

load("/home/biouser/Desktop/YMC-TFG/9.MatrizConteo/count.matrix.mods.RData")


# Obtener datos de conteos


conteosEsa = count.matrix.mods[["Esa1CountMatrix"]]

conteosGcn = count.matrix.mods[["Gcn5CountMatrix"]]

conteosSet = count.matrix.mods[["Set1CountMatrix"]]


# Normalizar datos

conteosEsa = rpkm(conteosEsa+1)

conteosGcn = rpkm(conteosGcn+1)

conteosSet = rpkm(conteosSet+1)
```

```
# Transformacion logaritmica

conteosEsa = log(conteosEsa)

conteosGcn = log(conteosGcn)

conteosSet = log(conteosSet)


# CENTRAR y NO escalar

conteosEsa = scale(conteosEsa, center = TRUE, scale = FALSE)

conteosGcn = scale(conteosGcn, center = TRUE, scale = FALSE)

conteosSet = scale(conteosSet, center = TRUE, scale = FALSE)



count.matrix.mods.cent<- list("Esa1CountMatrix" = conteosEsa,
                              "Gcn5CountMatrix" = conteosGcn,
                              "Set1CountMatrix" = conteosSet)


save(count.matrix.mods.cent, file = "/home/biouser/Desktop/MORE-
Analysis/count.matrix.mods.cent.RData")
```

MORE analysis of H3K9

```
########################
## MORE matrices
## pre-processing
########################

### Cargar datos

load("/home/biouser/Desktop/MORE-Analysis/count.matrix.mods.cent.RData") # datos de
conteos de enzimas, previamente normalizados, log transformados y centrados
```

```
load("/home/biouser/Desktop/MORE-Analysis/Matrixes_ForIntegration_POST_BatchCorrection_withK18ac.RData") # Datos de acetilaciones, con las mismas caracteristicas

# Quitra datos innecesarios

rm(H3k14ac, H3k14ac_TSSplus, H3k4me3, H3k4me3_TSSplus, H3k56ac, H3k56ac_TSSplus, H4k5ac,H4k5ac_TSSplus, H3k9ac2, H3k9ac2_TSSplus)


## Definir condiciones y opciones para MORE

options(stringsAsFactors = FALSE)

setwd("/home/biouser/Desktop/MORE-Analysis/")

# Cargar funciones necesarias para el MORE

source("auxFunctions.R")
source("ComputeGLM_function.R")
source("MORE_GLM.R")


### Trasponer las matrices de acetilaciones para que los puntos temporales sean las columnas

H3k18ac_TSSplus<-as.data.frame(t(H3k18ac_TSSplus))
genesH3k18ac_TSSplus <- unique(rownames(H3k18ac_TSSplus))


H3k9ac_TSSplus<-as.data.frame(t(H3k9ac_TSSplus))
genesH3k9ac_TSSplus <- unique(rownames(H3k9ac_TSSplus))


H3k18ac<-as.data.frame(t(H3k18ac))
genesH3k18ac <- unique(rownames(H3k18ac))
```

```
H3k9ac<-as.data.frame(t(H3k9ac))
genesH3k9ac <- unique(rownames(H3k9ac))



### Hacer media de los puntos 11 y 12 de los modificadores de histonas
## renombrar columnas



# Esa1

Esa1Counts <- count.matrix.mods.cent[["Esa1CountMatrix"]]

Esa1CountsA <- Esa1Counts[, c(11, 12)]
Esa1CountsA <- rowMeans(Esa1CountsA, na.rm = FALSE, dims = 1)

Esa1Counts <- cbind(Esa1Counts[, c(1:10)], Esa1CountsA, Esa1Counts[, c(13:14)])
colnames(Esa1Counts) <- c("t1", "t2", "t3", "t4", "t5", "t6", "t7", "t8", "t9", "t10", "t11", "t12",
"t13")



# Gcn5

Gcn5Counts <- count.matrix.mods.cent[["Gcn5CountMatrix"]]

Gcn5CountsA <- Gcn5Counts[, c(11, 12)]
Gcn5CountsA <- rowMeans(Gcn5CountsA, na.rm = FALSE, dims = 1)

Gcn5Counts <- cbind(Gcn5Counts[, c(1:10)], Gcn5CountsA, Gcn5Counts[, c(13:14)])
colnames(Gcn5Counts) <- c("t1", "t2", "t3", "t4", "t5", "t6", "t7", "t8", "t9", "t10", "t11",
"t12", "t13")



### Eliminar puntos 7 y 9 de acetilaciones y renombrar matrices
```

```
H3k9ac <- cbind(H3k9ac[, c(1:6)], H3k9ac[, 8], H3k9ac[, c(10:15)])
colnames(H3k9ac) <- c("t1", "t2", "t3", "t4", "t5", "t6", "t7", "t8", "t9", "t10", "t11", "t12", "t13")

H3k9ac_TSSplus <- cbind(H3k9ac_TSSplus[, c(1:6)], H3k9ac_TSSplus[, 8], H3k9ac_TSSplus[, c(10:15)])
colnames(H3k9ac_TSSplus) <- c("t1", "t2", "t3", "t4", "t5", "t6", "t7", "t8", "t9", "t10", "t11", "t12", "t13")

H3k18ac <- cbind(H3k18ac[, c(1:6)], H3k18ac[, 8], H3k18ac[, c(10:15)])
colnames(H3k18ac) <- c("t1", "t2", "t3", "t4", "t5", "t6", "t7", "t8", "t9", "t10", "t11", "t12", "t13")

H3k18ac_TSSplus <- cbind(H3k18ac_TSSplus[, c(1:6)], H3k18ac_TSSplus[, 8], H3k18ac_TSSplus[, c(10:15)])
colnames(H3k18ac_TSSplus) <- c("t1", "t2", "t3", "t4", "t5", "t6", "t7", "t8", "t9", "t10", "t11", "t12", "t13")



### Guardar los datos procesados que se van a usar para el MORE
# Se cogeran solo los conteos PLUS (tras el TSS), pues los datos del otro son muy similares

data.for.MORE.acetilation.enzymes <- list("Esa1Counts" = Esa1Counts,
                                          "Gcn5Counts" = Gcn5Counts,
                                          "H3k9ac_TSSplus" = H3k9ac_TSSplus,
                                          "H3k18ac_TSSplus" = H3k18ac_TSSplus)

save(data.for.MORE.acetilation.enzymes, file = "data.for.MORE.acetilation.enzymes.RData")



### A partir de datos de RGmatch, crear asociacion Gen - pico de enzima

library(rlist)

# Esa1

rgmatch.Esa1 = read.table("/home/biouser/Desktop/MORE-Analysis/RGMatch/Esa1SafGeneAso.txt", header=T, as.is=T)
```

```
Esa1genelist <- unique(rgmatch.Esa1$Gene)

listRGesa1 <- list()
gennamesesa1 <- c()

for (gene in Esa1genelist){
  aaa1 <- rgmatch.Esa1[rgmatch.Esa1$Gene%in%gene, "name"]
  listRGesa1 <- list.append(listRGesa1, aaa1)
  gennamesesa1 <- c(gennamesesa1, gene)
}

names(listRGesa1) <- gennamesesa1



# Gcn5

rgmatch.Gcn5 = read.table("/home/biouser/Desktop/MORE-
Analysis/RGMatch/Gcn5SafGeneAso.txt", header=T, as.is=T)

Gcn5genelist <- unique(rgmatch.Gcn5$Gene)

listRGgcn5 <- list()
gennamesgcn5 <- c()

for (gene in Gcn5genelist){
  aaa1 <- rgmatch.Gcn5[rgmatch.Gcn5$Gene%in%gene, "name"]
  listRGgcn5 <- list.append(listRGgcn5, aaa1)
  gennamesgcn5 <- c(gennamesgcn5, gene)
}

names(listRGgcn5) <- gennamesgcn5



### Crear lista de genes comun entre las acetilaciones, Gcn5 y Esa1

# Genes totales de Esa1 y Gcn5
```

```
#genesEsaGcn <- unique(c(gennamesgcn5, gennamesesa1))

# Genes comunes de enzimas con H3k9

genes.comunes.H3k9.Esa1 <- intersect(gennamesesa1, genesH3k9ac_TSSplus)

genes.comunes.H3k9.Gcn5 <- intersect(gennamesgcn5, genesH3k9ac_TSSplus)

genes.comunes.H3k9 <- unique(c(genes.comunes.H3k9.Esa1, genes.comunes.H3k9.Gcn5))


# Genes comunes de enzimas con H3k18

genes.comunes.H3k18.Esa1 <- intersect(gennamesesa1, genesH3k18ac_TSSplus)

genes.comunes.H3k18.Gcn5 <- intersect(gennamesgcn5, genesH3k18ac_TSSplus)

genes.comunes.H3k18 <- unique(c(genes.comunes.H3k18.Esa1,
genes.comunes.H3k18.Gcn5))


####### Preparar listas de asociaciones

###  Reducir la lista de conteo a solo los genes comunes

## Acetilaciones

H3k9filt <- H3k9ac_TSSplus[genes.comunes.H3k9,]

H3k18filt <- H3k18ac_TSSplus[genes.comunes.H3k18,]

# Poner etiqueta a cada fila de la matriz de acetilaciones

rownames(H3k18filt)<-paste0("H3k18ac_",rownames(H3k18filt))
rownames(H3k9filt)<-paste0("H3k9ac_",rownames(H3k9filt))
#rownames(H3k18ac)<-paste0("H3k18ac_","min300ToTSS_",rownames(H3k18ac))
#rownames(H3k9ac)<-paste0("H3k9ac_","min300ToTSS_",rownames(H3k9ac))
```

```
## Enzimas

# Gcn5

peak.int.gcn5.k9.n <- c()

for (gene in genes.comunes.H3k9.Gcn5){
  peak.int.gcn5.k9.n <- c(peak.int.gcn5.k9.n, listRGgcn5[[gene]])
}

peak.int.gcn5.k9 <- unique(peak.int.gcn5.k9.n)

Gcn5Countsfiltk9 <- Gcn5Counts[peak.int.gcn5.k9,]


listRGgcn5filtk9 <- list()
genenames2 <- c()

for (gene in genes.comunes.H3k9.Gcn5){
  listRGgcn5filtk9 <- list.append(listRGgcn5filtk9, listRGgcn5[[gene]])
  genenames2 <- c(genenames2, gene)
}

names(listRGgcn5filtk9) <- genenames2


#

peak.int.gcn5.k18.n <- c()

for (gene in genes.comunes.H3k18.Gcn5){
  peak.int.gcn5.k18.n <- c(peak.int.gcn5.k18.n, listRGgcn5[[gene]])
}

peak.int.gcn5.k18 <- unique(peak.int.gcn5.k18.n)
```

```
Gcn5Countsfiltk18 <- Gcn5Counts[peak.int.gcn5.k18,]


listRGgcn5filtk18 <- list()
genenames2 <- c()

for (gene in genes.comunes.H3k18.Gcn5){
  listRGgcn5filtk18 <- list.append(listRGgcn5filtk18, listRGgcn5[[gene]])
  genenames2 <- c(genenames2, gene)
}

names(listRGgcn5filtk18) <- genenames2

#

peak.int.esa1.k9.n <- c()

for (gene in genes.comunes.H3k9.Esa1){
  peak.int.esa1.k9.n <- c(peak.int.esa1.k9.n, listRGesa1[[gene]])
}

peak.int.esa1.k9 <- unique(peak.int.esa1.k9.n)

Esa1Countsfiltk9 <- Esa1Counts[peak.int.esa1.k9,]



listRGesa1filtk9 <- list()
genenames2 <- c()

for (gene in genes.comunes.H3k9.Esa1){
  listRGesa1filtk9 <- list.append(listRGesa1filtk9, listRGesa1[[gene]])
  genenames2 <- c(genenames2, gene)
}

names(listRGesa1filtk9) <- genenames2
```

```
#

peak.int.esa1.k18.n <- c()

for (gene in genes.comunes.H3k18.Esa1){
  peak.int.esa1.k18.n <- c(peak.int.esa1.k18.n, listRGesa1[[gene]])
}

peak.int.esa1.k18 <- unique(peak.int.esa1.k18.n)

Esa1Countsfiltk18 <- Esa1Counts[peak.int.esa1.k18,]



listRGesa1filtk18 <- list()
genenames2 <- c()

for (gene in genes.comunes.H3k18.Esa1){
  listRGesa1filtk18 <- list.append(listRGesa1filtk18, listRGesa1[[gene]])
  genenames2 <- c(genenames2, gene)
}

names(listRGesa1filtk18) <- genenames2


# Poner etiqueta a cada fila de la matriz de enzimas

rownames(Gcn5Countsfiltk18)<-paste0("Gcn5_",rownames(Gcn5Countsfiltk18))

rownames(Esa1Countsfiltk18)<-paste0("Esa1_",rownames(Esa1Countsfiltk18))

rownames(Gcn5Countsfiltk9)<-paste0("Gcn5_",rownames(Gcn5Countsfiltk9))

rownames(Esa1Countsfiltk9)<-paste0("Esa1_",rownames(Esa1Countsfiltk9))
```

```
###################
# MORE para H3k9
###################

### Crear tabla de asociacion Gen de acetilacion - Pico de enzima

PeaksNGenes <- list("Esa1" = listRGesa1filtk9,
                    "Gcn5" = listRGgcn5filtk9)

leng1 <- length(peak.int.esa1.k9.n)

leng2 <- length(peak.int.gcn5.k9.n)

leng3 <- leng1 + leng2

asociacion<-as.data.frame(matrix(data=NA, nrow = leng3, ncol = 2))
colnames(asociacion)<-c("identificador_gen","chip_regulator")
#names(ConsideredChipMatrixes)

contador <- 1
contadorlista <- 0

for (list in PeaksNGenes){
  contadorlista <- contadorlista + 1
  if (contadorlista == 1){
    contA <- 1
    for (gene in list){
      for (peak in gene){
        gene1 <- names(list[contA])
        chipname <- paste("Esa1_", peak, sep = "")
        asociacion[contador,"identificador_gen"] <- paste("H3k9ac_", gene1, sep = "")
        asociacion[contador,"chip_regulator"] <- chipname
        contador <- contador + 1
      }
      contA <- contA + 1
      }
```

```
  }
  else if (contadorlista == 2){
    contA <- 1
    for (gene in list){
      for (peak in gene){
        gene1 <- names(list[contA])
        chipname <- paste("Gcn5_", peak, sep = "")
        asociacion[contador,"identificador_gen"] <- paste("H3k9ac_", gene1, sep = "")
        asociacion[contador,"chip_regulator"] <- chipname
        contador <- contador + 1
      }
      contA <- contA + 1
      }
  }
}


#### Ordenar e identificar cada fila
asociacion<-asociacion[order(asociacion$identificador_gen),]
rownames(asociacion)<-paste0(asociacion$identificador_gen,"_",asociacion$chip_regulator)


## Comprobar que la cantidad total de asociaciones estÃ¡ bien
# 4787 associations in total, as leng3, OK

sum(asociacion$chip_regulator %in% rownames(Esa1Countsfiltk9)) # 2755
sum(asociacion$chip_regulator %in% rownames(Gcn5Countsfiltk9)) # 2032



# Lista de asociacion definitiva para el MORE

Asociaciones <- list("CHIP"=asociacion) #Chip is the omic we want to associate with the
RNA-Seq data
head(Asociaciones$CHIP)
```

```
### Aplicacion directa del MORE



# Guardar todas las matrices de enzimas juntas y hacer tabla conjunta

ConsideredChipMatrixesk9 <- list("Esa1Counts" = Esa1Countsfiltk9,
                                 "Gcn5Counts" = Gcn5Countsfiltk9)



CHIP <- do.call(rbind, ConsideredChipMatrixesk9)
NOMBRESFILAS<-unlist(lapply(ConsideredChipMatrixesk9,rownames),use.names =
FALSE)
rownames(CHIP)<-NOMBRESFILAS



data.omics<-list()
data.omics$"CHIP"<-CHIP
head(data.omics$CHIP)



# Computing GLMs ------------------------------------------------------------
####  Design=NULL
min.var = c(0) #one val fo each descriptive omic, minimum variance required, if less,
removed, we have already filtered per RNA-Seq


names(min.var)<-names("CHIP")


### MORE ###


library(MORE)


# Dado que hay genes conflictivos (falta de variabilidad), los quitamos de las matrices:
#  H3k9ac_YNL333W, H3k9ac_YHR053C, H3k9ac_YCR104W, H3k9ac_YOR388C,
H3k9ac_YOR390W, H3k9ac_YPL276W


H3k9filt <- H3k9filt[-which(rownames(H3k9filt)=="H3k9ac_YNL333W"),]
H3k9filt <- H3k9filt[-which(rownames(H3k9filt)=="H3k9ac_YHR053C"),]
H3k9filt <- H3k9filt[-which(rownames(H3k9filt)=="H3k9ac_YCR104W"),]
```

```
H3k9filt <- H3k9filt[-which(rownames(H3k9filt)=="H3k9ac_YOR388C"),]
H3k9filt <- H3k9filt[-which(rownames(H3k9filt)=="H3k9ac_YOR390W"),]
H3k9filt <- H3k9filt[-which(rownames(H3k9filt)=="H3k9ac_YPL276W"),]


GLMresults = GetGLM(GeneExpression = H3k9filt, #gene expression,our response variable
                    associations = Asociaciones,
                    data.omics = data.omics,
                    edesign=NULL, #the experimental covariates that we want to include
                    # degree = 1, no hacer caso, es de una previous versions
                    Res.df = 8, epsilon = 0.00001, # we have 15 time points, so 14 degrees of
freedom, if we leave 10  then just 4 chip variables can be significantly associated, by reducing it
to 8 (despite we lose statistical power) 2 variables more can enter to be putatively significant
associated
                    alfa=0.05, MT.adjust = "fdr",
                    family = gaussian(), #datos log transf...
                    elasticnet = 2, center = TRUE, scale = FALSE,
                    stepwise="two.ways.backward",
                    interactions.exp = TRUE, interactions.reg = 1,
                    min.variation=min.var,
                    correlation = 0.95, action = "mean", #to average the values of the higly correlated
variables
                    cont.var = NULL, min.obs = 10)


GLMresults.elasticnet2.H3k9.vs.Esa1.Gcn5 <- GLMresults


save(GLMresults.elasticnet2.H3k9.vs.Esa1.Gcn5, file =
"GLMresults.elasticnet2.H3k9.vs.Esa1.Gcn5.RData")



## Mirar los resultados



head(GLMresults$GlobalSummary$GoodnessOfFit)
head(GLMresults$GlobalSummary$ReguPerGene, 20)


GLMresults$ResultsPerGene$H3k9ac_YIL165C$significantRegulators
GLMresults$ResultsPerGene$H3k9ac_YIL165C$allRegulators
```

```
GLMresults$ResultsPerGene$H3k9ac_YIL165C$coefficients

GLMresults$ResultsPerGene[[3]]$coefficients

# Plotting significant regulations ---------------------------------------

mygene = "H3k9ac_YDR382W"
plotGLM(GLMresults, gene = mygene, regulator = NULL,
        xlab = "Time points", reguValues = NULL, plotPerOmic = FALSE,
        gene.col = 1, regu.col = "green4", replicates = FALSE,
        cont.var = c(1:13), cond2plot = NULL)

plotGLM(GLMresults, gene = mygene, regulator = "Gcn5_Peak_1457",
        xlab = "Time points", reguValues = NULL, plotPerOmic = FALSE,
        gene.col = 1, regu.col = "green4", replicates = TRUE,
        cont.var = c(1:13), cond2plot = NULL)



plot(x=1:13,y = H3k9filt["H3k9ac_YKR067W",],type = "l",col="blue")
lines(x=1:13,y = CHIP["Gcn5_Peak_1885",],col="red")



#### Obtener todos los reguladores significativos

library(rlist)

H3k9ResultsPerGene <- GLMresults.elasticnet2.H3k9.vs.Esa1.Gcn5[["ResultsPerGene"]]

H3k9SignRegs <- list()
genes.regs.nam <- c()
abbb <- 0

for (gene in H3k9ResultsPerGene){
  abbb <- abbb +1
  abbb2 <- names(H3k9ResultsPerGene[abbb])
  if (length(gene) == 5){
    if (length(gene[[5]])>0){
```

```
    abbb1 <- gene[["significantRegulators"]]
    H3k9SignRegs <- list.append(H3k9SignRegs, abbb1)
    genes.regs.nam <- c(genes.regs.nam, abbb2)
   }
  }
}

names(H3k9SignRegs) <- genes.regs.nam



# Obtener todos los genes regulados por enzima

H3k9Esa1Reg <- c()
H3k9Gcn5Reg <- c()

contD <- 0

for (gene in H3k9SignRegs){
  grepgcn <- grep("Gcn5_", gene)
  grepesa <- grep("Esa1_", gene)
  contD <- contD + 1
  if (isEmpty(grepgcn) == FALSE){
    H3k9Gcn5Reg <- c(H3k9Gcn5Reg, names(H3k9SignRegs[contD]))
  }
  if (isEmpty(grepesa) == FALSE){
    H3k9Esa1Reg <- c(H3k9Esa1Reg, names(H3k9SignRegs[contD]))
  }
}

H3k9Esa1Reg <- unique(H3k9Esa1Reg)
H3k9Gcn5Reg <- unique(H3k9Gcn5Reg)

#####
## Venn diagrams
#####
```

```
# Hacer Venn Diagram de genes regulados

library(VennDiagram)

diag.H3k9.regs <- draw.pairwise.venn(area1 = length(H3k9Esa1Reg),
                                     area2 = length(H3k9Gcn5Reg),
                                     cross.area = length(intersect(H3k9Esa1Reg, H3k9Gcn5Reg)),
                                     category = c("Esa1", "Gcn5"),
                                     fill = c("blue", "green"),
                                     scaled = TRUE)


grid.newpage()

### Sacar nombres de solo genes


# Comunes

intersect.H3k9.Esa1.Gcn5 <- intersect(H3k9Esa1Reg, H3k9Gcn5Reg)

Genes.H3k9.Esa1.Gcn5 <- c()

for (element in intersect.H3k9.Esa1.Gcn5){
  char <- as.character(element)
  Genes.H3k9.Esa1.Gcn5 <- c(Genes.H3k9.Esa1.Gcn5, substr(char, 8, nchar(char)))
}

# Gcn5-H3k9

Genes.H3k9.Gcn5 <- c()

for (element in H3k9Gcn5Reg){
  char <- as.character(element)
  Genes.H3k9.Gcn5 <- c(Genes.H3k9.Gcn5, substr(char, 8, nchar(char)))
}
```

```
# Esa1-H3k9

Genes.H3k9.Esa1 <- c()

for (element in H3k9Esa1Reg){
  char <- as.character(element)
  Genes.H3k9.Esa1 <- c(Genes.H3k9.Esa1, substr(char, 8, nchar(char)))
}



### Ver a quÃ© fase del cluster de DEGs del YMC pertenecen los genes

load("/home/biouser/Desktop/YMC-TFG/6.YMC-R-Objects/GenesYMCCluster.RData")



### Los comunes
# Genes.H3k9.Esa1.Gcn5.vs.OX

grid.newpage()
draw.pairwise.venn(area1 = length(Genes.H3k9.Esa1.Gcn5),
                   area2 = length(genes.YMC.Cluster[["Fase OX"]]),
                   cross.area = length(intersect(Genes.H3k9.Esa1.Gcn5, genes.YMC.Cluster[["Fase
OX"]])),
                   category = c("H3k9.Esa1.Gcn5", "Fase OX"),
                   fill = c("yellow", "red"))

grid.newpage()

draw.pairwise.venn(area1 = length(Genes.H3k9.Esa1.Gcn5),
                   area2 = length(genes.YMC.Cluster[["Fase RB"]]),
                   cross.area = length(intersect(Genes.H3k9.Esa1.Gcn5, genes.YMC.Cluster[["Fase
RB"]])),
                   category = c("H3k9.Esa1.Gcn5", "Fase RB"),
                   fill = c("yellow", "green"))
grid.newpage()
```

```
  draw.pairwise.venn(area1 = length(Genes.H3k9.Esa1.Gcn5),
                     area2 = length(genes.YMC.Cluster[["Fase RC"]]),
                     cross.area = length(intersect(Genes.H3k9.Esa1.Gcn5, genes.YMC.Cluster[["Fase
RC"]])),
                     category = c("H3k9.Esa1.Gcn5", "Fase RC"),
                     fill = c("yellow", "blue"))


  ### Los Gcn5-H3k9
  # Genes.H3k9.Gcn5.vs.OX


  grid.newpage()
  draw.pairwise.venn(area1 = length(Genes.H3k9.Gcn5),
                     area2 = length(genes.YMC.Cluster[["Fase OX"]]),
                     cross.area = length(intersect(Genes.H3k9.Gcn5, genes.YMC.Cluster[["Fase
OX"]])),
                     category = c("H3k9.Gcn5", "Fase OX"),
                     fill = c("yellow", "red"))

  grid.newpage()
  draw.pairwise.venn(area1 = length(Genes.H3k9.Gcn5),
                     area2 = length(genes.YMC.Cluster[["Fase RB"]]),
                     cross.area = length(intersect(Genes.H3k9.Gcn5, genes.YMC.Cluster[["Fase
RB"]])),
                     category = c("H3k9.Gcn5", "Fase RB"),
                     fill = c("yellow", "green"))

  grid.newpage()
  draw.pairwise.venn(area1 = length(Genes.H3k9.Gcn5),
                     area2 = length(genes.YMC.Cluster[["Fase RC"]]),
                     cross.area = length(intersect(Genes.H3k9.Gcn5, genes.YMC.Cluster[["Fase
RC"]])),
                     category = c("H3k9.Gcn5", "Fase RC"),
                     fill = c("yellow", "blue"))
```

```
### Los Esa1-H3k9
# Genes.H3k9.Esa1.vs.OX

grid.newpage()
draw.pairwise.venn(area1 = length(Genes.H3k9.Esa1),
                   area2 = length(genes.YMC.Cluster[["Fase OX"]]),
                   cross.area = length(intersect(Genes.H3k9.Esa1, genes.YMC.Cluster[["Fase
OX"]])),
                   category = c("H3k9.Esa1", "Fase OX"),
                   fill = c("yellow", "red"))

grid.newpage()
draw.pairwise.venn(area1 = length(Genes.H3k9.Esa1),
                   area2 = length(genes.YMC.Cluster[["Fase RB"]]),
                   cross.area = length(intersect(Genes.H3k9.Esa1, genes.YMC.Cluster[["Fase
RB"]])),
                   category = c("H3k9.Esa1", "Fase RB"),
                   fill = c("yellow", "green"))

grid.newpage()
draw.pairwise.venn(area1 = length(Genes.H3k9.Esa1),
                   area2 = length(genes.YMC.Cluster[["Fase RC"]]),
                   cross.area = length(intersect(Genes.H3k9.Esa1, genes.YMC.Cluster[["Fase
RC"]])),
                   category = c("H3k9.Esa1", "Fase RC"),
                   fill = c("yellow", "blue"))




########################################
### Hacer enriquecimiento de los genes
########################################



### Funciones de enriquecimiento

EnrichALLterms = function (test, notTest, annotation,
```

```
                    p.adjust.method = "fdr") {

  annot2test = unique(annotation[,2])

  resultat = t(sapply(annot2test, Enrich1term, test = test, notTest = notTest, annotation =
annotation))

  return (data.frame(resultat[,-6],
                     "pval" = as.numeric(resultat[,"pval"]),
                     "adjPval" = p.adjust(as.numeric(resultat[,"pval"]), method = p.adjust.method),
                     stringsAsFactors = F))

}

Enrich1term = function (term, test, notTest, annotation) {

  annotTest = length(intersect(test, annotation[annotation[,2] == term,1]))

  if ((annotTest) > 0) {
    annotNOTtest = length(intersect(notTest, annotation[annotation[,2] == term,1]))
    mytest = matrix(c(annotTest, length(test)-annotTest, annotNOTtest, length(notTest)-
annotNOTtest), ncol = 2)
    resultat = c(term, annotTest, length(test), annotNOTtest, length(notTest),
                 #TEST DE FISHER
                 fisher.test(mytest, alternative = "greater")$p.value)
    names(resultat) = c("term", "annotTest", "test", "annotNotTest", "notTest", "pval")
  } else {
    resultat = c(term, 0, 0, 0, 0, 100)
    names(resultat) = c("term", "annotTest", "test", "annotNotTest", "notTest", "pval")
  }

  return(resultat)

}



### Crear anotacion de los genes que hay en el organismo (gen asociado y descripciÃ³n del
GO term (name_1006) asociado)
```

```
library(biomaRt)

biomartYeast = useMart(biomart = "ensembl", dataset="scerevisiae_gene_ensembl")

annotation = getBM(attributes = c("ensembl_gene_id","name_1006"),
                   mart=biomartYeast)


######## Enriquecimiento contra organismo de todos los genes


#Obtener nombres de todos los genes de la levadura

notTest0 <-  getBM(attributes = c("ensembl_gene_id"),
                   mart=biomartYeast)
notTest0 <- as.vector(t(notTest0))


### Enriquecimiento en genes de Gcn5 y Esa1 comunes

#Obtener nombres de los genes asociados en la muestra

test1 <- intersect(H3k9Esa1Reg, H3k9Gcn5Reg)

test <- c()

for (element in test1){
  char <- as.character(element)
  test <- c(test, substr(char, 8, nchar(char)))
}


#Resta entre los genes totales en la levadura y los genes de la muestra

notTest <- setdiff(notTest0, test)
```

```
#Hacer analisis de enriquecimiento

GOEnrichMORE.H3k9.Esa1.Gcn5 = EnrichALLterms(test = test,
                        notTest = notTest,
                        # De la anotaciÃ³n de genes de la muestra, coger solo la columna 1
(nombre del gen)
                        # y 3 (nombre del termino GO)
                        annotation = annotation,
                        p.adjust.method = "fdr")

library("xlsx")

write.xlsx(GOEnrichMORE.H3k9.Esa1.Gcn5, file =
"GOEnrichMORE.H3k9.Esa1.Gcn5.xlsx", sheetName = "GOEnrichMORE.H3k9.Esa1.Gcn5",
          col.names = TRUE, row.names = TRUE, append = FALSE)



### Enriquecimiento en genes de Gcn5 y H3k9, no compartidos con Esa1

#Obtener nombres de los genes asociados en la muestra

test1 <- setdiff(H3k9Gcn5Reg, intersect(H3k9Esa1Reg, H3k9Gcn5Reg))

test <- c()

for (element in test1){
  char <- as.character(element)
  test <- c(test, substr(char, 8, nchar(char)))
}



#Resta entre los genes totales en la levadura y los genes de la muestra

notTest <- setdiff(notTest0, test)

#Hacer analisis de enriquecimiento
```

```
GOEnrichMORE.H3k9.Gcn5 = EnrichALLterms(test = test,
                                notTest = notTest,
                                # De la anotaciÃ³n de genes de la muestra, coger solo la columna
1 (nombre del gen)
                                # y 3 (nombre del termino GO)
                                annotation = annotation,
                                p.adjust.method = "fdr")


library("xlsx")

write.xlsx(GOEnrichMORE.H3k9.Gcn5, file = "GOEnrichMORE.H3k9.Gcn5.xlsx",
sheetName = "GOEnrichMORE.H3k9.Gcn5",
          col.names = TRUE, row.names = TRUE, append = FALSE)



### Enriquecimiento en genes de Esa1 y H3k9, no compartidos con Gcn5

#Obtener nombres de los genes asociados en la muestra

test1 <- setdiff(H3k9Esa1Reg, intersect(H3k9Esa1Reg, H3k9Gcn5Reg))

test <- c()

for (element in test1){
  char <- as.character(element)
  test <- c(test, substr(char, 8, nchar(char)))
}



#Resta entre los genes totales en la levadura y los genes de la muestra

notTest <- setdiff(notTest0, test)

#Hacer analisis de enriquecimiento

GOEnrichMORE.H3k9.Esa1 = EnrichALLterms(test = test,
                                notTest = notTest,
```

```
                        # De la anotaciÃ³n de genes de la muestra, coger solo la columna 1
(nombre del gen)
                        # y 3 (nombre del termino GO)
                        annotation = annotation,
                        p.adjust.method = "fdr")

  library("xlsx")

  write.xlsx(GOEnrichMORE.H3k9.Esa1, file = "GOEnrichMORE.H3k9.Esa1.xlsx",
sheetName = "GOEnrichMORE.H3k9.Esa1",
          col.names = TRUE, row.names = TRUE, append = FALSE)



  ######## Enriquecimiento contra fase del cluster de genes de cada enzima comunes al cluster


  ### Enriquecimiento en genes de Gcn5

  ## OX

  #Obtener nombres de todos los genes de la fase del cluster

  notTest0 <-  genes.YMC.Cluster[["Fase OX"]]

  #Obtener nombres de los genes asociados en la muestra

  test <- intersect(Genes.H3k9.Gcn5, genes.YMC.Cluster[["Fase OX"]])


  #Resta entre los genes totales en la levadura y los genes de la muestra

  notTest <- setdiff(notTest0, test)

  #Hacer analisis de enriquecimiento

  GOEnrichMORE.H3k9.Gcn5.OX = EnrichALLterms(test = test,
```

```
                                notTest = notTest,
                                # De la anotaciÃ³n de genes de la muestra, coger solo la columna
1 (nombre del gen)
                                # y 3 (nombre del termino GO)
                                annotation = annotation,
                                p.adjust.method = "fdr")

  library("xlsx")

  write.xlsx(GOEnrichMORE.H3k9.Gcn5.OX, file = "GOEnrichMORE.H3k9.Gcn5.OX.xlsx",
sheetName = "GOEnrichMORE.H3k9.Gcn5.OX",
          col.names = TRUE, row.names = TRUE, append = FALSE)



  ## RB

  #Obtener nombres de todos los genes de la fase del cluster

  notTest0 <-  genes.YMC.Cluster[["Fase RB"]]

  #Obtener nombres de los genes asociados en la muestra

  test <- intersect(Genes.H3k9.Gcn5, genes.YMC.Cluster[["Fase RB"]])



  #Resta entre los genes totales en la levadura y los genes de la muestra

  notTest <- setdiff(notTest0, test)

  #Hacer analisis de enriquecimiento

  GOEnrichMORE.H3k9.Gcn5.RB = EnrichALLterms(test = test,
                                notTest = notTest,
                                # De la anotaciÃ³n de genes de la muestra, coger solo la columna 1
(nombre del gen)
                                # y 3 (nombre del termino GO)
                                annotation = annotation,
                                p.adjust.method = "fdr")
```

```
library("xlsx")

write.xlsx(GOEnrichMORE.H3k9.Gcn5.RB, file = "GOEnrichMORE.H3k9.Gcn5.RB.xlsx",
sheetName = "GOEnrichMORE.H3k9.Gcn5.RB",
        col.names = TRUE, row.names = TRUE, append = FALSE)

## RC

#Obtener nombres de todos los genes de la fase del cluster

notTest0 <-  genes.YMC.Cluster[["Fase RC"]]

#Obtener nombres de los genes asociados en la muestra

test <- intersect(Genes.H3k9.Gcn5, genes.YMC.Cluster[["Fase RC"]])


#Resta entre los genes totales en la levadura y los genes de la muestra

notTest <- setdiff(notTest0, test)

#Hacer analisis de enriquecimiento

GOEnrichMORE.H3k9.Gcn5.RC = EnrichALLterms(test = test,
                                notTest = notTest,
                                # De la anotaciÃ³n de genes de la muestra, coger solo la columna 1
(nombre del gen)
                                # y 3 (nombre del termino GO)
                                annotation = annotation,
                                p.adjust.method = "fdr")

library("xlsx")

write.xlsx(GOEnrichMORE.H3k9.Gcn5.RC, file = "GOEnrichMORE.H3k9.Gcn5.RC.xlsx",
sheetName = "GOEnrichMORE.H3k9.Gcn5.RC",
        col.names = TRUE, row.names = TRUE, append = FALSE)
```

```
### Enriquecimiento en genes de Esa1

## OX

#Obtener nombres de todos los genes de la fase del cluster

notTest0 <-  genes.YMC.Cluster[["Fase OX"]]

#Obtener nombres de los genes asociados en la muestra

test <- intersect(Genes.H3k9.Esa1, genes.YMC.Cluster[["Fase OX"]])


#Resta entre los genes totales en la levadura y los genes de la muestra

notTest <- setdiff(notTest0, test)

#Hacer analisis de enriquecimiento

GOEnrichMORE.H3k9.Esa1.OX = EnrichALLterms(test = test,
                                notTest = notTest,
                                # De la anotaciÃ³n de genes de la muestra, coger solo la columna 1
(nombre del gen)
                                # y 3 (nombre del termino GO)
                                annotation = annotation,
                                p.adjust.method = "fdr")

library("xlsx")

write.xlsx(GOEnrichMORE.H3k9.Esa1.OX, file = "GOEnrichMORE.H3k9.Esa1.OX.xlsx",
sheetName = "GOEnrichMORE.H3k9.Esa1.OX",
          col.names = TRUE, row.names = TRUE, append = FALSE)


## RB
```

```
#Obtener nombres de todos los genes de la fase del cluster

notTest0 <-  genes.YMC.Cluster[["Fase RB"]]

#Obtener nombres de los genes asociados en la muestra

test <- intersect(Genes.H3k9.Esa1, genes.YMC.Cluster[["Fase RB"]])


#Resta entre los genes totales en la levadura y los genes de la muestra

notTest <- setdiff(notTest0, test)

#Hacer analisis de enriquecimiento

GOEnrichMORE.H3k9.Esa1.RB = EnrichALLterms(test = test,
                                notTest = notTest,
                                # De la anotaciÃ³n de genes de la muestra, coger solo la columna 1
(nombre del gen)
                                # y 3 (nombre del termino GO)
                                annotation = annotation,
                                p.adjust.method = "fdr")

library("xlsx")

write.xlsx(GOEnrichMORE.H3k9.Esa1.RB, file = "GOEnrichMORE.H3k9.Esa1.RB.xlsx",
sheetName = "GOEnrichMORE.H3k9.Esa1.RB",
        col.names = TRUE, row.names = TRUE, append = FALSE)

## RC

#Obtener nombres de todos los genes de la fase del cluster

notTest0 <-  genes.YMC.Cluster[["Fase RC"]]

#Obtener nombres de los genes asociados en la muestra
```

```
test <- intersect(Genes.H3k9.Esa1, genes.YMC.Cluster[["Fase RC"]])


#Resta entre los genes totales en la levadura y los genes de la muestra

notTest <- setdiff(notTest0, test)

#Hacer analisis de enriquecimiento

GOEnrichMORE.H3k9.Esa1.RC = EnrichALLterms(test = test,
                                notTest = notTest,
                                # De la anotaciÃ³n de genes de la muestra, coger solo la columna 1
(nombre del gen)
                                # y 3 (nombre del termino GO)
                                annotation = annotation,
                                p.adjust.method = "fdr")

library("xlsx")

write.xlsx(GOEnrichMORE.H3k9.Esa1.RC, file = "GOEnrichMORE.H3k9.Esa1.RC.xlsx",
sheetName = "GOEnrichMORE.H3k9.Esa1.RC",
        col.names = TRUE, row.names = TRUE, append = FALSE)



############################
## Dividir los genes en correlacion positiva y negativa

Genes.H3k9.CorrPos <- c()
Genes.H3k9.CorrPos.Esa1 <- c()
Genes.H3k9.CorrPos.Gcn5 <- c()
Genes.H3k9.CorrNeg <- c()
Genes.H3k9.CorrNeg.Esa1 <- c()
Genes.H3k9.CorrNeg.Gcn5 <- c()

cont2 <- 0
```

```
  for (gene in GLMresults[["ResultsPerGene"]]){
    cont2 <- cont2 + 1

    if (length(gene) == 5){

      if (length(gene[["significantRegulators"]])>0){

        for (signifreg in gene[["significantRegulators"]]){

          if (GLMresults[["ResultsPerGene"]][[cont2]][["allRegulators"]][signifreg, 5] ==
"Model"){
            corr <- GLMresults[["ResultsPerGene"]][[cont2]][["coefficients"]][signifreg,1]
          }
          else{
            chi.nam <- GLMresults[["ResultsPerGene"]][[cont2]][["allRegulators"]][signifreg, 5]
            corr <- GLMresults[["ResultsPerGene"]][[cont2]][["coefficients"]][chi.nam,1]
          }
          nombre.gen <- names(GLMresults[["ResultsPerGene"]][cont2])
          if (corr < 0){
            Genes.H3k9.CorrNeg <- c(Genes.H3k9.CorrNeg, nombre.gen)
            modif <- substr(signifreg, 1, 4)
            if (modif == "Esa1"){
              Genes.H3k9.CorrNeg.Esa1 <- c(Genes.H3k9.CorrNeg.Esa1, nombre.gen)
            }else if (modif == "Gcn5"){
              Genes.H3k9.CorrNeg.Gcn5 <- c(Genes.H3k9.CorrNeg.Gcn5, nombre.gen)}
          }else if (corr > 0){
            Genes.H3k9.CorrPos <- c(Genes.H3k9.CorrPos, nombre.gen)
            modif <- substr(signifreg, 1, 4)
            if (modif == "Esa1"){
              Genes.H3k9.CorrPos.Esa1 <- c(Genes.H3k9.CorrPos.Esa1, nombre.gen)
            }else if (modif == "Gcn5"){
              Genes.H3k9.CorrPos.Gcn5 <- c(Genes.H3k9.CorrPos.Gcn5, nombre.gen)}
          }
        }
      }
    }
  }
```

```
Genes.H3k9.CorrPos <- unique(Genes.H3k9.CorrPos)
Genes.H3k9.CorrPos.Esa1 <- unique(Genes.H3k9.CorrPos.Esa1)
Genes.H3k9.CorrPos.Gcn5 <- unique(Genes.H3k9.CorrPos.Gcn5)
Genes.H3k9.CorrNeg <- unique(Genes.H3k9.CorrNeg)
Genes.H3k9.CorrNeg.Esa1 <- unique(Genes.H3k9.CorrNeg.Esa1)
Genes.H3k9.CorrNeg.Gcn5 <- unique(Genes.H3k9.CorrNeg.Gcn5)


length(intersect(Genes.H3k9.CorrPos, Genes.H3k9.CorrNeg)) # 44

length(intersect(Genes.H3k9.CorrPos.Esa1, Genes.H3k9.CorrPos.Gcn5)) # 26
length(intersect(Genes.H3k9.CorrNeg.Esa1, Genes.H3k9.CorrNeg.Gcn5)) # 4

length(intersect(Genes.H3k9.CorrPos.Esa1, Genes.H3k9.CorrNeg.Esa1)) # 12
length(intersect(Genes.H3k9.CorrPos.Gcn5, Genes.H3k9.CorrNeg.Gcn5)) # 8



### Enriquecer genes

## Correlacion negativa

#Obtener nombres de todos los genes de la fase del cluster

notTest0 <-  getBM(attributes = c("ensembl_gene_id"),
                   mart=biomartYeast)
notTest0 <- as.vector(t(notTest0))

#Obtener nombres de los genes asociados en la muestra

test1 <- Genes.H3k9.CorrNeg

test <- c()

for (element in test1){
```

```
  char <- as.character(element)
  test <- c(test, substr(char, 8, nchar(char)))
}



#Resta entre los genes totales en la levadura y los genes de la muestra

notTest <- setdiff(notTest0, test)

#Hacer analisis de enriquecimiento

GOEnrichMORE.H3k9.CorrNeg = EnrichALLterms(test = test,
                                           notTest = notTest,
                                           # De la anotaciÃ³n de genes de la muestra, coger solo la columna 1
(nombre del gen)
                                           # y 3 (nombre del termino GO)
                                           annotation = annotation,
                                           p.adjust.method = "fdr")

library("xlsx")

write.xlsx(GOEnrichMORE.H3k9.CorrNeg, file = "GOEnrichMORE.H3k9.CorrNeg.xlsx",
sheetName = "GOEnrichMORE.H3k9.CorrNeg",
           col.names = TRUE, row.names = TRUE, append = FALSE)



## Correlacion positiva

#Obtener nombres de los genes asociados en la muestra

test1 <- Genes.H3k9.CorrPos

test <- c()

for (element in test1){
```

```
  char <- as.character(element)
  test <- c(test, substr(char, 8, nchar(char)))
 }



 #Resta entre los genes totales en la levadura y los genes de la muestra

 notTest <- setdiff(notTest0, test)

 #Hacer analisis de enriquecimiento

 GOEnrichMORE.H3k9.CorrPos = EnrichALLterms(test = test,
                                notTest = notTest,
                                # De la anotaciÃ³n de genes de la muestra, coger solo la columna 1
(nombre del gen)
                                # y 3 (nombre del termino GO)
                                annotation = annotation,
                                p.adjust.method = "fdr")

 library("xlsx")

 write.xlsx(GOEnrichMORE.H3k9.CorrPos, file = "GOEnrichMORE.H3k9.CorrPos.xlsx",
sheetName = "GOEnrichMORE.H3k9.CorrPos",
        col.names = TRUE, row.names = TRUE, append = FALSE)



 ###############
 ## Hacer Venn Diagrams segÃºn la correlacion
 ## contra las fases del cluster
 ##################
 ## Preparar nombres de genes
 # Positivo Esa1
 GenNames.H3k9.CorrPos.Esa1 <- c()

 for (element in Genes.H3k9.CorrPos.Esa1){
```

```
  char <- as.character(element)
  GenNames.H3k9.CorrPos.Esa1 <- c(GenNames.H3k9.CorrPos.Esa1, substr(char, 8,
nchar(char)))
 }

 # Negativo Esa1
 GenNames.H3k9.CorrNeg.Esa1 <- c()

 for (element in Genes.H3k9.CorrNeg.Esa1){
  char <- as.character(element)
  GenNames.H3k9.CorrNeg.Esa1 <- c(GenNames.H3k9.CorrNeg.Esa1, substr(char, 8,
nchar(char)))
 }

 # Positivo Gcn5
 GenNames.H3k9.CorrPos.Gcn5 <- c()

 for (element in Genes.H3k9.CorrPos.Gcn5){
  char <- as.character(element)
  GenNames.H3k9.CorrPos.Gcn5 <- c(GenNames.H3k9.CorrPos.Gcn5, substr(char, 8,
nchar(char)))
 }
 # Negativo Gcn5
 GenNames.H3k9.CorrNeg.Gcn5 <- c()

 for (element in Genes.H3k9.CorrNeg.Gcn5){
  char <- as.character(element)
  GenNames.H3k9.CorrNeg.Gcn5 <- c(GenNames.H3k9.CorrNeg.Gcn5, substr(char, 8,
nchar(char)))
 }
```