Faculty of Natural and Mathematical Sciences
Department of Informatics

King's College London
Strand Campus, London,
United Kingdom

7CCSMPRJ

Individual Project Submission 2023/24

Name:	Henry Cao
Student Number:	K23025410
Degree Programme:	MSc Data Science
Project Title:	Predicting Age-associated Changes Perturbing Muscle Stem Cell Function
Supervisor:	David Watson
Word Count:	14729

RELEASE OF PROJECT

Following the submission of your project, the Department would like to make it publicly available via the library electronic resources. You will retain copyright of the project.

☑ I **agree** to the release of my project

☐ I **do not** agree to the release of my project

Signature: Henry Cao **Date:** August 4, 2024

Department of Informatics
King's College London
United Kingdom

7CCSMPRJ Individual Project

Predicting Age-associated Changes Perturbing Muscle Stem Cell Function

---

Name: **Henry Cao**
Student Number: K23025410
Course: MSc Data Science

Supervisor: David Watson

This dissertation is submitted for the degree of MSc in MSc Data Science.

Acknowledgements

A large thanks to Dr David Watson and Dr Robert Knight for supervising my dissertation at King's College London. I would also like to thank the Buck Institute for Research on Aging for providing the CCR2KO dataset online for public use, and the University of British Columbia for putting the Ageing Muscle dataset online for public use.

Abstract

This research analyses age-associated changes that affect muscle stem cell function using two RNA-Seq datasets: a CCR2 knockout and an ageing muscle dataset. Three statistical methods are used to find differentially-expressed genes: DESeq2, Limma-Voom, and Bayesian generalised linear models. Gene Ontology analysis is also used to identify the biological functions of significant genes. Key findings include isolating a set of significant genes related to muscular regeneration and identifying the best methods for differential expression gene analysis involving small sample sizes. The research demonstrates that using multiple statistical and biological methods can provide a more comprehensive understanding of the genetic mechanisms underlying ageing muscle, and yield potential candidate genes for further research.

Nomenclature

GRN	Gene Regulatory Network
TF	Transcription Factor
EC	Endothelial Cell
FAP	Fibroadipogenic Progenitor
MP	Myogenic Progenitor
IC	Inflammatory Cells
PER	Pericytes
$M1$	Pro-inlammatory Macrophages
$M2$	Anti-inflammatory Macrophages
WT	Wild Type
KO	Knock Out
WT	Wild Type
EDA	Exploratory Data Analysis
$CCR2KO$	CCR2 Knockout
AM	Ageing Muscle

Contents

List of Figures

List of Tables

1 Introduction

Back in 1958, Francis Crick proposed the central dogma of molecular biology, which states that DNA encodes RNA, which encodes proteins [6]. Also known as the golden rule of molecular biology, Crick's statement articulates that the functions and mechanisms of cells are governed by information stored in RNA. RNA is a molecule that carries instruction for DNA and is encoded with four bases: adenine, guanin, cytosine, and uracil [7]. However, RNA and DNA are not the sole determinant of how cells function, as external factors, such as the environment and ageing, also affect how cells operate. Certain mechanisms, such as proteins, also function differently depending on the cell type [3].

The gene regulatory network (GRN) represents the practical implications of the golden rule of microbiology. It encompasses a broad range of mechanisms that control gene expression. Essentially, the GRN helps decide which genes should be switched on and off. This is mainly done through transcription factors (TF's), which are proteins that signal to the cell to begin or end the processing of transcribing a gene [8]. In the context of muscular regeneration, the GRN plays a crucial role in the inflammation and healing of damaged muscle tissue.

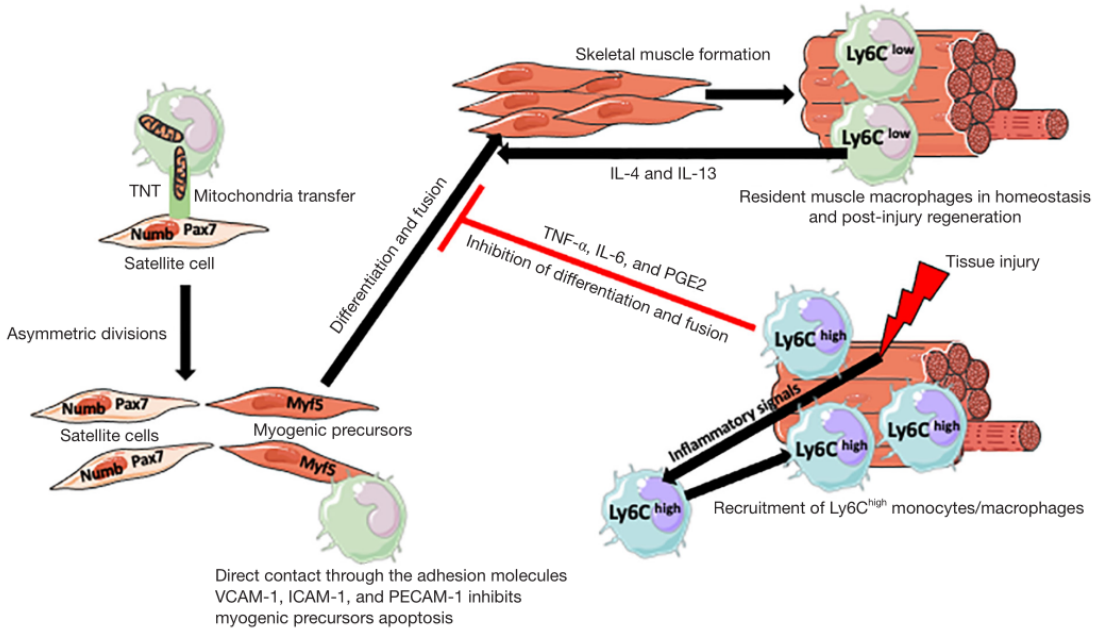

Figure 1: Interactions Between Satellite Cells, Myogenic Precursor Cells, and Macrophages in the Muscular Regeneration Process. Source: Figure 4 from "Macrophages and stem/progenitor cells interplay in adipose tissue and skeletal muscle: a review" by Kloc et al [1]

Figure 1 shows the roles different cells play in muscular regeneration. Satellite stem

cells and macrophages play a key role in the muscular regeneration process. Within skeletal muscle cells are small populations of satellite stem cells, which divide symmetrically into new satellite stem cells, or asymmetrically into both satellite stem cells and muscle fibre cells [1]. During muscle injury, macrophages, which are a type of white blood cell, are recruited to promote inflammation to jump start the healing process, and anti-inflammation, which return to the area of injury to a normal state [1]. More specifically, M1 macrophages promote inflammation while M2 macrophages promote anti-inflammation.

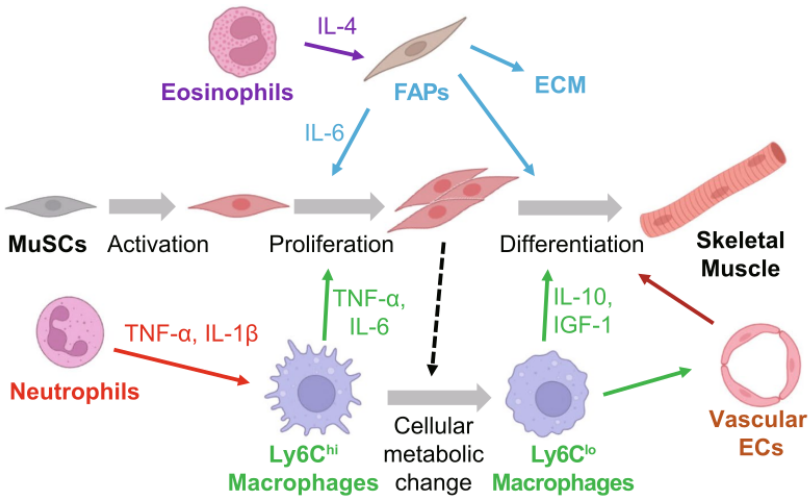

Figure 2: Interactions Between EC's, FAP's, and Neutrophils during muscular regeneration. Source: Figure 1 from "Mechanisms of cooperative cell-cell interactions in skeletal muscle regeneration" by Koike et all [2]

As shown in Figure 2, there are a few other cell types that are also involved in the muscular regeneration process. Endothelial cells (EC's) are responsible for forming new blood vessels during the healing process; Fibroadipogenic Progenitor Cells (FAP's) are able to differentiate into fibroblasts, which are connective tissue, adipocytes, which are fat cells, and myofibroblasts, which are scar-forming cells; Neutrophils are a type of white blood cell that are the first responders to muscle injury [3]. All of these cells work together to heal damaged muscle tissue. However, the ageing process alters the functioning of these cells, which can lead to a variety of adverse health conditions. As such, it is important to understand how the GRN changes as cells age. This kind of knowledge can help find the genes that are most affect by age, as well as lead to further advancements in the field of regenerative medicine.

The purpose of the research is to identify how ageing affects the GRN by identifying candidate genes that are differentially expressed in aged muscle tissue. The statistical methods are first apply to a muscle dataset involving knockout mice, in particular the gene CC2R [9]. Mice are often used as test subjects in biological research due to their genetic similarity to humans. This serves as the training phase.

One of the challenges is dealing with underpowered datasets with small sample sizes. The datasets are high-dimensional and cover tens of thousands of genes, but there are only a few specimens of mice that have actually been tested. This makes it difficult to obtain statistically significant results.

1.1 Background and Literature Survey

There is plenty of research that covers the biological, statistical, and methodological aspects of the research. The main topics covered in the literature survey are a paper that also reviewed the CCR2KO dataset, the muscular regeneration process, how ageing affects the muscular regeneration process, and the statistical methods that are best suited to analyse the data.

1.1.1 CCR2KO Dataset

Recently, a study was done to look at how different cell types work together in the muscular regeneration process, with the test subjects being mice. Published in 2023, the paper entitled "Spatial compartmentalization of signaling imparts source-specific functions on secreted factors" by Groppa et al [3] specifically looked at the role of VEGFA, a protein that promotes the formation of new blood vessels, and how which cell type it originates from affects its functioning. The paper examines the CCR2KO dataset and employs various methods that are commonly used in bioinformatics, such as Gene Ontology (GO) analysis and network analysis. Although the dataset used in this paper is utilised as the training dataset in this research, the research will be using a different methodology, as the significant genes found in the CCR2KO dataset are a means to helping confirm the results of the Ageing Muscle dataset.

The main findings of the study are that the origin of VEGFA affects its function, even though it is the same protein. For instance, the researchers found that VEGFA from macrophages, but not from FAP's, are important for muscular regeneration. However, the study did acknowledge that the location of the cells is highly correlated with the cell type, meaning that location could be a confounding variable concerning VEGFA function. This is a relevant issue, as the study looks at individual cells, meaning that location could be a significant factor in the data. The number of significant genes number in the many hundreds, as made clear in their supplementary materials.

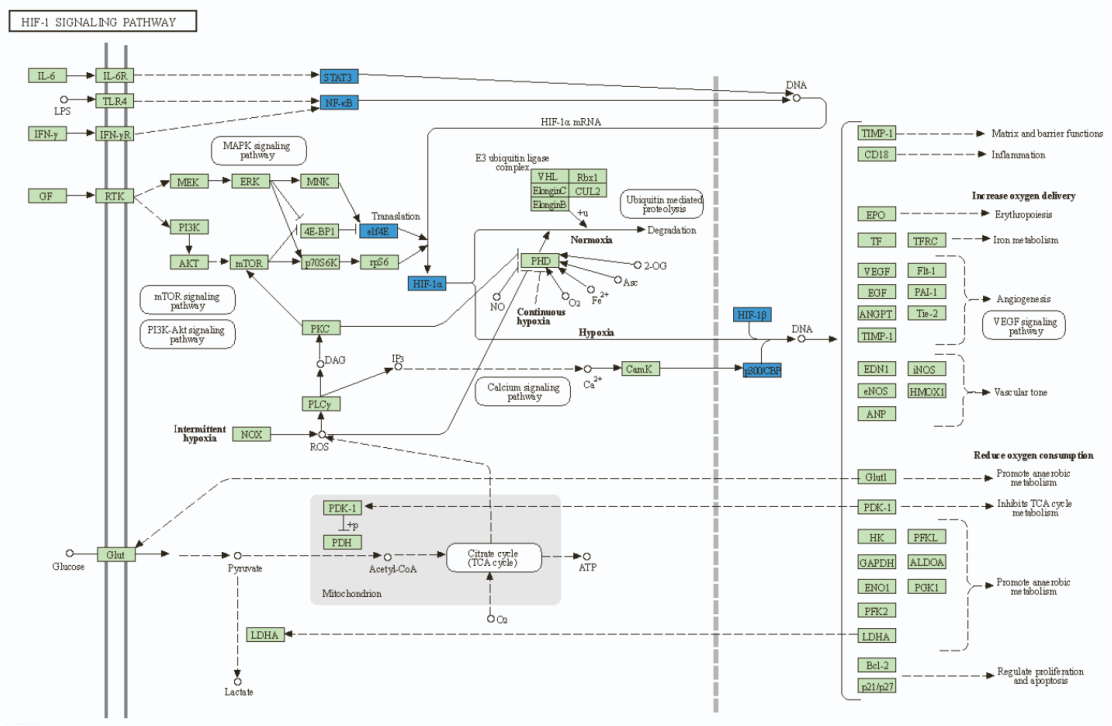

Figure 3: Network of Interactions Between Different Cell Types in the Muscular Regeneration Process. Source: Figure S6, subset D, from "Spatial compartmentalization of signaling imparts source-specific functions on secreted factors" by Groppa et al [3]

The study utilises a plethora of exploratory data analysis techniques, many of which are essential for gene expression analysis. For instance, dimensionality reduction is performed, which makes it easier to cluster cell types and find patterns in the data. Time-series analysis is also performed, which is essential for finding out how gene expression changes throughout the muscular regeneration process. GO analysis helps identify the function of different types of gene expression, allowing for the classification of signalling pathways into different types of biological processes. As shown in Figure 3. The study also builds a network of intercellular signalling, which helps visualise the upregulation and downregulation of VEGFA during the muscular regeneration process.

1.1.2 Muscular Regeneration Process

Regarding how ageing affects the muscular regeneration process, a study entitled "Control of satellite cell function in muscle regeneration and its disruption in ageing" by Sousa-Victor et al [4] provides a comprehensive overview of known differences in the functioning of muscular regeneration between young and old mice specimens. The ageing process is known to affect the GRN in a manner that disrupts the muscular regeneration process. This article is less intensive in terms of methodology and statistics, but is still valuable for covering the biological mechanisms that are affected by ageing. As such, the

study provides clues as to the specific TF's and signalling pathways that are affected by ageing. This will make it easier to narrow down genes of interest in my own research.

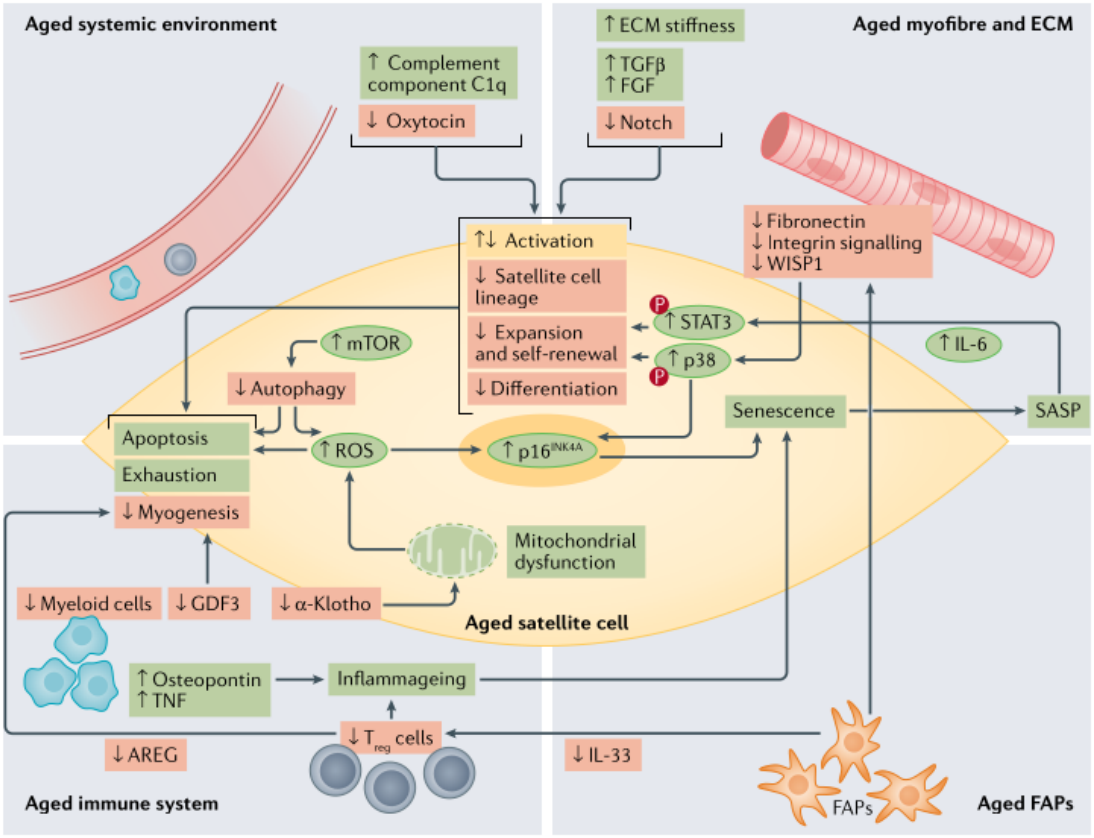

Figure 4: Decline of Satellite Cell Function due to Aging. Source: Figure 4 from "Control of satellite cell function in muscle regeneration and its disruption in ageing" by Sousa-Victor et al [4]

Figure 4 shows how satellite stem cells decline in function due to aging. In aged subjects, there is a quicker depletion of satellite stem cells due to various malfunctions. Aged satellite stems cells produce more FGF2, a TF that leads to activation of satellite stem cells, than younger cells. This leads to constant activation, which results in depletion of satellite cells and increase apoptosis, which is cell death. As such, there are fewer satellite stem cells available when injury occurs for real [4]. Similarly, aged satellite stem cells also produce mor mTORC1, a TF that promotes proliferation and differentiation of satellite stem cells. It similarly depletes the satellite stem cell population. In very old mice, some satellite stem cells enter a presenescent state, which is the state right before senescence, the state where cells stop dividing for good.

Apart from the depletion of the satellite stem cell population, the reduced regenerative potential of the remaining satellite stem cells is another issue for aged cells. A

key sign of withdrawal from the muscular regeneration cell cycle is the de-repression of the $p16^{INK4A}$ locus of the genome, which is a tumour suppressor gene. The lessening of the repression of said locus leads to a higher production of $p16^{INK4A}$, which leads to increased senescence of satellite stem cells. Aberrant autophagy, which is the process of consuming damaged organelles and proteins, and mitochondrial dysfunction also hinder the regenerative abilities of satellite stem cells. Although it is known that ageing is responsible for these changes, the exact causal mechanisms at the molecular level are not yet clear.

1.1.3 Gene Inhibition Treatment for Ageing Muscles

Treating age-related muscle loss, also called sarcopenia, is a major goal in the field of regenerative medicine. After 50 years of age, humans lose an average of 15 to 30 of their muscle mass in each subsequent decade [5]. This is relevant, as it helps to understand the current state of the art research and treatments for age-related muscle issues.

A study by Palla et al [5] examined the effects of inhibiting the enzyme 15-PGDH in aged mice, leading to increased muscle mass and strength. Enzymes are proteins that catalyse biochemical reactions [10]. The main mechanism is that PGDH inhibition results in higher levels of prostaglandin E2 (PGE2), which is responsible for muscle growth. As such, the study does open the possibility of inhibiting 15-PGDH in humans to treat sarcopenia. However, the sample size of the study is quite low, being anywhere from 4 to 14 depending on the experiment. This makes it difficult to generalise the results beyond the study. Figure 5 shows the effects of 15-PGDH inhibition on muscle mass and strength.

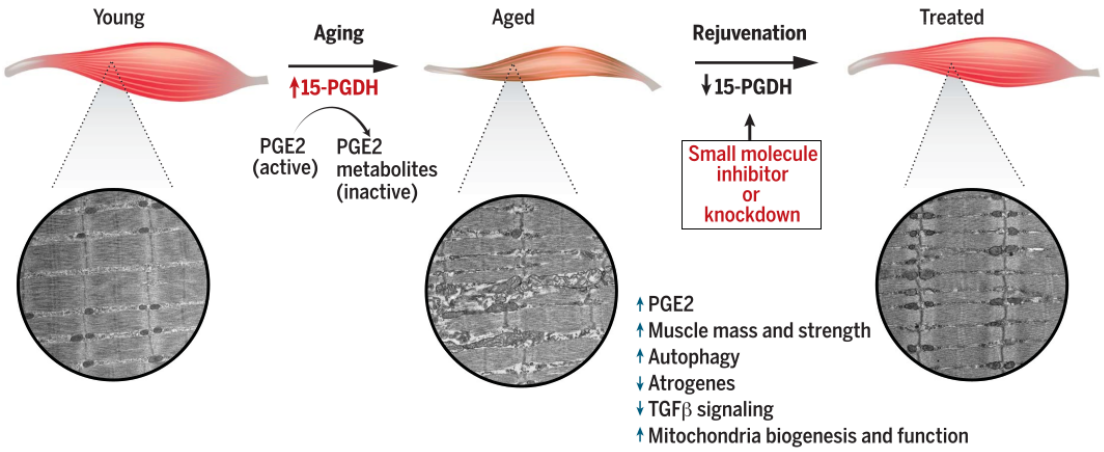

Figure 5: Effects of 15-PGDH Inhibition on Muscle Mass and Strength. Source: Figure 1 from "Inhibition of prostaglandin-degrading enzyme 15-PGDH rejuvenates aged muscle mass and strength" by Palla et al [5]

As much of a breakthrough as this study is, there are a few limitations. The study does not look at the overall GRN and instead focuses on the specific effects of PGDH

inhibition. The sample sizes are normal for biological research, but are small for statistical analysis. This can lead to low statistical power, which makes it hard to detect and replicate new phenomena.

1.1.4 Dealing with Low Statistical Power

As noted, low sample sizes are common in biological research. This results in low statistical power, which is the probability that a test will correctly reject a null hypothesis that is false. In the context of a COVID-19 test, strong statistical power means that the test will correctly identify a person as having COVID-19 if they are infected with the virus. In the context of gene expression analysis, strong statistical power means that the test will correctly find genes that are actually differentially expressed in aged muscle tissue. However, low statistical power means that the analysis will have a difficult time finding genuine differences in gene expression.

An article entitled "Sample Size, power and effect size revised: Simplified and practical approaches in pre-clinical, clinical and laboratory studies" by Serdar et al [11] discusses the limitations posed by underpowered data and potential solutions. The article recommends a minimal sample size of 5-7 animals provides a sufficient amount of degrees of freedom, as well as an extra 10% increase in sample size to account for attrition of test subjects.

In the article "Publication bias impacts on effect size, statistical power, and magnitude (Type M) and sign (Type S) errors in ecology and evolutionary biology" by Yang et al [12], the authors discuss how small samples sizes lead to results with poor replicability. Small sample sizes are common due to the high cost, large resource requirements for biological research, and lack of homogeneity within sample groups [12]. The article says that the potential for bias is so large that it can lead to overestimating the effect size and sign errors for the observed effects. This is a major problem, as it can lead to false positives in the data. That said, it is still possible to utilise different analytical methods to gauge the significance of the results.

The article "Bayesian statistics and modelling" by van de Schoot et al. explains how Bayesian statistics can be effective in analysing data with small sample sizes [13]. The main advantage of Bayesian methods is the use of priors, which incorporates previously established knowledge on the behaviours of certain variables. This can help reduce the uncertainty of the data, which is especially helpful when small sample size leads to low statistical power.

2 Objectives, Specification and Design

2.1 Objectives

The main objectives are primarily related to the data analysis, but are also dependent on the biological context of the data. Although some of the biological implications of the data will be investigated, the research mainly attempts to find significant genes without directly incorporating strong preconceptions about the existing knowledge on the Gene Regulatory Network. This is done to avoid any biases during the research.

Research into the Gene Regulatory Network is usually done to find specific causal mechanisms for various functions. In this case, the research examines how ageing affects how muscle regeneration works, particularly when there is a muscle injury. Successful findings can lead to further breakthroughs, such as treatments for muscle injuries or diseases. If the findings are specific enough, such treatments can even be customised for the genetic profiles of individual patients.

Apart from the identification of relevant genes, the research will also examine the performance of various statistical methods on the underpowered datasets. The evaluation will determine how similar the results are between the various methods, as well as how well the results match the existing knowledge and expectations of the Gene Regulatory Network.

2.2 Specification

The problem that the research question can be formally defined as follows. For each gene i covered by the control and experimental groups in the dataset, the following null and alternate hypotheses are tested:

- **Null Hypothesis:** $H_0 : \mu_{i,c} = \mu_{i,t}$
- **Alternate Hypothesis:** $H_a : \mu_{i,c} \neq \mu_{i,t}$

Where $\mu_{i,c}$ represents the mean expression level of gene i in the control group, and $\mu_{i,t}$ represents the mean expression level of gene i in the treatment group. However, each of the statistical methods used in this research has its own way of testing these hypotheses.

There are two main datasets used in this research: the CCR2KO dataset [9] and the Ageing Muscle dataset [14]. Although the two datasets have different treatments, their similarities in structure and content make them suitable for comparison.

Dataset	**Cell Types**	**Treatment**
CCR2KO	EC, FAP, MP, IC, PER	WT, CCR2KO
Ageing Muscle	MuSC, EC, FAP, Neutrophils, M1, M2	Young, Old
Common Cell Types	**EC, FAP, MP/MuSC**	

Table 1: Cell Types and Treatments in the CCR2KO and Ageing Muscle Datasets

As explained in Table 1, the CCR2KO dataset contains the following cell types: Endothelial Cells (EC's), Fibro-Adipogenic Progenitor Cells (FAP's), Myogenic Progenitor Cells (MP's), Inflammatory Cells (IC's), and Pericytes (PER's). However, only the EC's, FAP's, and MP's have both the Wild Type (WT), which is the control group, and the CCR2KO (CCR2KO), which is the treatment group.

There are a total of 24421 genes in the CCR2KO dataset. The gene counts are tracked over a period of 14 days from injury from injury, so for each gene, there is exactly one count for each day from injury. However, the exact sets of day from injury vary. Generally, there are more days from injury from injury recorded in the CCR2KO subsets than the WT subsets. This discrepancy can be resolved by only focusing on the days from injury present in both subsets. Furthermore, the sample ID's vary between subsets of the same cell type, so only the common sets of sample ID's are used as well.

The Ageing Muscle dataset contains the following cell types: Muscle Stem Cells (MuSC's), Endothelial Cells (EC's), Fibroadipogenic Progenitor Cells (FAP's), Neutrophils, Pro-inflammatory Macrophages (M1), Anti-inflammatory macrophages (M2). The young mice are the control group, and the old mice are the treatment group.

The Ageing Muscle dataset contains a total of genes. The gene counts are tracked over a period of 7 days, with each gene having exactly one gene count for each day. Unlike the CCR2KO dataset, the control and treatment groups have matching days from injury, but the number of days from injury tracked vary by cell type. Neutrophils have only Day 2 tracked, making it unsuitable for temporal analysis. Despite a 7-day window of observation, there are only a maximum of 4 unique days from injury tracked for each cell type. This renders most standard temporal analysis methods unsuitable, but there are still other ways to incorporate time as a variable. Sample ID's are usually consistent between treatment types, but occasionally a specific combination of cell type, treatment, and day from injury is missing. This can be resolved by dropping the corresponding sample ID from the dataset.

The common cell types across both datasets are EC, FAP, and MP/MuSC. Note that Myogenic Progenitor Cells (MP's) and Muscle Stem Cells (MuSC's) are often used interchangeably in research and refer to the same cell type for the purposes of comparative analysis [15].

2.3 Design

Both the CCR2KO and Ageing Muscle datasets are analysed using the same three statistical methods. The same descriptive statistics and visualisations are used for both datasets.

The data needs to be preprocessed before it can be analysed. The counts are first rounded to the nearest integer, as most software packages only work with integer count values. The datasets are then restructured by partitioning them by cell type and treatment. Metadata files are generated, which contain information about the cell types, treatments, days from injury, and sample ID.

The three statistical methods of analysis are as follows. Note that the full formulas and design of the statistical methods are explained in Section 3.4:

- **DESeq2**: This is a package available on R through Bioconductor. It uses a design formula to model the log-fold changes of the expression counts gene-by-gene, as well as a negative binomial distribution to model the counts. The results are measured using both p-value and p-adj, which factors in the decrease in statistical power due to multiple testing. The results are then filtered by p-adj < 0.05 to find the significant genes.
- **Limma-Voom**: This is a package available on R through Bioconductor. It also uses a design formula to model the log-fold changes of the expression counts gene-by-gene. The design formula is what "Limma" (Linear Modelling) refers to. There is also the use of mean-variance modelling, which is what "Voom" (Variance modelling at the observational level) refers to. Limma-Voom can also use p-adj to filter the results.
- **Bayesian Generalised Linear Model (GLM)**: This method utilises the rstanarm package in R. It incorporates a design formula, along with prior distributions of the intercept and slope. Markov Chain Monte Carlo (MCMC) Sampling is used to create posterior distributions for the parameters of the model. The model analyses each gene individually, with significant genes determined by the 95% credible interval of the posterior distribution.

With respect to the CCR2KO dataset, the formula for the design of the statistical methods are as follows:

$$\text{DESeq2 Version 1: } Raw\_Count_i = Treatment_i + Time_i + Treatment_i \times Time_i \quad (2.1)$$

$$\text{DESeq2 Version 2: } Raw\_Count_i = Treatment_i + Time_i \quad (2.2)$$

$$\text{Limma-Voom: } log2FoldChange_i = Group_i \quad (2.3)$$

$$\text{Bayesian GLM: } \Delta log2FoldChange_i = Time_i \quad (2.4)$$

For the DESeq2 design formulas, i is the gene, Raw_Count_i represents the base 2 log-fold change, $Treatment_i$ represents the cell type, and $Time_i$ represents the day from injury. The interaction term $Treatment_i \times Time_i$ is included to account for the effect of the treatment on the gene expression over the days from injury. Note that the model itself converts the raw counts to the base 2 log-fold change before producing the results.

Version 2 of the DESeq2 design formula is almost identical to Version 1, except that the interaction tern $Treatment_i \times Time_i$ is removed. This is done to lower the threshold for significant, as the interaction term can sometimes be too stringent [16]. Furthermore, the interaction terms complicates explainability, even if the interaction term is significant.

For the Limma-Voom design formula, i is the gene, $log2FoldChange_i$ is the base 2 log of the gene counts, and $Group_i$ represents coefficients for specific combinations of days from injury and cell type. As such, the design formula can capture specific time points for further analysis into how the treatment affects gene expression.

For the Bayesian GLM design formula, i is the gene, $\Delta log2FoldChange_i$ is the base 2 log of the difference between the CCR2KO and WT counts, and $Time_i$ represents the day from injury. The model is designed to capture the effect of the treatment on gene expression over the days from injury.

With respect to the Ageing Muscle dataset, the formulas for the designs of the statistical methods are as follows:

$$\text{DESeq2 Version 1: } Raw\_Count_i = Treatment_i + Time_i + Treatment_i \times Time_i \quad (2.5)$$

$$\text{DESeq2 Version 2: } Raw\_Count_i = Treatment_i + Time_i \quad (2.6)$$

$$\text{Limma-Voom: } log2FoldChange_i = Group_i \quad (2.7)$$

$$\text{Bayesian GLM: } \Delta log2FoldChange_i = Time_i \quad (2.8)$$

For Version 1 of DESeq2 design formula, i is the gene, Raw_Count_i represents the base 2 log-fold change, $Treatment_i$ represents the age category of the mouse, and $Time_i$ represents the day from injury. The interaction term $Treatment_i \times Time_i$ is included to account for the effect of ageing on the gene expression over the days from muscle injury. Note that the model itself converts the raw counts to the base 2 log-fold change before the results are finalised. It must be noted that due to the lack of the temporal dimension for the Neutrophil cell type, the formula used is simply $log2FoldChange_i = Treatment_i$.

Similar to the DESeq2 formulas in the CCR2KO dataset, Version 2 of the DESeq2 design formula is almost identical to Version 1, except that the interaction tern $Treatment_i \times Time_i$ is removed. This is also done to lower the threshold for significant, as explained previously in Section 2.3.

For the Limma-Voom design formula, i is the gene, $log2FoldChange_i$ is the base 2 log of the gene counts, and $Group_i$ represents coefficients for specific combinations of days from injury and age category. As such, the design formula can capture specific time points for further analysis into how ageing affects gene expression.

For the Bayesian GLM design formula, i is the gene, $log2FoldChange_i$ is the base 2 log of the difference between the old and young counts, and $Time_i$ represents the day from injury. The model is designed to capture the effect of ageing on gene expression over the days from muscle injury.

2.4 Rejected Methods

Given the high-dimensional nature of the data, alternative methods of analysis were considered. However, these methods were rejected for various reasons.

Tree-based methods, such as random forest, were considered for their ability to have very fine control over the model complexity, and for their suitability for non-linear data. However, the low sample size of the datasets made it impractical, as random forest has a tendency to overfit the training data, even when there is a large and diverse dataset [17]. Explainability was also a challenge, as some variants such as random forest use aggregating methods to make predictions.

Neural networks were also considered, given their ability to embed high-dimensional data into a lower-dimensional space. However, the lack of interpretability of neural networks, and it's unsuitability for small datasets [18], make them impractical for this research.

Data augmentation could work in increasing the size of the dataset, but this is highly dubious given the complexity of RNA-Seq data. A recent study shows that even a sample size of 10 is challenging for data augmentation, as the data is highly structured and has a high-dimensional nature [19]. Essentially, the signal-to-noise ratio is too low for data augmentation to be effective for both the CCR2KO and Ageing Muscle datasets. This effectively rules out any machine learning model and traditional statistic models that require large sample sizes.

3 Methodology and Implementation

3.1 Define Research Objectives

As stated in Section 2.1, the research aims to find differentially expressed genes related to ageing muscle. The research also aims to evaluate how the three different statistical methods perform on the datasets. To a lesser extent, the research aims to find the biological implications of the differentially expressed genes.

The reason for emphasising the statistical aspect of the research over the biological is due to the complexity of the GRN. The current inference methods are highly complex, with methods including, but not limited to, Boolean networks, differential equations, and advance machine learning and deep learning techniques [20]. Aside from the fact that the sample size of the datasets used in the research are too small for these methods, the latest methods still struggle to distinguish direct and indirect regulatory relationships between genes [21].

Furthermore, genes do not directly interact with each other, but instead through TF's, which further complicates the inference process [20]. As such, the research is focused on finding significant genes without extensive prior assumptions on the mechanics of the GRN, as well as not delving too deep into the implications of the findings with regards to the functioning of the overall GRN.

3.2 Design the Statistical Methods

3.3 Preprocess the Data

As explained in Section 2.3, the data needs to be preprocessed by rounding the counts to the nearest integer, restructuring the datasets by partitioning them by cell type and treatment, and generating metadata files. These steps are necessary, as different statistical methods require different data structures, and the metadata files are necessary for the EDA methods. In some cases, the log-fold changes are filtered by thresholding, mainly if the base-2 log-fold changes are greater than 1. This is done to reduce genes with minimal to no changes in the gene expression. As such, not all genes will have results calculated by the statistical methods, but those genes would not have significant changes in the gene expression anyway.

3.4 Exploratory Data Analysis (EDA)

The EDA methods are mainly diagnostic in nature to ensure that the data is adequately prepared for further analysis. The EDA methods include Volcano Plots, Multidimensional scaling (MDS) Plots, Principal Component Analysis (PCA) plots, Time Plots, and Heatmaps with Agglomerative Clustering.

Volcano Plots serve as a strong diagnostic tool regarding the distribution of the p-adj values and the log-fold changes. In the research, the volcano plots plot p-adj against the log2fold change in the gene expression counts. This is a great way to control for multiple testing errors, as p-adj factors in the False Discovery Rate [22]. Volcano plots are also great at detecting outliers, examining other selection criteria such as t-stat, and examining the variance of DEG counts [23].

PCA plots are used to visualise the variance of the predictors in the dataset. In this research, the PCA plots are applied to the full datasets, so that cell type clustering can be visualised in addition to treatment and days from injury. The PCA plots are useful when comparing to existing knowledge of the GRN, as existing knowledge of how ageing affects muscle regeneration can be used to infer the biological significant of the clustering in the PCA plots.

Time Plots are used to visualise the gene expression counts over the days from injury. They are done gene-by-gene and can include multiple plot lines for different sample ID's and different treatments. The scale of the gene expression counts will be in log base 10 to better accentuate differences in low counts, and to better visualise the changes in counts across different scales.

The MDS Plots and the Voom Mean-variance Trend Plots are great for identifying clusters and testing statistical assumptions. MDS Plots are great at dimensionality reduction, allowing the data to be represented in a two-dimensional feature space as evident in Figures 22 and 24 [24].The Voom Mean Variance Plot plot the square root of the standard deviations of the genes against the base-2 log of the counts of each gene. If the trend-line is rather flat and the distribution of the residuals is rather uniform, then the data is sufficiently unbiased [25].

The Heatmaps with Agglomerative Clustering are effective in showing the relationships between the various predictors, as well as indicating the upregulation and downregulation of different genes of interest. The horizontal axis show the various possible values for the predictors, while the vertical axis shows the different significant genes. The colour blue indicates downregulation, or count suppression, and the colour red indicates upregulation, or count stimulation.

3.5 Differential Expression Analysis (DEG)

The DEG methods are the main focus of the research, as they are used to find the significant genes. As explained before, the DEG methods are DESeq2, Limma-Voom, and Bayesian GLM.

3.5.1 DESeq2

This model serves as the baseline due to its simplicity and wide use within bioinformatics. Concerning the mathematics and statistics, DESeq2 is essentially a GLM model with a negative binomial distribution.

The definition of a GLM model in the context of the research is as follows:

$$\text{General Linear Model (GLM): } y_i = \sum_{j=1}^{p} x_{i,j}\beta_j + \epsilon_i \tag{3.1}$$

Where i is the sample gene, j is the predictor, y_i is the gene expression count, x_{ij} is the predictor value, β_j is the regression coefficient, and ϵ_i is the error term.

The definition of a negative binomial distribution is as follows:

$$\text{Negative Binomial: } y_i \sim \text{NB}(\mu_i, \alpha_i) \tag{3.2}$$

Where i is the sample gene, y_i is the gene expression count, μ_i is the mean expression count, and α_i is the dispersion parameter. The dispersion parameters is defined as the variance of the negative binomial distribution divided by the mean of the negative binomial distribution. This makes the negative binomial distribution suitable for the count data, as the variance of the count data can sometimes be greater than the expected variability in the statistical model. This is called overdispersion, and if left untouched, overdispersion can lead to false positives in the results [26]. The DESeq2 model uses p-adj < 0.05 to filter the results and find the significant genes.

DESeq2 is suitable for RNA-seq data with small sample sizes, mainly for its use of empirical Bayes shrinkage to reduce false positives and stabilise the estimations of the dispersion parameters [27]. Thus, it is an effective baseline method for this research.

3.5.2 Limma-Voom

Limma-Voom is a more advanced model than DESeq2, as it incorporates mean-variance modelling. This means that the model can factor in the variability of the counts.

The definition of mean-variance modelling is as follows:

$$\text{Var}(y_i) = \phi \cdot V(\mu_i) \tag{3.3}$$

Where i is the sample gene, y_i is the gene expression count, μ_i is the mean expression count, ϕ is the dispersion parameter, and $V(\mu_i)$ is a known function of the mean. Like DESeq2, Limma-Voom is a GLM model, and its mean-variance modelling is similar in nature to the negative binomial distribution. However, the Limma-Voom model is much more flexible in terms of the design formula.

$$\text{Limma-Voom: } y_i = \beta_0 + \sum_{j=1}^{p} x_{i,j}\beta_j + \epsilon_i \tag{3.4}$$

Where i is the sample gene, j is the first predictor, k is the second predictor, y_i is the base 2 log transformed count, x_{ij} is the first predictor value, x_{ik} is the second predictor value, $\beta_{j,k}$ is the regression coefficient, and ϵ_i is the error term.

As stated in Section 2.3, the Limma-Voom model is being used to capture the interaction between the predictor variables, which are treatment and days from injury. As such, each unique combination of treatment and days from injury is treated as a separate predictor, and the regression coefficients are calculated for each unique combination. This allows for a more detailed analysis of how the treatment affects the gene expression over the days from injury. Just like the DESeq2 model, Limma-Voom also uses p-adj $<$ 0.05 to filter the results and find the significant genes.

Limma-Voom is suitable for RNA-seq data with small sample sizes, mainly for its use of mean-variance modelling to reduce false positives and stabilise the estimations of the dispersion parameters [25]. Thus, it is an effective model for this research.

3.5.3 Bayesian GLM

The Bayesian GLM model is the most advanced model of the three, as it incorporates prior distributions of the intercept and slope and Markov Chain Monte Carlo (MCMC) Sampling to create posterior distributions for the parameters of the model.

MCMC sampling is a method of sampling from a probability distribution [28]. The underlying assumption is that there is a hidden low-dimensional structure to the high-dimensional data. More specifically, the rstanarm package used in the research utilises the Hamiltonian Monte Carlo (HMC) algorithm, specifically the No-U-Turn Sampler (NUTS) variant, to sample from complex probability distributions [29].

Below is the formal definition of the Hamiltonian Monte Carlo (HMC) algorithm [28]:

1. Let $\pi(\theta)$ be the target distribution (posterior) we want to sample from, where $\theta \in \mathbb{R}^d$.

2. Introduce an auxiliary momentum variable, $p \in \mathbb{R}^d$.

3. Define the Hamiltonian, which is the sum of the potential energy and kinetic energy:
$$H(\theta, p) = -\log \pi(\theta) + \frac{1}{2} p^T M^{-1} p \tag{3.5}$$
where M is a mass matrix (often set to the identity matrix).

4. The dynamics of the system are described by Hamilton's equations:
$$\frac{d\theta}{dt} = \frac{\partial H}{\partial p} = M^{-1} p \tag{3.6}$$
$$\frac{dp}{dt} = -\frac{\partial H}{\partial \theta} = \nabla_\theta \log \pi(\theta) \tag{3.7}$$

5. These equations are typically solved numerically using the leapfrog integrator:
$$p_{t+\epsilon/2} = p_t + \frac{\epsilon}{2} \nabla_\theta \log \pi(\theta_t) \tag{3.8}$$
$$\theta_{t+\epsilon} = \theta_t + \epsilon M^{-1} p_{t+\epsilon/2} \tag{3.9}$$
$$p_{t+\epsilon} = p_{t+\epsilon/2} + \frac{\epsilon}{2} \nabla_\theta \log \pi(\theta_{t+\epsilon}) \tag{3.10}$$
where ϵ is the step size.

6. After L steps, the proposal is accepted with probability:
$$\min(1, \exp(H(\theta_0, p_0) - H(\theta))) \tag{3.11}$$

The HMC algorithm takes inspiration from physics, specifically with its use of energy conservation. The potential energy is the negative log of the posterior distribution, whereas the kinetic energy is the $1/2$ times the squared momentum divided by the mass. The Hamiltonian is defined as the sum of the potential and kinetic energy. The dynamics of the system are described by the Hamiltonian equations, which are solved numerically using the leapfrog integrator.

The proposals are different samplings of the data, which are generated and evaluated using the Hamiltonian equations. The new proposal, which reflects the new state of the system, is always chosen if it has a smaller Hamiltonian than the current approach, and with a probability exponentially related to the distance between the two proposals if the new proposal has more energy than the current one. The larger the new Hamiltonian is, the less likely it will be accepted.

Below is the formal definition of the No-U-Turn Sampler (NUTS) algorithm:

1. Define the U-turn criterion:
$$\langle \theta_+ - \theta_-, \frac{\partial H}{\partial p}(\theta_+, p_+) \rangle < 0 \tag{3.12}$$

$$\langle \theta_+ - \theta_-, \frac{\partial H}{\partial p}(\theta_-, p_-) \rangle < 0 \tag{3.13}$$

where θ_+, p_+ and θ_-, p_- are the states at the ends of the trajectory.

2. Build a balanced binary tree of states by doubling the number of leapfrog steps until the U-turn criterion is met.

3. Sample uniformly from the set of states in the trajectory that satisfy detailed balance.

4. Adapt the step size using dual averaging to achieve a target acceptance rate:

$$\log \epsilon = \mu - \frac{\sqrt{t}}{\gamma} \frac{1}{t} \sum_{i=1}^{t} (\alpha_i - \alpha^*) \tag{3.14}$$

where μ is the initial value, t is the iteration, γ is the adaptation rate, α_i is the acceptance probability at iteration i, and α^* is the target acceptance rate.

The No-U-Turn Sampler (NUTS) is an extension of the HMC algorithm. The main distinction from the base HMC algorithm is that NUTS adaptively selects the number of leapfrog steps and the step size during the sampling process. In this case, the number of steps increases two-fold at each iteration until the U-turn criterion is met, which is when the trajectory doubles back on itself. The balanced binary tree of proposals is then used to sample uniformly from the set of proposals that satisfy detailed balance. The step size is then calculated using dual averaging to reach a target acceptance rate. This dynamic approach ensures that the proposals are more likely to be accepted.

The Bayesian GLM model is defined as follows:

$$\text{Bayesian GLM: } y_i = \beta_0 + \beta_1 t_i + \epsilon_i \tag{3.15}$$

Where y_i is the log2 fold change in expression for a gene at time point i, β_0 is the intercept, representing the initial log2 fold change, β_1 is the regression coefficient for time, representing the rate of change in log2 fold change over time, t_i is the time point for observation i, ϵ_i is the error term, assumed to be normally distributed: $\epsilon_i \sim N(0, \sigma^2)$

The prior distributions of the Bayesian GLM model are as follows:

$$\beta_0 \sim N(0, 1) \tag{3.16}$$

$$\beta_1 \sim N(0, 1) \tag{3.17}$$

The Bayesian GLM model uses a 95% credible interval for the coefficients, in conjunction with the probability of direction and ROPE (Region of Practical Equivalence) Proportion to find significant genes. The ROPE is defined as the range of values that are practically equivalent to the null hypothesis. In this case, the ROPE is defined as -0.1 to 0.1.

The Bayesian GLM model is suitable for RNA-seq data with small sample sizes, mainly for its use of MCMC sampling to create posterior distributions for the parameters

of the model [30]. The main advantage is the incorporation of data across multiple genes to provide an overall estimate of the parameters. Thus, it is an effective model for this research.

3.5.4 Gene Ontology (GO) Analysis

GO Analysis is a critical part of the research, as it helps identify the biological functions of select genes. Usually, GO Analysis is only done after sets of significant genes are found, as GO Analysis can be computationally intensive. There are multiple steps in converting the set of significant genes into the GO analysis results, which comprise of a set of GO terms and the genes associated with them.

The first step is to convert the gene symbols into Entrez ID's, which are unique identifiers for genes [31]. A hypergeometric test is performed, which calculates the chance of seeing at least n number of genes from a certain GO category in the list of significant genes [32]. The model factors in the entire set of genes in the organism's genome, in this case mice [31]. The Benjamini-Hochberg (BH) method is used for multiple testing correction, which controls for false discovery rate [33]. The significance thresholds are set at 0.10 for the adjust p-value and 0.20 for the q-value. Note that the adjusted p-value is much more stringent than p-value, which is set at 0.05. According to the original paper on the BH method by Bemjamini and Hochberg, adjusted p-value using the Benjamini-Hochberg method is almost interchangeable in function and strictness with q-value [34].

The steps for GO Analysis are as follows

1. Convert gene symbols to Entrez IDs:

$$\text{Gene Symbol} \rightarrow \text{Entrez ID}$$

2. Perform hypergeometric test:

$$P(X \geq n) = \sum_{i=n}^{\min(N,K)} \frac{\binom{K}{i}\binom{M-K}{N-i}}{\binom{M}{N}}$$

3. Apply Benjamini-Hochberg correction:

$$q_i = \min_{i \leq j \leq m} \left\{ \frac{m}{j} p_{(j)} \right\}$$

4. Set significance thresholds:

$$\begin{aligned} \text{Adjusted p-value} &< 0.10 \\ \text{q-value} &< 0.20 \\ \text{p-value} &< 0.05 \end{aligned}$$

Where N is the number of significant genes, X is the number of genes in the GO category, K is the number of genes in the GO category, M is the total number of genes in the mouse genome, and n is the number of significant genes in the GO category. The hypergeometric test calculates the chance of seeing at least n number of genes from a certain GO category in the list of significant genes. q_i is the adjusted p-value, m is the total number of tests, and $p_{(j)}$ is the j-th ordered p-value. The Benjamini-Hochberg correction is used to control for false discovery rate, and the significance thresholds are set at 0.10 for the adjusted p-value, 0.20 for the q-value, and 0.05 for the p-value.

The main visuals are the GO Enrichment plots, which plot the ratio of genes on the x-axis and the GO terms on the y-axis. Generally, the gene ratio increases the higher up the GO term is. The GO terms are categorical and nominal in nature, and there are many ways to group them together to declutter the GO Enrichment plots.

3.5.5 Evaluation of the Results

The evaluation is not straightforward, as the full functioning of the GRN has not been mapped out yet. There is literature on select genes whose functions are known, but this would hardly constitute as a proper evaluation. The evaluation is mainly done by comparing the results of the three statistical methods in terms of their statistical and biological significance, as well as comparing the results to the existing research on the GRN.

The genes will be filtered using a database entitled the "Human Skeletal Muscle Aging Atlas". The specific dataset from the database is for Mouse muscle, as both the CCR2KO and Ageing Muscle datasets are for mouse muscle. The database contains genes that are known to be associated with ageing muscle, so the proportion of significant genes found in the database can be used to assess the quality of the results. The CCR2KO dataset and the Ageing Muscle dataset do not test the same set of genes, and the proportion of the complete set of genes in the database that are found in the datasets is used as a metric to put the results in perspective.

The similarities of the results of the two datasets will be compared using distance metrics. As significant genes contain both text and numeric data, TF-IDF scores will be used for the description data, and cosine similarity, Jaccard similarity will be used to measure how similar two sets of Gene ID'S are. Ratio similarity will be used to measure how similar two sets of Gene Ratios are. Cosine similarity will be used to measure how similar two distance metrics are. The overall distance metric will be a weighted sum of these distance metrics.

Below are the key equations for the various similarity metrics:

1. TF-IDF:

$$\text{TF-IDF}_{i,j} = \text{TF}_{i,j} \times \text{IDF}_i \tag{3.18}$$

   (a) Term Frequency (TF):

$$\text{TF}_{i,j} = \frac{f_{i,j}}{\sum_k f_{k,j}} \tag{3.19}$$

(b) Inverse Document Frequency (IDF):

$$\text{IDF}_i = \log \frac{N}{n_i} \tag{3.20}$$

2. Cosine Similarity:

$$\cos(A, B) = \frac{A \cdot B}{||A|| \times ||B||} \tag{3.21}$$

3. Jaccard Similarity:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.22}$$

4. Overall set similarity:

$$\begin{aligned} \text{Overall\_set\_similarity} = 0.35 \times \text{Description\_similarity} + \\ 0.1 \times \text{GeneRatio\_similarity} + 0.2 \times \text{geneID\_similarity} + 0.35 \times \frac{\text{Common\_percentage}}{100} \end{aligned} \tag{3.23}$$

TF-IDF is a commonly use distance metric in Natural Language Processing(NLP) [35]. It measures how important a word with index i is to a document with index j. The term frequency (TF) is the frequency of word i in document j divided by the total number of words in document j. The inverse document frequency (IDF) is the logarithm of the total number of documents divided by the number of documents containing word i. The TF-IDF score is the product of the TF and IDF scores. Note that the more often a word appears in a document, the higher the TF score. The less often a word appears across all documents, the higher the IDF score. This means that words that are overabundant and generate little information is penalised, while words that are rare but generate plenty of information are rewarded. For the purposes of evaluation, TF-IDF score is used to identify the importance of specific terms in the descriptions of biological functions for significant genes, and then comparing how similar two sets of significant genes are based on the importance of the terms in the biological function labels.

Cosine similarity is used to compare the distance between two vectors. In this case, it is used to compare the distance between the TF-IDF vectors to measure how similar two sets of Gene Descriptions are. By scoring the similarity of gene descriptions, the research can infer how similar the biological functions of the significant genes are. It constitutes 35% of the overall set similarity metric.

Jaccard similarity is a distance metric that measures how similar two sets of values are [36]. The metric simply calculates the ratio of the intersection of the two sets and the union of the two sets. In the case of the research, Jaccard similarity is used to compare how similar two sets of Gene ID's are per each biological function label from the GO results. It constitutes 20% of the overall set similarity metric.

Gene Ratio and common percentage are much more straightforward. Gene Ratio is the proportion of significant genes that match a biological function label, and common

percentage is the proportion of significant genes shared between two sets of genes. Gene Ratio is aggregated for each specific biological function label, and common percentage is aggregated for all biological function labels. Gene Ratio constitutes 10% of the overall set similarity metric, and common percentage constitutes 35% of the overall set similarity metric.

The reason for weighting the overall set metric as specified in Equation 4.23 is that each individual metric provides some degree of insight into the overall similarity between two sets of genes and their biological functions. Description similarity is the most dynamic metric, as it looks at individual ngrams, which are phrases with n number of terms [37], instead of just the entire biological function label. Thus, there is more nuance, as the metric can capture labels with similar vocabulary instead of only exact matches. That said, exact matches are still valuable, as even minute differences in wording can result in distinct biological functions, which is why common percentage is given equal weighting with description similarity, with both at 35% weighting. GeneRatio is somewhat important, as the higher the gene ratio, the more significant a biological function is within a set of significant genes. However, the set of notable biological functions have cleared the p-adj threshold of 0.05, so the gene ratio is less informative than other metrics. Thus, GeneRatio is only given 10% weighting. GeneID similarity is quite useful, as each biological function label has accompanying GeneID's. The more similar these GeneID's are across multiple datasets for a specific biological function label, the more informative that biological function label is. However, exact matches of GeneID's are rare, as the datasets often yield diverse results. Thus, it is given 20% weighting, as it is more informative than GeneRatio, but less informative than Description similarity and common percentage.

It must be said that the CCR2KO dataset does not directly test for ageing muscle, but the results can be used to infer the effects of ageing muscle. This is because CCR2KO can be used to model the effects of ageing muscle, since CCR2 is known to be associated with muscle regeneration. In fact, inhibiting CCR2 can lead to enhanced aged regeneration and functional recovery [38]. However, the age of the mice in the CCR2KO dataset is not known, so the results are only used to infer the effects of ageing muscle.

After the results from both datasets are compared across three datasets, only results with high enough similarity scores and common GO terms as defined by direct matches will be included in the final results. Note that if the CCR2KO dataset doesn't return sufficient results for a specific method or cell type, then the results from the Ageing Muscle dataset will have to be excluded due to lack of comparison.

4 Results, Analysis and Evaluation

4.1 Results & Analysis

4.1.1 Results Summary

The combined findings summarise the findings, which will be further elaborated in the following sections.

Analysis Method	Subset	All Genes	Significant Genes	Atlas Genes
DESeq2 No Interaction	EC	24421	46	40
DESeq2 No Interaction	FAP	24421	14	7
DESeq2 No Interaction	MP	24421	32	23
DESeq2 With Interaction	EC	24421	510	478
DESeq2 With Interaction	FAP	24421	3397	3125
DESeq2 With Interaction	MP	24421	474	455
Limma Voom	EC 0 Days	15138	0	0
Limma Voom	EC 1 Days	15138	0	0
Limma Voom	EC 2 Days	15138	0	0
Limma Voom	EC 3 Days	15138	0	0
Limma Voom	EC 5 Days	15138	0	0
Limma Voom	EC 6 Days	15138	0	0
Limma Voom	EC 7 Days	15138	0	0
Limma Voom	EC 10 Days	15138	0	0
Limma Voom	EC 14 Days	15138	0	0
Limma Voom	FAP 0 Days	14513	0	0
Limma Voom	FAP 1 Days	14513	0	0
Limma Voom	FAP 2 Days	14513	0	0
Limma Voom	FAP 3 Days	14513	302	278
Limma Voom	FAP 5 Days	14513	0	0
Limma Voom	FAP 6 Days	14513	0	0
Limma Voom	FAP 7 Days	14513	0	0
Limma Voom	FAP 10 Days	14513	158	140
Limma Voom	FAP 14 Days	14513	0	0
Limma Voom	MP 0 Days	15324	0	0
Limma Voom	MP 1 Days	15324	0	0
Limma Voom	MP 3 Days	15324	0	0
Limma Voom	MP 4 Days	15324	0	0
Limma Voom	MP 6 Days	15324	0	0
Limma Voom	MP 10 Days	15324	0	0
Bayesian Inference	EC	18911	489	297
Bayesian Inference	FAP	19195	819	473
Bayesian Inference	MP	18535	539	320

Table 2: CCR2KO Gene Analysis Results

Analysis Method	Subset	All Genes	Significant Genes	Atlas Genes
DESeq2 No Interaction	EC	17797	0	0
DESeq2 No Interaction	FAP	17797	0	0
DESeq2 No Interaction	MuSC	17797	2	2
DESeq2 No Interaction	Inflammatory_Mac	17797	2	2
DESeq2 No Interaction	Resolving_Mac	17797	0	0
DESeq2	Neutrophil	17797	315	315
DESeq2 With Interaction	EC	17797	4832	4832
DESeq2 With Interaction	FAP	17797	4448	4448
DESeq2 With Interaction	MuSC	17797	6137	6137
DESeq2 With Interaction	Inflammatory_Mac	17797	6013	6013
DESeq2 With Interaction	Resolving_Mac	17797	6440	6440
Limma Voom	EC Day 0	16073	13	13
Limma Voom	EC Day 2	16073	36	36
Limma Voom	EC Day 4	16073	97	97
Limma Voom	EC Day 7	16073	91	91
Limma Voom	FAP Day 0	16274	404	404
Limma Voom	FAP Day 2	16274	78	78
Limma Voom	FAP Day 4	16274	336	336
Limma Voom	FAP Day 7	16274	323	323
Limma Voom	Inflammatory_Mac Day 2	13944	0	0
Limma Voom	Inflammatory_Mac Day 4	13944	1	1
Limma Voom	MuSC Day 0	16248	443	443
Limma Voom	MuSC Day 2	16248	3475	3475
Limma Voom	MuSC Day 4	16248	646	646
Limma Voom	MuSC Day 7	16248	153	153
Limma Voom	Resolving_Mac Day 2	14379	29	29
Limma Voom	Resolving_Mac Day 4	14379	7	7
Limma Voom	Resolving_Mac Day 7	14379	34	34
Bayesian Inference	EC	17483	850	850
Bayesian Inference	FAP	17603	1495	1495
Bayesian Inference	Inflammatory_Mac	17648	754	754
Bayesian Inference	MuSC	17646	1667	1667
Bayesian Inference	Resolving_Mac	17126	837	837

Table 3: Ageing Muscle Gene Analysis Results

Dataset	Total Unique Genes	Atlas Genes
CCR2KO	24421	14614
Ageing Muscle	17789	13815

Table 4: CCR2KO and Ageing Muscle Dataset Genes in the Human Skeletal Muscle Ageing Atlas

Figure 6: Gene Analysis Results and Dataset Comparison

4.1.2 Diagnostic Plots

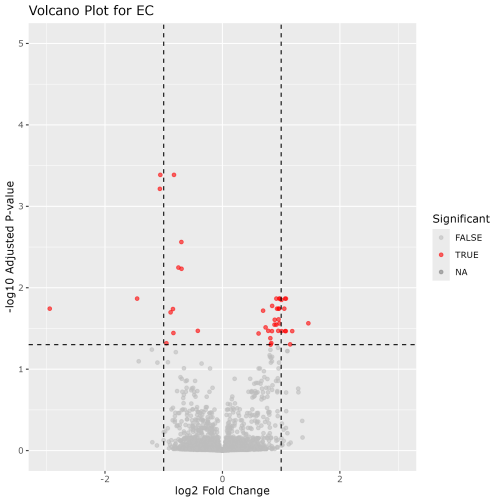

Figure 7: Volcano Plot of EC cell type for the CCR2KO Dataset Using DE-Seq2 With Interaction

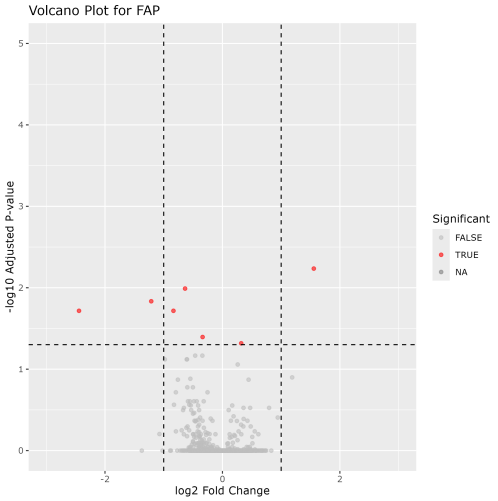

Figure 8: Volcano Plot of FAP cell type for the CCR2KO Dataset Using DE-Seq2 With Interaction

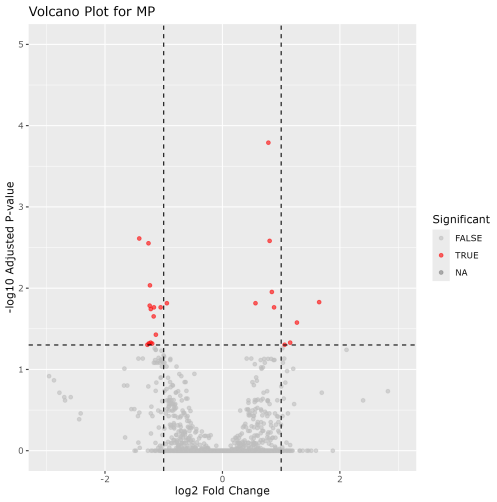

Figure 9: Volcano Plot of MP cell type for the CCR2KO Dataset Using DE-Seq2 With Interaction

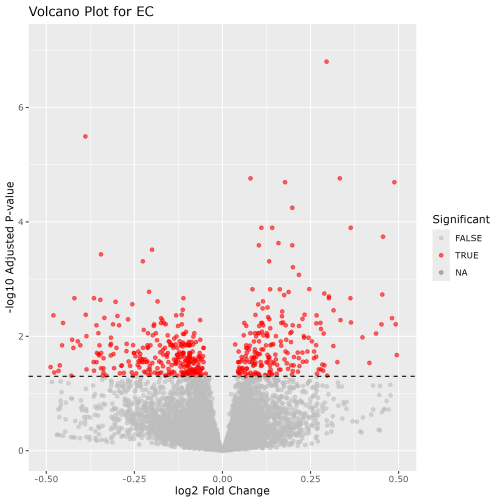

Figure 10: Volcano Plot of EC cell type for the CCR2KO Dataset Using DE-Seq2 No Interaction

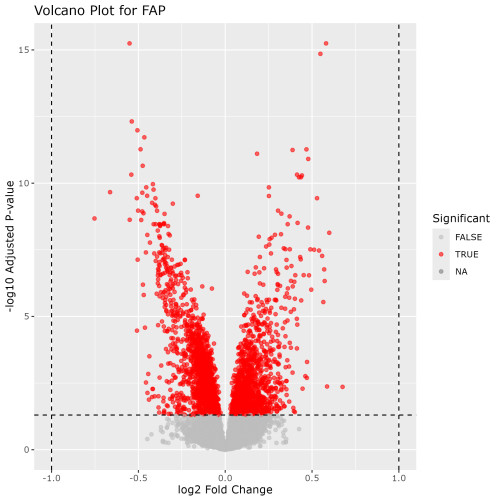

Figure 11: Volcano Plot of FAP cell type for the CCR2KO Dataset Using DESeq2 No Interaction

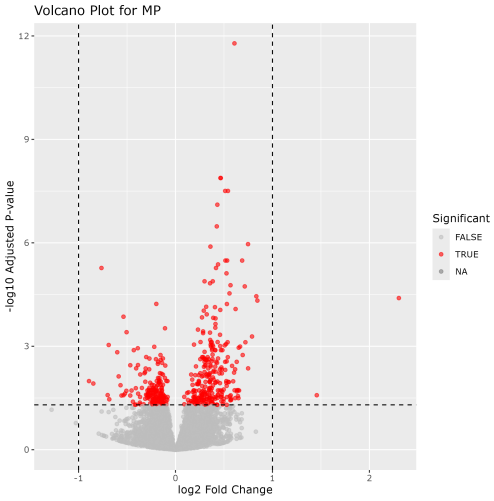

Figure 12: Volcano Plot of MP cell type for the CCR2KO Dataset Using DESeq2 No Interaction

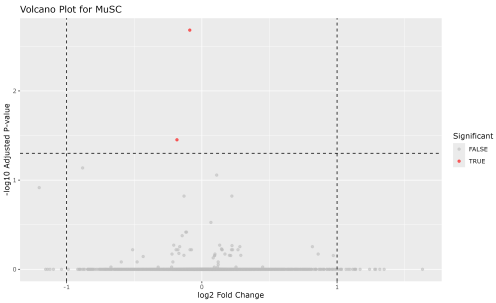

Figure 13: Volcano Plot of MuSC cell type for the Ageing Muscle Dataset Using DESeq2 With Interaction

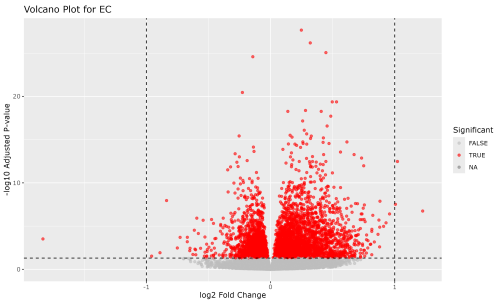

Figure 14: Volcano Plot of EC cell type for the Ageing Muscle Dataset Using DESeq2 No Interaction

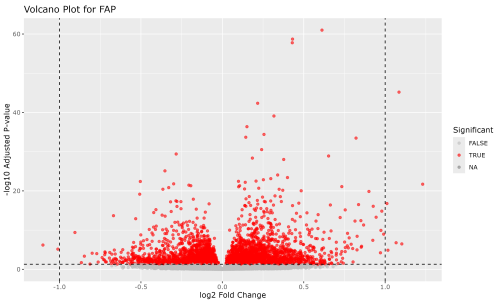

Figure 15: Volcano Plot of FAP cell type for the Ageing Muscle Dataset Using DESeq2 No Interaction

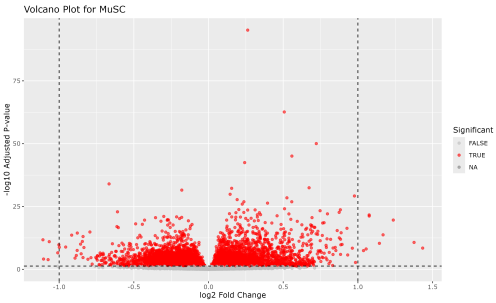

Figure 16: Volcano Plot of MuSC cell type for the Ageing Muscle Dataset Using DESeq2 No Interaction

The diagnostic plots show that despite the small sample size, the datasets meet both the statistical and biological assumptions held in bioinformatics. The volcano plots in Figures 7-16 exhibit the typical funnel-shape of volcano plots, along with varying numbers of outliers as defined by p-adj. As such, the data seems reasonably distributed and is suitable for further analysis.

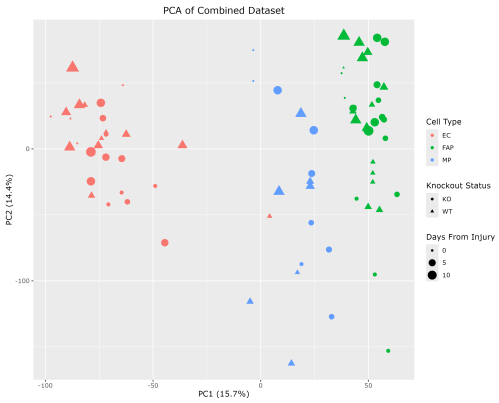

Figure 17: PCA Plot 1 of the CCR2KO Dataset

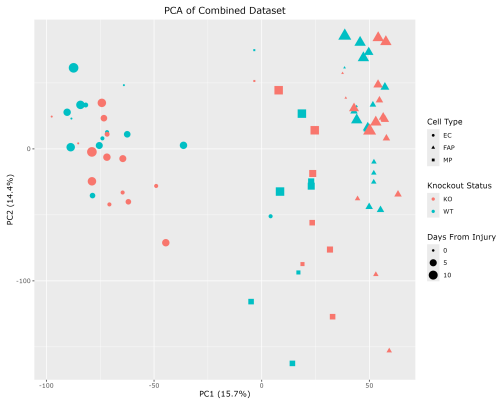

Figure 18: PCA Plot 2 of the CCR2KO Dataset

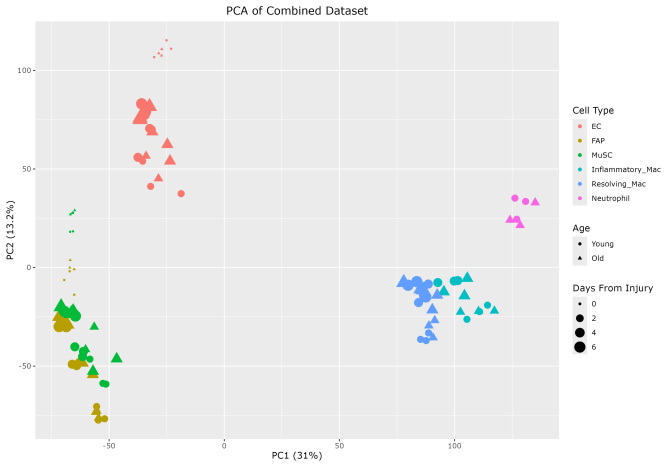

Figure 19: PCA Plot of the Ageing Muscle Dataset

The PCA plots in Figures 17-19 indicate clustering that is consistent with the current understanding of the muscle regeneration process. Previous research has shown that aging negatively affects the functioning of mouse FAP's, which indirectly affects the myogenic potential of MuSC's [39]. This meanings that the ability for MuSC's to become muscle cells is negatively affected by ageing. M1 and M2 cells tend to cluster together, as they are directly involved in the inflammation and resolution phases of muscle regeneration [40]. With the expected clustering occurred, the data is suitable for finer analysis.

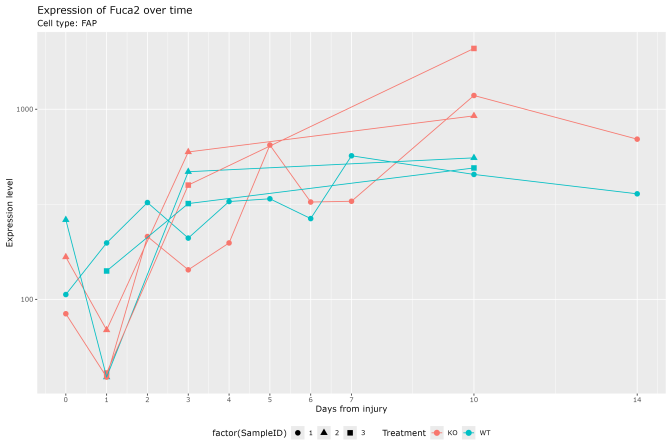

Figure 20: Time Plot of Fuca2 for the FAP cell type in the CCR2KO Dataset

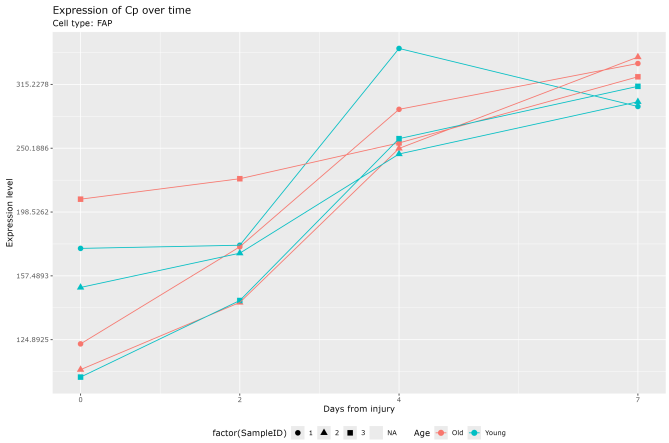

Figure 21: Time Plot of gene Cp for the FAP cell type in the CCR2KO Dataset in logarithmic scale

The time plots in Figures 20 and 21 are for the genes Fuca2 and Cp. Fuca2 was found to be significant by both the Limma-Voom and Bayesian GLM methods in the CCR2KO dataset, while Cp was found to be significant by both the Limma-Voom and Bayesian GLM methods in the Ageing Muscle dataset. The x-axis is the days from injury, while the y-axis is the log-scaled counts. In both cases, the time series deviate significantly when considering the different treatments.

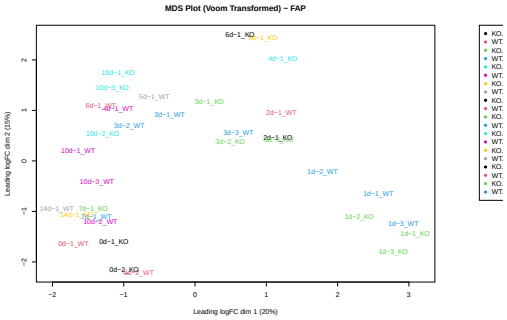

Figure 22: MDS Plot of the FAP cell type in the CCR2KO Dataset

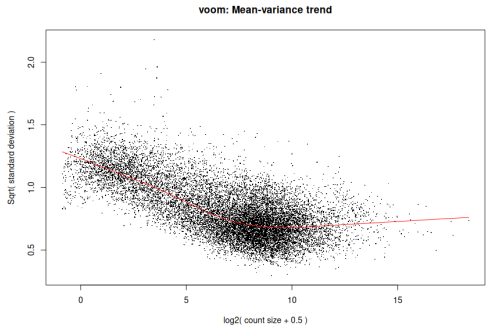

Figure 23: Voom Mean Variance Plot of the FAP cell type in the CCR2KO Dataset

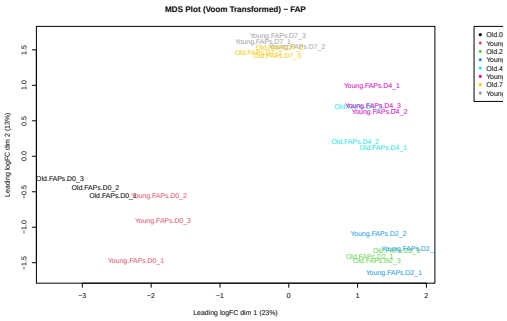

Figure 24: MDS Plot of the FAP cell type in the Ageing Muscle Dataset

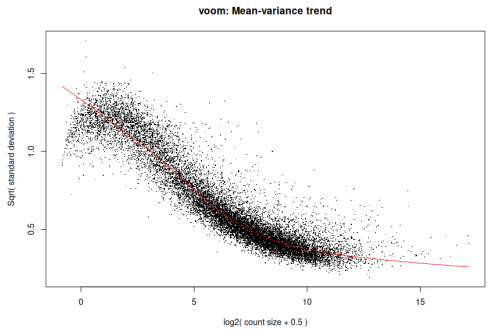

Figure 25: Voom Mean Variance Plot of the FAP cell type in the Ageing Muscle Dataset

Figure 23 adheres to these qualities, while Figure 25 deviates from the qualities slightly. However, Figure 19 still falls within the acceptable distribution for Voom Mean-variance plots [41]. It is also important to note that gene with insignificant expression counts have already been filtered out.

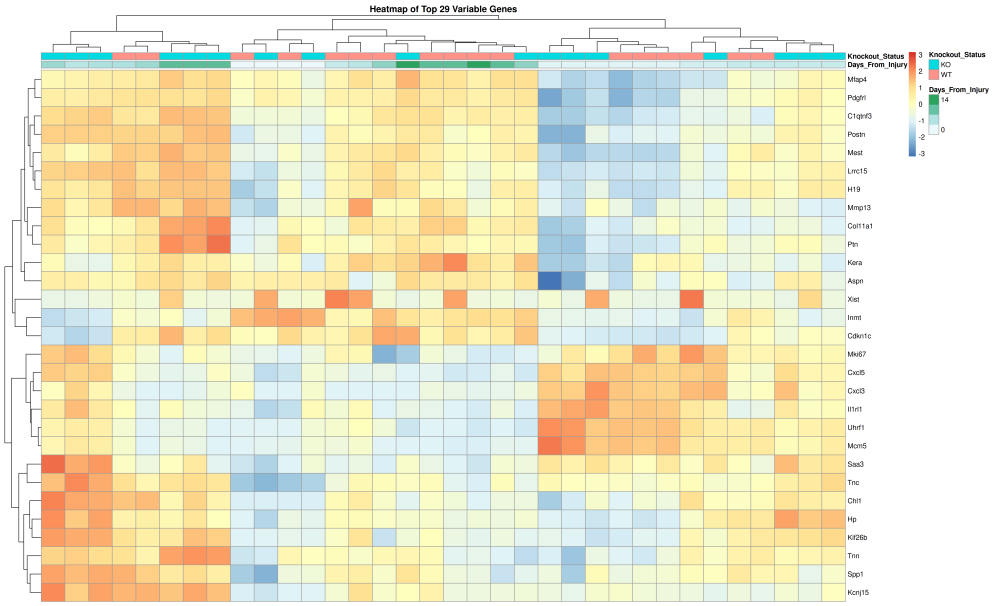

Figure 26: Heatmap of the FAP cell type in the CCR2KO Dataset

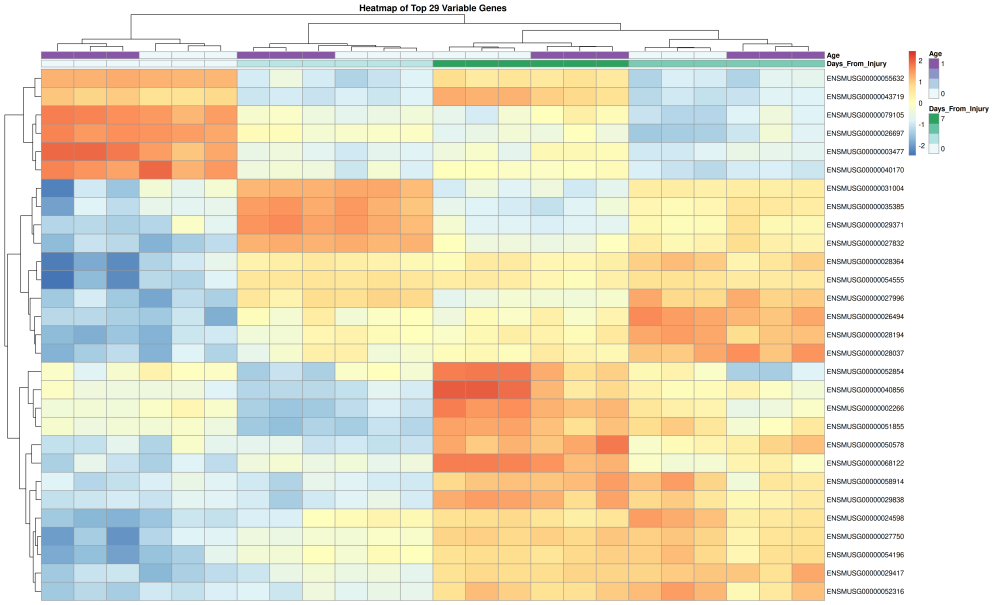

Figure 27: Heatmap of the FAP cell type in the Ageing Muscle Dataset

The Heatmaps do show some patterns regarding gene expression and predictor values., As indicated in Figure 26, the predictor variables do have some effect on gene expression. However, the upregulation and downregulation is much more pronounced in the Ageing Muscle Dataset, as indicated in Figure 27.

Both datasets appear to be suitable for DEG and GO analysis. The next sections

will go further into the significant genes, their biological functions, and the implications of the findings.

4.1.3 Differential Expression Analysis (DEG)

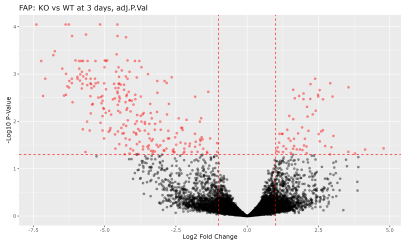

Figure 28: Volcano Plot of FAP cell type at 3 Days from Injury for the CCR2KO Dataset

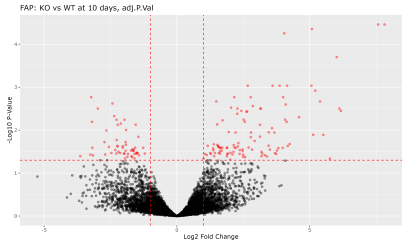

Figure 29: Volcano Plot of FAP cell type at 10 Days from Injury for the CCR2KO Dataset

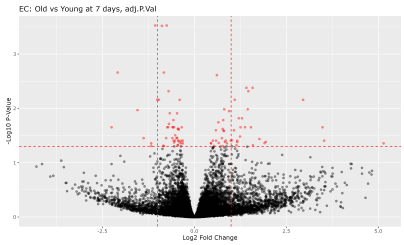

Figure 30: Volcano Plot of EC cell type at 7 Days from Injury for the Ageing Muscle Dataset

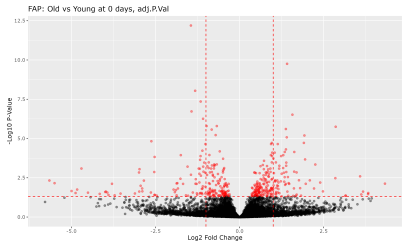

Figure 31: Volcano Plot of FAP cell type at 0 Days from Injury for the Ageing Muscle Dataset

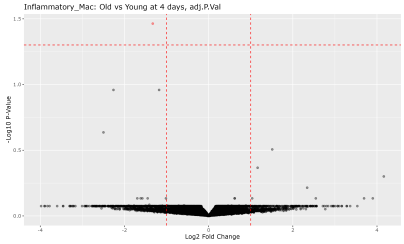

Figure 32: Volcano Plot of M1 cell type at 4 Days from Injury for the Ageing Muscle Dataset

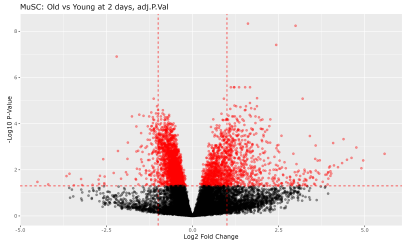

Figure 33: Volcano Plot of MuSC cell type at 2 Days from Injury for the Ageing Muscle Dataset

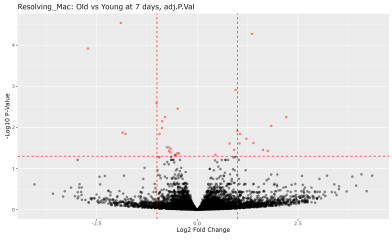

Figure 34: Volcano Plot of M2 cell type at 7 Days from Injury for the Ageing Muscle Dataset

The significant genes for each statistical method vary wildly for both datasets and warrant further inspection. For the purposes of discussion, significant genes are genes that come up as statistically significant without filtering them through the atlas, and are more often referred to when talking about genes of interest in a broader sense;atlas genes are significant genes that also appear in the Human Skeletal Muscle Aging Atlas and more often brought up when discussing actual numerical figures.

For the Ageing Muscle dataset, all significant genes across all statistical methods were found in the Human Skeletal Muscle Ageing Atlas. However, out of 17789 genes in the dataset appears in said atlas, so it is much easier to find atlas genes. Conversely, only genes out of genes in the CCR2KO dataset were found in the Human Skeletal Muscle Ageing Atlas. This translates to 59.8419% of CCR2KO genes being in the atlas. The significant genes found by the three methods and their subsets range from 50.0000% to 95.9916% rate of appearing in the atlas. This means that the Ageing Muscle dataset is indeed more useful in finding significant genes related to ageing muscle regeneration than the CCR2KO dataset, but this is to be expected as the CCR2KO does not measure the age of the mouse specimens.

The DESeq2 method was relatively sparse compared to other methods, but are still quite typical for RNA-Seq datasets [42]. According to a paper entitled "Evaluation of methods for differential expression analysis on multi-group RNA-seq count data" by Tang et al, DESeq2 consistently underperforms other bioinformatics packages such as edgeR, which the Bayesian GLM method utilises. According to Table 2, the number of atlas genes ranges from 7 to 40 for each cell type for the CCR2KO dataset. We can refer to Table 3 for the DESeq2 results for the Ageing Muscle dataset. In this table, the number of atlas genes ranges from 0 to 2 for all cell types except Neutrophil. The Neutrophil subset has a much higher number of atlas genes at 315. However, the lack of a temporal component to the Neutrophil subset means that it is not possible to discern whether the atlas genes vary generally with age, or it they are specific to ageing muscle regeneration.

Concerning the Limma-Voom method, significant genes were found, but not for all combinations of days from injury and cell type. Figures 28-34 show a selection of volcano plots, which all use the Limma-Voom method at specific days from injury and cell types. The figures are for some of Limma-Voom combinations of cell type, days from injury, and dataset that yielded atlas genes. According to Table 2, the number of atlas genes range from 140 to 278 for specific cell types and days from injury for the CCR2KO dataset.

Limma-Voom seems to work much better for the Ageing Muscle Dataset, as all but one combination of cell type and days from injury yielded significant genes. It must be noted that Neutrophil not having a temporal dimension precludes it from being used in the Limma Voom method. Generally speaking, the more distinct time points there are, the easier it is to capture the temporal nature of the data. Given that there are only 2-4 time points for the Ageing Muscle dataset, the Limma Voom results need to be taken with some caution. Still, the days from injury that are captured do reflect distinct phases within the muscle regeneration process, mainly inflammation in the early stage to promote healing and anti-inflammation in the later stage to return the injury site to

homeostasis [1].

When looking at Limma-Voom across both datasets, it is no surprise that FAP and MuSC have the most significant genes. As discussed before, aging can hurt the functioning of FAP's, which indirectly affects the functioning of MuSC's [39]. Cell types that are not FAP and MuSC do not even crack 100 atlas genes for individual time points, while FAP and MuSC time-points often exceed 100 atlas genes.

The Bayesian GLM model has the most consistent performance across cell types for both datasets. Hundreds of atlas genes are found for each cell type, except for Neutrophil, which does not have a temporal component and is thus ineligible for Bayesian GLM analysis. The number of significant genes is more uniform across cell types than with DESeq2 and Limma-Voom. This might be because the priors of the Bayesian GLM model are not customised and the model itself uses stochastic processes. That said, FAP has the highest number of atlas genes for the CCR2KO dataset, and FAP and MuSC have the highest numbers of atlas genes for the Ageing Muscle dataset.

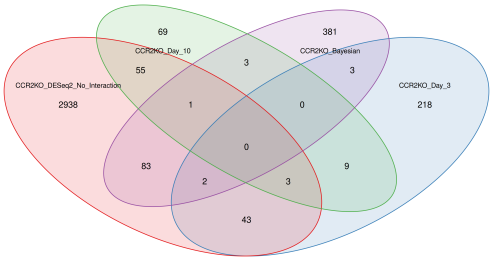

Figure 35: Venn Diagram of FAP Results in the CCR2KO Dataset

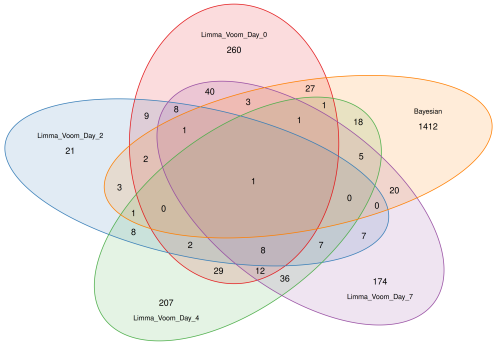

Figure 36: Venn Diagram of FAP Results in the Ageing Muscle Dataset

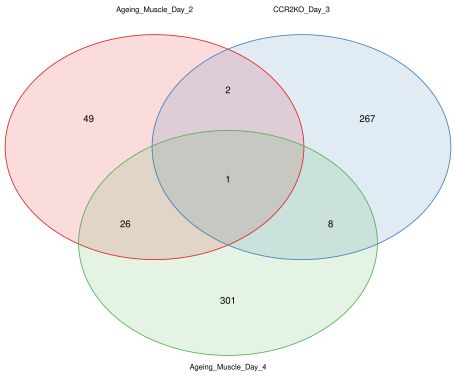

Figure 37: Venn Diagram of FAP Results on 2,3,4 Days from Injury using Limma-Voom across both datasets

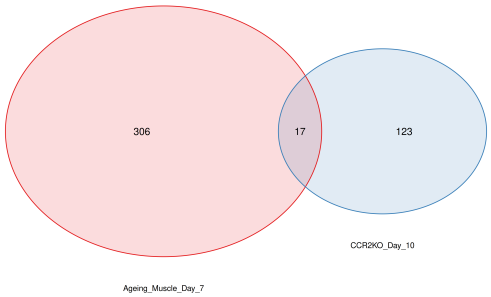

Figure 38: Venn Diagram of FAP Results on 7,10 Days from Injury using Limma-Voom across both datasets

When comparing the different methods it is clear that they have limited overlap in terms of the actual genes found. For the CCR2KO dataset, only FAP has all three methods return results. According to Figure 35, even though Bayesian GLM returned 473 atlas genes, only 92 of them actually appear in the DESeq2 or Limma-Voom atlas genes. For the Ageing Muscle dataset, there are atlas genes, but only 83 of these genes appear in the Limma-Voom atlas genes according to Figure 36. Note that DESeq2 did not return any significant genes.

Looking at the common genes between the CCR2KO and Ageing Muscle datasets show that the two datasets do not have much overlap. Given that the FAP cell type was the only one where Limma-Voom and Bayesian returned results for both datasets, it is best to focus on just this cell type for now. According to Figure 37, there are 387 unique atlas genes between 2 days from injury and 4 days from injury for the FAP cell type when using Limma-Voom on the Ageing Muscle Dataset. There are 278 unique atlas genes for 3 days from injury for the FAP cell type when using Limma-Voom on the CCR2KO dataset. These findings were grouped together due to having the same method of analysis and cell type, as well as having similar days from injury. Overall, there are 11 atlas genes that belong to the Ageing Muscle dataset and at leats one of the two sets of the CCR2KO dataset.

According to Figure 38, there are exactly 17 atlas genes that appear in both the CCR2KO and Ageing Muscle datasets when looking at FAP Limma-Voom results for later days of injury. Further research is needed to determine the significance of these genes, but it is clear that they are differentially expressed at the later stages of muscle regeneration.

Overall, the findings so far do suggest that using the three different statistical methods can provide a more holistic view of the significant genes in the datasets.

4.1.4 Gene Ontology (GO) Analysis

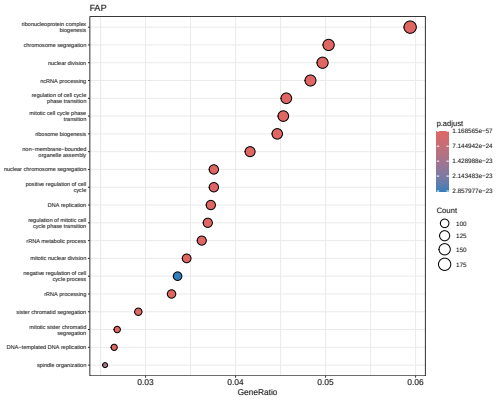

Figure 39: GO Analysis of FAP cell type for DESeq2 No Interaction genes in the CCR2KO Dataset

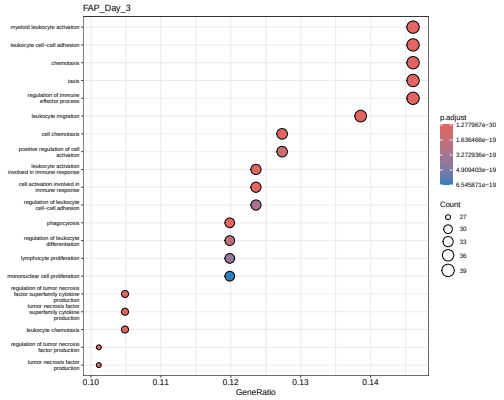

Figure 40: GO Analysis of FAP cell type for 3 Days from Injury for Limma-Voom genes in the CCR2KO Dataset

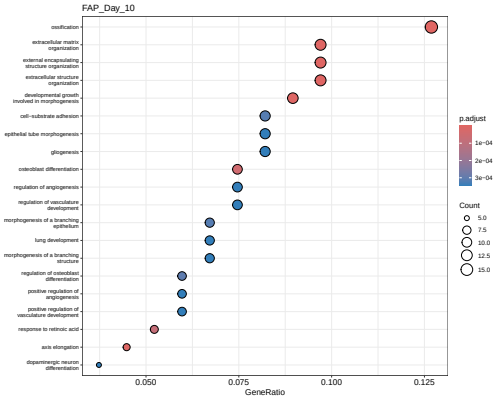

Figure 41: GO Analysis of FAP cell type for 10 Days from Injury for Limma-Voom genes in the CCR2KO Dataset

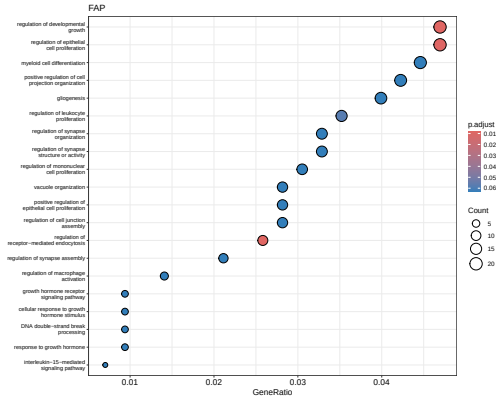

Figure 42: GO Analysis of FAP cell type for Bayesian GLM genes in the CCR2KO Dataset

The GO Analysis has yielded a diverse set of results concerning the biological functions of the significant genes. That said, there are quite a few functions that are associated with the muscular regeneration system. It is important to clarify that the GO plots only include functions with significant enough p-values and p-adj values, and that the full set of identified biological functions can number in the hundreds. EC, FAP, and MP/MuSC are present in both datasets. Much of the discussion will focus on the FAP cell type, as it has the most complete set of results for all three statistical methods across the two datasets. Due to the abundance of GO Enrichment labels, there will not be too much discussion on the specific labels, as Section 4.2 performs more in depth analysis on the labels.

In Figure 39, the most significant GO Analysis label for CCR2KO, cell type FAP, method DESeq2 no interaction is ribonucleoprotein complex biogenesis, which is related to muscular regeneration for its role in RNA metabolism and mRNA stability, as well as being linked to muscle-related diseases [43]. In Figure 40, one of the most significant biological function is the regulation of the immune effector process, which is the stage of the immune system that neutralises or eliminates a threat, such as bacteria, viruses, or cancer cells [44].

According to Figure 41, at 10 days from injury, much of the functions have to do with overall cell and structure development. Some of the most significant functions as defined by p-adj include ossification (bone formation), extracellular matrix organisation, and extracellular structure organisation. Per Figure 42, the Bayesian GLM method had three functions of note: regulation of developmental growth, regulation of epithelial cell proliferation, and myeloid cell differentiation. Epithelial cells serve are found in blood vessels and play a role in inflammation during muscle regeneration [45]. Myeloid cell differentiation is relevant to muscular regeneration, as macrophages are a type of myeloid cell [46]. Overall, the CCR2KO dataset does show a variety of biological functions that are associated with muscle regeneration.

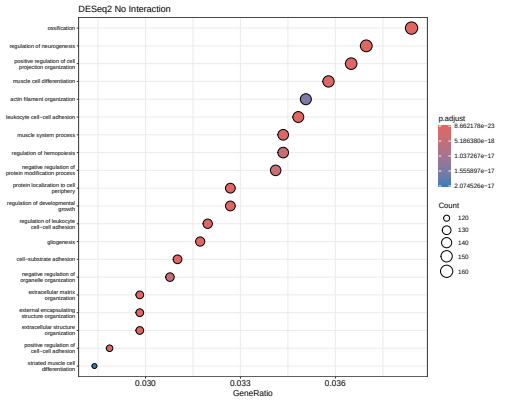

Figure 43: GO Analysis of FAP cell type at 2 Days of Injury for DESeq2 No Interaction genes in the Ageing Muscle Dataset

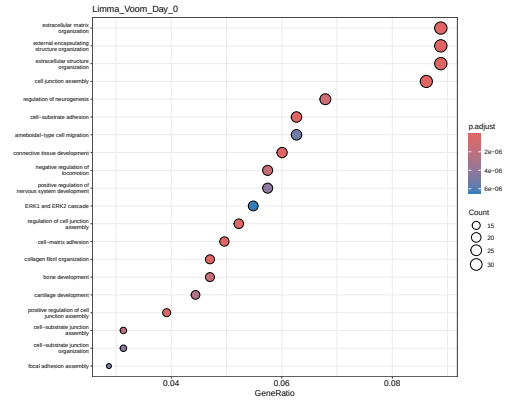

Figure 44: GO Analysis of FAP cell type at 0 Days of Injury for Limma-Voom genes in the Ageing Muscle Dataset

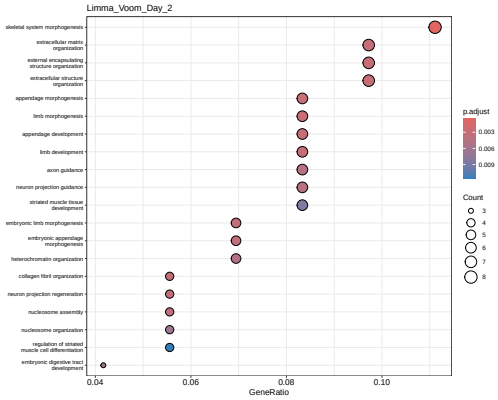

Figure 45: GO Analysis of FAP cell type at 2 Days of Injury for Limma-Voom genes in the Ageing Muscle Dataset

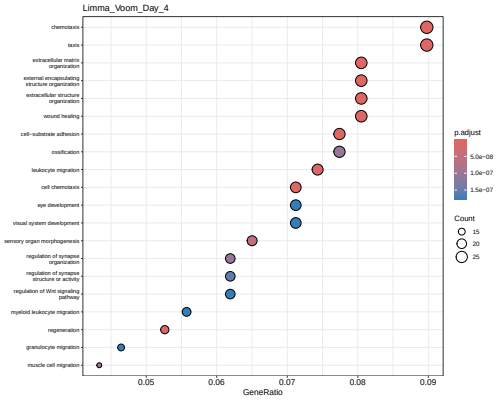

Figure 46: GO Analysis of FAP cell type at 4 Days of Injury for Limma-Voom genes in the Ageing Muscle Dataset

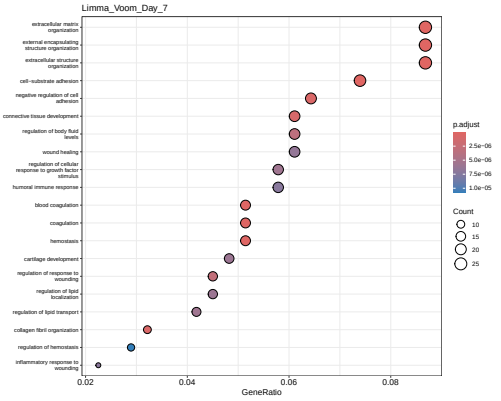

Figure 47: GO Analysis of FAP cell type at 7 Days of Injury for Limma-Voom genes in the Ageing Muscle Dataset

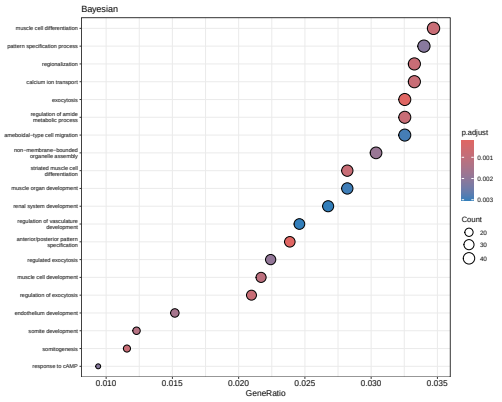

Figure 48: GO Analysis of FAP cell type for Bayesian GLM genes in the Ageing Muscle Dataset

The Ageing Muscle dataset also yields some noteworthy biological functions, with some of them even overlapping with the significant functions from the CCR2KO dataset. According to Figure 43, the most significant GO Analysis label is also ossification, just like in 10 days from injury in the CCR2KO dataset. This is a notable finding, showing that ossification is affected by both ageing and CCR2 knockout. Ossification itself is not directly related to muscle regeneration, but it can affect if if the condition of Heterotopic ossification arises. Although this condition is rare, it can develop after traumatic injuries and impairs muscle functions by forming bone in muscle tissue [47].

According to Figure 44, at 0 days from injury, the Limma-Voom method returns some significant functions, namely extracellular matrix organisation, external encapsulating structure organisation, extracellular structure organisation, cell junction assembly, and

ERK1 and ERK2 cascade. Extracellular matrix organisation, external encapsulating structure organisation, and extracellular structure organisation are essential for muscular regeneration, as they provide the overall structure for the muscle cells to grow and repair the site of injury [48]. These three functions also appear in the CCR2KO dataset, specifically the 10 days from injury subset. This suggests that these functions are impacted by ageing. ERK1 and ERK2 cascade is a signalling pathway essential for the proliferation and differentiation of satellite cells [49], which are the muscle stem cells that repair the site of injury.

According to Figures 45, 46, and 47, the rest of the Limma-Voom results for the FAP cell type in the Ageing Muscle dataset have similar biological functions. The functions extracellular matrix organisation, external encapsulating structure organisation, and extracellular structure organisation all appear at 2,4, and 7 days from injury. At 2 days from injury, striated muscle tissue development is a significant biological function, as it regulates both skeletal and cardiac muscle development [50]. The significant biological from 7 days from injury include wound healing, blood coagulation, cartilage development, and hemostasis. Hemostasis is the process of stopping bleeding, which comes at the earliest state of muscle regeneration before the inflammation phase [51].

Per Figure 48, the Bayesian GLM returns a slightly different set of biological functions, but they are still related to muscle regeneration. In fact, a high proportion of the most significant biological functions from the Bayesian GLM method are directly related to muscular regeneration. According to Figure 48, the top biological functions include, but are not limited to, muscle cell differentiation, pattern specification process, regionalisation, calcium ion transport, exocytosis, striated muscle cell differentiation, muscle organ development, muscle cell development, and endothelium development. The calcium ion transport function is of particular note, as a recent paper entitled "Calcium's Role and Signaling in Ageing Muscle, Cellular Senescence" suggests that alterations in the calcium ion transport system could be a key factor in aging muscle, and can even accelerate conditions such as sarcopenia [52]. Furthermore, calcium plays an important role in muscle contraction, so its role extends beyond muscle regeneration and into general muscle functioning. Exocytosis is the process of releasing materials from within cells, and is essential for releasing growth factors and other molecules to support muscle regeneration [53]. Endothelial development is essential for muscular regeneration, as they help form new blood vessels within muscle tissue [54].

The EC, MP, and MuSC cell types also return GO Analysis labels that are related to muscular regeneration. These can be seen in the Appendix. Some notable results include ribonucleoprotein complex biogenesis as the top GO Analysis label for the MuSC cell type at 2 days from injury in the Ageing Muscle dataset, and generation of precursor metabolites and energy as the top GO Analysis label for the EC cell type using the Bayesian GLM method in the Ageing Muscle dataset. Generation of precursor metabolites and energy is crucial for the production of ATP, which constitutes the energy supply for muscle cells [55]

Despite the atlas genes being quite diverse, their functions seem to be quite consistent across the different statistical methods. This suggests that the these methods are capable

of finding significant genes with biological significance.

4.2 Evaluation

Set 1	**Set 2**	**Cell Type**	**Overall Similarity**	**Percentage of Common GO Terms**
Day2 AM	Day3 CCR2KO	FAP	0.3958	15.8023%
Day4 AM	Day3 CCR2KO	FAP	0.5930	68.9039%
Day7 AM	Day10 CCR2KO	FAP	0.6215	88.6059%
DESeq2 No Interaction AM	DESeq2 No Interaction CCR2KO	FAP	0.7241	88.8695%
Bayesian AM	Bayesian CCR2KO	FAP	0.5619	91.0345%
DESeq2 No Interaction CCR2KO	Bayesian CCR2KO	FAP	0.5579	98.5690%
DESeq2 No Interaction AM	Bayesian AM	FAP	0.6590	97.0966%
DESeq2 No Interaction AM	DESeq2 No Interaction CCR2KO	MP	0.6454	96.6974%
Bayesian AM	Bayesian CCR2KO	MP, MuSC	0.5753	91.4085%
DESeq2 No Interaction AM	Bayesian AM	MuSC	0.6582	97.6699%
DESeq2 No Interaction CCR2KO	Bayesian CCR2KO	MP	0.5669	89.7436%
DESeq2 No Interaction CCR2KO	Bayesian CCR2KO	EC	0.5670	93.0370%
DESeq2 No Interaction AM	Bayesian AM	EC	0.5829	99.3884%
DESeq2 No Interaction CCR2KO	DESeq2 No Interaction AM	EC	0.5046	15.6491%
Bayesian CCR2KO	Bayesian AM	EC	0.3929	29.3269%

Table 5: Comparison of GO term similarities between different time points and datasets

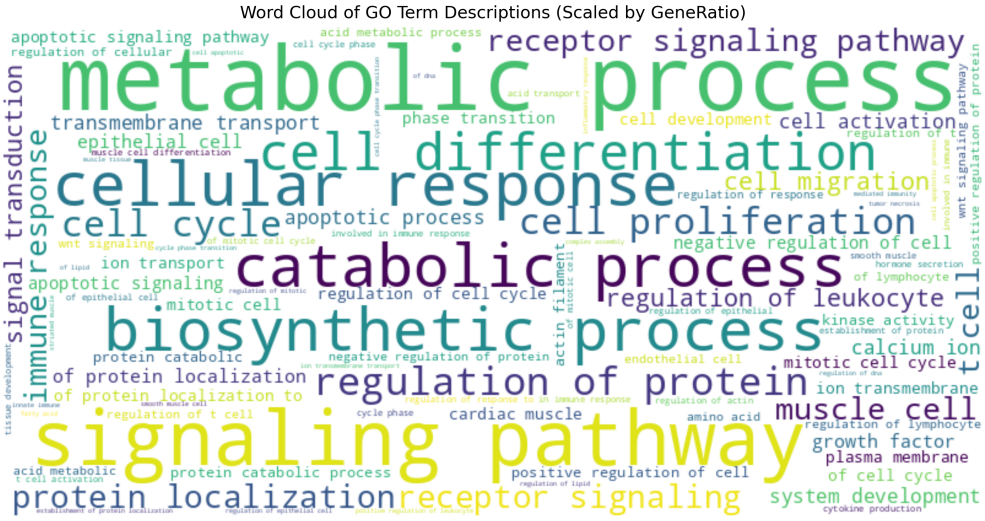

Figure 49: Word Cloud of Most Common Terms in the FAP GO Analysis Biological Function Labels

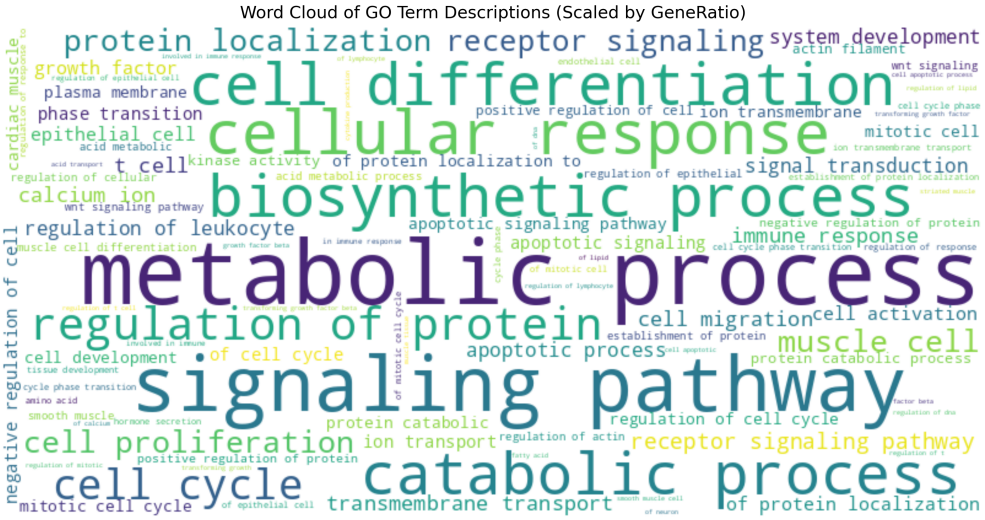

Figure 50: Word Cloud of Most Common Terms in the MuSC GO Analysis Biological Function Labels

Per Table 5, there is a significant amount of overlap between the two datasets looking at the FAP cell type and the Limma-Voom method. Note that the Bayesian GLM method did not yield enough distinct significant GO Analysis labels in the CCR2KO dataset to be included in the comparison. The results are paired up by time point, with the 3 days from Injury in the CCR2KO dataset paired up with the 2 and 4 days from injury in the Ageing Muscle dataset; 10 days from injury in the CCR2KO dataset is paired up with 7 days from injury in the Ageing Muscle dataset. The percentage of

common GO terms vary dramatically from 8.7214% to 82.1815%. But as stated before, emphasising exact matches can lead to overfitting, which is why overall set similarity presents a more holistic approach. The overall set similarity ranges from 0.3714 to 0.6038, which is more balanced. The differential gene expression appears to be much more consistent at later time points, with the 7 day and 10 day pairing yielding the highest similarity score.

Figure 49 presents a word cloud of the most common terms in the GO Analysis labels. The word cloud is scaled by gene ratio, so the more often a label appears, the more likely it is to be larger. Stop words and terms with little information value have been filtered out, so the word cloud is more focused on the most significant terms and phrases. Needless to say, many of the terms appear to be relevant to muscular regeneration, such as the terms "metabolic process", "catabolic process", "cell differentiation", "signalling pathway", "regulation of protein", and "immune response". Figure 50 had similar terms, such as "metabolic process", "signalling pathway", "cell differentiation", "muscle cell", and "receptor signalling".

The MP and MuSC results show high degrees of overlap, with the Bayesian results from both datasets having a similarity score of 0.5753 and a direct match of GO terms of 91.4085%. The EC results were more sparse and thus had lower similarity scores and common GO terms.

The DESeq2 formula produced diverging results, with Version 1 being very sparse, while Version 2 being greatly abundant. However, this is to be expected, as the interaction term raises the threshold of significance. As such, only Version 2 was used for evaluation, as Version 1 would not have enough significant genes to make comparison worthwhile. It is worth noting that the CCR2 Knockout paper by Groppa et al used a formula similar to Version 2 and also found a large number of significant genes [3].

The only real downside is that the Limma-Voom method is quite sparse in its results for the CCR2KO dataset, but still performed well for the Ageing Muscle dataset. The CCR2KO dataset provided the raw counts, which eliminates the possibility of smoothing. What is more likely the reason is the large number of genes that were filtered out during the Limma-Voom analysis. There were genes in the CCR2KO, but only to genes actually survived the initial filtering process. This is roughly a 59.4284% to 62.7493% retention rate. Conversely, the retention rate for the Ageing Muscle dataset when using Limma-Voom is 78.3503% to 91.4424%. This is a significant difference, as this can lead to lower statistical power and thus less significant genes being discovered [56]. As the same thresholds were used in both datasets for gene filtering, this could be a reason for the discrepancy in the results.

Regarding what the final set of genes are for the Ageing Muscle dataset, it is best to stick to the results that perform the best across the CCR2KO dataset and the Ageing Muscle dataset. DESeq2 can be eliminated, as Version 1 is too sparse, and Version 2 is too abundant in terms of number of significant genes. The Limma-Voom results can be used, but only for Day 4 and Day 7 of the Ageing Muscle dataset. The CCR2KO dataset didn't return many results for Limma Voom, and only Day 3 and Day 10 had significant genes for the purpose of cross comparison. The Bayesian GLM results can be

used for FAP and MP/MuSC, as the similarity score and the percentage of common GO terms are quite high for both cell types. The EC Bayesian results were not so similar for EC, so that would have to be eliminated. The final set of genes are shown below.

Cell Type	**Day**	**Method**	**Dataset**	**Number of Atlas Genes**
FAP	4	Limma-Voom	Ageing Muscle	336
FAP	7	Limma-Voom	Ageing Muscle	323
FAP	N/A	Bayesian GLM	Ageing Muscle	1495
MuSC	N/A	Bayesian GLM	Ageing Muscle	1667
Total Number of Unique Atlas Genes				3252

Table 6: Final Set of Significant Genes for the Ageing Muscle Dataset

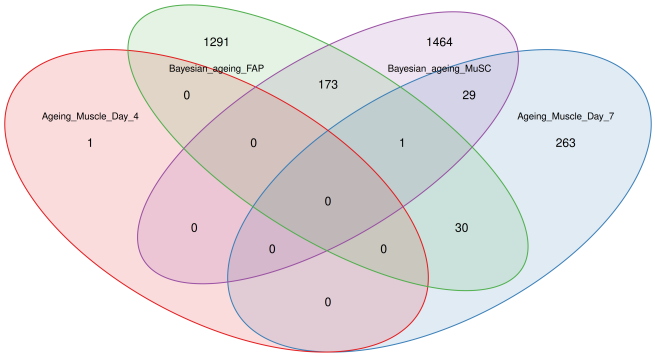

Figure 51: Venn Diagram of the Final Set of Significant Genes for the Ageing Muscle Dataset

The evaluation method shows that using the CCR2KO dataset as the training dataset and the Ageing Muscle dataset as the testing dataset is effective at finding significant genes related to ageing muscle. The three statistical methods used were able to return significant genes with biological significance, and the GO Analysis labels were reasonably consistent across the different methods. Note that the final set of atlas genes in Table 6 is 3252, which takes into account that some genes may appear in multiple methods, and thus the final count is the number of unique genes across all methods. Figure 51 shows the Venn Diagram of the final set of significant genes for the Ageing Muscle dataset.

5 Legal, Social, Ethical and Professional Issues

This project adheres to the principles and guidelines as described in the British Computing Society's Code of Conduct [57]. The project only uses publicly available RNA-seq datasets, in which the test subjects are mice. As the test subjects are mice, the RNA-seq data on them is not subject to any privacy restrictions [58]. Both the CCR2KO and Ageing Muscle datasets are available to download and analyse per their respective websites. I did not play a role in the generation of the datasets, so there is no conflict of interest regarding the use of the data.

The Human Cell Atlas, which has the Ageing Muscle dataset, states the following in their Data Release Policy: "Data users are free to use and analyze data in the DCP that they did not generate, but are asked to adhere to the Fort Lauderdale principles: they should not publish major analyses before the data generators do so; if there is any doubt, they should contact data generators to let them know how they are using the data and to agree on an appropriate scope for analysis; and they should cite the dataset and acknowledge the generators in any publications and presentations that make use of the data" [59]. Given that this dissertation is not in any stages of publication in research journals, there is not a strict requirement to contact the parties involved in generating the dataset. However, the dataset is cited in the references, and the generators of the data are acknowledged in the Acknowledgements section.

There is limited use of third party code, with the biggest contribution for the generation of GO Plots [60]. The GO Plots code was inspired by the source code from the research paper entitled "Spatial compartmentalization of signaling imparts source-specific functions on secreted factors" by E. Groppa, which also analysed the CCR2KO dataset [3]. Per the British Computing Society's Code of Conduct, oen should "...have due regard for the legitimate rights of third parties" [57]. The code was used for research and academic purposes and the source code was cited in the references.

6 Conclusion

The research shows that the proposed methods of analysis are effective in identifying significant genes in the GRN that are related to ageing muscle. Given that the process of mapping out the GRN isn't close to completion, using methods to identify both the statistical and biological significance of genes of interest are essential. Furthermore, using the CCR2KO to infer the differentially expressed genes in ageing muscle has shown to be a good complementary method to directly analysing the Ageing Muscle dataset. Not only does this provide a point of contrast, but also helps ensure that the statistical methods are robust.

It's important to note that the Human Skeletal Muscle Aging Atlas, which is used to identify atlas genes, isn't complete and definite. Genes that aren't found in this atlas can still have biological significance in relationship to ageing muscle. This is because gene atlases are often narrow in scope, so genes with low expression or are less studied may be omitted entirely from he atlas [61]. Further research must be done to investigate candidate interacting genes that are not found in the atlas. In fact, one notable database cited in research is the Single-cell Skeletal Muscle Regeneration Database (SCSMRD), which is directly analysed in the paper "SCSMRD: A database for single-cell skeletal muscle regeneration" [62]. The biggest change that could be made in this research is the incorporation of additional databases to identify genes of interest.

Apart from the limitations of the atlas, RNA-seq data must also be more robust to ensure they aren't underpowered. The statistical methods chosen in this research are ones that can still provide statistically significant results for underpowered data, but even then the results would be less robust than if the data had a larger sample size. The small sample sizes in many RNA-seq dataset also make it more difficult to implement state-of-the-art models, such as tree models and neural networks, which require large amounts of data to train effectively and can overfit on small dataset.

Still, the research has identified numerous significant genes across both dataset using a variety of methods. The results of the research can be used to further investigate specific genes and their related biological functions in ageing muscle.

References

[1] M. Kloc, A. Uosef, H. V. Ubelaker, J. Z. Kubiak, and R. M. Ghobrial, "Macrophages and stem/progenitor cells interplay in adipose tissue and skeletal muscle: A review," *Stem Cell Investigation*, vol. 10, p. 8, 2023.

[2] I. Koike, H.and Manabe and Y. Oishi, "Mechanisms of cooperative cell-cell interactions in skeletal muscle regeneration," *Inflammation and Regeneration*, vol. 42, p. 48, 11 2022.

[3] E. G. et al., "Spatial compartmentalization of signaling imparts source-specific functions on secreted factors," *Cell Reports*, vol. 42, no. 2, 2023.

[4] P. Sousa-Victor, L. García-Prat, and P. Muñoz-Cánoves, "Control of satellite cell function in muscle regeneration and its disruption in ageing," *Nature Reviews Molecular Cell Biology*, vol. 23, no. 23, 2021.

[5] A. R. Palla, M. Ravichandran, and Y. e. a. Wang, "Inhibition of prostaglandin-degrading enzyme 15-pgdh rejuvenates aged muscle mass and strength," *Science*, vol. 371, no. 6528, p. eabc8059, 2021.

[6] F. H. C. Crick, "On protein synthesis," *Symposia of the Society for Experimental Biology*, vol. 12, pp. 138–163, 1958.

[7] X. Cao, Y. Zhang, and Y. e. a. Ding, "Identification of rna structures and their roles in rna functions," *Nature Reviews Molecular Cell Biology*, 2024.

[8] D. Kim, A. Tran, and H. e. a. Kim, "Gene regulatory network reconstruction: harnessing the power of single-cell multi-omic data," *Nature*, vol. 9, no. 51, 2023.

[9] F. Rossi, "Spatial compartmentalization of signalling imparts source-specific functions on secreted factors.," 2022.

[10] P. K. Agarwal, "Enzymes: An integrated view of structure, dynamics and function," *Microbial Cell Factories*, vol. 5, p. 2, 2006.

[11] C. C. Serdar, M. Cihan, D. Yücel, and M. A. Serdar, "Sample size, power and effect size revisited: simplified and practical approaches in pre-clinical, clinical and laboratory studies," *Biochemia Medica*, vol. 31, no. 1, p. 010502, 2021.

[12] Y. Yang, A. Sánchez-Tójar, and R. e. a. O'Dea, "Publication bias impacts on effect size, statistical power, and magnitude (Type M) and sign (Type S) errors in ecology and evolutionary biology," *BMC Biology*, vol. 21, no. 71, 2023.

[13] R. van de Shoot, S. Depaoli, and R. e. a. King, "Bayesian statistics and modelling," *Nat Rev Methods Primers*, vol. 1, no. 1, 2021.

[14] S. Sealfon, "Single-cell transcriptional profiles in human and mouse skeletal muscle," 2022.

[15] H. Xi, J. Langerman, and S. et al, "A human skeletal muscle atlas identifies the trajectories of stem and progenitor cells across development and from human pluripotent stem cells," *Cell Stem Cell*, vol. 27, pp. 158–176.e10, July 2020. Epub 2020 May 11. Erratum in: Cell Stem Cell. 2020 Jul 2;27(1):181-185. doi: 10.1016/j.stem.2020.06.006. PMID: 32396864; PMCID: PMC7367475.

[16] J. Jaccard and C. K. Wan, "Measurement error in the analysis of interaction effects between continuous predictors using multiple regression: Multiple indicator and structural equation approaches," *Psychological Bulletin*, vol. 117, no. 2, pp. 348–357, 1995.

[17] R. Couronné, P. Probst, and A. L. Boulesteix, "Random forest versus logistic regression: a large-scale benchmark experiment," *BMC Bioinformatics*, vol. 19, p. 270, 2018.

[18] L. Brigato and L. Iocchi, "A close look at deep learning with small data," in *2020 25th International Conference on Pattern Recognition (ICPR)*, (Milan, Italy), pp. 2490–2497, 2021.

[19] G. Jiang, J. Zheng, and S. e. a. Ren, "A comprehensive workflow for optimizing rna-seq data analysis," *BMC Genomics*, vol. 25, p. 631, 2024.

[20] S. Wu, K. Jin, and M. e. a. Tang, "Inference of gene regulatory networks based on multi-view hierarchical hypergraphs," *Interdisciplinary Sciences: Computational Life Sciences*, 2024.

[21] A. Passemiers, Y. Moreau, and D. Raimondi, "Fast and accurate inference of gene regulatory networks through robust precision matrix estimation," *Bioinformatics*, vol. 38, pp. 2802–2809, May 2022.

[22] X. Cui and G. A. Churchill, "Statistical tests for differential expression in cdna microarray experiments," *Genome Biology*, vol. 4, p. 210, 2003.

[23] W. Li, "Volcano plots in analyzing differential expressions with mrna microarrays," *arXiv*, 2011.

[24] L. M. Urpa and S. Anders, "Focused multidimensional scaling: interactive visualization for exploration of high-dimensional data," *BMC Bioinformatics*, vol. 20, p. 221, 2019.

[25] C. W. Law, Y. Chen, W. Shi, and G. . Smyth, "voom: precision weights unlock linear model analysis tools for rna-seq read counts," *Genome Biology*, vol. 15, p. R29, 2014.

[26] E. Xekalaki, "On the distribution theory of over-dispersion," *Journal of Statistical Distributions and Applications*, vol. 1, p. 19, 2014.

[27] M. I. Love, W. Huber, and S. Anders, "Moderated estimation of fold change and dispersion for rna-seq data with deseq2," *Genome Biology*, vol. 15, no. 12, p. 550, 2014.

[28] Z. Yi, Y. Wei, and C. e. a. Cheng, "Improving sample efficiency of high dimensional bayesian optimization with mcmc," *arXiv preprint arXiv:2401.02650*, 2024.

[29] S. D. Team, "Bayesian applied regression modeling via stan," 2024. Accessed: 2024-06-24.

[30] B. Vestal, C. Moore, and E. e. a. Wynn, "Mcmseq: Bayesian hierarchical modeling of clustered and repeated measures rna sequencing experiments," *BMC Bioinformatics*, vol. 21, p. 375, 2020.

[31] M. Ashburner, C. A. Ball, and J. A. e. a. Blake, "Gene ontology: tool for the unification of biology," *Nature Genetics*, vol. 25, pp. 25–29, May 2000. The Gene Ontology Consortium.

[32] J. Cao and S. Zhang, "A bayesian extension of the hypergeometric test for functional enrichment analysis," *Biometrics*, vol. 70, pp. 84–94, March 2014.

[33] M. Bogdan, J. K. Ghosh, and S. T. Tokdar, "A comparison of the benjamini-hochberg procedure with some bayesian rules for multiple testing," in *Beyond Parametrics in Interdisciplinary Research: Festschrift in Honor of Professor Pranab K. Sen*, vol. 1, pp. 211–231, Institute of Mathematical Statistics, 2008.

[34] Y. Benjamini and Y. Hochberg, "Controlling the false discovery rate: A practical and powerful approach to multiple testing," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 57, no. 1, pp. 289–300, 1995.

[35] R. Shrivastava, A. Pujahari, S. P. Singh, and T. Bhowmik, "Efficient question answering in chatbot using tf-idf and cosine similarity," in *Innovations in Information and Communication Technologies. Algorithms for Intelligent Systems* (D. Garg, N. Kumar, R. Iqbal, and S. Gupta, eds.), pp. 45–58, Singapore: Springer, 2023.

[36] V. Verma and R. Aggarwal, "A comparative analysis of similarity measures akin to the jaccard index in collaborative recommendations: empirical and theoretical perspective," *Social Network Analysis and Mining*, vol. 10, no. 1, p. 43, 2020.

[37] J. Kruczek, P. Kruczek, and M. Kuta, "Are n-gram categories helpful in text classification?," in *Computational Science – ICCS 2020. ICCS 2020. Lecture Notes in Computer Science* (V. V. Krzhizhanovskaya *et al.*, eds.), vol. 12138, Cham: Springer, 2020.

[38] R. S. Blanc, J. G. Kallenbach, and J. F. e. a. Bachman, "Inhibition of inflammatory ccr2 signaling promotes aged muscle regeneration and strength recovery after injury," *Nature Communications*, vol. 11, p. 4167, Aug. 2020.

[39] L. Lukjanenko, S. Karaz, and P. e. a. Stuelsatz, "Aging disrupts muscle stem cell function by impairing matricellular wisp1 secretion from fibro-adipogenic progenitors," *Cell Stem Cell*, vol. 24, no. 3, pp. 433–446.e7, 2019.

[40] B. Pawlikowski, T. Betta, and T. e. a. Elston, "A cellular atlas of skeletal muscle regeneration and aging," *bioRxiv*, 2019.

[41] UC Davis Bioinformatics Training Program, "RNA-Seq Workshop: Differential Expression Analysis," 2018. Accessed: 2024-07-26.

[42] M. Tang, J. Sun, and K. e. a. Shimizu, "Evaluation of methods for differential expression analysis on multi-group rna-seq count data," *BMC Bioinformatics*, vol. 16, p. 360, 2015.

[43] Z. Li, H. Wei, and D. e. a. Hu, "Research progress on the structural and functional roles of hnrnps in muscle development," *Biomolecules*, vol. 13, p. 1434, 2023.

[44] Y. Mazor, C. Yang, and M. J. e. a. Borrok, "Enhancement of immune effector functions by modulating IgG's intrinsic affinity for target antigen," *PLOS ONE*, vol. 11, no. 6, p. e0157788, 2016.

[45] E. Alarcin, A. Bal-Öztürk, and H. e. a. Avci, "Current strategies for the regeneration of skeletal muscle tissue," *International Journal of Molecular Sciences*, vol. 22, no. 11, p. 5929, 2021.

[46] S. S. Welc, M. Wehling-Henricks, J. Antoun, *et al.*, "Differential effects of myeloid cell pparδ and il-10 in regulating macrophage recruitment, phenotype, and regeneration following acute muscle injury," *The Journal of Immunology*, vol. 205, no. 6, pp. 1664–1677, 2020.

[47] K. Alexander, H. Tseng, and M. e. a. Salga, "When the nervous system turns skeletal muscles into bones: How to solve the conundrum of neurogenic heterotopic ossification," *Current Osteoporosis Reports*, vol. 18, no. 5, pp. 666–676, 2020.

[48] L. Forcina, M. Cosentino, and A. Musarò, "Mechanisms regulating muscle regeneration: Insights into the interrelated and time-dependent phases of tissue healing," *Cells*, vol. 9, no. 5, p. 1297, 2020.

[49] X. Deschênes-Simard, M. Malleshaiah, and G. Ferbeyre, "Extracellular signal-regulated kinases: One pathway, multiple fates," *Cancers*, vol. 16, p. 95, 2024.

[50] I. Y. Shadrin, A. Khodabukus, and N. Bursac, "Striated muscle function, regeneration, and repair," *Cellular and Molecular Life Sciences*, vol. 73, no. 22, pp. 4175–4202, 2016.

[51] B. Choi, C. Lee, and J. W. Yu, "Distinctive role of inflammation in tissue repair and regeneration," *Archives of Pharmacal Research*, vol. 46, pp. 78–89, 2023.

[52] K. Terrell, S. Choi, and S. Choi, "Calcium's role and signaling in aging muscle, cellular senescence, and mineral interactions," *International Journal of Molecular Sciences*, vol. 24, p. 17034, 2023.

[53] J. Han, K. Pluhackova, and R. A. Böckmann, "The multifaceted role of snare proteins in membrane fusion," *Frontiers in Physiology*, vol. 8, p. 5, 2017.

[54] L. J. Greenspan and B. M. Weinstein, "To be or not to be: endothelial cell plasticity in development, repair, and disease," *Angiogenesis*, vol. 24, pp. 251–269, 2021.

[55] C. J. Barclay, "Energy demand and supply in human skeletal muscle," *Journal of Muscle Research and Cell Motility*, vol. 38, no. 2, pp. 143–155, 2017.

[56] C. Soneson and M. Delorenzi, "A comparison of methods for differential expression analysis of rna-seq data," *BMC Bioinformatics*, vol. 14, p. 91, 2013.

[57] T. C. I. f. I. BCS, "Bcs code of conduct," 2024. Accessed: 2024-06-29.

[58] R. Ogden, K. Bayne, and L. Kline, "Introduction: Global laws, regulations, and standards for animals in research," *ILAR Journal*, vol. 57, no. 3, pp. 261–264, 2016.

[59] H. C. Atlas, "Human cell atlas data release policy," 2024. Accessed: 2024-06-29.

[60] P. Martini, "Spatial compartmentalization of signalling imparts source-specific functions on secreted factors - groppa e. et al, cell reports 2023," 2023. Accessed: 2024-06-29.

[61] A. J. Tarashansky, J. M. Musser, and M. e. a. Khariton, "Mapping single-cell atlases throughout metazoa unravels cell type evolution," *eLife*, vol. 10, p. e66747, 2021.

[62] X. Feng, C. Xie, and Y. e. a. Li, "Scsmrd: A database for single-cell skeletal muscle regeneration," *Journal of Integrative Agriculture*, vol. 22, no. 3, pp. 864–871, 2023.

The content of “Appendix” is in “\contents\app_1.tex”

A Appendix

Supplementary materials (such as source code, user menu, etc) could be included. Each appendix must be labelled (for example, Appendix A, Appendix A.1, Appendix A.2, Appendix B, Appendix B.1, etc.) and with heading. All Appendices must be referred in the text.

A.1 DEG Analysis: Distribution of Time Coefficients

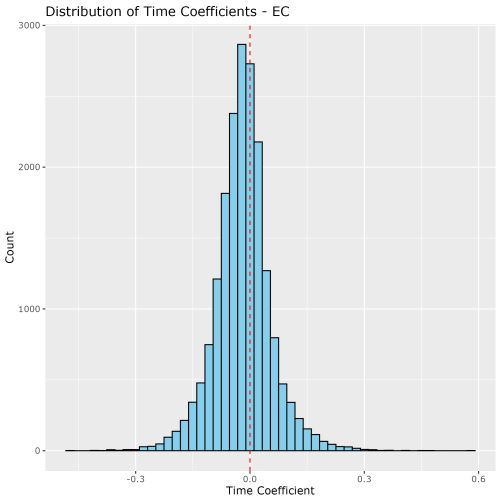

Figure 52: Distribution of Time Coefficients of EC cell type for the CCR2KO Dataset

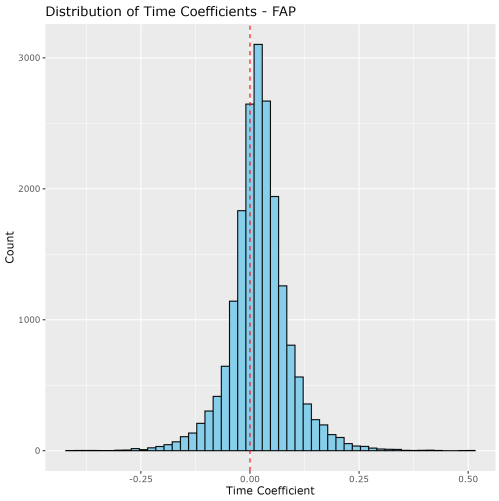

Figure 53: Distribution of Time Coefficients of FAP cell type for the CCR2KO Dataset

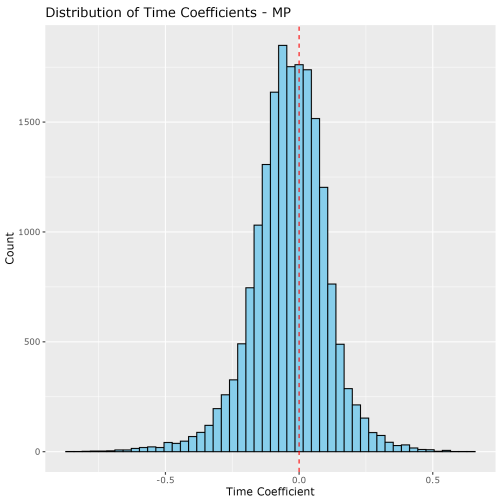

Figure 54: Distribution of Time Coefficients of MP cell type for the CCR2KO Dataset

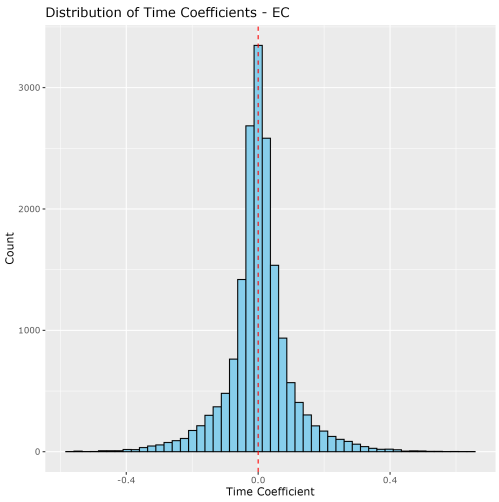

Figure 55: Distribution of Time Coefficients of EC cell type for the Ageing Muscle Dataset

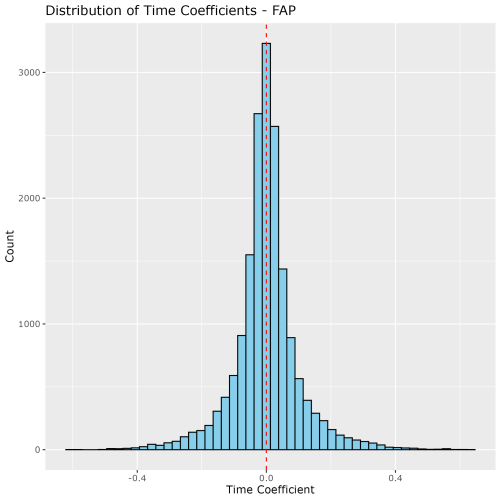

Figure 56: Distribution of Time Coefficients of FAP cell type for the Ageing Muscle Dataset

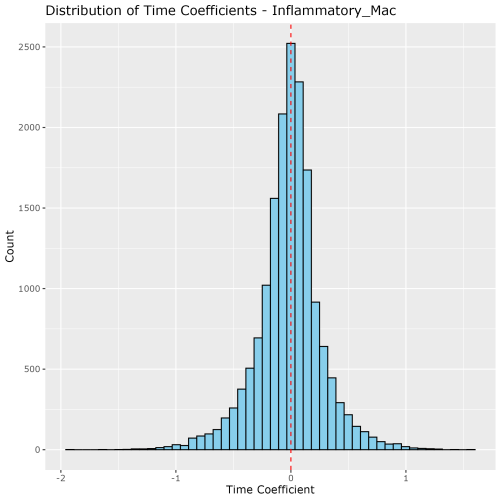

Figure 57: Distribution of Time Coefficients of M1 cell type for the Ageing Muscle Dataset

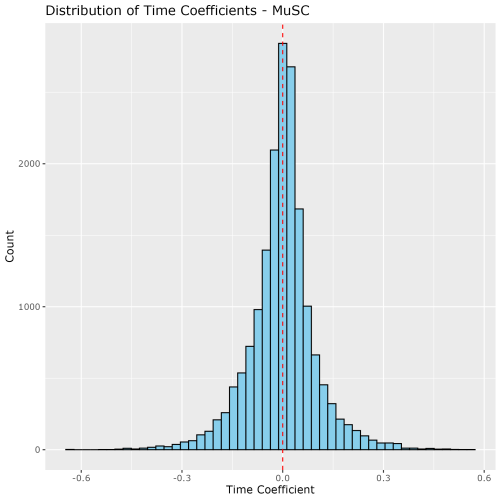

Figure 58: Distribution of Time Coefficients of MuSC cell type for the Ageing Muscle Dataset

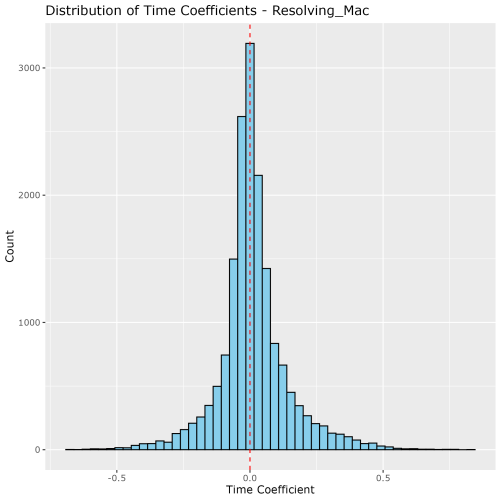

Figure 59: Distribution of Time Coefficients of M2 cell type for the Ageing Muscle Dataset

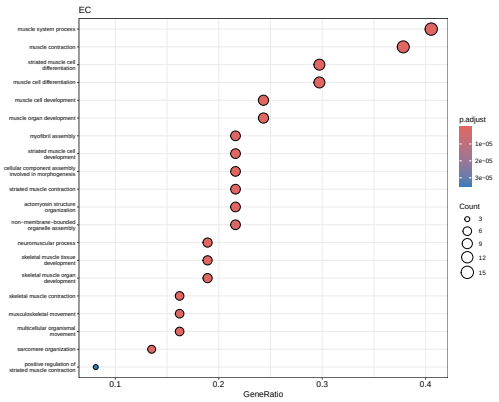

Figure 60: GO Analysis of EC cell type for DESeq2 genes for the CCR2KO Dataset

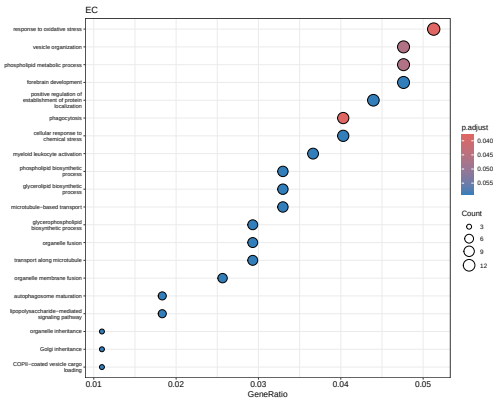

Figure 61: GO Analysis of EC cell type for Bayesian GLM genes in the CCR2KO Dataset

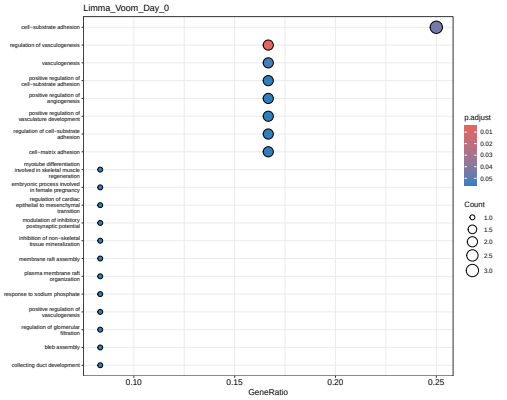

Figure 62: GO Analysis of EC cell type at 0 Days of Injury for Limma-Voom genes in the Ageing Muscle Dataset

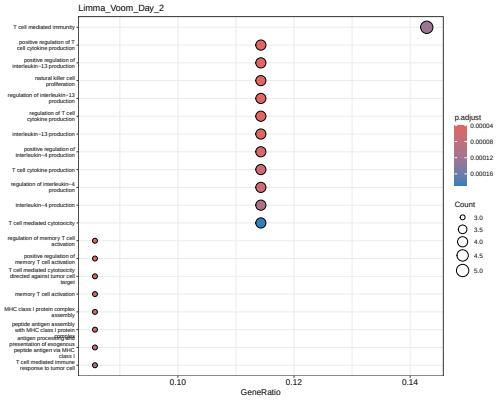

Figure 63: GO Analysis of EC cell type at 2 Days of Injury for Limma-Voom genes in the Ageing Muscle Dataset

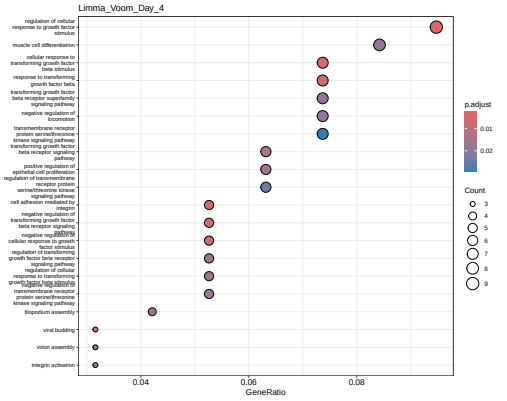

Figure 64: GO Analysis of EC cell type at 4 Days of Injury for Limma-Voom genes in the Ageing Muscle Dataset

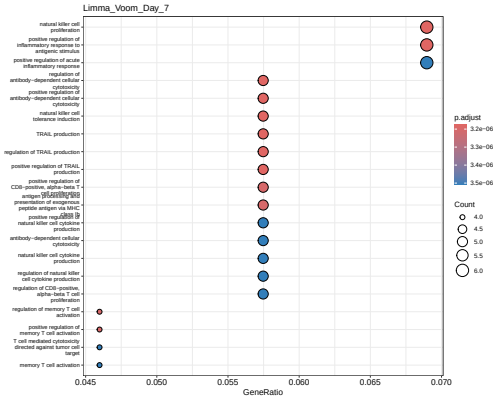

Figure 65: GO Analysis of EC cell type at 7 Days of Injury for Limma-Voom genes in the Ageing Muscle Dataset

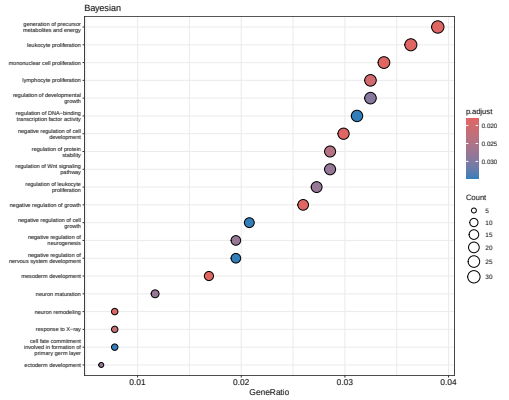

Figure 66: GO Analysis of EC cell type for Bayesian GLM genes in the Ageing Muscle Dataset

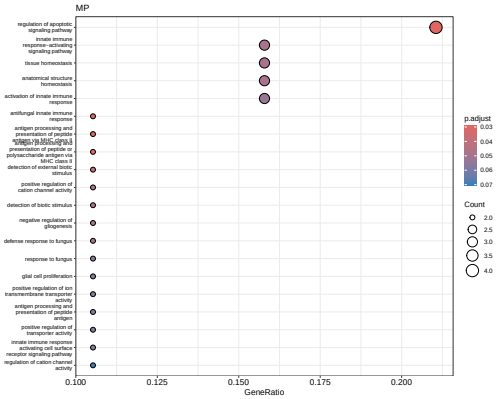

Figure 67: GO Analysis of MP cell type for DESeq2 genes in the CCR2KO

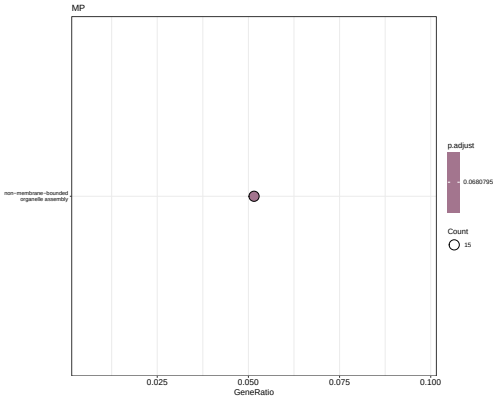

Figure 68: GO Analysis of MP cell type for Bayesian GLM genes in the CCR2KO

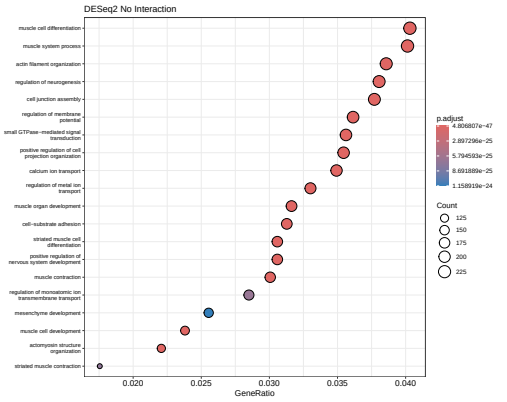

Figure 69: GO Analysis of MuSC cell type for DESeq2 genes No Interaction in the Ageing Muscle Dataset

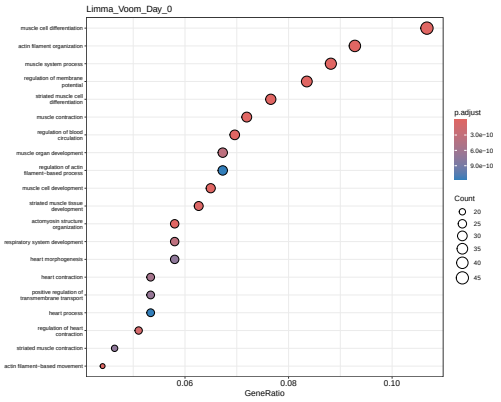

Figure 70: GO Analysis of MuSC cell type at 0 Days of Injury for Limma-Voom genes in the Ageing Muscle Dataset

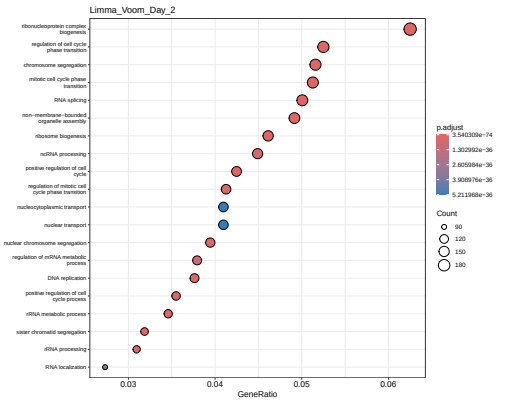

Figure 71: GO Analysis of MuSC cell type at 2 Days of Injury for Limma-Voom genes in the Ageing Muscle Dataset

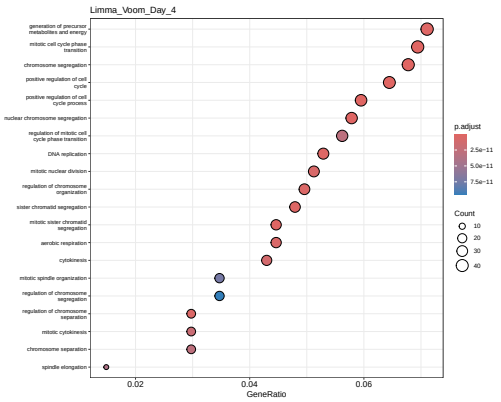

Figure 72: GO Analysis of MuSC cell type at 4 Days of Injury for Limma-Voom genes in the Ageing Muscle Dataset

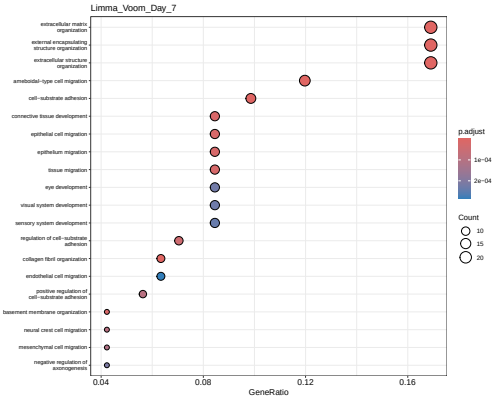

Figure 73: GO Analysis of MuSC cell type at 7 Days of Injury for Limma-Voom genes in the Ageing Muscle Dataset

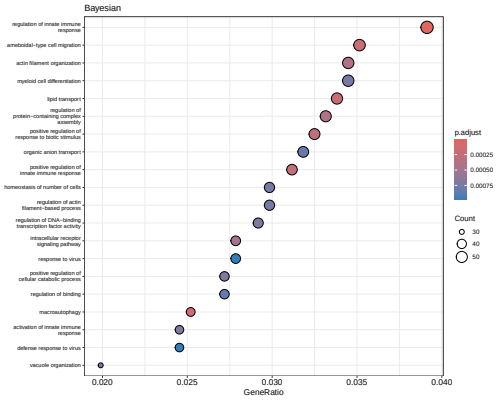

Figure 74: GO Analysis of MuSC cell type for Bayesian GLM genes in the Ageing Muscle Dataset

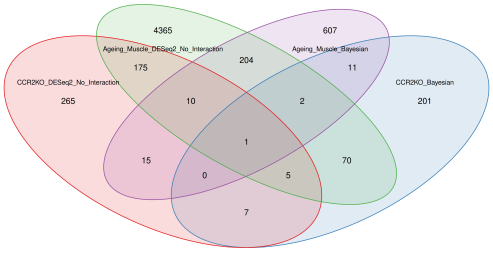

Figure 75: Venn Diagram of EC Results Across Both Datasets

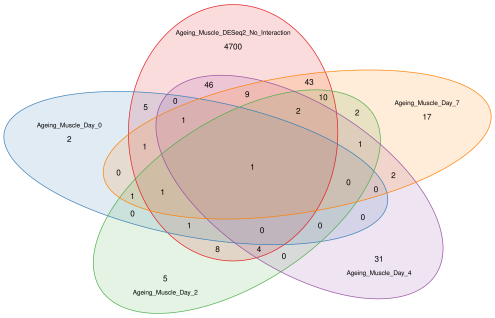

Figure 76: Venn Diagram of EC Results from Limma-Voom and DESeq2 No Interaction in the Ageing Muscle Dataset

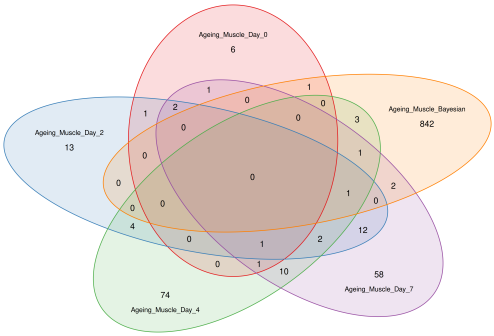

Figure 77: Venn Diagram of EC Results from Limma-Voom and Bayesian GLM in the Ageing Muscle Dataset

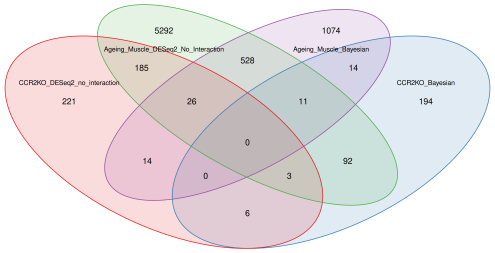

Figure 78: Venn Diagram of MP and MuSC Results from Both Datasets

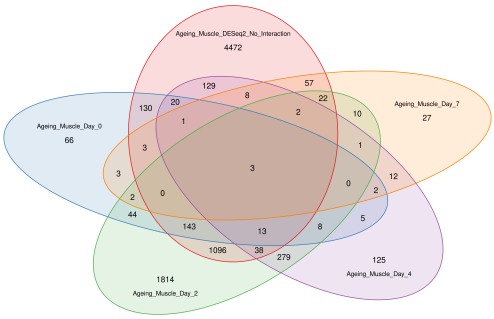

Figure 79: Venn Diagram of MuSC Results from Limma-Voom and DESeq2 No Interaction in the Ageing Muscle Dataset

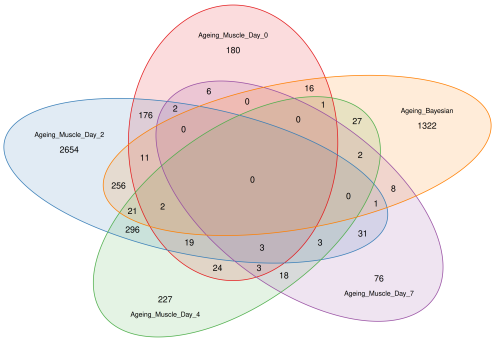

Figure 80: Venn Diagram of MuSC Results from Limma-Voom and Bayesian GLM in the Ageing Muscle Dataset

Figure 81: Venn Diagram of MuSC Results from Limma-Voom in the Ageing Muscle Dataset

A.2 Source Code

Due to the sheer volume of code, only the move relevant code is included in the Appendix. The supplementary files will include all source code and data files from the sourced databases and projects. The code below is an example of the R code used to preprocess and analyse the CCR2KO dataset. Similar code is used to preprocess and analyse the Ageing Muscle dataset. Additional Python code was used to generate the NLP comparative analysis of GO Analysis labels and the Word Cloud visuals.

A.3 CCR2KO Dataset Code in R

```
####################
####################
### 1. Packages ###
####################
```

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


if (!require("BiocManager", quietly = TRUE))
  install.packages("BiocManager")

library(BiocParallel)
library(parallel)
library(readr)
library(dplyr)
library(tximeta)
library(GenomicFeatures)
library(magrittr)
library(DESeq2)
library(vsn)
library(dplyr)
library(ggplot2)
library(pheatmap)
library(RColorBrewer)
library(PoiClaClu)
library(glmpca)
library(fastICA)
library(ggbeeswarm)
library(apeglm)
library(genefilter)
library(IHW)
library(biomaRt)
library(AnnotationDbi)
library(org.Hs.eg.db)
library(ReportingTools)
library(Gviz)
library(sva)
library(RUVSeq)
library(fission)
library(dtw)
library(clusterProfiler)
library(org.Mm.eg.db)
library(GOSemSim)
library(edgeR)
library(stringr)
library(limma)
library(Matrix)
library(cluster)
library(factoextra)
library(tidyr)
library(rstanarm)
library(VennDiagram)
library(tidyverse)
library(grid)
library(grDevices)
library(scales)
library(ggVennDiagram)

################################
################################
### 2. Essential Functions ###
################################
################################
```

```
# The function below is used to pre-process the dataframes
# The symbol columns in data_files must be made into rownames, then removed
# The dataframe must be converted into numeric values
# Any NA values must be removed and replaced with 0
# There can only be integer values in the dataframe
preprocessing <- function(data_files) {
  # Remove the symbol column and convert the dataframe to numeric
  data_files <- lapply(data_files, function(data_file) {
    rownames(data_file) <- data_file$symbol
    data_file <- data_file[, -1]
    # print(colnames(data))
    data_file <- data_file %>% mutate_all(as.numeric)
    data_file[is.na(data_file)] <- 0
    data_file <- round(data_file)
    data_file <- DGEList(data_file)
    data_file <- calcNormFactors(data_file)
    print(head(data_file))
    # Print the dimensions of the data
    print(dim(data_file))
    # Define cutoff for lowly expressed genes
    cutoff <- 1
    # Filter out lowly expressed genes
    drop <- which(apply(cpm(data_file), 1, max) < cutoff)
    data_file <- data_file[-drop,]
    print(dim(data_file))
  })
  return(data_files)
}

# Function to combine datasets
combine_datasets <- function(wt_data, ko_data) {
  print(colnames(wt_data))
  print(colnames(ko_data))
  common_columns <- intersect(colnames(wt_data), colnames(ko_data))
  print(common_columns)
  combined_data <- wt_data %>%
    select(common_columns) %>%
    rename_with(~ paste0(., "_WT"), -symbol) %>%
    left_join(ko_data %>%
                select(common_columns) %>%
                rename_with(~ paste0(., "_KO"), -symbol), by = "symbol")
  return(combined_data)
}

# This function splits the combined datasets by sample id, which is listed as
# "-1", "-2", "-3", etc. The new datasets will all have the symbol column
# and the columns containing the sample id
split_by_sample <- function(combined_data) {
  sample_ids <- str_extract(colnames(combined_data), "-[0-9]+") %>%  unique
  sample_ids <- sample_ids[!is.na(sample_ids)]
  print(sample_ids)
  split_data <- lapply(sample_ids, function(sample_id) {
    sample_columns <- grep(sample_id, colnames(combined_data))
    combined_data %>%
      select(symbol, all_of(sample_columns))
  })
  names(split_data) <- sample_ids
  return(split_data)
}
```

```
# Function to create volcano plot
create_volcano_plot <- function(fit, coef_index, title) {
  top_table <- topTable(fit, coef = coef_index, adjust.method = "BH", number =
      ↪ Inf)
  top_table$Significant <- top_table$adj.P.Val < 0.05
  top_table$symbol <- rownames(top_table)
  # Move symbol to the first column
  top_table <- top_table[, c(ncol(top_table), 1:(ncol(top_table) - 1))]
  # Order the table by adj.P.Val
  top_table <- top_table[order(top_table$adj.P.Val),]
  print(head(top_table))
  # Save the significant genes to a CSV file using the combined dataset
      ↪ directory with rownames
  write_csv(top_table, file = file.path(combined_dataset_directory, paste0(title
      ↪ , ".csv")))
  volcano_plot <- ggplot(top_table, aes(x = logFC, y = -log10(adj.P.Val), color
      ↪ = Significant)) +
    geom_point(alpha = 0.4) +
    ggtitle(title) +
    xlab("Log2 Fold Change") +
    ylab("-Log10 P-Value") +
    scale_color_manual(values = c("black", "red")) +
    geom_vline(xintercept = c(-1, 1), col = "red", linetype = "dashed") +
    geom_hline(yintercept = -log10(0.05), col = "red", linetype = "dashed")
  guides(color = FALSE)
  # Return the plot
  return(volcano_plot)
}

# Preprocess and analyze each dataset for volcano plots
analyse_and_plot <- function(combined_data, dataset_name) {
  symbols <- combined_data$symbol
  sample_csv <- combined_data[, -1]
  rownames(sample_csv) <- symbols

  # Create DGEList object
  d0 <- DGEList(sample_csv)
  d0 <- calcNormFactors(d0)
  cutoff <- 1
  drop <- which(apply(cpm(d0), 1, max) < cutoff)
  d <- d0[-drop, ]

  snames <- colnames(sample_csv)

  # Extract treatment and time
  treatment <- str_extract(snames, "(?<=_)[A-Z]+")
  time <- str_extract(snames, "\\d+(?=d)")

  group <- interaction(treatment, time)
  mm <- model.matrix(~0 + treatment)
  colnames(mm) <- c("KO", "WT")
  # colnames(mm) <- levels(group)

  # Voom transformation
  y <- voom(d, mm, plot = FALSE)

  # Fit linear model and empirical Bayes
  fit <- lmFit(y, mm)
```

```
  fit <- eBayes(fit)

  # Create volcano plot for the WT vs KO contrast
  contr <- makeContrasts(WTvsKO = WT - KO, levels = colnames(coef(fit)))
  # contr <- makeContrasts(WTvsKO = treatmentWT - treatmentKO, levels = colnames
      ↪ (coef(fit)))
  contrast_tmp <- contrasts.fit(fit, contr)
  contrast_tmp <- eBayes(contrast_tmp)
  create_volcano_plot(contrast_tmp, 1, paste0(dataset_name, ": WT vs KO, adj.P.
      ↪ Val"))
}

# Function to create volcano plot with respect to cell type and days from injury
create_volcano_plot_by_cell_type_and_days_from_injury <- function(fit,
    ↪ coef_index, title, output_dir) {
  top_table <- topTable(fit, coef = coef_index, adjust.method = "BH", number =
      ↪ Inf)
  top_table$Significant <- top_table$adj.P.Val < 0.05
  top_table$symbol <- rownames(top_table)

  # Move symbol to the first column
  top_table <- top_table[, c(ncol(top_table), 1:(ncol(top_table) - 1))]

  # Order the table by adj.P.Val
  top_table <- top_table[order(top_table$adj.P.Val),]
  print(head(top_table))

  # Save the significant genes to a CSV file using the combined dataset
      ↪ directory with rownames
  write_csv(top_table, file = file.path(output_dir, paste0(title, ".csv")))

  volcano_plot <- ggplot(top_table, aes(x = logFC, y = -log10(adj.P.Val), color
      ↪ = Significant)) +
    geom_point(alpha = 0.4) +
    ggtitle(title) +
    xlab("Log2 Fold Change") +
    ylab("-Log10 P-Value") +
    scale_color_manual(values = c("black", "red")) +
    geom_vline(xintercept = c(-1, 1), col = "red", linetype = "dashed") +
    geom_hline(yintercept = -log10(0.05), col = "red", linetype = "dashed") +
    guides(color = FALSE)

  # Save the volcano plot
  ggsave(file.path(output_dir, paste0(title, ".png")), plot = volcano_plot, dpi
      ↪ = 300, width = 10, height = 6)

  # Return the plot
  return(volcano_plot)
}


# Preprocess and analyze each dataset for volcano plots
analyse_and_plot_by_cell_type_and_days_from_injury <- function(combined_data,
    ↪ dataset_name, time_points, output_dir) {
  symbols <- combined_data$symbol
  sample_csv <- combined_data[, -1]
  rownames(sample_csv) <- symbols

  # Create DGEList object
```

```
  d0 <- DGEList(sample_csv)
  d0 <- calcNormFactors(d0)
  cutoff <- 1
  drop <- which(apply(cpm(d0), 1, max) < cutoff)
  d <- d0[-drop, ]

  snames <- colnames(sample_csv)

  # Extract treatment and time
  treatment <- factor(str_extract(snames, "(?<=_)[A-Z]+"))
  time <- factor(str_extract(snames, "\\d+(?=d)"))

  group <- interaction(treatment, time)
  plotMDS(d, col = as.numeric(group))

  mm <- model.matrix(~0 + group)
  y <- voom(d, mm, plot = TRUE)

  fit <- lmFit(y, mm)


  for (time_point in time_points) {
    for (cell_type in levels(treatment)) {
      contrast_name <- paste0("group", cell_type, ".", time_point, "-groupWT.",
          ↪ time_point)
      contr <- makeContrasts(contrasts = contrast_name, levels = colnames(coef(
          ↪ fit)))
      tmp <- tryCatch({
        tmp <- contrasts.fit(fit, contr)
        tmp <- eBayes(tmp)
        plot_title <- paste0(dataset_name, ": ", cell_type, " vs WT at ",
            ↪ time_point, " days, adj.P.Val")
        volcano_plot <- create_volcano_plot_by_cell_type_and_days_from_injury(
            ↪ tmp, 1, plot_title, output_dir)
      }, error = function(e) {
        message(paste("Skipping", cell_type, "at time point", time_point, "due
            ↪ to error:", e$message))
        NULL
      })
    }
  }
}

# Function to perform k-means clustering and silhouette score
k_means_clustering <- function(input_csv_path, num_clusters = 3) {
  # Read in the sample CSV
  sample_csv <- read_csv(input_csv_path)

  # Set sample_csv as a dataframe
  sample_csv <- as.data.frame(sample_csv)

  # Preprocess the data
  symbols <- sample_csv$symbol
  sample_csv <- sample_csv[, -1]
  rownames(sample_csv) <- symbols

  # Create DGEList object
  d0 <- DGEList(sample_csv)
  d0 <- calcNormFactors(d0)
```

```
cutoff <- 1
drop <- which(apply(cpm(d0), 1, max) < cutoff)
d <- d0[-drop, ]

# Extract normalised counts
normalised_counts <- cpm(d, log = FALSE)

# Check for negative values in the raw data
if (any(normalised_counts < 0)) {
  stop("Negative counts found in the raw data")
} else {
  print("No negative counts found in the raw data")
}

# Performs k-means clustering
set.seed(123)
kmeans_result <- kmeans(normalised_counts, centers = num_clusters, nstart =
    ↪ 25)

# Add cluster assignments to the data
cluster_assignment <- kmeans_result$cluster

# Print the number of samples in each cluster
print(table(cluster_assignment))

# Compute silhouette score
silhouette_result <- silhouette(cluster_assignment, dist(normalised_counts))
avg_silhouette <- mean(silhouette_result[, 3])
print(paste("Average silhouette score: ", avg_silhouette))

# Visualise the silhouette plot
silhouette_plot <- fviz_silhouette(silhouette_result)
print(silhouette_plot)

# Extract genes belonging to each cluster
genes_by_cluster <- split(rownames(normalised_counts), cluster_assignment)

# Function to perform differential expression analysis for a cluster
analyse_cluster <- function(cluster_genes) {
  cluster_data <- sample_csv[cluster_genes, ]
  cluster_data <- as.matrix(cluster_data)
  storage.mode(cluster_data) <- "numeric"
  if (any(cluster_data < 0)) {
    # Print the negative values triggering the error
    print(cluster_data[cluster_data < 0])
  }
  dge_cluster <- DGEList(cluster_data)
  dge_cluster <- calcNormFactors(dge_cluster)

  # Extract treatment groups
  snames <- colnames(cluster_data)
  treatment <- str_extract(snames, "(?<=_)[A-Z]+")
  group <- factor(treatment)

  # Check if group has at least two levels
  if (length(levels(group)) < 2) {
    cat("Skipping cluster with less than 2 levels in group.\n")
    return(NULL)
  }
```

```
    # group <- factor(c(rep("WT", ncol(cluster_data)/2), rep("KO", ncol(
        ↪ cluster_data)/2)))
    design <- model.matrix(~0 + group)
    colnames(design) <- levels(group)
    y_cluster <- voom(dge_cluster, design, plot = FALSE)
    fit_cluster <- lmFit(y_cluster, design)
    fit_cluster <- eBayes(fit_cluster)
    top_table_cluster <- topTable(fit_cluster, adjust.method = "BH", number =
        ↪ Inf)
    top_table_cluster$symbol <- cluster_genes
    # Make sure symbol is moved to the first column
    top_table_cluster <- top_table_cluster[, c(ncol(top_table_cluster), 1:(ncol(
        ↪ top_table_cluster) - 1))]
    return(as.data.frame(top_table_cluster))
  }

  # Perform differential expression analysis for each cluster
  cluster_results <- lapply(genes_by_cluster, function(genes) {
    if (length(genes) < 2) {
      cat("Skipping cluster with fewer than 2 samples.\n")
      return(NULL)
    }
    tryCatch({
      analyse_cluster(genes)
    }, error = function(e) {
      cat("Error in cluster: ", e$message, "\n")
      return(NULL)
    })
  })

  # Print top genes for each cluster
  lapply(cluster_results, function(result) {
    if (!is.null(result)) {
      head(result, 10)
    } else {
      cat("No result for this cluster.\n")
    }
  })

  # Print top genes for each cluster
  lapply(cluster_results, function(result) head(result, 10))

  # Add symbols to the kmeans_result
  kmeans_result_df <- data.frame(symbol = rownames(normalised_counts), cluster =
      ↪  kmeans_result$cluster)

  # Add symbols to the silhouette_result
  silhouette_result_df <- data.frame(symbol = rownames(normalised_counts),
      ↪ silhouette_result[, 1:3])


  return(list(
    kmeans_result = kmeans_result_df,
    silhouette_result = silhouette_result_df,
    avg_silhouette = avg_silhouette,
    cluster_results = cluster_results
  ))
}
```

```
# Function to perform ICA and k-means clustering
ica_k_means_clustering <- function(input_csv_path, num_clusters = 3,
    ↪ num_components = 2) {
  # Read in the sample CSV
  sample_csv <- read_csv(input_csv_path)

  # Set sample_csv as a dataframe
  sample_csv <- as.data.frame(sample_csv)

  # Preprocess the data
  symbols <- sample_csv$symbol
  sample_csv <- sample_csv[, -1]
  rownames(sample_csv) <- symbols

  # Create DGEList object
  d0 <- DGEList(sample_csv)
  d0 <- calcNormFactors(d0)
  cutoff <- 1
  drop <- which(apply(cpm(d0), 1, max) < cutoff)
  d <- d0[-drop, ]

  # Extract normalized counts
  normalized_counts <- cpm(d, log = FALSE)

  # Check for negative values in the raw data
  if (any(normalized_counts < 0)) {
    stop("Negative counts found in the raw data")
  } else {
    print("No negative counts found in the raw data")
  }

  # Perform ICA
  set.seed(123)
  ica_result <- fastICA(normalized_counts, n.comp = num_components, method = "C
    ↪ ")

  # Perform k-means clustering on ICA components
  ica_components <- as.data.frame(ica_result$S)
  kmeans_result <- kmeans(ica_components, centers = num_clusters, nstart = 25)
  cluster_assignment <- kmeans_result$cluster

  # Add cluster assignments to the ICA components
  ica_components$cluster <- cluster_assignment
  ica_components$symbol <- rownames(normalized_counts)

  # Print the number of samples in each cluster
  print(table(cluster_assignment))

  # Compute silhouette score
  silhouette_result <- silhouette(cluster_assignment, dist(ica_components[, -c(
    ↪ ncol(ica_components), ncol(ica_components)-1)]))
  avg_silhouette <- mean(silhouette_result[, 3])
  print(paste("Average silhouette score: ", avg_silhouette))

  # Visualize the silhouette plot
  silhouette_plot <- fviz_silhouette(silhouette_result)
  print(silhouette_plot)

  # Extract genes belonging to each cluster
```

```
genes_by_cluster <- split(rownames(normalized_counts), cluster_assignment)

# Function to perform differential expression analysis for a cluster
analyse_cluster <- function(cluster_genes) {
  cluster_data <- sample_csv[cluster_genes, ]
  cluster_data <- as.matrix(cluster_data)
  storage.mode(cluster_data) <- "numeric"
  if (any(cluster_data < 0)) {
    # Print the negative values triggering the error
    print(cluster_data[cluster_data < 0])
  }
  dge_cluster <- DGEList(cluster_data)
  dge_cluster <- calcNormFactors(dge_cluster)

  # Extract treatment groups
  snames <- colnames(cluster_data)
  treatment <- str_extract(snames, "(?<=_)[A-Z]+")
  group <- factor(treatment)

  # Check if group has at least two levels
  if (length(levels(group)) < 2) {
    cat("Skipping cluster with less than 2 levels in group.\n")
    return(NULL)
  }
  design <- model.matrix(~0 + group)
  colnames(design) <- levels(group)
  y_cluster <- voom(dge_cluster, design, plot = FALSE)
  fit_cluster <- lmFit(y_cluster, design)
  fit_cluster <- eBayes(fit_cluster)
  top_table_cluster <- topTable(fit_cluster, adjust.method = "BH", number =
      ↪ Inf)
  top_table_cluster$symbol <- cluster_genes
  # Make sure symbol is moved to the first column
  top_table_cluster <- top_table_cluster[, c(ncol(top_table_cluster), 1:(ncol(
      ↪ top_table_cluster) - 1))]
  return(as.data.frame(top_table_cluster))
}

# Perform differential expression analysis for each cluster
cluster_results <- lapply(genes_by_cluster, function(genes) {
  if (length(genes) < 2) {
    cat("Skipping cluster with fewer than 2 samples.\n")
    return(NULL)
  }
  tryCatch({
    analyse_cluster(genes)
  }, error = function(e) {
    cat("Error in cluster: ", e$message, "\n")
    return(NULL)
  })
})

# Print top genes for each cluster
lapply(cluster_results, function(result) {
  if (!is.null(result)) {
    head(result, 10)
  } else {
    cat("No result for this cluster.\n")
  }
```

```
  })

  # Add symbols to the kmeans_result
  kmeans_result_df <- data.frame(symbol = rownames(normalized_counts), cluster =
      ↪  kmeans_result$cluster)

  # Add symbols to the silhouette_result
  silhouette_result_df <- data.frame(symbol = rownames(normalized_counts),
      ↪ silhouette_result[, 1:3])

  return(list(
    ica_components = ica_components,
    kmeans_result = kmeans_result_df,
    silhouette_result = silhouette_result_df,
    avg_silhouette = avg_silhouette,
    cluster_results = cluster_results
  ))
}

# Save results to CSV
save_results_to_csv <- function(results, dataset_name, output_dir) {
  for (result_name in names(results)) {
    result <- results[[result_name]]
    if (inherits(result, "data.frame")) {
      # Save data frame results
      write_csv(result, file = file.path(output_dir, paste0(dataset_name, "_",
          ↪ result_name, ".csv")))
    } else if (is.numeric(result)) {
      # Save numeric result (avg_silhouette)
      avg_silhouette_df <- data.frame(avg_silhouette = result)
      write_csv(avg_silhouette_df, file = file.path(output_dir, paste0(
          ↪ dataset_name, "_", result_name, ".csv")))
    } else if (is.list(result)) {
      for (i in seq_along(result)) {
        sub_result <- result[[i]]
        if (is.data.frame(sub_result)) {
          write_csv(sub_result, file = file.path(output_dir, paste0(dataset_name
              ↪ , "_", result_name, "_cluster_", i, ".csv")))
        }
      }
    }
  }
}

############################
############################
### 3. Data Engineering ###
############################
############################

# Set the parent directory
# parent_dir <- "/path/to/your/parent/directory"
parent_dir <- "/home/henry-cao/Desktop/KCL/Individual_Project/Data/Final_Data/
    ↪ Code_Implementation/Code_Results"

# Define subdirectories
data_dir <- file.path(parent_dir, "Data")
output_dir <- file.path(parent_dir, "Output")
vis_dir <- file.path(output_dir, "Visuals")
```

```
# Create directories if they don't exist
dir.create(data_dir, recursive = TRUE, showWarnings = FALSE)
dir.create(output_dir, recursive = TRUE, showWarnings = FALSE)
dir.create(vis_dir, recursive = TRUE, showWarnings = FALSE)


# List of raw count files
raw_count_files <- list(
  EC_CCR2KO = "GSE210748_Raw_counts_EC_CCR2KO.csv",
  EC_WT = "GSE210748_Raw_counts_EC_WT.csv",
  FAP_CCR2KO = "GSE210748_Raw_counts_FAP_CCR2KO.csv",
  FAP_WT = "GSE210748_Raw_counts_FAP_WT.csv",
  Inflammatory_WT = "GSE210748_Raw_counts_Inflammatory_WT.csv",
  MP_CCR2KO = "GSE210748_Raw_counts_MP_CCR2KO.csv",
  MP_WT = "GSE210748_Raw_counts_MP_WT.csv",
  Pericyte_WT = "GSE210748_Raw_counts_Pericyte_WT.csv",
  Total_Counts = "GSE210748_Raw_counts_total_tissue_WT.csv"
)

# Read all CSV files into a list of data frames
data_list <- lapply(raw_count_files, function(file) {
  read_csv(file = file.path(data_dir, file))
})

# Create new metadata dataframes for each existing dataframe
for (i in seq_along(data_list)) {
  # Extract column names from the data
  col_names <- colnames(data_list[[i]])[-1]  # Drop the first column name

  # Create metadata data frame
  metadata <- data.frame(Sample = col_names)

  # Extract days from injury
  metadata <- metadata %>%
    mutate(
      Days_From_Injury = as.integer(str_extract(Sample, "\\d+(?=d)")),
      Sample_Number = as.integer(str_extract(Sample, "(?<=-)[0-9]+"))  # Extract
          ↪  sample number
    )

  # Save the metadata as a CSV file
  write.csv(metadata, file = file.path(data_dir, paste0("metadata_", names(
      ↪ data_list)[i], ".csv")), row.names = FALSE)
}


#####################################
#####################################
### 4. Exploratory Data Analysis ###
#####################################
#####################################

# Match up the wild type and knockout datasets
# EC_CCR2KO and EC_WT
# FAP_CCR2KO and FAP_WT
# MP_CCR2KO and MP_WT
# The datasets must be combined, so that the timepoints and the samples are
    ↪ matched
```

```
# This can be done by creating a new dataset that combines the columns of
# wild type and knockout datasets
# We first find the intersect of the column names of the two datasets.
# Then we add the columns of the two dataset together.
# Note that the symbol column only needs to appear once in the new dataset.
# The other columns would need to be renamed to avoid conflicts, preferrably
# by indicating knockout status

# Combine the datasets
# combined_dataset_directory should be a subfolder of the data directory
combined_dataset_directory <- file.path(data_dir, "Combined_Datasets")

# Create the combined dataset directory if it doesn't exist
dir.create(combined_dataset_directory, recursive = TRUE, showWarnings = FALSE)

combined_data_list <- list(
  EC_combined = combine_datasets(data_list$EC_WT, data_list$EC_CCR2KO),
  FAP_combined = combine_datasets(data_list$FAP_WT, data_list$FAP_CCR2KO),
  MP_combined = combine_datasets(data_list$MP_WT, data_list$MP_CCR2KO)
)

# For each combined dataset, save it as a csv file
for (name in names(combined_data_list)) {
  write_csv(combined_data_list[[name]], file = file.path(
      ↪ combined_dataset_directory, paste0(name, ".csv")))
}

# Split the combined datasets by sample
split_data_list <- list(
  EC_split = split_by_sample(combined_data_list$EC_combined),
  FAP_split = split_by_sample(combined_data_list$FAP_combined),
  MP_split = split_by_sample(combined_data_list$MP_combined)
)

# Print the split datasets
lapply(split_data_list, head)

# Sample ID dataset directory should be a subdirectory of the data directory
sample_id_dataset_directory <- file.path(data_dir, "Sample_ID_Datasets")

# Create the sample ID dataset directory if it doesn't exist
dir.create(sample_id_dataset_directory, recursive = TRUE, showWarnings = FALSE)

for (name in names(split_data_list)) {
  for (sample_id in names(split_data_list[[name]])) {
    # Strip the "-" from the sample id
    sample_id_clean <- gsub("-", "", sample_id)
    write_csv(split_data_list[[name]][[sample_id]] %>% as.data.frame, file =
        ↪ file.path(sample_id_dataset_directory, paste0(name, "_",
        ↪ sample_id_clean, ".csv")))
  }
}

# Function to read and label datasets
read_and_label_dataset <- function(file_path, cell_type) {
  data <- read_csv(file_path)
  colnames(data) <- paste0(colnames(data), ".", cell_type)
  colnames(data)[1] <- "symbol"
  return(data)
```

```
}

# Path to the datasets
ec_path <- file.path(combined_dataset_directory, "EC_combined.csv")
fap_path <- file.path(combined_dataset_directory, "FAP_combined.csv")
mp_path <- file.path(combined_dataset_directory, "MP_combined.csv")

# Read and label datasets
ec_data <- read_and_label_dataset(ec_path, "EC")
fap_data <- read_and_label_dataset(fap_path, "FAP")
mp_data <- read_and_label_dataset(mp_path, "MP")

# Combine datasets by adding the columns together, preserving the symbol column
combined_data <- ec_data %>%
  left_join(fap_data, by = "symbol") %>%
  left_join(mp_data, by = "symbol")

# Ensure that the combined_data is a dataframe
combined_data <- as.data.frame(combined_data)

# # Read in the metadata_combined file
# metadata_combined <- read_csv(file.path(combined_dataset_directory, "
    ↪ metadata_combined.csv"))

# Make sure the first column of the metadata is the colnames of the
    ↪ combined_data
colnames_combined <- colnames(combined_data)

# Print the colnames_combined
print(colnames_combined)

# Drop the first row of the colnames_combined
colnames_combined <- colnames_combined[-1]

# Create the metadata file for the combined dataset
# Days from injury is the number that precedes "d"
# Sample is the number that follows "-"
# Knockout status is the characters that follow "_" but precede "."
# Cell type is the characters after "."
metadata_combined <- data.frame(Sample = colnames(combined_data)[-1])  # Exclude
    ↪  the 'symbol' column
metadata_combined <- metadata_combined %>%
  mutate(
    Days_From_Injury = as.numeric(str_extract(Sample, "\\d+(?=d)")),
    Sample_Number = as.numeric(str_extract(Sample, "(?<=-)[0-9]+")),
    Knockout_Status = str_extract(Sample, "(?<=_)[A-Z]+(?=\\.)"),
    Cell_Type = str_extract(Sample, "(?<=\\.)[A-Z]+")
  )


# Rename the first column as colnames.data.
colnames(metadata_combined)[1] <- "colnames.data."
print(colnames(metadata_combined))

print(head(metadata_combined))

# Save the combined_data as a CSV file
write_csv(combined_data, file = file.path(combined_dataset_directory, "
    ↪ Combined_Dataset.csv"))
```

```
write_csv(metadata_combined, file = file.path(combined_dataset_directory, "
    ↪ metadata_combined.csv"))

#########################
### ICA and PCA Plots ###
#########################

# Convert combined data to matrix
cts <- as.matrix(combined_data %>% select(-symbol))
rownames(cts) <- combined_data$symbol

# Print how many rows in cts
print(dim(cts))

# Ensure the order of samples in metadata matches the columns in cts
metadata_combined <- metadata_combined %>% filter(colnames.data. %in% colnames(
    ↪ cts))
cts <- cts[, metadata_combined$colnames.data.]
print(head(cts))

# Create the DESeq2 dataset
dds <- DESeqDataSetFromMatrix(countData = cts,
                              colData = metadata_combined,
                              design = ~ Cell_Type + Days_From_Injury +
                                  ↪ Knockout_Status + Cell_Type:
                                  ↪ Days_From_Injury + Cell_Type:
                                  ↪ Knockout_Status + Days_From_Injury:
                                  ↪ Knockout_Status)

# Perform DESeq2 normalisation
dds <- DESeq(dds)

# Apply VST
vsd <- vst(dds, blind = FALSE)

# Extract the VST-transformed data
vst_data <- assay(vsd)

# Print the number of columns (genes) before removing zero variance columns
print(dim(vst_data))

# Remove columns (genes) with zero variance
vst_data <- vst_data[apply(vst_data, 1, var) != 0,]

# Print the number of columns (genes) after removing zero variance columns
print(dim(vst_data))

# Perform PCA for dimensionality reduction
pca_result <- prcomp(t(vst_data), center = TRUE, scale. = TRUE)

# Determine the number of components to keep (e.g., 50)
num_pcs <- 50
pca_data <- pca_result$x[, 1:num_pcs]

# Perform ICA
set.seed(123)
ica_result <- fastICA(pca_data, n.comp = 2, method = "C")
```

```
# Extract the independent components
ics <- ica_result$S

# Create a data frame with the IC's and sample information
ica_data <- data.frame(IC1 = ics[, 1], IC2 = ics[, 2])
ica_data$Cell_Type <- colData(vsd)$Cell_Type
ica_data$Days_From_Injury <- colData(vsd)$Days_From_Injury
ica_data$Knockout_Status <- colData(vsd)$Knockout_Status

# Convert factgors to numeric for plotting
ica_data$Cell_Type <- as.factor(ica_data$Cell_Type)
ica_data$Knockout_Status <- as.factor(ica_data$Knockout_Status)
ica_data$Days_From_Injury <- as.numeric(ica_data$Days_From_Injury)

# Saving the ICA component data
write_csv(ica_data, file = file.path(combined_dataset_directory, "ICA_Dataset.
    ↪ csv"))

# Create ICA plot version 1, with colour = Cell_Type, shape = Knockout_Status,
    ↪ size = Days_From_Injury
ica_plot_1 <- ggplot(ica_data, aes(x = IC1, y = IC2, color = Cell_Type, shape =
    ↪ Knockout_Status, size = Days_From_Injury)) +
  geom_point() +
  labs(title = "ICA of Combined Dataset",
       x = "Independent Component 1",
       y = "Independent Component 2",
       color = "Cell Type",
       shape = "Knockout Status",
       size = "Days From Injury") +
  theme(plot.title = element_text(hjust = 0.5))

# Display ICA plot version 1
print(ica_plot_1)

# Save ICA plot version 1
ggsave(file.path(vis_dir, "ICA_Plot_1.png"))

# Create ICA plot version 2, with colour = Knockout_Status, shape = Cell_Type,
    ↪ size = Days_From_Injury
ica_plot_2 <- ggplot(ica_data, aes(x = IC1, y = IC2, color = Knockout_Status,
    ↪ shape = Cell_Type, size = Days_From_Injury)) +
  geom_point() +
  labs(title = "ICA of Combined Dataset",
       x = "Independent Component 1",
       y = "Independent Component 2",
       color = "Knockout Status",
       shape = "Cell Type",
       size = "Days From Injury") +
  theme(plot.title = element_text(hjust = 0.5))

# Display ICA plot version 2
print(ica_plot_2)

# Save ICA plot version 2
ggsave(file.path(vis_dir, "ICA_Plot_2.png"))


# Extract the top two principal components for the PCA Plot
pca_result_top_two <- data.frame(pca_result$x[, 1:2])
```

```
colnames(pca_result_top_two) <- c("PC1", "PC2")

# Add Sample information to the PCA data
pca_result_top_two$Cell_Type <- colData(vsd)$Cell_Type
pca_result_top_two$Knockout_Status <- colData(vsd)$Knockout_Status
pca_result_top_two$Days_From_Injury <- colData(vsd)$Days_From_Injury

# Convert factors to numeric for plotting
pca_result_top_two$Cell_Type <- as.factor(pca_result_top_two$Cell_Type)
pca_result_top_two$Knockout_Status <- as.factor(
    ↪ pca_result_top_two$Knockout_Status)
pca_result_top_two$Days_From_Injury <- as.numeric(
    ↪ pca_result_top_two$Days_From_Injury)

# Saving the PCA component data
write_csv(pca_result_top_two, file = file.path(combined_dataset_directory, "
    ↪ PCA_Dataset.csv"))

# Calculate the percentage of variance explained by each principal component
percent_var <- round(100 * pca_result$sdev^2 / sum(pca_result$sdev^2), 1)
percent_var_df <- as.data.frame(percent_var)
rownames(percent_var_df) <- paste0("PC", 1:length(percent_var))
# Make sure the rownames show up in the csv
percent_var_df <- data.frame(PC = rownames(percent_var_df), Variance_Explained =
    ↪  percent_var_df$percent_var)
write_csv(percent_var_df, file = file.path(combined_dataset_directory, "
    ↪ PCA_Percentage_Variance_Explained.csv"))

# Create PCA Plot version 1, with colour = Cell_Type, shape = Knockout_Status,
    ↪ size = Days_From_Injury
pca_plot_1 <- ggplot(pca_result_top_two, aes(x = PC1, y = PC2, color = Cell_Type
    ↪ , shape = Knockout_Status, size = Days_From_Injury)) +
  geom_point() +
  labs(title = "PCA of Combined Dataset",
       x = paste0("PC1 (", percent_var_df$Variance_Explained[1], "%)"),
       y = paste0("PC2 (", percent_var_df$Variance_Explained[2], "%)"),
       color = "Cell Type",
       shape = "Knockout Status",
       size = "Days From Injury") +
  theme(plot.title = element_text(hjust = 0.5))

# Display PCA plot version 1
print(pca_plot_1)

# Save PCA plot version 1
ggsave(file.path(vis_dir, "PCA_Plot_1.png"))



############################
### Bayesian Inference ###
############################

# Load individual datasets
EC_KO_data <- read.csv(file.path(data_dir, "GSE210748_Raw_counts_EC_CCR2KO.csv")
    ↪ )
EC_WT_data <- read.csv(file.path(data_dir, "GSE210748_Raw_counts_EC_WT.csv"))
```

```
FAP_KO_data <- read.csv(file.path(data_dir, "GSE210748_Raw_counts_FAP_CCR2KO.csv
    ↪ "))
FAP_WT_data <- read.csv(file.path(data_dir, "GSE210748_Raw_counts_FAP_WT.csv"))

MP_KO_data <- read.csv(file.path(data_dir, "GSE210748_Raw_counts_MP_CCR2KO.csv")
    ↪ )
MP_WT_data <- read.csv(file.path(data_dir, "GSE210748_Raw_counts_MP_WT.csv"))



# Add cell type and knockout status columns
add_metadata <- function(data, cell_type, knockout_status) {
  data$Cell_Type <- cell_type
  data$Knockout_Status <- knockout_status
  return(data)
}

EC_KO_data <- add_metadata(EC_KO_data, "EC", "KO")
EC_WT_data <- add_metadata(EC_WT_data, "EC", "WT")
FAP_KO_data <- add_metadata(FAP_KO_data, "FAP", "KO")
FAP_WT_data <- add_metadata(FAP_WT_data, "FAP", "WT")
MP_KO_data <- add_metadata(MP_KO_data, "MP", "KO")
MP_WT_data <- add_metadata(MP_WT_data, "MP", "WT")


perform_bayesian_inference <- function(data_subset, cell_type, output_dir) {
  print(paste("Processing cell type:", cell_type))

  library(data.table)
  library(rstanarm)
  library(ggplot2)
  library(parallel)
  library(stringr)

  # Convert to data.table for faster operations
  setDT(data_subset)
  print("Data converted to data.table")

  # Separate KO and WT data
  ko_data <- data_subset[Knockout_Status == "KO"]
  wt_data <- data_subset[Knockout_Status == "WT"]

  # Ensure KO and WT data have the same genes in the same order
  common_genes <- intersect(ko_data$symbol, wt_data$symbol)
  ko_data <- ko_data[symbol %in% common_genes][order(symbol)]
  wt_data <- wt_data[symbol %in% common_genes][order(symbol)]

  # Calculate log fold change (KO / WT)
  ko_expression <- ko_data[, !(c("symbol", "Cell_Type", "Knockout_Status")),
      ↪ with = FALSE]
  wt_expression <- wt_data[, !(c("symbol", "Cell_Type", "Knockout_Status")),
      ↪ with = FALSE]
  log_fold_change <- log2((ko_expression + 1) / (wt_expression + 1))
  print("Log fold change calculated")

  # Extract time points from column names
  time_points <- as.numeric(str_extract(colnames(log_fold_change), "(?<=X)\\d
      ↪ +(?=d)"))
  print(paste("Time points:", paste(time_points, collapse = ", ")))
```

```
fit_gene_model <- function(gene_data, gene_name, time_points) {
  df <- data.frame(y = as.numeric(gene_data), time = time_points)
  fit <- stan_glm(y ~ time, data = df, family = gaussian(),
                  prior = normal(0, 1), prior_intercept = normal(0, 1), seed =
                      ↪ 123)

  posterior_samples <- as.matrix(fit)

  # Get actual column names
  col_names <- colnames(posterior_samples)

  # Identify intercept and time coefficient columns
  intercept_col <- col_names[grep("Intercept", col_names, ignore.case = TRUE)]
  time_col <- col_names[grep("time", col_names, ignore.case = TRUE)]

  if (length(intercept_col) == 0 || length(time_col) == 0) {
    warning(paste("Unable to identify intercept or time coefficient for gene
        ↪ :", gene_name))
    return(NULL)
  }

  # Calculate metrics
  result <- data.table(
    Gene = gene_name,
    Intercept_Mean = mean(posterior_samples[, intercept_col]),
    Intercept_SD = sd(posterior_samples[, intercept_col]),
    Intercept_CI_Lower = quantile(posterior_samples[, intercept_col], 0.025),
    Intercept_CI_Upper = quantile(posterior_samples[, intercept_col], 0.975),
    Time_Coefficient_Mean = mean(posterior_samples[, time_col]),
    Time_Coefficient_SD = sd(posterior_samples[, time_col]),
    Time_Coefficient_CI_Lower = quantile(posterior_samples[, time_col], 0.025)
        ↪ ,
    Time_Coefficient_CI_Upper = quantile(posterior_samples[, time_col], 0.975)
        ↪ ,
    Prob_Direction = mean(posterior_samples[, time_col] > 0),
    ROPE_Proportion = mean(posterior_samples[, time_col] > -0.1 &
        ↪ posterior_samples[, time_col] < 0.1)
  )

  return(result)
}

# Prepare data for parallel processing
gene_data_list <- lapply(1:nrow(log_fold_change), function(i) {
  list(gene_data = log_fold_change[i,],
       gene_name = common_genes[i],
       time_points = time_points)
})

# Use parallel processing to apply the model to each gene
num_cores <- detectCores() - 1  # Use all but one core
cl <- makeCluster(num_cores)
clusterExport(cl, c("fit_gene_model", "gene_data_list", "time_points"))
clusterEvalQ(cl, {
  library(rstanarm)
  library(data.table)
})
```

```
  gene_results <- parLapply(cl, gene_data_list, function(x) {
    fit_gene_model(x$gene_data, x$gene_name, time_points)
  })

  stopCluster(cl)

  # Remove NULL results and combine
  gene_results <- gene_results[!sapply(gene_results, is.null)]
  all_gene_results <- rbindlist(gene_results)

  # Identify significant genes
  significant_genes <- all_gene_results[Time_Coefficient_CI_Lower > 0 |
      ↪ Time_Coefficient_CI_Upper < 0]

  # Save all gene results
  fwrite(all_gene_results, file.path(output_dir, paste0(cell_type, "
      ↪ _All_Genes_Bayesian_Inference.csv")))

  # Save significant genes
  if (nrow(significant_genes) > 0) {
    fwrite(significant_genes, file.path(output_dir, paste0(cell_type, "
        ↪ _Significant_Genes_Bayesian_Inference.csv")))
    print(paste("Number of significant genes found:", nrow(significant_genes)))
  } else {
    print("No significant genes found.")
  }

  # Plot distribution of time coefficients
  p <- ggplot(all_gene_results, aes(x = Time_Coefficient_Mean)) +
    geom_histogram(bins = 50, fill = "skyblue", color = "black") +
    geom_vline(xintercept = 0, linetype = "dashed", color = "red") +
    labs(title = paste0("Distribution of Time Coefficients - ", cell_type),
         x = "Time Coefficient", y = "Count")

  # Save the plot
  ggsave(file.path(output_dir, paste0(cell_type, "
      ↪ _Time_Coefficients_Distribution.png")), p)

  print("Analysis completed successfully.")
  print(paste("Number of genes analysed:", nrow(all_gene_results)))
}

# Combine datasets by cell type, only accepting columns present in both datasets
combined_data_EC <- bind_rows(EC_KO_data, EC_WT_data) %>% select(symbol,
    ↪ everything())
combined_data_FAP <- bind_rows(FAP_KO_data, FAP_WT_data) %>% select(symbol,
    ↪ everything())
combined_data_MP <- bind_rows(MP_KO_data, MP_WT_data) %>% select(symbol,
    ↪ everything())

# Create a list assigning cell type to each dataset
cell_type_list <- list(EC = combined_data_EC, FAP = combined_data_FAP, MP =
    ↪ combined_data_MP)

# Loop through each cell type, running the dataset through
    ↪ perform_bayesian_inference
for (cell_type in names(cell_type_list)) {
  data_subset <- cell_type_list[[cell_type]]
  perform_bayesian_inference(data_subset, cell_type, vis_dir)
```

```
}

########################
### DESeq2 Analysis ###
########################

# Generate the metadata files for each cell type
generate_combined_metadata <- function(combined_data) {
  metadata <- data.frame(Sample = colnames(combined_data)[-1])  # Exclude the '
      ↪ symbol' column
  metadata <- metadata %>%
    mutate(
      Days_From_Injury = as.numeric(str_extract(Sample, "\\d+(?=d)")),
      Sample_Number = as.numeric(str_extract(Sample, "(?<=-)[0-9]+")),
      # Knockout status is the last two characters after "_"
      Knockout_Status = str_extract(Sample, "(?<=_)[A-Z]{2}$"),
    )
  return(metadata)
}

# Read the combined datasets for each cell type
combined_EC <- read_csv(file.path(combined_dataset_directory, "EC_combined.csv")
    ↪ )
combined_FAP <- read_csv(file.path(combined_dataset_directory, "FAP_combined.csv
    ↪ "))
combined_MP <- read_csv(file.path(combined_dataset_directory, "MP_combined.csv")
    ↪ )


# Generate metadata for each cell type
metadata_EC <- generate_combined_metadata(combined_EC)
metadata_FAP <- generate_combined_metadata(combined_FAP)
metadata_MP <- generate_combined_metadata(combined_MP)

# Make sure the symbol column information is saved, then drop the symbol column,
    ↪  then set the
# symbol information as rownames
symbols_EC <- combined_EC$symbol
symbols_FAP <- combined_FAP$symbol
symbols_MP <- combined_MP$symbol

combined_EC <- combined_EC %>% select(-symbol)
combined_FAP <- combined_FAP %>% select(-symbol)
combined_MP <- combined_MP %>% select(-symbol)

rownames(combined_EC) <- symbols_EC
rownames(combined_FAP) <- symbols_FAP
rownames(combined_MP) <- symbols_MP


# Save each metadata file
write_csv(metadata_EC, file = file.path(combined_dataset_directory, "
    ↪ metadata_combined_EC.csv"))
write_csv(metadata_FAP, file = file.path(combined_dataset_directory, "
    ↪ metadata_combined_FAP.csv"))
write_csv(metadata_MP, file = file.path(combined_dataset_directory, "
    ↪ metadata_combined_MP.csv"))

# Create the DESeq2 datasets for each cell type
```

```
dds_EC <- DESeqDataSetFromMatrix(countData = as.matrix(combined_EC),
                                 colData = metadata_EC,
                                 design = ~ Knockout_Status + Days_From_Injury +
                                     ↪  Knockout_Status:Days_From_Injury)

dds_FAP <- DESeqDataSetFromMatrix(countData = as.matrix(combined_FAP),
                                  colData = metadata_FAP,
                                  design = ~ Knockout_Status + Days_From_Injury
                                      ↪ + Knockout_Status:Days_From_Injury)

dds_MP <- DESeqDataSetFromMatrix(countData = as.matrix(combined_MP),
                                 colData = metadata_MP,
                                 design = ~ Knockout_Status + Days_From_Injury +
                                     ↪  Knockout_Status:Days_From_Injury)

# Perform DESeq2 normalisation
dds_EC <- DESeq(dds_EC)
dds_FAP <- DESeq(dds_FAP)
dds_MP <- DESeq(dds_MP)

# Generate the results dataframe for each cell type
results_EC <- results(dds_EC)
results_FAP <- results(dds_FAP)
results_MP <- results(dds_MP)

# Convert the results to a data frame
results_EC <- as.data.frame(results_EC)
results_FAP <- as.data.frame(results_FAP)
results_MP <- as.data.frame(results_MP)

# Make sure the symbol column is in the results
results_EC$symbol <- rownames(results_EC)
results_FAP$symbol <- rownames(results_FAP)
results_MP$symbol <- rownames(results_MP)

# Make sure the symbol column is in the first column
results_EC <- results_EC[, c(ncol(results_EC), 1:(ncol(results_EC) - 1))]
results_FAP <- results_FAP[, c(ncol(results_FAP), 1:(ncol(results_FAP) - 1))]
results_MP <- results_MP[, c(ncol(results_MP), 1:(ncol(results_MP) - 1))]

# Save the results of the DESeq2 normalisation
write_csv(results_EC, file = file.path(combined_dataset_directory, "
    ↪ DESeq2_Results_EC.csv"))
write_csv(results_FAP, file = file.path(combined_dataset_directory, "
    ↪ DESeq2_Results_FAP.csv"))
write_csv(results_MP, file = file.path(combined_dataset_directory, "
    ↪ DESeq2_Results_MP.csv"))


# Reading in the DESeq2 results data for each cell type
results_EC <- read.csv(file.path(combined_dataset_directory, "DESeq2_Results_EC.
    ↪ csv"))
results_FAP <- read.csv(file.path(combined_dataset_directory, "
    ↪ DESeq2_Results_FAP.csv"))
results_MP <- read.csv(file.path(combined_dataset_directory, "DESeq2_Results_MP.
    ↪ csv"))

# Create a volcano plot for each cell type
volcano_plot <- function(results, cell_type) {
```

```
  v_plot <- ggplot(results, aes(x = log2FoldChange, y = -log10(padj))) +
    geom_point(aes(color = padj < 0.05), alpha = 0.6) +
    scale_color_manual(values = c("FALSE" = "gray", "TRUE" = "red")) +
    geom_vline(xintercept = c(-1,1), linetype = "dashed") +
    geom_hline(yintercept = -log10(0.05), linetype = "dashed") +
    # Manually set the x scale
    scale_x_continuous(limits = c(-3, 3)) +
    # Manually set the y scale
    scale_y_continuous(limits = c(0, 5)) +
    labs(
      title = paste0("Volcano Plot for ", cell_type),
      x = "log2 Fold Change",
      y = "-log10 Adjusted P-value",
      color = "Significant"
    )
}

# Generate the volcano plots for each cell type
volcano_plot_EC <- volcano_plot(results_EC, "EC")
volcano_plot_FAP <- volcano_plot(results_FAP, "FAP")
volcano_plot_MP <- volcano_plot(results_MP, "MP")

# Save each volcano plot
ggsave(file = file.path(vis_dir, "DESeq2_Volcano_Plot_EC.png"), plot =
    ↪ volcano_plot_EC)
ggsave(file = file.path(vis_dir, "DESeq2_Volcano_Plot_FAP.png"), plot =
    ↪ volcano_plot_FAP)
ggsave(file = file.path(vis_dir, "DESeq2_Volcano_Plot_MP.png"), plot =
    ↪ volcano_plot_MP)

# Read in the combined datasets by cell type
combined_dataset_directory <- "/home/henry-cao/Desktop/KCL/Individual_Project/
    ↪ Data/Groppa_Materials/Combined_Datasets"
combined_EC <- read_csv(file.path(combined_dataset_directory, "EC_combined.csv")
    ↪ )
combined_FAP <- read_csv(file.path(combined_dataset_directory, "FAP_combined.csv
    ↪ "))
combined_MP <- read_csv(file.path(combined_dataset_directory, "MP_combined.csv")
    ↪ )

# # Ensure that each dataset is a dataframe
# combined_EC <- as.data.frame(combined_EC)
# combined_FAP <- as.data.frame(combined_FAP)
# combined_MP <- as.data.frame(combined_MP)

# Define the output directory
output_dir <- "/home/henry-cao/Desktop/KCL/Individual_Project/Data/
    ↪ Groppa_Materials/Combined_Datasets/DESeq2_No_Interaction"

# Generate the metadata files for each cell type
generate_combined_metadata <- function(combined_data) {
  metadata <- data.frame(Sample = colnames(combined_data)[-1])  # Exclude the '
      ↪ symbol' column
  metadata <- metadata %>%
    mutate(
      Days_From_Injury = as.numeric(str_extract(Sample, "\\d+(?=d)")),
      Sample_Number = as.numeric(str_extract(Sample, "(?<=-)[0-9]+")),
      # Knockout status is the last two characters after "_"
      Knockout_Status = str_extract(Sample, "(?<=_)[A-Z]{2}$"),
```

```
    )
  return(metadata)
}

# Generate metadata for each cell type
metadata_EC <- generate_combined_metadata(combined_EC)
metadata_FAP <- generate_combined_metadata(combined_FAP)
metadata_MP <- generate_combined_metadata(combined_MP)

# Make sure the symbol column information is saved, then drop the symbol column,
    ↪  then set the
# symbol information as rownames
symbols_EC <- combined_EC$symbol
symbols_FAP <- combined_FAP$symbol
symbols_MP <- combined_MP$symbol

# combined_EC <- combined_EC %>% select(-symbol)
# combined_FAP <- combined_FAP %>% select(-symbol)
# combined_MP <- combined_MP %>% select(-symbol)

# Drop the symbol column for each dataset without using select()
combined_EC <- combined_EC[, -1]
combined_FAP <- combined_FAP[, -1]
combined_MP <- combined_MP[, -1]

rownames(combined_EC) <- symbols_EC
rownames(combined_FAP) <- symbols_FAP
rownames(combined_MP) <- symbols_MP


# Save each metadata file
write_csv(metadata_EC, file = file.path(output_dir, "metadata_combined_EC.csv"))
write_csv(metadata_FAP, file = file.path(output_dir, "metadata_combined_FAP.csv
    ↪ "))
write_csv(metadata_MP, file = file.path(output_dir, "metadata_combined_MP.csv"))

# Create the DESeq2 datasets for each cell type
dds_EC <- DESeqDataSetFromMatrix(countData = as.matrix(combined_EC),
                                 colData = metadata_EC,
                                 design = ~ Knockout_Status + Days_From_Injury)

dds_FAP <- DESeqDataSetFromMatrix(countData = as.matrix(combined_FAP),
                                  colData = metadata_FAP,
                                  design = ~ Knockout_Status + Days_From_Injury)

dds_MP <- DESeqDataSetFromMatrix(countData = as.matrix(combined_MP),
                                 colData = metadata_MP,
                                 design = ~ Knockout_Status + Days_From_Injury)

# Perform DESeq2 normalisation
dds_EC <- DESeq(dds_EC)
dds_FAP <- DESeq(dds_FAP)
dds_MP <- DESeq(dds_MP)

# Generate the results dataframe for each cell type
results_EC <- results(dds_EC)
results_FAP <- results(dds_FAP)
results_MP <- results(dds_MP)
```

```
# Convert the results to a data frame
results_EC <- as.data.frame(results_EC)
results_FAP <- as.data.frame(results_FAP)
results_MP <- as.data.frame(results_MP)

# Make sure the symbol column is in the results
results_EC$symbol <- rownames(results_EC)
results_FAP$symbol <- rownames(results_FAP)
results_MP$symbol <- rownames(results_MP)

# Make sure the symbol column is in the first column
results_EC <- results_EC[, c(ncol(results_EC), 1:(ncol(results_EC) - 1))]
results_FAP <- results_FAP[, c(ncol(results_FAP), 1:(ncol(results_FAP) - 1))]
results_MP <- results_MP[, c(ncol(results_MP), 1:(ncol(results_MP) - 1))]

# Print out how many significant genes there are as defined by padj < 0.05
print(paste("Number of significant genes in EC:", length(which(results_EC$padj <
    ↪  0.05)), " out of ", nrow(results_EC)))
print(paste("Number of significant genes in FAP:", length(which(results_FAP$padj
    ↪  < 0.05)), " out of ", nrow(results_FAP)))
print(paste("Number of significant genes in MP:", length(which(results_MP$padj <
    ↪  0.05)), " out of ", nrow(results_MP)))


# Save the results of the DESeq2 normalisation
write_csv(results_EC, file = file.path(output_dir, "DESeq2_Results_EC.csv"))
write_csv(results_FAP, file = file.path(output_dir, "DESeq2_Results_FAP.csv"))
write_csv(results_MP, file = file.path(output_dir, "DESeq2_Results_MP.csv"))

# Save the significant results of the DESeq2 Normalisation, making sure they're
# sorted in ascending order by padj and with padj < 0.05
significant_result_EC <- results_EC[which(results_EC$padj < 0.05),]
significant_result_FAP <- results_FAP[which(results_FAP$padj < 0.05),]
significant_result_MP <- results_MP[which(results_MP$padj < 0.05),]
# Sort by padj wrhen writing
write_csv(significant_result_EC[order(significant_result_EC$padj),], file = file
    ↪ .path(output_dir, "DESeq2_Significant_Results_EC.csv"))
write_csv(significant_result_FAP[order(significant_result_FAP$padj),], file =
    ↪ file.path(output_dir, "DESeq2_Significant_Results_FAP.csv"))
write_csv(significant_result_MP[order(significant_result_MP$padj),], file = file
    ↪ .path(output_dir, "DESeq2_Significant_Results_MP.csv"))

# Reading in the DESeq2 results data for each cell type
results_EC <- read.csv(file.path(output_dir, "DESeq2_Results_EC.csv"))
results_FAP <- read.csv(file.path(output_dir, "DESeq2_Results_FAP.csv"))
results_MP <- read.csv(file.path(output_dir, "DESeq2_Results_MP.csv"))

# Create a volcano plot for each cell type
volcano_plot <- function(results, cell_type) {
  v_plot <- ggplot(results, aes(x = log2FoldChange, y = -log10(padj))) +
    geom_point(aes(color = padj < 0.05), alpha = 0.6) +
    scale_color_manual(values = c("FALSE" = "gray", "TRUE" = "red")) +
    geom_vline(xintercept = c(-1,1), linetype = "dashed") +
    geom_hline(yintercept = -log10(0.05), linetype = "dashed") +
    # Have automatic x scale
    scale_x_continuous() +
    # Have automatic y scale
    scale_y_continuous() +
    labs(
```

```
      title = paste0("Volcano Plot for ", cell_type),
      x = "log2 Fold Change",
      y = "-log10 Adjusted P-value",
      color = "Significant"
    )
}

# Generate the volcano plots for each cell type
volcano_plot_EC <- volcano_plot(results_EC, "EC")
volcano_plot_FAP <- volcano_plot(results_FAP, "FAP")
volcano_plot_MP <- volcano_plot(results_MP, "MP")

# Save each volcano plot
ggsave(file = file.path(output_dir, "DESeq2_Volcano_Plot_EC.png"), plot =
    ↪ volcano_plot_EC)
ggsave(file = file.path(output_dir, "DESeq2_Volcano_Plot_FAP.png"), plot =
    ↪ volcano_plot_FAP)
ggsave(file = file.path(output_dir, "DESeq2_Volcano_Plot_MP.png"), plot =
    ↪ volcano_plot_MP)

######################################################################
### Creating Venn Diagrams for Significant Genes by Cell Type ###
######################################################################

# Read in the DESeq2 results for each cell type
results_DESeq2_EC <- read.csv(file.path(combined_dataset_directory, "
    ↪ DESeq2_Results_EC.csv"))
results_DESeq2_FAP <- read.csv(file.path(combined_dataset_directory, "
    ↪ DESeq2_Results_FAP.csv"))
results_DESeq2_MP <- read.csv(file.path(combined_dataset_directory, "
    ↪ DESeq2_Results_MP.csv"))

# Read in the Limma Voom results
results_limma_voom_FAP_Day_3 <- read.csv(file.path(combined_dataset_directory, "
    ↪ Limma_Voom_FAP_3_Days_Atlas_Genes.csv"))
results_limma_voom_FAP_Day_10 <- read.csv(file.path(combined_dataset_directory,
    ↪ "Limma_Voom_FAP_10_Days_Atlas_Genes.csv"))

# Read in the Bayesian Inference Results
results_bayesian_EC <- read.csv(file.path(combined_dataset_directory, "
    ↪ EC_Significant_Genes_Bayesian_Inference.csv"))
results_bayesian_FAP <- read.csv(file.path(combined_dataset_directory, "
    ↪ FAP_Significant_Genes_Bayesian_Inference.csv"))
results_bayesian_MP <- read.csv(file.path(combined_dataset_directory, "
    ↪ MP_Significant_Genes_Bayesian_Inference.csv"))

# Extract the significant genes for each cell type and method
significant_genes_DEseq2_EC <- results_DESeq2_EC$symbol
significant_genes_DESeq2_FAP <- results_DESeq2_FAP$symbol
significant_genes_DESeq2_MP <- results_DESeq2_MP$symbol

significant_genes_limma_voom_FAP_Day_3 <- results_limma_voom_FAP_Day_3$symbol
significant_genes_limma_voom_FAP_Day_10 <- results_limma_voom_FAP_Day_10$symbol

significant_genes_bayesian_EC <- results_bayesian_EC$Gene
significant_genes_bayesian_FAP <- results_bayesian_FAP$Gene
significant_genes_bayesian_MP <- results_bayesian_MP$Gene

# Create a list of gene sets by cell type
```

```
gene_sets_EC <- list(
  DESeq2 = significant_genes_DEseq2_EC,
  Bayesian = significant_genes_bayesian_EC
)

gene_sets_FAP <- list(
  DESeq2 = significant_genes_DESeq2_FAP,
  Limma_Voom_Day_3 = significant_genes_limma_voom_FAP_Day_3,
  Limma_Voom_Day_10 = significant_genes_limma_voom_FAP_Day_10,
  Bayesian = significant_genes_bayesian_FAP
)

gene_sets_MP <- list(
  DESeq2 = significant_genes_DESeq2_MP,
  Bayesian = significant_genes_bayesian_MP
)

# Function to get overlapping genes and save as CSV
get_overlapping_genes <- function(gene_sets, cell_type, output_directory) {
  # Get all unique genes
  all_genes <- unique(unlist(gene_sets))

  # Create a data frame with gene presence in each set
  gene_presence <- data.frame(Gene = all_genes)
  for (set_name in names(gene_sets)) {
    gene_presence[[set_name]] <- all_genes %in% gene_sets[[set_name]]
  }

  # Function to get genes present in specific combination of sets
  get_genes_in_sets <- function(set_combination) {
    gene_presence %>%
      filter(if_all(set_combination, ~.)) %>%
      filter(if_all(setdiff(names(gene_sets), set_combination), ~!.)) %>%
      pull(Gene)
  }

  # Get all possible combinations of sets
  set_combinations <- lapply(1:length(gene_sets), function(n) combn(names(
      ↪ gene_sets), n, simplify = FALSE)) %>% unlist(recursive = FALSE)

  # Get overlapping genes for each combination
  overlap_results <- lapply(set_combinations, function(combo) {
    genes <- get_genes_in_sets(combo)
    data.frame(
      Combination = paste(combo, collapse = " & "),
      Count = length(genes),
      Genes = paste(genes, collapse = ", ")
    )
  }) %>% bind_rows()

  # Save results as CSV
  write.csv(overlap_results, file.path(output_directory, paste0(cell_type, "
      ↪ _overlapping_genes.csv")), row.names = FALSE)

  return(overlap_results)
}

venn_diagram <- function(gene_sets, cell_type, output_directory) {
  # Define a color palette
```

```
color_palette <- c("#E41A1C", "#377EB8", "#4DAF4A", "#984EA3", "#FF7F00", "#
    ↪ FFFF33")

# Adjust the number of colors based on the number of sets
num_sets <- length(gene_sets)
colors <- color_palette[1:num_sets]

# Create semi-transparent colors for fill
fill_colors <- sapply(colors, function(x) adjustcolor(x, alpha.f = 0.3))

# Determine the longest label length
max_label_length <- max(nchar(names(gene_sets)))

# Adjust width based on number of sets and longest label
width <- 3000 + (num_sets * 200) + (max_label_length * 20)

# Adjust label positions based on number of sets
if (num_sets == 2) {
  cat.pos <- c(-30, 30)
  cat.dist <- c(0.05, 0.05)
} else if (num_sets == 3) {
  cat.pos <- c(0, 60, -60)
  cat.dist <- c(0.05, 0.05, 0.05)
} else {
  cat.pos <- c(0, 0, 180, 180)
  cat.dist <- c(0.05, 0.05, 0.05, 0.05)
}

# Create the Venn diagram
venn_plot <- venn.diagram(
  x = gene_sets,
  category.names = names(gene_sets),
  filename = NULL,
  output = TRUE,
  imagetype = "png",
  height = 3000,
  width = width,
  resolution = 300,
  compression = "lzw",
  lwd = 2,
  col = colors,
  fill = fill_colors,
  cex = 1.5,
  fontfamily = "sans",
  cat.cex = 1.2,
  cat.fontfamily = "sans",
  cat.pos = cat.pos,
  cat.dist = cat.dist,
  euler.d = TRUE,
  scaled = TRUE
)

# Save the plot
png(file = file.path(output_directory, paste0("Venn_Diagram_", cell_type, ".
    ↪ png")),
    width = width, height = 3000, res = 300)
grid.draw(venn_plot)
dev.off()
```

```
  # Get and save overlapping genes
  get_overlapping_genes(gene_sets, cell_type, output_directory)
}

# Create Venn Diagrams for each cell type
venn_diagram(gene_sets_EC, "EC", vis_dir)
venn_diagram(gene_sets_FAP, "FAP", vis_dir)
venn_diagram(gene_sets_MP, "MP", vis_dir)

###################
### Time Plots ###
###################

generate_significant_gene_time_plots <- function(data_subset, significant_genes,
    ↪  cell_type, output_dir) {
  print(paste("Generating time plots for significant genes in cell type:",
      ↪ cell_type))

  # Ensure significant_genes is a character vector
  significant_genes <- as.character(significant_genes)

  # Filter data_subset to include only significant genes
  significant_data <- data_subset %>%
    filter(symbol %in% significant_genes)

  # Reshape the data
  long_data <- significant_data %>%
    pivot_longer(cols = -symbol, names_to = "Sample", values_to = "Expression")
        ↪ %>%
    mutate(
      # Time comes after "X", but before the "d"
      Time = as.numeric(str_extract(Sample, "(?<=X)\\d+(?=d)")),
      # SampleID comes after the "., but before the "_"
      SampleID = as.numeric(str_extract(Sample, "(?<=\\.)(\\d+)(?=_)")),
      Treatment = str_extract(Sample, "(?<=_)\\w+$")
    )

  # Print diagnostic information
  print("Sample of reshaped data:")
  print(head(long_data, 20))

  # Create a directory for the plots
  plot_dir <- file.path(output_dir, paste0(cell_type, "
      ↪ _Significant_Gene_Time_Plots"))
  dir.create(plot_dir, showWarnings = FALSE, recursive = TRUE)

  # Generate a plot for each significant gene
  for (gene in significant_genes) {
    gene_data <- long_data %>% filter(symbol == gene)

    if(nrow(gene_data) > 0) {
      print(paste("Generating plot for gene:", gene))

      p <- ggplot(gene_data, aes(x = Time, y = log10(Expression), color =
          ↪ Treatment)) +
        geom_point(aes(shape = factor(SampleID)), size = 3) +
        geom_line(aes(group = interaction(Treatment, SampleID))) +
        scale_x_continuous(breaks = sort(unique(gene_data$Time))) +
        scale_y_continuous(
```

```
          labels = function(x) format(10^x, scientific = FALSE),
          breaks = log10(c(1, 10, 100, 1000, 10000))
        ) +
        labs(title = paste("Expression of", gene, "over time"),
             subtitle = paste("Cell type:", cell_type),
             x = "Days from injury",
             y = "Expression level") +
        theme(legend.position = "bottom")

      # Save the plot
      plot_path <- file.path(plot_dir, paste0(gene, "_time_plot.png"))
      ggsave(filename = plot_path, plot = p, width = 12, height = 8)
      print(paste("Plot saved to:", plot_path))
    } else {
      warning(paste("No data found for gene", gene, "in cell type", cell_type))
    }
  }

  print(paste("Time plots generated for", length(significant_genes), "
      ↪ significant genes in", cell_type))
}

# Read in the combined datasets
combined_EC <- read.csv("/home/henry-cao/Desktop/KCL/Individual_Project/Data/
    ↪ Groppa_Materials/Combined_Datasets/EC_combined.csv")
combined_FAP <- read.csv("/home/henry-cao/Desktop/KCL/Individual_Project/Data/
    ↪ Groppa_Materials/Combined_Datasets/FAP_combined.csv")
combined_MP <- read.csv("/home/henry-cao/Desktop/KCL/Individual_Project/Data/
    ↪ Groppa_Materials/Combined_Datasets/MP_combined.csv")

# Read in the significant genes datasets
significant_genes_DESeq2_EC <- read.csv("/home/henry-cao/Desktop/KCL/
    ↪ Individual_Project/Data/Groppa_Materials/Final_Significant_Genes/EC/
    ↪ DESeq2_EC_Atlas_Genes.csv")
significant_genes_DESeq2_FAP <- read.csv("/home/henry-cao/Desktop/KCL/
    ↪ Individual_Project/Data/Groppa_Materials/Final_Significant_Genes/FAP/
    ↪ DESeq2_FAP_Atlas_Genes.csv")
significant_genes_DESeq2_MP <- read.csv("/home/henry-cao/Desktop/KCL/
    ↪ Individual_Project/Data/Groppa_Materials/Final_Significant_Genes/MP/
    ↪ DESeq2_MP_Atlas_Genes.csv")

significant_genes_limma_voom_FAP_Day_3 <- read.csv("/home/henry-cao/Desktop/KCL/
    ↪ Individual_Project/Data/Groppa_Materials/Final_Significant_Genes/FAP/
    ↪ Limma_Voom_FAP_3_Days_Atlas_Genes.csv")
significant_genes_limma_voom_FAP_Day_10 <- read.csv("/home/henry-cao/Desktop/KCL
    ↪ /Individual_Project/Data/Groppa_Materials/Final_Significant_Genes/FAP/
    ↪ Limma_Voom_FAP_10_Days_Atlas_Genes.csv")

significant_genes_bayesian_EC <- read.csv("/home/henry-cao/Desktop/KCL/
    ↪ Individual_Project/Data/Groppa_Materials/Final_Significant_Genes/EC/
    ↪ Bayes_EC_Atlas_Genes.csv")
significant_genes_bayesian_FAP <- read.csv("/home/henry-cao/Desktop/KCL/
    ↪ Individual_Project/Data/Groppa_Materials/Final_Significant_Genes/FAP/
    ↪ Bayes_FAP_Atlas_Genes.csv")
significant_genes_bayesian_MP <- read.csv("/home/henry-cao/Desktop/KCL/
    ↪ Individual_Project/Data/Groppa_Materials/Final_Significant_Genes/MP/
    ↪ Bayes_MP_Atlas_Genes.csv")

# Define output directories for each set of genes
```

```
output_dir_DESeq2_EC <- "/home/henry-cao/Desktop/KCL/Individual_Project/Data/
    ↪ Groppa_Materials/Final_Visuals/Timeplots/EC/DESeq2"
output_dir_DESeq2_FAP <- "/home/henry-cao/Desktop/KCL/Individual_Project/Data/
    ↪ Groppa_Materials/Final_Visuals/Timeplots/FAP/DESeq2"
output_dir_DESeq2_MP <- "/home/henry-cao/Desktop/KCL/Individual_Project/Data/
    ↪ Groppa_Materials/Final_Visuals/Timeplots/MP/DESeq2"

output_dir_limma_voom_FAP_Day_3 <- "/home/henry-cao/Desktop/KCL/
    ↪ Individual_Project/Data/Groppa_Materials/Final_Visuals/Timeplots/FAP/
    ↪ Limma_Voom"
output_dir_limma_voom_FAP_Day_10 <- "/home/henry-cao/Desktop/KCL/
    ↪ Individual_Project/Data/Groppa_Materials/Final_Visuals/Timeplots/FAP/
    ↪ Limma_Voom"

output_dir_bayesian_EC <- "/home/henry-cao/Desktop/KCL/Individual_Project/Data/
    ↪ Groppa_Materials/Final_Visuals/Timeplots/EC/Bayesian_Inference"
output_dir_bayesian_FAP <- "/home/henry-cao/Desktop/KCL/Individual_Project/Data/
    ↪ Groppa_Materials/Final_Visuals/Timeplots/FAP/Bayesian_Inference"
output_dir_bayesian_MP <- "/home/henry-cao/Desktop/KCL/Individual_Project/Data/
    ↪ Groppa_Materials/Final_Visuals/Timeplots/MP/Bayesian_Inference"

# Generate time plots for each set of significant genes
generate_significant_gene_time_plots(combined_EC,
    ↪ significant_genes_DESeq2_EC$symbol, "EC", output_dir_DESeq2_EC)
generate_significant_gene_time_plots(combined_FAP,
    ↪ significant_genes_DESeq2_FAP$symbol, "FAP", output_dir_DESeq2_FAP)
generate_significant_gene_time_plots(combined_MP,
    ↪ significant_genes_DESeq2_MP$symbol, "MP", output_dir_DESeq2_MP)

generate_significant_gene_time_plots(combined_FAP,
    ↪ significant_genes_limma_voom_FAP_Day_3$symbol, "FAP_3_Days",
    ↪ output_dir_limma_voom_FAP_Day_3)
generate_significant_gene_time_plots(combined_FAP,
    ↪ significant_genes_limma_voom_FAP_Day_10$symbol, "FAP_10_Days",
    ↪ output_dir_limma_voom_FAP_Day_10)

generate_significant_gene_time_plots(combined_EC,
    ↪ significant_genes_bayesian_EC$Gene, "EC", output_dir_bayesian_EC)
generate_significant_gene_time_plots(combined_FAP,
    ↪ significant_genes_bayesian_FAP$Gene, "FAP", output_dir_bayesian_FAP)
generate_significant_gene_time_plots(combined_MP,
    ↪ significant_genes_bayesian_MP$Gene, "MP", output_dir_bayesian_MP)


#################
### Heatmaps ###
#################

# Read in the combined datasets by cell type
combined_dataset_directory <- "/home/henry-cao/Desktop/KCL/Individual_Project/
    ↪ Data/Groppa_Materials/Combined_Datasets"
combined_EC <- read_csv(file.path(combined_dataset_directory, "EC_combined.csv")
    ↪ )
combined_FAP <- read_csv(file.path(combined_dataset_directory, "FAP_combined.csv
    ↪ "))
combined_MP <- read_csv(file.path(combined_dataset_directory, "MP_combined.csv")
    ↪ )

# Generate the metadata files for each cell type
```

```
generate_combined_metadata <- function(combined_data) {
  metadata <- data.frame(Sample = colnames(combined_data)[-1])  # Exclude the '
      ↪ symbol' column
  metadata <- metadata %>%
    mutate(
      Days_From_Injury = as.numeric(str_extract(Sample, "\\d+(?=d)")),
      Sample_Number = as.numeric(str_extract(Sample, "(?<=-)[0-9]+")),
      # Knockout status is the last two characters after "_"
      Knockout_Status = str_extract(Sample, "(?<=_)[A-Z]{2}$"),
    )
  return(metadata)
}

# Generate metadata for each cell type
metadata_EC <- generate_combined_metadata(combined_EC)
metadata_FAP <- generate_combined_metadata(combined_FAP)
metadata_MP <- generate_combined_metadata(combined_MP)

# Make sure the symbol column information is saved, then drop the symbol column,
    ↪  then set the
# symbol information as rownames
symbols_EC <- combined_EC$symbol
symbols_FAP <- combined_FAP$symbol
symbols_MP <- combined_MP$symbol

combined_EC <- combined_EC %>% select(-symbol)
combined_FAP <- combined_FAP %>% select(-symbol)
combined_MP <- combined_MP %>% select(-symbol)

rownames(combined_EC) <- symbols_EC
rownames(combined_FAP) <- symbols_FAP
rownames(combined_MP) <- symbols_MP


# Save each metadata file
write_csv(metadata_EC, file = file.path(combined_dataset_directory, "
    ↪ metadata_combined_EC.csv"))
write_csv(metadata_FAP, file = file.path(combined_dataset_directory, "
    ↪ metadata_combined_FAP.csv"))
write_csv(metadata_MP, file = file.path(combined_dataset_directory, "
    ↪ metadata_combined_MP.csv"))

# Create the DESeq2 datasets for each cell type
dds_EC <- DESeqDataSetFromMatrix(countData = as.matrix(combined_EC),
                                 colData = metadata_EC,
                                 design = ~ Knockout_Status + Days_From_Injury +
                                     ↪  Knockout_Status:Days_From_Injury)

dds_FAP <- DESeqDataSetFromMatrix(countData = as.matrix(combined_FAP),
                                  colData = metadata_FAP,
                                  design = ~ Knockout_Status + Days_From_Injury
                                      ↪ + Knockout_Status:Days_From_Injury)

dds_MP <- DESeqDataSetFromMatrix(countData = as.matrix(combined_MP),
                                 colData = metadata_MP,
                                 design = ~ Knockout_Status + Days_From_Injury +
                                     ↪  Knockout_Status:Days_From_Injury)

create_top_var_heatmap <- function(dds, output_file, n_top_genes = 29,
```

```
                                    annotation_cols = c("Days_From_Injury", "
                                        ↪ Knockout_Status")) {
  # Ensure required libraries are loaded
  library(DESeq2)
  library(pheatmap)

  # Check if annotation_cols are present in colData(dds)
  missing_cols <- setdiff(annotation_cols, colnames(colData(dds)))
  if (length(missing_cols) > 0) {
    stop(paste("The following columns are missing from colData:", paste(
        ↪ missing_cols, collapse = ", ")))
  }

  # Perform variance stabilizing transformation
  vsd <- vst(dds, blind = FALSE)

  # Select top variable genes
  topVarGenes <- head(order(rowVars(assay(vsd)), decreasing = TRUE), n_top_genes
      ↪ )

  # Extract and center the matrix
  mat <- assay(vsd)[topVarGenes, ]
  mat <- mat - rowMeans(mat)

  # Create annotation data frame
  anno <- as.data.frame(colData(vsd)[, annotation_cols, drop = FALSE])

  # Open a PNG file device
  png(output_file, width = 20, height = 12, units = "in", res = 300)

  # Create the heatmap
  pheatmap(mat,
           annotation_col = anno,
           show_rownames = TRUE,
           show_colnames = FALSE,
           cluster_rows = TRUE,
           cluster_cols = TRUE,
           scale = "row",
           main = paste("Heatmap of Top", n_top_genes, "Variable Genes"))

  # Close the file device to save the plot
  dev.off()

  # Return the matrix and annotation for further use if needed
  return(list(matrix = mat, annotation = anno))
}

# Create a heatmap for each cell type
output_dir_heatmap <- "/home/henry-cao/Desktop/KCL/Individual_Project/Data/
    ↪ Final_Data/Groppa/Final_Visuals/Heatmaps"
create_top_var_heatmap(dds_EC, file.path(output_dir_heatmap, "EC_heatmap.png"))
create_top_var_heatmap(dds_FAP, file.path(output_dir_heatmap, "FAP_heatmap.png")
    ↪ )
create_top_var_heatmap(dds_MP, file.path(output_dir_heatmap, "MP_heatmap.png"))

############################
############################
### Limma-Voom Analysis ###
############################
```

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

# Function to create volcano plot with respect to cell type and days from injury
create_volcano_plot_by_cell_type_and_days_from_injury <- function(fit,
    ↪ coef_index, title, output_dir) {
  top_table <- topTable(fit, coef = coef_index, adjust.method = "BH", number =
      ↪ Inf)
  top_table$Significant <- top_table$adj.P.Val < 0.05
  top_table$symbol <- rownames(top_table)

  # Move symbol to the first column
  top_table <- top_table[, c(ncol(top_table), 1:(ncol(top_table) - 1))]

  # Order the table by adj.P.Val
  top_table <- top_table[order(top_table$adj.P.Val),]
  print(head(top_table))

  # Save the significant genes to a CSV file using the combined dataset
      ↪ directory with rownames
  write_csv(top_table, file = file.path(output_dir, paste0(title, ".csv")))

  volcano_plot <- ggplot(top_table, aes(x = logFC, y = -log10(adj.P.Val), color
      ↪ = Significant)) +
    geom_point(alpha = 0.4) +
    ggtitle(title) +
    xlab("Log2 Fold Change") +
    ylab("-Log10 P-Value") +
    scale_color_manual(values = c("black", "red")) +
    geom_vline(xintercept = c(-1, 1), col = "red", linetype = "dashed") +
    geom_hline(yintercept = -log10(0.05), col = "red", linetype = "dashed") +
    guides(color = FALSE)

  # Save the volcano plot
  ggsave(file.path(output_dir, paste0(title, ".png")), plot = volcano_plot, dpi
      ↪ = 300, width = 10, height = 6)

  # Return the plot
  return(volcano_plot)
}


# Preprocess and analyze each dataset for volcano plots
analyse_and_plot_by_cell_type_and_days_from_injury <- function(combined_data,
    ↪ dataset_name, time_points, output_dir) {
  symbols <- combined_data$symbol
  sample_csv <- combined_data[, -1]
  rownames(sample_csv) <- symbols

  # Create DGEList object
  d0 <- DGEList(sample_csv)
  d0 <- calcNormFactors(d0)
  cutoff <- 1
  drop <- which(apply(cpm(d0), 1, max) < cutoff)
  d <- d0[-drop, ]

  snames <- colnames(sample_csv)

  # Extract treatment and time
  treatment <- factor(str_extract(snames, "(?<=_)[A-Z]+"))
```

```
  time <- factor(str_extract(snames, "\\d+(?=d)"))

  group <- interaction(treatment, time)
  plotMDS(d, col = as.numeric(group))

  mm <- model.matrix(~0 + group)
  y <- voom(d, mm, plot = TRUE)

  fit <- lmFit(y, mm)


  for (time_point in time_points) {
    for (cell_type in levels(treatment)) {
      contrast_name <- paste0("group", cell_type, ".", time_point, "-groupWT.",
          ↪ time_point)
      contr <- makeContrasts(contrasts = contrast_name, levels = colnames(coef(
          ↪ fit)))
      tmp <- tryCatch({
        tmp <- contrasts.fit(fit, contr)
        tmp <- eBayes(tmp)
        plot_title <- paste0(dataset_name, ": ", cell_type, " vs WT at ",
            ↪ time_point, " days, adj.P.Val")
        volcano_plot <- create_volcano_plot_by_cell_type_and_days_from_injury(
            ↪ tmp, 1, plot_title, output_dir)
      }, error = function(e) {
        message(paste("Skipping", cell_type, "at time point", time_point, "due
            ↪ to error:", e$message))
        NULL
      })
    }
  }
}


# Load combined datasets
combined_data_EC <- read_csv(file.path(combined_dataset_directory, "EC_combined.
    ↪ csv"))
combined_data_FAP <- read_csv(file.path(combined_dataset_directory, "
    ↪ FAP_combined.csv"))
combined_data_MP <- read_csv(file.path(combined_dataset_directory, "MP_combined.
    ↪ csv"))

# Define time points for each cell type
time_points_EC <- c("0", "1", "2", "3", "5", "6", "7", "10", "14")
time_points_FAP <- c("0", "1", "2", "3", "5", "6", "7", "10", "14")
time_points_MP <- c("0", "1", "3", "4", "6", "10")

# Run the analysis and plot for each cell type dataset
analyse_and_plot_by_cell_type_and_days_from_injury(combined_data_EC, "EC",
    ↪ time_points_EC, vis_dir)
analyse_and_plot_by_cell_type_and_days_from_injury(combined_data_FAP, "FAP",
    ↪ time_points_FAP, vis_dir)
analyse_and_plot_by_cell_type_and_days_from_injury(combined_data_MP, "MP",
    ↪ time_points_MP, vis_dir)

####################
####################
### GO Analysis ###
####################
```

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

library(org.Mm.eg.db)
library(clusterProfiler)
library(ggplot2)
library(cowplot)

# Function to perform GO enrichment and create a plot
create_go_plot <- function(gene_list, title, ontology = "BP", organism = "mouse
    ↪ ") {
  # Set the appropriate organism database
  org_db <- if(organism == "mouse") org.Mm.eg.db else org.Hs.eg.db

  # Convert gene symbols to Entrez IDs
  entrez_ids <- bitr(gene_list, fromType = "SYMBOL", toType = "ENTREZID", OrgDb
      ↪ = org_db)

  # Perform GO enrichment analysis
  go_enrichment <- enrichGO(gene = entrez_ids$ENTREZID,
                            OrgDb = org_db,
                            ont = ontology,
                            pAdjustMethod = "BH",
                            pvalueCutoff = 0.10,
                            qvalueCutoff = 0.20)

  # print(head(go_enrichment@result, n = 20))

  # Create dot plot
  dot_plot <- dotplot(go_enrichment, showCategory = 20, title = title) +
    theme(axis.text.y = element_text(size = 8))

  return(list(plot = dot_plot, data = go_enrichment))
}

# Load data of DESeq2 with interaction
EC_Data <- read.csv("/home/henry-cao/Desktop/KCL/Individual_Project/Data/
    ↪ Final_Data/Groppa/Final_Significant_Genes/EC/DESeq2_EC_Atlas_Genes.csv")
FAP_Data <- read.csv("/home/henry-cao/Desktop/KCL/Individual_Project/Data/
    ↪ Final_Data/Groppa/Final_Significant_Genes/FAP/DESeq2_FAP_Atlas_Genes.csv
    ↪ ")
MP_Data <- read.csv("/home/henry-cao/Desktop/KCL/Individual_Project/Data/
    ↪ Final_Data/Groppa/Final_Significant_Genes/MP/DESeq2_MP_Atlas_Genes.csv")

EC_genes <- EC_Data$symbol
FAP_genes <- FAP_Data$symbol
MP_genes <- MP_Data$symbol

# Create gene lists
gene_lists <- list(EC = EC_genes, FAP = FAP_genes, MP = MP_genes)

# Generate plots and save data
plots <- list()
for (cell_type in names(gene_lists)) {
  result <- create_go_plot(gene_lists[[cell_type]], cell_type)
  plots[[cell_type]] <- result$plot

  # Save individual plot
  ggsave(file.path(output_directory, paste0("DESeq_2_With_Interaction/",
      ↪ cell_type, "_GO_plot.pdf")), result$plot, width = 10, height = 8)
```

```
  # Save data as CSV
  write.csv(as.data.frame(result$data), file = file.path(output_directory,
      ↪ paste0("DESeq_2_With_Interaction/", cell_type, "_DESeq2_GO_data.csv")),
      ↪  row.names = FALSE)
}

# Combine plots
combined_plot <- plot_grid(plotlist = plots, ncol = 1)

# Save combined plot to output directory
ggsave(file.path(output_directory, "DESeq_2_With_Interaction/combined_GO_plot.
    ↪ pdf"), combined_plot, width = 10, height = 8 * length(plots))

# Function to create a custom multi-panel plot
create_multi_panel_plot <- function(gene_lists, main_title) {
  plot_list <- lapply(names(gene_lists), function(name) {
    create_go_plot(gene_lists[[name]], name)$plot
  })

  combined_plot <- plot_grid(plotlist = plot_list, ncol = 2)
  title <- ggdraw() +
    draw_label(main_title, fontface = 'bold', x = 0, hjust = 0) +
    theme(plot.margin = margin(0, 0, 0, 7))

  plot_grid(title, combined_plot, ncol = 1, rel_heights = c(0.1, 1))
}

# Create and save multi-panel plot
multi_panel_plot <- create_multi_panel_plot(gene_lists, "Multi-panel GO
    ↪ Enrichment")
ggsave(file.path(output_directory, "multi_panel_GO_plot.pdf"), multi_panel_plot,
    ↪  width = 15, height = 20)

# Load data for DESeq2 No Interaction
EC_Data <- read.csv("/home/henry-cao/Desktop/KCL/Individual_Project/Data/
    ↪ Final_Data/Groppa/Final_Significant_Genes/EC/
    ↪ DESeq2_EC_No_Interaction_atlas_genes.csv")
FAP_Data <- read.csv("/home/henry-cao/Desktop/KCL/Individual_Project/Data/
    ↪ Final_Data/Groppa/Final_Significant_Genes/FAP/
    ↪ DESeq2_FAP_No_Interaction_atlas_genes.csv")
MP_Data <- read.csv("/home/henry-cao/Desktop/KCL/Individual_Project/Data/
    ↪ Final_Data/Groppa/Final_Significant_Genes/FAP/
    ↪ DESeq2_FAP_No_Interaction_atlas_genes.csv")

EC_genes <- EC_Data$symbol
FAP_genes <- FAP_Data$symbol
MP_genes <- MP_Data$symbol

# Create gene lists
gene_lists <- list(EC = EC_genes, FAP = FAP_genes, MP = MP_genes)

# Generate plots and save data
plots <- list()
for (cell_type in names(gene_lists)) {
  result <- create_go_plot(gene_lists[[cell_type]], cell_type)
  plots[[cell_type]] <- result$plot

  # Save individual plot
```

```
  ggsave(file.path(output_directory, paste0("DESeq_2_No_Interaction/", cell_type
      ↪ , "_GO_plot.pdf")), result$plot, width = 10, height = 8)

  # Save data as CSV
  write.csv(as.data.frame(result$data), file = file.path(output_directory,
      ↪ paste0("DESeq_2_No_Interaction/", cell_type, "_DESeq2_GO_data.csv")),
      ↪ row.names = FALSE)
}

# Combine plots
combined_plot <- plot_grid(plotlist = plots, ncol = 1)

# Save combined plot to output directory
ggsave(file.path(output_directory, "DESeq_2_No_Interaction/combined_GO_plot.pdf
    ↪ "), combined_plot, width = 10, height = 8 * length(plots))

# Function to create a custom multi-panel plot
create_multi_panel_plot <- function(gene_lists, main_title) {
  plot_list <- lapply(names(gene_lists), function(name) {
    create_go_plot(gene_lists[[name]], name)$plot
  })

  combined_plot <- plot_grid(plotlist = plot_list, ncol = 2)
  title <- ggdraw() +
    draw_label(main_title, fontface = 'bold', x = 0, hjust = 0) +
    theme(plot.margin = margin(0, 0, 0, 7))

  plot_grid(title, combined_plot, ncol = 1, rel_heights = c(0.1, 1))
}

# Create and save multi-panel plot
multi_panel_plot <- create_multi_panel_plot(gene_lists, "Multi-panel GO
    ↪ Enrichment")
ggsave(file.path(output_directory, "multi_panel_GO_plot.pdf"), multi_panel_plot,
    ↪  width = 15, height = 20)

# Read in another set of datasets which used limma-voom
FAP_Day_3 <- read.csv("/home/henry-cao/Desktop/KCL/Individual_Project/Data/
    ↪ Final_Data/Groppa/Final_Significant_Genes/FAP/
    ↪ Limma_Voom_FAP_3_Days_Atlas_Genes.csv")
FAP_Day_10 <- read.csv("/home/henry-cao/Desktop/KCL/Individual_Project/Data/
    ↪ Final_Data/Groppa/Final_Significant_Genes/FAP/
    ↪ Limma_Voom_FAP_10_Days_Atlas_Genes.csv")

# Only keep the samples where the column Significant = 'TRUE'
FAP_Day_3 <- FAP_Day_3[FAP_Day_3$Significant == 'TRUE', ]
FAP_Day_10 <- FAP_Day_10[FAP_Day_10$Significant == 'TRUE', ]

# Extract the gene symbols
FAP_Day_3_genes <- FAP_Day_3$symbol
FAP_Day_10_genes <- FAP_Day_10$symbol

# Create gene lists
gene_lists_limma <- list(FAP_Day_3 = FAP_Day_3_genes, FAP_Day_10 =
    ↪ FAP_Day_10_genes)

# Generate plots and save data
plots_limma <- list()
```

```
# Create new output directory
output_directory_limma <- file.path(output_directory, "Limma_Voom")

for (comparison in names(gene_lists_limma)) {
  result <- create_go_plot(gene_lists_limma[[comparison]], comparison)
  plots_limma[[comparison]] <- result$plot

  # Save individual plot
  ggsave(file.path(output_directory_limma, paste0(comparison, "_GO_plot_limma.
      ↪ pdf")), result$plot, width = 10, height = 8)

  # Save data as CSV
  write.csv(as.data.frame(result$data), file = file.path(output_directory_limma,
      ↪  paste0(comparison, "_GO_data_limma.csv")), row.names = FALSE)
}

# Create and save multi-panel plot for limma-voom results
multi_panel_plot_limma <- create_multi_panel_plot(gene_lists_limma, "Multi-panel
    ↪  GO Enrichment (Limma-Voom)")
ggsave(file.path(output_directory_limma, "multi_panel_GO_plot_limma.pdf"),
    ↪ multi_panel_plot_limma, width = 15, height = 20)

# Read in the Bayesian Inference results
EC_Bayesian <- read.csv("/home/henry-cao/Desktop/KCL/Individual_Project/Data/
    ↪ Final_Data/Groppa/Final_Significant_Genes/EC/Bayes_EC_Atlas_Genes.csv")
FAP_Bayesian <- read.csv("/home/henry-cao/Desktop/KCL/Individual_Project/Data/
    ↪ Final_Data/Groppa/Final_Significant_Genes/FAP/Bayes_FAP_Atlas_Genes.csv")
MP_Bayesian <- read.csv("/home/henry-cao/Desktop/KCL/Individual_Project/Data/
    ↪ Final_Data/Groppa/Final_Significant_Genes/MP/Bayes_MP_Atlas_Genes.csv")

# Extract the gene symbols
EC_Bayesian_genes <- EC_Bayesian$Gene
FAP_Bayesian_genes <- FAP_Bayesian$Gene
MP_Bayesian_genes <- MP_Bayesian$Gene

# Create gene lists
gene_lists_bayesian <- list(EC = EC_Bayesian_genes, FAP = FAP_Bayesian_genes, MP
    ↪  = MP_Bayesian_genes)

# Generate plots and save data
plots_bayesian <- list()
output_directory_bayesian <- file.path(output_directory, "Bayesian_Inference")
for (cell_type in names(gene_lists_bayesian)) {
  result <- create_go_plot(gene_lists_bayesian[[cell_type]], cell_type)

  # print(head(result$data@result, n = 20))

  plots_bayesian[[cell_type]] <- result$plot

  # Save individual plot
  ggsave(file.path(output_directory_bayesian, paste0(cell_type, "
      ↪ _GO_plot_bayesian.pdf")), result$plot, width = 10, height = 8)

  # Save data as CSV
  write.csv(as.data.frame(result$data), file = file.path(
      ↪ output_directory_bayesian, paste0(cell_type, "_GO_data_bayesian.csv")),
      ↪  row.names = FALSE)
}
```

```
# Create and save multi-panel plot for Bayesian Inference results
multi_panel_plot_bayesian <- create_multi_panel_plot(gene_lists_bayesian, "Multi
    ↪ -panel GO Enrichment (Bayesian Inference)")
ggsave(file.path(output_directory_bayesian, "multi_panel_GO_plot_bayesian.pdf"),
    ↪  multi_panel_plot_bayesian, width = 15, height = 20)
```