



GENOME TO PHENOME: INTEGRATED FRAMEWORK FOR GENE EXPRESSION IN *MUS MUSCULUS*

Data Cleaning & Data Management Project group 9



Introduction

The International Mouse Phenotyping Consortium (IMPC) is a global research initiative that aims to create a comprehensive functional catalog of the mammalian genome by generating and characterizing knockout mouse lines for every protein-coding gene (IMPC, n.d.). This is accomplished by employing a combination of experimental measurements, statistical analysis, and scoring methods to detect and quantify phenotypic changes between mice with targeted gene knockouts and controls (wild-type mice). Subsequently, this enhances our understanding of gene function and development while promoting the identification of new models for studying genetic disorders.

Mice are characterized at multiple research centers using a standardized phenotyping pipeline known as IMPReSS (International Mouse Phenotyping Resource of Standardized Screens) to assess a wide range of biological domains, such as behavior, metabolism, cardiovascular health, immune function, and fertility. This pipeline adheres to strict data quality standards and ensures the use of the minimum number of animals required to achieve statistical significance for each test (Hrabe de Angelis et al., 2015).

Results generated by IMPC involve both continuous (e.g. body weight) and qualitative (e.g. coat color abnormalities) phenotypic data along with metadata of the experimental design, involving gene and animal background information (e.g. gene symbols, mouse strain, age, etc.) and detailed descriptions of the phenotyping tests performed. Moreover, specialised imaging data are also collected as part of the protocols, including high-resolution 3D imaging of embryos, X rays of skeletal structures, echocardiograms and histology of various tissues.

The IMPC utilises linear mixed models to analyse continuous phenotypic data and identify significant deviations, ensuring that observed differences between mutants and controls are not attributable to random variation (Haselimashhadi et al., 2020). This process assesses the strength of the link between a gene knockout and observed phenotypic changes through statistical significance scores (p-values). To reduce the risk of false positives, p-values are corrected for multiple comparisons using techniques like the False Discovery Rate (FDR), and a standard threshold of <0.05 is applied to identify significant phenotypic associations. Fisher's exact test is used for the analysis of categorical data by constructing a contingency table to compare phenotypic frequencies between mutants and controls (Mallon et al., 2012).

Furthermore, mouse phenotypic results are systematically compared to human clinical data to identify shared characteristics. This process is facilitated by tools like PhenoDigm, which calculates similarity scores between the phenotypic traits of mouse models and the clinical features of human conditions. Besides, the main goal of IMPC is to extend these findings to human biology and reveal potential genetic contributors to diseases.

IMPC provides a web-based platform (<http://www.mousephenotype.org>) that serves as a comprehensive resource for displaying genotype–phenotype data related to knockout mouse lines. Designed primarily for the biomedical community, the portal enables users to conduct free-text queries and explore structured data through facets that combine multiple search terms.

In this context, we aim to develop a similar integrated and interactive data management solution for the IMPC collaborators, exploiting raw experimental data from the IMPC. This solution will include data cleaning, collation, database creation, and visualization tools, ensuring that phenotypic data from knockout mice are accurately and effectively analyzed, stored, and presented. By fulfilling this

aim, the project will facilitate the exploration of genotype-phenotype associations, enabling researchers to derive meaningful insights into gene function and their implications for human genetic disorders.

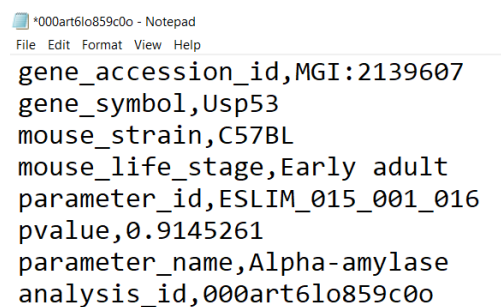
To accomplish this, the project will focus on a few key steps, starting with rigorous data cleaning to ensure accuracy and reliability, followed by organisation of the data into clear, meaningful categories. A scalable and normalised MySQL database will be created to store and manage the data efficiently, enabling easy querying and future expansion. Additionally, an intuitive RShiny dashboard will be developed to visualize gene-phenotype relationships, allowing researchers to explore data interactively and identify meaningful patterns. Finally, comprehensive documentation will be provided to ensure the system is accessible, easy to use, and well-suited for research needs.

Methods

Data collation

Raw experimental data, comprising 156,024 CSV files, were egressed from King's Trusted Research Environment (TRE) to a local machine and subsequently transferred to King's CREATE HPC facility to ensure accessibility for all group members. These files were retrieved using Secure File Transfer Protocol (SFTP) and included statistical outputs for individual genotype-phenotype pairs, along with additional files such as a Standard Operating Procedures (SOP) document, PhenoDigm scores linking human diseases to mouse phenotypes, and metadata files detailing tested phenotypes and procedures.

Each CSV file contained results for a single genotype-phenotype pair, with a unique filename referencing its respective analysis identifier (Figure 1). To merge the files, two approaches were employed—one using R and another using Python.

A screenshot of a Notepad window titled '*000art6lo859c0o - Notepad'. The window contains a single line of CSV data: 'gene_accession_id,MGI:2139607', 'gene_symbol,Usp53', 'mouse_strain,C57BL', 'mouse_life_stage,Early adult', 'parameter_id,ESLIM_015_001_016', 'pvalue,0.9145261', 'parameter_name,Alpha-amylase', 'analysis_id,000art6lo859c0o'.

```
*000art6lo859c0o - Notepad
File Edit Format View Help
gene_accession_id,MGI:2139607
gene_symbol,Usp53
mouse_strain,C57BL
mouse_life_stage,Early adult
parameter_id,ESLIM_015_001_016
pvalue,0.9145261
parameter_name,Alpha-amylase
analysis_id,000art6lo859c0o
```

Figure 1. Statistical analysis output of a tested genotype-phenotype pair as those stored in individual CSV files. Each CSV file contains a unique analysis Id; p-value score; a unique parameter identifier that corresponds to the phenotype tested along with the name of this parameter; and information on the knockout mice such as gene identifier, gene symbol, the development stage of the mice being tested and the mice strain.

In R (version 4.4.1), a script was created to process and merge the CSV files into a unified data frame. The script identified all CSV files within a specified directory and used the `fread()` function from the `data.table` package (version 1.16.4) to read each file. Each column was normalized by stripping whitespace, converting text to lowercase, and replacing missing values with NA.

```
# Function to process a single file
process_file = function(file) {

  # Read the CSV file
  data = fread(file, header = FALSE, col.names = c("key", "value"))

  # Clean and normalize the data by trimming white space and converting to lowercase
  data = data %>%
    mutate(
      key = tolower(trimws(as.character(key))),
      value = ifelse(!is.na(value) & trimws(value) != "",
                     tolower(trimws(as.character(value))),
                     NA)) # Missing or empty values replaced to NA
}
```

This normalisation of the data was deemed necessary as in the merged data frame we initially found 200 unique gene accession IDs but 388 gene symbols. This discrepancy was due to inconsistencies in the formatting of entries with case sensitive characters confusing the total count of unique gene symbols. Unique gene symbols were found to be 200 after implementing the aforementioned cleaning steps. Following this normalisation, the data are reshaped into a wide format using the `spread()` function from the `tidyr` package (version 1.3.1). In this format, each unique key from the data becomes a column header, with its corresponding value filling the appropriate cell in the data frame.

```
# Group everything by key and mutate key names. Then ungroup for further processing.
data = data %>%
  group_by(key) %>%
  mutate(key) %>%
  ungroup()

# Pivot key-value pairs into wide format
row_data = data %>%
  spread(key = key, value = value)

return(row_data)
}
```

To improve efficiency, the script employs parallel processing using the `parallel` package (version 4.4.1), enabling multiple files to be processed simultaneously across multiple CPU cores. A cluster of worker nodes is created to distribute the workload, and the `parLapply()` function applies the “process_file” function to each CSV file in the list. Once all files are processed, the resulting wide-format data frames are combined into a single cohesive data frame using `bind_rows()` from the `dplyr` package (version 1.1.4).

```
# Set up parallel processing
cl = makeCluster(detectCores() - 1) # Use all but one core
clusterExport(cl, varlist = c("fread", "trimws", "tolower", "process_file",
                              "spread", "mutate", "replace", "bind_rows", "group_by", "row_number"))
clusterEvalQ(cl, library(dplyr))
clusterEvalQ(cl, library(data.table))
clusterEvalQ(cl, library(tidyr))

# Process files in parallel
combined_data = parLapply(cl, csv_files, process_file)
stopCluster(cl)

# Combine all rows into one data frame
combined_data = bind_rows(combined_data)
```

If a file does not contain a particular key that exists in others, the corresponding values for that key are automatically filled with NA, ensuring that the structure of the combined dataset remains consistent. After combining the data, the script performs an optional post-processing step to handle specific columns such as “pvalue”, which is converted to a numeric data type to prepare it for further

analysis. Merging of the CSV files in R is feasible but even though the script utilises most available processing power, it is less efficient in terms of speed compared to applying a similar logic in python.

Data Exploration & Cleaning

Following merging of the CSV files into a single data frame, a thorough exploration and cleaning of the data was performed in R, focusing on validating critical variables based on the provided SOP file and summarising key attributes prior to subsequent analysis of the data. One of the central tasks was examining the “pvalue” column to ensure it comprised valid probabilities within the range of 0 to 1. 780 p-values were found to be out of range, and they were replaced by NA.

```
##{r}
# Calculate number of entries above 1.0 and below 0.0
out_of_range = sum(combined_data$pvalue < 0.0 | combined_data$pvalue > 1.0)
out_of_range
##[1] 780
```

```
##{r}
# Replace pvalues above 1 with NA(missing value).
clean_data = combined_data %>%
  mutate(pvalue = ifelse(pvalue > 1, NA, pvalue))
##
```

The dataset’s categorical variables “mouse life stage” and “mouse strain” were explored to validate their unique values and understand their distribution patterns. 11 distinct “mouse strain” values were found, of which only two corresponded to those presented by the SOP (C57BL, 129SV). After observing the distribution of these values, we found that 99% is shared between C57BL and 129SV and the rest 1% is almost evenly distributed between the other 9 strains (C50BL, C51BL, C52BL, C53BL, C54BL, C55BL, C56BL, C58BL and C59BL). More specifically, C57BL accounts for 96% (149,809) of the total values, with 129SV having the other 3.1% (4,929) and the rest presenting about 0.1% each (Figure 2). Therefore, we decided to ignore these 9 strains and not exclude them even though they do not correspond to the SOP, as we are not certain whether they are erroneous or not. Similar analysis for the unique mouse life stages indicated the same results as those stated in the SOP. Unique analysis Ids were also found to be 156,024 as expected.

To ensure clarity in parameter mappings, the relationship between “parameter Id” and “parameter name” was also examined. Cases where a single parameter name is associated with multiple Ids were identified and summarized. We observed 274 unique parameter Ids but only 232 parameter names, thus some parameter Ids correspond to the same parameter name multiple times, and we require additional variables to distinguish between those. Specifically, we observed one unique parameter name mapped to two unique parameter Ids 36 times while three unique parameter Ids

mouse_strain	Frequency	Percentage
129sv	4929	3.1591294
c50bl	148	0.0948572
c51bl	152	0.0974209
c52bl	146	0.0935753
c53bl	141	0.0903707
c54bl	143	0.0916526
c55bl	113	0.0724248
c56bl	160	0.1025483
c57bl	149809	96.0166385
c58bl	148	0.0948572
c59bl	135	0.0865251

Figure 2. Distinct mouse strains with their respective frequencies and percentages as those found in the dataset.

corresponded to one unique parameter name 3 times. This situation is caused by parameter ids originating from different experimental pipelines, even though the parameter name describes the same biological or phenotypic measurement.

The cleaning of the other three files, namely “parameter description”, “IMPC procedure” and “disease information”, was also approached in two different ways, but this time all scripts have been created in R. The primary issue that all these files had in common was the description field, where in-text commas were preventing proper parsing of the values to each respective column. Redundant columns, multiline rows, quotation marks and other formatting discrepancies such as html entities, multiple spaces and missing periods were also present in some instances.

The first approach was straight forward and focused on another common formatting feature shared between all three files. Every row was normally comprised of 4 columns for parameter description and disease information files, or 5 columns in the case of IMPC procedure file. All columns except for the description section had properly defined comma delimiters, allowing for easy parsing and further processing. One function per file has been defined to address file specific structure discrepancies, but they all followed the same general methodology. Each function starts by reading the file line by line and then if a row consists of more than 4 elements, the fields are separated one by one based on the consistent comma delimiters outside the description section, leaving this problematic column to be handled last.

In all cases, the first field (usually “name”) had to be cleaned by removing redundant row numbers at the beginning of each entry. This was achieved by exploiting a pattern in the first field that was common in all rows. The starting row numbers were separated by a space from the rest of the field and thus, it was easy to remove them by replacing this pattern (“^\\d+\\s+”) with nothing (“”).

```
# Process each line
processed_data <- lapply(csv_lines, function(line) {
  # Split the line by ","
  split_line <- strsplit(line, ",")[1]

  # If there are more than 4 elements, process accordingly
  if (length(split_line) > 4) {
    name <- trimws(split_line[1]) # First element
    isMandatory <- trimws(split_line[length(split_line) - 1]) # Second last element
    impcParameterOrigId <- trimws(split_line[length(split_line)]) # Last element
    # Remaining elements as description
    description <- trimws(paste(split_line[2:(length(split_line) - 2)], collapse = ","))
  }
  else if (length(split_line) == 4) {
    name <- trimws(split_line[1])
    description <- trimws(split_line[2])
    isMandatory <- trimws(split_line[3])
    impcParameterOrigId <- trimws(split_line[4])
  }
  else {
    print(split_line) # To catch incorrectly formatted lines
  }

  # Remove leading numbers from the name
  name <- trimws(gsub("^\\d+\\s+", "", name))
})
```

```
# Return as a named list
return(list(
  name = name,
  description = description,
  isMandatory = isMandatory,
  impcParameterOrigId = impcParameterOrigId
))
})

# Remove the header if present (optional: depends on your file structure)
processed_data <- processed_data[-1]

# Convert the processed data into a dataframe
df <- do.call(rbind, lapply(processed_data, as.data.frame))
rownames(df) <- NULL

# Return the dataframe
return(df)
}
```

After parsing was complete, the fields were reordered, header names were assigned to each one of them, and they were all bound to a single data frame, using base functions `rbind()` and `lapply()`, which was also the final output of the function.

The data frame output from IMPC procedure file went through further cleaning to handle additional file specific formatting issues, mainly present in the description section. The script starts by replacing specific HTML entities in the description column to improve readability and consistency. It replaces “ ” with a single space, “’” with a right single quotation mark (’), and “&” with the ampersand symbol (&). Next, the script addresses formatting issues such as missing spaces after periods and the presence of multiple consecutive spaces. It ensures that every period is followed by a space if not already, using a regular expression to detect cases where a period is immediately followed by a character without spacing. The script further enhances readability by adding a period before any word starting with a capital letter that follows a lowercase letter or a closing parenthesis.

```
# Replace "&nbsp;" with a space, "&rsquo;" with the right single quotation mark and
"&amp;" with an ampersand symbol (&)
procedures$description = procedures$description %>%
  gsub("&nbsp;", " ", .) %>%
  gsub("&rsquo;", "'", .) %>%
  gsub("&amp;", "&", .)

# Fix missing spaces after periods and replace multiple spaces with a single space
procedures$description = procedures$description %>%
  gsub("\\. (?=[^\\s])", ". ", ., perl = TRUE) %>% # Ensure spaces after periods
  gsub(" {2,}", " ", .) # Replace multiple spaces with a single space

# Add periods before words starting with a capital letter after a lowercase letter or
closing parenthesis
procedures$description <- gsub("([a-z\\)])([A-Z])", "\\1. \\2", procedures$description)
```

```
# Combine multiline rows into single lines
processed_data = c()
current_row = ""

for (line in raw_data) {
  # Remove surrounding double quotes and trim white spaces
  line <- str_trim(gsub('"', '', line))

  # Skip completely empty rows
  if (line == "") next

  # Check if the line starts with a number (line_number)
  if (grepl("^\\d+\\s", line)) { # Match numeric rows
    if (current_row != "") {
      # Save the previous row
      processed_data = c(processed_data, current_row)
    }
    current_row = line # Start a new row
  } else {
    # Otherwise, append to the current row for multiline descriptions
    current_row = paste(current_row, line, sep = " ")
  }
}

# Add the last row
if (current_row != "") {
  processed_data = c(processed_data, current_row)
}
```

```
# Split the first column into "line_number" and "rest"
procedure_split = procedure_raw %>%
  mutate(V1 = gsub("\\", "", V1)) %>% # Remove quotes
  separate(V1, into = c("line_number", "rest"), sep = " ", extra = "merge", fill =
"right")

# Extract column "name" from "rest"
procedure_split = procedure_split %>%
  mutate(
    name = str_extract(rest, "^([,]+)", # Extract "name" up to the first comma
    rest = str_remove(rest, "^([,]+)") # Remove "name" and the comma from "rest"
  ) %>%
  mutate(
    # Escape commas inside parentheses
    rest = str_replace_all(rest, "\\([([()]*?)\\)", function(x) {
      gsub(",", "|", x) # Replace all commas within parentheses with "|"
    }),
    # Escape commas up to and excluding those in ".," and skip others
    rest = str_replace_all(rest, "(?=.*\\.\\.\\.),", "| ") %>% # Replace commas up to
    ".," pattern
    str_replace_all(",(?=.)", ",") # Ignore commas after the ".," pattern
  )

# Split the "rest" column into additional fields
procedure_split = procedure_split %>%
  separate(rest, into = c("description", "isMandatory", "impcParameterOrigId"),
    sep = ",", extra = "merge", fill = "right")

# Restore escaped commas in specific cases in "description" column
procedure_split = procedure_split %>%
  mutate(description = str_replace_all(description, "\\|", ","))
```


Moreover, this script also standardises missing data by ensuring all variations of missing or placeholder (manually entered NAs) values are uniformly treated as *NA*. These detailed corrections are absent in the previous script, which focuses more on proper parsing of the values and then a second script follows to handle these issues in the specific data frame outputs.

```
# Clean manually entered "NA" values and missing values by trimming white spaces
# Convert both manually entered "NA" values and missing values to NA
Prmt_description_cleaned = Prmt_description_split %>%
  mutate_all(~ifelse(is.na(.) | str_trim(.) == "" | str_trim(.) == "NA",
                     NA, str_trim(.)))
```

Database Creation & Management

To meet the requirements of this project, we implemented a robust workflow to design and populate a scalable MySQL database tailored to store and query IMPC data. The database, named “database9”, was designed to ensure the structured storage of raw experimental analysis data, along with associated metadata such as parameter names and descriptions, test procedure names and descriptions, gene chromosomes, parameter groupings and disease associations. The design ensures normalization and facilitates the seamless integration of additional datasets or tables in the future.

The database was established directly in RStudio (version 2024.12.0.467; Posit, 2024) by programmatically connecting locally to a MySQL server using R. Connections to the database were managed through the DBI (version 1.2.3) and RMySQL (version 0.10.29) libraries, enabling efficient data interactions. The database schema consists of seven interconnected tables through primary-to-foreign key relationships: “analysis”, “chromosome gene”, “parameter groups”, “parameter description grouped”, “procedures grouped”, “impcParameterOrigIds map expanded”, and “parameter id map” (Figure 3). Additionally, the database includes a separate “diseases” table, which is not linked to other tables due to the lack of matching gene accession Id values with the main dataset, leading to blank outputs during queries. This table initially contained multiple entries for the same gene, with varying phenodigm scores. To address this, we retained only unique gene-disease pairs by selecting the entry with the highest phenodigm score for each gene.

The table “analysis” comprises of the experimental raw data integrating information such as the unique “analysis id”, gene-related fields like “gene symbol” and “gene accession id”, mouse-specific attributes including “mouse strain” and “mouse life stage”, and tested phenotypes as “parameter name” alongside their statistical results represented by “pvalue”. Furthermore, enriched metadata of the phenotypes and the procedures used for testing are incorporated into tables “parameter description grouped” and “procedures grouped” respectively.

To manage parameter complexity and reveal biologically meaningful associations, ten groups were created based on similarities in parameter names and/or the procedures used for testing. For instance, meaningful imaging groupings were achieved by considering testing techniques such as X-rays, gross morphology of embryos or placentas, and eye morphology assessments. Our grouping categories include “brain”, “weight”, “image”, “metabolic”, “hematopoietic”, “strength”,

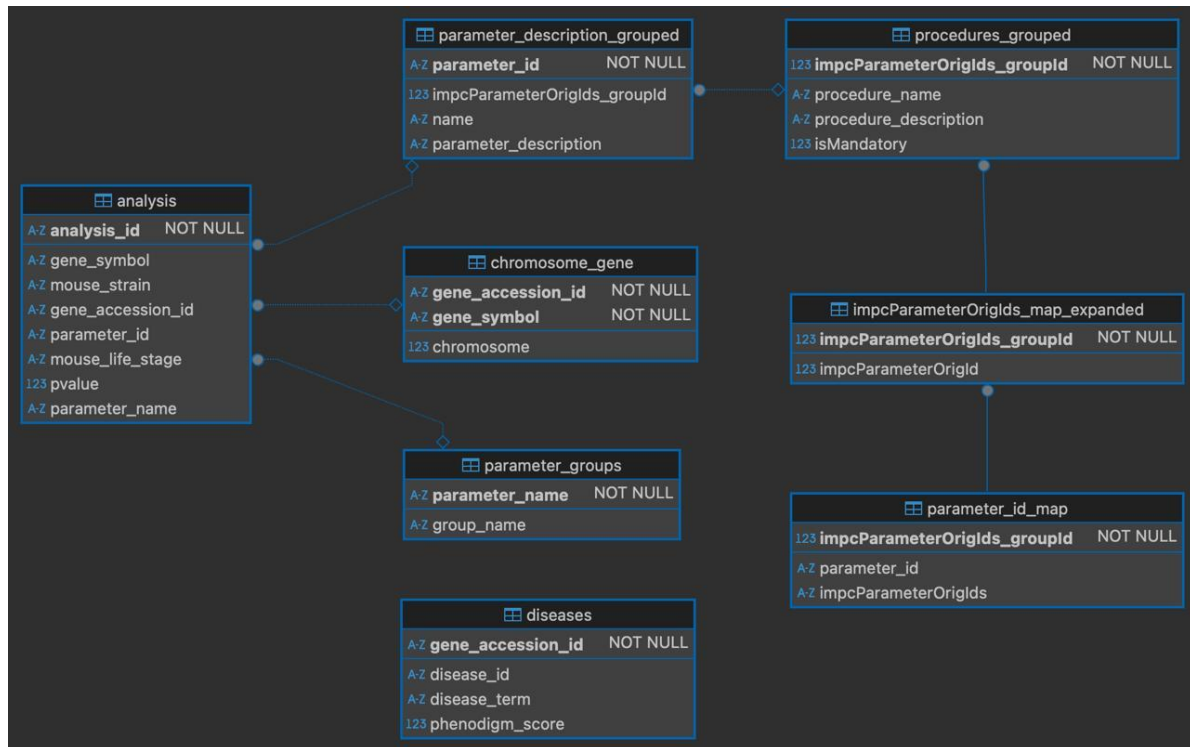


Figure 3. Database schema. The schema consists of seven interconnected tables (“analysis”, “chromosome gene”, “parameter groups”, “parameter description grouped”, “procedures grouped”, “impcParameterOrigIds map expanded”, and “parameter id map”) linked through primary-to-foreign key relationships. These tables store raw experimental data, parameter groupings, gene-to-chromosome mappings, and detailed descriptions of parameters and procedures used for testing. An additional table, “diseases”, captures information about human disease associations, including disease terms and phenodigm scores, but remains unlinked due to the absence of matching values with the other tables. The schema image was generated with DBeaver version 24.3.1

“cardiovascular”, “locomotion”, “morphology”, and “others” for those parameters not falling in any other category. Table “parameter groups” contains all these information.

The database was populated with tables created from pre-processed data frames in R. To establish primary-to-foreign key relationships between the tables, we ensured that each table contained at least one field with unique and non-null values. While the data frames “parameter description” and “IMPC procedures” both included a common unique field, “impcParameterOrigId”, allowing them to be linked directly, a primary-to-foreign key relationship with the “analysis” data frame was not feasible. This was due to the “parameter id” field, which, despite being common across the data frames, contained duplicate values. On further inspection, we found that the duplications were caused solely by the values in the “impcParameterOrigId” field.

To resolve this issue, we created a new table, “parameter id map”, to establish groupings between unique “parameter id” values and their corresponding “impcParameterOrigId” values. For each “parameter id”, one to three corresponding “impcParameterOrigId” values were concatenated into a single string, separated by commas. Additionally, a unique row number, referred to as “impcParameterOrigIds groupId”, was assigned to each parameter id to enable mapping. The resulting “parameter id map” table contains three columns: unique “parameter id” values, their

	parameter_id	impcParameterOrigIds	impcParameterOrigIds groupId
1	impc_abr_001_001	29207, 18547	1
2	impc_abr_002_001	29208, 18548	2
3	impc_abr_003_001	29209, 18549	3
4	impc_abr_004_001	29210, 18550	4

corresponding concatenated “impcParameterOrigId” values, and the grouping identifier, “impcParameterOrigIds groupId”.

To enable mapping back to the “IMPC procedures” data frame, we created another table, “impcParameterOrigIds map expanded”, which resolves the concatenated “impcParameterOrigId” values into individual rows. Each row retains its associated grouping number (“impcParameterOrigIds groupId”), facilitating linkage with the “IMPC procedures” data frame through the shared “impcParameterOrigId” field. This step allowed us to connect all three data frames (“parameter description”, “IMPC procedures”, and “analysis”) by bridging the gaps caused by duplicated or concatenated values.

	impcParameterOrigIds_groupId	impcParameterOrigId
1	1	29207
2	1	18547
3	2	29208
4	2	18548

In the final database, relationships were established as follows: the “analysis” table was connected to the “parameter description grouped” table via the common “parameter id” field. Additionally, a primary-to-foreign key linkage was created between the “parameter description grouped” and “procedures grouped” tables using the “impcParameterOrigIds groupId” field. This structured approach ensured seamless integration of the data and maintained referential integrity across the tables.

Scalability was enhanced by incorporating new tables beyond the initial dataset, such as the “chromosome gene” table, which added chromosomal information. Gene-to-chromosome associations in mice were obtained through a batch query of all unique gene accession IDs in our dataset using www.informatics.jax.org (Baldarelli et al., 2024). Additionally, data types for each field were carefully defined to optimize memory usage and support future expansions. For instance, setting appropriate VARCHAR limits reduced resource demands while maintaining flexibility. To ensure accuracy in field limits, we adhered to the guidelines outlined in the SOP.

Interactive Dashboard

The interactive dashboard, developed using RShiny (version 1.9.1), provides a user-friendly platform for visualizing IMPC mouse phenotypic data with real-time queries from the “database9” MySQL database. It supports three visualization modes: statistical scores of phenotypes tested on a selected knockout mouse, scores of all knockout mice for a chosen phenotype, and clusters of genes with similar phenotype scores based on p-values. The dashboard integrates several libraries to enhance functionality, including Shiny for interactivity, ggplot2 (version 3.5.1) and plotly (version 4.10.4) for vibrant and interactive visualizations, and the umap library (version 0.2.10.0) for clustering and dimensionality reduction. Additionally, scales (version 1.3.0) and dplyr ensure efficient data filtering, transformations, and visual enhancements.

The first visualization mode allows users to select a knockout mouse and explore the statistical significance of tested phenotypes. It includes filtering options for gene symbol, mouse strain, and life stage, which were identified in our data as influencing the phenotypic score. This mode enables users to view detailed information, such as p-values, associated phenotypes, and procedural descriptions, through interactive data points. A horizontal dashed line marks the p-value threshold of 1×10^{-4} , following IMPC guidelines to maintain a false discovery rate (FDR) below 5% (Haselimashhadi et al., 2020). Color-coded groupings further highlight related phenotypes. The second mode shifts the focus to a specific phenotype, allowing users to examine all associated

knockout mice. This perspective helps identify genotype-phenotype patterns and significant gene interactions.

The third mode employs UMAP (Uniform Manifold Approximation and Projection) to group genes based on phenotype scores, particularly their statistical significance (p-values). Only genes with significant p-values are displayed, and users can customize the number of clusters, choosing from 15 options to suit their analysis. This clustering approach helps uncover patterns by grouping genes with similar phenotypic effects, offering insights into potential shared biological roles or pathways.

The dashboard prioritizes user experience with dynamically populated dropdown menus, customizable visualizations, and interactive tooltips for detailed exploration. For instance, hovering over a point reveals key details such as phenotype name or gene symbol, grouping category, and p-value. Zooming features allow users to focus on specific areas of interest, either by selecting directly on the plot or using toolbar options. The system's design ensures seamless scalability by automatically adapting to new data entries or structural changes in the database, supporting the integration of expanding datasets and meeting evolving research demands.

Results

To evaluate the functionality of the first visualization mode, four specific knockout mice were selected to test different output scenarios. Each knockout mouse is characterized by a unique combination of gene symbol, mouse strain, and developmental stage. The first selection was a knockout mouse with the gene symbol 1700003F12RIK, of the C57BL strain, and at the E15.5 life stage. This mouse is associated with one statistically significant phenotype related to heart morphology, with a p-value rounded to 0 (Figure 4). The associated test procedure involved gross morphology embryo screening conducted between embryonic days 14.5 (E14.5) and 15.5 (E15.5).

The second selection involved a mouse knockout of the PROM2 gene, also of the C57BL strain, but at the early adult life stage. This mouse is associated with six phenotypes presenting statistically significant scores ($p < 1 \times 10^{-4}$), spanning the metabolic, hematopoietic, morphology, and strength categories (Figure 5). Specifically, these phenotypes correspond to calcium levels, mean cell hemoglobin concentration, retina abnormalities, forelimb and hindlimb grip strength measurement mean, forelimb grip strength measurement mean, and forelimb strength normalized against body weight.

The third selection combination yielded no results in the database (Figure 6), demonstrating a scenario where no phenotypic data is available for the selected parameters. Lastly, the fourth selection tested a knockout mouse with the gene symbol ALG14, of the C57BL strain, and at the early adult life stage. This mouse had no associated statistically significant phenotypes (Figure 7).

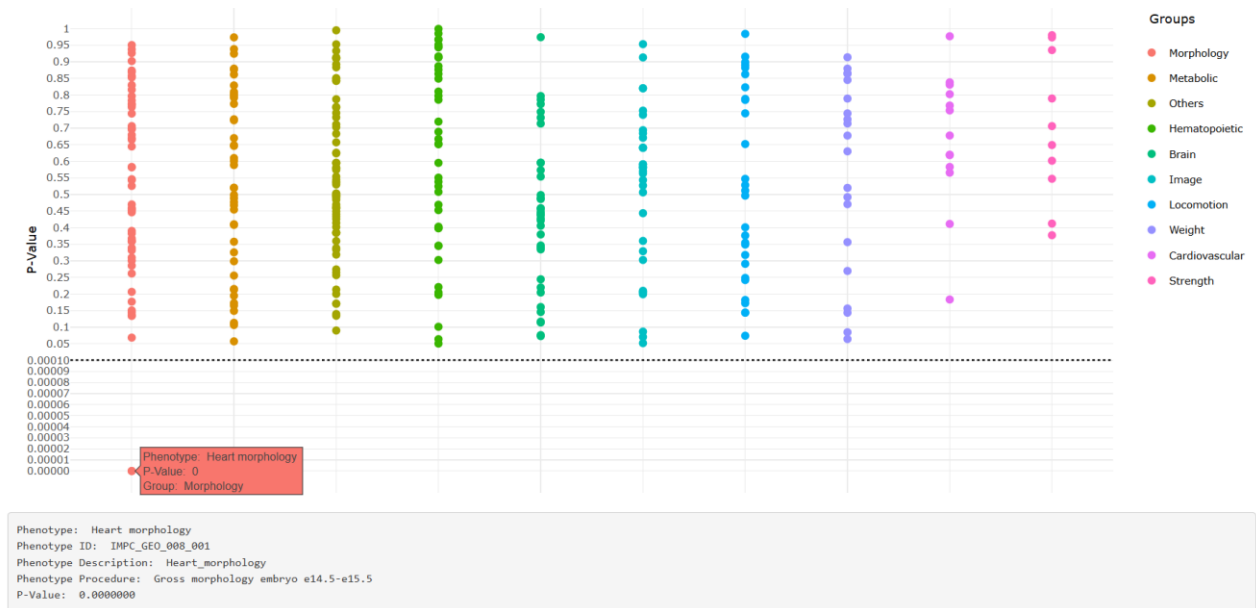


Figure 4. Plot generated by the first visualisation mode of the interactive dashboard. The plot depicts among all the tested phenotypes for this specific knockout mouse (gene symbol: 1700003F12RIK, mouse strain: C57BL, life stage: E15.5) a single associated significant phenotype. Phenotypes below the dashed line ($p\text{-value} = 1 \times 10^{-4}$) are considered statistically significant. In this case, the phenotype of interest corresponds to heart morphology with a p-value score close to 0. It is grouped under the morphology category, and Gross morphology embryo screening between days 14.5 and 15.5 was conducted to test this phenotype.

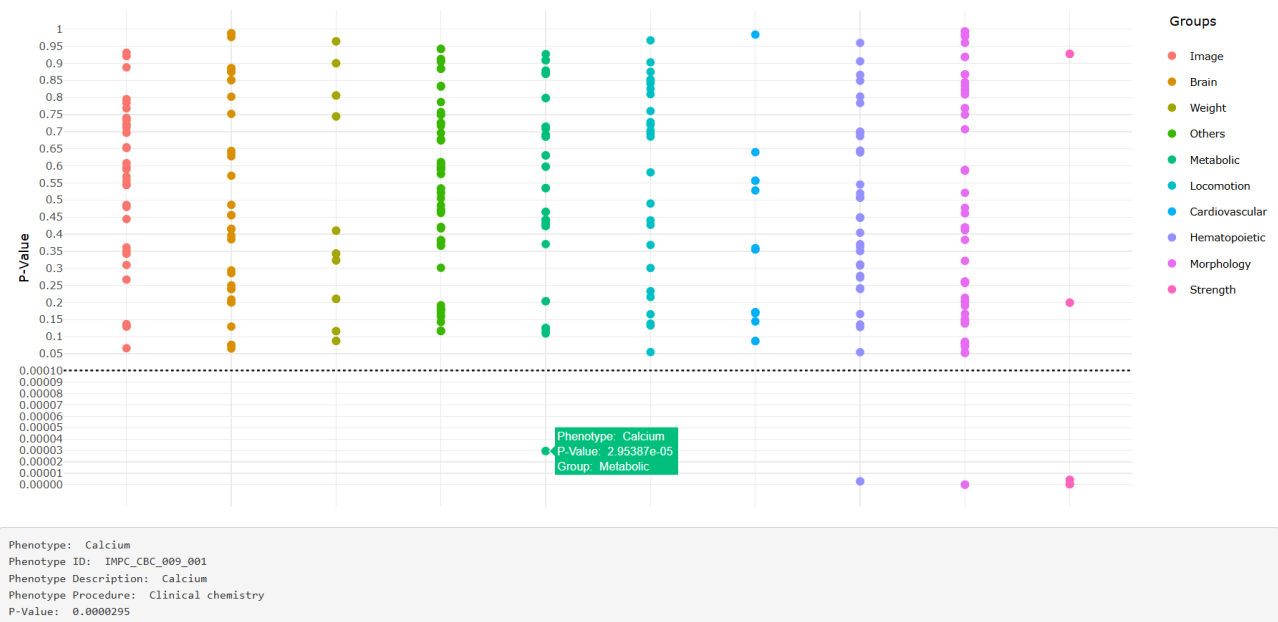


Figure 5. Plot generated by the first visualisation mode of the interactive dashboard. The plot depicts among all the tested phenotypes for this specific knockout mouse (gene symbol: PROM2, mouse strain: C57BL, life stage: early adult) six associated significant phenotypes. Phenotypes below the dashed line ($p\text{-value} = 1 \times 10^{-4}$) are considered statistically significant. In this case, the phenotypes of interest correspond to metabolic, hematopoietic, morphology and strength groups. More specifically, the strength category (pink) is comprised of three statistically significant phenotypes associated with this knockout mouse.



Figure 6. Plot generated by the first visualisation mode of the interactive dashboard. The plot depicts the phrase “No data available for the selection”. This output is the result of a selected combination—gene symbol: APH1A, mouse strain: 129SV, life stage: middled aged adult— associated with no data in the database.

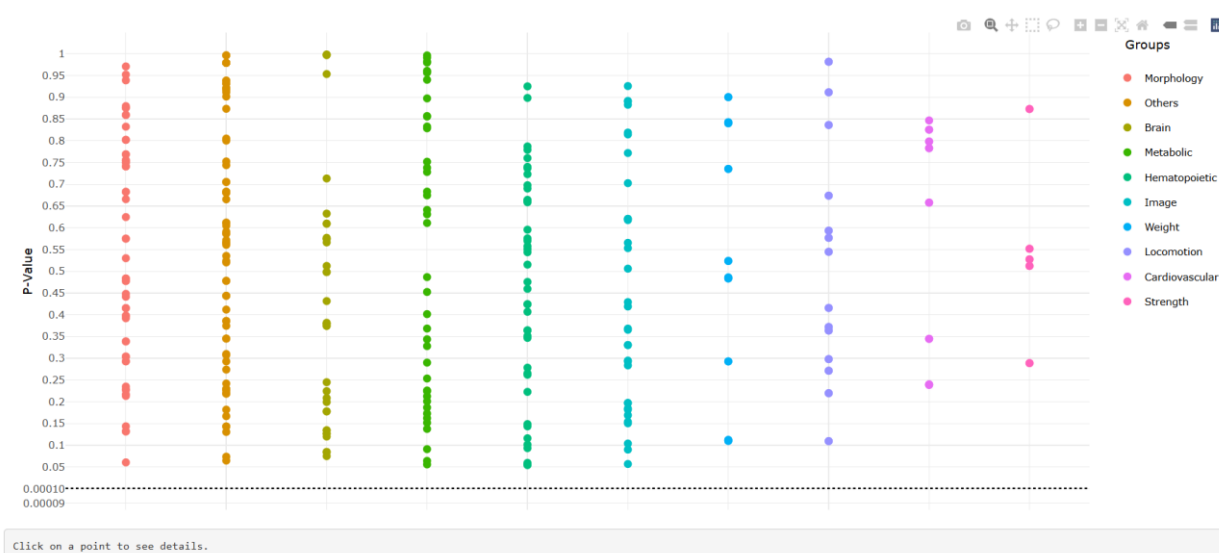


Figure 7. Plot generated by the first visualisation mode of the interactive dashboard. The plot depicts among all the tested phenotypes for this specific knockout mouse (gene symbol: ALG14, mouse strain: C57BL, life stage: early adult) no associated significant phenotypes. Phenotypes below the dashed line ($p\text{-value} = 1 \times 10^{-4}$) are considered statistically significant.

Discussion

The development of a scalable MySQL database and an interactive RShiny dashboard for managing and visualizing IMPC phenotypic data reflects an integrative approach to data curation and user-centered design. The project required the incorporation of robust SQL queries and dynamic R-based scripts to address the complexities of large-scale genotype-phenotype data analysis. In the interactive dashboard, SQL commands play a critical role in retrieving and filtering data dynamically from our MySQL database for the three visualization modes available: phenotype significance, gene significance, and clustering of genes.

For the first mode (phenotype significance visualisation), the SQL query dynamically filters data to display statistical scores for all tested phenotypes of a selected knockout mouse. The dropdown menus for gene symbol, mouse strain, and mouse life stage are populated using the *SELECT DISTINCT* SQL command. This query extracts all unique gene symbols, mouse strains and life stages respectively from the “analysis” table, converting them to uppercase for standardization.

```
# Populate dropdown menus with distinct values from database
observe({
  updateSelectInput(session, "gene_symbol",
    choices = dbGetQuery(con, "SELECT DISTINCT
                                UPPER(gene_symbol) as gene_symbol
                                FROM analysis")$gene_symbol)
  updateSelectInput(session, "mouse_strain",
    choices = dbGetQuery(con, "SELECT DISTINCT
                                UPPER(mouse_strain) as mouse_strain
                                FROM analysis")$mouse_strain)
  updateSelectInput(session, "mouse_life_stage",
    choices = dbGetQuery(con, "SELECT DISTINCT
                                UPPER(mouse_life_stage) as mouse_life_stage
                                FROM analysis")$mouse_life_stage)
})
```

When the user selects specific filters and clicks "Run," the following query is executed:

```
# Query the database based on user input and filter pvalue
filtered_data <- eventReactive(input$update_plot, {
  query <- paste0(
    "SELECT UPPER(a.parameter_id) AS parameter_id,
    CONCAT(UCASE(SUBSTRING(a.parameter_name, 1, 1)),
    LOWER(SUBSTRING(a.parameter_name, 2))) AS parameter_name,
    CONCAT(UCASE(SUBSTRING(pdg.parameter_description, 1, 1)),
    LOWER(SUBSTRING(pdg.parameter_description, 2))) AS parameter_description,
    CONCAT(UCASE(SUBSTRING(pg.group_name, 1, 1)),
    LOWER(SUBSTRING(pg.group_name, 2))) AS group_name,
    CONCAT(UCASE(SUBSTRING(prg.procedure_name, 1, 1)),
    LOWER(SUBSTRING(prg.procedure_name, 2))) AS procedure_name,
    a.pvalue
    FROM analysis a
    LEFT JOIN parameter_description_grouped pdg ON a.parameter_id = pdg.parameter_id
    LEFT JOIN parameter_groups pg ON a.parameter_name = pg.parameter_name
    LEFT JOIN procedures_grouped prg ON
    pdg.impcParameterOrigIds_groupId = prg.impcParameterOrigIds_groupId
    WHERE a.gene_symbol = '", input$gene_symbol, "' AND
    a.mouse_strain = '", input$mouse_strain, "' AND
    a.mouse_life_stage = '", input$mouse_life_stage, "'"
  )
})
```

Dynamic filtering is achieved through the *WHERE* clause, which applies user-selected filters such as gene symbol, mouse strain, and life stage. For example, if a user selects “PROM2” as the gene symbol, “C57BL” as the strain, and a specific life stage, the query isolates the phenotypes tested under these conditions. This ensures that only the most relevant data is retrieved for visualization. To further enhance the presentation, the queries employ SQL formatting functions. To gather data from multiple tables, the query uses *LEFT JOIN* statements. This allows for the inclusion of

additional metadata such as parameter descriptions, grouping categories, and procedural information.

Once the data is queried, it is further processed in R using the *mutate()* function. A new column, “Significance”, is added to classify p-values below 1×10^{-4} as “Significant” and those above as “Not Significant.” This transformation allows users to quickly identify statistically significant results directly within the dashboard’s visualizations.

```
result <- dbGetQuery(con, query) %>%  
  mutate(Significance = ifelse(pvalue < 0.0001, "Significant", "Not Significant"))  
})
```

The second mode focuses on statistical scores of all knockout mice tested for a selected phenotype. It uses the same logic to populate the dropdown menu, but this time with unique parameter names. Upon selection, the following query not only retrieves gene symbols and p-values but also includes chromosome information by joining the “chromosome gene” table. Formatting standardization is also applied in this case as well.

```
# Query the database based on user input and filter pvalue  
filtered_data <- eventReactive(input$update_plot, {  
  query <- paste0(  
    "SELECT UPPER(a.gene_symbol) AS gene_symbol,  
    UPPER(a.gene_accession_id) AS gene_accession_id,  
    UPPER(a.mouse_strain) AS mouse_strain,  
    UPPER(a.mouse_life_stage) AS mouse_life_stage,  
    cg.chromosome,  
    a.pvalue,  
    CONCAT(UCASE(SUBSTRING(pg.group_name, 1, 1)),  
    LOWER(SUBSTRING(pg.group_name, 2))) AS group_name  
    FROM analysis a  
    LEFT JOIN parameter_groups pg ON a.parameter_name = pg.parameter_name  
    LEFT JOIN chromosome_gene cg ON a.gene_accession_id = cg.gene_accession_id  
    AND a.gene_symbol = cg.gene_symbol  
    WHERE a.parameter_name = '", input$phenotype, "'"  
  )  
  result <- dbGetQuery(con, query) %>%  
    mutate(Significance = ifelse(pvalue < 0.0001, "Significant", "Not Significant"))  
})
```

In the third and final visualisation mode of our interactive dashboard, clustering is performed based on phenotype scores, leveraging SQL to fetch necessary data for clustering algorithms like UMAP. The query retrieves phenotype names, gene symbols, p-values, and grouping information. This is achieved by linking phenotype names to their corresponding groups by joining the “parameter groups” table to the “analysis” table.

```
# Reactive data filtered from the database  
filtered_data <- eventReactive(input$update_plot, {  
  query <- "SELECT UPPER(a.gene_symbol) AS gene_symbol,  
    CONCAT(UCASE(SUBSTRING(a.parameter_name, 1, 1)),  
    LOWER(SUBSTRING(a.parameter_name, 2))) AS parameter_name,  
    a.pvalue,  
    CONCAT(UCASE(SUBSTRING(pg.group_name, 1, 1)),  
    LOWER(SUBSTRING(pg.group_name, 2))) AS group_name  
    FROM analysis a  
    LEFT JOIN parameter_groups pg ON a.parameter_name = pg.parameter_name"  
  dbGetQuery(con, query)  
})
```

Clustering through UMAP helps identify patterns in the data by grouping genes that exhibit similar phenotypic effects when knocked out. These clusters can indicate potential shared biological roles or pathways, as genes producing similar statistical profiles across related phenotypes are more likely to participate in the same or interconnected biological processes (Gui et al., 2017). While the clustering is primarily based on p-values in this case, the addition of other contextual data—such as grouping categories or pathway annotations—could further enrich the biological insights derived from the clusters.

A significant strength of the system is its scalability, achieved through the database's dynamic structure and the dashboard's adaptability to new data entries or changes in the database schema. The automated population of dropdown menus and the ability to reflect updates without manual intervention ensure the platform remains functional and relevant as datasets grow. Furthermore, the incorporation of interactive tooltips and customizable visualizations enhances the user experience, making the platform accessible even to non-specialist users.

Despite its strengths, the system has limitations. Addressing duplicate fields through intermediary mapping tables, while effective, introduced additional complexity to the database design. Similarly, the reliance on phenotype scores for clustering genes, without integrating functional annotations or pathway analyses, limits the biological inferences that can be drawn directly from the dashboard. Future iterations could incorporate external ontologies or functional data to refine clustering outputs further.

Overall, the project highlights the effective use of SQL for managing and querying large datasets and the power of RShiny in creating a user-friendly analytical interface. By addressing the challenges of large-scale data integration and visualization, the platform provides a robust foundation for future expansions, enabling researchers to explore genotype-phenotype relationships with greater precision and ease.

Supplementary materials

All scripts, along with detailed project information and instructions, are available on our GitHub repository: <https://github.com/bpulugundla/Genotype2Phenotype.git>. The repository includes the R scripts used for data processing, database creation, and interactive dashboard development, as well as relevant documentation to facilitate reproducibility.

The database associated with this project is named **database9**

References

1. Baldarelli, R.M., Smith, C.L., Ringwald, M., Richardson, J.E., Bult, C.J., Mouse Genome Informatics Group, 2024. Mouse Genome Informatics: an integrated knowledgebase system for the laboratory mouse. *Genetics*, 227(1):iyae031.
2. Gui, H., Kwan, J.S., Sham, P.C., Cherny, S.S. and Li, M., 2017. Sharing of genes and pathways across complex phenotypes: a multilevel genome-wide analysis. *Genetics*, 206(3), pp.1601-1609.
3. Haselimashhadi, H., Mason, J.C., Mallon, A.M., Smedley, D., Meehan, T.F. and Parkinson, H., 2020. OpenStats: A robust and scalable software package for reproducible analysis of high-throughput phenotypic data. *PLoS One*, 15(12), p.e0242933.
4. Hrabě de Angelis, M., Nicholson, G., Selloum, M., White, J.K., Morgan, H., Ramirez-Solis, R., Sorg, T., Wells, S., Fuchs, H., Fray, M. and Adams, D.J., 2015. Analysis of mammalian gene function through broad-based phenotypic screens across a consortium of mouse clinics. *Nature genetics*, 47(9), pp.969-978.
5. International Mouse Phenotyping Consortium (IMPC), (n.d.). *About IMPC*. [online] Available at: <https://www.mousephenotype.org/about-imp/> [Accessed 9 Jan. 2025].
6. Mallon, A.M., Iyer, V., Melvin, D., Morgan, H., Parkinson, H., Brown, S.D., Flicek, P. and Skarnes, W.C., 2012. Accessing data from the International Mouse Phenotyping Consortium: state of the art and future plans. *Mammalian genome*, 23, pp.641-652.
7. Posit team, 2024. RStudio: Integrated development environment for R [computer software]. *Boston, MA: Posit Software, PBC*.