Connect R to Microsoft Teams

Taru Dutt

2022-10-19

Contents

Chapter 1

Overview

1.1 About the project

The objective of the Immune Mechanisms of Protection against Mycobacterium tuberculosis (IMPAc-TB) program is to get a thorough understanding of the immune responses necessary to avoid initial infection with *Mycobacterium tuberculosis (Mtb)*, formation of latent infection, and progression to active TB illness. To achieve these goals, the National Institute of Allergy and Infectious Diseases awarded substantial funding and established multidisciplinary research teams that will analyze immune responses against *Mtb* in animal models (mice, guinea pigs, and non-human primates) and humans, as well as immune responses elicited by promising vaccine candidates. The contract awards establish and give up to seven years of assistance for IMPAc-TB Centers to explain the immune responses required for *Mtb* infection protection.

The seven centers that are part of the study are (in alphabetical order):

1. Colorado State University
2. Harvard T.H. Chan School of Public Health
3. Seattle Children Hospital
4. [more]

Colorado State University Team and role of each member:

- Dr. Marcela Henao-Tamayo: Principal Investigator
- Dr. Brendan Podell: Principal Investigator
- Dr. Andres Obregon-Henao: Research Scientist-III
- Dr. Taru S. Dutt: Research Scientist-I
- [more]

1.2 About this book

The aim of this book is to provide data protocols and data collection templates for key types of data that are collected over the course of this project. By using standard templates to record data, as well as starting from defined pipelines to process and analyze the data, we aim to standardize the collection and processing of data across this project.

Here, we have built a comprehensive guide to wet lab data collection, sample processing, and computational tool creation for robust and efficient data analysis and dissemination.

Chapter 2

Experimental metadata

Metadata for an experiment:

- `species`
- `start_date`
- `end_date`
- `experimental_groups`

Chapter 3

Initial mouse characteristics

At the start of each experiment with a mouse model, we record several measurements or characteristics of each mouse. We record these measurements along with an identifier for each mouse (for example, based on tags or ear notches), so that we can later link the initial characteristics of each mouse with later measurements on the same mouse.

The values that we initially record for each mouse include:

- `group`: An identifier for the experimental group to which the mouse is assigned (e.g., "Control", "Group 1")
- `group_detail`: A longer description of the mouse's treatment group (e.g., "Vaccinated with vaccine candidate A at 4 and 8 weeks")
- `notch_id`: The ear notch pattern of the mouse (e.g., "0" for no notch, "1R" for one notch in the right ear)
- `mouse_number`: A number that corresponds with the mouse's ear notch patter; together with the mouse's group number, this provides a unique identifier for each mouse in the experiment
- `cage_number`: The number of the cage to which the mouse is first assigned. This may change over the course of the experiment, as mice might be removed from a cage due to fighting, etc. Any of these later changes of cage will be recorded [where]
- `sex`: Whether the mouse is male ("m") or female ("f")
- `age`: Age
- `strain`: The strain of the mouse (e.g., "C57BL/6J" for Black 6, "C3HeB/FeJ" for Kramnik)

We have created a spreadsheet template that can be used to record these data, which you can download by clicking here. This template currently includes example data (colored in blue to help you remember that it's only there as an example). To Use this template, take a look at the example data, then delete it and replace with the real data for your experiment.

Here is an example of how the first rows of this template might look once it's filled out:

	A	B	C	D	E	F
1	group	group_detail	cage_number	m/f	ear_punch_id	mouse_number
2	1	bcg		f	0	1
3	1	bcg		f	1R	2
4	1	bcg		f	1L	3
5	1	bcg		f	1R1L	4
6	2	bcg+id93		f	0	1
7	2	bcg+id93		f	1R	2
8	2	bcg+id93		f	1L	3
9	2	bcg+id93		f	1R1L	4
10	3	saline		f	0	1
11	3	saline		f	1R	2
12	3	saline		f	1L	3
13	3	saline		f	1R1L	4
14	4	saline+id93		f	0	1
15	4	saline+id93		f	1R	2

This template should be used at the initial time when mice are brought into the experiment. The file format is an Excel file, so you can use it by saving it to your computer and then opening and recording data with Excel. Later code in this chapter will read in a file in this template format to provide basic summaries of the data. Later code will read in these files to record the data in a project-wide database, which will allow us to integrate it with other data collected over the course of the experiment.

[Rules for naming the file. Include experiment name / study ID?]

Chapter 4

Mouse Weights

4.0.1 Overview

Extreme weight loss and loss of muscle mass, also known as cachexia, typically presents along side chronic inflammatory illnesses like Tuberculosis disease (Baazim et al., 2022). We now recognize that cachexia is part of a systemic response to inflammation, and has been linked to upregulation of pro-inflammatory cytokines such as TNF, IL-6, and IFNg in humans (Baazim et al., 2022). Additionally, studies support the role of key immune cell populations such as CD8+ T-cells that, when depleted, counteract muscle and fat deterioration (Baazim et al., 2019), and suggest that CD8+ T-cells may metabolically reprogram adipose tissue.

In recognition of cachexia related illnesses and diseases, we tracked the progression of weight loss over the course of this study, as is done with many TB-mouse studies (Smith et al., 2022; Segueni et al., 2016). These data is also useful when correlating to CFU count as well as expression of cytokines and other biological markers (Smith et al., 2022). Here, mice are weighed in grams weekly to monitor clinical status as TB patients frequently display weight loss as clinical symptom associated with disease progression.

The following contains information about how the data was collected, organized, and curated for analysis in RStudio.

4.0.2 Parameters

Weights are recorded in an excel worksheet.

Column titles are as follows:

- `who_collected`: Record the first name of the person who actually handled the mouse from the scale.

- `date_collected`: Record the date using quotation marks, with the month, then day, then year. For example, "May 31, 2022".
- `sex`: Record as "m" for male or "f" for female
- `notch_id`: Record the ear notch pattern in the mouse. Make sure that you record consistently across all timepoints, so that each mouse can be tracked across dates. If you are doing single notches, for example, this might be "0" for no notches, "1R" for one notch in the right ear, "1L" for one notch in the left ear, and "1R1L" for one notch in each ear.
- `mouse_number`: Record as a number identifying mice within a cage. This should be consistent across all recorded timepoints (i.e., the mouse in a cage with ear notch "1L" might be given the number "1"; if so, its number should always be recorded in this column as "1"). This number will repeat across different cages, but if you combine this number with the cage number, it gives a unique identifier for each mouse in the study.
- `weight`: Record as a number, with a unit in this column. The next column will be used for the units.
- `unit`: Provide the units that were used to take the weight (e.g., "g" for grams)
- `cage_number`: Provide the cage number. If a provisional cage number is being used, … If the mouse was switched from its original cage, …
- `group`: Provide the experimental group of the mouse. Be sure that you use the same abbreviation or notation across each timepoint. Examples of group designations might be: bcg, saline, bcg+id93, saline+id93, saline+noMtb
- `notes`: The notes column contains information regarding clinical observations.

good reference: https://elifesciences.org/articles/74419#s4

```
library(readxl)
library(tidyverse)
```

4.0.3 Read in data

Data is stored in one excel sheet, each week is one sheet named as the date -> return vector for each sheet name

```
# Function to add unique mouse IDs based on their most up-to-date cages
# By using the most up-to-date cages, we can use an ID that will be easy
# to use to identify a specific mouse at the current time (for example,
# if you need to ID a mouse with a large change in weight that has happened
# recently.)
#
# !! NEED TO CHANGE: Right now, it would miss mice that were sacrificed / died
# !! before the latest time point that was collected
add_latest_mouse_id <- function(mouse_weight_data) {
```

```
  mouse_weight_data <- mouse_weight_data %>%
  mutate(current_cage = case_when(
    date_collected != last(date_collected) ~ as.character(NA),
    !is.na(new_cage_number) ~ new_cage_number,
    TRUE ~ existing_cage_number
  ),
         mouse_id = if_else(!is.na(current_cage),
                            paste(current_cage, notch_id, sep = "-"),
                            as.character(NA)))
  return(mouse_weight_data)
}

ex <- add_latest_mouse_id(ex)

ex %>%
  mutate(mouse_id = case_when(
    # Only create for the first time point, otherwise set to NA
    date_collected != first(date_collected) ~ NA_character_,
    # If the new cage number is missing, it means the animal did not
    # change cages, so you can use the existing cage number
    is.na(new_cage_number) ~ paste(existing_cage_number, notch_id, sep = "-"),
    TRUE ~ paste(new_cage_number, notch_id, sep = "-")
  ))
```

```
## # A tibble: 980 x 14
##    who_collected date_collected sex   dob        notch_id mouse_number weight
##    <chr>         <date>         <fct> <date>     <chr>           <dbl>  <dbl>
##  1 Taru          2022-05-26     f     2022-04-05 0                   1   18.4
##  2 Taru          2022-05-26     f     2022-04-05 1R                  2   17.2
##  3 Taru          2022-05-26     f     2022-04-05 1L                  3   17
##  4 Taru          2022-05-26     f     2022-04-05 1R1L                4   18.8
##  5 Taru          2022-05-26     f     2022-04-05 0                   1   18.4
##  6 Taru          2022-05-26     f     2022-04-05 1R                  2   17.7
##  7 Taru          2022-05-26     f     2022-04-05 1L                  3   20.2
##  8 Taru          2022-05-26     f     2022-04-05 1R1L                4   17.1
##  9 Taru          2022-05-26     f     2022-04-05 0                   1   17.6
## 10 Taru          2022-05-26     f     2022-04-05 1R                  2   20
## # ... with 970 more rows, and 7 more variables: unit <chr>,
## #   existing_cage_number <chr>, new_cage_number <chr>, group <chr>,
## #   notes <chr>, current_cage <chr>, mouse_id <chr>
```

```
add_first_mouse_id <- function(mouse_weight_data) {

  mouse_weight_data <- mouse_weight_data %>%
  mutate(mouse_id = case_when(
    # Only create for the first time point, otherwise set to NA
```

```
    date_collected != first(date_collected) ~ NA_character_,
    # If the new cage number is missing, it means the animal did not
    # change cages, so you can use the existing cage number
    is.na(new_cage_number) ~ paste(existing_cage_number, notch_id, sep = "-"),
    TRUE ~ paste(new_cage_number, notch_id, sep = "-")
  ))
  return(mouse_weight_data)
}

add_latest_mouse_id(ex)
```

```
## # A tibble: 980 x 14
##    who_collected date_collected sex   dob        notch_id mouse_number weight
##    <chr>         <date>         <fct> <date>     <chr>           <dbl>  <dbl>
##  1 Taru          2022-05-26     f     2022-04-05 0                   1   18.4
##  2 Taru          2022-05-26     f     2022-04-05 1R                  2   17.2
##  3 Taru          2022-05-26     f     2022-04-05 1L                  3   17
##  4 Taru          2022-05-26     f     2022-04-05 1R1L                4   18.8
##  5 Taru          2022-05-26     f     2022-04-05 0                   1   18.4
##  6 Taru          2022-05-26     f     2022-04-05 1R                  2   17.7
##  7 Taru          2022-05-26     f     2022-04-05 1L                  3   20.2
##  8 Taru          2022-05-26     f     2022-04-05 1R1L                4   17.1
##  9 Taru          2022-05-26     f     2022-04-05 0                   1   17.6
## 10 Taru          2022-05-26     f     2022-04-05 1R                  2   20
## # ... with 970 more rows, and 7 more variables: unit <chr>,
## #   existing_cage_number <chr>, new_cage_number <chr>, group <chr>,
## #   notes <chr>, current_cage <chr>, mouse_id <chr>
```

```
# Function to get the cage history of a mouse based on it's latest mouse ID
# and the mouse weight data. This works back through cage changes to track the
# cages that a mouse has been in over the course of the experiment.
get_cage_history <- function(latest_mouse_id,
                             mouse_weight_data) {

  cage_history <- latest_mouse_id %>%
    str_remove("\\-.+")
  last_cage <- cage_history[1]

  while(length(last_cage) > 0){
    last_cage <- mouse_weight_data %>%
      filter(new_cage_number == last_cage) %>%
      pull(existing_cage_number)
    if(length(last_cage) > 0) {
      cage_history <- c(cage_history, last_cage)
    }
  }
```

```
  cage_history <- unique(cage_history)

  return(cage_history)
}

get_cage_history(latest_mouse_id = "22009A-0",
                 mouse_weight_data = ex)
```

```
## [1] "22009A" "22009"
```

```
ex_cage_history <- get_cage_history(latest_mouse_id = "22009A-0",
                 mouse_weight_data = ex)


# Use the function to get cage histories for all the mice
latest_mouse_ids <- ex %>%
  filter(!is.na(mouse_id)) %>%
  pull(mouse_id) %>%
  unique()

mouse_cage_histories <- purrr::map(latest_mouse_ids, get_cage_history,
                                   mouse_weight_data = ex)
names(mouse_cage_histories) <- latest_mouse_ids
```

```
add_one_mouse_id <- function(latest_mouse_id, cage_history, mouse_weight_data) {

  mouse_notch <- str_remove(latest_mouse_id, ".+\\-")

  mouse_weight_data <- mouse_weight_data %>%
    mutate(mouse_id = if_else((existing_cage_number %in% cage_history) &
                                notch_id == mouse_notch,
           latest_mouse_id,
           mouse_id))

  return(mouse_weight_data)
}

ex2 <- add_one_mouse_id(latest_mouse_id = names(mouse_cage_histories)[1],
                 cage_history = mouse_cage_histories[[1]],
                 mouse_weight_data = ex)

ex2 %>% filter(existing_cage_number == "22003" & notch_id == "0")
```

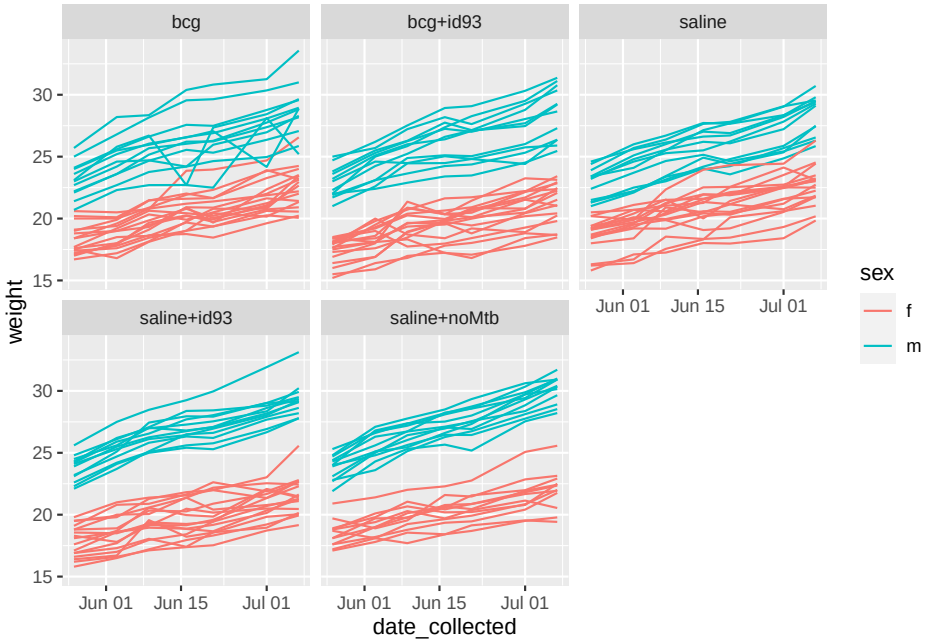
```
## # A tibble: 7 x 14
##   who_collected date_collected sex   dob        notch_id mouse_number weight
##   <chr>         <date>         <fct> <date>     <chr>           <dbl>  <dbl>
```

```
## 1 Taru          2022-05-26     f      2022-04-05 0                    1   18.4
## 2 Forrest       2022-06-03     f      2022-04-05 0                    1   19
## 3 Pablo         2022-06-09     f      2022-04-05 0                    1   19.3
## 4 Forrest       2022-06-16     f      2022-04-05 0                    1   20.9
## 5 Amanda        2022-06-21     f      2022-04-05 0                    1   20.9
## 6 Forrest       2022-07-01     f      2022-04-05 0                    1   22
## 7 Pablo         2022-07-07     f      2022-04-05 0                    1   23.1
## # ... with 7 more variables: unit <chr>, existing_cage_number <chr>,
## #   new_cage_number <chr>, group <chr>, notes <chr>, current_cage <chr>,
## #   mouse_id <chr>
```

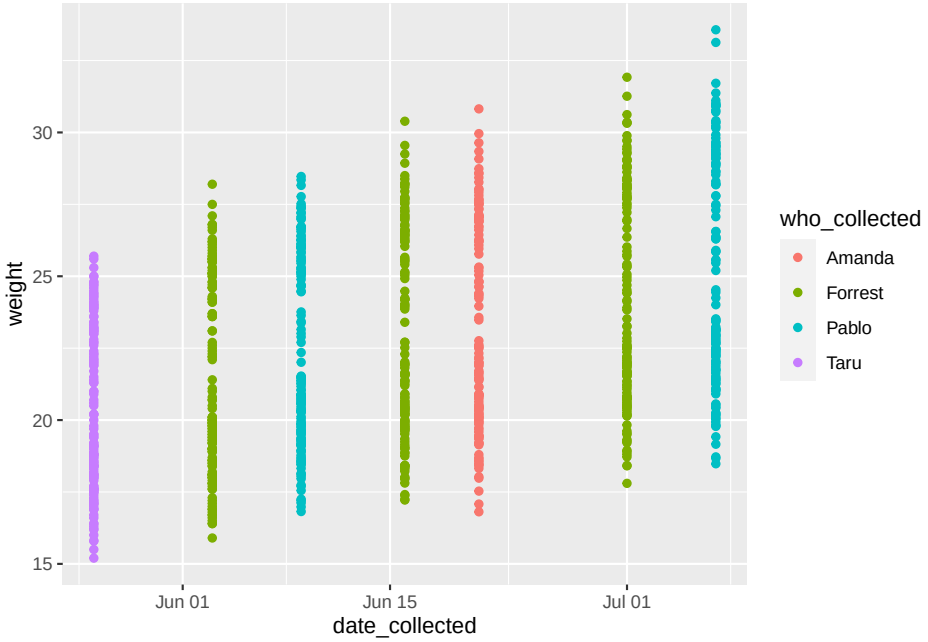
```
for(i in 1:length(mouse_cage_histories)) {
  ex <- add_one_mouse_id(latest_mouse_id = names(mouse_cage_histories)[i],
                         cage_history = mouse_cage_histories[[i]],
                         mouse_weight_data = ex)
}

ex
```

```
## # A tibble: 980 x 14
##    who_collected date_collected sex   dob        notch_id mouse_number weight
##    <chr>         <date>         <fct> <date>     <chr>            <dbl>  <dbl>
##  1 Taru          2022-05-26     f     2022-04-05 0                    1   18.4
##  2 Taru          2022-05-26     f     2022-04-05 1R                   2   17.2
##  3 Taru          2022-05-26     f     2022-04-05 1L                   3   17
##  4 Taru          2022-05-26     f     2022-04-05 1R1L                 4   18.8
##  5 Taru          2022-05-26     f     2022-04-05 0                    1   18.4
##  6 Taru          2022-05-26     f     2022-04-05 1R                   2   17.7
##  7 Taru          2022-05-26     f     2022-04-05 1L                   3   20.2
##  8 Taru          2022-05-26     f     2022-04-05 1R1L                 4   17.1
##  9 Taru          2022-05-26     f     2022-04-05 0                    1   17.6
## 10 Taru          2022-05-26     f     2022-04-05 1R                   2   20
## # ... with 970 more rows, and 7 more variables: unit <chr>,
## #   existing_cage_number <chr>, new_cage_number <chr>, group <chr>,
## #   notes <chr>, current_cage <chr>, mouse_id <chr>
```

```
# Explore this data a bit
ex %>%
  ggplot(aes(x = date_collected, y = weight, group = mouse_id, color = sex)) +
  geom_line() +
  facet_wrap(~ group)
```

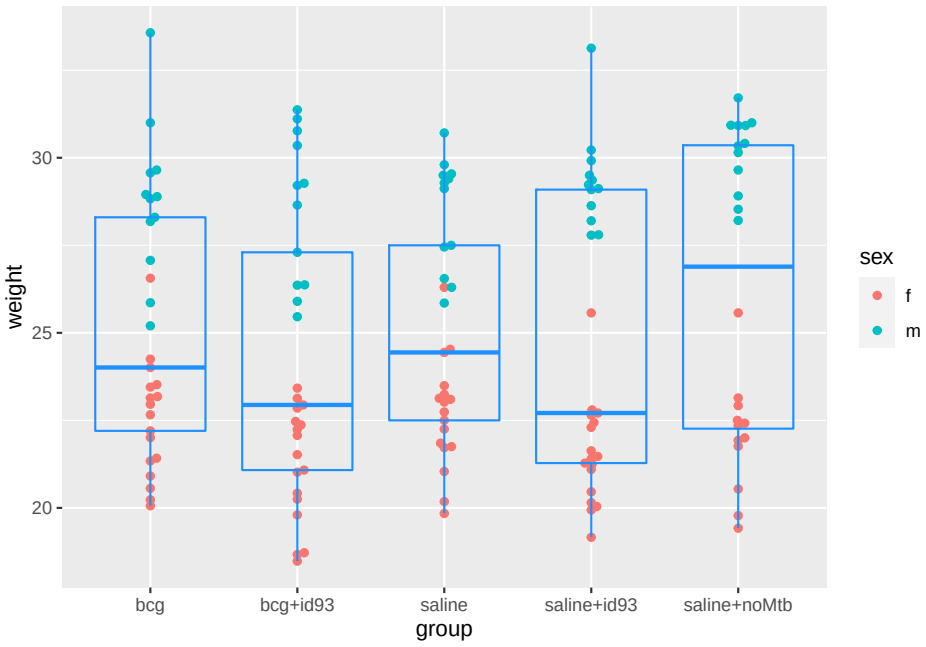



```
ex %>%
  ggplot(aes(x = date_collected, y = weight, color = who_collected)) + geom_point()
```




```
library(ggbeeswarm)
```

```
ex %>%
  filter(date_collected == last(date_collected)) %>%
  ggplot(aes(x = group, y = weight)) +
  geom_beeswarm(aes(color = sex)) +
  geom_boxplot(fill = NA, color = "dodgerblue")
```



4.0.4 Can also use rio to read in the data, more streamlined

```
dataset <- data$before_vaccination %>%
 select("sex", "mouse_number", "weight", "existing_cage_number", "group")
```

```
# combining columns mouse_number and cage_number

dataset$mouse_id <- paste(dataset$mouse_number, "-", dataset$cage_number)
```

4.0.6 Body weight over time graph and statistics

4.0.7 Weight loss over time graph and statistics

4.0.8 Weight vs CFU

4.0.9 Weight vs ELISA results

4.0.10 Weight vs lesion burden

Chapter 5

Colony forming units to determine bacterial counts

5.1 Data description

The data are collected in a spreadsheet with multiple sheets. The first sheet (named "[x]") is used to record some metadata for the experiment, while the following sheets are used to record CFUs counts from the plates used for samples from each organ, with one sheet per organ. For example, if you plated data from both the lung and spleen, there would be three sheets in the file: one with the metadata, one with the plate counts for the lung, and one with the plate counts for the spleen.

The metadata sheet is used to record information about the overall process of plating the data. Values from this sheet will be used in calculating the bacterial load in the original sample based on the CFU counts. This spreadsheet includes the following columns:

- `organ`: Include one row for each organ that was plated in the experiment. You should name the organ all in lowercase (e.g., "lung", "spleen"). You should use the same name to also name the sheet that records data for that organ for example, if you have rows in the metadata sheet for "lung" and "spleen", then you should have two other sheets in the file, one sheet named "lung" and one named "spleen", which you'll use to store the plate counts for each of those organs.
- `prop_resuspended`: In this column, give the proportion of that organ that was plated. For example, if you plated half the lung, then in the "lung" row of this spread sheet, you should put 0.5 in the `prop_resuspended` column.
- `total_resuspended_uL`: This column contains an original volume of tissue

homogenate. For example, raw lung tissue is homogenized in 500 uL of PBS in a tube containing metal beads.

- `og_aliquot_uL`: 100 uL of th total_resuspended slurry would be considered an original aliquot and is used to peform serial dilutions.
- `dilution_factor`: Amount of the original stock solution that is present in the total solution, after dilution(s)
- `plated_uL`: Amount of suspension + diluent plated on section of solid agar

5.2 Read in data

```
library(readxl)
library(dplyr)
library(purrr)
library(tidyr)
library(stringr)
library(tidyverse)
library(gridExtra)
library(ggplot2)
library(ggpubr)

#Replace w/ path to CFU sheet
path <- c("DATA/Copy of baa_cfu_sheet.xlsx")

sheet_names <- excel_sheets(path)
sheet_names <- sheet_names[!sheet_names %in% c("metadata")]

merged_data <- list()

for(i in 1:length(sheet_names)){

  data <- read_excel(path, sheet = sheet_names[i]) %>%
    mutate(organ = paste0(sheet_names[i]))

  data <- data %>%
    #mutate(missing_col = NA) %>%
    mutate_if(is.double, as.numeric) %>%
    mutate_if(is.numeric, as.character) %>%
    pivot_longer(starts_with("dil_"), names_to = "dilution",
                 values_to = "CFUs") %>%
    mutate(dilution = str_extract(dilution, "[0-9]+"),
           dilution = as.numeric(dilution))
```

```
  merged_data[[i]] <- data


}

all_data <- bind_rows(merged_data, .id = "column_label") %>%
    select(-column_label)

head(merged_data)
```

```
## [[1]]
## # A tibble: 342 x 8
##    count_date           who_plated who_counted groups mouse organ dilution CFUs
##    <chr>                <chr>      <chr>       <chr>  <chr> <chr>    <dbl> <chr>
##  1 "\"February 21 2022~ BK         BK          group~ A     lung         0 TNTC
##  2 "\"February 21 2022~ BK         BK          group~ A     lung         1 TNTC
##  3 "\"February 21 2022~ BK         BK          group~ A     lung         2 TNTC
##  4 "\"February 21 2022~ BK         BK          group~ A     lung         3 53
##  5 "\"February 21 2022~ BK         BK          group~ A     lung         4 9
##  6 "\"February 21 2022~ BK         BK          group~ A     lung         5 4
##  7 "\"February 21 2022~ BK         BK          group~ A     lung         6 2
##  8 "\"February 21 2022~ BK         BK          group~ A     lung         7 1
##  9 "\"February 21 2022~ BK         BK          group~ A     lung         8 0
## 10 "\"February 21 2022~ BK         BK          group~ B     lung         0 TNTC
## # ... with 332 more rows
##
## [[2]]
## # A tibble: 112 x 8
##    count_date          who_plated who_counted groups  mouse organ dilution CFUs
##    <chr>               <chr>      <chr>       <chr>   <chr> <chr>    <dbl> <chr>
##  1 "\"April 25 2022\"" JR         JR          group_1 A     sple~        0 TNTC
##  2 "\"April 25 2022\"" JR         JR          group_1 A     sple~        1 TNTC
##  3 "\"April 25 2022\"" JR         JR          group_1 A     sple~        2 53
##  4 "\"April 25 2022\"" JR         JR          group_1 A     sple~        3 9
##  5 "\"April 25 2022\"" JR         JR          group_1 A     sple~        4 4
##  6 "\"April 25 2022\"" JR         JR          group_1 A     sple~        5 2
##  7 "\"April 25 2022\"" JR         JR          group_1 A     sple~        6 1
##  8 "\"April 25 2022\"" JR         JR          group_1 A     sple~        7 0
##  9 "\"April 25 2022\"" JR         JR          group_1 B     sple~        0 TNTC
## 10 "\"April 25 2022\"" JR         JR          group_1 B     sple~        1 TNTC
## # ... with 102 more rows
```

```
head(all_data)
```

```
## # A tibble: 6 x 8
##   count_date           who_plated who_counted groups mouse organ dilution CFUs
```

```
##   <chr>                 <chr>      <chr>       <chr>  <chr> <chr>    <dbl> <chr>
## 1 "\"February 21 2022\~ BK         BK          group~ A     lung         0 TNTC
## 2 "\"February 21 2022\~ BK         BK          group~ A     lung         1 TNTC
## 3 "\"February 21 2022\~ BK         BK          group~ A     lung         2 TNTC
## 4 "\"February 21 2022\~ BK         BK          group~ A     lung         3 53
## 5 "\"February 21 2022\~ BK         BK          group~ A     lung         4 9
## 6 "\"February 21 2022\~ BK         BK          group~ A     lung         5 4
```

5.3 Example one

5.4 Exploratory analysis and quality checks

5.5 Exploratory analysis

Dimensions of input data:

Based on the input data, data were collected for the following organ or organs:

The following number of mice were included for each:

The following number of replicates were recorded at each count date for each experimental group:

The following number of dilutions and dilution level were recorded for each organ:

People who plated and collected the data. Date or dates of counting:

Based on the input data, the plates included in these data were counted by the following person or persons: Based on the input data, the plates included in these data were counted on the following date or dates:

```
all_data %>%
  select(organ, who_plated, who_counted, count_date) %>%
  distinct()
```

```
## # A tibble: 3 x 4
##   organ  who_plated who_counted count_date
##   <chr>  <chr>      <chr>       <chr>
## 1 lung   BK         BK          "\"February 21 2022\""
## 2 lung   BK         BK          "\"April 18 2022\""
## 3 spleen JR         JR          "\"April 25 2022\""
```

```
head(all_data)
```

```
## # A tibble: 6 x 8
##   count_date            who_plated who_counted groups mouse organ dilution CFUs
##   <chr>                 <chr>      <chr>       <chr>  <chr> <chr>    <dbl> <chr>
## 1 "\"February 21 2022\~ BK         BK          group~ A     lung         0 TNTC
```

```
## 2 "\"February 21 2022\~ BK         BK          group~ A     lung         1 TNTC
## 3 "\"February 21 2022\~ BK         BK          group~ A     lung         2 TNTC
## 4 "\"February 21 2022\~ BK         BK          group~ A     lung         3 53
## 5 "\"February 21 2022\~ BK         BK          group~ A     lung         4 9
## 6 "\"February 21 2022\~ BK         BK          group~ A     lung         5 4
```

Distribution of CFUs at each dilution:

Here's a plot that shows how many plates were too numerous to count at each dilution level:

Here is a plot that shows how the CFU counts were distributed by dilution level in the data:

5.6 Identify a good dilution for each sample

```
# Make all_data into tidy data and filter for CFUs between 10-75

tidy_cfu_data <- all_data %>%
  mutate(dilution = str_extract(dilution, "[0-9]+"),
         dilution = as.numeric(dilution)) %>%
  filter((CFUs >= 5 & CFUs <= 95) | groups == "control") %>%
  mutate(CFUs = as.numeric(CFUs))


head(tidy_cfu_data)
```

```
## # A tibble: 6 x 8
##   count_date            who_plated who_counted groups mouse organ dilution  CFUs
##   <chr>                 <chr>      <chr>       <chr>  <chr> <chr>    <dbl> <dbl>
## 1 "\"February 21 2022\~ BK         BK          group~ A     lung         3    53
## 2 "\"February 21 2022\~ BK         BK          group~ A     lung         4     9
## 3 "\"February 21 2022\~ BK         BK          group~ C     lung         5     8
## 4 "\"February 21 2022\~ BK         BK          group~ D     lung         3    53
## 5 "\"February 21 2022\~ BK         BK          group~ A     lung         2    92
## 6 "\"February 21 2022\~ BK         BK          group~ A     lung         4     7
```

5.7 Calculate CFUs from best dilution/Estimate bacterial load for each sample based on good dilution

```
# Calculating CFU/ml for every qualifying replicate between 10-75 CFUs. Column binding by organ n
meta <- read_excel(path, sheet = "metadata")
```

```
tidy_cfu_meta_joined <- inner_join(meta, tidy_cfu_data) %>%
  group_by(groups) %>%
  mutate(CFUs_per_ml = (CFUs * (dilution_factor^dilution) *
                          (total_resuspension_mL/volume_plated_ul) * 1000)) %>%
  select(organ, count_date, who_plated, who_counted, groups,  mouse, dilution,
         CFUs, CFUs_per_ml) %>%
  ungroup()
```

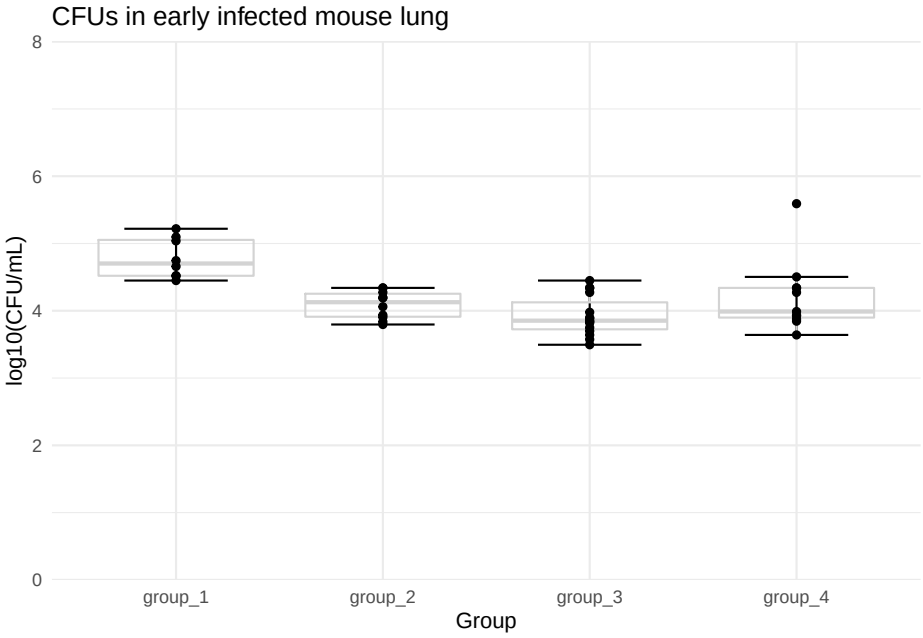
```
## Joining, by = "organ"
```

```
head(tidy_cfu_meta_joined)
```

```
## # A tibble: 6 x 9
##   organ count_date            who_plated who_counted groups mouse dilution  CFUs
##   <chr> <chr>                 <chr>      <chr>       <chr>  <chr>    <dbl> <dbl>
## 1 lung  "\"February 21 2022\~ BK         BK          group~ A            3    53
## 2 lung  "\"February 21 2022\~ BK         BK          group~ A            4     9
## 3 lung  "\"February 21 2022\~ BK         BK          group~ C            5     8
## 4 lung  "\"February 21 2022\~ BK         BK          group~ D            3    53
## 5 lung  "\"February 21 2022\~ BK         BK          group~ A            2    92
## 6 lung  "\"February 21 2022\~ BK         BK          group~ A            4     7
## # ... with 1 more variable: CFUs_per_ml <dbl>
```

5.8 Create initial report information for these data

```
tidy_lung_cfu_plot <- tidy_cfu_meta_joined %>%
  filter(organ == "lung") %>%
  mutate(group = fct_relevel(groups, "group_1", "group_2", "group_3", "group_4")) %>%
  ggplot(aes(x = groups, y = log10(CFUs_per_ml), fill = groups))+
  stat_boxplot( aes(x = groups, y = log10(CFUs_per_ml)),
    geom='errorbar', linetype=1, width=0.5)+
  geom_boxplot(aes(group = groups), fill = NA, show.legend = FALSE, color = "lightgrey"
  geom_point(show.legend = FALSE)+
  labs(title = paste0("CFUs in early infected mouse lung"), x = "Group", y = "log10(CFU
       color = "Group")+
  guides(shape = "none")+
  theme_minimal()+
  stat_compare_means(label = "p.signif", method = "t.test", ref.group = "group_1") +
  scale_y_continuous(expand = c(0, 0), limits = c(0, 8))

tidy_lung_cfu_plot
```



5.9 Sample ANOVA

```
cfu_stats <- tidy_cfu_meta_joined %>%
  group_by(organ) %>%
  nest() %>%
  mutate(aov_result = map(data, ~aov(CFUs_per_ml ~ groups, data = .x)),
         tukey_result = map(aov_result, TukeyHSD),
         tidy_tukey = map(tukey_result, broom::tidy)) %>%
  unnest(tidy_tukey, .drop = TRUE) %>%
  separate(contrast, into = c("contrast1", "contrast2"), sep = "-") %>%
  select(-data, -aov_result, -tukey_result, -term, -null.value)# %>%
```

```
## Warning: The `.drop` argument of `unnest()` is deprecated as of tidyr 1.0.0.
## All list-columns are now preserved.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

```
  # filter(adj.p.value <= 0.05)

cfu_stats
```

```
## # A tibble: 9 x 7
## # Groups:   organ [2]
##   organ  contrast1 contrast2 estimate conf.low conf.high adj.p.value
##   <chr>  <chr>     <chr>        <dbl>    <dbl>     <dbl>       <dbl>
```

```
## 1 lung   group_2   group_1    -60953. -138742.    16836.      0.171
## 2 lung   group_3   group_1    -63903. -135699.     7893.      0.0963
## 3 lung   group_4   group_1    -26214. -102416.    49987.      0.793
## 4 lung   group_3   group_2     -2950.  -69900.    64000.      0.999
## 5 lung   group_4   group_2     34739.  -36915.   106393.      0.569
## 6 lung   group_4   group_3     37689.  -27410.   102787.      0.417
## 7 spleen group_2   group_1     -6565   -13529.      399.      0.0656
## 8 spleen group_3   group_1     -7310   -13341.    -1279.      0.0178
## 9 spleen group_3   group_2      -745.   -6776.     5286.      0.943
```

5.10 Save processed data to database

5.11 Example two

Chapter 6

Enzyme-linked immunosorbest assay (ELISA)

ELISA is a standard molecular biology assay for detecting and quantifying a variety of compounds, including peptides, proteins, and antibodies in a sample. The sample could be serum, plasma, or bronchoalveolar lavage fluid (BALF).

6.1 Importance of ELISA

An antigen-specific reaction in the host results in the production of antibodies, which are proteins found in the blood. In the event of an infectious disease, it aids in the detection of antibodies in the body. ELISA is distinguishable from other antibody-assays in that it produces quantifiable findings and separates non-specific from specific interactions by serial binding to solid surfaces, which is often a polystyrene multi-well plate.

In IMPAc-TB project, it is crucial to evaluate the if the vaccine is eliciting humoral immunity and generating antibodies against vaccine antigen. ELISA will be used to determine the presence of Immunoglobulin (Ig) IgG, IgA, and IgM in the serum different time points post-vaccination.

6.1.1 Principle of ELISA

ELISA is based on the principle of antigen-antibody interaction. An antigen must be immobilized on a solid surface and then complexed with an enzyme-linked antibody in an ELISA. The conjugated enzyme's activity is evaluated

by incubating it with a substrate to yield a quantifiable result, which enables detection. There are four basic steps of ELISA:

1. Coating multiwell plate with antigen/antibody: This step depends on what we want to detect the sample. If we need to evaluate the the presence of antibody, the plate will be coated with the antigen, and vice versa. To coat the plate, a fixed concentration of antigen (protein) is added to a 96 well high-binding plate (charged plate). Plate is incubated over night with the antigen at 4 degree celsius (as proteins are temperature sensitive) so that antigens are completely bound to the well.

2. Blocking: It is possible that not each and every site of the well is coated with the targeted antigen, and there could be uncovered areas. It is important to block those empty spaces so that primary antibody (which we will add to the next step) binds to these spaces and give us false positive results. For this, microplate well surface-binding sites are blocked with an unrelated protein or other substance.Most common blocking agents are bovine serum albumin, skim milk, and casein. One of the best blocking agents is to use the serum from the organism in which your secondary (detection antibody) is raised. For example, if the secondary antibody is raised in goat, then we can use goat serum as a blocking agent.

3. Probing: Probing is the step where we add sample containing antibodies that we want to detect. This will be the primary antibody. If the antibodies against the antigen (which we have coated) are present in the sample, it will bind to the antigen with high affinity.

4. Washing: After the incubation of sample containing primary antibody, the wells are washed so that any unbound antibody is washed away. Washing solution contains phosphate buffer saline + 0.05% tween-20 (a mild detergent). 0.05% tween-20 washes away all the non-specific interactions as those are not strong, but keeps all the specific interaction as those are strong and cannot be detached with mild detergent.

5. Detection: To detect the presence of antibody-antigen complex, a secondary antibody labelled with an enzyme (usually horseradish peroxidase) is added to the wells, incubated and washed.

6. Signal Measurement: Finally to detect "if" and "how much" of the antibody is present, a chromogenic substrate (like 3,3',5,5'-Tetramethylbenzidine) is added to the wells, which can be cleaved the the enzyme that is tagged to the secondary antibody. The color compund is formed after the addition of the substrate, which is directly proportional to the amount of antibody present in the sample. The plate is read on a plate reader, where color is converted to numbers.

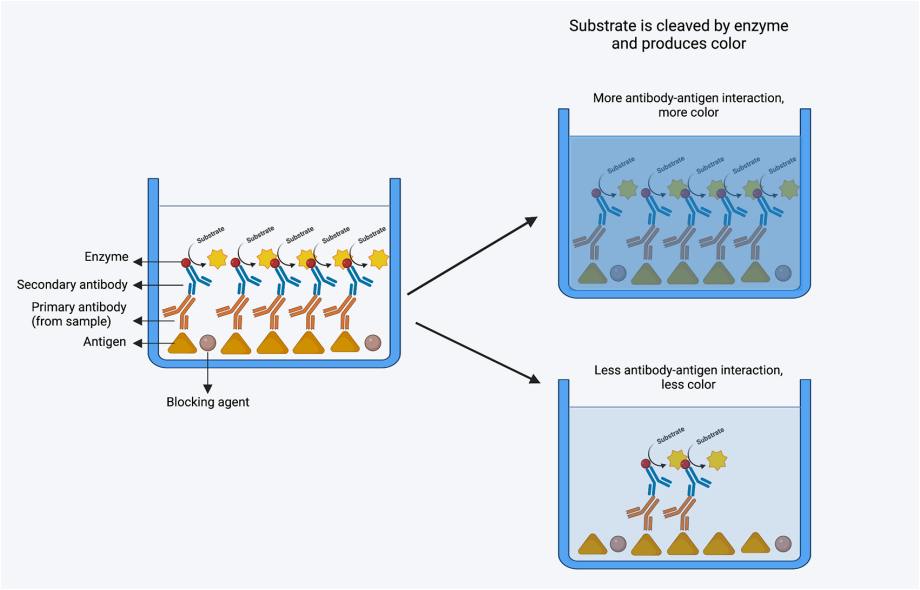


Figure 6.1: A caption

6.1.2 Loading libraries

```
library(readxl)
library(tidyverse)
library(minpack.lm)
library(broom)
library(purrr)
library(ggbeeswarm)
```

6.2 ELISA data analysis

Analysis of ELISA data is the most important part of the ELISA experiment. ELISA data can be analyzed in different ways based on how the data is acquired. There are a a few examples of the type of ELISA data :

1. With standard curve: ELISA can be used to determine the concentrations of the antigen and antibody. This type of ELISA data usually have a standard curve with different concentrations of the known analyte and the concentration in the sample is determined by extrapolating the unknown values in the curve. This type of assay is straightforward, easy to interpret and are more robust.

2. Without standard curve: Usually vaccine studies involve investigating the presence of high-affinity (and novel) antibodies against the vaccine antigens. Therefore, plotting a standard curve is not feasible as there is no previous infor-

mation available for antibody concentration or type of antibody. Also, because antibody response to a vaccine will differ depending on the individual, it is not practical to generate a calibration curve from which absolute concentrations can be extrapolated. For this type of ELISA, quantification of the antibody titers is performed using serial dilutions of the test samples, and analysis can be performed using the following three methods (Hartman et al., 2018):

1. Fitting sigmoid model
2. Endpoint titer method 3: Absorbance summation method

Let's have a look at these methods, how we can apply these methods in our data, and R-based packages that we can utilize to perform this analysis.

6.3 1. Curve fitting model:

The curve in ELISA data represents a plot of known concentrations versus their corresponding signal responses. The typical range of these calibration curves is one to two orders of magnitude on the response axis and two or more orders of magnitude on the concentration axis. The real curve of each assay could be easily identified if an infinite number of concentration dilutions with an infinite number of repetitions could be tested. The correct curve must be approximated from a relatively small number of noisy points, though, because there are a finite number of dilutions that may be performed. To estimate the dose-response relationship between standard dilutions, a method of interpolating between standards is required because there cannot be a standard at every concentration. This process is typically performed using a mathematical function or regression to approximate the true shape of the curve. A curve model is the name given to this approximating function, which commonly uses two or more parameters to describe a family of curves, and are then adjusted in order to find the curve from the family of curves that best fits the assay data.

Three qualities should be included in a good curve fitting model. 1. The true curve's shape must be accurately approximated by the curve model. If the curve model does not accomplish this, there is no way to adjust for this component of the total error that results from a lack of fit. 2. In order to get concentration estimates with minimal inaccuracy, a decent curve model must be able to average away as much of the random variation as is practical. 3. A successful curve model must be capable of accurately predicting concentration values for points between the anchor points of the standard dilutions.

6.3.1 How do we perform curve fitting model

There are two major steps in performing curve fitting model for non-linear data like ELISA: 1. Finding the initial starting estimates of the parameters 2. locating the optimal solution in a region of the initial estimates

We have presented an example below where we have performed a 8-10 point

serial dilution of our sample and fitted a 4 parameter curve model.

6.3.2 An example of the curve fitting model

6.3.2.1 Read in the data

This information comes from the 2018 study conducted by Hartman et al. Hartman et al. analyzed the ELISA data in their study utilizing fitted sigmoid analysis, end point titer, and absorbance summation. We utilized this information to determine whether our formulas and calculations provide the same outcomes and values as theirs.

```
elisa_example_data <- read_excel("DATA/example_elisa_data.xlsx")
```

6.3.2.2 Tidying the data

We next performed tidying the data and make it in a format so that we can plot a sigmoid curve with that.

```
# Divide dilution column into two seoparate columns

elisa_example_data <- separate(elisa_example_data,
                               col = "dilution",
                               into = c("numerator", "denominator"),
                               sep = "\\/")

# Convert the tabke from character to numeric
elisa_example_data <- elisa_example_data %>%
  mutate_if(is.character, as.numeric)


elisa_example_data$dilution <-
  elisa_example_data$numerator/elisa_example_data$denominator

elisa_example_data <- elisa_example_data %>%
  mutate(log_dilution = log(dilution, base = 3))

head(elisa_example_data)
```

```
## # A tibble: 6 x 5
##   numerator denominator absorbance dilution log_dilution
##       <dbl>       <dbl>      <dbl>    <dbl>        <dbl>
## 1         1          30       4    0.0333          -3.10
## 2         1          90       3.73 0.0111          -4.10
## 3         1         270       2.34 0.00370         -5.10
## 4         1         810       1.1  0.00123         -6.10
## 5         1        2430       0.51 0.000412        -7.10
## 6         1        7290       0.22 0.000137        -8.10
```

6.3.2.3 Create function for curve fitting model

We next created the curve fitting model function by using nlsLM function from "minpack.lm" package. The purpose of nlslm is to minimize the sum square of the vector returned by the function fn, by a modification of the Levenberg-Marquardt algorithm. In the early 1960s, the Levenberg-Marquardt algorithm was developed to address nonlinear least squares problems. Through a series of well-chosen updates to model parameter values, Levenberg-Marquardt algorithm lower the sum of the squares of the errors between the model function and the data points.

```
mod_1 <- nlsLM(absorbance ~
                ((a-d)/(1+(log_dilution/c)^b)) + d,
data = elisa_example_data,
start = list (a = 4, d= 0, c = -5, b = 1))

# a = maximum absorbance
# d = minimum absobance
# c = point of maximum growth
# b = slope at c

mod_1
```

```
## Nonlinear regression model
##   model: absorbance ~ ((a - d)/(1 + (log_dilution/c)^b)) + d
##    data: elisa_example_data
##        a        d        c        b
##  4.12406  0.04532 -5.31056  7.62972
##  residual sum-of-squares: 0.02221
##
## Number of iterations to convergence: 9
## Achieved convergence tolerance: 1.49e-08
```

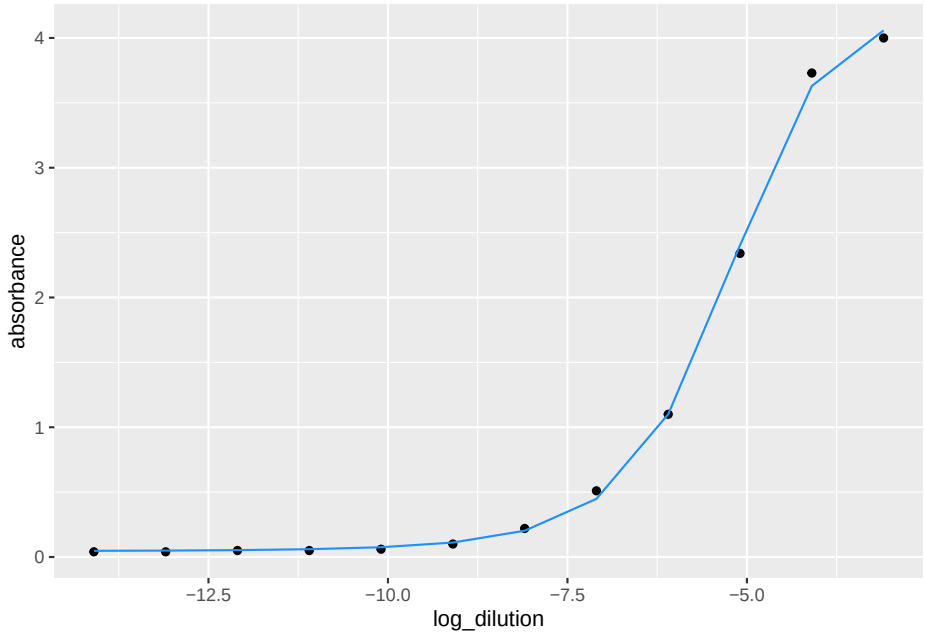
```
summary(mod_1)
```

```
##
## Formula: absorbance ~ ((a - d)/(1 + (log_dilution/c)^b)) + d
##
## Parameters:
##   Estimate Std. Error  t value Pr(>|t|)
## a  4.12406    0.05820   70.860 1.75e-12 ***
## d  0.04532    0.02268    1.998   0.0808 .
## c -5.31056    0.03933 -135.037 1.01e-14 ***
## b  7.62972    0.35854   21.280 2.50e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.05269 on 8 degrees of freedom
##
## Number of iterations to convergence: 9
## Achieved convergence tolerance: 1.49e-08
```

```
tidy_params <- mod_1 %>% tidy()

a <- tidy_params$estimate[tidy_params$term == "a"]
b <- tidy_params$estimate[tidy_params$term == "b"]
c <- tidy_params$estimate[tidy_params$term == "c"]
d <- tidy_params$estimate[tidy_params$term == "d"]

elisa_example_data <- elisa_example_data %>%
  mutate(fitted = predict(mod_1))

elisa_example_data <- elisa_example_data %>%
  mutate(fitted = predict(mod_1))
```

```
elisa_example_data %>%
  ggplot(aes(x = log_dilution, y = absorbance)) +
  geom_point() +
  geom_line(aes(y=fitted), color = "dodgerblue")
```



6.4 2. Endpoint titer method

The endpoint titer approach chooses an absorbance value just above the background noise (or the lower asymptotic level). **The highest dilution with an absorbance greater than this predetermined value is the endpoint titer.** This method is based on the assumption that a sample with a higher protein concentration will require a higher dilution factor to achieve an absorbance just above the level of background noise.

6.4.1 Create an endpoint titer function and apply it to the output of the fitted sigmoid model values.

```
endpoint_titer <- c * (((a - d) / (0.2 - d)) - 1) ^ (1 / b)

summary(endpoint_titer)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -8.113  -8.113  -8.113  -8.113  -8.113  -8.113
```

```
endpoint_titer
```

```
## [1] -8.113285
```

6.4.2 Other methods to analyze ELISA data

6.4.2.1 Absorption summation

6.4.2.2 Area under the curve

In this model of data analysis, we sum all the absorbance values from each sample to obtain one value. This value is termed as absorption summation (AS). Using the above data, the AS will be calculated as below:

```
AS = 0.04 + 0.04 + 0.05 + 0.05 + 0.06 +
  0.1 + 0.22 + 0.51 + 1.1 + 2.34 + 3.73 + 4.0

AS
```

```
## [1] 12.24
```

6.5 Apply the fitting sigmoid model and endpoint titer function in our dataset

The presented data is from a mouse study. In this data, presence of IgG antibody has been evaluated against receptor binding domain (RBD) of SARS-CoV-2 virus in two different groups of mice. We need to elucidate which group has higher concentration of the antibodies.

6.5.0.1 Read in the data

```
elisa_data <- read_excel("DATA/elisa_data_serial_dilution.xlsx")
```

6.5.0.2 Tidy the data

```
elisa_data <- pivot_longer(data = elisa_data,
                           cols = "Mouse_1":"Mouse_5",
                           names_to = "mouse_id",
                           values_to = "absorbance")

head(elisa_data)
```

```
## # A tibble: 6 x 4
##   Groups  Dilution mouse_id absorbance
##   <chr>   <chr>    <chr>         <dbl>
## 1 Group 1 1/50     Mouse_1         4.1
## 2 Group 1 1/50     Mouse_2         3.9
## 3 Group 1 1/50     Mouse_3         4.3
## 4 Group 1 1/50     Mouse_4         4.2
## 5 Group 1 1/50     Mouse_5         4
## 6 Group 1 1/100    Mouse_1         3.9
```

```
# separate dilution column and convert it to log2

elisa_data <- separate(elisa_data,
                       col = "Dilution",
                       into = c("numerator",
                                "denomenator"),
                       sep = "\\/")

elisa_data <- elisa_data %>%
  transform(numerator = as.numeric(numerator),
            denomenator = as.numeric(denomenator))

elisa_data <- elisa_data %>%
  mutate(dilution =
           elisa_data$numerator/elisa_data$denomenator)

elisa_data <- elisa_data %>%
  mutate(log_dilution = log2(dilution))

head(elisa_data)
```

```
##    Groups numerator denomenator mouse_id absorbance dilution log_dilution
## 1 Group 1         1          50  Mouse_1        4.1     0.02    -5.643856
## 2 Group 1         1          50  Mouse_2        3.9     0.02    -5.643856
## 3 Group 1         1          50  Mouse_3        4.3     0.02    -5.643856
## 4 Group 1         1          50  Mouse_4        4.2     0.02    -5.643856
## 5 Group 1         1          50  Mouse_5        4.0     0.02    -5.643856
## 6 Group 1         1         100  Mouse_1        3.9     0.01    -6.643856
```

```
elisa_data_df <- elisa_data %>%
  group_by(Groups, mouse_id) %>%
  summarize(log_dilution = log_dilution,
            absorbance = absorbance)
```

6.5.0.2.1 converting data into dataframe

```
## `summarise()` has grouped output by 'Groups', 'mouse_id'. You can override using
## the `.groups` argument.
```

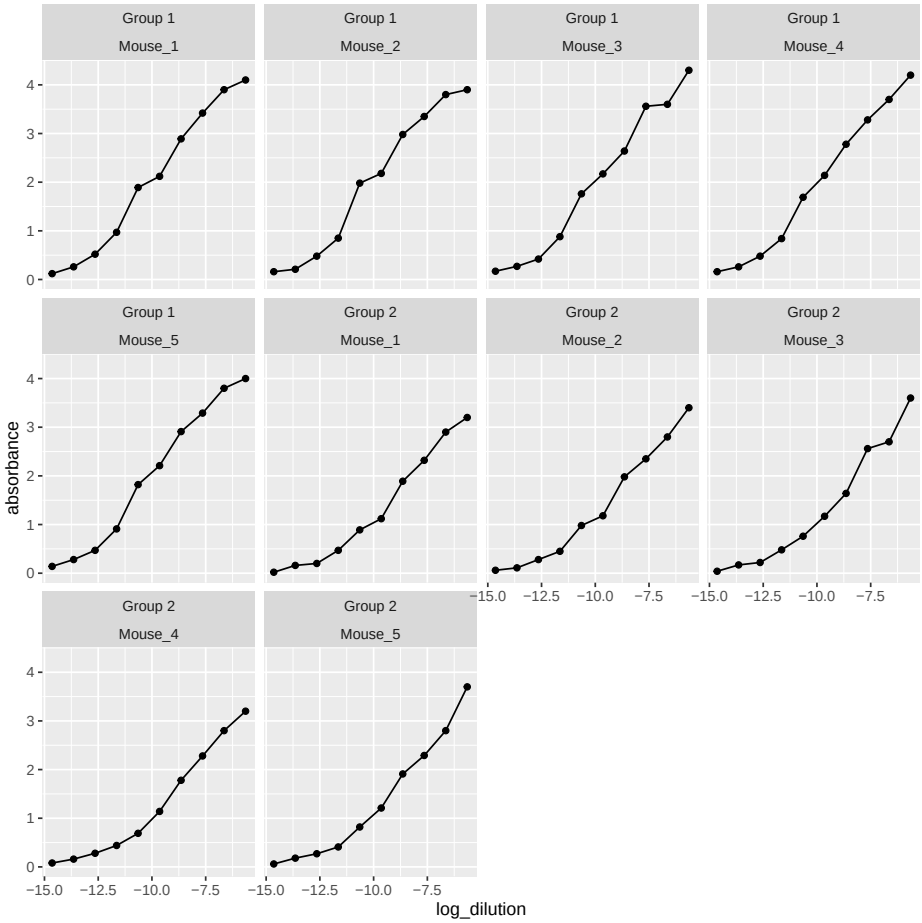
```
elisa_data_nested <- elisa_data %>%
  group_by(Groups, mouse_id) %>%
  nest()

head(elisa_data_nested)
```

```
## # A tibble: 6 x 3
```

```
## # Groups:   Groups, mouse_id [6]
##   Groups  mouse_id data
##   <chr>   <chr>    <list>
## 1 Group 1 Mouse_1  <tibble [10 x 5]>
## 2 Group 1 Mouse_2  <tibble [10 x 5]>
## 3 Group 1 Mouse_3  <tibble [10 x 5]>
## 4 Group 1 Mouse_4  <tibble [10 x 5]>
## 5 Group 1 Mouse_5  <tibble [10 x 5]>
## 6 Group 2 Mouse_1  <tibble [10 x 5]>
```

```
elisa_data %>%
  ggplot(aes(x = log_dilution, y = absorbance)) +
  geom_point() +
  geom_line() +
  facet_wrap(Groups ~ mouse_id)
```

6.5.0.2.2 plot the curves to evaluate the a, d, c, and b



Based on the curve, the values are:

a = 4, d = 0 c = 2 b = 1

6.5.1 Creating a function for fitting model

```
fitted_model_elisa <- function(df_elisa,
                               start_a, start_d,
                               start_c, start_b) {
  mod_1 <- nlsLM(absorbance ~
                   ((a-d)/(1+(log_dilution/c)^b)) + d,
data = df_elisa,
start = list(a = start_a, d = start_d, c = start_c, b = start_b))
  return(mod_1)
}
```

6.5.1.1 Fitting the model into the dataset

```
fitted_model_elisa(elisa_data_nested$data[[1]],
                   start_a = 4,
                   start_d = 0,
                   start_c = -8,
                   start_b = 1)
```

```
## Nonlinear regression model
##   model: absorbance ~ ((a - d)/(1 + (log_dilution/c)^b)) + d
##    data: df_elisa
##        a        d        c        b
##   4.3070  -0.6009 -10.2577   5.2893
##  residual sum-of-squares: 0.1199
##
## Number of iterations to convergence: 7
## Achieved convergence tolerance: 1.49e-08
```

6.5.1.2 Apply the fitted model function to the whole dataframe

```
elisa_fitted_data <- elisa_data_nested %>%
  mutate(fitted_data =
           purrr::map(data, ~
                        fitted_model_elisa(.x,start_a = 4,
                                           start_d = 0,
                                           start_c = -8,
                                           start_b = 1)))

head(elisa_fitted_data)
```

```
## # A tibble: 6 x 4
## # Groups:   Groups, mouse_id [6]
##   Groups  mouse_id data              fitted_data
##   <chr>   <chr>    <list>            <list>
## 1 Group 1 Mouse_1  <tibble [10 x 5]> <nls>
## 2 Group 1 Mouse_2  <tibble [10 x 5]> <nls>
## 3 Group 1 Mouse_3  <tibble [10 x 5]> <nls>
## 4 Group 1 Mouse_4  <tibble [10 x 5]> <nls>
## 5 Group 1 Mouse_5  <tibble [10 x 5]> <nls>
## 6 Group 2 Mouse_1  <tibble [10 x 5]> <nls>
```

6.5.1.3 Take out the summary of the data

```
elisa_fitted_data_summary <- elisa_fitted_data %>%
  mutate(elisa_fitted_data_summary =
           purrr::map(fitted_data, broom::glance))
```

```
unnested <- elisa_fitted_data_summary %>%
unnest(elisa_fitted_data_summary) %>%
ungroup() %>%
dplyr::select(Groups, mouse_id, fitted_data)

unnested$fitted_data[[1]]
```

```
## Nonlinear regression model
##   model: absorbance ~ ((a - d)/(1 + (log_dilution/c)^b)) + d
##    data: df_elisa
##        a        d        c        b
##   4.3070  -0.6009 -10.2577   5.2893
##  residual sum-of-squares: 0.1199
##
## Number of iterations to convergence: 7
## Achieved convergence tolerance: 1.49e-08
```

6.6 Create function of Fitted model and endpoint titer, where the output of the fitted model data will be the input of the endpoint titer

```
# Fitted model function
fitted_model_elisa <- function(df_elisa,
                               start_a,
                               start_d,
                               start_c,
                               start_b) {
  mod_1 <- nlsLM(absorbance ~
                   ((a-d)/(1+(log_dilution/c)^b)) + d,
data = df_elisa,
start = list(a = start_a, d = start_d, c = start_c, b = start_b))
  return(mod_1)
}

# Endpoint titer function

endpoint_titer_elisa <- function(fitted_data, back_value) {
  tidy_fitted <- broom::tidy(fitted_data)
  est_a <- tidy_fitted$estimate[tidy_fitted$term == "a"]
  est_b <- tidy_fitted$estimate[tidy_fitted$term == "b"]
  est_c <- tidy_fitted$estimate[tidy_fitted$term == "c"]
```

```
  est_d <- tidy_fitted$estimate[tidy_fitted$term == "d"]
  endpoint_titer <- est_c * (((est_a - est_d) / (back_value - est_d)) - 1) ^ (1 / est_b)
  return(endpoint_titer)
}
```

6.6.0.1 Apply the fitted model fuction into the nested data and use the output of the fitted data as the input for endpoint titer value evaluation

```
elisa_data_with_fit_model <- elisa_data_nested %>%
  mutate(fitted_data = purrr::map(data,
                                  ~ fitted_model_elisa(.x, start_a = 4,
                                                       start_d = 0,
                                                       start_c = -8,
                                                       start_b = 1)))

head(elisa_data_with_fit_model)
```

6.6.0.1.1 Run fitted model on the data

```
## # A tibble: 6 x 4
## # Groups:   Groups, mouse_id [6]
##   Groups  mouse_id data              fitted_data
##   <chr>   <chr>    <list>            <list>
## 1 Group 1 Mouse_1  <tibble [10 x 5]> <nls>
## 2 Group 1 Mouse_2  <tibble [10 x 5]> <nls>
## 3 Group 1 Mouse_3  <tibble [10 x 5]> <nls>
## 4 Group 1 Mouse_4  <tibble [10 x 5]> <nls>
## 5 Group 1 Mouse_5  <tibble [10 x 5]> <nls>
## 6 Group 2 Mouse_1  <tibble [10 x 5]> <nls>
```

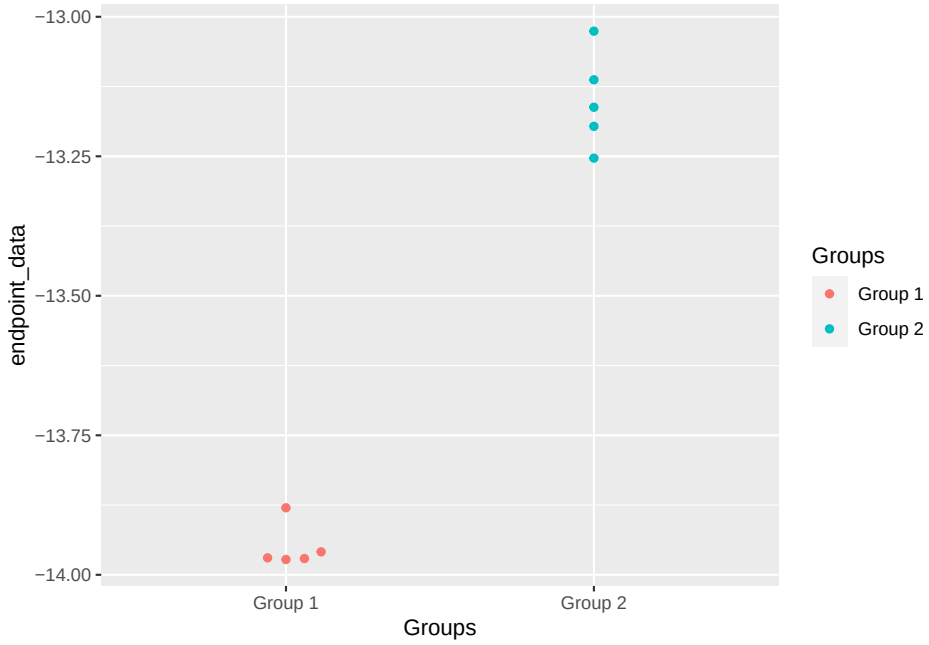
```
elisa_data_with_endpoint_titer <- elisa_data_with_fit_model %>%
  mutate(endpoint_data =
           purrr::map(fitted_data,
                      ~ endpoint_titer_elisa(.x, back_value = 0.2)))
```

6.6.0.1.2 Taking output of the fitted model function and into endpoint titer function

6.6.0.2 Plot the endpoint titer data for the two groups

```
elisa_data_with_endpoint_titer$endpoint_data=
  as.numeric(elisa_data_with_endpoint_titer$endpoint_data)
```

```
elisa_data_with_endpoint_titer %>%
  ggplot(aes(x = Groups, y = endpoint_data, color = Groups)) +
 geom_beeswarm(cex = 3)
```



6.6.0.3 Perform statistical analysis on the data

```
elisa_data_stats <- t.test(endpoint_data ~ Groups,
                           data = elisa_data_with_endpoint_titer)

elisa_data_stats %>%
  tidy()
```

```
## # A tibble: 1 x 10
##   estimate estimate1 estimate2 statistic    p.value parameter conf.low conf.high
##      <dbl>     <dbl>     <dbl>     <dbl>      <dbl>     <dbl>    <dbl>     <dbl>
## 1   -0.800     -14.0     -13.2     -18.8 0.00000268      5.63   -0.906    -0.695
## # ... with 2 more variables: method <chr>, alternative <chr>
```

6.6.0.4 Statistical data analysis for more than two groups

Chapter 7

ELISA data processing

We read ELISA plate in a 96 well plate using a plate reader. The plate reader generates the data in form of number in an excel sheet. We have created this pipeline/worksheet to bring out the information from the excl sheet to a tidy format in which the above created fitted model and endpoint titer functions can be applied.

7.0.0.1 Read in the first dataset

Below is the example ELISA data that has came straight out of the plate reader. This data is arranged in a 96-well plate format and contains Optical Density (OD) values.

```
elisa_raw_data <- read_excel("DATA/elisa_s1_07-25-20.xlsx",
                             sheet = "S1", col_names = FALSE,
                             range = "B2:M9")
```

```
## New names:
## * `` -> ...1
## * `` -> ...2
## * `` -> ...3
## * `` -> ...4
## * `` -> ...5
## * ...
```

```
head(elisa_raw_data)
```

```
## # A tibble: 6 x 12
##   ...1          ...2  ...3 ...4   ...5  ...6 ...7   ...8  ...9 ...10 ...11 ...12
##   <chr>        <dbl> <dbl> <chr> <dbl> <dbl> <chr> <dbl> <dbl> <chr> <dbl> <dbl>
## 1 5.199999999~ 0.05  0.069 6.3E~ 0.061 0.122 0.16~ 0.145 0.135 6.80~ 0.053 0.05
## 2 7.900000000~ 0.098 0.069 6.80~ 0.115 0.202 5.89~ 0.134 0.069 0.106 0.05  0.075
```

```
## 3 8.899999999~ 0.133 0.119 OVRF~ 3.87  2.32  OVRF~ 3.85  2.12  OVRF~ 3.21  1.02
## 4 OVRFLW       3.46  1.16  OVRF~ 3.80  2.36  OVRF~ 3.70  1.49  OVRF~ 3.68  1.63
## 5 3.815999999~ 1.82  0.446 3.89~ 3.42  1.13  OVRF~ 2.33  0.608 OVRF~ 3.41  1.10
## 6 OVRFLW       3.69  1.43  OVRF~ 3.66  1.27  3.839 1.74  0.444 2.49~ 0.637 0.704
```

7.0.0.2 Tidy dataset 1

It is important to clean the data and arrange it in a format on which we can apply formulas and functions.

```
# Convert all columns to numeric

elisa_raw_data_numeric <- elisa_raw_data %>%
  mutate_if(is.character, as.numeric)
```

```
## Warning in mask$eval_all_mutate(quo): NAs introduced by coercion

## Warning in mask$eval_all_mutate(quo): NAs introduced by coercion

## Warning in mask$eval_all_mutate(quo): NAs introduced by coercion

## Warning in mask$eval_all_mutate(quo): NAs introduced by coercion
```

```
# pivot longer the data

elisa_raw_data_tidy <- pivot_longer(data = elisa_raw_data_numeric, cols = "...1":"...1

# remove "..." from the first column

elisa_raw_data_tidy$well_id <- str_replace(elisa_raw_data_tidy$well_id, "...", "")

# Add new column to the data_frame

elisa_raw_data_tidy_new <- elisa_raw_data_tidy %>%
  mutate(name = rep(LETTERS[1:8], each = 12))

elisa_raw_data_tidy_new <- elisa_raw_data_tidy_new %>%
  mutate(well_id = paste0(name, well_id)) %>%
  select(-name)

head(elisa_raw_data_tidy_new)
```

```
## # A tibble: 6 x 2
##   well_id od_450nm
##   <chr>      <dbl>
## 1 A1         0.052
## 2 A2         0.05
## 3 A3         0.069
```

```
## 4 A4          0.063
## 5 A5          0.061
## 6 A6          0.122
```

7.0.0.3 Read in the second data set

The second dataset contains the information such as groups, mouse id, and dilutions for the respective wells of the 96 well plate for the dataset-1.

```
elisa_label_data <- read_excel("DATA/elisa_s1_07-25-20.xlsx",
                               sheet = "S1", col_names = FALSE,
                               range = "Q2:AB9")
```

```
## New names:
## * `` -> ...1
## * `` -> ...2
## * `` -> ...3
## * `` -> ...4
## * `` -> ...5
## * ...
```

```
head(elisa_label_data)
```

```
## # A tibble: 6 x 12
##   ...1        ...2   ...3  ...4  ...5  ...6  ...7  ...8  ...9  ...10 ...11 ...12
##   <chr>       <chr>  <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
## 1 blank       secon~ naïv~ 1A-1~ 1A-1~ 1A-1~ 1A-2~ 1A-2~ 1A-2~ 1A-3~ 1A-3~ 1A-3~
## 2 1A-4 (1/250 1A-4 ~ 1A-4~ 1B-1~ 1B-1~ 1B-1~ 1B-2~ 1B-2~ 1B-2~ 1B-3~ 1B-3~ 1B-3~
## 3 1B-4 (1/250 1B-4 ~ 1B-4~ 2A-1~ 2A-1~ 2A-1~ 2A-2~ 2A-2~ 2A-2~ 2A-3~ 2A-3~ 2A-3~
## 4 2B-1 (1/250 2B-1 ~ 2B-1~ 2B-2~ 2B-2~ 2B-2~ 2B-3~ 2B-3~ 2B-3~ 2B-4~ 2B-4~ 2B-4~
## 5 3A-1 (1/250 3A-1 ~ 3A-1~ 3A-2~ 3A-2~ 3A-2~ 3A-3~ 3A-3~ 3A-3~ 3A-4~ 3A-4~ 3A-4~
## 6 3B-1 (1/250 3B-1 ~ 3B-1~ 3B-2~ 3B-2~ 3B-2~ 3B-3~ 3B-3~ 3B-3~ 3B-4~ 3B-4~ 3B-4~
```

7.0.0.4 Tidy dataset-2

```
# pivot longer the data

elisa_label_data_tidy <- pivot_longer(data = elisa_label_data,
                                      cols = "...1":"...12",
                                      names_to = "well_id",
                                      values_to = "information")

# remove "..." from the first column

elisa_label_data_tidy$well_id <- str_replace(elisa_label_data_tidy$well_id, "...", "")

# Add new column to the data_frame
```

```
elisa_label_data_tidy_new <- elisa_label_data_tidy %>%
  mutate(name = rep(LETTERS[1:8], each = 12))

elisa_label_data_tidy_new <- elisa_label_data_tidy_new %>%
  mutate(well_id = paste0(name, well_id)) %>%
  select(-name)

head(elisa_label_data_tidy_new)
```

```
## # A tibble: 6 x 2
##   well_id information
##   <chr>   <chr>
## 1 A1      blank
## 2 A2      secondary
## 3 A3      naïve (1/250)
## 4 A4      1A-1 (1/250
## 5 A5      1A-1 (1/1250
## 6 A6      1A-1 (1/6250
```

7.0.0.5 Merge dataset-1 (with OD information) with dataset-2 (with respective data information)

To create a complete full dataset with Groups, mouse-id, dilutions, and OD, we merged the dataset-1 and dataset-2 together. We also cleaned the data set so that mouse-ID and dilution columns are separate and have their own columns.

```
#Merge the two datasets

elisa_data = elisa_raw_data_tidy_new %>% inner_join(elisa_label_data_tidy_new,
                                                    by="well_id")

head(elisa_data)
```

```
## # A tibble: 6 x 3
##   well_id od_450nm information
##   <chr>      <dbl> <chr>
## 1 A1         0.052 blank
## 2 A2         0.05  secondary
## 3 A3         0.069 naïve (1/250)
## 4 A4         0.063 1A-1 (1/250
## 5 A5         0.061 1A-1 (1/1250
## 6 A6         0.122 1A-1 (1/6250
```

```
### Separate the information table into sample ID and dilution columns

tidy_elisa_data <- separate(elisa_data, col = "information",
```

```
                            into = c("sample_id", "dilution"),
                            sep = "\\(")
```

```
## Warning: Expected 2 pieces. Missing pieces filled with `NA` in 2 rows [1, 2].
```

```
head(tidy_elisa_data)
```

```
## # A tibble: 6 x 4
##   well_id od_450nm sample_id   dilution
##   <chr>      <dbl> <chr>       <chr>
## 1 A1         0.052 "blank"     <NA>
## 2 A2         0.05  "secondary" <NA>
## 3 A3         0.069 "naïve "    1/250)
## 4 A4         0.063 "1A-1 "     1/250
## 5 A5         0.061 "1A-1 "     1/1250
## 6 A6         0.122 "1A-1 "     1/6250
```

```
tidy_elisa_data <- tidy_elisa_data %>%
  mutate(dilution = str_extract(dilution, "(/)[0-9]+"),
         dilution = str_replace(dilution, "/", ""),
         dilution = as.numeric(dilution))

tidy_elisa_data <- tidy_elisa_data %>%
  select(well_id, sample_id, dilution, od_450nm)

head(tidy_elisa_data)
```

```
## # A tibble: 6 x 4
##   well_id sample_id   dilution od_450nm
##   <chr>   <chr>          <dbl>    <dbl>
## 1 A1      "blank"           NA    0.052
## 2 A2      "secondary"       NA    0.05
## 3 A3      "naïve "         250    0.069
## 4 A4      "1A-1 "          250    0.063
## 5 A5      "1A-1 "         1250    0.061
## 6 A6      "1A-1 "         6250    0.122
```

Chapter 8

Flow cytometry

Flow cytometry data can be quantified in many different ways and with different techniques. For the purpose of these data analyses, manual gating has been achieved in FlowJo and cell frequencies and populations exported as a `.csv` file. This `.csv` file is the primary input for this R pipeline which aims to output box plots for each gated cell population.

This example data set is from an innate response study whcih investigated the immune response in the lungs during the first 28 days of infection.

8.1 Loading packages

```
library(readxl)
library(ggplot2)
library(RColorBrewer)
library(dplyr)
library(tidyverse)
library(scales)
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:purrr':
##
##     discard
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
library(stringr)
library(tidyr)
library(knitr)
library(forcats)
library(broom)
library(ggfortify)
library(stats)
library(ggpubr)
library(grDevices)
library(rstatix)
```

```
##
## Attaching package: 'rstatix'
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
library(writexl)
```

8.2 panel information

```
# antibody_panel <- read_excel
```

8.3 Loading data

```
Df <- read_excel("DATA/innate_normalizedto45.xlsx", sheet = "CD3CD11b No Day 14")

marker_legend <- read_excel("DATA/marker legend.xlsx")

# Remove Freq of Parent columns
Df1 <- Df %>%
  select(-matches("Parent"))
# Remove "Leukocytes/LIVE/Single Cells/" from col names
names(Df1) <- str_remove(names(Df1), "Leukocytes/LIVE/Single Cells/")

Df1 <- Df1 %>%
  rename_all(funs(str_replace(., "\\|.+", "")))# Remove "|Freq of..." from col names
```

```
## Warning: `funs()` was deprecated in dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
```

```
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
```

```
Df1 <- Df1 %>%
  rename_all(funs(str_replace_all(., "\\/Q[:digit:]+\\:", ""))) %>%
  rename_all(funs(str_replace(., "\\/", " "))) %>%
  rename_all(funs(str_replace(., "\\,", " "))) %>%
  rename_all(funs(str_replace(., "\\ \\,", " ")))

# str_extract_all(names(Df1), "[:alpha:]+[:digit:]+[\\+\\-]")
#
#
#
#
#
#
# marker_select <- function(col_title) {
#   marker_df <- str_detect(names(DATA1), "[\\+\\-]")
#   return(marker_df)
# }
```

8.4 Making the data tidy for plotting

```
tidy_Df1 <- pivot_longer(data = Df1, cols =  starts_with("CD45+"), names_to = "cell_types", value

tidy_Df1 <- tidy_Df1 %>%
  separate(col = "SAMPLE", into = c("day", "replicate"))


tidy_Df1 %>%
  select(cell_types) %>%
  unique()
```

```
## # A tibble: 128 x 1
##    cell_types
##    <chr>
##  1 "CD45+ "
##  2 "CD45+ CD3-   CD11b+ "
##  3 "CD45+ CD3-   CD11b+ CD25+ "
```

```
##  4 "CD45+ CD3-   CD11b+ CD103+ "
##  5 "CD45+ CD3-   CD11b+ gamma_delta "
##  6 "CD45+ CD3-   CD11b+ NKp46+ "
##  7 "CD45+ CD3-   CD11b+ CD11c+  CD64- "
##  8 "CD45+ CD3-   CD11b+ CD11c-  CD64- "
##  9 "CD45+ CD3-   CD11b+ CD86-  CD64+ "
## 10 "CD45+ CD3-   CD11b+ CD86+  CD64+ "
## # ... with 118 more rows
```

```
tidy_Df1 <- tidy_Df1 %>%
  filter(percentage_of_CD45 > 0.005)

head(tidy_Df1, n=10)
```

```
## # A tibble: 10 x 4
##    day   replicate cell_types                          percentage_of_CD45
##    <chr> <chr>     <chr>                                            <dbl>
##  1 CNT   1         "CD45+ "                                          82.9
##  2 CNT   1         "CD45+ CD3-   CD11b+ "                            29.3
##  3 CNT   1         "CD45+ CD3-   CD11b+ CD25+ "                       0.88
##  4 CNT   1         "CD45+ CD3-   CD11b+ CD103+ "                      0.75
##  5 CNT   1         "CD45+ CD3-   CD11b+ gamma_delta "                 4.77
##  6 CNT   1         "CD45+ CD3-   CD11b+ NKp46+ "                      7.3
##  7 CNT   1         "CD45+ CD3-   CD11b+ CD11c+  CD64- "               3.65
##  8 CNT   1         "CD45+ CD3-   CD11b+ CD11c-  CD64- "              24.3
##  9 CNT   1         "CD45+ CD3-   CD11b+ CD86-  CD64+ "               0.43
## 10 CNT   1         "CD45+ CD3-   CD11b+ CD86+  CD64+ "               0.85
```

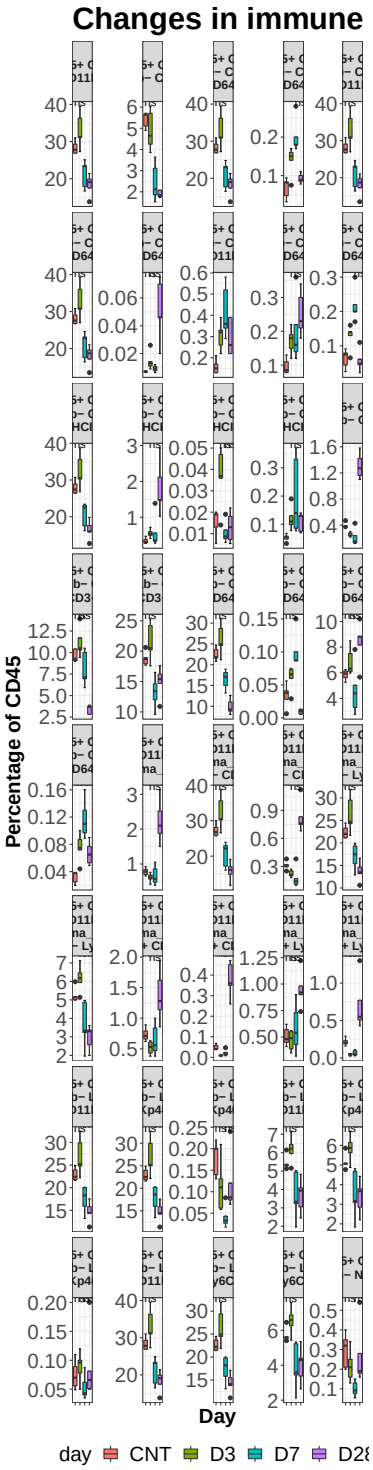
```
# Select CD3 & CD11b populations and create new data frames

CD3pos_CD11bneg <- tidy_Df1 %>%
  filter(str_detect(cell_types, "CD3\\+ + CD11b\\-"))

CD3neg_CD11bpos <- tidy_Df1 %>%
  filter(str_detect(cell_types, "CD3\\- + CD11b\\+"))

CD3neg_CD11bneg <- tidy_Df1 %>%
  filter(str_detect(cell_types, "CD3\\- + CD11b\\-"))
```

8.5 boxplot

```
CD3pos_CD11bneg_bar_plot <- CD3pos_CD11bneg %>%
mutate(day = fct_relevel(day,
            "CNT", "D3", "D7",
            "D28")) %>%
```

```
  ggplot(aes(x = day, y = percentage_of_CD45, fill= day)) +
  stat_boxplot( aes(day, percentage_of_CD45),
    geom='errorbar', linetype=1, width=0.5)+
  geom_boxplot(aes(day, percentage_of_CD45)) +
  facet_wrap(~cell_types, scale = "free_y", labeller = label_wrap_gen(width=15), ncol = 5, nrow =
  theme_bw() +
  theme(axis.text.x = element_blank(), axis.text.y = element_text(size = 20),
        axis.title.x = element_text(size = 20, face = "bold"),
        axis.title.y = element_text(size = 20, face = "bold"),
        legend.text = element_text(size = 20),
        legend.title = element_text(size = 20),
        plot.title = element_text(color="black", size=30, face="bold")) +
  labs (y="Percentage of CD45", x = "Day") +
  theme(strip.text = element_text(size=12, face = "bold")) + theme(legend.position="bottom") +
  ggtitle("Changes in immune cell populations (lung) CD3+ CD11b-") +
  stat_compare_means(label = "p.signif", method = "t.test",
                     ref.group = "CNT")



CD3neg_CD11bpos_bar_plot <- CD3neg_CD11bpos %>%
mutate(day = fct_relevel(day,
             "CNT", "D3", "D7",
             "D28")) %>%
  ggplot(aes(x = day, y = percentage_of_CD45, fill= day)) +
  stat_boxplot( aes(day, percentage_of_CD45),
    geom='errorbar', linetype=1, width=0.5)+
  geom_boxplot( aes(day, percentage_of_CD45)) +
  facet_wrap(~cell_types, scale = "free_y", labeller = label_wrap_gen(width=15), ncol = 5, nrow =
  theme_bw() +
  theme(axis.text.x = element_blank(), axis.text.y = element_text(size = 20),
        axis.title.x = element_text(size = 20, face = "bold"),
        axis.title.y = element_text(size = 20, face = "bold"),
        legend.text = element_text(size = 20),
        legend.title = element_text(size = 20),
        plot.title = element_text(color="black", size=30, face="bold")) +
  labs (y="Percentage of CD45", x = "Day") +
  theme(strip.text = element_text(size=12, face = "bold")) + theme(legend.position="bottom") +
  ggtitle("Changes in immune cell populations (lung) CD3- CD11b+") +
  stat_compare_means(label = "p.signif", method = "t.test",
                     ref.group = "CNT")



CD3neg_CD11bneg_bar_plot <- CD3neg_CD11bneg %>%
```

```
mutate(day = fct_relevel(day,
            "CNT", "D3", "D7",
            "D28")) %>%
  ggplot(aes(x = day, y = percentage_of_CD45, fill= day)) +
  stat_boxplot( aes(day, percentage_of_CD45),
    geom='errorbar', linetype=1, width=0.5)+
  geom_boxplot( aes(day, percentage_of_CD45)) +
  facet_wrap(~cell_types, scale = "free_y", labeller = label_wrap_gen(width=15), ncol =
  theme_bw() +
  theme(axis.text.x = element_blank(), axis.text.y = element_text(size = 20),
        axis.title.x = element_text(size = 20, face = "bold"),
        axis.title.y = element_text(size = 20, face = "bold"),
        legend.text = element_text(size = 20),
        legend.title = element_text(size = 20),
        plot.title = element_text(color="black", size=30, face="bold")) +
  labs (y="Percentage of CD45", x = "Day") +
  theme(strip.text = element_text(size=12, face = "bold")) + theme(legend.position="bot
  ggtitle("Changes in immune cell populations (lung) CD3- CD11b-") +
  stat_compare_means(label = "p.signif", method = "t.test",
                     ref.group = "CNT")

CD3pos_CD11bneg_bar_plot
```


Changes in immune
Percentage of CD45
Day
day CNT D3 D7 D28

```
# CD3neg_CD11bpos_bar_plot
# CD3neg_CD11bneg_bar_plot
```

Chapter 9

Pathology

Chapter 10

Proteomics

For proteomics data, we will be getting data that have already been collected and pre-processed by another part of the team. The following shows an example of the type of data we will get as an input:

```
library(tidyverse)

prot_a <- read_csv("DATA/Transition Results_CCTSI_A.csv")
```

```
## Rows: 3393 Columns: 18
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (7): Peptide, Protein, Replicate, Fragment Ion, Ratio Dot Product, Tota...
## dbl (11): Precursor Mz, Precursor Charge, Product Mz, Product Charge, Retent...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
prot_a
```

```
## # A tibble: 3,393 x 18
##    Peptide     Protein Replicate   `Precursor Mz` `Precursor Char~` `Product Mz`
##    <chr>       <chr>   <chr>                <dbl>             <dbl>        <dbl>
##  1 QELDEISTNIR Cfp10   091322_LT1            659.                 2        1061.
##  2 QELDEISTNIR Cfp10   091322_LT2            659.                 2        1061.
##  3 QELDEISTNIR Cfp10   091322_LT3            659.                 2        1061.
##  4 QELDEISTNIR Cfp10   091322_LT4            659.                 2        1061.
##  5 QELDEISTNIR Cfp10   091322_LT5            659.                 2        1061.
##  6 QELDEISTNIR Cfp10   091322_LT6            659.                 2        1061.
##  7 QELDEISTNIR Cfp10   091322_LT7            659.                 2        1061.
##  8 QELDEISTNIR Cfp10   091322_LT8            659.                 2        1061.
##  9 QELDEISTNIR Cfp10   091322_LT10           659.                 2        1061.
```

```
## 10 QELDEISTNIR Cfp10   091322_LT11           659.                 2        1061.
## # ... with 3,383 more rows, and 12 more variables: `Product Charge` <dbl>,
## #   `Fragment Ion` <chr>, `Retention Time` <dbl>, Area <dbl>, Background <dbl>,
## #   `Peak Rank` <dbl>, `Ratio Dot Product` <chr>,
## #   `Total Area Normalized` <chr>, `Total Area Ratio` <chr>,
## #   `Library Dot Product` <dbl>, RatioLightToHeavy <dbl>,
## #   DotProductLightToHeavy <dbl>
```

These data include the following columns:

- `Peptide`: A short string of peptides that are being measured
- `Protein`: The protein that those peptides come from
- `Replicate`: An identifier for the sample that the measurement was taken on
- `Precursor Mz`, `Precursor Charge`, `Product Mz`, `Product Charge`, `Fragment Ion`, `Retention Time`: Measurements that help in identifying the peptide that is being measured (?)
- `Area`:
- `Background`:
- `Peak Rank`:
- `Ratio Dot Product`:
- `Total Area Normalized`:
- `Total Area Ratio`
- `Library Dot Product`:
- `RatioLightToHeavy`:
- `DotProductLightToHeavy`:

[More about how these data were pre-processed. Softwarei: Skyline]

Here are all the unique replicates in this file:

```
prot_a %>%
  pull(Replicate) %>%
  unique()
```

```
##  [1] "091322_LT1"  "091322_LT2"  "091322_LT3"  "091322_LT4"  "091322_LT5"
##  [6] "091322_LT6"  "091322_LT7"  "091322_LT8"  "091322_LT10" "091322_LT11"
## [11] "091322_LT12" "091322_LT13" "091322_LT14" "091322_H1"   "091322_H2"
## [16] "091322_H3"   "091322_H4"   "091322_H5"   "091322_H6"   "091322_H7"
## [21] "091322_H8"   "091322_H9"   "091322_H10"  "091322_H11"  "091322_H12"
## [26] "091322_H13"  "091322_H14"  "091322_TB1"  "091322_TB2"  "091322_TB3"
## [31] "091322_TB4"  "091322_TB5"  "091322_TB6"  "091322_TB7"  "091322_TB8"
## [36] "091322_TB9"  "091322_TB10" "091322_TB11" "091322_TB12"
```

The three groups in this data are labeled with “LT”, “H”, and “TB” somewhere in the identifier. We can create a new column in the dataset that pulls out this treatment group information:

```
prot_a <- prot_a %>%
  mutate(treatment_group = str_extract(Replicate, "[A-Z]+"))

prot_a %>%
  filter(Peptide == first(Peptide)) %>%
  group_by(treatment_group) %>%
  count()
```

```
## # A tibble: 3 x 2
## # Groups:   treatment_group [3]
##   treatment_group     n
##   <chr>           <int>
## 1 H                 140
## 2 LT                130
## 3 TB                120
```

```
prot_a %>%
  filter(Peptide == first(Peptide) &
           Replicate == first(Replicate))
```

```
## # A tibble: 10 x 19
##    Peptide     Protein Replicate  `Precursor Mz` `Precursor Charge` `Product Mz`
##    <chr>       <chr>   <chr>               <dbl>              <dbl>        <dbl>
##  1 QELDEISTNIR Cfp10   091322_LT1           659.                  2        1061.
##  2 QELDEISTNIR Cfp10   091322_LT1           659.                  2         832.
##  3 QELDEISTNIR Cfp10   091322_LT1           659.                  2         703.
##  4 QELDEISTNIR Cfp10   091322_LT1           659.                  2         590.
##  5 QELDEISTNIR Cfp10   091322_LT1           659.                  2         503.
##  6 QELDEISTNIR Cfp10   091322_LT1           664.                  2        1071.
##  7 QELDEISTNIR Cfp10   091322_LT1           664.                  2         842.
##  8 QELDEISTNIR Cfp10   091322_LT1           664.                  2         713.
##  9 QELDEISTNIR Cfp10   091322_LT1           664.                  2         600.
## 10 QELDEISTNIR Cfp10   091322_LT1           664.                  2         513.
## # ... with 13 more variables: `Product Charge` <dbl>, `Fragment Ion` <chr>,
## #   `Retention Time` <dbl>, Area <dbl>, Background <dbl>, `Peak Rank` <dbl>,
## #   `Ratio Dot Product` <chr>, `Total Area Normalized` <chr>,
## #   `Total Area Ratio` <chr>, `Library Dot Product` <dbl>,
## #   RatioLightToHeavy <dbl>, DotProductLightToHeavy <dbl>,
## #   treatment_group <chr>
```

```
prot_a %>%
  pull(Protein) %>%
  unique()
```

```
## [1] "Cfp10"                  "acpM"                   "Ag85A"
## [4] "MtbH37Rv|Rv3841|BfrB"   "MtbH37Rv|Rv1837c|GlcB"  "MtbH37Rv|Rv3418c|GroES"
## [7] "MtbH37Rv|Rv3248c|SahH"  "MtbH37Rv|Rv2031c|hspX"
```

- Cfp10
- acpM
- Ag85A
- MtbH37Rv|Rv3841|BfrB
- MtbH37Rv|Rv1837c|GlcB
- MtbH37Rv|Rv3418c|GroES
- MtbH37Rv|Rv3248c|SahH
- MtbH37Rv|Rv2031c|hspX

10.1 Download Microsoft365R package

```
install.packages("Microsoft365R")
install.packages("AzureGraph")
install.packages("RODBC")
```

10.2 Load the package

```
library(Microsoft365R)
library(AzureGraph)
library(AzureAuth)
```

10.3 SEt up teams via R

```
my_team <- get_team("CVMBS T_cell_mask_study",
             app = "04b07795-8ddb-461a-bbee-02f9e1bf7b46")

my_team$list_channels()
```

Bibliography

Baazim, H., Antonio-Herrera, L., and Bergthaler, A. (2022). The interplay of immunology and cachexia in infection and cancer. *Nature Reviews Immunology*, 22(5):309–321.

Baazim, H., Schweiger, M., Moschinger, M., Xu, H., Scherer, T., Popa, A., Gallage, S., Ali, A., Khamina, K., Kosack, L., et al. (2019). Cd8+ t cells induce cachexia during chronic viral infection. *Nature immunology*, 20(6):701–710.

Hartman, H., Wang, Y., Schroeder Jr, H. W., and Cui, X. (2018). Absorbance summation: a novel approach for analyzing high-throughput elisa data in the absence of a standard. *PloS one*, 13(6):e0198528.

Segueni, N., Tritto, E., Bourigault, M.-L., Rose, S., Erard, F., Le Bert, M., Jacobs, M., Di Padova, F., Stiehl, D. P., Moulin, P., et al. (2016). Controlled mycobacterium tuberculosis infection in mice under treatment with anti-il-17a or il-17f antibodies, in contrast to tnf-alpha neutralization. *Scientific Reports*, 6(1):1–17.

Smith, C. M., Baker, R. E., Proulx, M. K., Mishra, B. B., Long, J. E., Park, S. W., Lee, H.-N., Kiritsy, M. C., Bellerose, M. M., Olive, A. J., et al. (2022). Host-pathogen genetic interactions underlie tuberculosis susceptibility in genetically diverse mice. *Elife*, 11:e74419.