# MECH481A6: Engineering Data Analysis in R

Chapter 10 Homework: Measurement

Ethan Rutledge

13 December, 2023

## Load packages

```r
# load packages for current session
library(tidyverse)
library(gridExtra)
library(MASS)
library(lubridate)
```

## Chapter 10 Homework

This homework will give you practice at working with a measurement dataset: `airlift_mass_repeatability.csv`. This data set represents repeated measures of "blank" air sampling filters.

A couple notes to consider when reporting answers in response to questions. The microbalance used to make these measurements reads out to the nearest microgram ($\mu g$), which is 0.000001 $g$ or 0.001 $mg$. Thus, be careful when reporting descriptive statistics so as not to overstate your **precision**. Use the `round()` function to avoid reporting more than 0.1 $\mu g$ of precision (or 0.0001 $mg$). Here is some example code that uses the `across()` function from `dplyr::` to round numeric output to just four digits (appropriate for $mg$ units in this exercise):

`dplyr::mutate(across(.cols = where(is.numeric), .fns = round, 3))`

## Question 1

Import the `airlift_mass_repeatability.csv` file into a data frame called `blanks` and perform the following data wrangling in a single pipe:

- retain only the first 3 columns of data;
- rename the columns with the names `date`, `id`, and `mass_mg`;
- convert the `date` column vector into a date class object using `lubridate::`
- convert the `id` variable to a class `factor` (this can be accomplished using `base::as.factor()` or `purrr::as_factor()`)
- create a new column vector named `mass_mg` by rescaling the `mass_g` data (i.e., convert *g* to *mg* by multiplying `mass_g` by 1000)

```
blanks <- read_csv("../data/AIRLIFT_mass_repeatability.csv") %>%
  dplyr::select(1:3) %>%
  rename(mass_g = 'Mass (g)', date = 'Date', id = 'Filter ID') %>%
  mutate(date = dmy(date), id = as.factor(id)) %>%
  mutate(mass_mg = mass_g * 1000)
```

## Question 2:

2a. Are there any NAs present in the data frame?
2b. How many unique filter IDs are present in this data frame?
2c. How many samples are present for each filter ID? Hint: look up the `dplyr::count()` function.
2d. Over how long of a period were these blank measurements made? Hint: this can be done in base R with a `max() - min()` or with `lubridate::interval() %>% as.duration()`.

```
na_chk <- any(is.na(blanks))
na_chk
```

```
## [1] FALSE
```

```
levels(blanks$id)
```

```
## [1] "41666" "41667" "41668" "41669" "41671"
```

```
sample_cnt <- blanks %>%
  count(id)
sample_cnt
```

```
## # A tibble: 5 x 2
##   id        n
##   <fct> <int>
## 1 41666    78
## 2 41667    78
## 3 41668    78
## 4 41669    76
## 5 41671    78
```

```r
max(blanks$date) - min(blanks$date)
```

```
## Time difference of 35 days
```

# Answers:

2a - No 2b - 5 2c - see tibble 2d - 35 days

## Question 3

Group the `blanks` data frame by `id` and calculate mean, median, and standard deviations for each filter id. Hint: use `group_by() %>% summarise()` to do this efficiently.

```r
blanks_sum <- blanks%>%
  group_by(id) %>%
  summarise(mean_val = mean(mass_mg), median_val = median(mass_mg), sd_val = sd(mass_mg))

blanks_sum
```

```
## # A tibble: 5 x 4
##   id    mean_val median_val   sd_val
##   <fct>    <dbl>      <dbl>    <dbl>
## 1 41666     98.3       98.3 0.000767
## 2 41667     95.5       95.5 0.000534
## 3 41668     98.0       98.0 0.000834
## 4 41669     97.8       97.8 0.00113
## 5 41671     97.6       97.6 0.000834
```

## Question 4

Calculate the limit of detection (LOD) for this measurement method. Note: you will need to calculate standard deviations for each filter `id` (as done in question 3) and then estimate LOD from $LOD = 3 \cdot \sigma_b$ where $\sigma_b$ is calculated for each filter `id`.

```r
blanks_LOD <- blanks_sum%>%
  mutate(LOD = 3 * sd_val)

blanks_LOD
```

```
## # A tibble: 5 x 5
##   id    mean_val median_val   sd_val     LOD
##   <fct>    <dbl>      <dbl>    <dbl>   <dbl>
## 1 41666     98.3       98.3 0.000767 0.00230
## 2 41667     95.5       95.5 0.000534 0.00160
## 3 41668     98.0       98.0 0.000834 0.00250
## 4 41669     97.8       97.8 0.00113  0.00340
## 5 41671     97.6       97.6 0.000834 0.00250
```