sheet 1 exercise one solutions

2023-10-09

Introduction

In the first lecture of this course we got introduced to a study about the **adverse effects of valporic acid on neurological functionality in embryos during pregnancy** (Kurg. A.K. et. a, (2013)). Valporic acid is typically used for treatment of epilepsy, migraines, headaches and even specific types of mental illnesses as a mood stabilizer. However, according to the World Health Organization, it is strongly advised to not take this medication during pregnancy as it bears a high risk of developmental disorders (more information, like if you have literally nothing else to do here). The study has been done in vitro (in petri dishes) on (multiplied) embryonic stem cells. As the **response-data** we therefore use **gen-expressions, ranging from 2 to 14**. Beside the **6 negative controls** we have seen **7 response-dosages on 3 samples each**.

The data-set has been reduced quite a bit: out of the 10.000 gen expressions, only 500 were chosen based on their variance across **all** samples.

Requirements

As part of this analysis, we are using tidyverse (especially ggplot).

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(gridExtra)
```

```
## Warning: Paket 'gridExtra' wurde unter R Version 4.3.1 erstellt
```

```
##
## Attache Paket: 'gridExtra'
##
## Das folgende Objekt ist maskiert 'package:dplyr':
##
##     combine
```

Importing the data

To import the data we use the `load()` function. Make sure to have the right working directory, when importing the data.

```r
# use the setwd(...) function if necessary.
load("./data/VPAData-Random.Rda")
```

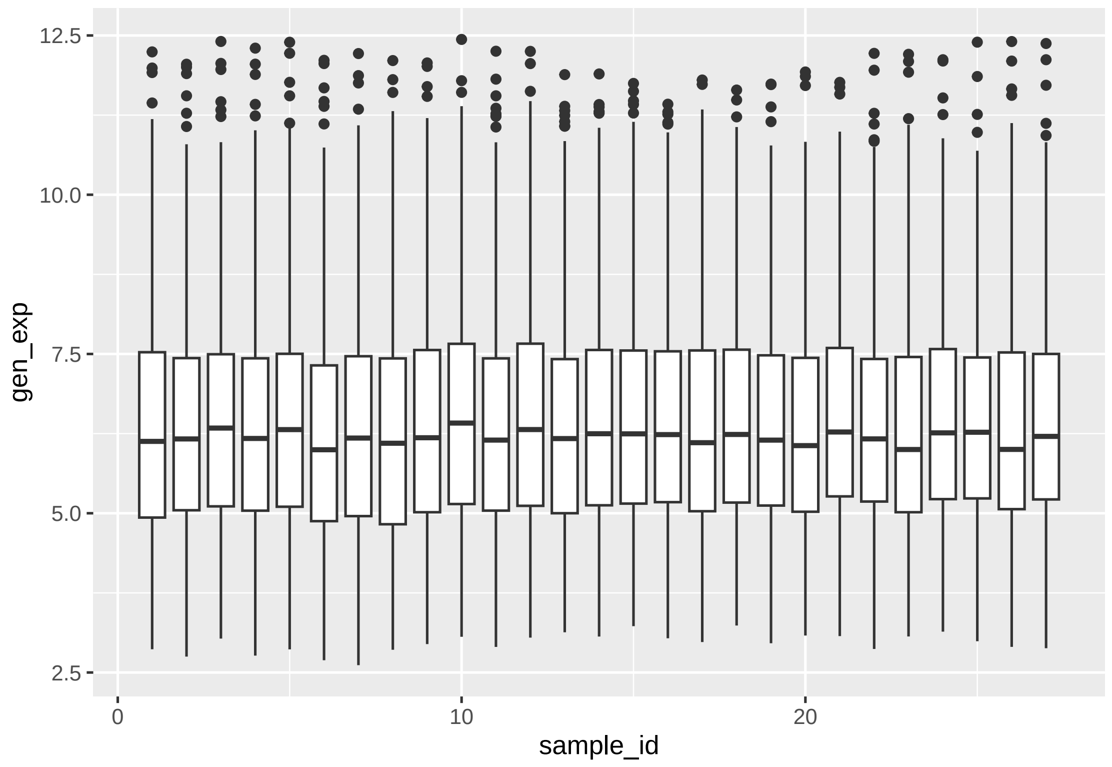
We will refactor the data to have an easier time with the given tasks:

```r
# we will create the variables for gen expression, sample_id, concentration and gen id
# length of the data
n <- 27*500
data <- data.frame(gen_exp = rep(NA, n),
                   gen_id = rep(1:500, 27),
                   concentration = rep(NA, n),
                   sample_id = rep(NA, n))

# this part of the code is very slow and cheap. Feel free to find a more elegant solution
response_dose <- c(25, 150, 350, 450, 550, 800, 1000)

data$concentration[1:(6*500)] <- 0

for (i in 1:7) {
  lower_bound <- 6*500 + (i-1)*1500
  upper_bound <- 6*500 + i*1500
  data$concentration[lower_bound:upper_bound] <- response_dose[i]
}

sample_ids <- factor(as.character(1:27),
                     levels = as.character(1:27))

for (i in 1:27) {
  data$sample_id[((i-1)*500):(i*500)] <- sample_ids[i]
  for (j in 1:500) {
    coordinate_to_row <- (i-1)*500 + j
    data$gen_exp[coordinate_to_row] <- randomVPA[j,i]

  }
}
```

a) Boxplot from the lecture

The plot from the lecture will be realized using ggplot2

```r
data %>%
  ggplot(aes(y = gen_exp, x = sample_id, group = sample_id)) +
  geom_boxplot()
```



b) Variance of genes after response-dosage

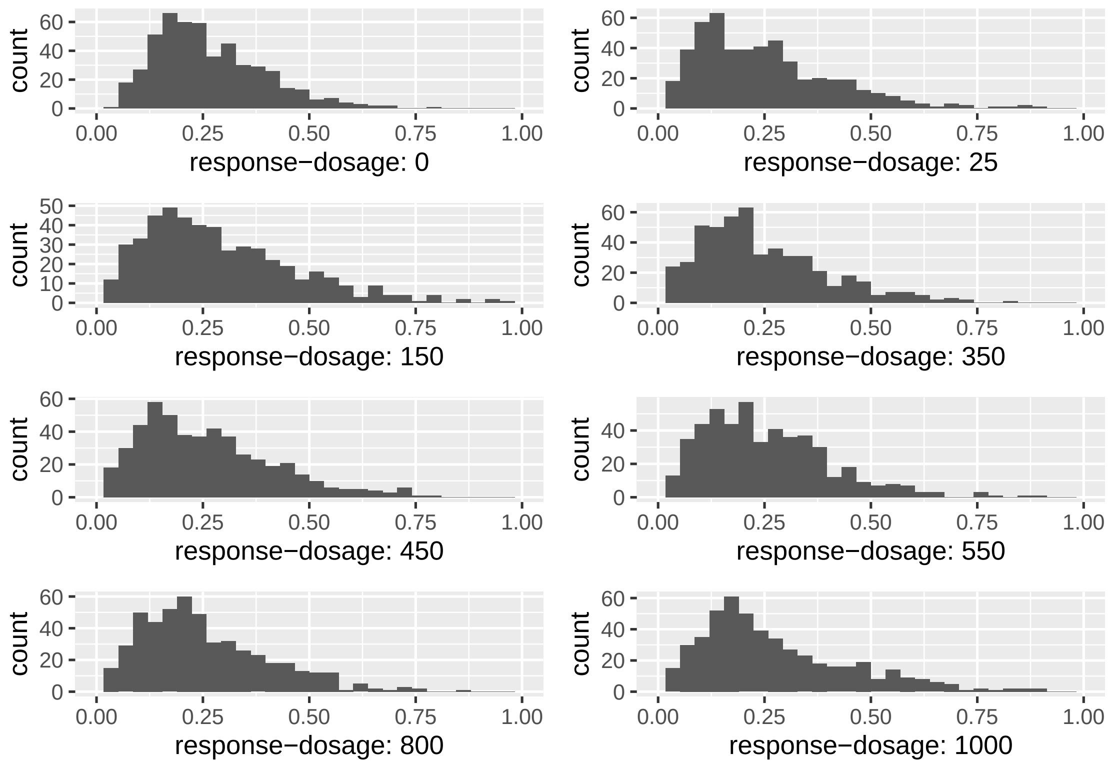
We are going to use tidyverse to calculate the variances. Afterwards using ggplot2 to plot the histogram.

```
# using new data set
var_data <- data %>%
  group_by(gen_id, concentration) %>%
  summarise(var_gen_exp = sd(gen_exp)) %>%
  unique()
```

```
## ‘summarise()‘ has grouped output by ’gen_id’. You can override using the
## ‘.groups‘ argument.
```

```
# histograms by concentration
for (i in 1:8) {
  plot_name <- paste("hist_plot_", i,
                     sep = "")
  # filtering data and creating a fitting x label
  if (i == 1) {
    plot_data <- var_data %>%
      filter(concentration == 0)
    plot_x_label <- paste("response-dosage:", 0)
  } else {
    plot_data <- var_data %>%
      filter(concentration == response_dose[i-1])
    plot_x_label <- paste("response-dosage:", response_dose[i-1])
  }
  # creating a plot and assigning it a name
  assign(plot_name,
         plot_data %>%
           ggplot(aes(x = var_gen_exp)) +
           geom_histogram() +
           xlim(0,1) +
           xlab(plot_x_label))
}
grid.arrange(hist_plot_1,
             hist_plot_2,
             hist_plot_3,
             hist_plot_4,
             hist_plot_5,
             hist_plot_6,
             hist_plot_7,
             hist_plot_8,
             ncol = 2
             )
```

```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```

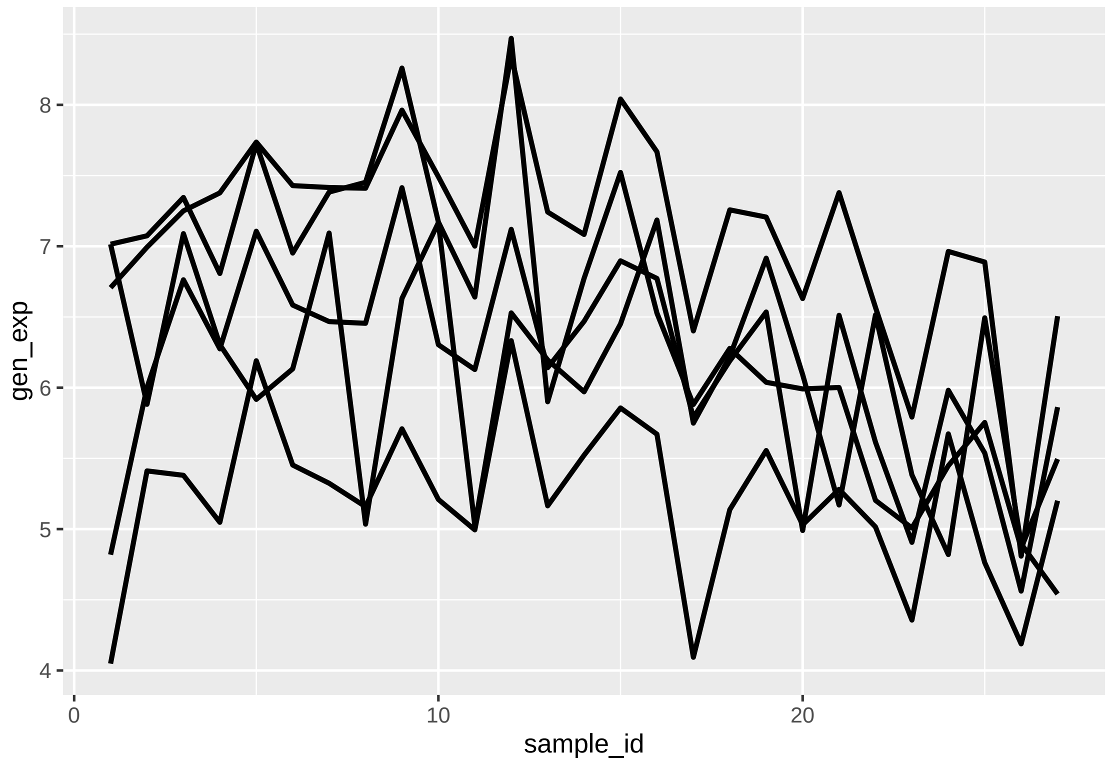


c) Ploting the gene profiles with the highest variance

For this we are going to use `var_data` again

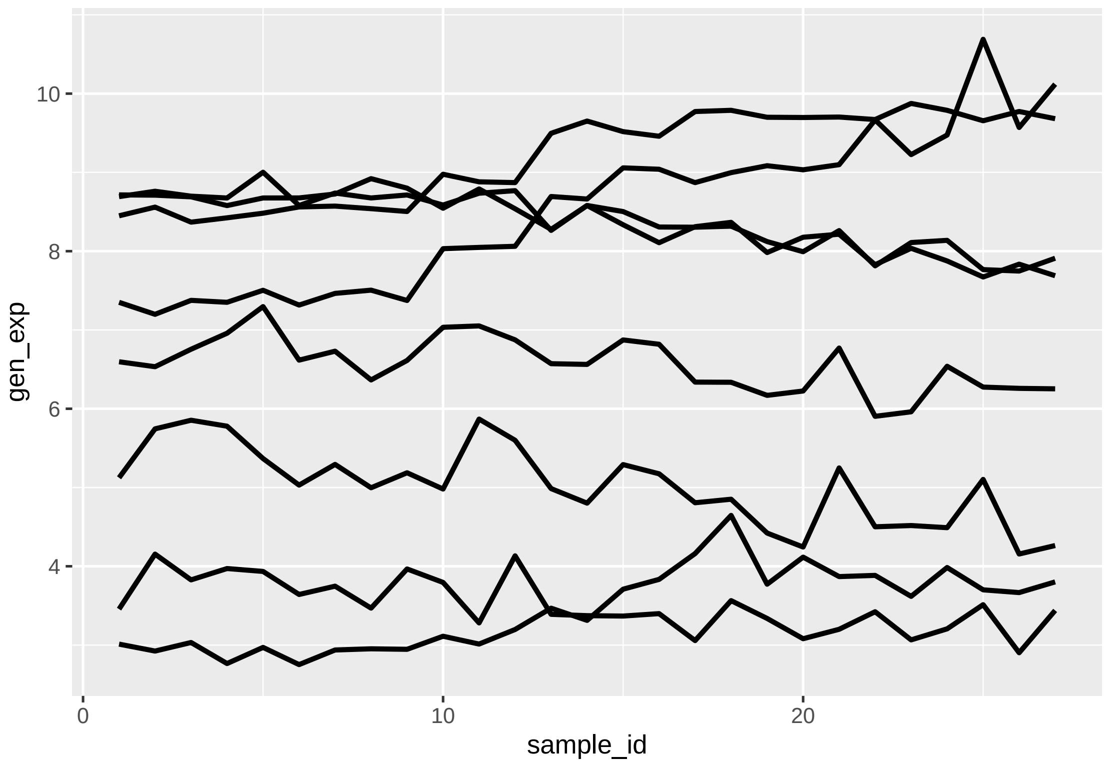
```r
# finding the max values after concentration
max_var_data <- var_data %>%
  group_by(concentration) %>%
  summarise(max_var = max(var_gen_exp),
            # find the gen id that corresponds to the maximum value
            gen_id = gen_id[which(var_gen_exp == max(var_gen_exp))]) %>%
  unique()

# inner join with processed data
max_var_plot_data <- data %>%
  filter(gen_id %in% max_var_data$gen_id)

# plotting
max_var_plot_data %>% ggplot(aes(x = sample_id,
                                 y = gen_exp,
                                 group = gen_id)) +
  geom_line(size = 1)
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



afterwards we are going to repeat the same procedure for min values:

```
# finding the max values after concentration
min_var_data <- var_data %>%
  group_by(concentration) %>%
  summarise(min_var = min(var_gen_exp),
            # find the gen id that corresponds to the maximum value
            gen_id = gen_id[which(var_gen_exp == min(var_gen_exp))]) %>%
  unique()

# inner join with processed data
min_var_plot_data <- data %>%
  filter(gen_id %in% min_var_data$gen_id)

# plotting
min_var_plot_data %>% ggplot(aes(x = sample_id,
                                 y = gen_exp,
                                 group = gen_id)) +
  geom_line(size = 1)
```

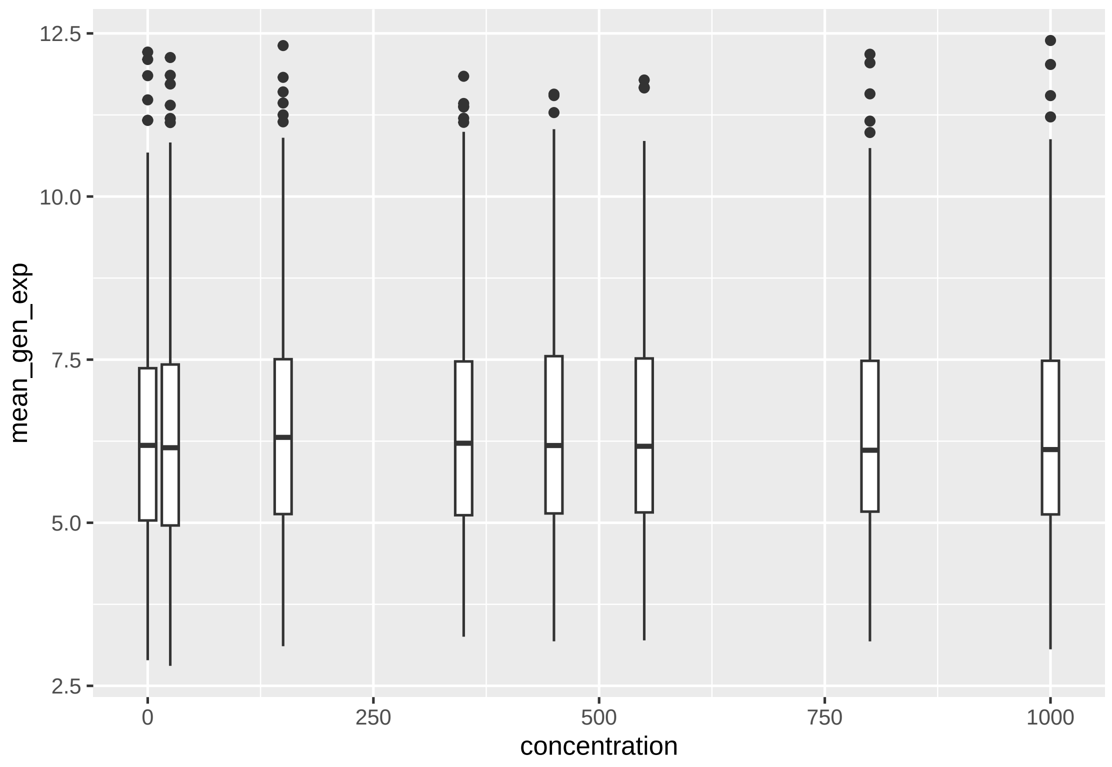


d) Mean gene expression after concentration per gene

```
# generating the data same way as in (c)
mean_data <- data %>%
  group_by(gen_id, concentration) %>%
  summarise(mean_gen_exp = mean(gen_exp)) %>%
  unique()
```

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

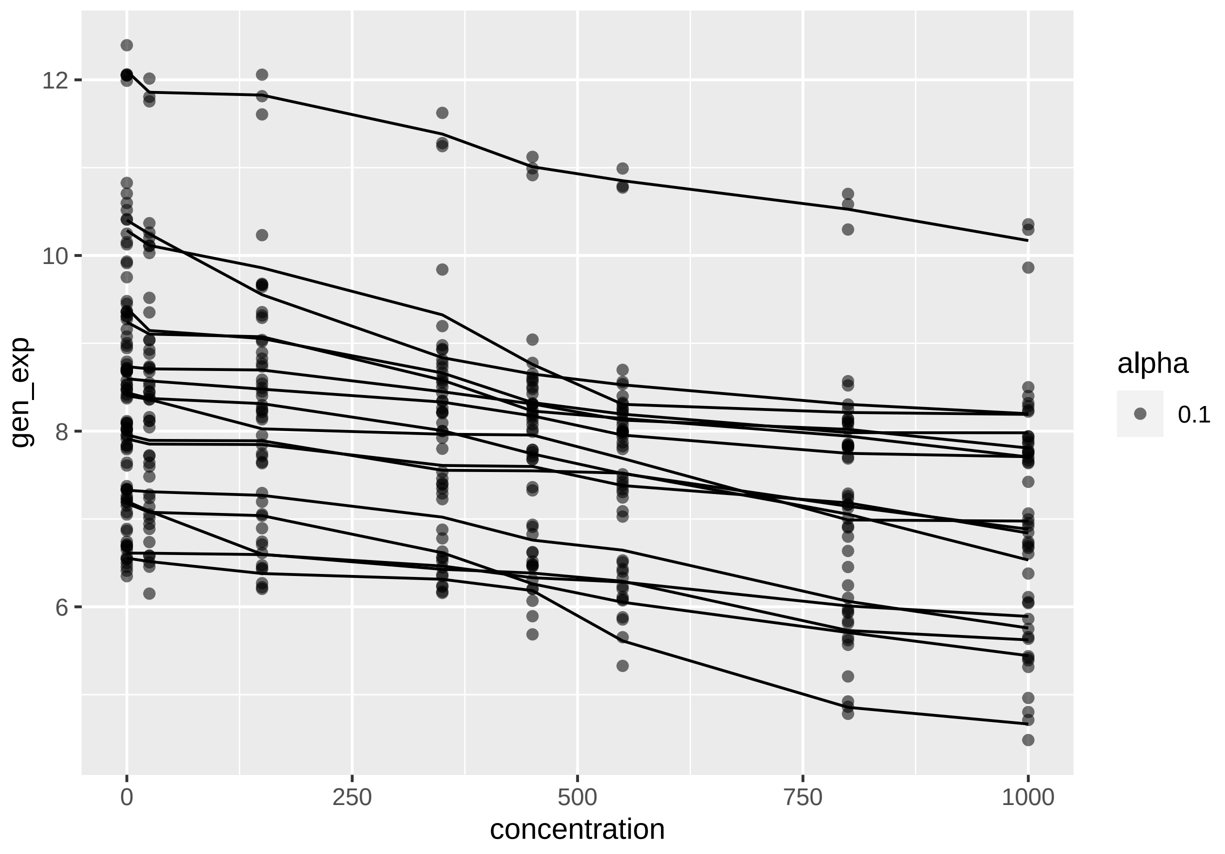
```
box_plot_mean <- mean_data %>%
  ggplot(aes(x = concentration,
             y = mean_gen_exp,
             group = concentration)) +
  geom_boxplot()

box_plot_mean
```



e) Grouping by monotonically increasing and decreasing

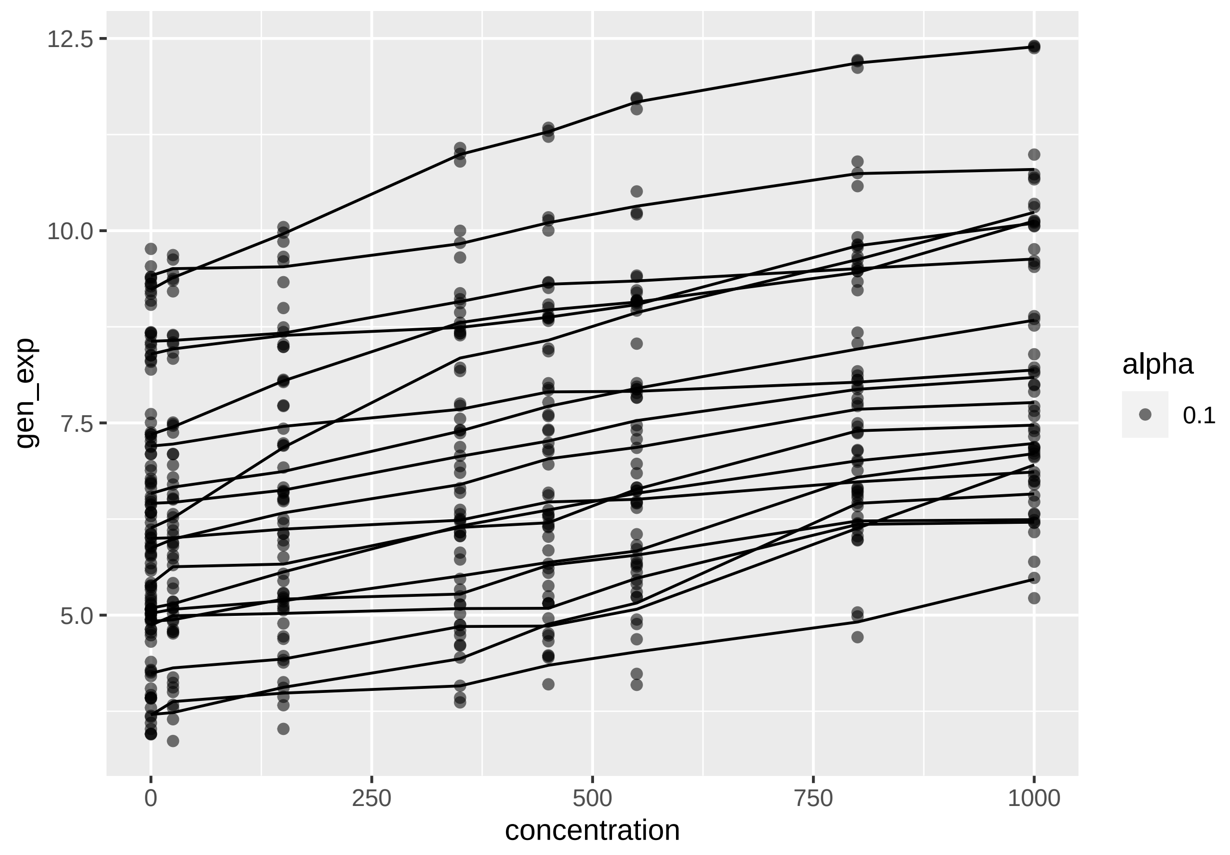
```
is_increasing <- function(v) {
  n <- length(v)
  return(sum(order(v) != 1:n) == 0)
}

is_decreasing <- function(v) {
  n <- length(v)
  return(sum(order(v) != n:1) == 0)
}

mean_data <- arrange(mean_data, gen_id, concentration)

# generating additional variables for increasing and decreasing
```

```
mean_data$is_increasing <- rep(NA, 4000)
mean_data$is_decreasing <- rep(NA, 4000)

for (i in 1:500) {
  lower <- ((i-1)*8+1)
  upper <- (i*8)
  mean_data$is_increasing[lower:upper] <- is_increasing(mean_data$mean_gen_exp[lower:upper])
  mean_data$is_decreasing[lower:upper] <- is_decreasing(mean_data$mean_gen_exp[lower:upper])
}

data <- left_join(data, mean_data, by=c("gen_id", "concentration"))

data_decreasing <- data %>%
  filter(is_decreasing)

data_decreasing %>%
  ggplot(aes(x = concentration,
             y = gen_exp)) +
  geom_point(aes(alpha = 0.1)) +
  geom_line(aes(group = gen_id,
                y = mean_gen_exp))
```



```
data_increasing <- data %>%
  filter(is_increasing)
```

```
data_increasing %>%
  ggplot(aes(x = concentration,
             y = gen_exp)) +
  geom_point(aes(alpha = 0.1)) +
  geom_line(aes(group = gen_id,
                y = mean_gen_exp))
```

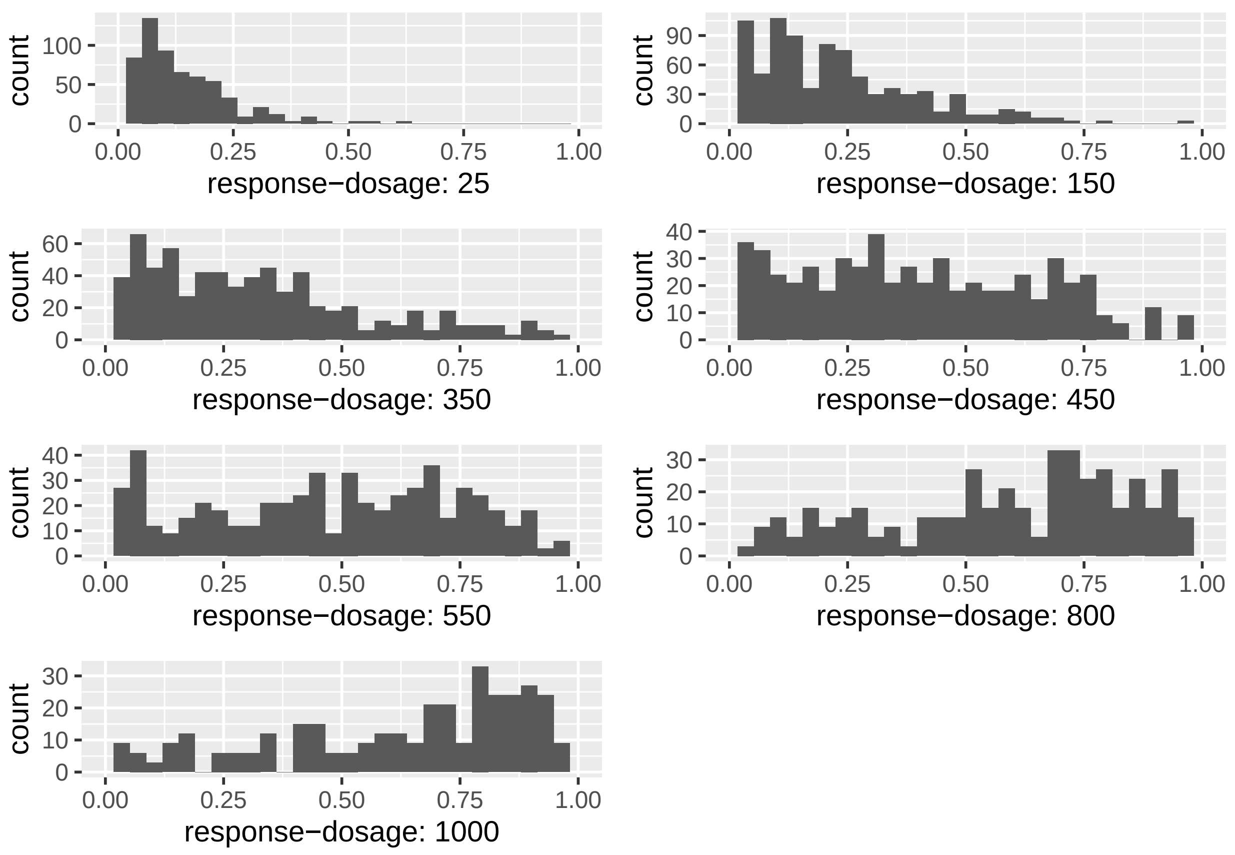


f) plotting differences from controll

```
# first compute the mean of control-dosages
controll_data <- data %>%
  filter(concentration == 0) %>%
  group_by(gen_id) %>%
  summarise(mean_controll = mean(gen_exp)) %>%
  unique()

#
final_data <- data %>% left_join(controll_data, by=c("gen_id"))

# computing differences
final_data$diff_controll <- final_data$mean_gen_exp - final_data$mean_controll

# removing control-dosages
```

```r
final_data <- final_data %>% filter(concentration != 0)

# histograms by concentration
for (i in 1:7) {
  plot_name <- paste("hist_plot_", i,
                     sep = "")
  plot_data <- final_data %>%
    filter(concentration == response_dose[i])

  plot_x_label <- paste("response-dosage:", response_dose[i])

  # creating a plot and assigning it a name
  assign(plot_name,
         plot_data %>%
           ggplot(aes(x = diff_controll)) +
           geom_histogram() +
           xlim(0,1) +
           xlab(plot_x_label))
}
grid.arrange(hist_plot_1,
             hist_plot_2,
             hist_plot_3,
             hist_plot_4,
             hist_plot_5,
             hist_plot_6,
             hist_plot_7,
             ncol = 2
             )
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```


count
response−dosage: 25
response−dosage: 150
response−dosage: 350
response−dosage: 450
response−dosage: 550
response−dosage: 800
response−dosage: 1000
0.00
0.25
0.50
0.75
1.00