Assignment_1

Yuying Cheng

Task 1

Literature: Unbiased integration of single cell transcriptome replicates, Loza et al.

1. What is the medically relevant insight from the article?

   This is a computational paper that is focused on horizontal integration of single-cell transcriptome data where there are multiple replicates measuring the same features, which potentially leads to the presence of batch effect. This paper aims to correct this batch effect that is assumed to be almost orthogonal to and smaller than the biological effect and thus fulfilling the integration of the replicates.

2. Which genomics technology/technologies were used?

   Single-cell RNA-sequencing

3. Further related research questions:

   In the paper, MNN pairs were determined by e.g. finding k-closest cells in the query batch for every cell in the ref. batch, then k-closest cells in the ref. batch were found for every cell in the query batch, then intersections of these pairs were selected as the MNN pairs. I read it from the paper that here k is set to 30. I wonder here the number 30(parameter k) represents, in a way, an assumption for the heterogeneity of the cell populations. For example, if a cell population is more homogeneous, we want the k to be larger whereas if more heterogeneous, k smaller. So in this way, I wonder if we could set k as a hyperparameter, and let some resulting metrics to pick an optimal k for the given datasets.

~~~
```{r setup}
#| message: false
#| warning: false

library(ggplot2)
library(knitr)
~~~

```
library(kableExtra)
library(dplyr)
library(tidyverse)
library(stringr)
library(RColorBrewer)

fixed_palette <- "Dark2"
set_style <- function(p){
  return(p +
           theme_classic() +
           theme(legend.position = "top") +
           scale_color_brewer(palette = fixed_palette)+
           scale_fill_brewer(palette = fixed_palette))
}
```
```

Task 4 - R basic operations

```
sqrt(10)
```

```
[1] 3.162278
```

```
log2(32)
```

```
[1] 5
```

```
sum <- 0
for (i in seq(1, 1000)){
  sum <- sum + i
}
sum
```

```
[1] 500500
```

```
sum <- 0
for (i in seq(2, 1000, 2)){
```
```

```
  sum <- sum + i
}
sum
```

```
[1] 250500
```

```
choose(100, 2)
```

```
[1] 4950
```

```
choose(100, 3)
```

```
[1] 161700
```

Task 5 - Using R example datasets

Describe briefly the content of the CO2 dataset using the help function.

`CO2` is data frame contains data from an experiment on cold tolerance of the grass species Echinochloa crus-galli. The experiment subjects are 6 plants originated either from Quebec or Mississippi. Their CO_2 uptake rate was measured at several levels of ambient CO_2 , with 2 treatment conditions - chilled and not chilled before the measurement.

What is the average and median CO2 uptake of the plants from Quebec and Mississippi?

```
data(CO2)
head(CO2) %>%
  kbl() %>%
  kable_styling()
CO2 %>%
  group_by(Type) %>%
  summarise(median = median(uptake), average = mean(uptake)) %>%
  kbl() %>%
  kable_styling()
```

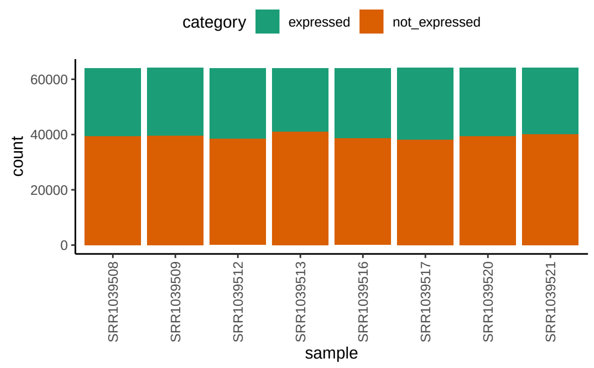
Plant	Type	Treatment	conc	uptake
Qn1	Quebec	nonchilled	95	16.0
Qn1	Quebec	nonchilled	175	30.4
Qn1	Quebec	nonchilled	250	34.8
Qn1	Quebec	nonchilled	350	37.2
Qn1	Quebec	nonchilled	500	35.3
Qn1	Quebec	nonchilled	675	39.2

Type	median	average
Quebec	37.15	33.54286
Mississippi	19.30	20.88333

In the "airway" example data from Bioconductor, how many genes are expressed in each sample? How many genes are not expressed in any sample?

```r
library(airway)
data(airway)
expressed_no <- list(
  expressed = ~sum(.x > 0),
  not_expressed = ~sum(.x == 0)
)
df <- as.data.frame(assay(airway)) %>%
  summarise_all(.fun = expressed_no,
                .names = "{.col}.{.fn}") %>%
  gather(sample, count) %>%
  mutate(category = str_extract(sample, "expressed|not_expressed"),
         sample = str_remove(sample, ".expressed|.not_expressed"))
df %>%
  kbl() %>%
  kable_styling()
p <- ggplot(df, aes(x = sample, y = count, fill = category)) +
  geom_bar(stat = "identity")
set_style(p) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```

sample	count	category
SRR1039508	24633	expressed
SRR1039509	24527	expressed
SRR1039512	25699	expressed
SRR1039513	23124	expressed
SRR1039516	25508	expressed
SRR1039517	25998	expressed
SRR1039520	24662	expressed
SRR1039521	23991	expressed
SRR1039508	39469	not_expressed
SRR1039509	39575	not_expressed
SRR1039512	38403	not_expressed
SRR1039513	40978	not_expressed
SRR1039516	38594	not_expressed
SRR1039517	38104	not_expressed
SRR1039520	39440	not_expressed
SRR1039521	40111	not_expressed

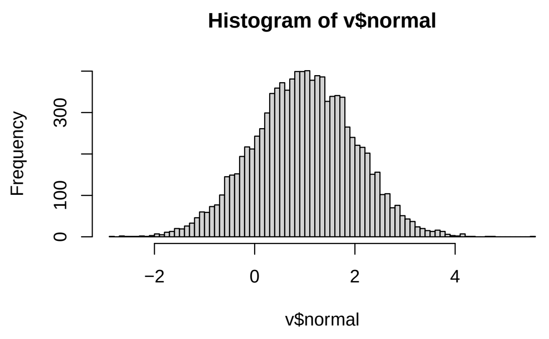


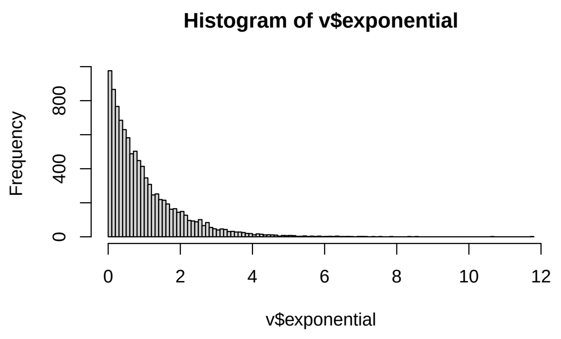
Task 6 - R Functions

Write a function that calculates the ratio of the mean and the median of a given vector.

```r
#Calculates the ratio of mean and median of a given vector
#input: a numeric vector
#output: the ratio of mean and median of the vector
r_mean_median <- function(v){
  return(mean(v)/median(v))
}

#test
v <- list(normal = rnorm(10000, mean = 1),
          exponential = rexp(10000, rate = 1))
hist(v$normal, breaks = 100)
hist(v$exponential, breaks = 100)
lapply(v, r_mean_median)
```





```
$normal
[1] 0.995326

$exponential
[1] 1.42902
```

Write a function that ignores the lowest and the highest value from a given vector and calculate the mean.

```
#Calculates mean of a vectore after removing *one* maximum and *one* minimum
#input: a vector
#output: mean value after removal of one maximum and one minimum
adjusted_average <- function(v){
  x = sum(v) - min(v) - max(v)
  return(x/(length(v)-2))
}

#test
v = c(seq(1,3), rep(10,3))
v
```

```
[1]  1  2  3 10 10 10
```

```
adjusted_average(v)
```

```
[1] 6.25
```

Pipes

Pipe is a tool predominantly for a linear sequence of operations. To use it, connection the operations with "%>%". However because of its design, there are several situations when it is not appropriate to use pipe, including:

1. when the number of operations is too large, piping makes it hard to debug.
2. when there are multiple inputs and outputs.
3. when there's a non-linear dependency structure of the operations.

Apply family

The apply family is designed to replace the use of loops when it fits. In my work, the apply family is handy when i have several data sets but all subjected to the same pre-processing processes. For example, when i want to get the read counts from bigwig files, instead of writing loops I can lump the sets into a list and use `lapply` so I transform each bigwig file into a read count matrix with one line of code.

Task 7 - Basic visualization

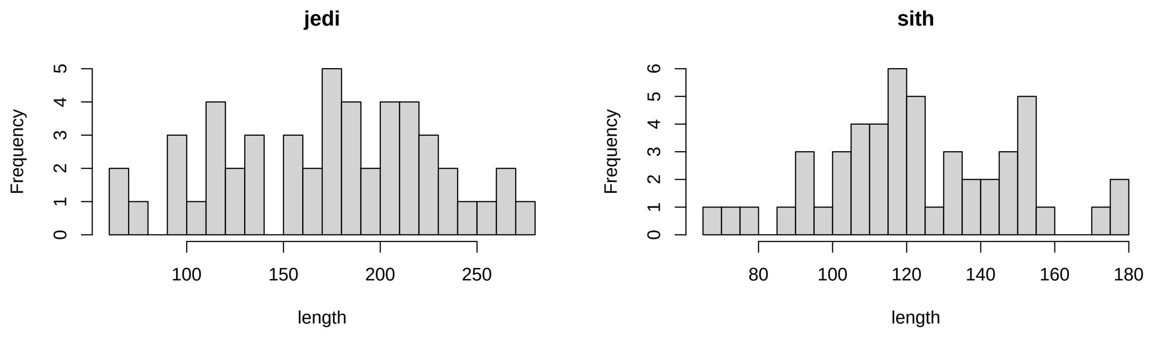
Compare the distributions of the body heights of the two species from the 'magic_guys.csv' dataset graphically.

```r
df <- read.csv("magic_guys.csv")
unique(df$species)
```

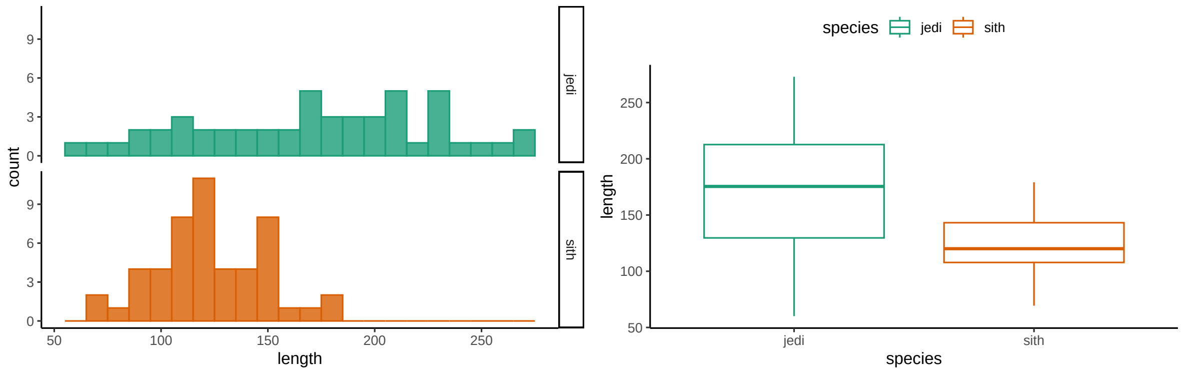
```
[1] "jedi" "sith"
```

```r
hist(df[df$species == "jedi", "length"], breaks = 20, xlab = "length", main = "jedi")
hist(df[df$species == "sith", "length"], breaks = 20, xlab = "length", main = "sith")
```

```r
p1 <- ggplot(data = df, aes(x = length, fill = species, color = species)) +
  geom_histogram(binwidth = 10, alpha = 0.8) +
  facet_grid(species ~ .)
p1 <- set_style(p1) + theme(legend.position = "none")
p1
p2 <- ggplot(data = df, aes(x = species, y = length, color = species)) +
  geom_boxplot()
```



```r
p2 <- set_style(p2)
p2
```



```r
plots <- list(p1, p2)
names(plots) <- c("gg_hist.png", "gg_box.png")
lapply(names(plots), function(x) ggsave(x, plots[[x]]))
```

```
Saving 5.5 x 3.5 in image
Saving 5.5 x 3.5 in image

[[1]]
[1] "gg_hist.png"

[[2]]
[1] "gg_box.png"
```

Load the gene expression data matrix from the 'microarray_data.tab' dataset provided in the shared folder, it is a big tabular separated matrix.

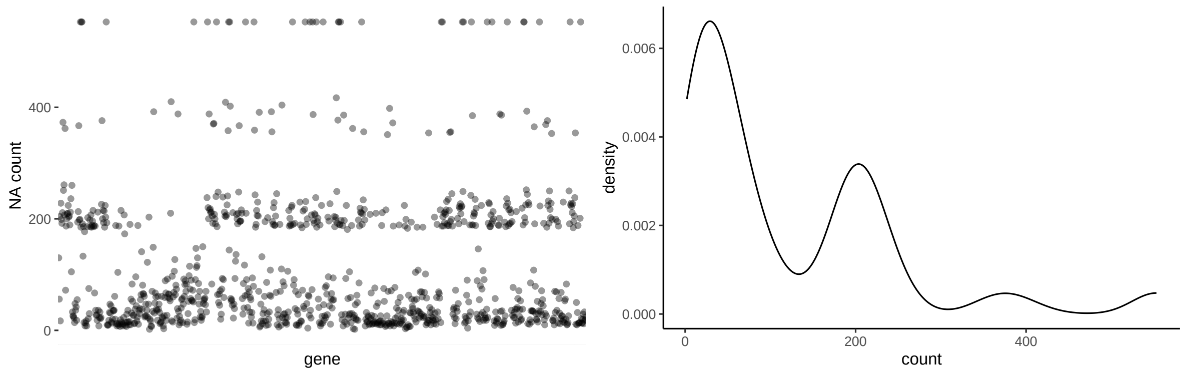
1. How big is the matrix in terms of rows and columns?

```r
df <- read.table("microarray_data.tab", sep = "\t", header = T)
size_sum(df)
```

```
[1] "[553 x 1,000]"
```

2. Count the missing values per gene and visualize this result.

```r
gene_na <- df %>%
  summarise_all(~sum(is.na(.))) %>%
  gather(key = "gene", value = "count")
p1 <- ggplot(gene_na, aes(x = gene, y = count)) +
  geom_point(alpha = 0.4) +
  ylab("NA count") +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank())
p1
p2 <- ggplot(gene_na, aes(x = count)) +
  geom_density()
set_style(p2)
```

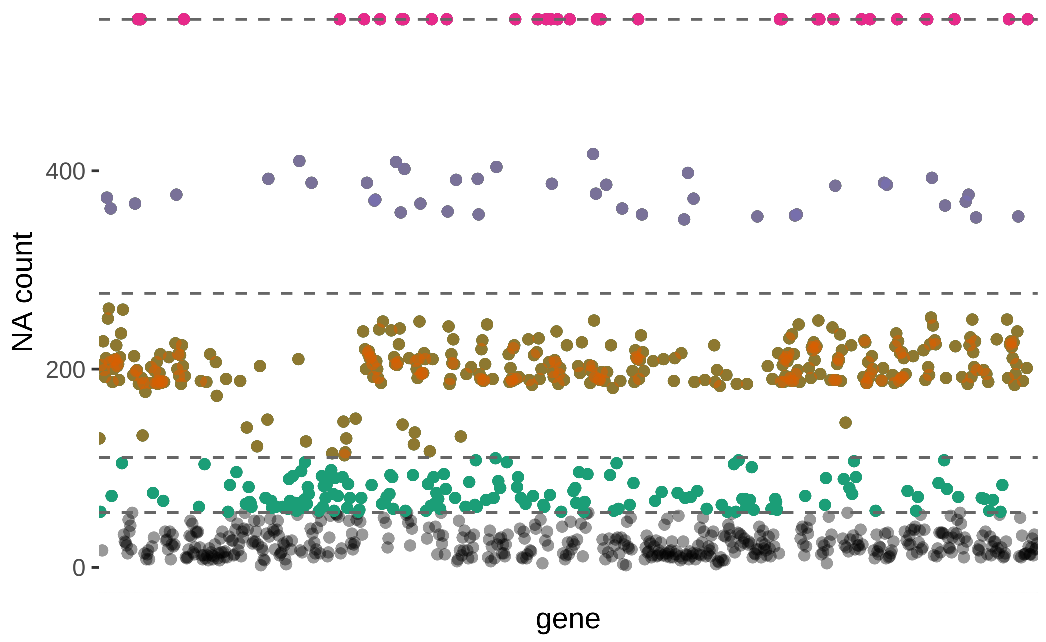


3. Find the genes for which there are more than X% (X=10%, 20%, 50%) missing values.

```r
set_10 <- gene_na %>%
  filter(count > nrow(df) * 0.1) %>%
  mutate(category = ">10%")
```

```r
set_20 <- set_10 %>%
  filter(count > nrow(df) * 0.2) %>%
  mutate(category = ">20%")
set_50 <- set_20 %>%
  filter(count > nrow(df) * 0.5) %>%
  mutate(category = ">50%")
set_na <- gene_na %>%
  filter(count == nrow(df))

p1 +
  geom_point(data = set_10, color = brewer.pal(4, fixed_palette)[1], alpla = 0.4) +
  geom_point(data = set_20, color = brewer.pal(4, fixed_palette)[2], alpha = 0.6) +
  geom_point(data = set_50, color = brewer.pal(4, fixed_palette)[3], alpha = 0.8) +
  geom_point(data = set_na, color = brewer.pal(4, fixed_palette)[4]) +
  geom_hline(yintercept = c(0.1, 0.2, 0.5, 1) * nrow(df), color = "grey40", linetype = "da
```



4. Replace the missing values by the average expression value for the particular gene. (Note: Imputing data has to be used with caution!)

```r
replace_with_mean <- function(v){
  v[is.na(v)] <- mean(v, na.rm = T)
}

df_imputed <- df %>%
  mutate(across(where(is.numeric), ~replace_na(., mean(., na.rm = T))))
df_imputed[1:8, 1:6]
```

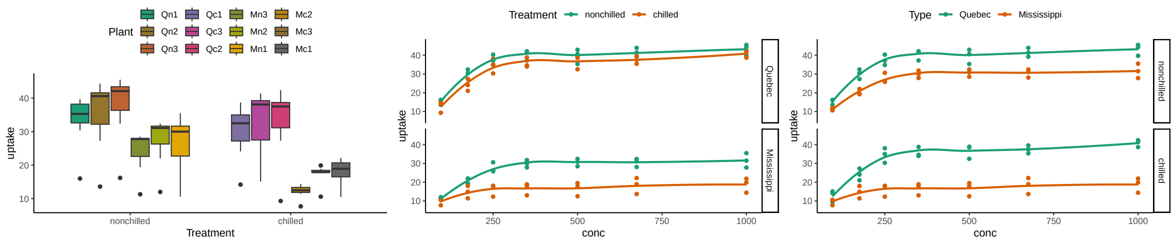
```
           g1        g2         g3          g4          g5         g6
1  1.80200000 0.1656927 -0.1820000  1.31200000  3.49700000  0.4390000
2  0.02547518 0.1656927  7.6930000 -0.06731957  0.19300000 -1.3830000
3  1.07900000 0.1656927  1.5560000  1.65200000 -0.01812288  0.4600000
4  3.60700000 0.1656927  1.9140000 -0.06731957  1.40000000  1.1090000
5 -1.70000000 0.1656927  0.9430000 -0.06731957 -0.17000000 -0.1571338
6  0.02547518 0.1656927  0.0430000 -0.06731957  0.72900000 -0.0890000
7  0.02547518 0.1656927 -0.1230605 -0.06731957 -0.01812288 -0.1571338
8  0.02547518 0.1656927 -0.1230605 -0.06731957 -0.01812288 -1.2970000
```

Visualize the data in the CO2 dataset in a way that gives you a deeper understanding of the data. What do you see?

```r
getPallete <- colorRampPalette(brewer.pal(8, fixed_palette))
N_color <- length(unique(CO2$Plant))
p1<- ggplot(CO2, aes(x = Treatment, y = uptake, fill = Plant)) +
  geom_boxplot()
set_style(p1) + scale_fill_manual(values = getPallete(N_color))
p2 <- ggplot(CO2, aes(x = conc, y = uptake, color = Treatment)) +
  geom_point() +
  geom_smooth(method = "loess", fill = NA) +
  facet_grid(Type ~ .)
set_style(p2)
p3 <- ggplot(CO2, aes(x = conc, y = uptake, color = Type)) +
  geom_point() +
  geom_smooth(method = "loess", fill = NA) +
  facet_grid(Treatment ~ .)
set_style(p3)
```

From the above plots I noticed the following:

1. From the boxplot, plants from Quebec have a higher CO_2 uptake rate than Mississippi, whether they were chilled or not before the measurement.
2. Comparing the treatments, chilling decreases the CO_2 uptake rate in general.



3. From the second and the third plot, we can learn about the relationship between ambient CO_2 level(conc) and the CO_2 uptake rate of the plants. This relationship demonstrates a non-linear, monotonically increasing trend until it reaches a plateau, i.e. CO_2 uptake rate increases as the level of conc increases, with a deceasing rate of change, and eventually reaches a plateau.
4. Whether or not having chilled before measurement affects(decreases the uptake rate) the plants from Mississippi more than Quebec.

Task 8

Install the Tidybiology package, which includes the data 'chromosome' and 'proteins'.

1. Extract summary statistics (mean, median and maximum) for the following variables from the 'chromosome' data: variations, protein coding genes, and miRNAs. Utilize the tidyverse functions to make this as simply as possible.

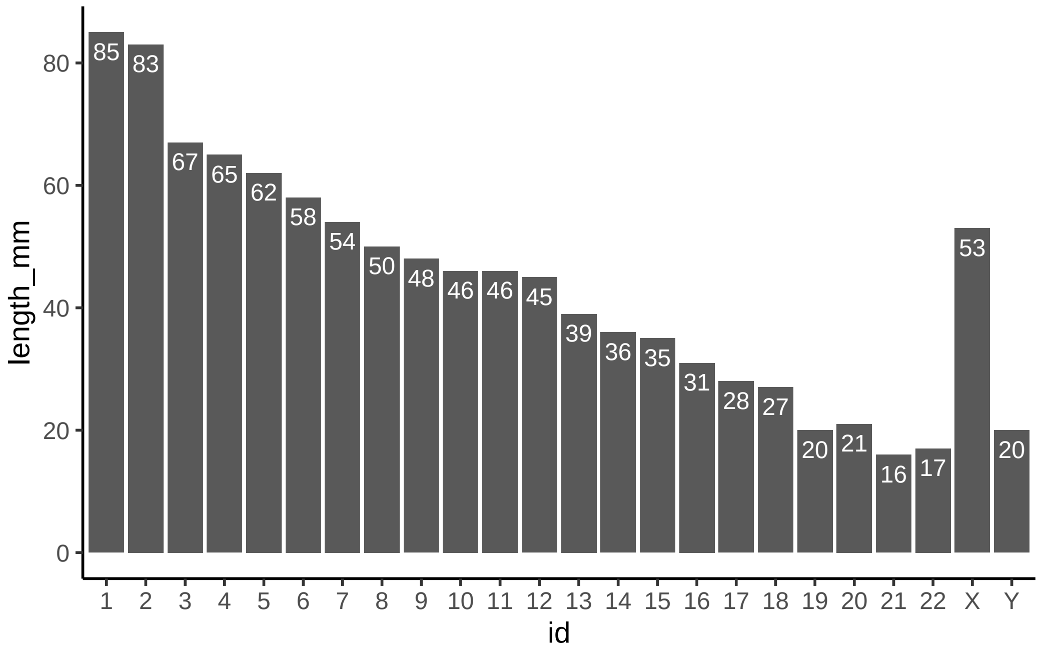
```r
library(tidybiology)
#colnames(chromosome)
chromosome %>% select(c("variations", "protein_codinggenes", "mi_rna")) %>%
  summarise_all(.funs = list(mean = ~round(mean(.x)), median = median, max = max),
                .names = "{.col}.{.fn}") %>%
  gather(key = "statistics") %>%
  kbl() %>%
  kable_styling()
```

2. How does the chromosome size distribute? Plot a graph that helps to visualize this by using ggplot2 package functions.

```r
p <- ggplot(chromosome, aes(x = id, y = length_mm)) +
  geom_bar(stat = "identity") +
  geom_text(aes(label = length_mm), vjust = 1.6, color = "white", size = 3)
```

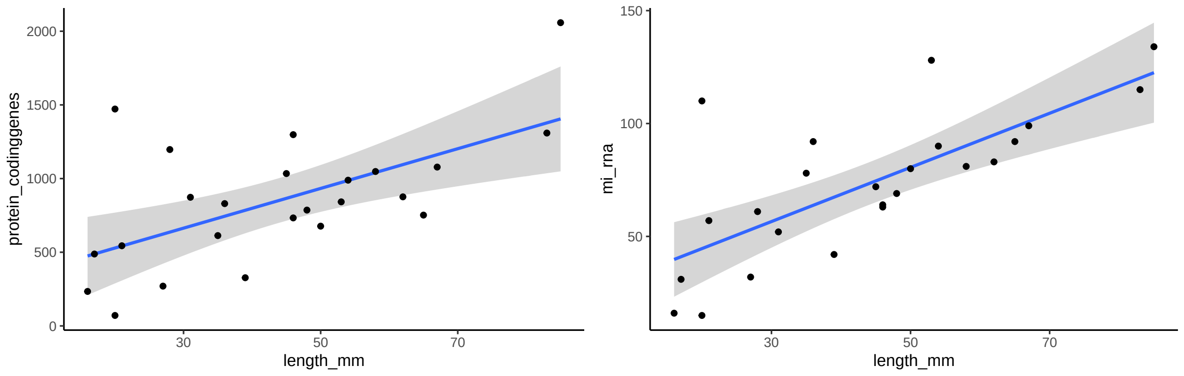
statistics	value
variations_mean	6484572
protein_codinggenes_mean	850
mi_rna_mean	73
variations_median	6172346
protein_codinggenes_median	836
mi_rna_median	75
variations_max	12945965
protein_codinggenes_max	2058
mi_rna_max	134

```
set_style(p)
```



3. Does the number of protein coding genes or miRNAs correlate with the length of the chromosome? Make two separate plots to visualize these relationships.

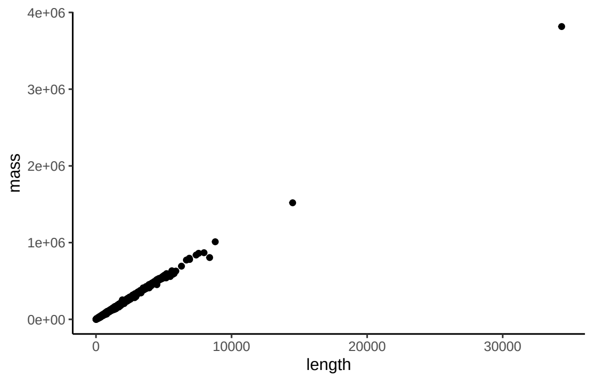
```
p1 <- ggplot(chromosome, aes(x = length_mm, y = protein_codinggenes)) +
  geom_smooth(method = "lm") +
  geom_point()
set_style(p1)
p2 <- ggplot(chromosome, aes(x = length_mm, y = mi_rna)) +
  geom_smooth(method = "lm") +
  geom_point()
set_style(p2)
```



4. Calculate the same summary statistics for the 'proteins' data variables length and mass. Create a meaningful visualization of the relationship between these two variables by utilizing the ggplot2 package functions. Play with the colors, theme- and other visualization parameters to create a plot that pleases you.

```
proteins %>% select(c("length", "mass")) %>%
  summarise_all(.funs = list(mean = mean, median = median, max = max),
                .names = "{.col}.{.fn}") %>%
  gather(key = "statistics") %>%
  kbl() %>%
  kable_styling()
p <- ggplot(proteins, aes(x = length, y = mass)) +
  geom_point()
set_style(p)
```

statistics	value
length_mean	557.160254527655
mass_mean	62061.3791483113
length_median	414
mass_median	46140.5
length_max	34350
mass_max	3816030


4e+06
3e+06
2e+06
1e+06
0e+00
mass
length