2023 KI-UTokyo Joint Doctoral Course

CUI YANG

2023/03/10

Task 1 - Literature

Paper: Altered cervicovaginal microbiota in premenopausal ovarian cancer patients PMID: 34856363

What is the medically relevant insight from the article?

The medically relevant insight from the article is that the cervicovaginal microbiota in premenopausal ovarian cancer patients is frequently a diversified community and similar to those in healthy subjects at postmenopausal ages. The diverse microbiota was associated with the major histotypes of epithelial ovarian cancer, including serous ovarian cancer and ovarian clear cell cancer. The study implies the potential of a cervicovaginal microbiome biomarker in screening ovarian cancer in premenopausal women. However, further research is needed to confirm the findings and determine the clinical utility of cervicovaginal microbiota as a biomarker for ovarian cancer diagnosis.

Which genomics technology/ technologies were used?

The study used 16S rRNA amplicon sequencing to determine the composition of the cervicovaginal microbial community at the genus level.

List and explain at least three questions/ hypotheses you can think of that extend the analysis presented in the paper.

1. Can the cervicovaginal microbiota be used as a biomarker for ovarian cancer diagnosis in postmenopausal women?

   This question would extend the analysis presented in the paper by investigating whether the cervicovaginal microbiota can be used as a biomarker for ovarian cancer diagnosis in postmenopausal women, given that the study only focused on premenopausal women.

2. How does the cervicovaginal microbiota change over time in women with ovarian cancer?

   This question would extend the analysis presented in the paper by investigating how the cervicovaginal microbiota changes over time in women with ovarian cancer, which could provide insights into disease progression and potential therapeutic targets.

3. What is the role of specific bacterial species or strains in ovarian cancer development?

   This question would extend the analysis presented in the paper by investigating the role of specific bacterial species or strains in ovarian cancer development, which could provide insights into disease mechanisms and potential therapeutic targets.

Test 4 - R basic operations

```r
library(tidyr)
```

#1

```r
sqrt(100) %>% print() # Calculate the result and print the result
```

```
## [1] 10
```

#2

```r
log2(32) %>% print() # Calculate the result and print the result
```

```
## [1] 5
```

#3

```r
sum(1:1000) %>% print() # Calculate the sum of these numbers and print the result
```

```
## [1] 500500
```

#4

```r
even_numbers <- seq(from = 2, to = 1000, by = 2) # Generate a sequence of even numbers from 2 to 1000
sum_of_even_numbers <- sum(even_numbers) # Calculate the sum of these numbers
print(sum_of_even_numbers) # Print the result
```

```
## [1] 250500
```

#5

```r
n_genes <- 100 # Genes number
pairwise_comparisons <- n_genes * (n_genes - 1) / 2# Calculate the number of pairwise comparisons
print(pairwise_comparisons) # Print the result
```

```
## [1] 4950
```

#6

```r
n_genes <- 100 # Genes number
arrangements_in_triples <- choose(n_genes, 3) # Calculate the number of ways to arrange 100 genes in tr
print(arrangements_in_triples) # Print the result
```

```
## [1] 161700
```

Task 5 - Using R example datasets

#1

```
data(CO2)
```

#2

```
help(CO2)
```

#3

```
Quebec <- subset(CO2, CO2$Type == 'Quebec')# Take the subset of the CO2 data set that 'Type' is Quebec
Quebec$uptake %>% mean() %>% print()# Calculate the mean and print the result
```

```
## [1] 33.54286
```

```
Quebec$uptake %>% median() %>% print()# Calculate the median and print the result
```

```
## [1] 37.15
```

```
Mississippi <- subset(CO2, CO2$Type == 'Mississippi')# Take the subset of the CO2 data set that 'Type'
Mississippi$uptake %>% mean() %>% print()# Calculate the mean and print the result
```

```
## [1] 20.88333
```

```
Mississippi$uptake %>% median() %>% print()# Calculate the median and print the result
```

```
## [1] 19.3
```

Task 6 - R Functions

#1

```
mean_median_ratio <- function(x) {      # Define a function to calculate the mean/median ratio of a vec
  x_mean <- mean(x)                     # Calculate the mean of the input vector
  x_median <- median(x)                 # Calculate the median of the input vector
  ratio <- x_mean / x_median            # Calculate the ratio of the mean and median
  return(ratio)                         # Return the ratio as the output of the function
}
```

#2

```
mean_without_extremes <- function(x) {                  # Define a function to calculate the mean of
  x_without_extremes <- x[-which.min(x)][-which.max(x)] # Remove the lowest and highest values from t
  mean_x <- mean(x_without_extremes)                    # Calculate the mean of the remaining values
  return(mean_x)                                        # Return the mean as the output of the functi
}
```

#3

Pipes in R allow us to chain together multiple operations by sending the output of one function as input to the next function, making code more readable and concise.Piping is useful for performing a series of transformations on a dataset. However, pipes should be used with caution when working with large datasets or when there is a need to save intermediate results, as piping can result in unnecessary copying of data and increased memory usage. Additionally, pipes may not be the best choice for complex or nested operations where it is difficult to maintain clarity and readability.

#4

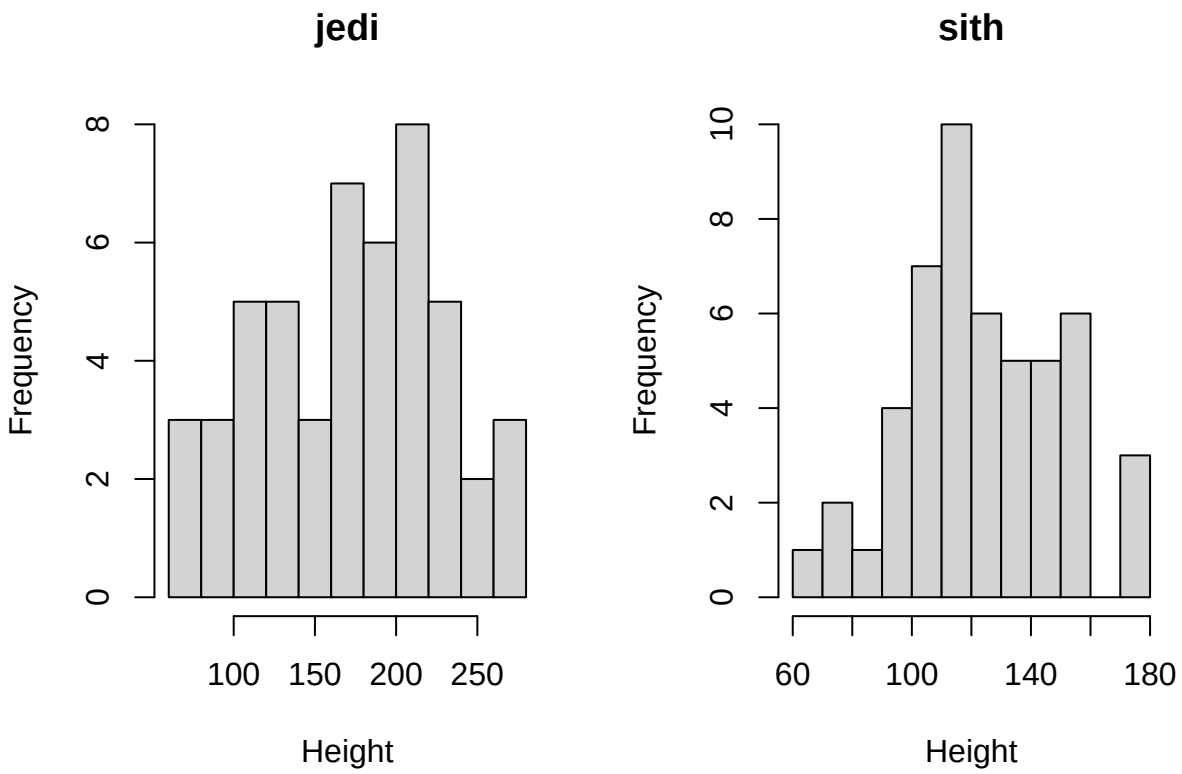
The apply-family of functions in R (apply, lapply, sapply, etc.) are useful for iterating over arrays (e.g., matrices, data frames) and applying a function to each element, row, or column. This can save time and reduce code duplication, especially when working with large datasets or performing repetitive operations. The apply-family functions are also versatile and can be used with user-defined functions or functions from other packages, allowing for flexible data manipulation and analysis.

Task 7 - Basic visualization with R

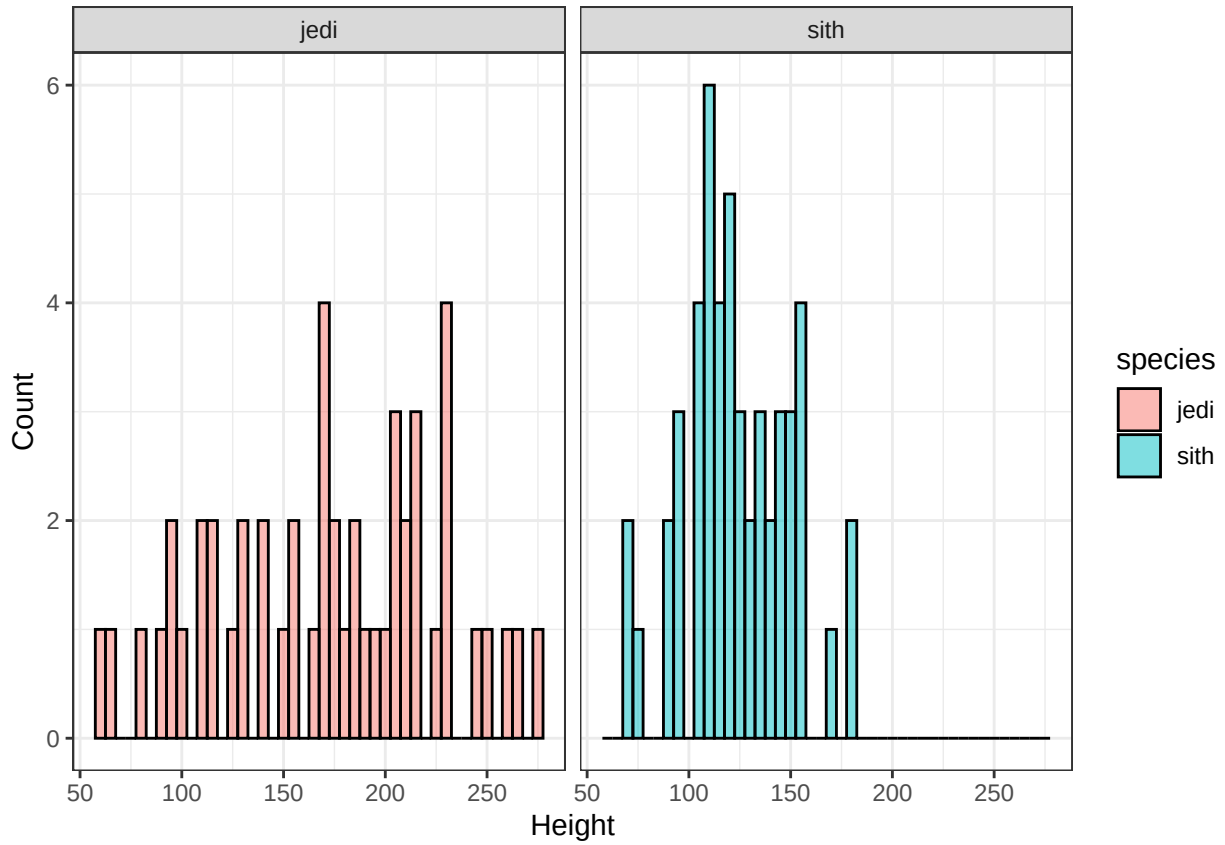
#1

#a. Histograms

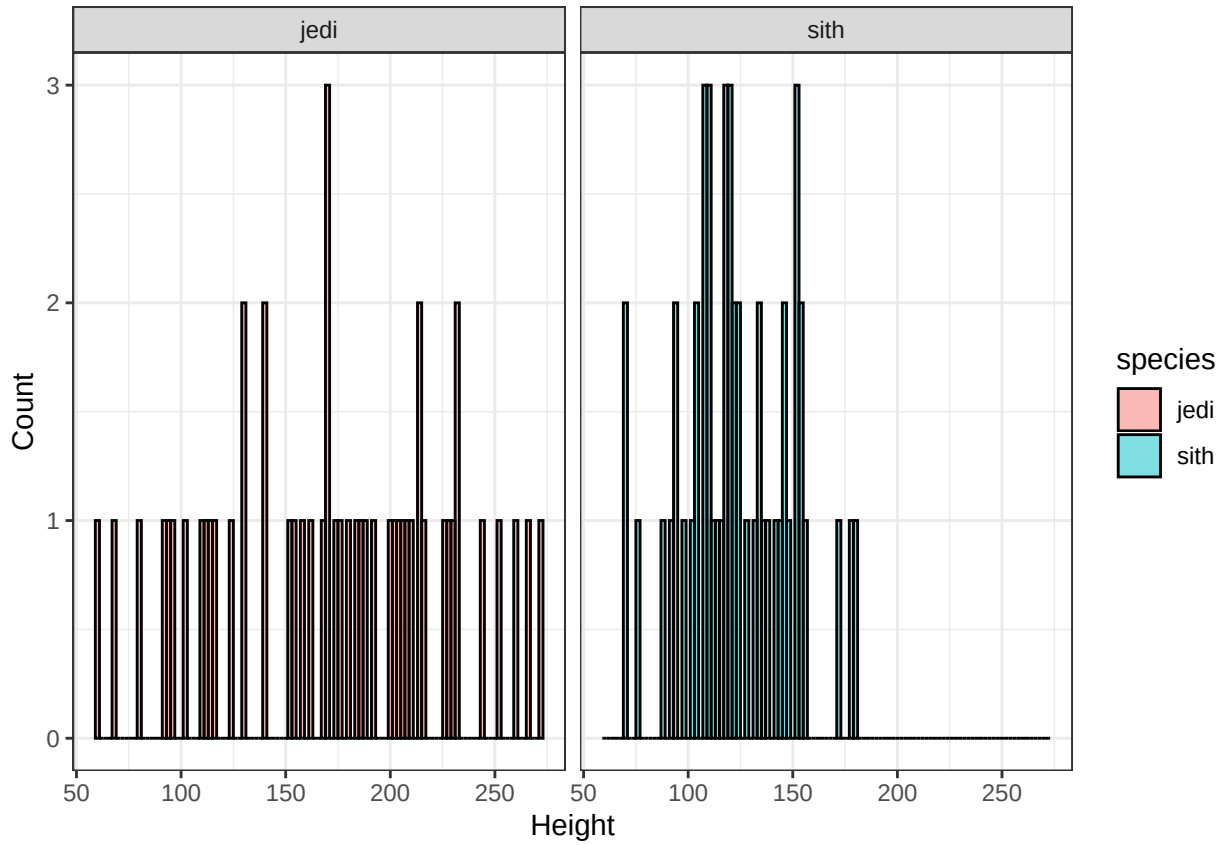
```
library(ggplot2)
library(gridExtra)

# load the data
data <- read.csv("magic_guys.csv")

# create histogram using hist function
par(mfrow=c(1,2)) # display plots side by side
hist(data[data$species=="jedi",]$length, main="jedi", xlab="Height", breaks=10)
hist(data[data$species=="sith",]$length, main="sith", xlab="Height", breaks=10)
```




```
#create histograms using ggplot2 package
ggplot(data, aes(x=length, fill=species)) +
  geom_histogram(binwidth=5, color="black", alpha=0.5) +
  facet_grid(.~species) +
  labs(x="Height", y="Count") +
  theme_bw()
```

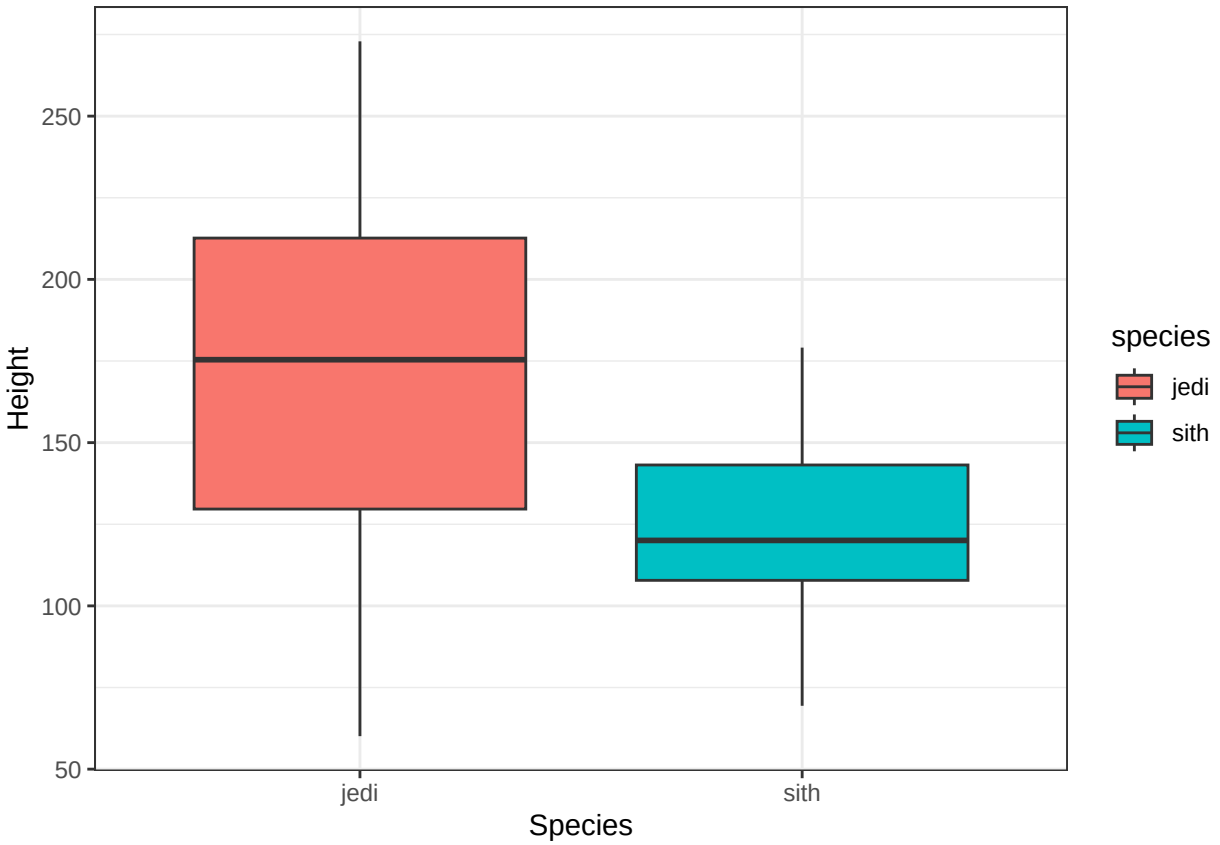



```
ggplot(data, aes(x=length, fill=species)) +
  geom_histogram(binwidth=2, color="black", alpha=0.5) +
  facet_grid(.~species) +
  labs(x="Height", y="Count") +
  theme_bw()
```



#b. Boxplots

```
ggplot(data, aes(x=species, y=length, fill=species)) +
  geom_boxplot() +
  labs(x="Species", y="Height") +
  theme_bw()
```



#c. save

PNG: Use PNG format for images that will be displayed on the web or in a document.

PDF: Use PDF format for high-quality prints, especially when the plot contains text.

SVG: Use SVG format for scalable vector graphics that can be edited in vector graphics software.

```
# save histograms as PNG, PDF, and SVG
ggsave("histograms.png", width=7, height=4, dpi=300)
ggsave("histograms.pdf", width=7, height=4)
ggsave("histograms.svg", width=7, height=4)

# save boxplots as PNG, PDF, and SVG
ggsave("boxplots.png", width=5, height=5, dpi=300)
ggsave("boxplots.pdf", width=5, height=5)
ggsave("boxplots.svg", width=5, height=5)
```

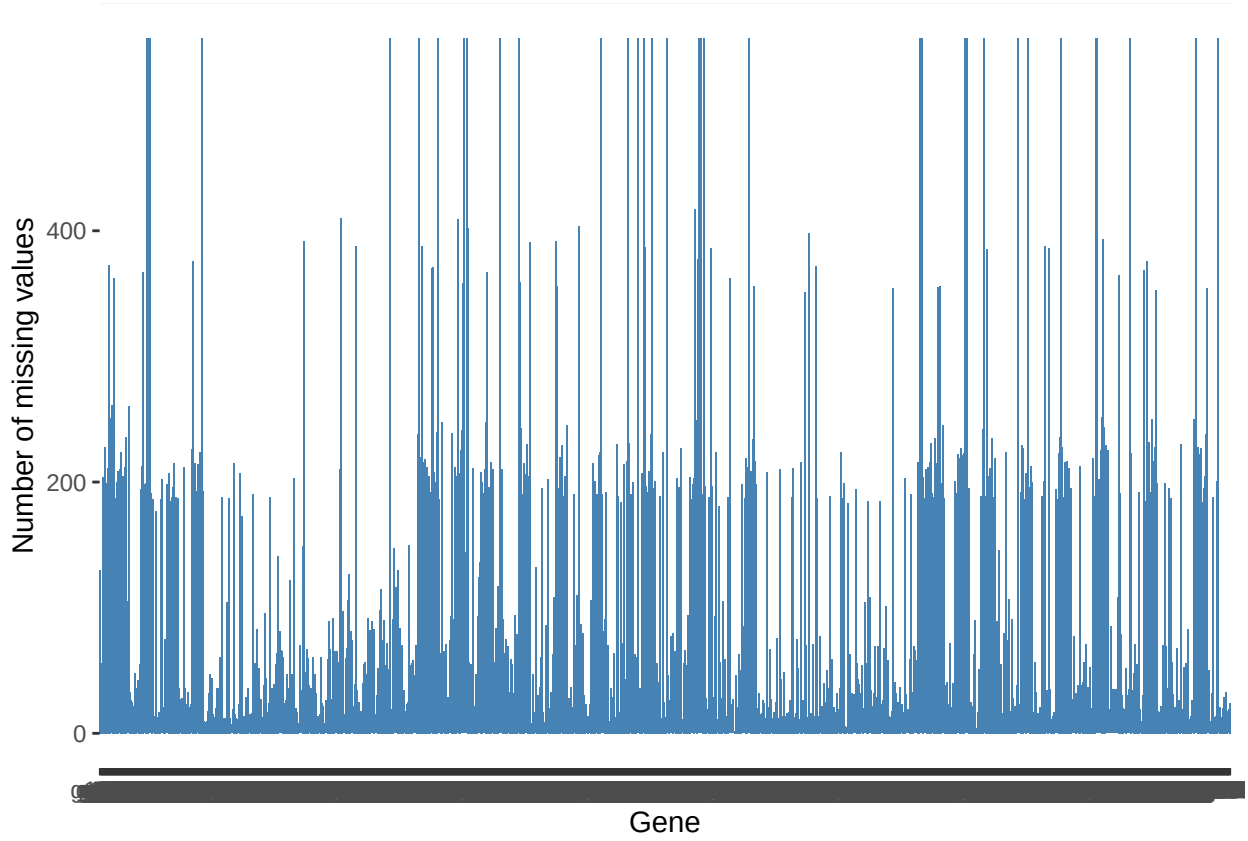
#2

```
#load the data
microarray <- read.table("microarray_data.tab", header = TRUE, sep = "\t")

#a
dim(microarray)# determine size
```

```
## [1]  553 1000
```

#b

```
# Count the missing values per gene
missing_vals <- apply(microarray, 2, function(x) sum(is.na(x)))

# Create a data frame with gene names and missing value counts
missing_df <- data.frame(genes = colnames(microarray), missing_vals = missing_vals)

# Create a bar plot of missing value counts
ggplot(missing_df, aes(x = genes, y = missing_vals)) +
  geom_bar(stat = "identity", fill = "steelblue") +
  xlab("Gene") +
  ylab("Number of missing values")
```



#c

```
# calculate percentage missing values per gene
pct_missing <- apply(microarray, 2, function(x) mean(is.na(x))*100)

# find genes with more than X% missing values
x <- 10 # set X to 10%
genes_10 <- names(pct_missing[pct_missing > x])

x <- 20 # set X to 20%
genes_20 <- names(pct_missing[pct_missing > x])
```

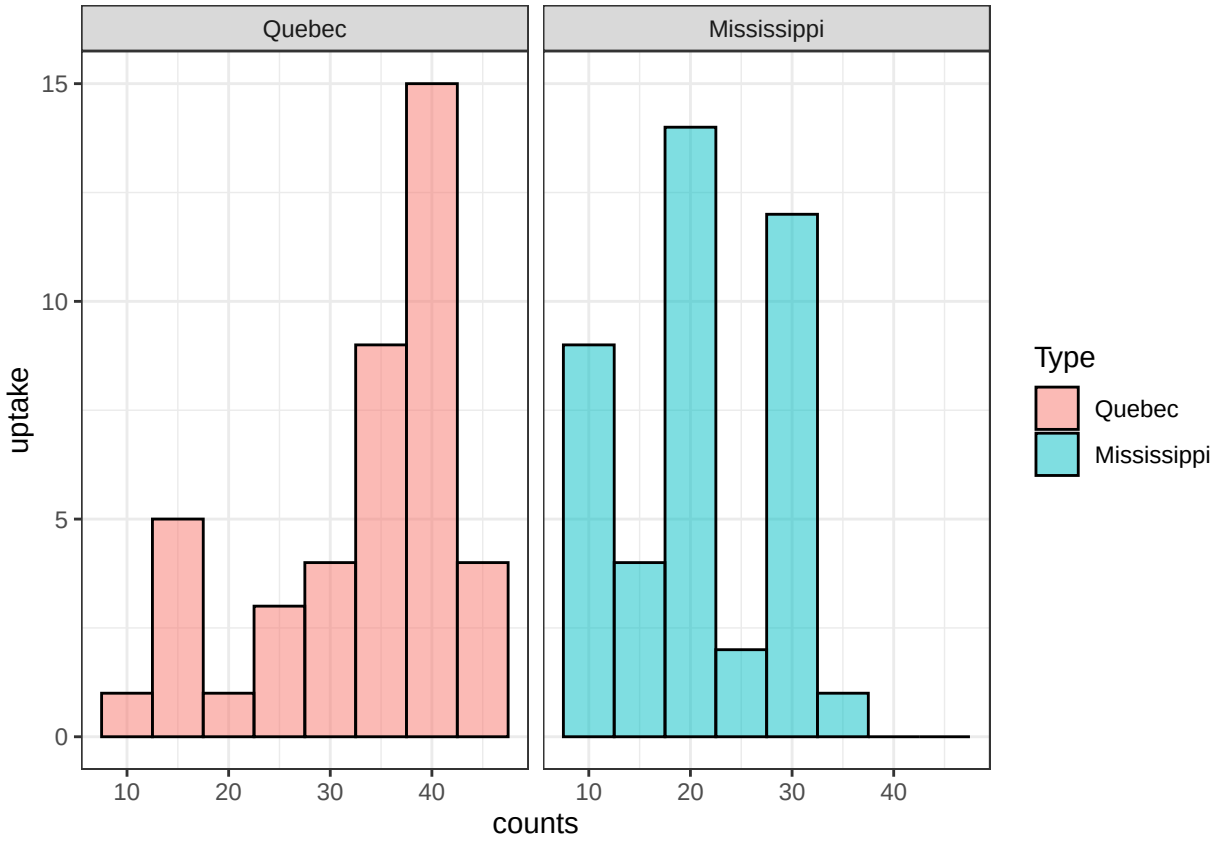
```
x <- 50 # set X to 50%
genes_50 <- names(pct_missing[pct_missing > x])
```

#d

```
# replace missing values by mean
microarray_imputed <- apply(microarray, 2, function(x) ifelse(is.na(x), mean(x, na.rm=TRUE), x))
```

#3

```
# create histograms using ggplot2 package
ggplot(CO2, aes(x=uptake, fill=Type)) +
  geom_histogram(binwidth=5, color="black", alpha=0.5) +
  facet_grid(.~Type) +
  labs(x="counts", y="uptake") +
  theme_bw()
```



I saw the distribution of this data set

Task 8

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v tibble  3.1.8     v dplyr   1.1.0
## v readr   2.1.4     v stringr 1.5.0
## v purrr   1.0.1     v forcats 1.0.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::combine() masks gridExtra::combine()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
```

```
library(tidybiology)
```

#1

#a

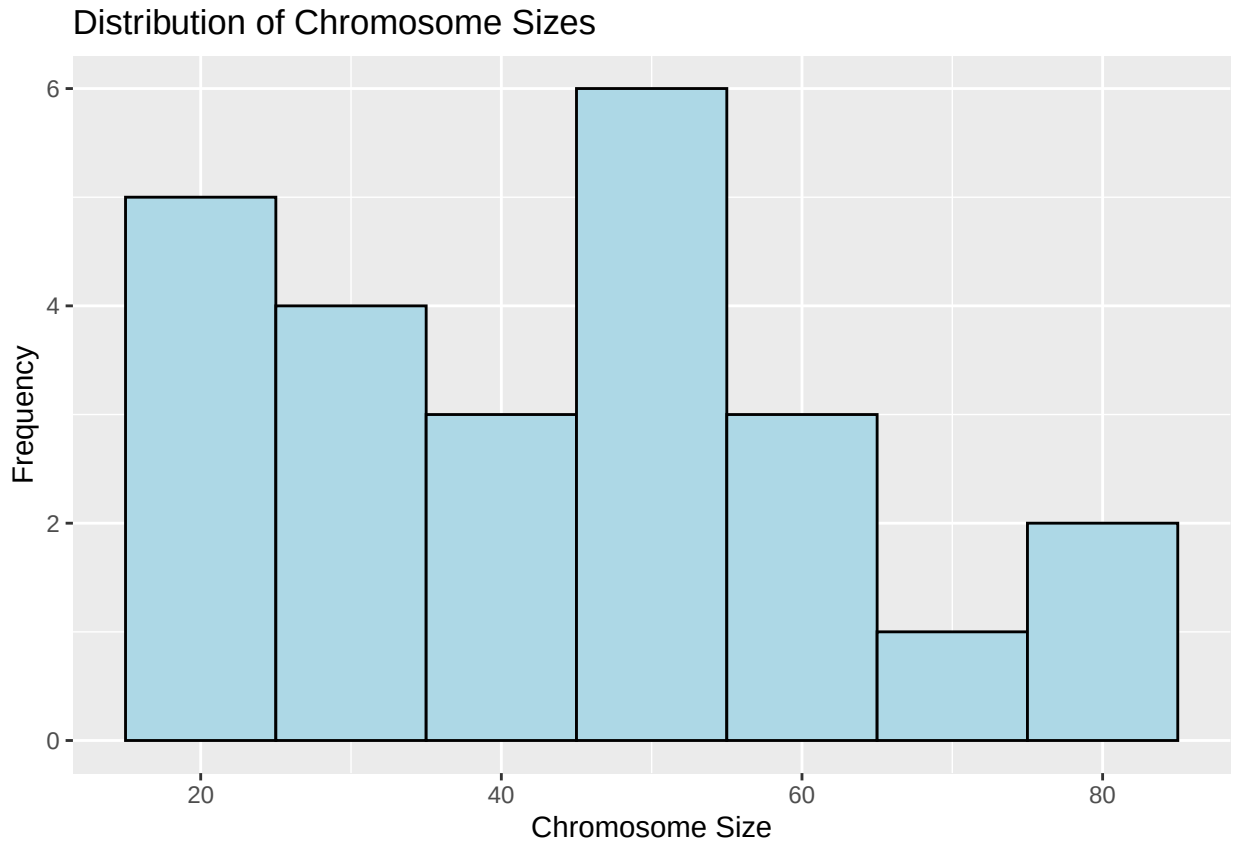
```
chromosome %>%
  summary(mean_variations = mean(variations),
            median_variations = median(variations),
            max_variations = max(variations),
            mean_protein_coding_genes = mean(protein_coding_genes),
            median_protein_coding_genes = median(protein_coding_genes),
            max_protein_coding_genes = max(protein_coding_genes),
            mean_miRNAs = mean(miRNAs),
            median_miRNAs = median(miRNAs),
            max_miRNAs = max(miRNAs))
```

```
##        id       length_mm       basepairs           variations
##  1      : 1   Min.   :16.00   Min.   : 46709983   Min.   :  211643
##  2      : 1   1st Qu.:27.75   1st Qu.: 82536402   1st Qu.: 4395298
##  3      : 1   Median :45.50   Median :133536366   Median : 6172346
##  4      : 1   Mean   :43.83   Mean   :128677910   Mean   : 6484572
##  5      : 1   3rd Qu.:55.00   3rd Qu.:162210974   3rd Qu.: 8742592
##  6      : 1   Max.   :85.00   Max.   :248956422   Max.   :12945965
##  (Other):18
##  protein_codinggenes  pseudo_genes    totallongnc_rna  totalsmallnc_rna
##  Min.   :  71.0      Min.   : 185.0   Min.   :  71.0   Min.   : 30.0
##  1st Qu.: 595.8      1st Qu.: 445.8   1st Qu.: 439.0   1st Qu.:167.0
##  Median : 836.0      Median : 590.5   Median : 633.5   Median :220.5
##  Mean   : 850.0      Mean   : 608.3   Mean   : 613.6   Mean   :208.9
##  3rd Qu.:1055.5      3rd Qu.: 772.5   3rd Qu.: 751.0   3rd Qu.:236.0
##  Max.   :2058.0      Max.   :1220.0   Max.   :1200.0   Max.   :496.0
##
##      mi_rna           r_rna           sn_rna          sno_rna
##  Min.   : 15.00   Min.   : 5.00   Min.   : 17.00   Min.   :  3.00
##  1st Qu.: 55.75   1st Qu.:13.00   1st Qu.: 49.75   1st Qu.: 36.75
##  Median : 75.00   Median :23.00   Median : 77.00   Median : 59.50
##  Mean   : 73.17   Mean   :22.08   Mean   : 81.00   Mean   : 63.38
##  3rd Qu.: 92.00   3rd Qu.:27.25   3rd Qu.:106.00   3rd Qu.: 76.00
##  Max.   :134.00   Max.   :66.00   Max.   :221.00   Max.   :145.00
##
```

```
##    miscnc_rna     centromereposition_mbp
##  Min.   :  8.00   Min.   : 12.50
##  1st Qu.: 66.50   1st Qu.: 18.73
##  Median : 94.50   Median : 38.40
##  Mean   : 92.21   Mean   : 43.36
##  3rd Qu.:107.50   3rd Qu.: 55.25
##  Max.   :192.00   Max.   :125.00
##
```

#b

```
ggplot(chromosome, aes(x = length_mm)) +
  geom_histogram(binwidth = 10, fill = "lightblue", color = "black") +
  labs(x = "Chromosome Size", y = "Frequency", title = "Distribution of Chromosome Sizes")
```



#c #protein coding genes&length

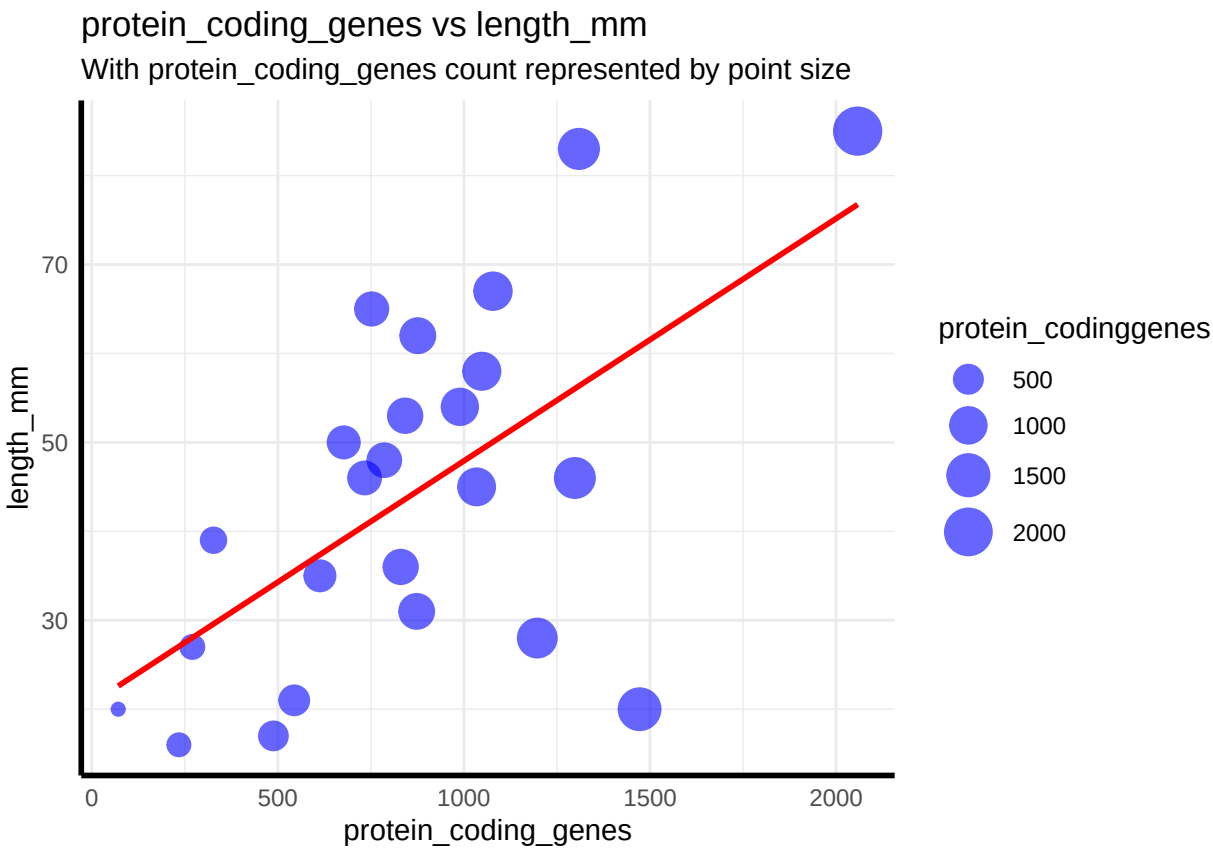
```
ggplot(chromosome, aes(x = protein_codinggenes, y = length_mm)) +
  geom_point(aes(size = protein_codinggenes), alpha = 0.6, color = "blue") +
  scale_size_continuous(range = c(2, 8)) +
  geom_smooth(method = "lm", se = FALSE, color = "red", size = 1) +
  labs(x = "protein_coding_genes",
       y = "length_mm",
       title = "protein_coding_genes vs length_mm",
       subtitle = "With protein_coding_genes count represented by point size") +
```

```
  theme_minimal() +
  theme(axis.line = element_line(color = "black", size = 1))
```

```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
```

```
## Warning: The 'size' argument of 'element_line()' is deprecated as of ggplot2 3.4.0.
## i Please use the 'linewidth' argument instead.
```

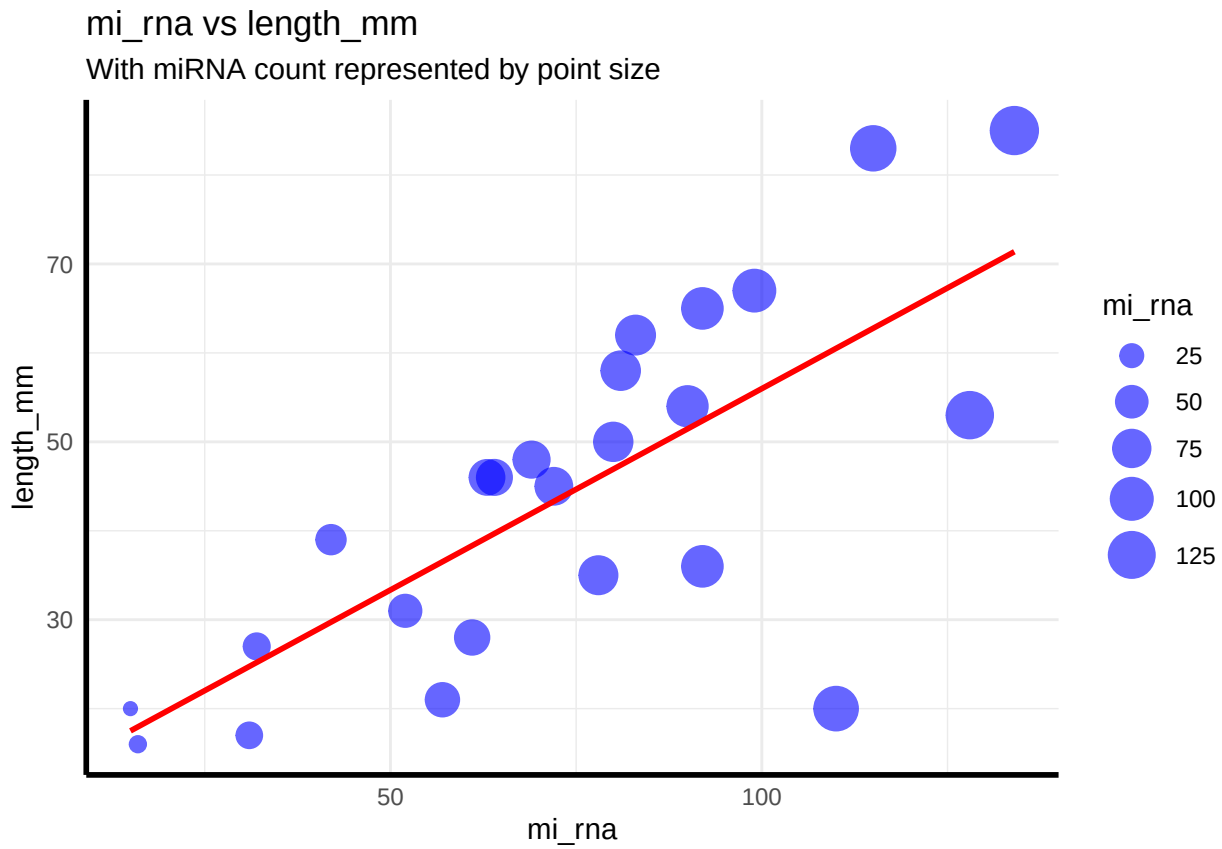
```
## 'geom_smooth()' using formula = 'y ~ x'
```



#mirna&length

```
ggplot(chromosome, aes(x = mi_rna, y = length_mm)) +
  geom_point(aes(size = mi_rna), alpha = 0.6, color = "blue") +
  scale_size_continuous(range = c(2, 8)) +
  geom_smooth(method = "lm", se = FALSE, color = "red", size = 1) +
  labs(x = "mi_rna",
       y = "length_mm",
       title = "mi_rna vs length_mm",
       subtitle = "With miRNA count represented by point size") +
  theme_minimal() +
  theme(axis.line = element_line(color = "black", size = 1))
```

```
## 'geom_smooth()' using formula = 'y ~ x'
```



#d

```
proteins %>%
  summary(mean_length = mean(length),
            median_length = median(length),
            max_length = max(length),
            mean_mass = mean(mass),
            median_mass = median(mass),
            max_mass = max(mass))
```

```
##   uniprot_id         gene_name         gene_name_alt      protein_name
##  Length:20430       Length:20430       Length:20430       Length:20430
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##  protein_name_alt     sequence            length             mass
##  Length:20430       Length:20430       Min.   :    2.0   Min.   :    260
##  Class :character   Class :character   1st Qu.:  251.0   1st Qu.:  27940
##  Mode  :character   Mode  :character   Median :  414.0   Median :  46140
##                                        Mean   :  557.2   Mean   :  62061
##                                        3rd Qu.:  669.0   3rd Qu.:  74755
##                                        Max.   :34350.0   Max.   :3816030
```

```r
ggplot(proteins, aes(x = length, y = mass)) +
  geom_point(aes(size = length), alpha = 0.6, color = "blue") +
  scale_size_continuous(range = c(2, 8)) +
  geom_smooth(method = "lm", se = FALSE, color = "red", size = 1) +
  labs(x = "length",
       y = "mass",
       title = "length vs mass") +
  theme_minimal() +
  theme(axis.line = element_line(color = "black", size = 1))
```

```
## `geom_smooth()` using formula = 'y ~ x'
```