Home Work 1

Dan Getty

2024-01-25

Set Environment

```
# configure knitr
knitr::opts_chunk$set(message = TRUE, echo = TRUE)
# directory
dir <-
'G:\\My Drive\\H Drive\\Course Work\\CERG-Data Science\\CSC_587_Advanced_Data_Mining\\HW\\HW1_DataMining
# Set the working directory.
setwd(dir)
# load ggplot2 package
library(ggplot2)
# load dplyr package
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# load tidyr package
library(tidyr)
```

Homework 1

1.a: Use read.delim function to read Su_raw_matrix.txt into a variable called su.(Notice that su has become a data frame now) Load in the data set from disk.

```
# build path to Su_raw_matrix.txt
data_file <- file.path('Su_raw_matrix.txt')
# Build data frame from the data set.
su <- read.delim(data_file, header = TRUE, sep = '\t')
```

1.b: Use mean and sd functions to find mean and standard deviation of

```
# print the Standard Deviation
sd(su$Liver_2.CEL)
```

```
## [1] 1133.352
```

```
# print the Mean
mean(su$Liver_2.CEL)
```

```
## [1] 241.8246
```

1.c: Use colMeans and colSums functions to get the average and total values of each column.

Column Means

```
colMeans(su[, -1])  # Exclude the first column
```

```
##       Brain_1.CEL       Brain_2.CEL Fetal_brain_1.CEL Fetal_brain_2.CEL
##          204.9763          315.0924          198.3439          267.6551
## Fetal_liver_1.CEL Fetal_liver_2.CEL       Liver_1.CEL       Liver_2.CEL
##          209.8722          399.1482          160.8558          241.8246
```

Column Sums

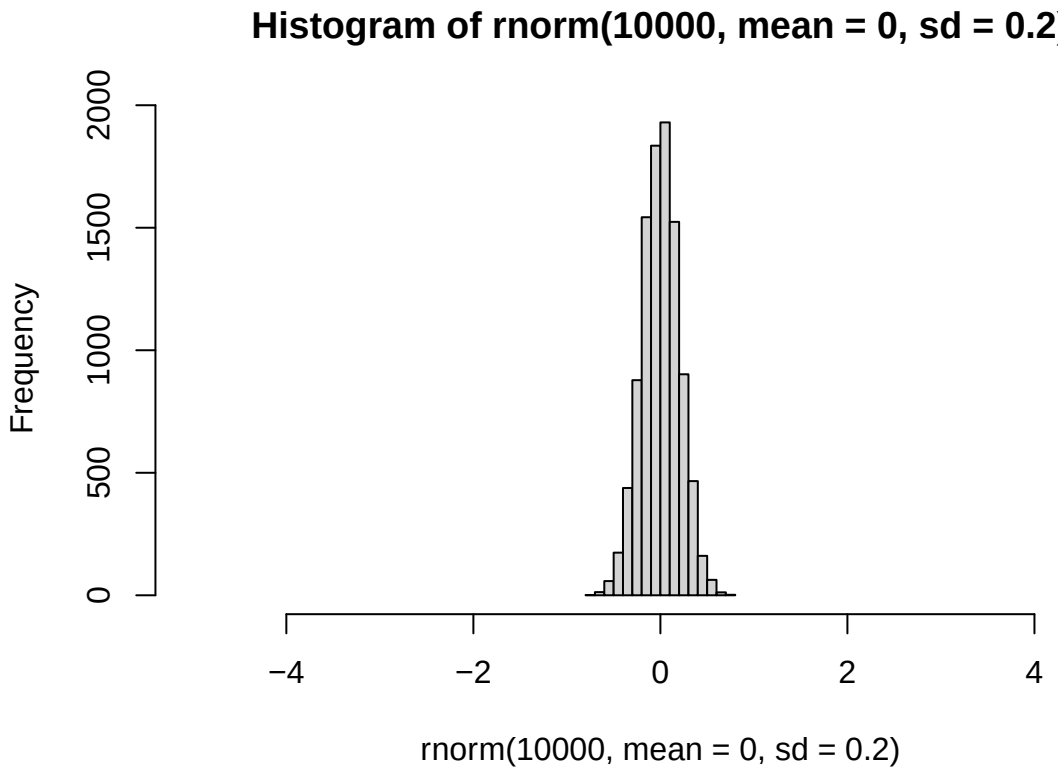
```
colSums(su[, -1])  # Exclude the first Column
```

```
##       Brain_1.CEL       Brain_2.CEL Fetal_brain_1.CEL Fetal_brain_2.CEL
##           2588031           3978357           2504290           3379413
## Fetal_liver_1.CEL Fetal_liver_2.CEL       Liver_1.CEL       Liver_2.CEL
##           2649846           5039645           2030966           3053278
```

2 Use rnorm(n, mean = 0, sd = 1) function to generate 10000 numbers for the following(mean, sigma) pairs and plot histogram for each, meaning you need to change the function parameter accordingly. Then comment on how these histograms are different from each other and state the reason. Please save your figures as image from RStudio. (Hint: to see the difference in plots you may need to set the xlim parameter in plot function to c(-5,5))

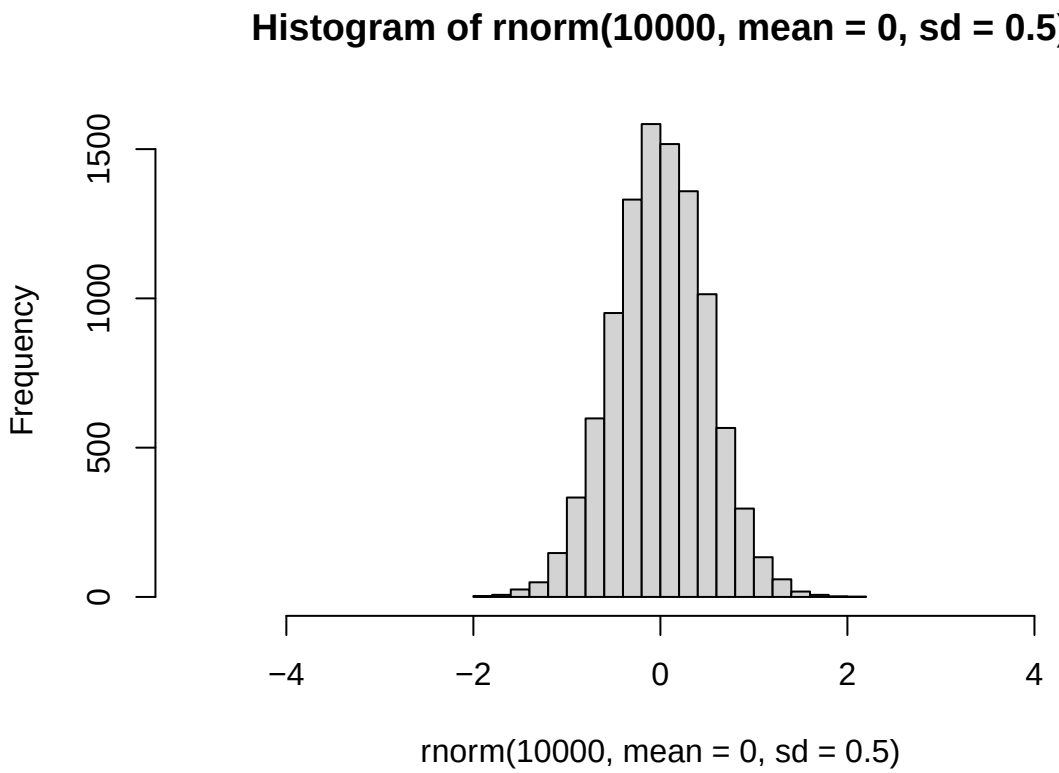
2.a: mean = 0, sigma = 0.2

```
hist(rnorm(10000, mean = 0, sd = 0.2),xlim = c(-5,5))
```



2.b: mean = 0, sigma = 0.5

```
hist(rnorm(10000, mean = 0, sd = 0.5),xlim = c(-5,5))
```



The difference between the two histograms is the standard deviation. The first being 0.2 and the second 0.5. "Standard Deviation (sigma) is a measure of how dispersed the data is in relation to the mean. Low, or small, standard deviation indicates data are clustered tightly around the mean, and high, or large, standard deviation indicates data are more spread out. A standard deviation close to zero indicates that data points are very close to the mean, whereas a larger standard deviation indicates data points are spread further away from the mean." (https://www.nlm.nih.gov/oet/ed/stats/02-900.html)

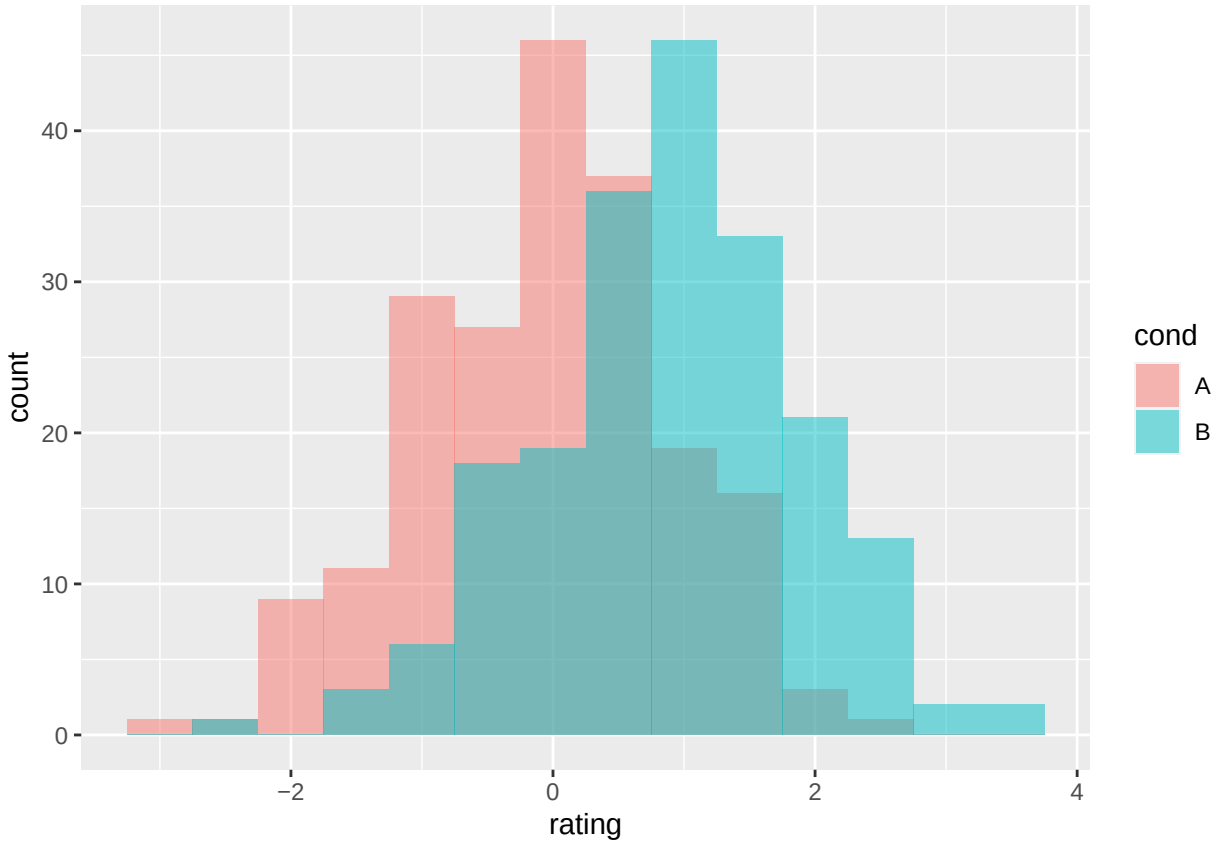
3 Perform the steps below with "dat" dataframe which is just a sample data for you to observe how each plot function (3b through 3e) works. Notice that you need to have ggplot2 library installed on your system. Please refer slides how to install and import a library. Installation is done only once, but you need to import the library every time you need it by saying library(ggplot2). Then run the following commands for questions from 3a through 3e and observe how the plots are generated first.

3.a:

```
dat <- data.frame(cond = factor(rep(c("A","B"), each=200)), rating = c(rnorm(200),rnorm(200, mean=.8)))
```

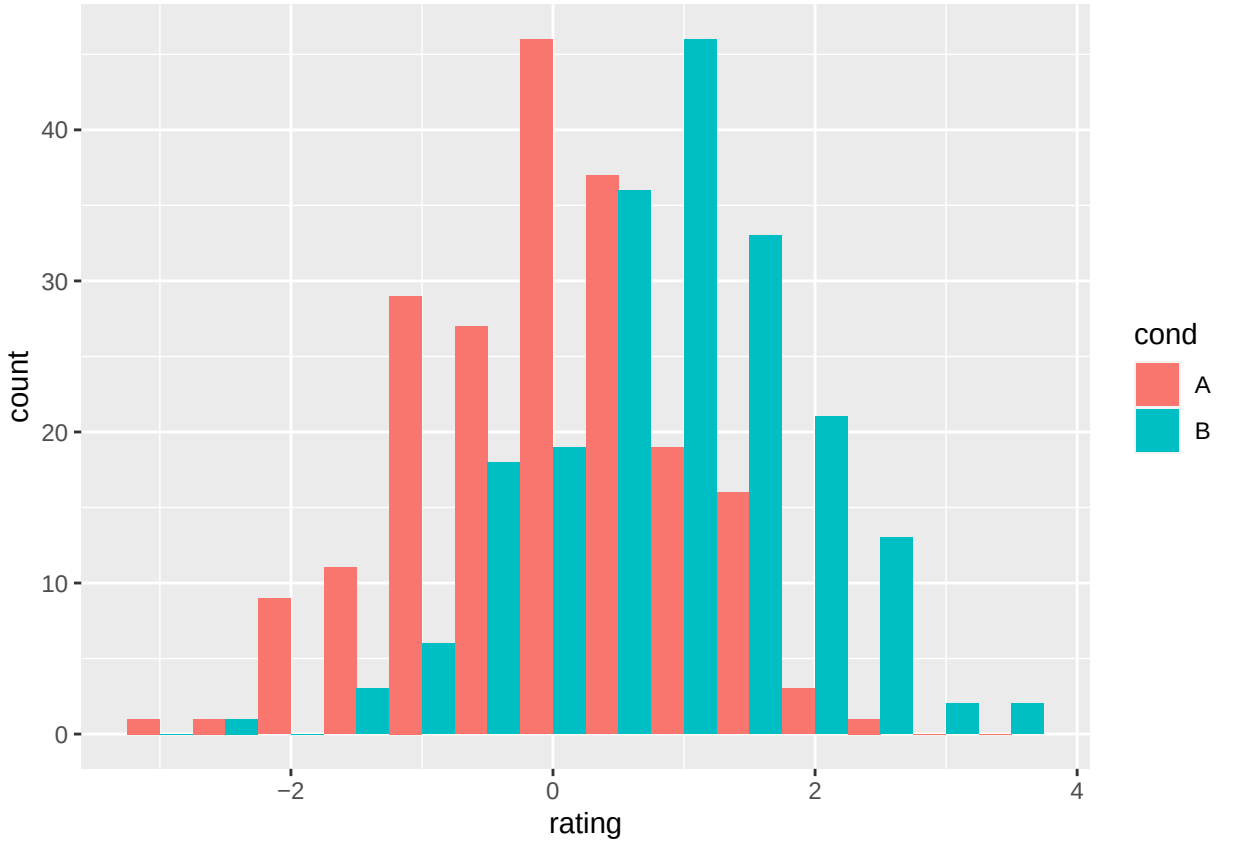
3.b: Overlaid histograms

```
ggplot(dat, aes(x=rating, fill=cond)) + geom_histogram(binwidth=.5, alpha=.5, position="identity")
```



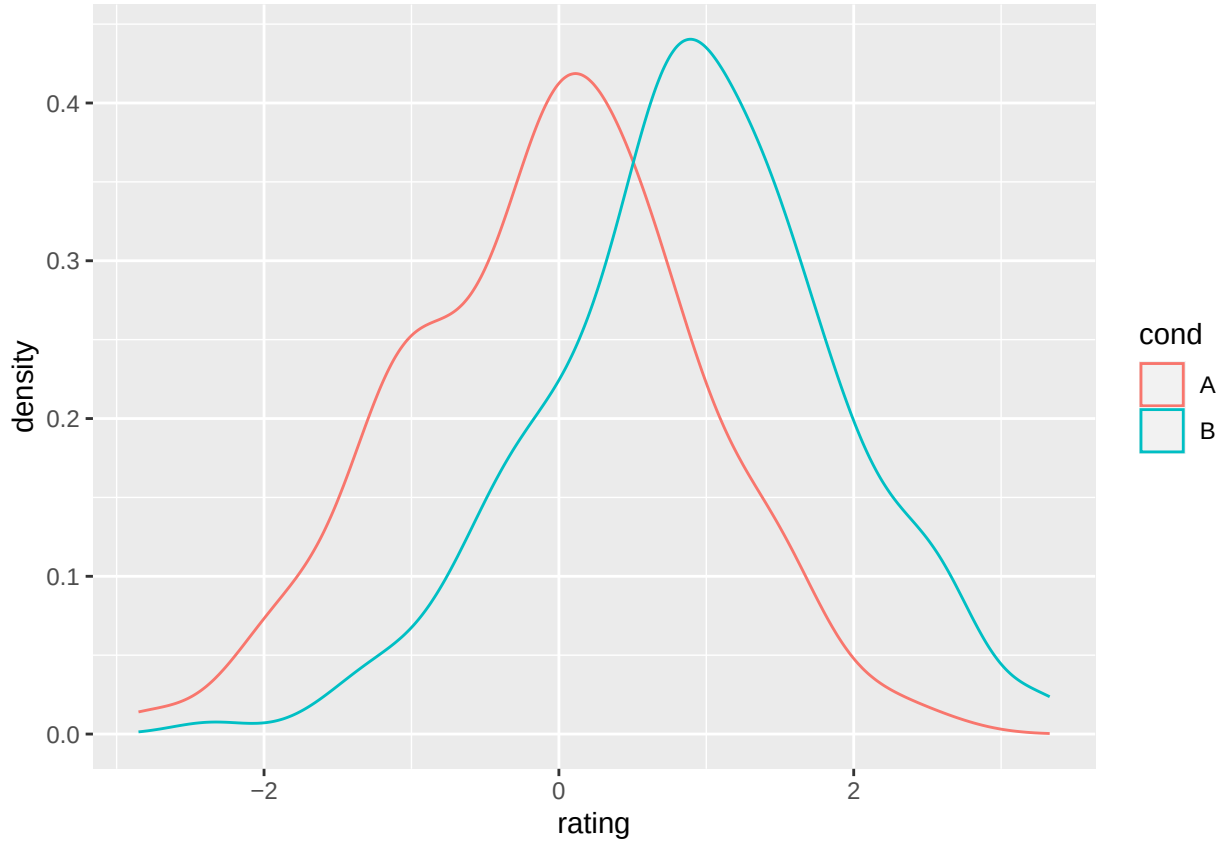
3.c: Interleaved Histograms

```
ggplot(dat, aes(x=rating, fill=cond)) + geom_histogram(binwidth=.5, position="dodge")
```



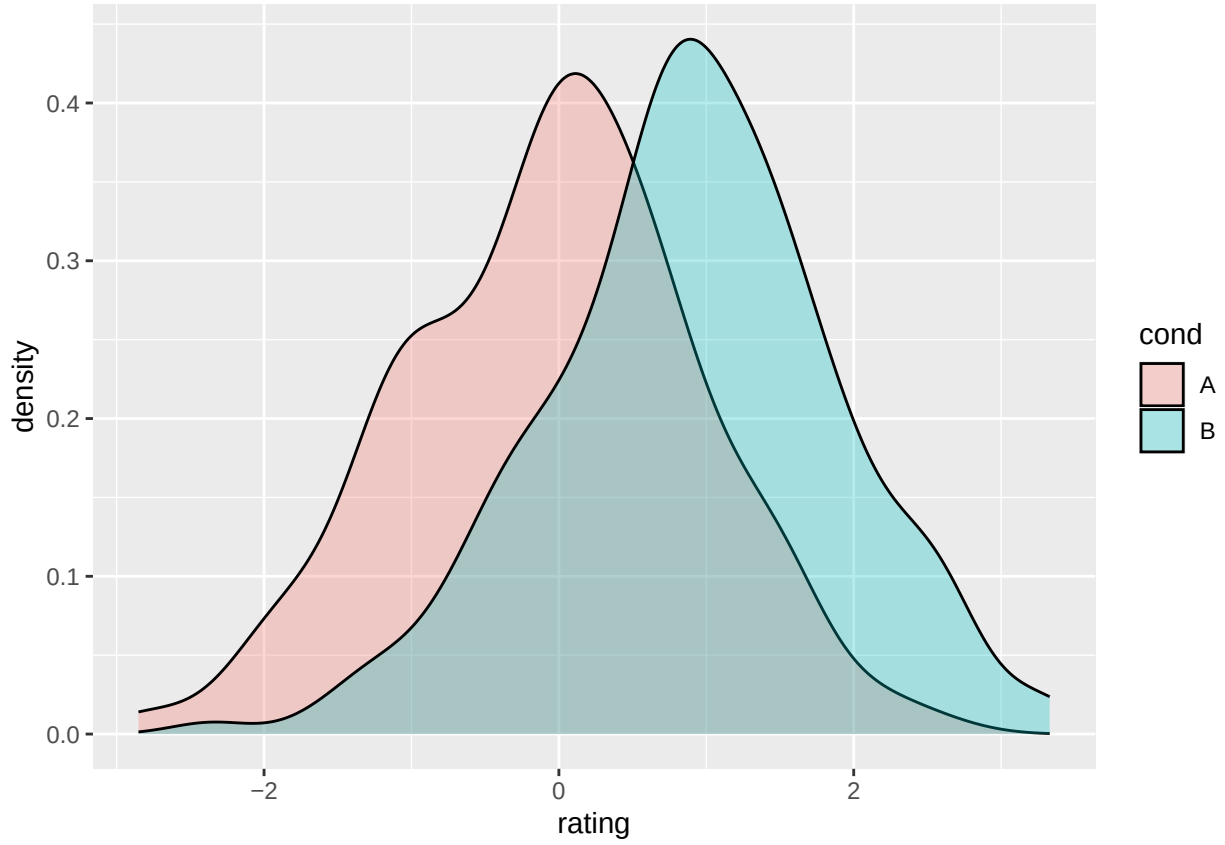
3.d: Density plots

```
ggplot(dat, aes(x=rating, colour=cond)) + geom_density()
```



3.e: Density plots with semitransparent fill

```
ggplot(dat, aes(x=rating, fill=cond)) + geom_density(alpha=.3)
```



3.f: Read "diabetes_train.csv" into a variable called diabetes and apply the same functions 3b through 3e for the mass attribute of diabetes and save the images. (Hint: instead of cond above, use the class attribute to color your groups. When you have fill option, your plots should show same type of chart for both groups in different colors on the same figure. Keep in mind that diabetes and dat are both DataFrames)

Load Diabetes Data

```
df_path <- file.path('diabetes_train.csv')
# read csv file
diabetes <- read.csv(df_path, header = TRUE, sep = ',')
glimpse(diabetes)
```

```
## Rows: 758
## Columns: 9
## $ preg  <int> 6, 1, 8, 1, 0, 5, 3, 10, 2, 8, 4, 10, 10, 1, 5, 7, 0, 7, 1, 1, 3~
## $ plas  <int> 148, 85, 183, 89, 137, 116, 78, 115, 197, 125, 110, 168, 139, 18~
## $ pres  <int> 72, 66, 64, 66, 40, 74, 50, 0, 70, 96, 92, 74, 80, 60, 72, 0, 84~
## $ skin  <int> 35, 29, 0, 23, 35, 0, 32, 0, 45, 0, 0, 0, 0, 23, 19, 0, 47, 0, 3~
## $ insu  <int> 0, 0, 0, 94, 168, 0, 88, 0, 543, 0, 0, 0, 0, 846, 175, 0, 230, 0~
## $ mass  <dbl> 33.6, 26.6, 23.3, 28.1, 43.1, 25.6, 31.0, 35.3, 30.5, 0.0, 37.6,~
## $ pedi  <dbl> 0.627, 0.351, 0.672, 0.167, 2.288, 0.201, 0.248, 0.134, 0.158, 0~
## $ age   <int> 50, 31, 32, 21, 33, 30, 26, 29, 53, 54, 30, 34, 57, 59, 51, 32, ~
## $ class <chr> "tested_positive", "tested_negative", "tested_positive", "tested~
```

```
dbclass = diabetes$class
glimpse(dbclass)
```

```
##  chr [1:758] "tested_positive" "tested_negative" "tested_positive" ...
```

```
mass = diabetes$mass
glimpse(mass)
```

```
##  num [1:758] 33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
```

```
massmean <- mean(diabetes$mass)
glimpse(massmean)
```

```
##  num 32
```

```
masssd <- sd(diabetes$mass)
glimpse(masssd)
```

```
##  num 7.91
```

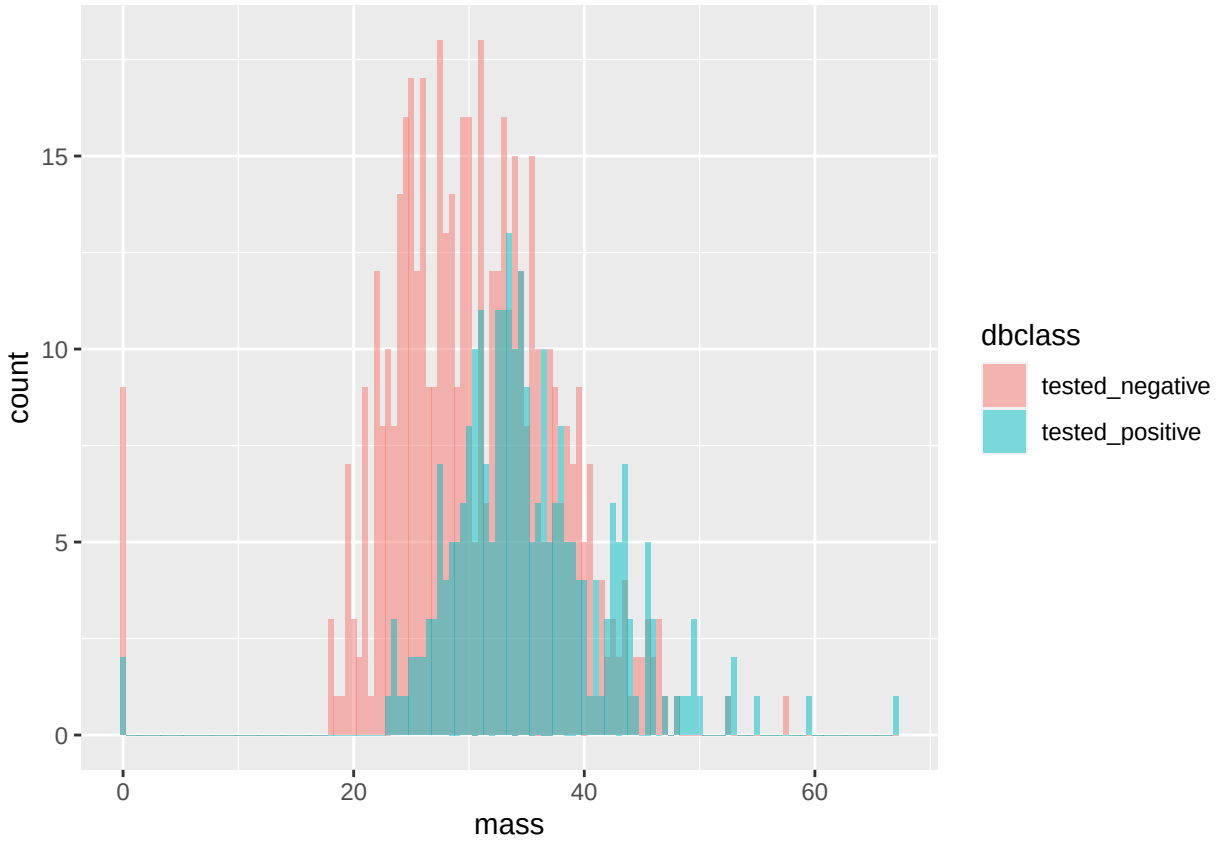
```
dbmass <- data.frame(mass, outcome=dbclass)
glimpse(dbmass)
```

```
## Rows: 758
## Columns: 2
## $ mass    <dbl> 33.6, 26.6, 23.3, 28.1, 43.1, 25.6, 31.0, 35.3, 30.5, 0.0, 37.~
## $ outcome <chr> "tested_positive", "tested_negative", "tested_positive", "test~
```

3.f.b: Overlaid histograms:

- These histograms show the distribution of the mass attribute for the two classes of diabetes(tested_positive/tested_negative). The x-axis is the mass and the y-axis is the count of the mass.
- The fill color is used to differentiate the two classes of diabetes. The binwidth parameter is used to set the width of the bins to .5.
- The alpha parameter is used to make the fill color semi-transparent.
- The position "identity" parameter is used to overlay the histograms.

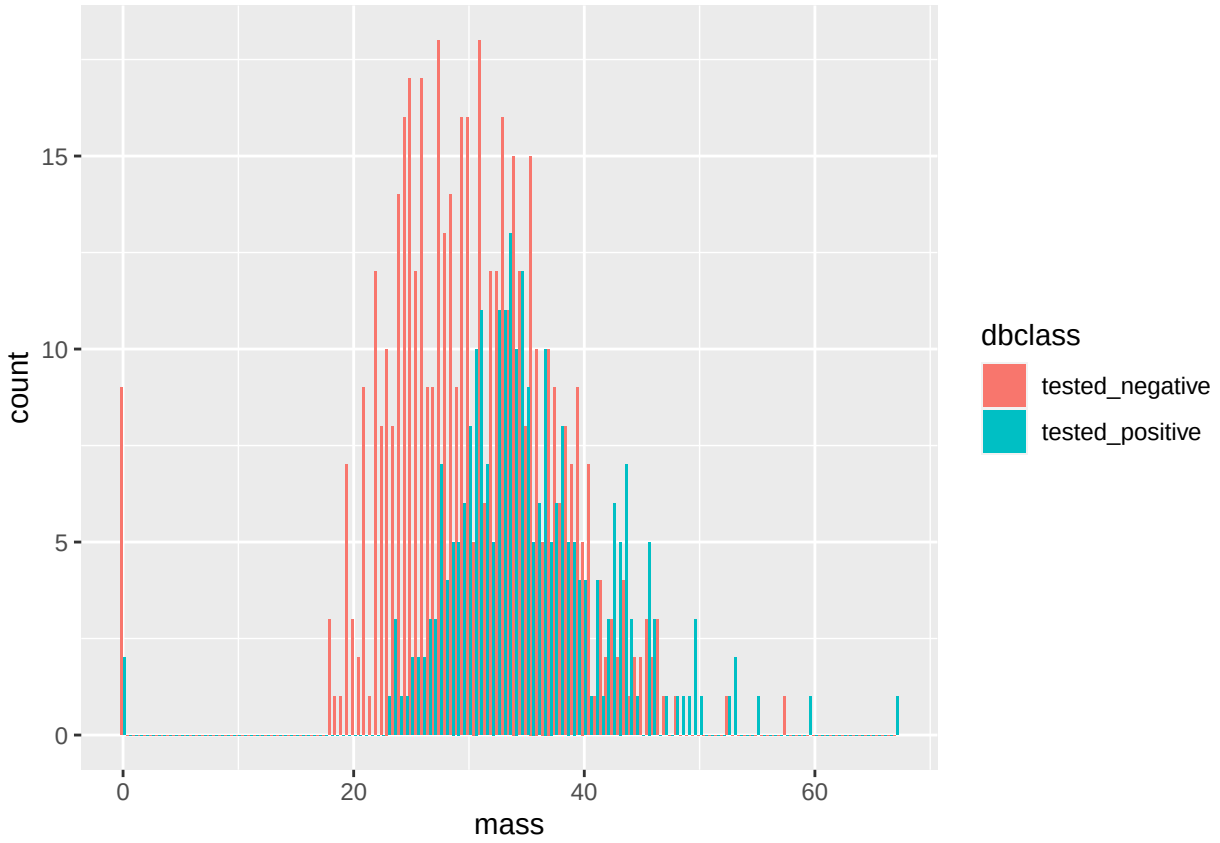
```
ggplot(dbmass, aes(x=mass, fill=dbclass)) + geom_histogram(binwidth=.5, alpha=.5, position="identity")
```



3.f.c: Interleaved Histograms:

- These histograms show the distribution of the mass attribute for the two classes of diabetes(tested_positive/tested_negative).
- The x-axis is the mass and the y-axis is the count of the mass. The fill color is used to differentiate the two classes of diabetes.
- The binwidth parameter is used to set the width of the bins to .5.
- No Alpha parameter is used because of the dodge positioning causing the distributions not to overlap
- The position "dodge" parameter is used to interleave the histograms.

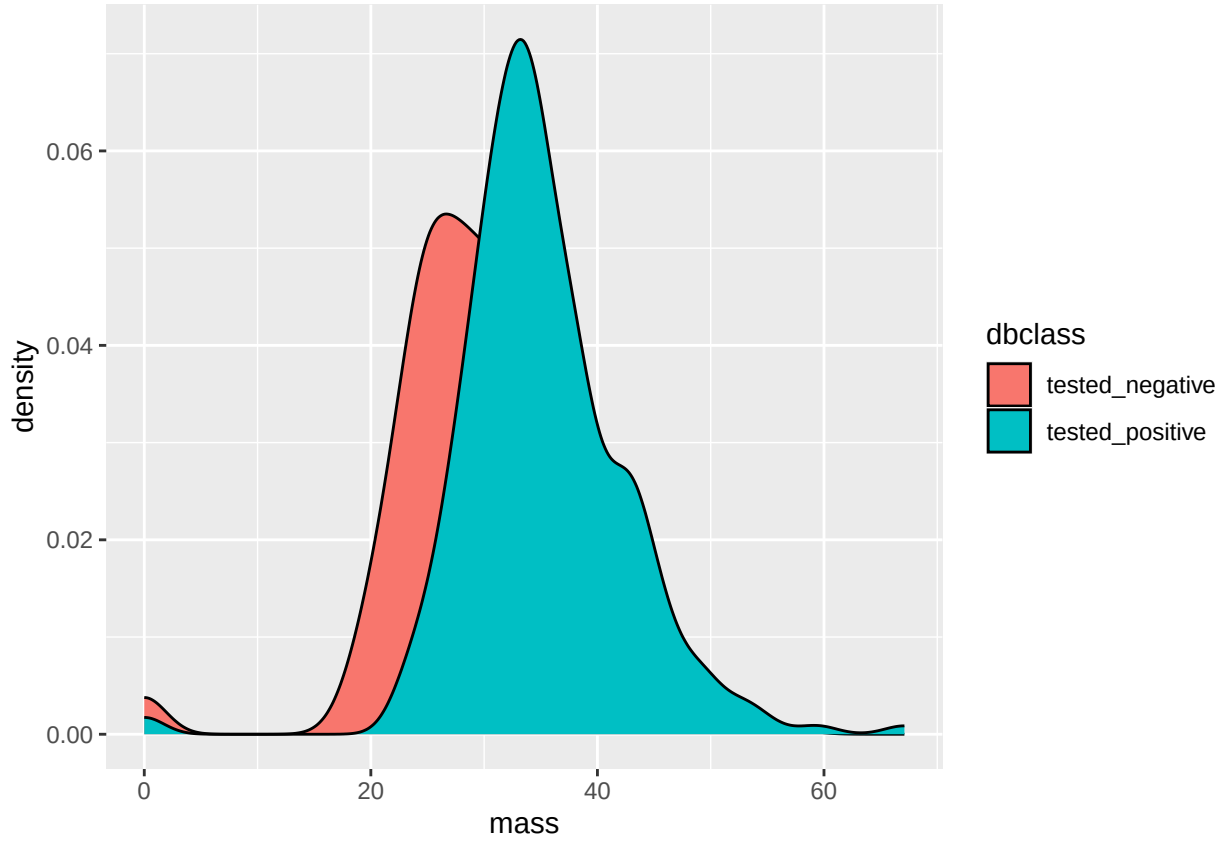
```
ggplot(dbmass, aes(x=mass, fill=dbclass)) + geom_histogram(binwidth=.5, position="dodge")
```



3.f.d: Density plots:

- These density plots show the distribution of the mass attribute for the two classes of diabetes(tested_positive/tested_negative).
- The x-axis is the mass and the y-axis is the density of the mass.
- The fill color is used to differentiate the two classes of diabetes.

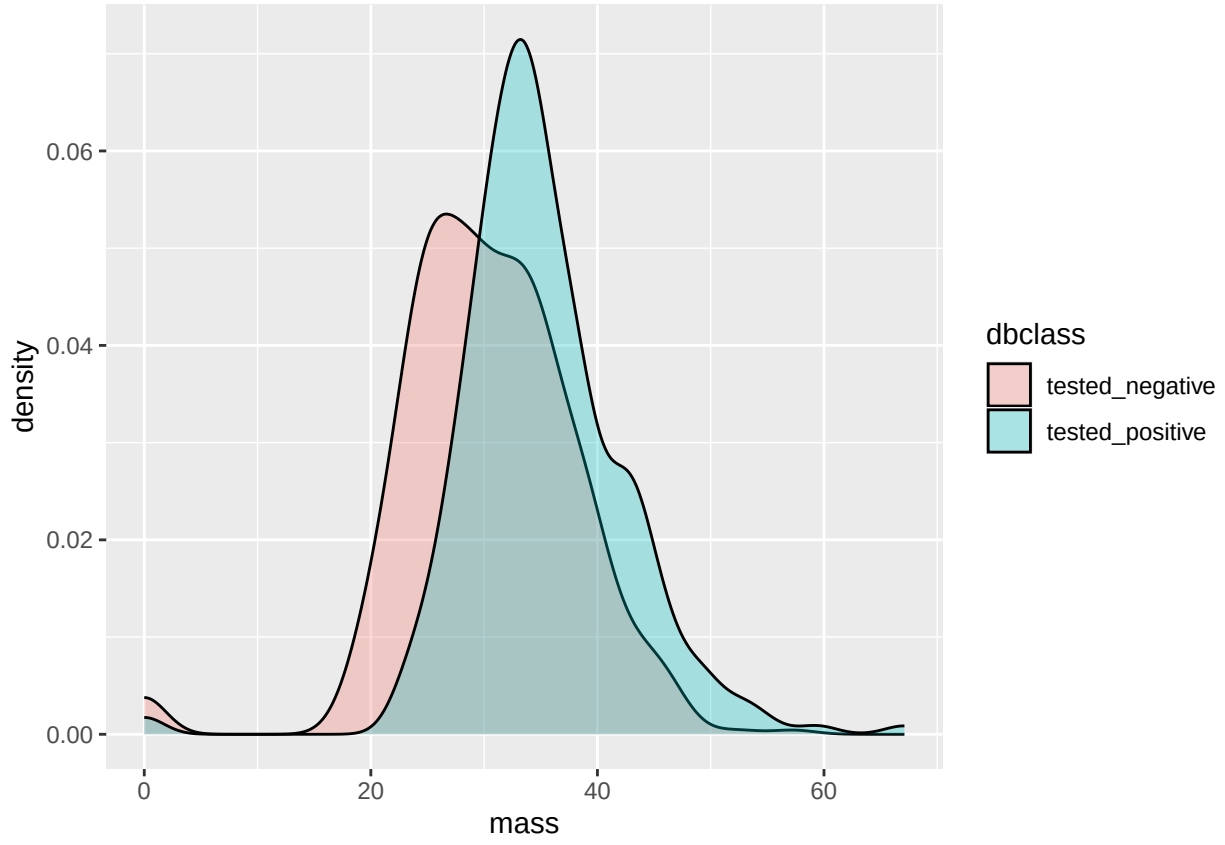
```
ggplot(dbmass, aes(x=mass, fill=dbclass)) + geom_density()
```



3.f.e: Density plots with semitransparent fill:

- These density plots show the distribution of the mass attribute for the two classes of diabetes(tested_positive/tested_negative).
- The x-axis is the mass and the y-axis is the density of the mass.
- The fill color is used to differentiate the two classes of diabetes.
- The alpha parameter is used to make the fill color semi-transparent.

```
ggplot(dbmass, aes(x=mass, fill=dbclass)) + geom_density(alpha=.3)
```



4 Read the titanic.csv file from DATA folder to a variable named passengers and perform the following steps and explain the operation very briefly. Please make sure you have tidyverse installed on your system and you may specifically need to import the tidyr library. Otherwise, the chain of operations through "piping" won't work.

4 Environment Setup

```
# build path to titanic.csv
tit_file <- file.path("titanic.csv")
# Build data frame from the data set.
passengers <- read.delim(tit_file, header = TRUE, sep = ',')
glimpse(passengers)
```

```
## Rows: 891
## Columns: 13
## $ X           <int> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, ~
## $ PassengerId <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,~
## $ Survived    <int> 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1~
## $ Pclass      <chr> "3", "1", "3", "1", "3", "3", "1", "3", "3", "2", "3", "1"~
## $ Name        <chr> "Braund, Mr. Owen Harris", "Cumings, Mrs. John Bradley (Fl~
## $ Sex         <chr> "male", "female", "female", "female", "male", "male", "mal~
## $ Age         <dbl> 22, 38, 26, 35, 35, NA, 54, 2, 27, 14, 4, 58, 20, 39, 14, ~
## $ SibSp       <int> 1, 1, 0, 1, 0, 0, 0, 3, 0, 1, 1, 0, 0, 1, 0, 0, 4, 0, 1, 0~
## $ Parch       <int> 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 1, 0, 0, 5, 0, 0, 1, 0, 0, 0~
## $ Ticket      <chr> "A/5 21171", "PC 17599", "STON/O2. 3101282", "113803", "37~
## $ Fare        <dbl> 7.2500, 71.2833, 7.9250, 53.1000, 8.0500, 8.4583, 51.8625,~
## $ Cabin       <chr> "", "C85", "", "C123", "", "", "E46", "", "", "", "G6", "C~
## $ Embarked    <chr> "S", "C", "S", "S", "S", "Q", "S", "S", "S", "C", "S", "S"~
```

4.a passengers %>% drop_na() %>% summary():

This code starts with the passengers data frame then pipes it to the drop_na() function which drops all rows with NA values. The resulting data frame is then piped to the summary() function which returns a summary of the data frame.

```
passengers %>% drop_na() %>% summary()
```

```
##        X          PassengerId       Survived         Pclass
##  Min.   :  0.0   Min.   :  1.0   Min.   :0.0000   Length:714
##  1st Qu.:221.2   1st Qu.:222.2   1st Qu.:0.0000   Class :character
##  Median :444.0   Median :445.0   Median :0.0000   Mode  :character
##  Mean   :447.6   Mean   :448.6   Mean   :0.4062
##  3rd Qu.:676.8   3rd Qu.:677.8   3rd Qu.:1.0000
##  Max.   :890.0   Max.   :891.0   Max.   :1.0000
##      Name               Sex                 Age            SibSp
##  Length:714         Length:714         Min.   : 0.42   Min.   :0.0000
##  Class :character   Class :character   1st Qu.:20.12   1st Qu.:0.0000
##  Mode  :character   Mode  :character   Median :28.00   Median :0.0000
##                                        Mean   :29.70   Mean   :0.5126
##                                        3rd Qu.:38.00   3rd Qu.:1.0000
##                                        Max.   :80.00   Max.   :5.0000
##      Parch           Ticket               Fare           Cabin
##  Min.   :0.0000   Length:714         Min.   :  0.00   Length:714
##  1st Qu.:0.0000   Class :character   1st Qu.:  8.05   Class :character
##  Median :0.0000   Mode  :character   Median : 15.74   Mode  :character
##  Mean   :0.4314                      Mean   : 34.69
##  3rd Qu.:1.0000                      3rd Qu.: 33.38
##  Max.   :6.0000                      Max.   :512.33
##    Embarked
##  Length:714
##  Class :character
##  Mode  :character
##
##
##
```

4.b passengers %>% filter(Sex == “male”) This code starts with the passengers data frame then pipes it to the filter function at which point the sex is filtered to males only. The Glimpse funciton was added by me to make the output more readable.

```
passengers %>% filter(Sex == "male")  %>% glimpse()
```

```
## Rows: 577
## Columns: 13
## $ X           <int> 0, 4, 5, 6, 7, 12, 13, 16, 17, 20, 21, 23, 26, 27, 29, 30,~
## $ PassengerId <int> 1, 5, 6, 7, 8, 13, 14, 17, 18, 21, 22, 24, 27, 28, 30, 31,~
## $ Survived    <int> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1~
## $ Pclass      <chr> "3", "3", "3", "1", "3", "3", "3", "3", "2", "2", "2", "1"~
## $ Name        <chr> "Braund, Mr. Owen Harris", "Allen, Mr. William Henry", "Mo~
## $ Sex         <chr> "male", "male", "male", "male", "male", "male", "male", "m~
## $ Age         <dbl> 22, 35, NA, 54, 2, 20, 39, 2, NA, 35, 34, 28, NA, 19, NA, ~
## $ SibSp       <int> 1, 0, 0, 0, 3, 0, 1, 4, 0, 0, 0, 0, 0, 3, 0, 0, 0, 1, 1, 0~
## $ Parch       <int> 0, 0, 0, 0, 1, 0, 5, 1, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0~
## $ Ticket      <chr> "A/5 21171", "373450", "330877", "17463", "349909", "A/5. ~
## $ Fare        <dbl> 7.2500, 8.0500, 8.4583, 51.8625, 21.0750, 8.0500, 31.2750,~
## $ Cabin       <chr> "", "", "", "E46", "", "", "", "", "", "", "D56", "A6", ""~
## $ Embarked    <chr> "S", "S", "Q", "S", "S", "S", "S", "Q", "S", "S", "S", "S"~
```

4.c passengers %>% arrange(desc(Fare)) This code starts with the passengers data frame then pipes it to the arrange function at which point the data frame is sorted in descending order by the Fare column. The Glimpse funciton was added by me to make the output more readable.

```
passengers %>% arrange(desc(Fare)) %>% glimpse()
```

```
## Rows: 891
## Columns: 13
## $ X           <int> 258, 679, 737, 27, 88, 341, 438, 311, 742, 118, 299, 380, ~
## $ PassengerId <int> 259, 680, 738, 28, 89, 342, 439, 312, 743, 119, 300, 381, ~
## $ Survived    <int> 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1~
## $ Pclass      <chr> "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", "1", "?"~
## $ Name        <chr> "Ward, Miss. Anna", "Cardeza, Mr. Thomas Drake Martinez", ~
## $ Sex         <chr> "female", "male", "male", "male", "female", "female", "mal~
## $ Age         <dbl> 35, 36, 35, 19, 23, 24, 64, 18, 21, 24, 50, 42, NA, 18, 38~
## $ SibSp       <int> 0, 0, 0, 3, 3, 3, 1, 2, 2, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0~
## $ Parch       <int> 0, 1, 0, 2, 2, 2, 4, 2, 2, 1, 1, 0, 0, 0, 0, 0, 2, 1, 0, 1~
## $ Ticket      <chr> "PC 17755", "PC 17755", "PC 17755", "19950", "19950", "199~
## $ Fare        <dbl> 512.3292, 512.3292, 512.3292, 263.0000, 263.0000, 263.0000~
## $ Cabin       <chr> "", "B51 B53 B55", "B101", "C23 C25 C27", "C23 C25 C27", "~
## $ Embarked    <chr> "C", "C", "C", "S", "S", "S", "S", "C", "C", "C", "C", "C"~
```

4.d passengers %>% mutate(FamSize = Parch + SibSp) This code starts with the passengers data frame then pipes it to the mutate function at which point a new column is created called FamSize which is the sum of the Parch and SibSp columns. The Glimpse funciton was added by me to make the output more readable.

```
passengers %>% mutate(FamSize = Parch + SibSp) %>% glimpse()
```

```
## Rows: 891
## Columns: 14
## $ X           <int> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, ~
## $ PassengerId <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,~
## $ Survived    <int> 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1~
## $ Pclass      <chr> "3", "1", "3", "1", "3", "3", "1", "3", "3", "2", "3", "1"~
## $ Name        <chr> "Braund, Mr. Owen Harris", "Cumings, Mrs. John Bradley (Fl~
## $ Sex         <chr> "male", "female", "female", "female", "male", "male", "mal~
## $ Age         <dbl> 22, 38, 26, 35, 35, NA, 54, 2, 27, 14, 4, 58, 20, 39, 14, ~
## $ SibSp       <int> 1, 1, 0, 1, 0, 0, 0, 3, 0, 1, 1, 0, 0, 1, 0, 0, 4, 0, 1, 0~
## $ Parch       <int> 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 1, 0, 0, 5, 0, 0, 1, 0, 0, 0~
## $ Ticket      <chr> "A/5 21171", "PC 17599", "STON/O2. 3101282", "113803", "37~
## $ Fare        <dbl> 7.2500, 71.2833, 7.9250, 53.1000, 8.0500, 8.4583, 51.8625,~
## $ Cabin       <chr> "", "C85", "", "C123", "", "", "E46", "", "", "", "G6", "C~
## $ Embarked    <chr> "S", "C", "S", "S", "S", "Q", "S", "S", "S", "C", "S", "S"~
## $ FamSize     <int> 1, 1, 0, 1, 0, 0, 0, 4, 2, 1, 2, 0, 0, 6, 0, 0, 5, 0, 1, 0~
```

4.e passengers %>% group_by(Sex) %>% summarise(meanFare = mean(Fare), numSurv = sum(Survived)) This code starts with the passengers data frame then pipes it to the group_by function at which point the data frame will be filtered by sex. The data frame is then piped to the summarise function at which point the meanFare and numSurv columns are created. The meanFare column is the mean of the Fare column and the numSurv column is the sum of the Survived column. The The Glimpse funciton funciton was added by me to make the output more readable.

```
passengers %>% group_by(Sex) %>% summarise(meanFare = mean(Fare), numSurv = sum(Survived)) %>% glimpse()
```

```
## Rows: 2
## Columns: 3
## $ Sex      <chr> "female", "male"
## $ meanFare <dbl> 44.47982, 25.52389
## $ numSurv  <int> 233, 109
```

5 Using quantile(), calculate 10th; 30th; 50th; 60th percentiles of skin attribute of diabetes data.
These are representative of the 10th, 30th, 50th, and 60th percentiles of the skin attribute of diabetes data. Meaning that 10% of the data is less than the 10th percentile, 30% of the data is less than the 30th percentile, etc.

```
quantile(diabetes$skin, probs = c(0.1, 0.3, 0.5, 0.6))
```

```
## 10% 30% 50% 60%
##   0  10  23  27
```