CSC487: Data Mining - Homework #1

Cason Konzer
github

---

1. Use `Su_raw_matrix.txt` for the following questions *(30 points)*.

(a) Use `read.delim` function to read `Su_raw_matrix.txt` into a variable called `su`. (Notice that `su` has become a data frame

```
su <- read.delim(file = "Data/Su_raw_matrix.txt")
```

(b) Use `mean` and `sd` functions to find mean and standard deviation of `Liver_2.CEL` column.

```
m_sd <- function(col)
{
    m <- mean(col)
    sd <- sd(col)
    return <- rbind(m, sd)
    colnames(return) <- ""
    print(return)
    return
}
p_1_b <- m_sd(col = su$Liver_2.CEL)
```

```
##
## m   241.8246
## sd 1133.3523
```

(c) Use `colMeans` and `colSums` functions to get the average and total values of each column.

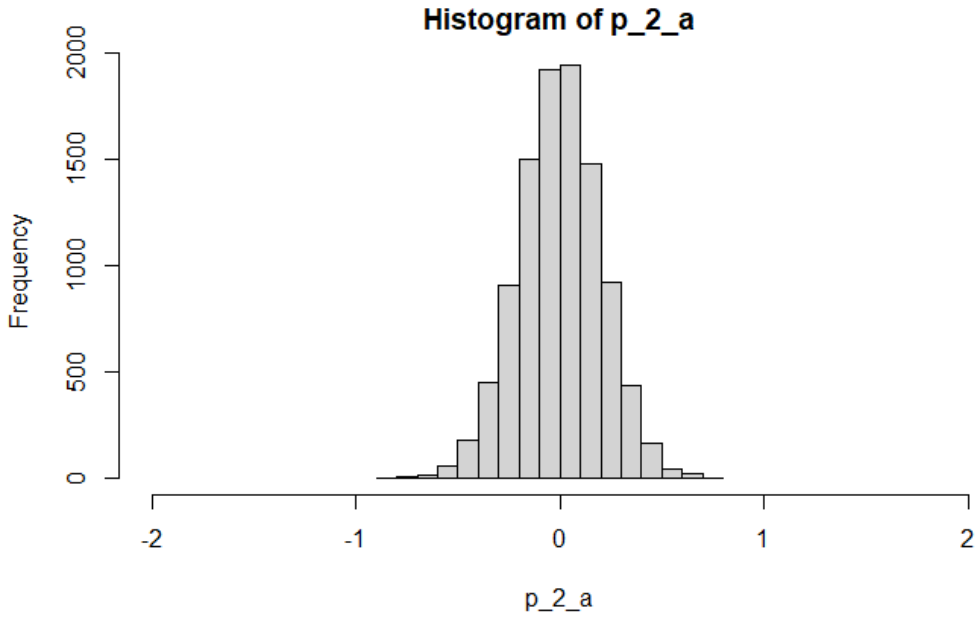
```
cm_cs <- function(df)
{
    cm <- colMeans(df)
    cs <- colSums(df)
    return <- rbind(t(cm), t(cs))
    rownames(return) <- c("m", "sd")
    print(return)
    return
}
p_1_c <- cm_cs(df = su)
```

```
##     Brain_1.CEL  Brain_2.CEL Fetal_brain_1.CEL Fetal_brain_2.CEL
## m      204.9763     315.0924          198.3439          267.6551
## sd 2588031.1500 3978356.6500      2504289.5500      3379413.0500
##    Fetal_liver_1.CEL Fetal_liver_2.CEL  Liver_1.CEL  Liver_2.CEL
## m           209.8722          399.1482     160.8558     241.8246
## sd      2649846.0000      5039644.7500 2030965.7500 3053277.5500
```

2. Use `rnorm(n, mean = 0, sd = 1)` function in R 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. *(20 points)*.

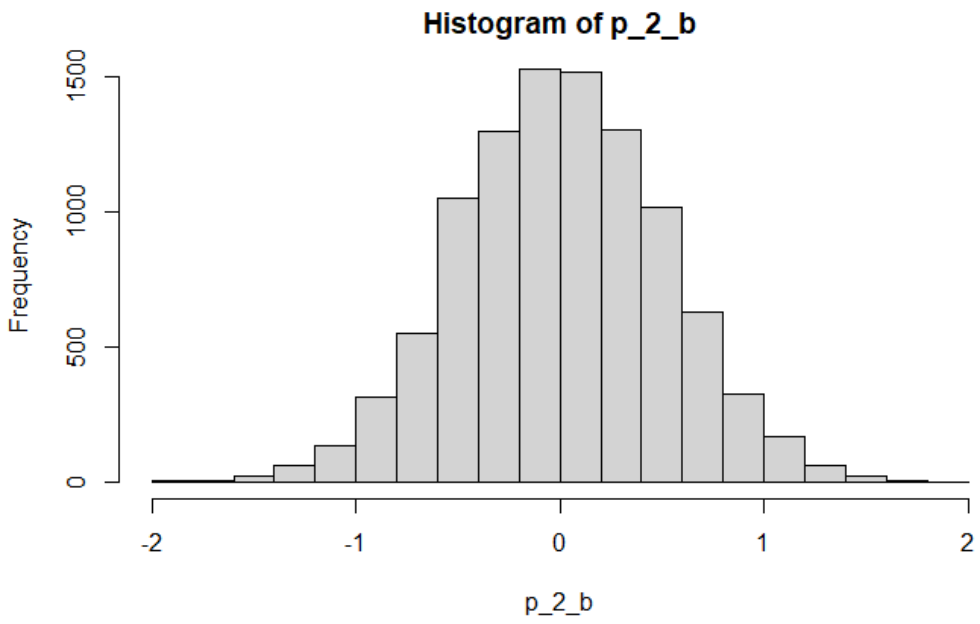
(a) `mean=0, sigma=0.2`

```
p_2_a <- rnorm(n = 10000, mean = 0, sd = 0.2)
hist(p_2_a, xlim = c(-2,2))
```



(b) `mean=0, sigma=0.5`

```
p_2_b <- rnorm(n = 10000, mean = 0, sd = 0.5)
hist(p_2_b, xlim = c(-2,2))
```



(*) Compare and Contrast

We can clearly see that p_2_a has a much tighter distribution that p_2_b.

Note: This is because , $\sigma = 0.2$, has a smaller standard deviation than p_2_b, $\sigma = 0.5$.

We also can see both samples have sample mean around 0 as they were drawn from a random normal distribution with population mean 0.

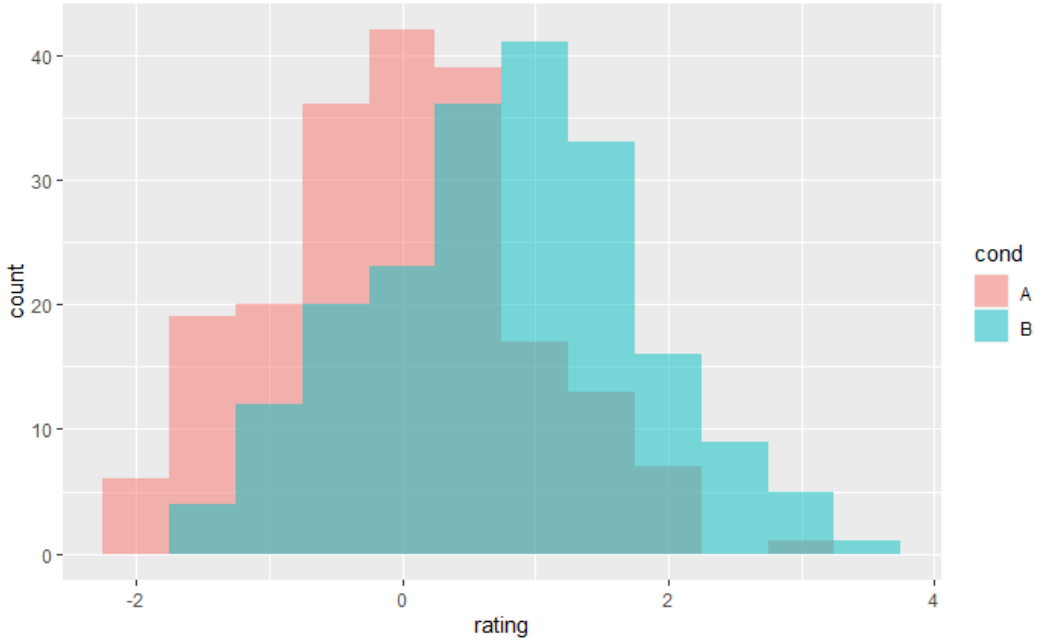
3. Perform the steps below with "dat" dataframe which is just a sample data for you to observe how each plot function (3b through 3e) works. Notice that you need to have `ggplot2` library installed on your system. Please refer slides how to install and import a library. Installation is done only once, but you need to import the library every time you need it by saying `library(ggplot2)`. Then Run the following commands and observe how the plots are generated. ***(40 points).***

(a) Data generation

```
dat <- data.frame(cond = factor(rep(c("A", "B"), each = 200)),
                  rating = c(rnorm(200), rnorm(200, mean = 0.8)))
```

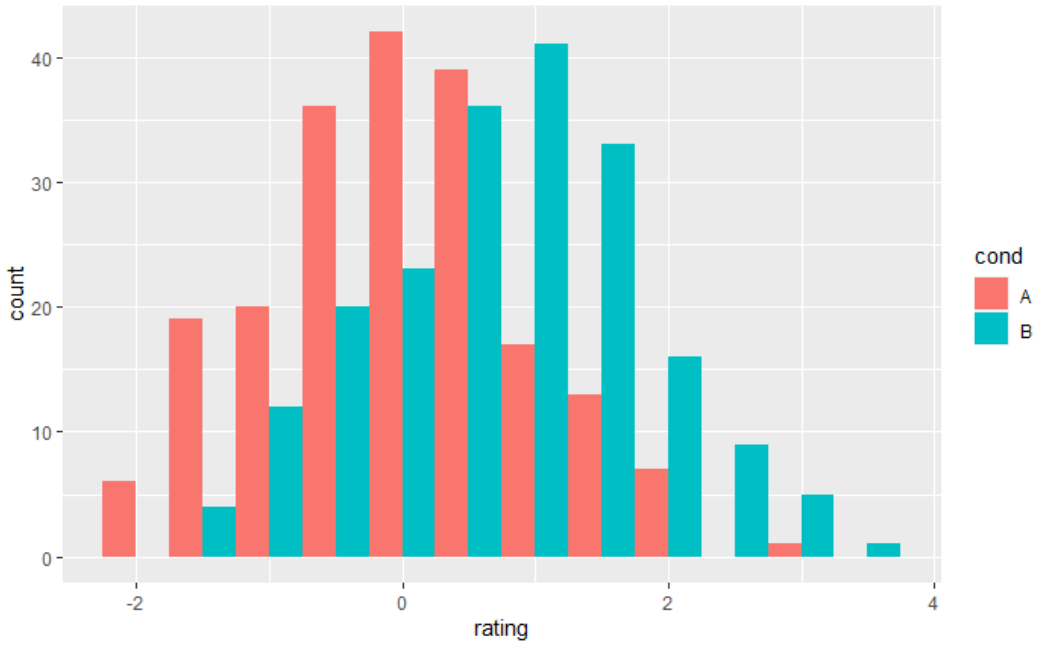
(b) Overlaid histograms

```
plot(ggplot(dat, aes(x = rating, fill = cond)) +
       geom_histogram(binwidth = 0.5, alpha = 0.5, position = "identity"))
```



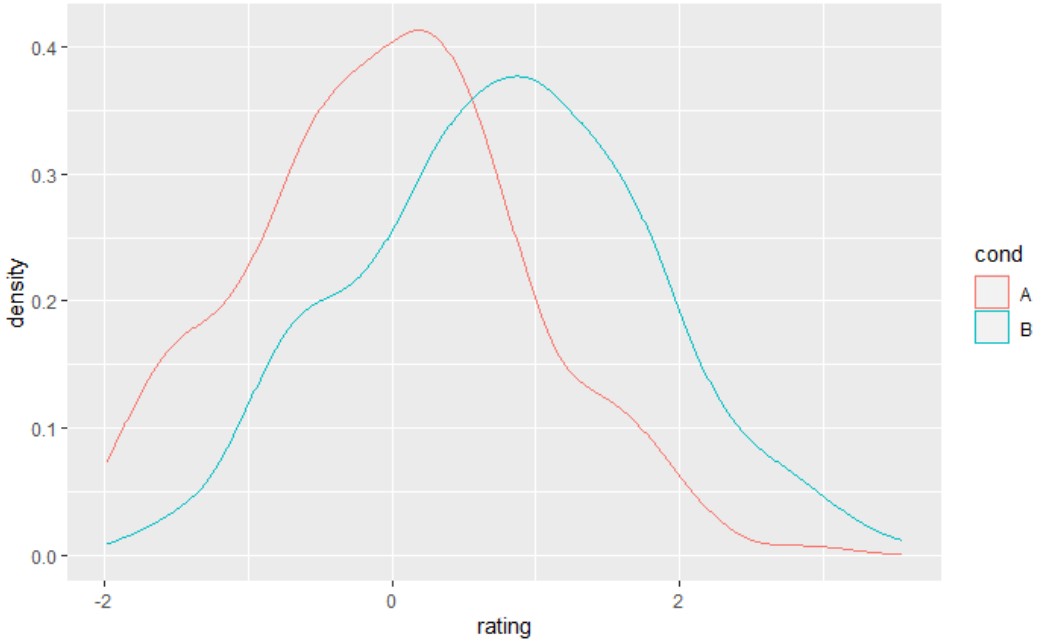
(c) Interleaved histograms

```
plot(ggplot(dat, aes(x = rating, fill = cond)) +
       geom_histogram(binwidth = 0.5, position = "dodge"))
```



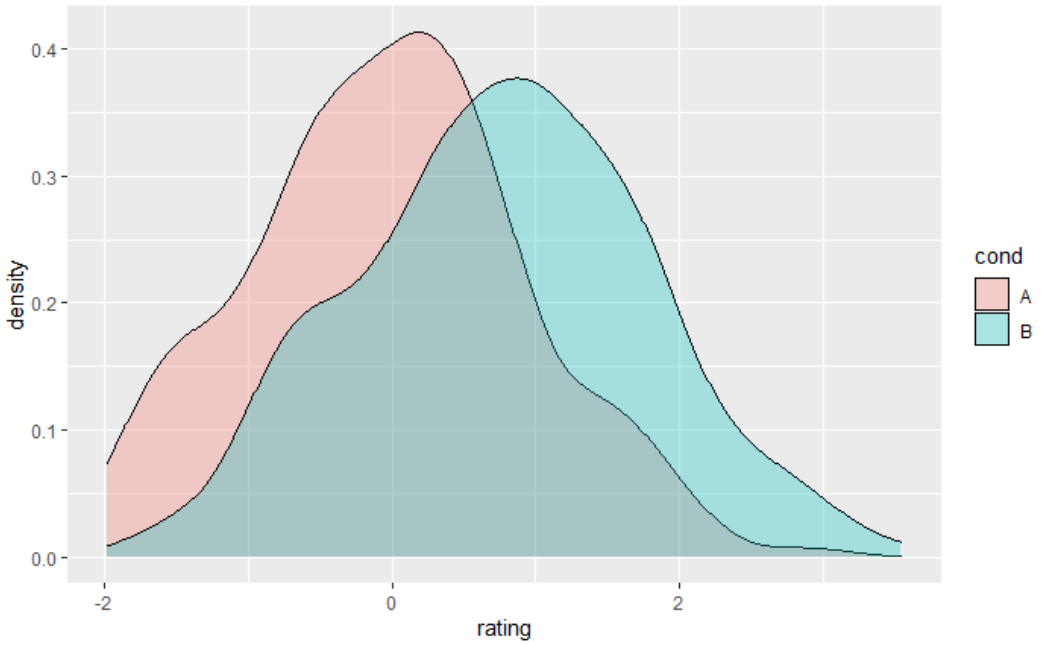
(d) Density plots

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



(e) Density plots w/ semi-transparent fill

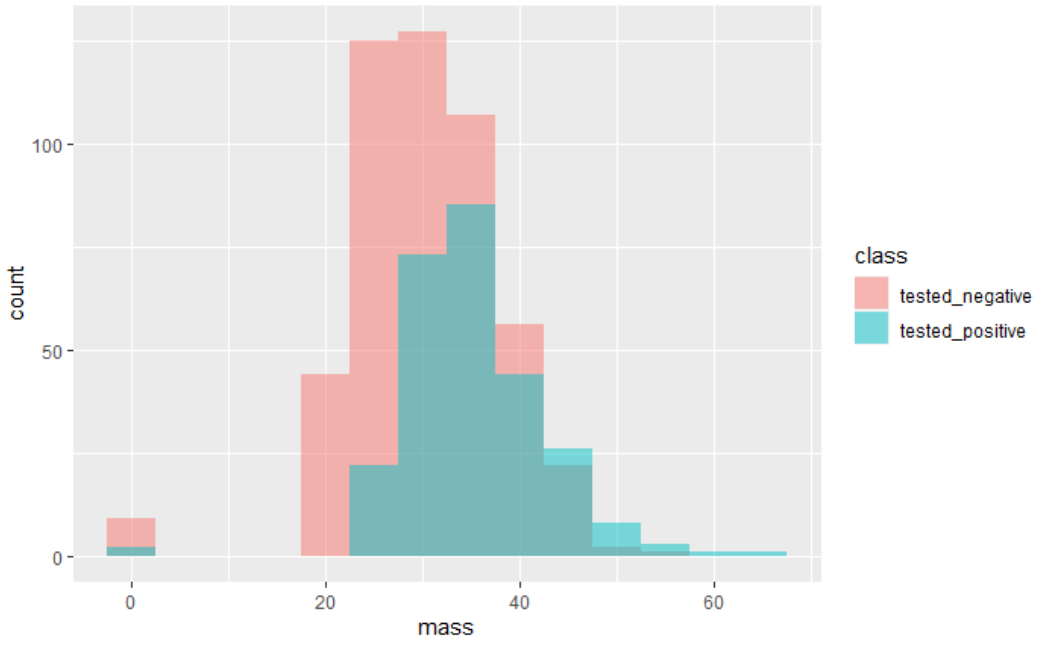
```
plot(ggplot(dat, aes(x = rating, fill = cond)) +
       geom_density(alpha = 0.3))
```



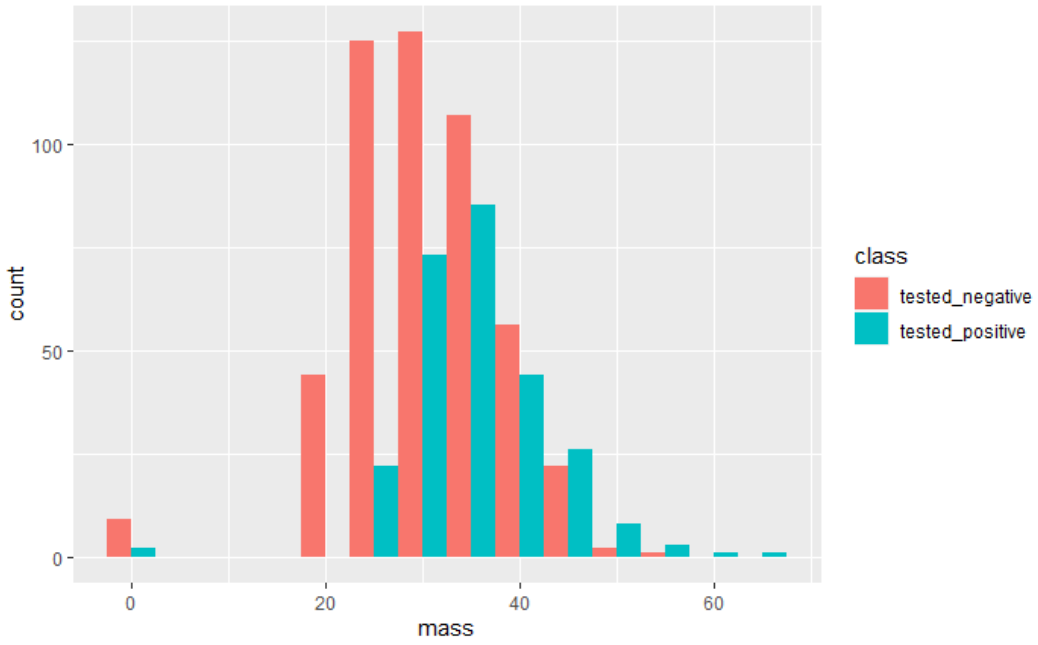
(f) Using `diabetes_train.csv`

```
diabetes <- read.csv("Data/diabetes_train.csv")
```

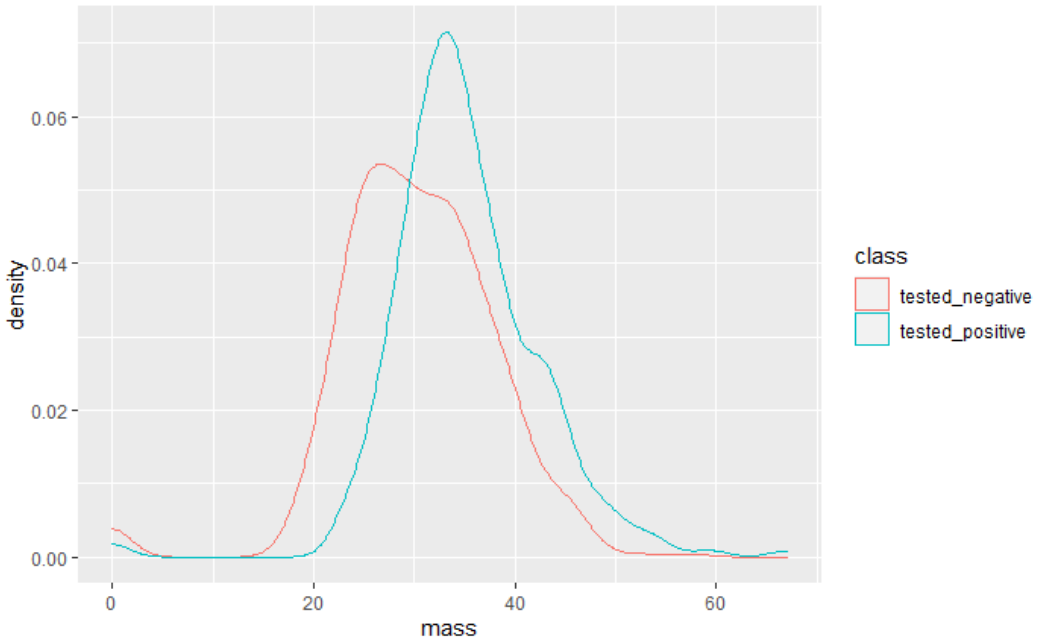
```
plot(ggplot(diabetes, aes(x = mass, fill = class)) +
       geom_histogram(binwidth = 5, alpha = 0.5, position = "identity"))
```



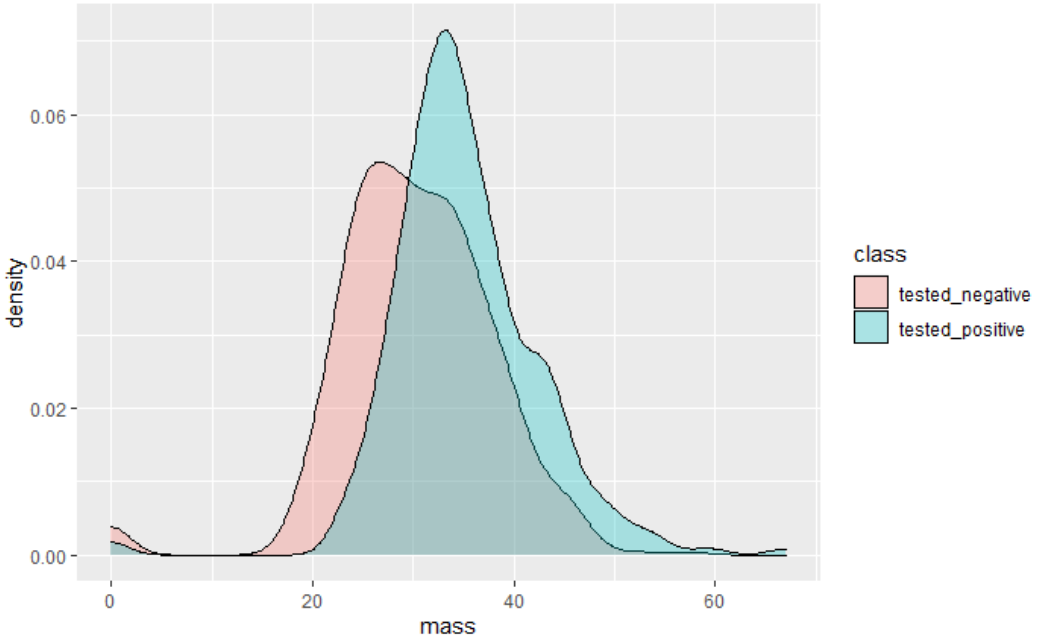

```
plot(ggplot(diabetes, aes(x = mass, fill = class)) +
       geom_histogram(binwidth = 5, position = "dodge"))
```




```
plot(ggplot(diabetes, aes(x = mass, color = class)) +
       geom_density())
```




```
plot(ggplot(diabetes, aes(x = mass, fill = class)) +
       geom_density(alpha = 0.3))
```


density
0.06
0.04
0.02
0.00
0
20
40
60
mass
class
tested_negative
tested_positive

4. By using `quantile()`, calculate 10th, 30th, 50th, 60th percentiles of skin attribute of diabetes data. *(10 points)*.

```
print(quantile(diabetes$skin, probs = c(0.10, 0.30, 0.50, 0.60)))
```

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