compgen2021: Week 1 exercises

Yuna Son

8/7/2021

Exercises for Week1

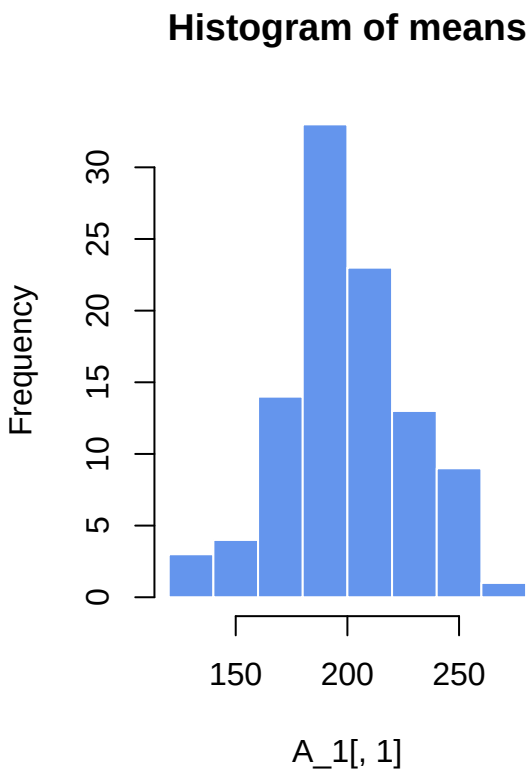
Statistics for genomics

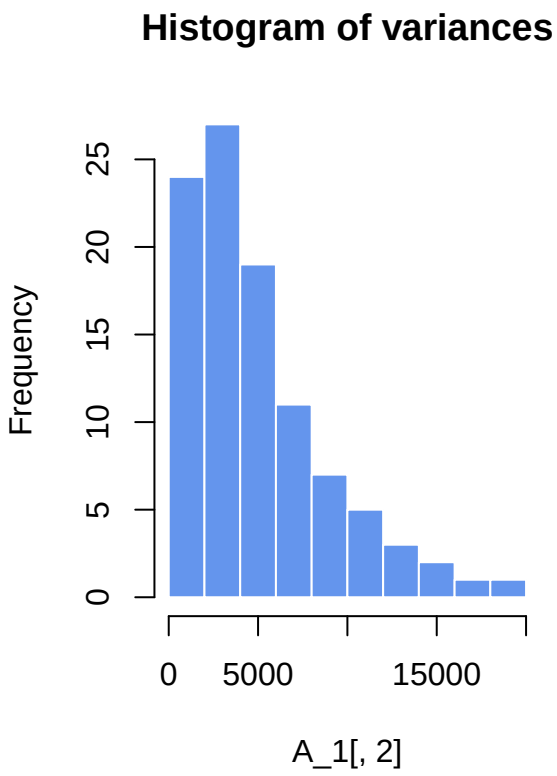
How to summarize collection of data points: The idea behind statistical distributions

1. Calculate the means and variances of the rows of the following simulated data set, and plot the distributions of means and variances using `hist()` and `boxplot()` functions. [Difficulty: **Beginner/Intermediate**]

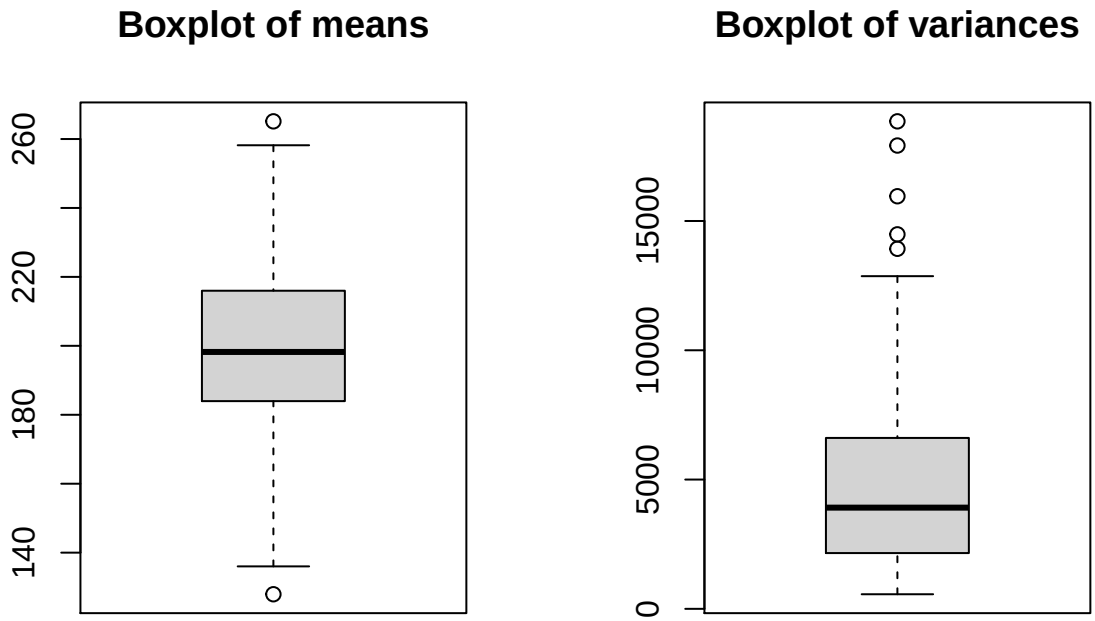
```
set.seed(100)
#sample data matrix from normal distribution
gset=rnorm(600,mean=200,sd=70)
data=matrix(gset,ncol=6)
A_1 <- matrix(NA,100,2)
for (i in 1:100){
A_1[i,] <- c(mean(data[i,]), sd(data[i,])^2)
}
```

```
# Plotting histograms and boxplots
par(mfrow = c(1,2))
hist(A_1[,1], col = "cornflowerblue", border="white",main = "Histogram of means")
hist(A_1[,2], col = "cornflowerblue", border="white",main = "Histogram of variances")
```






```
par(mfrow = c(1,2))
boxplot(A_1[,1], main = "Boxplot of means")
boxplot(A_1[,2],main = "Boxplot of variances")
```



2. Using the data generated above, calculate the standard deviation of the distribution of the means using the `sd()` function. Compare that to the expected standard error obtained from the central limit theorem keeping in mind the population parameters were $\sigma = 70$ and $n = 6$. How does the estimate from the random samples change if we simulate more data with `data=matrix(rnorm(6000,mean=200,sd=70),ncol=6)`? [Difficulty: **Beginner/Intermediate**]

```
# Calculate the standard deviation of the distribution of the means
sd(A_1[,1])
```

```
## [1] 28.07929
```

```
# Compare that to the expected standard error obtained from the central limit theorem.
70/sqrt(6)
```

```
## [1] 28.57738
```

```
# Do simulation with more samples
data2=matrix(rnorm(6000,mean=200,sd=70),ncol=6)
A_2 <- matrix(NA,100,2)
for (i in 1:100){
  A_2[i,] <- c(mean(data2[i,]), sd(data2[i,])^2)
}
sd(A_2[,1])
```

```
## [1] 25.55486
```

When we do more simulation with more data, the standard deviation of the distribution of the means reduces.

3. Simulate 30 random variables using the `rpois()` function. Do this 1000 times and calculate the mean of each sample. Plot the sampling distributions of the means using a histogram. Get the 2.5th and 97.5th percentiles of the distribution. [Difficulty: **Beginner/Intermediate**]

```
# Generate 30 random variables from a Poission distribution with mean 1 (1000 times)
library(mosaic)
```

```
## Warning: package 'mosaic' was built under R version 4.0.5
```

```
## Registered S3 method overwritten by 'mosaic':
##   method                           from
##   fortify.SpatialPolygonsDataFrame ggplot2
```

```
##
## The 'mosaic' package masks several functions from core packages in order to add
## additional features.  The original behavior of these functions should not be affected by this.
```

```
##
## Attaching package: 'mosaic'
```

```
## The following objects are masked from 'package:dplyr':
##
##     count, do, tally
```

```
## The following object is masked from 'package:Matrix':
##
##     mean
```

```
## The following object is masked from 'package:ggplot2':
##
##     stat
```

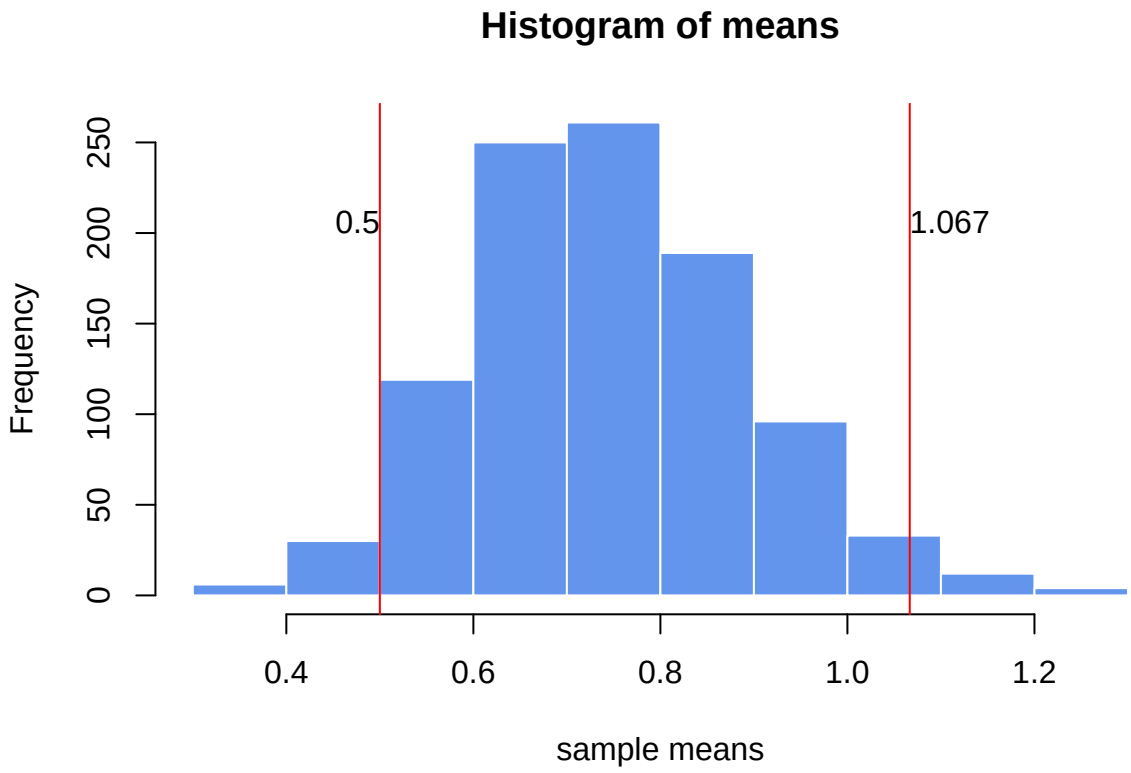
```
## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cor.test, cov, fivenum, IQR, median, prop.test,
##     quantile, sd, t.test, var
```

```
## The following objects are masked from 'package:base':
##
##     max, mean, min, prod, range, sample, sum
```

```
set.seed(1000)
sample = rpois(30, 1)
my_pois = do(1000) * mean(resample(sample))


# get percentiles
q = quantile(my_pois[,1], p = c(0.025, 0.975))
q
```

```
##     2.5%    97.5%
## 0.500000 1.066667
```

```
# Plotting histograms
hist(my_pois[,1], col = "cornflowerblue", border="white", xlab = "sample means", main = "Histogram of me
abline(v=c(q[1], q[2] ),col="red")
text(x=q[1], y = 200, round(q[1], 3), adj = c(1, 0))
text(x=q[2],y=200,round(q[2],3),adj=c(0,0))
```



4. Use the `t.test()` function to calculate confidence intervals of the mean on the first random sample `pois1` simulated from the `rpois()` function below. [Difficulty: **Intermediate**]

```
#HINT
set.seed(100)
#sample 30 values from poisson dist with lamda paramater =30
pois1=rpois(30,lambda=5)
t.test(pois1)
```

```
##
##  One Sample t-test
##
## data:  pois1
## t = 17.663, df = 29, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  4.362104 5.504563
```

```
## sample estimates:
## mean of x
##  4.933333
```

95 percent confidence interval is (4.362104, 5.504563).

5. Use the bootstrap confidence interval for the mean on `pois1`. [Difficulty: **Intermediate/Advanced**]

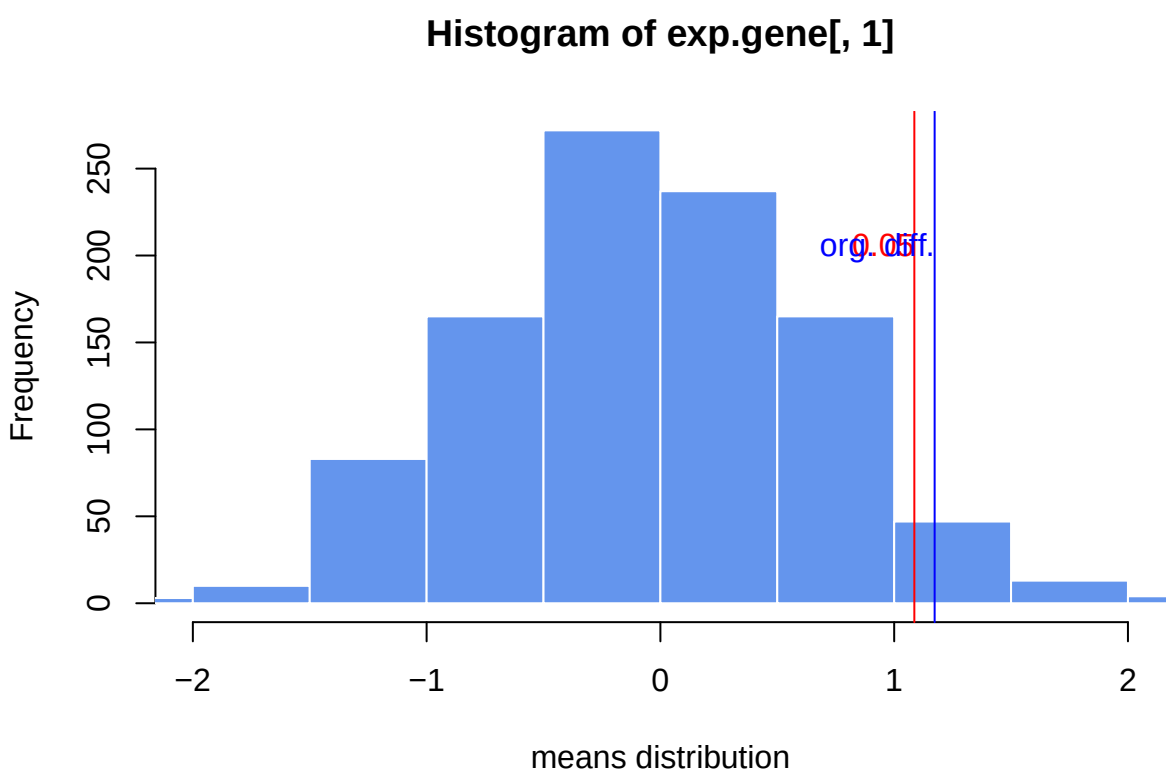
```
pois1=rpois(30,lambda=5)
# Simulate 100 times
my_pois = do(100) * mean(resample(pois1))

# get percentiles
q = quantile(my_pois[,1], p = c(0.025, 0.975))
q
```

```
##     2.5%    97.5%
## 4.482500 5.784167
```

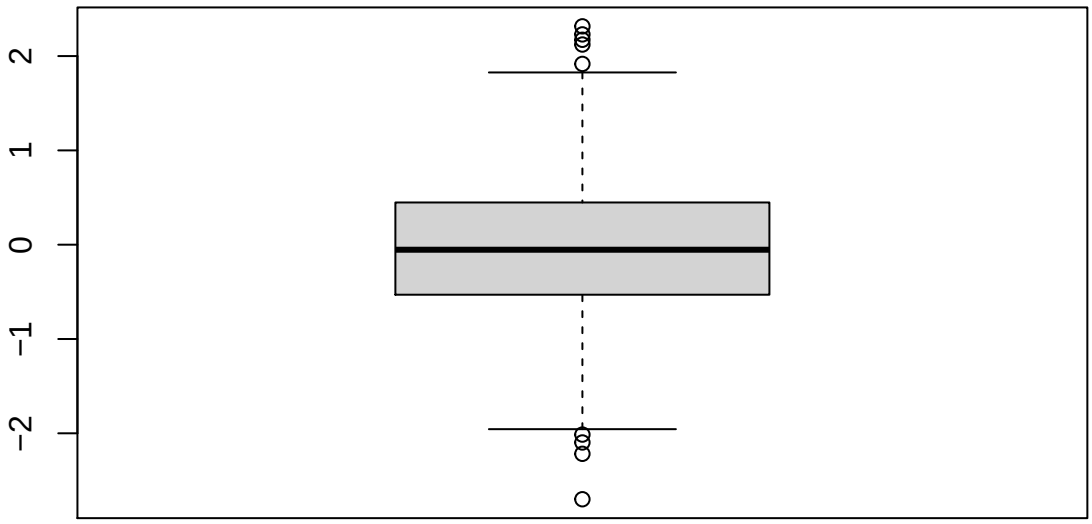
How to test for differences in samples

1. Test the difference of means of the following simulated genes using the randomization, `t-test()`, and `wilcox.test()` functions. Plot the distributions using histograms and boxplots. [Difficulty: **Intermediate/Advanced**]

```
set.seed(101)
gene1=rnorm(30,mean=4,sd=3)
gene2=rnorm(30,mean=3,sd=3)

org.diff = mean(gene1) - mean(gene2)
gene.df = data.frame(exp=c(gene1, gene2),
                     group = c(rep("gene1", 30), rep("control", 30)))
exp.gene <- do(1000) * diff(mosaic::mean(exp ~ shuffle(group), data = gene.df))

hist(exp.gene[,1],xlab="means distribution",
     xlim=c(-2,2),col="cornflowerblue",border="white")


abline(v=quantile(exp.gene[,1],0.95),col="red")
abline(v=org.diff,col="blue" )
text(x=quantile(exp.gene[,1],0.95),y=200,"0.05",adj=c(1,0),col="red")
text(x=org.diff,y=200,"org. diff.",adj=c(1,0),col="blue")
```




```
boxplot(exp.gene[,1],xlab="means distribution")
```



means distribution

```
t.test(gene1, gene2)
```

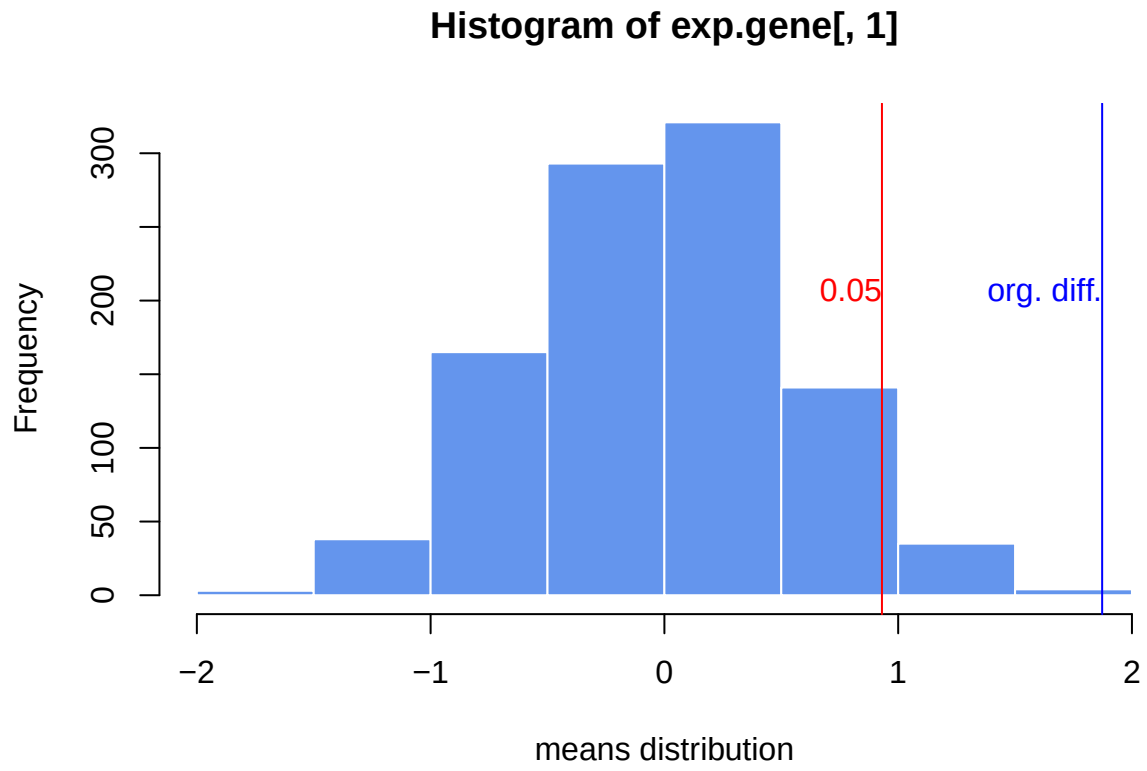
```
##
##  Welch Two Sample t-test
##
## data:  gene1 and gene2
## t = 1.6188, df = 55.719, p-value = 0.1111
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.2787179  2.6248604
## sample estimates:
## mean of x mean of y
##  3.751368  2.578297
```

```
wilcox.test(gene1, gene2)
```

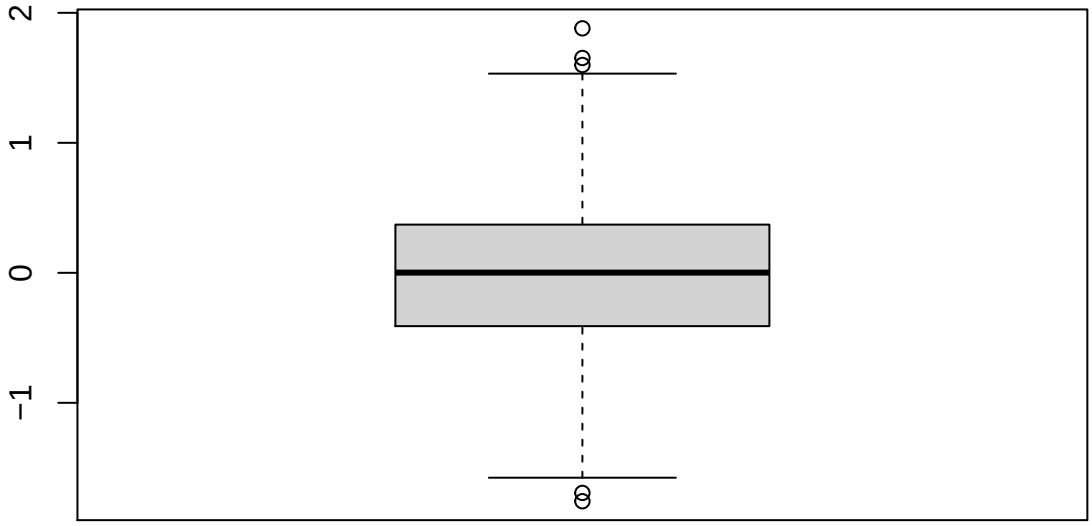
```
##
##  Wilcoxon rank sum exact test
##
## data:  gene1 and gene2
## W = 530, p-value = 0.2418
## alternative hypothesis: true location shift is not equal to 0
```

2. Test the difference of the means of the following simulated genes using the randomization, `t-test()` and `wilcox.test()` functions. Plot the distributions using histograms and boxplots. [Difficulty: **Intermediate/Advanced**]

```r
set.seed(100)
gene1=rnorm(30,mean=4,sd=2)
gene2=rnorm(30,mean=2,sd=2)
org.diff = mean(gene1) - mean(gene2)
gene.df = data.frame(exp=c(gene1, gene2),
                     group = c(rep("gene1", 30), rep("control", 30)))
exp.gene <- do(1000) * diff(mosaic::mean(exp ~ shuffle(group), data = gene.df))

hist(exp.gene[,1],xlab="means distribution",
     xlim=c(-2,2),col="cornflowerblue",border="white")


abline(v=quantile(exp.gene[,1],0.95),col="red")
abline(v=org.diff,col="blue" )
text(x=quantile(exp.gene[,1],0.95),y=200,"0.05",adj=c(1,0),col="red")
text(x=org.diff,y=200,"org. diff.",adj=c(1,0),col="blue")
```




```r
boxplot(exp.gene[,1],xlab="means distribution")
```



means distribution

```
t.test(gene1, gene2)
```

```
##
##  Welch Two Sample t-test
##
## data:  gene1 and gene2
## t = 3.7653, df = 47.552, p-value = 0.0004575
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.872397 2.872761
## sample estimates:
## mean of x mean of y
##  4.057728  2.185149
```

```
wilcox.test(gene1, gene2)
```

```
##
##  Wilcoxon rank sum exact test
##
## data:  gene1 and gene2
## W = 658, p-value = 0.001795
## alternative hypothesis: true location shift is not equal to 0
```

3. We need an extra data set for this exercise. Read the gene expression data set as follows: `gexpFile=system.file("extdata","geneExpMat.rds",package="compGenomRData")`

`data=readRDS(gexpFile)`. The data has 100 differentially expressed genes. The first 3 columns are the test samples, and the last 3 are the control samples. Do a t-test for each gene (each row is a gene), and record the p-values. Then, do a moderated t-test, as shown in section "Moderated t-tests" in this chapter, and record the p-values. Make a p-value histogram and compare two approaches in terms of the number of significant tests with the 0.05 threshold. On the p-values use FDR (BH), Bonferroni and q-value adjustment methods. Calculate how many adjusted p-values are below 0.05 for each approach. [Difficulty: **Intermediate/Advanced**]

```
gexpFile=system.file("extdata","geneExpMat.rds",package="compGenomRData")
data=readRDS(gexpFile)

# t.tests
group1=1:3
group2=4:6
n1=3
n2=3
dx=rowMeans(data[,group1])-rowMeans(data[,group2])

require(matrixStats)
```

```
## Loading required package: matrixStats
```

```
## Warning: package 'matrixStats' was built under R version 4.0.5
```

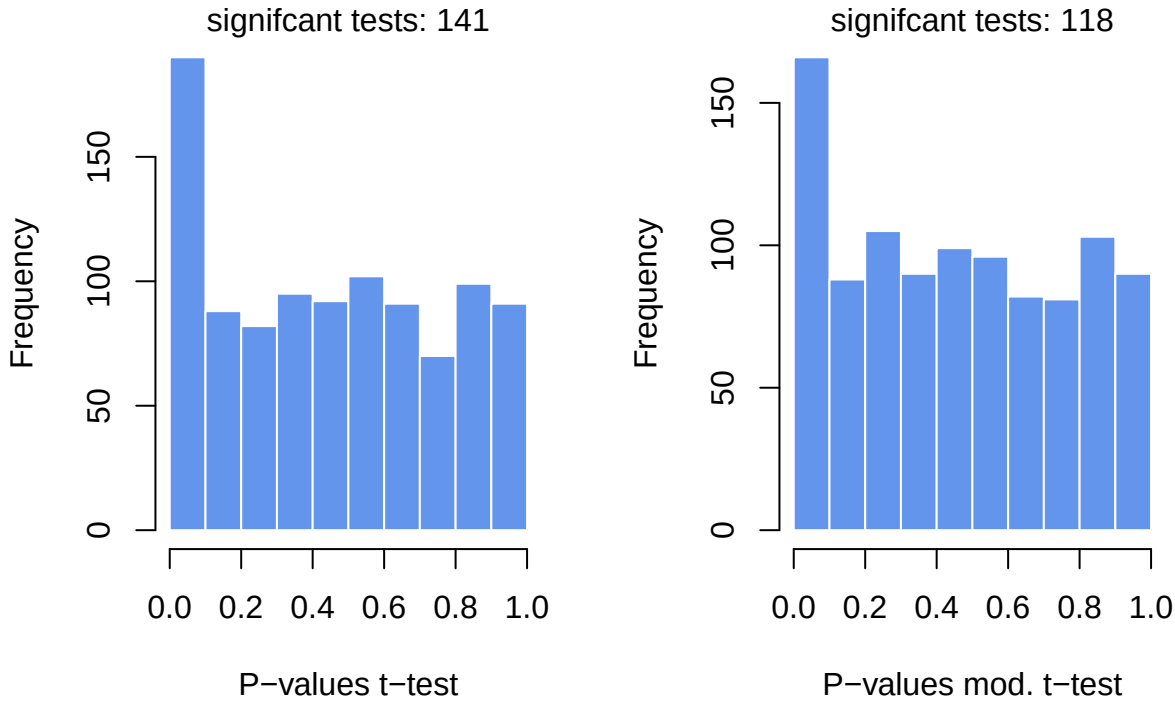
```
##
## Attaching package: 'matrixStats'
```

```
## The following objects are masked from 'package:mosaic':
##
##     count, iqr
```

```
## The following object is masked from 'package:dplyr':
##
##     count
```

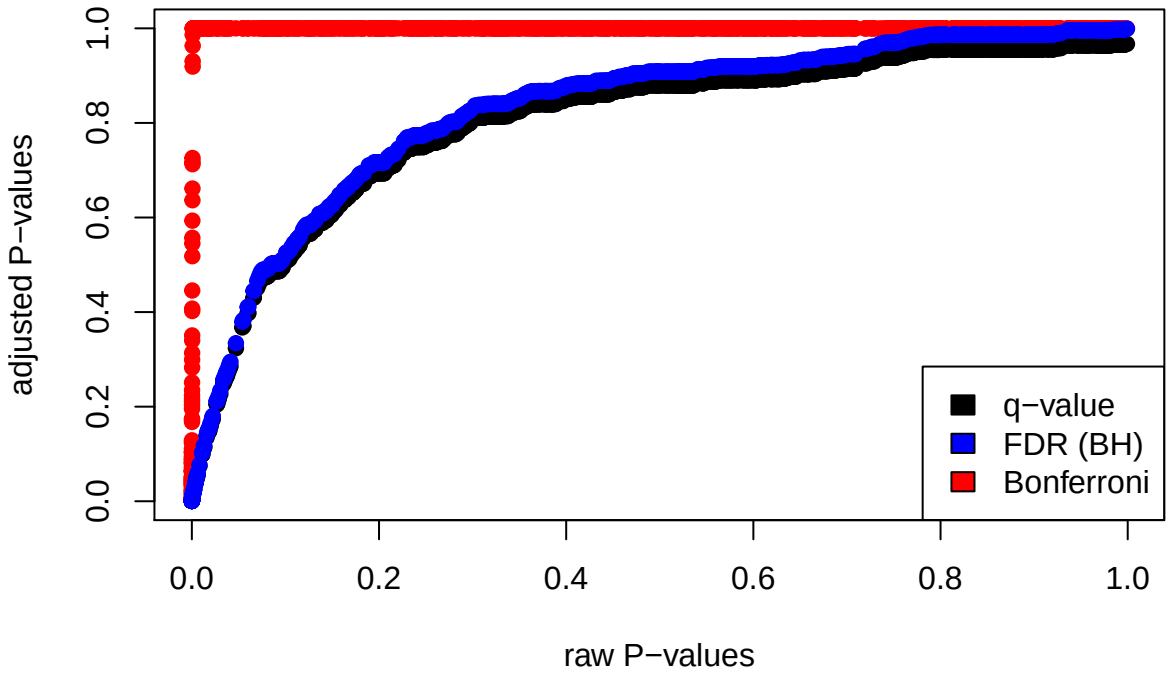
```
# get the esimate of pooled variance
stderr = sqrt( (rowVars(data[,group1])*(n1-1) +
       rowVars(data[,group2])*(n2-1)) / (n1+n2-2) * ( 1/n1 + 1/n2 ))

# do the shrinking towards median
mod.stderr = (stderr + median(stderr)) / 2 # moderation in variation

# esimate t statistic with moderated variance
t.mod <- dx / mod.stderr

# calculate P-value of rejecting null
p.mod = 2*pt( -abs(t.mod), n1+n2-2 )

# esimate t statistic without moderated variance
t = dx / stderr

# calculate P-value of rejecting null
```

```
p = 2*pt( -abs(t), n1+n2-2 )

par(mfrow=c(1,2))
hist(p,col="cornflowerblue",border="white",main="",xlab="P-values t-test")
mtext(paste("signifcant tests:",sum(p<0.05))  )
hist(p.mod,col="cornflowerblue",border="white",main="",
     xlab="P-values mod. t-test")
mtext(paste("signifcant tests:",sum(p.mod<0.05))  )
```



moderated t-test has more stringent results.

```
library(qvalue)

qvalues <- qvalue(p)$q
bonf.pval=p.adjust(p,method ="bonferroni")
fdr.adj.pval=p.adjust(p,method ="fdr")

plot(p,qvalues,pch=19,ylim=c(0,1),
     xlab="raw P-values",ylab="adjusted P-values")
points(p,bonf.pval,pch=19,col="red")
points(p,fdr.adj.pval,pch=19,col="blue")
legend("bottomright",legend=c("q-value","FDR (BH)","Bonferroni"),
       fill=c("black","blue","red"))
```



Bonferroni showed the most stringent p-values.

Relationship between variables: Linear models and correlation

Below we are going to simulate X and Y values that are needed for the rest of the exercise.

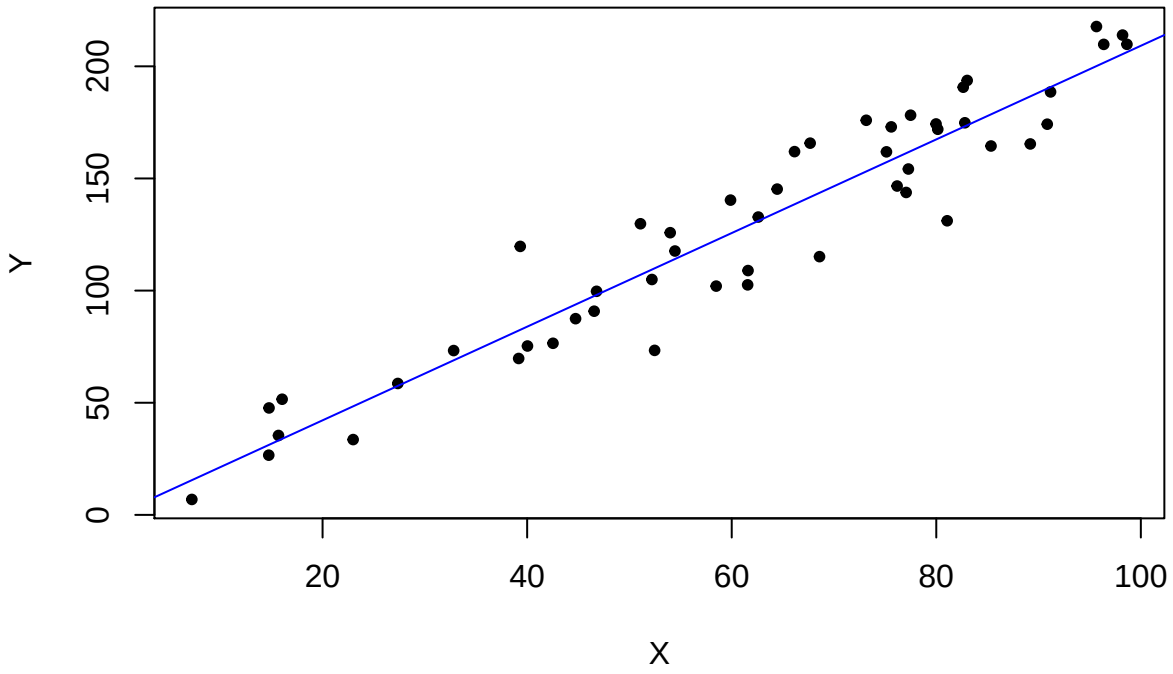
1. Run the code then fit a line to predict Y based on X. [Difficulty:**Intermediate**]

```
# set random number seed, so that the random numbers from the text
# is the same when you run the code.
set.seed(32)
# get 50 X values between 1 and 100
x = runif(50,1,100)
# set b0,b1 and variance (sigma)
b0 = 10
b1 = 2
sigma = 20
# simulate error terms from normal distribution
eps = rnorm(50,0,sigma)
# get y values from the linear equation and addition of error terms
y = b0 + b1*x+ eps

mod1=lm(y~x)

coef(mod1)
```

```
## (Intercept)           x
##   0.4095054   2.0874167
```

2. Plot the scatter plot and the fitted line. [Difficulty:**Intermediate**]

```
# plot the data points
plot(x,y,pch=20,
     ylab="Y",xlab="X")
# plot the linear fit
abline(mod1,col="blue")
```



3. Calculate correlation and R^2. [Difficulty:**Intermediate**]

```
cor(x, y, method = "pearson")
```

```
## [1] 0.9511939
```

```
summary(mod1)$r.squared
```

```
## [1] 0.9047699
```

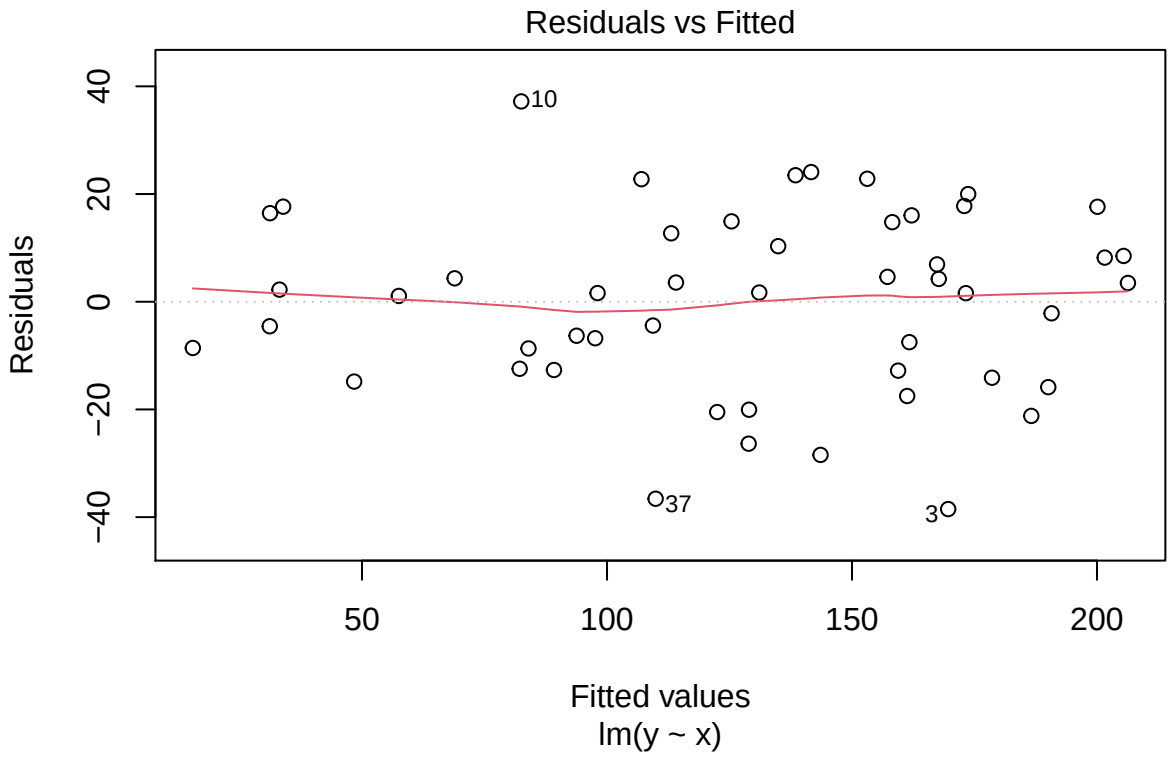
4. Run the `summary()` function and try to extract P-values for the model from the object returned by `summary`. See `?summary.lm`. [Difficulty:**Intermediate/Advanced**]

```
with(summary(mod1), pf(fstatistic[1],fstatistic[2],fstatistic[3],lower.tail=F))
```

```
##        value
## 3.719269e-26
```

5. Plot the residuals vs. the fitted values plot, by calling the `plot()` function with `which=1` as the second argument. First argument is the model returned by `lm()`. [Difficulty:**Advanced**]

```
# Plot residual vs. fitted
plot(mod1, which = 1)
```

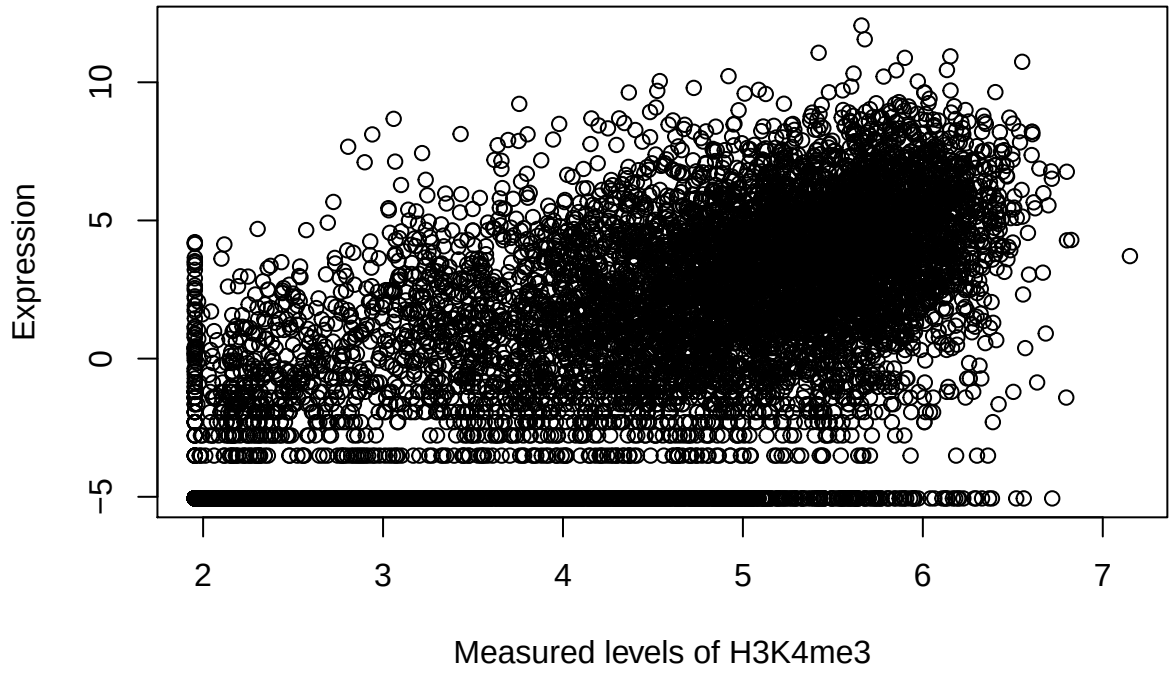


6. For the next exercises, read the data set histone modification data set. Use the following to get the path to the file:

```
hmodFile=system.file("extdata",
                     "HistoneModeVSgeneExp.rds",
                      package="compGenomRData")
```

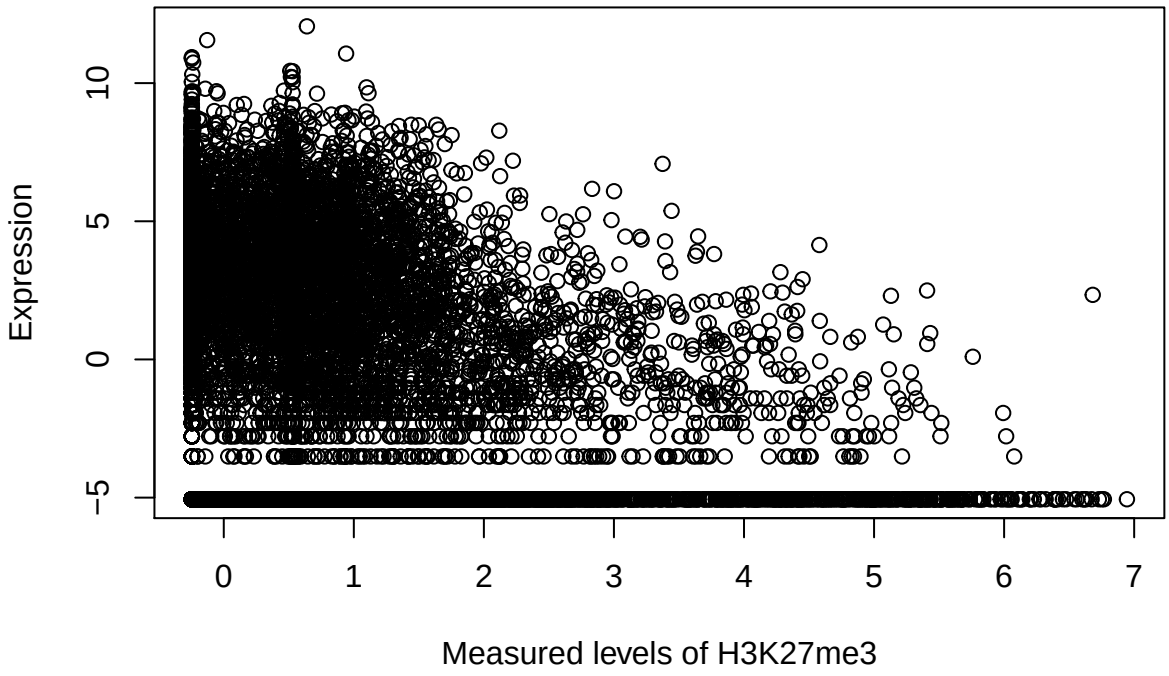
There are 3 columns in the dataset. These are measured levels of H3K4me3, H3K27me3 and gene expression per gene. Once you read in the data, plot the scatter plot for H3K4me3 vs. expression. [Difficulty:**Beginner**]

```
hmodFile=system.file("extdata",
                     "HistoneModeVSgeneExp.rds",
                      package="compGenomRData")
data=readRDS(hmodFile)

plot(data$H3k4me3, data$measured_log2, xlab="Measured levels of H3K4me3", ylab="Expression")
```



7. Plot the scatter plot for H3K27me3 vs. expression. [Difficulty:**Beginner**]

```
plot(data$H3k27me3, data$measured_log2, xlab="Measured levels of H3K27me3", ylab="Expression")
```



8. Fit the model for prediction of expression data using: 1) Only H3K4me3 as explanatory variable, 2) Only H3K27me3 as explanatory variable, and 3) Using both H3K4me3 and H3K27me3 as explanatory variables. Inspect the `summary()` function output in each case, which terms are significant. [Difficulty:**Beginner/Intermediate**]

```
fit1 <- lm(measured_log2 ~ H3k4me3, data = data)
summary(fit1)
```

```
##
## Call:
## lm(formula = measured_log2 ~ H3k4me3, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.0503  -1.0218  -0.1054   1.2508  11.2229
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -9.68153    0.05853  -165.4   <2e-16 ***
## H3k4me3      2.33263    0.01457   160.1   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.565 on 13729 degrees of freedom
## Multiple R-squared:  0.6511, Adjusted R-squared:  0.6511
```

```
## F-statistic: 2.563e+04 on 1 and 13729 DF,  p-value: < 2.2e-16
```

```
fit2 <- lm(measured_log2 ~ H3k27me3, data = data)
summary(fit2)
```

```
##
## Call:
## lm(formula = measured_log2 ~ H3k27me3, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.8885 -4.0716 -0.5573  3.8014 12.9216
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.36390    0.04394  -8.282   <2e-16 ***
## H3k27me3    -0.78807    0.03117 -25.283   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.245 on 13729 degrees of freedom
## Multiple R-squared:  0.04449,    Adjusted R-squared:  0.04442
## F-statistic: 639.2 on 1 and 13729 DF,  p-value: < 2.2e-16
```

H3K4me3 showed better correlation than H3K27me3 only fitted model by R-squared values (H3K4me3 is closer to 1) and F statistics (higher values in H3K4me3).

```
fit3 <- lm(measured_log2 ~ H3k4me3 + H3k27me3, data = data)
summary(fit3)
```

```
##
## Call:
## lm(formula = measured_log2 ~ H3k4me3 + H3k27me3, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.0564  -1.0702  -0.3066   1.2486  10.8950
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -9.11803    0.05980 -152.48   <2e-16 ***
## H3k4me3      2.29815    0.01417  162.17   <2e-16 ***
## H3k27me3    -0.54532    0.01832  -29.77   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.486 on 13728 degrees of freedom
## Multiple R-squared:  0.6723, Adjusted R-squared:  0.6723
## F-statistic: 1.408e+04 on 2 and 13728 DF,  p-value: < 2.2e-16
```

10. Is using H3K4me3 and H3K27me3 better than the model with only H3K4me3? [Difficulty:**Intermediate**]

Multiple regression model shows lower F-statistic than H3K4me3 only model. So, Multiple regression model may not be better to reject the null hypothesis.