STT 461 HW 4

Derien Weatherspoon

2023-03-05

```
library(ggplot2)
library(pwr)
library(boot)
library(DirichletReg)
```

```
## Loading required package: Formula
```

```
library(bayesboot)
library(infer)
library(MASS)
library(dplyr)
```

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

## The following object is masked from 'package:MASS':
##
##     select

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

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

```
cm <- read.csv("cars_multi.csv")
```

Question 1

Suppose a 100-item sample which is drawn from a binomial distribution with n = 50 has mean value 30.7. Construct a simulation to determine whether we can say at a 99% that the p parameter is greater than 0.6.

```
n <- 50
mean <- 30.7
set.seed(123)
samples <- replicate(10000, rbinom(100, n, 0.6))
sample_means <- apply(samples, 2, mean)
p_value <- mean(sample_means >= 30.7)
p_value
```

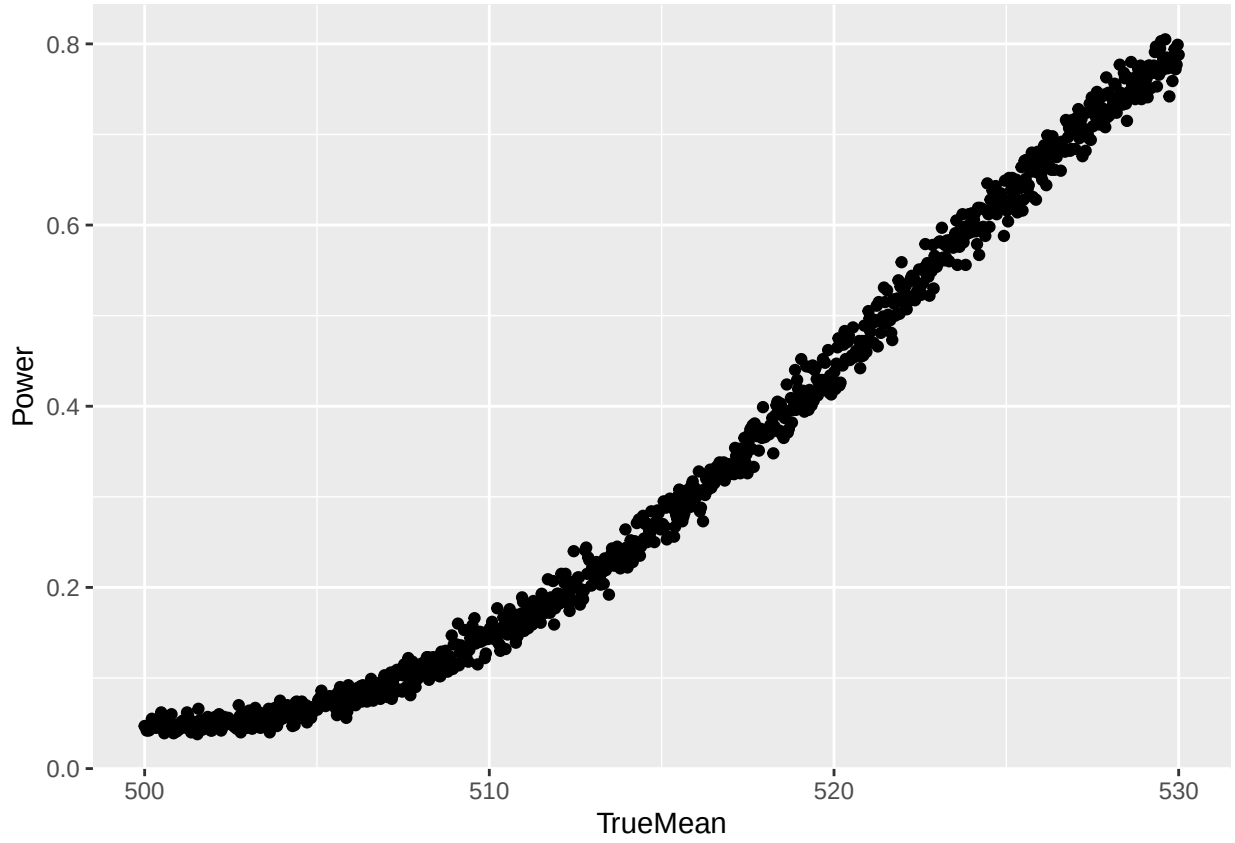
```
## [1] 0.0227
```

We can reject the null hypothesis.

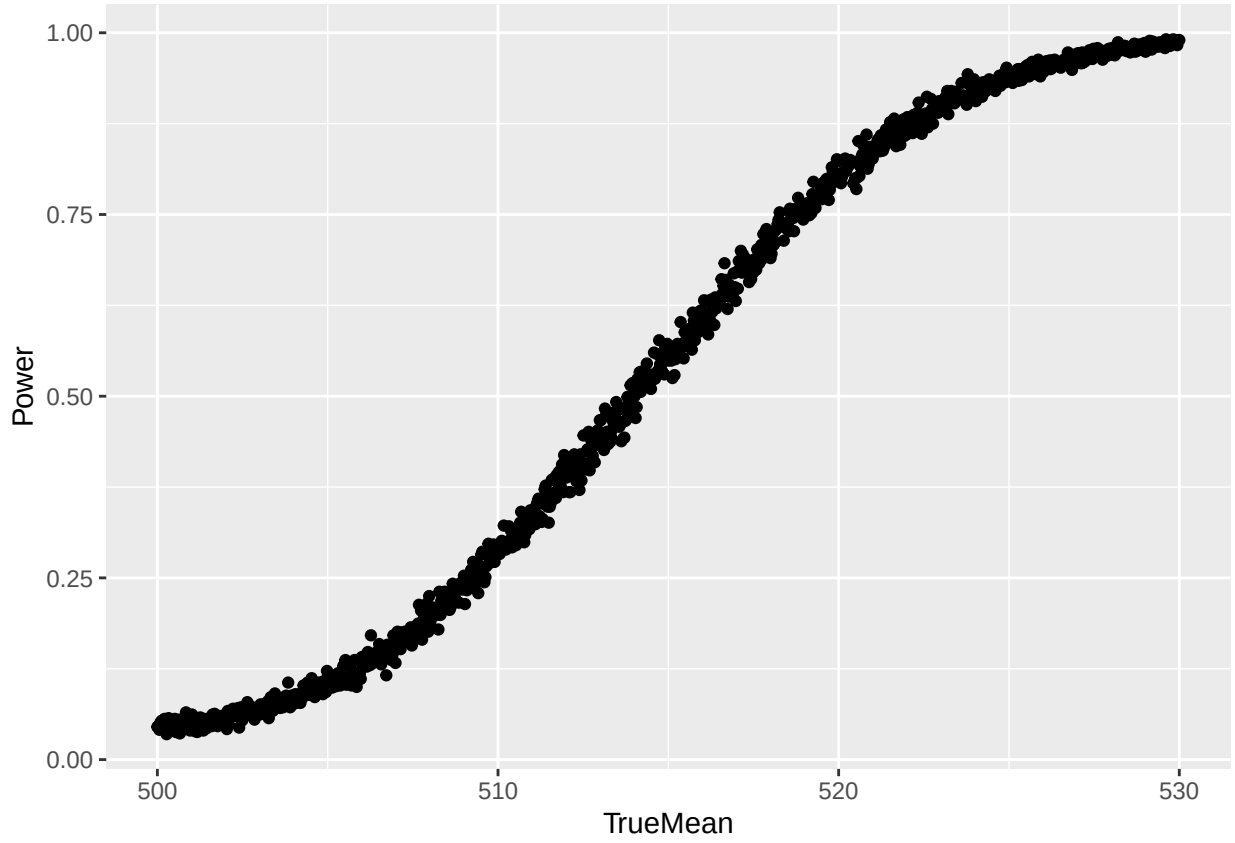
Question 2

Suppose a sample is to be drawn from a normally-distributed population to disprove a null hypothesis that the mean value is 500. For samples that have a standard deviation of 30, create powers curves for n = 10, 20, and 50, with the graphs extending for effect sizes from 0 to 1.

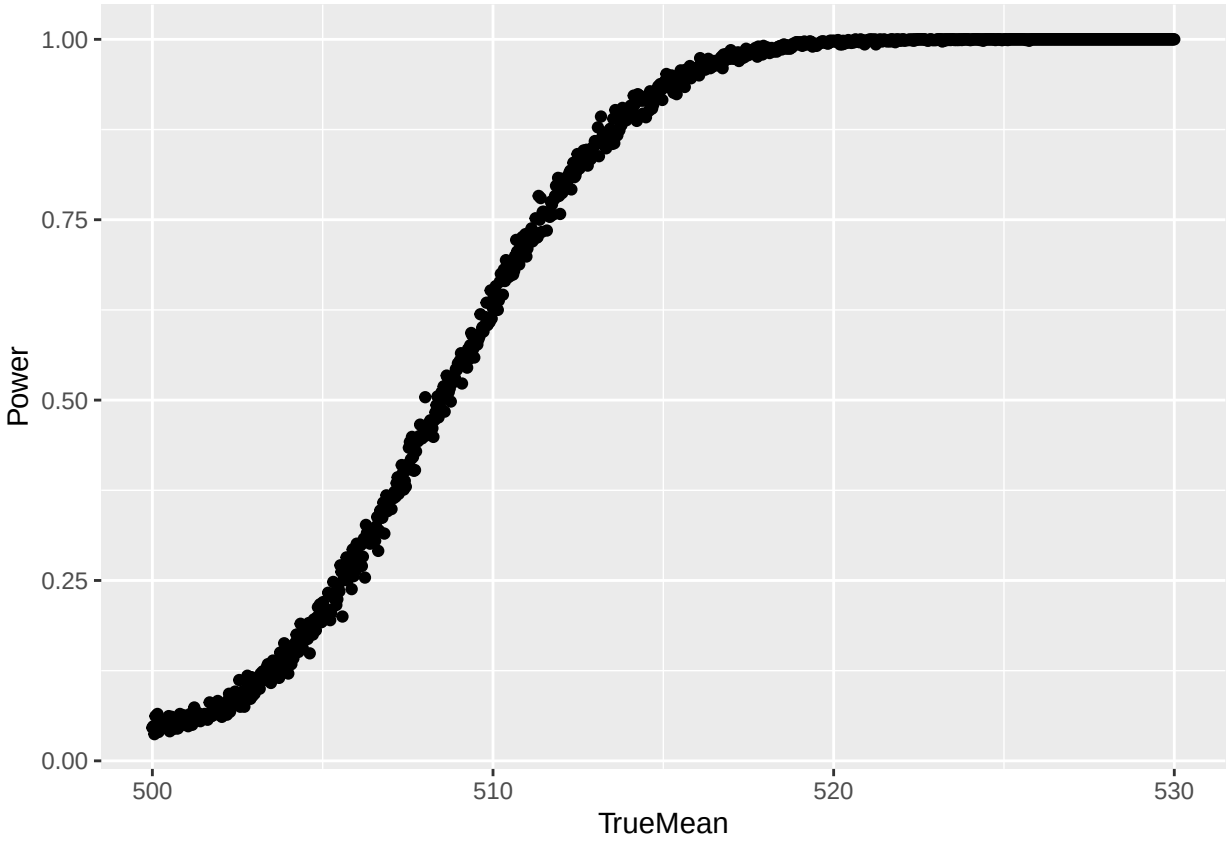
```
# n = 10
stddev <- 30
H0mean <- 500
truemean <- seq(H0mean, H0mean + stddev, by = 0.001*stddev)
n <- 10
alpha <- 1 - .95
stderror <- stddev/sqrt(n)
pwr <- rep(0,length(truemean))
for(i in 1:length(truemean)){
  meansim <- rep(0,1000)
  for(j in 1:1000){
    simsamp <- truemean[i] + stddev * rt(n, n - 1)
    meansim[j] <- mean(simsamp)
  }
  pwr[i] <- 1 - sum(pt(abs(meansim - H0mean)/stderror, n - 1) < (1-alpha)/2 + 0.5)/1000
}
powerdf <- data.frame('TrueMean' = truemean, 'Power' = pwr)
ggplot(data = powerdf, aes(x = TrueMean, y = Power)) +geom_point()
```




```
# n = 20
stddev <- 30
H0mean <- 500
truemean <- seq(H0mean, H0mean + stddev, by = 0.001*stddev)
n <- 20
alpha <- 1 - .95
stderror <- stddev/sqrt(n)
pwr <- rep(0,length(truemean))
for(i in 1:length(truemean)){
  meansim <- rep(0,1000)
  for(j in 1:1000){
    simsamp <- truemean[i] + stddev * rt(n, n - 1)
    meansim[j] <- mean(simsamp)
  }
  pwr[i] <- 1 - sum(pt(abs(meansim - H0mean)/stderror, n - 1) < (1-alpha)/2 + 0.5)/1000
}
powerdf <- data.frame('TrueMean' = truemean, 'Power' = pwr)
ggplot(data = powerdf, aes(x = TrueMean, y = Power)) +geom_point()
```




```
# n = 50
stddev <- 30
H0mean <- 500
truemean <- seq(H0mean, H0mean + stddev, by = 0.001*stddev)
n <- 50
alpha <- 1 - .95
stderror <- stddev/sqrt(n)
pwr <- rep(0,length(truemean))
for(i in 1:length(truemean)){
  meansim <- rep(0,1000)
  for(j in 1:1000){
    simsamp <- truemean[i] + stddev * rt(n, n - 1)
    meansim[j] <- mean(simsamp)
  }
  pwr[i] <- 1 - sum(pt(abs(meansim - H0mean)/stderror, n - 1) < (1-alpha)/2 + 0.5)/1000
}
powerdf <- data.frame('TrueMean' = truemean, 'Power' = pwr)
ggplot(data = powerdf, aes(x = TrueMean, y = Power)) +geom_point()
```



Question 4

With the weight field in the cars_multi dataset, use (i) standard bootstrapping, and (ii) weighted Bayesian bootstrapping with the Dirichlet distribution to create 95% confidence intervals for (a) the mean (b) the 95th quantile (c) the standard deviation Also compare the confidence interval for mean to what you would get with using the t distribution.

```
weightmean <- rep(0,10000)
weight95 <- rep(0,10000)
weightsd <- rep(0,10000)
for(i in 1:10000){
  weightsamp <- sample(cm$weight, length(cm$weight), replace = TRUE)
  weightmean[i] <- mean(weightsamp)
  weight95[i] <- quantile(weightsamp, 0.95)
  weightsd[i] <- sd(weightsamp)
}
quantile(weightmean, c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 2888.623 3055.575
```

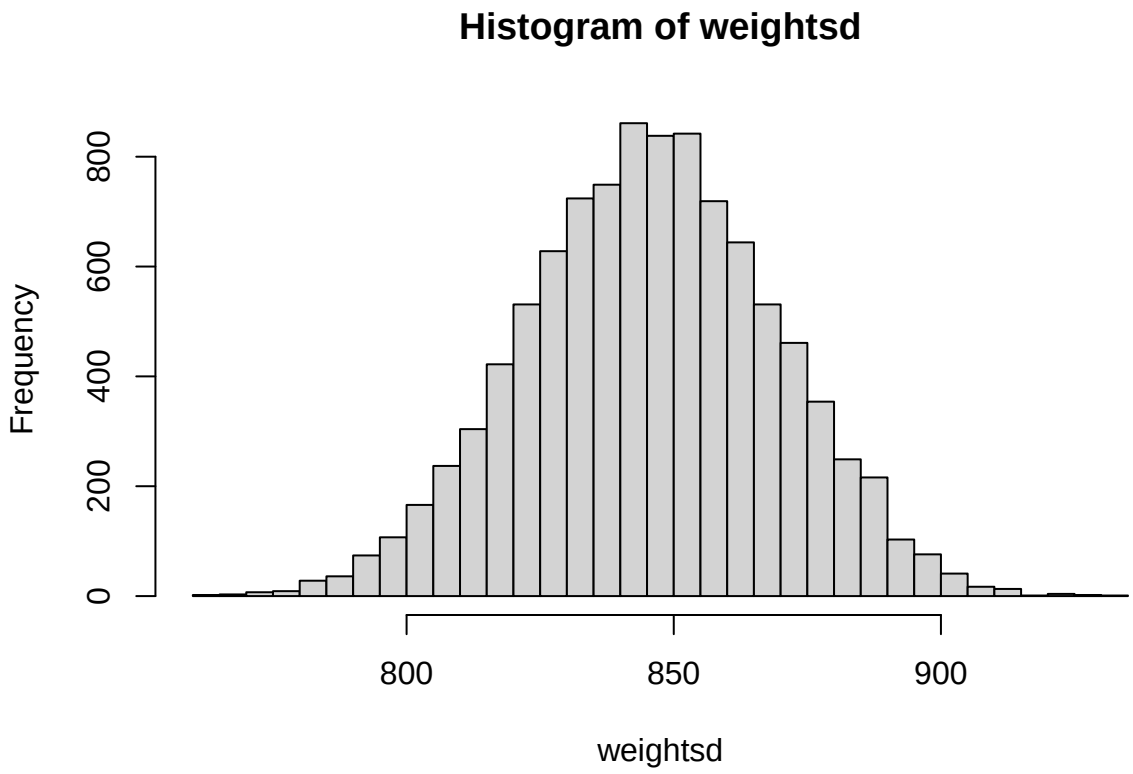
```
quantile(weight95, c(0.025, 0.975))
```

```
##    2.5%   97.5%
## 4364.95 4640.85
```

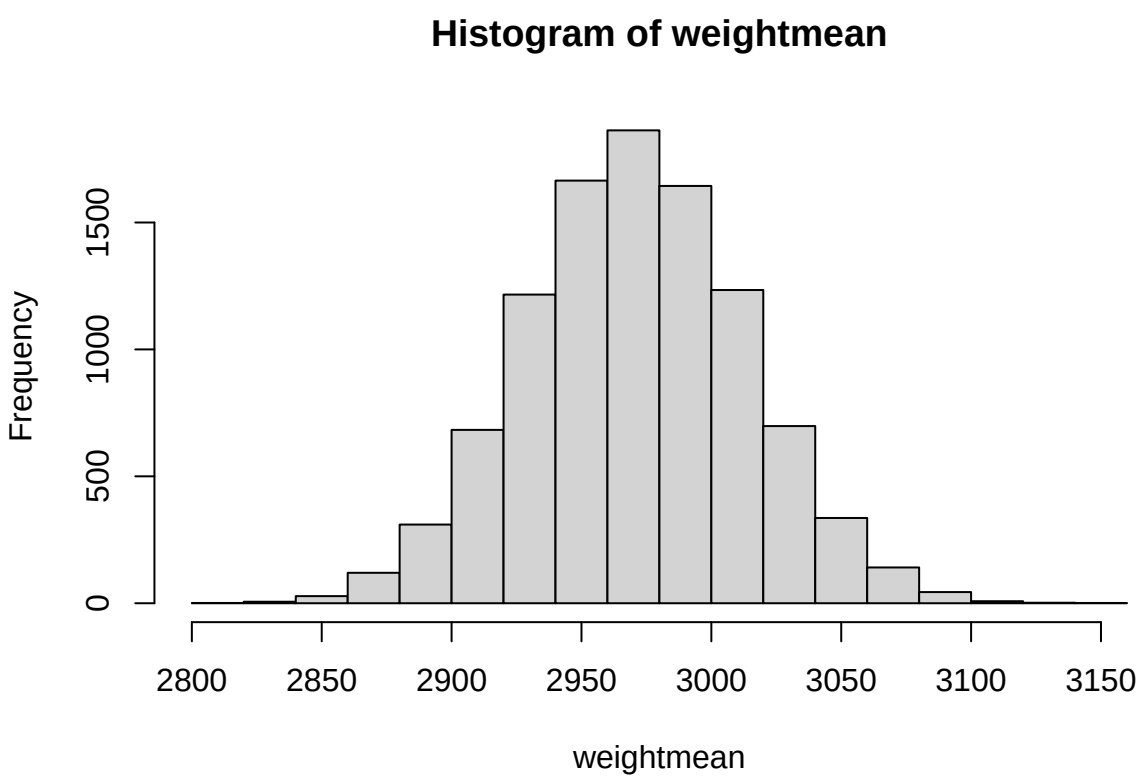
```r
quantile(weightsd, c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 799.2092 890.1716
```

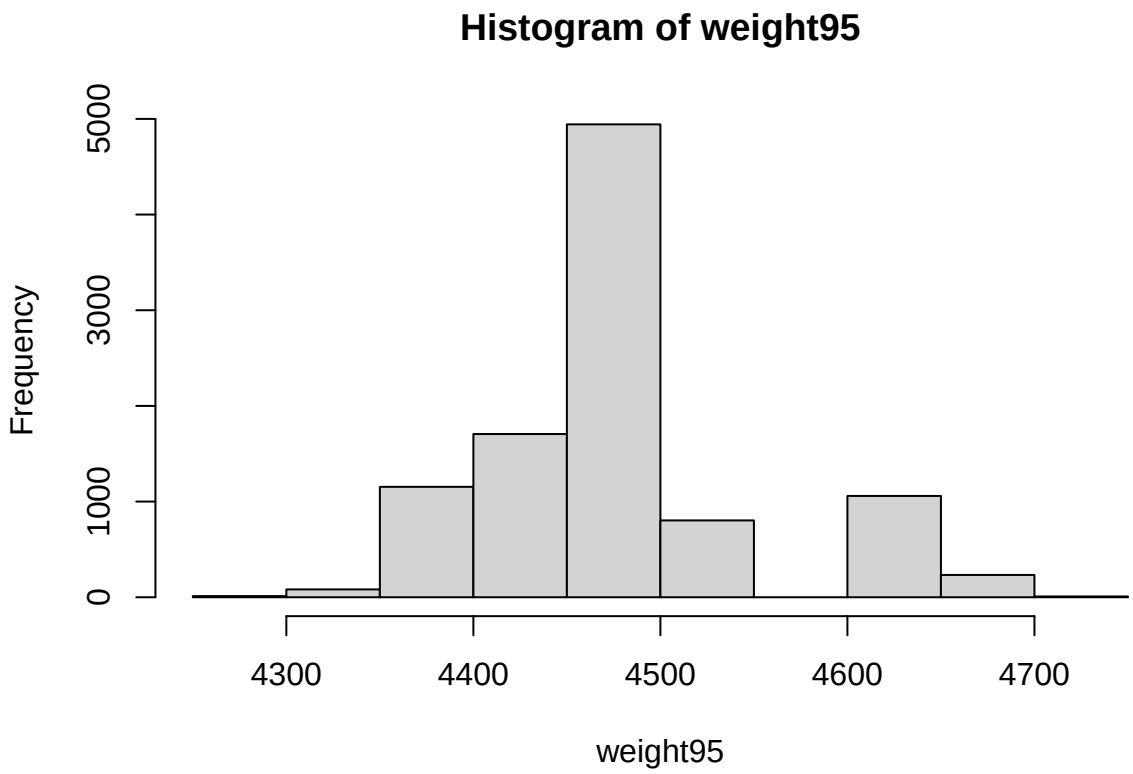
```r
hist(weightsd, breaks = 30)
```




```r
hist(weightmean)
```




```
hist(weight95)
```




```
weight <- rep(0,length(cm$weight))
weightmeanbayes <- rep(0,10000)
weight95bayes <- rep(0,10000)
weightsdbayes <- rep(0,10000)
for(i in 1:10000){
  weight <- rdirichlet(1, rep(1,length(cm$weight)))
  weightsampbayes <- sample(cm$weight, length(cm$weight), prob = weight, replace = TRUE)
  weightmeanbayes[i] <- mean(weightsampbayes)
  weight95bayes[i] <- quantile(weightsampbayes, 0.95)
  weightsdbayes[i] <- sd(weightsampbayes)
}
quantile(weightmeanbayes, c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 2854.733 3090.888
```

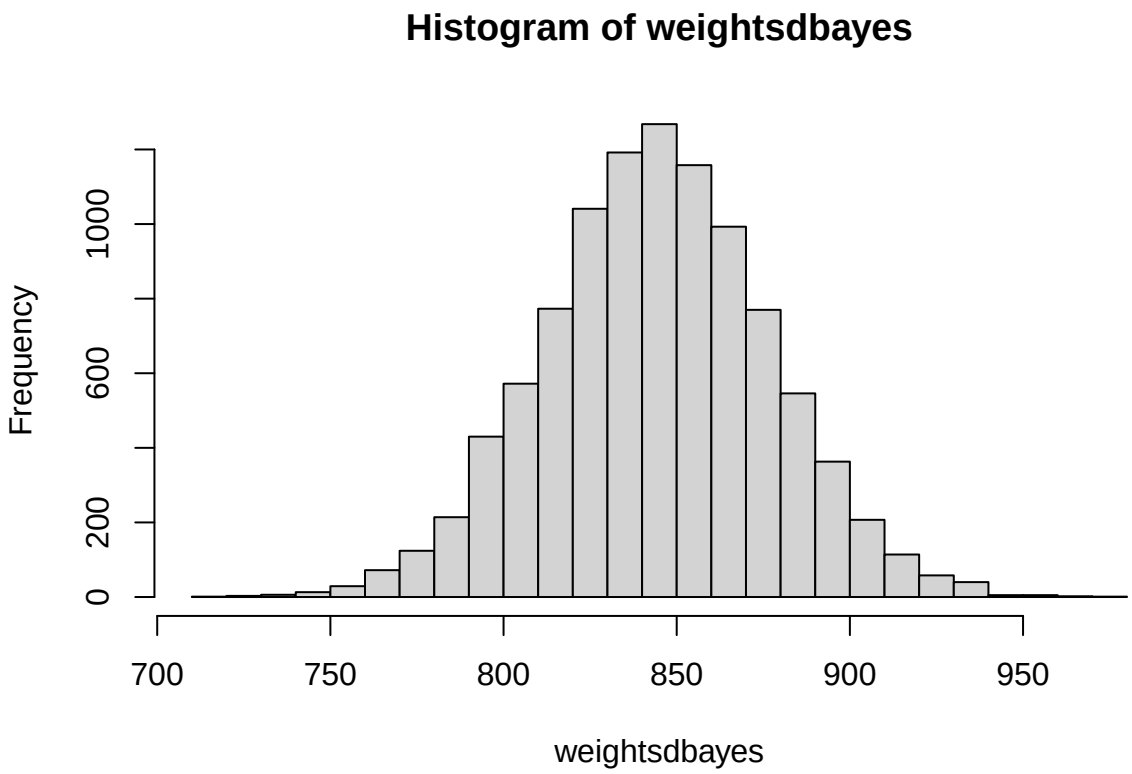
```
quantile(weight95bayes, c(0.025, 0.975))
```

```
##  2.5% 97.5%
##  4335  4668
```

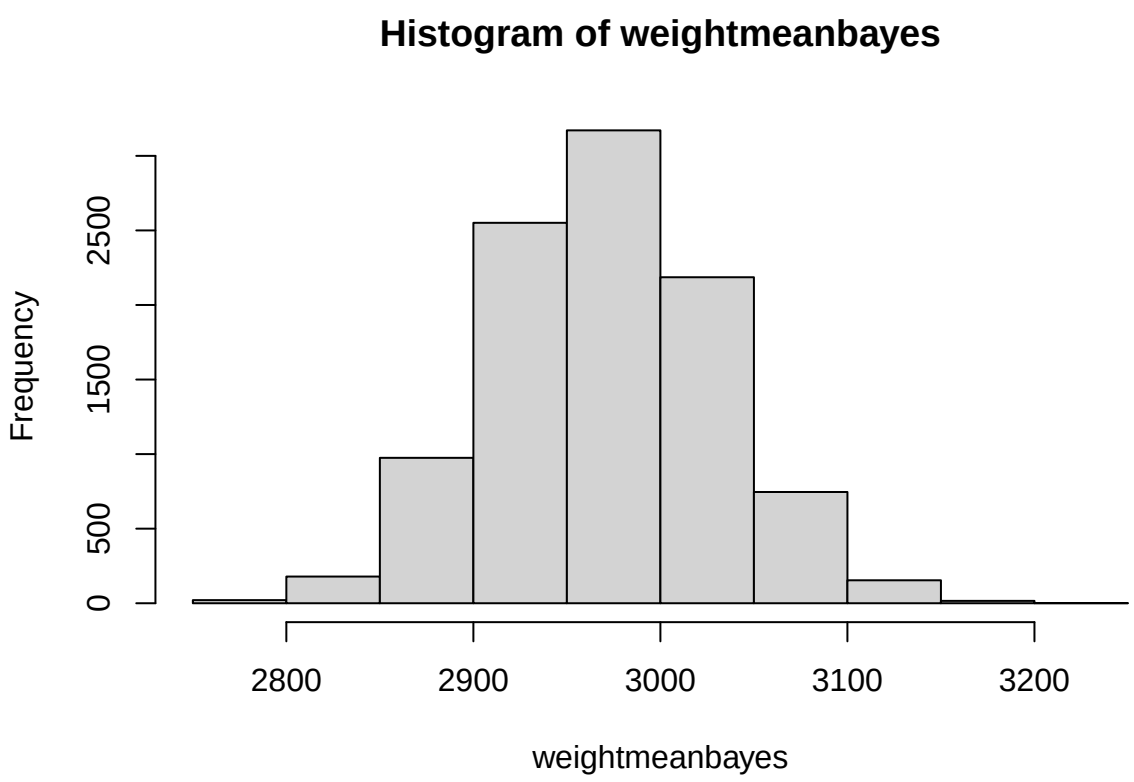
```
quantile(weightsdbayes, c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 780.2057 908.4386
```

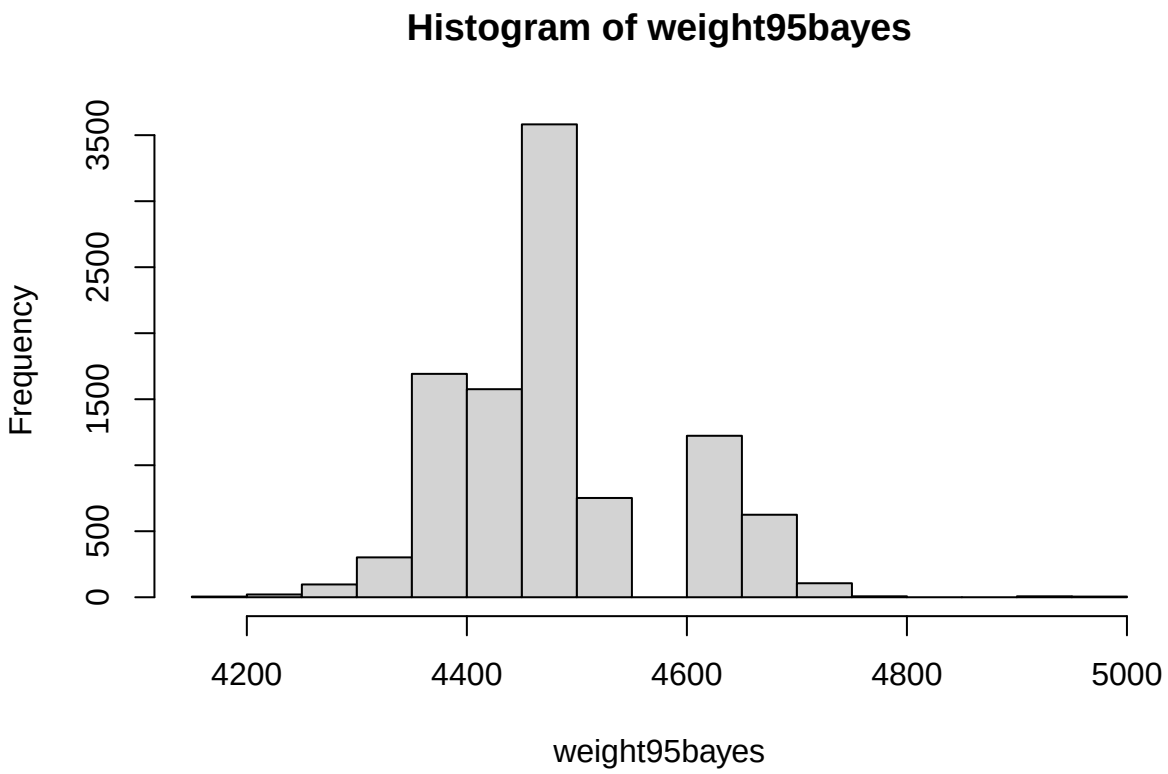
```
hist(weightsdbayes, breaks = 30)
```




```
hist(weightmeanbayes)
```




```
hist(weight95bayes)
```

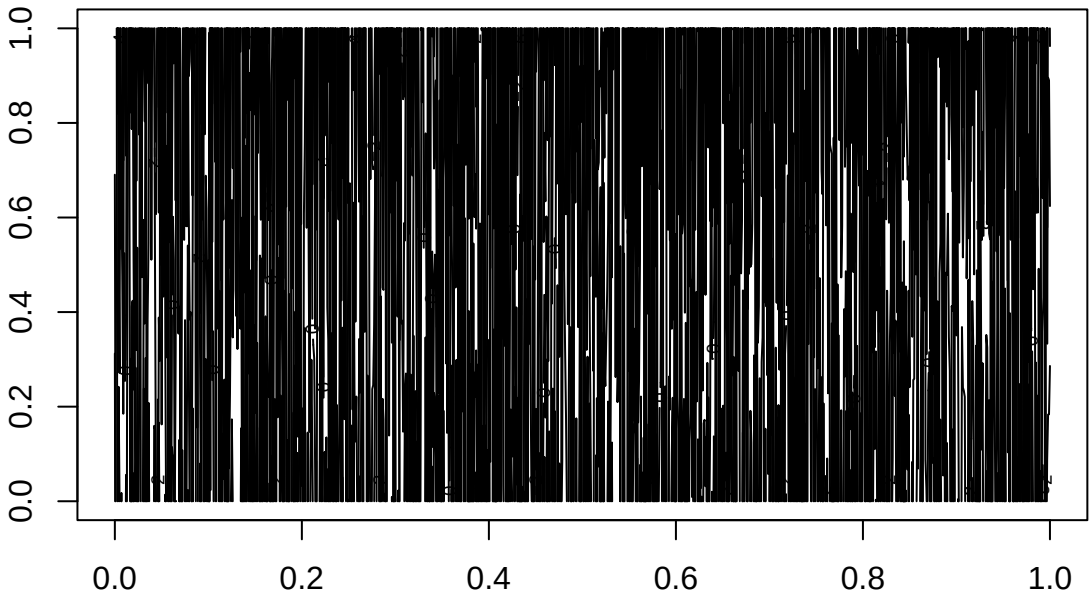


Question 5

Take a bivariate normal distribution with two random variables X and Y, with mean value = (1, -1), var(X) = 3, var(Y) = 6, and cor(X,Y) = -0.5. (a) create a contour plot for this data

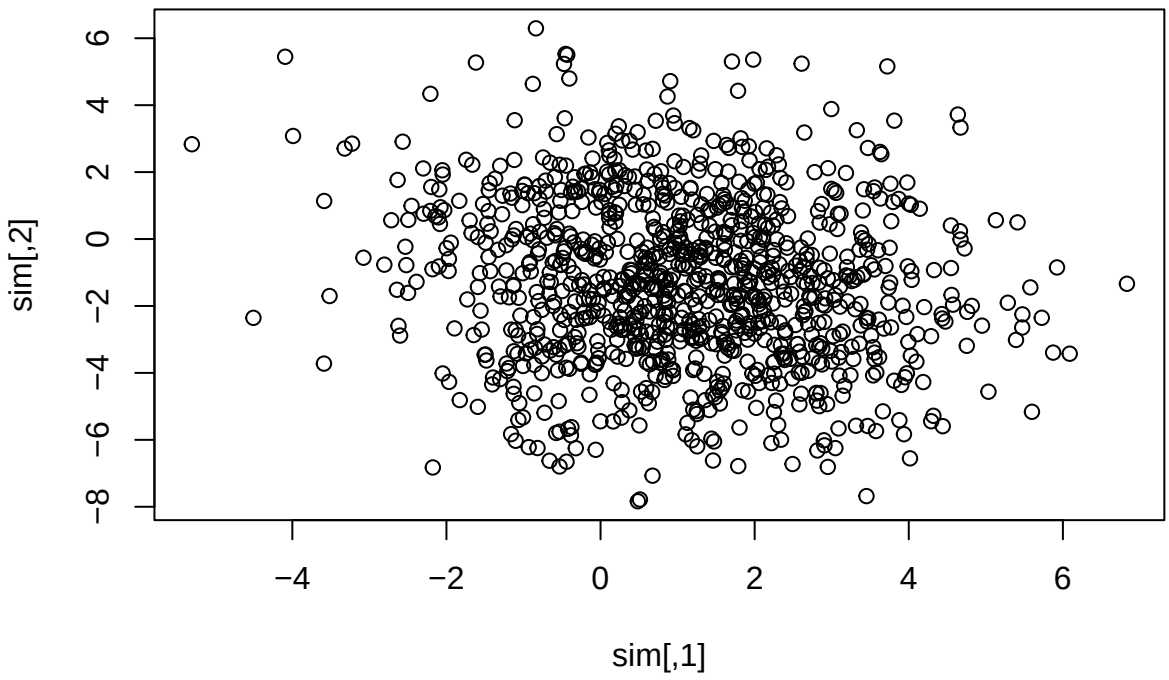
```
# set values first
mu <- c(1, -1)
varX <- 3
varY <- 6
corXY <- -0.5
# create cov matrix
cov_matrix <- matrix(c(varX, corXY, corXY, varY), nrow = 2)

#the distribution
biv_norm <- mvrnorm(1000, mu, matrix(c(varX, corXY, corXY, varY), 2))
contour(biv_norm) #definitely wrog
```



(b) plot 1,000 simulations of this distribution

```
# plot 1000 simulations
sim <- mvrnorm(1000, mu, cov_matrix)
plot(sim)
```



(c)Using 1,000,000 simulations, (i) Verify that var(X + Y) = var(X) + var(Y) + 2Cov(X, Y) (ii) find: (1) the expected value of Y (2) the expected value of Y, given that X > 2 (3) the expected value of Y, given that X = 2

```
# 1,000,000 simulations
sim1M <- mvrnorm(1000000, mu, matrix(c(varX, corXY, corXY, varY), 2))

# sum of X and Y
sumXY <- rowSums(sim1M)

# variance of X + Y
varSumXY <- var(sumXY)

# variance of X and Y
varXY <- varX + varY + 2 * corXY

# var(X + Y) = var(X) + var(Y) + 2Cov(X, Y) , verify if they do or don't
varSumXY == varXY
```

```
## [1] FALSE
```

```
# expected value of Y
muY <- mu[2]
muY
```

```
## [1] -1
```

```r
# expected value of Y, given that X > 2
simX2 <- sim1M[sim1M[, 1] > 2, ]
meanYX2 <- mean(simX2[, 2])
meanYX2
```

```
## [1] -1.346639
```

```r
# (3) Find the expected value of Y, given that X = 2
simX3 <- sim1M[sim1M[, 1] == 2, ]
meanYX3 <- mean(simX3[, 2])
meanYX3
```

```
## [1] NaN
```

Question 6

(a) Find the eigenvectors and eigenvalues for the matrix (on pdf)

```r
# define matrix
A <- matrix(c(1,7,3,7,4,5,3,5,0), nrow = 3, ncol = 3)
eigen(A)$values
```

```
## [1] 12.364127 -2.454628 -4.909498
```

```r
eigen(A)$vectors
```

```
##            [,1]       [,2]       [,3]
## [1,] -0.5534805 -0.5016948  0.6648020
## [2,] -0.7167664 -0.1195787 -0.6869839
## [3,] -0.4241524  0.8567399  0.2934135
```

(b) Express the vectors x = <1, 3, 1> and y = <-1, 4, 9> in terms of the eigenvectors basis for the above matrix.

```r
x <- c(1,3,1)
y <- c(-1,4,9)
Ax <- solve(eigen(A)$vectors, x) * x
Ay <- solve(eigen(A)$vectors, y) * y
Ax
```

```
## [1] -3.12793216 -0.01107283 -1.10273603
```

```r
Ay
```

```
## [1]  6.130957 30.936157 -6.948141
```

(c) Find the inner product of x and y in the original coordinates. Then find the inner product of x and y in terms of the eigenvector basis. Do you get the same value?

```r
xy <- sum(x * y)
xy
```

```
## [1] 20
```

```r
eigen_xy <- sum(t(x) %*% eigen(A)$vectors %*% y)
eigen_xy
```

```
## [1] -6.811456
```