Homework Assignment 1

Seokjun Choi

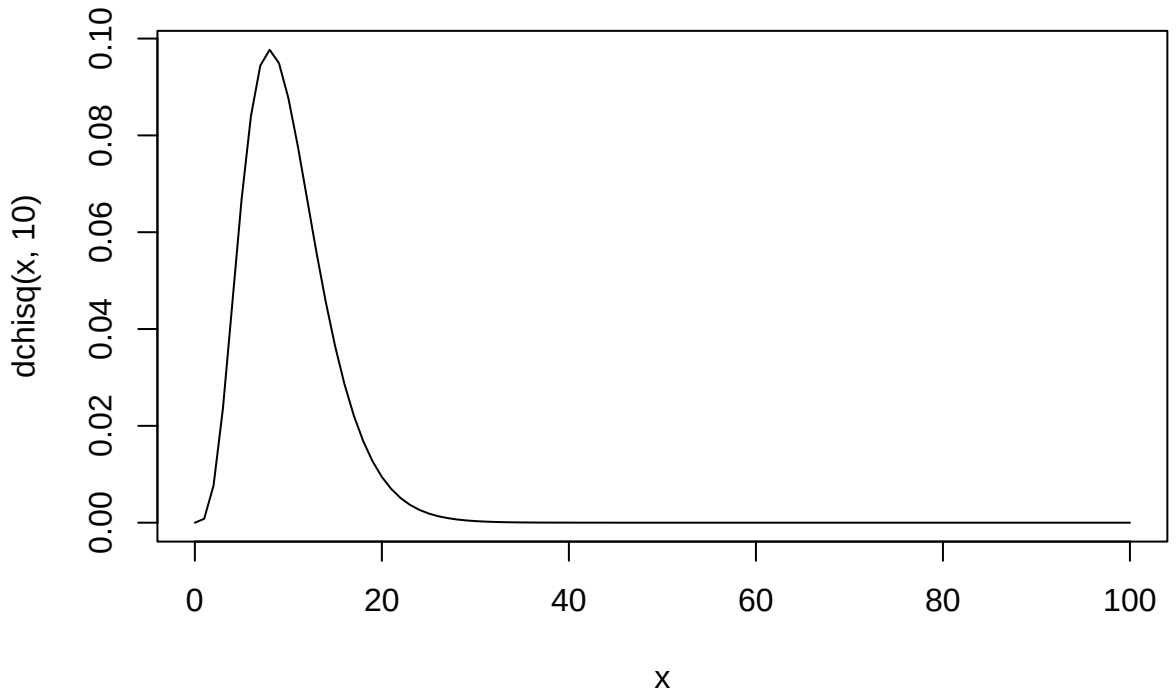
Problem 1. Q: Use the qnorm function in R to find the quartiles (i.e., 25th, 50th and 75th percentiles) of the normal distribution with mean 100 and standard deviation 10. A:

```
qnorm(c(0.25, 0.5, 0.75), 100, 10)
```

```
## [1]  93.2551 100.0000 106.7449
```

Problem 2. Use the curve function in R to display the graph of a $\chi^2(10)$ (10 corresponds to the degrees of freedom). Use a range of 0 to 100 for the x-axis. The chi-square density function is dchisq.

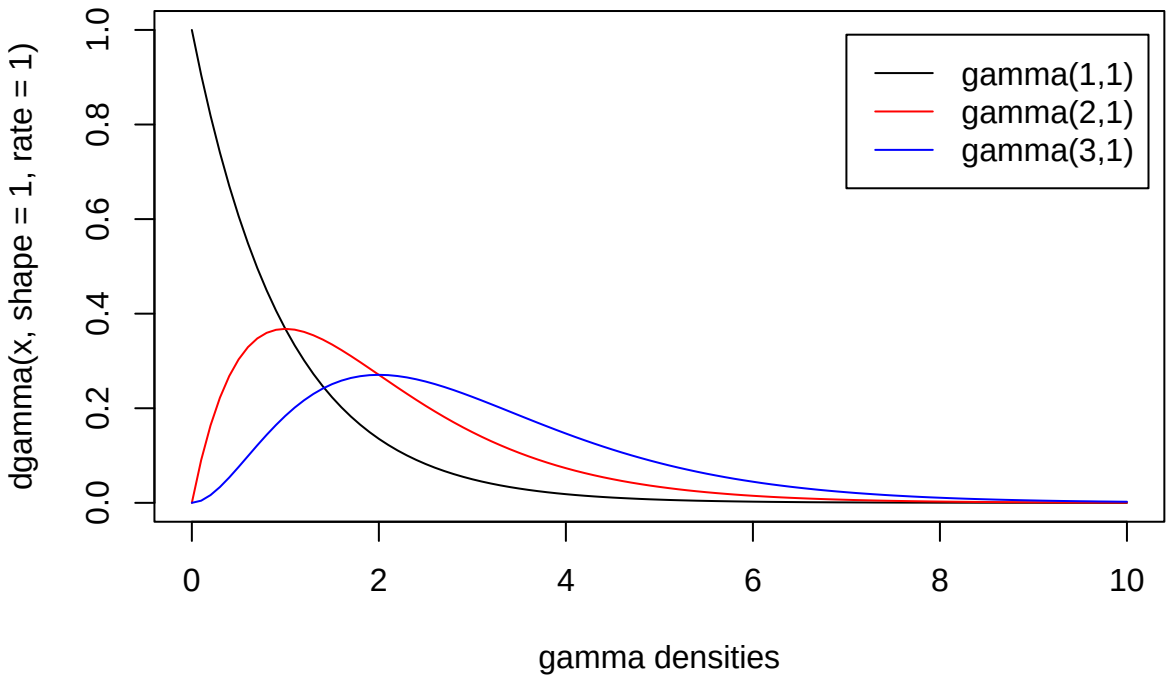
```
curve(dchisq(x, 10), xlim = c(0, 100))
```



Problem3. (Albert and Rizzo Chapter 1, problem 3.) (Gamma densities). Use the curve function to display the graph of the gamma density with shape parameter 1 and rate parameter 1. Then use the curve function with add=TRUE to display the graphs of the gamma density with shape parameter k and rate 1 for 2,3, all in the same graphics window. The gamma density function is dgamma. Consult the help file ?dgamma to see

how to specify the parameters.

```
curve(dgamma(x, shape = 1, rate = 1), xlim = c(0, 10),
  ylim = c(0, 1), xlab = "gamma densities")
curve(dgamma(x, shape = 2, rate = 1), add = TRUE, col = "red")
curve(dgamma(x, shape = 3, rate = 1), add = TRUE, col = "blue")
legend(7, 0.99, c("gamma(1,1)", "gamma(2,1)", "gamma(3,1)"), lty = 1,
  col = c("black", "red", "blue"), cex = 1)
```



Problem4. (Albert and Rizzo Chapter 1, problem 5.) (Binomial CDF). Let X be the number of "ones" obtained in 12 rolls of a fair die. Then X has a Binomial(n = 12, p = 1/3) distribution. Compute a table of cumulative binomial probabilities (the CDF) for x = 0,1,...,12 by two methods: (1) using cumsum and the result of Exercise 1.4, and (2) using the pbinom function. What is P(X > 7)?

Following the direction of (1), I'll use 'dbinom' function

```
cumsum(dbinom(0:12, 12, 1 / 3))
```

```
##  [1] 0.007707347 0.053951426 0.181122646 0.393074678 0.631520714 0.822277544
##  [7] 0.933552360 0.981241568 0.996144445 0.999456196 0.999952958 0.999998118
## [13] 1.000000000
```

```
1 - cumsum(dbinom(0:12, 12, 1 / 3))[8]
```

```
## [1] 0.01875843
```

Next, following (2), using 'pbinom' function,

```
pbinom(0:12, 12, 1 / 3)
```

```
##  [1] 0.007707347 0.053951426 0.181122646 0.393074678 0.631520714 0.822277544
##  [7] 0.933552360 0.981241568 0.996144445 0.999456196 0.999952958 0.999998118
## [13] 1.000000000
```

```
1 - pbinom(7, 12, 1 / 3)
```

```
## [1] 0.01875843
```

Two vectors of cdf values are exactly same. The P(x>7) values are also same.

Problem5. (Albert and Rizzo Chapter 1, problem 7.) (Simulated "horsekicks" data). The rpois function generates random observations from a Poisson distribution. In Example 1.3, we compared the deaths due to horsekicks to a Poisson distribution with mean $\lambda = 0.61$, and in Example 1.4 we simulated random Poisson($\lambda = 0.61$) data. Use the rpois function to simulate very large (n = 1000 and n = 10000) Poisson($\lambda = 0.61$) random samples. Find the frequency distribution, mean and variance for the sample. Compare the theoretical Poisson density with the sample proportions (see Example 1.4).

When n=1000,

```
pois_sample_1000 <- rpois(1000, 0.61)
empirical_1000 <-  table(pois_sample_1000) / 1000
theoretical_1000 <- dpois(0:(length(empirical_1000) - 1), 0.61)
rbind(empirical_1000,  theoretical_1000)
```

```
##                          0        1         2          3           4
## empirical_1000   0.5540000 0.322000 0.0950000 0.02700000 0.002000000
## theoretical_1000 0.5433509 0.331444 0.1010904 0.02055505 0.003134646
```

```
mean(pois_sample_1000)
```

```
## [1] 0.601
```

```
var(pois_sample_1000)
```

```
## [1] 0.6164154
```

When n=10000,

```
pois_sample_10000 <- rpois(10000, 0.61)
empirical_10000 <-  table(pois_sample_10000) / 10000
theoretical_10000 <- dpois(0:(length(empirical_10000) - 1), 0.61)
rbind(empirical_10000,  theoretical_10000)
```

```
##                           0        1         2          3           4
## empirical_10000   0.5425000 0.331700 0.0990000 0.02190000 0.004300000
## theoretical_10000 0.5433509 0.331444 0.1010904 0.02055505 0.003134646
##                             5
## empirical_10000   0.0006000000
## theoretical_10000 0.0003824268
```

```
mean(pois_sample_10000)
```

```
## [1] 0.6156
```

```
var(pois_sample_10000)
```

```
## [1] 0.6296996
```

The result of n=10000 is closer to the theoretical result than the case of n=1000.

Problem6. (Albert and Rizzo Chapter 1, problem 8.) (horsekicks, continued). Refer to Example 1.3. Using the ppois function, compute the cumulative distribution function (CDF) for the Poisson distribution with

mean $\lambda = 0.61$, for the values 0 to 4. Compare these probabilities with the empirical CDF. The empirical CDF is the cumulative sum of the sample proportions p, which is easily computed using the cumsum function. Combine the values of 0:4, the CDF, and the empirical CDF in a matrix to display these results in a single table.

```
pois_sample_10000 <- rpois(10000, 0.61)
empirical_density <-  (table(pois_sample_10000) / 10000)[1:5]
empirical_cdf <- cumsum(empirical_density)
theoretical_cdf <- ppois(0:4, 0.61)
print(rbind(empirical_cdf, theoretical_cdf))
```

```
##                          0         1         2         3        4
## empirical_cdf   0.5371000 0.8737000 0.9750000 0.9964000 1.000000
## theoretical_cdf 0.5433509 0.8747949 0.9758853 0.9964404 0.999575
```

the CDF and the empirical CDF are quite simillar. If I generate more samples, they will be more similar.

Problem7. (Albert and Rizzo Chapter 1, problem 9.) (Custom standard deviation function) Write a function sd.n similar to the function var.n in Example 1.5 that will return the estimate $\hat{\sigma}$ (the square root of $\hat{\sigma}^2$). Try this function on the temperature data of Example 1.1

```
sd.n <- function(x) {
  n <- NROW(x)
  var_n <- var(x) * (n - 1) / n
  return(sqrt(var_n))
}
temperature_data <- c(51.9, 51.8, 51.9, 53)
names(temperature_data) <- 1968 : 1971
print(temperature_data)
```

```
## 1968 1969 1970 1971
## 51.9 51.8 51.9 53.0
```

```
print(sd.n(temperature_data))
```

```
## [1] 0.4924429
```

Problem8. (Albert and Rizzo Chapter 1, problem 10.) (Euclidean norm function) Write a function norm that will compute the Euclidean norm of a numeric vector. The Euclidean norm of a vector $x = (x_1, ..., x_n)$ is

$$||x|| = \sqrt{\sum_{i=1}^{n} x_i^2}$$

Use vectorized operations to compute the sum. Try this function on the vectors (0,0,0,1) and (2,5,2,4) to check that your function result is correct.

```
euclidean_norm <- function(vec) {
  return(sqrt(sum(vec^2)))
}

test_vector1 <- c(0, 0, 0, 1)
test_vector2 <- c(2, 5, 2, 4)

print(euclidean_norm(test_vector1))
```

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

```
print(euclidean_norm(test_vector2))
```

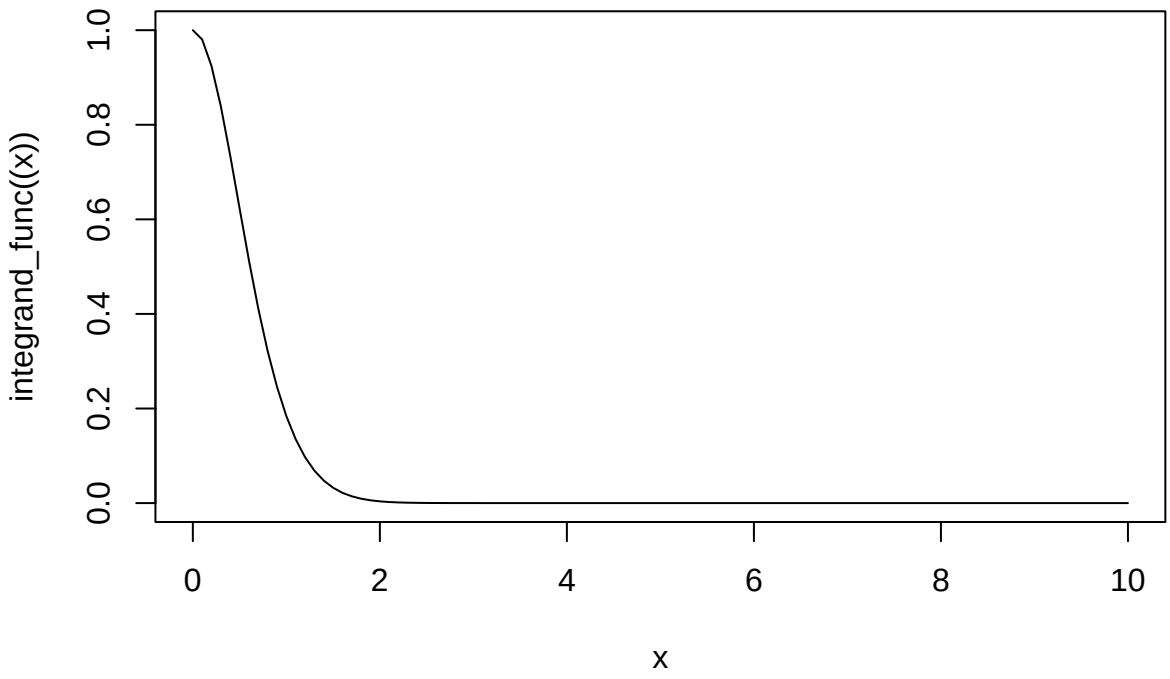
```
## [1] 7
```

Problem9. (Albert and Rizzo Chapter 1, problem 11.) (Numerical integration) Use the curve function to display the graph of the function $f(x) = e^{-x^2}/(1+x^2)$ on the interval $0 \leq x \leq 10$. Then use the integrate function to compute the value of the integral

$$\int_0^\infty \frac{e^{-x^2}}{1+x^2} dx$$

The upper limit at infinity is specified by upper=Inf in the integrate function.

```
integrand_func <- function(x) {
  return(exp(-x^2) / (1 + x^2))
}
curve(integrand_func((x)), from = 0, to = 10)
```




```
print(integrate(integrand_func, 0, Inf))
```

```
## 0.6716467 with absolute error < 8.3e-05
```

Problem10. (Albert and Rizzo Chapter 1, problem 12.) (Bivariate normal) Construct a matrix with 10 rows and 2 columns, containing random standard normal data:

x = matrix(rnorm(20), 10, 2)

This is a random sample of 10 observations from a standard bivariate normal distribution. Use the apply function and your norm function from Exercise 1.10 to compute the Euclidean norms for each of these 10 observations.

```r
euclidean_norm <- function(vec) {
  return(sqrt(sum(vec^2)))
}

x <- matrix(rnorm(20), 10, 2)
euclidean_norm(x[1, ]) #for check
```

```
## [1] 0.7885456
```

```r
print(apply(x, 1, euclidean_norm)) #margin 1 : row-wise
```

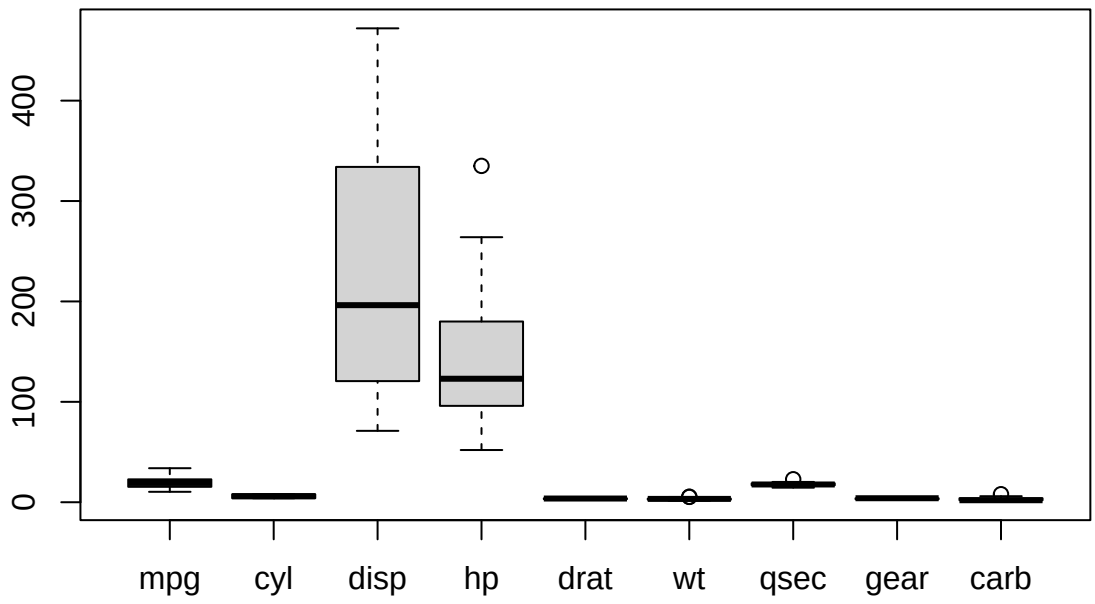
```
##  [1] 0.7885456 1.3535537 1.0035891 0.8025339 2.9266485 1.8386097 2.2349809
##  [8] 0.6365798 0.4108031 0.5940520
```

Problem11. (Albert and Rizzo Chapter 2, problem 3.) Briefly describe your findings. (mtcars data) Display the mtcars data included with R and read the documentation using ?mtcars. Display parallel boxplots of the quantitative variables. Display a pairs plot of the quantitative variables. Does the pairs plot reveal any possible relations between the variables?

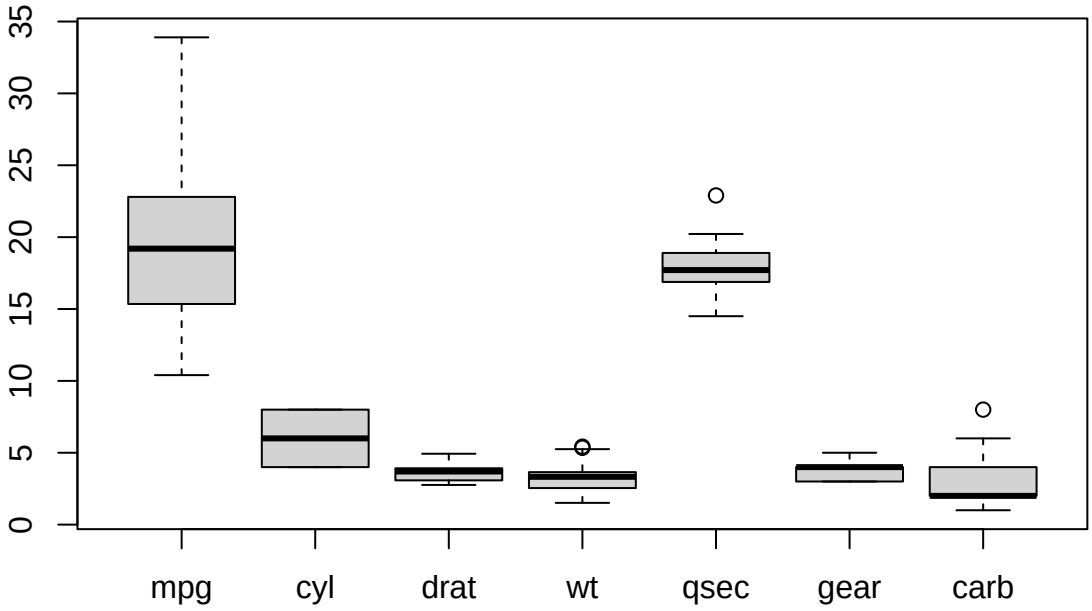
```r
data(mtcars)
# ?mtcars
head(mtcars)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

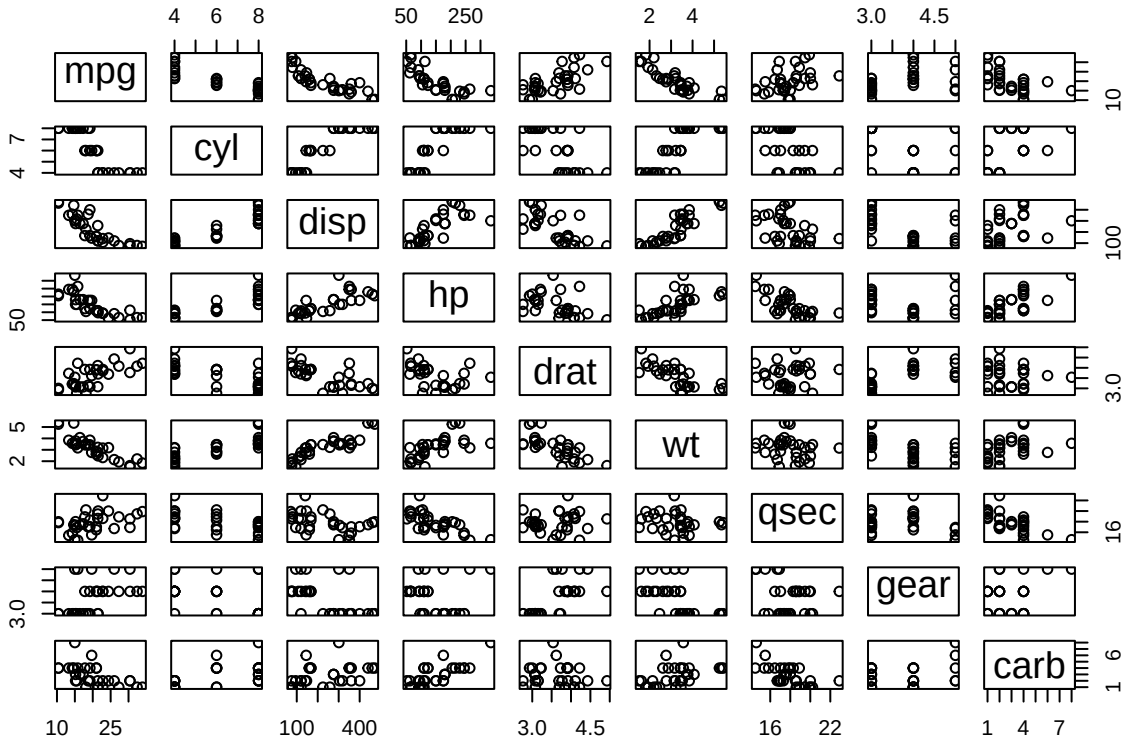
```r
mtcars_quantitative <- mtcars[, c(-8, -9)]
boxplot(mtcars_quantitative)
```




```
boxplot(mtcars_quantitative[, c(-3, -4)])
```




```
pairs(mtcars_quantitative)
```



Yes, the plot reveal some relations between the variables. Pairs such as mpg-disp, mpg-hp, mpg-drat, mpg-wt, mpg-qsec, mpg-carb cyl-disp, cyl-hp, cyl-wt, cyl-qsec, disp-hp, disp-wt, hp-wt, wp-qsec, hp-carb, drat-wp seem to have linear relations.

Problem12. (Albert and Rizzo Chapter 2, problems 4 and 5.) Briefly describe your findings. Refer to Example 2.7. Create a new variable r equal to the ratio of brain size over body size. Using the full mammals data set, order the mammals data by the ratio r. Which mammals have the largest ratios of brain size to body size? Which mammals have the smallest ratios? (Hint: use head and tail on the ordered data.)

```
library(MASS)
data(mammals)
# ?mammals
mammals[which.max(mammals$brain), ] #max brain
```

```
##                  body brain
## African elephant 6654  5712
```

```
mammals[which.min(mammals$brain), ] #min brain
```

```
##                             body brain
## Lesser short-tailed shrew 0.005  0.14
```

```
r <- mammals$brain / mammals$body
mammals[which.max(r), ] #max ratio
```

```
##                  body brain
## Ground squirrel 0.101     4
```

```
mammals[which.min(r), ] #min ratio
```

```
##                  body brain
## African elephant 6654  5712
```

```
mammals$ratio <- r
ordered_mammals <- mammals[order(mammals$ratio), ]

head(ordered_mammals)
```

```
##                    body  brain     ratio
## African elephant 6654.0 5712.0 0.8584310
## Cow               465.0  423.0 0.9096774
## Pig               192.0  180.0 0.9375000
## Brazilian tapir   160.0  169.0 1.0562500
## Water opossum       3.5    3.9 1.1142857
## Horse             521.0  655.0 1.2571977
```

```
tail(ordered_mammals)
```

```
##                             body  brain    ratio
## Galago                     0.200   5.00 25.00000
## Little brown bat           0.010   0.25 25.00000
## Rhesus monkey              6.800 179.00 26.32353
## Lesser short-tailed shrew 0.005   0.14 28.00000
## Owl monkey                 0.480  15.50 32.29167
## Ground squirrel            0.101   4.00 39.60396
```

Although African elephants have the most weighted brain, their ratio is the smallest. This is because their bodies are very heavy.

Problem13. (Albert and Rizzo Chapter 2, problems 7 and 8.) Briefly describe your findings. (2.7) (Central Limit Theorem with simulated data). Refer to Example 2.6, where we computed sample means for each row of the randu data frame. Repeat the analysis, but instead of randu, create a matrix of random numbers using runif.

```
unif_3samples_mat <- matrix(runif(400 * 3, 0, 1), 400, 3)
unif_3samples_df <- as.data.frame(unif_3samples_mat)
colnames(unif_3samples_df) <- c("x", "y", "z")
head(unif_3samples_df)
```

```
##           x          y          z
## 1 0.60041766 0.08880099 0.60447679
## 2 0.22968132 0.10774204 0.57528284
## 3 0.40007380 0.30734595 0.08284707
## 4 0.07487847 0.40516203 0.42598210
## 5 0.69316587 0.70422701 0.88747480
## 6 0.10039409 0.17244500 0.84960543
```

```
colMeans(unif_3samples_df) # theoretically: 0.5
```

```
##         x         y         z
## 0.5028975 0.5134410 0.4867951
```

```
var(unif_3samples_df) # theoretically: 0.08333 for diagonal elements, otherwise 0
```

```
##             x           y           z
## x  0.085644253  0.000370557 -0.001068345
## y  0.000370557  0.080364601 -0.001134955
## z -0.001068345 -0.001134955  0.087747684
```

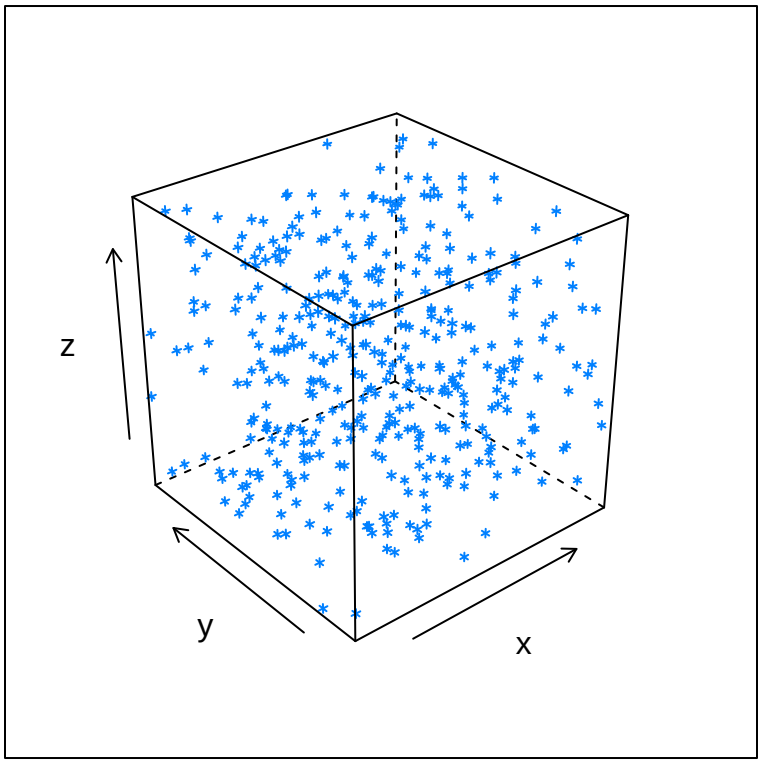
```
diag(var(unif_3samples_df))
```

```
##          x          y          z
## 0.08564425 0.08036460 0.08774768
```

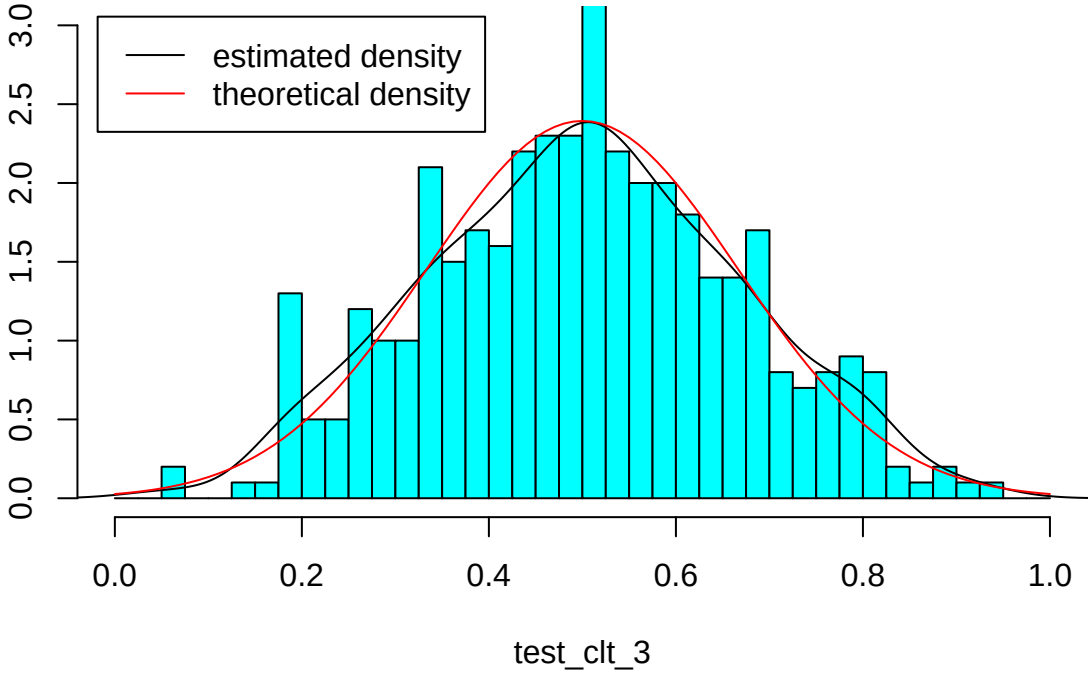
```
cor(unif_3samples_df)
```

```
##              x            y           z
## x  1.000000000  0.004466563 -0.01232380
## y  0.004466563  1.000000000 -0.01351539
## z -0.012323797 -0.013515390  1.00000000
```

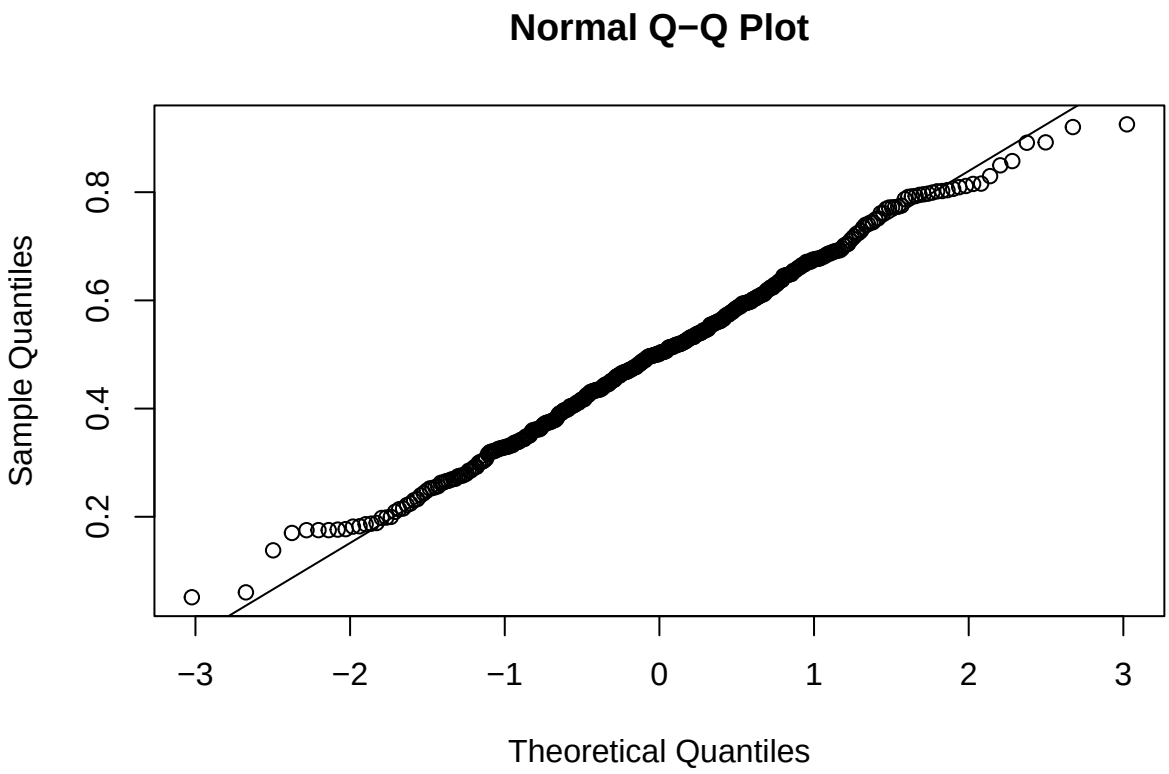
```
library(lattice)
cloud(z ~ x + y, data = unif_3samples_df) #seems good!
```




```
test_clt_3 <- rowMeans(unif_3samples_df)
library(MASS)
truehist(test_clt_3, xlim = c(0, 1), ylim = c(0, 3), breaks = seq(0, 1, by = 0.025))
lines(density(test_clt_3))
curve(dnorm(x, 1 / 2, sd = sqrt(1 / 12 / 3)), add = TRUE, col = "red")
legend("topleft", lty = c(1, 1), col = c("black", "red"), legend = c("estimated density", "theoretical
```




```
qqnorm(test_clt_3)
qqline(test_clt_3)
```



The 'runif' function of R generates good pseudo random numbers. The samples from the 'runif' function have reasonable mean, variance and correlation values. And, the CLT is realized on the samples. (Sometimes, samples from 'wrongly implemented random number generater' fail to realize the CLT.) Moreover, when drawing the 3-D plot, samples from the 'runif' are better than the samples of 'randu' data frame, because there is no linear relationship like samples of 'randu'.

(2.8) (Central Limit Theorem, continued). Refer to Example 2.6 and Exercise 2.7, where we computed sample means for each row of the data frame. Repeat the analysis in Exercise 2.7, but instead of sample size 3 generate a matrix that is 400 by 10 (sample size 10). Compare the histogram for sample size 3 and sample size 10. What does the Central Limit Theorem tell us about the distribution of the mean as sample size increases?

```
unif_10samples_mat <- matrix(runif(400 * 10, 0, 1), 400, 10)
unif_10samples_df <- as.data.frame(unif_10samples_mat)
colnames(unif_10samples_df) <- 1:10
head(unif_10samples_df)
```

```
##           1         2         3           4         5         6          7
## 1 0.1259570 0.69506353 0.5814667 0.217379879 0.5466410 0.83876255 0.30658215
## 2 0.1605833 0.45686170 0.8292227 0.280352064 0.7952748 0.09743505 0.13944860
## 3 0.7315002 0.90596056 0.6205585 0.441809678 0.9031549 0.03448270 0.05487191
## 4 0.3770060 0.08169967 0.5205734 0.001682727 0.3734106 0.75416846 0.64614174
## 5 0.9857340 0.38802383 0.4294132 0.281744237 0.5831644 0.49944368 0.52491085
## 6 0.7894868 0.05560859 0.4022471 0.460116485 0.6153189 0.51223548 0.70281827
##           8         9        10
## 1 0.2537005 0.4192296 0.8325038
## 2 0.9696313 0.4062205 0.9834559
## 3 0.6595015 0.2745265 0.6570130
```

```
## 4 0.4566412 0.3085388 0.5193305
## 5 0.8759892 0.7807173 0.2541026
## 6 0.6628377 0.6360937 0.6873998
```

```
colMeans(unif_10samples_df) # theoretically: 0.5
```

```
##         1         2         3         4         5         6         7         8
## 0.4598283 0.4720451 0.5021333 0.4640629 0.5016000 0.5330650 0.4997752 0.5114539
##         9        10
## 0.5114239 0.5110013
```

```
var(unif_10samples_df) # theoretically: 0.08333 for diagonal elements, otherwise 0
```

```
##                1             2             3             4             5
## 1   0.0853201349 -1.056443e-03  0.0009870078  0.0004480662 -0.0014335353
## 2  -0.0010564432  8.390285e-02  0.0015257575 -0.0023430897 -0.0020180657
## 3   0.0009870078  1.525758e-03  0.0824920728  0.0031376468  0.0012567377
## 4   0.0004480662 -2.343090e-03  0.0031376468  0.0848024337  0.0065376928
## 5  -0.0014335353 -2.018066e-03  0.0012567377  0.0065376928  0.0763092675
## 6  -0.0010088237 -9.327446e-04 -0.0036372836 -0.0019458676  0.0038527680
## 7   0.0097684719 -3.607321e-03  0.0047163411  0.0032662891  0.0011497810
## 8   0.0005267709 -2.100827e-03  0.0005023218 -0.0023745651  0.0024050543
## 9   0.0043103557 -7.853587e-03 -0.0019896769 -0.0007518397 -0.0046108838
## 10 -0.0034074303 -3.233988e-05  0.0035736600 -0.0043220593  0.0004534634
##                6             7             8             9            10
## 1  -1.008824e-03  0.0097684719  5.267709e-04  0.0043103557 -3.407430e-03
## 2  -9.327446e-04 -0.0036073209 -2.100827e-03 -0.0078535870 -3.233988e-05
## 3  -3.637284e-03  0.0047163411  5.023218e-04 -0.0019896769  3.573660e-03
## 4  -1.945868e-03  0.0032662891 -2.374565e-03 -0.0007518397 -4.322059e-03
## 5   3.852768e-03  0.0011497810  2.405054e-03 -0.0046108838  4.534634e-04
## 6   8.895230e-02 -0.0067955462  8.470348e-05 -0.0070613765  1.098142e-02
## 7  -6.795546e-03  0.0808679591 -3.046378e-04  0.0074691880 -1.324658e-03
## 8   8.470348e-05 -0.0003046378  8.876972e-02 -0.0007948907  1.499110e-03
## 9  -7.061377e-03  0.0074691880 -7.948907e-04  0.0872235769 -6.143095e-03
## 10  1.098142e-02 -0.0013246585  1.499110e-03 -0.0061430949  7.846911e-02
```

```
diag(var(unif_10samples_df))
```

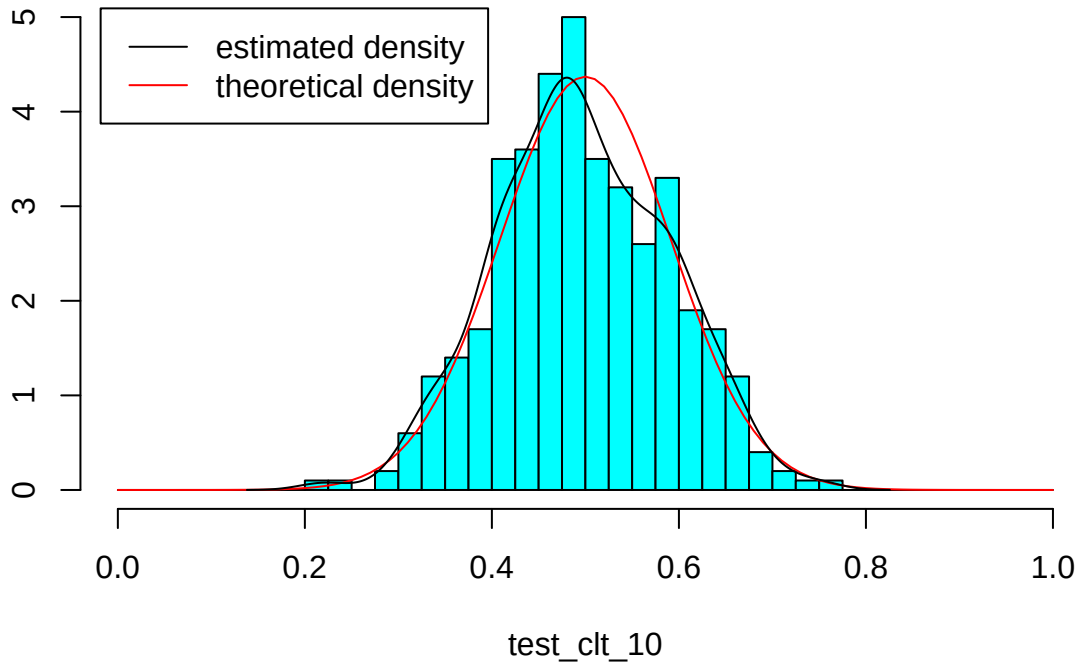
```
##          1          2          3          4          5          6          7
## 0.08532013 0.08390285 0.08249207 0.08480243 0.07630927 0.08895230 0.08086796
##          8          9         10
## 0.08876972 0.08722358 0.07846911
```

```
cor(unif_10samples_df)
```

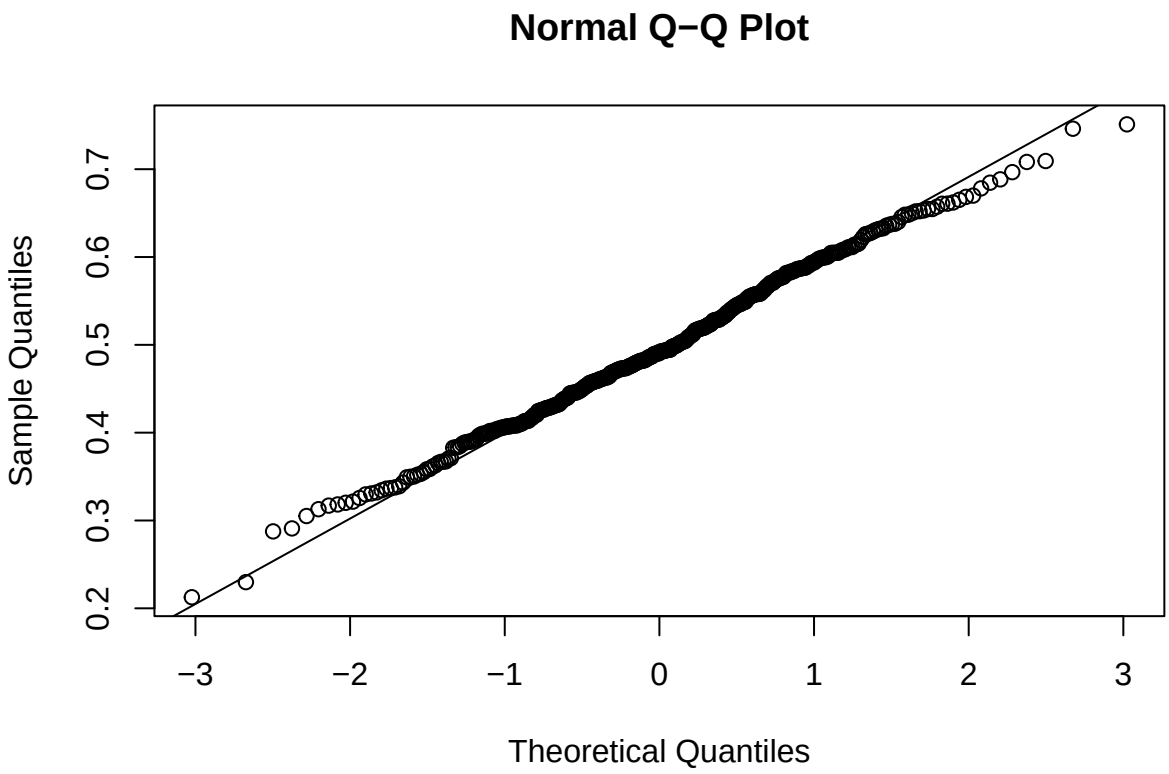
```
##               1             2            3             4           5
## 1   1.000000000 -0.0124862493  0.011764912  0.005267594 -0.01776618
## 2  -0.012486249  1.0000000000  0.018339650 -0.027777707 -0.02522077
## 3   0.011764912  0.0183396499  1.000000000  0.037514036  0.01583981
## 4   0.005267594 -0.0277777066  0.037514036  1.000000000  0.08127027
## 5  -0.017766177 -0.0252207672  0.015839807  0.081270271  1.00000000
## 6  -0.011580063 -0.0107968176 -0.042461220 -0.022404257  0.04676335
## 7   0.117601435 -0.0437933506  0.057744531  0.039442299  0.01463653
## 8   0.006052901 -0.0243427416  0.005870072 -0.027368279  0.02922159
## 9   0.049965518 -0.0918042415 -0.023456299 -0.008741866 -0.05651693
## 10 -0.041643942 -0.0003985663  0.044417872 -0.052983078  0.00586009
##                6            7             8            9            10
```

```
## 1  -0.0115800633  0.117601435  0.0060529013  0.049965518 -0.0416439422
## 2  -0.0107968176 -0.043793351 -0.0243427416 -0.091804241 -0.0003985663
## 3  -0.0424612200  0.057744531  0.0058700724 -0.023456299  0.0444178716
## 4  -0.0224042570  0.039442299 -0.0273682794 -0.008741866 -0.0529830779
## 5   0.0467633514  0.014636534  0.0292215881 -0.056516933  0.0058600896
## 6   1.0000000000 -0.080123062  0.0009532136 -0.080166668  0.1314409340
## 7  -0.0801230620  1.000000000 -0.0035955316  0.088934077 -0.0166290064
## 8   0.0009532136 -0.003595532  1.0000000000 -0.009033542  0.0179618777
## 9  -0.0801666683  0.088934077 -0.0090335416  1.000000000 -0.0742541789
## 10  0.1314409340 -0.016629006  0.0179618777 -0.074254179  1.0000000000
```

```
test_clt_10 <- rowMeans(unif_10samples_df)
# library(MASS)
truehist(test_clt_10, xlim = c(0, 1), ylim = c(0, 5), breaks = seq(0, 1, by = 0.025))
curve(dnorm(x, 1 / 2, sd = sqrt(1 / 12 / 10)), add = TRUE, col = "red")
lines(density(test_clt_10))
legend("topleft", lty = c(1, 1), col = c("black", "red"), legend = c("estimated density", "theoretical
```

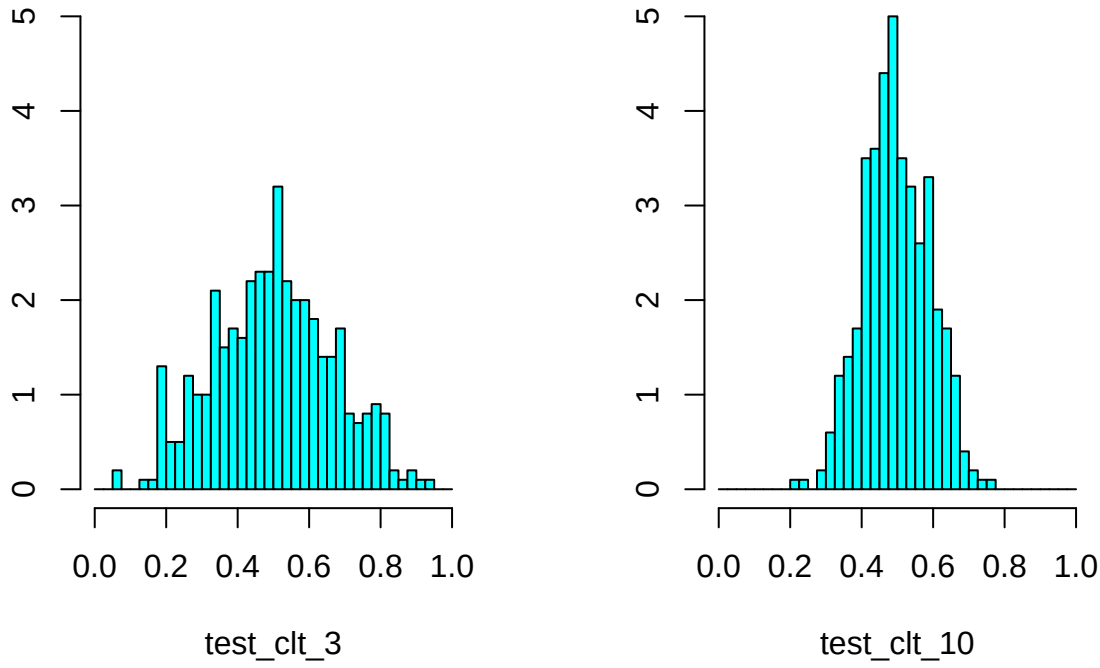



```
qqnorm(test_clt_10)
qqline(test_clt_10)
```



To compare case of 3-sample with 10-sample,

```
par(mfrow = c(1, 2))
hist_ax <- seq(0, 1, by = 0.025)
truehist(test_clt_3, breaks = hist_ax, xlim = c(0, 1), ylim = c(0, 5))
truehist(test_clt_10, breaks = hist_ax, xlim = c(0, 1), ylim = c(0, 5))
```



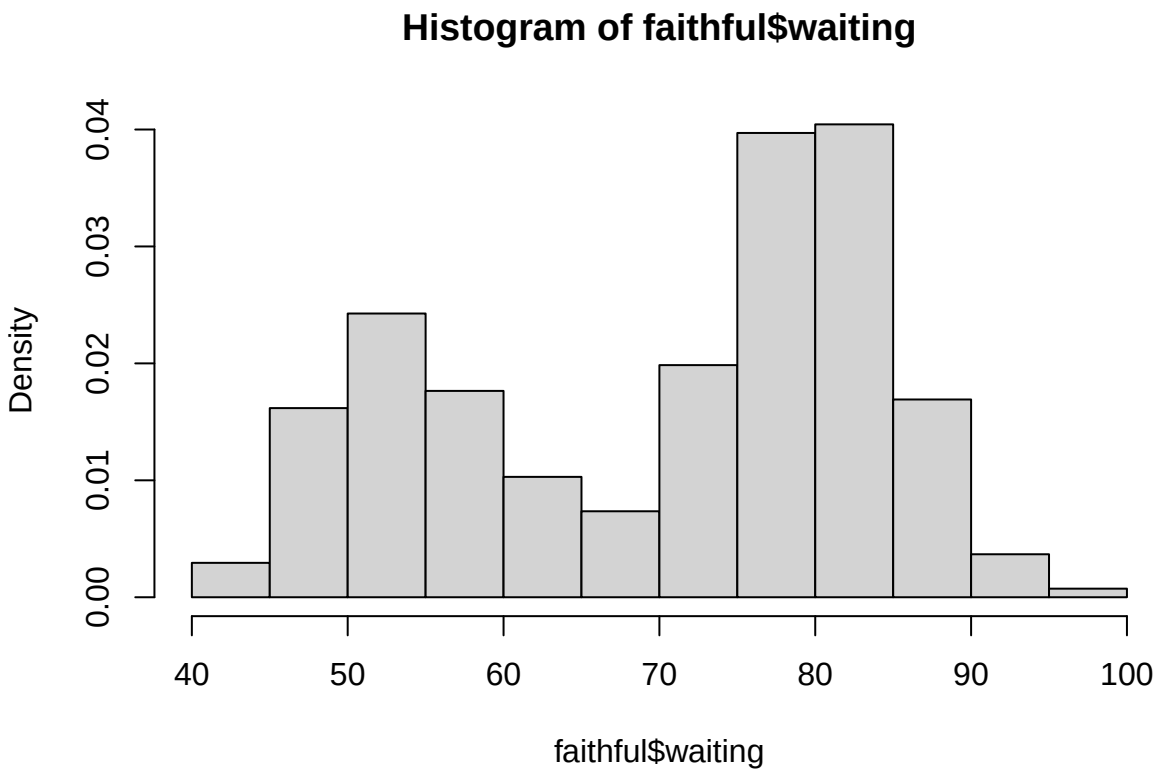
The variance of the mean values' distribution becomes small as the sample size increases. Thus, sample mean tends to be close to the true mean value.

Problem14. (Albert and Rizzo Chapter 2, problems 10 and 11.) (2.10) ("Old Faithful" histogram). Use hist to display a probability histogram of the waiting times for the Old Faithful geyser in the faithful data set (see Example A.3). (Use the argument prob=TRUE or freq=FALSE.)

```
data(faithful)
head(faithful)
```

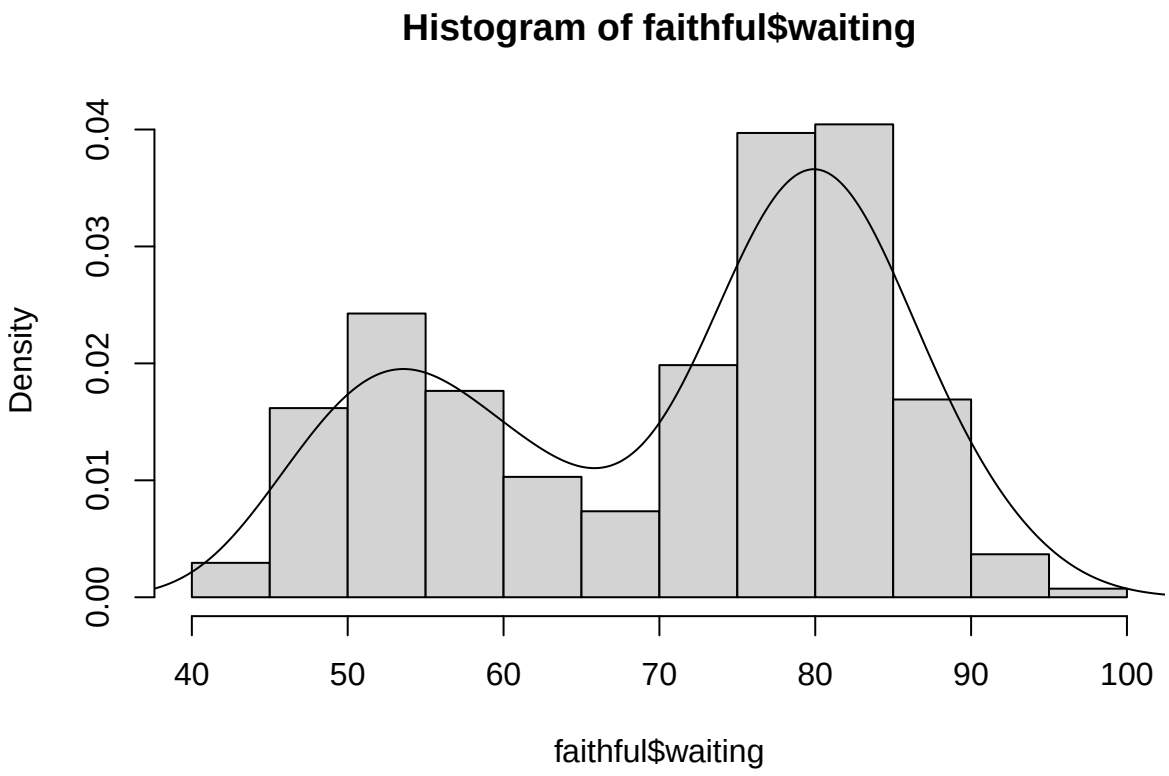
```
##   eruptions waiting
## 1     3.600      79
## 2     1.800      54
## 3     3.333      74
## 4     2.283      62
## 5     4.533      85
## 6     2.883      55
```

```
par(mfrow = c(1, 1))
hist(faithful$waiting, probability = TRUE)
```



(2.11) ("Old Faithful" density estimate). Use hist to display a probability histogram of the waiting times for the Old Faithful geyser in the faithful data set (see Example A.3) and add a density estimate using lines

```
hist(faithful$waiting, probability = TRUE)
lines(density(faithful$waiting))
```



problem15. (James et al. Chapter 2, problem 8.) This exercise relates to the College data set, which can be found in the file College.csv. It contains a number of variables for 777 different universities and colleges in the US. The variables are

- Private : Public/private indicator
- Apps : Number of applications received
- Accept : Number of applicants accepted
- Enroll : Number of new students enrolled
- Top10perc : New students from top 10 % of high school class
- Top25perc : New students from top 25 % of high school class
- F.Undergrad : Number of full-time undergraduates
- P.Undergrad : Number of part-time undergraduates
- Outstate : Out-of-state tuition
- Room.Board : Room and board costs
- Books : Estimated book costs
- Personal : Estimated personal spending
- PhD : Percent of faculty with Ph.D.'s
- Terminal : Percent of faculty with terminal degree
- S.F.Ratio : Student/faculty ratio
- perc.alumni : Percent of alumni who donate
- Expend : Instructional expenditure per student
- Grad.Rate : Graduation rate

Before reading the data into R, it can be viewed in Excel or a text editor.

(a) Use the read.csv() function to read the data into R. Call the loaded data college. Make sure that you have the directory set to the correct location for the data.

Instead of loading from the 'College.csv' file, I load the data from the ISLR package.

```
library(ISLR)
data(College)
# ?College
head(College)
```

```
##                              Private Apps Accept Enroll Top10perc Top25perc
## Abilene Christian University     Yes 1660   1232    721        23        52
## Adelphi University               Yes 2186   1924    512        16        29
## Adrian College                   Yes 1428   1097    336        22        50
## Agnes Scott College              Yes  417    349    137        60        89
## Alaska Pacific University        Yes  193    146     55        16        44
## Albertson College                Yes  587    479    158        38        62
##                              F.Undergrad P.Undergrad Outstate Room.Board Books
## Abilene Christian University        2885         537     7440       3300   450
## Adelphi University                  2683        1227    12280       6450   750
## Adrian College                      1036          99    11250       3750   400
## Agnes Scott College                  510          63    12960       5450   450
## Alaska Pacific University            249         869     7560       4120   800
## Albertson College                    678          41    13500       3335   500
##                              Personal PhD Terminal S.F.Ratio perc.alumni Expend
## Abilene Christian University     2200  70       78      18.1          12   7041
## Adelphi University               1500  29       30      12.2          16  10527
## Adrian College                   1165  53       66      12.9          30   8735
## Agnes Scott College               875  92       97       7.7          37  19016
## Alaska Pacific University        1500  76       72      11.9           2  10922
## Albertson College                 675  67       73       9.4          11   9727
##                              Grad.Rate
## Abilene Christian University        60
## Adelphi University                  56
## Adrian College                      54
## Agnes Scott College                 59
## Alaska Pacific University           15
## Albertson College                   55
```

(b) Look at the data using the fix() function. You should notice that the first column is just the name of each university. We don't really want R to treat this as data. However, it may be handy to have these names for later. Try the following commands:

```
college <- College
# rownames(college) <- college [, 1]
# fix(college)
```

You should see that there is now a row.names column with the name of each university recorded. This means that R has given each row a name corresponding to the appropriate university. R will not try to perform calculations on the row names. However, we still need to eliminate the first column in the data where the names are stored. Try

```
# college <- college[, -1]
# fix(college)
head(college)
```

```
##                              Private Apps Accept Enroll Top10perc Top25perc
## Abilene Christian University     Yes 1660   1232    721        23        52
## Adelphi University               Yes 2186   1924    512        16        29
## Adrian College                   Yes 1428   1097    336        22        50
```

```
## Agnes Scott College              Yes  417    349    137        60        89
## Alaska Pacific University        Yes  193    146     55        16        44
## Albertson College                Yes  587    479    158        38        62
##                               F.Undergrad P.Undergrad Outstate Room.Board Books
## Abilene Christian University        2885         537     7440       3300   450
## Adelphi University                  2683        1227    12280       6450   750
## Adrian College                      1036          99    11250       3750   400
## Agnes Scott College                  510          63    12960       5450   450
## Alaska Pacific University            249         869     7560       4120   800
## Albertson College                    678          41    13500       3335   500
##                               Personal PhD Terminal S.F.Ratio perc.alumni Expend
## Abilene Christian University     2200  70       78      18.1          12   7041
## Adelphi University               1500  29       30      12.2          16  10527
## Adrian College                   1165  53       66      12.9          30   8735
## Agnes Scott College               875  92       97       7.7          37  19016
## Alaska Pacific University        1500  76       72      11.9           2  10922
## Albertson College                 675  67       73       9.4          11   9727
##                               Grad.Rate
## Abilene Christian University         60
## Adelphi University                   56
## Adrian College                       54
## Agnes Scott College                  59
## Alaska Pacific University            15
## Albertson College                    55
```

(Note: College data in the ISLR package, this work is already done.)

Now you should see that the first data column is Private. Note that another column labeled row.names now appears before the Private column. However, this is not a data column but rather the name that R is giving to each row.

(c)

1. Use the summary() function to produce a numerical summary of the variables in the data set.

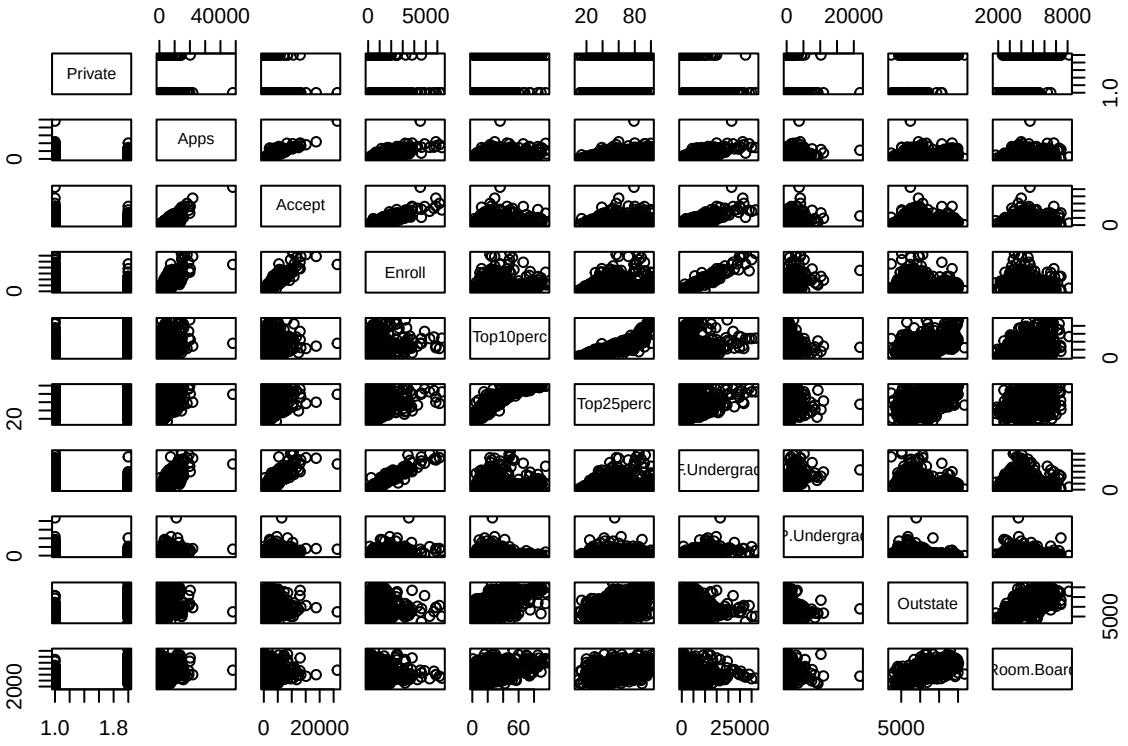
```
summary(college)
```

```
##  Private        Apps           Accept          Enroll       Top10perc
##  No :212   Min.   :   81   Min.   :   72   Min.   :  35   Min.   : 1.00
##  Yes:565   1st Qu.:  776   1st Qu.:  604   1st Qu.: 242   1st Qu.:15.00
##            Median : 1558   Median : 1110   Median : 434   Median :23.00
##            Mean   : 3002   Mean   : 2019   Mean   : 780   Mean   :27.56
##            3rd Qu.: 3624   3rd Qu.: 2424   3rd Qu.: 902   3rd Qu.:35.00
##            Max.   :48094   Max.   :26330   Max.   :6392   Max.   :96.00
##    Top25perc      F.Undergrad     P.Undergrad         Outstate
##  Min.   :  9.0   Min.   :  139   Min.   :    1.0   Min.   : 2340
##  1st Qu.: 41.0   1st Qu.:  992   1st Qu.:   95.0   1st Qu.: 7320
##  Median : 54.0   Median : 1707   Median :  353.0   Median : 9990
##  Mean   : 55.8   Mean   : 3700   Mean   :  855.3   Mean   :10441
##  3rd Qu.: 69.0   3rd Qu.: 4005   3rd Qu.:  967.0   3rd Qu.:12925
##  Max.   :100.0   Max.   :31643   Max.   :21836.0   Max.   :21700
##    Room.Board       Books           Personal         PhD
##  Min.   :1780   Min.   :  96.0   Min.   : 250   Min.   :  8.00
##  1st Qu.:3597   1st Qu.: 470.0   1st Qu.: 850   1st Qu.: 62.00
##  Median :4200   Median : 500.0   Median :1200   Median : 75.00
##  Mean   :4358   Mean   : 549.4   Mean   :1341   Mean   : 72.66
##  3rd Qu.:5050   3rd Qu.: 600.0   3rd Qu.:1700   3rd Qu.: 85.00
```

```
##  Max.   :8124   Max.   :2340.0   Max.   :6800   Max.   :103.00
##     Terminal       S.F.Ratio      perc.alumni        Expend
##  Min.   : 24.0   Min.   : 2.50   Min.   : 0.00   Min.   : 3186
##  1st Qu.: 71.0   1st Qu.:11.50   1st Qu.:13.00   1st Qu.: 6751
##  Median : 82.0   Median :13.60   Median :21.00   Median : 8377
##  Mean   : 79.7   Mean   :14.09   Mean   :22.74   Mean   : 9660
##  3rd Qu.: 92.0   3rd Qu.:16.50   3rd Qu.:31.00   3rd Qu.:10830
##  Max.   :100.0   Max.   :39.80   Max.   :64.00   Max.   :56233
##    Grad.Rate
##  Min.   : 10.00
##  1st Qu.: 53.00
##  Median : 65.00
##  Mean   : 65.46
##  3rd Qu.: 78.00
##  Max.   :118.00
```

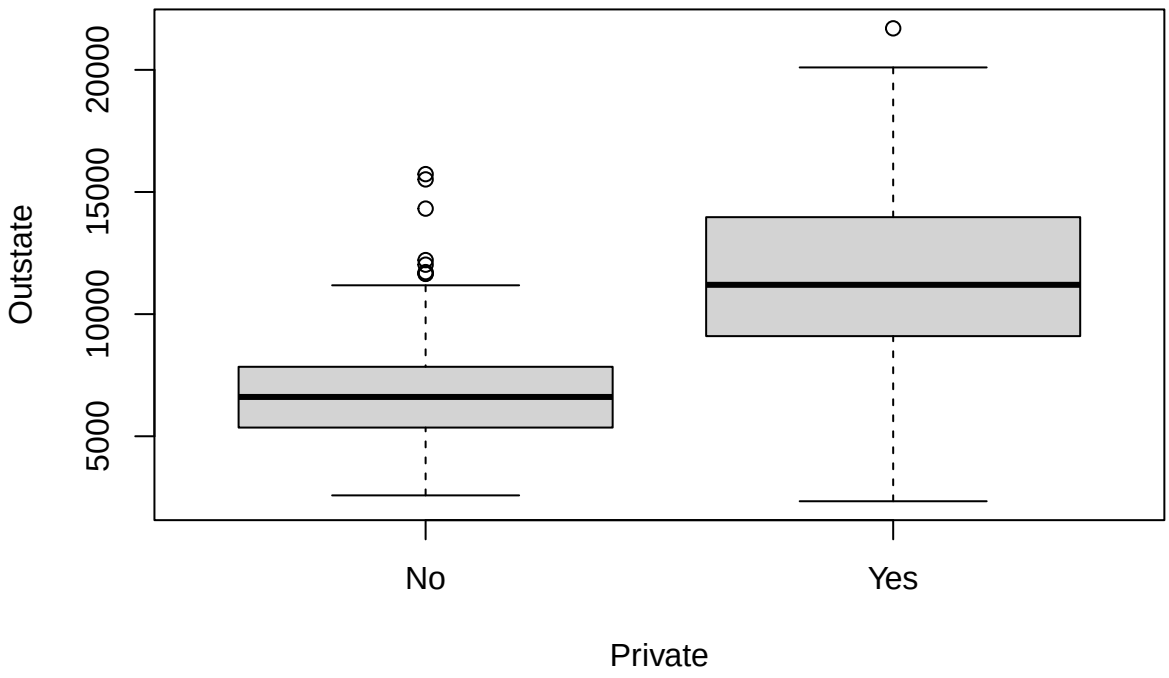
2. Use the pairs() function to produce a scatterplot matrix of the first ten columns or variables of the data. Recall that you can reference the first ten columns of a matrix A using A[, 1:10].

```
pairs(college[, 1:10])
```



3. Use the plot() function to produce side-by-side boxplots of Outstate versus Private.

```
plot(Outstate ~ Private, data = college)
```



We can find a difference by the value of Private. Although whiskers overlap, boxes do not.

4. Create a new qualitative variable, called Elite, by binning the Top10perc variable. We are going to divide universities into two groups based on whether or not the proportion of students coming from the top 10 % of their high school classes exceeds 50 %.

```
Elite <- rep("No", nrow(college))
Elite[college$Top10perc > 50] <- "Yes"
Elite <- as.factor(Elite)
college <- data.frame(college, Elite)
```

Use the summary() function to see how many elite universities there are.

```
summary(college)
```

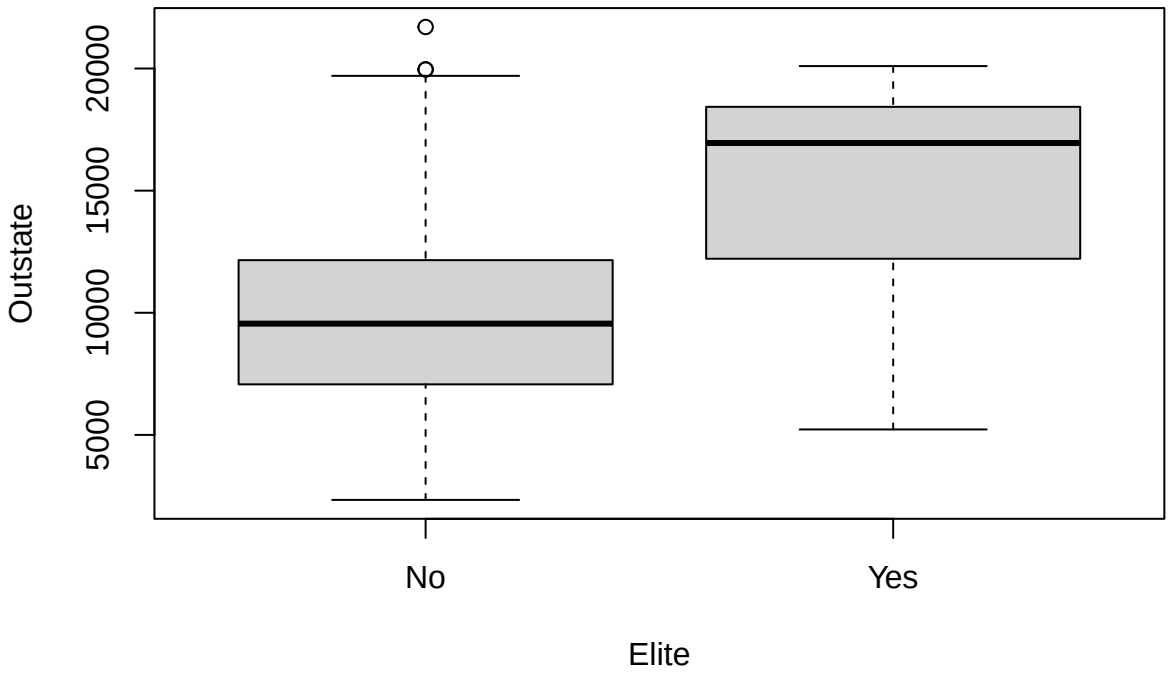
```
##  Private        Apps           Accept          Enroll       Top10perc
##  No :212   Min.   :   81   Min.   :   72   Min.   :  35   Min.   : 1.00
##  Yes:565   1st Qu.:  776   1st Qu.:  604   1st Qu.: 242   1st Qu.:15.00
##            Median : 1558   Median : 1110   Median : 434   Median :23.00
##            Mean   : 3002   Mean   : 2019   Mean   : 780   Mean   :27.56
##            3rd Qu.: 3624   3rd Qu.: 2424   3rd Qu.: 902   3rd Qu.:35.00
##            Max.   :48094   Max.   :26330   Max.   :6392   Max.   :96.00
##    Top25perc      F.Undergrad     P.Undergrad         Outstate
##  Min.   :  9.0   Min.   :  139   Min.   :    1.0   Min.   : 2340
##  1st Qu.: 41.0   1st Qu.:  992   1st Qu.:   95.0   1st Qu.: 7320
##  Median : 54.0   Median : 1707   Median :  353.0   Median : 9990
##  Mean   : 55.8   Mean   : 3700   Mean   :  855.3   Mean   :10441
##  3rd Qu.: 69.0   3rd Qu.: 4005   3rd Qu.:  967.0   3rd Qu.:12925
##  Max.   :100.0   Max.   :31643   Max.   :21836.0   Max.   :21700
```

```
##    Room.Board       Books           Personal         PhD
##  Min.   :1780   Min.   :  96.0   Min.   : 250   Min.   :  8.00
##  1st Qu.:3597   1st Qu.: 470.0   1st Qu.: 850   1st Qu.: 62.00
##  Median :4200   Median : 500.0   Median :1200   Median : 75.00
##  Mean   :4358   Mean   : 549.4   Mean   :1341   Mean   : 72.66
##  3rd Qu.:5050   3rd Qu.: 600.0   3rd Qu.:1700   3rd Qu.: 85.00
##  Max.   :8124   Max.   :2340.0   Max.   :6800   Max.   :103.00
##     Terminal       S.F.Ratio      perc.alumni        Expend
##  Min.   : 24.0   Min.   : 2.50   Min.   : 0.00   Min.   : 3186
##  1st Qu.: 71.0   1st Qu.:11.50   1st Qu.:13.00   1st Qu.: 6751
##  Median : 82.0   Median :13.60   Median :21.00   Median : 8377
##  Mean   : 79.7   Mean   :14.09   Mean   :22.74   Mean   : 9660
##  3rd Qu.: 92.0   3rd Qu.:16.50   3rd Qu.:31.00   3rd Qu.:10830
##  Max.   :100.0   Max.   :39.80   Max.   :64.00   Max.   :56233
##    Grad.Rate      Elite
##  Min.   : 10.00   No :699
##  1st Qu.: 53.00   Yes: 78
##  Median : 65.00
##  Mean   : 65.46
##  3rd Qu.: 78.00
##  Max.   :118.00
```

There are 78 elite universities.

Now use the plot() function to produce side-by-side boxplots of Outstate versus Elite.

```
plot(Outstate ~ Elite, data = college)
```



Not more than Outstate versus Privates, two boxes still does not overlap.

5. Use the hist() function to produce some histograms with differing numbers of bins for a few of the quantitative variables. You may find the command par(mfrow=c(2,2)) useful: it will divide the print window into four regions so that four plots can be made simultaneously. Modifying the arguments to this function will divide the screen in other ways.

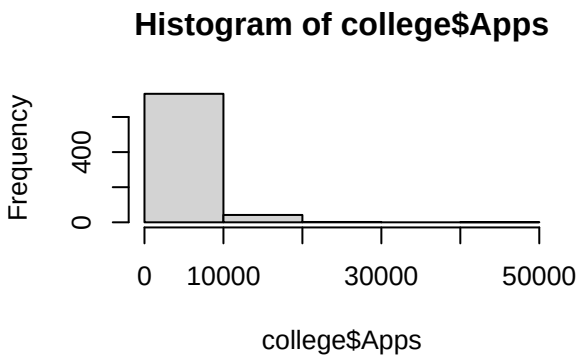
```
summary(college)
```

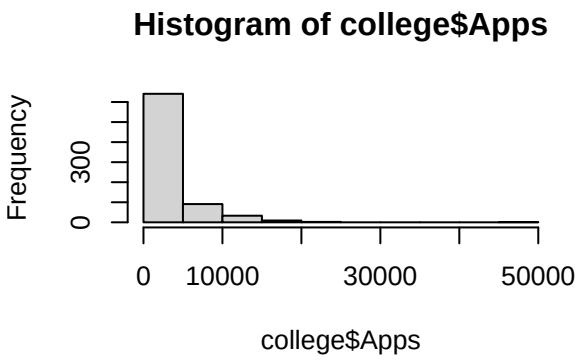
```
##  Private        Apps           Accept          Enroll       Top10perc
##  No :212   Min.   :   81   Min.   :   72   Min.   :  35   Min.   : 1.00
##  Yes:565   1st Qu.:  776   1st Qu.:  604   1st Qu.: 242   1st Qu.:15.00
##            Median : 1558   Median : 1110   Median : 434   Median :23.00
##            Mean   : 3002   Mean   : 2019   Mean   : 780   Mean   :27.56
##            3rd Qu.: 3624   3rd Qu.: 2424   3rd Qu.: 902   3rd Qu.:35.00
##            Max.   :48094   Max.   :26330   Max.   :6392   Max.   :96.00
##    Top25perc      F.Undergrad     P.Undergrad         Outstate
##  Min.   :  9.0   Min.   :  139   Min.   :    1.0   Min.   : 2340
##  1st Qu.: 41.0   1st Qu.:  992   1st Qu.:   95.0   1st Qu.: 7320
##  Median : 54.0   Median : 1707   Median :  353.0   Median : 9990
##  Mean   : 55.8   Mean   : 3700   Mean   :  855.3   Mean   :10441
##  3rd Qu.: 69.0   3rd Qu.: 4005   3rd Qu.:  967.0   3rd Qu.:12925
##  Max.   :100.0   Max.   :31643   Max.   :21836.0   Max.   :21700
##    Room.Board       Books           Personal         PhD
##  Min.   :1780   Min.   :  96.0   Min.   : 250   Min.   :  8.00
##  1st Qu.:3597   1st Qu.: 470.0   1st Qu.: 850   1st Qu.: 62.00
##  Median :4200   Median : 500.0   Median :1200   Median : 75.00
##  Mean   :4358   Mean   : 549.4   Mean   :1341   Mean   : 72.66
##  3rd Qu.:5050   3rd Qu.: 600.0   3rd Qu.:1700   3rd Qu.: 85.00
##  Max.   :8124   Max.   :2340.0   Max.   :6800   Max.   :103.00
##     Terminal       S.F.Ratio      perc.alumni        Expend
##  Min.   : 24.0   Min.   : 2.50   Min.   : 0.00   Min.   : 3186
##  1st Qu.: 71.0   1st Qu.:11.50   1st Qu.:13.00   1st Qu.: 6751
##  Median : 82.0   Median :13.60   Median :21.00   Median : 8377
##  Mean   : 79.7   Mean   :14.09   Mean   :22.74   Mean   : 9660
##  3rd Qu.: 92.0   3rd Qu.:16.50   3rd Qu.:31.00   3rd Qu.:10830
##  Max.   :100.0   Max.   :39.80   Max.   :64.00   Max.   :56233
##    Grad.Rate      Elite
##  Min.   : 10.00   No :699
##  1st Qu.: 53.00   Yes: 78
##  Median : 65.00
##  Mean   : 65.46
##  3rd Qu.: 78.00
##  Max.   :118.00
```

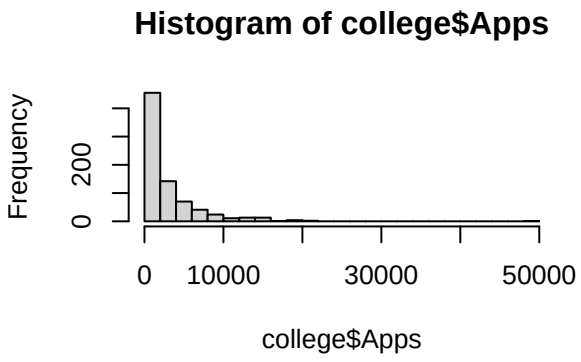
```
?College
```

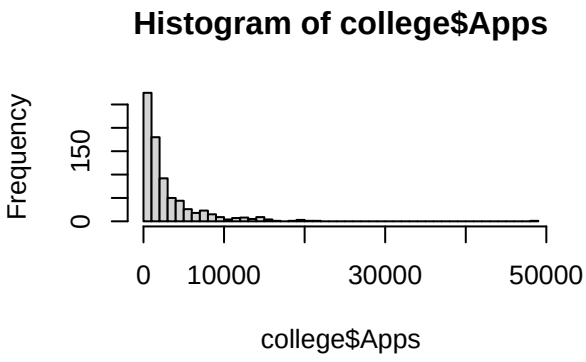
```
## starting httpd help server ... done
```

```
par(mfrow = c(2, 2))
hist(college$Apps, breaks = 5)
hist(college$Apps, breaks = 10)
hist(college$Apps, breaks = 20)
hist(college$Apps, breaks = 50)
```

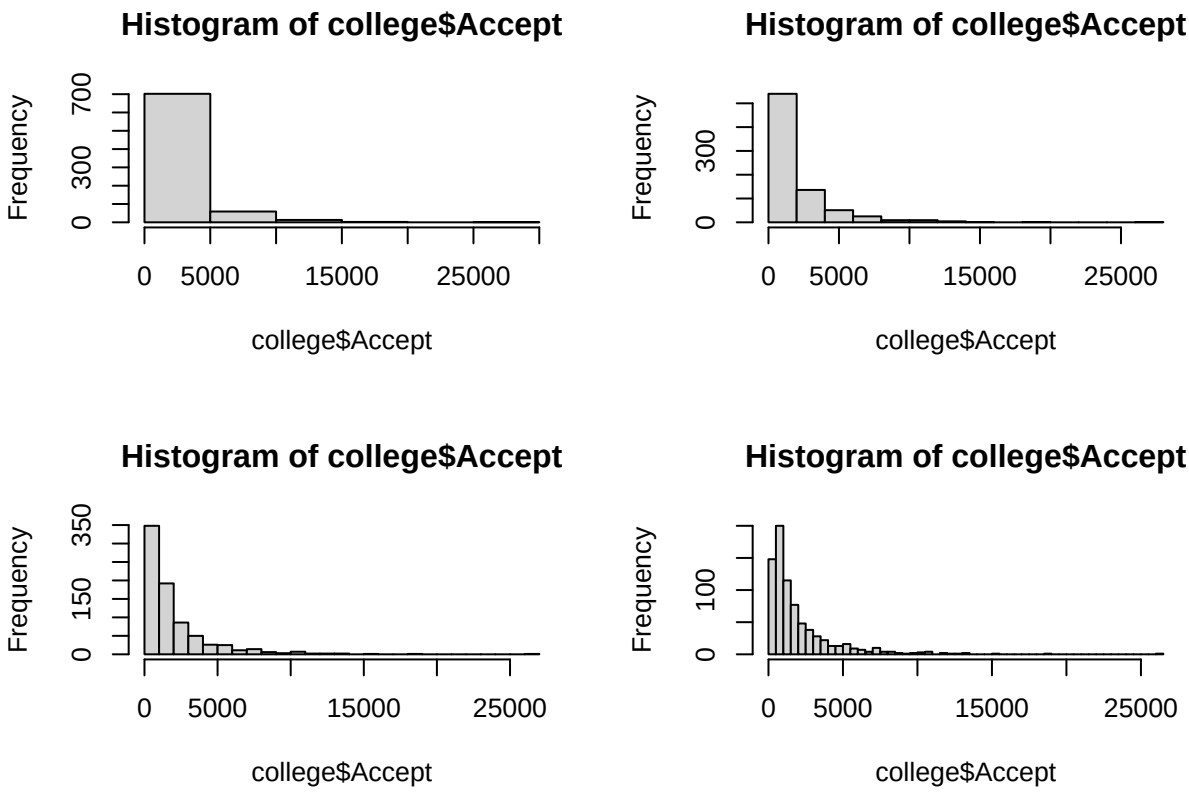




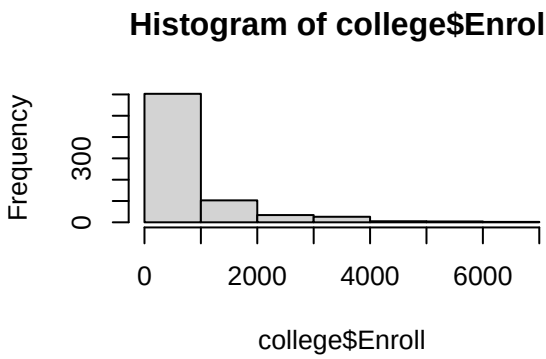


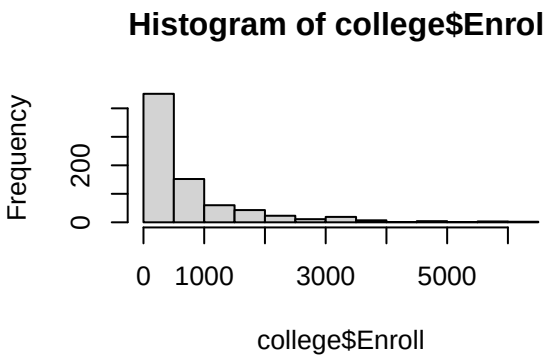


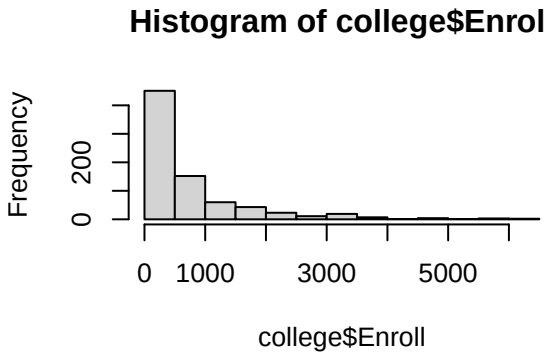

```
par(mfrow = c(2, 2))
hist(college$Accept, breaks = 5)
hist(college$Accept, breaks = 10)
hist(college$Accept, breaks = 20)
hist(college$Accept, breaks = 50)
```

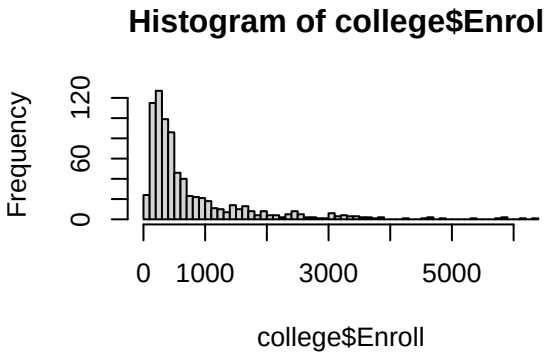



```
par(mfrow = c(2, 2))
hist(college$Enroll, breaks = 5)
hist(college$Enroll, breaks = 10)
hist(college$Enroll, breaks = 20)
hist(college$Enroll, breaks = 50)
```

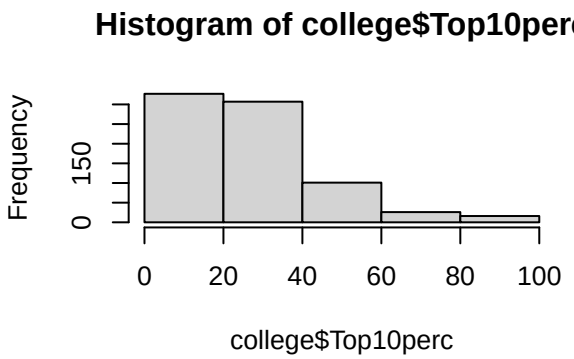


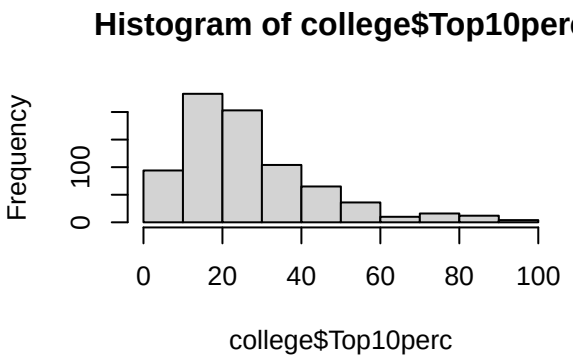


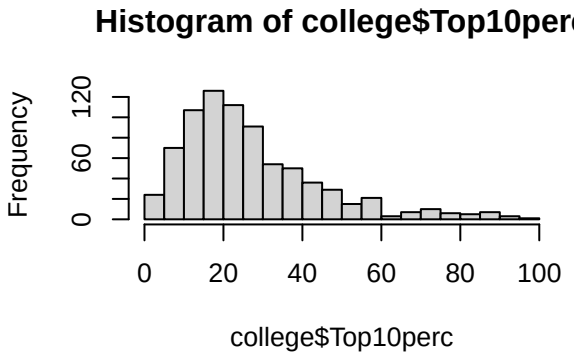


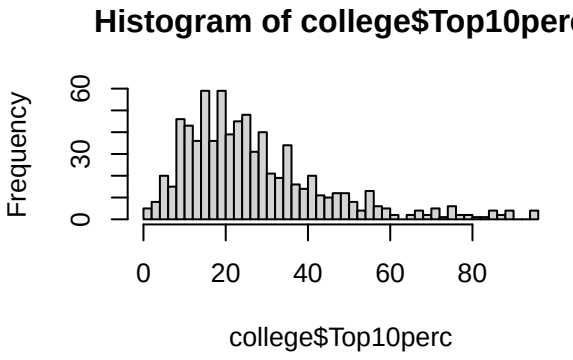



```
par(mfrow = c(2, 2))
hist(college$Top10perc, breaks = 5)
hist(college$Top10perc, breaks = 10)
hist(college$Top10perc, breaks = 20)
hist(college$Top10perc, breaks = 50)
```

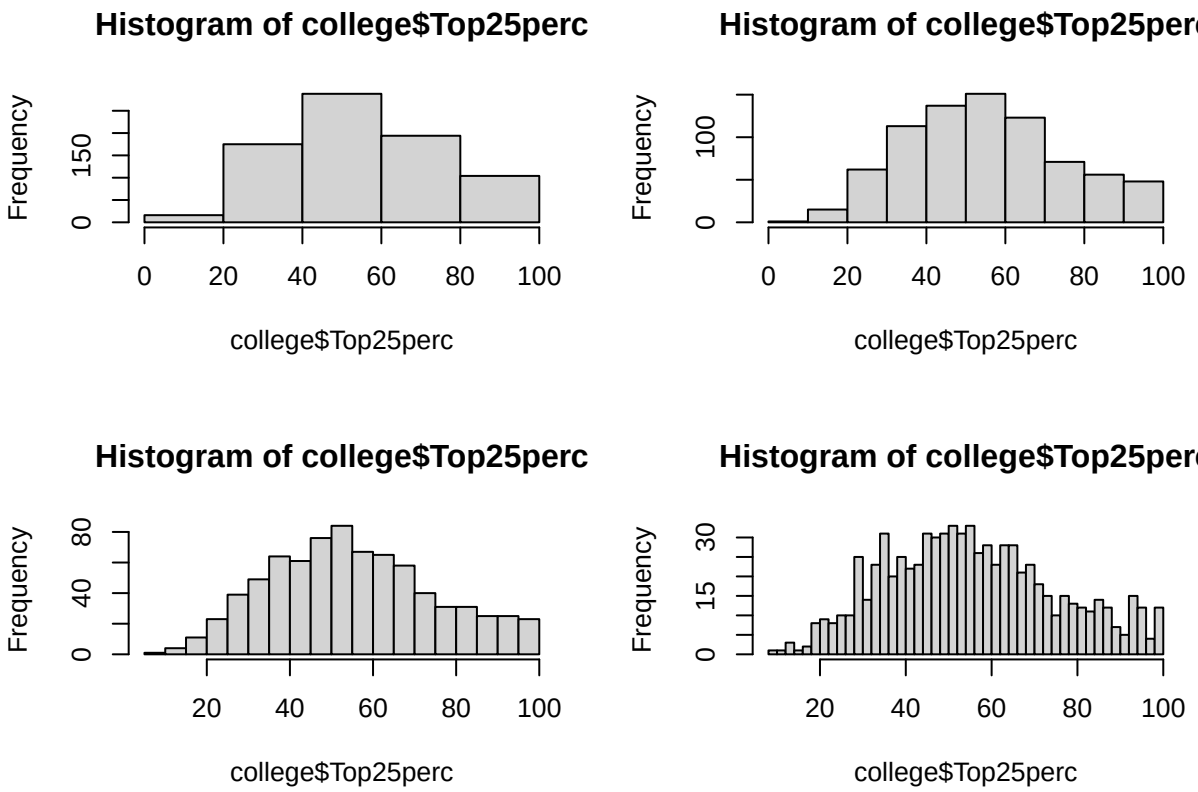




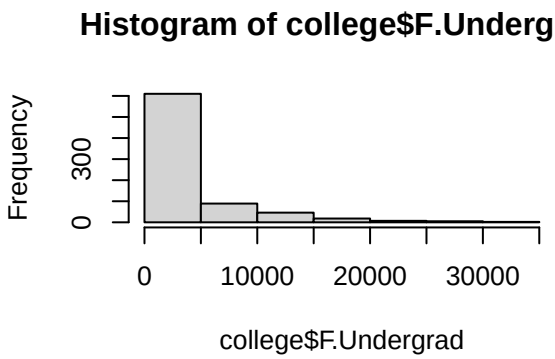


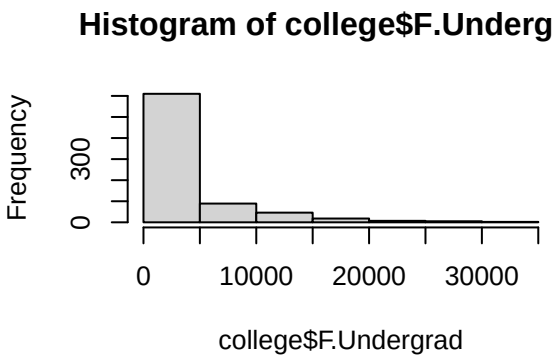


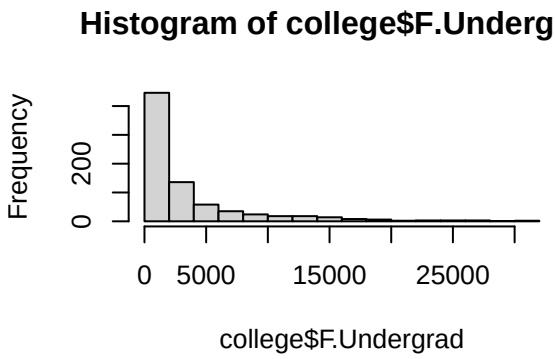

```
par(mfrow = c(2, 2))
hist(college$Top25perc, breaks = 5)
hist(college$Top25perc, breaks = 10)
hist(college$Top25perc, breaks = 20)
hist(college$Top25perc, breaks = 50)
```

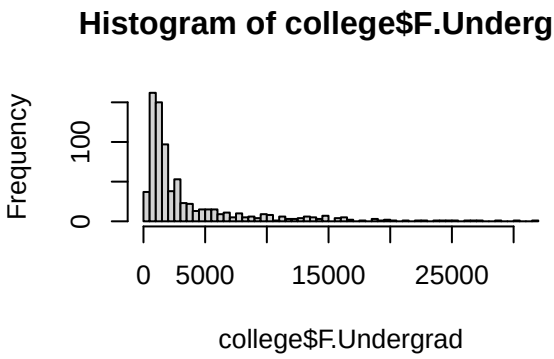



```
par(mfrow = c(2, 2))
hist(college$F.Undergrad, breaks = 5)
hist(college$F.Undergrad, breaks = 10)
hist(college$F.Undergrad, breaks = 20)
hist(college$F.Undergrad, breaks = 50)
```

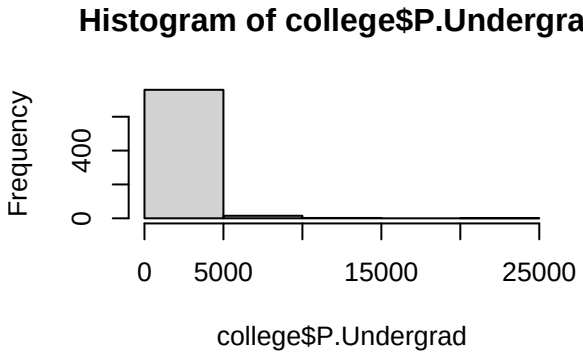


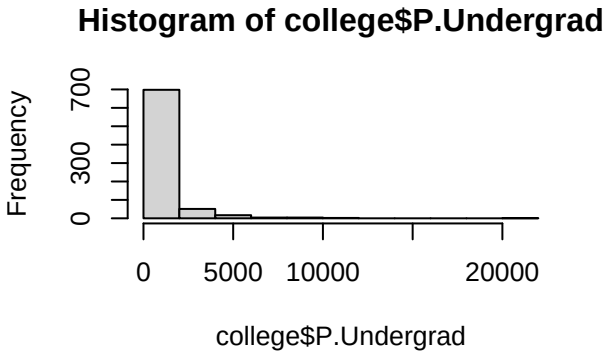


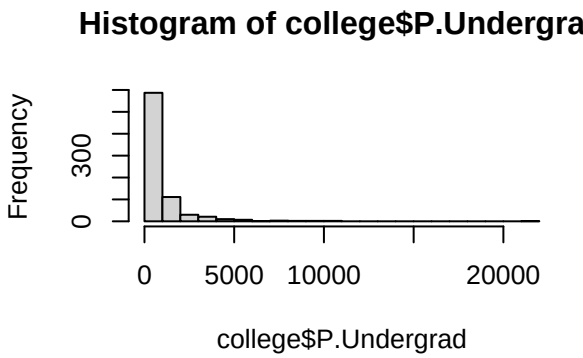


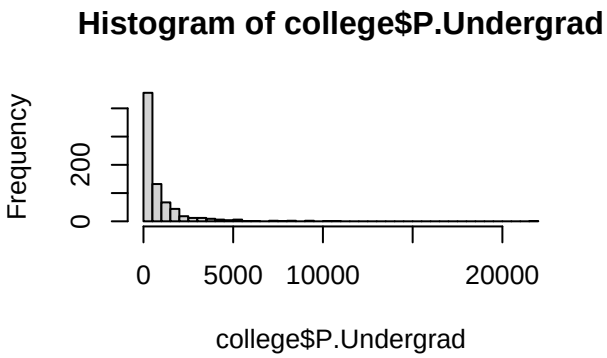



```
par(mfrow = c(2, 2))
hist(college$P.Undergrad, breaks = 5)
hist(college$P.Undergrad, breaks = 10)
hist(college$P.Undergrad, breaks = 20)
hist(college$P.Undergrad, breaks = 50)
```

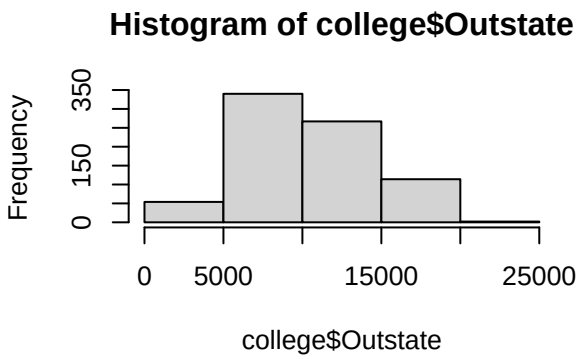


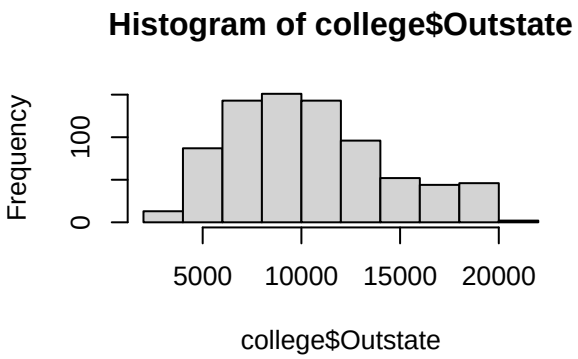


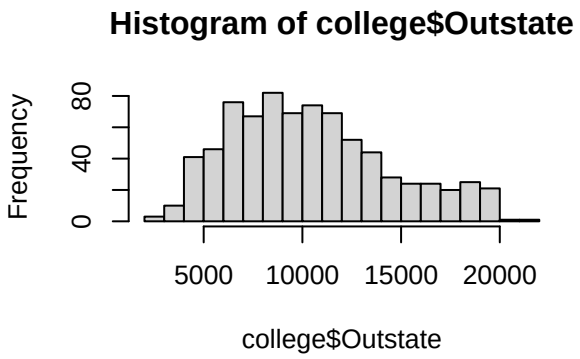


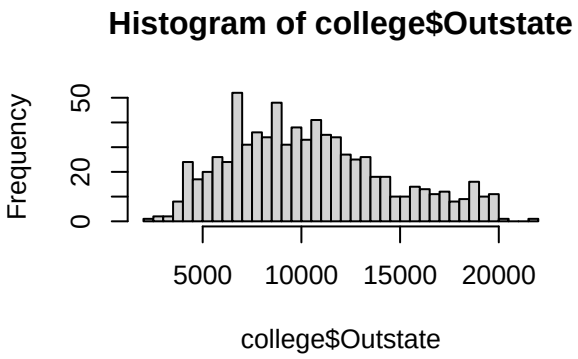



```
par(mfrow = c(2, 2))
hist(college$Outstate, breaks = 5)
hist(college$Outstate, breaks = 10)
hist(college$Outstate, breaks = 20)
hist(college$Outstate, breaks = 50)
```

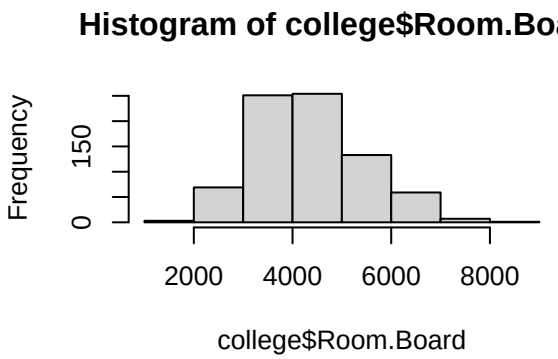


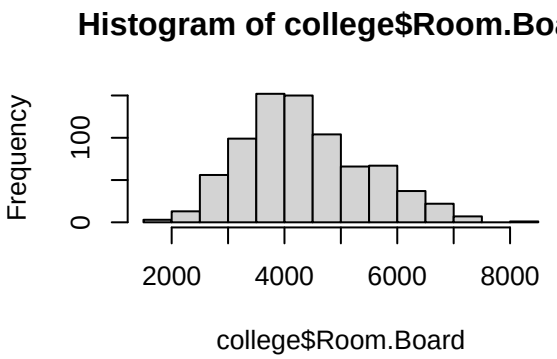


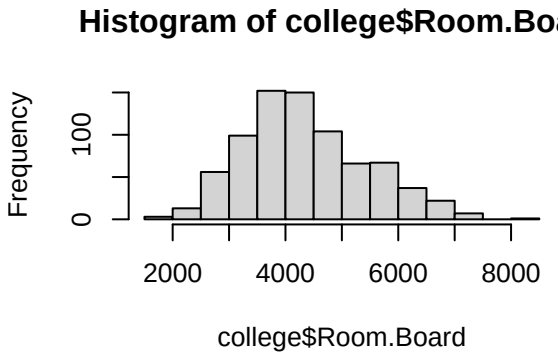


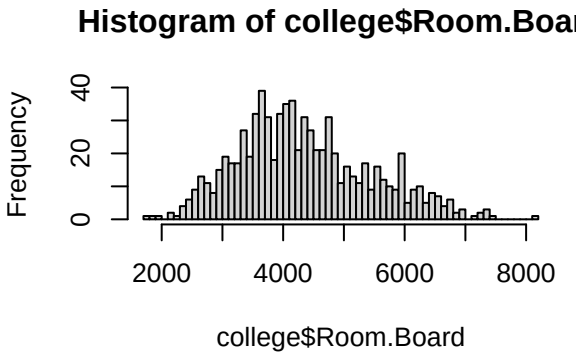



```
par(mfrow = c(2, 2))
hist(college$Room.Board, breaks = 5)
hist(college$Room.Board, breaks = 10)
hist(college$Room.Board, breaks = 20)
hist(college$Room.Board, breaks = 50)
```

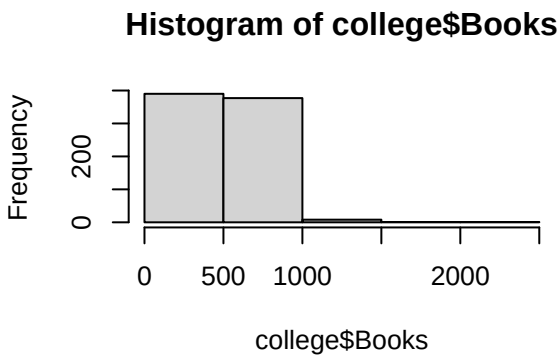


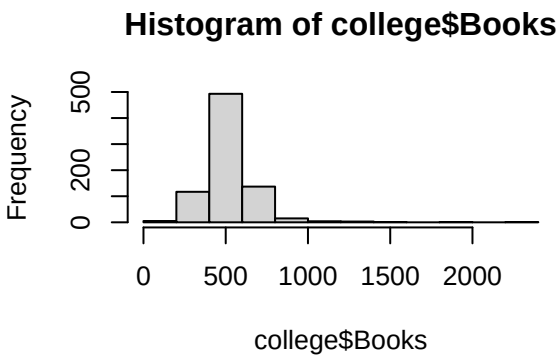


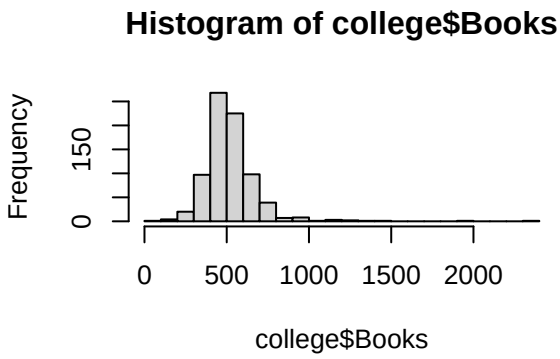


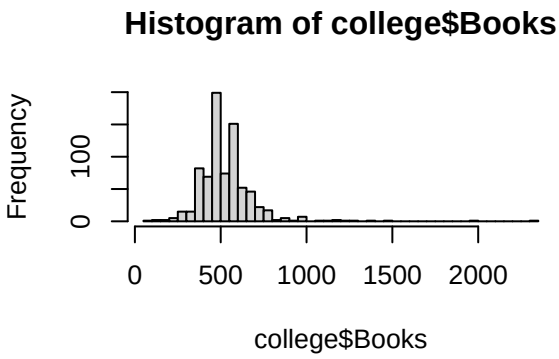



```
par(mfrow = c(2, 2))
hist(college$Books, breaks = 5)
hist(college$Books, breaks = 10)
hist(college$Books, breaks = 20)
hist(college$Books, breaks = 50)
```

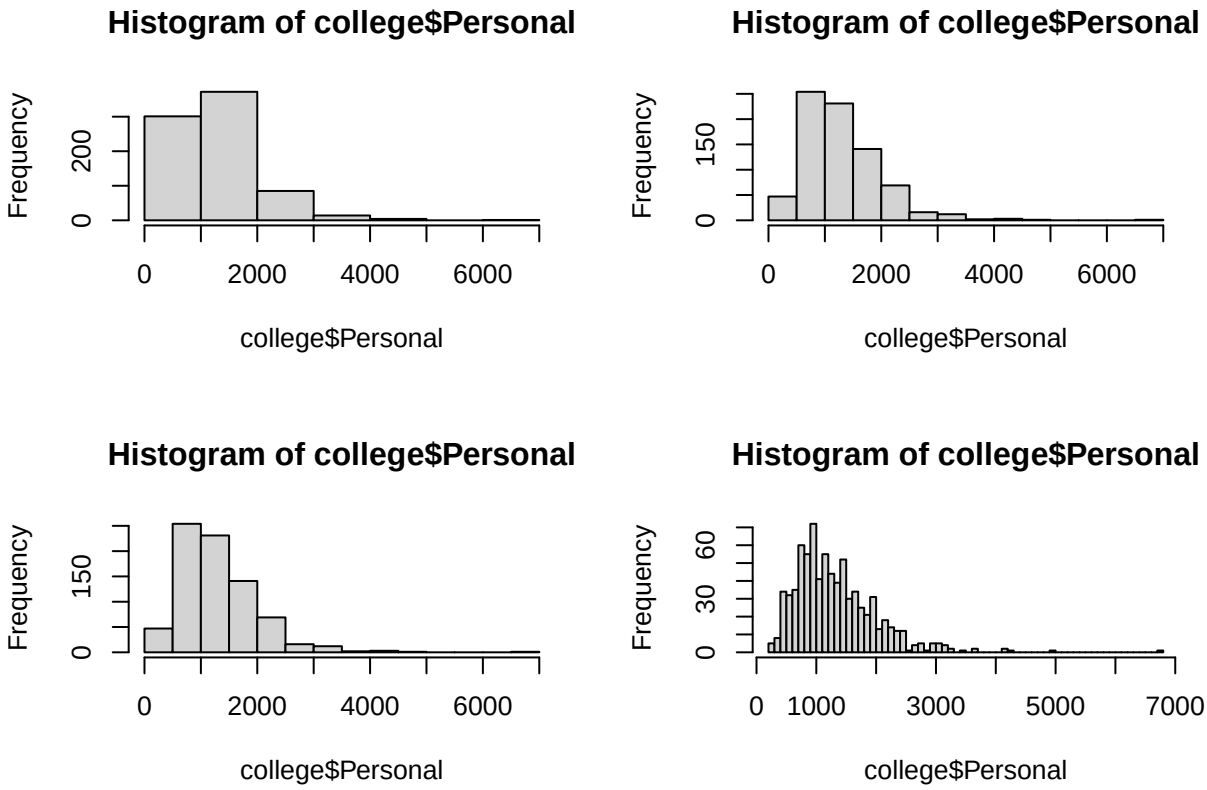




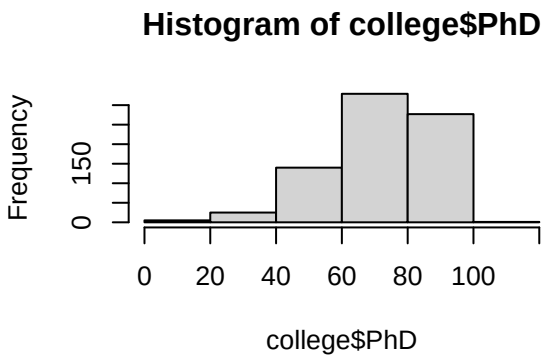


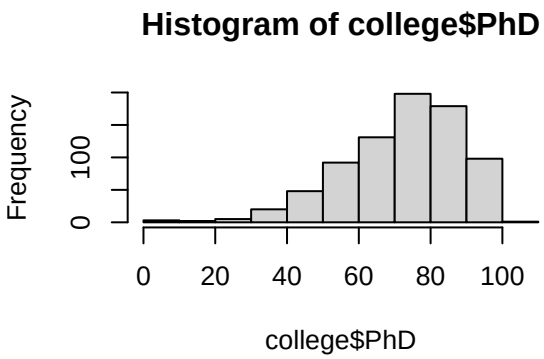


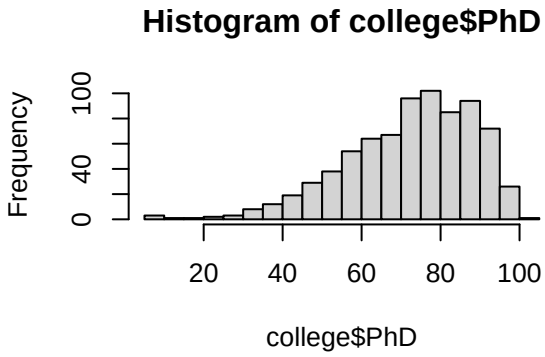

```
par(mfrow = c(2, 2))
hist(college$Personal, breaks = 5)
hist(college$Personal, breaks = 10)
hist(college$Personal, breaks = 20)
hist(college$Personal, breaks = 50)
```

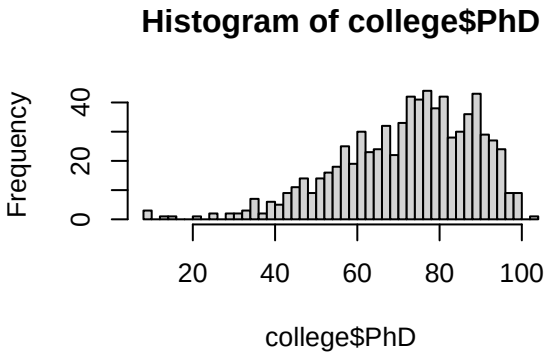



```
par(mfrow = c(2, 2))
hist(college$PhD, breaks = 5)
hist(college$PhD, breaks = 10)
hist(college$PhD, breaks = 20)
hist(college$PhD, breaks = 50)
```

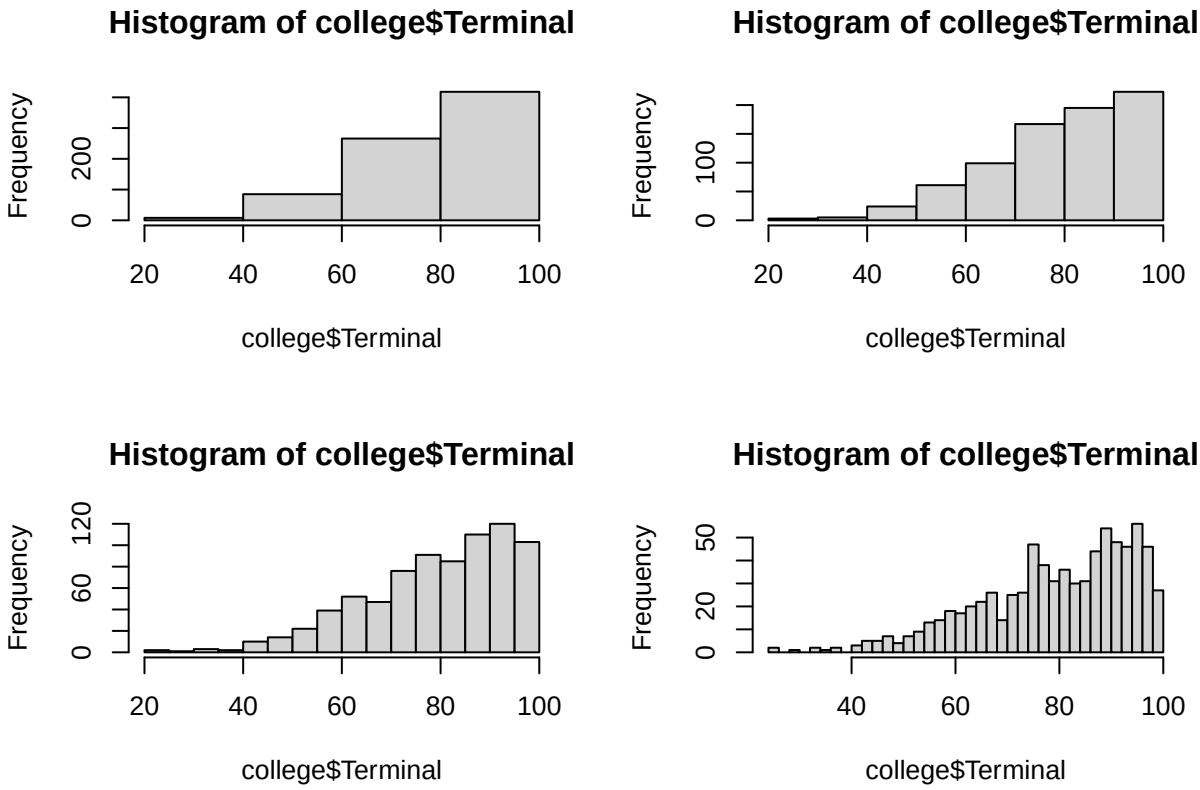




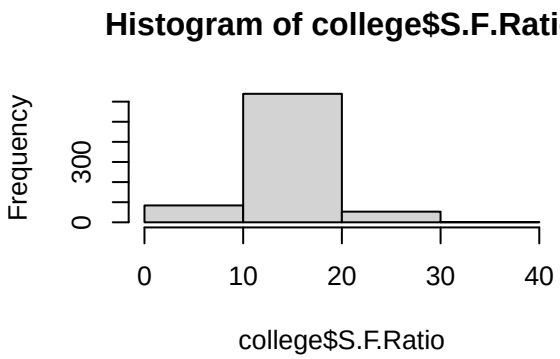


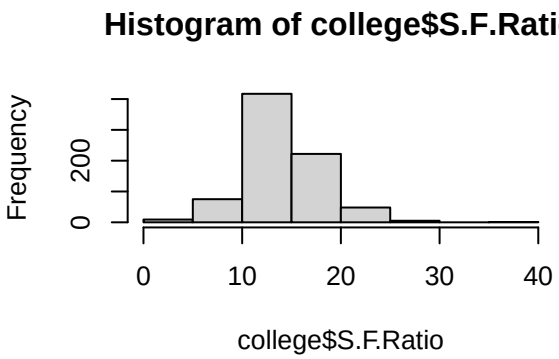


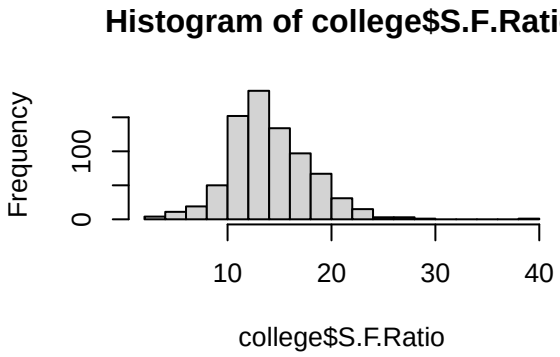

```
par(mfrow = c(2, 2))
hist(college$Terminal, breaks = 5)
hist(college$Terminal, breaks = 10)
hist(college$Terminal, breaks = 20)
hist(college$Terminal, breaks = 50)
```

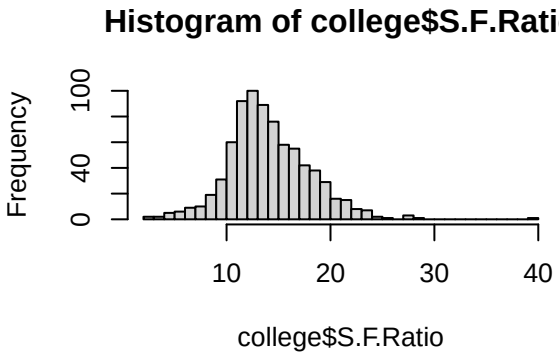



```
par(mfrow = c(2, 2))
hist(college$S.F.Ratio, breaks = 5)
hist(college$S.F.Ratio, breaks = 10)
hist(college$S.F.Ratio, breaks = 20)
hist(college$S.F.Ratio, breaks = 50)
```

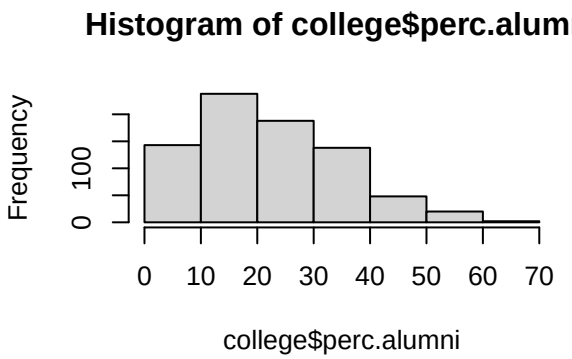


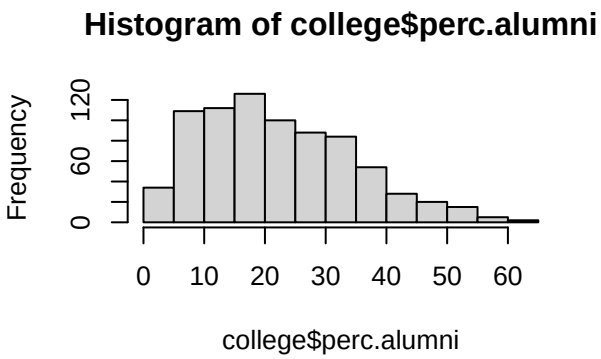


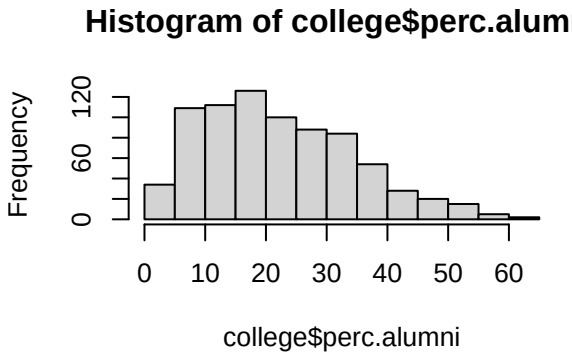


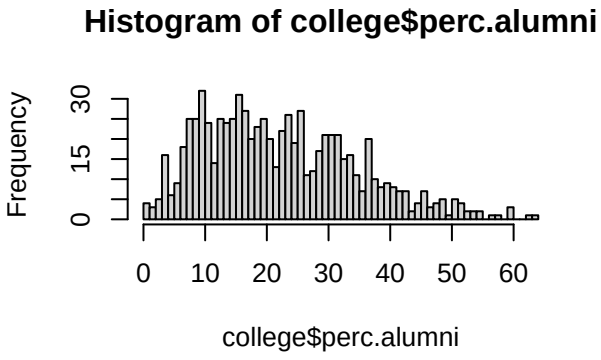



```
par(mfrow = c(2, 2))
hist(college$perc.alumni, breaks = 5)
hist(college$perc.alumni, breaks = 10)
hist(college$perc.alumni, breaks = 20)
hist(college$perc.alumni, breaks = 50)
```

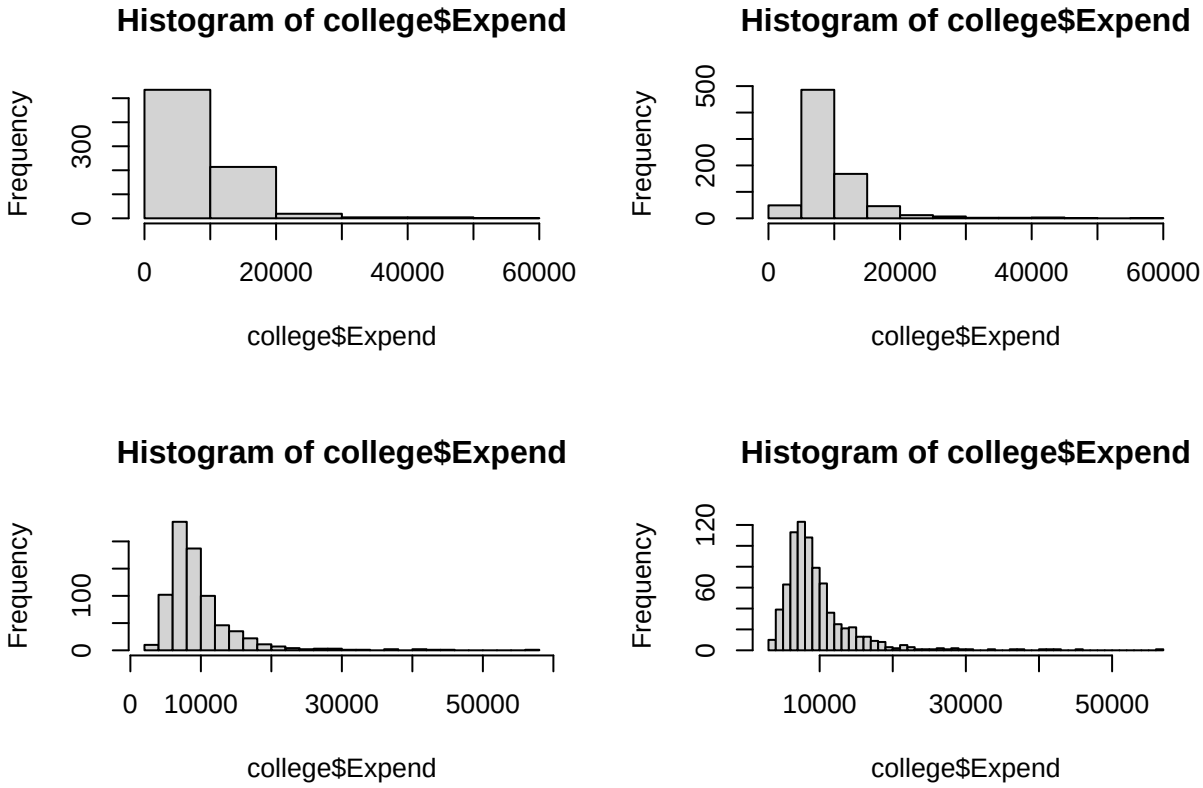




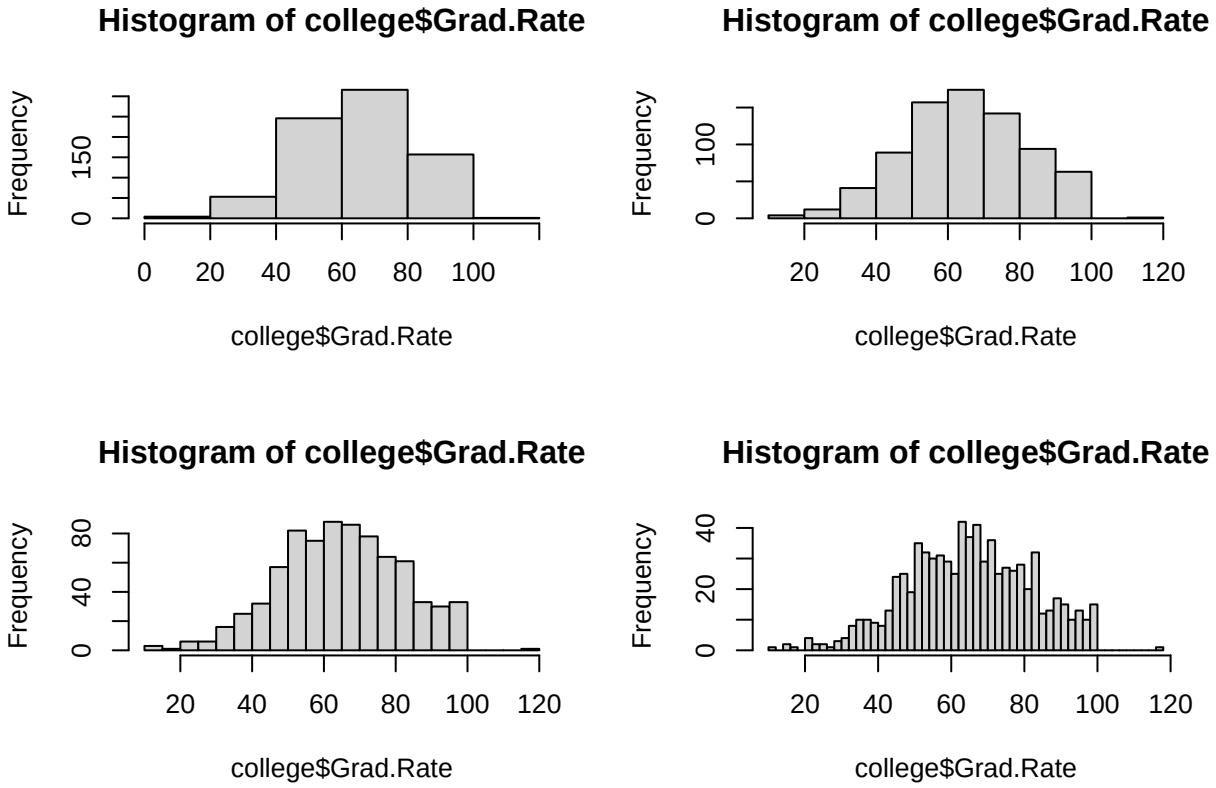





```
par(mfrow = c(2, 2))
hist(college$Expend, breaks = 5)
hist(college$Expend, breaks = 10)
hist(college$Expend, breaks = 20)
hist(college$Expend, breaks = 50)
```



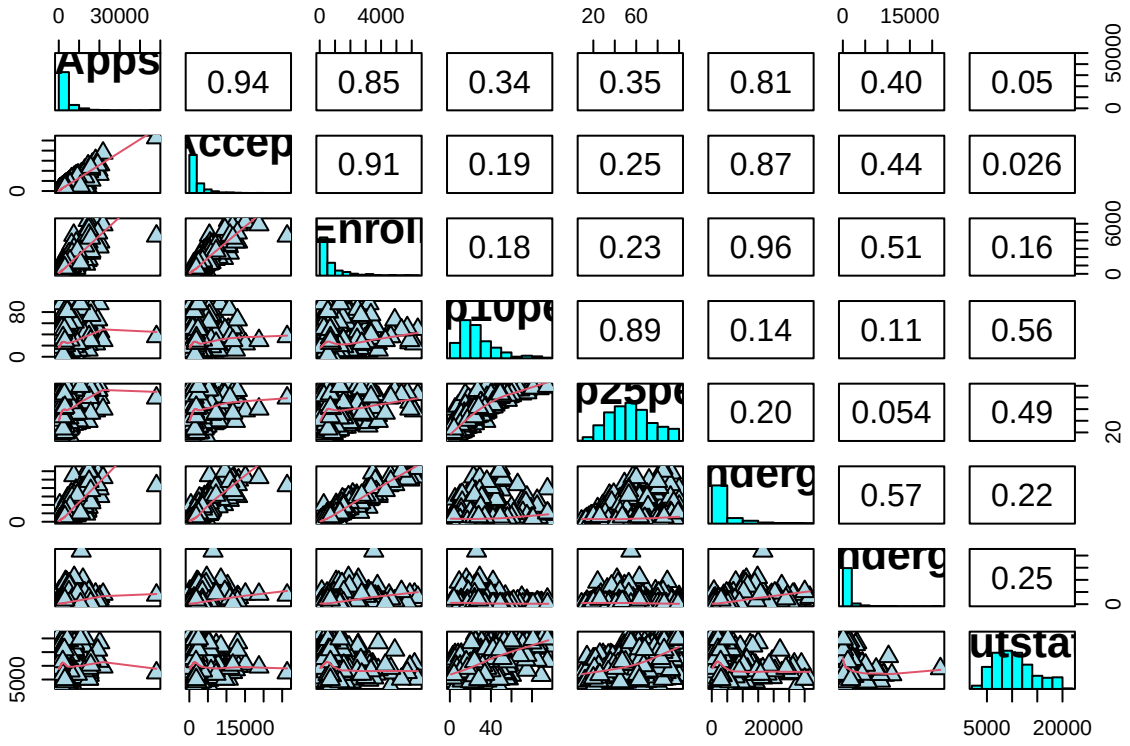

```
par(mfrow = c(2, 2))
hist(college$Grad.Rate, breaks = 5)
hist(college$Grad.Rate, breaks = 10)
hist(college$Grad.Rate, breaks = 20)
hist(college$Grad.Rate, breaks = 50)
```



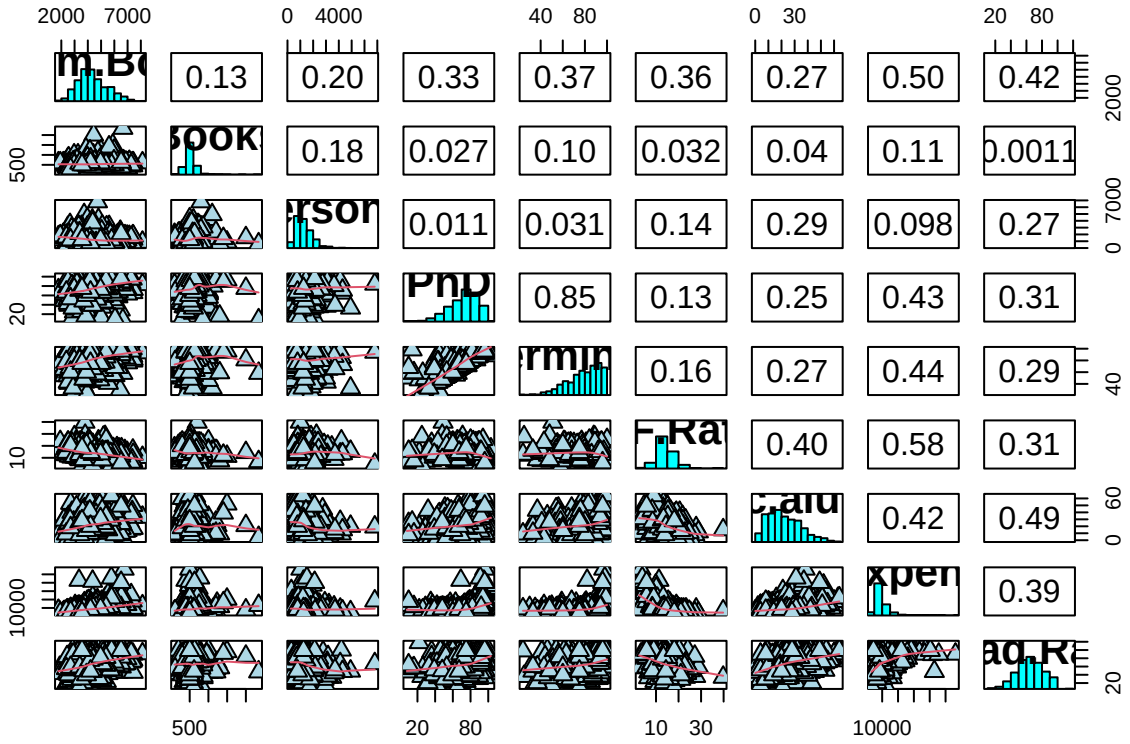
6. Continue exploring the data, and provide a brief summary of what you discover.

```
panel.hist <- function(x, ...)
{
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(usr[1:2], 0, 1.5) )
    h <- hist(x, plot = FALSE)
    breaks <- h$breaks; nB <- length(breaks)
    y <- h$counts; y <- y/max(y)
    rect(breaks[-nB], 0, breaks[-1], y, col = "cyan", ...)
}
panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...)
{
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    r <- abs(cor(x, y))
    txt <- format(c(r, 0.123456789), digits = digits)[1]
    txt <- paste0(prefix, txt)
    if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
    text(0.5, 0.5, txt, cex = cex.cor)
}

pairs(college[, 2:9],  upper.panel = panel.cor, panel = panel.smooth,
      cex = 1.5, pch = 24, bg = "light blue", horOdd=TRUE,
      diag.panel = panel.hist, cex.labels = 2, font.labels = 2)
```




```
pairs(college[, 10:18],  upper.panel = panel.cor, panel = panel.smooth,
      cex = 1.5, pch = 24, bg = "light blue", horOdd=TRUE,
      diag.panel = panel.hist, cex.labels = 2, font.labels = 2)
```




```
round(cor(college[2:18]), 3)
```

```
##                Apps Accept Enroll Top10perc Top25perc F.Undergrad P.Undergrad
## Apps          1.000  0.943  0.847     0.339     0.352        0.814        0.398
## Accept        0.943  1.000  0.912     0.192     0.247        0.874        0.441
## Enroll        0.847  0.912  1.000     0.181     0.227        0.965        0.513
## Top10perc     0.339  0.192  0.181     1.000     0.892        0.141       -0.105
## Top25perc     0.352  0.247  0.227     0.892     1.000        0.199       -0.054
## F.Undergrad   0.814  0.874  0.965     0.141     0.199        1.000        0.571
## P.Undergrad   0.398  0.441  0.513    -0.105    -0.054        0.571        1.000
## Outstate      0.050 -0.026 -0.155     0.562     0.489       -0.216       -0.254
## Room.Board    0.165  0.091 -0.040     0.371     0.331       -0.069       -0.061
## Books         0.133  0.114  0.113     0.119     0.116        0.116        0.081
## Personal      0.179  0.201  0.281    -0.093    -0.081        0.317        0.320
## PhD           0.391  0.356  0.331     0.532     0.546        0.318        0.149
## Terminal      0.369  0.338  0.308     0.491     0.525        0.300        0.142
## S.F.Ratio     0.096  0.176  0.237    -0.385    -0.295        0.280        0.233
## perc.alumni  -0.090 -0.160 -0.181     0.455     0.418       -0.229       -0.281
## Expend        0.260  0.125  0.064     0.661     0.527        0.019       -0.084
## Grad.Rate     0.147  0.067 -0.022     0.495     0.477       -0.079       -0.257
##               Outstate Room.Board  Books Personal    PhD Terminal S.F.Ratio
## Apps             0.050      0.165  0.133    0.179  0.391    0.369     0.096
## Accept          -0.026      0.091  0.114    0.201  0.356    0.338     0.176
## Enroll          -0.155     -0.040  0.113    0.281  0.331    0.308     0.237
## Top10perc        0.562      0.371  0.119   -0.093  0.532    0.491    -0.385
## Top25perc        0.489      0.331  0.116   -0.081  0.546    0.525    -0.295
```

```
## F.Undergrad   -0.216     -0.069  0.116    0.317  0.318    0.300     0.280
## P.Undergrad   -0.254     -0.061  0.081    0.320  0.149    0.142     0.233
## Outstate       1.000      0.654  0.039   -0.299  0.383    0.408    -0.555
## Room.Board     0.654      1.000  0.128   -0.199  0.329    0.375    -0.363
## Books          0.039      0.128  1.000    0.179  0.027    0.100    -0.032
## Personal      -0.299     -0.199  0.179    1.000 -0.011   -0.031     0.136
## PhD            0.383      0.329  0.027   -0.011  1.000    0.850    -0.131
## Terminal       0.408      0.375  0.100   -0.031  0.850    1.000    -0.160
## S.F.Ratio     -0.555     -0.363 -0.032    0.136 -0.131   -0.160     1.000
## perc.alumni    0.566      0.272 -0.040   -0.286  0.249    0.267    -0.403
## Expend         0.673      0.502  0.112   -0.098  0.433    0.439    -0.584
## Grad.Rate      0.571      0.425  0.001   -0.269  0.305    0.290    -0.307
##              perc.alumni Expend Grad.Rate
## Apps             -0.090  0.260     0.147
## Accept           -0.160  0.125     0.067
## Enroll           -0.181  0.064    -0.022
## Top10perc         0.455  0.661     0.495
## Top25perc         0.418  0.527     0.477
## F.Undergrad      -0.229  0.019    -0.079
## P.Undergrad      -0.281 -0.084    -0.257
## Outstate          0.566  0.673     0.571
## Room.Board        0.272  0.502     0.425
## Books            -0.040  0.112     0.001
## Personal         -0.286 -0.098    -0.269
## PhD               0.249  0.433     0.305
## Terminal          0.267  0.439     0.290
## S.F.Ratio        -0.403 -0.584    -0.307
## perc.alumni       1.000  0.418     0.491
## Expend            0.418  1.000     0.390
## Grad.Rate         0.491  0.390     1.000
```

problem16. (James et al. Chapter 2, problem 10.) This exercise involves the Boston housing data set. 1. To begin, load in the Boston data set. The Boston data set is part of the MASS library in R.

```
library(MASS)
data(Boston)
?Boston
```

There 506 rows and 14 columns. Each column is

- crim : per capita crime rate by town.
- zn : proportion of residential land zoned for lots over 25,000 sq.ft.
- indus : proportion of non-retail business acres per town.
- chas : Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).
- nox : nitrogen oxides concentration (parts per 10 million).
- rm : average number of rooms per dwelling.
- age : proportion of owner-occupied units built prior to 1940.
- dis : weighted mean of distances to five Boston employment centres.
- rad : index of accessibility to radial highways.
- tax : full-value property-tax rate per $10,000.
- ptratio : pupil-teacher ratio by town.
- black : 1000(Bk - 0.63)ˆ2 where Bk is the proportion of blacks by town.
- lstat : lower status of the population (percent).
- medv : median value of owner-occupied homes in $1000s.

2. Make some pairwise scatterplots of the predictors (columns) in this data set. Describe your findings.

3. Are any of the predictors associated with per capita crime rate? If so, explain the relationship.
4. Do any of the suburbs of Boston appear to have particularly high crime rates? Tax rates? Pupil-teacher ratios? Comment on the range of each predictor.
5. How many of the suburbs in this data set bound the Charles river?
6. What is the median pupil-teacher ratio among the towns in this data set?
7. Which suburb of Boston has lowest median value of owner occupied homes? What are the values of the other predictors for that suburb, and how do those values compare to the overall ranges for those predictors? Comment on your findings.
8. In this data set, how many of the suburbs average more than seven rooms per dwelling? More than eight rooms per dwelling? Comment on the suburbs that average more than eight rooms per dwelling.