2.0 Multivariate Visualization Assignment

JD Santos

Problem 1

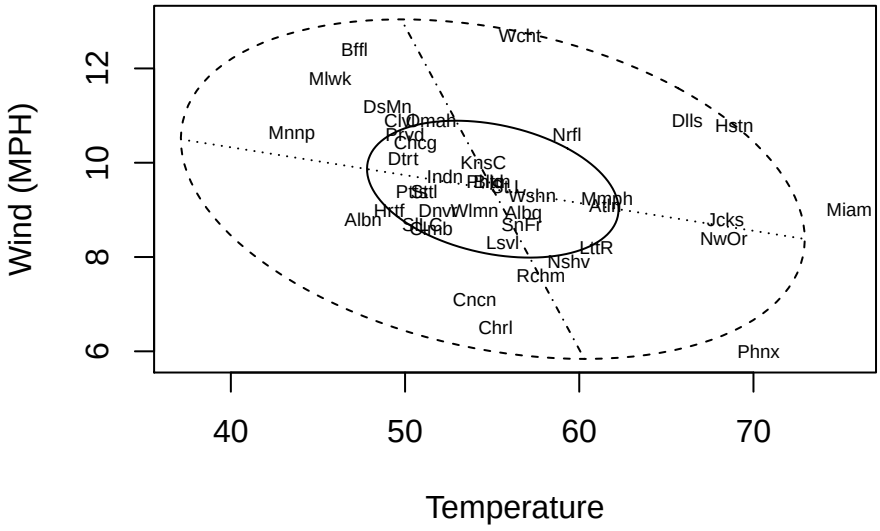
> Use the bivariate boxplot on the scatterplot of pairs of variables ((temp, wind), (temp, precip)) in the air pollution data to identify any outliers. Calculate the correlation between each pair of variables using all the data and the data with any identified outliers removed. Comment on the results.

```
data("USairpollution", package = "HSAUR2")
temp_wind <- USairpollution[, c("temp", "wind")]
temp_precip <- USairpollution[, c("temp", "precip")]
```

1. Begin by pulling the variable pairs from USairpollution into objects

```
bvbox(temp_wind, xlab = "Temperature", ylab = "Wind (MPH)", type = "n")

text(USairpollution$temp, USairpollution$wind,
     cex = 0.6,
     labels = abbreviate(row.names(USairpollution)))
```

2. Create the bivariate boxplot for temp/wind pair



```
outcity <- match(c("Miami", "Phoenix"),
                 rownames(USairpollution))
```

3. Phoenix and Miami are identified as outliers, we will perform a match to obtain their positions and store them in `outcity`

4. Calculate correlations with and without these outliers.

- The correlation drops significantly from -0.35 to -0.26 with the outliers removed. This makes sense if you examine the visualization and notice that these outliers are generally in line with the regression line. Removing them reduces the fit of this line.

```
cor(temp_wind$temp, temp_wind$wind)
```

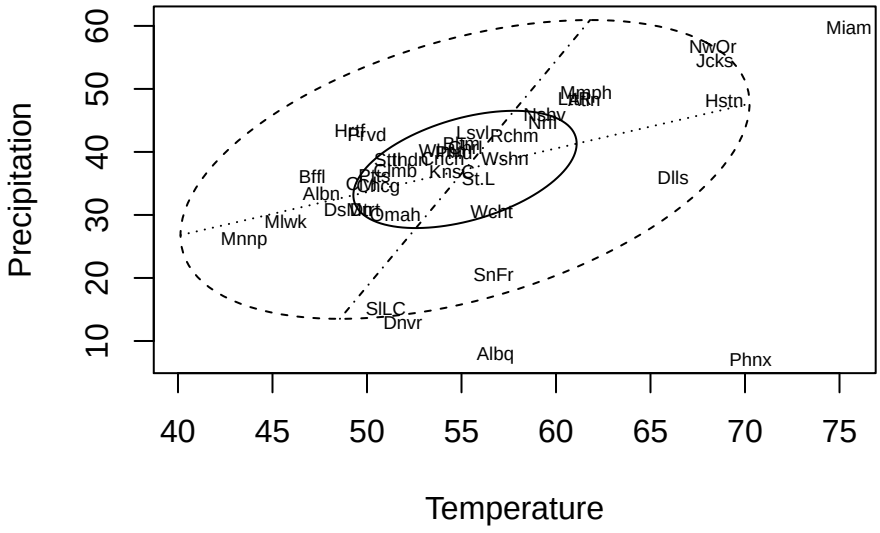
```
## [1] -0.3497396
```

```
cor(temp_wind$temp[-outcity], temp_wind$wind[-outcity])
```

```
## [1] -0.2587808
```

```
bvbox(temp_precip, xlab = "Temperature", ylab = "Precipitation", type = "n")

text(USairpollution$temp, USairpollution$precip, cex = 0.6,
     labels = abbreviate(row.names(USairpollution)))
```

5. Create the bivariate boxplot for temp/precip pair




```
outcity2 <- match(c("Miami", "Phoenix", "Albuquerque"),
                 rownames(USairpollution))
```

6. This time we identify Albuquerque as well as Miami and Phoenix as outliers. Perform the same operation as above to store this in an object

7. Calculate correlations with and without these outliers.

- This time we have the opposite effect; the correlation increases from 0.39 to 0.62. Looking at the bivariate boxplot, we can see that at least two of the outliers are extremely 'far' from the regression lines.

```
cor(temp_precip$temp, temp_precip$precip)
```

```
## [1] 0.3862534
```

```
cor(temp_precip$temp[-outcity2], temp_precip$precip[-outcity2])
```

```
## [1] 0.6227856
```

Problem 2

The banknote dataset contains measurements on 200 Swiss banknotes: 100 genuine and 100 counterfeits. The variables are the status of the "note," length of the bill, width of the left edge, width of the right edge, bottom margin width, and top margin width. All measurements are in millimeters. Read the data and pick the variables: "note," "top_margin," and "diag_length."

```
banknote <- read.csv("http://westfall.ba.ttu.edu/isqs6348/Rdata/swiss.csv")
notedata <- banknote[,c(1,6,7)]
head(notedata)
```

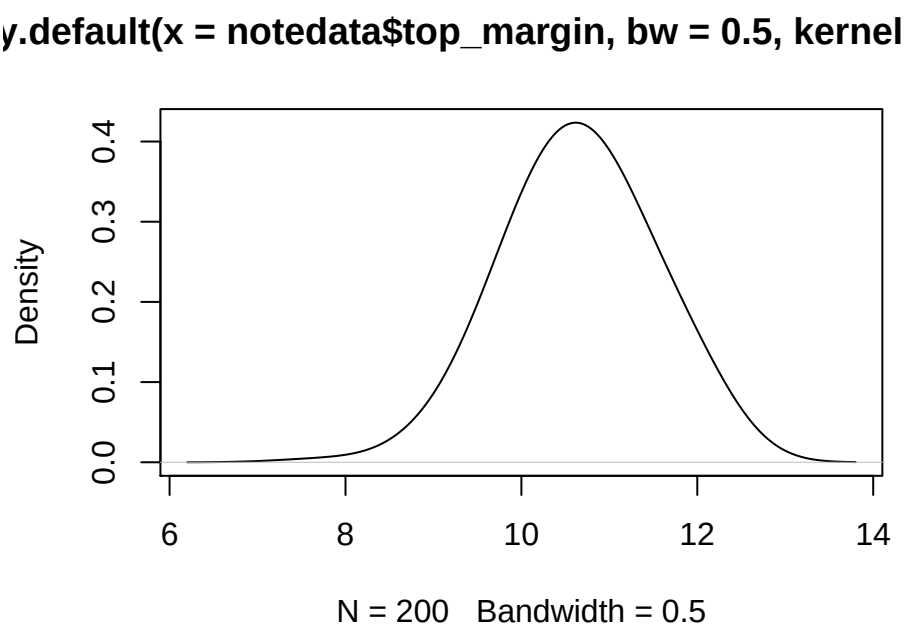
```
##   note top_margin diag_length
## 1 real        9.7       141.0
## 2 real        9.5       141.7
## 3 real        9.6       142.2
## 4 real       10.4       142.0
## 5 real        7.7       141.8
## 6 real       10.1       141.4
```

A

a) Construct separate univariate kernel estimates (Gaussian kernel) of the distributions of these two variables.

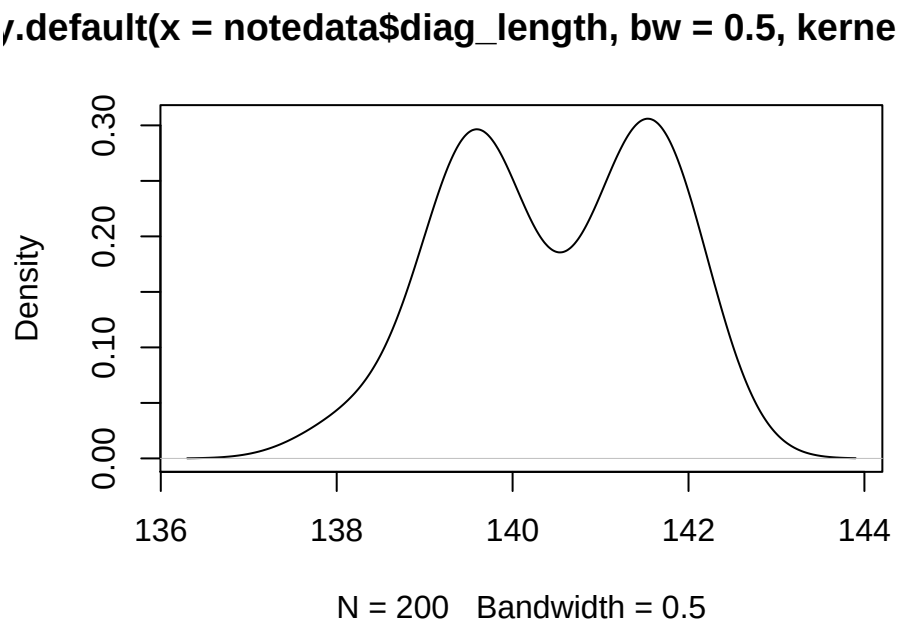
```
plot(density(notedata$top_margin, bw = 0.5, kernel = "gaussian"))
```

1. First, we find the kernel density estimator of top_margin with the Gaussian kernel:




```
plot(density(notedata$diag_length, bw = 0.5, kernel = "gaussian"))
```

2. Next, the same Gaussian kernel density estimator for the diagonal length:



B

b) Using the bivariate Gaussian kernel, estimate the two variables' bivariate density using (i) a contour plot and (ii) a 3-D perspective plot.

```
bw <- c(dpik(notedata$top_margin), dpik(notedata$diag_length))
```

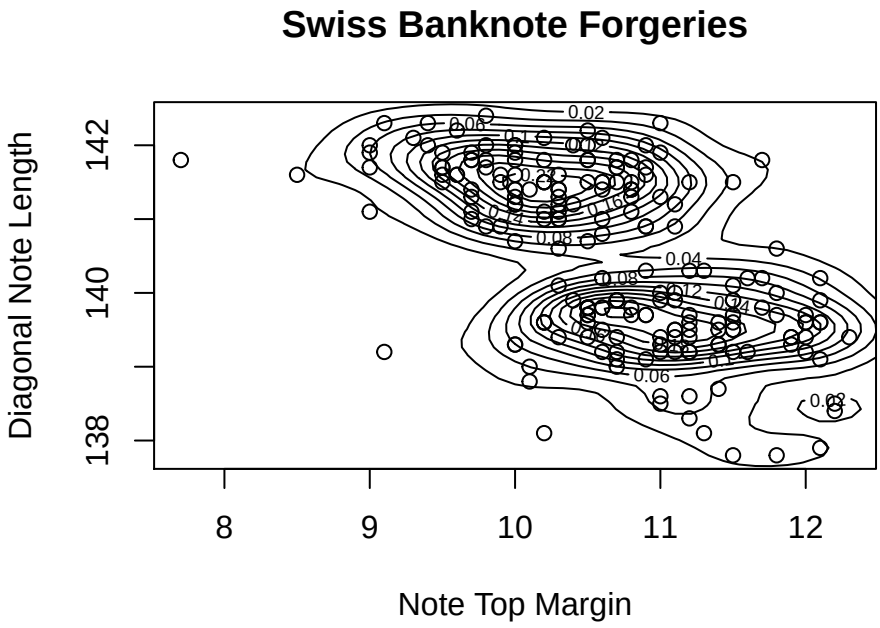
1. Determine appropriate bandwidth for both variables with `dpik()`

```
notedimensions <- notedata[,c(2,3)]
density <- bkde2D(notedimensions, bandwidth = bw)
```

2. Save the density for contouring

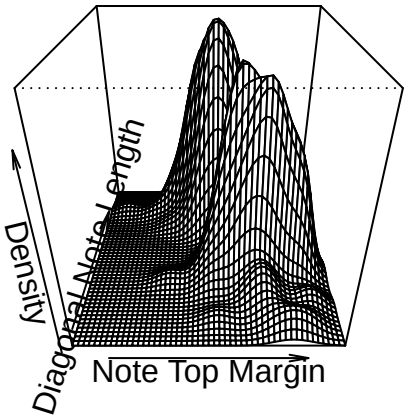
```
plot(notedata$top_margin, notedata$diag_length,
     xlab = "Note Top Margin",
     ylab = "Diagonal Note Length",
     main="Swiss Banknote Forgeries")

# Add countours
contour(x = density$x1, y = density$x2, z = density$fhat, add = TRUE)
```

3. Create the scatter plot with the (2D) density contours




```
persp(x = density$x1, y = density$x2,
      z = density$fhat,
      xlab = "Note Top Margin",
      ylab = "Diagonal Note Length",
      zlab = "Density",
      main="Swiss Banknote Forgeries", phi = 30)
```

4. Create the scatterplot with 3D density perspective

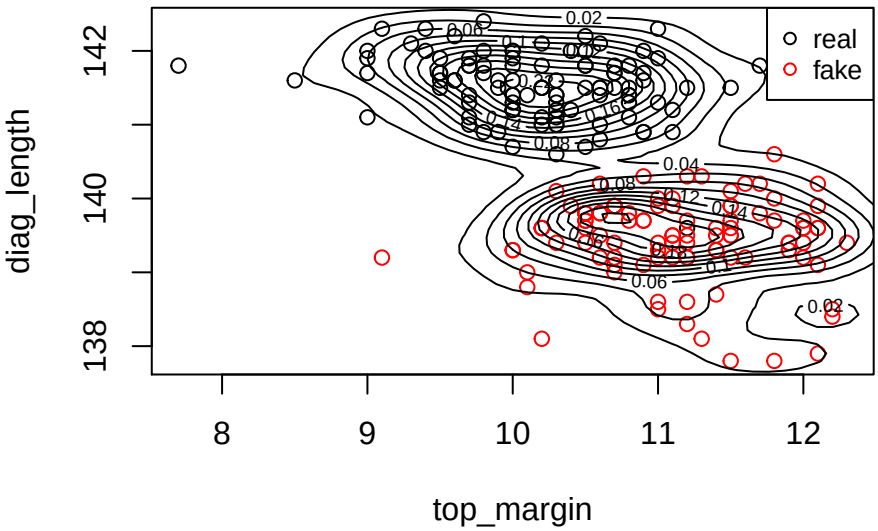


C

c) Plot the scatterplot, highlighting points with different colors according to whether the bills are real or fake (the "note" variable in the data set has that information). Explain your findings.

We create a simple scatterplot with color codings for real notes = black and fake notes = red. - By doing this we can see that this dataset is not only extremely bimodal, but that these correspond to real and fake banknotes - The fake banknotes appear to have larger top margins, but less diagonal length than the real banknotes

```
# Create plot with color coding
plot(notedata[,2:3], col = ifelse(notedata[,1] == "real", "black", "red"))

# Create legend
legend("topright",
       legend = c("real", "fake"),
       col = c("black", "red"),
       pch = 1,
       cex = .8)

# Add countours
contour(x = density$x1, y = density$x2, z = density$fhat, add = TRUE)
```



Problem 3

Examine the multivariate normality (MVN) of the banknote data (excluding the "note" variable) by creating the chi-square plot of the data. Load the data as follow. Follow the listed steps to examine the multivariate normality.

```
notedata2 <- banknote[,-1]
head(notedata2)
```

```
##   length left_width right_width bottom_margin top_margin diag_length
## 1  214.8      131.0       131.1           9.0        9.7       141.0
## 2  214.6      129.7       129.7           8.1        9.5       141.7
## 3  214.8      129.7       129.7           8.7        9.6       142.2
## 4  214.8      129.7       129.6           7.5       10.4       142.0
## 5  215.0      129.6       129.7          10.4        7.7       141.8
## 6  215.7      130.8       130.5           9.0       10.1       141.4
```

A) Find the column-means vector.

Here we find the column means and store them in Xbar

```
xbar <- colMeans(notedata2)
xbar
```

```
##        length    left_width   right_width bottom_margin    top_margin
##      214.8960      130.1215      129.9565        9.4175       10.6505
##   diag_length
##      140.4835
```

B) Find the covariance matrix of the data.

** Save the covariance of our Swiss notes data in S:**

```
S <- cov(notedata2)
S
```

```
##                    length  left_width right_width bottom_margin top_margin
## length          0.14179296  0.03144322  0.02309146    -0.1032462 -0.0185407
## left_width      0.03144322  0.13033945  0.10842739     0.2158028  0.1050394
## right_width     0.02309146  0.10842739  0.16327412     0.2841319  0.1299967
## bottom_margin -0.10324623  0.21580276  0.28413191     2.0868781  0.1645389
## top_margin    -0.01854070  0.10503945  0.12999673     0.1645389  0.6447234
## diag_length    0.08430553 -0.20934196 -0.24047010    -1.0369962 -0.5496148
##                diag_length
## length          0.08430553
## left_width     -0.20934196
## right_width    -0.24047010
## bottom_margin -1.03699623
## top_margin     -0.54961482
## diag_length     1.32771633
```

C) Find the Mahalanobis distances given using the data and the result of parts a and b.

Using the `mahalanobis()` function, we plug in the data with the column means and covariance to find the Mahalanobis distances:

```
d2 <- mahalanobis(notedata2, xbar, S)
d2
```

```
##   [1] 24.1822330  4.4554815  3.3862349  2.8429463 17.4722855  9.3219107
##   [7]  9.8042238  5.7728142  8.5299136  3.1996948  9.6407351  5.4062400
##  [13] 17.4950978  3.7504885  2.6343394 10.3456632  6.2974792  4.2870447
##  [19]  7.7004403  4.0536247  3.6330789  5.9255552  6.4372035  5.1073406
##  [25]  5.1089894  9.6852215  6.1461050  6.0985269  3.3328174  5.2250235
##  [31]  2.6551357  4.9356472  2.7281070  4.3450616  8.1854385  6.3338524
##  [37]  4.5880792  3.9323394  6.0046395 28.4939129  8.3845177  2.5361408
##  [43]  5.3738641  5.3809814  6.0594238  5.2350319  1.0980743  4.2016621
##  [49]  3.7321367 14.9317002  4.9166117  6.1811642  4.3628039  4.7271556
##  [55]  5.4578142  4.9743699 10.1261952  7.4588354  1.9647956  0.9801036
##  [61]  3.7165107  4.3121951  5.0359862  4.7084987  1.1686711  4.2677333
##  [67]  4.3789534  3.7423529  4.4801803  5.2534522 11.4668468  2.3030182
##  [73]  9.9720370  5.4007099  3.3740429  8.5119132  3.1278829  3.9106167
##  [79]  5.6256017  3.9169904  3.4100169  2.7745593  7.6604553  1.7542228
##  [85]  6.9110882  1.6180346  2.7126363  5.1216222  5.3262090  2.2888119
##  [91]  3.3487717  3.5331891  4.8545708  4.9628844  2.8134114  6.3416884
##  [97]  2.8217861  3.5043815  1.1821895  4.4416822  4.8431122  2.9901218
## [103]  2.4448228  3.3937963  3.6264349  1.1502206  5.9386644  3.6549450
## [109]  4.1826648  5.9051392  7.2361317  3.5312284 13.6721778  5.2237190
## [115]  1.8236887  8.9013403  5.9074713  4.9394296  2.7285680  5.1370863
## [121]  4.3177627  6.1828397 11.2545134  2.8181005  4.6186889  3.8217062
## [127]  2.6596084  7.3967043  2.1095077  8.9522262  3.3024746  8.0516287
## [133]  3.0533084  4.4710959  5.3391935  4.5188320  5.8781588  9.2269819
## [139]  2.9444483  5.3028022  3.1001636  7.0968556  3.3780129  1.6926028
## [145]  6.7357690  4.2728377  2.7194444 10.9317315  1.1955684  7.4720971
## [151]  5.2551247  8.0500047  8.4150356  4.8141217  3.8406819  6.3366302
## [157]  6.5304754  3.3376580  7.7133550 15.9965806 21.4697051 14.5249697
## [163]  2.7899519  3.4533610  4.7555441  6.1859662 25.1679604 10.1836802
## [169]  5.1673717  3.9762942 25.6509644  9.4815329  3.3033659  9.8294635
## [175]  2.6094774  3.6080347  4.3702953  5.5375330  3.5997698 18.7619304
## [181]  3.2896745 13.7484228  3.3646008  2.7510298  1.1711986  3.4171748
```

```
## [187] 11.6212331  5.3446211  5.7222961 12.6440935  6.7329757 13.0130250
## [193]  1.4372402  6.5388802  3.8545002  3.7402127  3.6216497  1.9835372
## [199]  5.4698103  4.1498014
```

D) Sort the Mahalanobis distances from smallest to largest.

Sort the distances to see which are the closest and furthest from the mean. Save into sorted_d2 for future use.

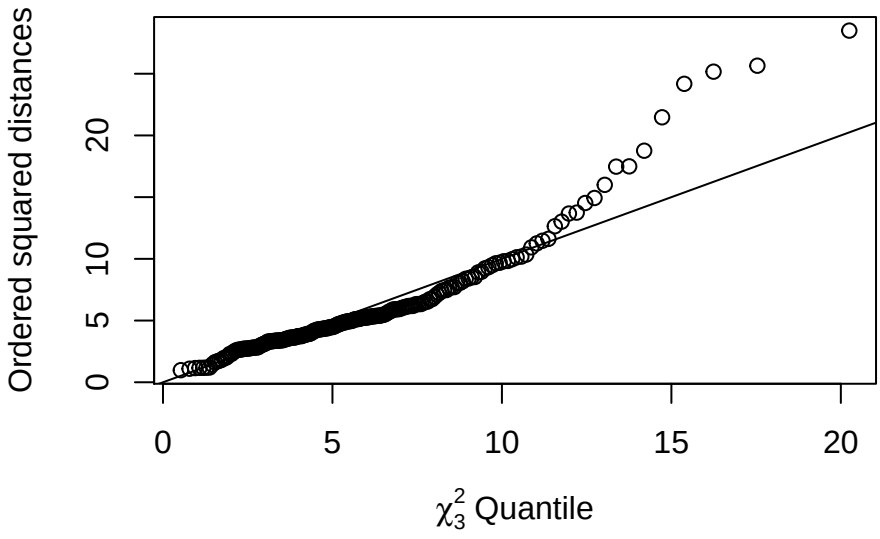
```
sorted_d2 <- sort(d2, decreasing = FALSE)
```

E) Chi-Square Distribution

> Suppose your data is multivariate normal, by definition. In that case, the Mahalanobis distances should follow the chi-sq distribution (with df = number of columns), so create a chi-square quantile values list and plot them vs. the sorted Mahalanobis distances. The quantiles should be on the x-axis, and the sorted Mahalonobis distances must be on the y-axis. Please label your chart correctly. You also need to add a 45-degree line to your plot by running this code abline(a = 0, b = 1) after the plot code.

** Create a chi-square distribution to plot against the sorted Mahalanobis distances. We add a 45-degree line to see how close our squared distances are to the line.**

```
# For concise code, store notedata2 in x
x <- notedata2

# Define the chi square distribution
quantiles <- qchisq((1:nrow(x) - 1/2) / nrow(x), df = ncol(x))

# Plot chi square against sorted distances
plot(quantiles, sorted_d2,
xlab = expression(paste(chi[3]^2, " Quantile")), ylab = "Ordered squared distances")

# Add 45 degree reference line
abline(a = 0, b = 1)
```



F) Interpret the plot of part e. Is the data MVN?

Based on the closeness of the data to the reference line and zero, the data does appear to be a multivariate normal distribution.

Problem 4

Use TTU graduate student exit survey data to answer the following questions.

```
grad <- read.csv("http://westfall.ba.ttu.edu/isqs6348/Rdata/pgs.csv")
```

A) This data contains a rating of how many students?

This dataset contains ratings from 2002 students:

```
nrow(grad)
```

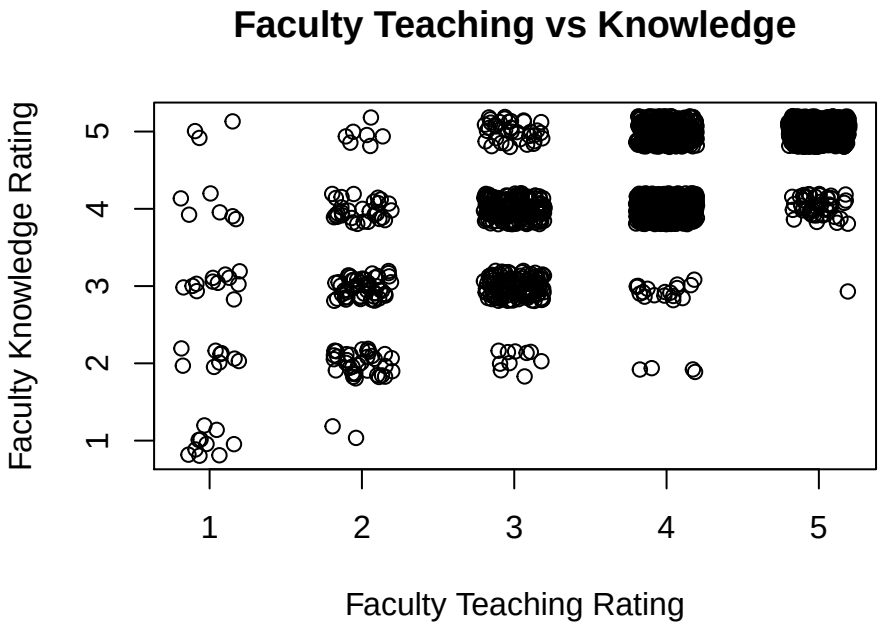
```
## [1] 2002
```

B) Make a scatterplot of "Facteaching" and "FacKnowledge"

If your plot looks odd, use jitter() of each variable, then plot.

Plotting the scores of faculty teaching and knowledge against each other results in an unhelpful plot, so we add a `jitter()` so add some noise and see individual datapoints more easily:

```
plot(jitter(grad$FacTeaching), jitter(grad$FacKnowledge),
     xlab = "Faculty Teaching Rating",
     ylab = "Faculty Knowledge Rating",
     main = "Faculty Teaching vs Knowledge")
```



C) Create new dataframe for "FacTeaching," "FacKnowledge", and "Housing"

Let's create an object with these three variables so we can compare them in the next section:

```
grad_3var <- grad[,c("FacTeaching", "FacKnowledge", "Housing")]
```

D) Find a correlation matrix for the data of part (c).

> If there are NAs (missing values) in your data, estimate the correlation matrix by all three following methods.

Let's attempt a correlation first to see if it will calculate: - Spoiler: it did not. It's time to handle some nulls!

```
cor(grad_3var)
```

```
##              FacTeaching FacKnowledge Housing
## FacTeaching            1           NA      NA
## FacKnowledge          NA            1      NA
## Housing               NA           NA       1
```

i. Complete-case analysis. First we will calculate the correlation using complete-case analysis, which deletes all rows that contain any NAs.

```
cor(na.omit(grad_3var))
```

```
##              FacTeaching FacKnowledge   Housing
## FacTeaching    1.0000000    0.7113826 0.1536942
## FacKnowledge   0.7113826    1.0000000 0.2108483
## Housing        0.1536942    0.2108483 1.0000000
```

ii. Available-case analysis. Next we calculate the correlation matrix using available-case, or "pairwise" analysis:

```
cor(grad_3var, use = "pairwise")
```

```
##              FacTeaching FacKnowledge   Housing
## FacTeaching    1.0000000    0.7109232 0.1556915
## FacKnowledge   0.7109232    1.0000000 0.2102034
## Housing        0.1556915    0.2102034 1.0000000
```

iii. Maximum likelihood estimation. Using the `norm` package, we can calculate the maximum likelihood estimate:

```
pre <- prelim.norm(as.matrix(grad_3var))
theta_hat <- em.norm(pre)
```

```
## Iterations of EM:
## 1...2...3...4...5...6...
```

```
ml.fit <- getparam.norm(pre,theta_hat)
```

```
corr.ml <- cov2cor(ml.fit$sigma)
corr.ml
```

```
##           [,1]      [,2]      [,3]
## [1,] 1.0000000 0.7120454 0.1541005
## [2,] 0.7120454 1.0000000 0.2103328
## [3,] 0.1541005 0.2103328 1.0000000
```

Is there a noticeable difference between the three methods of missing value treatment? Which method do you suggest? The correlation matrices are extremely close, in this case it appears that any of these methods would give you an accurate value. From this, I would speculate that there are few nulls in the data.

This could be because some responses are enforced through the survey software, or that students don't feel the need to skip questions. However, knowing that our data is distributed fairly normally it would be best to use maximum likelihood estimates to impute these nulls.