Z-Scores and Correlation

replace with your names

replace with the date

Part A

Before you begin

This homework exercise involves having you answer some questions, write some code, and create a nice HTML file with your results. When asked different questions, simply either type your coded or written responses after the ANSWER message. When asked to write code to complete sections, type your code in the empty code blocks that follow the ANSWER message (between the back ticks). After adding that code, you must make sure that it will execute. So remember to read the content and run each line of your code as you write it so that you know it executes correctly. If your code does not execute, then RMarkdown won't know what you are telling it to do and your HTML file will not be produced. Also, don't create your HTML file until you finish and know that all of your code works correctly.

1. Installing and using libraries in RStudio

1.1. Use the RStudio interface to install packages/libraries. Go to the Tools option and select Install Packages. Type the package name(s) correctly using the proper letter casing. Also, make sure that you check the box to Install Dependencies. Do not install with code.

- PerformanceAnalytics
- asbio (install if you were not in class/did not do this)
- Mac users, please make sure that you have XQuartz downloaded and in your Applications folder

1.2. Functions used for this assignment: c(), cor(), cov(), chat.Correlations(), getwd(), histogram(), kurtosis(), length(), library(), mean(), one.sample.z(), read.csv(), skewness(), sqrt(), str(), subset(), summary(), var()

2. Loading libraries

Use the library() function to load the following libraries: moments, lattice, PerformanceAnalytics, psych

ANSWER:

```
library("PerformanceAnalytics")
library("moments")
library("lattice")
library("psych")
```

3. Checking your working directory

Always make sure that your working directory points to Psyc109 on your desktop.

```
getwd()
```

```
## [1] "G:/Dropbox/Progs/_install/R/Exercises/Homework"
```

Part B

1. Comparing a sample mean to a population mean

1.1. The z-test is used to compare a sample mean to a population mean in order to determine if the population from which the sample was drawn is different from another population. The z-test is also only used for variables that are distributed normally. The NHST perspective assumes equivalence between the two populations (e.g., H0: mu1 = mu2), so a difference between the two groups should be 0 on average IF the populations have equivalent means. If, however, the sample produces a mean that differs from the population mean, you need to determine how likely such a result would occur under H0. If the likelihood is below your level of acceptable risk of making a Type I error, you could claim that the populations likely differ.

1.2. In order to make such a comparision for the z-test, you will need to know the **(a)** population mean, mu, **(b)** the population standard deviation, sigma, **(c)** the sample mean, xbar, and **(d)** the sample size, n. Because the standard deviation of sampling distributions is called the standard error of the mean, the z-test tells you how many standard errors the sampe mean deviates from the population mean. As with extreme z-scores, large z-test values are less likely to occur due to chance if there is really no difference between the population and your sample on your dependent variable.

1.3. Load the asbio library so that you can use the one.sample.z() function that is part of that library.
ANSWER:

```
#hide
library("asbio")
```

```
## Warning: package 'asbio' was built under R version 3.2.3
## Loading required package: tcltk
##
## Attaching package: 'asbio'
##
## The following object is masked from 'package:psych':
##
##     skew
##
## The following object is masked from 'package:PerformanceAnalytics':
##
##     Kappa
```

1.4. Read the cdc.csv file the you have used before. Assign it to an object named health.
ANSWER:

```
#hide data <- read.csv("cdc.csv")
health <- read.csv("C:/users/gcook/Desktop/Psyc109/cdc.csv")
```

1.5. If we wanted to test the average heights of people in our sample to that of an average height of a population, we need to specify those values. Obtain the mean height (inches) from your sample data from the cdc.csv file. You can use the mean() function or the summary() function from the psych library. If you use summary(), you will need to load the psych library if you have not already done so. **ANSWER:**

```
#hide
library("psych")
#hide
mean(health$height)
```

```
## [1] 67.1829
```

```
#hide
summary(health$height)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   48.00   64.00   67.00   67.18   70.00   93.00
```

1.6. You would normally examine whether the heights are distributed normally. We won't do that here, however, doing so is important. Now that you know the sample's mean height, you will need to compare that to the population mean. let's assume that the average adult height of the population is 67 inches tall (5'7") with a standard deviation of 3 inches. Create an object called popmean and assign it a value of 67; create an object called popsd and assign it a value of 3. **ANSWER:**

```
#hide
popmean <- 67
#hide
popsd <- 3
```

1.7. The cdc data set contains data from very many people (20K). Let' still just assume that's a sample for this exercise. However, your samples will be much smaller of course. In order to calculate the standard error of the mean for the z-test, you will need to know the size of your sample. Use the length() function to obtain the number of people in your height variable and assign that to a variable named samplesize. **ANSWER:**

```
#hide
samplesize <- length(health$height)

#take a look at it
samplesize
```

```
## [1] 20000
```

1.8. Now use all of those pieces of information to plug into the one.sample.z() function below where there is a ??. For greatest flexibility, replace the ?? with the objects you created rather than the actual values. The data argument in the function stands for the sample variable you are testing, so I've started it for you by identifying the health data frame. You can copy your modified code and then put it in the R code block to execute. Remember all R code needs to be in those code blocks.

one.sample.z(data = health$??, null.mu = ??, sigma = ??, n = ??)

ANSWER:

```
#hide one.sample.z(data = health$height, null.mu = popmean, sigma = popsd, n = samplesiz
e)
```

1.9. Notice the z-test value and the p-value for the test. Compare your *p*-value to your alpha level and decided whether you would retain or reject H0. State your answer in words. **ANSWER:** I would reject H0 because the *p*-value is less than an alpha of .05. The difference between the means that that many standard errors is unlikely due to change for a sample of 20K.

1.10. One thing you will have noticed about this test is that when you have very, very, large sample sizes, the standard error of the mean (sigma/sqrt(n)) will be very, very small. Consequently, you may reject H0 even when the sample mean is very close to the population mean. I used this example to illustrate that point and make you become vigilant about data using very large samples.

Part B

1. Correlations

A correlation describes the relationship between two variables. More precisely, the usual measure of a correlation describes the relationship between two equal interval numeric variables. This part will examine how to use the Pearson *r* correlation.

2. Assumptions of bivariate linear correlation using Pearson r

First, let's take a look at the assumptions of correlations the Pearson *r* correlation coefficient:

1. Our two variables should be measured at the interval or ratio level (i.e., they are continuous).
2. There needs to be a linear relationship between the two variables.
3. There should be no significant outliers.
4. Your variables should each be approximately normally distributed.

3. Reading in a data file

3.1. Read a data file and assign its contents to a data frame object named icecream. We could name this object any name we want (e.g., bananasundae, ketchup, or ice). In other words, the object name does not need to match the name of the file. In this case, however, icecream seems appropriate as a name.

ANSWER:

3.2. Examine the structure of your data frame and its contents as you have in earlier assignments. Pay attention to the names and spelling of the variables. Just for practice, use the summary() function from the psych library in order to examine the means of the two variables. If you run into any problems it might help to view your data by graphing a histogram.

ANSWER:

```
#Check the variable names
str(icecream)
```

```
## 'data.frame':    13 obs. of  3 variables:
##  $ Id          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Temp        : num  14.2 16.4 11.9 15.2 18.5 22.1 19.4 25.1 23.4 18.1 ...
##  $ IceCreamSold: int  215 325 185 332 406 522 412 614 544 421 ...
```

```
#Calculate the and median mean for both variables; compare them and ask yourself if their
values inform you about skew. Remember you need to specify the data frame object in order
to tell R to obtain some of the data frame's contents.

summary(icecream$Temp)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   11.90   16.40   18.10   18.62   22.10   25.10
```

```
summary(icecream$IceCreamSold)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   185.0   332.0   412.0   403.4   445.0   614.0
```

3.3. As you can probably guess from the name, the data frame you just loaded contains information about temperature and amount of icecream sold. There is also an Id variable, which corresponds to the id number of the experimental unit. I advise that you always create an Id variable so that you can keep track of your participants and because that variable will be useful for identifying specific data points. Use the appropriate fucntions from the moments library to calculate the skewness and kurtosis of both variables. Consider whether the variables are distributed normally.

ANSWER:

```
skewness(icecream$Temp)
```

```
## [1] 0.06307683
```

```
skewness(icecream$IceCreamSold)
```

```
## [1] -0.2008352
```

```
kurtosis(icecream$Temp)
```

```
## [1] 2.176456
```

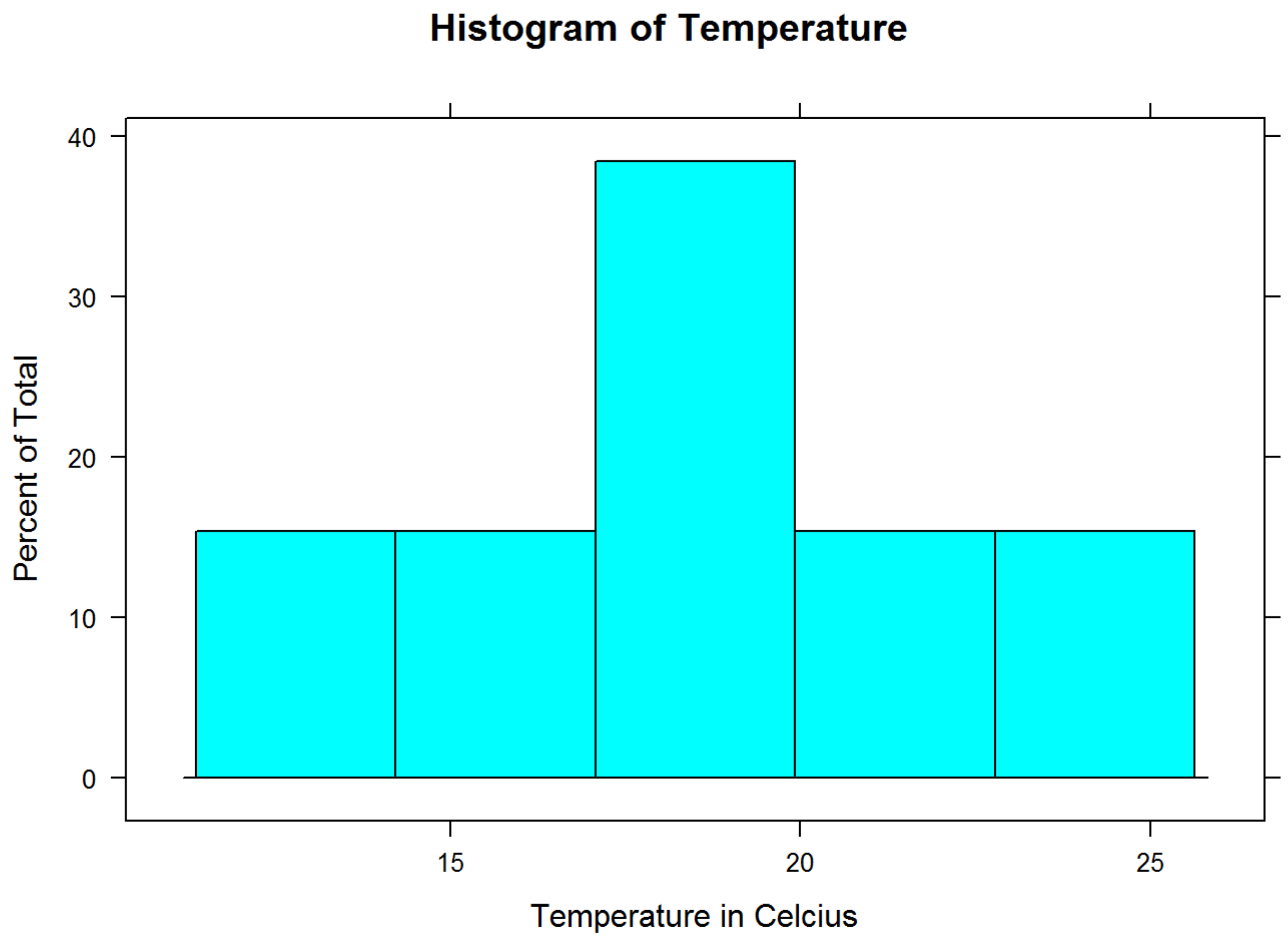
```
kurtosis(icecream$IceCreamSold)
```

```
## [1] 2.61672
```

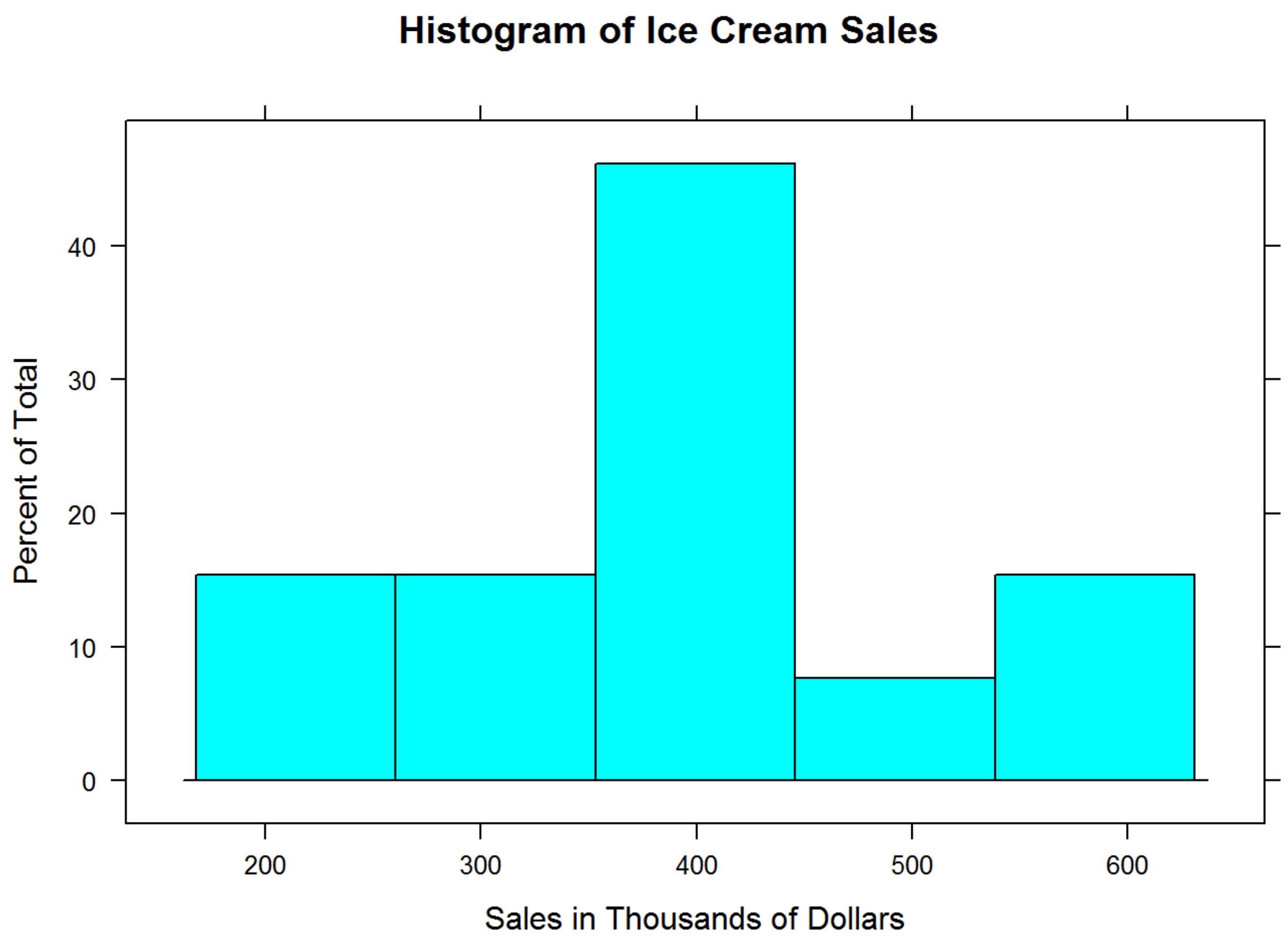
3.4. The assumptions of a correlation require that our variables be distributed normally. We can check this assumption both graphically and statistically. Use the histogram function from the lattice library to create a distribution for both variables. You can use the default distribution (a percentage breakdown) rather than one using the frequency counts. For practice, consider modifying the main and xlab arguments that you modified in the previous homework so that the labels are labeled clearly. You could also use the qqplot() and qqline() functions, but for simiplicity we won't do so here.

ANSWER:

```
library(lattice)
histogram(~icecream$Temp,
          main="Histogram of Temperature",
          xlab="Temperature in Celcius")
```



```
histogram(~icecream$IceCreamSold,
          main="Histogram of Ice Cream Sales",
          xlab="Sales in Thousands of Dollars")
```



4. Scatterplots

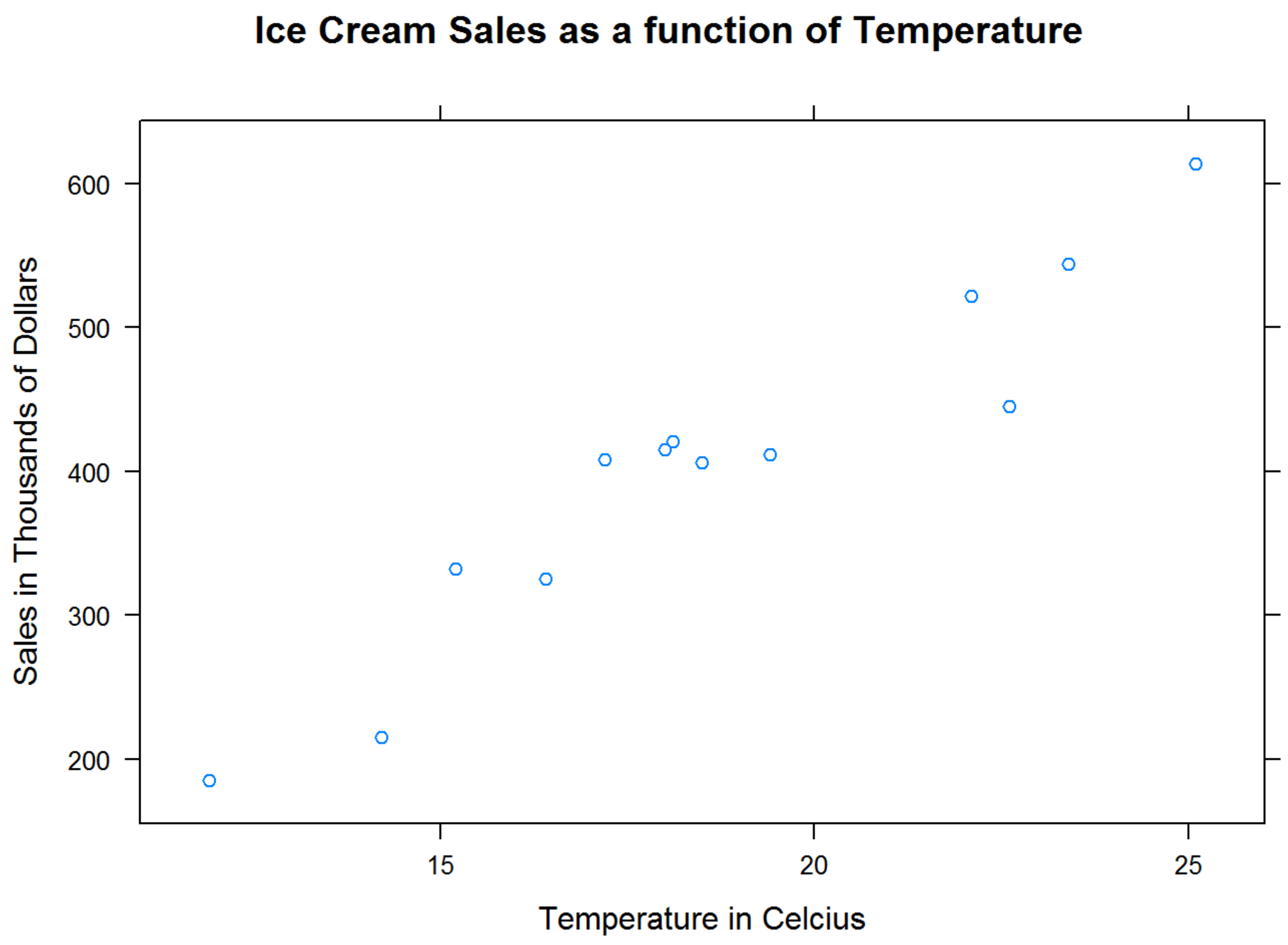
4.1. A scatterplot is one of the best ways to visualize the nature of a relationship between two variables. Scatterplots plot individual x and y variables along the x and y axes. When using the lattice library, you will need to specify the data frame object that contains your data as well as the variables and then write the formula for the plot: y ~ x ; y (weight) plotted as a function of x (height).

Or you may want to plot x (height) as a function of y (weight) as: x ~ y

Use the xyplot() function to plot IceCreamSold as a function of Temp from the icecream data frame. Also, make sure to add labels to your scatterplot that are specific. The temperature is in Celsius and sales are in thousand of dollars. To make the code legible, we will make each argument in the xyplot() function on its own line of code; each line after the first needs to be indented so that R knows all the lines of code go together.

ANSWER:

```
xyplot(IceCreamSold ~ Temp,
       data = icecream,
       main = "Ice Cream Sales as a function of Temperature",
       xlab = "Temperature in Celcius",
       ylab = "Sales in Thousands of Dollars")
```



Based on the scatterplot, do the data appear to follow a linear pattern? Curvilinear? **ANSWER:** Looks fairly linear; more linear than curvilinear

If you wanted to add a regression line, you could add the following argument to your xyplot() function: type = c("p", "r")

5. Evaluating the magnitude of the correlation coefficient

The graph provides useful information, but it does not provide the magnitude of the correlation.

5.1. A correlation is a statistical measure the represents the linear relationship between two variables.

5.2. You can calculate the Pearsons *r* correlation by dividing the covaraince of the two variables you are interesed in (In this case Temperature and Ice Cream sold) by the two standard deviations of the variables. For example:

```
#Calculate the covariance of the two variables (or sum of cross products)
spxy <- cov(icecream$Temp, icecream$IceCreamSold)

#Then you need to calculate the standard deviations for both variables
xvar <- var(icecream$IceCreamSold)
yvar <- var(icecream$Temp)

#You can then divide the covariance by square root of the variances:
rxy <- spxy/sqrt(xvar * yvar)
rxy
```

```
## [1] 0.9545637
```

5.3. However, we can use the built-in cor() function to avoid all of those steps. We will calculate the magnitude of Pearson's *r* correlation coefficient using the cor() function and assign that value it to an obect named r. Then we will square it in order to obtain the coefficient of determination, which provides a measure of the variance in one variable that is determined by (explained by/accounted for by) another variable. In this case, r-squared will provide the proportion of variance in IceCreamSold that can be accounted for by Temp (or the proportion of variance in Temp that can be accounted for by IceCreamSold).

```
# assign the r value to a an obect named r
r <- cor(icecream$Temp, icecream$IceCreamSold)

# square r and assign it to an obejct named r2
r2 <- r^2

# display both
r; r2
```

```
## [1] 0.9545637
```

```
## [1] 0.9111919
```

5.4. Use the cor.test() function to obtain the r score and the exact *p*-value associated with obtaining a correlation of that size or larger if, under H0, the variables were not related and therefore the correlation was 0. When you need the exact p-value for your r values, you should use this test.

```
cor.test(icecream$Temp, icecream$IceCreamSold)
```

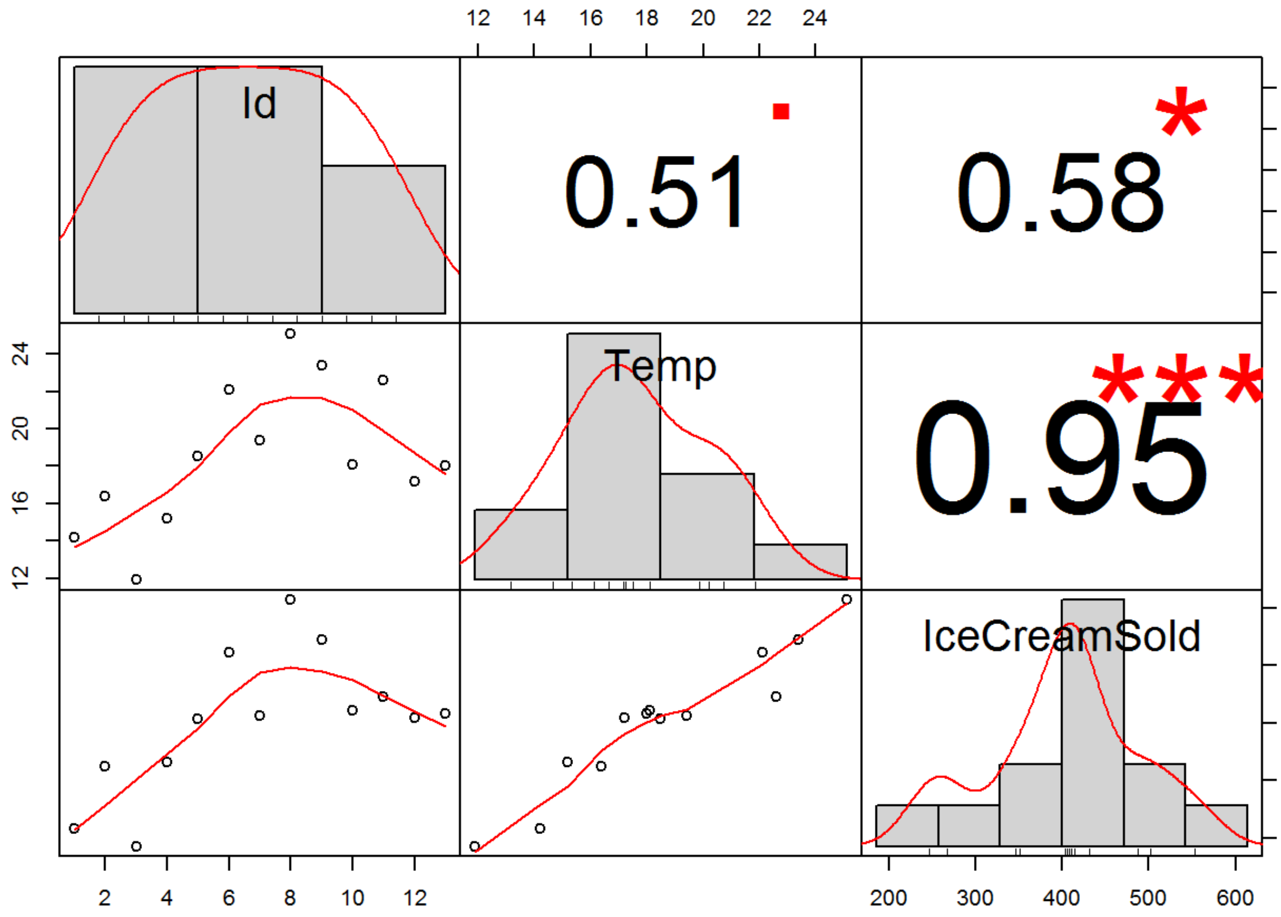
```
## 
##  Pearson's product-moment correlation
## 
## data:  icecream$Temp and icecream$IceCreamSold
## t = 10.624, df = 11, p-value = 4.026e-07
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8513425 0.9866303
## sample estimates:
##       cor 
## 0.9545637
```

5.5. You could write your own R function to combine histograms for x and y your variables, the scatterplot, and the correlation coefficient (r) to make that information appear any way you like to see them. However, we can use a function that does this already.

The chart.Correlation() function from the PerformanceAnalytics library provides: **(a)** histograms for each variable along with a density plot overtop the histograms, **(b)** the scatterplot along with a line that fits the points, and **(c)** the correlation coefficient. You can apply the function on an entire data frame object if you wanted to examine all variables; in our case, we only have 3 variables so the visual is a simple 3 x 3 matix.

When you use some functions, you might obtain errors. You should take a look a the errors that R reports in order to determine if they are important for your data. If you know they are not an issue and do not want them to print out in your RMarkdown file, you can choose not to display them by setting the warning argument to FALSE as well as not display messages by setting that to FALSE as well.

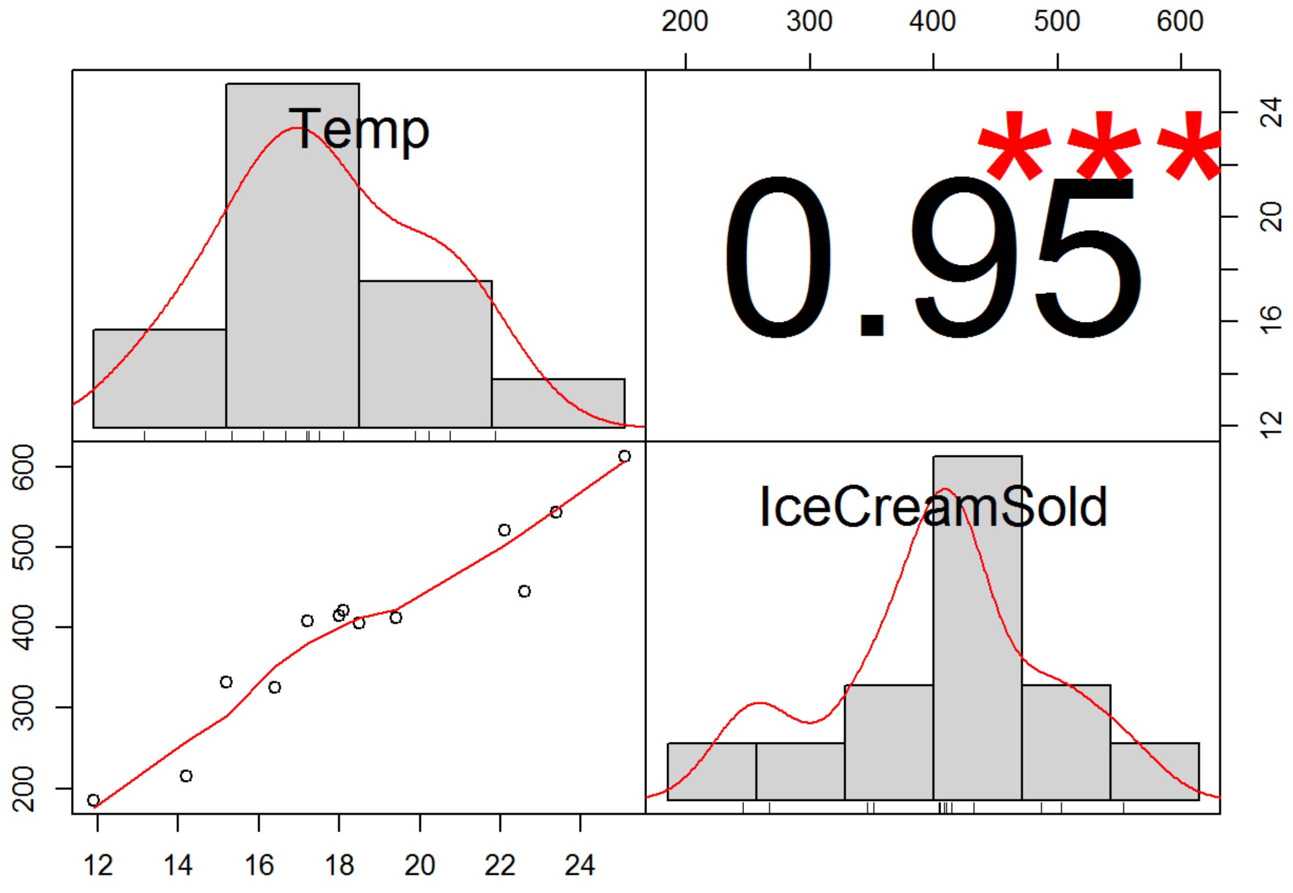
```
#Notice the header change to hide warnings and messages from appearing in you htlm file.
chart.Correlation(icecream)
```



5.6. One problem is that the ID variable isn't really useful here, so you would want to remove it. We can select subsets of our data frame and only examine the variables of interest. There are different ways we can do so, which are only a matter of personal preference. Here are two ways.

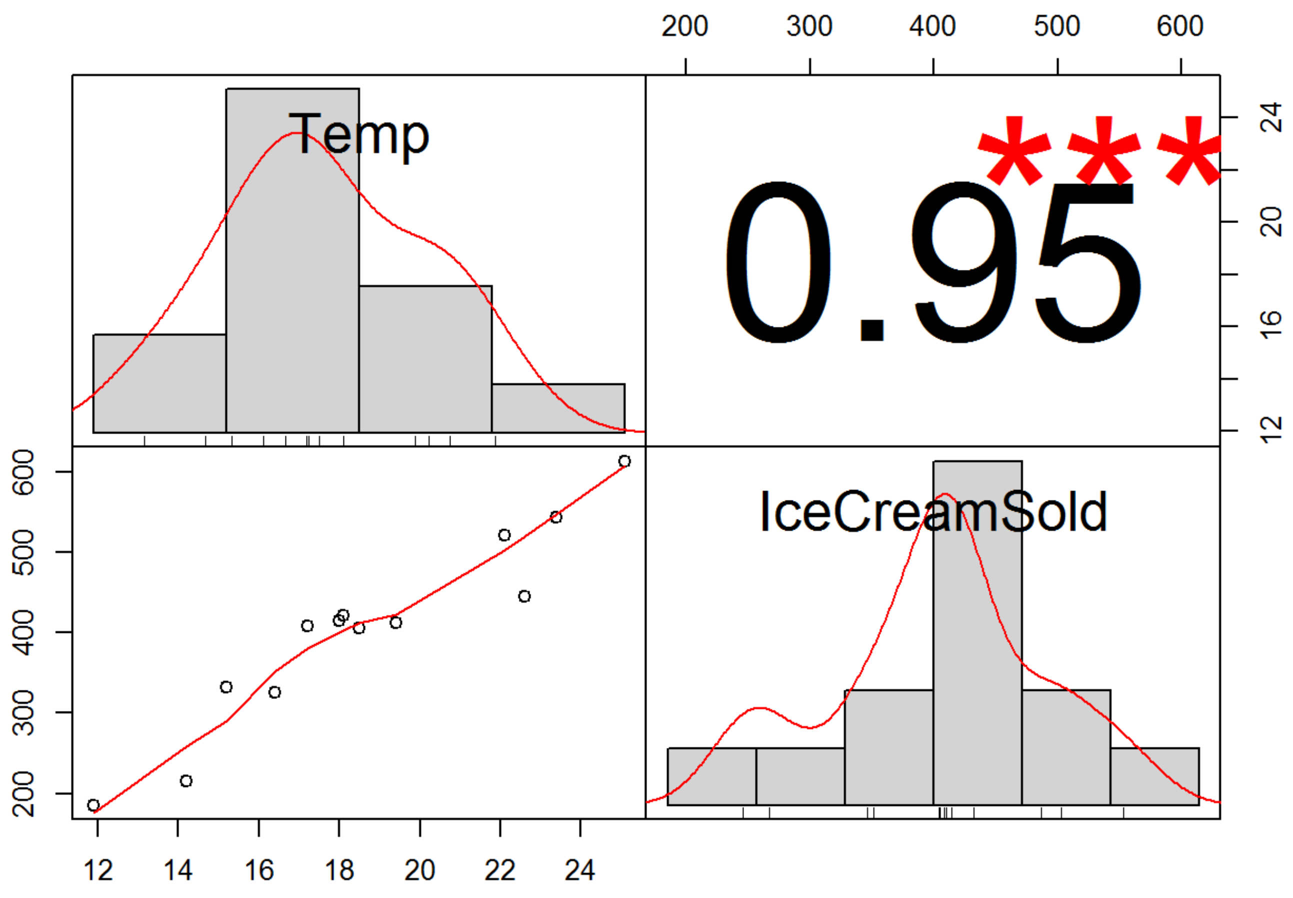
5.7. Because a data frame is composed of [rows, columns] , we can present all rows of data (all people), but remove Column 1 because that column is the Id variable.

```
chart.Correlation(icecream[, -1])
```



5.8. If thinking about a data frame as rows and columns is confusing, you can pass all of the rows and then specify only the column names that you wish to examine. In this case, we have only two variables to examine.

```
chart.Correlation(icecream[, c("Temp", "IceCreamSold")])
```



5.9. By NHST interpretation, a significant correlation would be one for which its value would be very unlikely under the null hypothesis that would assume $r = 0$. Larger correlations (especially those with with large sample sizes) would be rare to choose at random from a population of r values that follow a normal distribution having a mean = 0.

Note that the scatterplot is shown in the bottom left corner, graphs of the variables are shown in the middle and the correlation is in the top right.

The number of asterisks above the correlation value indicate whether the corresponoding *p*-value for the Pearson *r* coefficient is below an alpha of (0.05 = *, 0.01 =* ***, 0.001 =***).

The magnitude of the correlation indicates a measures of strength of a linear relationships between the variables. For example, we see that this is a positive linear correlation between temperatue and amount of ice cream sold, where high temperatures are assocaited with a high amount of ice cream sold. Like wise low temperatures are associated with low amount of ice cream being sold. This can be visually understood by looking at the scatterplot and seeing that dots that are low on temperature and also low on ice cream sold.

6. Identifying outliers

6.1. The reason why this chart is usefull is that it can allow for you graph the variables, potentially seeing something that might not show up when only viewing the descriptive statistics.

6.2. For example, load in the next dataset and calculate the mean and skewness of the variables. **ANSWER:**

```
#hide skewed <- read.csv("skewedicecream.csv")
#
skewed <- read.csv("c:/users/gcook/desktop/Psyc109/skewedicecream.csv")
str(skewed)
```

```
## 'data.frame':    13 obs. of  3 variables:
##  $ Id          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Temp        : num  14.2 16.4 11.9 15.2 18.5 22.1 19.4 25.1 23.4 18.1 ...
##  $ IceCreamSold: int  215 325 185 332 406 522 412 614 544 421 ...
```

```
#Calculate the mean of both variables
mean(skewed$Temp); mean(skewed$IceCreamSold);
```

```
## [1] 19.54615
```

```
## [1] 386.8462
```

```
#Calculate the skewness and kurtosis of both variables
skewness(skewed$Temp); kurtosis(skewed$Temp)
```

```
## [1] 0.4872131
```

```
## [1] 2.67761
```

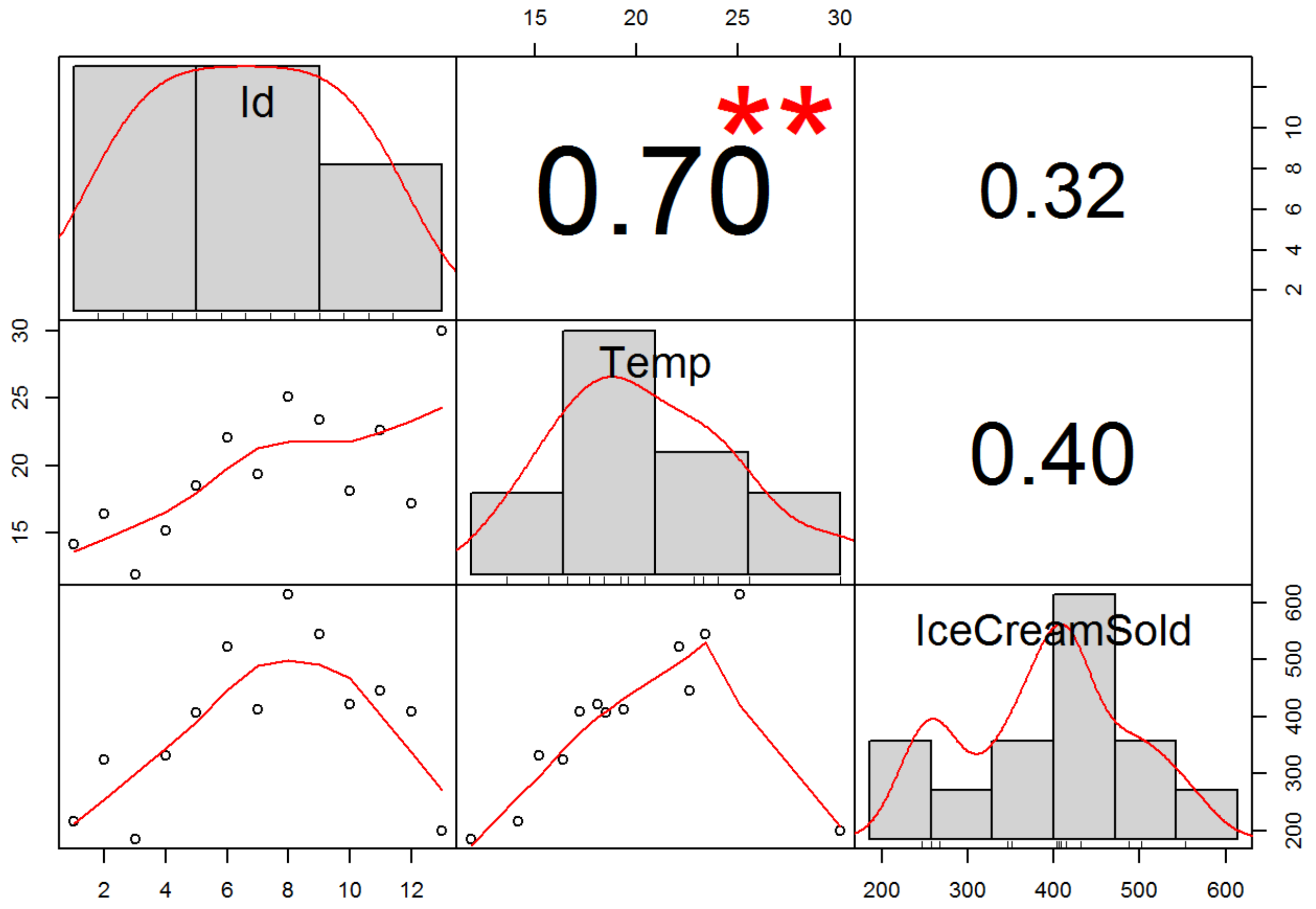
```
skewness(skewed$IceCreamSold); kurtosis(skewed$IceCreamSold)
```

```
## [1] -0.0692761
```

```
## [1] 2.12716
```

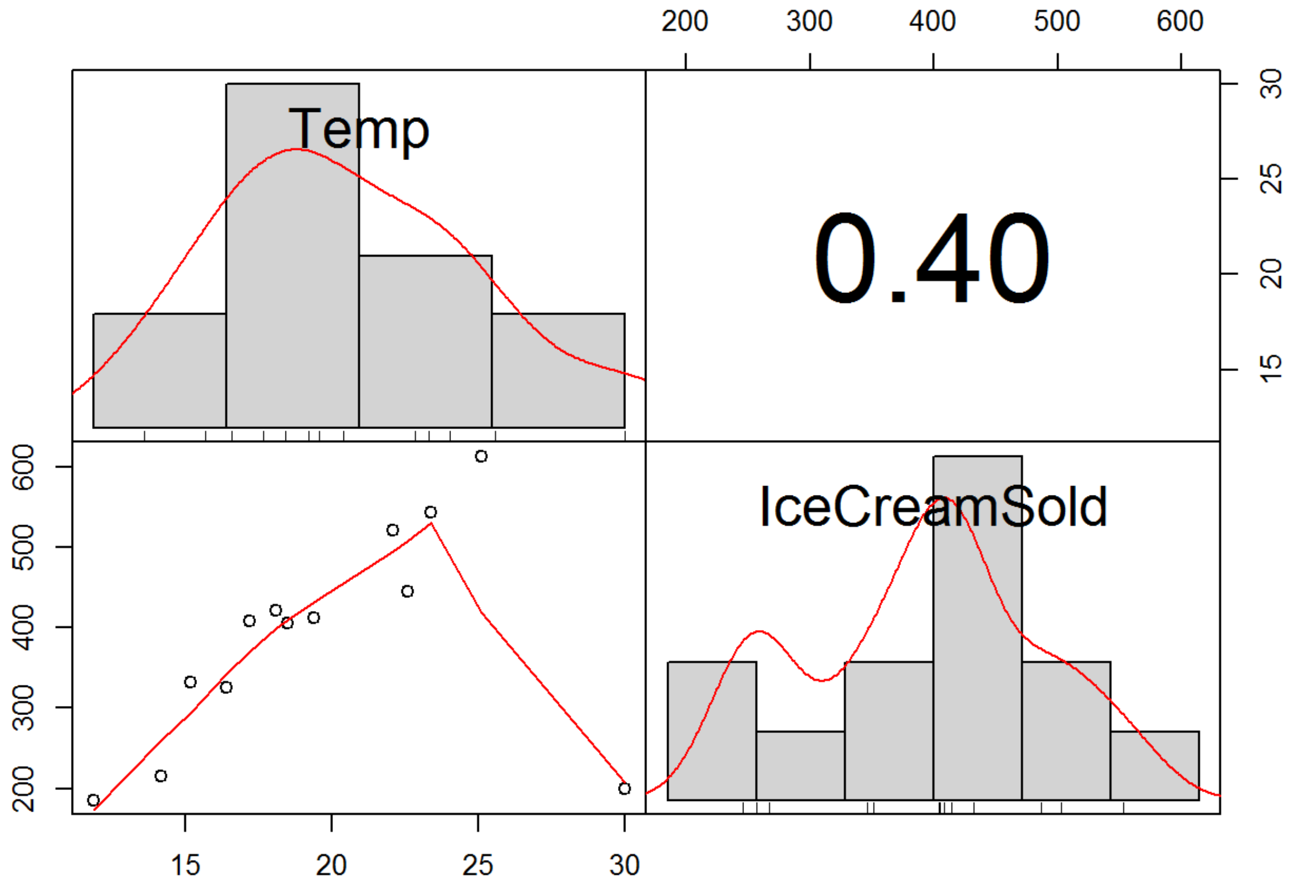
6.3. After loading in the new dataset, use chart.Correlation() to chart the correlations of variables in the skewed data frame. **ANSWER:**

```
#hide
chart.Correlation(skewed)
```



6.4. As before, the chart.Correlation() function will create a chart with all of the variables in the data frame unless you exclude some.

```
chart.Correlation(skewed[ , -1])
```



6.5. As you can see the magnitude of the correlation has decreased and the relationship between the variables has been altered greatly when the one score that is very different from the others is part of the data set. You may also notice that the graph for IceCreamSold is not as clean as it was before. This is what happens when you do not have a lot of data points and when one of those data points is an outlier. Outliers can greatly influence the relationship between two variables, as well as the distribution of one of your variables. In this case not a lot of ice-cream was sold on a very hot day. When running a study it would be very possible to collect data on a day like that. The next few steps will go over how to determine if the data point is an outlier.

7. Removing Outliers: Something important to consider

7.1 Now we will determine if the score might be considered and outlier by graphing out the correlation and determine if a specific data point is very different from the others. In the previous graph there is a data point in the far left corner that is high on temperature and low on ice cream sold. It appears that this data value is greatly influencing the correlation, bringing it down to a .40. First we view the data set and attempt to determine which data point is the outlier.

```
#Let's view the skewed dataset
View(skewed)
```

7.2. Luckily, because of the small dataset, data point 13 can be identified as the outlier. If we determine a score in an outlier, we can simply make the score a missing value, or not available (NA). In order to do this, we can modify the Temp values to replace the scores with NA by telling R to take the skewed$Temp variable and check to see if the Id value is equal to (==). The double equal evaluates the Id number to see if it matches the value. In the example below, we use a vector c(13) rather than simply using a specfic of 13 because the vector would allow us to evaluate matches for more than one Id number (e.g., c(1,2,13)).

To remove this variable we will create an outlier object which can be used to remove all the outliers that we identify.

```
skewed$Temp[skewed$Id == c(13)] <- NA # or more than one Id number if you have more than
1 outlier
View(skewed) # notice how the value is not NA for participant 13
```

7.3. Now that you have made some values NA, you can use a function that selects variables that has only the x and y data pairs. The complete.cases() function will allow you to select only the complete x and y cases (not any row with NA). We will also need to create a small data frame that contains this subset of data so that we can run the correlation; the subset() function makes this easier to so. The subest() function also allows you to specifty the variables you wish to select for your subset.

7.4. The following code is using subset() to create a subset of data. The first argument is the original data that you want to subset; in this case skewed. The second argument tells R to use only the compete cases of the skewed data frame, and the third argument selects your variable(s) of interest; the c() is used to combine the variables of interest. The new data frame is assigned to an object named notskewed.

Run the function on the skewed data frame and assign the new data frame to a new object (e.g., notskewed). If you look at the contents of your new data frame, you will see that you have a smaller data frame now.

```
notskewed <- subset(skewed,
                    complete.cases(skewed),
                    select = c(Temp, IceCreamSold))

# Take a look at the new data frame
notskewed
```

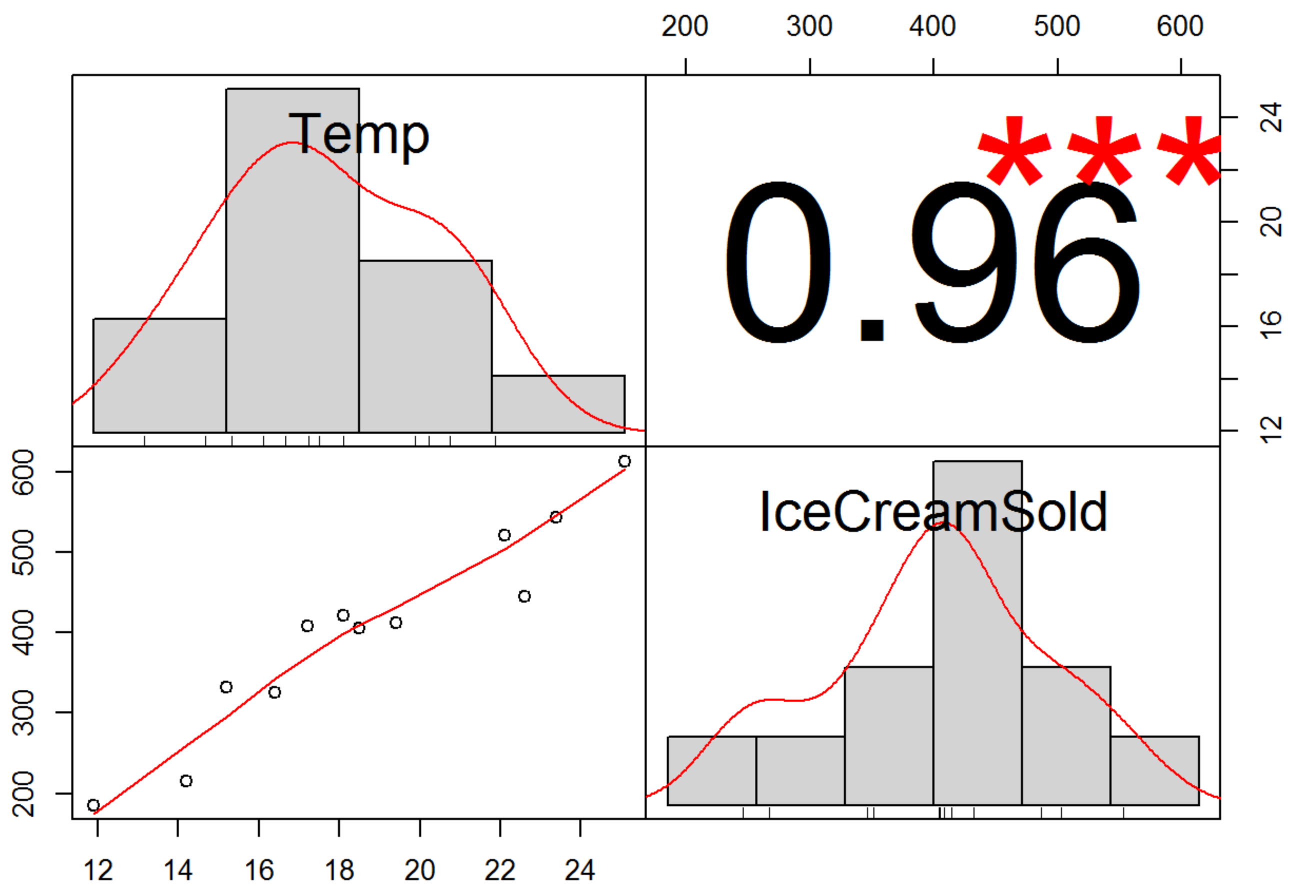
```
##    Temp IceCreamSold
## 1  14.2          215
## 2  16.4          325
## 3  11.9          185
## 4  15.2          332
## 5  18.5          406
## 6  22.1          522
## 7  19.4          412
## 8  25.1          614
## 9  23.4          544
## 10 18.1          421
## 11 22.6          445
## 12 17.2          408
```

7.5. Now, in order to calculate the correlation and see the chart as before, we need to apply the functions to the new data frame.

```
notskewedr <- cor(notskewed$Temp, notskewed$IceCreamSold)
notskewedr
```

```
## [1] 0.9575066
```

```
chart.Correlation(notskewed)
```



7.5. As you can now see, removing the data point (outlier) has greatly increased the correlation and reveals the strong relationship between Temperature and Ice Cream sold. If there was more than out outlier, both can be placed in the Outlier object.

8. Try it yourself!

8.1. Load the “Book.csv” from your working directory into an object named Book and look at its structure. **ANSWER:**

```
# Load the csv
#hide Book <- read.csv("Book.csv")
Book <- read.csv("C:/users/gcook/desktop/Psyc109/Book.csv")
```

8.2. Now examine the names of the variables, mean, median, skewness, and kurtosis of the relevant variables. **ANSWER:**

```
# Names
str(Book); names(Book)
```

```
## 'data.frame':    40 obs. of  4 variables:
##  $ ID    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ BOOKS : int  0 1 0 2 4 4 1 4 3 0 ...
##  $ ATTEND: int  9 15 10 16 10 20 11 20 15 15 ...
##  $ GRADE : int  45 57 45 51 65 88 44 87 89 59 ...
```

```
## [1] "ID"     "BOOKS"  "ATTEND" "GRADE"
```

```
# Mean
mean(Book$BOOKS); mean(Book$ATTEND); mean(Book$GRADE)
```

```
## [1] 2
```

```
## [1] 14.1
```

```
## [1] 63.55
```

```
# Median
median(Book$BOOKS); median(Book$ATTEND); median(Book$GRADE)
```

```
## [1] 2
```

```
## [1] 15
```

```
## [1] 60.5
```

```
# Skewness
skewness(Book$BOOKS); skewness(Book$ATTEND); skewness(Book$GRADE)
```

```
## [1] 0
```

```
## [1] -0.01795583
```

```
## [1] 0.5010131
```

```
# kurtosis
kurtosis(Book$BOOKS); kurtosis(Book$ATTEND); kurtosis(Book$GRADE)
```

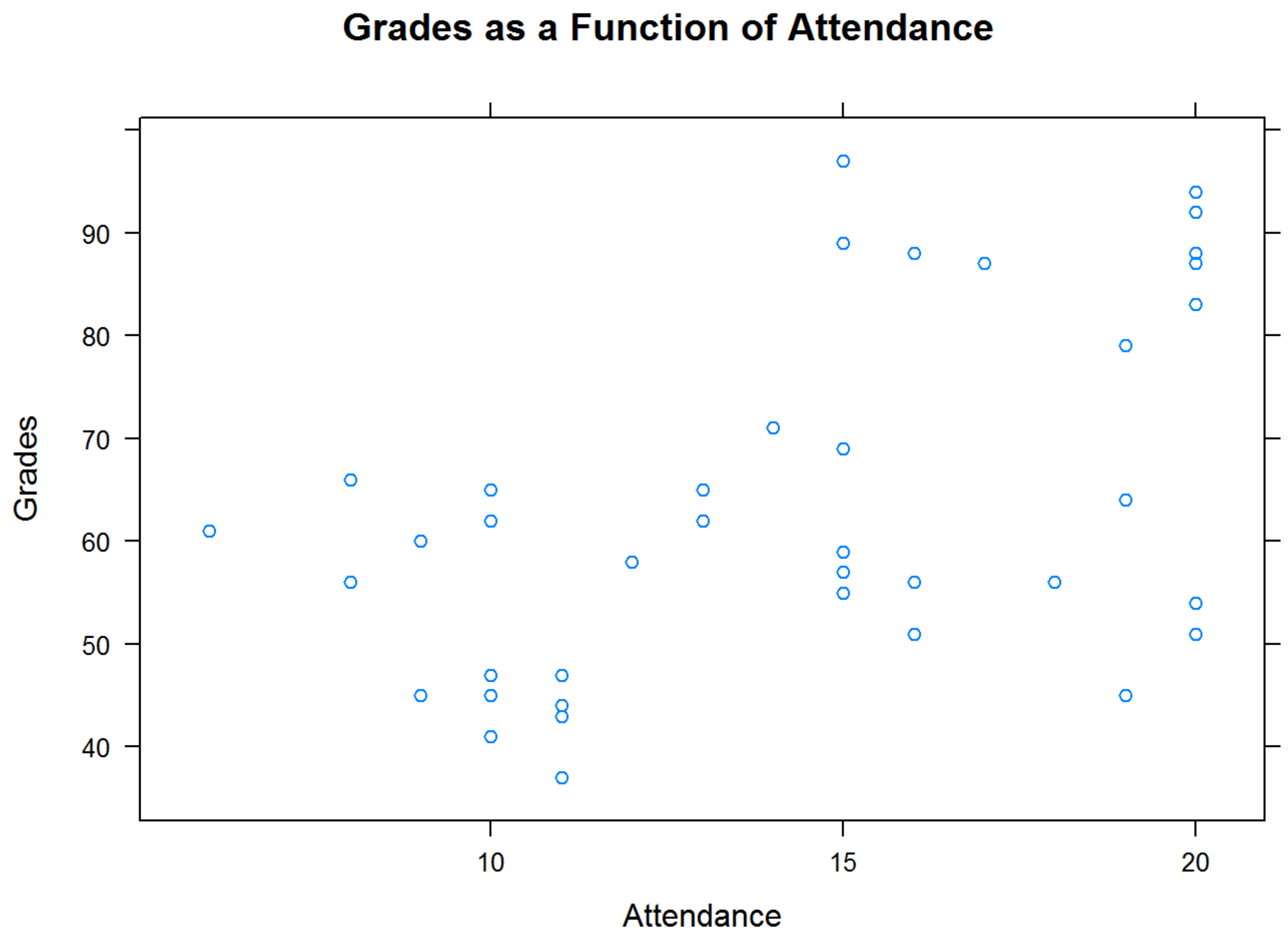
```
## [1] 1.7
```

```
## [1] 1.741692
```

```
## [1] 2.151475
```

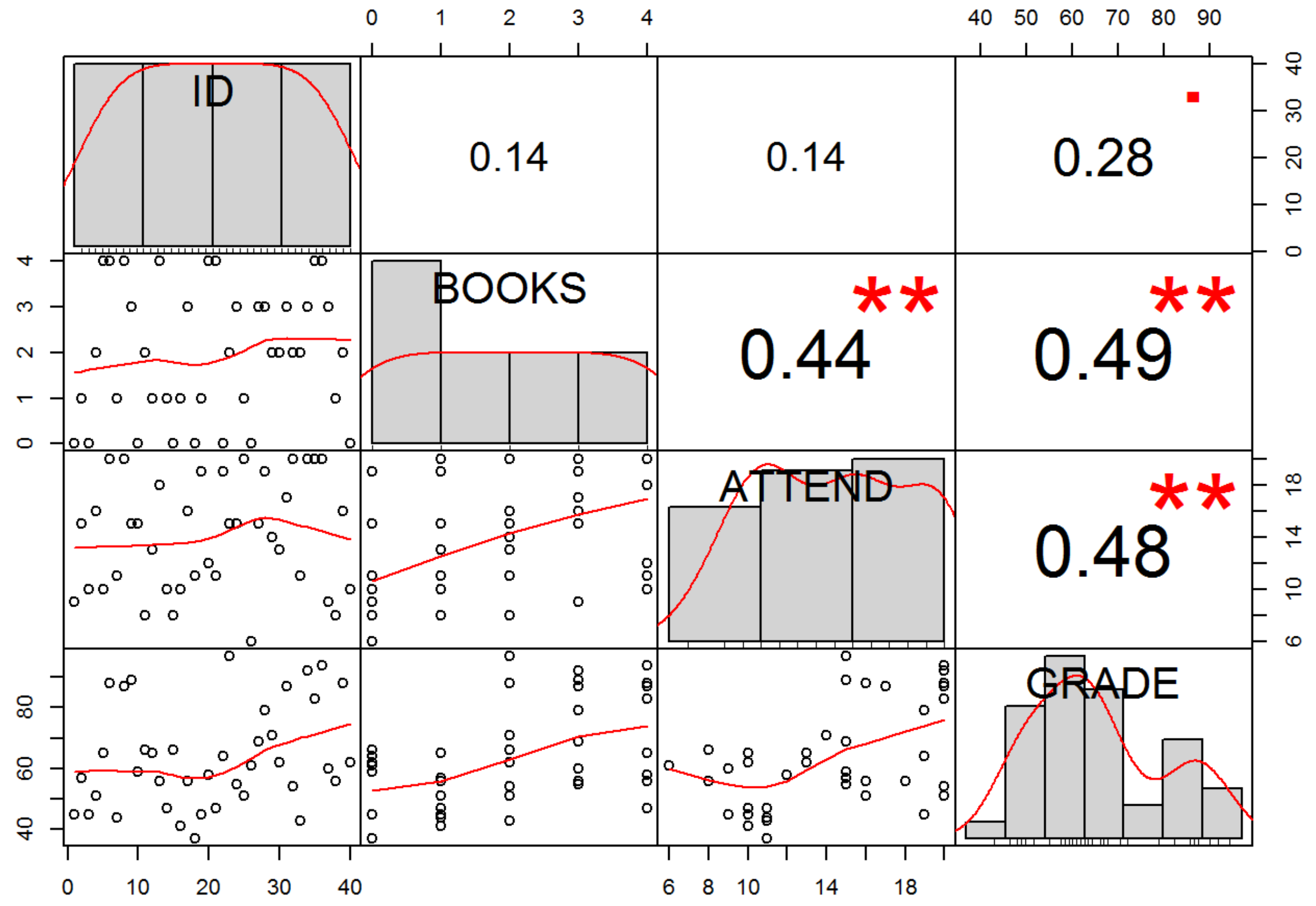
8.3. Plot out a correlation using the xyplot() function to see if there appears to be a linear relationship between the two variables of your choice **ANSWER:**

```
xyplot(GRADE ~ ATTEND,
       data = Book,
       main = "Grades as a Function of Attendance",
       xlab = "Attendance",
       ylab = "Grades")
```



8.4. Now using the chart.Correlation() function, chart out the data set. **ANSWER:**

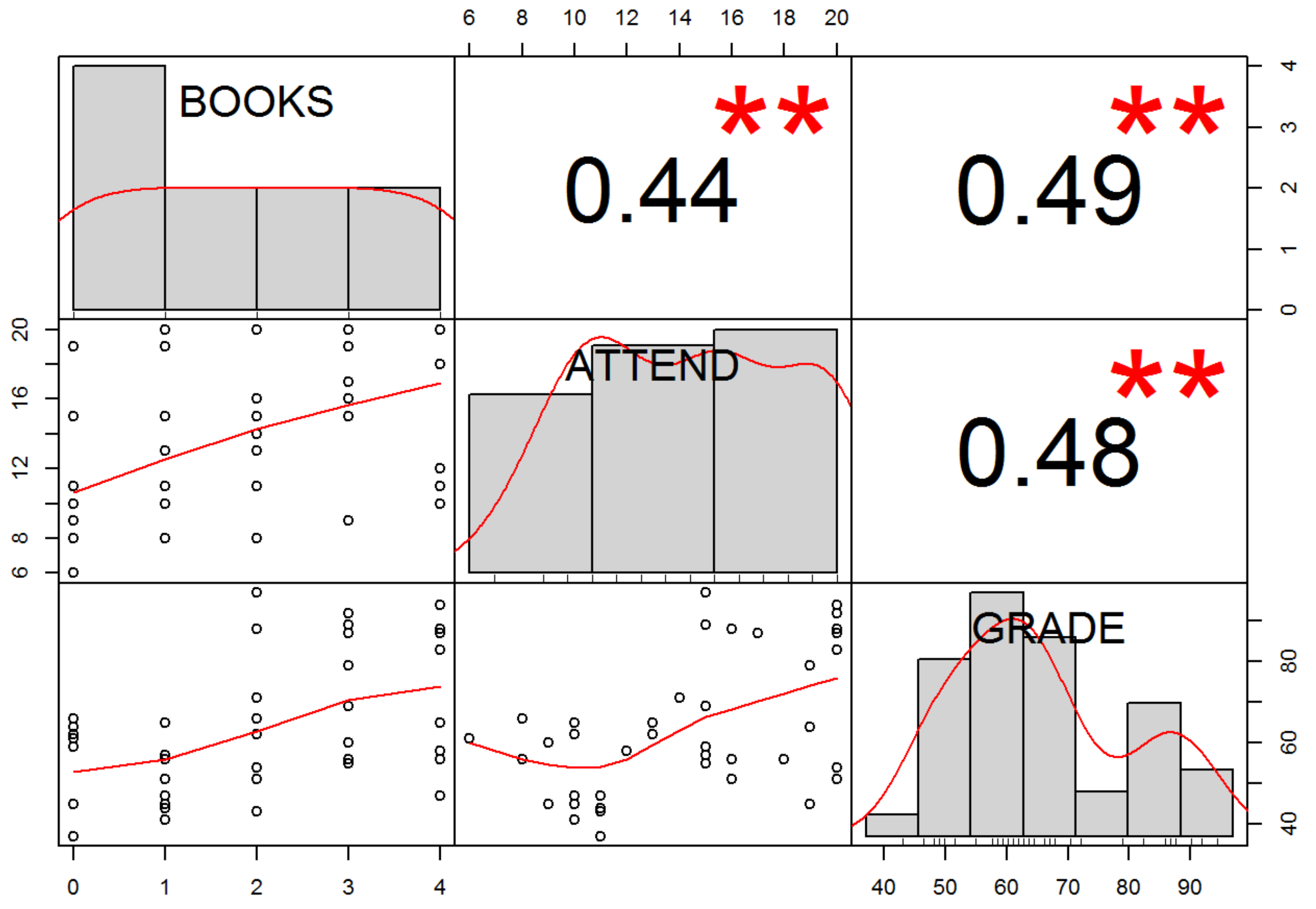
```
#hide
chart.Correlation(Book)
```



Book

8.5. Now using the chart.Correlation() function, chart out the data set, by removing the ID variable. Make sure you spell the variable name correctly; use str() to help you if necessary. **ANSWER:**

```
#hide
chart.Correlation(Book[, -1])
```



8.6. Get a subset of the data that does not include the Id variable. **ANSWER:**

```
mysubset <- subset(Book,
                   complete.cases(Book),
                   select = c(BOOKS, ATTEND, GRADE))
# take a look
mysubset
```

```
##    BOOKS ATTEND GRADE
## 1      0      9    45
## 2      1     15    57
## 3      0     10    45
## 4      2     16    51
## 5      4     10    65
## 6      4     20    88
## 7      1     11    44
## 8      4     20    87
## 9      3     15    89
## 10     0     15    59
## 11     2      8    66
## 12     1     13    65
## 13     4     18    56
## 14     1     10    47
## 15     0      8    66
## 16     1     10    41
## 17     3     16    56
## 18     0     11    37
## 19     1     19    45
## 20     4     12    58
## 21     4     11    47
## 22     0     19    64
## 23     2     15    97
## 24     3     15    55
## 25     1     20    51
## 26     0      6    61
## 27     3     15    69
## 28     3     19    79
## 29     2     14    71
## 30     2     13    62
## 31     3     17    87
## 32     2     20    54
## 33     2     11    43
## 34     3     20    92
## 35     4     20    83
## 36     4     20    94
## 37     3      9    60
## 38     1      8    56
## 39     2     16    88
## 40     0     10    62
```

8.7. For your correlation, get the exact *p*-value. **ANSWER:** The p-value for GRADE and ATTEND is 0.001; r=.48

```
cor.test(Book$GRADE, Book$ATTEND)
```

```
##
##  Pearson's product-moment correlation
##
## data:  Book$GRADE and Book$ATTEND
## t = 3.3929, df = 38, p-value = 0.001629
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.2008451 0.6900469
## sample estimates:
##       cor
## 0.4821864
```

8.8. How many of the correlations are significant (test at alpha = .05) and how did you make that decision? **ANSWER:** 3 are. The asterisks indicate a significant relationship for all bivariate correlations at the .01 level.

8.9. Describe how strong the correlations are. **ANSWER:** The correlation between books and attendance is .44; between books and grades is .49; and between attendance and grades is .48. All three correlations are moderate in strenth. The coefficients of determination are all approximately .2 (20%)