Assignment 1

Nicholas Jacob

2024-08-19

```
knitr::opts_chunk$set(echo = TRUE)
library(plyr)
```

Using R: Vectors

Using `c` to combine the values, we see that x is a vector.

```
x<- c(3,12,6,-5,0,8,15,1,-10,7)
is.vector(x)
```

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

To create the new vector y as a sequence from the min of x to the max of x, we do the following:

```
y <-seq(min(x),max(x), length.out = 10)
y
```

```
##  [1] -10.000000  -7.222222  -4.444444  -1.666667   1.111111   3.888889
##  [7]   6.666667   9.444444  12.222222  15.000000
```

I was not familiar with the `length.out` command but found it in the Help package to see that it would restrict the output to that many elements.

We compute the desired stats next

```
#consider changing this one with some tidy code
sum(x)
```

```
## [1] 37
```

```
sum(y)
```

```
## [1] 25
```

```
mean(x)
```

```
## [1] 3.7
```

```
mean(y)
```

```
## [1] 2.5
```

```
sd(x)
```

```
## [1] 7.572611
```

```
sd(y)
```

```
## [1] 8.41014
```

```r
var(x)
```

```
## [1] 57.34444
```

```r
var(y)
```

```
## [1] 70.73045
```

```r
mad(x)
```

```
## [1] 5.9304
```

```r
mad(y)
```

```
## [1] 10.29583
```

```r
quantile(x,1/4)
```

```
##  25%
## 0.25
```

```r
quantile(y,1/4)
```

```
##   25%
## -3.75
```

```r
quantile(x,3/4)
```

```
##  75%
## 7.75
```

```r
quantile(y,3/4)
```

```
##  75%
## 8.75
```

```r
quantile(x,1/5)
```

```
## 20%
##  -1
```

```r
quantile(y,1/5)
```

```
## 20%
##  -5
```

```r
quantile(x,3/5)
```

```
## 60%
## 6.4
```

```r
quantile(y,3/5)
```

```
## 60%
##   5
```

```r
quantile(x,2/5)
```

```
## 40%
## 2.2
```

```r
quantile(y,2/5)
```

```
##           40%
## -1.665335e-15
```

```
quantile(x,4/5)
```

```
## 80%
## 8.8
```

```
quantile(y,4/5)
```

```
## 80%
##  10
```

To do sampling with replacement we do the following

```
sample(x,7,TRUE)
```

```
## [1]  7  3  7 -5  1  8  0
```

The `TRUE` gives the replacement. Some instances do see repeated vales.

Next we do the `t.test`

```
t.test(x,y)
```

```
##
##  Welch Two Sample t-test
##
## data:  x and y
## t = 0.33531, df = 17.805, p-value = 0.7413
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -6.324578  8.724578
## sample estimates:
## mean of x mean of y
##       3.7       2.5
```

We fail to reject the null hypothesis here. There is no evidence to suggest that the mean values are different.

Next we explore the `order` function.

```
order(x)
```

```
##  [1]  9  4  5  8  1  3 10  6  2  7
```

We see this gives the order of the elements of x, indexing at 1 as the lowest value. To sort x we could do the following.

```
sort(x)
```

```
##  [1] -10  -5   0   1   3   6   7   8  12  15
```

We could also use the order function as follows:

```
x[order(x)]
```

```
##  [1] -10  -5   0   1   3   6   7   8  12  15
```

Inside the [] we are giving the index of the value we want. So this will return the values in the proper order. Lastly we will preform the paired t.test.

```
t.test(sort(x),y,paired = TRUE)
```

```
##
```

```
##  Paired t-test
##
## data:  sort(x) and y
## t = 2.164, df = 9, p-value = 0.05868
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  -0.05440584  2.45440584
## sample estimates:
## mean difference
##             1.2
```

The result here is still not significant (for p =0.05) but is much closer than in the non-paired data. I am actually quite surprised at that result but since y is build off of x and now they are both sequential I could see why they might be statistically equivalent on average.

A logical test for negativity is simply

```
x>0
```

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

Since this gives the Boolean, we can use that as the index for x and overwrite x

```
x <- x[x>0]
x
```

```
## [1]  3 12  6  8 15  1  7
```

Using R: Some Missing Values

```
col1 <- c(1,2,3,NA,5)
col2 <- c(4,5,6,89,101)
col3 <- c(45,NA,66,121,201)
col4 <- c(14,NA,13,NA,27)
X <- rbind (col1,col2,col3,col4)

X
```

```
##      [,1] [,2] [,3] [,4] [,5]
## col1    1    2    3   NA    5
## col2    4    5    6   89  101
## col3   45   NA   66  121  201
## col4   14   NA   13   NA   27
```

So we see X has NA in three rows. We can find the NAs with the following

```
is.na(X)
```

```
##       [,1]  [,2]  [,3]  [,4]  [,5]
## col1 FALSE FALSE FALSE  TRUE FALSE
## col2 FALSE FALSE FALSE FALSE FALSE
## col3 FALSE  TRUE FALSE FALSE FALSE
## col4 FALSE  TRUE FALSE  TRUE FALSE
```

To get to which rows have the NAs, we sum across the booleans and ask that the sum in that row is larger than 0. Then we use the rownames command to give out those rows names that do have some NAs.

```
rownames(X)[rowSums(is.na(X))>0]
```

```
## [1] "col1" "col3" "col4"
```

For the next piece, we define y

```
y <- c(3,12,99,99,7,99,21)
y
```

```
## [1]  3 12 99 99  7 99 21
```

We will find the 99s with this peice of code

```
y == 99
```

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

We set that to the NA value with this which overwrites y values.

```
y[y==99] = NA

y
```

```
## [1]  3 12 NA NA  7 NA 21
```

I count the NA values with a sum of the booleans

```
sum(is.na(y))
```

```
## [1] 3
```

##Using R: IDE

Here I have read the data in. I utilize the head command to display the first 6 rows.

```
college = read.csv('college.csv')
head(college)
```

```
##                              X Private Apps Accept Enroll Top10perc Top25perc
## 1 Abilene Christian University     Yes 1660   1232    721        23        52
## 2           Adelphi University     Yes 2186   1924    512        16        29
## 3               Adrian College     Yes 1428   1097    336        22        50
## 4          Agnes Scott College     Yes  417    349    137        60        89
## 5    Alaska Pacific University     Yes  193    146     55        16        44
## 6            Albertson College     Yes  587    479    158        38        62
##   F.Undergrad P.Undergrad Outstate Room.Board Books Personal PhD Terminal
## 1        2885         537     7440       3300   450     2200  70       78
## 2        2683        1227    12280       6450   750     1500  29       30
## 3        1036          99    11250       3750   400     1165  53       66
## 4         510          63    12960       5450   450      875  92       97
## 5         249         869     7560       4120   800     1500  76       72
## 6         678          41    13500       3335   500      675  67       73
##   S.F.Ratio perc.alumni Expend Grad.Rate
## 1      18.1          12   7041        60
## 2      12.2          16  10527        56
## 3      12.9          30   8735        54
## 4       7.7          37  19016        59
## 5      11.9           2  10922        15
## 6       9.4          11   9727        55
```

Next, I change the rownames to the university name and delete that column.

```
rownames (college) <- college [,1]
college <- college [,-1]
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
```

Next we examine some stats on the data

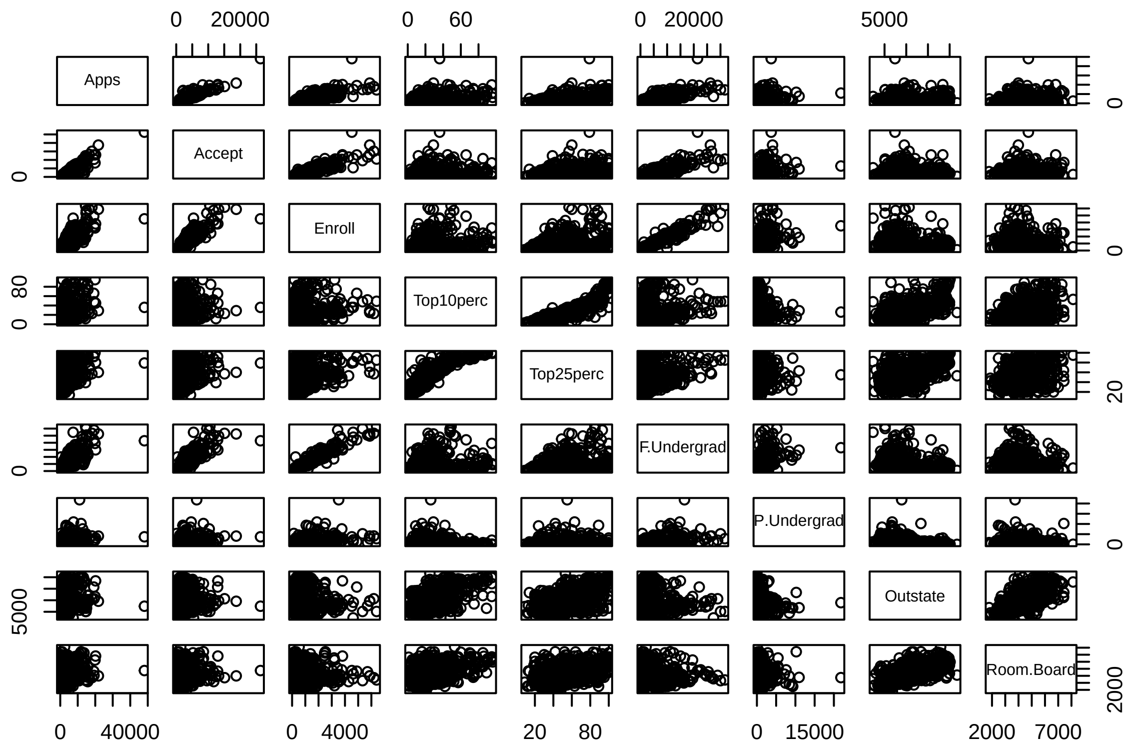
```r
summary(college)
```

```
##    Private               Apps           Accept          Enroll
##  Length:777         Min.   :   81   Min.   :   72   Min.   :  35
##  Class :character   1st Qu.:  776   1st Qu.:  604   1st Qu.: 242
##  Mode  :character   Median : 1558   Median : 1110   Median : 434
##                     Mean   : 3002   Mean   : 2019   Mean   : 780
##                     3rd Qu.: 3624   3rd Qu.: 2424   3rd Qu.: 902
##                     Max.   :48094   Max.   :26330   Max.   :6392
##    Top10perc       Top25perc      F.Undergrad     P.Undergrad
##  Min.   : 1.00   Min.   :  9.0   Min.   :  139   Min.   :    1.0
##  1st Qu.:15.00   1st Qu.: 41.0   1st Qu.:  992   1st Qu.:   95.0
##  Median :23.00   Median : 54.0   Median : 1707   Median :  353.0
##  Mean   :27.56   Mean   : 55.8   Mean   : 3700   Mean   :  855.3
##  3rd Qu.:35.00   3rd Qu.: 69.0   3rd Qu.: 4005   3rd Qu.:  967.0
##  Max.   :96.00   Max.   :100.0   Max.   :31643   Max.   :21836.0
##     Outstate       Room.Board       Books           Personal
##  Min.   : 2340   Min.   :1780   Min.   :  96.0   Min.   : 250
##  1st Qu.: 7320   1st Qu.:3597   1st Qu.: 470.0   1st Qu.: 850
##  Median : 9990   Median :4200   Median : 500.0   Median :1200
##  Mean   :10441   Mean   :4358   Mean   : 549.4   Mean   :1341
##  3rd Qu.:12925   3rd Qu.:5050   3rd Qu.: 600.0   3rd Qu.:1700
##  Max.   :21700   Max.   :8124   Max.   :2340.0   Max.   :6800
##       PhD            Terminal       S.F.Ratio      perc.alumni
```

```
##  Min.   :  8.00   Min.   : 24.0   Min.   : 2.50   Min.   : 0.00  
##  1st Qu.: 62.00   1st Qu.: 71.0   1st Qu.:11.50   1st Qu.:13.00  
##  Median : 75.00   Median : 82.0   Median :13.60   Median :21.00  
##  Mean   : 72.66   Mean   : 79.7   Mean   :14.09   Mean   :22.74  
##  3rd Qu.: 85.00   3rd Qu.: 92.0   3rd Qu.:16.50   3rd Qu.:31.00  
##  Max.   :103.00   Max.   :100.0   Max.   :39.80   Max.   :64.00  
##      Expend        Grad.Rate     
##  Min.   : 3186   Min.   : 10.00  
##  1st Qu.: 6751   1st Qu.: 53.00  
##  Median : 8377   Median : 65.00  
##  Mean   : 9660   Mean   : 65.46  
##  3rd Qu.:10830   3rd Qu.: 78.00  
##  Max.   :56233   Max.   :118.00
```

I am not familiar with the `pairs` command but here goes

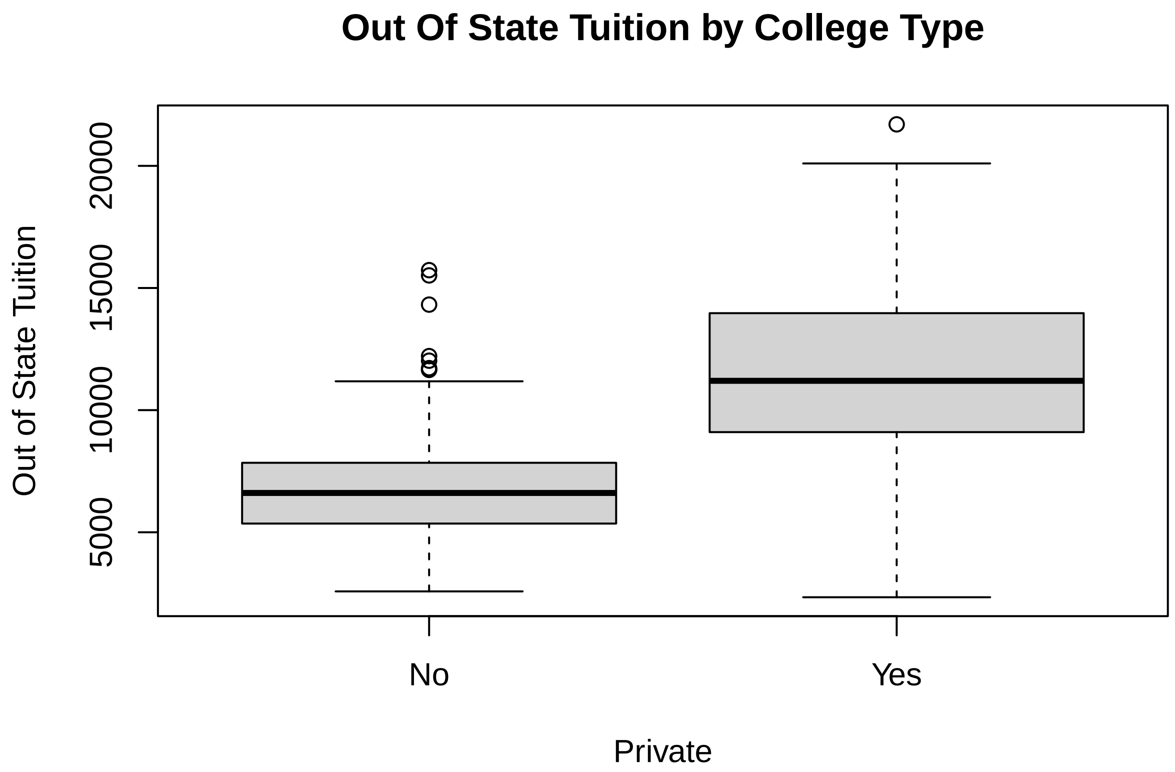
```r
pairs(college[,2:10])
```



That is a nice graphic although a bit too small for my tastes. I hope it compiles correctly in the pdf...

Next I'll create the boxplot for out of state tution vs the public or private.

```r
boxplot(Outstate ~ Private, data = college, main = "Out Of State Tuition by College Type", ylab = "Out
```



This looks fine although I do prefer `ggplot2`.

Next I comment the code as requested

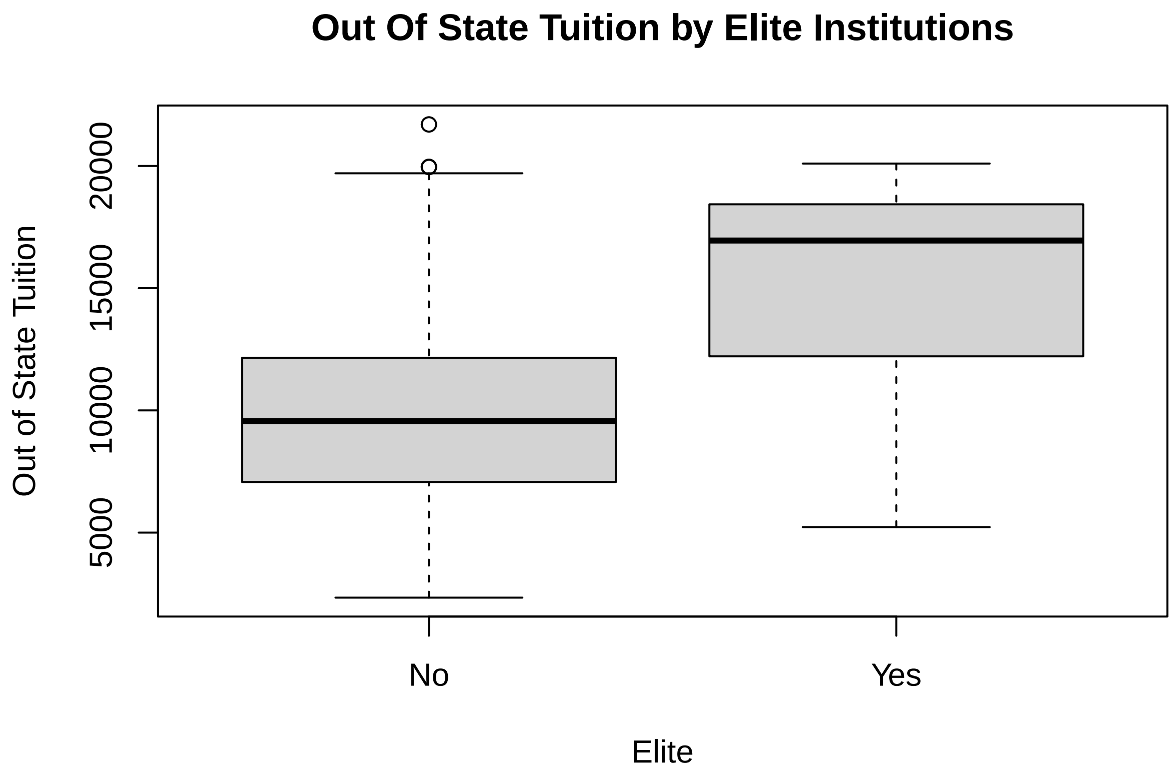
```
Elite <- rep ("No", nrow(college )) #This creates a vector that full of No that is the same width as th
Elite [college$Top10perc >50] <- "Yes" #this changes some of the nos to yes if the top10 is more than 5
Elite <- as.factor (Elite) #this casts the vector as a factor vector.  This is useful in that Elite now
college <- data.frame(college ,Elite) #this adds the column to the original dataframe and saves it

summary(Elite)
```

```
##  No Yes
## 699  78
```

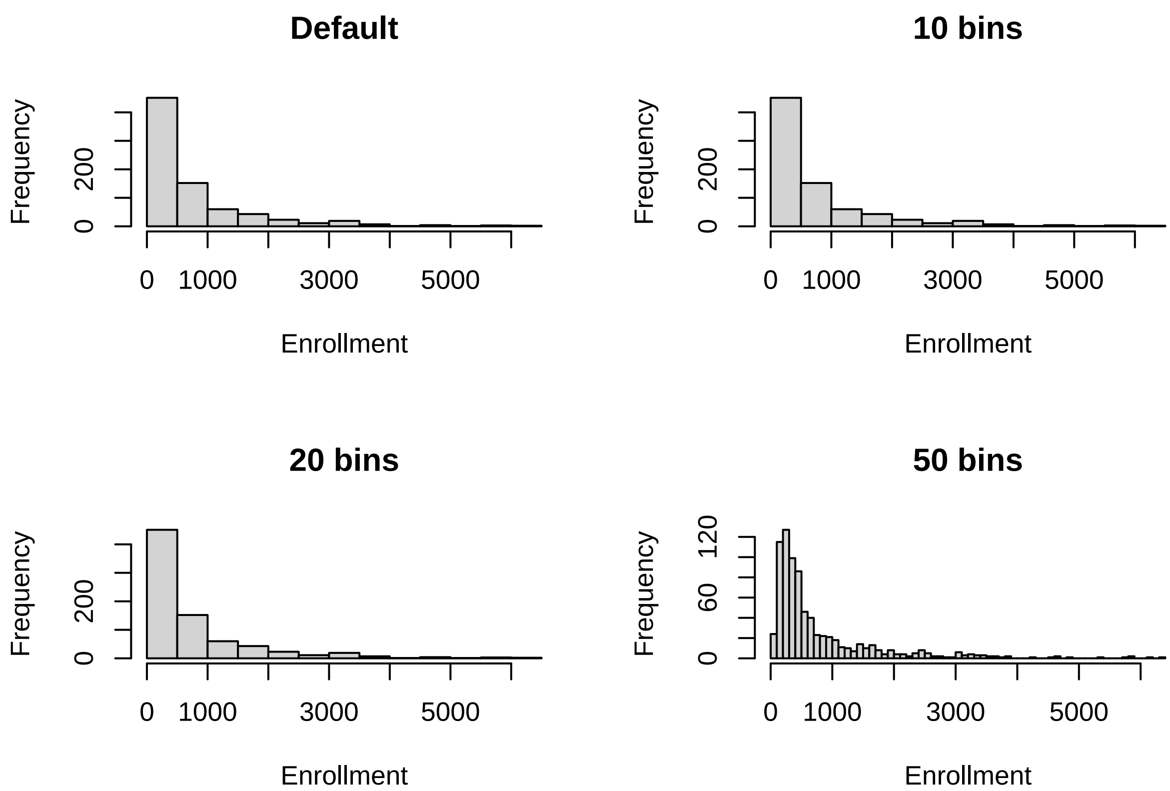
It appears that there are 78 elite universities. Let's explore tutions with this new factor

```
boxplot(Outstate ~ Elite, data = college, main = "Out Of State Tuition by Elite Institutions", ylab = "(
```



Next we look at a few histograms with differing number of bins.

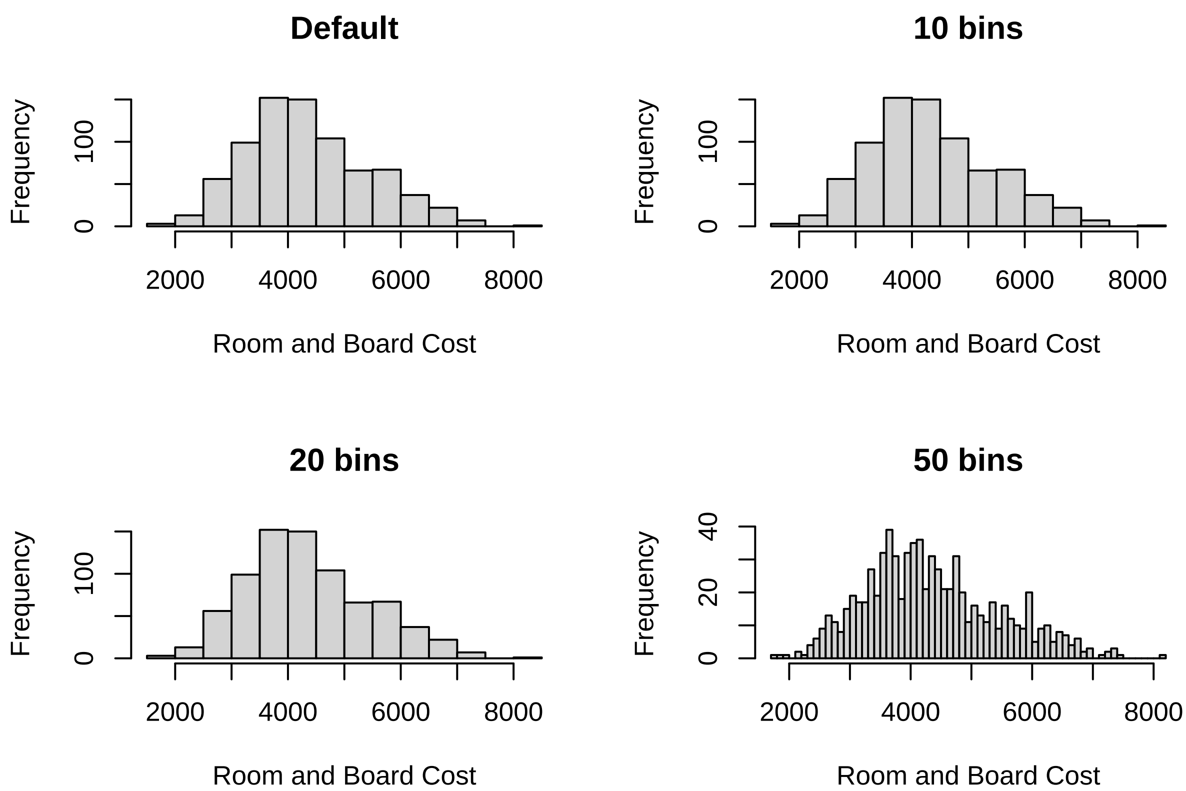
```r
par(mfrow=c(2,2))
hist(college[,'Enroll'], main = "Default", xlab = "Enrollment")
hist(college[,'Enroll'], main = "10 bins",breaks = 10, xlab = "Enrollment")
hist(college[,'Enroll'], main = "20 bins",breaks = 20, xlab = "Enrollment")
hist(college[,'Enroll'], main = "50 bins",breaks = 50, xlab = "Enrollment")
```



I don't see much difference between the default, 10 nor 20. The 50 does look a bit different.

Again just to try it once more

```
par(mfrow=c(2,2))
hist(college[,'Room.Board'], main = "Default", xlab = "Room and Board Cost")
hist(college[,'Room.Board'], main = "10 bins",breaks = 10, xlab = "Room and Board Cost")
hist(college[,'Room.Board'], main = "20 bins",breaks = 20, xlab = "Room and Board Cost")
hist(college[,'Room.Board'], main = "50 bins",breaks = 50, xlab = "Room and Board Cost")
```



It kind of looks like more than 10 breaks Maybe the default overrides that option if you set it too low…

Using R: Manipulating Data in Data Frames

First, I'll load some data directly from a package. This `baseball` data comes from the `plyr` package loaded earlier.

```
head(baseball)
```

```
##            id year stint team lg  g  ab  r  h X2b X3b hr rbi sb cs bb so ibb
## 4   ansonca01 1871     1  RC1    25 120 29 39  11   3  0  16  6  2  2  1  NA
## 44  forceda01 1871     1  WS3    32 162 45 45   9   4  0  29  8  0  4  0  NA
## 68  mathebo01 1871     1  FW1    19  89 15 24   3   1  0  10  2  1  2  0  NA
## 99  startjo01 1871     1  NY2    33 161 35 58   5   1  1  34  4  2  3  0  NA
## 102 suttoez01 1871     1  CL1    29 128 35 45   3   7  3  23  3  1  1  0  NA
## 106 whitede01 1871     1  CL1    29 146 40 47   6   5  1  21  2  2  4  1  NA
##     hbp sh sf gidp
## 4    NA NA NA   NA
## 44   NA NA NA   NA
## 68   NA NA NA   NA
## 99   NA NA NA   NA
## 102  NA NA NA   NA
## 106  NA NA NA   NA
```

Lots of baseball data!

```
baseball[baseball$year<1954,'sf'] = 0 #set all sf before 1954 to 0
baseball[is.na(baseball$hbp),'hbp'] = 0 #set all null values for hit by pitch to 0
```

```r
baseball <- baseball[baseball$ab>=50,]
```

Now that the data is clean, we will apply the obp formula of

$$obp = \frac{h + bb + hbp}{ab + bb + hpb + sf}$$

```r
baseball <- mutate(baseball, obp = (h+bb+hbp)/(ab+bb+hbp+sf))

head(baseball)
```

```
##            id year stint team lg  g  ab  r  h X2b X3b hr rbi sb cs bb so ibb
## 4   ansonca01 1871     1  RC1    25 120 29 39  11   3  0  16  6  2  2  1  NA
## 44  forceda01 1871     1  WS3    32 162 45 45   9   4  0  29  8  0  4  0  NA
## 68  mathebo01 1871     1  FW1    19  89 15 24   3   1  0  10  2  1  2  0  NA
## 99  startjo01 1871     1  NY2    33 161 35 58   5   1  1  34  4  2  3  0  NA
## 102 suttoez01 1871     1  CL1    29 128 35 45   3   7  3  23  3  1  1  0  NA
## 106 whitede01 1871     1  CL1    29 146 40 47   6   5  1  21  2  2  4  1  NA
##     hbp sh sf gidp       obp
## 4     0 NA  0   NA 0.3360656
## 44    0 NA  0   NA 0.2951807
## 68    0 NA  0   NA 0.2857143
## 99    0 NA  0   NA 0.3719512
## 102   0 NA  0   NA 0.3565891
## 106   0 NA  0   NA 0.3400000
```

Now that we have that info added, let's find the top five players for obp of all time.

```r
arrange(baseball, -obp)[1:5,c('year','id','obp')] #I get the top records with 1:5, restrict on to the c
```

```
##   year        id       obp
## 1 2004 bondsba01 0.6094003
## 2 2002 bondsba01 0.5816993
## 3 1941 willite01 0.5528053
## 4 1899 mcgrajo01 0.5474860
## 5 1923  ruthba01 0.5445402
```

We see here Barry Bonds (from the 'roids era twice), Ted Williams(a year he hit .400), John McGraw (a player I was not familiar with though he did have a season with my home team Cardinals in 1900) and the babe himself Babe Ruth.

##Using R: `aggregate()` Function

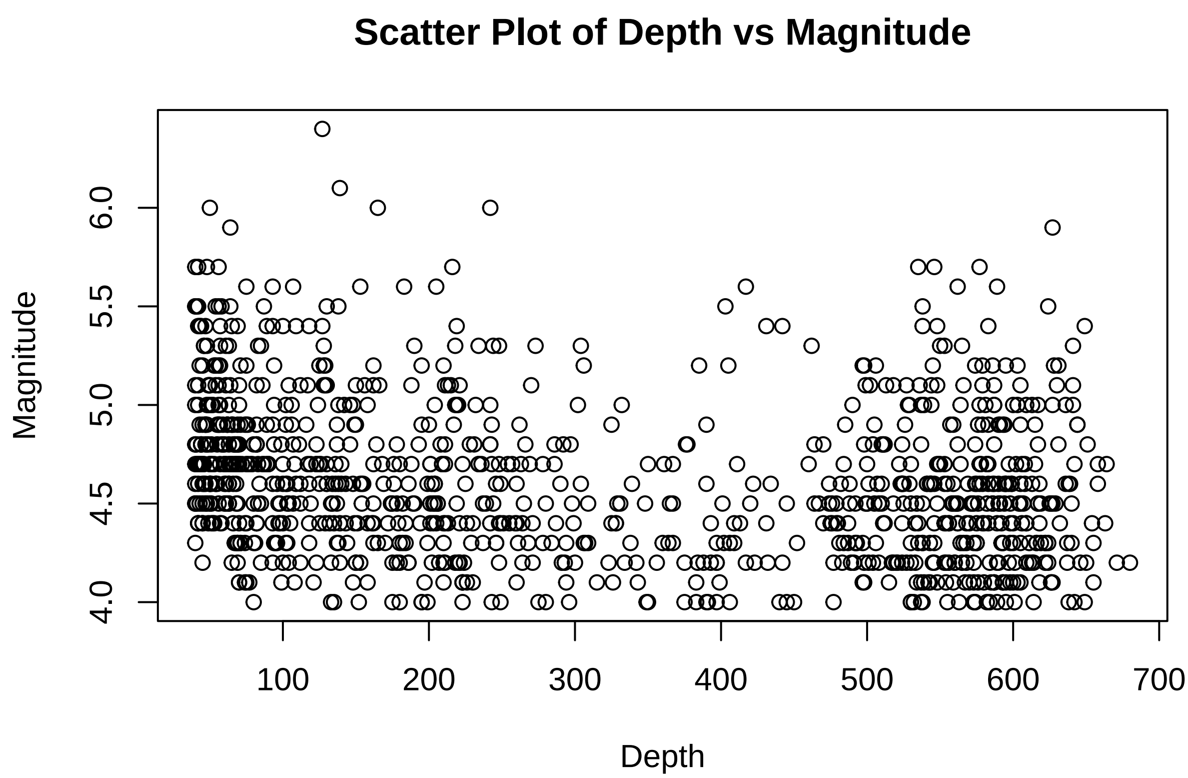
I am going to grab the `quakes` dataset.

```r
head(quakes)
```

```
##      lat   long depth mag stations
## 1 -20.42 181.62   562 4.8       41
## 2 -20.62 181.03   650 4.2       15
## 3 -26.00 184.10    42 5.4       43
## 4 -17.97 181.66   626 4.1       19
## 5 -20.42 181.96   649 4.0       11
## 6 -19.68 184.31   195 4.0       12
```

Next we will examine magnitude versus depth with a scatter plot.

```r
plot(quakes$depth,quakes$mag, xlab = 'Depth', ylab = 'Magnitude', main = 'Scatter Plot of Depth vs Magn
```



Next we will aggregate the data to look at the average depth for each of the magnitude levels

```
quakeAvgDepth = aggregate(quakes$depth, list(mag = quakes$mag), mean)
```

Not too bad when you follow the example in the help menu.

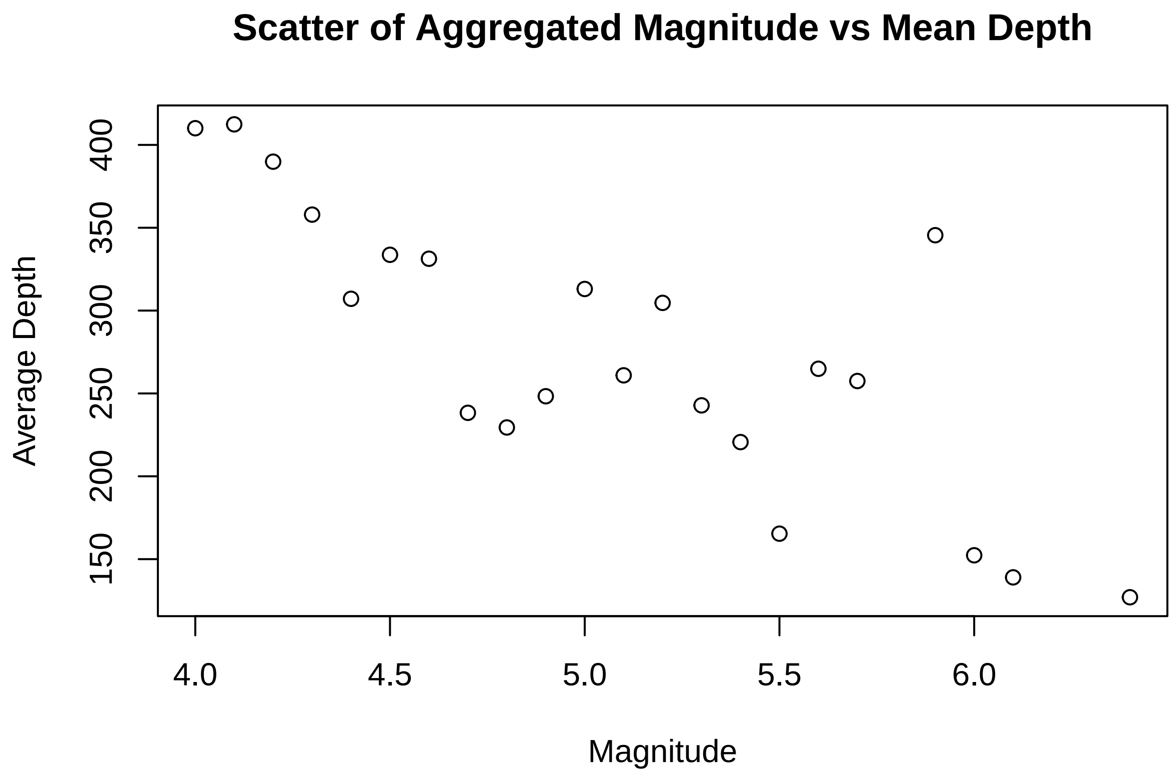
Next I rename the dataframe to have useful column names and print it to see the nice output.

```
colnames(quakeAvgDepth) = c('mag','meanDepth')
head(quakeAvgDepth)
```

```
##   mag meanDepth
## 1 4.0  410.0652
## 2 4.1  412.4000
## 3 4.2  389.8778
## 4 4.3  357.9294
## 5 4.4  307.1188
## 6 4.5  333.6729
```

Now we plot again to see if there is a relationship in the aggregate

```
plot(quakeAvgDepth$mag,quakeAvgDepth$meanDepth, xlab = 'Magnitude', ylab = 'Average Depth',main = 'Scat
```



There clearly appears to be a relationship here. It was not as obvious in the full data case but the relationship appears in the aggregate. I do question a bit of this methodology though. We are aggregating a continuous variable that has been truncated to two decimals. Richter scale (magnitude) is a famous example of a logarithmic scale so small rounding errors are amplified in varying degrees as you increase the scale. While yes, I believe there is a relationship, I'd be worried about generalizing too far based on this data.