CIND 123 SUMMER 2018 - Assignment #3

Paul Ycay

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

Use RStudio for this assignment. Edit the file `assignment-4.Rmd` and insert your R code where wherever you see the string "INSERT YOUR ANSWER HERE"

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document.

Sample Question and Solution

Use `seq()` to create the vector (2,4,6, ... ,20).

```
#Insert your code here.
seq(2,20,by = 2)
```

```
##  [1]  2  4  6  8 10 12 14 16 18 20
```

In this assignment, questions 1 - 4 make use of data that is provided by the `mosaic` package. (install mosaic package and load KidsFeet using data(KidsFeet)).

```
install.packages('mosaic',repos = "http://cran.us.r-project.org")
```

```
## Installing package into 'D:/Users/pycay/Documents/R/win-library/3.4'
## (as 'lib' is unspecified)

## package 'mosaic' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  D:\Users\pycay\AppData\Local\Temp\RtmpAT3XQd\downloaded_packages
```

```
library(mosaic)
```

```
## Warning: package 'mosaic' was built under R version 3.4.4

## Loading required package: dplyr

## Warning: package 'dplyr' was built under R version 3.4.4

##
## Attaching package: 'dplyr'

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

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

## Loading required package: lattice

## Loading required package: ggformula

## Warning: package 'ggformula' was built under R version 3.4.4

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.4.4

##
## New to ggformula?  Try the tutorials:
##  learnr::run_tutorial("introduction", package = "ggformula")
##  learnr::run_tutorial("refining", package = "ggformula")

## Loading required package: mosaicData

## Warning: package 'mosaicData' was built under R version 3.4.4

## Loading required package: Matrix

##
## The 'mosaic' package masks several functions from core packages in order
to add
## additional features.  The original behavior of these functions should not
be affected by this.
##
## Note: If you use the Matrix package, be sure to load it BEFORE loading
mosaic.

##
## Attaching package: 'mosaic'

## The following object is masked from 'package:Matrix':
##
##     mean

## The following object is masked from 'package:ggplot2':
##
##     stat

## The following objects are masked from 'package:dplyr':
##
##     count, do, tally

## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cor.test, cov, fivenum, IQR, median,
##     prop.test, quantile, sd, t.test, var
```

```
## The following objects are masked from 'package:base':
##
##     max, mean, min, prod, range, sample, sum

data(KidsFeet)
```

Question 1 - 30%

This question makes use of package "plm", and load Crime dataset as following:

```
install.packages("plm",repos = "http://cran.us.r-project.org")

## Installing package into 'D:/Users/pycay/Documents/R/win-library/3.4'
## (as 'lib' is unspecified)

## package 'plm' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  D:\Users\pycay\AppData\Local\Temp\RtmpAT3XQd\downloaded_packages

library(plm)

## Warning: package 'plm' was built under R version 3.4.4

## Loading required package: Formula

## Warning: package 'Formula' was built under R version 3.4.4

##
## Attaching package: 'plm'

## The following object is masked from 'package:mosaic':
##
##     r.squared

## The following objects are masked from 'package:dplyr':
##
##     between, lag, lead

data(Crime)
```

a) Display the first 8 rows of crime and make note of all the variables and print all column (variable) names. Display summary of each variable.

```
head(Crime,8)

##   county year    crmrte   prbarr  prbconv  prbpris avgsen     polpc
## 1      1   81 0.0398849 0.289696 0.402062 0.472222   5.61 0.0017868
## 2      1   82 0.0383449 0.338111 0.433005 0.506993   5.59 0.0017666
## 3      1   83 0.0303048 0.330449 0.525703 0.479705   5.80 0.0018358
## 4      1   84 0.0347259 0.362525 0.604706 0.520104   6.89 0.0018859
## 5      1   85 0.0365730 0.325395 0.578723 0.497059   6.55 0.0019244
## 6      1   86 0.0347524 0.326062 0.512324 0.439863   6.90 0.0018952
## 7      1   87 0.0356036 0.298270 0.527596 0.436170   6.71 0.0018279
```

```
## 8      3   81 0.0163921 0.202899 0.869048 0.465753   8.45 0.0005939
##    density    taxpc  region smsa   pctmin     wcon      wtuc     wtrd
## 1 2.307159 25.69763 central   no 20.21870 206.4803  333.6209 182.3330
## 2 2.330254 24.87425 central   no 20.21870 212.7542  369.2964 189.5414
## 3 2.341801 26.45144 central   no 20.21870 219.7802 1394.8030 196.6395
## 4 2.346420 26.84235 central   no 20.21870 223.4238  398.8604 200.5629
## 5 2.364896 28.14034 central   no 20.21870 243.7562  358.7830 206.8827
## 6 2.385681 29.74098 central   no 20.21870 257.9139  369.5465 218.5165
## 7 2.422633 30.99368 central   no 20.21870 281.4259  408.7245 221.2701
## 8 0.976834 14.56088 central   no  7.91632 188.7683  292.6422 151.4234
##       wfir     wser   wmfg   wfed   wsta   wloc       mix   pctymle
## 1 272.4492 215.7335 229.12 409.37 236.24 231.47 0.0999179 0.0876968
## 2 300.8788 231.5767 240.33 419.70 253.88 236.79 0.1030491 0.0863767
## 3 309.9696 240.1568 269.70 438.85 250.36 248.58 0.0806787 0.0850909
## 4 350.0863 252.4477 281.74 459.17 261.93 264.38 0.0785035 0.0838333
## 5 383.0707 261.0861 298.88 490.43 281.44 288.58 0.0932486 0.0823065
## 6 409.8842 269.6129 322.65 478.67 286.91 306.70 0.0973228 0.0800806
## 7 453.1722 274.1775 334.54 477.58 292.09 311.91 0.0801688 0.0778710
## 8 202.4292 191.3742 210.75 381.72 247.38 213.17 0.0561224 0.0870046
```

```
summary(head(Crime,8))
```

```
##      county          year           crmrte            prbarr
##  Min.   :1.00   Min.   :81.00   Min.   :0.01639   Min.   :0.2029
##  1st Qu.:1.00   1st Qu.:81.75   1st Qu.:0.03362   1st Qu.:0.2961
##  Median :1.00   Median :83.50   Median :0.03518   Median :0.3257
##  Mean   :1.25   Mean   :83.62   Mean   :0.03332   Mean   :0.3092
##  3rd Qu.:1.00   3rd Qu.:85.25   3rd Qu.:0.03702   3rd Qu.:0.3324
##  Max.   :3.00   Max.   :87.00   Max.   :0.03988   Max.   :0.3625
##     prbconv          prbpris          avgsen          polpc
##  Min.   :0.4021   Min.   :0.4362   Min.   :5.590   Min.   :0.0005939
##  1st Qu.:0.4925   1st Qu.:0.4593   1st Qu.:5.753   1st Qu.:0.0017817
##  Median :0.5266   Median :0.4760   Median :6.630   Median :0.0018319
##  Mean   :0.5566   Mean   :0.4772   Mean   :6.562   Mean   :0.0016896
##  3rd Qu.:0.5852   3rd Qu.:0.4995   3rd Qu.:6.893   3rd Qu.:0.0018882
##  Max.   :0.8690   Max.   :0.5201   Max.   :8.450   Max.   :0.0019244
##     density           taxpc           region   smsa       pctmin
##  Min.   :0.9768   Min.   :14.56   other  :0   no :8   Min.   : 7.916
##  1st Qu.:2.3245   1st Qu.:25.49   west   :0   yes:0   1st Qu.:20.219
##  Median :2.3441   Median :26.65   central:8           Median :20.219
##  Mean   :2.1845   Mean   :25.91                       Mean   :18.681
##  3rd Qu.:2.3701   3rd Qu.:28.54                       3rd Qu.:20.219
##  Max.   :2.4226   Max.   :30.99                       Max.   :20.219
##       wcon            wtuc             wtrd            wfir
##  Min.   :188.8   Min.   : 292.6   Min.   :151.4   Min.   :202.4
##  1st Qu.:211.2   1st Qu.: 352.5   1st Qu.:187.7   1st Qu.:293.8
##  Median :221.6   Median : 369.4   Median :198.6   Median :330.0
##  Mean   :229.3   Mean   : 490.8   Mean   :195.9   Mean   :335.2
##  3rd Qu.:247.3   3rd Qu.: 401.3   3rd Qu.:209.8   3rd Qu.:389.8
##  Max.   :281.4   Max.   :1394.8   Max.   :221.3   Max.   :453.2
```

```
##       wser            wmfg            wfed            wsta      
##  Min.   :191.4   Min.   :210.8   Min.   :381.7   Min.   :236.2  
##  1st Qu.:227.6   1st Qu.:237.5   1st Qu.:417.1   1st Qu.:249.6  
##  Median :246.3   Median :275.7   Median :449.0   Median :257.9  
##  Mean   :242.0   Mean   :273.5   Mean   :444.4   Mean   :263.8  
##  3rd Qu.:263.2   3rd Qu.:304.8   3rd Qu.:477.9   3rd Qu.:282.8  
##  Max.   :274.2   Max.   :334.5   Max.   :490.4   Max.   :292.1  
##       wloc            mix            pctymle       
##  Min.   :213.2   Min.   :0.05612   Min.   :0.07787  
##  1st Qu.:235.5   1st Qu.:0.07975   1st Qu.:0.08175  
##  Median :256.5   Median :0.08696   Median :0.08446  
##  Mean   :262.7   Mean   :0.08613   Mean   :0.08378  
##  3rd Qu.:293.1   3rd Qu.:0.09797   3rd Qu.:0.08653  
##  Max.   :311.9   Max.   :0.10305   Max.   :0.08770
```

b) Calculate the mean,variance and standard deviation of tax revenue per capita (taxpc) by omitting the missing values, if any.

```
mean(Crime$taxpc,na.rm=TRUE)

## [1] 30.23919

var(Crime$taxpc,na.rm=TRUE)

## [1] 131.21

sd(Crime$taxpc,na.rm=TRUE)

## [1] 11.4547
```

c) Use `density` and `smsa` variables build a multiple linear regression model to predict tax per capita (taxpc), display a summary of your model indicating Residuals, Coefficients..etc. What can you say about your model?

```
model<-lm(taxpc~density+smsa,data=Crime)
summary(model)

##
## Call:
## lm(formula = taxpc ~ density + smsa, data = Crime)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -16.960  -6.693  -2.083   3.173  90.320 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  29.5615     0.7134  41.436  < 2e-16 ***
## density      -0.2345     0.5329  -0.440     0.66    
## smsayes      11.2808     2.6939   4.188 3.22e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 11.09 on 627 degrees of freedom
## Multiple R-squared:  0.06603,    Adjusted R-squared:  0.06305
## F-statistic: 22.16 on 2 and 627 DF,  p-value: 5.011e-10
```

d)Based on the output of your model, write the equations based on the intercept and factors of `smsa` when `density` is set to 2.4. and compare the result with `predict()` function. Hint: Explore `predict()` function

```
smsaYES<-29.5615-0.2345*2.4+11.280*1
smsaYES

## [1] 40.2787

smsaNO<-29.5615-0.2345*2.4
smsaNO

## [1] 28.9987

dfyes<-data.frame(density=c(2.4),smsa=c('yes'))
predict(model,dfyes)

##        1
## 40.27948

dfno<-data.frame(density=c(2.4),smsa=c('no'))
predict(model,dfno)

##        1
## 28.99864

#the results are very close to each other
```

e)Find Pearson correlation between density and tax per capita; between density and police per capita (polpc) Please comment on the result with a sentence.

```
cor_density_tax<-cor(Crime$density,Crime$taxpc)
cor_density_tax

## [1] 0.1997634

cor_density_police<-cor(Crime$density,Crime$polpc)
cor_density_police

## [1] -0.03969574

#density vs. tax has a weak but positive correlation, dnsity vs police has a
weak but negative correlation
```

f)Write the correlation matrix of the variables: avgsen, polpc, density, taxpc. Hint: Explore the variables by ?Crime. Comment on the result with a sentence.

```
cor(Crime[,7:10])
```

```
##             avgsen       polpc     density      taxpc
## avgsen  1.00000000  0.01712970  0.07807510 0.02818939
## polpc   0.01712970  1.00000000 -0.03969574 0.10828664
## density 0.07807510 -0.03969574  1.00000000 0.19976339
## taxpc   0.02818939  0.10828664  0.19976339 1.00000000
```

Question 2 -30%

a) First and second midterm grades of some students are given as c(55,76,48,58,80,75,32,22) and c(85,76,78,88,90,95,42,31). Set R variables `first` and `second` respectively.

```
first <- c(85,76,78,88,90,95,42,31)
second <- c(55,76,48,58,80,75,32,22)
```

b) Apply the lm() function to observe the relationship between the first and the second midterm grades. Hint: Second midterm is the response variable.

```
model2 <- lm(second~first)
summary(model2)
```

```
##
## Call:
## lm(formula = second ~ first)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.5445  -9.4942  -0.2429   4.4465  18.0122
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.1669    13.8930  -0.084  0.93579
## first         0.7784     0.1819   4.280  0.00521 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.36 on 6 degrees of freedom
## Multiple R-squared:  0.7532, Adjusted R-squared:  0.7121
## F-statistic: 18.31 on 1 and 6 DF,  p-value: 0.005208
```

c) Find the second midterm grade of a student given that his/her first midterm grade is 72. Print the result by using print() function.

```
result <-  predict(model2,data.frame(first = 72))
print(result)
```

```
##        1
## 54.87436
```

Question 3 - 40%

π appears in the formula for the standard normal distribution, the most important probability distribution in statistics. Why not give it a try to calculate π using statistics! In fact, you'll use a simulation technique called the *Monte Carlo Method*.

Recall that the area of a circle of radius r is $A = \pi r^2$. Therefore the area of a circle of radius 1, aka a *unit circle*, is π. You'll compute an approximation to the area of this circle using the Monte Carlo Method.

a) The Monte Carlo Method uses random numbers to simulate some process. Here the process is throwing darts at a square. Assume the darts are uniformly distributed over the square. Imagine a unit circle enclosed by a square whose sides are of length 2. Set an R variable `area.square` to be the area of a square whose sides are of length 2.

```
area.square <- 2*2
area.square

## [1] 4
```

b) The points of the square can be given x-y coordinates. Let both x and y range from -1 to +1 so that the square is centred on the origin of the coordinate system. Throw some darts at the square by generating random numeric vectors x and y, each of length `N = 10,000`. Set R variables `x` and `y` each to be uniformly distributed random numbers in the range -1 to +1. (hint: runif() generates random number for the uniform distribution)

```
N  <- 10000
x <- runif(N, min=-1, max=1)
y <- runif(N, min=-1, max=1)
```

c) Now count how many darts landed inside the unit circle. Recall that a point is inside the unit circle when $x^2 + y^2 < 1$. Save the result of successful hits in a variable named hit. (hint: a for loop over the length of x and y is one option to reach hit)

```
hit <- 0
for (i in 1:N){
  if((x[i]^2+y[i]^2)<=1)
    hit<-hit+1}
print (hit)

## [1] 7871
```

d) The probability that a dart hits inside the circle is proportional to the ratio of the area of the circle to the area of the square. Use this fact to calculate an approximation to π and print the result

```
pi <- (4*hit)/N
print(pi)

## [1] 3.1484
```

Wow you got the first estimate for pi π, congratulations you have completed the first run of the Monte Carlo simulation. If there is further interest put all the above logic in a function, and call it 50 times store the results in a vector called pi then take the mean of pi vector.