CIND 123 Winter 2018 - Assignment #4

Brendan Dagys

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 .

#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')  
library(mosaic)

## Loading required package: dplyr

##   
## 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

## Loading required package: ggplot2

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

## Loading required package: mosaicData

## 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 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")  
library(plm)

## Loading required package: Formula

##   
## 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)

1. Display the first 10 rows of crime and make note of all the variables.

head(Crime, 10)

## 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  
## 9 3 82 0.0190651 0.162218 0.772152 0.377049 5.71 0.0007047  
## 10 3 83 0.0151492 0.181586 1.028170 0.438356 8.69 0.0006587  
## 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  
## 9 0.992278 35.64073 central no 7.91632 186.9658 345.7217 156.8826  
## 10 1.003861 19.26188 central no 7.91632 193.5983 604.9115 157.1295  
## 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  
## 9 225.0409 208.8190 217.77 386.42 374.07 219.18 0.0473118 0.0864722  
## 10 248.1390 219.0847 236.64 382.65 268.90 223.06 0.0596206 0.0859426

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

Crime$taxpc = na.omit(Crime$taxpc)  
cat('The mean is:', mean(Crime$taxpc))

## The mean is: 30.23919

cat('\n\nThe variance is:', var(Crime$taxpc))

##   
##   
## The variance is: 131.21

cat('\n\nThe standard deviation is:', sd(Crime$taxpc))

##   
##   
## The standard deviation is: 11.4547

1. Use density and smsa to predict tax per capita and build a univariate linear regression model, display a summary of your model indicating Residuals, Coefficients..etc. What can you say about your model?

model1 = with(Crime, lm(taxpc ~ density + smsa))  
summary(model1)

##   
## Call:  
## lm(formula = taxpc ~ density + smsa)  
##   
## 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

#It appears that the intercept is equal to ~29.5, and the coefficients for 'density' and 'smsa' (with a value of 'yes') are -0.2345 and 11.2808, respectively. This indicates that as the density of people per square mile increases, the tax revenue per capita slightly decreases. As well, the coefficient of 11.2808 for 'smsayes' (standard metropolitan statistical area) shows a positive relationship between 'smsa' and the 'taxpc' value. The p-values indicate that the result for 'density' are not very significant (0.66), and the result for 'smsayes' is very significant (3.22e-05).  
  
#Finally, the adjusted R-squared value is 0.06305 (or 6%). This is very low, and indicates that the regression model explains a small percentage of the variation of the response data.

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

predictedNo = 29.5615 - 0.2345\*(2.4)  
predictedYes = 29.5615 - 0.2345\*(2.4) + 11.2808  
  
cat('Calculated average prediction with \'smsa\' = \'no\':', predictedNo)

## Calculated average prediction with 'smsa' = 'no': 28.9987

cat('\n\nCalculated average prediction with \'smsa\' = \'yes\':', predictedYes)

##   
##   
## Calculated average prediction with 'smsa' = 'yes': 40.2795

predictiondf = data.frame(Crime$density, Crime$smsa)  
prediction = predict(model1, predictiondf)  
cat('\n\nAverage prediction for \'smsa\' = \'no\' using predict( ) function:', mean(prediction[Crime$smsa == 'no']))

##   
##   
## Average prediction for 'smsa' = 'no' using predict( ) function: 29.32259

cat('\n\nAverage prediction for \'smsa\' = \'yes\' using predict( ) function:', mean(prediction[Crime$smsa == 'yes']))

##   
##   
## Average prediction for 'smsa' = 'yes' using predict( ) function: 39.63443

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

cat('Pearson correlation between tax per capita and density:', with(Crime, cor(taxpc, density)))

## Pearson correlation between tax per capita and density: 0.1997634

#The correlation coefficient is positive, which means that there is a positive relationship between the two variables; as one increases, so does the other. The correlation between the two is not so great, however, as the maximum possible correlation is 1, and the correlation in this case is 1/5 of that. So, overall, there is slight positive 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[c("avgsen", "polpc", "density", "taxpc")])

## 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

#It is clear that all variables are positively correlated (each increases with an increase in the other) with the exception of 'polpc' x 'density', which makes sense, as the number of police officers per person will decrease as the count of people per square mile increases.

## Question 2 -30%

1. First and second midterm grades of some students are given as c(85,76,78,88,90,95,42,31) and c(55,76,48,58,80,75,32,22). 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)

1. 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

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

firstmarkdf = data.frame(first = 72)  
cat('Predicted second midterm grade if first grade was 72: ', predict(model2, firstmarkdf))

## Predicted second midterm grade if first grade was 72: 54.87436

## Question 3 - 40%

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

Recall that the area of a circle of radius is . Therefore the area of a circle of radius 1, aka a *unit circle*, is . You’ll compute an approximation to the area of this circle using the Monte Carlo Method.

1. 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

1. 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)

x = runif(10000, -1)  
y = runif(10000, -1)

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

hits = 0  
  
for (i in 1:length(x)) if (x[i]^2 + y[i]^2 < 1) hits = hits + 1  
cat('Successful hits:', hits)

## Successful hits: 7790

1. 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 and print the result

proportion = hits/length(x)  
cat('The approximation of the unit circle area is:', proportion \* area.square)

## The approximation of the unit circle area is: 3.116

Wow you got the first estimate for pi , 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.

montecarlo = function(numcalls = 50) { pivector = numeric(numcalls) area.square = 2^2

for (j in 1:numcalls) { x = runif(10000, -1) y = runif(10000, -1) hits = 0 for (i in 1:length(x)) if (x[i]^2 + y[i]^2 < 1) hits = hits + 1 pivector[j] = hits } cat(pivector, ‘of hits using Monte Carlo:’, mean(pivector))

newvector = pivector newvector = (newvector/length(x)) \* area.square cat(‘of unit circle area prediction:’, mean(newvector)) }