BUS212 Session 1

Rob Carver, Brandeis IBS

March 2017

This script illustrates some common tasks and "best practices". It also illustrates the value and utility of using *R Markdown* when creating reports in R.

Before trying to run this code, visit our class GitHub repository and grab the r script bigdatapackages.R and the Word document called "Making your class directory the default start.docx". Run the script *once* (it will take a while), and follow the defaulty starting directory instructions.

I like to start with a code block that invokes packages that will be used in the code.

library(readr) # provide additional data-reading functions  
library(corrplot) # attractive correlation graphs

Each user must set a working directory for her/his own computer. I urge you to set up a directory for all of your R work. Within that, create a folder for BUS212, and then within *that* have 2 folders: one for Data, one for Scripts.

See the document titled "How to set your default working directory" on LATTE under the *R Resources section*.

In this example, my main R directory is located in my Brandeis Box account. We read the data using read\_csv from package readr, rather than the usualread.csv. We'll work with the data as a "tibble" rather than a data frame.

setwd("C:/Users/Rob/Box Sync/My R Work/BUS212")  
mydata <- read\_csv("Data/On\_Time\_50.csv")

In this example, we'll eventually want to do some modeling on the variable called ArrDelay. First let's get familiar with the data.

Helpful exploratory commands:

* dim -- report # rows & # columns
* names -- report column names
* str -- show the structure of object
* head -- show the first several rows of data
* summary -- compute Tukey's 5 number summary + mean for a variable

dim(mydata)

## [1] 49101 50

head(mydata)

## # A tibble: 6 × 50  
## Year Quarter Month DayofMonth DayOfWeek FlightDate UniqueCarrier  
## <int> <int> <int> <int> <int> <chr> <chr>  
## 1 2014 4 10 23 4 10/23/2014 DL  
## 2 2014 4 10 23 4 10/23/2014 DL  
## 3 2014 4 10 21 2 10/21/2014 DL  
## 4 2014 4 10 21 2 10/21/2014 DL  
## 5 2014 4 10 21 2 10/21/2014 DL  
## 6 2014 4 10 21 2 10/21/2014 DL  
## # ... with 43 more variables: TailNum <chr>, FlightNum <int>,  
## # Origin <chr>, OriginCityName <chr>, OriginState <chr>,  
## # OriginStateFips <int>, OriginStateName <chr>, Dest <chr>,  
## # DestCityName <chr>, DestState <chr>, DestStateFips <int>,  
## # DestStateName <chr>, CRSDepTime <int>, DepTime <int>, DepDelay <int>,  
## # DepDelayMinutes <int>, DepDel15 <int>, DepartureDelayGroups <int>,  
## # DepTimeBlk <chr>, TaxiOut <int>, WheelsOff <int>, WheelsOn <int>,  
## # TaxiIn <int>, CRSArrTime <int>, ArrTime <int>, ArrDelay <int>,  
## # ArrDelayMinutes <int>, ArrDel15 <int>, ArrivalDelayGroups <int>,  
## # ArrTimeBlk <chr>, Cancelled <int>, CancellationCode <chr>,  
## # Diverted <int>, CRSElapsedTime <int>, Distan <int>, AirTime <int>,  
## # Distance <int>, DistanceGroup <int>, CarrierDelay <int>,  
## # WeatherDelay <int>, NASDelay <int>, SecurityDelay <int>,  
## # LateAircraftDelay <int>

summary(mydata$ArrDelay) # compute stats for one variable

## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's   
## -65.00 -11.00 -3.00 4.87 8.00 1308.00 606

Suppose that we want to create a model to account for arrival delays longer than 15 minutes. This table has 50 variables, and surely many of the variables are not relevant to our purpose. Somewhat arbitrarily, let's select some columns that might be useful in such a model.

Next we create a new object containing 4 of the 50 variables (as specified), and only use rows where ArrDelay > 15 minutes. This is simliar to *Select... Where* in SQL.

# to simplify the subset command, first set up a vector of column names  
varlist <- c("DepDelay", "ArrDelay", "AirTime", "Distance")  
myvars <-subset (mydata, ArrDelay > 15, select=varlist)  
str(myvars)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 8759 obs. of 4 variables:  
## $ DepDelay: int -4 11 73 39 10 85 68 139 25 47 ...  
## $ ArrDelay: int 28 28 64 42 18 67 67 134 68 31 ...  
## $ AirTime : int 100 206 130 125 272 332 103 67 79 76 ...  
## $ Distance: int 746 1399 1010 980 1990 2586 746 440 488 488 ...

This creates a more compact data table with fewer than 9000 rows. Suppose we want to build a regression model to estimate arrival delays. As standard practice in model-building with "big data" we first split the large data frame into two partitions:

* a TRAINING set, used to initially develop a model for use
* a TEST set, used to validate the model with a new "out of sample" batch of data

# Demonstrate concept of partitioning a dataframe into Training  
# and Testing samples  
  
set.seed(1234) # initialize the random number generator for reproducibilty.  
# Use an integer of your choosing.  
  
# Below, "ind" will be a vectors of randomly generated 1s and 2s. There will be as  
# many values as there are rows of data in the myvars df. 60% of the ind values   
# will equal 1 nad 40% will equal 2.  
ind <- sample(2,nrow(myvars),replace=TRUE, prob=c(0.6,0.4))   
table(ind)

## ind  
## 1 2   
## 5274 3485

train <- myvars[ind==1,] # new df "train" will consists of randomly   
 # chosen rows from myvars, corresponding to the 1's in ind.  
test <- myvars[ind==2,]  
  
dim(train)

## [1] 5274 4

dim(test)

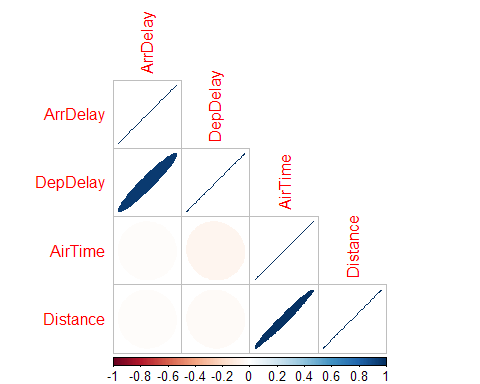
## [1] 3485 4

Before creating models, lets look at the correlations among the numeric variables. To help visualize the correlations, we'll use the package called "corrplot".

tr <- subset(train, select=c("ArrDelay","DepDelay","AirTime", "Distance"))  
  
cm <- cor(tr, method="pearson")  
cm

## ArrDelay DepDelay AirTime Distance  
## ArrDelay 1.00000000 0.95447643 -0.01176882 -0.01592223  
## DepDelay 0.95447643 1.00000000 -0.05361880 -0.02675997  
## AirTime -0.01176882 -0.05361880 1.00000000 0.98126974  
## Distance -0.01592223 -0.02675997 0.98126974 1.00000000

corrplot(cm, method= "ellipse", type="lower" )



Now we'll estimate 2 "candidate" linear regression models to estimate arrival. We'll choose the better model to re-run with the training data.

The important R function is lm (for linear model). We create the two models and then use summary to report results.

lm1 <- lm(ArrDelay ~ DepDelay, data = tr)  
summary(lm1)

##   
## Call:  
## lm(formula = ArrDelay ~ DepDelay, data = tr)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -49.263 -11.307 -3.147 8.465 148.188   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 10.287218 0.308209 33.38 <2e-16 \*\*\*  
## DepDelay 0.917482 0.003949 232.34 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 17.24 on 5272 degrees of freedom  
## Multiple R-squared: 0.911, Adjusted R-squared: 0.911   
## F-statistic: 5.398e+04 on 1 and 5272 DF, p-value: < 2.2e-16

lm2 <- lm(ArrDelay ~ DepDelay + Distance, data = tr)  
summary(lm2)

##   
## Call:  
## lm(formula = ArrDelay ~ DepDelay + Distance, data = tr)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -50.038 -11.203 -3.122 8.246 147.813   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 9.5325641 0.4456644 21.390 <2e-16 \*\*\*  
## DepDelay 0.9177294 0.0039486 232.416 <2e-16 \*\*\*  
## Distance 0.0009414 0.0004017 2.343 0.0191 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 17.23 on 5271 degrees of freedom  
## Multiple R-squared: 0.9111, Adjusted R-squared: 0.9111   
## F-statistic: 2.702e+04 on 2 and 5271 DF, p-value: < 2.2e-16

The second model appears to be barely better than the first. Now re-estimate it with the *test* data set and see if the quality of the model persists.

lm1t <- lm(ArrDelay ~ DepDelay, data = test)  
summary(lm1t)

##   
## Call:  
## lm(formula = ArrDelay ~ DepDelay, data = test)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -43.453 -11.223 -3.352 7.940 127.761   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 11.121911 0.404184 27.52 <2e-16 \*\*\*  
## DepDelay 0.901678 0.005684 158.63 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 17.61 on 3483 degrees of freedom  
## Multiple R-squared: 0.8784, Adjusted R-squared: 0.8784   
## F-statistic: 2.516e+04 on 1 and 3483 DF, p-value: < 2.2e-16

lm2t <- lm(ArrDelay ~ DepDelay + Distance, data = test)  
summary(lm2t)

##   
## Call:  
## lm(formula = ArrDelay ~ DepDelay + Distance, data = test)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -45.266 -11.231 -3.391 7.956 127.499   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.013e+01 5.786e-01 17.513 <2e-16 \*\*\*  
## DepDelay 9.024e-01 5.689e-03 158.635 <2e-16 \*\*\*  
## Distance 1.199e-03 5.029e-04 2.384 0.0172 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 17.6 on 3482 degrees of freedom  
## Multiple R-squared: 0.8786, Adjusted R-squared: 0.8785   
## F-statistic: 1.26e+04 on 2 and 3482 DF, p-value: < 2.2e-16