Intro R and Regression

Ross Jacobucci

Outline

This script will go over the basics of programming in R along with covering Smoothing and Stepwise Regression

To get started, I recommend downloading Rstudio as an interface to R. Benefits include:

- Easier to download packages
- Easier to view plots
- Can load datasets without code

Loading Datasets

Many datasets that we will be using come with installed packages

```
library(MASS) # for boston data
data(Boston)
```

Now the Boston dataset is in your workplace

```
head(Boston[,1:7],6)
```

```
##       crim zn indus chas   nox    rm  age
## 1 0.00632 18  2.31    0 0.538 6.575 65.2
## 2 0.02731  0  7.07    0 0.469 6.421 78.9
## 3 0.02729  0  7.07    0 0.469 7.185 61.1
## 4 0.03237  0  2.18    0 0.458 6.998 45.8
## 5 0.06905  0  2.18    0 0.458 7.147 54.2
## 6 0.02985  0  2.18    0 0.458 6.430 58.7
```

Load your own datasets

Data saved as .dat or .txt

```
data = read.table(file.choose(),sep="",header=F,na.strings="NA")
```

.csv

```
data = read.csv(file.choose(),header=T,na.strings=".")
```

.sav for SPSS files

```
library(foreign)
data = read.spss(file.choose(),to.data.frame=TRUE)
```

.sas7bdat for SAS data files – note doesn't work very well, best to convert first

```
library(sas7bdat)
data = read.sas7bdat(file.choose())
# note many options
```

Subsetting datasets

Let's say I have a dataset read in fine. How about subsetting columns and rows: * specific columns – two ways

```
library(MASS)
data(Boston)
## I want specific columns

Boston.sub <- Boston[,c(1,2,8,9)]
## or
Boston.sub <- Boston[,c("crim","zn","dis","rad")]
```

Have to use c() to combine non-continuous elements in R !!!!!! VERY IMPORTANT !!!!!

How about rows – Same process but better shortcuts

```
comps <- complete.cases(Boston)
Boston.comp <- Boston[comps,]

subs <- Boston$zn == 0
Boston.sub <- Boston[subs,]
dim(Boston.sub)
```

```
## [1] 372  14
```

Notice: left of comma = row indicator; right = column indicator

Dataset Types

For manipulating datasets, almost always best to make sure in data.frame

```
library(MASS)
data(Boston)

Boston.df <- data.frame(Boston)
```

Can tell type by running str()

Some R packages require X and Y variables to be in separate matrices

```
library(MASS)
data(Boston)

Y <- as.numeric(Boston$medv)
X <- data.matrix(Boston[,-14])

out <- lm.fit(X,Y)
```

This is very specific to data mining packages

Missing Data

Can be summed up in two words: Not Fun

R requires all missingness to be coded as NA, therefore it is best to deal with when reading in data:

```
data = read.csv(file.choose(),header=T,na.strings=".")
```

This takes all missing values coded with a period in the original dataset and converts them to NA for R

Once you have a dataset in R and missingness coded as NA:

```
# have to use is.na a lot
is.na(c(1,2,NA,6))
```

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

Returns a logical indicator. Can save this to subset dataset based on number of missings

Missing Data Continued

```
library(psych)
data(bfi)
dim(bfi)
```

```
## [1] 2800   28
```

Now, subsetting based on number of missing values per person

```
ids <- rowSums(is.na(bfi)) < 3
bfi.sub <- bfi[ids,]
dim(bfi.sub)
```

```
## [1] 2779   28
```

Variable Types

This is really important as it will change the type of estimator used e.g. logistic vs. linear regression

```r
bfi.sub <- bfi[,1:5]
library(rpart)
out1 <- rpart(A1 ~ ., bfi.sub)
out1
```

```
## n=2784 (16 observations deleted due to missingness)
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
## 1) root 2784 5515.1380 2.413434
##   2) A2>=4.5 1907 3382.5430 2.126377
##     4) A2>=5.5 874 1618.1660 1.925629 *
##     5) A2< 5.5 1033 1699.3550 2.296225 *
##   3) A2< 4.5 877 1633.7580 3.037628
##     6) A2>=3.5 554  879.9495 2.819495 *
##     7) A2< 3.5 323  682.2353 3.411765 *
```

Variable Types Continued

```
out2 <- rpart(as.factor(A1) ~ ., bfi.sub)
out2
```

```
## n=2784 (16 observations deleted due to missingness)
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 2784 1862 1 (0.33 0.29 0.14 0.12 0.08 0.029)
##   2) A2>=5.5 874  386 1 (0.56 0.22 0.072 0.064 0.055 0.03) *
##   3) A2< 5.5 1910 1285 2 (0.23 0.33 0.18 0.15 0.092 0.029) *
```

Changing the variable type, from integer to factor also changes the cost function. This results in very different equations and also very different results.

Most important is the variable type of the Y variable. Also can change it in the actual function by setting "family ="

Also can change variable type in dataset and the data mining function will recognize this. However, I almost always just include it in the actual script to make sure it changes to the correct estimator.

How to get help

```
library(caret)
?train
```

However, I usually just google the R package and look at the manual

Once you have run a script and have it save as an object, you can

```
library(MASS)
data(Boston)
out <- lm(medv ~ ., data=Boston)
str(out)
```

Using str() will list all of the attributes of the "out" object. This is important to use if print() or summary () dont give you the information you want. Most applicable if you need nitty gritty details

Installing Packages

```
install.packages("MASS",dependencies=TRUE)
```

Since I use Rstudio, I almost always use the button interface in the bottom right panel to install packages. This automatically installs the other packages the package you want to install depends on.

Also, it is also possible to install package directly from source. Later in the week we will be working with the longRPart package that is currently not maintained in CRAN, thus you have to download a version from the archiv and install from source.

```
install.packages("~/GitHub/SearchWkshp_labs16/longRPart_1.0.tar.gz",
                 type = "source")
```

Also note that packages depend on the build of R. For many packages, you have to have the version of R that matches the R version that the package was built under. This is why when you update your R version, you have to update a lot of packages, and vice versa.

Parallelization

The caret package makes it easy to parallelize different methods. Its built in to their control function http://topepo.github.io/caret/parallel.html

```
library(MASS);library(caret)
data(Boston)
library(doParallel)
cl <- makeCluster(detectCores())
registerDoParallel(cl)

out <- train(medv ~ ., data=Boston,method="rf")
```

A lot of the methods we will be talking about are very parallelizable and will vastly decrease the amount of time it takes to run

Most Important Details for Learning R

1. Read in data
2. How to subset dataset
3. Missing data handling
4. Variable types
5. Understand where the results go and how to access them
6. How to install and the intricacies to each package
7. How to parallelize analyses

This is obviously not enought time to cover R as a programming language but our goal is to give you a working knowledge and learn enough as we proceed that you can implement all of the models we talk about.

Questions?

```
data.1<-read.table("/Volumes/GoogleDrive/My Drive/Statistical_Horizons/May2022/
names(data.1) = c('id', 'apexpos', 'fsiq7')
head(data.1)
```

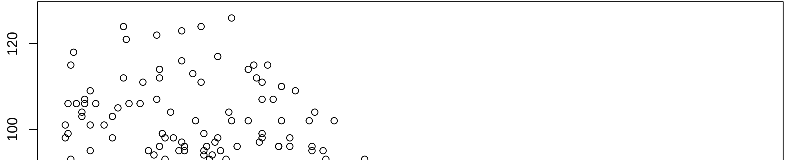
```
##   id apexpos fsiq7
## 1  1     2.2   121
## 2  2     5.5    98
## 3  3     5.2    93
## 4  4     3.6    98
## 5  5     7.2    89
## 6  6     7.3   115
```

```
# Sort Data
data.2 <- data.1[order(data.1$apexpos),]

attach(data.2)

## Plotting empirical data ##

plot(apexpos, fsiq7, ylab = 'Full Scale IQ', xlab = 'PHE Exposure')
```



Linear Regression Continued

```r
lm.1 = lm(fsiq7 ~ apexpos)

summary(lm.1)
```

```
##
## Call:
## lm(formula = fsiq7 ~ apexpos)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -33.108  -8.790  -0.232   8.004  39.749
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 107.4337     1.4289   75.19   <2e-16 ***
## apexpos      -2.9068     0.1079  -26.93   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.25 on 312 degrees of freedom
## Multiple R-squared:  0.6992, Adjusted R-squared:  0.6982
## F-statistic: 725.1 on 1 and 312 DF,  p-value: < 2.2e-16
```

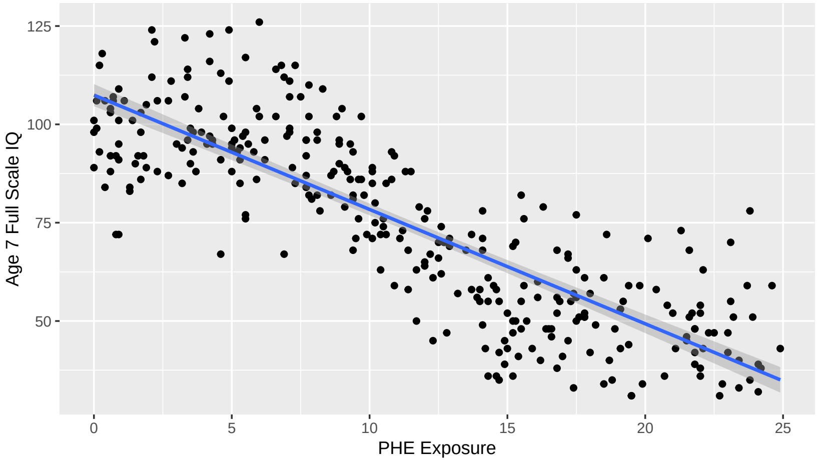
```r
#plot(apexpos, fitted(lm.1), ylab = 'Predicted Full Scale IQ', xlab = 'PHE Expos

#plot(apexpos, fsiq7, ylab = 'Full Scale IQ', xlab = 'PHE Exposure')
#abline(lm(fsiq7 ~ apexpos), col="red")
```

Plot Model

```
library(ggplot2)

plot.1 = ggplot(data.1, aes(x=apexpos, y=fsiq7)) + geom_point() +
              stat_smooth(method='lm', formula = y ~ x, size = 1) +
              xlab('PHE Exposure') + ylab('Age 7 Full Scale IQ')
print(plot.1)
```



Regression Diagnostics on Linear Regression Model

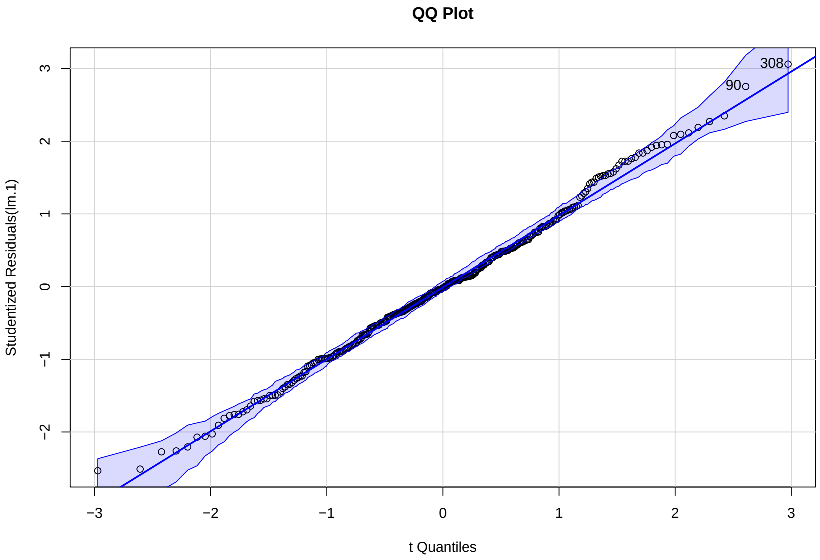
```
library(car)

#Outlier Test
outlierTest(lm.1) # Bonferonni p-value for most extreme obs
```

```
## No Studentized residuals with Bonferroni p < 0.05
## Largest |rstudent|:
##     rstudent unadjusted p-value Bonferroni p
## 308 3.060391          0.0024031      0.75457
```

QQ Plot

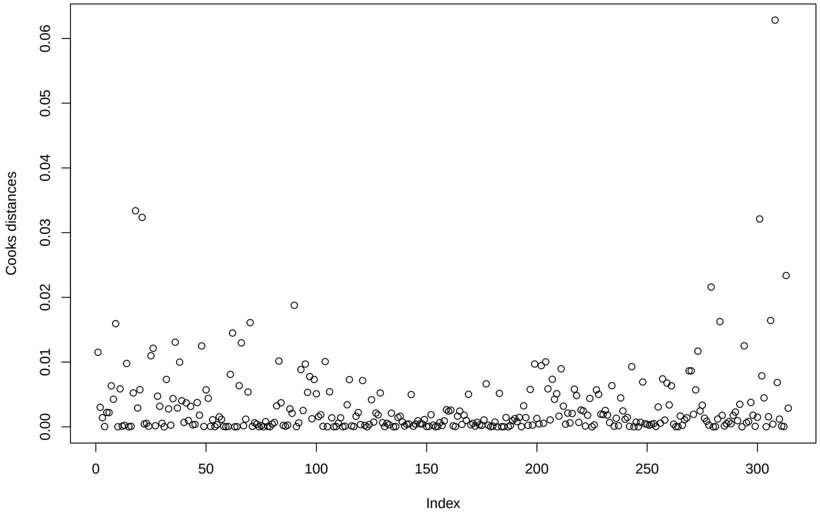
```
qqPlot(lm.1, main="QQ Plot") #qq plot for studentized resid
```




```
## [1]  90 308
```

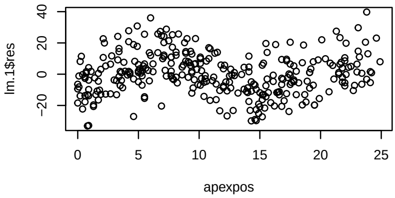
Cook's Distance

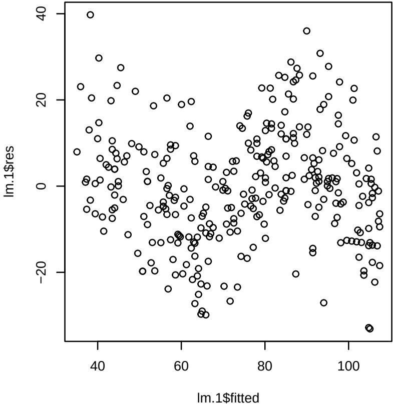
```
cook = cooks.distance(lm.1)
plot(cook,ylab="Cooks distances")
```



Plot Residuals

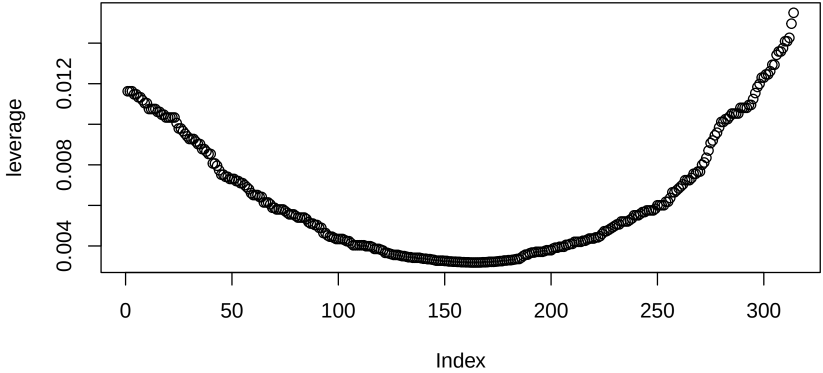
```
# Plots
layout( cbind( c(0,0,1,1,1,1,1,0,0), rep(2,9) ) )
plot(apexpos, lm.1$res)
plot(lm.1$fitted, lm.1$res)
```





Leverage Plot

```
leverage = hat(model.matrix(lm.1))
plot(leverage)
```



Quadratic Regression

```
apexpos2 = apexpos^2

lm.2 = lm(fsiq7 ~ apexpos + apexpos2)

summary(lm.2)
```

```
##
## Call:
## lm(formula = fsiq7 ~ apexpos + apexpos2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -33.939  -8.731  -0.351   7.826  38.618
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 108.47508    2.05289  52.840  < 2e-16 ***
## apexpos      -3.17979    0.40087  -7.932 3.91e-14 ***
## apexpos2      0.01163    0.01644   0.707     0.48
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.26 on 311 degrees of freedom
## Multiple R-squared:  0.6997, Adjusted R-squared:  0.6977
## F-statistic: 362.2 on 2 and 311 DF,  p-value: < 2.2e-16
```

Didn't really improve the fit, and it looks like we are still violating some assumptions (if you run the same plots as with lm.1)

Easier Way to do Quadratic+

```
lm.3 = lm(fsiq7 ~ poly(apexpos,3))

summary(lm.3)
```

```
##
## Call:
## lm(formula = fsiq7 ~ poly(apexpos, 3))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -32.538  -7.332  -0.282   7.248  29.582
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          74.6465     0.6708 111.281   <2e-16 ***
## poly(apexpos, 3)1 -356.8429    11.8865 -30.021   <2e-16 ***
## poly(apexpos, 3)2    9.3768    11.8865   0.789    0.431
## poly(apexpos, 3)3  104.4101    11.8865   8.784   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.89 on 310 degrees of freedom
## Multiple R-squared:  0.7595, Adjusted R-squared:  0.7572
## F-statistic: 326.3 on 3 and 310 DF,  p-value: < 2.2e-16
```

Get standardized coefficients

```
library(QuantPsyc)
lm.beta(lm.1)
```

```
##    apexpos
## -0.8361638
```

Logistic Regression

Can also use glm() to run other types of regression. See ?family

```
library(ISLR); data(Default)
lr.out <- glm(default~.,family="binomial",data=Default)
summary(lr.out)
```

```
##
## Call:
## glm(formula = default ~ ., family = "binomial", data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4691  -0.1418  -0.0557  -0.0203   3.7383
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.087e+01  4.923e-01 -22.080  < 2e-16 ***
## studentYes  -6.468e-01  2.363e-01  -2.738  0.00619 **
## balance      5.737e-03  2.319e-04  24.738  < 2e-16 ***
## income       3.033e-06  8.203e-06   0.370  0.71152
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1571.5  on 9996  degrees of freedom
## AIC: 1579.5
```