Final

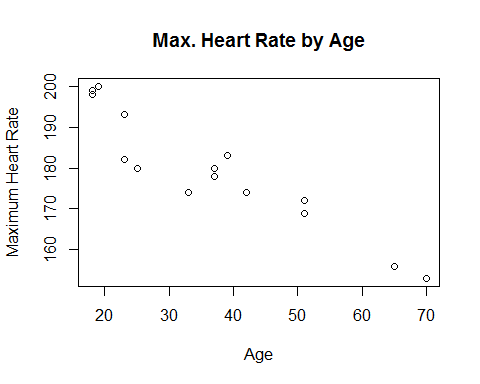
Mike Lehman

December 12, 2016

## #1

#1a  
age <- c(19,23,25,37,65,51,33,51,70,18,23,42,18,39,37)  
maxHR <- c(200,182,180,180,156,169,174,172,153, 199, 193, 174, 198, 183, 178)

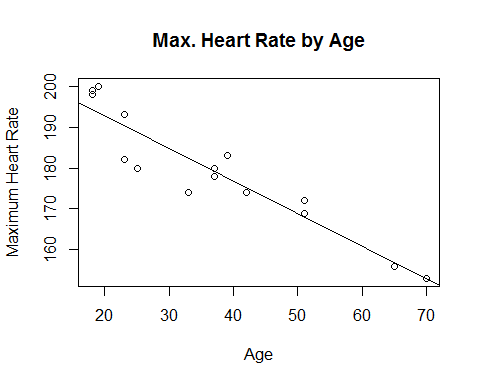
#1b  
plot(age, maxHR, xlab = "Age", ylab = "Maximum Heart Rate", main = "Max. Heart Rate by Age")



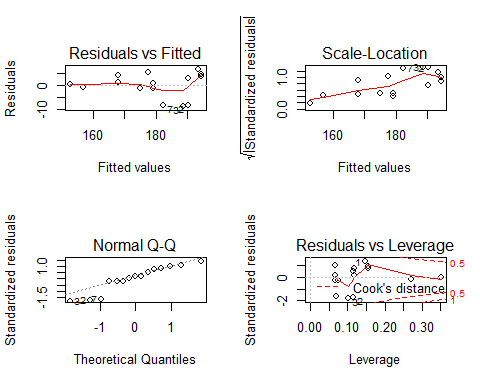
#1c  
ageHR.df <- as.data.frame(cbind(age, maxHR))  
fit <- lm(maxHR ~ age, ageHR.df)  
fit

##   
## Call:  
## lm(formula = maxHR ~ age, data = ageHR.df)  
##   
## Coefficients:  
## (Intercept) age   
## 208.7079 -0.7979

#1d  
plot(age, maxHR, xlab = "Age", ylab = "Maximum Heart Rate", main = "Max. Heart Rate by Age")  
abline(208.7079, -0.7979)



#1e  
layout(matrix(1:4, nrow = 2))  
plot(fit)



### Residuals vs. Fitted

The residuals vs. fitted plot shows the difference between actual and predicted (fitted) values on the y-axis, and the predicted (fitted) values on the x-axis. This plot shows if the residuals have non-linear patterns which the model may not have found, but would show up in this plot. The results show that there aren't any non-linear patterns about which we should be concerned. The points cluster around the horizontal line and do not have any distinct non-linear pattern.

### Scale Location

The scale location plot shows if residuals are spread equally along the ranges of predictors. If the points are randomly scattered along a horizontal line. In our case the line has an upward angle, and the points scatter wider as the slope increases. This means the residuals (difference between actual and predicted values) increased as the predicted value increased.

### Q-Q Plot

Q-Q plots show if residuals are normally distributed. The points should fall along a straight dashed line without much random spread. In our case the points are almost all distributed along the dashed line, but some of the smaller values fall well below the line.

### Residuals vs. Leverage

Residuals vs. lerverage plots help to fund influential cases of outliers. Large or small outliers may or may not have a significant impact in a linear model. Meaning, it may be possible to discard them and not effect the model's outcome all that much. Values in the upper-right or lower-right corner indicate significant outliers outside of the regression line. In our case there do not appear to be any significant outliers.

#1f  
fit

##   
## Call:  
## lm(formula = maxHR ~ age, data = ageHR.df)  
##   
## Coefficients:  
## (Intercept) age   
## 208.7079 -0.7979

The coefficient of -0.7979 for age implies that for every unit increase in age, maximum heart rate decreases by 0.7979.

#1g  
summary(fit)

##   
## Call:  
## lm(formula = maxHR ~ age, data = ageHR.df)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -8.7615 -1.1926 0.8128 3.8181 6.4514   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 208.70787 3.33992 62.489 < 2e-16 \*\*\*  
## age -0.79785 0.08327 -9.581 2.95e-07 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 5.194 on 13 degrees of freedom  
## Multiple R-squared: 0.876, Adjusted R-squared: 0.8664   
## F-statistic: 91.8 on 1 and 13 DF, p-value: 2.948e-07

### Residual Standard Error

The Residual Standard Error is the sum of the squares of the residuals (differences between actual and predicted values) divided by degrees of freedom (number of rows - number of coefficients for which to solve).

### R-Squared

R-Squared is the sum of squares of residuals divided by the sum of squares of the difference between actuals and mean values. The value will range between 0 and 1 with a value closer to 1 meaning a better fit. Basically, how close the predicted values are to the regression line and can be summarized as: explain variation / total variation.

### Adjusted R-Squared

The Adjusted R-Squared is an R-Squared value that has been adjusted based on the number of predictors in the model. This is useful to compare multiple models trained on the same data but with different numbers of predictors. In general, it is a truer to actual performance metric than the raw R-Squared.

#1h  
new.ages <- data.frame(age = c(50, 60))  
fit.prediction <- predict(fit, newdata = new.ages)  
fit.prediction

## 1 2   
## 168.8151 160.8366

## #2

library(igraph)

##   
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':  
##   
## decompose, spectrum

## The following object is masked from 'package:base':  
##   
## union

library(igraphdata)  
data(USairports)

#2a  
airports <- USairports  
  
# vertices  
V(airports)

## + 755/755 vertices, named:  
## [1] BGR BOS ANC JFK LAS MIA EWR BJC TEB LAX AEX BFI ELM GEG ICT PBI PIT  
## [18] SFO VCT IAD ABE AGS AVL AVP BDL BHM BNA BTR BUF BWI CAE CAK CHA CHO  
## [35] CHS CLE CLT CMH CRW CVG DAB DAY DCA DTW EWN FAY GNV GPT GSO GSP HPN  
## [52] HSV ILM IND JAN LEX LGA LIT MDT MGM MKE MLB MOB MSP MSY MYR OAJ ORF  
## [69] PGV PHF PHL PNS PWM RDU RIC ROA SAV SDF SRQ STL SYR TLH TRI TYS VPS  
## [86] XNA ALB BGM BTV ERI FLO HHH HTS HVN IPT ISP ITH LYH MHT PVD ROC SBY  
## [103] SCE SWF BFD DUJ EYW FKL FLL JHW LWB MCO PKB TPA ACT AOO BHB BKW BPT  
## [120] CKB CLL DRT GRK IAH JST LCH LFT MAF MGW MLU ORD PBG PQI SHD SHV TYR  
## [137] CID MCI MLI MSN OKC OMA SBN SGF TUL GKN ABQ ATL AUS BKG DEN DFW HOU  
## [154] MDW PDX PHX PIE RSW SAN SAT SEA SLC SMF SNA TUS ABY ACY BMI DSM FNT  
## + ... omitted several vertices

# edges  
E(airports)

## + 23473/23473 edges (vertex names):  
## [1] BGR->JFK BGR->JFK BOS->EWR ANC->JFK JFK->ANC LAS->LAX MIA->JFK  
## [8] EWR->ANC BJC->MIA MIA->BJC TEB->ANC JFK->LAX LAX->JFK LAX->SFO  
## [15] AEX->LAS BFI->SBA ELM->PIT GEG->SUN ICT->PBI LAS->LAX LAS->PBI  
## [22] LAS->SFO LAX->LAS PBI->AEX PBI->ICT PIT->VCT SFO->LAX VCT->DWH  
## [29] IAD->JFK ABE->CLT ABE->HPN AGS->CLT AGS->CLT AVL->CLT AVL->CLT  
## [36] AVP->CLT AVP->PHL BDL->CLT BHM->CLT BHM->CLT BNA->CLT BNA->CLT  
## [43] BNA->DCA BNA->PHL BTR->CLT BUF->CLT BUF->DCA BUF->PHL BWI->PHL  
## [50] CAE->CLT CAE->CLT CAE->DCA CAK->CLT CAK->CLT CAK->DCA CAK->PHL  
## [57] CHA->CLT CHA->DCA CHO->CLT CHS->CLT CHS->CLT CHS->DCA CLE->CLT  
## [64] CLE->CLT CLE->PHL CLT->ABE CLT->AGS CLT->AGS CLT->AVL CLT->AVL  
## + ... omitted several edges

# directed?  
is\_directed(airports)

## [1] TRUE

#2b  
# maximum degrees  
max(degree(airports))

## [1] 1700

# minimum degrees  
min(degree(airports))

## [1] 1

# average degrees  
mean(degree(airports))

## [1] 62.18013

#2c  
# busiest  
V(airports)$name[degree(airports) == max(degree(airports))]

## [1] "ATL"

# least busy  
V(airports)$name[degree(airports) == min(degree(airports))]

## [1] "GKN" "FNR" "BIG" "PML" "BKL" "LCK" "PNE" "TVL" "FTW" "MPV" "PWK"  
## [12] "RIL" "AND" "GYY" "VNY" "STJ" "DWH" "MXY" "SVW" "CFA" "FXE" "LFI"  
## [23] "FPR"

#2d  
mean(E(airports))

## [1] 11737

#2e  
E(airports) [ to("JFK") ] #23,473

## + 313/23473 edges (vertex names):  
## [1] BGR->JFK BGR->JFK ANC->JFK MIA->JFK LAX->JFK IAD->JFK CLT->JFK  
## [8] CLT->JFK PHX->JFK AUS->JFK AUS->JFK DFW->JFK DFW->JFK EGE->JFK  
## [15] FLL->JFK FLL->JFK FLL->JFK IAD->JFK LAS->JFK LAS->JFK LAX->JFK  
## [22] LAX->JFK LAX->JFK MCO->JFK MCO->JFK MCO->JFK MIA->JFK MIA->JFK  
## [29] MIA->JFK MIA->JFK ORD->JFK ORD->JFK ORD->JFK ORD->JFK ORD->JFK  
## [36] PHL->JFK RDU->JFK SAN->JFK SEA->JFK SFO->JFK SFO->JFK SFO->JFK  
## [43] SJU->JFK SJU->JFK STL->JFK STL->JFK STT->JFK TPA->JFK ACY->JFK  
## [50] AUS->JFK AVP->JFK BOS->JFK BOS->JFK BQN->JFK BTV->JFK BTV->JFK  
## [57] BUF->JFK BUF->JFK BUR->JFK CLT->JFK DEN->JFK FLL->JFK HOU->JFK  
## [64] HOU->JFK IAD->JFK IAD->JFK JAX->JFK JAX->JFK LAS->JFK LAX->JFK  
## + ... omitted several edges

E(airports) [ from("JFK") ] #23,473

## + 294/23473 edges (vertex names):  
## [1] JFK->ANC JFK->LAX JFK->CLT JFK->CLT JFK->PHX JFK->AUS JFK->AUS  
## [8] JFK->DFW JFK->EGE JFK->FLL JFK->FLL JFK->FLL JFK->LAS JFK->LAS  
## [15] JFK->LAX JFK->LAX JFK->LAX JFK->MCO JFK->MCO JFK->MCO JFK->MIA  
## [22] JFK->MIA JFK->MIA JFK->MIA JFK->ORD JFK->ORD JFK->ORD JFK->ORD  
## [29] JFK->RDU JFK->SAN JFK->SEA JFK->SFO JFK->SFO JFK->SFO JFK->SJU  
## [36] JFK->SJU JFK->STL JFK->STT JFK->TPA JFK->AUS JFK->BOS JFK->BOS  
## [43] JFK->BQN JFK->BTV JFK->BTV JFK->BUF JFK->BUF JFK->BUR JFK->CLT  
## [50] JFK->DEN JFK->DTW JFK->FLL JFK->HOU JFK->HOU JFK->IAD JFK->IAD  
## [57] JFK->JAX JFK->JAX JFK->LAS JFK->LAX JFK->LGB JFK->MCO JFK->MCO  
## [64] JFK->MSY JFK->MSY JFK->OAK JFK->ORD JFK->ORD JFK->PBI JFK->PBI  
## + ... omitted several edges

Total flights to JFK: 23,473 Total flights from JFK: 23,473

Total: 46,946

## #3

library(TH.data)

## Loading required package: survival

## Loading required package: MASS

##   
## Attaching package: 'TH.data'

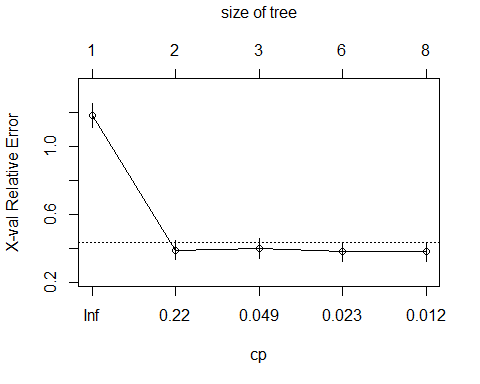
## The following object is masked from 'package:MASS':  
##   
## geyser

glaucoma <- GlaucomaM

#3a  
library(rpart)  
tree <- rpart(Class ~ ., data = glaucoma, method="class", parms = list(split="information"))  
tree$cptable

## CP nsplit rel error xerror xstd  
## 1 0.65306122 0 1.0000000 1.1836735 0.07021338  
## 2 0.07142857 1 0.3469388 0.3877551 0.05647630  
## 3 0.03401361 2 0.2755102 0.3979592 0.05703324  
## 4 0.01530612 5 0.1734694 0.3775510 0.05590431  
## 5 0.01000000 7 0.1428571 0.3775510 0.05590431

#3b  
plotcp(tree)



Select the minimum xerror and corresponding standard deviation xstd. Select the smallest tree within 1 standard deviation of the minimum.

Select the smallest tree whose xerror is below sum of xerror and xstd:

min xerror = 0.3673469

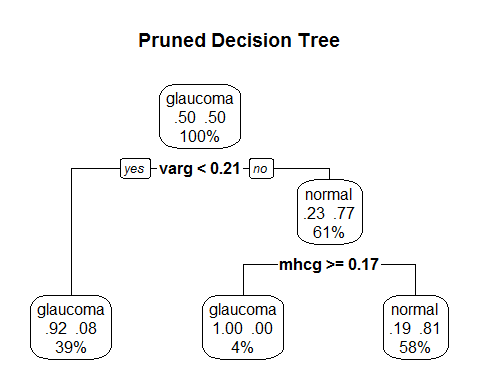
min xstd = 0.5531681

0.37 + 0.55 = 0.92

smallest tree (nsplit) with xerror below 0.92 = tree #2 (1 split)

cp = 0.07142857 ~ 0.071

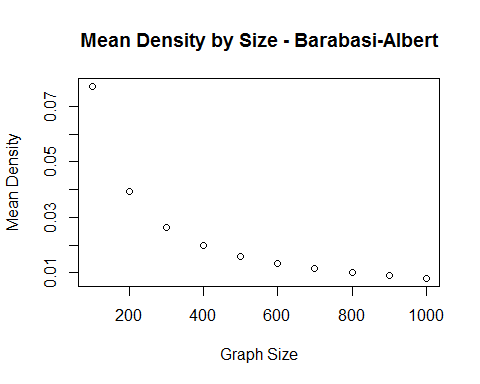
#3c  
tree.pruned <- prune(tree, cp = .071)  
  
library(rpart.plot)  
prp(tree.pruned, type = 2, extra = 104, fallen.leaves = TRUE, main = "Pruned Decision Tree")



## #4

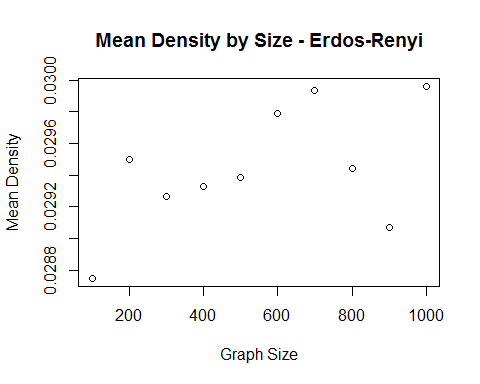
### Barabasi-Albert

#4a  
# Barabasi-Albert   
g.b.densities <- matrix(ncol = 10, nrow = 10)  
  
for (n in 1:10) {  
 for (i in seq(100, 1000, 100)) {  
 g.b <- barabasi.game(i, power = 1, m = 8)  
 g.b.densities[i/100, n] = graph.density(g.b)  
 }  
}  
  
g.b.means <- vector(mode = "numeric", length = 10)  
  
for (i in 1:10) {  
 g.b.means[i] = mean(g.b.densities[i,1:10])  
}  
  
sizes <- c(100,200,300,400,500,600,700,800,900,1000)  
g.b.df <- data.frame(sizes, g.b.means)  
names(g.b.df) <- c("size","mean\_density")  
  
plot(g.b.df$size, g.b.df$mean\_density, xlab = "Graph Size", ylab = "Mean Density", main = "Mean Density by Size - Barabasi-Albert")



### Erdos-Renyi

g.er.densities <- matrix(ncol = 10, nrow = 10)  
  
for (n in 1:10) {  
 for (i in seq(100, 1000, 100)){  
 g.er <- erdos.renyi.game(i, p = (10/(i-1)))   
 g.er.densities[n, i/100] = graph.density(g.er)  
 }  
}  
  
g.er.means <- vector(mode = "numeric", length = 10)  
  
for (i in 1:10) {  
 g.er.means[i] = mean(g.er.densities[i,1:10])  
}  
  
g.er.df <- data.frame(sizes, g.er.means)  
names(g.er.df) <- c("size","mean\_density")  
  
plot(g.er.df$size, g.er.df$mean\_density, xlab = "Graph Size", ylab = "Mean Density", main = "Mean Density by Size - Erdos-Renyi")



### Barabasi-Albert

For the Barabasi-Albert graphs, as graph size increases, mean density decreases. Graph density refers to the total number of edges actually in the graph versus the total possible edges. For the random Barabasi-Albert graphs, there is a steady decrease in density as graph size increases. The mean densities approach 0. There appears to be a direct correlation between BA graph size and density.

### Erdos-Renyi

As Erdos-Renyi graph size increases, the mean densities of the graphs appear to fluctuate randomly. This seems to indicate that, unlike Barabasi-Albert graphs, there does not appear to be a direct correlation between graph size and graph density.

## #5

#5a  
library(bigmemory)

## Loading required package: bigmemory.sri

airline\_08 <- read.big.matrix("2008.csv", type = "integer", header = TRUE, backingfile = "2008.bin", descriptorfile = "2008.desc")  
  
summary(airline\_08)

## Length Class Mode   
## 203282112 big.matrix S4

#5b  
na\_tail\_flights <- mwhich(airline\_08, "TailNum", NA, "eq")  
nrow(airline\_08[na\_tail\_flights,])

## [1] 6754265

6,754,265

#5c  
days <- c("Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday")  
flightCountByDays <- vector(mode = "numeric", length = 7)  
  
for (i in 1:7) {  
 day.index <- mwhich(airline\_08, "DayOfWeek", i, "eq")  
 flightCountByDays[i] <- nrow(airline\_08[day.index,])   
}  
  
days.df <- data.frame(days,flightCountByDays)  
names(days.df) <- c("day\_of\_week","flights")  
  
days.df

## day\_of\_week flights  
## 1 Monday 1036201  
## 2 Tuesday 1032049  
## 3 Wednesday 1039665  
## 4 Thursday 1032224  
## 5 Friday 1035166  
## 6 Saturday 857536  
## 7 Sunday 976887