# Class 28 R Logistic Regression Assessing the model

library(Stat2Data)  
library(readr)  
  
GoldenBalls <- read\_csv("https://raw.githubusercontent.com/JA-McLean/STOR455/master/data/GoldenBalls.csv")  
  
logit = function(B0, B1, x)  
 {  
 exp(B0+B1\*x)/(1+exp(B0+B1\*x))  
 }

**Checking Linearity** Three methods depending on the type of dataset: - Datasets with a binary predictor – nothing to check! - Datasets with a quantitative predictor with many response values for each predictor - Datasets with a quantitative predictor with many values for the predictor but few response values for each predictor value.

*Example Golf Putts* We looked at the log odds formula there wre no big deviations

what about the Datsets with binary predicotrs? We are going to look at golden balls; split or steal the money how likeyly are they to split or steal based on teh age of teh contestant?

head(GoldenBalls)

## # A tibble: 6 x 2  
## Over40 Split  
## <dbl> <dbl>  
## 1 1 1  
## 2 1 1  
## 3 1 1  
## 4 1 1  
## 5 1 1  
## 6 1 1

table(GoldenBalls$Split, GoldenBalls$Over40)

##   
## 0 1  
## 0 195 76  
## 1 187 116

Claim: There is a difference in the proportion of people who would split or steal based on their age.

He table above shows the table for if the preson is over40 and has split or not; so row, column; where row = if they split and coloumn = age over or below 40. over forty = 1; under = 0

we see that there is a low p value below that says that we have evidence to say there is a realtionship between age and their saying yes or no to steal

then we pull out teh coeffs. we plot the raw data with some jitter, but we dont think tis super needed right now.

if we want to make a logit plot,then we want to find out where the table is, why do we table 2 and coloumn; p-hat will give you the samplel porp[tion, he is just pulling from the table

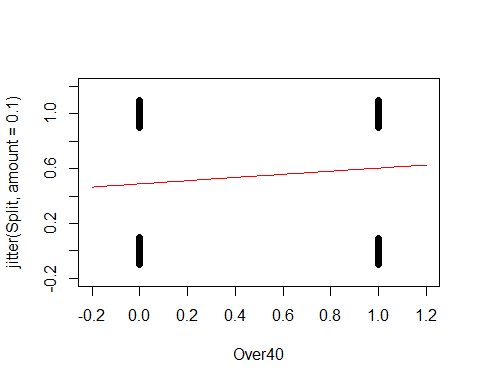
then goldenball odds is just loged

**Golden Balls: Logistic Regression**

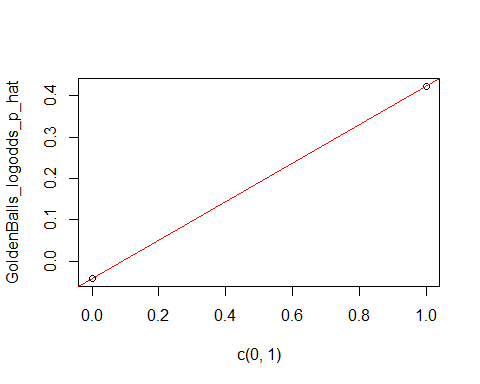
GBmod = glm(Split~Over40, data=GoldenBalls, family=binomial)  
# Make logistic model  
summary(GBmod)

##   
## Call:  
## glm(formula = Split ~ Over40, family = binomial, data = GoldenBalls)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.361 -1.160 1.004 1.195 1.195   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.04189 0.10235 -0.409 0.68233   
## Over40 0.46475 0.17960 2.588 0.00966 \*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 793.95 on 573 degrees of freedom  
## Residual deviance: 787.17 on 572 degrees of freedom  
## AIC: 791.17  
##   
## Number of Fisher Scoring iterations: 4

# See summary of logistic mdoel   
  
B0 = summary(GBmod)$coef[1] # Pull out intercept  
B1 = summary(GBmod)$coef[2] # Pull out slope  
  
#Plot the GBMod data  
plot(jitter(Split,amount=0.1)~Over40,data=GoldenBalls, xlim=c(-.2, 1.2), ylim=c(-.2, 1.2) )  
# Plot the GBMod  
curve(logit(B0, B1, x),add=TRUE, col="red")



GoldenBalls\_table=table(GoldenBalls$Split, GoldenBalls$Over40)  
GoldenBalls\_p\_hat=as.vector(GoldenBalls\_table[2,]/colSums(GoldenBalls\_table))  
GoldenBalls\_logodds\_p\_hat = log(GoldenBalls\_p\_hat/(1-GoldenBalls\_p\_hat))  
  
plot(GoldenBalls\_logodds\_p\_hat~c(0,1))  
abline(B0, B1, col="red")



**Quantitative predictor: Few response values for each predictor** - This process breaks down if there are not many values of the response, but the previous process can be mimicked by manipulating the data first.

* We can manipulate the data by:

1. Slicing the x-axis into intervals.
2. Compute the average x-value and empirical logit foreach slice
3. Plot the values as before

Medical School Acceptance Dataset: Is GPA a useful predictor for acceptance to medical school?

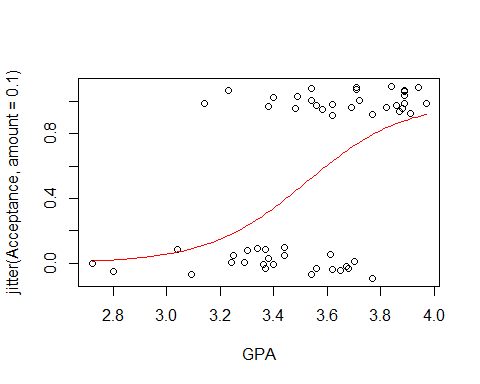
data("MedGPA")  
head(MedGPA)

## Accept Acceptance Sex BCPM GPA VR PS WS BS MCAT Apps  
## 1 D 0 F 3.59 3.62 11 9 9 9 38 5  
## 2 A 1 M 3.75 3.84 12 13 8 12 45 3  
## 3 A 1 F 3.24 3.23 9 10 5 9 33 19  
## 4 A 1 F 3.74 3.69 12 11 7 10 40 5  
## 5 A 1 F 3.53 3.38 9 11 4 11 35 11  
## 6 A 1 M 3.59 3.72 10 9 7 10 36 5

WE’re hoping that therea re different cases for the low, medium and high; if they didn’t follow the logit curve, ten they would be weird and you cant use logit

below we are predicting the mdoel with acceptabnce and gpa

MedGPA.glm = glm(Acceptance~GPA, data=MedGPA, family = binomial)  
  
B0 = summary(MedGPA.glm)$coef[1]  
B1 = summary(MedGPA.glm)$coef[2]  
  
plot(jitter(Acceptance,amount=0.1)~GPA,data=MedGPA)  
curve(logit(B0, B1, x),add=TRUE, col="red")

 It makes sense that teh logit model fits here, the ost people who get in have high gpa it looks like its a jittered acceptabnce rate we dont know other things about these people, there are probablyother reasons they arent getting or they gota ccepted into med school

below we are shorting based on how good their gpa is, this is just ordering it by gpa; so its going to look at the levels **Create a new dataframe with the predictor sorted smallest to largest**

sorted.MedGPA = MedGPA[order(MedGPA$GPA),]  
GPA = sorted.MedGPA$GPA  
Acceptance = sorted.MedGPA$Acceptance  
  
#we want to pull out GPA so we can just work with that; so we do gpa = sorted.medgpa$gpa   
#then we also want to know if thery got acepted, so see teh Acceptance object in R   
  
# Select a number of “slices” or groups for the data and find the mean value of the predictor for each slice  
# WE slect slices so we can look at the mean of the groups   
# We want to see if our model follows the means of the groups well   
  
groups = 5  
group.size = 11  
  
GPA.means = 0  
Acceptance.sums = 0  
  
for(i in 1:groups){  
 GPA.means[i] = mean(  
 GPA[((group.size\*i)-(group.size-1)):(group.size\*i)])  
 }  
  
GPA.means

## [1] 3.130909 3.410000 3.585455 3.734545 3.905455

above does a loop where it takes teh mean of teh first 11 elements, tehn it’s teh second 11 elements; so its not easy to read this, there is a nicer way to do this:

library(TTR)  
runMean(GPA, 11)

## [1] NA NA NA NA NA NA NA NA  
## [9] NA NA 3.130909 3.189091 3.240909 3.270909 3.297273 3.319091  
## [17] 3.334545 3.349091 3.366364 3.380000 3.396364 3.410000 3.426364 3.441818  
## [25] 3.457273 3.473636 3.490000 3.506364 3.525455 3.541818 3.558182 3.570909  
## [33] 3.585455 3.597273 3.610000 3.623636 3.636364 3.650000 3.661818 3.671818  
## [41] 3.685455 3.699091 3.717273 3.734545 3.751818 3.769091 3.786364 3.803636  
## [49] 3.820000 3.836364 3.851818 3.864545 3.880000 3.893636 3.905455

the above will do the same thing as teh loop, but it’s from the TTR package this gives us all of the 11, we jsut want the 11th, 22th, 33rd, and 44th values; how do we get that? look below

runMean(GPA, 11)[seq(11,length(GPA),11)]

## [1] 3.130909 3.410000 3.585455 3.734545 3.905455

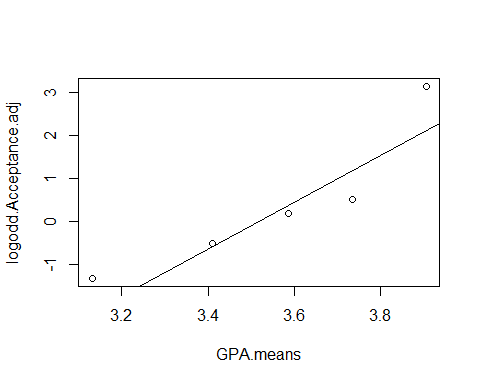
THe above gives you what we want

Accepted sums **Find the number of acceptances for each slice**

for(i in 1:groups){  
 Acceptance.sums[i] =sum(   
 Acceptance[((group.size\*i)-(group.size-1)):(group.size\*i)])  
 }  
  
Acceptance.sums

## [1] 2 4 6 7 11

# “Fudge” the proportions slightly and find the log of the predicted odds  
# Why “fudge”?  
# Proportions of 0 and 1 cause issues.  
  
Acceptance.prop.adj = (Acceptance.sums +0.5)/(group.size+1)  
  
logodd.Acceptance.adj = log(Acceptance.prop.adj/(1-Acceptance.prop.adj))  
  
# Plot the logodds of the adjusted proportions by the means of the predictor variables and a linear model  
# ASk self if the data ppears linear and if the group numbers matter (YES! THEY DO)  
plot(logodd.Acceptance.adj~GPA.means)  
abline(B0,B1)

 above does it in loop format, but you can do it better with teh ttr package

acceptance.sums = runSum(Acceptance, 11)  
acceptance.sums

## [1] NA NA NA NA NA NA NA NA NA NA 2 2 2 2 3 2 2 2 2 2 3 4 5 6 6  
## [26] 6 6 6 6 6 7 7 6 5 4 5 4 5 5 6 7 6 6 7 8 9 9 10 10 10  
## [51] 10 10 11 11 11

THis is saysing tha ttherea re not enought o give me an 11 people sum; the above gives you a running cumumlaitve sum for 11, then 12, then 13 then etc

so to get what we want do below:

acceptance.sums = runSum(Acceptance, 11)[seq(11,length(GPA), 11)]  
acceptance.sums

## [1] 2 4 6 7 11

we need to fudge so we tell R to nto give us exactly 0 and not eactly 1 we get log of 0 andlog of 1 aer issues;

we are just goint o add 0.5 to the sums and divivde by the group size sot aht we never get exactly 1 or a little less than that.

look at teh group size and numbers way above.

WE’re a bit lost because he deviated adn I’m not paying as much attention as I could

acceptance.propo.ad = (Acceptance.sums + 0.5)/(group.size + 1)  
acceptance.propo.ad

## [1] 0.2083333 0.3750000 0.5416667 0.6250000 0.9583333

logodd.accept.ad = log(acceptance.propo.ad/(1-acceptance.propo.ad))  
logodd.accept.ad

## [1] -1.3350011 -0.5108256 0.1670541 0.5108256 3.1354942

below, we want to make sure we load stat2data

we give it teh raw data, so that it does the thing taht we want itt to; we want to group it by 5, but i dont know why

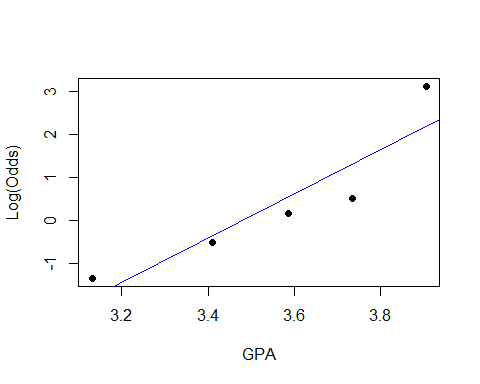
we choose 5 because that’s how we split teh data earlier in the code;

this will give you the same plot, but we the othere xtra work in teh past; its a fsater way to do the thing

there may be issues with this on teh data; if you try and slice the data ti migth overlap and cause errors; certain groupsing might work differently for different data; it defpending on teh datatset

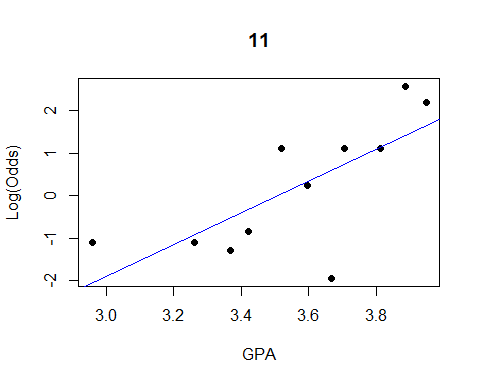
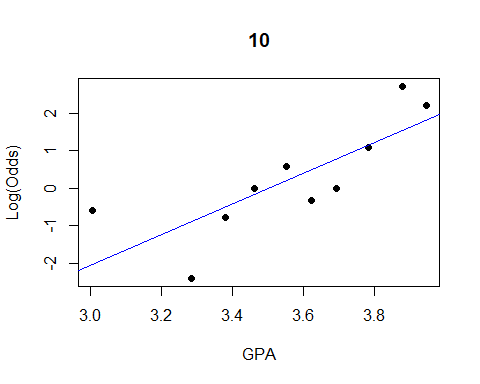
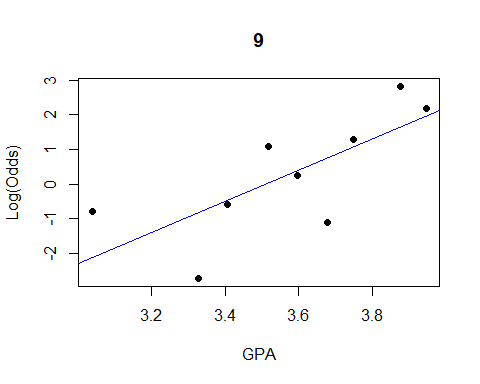
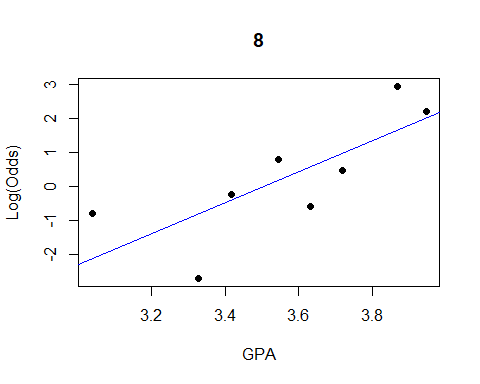
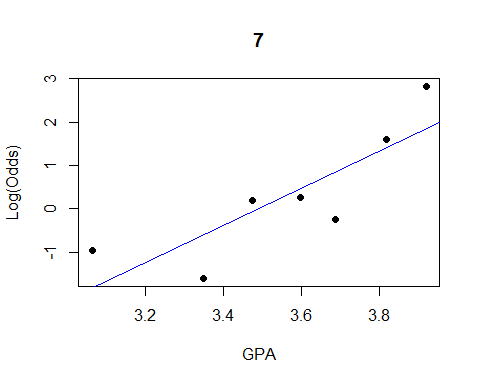
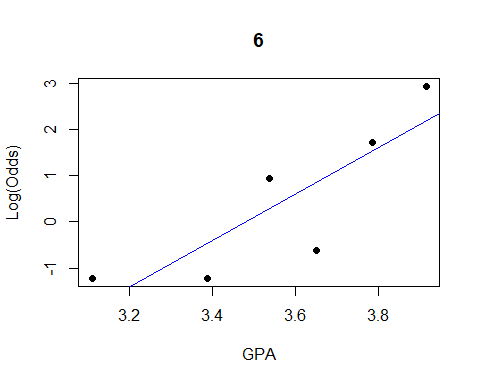
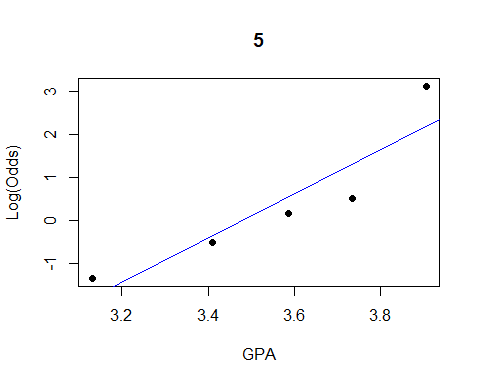
you just haev to ttiral and error it **Check linear conditions**

emplogitplot1(Acceptance~GPA, data=MedGPA, ngroups=5)

 this will check the groups of the other names; this is how you test other nimber grouping, which is useful

these arent residuals, but we think about them that way; if tehre are different patterns, we could try transformations; the logit plot works teh same as teh full things we did in the other classes **This checks different grouping types**

for(j in 5:11){emplogitplot1(Acceptance~GPA, data=MedGPA, ngroups=j, main=j)}



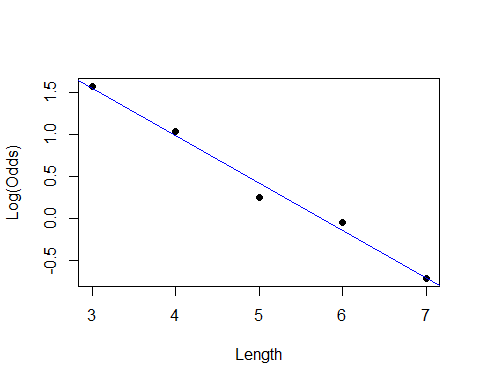
below shows you a shortcut on how to do the long thing int eh short; with teh logitplot function

ngroups = all shows that all possible predictor wvalue and will make teh own group

that doesnt work for medgpa data beause the grouping are different; there are vary few outcomes for each logodd for med gpa and it doesnt tell us much aout ht eoutcomes

the goldenballgraph is less exciting because it’s a graph with a thing between two lines

data("Putts1")  
emplogitplot1(Made~Length, data=Putts1, ngroups="all")



emplogitplot1(Split~Over40, data=GoldenBalls, ngroups="all")

