StatR 201 HW 3 Key

Scott Rinnnan and Gregor Thomas

Thursday, Jan 29, 2015

Contents


```
library(xlsx)
library(magrittr)
library(reshape2)
library(dplyr)
library(stringr)
library(ggplot2)
library(MASS)
library(arm)
library(boot)
setwd("~/Dropbox/STATR 201/Week 3")
```

Problem 1: Webscraping

(a) We start by standardizing the county names to match the xlsx files:

```
counties = c("Adams", "Asotin", "Benton", "Chelan", "Clallam", "Clark", "Columbia",
    "Cowlitz", "Douglas", "Ferry", "Franklin", "Garfield", "Grant", "Grays Harbor",
    "Island", "Jefferson", "King", "Kitsap", "Kittitas", "Klickitat", "Lewis",
    "Lincoln", "Mason", "Okanogan", "Pacific", "Pend Oreille", "Pierce", "San Juan",
    "Skagit", "Skamania", "Snohomish", "Spokane", "Stevens", "Thurston", "Wahkiakum",
    "Walla Walla", "Whatcom", "Whitman", "Yakima")

counties <- tolower(counties)
counties <- gsub(" ", "", counties)
cts <- substr(counties, 1, 4)
head(cts)
```

```
## [1] "adam" "asot" "bent" "chel" "clal" "clar"
```

(b)

Create the urls by appending the county strings to the base url:

```r
base.url <- "http://www.ofm.wa.gov/sac/cjdatabook/"

urls<-NULL
for(i in 1:length(cts)){
  urls[i]<-paste0(base.url,cts[i],".xlsx")
}
head(urls)
```

```
## [1] "http://www.ofm.wa.gov/sac/cjdatabook/adam.xlsx"
## [2] "http://www.ofm.wa.gov/sac/cjdatabook/asot.xlsx"
## [3] "http://www.ofm.wa.gov/sac/cjdatabook/bent.xlsx"
## [4] "http://www.ofm.wa.gov/sac/cjdatabook/chel.xlsx"
## [5] "http://www.ofm.wa.gov/sac/cjdatabook/clal.xlsx"
## [6] "http://www.ofm.wa.gov/sac/cjdatabook/clar.xlsx"
```

(c)

I created the folder manually, but it can also be created using the `dir.create()` function. We can now download the files into that directory:

```r
for(i in 1:length(urls)){
  download.file(urls[i],destfile=paste0("Counties/",cts[i],".xlsx"))
}
```

You should now see the county xlsx files downloaded in your specified directory.

(d)

Pulling out the crime data:

```r
getCrime<-function(File){
  read.xlsx(File,
            sheetIndex = 1,
            rowIndex = 18:26,
            header = T,
            colClasses = "character",
            stringsAsFactors = FALSE)
}
```

(e)

Creating a list in which to store the crime data:

```r
ctyfiles<-list.files("Counties",pattern=".xlsx",full.names=T)
crime.dat<-list()
for(i in 1:length(ctyfiles)){
  crime.dat[[i]]<-getCrime(ctyfiles[i])
}
```

(f)

Combining the data:

```
crime<-do.call(rbind,crime.dat)
head(crime)
```

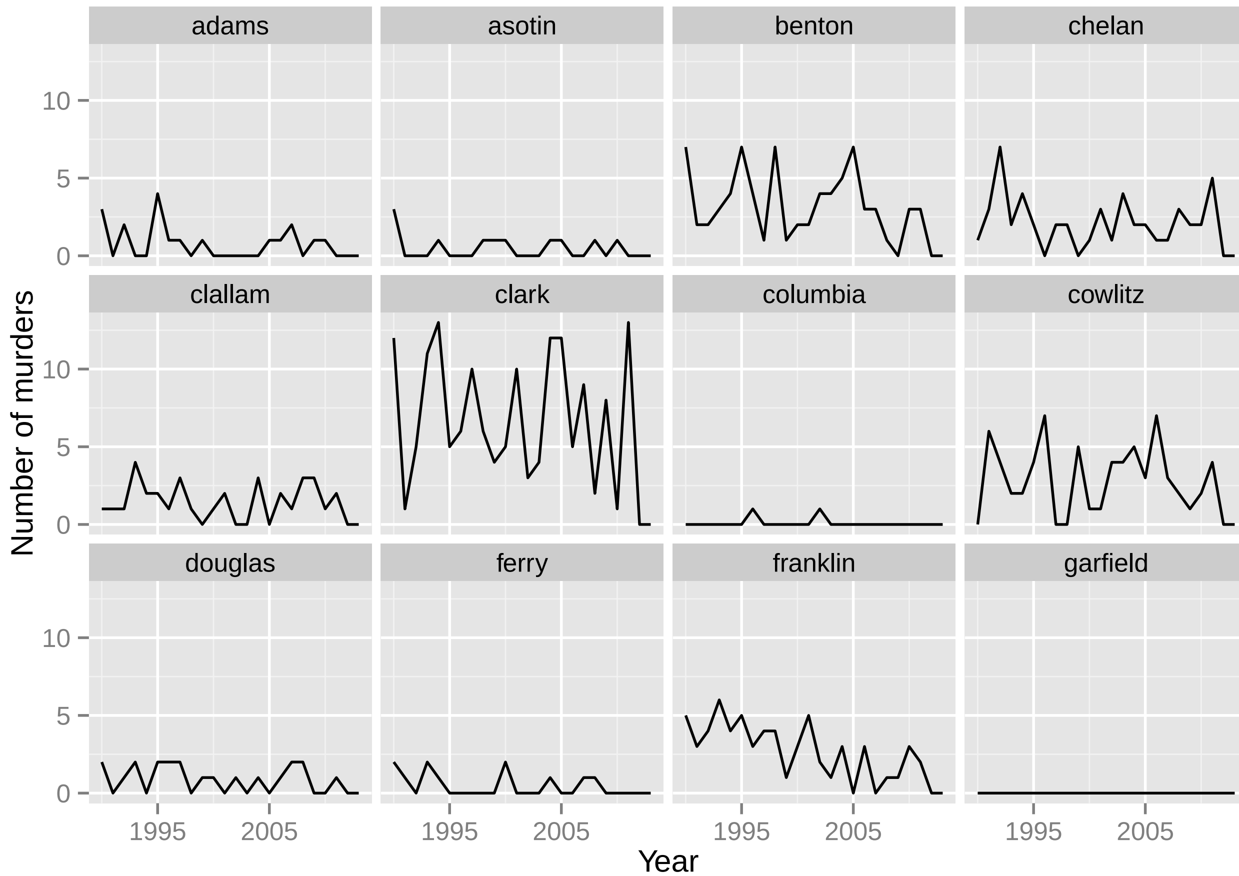
```
##        Calendar.Year X1990 X1991 X1992 X1993 X1994 X1995 X1996 X1997 X1998
## 1             Murder     3     0     2     0     0     4     1     1     0
## 2      Forcible Rape    14     6    13     3     5     5     6     9    10
## 3 Aggravated Assault    80    52    28    45    57    34    42    46    32
## 4            Robbery     4     2     2     3     5     3     5     2     5
## 5              Arson     3     0     9     6     7     3    11     8    15
## 6           Burglary   136   143   171   141   132   150   142   185   205
##   X1999 X2000 X2001 X2002 X2003 X2004 X2005 X2006 X2007 X2008 X2009 X2010
## 1     1     0     0     0     0     0     1     1     2     0     1     1
## 2     4    14    12     5    16     9    14     7     9     9     6     8
## 3    42    21    25    16    22    41    29    24    26    43    31    37
## 4     4     5     7     4     4    11    13     1     4     5    19    10
## 5     6     7    10     8     8     8     7     5     9     5    10     6
## 6   168   161   109   173   226   265   158   172   255   183   185   229
##   X2011 X2012 X2013
## 1     0     0     0
## 2     4     0     0
## 3    39     0     0
## 4    10     0     0
## 5     9     0     0
## 6   209     0     0
```

(g)

I picked the first twelve counties for simplicity, but you are of course welcome to choose them however you like:

```
crimesub<-crime[1:96,]
crimesub %<>% subset(Calendar.Year=="Murder")
crimesub$County<-counties[1:12]
crimesub %<>%
  melt(id = c("Calendar.Year","County"), variable.name = "year", value.name = "count") %>%
  mutate(crimesub = factor(Calendar.Year),
         year = as.numeric(str_replace(year, "X", "")))

p<-ggplot(crimesub, aes(x = year, y = count))
p + geom_line() +
  facet_wrap(~ County) +
  scale_x_continuous("Year",breaks=c(1995,2005)) +
  ylab("Number of murders")
```



Problem 2: Obesity data

(a)

Reading in and cleaning up the obesity data:

```
obese<-read.csv("obese11.csv",colClasses="factor")
obese<-obese[,-1] #let's remove id column, since we really don't need it
obese$age %<>% as.integer
obese %<>% na.omit
```

Let's start by making a model with all the variables considered:

```
mod1<-glm(obese~.,family=binomial,data=obese)
summary(mod1)
```

```
##
## Call:
## glm(formula = obese ~ ., family = binomial, data = obese)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.2216  -0.3198  -0.1668  -0.0001   3.1489
##
```

```
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -3.16712    0.21257 -14.899  < 2e-16 ***
## age            -0.03097    0.03278  -0.945  0.34486
## female1        -1.56650    0.08803 -17.795  < 2e-16 ***
## demog2Black     1.06167    0.11539   9.201  < 2e-16 ***
## demog3Latino    0.49326    0.09512   5.186 2.15e-07 ***
## demog4NatAm     0.54942    0.25865   2.124  0.03366 *
## demog5Asian    -0.45415    0.26463  -1.716  0.08613 .
## demogOthUknwn   0.23017    0.16430   1.401  0.16125
## active51       -0.12468    0.08885  -1.403  0.16054
## active01        0.08311    0.11489   0.723  0.46944
## screen3TRUE     0.11451    0.07966   1.437  0.15058
## image1          3.93602    0.09442  41.688  < 2e-16 ***
## breakfast1     -0.01080    0.08561  -0.126  0.89960
## sleep81         0.25538    0.08762   2.915  0.00356 **
## schoolTalk1     0.01466    0.08616   0.170  0.86489
## smoke201        0.06265    0.16622   0.377  0.70625
## frveg51         0.29069    0.12290   2.365  0.01802 *
## veg31           0.10131    0.14227   0.712  0.47641
## overwt1       -19.15448  230.44331  -0.083  0.93376
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8054.9  on 10046  degrees of freedom
## Residual deviance: 4455.2  on 10028  degrees of freedom
## AIC: 4493.2
##
## Number of Fisher Scoring iterations: 18
```

Looks like a lot of the variables aren't contributing that much. After futzing around for a bit, here's a better one I found:

```r
mod2 <- glm(obese ~ female + demog + active5 + screen3 + image + sleep8 + frveg5,
    family = binomial, data = obese)
summary(mod2)
```

```
##
## Call:
## glm(formula = obese ~ female + demog + active5 + screen3 + image +
##     sleep8 + frveg5, family = binomial, data = obese)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6914  -0.3586  -0.2556  -0.1494   3.0835
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -3.25833    0.10417 -31.279  < 2e-16 ***
## female1       -1.31291    0.07346 -17.874  < 2e-16 ***
## demog2Black    0.84595    0.10057   8.411  < 2e-16 ***
```

```
## demog3Latino   0.30814    0.08263   3.729 0.000192 ***
## demog4NatAm    0.22388    0.22098   1.013 0.310999
## demog5Asian   -0.42542    0.23896  -1.780 0.075023 .
## demogOthUknwn  0.13582    0.14503   0.937 0.349007
## active51      -0.17421    0.07232  -2.409 0.016007 *
## screen3TRUE    0.08203    0.07040   1.165 0.243956
## image1         3.21741    0.08334  38.604  < 2e-16 ***
## sleep81        0.14219    0.07541   1.886 0.059357 .
## frveg51        0.27005    0.08123   3.324 0.000886 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8054.9  on 10046  degrees of freedom
## Residual deviance: 5629.3  on 10035  degrees of freedom
## AIC: 5653.3
##
## Number of Fisher Scoring iterations: 6
```

```r
AIC(mod1, mod2)
```

```
##      df      AIC
## mod1 19 4493.180
## mod2 12 5653.276
```

Despite the jump in AIC, I don't think it makes much sense to include overweight as a predictor of obesity, since they are mutually exclusive.

Including an interaction term, we can do even better:

```r
mod3 <- glm(obese ~ female * demog + active5 + screen3 + image + sleep8 + frveg5,
    family = binomial(link = "logit"), data = obese)
summary(mod3)
```

```
##
## Call:
## glm(formula = obese ~ female * demog + active5 + screen3 + image +
##     sleep8 + frveg5, family = binomial(link = "logit"), data = obese)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5537  -0.3543  -0.2671  -0.1435   3.2862
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -3.16808    0.10900 -29.066  < 2e-16 ***
## female1               -1.47482    0.10960 -13.456  < 2e-16 ***
## demog2Black            0.47858    0.14408   3.322 0.000895 ***
## demog3Latino           0.32696    0.10887   3.003 0.002671 **
## demog4NatAm           -0.08818    0.30389  -0.290 0.771699
## demog5Asian           -0.21873    0.28269  -0.774 0.439077
## demogOthUknwn         -0.20282    0.20714  -0.979 0.327498
```

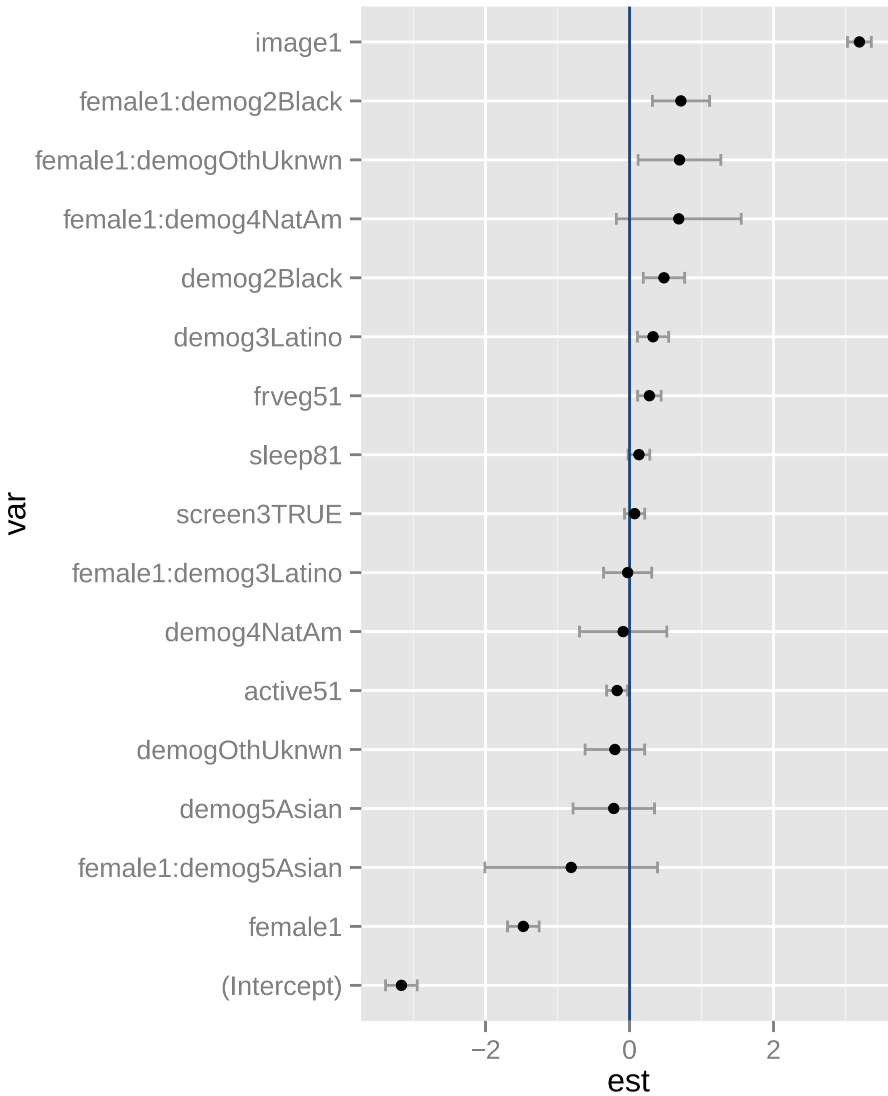
```
## active51              -0.17205    0.07260  -2.370 0.017797 *
## screen3TRUE            0.07141    0.07071   1.010 0.312549
## image1                 3.19326    0.08279  38.572  < 2e-16 ***
## sleep81                0.13193    0.07570   1.743 0.081388 .
## frveg51                0.27638    0.08155   3.389 0.000701 ***
## female1:demog2Black    0.71416    0.19862   3.596 0.000324 ***
## female1:demog3Latino  -0.02539    0.16755  -0.152 0.879533
## female1:demog4NatAm    0.68401    0.43388   1.577 0.114909
## female1:demog5Asian   -0.80980    0.59892  -1.352 0.176346
## female1:demogOthUknwn  0.69441    0.28745   2.416 0.015702 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 8054.9  on 10046  degrees of freedom
## Residual deviance: 5605.2  on 10030  degrees of freedom
## AIC: 5639.2
##
## Number of Fisher Scoring iterations: 6
```

```
AIC(mod2, mod3)
```

```
##      df      AIC
## mod2 12 5653.276
## mod3 17 5639.238
```

We can make a ladder plot of our model:

```
mod3coef <- data.frame(var = names(coef(mod3)), est = coef(mod3), se = se.coef(mod3))  #arm::se.coef
mod3coef$var %<>% reorder(mod3coef$est)
p <- ggplot(mod3coef, aes(x = est, y = var))
p + geom_errorbarh(aes(xmin = est - 2 * se, xmax = est + 2 * se), height = 0.2,
    color = "gray60") + geom_vline(xintercept = 0, color = "dodgerblue4") +
    geom_point()
```



(b)

The addition of interaction effects allows a different line to be fit to each demographic, each with its own slope. Since we see major differences between the demographics (see lecture), this seems like a pretty reasonable thing to do. Based on AIC values, we would expect model 3 to have better predictive accuracy than model 2.

G&H #3:

This is really just an algebra problem. Given two point, find the formula for the line that connects them. Putting the two points into our equation, we have:

$$\text{logit}(.27) = a + b \cdot 0$$
$$\text{logit}(.88) = a + b \cdot 6.$$

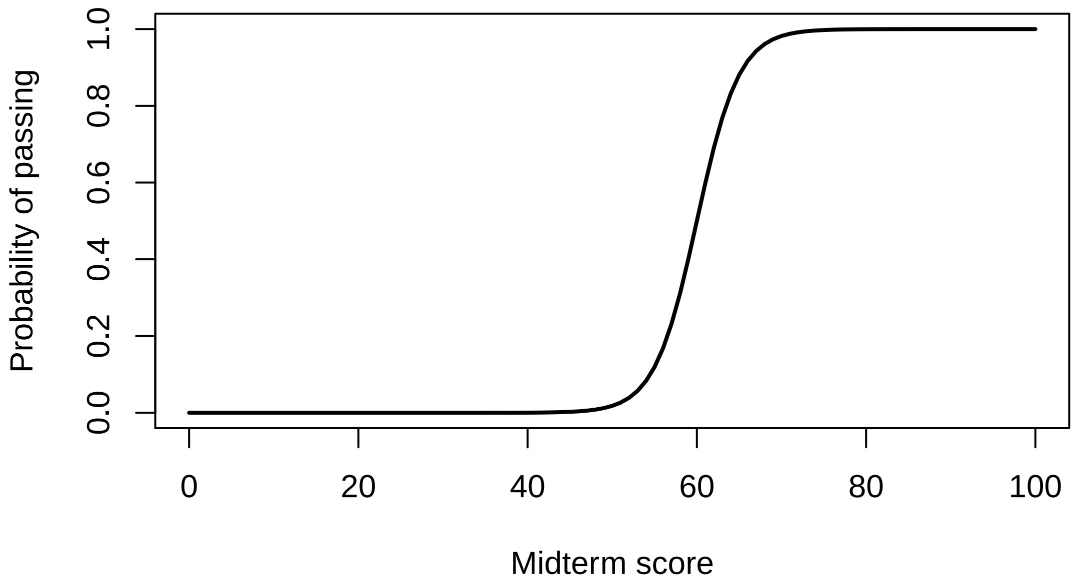
Solving the first equation for a, we get $a = \text{logit}(.27) \approx -1$. Putting that into the second equation, we get $b = (\text{logit}(.88) + 1)/6 \approx 0.5$. Hence, our model is $\text{logit}(p) = -1 + 0.5x$, where x is income in units of \$10,000.

G&H #5:

(a)

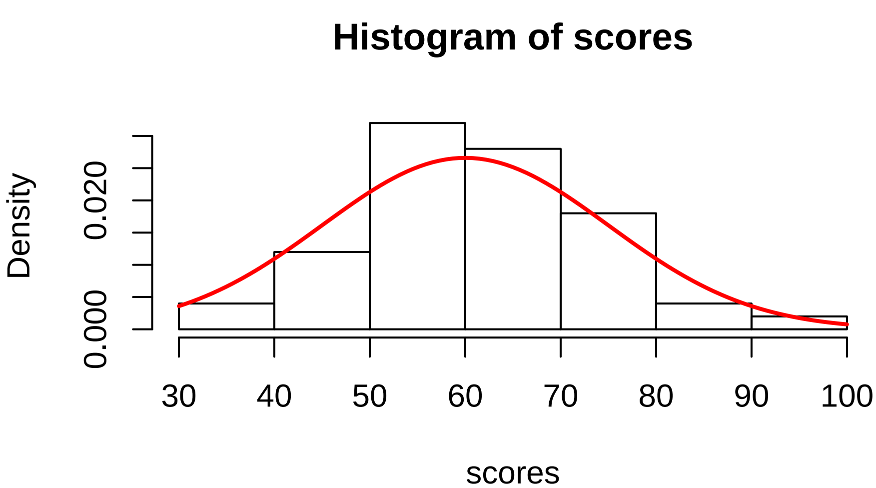
Let's start by taking a look at the plot:

```
curve(inv.logit(-24+.4*x),from=0,to=100,lwd=2,
      xlab="Midterm score",ylab="Probability of passing") #boot::inv.logit()
```



As you can see from the graph, the higher a student's score on the midterm, the higher the probability that they will pass the class. We would like to simulate some data that fits this model. We are told that the scores or normally distributed with $\mu = 60$, $\sigma = 15$. We will begin by sampling 50 test scores from this distribution. (I will set a seed so that my results are reproducible.)

```
set.seed(100)
scores<-rnorm(50,60,15)
hist(scores,prob=T)
curve(dnorm(x,60,15),add=T,col=2,lwd=2)
```



So we see that the simulated scores indeed fit the distribution. Now we can take those simulated test scores and calculate the students' probability of passing the class:

```
pscores<-inv.logit(-24+.4*scores)
```

These numbers give us the probability that a student will pass, but in actuality, each student really only has one of two outcomes: they either pass or they don't. One way we can assign a pass or fail to each student is to just randomly sample from `c(0,1)` with probability weights that reflect each student's probability of passing:

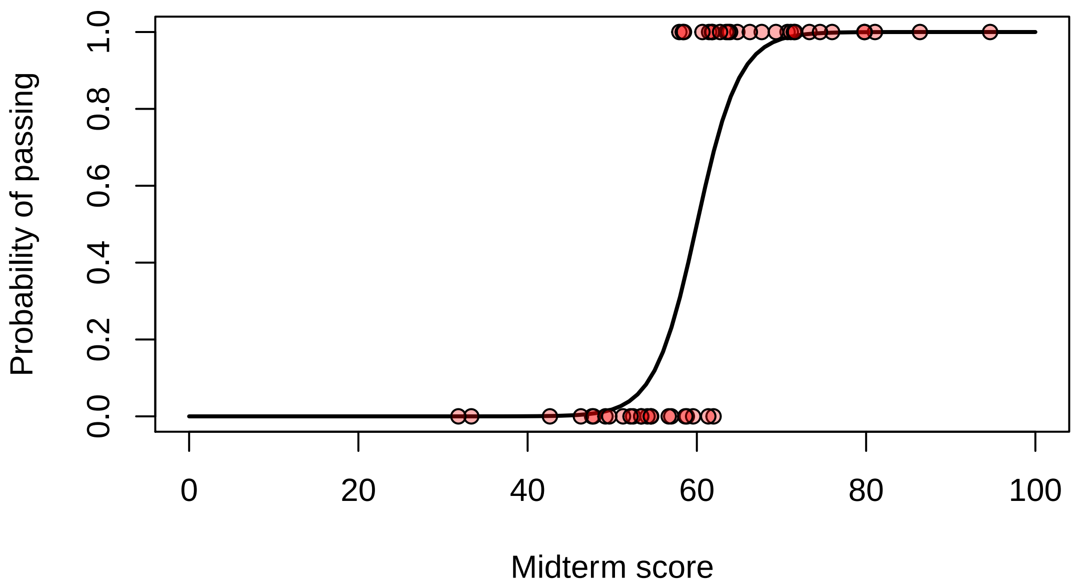
```
results<-NULL
for(i in 1:50){
  results[i]<-sample(0:1,1,prob=c(1-pscores[i],pscores[i]))
}
```

Or, much more succinctly:

```
results<-rbinom(50,1,pscores)
```

Now we can finally add these results to our model fit:

```
curve(inv.logit(-24+.4*x),from=0,to=100,lwd=2,
      xlab="Midterm score",ylab="Probability of passing")
points(scores,results,bg="#FF000050",pch=21)
```



(b)

The logistic regression should not change under standardization of the data. The probability of passing will be the same regardless of whether a student's score has been transformed or if it's the original raw score.

```
zscores<-(scores-mean(scores)/sd(scores))
mod1<-glm(results~scores,family=binomial)
mod2<-glm(results~zscores,family=binomial)
summary(mod1)
```

```
##
```

```
## Call:
## glm(formula = results ~ scores, family = binomial)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.85730  -0.15759   0.00109   0.15949   1.60521
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -36.7152    12.8318  -2.861  0.00422 **
## scores        0.6171     0.2149   2.872  0.00408 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 68.994  on 49  degrees of freedom
## Residual deviance: 20.705  on 48  degrees of freedom
## AIC: 24.705
##
## Number of Fisher Scoring iterations: 8
```

```r
summary(mod2)
```

```
##
## Call:
## glm(formula = results ~ zscores, family = binomial)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.85730  -0.15759   0.00109   0.15949   1.60521
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -33.6399    11.7621  -2.860  0.00424 **
## zscores       0.6171     0.2149   2.872  0.00408 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 68.994  on 49  degrees of freedom
## Residual deviance: 20.705  on 48  degrees of freedom
## AIC: 24.705
##
## Number of Fisher Scoring iterations: 8
```

(c)

Adding some noise:

```r
newpred<-rnorm(50,0,1)
mod3<-glm(results~scores+newpred,family=binomial)
summary(mod3)
```

```
##
## Call:
## glm(formula = results ~ scores + newpred, family = binomial)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.66137  -0.13811   0.00017   0.14535   1.98950
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -41.6271    15.6969  -2.652  0.00800 **
## scores        0.6992     0.2630   2.658  0.00786 **
## newpred      -1.0287     1.0618  -0.969  0.33262
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 68.994  on 49  degrees of freedom
## Residual deviance: 19.638  on 47  degrees of freedom
## AIC: 25.638
##
## Number of Fisher Scoring iterations: 8
```

We observe no change in the amount of null deviance, because the new predictor only adds noise to the model, and not any meaningful new information. There is a small amount of decrease in the residual deviance.