MATH523 Assignment 3

Kabilan Sriranjan

March 15, 2018

A8

a)

Treating the data as Binomial we have as the log-likelihood

$$\sum_{i=1}^{N} log\binom{n_i}{y_i} + y_i log(\pi_i) + (n_i - y_i) log(1 - \pi_i)$$

And then for Bernoulli we get

$$\begin{aligned} &\sum_{i=1}^{N}\sum_{j=1}^{n_i} y_{ij} log(\pi_i) + (1 - y_{ij}) log(1 - \pi_i) \\ =&\sum_{i=1}^{N} y_i log(\pi_i) + (n_i - y_i) log(1 - \pi_i) \end{aligned}$$

When differentiated, you get the same derivatives, and hence the same score equations. Thus the estimates are identical.

b)

In the saturated model there are as many parameters as there are observations. When we interpret the observations as Bernoulli then there are n observations, and thus n parameters. However interpreting the observations as Binomial means that the saturated model has only N parameters. Thus considering the observations as Binomial or Bernoulli affects what the saturated model is, and hence the deviance.

c)

```
a.bin <- c(0,1,3)
bin <- cbind(c(1,2,4),c(2,2,1))
a.bern <- c(0,0,0,1,1,1,1,3,3,3,3,3)
bern <- cbind(c(1,0,0,1,1,0,0,1,1,1,1,0), c(0,1,1,0,0,1,1,0,0,0,0,1))
mbin <- glm(bin~a.bin, family=binomial)
mbin.int <- glm(bin~1, family=binomial)
mbern <- glm(bern~a.bern, family=binomial)
mbern.int <- glm(bern~1, family=binomial)
summary(mbin)
```

```
##
## Call:
## glm(formula = bin ~ a.bin, family = binomial)
##
## Deviance Residuals:
## 1  2  3
## 0  0  0
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.6931     0.9727  -0.713    0.476
## a.bin         0.6931     0.5335   1.299    0.194
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance:  1.9324e+00  on 2  degrees of freedom
## Residual deviance: -2.9582e-31  on 1  degrees of freedom
## AIC: 9.3687
##
## Number of Fisher Scoring iterations: 4
```

```
summary(mbin.int)
```

```
##
## Call:
## glm(formula = bin ~ 1, family = binomial)
##
## Deviance Residuals:
##       1        2        3
## -0.8722  -0.3357   1.0290
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.3365     0.5855   0.575    0.566
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1.9324  on 2  degrees of freedom
## Residual deviance: 1.9324  on 2  degrees of freedom
## AIC: 9.301
##
## Number of Fisher Scoring iterations: 4
```

```
summary(mbern)
```

```
##
## Call:
## glm(formula = bern ~ a.bern, family = binomial)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7941  -0.9697   0.6681   0.7954   1.4823
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)  -0.6931     0.9727  -0.713    0.476
## a.bern        0.6931     0.5335   1.299    0.194
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 16.301  on 11  degrees of freedom
## Residual deviance: 14.368  on 10  degrees of freedom
## AIC: 18.368
##
## Number of Fisher Scoring iterations: 4
```

```
summary(mbern.int)
```

```
##
## Call:
## glm(formula = bern ~ 1, family = binomial)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.323  -1.323   1.038   1.038   1.038
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.3365     0.5855   0.575    0.566
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 16.301  on 11  degrees of freedom
## Residual deviance: 16.301  on 11  degrees of freedom
## AIC: 18.301
##
## Number of Fisher Scoring iterations: 4
```

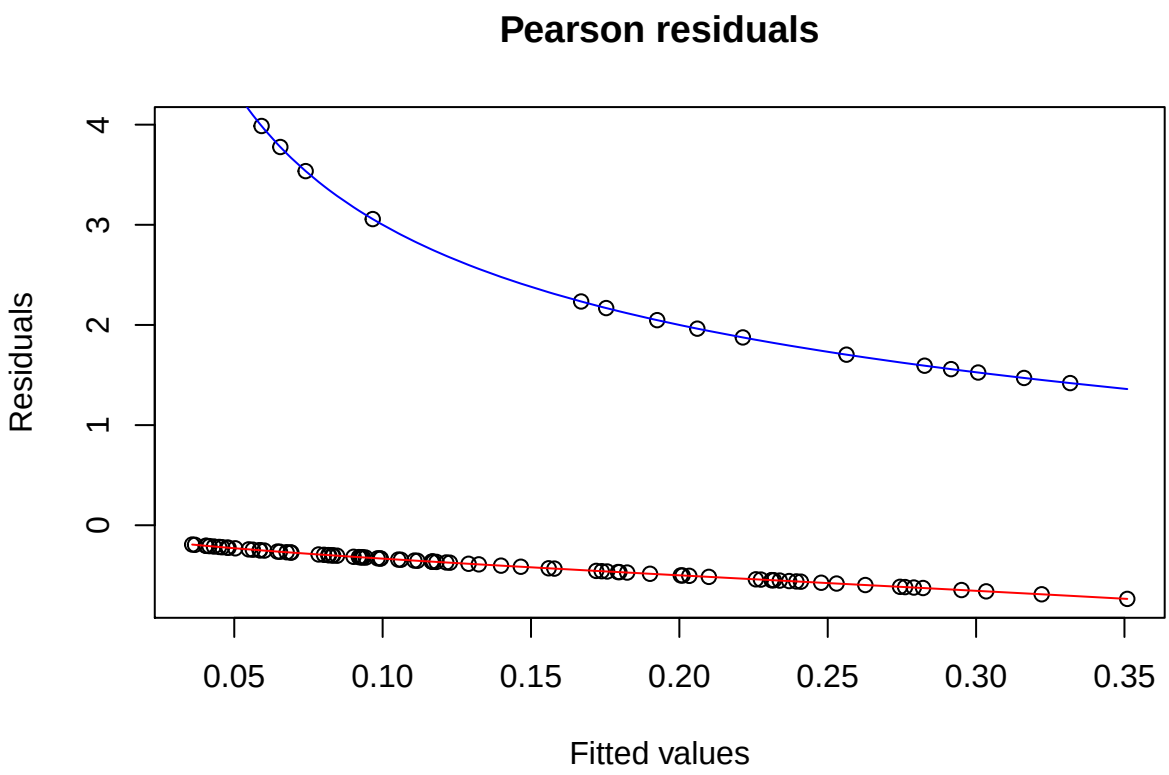
As we can see the parameter estimates are the same no matter how we enter the data. The deviance seems to increase when you have only the intercept.

A9

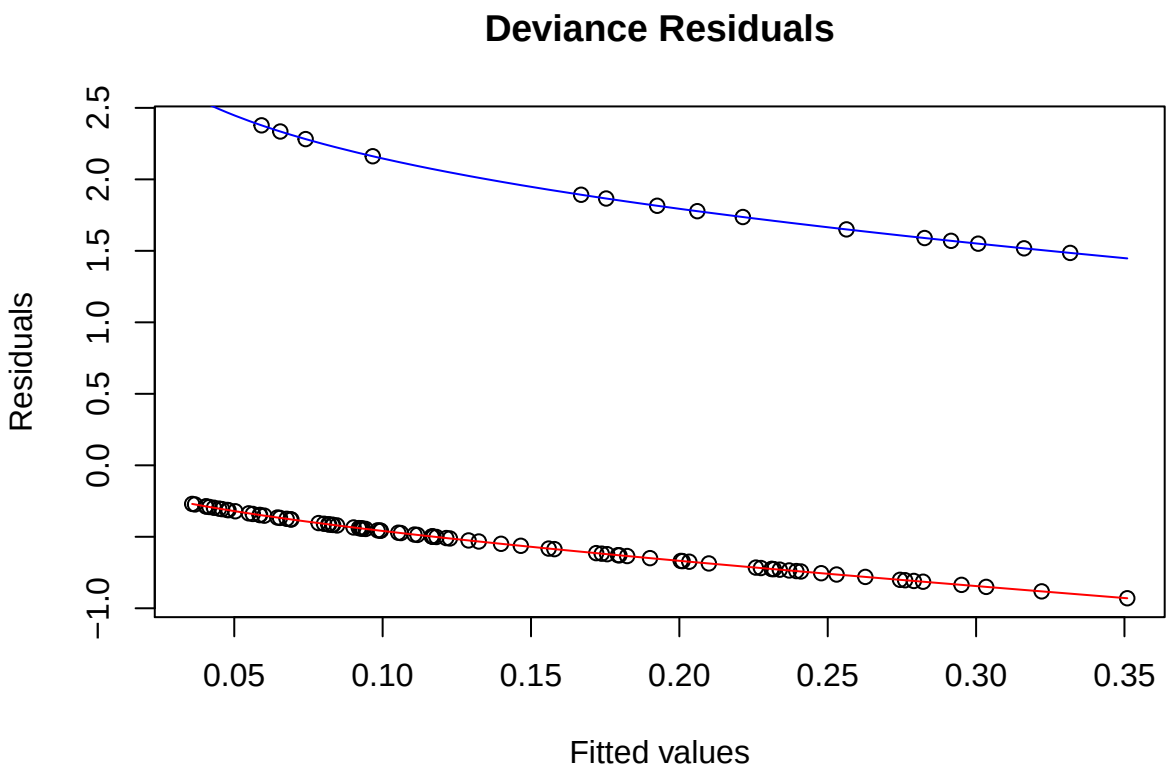
a)

```
set.seed(1)
n <- 100
x <- runif(n, 0, 1)
logit.pi <- 0.4*x - 2
pi <- exp(logit.pi)/(exp(logit.pi)+1)
y <- rbinom(n, 1, pi)


model <- glm(y~x, family=binomial)
fit <- fitted(model)
beta <- coefficients(model)

sx <- sort(x)
```

```r
seta <- beta[1]+beta[2]*sx
sfit <- exp(seta)/(1+exp(seta))

pres <- residuals(model, "pearson")
plot(fit, pres, xlab="Fitted values", ylab="Residuals", main="Pearson residuals")
lines(sfit, (1-sfit)/(sqrt(sfit*(1-sfit))), col="blue")
lines(sfit, (0-sfit)/(sqrt(sfit*(1-sfit))) ,col="red")
```



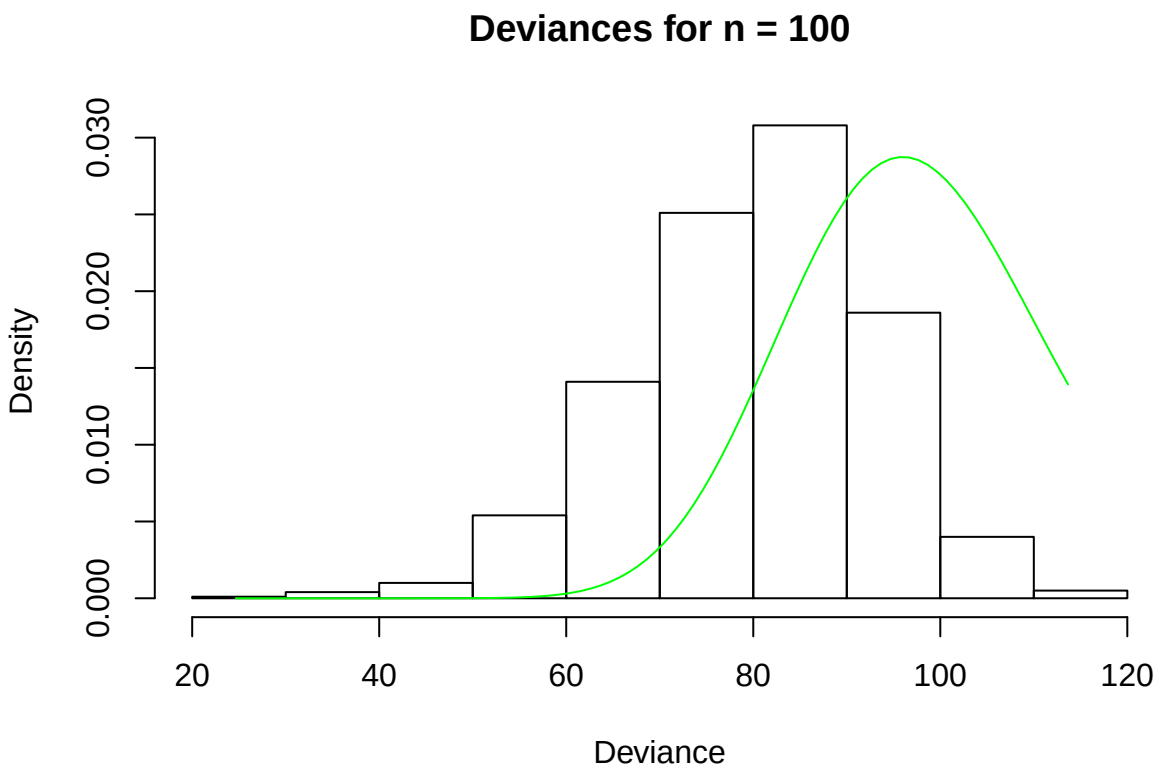

```r
dres <- residuals(model, "deviance")
plot(fit, dres, xlab="Fitted values", ylab="Residuals", main="Deviance Residuals")
lines(sfit, sqrt(2*log(1/sfit)), col="blue")
lines(sfit, -sqrt(2*log(1/(1-sfit))), col="red")
```



Our residual plots look like this because our reponse is binary. One of the curves correspond to when the response is 0 and the other for when the response is 1. We can find the equations of the lines by taking the formula for the residual and replacing y_i with either 0 or 1.

b)

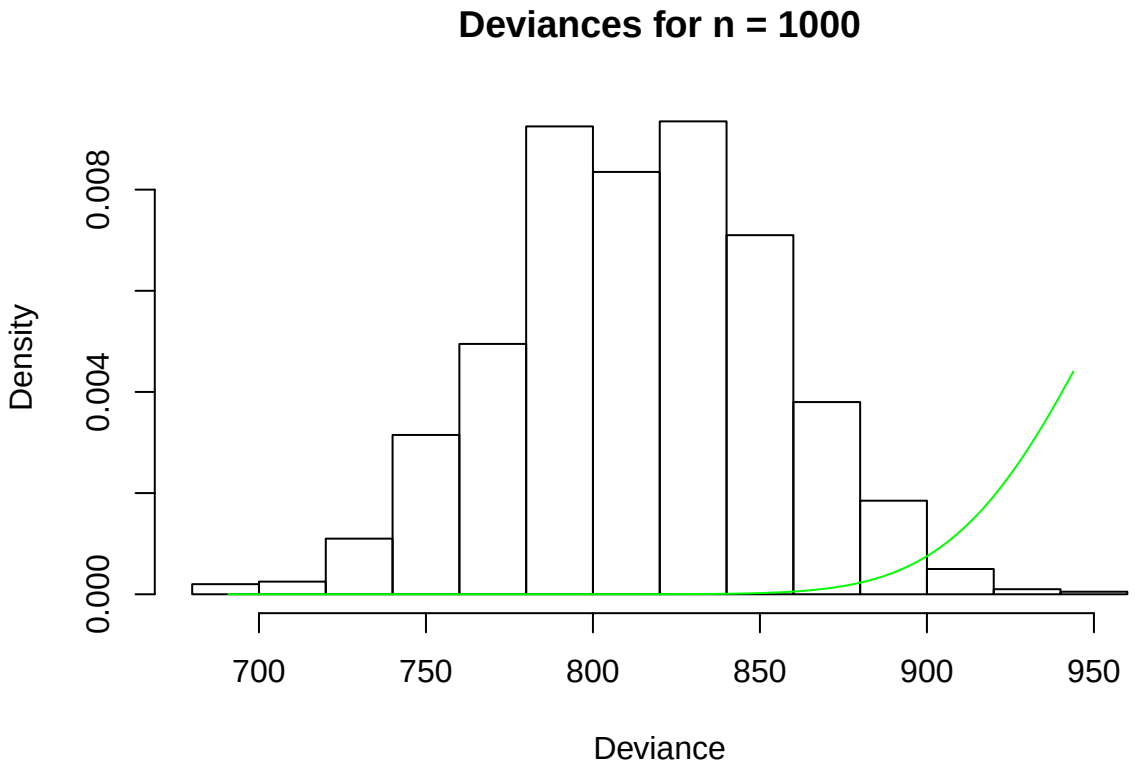
```
getHistogram <- function(n, s){
  set.seed(s)
  k <- n-2
  N <- 1000
  x <- runif(n, 0, 1)
  logit.pi <- 0.4*x - 2
  pi <- exp(logit.pi)/(exp(logit.pi)+1)
  D <- c()
  for (i in 1:N){
    y <- rbinom(n, 1, pi)
    M = glm(y~x, family=binomial)
    D[i] <- deviance(M)
  }
  hist(D, xlab="Deviance", main=paste("Deviances for n =",n), freq=FALSE)
  nodes <- min(D):max(D)
  lines(nodes, dchisq(nodes, k) ,col="green")
}
getHistogram(100, 1)
```



The histogram shows that the χ^2 pdf doesn't seem to model the deviance well. If we believed the deviance to be χ^2 then a lot of actual deviances would be lower than expected.

c)

```
getHistogram(1000, 1)
```



The approximation seems to get worse for larger n. This could be due to the fact that the data is binary.

A10

We have the two inequalities

$$0.8 < \frac{e^{\beta_0+18\beta_1}}{1+e^{\beta_0+18\beta_1 8}} < 0.9$$
$$0.2 < \frac{e^{\beta_0+65\beta_1}}{1+e^{\beta_0+65\beta_1 8}} < 0.3$$

From this we can deduce the inequalities

$$4 < e^{\beta_0+18\beta_1} < 9$$
$$0.25 < e^{\beta_0+65\beta_1} < 0.428$$

Dividing the two inequalities causes the β_0s to cancel out. Taking care to divide the bottom one by the reverse of the top one we get

$$0.027 < e^{47\beta_1} < 0.107$$
$$-3.61 < 47\beta_1 < -2.23$$
$$-0.076 < \beta_1 < -0.047$$

Note that this makes sense because age should have a negative effect on the probability of using social media.

A11

a)

```
wells <- read.table("wells.dat")
attach(wells)
dist100 <- dist/100
m1 <- glm(switch~dist100, family=binomial)
summary(m1)
```

```
##
## Call:
## glm(formula = switch ~ dist100, family = binomial)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4406  -1.3058   0.9669   1.0308   1.6603
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.60596    0.06031  10.047  < 2e-16 ***
## dist100     -0.62188    0.09743  -6.383 1.74e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4118.1  on 3019  degrees of freedom
## Residual deviance: 4076.2  on 3018  degrees of freedom
## AIC: 4080.2
##
## Number of Fisher Scoring iterations: 4
```

```
bins <- seq(floor(min(dist100)),ceiling(max(dist100)),0.2)
counts.succ <- rep(0, length(bins))
m <- rep(0, length(bins))
for (i in 1:length(dist100)){
  low <- 1
  high <- length(bins)
  while (low<=high){
    k <- floor((low+high)/2)
    if (abs(bins[k]-dist100[i])<=0.1){
      if (switch[i]==1){
        counts.succ[k] <- counts.succ[k]+1
      }
      m[k] <- m[k]+1
      break
    } else if (dist100[i]<bins[k]){
      high <- k-1
    } else {
      low <- k+1
    }
  }
```

```
}
ftd <- exp(m1$coefficients[1]+m1$coefficients[2]*bins)/(exp(m1$coefficients[1]+m1$coefficients[2]*bins)+
groupDevs <- counts.succ*log(counts.succ/(m*ftd)) + (m-counts.succ)*log((m-counts.succ)/(m-m*ftd))
groupDevs[13:length(groupDevs)] <- 0
G2 <- 2*sum(groupDevs)
pchisq(G2, df=19, lower.tail=FALSE)
```

```
## [1] 0.4997715
```

```
pearson.test <- sum(residuals(m1, type="pearson")^2)
pchisq(pearson.test, df=3018, lower.tail=FALSE)
```

```
## [1] 0.4837302
```

We have high p-values for both the G^2 and Pearson tests. This leads us to believe that the model is adequate.

b)

```
m2 <- glm(switch ~ dist100+arsenic, family=binomial)
anova(m1, m2, test="Chi")
```

```
## Analysis of Deviance Table
##
## Model 1: switch ~ dist100
## Model 2: switch ~ dist100 + arsenic
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1      3018     4076.2
## 2      3017     3930.7  1   145.57 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
m3 <- glm(switch ~ (dist100+arsenic)^2, family=binomial)
anova(m2, m3, test="Chi")
```

```
## Analysis of Deviance Table
##
## Model 1: switch ~ dist100 + arsenic
## Model 2: switch ~ (dist100 + arsenic)^2
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      3017     3930.7
## 2      3016     3927.6  1   3.0399  0.08124 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
m4 <- glm(switch ~ dist100+arsenic+educ, family=binomial)
anova(m1, m4, test="Chi")
```

```
## Analysis of Deviance Table
##
## Model 1: switch ~ dist100
## Model 2: switch ~ dist100 + arsenic + educ
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1      3018     4076.2
## 2      3016     3910.4  2   165.81 < 2.2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
m5 <- glm(switch ~ dist100+arsenic+educ+dist100*educ, family=binomial)
anova(m4, m5, test="Chi")
```

```
## Analysis of Deviance Table
##
## Model 1: switch ~ dist100 + arsenic + educ
## Model 2: switch ~ dist100 + arsenic + educ + dist100 * educ
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1      3016     3910.4
## 2      3015     3896.2  1   14.283 0.0001573 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
m6 <- glm(switch ~ dist100+arsenic+educ+dist100*educ+arsenic*educ, family=binomial)
anova(m5, m6, test="Chi")
```

```
## Analysis of Deviance Table
##
## Model 1: switch ~ dist100 + arsenic + educ + dist100 * educ
## Model 2: switch ~ dist100 + arsenic + educ + dist100 * educ + arsenic *
##     educ
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      3015     3896.2
## 2      3014     3893.0  1    3.117  0.07748 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
m7 <- glm(switch ~ dist100+arsenic+educ+dist100*educ+assoc, family=binomial)
anova(m5, m7, test="Chi")
```

```
## Analysis of Deviance Table
##
## Model 1: switch ~ dist100 + arsenic + educ + dist100 * educ
## Model 2: switch ~ dist100 + arsenic + educ + dist100 * educ + assoc
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      3015     3896.2
## 2      3014     3893.1  1   3.0266  0.08191 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(m5)
```

```
##
## Call:
## glm(formula = switch ~ dist100 + arsenic + educ + dist100 * educ,
##     family = binomial)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6603  -1.2085   0.7535   1.0613   1.9448
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.0004956  0.1096145   0.005 0.996392
## dist100      -1.3898523  0.1718840  -8.086 6.17e-16 ***
```

```
## arsenic       0.4805993  0.0419866  11.446  < 2e-16 ***
## educ         -0.0020771  0.0152548  -0.136 0.891693
## dist100:educ  0.0956362  0.0256798   3.724 0.000196 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4118.1  on 3019  degrees of freedom
## Residual deviance: 3896.2  on 3015  degrees of freedom
## AIC: 3906.2
##
## Number of Fisher Scoring iterations: 4
```

```
anova(m5, m2, test="Chi")
```

```
## Analysis of Deviance Table
##
## Model 1: switch ~ dist100 + arsenic + educ + dist100 * educ
## Model 2: switch ~ dist100 + arsenic
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1      3015     3896.2
## 2      3017     3930.7 -2  -34.518 3.195e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

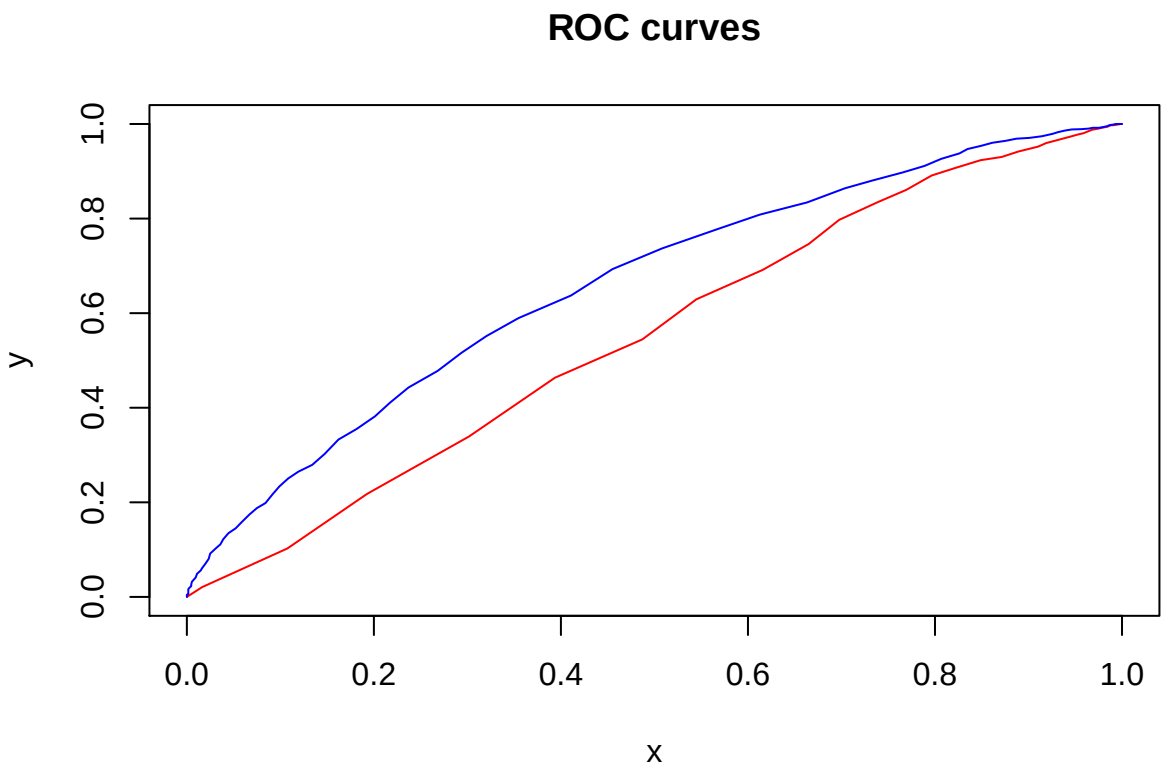
Starting with just the model with dist100 as a predictor, we gradually add more predictors until we find a good model. In the end we are left with the predictors dist100, arsenic, educ, and the interaction between dist100 and education. In the summary output it says that educ is not significant, but the interaction between educ and dist100 is. We generally don't want to keep an interaction term but remove the main effect so we leave educ in the model. This is justified by the fact that testing our final model with the model that just has dist100 and arsenic show that we do indeed want the extra predictors.

The model suggests that households that live far away from clean wells will likely not switch. But higher arsenic levels increase the probability of switching. These are both quite obvious as if you have a contaminated well and there is clean one near you, of course you would switch. When a higher education household lives far away from a well, they are more likely to move anyway than a lower education one would. This could perhaps be due to the fact that those with higher education are wealthier and are more capable of affording to move farther away to a new well.

c)

```
roc.curve <- function(y,pred){
  p <- seq(from=0,to=1,by=0.01)
  out <- matrix(ncol=2,nrow=length(p))
  for(i in 1:length(p)){
    y.hat <- as.numeric(pred>p[i])
    tmp <- cbind(y,y.hat)
    I1 <- as.numeric(y==1)
    I2 <- as.numeric(y.hat==1)
    a <- sum(I1*I2)
    b <- sum(I1*(1-I2))
    c <- sum((1-I1)*I2)
    d <- sum((1-I1)*(1-I2))
```

```
    sens <- a/(a+b)
    spec <- d/(c+d)
    out[i,1] <- 1-spec
    out[i,2] <- sens
  }
  out
}

pred1 <- predict(m1, type="response")
roc1 <- roc.curve(switch, pred1)
pred6 <- predict(m6, type="response")
roc6 <- roc.curve(switch, pred6)

plot(roc1,type="l",xlab="x",ylab="y",main="ROC curves",col="red")
lines(roc6, col="blue")
```



The ROC curve for m1 is almost the same as the y=x line. The ROC curve for m5 is farther away from the y=x line. Since the y=x line reprsents a model that is just guessing, we know that m5 must be better than m1.