ch11 output

```
#install.packages("aod")
library(aod)
library(ggplot2)

ex.data <- read.csv("~/Desktop/jenn/teaching/ADA2/data/admission.csv",header=TRUE)
nrow(ex.data)
```

```
## [1] 400
```

```
#This dataset has a binary response (outcome, dependent) variable called admit, 1 admit, 0 no admission
#There are three predictor variables: gre, gpa and rank.
#variables gre and gpa are continuous. The variable rank takes on the values 1 through 4. Institutions
#with a rank of 1 have the highest prestige,
#while those with a rank of 4 have the lowest.

## view the first few rows of the data
head(ex.data)
```

```
##   admit gre  gpa rank
## 1     0 380 3.61    3
## 2     1 660 3.67    3
## 3     1 800 4.00    1
## 4     1 640 3.19    4
## 5     0 520 2.93    4
## 6     1 760 3.00    2
```

```
#Descriptive Statistics
summary(ex.data)
```

```
##      admit             gre             gpa             rank
##  Min.   :0.0000   Min.   :220.0   Min.   :2.260   Min.   :1.000
##  1st Qu.:0.0000   1st Qu.:520.0   1st Qu.:3.130   1st Qu.:2.000
##  Median :0.0000   Median :580.0   Median :3.395   Median :2.000
##  Mean   :0.3175   Mean   :587.7   Mean   :3.390   Mean   :2.485
##  3rd Qu.:1.0000   3rd Qu.:660.0   3rd Qu.:3.670   3rd Qu.:3.000
##  Max.   :1.0000   Max.   :800.0   Max.   :4.000   Max.   :4.000
```

```
sapply(ex.data, sd) #use sapply to apply the sd function to each variable in the dataset.
```

```
##       admit         gre         gpa        rank
##   0.4660867 115.5165364   0.3805668   0.9444602
```

```
tapply(ex.data$gpa,ex.data$rank,mean)
```

```
##        1        2        3        4
## 3.453115 3.361656 3.432893 3.318358
```

```
tapply(ex.data$gre,ex.data$rank,mean)
```

```
##        1        2        3        4
## 611.8033 596.0265 574.8760 570.1493
```

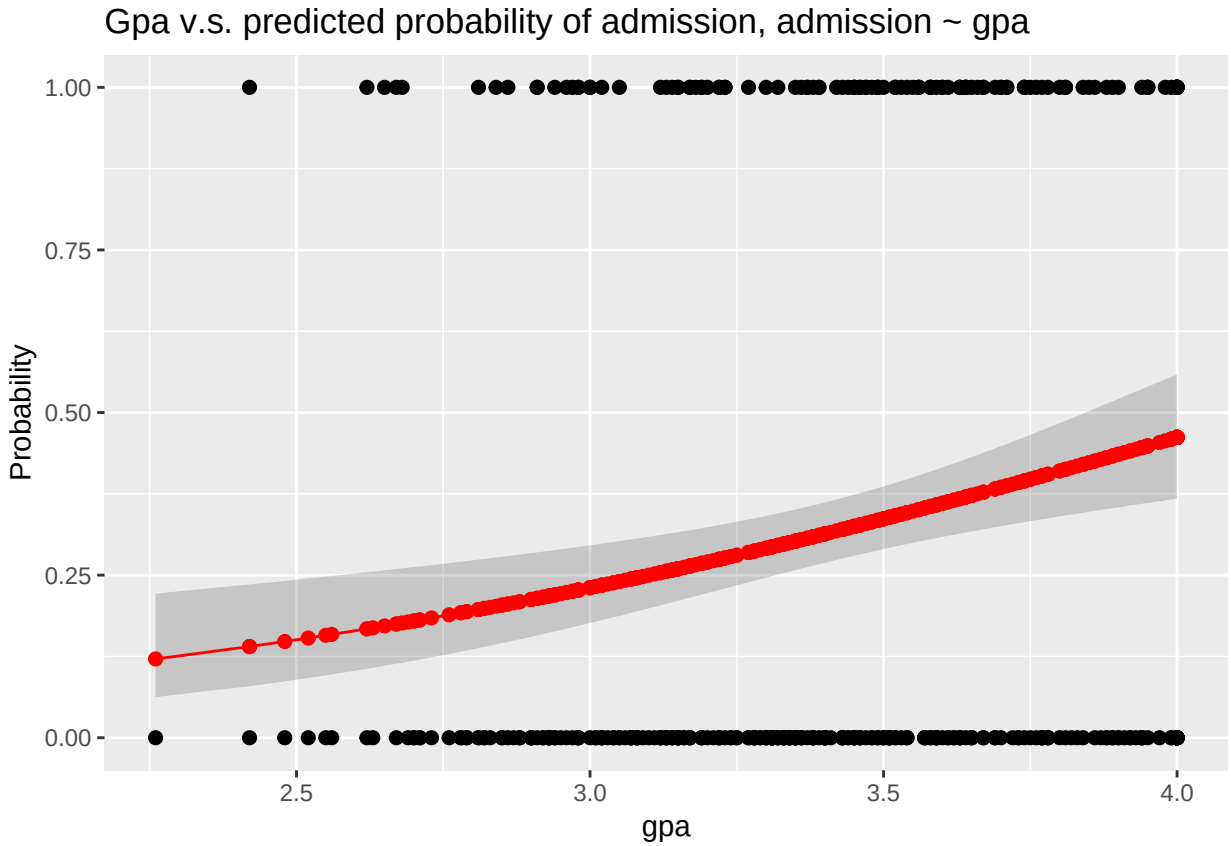
```
## two-way contingency table of categorical outcome and predictors

xtabs(~admit + rank, data = ex.data)
```

```
##      rank
## admit  1  2  3  4
##     0 28 97 93 55
##     1 33 54 28 12
```

```
####Logistic regression modeling
ex.data$rank <- factor(ex.data$rank) #convert rank to a factor to indicate that rank should be treated
#categorical variable.

##fit data with only gpa
myfit_gpa<-glm(admit ~ gpa, data = ex.data, family = "binomial")
summary(myfit_gpa)
```

```
##
## Call:
## glm(formula = admit ~ gpa, family = "binomial", data = ex.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1131  -0.8874  -0.7566   1.3305   1.9824
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -4.3576     1.0353  -4.209 2.57e-05 ***
## gpa           1.0511     0.2989   3.517 0.000437 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 499.98  on 399  degrees of freedom
## Residual deviance: 486.97  on 398  degrees of freedom
## AIC: 490.97
##
## Number of Fisher Scoring iterations: 4
```

```
pred_gpa <- predict(myfit_gpa, type = "link", se.fit = TRUE)

fit.lower_gpa = exp(pred_gpa$fit - 1.96 * pred_gpa$se.fit) / (1 + exp(pred_gpa$fit - 1.96 * pred_gpa$se
fit.upper_gpa = exp(pred_gpa$fit + 1.96 * pred_gpa$se.fit) / (1 + exp(pred_gpa$fit + 1.96 * pred_gpa$se

p <- ggplot(ex.data, aes(x = gpa, y = admit))
# predicted curve and point-wise 95% CI
p <- p + geom_ribbon(aes(x = gpa, ymin = fit.lower_gpa, ymax = fit.upper_gpa), alpha = 0.2)
p <- p + geom_line(aes(x = gpa, y = myfit_gpa$fitted.values),colour="red")
# fitted values
p <- p + geom_point(aes(y = myfit_gpa$fitted.values), colour="red",size=2)
# observed values
p <- p + geom_point(size = 2)
p <- p + ylab("Probability")
p <- p + labs(title = "Gpa v.s. predicted probability of admission, admission ~ gpa")
print(p)
```




```
dev.copy(jpeg,filename="~/Desktop/jenn/teaching/ADA2/lecture notes/plots/ch11plot1.jpg")
```

```
## jpeg
##    3
```

```
dev.off()
```

```
## pdf
##   2
```

```
###fit data with all variables
myfit <- glm(admit ~ gre + gpa + rank, data = ex.data, family = "binomial")
summary(myfit)
```

```
##
## Call:
## glm(formula = admit ~ gre + gpa + rank, family = "binomial",
##     data = ex.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6268  -0.8662  -0.6388   1.1490   2.0790
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.989979   1.139951  -3.500 0.000465 ***
## gre          0.002264   0.001094   2.070 0.038465 *
## gpa          0.804038   0.331819   2.423 0.015388 *
## rank2       -0.675443   0.316490  -2.134 0.032829 *
```

```
## rank3       -1.340204   0.345306  -3.881 0.000104 ***
## rank4       -1.551464   0.417832  -3.713 0.000205 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 499.98  on 399  degrees of freedom
## Residual deviance: 458.52  on 394  degrees of freedom
## AIC: 470.52
##
## Number of Fisher Scoring iterations: 4
```

```
anova(myfit)
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: admit
##
## Terms added sequentially (first to last)
##
##
##      Df Deviance Resid. Df Resid. Dev
## NULL                   399     499.98
## gre   1  13.9204       398     486.06
## gpa   1   5.7122       397     480.34
## rank  3  21.8265       394     458.52
```

```
confint(myfit) ## CIs using profiled log-likelihood
```

```
## Waiting for profiling to be done...
```

```
##                     2.5 %       97.5 %
## (Intercept) -6.2716202334 -1.792547080
## gre          0.0001375921  0.004435874
## gpa          0.1602959439  1.464142727
## rank2       -1.3008888002 -0.056745722
## rank3       -2.0276713127 -0.670372346
## rank4       -2.4000265384 -0.753542605
```

```
confint.default(myfit)   ## CIs using standard errors
```

```
##                     2.5 %       97.5 %
## (Intercept) -6.2242418514 -1.755716295
## gre          0.0001202298  0.004408622
## gpa          0.1536836760  1.454391423
## rank2       -1.2957512650 -0.055134591
## rank3       -2.0169920597 -0.663415773
## rank4       -2.3703986294 -0.732528724
```

```
myfit$deviance   #G^2
```

```
## [1] 458.5175
```

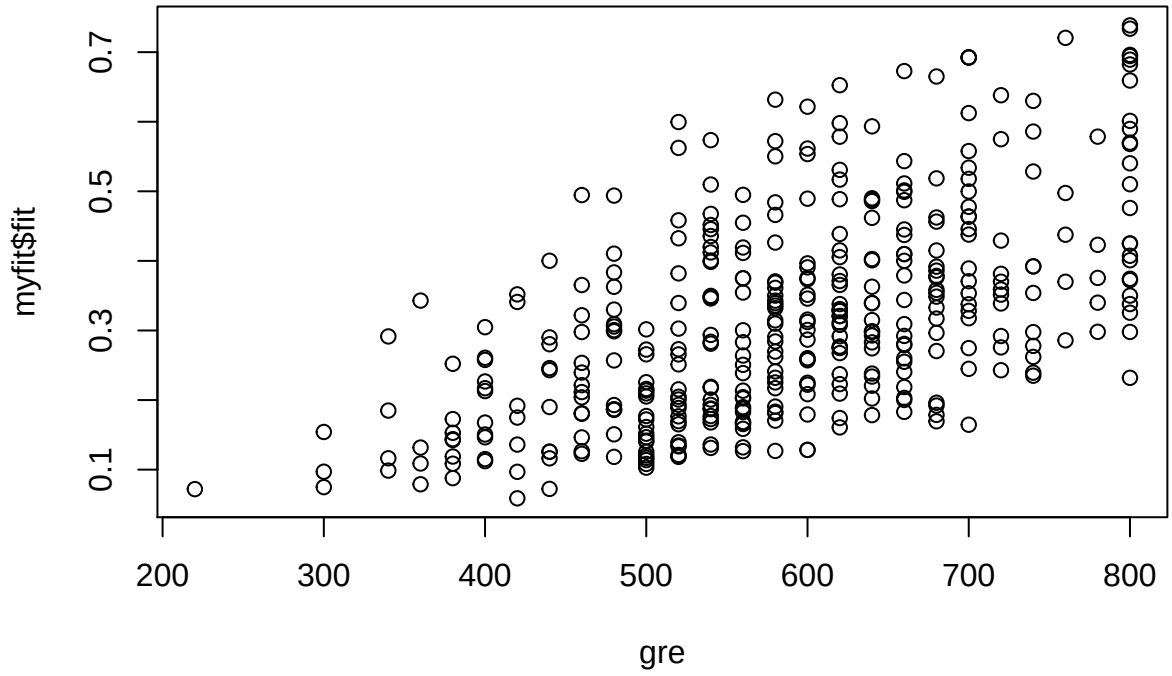
```
logLik(myfit) #log likelihood ratio
```

```
## 'log Lik.' -229.2587 (df=6)
```

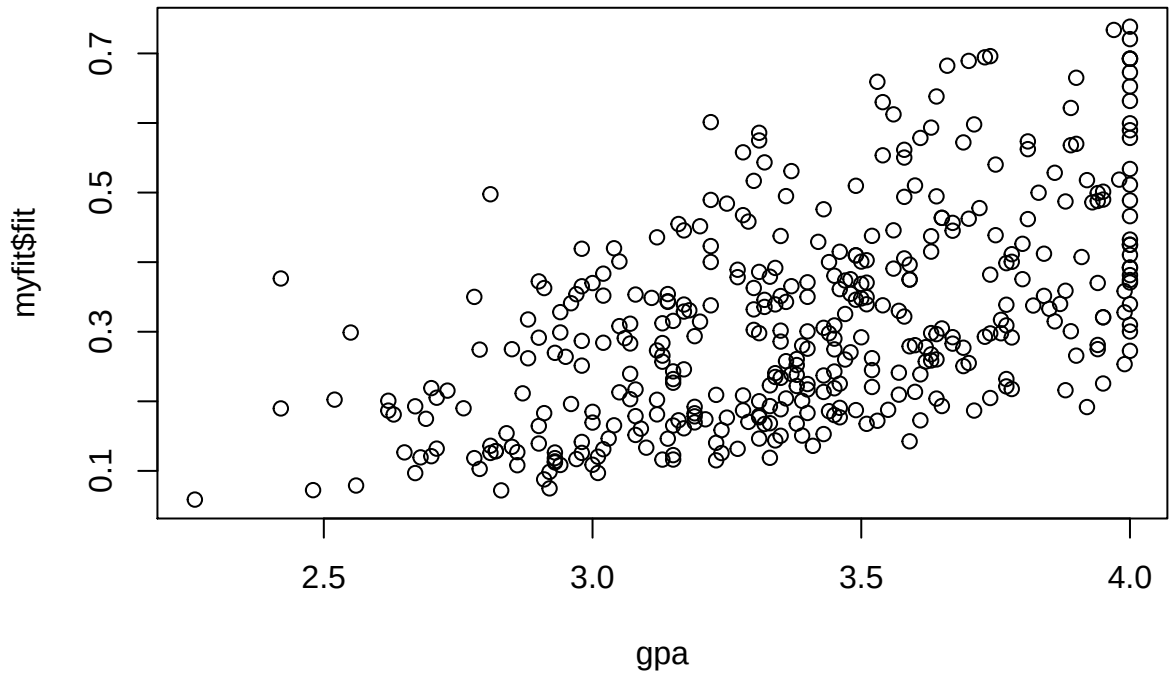
```
# -2*(-229.2587) =458.52

###prediction or fitted probability
#predict the probability of admission for the first observation
myfit$fitted.values[1]
```

```
##         1
## 0.1726265
```

```
fit<-myfit$fit     #fitted probabilities
plot(myfit$fit ~ gre, data=ex.data)
```




```
plot(myfit$fit ~ gpa, data=ex.data)
```




```
#default se=TRUE will give confidence bands

pred <- predict(myfit, type = "link", se.fit = TRUE)
ex.data$fit    <- pred$fit
ex.data$se.fit <- pred$se.fit
fitted    = exp(fit) / (1 + exp(fit))
fit.lower = exp(ex.data$fit - 1.96 * ex.data$se.fit) / (1 + exp(ex.data$fit - 1.96 * ex.data$se.fit))
fit.upper = exp(ex.data$fit + 1.96 * ex.data$se.fit) / (1 + exp(ex.data$fit + 1.96 * ex.data$se.fit))

p <- ggplot(ex.data, aes(x = gpa, y = admit,colour = rank, fill = rank))
# predicted curve and point-wise 95% CI
p <- p + geom_ribbon(aes(x = gpa, ymin = fit.lower, ymax = fit.upper), alpha = 0.2)
p <- p + geom_line(aes(x = gpa, y = myfit$fitted.values))
# fitted values
p <- p + geom_point(aes(y = myfit$fitted.values), size=2)
# observed values
p <- p + geom_point(size = 2)
p <- p + ylab("Probability")
p <- p + labs(title = "Gpa v.s. predicted probability of admission, admission ~ gpa +gre +rank")
print(p)
```


```
dev.copy(jpeg,filename="~/Desktop/jenn/teaching/ADA2/lecture notes/plots/ch11plot2.jpg")
```

```
## jpeg
##    3
```

```
dev.off()
```

```
## pdf
##   2
```

```
mgre<-tapply(ex.data$gre, ex.data$rank, mean) # mean of gre by rank
mgpa<-tapply(ex.data$gpa, ex.data$rank, mean) # mean of gpa by rank
newdata1 <- with(ex.data, data.frame(gre = mgre, gpa = mgpa, rank = factor(1:4)))
newdata1
```

```
##        gre      gpa rank
## 1 611.8033 3.453115    1
## 2 596.0265 3.361656    2
## 3 574.8760 3.432893    3
## 4 570.1493 3.318358    4
```

```
newdata1$rankP <- predict(myfit, newdata = newdata1, type = "response")
newdata1
```

```
##        gre      gpa rank     rankP
## 1 611.8033 3.453115    1 0.5428541
## 2 596.0265 3.361656    2 0.3514055
## 3 574.8760 3.432893    3 0.2195579
## 4 570.1493 3.318358    4 0.1704703
```

```
fitted1<-predict(myfit) #fitted 1 is the fitted value of log odds
phat<-exp(fitted1)/(1+exp(fitted1))
phat[1:10]
```

```
##         1         2         3         4         5         6         7
## 0.1726265 0.2921750 0.7384082 0.1783846 0.1183539 0.3699699 0.4192462
##         8         9        10
## 0.2170033 0.2007352 0.5178682
```

```
fitted2<-predict(myfit,type="response")  #fitted 2 calculate the phat directly
fitted2[1:10]
```

```
##         1         2         3         4         5         6         7
## 0.1726265 0.2921750 0.7384082 0.1783846 0.1183539 0.3699699 0.4192462
##         8         9        10
## 0.2170033 0.2007352 0.5178682
```

```
## odds ratios
exp(coef(myfit))
```

```
## (Intercept)         gre         gpa       rank2       rank3       rank4
##   0.0185001   1.0022670   2.2345448   0.5089310   0.2617923   0.2119375
```

```
## odds ratios and 95% CI
exp(cbind(OR = coef(myfit), confint(myfit)))
```

```
## Waiting for profiling to be done...
```

```
##                    OR       2.5 %    97.5 %
## (Intercept) 0.0185001 0.001889165 0.1665354
## gre         1.0022670 1.000137602 1.0044457
## gpa         2.2345448 1.173858216 4.3238349
## rank2       0.5089310 0.272289674 0.9448343
## rank3       0.2617923 0.131641717 0.5115181
## rank4       0.2119375 0.090715546 0.4706961
```

```
#test for an overall effect of rank using the wald.test function
wald.test(b = coef(myfit), Sigma = vcov(myfit), Terms = 4:6)
```

```
## Wald test:
## ----------
##
## Chi-squared test:
## X2 = 20.9, df = 3, P(> X2) = 0.00011
```

```
#test that the coefficient for rank=2 is equal to the coefficient for rank=3
l <- cbind(0, 0, 0, 1, -1, 0)
wald.test(b = coef(myfit), Sigma = vcov(myfit), L = l)
```

```
## Wald test:
## ----------
##
## Chi-squared test:
## X2 = 5.5, df = 1, P(> X2) = 0.019
```

```
##comparing models

myfit0<-glm(admit ~ 1, data = ex.data, family = "binomial")
summary(myfit0)
```

```
##
## Call:
## glm(formula = admit ~ 1, family = "binomial", data = ex.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.8741  -0.8741  -0.8741   1.5148   1.5148
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.7653     0.1074  -7.125 1.04e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 499.98  on 399  degrees of freedom
## Residual deviance: 499.98  on 399  degrees of freedom
## AIC: 501.98
##
## Number of Fisher Scoring iterations: 4
```

```
myfit2<-glm(admit ~ gre + gpa, data = ex.data, family = "binomial")
summary(myfit2)
```

```
##
## Call:
## glm(formula = admit ~ gre + gpa, family = "binomial", data = ex.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2730  -0.8988  -0.7206   1.3013   2.0620
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.949378   1.075093  -4.604 4.15e-06 ***
## gre          0.002691   0.001057   2.544   0.0109 *
## gpa          0.754687   0.319586   2.361   0.0182 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 499.98  on 399  degrees of freedom
## Residual deviance: 480.34  on 397  degrees of freedom
## AIC: 486.34
##
## Number of Fisher Scoring iterations: 4
```

```
myfit3<-glm(admit ~ gpa+rank, data = ex.data, family = "binomial")
summary(myfit3)
```

```
##
## Call:
```

```
## glm(formula = admit ~ gpa + rank, family = "binomial", data = ex.data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5055  -0.8663  -0.6590   1.1505   2.0913
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -3.4636     1.1003  -3.148 0.001645 **
## gpa           1.0521     0.3102   3.392 0.000694 ***
## rank2        -0.6810     0.3141  -2.168 0.030181 *
## rank3        -1.3919     0.3419  -4.071 4.68e-05 ***
## rank4        -1.5943     0.4152  -3.840 0.000123 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 499.98  on 399  degrees of freedom
## Residual deviance: 462.88  on 395  degrees of freedom
## AIC: 472.88
##
## Number of Fisher Scoring iterations: 4
```

```
anova(myfit0,myfit,test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: admit ~ 1
## Model 2: admit ~ gre + gpa + rank
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1       399     499.98
## 2       394     458.52  5   41.459 7.578e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
qchisq(0.95,5)
```

```
## [1] 11.0705
```

```
pchisq(41.459,5,lower.tail = FALSE)
```

```
## [1] 7.578283e-08
```

```
anova(myfit, myfit2)
```

```
## Analysis of Deviance Table
##
## Model 1: admit ~ gre + gpa + rank
## Model 2: admit ~ gre + gpa
##   Resid. Df Resid. Dev Df Deviance
## 1       394     458.52
## 2       397     480.34 -3  -21.826
```

```
qchisq(0.95,3)
```

```
## [1] 7.814728
```

```r
pchisq(21.826,3,lower.tail = FALSE)
```

```
## [1] 7.090117e-05
```

```r
anova(myfit3,myfit)
```

```
## Analysis of Deviance Table
##
## Model 1: admit ~ gpa + rank
## Model 2: admit ~ gre + gpa + rank
##   Resid. Df Resid. Dev Df Deviance
## 1       395     462.88
## 2       394     458.52  1   4.3578
```

```r
qchisq(0.95,1)
```

```
## [1] 3.841459
```

```r
pchisq(4.3578,1,lower.tail = FALSE)
```

```
## [1] 0.03683985
```

```r
#model selection
upper<-formula(~gre+gpa+rank,data=ex.data)
model.aic = step(myfit0, scope=list(lower= ~., upper= upper))
```

```
## Start:  AIC=501.98
## admit ~ 1
##
##        Df Deviance    AIC
## + rank  3   474.97 482.97
## + gre   1   486.06 490.06
## + gpa   1   486.97 490.97
## <none>      499.98 501.98
##
## Step:  AIC=482.97
## admit ~ rank
##
##        Df Deviance    AIC
## + gpa   1   462.88 472.88
## + gre   1   464.53 474.53
## <none>      474.97 482.97
## - rank  3   499.98 501.98
##
## Step:  AIC=472.88
## admit ~ rank + gpa
##
##        Df Deviance    AIC
## + gre   1   458.52 470.52
## <none>      462.88 472.88
## - gpa   1   474.97 482.97
## - rank  3   486.97 490.97
##
## Step:  AIC=470.52
## admit ~ rank + gpa + gre
##
##        Df Deviance    AIC
```

```
## <none>      458.52 470.52
## - gre   1   462.88 472.88
## - gpa   1   464.53 474.53
## - rank  3   480.34 486.34
```

```
##diagnostics
residuals(myfit)[1:10] # deviance residuals
```

```
##          1          2          3          4          5          6
## -0.6156283  1.5686953  0.7787919  1.8567786 -0.5019254  1.4102011
##          7          8          9         10
##  1.3185576 -0.6994666  1.7920763 -1.2079220
```

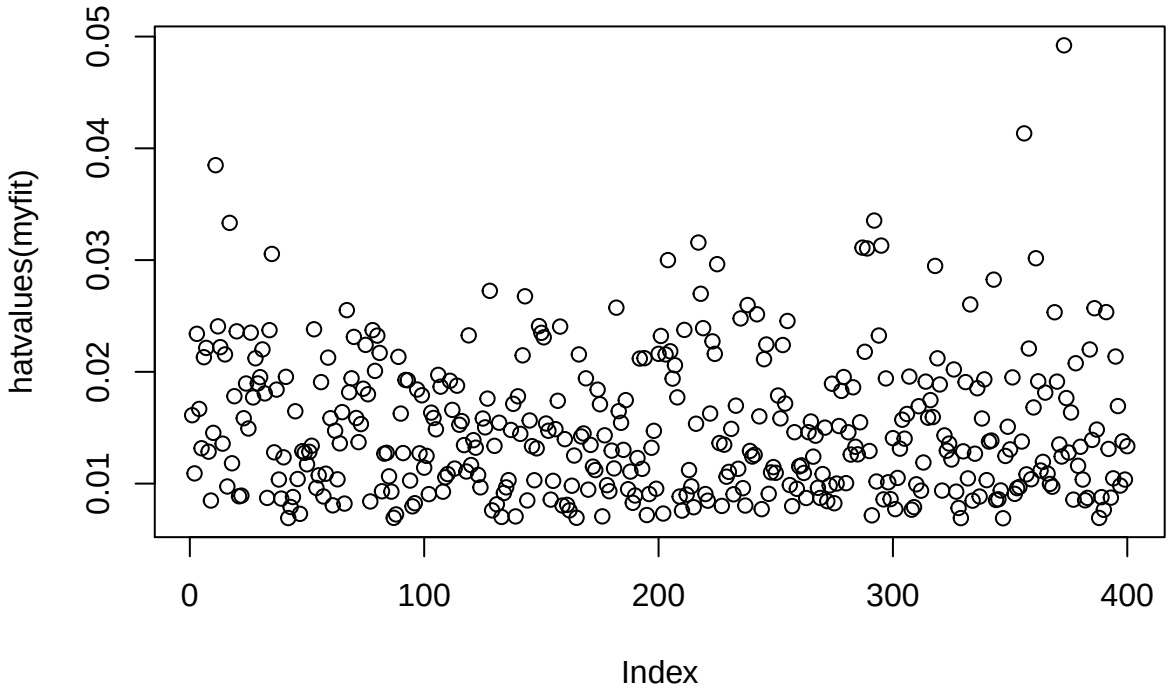
```
residuals(myfit, "pearson")[1:10] # pearson residuals
```

```
##          1          2          3          4          5          6
## -0.4567757  1.5564726  0.5952011  2.1461279 -0.3663905  1.3049606
##          7          8          9         10
##  1.1769594 -0.5264452  1.9954167 -1.0363984
```

```
smyfit<-summary(myfit)
infv <- c(ex.data$admit,myfit$fit,hatvalues(myfit),rstandard(myfit),cooks.distance(myfit))
inf<-matrix(infv,I(smyfit$df[1]+smyfit$df[2]),5,dimnames = list(NULL,
                                                                c("y", "yhat", "lev","r","C")))
inf[1:10,]
```

```
##       y      yhat         lev          r            C
##  [1,] 0 0.1726265 0.016121436 -0.6206515 0.0005791292
##  [2,] 1 0.2921750 0.010913543  1.5773260 0.0045043171
##  [3,] 1 0.7384082 0.023401648  0.7880675 0.0014487409
##  [4,] 1 0.1783846 0.016681487  1.8724620 0.0132436067
##  [5,] 0 0.1183539 0.013165325 -0.5052624 0.0003024683
##  [6,] 1 0.3699699 0.021309398  1.4254708 0.0063142813
##  [7,] 1 0.4192462 0.022142758  1.3334028 0.0053462903
##  [8,] 0 0.2170033 0.012860089 -0.7040081 0.0006095954
##  [9,] 1 0.2007352 0.008484274  1.7997272 0.0057270558
## [10,] 0 0.5178682 0.014541377 -1.2168014 0.0026805935
```

```
#leverages
plot(hatvalues(myfit))
```




```
highleverage <- which(hatvalues(myfit) > .045)
highleverage
```

```
## 373
## 373
```

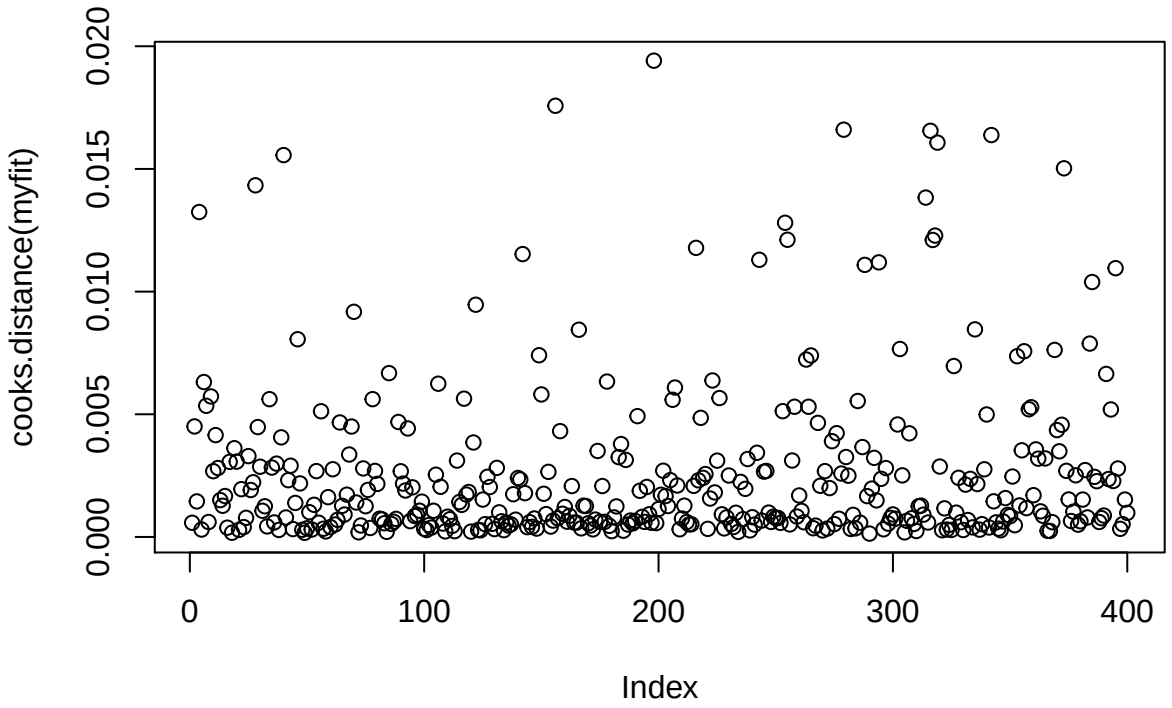
```
hatvalues(myfit)[highleverage]
```

```
##        373
## 0.04921401
```

```
ex.data[373,]
```

```
##     admit gre  gpa rank        fit    se.fit
## 373     1 680 2.42    1 -0.5043987 0.4578697
```

```
#cooks distance
plot(cooks.distance(myfit))
```




```
highcook <- which((cooks.distance(myfit)) > .05)
cooks.distance(myfit)[highcook]
```

```
## named numeric(0)
```

```
##prediction
#use 0.5 as a cutoff to calculate classification error
table(ex.data$admit,fitted(myfit)>.5)
```

```
##
##     FALSE TRUE
##   0   254   19
##   1    97   30
```

```
t1<-table(ex.data$admit,fitted(myfit)>.5)
(t1[1,2]+t1[2,1])/sum(t1)
```

```
## [1] 0.29
```

```
#finding a classification rule that minimizes
#misclassification
for(p in seq(.15,.7,.05))
{t1<-table(ex.data$admit,fitted(myfit)>p)
cat(p,(t1[1,2]+t1[2,1])/sum(t1),"\n")
}
```

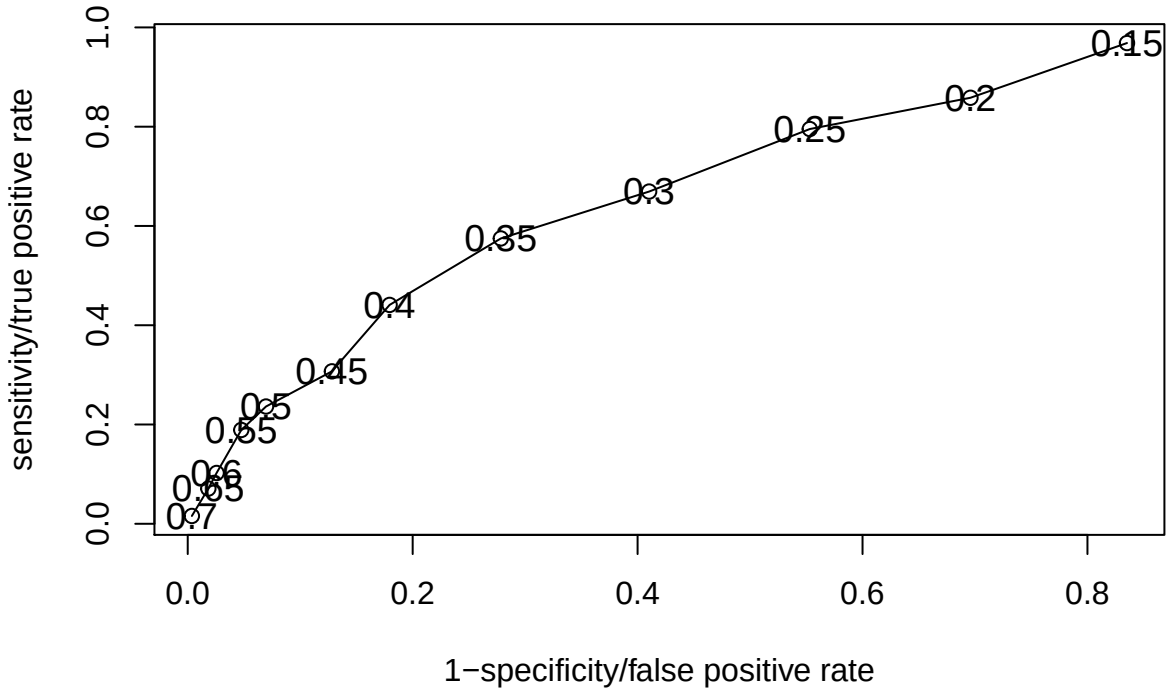
```
## 0.15 0.58
## 0.2 0.52
## 0.25 0.4425
## 0.3 0.385
## 0.35 0.325
## 0.4 0.3
## 0.45 0.3075
## 0.5 0.29
```

```
## 0.55 0.29
## 0.6 0.3025
## 0.65 0.3075
## 0.7 0.315
```

```r
max(fitted(myfit))
```

```
## [1] 0.7384082
```

```r
#Roc curve
p1<-matrix(0,nrow=12,ncol=3)
i=1
for(p in seq(0.15,.7,.05)){
  t1<-table(ex.data$admit,fitted(myfit)>p)
  p1[i,]=c(p,1-(t1[1,1])/sum(t1[1,]),(t1[2,2])/sum(t1[2,]))
  i=i+1
}
plot(p1[,2],p1[,3],type = "o",xlab="1-specificity/false positive rate",
     ylab="sensitivity/true positive rate")
text(p1[,2],p1[,3],p1[,1],cex=1.2)
```




```r
#p1[,2] false positive rate (type I error)
#p1[,3] true postive rate (power)
dp1<-data.frame(p1)
names(dp1)<-c("cutt off prob","type I error","power")
print(dp1)
```

```
##    cutt off prob type I error      power
## 1           0.15  0.835164835 0.96850394
## 2           0.20  0.695970696 0.85826772
## 3           0.25  0.553113553 0.79527559
## 4           0.30  0.410256410 0.66929134
## 5           0.35  0.278388278 0.57480315
## 6           0.40  0.179487179 0.44094488
```

```
## 7          0.45  0.128205128 0.30708661
## 8          0.50  0.069597070 0.23622047
## 9          0.55  0.047619048 0.18897638
## 10         0.60  0.025641026 0.10236220
## 11         0.65  0.018315018 0.07086614
## 12         0.70  0.003663004 0.01574803
```