Logistic Regression - Programming Task

Dr. Phil

This data shows success at completing a programming task as a function of the months of training each person received.

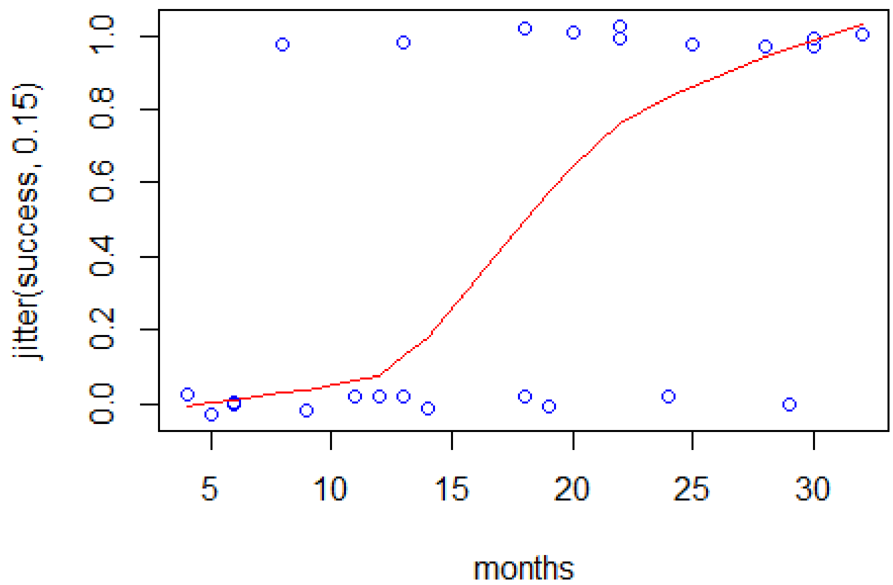
Y = Programming task completed successfully

X = Months of training

```
pgmtask <- read.table(file="C:/Users/iverph01/Documents/Fall 2015/Stat 327/KutnerData/Chapter 14 Data Sets/CH14TA01.txt",header=FALSE, col.names = c('months', 'success', 'fitprob'))

attach(pgmtask)
```

success vs. months with a Lowess fit - showing an approximate S-shaped curve

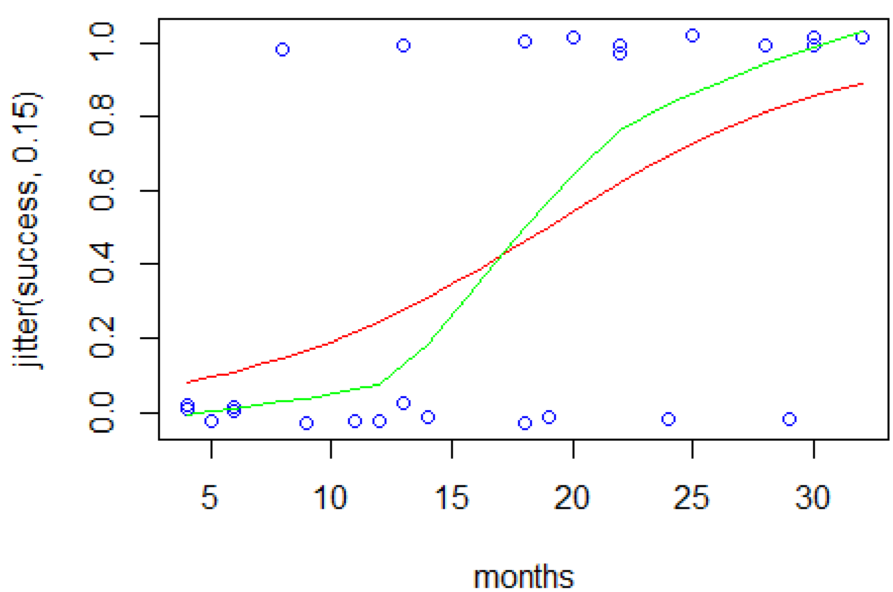
```
summary(months)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    4.00    9.00   18.00   16.88   24.00   32.00

plot(months, jitter(success, .15),col="blue" )
lines(lowess(success~months),col="red")
```



Fit the simple logistic regression model. The function is now ‘glm’ and must include the option, ‘family=binomial’.

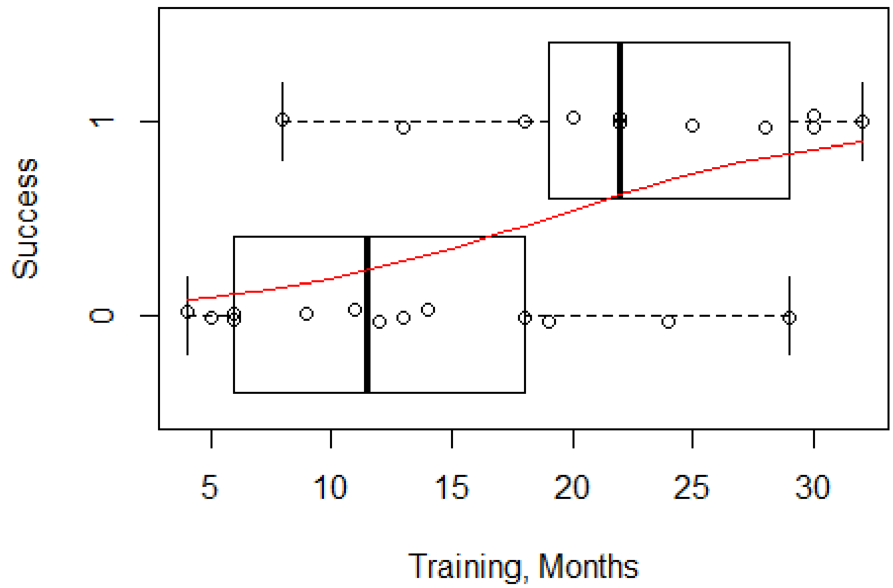
```
# Fitting the regression model:
success.logit = glm(success ~ months, family=binomial)
summary(success.logit)

##
## Call:
## glm(formula = success ~ months, family = binomial)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8992  -0.7509  -0.4140   0.7992   1.9624
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.05970    1.25935  -2.430   0.0151 *
## months       0.16149    0.06498   2.485   0.0129 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 34.296  on 24  degrees of freedom
## Residual deviance: 25.425  on 23  degrees of freedom
## AIC: 29.425
```

```
##
## Number of Fisher Scoring iterations: 4

# plot the model with the data.

plot(months, jitter(success, .15), col="blue")

# Plot the fitted values
# X is a vector of 1's and a grid of month values, like an X-matrix
# beta.logit is the vector of parameter estimates
X <- cbind(1, seq(from=min(months), to=max(months), by=1))
beta.logit = coefficients(success.logit)

# Vector multiplication with the symbol, %*%
# Xb is a vector of fitted values on the logit scale
Xb <- X %*% beta.logit

# exp(Xb) are odds, which are converted to probabilities
prob <- exp(Xb)/(1+exp(Xb))

lines(seq(from=min(months), to=max(months), by=1), prob, col="red")

# Add the Lowess fit for comparison
lines(lowess(success~months),col="green")
```



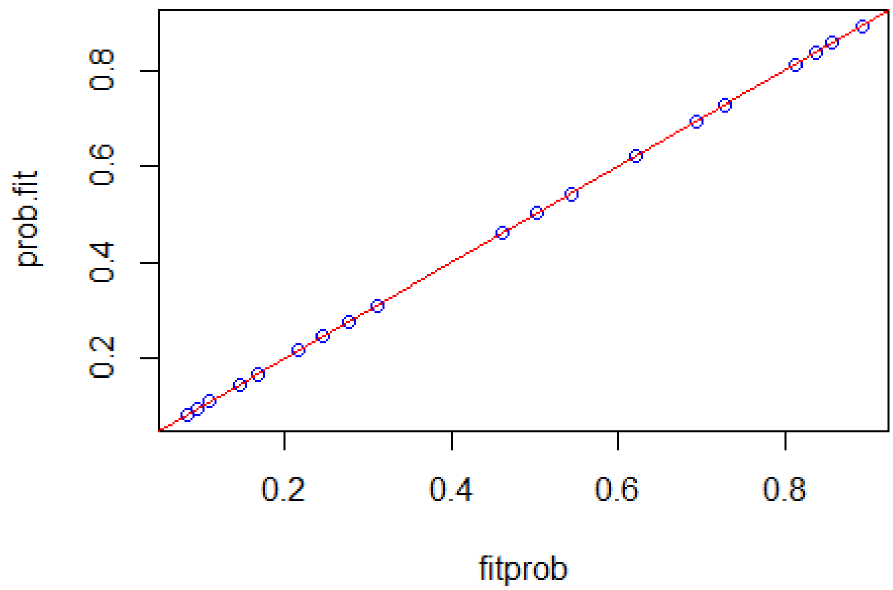
horizontal box plots are another way to plot the data.

```r
boxplot (months ~ success, horizontal=T, ylab='Success', xlab='Training, Months')
# Must add 1 to success, because the underlying y-axis values
# are 1 and 2, not 0 and 1.
points (months, jitter (success+1, 0.2))
lines(seq(from=min(months), to=max(months), by=1), prob+1, col="red")
```

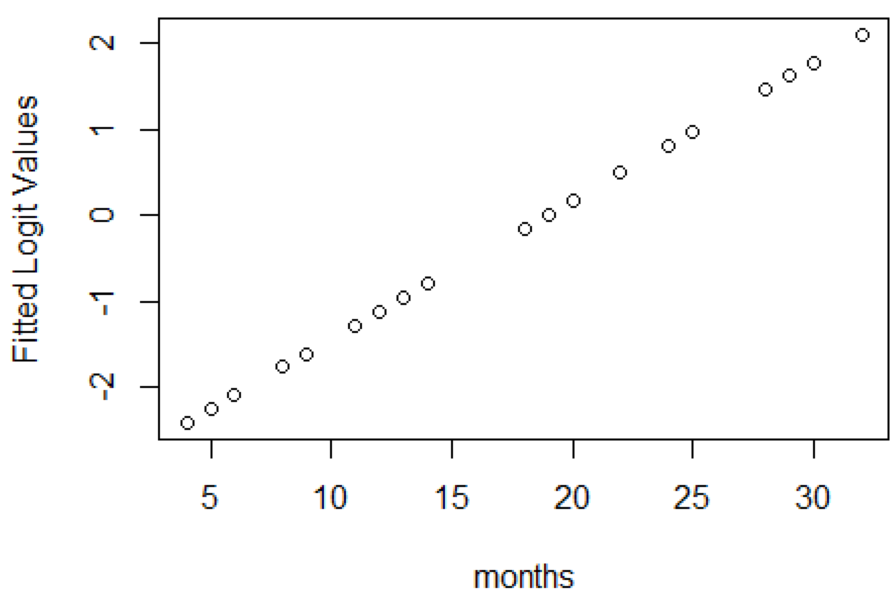


What about the “fitprob” values that came with the data set? They match the fitted probabilities calculated from the data set:

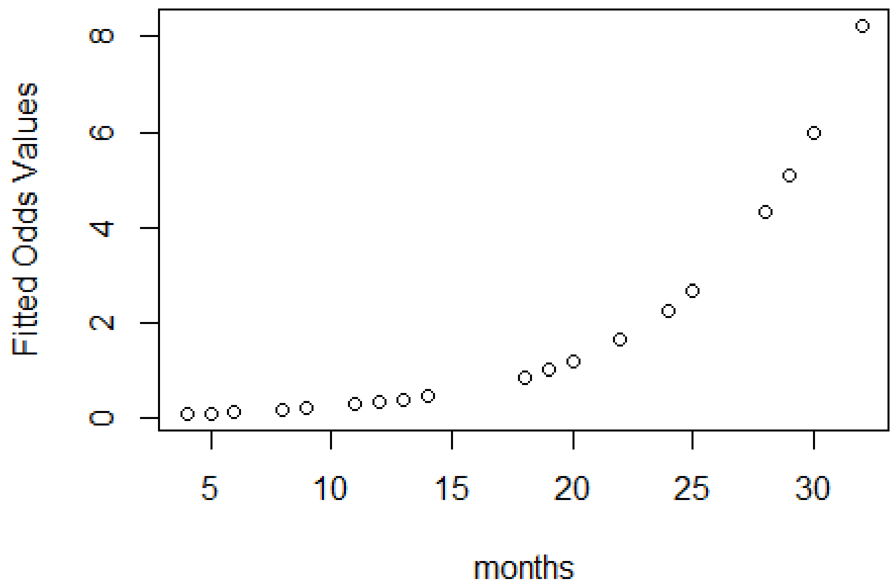
```r
# X.data is the X-matrix
X.data = cbind (1, months)

# Xb.data = the matrix product, X*b, which are the fitted values on the logit scale
Xb.data = X.data %*% beta.logit

# Convert fitted logit values to fitted odds values
odds.fit = exp (Xb.data)

# Convert fitted odds values to fitted probabilities
prob.fit= exp (Xb.data) / (1 + exp(Xb.data))

# Compare fitted probabilities
plot (fitprob, prob.fit, col='blue')
abline (0, 1, col='red')
```



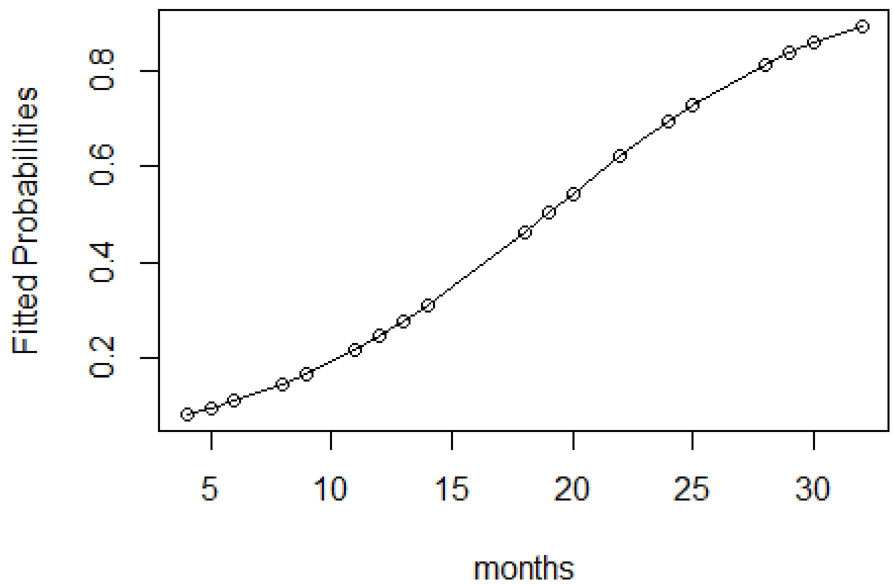

```
# A plot of Xb vs X is a straight line (like Y-hat vs X in linear regression)
plot (months, Xb.data, ylab="Fitted Logit Values")
```



```r
# Odds vs X
plot (months, odds.fit, ylab="Fitted Odds Values")
```




```r
# Probability vs X
plot (months, prob.fit, ylab="Fitted Probabilities")
# lines (months, prob.fit)  # does not work, since data are not sorted by
month
lines (months [order (months)], prob.fit [order (months)])
```



Logistic regression can be used as a classifier method. A cutoff is chosen for the fitted probabilities. Assuming the response variable is coded as 0 or 1, the predicted response category for a particular observation is zero if the fitted probability is below the cutoff; otherwise, it is one.

Let's choose 0.5 as the cutoff. The print the table in order of fitprob.

```
pgmtask$pred.cat = ifelse (prob.fit < 0.5, 0, 1)
pgmtask[order(fitprob),]

##    months success  fitprob pred.cat
## 6       4       0 0.082130        0
## 22      4       0 0.082130        0
## 14      5       0 0.095154        0
## 3       6       0 0.109996        0
## 10      6       0 0.109996        0
## 25      8       1 0.145815        0
## 17      9       0 0.167100        0
## 12     11       0 0.216980        0
## 8      12       0 0.245666        0
## 16     13       0 0.276802        0
## 20     13       1 0.276802        0
## 1      14       0 0.310262        0
## 5      18       1 0.461837        0
## 7      18       0 0.461837        0
## 21     19       0 0.502134        1
## 15     20       1 0.542404        1
```

```
## 9      22       1 0.620812        1
## 24     22       1 0.620812        1
## 19     24       0 0.693379        1
## 4      25       1 0.726602        1
## 23     28       1 0.811825        1
## 2      29       0 0.835263        1
## 11     30       1 0.856299        1
## 13     30       1 0.856299        1
## 18     32       1 0.891664        1
```

Here's a summary table of observed outcome vs fitted outcome.

```
attach (pgmtask)

## The following objects are masked from pgmtask (pos = 3):
##
##     fitprob, months, success

table1 = table (success, pred.cat)
table1

##        pred.cat
## success  0  1
##       0 11  3
##       1  3  8

sensitivity = table1[2,2]/sum(table1[2,])
sensitivity

## [1] 0.7272727

specificity = table1[1,1]/sum(table1[1,])
specificity

## [1] 0.7857143
```

INFERENCE IN LOGISTIC REGRESSION

Deviance test for lack of fit. Hosmer-Lemeshow test, p. 589-90.

```
#-----------------------Deviance test of lack of fit--------------------

# Note: The lower=F option calculates the upper tail probability, which is
what we want
pchisq(deviance(success.logit), df.residual(success.logit), lower=F)

## [1] 0.3287723

############ Goodness of Fit: ##############
# A function to do the Hosmer-Lemeshow test in R.
# R Function is due to Peter D. M. Macdonald, McMaster University.
```

```r
#
hosmerlem <-
function (y, yhat, g = 10)
{
    cutyhat <- cut(yhat, breaks = quantile(yhat, probs = seq(0,
        1, 1/g)), include.lowest = T)
    obs <- xtabs(cbind(1 - y, y) ~ cutyhat)
    expect <- xtabs(cbind(1 - yhat, yhat) ~ cutyhat)
    chisq <- sum((obs - expect)^2/expect)
    P <- 1 - pchisq(chisq, g - 2)
    c("X^2" = chisq, Df = g - 2, "P(>Chi)" = P)
}

# Doing the Hosmer-Lemeshow test

hosmerlem(success, fitted(success.logit))

##       X^2        Df   P(>Chi)
## 6.5599688 8.0000000 0.5847638
```

Likelihood ratio test - this is like the overall F test in linear regression.

```r
#Getting the LR test statistic and P-value in R (simple logistic regression):

pchisq(success.logit$null.deviance - success.logit$deviance,
       success.logit$df.null - success.logit$df.residual, lower=F)

## [1] 0.002895911

1-pchisq(success.logit$null.deviance - success.logit$deviance,
         success.logit $df.null - success.logit$df.residual)

## [1] 0.002895911

# Another way to get the LR test p-value

anova(success.logit)

## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: success
##
## Terms added sequentially (first to last)
##
##
##        Df Deviance Resid. Df Resid. Dev
## NULL                      24     34.296
## months  1   8.8719        23     25.425
```

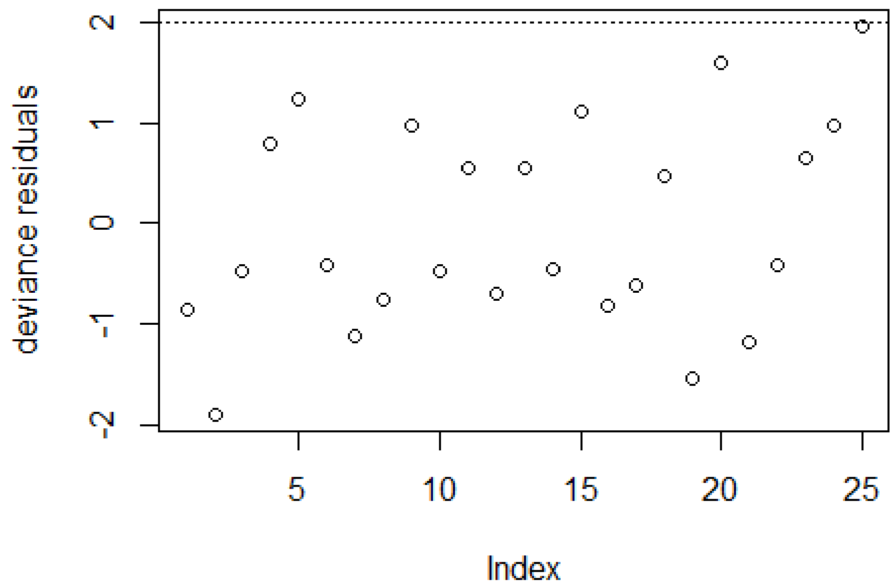
```
LR.test.stat <- sum(anova(success.logit)[2,2])
LR.test.stat

## [1] 8.871916

LR.test.df <- sum(anova(success.logit)[2,1])

LR.test.Pvalue <- 1 - pchisq(LR.test.stat, df=LR.test.df)
LR.test.Pvalue

## [1] 0.002895911
```

Confidence intervals for the estimated parameters - two ways:

```
# An approximate 95% CI for the odds ratio
# associated with each predictor (an indirect way):

alpha <- 0.05
b1 <- summary(success.logit)$coef[2,1]
s.b1 <- summary(success.logit)$coef[2,2]
lower.OR1 <- exp(b1 - qnorm(1-alpha/2)*s.b1)
upper.OR1 <- exp(b1 + qnorm(1-alpha/2)*s.b1)
print(paste(100*(1-alpha), "percent CI for odds ratio 1:", lower.OR1,
upper.OR1))

## [1] "95 percent CI for odds ratio 1: 1.03471646443219 1.3348840012021"

# To get, 99% CIs, just change the specified alpha to 0.01.

# Another way:
confint (success.logit)

## Waiting for profiling to be done...

##                   2.5 %     97.5 %
## (Intercept) -6.03725238 -0.9160349
## months       0.05002505  0.3140397
```

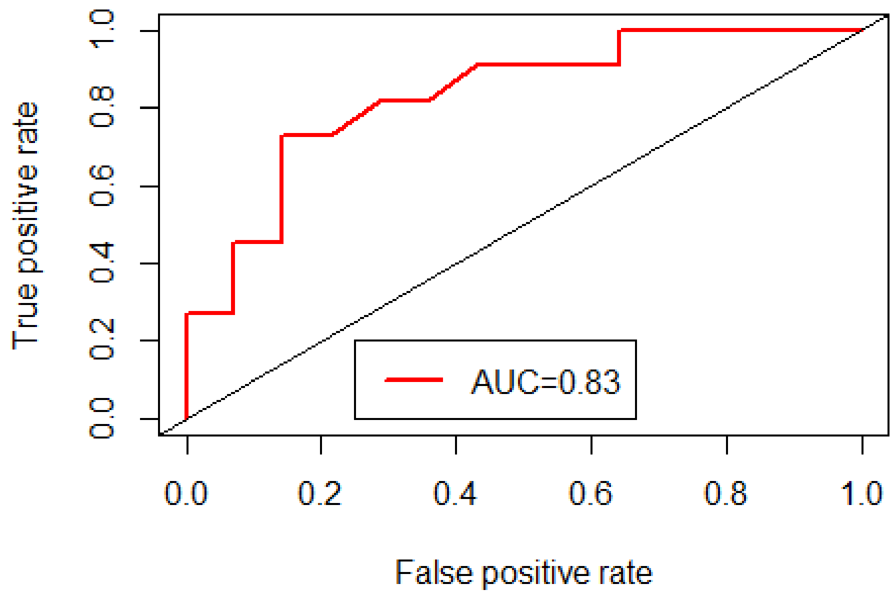
Residual plot

```
# Residual plot

dev<-residuals(success.logit)
plot(dev, ylab="deviance residuals")
abline(h=2, lty=3)
```



ROC (Receiver Operating Characteristic) Curve

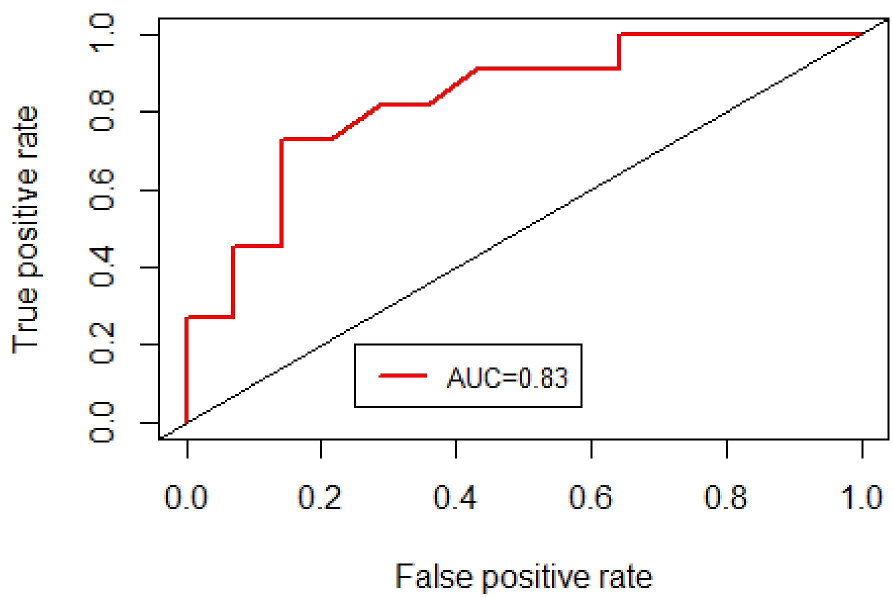
```
# ROC curve - install package ROCR

library(ROCR)

## Loading required package: gplots

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess

pred1 <- prediction(success.logit$fitted.values, success.logit$y)
perf1 <- performance(pred1,"tpr","fpr")
auc1 <- performance(pred1,"auc")@y.values[[1]]
auc1

## [1] 0.8311688

plot(perf1, lwd=2, col=2)
abline(0,1)
legend(0.25, 0.2, c(paste ("AUC=", round(auc1, 2), sep="")),   lwd=2, col=2)
```



Here is a function that creates the ROC plot and returns a table results related to the ROC curve:

```
roc.logistic = function (fit) {
  fitvals = fit$fitted.values
  pred1 <- prediction(fitvals, fit$y)
  perf1 <- performance(pred1,"tpr","fpr")
  auc1 <- performance(pred1,"auc")@y.values[[1]]
  plot(perf1, lwd=2, col=2)
  abline(0,1)
  legend(0.25, 0.2, c(paste ("AUC=", round(auc1, 2), sep="")),
         cex=0.8, lwd=2, col=2)
  roc.table = cbind.data.frame (pred1@tn, pred1@fp, pred1@fn, pred1@tp,
                                pred1@cutoffs, perf1@x.values,
perf1@y.values)
  roc.table$spec = 1 - perf1@x.values[[1]]
  roc.table$ppv = pred1@tp[[1]] / (pred1@tp[[1]] + pred1@fp[[1]])
  roc.table$npv = pred1@tn[[1]] / (pred1@tn[[1]] + pred1@fn[[1]])
  roc.table$pctcorr = (pred1@tn[[1]] + pred1@tp[[1]]) /
                 (pred1@tn[[1]] + pred1@tp[[1]] + pred1@fn[[1]] +
pred1@fp[[1]])
  roc.table$optdist = sqrt ((perf1@x.values[[1]] - 0)^2 +
                            (perf1@y.values[[1]] - 1)^2)
  names (roc.table) = c("TN", "FP", "FN", "TP", "Cutoff", "FPR", "TPR",
"Spec",
                        "PPV", "NPV", "PctCorr", "OptDist")
  return (roc.table)
```

```
}

roc.table = roc.logistic (success.logit)
```




```
# Find the row(s) in the ROC table with the largest percent correctly
classified

roc.table [which.max (roc.table$PctCorr), ]

##    TN FP FN TP    Cutoff       FPR       TPR      Spec PPV NPV PctCorr
## 15 12  2  3  8 0.5424035 0.1428571 0.7272727 0.8571429 0.8 0.8     0.8
##      OptDist
## 15 0.3078771

# Find the row(s) in the ROC table that are closest to the (0, 1) corner.

roc.table [which.min (roc.table$OptDist), ]

##    TN FP FN TP    Cutoff       FPR       TPR      Spec PPV NPV PctCorr
## 15 12  2  3  8 0.5424035 0.1428571 0.7272727 0.8571429 0.8 0.8     0.8
##      OptDist
## 15 0.3078771
```

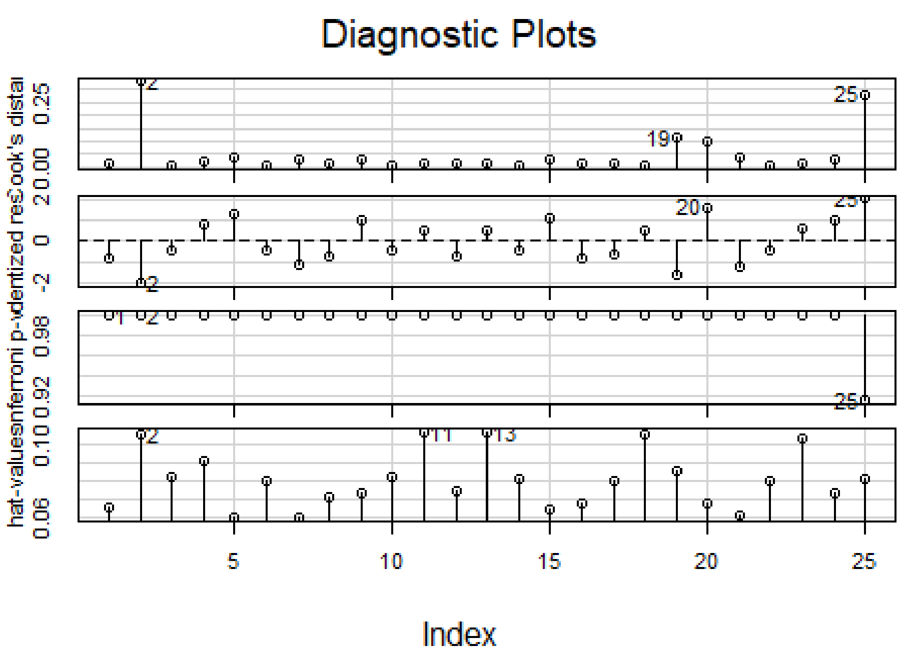
AIC and BIC:

```
########################################################################
```

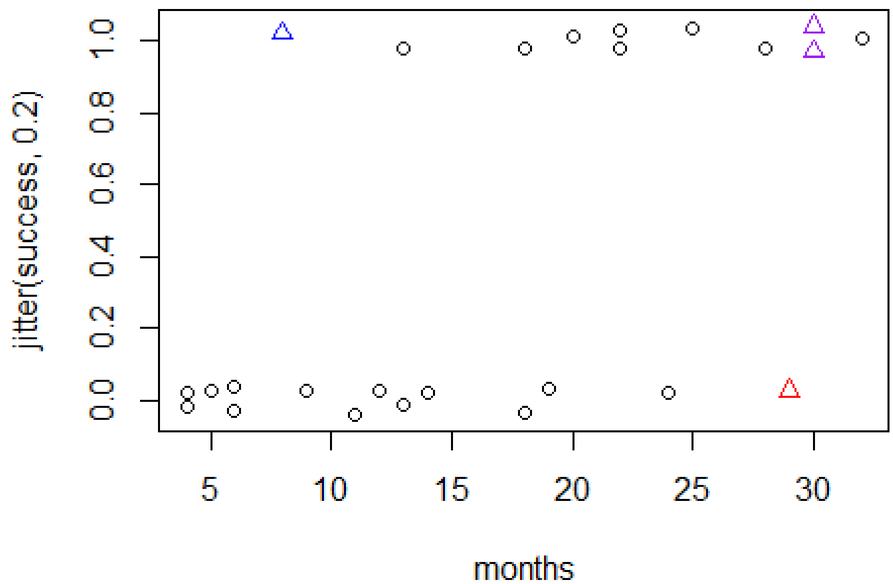
```
# Model Selection Criteria:

# Note for this model, AIC = 108.259 and BIC = 116.014.
# The AIC value is in the R summary output
# BIC can be calculated via:

BIC.value <- AIC(success.logit, k=log(nrow(pgmtask)))
BIC.value

## [1] 31.86233
```

Influence plot

```
#-------------------------------------------------------------------
library(car)

## Loading required package: carData

influenceIndexPlot(success.logit, id=list(n=3))
```




```
# Plot data with rows 2 and 25 in red
my.colors = rep ('black', length (months))
my.colors [2] = 'red'
my.colors [25] = 'blue'
my.colors [c(11,13)] = 'purple'
plotsym = ifelse (my.colors == 'black', 1, 2)
plot (months, jitter (success, 0.2), col=my.colors, pch=plotsym)
```


jitter(success, 0.2)
0.0
0.2
0.4
0.6
0.8
1.0
5
10
15
20
25
30
months