R Notebook

Modeling Binomial Data

1. Describe data
2. Explore data - Exploratory Data Analysis
3. Estimate parameters
4. Make Inference
5. Make Prediction
6. Interpretation

Locate files

- Browse your folders.
- Look for the files.
- Check the path to the folder containing the files

Set the folder

Set our working directory. REMEMBER! your working directory (working folder) is different from my working directory

```
# this is my working directory. You have to specify yours
setwd("E:/Epi_Stat_Matters/LectureNotes2015/binary-logistic/binary-logistic-DrPH-2015/BinaryLogisticDrPH

getwd()
```

```
## [1] "E:/Epi_Stat_Matters/LectureNotes2015/binary-logistic/binary-logistic-DrPH-2015/BinaryLogisticDrP
```

Read data

- Read our data in the working folder
- Then, save as a csv file in our working directory

```
mydata <- read.csv("http://www.ats.ucla.edu/stat/data/binary.csv")
write.csv2(mydata,'logistic.csv')
```

Describe data

```
#observe data
head(mydata,10)
```

Rank is taken as numerical variable which does not make sense. We need to convert it to a categorical (factor) variable

```
summary(mydata)
```

```
##      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
```

```
mydata$rank<-factor(mydata$rank)
summary(mydata$rank)
```

```
##   1   2   3   4
##  61 151 121  67
```

More fancy, we can use *psych* package

```
library(psych)
describe(mydata)
```

```
##       vars   n   mean     sd median trimmed    mad    min max  range  skew
## admit    1 400   0.32   0.47    0.0    0.27   0.00   0.00   1   1.00  0.78
## gre      2 400 587.70 115.52  580.0  589.06 118.61 220.00 800 580.00 -0.14
## gpa      3 400   3.39   0.38    3.4    3.40   0.40   2.26   4   1.74 -0.21
## rank*    4 400   2.48   0.94    2.0    2.48   1.48   1.00   4   3.00  0.10
##       kurtosis   se
## admit    -1.39 0.02
## gre      -0.36 5.78
## gpa      -0.60 0.02
## rank*    -0.91 0.05
```

Explore data

Use plots like * Histogram for numerical variables * and barplot for categorical variables, at least.

Estimate parameters

- we estimate the logit or the log odds.
- We used **summary** to see the results stored as **mylogit**
- We used **coefficients** to examine the regression coefficients

```
mylogit <- glm(admit~gre+gpa+rank,family = 'binomial'(link = logit),data=mydata)
summary(mylogit)
```

```
##
## Call:
## glm(formula = admit ~ gre + gpa + rank, family = binomial(link = logit),
##     data = mydata)
##
## 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
```

```
coefficients(mylogit)
```

```
##  (Intercept)          gre          gpa        rank2        rank3
## -3.989979073  0.002264426  0.804037549 -0.675442928 -1.340203916
##        rank4
## -1.551463677
```

To obtain the odds ratios and their 95% CI, we need to exponentiate using **exp** the regression coefficients or the betas

```
exp(coefficients(mylogit))
```

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

```
exp(confint(mylogit))
```

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

##                   2.5 %    97.5 %
## (Intercept) 0.001889165 0.1665354
## gre         1.000137602 1.0044457
## gpa         1.173858216 4.3238349
## rank2       0.272289674 0.9448343
## rank3       0.131641717 0.5115181
## rank4       0.090715546 0.4706961
```

Make inference

Here, we examine the p-values (hypothesis testing) and the confidence intervals.

- First, using the method of maximum likelihood
- Next, using the SE method (function **confint.default**)

```
confint(mylogit)
```

```
## 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(mylogit)
```

```
##                     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
```

Make prediction

We now can:

1. Predict the log odds for having the outcome
2. Predict the conditional probability for having the outcome

```
pred.logit<-predict(mylogit,type='link')
head(pred.logit)
```

```
##          1          2          3          4          5          6
## -1.5671256 -0.8848442  1.0377118 -1.5273305 -2.0081113 -0.5323458
```

```
-3.99+0.00226*380+0.8041*3.61-1.34
```

```
## [1] -1.568399
```

Notice, that similarity between **predict(x, type='response'** and **fitted** Remember, we can calculate the conditional probability of having the outcome

```
pred.prob<-predict(mylogit,type='response')
head(pred.prob)
```

```
##         1         2         3         4         5         6
## 0.1726265 0.2921750 0.7384082 0.1783846 0.1183539 0.3699699
```

```
head(fitted(mylogit))
```

```
##         1         2         3         4         5         6
## 0.1726265 0.2921750 0.7384082 0.1783846 0.1183539 0.3699699
```

```
exp(-1.567)/(1+exp(-1.567))
```

```
## [1] 0.1726445
```

Compare models

We compare a model with **vs** and without **gre**. This is done using the deviance

```
summary(mylogit)
```

```
##
## Call:
## glm(formula = admit ~ gre + gpa + rank, family = binomial(link = logit),
##     data = mydata)
##
## 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
```

```r
mylogit2 <- glm(admit~gpa+rank,family = 'binomial'(link = logit),data=mydata)
summary(mylogit2)
```

```
##
## Call:
## glm(formula = admit ~ gpa + rank, family = binomial(link = logit),
##     data = mydata)
##
## 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
```

```r
anova(mylogit,mylogit2,test = 'Chisq')
```

```
## Analysis of Deviance Table
##
## Model 1: admit ~ gre + gpa + rank
## Model 2: admit ~ gpa + rank
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       394     458.52
## 2       395     462.88 -1  -4.3578  0.03684 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Linearity in logits

gre is tested for linearity in logit. **gre** is linear but it is rescaled to produce less decimals

The linearity of logits is tested using library **mfp**

```r
library(mfp)
```

```
## Loading required package: survival
```

```r
mylogit3 <- mfp(admit~fp(gre)+gpa+rank,family = 'binomial'(link = logit),data=mydata,verbose=T)
```

```
##
##  Variable    Deviance    Power(s)
## ------------------------------------------------
## Cycle 1
##  rank2
##              463.096
##              458.517     1
##
##
##
##  rank3
##              474.043
##              458.517     1
##
##
##
##  rank4
##              473.551
##              458.517     1
##
##
##
##  gpa
##              464.532
##              458.517     1
##
##
##
##  gre
##              462.875
##              458.517      1
##              458.415      0
##              458.366      -2 -2
##
##
## Tansformation
##       shift scale
## rank2     0     1
## rank3     0     1
## rank4     0     1
## gpa       0     1
## gre       0  1000
##
## Fractional polynomials
##       df.initial select alpha df.final power1 power2
## rank2          1      1  0.05        1      1      .
## rank3          1      1  0.05        1      1      .
## rank4          1      1  0.05        1      1      .
## gpa            1      1  0.05        1      1      .
## gre            4      1  0.05        1      1      .
##
##
## Transformations of covariates:
##              formula
## gre  I((gre/1000)^1)
## gpa              gpa
## rank            rank
##
##
## Deviance table:
##           Resid. Dev
## Null model    499.9765
## Linear model  458.5175
## Final model   458.5175
```

```r
summary(mylogit3)
```

```
##
## Call:
## glm(formula = admit ~ rank + gpa + I((gre/1000)^1), family = binomial(link = logit),
##     data = mydata)
##
## 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)      -3.9900     1.1400  -3.500 0.000465 ***
## rank2            -0.6754     0.3165  -2.134 0.032829 *
## rank3            -1.3402     0.3453  -3.881 0.000104 ***
## rank4            -1.5515     0.4178  -3.713 0.000205 ***
## gpa               0.8040     0.3318   2.423 0.015388 *
## I((gre/1000)^1)   2.2644     1.0940   2.070 0.038465 *
## ---
## 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
```

```r
mylogit3$fptable
```

```
##       df.initial select alpha df.final power1 power2
## rank2          1      1  0.05        1      1      .
## rank3          1      1  0.05        1      1      .
## rank4          1      1  0.05        1      1      .
## gpa            1      1  0.05        1      1      .
## gre            4      1  0.05        1      1      .
```

Diagnostics for a model with a binomial response

To do these diagnostics, you need to load **library('LogisticDx')**.

First, we produce the diagnostic measures for a binary regression model by covariate pattern

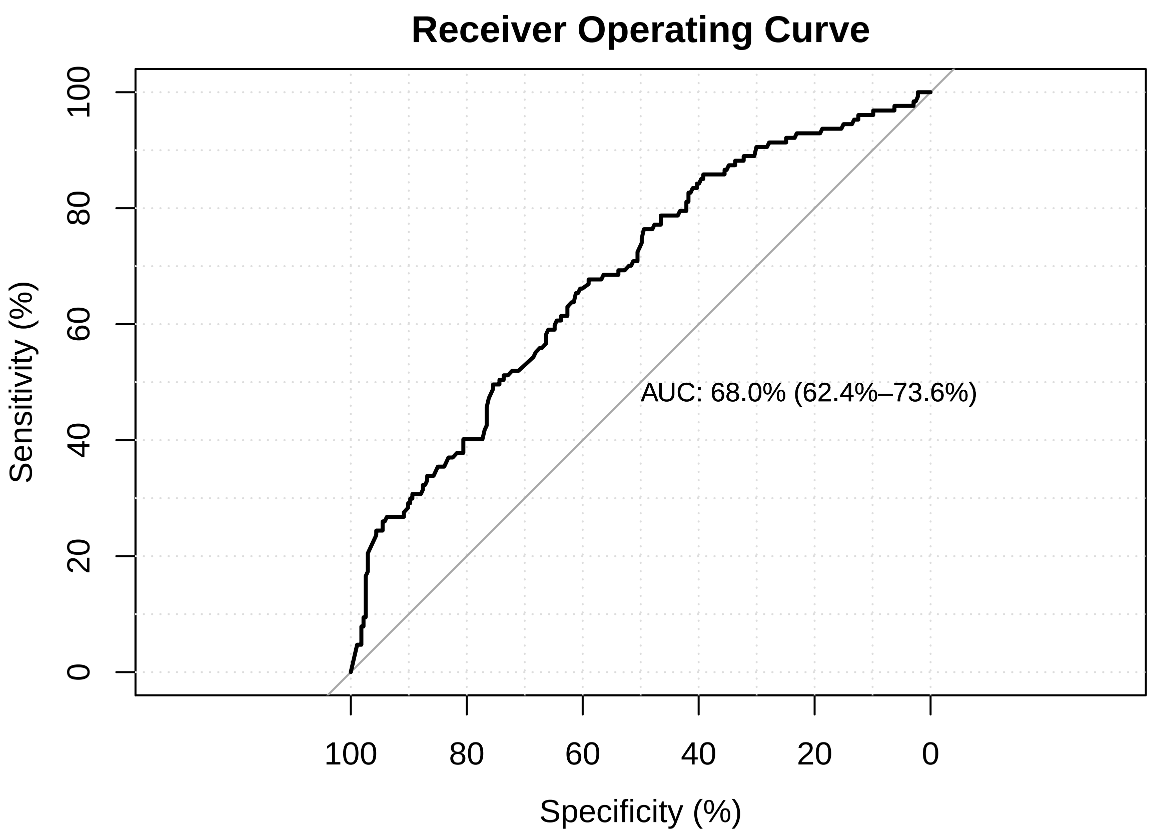
Next, we produce the Goodness-of-fit for binomial regression. Usually, the number of groups (quantiles) equal 10 to perform the Hosmer-Lemeshow test. At the same time, we plot the ROC curve

Similarly, we can check the **auc** value

```r
library('LogisticDx')
dx(mylogit2,byCov=T)
```

```
##      (Intercept)  gpa rank2 rank3 rank4 y         P n      yhat
##   1:           1 3.94     1     0     0 1 0.5001242 2 1.0002483
##   2:           1 3.30     1     0     0 1 0.3378663 3 1.0135990
##   3:           1 3.99     0     1     0 1 0.3412259 3 1.0236777
##   4:           1 3.35     1     0     0 1 0.3497327 3 1.0491980
##   5:           1 3.17     1     0     0 1 0.3079792 3 0.9239375
##  ---
## 256:           1 3.52     0     0     1 2 0.2051122 3 0.6153366
## 257:           1 2.68     0     1     0 1 0.1154721 1 0.1154721
## 258:           1 3.00     0     0     1 1 0.1299149 1 0.1299149
## 259:           1 2.42     0     0     0 1 0.2854368 1 0.2854368
## 260:           1 2.65     0     1     0 1 0.1122874 1 0.1122874
##                 Pr            dr           h           sPr           sdr
##   1: -0.0003511985 -0.0003511985 0.014377039 -0.0003537507 -0.0003537506
##   2: -0.0165997514 -0.0166181074 0.006902268 -0.0166573376 -0.0166757573
##   3: -0.0288329487 -0.0288874508 0.015618469 -0.0290607843 -0.0291157171
##   4: -0.0595625368 -0.0597856465 0.006844794 -0.0597674360 -0.0599913131
##   5:  0.0951239167  0.0944359041 0.007474218  0.0954814103  0.0947908121
##  ---
## 256:  1.9798618855  1.7253451406 0.017098521  1.9970084425  1.7402874601
## 257:  2.7676886108  2.0778480687 0.011720502  2.7840519420  2.0901328741
## 258:  2.5879228606  2.0203343433 0.013448875  2.6055027084  2.0340585431
## 259:  1.5822145401  1.5834991249 0.034641074  1.6103525417  1.6116599714
## 260:  2.8117113691  2.0912646219 0.011927131  2.8286306943  2.1038486967
##            dChisq         dDev        dBhat
##   1: 1.251395e-07 1.251395e-07 1.825379e-09
##   2: 2.774669e-04 2.780809e-04 1.928462e-06
##   3: 8.445292e-04 8.477250e-04 1.339953e-05
##   4: 3.572146e-03 3.598958e-03 2.461912e-05
##   5: 9.116700e-03 8.985298e-03 6.865333e-05
##  ---
## 256: 3.988043e+00 3.028600e+00 6.937586e-02
## 257: 7.750945e+00 4.368655e+00 9.192235e-02
## 258: 6.788644e+00 4.137394e+00 9.254425e-02
## 259: 2.593235e+00 2.597448e+00 9.305602e-02
## 260: 8.001152e+00 4.426179e+00 9.658274e-02
```

```
fit.mylogit2<-gof(mylogit2,g=10,plotROC = T)
```



```
fit.mylogit2
```

```
##       chiSq  df     pVal
## PrI  400.78 395 0.409645
## drI  462.88 395 0.010379 *
## PrG  255.36 255 0.481870
## drG  294.60 255 0.044609 *
## PrCT 255.36 255 0.481870
## drCT 294.60 255 0.044609 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##                          val df     pVal
## HL chiSq         8.190458  8 0.415090
## mHL F            1.484491  9 0.153918
## OsRo Z           0.027903 NA 0.977740
## SstPgeq0.5 Z     0.217564 NA 0.827769
## SstPl0.5 Z       0.422859 NA 0.672398
## SstBoth chiSq    0.226144  2 0.893086
## SllPgeq0.5 chiSq 0.047739  1 0.827045
## SllPl0.5 chiSq   0.176745  1 0.674186
## SllBoth chiSq    0.177398  2 0.915121
```

```
#area under curve
fit.mylogit2$auc
```

```
##          auc lower 95% CI upper 95% CI
##     68.01073     62.39737     73.62409
## attr(,"interpret")
## [1] "auc = 0.5       --> useless"   "0.7 < auc < 0.8 --> good"
## [3] "0.8 < auc < 0.9 --> excellent"
```

```
#chi square test for gof
fit.mylogit2$chiSq
```

```
##    test    chiSq  df       pVal
## 1:  PrI 400.7833 395 0.40964543
## 2:  drI 462.8753 395 0.01037867
## 3:  PrG 255.3601 255 0.48187033
## 4:  drG 294.6050 255 0.04460896
## 5: PrCT 255.3601 255 0.48187033
## 6: drCT 294.6050 255 0.04460896
```

```
#contigency table for HL test
fit.mylogit2$ctHL
```

```
##         P y1     y1hat y0    y0hat  n      Pbar
##  1: 0.149  6  5.037833 34 34.96217 40 0.1259458
##  2: 0.192  5  6.824095 35 33.17591 40 0.1706024
##  3: 0.219  7  8.260683 33 31.73932 40 0.2065171
##  4: 0.258 11  9.327547 28 29.67245 39 0.2391679
##  5: 0.293 12 11.338710 29 29.66129 41 0.2765539
##  6:  0.34 15 12.556219 25 27.44378 40 0.3139055
##  7: 0.376 14 13.796927 25 25.20307 39 0.3537674
##  8: 0.437 14 16.561867 27 24.43813 41 0.4039480
##  9: 0.511 13 18.253216 25 19.74678 38 0.4803478
## 10: 0.678 30 25.042903 12 16.95710 42 0.5962596
```

```
#GOF test
fit.mylogit2$gof
```

```
##          test  stat        val df      pVal
## 1:         HL chiSq 8.19045835  8 0.4150903
## 2:        mHL     F 1.48449073  9 0.1539182
## 3:       OsRo     Z 0.02790263 NA 0.9777398
## 4: SstPgeq0.5     Z 0.21756392 NA 0.8277689
## 5:   SstPl0.5     Z 0.42285933 NA 0.6723979
## 6:    SstBoth chiSq 0.22614408  2 0.8930863
## 7: SllPgeq0.5 chiSq 0.04773921  1 0.8270450
## 8:   SllPl0.5 chiSq 0.17674490  1 0.6741857
## 9:    SllBoth chiSq 0.17739848  2 0.9151208
```

```
# another package for ROC
library(ROCR)
```

```
## Loading required package: gplots
```

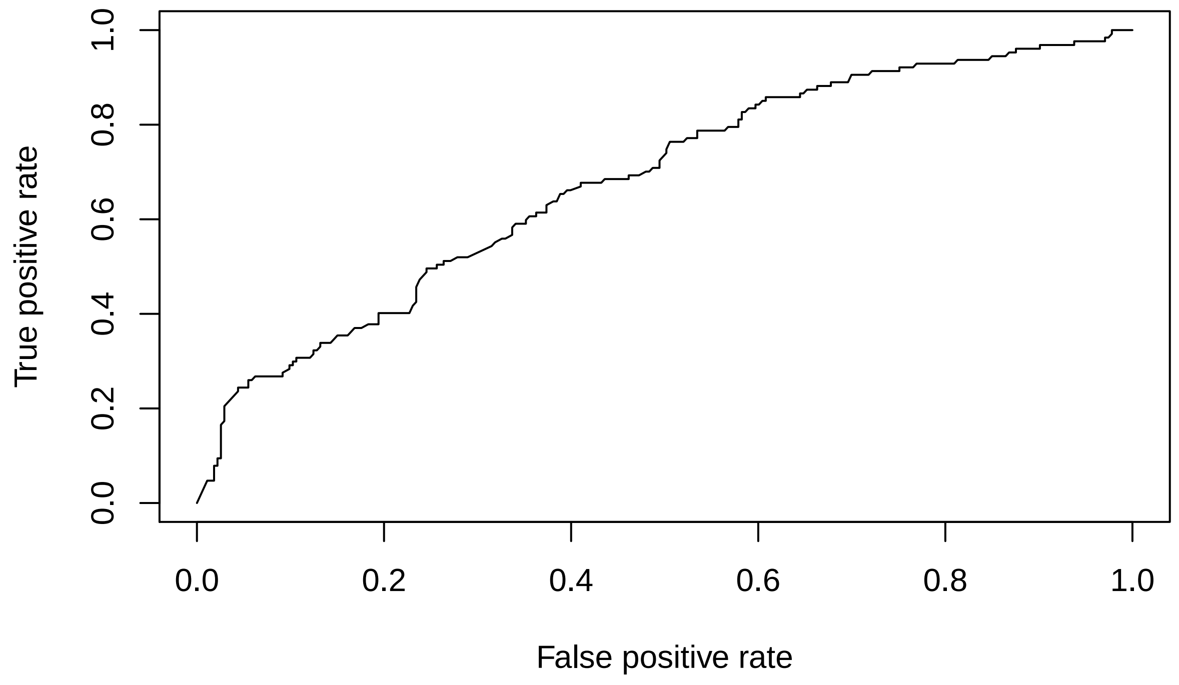
```
##
## Attaching package: 'gplots'
```

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

```
pred.prob2<-predict(mylogit2, type='response')
head(pred.prob2)
```

```
##         1         2         3         4         5         6
## 0.2577653 0.2700256 0.6779993 0.1542276 0.1218149 0.2712217
```

```
pred.prob22<-prediction(pred.prob2, mydata$admit )
pred.prob22f<-performance(pred.prob22, measure='tpr', x.measure='fpr')
plot(pred.prob22f)
```



```
auc2<-performance(pred.prob22, measure='auc')
auc2@y.values[[1]]
```

```
## [1] 0.6801073
```

Diagnostic plot

Will return many diagnostic plot

```
plot(mylogit2)
```

Interaction

Let use see how we deal an interaction. First, read data from this text file.

Columns (variables) no 2, and from 5 to 10 need to be converted to categorical (factor) variables

```
data.l<-read.table("LOWBWT.txt",header=T)
data.l[,c(2,5:10)]<-lapply(data.l[,c(2,5)],factor)
```

To simulate a binary predictor variable, we now recode LWT to LWD (LWT<110 vs >=110)

```
data.l$LWD<-findInterval(data.l$LWT,110)
data.l$LWD<-factor(data.l$LWD,labels = c("less 110",">=110"))
head(data.l$LWD,10)
```

```
##  [1] >=110    >=110    >=110    less 110 less 110 >=110    less 110
##  [8] >=110    >=110    >=110
## Levels: less 110 >=110
```

```
head(data.l$LWT,10)
```

```
##  [1] 120 130 187 105  85 150  97 128 132 165
```

```
str(data.l$LWD)
```

```
##  Factor w/ 2 levels "less 110",">=110": 2 2 2 1 1 2 1 2 2 2 ...
```

Model the relationship; outcome (LOW=0,1) with predictors of LWD and AGE interact with each other. You may try with using #

```
mod.lwd.age<-glm(LOW~LWD*AGE,family = binomial(link =logit ),data=data.l)
summary(mod.lwd.age)
```

```
##
## Call:
## glm(formula = LOW ~ LWD * AGE, family = binomial(link = logit),
##     data = data.l)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4257  -0.8554  -0.6960   1.1602   2.0329
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.16959    1.46515  -0.798   0.4247
## LWD>=110      1.94409    1.72481   1.127   0.2597
## AGE           0.05262    0.06449   0.816   0.4145
## LWD>=110:AGE -0.13220    0.07570  -1.746   0.0807 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 234.67  on 188  degrees of freedom
## Residual deviance: 221.14  on 185  degrees of freedom
## AIC: 229.14
##
## Number of Fisher Scoring iterations: 4
```

Predict our model using new data. Before doing so, we need to create a dataset containing new data

```
newdata.2<-data.frame(AGE=c(15,15,20,20),LWD=rep(c("less 110",">=110"),2))
```

Now let us predict the log odds

```
predict(mod.lwd.age,newdata=newdata.2)
```

```
##          1          2          3          4
## -0.3802252 -0.4190874 -0.1171023 -0.8169483
```

```
newdata.2
```

```
##   AGE      LWD
## 1  15 less 110
## 2  15    >=110
## 3  20 less 110
## 4  20    >=110
```

Can you prove these?

```
-1.1696+0+0.0526*15+0
```

```
## [1] -0.3806
```

```
-1.1696+1.944*1+0.0526*15-0.1322*1*15
```

```
## [1] -0.4196
```

```
-1.1696+0+0.0526*20+0
```

```
## [1] -0.1176
```

```
-1.1696+1.944*1+0.0526*20-0.1322*1*20
```

```
## [1] -0.8176
```

Resources

1. http://www.ats.ucla.edu/stat/r/dae/logit.htm
2. https://cran.r-project.org/web/packages/HSAUR/vignettes/Ch_logistic_regression_glm.pdf

Additional materials

1. http://www.shizukalab.com/toolkits/plotting-logistic-regression-in-r

First, we'll create a fake dataset of 20 individuals of different body sizes:

```
bodysize<-rnorm(20,30,2) # generates 20 values, with mean of 30 & s.d.=2
bodysize<-sort(bodysize) # sorts these values in ascending order.
survive<-c(0,0,0,0,0,1,0,1,0,0,1,1,0,1,1,1,0,1,1,1) # assign 'survival' to these 20 individuals non-ran
```

```
dat<-as.data.frame(cbind(bodysize,survive)) # saves dataframe with two columns: body size & survival
dat # just shows you what your dataset looks like. It will look something like this:
```

```
##    bodysize survive
## 1  26.45814       0
## 2  26.67988       0
## 3  27.08813       0
## 4  27.48125       0
## 5  27.72376       0
## 6  27.76024       1
## 7  27.89036       0
## 8  28.81809       1
## 9  28.84867       0
## 10 29.16689       0
## 11 29.19801       1
## 12 29.47204       1
## 13 29.56624       0
## 14 30.30645       1
## 15 30.59708       1
## 16 30.80908       1
## 17 31.85479       0
## 18 32.94948       1
## 19 33.04617       1
## 20 33.62130       1
```

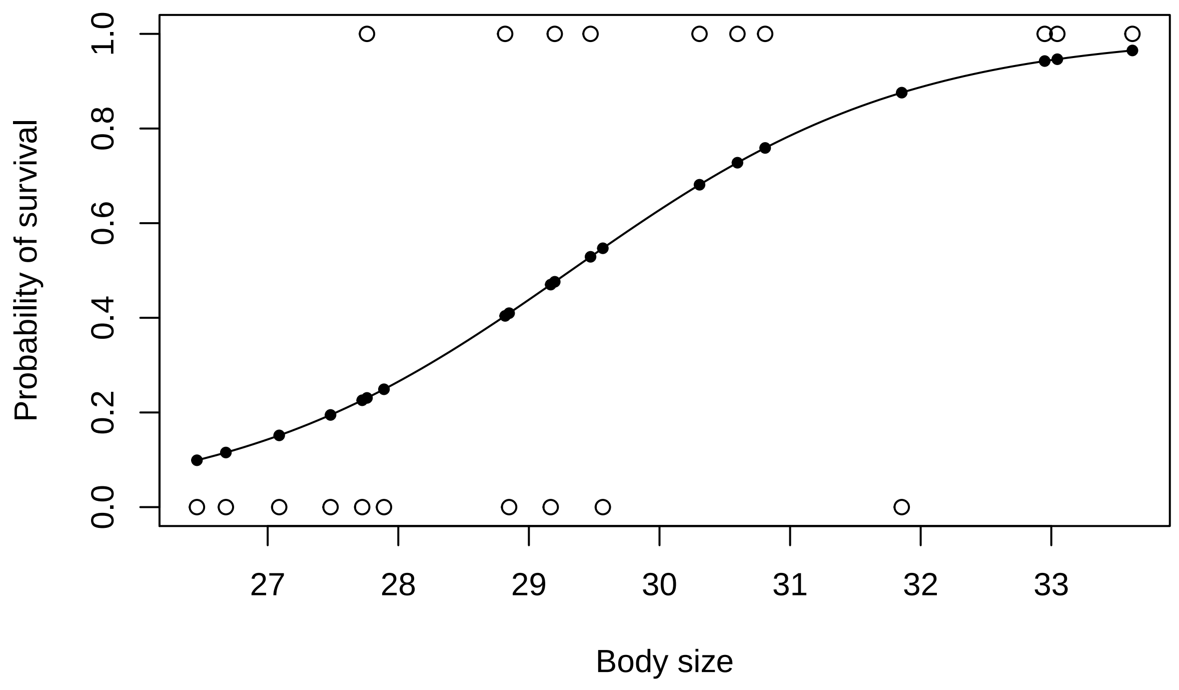
Plot

```
#quartz(title="bodysize vs. survival") # creates a quartz window with title

plot(bodysize,survive,xlab="Body size",ylab="Probability of survival") # plot with body size on x-axis
g=glm(survive~bodysize,family=binomial,dat) # run a logistic regression model (in this case, generalize

curve(predict(g,data.frame(bodysize=x),type="resp"),add=TRUE) # draws a curve based on prediction from

points(bodysize,fitted(g),pch=20) # optional: you could skip this draws an invisible set of points of b
```



2. http://www.cookbook-r.com/Statistical_analysis/Logistic_regression/

```
data(mtcars)
dat2 <- subset(mtcars, select=c(mpg, am, vs))
dat2
```

```
##                      mpg am vs
## Mazda RX4           21.0  1  0
## Mazda RX4 Wag       21.0  1  0
## Datsun 710          22.8  1  1
## Hornet 4 Drive      21.4  0  1
## Hornet Sportabout   18.7  0  0
## Valiant             18.1  0  1
## Duster 360          14.3  0  0
## Merc 240D           24.4  0  1
## Merc 230            22.8  0  1
## Merc 280            19.2  0  1
## Merc 280C           17.8  0  1
## Merc 450SE          16.4  0  0
## Merc 450SL          17.3  0  0
## Merc 450SLC         15.2  0  0
## Cadillac Fleetwood  10.4  0  0
## Lincoln Continental 10.4  0  0
## Chrysler Imperial   14.7  0  0
## Fiat 128            32.4  1  1
## Honda Civic         30.4  1  1
## Toyota Corolla      33.9  1  1
## Toyota Corona       21.5  0  1
## Dodge Challenger    15.5  0  0
## AMC Javelin         15.2  0  0
## Camaro Z28          13.3  0  0
## Pontiac Firebird    19.2  0  0
## Fiat X1-9           27.3  1  1
## Porsche 914-2       26.0  1  0
## Lotus Europa        30.4  1  1
## Ford Pantera L      15.8  1  0
## Ferrari Dino        19.7  1  0
## Maserati Bora       15.0  1  0
## Volvo 142E          21.4  1  1
```

Continuous predictor

```
# Do the logistic regression - both of these have the same effect.
# ("logit" is the default model when family is binomial.)
logr_vm <- glm(vs ~ mpg, data=dat2, family=binomial)
logr_vm <- glm(vs ~ mpg, data=dat2, family=binomial(link="logit"))
```

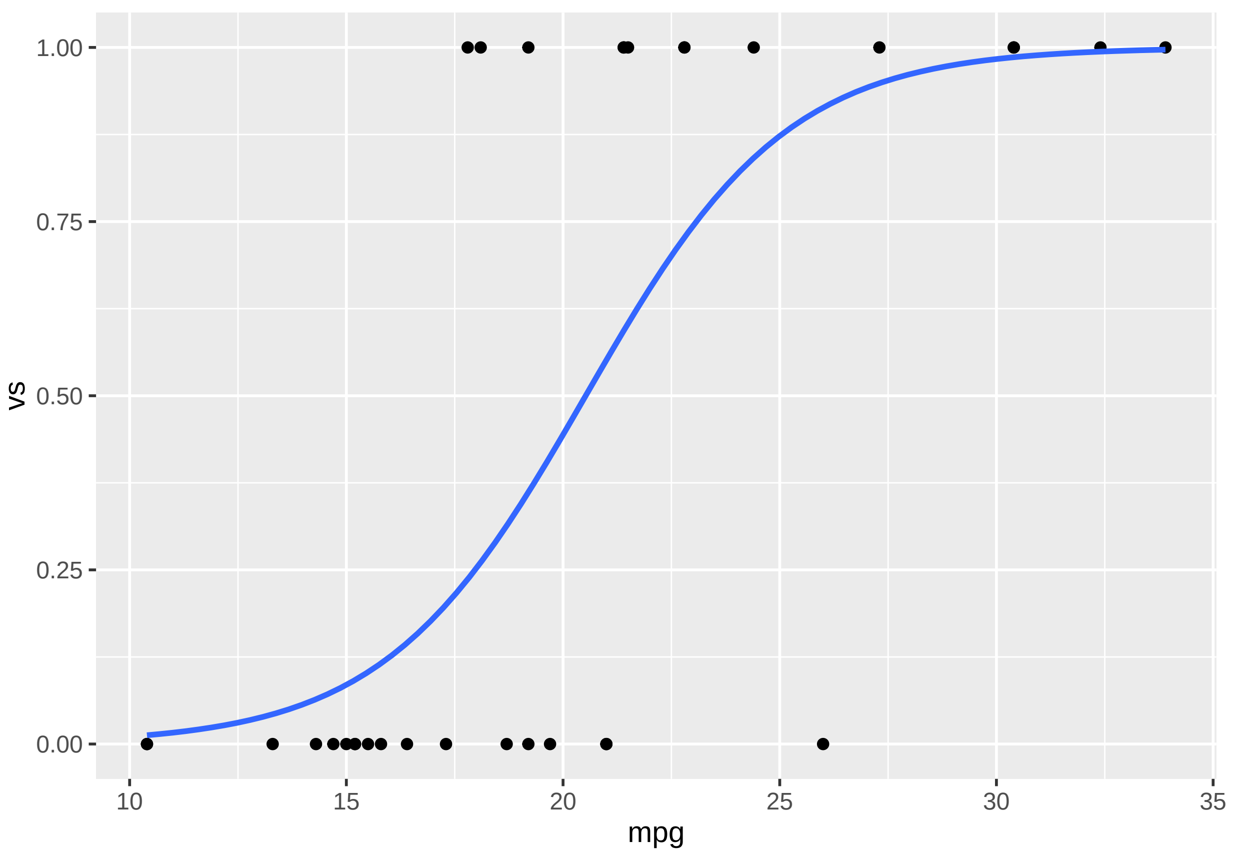
Plotting, first using ggplot2 then base graphics

```
library(ggplot2)
```

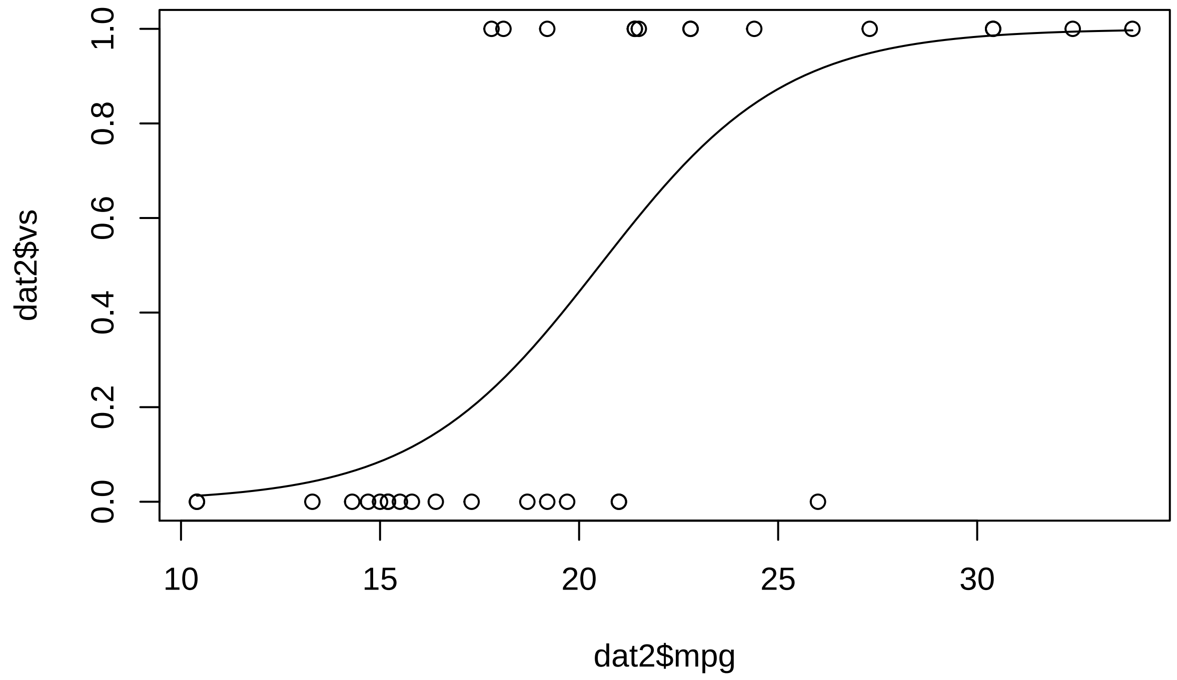
```
##
## Attaching package: 'ggplot2'

## The following objects are masked from 'package:psych':
##
##     %+%, alpha
```

```
ggplot(dat2, aes(x=mpg, y=vs)) + geom_point()+
  stat_smooth(method="glm",method.args  ="binomial",se=FALSE)
```



```
plot(dat2$mpg, dat2$vs)
curve(predict(logr_vm, data.frame(mpg=x), type="response"), add=TRUE)
```


dat2$vs
0.0
0.2
0.4
0.6
0.8
1.0
10
15
20
25
30
dat2$mpg