logistic_1.R

SANGHOOJEFFREY

Wed Jun 27 22:29:55 2018

```
# 로지스틱 회귀모형은 반응변수 Y 가 0 또는 1 로 구분되는 이분형 자료에 적용되는 모형
# 다음과 같은 선형모형을 가정한다.

# log (p /(1-p))= beta_0 + beta_1*X   : logit 변환

data(iris)

d<- iris[iris$Species == "virginica" | iris$Species =="versicolor",]  # 종이 Virginica 와 versicolor 선택
d$Species <- factor(d$Species)
str(d)  # 이범주형 자료로 변환
```

```
## 'data.frame':    100 obs. of  5 variables:
##  $ Sepal.Length: num  7 6.4 6.9 5.5 6.5 5.7 6.3 4.9 6.6 5.2 ...
##  $ Sepal.Width : num  3.2 3.2 3.1 2.3 2.8 2.8 3.3 2.4 2.9 2.7 ...
##  $ Petal.Length: num  4.7 4.5 4.9 4 4.6 4.5 4.7 3.3 4.6 3.9 ...
##  $ Petal.Width : num  1.4 1.5 1.5 1.3 1.5 1.3 1.6 1 1.3 1.4 ...
##  $ Species     : Factor w/ 2 levels "versicolor","virginica": 1 1 1 1 1 1 1 1 1 1 ...
```

```
# 만약 회귀모형으로 세운다면
# d$Species 가 factor 이므로

d$Species2 = as.numeric(d$Species)-1 #0 과 1 로 구성된 자료 생성
lm.out <- lm(Species2~Sepal.Length+Sepal.Width+Petal.Length+Petal.Width, data=d)
summary(lm.out)
```

```
##
## Call:
## lm(formula = Species2 ~ Sepal.Length + Sepal.Width + Petal.Length +
##     Petal.Width, data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.62692 -0.15178  0.01562  0.14191  0.53094
##
```

```
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.41864    0.26681  -1.569  0.11996
## Sepal.Length -0.19606    0.07223  -2.714  0.00789 **
## Sepal.Width  -0.30755    0.09452  -3.254  0.00158 **
## Petal.Length  0.38426    0.07818   4.915 3.70e-06 ***
## Petal.Width   0.68284    0.11212   6.090 2.38e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2385 on 95 degrees of freedom
## Multiple R-squared:  0.7839, Adjusted R-squared:  0.7748
## F-statistic: 86.15 on 4 and 95 DF,  p-value: < 2.2e-16

summary(lm.out$fitted.values) # 예측값을 살펴보면 -0.3, 1.49 와 같이 0 과 1 의 범
위를 넘어가는 예측값 존재

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.3080  0.1291  0.5350  0.5000  0.8740  1.4940

# 로지스틱 회귀모형의 예측값은 0 과 1 범위 내에서만 존재

out <- glm(Species2~Sepal.Length+Sepal.Width+Petal.Length+Petal.Width, data=d,
 family=binomial(link="logit"))
summary(out)

##
## Call:
## glm(formula = Species2 ~ Sepal.Length + Sepal.Width + Petal.Length +
##     Petal.Width, family = binomial(link = "logit"), data = d)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.01105  -0.00541  -0.00001   0.00677   1.78065
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -42.638     25.707  -1.659   0.0972 .
## Sepal.Length   -2.465      2.394  -1.030   0.3032
## Sepal.Width    -6.681      4.480  -1.491   0.1359
## Petal.Length    9.429      4.737   1.991   0.0465 *
## Petal.Width    18.286      9.743   1.877   0.0605 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 138.629  on 99  degrees of freedom
```

```
## Residual deviance:  11.899  on 95  degrees of freedom
## AIC: 21.899
##
## Number of Fisher Scoring iterations: 10

out$fitted.values # 예측 확률

##           51           52           53           54           55
## 1.171672e-05 4.856237e-05 1.198626e-03 4.220049e-05 1.408470e-03
##           56           57           58           59           60
## 1.018578e-04 1.305727e-03 5.351876e-10 1.458241e-05 1.481064e-05
##           61           62           63           64           65
## 3.990780e-08 3.744346e-05 9.947107e-08 7.988665e-04 1.378280e-08
##           66           67           68           69           70
## 2.828836e-06 1.326003e-03 1.481153e-08 5.959820e-02 8.712675e-08
##           71           72           73           74           75
## 4.048381e-01 3.405812e-07 2.248338e-01 4.023809e-05 1.410660e-06
##           76           77           78           79           80
## 7.060188e-06 7.124099e-04 2.760617e-01 9.651525e-04 1.290424e-10
##           81           82           83           84           85
## 8.469327e-08 5.298820e-09 8.707382e-08 8.676299e-01 2.169221e-03
##           86           87           88           89           90
## 2.129823e-04 2.979719e-04 2.551360e-04 7.884147e-07 1.109268e-05
##           91           92           93           94           95
## 3.969831e-05 1.596216e-04 4.360614e-07 8.158121e-10 1.502115e-05
##           96           97           98           99          100
## 2.541253e-07 3.085679e-06 2.309662e-06 6.163826e-11 2.344150e-06
##          101          102          103          104          105
## 1.000000e+00 9.996139e-01 9.999990e-01 9.997188e-01 9.999999e-01
##          106          107          108          109          110
## 1.000000e+00 8.908123e-01 9.999955e-01 9.999921e-01 1.000000e+00
##          111          112          113          114          115
## 9.902584e-01 9.997429e-01 9.999800e-01 9.999673e-01 9.999999e-01
##          116          117          118          119          120
## 9.999952e-01 9.976994e-01 9.999999e-01 1.000000e+00 9.204923e-01
##          121          122          123          124          125
## 9.999996e-01 9.995130e-01 1.000000e+00 9.484339e-01 9.999824e-01
##          126          127          128          129          130
## 9.995586e-01 8.245440e-01 8.022990e-01 9.999992e-01 9.712013e-01
##          131          132          133          134          135
## 9.999969e-01 9.999189e-01 9.999999e-01 2.048741e-01 9.664047e-01
##          136          137          138          139          140
## 1.000000e+00 9.999999e-01 9.964973e-01 6.691425e-01 9.998717e-01
##          141          142          143          144          145
## 1.000000e+00 9.999440e-01 9.996139e-01 1.000000e+00 1.000000e+00
##          146          147          148          149          150
## 9.999932e-01 9.991067e-01 9.989939e-01 9.999956e-01 9.776789e-01
```

```r
pred<-(ifelse(out$fitted.values>0.5,1,0)) #예측값이 0.5 이상이면 1 로 예측하고 0.5 미만이면 0 으로 예측

xtabs(~pred+d$Species2) # 실제값과 예측값에 분류표
```

```
##     d$Species2
## pred  0  1
##    0 49  1
##    1  1 49
```

```r
# 좋은 모형을 만들려면 먼저 어떤 모형이 좋은 것인지 결정해야한다.
# 이를 평가하기 위해 평가메트릭, ROC 커브, 교차검증 등이 제시됨
# p.20~30

# 평가 메트릭
# 분류가 Y, N 두 종류가 있다고 할 때 실제값과 예측값의 빈도를 그리면
# 혼돈행렬(confusion matrix)로 그려진다.

#                       실제값
#              Y                          N
# 예측값   Y   True Positive(TP)          False Positive(FP)
#          N   False Nagative(FN)         True Nagative (TN)

# Precision = TP / (TP + FP) : Y 로 예측된 것 중에 Y 의 비율
# Accuracy = TP+TN / (TP+FP+FN+TN) : 전체 예측 중 옳은 예측의 비율
# Recall   = TP / (TP+FN) : 실제로 Y 인 것들 중 예측이 Y 인 비율
# Specificity = TN / (FP + TN) : 실제로 N 인 것들 중 예측이 N 인 비율
# FP rate   = FP / (FP + TN) : Y 가 아닌데 Y 로 예측된 비율
# F measure = 2 * Precision*Recall/(Precision+Recall) : Precision 과 Recall 의 조화 평균

predicted <- c(1,0,0,1,1,1,0,0,0,1,1,1)
actual <- c(1,0,0,1,1,0,1,1,0,1,1,1)

xtabs(~predicted+actual)
```

```
##          actual
## predicted 0 1
##         0 3 2
##         1 1 6
```

```r
# Q1. Accuracy 는?

if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-proje
ct.org"); library(caret)
```

```
## Loading required package: caret
```

```
## Warning: package 'caret' was built under R version 3.4.4
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```r
# caret 패키지를 이용하면 평가 메트릭을 쉽게 계산할 수 있다.

confusionMatrix(as.factor(predicted), as.factor(actual))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 3 2
##          1 1 6
##
##                Accuracy : 0.75
##                  95% CI : (0.4281, 0.9451)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.3931
##
##                   Kappa : 0.4706
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.7500
##             Specificity : 0.7500
##          Pos Pred Value : 0.6000
##          Neg Pred Value : 0.8571
##              Prevalence : 0.3333
##          Detection Rate : 0.2500
##    Detection Prevalence : 0.4167
##       Balanced Accuracy : 0.7500
##
##        'Positive' Class : 0
##
```

```r
#iris 자료에 대한 평가메트릭
confusionMatrix(as.factor(pred), as.factor(d$Species2))
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction  0  1
##          0 49  1
##          1  1 49
##
##                Accuracy : 0.98
##                  95% CI : (0.9296, 0.9976)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.96
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.98
##             Specificity : 0.98
##          Pos Pred Value : 0.98
##          Neg Pred Value : 0.98
##              Prevalence : 0.50
##          Detection Rate : 0.49
##    Detection Prevalence : 0.50
##       Balanced Accuracy : 0.98
##
##        'Positive' Class : 0
##
```

 # ROC 커브를 통해 모형분석 결과를 정량화 시킬 수 있다.

```
if(!require(ROCR)) install.packages("ROCR", repos = "http://cran.us.r-project.
org"); library(ROCR)
```

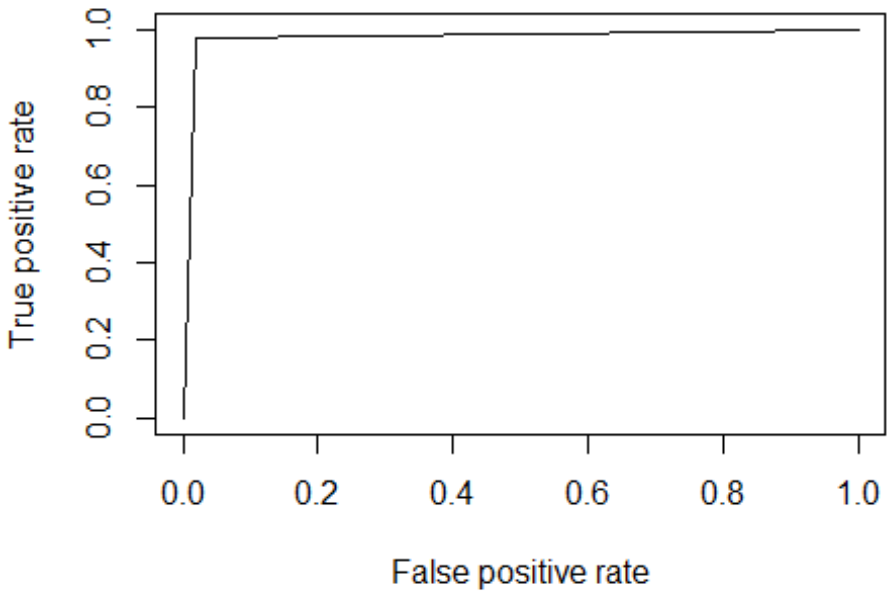
```
## Loading required package: ROCR
```

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

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

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

```
pr <- prediction(pred, d$Species2)
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf)
```




```
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.98
```

```
# AUC 를 판단하는 대략적인 기준
# excellent =  0.9~1
# good = 0.8~0.9
# fair = 0.7~0.8
# poor = 0.6~0.7
# fail = 0.5~0.6

# 주어진 데이터 전체를 사용해 모형을 세울 경우, 해당 데이터에는 잘 동작하지만
# 새로운 데이터에는 좋지 않은 성능을 보이는 모형을 만들수 있다.(과적합 문제)
# 과적합 발생 여부를 알아내려면 주어진 데이터 중 일부는 모델을 만드는 훈련데이터로
사용하고,
# 나머지는 테스트 데이터로 사용해 모형을 평가해야한다.

# 교차검증은 훈련 데이터와 테스트 데이터를 분리하여 모형을 만드는데 많이 사용되는
방법
# p. 17~18
```

```r
if(!require(cvTools)) install.packages("cvTools", repos = "http://cran.us.r-project.org"); library(cvTools)
```

```
## Loading required package: cvTools
```

```
## Warning: package 'cvTools' was built under R version 3.4.4
```

```
## Loading required package: robustbase
```

```
## Warning: package 'robustbase' was built under R version 3.4.2
```

```r
set.seed(1215124)
cvFolds(10, K=5, type="random")
```

```
##
## 5-fold CV:
## Fold   Index
##    1       9
##    2       5
##    3       8
##    4       3
##    5       1
##    1       4
##    2       7
##    3       2
##    4       6
##    5      10
```

```r
# 5 겹 교차검증에서 K=1 일때 9, 4 번째를 검증 데이터로 사용하고 나머지는 훈련 데이터로 사용
# K=2 일때 5, 7 번째를 검증 데이터로 사용하고 나머지는 훈련데이터로 사용

cvFolds(10, K=5, type="consecutive") # 연속된 자료를 검증자료로 사용할 때
```

```
##
## 5-fold CV:
## Fold   Index
##    1       1
##    1       2
##    2       3
##    2       4
##    3       5
##    3       6
##    4       7
##    4       8
##    5       9
##    5      10
```

```r
cvFolds(10, K=5, type="interleaved")
```

```
##
## 5-fold CV:
## Fold   Index
##    1       1
##    2       2
##    3       3
##    4       4
##    5       5
##    1       6
##    2       7
##    3       8
##    4       9
##    5      10
```

```r
#iris 자료에 대한 교차검증을 수행한다면....;
set.seed(15142)
cv <- cvFolds(nrow(d), K=5, R=3)

#첫번째 반복 K=1
sel<-cv$subset[which(cv$which==1),1]

train <- d[-sel,]
test <- d[sel,]

out <- glm(Species2~Sepal.Length+Sepal.Width+Petal.Length+Petal.Width, data=t
rain, family="binomial")
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
pred <- predict(out, newdata=test, type="response")
pred<-(ifelse(pred>0.5,1,0)) #예측값이 0.5 이상이면 1 로 예측하고 0.5 미만이면 0
으로 예측

pr <- prediction(pred, test$Species2)
auc <- performance(pr, measure = "auc")@y.values[[1]]
auc
```

```
## [1] 0.9166667
```

```r
#첫번째 반복 K=2
sel<-cv$subset[which(cv$which==2),1]

train <- d[-sel,]
test <- d[sel,]
```

```r
out <- glm(Species2~Sepal.Length+Sepal.Width+Petal.Length+Petal.Width, data=train, family="binomial")
pred <- predict(out, newdata=test, type="response")
pred<-(ifelse(pred>0.5,1,0)) #예측값이 0.5 이상이면 1 로 예측하고 0.5 미만이면 0 으로 예측

pr <- prediction(pred, test$Species2)
auc <- performance(pr, measure = "auc")@y.values[[1]]
auc
```

```
## [1] 0.875
```

```r
#첫번째 반복 K=3
sel<-cv$subset[which(cv$which==3),1]

train <- d[-sel,]
test <- d[sel,]

out <- glm(Species2~Sepal.Length+Sepal.Width+Petal.Length+Petal.Width, data=train, family="binomial")
pred <- predict(out, newdata=test, type="response")
pred<-(ifelse(pred>0.5,1,0)) #예측값이 0.5 이상이면 1 로 예측하고 0.5 미만이면 0 으로 예측

pr <- prediction(pred, test$Species2)
auc <- performance(pr, measure = "auc")@y.values[[1]]
auc
```

```
## [1] 0.9166667
```

```r
# 병렬처리를 통해 위의 결과를 자동으로 계산되도록 프로그래밍
if(!require(foreach)) install.packages("foreach", repos = "http://cran.us.r-project.org"); library(foreach)
```

```
## Loading required package: foreach
```

```
## Warning: package 'foreach' was built under R version 3.4.4
```

```r
set.seed(325312)
R=3
K=5
cv <- cvFolds(nrow(d), K=K, R=R)
foreach(r=1:R) %do% {
  foreach(k=1:K, .combine=c) %do% {
    sel<-cv$subset[which(cv$which==k),r]
```

```
    train <- d[-sel,]
    test <- d[sel,]

    out <- glm(Species2~Sepal.Length+Sepal.Width+Petal.Length+Petal.Width, da
ta=train, family="binomial")
    pred <- predict(out, newdata=test, type="response")

    pred1<-(ifelse(out$fitted.values>0.5,1,0)) # train 자료
    pred2<-(ifelse(pred>0.5,1,0))              # test 자료

    pr1 <- prediction(pred1, train$Species2)
    pr2 <- prediction(pred2, test$Species2)

    auc1 <- performance(pr1, measure = "auc")@y.values[[1]]  # train 자료
    auc2 <- performance(pr2, measure = "auc")@y.values[[1]]  # test 자료

    return(c(auc1, auc2))
  }
}
```

```
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## [[1]]
##  [1] 0.9749844 0.9545455 1.0000000 0.9090909 0.9750000 1.0000000 1.0000000
##  [8] 0.9090909 0.9749844 0.9545455
##
## [[2]]
##  [1] 1.0000000 0.8958333 0.9749373 0.9583333 0.9750000 1.0000000 0.9746032
##  [8] 1.0000000 0.9749844 0.9545455
##
## [[3]]
```

```
##  [1] 1.0000000 0.9500000 0.9613451 0.9615385 0.9749844 1.0000000 1.0000000
##  [8] 0.8750000 0.9749373 1.0000000
```

```
################################################################################
#######
# Another example
library(foreign)
mydata <- read.dta("http://dss.princeton.edu/training/Panel101.dta")

head(mydata)
```

```
##   country year           y y_bin        x1         x2          x3
## 1       A 1990  1342787840     1 0.2779036 -1.1079559  0.28255358
## 2       A 1991 -1899660544     0 0.3206847 -0.9487200  0.49253848
## 3       A 1992   -11234363     0 0.3634657 -0.7894840  0.70252335
## 4       A 1993  2645775360     1 0.2461440 -0.8855330 -0.09439092
## 5       A 1994  3008334848     1 0.4246230 -0.7297683  0.94613063
## 6       A 1995  3229574144     1 0.4772141 -0.7232460  1.02968037
##     opinion
## 1 Str agree
## 2     Disag
## 3     Disag
## 4     Disag
## 5     Disag
## 6 Str agree
```

```
logit <- glm(y_bin~ x1+x2+x3, family=binomial(link="logit"), data=mydata)
summary(logit)
```

```
##
## Call:
## glm(formula = y_bin ~ x1 + x2 + x3, family = binomial(link = "logit"),
##     data = mydata)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0277   0.2347   0.5542   0.7016   1.0839
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.4262     0.6390   0.667   0.5048
## x1            0.8618     0.7840   1.099   0.2717
## x2            0.3665     0.3082   1.189   0.2343
## x3            0.7512     0.4548   1.652   0.0986 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 70.056  on 69  degrees of freedom
```

```
## Residual deviance: 65.512  on 66  degrees of freedom
## AIC: 73.512
##
## Number of Fisher Scoring iterations: 5
```

```
if(!require(stargazer)) install.packages("stargazer", repos = "http://cran.us.
r-project.org");library(stargazer)
```

```
## Loading required package: stargazer
```

```
## Warning: package 'stargazer' was built under R version 3.4.4
```

```
##
## Please cite as:
```

```
##  Hlavac, Marek (2018). stargazer: Well-Formatted Regression and Summary St
atistics Tables.
```

```
##  R package version 5.2.2. https://CRAN.R-project.org/package=stargazer
```

```
stargazer(logit, type="text")
```

```
##
## ===============================================
##                         Dependent variable:
##                     ---------------------------
##                                y_bin
## -----------------------------------------------
## x1                              0.862
##                                (0.784)
##
## x2                              0.367
##                                (0.308)
##
## x3                             0.751*
##                                (0.455)
##
## Constant                        0.426
##                                (0.639)
##
## -----------------------------------------------
## Observations                     70
## Log Likelihood                -32.756
## Akaike Inf. Crit.             73.512
## ===============================================
## Note:               *p<0.1; **p<0.05; ***p<0.01
```

```
# stargazer()함수를 이용하면 로지스틱 모형의 결과를 보기 좋게 만들수 있음
```

```
# logistic 모형의 경우 오즈비(odds ratio)가 중요
```

```r
# 컴퓨터를 통한 오즈비 계산
cbind(Estimate=round(coef(logit),4),  OR=round(exp(coef(logit)),4))
```

```
##             Estimate     OR
## (Intercept)   0.4262 1.5314
## x1            0.8618 2.3674
## x2            0.3665 1.4427
## x3            0.7512 2.1196
```

```r
# 팩키지를 이용한 오즈비 계산
if(!require(mfx)) install.packages("mfx", repos = "http://cran.us.r-project.org");library(mfx)
```

```
## Loading required package: mfx
```

```
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'mfx'
```

```
## Installing package into 'C:/Users/SANGHOOJEFFREY/Documents/R/win-library/3.4'
## (as 'lib' is unspecified)
```

```
## package 'mfx' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\SANGHOOJEFFREY\AppData\Local\Temp\RtmpkrQhth\downloaded_packages
```

```
## Warning: package 'mfx' was built under R version 3.4.4
```

```
## Loading required package: sandwich
```

```
## Loading required package: lmtest
```

```
## Warning: package 'lmtest' was built under R version 3.4.4
```

```
## Loading required package: zoo
```

```
## Warning: package 'zoo' was built under R version 3.4.4
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: MASS
```

```
## Loading required package: betareg
```

```r
logitor(y_bin~ x1+x2+x3, data=mydata)
```

```
## Call:
## logitor(formula = y_bin ~ x1 + x2 + x3, data = mydata)
##
## Odds Ratio:
##    OddsRatio Std. Err.      z   P>|z|
## x1   2.36735   1.85600 1.0992 0.27168
## x2   1.44273   0.44459 1.1894 0.23427
## x3   2.11957   0.96405 1.6516 0.09861 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
logit.or = exp(coef(logit))
stargazer(logit, coef=list(logit.or), p.auto=FALSE, type="text")
```

```
##
## ============================================
##                       Dependent variable:
##                   -------------------------
##                              y_bin
## --------------------------------------------
## x1                            2.367
##                              (0.784)
##
## x2                            1.443
##                              (0.308)
##
## x3                           2.120*
##                              (0.455)
##
## Constant                      1.531
##                              (0.639)
##
## --------------------------------------------
## Observations                   70
## Log Likelihood               -32.756
## Akaike Inf. Crit.            73.512
## ============================================
## Note:             *p<0.1; **p<0.05; ***p<0.01
```

```
allmean <- data.frame(x1=mean(mydata$x1),
                      x2=mean(mydata$x2),
                      x3=mean(mydata$x3))
# 전체 설명변수의 평균 값을 생성
allmean
```

```
##          x1        x2       x3
## 1 0.6480006 0.1338694 0.761851
```

```r
allmean$pred.prob <- predict(logit, newdata=allmean, type="response")
# 전체 설명변수의 평균값으로 로지스틱회귀모형을 실행했을 때 확률값 계산
# 즉, 전체 설명변수가 평균값일 때, y 가 1 이될 확률은 약 83% 임
allmean
```

```
##          x1        x2       x3 pred.prob
## 1 0.6480006 0.1338694 0.761851 0.8328555
```

```r
logit <- glm(y_bin ~ x1+x2+x3+opinion, family=binomial(link="logit"), data=my
data)
# 범주형 자료가 들어간 로지스틱 회귀모형을 세워보자.

allmean <- data.frame(x1=rep(mean(mydata$x1),4),
                      x2=rep(mean(mydata$x2),4),
                      x3=rep(mean(mydata$x3),4),
                      opinion=as.factor(c("Str agree","Agree","Disag","Str di
sag")))
# 각 설명변수의 평균값과 의견에 따른 y 가 1 이될 확률을 계산해보자.

allmean <- cbind(allmean,predict(logit, newdata=allmean, type="response", se.
fit=TRUE))

allmean
```

```
##          x1        x2       x3   opinion       fit     se.fit
## 1 0.6480006 0.1338694 0.761851 Str agree 0.8764826 0.07394431
## 2 0.6480006 0.1338694 0.761851     Agree 0.5107928 0.15099064
## 3 0.6480006 0.1338694 0.761851     Disag 0.9077609 0.06734568
## 4 0.6480006 0.1338694 0.761851 Str disag 0.9339310 0.06446677
##   residual.scale
## 1              1
## 2              1
## 3              1
## 4              1
```

```r
# 설명변수가 평균값일 때, 의견에 따라 y 가 1 이될 확률을 알 수 있다.
# 예측 확률과 그에 대한 표준오차가 있으므로 95% 예측확률의 신뢰구간도 구할 수 있다.

# Renaming "fit" and "se.fit" columns
names(allmean)[names(allmean)=="fit"] = "prob"
names(allmean)[names(allmean)=="se.fit"] = "se.prob"

# Estimating confidence intervals
allmean$ll = allmean$prob - 1.96*allmean$se.prob
allmean$ul = allmean$prob + 1.96*allmean$se.prob
allmean
```

```
##          x1        x2       x3   opinion      prob    se.prob
## 1 0.6480006 0.1338694 0.761851 Str agree 0.8764826 0.07394431
## 2 0.6480006 0.1338694 0.761851     Agree 0.5107928 0.15099064
## 3 0.6480006 0.1338694 0.761851     Disag 0.9077609 0.06734568
## 4 0.6480006 0.1338694 0.761851 Str disag 0.9339310 0.06446677
##   residual.scale        ll        ul
## 1              1 0.7315518 1.0214134
## 2              1 0.2148511 0.8067344
## 3              1 0.7757634 1.0397585
## 4              1 0.8075762 1.0602859
```

```
# errorplot 을 통한 의견에 따른 결과 시각화
if(!require(Hmisc)) install.packages("Hmisc", repos = "http://cran.us.r-proje
ct.org");library(Hmisc)
```

```
## Loading required package: Hmisc
```

```
## Warning: package 'Hmisc' was built under R version 3.4.4
```

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

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:robustbase':
##
##     heart
```

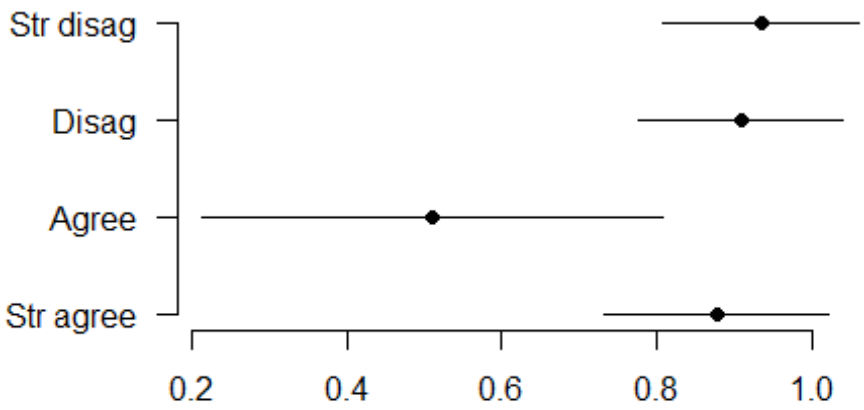
```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: Formula
```

```
##
## Attaching package: 'Hmisc'
```

```
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
errbar(allmean$opinion, allmean$prob, allmean$ul, allmean$ll)
```




```
# 평가 메트릭을 통한 결과확인
pred.opi<-ifelse(logit$fitted.values>0.5,1,0)
xtabs(~pred.opi+mydata$y_bin)
```

```
##         mydata$y_bin
## pred.opi  0  1
##        0  4  4
##        1 10 52
```

```
confusionMatrix(as.factor(pred.opi), as.factor(mydata$y_bin))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  4  4
##          1 10 52
##
##                Accuracy : 0.8
##                  95% CI : (0.6873, 0.8861)
##     No Information Rate : 0.8
##     P-Value [Acc > NIR] : 0.5709
##
##                   Kappa : 0.2553
##  Mcnemar's Test P-Value : 0.1814
##
```

```
##             Sensitivity : 0.28571
##             Specificity : 0.92857
##          Pos Pred Value : 0.50000
##          Neg Pred Value : 0.83871
##              Prevalence : 0.20000
##          Detection Rate : 0.05714
##    Detection Prevalence : 0.11429
##       Balanced Accuracy : 0.60714
##
##        'Positive' Class : 0
##
```

```
# 다항 로지스틱 회귀모형은 반응변수 Y 가 두개가 아니라 여러 개가 될 수 있는 경우
# nnet 팩키지의 multinom()로 쉽게 모델링 할 수 있다.
```

```
if (!require(nnet)) install.packages("nnet", repos = "http://cran.us.r-projec
t.org"); library(nnet)
```

```
## Loading required package: nnet
```

```
m <- multinom(Species~Sepal.Length+Sepal.Width+Petal.Length+Petal.Width, data
=iris)
```

```
## # weights:  18 (10 variable)
## initial  value 164.791843
## iter  10 value 16.177348
## iter  20 value 7.111438
## iter  30 value 6.182999
## iter  40 value 5.984028
## iter  50 value 5.961278
## iter  60 value 5.954900
## iter  70 value 5.951851
## iter  80 value 5.950343
## iter  90 value 5.949904
## iter 100 value 5.949867
## final  value 5.949867
## stopped after 100 iterations
```

```
m$fitted.values
```

```
##           setosa   versicolor    virginica
## 1   1.000000e+00 1.526406e-09 2.716417e-36
## 2   9.999996e-01 3.536476e-07 2.883729e-32
## 3   1.000000e+00 4.443506e-08 6.103424e-34
## 4   9.999968e-01 3.163905e-06 7.117010e-31
## 5   1.000000e+00 1.102983e-09 1.289946e-36
## 6   1.000000e+00 3.521573e-10 1.344907e-35
## 7   1.000000e+00 4.098064e-08 3.016154e-33
## 8   1.000000e+00 2.615330e-08 2.972971e-34
```

```
## 9   9.999871e-01 1.294210e-05 7.048364e-30
## 10  9.999992e-01 8.386603e-07 1.454198e-32
## 11  1.000000e+00 2.161864e-10 1.241888e-37
## 12  9.999997e-01 3.238036e-07 1.545112e-32
## 13  9.999992e-01 8.320656e-07 1.402024e-32
## 14  9.999998e-01 1.776283e-07 6.091969e-34
## 15  1.000000e+00 2.490019e-14 4.289244e-44
## 16  1.000000e+00 5.099113e-14 5.053040e-42
## 17  1.000000e+00 1.180774e-12 1.043681e-39
## 18  1.000000e+00 1.119797e-09 1.233997e-35
## 19  1.000000e+00 2.229749e-10 1.278090e-36
## 20  1.000000e+00 3.414358e-10 1.306813e-36
## 21  9.999999e-01 5.088458e-08 1.418328e-33
## 22  1.000000e+00 5.983234e-10 2.761055e-35
## 23  1.000000e+00 3.282647e-11 2.381898e-39
## 24  9.999998e-01 2.467861e-07 6.662407e-30
## 25  9.999768e-01 2.323802e-05 1.868716e-29
## 26  9.999965e-01 3.538327e-06 1.482164e-30
## 27  9.999999e-01 5.849351e-08 6.536682e-32
## 28  1.000000e+00 3.674991e-09 1.310414e-35
## 29  1.000000e+00 2.112377e-09 5.720335e-36
## 30  9.999968e-01 3.188981e-06 7.381858e-31
## 31  9.999956e-01 4.413191e-06 1.554498e-30
## 32  1.000000e+00 1.585769e-09 2.578398e-34
## 33  1.000000e+00 2.696754e-11 2.849881e-40
## 34  1.000000e+00 3.875622e-13 2.425003e-42
## 35  9.999994e-01 6.152555e-07 6.606045e-32
## 36  1.000000e+00 2.079286e-09 5.317228e-36
## 37  1.000000e+00 4.138112e-11 1.071492e-38
## 38  1.000000e+00 2.595111e-09 6.271520e-37
## 39  9.999987e-01 1.303796e-06 1.422388e-31
## 40  1.000000e+00 1.515201e-08 1.346082e-34
## 41  1.000000e+00 4.651074e-10 2.558009e-36
## 42  9.997542e-01 2.458213e-04 1.376952e-26
## 43  9.999998e-01 2.285045e-07 6.575528e-33
## 44  1.000000e+00 1.317919e-08 2.900340e-31
## 45  9.999999e-01 7.470478e-08 7.649899e-32
## 46  9.999996e-01 4.478126e-07 2.893285e-31
## 47  1.000000e+00 1.934115e-09 3.064974e-36
## 48  9.999997e-01 3.187312e-07 1.436229e-32
## 49  1.000000e+00 3.731511e-10 2.742847e-37
## 50  1.000000e+00 1.503286e-08 1.297787e-34
## 51  2.427101e-07 9.999877e-01 1.201699e-05
## 52  2.160475e-07 9.999501e-01 4.968516e-05
## 53  4.640834e-09 9.987828e-01 1.217158e-03
## 54  4.185792e-10 9.999567e-01 4.326447e-05
## 55  2.752538e-09 9.985711e-01 1.428890e-03
## 56  7.824187e-11 9.998954e-01 1.045901e-04
## 57  2.356899e-08 9.986727e-01 1.327314e-03
```

```
## 58  3.195371e-07 9.999997e-01 5.641233e-10
## 59  6.116463e-09 9.999850e-01 1.497847e-05
## 60  1.501151e-08 9.999848e-01 1.523161e-05
## 61  9.809848e-10 1.000000e+00 4.165185e-08
## 62  1.773719e-07 9.999615e-01 3.834000e-05
## 63  1.060055e-09 9.999999e-01 1.034374e-07
## 64  1.308456e-10 9.991850e-01 8.150241e-04
## 65  4.002682e-05 9.999600e-01 1.436141e-08
## 66  1.418052e-06 9.999957e-01 2.908759e-06
## 67  4.799737e-10 9.986481e-01 1.351871e-03
## 68  6.658268e-09 1.000000e+00 1.551529e-08
## 69  1.127345e-11 9.401019e-01 5.989806e-02
## 70  9.220385e-09 9.999999e-01 9.072544e-08
## 71  2.958914e-10 5.945365e-01 4.054635e-01
## 72  8.608392e-07 9.999988e-01 3.522422e-07
## 73  7.324234e-13 7.743208e-01 2.256792e-01
## 74  2.950369e-11 9.999586e-01 4.141866e-05
## 75  1.473401e-07 9.999984e-01 1.455234e-06
## 76  3.439354e-07 9.999924e-01 7.246952e-06
## 77  6.017178e-10 9.992755e-01 7.245125e-04
## 78  2.112470e-10 7.236305e-01 2.763695e-01
## 79  1.784210e-09 9.990177e-01 9.822717e-04
## 80  8.317614e-06 9.999917e-01 1.361048e-10
## 81  9.293464e-09 9.999999e-01 8.816365e-08
## 82  2.833280e-08 1.000000e+00 5.553317e-09
## 83  2.136523e-07 9.999997e-01 9.050639e-08
## 84  1.096390e-14 1.323524e-01 8.676476e-01
## 85  1.609647e-10 9.977885e-01 2.211499e-03
## 86  1.892766e-07 9.997823e-01 2.175106e-04
## 87  2.692561e-08 9.996965e-01 3.034535e-04
## 88  1.105514e-10 9.997399e-01 2.600700e-04
## 89  7.714596e-08 9.999991e-01 8.170920e-07
## 90  2.388398e-09 9.999886e-01 1.141228e-05
## 91  1.403301e-11 9.999591e-01 4.089587e-05
## 92  1.299698e-09 9.998366e-01 1.633724e-04
## 93  2.152323e-08 9.999995e-01 4.518083e-07
## 94  2.308979e-07 9.999998e-01 8.584159e-10
## 95  1.362045e-09 9.999845e-01 1.546367e-05
## 96  2.350697e-08 9.999997e-01 2.643923e-07
## 97  1.341431e-08 9.999968e-01 3.187736e-06
## 98  4.945474e-08 9.999976e-01 2.382636e-06
## 99  2.224095e-04 9.997776e-01 6.500522e-11
## 100 2.333746e-08 9.999976e-01 2.420920e-06
## 101 9.453717e-25 2.718072e-10 1.000000e+00
## 102 2.762230e-17 3.922358e-04 9.996078e-01
## 103 2.413930e-20 9.974371e-07 9.999990e-01
## 104 1.039086e-18 2.851578e-04 9.997148e-01
## 105 4.877802e-22 9.409138e-08 9.999999e-01
## 106 8.139586e-26 4.698713e-09 1.000000e+00
```

```
## 107 2.747116e-14 1.091926e-01 8.908074e-01
## 108 1.841814e-22 4.609074e-06 9.999954e-01
## 109 4.655966e-22 8.093448e-06 9.999919e-01
## 110 1.116285e-20 7.196079e-09 1.000000e+00
## 111 3.360175e-12 9.861345e-03 9.901387e-01
## 112 2.824675e-17 2.619406e-04 9.997381e-01
## 113 2.887245e-17 2.057044e-05 9.999794e-01
## 114 1.356507e-18 3.348943e-05 9.999665e-01
## 115 6.643324e-20 8.391928e-08 9.999999e-01
## 116 1.443873e-16 4.987152e-06 9.999950e-01
## 117 2.506556e-16 2.325939e-03 9.976741e-01
## 118 8.132508e-22 7.823403e-08 9.999999e-01
## 119 1.539275e-32 6.473411e-13 1.000000e+00
## 120 2.586465e-16 7.964338e-02 9.203566e-01
## 121 5.888460e-19 3.959256e-07 9.999996e-01
## 122 6.580602e-16 4.950994e-04 9.995049e-01
## 123 3.543398e-27 3.830263e-09 1.000000e+00
## 124 7.099730e-13 5.193896e-02 9.480610e-01
## 125 1.158605e-17 1.805360e-05 9.999819e-01
## 126 1.014284e-17 4.479026e-04 9.995521e-01
## 127 1.384328e-11 1.760948e-01 8.239052e-01
## 128 1.238609e-11 1.980731e-01 8.019269e-01
## 129 5.264982e-21 7.894776e-07 9.999992e-01
## 130 1.067125e-15 2.892881e-02 9.710712e-01
## 131 2.185577e-21 3.215285e-06 9.999968e-01
## 132 9.900467e-17 8.276525e-05 9.999172e-01
## 133 1.158989e-21 1.274946e-07 9.999999e-01
## 134 5.926801e-13 7.939466e-01 2.060534e-01
## 135 8.716903e-19 3.353546e-02 9.664645e-01
## 136 1.196029e-21 1.736953e-08 1.000000e+00
## 137 2.573884e-19 1.415958e-07 9.999999e-01
## 138 5.272004e-16 3.535048e-03 9.964650e-01
## 139 4.984248e-11 3.310585e-01 6.689415e-01
## 140 3.159583e-15 1.313812e-04 9.998686e-01
## 141 6.087418e-20 5.142118e-08 9.999999e-01
## 142 1.851909e-13 5.774763e-05 9.999423e-01
## 143 2.762230e-17 3.922358e-04 9.996078e-01
## 144 2.348662e-21 4.707320e-08 1.000000e+00
## 145 2.720648e-20 1.227942e-08 1.000000e+00
## 146 7.661759e-16 7.065708e-06 9.999929e-01
## 147 7.146172e-16 9.093936e-04 9.990906e-01
## 148 1.470964e-14 1.023609e-03 9.989764e-01
## 149 6.009635e-17 4.504137e-06 9.999955e-01
## 150 2.726745e-14 2.243538e-02 9.775646e-01
```

```r
# 1 번째 50 번째 100 번째 자료의 모형의 통한 예측결과는?
predict(m, newdata=iris[c(1,50,100),], type='class')
```

```
## [1] setosa     setosa     versicolor
## Levels: setosa versicolor virginica

# 만약 각 분류에 속할 확률을 예측하고자 한다면?
predict(m, newdata=iris[c(1,50,100),], type='probs')

##           setosa   versicolor    virginica
## 1   1.000000e+00 1.526406e-09 2.716417e-36
## 50  1.000000e+00 1.503286e-08 1.297787e-34
## 100 2.333746e-08 9.999976e-01 2.420920e-06

# 원자료의 다항회귀모형을 통한 결과
predicted <- predict(m, newdata=iris)

xtabs(~predicted+iris$Species) #분할표를 이용한 결과

##             iris$Species
## predicted    setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         49         1
##   virginica       0          1        49

confusionMatrix(as.factor(predicted), as.factor(iris$Species)) #혼돈 행렬을 이
용한 결과

## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         49         1
##   virginica       0          1        49
##
## Overall Statistics
##
##                Accuracy : 0.9867
##                  95% CI : (0.9527, 0.9984)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.98
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.9800           0.9800
## Specificity                 1.0000            0.9900           0.9900
```

```
## Pos Pred Value              1.0000            0.9800           0.9800
## Neg Pred Value              1.0000            0.9900           0.9900
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.3267           0.3267
## Detection Prevalence        0.3333            0.3333           0.3333
## Balanced Accuracy           1.0000            0.9850           0.9850
```