R을 활용한 의생명 데이터 분석 - Part 2. R을 활용한 의생명 기계학습 분석

Friday May 31 2019

Contents

1 Simple Linear Regression with lm

```
# install.packages('car')
library(car)
```

```
## Loading required package: carData
```

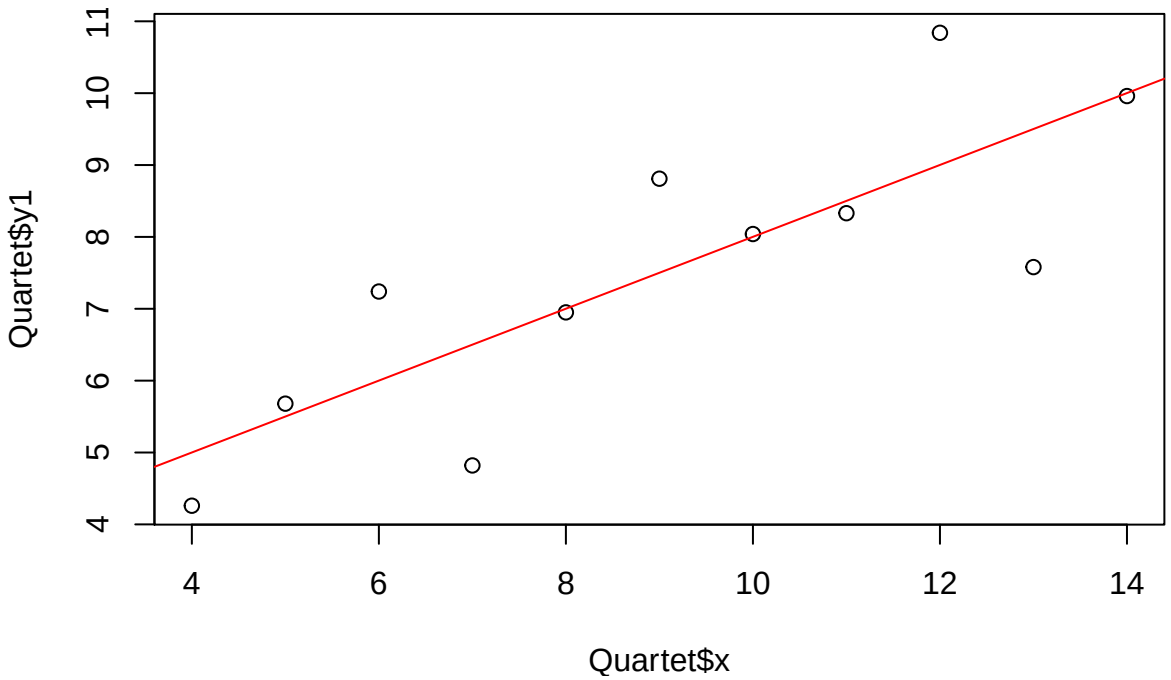
```
data(Quartet)
str(Quartet)
```

```
## 'data.frame':    11 obs. of  6 variables:
##  $ x : int  10 8 13 9 11 14 6 4 12 7 ...
##  $ y1: num  8.04 6.95 7.58 8.81 8.33 ...
##  $ y2: num  9.14 8.14 8.74 8.77 9.26 8.1 6.13 3.1 9.13 7.26 ...
##  $ y3: num  7.46 6.77 12.74 7.11 7.81 ...
##  $ x4: int  8 8 8 8 8 8 8 19 8 8 ...
##  $ y4: num  6.58 5.76 7.71 8.84 8.47 7.04 5.25 12.5 5.56 7.91 ...
```

```
head(Quartet)
```

```
##    x   y1   y2    y3 x4   y4
## 1 10 8.04 9.14  7.46  8 6.58
## 2  8 6.95 8.14  6.77  8 5.76
## 3 13 7.58 8.74 12.74  8 7.71
## 4  9 8.81 8.77  7.11  8 8.84
## 5 11 8.33 9.26  7.81  8 8.47
## 6 14 9.96 8.10  8.84  8 7.04
```

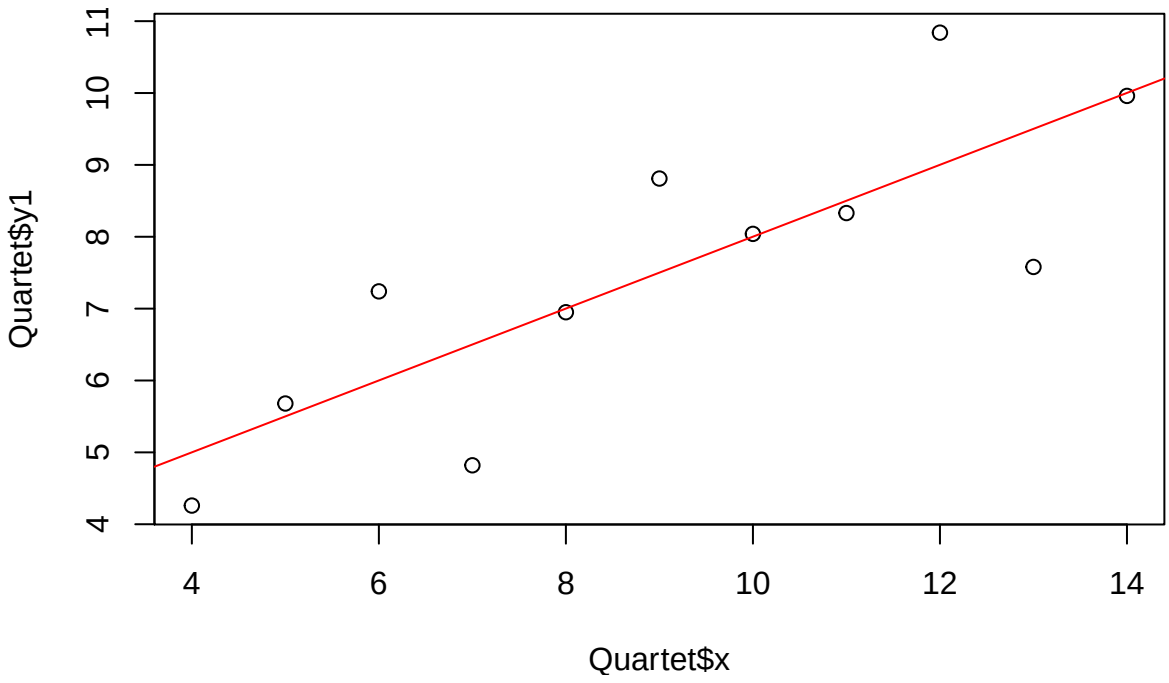
```
# Linear regression
plot(Quartet$x, Quartet$y1)
lmfit = lm(Quartet$y1 ~ Quartet$x)
abline(lmfit, col = "red")
```



```
lmfit
```

```
##
## Call:
## lm(formula = Quartet$y1 ~ Quartet$x)
##
## Coefficients:
## (Intercept)    Quartet$x
##      3.0001       0.5001
```

```
# Least squares fit
plot(Quartet$x, Quartet$y1)
lmfit2 = lsfit(Quartet$x, Quartet$y1)
abline(lmfit2, col = "red")
```



1.1 Summarizing Linear Model Fits

```
summary(lmfit)
```

```
##
## Call:
## lm(formula = Quartet$y1 ~ Quartet$x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.92127 -0.45577 -0.04136  0.70941  1.83882
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.0001     1.1247   2.667  0.02573 *
## Quartet$x     0.5001     0.1179   4.241  0.00217 **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.237 on 9 degrees of freedom
## Multiple R-squared:  0.6665, Adjusted R-squared:  0.6295
## F-statistic: 17.99 on 1 and 9 DF,  p-value: 0.00217
```

1.2 See also Advanced

```
coefficients(lmfit)
```

```
## (Intercept)   Quartet$x
##   3.0000909   0.5000909
```

```
confint(lmfit, level = 0.95)  # Confidence interval
```

```
##                 2.5 %    97.5 %
## (Intercept) 0.4557369 5.5444449
## Quartet$x   0.2333701 0.7668117
```

```
fitted(lmfit)  # Extract model fitted values
```

```
##         1         2         3         4         5         6         7
##  8.001000  7.000818  9.501273  7.500909  8.501091 10.001364  6.000636
##         8         9        10        11
##  5.000455  9.001182  6.500727  5.500545
```

```
residuals(lmfit)  # Extract model residuals
```

```
##           1           2           3           4           5           6
##  0.03900000 -0.05081818 -1.92127273  1.30909091 -0.17109091 -0.04136364
##           7           8           9          10          11
##  1.23936364 -0.74045455  1.83881818 -1.68072727  0.17945455
```

```
anova(lmfit)  # Compute analysis of variance tables for fitted model object
```

```
## Analysis of Variance Table
##
## Response: Quartet$y1
##           Df Sum Sq Mean Sq F value  Pr(>F)
## Quartet$x  1 27.510 27.5100   17.99 0.00217 **
## Residuals  9 13.763  1.5292
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
vcov(lmfit)  # Calculate variance-covariance matrix for a fitted model object
```

```
##             (Intercept)   Quartet$x
## (Intercept)   1.2650553 -0.12511536
## Quartet$x    -0.1251154  0.01390171
```

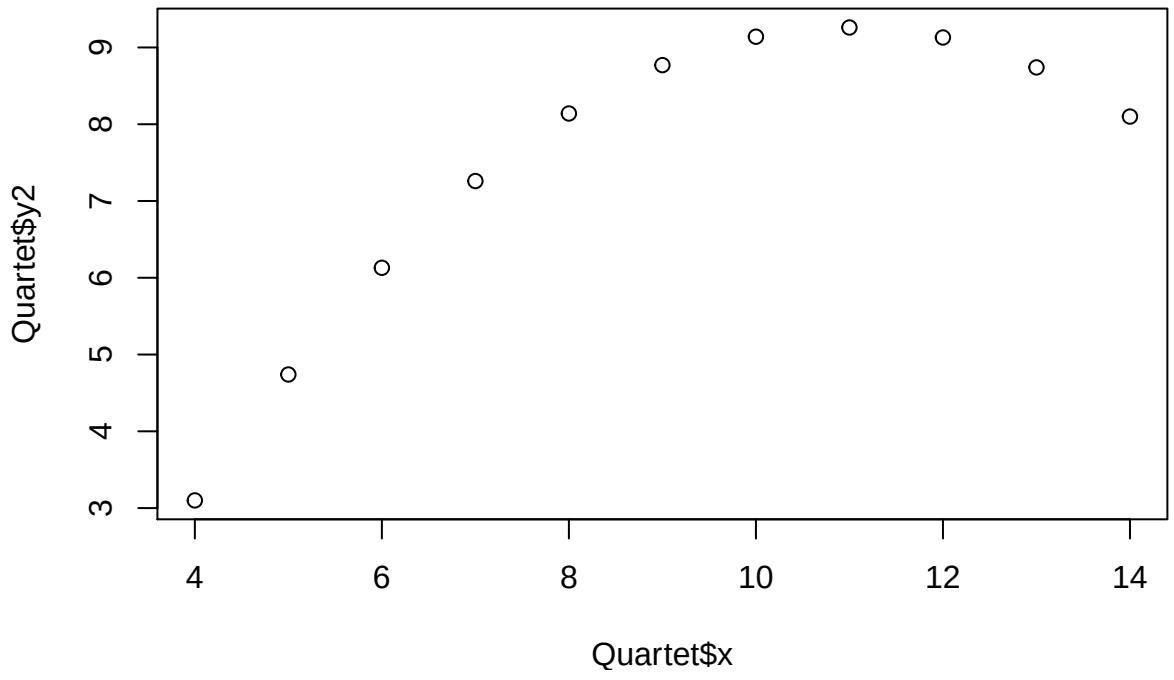
```
influence(lmfit)  # Diagnose quality of regression fits
```

```
## $hat
##          1          2          3          4          5          6
## 0.10000000 0.10000000 0.23636364 0.09090909 0.12727273 0.31818182
##          7          8          9         10         11
## 0.17272727 0.31818182 0.17272727 0.12727273 0.23636364
```

```
##
## $coefficients
##      (Intercept)     Quartet$x
## 1   0.0003939394  3.939394e-04
## 2  -0.0097529844  5.133150e-04
## 3   0.5946796537 -9.148918e-02
## 4   0.1309090909 -1.323195e-18
## 5   0.0142575758 -3.564394e-03
## 6   0.0193030303 -2.757576e-03
## 7   0.5039170829 -4.085814e-02
## 8  -0.5430000000  4.936364e-02
## 9  -0.3435154845  6.062038e-02
## 10 -0.4902121212  3.501515e-02
## 11  0.0982727273 -8.545455e-03
##
## $sigma
##        1        2        3        4        5        6        7        8
## 1.311535 1.311479 1.056460 1.218483 1.310017 1.311496 1.219936 1.272721
##        9       10       11
## 1.099742 1.147055 1.309605
##
## $wt.res
##           1           2           3           4           5           6
##  0.03900000 -0.05081818 -1.92127273  1.30909091 -0.17109091 -0.04136364
##           7           8           9          10          11
##  1.23936364 -0.74045455  1.83881818 -1.68072727  0.17945455
```

1.3 Using Linear Regression to Predict Unknown Values

```
plot(Quartet$x, Quartet$y2)
```



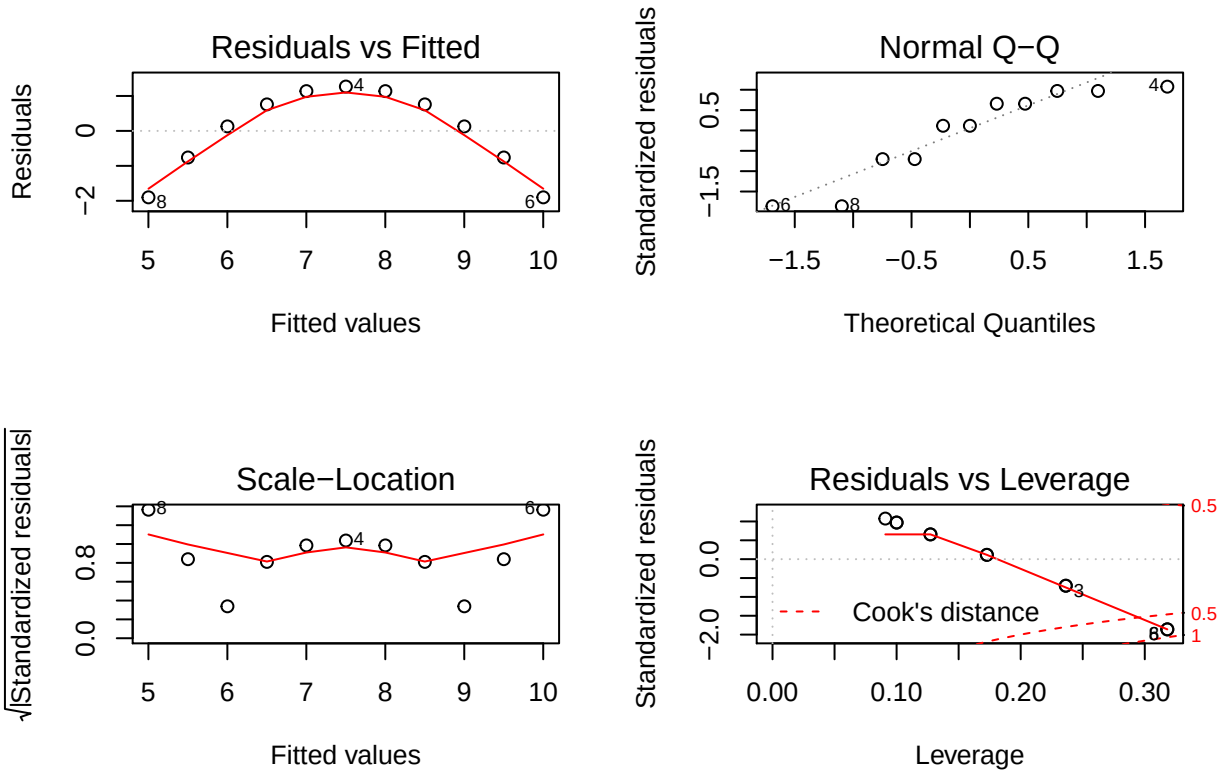
```r
lmfit <- lm(y2 ~ x, Quartet)

newdata = data.frame(x = c(3, 6, 15))

predict(lmfit, newdata, interval = "confidence", level = 0.95)
```

```
##         fit      lwr       upr
## 1  4.500909 2.691027  6.310791
## 2  6.000909 4.837726  7.164092
## 3 10.500909 8.691027 12.310791
```

```r
predict(lmfit, newdata, interval = "predict")
```

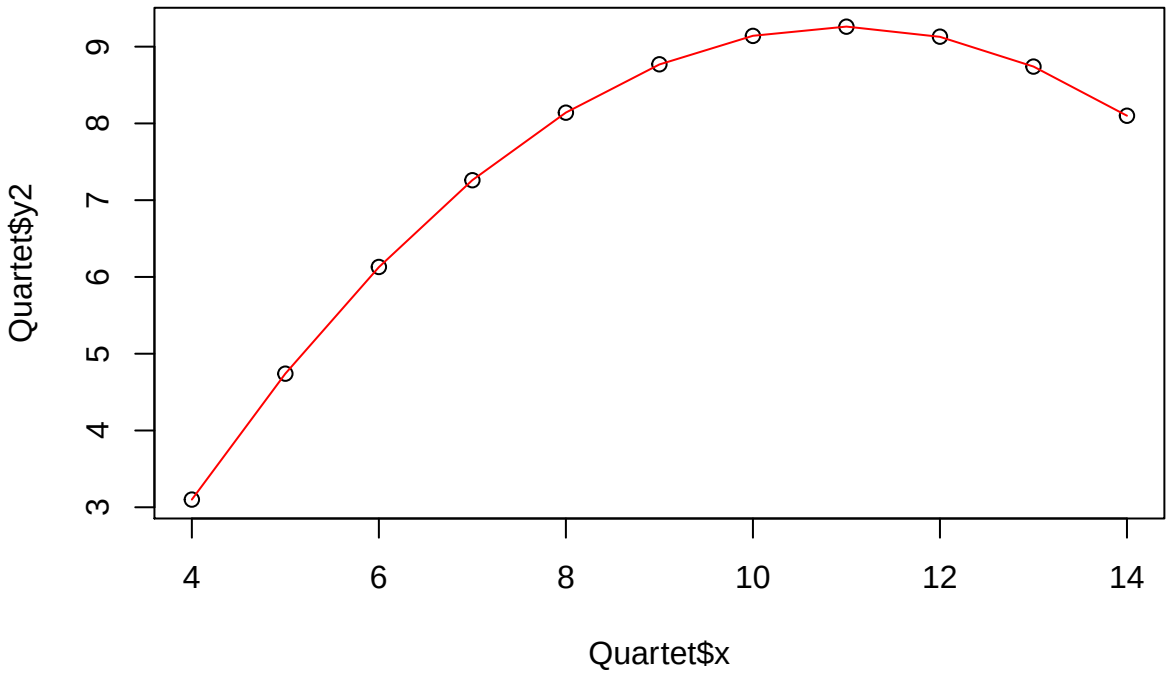
```
##         fit      lwr       upr
## 1  4.500909 1.167922  7.833896
## 2  6.000909 2.970047  9.031771
## 3 10.500909 7.167922 13.833896
```

```r
lmfit <- lm(y2 ~ x, Quartet)
par(mfrow = c(2, 2))
plot(lmfit)
```



2 Fitting a polynomial regression model with lm

```r
lmfit = lm(Quartet$y2 ~ I(Quartet$x) + I(Quartet$x^2))
lmfit = lm(Quartet$y2 ~ poly(Quartet$x, 2))
plot(Quartet$x, Quartet$y2)
lines(sort(Quartet$x), lmfit$fit[order(Quartet$x)], col = "red")
```




```
lmfit = lm(Quartet$y2 ~ I(Quartet$x) + I(Quartet$x^2))
```

3 Generalized Addictive Model (GAM) Advanced

Fit Generalized Addictive Model to Data

```
# install.packages('mgcv')
library(mgcv)
```

```
## Loading required package: nlme
```

```
## This is mgcv 1.8-24. For overview type 'help("mgcv-package")'.
```

```
# install.packages('MASS')
library(MASS)
attach(Boston)
str(Boston)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

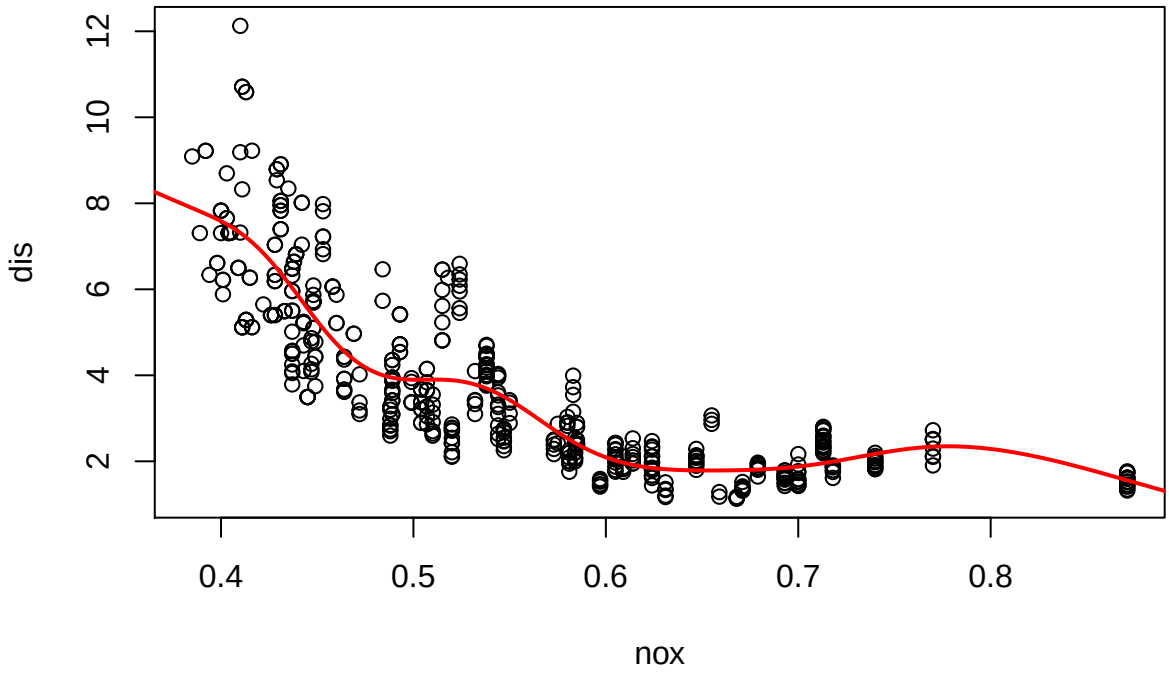
```r
fit <- gam(dis ~ s(nox))
summary(fit)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## dis ~ s(nox)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.79504    0.04507   84.21   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##          edf Ref.df   F p-value
## s(nox) 8.434  8.893 189  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.768   Deviance explained = 77.2%
## GCV = 1.0472  Scale est. = 1.0277    n = 506
```

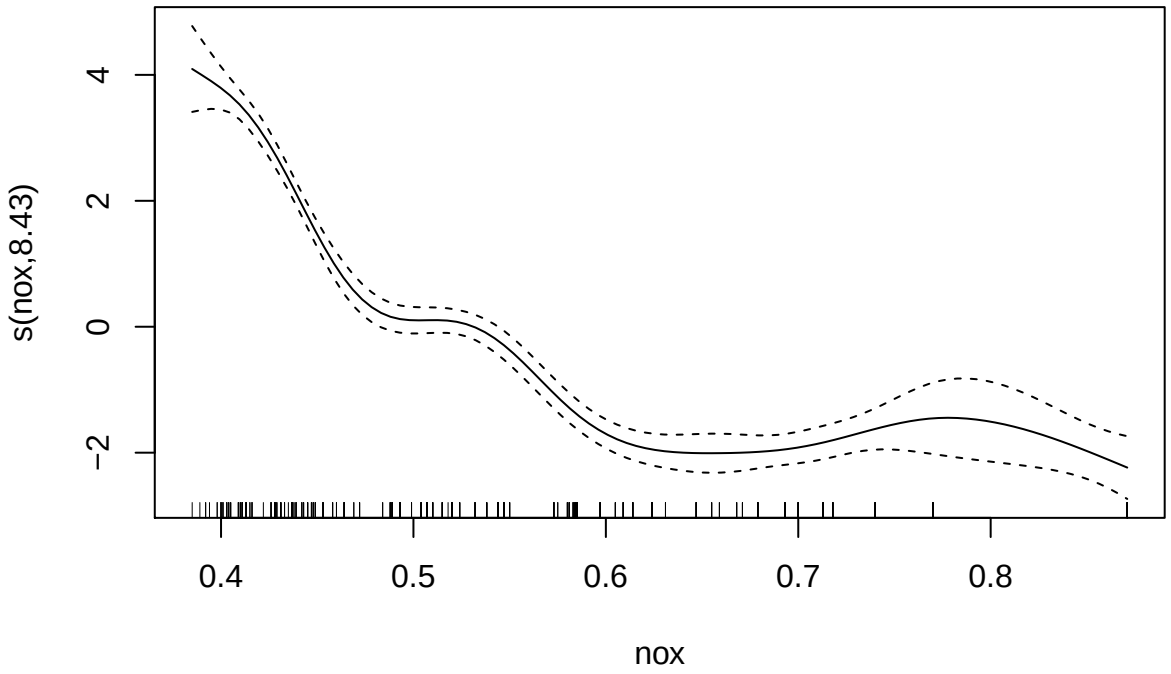
```r
# ? bam
```

Visualize Generalized Addictive Model

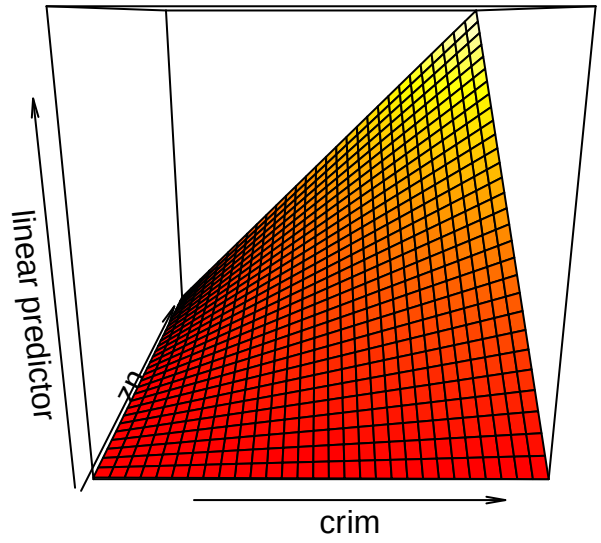
```r
plot(nox, dis)
x <- seq(0, 1, length = 500)
y <- predict(fit, data.frame(nox = x))
lines(x, y, col = "red", lwd = 2)
```



```
plot(fit)
```

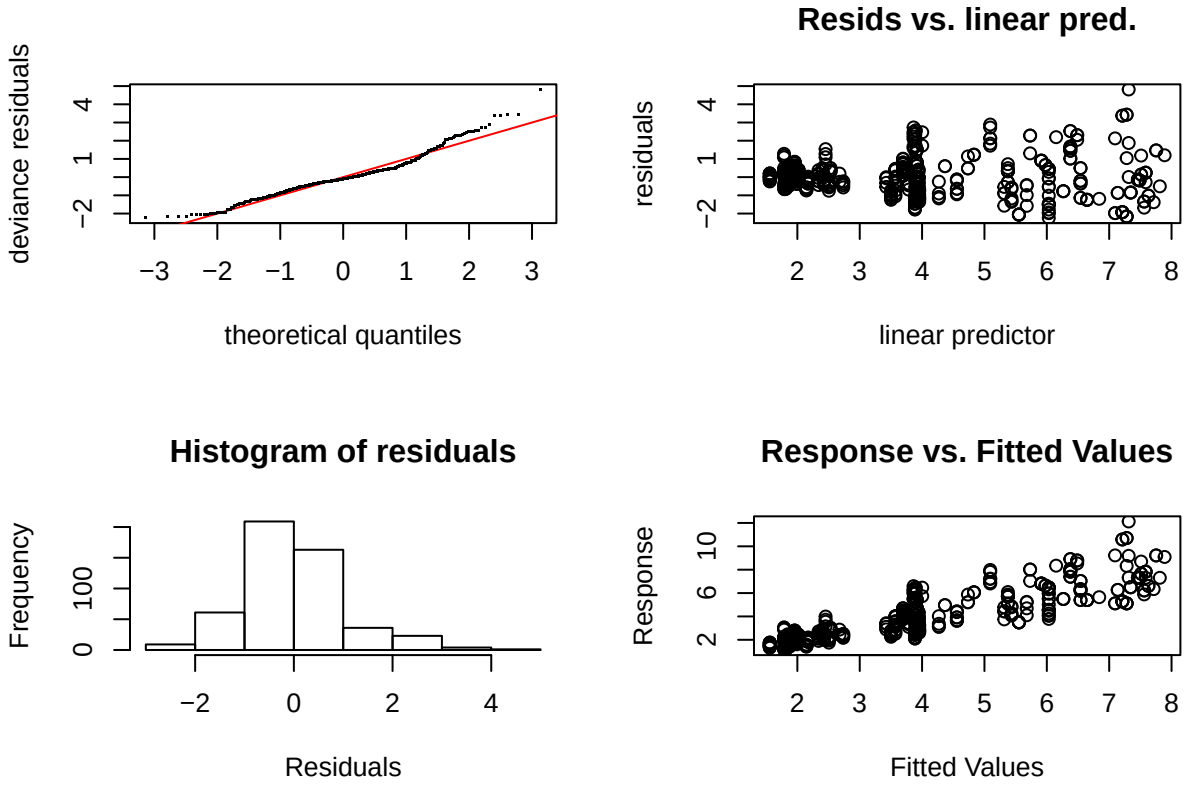



```
fit2 = gam(medv ~ crim + zn + crim:zn, data = Boston)
vis.gam(fit2)
```



3.1 Diagnostic of Generalized Addictive Model Advanced

```
gam.check(fit)
```




```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 7 iterations.
## The RMS GCV score gradient at convergence was 8.79622e-06 .
## The Hessian was positive definite.
## Model rank =  10 / 10
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##           k'   edf k-index p-value
## s(nox) 9.000 8.434   0.397       0
```

```
# ?gam.check ?choose.k
```

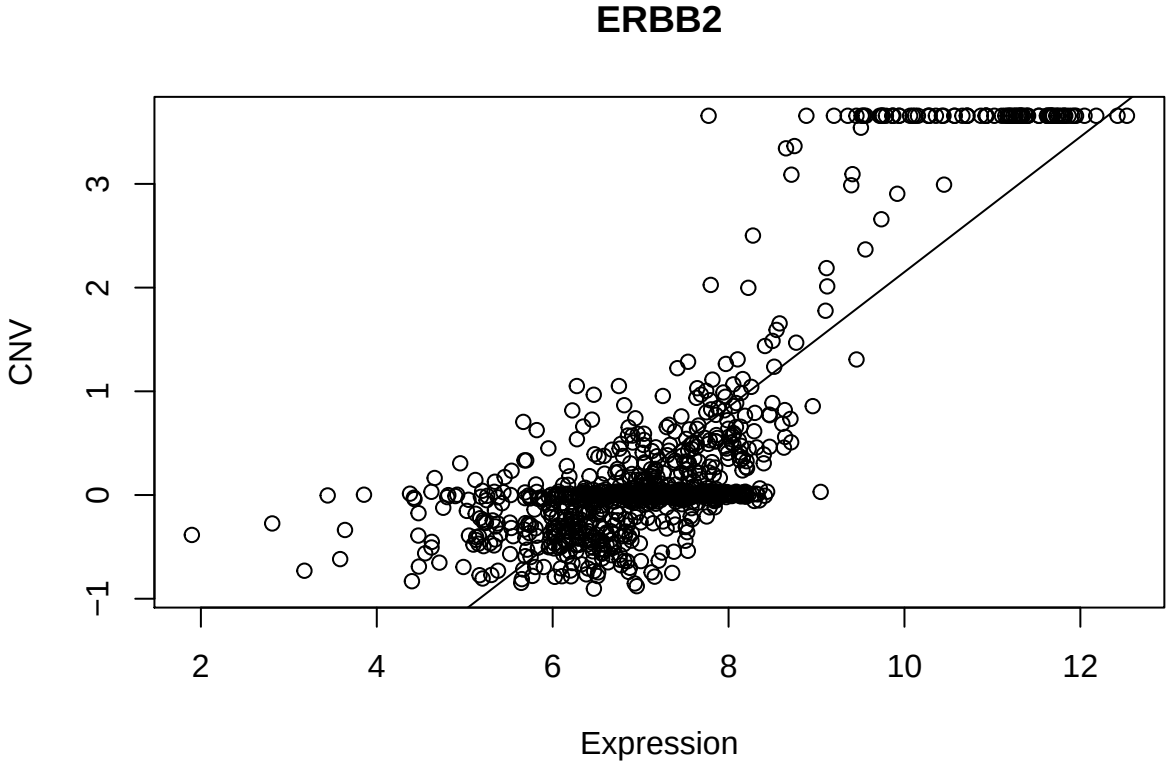
4 Exercise 1

4.1 TCGA_BRCA_CNV_processed.txt, TCGA_BRCA_Expr_processed.txt 파일을 읽어들이고 head함수를 이용해서 읽어들인 데이터의 첫 5행과 5열을 출력하세요.

```
##                 ABL2_CN ACVR1B_CN AKT1_CN AKT2_CN AKT3_CN
## TCGA.A2.A0CK.01   0.683    -0.001   0.000   0.005   0.699
## TCGA.BH.A1FE.01  -0.006     0.019  -0.381  -0.203   0.008
## TCGA.BH.A0BJ.01   0.840    -0.003  -0.320   0.237   0.840
## TCGA.AQ.A04J.01   0.384    -0.841  -0.415   0.008   0.353
## TCGA.AR.A252.01   0.130    -0.112   0.007  -0.122   0.130
```

```
##                 ABL2_Expr ACVR1B_Expr AKT1_Expr AKT2_Expr AKT3_Expr
## TCGA.A2.A0CK.01  2.973852    4.789937  7.098917  5.662665  5.115737
## TCGA.BH.A1FE.01  4.204047    4.840992  6.254114  5.224286  5.217036
## TCGA.BH.A0BJ.01  3.623733    4.732377  6.820766  5.970395  4.510637
## TCGA.AQ.A04J.01  3.561579    2.597737  6.405937  5.490886  4.406105
## TCGA.AR.A252.01  3.702411    4.423491  7.147624  5.495835  5.032941
```

4.2 ERBB2 유전자의 expression과 copy number를 각각 x축과 y축으로하는 scatter plot을 그리고 linear model을 만들어 추세선을 그리세요.

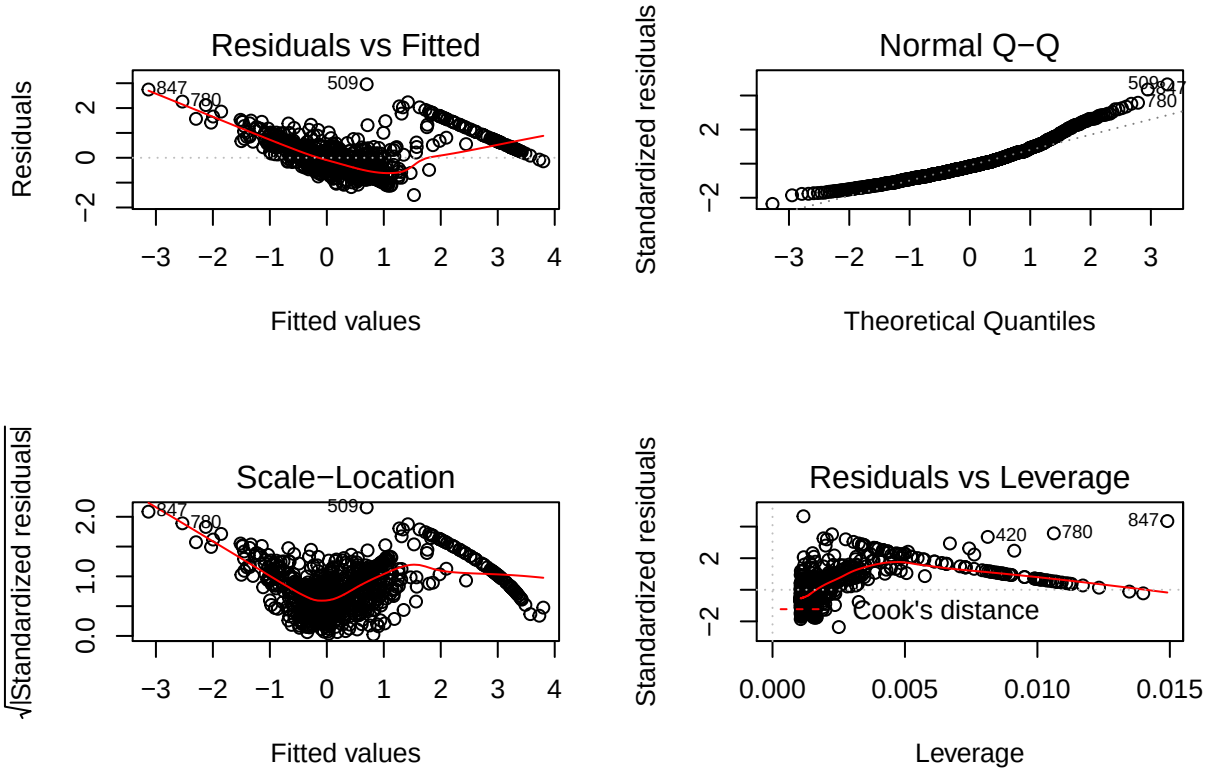


4.3 linear model의 summary를 출력하세요.

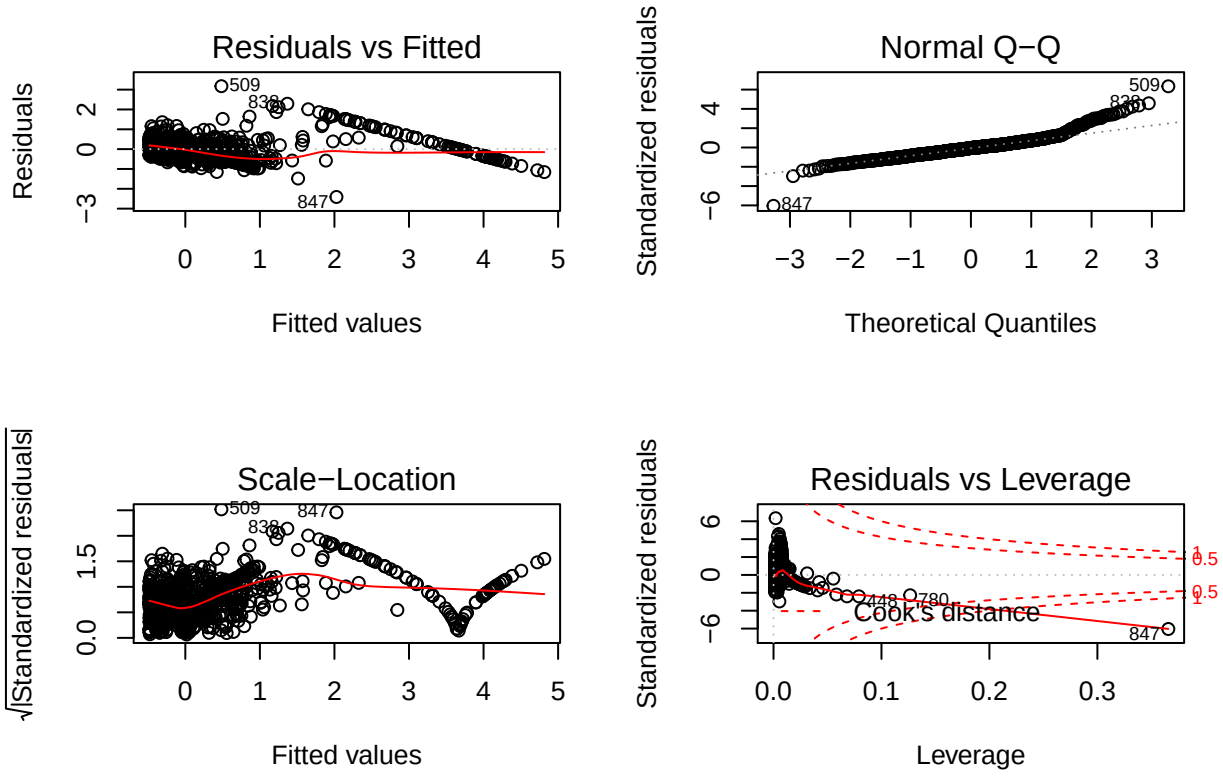
```
##
## Call:
## lm(formula = brca.cnv$ERBB2_CN ~ brca.expr$ERBB2_Expr)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.4998 -0.4456 -0.1032  0.3235  2.9580
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -4.36739    0.10319  -42.32   <2e-16 ***
## brca.expr$ERBB2_Expr  0.65181    0.01384   47.09   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.6361 on 938 degrees of freedom
## Multiple R-squared:  0.7028, Adjusted R-squared:  0.7025
## F-statistic:  2218 on 1 and 938 DF,  p-value: < 2.2e-16
```

4.4 plot 함수를 이용하여 모델을 평가하는 plot 4개를 그리세요.



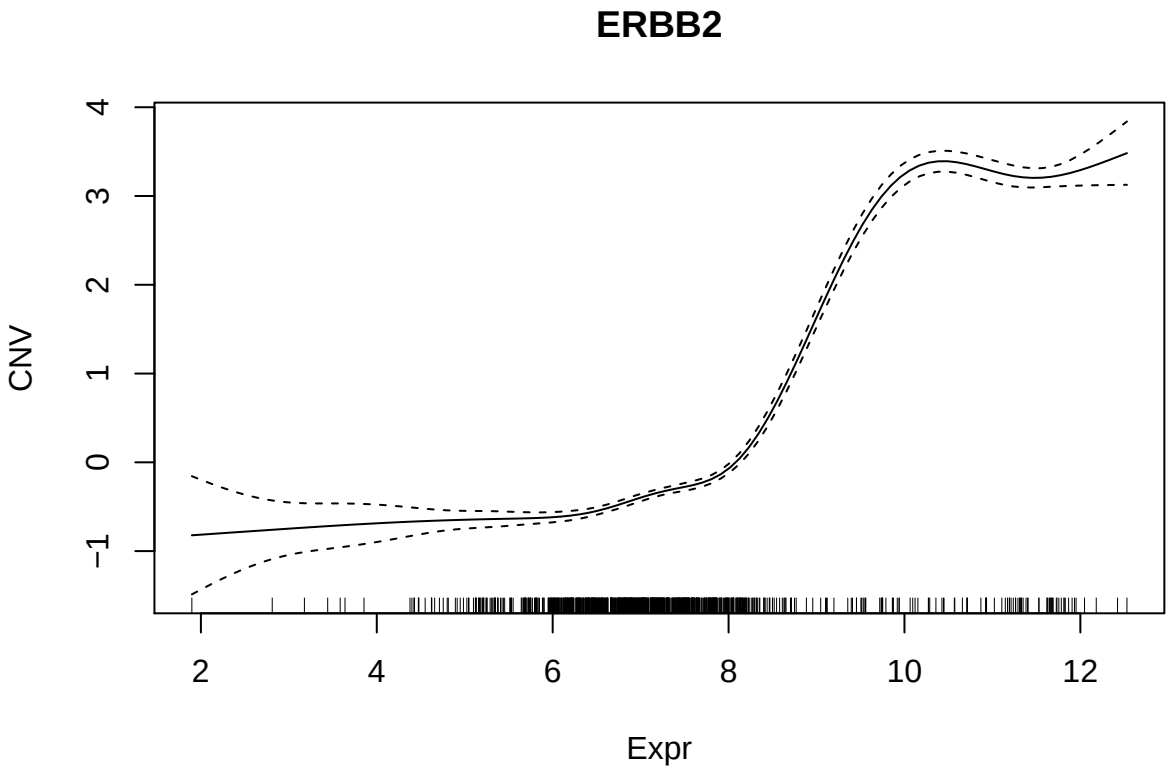
4.5 polynomial regression을 이용해서 model을 만들고 평가 plot을 그리세요



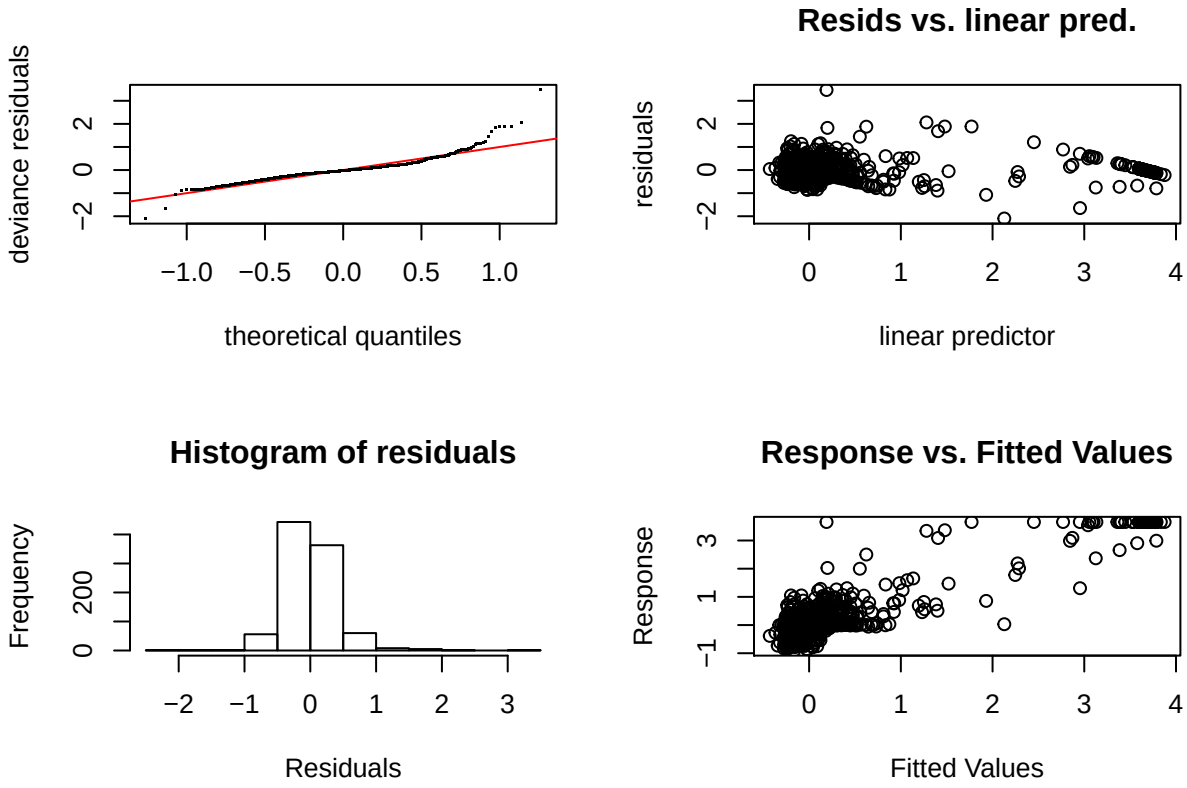
4.6 Generalized additive model을 만들고 summary를 출력하세요. Advanced

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## CN ~ s(Expr)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.39329    0.01257   31.29   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##           edf Ref.df     F p-value
## s(Expr) 8.731   8.98 852.6  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.891   Deviance explained = 89.2%
## GCV = 0.15009  Scale est. = 0.14854   n = 940
```

4.7 아래와 같이 모델의 추세곡선을 그리세요. Advanced



4.8 모델의 평가 plot을 그리세요. Advanced




```
##
## Method: GCV   Optimizer: magic
## Smoothing parameter selection converged after 7 iterations.
## The RMS GCV score gradient at convergence was 1.595668e-06 .
## The Hessian was positive definite.
## Model rank =  10 / 10
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##           k'  edf k-index p-value
## s(Expr) 9.00 8.73    1.03     0.8
```

5 Classification

5.1 Preparing the training and testing datasets

```
# install.packages('C50') library(C50)
load("../data/churn.RData")
# data(churn)
str(churnTrain)
```

```
## 'data.frame':    3333 obs. of  20 variables:
##  $ state                         : Factor w/ 51 levels "AK","AL","AR",..: 17 36 32 36 37 2 20 25 19 50
##  $ account_length                : int  128 107 137 84 75 118 121 147 117 141 ...
```

```
##  $ area_code                    : Factor w/ 3 levels "area_code_408",..: 2 2 2 1 2 3 3 2 1 2 ...
##  $ international_plan           : Factor w/ 2 levels "no","yes": 1 1 1 2 2 2 1 2 1 2 ...
##  $ voice_mail_plan              : Factor w/ 2 levels "no","yes": 2 2 1 1 1 1 2 1 1 2 ...
##  $ number_vmail_messages        : int  25 26 0 0 0 0 24 0 0 37 ...
##  $ total_day_minutes            : num  265 162 243 299 167 ...
##  $ total_day_calls              : int  110 123 114 71 113 98 88 79 97 84 ...
##  $ total_day_charge             : num  45.1 27.5 41.4 50.9 28.3 ...
##  $ total_eve_minutes            : num  197.4 195.5 121.2 61.9 148.3 ...
##  $ total_eve_calls              : int  99 103 110 88 122 101 108 94 80 111 ...
##  $ total_eve_charge             : num  16.78 16.62 10.3 5.26 12.61 ...
##  $ total_night_minutes          : num  245 254 163 197 187 ...
##  $ total_night_calls            : int  91 103 104 89 121 118 118 96 90 97 ...
##  $ total_night_charge           : num  11.01 11.45 7.32 8.86 8.41 ...
##  $ total_intl_minutes           : num  10 13.7 12.2 6.6 10.1 6.3 7.5 7.1 8.7 11.2 ...
##  $ total_intl_calls             : int  3 3 5 7 3 6 7 6 4 5 ...
##  $ total_intl_charge            : num  2.7 3.7 3.29 1.78 2.73 1.7 2.03 1.92 2.35 3.02 ...
##  $ number_customer_service_calls: int  1 1 0 2 3 0 3 0 1 0 ...
##  $ churn                        : Factor w/ 2 levels "yes","no": 2 2 2 2 2 2 2 2 2 2 ...
```

```
churnTrain = churnTrain[, !names(churnTrain) %in% c("state", "area_code", "account_length")]
set.seed(2)
ind <- sample(2, nrow(churnTrain), replace = TRUE, prob = c(0.7, 0.3))
trainset = churnTrain[ind == 1, ]
testset = churnTrain[ind == 2, ]
dim(trainset)
```

```
## [1] 2315   17
```

```
dim(testset)
```

```
## [1] 1018   17
```

```
split.data = function(data, p = 0.7, s = 666) {
    set.seed(s)
    index = sample(1:dim(data)[1])
    train = data[index[1:floor(dim(data)[1] * p)], ]
    test = data[index[((ceiling(dim(data)[1] * p)) + 1):dim(data)[1]], ]
    return(list(train = train, test = test))
}
```

5.2 Recursive partitioning trees

```
library(rpart)
churn.rp <- rpart(churn ~ ., data = trainset)
churn.rp
```

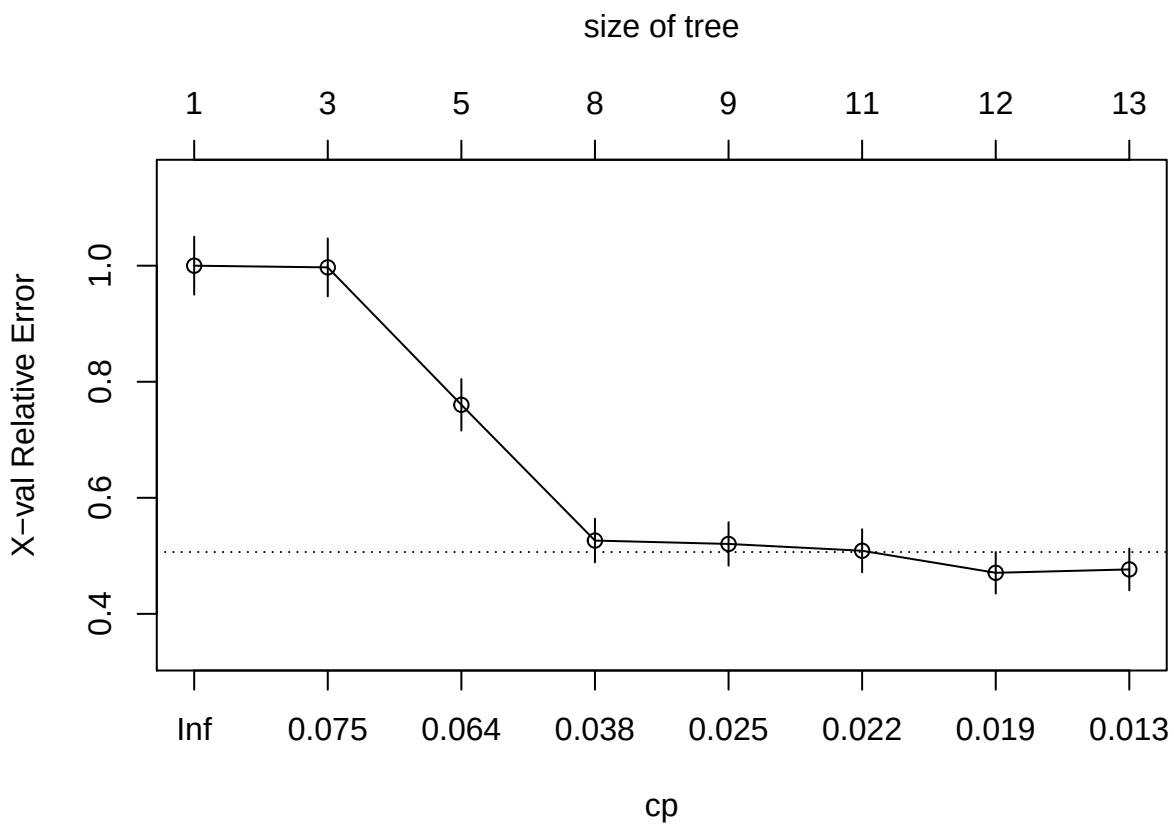
```
## n= 2315
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 2315 342 no (0.14773218 0.85226782)
##    2) total_day_minutes>=265.45 144  59 yes (0.59027778 0.40972222)
##      4) voice_mail_plan=no 110  29 yes (0.73636364 0.26363636)
##        8) total_eve_minutes>=188.5 67   3 yes (0.95522388 0.04477612) *
```

```
##        9) total_eve_minutes< 188.5 43  17 no (0.39534884 0.60465116)
##         18) total_day_minutes>=282.7 19   6 yes (0.68421053 0.31578947) *
##         19) total_day_minutes< 282.7 24   4 no (0.16666667 0.83333333) *
##      5) voice_mail_plan=yes 34   4 no (0.11764706 0.88235294) *
##    3) total_day_minutes< 265.45 2171 257 no (0.11837863 0.88162137)
##      6) number_customer_service_calls>=3.5 168  82 yes (0.51190476 0.48809524)
##       12) total_day_minutes< 160.2 71  10 yes (0.85915493 0.14084507) *
##       13) total_day_minutes>=160.2 97  25 no (0.25773196 0.74226804)
##         26) total_eve_minutes< 155.5 20   7 yes (0.65000000 0.35000000) *
##         27) total_eve_minutes>=155.5 77  12 no (0.15584416 0.84415584) *
##      7) number_customer_service_calls< 3.5 2003 171 no (0.08537194 0.91462806)
##       14) international_plan=yes 188  76 no (0.40425532 0.59574468)
##         28) total_intl_calls< 2.5 38   0 yes (1.00000000 0.00000000) *
##         29) total_intl_calls>=2.5 150  38 no (0.25333333 0.74666667)
##           58) total_intl_minutes>=13.1 32   0 yes (1.00000000 0.00000000) *
##           59) total_intl_minutes< 13.1 118   6 no (0.05084746 0.94915254) *
##       15) international_plan=no 1815  95 no (0.05234160 0.94765840)
##         30) total_day_minutes>=224.15 251  50 no (0.19920319 0.80079681)
##           60) total_eve_minutes>=259.8 36  10 yes (0.72222222 0.27777778) *
##           61) total_eve_minutes< 259.8 215  24 no (0.11162791 0.88837209) *
##         31) total_day_minutes< 224.15 1564  45 no (0.02877238 0.97122762) *
```

```
printcp(churn.rp)
```

```
##
## Classification tree:
## rpart(formula = churn ~ ., data = trainset)
##
## Variables actually used in tree construction:
## [1] international_plan            number_customer_service_calls
## [3] total_day_minutes             total_eve_minutes
## [5] total_intl_calls              total_intl_minutes
## [7] voice_mail_plan
##
## Root node error: 342/2315 = 0.14773
##
## n= 2315
##
##         CP nsplit rel error  xerror     xstd
## 1 0.076023      0   1.00000 1.00000 0.049920
## 2 0.074561      2   0.84795 0.99708 0.049860
## 3 0.055556      4   0.69883 0.76023 0.044421
## 4 0.026316      7   0.49415 0.52632 0.037673
## 5 0.023392      8   0.46784 0.52047 0.037481
## 6 0.020468     10   0.42105 0.50877 0.037092
## 7 0.017544     11   0.40058 0.47076 0.035788
## 8 0.010000     12   0.38304 0.47661 0.035993
```

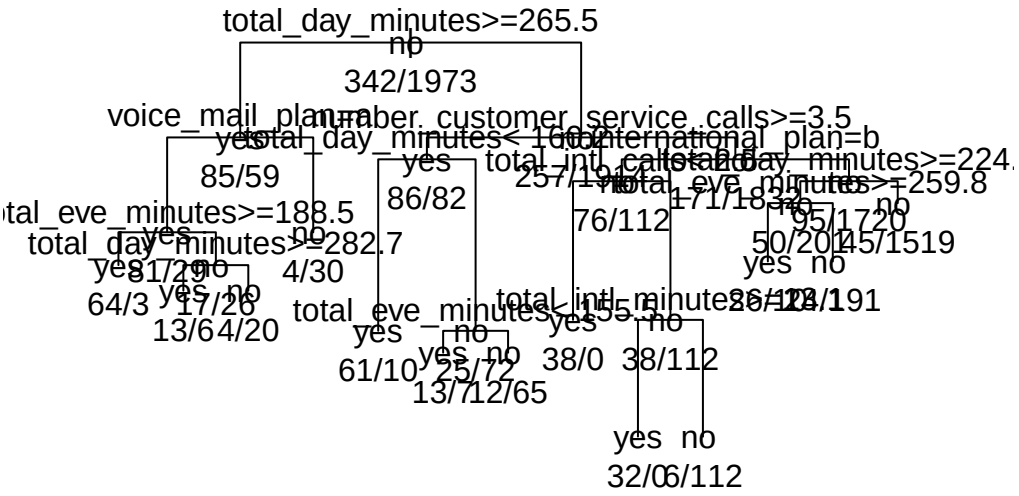
```
plotcp(churn.rp)
```



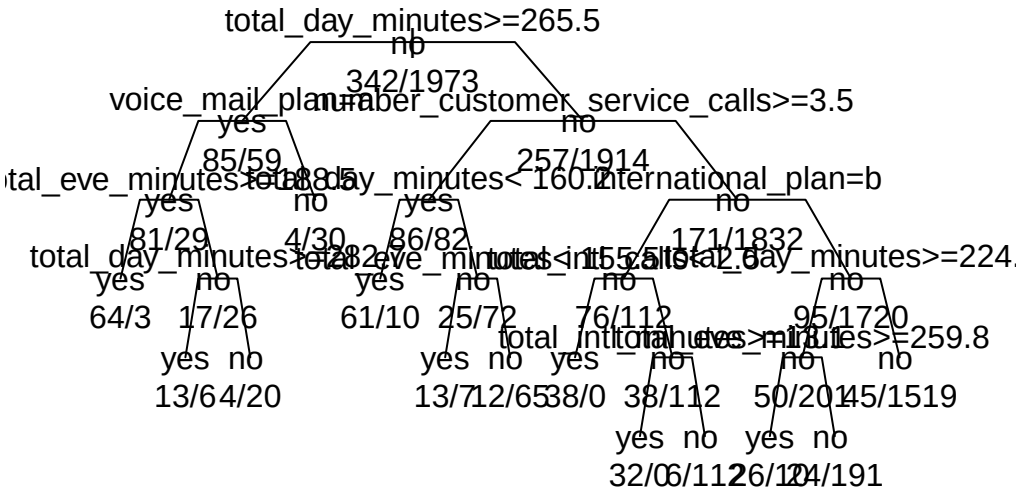

```
# summary(churn.rp)
```

“‘ ?rpart ?printcp ?summary.rpart

```
plot(churn.rp, margin = 0.1)
text(churn.rp, all = TRUE, use.n = TRUE)
```




```
plot(churn.rp, uniform = TRUE, branch = 0.6, margin = 0.1)
text(churn.rp, all = TRUE, use.n = TRUE)
```




```
predictions <- predict(churn.rp, testset, type = "class")
table(testset$churn, predictions)
```

```
##      predictions
##       yes  no
##   yes 100  41
##   no   18 859
```

```
# install.packages('caret')
library(caret)
```

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

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

```
confusionMatrix(table(predictions, testset$churn))
```

```
## Confusion Matrix and Statistics
##
##
## predictions yes  no
##         yes 100  18
##         no   41 859
##
##                Accuracy : 0.942
##                  95% CI : (0.9259, 0.9556)
##     No Information Rate : 0.8615
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7393
##  Mcnemar's Test P-Value : 0.004181
##
##             Sensitivity : 0.70922
##             Specificity : 0.97948
##          Pos Pred Value : 0.84746
##          Neg Pred Value : 0.95444
##              Prevalence : 0.13851
##          Detection Rate : 0.09823
##    Detection Prevalence : 0.11591
##       Balanced Accuracy : 0.84435
```

```
##
##        'Positive' Class : yes
##
```

```
min(churn.rp$cptable[, "xerror"])
```

```
## [1] 0.4707602
```

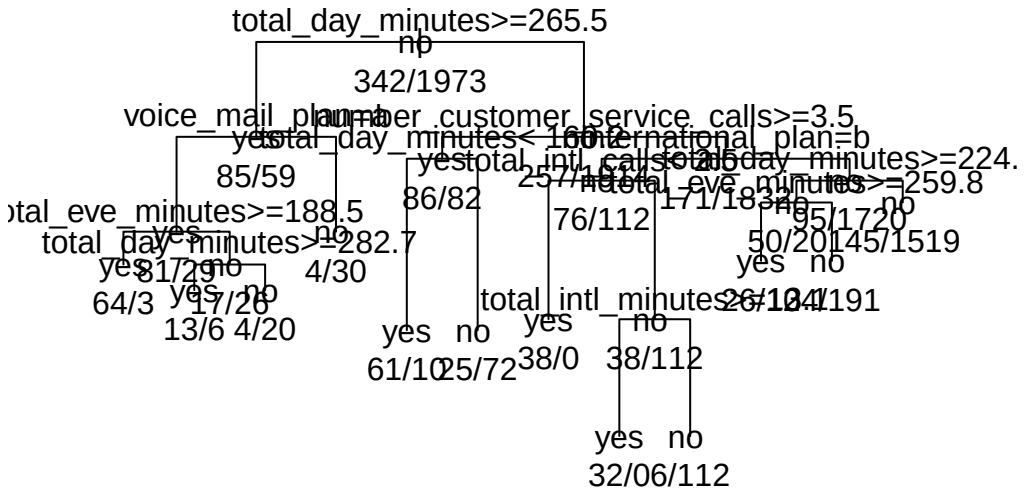
```
which.min(churn.rp$cptable[, "xerror"])
```

```
## 7
## 7
```

```
churn.cp = churn.rp$cptable[7, "CP"]
churn.cp
```

```
## [1] 0.01754386
```

```
prune.tree <- prune(churn.rp, cp = churn.cp)
plot(prune.tree, margin = 0.1)
text(prune.tree, all = TRUE, use.n = TRUE)
```




```
predictions <- predict(prune.tree, testset, type = "class")
table(testset$churn, predictions)
```

```
##      predictions
##       yes  no
##   yes  95  46
##   no   14 863
```

```
confusionMatrix(table(predictions, testset$churn))
```

```
## Confusion Matrix and Statistics
##
##
## predictions yes  no
##         yes  95  14
##         no   46 863
##
##                Accuracy : 0.9411
##                  95% CI : (0.9248, 0.9547)
##     No Information Rate : 0.8615
```

```
##     P-Value [Acc > NIR] : 2.786e-16
##
##                   Kappa : 0.727
##  Mcnemar's Test P-Value : 6.279e-05
##
##             Sensitivity : 0.67376
##             Specificity : 0.98404
##          Pos Pred Value : 0.87156
##          Neg Pred Value : 0.94939
##              Prevalence : 0.13851
##          Detection Rate : 0.09332
##    Detection Prevalence : 0.10707
##       Balanced Accuracy : 0.82890
##
##        'Positive' Class : yes
##
```

5.3 Classifying data with logistic regression

```
fit <- glm(churn ~ ., data = trainset, family = binomial)
summary(fit)
```

```
##
## Call:
## glm(formula = churn ~ ., family = binomial, data = trainset)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1519   0.1983   0.3460   0.5186   2.1284
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   8.3462866  0.8364914   9.978  < 2e-16 ***
## international_planyes        -2.0534243  0.1726694 -11.892  < 2e-16 ***
## voice_mail_planyes            1.3445887  0.6618905   2.031 0.042211 *
## number_vmail_messages        -0.0155101  0.0209220  -0.741 0.458496
## total_day_minutes             0.2398946  3.9168466   0.061 0.951163
## total_day_calls              -0.0014003  0.0032769  -0.427 0.669141
## total_day_charge             -1.4855284 23.0402950  -0.064 0.948592
## total_eve_minutes             0.3600678  1.9349825   0.186 0.852379
## total_eve_calls              -0.0028484  0.0033061  -0.862 0.388928
## total_eve_charge             -4.3204432 22.7644698  -0.190 0.849475
## total_night_minutes           0.4431210  1.0478105   0.423 0.672367
## total_night_calls             0.0003978  0.0033188   0.120 0.904588
## total_night_charge           -9.9162795 23.2836376  -0.426 0.670188
## total_intl_minutes            0.4587114  6.3524560   0.072 0.942435
## total_intl_calls              0.1065264  0.0304318   3.500 0.000464 ***
## total_intl_charge            -2.0803428 23.5262100  -0.088 0.929538
## number_customer_service_calls -0.5109077  0.0476289 -10.727  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 1938.8  on 2314  degrees of freedom
## Residual deviance: 1515.3  on 2298  degrees of freedom
## AIC: 1549.3
##
## Number of Fisher Scoring iterations: 6
```

```
pred = predict(fit, testset, type = "response")
Class = pred > 0.5
summary(Class)
```

```
##    Mode   FALSE    TRUE    NA's
## logical      58     960       0
```

```
tb = table(testset$churn, Class)
tb
```

```
##      Class
##       FALSE TRUE
##   yes    29  112
##   no     29  848
```

```
churn.mod = ifelse(testset$churn == "yes", 1, 0)
pred_class = churn.mod
pred_class[pred <= 0.5] = 1 - pred_class[pred <= 0.5]
ctb = table(churn.mod, pred_class)
ctb
```

```
##          pred_class
## churn.mod   0   1
##         0 848  29
##         1  29 112
```

```
confusionMatrix(ctb)
```

```
## Confusion Matrix and Statistics
##
##          pred_class
## churn.mod   0   1
##         0 848  29
##         1  29 112
##
##                Accuracy : 0.943
##                  95% CI : (0.927, 0.9565)
##     No Information Rate : 0.8615
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7613
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9669
##             Specificity : 0.7943
##          Pos Pred Value : 0.9669
##          Neg Pred Value : 0.7943
##              Prevalence : 0.8615
##          Detection Rate : 0.8330
##    Detection Prevalence : 0.8615
```

```
##       Balanced Accuracy : 0.8806
##
##        'Positive' Class : 0
##
```

5.4 Training neural network with neuralnet

```
data(iris)
ind <- sample(2, nrow(iris), replace = TRUE, prob = c(0.7, 0.3))
trainset = iris[ind == 1, ]
testset = iris[ind == 2, ]
head(trainset)
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 2           4.9         3.0          1.4         0.2  setosa
## 5           5.0         3.6          1.4         0.2  setosa
## 6           5.4         3.9          1.7         0.4  setosa
## 8           5.0         3.4          1.5         0.2  setosa
## 9           4.4         2.9          1.4         0.2  setosa
## 10          4.9         3.1          1.5         0.1  setosa
```

```
# install.packages('neuralnet')
library(neuralnet)
trainset$setosa = trainset$Species == "setosa"
trainset$virginica = trainset$Species == "virginica"
trainset$versicolor = trainset$Species == "versicolor"
head(trainset)
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species setosa
## 2           4.9         3.0          1.4         0.2  setosa   TRUE
## 5           5.0         3.6          1.4         0.2  setosa   TRUE
## 6           5.4         3.9          1.7         0.4  setosa   TRUE
## 8           5.0         3.4          1.5         0.2  setosa   TRUE
## 9           4.4         2.9          1.4         0.2  setosa   TRUE
## 10          4.9         3.1          1.5         0.1  setosa   TRUE
##    virginica versicolor
## 2      FALSE      FALSE
## 5      FALSE      FALSE
## 6      FALSE      FALSE
## 8      FALSE      FALSE
## 9      FALSE      FALSE
## 10     FALSE      FALSE
```

```
network = neuralnet(versicolor + virginica + setosa ~ Sepal.Length + Sepal.Width +
    Petal.Length + Petal.Width, trainset, hidden = 3)
# network
```

5.5 Visualizing neural network trained by neuralnet

```
plot(network)
par(mfrow = c(2, 2))
gwplot(network, selected.covariate = "Petal.Width")
gwplot(network, selected.covariate = "Sepal.Width")
```

```r
gwplot(network, selected.covariate = "Petal.Length")
gwplot(network, selected.covariate = "Petal.Width")
# ?gwplot
```

5.6 Predicting labels based upon a model trained by neuralnet

```r
predict = compute(network, testset[-5])$net.result
prediction = c("versicolor", "virginica", "setosa")[apply(predict, 1, which.max)]
predict.table = table(testset$Species, prediction)
predict.table
```

```
##             prediction
##              setosa versicolor virginica
##   setosa         15          0         0
##   versicolor      0         14         0
##   virginica       0          0        12
```

```r
confusionMatrix(predict.table)
```

```
## Confusion Matrix and Statistics
##
##             prediction
##              setosa versicolor virginica
##   setosa         15          0         0
##   versicolor      0         14         0
##   virginica       0          0        12
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9139562, 1)
##     No Information Rate : 0.3658537
##     P-Value [Acc > NIR] : < 0.00000000000000022204
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity              1.0000000         1.0000000        1.0000000
## Specificity              1.0000000         1.0000000        1.0000000
## Pos Pred Value           1.0000000         1.0000000        1.0000000
## Neg Pred Value           1.0000000         1.0000000        1.0000000
## Prevalence               0.3658537         0.3414634        0.2926829
## Detection Rate           0.3658537         0.3414634        0.2926829
## Detection Prevalence     0.3658537         0.3414634        0.2926829
## Balanced Accuracy        1.0000000         1.0000000        1.0000000
```

```r
# compute(network, testset[-5])
```

6 Unsupervised learning

6.1 Clustering Data With Hierarchical Clustering

```
customer = read.csv("../data/customer.csv", header = TRUE)
head(customer)
```

```
##   ID Visit.Time Average.Expense Sex Age
## 1  1          3             5.7   0  10
## 2  2          5            14.5   0  27
## 3  3         16            33.5   0  32
## 4  4          5            15.9   0  30
## 5  5         16            24.9   0  23
## 6  6          3            12.0   0  15
```

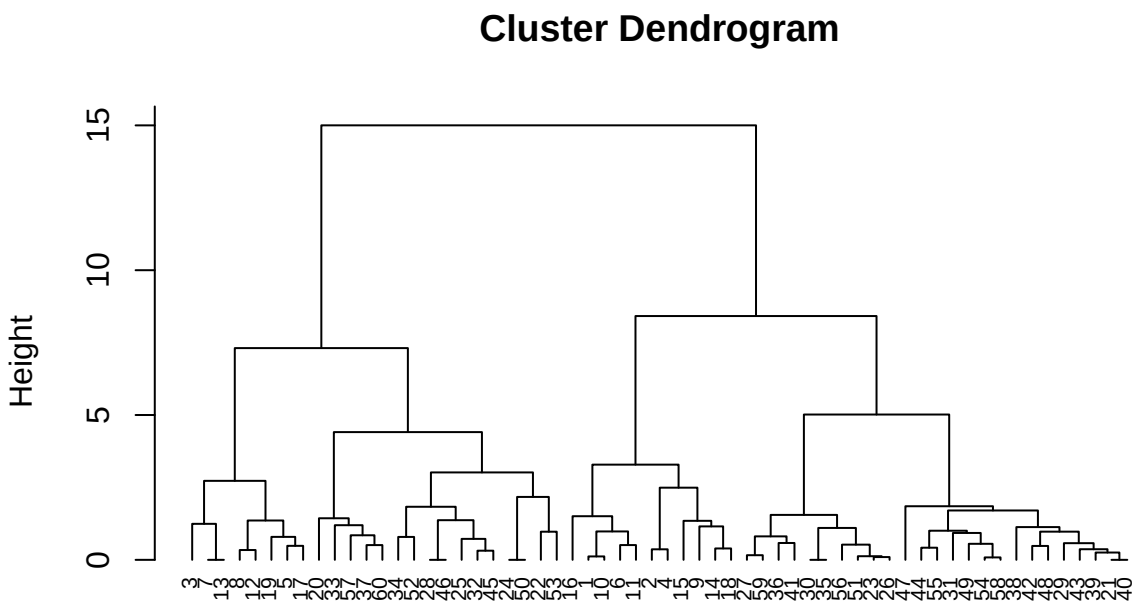
```
str(customer)
```

```
## 'data.frame':    60 obs. of  5 variables:
##  $ ID             : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Visit.Time     : int  3 5 16 5 16 3 12 14 6 3 ...
##  $ Average.Expense: num  5.7 14.5 33.5 15.9 24.9 12 28.5 18.8 23.8 5.3 ...
##  $ Sex            : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Age            : int  10 27 32 30 23 15 33 27 16 11 ...
```

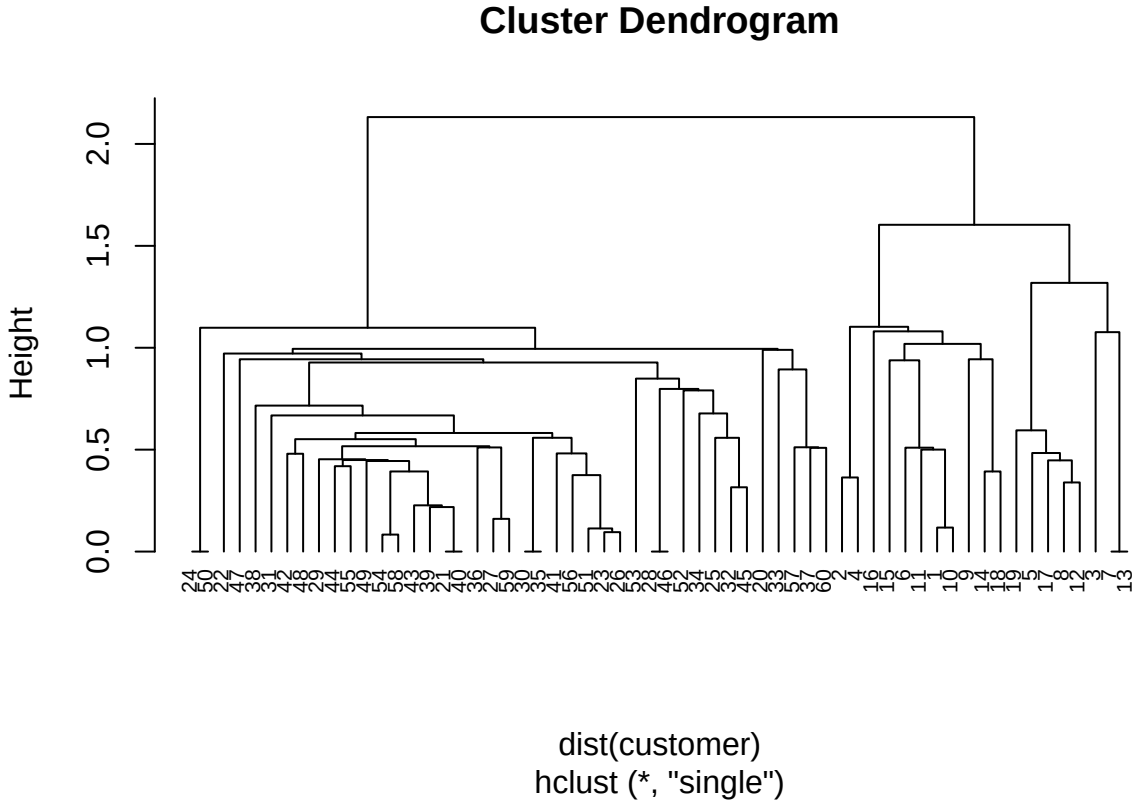
```
customer = scale(customer[, -1])
hc = hclust(dist(customer, method = "euclidean"), method = "ward.D2")
hc
```

```
##
## Call:
## hclust(d = dist(customer, method = "euclidean"), method = "ward.D2")
##
## Cluster method   : ward.D2
## Distance         : euclidean
## Number of objects: 60
```

```
plot(hc, hang = -0.01, cex = 0.7)
```




```
hc2 = hclust(dist(customer), method = "single")
plot(hc2, hang = -0.01, cex = 0.7)
```



```r
# ? diste ? hclust install.packages('dendextend')
# install.packages('margrittr')
library(dendextend)
```

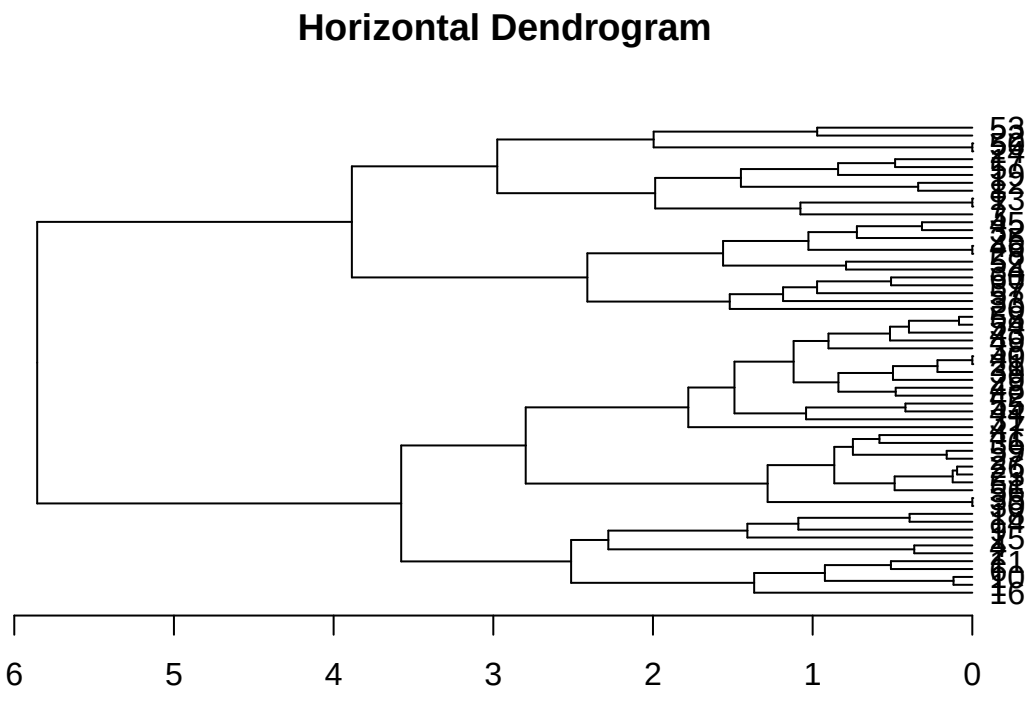
```
##
## ---------------------
## Welcome to dendextend version 1.8.0
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## Or contact: <tal.galili@gmail.com>
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------
```

```
##
## Attaching package: 'dendextend'
```

```
## The following object is masked from 'package:rpart':
##
##     prune
```

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

```r
library(magrittr)
dend = customer %>% dist %>% hclust %>% as.dendrogram
dend %>% plot(horiz = TRUE, main = "Horizontal Dendrogram")
```



6.2 Cutting Tree Into Clusters

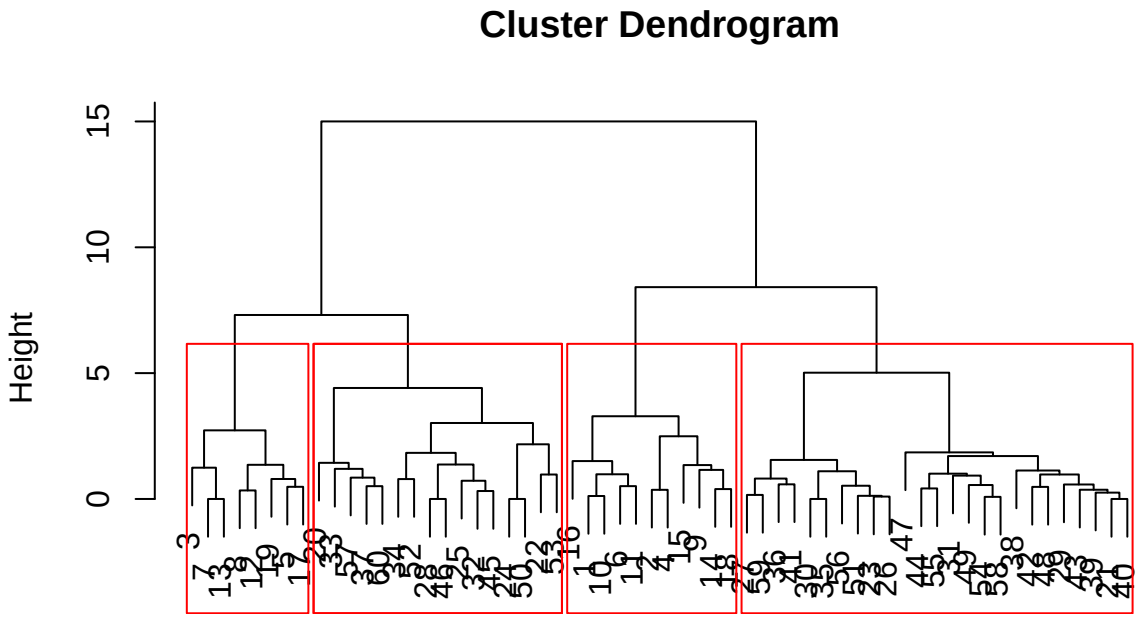
```
fit = cutree(hc, k = 4)
fit
```

```
##  [1] 1 1 2 1 2 1 2 2 1 1 1 2 2 1 1 1 2 1 2 3 4 3 4 3 3 4 4 3 4 4 4 3 3 3 4
## [36] 4 3 4 4 4 4 4 4 4 4 3 3 4 4 4 3 4 3 3 4 4 4 3 4 4 3
```

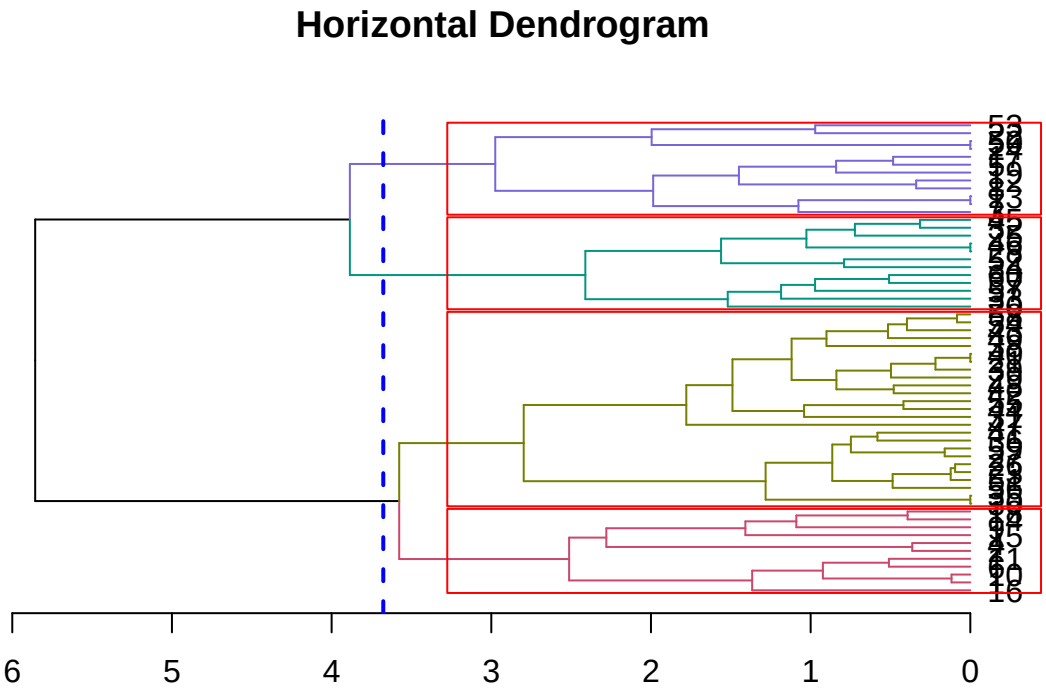
```
table(fit)
```

```
## fit
##  1  2  3  4
## 11  8 16 25
```

```
plot(hc)
rect.hclust(hc, k = 4, border = "red")
rect.hclust(hc, k = 4, which = 2, border = "red")
```




```
dend %>% color_branches(k = 4) %>% plot(horiz = TRUE, main = "Horizontal Dendrogram")
dend %>% rect.dendrogram(k = 4, horiz = TRUE)
abline(v = heights_per_k.dendrogram(dend)["4"] + 0.1, lwd = 2, lty = 2, col = "blue")
```

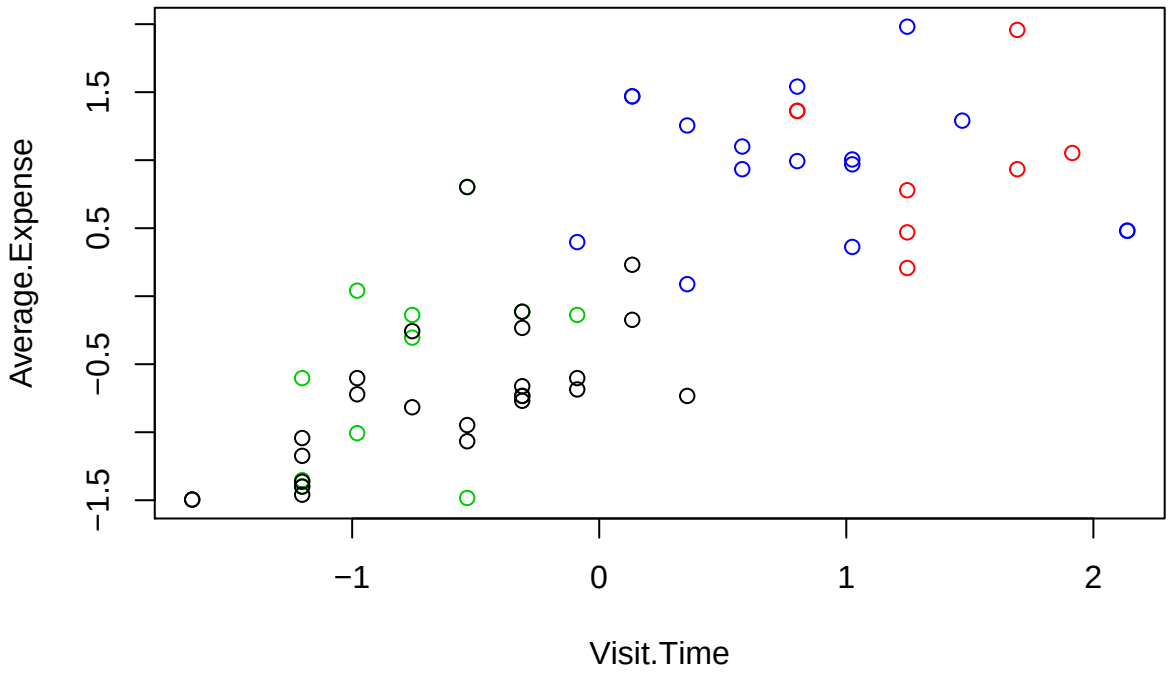


6.3 Clustering Data With Kmeans Method

```
fit = kmeans(customer, 4)
fit
```

```
## K-means clustering with 4 clusters of sizes 25, 8, 11, 16
##
## Cluster means:
##      Visit.Time Average.Expense           Sex           Age
## 1 -0.6322631734   -0.7299062815  0.6750489293 -0.6411604429
## 2  1.3302015708    1.0155226457 -1.4566845316  0.5591306774
## 3 -0.7771736830   -0.5178412319 -1.4566845316 -0.4774599043
## 4  0.8571173301    0.9887330889  0.6750489293  1.0505015376
##
## Clustering vector:
##  [1] 3 3 2 3 2 3 2 2 3 3 3 2 2 3 3 3 2 3 2 4 1 4 1 4 4 1 1 4 1 1 1 4 4 4 1
## [36] 1 4 1 1 1 1 1 1 1 4 4 1 1 1 4 1 4 4 1 1 1 4 1 1 4
##
## Within cluster sum of squares by cluster:
## [1] 20.89159492  5.90040041 11.97454479 22.58235962
##  (between_SS / total_SS =  74.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

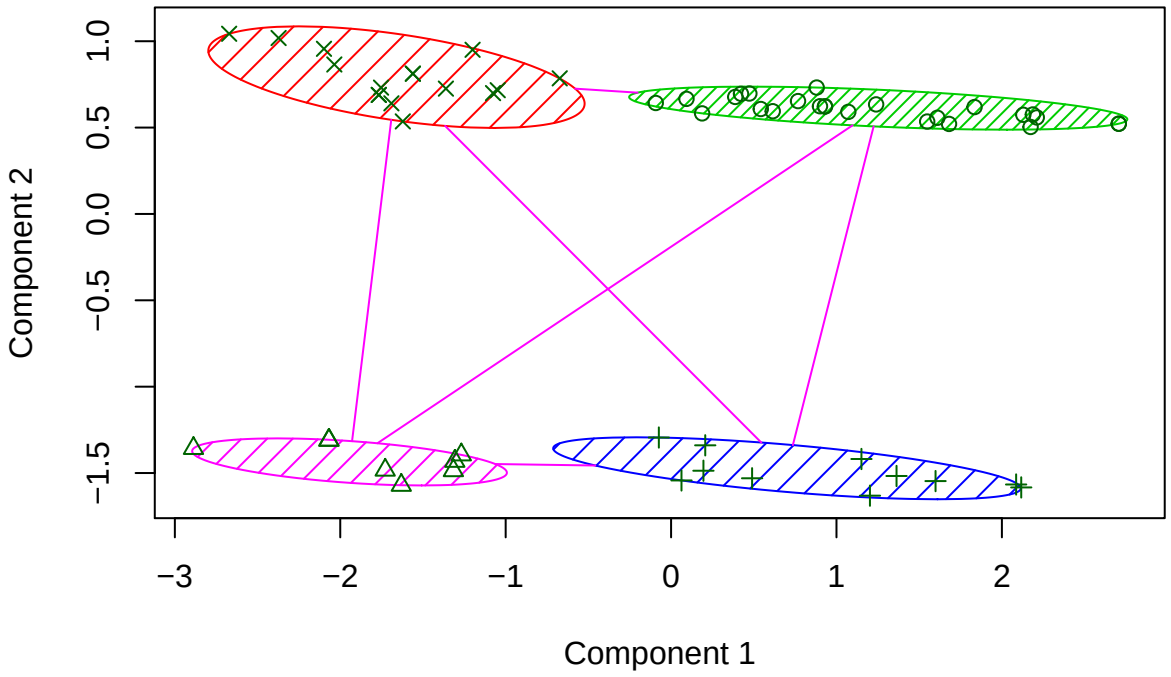
```
# barplot(t(fit$centers), beside = TRUE,xlab='cluster', ylab='value')
plot(customer, col = fit$cluster)
```



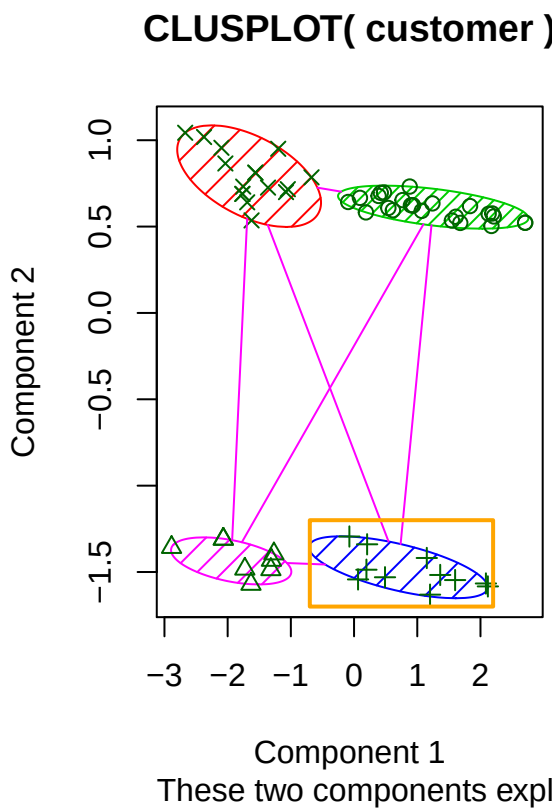

```
# help(kmeans)
```

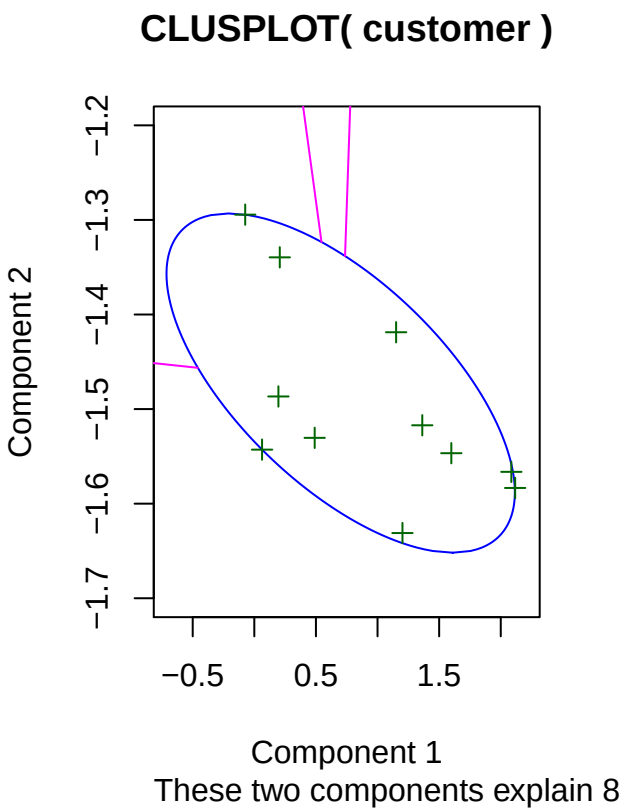
6.4 Drawing Bivariate Cluster Plot Advanced

```
# install.packages('cluster')
library(cluster)
clusplot(customer, fit$cluster, color = TRUE, shade = TRUE)
```

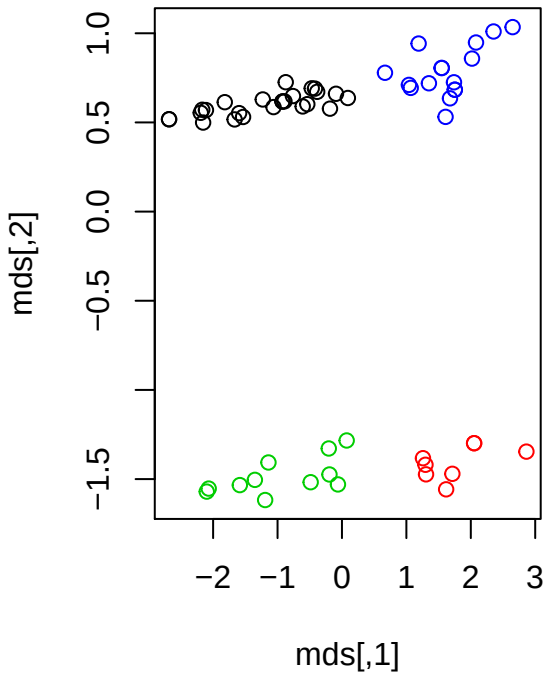



```
par(mfrow = c(1, 2))
clusplot(customer, fit$cluster, color = TRUE, shade = TRUE)
rect(-0.7, -1.7, 2.2, -1.2, border = "orange", lwd = 2)
clusplot(customer, fit$cluster, color = TRUE, xlim = c(-0.7, 2.2), ylim = c(-1.7,
    -1.2))
```



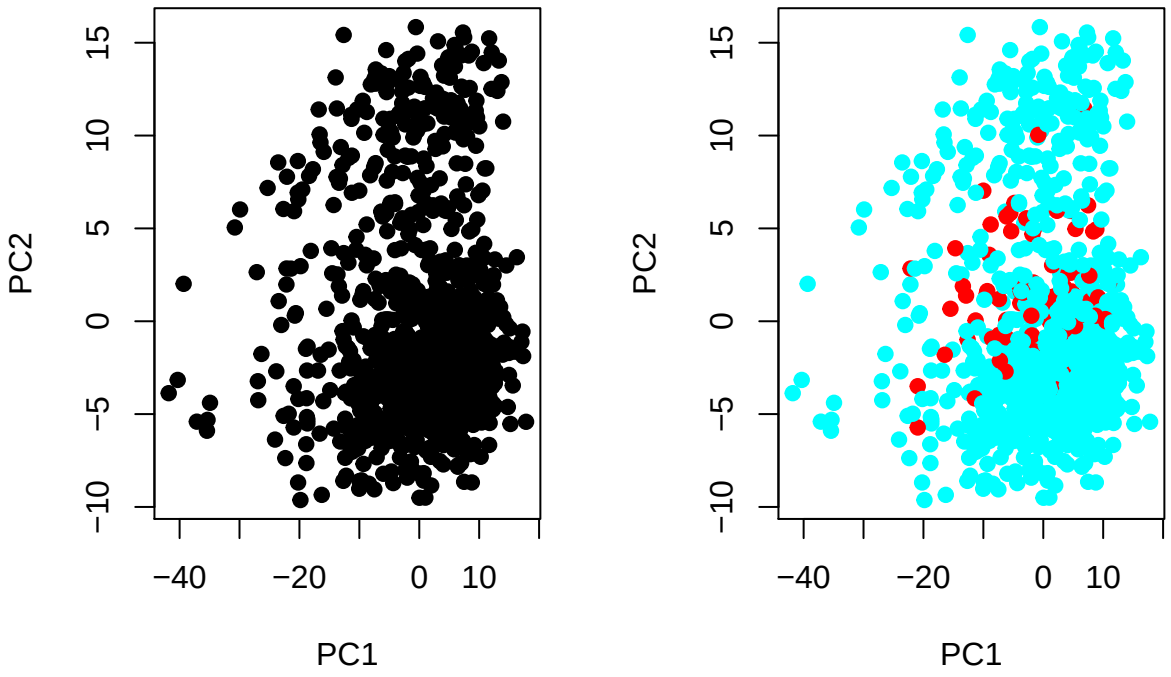



```
# help(cmdscale) help(princomp)
mds = cmdscale(dist(customer), k = 2)
plot(mds, col = fit$cluster)
```

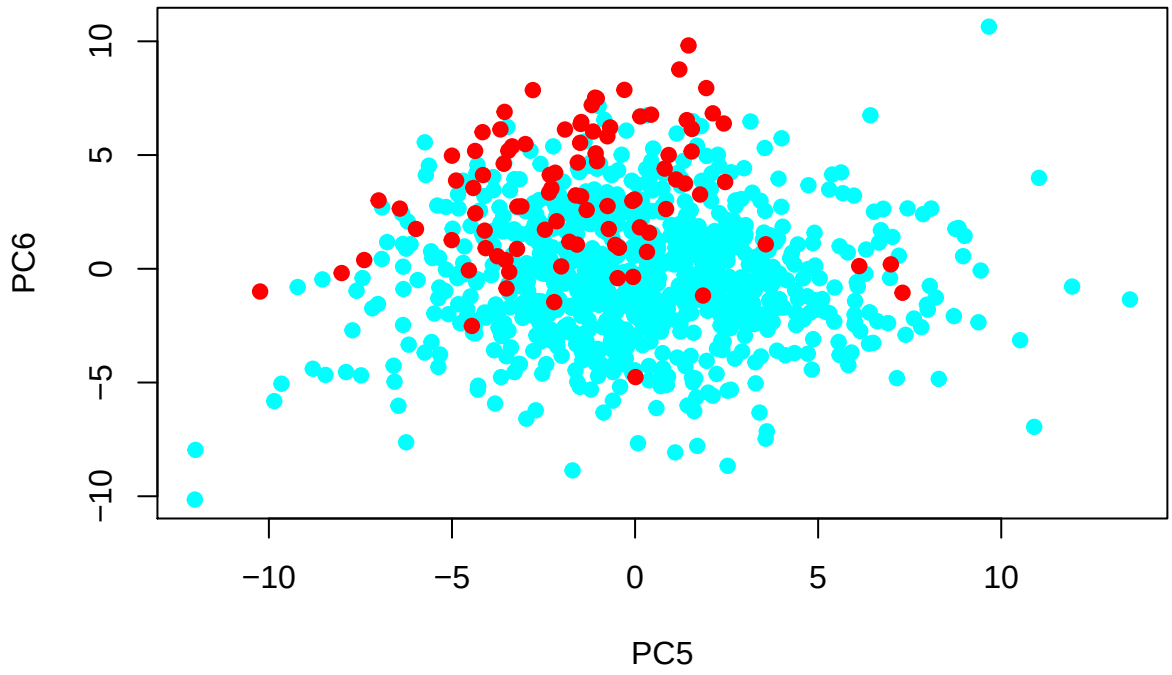


6.5 PCA

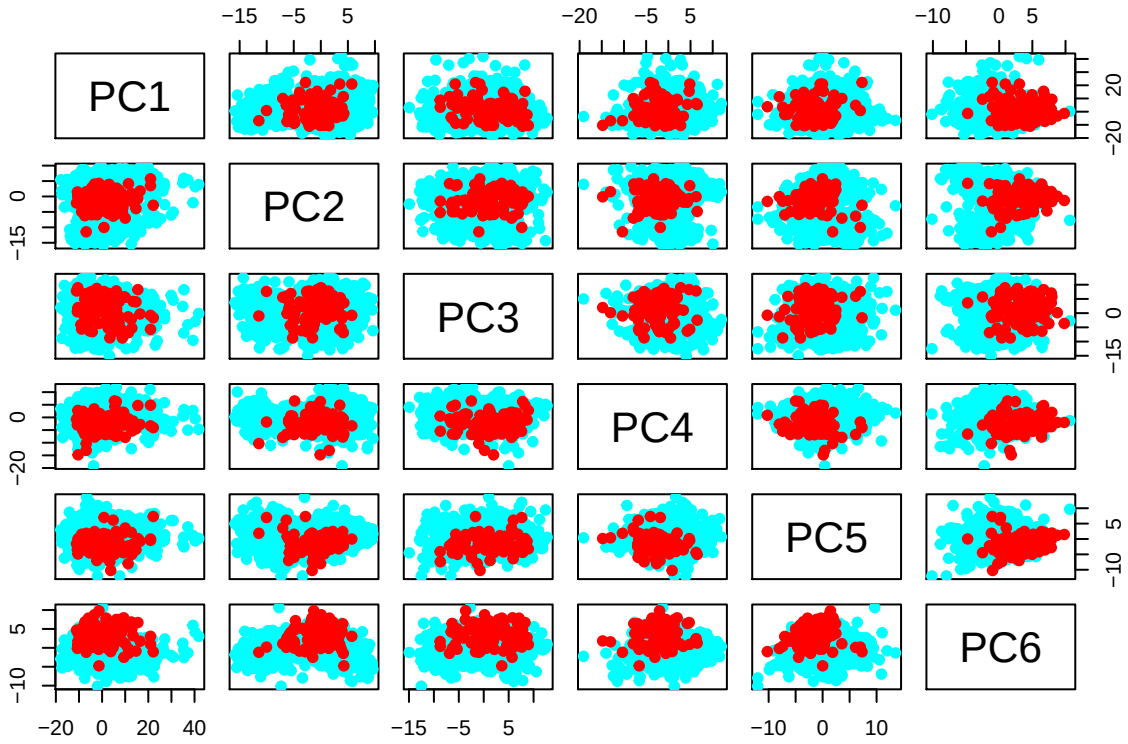
```
brca.cnv <- read.delim("../data/TCGA_BRCA_CNV_processed.txt")
brca.expr <- read.delim("../data/TCGA_BRCA_Expr_processed.txt")

pr.res <- prcomp(brca.expr, scale = TRUE)
par(mfrow = c(1, 2))
plot(pr.res$x[, c(1, 2)], pch = 19)
plot(pr.res$x[, c(1, 2)], pch = 19, col = ifelse(brca.cnv[, "ERBB2_CN"] > 3,
    rainbow(2)[1], rainbow(2)[2]))
```




```
par(mfrow = c(1, 1))

order.ERBB2.CNV <- order(brca.cnv[, "ERBB2_CN"])
brca.cnv <- brca.cnv[order.ERBB2.CNV, ]
brca.expr <- brca.expr[rownames(brca.cnv), ]

pr.res <- prcomp(brca.expr, scale = TRUE)
plot(pr.res$x[, c(5, 6)], pch = 19, col = ifelse(brca.cnv[, "ERBB2_CN"] > 3,
    rainbow(2)[1], rainbow(2)[2]))
```




```
pairs(pr.res$x[, 1:6], col = ifelse(brca.cnv[, "ERBB2_CN"] > 3, rainbow(2)[1],
    rainbow(2)[2]), pch = 19)
```



7 Exercise 2

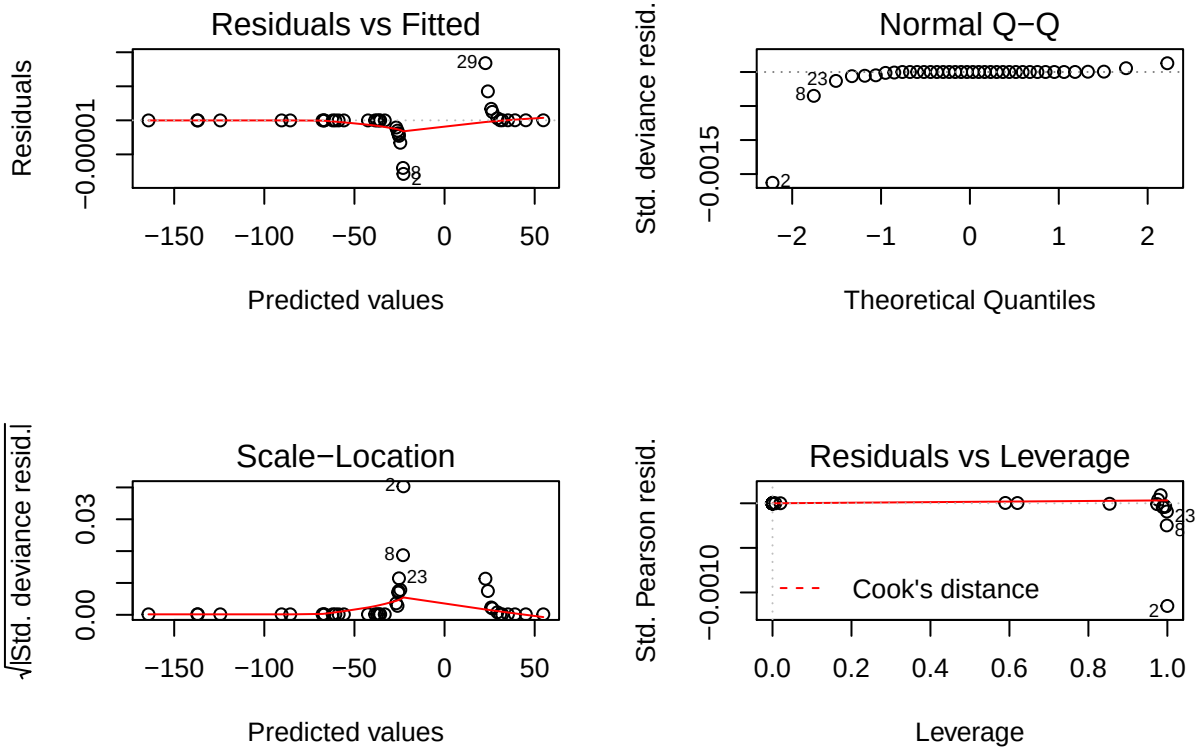
7.1 Data loading

```
## [1] TRUE
```

7.2 Training 데이터를 이용해서 top 10 predictor 도출하기 (t-test 이용)

7.3 Logistic regression 모델을 만들고 평가하기

t-test로 도출된 top 10 gene들만 predictor (feature)로 이용




```
## Confusion Matrix and Statistics
##
##      prediction
##       ALL AML
##   ALL  20   0
##   AML   4  10
##
##                Accuracy : 0.8823529
##                  95% CI : (0.7254965, 0.9669983)
##     No Information Rate : 0.7058824
##     P-Value [Acc > NIR] : 0.0139774
##
##                   Kappa : 0.7462687
##  Mcnemar's Test P-Value : 0.1336144
##
##             Sensitivity : 0.8333333
##             Specificity : 1.0000000
```

```
##          Pos Pred Value : 1.0000000
##          Neg Pred Value : 0.7142857
##              Prevalence : 0.7058824
##          Detection Rate : 0.5882353
##    Detection Prevalence : 0.5882353
##       Balanced Accuracy : 0.9166667
##
##        'Positive' Class : ALL
##
```

7.4 Neural network 모델을 만들고 평가하기

t-test로 도출된 top 10 gene들만 predictor (feature)로 이용

```
##
## Attaching package: 'nnet'
```

```
## The following object is masked from 'package:mgcv':
##
##     multinom
```

```
## # weights:  25
## initial  value 26.545167
## iter  10 value 4.013046
## iter  20 value 3.520613
## iter  30 value 3.519491
## iter  40 value 3.519395
## iter  50 value 0.555763
## iter  60 value 0.194574
## iter  70 value 0.161835
## iter  80 value 0.156446
## iter  90 value 0.156365
## iter 100 value 0.156300
## iter 110 value 0.156241
## iter 120 value 0.156188
## iter 130 value 0.156184
## iter 140 value 0.156133
## iter 150 value 0.126650
## iter 160 value 0.087183
## iter 170 value 0.077010
## iter 180 value 0.074983
## iter 190 value 0.074968
## iter 200 value 0.074966
## final  value 0.074966
## stopped after 200 iterations
```

```
##  [1] "ALL" "ALL" "ALL" "ALL" "ALL" "ALL" "ALL" "ALL" "ALL" "ALL" "ALL"
## [12] "AML" "AML" "ALL" "AML" "AML" "ALL" "ALL" "AML" "AML" "ALL" "AML"
## [23] "AML" "ALL" "AML" "AML" "AML" "ALL" "ALL" "ALL" "ALL" "ALL" "ALL"
## [34] "ALL"
```

```
## Confusion Matrix and Statistics
##
##      ar.predict
##       ALL AML
##   ALL  20   0
```

```
##   AML   3  11
##
##                Accuracy : 0.9117647
##                  95% CI : (0.7632247, 0.9814205)
##     No Information Rate : 0.6764706
##     P-Value [Acc > NIR] : 0.001354127
##
##                   Kappa : 0.8118081
##  Mcnemar's Test P-Value : 0.248213079
##
##             Sensitivity : 0.8695652
##             Specificity : 1.0000000
##          Pos Pred Value : 1.0000000
##          Neg Pred Value : 0.7857143
##              Prevalence : 0.6764706
##          Detection Rate : 0.5882353
##    Detection Prevalence : 0.5882353
##       Balanced Accuracy : 0.9347826
##
##        'Positive' Class : ALL
##
```

7.5 heatmap 함수를 이용해서 hierarchical clustering이 된 형태의 heat map을 그리세요.

```
## Loading required package: grid
```