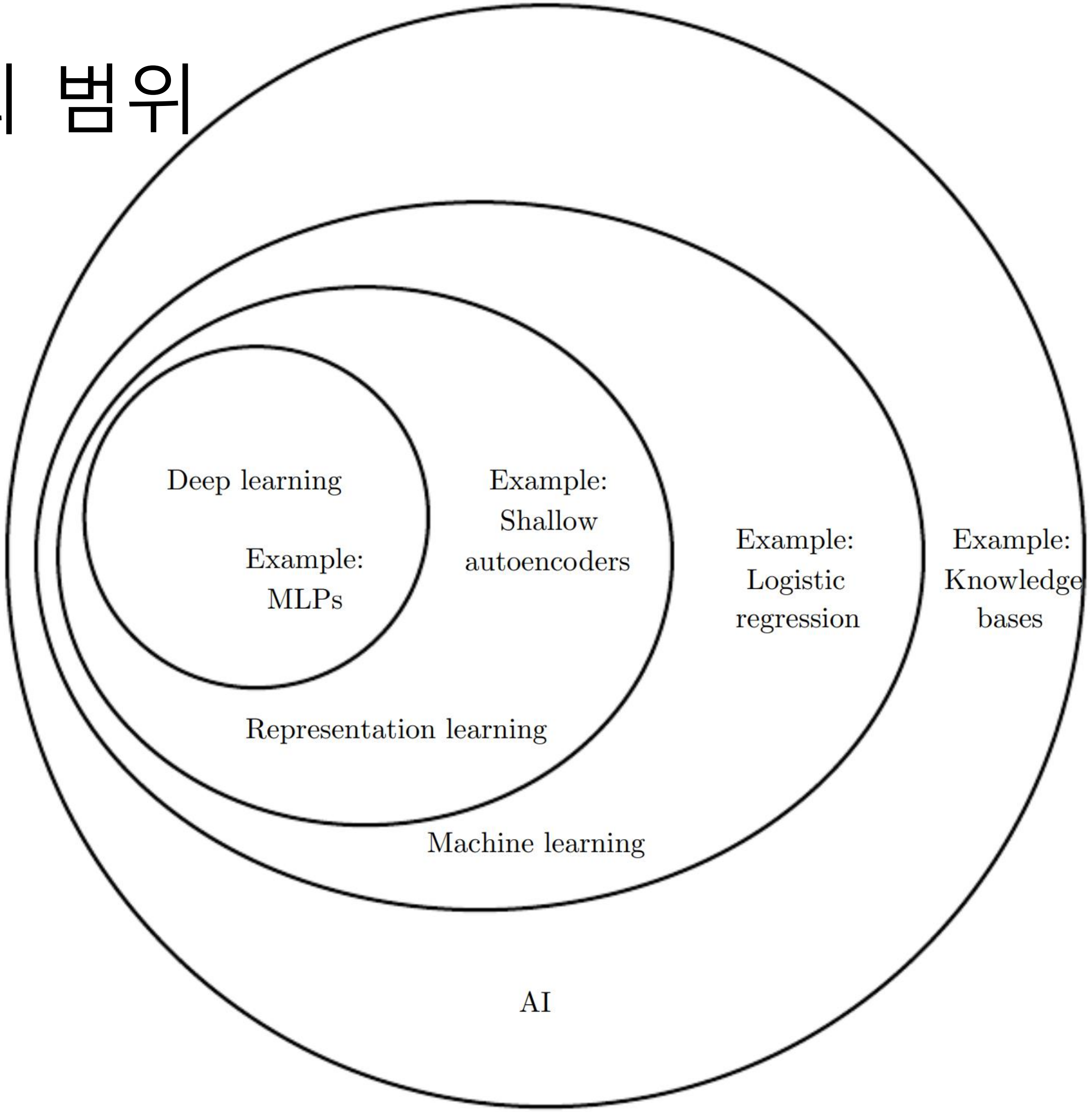
R을 활용한 의생명 기계학습 분석

기계학습의 범위

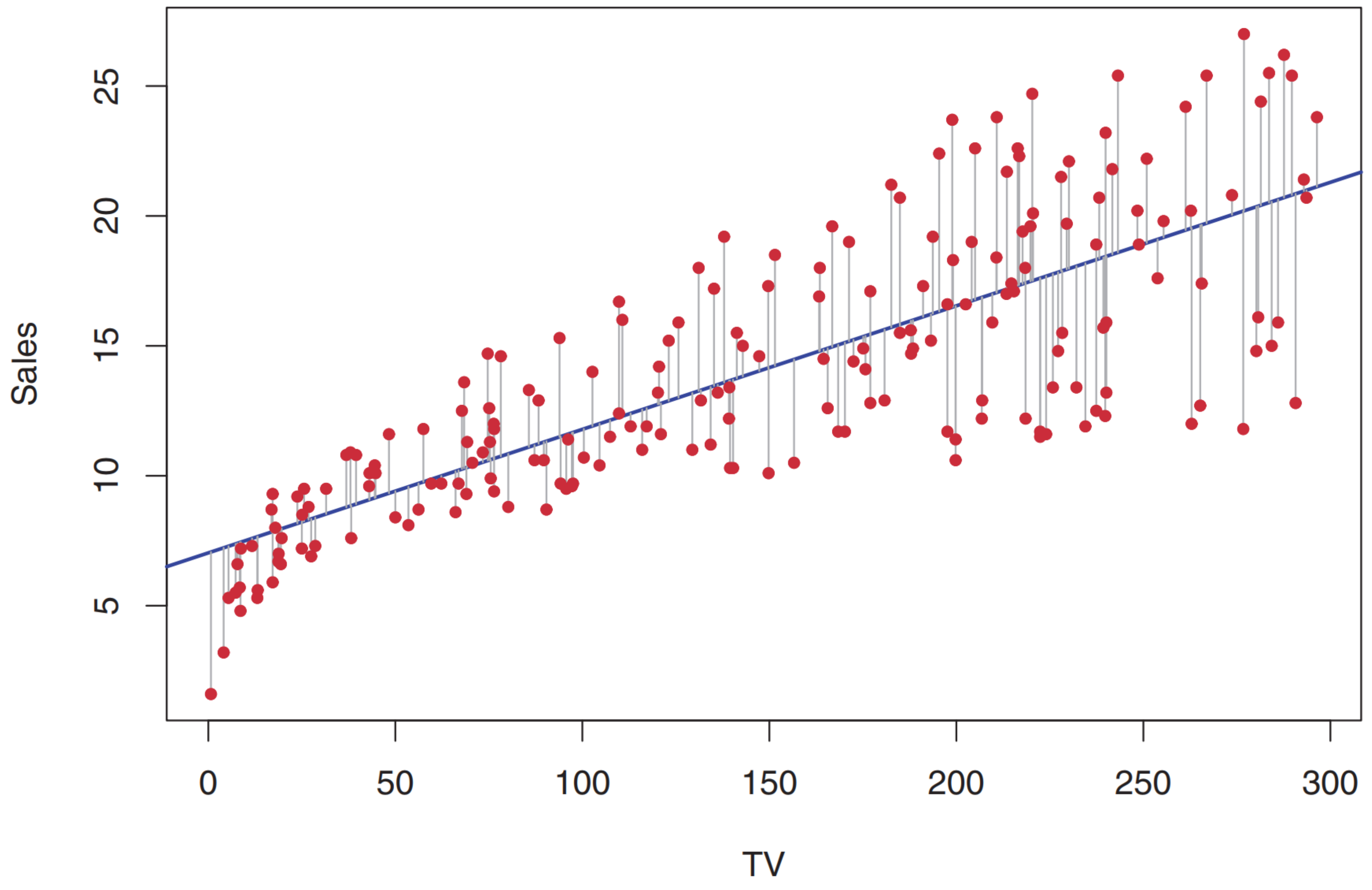

Deep learning
Example: MLPs
Example: Shallow autoencoders
Representation learning
Example: Logistic regression
Machine learning
Example: Knowledge bases
AI

학습 목표

- 자주 사용되는 기계학습 기법들을 R로 실행하는 방법을 익힌다. (주의! 기계학습의 자세한 원리를 배우는 것은 아님)
  - 자주 사용되는 기계학습 기법들이 무엇인지 안다.
  - 어떤 경우에 사용하는지 이해한다.
  - 결과에서 핵심적인 내용을 찾을 수 있다.
- 이 강의에서 다루는 기법들
  - Linear regression
  - Clustering (Hierarchical clustering, k-Means clustering)
  - Classifications (Decision tree, Logistic regression, Neural network)

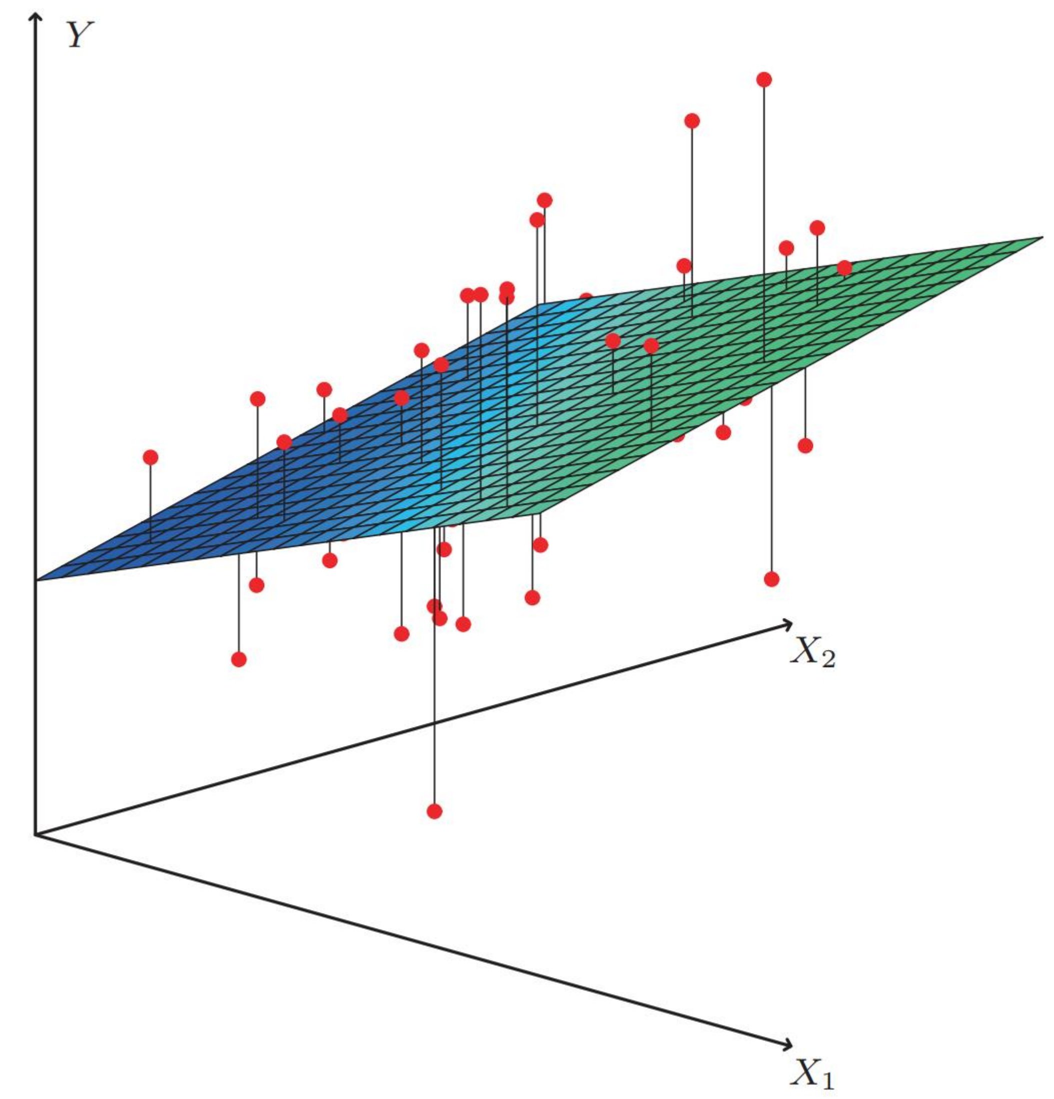
Regression

Linear regression

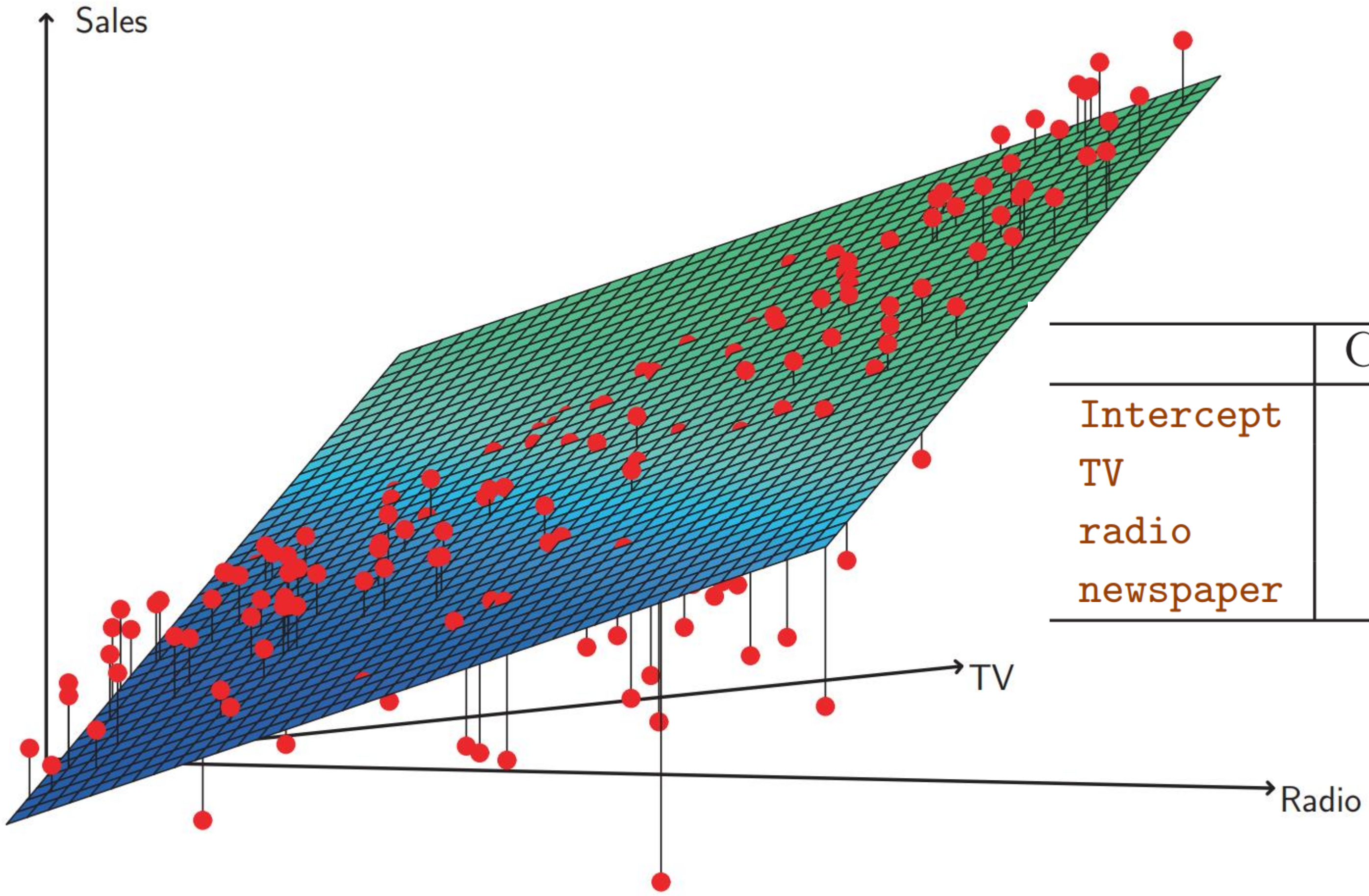


Introduction to statistical learning using R

Multiple linear regression



Coefficients

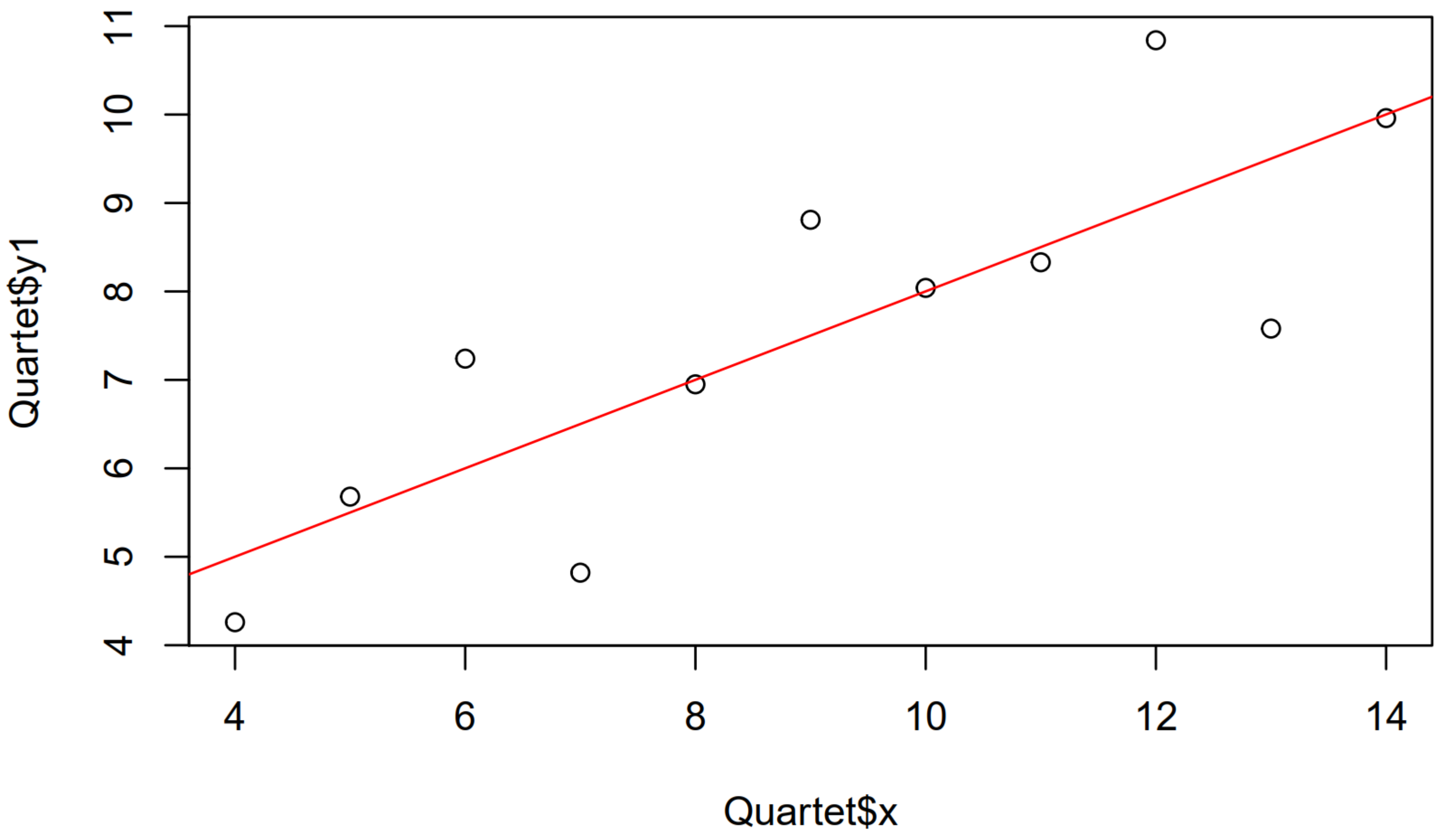


	Coefficient	Std. error	t-statistic	p-value
Intercept	2.939	0.3119	9.42	< 0.0001
TV	0.046	0.0014	32.81	< 0.0001
radio	0.189	0.0086	21.89	< 0.0001
newspaper	-0.001	0.0059	-0.18	0.8599

Simple Linear Regression with lm

```
library(car)
data(Quartet)
str(Quartet)

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



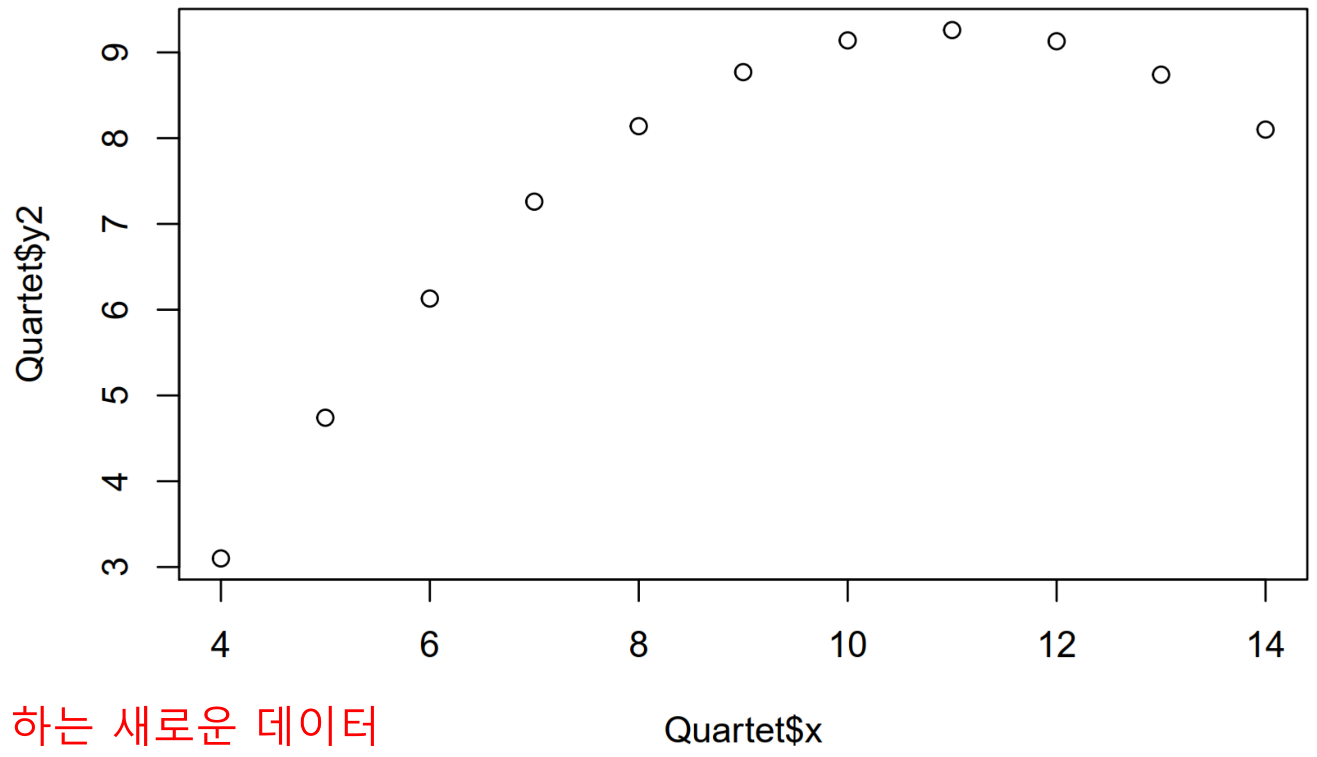
Result of linear regression

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
```

Using Linear Regression to Predict Unknown Values

```
plot(Quartet$x, Quartet$y2)
    # 데이터의 형태 살펴보기
```

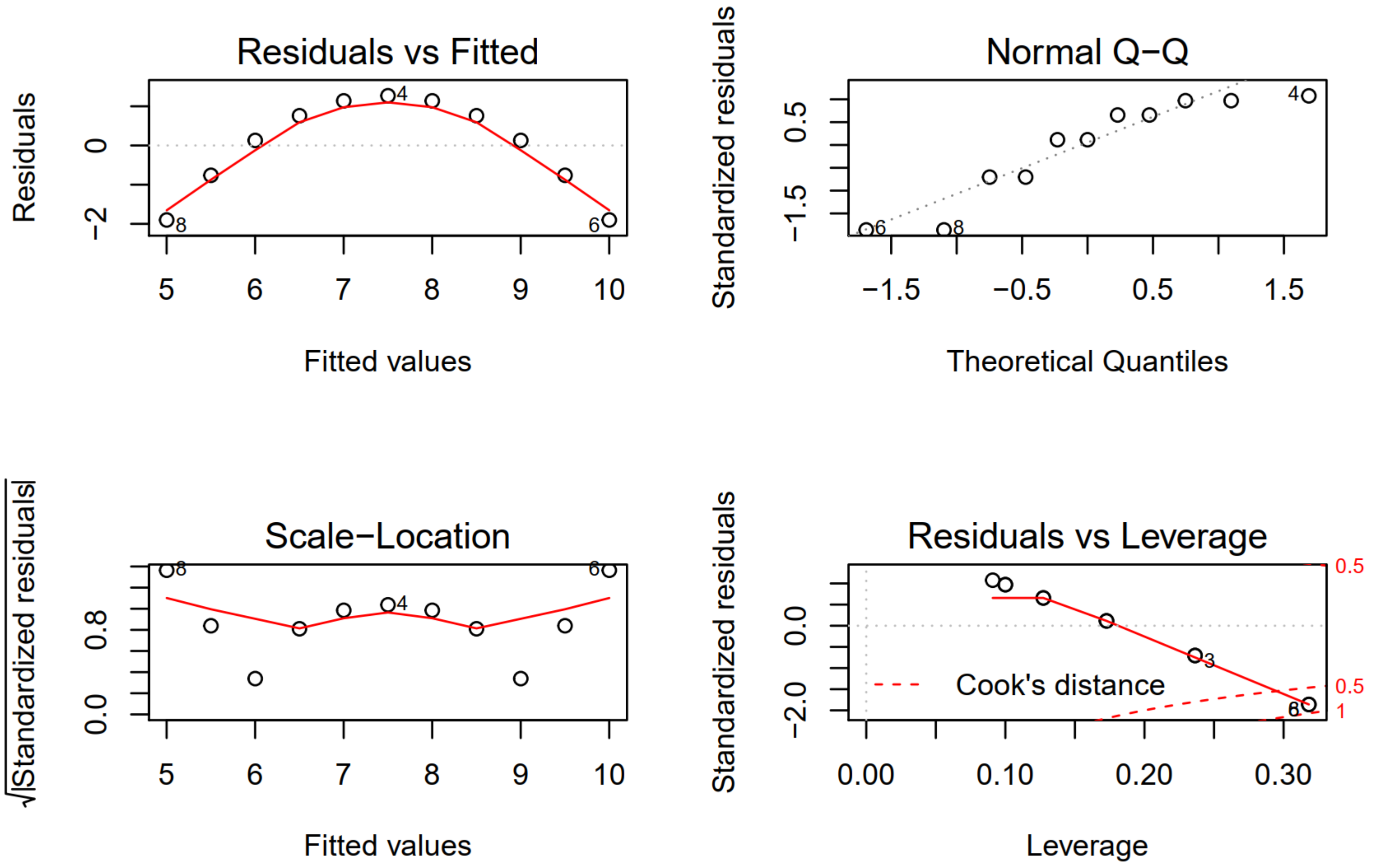


```
lmfit <- lm(y2 ~ x, Quartet)
newdata = data.frame(x = c(3,6,15))
 # 만들어진 모델을 이용하여 예측하고자 하는 새로운 데이터
predict(lmfit, newdata, interval="confidence", level=0.95)
```

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

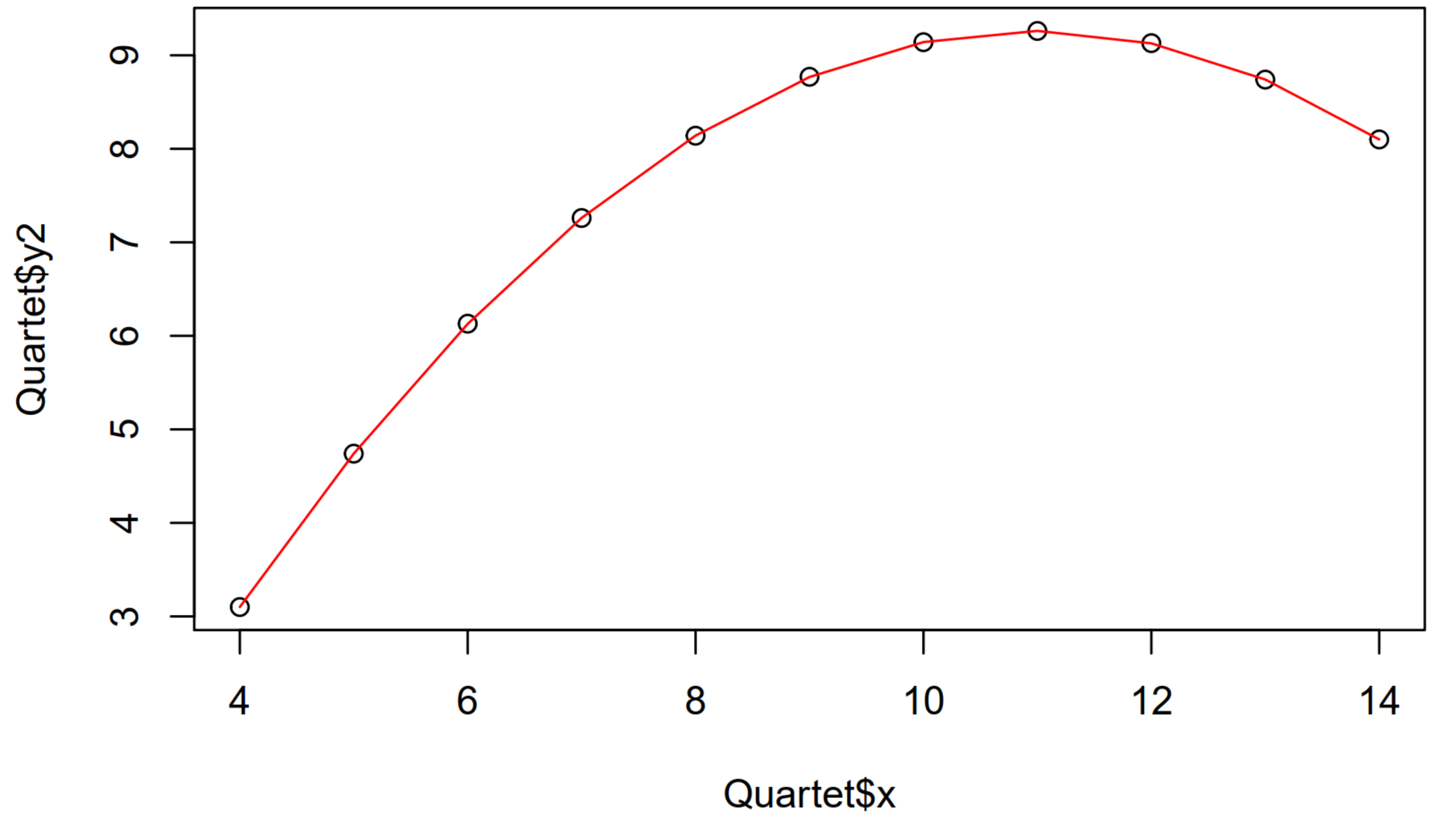
Diagnosis of linear fit

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



Fitting a polynomial regression model with lm

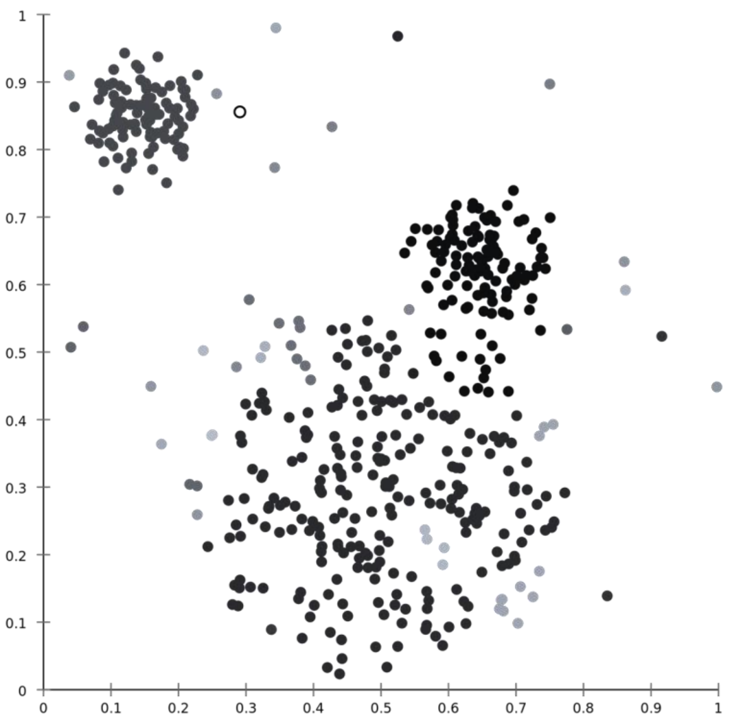
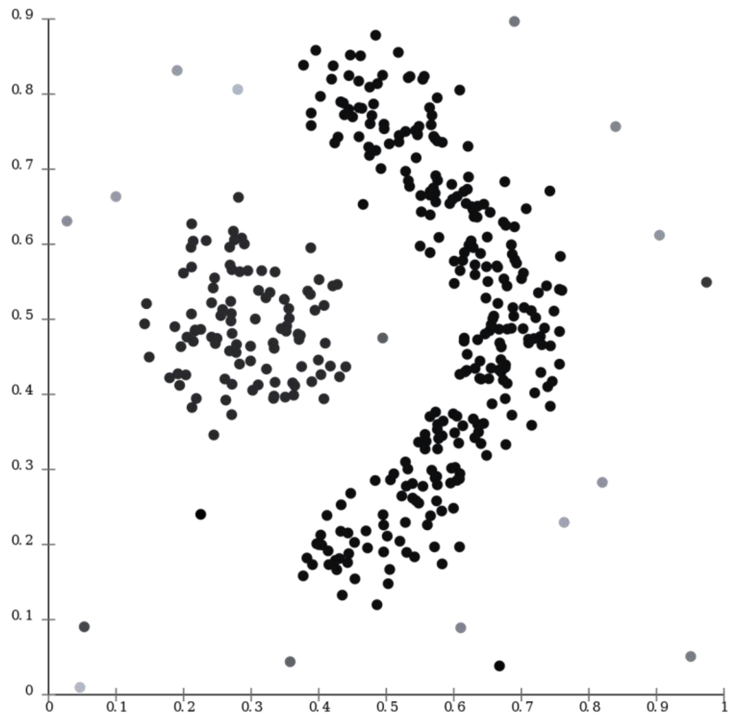
```
lmfit = lm(Quartet$y2~ I(Quartet$x)+I(Quartet$x^2))
lmfit = lm(Quartet$y2~poly(Quartet$x,2)) # 위의 명령줄과 같은 결과
plot(Quartet$x, Quartet$y2) # 데이터의 형태
lines(sort(Quartet$x), lmfit$fit[order(Quartet$x)], col = "red") # 모델의 형태
```

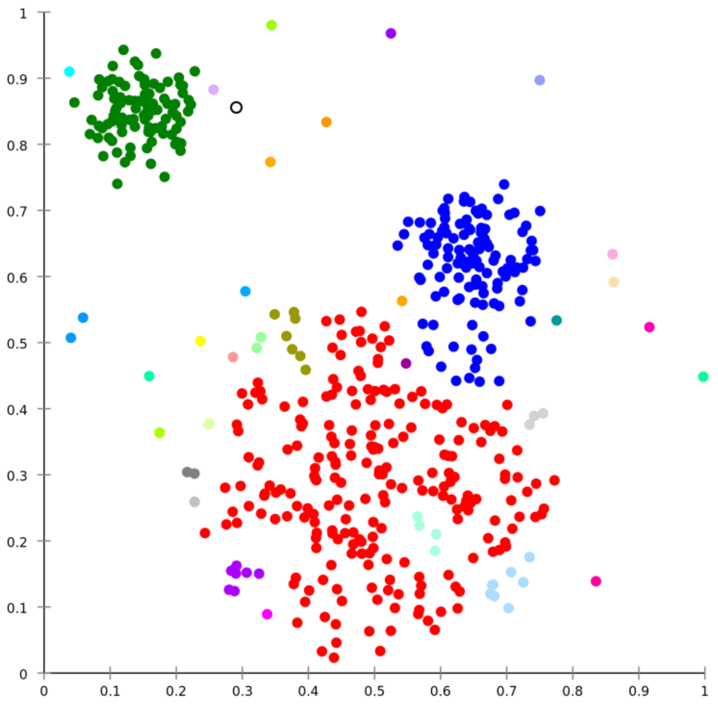


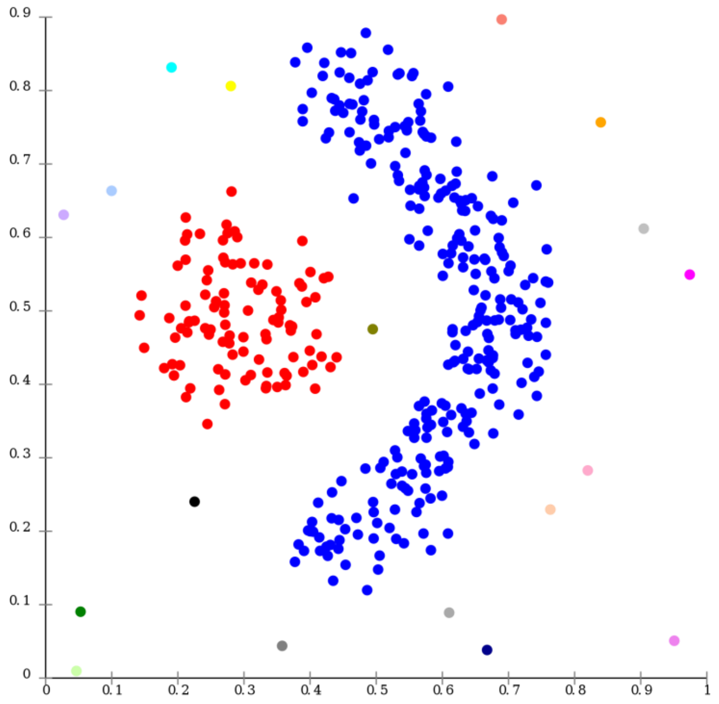
Clustering

Clustering analysis

Cluster analysis or clustering is the task of <u>grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters).</u> It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics. - Wikipedia


0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

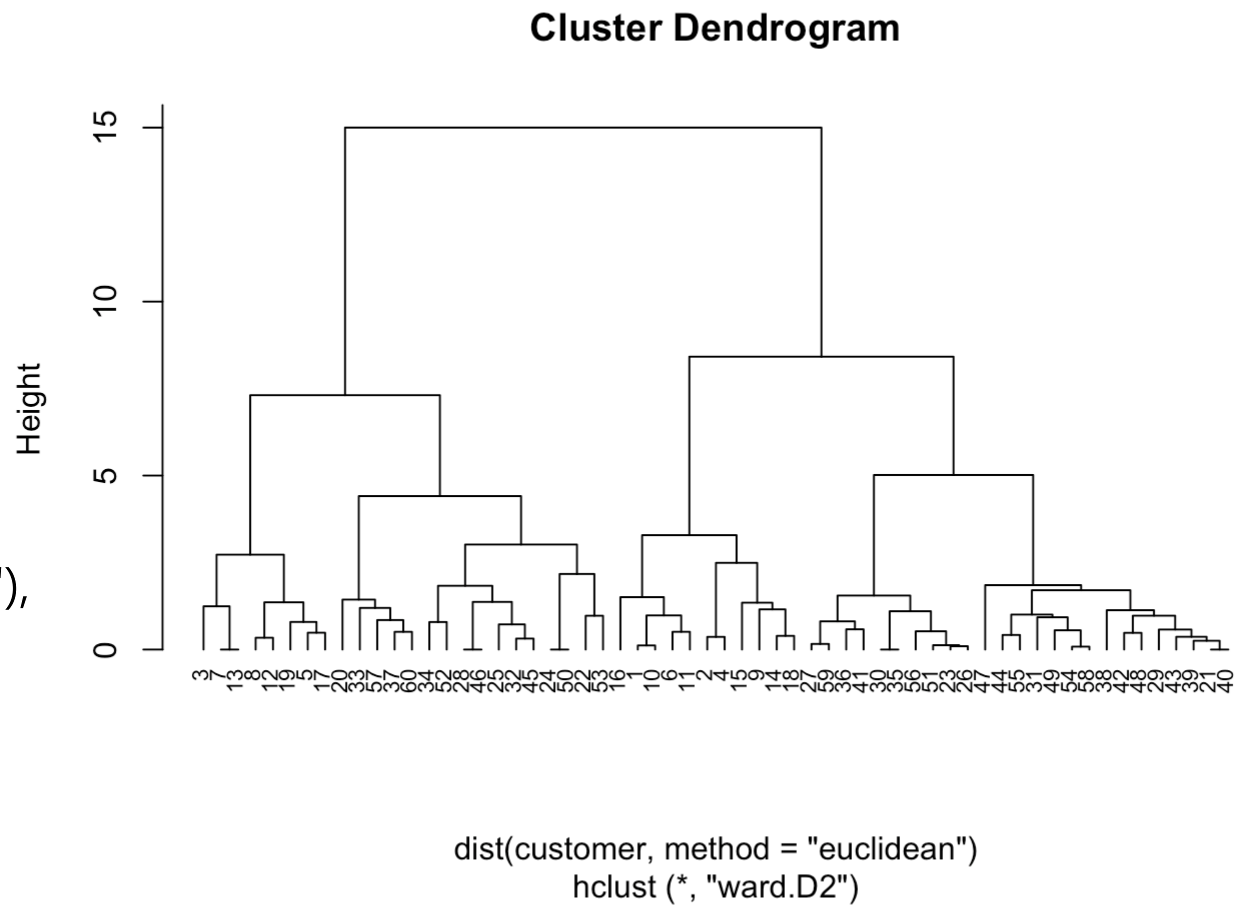
Clustering Data With Hierarchical Clustering

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

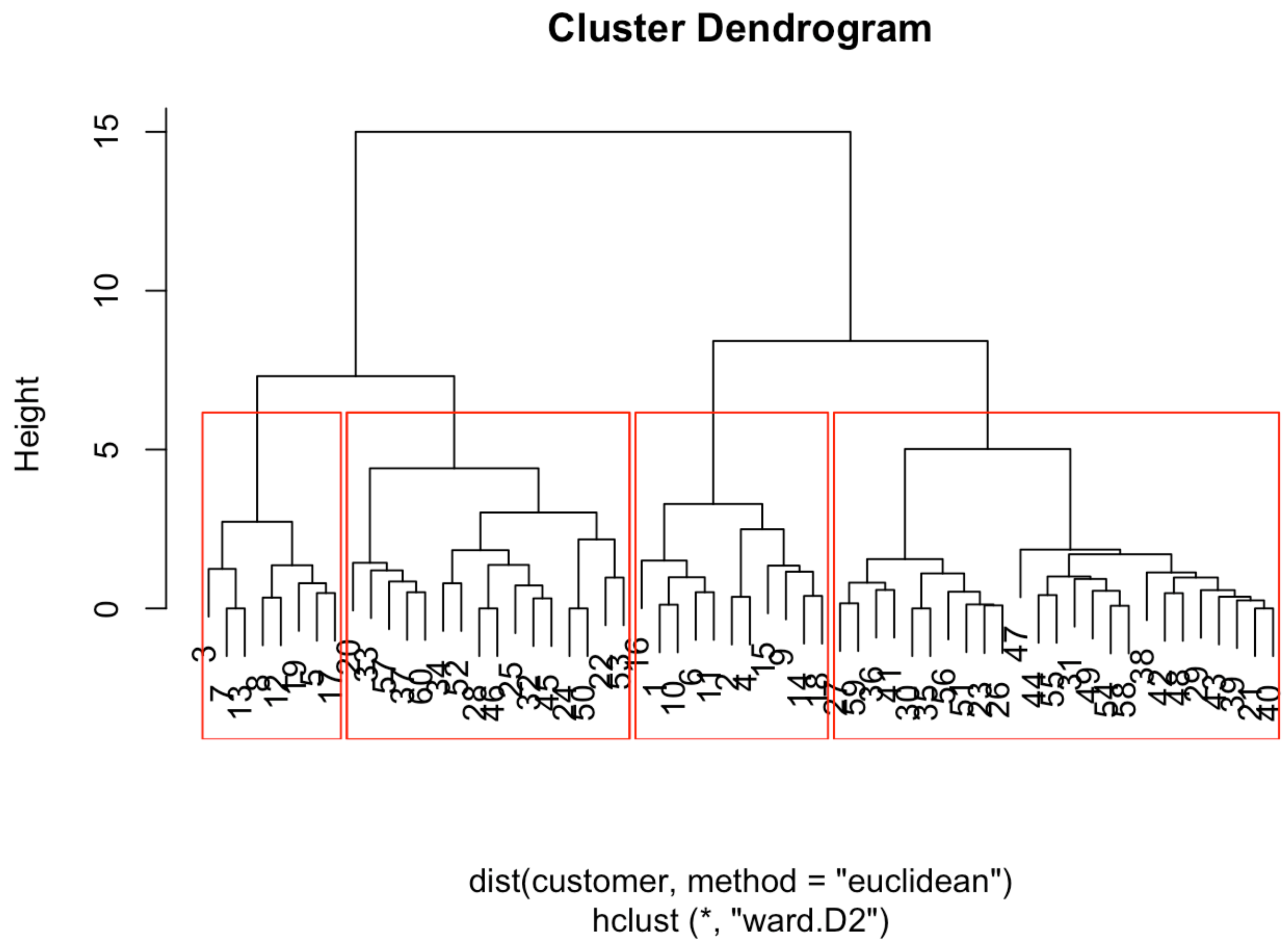
```
##   ID Visit.Time Average.Expense Sex Age
## 1  1          3             5.7   0  10
## 2  2          5            14.5   0  27
## 3  3         16            33.5   0  32
## 4  4          5            15.9   0  30
## 5  5         16            24.9   0  23
## 6  6          3            12.0   0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

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

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



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

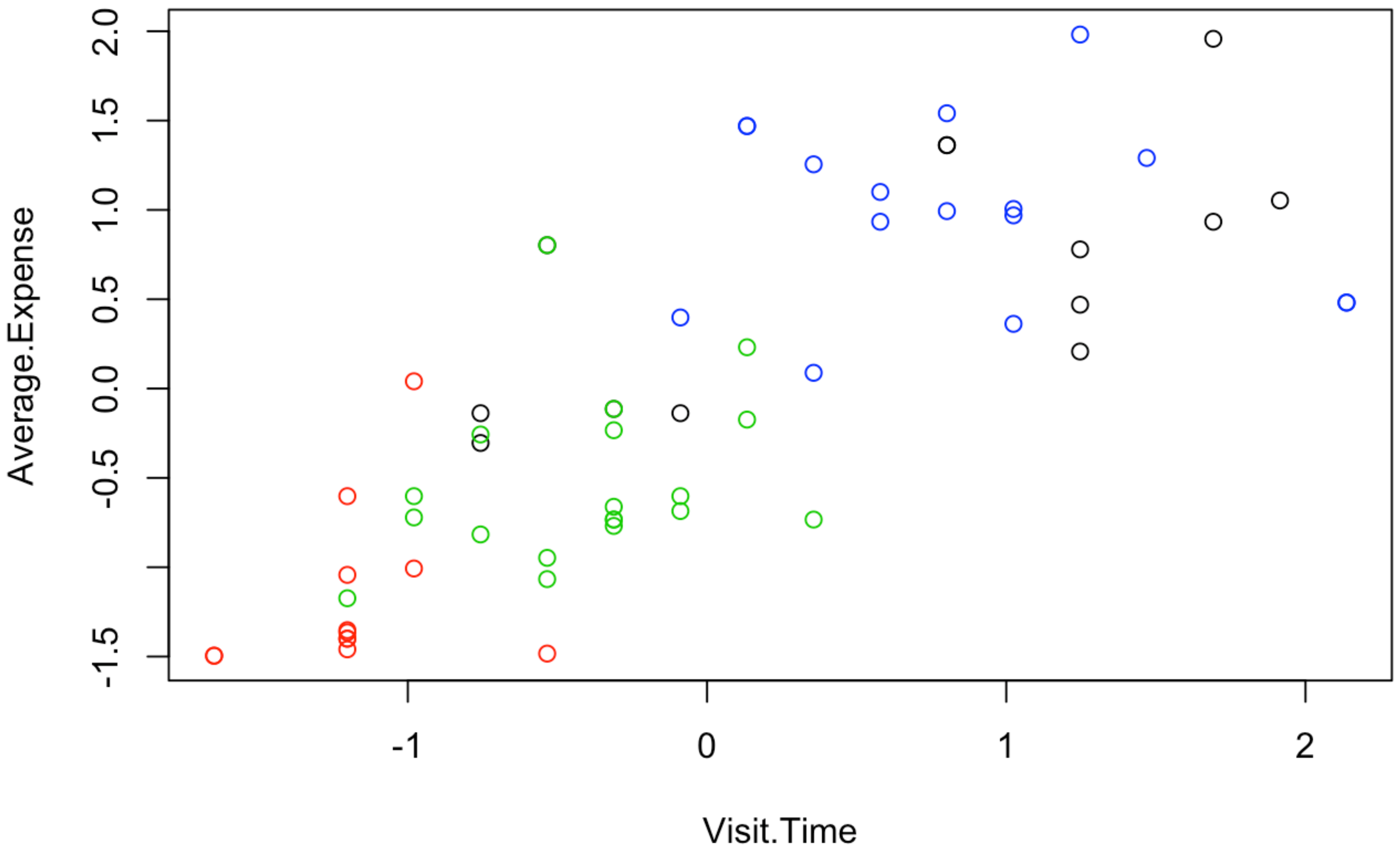


K-means clustering

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

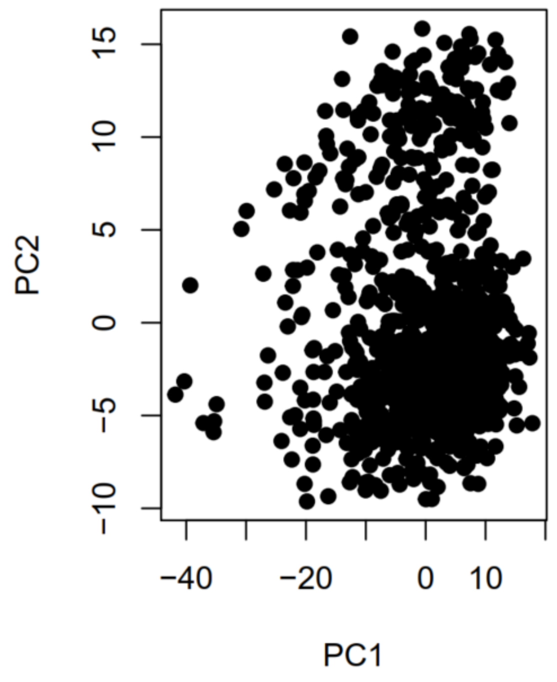
```
## K-means clustering with 4 clusters of sizes 13, 12, 19, 16
##
## Cluster means:
##   Visit.Time Average.Expense        Sex        Age
## 1  0.6302081       0.6332563 -1.4566845  0.3509841
## 2 -1.1836382      -1.1717951 -0.3908178 -1.0697974
## 3 -0.4054171      -0.5258169  0.6750489 -0.4491184
## 4  0.8571173       0.9887331  0.6750489  1.0505015
##
## Clustering vector:
##  [1] 2 1 1 1 1 2 1 1 1 2 2 1 1 1 2 2 1 1 1 4 3 4 2 4 4 2 3 4 3 2
3 4 4 4 2
## [36] 3 4 3 3 3 2 3 3 3 4 4 3 3 3 4 2 4 4 3 3 3 4 3 3 4
##
## Within cluster sum of squares by cluster:
## [1] 23.118195 18.723003  9.947449 22.582360
##  (between_SS / total_SS =  68.5 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

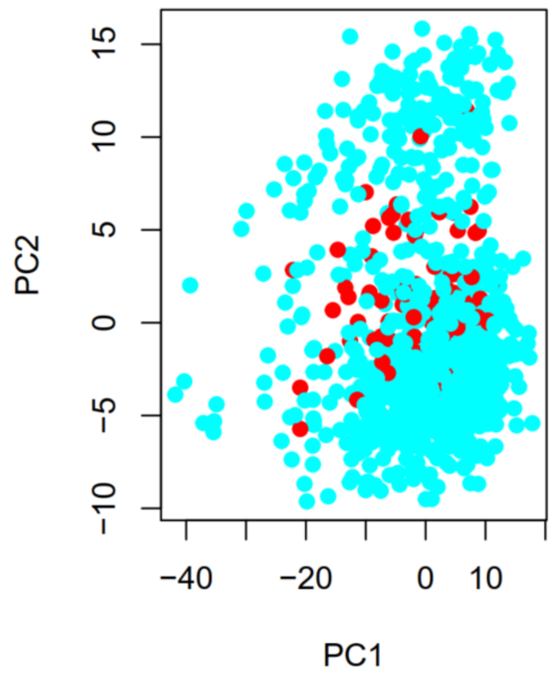
```
plot(customer, col = fit$cluster)
```



Principal component analysis (PCA)

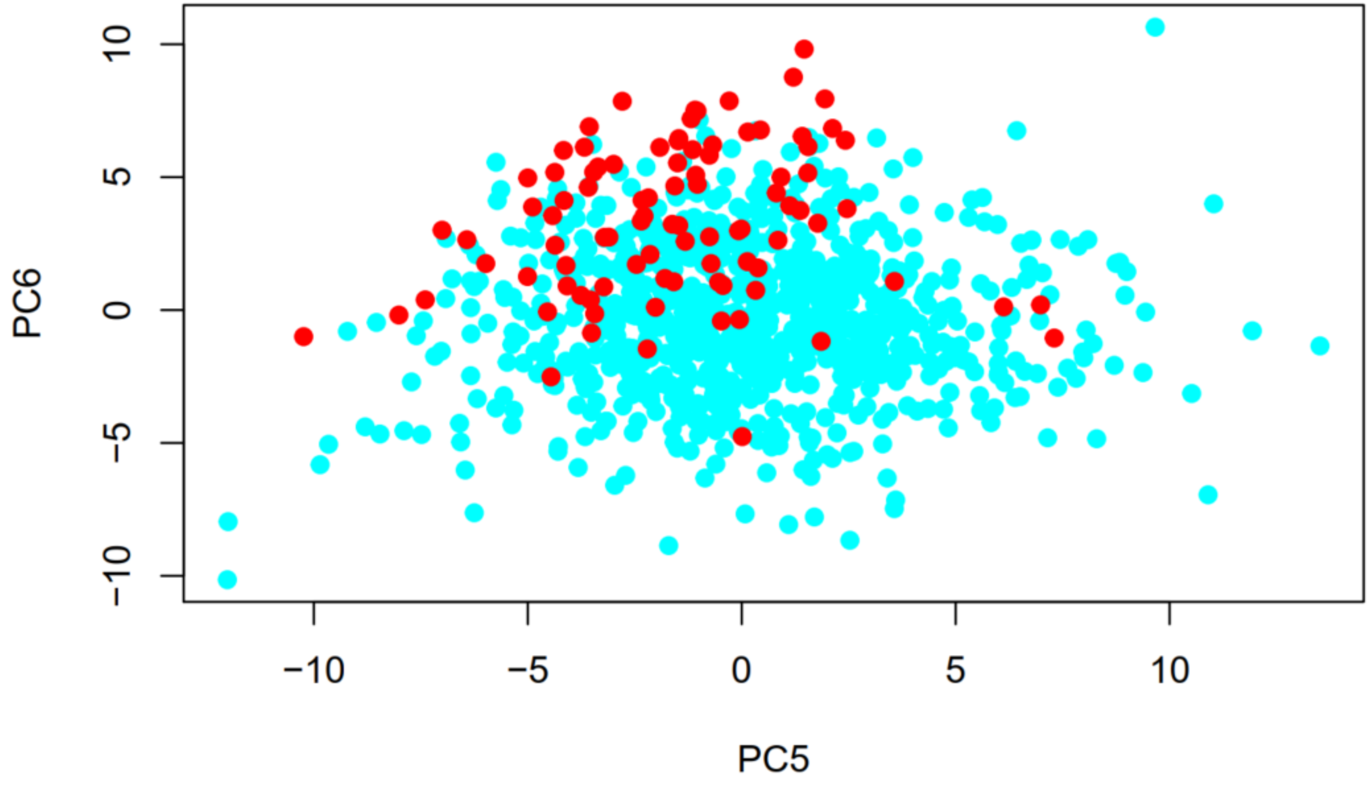
```
brca.cnv <-
read.delim("../data/TCGA_BRCA_CNV_processed.txt")
brca.expr <-
read.delim("../data/TCGA_BRCA_Expr_processed.txt")
pr.res <- prcomp(brca.expr, scale = TRUE)
par(mfrow = c(1, 2))
plot(pr.res$x[, c(1, 2)], pch = 19)
plot(pr.res$x[, c(1, 2)], pch = 19, col = ifelse(brca.cnv[,
"ERBB2_CN"]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





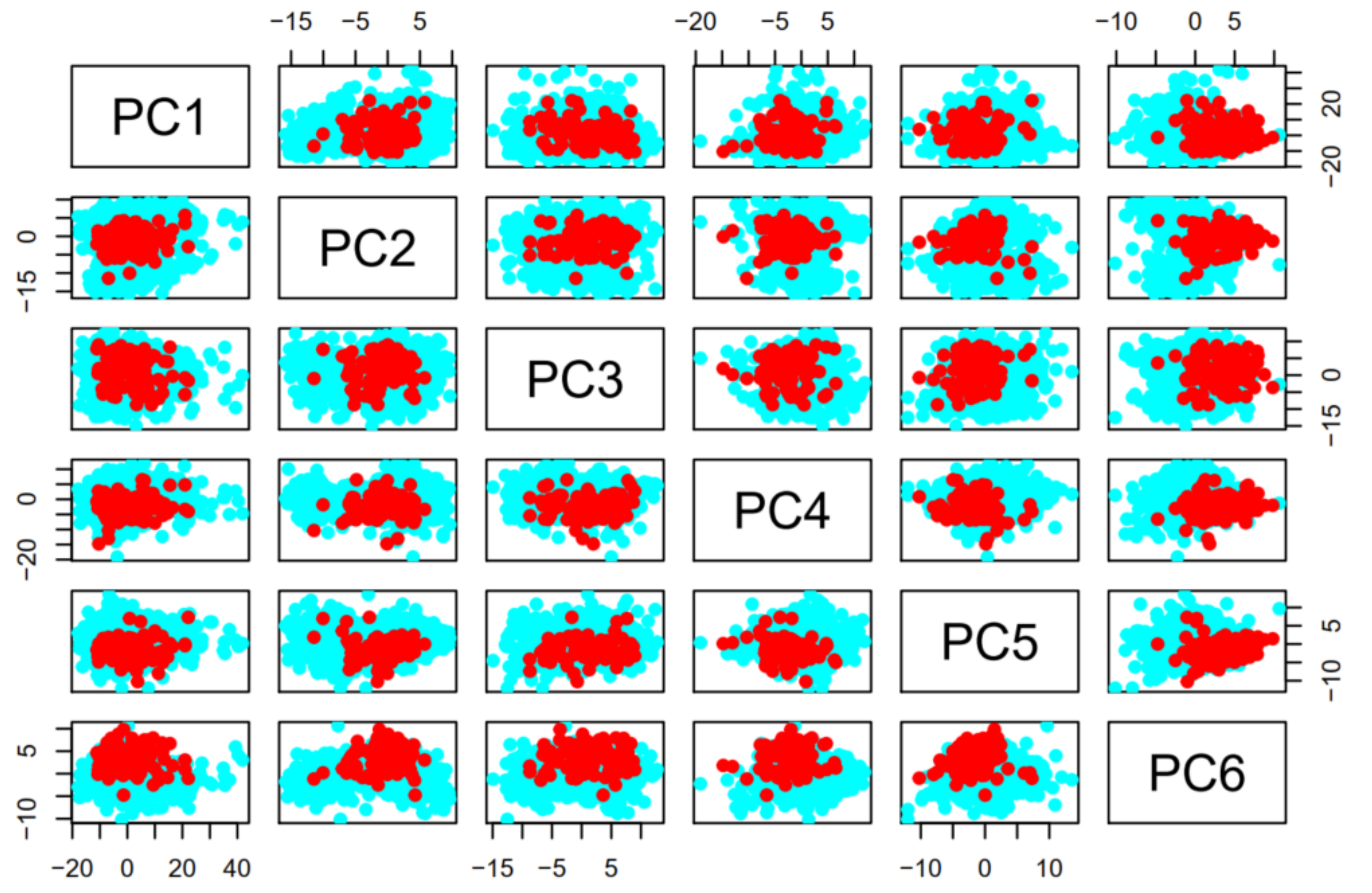
원하는 포인트를 위로

```
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]))
```

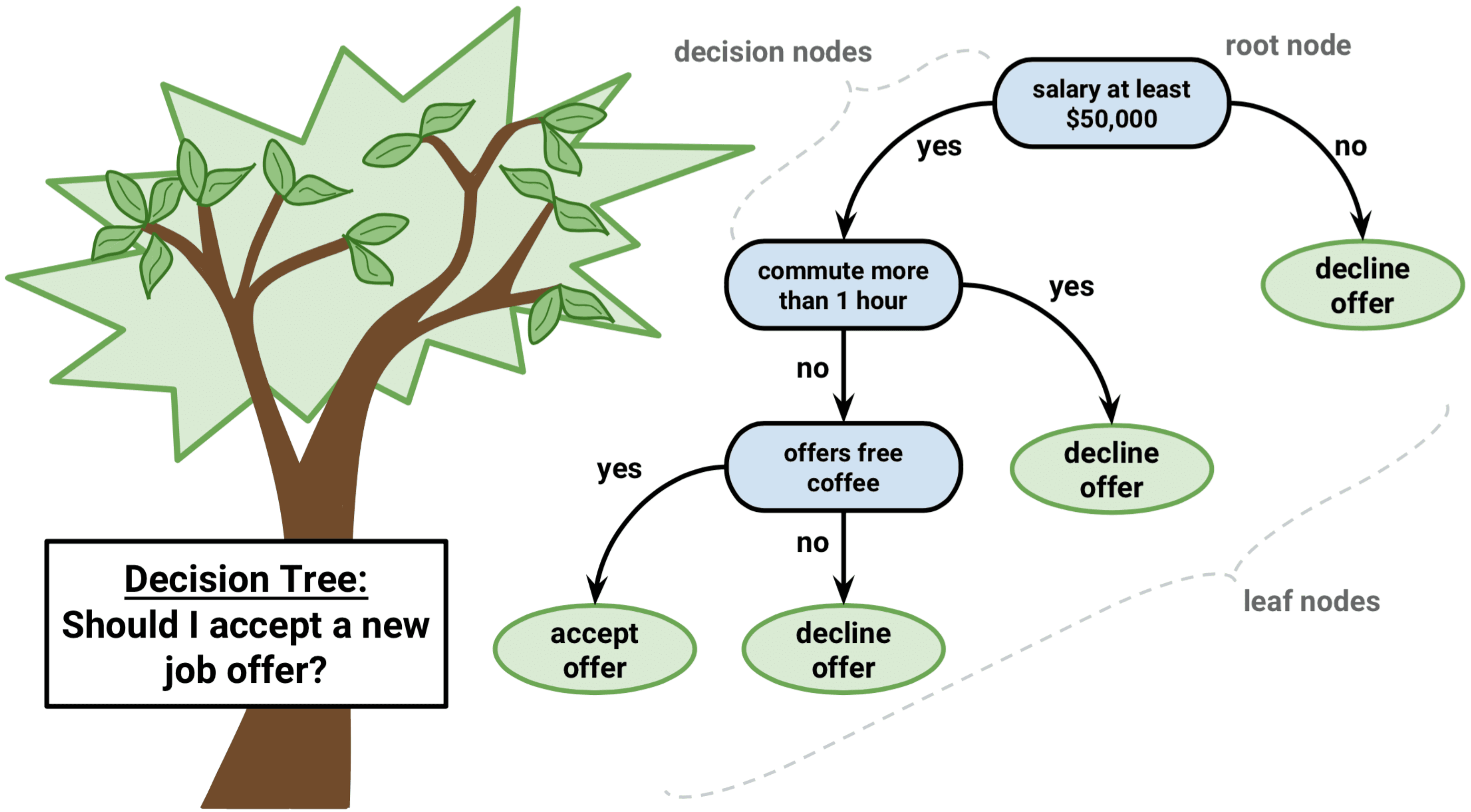


pairs 함수의 활용

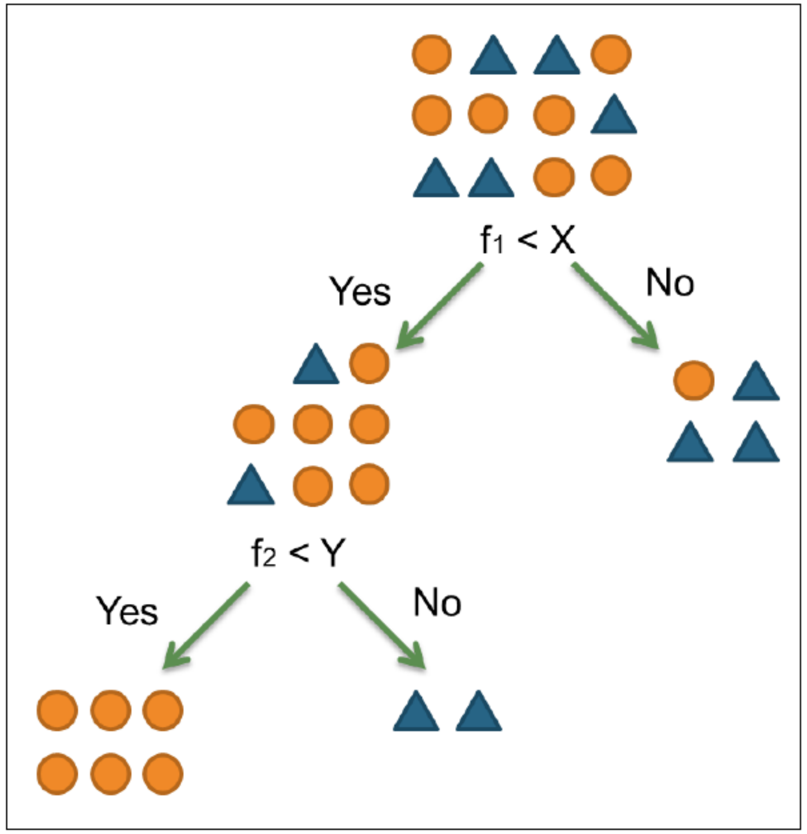
```
pairs(pr.res$x[, 1:6], col = ifelse(brca.cnv[, "ERBB2_CN"] > 3, rainbow(2)[1],
rainbow(2)[2]), pch = 19)
```

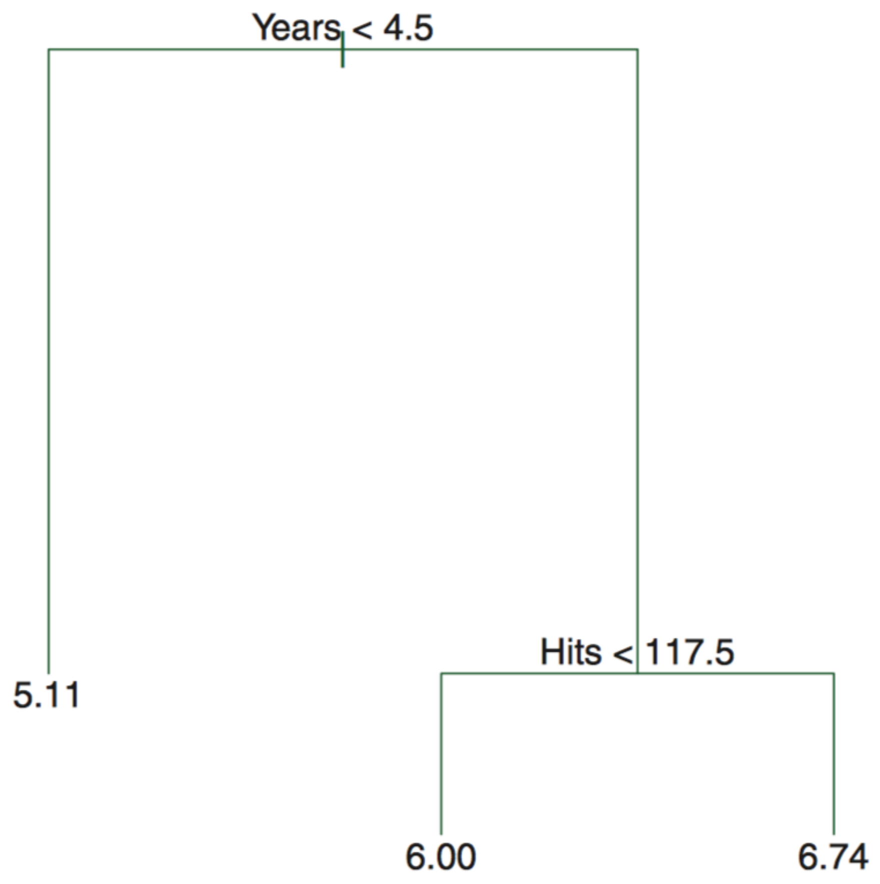


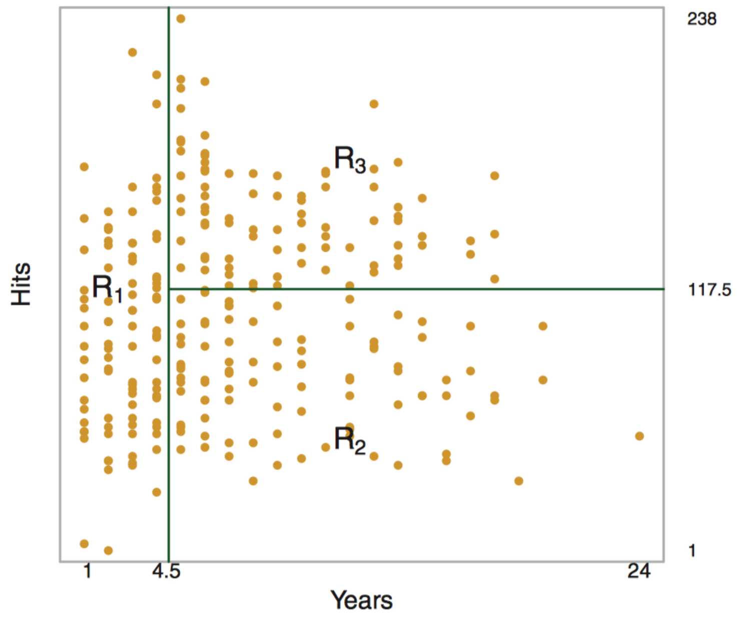
Classification


Decision Tree:
Should I accept a new job offer?
decision nodes
root node
salary at least $50,000
yes
no
commute more than 1 hour
yes
decline offer
no
offers free coffee
decline offer
yes
no
accept offer
decline offer
leaf nodes

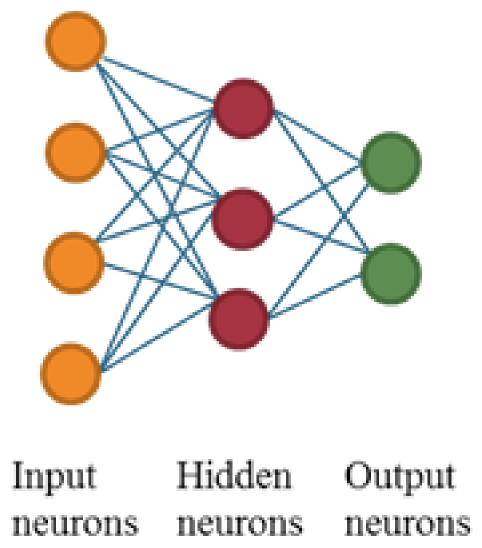
Recursive partitioning tree




Years < 4.5
Hits < 117.5
5.11
6.00
6.74


Hits
Years
R_1
R_2
R_3
238
117.5
1
1
4.5
24

Artificial Neural Network



```
> network = neuralnet(versicolor + virginica + seto
sa~ Sepal. Length + Sepal.Width + Petal.Length + P
etal.Width, trainset, hidden=3)
```

Preparing the training and testing datasets

```
install.packages("C50")
```

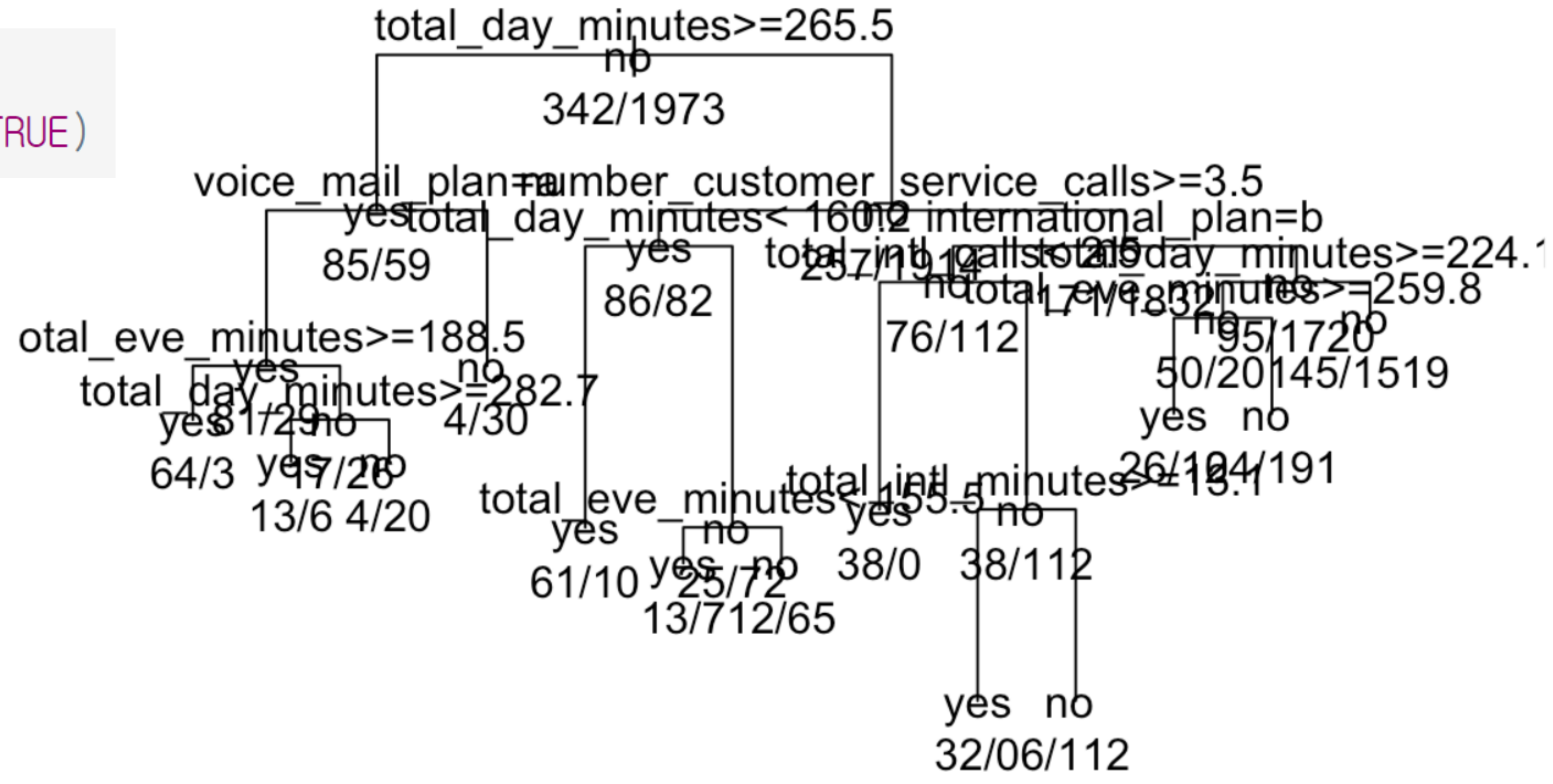
```
library(C50)
data(churn)
str(churnTrain)
```

```
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, ]
```

Recursive partitioning trees

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

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



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

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

```r
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
```

Classifying data with logistic regression

```r
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_plan1         -2.0534243  0.1726694 -11.892  < 2e-16 ***
## voice_mail_plan1             1.3445887  0.6618905   2.031 0.042211 *
## number_vmail_messages       -0.0155101  0.0209220  -0.741 0.458496
```

```
## 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
```

```r
pred = predict(fit, testset, type = "response")
```

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

```r
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)
```

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

```r
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
```

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, ]
```

```
library(neuralnet)
trainset$setosa = trainset$Species == "setosa"
trainset$virginica = trainset$Species == "virginica"
trainset$versicolor = trainset$Species == "versicolor"
```

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

```
plot(network)


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


confusionMatrix(predict.table)
```

```
## Confusion Matrix and Statistics
##
##               prediction
##                setosa versicolor virginica
##   setosa          14          0         0
##   versicolor       0         11         1
##   virginica        0          1        13
##
## Overall Statistics
##
##                  Accuracy : 0.95
##                    95% CI : (0.8308031, 0.9938864)
##      No Information Rate : 0.35
##      P-Value [Acc > NIR] : 0.00000000000001601378
##
##                     Kappa : 0.924812
##  Mcnemar's Test P-Value : NA
```

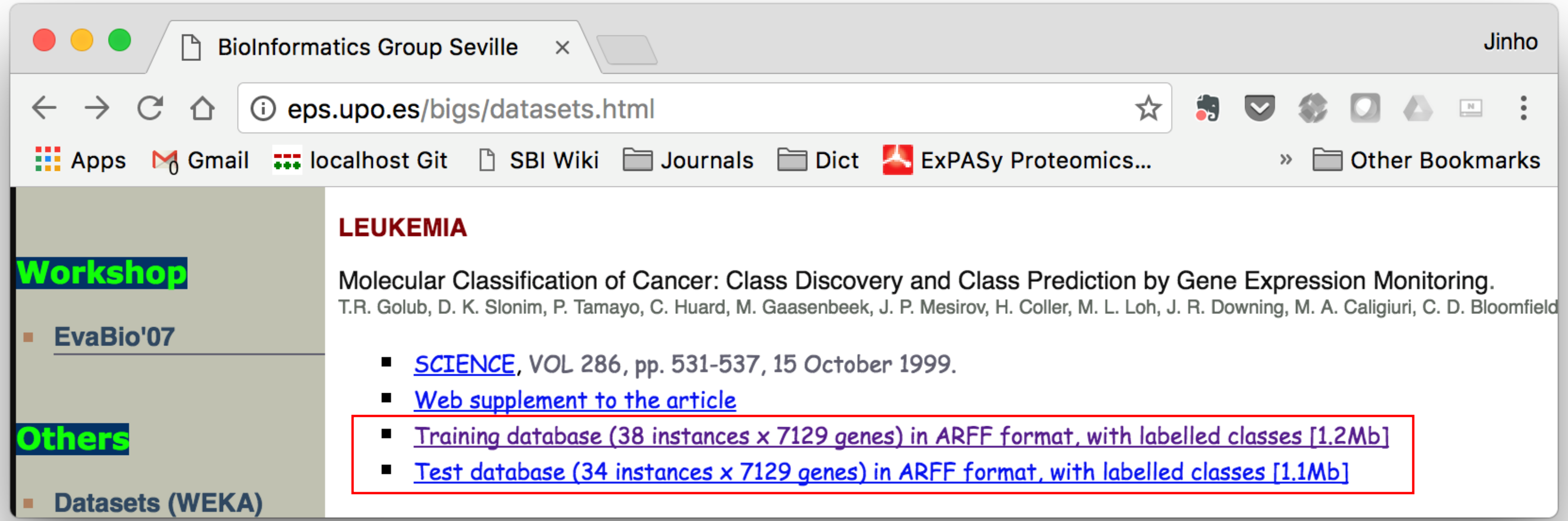
Exercise 1

- CGA_BRCA_CNV_processed.txt, TCGA_BRCA_SNV_processed.txt, TCGA_BRCA_Expr_processed.txt 파일을 읽어들이고 head함수를 이용해서 읽어들인 데이터의 첫 5행과 5열을 출력하세요.
- ERBB2 유전자의 expression과 copy number를 각각 x축과 y축으로하는 scatter plot을 그리고 linear model을 만들어 추세선을 그리세요.
- linear model의 summary를 출력하세요.
- plot 함수를 이용하여 모델을 평가하는 plot 4개를 그리세요.
- polynomial regression을 이용해서 model을 만들고 평가 plot을 그리세요

Exercise 2

아래 웹페이지에서 데이터 다운 받아서 logistic regression과 neural network으로 모델 만들고 테스트하기

http://eps.upo.es/bigs/datasets.html



LEUKEMIA

Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring.
T.R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield

- SCIENCE, VOL 286, pp. 531-537, 15 October 1999.
- Web supplement to the article
- Training database (38 instances x 7129 genes) in ARFF format, with labelled classes [1.2Mb]
- Test database (34 instances x 7129 genes) in ARFF format, with labelled classes [1.1Mb]

```
library(foreign)
Leu.training <- read.arff("../data/leukemia_train_38x7129.arff")
Leu.test <- read.arff("../data/leukemia_test_34x7129.arff")
```

Exercise 2

- Training 데이터를 이용해서 top 10 predictor 도출하기 (t-test 이용)
- Logistic regression 모델을 만들고 평가하기
- Neural network 모델을 만들고 평가하기
- heatmap 함수를 이용해서 hierarchical clustering이 된 형태의 heat map을 그리세요.