R

Yi-Ju Tseng

2020-07-20

Contents

List of Tables

List of Figures

Placeholder

Chapter 1

R 101

Placeholder

1.1 R

1.2

1.3

1.4

1.5

1.5.1 numeric

1.5.2 character

1.5.3 logic

1.5.4 (Date)

1.6

1.6.1

1.6.2

1.7

1.8 Help

Chapter 2

R

Placeholder

2.1 vector

2.1.1

2.1.2

2.2 factor

2.3 list

2.3.1

2.3.2

2.4 matrix

2.5 data.frame

2.6 data.table

2.7

Chapter 3

Placeholder

3.1

3.1.1 if-else

3.1.2 if-else if-else

Chapter 4

Placeholder

4.1

4.2

4.3

4.4

4.5 Functional programming

4.6 purrr

4.7 map2 family

4.8

Chapter 5

Placeholder

5.0.1 ( )

5.0.2

5.0.3 Open Data

5.0.4 API (Application programming interfaces)

5.0.5 XML

5.0.5.1 XML

5.0.5.2 xml2

5.0.6 Webscraping

5.1 Facebook

5.1.1 Graph API in R

5.1.2 Rfacebook package

5.2

5.2.1 .txt

5.2.2 CSV .csv

5.2.3 R .rds

Chapter 6

Placeholder

6.1 Tidy Data

6.2

6.2.1

6.2.2

6.3

6.3.1

6.3.2

6.3.3 (Regular Expression)

6.3.3.1

6.3.3.2

6.3.3.3

6.3.3.4

6.3.3.5

6.3.3.6

6.4 Subset

6.4.1 ( )

6.4.2

6.5

6.5.1 sort

6.5.2 order

6.6

6.7 (Join)

6.8

6.9

6.10 Case study

Chapter 7

Placeholder

7.1

7.2 data.table

7.2.1 i

7.2.2 j

7.2.3 by

7.2.4

7.3 dplyr

7.3.1 select()

7.3.2 filter()

7.3.3 mutate()

7.3.4 summarise()

7.3.5 group_by()

7.3.6 arrange()

7.3.7 rename()

7.3.8

Chapter 8

Placeholder

8.1

8.2 ggplot2

8.2.1 qplot()

8.2.2 ggplot()

8.3 ggplot2+

8.3.1 Choropleth map

8.3.2 ggmap()

8.4 Taiwan

8.4.1 ggmap+

8.5 Treemap

8.6

Chapter 9

Placeholder

9.1 ggvis

9.2 Plot.ly

9.3 Shiny

Chapter 10

10.1

Data mining

- /
- /
- /

-
-
-
-
-
-

- Supervised learning
  - Regression ' '
  - Classification P/N, Yes/No, M/F, Sick/Not sick / (A/B/C/D)
- Unsupervised learning

  - Clustering
  - Association Rules

- Linear Regression
- Logistic Regression
- Support Vector Machines
- Decision Trees
- K-Nearest Neighbor
- Neural Networks
- Deep Learning

- Hierarchical clustering
- K-means clustering
- Neural Networks
- Deep Learning

R

10.2 Regression

Regression Analysis

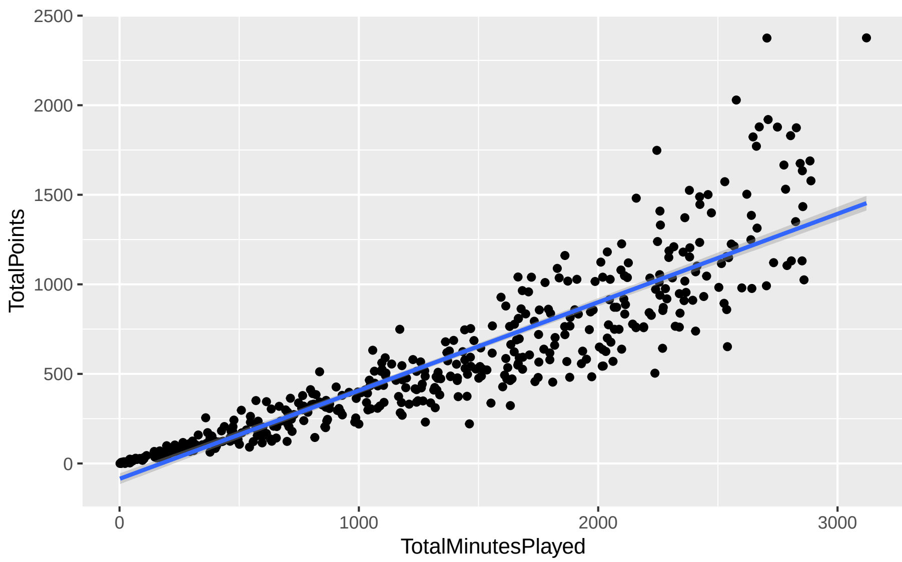
- Linear Regression
- Logistic Regression

10.2.1 Linear Regression

Linear Regression NBA NBA

```
# SportsAnalytics package
library(SportsAnalytics)
# 2015-2016
NBA1516<-fetch_NBAPlayerStatistics("15-16")
```

```
library(ggplot2)
ggplot(NBA1516,aes(x=TotalMinutesPlayed,y=TotalPoints))+
    geom_point()+geom_smooth(method = "glm")
```



R `lm()` `lm(formula,data= )` formula formula ~ 1 2 ….

```r
lm(TotalPoints~TotalMinutesPlayed,data =NBA1516)
```

```
##
## Call:
## lm(formula = TotalPoints ~ TotalMinutesPlayed, data = NBA1516)
##
## Coefficients:
##        (Intercept)  TotalMinutesPlayed
##           -85.9071              0.4931
```

TotalPoints `0.4931` * `-85.9071`

TotalPoints = `0.4931` * TotalMinutesPlayed `-85.9071`

generalized linear models (glm) `glm()` `lm()` family

```
- `family="gaussian"`
- `family="binomial"`
- `family="poisson"`
```

Gaussian distribution

Binomial distribution `n` `/`

Poisson distribution

-
-
-
-
- DNA …..

```r
# e+01: 10^1 / e-04: 10^(-4)
glm(TotalPoints~TotalMinutesPlayed+FieldGoalsAttempted,
    data =NBA1516)
```

```
##
## Call:  glm(formula = TotalPoints ~ TotalMinutesPlayed + FieldGoalsAttempted,
##     data = NBA1516)
##
## Coefficients:
##         (Intercept)   TotalMinutesPlayed  FieldGoalsAttempted
##          -1.799e+01           -2.347e-04            1.256e+00
##
## Degrees of Freedom: 475 Total (i.e. Null);  473 Residual
## Null Deviance:     99360000
## Residual Deviance: 2160000  AIC: 5367
```

`-0.0002347` * + `1.255794` * `-17.99` TotalPoints = `-0.0002347` * TotalMinutesPlayed + `1.255794` * FieldGoalsAttempted `-17.99`

formula

```r
glm(TotalPoints~TotalMinutesPlayed+FieldGoalsAttempted+Position,
    data =NBA1516)
```

```
##
## Call:  glm(formula = TotalPoints ~ TotalMinutesPlayed + FieldGoalsAttempted +
##     Position, data = NBA1516)
##
## Coefficients:
##         (Intercept)   TotalMinutesPlayed  FieldGoalsAttempted
##           22.852223            -0.006537             1.275721
##          PositionPF           PositionPG           PositionSF
##          -39.416327           -65.034646           -38.522299
##          PositionSG
##          -52.175144
##
## Degrees of Freedom: 474 Total (i.e. Null);  468 Residual
```

```
##   (1 observation deleted due to missingness)
## Null Deviance:     99080000
## Residual Deviance: 1975000  AIC: 5322
```

```
# e+01: 10^1 / e-04: 10^(-4)
```

TotalPoints TotalPoints = -0.0065 * TotalMinutesPlayed + 1.28 *FieldGoalsAttempted +22.85 + 22.85* PositionPF + -65.03 * PositionPG + -38.52 * PositionSF + -52.18 * PositionSG

Dummy Variable PositionPF PositionPG PositionSF PositionSG PG

- PositionPF=0
- PositionPG=1
- PositionSF=0
- PositionSG=0

(C)

- PositionPF=0
- PositionPG=0
- PositionSF=0
- PositionSG=0

-
- X
- R factor R
-

10.2.2 Logistic Regression

Logistic Regression 0 1 - / - / - family="binomial"

/

```
mydata <- read.csv("https://raw.githubusercontent.com/CGUIM-BigDataAnalysis/BigDataCGUIM/master/b
```

```
# GRE:   , GPA:    , rank:
head(mydata)
```

admit	gre	gpa	rank
0	380	3.61	3
1	660	3.67	3
1	800	4.00	1
1	640	3.19	4
0	520	2.93	4
1	760	3.00	2

```
mydata$rank <- factor(mydata$rank)
mylogit <- glm(admit ~ gre + gpa + rank,
               data = mydata, family = "binomial")
sum<-summary(mylogit)
sum$coefficients
```

```
##                 Estimate  Std. Error   z value     Pr(>|z|)
## (Intercept) -3.989979073 1.139950936 -3.500132 0.0004650273
## gre          0.002264426 0.001093998  2.069864 0.0384651284
## gpa          0.804037549 0.331819298  2.423119 0.0153878974
## rank2       -0.675442928 0.316489661 -2.134171 0.0328288188
## rank3       -1.340203916 0.345306418 -3.881202 0.0001039415
## rank4       -1.551463677 0.417831633 -3.713131 0.0002047107
```

10.2.3

- Akaike's Information Criterion (AIC)
- Bayesian Information Criterion (BIC)

AIC BIC AIC

```
OneVar<-glm(TotalPoints~TotalMinutesPlayed,data =NBA1516)
TwoVar<-glm(TotalPoints~TotalMinutesPlayed+FieldGoalsAttempted,
            data =NBA1516)
ThreeVar<-glm(TotalPoints~TotalMinutesPlayed+FieldGoalsAttempted+Position,
              data =NBA1516)
c(OneVar$aic,TwoVar$aic,ThreeVar$aic)
```

```
## [1] 6338.913 5366.763 5321.972
```

coefficients “ ”

```r
sum2<-summary(TwoVar)
sum2$coefficients
```

```
##                          Estimate  Std. Error     t value      Pr(>|t|)
## (Intercept)         -1.798855e+01 5.659758251 -3.17832538  1.578333e-03
## TotalMinutesPlayed  -2.347183e-04 0.009474631 -0.02477334  9.802462e-01
## FieldGoalsAttempted  1.255794e+00 0.022239494 56.46682752 2.474028e-212
```

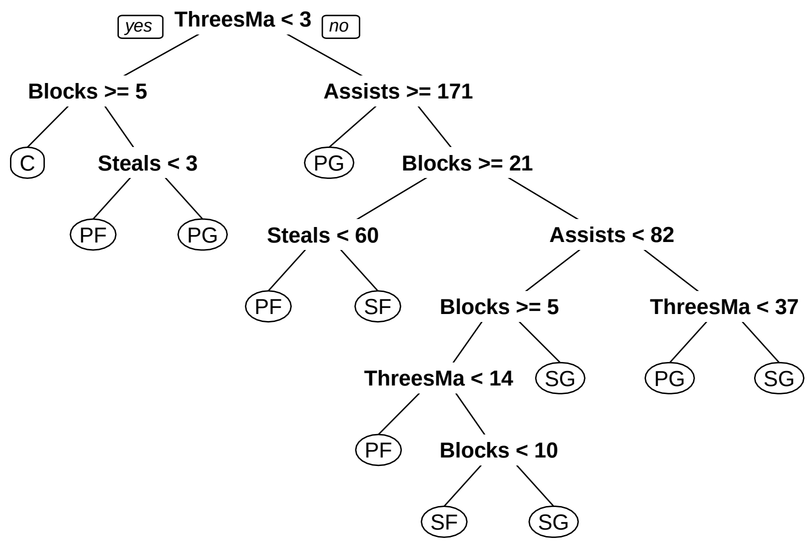
```r
sum3<-summary(ThreeVar)
sum3$coefficients
```

```
##                          Estimate   Std. Error    t value      Pr(>|t|)
## (Intercept)           22.852222668  9.014714391  2.5349913  1.156964e-02
## TotalMinutesPlayed    -0.006536874  0.009199968 -0.7105322  4.777281e-01
## FieldGoalsAttempted    1.275721212  0.021647176 58.9324535 1.144607e-218
## PositionPF           -39.416326742  9.936541704 -3.9668053  8.425605e-05
## PositionPG           -65.034646215 10.269250388 -6.3329497  5.648565e-10
## PositionSF           -38.522298887 10.488170409 -3.6729284  2.674727e-04
## PositionSG           -52.175143670  9.985331185 -5.2251791  2.625062e-07
```

10.3 Decision Trees

(Node)

```
## Warning: package 'rpart.plot' was built under R version 4.0.2
```



Classification And Regression Tree (CART) `rpart` packages (Therneau and Atkinson, 2019)

```
install.packages("rpart")
```

NBA / / / `rpart()` `rpart(formula, data)`

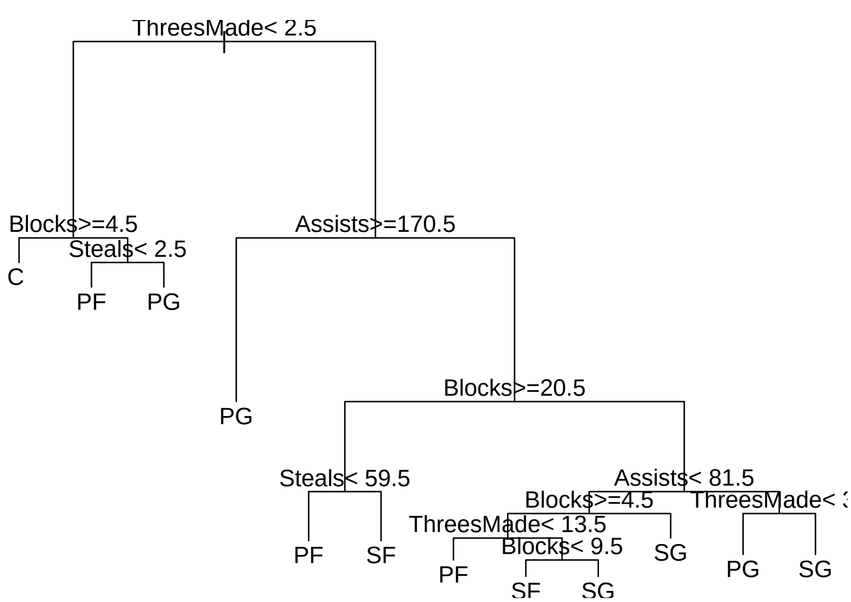
```
library(rpart)
DT<-rpart(Position~Blocks+ThreesMade+Assists+Steals,data=NBA1516)
DT
```

```
## n=475 (1 observation deleted due to missingness)
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 475 364 PF (0.15 0.23 0.21 0.18 0.23)
##     2) ThreesMade< 2.5 132  74 C (0.44 0.35 0.098 0.053 0.061)
##       4) Blocks>=4.5 89  37 C (0.58 0.38 0.011 0.011 0.011) *
##       5) Blocks< 4.5 43  31 PF (0.14 0.28 0.28 0.14 0.16)
##        10) Steals< 2.5 29  19 PF (0.17 0.34 0.14 0.21 0.14) *
##        11) Steals>=2.5 14   6 PG (0.071 0.14 0.57 0 0.21) *
##     3) ThreesMade>=2.5 343 242 SG (0.035 0.19 0.25 0.23 0.29)
##       6) Assists>=170.5 96  39 PG (0.031 0.052 0.59 0.15 0.18) *
```

```
##       7) Assists< 170.5 247 163 SG (0.036 0.24 0.12 0.26 0.34)
##        14) Blocks>=20.5 80  42 PF (0.062 0.48 0 0.26 0.2)
##          28) Steals< 59.5 58  21 PF (0.069 0.64 0 0.14 0.16) *
##          29) Steals>=59.5 22   9 SF (0.045 0.045 0 0.59 0.32) *
##        15) Blocks< 20.5 167  99 SG (0.024 0.13 0.17 0.26 0.41)
##          30) Assists< 81.5 110  68 SG (0.027 0.18 0.091 0.32 0.38)
##            60) Blocks>=4.5 63  39 SF (0.032 0.29 0.016 0.38 0.29)
##             120) ThreesMade< 13.5 19   9 PF (0.11 0.53 0 0.26 0.11) *
##             121) ThreesMade>=13.5 44  25 SF (0 0.18 0.023 0.43 0.36)
##               242) Blocks< 9.5 17   7 SF (0 0.18 0.059 0.59 0.18) *
##               243) Blocks>=9.5 27  14 SG (0 0.19 0 0.33 0.48) *
##            61) Blocks< 4.5 47  23 SG (0.021 0.043 0.19 0.23 0.51) *
##          31) Assists>=81.5 57  31 SG (0.018 0.035 0.33 0.16 0.46)
##            62) ThreesMade< 37 17   5 PG (0 0.12 0.71 0.059 0.12) *
##            63) ThreesMade>=37 40  16 SG (0.025 0 0.17 0.2 0.6) *
```

```
#   PG    SG    SF    PF   C
```

```
par(mfrow=c(1,1), mar = rep(1,4)) # , , ,
plot(DT)
text(DT, use.n=F, all=F, cex=1)
```

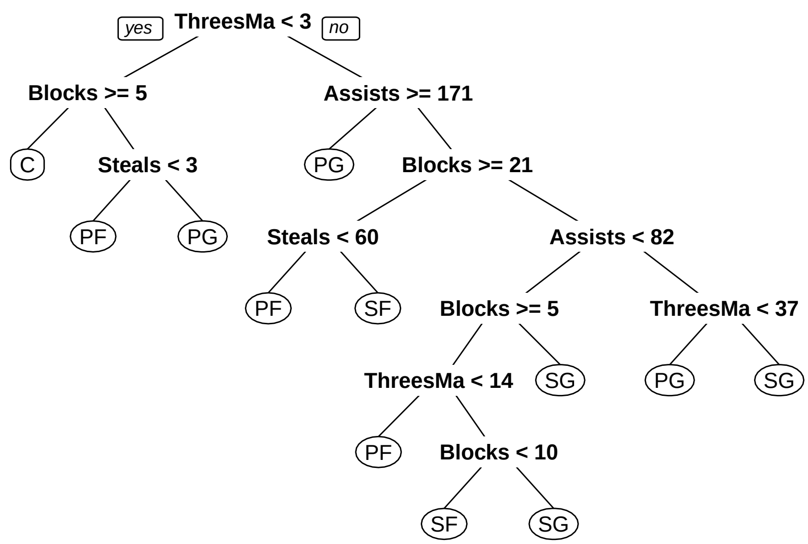


```
#   PG    SG   SF   PF   C
```

`plot()` `rpart.plot` package (Milborrow, 2019) `prp()`

```r
install.packages("rpart.plot") #
```

```r
library(rpart.plot)
prp(DT)
```



- Gini impurity
- Information gain
- Variance reduction

10.4 Clustering

Clustering

-

-
-
-

10.4.1 Hierarchical clustering

- An agglomerative approach
  - Find closest two things
  - Put them together
  - Find next closest
- Requires
  - A defined distance
  - A merging approach
- Produces
  - A tree showing how close things are to each other

`distance`

- Distance or similarity
  - Continuous - euclidean distance
  - Continuous - correlation similarity
  - Binary - manhattan distance
- Pick a distance/similarity that makes sense for your problem

Example distances - Euclidean

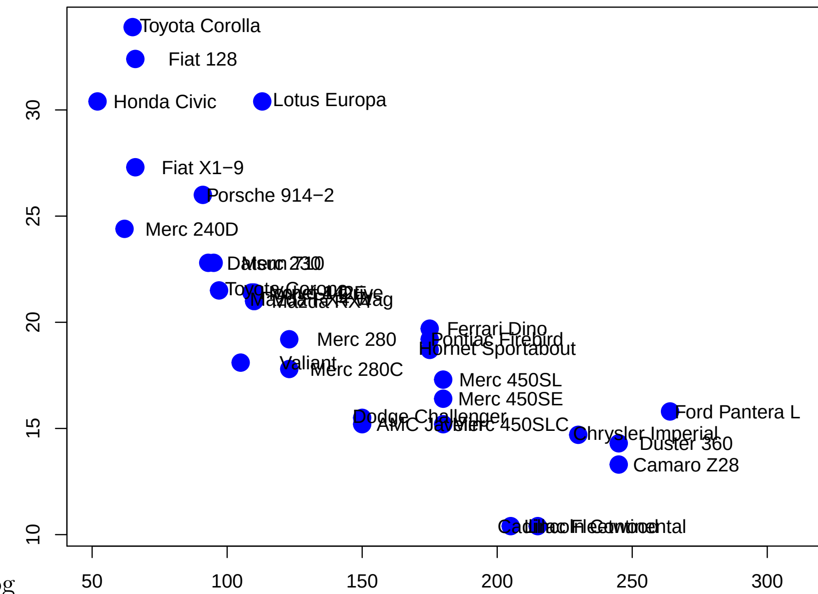
$$\sqrt{(A_1 - A_2)^2 + (B_1 - B_2)^2 + ... + (Z_1 - Z_2)^2}$$

Example distances - Manhattan

$$|A_1 - A_2| + |B_1 - B_2| + ... + |Z_1 - Z_2|$$

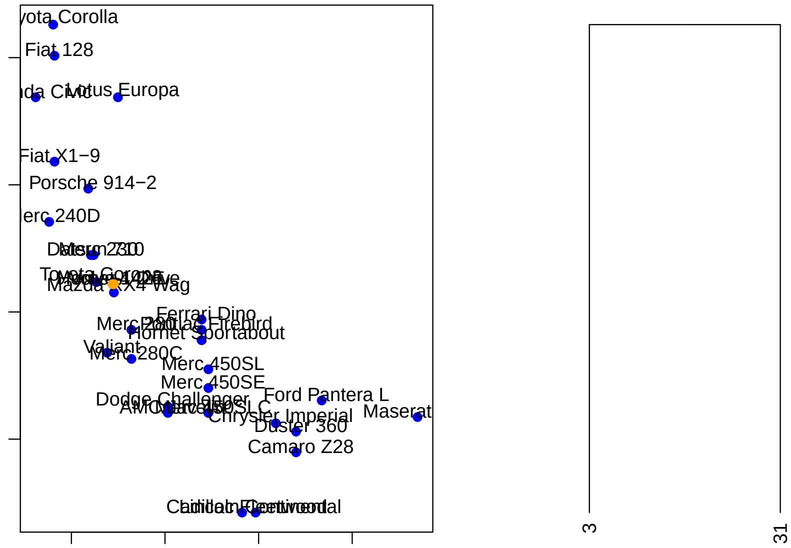
Merging apporach

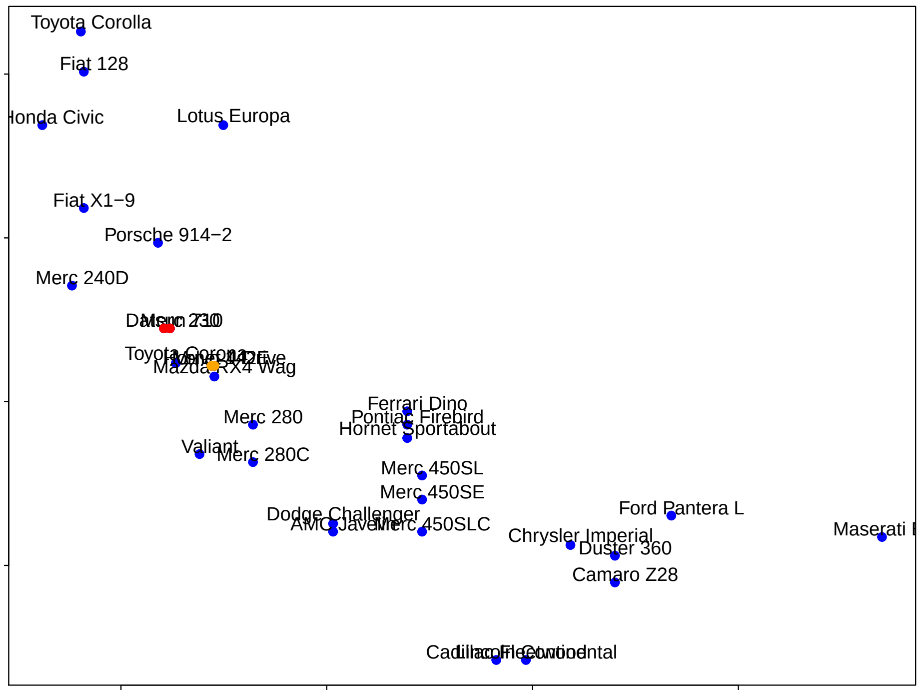
- Agglomerative
  - Single-linkage
  - Complete-linkage
  - Average-linkage



Hierarchical clustering - hp vs. mpg

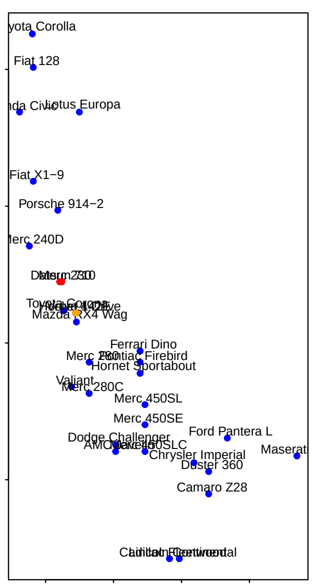
Hierarchical clustering - #1

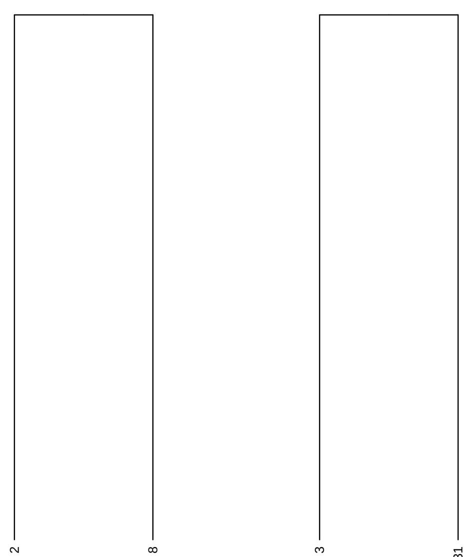




Hierarchical clustering - #2

Hierarchical clustering - #3





`dist()` method=""

```r
mtcars.mxs<-as.matrix(mtcars)
d<-dist(mtcars.mxs) #  euclidean
head(d)
```

```
## [1]   0.6153251  54.9086059  98.1125212 210.3374396  65.4717710 241.4076490
```

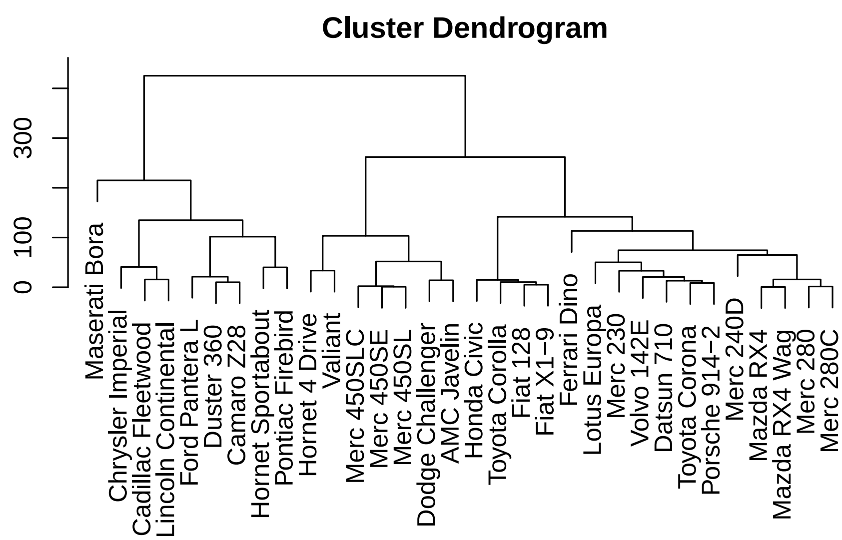
`dist()` "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski"

```r
d<-dist(mtcars.mxs, method="manhattan") # manhattan
head(d)
```

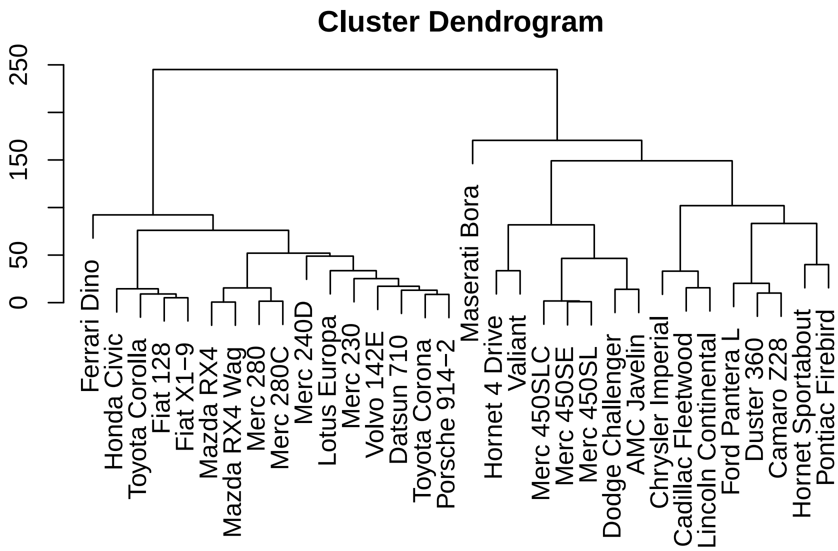
```
## [1]   0.815  79.300 108.795 275.430  84.640 347.960
```

`hclust` `dist()`

```r
par(mar=rep(2,4),mfrow=c(1,1))
hc<-hclust(dist(mtcars.mxs)) # method      complete
plot(hc)
```



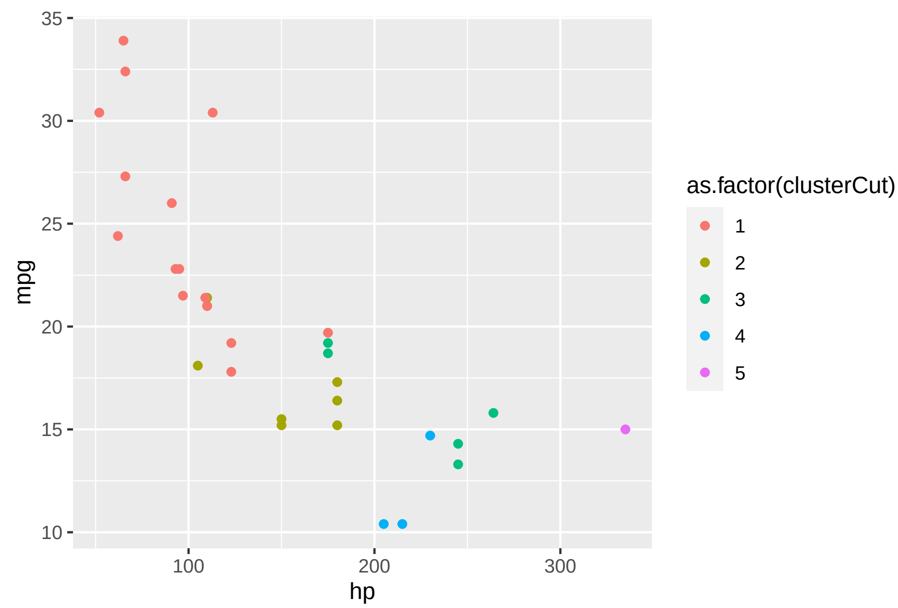

```r
par(mar=rep(2,4),mfrow=c(1,1))
hc<-hclust(dist(mtcars.mxs),method="average") #
plot(hc)
```




```r
clusterCut <- cutree(hc, k=5) # 5
sort(clusterCut)
```

```
##           Mazda RX4       Mazda RX4 Wag          Datsun 710           Merc 240D
##                   1                   1                   1                   1
##            Merc 230            Merc 280           Merc 280C            Fiat 128
##                   1                   1                   1                   1
##         Honda Civic      Toyota Corolla       Toyota Corona           Fiat X1-9
##                   1                   1                   1                   1
##       Porsche 914-2        Lotus Europa        Ferrari Dino          Volvo 142E
##                   1                   1                   1                   1
##      Hornet 4 Drive             Valiant          Merc 450SE          Merc 450SL
##                   2                   2                   2                   2
##         Merc 450SLC    Dodge Challenger         AMC Javelin   Hornet Sportabout
##                   2                   2                   2                   3
##          Duster 360          Camaro Z28    Pontiac Firebird      Ford Pantera L
##                   3                   3                   3                   3
##  Cadillac Fleetwood Lincoln Continental   Chrysler Imperial       Maserati Bora
##                   4                   4                   4                   5
```

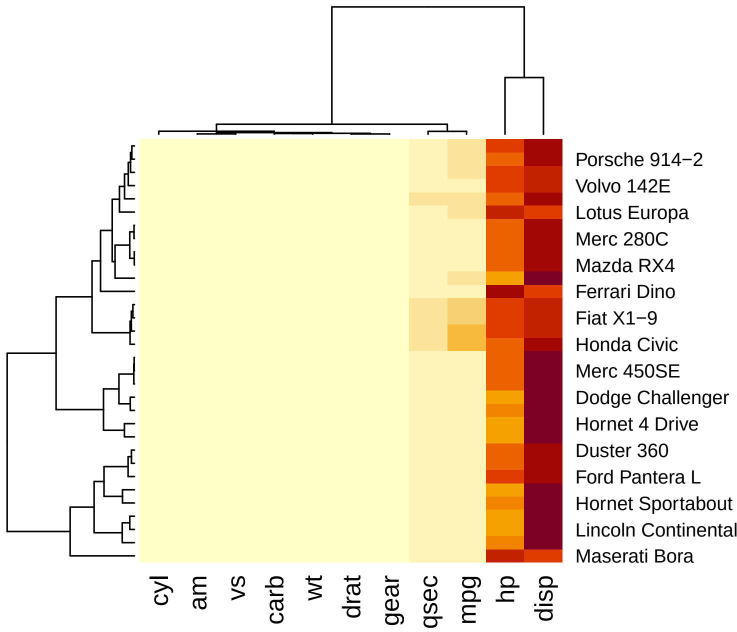
```r
ggplot()+geom_point(data=mtcars,
                    aes(x=hp,y=mpg,color=as.factor(clusterCut)))
```



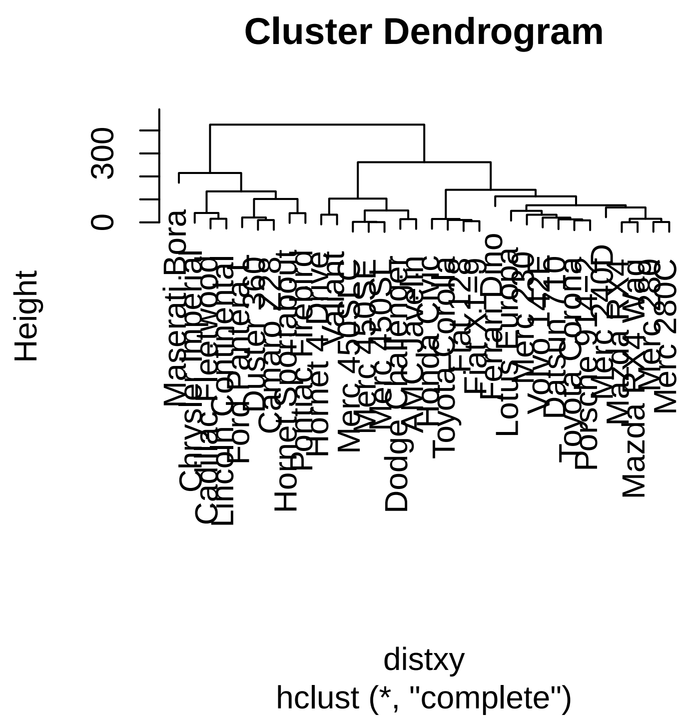

```
clusterCut <- cutree(hc,h =4) #   =4    =4
sort(clusterCut)
```

```
##          Mazda RX4       Mazda RX4 Wag          Datsun 710      Hornet 4 Drive
##                  1                   1                   2                   3
##  Hornet Sportabout             Valiant          Duster 360           Merc 240D
##                  4                   5                   6                   7
##           Merc 230            Merc 280           Merc 280C          Merc 450SE
##                  8                   9                   9                  10
##         Merc 450SL         Merc 450SLC  Cadillac Fleetwood Lincoln Continental
##                 10                  10                  11                  12
##  Chrysler Imperial            Fiat 128         Honda Civic      Toyota Corolla
##                 13                  14                  15                  16
##      Toyota Corona    Dodge Challenger         AMC Javelin          Camaro Z28
##                 17                  18                  19                  20
##   Pontiac Firebird           Fiat X1-9       Porsche 914-2        Lotus Europa
##                 21                  22                  23                  24
##     Ford Pantera L        Ferrari Dino       Maserati Bora          Volvo 142E
##                 25                  26                  27                  28
```

```
par(mar=rep(0.2,4),mfrow=c(1,1))
heatmap(mtcars.mxs)
```




```r
distxy <- dist(mtcars.mxs)
hClustering <- hclust(distxy)
plot(hClustering)
```



Hierarchical clustering: summary -

-
  -
  -
  -
-
  -

10.4.2 K-means clustering

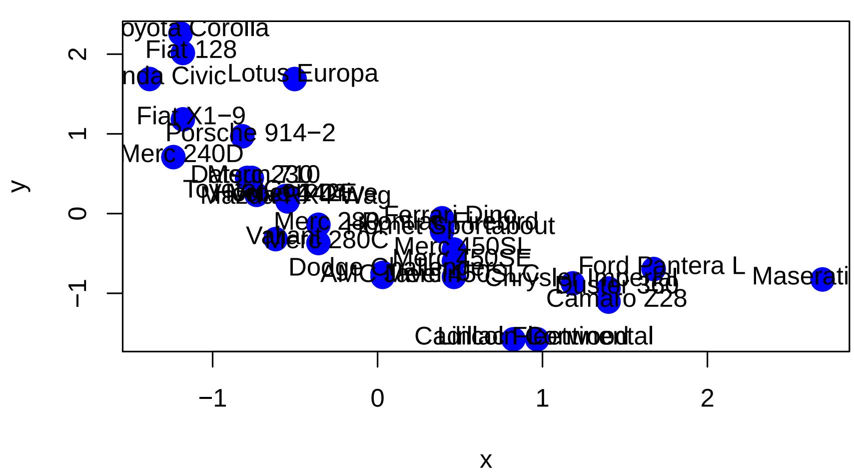
-
  -
  -
  - /
  -
-

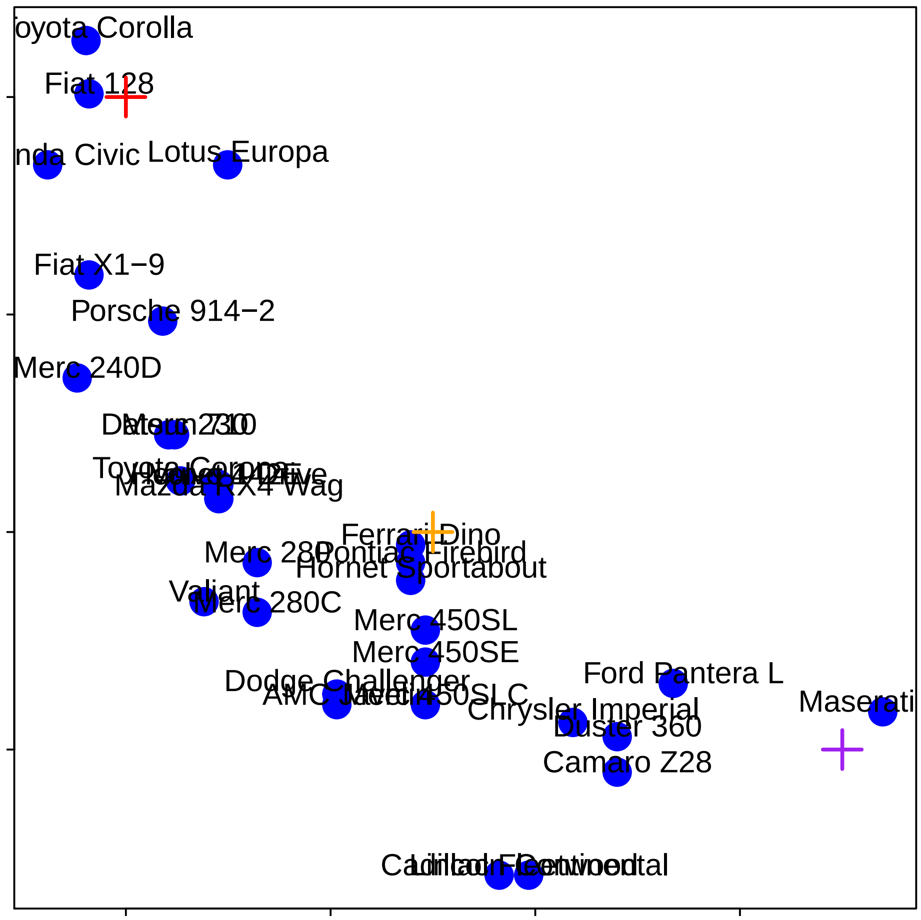
- –
- – # of clusters
- –

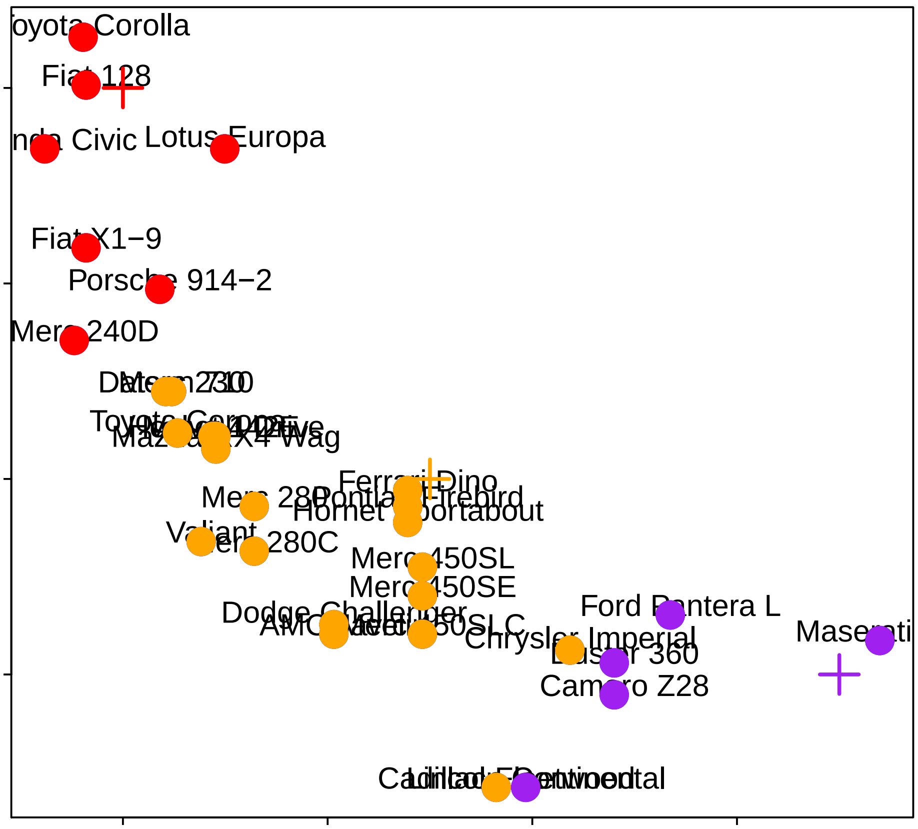
•

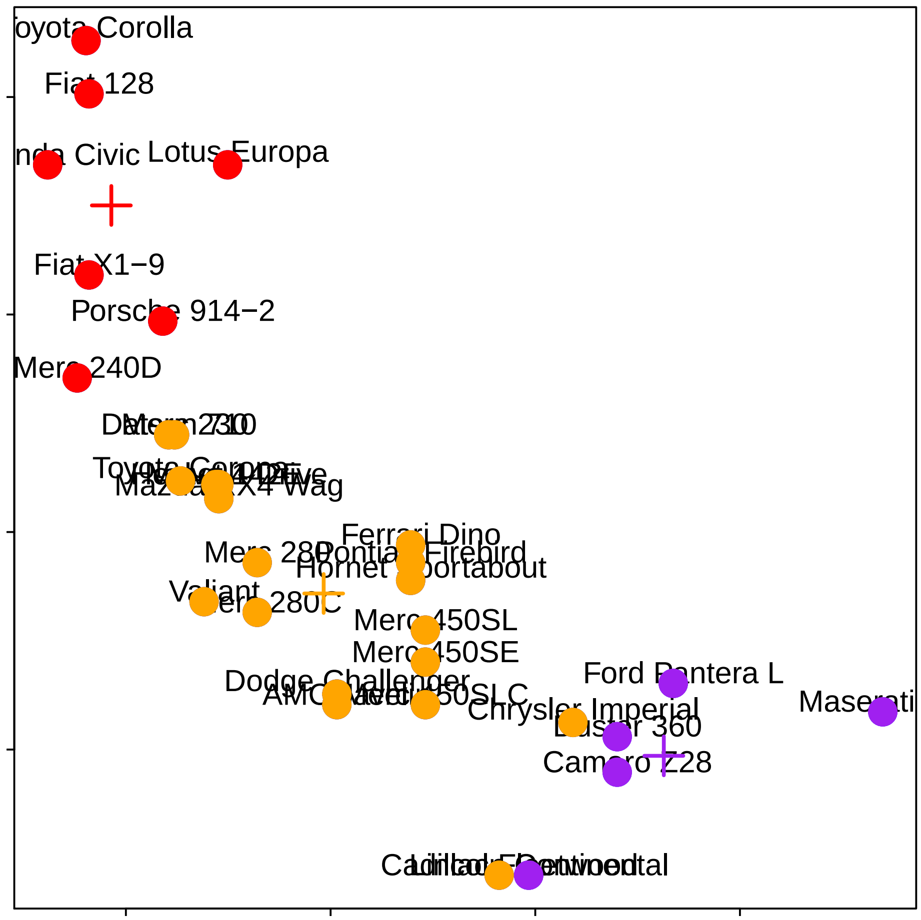
- – ’ ‘
- – ’ ‘

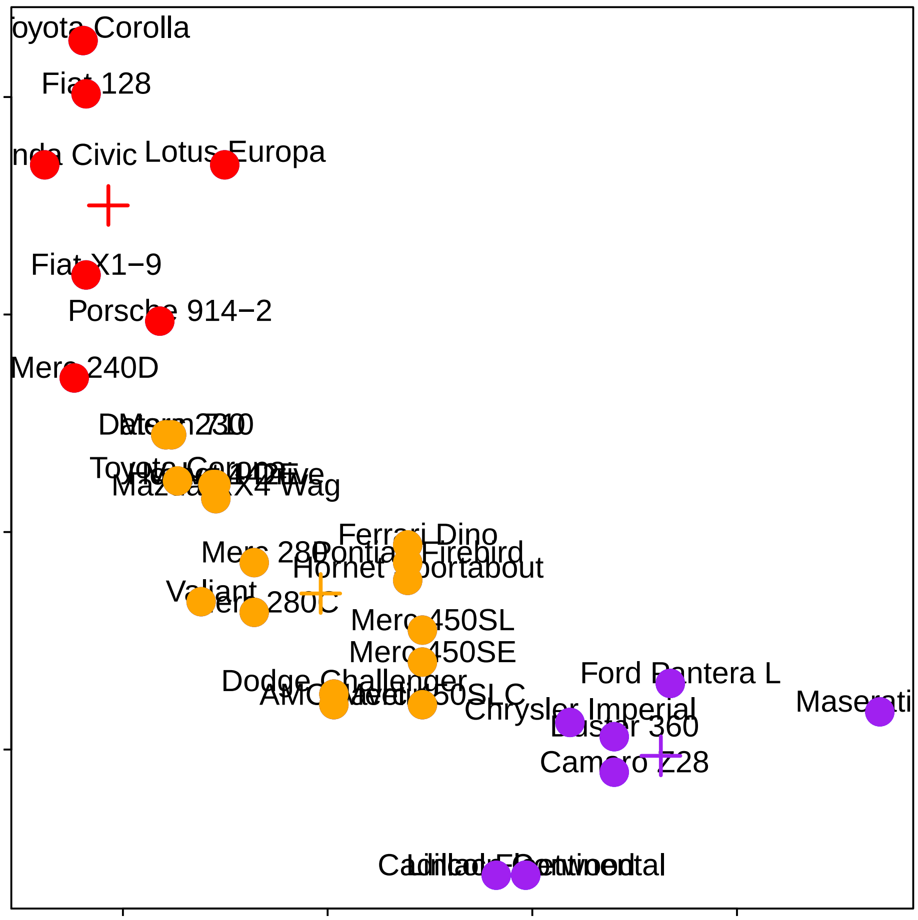
```
x<-scale(mtcars$hp[-1]);y<-scale(mtcars$mpg[-1])
plot(x,y,col="blue",pch=19,cex=2)
text(x+0.05,y+0.05,labels=labelCar)
```

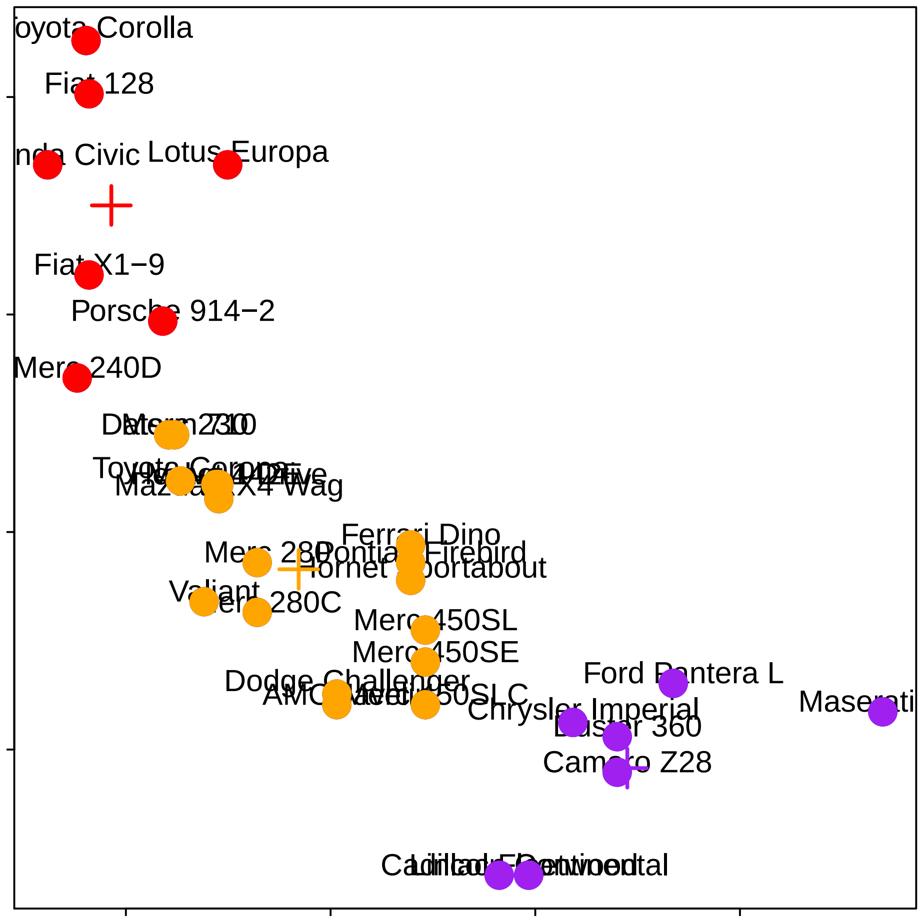



Fiat 128
Lotus Europa
Fiat X1−9
Porsche 914−2
Merc 240D
Mazda RX4 Wag
Ferrari Dino
Hornet Sportabout
Valiant
Merc 280C
Merc 450SL
Merc 450SE
Dodge Challenger
Ford Pantera L
Chrysler Imperial
Duster 360
Camaro Z28
Maserati


Toyota Corolla
Fiat 128
Honda Civic
Lotus Europa
Fiat X1−9
Porsche 914−2
Merc 240D
Ferrari Dino
Pontiac Firebird
Hornet Sportabout
Merc 450SL
Merc 450SE
Ford Pantera L
Maserati
Chrysler Imperial
Duster 360
Camaro Z28
Merc 280C
Valiant
Dodge Challenger
Mazda RX4 Wag
Lincoln Continental
Cadillac Fleetwood


Toyota Corolla
Fiat 128
Honda Civic
Lotus Europa
Fiat X1−9
Porsche 914−2
Merc 240D
Datsun 710
Merc 230
Toyota Corona
Volvo 142E
Mazda RX4
Mazda RX4 Wag
Ferrari Dino
Merc 280
Pontiac Firebird
Hornet Sportabout
Valiant
Merc 280C
Merc 450SL
Merc 450SE
Ford Pantera L
Dodge Challenger
AMC Javelin
Merc 450SLC
Maserati
Chrysler Imperial
Duster 360
Camaro Z28
Cadillac Fleetwood
Lincoln Continental


oyota Corolla
Fiat 128
nda Civic
Lotus Europa
Fiat X1−9
Porsche 914−2
Merc 240D
Mazda RX4 Wag
Ferrari Dino
Pontiac Firebird
Hornet Sportabout
Valiant
Merc 450SL
Merc 450SE
Dodge Challenger
Ford Pantera L
Maserati
Chrysler Imperial
Duster 360
Camaro Z28



`kmeans()`

- Important parameters: `x`, `centers`, `iter.max`, `nstart`

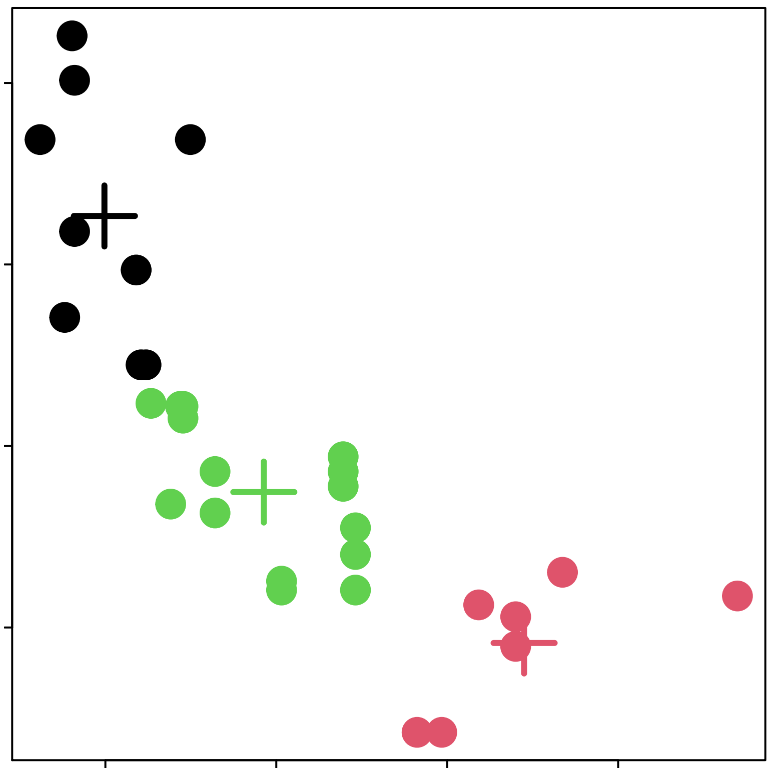
```
dataFrame <- data.frame(x,y)
kmeansObj <- kmeans(dataFrame,centers=3)
names(kmeansObj)
```

```
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
kmeansObj$cluster
```

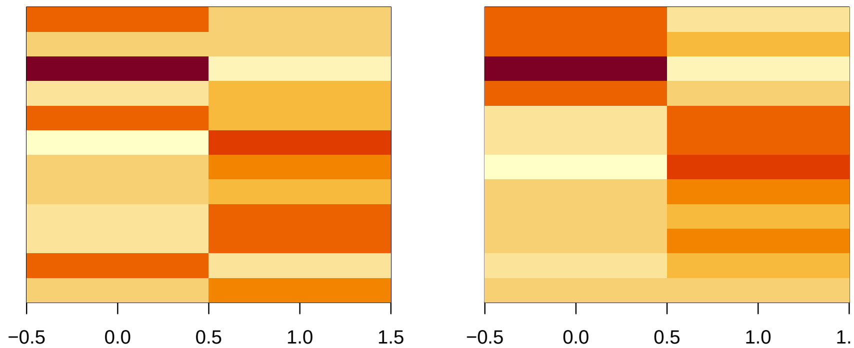
```
##  [1] 3 1 3 3 3 2 1 1 3 3 3 3 3 2 2 2 1 1 1 3 3 3 2 3 1 1 1 2 3 2 3
```

```
par(mar=rep(0.2,4))
plot(x,y,col=kmeansObj$cluster,pch=19,cex=2)
points(kmeansObj$centers,col=1:3,pch=3,cex=3,lwd=3)
```



Heatmaps

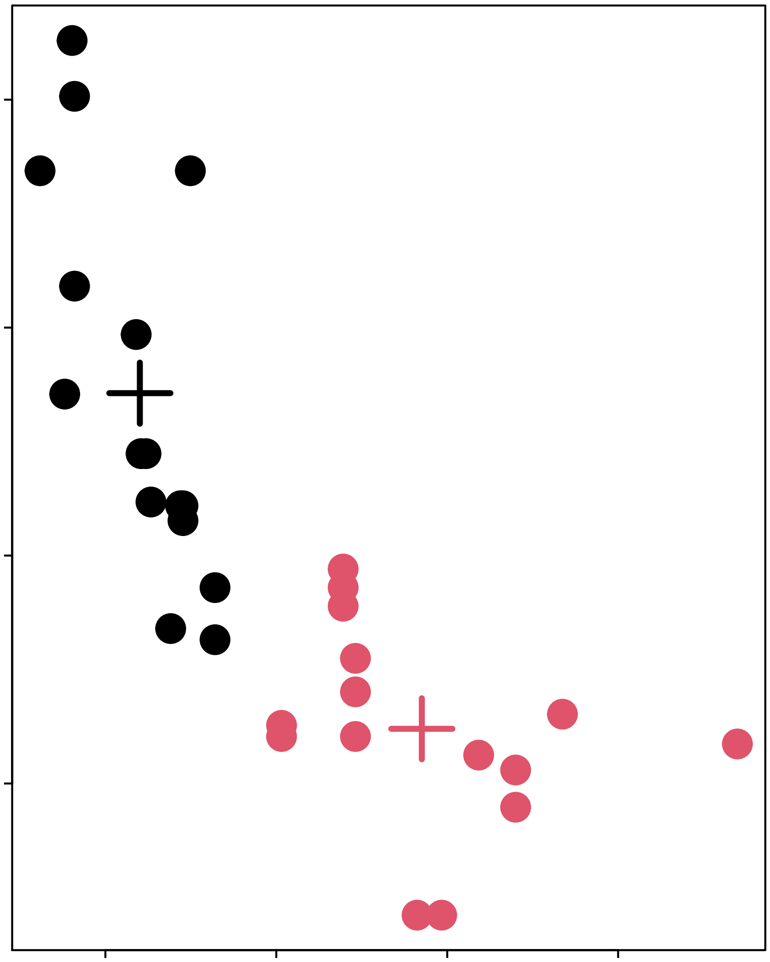
```
set.seed(1234)
dataMatrix <- as.matrix(dataFrame)[sample(1:12),]
kmeansObj <- kmeans(dataMatrix,centers=3)
par(mfrow=c(1,2), mar = c(2, 4, 0.1, 0.1))
image(t(dataMatrix)[,nrow(dataMatrix):1],yaxt="n")
image(t(dataMatrix)[,order(kmeansObj$cluster)],yaxt="n")
```



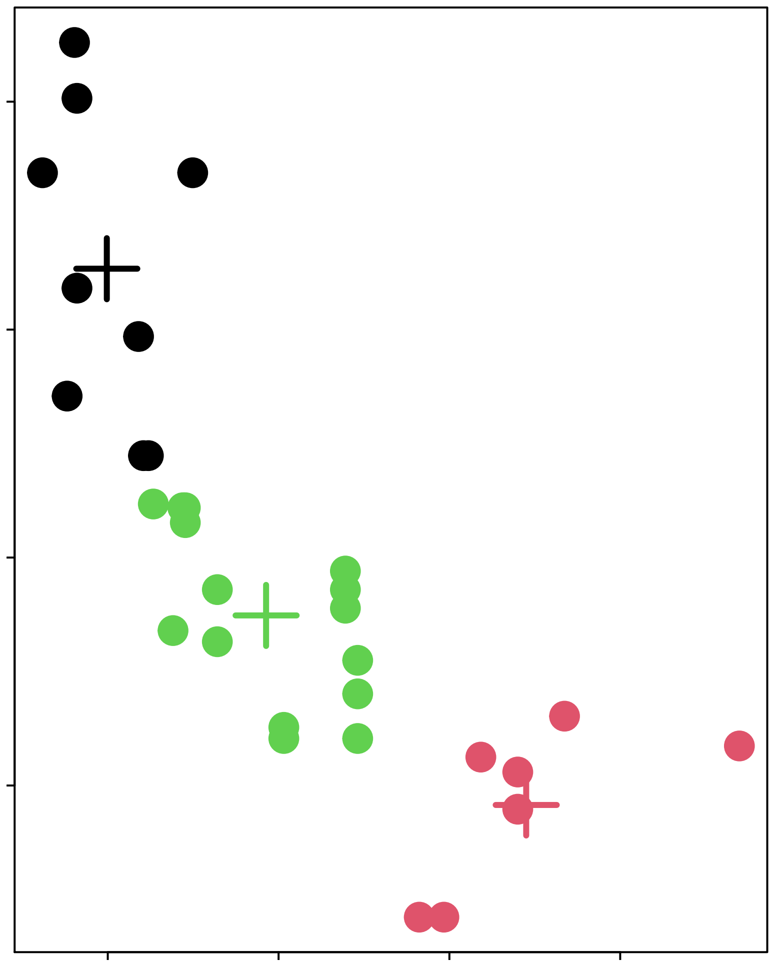
K-means

- \# of clusters
  - / /
  - cross validation/information theory
  - Determining the number of clusters
- K-means
  - \# of clusters
  - \# of iterations

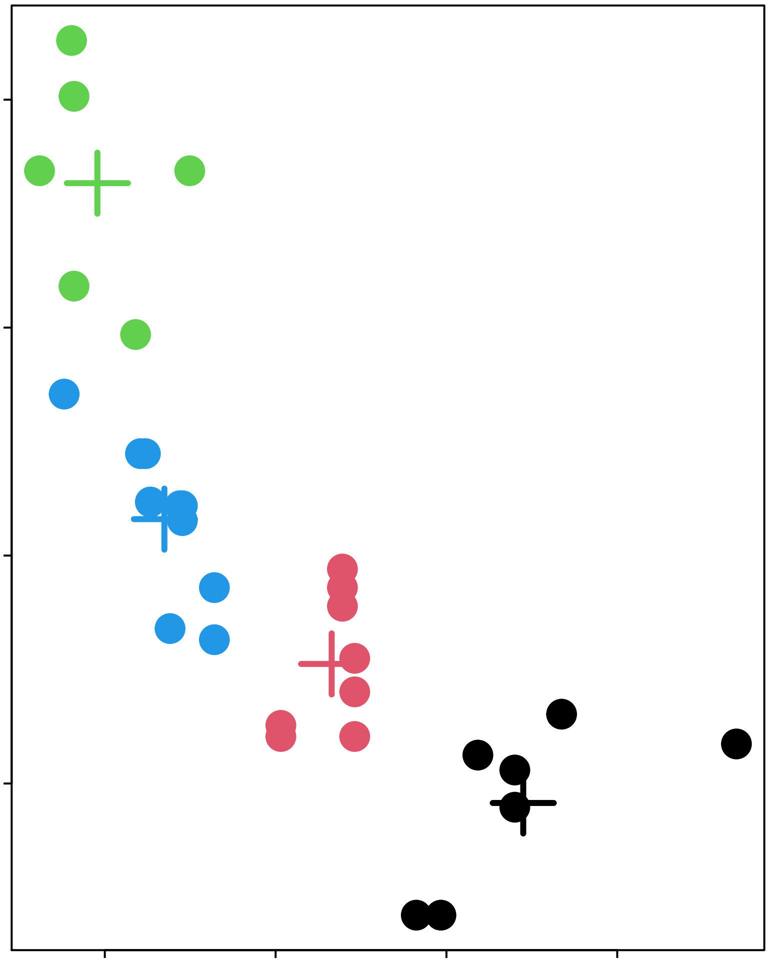
`kmeans()`, k=2



`kmeans()`, k=3



`kmeans()`, k=4



10.5 Association Rules

`(Market Basket Analysis)` **Apriori** (Boolean association rules) R `arules`(Hahsler et al., 2019)

```
# Load the libraries
if (!require('arules')){
  install.packages("arules");
  library(arules) #for Apriori
}
```

```
if (!require('datasets')){
  install.packages("datasets");
  library(datasets) #for Groceries data
}
data(Groceries) # Load the data set
Groceries@data@Dim #169   9835
```

```
## [1]  169 9835
```

	A	B	C	D	E	F	G	
1	citrus fruit	semi-finished	margarine	ready soups				
2	tropical fruit	yogurt	coffee					
3	whole milk							
4	pip fruit	yogurt	cream chees	meat spreads				
5	other vegeta	whole milk	condensed m	long life bakery product				
6	whole milk	butter	yogurt	rice	abrasive cleaner			
7	rolls/buns							
8	other vegeta	UHT-milk	rolls/buns	bottled beer	liquor (appetizer)			
9	pot plants							
10	whole milk	cereals						
11	tropical fruit	other vegeta	white bread	bottled wate	chocolate			
12	citrus fruit	tropical fruit	whole milk	butter	curd	yogurt	flour	bott
13	beef							
14	frankfurter	rolls/buns	soda					
15	chicken	tropical fruit						
16	butter	sugar	fruit/vegetab	newspapers				
17	fruit/vegetable juice							
18	packaged fruit/vegetables							
19	chocolate							
20	specialty bar							
21	other vegetables							
22	butter milk	pastry						
23	whole milk							
24	tropical fruit	cream chees	processed ch	detergent	newspapers			
25	tropical fruit	root vegetab	other vegeta	frozen desse	rolls/buns	flour	sweet spread	salty
26	bottled wate	canned beer						
27	yogurt							
28	sausage	rolls/buns	soda	chocolate				
29	other vegetables							
30	brown bread	soda	fruit/vegetab	canned beer	newspapers	shopping bags		
31	yogurt	beverages	bottled wate	specialty bar				
32	hamburger n	other vegeta	rolls/buns	spices	bottled wate	hygiene artic	napkins	

arules apriori apriori

```r
# Get the rules
rules <- apriori(Groceries, # data= Groceries
                 parameter = list(supp = 0.001, conf = 0.8), #
                 control = list(verbose=F)) #  output
options(digits=2) # Only 2 digits
inspect(rules[1:5]) # Show the top 5 rules
```

```
##     lhs                        rhs            support confidence coverage lift
## [1] {liquor,red/blush wine} => {bottled beer} 0.0019  0.90       0.0021   11.2
## [2] {curd,cereals}          => {whole milk}   0.0010  0.91       0.0011    3.6
## [3] {yogurt,cereals}        => {whole milk}   0.0017  0.81       0.0021    3.2
## [4] {butter,jam}            => {whole milk}   0.0010  0.83       0.0012    3.3
## [5] {soups,bottled beer}    => {whole milk}   0.0011  0.92       0.0012    3.6
##     count
## [1] 19
## [2] 10
## [3] 17
## [4] 10
## [5] 11
```

=>

- Support:
- Confidence: A B
- Lift: /
  - lift=1: items on the left and right are independent.

confidence

```r
rules<-sort(rules, by="confidence", decreasing=TRUE) # confidence
inspect(rules[1:5]) # Show the top 5 rules
```

```
##     lhs                     rhs          support confidence coverage lift count
## [1] {rice,
##      sugar}              => {whole milk}  0.0012          1   0.0012  3.9    12
## [2] {canned fish,
##      hygiene articles}   => {whole milk}  0.0011          1   0.0011  3.9    11
## [3] {root vegetables,
##      butter,
##      rice}               => {whole milk}  0.0010          1   0.0010  3.9    10
## [4] {root vegetables,
```

```
##       whipped/sour cream,
##       flour}              => {whole milk}  0.0017          1   0.0017  3.9    17
## [5] {butter,
##       soft cheese,
##       domestic eggs}      => {whole milk}  0.0010          1   0.0010  3.9    10
```

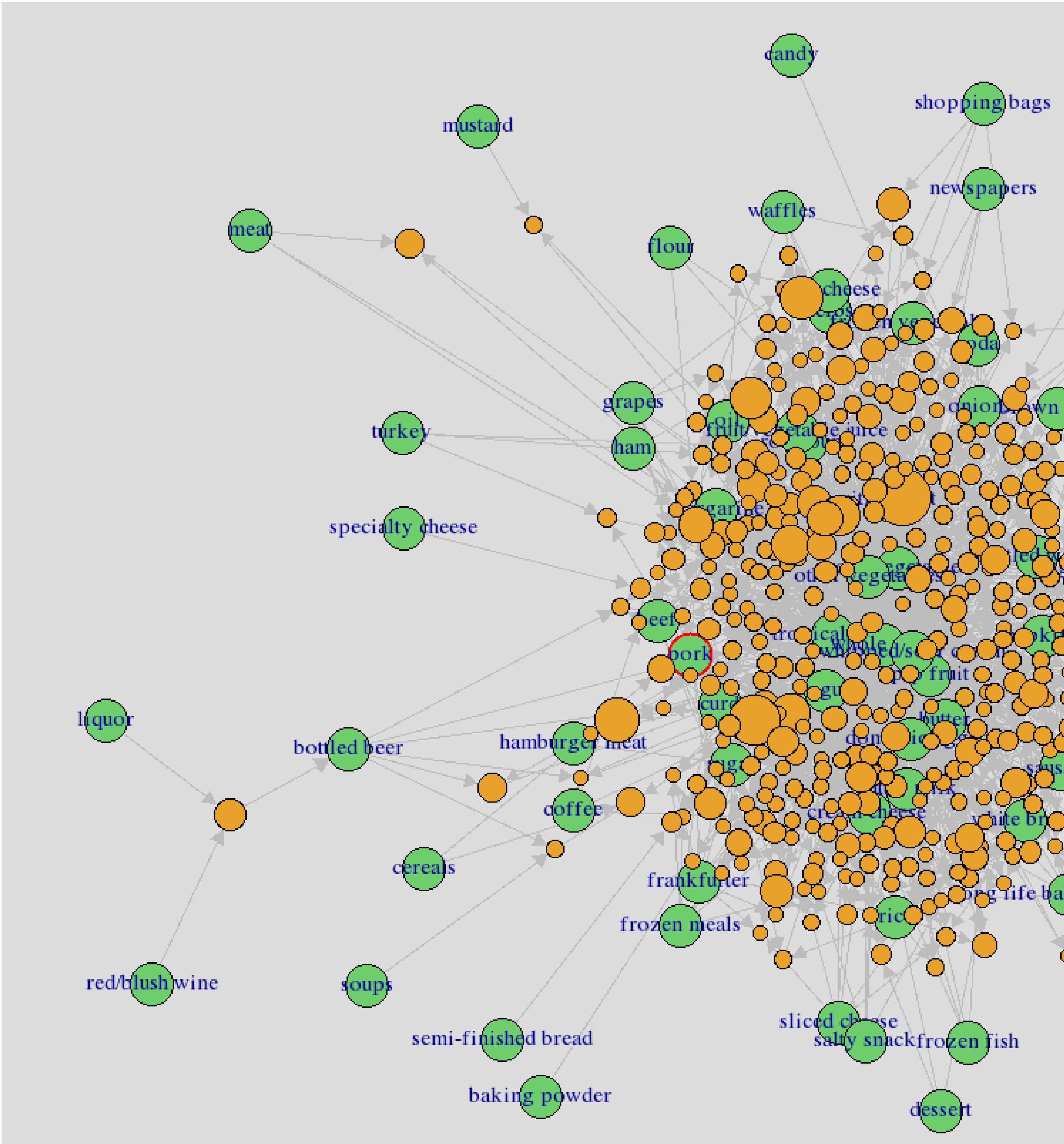
```
rulesR<-apriori(data=Groceries, parameter=list(supp=0.001,conf = 0.08),
        appearance = list(default="lhs",rhs="whole milk"), #
        control = list(verbose=F)) #  output
rulesR<-sort(rulesR, decreasing=TRUE,by="confidence") # confidence
inspect(rulesR[1:5]) # Show the top 5 rules
```

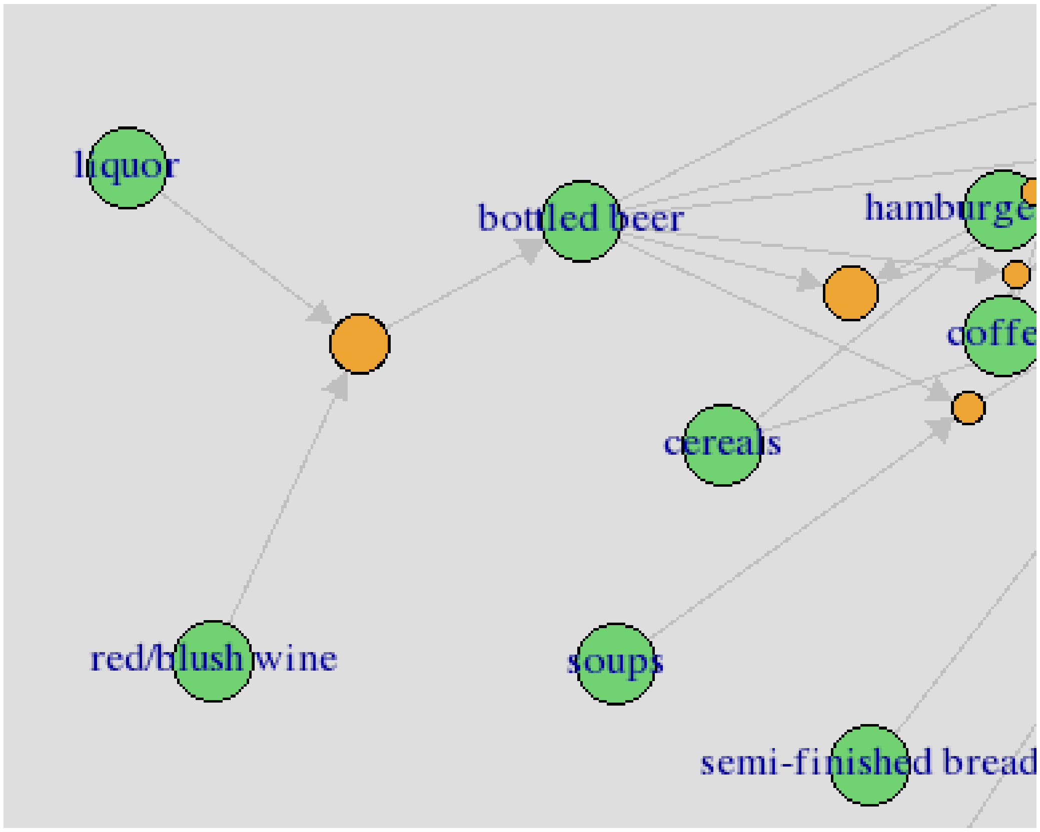
```
##     lhs                        rhs          support confidence coverage lift count
## [1] {rice,
##       sugar}              => {whole milk}  0.0012          1   0.0012  3.9    12
## [2] {canned fish,
##       hygiene articles}   => {whole milk}  0.0011          1   0.0011  3.9    11
## [3] {root vegetables,
##       butter,
##       rice}               => {whole milk}  0.0010          1   0.0010  3.9    10
## [4] {root vegetables,
##       whipped/sour cream,
##       flour}              => {whole milk}  0.0017          1   0.0017  3.9    17
## [5] {butter,
##       soft cheese,
##       domestic eggs}      => {whole milk}  0.0010          1   0.0010  3.9    10
```

```
rulesL<-apriori(data=Groceries, parameter=list(supp=0.001,conf = 0.15,minlen=2),
        appearance = list(default="rhs",lhs="whole milk"), #
        control = list(verbose=F)) #  output
rulesL<-sort(rulesL, decreasing=TRUE,by="confidence") # confidence
inspect(rulesL[1:5]) # Show the top 5 rules
```

```
##     lhs             rhs                  support confidence coverage lift count
## [1] {whole milk} => {other vegetables} 0.075   0.29       0.26     1.5  736
## [2] {whole milk} => {rolls/buns}       0.057   0.22       0.26     1.2  557
## [3] {whole milk} => {yogurt}           0.056   0.22       0.26     1.6  551
## [4] {whole milk} => {root vegetables}  0.049   0.19       0.26     1.8  481
## [5] {whole milk} => {tropical fruit}   0.042   0.17       0.26     1.6  416
```

```r
if (!require('arulesViz')){
  install.packages("arulesViz");
  library(arulesViz)
}
#Mac->http://planspace.org/2013/01/17/fix-r-tcltk-dependency-problem-on-mac/
plot(rules,method="graph",interactive=TRUE,shading=NA) #
```


candy
shopping bags
mustard
newspapers
waffles
meat
flour
grapes
turkey
ham
specialty cheese
pork
liquor
bottled beer
hamburger meat
coffee
cereals
frankfurter
frozen meals
red/blush wine
soups
semi-finished bread
baking powder
sliced cheese
salty snack
frozen fish
dessert



10.6 Open Source Packages

10.6.1 Prophet

Prophet Facebook 2017 Prophet for R

- C/C++ Tool
  - R Tools on Windows
  - Command Line Tools on OS X

```
install.packages('prophet')
```

R API

```
library(prophet)
library(dplyr)
df <- read.csv('https://raw.githubusercontent.com/facebookincubator/prophet/master/examples/examp
    mutate(y = log(y))
m <- prophet(df)
future <- make_future_dataframe(m, periods = 365)
tail(future)
forecast <- predict(m, future)
tail(forecast[c('ds', 'yhat', 'yhat_lower', 'yhat_upper')])
plot(m, forecast)
prophet_plot_components(m, forecast)
```

Prophet

10.6.2 TensorFlow

- Python 3.5.3 **64 bit**
    - Windows x86-64 executable installer
- TensorFlow 1.0.1
    - pip3 install –upgrade tensorflow
    - pip3 install –upgrade tensorflow-gpu
- C/C++ Tool
    - R Tools on Windows
    - Command Line Tools on OS X
- tensorflow package for R

```
devtools::install_github("rstudio/tensorflow")
```

TensorFlow for R

- Locating TensorFlow (optional)
- Hello World

```
library(tensorflow)
sess = tf$Session()
hello <- tf$constant('Hello, TensorFlow!')
sess$run(hello)
```

10.6.3 MXNet

Amazon Install MXNet for R MXNet for R Tutorials

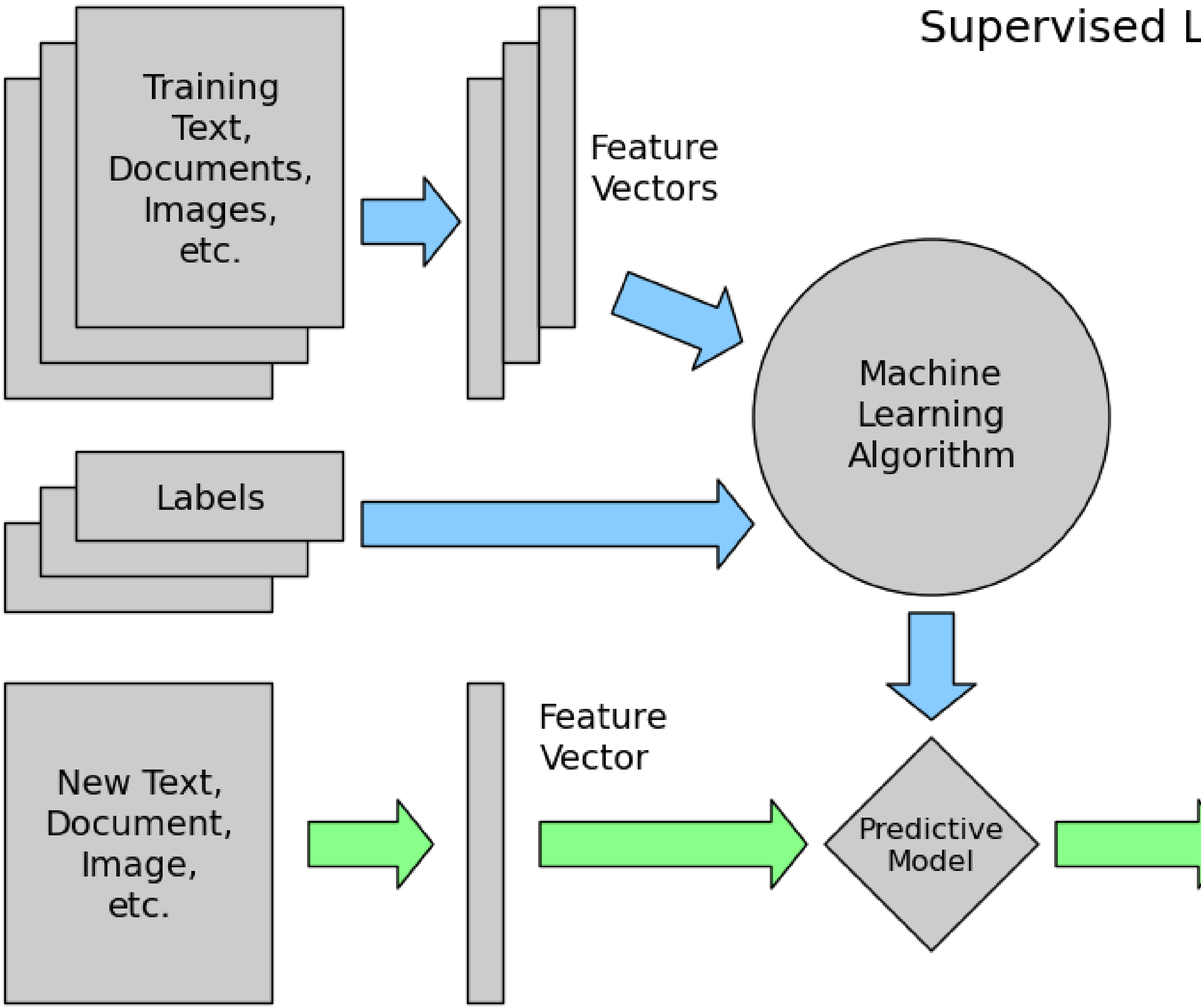
MXNet for R

```
install.packages("drat", repos="https://cran.rstudio.com")
drat:::addRepo("dmlc")
install.packages("mxnet")
```

10.7

- Training set, Development set:
- Test set, Validation set:

Training set Test set 2/3 `Training set` 1/3 `Test set`



10.7.1 Regression

NBA

```r
# SportsAnalytics package
if (!require('SportsAnalytics')){
    install.packages("SportsAnalytics")
```

```
    library(SportsAnalytics)
}
# 2015-2016
NBA1516<-fetch_NBAPlayerStatistics("15-16")
NBA1516<-NBA1516[complete.cases(NBA1516),]
```

- Training set
- Test set
- …..
- Training set Training set
- Test set

```
sample(1:10,3) #  1 10
```

```
## [1] 8 3 4
```

```
sample(1:nrow(NBA1516),nrow(NBA1516)/3) #       1/3
```

```
##   [1]  93 122 389  66 175 424 379 468 304 108 131 343  41 115 228 328 416 298
##  [19] 299 258 117  79 182 305 358 184 307 390 452 221 224  49 313 136 282 145
##  [37] 123 264 234  96  22 291 297 208 465 342  57  10 406 248 365 153 431  83
##  [55] 245 426 218 215 326 276 169  71  61 352 417 383 155 460 467  60  36 375
##  [73]  19 137 126 158 319 116 440 102 214 314 448  85 392 160  77  17 401 262
##  [91] 130 181 267 316 356 163 461 277 396 134 265 403 249 435  40  29 425 185
## [109] 294  88 400 363 411 335  86 142 147 414 188 355  26 372 418  28 101 296
## [127] 323 408 359 189 196  84 422 250 388 281 380 471  30 428 354 444  80  73
## [145] 148  12 293 195 303 361 166 347 146 107 240  31   6 263
```

1/3 NBA Training Test set

```
NBA1516$Test<-F #
#  1/3 Test set
NBA1516[sample(1:nrow(NBA1516),nrow(NBA1516)/3),"Test"]<-T
# Training set : Test set
c(sum(NBA1516$Test==F),sum(NBA1516$Test==T))
```

```
## [1] 317 158
```

NBA1516$Test==F

```r
fit<-glm(TotalPoints~TotalMinutesPlayed+FieldGoalsAttempted+
              Position+ThreesAttempted+FreeThrowsAttempted,
               data =NBA1516[NBA1516$Test==F,])
summary(fit)$coefficients
```

```
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)            9.7517     7.8573    1.24  2.2e-01
## TotalMinutesPlayed    -0.0028     0.0078   -0.36  7.2e-01
## FieldGoalsAttempted    0.9921     0.0234   42.36 1.7e-130
## PositionPF           -14.5514     8.3559   -1.74  8.3e-02
## PositionPG           -34.5378     9.1477   -3.78  1.9e-04
## PositionSF           -14.2217     9.2792   -1.53  1.3e-01
## PositionSG           -25.6675     9.3777   -2.74  6.6e-03
## ThreesAttempted        0.1016     0.0315    3.23  1.4e-03
## FreeThrowsAttempted    0.7903     0.0390   20.28  1.2e-58
```

stepwise

```r
library(MASS)
## AIC    ,      direction = "backward"
##trace=FALSE:
finalModel_B<-stepAIC(fit,direction = "backward",trace=FALSE)
summary(finalModel_B)$coefficients
```

```
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)              8.70      7.275     1.2  2.3e-01
## FieldGoalsAttempted      0.99      0.017    56.6 4.3e-165
## PositionPF             -14.34      8.322    -1.7  8.6e-02
## PositionPG             -34.14      9.068    -3.8  2.0e-04
## PositionSF             -14.01      9.246    -1.5  1.3e-01
## PositionSG             -25.26      9.294    -2.7  6.9e-03
## ThreesAttempted          0.10      0.031     3.2  1.4e-03
## FreeThrowsAttempted      0.79      0.039    20.4  4.3e-59
```

stepwise

```r
## AIC    ,     direction = "forward"
finalModel_F<-stepAIC(fit,direction = "forward",trace=FALSE)
summary(finalModel_F)$coefficients
```

```
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)            9.7517     7.8573    1.24  2.2e-01
## TotalMinutesPlayed    -0.0028     0.0078   -0.36  7.2e-01
```

```
## FieldGoalsAttempted   0.9921     0.0234   42.36 1.7e-130
## PositionPF          -14.5514     8.3559   -1.74  8.3e-02
## PositionPG          -34.5378     9.1477   -3.78  1.9e-04
## PositionSF          -14.2217     9.2792   -1.53  1.3e-01
## PositionSG          -25.6675     9.3777   -2.74  6.6e-03
## ThreesAttempted       0.1016     0.0315    3.23  1.4e-03
## FreeThrowsAttempted   0.7903     0.0390   20.28  1.2e-58
```

stepwise

```r
## AIC    ,     direction = "both"
finalModel_Both<-stepAIC(fit,direction = "both",trace=FALSE)
summary(finalModel_Both)$coefficients
```

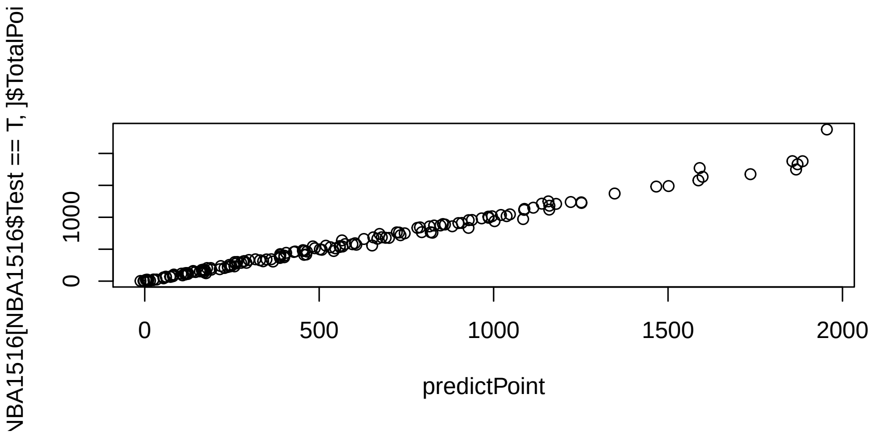
```
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)             8.70      7.275     1.2  2.3e-01
## FieldGoalsAttempted     0.99      0.017    56.6 4.3e-165
## PositionPF            -14.34      8.322    -1.7  8.6e-02
## PositionPG            -34.14      9.068    -3.8  2.0e-04
## PositionSF            -14.01      9.246    -1.5  1.3e-01
## PositionSG            -25.26      9.294    -2.7  6.9e-03
## ThreesAttempted         0.10      0.031     3.2  1.4e-03
## FreeThrowsAttempted     0.79      0.039    20.4  4.3e-59
```

Test set predict

```r
predictPoint<-predict(finalModel_Both, #Test==T, test data
                      newdata = NBA1516[NBA1516$Test==T,])
cor(x=predictPoint,y=NBA1516[NBA1516$Test==T,]$TotalPoints) #
```

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

```
plot(x=predictPoint,y=NBA1516[NBA1516$Test==T,]$TotalPoints)
```



10.7.2 Logistic Regression

Training Test set Level 2 -> / …

```
mydata <- read.csv("https://raw.githubusercontent.com/CGUIM-BigDataAnalysis/BigDataCGUIM/master/b
mydata$admit <- factor(mydata$admit) #     factor
mydata$rank <- factor(mydata$rank) #     factor
mydata$Test<-F #
mydata[sample(1:nrow(mydata),nrow(mydata)/3),"Test"]<-T #  1/3 Test set
c(sum(mydata$Test==F),sum(mydata$Test==T)) # Training set : Test set
```

```
## [1] 267 133
```

```
#  factor level:  Level 2  1  -->Level 2
mydata$admit<-factor(mydata$admit,levels=c(0,1))
```

```
# GRE:  , GPA:   , rank:
mylogit <- glm(admit ~ gre + gpa + rank,
               data = mydata[mydata$Test==F,], family = "binomial")
finalFit<-stepAIC(mylogit,direction = "both",trace=FALSE) #
summary(finalFit)
```

```
##
## Call:
```

```
## glm(formula = admit ~ gpa + rank, family = "binomial", data = mydata[mydata$Test ==
##     F, ])
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.578  -0.893  -0.632   1.085   2.146
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -4.022      1.437   -2.80  0.00514 **
## gpa            1.232      0.400    3.08  0.00206 **
## rank2         -0.641      0.387   -1.66  0.09783 .
## rank3         -1.440      0.427   -3.37  0.00074 ***
## rank4         -1.589      0.516   -3.08  0.00207 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 339.9  on 266  degrees of freedom
## Residual deviance: 309.8  on 262  degrees of freedom
## AIC: 319.8
##
## Number of Fisher Scoring iterations: 4
```

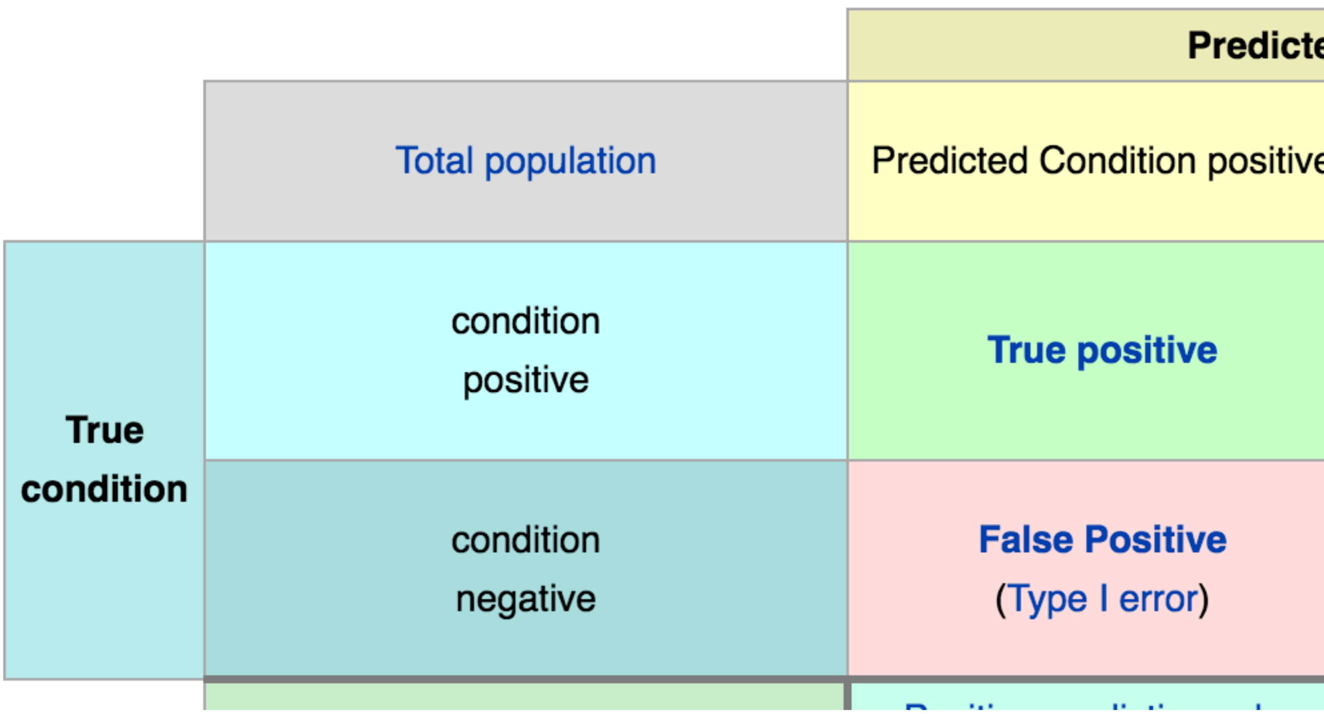
```r
AdmitProb<-predict(finalFit, #  Training set
                   newdata = mydata[mydata$Test==T,], #Test==T, test data
                   type="response") #
head(AdmitProb)
```

```
##    1    2   10   11   13   14
## 0.27 0.28 0.54 0.34 0.71 0.30
```

```r
table(AdmitProb<0.5,mydata[mydata$Test==T,]$admit) # row,column
```

```
##
##          0  1
##   FALSE 11  9
##   TRUE  84 29
```

- Sensitivity
- Specificity
- Positive Predictive Value (PPV)
- Negative Predictive Value (NPV)



		Predicte
	Total population	Predicted Condition positive
True condition	condition positive	True positive
	condition negative	False Positive (Type I error)

- TP:
- TN:
- FP:
- FN:

		Patients with bowel cancer (as confirmed on endoscopy)		
		Condition positive	Condition negative	
Fecal occult blood screen test outcome	Test outcome positive	**True positive** (TP) = 20	**False positive** (FP) = 180	Pos = = =
	Test outcome negative	**False negative** (FN) = 10	**True negative** (TN) = 1820	Neg = = ≈
		Sensitivity = TP / (TP + FN) = 20 / (20 + 10) ≈ **67%**	Specificity = TN / (FP + TN) = 1820 / (180 + 1820) = **91%**	

- Sensitivity
- Specificity
- Positive Predictive Value (PPV)
- Negative Predictive Value (NPV)

```
table(AdmitProb<0.5,mydata[mydata$Test==T,]$admit) # row,column
```

```
##
```

```
##          0  1
##   FALSE 11  9
##   TRUE  84 29
```

		Patients with bowel cancer (as confirmed on endoscopy)		
		Condition positive	Condition negative	
Fecal occult blood screen test outcome	Test outcome positive	True positive (TP) = 20	False positive (FP) = 180	Positive pred = TP / (TP = 20 / (20 + = 10%
	Test outcome negative	False negative (FN) = 10	True negative (TN) = 1820	Negative pred = TN / (FN = 1820 / (1 ≈ 99.5%
		Sensitivity = TP / (TP + FN) = 20 / (20 + 10) ≈ 67%	Specificity = TN / (FP + TN) = 1820 / (180 + 1820) = 91%	

```
AdmitProb<-predict(finalFit,
                   newdata = mydata[mydata$Test==T,], #Test==T, test data
                   type="response") #
AdmitAns<-factor(ifelse(AdmitProb>0.5,1,0),levels=c(0,1))
str(AdmitAns)
```

```
##  Factor w/ 2 levels "0","1": 1 1 2 1 2 1 1 1 1 1 ...
```

```
##  - attr(*, "names")= chr [1:133] "1" "2" "10" "11" ...
```

```
library(caret) # install.packages("caret") #   packages
sensitivity(AdmitAns,mydata[mydata$Test==T,]$admit)
```

```
## [1] 0.88
```

```
specificity(AdmitAns,mydata[mydata$Test==T,]$admit)
```

```
## [1] 0.24
```

```
posPredValue(AdmitAns,mydata[mydata$Test==T,]$admit)
```

```
## [1] 0.74
```

```
negPredValue(AdmitAns,mydata[mydata$Test==T,]$admit)
```

```
## [1] 0.45
```

10.7.3 Decision Trees

/ / / / -

```
if (!require('rpart')){
    install.packages("rpart"); library(rpart)
}
DT<-rpart(Position~Blocks+TotalRebounds+ThreesMade+Assists+Steals,
          data=NBA1516[NBA1516$Test==F,]) #   Training set
#   PG    SG    SF    PF   C
DT
```

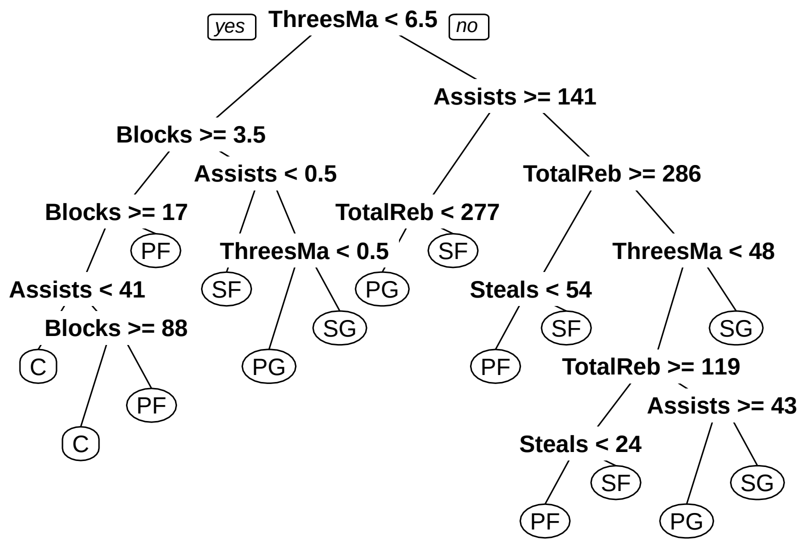
```
## n= 317
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 317 240 PF (0.16 0.23 0.21 0.18 0.22)
##     2) ThreesMade< 6.5 121  76 C (0.37 0.36 0.091 0.091 0.091)
##       4) Blocks>=3.5 78  37 C (0.53 0.44 0.013 0.026 0)
```

```
##         8) Blocks>=17 55  20 C (0.64 0.35 0.018 0 0)
##          16) Assists< 40 16   1 C (0.94 0.063 0 0 0) *
##          17) Assists>=40 39  19 C (0.51 0.46 0.026 0 0)
##            34) Blocks>=88 12   1 C (0.92 0.083 0 0 0) *
##            35) Blocks< 88 27  10 PF (0.33 0.63 0.037 0 0) *
##         9) Blocks< 17 23   8 PF (0.26 0.65 0 0.087 0) *
##       5) Blocks< 3.5 43  32 SG (0.093 0.21 0.23 0.21 0.26)
##        10) Assists< 0.5 9   4 SF (0 0.33 0.11 0.56 0) *
##        11) Assists>=0.5 34  23 SG (0.12 0.18 0.26 0.12 0.32)
##          22) ThreesMade< 0.5 13   8 PG (0.31 0.23 0.38 0 0.077) *
##          23) ThreesMade>=0.5 21  11 SG (0 0.14 0.19 0.19 0.48) *
##     3) ThreesMade>=6.5 196 140 SG (0.031 0.15 0.29 0.23 0.3)
##       6) Assists>=1.4e+02 75  32 PG (0.027 0.04 0.57 0.15 0.21)
##        12) TotalRebounds< 2.8e+02 48   9 PG (0 0 0.81 0 0.19) *
##        13) TotalRebounds>=2.8e+02 27  16 SF (0.074 0.11 0.15 0.41 0.26) *
##       7) Assists< 1.4e+02 121  79 SG (0.033 0.22 0.11 0.29 0.35)
##        14) TotalRebounds>=2.9e+02 29  13 PF (0.069 0.55 0 0.34 0.034)
##          28) Steals< 54 16   3 PF (0.12 0.81 0 0.062 0) *
##          29) Steals>=54 13   4 SF (0 0.23 0 0.69 0.077) *
##        15) TotalRebounds< 2.9e+02 92  51 SG (0.022 0.12 0.14 0.27 0.45)
##          30) ThreesMade< 48 62  41 SG (0.032 0.15 0.18 0.31 0.34)
##            60) TotalRebounds>=1.2e+02 21   9 SF (0.048 0.24 0 0.57 0.14)
##             120) Steals< 24 8   3 PF (0.12 0.62 0 0.12 0.12) *
##             121) Steals>=24 13   2 SF (0 0 0 0.85 0.15) *
##            61) TotalRebounds< 1.2e+02 41  23 SG (0.024 0.098 0.27 0.17 0.44)
##             122) Assists>=43 14   5 PG (0 0.071 0.64 0 0.29) *
##             123) Assists< 43 27  13 SG (0.037 0.11 0.074 0.26 0.52) *
##          31) ThreesMade>=48 30  10 SG (0 0.067 0.067 0.2 0.67) *
```

/ / / /   -

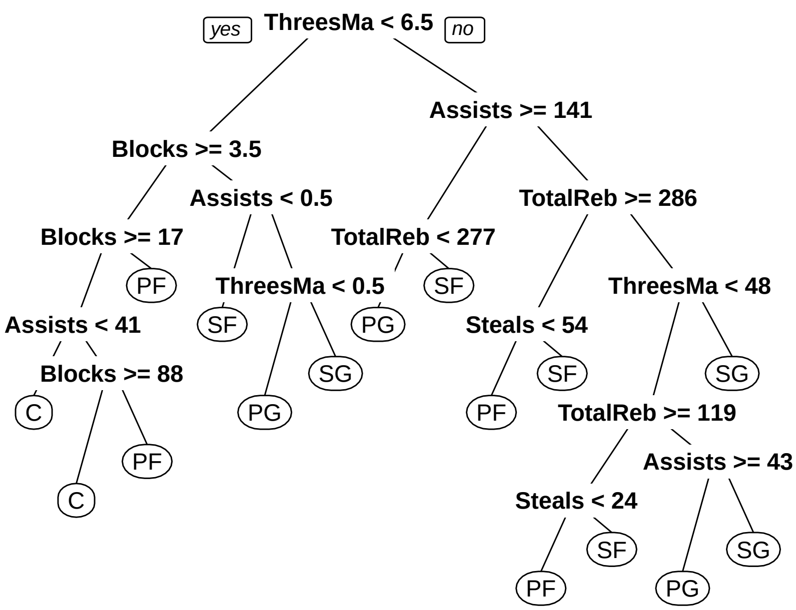
plot()   rpart.plot package prp()

```
if (!require('rpart.plot')){
  install.packages("rpart.plot");
  library(rpart.plot)
}
prp(DT)        #
```


yes
ThreesMa < 6.5
no
Assists >= 141
Blocks >= 3.5
Assists < 0.5
TotalReb >= 286
Blocks >= 17
TotalReb < 277
PF
ThreesMa < 0.5
SF
ThreesMa < 48
Assists < 41
SF
PG
Steals < 54
Blocks >= 88
SG
SF
SG
C
PG
PF
TotalReb >= 119
PF
Assists >= 43
C
Steals < 24
SF
SG
PF
PG

/ / / / -

```
prp(DT)
```



–

```
posPred<-predict(DT,newdata= NBA1516[NBA1516$Test==T,]) #Test==T, test data
#  class probabilities, type = "prob"
head(posPred)
```

```
##         C   PF    PG    SF    SG
## 4  0.000 0.00 0.812 0.000 0.188
## 10 0.000 0.23 0.000 0.692 0.077
## 15 0.037 0.11 0.074 0.259 0.519
## 22 0.261 0.65 0.000 0.087 0.000
## 30 0.125 0.62 0.000 0.125 0.125
## 36 0.074 0.11 0.148 0.407 0.259
```

–

```
result<-cbind(round(posPred,digits = 2),
              NBA1516[NBA1516$Test==T,]$Name,
      as.character(NBA1516[NBA1516$Test==T,]$Position))
head(result)
```

```
##     C      PF      PG      SF      SG
```

```
## 4  "0"    "0"    "0.81" "0"    "0.19" "Arron Afflalo"  "SG"
## 10 "0"    "0.23" "0"    "0.69" "0.08" "Tony Allen"     "SG"
## 15 "0.04" "0.11" "0.07" "0.26" "0.52" "James Anderson" "SG"
## 22 "0.26" "0.65" "0"    "0.09" "0"    "Joel Anthony"   "C"
## 30 "0.12" "0.62" "0"    "0.12" "0.12" "Luke Babbitt"   "SF"
## 36 "0.07" "0.11" "0.15" "0.41" "0.26" "Matt Barnes"    "SF"
```

– -2

```
posPredC<-predict(DT,newdata= NBA1516[NBA1516$Test==T,],type = "class")
# type = "class"
head(posPredC)
```

```
##  4 10 15 22 30 36
## PG SF SG PF PF SF
## Levels: C PF PG SF SG
```

– -2

```
resultC<-cbind(as.character(posPredC),NBA1516[NBA1516$Test==T,]$Name,
      as.character(NBA1516[NBA1516$Test==T,]$Position))
head(resultC)
```

```
##      [,1] [,2]             [,3]
## [1,] "PG" "Arron Afflalo"  "SG"
## [2,] "SF" "Tony Allen"     "SG"
## [3,] "SG" "James Anderson" "SG"
## [4,] "PF" "Joel Anthony"   "C"
## [5,] "PF" "Luke Babbitt"   "SF"
## [6,] "SF" "Matt Barnes"    "SF"
```

10.8 Case Study

-
- Sonar, Mines vs. Rocks

1.1:

```r
#install.packages("mlbench") # package  dataset
library(mlbench)
data(Sonar)
str(Sonar) #          factor
```

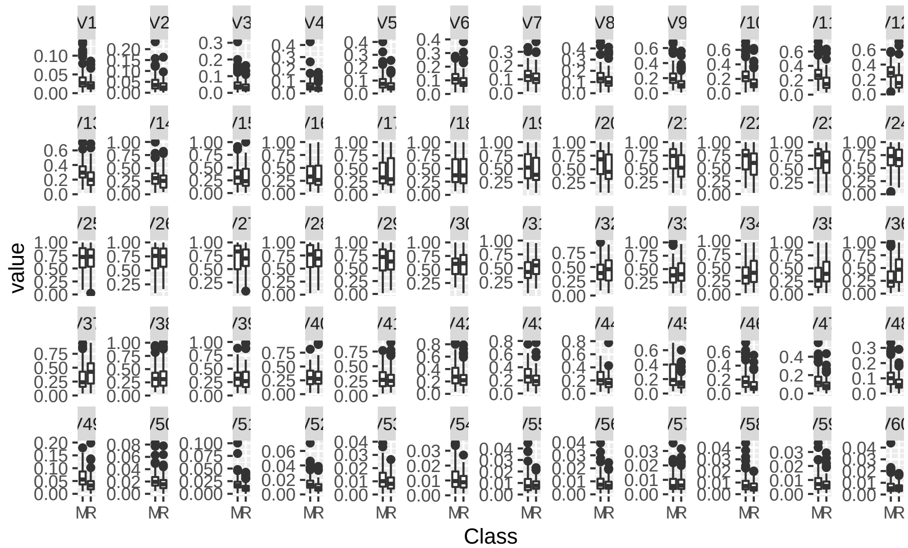
```
## 'data.frame': 208 obs. of  61 variables:
##  $ V1   : num  0.02 0.0453 0.0262 0.01 0.0762 0.0286 0.0317 0.0519 0.0223 0.0164 ...
##  $ V2   : num  0.0371 0.0523 0.0582 0.0171 0.0666 0.0453 0.0956 0.0548 0.0375 0.0173 ...
##  $ V3   : num  0.0428 0.0843 0.1099 0.0623 0.0481 ...
##  $ V4   : num  0.0207 0.0689 0.1083 0.0205 0.0394 ...
##  $ V5   : num  0.0954 0.1183 0.0974 0.0205 0.059 ...
##  $ V6   : num  0.0986 0.2583 0.228 0.0368 0.0649 ...
##  $ V7   : num  0.154 0.216 0.243 0.11 0.121 ...
##  $ V8   : num  0.16 0.348 0.377 0.128 0.247 ...
##  $ V9   : num  0.3109 0.3337 0.5598 0.0598 0.3564 ...
##  $ V10  : num  0.211 0.287 0.619 0.126 0.446 ...
##  $ V11  : num  0.1609 0.4918 0.6333 0.0881 0.4152 ...
##  $ V12  : num  0.158 0.655 0.706 0.199 0.395 ...
##  $ V13  : num  0.2238 0.6919 0.5544 0.0184 0.4256 ...
##  $ V14  : num  0.0645 0.7797 0.532 0.2261 0.4135 ...
##  $ V15  : num  0.066 0.746 0.648 0.173 0.453 ...
##  $ V16  : num  0.227 0.944 0.693 0.213 0.533 ...
##  $ V17  : num  0.31 1 0.6759 0.0693 0.7306 ...
##  $ V18  : num  0.3 0.887 0.755 0.228 0.619 ...
##  $ V19  : num  0.508 0.802 0.893 0.406 0.203 ...
##  $ V20  : num  0.48 0.782 0.862 0.397 0.464 ...
##  $ V21  : num  0.578 0.521 0.797 0.274 0.415 ...
##  $ V22  : num  0.507 0.405 0.674 0.369 0.429 ...
##  $ V23  : num  0.433 0.396 0.429 0.556 0.573 ...
##  $ V24  : num  0.555 0.391 0.365 0.485 0.54 ...
##  $ V25  : num  0.671 0.325 0.533 0.314 0.316 ...
##  $ V26  : num  0.641 0.32 0.241 0.533 0.229 ...
##  $ V27  : num  0.71 0.327 0.507 0.526 0.7 ...
##  $ V28  : num  0.808 0.277 0.853 0.252 1 ...
##  $ V29  : num  0.679 0.442 0.604 0.209 0.726 ...
##  $ V30  : num  0.386 0.203 0.851 0.356 0.472 ...
##  $ V31  : num  0.131 0.379 0.851 0.626 0.51 ...
##  $ V32  : num  0.26 0.295 0.504 0.734 0.546 ...
##  $ V33  : num  0.512 0.198 0.186 0.612 0.288 ...
##  $ V34  : num  0.7547 0.2341 0.2709 0.3497 0.0981 ...
##  $ V35  : num  0.854 0.131 0.423 0.395 0.195 ...
##  $ V36  : num  0.851 0.418 0.304 0.301 0.418 ...
##  $ V37  : num  0.669 0.384 0.612 0.541 0.46 ...
##  $ V38  : num  0.61 0.106 0.676 0.881 0.322 ...
##  $ V39  : num  0.494 0.184 0.537 0.986 0.283 ...
```

```
##  $ V40  : num  0.274 0.197 0.472 0.917 0.243 ...
##  $ V41  : num  0.051 0.167 0.465 0.612 0.198 ...
##  $ V42  : num  0.2834 0.0583 0.2587 0.5006 0.2444 ...
##  $ V43  : num  0.282 0.14 0.213 0.321 0.185 ...
##  $ V44  : num  0.4256 0.1628 0.2222 0.3202 0.0841 ...
##  $ V45  : num  0.2641 0.0621 0.2111 0.4295 0.0692 ...
##  $ V46  : num  0.1386 0.0203 0.0176 0.3654 0.0528 ...
##  $ V47  : num  0.1051 0.053 0.1348 0.2655 0.0357 ...
##  $ V48  : num  0.1343 0.0742 0.0744 0.1576 0.0085 ...
##  $ V49  : num  0.0383 0.0409 0.013 0.0681 0.023 0.0264 0.0507 0.0285 0.0777 0.0092
##  $ V50  : num  0.0324 0.0061 0.0106 0.0294 0.0046 0.0081 0.0159 0.0178 0.0439 0.0198
##  $ V51  : num  0.0232 0.0125 0.0033 0.0241 0.0156 0.0104 0.0195 0.0052 0.0061 0.0118
##  $ V52  : num  0.0027 0.0084 0.0232 0.0121 0.0031 0.0045 0.0201 0.0081 0.0145 0.009
##  $ V53  : num  0.0065 0.0089 0.0166 0.0036 0.0054 0.0014 0.0248 0.012 0.0128 0.0223
##  $ V54  : num  0.0159 0.0048 0.0095 0.015 0.0105 0.0038 0.0131 0.0045 0.0145 0.0179
##  $ V55  : num  0.0072 0.0094 0.018 0.0085 0.011 0.0013 0.007 0.0121 0.0058 0.0084 .
##  $ V56  : num  0.0167 0.0191 0.0244 0.0073 0.0015 0.0089 0.0138 0.0097 0.0049 0.0068
##  $ V57  : num  0.018 0.014 0.0316 0.005 0.0072 0.0057 0.0092 0.0085 0.0065 0.0032 .
##  $ V58  : num  0.0084 0.0049 0.0164 0.0044 0.0048 0.0027 0.0143 0.0047 0.0093 0.0035
##  $ V59  : num  0.009 0.0052 0.0095 0.004 0.0107 0.0051 0.0036 0.0048 0.0059 0.0056
##  $ V60  : num  0.0032 0.0044 0.0078 0.0117 0.0094 0.0062 0.0103 0.0053 0.0022 0.004
##  $ Class: Factor w/ 2 levels "M","R": 2 2 2 2 2 2 2 2 2 2 ...
```

… Exploratory data analysis …

Exploratory data analysis

```
library(ggplot2);library(reshape2) #install.packages(c("ggplot2","reshape2"))
Sonar.m<-melt(Sonar,id.vars = c("Class"))
ggplot(Sonar.m)+geom_boxplot(aes(x=Class,y=value))+
    facet_wrap(~variable, nrow=5,scales = "free_y") #
```



1.2:

-
  -
-
  - `Class` M: mine –>+, R: rock–>-
- factor
  -
-
  -

2:

1/3 1/5...

```
Sonar$Test<-F #
#  1/3 Test set
Sonar[sample(1:nrow(Sonar),nrow(Sonar)/3),"Test"]<-T
#   Training set : Test set
c(sum(Sonar$Test==F),sum(Sonar$Test==T))
```

```
## [1] 139  69
```

3:

- Test ==F
- X
-

```
fit<-glm(Class~., Sonar[Sonar$Test==F,],family="binomial")
finalFit<-stepAIC(fit,direction = "both",trace = F)
summary(finalFit)$coefficients
```

```
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)      8637     253120   0.034     0.97
## V1             -27931     828078  -0.034     0.97
## V4             -29704     865613  -0.034     0.97
## V7              19584     569191   0.034     0.97
## V12             -3853     115443  -0.033     0.97
## V15             -4393     126979  -0.035     0.97
## V16              9634     278340   0.035     0.97
## V18             -6121     177507  -0.034     0.97
## V24             -6950     204480  -0.034     0.97
## V29              4569     132801   0.034     0.97
## V30            -13257     385129  -0.034     0.97
## V31             10863     313890   0.035     0.97
## V35             -7224     207966  -0.035     0.97
## V36             13153     378812   0.035     0.97
## V39             -9482     282208  -0.034     0.97
## V40             10567     313200   0.034     0.97
## V42             -5664     167265  -0.034     0.97
## V44            -11255     322840  -0.035     0.97
## V48            -25776     746398  -0.035     0.97
## V56            159309    4633625   0.034     0.97
## V58           -179362    5140624  -0.035     0.97
```

4.1: -

```
MinePred<-predict(finalFit,newdata = Sonar[Sonar$Test==T,])
MineAns<-ifelse(MinePred>0.5,"R","M") #>0.5: Level 2
MineAns<-factor(MineAns,levels = c("M","R"))
MineAns
```

```
##   2   5   6   8  14  17  22  24  27  32  34  37  39  40  43  45  48  51  53  55
##   R   R   R   M   R   R   M   R   M   R   R   R   R   R   R   M   M   R   R   M
##  56  60  68  74  75  80  83  84  94  95 103 105 109 112 113 115 123 126 128 130
##   M   R   R   M   R   M   M   M   M   R   M   M   M   M   M   M   M   M   M   M
## 131 132 133 135 143 144 150 151 154 158 160 161 162 163 166 168 169 175 179 183
##   M   M   R   R   M   M   M   R   M   M   M   M   M   M   M   M   M   M   R   M
```

```
## 184 188 190 192 199 200 201 202 203
##   M   M   R   M   M   M   M   M   M
## Levels: M R
```

4.2: -

```
library(caret) # install.packages("caret") #   packages
sensitivity(MineAns,Sonar[Sonar$Test==T,]$Class)
```

```
## [1] 0.87
```

```
specificity(MineAns,Sonar[Sonar$Test==T,]$Class)
```

```
## [1] 0.6
```

```
posPredValue(MineAns,Sonar[Sonar$Test==T,]$Class)
```

```
## [1] 0.74
```

```
negPredValue(MineAns,Sonar[Sonar$Test==T,]$Class)
```

```
## [1] 0.78
```

-

UCI Machine Learning Repository

60 (M) (R)

-

V1 + V2 + V3 + V4 + V7 + V11 + V12 + V13 + V17 + V18 + V22 + V24 + V25 + V26 + V30 + V31 + V32 + V38 + V39 + V48 + V50 + V52 + V53 + V58 + V59 25

-

97% 89% 89% 97%

10.9

-
  - Machine Learning Foundations
  - Machine Learning Techniques
- Market Basket Analysis with R
- Deep Learning in R

Chapter 11

Placeholder

11.1 R + Hadoop

11.2 RHadoop (Cloudera)

11.2.1 /

11.2.2

11.2.3

11.2.3.1 Cloudera CDH QuickStart VM

11.2.3.2 R

11.2.3.3 RHadoop-1

11.2.3.4 RHadoop-2 rmr2

11.2.3.5 RHadoop-3 rhdfs

11.2.4

11.2.5

11.2.6 RStudio Server

11.3 RHadoop MapReduce: easy word count

11.4 R + Spark

Chapter 12

Placeholder

12.1 R

12.2 RStudio

12.3 RStudio

12.3.1

12.3.2 RStudio

Chapter 13

13.1

R YouTube

13.2

1. R 101 & GitHub 101 - [R]
2. - [R]
3. - [R]
4. - [R]
5. - [R]
6. - [R]
7. - [R]
8. - [R]
9. R Function and Package [R]

13.3

1. R101 [R]
2. [R]
3. [R]
4. [R]
5. [R]

6. [R]

Yi-Ju Tseng

Lab:

Bibliography

Hahsler, M., Buchta, C., Gruen, B., and Hornik, K. (2019). *arules: Mining Association Rules and Frequent Itemsets*. R package version 1.6-4.

Milborrow, S. (2019). *rpart.plot: Plot 'rpart' Models: An Enhanced Version of 'plot.rpart'*. R package version 3.0.8.

Therneau, T. and Atkinson, B. (2019). *rpart: Recursive Partitioning and Regression Trees*. R package version 4.1-15.