R-1

```
# 导入安装包

library(Hmisc)
library(MASS)
```

```
# 存储变量

vector <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)                      # 关键字c不可少
vector <- 1:10                                                  # 1-10包含1和10
vector <- seq(1,10,2)                                           # [1-10]内隔2取值

vector[1]                                                       # R的最低索引是1不是0
vector[2:5]                                                     # 读取位置2-5上的数
vector[c(3,6,2)]                                                # 读取位置3，6，2上的数
vector[-2]                                                      # 读取除了第2个位置上的数
vector[-(2:5)]                                                  # 读取除2-5位置上的数

which(vector <= 4)                                              # 获取对应索引

vector2 <- c(11,12,13,14,15,16,17,18,19,20)                     # 利用上面得到的T/F索引列表
vector2[vector<=4]                                              # 11 12 13 14

# 数学分析

summary(vector)                                                 # 获取max,min,mean等分析
mean(vector)                                                    # 平均值
sd(vector)                                                      # 标准差
sum(vector)                                                     # 总和
length(vector)                                                  # 长度

# 向量逐元素加法、乘法 -- 基本的加是逐元素加

v1 = c(1,2,3,4,5)
v2 = c(2,3,4,5,6)
v1 + v2                                                         # 3 5 7 9 11
v1 * v2                                                         # 2 6 12 20 30

# 倍数加、乘向量 -- 标量加乘

v1 = c(1,2,3,4,5)
v1 + 4                                                          # 5 6 7 8 9
v1 * 4                                                          # 4 8 12 16 20
```

```
# 向量转矩阵

v1 = c(1,2,3,4,5,6,7,8,9,10)
m1 <- matrix(v1, 2, 5)                                          # 2行3列按列序排
m2 <- matrix(v1, 2, 5, byrow = TRUE)                            # 这里要求按行来排列

dim(m2)                                                         # 获取维度: 2 5
m2[1,2]                                                         # 获取矩阵里面元素: 2
m2[1,]                                                          # 取第1行所有元素，注意,不要
缺

# 矩阵按行、列求和

m1 <- matrix(1:10, 2, 5)
apply(m1, 1, sum)                                               # 按行求和
apply(m1, 2, sum)                                               # 按列求和
```

```
# 多维矩阵（注意：超过2维不可用matrix()，需要用array()）
# 1维度：一个列竖排列：对应每个vector中的单个元素，在矩阵中维度叫行
# 2维度：好几个竖排列：对应每个vector中的所有元素，在矩阵中维度叫列
# 3维度：外面在嵌套这样的有几个
# 4维度：外面再嵌套
# c(3,4,2,5)左边的是基础维度，外面是嵌套维度

a1 <- array(1:24, c(3,4,2))                                     # 3行4列这样矩阵有2个
a2 <- array(1:120, c(3,4,2,5))                                  # 3行4列有2个，这样的有5个

# 多维数组按维度求和

a1 <- array(1:24, c(3,4,2))
apply(a1, c(1, 2), sum)                 # 缺失第3维，1,2结构不变看作整体，三个每个对应位置求和
a1[, , 1] + a1[, , 2]                   # 效果同上
apply(a1, c(1, 2, 3), sum)              # 这里1,2,3看作整体，其他嵌套求和，这里因为只有3维所以不变

a2 <- array(1:120, c(3,4,2,5))
apply(a2, c(1, 2, 3), sum)              # 1,2,3维结构是整体，后面每5个对应求和
apply(a2, c(1, 2, 4), sum)              # 1,2,4维结构是整体，第3维每2个对应求和
```

```
# data frame 数据框（其实就是table）

v1 <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
v2 <- c(11,12,13,14,15,16,17,18,19,20)
dataFrame_1 <- data.frame(v1, v2)                               # 竖排组合了两个向量
head(dataFrame_1)                                               # 列出前面部分内容

# 赋予列名

colnames(dataFrame_1) <- c("A", "B")

# 访问列、指定元素
```

```
dataFrame_1$B                                                   # 指定列名访问列（横排）
dataFrame_1[, 2]                                                # 同上，第2列所有行（横排）
dataFrame_1["B"]                                                # 访问B列（竖排）
dataFrame_1[,"B"]                                               # B列所有元素（横排）
dataFrame_1[["B"]]                                              # 同上,名字为B组的所有列表元
素

dataFrame_1[2, ]                                                # 行也可以正常访问

dataFrame_1$B[3]                                                # 获取B列第三个元素
dataFrame_1[3, 2]
dataFrame_1[3, "B"]                                             # 可以用index或者名字来存取
dataFrame_1[["B"]][3]

dataFrame_1[dataFrame_1$A >= 4, ]                               # 符合A中的元素>=4的所有记录
subset(dataFrame_1, A >= 4)                                     # 效果同上

dataFrame_1[dataFrame_1$A >= 4 & dataFrame_1$B <= 18 , ]        # &逐元素比较，&是and
dataFrame_1[dataFrame_1$A >= 4 && dataFrame_1$B <= 18 , ]       # &&会报错，因为&&只比较第1个

# 数据框的锁定

attach(dataFrame_1)                                             # 这里锁定df数据
A                                                               # 之后可直接用A来获取里面数据
detach(dataFrame_1)                                             # 解绑
A                                                               # 现在会报错

# with引用资源

with(dataFrame_1, A+B)                                          # 引用数据帧，返回A+B
with(dataFrame_1, {
  tem <- A + B
  sum(tem * tem)
})
```

```
# Loop循环

for (i in v1){
  print(i)
}

# if-else

if (2+2==4) print("Correct") else print("wrong")

if (2+2==4){                                                    # 如果竖排格式要加大括号
  print("Correct")
}else{                                                          # 且这里的else不能换行要在}后
  print("wrong")
}
```

```
# 查看当前工作目录

getwd()

# 切换当前工作目录

setwd("F:/File")
```

```
# 读取csv文件

data_1 <- read.table("plane_excel.csv", sep = ",", header = TRUE)
dim(data_1)

data_2 <- read.csv("plane_excel.csv")           # 与read.table不同在于不用指定后面2个默认参数
dim(data_1)

data_3 <- read.csv("sulphur_oxide.txt", sep = " ")              # 这里txt以空格为间隔符
dim(sulphur_data)                                               # 读完之后与数据帧同操作

# 查看是否是数据框、行列数

print(is.data.frame(data_2))
print(ncol(data))
print(nrow(data))

# 删除指定列

Sulphur_Data <- Sulphur_Data[, -2]
dim(sulphur_data)
sulphur_data <- sulphur_data[, -2]
dim(sulphur_data)
sulphur_data <- sulphur_data[, -2]                      # 最后一次只剩下1列自动变向量不包含列名
dim(sulphur_data)

# 查询

like <- max(data$likes)
retval <- subset(data, likes==222)                              # 效果同上
retval <- subset(data, likes > 1 & name=="Runoob")

# 写入csv文件

write.csv(retval,"runoob.csv")
newdata <- read.csv("runoob.csv")
print(newdata)
```

```
# 只保存输出到文本

sink("output.txt", append=FALSE, split = FALSE)             # split决定控制台还要不要显示输出
a = c(1,2,3)
```

```
print(a)
a = a*4
print(a)
sink()

# 保存 运行语句和对应输出 到文本

con <- file("lab.log")
sink(con, append = TRUE)                                        # 追加为TRUE否则会重写
sink(con, append = TRUE, type = "message")                      # message表示报错信息

source("1.R", echo = TRUE, max.deparse.length = 10000)          # 指定脚本文件或者直接写

sink()                                                          # 关闭sink()
sink(type = "message")                                          # 关闭第二个sink
```

```
# list和vector区别在于list更像是组合

lt <- list(c("Google","Runoob"), matrix(c(1,2), nrow = 2), list("runoob",12.3))
print(lt[1])

# list 如果是多元素的话可遍历，单元素不行

for (i in lt){
  print(i)
}
```

```
# 字符串分割

s <-"aa;bb;cc"
a <-strsplit(s, split=";")
b <- unlist(strsplit(s,split=";"))                              # ";"可改为""

# 注意第一个遍历不了，因为a里面就一个元素，所视的分开是一个元素的组合

a                                                               # [1] "aa" "bb" "cc"
class(a)                                                        # [[1]] [1] "list"
length(a)                                                       # 1
a[2]                                                            # [[1]] NULL

for (i in a[1]){                                                # 这样可行
  print(i)
}

# character的组可以遍历

b                                                               # [1] "aa" "bb" "cc"
class(b)                                                        # [1] "character"
length(b)                                                       # 3
b[2]                                                            # [1] "bb"
```

> Write a loop to spell out the letters of your family name one letter at a time

```
a = "Tang Jin"
b <- unlist(strsplit(a,split=""))

for (i in b){
  print(i)
}
```

R-2

```
# 单列数量变量

hist(x, xlab, ylab)
ecdf(x)                                                     # x是一列
quartile(x, c(0.25, 0.75))
boxplot(x)

# 多列数量变量

plot(x, y)                                                  # 散点图
pairs(data)
parcoord(data)
boxplot(x ~ grouped_by_var)
table(x, y)
mosaicplot(~ x+y, data = data)

# 种类变量

barplot(x)                                                  # x是一列
barplot(data, beside=TRUE)                                  # data是matrix, stack or group
mosaicplot(data)

# PCA

pca <- prcomp(data)
plot(pca)
biplot(pca)
```

```
# 数据分析

var <- as.data.frame(apply(data, 2, var))
min <- as.data.frame(apply(data, 2, min))
max <- as.data.frame(apply(data, 2, max))
```

```
mean <- as.data.frame(apply(data, 2, mean))
analysis <- cbind(mean, var, min, max)

# SOM

library(kohonen)

data <- as.matrix(data)
som_grid <- somgrid(xdim = 14, ydim = 14, topo = "hexagonal")
som_model <- kohonen::som(data, grid = som_grid)
system.time(som_model)

plot(som_model, type = "changes")
plot(som_model, type = "counts")
plot(som_model, type = "quality")
plot(som_model, type = "dist.neighbours")
plot(som_model, type = "codes")
plot(som_model, type = "property", property = som_model$codes[, var]) # 单个属性
plot(som_model, type = "mapping") # SOM图

# MLP

library(RSNNS)

Data <- subset(data, fullDataSet[,46] == 0)
train <- Data[, c(1, 2, 3, 6, 8, 33)]

model <- mlp(trainset$inputs, trainset$targets,
             inputsTest = trainset$inputs, targetsTest = trainset$targets)

predict <- predict(model, trainset$inputsTest)
confusionMatrix(trainset$targetsTest, predict)
precision <- (confusionM[1,1]+confusionM[2,2]+confusionM[3,3])/sum(trainset$inputsTest)
```

R-3

```
# 数据处理

data <- data[data$credit.rating > 0, ]
set <- sample(1:nrow(data), nrow(data)/2, replace=FALSE)
train <- data[set, ]
test <- data[-set, ]

sub <- c(sample(1:50, 25), sample(51:100, 25), sample(101:150, 25))

train$credit.rating <- sapply(train$credit.rating, as.factor)   # as.ordered

# 模型

model <- svm(credit.rating ~ ., data = train)
model <- tune.svm(credit.rating ~ ., data = train)
model <- ksvm(credit.rating ~ ., data = train, prob.model = TRUE, kernel = "rbfdot")

model <- rpart(credit.rating ~ ., data = train)     # control = rpart.control(minsplit=1)

model <- randomForest(train_X, train_y, ntree = 500, mtry = 9,  # 特殊
                      importance=TRUE, proximity = TRUE)

model <- NaiveBayes(credit.rating ~ ., train_data)

model <- glm(I(credit.rating==1) ~ ., data = train_data, family = binomial("logit"))
model <- step(glm(I(credit.rating==1) ~ ., data = train_data, family=binomial("logit")))


print(model)
plot(model)
text(model)


predict(model, train_data[982, -46])
predict(model, test_data)
predict(svm, test_data, type = "probabilities")
predict(glm, test_data, type = "response")


table(truth = test_data[, 46], prediction = pred)       # table 是truth和pred
acc <- as.character(f[1,1]+f[2,2]+f[3,3])/sum(f)
print(paste("Accuracy: ", acc, sep=""))


# ROC

library(ROCR)

prediction <- prediction(predicted, true)               # 预测是假真
perf <- performance(prediction, "tpr", "fpr")
plot(perf)

plot(perf1, col = 2)
plot(perf2, add = TRUE, col = 3)
abline(0, 1)
legend("bottomright", c("A", "B"), lty = 1, col = 2:3)


# 其他

plot(iris.svm, data = iris, Petal.Width ~ Petal.Length,
     slice = list(Sepal.Width = 3, Sepal.Length = 6))
```

```
logit <- function(p) {
    log(p/(1 - p))
}

a = sort(rules, by="lift")
```

```
sample1.apriori <- apriori(data)
inspect(sample1.apriori)
sample1.apriori <- apriori(data,
                           parameter=list(supp=0.5, conf=1.0),
                           appearance=list(rhs=c("Survived=No", "Survived=Yes"),)

plot(sample1.apriori,method="graph")
```