**R语言自学**

**一、R基础**

1. 环境设置：install.packages()

2. 输出与赋值：myString <- “Hello, World!” print(myString) =也可以赋值

->：右分配 c(3,1,TRUE,2+3i) -> v1

3. 数据类型：

（1）向量 apple <- c(‘red’,’green’,’yellow’)

（2）列表 list1 <- list(c(2,5,3),21.3,sin)

（3）矩阵 M = matrix( c(‘a’,’a’,’b’,’c’,’b’,’a’), nrow = 2, ncol = 3, byrow = TRUE)

（4）数组 a <- array(c(‘green’,’yellow’), dim=c(3,3,2)) dim创造所需要的维数

（5）因子

apple\_colors <- c('green','green','yellow','red','red','red','green')

factor\_apple <- factor(apple\_colors)

print(factor\_apple)

输出：green green yellow red red red yellow green

Levels: green red yellow

（6）数据帧：表格数据对象

BMI <- data.frame(

gender = c("Male", "Male","Female"),

height = c(152, 171.5, 165),

weight = c(81,93, 78),

Age = c(42,38,26)

)

print(BMI)

输出：gender height weight Age

1 Male 152.0 81 42

2 Male 171.5 93 38

3 Female 165.0 78 26

4. 变量 数字、字母、\_和.：只能以字母或不以数字后跟的点开头

（1）cat()函数将多个项目组合成连续打印输出

e.g. cat ("var.1 is ", var.1 ,"

")

cat ("var.2 is ", var.2 ,"

")

cat ("var.3 is ", var.3 ,"

")

输出：var.1 is 0 1 2 3

var.2 is learn R

var.3 is 1 1

（2）查找变量：print(ls())

（3）删除变量：rm(varname)

5. 运算符

（1）%%：两个向量求余； （2）%/%：两个向量相除求商；

（3）逻辑运算符&&和||只考虑向量的第一个元素，给出单个元素的向量作为输出

（4）%in%：用于标识元素是否属于向量； （5）%\*%：将矩阵与其转置相乘

6. 决策

（1）if(条件){语句} if(条件){语句}else{语句} if(条件){语句}else if(条件){语句}else{语句}

switch(expression, case1, case2, case3....)

（2）包 library()列出所有包； search()获取当前在R环境中加载的所有包

（3）循环

repeat {

commands

if(condition) { 类似do while

break

}

}

while (test\_expression) {statement} 与C语言一致

for (test\_expression) {statement}

break用法与C一致，continue变成了next

7. 数据重塑

（1）cbind连接多个向量来创建数据帧

city <- c("Tampa","Seattle","Hartford","Denver")

state <- c("FL","WA","CT","CO")

zipcode <- c(33602,98104,06161,80294)

addresses <- cbind(city,state,zipcode)

输出：city state zipcode

[1,] "Tampa" "FL" "33602"

[2,] "Seattle" "WA" "98104"

[3,] "Hartford" "CT" "6161"

[4,] "Denver" "CO" "80294"

（2）rbind合并两个数据帧 all.addresses <- rbind(addresses, new.address)

（3）melt拆分数据 molten.ships <- melt(ships, id = c("type","year"))

将除类型和年份以外的所有列转换为多行显示

8. 函数

function\_name <- function(arg\_1, arg\_2, …){

Function body

}

（1）内置函数： seq(1,10)：1 2 3 4 5 6 7 8 9 10

mean(1,10)：5.5 sum(1:10)：55

（2）自定义函数的调用：

new.function <- function(a,b,c) new.function(5,3,11)或new.function(a=11,b=5,c=3)

9. 字符串

（1）无效示例：f <- 'Single quote ' inside single quote' print(f)

g <- "Double quotes " inside double quotes" print(g)

（2）连接字符串：print(paste(a,b,c, sep = "", collapse = "")) sep表示参数之间的任何分隔符，collapse表示消除两个字符串之间的空格

其中a<-“Hello” b<-‘How’ c<-“are you?”

输出：”HelloHoware you?“

（3）格式化：format(x, digits, nsmall, scientific, width, justify = c("left", "right", "centre", "none"))

x是向量输入，nsmall是小数点右边的最小位数，digits是显示的总位数，scientific为TRUE显示科学计数法，justify表示字符串向左、右或中心显示

e.g. result <- format(23.123456789, digits = 9)

result <- format(c(6, 13.14521), scientific = TRUE)

result <- format(23.47, nsmall = 5)

result <- format("Hello", width = 8, justify = "c")

输出："23.1234568"

"6.000000e+00" "1.314521e+01"

"23.47000"

" Hello “

（4）统计字符数：nchar(x)

（5）改变大小写：toupper(x) tolower(x)

（6）提取字符串的部分：substring(x,first,last)

10. 向量

（1）创建多元素向量：冒号表达式：v <-6.6:12.6

序列运算符：seq(5,9,by=0.4)

（2）如果我们对不等长的两个向量应用算术运算，则较短向量的元素被循环以完成操作。

（3）排序：sort(v, decreasing= TRUE)

11. 列表

（1）命名列表元素：list\_data <- list(c("Jan","Feb","Mar"), matrix(c(3,9,5,1,-2,8), nrow = 2),

list("green",12.3))

names(list\_data) <- c("1st Quarter", "A\_Matrix", "A Inner list")

访问列表元素：list\_name[]

（2）合并列表：merge.list <- c(list1, list2)

（3）将列表转化为向量：unlist(list1)，unlist(list2)

12. 矩阵

（1）matrix(data, nrow, ncol, byrow, dimnames)

nrow：创建的行数；ncol：创建的列数；byrow：如果为TRUE，按行排列；dimname是分配给行和列的名称

rownames = c("row1", "row2", "row3", "row4")

colnames = c("col1", "col2", "col3")

P <- matrix(c(3:14), nrow = 4, byrow = TRUE, dimnames = list(rownames, colnames))

输出：col1 col2 col3

row1 3 4 5

row2 6 7 8

row3 9 10 11

row4 12 13 14

（2）访问矩阵：P[1,3]，P[2,]

13. 数组

（1）创建一个由两个3×3矩阵组成的数组，每个矩阵有3行3列，并命名：

vector1 <- c(5,9,3)

vector2 <- c(10,11,12,13,14,15)

column.names <- c("COL1","COL2","COL3")

row.names <- c("ROW1","ROW2","ROW3")

matrix.names <- c("Matrix1","Matrix2")

result <- array(c(vector1,vector2),dim = c(3,3,2),dimnames = list(row.names,column.names,

matrix.names))

print(result)

输出：, , Matrix1

COL1 COL2 COL3

ROW1 5 10 13

ROW2 9 11 14

ROW3 3 12 15

, , Matrix2

COL1 COL2 COL3

ROW1 5 10 13

ROW2 9 11 14

ROW3 3 12 15

（2）跨数组元素的计算：apply(x, margin, fun) x是一个数组，margin是所使用的数据集的名称，fun是要应用于数组元素的函数

e.g. apply(new.array, c(1), sum) 最终呈现一个单行向量，对两数组的每一行求和的结果

14. 因子

（1）判断：is.factor()

（2）更改级别顺序：factor(因子名,级别)

e.g. new\_order\_data <- factor(factor\_data,levels = c("East","West","North"))

（3）生成因子级别：gl(n,k,labels) n是给出级数的整数，k是给出复制数目的整数，labels是所得因子水平的标签向量

e.g. v<- gl(3,4,labels=c(“Tampa”, “Seattle”, “Boston”)

Tampa Tampa Tampa Tampa Seattle Seattle Seattle Seattle Boston Boston Boston Boston

Levels: Tampa Seattle Boston

15. 数据帧

·列名称应为非空。 ·行名称应该是唯一的。

··存储在数据帧中的数据可以是数字，因子或字符类型。 ·每个列应包含相同数量的数据项。

emp.data <- data.frame(

emp\_id = c (1:5),

emp\_name = c("Rick","Dan","Michelle","Ryan","Gary"),

salary = c(623.3,515.2,611.0,729.0,843.25),

start\_date = as.Date(c("2012-01-01", "2013-09-23", "2014-11-15", "2014-05-11",

"2015-03-27")),

stringsAsFactors = FALSE

)

（1）查看数据帧结构：str(emp.data)

（2）提取指定列：法1：data.frame(emp.data$emp\_name, emp.data$salary)

提取名为emp.data的数据帧中的emp\_name和salary两列

法2：emp.data[1:2,] 利用矩阵形式提取

法3：emp.data[c(3,5),c(2,4)] 利用向量提取某几行的某几列

（3）添加列：数据帧名 <- c(值1, 值2, …)

e.g. emp.data$dept <- c(“IT”, “Operations”, “IT”, “HR”, “Finance”)

（4）添加行：利用rebind函数：

emp.newdata <- data.frame(

emp\_id = c (6:8),

emp\_name = c("Rasmi","Pranab","Tusar"),

salary = c(578.0,722.5,632.8),

start\_date = as.Date(c("2013-05-21","2013-07-30","2014-06-17")),

dept = c("IT","Operations","Fianance"),

stringsAsFactors = FALSE

)

emp.finaldata <- rbind(emp.data,emp.newdata)

**二、图表**

1. 条形图

barplot(H, xlab, ylab, main, names.arg, col)

其中H是包含在条形图中使用的数值的向量或矩阵，xlab是x轴的标签，ylab是y轴的标签，main是条形图的标题。names.arg是在每个条下出现的名称的向量。col用于向图中的条形提供颜色。

e.g. H <- c(7,12,28,3,41)

png(file = "barchart.png") （图像名称）

barplot(H)

保存图片：dev.off()

barplot(H,names.arg = M,xlab = "Month",ylab = "Revenue",col = "blue",

main = "Revenue chart",border = "red") 边界颜色，填充颜色可分别测试

组合条形图与堆叠条形图：

# Create the input vectors.

colors <- c("green","orange","brown")

months <- c("Mar","Apr","May","Jun","Jul") 创建堆叠标签

regions <- c("East","West","North")

# Create the matrix of the values. 创建堆叠数据

Values <- matrix(c(2,9,3,11,9,4,8,7,3,12,5,2,8,10,11),nrow = 3,ncol = 5,byrow = TRUE)

# Give the chart file a name.

png(file = "barchart\_stacked.png") 创建图表名称

# Create the bar chart.

barplot(Values,main = "total revenue",names.arg = months,xlab = "month",ylab = "revenue",

col = colors) 绘制图标，创建横纵坐标轴和颜色

# Add the legend to the chart.

legend("topleft", regions, cex = 1.3, fill = colors) 添加图例

# Save the file.

dev.off()

2. 箱线图

boxplot(x, data, notch, varwidth, names, main)

·x是向量或公式。

·data是数据帧。

·notch是逻辑值。 设置为TRUE以绘制凹口。

·varwidth是一个逻辑值。 设置为true以绘制与样本大小成比例的框的宽度。

·names是将打印在每个箱线图下的组标签。

·main用于给图表标题。

e.g. boxplot(mpg ~ cyl, data = mtcars, xlab = "Number of Cylinders", ylab = "Miles Per Gallon", main = "Mileage Data")

3. 直方图

hist(v,main,xlab,xlim,ylim,breaks,col,border)

·v是包含直方图中使用的数值的向量。

·main表示图表的标题。

·col用于设置条的颜色。

·border用于设置每个条的边框颜色。

·xlab用于给出x轴的描述。

·xlim用于指定x轴上的值的范围。

·ylim用于指定y轴上的值的范围。

·break用于提及每个条的宽度。

e.g. hist(v,xlab = "Weight",col = "green",border = "red", xlim = c(0,40), ylim = c(0,5),

breaks = 5)

4. 折线图

（1）plot(v,type,col,xlab,ylab)

·v是包含数值的向量。类型采用值“p”仅绘制点，“l”仅绘制线和“o”绘制点和线。

·xlab是x轴的标签。

·ylab是y轴的标签。

·main是图表的标题。

·col用于给点和线的颜色。

e.g. plot(v,type = "o", col = "red", xlab = "Month", ylab = "Rain fall",

main = "Rain fall chart")

（2）通过使用lines()函数，可以在同一个图表上绘制多条线。

在绘制第一行之后，lines()函数可以使用一个额外的向量作为输入来绘制图表中的第二行。

e.g. plot(v,type = "o",col = "red", xlab = "Month", ylab = "Rain fall", main = "Rain fall chart")

lines(t, type = "o", col = "blue")

5. 散点图

（1）plot(x, y, main, xlab, ylab, xlim, ylim, axes)

·x是其值为水平坐标的数据集。

·y是其值是垂直坐标的数据集。

·main要是图形的图块。

·xlab是水平轴上的标签。

·ylab是垂直轴上的标签。

·xlim是用于绘图的x的值的极限。

·ylim是用于绘图的y的值的极限。

·axes指示是否应在绘图上绘制两个轴。

e.g. plot(x = input$wt,y = input$mpg,

xlab = "Weight",

ylab = "Milage",

xlim = c(2.5,5),

ylim = c(15,30),

main = "Weight vs Milage"

)

6. 饼状图

（1）pie(x, labels, radius, main, col, clockwise)

·x是包含饼图中使用的数值的向量。

·labels用于给出切片的描述。

·radius表示饼图圆的半径（值-1和+1之间）。

·main表示图表的标题。

·col表示调色板。

·clockwise是指示片段是顺时针还是逆时针绘制的逻辑值。

e.g. # Create data for the graph.

x <- c(21, 62, 10, 53)

labels <- c("London", "New York", "Singapore", "Mumbai")

# Give the chart file a name.

png(file = "city\_title\_colours.jpg")

# Plot the chart with title and rainbow color pallet.

pie(x, labels, main = "City pie chart", col = rainbow(length(x)))

# Save the file.

dev.off()

\*\*\*切片百分比及图例：

piepercent<- round(100\*x/sum(x), 1)

pie(x, labels = piepercent, main = "City pie chart",col = rainbow(length(x)))

legend("topright", c("London","New York","Singapore","Mumbai"), cex = 0.8, fill = rainbow(length(x)))

（2）3D饼图：软件包plotrix有一个名为pie3D（）的函数

# Get the library.

library(plotrix)

# Create data for the graph.

x <- c(21, 62, 10,53)

lbl <- c("London","New York","Singapore","Mumbai")

# Give the chart file a name.

png(file = "3d\_pie\_chart.jpg")

# Plot the chart.

pie3D(x,labels = lbl,explode = 0.1, main = "Pie Chart of Countries ")

# Save the file.

dev.off()

下载新的包：

install.packages("包名")

**三、数据接口**

1. csv文件

（1）获取和设置工作目录：

# Get and print current working directory.

print(getwd())

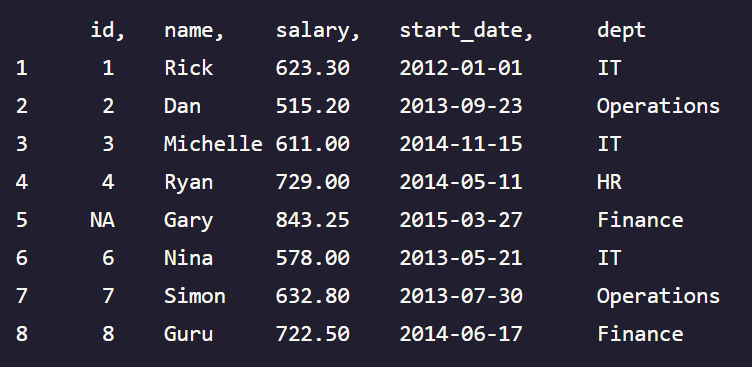
# Set current working directory.

setwd("/web/com")

（2）输入为CSV文件：data <- read.csv(“input.csv”)

以数据帧形式输入：data <- read.csv(“input.csv”)

（3）数据处理：e.g.



最大值：sal <- max(data$salary)

输出那一行的数据：retval <- subset(data, salary == max(salary))

获取某个条件的行的信息：

retval <- subset( data, dept == "IT")

info <- subset(data, salary > 600 & dept == "IT")

retval <- subset(data, as.Date(start\_date) > as.Date("2014-01-01"))

写入CSV文件：write.csv(retval,"output.csv", row.names = FALSE)

2. excel文件

data <- read.xlsx("input.xlsx", sheetIndex = 1)

3. JSON文件

result <- fromJSON(file = "input.json")

转化为数据帧：json\_data\_frame <- as.data.frame(result)

**四、统计**

1. 平均值：

mean(x, trim = 0, na.rm = FALSE, ...)

·x是输入向量。

·trim用于从排序向量的两端丢弃一些观察结果。

·na.rm用于从输入向量中删除缺失值。na.rm=TRUE即去除NA值。

2. 中位数：

median(x, na.rm = FALSE)

3. 找出出现频率最高的值：

charv <- c("o","it","the","it","it")

# Calculate the mode using the user function.

result <- getmode(charv)

print(result)

4. 线性回归

（1）lm函数：

x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)

y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

# Apply the lm() function.

relation <- lm(y~x)

print(summary(relation))

输出：Call:

lm(formula = y ~ x)

Residuals:

Min 1Q Median 3Q Max

-6.3002 -1.6629 0.0412 1.8944 3.9775

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -38.45509 8.04901 -4.778 0.00139 \*\*

x 0.67461 0.05191 12.997 1.16e-06 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.253 on 8 degrees of freedom

Multiple R-squared: 0.9548, Adjusted R-squared: 0.9491

F-statistic: 168.9 on 1 and 8 DF, p-value: 1.164e-06

（2）predict函数：

predict(object, newdata)

·object是已使用lm()函数创建的公式。

·newdata是包含预测变量的新值的向量。

# The predictor vector.

x <- c(151, 174, 138, 186, 128, 136, 179, 163, 152, 131)

# The resposne vector.

y <- c(63, 81, 56, 91, 47, 57, 76, 72, 62, 48)

# Apply the lm() function.

relation <- lm(y~x)

# Find weight of a person with height 170. 找到x=170时对应的y值

a <- data.frame(x = 170)

result <- predict(relation,a)

print(result)

图形可视化回归：

# Give the chart file a name.

png(file = "linearregression.png")

# Plot the chart.

plot(y,x,col = "blue",main = "Height & Weight Regression",

abline(lm(x~y)),cex = 1.3,pch = 16,xlab = "Weight in Kg",ylab = "Height in cm")

# Save the file.

dev.off()