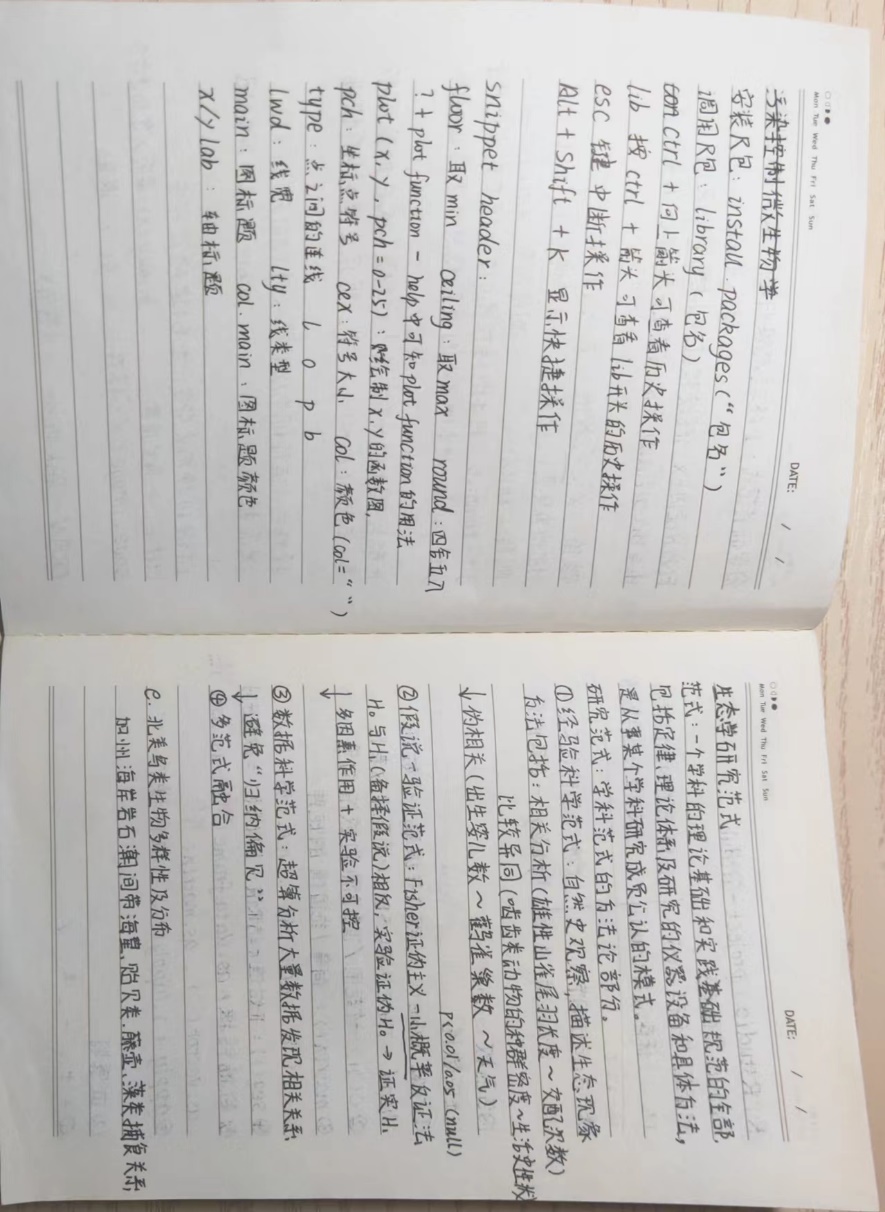
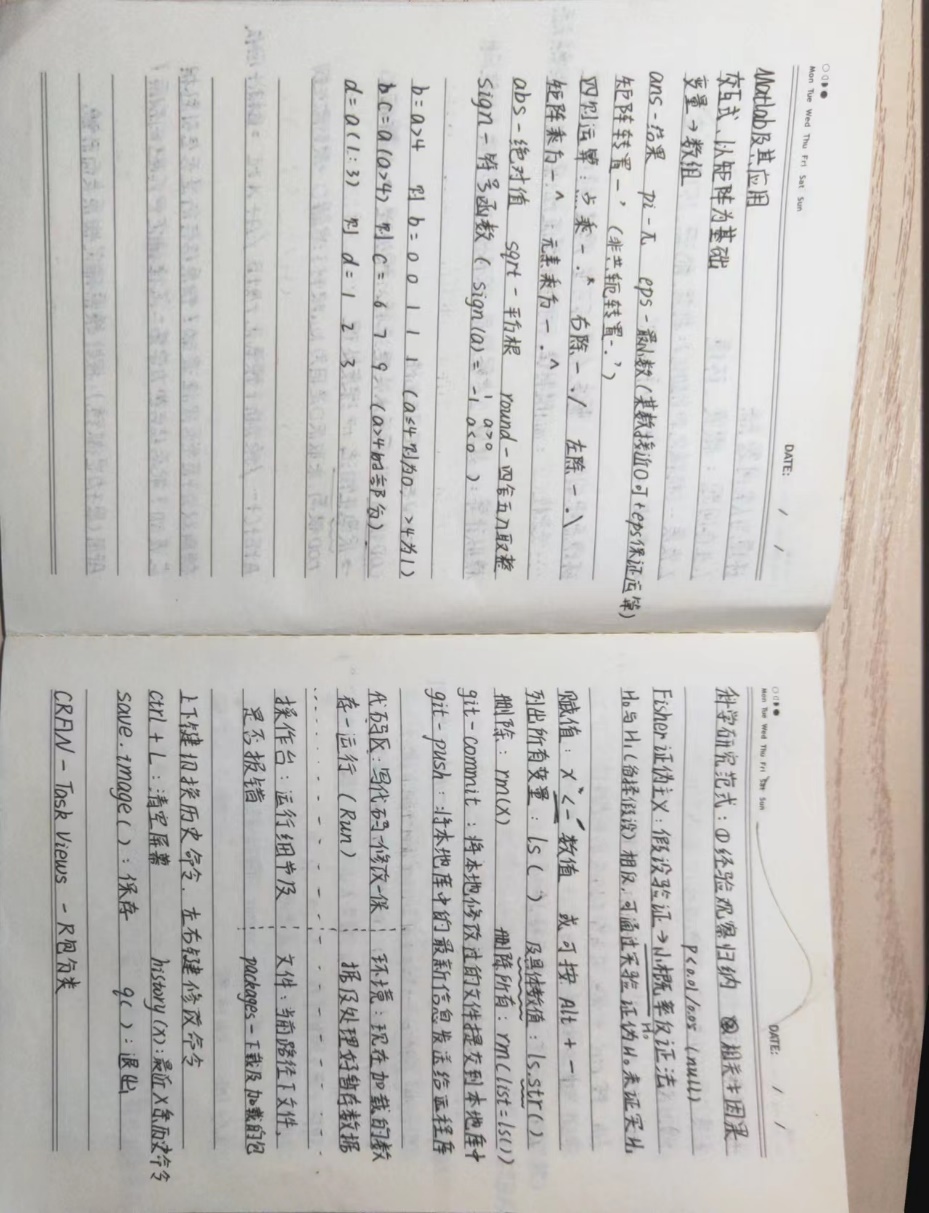
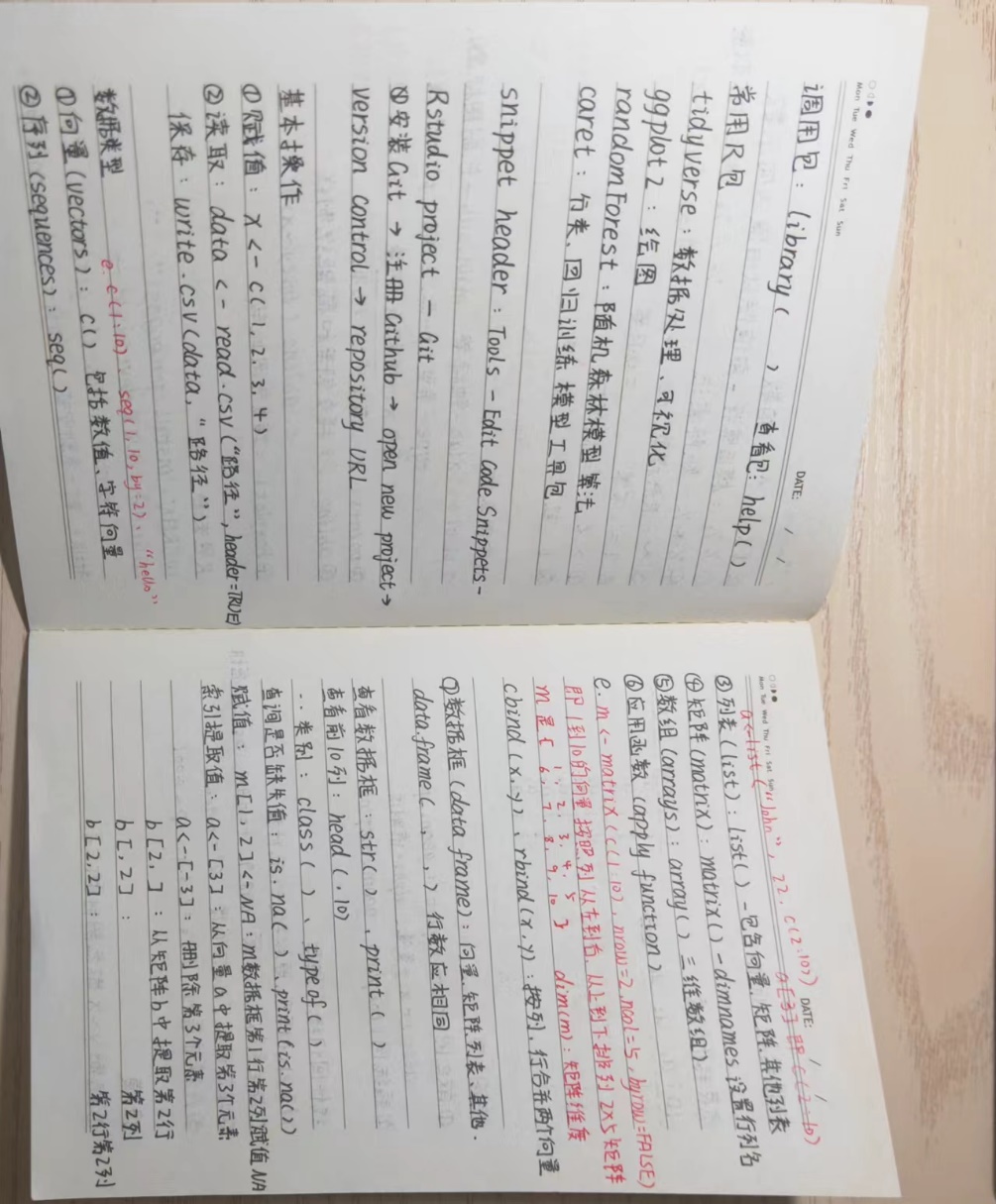
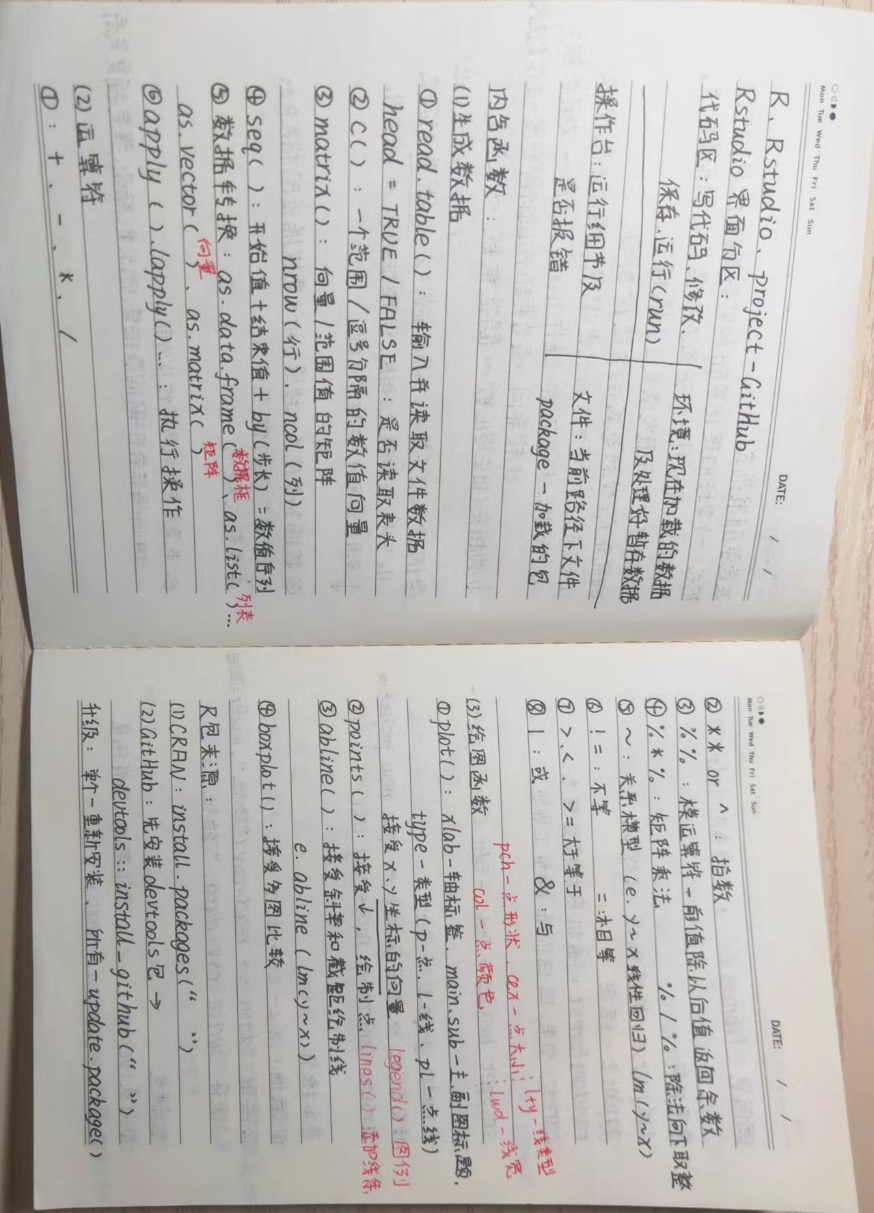
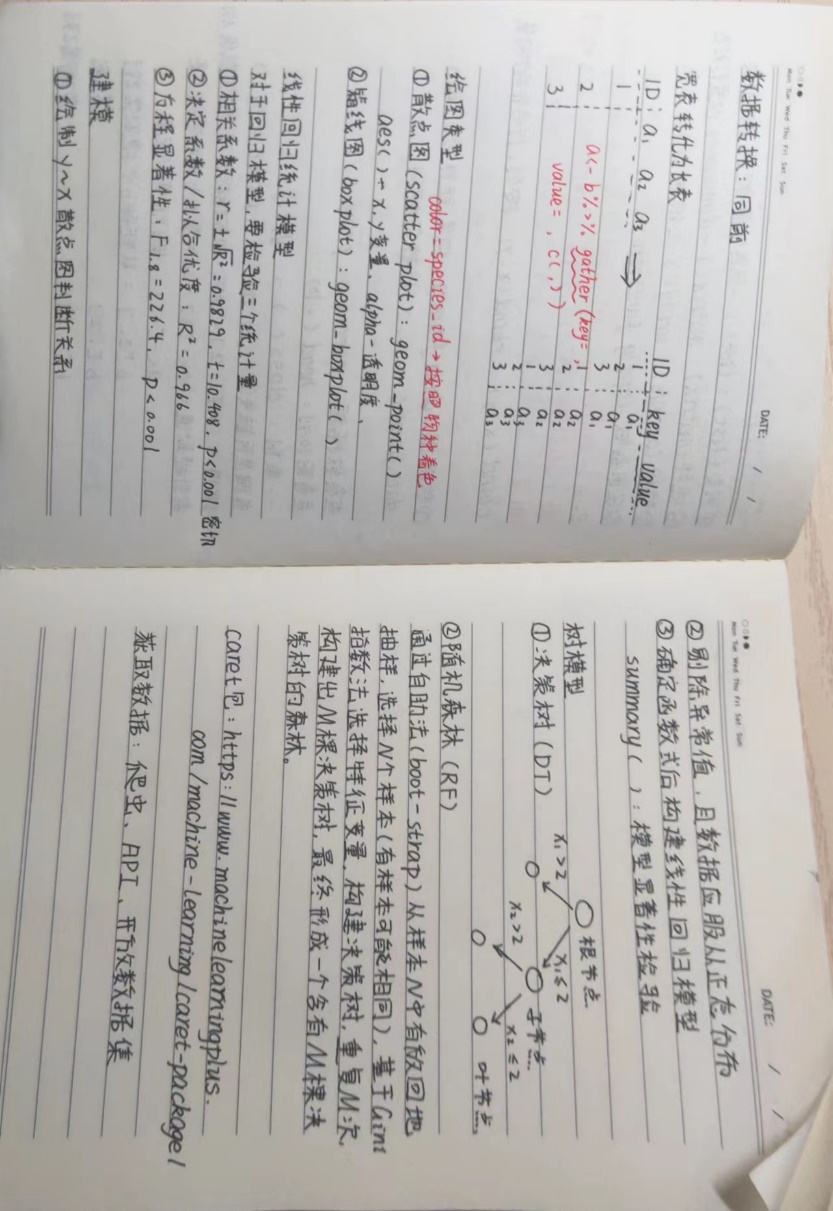
**数据驱动的生态学研究方法课堂笔记**

**孙钰钰 SA23008215**

**一、课堂理论笔记**







**二、课堂代码笔记**

**（一）3月13日课堂代码笔记**

### 1.赋值

<-：将右侧的值赋给左侧的变量

->：与<-相反，将左侧的变量赋值为右侧的值

=：也用于赋值操作，但通常在函数调用中使用

### 2.简单运算

+：加法

-：减法

\*：乘法

/：除法

^：指数运算

%/%：整数除法（向下取整）

%%：取余数

### 3.基础代码运算

floor(8.9)：向下取整

ceiling(8.9)：向上取整

round(8.4)：四舍五入

<：小于

>：大于

<=：小于等于

>=：大于等于

==：等于

!=：不等于

!：逻辑非

|：逻辑或

&：逻辑与

### 4.绘图

x：数据点的横坐标。

y：数据点的纵坐标。

pch：数据点的绘制形状。

cex：数据点的大小。

col：数据点、线条或填充区域的颜色。

bg：数据点的背景颜色。

type：绘制的类型，如散点图（"p"）、线图（"l"）、直方图（"h"）等。

lwd：线条的宽度。

lty：线条的类型，如实线（1）、虚线（2）、点线（3）等。

lineend：线条的端点样式，如圆形（"round"）、方形（"square"）等。

main：图形的主标题。

sub：图形的副标题。

xlab：x轴的标签。

ylab：y轴的标签。

xlim：x轴的范围。

ylim：y轴的范围。

xaxt：x轴的刻度线类型。

yaxt：y轴的刻度线类型。

legend：添加图例。

inset：图例的位置与图形的距离。

abline()：用斜率和截距在图形中绘制直线。

举例：

plot(x, y, pch =0, cex = 1.5, col="red", type="l",lwd=2,lty=1,

main="This is a graph",col.main='blue',

xlab="temperature",ylab="body size")

legend('topleft',inset=0.05,"body size",lty=3,col='green',lwd=4)

abline(a=4,b=5,col='blue') # a是斜率，b是截距

abline(h=c(4,6,8),col="dark green",lty=2)

abline(v=c(4,6,8),col="dark green",lty=2)

### 6.安装并加载包

install.packages("packagefinder", dependencies = TRUE)

library(packagefinder)

findPackage("community ecology")

### 7.包的解释

help(package="tidyverse")

**（二）3月15日课堂代码笔记**

getwd( ) # 当前工作目录

setwd( ) # 设置工作目录

### 数据类型

### 1. 向量（Vectors）：c( )函数创建，包括数值向量（numeric）和字符向量（character）

vec <- c(3,4,5,6) # 创建向量

char\_vec<-c("shubham","nishka","gunjan","sumit")

logic\_vec<-c(TRUE,FALSE,FALSE,TRUE)

seq\_vec<-seq(1,4,by=0.5) # 从1到4，以0.5为步长逐步增加的序列向量

length(a) # 向量长度（元素个数）

### 2. 列表（Lists）：list() 函数创建，包含向量、矩阵和其他列表

out\_list<-list(vec,char\_vec,logic\_vec) # 创建列表

list\_data <- list(c("Shubham","Arpita","Nishka"),

matrix(c(40,80,60,70,90,80), nrow = 2),

list("BCA","MCA","B.tech"))

list1 <- list(10:20) # 列表包含向量、矩阵、字符

### 3. 矩阵（Matrices）：matrix() 函数创建，使用 dimnames 参数设置行列的名称

P <- matrix(c(5:16), nrow = 4, byrow = TRUE) # 矩阵数组

### 4.数据框（Dataframe）: data.frame() 函数创建, 包含许多组分

emp.data<- data.frame(employee\_id = c (1:5),

employee\_name = c("Shubham","Arpita","Nishka","Gunjan","Sumit"),

sal = c(623.3,915.2,611.0,729.0,843.25),

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

stringsAsFactors = FALSE) # 数据框包含众多组分

### 5.确定数据类型

class(pi) # 查看变量分类

typeof(pi) # 查看数据类型

print(is.factor(data)) # 检查向量 data 是否为因子（factor）类型

### 6.转化数据类型

factor(data):将字符向量 data 转换为因子（factor）类型的数据

unlist():转换为向量

as.data.frame()：转换为数据框

data.matrix()：转换为矩阵

array():转化为多维数据矩阵

### 7.将数据框保存为xlsx类型

library("xlsx")

write.xlsx(emp.data, file = "data/employee.xlsx",col.names=TRUE,

row.names=FALSE,sheetName="Sheet4",append = TRUE)

保存为csv类型

write.csv(excel\_data, "data/exployee.csv")

csv\_data <- read.csv("data/exployee.csv")

### 8.查看数据

str( )

head( , x)

print( )

**（三）3月20日课堂代码笔记**

### 1.将向量转换为矩阵，指定矩阵有4行,数据从1到8顺序填充

matrix\_data=matrix(c(1,2,3,4,5,6,7,8),nrow = 4)

dataframe\_data1<-data.frame(a=1:3,b=letters[10:12], c=seq(as.Date("2004-01-01"),

by="week",length.out=3), stringsAsFactors = TRUE)

# 用于控制字符型变量是否转换为因子（factor），因子是一种特殊数据类型用于表示分类变量，通常用于有限个数的离散取值。

matrix\_data<-data.matrix(dataframe\_data1[1:2])

# [1:2]是指第1和2列的数据，[1,2]是指第一行第二列的数据

df1<-data.frame(x=1:10,y=10:1)

df2<-data.frame(x=11:15,y=15:11)

# 创建数组并指定维度名称

Array1<-array(data = c(unlist(df1),unlist(df2), dim=c(5,2,2),

dimnames = list(rownames(df1), colnames(df2))))

# c(5,2,2)中最后一个2是指第二个数组

### 2.apply

m1<-matrix(c(1:10),nrow = 5,ncol = 6)

a\_m1<-apply(m1,2,sum)

# 2是margin,margin=1(行)或者2(列)，sum是指对m1中每一列的数据进行求和

### 3.lapply

movies<-c("A","B","C")

movies\_lower<-lapply(movies,tolower) # 将小写功能应用于向量

# 将包含电影名称的向量 movies 中的每个元素转换为小写字母，并将结果存储在一个新的列表 movies\_lower 中

films\_lower<-unlist(movies\_lower)

# 使用 unlist() 函数将存储在列表 movies\_lower 中的元素转换为一个单独的向量。list函数用于将列表转换为向量。

dt<-cars

lmn\_cars<-lapply(dt,min)

# 使用lapply()函数从cars数据集中提取每列的最小值，并将结果存储在一个列表lmn\_cars中

### 4.sapply

avg<-function(x){(min(x)+max(x))/2}

fcars<-sapply(dt,avg)

# 对cars数据集中的每一列应用一个自定义函数avg，该函数计算每列数据的最小值和最大值的平均值，并将这些平均值存储在向量fcars中

### 5.tapply

data(iris)

tapply(iris$Sepal.Width, iris$Species, median)

# 用了tapply()函数。第一个参数iris$Sepal.Width是要进行分组的数值向量，第二个参数iris$Species指定了分组的依据，

# 按照鸢尾花的种类进行分组。最后一个参数median表示对每个组应用的函数，即计算每个组的中位数。

### 6.xlsx类型文件

any(grepl("xlsx",install.packages())) # 确定是否有安装这个包

library(xlsx)

write.xlsx(emp.data,file=文件地址"data/employee.xlsx",

col.name=TRUE,row.name=TRUE,sheetName="sheet2",append=TRUE)

# emp.data是要写入到 Excel 文件中的数据框（data frame）

# file="data/employee.xlsx"表示要写入的 Excel 文件的文件路径和文件名。在这个例子中，文件将被写入到名为 "employee.xlsx" 的 "data" 文件夹中。

# col.name=TRUE表示在写入数据时包括列名。

# sheetName="sheet2"表示要将数据写入到 Excel 文件中的工作表的名称。在这个例子中，数据将被写入到名为 "sheet2" 的工作表中。

# append=TRUE表示是否将数据追加到现有的 Excel 文件中，设置为 TRUE 表示追加数据。如果设置为 FALSE，则会覆盖现有文件中的数据。

excel\_data<-read.xlsx("data/employee.xlsx",sheetIndex=2)

# sheetIndex=2表示要读取的工作表在 Excel 文件中的索引号，代码尝试读取索引号为2的工作表

write.csv(excel\_data,"data/employee.csv")

# 将名为 excel\_data 的数据写入到名为 "employee.csv" 的 CSV 文件中

### 7.数据检查

# is.na() 函数用于检查单个数值和向量中的元素是否是缺失值

surveys\_sml<-surveys %<% filter(weight<5)%>% select(species\_id,sex,weight)

# 从原始的 surveys 数据集中选择出体重小于5的记录，并且只保留了 species\_id、sex 和 weight 这三个变量，将处理后的结果保存到了 surveys\_sml 中

**（四）3月22日课堂代码笔记**

### 1.下载数据文件

download.file("tinyurl.com/dcmac2017dec/data/surveys\_wide.csv",dest="D:/Ryuyans huju/SA23008215/surveys\_wide.csv")

surveys\_wide<-read.csv("D:/Ryuyanshuju/SA23008215/surveys\_wide.csv")

### 2.gather功能

surveys\_gather<-surveys\_wide %>% gather(key = year, value = plot\_id)

tail(surveys\_gather)

### 3.spread功能

spread(surveys\_gather,key = year, value = plot\_id)

surveys\_gw<-surveys

### 4.绘图

ggplot(data=surveys\_wide, aes(x=weight,y=hindfoot\_length))

ggplot(data = surveys, aes(x = species\_id, y = weight)) +

geom\_boxplot(alpha = 0) +geom\_jitter(alpha = 0.3, color = "green")

**（五）3月27日课堂代码笔记**

### 1.cart for classification

class<-as.factor(c(0,0,0,0,0,1,1,1,1,1))

x1<-c(4,4,5,5,5,5,3,5.6,6,6.5,6.2,5.9)

x2<-c(9,10,11,10,9,8,7,8,7,8)

df<-cbind.data.frame(class,x1,x2)

# 把这三个数据集以列的方式组合成一个数据框

plot(x1,x2,col="white")

points(x1[class=="0"],x2[class=="0"],col="blue",pch=19)

points(x1[class=="1"],x2[class=="1"],col="red",pch=19)

predictor1test<-seq(from=3,to=7,by=0.1)

predictor2test<-seq(from=7,to=11,by=0.1)

calculatep<-function(i,index,m,k){if(m=="L")

{Nm<-length(df$class[which(df[,index]<=i)])

count<-df$class[which(df[,index]<=i)]==k}else

{Nm<-length(df$class[which(df[,index]>i)])

count<-df$class[which(df[,index]>i)]==k}

p<-length(count[count==TRUE])/Nm return(c(p , Nm))}

# 根据阈值 i、变量索引 index 和左侧或右侧标志 m 来确定所计算的样本数量 Nm，然后统计满足条件的类别 k 的数量，并计算该类别的概率。最后返回一个包含概率和样本数量的向量。

calculategini<-function(x,index){gini<-NULL for(i in x)

{pl0<-calculatep(i,index,"L",0)

pl1<-calculatep(i,index,"L",1)

ginil<-pl0[1]\*(1-pl0[1])+pl1[1]\*(1-pl1[1])

pr0<-calculatep(i,index,"R",0)

pr1<-calculatep(i,index,"R",1)

gini<-rbind(gini,sum(ginil\*pl0[2]/(pl0[2]+pr0[2]),ginir\*pr))}return(gini)}

# 通过在给定的阈值范围内不断尝试不同的阈值来计算每个阈值对应的基尼系数，以便于选择最佳的阈值来划分数据。

**（六）3月29日课堂代码笔记**

利用caret包训练和调优一个模型，参考<https://www.machinelearningplus.com/machine-learning/caret-package/>

### 1.caret包

install.packages(c('caret', 'skimr', 'RANN', 'randomForest', 'fastAdaboost', 'gbm',

'xgboost', 'caretEnsemble', 'C50', 'earth'))

install.packages("caret")

library(caret)

orange <- read.csv('https://raw.githubusercontent.com/selva86/datasets/master/orange\_juice\_withmissing.csv')

orange<-read.csv()

### 2.创建训练和测试数据集

set.seed(100)

trainRowNumbers <- createDataPartition(orange$Purchase, p=0.8, list=FALSE)

# 将源数据集中的 Y 变量和应进入训练的百分比数据作为参数作为输入

# 返回应构成训练数据集的行号，防止将结果作为列表返回

trainData <- orange[trainRowNumbers,]

testData <- orange[-trainRowNumbers,]

### 3.提取训练数据集数据存储为x,y变量

x = trainData[, 2:18]

y = trainData$Purchase

### 4. 对数据的摘要统计分析

install.packages("skimr")

library(skimr)

skimmed <- skim\_to\_wide(trainData)

skimmed[, c(1:5, 9:11, 13, 15:16)]

### 5. 利用KNN插补方法对训练数据中的缺失值进行处理，并生成预处理模型

preProcess\_missingdata\_model <- preProcess(trainData, method='knnImpute')

preProcess\_missingdata\_model # 预测缺失值

install.packages("RANN")

library(RANN)

trainData <- predict(preProcess\_missingdata\_model, newdata = trainData)

anyNA(trainData) # 所有缺失值均已成功插补

**（七）4月3日课堂代码笔记**

在R环境中设置Python解释器，通过retriever包来进行数据检索和管理

install.packages('reticulate') # Install R package for interacting with Python

library(reticulate)

use\_python("C:/Users/22651/AppData/Local/r-miniconda/python.exe")

# 指定Python解释器的路径

py\_config()

py\_install("retriever")

retriever <- import("retriever")

reticulate::install\_miniconda() # Install Python

reticulate::py\_install('retriever') # Install the Python retriever package

install.packages('rdataretriever') # Install the R package for running the retriever

library(rdataretriever)

rdataretriever::get\_updates() # Update the available datasets

reticulate::py\_config()

install.packages("rdataretriever", dependencies = TRUE)