生态学研究范式：经验科学范式→假说-验证范式→理论、实验、模拟多范式→数据科学范式

**一、R语言基础**

**1.生成数据**

(1) read. Table()。通常接受一个文件输入，并从文件读取数据。函数的控制参数较多,常见参数为header,指示是否读取表头(header = TRUE或header= FALSE)。

(2) c()。接受一个范围或逗号分隔的数值列表，并用它创建一个向量。

(3) matrix()。接受一组向量或一个范围值，创建一个矩阵。常见参数nrow(指定矩阵中的行数)、ncol(指定列数)和“”。当byrow= TRUE时，数据应该逐行读取，当byrow=FALSE时,逐列读取数据。

(4) seq()。接受一个开始值和结束值，产生-一个数值序列。主要参数是by,它表示序列中的步长。

(5)数据转换。as. data. frame()、as. vector()、as. list()、as. matrix()接受-一个输入，并强制将该值分别转换为数据框、向量、列表和矩阵。

(6)apply家族。用于在特定数据类型(列表、矩阵等)的项上重复执行，解决数据循环处理。其中，常用的是apply()和lapply()函数。

**2.各种运算符**

(1)“+”“一”“\*”“/”。用于数字、向量或矩阵的加、减、乘、除运算。

(2)“\* \*”或“^”。作为指数函数用，运算符前面的值是基，后面的值是幂。

(3)“%%”。模运算符，运算符前面的值除以运算符后面的值，并返回两个值相除的余数。

(4)“% \* %”。矩阵乘法符号,它的前、后各有一个矩阵，此操作将返回一个矩阵,该矩阵是前、后两个矩阵的乘积。

(5)“~”。用于R定义一个关系模型。一个简单的线性回归的格式是y~x,y~x+ z表示多元回归模型。为了得到这样一个模型的系数,使用lm作为模型参数。

(6)常见逻辑运算符:

①! =表示“不等”;

②=表示“相等”;

③>表示“大于”;

④>=表示“大于等于”;

⑤ | 表示“或”;

⑥ & 表示“与”。

**3. 检验函数**

(1)t-test。对两个数据样本进行Student’s t-test检验，可接收多个参数，其中最基本的是数据样本的两个向量，以及var.equal(指定样本方差是否相等)。

(2)var.test()。接受样本向量作为输入参数，执行F-test，比较两个数据样本的方差。

**4.绘图函数**

(1) plot()。 接受x和y坐标的向量,并在图形窗口中绘制它们。常用参数包括xlab和ylab,为两轴标签，main和sub分别为主、副图标题，type为控制绘图类型参数(p绘制点图，1绘制线型图，p与1组合绘制点线图)。

(2) pairs()。接受数字矩阵或数据框输人，并生成一系列由输人中任意两列构成的配对图。

(3)points()。接受x和y坐标的向量，并在图形窗口中绘制指定的点。

(4) abline()。接受要绘制的直线(a和b)的斜率和截距，并在图形窗口上绘制直线。例如，函数abline(lm(y~ x))可用于在向量y和x之间绘制回归线。

(5) hist()。接受个向量，并绘制一个直方图,其中,x 轴上的值的频率在y 轴上。

(6) boxplot()。 生成多个输人的比较图,对于比较多个数据样本特别有用。

**5.数据操作**

(1) dplyr。 可以对数据集做subset、summarize、rearrange、join等处理。

(2) tidyr。 利用gather和spread函数将数据集转化成格式更工整的数据集。

(3) stringr.对字符串类型的数据进行正则表达式处理的工具。

(4) lubridate。处理日期和时间类型数据的工具。

(5) httr。 处理http链接的工具集合。

**6.可视化**

(1) ggplot2。功能强大的绘图工具包。

(2)ggvis。一个可以做基于Web的交互可视化工具包。

(3)rgl。在R中做3D交互可视化包。

(4) htmlwidgets。一个在R中快速建立基于JavaScript内核的交互可视化工具包。

(5) googleVis。利用Google Chart工具在R中做数据可视化。

**7.数据建模**

(1)统计分析与模型。包括car(方差分析)、multcomp(多重比较分析)、mgcv(广义相加模型)、lme4/ nlme(线性/非线性混合效应模型)。

(2)机器学习建模。包括rendomForest(随机森林模型)、caret(模型工具包)。

**8.编程**

(1) Shiny。用于R交互可视化。

(2) R Makdown。处理数据分析报告的工具。

(3) jupyter notebook。提供编程环境。

**R基本操作**

**1.读取和保存数据**

最常用的是read. csv()和write. csv()函数。read. csv()可读取矩阵或表类型的数据，并将数据(包括表头)作为数据框读到R中,经过处理可用write. csv()把数据保存起来，两个函数的语法如下:

data <- read. csv(".. /Data/ACS. csv", header = TRUE) (header = TRUE意思是在读取数据时将csv文件的第一行视为列名。)

write. csv(data, "../Data/data1. csv")

**2.数据类型及其操作**

在R中，常用数据类型为向量、矩阵、列表和数据框。

**向量**

常用来存储一维数据，包括数字向量(存储整数或小数)和字符向量(存储字符串)。

a<- c(1:10) #创建1~10的数字向量

b <- seq(1,10,by=2) #生成序列1到10,步长为2的序列

h<-"hello" #创建字符向量或字符串

**矩阵**

经常被用来保存多维数据,A %\*% B表示矩阵A与B相乘。

m <- matrix(c(1:10)，nrow= 2,ncol= 5,byrow= FALSE) #先由c()创建1到10个数的向量，

然后按照列(byrow= FALSE)从左到右、再从上到下排列2x5矩阵

dim (m)井查看矩阵维度，也可用attributes (m)

可用dim()将一个向量转换为一个矩阵。

age<- C (23，44, 15，12，31, 16) #一个向量

dim (age) <-c (2, 3)井从一个向量创建一个矩阵

class (age)

使用cbind()和rbind()函数连接两个向量。但要确保两个向量都有相同的元素数，否则，它将返回NA值。

x<-c(1,2,3,4,5，6)

y<-c (20，30, 40，50，60，70)

cbind (x, y) #按列合并

**列表**

它是其他对象的集合，单个列表可以同时包含一个字符向量(字符串)、一个数字向量和一个矩阵。

my\_ list<- list(name = "John", age = 22, lucknumbers = c(21.32.1))

#用list()函数创建列表

my\_ list #查看列表组分

my\_list$ lucknumbers #查看子列表组分

my\_ list[3] #等同于my list $ lucknumbers

my lst[[2]]#查看子列表中的元素

**数据框**

带有dafame的列表，组件可以是向量矩阵、列表或其他，允许多个类型数据放在一起，添加到数据框中的每个组件必须具有相同的行数。在R中，主要数据格式为数据框，理解数据框是非常重要的。

df<- data. frame (name = c( "ash", "jane" ，"paul", "mark"), score = c(67, 56，87，91))

#用data. frame ()函数创建数据框

str (df) #查看数据框结构

df[1:2,2]<- NA #对数据框1~2行的第2列赋值为NA

is.na(df) #查询有没有缺失值

table (is. na (df))

**缺值会妨碍数据集中的正常计算。**上述数据集有两个缺少的值,不能直接计算均值。这时添加参数na. rm= TRUE,告诉R忽略NA并计算所选列(Score值)中剩余值的平均值。若要删除数据框中具有NA值的行，就可以使用na. omit:

mean (df $ score)

mean (df$ score, na.rm = TRUE)

new df <- na. omit (df)

new\_ df

**索引并取值**

R允许方便地访问大数据集，从而提取样本。索引运算符“口”不仅可以用于从向量。列表、矩阵或数据框中选择单个元素，而且可以用于执行其他任务。

a<- c(1, 2, 3, 4, 5)

a<-[3]井从向量a中提取第3个元素

a<-[-3] #从向量a中删除第3个元素

a[2:4] #从向量a中提取2~4元素

b = matrix(c(1:9)，nrow= 3, ncol= 3)

b[2,]#从矩阵b中提取第2行

b[,2] #从矩阵b中提取第2列

b[2,2]#从矩阵b中提取第2行、第2列

**数据转换**

R提供一-整套函数，用于将数据从--种类型转换为其他类型。如数据之间的转换，可执行下面的R代码:

b = matrix(c(1:4), nrow=2, ncol = 2,byrow= TRUE)

class(b)

df <- as. data. frame(b)

class(df)

v <- as. vector(b)

class(v)

list <- as. list(b)

class(list)

**2.简单运算**

**加、减、乘、除运算**

R console可以用作交互式计算器,从事些基本的计算或各种计算组合。相关代码如下:

10-(2+3) #加、减

(3\*8)/(2\*3)#乘、除

log(12) #对数

sqrt (121) #开方

**创建变量**

使用<-或=符号创建变量。如创建一个变量x来计算7与8的和，可以写为:

x< - 7 +8 #创建变量(必须是字符,或字符与数字组合)

x #输出结果

**3.条件与循环**

对数据进行操作的另一类重要手段是使用R控制语句，包括条件执行(if表达式)和重复执行(for和while循环)。此外，使用apply系列函数来执行重复。

**if, else**

此结构用于测试个 条件。其语法如下:

if (< condition > ) {

#满足条件下执行的任务

} else {

井否则执行的任务

}

N<- 10井初始化一个值

if (N\*5> 40) { #检测条件

print ("This is easy!")

}else{

print ("It is not easy!")

}

**for**

当执行循环时使用此结构，通常用于迭代对象的元素(列表、向量)，其语法如下:

for (<search condition>) {

#搜索条件执行

}

y<-c(99, 45, 34, 65, 76, 23) #初始化一个向量

for(in1: 4){ #打印这个向量的第一个数字

print (y [i])

}

**while**

从测试一个条件开始，并且只在条件被发现为真时才执行，一旦循环被执行，条件就会再次被测试，直到不符合条件才停止。其语法如下:

v<- c( "female")

cnt<-5 #初始化一个条件

while (cnt<7) { #检测条件

print(v)

cnt=cnt+ 1

}

**apply 家族**

在许多情况下，频繁使用循环会导致代码出现混乱，应用apply()函数(用于矩阵)和lapply函数(用于列表),分别对矩阵和列表的元素执行重复。关于apply()和lapply()使用的例子如下:

x= matrix(c(1:16)), nrow = 4, ncol=4, byrow= TRUE) #创建一个4x4矩阵,按行顺序排

apply(x, 1, max) #第2个参数“1”指按行找最大值，并返回一个向量

apply(x, 2, max) #第2个参数“1”指按行找最大值，并返回一个向量

y = list(c(1:4),c(5:8),c(9:12),c(13:16))

lapply(y,max) #在每个子列表中找到最大值元素,并返回元素值

sapply(y, max) #与lpply相似，但返回的不是最大元素，而是最大元素构成的向量

**4.数据可视化**

R具有强大的绘图能力，可完成可视化。下面是一个简单的用R中内置函数实现数据的可视化代码:

x = c(1,3,4,5,8)

y = c(2,4,7,3,9)

plot(x,y,type= "l",col= "black" ，xlab= "X" ,ylab= "Y"，

font. main= 2, font. lab= 1,pch= 19)

**5.R自带数据集**

在R中带有不少数据集，一.些R包也自带一些数据集,这些自带的数据集主要用来说明R或R包的一些功能可以免费使用，不涉及版权问题。初学者可以利用这些数据集练习R语言。

**R中的ade4包中内置Doubs数据集**

data() #查看R内存中datasets 包中的数据集

data( package = packages( al available = TRUE))井查看所有数据集

data( package = "ade4") #仅查看ade包中数据，先要加载ade4包

data(Doubs) #加载ade包中Doubs数据集,在R environment中检查是否加载成功

help(' "Doubs")或?? Doubs# 在Graphical Output中查看数据集的信息

一般采用write. table 或write. csv可将数据保存到当前工作目录。因Doubs数据集包括3个文件,需要分别提取并保存。

Doubs. env <- Doubs $ env #提取Doubs中环境变量

View(Doubs. env)井查看环境数据

Write.csv(Doubs.env, file = "Doubs\_env. csv") #导出数据,以csv格式保存在当前工作目录或其他路径(write. csv(Doubs.env, file = "path Doubs en.csv"))

Doubs.spe <- Doubs $ specices #提取物种数据

View(Doubs. spe)

write. csv(Doubs. spe, file = "Doubs spe. csv")

Doubs. spa <- DoubsS xy #提取空间数据

View( Doubs. spa)

Write.csv(Doubs.spa, file = “Doubs\_spa.csv”)

**6.tidyverse包**

tidyverse 是一个 R 语言中的集合包，包含了一组用于数据科学的高效、流行的包。这些包旨在简化数据处理和分析流程，并提供一致的语法和接口。tidyverse 包括了以下主要包：

ggplot2：用于数据可视化，基于图形语法。

dplyr：用于数据操作，如过滤、排序、聚合等。

tidyr：用于整理数据，使其整洁。

readr：用于读取数据文件，如 CSV、TSV 等。

purrr：用于函数式编程，提供了一组工具来处理函数和迭代。

tibble：用于创建增强的数据框（tibbles），更适合 tidyverse 的操作。

stringr：用于处理字符串。

forcats：用于处理因子变量

(1)要安装和加载 tidyverse 包，可以使用以下命令：

install.packages("tidyverse")

library(tidyverse)

(2)使用 "readr" 读取 CSV 文件：

library(tidyverse)

data <- read\_csv("../Data/ACS.csv") # 读取 CSV 文件

(3) 使用 "dplyr" 进行数据操作：

filtered\_data <- data %>%

filter(Age > 25) # 筛选年龄大于25的人

grouped\_data <- data %>%

group\_by(Gender) %>%

summarize(mean\_age = mean(Age, na.rm = TRUE)) # 按年龄分组并计算平均年龄

(4) 使用 "tidyr" 整理数据：

long\_data <- data %>%

pivot\_longer(cols = starts\_with("variable"), names\_to = "variable", values\_to = "value") # 将宽数据转换为长数据

(5) 使用 "ggplot2" 进行数据可视化：

ggplot(data, aes(x = Age)) +

geom\_histogram(binwidth = 5) +

labs(title = "Age Distribution", x = "Age", y = "Count") # 绘制年龄分布的柱状图

(5) 使用 "stringr" 处理字符串：

data <- data %>%

mutate(name\_length = str\_length(Name)) # 计算名字的长度

(6) 使用 "forcats" 处理因子变量：

data <- data %>%

mutate(Gender = fct\_relevel(Gender, "Female", "Male")) # 重排序因子水平

**二、数据处理**

**1.模型函数**

要确定x与y之间的关系，可建立统计模型,建模步骤大致如下:

第一步,将数据导入R，绘制散点图,并根据散点图，执行如下代码:

data <- read. csv( "xy. csv") #读取并查看数据

x<- data$x

y<- data$ y

plot(y~ x)#散点图检查y与x之间关系模型

abline(lm(y ~ x))

第二步，建模前,需要剔除异常值，而且数据要服从正态分布，否则影响模型的可靠性。执行如下代码:

par(mfrow = c(1, 2)) #将绘图区分为2列

boxplot(data$ x，main= "x"，sub= paste( "Outier rows: ",

box plot. stats(data$ x) $ out))

#统计离群值、即boxplot. stats(data$ x) $ out，并输出$out，并将该值作为副标，其中paste()是将字符连接起来,即Outier rows; out

boxplot(datas$ y, main= "y"，sub paste( "Outier rows: ", boxplot. stats(data$ y) $ out))

library(e1071) # 用于检测正态分布的包

plot(density(data$ x)，main= "Density Plot: x", ylab= "Frequency",

sub= paste( "Skewness:", round(e1071::skewness (data$x)，2))

#绘制密度分布图.round ()函数部分是计算偏倚e1071::skewness (data$x)后保留2位小数点，然后round再进行一次四舍五入。

plot(density(data$ y), main= "Density Plot:y "，ylab= "Frequency"，

sub = paste( "Skewness", round(e1071:::skewness data$y), 2))

第三步,在确定函数式之后，根据观察数据,通过最小二乘法求出模型中的参数a和b，得到一个线性回归方程。在R中,执行如下代码:

cor(x, y)井计算相关系数

linearMod <- lm(y ~ x) # 构建线性模型

print( linearMod)

第四步,对模型进行评估,执行如下代码:

summary( linearMod)井模型显著性检验

**2.数据框操作**

**导入和保存数据框**

library(tidyverse)

sample\_data <- tibble(

) #创建数据框

write\_csv(sample\_data, "sample\_data.csv") # 保存为csv文件

data <- read\_csv("sample\_data.csv") # 保存csv文件

**检查数据结构**

str(data) # 查看数据结构

summary(data) # 总结数据

**检查缺失值**

missing\_data\_columns <- data %>%

summarize\_all(~ sum(is.na(.))) # 检查每一列的缺失值

print(missing\_data\_columns) #展示含有缺失值的列

missing\_data\_rows <- data %>%

mutate(row\_missing = rowSums(is.na(.))) %>%

filter(row\_missing > 0) # 检查每一行的缺失值

print(missing\_data\_rows) #展示含有缺失值的行

**从列中提取值或选择/添加列**

ages <- data$Age

print(ages) # 提取 'Age' 列

selected\_data <- data %>%

select(Name, Age)

print(selected\_data) # 选择 'Name' and 'Age' 列

data <- data %>%

mutate(Weight = c(70, 55, 75, 60))

print(data) # 添加新列 'Weight'

**宽表格转换为长表格**

long\_data <- data %>%

pivot\_longer(cols = c(Height, Weight), names\_to = "Measurement", values\_to = "Value")

print(long\_data)

**数据可视化**

ggplot(data, aes(x = Age)) +

geom\_histogram(binwidth = 5, fill = "blue", color = "black") +

labs(title = "Age Distribution", x = "Age", y = "Count") #使用条形图可视化年龄分布

ggplot(long\_data, aes(x = Gender, y = Value, fill = Measurement)) +

geom\_boxplot() +

labs(title = "Height and Weight by Gender", x = "Gender", y = "Measurement Value") +

facet\_wrap(~ Measurement, scales = "free") # 使用箱线图按性别可视化身高和体重

**3.常用树模型**

在生态学研究中，解释变量通常比较多。例如，鸟类的体重与年龄、性别和取食集团(Guild)都有关，植被类型与坡向、降雨和土壤养分有关等等。在大数据时代，数据量大，数据类型更加多样，变量之间存在复杂交互作用，依靠传统统计分析很难完成。机器学习中的树模型比较直观，且模型不受预测变量类型、标度、缺省值影响，特别适用于生态学建模，备受生态学研究者欢迎。

树模型包括决策树(Decision Tree, DT)、随机森林( Random Forest, RF) 、提升回归树( Boosting Regression Tree, BRT)等。DT是基础模型,RF基于许多决策树，最终预测是从许多树平均得出的。BRT的起点也是决策树，与随机森林不同，每棵树不是相互独立的，而设置为前棵树 的残 差(预测误差) ，通过对建模不良特征给 予更多权重，逐步减少模型与观察到的目标数据之间的差异。下面具体说明各自算法。

**分类树**

分类树（Classification Tree）是一种决策树，用于对数据进行分类。分类树通过一系列决策节点将数据划分成不同的类别。每个节点根据某个特征的值将数据分成两个或多个子集，这样的过程从根节点开始，逐步递归，直到达到叶节点。叶节点表示分类结果或类别。

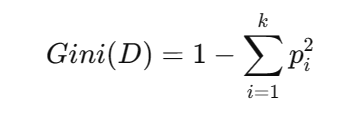
**CART算法**

CART（Classification and Regression Trees）是一种构建决策树的算法，适用于分类任务和回归任务。对于分类任务，CART构建的是分类树；对于回归任务，CART构建的是回归树。以下是CART算法的关键步骤和原理：

**树的构建**

①选择最佳分裂特征和分裂点：在每个节点，CART算法会遍历所有可能的分裂特征和分裂点，选择能够最大化某个指标（如基尼系数或熵）的分裂方式。对于分类问题，CART通常使用基尼系数作为选择标准。

②基尼系数（Gini Index）：衡量数据集的不纯度，计算公式如下：



其中，pi是第i类样本的比例，k是类别数。基尼系数越小，节点的不纯度越低。

③递归分裂：选择最佳分裂后，递归地对子节点继续进行分裂，直到满足停止条件（如达到最大树深度、节点样本数小于最小样本数或不再能显著降低不纯度）。

**树的剪枝**

①预剪枝（Pre-pruning）：在构建树的过程中，通过设定参数（如最大深度、最小样本数）来限制树的生长，防止过拟合。

②后剪枝（Post-pruning）：在树构建完成后，通过对叶节点进行合并，减少树的复杂度。这通常是通过交叉验证来确定最佳的剪枝点。

**使用R中的"rpart"包来构建分类树和应用CART算法：**

library(rpart)

library(rpart.plot) # 加载必要的包

data(iris) # 加载示例数据集

model <- rpart(Species ~ ., data = iris, method = "class") # 构建分类树模型

rpart.plot(model) # 可视化分类树

print(model) # 打印模型信息

predictions <- predict(model, newdata = iris, type = "class") # 预测新数据

accuracy <- sum(predictions == iris$Species) / nrow(iris) # 计算准确率

print(paste("Accuracy:", accuracy))

**CART算法的优缺点**

**优点：**

直观易懂：决策树的结构类似于人类的决策过程，便于解释和理解。

不需要特征缩放：决策树对特征的尺度不敏感，不需要进行标准化或归一化处理。

可以处理多种数据类型：既可以处理数值型特征，也可以处理分类特征。

**缺点：**

容易过拟合：尤其是当树的深度较大时，决策树容易对训练数据过拟合，导致在新数据上的泛化能力较差。

不稳定：对数据的小变化敏感，数据的微小变化可能导致完全不同的树结构。

无法捕捉线性关系：决策树更擅长处理非线性关系，可能无法有效捕捉数据中的线性关系。

**最小二乘回归树**

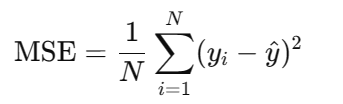
最小二乘回归树（Least Squares Regression Tree，LSRT）是一种用于回归任务的决策树方法。与分类树不同，最小二乘回归树的目标是预测连续变量的值，而不是分类。最小二乘回归树的基本原理是通过递归地分割数据集，使得每个叶节点上的数据尽可能具有相似的目标值。

**基本原理**

最小二乘回归树的核心思想是选择使得数据集的目标变量方差最小的分割点，从而使每个叶节点中的数据更加集中。具体步骤如下：

**树的构建**

①选择最佳分割特征和分割点：在每个节点上，遍历所有可能的分割特征和分割点，选择能最小化均方误差（MSE）的分割方式。均方误差的计算公式如下：



其中yi是实际值，y^是预测值，N是样本数量。

②递归分割：选择最佳分割后，递归地对子节点继续进行分割，直到满足停止条件（如达到最大树深度、节点样本数小于最小样本数或分割不再显著减少误差）。

**树的剪枝**

预剪枝（Pre-pruning）：在树的构建过程中，通过设定参数（如最大深度、最小样本数）来限制树的生长，防止过拟合。

后剪枝（Post-pruning）：在树构建完成后，通过对叶节点进行合并，减少树的复杂度。通常通过交叉验证来确定最佳的剪枝点。

**使用R中的"rpart"包来构建最小二乘回归树：**

library(rpart)

library(rpart.plot) # 加载必要的包

data(mtcars) # 加载示例数据集

model <- rpart(mpg ~ ., data = mtcars, method = "anova") # 构建回归树模型

rpart.plot(model) # 可视化回归树

print(model) # 打印模型信息

predictions <- predict(model, newdata = mtcars) # 预测新数据

mse <- mean((predictions - mtcars$mpg)^2) # 计算均方误差

print(paste("MSE:", mse))

**最小二乘回归树的优缺点**

**优点：**

直观易懂：回归树的结构类似于人类的决策过程，便于解释和理解。

不需要特征缩放：回归树对特征的尺度不敏感，不需要进行标准化或归一化处理。

灵活性：可以处理非线性关系，适用于各种复杂的回归问题。

**缺点：**

容易过拟合：尤其是当树的深度较大时，回归树容易对训练数据过拟合，导致在新数据上的泛化能力较差。

不稳定：对数据的小变化敏感，数据的微小变化可能导致完全不同的树结构。

局部最优：回归树使用贪心算法选择分割点，可能无法找到全局最优的树结构。

**随机森林**

集成学习(Ensemble Learning)是使用多个学习器进行学习，然后整合各个学习器的结果得到更好学习效果的机器学习算法。主要有两类算法:

(1) bagging集成学习。即从总体数据集中采用有放回抽样得到N个数据集，也称为bagging方法，每个数据集训练出一-个模型，然后采用N个模型预测投票(分类问题)或计算均值(回归问题输出最后结果。随机森林采用bagging集成学习算法。

(2) boosting 集成学习。首先根据训练集用一个初始权 重训练出第一个弱学习器，再根据第一个弱学习器误差率调整样本权重，加大误差大的训练样本的权重，然后利用调整权重后样本训练第二个学习器，如此重复，直到弱学习器达到预先指定数，最后通过均值输出结果。提升回归树采用boosting集成学习算法。

随机森林是通过自助法( boot-strap)从样本N中有放回地抽样，选择N个样本(有样本可能相同),基于Gini指数法选择特征变量，构建决策树,重复M次，构建出M棵决策树，最终形成一个含有M棵决策树的森林。

**使用R中的"randomForest"包来构建随机森林模型。**

library(randomForest)

library(ggplot2) # 加载必要的包

data(iris) # 加载示例数据集

set.seed(42)

rf\_model <- randomForest(Species ~ ., data = iris, ntree = 100, mtry = 2, importance = TRUE) # 构建随机森林分类模型

print(rf\_model) # 打印模型信息

predictions <- predict(rf\_model, newdata = iris) # 预测新数据

accuracy <- sum(predictions == iris$Species) / nrow(iris)

print(paste("Accuracy:", accuracy)) # 计算准确率

importance <- importance(rf\_model)

varImpPlot(rf\_model) # 可视化特征重要性

**随机森林的优缺点**

**优点：**

高准确性：通过集成多棵决策树，随机森林通常能提供比单棵决策树更高的预测准确性。

抗过拟合：通过Bootstrap抽样和随机特征选择，随机森林能有效减少过拟合，尤其在高维数据和大数据集上表现良好。

特征重要性：随机森林可以计算特征的重要性，帮助我们理解哪些特征对预测结果影响最大。

鲁棒性：对数据中的噪声和缺失值不敏感，具有较好的稳健性。

**缺点：**

计算复杂度高：由于需要构建和评估多棵树，随机森林的计算和内存消耗较大，训练时间较长。

可解释性差：虽然比单棵决策树更稳定和准确，但随机森林的复杂性使得它的可解释性较差。

**梯度下降算法**

梯度下降（Gradient Descent）是一种优化算法，常用于最小化（或最大化）目标函数。它是机器学习和深度学习中最常用的优化方法之一，特别适用于训练神经网络和其他参数化模型。

梯度下降算法的基本原理是通过迭代地调整模型参数，以使目标函数的值逐渐减小。目标函数通常是一个损失函数（如均方误差、交叉熵等），反映了模型预测与真实值之间的差异。

以下是使用 R 语言实现梯度下降算法的一个简单示例，目标是最小化一个二次函数：

f <- function(x) {

return(x^2)

} # 定义目标函数

grad\_f <- function(x) {

return(2 \* x)

} # 定义目标函数的梯度

gradient\_descent <- function(starting\_point, learning\_rate, iterations) {

x <- starting\_point

for (i in 1:iterations) {

gradient <- grad\_f(x)

x <- x - learning\_rate \* gradient

print(paste("Iteration:", i, "x:", x, "f(x):", f(x)))

}

return(x)

} # 梯度下降算法

starting\_point <- 10

learning\_rate <- 0.1

iterations <- 20 # 参数设置

optimal\_x <- gradient\_descent(starting\_point, learning\_rate, iterations)

print(paste("Optimal x:", optimal\_x)) # 执行梯度下降

**三、数据挖掘**

**Rattle**

数据挖掘是从数据中发现有意义的见解和知识，并将这种发现表示为模型。因此，数据挖掘就是一个建立模型,用来帮助理解世界，也可用来预测未来。如今，数据挖掘是一门利用计算机科学、机器学习和统计学等复杂技能的学科。

现已发展了一些数据分析与计算工具，如NESCent (http://www. nescent. org/)、NCEAS( https://www. nceas. ucsb. edu/)、SESYNC( https://www. sesync. org/)。另外，通过EcologicalDataWiki,可帮助了解各种数据集适合做哪些分析，以及如何严格地进行分析。software carpentry( https://software-carpentry. org/)、DataONE还提供专门培训教程。

Rattle( R analytical tool to learn easily)是R的一个程序包，一个界面友好的数据挖掘工具，从数据探索→数据清洗→特征工程→模型构建→模型评估→模型应用等步骤，Rattle给出了完整的解决方案。但是，Rattle不直接支持时间序列分析。另外,对于空间数据分析，Rattle提供的帮助也是有限的。对于初学者而言,可以利用Rattle学习数据挖掘的一些知识。

Rattle集成了40多个优秀程序包。此外，Rattle本身还提供了不少Rattle内部使用的实用函数。通过学习Rattle, 可以了解数据处理、机器学习建模等操作的各种常用R包。下面是关于Rattle常用包的介绍:

(1)导人数据。Rattle集成了data. table、readr、RMySQL、sqldf、 jsonite等包,用于导人任何格式数据，如. txt、csv.. json、. sql等,以便快速导人数据文件。

(2)数据可视化。Rattle也内置函数用于创建简单图形,但创建高级图形时，应安装专门的绘图包ggplot2。

(3)数据操作。Rattle具有一个非常好的数据操作包集合，这些包允许快速执行基本和高级计算，包括dplyr、plyr、tidyr、lubridate、stringr。

(4)机器学习建模。Rattle 集成了些建模包， 如randomForest、 rpart、 gbm等，能够满足创建机器学习模型的每一个需求。

**以下是如何使用 "rattle" 包构建一个简单决策树模型的代码示例：**

library(rattle)

library(rpart)

library(rpart.plot) # 加载必要的包

data(iris) # 加载数据集

rattle() # 打开 rattle GUI 进行交互操作

set.seed(42) # 设置随机种子以确保结果可重现

model <- rpart(Species ~ ., data = iris) # 构建决策树模型

print(model) # 打印模型

rpart.plot(model) # 可视化决策树

predictions <- predict(model, newdata = iris, type = "class") # 预

accuracy <- sum(predictions == iris$Species) / nrow(iris)

print(paste("Accuracy:", accuracy)) # 计算准确率

**Caret**

生态学研究中常用树模型分析预测解释和目标变量的关系，目前已发展了不少包，用于构建树模型方要用户选择使用。但不同包的语法和使用的参数不尽相同，不利于比较各种算法及其结果，建议尽可能采用caret包，它为建模提供了统框架。

**树模型与caret 包**

构建决策树可用cre tre tparr等包，构建随机森林可用ranger rndomFrors包，用gbm包构建BRT模型。例如，基于tpart构建分类树模型,执行如下代码:

DT model <- rpart(target~. data = data, na. action = na. rpart.

method = "class" ，

parms= list(split = "infomation"))

主要参数及其含义如下:

(1) na. action。 默认为na. rpart,表示剔除target 和预测变量中有缺失的obs。

(2) method。分类树选择class或poisson,回归树选择anova,生存分析选择exp。

(3) parms。此参数只适合分类树,对于method = class, parms = list (split=“information"(ID3算法)或split =“gini"(CART))

另外，基于randomForest包，构建随机森林模型，代码如下:

RF model < <- randomForest( target~. ,data = data, mtry= 2,ntree = 1000)

该模型涉及两个重要参数，参数的含义如下:

(1)树数目ntree。指定随机森林中包含的决策树数目，默认为500。

(2)分割变量数mtry。选择用于二叉分支的变量个数，如果数据集预测变量为n，对于分类问题，mtry为n的开方，对于回归问题，选择1/3。

基于gbm包，构建BRT模型,可执行如下代码:

BRT model <- gbm( target ~，，data = data, shrinkage = 0.01，distribution = "bernouli"，cv. folds = 5,n. trees = 3000, verbose = F)

该模型涉及四个基本参数,各个参数的含义如下:

(1)收缩率(shrinkage)。也叫学习速率，般选择在0.01~0.001之间。

(2)损失函数形式(distribution)。对于分类问题，采用Bernoulli分布;对于回归问题，采用Gassan分布。

(3)迭代次数(n.trees)。迭代回归树的数量，一般来说先越大越好，然后选择合适的数目n.tres参数可在500~1000之间。

(4)交叉验证的折数(cv.folds)。可以用来提取最适的回归树数目。

caret(C\_lassification\_A\_ nd\_ RE\_gression\_T\_raining)包是一系列函数的集合，具有强大的数据挖掘功能，包括数据预处理、可视化、模型训练、调参和变量重要性评估等，试图对数据挖掘过程进行流程化。

caret 包支持多种机器学习模型，可执行如下代码查询:

modelnames <- paste names(getModelInfo())，collapse = ",")

modelnames

Care包通过train()函数建模，提供了统一建模范式，第一个参数为"~ "，表达一个函数式formula，指明目标变量和特征变量，第二个参数为"data"，指定训练集，第三个参数"method"，指定建模方法。例如，训练集为train，测试集为test，目标变量为target，构建一个随机森林模型,并测试其性能，可执行如下代码:

rf\_ fit <- train target~., data = train, method = "ranger")

rf\_ fit

rf pred <- predict(rf fit, test) #性能测试

confusionMatrix(rf\_ pred, test $ target) #目标为分类变量

这是一种默认情况,train()函数用bootstrap resampling采样,重复25次,每个分割点随机选择3个特征(mtry=3),构建25棵树，最后得到25棵树的平均结果。

利用caret,同时构造不同模型类型,并选择最佳模型。网上有不少资料专门介绍，可以参阅学习。caret 包参数众多,且有的函数依赖于其他包,对于初学者有一定难度。在此,围绕生态学上常用树模型,说明caret包用法，加深对caret包的理解。

现安装caret包，该包可从GitHub安装,也可直接从Rstudio安装。在Rstudio安装和加载caret,可执行如下代码:

insall. packages( caret", dependencies = TRUE, INSTALL \_opts "- -no- lock")

library(caret)

**读取数据并预处理**

先将数据读进R,执行如下代码:

data <- read. csv( "npel. csv") #读取数据

head( data)

数据预处理是机器学习建模中的一个重要环节，主要针对预测变量，包括类别变量的独热编码、补齐缺失数据、删除近零方差数据和共线性数据、数据中心化和归--化等。可先用skim包中skim to\_ wide()函数做基本统计分析，再用caret包中dummyVars()创建独热编码，以及用preProcess( )函数补齐缺失值等，具体可参阅其他资料。

因树模型对缺失数据、变量类型、近零方差变量(每个数据都为1、均数为1)和共线性变量等不敏感，所以可跳过这个环节,读取数据后，直接进入数据分割环节。将目标变量loss\_rate划分为serious和normal两个类别,将问题转化为一个二分类问题,执行如下代码:

library( tidyverse)

data class <- data[,-1] %> % mutate(loss rate = case\_ when(los \_rate>= 0.4~ "serious”" ,loss rate<0.4 ~ "normal")) %>% rename(ls degree= loss rate)

data class <- data %>% muate(sries = loss rate> 0.4) %>% #将大于0.4分为serious select(-loss\_rate) #删除原来的loss \_rate

head(data class)

**数据分割与交叉验证**

建模时，常将数据集拆分为训练集(traning\_dataset,和测试集(esting daseDl分别用于训练模型和估计模型误差。caret 包提供了几个函数分割数据，如ceraml coeaerads ctemests和gopkFod等，常用函数为raeaPrtia .将数据集拆分为测试训练集，例如，将数据集拆分为训练(80%)和测试(20%)数据集，可执行如下代码:

set. seed(100)

train\_ idx <- createDataPartition(data class$lose\_degree, p=0.8, list= FALSE)

train <- data\_ class[ index,]

test <- data class[index,]

在建模时，利用验证数据集通过计算混淆矩阵(Confuse Matrix)来计算模型误差，根据模型性能调整相关参数。在将数据集分割训练和测试数据后.caret对训练集进行重采样，一部分用于训练，另员一部分作为验证集。重采样方法包括:

**K-折交叉验证(k-fold CV)**

当数据不充分时，如样本量N小于50，将样本数据分为k份，随机选择k-1作为训练集,剩余1份作为验证集,重复n次(n≤k),根据损失函数评估最优模型和参数。

**留一交叉验证(leave -one out CV)**

k-折交叉验证的特例,即k等于样本量,对于N个样本,每次选择k-1个样本训练,留下1个样本验证好坏。

**留出法(holdout cross validation)**

如样本量超过1万条，常用做法是将数据分为训练集(用于训练模型)、验证集(用于评价和调整模型参数)、测试集(用于选择模型和参数),这种验证称为留出法，也称为简单交叉验证。

caret提供了trainControl()函数指定交叉验证:方法,执行代码如下:

ctrl <- trainControl( method = "CV", number = 5) #5-折交叉检验

ctrl <- trainControl( method = "repeatcv", number = 5, repeats=3) #迭代3次的5-折交叉检验

ctrl <- trainControl( method = "LOOCV") #留一交叉验

ctrl <- trainControl(method = "LOOCV"，number = 1，p) #p为训练集,1~p为验证集

**变量重要性与选择**

选择些重要变量特征，对大多数机器学习建模很重要，特征选择的种方法是通过可视化方法，直接选出对目标变量影响较大的特征。可视化变量，可执行如下代码:

x = as.matrix(data\_class[, 1:11])

y = as.factor(data\_class $ loss\_degree)

featurePlot(x, y, plot = "box",

strip = strip.custom(par. strip. text= list(cex= 7)),

scales = list(x = list(relation = "free"),

y=list(relation = "free")))

也可执行如下代码，检查特征与目标变量的关系:

featurePlot(x, y, plot = "density",

strip = strip.custom(par. strip. text= list(cex= 7)),

scales = list(x = list(relation = "free"),

y=list(relation = "free")))

有时可能需要严格方法来确定重要特征，然后再将它们输人机器学习，选择特征的一个较好方法是递归特征消除( Recursive Feature Elimination, RFE)。递归消除特征法使用个基模型来进行多轮训练，首先用所有特征训练模型得到每个特征权重，并剔除拥有最小权重特征,之后再基于其余特征训练，重复前面的过程，如此往复递归，直至剩余的特征数量达到所需的特征数量。

caret使用rfe()函数实现RFE,并且可以通过定义refControl()函数来控制rfe使用什么算法以及何种交叉验证方法，执行如下代码:

options(warn = -1)

set. seed(100)

ctrl <- rfeControl( functions = rfFuncs,

method = "repeatedcv",

repeats = 5，

verbose = FALSE)

lmProfile <- rfe(x, y,rfeControl = ctrl)

ImProfile

另外，在特征比较多的情况下，可通过size指定模型大小，即拟需要的特征数量。

**构建模型并优化模型参数**

筛选特征后，利用arer包构建模型相对简单。例如，利用本例数据构建随机森林模型，并评估模型，可执行如下代码:

set. seed(100)

rf\_ fit <- train(as. factor(loss degree) ~IA+ PA+CA+Q+G，

data = training,

method = "rf")

rf\_fit

plot(rf\_ fit)

rf\_ pred <- predict(rf\_fit, test)

rf\_pred

confusionMatrix reference (reference = as.factor (test $ loss\_degree)，

data = rf\_pred,

mode = "everything")

caret包中的train()函数功能强大,配合trontrol,tuneGrid或tuneLength参数，可进一步优化模型。trainControl()函数不仅可以指定交叉验证方法，还可以指定模型输出，如对分类变量，可指定输出classProbs。

另外,不同机器学习模型有不同参数，包括决策树模型中的复杂性参数cp(0~1)、随机森林模型中的树数ntree(10~1000)和每次分割用的特征数mtry、提升回归树中的学习速率lr和树复杂度tc。优化模型参数有两种方式:

(1)通过tuneLength,指定系统自动调整每个参数可取值的数量。以随机森林算法为例,tunelength =5意味着尝试5个不同的mtry值，并根据这5个值找到最佳mtry值。优化上述随机森林模型，可执行如下代码:

trControl <- trainControl(

method = "cv"

number = 5，

savePredictions = "final",

classProbs = T,

summaryFunction = twoClassSummary)

rf\_ fit<- train(as. factor(loss degree)~.，

data = training,

method = "rf"，

tuneLength = 5,

trControl = trControl,

verbose = FALSE

)

rf\_ pred<- predict(rf\_fit, test)

rf\_pred

confusionMatrix(reference = as. factor(test $ loss\_degree),

data = rf\_pred,

mode = "everything")

library(MLeval)

x <- evalm(rf\_ fit)

x $ roc

x $ stdres

(2) TuneCrid为用户必须手动指定调参。在网格中,每个算法参数被指定为可能值的向量，这些向量组合定又所有可能。以提升回归树为例，优化参数,可执行如下代码：

myGrid <- expand. Grid(n.tree = c(150,175,200,225) #树的数目

interaction depth =c(5,6,7,8,9)，

shrinkage = c(0. 075,0.1,0.125,0.15,0.2)， #学习效率

n. minobsinnode=c(7, 10, 12, 15)) #终节点最少示例数

**比较并选择模型**

caret包中的train()函数，作为标准接口,可训练不同机器学习模型，同时caret包提供了评价模型的参数metrie,可根据模型类型，选择Acuracy、Kappa值，也可用灵敏度、特异性、ROC曲线下的面积(AUC)等。

(1)对于分类模型,一般用“Kappa”和“Accuracy"评估。

(2)对于概率模型，用“ROC"评估。

(3)对于随机森林模型，用“obb”评估。

(4)对于回归模型,用“RMSE"或“Rsquared”。

利用本例数据，运行多个模型，包括支持向量机、随机森林等,执行如下代码：

library( caretEnsemble)

trControl<- trainControl( method = "repeatedev",

number= 10,

repeats= 3,

savePredictions = TRUE,

classProbs = TRUE)

algorithmList <- c( "rf"，"rpart", "svmRadial")

set. seed(100)

models <一caretList(y ~ .，data = train,

trControl = trControl,

methodL ist = algorithmList)

results<- resamples( models)

summary( results)

scales <- list(x= list(relation= "free"), y= list(relation= "free"))

bwplot( results, scales = scales)

总之,caret包中的train()函数功能强大,创建模型、预测与保存结果的基本语法如下:

trControl<- trainControl(method = "repeatcv", number = 10, repeats = 3)

#重复3遍10-折交叉检验

model <- train(y = train[, 1],x tain[,2:ncol(train)]，

method = "rpart",

preProcess = c( "center", "scale"),

trControl = trControl,

tuneLength = 5，#用5个不同的mtry值，选择最优

metric = "ROC")

model. prediction <- predict( model, newdata) #预测为数据表

write. dbf() #将结果保存为数据库形式

除了caret包外，mlr包也提供了一个统一的机器学习框架，而且能够设置更多参数更加多。

**RSQL和PostgreSQL**

PostgreSQL 是一个功能强大、企业级的开源关系型数据库管理系统，以其稳定性、扩展性和标准合规性著称。它支持复杂查询、大数据量处理和高并发访问，具有丰富的特性，包括高级 SQL 功能、多种数据类型、存储过程和插件系统，适用于需要高数据完整性和复杂查询的大中型企业业务系统、数据仓库和分析平台。其强大的事务管理能力和安全性进一步增强了它在企业应用中的可靠性和信任度。

安装方式：https://www.postgresql.org/download/linux/ubuntu/

SQLite 是一个轻量级、嵌入式的开源关系型数据库管理系统，因其简单、快速和自包含而受到广泛欢迎。它无需独立服务器进程，整个数据库存储在一个单一文件中，便于移动和备份，适合嵌入在移动应用、桌面应用和嵌入式系统中。SQLite 通过零配置和快速启动，提供了便捷的数据存储解决方案，适用于小规模、资源受限的应用场景以及原型开发和测试环境。

安装方式：https://datawookie.dev/blog/2015/09/setting-up-odbc-for-sqlite-on-ubuntu/

**四、数据获取**

**rdataretriever**

rdataretriever 是一个 R 包，它提供了一个简单的方法来获取、管理和处理生态学和环境科学数据。它是 Data Retriever 项目的一部分，旨在帮助研究人员快速访问和处理各种格式和来源的大规模数据集，而无需手动下载和预处理数据。

rdataretriever 包实际上是 R 和 Python 的结合工具，依赖于 Data Retriever，这是一个用 Python 编写的数据获取工具。因此，在安装和使用 rdataretriever 包时，确保 Python 环境正确配置是非常重要的。以下是安装 rdataretriever 时需要注意的事项，尤其是与 Python 有关的部分。

安装方法：<https://github.com/ropensci/rdataretriever?tab=readme-ov-file>

**使用示例：**

library(rdataretriever) # 安装加载包

available\_datasets <- datasets()

print(available\_datasets) # 列出所有可用的数据集

install(dataset = "portal", format = "csv") # 安装 'portal' 数据集到 CSV 文件

portal\_data <- fetch(dataset = "portal")

head(portal\_data$species) # 加载 'portal' 数据集

**全球生物多样性信息机构（GBIF）**

GBIF 是一个国际性的开放数据平台，提供全球范围内的生物多样性数据。研究人员可以通过 GBIF 获取植物、动物、真菌等物种的分布记录。

访问方式：访问 GBIF 网站，使用其搜索和下载功能。

数据类型：物种分布、观察记录、标本数据等。

**使用示例：**

install.packages("rgbif")

library(rgbif)

occ\_data <- occ\_search(scientificName = "Panthera leo")

print(occ\_data) # 搜索物种

**国家生态观测网（NEON）**

NEON 是一个长期的大规模生态监测项目，收集美国范围内的生态和环境数据。数据包括气象、水文、植被、土壤和动物等方面的信息。

访问方式：访问 NEON 数据门户下载数据。

数据类型：气象数据、水文数据、植被数据、土壤数据、动物监测数据等。

**使用示例：**

install.packages("neonUtilities")

library(neonUtilities) # 安装加载包

plant\_data <- loadByProduct(dpID = "DP1.10098.001")

head(plant\_data) # 下载和加载 NEON 数据

**DataONE**

DataONE 是一个综合的数据网络，提供生态学、环境科学、地球科学等领域的研究数据。它整合了多个数据中心的数据。

访问方式：访问 DataONE 数据门户，使用其搜索和下载功能。

数据类型：生态学数据、环境数据、地球科学数据等。

**使用示例：**

install.packages("dataone")

library(dataone)

cn <- CNode("PROD")

mn <- getMNode(cn, "urn:node:KNB")

query <- "observation"

result <- query(mn, query)

print(result)

**PANGAEA**

PANGAEA 是一个地球和环境科学数据中心，提供各种生态学和环境科学数据。数据包括气候、海洋学、地质学等方面的信息。

访问方式：访问 PANGAEA 数据库下载数据。

数据类型：气候数据、海洋数据、地质数据等。

**使用示例：**

install.packages("pangaear")

library(pangaear)

res <- pg\_search(query = "climate change")

print(res) # 搜索和获取数据

**Biodiversity Heritage Library (BHL)**

BHL 提供大量关于生物多样性文献的开放访问，包含早期的科学出版物、图书、期刊等。

访问方式：访问 BHL 网站搜索和下载文献。

数据类型：历史文献、科学出版物等。

**使用示例：**

install.packages("rbhl")

library(rbhl)

res <- bhl\_booksearch(title = "birds")

print(res) # 搜索文献

**eBird**

eBird 是一个由康奈尔大学鸟类学实验室管理的全球鸟类观测数据库，提供鸟类观察记录。

访问方式：访问 eBird 数据库下载数据。

数据类型：鸟类观察记录、物种分布数据等。

**使用示例：**

install.packages("auk")

library(auk)

auk\_ebd("ebd\_relJun-2022.txt", "ebd\_sampling\_relJun-2022.txt") # 下载和处理 eBird 数据

**五、探索性数据分析**

生态学研究的一个基本目标是量化生物与环境之间的关系，所采用的数据来源不同，数据维数高、数据类型多，数据结构也比较复杂。如何选择变量、采用何种算法等，需要对生态数据结构进行探索性分析( Exploratory Data Analysis,EDA)。

多元分析特别适合生态学大数据探索分析，它是以关联矩阵(通常为相关系数或指数组成的对称方阵)为对象，通过矩阵的对角化(matrix diagonalization)和关联计算(associated computations)来识别大数据集的主要结构。

**数据表结构**

由于生态研究数据类型多、维度高，数据结构一般都比较复杂，但通常可以简化为更简单的形式。常见的数据结构是长形的数据表,行为采样地点或样方列为变量。除了单个数据表外,有时数据表还增加时间维度,形成K表。

以ade4包中的Doubs数据集为例，说明生态数据的结构。为查看该数据集，可执行如下代码:

library (ade4)

data (Doubs)

Doubs

运行后显示$env、Sfish、$xy、$species四个数据表。第一个是环境数据表，记录了30个样点与河流水文、地貌和化学有关的11个环境变量;第二个为物种数据表，记录了30个样点观察的鱼类数据,共27种鱼及丰度;第三个是样点的地理坐标(x和y)。这三个数据表就是采样地点物种和采样地点变量表。第四个为物种名称和编码表。这些数据表与R数据框结构是兼容的,适合R环境运行。

**变量或特征**

环境数据表中的变量在统计学分析通常将其称为因子或自变量,它们是统计模型的输入项，而对应物种数据表中的物种丰度数据被称为因变量，因为统计建模就是寻找变量间的因果关系。

数据挖掘关注的不是变量之间的因果关系，而是它们之间的相关性，不再区分“自变量”和“因变量”。例如,对于Doubs数据集,在水环境与鱼群关系研究中，环境数据表中的列变量das,alt、slo、flo、ph、har、pho等被称为特征或属性( feature/attribute)，对应列中的数值变量(target)。

数据挖掘涉及统计学方法、机器学习方法，为简单起见，将上述两类变量分别称为预测变量(predictors)/响应变量( responses),或输人(input>)/输出(outputs)变量。常见输入输出变量包括如下几类:

(1)分类变量。包括名义变量和序变量。

①名义变量。也称无序变量( nominal)。这种变量既无等级关系,也无数量关系，只有量之间没有数量关系，只有次序关系。如Doubs数据集中的鱼类丰度。

②有序变量(ordinal)。用于描述事物等级或顺序，变量值可以是数值型或字符型。变量之间没有数量关系，只有次序关系。如Doubs数据集中的鱼类丰度。

(2)定量变量。包括离散的或连续的变量。

①离散型。如长度、重量、产量等，,由测量或计数、统计所得到的量,这些变量具有数值特征。

②连续型。在一定区间内可以任意取值，其数值是连续不断的，相邻两个数值可作无限分割，即可取无限个数值。

**数据概要**

拿到数据后,接下来就是选择变量和算法,为此要进行探索性分析,确定数据结构，以便进一步分析。所谓探索性数据分析是指在缺少先验知识的情况下，利用散点图、制表和一些描述性统计量，以获得整个数据概况。数据探索主要内容包括:

(1)数据概况(summary)。包括数据概要、频度、异常值分布等。

(2)数据转化(transfer)。 涉及内容较多，主要包括消除量纲、分类编码等。

①消除量纲影响，如归一化、z-scores标准化。

②稳定方差，使其具有正态分布，如平方根、对数转化。

③分类数据编码。

**以Ade4包为例子**

①安装和加载"ade4"包

install.packages("ade4")

library(ade4)

②数据导入和准备

data(doubs) # 查看内置数据集

str(doubs) # 显示数据集结构

③数据清洗和检查

sum(is.na(doubs)) # 检查缺失值

summary(doubs$env) # 查看环境变量的数据统计信息

④数据可视化

主成分分析（PCA）

pca\_result <- dudi.pca(doubs$env, scannf = FALSE, nf = 3) #执行主成分分析

print(pca\_result) #查看主成分分析结果

s.label(pca\_result$li, main = "PCA of Environmental Variables") #绘制主成分分析的图形

对应分析（CA）

coa\_result <- dudi.coa(doubs$fish, scannf = FALSE, nf = 3) # 执行对应分析

print(coa\_result) # 查看对应分析结果

s.label(coa\_result$li, main = "CA of Fish Species") # 绘制对应分析的图形

⑤数据总结统计

# 计算环境变量的基本统计量

env\_stats <- doubs$env %>%

summarise(across(everything(), list(mean = mean, sd = sd, median = median)))

print(env\_stats)

⑥异常值检测

# 使用箱线图检测环境变量中的异常值

boxplot(doubs$env, main = "Boxplot for Environmental Variables", las = 2)

# 使用散点图检测鱼类物种数据中的异常值

plot(doubs$fish, main = "Scatter Plot for Fish Species Data")

⑦变量间关系分析

# 计算环境变量的相关矩阵

cor\_matrix <- cor(doubs$env, use = "complete.obs")

print(cor\_matrix)

# 绘制散点图矩阵

pairs(doubs$env, main = "Scatter Plot Matrix for Environmental Variables")

**多元分析**

多元分析（Multivariate Analysis）是数据生态学中常用的一组统计方法，用于分析包含多个变量的数据集。多元分析技术能够帮助研究人员理解复杂数据集中的变量关系、模式和结构，从而揭示生态系统中的潜在规律和生态过程。以下是一些常见的多元分析方法在数据生态分析中的应用：

**Q模式分析**

·定义与概念：

Q 模式（Q-mode）分析主要研究样本（个体或样点）之间的关系。

目标是找出样本之间的相似性或差异性，即哪些样本在给定的变量上表现相似。

样本（行）作为分析的对象，变量（列）作为描述这些对象的手段。

·适用场景：

分类和聚类分析：识别样本群体间的相似性，进行分类或聚类。

生态学研究：根据物种组成或环境变量来分类样点，理解样点之间的生态关系。

·常用方法：

主成分分析（PCA）：降维和提取主要成分，揭示样本之间的差异。

多维标度分析（MDS）：在低维空间中表示样本之间的距离。

聚类分析：将样本分类到不同的群组中，根据相似性进行分类。

示例：

假设我们有一个数据矩阵，其中行表示样本（如生态样点），列表示变量（如物种丰度）。

library(ade4) # 加载 ade4 包

data(doubs) # 加载示例数据集 doubs

pca\_result <- dudi.pca(doubs$env, scannf = FALSE, nf = 3) #执行主成分分析（PCA）

print(pca\_result) # 查看主成分分析结果

# 绘制主成分分析的图形

s.label(pca\_result$li, main = "PCA of Environmental Variables in Q-mode")

**R 模式分析**

·定义与概念

R 模式（R-mode）分析主要研究变量（特征或属性）之间的关系。

目标是找出变量之间的相似性或差异性，即哪些变量在不同样本中的表现相似。

变量（列）作为分析的对象，样本（行）作为这些变量的观察值。

·适用场景

变量相关性研究：研究变量之间的相关性或冗余性。

生态学研究：分析不同物种之间的相似性，理解它们的生态关系。

生物统计学：研究基因之间的相互作用，找出基因表达模式。

·常用方法

主成分分析（PCA）：降维和提取主要成分，揭示变量之间的差异。

因子分析（FA）：识别和解释隐藏在数据中的潜在因素。

冗余分析（RDA）：分析环境变量对物种丰度的解释程度。

示例：

假设我们有一个数据矩阵，其中行表示样本（如生态样点），列表示变量（如物种丰度）。

library(ade4) # 加载 ade4 包

data(doubs) # 加载示例数据集 doubs

# 执行主成分分析（PCA）在 R 模式下

pca\_result <- dudi.pca(doubs$fish, scannf = FALSE, nf = 3, scale = TRUE)

print(pca\_result) # 查看主成分分析结果

# 绘制主成分分析的图形

s.label(pca\_result$c1, main = "PCA of Fish Species in R-mode")

**比较：**

Q模式：主要分析样本之间的相似性。通过样本的多变量数据揭示样本间的关系。

R模式：主要分析变量之间的相似性。通过变量在多个样本中的表现揭示变量间的关系。

**Ade4包**

ade4 包是 R 语言中的一个功能强大的多元数据分析工具包，广泛用于生态学、环境科学、生物统计学和进化生物学等领域。ade4 包的设计初衷是为用户提供各种多元统计分析方法，特别是适用于生态学数据分析的方法。它提供了主成分分析（PCA）、对应分析（CA）、判别分析（DA）、多维标度分析（MDS）、因子分析等多种方法。

ade4 包内置了多个数据集，每个数据集适用于不同类型的多元分析和领域研究。以下是对主要数据集的总结：

**doubs**

描述：包含法国杜布斯河流域的鱼类物种数据和环境变量。

数据结构：

doubs$fish：鱼类物种丰度矩阵（行是样点，列是物种）。

doubs$env：环境变量数据（行是样点，列是环境变量）。

doubs$dfs：样点位置及其它信息。

用途：适用于生态学研究，尤其是鱼类物种分布和环境变量的关系分析。

**microsatt**

描述：包含鼠群体中微卫星标记的数据，用于群体遗传学分析。

数据结构：

microsatt$tab：微卫星标记数据矩阵（行是个体，列是位点）。

microsatt$pop：个体的群体分类信息。

用途：适用于遗传学研究，尤其是群体结构和遗传变异的分析。

**mite**

描述：包含苔藓丛中土壤螨虫的物种丰度数据和环境变量。

数据结构：

mite$tab：物种丰度矩阵（行是样点，列是物种）。

mite$env：环境变量数据。

用途：适用于生态学研究，尤其是土壤螨虫群落和环境变量的关系分析。

**aflw**

描述：包含不同国家橄榄油的化学特性数据，用于质量控制和分类分析。

数据结构：

aflw$tab：化学特性数据矩阵（行是样本，列是化学特性）。

aflw$fac：样本的分类信息（国家）。

用途：适用于食品科学研究，尤其是橄榄油质量控制和分类分析。

**olfacto**

描述：包含关于不同气味化学特性的感知数据。

数据结构：

olfacto$tab：气味化学特性数据矩阵（行是气味样本，列是化学特性）。

用途：适用于感官科学和化学分析研究，尤其是气味化学特性和感知之间的关系分析。

**以doubs数据集为例，展示如何加载和使用这些数据集：**

library(ade4) # 加载 ade4 包

data(doubs) # 加载 doubs 数据集

str(doubs) # 查看数据结构

pca\_result <- dudi.pca(doubs$env, scannf = FALSE, nf = 3)

s.label(pca\_result$li, main = "PCA of Environmental Variables") # 主成分分析

**六、空间数据获取与操作**

在数据生态学中，获取和处理空间数据是关键步骤。空间数据包含地理信息，可以展示物种分布和环境特征在空间上的变化。为了进行有效的生态学研究，我们需要利用各种工具和数据源来获取和操作这些空间数据。

Doubs 数据集是一个经典的生态数据集，包含丰富的环境和物种数据。图片中提到，环境变量数据包括 alt（海拔）、oxy（溶解氧）、dbo（生物需氧量）等。根据 RDA（冗余分析），这些变量能解释 73.4% 的变异，应被重视。为了进行有效的分析，需要获取高质量的地理数据。可以利用 QGIS 等工具从实际地理数据中提取所需的信息。这些工具可以帮助我们进行数据的地理配准和坐标转换。

**空间数据及坐标系**

空间数据是指包含地理位置信息的数据，广泛用于描述地球表面上的物体及其空间分布。空间数据在数据生态学中非常重要，因为它们可以帮助我们理解生态系统的动态、物种的分布以及环境的变化。空间数据主要分为两种类型：矢量数据和栅格数据。

**矢量数据 (Vector Data)**

矢量数据使用点、线和面的几何形状来表示地理特征：

·点：表示具体的位置，如树木或城市。

·线：表示连接点的路径，如河流或道路。

·面：表示封闭的区域，如湖泊或森林。

常见的矢量数据格式包括 Shapefile、GeoJSON 和 KML。矢量数据适用于表示有明确边界和离散的地理特征，如土地利用分类、道路网络和行政区划等。

**栅格数据 (Raster Data)**

栅格数据使用像素网格来表示连续的空间数据，每个像素包含一个数值，代表某种测量值（例如温度或高度）。常见的栅格数据格式包括 GeoTIFF、JPEG 和 PNG。栅格数据适用于表示连续变化的地理现象，如遥感影像、数字高程模型（DEM）和土地覆盖等。

**坐标系概述**

坐标系用于定义地理特征在空间中的位置。坐标系主要分为两类：地理坐标系和投影坐标系。

**地理坐标系 (Geographic Coordinate System)**

地理坐标系基于经度和纬度（longitude 和 latitude）来定义位置，单位为度（°）。常见的地理坐标系包括：

·WGS84（World Geodetic System 1984）：全球最广泛使用的地理坐标系，也是 GPS 使用的参考坐标系。

·NAD83（North American Datum 1983）：北美常用的地理坐标系。

**投影坐标系 (Projected Coordinate System)**

投影坐标系通过将地球的三维表面投影到二维平面上来定义位置，单位为米（m）或英尺（ft）。常见的投影坐标系包括：

UTM（Universal Transverse Mercator）：将地球表面划分为多个投影带，每个带覆盖 6 度经度范围。

国家平面坐标系（State Plane Coordinate System, SPCS）：美国使用的一种平面坐标系，适用于小范围区域。

**坐标系转换**

为了进行有效的空间分析和数据整合，常常需要将不同坐标系的数据转换到同一坐标系中。以下是一些常见的坐标系转换操作示例。

**使用 R 进行坐标系转换**

在 R 中，可以使用 sf 和 raster 包进行坐标系转换。示例如下：

library(sf)

library(raster)

# 导入矢量数据

vector\_data <- st\_read("path/to/vector\_data.shp")

# 转换坐标系到 WGS84

vector\_data\_transformed <- st\_transform(vector\_data, crs = 4326)

# 导入栅格数据

raster\_data <- raster("path/to/raster\_data.tif")

# 转换坐标系到 UTM Zone 33N

raster\_data\_transformed <- projectRaster(raster\_data, crs = CRS("+proj=utm +zone=33 +datum=WGS84"))

**QGIS**

QGIS（Quantum GIS）是一款开源的地理信息系统（GIS）软件，它提供了一整套工具用于编辑、分析和可视化地理空间数据。QGIS 支持多种数据格式和坐标参考系统，是处理和分析地理信息数据的强大工具。它的开源性质使其不仅适用于学术研究和教学，也适用于政府和商业项目。

QGIS 支持多种矢量和栅格数据格式的导入和导出，包括 Shapefile、GeoJSON、KML、GPX、GeoTIFF、JPEG 等。用户可以轻松地将各种地理空间数据导入到 QGIS 中进行处理和分析。导入数据的步骤简单明了，只需通过菜单栏选择相应的选项即可实现。

QGIS 提供了丰富的数据编辑工具，允许用户对矢量数据进行创建、编辑和修改操作。用户可以通过点、线和多边形等几何图形来表示和编辑地理特征。这些工具使得用户可以在地图上进行精细的地理信息编辑，满足各种专业需求

QGIS 提供了强大的可视化功能，用户可以通过符号化、标注和样式设置来自定义地图的显示效果。QGIS 支持多种投影方式和坐标参考系统，可以生成高质量的地图和图表。用户可以利用这些工具来创建直观且信息丰富的地理数据可视化，便于分析和决策。

QGIS 包含丰富的空间分析工具，包括缓冲区分析、叠加分析、地形分析、网络分析等。用户可以使用这些工具对地理空间数据进行各种分析操作，从而揭示数据中的空间关系和模式。这些分析工具能够帮助用户深入理解地理数据的内在结构和分布规律。

QGIS 拥有强大的插件系统，用户可以通过安装和使用各种插件来扩展 QGIS 的功能。插件库中有许多用于特定任务和应用的插件，如地理编码、地理配准、遥感数据处理等。用户可以根据自己的需要选择和安装插件，以增强 QGIS 的功能和适用性

QGIS 允许用户进行地理配准操作，将扫描地图或图像与实际地理坐标对齐，使其与其他地理空间数据兼容。地理配准工具可以将图像中的控制点与地理参考点匹配，从而生成地理坐标信息。这对于将历史地图或遥感图像整合到现有地理数据中尤为重要。

QGIS 支持地理编码功能，可以将地址或地名转换为地理坐标。通过安装地理编码插件，用户可以输入地址信息并获取其对应的地理坐标，从而在地图上进行定位和分析。地理编码是城市规划和物流等领域中的关键功能。

**使用 QGIS 进行坐标系转换**

在 QGIS 中，转换坐标系的步骤如下：

①打开 QGIS 并导入矢量或栅格数据。

②右键点击图层，选择 "Export" > "Save Features As..."。

③在 "CRS" 部分选择目标坐标系（例如 WGS84）。

④点击 "OK" 保存转换后的数据。

**使用QGIS获取点坐标系的步骤：**

①导入矢量数据（如点数据）或栅格数据：

矢量数据文件格式通常为 Shapefile (.shp)、GeoJSON (.geojson) 等。

栅格数据文件格式通常为 GeoTIFF (.tif)、JPEG (.jpg) 等。

②查看当前数据的坐标系：

在 QGIS 中，右键点击数据图层，选择 "Properties"。

在 “Information” 标签页中，可以查看当前图层的坐标参考系统（CRS）。

③设置或转换数据的坐标系：

矢量数据：

右键点击图层，选择 "Export" > "Save Features As..."。

在弹出的对话框中，选择目标坐标系（例如 WGS84 EPSG:4326）。

点击 "OK" 保存转换后的数据。

栅格数据：

右键点击图层，选择 "Export" > "Save As..."。

在弹出的对话框中，选择目标坐标系。

点击 "OK" 保存转换后的数据。

④使用 QGIS 的 geocoding 插件进行地理编码：

在 QGIS 中，安装和使用 geocoding 插件，输入地址信息，获取对应的地理坐标。

将获取的地理坐标数据导入到项目中，设置合适的坐标系。

除了 QGIS，还有其他工具和方法可以用来获取和处理空间坐标系。例如，在 R 语言中，可以使用 sf 和 raster 包进行坐标系的设置和转换：

library(sf)

library(raster)

# 导入矢量数据

vector\_data <- st\_read("path/to/vector\_data.shp")

# 查看当前坐标系

st\_crs(vector\_data)

# 转换坐标系到 WGS84

vector\_data\_transformed <- st\_transform(vector\_data, crs = 4326)

# 导入栅格数据

raster\_data <- raster("path/to/raster\_data.tif")

# 查看当前坐标系

crs(raster\_data)

# 转换坐标系到 UTM Zone 33N

raster\_data\_transformed <- projectRaster(raster\_data, crs = CRS("+proj=utm +zone=33 +datum=WGS84"))

**获取遥感数据**

利用 QGIS 工具获取遥感数据：

一些开源数据来源，如 USGS 或 Sentinel-2，提供遥感图像数据。可以使用 QGIS 获取这些数据，进行处理和分析。在 R 语言中，可以使用 getSpatialData 包获取遥感数据：

library(getSpatialData)

# 设置账户信息（若需要）

set\_credentials(user = "your\_username", password = "your\_password")

# 获取指定区域和时间范围的 Sentinel-2 数据

aoi <- st\_as\_sf(data.frame(lon = 0, lat = 0), coords = c("lon", "lat"), crs = 4326) # 设置感兴趣区域（AOI）

time\_range <- c("2022-01-01", "2022-12-31") # 设置时间范围

products <- getSpatialData::getSentinel\_query(aoi, time\_range)

# 下载数据

download\_data(products)

**地理坐标的配准**

地理坐标的配准是将地图或图像上的特定点与地理坐标系统中的实际地理位置相对应的过程。这种配准过程在地理信息系统（GIS）和遥感图像处理中经常用到。在配准过程中，有两种常用的方法：Freehand和Georeferencer。

**Freehand（手绘）配准：**

在Freehand配准中，用户通过手动标记图像上的几个关键点，然后将这些点与其在地图上的对应点进行匹配。这种方法适用于简单的图像，其中特征点明显且易于识别。

Freehand配准的优点是简单易用，不需要复杂的算法或工具。它适用于需要快速完成配准的情况，例如在小范围内进行简单地图叠加或比较。

**Georeferencer（地理参考器）配准：**

Georeferencer是一种专门的工具，用于将非地理空间的图像或地图参照地理坐标系统进行配准。它通过一系列算法自动匹配图像上的特征点与地理数据中的已知地理坐标点。

通常，Georeferencer配准包括以下步骤：

·选择地图或图像。

·选择地理参照图层（通常是现有的地图或地理空间数据）。

·标记图像上的特征点，并将它们与地理参照图层上的对应点进行匹配。

·系统使用这些匹配点之间的变换关系来调整图像的位置和方向，以实现地理坐标的匹配。

Georeferencer配准的优点是准确性高，特别适用于需要高精度地理坐标的应用，例如地图制图、环境监测和资源管理等。

**七、时间序列的创建与重表达**

**生态学中的时间序列数据：**

在生态学中，时间序列数据是对生态系统中各种生物和环境变量随时间变化的观测数据。这些数据通常涉及到生物群落、种群数量、环境条件等方面的变化。生态学中的时间序列数据具有以下特点和应用：

·种群数量变化：生态学家经常关注某一物种或多个物种的数量随时间的变化。这种数据可能涉及种群的增长、减少、周期性波动等现象，用于研究种群动态和生态系统稳定性。

·物种多样性：除了数量变化，时间序列数据也可以描述生态系统中的物种多样性随时间的变化。这包括物种组成的变化、生物多样性指数的计算等，用于评估生态系统的健康状况和生物多样性的保护情况。

·环境因子：时间序列数据还可以记录生态系统中的环境因子随时间的变化，如温度、湿度、光照强度、土壤水分等。这些数据对于理解生态系统的生物和环境相互作用、生态位分配、物种适应性等具有重要意义。

·生态事件：时间序列数据也可以用于记录生态系统中的重要事件或现象，如自然灾害、物种迁徙、生境变化等。这些数据对于生态系统的恢复力和稳健性研究具有重要意义。

**创建和重表达时间序列在数据生态学中是常见的任务，主要包括以下几个步骤：**

①数据收集：首先，需要收集与研究对象相关的原始数据。这些数据可以来自各种来源，包括传感器、观测站、遥感数据、文献报道等。确保数据的质量和可靠性对后续分析至关重要。

②时间序列构建：一旦收集到数据，就可以将其组织成时间序列。时间序列通常由时间戳（日期或时间）和相应的数值组成。在构建时间序列时，需要注意选择合适的时间间隔（例如日、月、年），以及如何处理缺失值和异常值。

③数据清洗和预处理：在创建时间序列之后，需要对数据进行清洗和预处理。这包括去除异常值、填补缺失值、平滑数据等操作，以确保数据的质量和可用性。

④时间序列重表达：时间序列重表达是将原始时间序列转换为新的表示形式的过程，其目的是为了更好地揭示数据的特征或提高数据的可解释性。常见的重表达方法包括：

⑤聚合和分组：将时间序列数据按照不同的时间间隔进行聚合，例如将日度数据聚合为月度或年度数据，以减少数据的复杂度。

⑥平滑和插值：使用平滑方法（如移动平均）或插值方法（如线性插值）来处理数据中的噪声或缺失值，以获得更平稳或完整的时间序列。

⑦周期性分析：对时间序列进行周期性分析，例如分析季节性变化或周期性趋势，以揭示数据的周期性特征。

⑧转换和归一化：对时间序列进行数学变换或归一化操作，以改变数据的尺度或分布特征，例如对数变换或标准化。

通过创建和重表达时间序列，可以更好地理解数据的演变规律和趋势，为数据生态系统的分析、建模和预测提供更可靠的基础。

**时间戳和R中的三个基本时间类：**

在R语言中，时间戳是指表示日期和时间的数据类型。R语言提供了多种用于处理时间序列数据的内置数据类型和函数。其中，三个基本的时间类包括：

**Date（日期）：**

Date类用于表示日期，即年、月、日，但不包含具体的时间信息。

在R中，日期可以用YYYY-MM-DD的格式表示，例如"2024-05-31"。

Date类的对象可以使用as.Date()函数来创建，也可以从其他对象转换而来。

**POSIXct（带时区的日期时间）：**

POSIXct类表示带有时区信息的日期和时间。

它以秒数形式存储自协调世界时（UTC）1970年1月1日午夜开始的时间戳，并附带有时区信息。

在R中，POSIXct对象可以使用as.POSIXct()函数创建，也可以从其他对象转换而来。

**POSIXlt（列表形式的日期时间）：**

POSIXlt类表示日期和时间的列表形式，包含年、月、日、时、分、秒等信息。

与POSIXct不同，POSIXlt对象以列表形式存储日期时间信息，可以更方便地访问年、月、日等组成部分。

在R中，POSIXlt对象可以使用as.POSIXlt()函数创建，也可以从其他对象转换而来。

这三种基本的时间类在R中提供了灵活且功能丰富的时间处理功能，可用于处理时间序列数据、日期计算、时间戳转换等任务。根据具体的需求和数据类型，选择合适的时间类进行操作和分析。

**创建时间序列：**

在R语言中，可以使用内置的时间序列函数来创建时间序列数据。以下是一些创建时间序列的方法：

①使用ts()函数创建时间序列：

# 创建一个简单的时间序列，从2010年1月1日开始，每月一个观测值

ts\_data <- ts(data = rnorm(60), start = c(2010, 1), frequency = 12)

# 查看时间序列的结构

print(ts\_data)

②使用xts()函数创建时间序列：

library(xts)

# 创建一个包含日期时间索引的时间序列

dates <- seq(as.POSIXct("2024-01-01"), by = "day", length.out = 365)

values <- rnorm(365)

xts\_data <- xts(x = values, order.by = dates)

# 查看时间序列的结构

print(xts\_data)

③使用zoo()函数创建时间序列：

library(zoo)

# 创建一个包含日期索引的时间序列

dates <- seq(as.Date("2024-01-01"), by = "month", length.out = 12)

values <- rnorm(12)

zoo\_data <- zoo(x = values, order.by = dates)

# 查看时间序列的结构

print(zoo\_data)

④直接创建时间序列对象：

# 创建一个时间序列对象

dates <- seq(as.Date("2024-01-01"), by = "month", length.out = 12)

values <- rnorm(12)

time\_series <- ts(values, start = c(2024, 1), frequency = 12)

# 查看时间序列的结构

print(time\_series)

**时序重表达**

时间序列的每一次观测都是数据的一个维度，因此时间序列数据是超高维的，降低维度是时间序列数据预处理过程的重要任务。由于时序数据可能存在大量的噪声，般不直接分析原始数据，用更简洁的形式表达时间序列数据，达到减少维数(减少数据点)、降低计算复杂度的目的。

**分段表达时间序列是最简单的降维方法**，即通过重采样.将时序P的长度m减少到n。该方法大致分为:

**①分段均值（Piecewise Aggregate Approximation, PAA）**

分段均值方法将时间序列分成固定长度的段，然后计算每段的均值。

set.seed(123)

time\_series <- ts(rnorm(100), frequency = 12, start = c(2020, 1)) # 创建一个示例时间序列

paa <- function(series, segments) {

n <- length(series)

segment\_length <- floor(n / segments)

result <- numeric(segments)

for (i in 1:segments) {

start\_index <- (i - 1) \* segment\_length + 1

end\_index <- min(i \* segment\_length, n)

result[i] <- mean(series[start\_index:end\_index])

}

return(result)

} # 分段函数

paa\_result <- paa(time\_series, 10) # 将时间序列分成10段=

print(paa\_result) # 打印结果

**②分段最大值/最小值（Piecewise Maximum/Minimum Representation）**

类似于分段均值，可以使用每段的最大值或最小值来表示时间序列。

piecewise\_max <- function(series, segments) {

n <- length(series)

segment\_length <- floor(n / segments)

result <- numeric(segments)

for (i in 1:segments) {

start\_index <- (i - 1) \* segment\_length + 1

end\_index <- min(i \* segment\_length, n)

result[i] <- max(series[start\_index:end\_index])

}

return(result)

} # 分段最大值函数

max\_result <- piecewise\_max(time\_series, 10) # 将时间序列分成10段，取每段的最大值

print(max\_result) # 打印结果

**③分段和（Piecewise Sum Representation）**

使用每段的和来表示时间序列。

piecewise\_sum <- function(series, segments) {

n <- length(series)

segment\_length <- floor(n / segments)

result <- numeric(segments)

for (i in 1:segments) {

start\_index <- (i - 1) \* segment\_length + 1

end\_index <- min(i \* segment\_length, n)

result[i] <- sum(series[start\_index:end\_index])

}

return(result)

} # 分段和函数

sum\_result <- piecewise\_sum(time\_series, 10) # 将时间序列分成10段，取每段的和

print(sum\_result) # 打印结果

**④分段中位数（Piecewise Median Representation）**

使用每段的中位数来表示时间序列。

piecewise\_median <- function(series, segments) {

n <- length(series)

segment\_length <- floor(n / segments)

result <- numeric(segments)

for (i in 1:segments) {

start\_index <- (i - 1) \* segment\_length + 1

end\_index <- min(i \* segment\_length, n)

result[i] <- median(series[start\_index:end\_index])

}

return(result)

} # 分段中位数函数

median\_result <- piecewise\_median(time\_series, 10) #将时间序列分成10段，取每段的中位数

print(median\_result) # 打印结果

**八、生态网络与图挖掘**

**网络数据表达**

网络数据表达是指将数据以图或网络的形式表示，使得数据中的实体（节点）及其相互关系（边）可以被直观地展示和分析。在生态学中，网络数据表达主要用于描述和分析生态系统中的生物相互关系，如食物网、相互作用网等。

**基本概念**

**①节点（Nodes）：**

表示网络中的实体。例如，在生态网络中，节点可以表示物种或个体。

**②边（Edges）：**

表示节点之间的关系或交互。例如，捕食关系、竞争关系等。边可以是有向的（directional）或无向的（undirectional），也可以有权重（weighted）或无权重（unweighted）。

**③邻接矩阵（Adjacency Matrix）：**

一个方阵，用于表示节点之间的连接情况。矩阵的行和列对应节点，矩阵元素值表示相应节点对之间是否存在边及边的权重。

**④邻接表（Adjacency List）：**

以列表形式表示网络，每个节点的邻居节点及其连接关系存储在列表中。

**⑤图（Graph）：**

网络的数学表示形式，包括节点集和边集。

**R语言中的网络数据表达**

R语言有许多用于创建和分析网络数据的包，其中最常用的是“igraph”和“network”。下面是一些示例代码，展示如何使用这些包来表达和分析网络数据。

**使用“igraph”包**

# 安装并加载igraph包

install.packages("igraph")

library(igraph)

# 创建一个简单的有向图

nodes <- data.frame(name = c("Plant", "Herbivore", "Carnivore"))

edges <- data.frame(from = c("Plant", "Herbivore"), to = c("Herbivore", "Carnivore"))

# 创建网络对象

g <- graph\_from\_data\_frame(d = edges, vertices = nodes, directed = TRUE)

# 绘制网络

plot(g, vertex.label = V(g)$name)

# 查看邻接矩阵

adj\_matrix <- as\_adjacency\_matrix(g)

print(adj\_matrix)

**使用“network”包**

# 安装并加载network包

install.packages("network")

library(network)

# 创建一个简单的网络

net <- network.initialize(3, directed = TRUE)

network.vertex.names(net) <- c("Plant", "Herbivore", "Carnivore")

add.edges(net, tail = c(1, 2), head = c(2, 3))

# 绘制网络

plot(net, displaylabels = TRUE)

# 查看邻接矩阵

adj\_matrix <- as.matrix.network.adjacency(net)

print(adj\_matrix)

**网络数据表达方式**

**邻接矩阵（Adjacency Matrix）：**

一个方阵，用于表示节点之间的连接情况。

矩阵的行和列对应节点，矩阵元素值表示相应节点对之间是否存在边及边的权重。

library(igraph)

# 创建邻接矩阵

adj\_matrix <- matrix(c(0, 1, 0, 1, 0, 1, 0, 0, 0), nrow = 3, byrow = TRUE)

colnames(adj\_matrix) <- rownames(adj\_matrix) <- c("Plant", "Herbivore", "Carnivore")

**邻接表（Adjacency List）：**

以列表形式表示网络，每个节点的邻居节点及其连接关系存储在列表中。

# 创建邻接表

adj\_list <- list(

Plant = c("Herbivore"),

Herbivore = c("Carnivore"),

Carnivore = character(0)

)

**边列表（Edge List）：**

使用边的列表来表示网络。每条边用一对节点表示。

edges <- data.frame(from = c("Plant", "Herbivore"), to = c("Herbivore", "Carnivore"))

**生态网络分析方法**

**节点中心性分析（Node Centrality Analysis）：**

评估节点在网络中的重要性。常用指标包括度中心性、接近中心性、中介中心性等。

# 计算度中心性

degree\_centrality <- degree(g, mode = "all")

print(degree\_centrality)

**社区检测（Community Detection）：**

识别网络中的紧密连接子集（社区），揭示物种的功能群和生态模块。

# 社区检测

community <- cluster\_louvain(g)

plot(g, vertex.label = V(g)$name, mark.groups = communities(community))

**网络拓扑分析（Network Topology Analysis）：**

分析网络的整体结构特征，如网络密度、平均路径长度、聚类系数等。

# 计算网络密度

network\_density <- edge\_density(g)

print(network\_density)

# 计算平均路径长度

avg\_path\_length <- average.path.length(g, directed = TRUE)

print(avg\_path\_length)

# 计算聚类系数

clustering\_coeff <- transitivity(g, type = "global")

print(clustering\_coeff)

**生态网络探索性分析**

生态网络探索性分析是研究生态系统中生物及其相互关系的结构和功能的一系列方法。通过这些分析，科学家可以揭示生态网络的复杂性、发现重要的物种和互动关系，以及预测生态系统在不同环境压力下的响应。

生态网络探索性分析中的节点（node）和网络（network）特征分析是理解生态系统结构和功能的关键。以下介绍在这两个层面上的特征分析方法和常用的R代码示例。

**节点（Node）水平上的特征分析**

节点特征分析旨在评估网络中每个节点（物种或个体）的重要性及其在网络中的角色。常用的节点特征指标包括：

**①度中心性（Degree Centrality）：**

节点的度数是指与该节点直接相连的边数。

在生态网络中，度中心性高的物种通常具有较多的互动关系，可能在网络中扮演关键角色。

degree\_centrality <- degree(g, mode = "all")

print(degree\_centrality)

**②接近中心性（Closeness Centrality）：**

衡量一个节点到网络中所有其他节点的平均最短路径距离。

接近中心性高的节点能够更快地影响整个网络。

closeness\_centrality <- closeness(g, mode = "all")

print(closeness\_centrality)

**③中介中心性（Betweenness Centrality）：**

衡量一个节点在所有最短路径中出现的频率。

中介中心性高的节点通常是网络中重要的“桥梁”，在信息或能量流动中起关键作用。

betweenness\_centrality <- betweenness(g, directed = TRUE)

print(betweenness\_centrality)

**④特征向量中心性（Eigenvector Centrality）：**

衡量一个节点的影响力，不仅考虑其直接连接的节点，还考虑这些邻居节点的中心性。

特征向量中心性高的节点通常与其他高中心性的节点相连，具有较大的影响力。

eigenvector\_centrality <- evcent(g)$vector

print(eigenvector\_centrality)

**网络（Network）水平上的特征分析**

网络特征分析旨在评估整个网络的结构特性和功能。常用的网络特征指标包括：

**①网络密度（Network Density）：**

衡量网络中实际存在的边与可能存在的最大边数的比值。

网络密度高表示网络中节点之间的连接较为紧密。

network\_density <- edge\_density(g)

print(network\_density)

**②平均路径长度（Average Path Length）：**

网络中所有节点对之间最短路径长度的平均值。

平均路径长度短的网络通常具有较高的连通性和传播效率。

avg\_path\_length <- average.path.length(g, directed = TRUE)

print(avg\_path\_length)

**③聚类系数（Clustering Coefficient）：**

衡量节点的邻居之间实际连接的边数与可能的最大连接数之比。

高聚类系数表示网络中存在较多的紧密连接的节点群体（社区）。

clustering\_coeff <- transitivity(g, type = "global")

print(clustering\_coeff)

**④网络直径（Network Diameter）：**

网络中所有节点对之间最短路径的最大值。

网络直径反映了网络的扩展程度。

network\_diameter <- diameter(g, directed = TRUE)

print(network\_diameter)

**⑤模块度（Modularity）：**

衡量网络中社区结构的质量，模块度高表示网络中存在明显的社区结构。

通过社区检测算法（如Louvain方法）可以计算网络的模块度。

**R中的生态网络分析**

以下是一个综合示例，展示如何在R中进行节点和网络特征分析：

library(igraph)

# 创建一个简单的有向图

nodes <- data.frame(name = c("Plant", "Herbivore", "Carnivore"))

edges <- data.frame(from = c("Plant", "Herbivore"), to = c("Herbivore", "Carnivore"))

g <- graph\_from\_data\_frame(d = edges, vertices = nodes, directed = TRUE)

# 节点特征分析

degree\_centrality <- degree(g, mode = "all")

closeness\_centrality <- closeness(g, mode = "all")

betweenness\_centrality <- betweenness(g, directed = TRUE)

eigenvector\_centrality <- evcent(g)$vector

cat("Degree Centrality:", degree\_centrality, "\n")

cat("Closeness Centrality:", closeness\_centrality, "\n")

cat("Betweenness Centrality:", betweenness\_centrality, "\n")

cat("Eigenvector Centrality:", eigenvector\_centrality, "\n")

# 网络特征分析

network\_density <- edge\_density(g)

avg\_path\_length <- average.path.length(g, directed = TRUE)

clustering\_coeff <- transitivity(g, type = "global")

network\_diameter <- diameter(g, directed = TRUE)

community <- cluster\_louvain(g)

modularity\_value <- modularity(community)

cat("Network Density:", network\_density, "\n")

cat("Average Path Length:", avg\_path\_length, "\n")

cat("Clustering Coefficient:", clustering\_coeff, "\n")

cat("Network Diameter:", network\_diameter, "\n")

cat("Modularity:", modularity\_value, "\n")