数据分析与处理技术

流程设计

流程控制

从函数开始，代码往往不再是一行执行一次，而是许多行命令来完成一个计算任务。
如：{ }括起的一个代码段；R的脚本文件里的所有代码
在一个代码段或脚本文件里，代码从上往下依次执行，在一行代码里根据运算符优先级高的先执行
简单顺序的代码流程还不足以完成全部工作，在此引入两个基本的流程控制结构：分支选择结构和循环控制结构

分支选择结构 if/else

ifelse选择计算函数

```
> a=rnorm(10,mean=5,sd=3)
> a
 [1]  6.953861 12.299455  8.574015  7.767328  6.863292  6.069000  5.213439  7.195408
 [9]  6.859369  4.272321
> b=ifelse(a>6,round(a)*2,floor(a)/2)
> b
 [1] 14.0 24.0 18.0 16.0 14.0 12.0  2.5 14.0 14.0  2.0
```

if/else 选择分支结构

```
if(条件) {
   statement1
}else{
   statement2
}
```

```
if(t>50){
  cat('t is bigger than 50')
}else
{
  cat('t is smaller than 50')
}
```

分支嵌套

当分支条件不止一个时，if/else结构可以轻松嵌套起来，构成多分支选择

例如：

```
if(i>=2000){
  print(">=2000")
}else{
  if(i>1000&& i<2000){
    print("1000<i<2000")
  }else{
    print("smaller than 1000")
  }
}
```

条件组合

if(条件) 中的条件是逻辑取值，TRUE时执行，FALSE跳转至else的代码段

```
if(TRUE){
  cat('t is bigger than 50')
}else
{
  cat('t is smaller than 50')
}
```

例如：if后的条件直接设置为TRUE，
相当于指定选择if的代码段

```
if(t>50 && t<100){
  cat('yes')
}else
{
  cat('this else')
}
```

回顾课件2逻辑运算，any all以及&& ||
的用法

循环结构

正如rep重复生成函数

```
> rep(1:3,2)
[1] 1 2 3 1 2 3
```

循环结构控制一个代码段重复执行指定次数

循环结构常用的结构有repeat结构，for结构（指定次数），while结构（按条件循环），

$$\sum_{i=1}^{100} i$$

循环次数需要一个专门的变量来控制，三种循环结构的主要差别在于控制变量形式

```
answer=0
i=1
repeat{
  answer=answer+i
  i=i+1
  if(i>100) break
}
```

```
answer=0
for(i in 1:100){
  answer=answer+i
}
```

```
answer=0
i=1
while(i<=100){
  answer=answer+i
  i=i+1
}
```

循环的使用

使用第一次作业的数据yunliang矩阵，截取第一行数据为例，计算一个简单的加总

```
> m=yunliang[1,]

answer=0
for(i in m){
  answer=answer+i
}
```

tips:仍然需要提前设置一个变量answer来装计算结果

```
> dongbei=c('沈阳','长春','哈尔滨')

answer=0
for(i in dongbei){
  answer=answer+m[i]
}
```

```
> which(names(m) %in% dongbei)
[1] 10 11 12

answer=0
for(i in c(10,11,12)){
  answer=answer+m[i]
}
```

循环嵌套

举例：双层累加

$$\sum_{j=1}^{10}\sum_{i=1}^{10} i\sin(j\cdot\pi)$$

```
answer=0
for(i in 1:10){
  for(j in 1:10){
    answer=answer+i*sin(j*pi)
  }
}
```

计算东北三个物流中心的物流流量

```
answer=rep(0,ncol(yunliang))
for(i in 1:ncol(yunliang)){
  for(j in dongbei){
    answer[i]=answer[i]+yunliang[i,j]
  }
}
```

进一步，计算 $\sum_{i=1}^{10}\sum_{j=1}^{i} i\sin(j\cdot\pi)$

循环与分支的嵌套

计算东北三个物流中心的物流流量，
但是内部三城市的数据不计入

```
answer=rep(0,ncol(yunliang))
for(i in 1:ncol(yunliang)){
  for(j in dongbei){
    if(j==colnames(yunliang)[i]){
      answer[i]=0
      break
    }
    answer[i]=answer[i]+yunliang[i,j]

  }
}
```

尝试用循环结构设计程序完成作业1中的运量表合并问题

函数封装计算流程

将上述计算过程装入函数中

```
myfunc<-function(diqu){
answer=rep(0,ncol(yunliang))
for(i in 1:ncol(yunliang)){
  for(j in diqu){
    if(j==colnames(yunliang)[i]){
      answer[i]=0
      break
    }
    answer[i]=answer[i]+yunliang[i,j]

  }
}
return(answer)
}
```

那么我们不必在计算其他中心时做重复事情，而是在循环中调用函数

```
for(i in 1:length(center)){
  if(i==1){
    hebing=matrix(myfunc(center[[i]]),nrow=1)
  }else{
    hebing=rbind(hebing,myfunc(center[[i]]))
  }
}
```

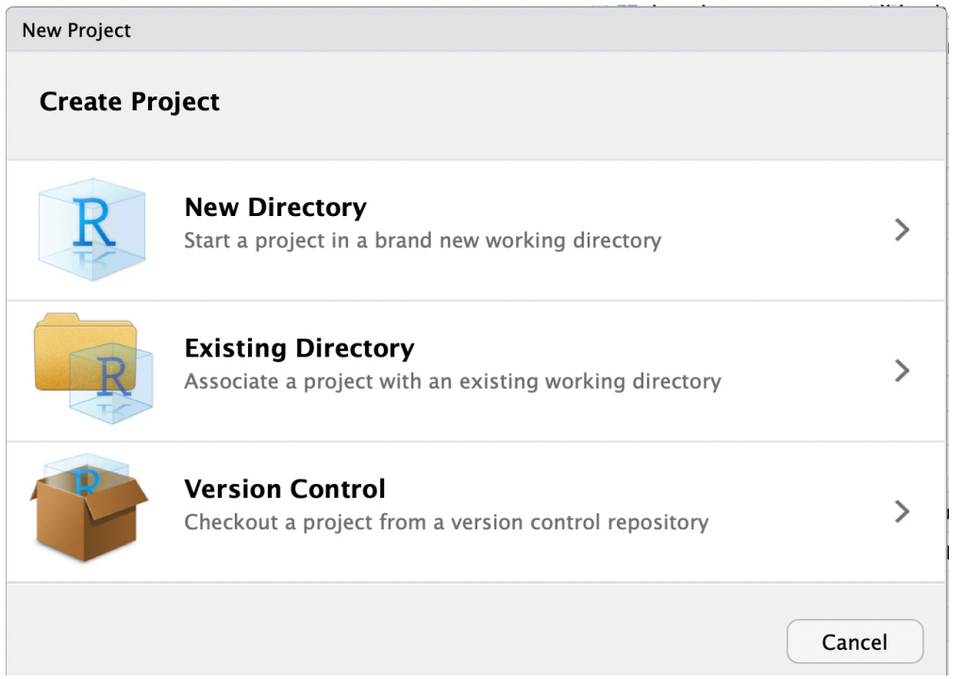
项目环境

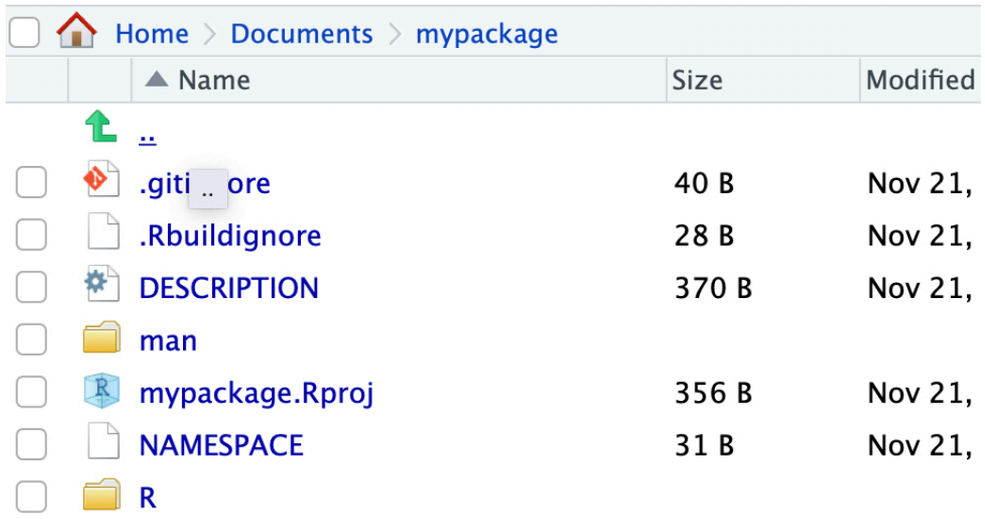
将一个工作主题作为一个隔离的工作环境是不错的主意，尤其是做成package形式可以相互交流使用。Rstudio中提供了选菜单方式建立project和package环境。

建立R包需要从CRAN上下载安装Rtools作为基础。

https://cran.r-project.org/bin/windows/Rtools/

建立package时,Rstudio自动生成右侧下方文件，自己编写的函数放入R文件夹下。





—在Rstudio中建立一个新的project，为其命名（如：project1）选择制作package；
—将你要做进包里的函数和数据调入这个project，然后进入project的环境进行调试；
—选择Build & Reload，进入调试状态，此时可以对调入的函数代码进行测试和修改；
—Check
—Build Banary Package
到制定目录（默认在工作目录\practice\）找做好的package，即一个叫project1_0.1.1的压缩包

建立一个包有许多繁琐的事情要做，所幸现在已经有了完善的制作包辅助工具，以下几个是必要的工具

```
> intall.packages(c('devtools','roxygen2','testthat','knitr'))
```

使用Rstudio中的git系统可以将包上传到自己的github上，而直接在线加载github上的开发包时需要用devtools包中的命令install_github(‘github名/package名’)
如：

```
> library(devtools)
> ?install_github
> install_github('exoplanetX/greyforecasting')
Downloading GitHub repo exoplanetX/greyforecasting@master
```

参考文献推荐：

《R包开发》杨学辉 译(2016)，Hadley Wickham(2015)