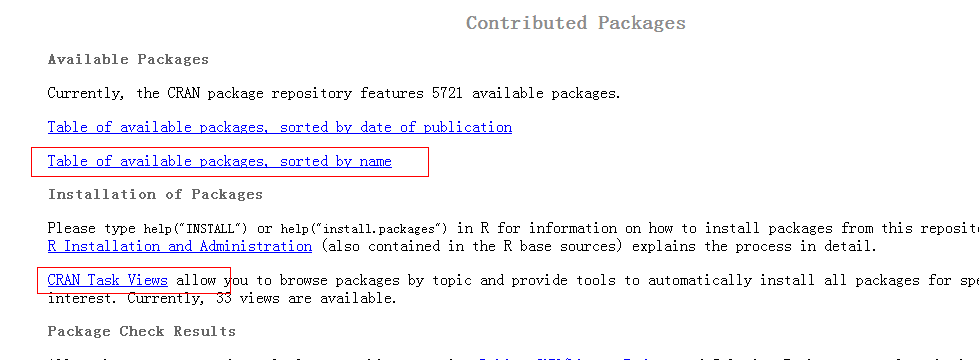
# R programing

# Week 1学习笔记

## 1 overview and history of R

R系统可分为两大部分，

* The “base” R system 安装的时候会默认安装一些基本的package
* Everything else。 以后在使用一些功能的时候，需要自行下载package安装

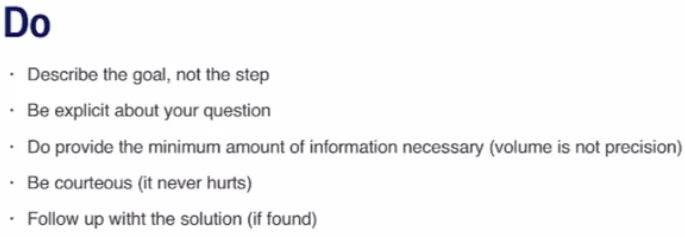
Package的查找可以是在<http://cran.r-project.org/> 上可以按照package的名称或者是要实现的功能类别。

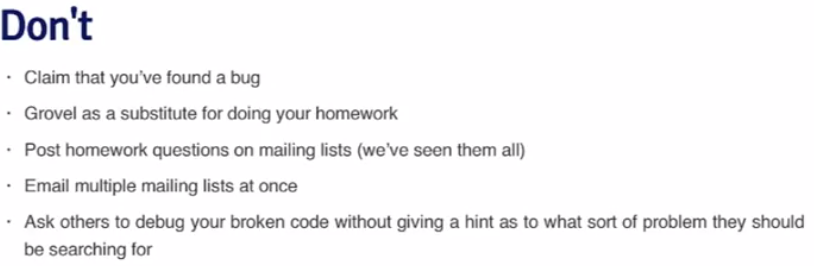
R主要上的《An Introduction to R》，在群里上传了一个中文版的，适合于快速入门。

## 2 Get help

Asking questions







## 3 data types

R has five basic or “atomic” classes of objects

* character
* numeric (real numbers)

1L，inf，NaN

* integer
* complex
* logical (True/False)

### R里面常用的几种对象形式

* **向量**
* **矩阵**（matrix）或者更为一般的数组（array）是多维的广义向量。
* **因子**（factor）为处理分类数据提供的一种有效方法。
* **列表**（list）是一种泛化（general form）的向量。它没有要求所有元素是同一类型，许多时候它本身就是向量和列表类型。列表为统计计算的结果返回提供了一种便利的方法。
* **数据框**（data frame）是和矩阵类似的一种结构。在数据框中，列可以是不同的对象。可以把数据框看作是一个行表示观测个体并且（可能）同时拥有数值变量和分类变量的`数据矩阵'
* **函数**（function）是可以保存在项目工作空间的 R 对象。该对象为 R 提供了一个简单而又便利的功能扩充方法。

### (2) 对象的类

R 里面的所有对象都属于一个类（class），可以通过函数**class**()函数查看。对于简单的向量，就是对应的模式"numeric"，"logical"，"character" 或者"list"，但

是"matrix"，"array"，"factor" 和"data.frame" 就可能是其他值

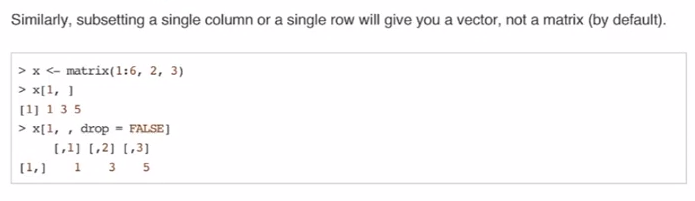
### (3) Missing value——NA NaN

NA: 在某些情况下，向量的元素可能有残缺。当一个元素或者值在统计的时候\不可得到"（not available）或者\值丢失" （missing value），相关位置可能会被保留并且赋予一个特定的值NA。任何含有NA 数据的运算结果都将是NA

NaN: 数值计算会产生第二种\缺损"值，也称为非数值（Not a Number）NaN

总之，对于NA 和NaN 用is.na(xx) 检验都是 TRUE。为了区分它们，is.nan(xx)就只对是NaN 元素显示TRUE。

## 4 Subsets





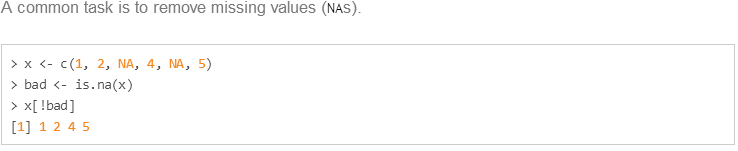
List的形式比较麻烦，在转化的时候有时也经常出错。

* 列表的分量可以用Lst[[1]], Lst[[2]]这样的形式独立访问。如果Lst[[2]] 是一个有下标的数组，那么Lst[[2]][1] 就是该数组的第一个元素。
* 列表的分量可以被命名，这种情况下可以通过名字访问

Lst$component\_name，这样非常方便。并且等价于Lst[[component\_name]]

### Missing values

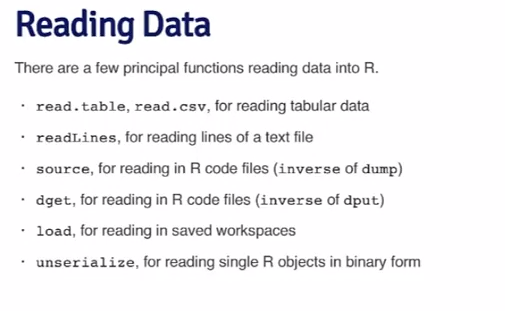
1. 删除缺失值



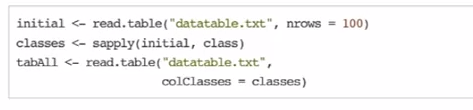
1. **Complete.cases()**

返回值：A logical vector specifying which observations/rows have no missing values across the entire sequence.

## 5 reading and writing data



当读入的数据量较大时，可以设置一些参数减少R自行计算的时间，比如ColClasses



# Week 2学习笔记

## 1. coding standards

* use text editor
* indent your code
* limit the width of your code.
* limit the length of your functions.

## 2. Scoping rules

当R在将一个值赋值给一个符号的时候，它会在一系列的environment中去找，比如在当前的workspace中编写了一个lm()的函数。具体的顺序如下：

- globalenvironment：workspace

- search list中的包的namespaces，即去各种包中去找有没有符合的名字

其中serch list可以通过 search函数找到

search()

## [1] ".GlobalEnv" "package:stats" "package:graphics"   
## [4] "package:grDevices" "package:utils" "package:datasets"   
## [7] "package:methods" "Autoloads" "package:base"

* search list上的顺序是会变的，global environment始终在第一个，如果你新library()了一个package，他会被放在第二个顺序，base包始终是最后的位置。
* R会区分函数名和非函数名，所以两者名字一样是可以的

R使用lexical scoping

* free variables的值在函数定义的environment中去寻找  
   free variable示例如下：函数中的z
* f<-function(x,y){  
   x+y+z  
  }
* environment：symbol——value对的集合
* A function+an environment=a function closure
* free variiable搜寻路线：  
   函数定义的环境==>parent environment ==>top-level environment(workspace或者package namespace) ==> empty environment ==> 报错
* fuction closure可以通过ls(environment(fun))的形式查看

lexical scoping与dynamic scoping  
例子：

y<-10  
f<-function(x) {  
 y<-2  
 y^2+g(x)  
}  
g<-function(x) {  
 x\*y  
}

lexical/static scoping:在g()中y是free variable，所以在函数定义的环境即global environment中寻找，所以y=10  
dynamic scoping：y的值在该函数被调用的环境(calling environment)中去寻找。所以y=2

## 3.vectorized operation

## 4. Dates and Time in R

Dates是用Date类表示的，time是用POSIXct或者POSIXlt类表示的。最终的储存方式都是按照距离1970-01-01的日期存储的数字。

x<-as.Date("1970-01-01")  
x

## [1] "1970-01-01"

unclass(x)

## [1] 0

常用的函数  
- as.Date()

- as.POSIXct,as.POSIXlt

- strptime() 将字符串时间转化为时间格式

- weekdays(),months(),quarters()

months(Sys.time())

## [1] "七月"

# week 3 学习笔记

loop functions

- lapply: list，fun是针对list中的每一个element

- sapply:简化lapply的结果

- apply：apply a function over margins of an array(行/列维度)

- tapply：对一个向量的子向量

- mapply:lapply的多元版本

## 1. lapply

用法： lapply(x,fun,..)  
返回：a list  
e.g.1

x<-list(a=1:4,b=rnorm(10))  
lapply(x,mean)

## $a  
## [1] 2.5  
##   
## $b  
## [1] 0.08154

e.g.2 如果函数需要传递参数，可以这么写:

x<-1:3  
lapply(x,runif,min=0,max=10) #设置runif的参数

## [[1]]  
## [1] 8.664  
##   
## [[2]]  
## [1] 7.5286 0.2912  
##   
## [[3]]  
## [1] 5.648 4.145 2.047

e.g.3自定义函数  
对于lapply中的函数可以编写自己的函数

## 2. sapply

lapply的简化版:可能简化成vector,matrix,list

## 3. apply

* 多用在计算矩阵或array的行、列、维度上
* 输入简单，运算并不快

用法：apply(x,margin,fun,...)  
margin是标志维度的数字向量。此函数可以求矩阵的行和、列和。均值等等。有一些计算会有一些快捷的方式，比如：  
- rowSums，rowMeans - colSums，colMeans

e.g.

a<-array(rnorm(2\*2\*10),c(2,2,10))  
apply(a,c(1,2),mean)

## [,1] [,2]  
## [1,] -0.16127 -0.60748  
## [2,] 0.01678 -0.03511

rowMeans(a,dims=2)

## [,1] [,2]  
## [1,] -0.16127 -0.60748  
## [2,] 0.01678 -0.03511

说明：For row##, the sum or mean is over dimensions dims+1, ...; for col## it is over dimensions 1:dims

## 4.tapply

apply a function over subsets of a vector.  
用法:tapply(x,INDEX,fun,...,simply=TRUE) - x是向量 - INDEX是factor或者是factor的list

e.g

x<-c(rnorm(10),runif(10),rnorm(10,1))  
#gl函数产生指定数量和水平的factor数据  
f<-gl(3,10,labels=c("Group1","Group2","Group3"))  
f

## [1] Group1 Group1 Group1 Group1 Group1 Group1 Group1 Group1 Group1 Group1  
## [11] Group2 Group2 Group2 Group2 Group2 Group2 Group2 Group2 Group2 Group2  
## [21] Group3 Group3 Group3 Group3 Group3 Group3 Group3 Group3 Group3 Group3  
## Levels: Group1 Group2 Group3

tapply(x,f,mean)

## Group1 Group2 Group3   
## 0.01867 0.39559 1.57763

## 5.split

split divides the data in the vector x into the groups defined by f. The replacement forms replace values corresponding to such a division. unsplit reverses the effect of split.  
用法: split(x, f, drop = FALSE, ...)  
e.g.1一个factor

x<-c(rnorm(10),runif(10),rnorm(10,1))  
f<-gl(3,10,labels=c("Group1","Group2","Group3"))  
split(x,f)

## $Group1  
## [1] -0.5247 -0.8065 1.3797 -0.8151 0.3902 1.0709 0.7742 -0.7792  
## [9] 0.5801 1.1859  
##   
## $Group2  
## [1] 0.49893 0.61792 0.29288 0.53044 0.09394 0.23518 0.06262 0.79696  
## [9] 0.53925 0.06700  
##   
## $Group3  
## [1] 0.7936 1.5305 1.2961 1.9638 1.7261 1.8081 2.3906 2.4395 1.2061 3.4965

#  
lapply(split(x,f),mean)

## $Group1  
## [1] 0.2456  
##   
## $Group2  
## [1] 0.3735  
##   
## $Group3  
## [1] 1.865

e.g2多个factor

x<-rnorm(10)  
f1<-gl(2,5)  
f2<-gl(5,2)  
interaction(f1,f2)

## [1] 1.1 1.1 1.2 1.2 1.3 2.3 2.4 2.4 2.5 2.5  
## Levels: 1.1 2.1 1.2 2.2 1.3 2.3 1.4 2.4 1.5 2.5

注：interaction()函数将factor进行交互，返回的也是factor。默认以.分隔  
或者直接通过list的方式：  
split(x,list(f1,f2),drop=T)   
drop会将没有的factor去掉

## 6.mapply

lapply或者sapply的多元版本。比如有两个list，对每一个list使用不同的函数进行操作。  
用法：mapply(FUN,...,args) e.g.1

list(rep(1:4),rep(2,3),rep(3,2),rep(4,1))等价于   
mapply(rep,1:4,4:1)

## aggregate()

aggregate(NEI\_1$Emissions, by=list(NEI\_1$year,NEI\_1$type), FUN="sum")

实现类似于 分类汇总 group by的功能

## 7.Debugging

* traceback： 最近的一次错误，函数的调用层次
* debug
* brower
* trace
* recover

# week 4 学习笔记

R中常用的函数

## 1.str

展示对象的结构  
summary, str

## 2.Simulation

### 产生随机数

* rnorm() 正态分布随机数
* d：密度函数
* r：随机数
* p:累积分布
* q:分位数

set.seed() 设置随机种子，可以重复产生相同的随机数

### 抽样函数

sample(,,replace=)

## 3.profile

一些有用函数  
1. system.time()  
返回一个proc\_time的类型 - user time - elapsed time system.time({一段比较长的表达式})

1. Rprof()  
   Rprof() 计算每段函数的时间  
   summaryRprof()

## 4.optimization