数据分析与处理技术

泛函与优化

从函数到泛函

函数封装了对数据进行计算的代码，以此避免重复类似的计算过程
而泛函将函数作为参数使用，避免了循环
为了理解泛函，首先建立一个简单函数

```
> f=function(x){
+ a=quantile(x)
+ s=a[4]-a[2]
+ names(s)=NULL
+ return(s)
+ }
> f(1:5)
[1] 2
```

函数f用于计算向量x的四分位距，并将附带的数据名称删除掉

认识一个重要的泛函lapply，对list类型变量按元素施加函数f的作用，即将iris[,1:4]四个元素依次作为参数放入f函数中运算

```
> lapply(iris[,1:4],f)
$Sepal.Length
[1] 1.3

$Sepal.Width
[1] 0.5

$Petal.Length
[1] 3.5

$Petal.Width
[1] 1.5
```

既然是以list变量方式运算，不难理解它的结果依然是以list的元素方式返回

```
> unlist(lapply(iris[,1:4],f))
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
         1.3          0.5          3.5          1.5
```

为了得到满意的格式，需要先拆掉list类的格式，即退回了原子向量，之后再用as.一族函数进行转类型

并行运算

lapply并非真正的并行运算，而是类似循环的方式调用函数施加运算。但它的parallel包中的升级版则真正实现了并行运算。

首先检测能够调用计算机几个运算核心

```
> detectCores()
[1] 4
```

利用可调用的核心创建并行运算集群，然后使用并行泛函

```
> cluster_p<-makePSOCKcluster(4)
> parLapply(cluster_p,iris[1:4],f)
```

切割数据split

数据集的切割函数split，将数据集按照某分类属性拆分成多个子集，各子集以list元素方式存储。观察下列结果，按照iris花的种类进行切割

```
> g=split(iris,iris$Species)
```

切割数据为使用泛函带来了便利，可以分段处理数据

```
> lapply(g$setosa[,1:4],f)
> lapply(g$versicolor[,1:4],f)
> lapply(g$virginica[,1:4],f)
```

上述三行代码分别对iris的三个子集中的前四属性进行了f函数作用，但计算结果是以列表方式分别列出，如果需要整齐格式则比较麻烦

apply：矩阵操作	lapply:操作list大类变	tapply：因子切割向量
	同类变形代码	
	sapply	by: 作用于数据框
	vapply	

```
> test_df
     col1 col2 col3 var4
[1,]    1    4    7   -1
[2,]    2    5    8   -6
[3,]    3   99    9   99
[4,]   15   12  -90  -70
> apply(test_df,1,mean)
[1]   2.75   2.25  52.50 -33.25
```

```
> tapply(cholesterol$response,cholesterol$trt,mean)
   1time   2times   4times    drugD    drugE
 5.78197  9.22497 12.37478 15.36117 20.94752
```

矩阵泛函中第二个参数1表示按行计算，2表示按列计算

聚合函数aggregate

理解了基本泛函apply函数族，再进一步认识几个具备整合功能的泛函

```
> aggregate(.~Species,data=iris,f)
     Species Sepal.Length Sepal.Width Petal.Length Petal.Width
1     setosa        0.400       0.475        0.175         0.1
2 versicolor        0.700       0.475        0.600         0.3
3  virginica        0.675       0.375        0.775         0.5
```

aggregate能够直接按照分类属性作为控制变量进行分子集计算，并将结果整齐的呈现

除以上几个泛函外，dplyr包里的summarise也是一个具备整合功能的泛函

Split-apply-combination

更具创造性的是Hadley Wickham近年来做的plyr包，其实dpylr便是专用拓展包之一。plyr将切割-使用-组合做了完美整合，注意下表中plyr泛函名称变化

输入格式

输出格式	**array**	**data.frame**	**list**
array	aaply	daply	laply
data.frame	adply	ddply	ldply
list	alply	dlply	llply
_(无输出)	a_ply	d_ply	l_ply

plyr包的泛函效率极高，目前已经得到数据科学广泛应用

```
> adply(m,1,f)
  X1  V1
1  1 4.5
2  2 4.5
3  3 4.5
> aaply(m,1,f)
  1   2   3
4.5 4.5 4.5
```

作者原文将思路和用法介绍的非常清楚，参见
https://www.jstatsoft.org/article/view/v040i01

练习

对mtcars数据集，测算

```
> lapply(mtcars,mean)
```

注意：泛函计算list类型时按元素处理数据，data.frame显然也是适用

```
> b<-unlist(lapply(mtcars,mean))
```

由于计算结果固定返回为list结构，我们可以强制取消list结构还原回原子向量

单变量函数优化

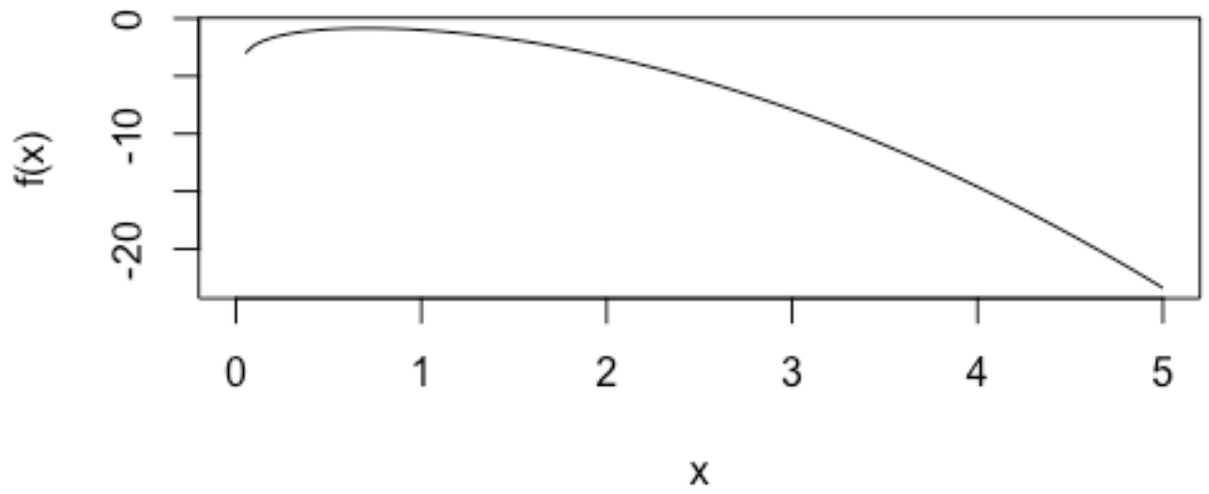
求解函数 $f(x)=\ln(x)-x^2$ 在0到5区间上的最小值

```
> optimize(f,c(0,5))
$minimum
[1] 4.999922

$objective
[1] -23.3898
```

```
> optimize(f,c(0,5),maximum = T)
$maximum
[1] 0.7070898

$objective
[1] -0.8465736
```



规划求解器

求解规划问题原理不复杂，但求解大型规划问题却极为困难，目前已有的规划求解器主要是Gurobi,IBM的Cplex,Glpk等。这些求解器在R中都有应用接口包，并且格式比较接近。以glpk的包rglpk包为例：

求解线性规划问题：

$$\max: z = 2x_1 + 4x_2 + 3x_3$$

$$\begin{cases} 3x_1 + 4x_2 + 2x_3 \le 60 \\ 2x_2 + x_2 + 2x_3 \le 40 \\ x_1 + 3x_2 + 2x_3 \le 80 \\ x_1, x_2, x_3 \ge 0 \end{cases}$$

```
> library(slam)
> library(Rglpk)

> obj<-c(2,4,3)
> mat<-matrix(c(3,2,1,4,1,3,2,2,2),nrow=3)
> dire<-c("<=","<=","<=")
> rhs<-c(60,40,80)
> Rglpk_solve_LP(obj,mat,dire,rhs,max=TRUE)
```

参数的含义对比左侧线性规划式，其中max参数控制求最大问题还是最小问题