beta

[原]R语言与点估计学习笔记（矩估计与MLE）

2012-10-13 阅读5261 评论3

众所周知，R语言是个不错的统计软件。今天分享一下利用R语言做点估计的内容。主要有：矩估计、极大似然估计、EM算法、最小二乘估计、刀切法（Jackknife）、自助法（Bootstrap）的相关内容。

点估计是参数估计的一个组成部分。有许多的估计方法与估计理论，具体内容可以参见lehmann的《点估计理论》（推荐第一版，第二版直接从UMVU估计开始的）

一、矩估计

对于随机变量来说，矩是其最广泛，最常用的数字特征，母体的各阶矩一般与的分布中所含的未知参数有关，有的甚至就等于未知参数。由辛钦大数定律知，简单随机子样的子样原点矩依概率收敛到相应的母体原点矩。这就启发我们想到用子样矩替换母体矩，进而找出未知参数的估计，基于这种思想求估计量的方法称为**矩法**。用矩法求得的估计称为**矩法估计**，简称**矩估计**。它是由英国统计学家皮尔逊Pearson于1894年提出的。

因为不同的分布有着不同的参数，所以在R的基本包中并没有给出现成的函数，我们通常使用人机交互的办法处理矩估计的问题，当然也可以自己编写一些函数。

首先，来看看R中给出的一些基本分布，如下表：

概率分布	R 对应的名字	附加参数
β 分布	beta	shape1, shape2, ncp
二项式分布	binom	size, prob
Cauchy 分布	cauchy	location, scale
卡方分布	chisq	df, ncp
指数分布	exp	rate
F分布	f	df1, df1, ncp
γ 分布	gamma	shape, scale
几何分布	geom	prob
超几何分布	hyper	m, n, k
对数正态分布	lnorm	meanlog, sdlog
logistic 分布	logis	location, scale
负二项式分布	nbinom	size, prob
正态分布	norm	mean, sd
Poisson 分布	pois	lambda
t 分布	t	df, ncp
均匀分布	unif	min, max
Weibull 分布	weibull	shape, scale
Wilcoxon 分布	wilcox	m, n

虽然R中基本包中没有现成求各阶矩的函数，但是对于给出的样本，R可以求出其平均值（函数：mean），方差（var），标准差（sd），在fBasics包中还提供了计算偏度的函数skewness(),以及计算峰度的kurtosis()。这样我们也可以间接地得到分布一到四阶矩的数据。由于低阶矩包含信息较为丰富，矩估计也一般采用低阶矩去处理。

注：在actuar包中，函数emm（）可以计算样本的任意阶原点矩。但在参数估计时需要注意到原点矩的存在性

例如我们来看看正态分布N（0,1）的矩估计效果。

```
> x<-rnorm(100)    #产生N（0,1）的100个随机数
> mu<-mean(x)    #对N(mu,sigma)中的mu做矩估计
> sigma<-var(x)    #这里的var并不是样本方差的计算函数，而是修正的样本方差，其实也就是x的总体方差
> mu
[1] -0.1595923
> sigma
[1] 1.092255
```

可以看出，矩估计的效果还是可以的。

我们再来看一个矩估计的例子：设总体X服从二项分布B(k,p)，X1,X2,…,Xn，是总体的一个样本。K,p为未知参数。那么k,p的矩估计满足方程：$kp - M_1 = 0$, $kp(1 - p) - M2 = 0$.

我们可以编写函数：

```
moment <-function(p){
f<-c(p[1]*p[2]-M1,p[1]*p[2]-p[1]*p[2]^2-M2)
J<-matrix(c(p[2],p[1],p[2]-p[2]^2,p[1]-2*p[1]*p[2]),nrow=2,byrow=T)#jicobi矩阵
list(f=f,J=J)
}# p[2]=p, p[1]=k,
```

检验程序

```
 x<-rbinom(100, 20, 0.7); n<-length(x)
 M1<-mean(x);M2<-(n-1)/n*var(x)
 p<-c(10,0.5)
Newtons(moment, p)$root #是用牛顿法解方程的程序，见附件1
```

运行结果为：

[1]22.973841 0.605036

可以得到k,p的数值解

二、极大似然估计（MLE）

极大似然估计的基本思想是：基于样本的信息参数的合理估计量是产生获得样本的最大概率的参数值。值得一提的是：极大似然估计具有不变性，这也为我们求一些奇怪的参数提供了便利。

在单参数场合，我们可以使用函数optimize（）来求极大似然估计值，函数的介绍如下：

optimize(f = , interval = , ..., lower = min(interval),

upper = max(interval), maximum = FALSE,

tol = .Machine$double.eps^0.25)

例如我们来处理Poisson分布参数lambda的MLE。

设X1,X2,…,X100为来自P（lambda）的独立同分布样本，那么似然函数为：

L（lambda，x）=lambda^(x1+x2+…+x100)*exp(10*lambda)/(gamma(x1+1)…gamma(x100+1))

这里涉及到的就是一个似然函数的选择问题：是直接使用似然函数还是使用对数似然函数，为了说明这个问题，我们可以看这样一段R程序：

> x<-rpois(100,2)

> sum(x)

[1] 215

> ga(x)#这是一个求解gamma(x1+1)…gamma(x100+1)的函数，用gamma函数求阶乘是为了提高计算效率（源代码见附1）

[1] 1.580298e+51

> f<-function(lambda)lambda^(215)*exp(10*lambda)/(1.580298*10^51)#这里有一些magic number + hard code 的嫌疑，其实用ga(x)带入，在函数参数中多加一个x就好

> optimize(f,c(1,3),maximum=T)

$maximum

[1] 2.999959

$objective

[1] 2.568691e+64

> fun<-function(lambda)-100*lambda+215*log(lambda)-log(1.580298*10^51)

> optimize(fun,c(1,3),maximum=T)

$maximum

[1] 2.149984 #MLE

$objective

[1] -168.3139

为什么会有这样的差别？这个源于函数optimize，这个函数本质上就是求一个函数的最大值以及取最大值时的自变量。但是这里对函数的稳定性是有要求的，取对数无疑增加了函数的稳定性，求极值才会合理。这也就是当你扩大了MLE存在区间时warning会出现的原因。当然，限定范围时，MLE会在边界取到，但是，出现边界时，我们需要更多的信息去判断它。这个例子也说明多数情况下利用对数似然估计要优于似然函数。

在多元ML估计时，你能用的函数将变为optim,nlm,nlminb它们的调用格式分别为：

```
optim(par, fn, gr = NULL, ...,
      method = c("Nelder-Mead", "BFGS", "CG", "L-BFGS-B", "SANN", "B
      lower = -Inf, upper = Inf,
      control = list(), hessian = FALSE)
nlm(f, p, ..., hessian = FALSE, typsize = rep(1, length(p)),
    fscale = 1, print.level = 0, ndigit = 12, gradtol = 1e-6,
    stepmax = max(1000 * sqrt(sum((p/typsize)^2)), 1000),
    steptol = 1e-6, iterlim = 100, check.analyticals = TRUE)
nlminb(start, objective, gradient = NULL, hessian = NULL, ...,
       scale = 1, control = list(), lower = -Inf, upper = Inf)
```

> x<-rnorm(1000,2,6) #6是标准差，而我们估计的是方差

> ll<-function(theta,data){

+ n<-length(data)

+ ll<--0.5*n*log(2*pi)-0.5*n*log(theta[2])-1/2/theta[2]*sum((data-theta[1])^2)

```
+ return(-ll)
+ }
>nlminb(c(0.5,2),ll,data=x,lower=c(-100,0),upper=c(100,100)) $par
 [1]  1.984345  38.926692
```

看看结果估计的还是不错的，可以利用函数mean，var验证对正态分布而言，矩估计与MLE是一致.

然而这里还有一些没有解决的问题，比如使用nlminb初始值的选取。希望阅读到这的朋友给出些建议。

最后指出在stata4，maxLik等扩展包中有更多关于mle的东西，你可以通过查看帮助文档来学习它。

附1：辅助程序代码

```
Newtons<-function (fun, x, ep=1e-5,it_max=100){
 index<-0; k<-1
 while (k<=it_max){
 x1 <- x; obj <- fun(x);
 x <- x - solve(obj$J, obj$f);
 norm <- sqrt((x-x1) %*% (x-x1))
 if (norm<ep){
 index<-1; break
 }
 k<-k+1
 }
 obj <- fun(x);
 list(root=x, it=k, index=index, FunVal= obj$f)
 }


ga<-function(x){
ga<-1
for(i in 1:length(x)){
ga<-ga*gamma(x[i]+1)
}
```

```
ga
}
```

上一篇 下一篇

请先登录后，再发表评论！

查看评论

3楼 yujun7654321 2015-02-07 22:52
是的

2楼 tianbwin2995 2015-02-07 18:05
大神，MLE估计的似然函数是每次得根据分布的不同自己拿函数来写出来吗?

1楼 yujun7654321 2012-10-13 03:08
EM算法、最小二乘估计、刀切法（Jackknife）、自助法（Bootstrap）的相关内容会在本周内补完

更多评论（3）

回顶部