§3.4 R 中的参数估计和假设检验

1. **参数估计**

- **矩估计**：假设总体 X 的分布函数含有 k 个未知参数 $\theta_1,\ldots,\theta_k$, 且分布的前 k 阶矩存在且都是 $\theta_1,\ldots,\theta_k$ 的函数. 求 θ_j 的矩估计的具体步骤如下：

  (i) 求出 $E(X^j)=\mu_j, j=1,\ldots,k$, 并假定

  $$\mu_j = g_j(\theta_1,\ldots,\theta_k), j=1,\ldots,k.$$

  (ii) 解上面的方程组得到

  $$\theta_j = h_j(\mu_1,\ldots,\mu_k), j=1,\ldots,k.$$

  (iii) 用样本的 k 阶原点矩 $m_k=\frac{1}{n}\sum_{i=1}^n X_i$ 来代替 μ_j 则得到 θ_j 的矩估计 $\widehat{\theta_j}=h_j(m_1,\ldots,,m_k)$.

  如果 h_j 可以显示表达, 则我们只需要适当的 R编程即可直接获得矩估计. 当然, 当方程组是非线性的时候, 我们根本无法得到显示的 h_j. 这时我们直接将 m_j 代入 (i), 则直接得到 k元的方程组. 解这样的方程求根问题一般采用迭代算法数值求解, 我们将会在稍后的章节中介绍一般的方法. R 中有不少解这种非线性方程组的 package, 比如 nleqlsv.

- **极大似然估计**: 设 $X_1,\ldots,X_n$ 为取自总体 X 的样本且其联合密度函数为 $L(\boldsymbol{\theta})=\prod_{i=1}^n f(X_i|\boldsymbol{\theta})$. 极大似然估计即是求使得 $L(\boldsymbol{\theta})$ 或对数似然函数 $l(\boldsymbol{\theta})=\log(L(\boldsymbol{\theta}))$ 达到最大的参数 $\boldsymbol{\theta}$ 的值. 对于没有显示解的情形, 我们通常要使用优化方法来求极大似然估计值. 在 R 中, 对于单参数场合, 我们可以使用函数 optimize() 求极大似然估计值.

  optimize(f=, interval=, maximum=F, tol=)

  其中, f 是似然函数, interval 是参数 θ 的取值范围, maximum=F 是求最小值, tol 则是求值的精度. 对于多参数场合, 我们可以使用函数 nlm() 或者 optim() 求极大似然估计值. 这里仅给出 nlm 的法, 关于 opti()可参见帮助, 我们在下一章介绍了各种优化方法后会再介绍 opti. nlm 的基本用法是

  nlm(f, p, steptol=1e-6, iterlim=100)

  其中 f 是似然函数, p 为给定的初始值, steptol 为求值的精度, 而 iterlim 是我们所容许的最大的迭代次数. 该函数采用 Newton-Raphson 算法, 函数的返回值是一个列表, 包含极小值、极小点的估计值等.

- **区间估计**. 矩估计和极大似然估计都是点估计, 是参数的一个近似值, 因而它们都不能给出估计的误差范围, 亦没能指出这个误差范围以多大的概率包括未知参数. 而区间估计则能够给出的估计的可信程度. 最常用的区间估计主要是基于正态总体(或

是某个参数分布)假设下的, 我们在这里介绍的内容也基于这一假设, 而基于非参数的区间估计的推断我们将在后面的章节中介绍.

(1) 正态总体的单样本均值和两样本均值差的区间估计.

R 中提供了函数 t.test 来处理方差未知情形下该类区间估计的问题. 而对于方差已知的情形, 其仅限于书本上的讨论, 在实际问题中并不常用, 因为我们通常在仅给定样本情况下假设方差或均值已知都是不合理的.

单样本的均值 μ 的置信区间是根据 $\frac{\bar{X}_n-\mu}{S_n/\sqrt{n}} \sim t(n-1)$ 得到的, 即 μ 的置信水平为 $1-\alpha$ 的置信区间是

$$\left(\bar{X}_n - \frac{S}{\sqrt{n}} t_{1-\frac{\alpha}{2}}(n-1), \bar{X}_n + \frac{S}{\sqrt{n}} t_{1-\frac{\alpha}{2}}(n-1)\right),$$

其中 $t_p(n)$ 为自由度为 n 的 t 分布的下侧 p 分位数.

对于两样本问题的均值差 $\mu_1-\mu_2$ 当两方差均未知时仅当 $\sigma_1^2=\sigma_2^2=\sigma^2$ 时(或 $\sigma_1^2=\theta\sigma_2^2, \theta$ 已知) 我们才可得到精确的置信区间. 其是根据 $\frac{\bar{X}_n-\bar{Y}_n-(\mu_1-\mu_2)}{S_w\sqrt{\frac{n_1+n_2}{n_1n_2}}} \sim t(n_1+n_2-2)$ 得到的, 其中 $S_w^2=\frac{(n_1-1)S_1^2+(n_2-1)S_2^2}{n_1+n_2-2}$, 即 $\mu_1-\mu_2$ 的置信水平为 $1-\alpha$ 的置信区间是

$$\bar{X}_n - \bar{Y}_n \pm S_w\sqrt{\frac{n_1+n_2}{n_1n_2}} t_{1-\frac{\alpha}{2}}(n_1+n_2-2)$$

当方差不等时, 我们通常采用 Welch 近似的方法, 所得到的置信区间形如.

$$\bar{X}_n - \bar{Y}_n \pm t_{1-\frac{\alpha}{2}}(v)S^*,$$

其中 v是近似的自由度, $S^*=\frac{S_1^2}{n_1}+\frac{S_2^2}{n_2}$.

t.test一般用法如下：t.test(x, y=null, alternative=c("two.sided", "less", "greater"), mu=0, paired=F, var.equal=F, conf.level=0.95)

由于置信区间和显著性水平假设检验通常来说是统一的, t.test 即可同时完成二者. 上面的命令中: 若仅有数据 x, 则进行单样本 t 检验; 若给出数据 y, 则进行两样本的 t 检验; alternative 用于指定所求置信区间的类型, 即置信区间, 置信下限, 置信上限; mu 仅在假设检验中起作用.

举例如下：我们考虑 possum 数据. 一般我们在使用基于正态假设构造置信区间前需要做正态性检验, 如果我们接受正态的假设, 则我们可认为构造出来的置信区间的精确程度应该很高; 而如果拒绝正态假设, 由统计大样本理论知道我们仍可使用同样的方法来构造, 但此时我们自己应该牢记可能构造出来的置信区间的覆盖率与我们实际的要求相差一定距离, 当然如果 n 足够大时, 这样的置信区间的覆盖率会接近我们想要的值.

先来看一个单样本的例子：

```
library(DAAG); shapiro.test(possum$hdlngth);
```

qqnorm(possum$hdlngth); qqline(possum$hdlngth)

t.test(x1); ci<–t.test(x1)$conf.int (提取 t.test 的置信区间的输出结果);

attributes(ci)<–NULL (表示去掉 ci 的属性使之成为一标准的二维向量);

ci<–t.test(x1, alternative="less", conf.level=0.99)$conf.int

假如我们想要得到置信水平为 0.01, 0.05, 0.10 的区间, 可以

alpha<–c(0.01, 0.05, 0.10); ci<–array(0,c(3,2)); for (i in 1:3){ x2<–t.test(x1, conf.level=1-alpha[i])$conf.int; attributes(x2)<–NULL; ci[i,]<–x2}; ci

再来看一个两样本的例子：

x1<-possum$hdlngth[possum$sex=="f"]; n1<–length(x1); var(x1)

x2<-possum$hdlngth[possum$sex=="m"]; n2<–length(x2); var(x2)

shapiro.test(x1); shapiro.test(x2);

t.test(x1,x2)$conf.int; t.test(x1,x2,var.equal=T)$conf.int

(2) 正态总体的单样本方差及两样本方差比的区间估计.

单样本的方差 σ^2 的置信区间是根据 $\frac{(n-1)S_n^2}{\sigma^2} \sim \chi^2(n-1)$ 得到的, 即 σ 的置信水平为 $1-\alpha$ 的置信区间是

$$\left(\frac{(n-1)S_n^2}{\chi^2_{1-\frac{\alpha}{2}}(n-1)}, \frac{(n-1)S_n^2}{\chi^2_{\frac{\alpha}{2}}(n-1)} \right),$$

R 中没有提供求单样本 σ^2 置信区间的函数, 需要我们自己编写, 留作作业.

对于两样本问题的方差比 σ_1^2/σ_2^2 的置信区间, 其是根据 $\frac{S_1^2/\sigma_1^2}{S_2^2/\sigma_2^2} \sim F(n_1-1, n-2-1)$ 得到的, 即 σ_1^2/σ_2^2 的置信水平为 $1-\alpha$ 的置信区间是

$$\left(\frac{S_1^2/S_2^2}{F_{\alpha/2}(n_1-1,n_2-1)}, \frac{S_1^2/S_2^2}{F_{1-\alpha/2}(n_1-1,n_2-1)} \right).$$

R 软件中提供了该两样本方差比区间估计的函数 var.test(), 其一般调用格式如下：

var.test(x, y, ratio=1, alternative=c("two.sided", "less", "greater"), conf.level=0.95)

其中 ratio 只是在作假设检验的时候才有用. 仍以 possum 数据为例:

var.test(x1, x2)$conf.int

2. **假设检验**

在 R 中假设检验的作法与上述的区间估计基本是一致的, 使用同样的命令即可获得检验的结果. 这里不再赘述. 仅举例如下：

t.test(possum$hdlngth, mu=93.0);

t.test(possum$hdlngth, alternative="less", mu=93.0); 这里 less 表示单边检验 $H_0: \mu \geq \mu_0$ vs $H_1: \mu < \mu_0$.

t.test(x1, x2); t.test(t.test(x1, x2), var.equal=T);

t.test(x1, x2, alternative="less"); 这里 less 为单边检验 $H_0 : \mu_1 \geq \mu_2$ vs $H_1 : \mu_1 < \mu_2$.

另外, 我们有时要做成对数据的 t 检验, 即指两个样本的样本容量相同, 两组数据分别是同一观测个体在两种不同状况下得到的观测. 这时, 我们仅需要在 t.test() 函数中添加 paired=T, 即可, 举例来说：

shapiro.test(intake$pre); shapiro.test(intake$pre);

t.test(intake$pre, intake$post, paired=T)

检验两方差比例的检验 var.test(x1, x2, ratio=1)

另外, R 中还提供了关于二项分布比率检验的函数, binom.test(). 也就是假设 $X_1, \ldots, X_n$ 来自两点分布 BIN$(p, 1)$, 而 $T = \sum_{i=1}^{n} X_i \sim$ BIN(n, p). 检验 $H_0 : p = p_0$ vs $H_1 : p \neq p_0$. 具体用法参见帮助, 这里不详细介绍了.

第4章　非线性方程数值解及优化方法

统计学中的一个重要问题是MLE估计问题, 解决这样问题的关键是寻找似然方程的最优解. 在很多情况下, 我们不能直接得到似然方程的显式解, 需要通过数值分析的方法得到方程的解. 除极大似然外, 统计学中也有许多其它的优化问题, 如在 Bayes 决策问题中的最小风险、非线性最小二乘问题的求解问题等.

上述求解都属于如下的一般问题：

$$\arg\min_{\boldsymbol{\theta}\in D} g(\boldsymbol{\theta}), \tag{4.1}$$

其中 g 是参数向量 $\boldsymbol{\theta}$的函数，称之为目标函数 (*objective function*), 而 $\boldsymbol{\theta}$ 我们有时亦称之为决策变量 (*decision variable*). 决策变量的取值区域 D 称为可行集合 (feasible set) 或者 候选集合 (candidate set). 由于最大化一个函数等价于其负值的最小化 ($-g(\boldsymbol{\theta})$), 故区别最大与最小的意义不大. 于是作为惯例, 我们一般将考虑求取最小值的算法.

这里我们需要区分有约束和无约束两种优化问题. 当 D 就是 $g(\boldsymbol{\theta})$ 的定义域时, 问题4.1就是一无约束优化问题 (*unconstrained optimization problem*); 否则, 即是有约束优化问题 (*constrained optimization problem*). 另外, 在取值区域内 D 的某个特定子域内, 可能有某个局部最小值, 而在另外一个特定子域内存在另一个局部最小值. 我们之后称全局最优 (*global optimum*) 即指在取值区域内 D 的最小值; 称局部最优 (*local optimum*) 即指在取值区域内 D 的某个子域内的最小值.

在本章, 我们将主要考虑 g 关于 θ 为光滑且可微的情形, 而 g 在离散区域上的优化问题将在最后给予介绍.

§4.1　单变量方程求根问题

我们知道, 很多优化问题都等价于是一个方程求根问题. 因此, 我们首先来讨论一元方程求根问题的数值解法, 即对于给定的关于 x 的函数 g 寻找 x 使得 $g(x) = 0$.

问题：求解函数 $\log x/(1+x)$ 的最大值? 其等价于求方程

$$g(x) = \frac{1 + 1/x - \log x}{(1+x)^2} = 0 \Leftrightarrow 1 + 1/x - \log x = 0$$

的解. 显然没有显式解。

我们介绍一些常用的迭代算法. 我们接下来假设 g 是一连续函数.

§4.1.1　二分法 (bisection method)

一、原理：如果 g 在区间 $[a_0, b_0]$ 上连续, 且 $g(a_0)g(b_0) \leqslant 0$, 则由中值定理知, 至少存在一个 $x^* \in [a_0, b_0]$，使得 $g(x^*) = 0$. 我们称 $[a_0, b_0]$ 为有根区间.

取区间 $[a_0, b_0]$ 中点 $x_0 = (a_0 + b_0)/2$ 将它分为两半, 若 $g(x_0) = 0$, 则 x_0 就是 $g(x) = 0$ 的一个根; 否则由 x_0 分成的两个区间中必有一个是有根区间, 记为 $[a_1, b_1]$. 此时即把区间 $[a_0, b_0]$ 缩短至 $[a_1, b_1]$, 其长度为原区间的一半. 如此反复进行, 可得到一系列有根区间 $[a_0, b_0] \supset [a_1, b_2] \supset [a_2, b_2] \supset \cdots$, 其中每个区间都是前一个区间的一半, 因此 $[a_k, b_k]$ 的长度 $b_k - a_k = (b_0 - a_0)/2^k$. 当 $k \to \infty$ 时其趋于零, 就是说, 如果二分过程无限地继续下去, 这些区间将最终收缩于一点, 该点显然为所求的根 x^*. 当然在实际计算时, 我们无法完成这一无限过程, 且也无这种必要, 因为我们数值计算的结果是允许误差的. 将以上描述总结为如下算法:

算法 4.1.1. 二分法求方程根:

(0) 令 $k = 0$, 且找到一有根区间 $[a_0, b_0]$.

(1) 令 $k = k + 1$ 且 $x_k = (a_{k-1} + b_{k-1})/2$.

(2) 如果 $\mathrm{sgn}(g(x_k)) = \mathrm{sgn}(g(a_{k-1}))$ 则令 $a_k = x_k$; 否则 $b_k = x_k$.

(3) 如果达到某一收敛准则, 则停止并返回解 x_k. 否则返回第一步.

关于收敛准则：绝对收敛和相对收敛. 绝对收敛的停止准测：

$$|x_{t+1} - x_t| < \epsilon, \tag{4.2}$$

其中常数 ϵ 是选定的可容忍精度. 对于上述二分法, 不难看出 $|x_{t+1} - x_t| = (b_t - a_t)/2 = (b_0 - a_0)/2^{t+1}$. 只要 t 足够大，则 (4.2) 就能成立.

相对收敛准测：

$$\frac{|x_{t+1} - x_t|}{|x_t|} < \epsilon \tag{4.3}$$

时停止迭代. 此准则可以在不必考虑 x 的单位的情况下达到指定的目标精度，如1%.

我们应该根据实际问题来选择应用绝对还是相对收敛准则. 如果 x 的刻度相对于 ϵ 很大（或很小）时，绝对收敛准则有时将不如我们所愿地停止迭代或迭代很快就停止了. 相对收敛准则对 x 的刻度做了校正，但当 $x^{(t)}$ 的值（或真值）与 0 非常接近时，它将变得不稳定，此时，我们可以通过 $\frac{|x_{t+1} - x_t|}{|x_t| + \eta} < \epsilon$ 来修正相对收敛准则.

在今后使用各种迭代算法的过程中, 我们通常还需要给出一个标记不收敛的停止准则. 也就是说, 不论收敛与否，N 步迭代后停止运算(当然这里 N 通常来说就是指我们所能容许的最多的迭代次数). 超过这样一个值, 即便它能够收敛, 我们也需要将其标示出来, 需要分析其如此多的迭代次数仍为达到收敛的原因.

此外, 使用收敛准则的时候, 可考虑一个或多个收敛度量, 比如 $|x_{t+1} - x_t|$, $|x_{t+1} - x_t| / |x_t|$, 或 $|g(x_{t+1})|$. 如果每一个都不单减或若干次迭代后出现了周期，则迭代停止.

最后, 总结二分法的优缺点：算法简单, 而且收敛性总能得到保证. 其收敛速度很慢, 即它相对于后面讨论的其它方法而言, 为达到要求的精度, 它需要更多次的迭代; 但相对于其它方法具有明显的优势, 也就是如果 g 在区间 $[a_0, b_0]$ 上连续, 则不论 g' 是否存在或是否容易导出, 其根都可以由括入根法找到，因为它们不必考虑 g' . 故, 相对其它强烈依赖 g 的光滑性的方法, 二分法有其合理的一面; 另外, 如果 g 在初始区间内的根多于 1，则容易看到二分法将找到其中一个，而找不到其余的.

例 4.1.1. (**分位数的计算**) 设连续型随机变量 X 的分布函数为 $F(x)$, 对任给的 $\alpha \in (0,1)$, 我们求取 $F(x)$ 的 α 下分位点, 即求取 $g(x) = F(x) - \alpha = 0$ 的根. 我们需要找到一个有根区间来开始使用算法 4.1.1. 一个简单方便的做法是利用 $g(x)$ 的单调性任取一初始点 a. 比如通常我们可取 a 为均值. 如果 $g(a) < 0$, 那么我们寻找第一个使得 $g(a+k) > 0$ 的正整数 k. 在找到这样的 k 之后, 显然 $[a+k-1, a+k]$ 就是有根区间; 反之, 若 $g(a) > 0$, 我们寻找第一个使得 $g(a+k) < 0$ 的负整数 k, 而 $[a+k, a+k+1]$ 有根区间. 当然, 我们亦可将这个搜索的步长 1 改为其它任何有可能加快搜索到有根区间进程的步长, 比如, 有时可取该步长为 $F(x)$ 的标准差.