STATISTICA 1, metodi matematici e statistici

Introduzione al linguaggio R

Esercitazione4: 29-03-2004

Andrea Tancredi

Università di Roma “La Sapienza”, Rome, Italy

andrea.tancredi@uniroma1.it
http://3w.eco.uniroma1.it/utenti/tancredi

La v.a. Cauchy standard ha densità

$$f(x) = \frac{1}{\pi}\frac{1}{1+x^2}, \quad -\infty < x < \infty.$$

Consideriamo allora la v.a. $X = \mu + \sigma X$ la cui densità è

$$f_{\mu,\sigma}(x) = \frac{1}{\sigma} \cdot f\left(\frac{x-\mu}{\sigma}\right)$$

Chiameremo μ parametro di posizione e σ parametro di scala. Poichè X è simmetrica rispetto a μ possiamo considerare la mediana campionaria come stimatore di μ.

Inoltre, indicando con x_p il quantile di livello p della Cauchy(μ, σ) (ovvero il valore tale che $\int_{-\infty}^{x_p} f_{\mu,\sigma}(x)dx = p$) abbiamo che

$$x_{0.75} - x_{0.25} = 2\sigma$$

Infatti

```
> qcauchy(3/4, scale = 2) - qcauchy(1/4, scale = 2)
```

Per cui una buona stima di σ è lo scarto iterquantilico diviso per due

Consideriamo per ora solo μ incognito e osserviamo che in R possiamo scrivere la meno log-verosimiglianza in due modi diversi, il primo utilizzando `dcauchy`

```
> mlogl <- function(mu, x) {
+     sum(-dcauchy(x, location = mu, log = TRUE))
+ }
```

il secondo utilizzando direttamente l'espressione della densità

```
> mlogl2 <- function(mu, x) {
+     sum(log(1 + (x - mu)^2))
+ }
```

I due modi di scrivere la logverosimiglianza dovrebbero portare agli stessi risultatati. Consideriamo allora i seguenti dati simulati

```
> n <- 30
> set.seed(42)
> x <- rcauchy(n)
```

In questo caso quindi μ è 0, ma facciamo finta di non saperlo e ci calcoliamo la stima di max verosimiglianza

```
> mu.start <- median(x)
> out <- nlm(mlogl, mu.start, x = x)
> mu.hat <- out$estimate
> mu.hat

[1] -0.1816501
```

Vediamo quello che succede con l'altra funzione di log-verosimiglianza

```
> out2 <- nlm(mlogl2, mu.start, x = x)
> mu.hat <- out$estimate
> mu.hat

[1] -0.1816501
```

Per questi dati la stima di massima verosimiglianza pari a -0.182 è meglio della mediana, pari a -0.195 mentre può succedere che per altri campioni la mediana sia meglio. Per capire quale dei due stimatori funziona meglio simuleremo la distribuzione dei due stimatori.

```
> nsim <- 100
> mu <- 0
> mu.hat <- double(nsim)
> mu.tilde <- double(nsim)
> for (i in 1:nsim) {
+     xsim <- rcauchy(n, location = mu)
+     mu.start <- median(xsim)
+     out <- nlm(mlogl, mu.start, x = xsim)
+     mu.hat[i] <- out$estimate
+     mu.tilde[i] <- mu.start
+ }
> mean((mu.hat - mu)^2)

[1] 0.06203118

> mean((mu.tilde - mu)^2)

[1] 0.08242236
```

I due numeri che abbiamo riportato alla fine della simulazione sono gli errori quadratici medi dei due stimatori. Lo stimatore di massima verosimiglianza ha un errore quadratico medio più piccolo ed è quindi migliore della mediana campionaria.

Grafi ci di verosimiglianze con due parametri

Modello Weibull

Sia $y = (y_1, \ldots, y_n)$ un campione casuale semplice da una v.a. Weibull di parametri γ, λ, $Y \sim WE(\gamma, \lambda)$. La densità di Y è

$$p(y; \gamma, \lambda) = \gamma \lambda y^{\gamma - 1} e^{-\lambda y^{\gamma}}$$

per cui la log-verosimiglianza associata al campione y è

$$\ell(\gamma, \lambda) = n \log \lambda + n \log \gamma + \gamma \sum_{i=1}^{n} \log y_i - \lambda \sum_{i=1}^{n} y_i^{\gamma}$$

.

In R possiamo scrivere la log-verosimiglianza nel modo seguente

```
> weib.llik <- function(par, data) {
+     n <- length(data)
+     s1 <- sum(log(data))
+     s2 <- sum(data^par[1])
+     n * log(par[2]) + n * log(par[1]) + par[1] * s1
+     - par[2] * s2
+ }
```

Consideriamo un campione di numerosità 20 generato da una Weibull di parametri $(\gamma, \lambda) = (1, 1)$

```
> w.y <- rweibull(20, shape = 1, scale = 1)
```

Osserviamo, tramite `help(rweibull)` che la parametrizzazione che usa R è diversa dalla nostra.

Vediamo ora come ottenere le curve di livello della verosimiglianza.

Definiamo prima una griglia di valori su cui calcolare la log verosimiglianza attraverso il comando `expand.grid`

```
> gamma <- seq(0.01, 2.5, length = 100)
> lambda <- seq(0.01, 2, length = 100)
> griglia <- expand.grid(gamma, lambda)
```

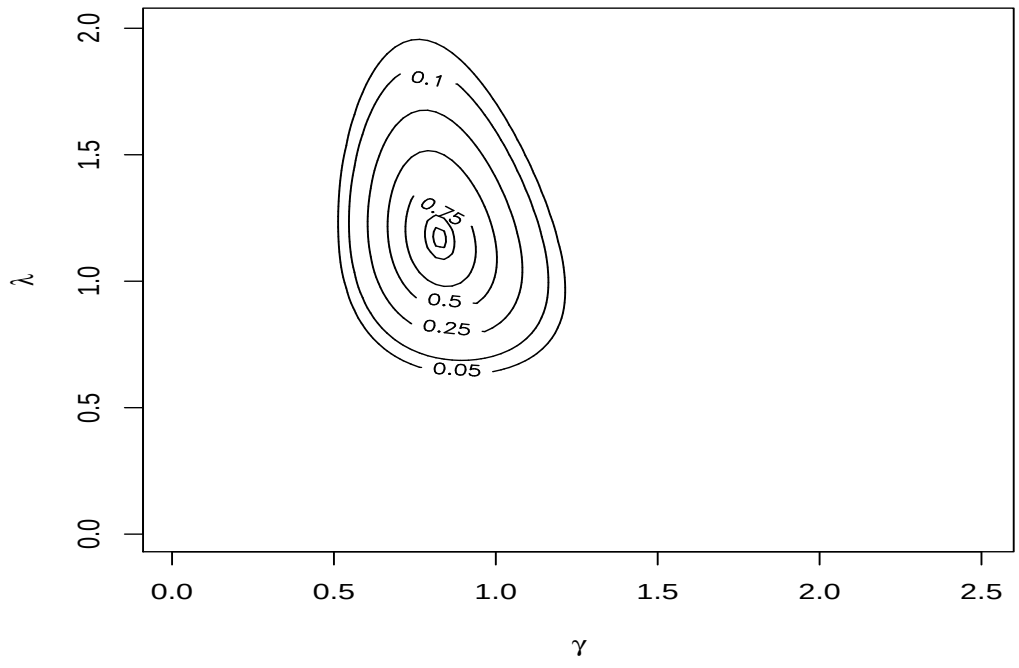
Le prime 100 righe della griglia saranno le coppie `gamma[i]`,`lambda[1]` per $i = 1, \ldots, 100$, le seconde 100 righe saranno le coppie `gamma[i]`,`lambda[2]` per $i = 1, \ldots, 100$ e così via...

Calcoliamo la log-verosimiglianza con i parametri dati dalle righe della matrice `griglia`

```
> llik.val <- apply(griglia, MAR = 1, FUN = weib.llik,
+     data = w.y)
```

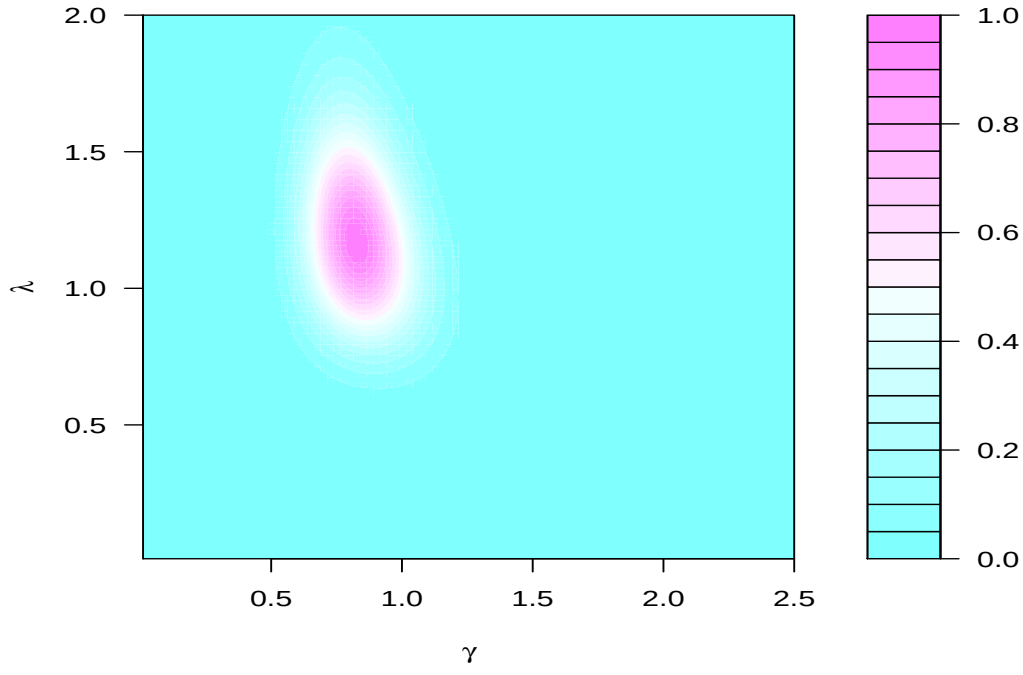
Per disegnare le curve di livello abbiamo bisogno del comando `contour` i cui input sono i punti dell'asse x (i nostri `gamma`) , i punti dell'asse y (i nostri `lambda`), e i valori della funzione di cui vogliamo le curve di livello. Tali valori devono essere dati in forma di matrice dove nel posto i, j abbiamo il valore della funzione nell'iesimo punto dell'asse y e nel jesimo punto dell'asse x.

```
> llik.val <- matrix(llik.val, nrow = length(gamma),
+     length(lambda), byrow = F)
> l.levels <- c(0.05, 0.1, 0.25, 0.5, 0.75, 0.95, 0.99)
> contour(gamma, lambda, exp(llik.val - max(llik.val)),
+     xlab = expression(gamma), ylab = expression(lambda),
+     levels = l.levels)
```

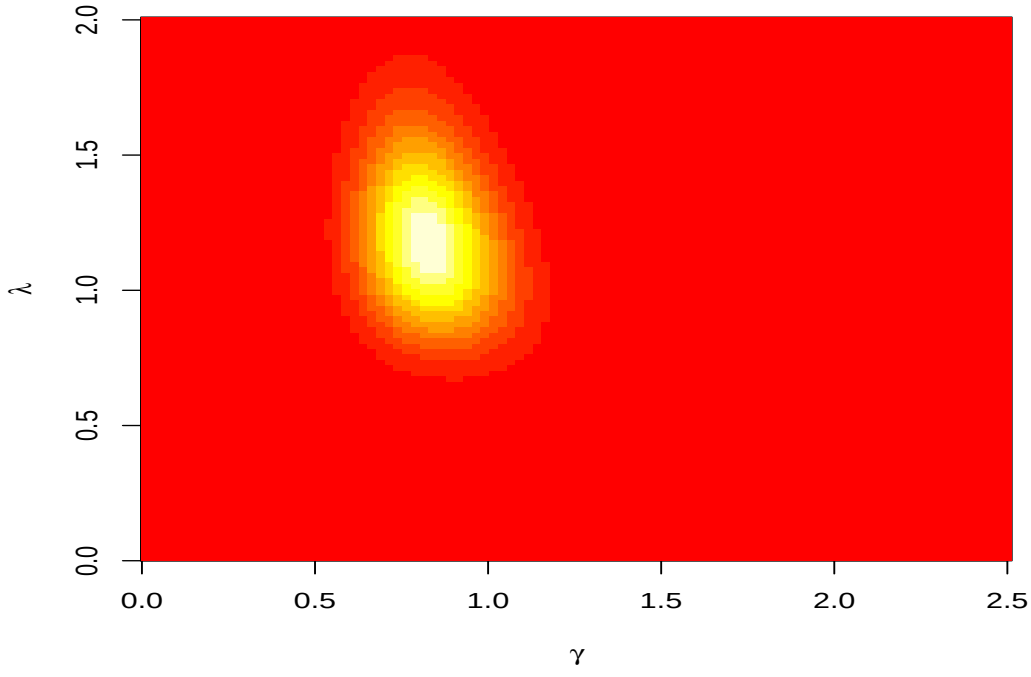

0.1
0.75
0.5
0.25
0.05
λ
γ
0.0
0.5
1.0
1.5
2.0
2.5

Possiamo produrre grafici più sofisticati attraverso i comandi `filled.contour` e `image`. Ad esempio

```
> filled.contour(gamma, lambda, exp(llik.val - max(llik.val
+     xlab = expression(gamma), ylab = expression(lambda))
```


2.0
1.5
1.0
0.5
λ
0.5
1.0
1.5
2.0
2.5
γ
1.0
0.8
0.6
0.4
0.2
0.0

```
> image(gamma, lambda, exp(llik.val - max(llik.val)),
+ xlab = expression(gamma), ylab = expression(lambda))
```


λ
0.0
0.5
1.0
1.5
2.0
γ
0.0
0.5
1.0
1.5
2.0
2.5