Statistica Computazionale Progredito
esame del 3 febbraio 2020

Istruzioni: lo studente deve produrre un file in formato pdf in cui riporta il nome, cognome, numero di matricola e, per ogni esercizio a cui risponde, il codice R utilizzato per produrre il risultato, il risultato (valori, grafici,...), i commenti e quant'altro richiesto dall'esercizio.

Soluzione: **NOTA BENE:** Si danno qui solo i risultati relativi al codice e all'output (è una soluzione, non è detto che sia l'unica via possibile per arrivare allo stesso risultato); sono omessi i commenti che naturalmente lo studente DEVE includere nel compito al momento dell'esame.

1. I dati nell'oggetto `Clotting` si riferiscono ad un esperimento sul tempo di coagulazione del sangue. In particolare, è stato misurato il tempo di coagulazione del plasma sanguigno (in secondi) (`tempo`) in 18 campioni di plasma normale diluito con plasma privo di protrombina in modo da ottenere 9 diverse concentrazioni percentuali (`u`). La coagulazione è stata indotta con due diversi lotti di tromboplastina (`lotto`).

   (a) Stimare un modello di regressione Gamma con link canonico (`inverse`) in cui `tempo` è la variabile risposta e `u` e `lotto` sono le variabili esplicative. Commentare il modello stimato e valutare l'eventuale inserimento di un termine di interazione tra le variabili esplicative.

```
model1 <- glm(tempo ~ lotto + u, family = Gamma(inverse), data=Clotting)
summary(model1)

##
## Call:
## glm(formula = tempo ~ lotto + u, family = Gamma(inverse), data = Clotting)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -0.297  -0.212  -0.106   0.132   0.523
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.01964    0.00413    4.76  0.00025 ***
## lottouno    -0.01247    0.00411   -3.03  0.00836 **
## u            0.00076    0.00011    6.93  4.8e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Gamma family taken to be 0.081425)
##
##     Null deviance: 7.7087  on 17  degrees of freedom
## Residual deviance: 1.0884  on 15  degrees of freedom
## AIC: 126.4
##
## Number of Fisher Scoring iterations: 5

model1bis <- glm(tempo ~ lotto * u, family = Gamma(inverse), data=Clotting)
summary(model1bis)
```

```
##
## Call:
## glm(formula = tempo ~ lotto * u, family = Gamma(inverse), data = Clotting)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -0.285  -0.219  -0.115   0.148   0.377
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.015771   0.004482    3.52  0.00341 **
## lottouno    -0.007085   0.005226   -1.36  0.19667
## u            0.000978   0.000189    5.17  0.00014 ***
## lottouno:u  -0.000320   0.000224   -1.43  0.17575
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Gamma family taken to be 0.071594)
##
##     Null deviance: 7.70867  on 17  degrees of freedom
## Residual deviance: 0.93818  on 14  degrees of freedom
## AIC: 125.7
##
## Number of Fisher Scoring iterations: 4
```

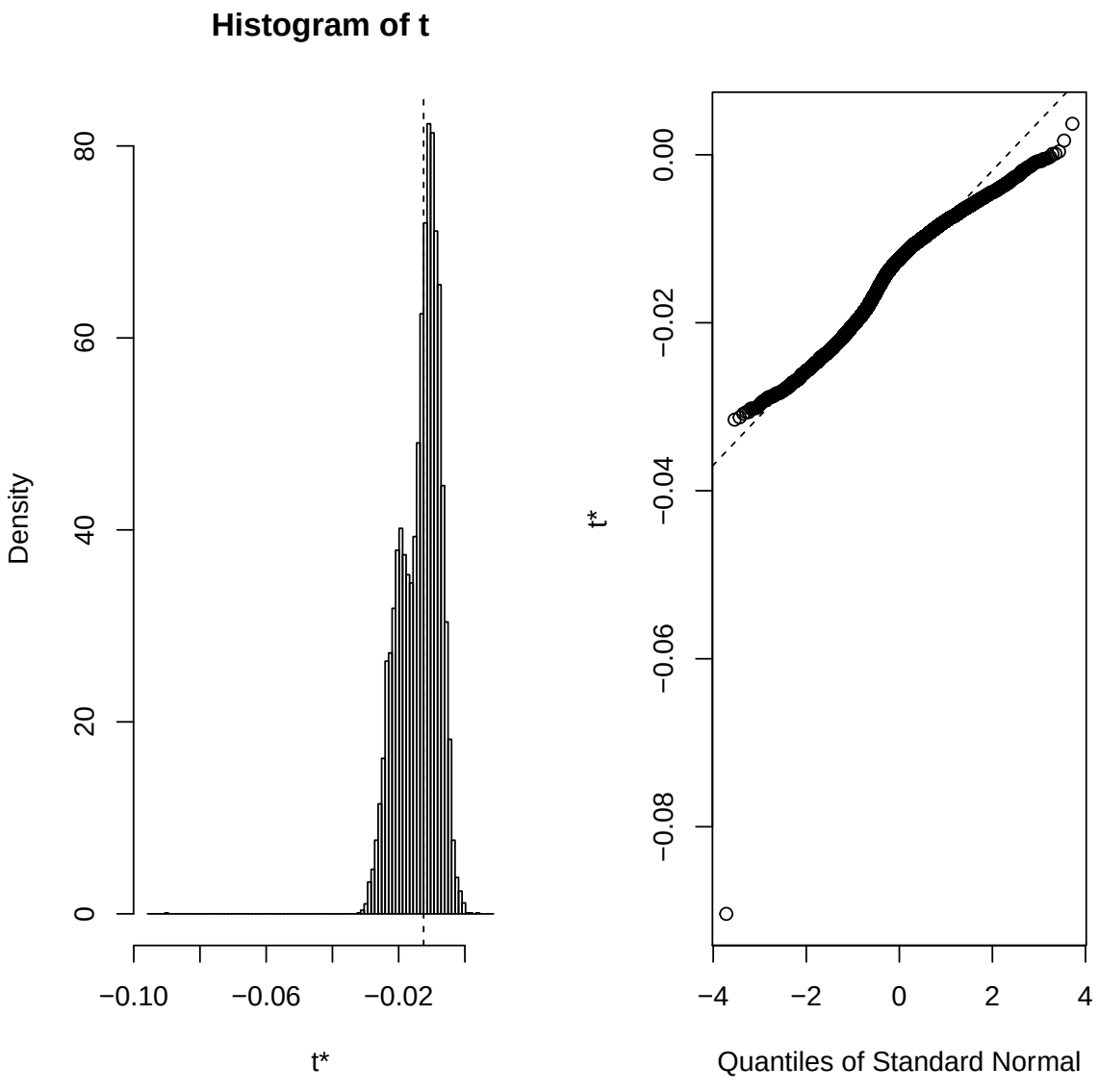
```
anova(model1,model1bis,test="F")
```

```
## Analysis of Deviance Table
##
## Model 1: tempo ~ lotto + u
## Model 2: tempo ~ lotto * u
##   Resid. Df Resid. Dev Df Deviance   F Pr(>F)
## 1        15      1.088
## 2        14      0.938  1     0.15 2.1   0.17
```

(b) Effettuare un bootstrap delle unità statistiche e commentare i risultati ottenuti, in particolare con riferimento alla variabile `lotto` di cui si vuole calcolare anche un intervallo di confidenza di livello 0.95. Si commenti anche la scelta del tipo di bootstrap rispetto alla natura dei dati.

```
library(boot)
boot.obs <- function(data, indices) {
  datai <- data[indices,]
  mod <- glm(tempo~ lotto + u, data=datai, family=Gamma(inverse))
  coefficients(mod)
}

clotting.boot <- boot(Clotting, boot.obs, R=10000)
# or
# clotting.boot <- boot(Clotting, boot.obs, R=10000, strata = Clotting$lotto)
```

```
clotting.boot

##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = Clotting, statistic = boot.obs, R = 10000)
##
##
## Bootstrap Statistics :
##        original      bias    std. error
## t1*  0.01963927  2.3296e-03  0.00697518
## t2* -0.01247172 -1.1060e-03  0.00584548
## t3*  0.00076047 -2.1899e-05  0.00013422

plot(clotting.boot, index=2)
```



```r
boot.ci(clotting.boot, index=2)
```

```
## Warning in boot.ci(clotting.boot, index = 2):  bootstrap variances needed
for studentized intervals
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = clotting.boot, index = 2)
##
## Intervals :
## Level      Normal              Basic
## 95%   (-0.0228,  0.0001 )   (-0.0204,  0.0006 )
##
## Level     Percentile            BCa
## 95%   (-0.0256, -0.0045 )   (-0.0240, -0.0029 )
## Calculations and Intervals on Original Scale
```

(c) Il modello stimato nei punti precedenti assume una funzione legame inversa tra la media del tempo di coagulazione e il predittore lineare. Si vuole valutare una funzione di legame alternativa, in particolare si assume una funzione legame logaritmica (`log`). Per confrontare i due modelli si può utilizzare come test la differenza delle log verosimiglianze stimate. Dare il valore del test osservato nel campione. (Suggerimento: la funzione `logLik` potrebbe essere utile...).

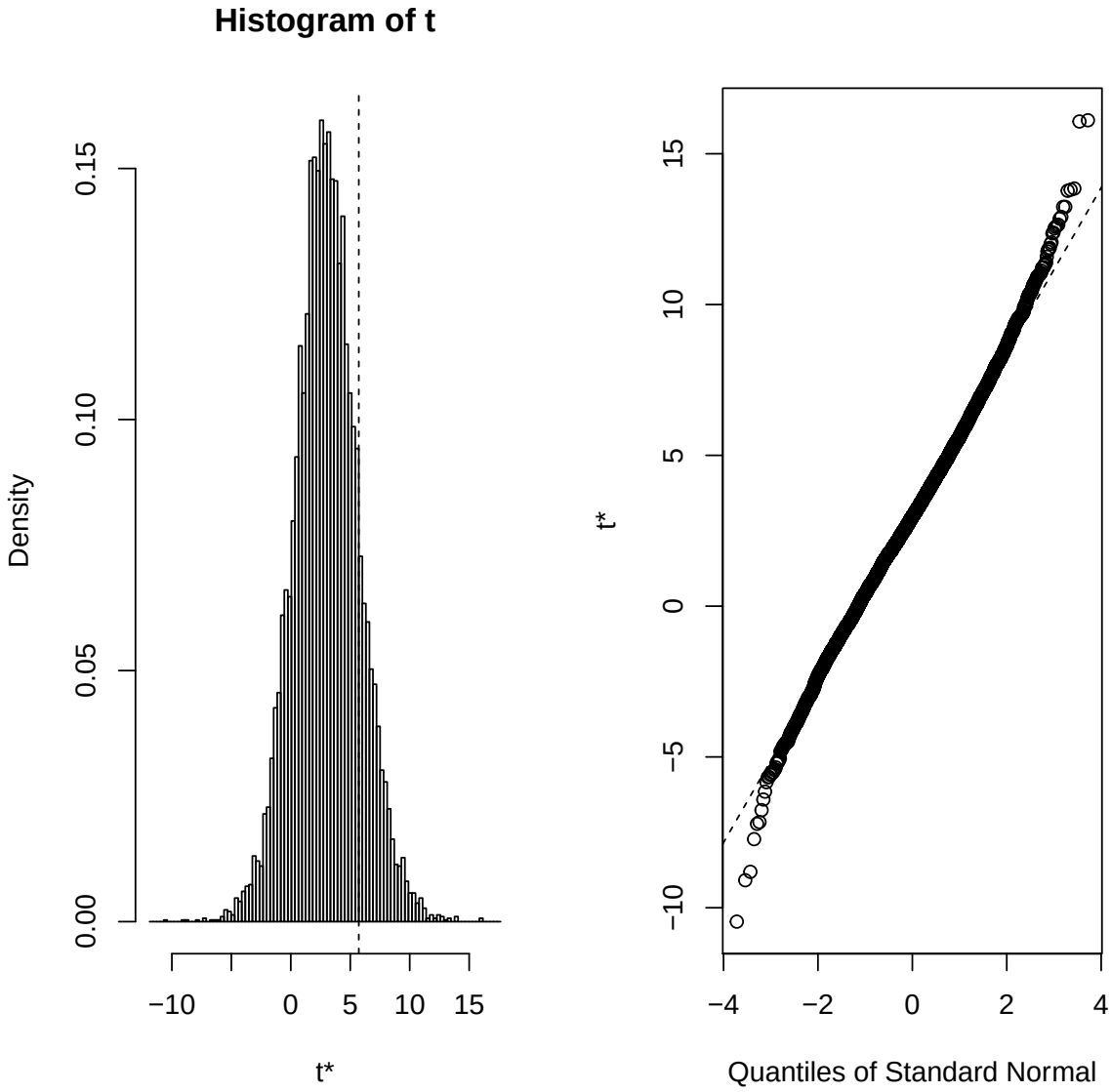
```r
model2 <- glm(tempo ~ lotto + u, family = Gamma(log), data=Clotting)
summary(model2)
```

```
##
## Call:
## glm(formula = tempo ~ lotto + u, family = Gamma(log), data = Clotting)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.3936  -0.3006  -0.2005   0.0295   0.7466
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.70916    0.18300   20.27  2.6e-12 ***
## lottouno     0.47525    0.19218    2.47  0.02585 *
## u           -0.01558    0.00306   -5.08  0.00013 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Gamma family taken to be 0.16619)
##
##     Null deviance: 7.7087  on 17  degrees of freedom
## Residual deviance: 2.0364  on 15  degrees of freedom
## AIC: 137.8
```

```
##
## Number of Fisher Scoring iterations: 7

logLik(model1)-logLik(model2)

## 'log Lik.' 5.7172 (df=4)
```

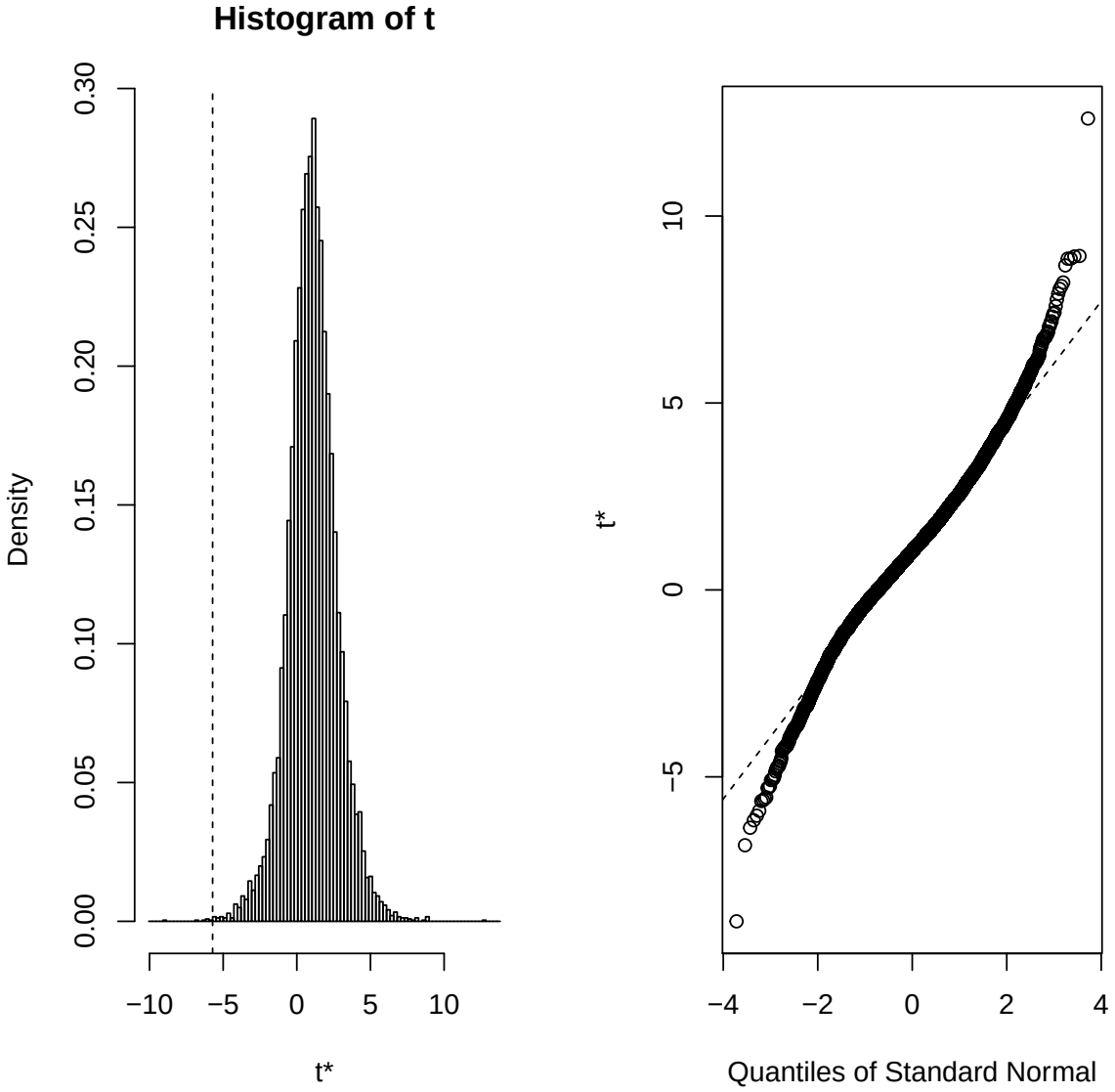
(d) Trovare la distribuzione nulla stimata simulata del test al punto precedente attraverso un bootstrap parametrico, quando si vuole verificare l'ipotesi nulla che il modello abbia legame inverso contro l'alternativa che il legame sia logartmico. Dare una stima del relativo p-value. (Suggerimento: se `modello` è il modello stimato, `modello$fitted.values` restituisce le medie stimate della variabile risposta, $\hat{\mu}_i$, e `summary(modello)$dispersion` è la stima del parametro di dispersione, $\tilde{\phi}$; inoltre $\mathrm{E}(Y_i) = \mu_i$ e $\mathrm{Var}(Y_i) = \phi\mu_i^2$)

```
test1 <- function(data)
{
        mod1 <- glm(tempo ~ lotto + u,
                    family = Gamma(inverse), data=data)
        mod2 <- glm(tempo ~ lotto + u,
                    family = Gamma(log), data=data)
# deviance(mod1)-deviance(mod2)
        logLik(mod1)-logLik(mod2)
}

# simulare dal modello stimato
# Y ~ Gamma(shape=1/dispersion,rate=1/(dispersion*mu))

null.gen <- function(data,est)
{
  out <- data
        out$tempo <- rgamma(length(est)-1,shape = 1/est[1],
                            scale = est[1]*est[-1])
        out
}

est1 <- c(summary(model1)$dispersion,fitted(model1))

lvc1 <- boot(Clotting, statistic = test1, sim = 'parametric',
             ran.gen = null.gen, mle = est1, R = 10000)
mean(lvc1$t <= lvc1$t0)

## [1] 0.8501

plot(lvc1)
```



(e) Ripetere il punto precedente invertendo il ruolo delle due ipotesi.

```
test2 <- function(data)
{
        mod1 <- glm(tempo ~ lotto + u, family = Gamma(log), data=data)
        mod2 <- glm(tempo ~ lotto + u, family = Gamma(inverse), data=data)
# deviance(mod1)-deviance(mod2)
        logLik(mod1)-logLik(mod2)
}

est2 <- c(summary(model2)$dispersion,fitted(model2))
cvl2 <- boot(Clotting, statistic = test2, sim = 'parametric',
             ran.gen = null.gen, mle = est2, R = 10000)
mean(cvl2$t <= cvl2$t0)

## [1] 6e-04

plot(cvl2)
```



(f) Alla luce dei p-value calcolati ai due punti precedenti, quale funzione legame sembra preferibile?

2. L'oggetto `Venicemax` contiene dati relativi ai livelli massimi annui della marea registrati a Venezia tra il 1875 e il 2019. In particolare, la variabile `valore` indica il valore massimo della marea (in cm) e la variabile `anno` il corrispondente anno. Si assume che i massimi annuali siano un campione casuale semplice da una distribuzione Gumbel con funzione di ripartizione

$$F_Y(y; \mu, \sigma) = \exp[-\exp\{-(y-\mu)/\sigma\}],$$

con $\mu \in \mathbb{R}$, $\sigma > 0$ e $y \in \mathbb{R}$.

(a) Mostrare che la log verosimiglianza per $\theta = (\mu, \sigma)$ è pari a

$$\ell(\theta) = \begin{cases} -n \log \sigma - \sum_{i=1}^n \frac{y_i - \mu}{\sigma} - \sum_{i=1}^n \exp\left\{-\frac{y_i-\mu}{\sigma}\right\}, & \text{se } \sigma > 0, \\ -\infty, & \text{altrimenti.} \end{cases}$$

Scrivere una funzione in R che calcola $\ell(\theta)$.

```
nlogL <- function(param,data)
{
  term <- (data-param[1])/param[2]
        if (param[2] < 0) return(Inf)
        else return(length(data) * log(param[2]) +
                      sum(term) + sum(exp(-term)))
}
```

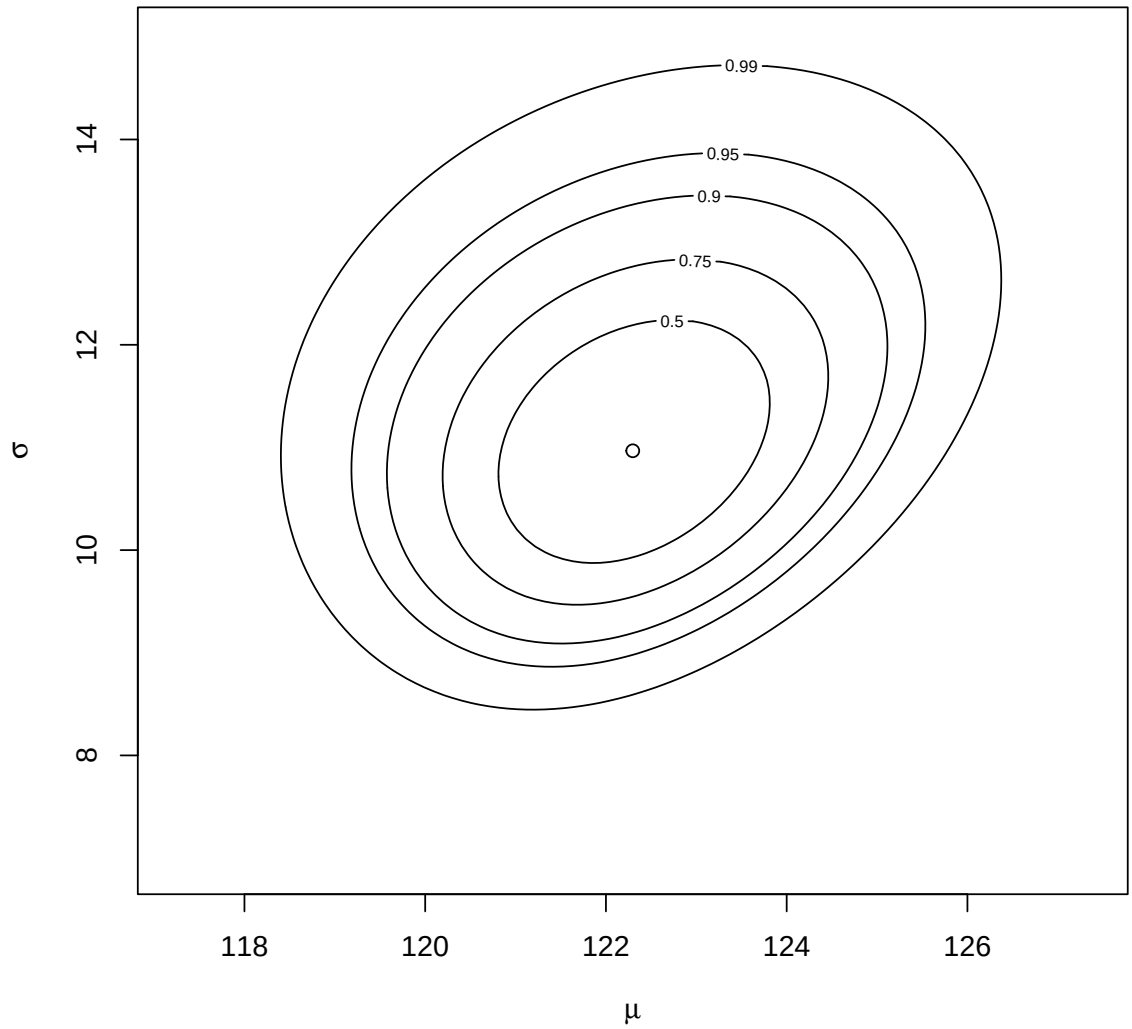
(b) Trovare numericamente la stima di massima verosimiglianza di θ e dare una valutazione numerica dello standard error di ciascuna componente della stima.

```
venice.mle <- nlminb(c(0, 1), nlogL, data = Venicemax$valore,
                lower = c(-Inf, 1e-10), upper = rep(Inf, 2))


jhat <- optimHess(fn = nlogL, par = venice.mle$par,
                  data = Venicemax$valore)

vcov <- solve(jhat)
venice.mle.se <- sqrt(diag(vcov))

venice.mle$par

## [1] 122.297  10.968

venice.mle.se

## [1] 1.26782 0.99991
```

(c) Fare un grafico della log verosimiglianza in un'opportuna regione dello spazio parametrico.

```
mu.val <- seq(venice.mle$par[1] - 4 * venice.mle.se[1],
              venice.mle$par[1] + 4 * venice.mle.se[1], length = 100)
sig.val <- seq(venice.mle$par[2] - 4 * venice.mle.se[2],
              venice.mle$par[2] + 4 * venice.mle.se[2], length = 100)
```

```
parvalues <- expand.grid(mu.val, sig.val)
llikvalues <- apply(parvalues, 1, nlogL, data=Venicemax$valore)
llikvalues <- matrix(-llikvalues, nrow = length(mu.val),
                     ncol = length(sig.val), byrow = FALSE)
conf.levels<-c(0.5,0.75,0.9,0.95,0.99)
contour(mu.val, sig.val, llikvalues - max(llikvalues),
        xlab = expression(mu), ylab = expression(sigma),
        levels = -qchisq(conf.levels,2)/2,
        labels = as.character(conf.levels))
points(venice.mle$par[1],venice.mle$par[2])
```



(d) Indicato con y_p il quantile di Y che lascia probabilità p a destra, dimostrare che è pari a

$$y_p = \mu - \sigma \log\{-\log(1-p)\}\,.$$

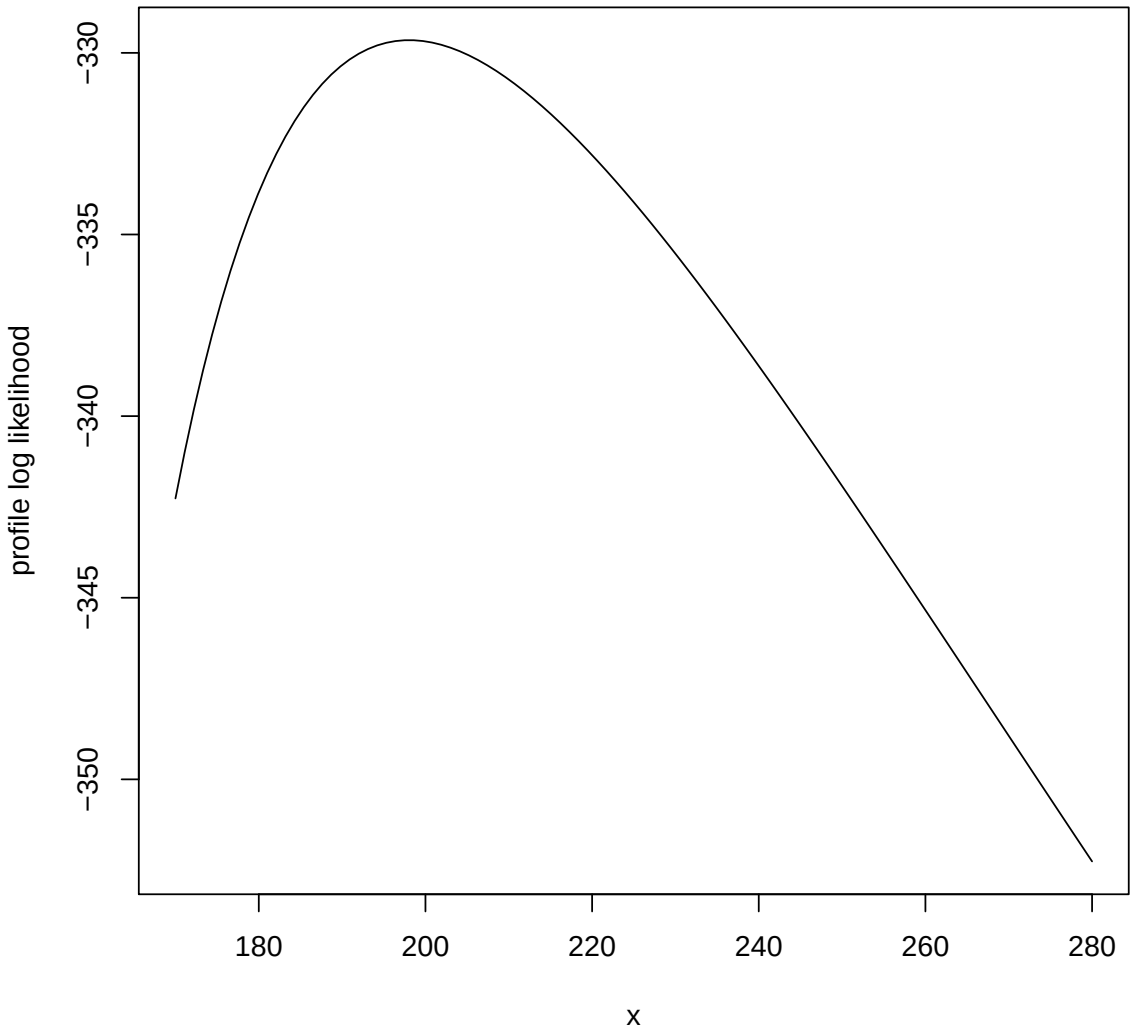
Trovare la stima di massima verosimiglianza di y_p, quando $p = 0.001$.

```r
# return level
ret.level <- function(param,p)
  param[1] - param[2] * log(-log(1 - p))


p <- 0.001
yp.mle <- ret.level(venice.mle$par,p)
yp.mle
```

```
## [1] 198.05
```

(e) Scrivere una funzione che calcola la log verosimiglianza profilo per $y_{0.001}$ e farne un grafico in un intervallo opportuno. Trovare anche un intervallo di confidenza di livello 0.95 per $y_{0.001}$ basato sul log rapporto di verosimiglianza.

```r
nlogL2 <-  function(param2,data,p)
{
        yp <- param2[1]
        sigma <- param2[2]
        mu <- yp + sigma * log(-log(1 - p))
        nlogL(c(mu, sigma), data)
}

nlogL2.P <- function(yp,data,p)
  nlminb(venice.mle$par[2],
         function(x) nlogL2(c(yp, x), data = data, p = p),
         lower = 1e-10, upper = Inf)$objective

nlogL2.Pv=Vectorize(nlogL2.P,"yp")

plot(function(x) -nlogL2.Pv(x,Venicemax$valore,0.001),
     170, 280, ylab="profile log likelihood")
```




```
# # oppure
#
# psi.fun <- function(param,p)
#   param[1]-param[2] * log(-log(1 - p))
#
# psi.fun(venice.mle£par,0.001)
#
# library(Rsolnp)
#
# nlogL.prof <- function(psi,data,p,init=NULL) {
#   if (is.null(init)) init <- c(100,100)
#   out <- solnp(init,fun = nlogL, eqfun = function(param,data)
#     psi.fun(param,p=p), eqB = psi, control=list(trace=0),data=data)
#   out£values[length(out£values)]
#   }
#
# nlogL.profv <- Vectorize(nlogL.prof,"psi")
```

```r
# plot(function(x) -nlogL.profv(x,Venicemax$valore,0.001,
#       init=venice.mle$par),
#       170, 280, ylab="profile log likelihood")


# deviance CI

lrt.ci <- uniroot(function(x)
  -nlogL2.Pv(x, data = Venicemax$valore, p) +
    venice.mle$objective + qchisq(0.95, 1) / 2,
  c(yp.mle - 100, yp.mle))$root
lrt.ci <- c(lrt.ci, uniroot(function(x)
  -nlogL2.Pv(x, data=Venicemax$valore, p) +
    venice.mle$objective + qchisq(0.95, 1) / 2,
    c(yp.mle, yp.mle + 100))$root)
lrt.ci

## [1] 185.17 214.50
```

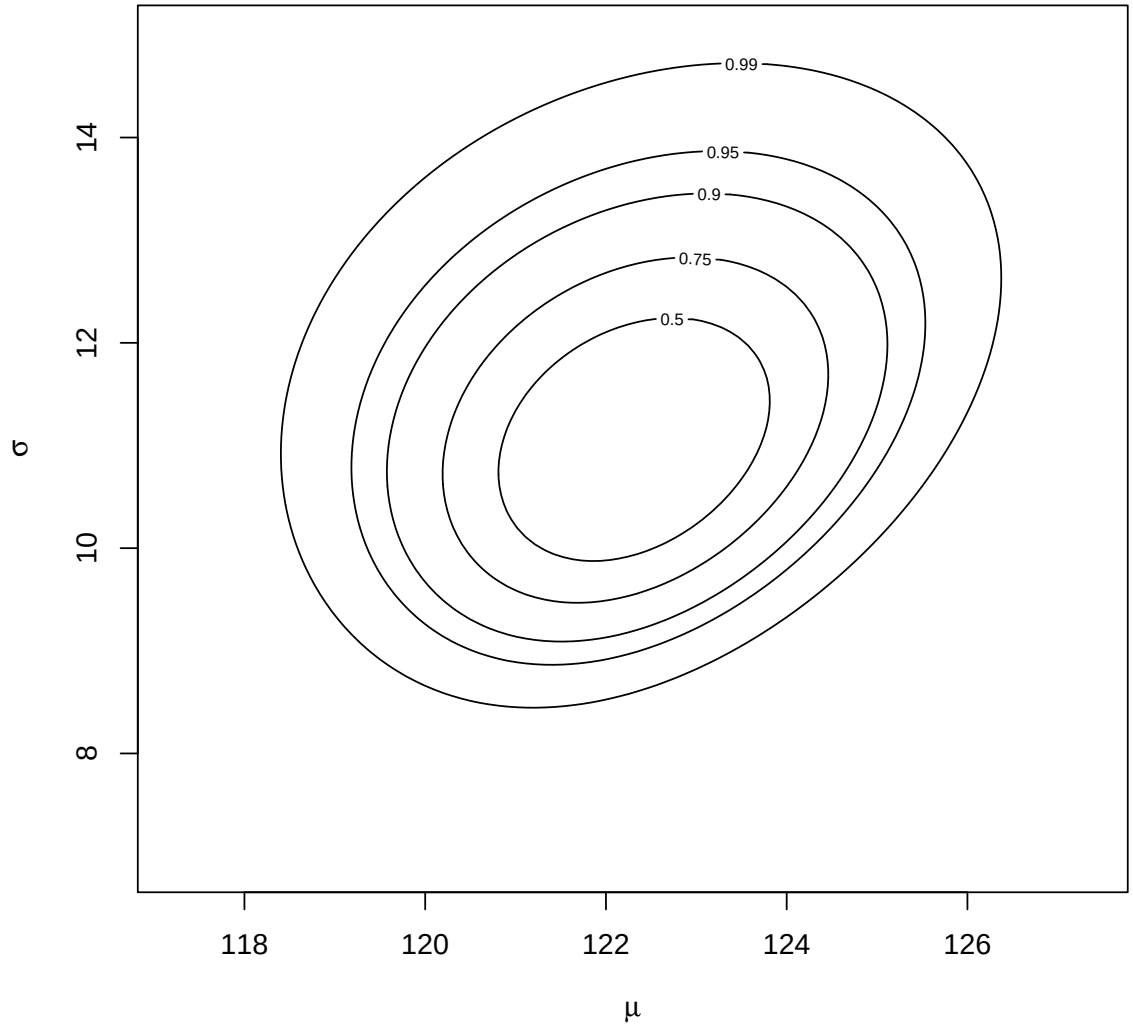
(f) Si assumono per μ e σ distribuzioni a priori indipendenti con

$$\mu \sim \mathrm{U}(a_1, b_1), \quad \sigma \sim \mathrm{Gamma}(a_2, b_2)\,,$$

con $a_1 = 80$, $b_1 = 250$, $a_2 = a_2 = 0.01$. Dare un'interpretazione alla scelta della distribuzione a priori. Sembra una distribuzione particolarmente non informativa?

(g) Scrivere una funzione che calcola la distribuzione a posteriori per θ ed implementare un algoritmo Metropolis-Hastings per simulare da tale distribuzione. Scegliere i parametri dell'algoritmo in modo da ottenere una convergenza soddisfacente. Controllare quest'ultima sia con indicatori numerici che grafici, possibilmente usando più catene e commentando.

```r
# log priori
lprior <- function(param,hyppar)
  dunif(param[1],hyppar[1],hyppar[2],log=TRUE) +
  dgamma(param[2],hyppar[3],hyppar[4])

lposterior <- function(param,data,hyppar)
  -nlogL(param,data) + lprior(param,hyppar)

hyppar0 <- c(80,250,0.01,0.01)

lpostvalues <- apply(parvalues, 1, lposterior,data=Venicemax$valore,
                     hyppar = hyppar0)
lpostvalues <- matrix(lpostvalues, nrow = length(mu.val),
                      ncol = length(sig.val), byrow = F)
conf.levels<-c(0.5,0.75,0.9,0.95,0.99)
contour(mu.val,sig.val,lpostvalues-max(lpostvalues),
xlab=expression(mu),ylab=expression(sigma),
levels=-qchisq(conf.levels,2)/2,labels=as.character(conf.levels))
```




```
# simulazione dalla posteriori con RW normale su ogni componente
gumbel.mh <- function(R,data,hyppar,eps,par0)
{ # Metropolis-Hasting con proposta passeggiata casuale uniforme
  d <- length(par0)
  out <- array(0,dim = c(R,d))
  accepted <- numeric(d)
  par <-  par0
  for (i in 1:R)
  {

        for (j in 1:d)
        {
                pars <- par
                pars[j] <- par[j] + runif(1, -eps[j], eps[j])
                alpha <- min(1,exp(lposterior(pars,data,hyppar) -
                                   lposterior(par,data,hyppar)))
                if (runif(1) < alpha)
```

```
                {
                        accepted[j] <- accepted[j]+1
                        par[j] <- pars[j]
                }
        }
        out[i,] <- par
  }
  list(values = out, accepted = accepted / R)
}


R <- 10000


library(mvtnorm)
library(coda)
library(plyr)
# 5 punti iniziali (potenzialmente) dispersi
R <- 10^4
start <- rmvnorm(5,mean = venice.mle$par,sigma=3*vcov)
start
```

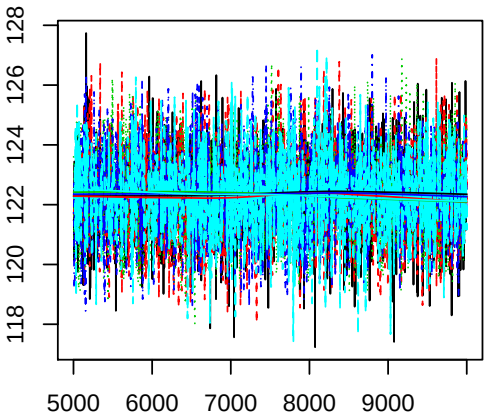
```
##        [,1]    [,2]
## [1,] 119.98  9.9271
## [2,] 121.47 10.8932
## [3,] 122.93 10.4046
## [4,] 122.86 13.4851
## [5,] 122.12  8.2003
```

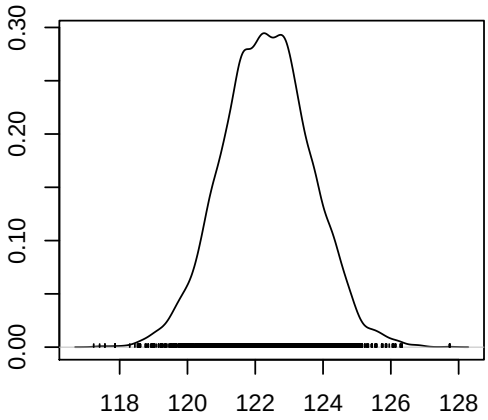
```
res <- list()
for (j in 1:nrow(start))
  res[[j]] <- gumbel.mh(R, Venicemax$valore,
                        hyppar0, c(8, 6), start[j,])$values

res1 <- llply(res,function(x) mcmc(window(x,start=5001),start=5001))
res1 <- mcmc.list(res1)
summary(res1)
```

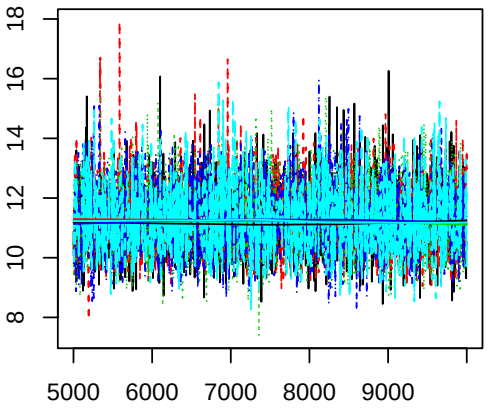
```
##
## Iterations = 5001:10000
## Thinning interval = 1
## Number of chains = 5
## Sample size per chain = 5000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##       Mean   SD Naive SE Time-series SE
## [1,] 122.3 1.31  0.00829         0.0206
## [2,]  11.3 1.05  0.00666         0.0168
```

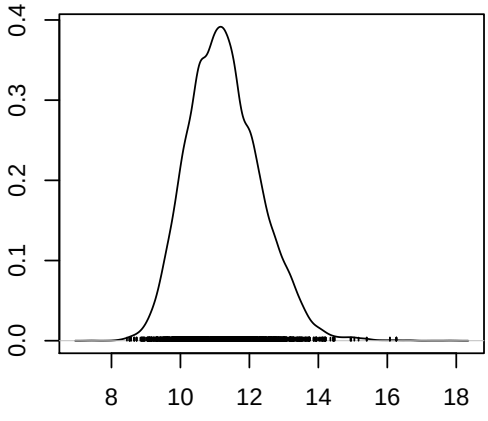
```
##
## 2. Quantiles for each variable:
##
##        2.5%   25%   50%   75% 97.5%
## [1,] 119.77 121.4 122.3 123.2 124.9
## [2,]   9.44  10.5  11.2  11.9  13.5
```

```
plot(res1)
```

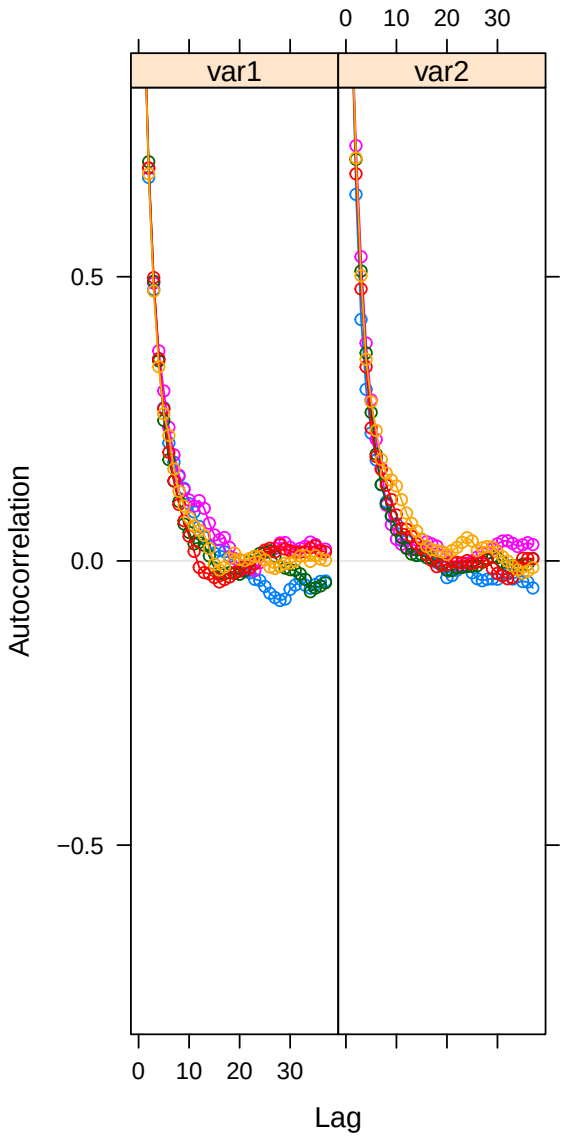




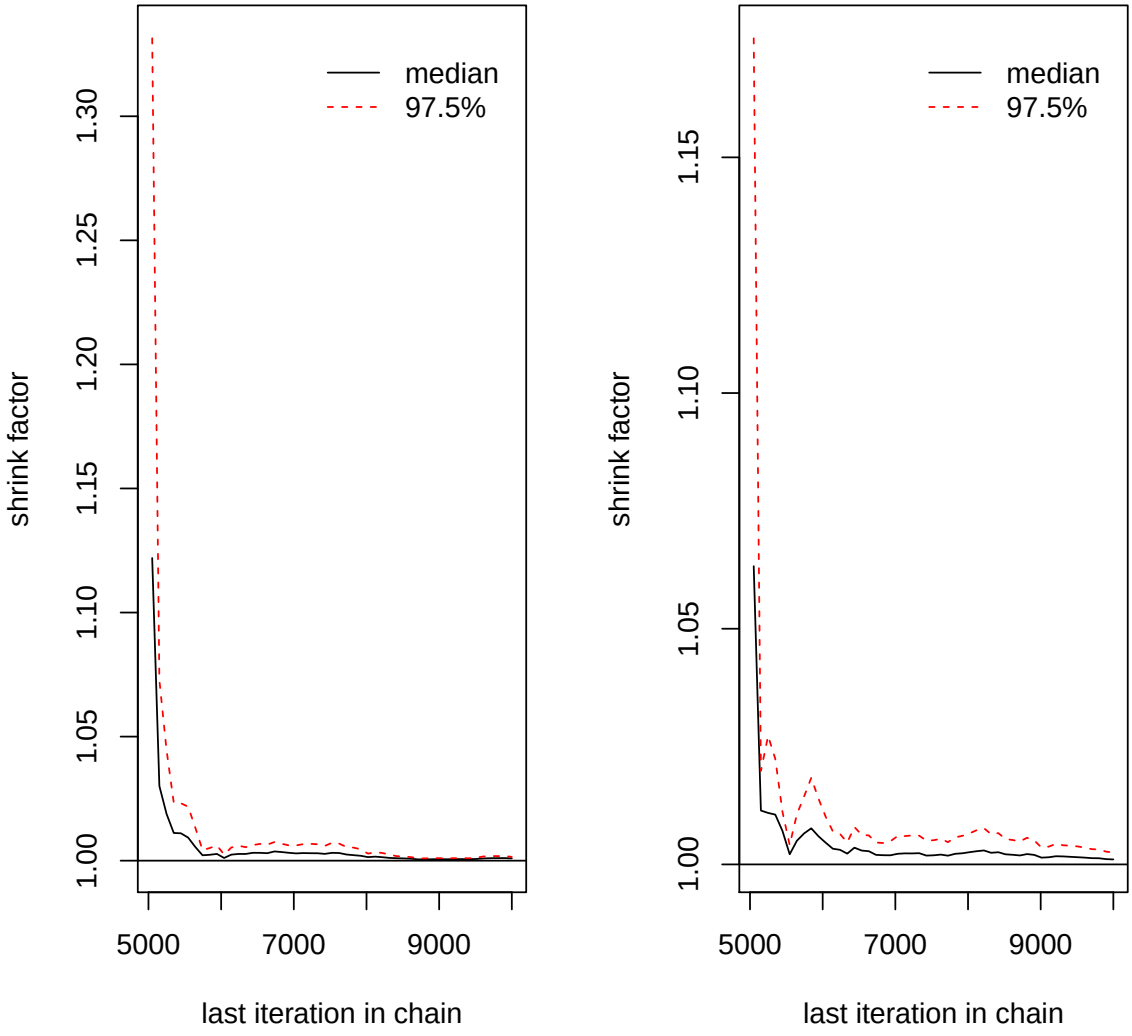





```
acfplot(res1)
```



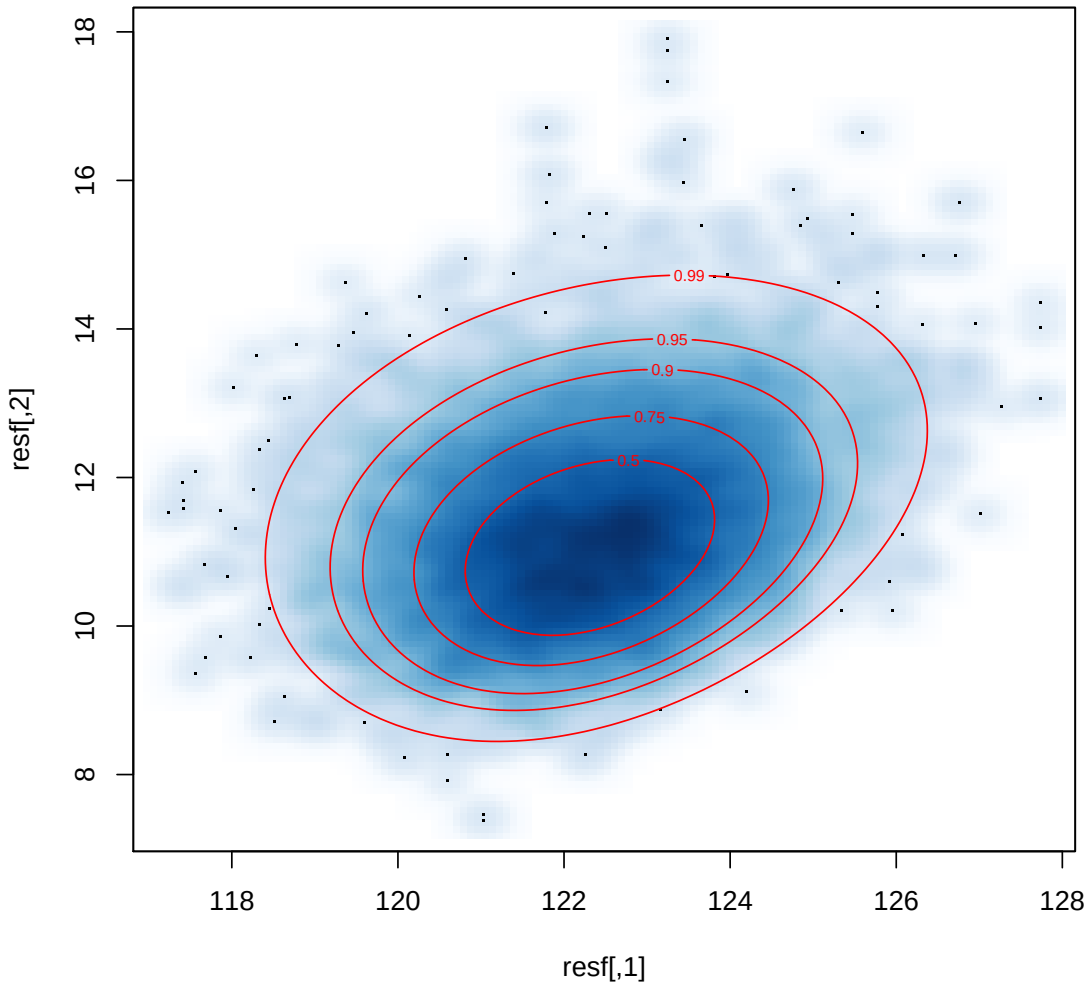

```
effectiveSize(res1)

##   var1   var2
## 4084.9 3982.0

gelman.diag(res1)

## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1          1
## [2,]          1          1
##
## Multivariate psrf
##
## 1

gelman.plot(res1)
```



(h) Utilizzare i valori simulati al punto precedente per dare una

i. rappresentazione grafica della distribuzione a posteriori di θ,

```
resf <- NULL
for (i in 1:5)
 resf <- rbind(resf,res1[[i]])
par(mfrow=c(1,1))
smoothScatter(resf)
contour(mu.val,sig.val,lpostvalues-max(lpostvalues),
xlab=expression(mu),ylab=expression(sigma),
levels=-qchisq(conf.levels,2)/2,
labels=as.character(conf.levels),col=2,add=TRUE)
```



ii. stima delle medie e degli standard error a posteriori di μ e σ,

```
mean(resf[,1])
## [1] 122.33
mean(resf[,2])
## [1] 11.266
sd(resf[,1])
## [1] 1.31
sd(resf[,2])
## [1] 1.053
```

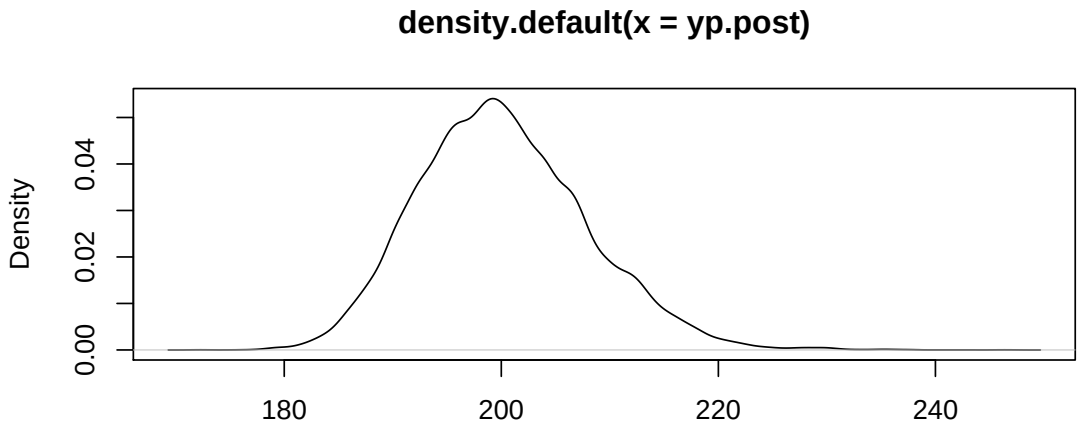
iii. stima di intervalli di credibilità di probabilità 0.95 per μ e σ,

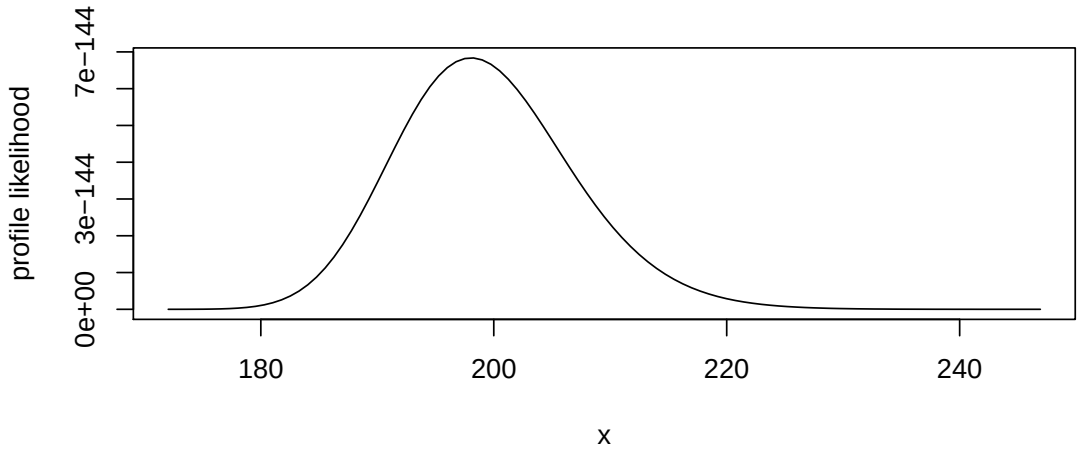
```
library(TeachingDemos)
emp.hpd(resf[,1])
## [1] 119.81 124.88
emp.hpd(resf[,2])
## [1]  9.3741 13.3544
```

```
quantile(resf[,1],c(0.025,0.975))
##   2.5%  97.5%
## 119.77 124.86
quantile(resf[,2],c(0.025,0.975))
##    2.5%   97.5%
##  9.4415 13.4735
```

iv. stima della media, di un intervallo di credibilità e una rappresentazione grafica della distribuzione a posteriori di $y_{0.001}$. Fare un confronto con i risultati frequentisti ottenuti in precedenza e commentare.

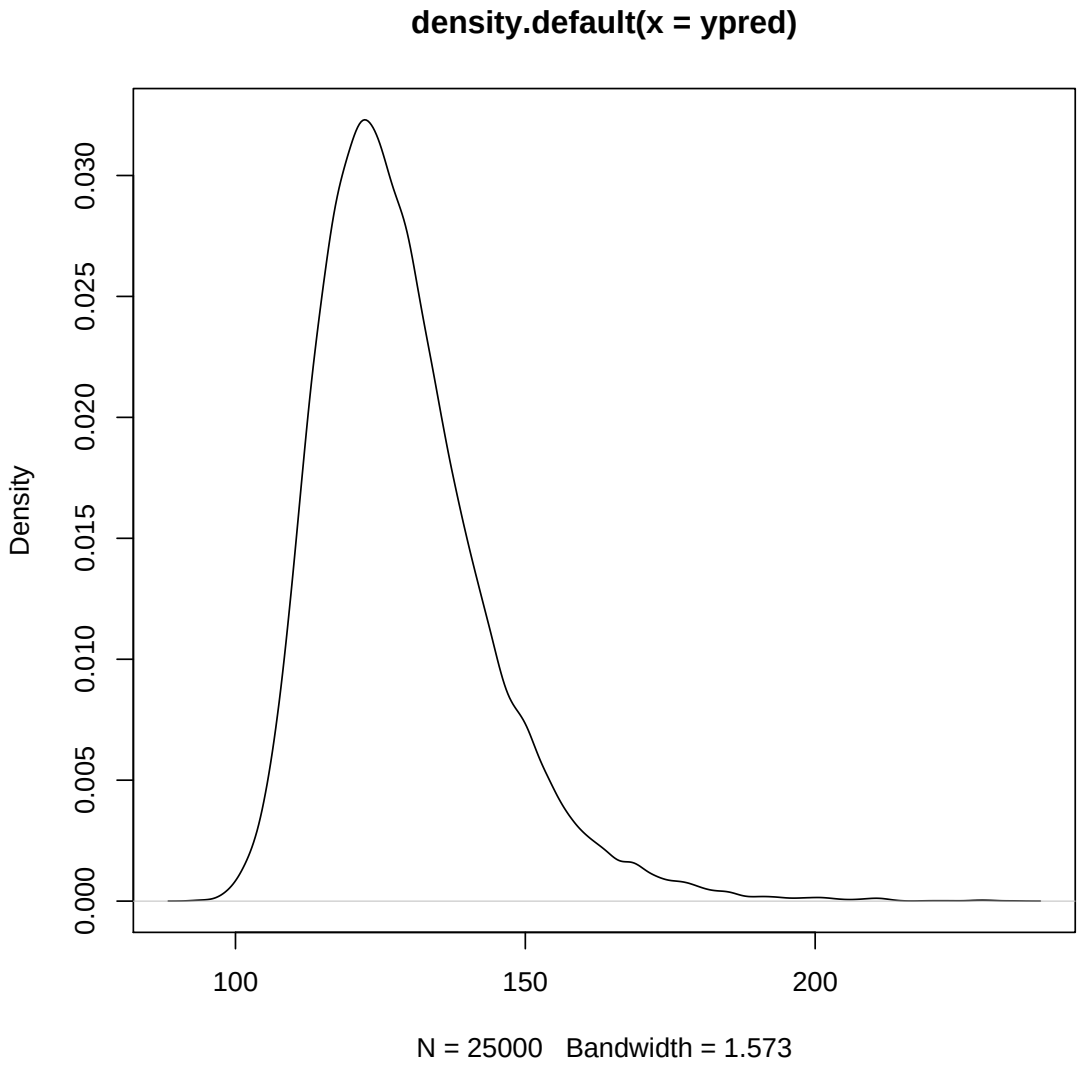
```
yp.post <- apply(resf,1,ret.level,p=p)

mean(yp.post)
## [1] 200.15
emp.hpd(yp.post)
## [1] 185.68 215.29
par(mfrow=c(2,1))
#hist(xp.post,nclass=50)
plot(density(yp.post))
plot(function(x) exp(-nlogL2.Pv(x,Venicemax$valore,0.001)),
     min(yp.post),
max(yp.post),ylab="profile likelihood")
```





v. Dare un intervallo di previsione *equi-tailed* di probabilità 0.95 per il valore massimo della marea nel 2020. (Suggerimento: è facile simulare da Y tramite inversione),

```
rgumbel <- function(n,par)
  par[1] - par[2] * log(-log(runif(n)))

ypred <- apply(resf,1, function(x) rgumbel(1,par=x))
plot(density(ypred))
```




```
#emp.hpd(ypred)
quantile(ypred,c(0.025,0.975))
##   2.5%  97.5%
## 107.44 164.32
```