Assignment 6: MLE for various distributions

Tianzhi, Abhishek, Harshini

2020-10-20

```
suppressWarnings({
  MLEestimator<-function(vec0,funcname){
    n<<-length(vec0)
    meanv<<-mean(vec0)
    snv=sum((vec0-meanv)^2)
    sumv<<-sum(vec0)
    sqrv=sum(vec0^2)

    if (funcname=="Bernoulli"){
      cat(sprintf("\n ---- Bernoulli ----\n"))
      funct<-function(x)(x^sumv)*((1-x)^(n-sumv))
      pval=optimize(funct,c(0,1),tol=0.0001,maximum = TRUE)
      print(paste ("p:", pval$maximum))
    }
    if (funcname=="Geometric"){
      cat(sprintf("\n ---- Geometric ----\n"))
      funct<-function(x)(n*log2(x))+(sumv*log2(1-x))
      pval=optimize(funct,c(0,1),tol=0.0001,maximum = TRUE)
      print(paste ("p:", pval$maximum))
    }
    if (funcname=="Normal"){
      cat(sprintf("\n ---- Normal ----\n"))
      mu=sumv/n
      paste("mu:",mu)
      samu=sum((vec0-mu)^2)
      funct<-function(x)-(n/2)*(log(2*pi*x^2)) + (-1/(2*x^2)) *samu
      pval=optimize(funct,c(0,1000),tol=0.001,maximum = TRUE)
      print(paste ("sigma:", pval$maximum))
    }
    if (funcname=="Chisq"){
      cat(sprintf("\n ---- Chisq ----\n"))
      funct<-function(x)(x/2-1)*sum(log2(vec0))-(0.5*sumv)-(n*log2(gamma(x/2)))-((n*x/2)*log2(2))
      pval=optimize(funct,c(1,100),tol=0.001,maximum = TRUE)
      print(paste ("k:", pval$maximum))
    }
    if (funcname=="Multivariate Normal"){
      cat(sprintf("\n ---- Multivariate Normal ----\n"))
      mu=apply(vec0,2,mean)
      print(mu)
      sumi=t(vec0-mu)%*%(vec0-mu)
      sumi=sumi/1000
      print (sumi)
```

```
}

if (funcname=="Binomial"){
  cat(sprintf("\n ---- Binomial ----\n"))
  nval = (1.0/p_true)*mean(vec0)

  # max log likelihood
  funct<-function(x) (sumv*log(x) + (length(vec0)*nval-sumv)*log(1-x))
  pval=optimize(funct,c(0,1),tol=0.0001,maximum = TRUE)
  print(paste ("p:", pval$maximum))
}
if (funcname=="Exponential"){
  cat(sprintf("\n ---- Exponential ----\n"))
  # max log likelihood
  funct<-function(x) (length(vec0)*log(x) - x*sum(vec0))
  lambdaval=optimize(funct,c(0,lambda_true+1000),tol=0.0001,maximum = TRUE)
  print(paste ("lambda:", lambdaval$maximum))
}
if(funcname=="Beta"){
  cat(sprintf("\n ---- Beta ----\n"))
  loglik <- function(mu, x) {
    sum(-dbeta(x,mu[1],mu[2],log = TRUE))
  }
  # max log likelihood
  out <- optim(par = c(1,1), fn=loglik,x=vec0,method = "L-BFGS-B",lower=c(0,0))
  print(paste("alpha:", out$par[1]))
  print(paste("beta:", out$par[2]))
}
if(funcname=="Multinomial"){
  cat(sprintf("\n ---- Multinomial ----\n"))
  n0=sum(vec0[,1])
  vec1<-apply(vec0, 2, "/", n0)
  out<-rowMeans(vec1)
  print("pvec:")
  print(out)
}

if (funcname == "Poisson") {
  cat(sprintf("\n ---- Poisson ---- \nParameters: Lambda = 0.5 \n"))

  theta_hat = meanv
  cat(sprintf("Parameter Estimates: %s \n", theta_hat))

  q_hat <- qpois(c(1:n)/(n+1), theta_hat)
  D0 <- ks.test(vec0, q_hat)$statistic

  D_vec<-NULL

  nboot = 1000
  for(i in 1 : nboot){
    x_star <- rpois(n, theta_hat)
    theta_hat_star <- mean(x_star)
```

```
    q_hat_star <- qpois(c(1:n)/(n+1), theta_hat_star)
    D_star <- ks.test(x_star, q_hat_star)$statistic
    D_vec <- c(D_vec, D_star)
  }

  p_value <- sum(D_vec > D0)/nboot
  cat(sprintf("\nThe p-value is: %s \n", p_value))
}
if (funcname == "Gamma") {
  alpha = 5
  beta = 20
  cat(sprintf("\n ---- Gamma ---- \nParameters: alpha = 5 beta = 20 \n"))

  input_data <- vec0 + 1e-6

  theta_hat_func <- function(data) {
    s = log(mean(data)) - (sum(log(data)))/length(data)
    estimated_alpha <- ((3 - s) + sqrt( ((s-3)**2) + (24*s) ))/(12*s)
    estimated_beta <- mean(data)/estimated_alpha
    return (c(estimated_alpha, estimated_beta))
  }

  theta_hat <- theta_hat_func(input_data)
  cat(sprintf("Parameter Estimates: %s %s\n", theta_hat[1], theta_hat[2]))

  nboot <- 1000
  q_hat <- qgamma(c(1:n)/(n+1), shape = theta_hat[1],  scale = theta_hat[2])

  D0 <- ks.test(input_data, q_hat)$statistic
  D_vec<-NULL

  for(i in 1 : nboot){
    x_star <- rgamma(n, shape = theta_hat[1], scale = theta_hat[2])
    theta_hat_star <- theta_hat_func(x_star)

    q_hat_star <- qgamma(c(1:n)/(n+1), shape = theta_hat_star[1], scale = theta_hat_star[2])
    D_star <- ks.test(x_star, q_hat_star)$statistic
    D_vec <- c(D_vec, D_star)
  }
  p_value <- sum(D_vec > D0)/nboot

  cat(sprintf("\nThe p-value is: %s \n", p_value))
}
if (funcname == "Uniform") {
  a = 0
  b = 100
  cat(sprintf("\n ---- Uniform ---- \nParameters: a = 0 b = 100 \n"))

  theta_hat_func <- function(data) {
    estimated_a <- min(data)
    estimated_b <- max(data)
    return (c(estimated_a, estimated_b))
  }
```

```
    theta_hat <- theta_hat_func(vec0)

    cat(sprintf("Parameter Estimates: %s %s\n", theta_hat[1], theta_hat[2]))
    nboot <- 1000

    q_hat <- qunif(c(1:n)/(n+1), theta_hat[1], theta_hat[2])

    D0 <- ks.test(vec0, q_hat)$statistic
    D_vec<-NULL

    for(i in 1:nboot){
      x_star <- runif(n, theta_hat[1], theta_hat[2])
      theta_hat_star <- theta_hat_func(x_star)

      q_hat_star <- qunif(c(1:n)/(n+1), theta_hat_star[1], theta_hat_star[2])
      D_star <- ks.test(x_star, q_hat_star)$statistic
      D_vec <- c(D_vec, D_star)
    }
    p_value <- sum(D_vec > D0)/nboot
    cat(sprintf("\nThe p-value is: %s \n", p_value))
  }
  if (funcname == "PointMass") {
    unique_ele <- unique(vec0)
    cat(sprintf("\n ---- PointMass ---- \n"))
    cat(sprintf("\nThe p-value is: %s \n", (unique_ele
                                           [which.max(
                                             tabulate(
                                               match(vec0, unique_ele)))])))
  }
}

library(MASS)

MLEestimator(rbinom(1000,1,0.6),"Bernoulli")
MLEestimator(rgeom(1000,0.6),"Geometric")
MLEestimator(rnorm(1000,3,3),"Normal")
MLEestimator(rchisq(1000,2),"Chisq")

Sigma <- matrix(c(10,3,3,2),2,2)
mvn <- mvrnorm(n=1000, rep(0, 2), Sigma)
MLEestimator(mvn,"Multivariate Normal")

n_true<-10000
p_true<-0.76
MLEestimator(rbinom(1000,n_true,p_true),"Binomial")

lambda_true <- 13
MLEestimator(rexp(1000,rate=lambda_true),"Exponential")

alpha_true <- 7
beta_true <- 4
MLEestimator(rbeta(1000,alpha_true,beta_true),"Beta")
```

```r
  n_true<-10000
  pvec_true<-c(0.4, 0.21, 0.095, 0.043, 0.067, 0.185)
  MLEestimator(rmultinom(1, n_true, pvec_true),"Multinomial")

  MLEestimator(sample(c(0,1), replace=TRUE, size=1000),"PointMass")
  MLEestimator(rpois(1000, 0.5) ,"Poisson")
  MLEestimator(runif(1000, 0, 100)  ,"Uniform")
  MLEestimator(rgamma(1000, shape = 5, scale = 20),"Gamma")
  #MLEestimator(rhyper(1000, 5, 35, 3),"Hypergeometric")

})
```

```
##
##  ---- Bernoulli ----
## [1] "p: 0.615001742102675"
##
##  ---- Geometric ----
## [1] "p: 0.600238640444926"
##
##  ---- Normal ----
## [1] "sigma: 3.01711839624172"
##
##  ---- Chisq ----
## [1] "k: 1.99533467665807"
##
##  ---- Multivariate Normal ----
## [1] -0.005479542  0.019424359
##           [,1]     [,2]
## [1,] 10.810099 3.175433
## [2,]  3.175433 2.059336
##
##  ---- Binomial ----
## [1] "p: 0.759985828383783"
##
##  ---- Exponential ----
## [1] "lambda: 13.5316660353757"
##
##  ---- Beta ----
## [1] "alpha: 7.07921599889271"
## [1] "beta: 4.08515068724437"
##
##  ---- Multinomial ----
## [1] "pvec:"
## [1] 0.4000 0.2069 0.0942 0.0463 0.0695 0.1831
##
##  ---- PointMass ----
##
## The p-value is: 1
##
##  ---- Poisson ----
## Parameters: Lambda = 0.5
## Parameter Estimates: 0.514
##
```

```
## The p-value is: 0.704
##
##  ---- Uniform ----
## Parameters: a = 0 b = 100
## Parameter Estimates: 0.20229616202414 99.8849296011031
##
## The p-value is: 0.106
##
##  ---- Gamma ----
## Parameters: alpha = 5 beta = 20
## Parameter Estimates: 4.92444023657612 20.7075083042067
##
## The p-value is: 0.842
```