Assignment 8: Goodness of fit for MLE of various distributions

Tianzhi, Abhishek, Harshini

2020-10-28

```
suppressWarnings({
  MLEestimatorWithGoodnessOfFit<-function(vec0,funcname){
    n<<-length(vec0)
    meanv<<-mean(vec0)
    snv=sum((vec0-meanv)^2)
    sumv<<-sum(vec0)
    sqrv=sum(vec0^2)
    res <- NULL
    if (funcname=="Bernoulli"){
      funct<-function(x)(x^sumv)*((1-x)^(n-sumv))
      pval=optimize(funct,c(0,1),tol=0.0001,maximum = TRUE)
      paste ("p:", pval$maximum)
      hat=ks.test(vec0,"pbinom",1,pval$maximum)$statistic
      res<-list(para=c(pval$maximum),hatks=hat)
    }
    if (funcname=="Geometric"){
      funct<-function(x)(n*log2(x))+(sumv*log2(1-x))
      pval=optimize(funct,c(0,1),tol=0.0001,maximum = TRUE)
      hat=ks.test(vec0,"pbinom",1,pval$maximum)$statistic
      res<-list(para=c(pval$maximum),hatks=hat)
    }
    if (funcname=="Normal"){
      mu=sumv/n
      paste("mu:",mu)
      samu=sum((vec0-mu)^2)
      funct<-function(x)-(n/2)*(log(2*pi*x^2)) + (-1/(2*x^2)) *samu
      pval=optimize(funct,c(0,1000),tol=0.001,maximum = TRUE)
      hat=ks.test(vec0,"pnorm",mu,pval$maximum)$statistic
      paste ("sigma:", pval$maximum)
      res<-list(para=c(mu,pval$maximum),hatks=hat)
    }
    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
```

```
    #parametric bootstrap
    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

    #parametric bootstrap
    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

  #parametric bootstrap
  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 == "Binomial") {
  n = 1000
  p = 0.529
  cat(sprintf("\n ---- Binomial ---- \nParameters: n = 1000 p = 0.5 \n"))

  theta_hat_func <- function(data) {
    n <- length(data)
    estimated_p <- (1 / n) * (sum(data)/n)
    return(estimated_p)
  }

  theta_hat <- theta_hat_func(vec0)

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

}
if (funcname == "Exponential") {
  theta = 2
  cat(sprintf("\n ---- Exponential ---- \nParameters: theta = 2 \n"))

  theta_hat_func <- function(data) {
    estimated_theta <- mean(data)
    return(estimated_theta)
```

```
    }

    theta_hat <- theta_hat_func(vec0)
    cat(sprintf("Parameter Estimates: %s \n", theta_hat))

    nboot <- 1000
    q_hat <- qexp(c(1:n) / (n+1), theta_hat)

    D_0 <- ks.test(vec0, q_hat)$statistic
    D_vec<-NULL

    for(i in 1:nboot){
      x_star <- rexp(n, theta_hat)
      theta_hat_star <- theta_hat_func(x_star)

      q_hat_star <- qexp(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 > D_0)/nboot
    cat(sprintf("\nThe p-value is: %s \n", p_value))
  }
  if (funcname == "Beta") {

    alpha = 5.5
    beta = 2.5

    cat(sprintf("\n ---- Beta ---- \nParameters: alpha = 5.5 beta = 2.5 \n"))

    theta_hat_func <- function(data) {
      n <- length(data)
      mean <- mean(data)
      variance <- (sum(data * data)) / n
      alpha <- ((mean ^ 2) - (mean * variance))/(variance - (mean ^ 2))
      beta <- (alpha * (1 - mean))/(mean)

      estimates <- c(alpha, beta)

      # Let us run the optimization step
      for(index in 1:100){
        g1 <- digamma(alpha) - digamma(alpha + beta) - (sum(log(data)))/n
        g2 <- digamma(beta) - digamma(alpha + beta) - (sum(log(1 - data))/n)
        g <- c(g1, g2)

        G1_val <- trigamma(alpha) - trigamma(alpha + beta)
        G2_val <- -trigamma(alpha + beta)
        G3_val <- trigamma(beta) - trigamma(alpha + beta)
        G <- matrix(c(G1_val, G2_val, G2_val, G3_val), nrow = 2, ncol = 2, byrow = TRUE)
        G_inverse <- solve(G)

        # Final values
        estimates <- estimates - t(G_inverse %*% g)
```

```
        alpha <- estimates[1]
        beta <- estimates[2]

        return(c(alpha, beta))
      }

    }
    nboot <- 1000
    theta_hat <- theta_hat_func(vec0)
    cat(sprintf("Parameter Estimates: %s %s\n", theta_hat[1], theta_hat[2]))

    # Parametric bootstrap
    q_hat <- qbeta(c(1 : n)/(n + 1), shape1 = theta_hat[1], shape2 = theta_hat[2])
    D0 <- ks.test(vec0, q_hat)$statistic
    D_vec<-NULL

    for(i in 1 : nboot){
      x_star <- rbeta(n, shape1 =  theta_hat[1], shape2 =  theta_hat[2])
      theta_hat_star <- theta_hat_func(x_star)

      q_hat_star <- qbeta(c(1:n)/(n+1), shape1 =  theta_hat_star[1], shape2 = 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))
  }
  return (res)
}
Paraboot<-function(funcname,parameter,l,n,hatks){
  correctrate=0
  cat("hat", hatks)
  for (i in 1:n){
    star=0
    hat=0
    if (funcname=="Bernoulli"){
      data=rbinom(l,1,parameter[1])
      parahatstar=MLEestimatorWithGoodnessOfFit(data,"Bernoulli")$para
      star=ks.test(data,"pbinom",1,parahatstar[1])$statistic
    }
    else if (funcname=="Geometric"){
      data=rgeom(l,parameter[1])
      parahatstar=MLEestimatorWithGoodnessOfFit(data,"Geometric")$para
      star=ks.test(data,"pgeom",parahatstar[1])$statistic
    }
    else if (funcname=="Normal"){
      data=rnorm(l,parameter[1],parameter[2])
      parahatstar=MLEestimatorWithGoodnessOfFit(data,"Normal")$para
      star=ks.test(data,"pnorm",parahatstar[1],parahatstar[2])$statistic
    }
    if (star>hatks){
      correctrate=correctrate+1
```

```
      }
    }
    cat (correctrate/n)
  }

  lis<-MLEestimatorWithGoodnessOfFit(rbinom(1000,1,0.6),"Bernoulli")
  hat<-lis$hatks
  para<-lis$para
  Paraboot("Bernoulli",para,1000,1000,hat)

  lis<-MLEestimatorWithGoodnessOfFit(rgeom(1000,0.6),"Geometric")
  hat<-lis$hatks
  para<-lis$para
  Paraboot("Geometric",para,1000,1000,hat)

  lis<-MLEestimatorWithGoodnessOfFit(rnorm(1000,3,9),"Normal")
  hat<-lis$hatks
  para<-lis$para
  Paraboot("Normal",para,1000,1000,hat)

  MLEestimatorWithGoodnessOfFit(sample(rpois(10000, 0.5), 1000), "Poisson")
  MLEestimatorWithGoodnessOfFit(sample(runif(10000, 0, 100), 1000), "Uniform")
  MLEestimatorWithGoodnessOfFit(sample(rgamma(10000, shape = 5, scale = 20), 1000), "Gamma")
  MLEestimatorWithGoodnessOfFit(sample(rbinom(10000, 1, 0.5), 1000), "Binomial")
  MLEestimatorWithGoodnessOfFit(sample(rexp(10000, 2), 1000), "Exponential")
  MLEestimatorWithGoodnessOfFit(sample(rbeta(10000, shape1 = 5.5, shape2 = 2.5), 1000), "Beta")

})
```

```
## hat 0.6110.482hat 0.3931hat 0.019266980.508
##  ---- Poisson ----
## Parameters: Lambda = 0.5
## Parameter Estimates: 0.533
##
## The p-value is: 0.832
##
##  ---- Uniform ----
## Parameters: a = 0 b = 100
## Parameter Estimates: 0.00395150855183601 99.9664396280423
##
## The p-value is: 0.228
##
##  ---- Gamma ----
## Parameters: alpha = 5 beta = 20
## Parameter Estimates: 4.87979695361695 20.3595761774862
##
## The p-value is: 0.279
##
##  ---- Binomial ----
## Parameters: n = 1000 p = 0.5
## Parameter Estimates: 5e-04
##
##  ---- Exponential ----
```

```
## Parameters: theta = 2
## Parameter Estimates: 0.481113486608241
##
## The p-value is: 0.591
##
##  ---- Beta ----
## Parameters: alpha = 5.5 beta = 2.5
## Parameter Estimates: 5.33122743033706 2.37788646848474
##
## The p-value is: 0.066

## NULL
```