```r
library(mvtnorm)

library(micEcon)

# calculate mle of binomial distribution -

calculate_binom_mle <- function(sample) {

  # calculate negative log likelihood

  nll <- function(data, n, p) {

    -sum(dbinom(x=data, size=n, prob=p, log=TRUE))

  }


  mle = optim(par = c(p = 0.5), n = 10, fn = nll, data = sample, method = "BFGS")

  return(mle)

}


# calculate MLE of multivariate distribution

calculate_multivariate_mle <- function(sample) {

  # calculate negative log likelihood

  nll <- function(par, data) {

    mu = c(par[1], par[2])

    #print(mu)

    sigma = matrix(c(par[3], par[4], par[5], par[6]), 2, 2)

    sigma[lower.tri(sigma)] = t(sigma)[lower.tri(sigma)]

    #print(sigma)

    -sum(dmvnorm(x = data, mu, sigma, log = TRUE))

  }

  mle = optim(par = c(mu = c(2,2), sigma = matrix(c(2,2,2,2), 2, 2)), fn = nll, data = sample, method =
"SANN")

  return(mle)

}



# calculate mle of multinomial distribution -
```

```r
calculate_multinom_mle <- function(sample) {

  # calculate negative log likelihood

  nll <- function(parameters, data) {


    p = parameters
    # cat("p",p)
    #-sum(dmultinom(x=data[,1], size=NULL, prob=p, log=TRUE))


    -sum(apply(X = data,MARGIN = 2,FUN = dmultinom,size =1, prob = p, log = TRUE))
  }
  len = nrow(sample)
  mle = optim(par <- rep(1/len,len), fn = nll, data = sample, method = "L-BFGS-B",
          lower = rep(0.0000000001,len),upper = rep(1,len))
  return(mle)

}


# calculate mle of geometric
calculate_geometric_mle <- function(sample) {
  nll <- function(parameters, data) {
    prob = parameters[1]
    -sum(dgeom(x=data, prob=prob, log=TRUE))
  }


  mle = optim(par = c(prob=0.4), fn=nll, data=sample,
          control = list(parscale = c(prob=0.4)))
  return(mle)


}


# calculate mle of poisson
calculate_poisson_mle <- function(sample) {
```

```r
nll <- function(parameters, data) {

  lambda = parameters[1]

  -sum(dpois(x = data, lambda = lambda, log=TRUE))

 }


 mle = optim(par = c(lambda=8), fn = nll, data = sample,

         control = list(parscale = c(lambda = 8)))

 return(mle)

}


# calculate one step of poisson

onestep_pois <- function(sample, m, reptn = 100)

{

 #mean of sample poisson distribution

 x_mean <- mean(sample)


 for(i in 1:reptn)

 {

  #log-likelihood of PDF of poisson

  lglik <- expression(log((exp(-m)*(m^(sample))/factorial(sample))))

  #First partial derivative of log-likelihood wrt mu

  f_derv <- D(lglik, "m")

  frst_derv <- eval(f_derv)

  #Second partial derivative of log-likelihood wrt mu

  s_derv <- D(f_derv, "m")

  sec_derv <- eval(s_derv)


  sdbt <- sum(frst_derv)

  sdbtt <- sum(sec_derv)


  #sequence that converges to MLE
```

```
      theta <- m

      theta.hat <- theta - (sdbt / sdbtt)

      m <- theta.hat

   }


  return(m)

}


# calculate mle of exponential -

calculate_exp_mle <- function(sample) {

 nll <- function(parameters, data) {

   lambda = parameters[1]

   -sum(dexp(x = data, rate  = lambda, log=TRUE))

 }


  mle = optim(par = c(lambda=8), fn = nll, data = sample,

          control = list(parscale = c(lambda = 8)))

  return(mle)

}


# calculate mle of beta -

calculate_beta_mle <- function(sample) {

 nll <- function(parameters, data) {

   alpha = parameters[1]

   beta = parameters[2]

   -sum(dbeta(x = data, shape1 = alpha, shape2 = beta, log=TRUE))

 }


  mle = optim(par = c(alpha = 1, beta = 10), fn = nll, data = sample,

          control=list(parscale = c(alpha = 1, beta = 10)))

  return(mle)
```

```r
}


# calculate mle of uniform distribution
calculate_uniform_mle <- function(sample) {
  nll <- function(parameters, data) {
    min = parameters[1]
    max = parameters[2]
    -sum(dunif(x = data, min = min, max = max, log=TRUE))
  }


  mle = optim(par = c(min = -10, max = 10), fn = nll, data = sample,
          control=list(parscale = c(min = -10, max = 10)))
  return(mle)
}


# calculate mle of normal distribution
calculate_normal_mle <- function (sample) {
  # calculate negative log likelihood
  nll <- function(parameters, data) {
    mu = parameters[1]
    sigma = parameters[2]
    -sum(dnorm(x = data, mean = mu, sd = sigma, log = TRUE))
  }


  mle = optim(par = c(mu=0.2, sigma = 1.5), fn = nll, data = sample,
          control = list(parscale = c(mu = 0.2, sigma = 1.5)))


  return(mle)
}
```

```r
main <- function (sample_vec,dist_type)
{


  if(dist_type == "uniform" || dist_type == "Uniform" || dist_type == "UNIFORM")
  {
    uniform_sample = runif(10, -10, 10)
    uniform_mle = calculate_uniform_mle(uniform_sample)
    cat("\n\ndistribution type is ",dist_type)
    cat("\n\nsample is\n",sample,"\n")
    cat("MLE is given Below\n")
    print(uniform_mle$par)


  }

  else if(dist_type == "beta" || dist_type == "Beta" || dist_type == "BETA")
  {
    sample = rbeta(10,1,10)
    cat(sample)
    sample_mle = calculate_beta_mle(sample)
    cat("\n\ndistribution type is ",dist_type)
    cat("\n\nsample is\n",sample,"\n")
    cat("MLE is given Below\n")
    print(sample_mle$par)


  }
  else if(dist_type == "normal" || dist_type == "Normal" || dist_type == "NORMAL")
  {
    sample = rnorm(10)
    normal_mle = calculate_normal_mle(sample)
    cat("\n\ndistribution type is ",dist_type)
```

```r
  cat("\n\nsample is\n",sample,"\n")

  cat("MLE is given Below\n")

  print(normal_mle$par)

}
else if(dist_type=="poisson"||dist_type=="Poisson"||dist_type=="POISSON")

{


  sample = rpois(10, 8)

  poisson_mle = calculate_poisson_mle(sample)

  cat("\n\ndistribution type is ",dist_type)

  cat("\n\nsample is\n",sample,"\n")

  cat("MLE is given Below\n")

  print(poisson_mle$par)

  posison_os = onestep_pois(sample, 8, 100)

  cat("The approximated MLE value using one-step is",posison_os,"\n")

}


else if(dist_type=="geometric"||dist_type=="Geometric"||dist_type=="GEOMETRIC")

{


  sample = rgeom(10, 0.4)

  cat("\n\ndistribution type is ",dist_type)

  cat("\n\nsample is\n",sample,"\n")

  cat("MLE is given Below\n")

  geom_mle = calculate_geometric_mle(sample)

  print(geom_mle$par)

}
else if(dist_type=="binomial"||dist_type=="Binomial"||dist_type=="BINOMIAL")

{

  sample = rbinom(4,10,0.5)

  cat("\n\ndistribution type is ",dist_type)
```

```r
    cat("\n\nsample is\n",sample,"\n")

    cat("MLE is given Below\n")

    binom_mle = calculate_binom_mle(sample)

    print(binom_mle$par)

  }
  else if(dist_type=="exponential"||dist_type=="Exponential"||dist_type=="EXPONENTIAL")

  {

    sample = rexp(5)

    cat("\n\ndistribution type is ",dist_type)

    cat("\n\nsample is\n",sample,"\n")

    cat("MLE is given Below\n")

    sample_mle = calculate_exp_mle(sample)

    print(sample_mle$par)

  }
  else if(dist_type=="multinomial"||dist_type=="Multinomial"||dist_type=="MULTINOMIAL")

  {

    sample = rmultinom(5,1,rep(1/10,10))

    cat("\n\ndistribution type is ",dist_type)

    cat("\n\nsample is\n",sample,"\n")

    cat("MLE is given Below\n")

    sample_mle = calculate_multinom_mle(sample)

    print(sample_mle$par)

  }


  else if(dist_type =="Multivariate Normal"||dist_type=="multivariate
normal"||dist_type=="MULTIVARIATE NORMAL")


  {

    sigma <- matrix(c(3,2,2,8), 2, 2)

    mu <- c(7,10)

    sample <- rmvnorm(10, mean = mu, sigma = sigma)
```

```r
    cat("\n\ndistribution type is ",dist_type)

    cat("\n\nsample is")

    print(sample)

    cat("MLE is given Below\n")

    sample_mle = calculate_multivariate_mle(sample)

    print(sample_mle$par)

  }


}


main(c(),'poisson')
```