Assignment 4

Maximum Likelihood Estimators for various probability distributions

Maximum Likelihood Estimators

Maximum Likelihood Estimation (MLE) is a method of estimating the parameters of a statistical model, given observations.

The method of maximum likelihood is used with a wide range of statistical analyses. As an example, suppose that we are interested in the heights of adult female penguins, but are unable to measure the height of every penguin in a population (due to cost or time constraints). Assuming that the heights are normally distributed with some unknown mean and variance, the mean and variance can be estimated with MLE while only knowing the heights of some sample of the overall population. MLE would accomplish that by taking the mean and variance as parameters and finding particular parametric values that make the observed results the most probable given the normal model.

In [1]:
```
#importing libraries
library(Rlab)
library(pracma)

#supressing warning in Jupyter Notebook
options(warn=-1)
```

```
Rlab 2.15.1 attached.


Attaching package: ‘Rlab’

The following objects are masked from ‘package:stats’:

    dexp, dgamma, dweibull, pexp, pgamma, pweibull, qexp, qgamma,
    qweibull, rexp, rgamma, rweibull

The following object is masked from ‘package:datasets’:

    precip
```

In [2]:

```r
# Bernoulli Distribution
mle_bernoulli <- function(data){
  p <- mean(data)
  return(p)
}

# Binomial Distribution
mle_binomial <- function(data){
  n <- length(data)
  p <- (1/length(data))*(sum(data)/n)
  return(p)
}

# Geometric Distribution
mle_geometric <- function(data){
  p <- 1.0/(mean(data))
  return(p)
}

# Poisson Distribution
mle_poisson <- function(data){
  estimated_lambda <- mean(data)
  return(estimated_lambda)
}

# Uniform Distribution
mle_uniform <- function(data){
  a <- min(data)
  b <- max(data)
  return(c(a, b))
}

# Normal Distribution
mle_normal <- function(data){
  # Estimating the parameters
  mu <- mean(data)
  var <- sum((data - mu)**2)/(length(data) - 1)
  return(c(mu, var))
}

# Exponential Distribution
mle_exponential <- function(data){
  theta <- mean(data)
  return(theta)
}

# Gamma Distribution
mle_gamma <- function(data){
  data <- data + 1e-6
  s = log(mean(data)) - (sum(log(data)))/length(data)
  alpha <- ((3 - s) + sqrt( ((s-3)**2) + (24*s) ))/(12*s)
  beta <- mean(data)/alpha
  return(c(alpha, beta))
}

# Beta Distribution
```

```
mle_beta <- function(data){
  data_mean <- mean(data)
  data_variance <- (sum(data * data))/length(data)
  alpha <- ((data_mean ^ 2) - (data_mean * data_variance))/(data_variance - (
  beta <- (alpha * (1 - data_mean))/(data_mean)

  final_val <- c(alpha, beta)

  # We will run the optimisation step for 100 iterations
  for(index in 1:100){
    g1 <- digamma(alpha) - digamma(alpha + beta) - (sum(log(data)))/length(da
    g2 <- digamma(beta) - digamma(alpha + beta) - (sum(log(1 - data))/length(
    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
    G_inverse <- inv(G)

    # Final values
    final_val <- final_val - t(G_inverse %*% g)
    alpha <- final_val[1]
    beta <- final_val[2]
  }

  return(c(c(alpha, beta)))
}

# Chi Square Distribution
mle_chisq <- function(data){
  # Intitial values for v from MOM estimator
  p_tilda <- mean(data)

  # We will use some approximations using the second derivative
  n <- length(data)
  del_p_numerator <- (-n/gamma(p_tilda/2) * digamma(p_tilda/2)) - (((n * log(
  del_p_denominator <- (-n * trigamma(p_tilda/2)/4)
  del_p <- del_p_numerator/del_p_denominator

  estimated_p <- (p_tilda + del_p)/2
  return(estimated_p)
}
```

Goodness of Fit Test

The goodness of fit of a statistical model describes how well it fits a set of observations. Measures of goodness of fit typically summarize the discrepancy between observed values and the values expected under the model in question.

In [3]:

```
# Goodness of fit function
gfit <- function(distribution, nboot = 1000, data)
{
  mle_name = get(paste("mle_", distribution, sep = ""))
  theta_hat = mle_name(data)
  n <- length(data)

  if(distribution == "poisson"){
    q_hat <- qpois(c(1:n)/(n+1),theta_hat)

    D0 <- ks.test(data, q_hat)$statistic
    Dvec<-NULL

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

      q_hat_star <- qpois(c(1:n)/(n+1), theta_hat_star)
      D_star <- ks.test(x_star, q_hat_star)$statistic
      Dvec <- c(Dvec, D_star)
    }
    p_value <- sum(Dvec > D0)/nboot
    return(p_value)
  }
  else if(distribution == "normal"){
    q_hat <- qnorm(c(1:n)/(n+1),mean = theta_hat[1], sd = theta_hat[2])

    D0 <- ks.test(data, q_hat)$statistic
    Dvec<-NULL

    for(i in 1:nboot){
      x_star <- rnorm(n,mean = theta_hat[1], sd =theta_hat[2])
      theta_hat_star <- mle_name(x_star)

      q_hat_star <- qnorm(c(1:n)/(n+1),mean = theta_hat_star[1], sd =theta_ha
      D_star <- ks.test(x_star, q_hat_star)$statistic
      Dvec <- c(Dvec, D_star)
    }
    p_value <- sum(Dvec > D0)/nboot
    return(p_value)
  }
  else if(distribution == "uniform"){
    q_hat <- qunif(c(1:n)/(n+1), theta_hat[1], theta_hat[2])

    D0 <- ks.test(data, q_hat)$statistic
    Dvec<-NULL

    for(i in 1:nboot){
      x_star <- runif(n, theta_hat[1], theta_hat[2])
      theta_hat_star <- mle_name(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
      Dvec <- c(Dvec, D_star)
    }
    p_value <- sum(Dvec > D0)/nboot
```

```
    return(p_value)
  }
  else if(distribution == "gamma"){
    q_hat <- qgamma(c(1:n)/(n+1), shape = theta_hat[1],  scale = theta_hat[2]

    D0 <- ks.test(data, q_hat)$statistic
    Dvec<-NULL

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

      q_hat_star <- qgamma(c(1:n)/(n+1), shape = theta_hat_star[1], scale = t
      D_star <- ks.test(x_star, q_hat_star)$statistic
      Dvec <- c(Dvec, D_star)
    }
    p_value <- sum(Dvec > D0)/nboot
    return(p_value)
  }
  else if(distribution == "beta"){
    q_hat <- qbeta(c(1:n)/(n+1),shape1 = theta_hat[1], shape2 = theta_hat[2])

    D0 <- ks.test(data, q_hat)$statistic
    Dvec<-NULL

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

      q_hat_star <- qbeta(c(1:n)/(n+1), shape1 =  theta_hat_star[1], shape2 =
      D_star <- ks.test(x_star, q_hat_star)$statistic
      Dvec <- c(Dvec, D_star)
    }
    p_value <- sum(Dvec > D0)/nboot
    return(p_value)
  }
  else if(distribution == "exponential"){
    q_hat <- qexp(c(1:n)/(n+1),theta_hat)

    D0 <- ks.test(data, q_hat)$statistic
    Dvec<-NULL

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

      q_hat_star <- qexp(c(1:n)/(n+1), theta_hat_star)
      D_star <- ks.test(x_star, q_hat_star)$statistic
      Dvec <- c(Dvec, D_star)
    }
    p_value <- sum(Dvec > D0)/nboot
    return(p_value)
  }
}
```

Wrapper funcitons for Goodness of Fit for Maximum Likelihood Estimator

In [4]:

```
mle_wrapper <- function(distribution, population = 0){
  p = 0.5
  lambda = 0.5
  a = 0
  b = 100
  theta = 2
  alpha = 4.7
  beta = 2.9
  dog = 5

  if (distribution == "bernoulli"){
    if (population == 0){
      p = 0.5
      data = rbinom(10000, 1, p)
    }
    print("Population parameter: ")
    print(p)
    sampled = sample(data, 1000)
    param_estimate <- mle_bernoulli(sampled)
    print("Parameter Estimates: ")
    print(param_estimate)
  }
  else if (distribution == "binomial"){
    if (population == 0){
      n = 1000
      p = 0.5
      data = rbinom(10000, n, p)
    }
    print("Population parameters: ")
    print(paste(p,",",n))
    sampled = sample(data, 1000)
    param_estimate <- mle_binomial(sampled)
    print("Parameter Estimates: ")
    print(param_estimate)
  }
  else if (distribution == "geometric"){
    if (population == 0){
      p = 0.5
      data = rgeom(10000, p)
    }
    print("Population parameters: ")
    print(p)
    sampled = sample(data, 1000)
    param_estimate <- mle_geometric(sampled)
    print("Parameter Estimates: ")
    print(param_estimate)
  }
  else if (distribution == "poisson"){
    if (population == 0){
      lambda = 0.5
      data = rpois(10000, lambda)
    }
    print("Population parameters: ")
    print(lambda)
    sampled = sample(data, 1000)
    param_estimate <- mle_poisson(data)
```

```
    print("Parameter Estimates: ")
    print(param_estimate)

    # Doing parametric bootstrap of MLE using ks test
    p_value <- gfit(distribution, data = data)
    print("The p-value is: ")
    print(p_value)
  }
  else if (distribution == "uniform"){
    if (population == 0){
      a = 0
      b = 100
      data = runif(10000,a,b)
    }
    print("Population parameters: ")
    print(paste(a,",",b))
    sampled = sample(data, 1000)
    estimator <- mle_uniform(sampled)
    print("Parameter Estimates: ")
    print(estimator)

    # Doing parametric bootstrap of MLE using ks test
    p_value <- gfit(distribution, data = data)
    print("The p-value is: ")
    print(p_value)
  }
  else if (distribution == "normal"){
    if (population == 0){
      data = rnorm(10000, 0, 1)
    }
    print("Population mean: ")
    print(mean(data))
    print("Population variance: ")
    print(var(data))
    sampled = sample(data, 1000)
    param_estimate <- mle_normal(sampled)
    print("Parameter Estimates: ")
    print(param_estimate)

    # Doing parametric bootstrap of MLE using ks test
    p_value <- gfit(distribution, data = data)
    print("The p-value is: ")
    print(p_value)
  }
  else if (distribution == "exponential"){
    if (population == 0){
      theta = 2
      data = rexp(10000, theta)
    }
    print("Population parameter: ")
    print(theta)
    sampled = sample(data, 1000)
    param_estimate <- mle_exponential(data = sampled)
    print("Parameter Estimates: ")
    print(param_estimate)

    # Doing parametric bootstrap of MLE using ks test
```

```
    p_value <- gfit(distribution, data = data)
    print("The p-value is: ")
    print(p_value)
  }
  else if (distribution == "gamma"){
    if (population == 0){
      alpha = 5
      beta = 20
      data = rgamma(10000, shape = alpha, scale = beta)
    }
    print("Population parameters: ")
    print(paste(alpha,",",beta))
    sampled = sample(data, 1000)
    param_estimate <- mle_gamma(sampled)
    print("Parameter Estimates: ")
    print(param_estimate)

    # Doing parametric bootstrap of MLE using ks test
    p_value <- gfit(distribution, data = data)
    print("The p-value is: ")
    print(p_value)
  }
  else if (distribution == "beta"){
    if (population == 0){
      alpha = 4.7
      beta = 2.9
      data = rbeta(10000, shape1 = alpha, shape2 = beta)
    }
    print("Population parameters: ")
    print(paste(alpha,",",beta))
    sampled = sample(data, 1000)
    param_estimate <- mle_beta(sampled)
    print("Parameter Estimates: ")
    print(param_estimate)

    # Doing parametric bootstrap of MLE using ks test
    p_value <- gfit(distribution, data = data)
    print("The p-value is: ")
    print(p_value)
  }
  else if (distribution == "chi square"){
    if (population == 0){
      dog = 5
      data = rchisq(10000, df = dog)
    }
    print("Population parameter: ")
    print(dog)
    sampled = sample(data, 1000)
    param_estimate <- mle_chisq(sampled)
    print("Parameter Estimates: ")
    print(param_estimate)
  }
}
```

Calling MLE for various distribution

In [5]:
```
#MLE for Bernoulli Distribution
mle_wrapper('bernoulli')
```

```
[1] "Population parameter: "
[1] 0.5
[1] "Parameter Estimates: "
[1] 0.489
```

In [6]:
```
#MLE for Binomial Distribution
mle_wrapper('binomial')
```

```
[1] "Population parameters: "
[1] "0.5 , 1000"
[1] "Parameter Estimates: "
[1] 0.500358
```

In [7]:
```
#MLE for Geometric Distribution
mle_wrapper('geometric')
```

```
[1] "Population parameters: "
[1] 0.5
[1] "Parameter Estimates: "
[1] 1.091703
```

In [8]:
```
#MLE for Poisson Distribution
mle_wrapper('poisson')
```

```
[1] "Population parameters: "
[1] 0.5
[1] "Parameter Estimates: "
[1] 0.4981
[1] "The p-value is: "
[1] 0.324
```

In [9]:
```
#MLE for Uniform Distribution
mle_wrapper('uniform')
```

```
[1] "Population parameters: "
[1] "0 , 100"
[1] "Parameter Estimates: "
[1]  0.007363642 99.960501422
[1] "The p-value is: "
[1] 0.726
```

In [10]:
```
#MLE for Normal Distribution
mle_wrapper('normal')
```

```
[1] "Population mean: "
[1] -0.004709556
[1] "Population variance: "
[1] 0.9681644
[1] "Parameter Estimates: "
[1] -0.02587081  1.00099350
[1] "The p-value is: "
[1] 0.988
```

In [11]:
```
#MLE for Exponential Distribution
mle_wrapper('exponential')
```

```
[1] "Population parameter: "
[1] 2
[1] "Parameter Estimates: "
[1] 0.4950785
[1] "The p-value is: "
[1] 0.589
```

In [12]:
```
#MLE for Gamma Distribution
mle_wrapper('gamma')
```

```
[1] "Population parameters: "
[1] "5 , 20"
[1] "Parameter Estimates: "
[1]  5.002731 20.369871
[1] "The p-value is: "
[1] 0.547
```

In [13]:
```
#MLE for Beta Distribution
mle_wrapper('beta')
```

```
[1] "Population parameters: "
[1] "4.7 , 2.9"
[1] "Parameter Estimates: "
[1] 4.504253 2.771081
[1] "The p-value is: "
[1] 0.683
```

In [14]:
```
#MLE for Chi Squared Distribution
mle_wrapper('chi square')
```

```
[1] "Population parameter: "
[1] 5
[1] "Parameter Estimates: "
[1] 9.284379
```

References:

- https://en.wikipedia.org/wiki/Maximum_likelihood_estimation (https://en.wikipedia.org/wiki/Maximum_likelihood_estimation)
- https://en.wikipedia.org/wiki/Goodness_of_fit (https://en.wikipedia.org/wiki/Goodness_of_fit)