Program

```
MLE_Beta<-function(x, iteration=100){
  mu=mean(x)
  s2<-var(x)
  theta<-c(mu*(-1+ mu*(1-mu)/s2),
    (1-mu)*(-1+ mu*(1-mu)/s2)
  )
  G<-matrix(0, ncol=2, nrow=2)
  g<-matrix(0, nrow=2, ncol=1)
  for(i in 1:iteration){
    g[1,1]=digamma(theta[1])-digamma(theta[1] + theta[2])-sum(log(x))/length(x)
    g[2,1]=digamma(theta[2])-digamma(theta[1] + theta[2])-sum(log(1-x))/length(x)
    G[1,1]=trigamma(theta[1])-trigamma(theta[1]+theta[2])
    G[1,2]=-trigamma(theta[1]+theta[2])
    G[2,1]=-trigamma(theta[1]+theta[2])
    G[2,2]=trigamma(theta[2])-trigamma(theta[1]+theta[2])
    theta<-theta-solve(G) %*% g
  }
  return( list(shape1=theta[1], shape2=theta[2] ) )
}

MLE_Gamma<-function(x, iteratin=100){
  n<-length(x)
  mean_x<-mean(x)
  alpha<-n*(mean_x^2)/sum((x-mean_x)^2)
  beta<-sum((x-mean_x)^2)/n/mean_x
  for(i in 1:iteratin){
    #first derivative of alpha_k-1
    der1<-n*log(alpha/mean_x)-n*digamma(alpha)+sum(log(x))
    #second derivative of alpha_k-1
    der2<-n/alpha-n*trigamma(alpha)
    #calculate next alpha
    alpha<-alpha-der1/der2
    beta<-mean_x/alpha
  }

  return(list(shape=alpha, scale=beta))
}

MLE<-function(x, type, bino=5){
  if(type=="PointMass") { ux <- unique(x) ; return(  ux[which.max(tabulate(match(x, ux)))]) }
  if (type=="Bernoulli") return(  list(size=1, prob=sum(x)/length(x)) )
  if(type=="Binomial") return(  list(size=bino, prob=mean(x)/bino) )
  if(type=="Geometric") return( list(prob=length(x)/mean(x) ))
  if(type=="Poisson") return( list( lambda=mean(x) ) )
  if(type=="Uniform") return( list(min=min(x), max=max(x) ) )
  if(type=="Normal" ) return( list(mean=mean(x), sd=sqrt(  sum( (x-mean(x) )^2)/ length(x)  )))
  if(type=="Exponential") return( list(rate=mean(x) ) )
  if(type=="Gamma") return(MLE_Gamma(x))
  if(type=="Beta") return(MLE_Beta(x))
  #if(type=="t") return(2*(a1^2+a2)/(a1^2+a2-1))
  #if(type=="chi") return(2* )
}
```

Test

We combine parametric bootstrapping and KS-test to test the goodness of fit.

Parametric Bootstrapping

In parametric bootstrapping, we sample from our "estimated" distribution $f(x;\hat{\theta}_n)$ instead of the empirical distribution $\hat{F}_n(x)$

Kolmogorov-Smirnov test

In statistics, the Kolmogorov-Smirnov test (K-S test or KS test) is a nonparametric test of the equality of continuous, one-dimensional probability distributions that can be used to compare a sample with a reference probability distribution (one-sample K-S test), or to compare two samples (two-sample K-S test). It is named after Andrey Kolmogorov and Nikolai Smirnov.

The Kolmogorov-Smirnov statistic quantifies a distance between the empirical distribution function of the sample and the cumulative distribution function of the reference distribution, or between the empirical distribution functions of two samples. The null distribution of this statistic is calculated under the null hypothesis that the sample is drawn from the reference distribution (in the one-sample case) or that the samples are drawn from the same distribution (in the two-sample case). In each case, the distributions considered under the null hypothesis are continuous distributions but are otherwise unrestricted.

The two-sample K-S test is one of the most useful and general nonparametric methods for comparing two samples, as it is sensitive to differences in both location and shape of the empirical cumulative distribution functions of the two samples.(https://en.wikipedia.org/wiki/Kolmogorov%E2%80%93Smirnov_test)

Bootstrap and KS-test code

I created two *partial* functions for later use, which can create a parameters-partially-fed function.

```r
partial <- function(f, ...) {
  l <- list(...)
  function(...) {
    do.call(f, c(l, list(...)))
  }
}
partial_list<-function(f, L){
  function(...){
    do.call(f, c(L, list(...)))
  }
}


KS<-function(x, nboot=10000, rvecFUN, Distritype, FUN){
  n=length(x)
  DVec<-NULL
  theta0<-MLE(x, type=Distritype)
  D0=unname( ks.test(x, partial_list(FUN, theta0))$statistic )

  for(i in 1:nboot){
    xstar<-partial_list(rvecFUN, theta0)(n)
    thetastar<-MLE(xstar, type = Distritype)
    Dstar=unname(ks.test(xstar, partial_list(FUN, thetastar))$statistic )
    DVec<-c(DVec, Dstar)

  }
  return(list(p= sum(DVec>D0)/nboot ))
}
```

Test with Normal Distribution

```
set.seed(553)
x0<-rnorm(n=100, mean=, sd=6)
print( KS(x0, rvecFUN =rnorm, Distritype = "Normal", FUN = pnorm) )
```

```
## $p
## [1] 0.3263
```

Test with Gamma Distribution

```
set.seed(239)
x1<-rgamma(n=100, shape=5, scale=6)

print(KS(x1, rvecFUN = rgamma, Distritype = "Gamma", FUN=pgamma))
```

```
## $p
## [1] 0.3167
```

Test with Beta Distribution

```
set.seed(461)
x2<-rbeta(n=100, shape1=45, shape2 = 13)
print( KS(x2,rvecFUN = rbeta, Distritype = "Beta", FUN = pbeta))
```

```
## $p
## [1] 0.5161
```