Assignment 3c

```
library(plyr)

# suppress warnings
defaultW <- getOption("warn")
options(warn = -1)
# to turn warnings on, warn = 1

# function to initCap input distribution name
initCap <- function(x) {
  s <- strsplit(x, " ")[[1]]
  paste(toupper(substring(s, 1,1)), tolower(substring(s, 2)),
        sep="", collapse=" ")
}

# generate re-sample data for bootstrap
gen_sample <- function(dist, n, mle){

  if (dist == "Binomial"){
    return (rbinom(n, 1, mle[1]))
  }
  if (dist == "Uniform"){
    return (runif(n, mle[1], mle[2]))
  }
  if (dist == "Normal"){
    return (rnorm(n, mle[1], mle[2]))
  }
  if (dist == "Geometric"){
    return (rgeom(n, mle[1]))
  }
  if (dist == "Exponential"){
    return (rexp(n, mle[1]))
  }
  if (dist == "Poisson"){
    return (rpois(n, mle[1]))
  }
  if (dist == "Beta"){
    return (rbeta(n, mle[1], mle[2]))
  }

}

# get theoretical cumulative distribution
th_cum_dist <- function(dist, resample, mle){

  if (dist == "Binomial"){
```

```
    x_1 <- pbinom(resample$freq[2], sum(resample$freq), mle[1])
    x_0 = 1 - x_1
    return (c(x_0, x_1))
  }
  if (dist == "Uniform"){
    return (punif(resample$x, mle[1], mle[2]))
  }
  if (dist == "Normal"){
    return (pnorm(resample$x, mle[1], mle[2]))
  }
  if (dist == "Geometric"){
    return (pgeom(resample$x, mle[1]))
  }
  if (dist == "Exponential"){
    return (pexp(resample$x, mle[1]))
  }
  if (dist == "Poisson"){
    return (ppois(resample$x, mle[1]))
  }
  if (dist == "Beta"){
    return (pbeta(resample$x, mle[1], mle[2]))
  }

}

# direct resample to correct mle functions
cal_mle <- function(dist, boot_resample){

  if (dist == "Binomial"){
    return (cal_binomial_mle(boot_resample))
  }
  if (dist == "Uniform"){
    return (cal_uniform_mle(boot_resample))
  }
  if (dist == "Normal"){
    return (cal_normal_mle(boot_resample))
  }
  if (dist == "Geometric"){
    return (cal_geometric_mle(boot_resample))
  }
  if (dist == "Exponential"){
    return (cal_exp_mle(boot_resample))
  }
  if (dist == "Poisson"){
    return (cal_poisson_mle(boot_resample))
  }
  if (dist == "Beta"){
```

```
    return (cal_beta_mle(boot_resample))
  }

}

# Parametric Bootstrap Re-sampling
bootstrap <- function(data_points, dist, mle, nboot=10000){
  n = length(data_points)
  boot_ks <- vector(mode = "list", length = nboot)
  for(i in 1:nboot){
    boot_resample <- gen_sample(dist, n, mle)
    re_mle <- cal_mle(dist, boot_resample)
    resample <- count(sort(boot_resample))
    Ft_resample <- th_cum_dist(dist, resample, re_mle)
    boot_ks[i] <- ks.test(resample, Ft_resample)
  }
  boot_ks <- as.numeric(unlist(boot_ks))
  return (boot_ks)
}

# function to perform Kolmogorov–Smirnov test
ks.test <- function(sample, Ft_x){
  cum_freq <- cumsum(sample$freq)
  Fs_x <- cum_freq/sum(sample$freq)
  d_n <- abs(Fs_x - Ft_x)
  data.summary <- data.frame(sample, cum_freq, Fs_x, Ft_x, d_n)
  d <- max(data.summary$d_n)
  return (d)
}

# calculate mle of beta
cal_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$par)
}

# calculate mle of poisson
cal_poisson_mle <- function(sample) {
  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$par)
}

# calculate mle of exponential
cal_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$par)
}

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

  mle = optim(par = c(prob=0.5), fn=nll, data=sample,
              control = list(parscale = c(prob=0.5)))
  return(mle$par)
}

# calculate mle of normal distribution
cal_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$par)
}
```

```
# calculate mle of binomial distribution
cal_binomial_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 = 1, fn = nll, data = sample, method = "BFGS")
  return(mle$par)
}

# calculate mle of uniform distribution
cal_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$par)
}

simtest <- function(sample.vec, dist){
  dist = initCap(dist)
  sample <- count(sort(sample.vec), vars = NULL, wt_var = NULL)
  nboot = 10000

  if(dist == "Binomial"){
    mle <- cal_binomial_mle(sample.vec)
    cat("Initial MLE Estimate, p:",mle[1],"\n")
    x_1 <- pbinom(sample$freq[2], length(sample.vec), mle[1])
    x_0 = 1 - x_1
    Ft_x <- c(x_0, x_1)
    d_0 <- ks.test(sample, Ft_x)
    cat("Inital KS value:",d_0,"\n")
    print("Resampling...")
    boot.d <- bootstrap(sample.vec, dist, mle, nboot)
    p_binom <- sum(boot.d > d_0)/nboot
    cat("p value for Binomial Distribution, p =",sum(boot.d >
d_0),"/",nboot,"=",p_binom,"\n")
  }

  if (dist == "Uniform"){
    mle <- cal_uniform_mle(sample.vec)
    cat("Initial MLE Estimate, min:",mle[1]," max:",mle[2],"\n")
```

```
    Ft_x <- punif(sample$x, mle[1], mle[2])
    d_0 <- ks.test(sample, Ft_x)
    cat("Inital KS value:",d_0,"\n")
    print("Resampling...")
    boot.d <- bootstrap(sample.vec, dist, mle, nboot)
    p_unif <- sum(boot.d > d_0)/nboot
    cat("p value for Uniform Distribution, p =",sum(boot.d > d_0),"/",nboot,"=",p_unif,"\n")
  }

  if (dist == "Normal"){
    mle <- cal_normal_mle(sample.vec)
    cat("Initial MLE Estimate, mu:",mle[1]," sigma:",mle[2],"\n")
    Ft_x <- pnorm(sample$x, mle[1], mle[2])
    d_0 <- ks.test(sample, Ft_x)
    cat("Inital KS value:",d_0,"\n")
    print("Resampling...")
    boot.d <- bootstrap(sample.vec, dist, mle, nboot)
    p_norm <- sum(boot.d > d_0)/nboot
    cat("p value for Normal Distribution, p =",sum(boot.d > d_0),"/",nboot,"=",p_norm,"\n")
  }

  if (dist == "Geometric"){
    mle <- cal_geometric_mle(sample.vec)
    cat("Initial MLE Estimate, prob:",mle[1],"\n")
    Ft_x <- pgeom(sample$x, mle[1])
    d_0 <- ks.test(sample, Ft_x)
    cat("Inital KS value:",d_0,"\n")
    print("Resampling...")
    boot.d <- bootstrap(sample.vec, dist, mle, nboot)
    p_geom <- sum(boot.d > d_0)/nboot
    cat("p value for Geometric Distribution, p =",sum(boot.d >
d_0),"/",nboot,"=",p_geom,"\n")
  }

  if (dist == "Exponential"){
    mle <- cal_exp_mle(sample.vec)
    cat("Initial MLE Estimate, lambda:",mle[1],"\n")
    Ft_x <- pexp(sample$x, mle[1])
    d_0 <- ks.test(sample, Ft_x)
    cat("Inital KS value:",d_0,"\n")
    print("Resampling...")
    boot.d <- bootstrap(sample.vec, dist, mle, nboot)
    p_exp <- sum(boot.d > d_0)/nboot
    cat("p value for Exponential Distribution, p =",sum(boot.d >
d_0),"/",nboot,"=",p_exp,"\n")
  }
```

```
  if (dist == "Poisson"){
    mle <- cal_poisson_mle(sample.vec)
    cat("Initial MLE Estimate, lambda:",mle[1],"\n")
    Ft_x <- ppois(sample$x, mle[1])
    d_0 <- ks.test(sample, Ft_x)
    cat("Inital KS value:",d_0,"\n")
    print("Resampling...")
    boot.d <- bootstrap(sample.vec, dist, mle, nboot)
    p_pois <- sum(boot.d > d_0)/nboot
    cat("p value for Poisson Distribution, p =",sum(boot.d > d_0),"/",nboot,"=",p_pois,"\n")
  }

  if (dist == "Beta"){
    mle <- cal_beta_mle(sample.vec)
    cat("MLE Estimate, alpha:",mle[1]," beta:",mle[2],"\n")
    Ft_x <- pbeta(sample$x, mle[1], mle[2])
    d_0 <- ks.test(sample, Ft_x)
    cat("Inital KS value:",d_0,"\n")
    print("Resampling...")
    boot.d <- bootstrap(sample.vec, dist, mle, nboot)
    p_beta <- sum(boot.d > d_0)/nboot
    cat("p value for Beta Distribution, p =",sum(boot.d > d_0),"/",nboot,"=",p_beta,"\n")
  }

}
```

<u>Output</u>:

```
# uniform data, uniform distribution
> simtest(runif(100, -10, 10), "Uniform")
Initial MLE Estimate, min: -9.91348  max: 9.669244
Inital KS value: 0.06906322
[1] "Resampling..."
p value for Uniform Distribution, p = 6303 / 10000 = 0.6303

# beta data, uniform distribution
> simtest(rbeta(100, 1, 4), "Uniform")
Initial MLE Estimate, min: 0.005513852  max: 0.7417489
Inital KS value: 0.3613492
[1] "Resampling..."
p value for Uniform Distribution, p = 0 / 10000 = 0

# binomial data, binomial distribution
> simtest(rbinom(100, 1, 0.5), "BINOMIAL")
Initial MLE Estimate, p: 0.5499996
Inital KS value: 0.4613256
[1] "Resampling..."
```

p value for Binomial Distribution, p = 5259 / 10000 = 0.5259

normal data, normal distribution
> **simtest(rnorm(100, 0, 1), "normal")**
Initial MLE Estimate, mu: -0.1729712 sigma: 0.926787
Inital KS value: 0.05416235
[1] "Resampling..."
p value for Normal Distribution, p = 5539 / 10000 = 0.5539

beta data, normal distribution
> **simtest(rbeta(100, 1, 4), "normal")**
Initial MLE Estimate, mu: 0.2321098 sigma: 0.1873948
Inital KS value: 0.1253937
[1] "Resampling..."
p value for Normal Distribution, p = 4 / 10000 = 4e-04

geom data, geom distribution
> **simtest(rgeom(100, 0.4), "GeoMetric")**
Initial MLE Estimate, prob: 0.3984375
Inital KS value: 0.0423081
[1] "Resampling..."
p value for Geometric Distribution, p = 4412 / 10000 = 0.4412

poisson data, geom distribution
> **simtest(rpois(100, 4), "GeoMetric")**
Initial MLE Estimate, prob: 0.2083008
Inital KS value: 0.2832123
[1] "Resampling..."
p value for Geometric Distribution, p = 0 / 10000 = 0

exponential data, exponential distribution
> **simtest(rexp(100, 1), "Exponential")**
Initial MLE Estimate, lambda: 1.094922
Inital KS value: 0.06115929
[1] "Resampling..."
p value for Exponential Distribution, p = 5678 / 10000 = 0.5678

poisson data, exponential distribution
> **simtest(rpois(100, 4), "Exponential")**
Initial MLE Estimate, lambda: 0.2480469
Inital KS value: 0.1610954
[1] "Resampling..."
p value for Exponential Distribution, p = 0 / 10000 = 0

```
# poisson data, poisson distribution
> simtest(rpois(100, 4), "poissoN")
Initial MLE Estimate, lambda: 3.929687
Inital KS value: 0.02732612
[1] "Resampling..."
p value for Poisson Distribution, p = 8689 / 10000 = 0.8689

# binomial data, poisson distribution
> simtest(rbinom(100, 1, 0.5), "poissoN")
Initial MLE Estimate, lambda: 0.4300781
Inital KS value: 0.08045828
[1] "Resampling..."
p value for Poisson Distribution, p = 2 / 10000 = 2e-04

# beta data, beta distribution
> simtest(rbeta(100, 1, 4), "BEta")
MLE Estimate, alpha: 0.8551802  beta: 3.798108
Inital KS value: 0.04187395
[1] "Resampling..."
p value for Beta Distribution, p = 8962 / 10000 = 0.8962
```