Goodness of Fit

Prepared by: Josh Levine (jl2108), Harsh Patel (hkp49), Jaini Patel (jp1891), Yifan Liao (yl1463) and Aayush Shah (avs93).

Assignment

Using the ks.test - Build a goodness of fit test for each distribution using the parametric bootstrap based on maximum likelihood for every instance in the MOM and MLE estimator program (not including multivariate normal).

What is Goodness of Fit?

The goodness of fit test is a statistical hypothesis test to see how well sample data fit a distribution from a population with a normal distribution.

Goodness-of-fit tests are statistical tests aiming to determine whether a set of observed values match those expected under the applicable model.
There are multiple types of goodness-of-fit tests, but the most common is the chi-square test.
These tests can show you whether your sample data fit an expected set of data from a population with normal distribution.

Understanding Goodness Of Fit

A small p-value (typically <= 0.05) indicates strong evidence against the null hypothesis, so you reject the null hypothesis.

A large p-value (> 0.05) indicates weak evidence against the null hypothesis, so you fail to reject the null hypothesis.

p-values very close to the cutoff (0.05) are considered to be marginal (could go either way). Always report the p-value so your readers can draw their own conclusions.

Build a goodness of fit test for each distribution using the parametric bootstrap based on maximum likelihood

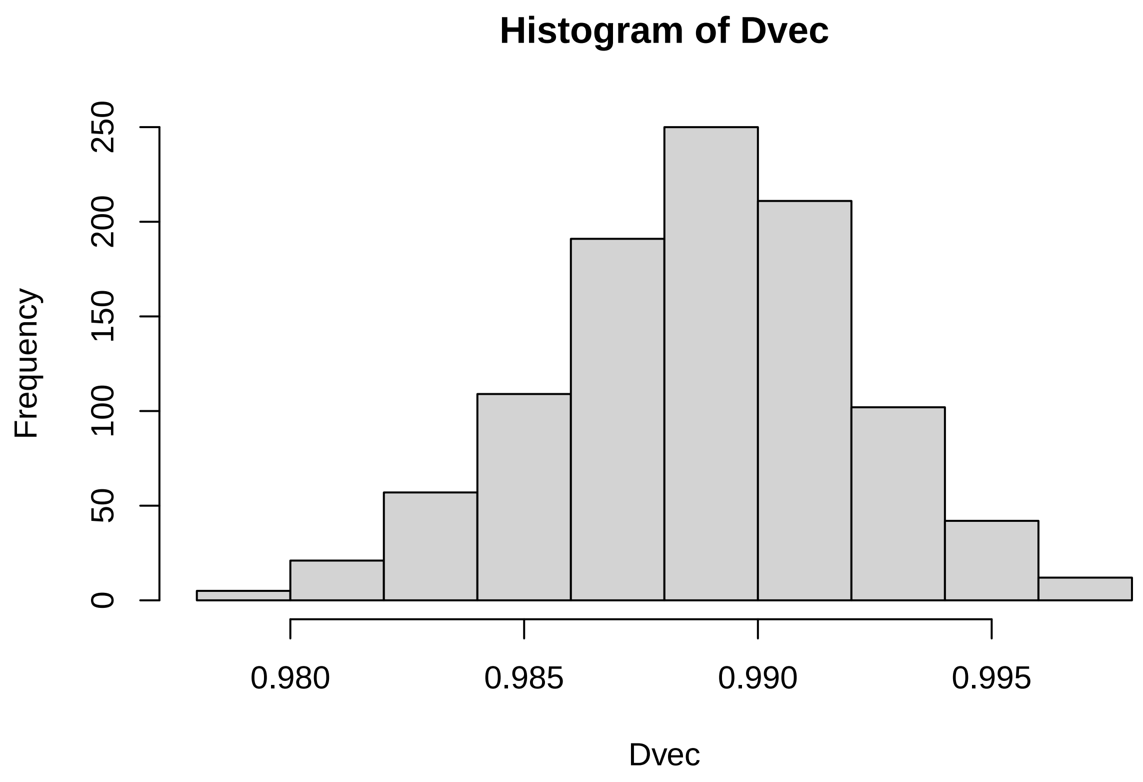
We will be using the same MLE functions from our previous report. Included below is the function that runs the goodness of fit test for a given distribution. Comments have been written for the first distribution as the other calculations are similar.

```r
# Parametric Bootstrap using KS Test
goodness_of_fit.func <- function(distribution, nboot = 1000, input_data)
{
  print(paste("DISTRIBUTION:", distribution, sep=" " ))
  mle_name = get(paste(distribution, "_mle",sep = ""))
  n <- length(input_data)

  #binomail calculates theta_hat slightly differently then the rest
  if(distribution == "binomial")
  {
    theta_hat = mle_name(input_data, n)
  }
  else
  {
    theta_hat = mle_name(input_data)
  }

  if(distribution == "geometric")
  {
    q_hat <- qgeom(c(1:n)/(n+1),theta_hat)#calculate q_hat using this distributions q function
    D0 <- ks.test(input_data, q_hat)$statistic#calculate D0 using ks test
    Dvec<-NULL

    for(i in 1:nboot){
      x_star <- rgeom(n, theta_hat)#generate a random distribution
      theta_hat_star <- mle_name(x_star)#use the mle function previously generated

      q_hat_star <- qgeom(c(1:n)/(n+1), theta_hat_star)#calculate q hat star
      D_star <- ks.test(x_star, q_hat_star)$statistic#use ks test on the data
      Dvec <- c(Dvec, D_star)#combine the vectors for each iteration of the loop
    }
    hist(Dvec)#plot the d vector
    p_value <- sum(Dvec > D0)/nboot #calculate and return the p value
    return(list("p_value" = p_value))
  }
  else if(distribution == "bernoulli")
  {
    Dvec<-NULL
    q_hat <- qbern(c(1:n)/(n+1),theta_hat)
    D0 <- ks.test(input_data, q_hat)$statistic#calculate D0 using ks test
    for(i in 1:nboot){
    vecb <- rbinom(1000, 10, 0.5)
    est.val = mle_name(vecb)
    est.val[2] = est.val[1]
    est.val[1] = 1
    Dvec <- c(Dvec, c(ks.test(vecb, est.val[1], est.val[2])$statistic))
    }
    hist(Dvec)
    p_value <- sum(Dvec > D0)/nboot #calculate and return the p value
    return(list("p_value" = p_value))
  }
  else if(distribution == "binomial")
  {
    q_hat <- qbinom(c(1:n)/(n+1), n,theta_hat)

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

    for(i in 1:nboot){
      x_star <- rbinom(1000, 10, 0.5)
      theta_hat_star <- mle_name(x_star, n)

      q_hat_star <- qbinom(c(1:n)/(n+1), n, theta_hat_star)
      D_star <- ks.test(x_star, q_hat_star)$statistic
      Dvec <- c(Dvec, D_star)
    }
    hist(Dvec)
    p_value <- sum(Dvec > D0)/nboot
    return(list("p_value" = p_value))
  }
  else if(distribution == "poisson"){
    q_hat <- qpois(c(1:n)/(n+1),theta_hat)

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

    D0 <- ks.test(input_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)
    }
    hist(Dvec)
    p_value <- sum(Dvec > D0)/nboot
    return(list("p_value" = 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(input_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 = theta_hat_star[2])
      D_star <- ks.test(x_star, q_hat_star)$statistic
      Dvec <- c(Dvec, D_star)
    }
    hist(Dvec)
    p_value <- sum(Dvec > D0)/nboot
    return(list("p_value" = p_value))
  }
  else if(distribution == "beta"){
    q_hat <- qbeta(c(1:n)/(n+1),shape1 = abs(theta_hat[1]), shape2 = abs(theta_hat[2]))
    D0 <- ks.test(input_data, q_hat)$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 =  abs(theta_hat_star[1]), shape2 = abs(theta_hat_star[2
      D_star <- ks.test(x_star, q_hat_star)$statistic
      Dvec <- c(Dvec, D_star)
    }
    hist(Dvec)
    p_value <- sum(Dvec > D0)/nboot
    return(list("p_value" = p_value))
  }
  else if(distribution == "exponential"){
    q_hat <- qexp(c(1:n)/(n+1),theta_hat)

    D0 <- ks.test(input_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)$statistic
      Dvec <- c(Dvec, D_star)
    }
    hist(Dvec)
    p_value <- sum(Dvec > D0)/nboot
    return(list("p_value" = p_value))
  }
  else if(distribution == "chi_square"){
    q_hat <- qchisq(c(1:n)/(n+1),theta_hat)
    D0 <- ks.test(input_data, q_hat)$statistic
    Dvec<-NULL

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

      q_hat_star <- qchisq(c(1:n)/(n+1), theta_hat_star)
      D_star <- ks.test(x_star, q_hat_star)$statistic
      Dvec <- c(Dvec, D_star)
    }
    hist(Dvec)
    p_value <- sum(Dvec > D0)/nboot
    return(list("p_value" = p_value))
  }
  else if(distribution == "multinomial"){
   q_hat <- qchisq(c(1:n)/(n+1),theta_hat)
    D0 <- ks.test(input_data, q_hat)$statistic
    Dvec<-NULL
    p = c(0.15,0.05,0.4,0.1,0.3)
    for(i in 1:nboot){
      x_star <- rmultinom(10000,size=5,p)
      theta_hat_star <- mle_name(x_star)

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

Below are the outputs (p-values) of the Goodness of Fit Tests for each of the distributions along with a graph of the d-values before the p-value is calculated.

```
goodness_of_fit.func("bernoulli", input_data = rbern(1000, 0.5))
```

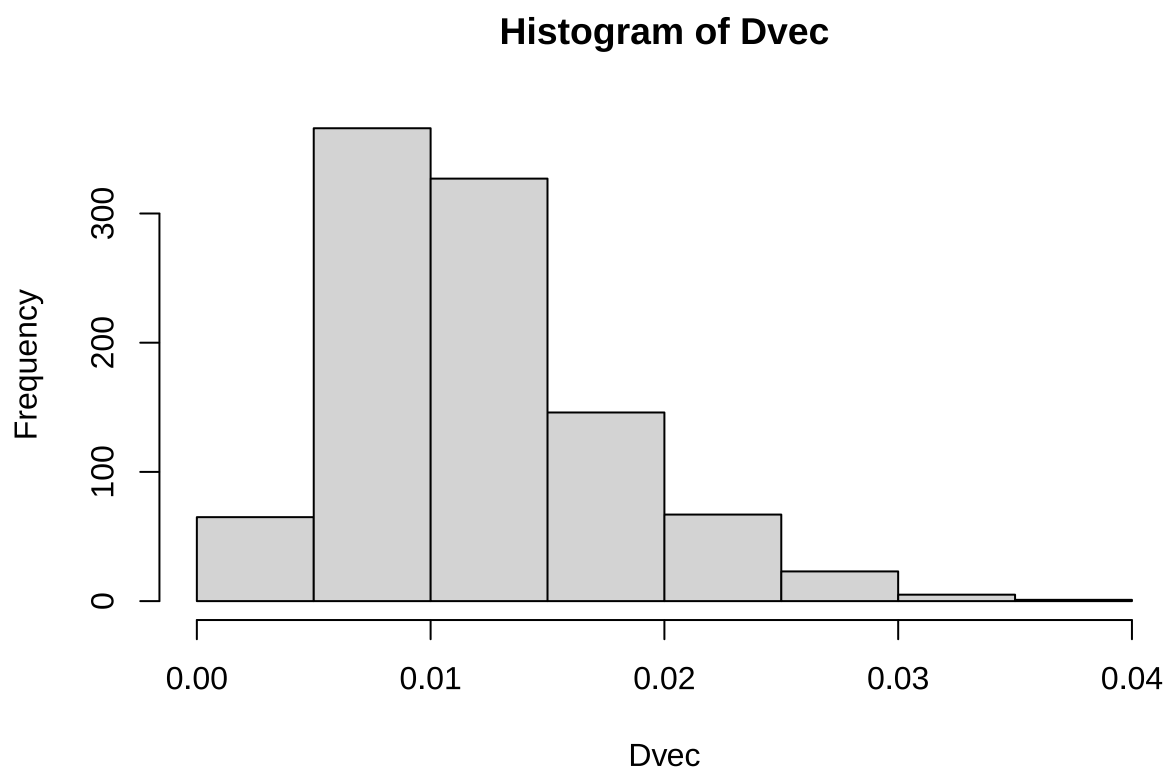
```
## [1] "DISTRIBUTION: bernoulli"
```



```
## $p_value
## [1] 1
```

```r
goodness_of_fit.func("geometric", input_data = rgeom(1000, 0.5))
```

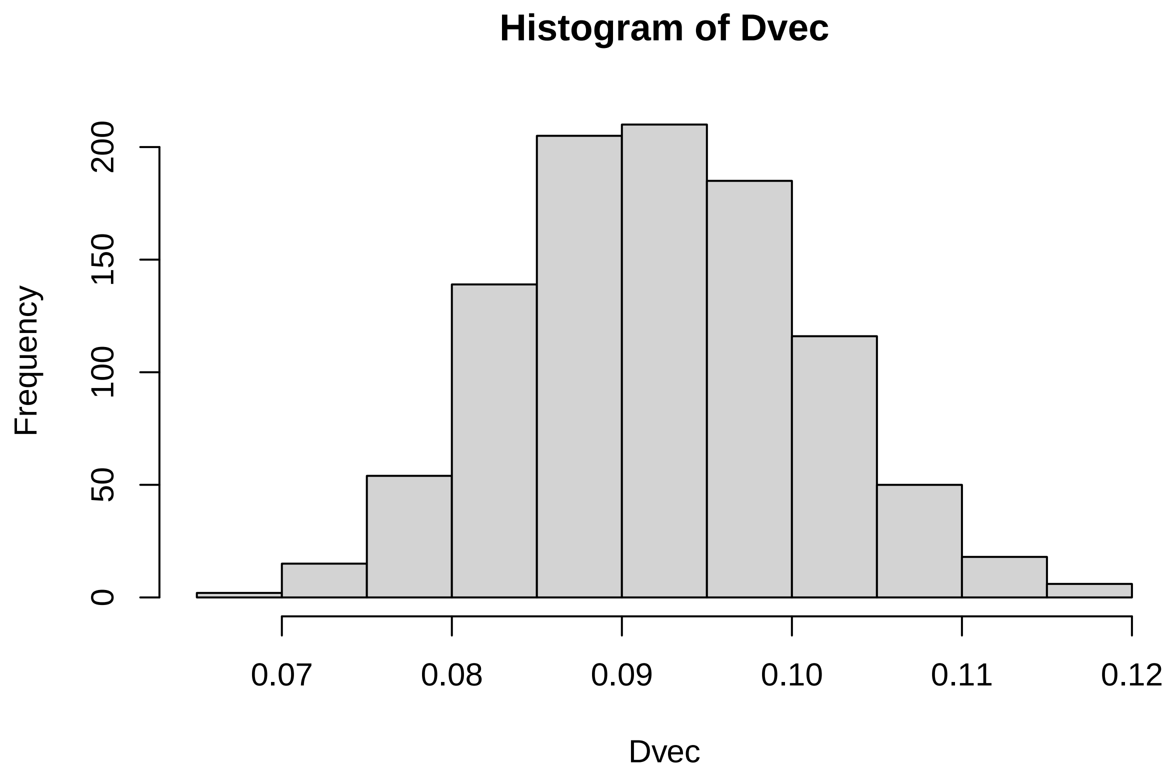
```
## [1] "DISTRIBUTION: geometric"
```



```
## $p_value
## [1] 0.291
```

```r
goodness_of_fit.func("binomial", input_data = rbinom(1000, 10, 0.5))
```

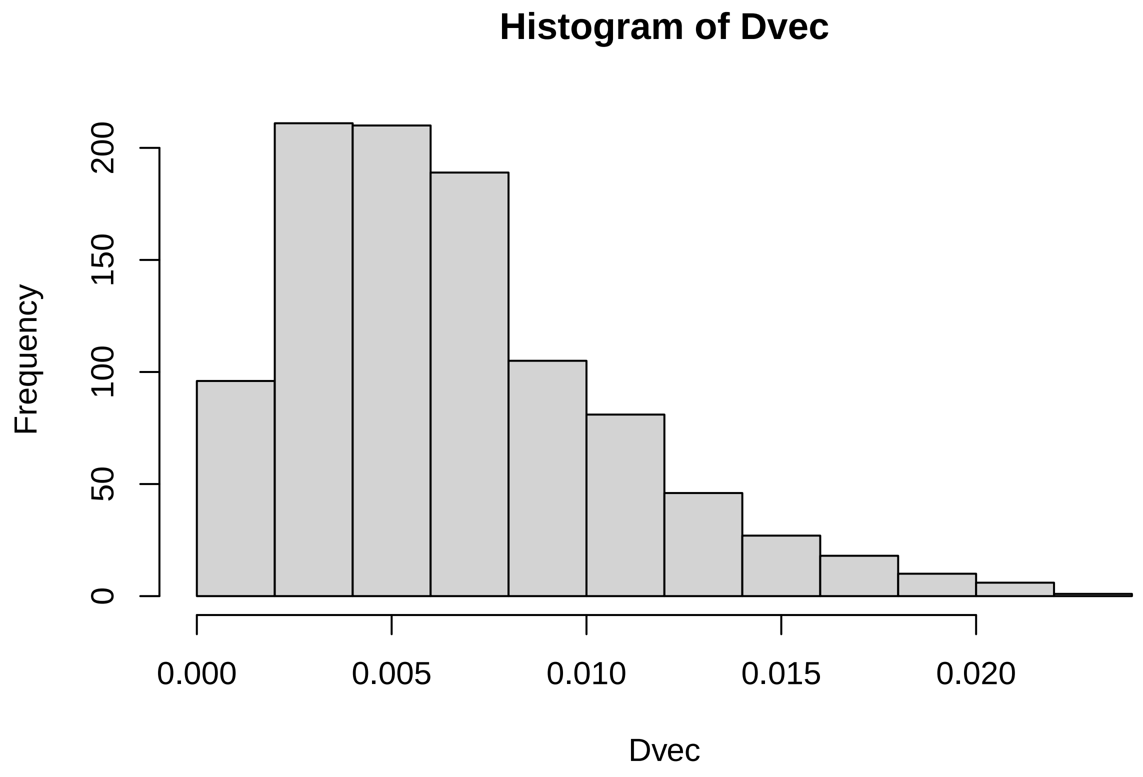
```
## [1] "DISTRIBUTION: binomial"
```



```
## $p_value
## [1] 0.063
```

```
goodness_of_fit.func("poisson", input_data = rpois(1000, 0.5))
```

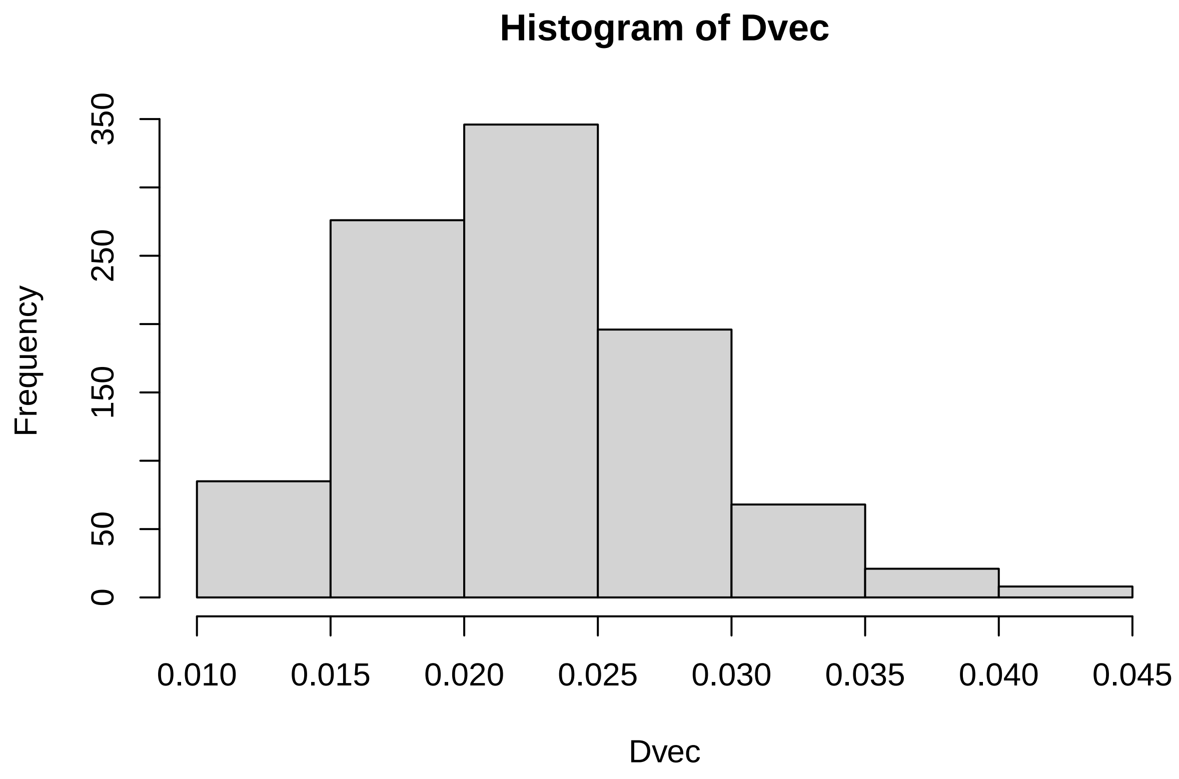
```
## [1] "DISTRIBUTION: poisson"
```



```
## $p_value
## [1] 0.483
```

```r
goodness_of_fit.func("normal", input_data = rnorm(1000, 0, 1))
```

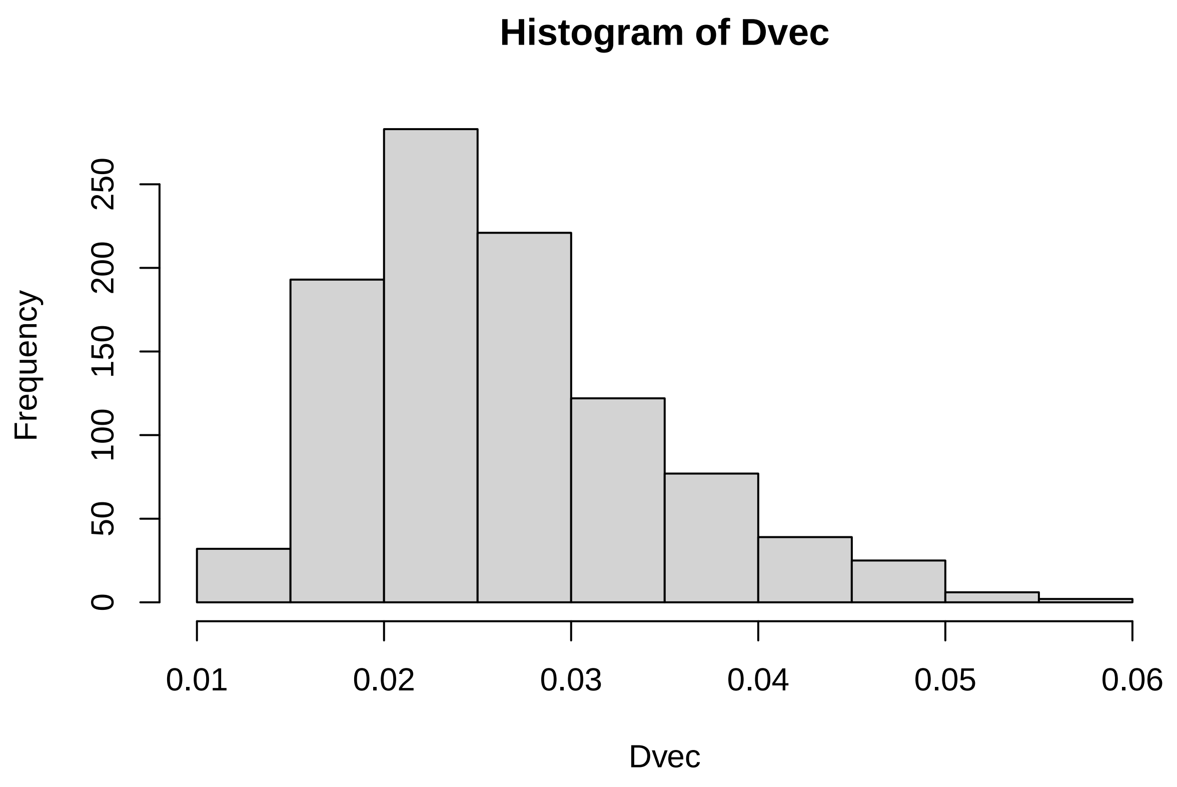
```
## [1] "DISTRIBUTION: normal"
```



```
## $p_value
## [1] 0.639
```

```r
goodness_of_fit.func("uniform", input_data = runif(1000, 0, 100))
```

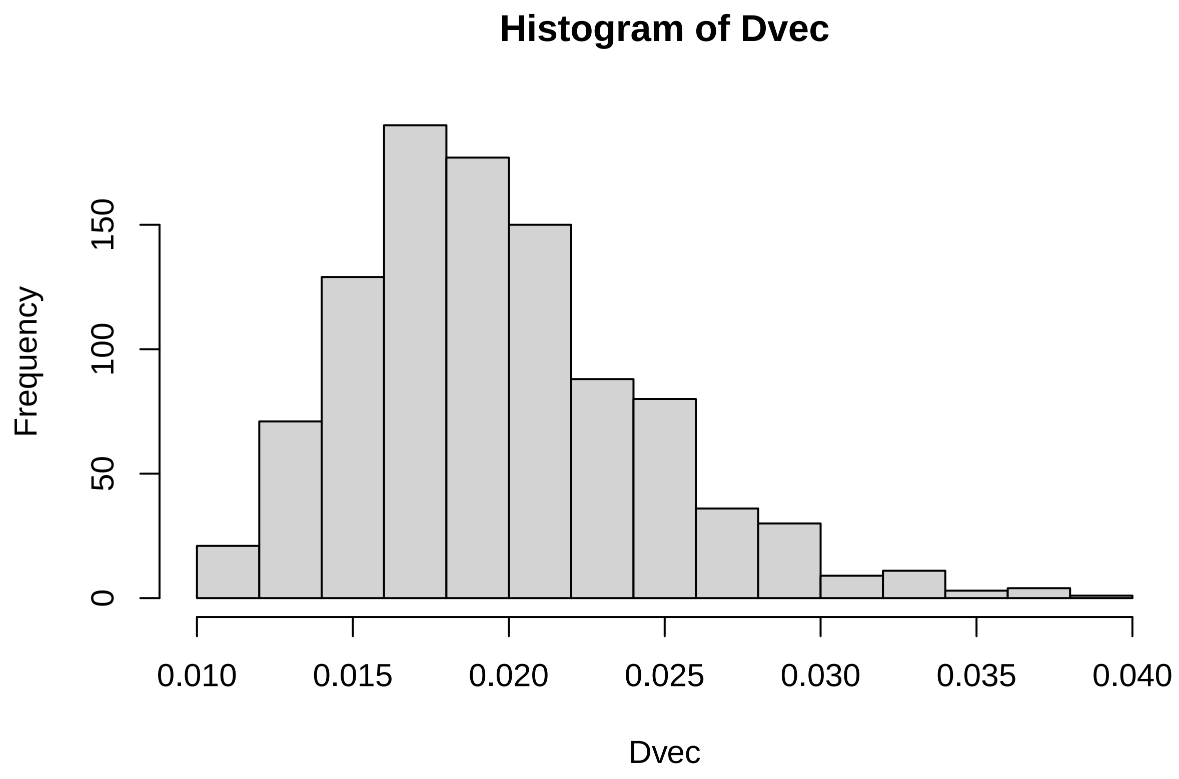
```
## [1] "DISTRIBUTION: uniform"
```



```
## $p_value
## [1] 0.636
```

```r
goodness_of_fit.func("gamma", input_data = rgamma(1000, shape = 5, scale = 20))
```

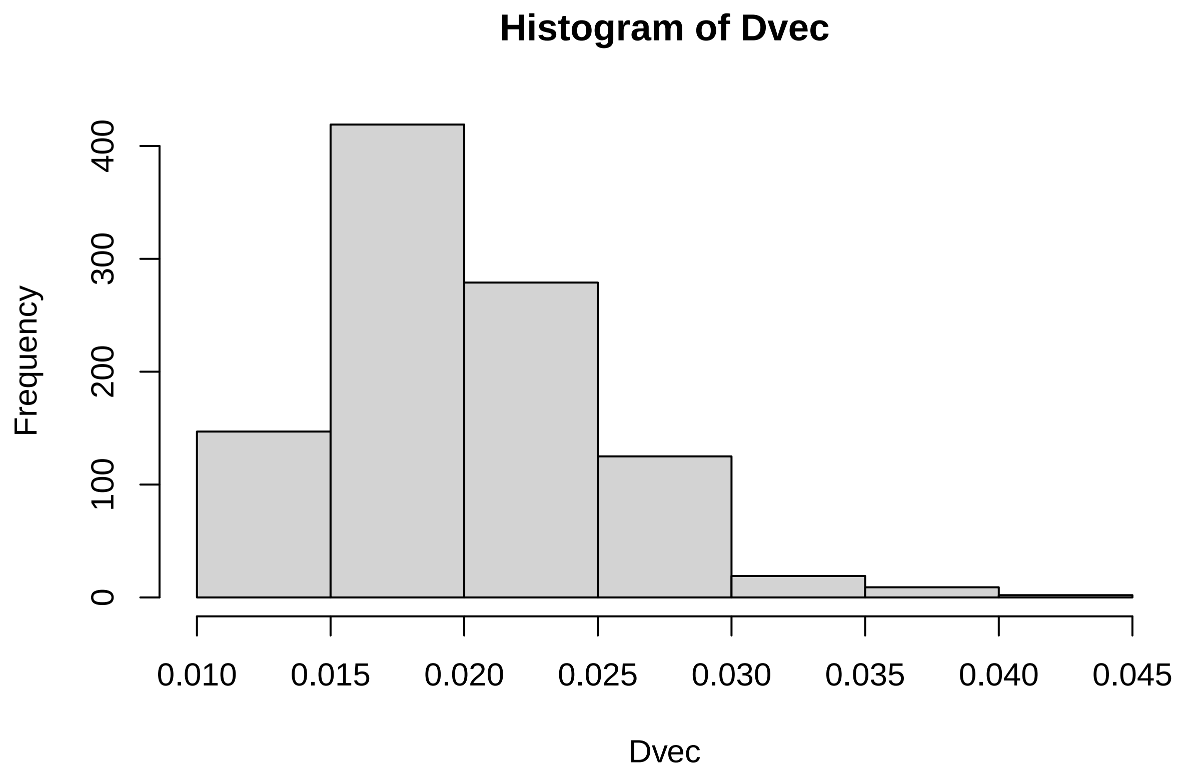
```
## [1] "DISTRIBUTION: gamma"
```



```
## $p_value
## [1] 0.262
```

```r
goodness_of_fit.func("beta", input_data = rbeta(1000, shape1 = 4.2, shape2 = 2.7))
```

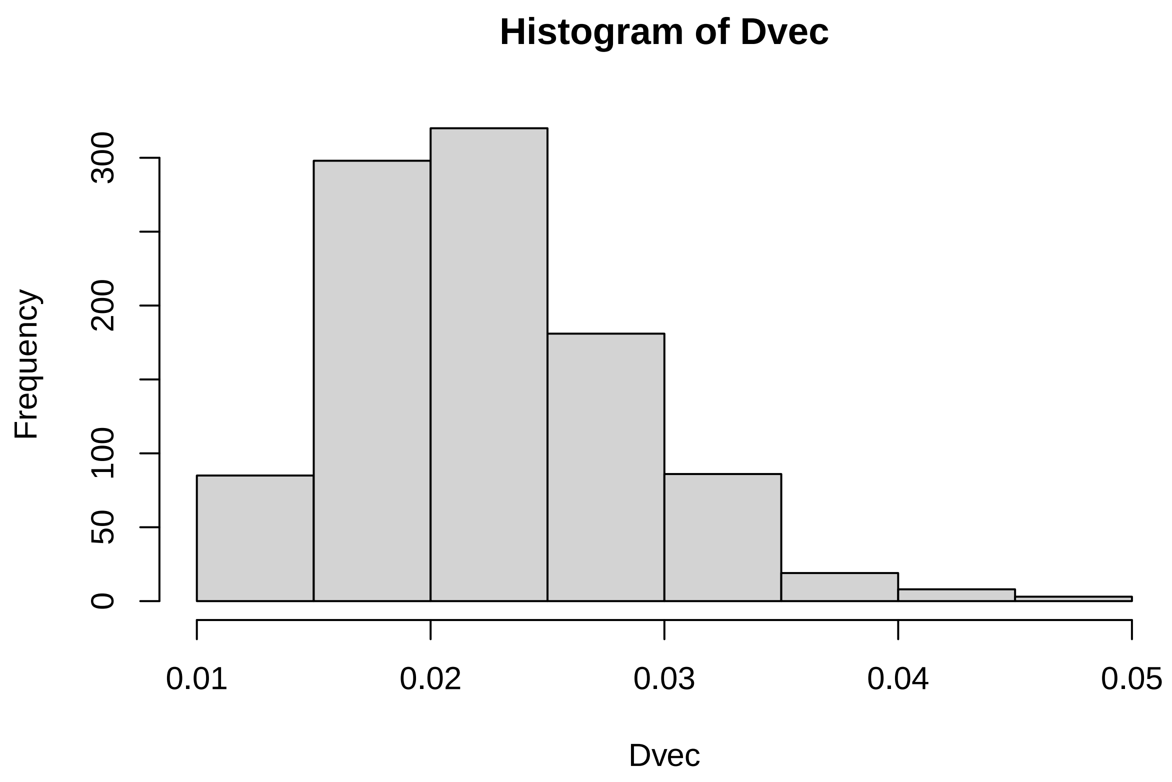
```
## [1] "DISTRIBUTION: beta"
```



```
## $p_value
## [1] 0.689
```

```r
goodness_of_fit.func("exponential", input_data = rexp(1000, 2))
```

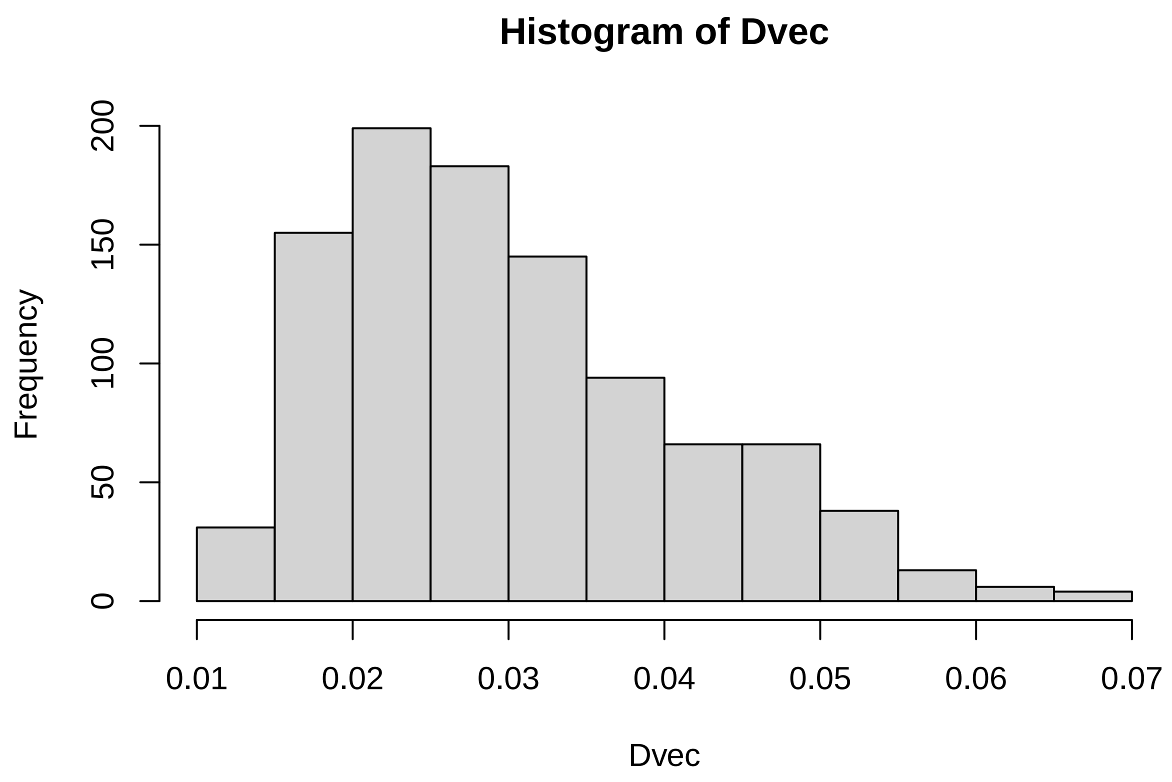
```
## [1] "DISTRIBUTION: exponential"
```



```
## $p_value
## [1] 0.409
```

```
goodness_of_fit.func("chi_square", input_data = rchisq(1000, 2))
```

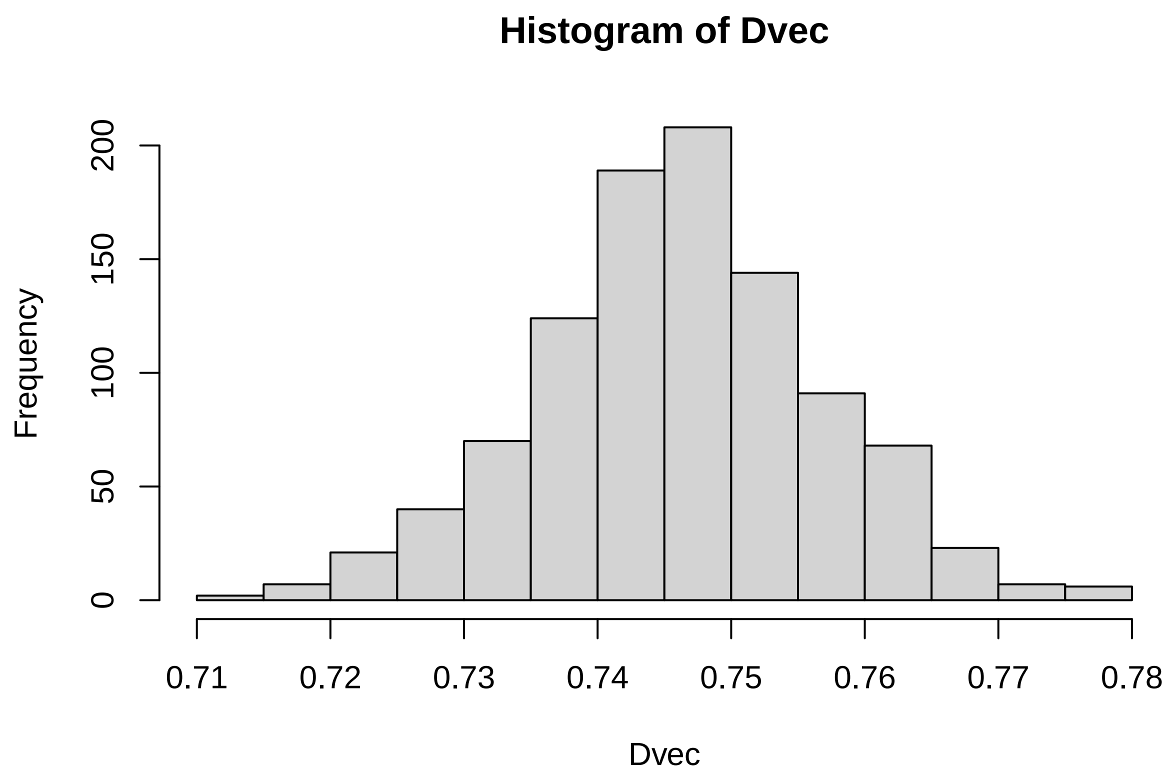
```
## [1] "DISTRIBUTION: chi_square"
```



```
## $p_value
## [1] 0.238
```

```r
p = c(0.15,0.05,0.4,0.1,0.3)
input_data = rmultinom(10000,size=5,p)
goodness_of_fit.func("multinomial", input_data = input_data)
```

```
## [1] "DISTRIBUTION: multinomial"
```



```
## $p_value
## [1] 0.946
```

Results:

Below is the table that shows the p-value of each distribution.

Distribution	$P-Value$
Bernoulli	1
Geometric	0.291
Binomial	0.063
Poisson	0.483
Normal	0.639
Uniform	0.636
Gamma	0.262
Beta	0.689
Exponential	0.409
Chi-Square	0.238
Multinomial	0.946

Conclusion:

Using $\alpha = 0.5$ as our p-value to test against each distribution. We notice that we reject the null hypothesis for the following distributions: Geometric, Binomial, Poisson, Gamma, Exponential, Chi-Squared. And we fail to reject the null hypothesis for the following distributions: Bernoulli, Normal, Uniform, Beta, and Multinomial. A p-value of one or close to one, as we see in some of our distributions, means the two groups we are testing are near identical.