Inferential Statistics Assignment

Costalonga Andrea

Introduction

I went through the first and the third assignment, here reported in this order. This report has been created using r markdown and latex.

1. First Assignment

First of all I've defined some constants and a function to make my code clearer and understandable. I've also included some libraries that I've used in this part of the assignment.

```
library(plotrix)
library(rmarkdown)
set.seed(29)

#tau function
tau <- function(theta){
    out <- theta/(1-theta)
    out2 <- log(out)
    return(out2)
}

N = 1e5 #numer of iterations
theta0 = 2/3 ##Neg bin success probability
r = 10 ##Neg bin size param
n <- c(10,50) ## Sample sizes
alpha = 0.05 ## Wald c.i. .95
```

In this segment I've created a 2*N matrix with N = 10^5 used to collect some results in order to simulate the distribution of $\hat{\tau}$. Every element is obtained evaluating $\hat{\theta}$ with the function *tau*. $\hat{\theta}$ is defined as

$$\hat{\theta} = \frac{r * n}{n * (r + \bar{\mathbf{y}})}$$

where r is the *index* parameter of the negative binomial, n is the size of the r. ve. and $\bar{\mathbf{y}}$ is the sample mean. Those results were obtained finding the maximum of the log likelihood function $l(\hat{\theta})$.

```
sim.N.tau <-matrix(NA, nrow = N, ncol = 2)
for(i in 1:N){
    xnegbin10 <- rnbinom(n=10,size=10,prob=theta0)
    xnegbin50 <- rnbinom(n=50,size=10,prob=theta0)
    sim.N.tau[i,1] <- tau(r*n[1]/(r*n[1]+sum(xnegbin10)))
    sim.N.tau[i,2] <- tau(r*n[2]/(r*n[2]+sum(xnegbin50)))
}
```

In the next cell I've managed to calculate what would have been the true value of mean and variance of the distribution of $\hat{\tau}$. In order to do so I've calculated the variance of $\hat{\theta}$ with Cramér-Rao's theorem, knowing that the Maximum Likelihood Estimator is asymptotically efficient. In formulas:

$$Var(\hat{\theta}) = I_{\mathrm{n}}(\theta)^{-1}$$

where $I_{\mathrm{n}}(\theta)$ is the Fisher information

$$I_{\mathrm{n}}(\theta) = n * I(\theta)$$

knowing that our r. ve. is built from $Y_1, ..., Y_{\mathrm{n}}$ $i.i.d.$ variables and

$$I(\theta) = \frac{r}{\theta^2 * (1-\theta)}$$

which is obtained by the definition of Fisher information. In the end we have:

$$Var(\hat{\theta}) = \frac{\theta^2 * (1-\theta)}{r * n}$$

Then I've used the Delta Method to obtain the variance and the mean of the gaussian curve that represent our ideal density (and distribution), and I've got the following results:

$\hat{\tau}$'s variance:

$$var(\hat{\tau}) = (\frac{dtau(\theta)}{d\theta})^2 * Var(\hat{\theta}) = (\frac{1}{\theta * (1-\theta)})^2 * \frac{\theta^2 * (1-\theta)}{r * n} = \frac{1}{(1-\theta) * r * n}$$

$\hat{\tau}$'s mean:

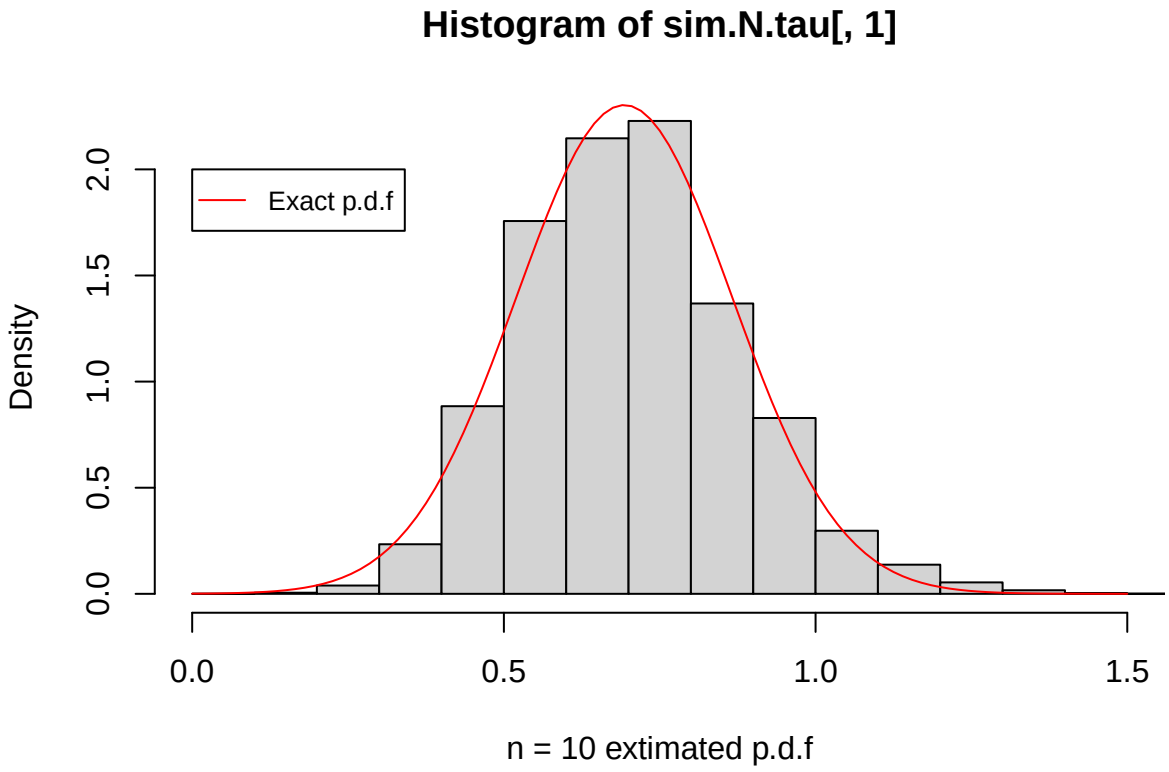
$$E[\hat{\tau}] = tau(\theta)$$

Evaluating variance and mean in θ_0 bring us to the real p.d.f. of $\hat{\tau}$:

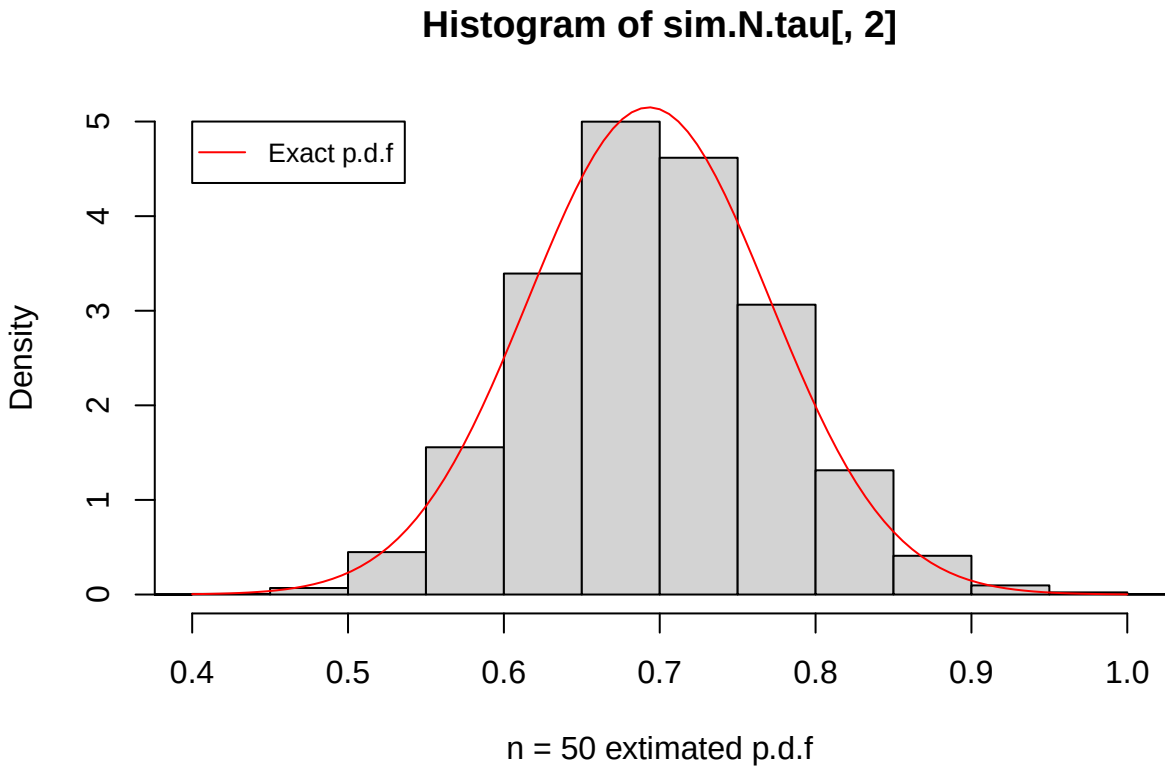
```
true_mean = (tau(theta0))
#True variance of tau hat (n=10, n=50)
true_var10_tau <- 1/(r*n[1]*(1-theta0))
true_var50_tau <- 1/(r*n[2]*(1-theta0))
```

I've made a couple of plot to see if our theoretical results are supported by our simulations.

Plot for n = 10:



Plot for n = 50:



We can observe that the extimated p.d.f. fits perfecly the exact p.d.f.. I've runned the simulation with just 10^5 iterations but to obtain something that fits even better I could have runned with a larger N with the perks of having a higher processing time.

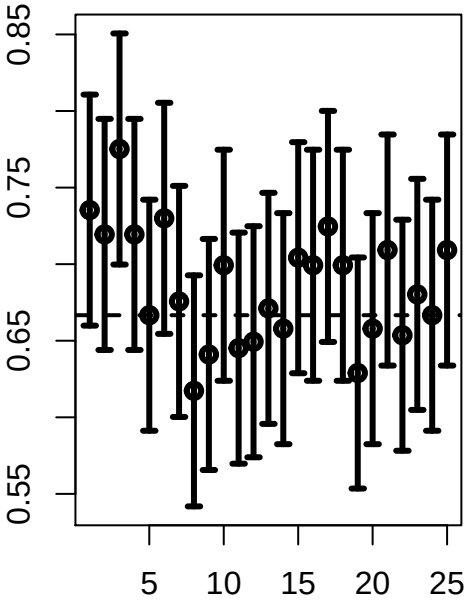
Coverage Probability, Wald C.I.

In the following lines I've made a simulation to study the coverage probability of the .95 Wald confidence interval for τ and θ

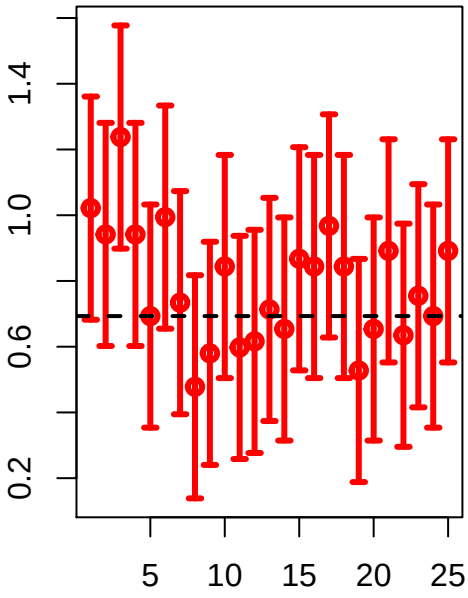
```
true_var10_th <- (theta0^2*(1-theta0))/(r*n[1])
true_var50_th <- (theta0^2*(1-theta0))/(r*n[2])

sim.N.CI_the <-matrix(NA, nrow = N, ncol = 4)
sim.N.CI_tau <-matrix(NA, nrow = N, ncol = 4)
for(i in 1:N){
    xnegbin10 <- rnbinom(n=n[1],size=r,prob=theta0)
    xnegbin50 <- rnbinom(n=n[2],size=r,prob=theta0)
    sim.N.CI_the[i,1:2] <- r*n[1]/(r*n[1]+sum(xnegbin10))
    sim.N.CI_the[i,1:2] <- sim.N.CI_the[i,1:2] + c(-1,1)*qnorm(p = alpha/2,
         lower.tail = FALSE)*sqrt(true_var10_th)
    sim.N.CI_the[i,3:4] <- r*n[2]/(r*n[2]+sum(xnegbin50))
    sim.N.CI_the[i,3:4] <- sim.N.CI_the[i,3:4] + c(-1,1)*qnorm(p = alpha/2,
         lower.tail = FALSE)*sqrt(true_var50_th)
    sim.N.CI_tau[i,1:2] <- tau(r*n[1]/(r*n[1]+sum(xnegbin10)))
    sim.N.CI_tau[i,1:2] <- sim.N.CI_tau[i,1:2] + c(-1,1)*qnorm(p = alpha/2,
        lower.tail = FALSE)*sqrt(true_var10_tau)
    sim.N.CI_tau[i,3:4] <- tau(r*n[2]/(r*n[2]+sum(xnegbin50)))
    sim.N.CI_tau[i,3:4] <- sim.N.CI_tau[i,3:4] + c(-1,1)*qnorm(p = alpha/2,
        lower.tail = FALSE)*sqrt(true_var50_tau)
}
theta10.inside <- apply(sim.N.CI_the[,1:2], MARGIN = 1,
    function(x)ifelse(theta0>=x[1]&theta0<=x[2],1,0))
theta50.inside <- apply(sim.N.CI_the[,3:4], MARGIN = 1,
    function(x)ifelse(theta0>=x[1]&theta0<=x[2],1,0))
tau10.inside <- apply(sim.N.CI_tau[,1:2], MARGIN = 1,
    function(x)ifelse(true_mean>=x[1]&true_mean<=x[2],1,0))
tau50.inside <- apply(sim.N.CI_tau[,3:4], MARGIN = 1,
    function(x)ifelse(true_mean>=x[1]&true_mean<=x[2],1,0))
```

In the plots below I've reported the first 25 confidence intervals for θ and τ

Confidence interval plot for θ and τ with n = 10:

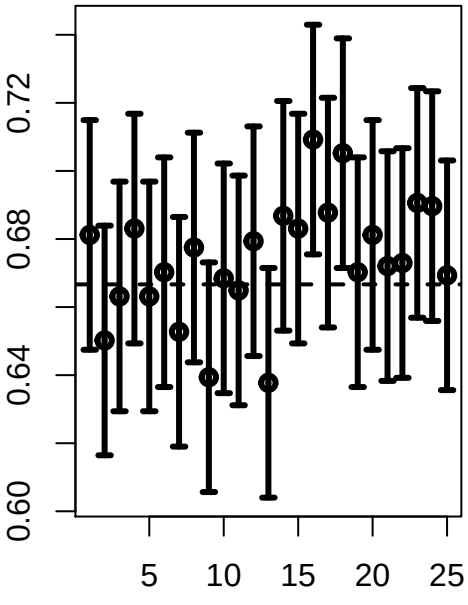


Observed C.I. for theta (n=10)

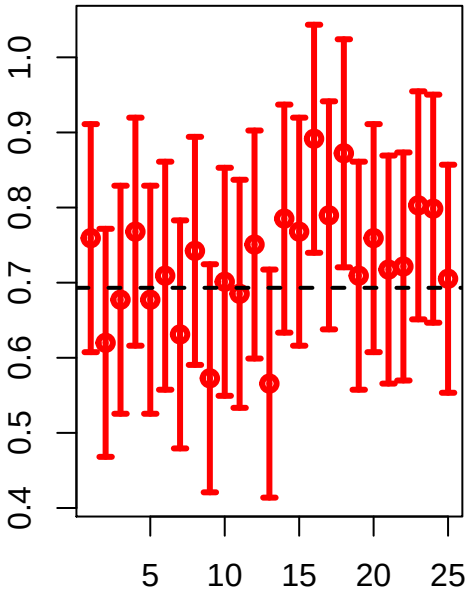


Observed C.I. for tau (n=10)

Confidence interval plot for θ and τ with n = 50:



Observed C.I. for theta (n=50)



Observed C.I. for tau (n=50)

Here's some results:

```
mean(theta10.inside) #Confidence probability for theta n=10
```

```
## [1] 0.95415
```

```
mean(theta50.inside)  #Confidence probability for theta n=50
```

```
## [1] 0.94976
```

```
mean(tau10.inside)  #Confidence probability for tau n=10
```

```
## [1] 0.94768
```

```
mean(tau50.inside)  #Confidence probability for tau n=50
```

```
## [1] 0.94857
```

We can notice that in every scenario the confidence probability is close to .95 (even better some times). It's easy to state that our confidence intervals for $\hat{\theta}$ and $\hat{\tau}$ have the same behaviour.

2. Third Assignment

I've defined some recurrent values and settings below:

```
set.seed(29)
n = c(10,15,10,15) #Number of elements for every col
var = c(1,1,1,1) #Variance for every col
mu = c(0,0,0,0) #Mean for every col
k=4 #Number of cols
```

In order to estimate the distribution of the F-Statistics for ANOVA, we need to calculate SSR and SSE for every trial, where SSR and SSE are defined as follows:

$$SSR := \sum_{j=1}^{k}(\sum_{i=1}^{n_j}(\bar{Y}_{*j} - \bar{Y})^2)$$

$$SSE := \sum_{j=1}^{k}(\sum_{i=1}^{n_j}(Y_{ij} - \bar{Y}_{*j})^2)$$

where $\bar{Y}_{*j}$ is the observed mean for treatment j, $\bar{Y}$ is the overall mean and Y_{ij} is the observed sample ij.

For every iteration l, after computing SSR_l and SSE_l, I'm going to store the value

$$F_{obs}[l] := \frac{\frac{SSR_l}{k-1}}{\frac{SSE_l}{n-k}}$$

in the array called $sim.N.F$.

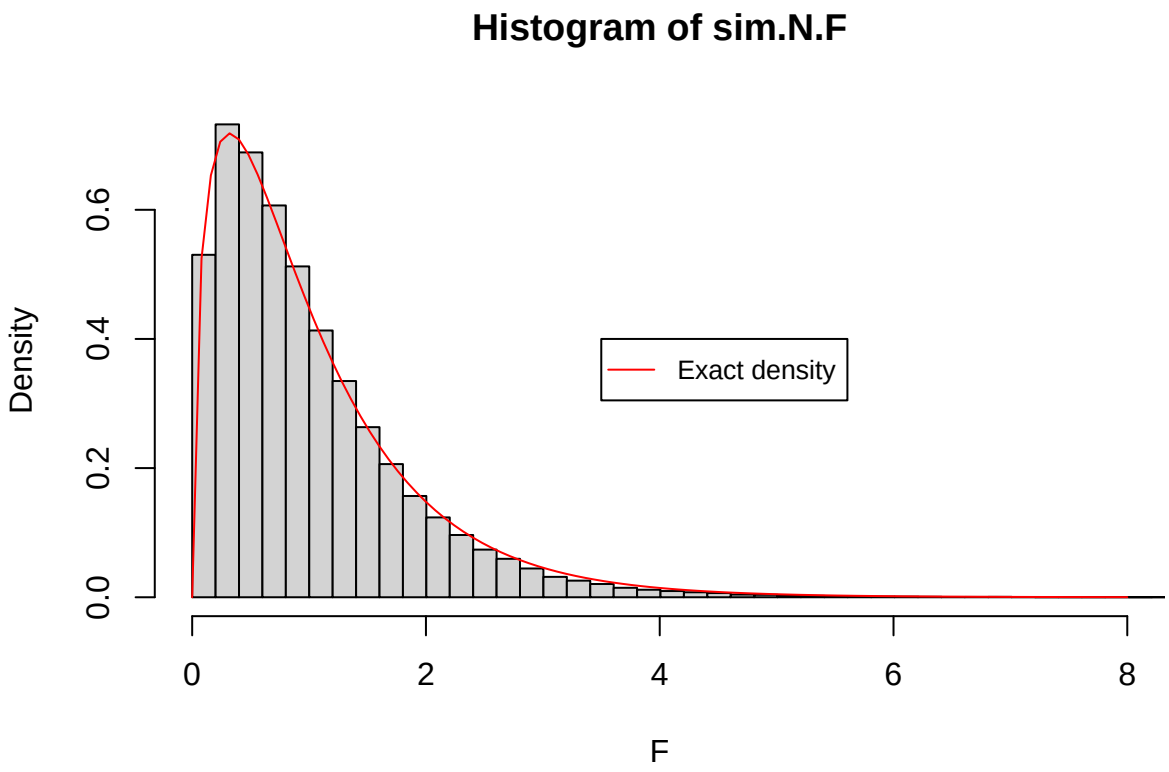
```
sim.N.F <- rep(0,N) #initialized the array
for(l in 1:N){
    #defined the observed sample as follows
    #used var[1] as variance as every sample comes from N(0,1)
    y <- rnorm(sum(n), mean = 0, sd = sqrt(var[1]))
    #Treatment j mean
    estMuj <- c(sum(y[1:10])/n[1],
            sum(y[11:25])/n[2],
            sum(y[26:35])/n[3],
            sum(y[36:50])/n[4])
    #Overall mean
    estMu <- sum(y)/sum(n)
    SSR <- 0 #init SSR
    SSE <- 0 #init SSE
    for (i in 1:k){
        offset <- sum(n[1:(i-1)])
        for(j in 1:n[i]){
            SSR = SSR + (estMuj[i]-estMu)^2
```

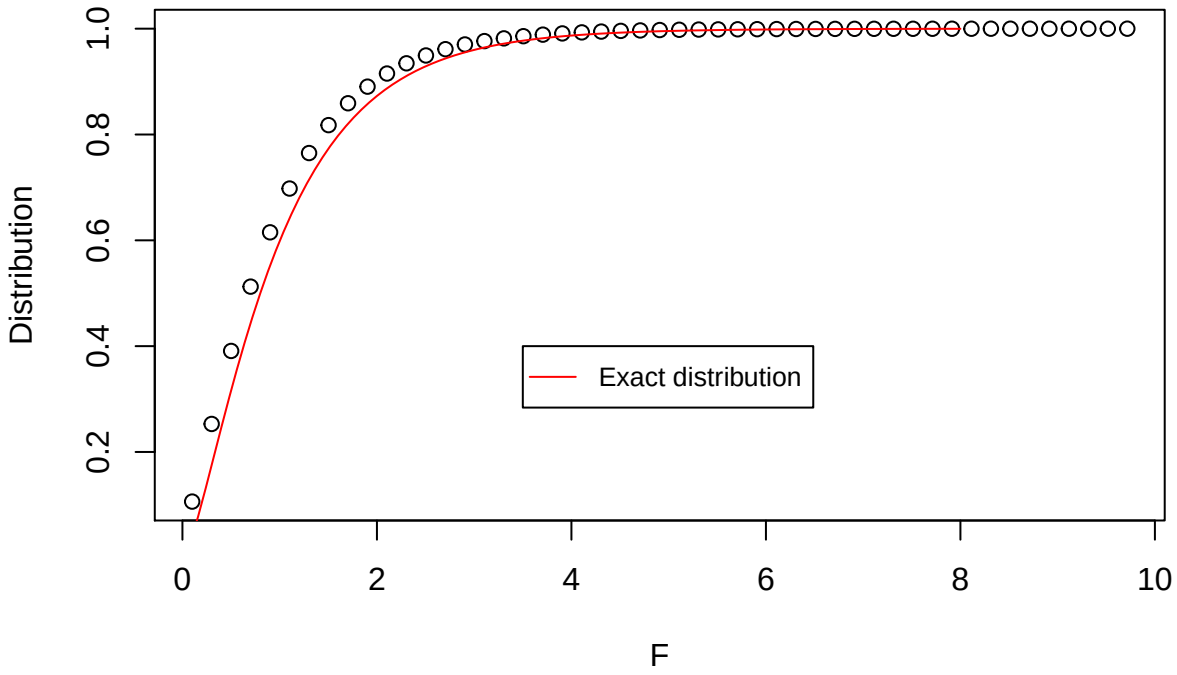
```
            SSE = SSE + (y[offset+j]-estMuj[i])^2
        }
    }
    sim.N.F[l] <- (SSR/(k-1))/(SSE/(sum(n)-k)) #value of F observed (Fobs)
}
```

From theory we know that

$$\frac{\frac{SSR}{k-1}}{\frac{SSE}{n-k}} \sim \frac{\chi_{\text{k-1}}{}^2}{\chi_{\text{n-k}}{}^2}$$

This new r.v. should follow a F distribution with parameters k-1 and n-k. In the plots below there's a comparison between our simulated sample density/distribution and the real density/distribution of F.





Now we are asked to change the value of the variance as follows:

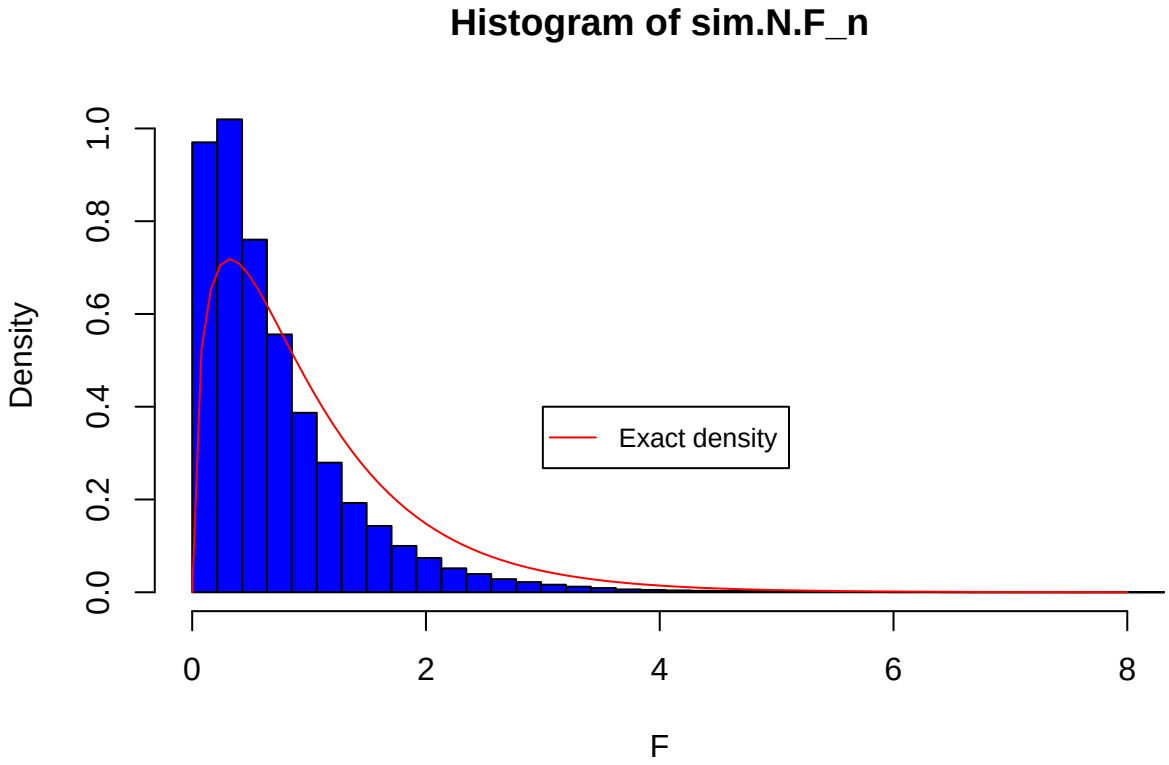
```
var <- c(1/2,2,1/2,2) #Vector of variances
```

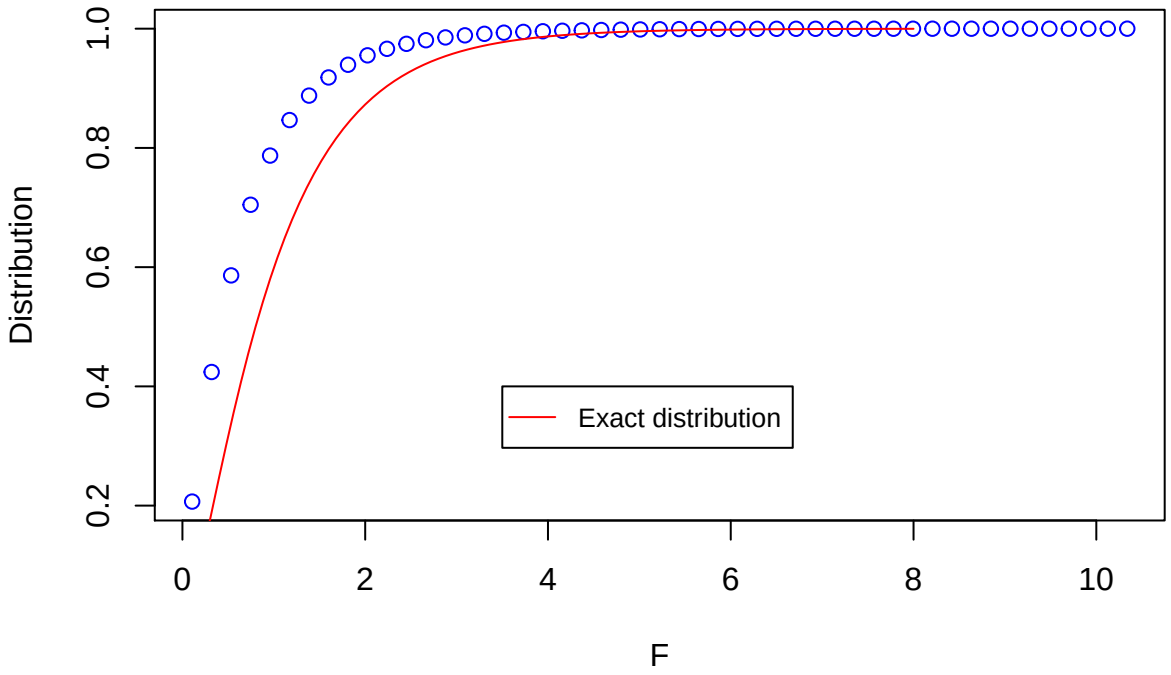
From theory we know that the ANOVA test works with $r.v.'s$ with the same variance, so I expect to see that our extimated p.d.f. won't fit the real p.d.f. as good as the previous simulation. In the next code lines I've re-executed the analysis with the changes I just made.

```
sim.N.F_n <- rep(0, N)
for(l in 1:N){
    offset <- 1
    y_n <- rep(0, 50)
    for (i in 1:k){
        y_n[offset:sum(n[1:i])]<-rnorm(n[i],
            mean = mu[i], sd = sqrt(var[i]))
        offset<- offset + n[i]
    }
    #Treatment j mean
    estMuj <- c(sum(y_n[1:10])/n[1],
            sum(y_n[11:25])/n[2],
            sum(y_n[26:35])/n[3],
            sum(y_n[36:50])/n[4])
    #Overall mean
    estMu <- sum(y_n)/sum(n)
    SSR <- 0
    SSE <- 0
    for (i in 1:k){
        offset <- sum(n[1:(i-1)])
```

```
        for(j in 1:n[i]){
            SSR = SSR + (estMuj[i]-estMu)^2
            SSE = SSE + (y_n[offset+j]-estMuj[i])^2
        }
    }
    #value of new F observed (Fobs)
    sim.N.F_n[l] <- (SSR/(k-1))/(SSE/(sum(n)-k))
}
```

Those are the plots obtained from the latest analysis:





We can observe that, with this changes in the variance array, our simulated p.d.f. for f-statistics does not follow the exact p.d.f.. As I've stated before, It's probably due to the fact that ANOVA works for r.v.'s with the same variance and this is not the case.