Simulating critical values: Power and size studies

Yifan Jin

29/06/2020

```
library(matlab)
library(MASS)
library(pracma)
```

```
set.seed(0)
# A simple regression model, just to see that matrix inversion works
betatrue=t(t(c(4,-9,7,0,-2)))
x=cbind(ones(100,1),matrix(rnorm(100*4),nrow = 100,ncol = 4))
y=x%*%betatrue+rnorm(100)
betahat1=solve(t(x)%*%x)%*%t(x)%*%y
```

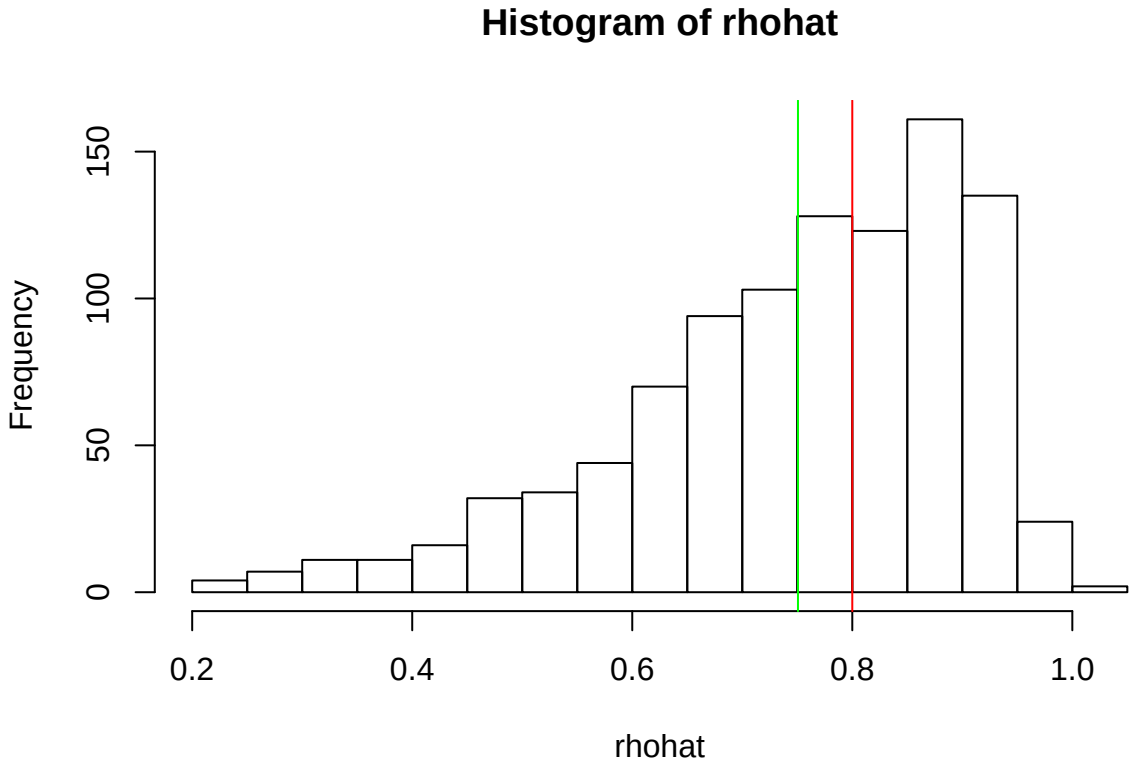
```
# ols estimator function
olsreg=function(y,x){
  betahat=mldivide(x,y)
  e=y-x%*%betahat
  n=size(x)[1]
  k=size(x)[2]
  s2=as.numeric((t(e)%*%e)/(n-k))
  varbeta=s2 * solve(t(x)%*%x)
  se_beta=sqrt(diag(varbeta))
  tstat=betahat/se_beta
  cbind(betahat,se_beta,tstat)
}
olsreg(y,x)
```

```
##                    se_beta
## [1,]  3.94306815 0.1113406  35.4144528
## [2,] -8.86636364 0.1292932 -68.5756562
## [3,]  6.93878350 0.1166066  59.5059094
## [4,]  0.03253265 0.1062259   0.3062591
## [5,] -1.88733568 0.1194938 -15.7944222
```

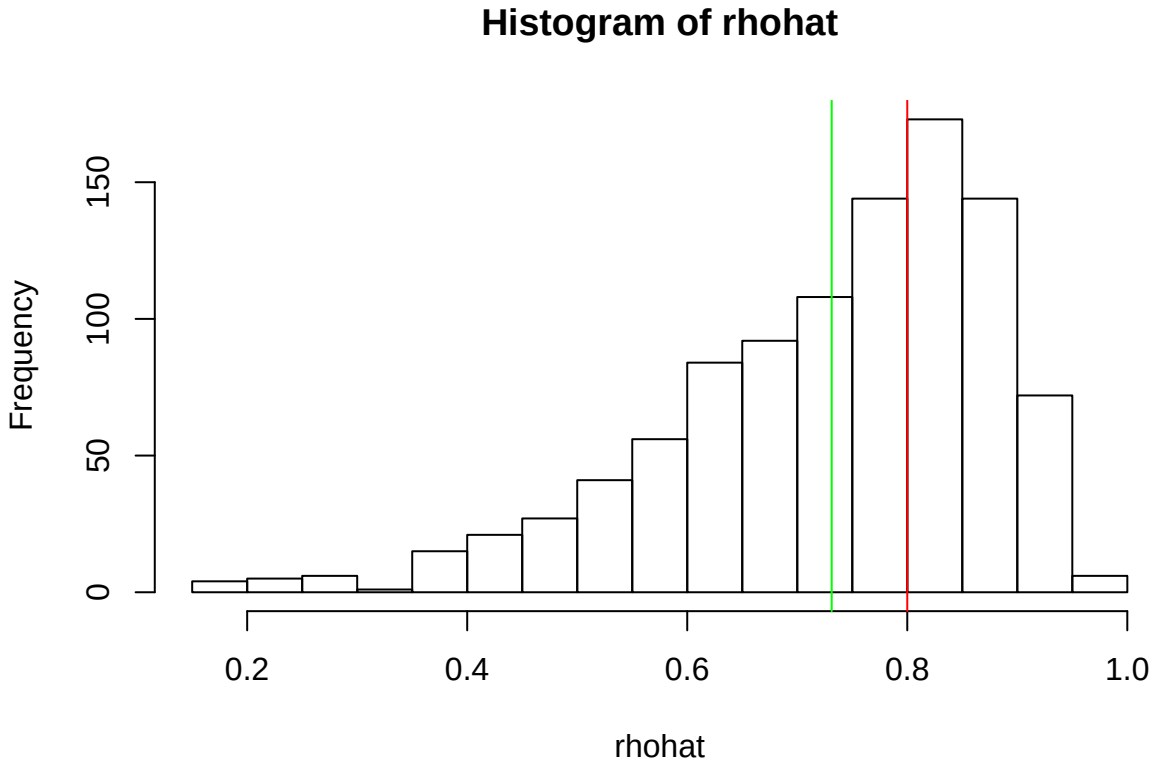
```
# stationary AR studies
ar1=function(rho,sigma,T,S){
  epsilon=sigma*matrix(rnorm(T+1*S),nrow=T+1,ncol=S,byrow=T)
  y=zeros(T+1,S)
  y[1,]=epsilon[1,]
  for(t in 1:T){
    y[t+1,]=rho*y[t,]+epsilon[t+1,]
  }
  rhohat=zeros(S,1)
  for(s in 1:S){
    rhohat[s]=mldivide(y[1:T,s],y[2:(T+1),s],T)
```

```
  }
  mrh=mean(rhohat)
  bias=mrh-rho
  variance=mean((rhohat-mrh)^2)
  MSE=mean((rhohat-rho)^2)
  print(c(bias,variance,MSE))
  hist(rhohat,breaks=20)
  abline(v=c(mrh,rho),col=c("green","red"))
}
for (rho in c(0.8,1)){
  for (T in c(25,50,100,200)){
    for (sigma in c(0.5,1,2)){
      c(print(c(rho,T,sigma)),ar1(rho,sigma,T,999))
    }
  }
}
```

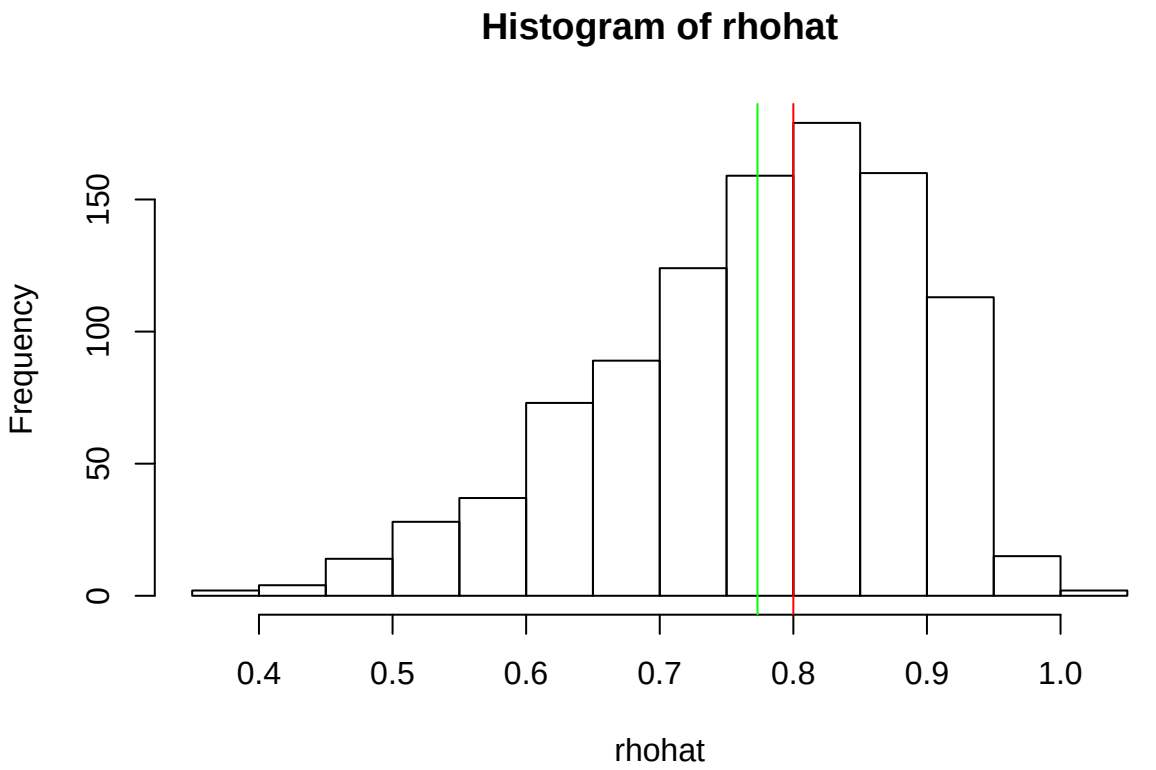
```
## [1]  0.8 25.0  0.5
## [1] -0.04937280  0.02360556  0.02604323
```



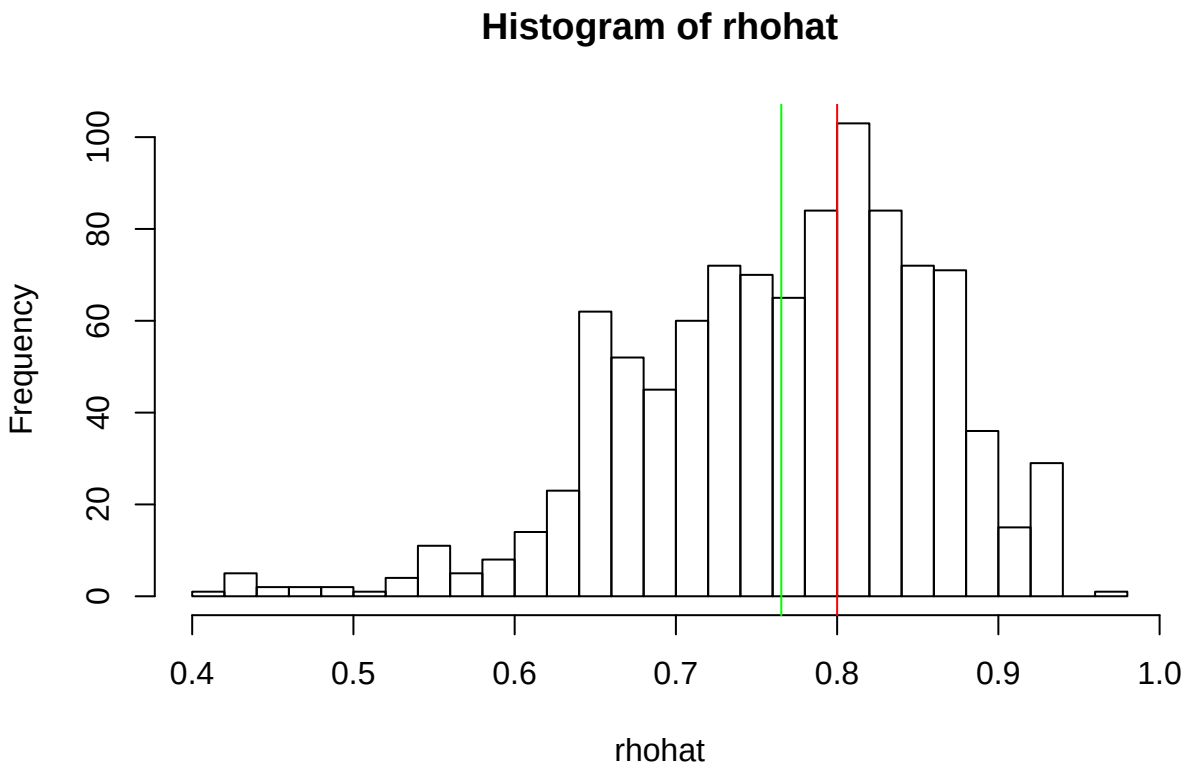

```
## [1]  0.8 25.0  1.0
## [1] -0.06871279  0.02176439  0.02648584
```



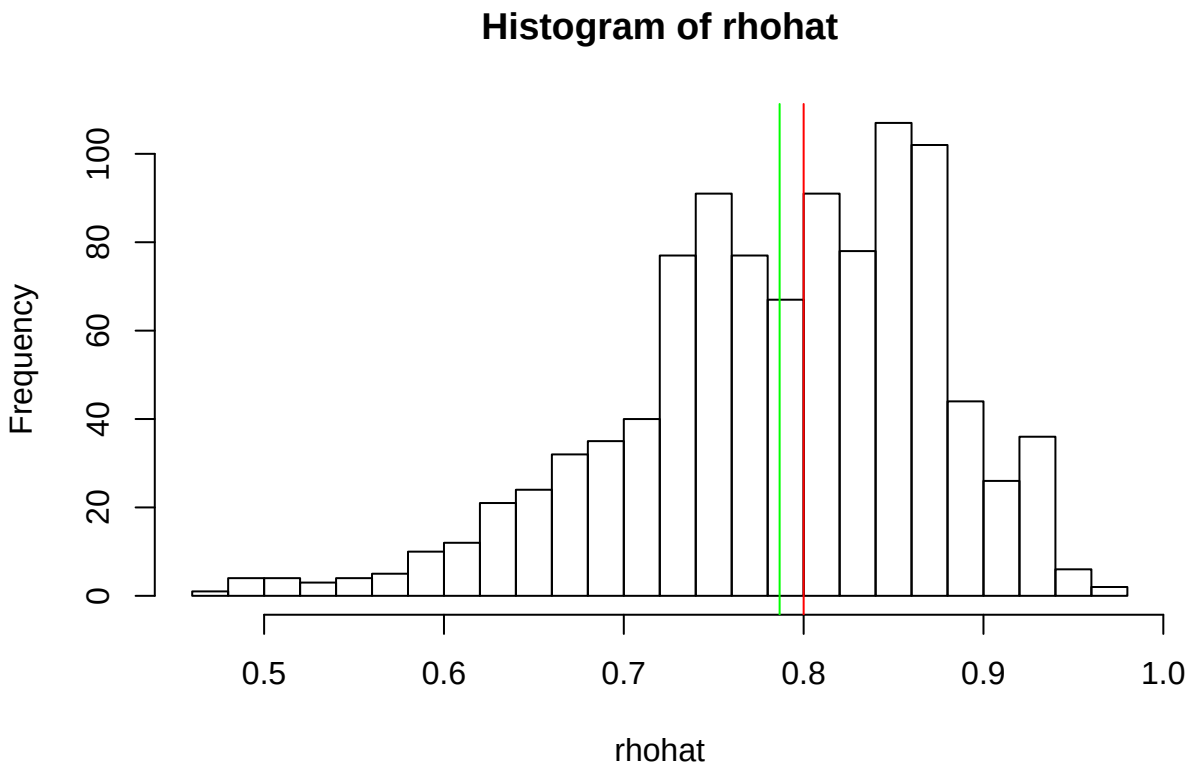

```
## [1]  0.8 25.0  2.0
## [1] -0.02683295  0.01308377  0.01380378
```



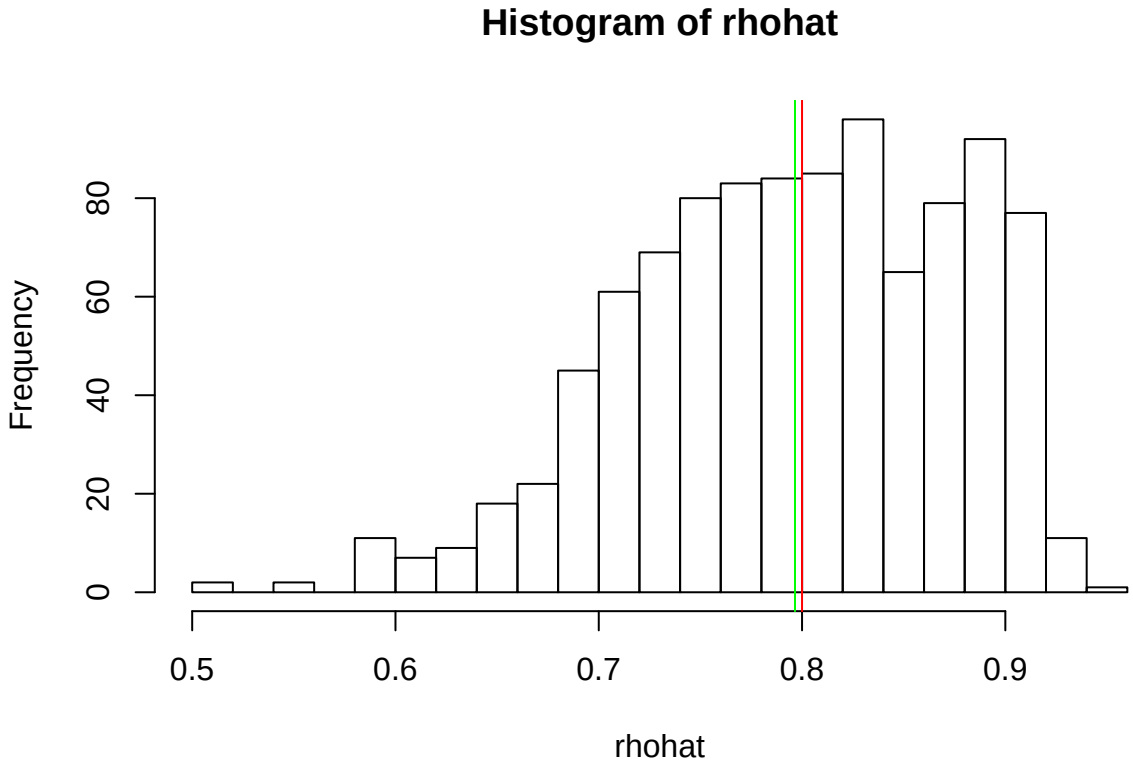

```
## [1]  0.8 50.0  0.5
## [1] -0.034625899  0.008667557  0.009866510
```



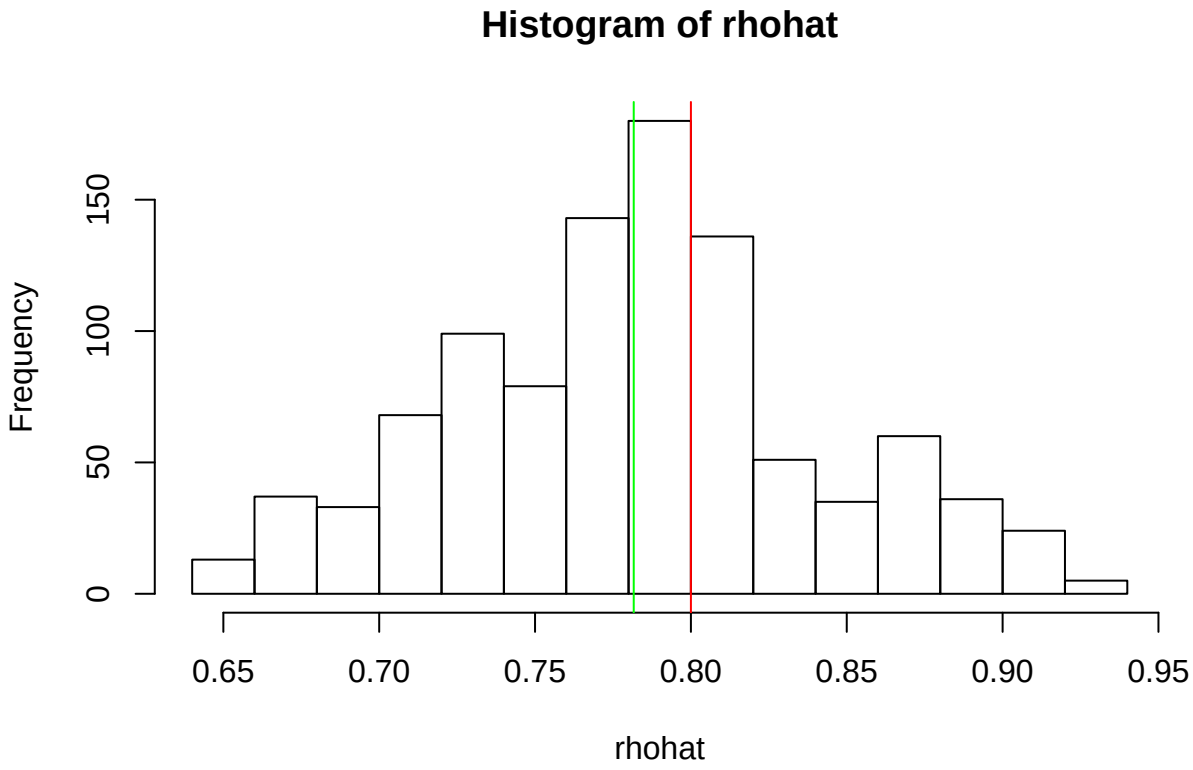

```
## [1]  0.8 50.0  1.0
## [1] -0.013364211  0.007685574  0.007864176
```



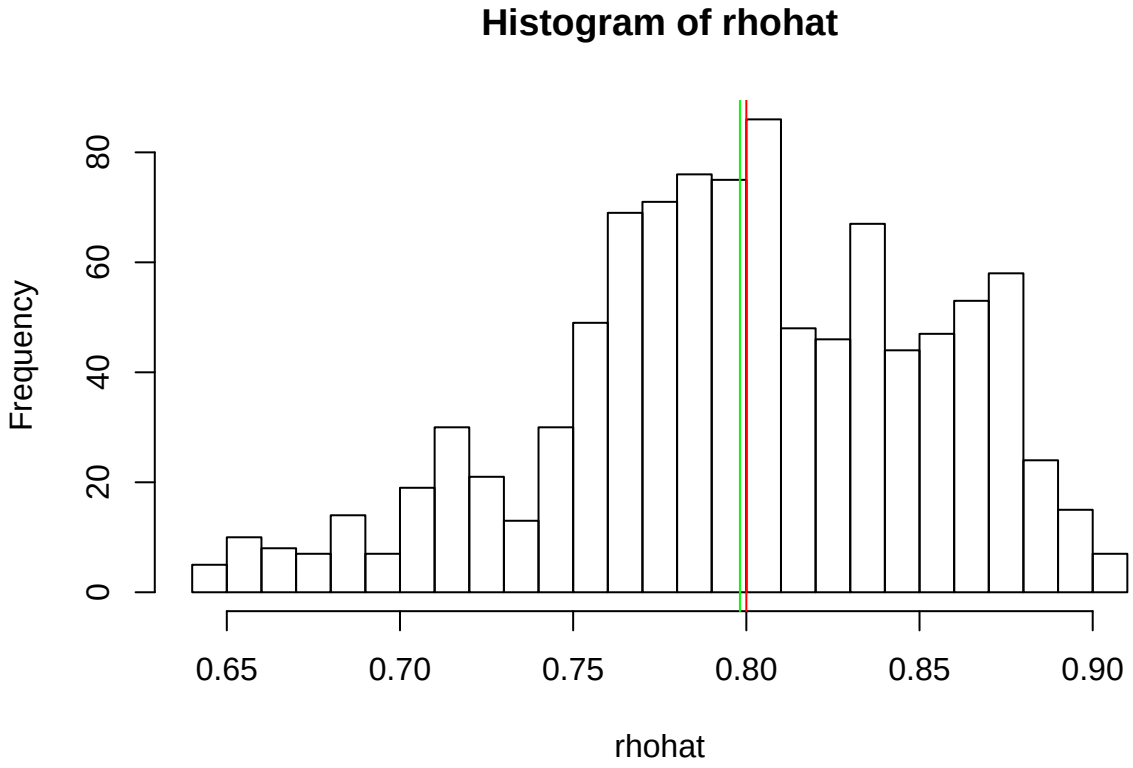

```
## [1]  0.8 50.0  2.0
## [1] -0.003484948  0.006158243  0.006170388
```



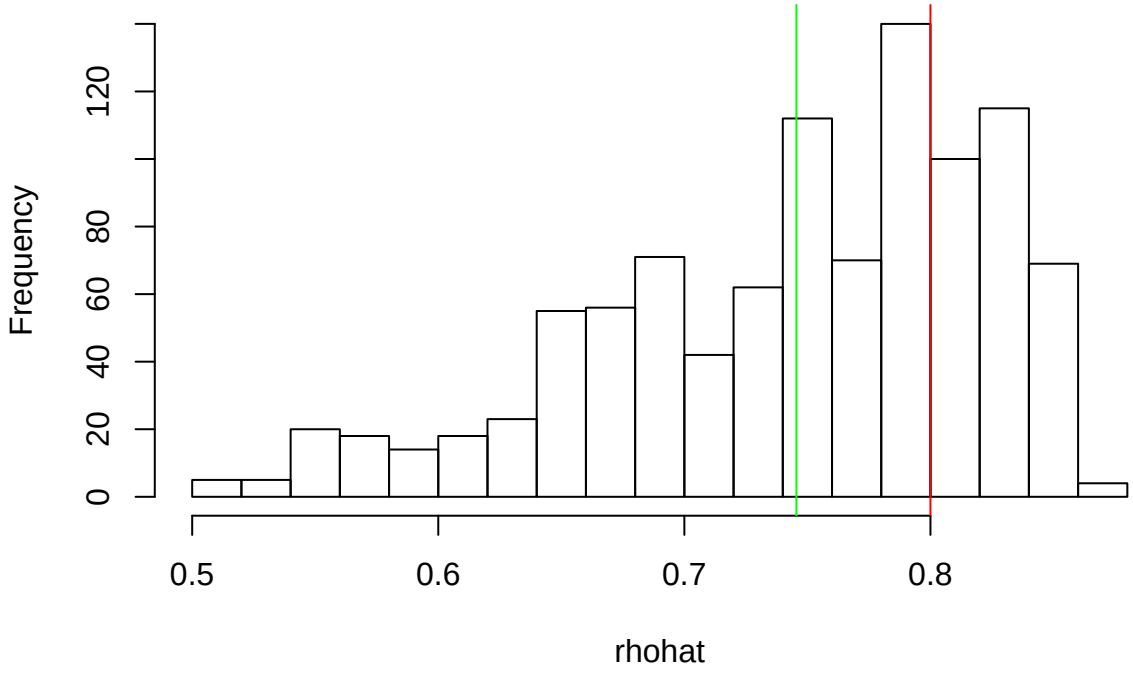

```
## [1]   0.8 100.0   0.5
## [1] -0.018341999  0.003437982  0.003774410
```



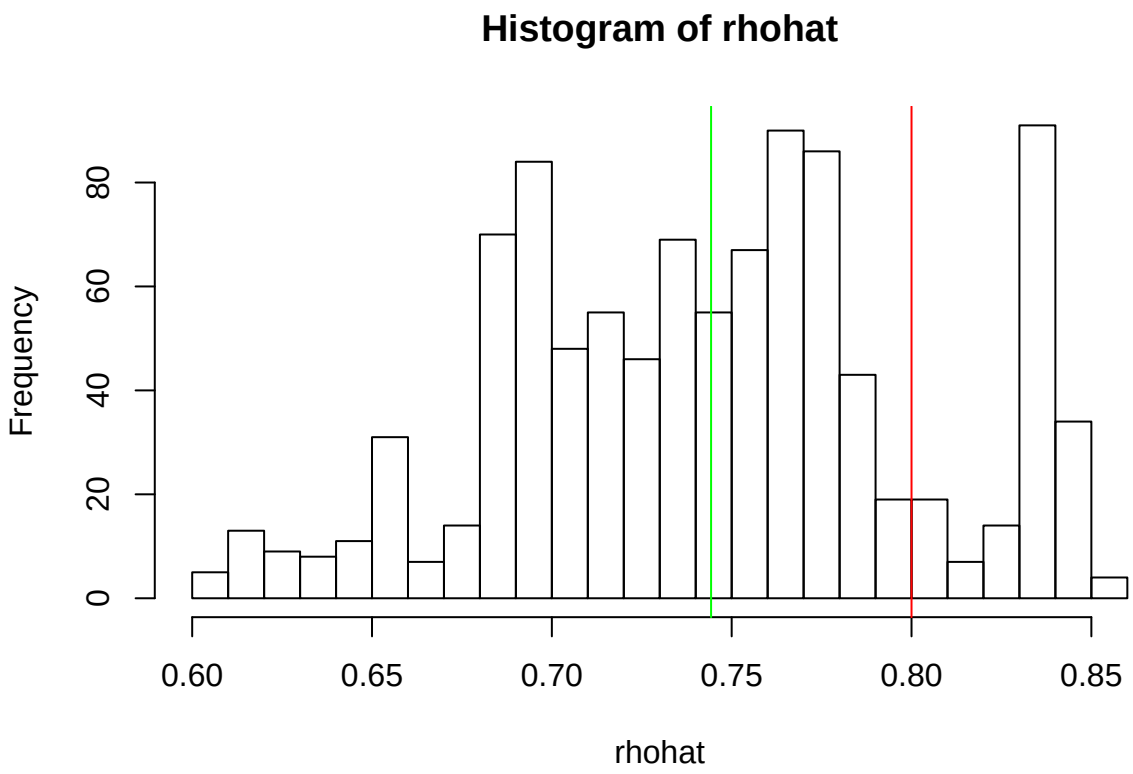

```
## [1]   0.8 100.0   1.0
## [1] -0.001857390  0.003021104  0.003024553
```



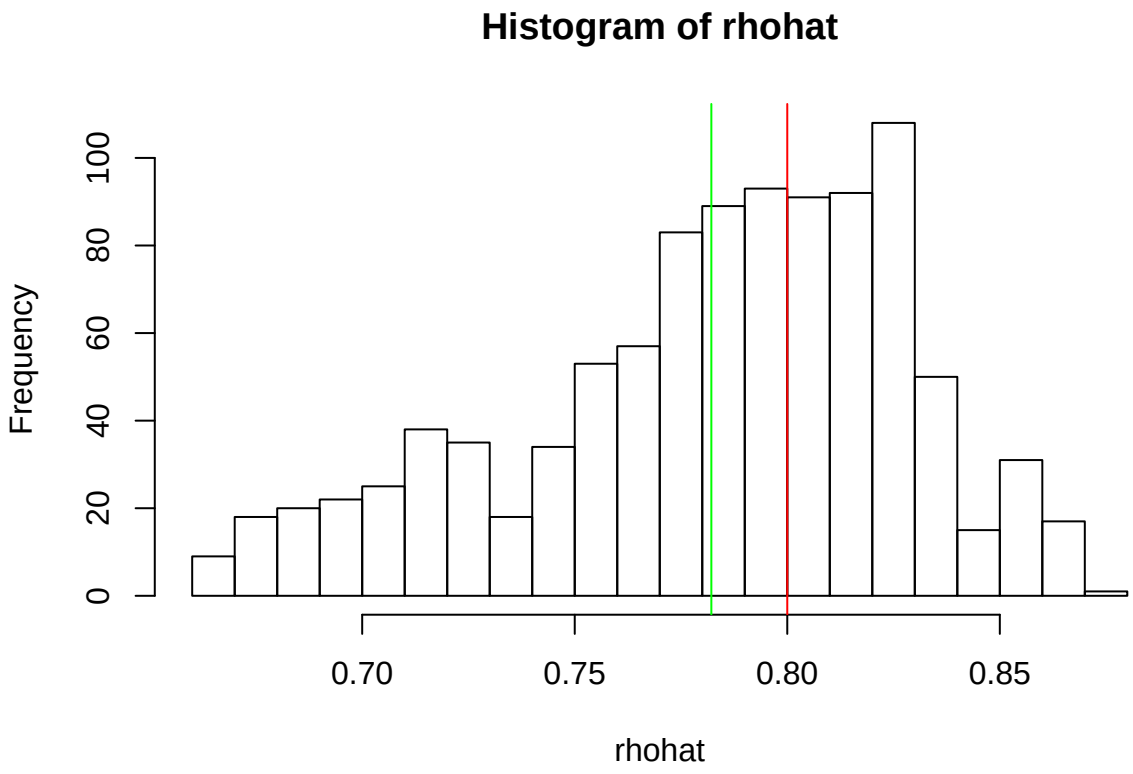

```
## [1]   0.8 100.0   2.0
## [1] -0.054466641  0.006282519  0.009249134
```



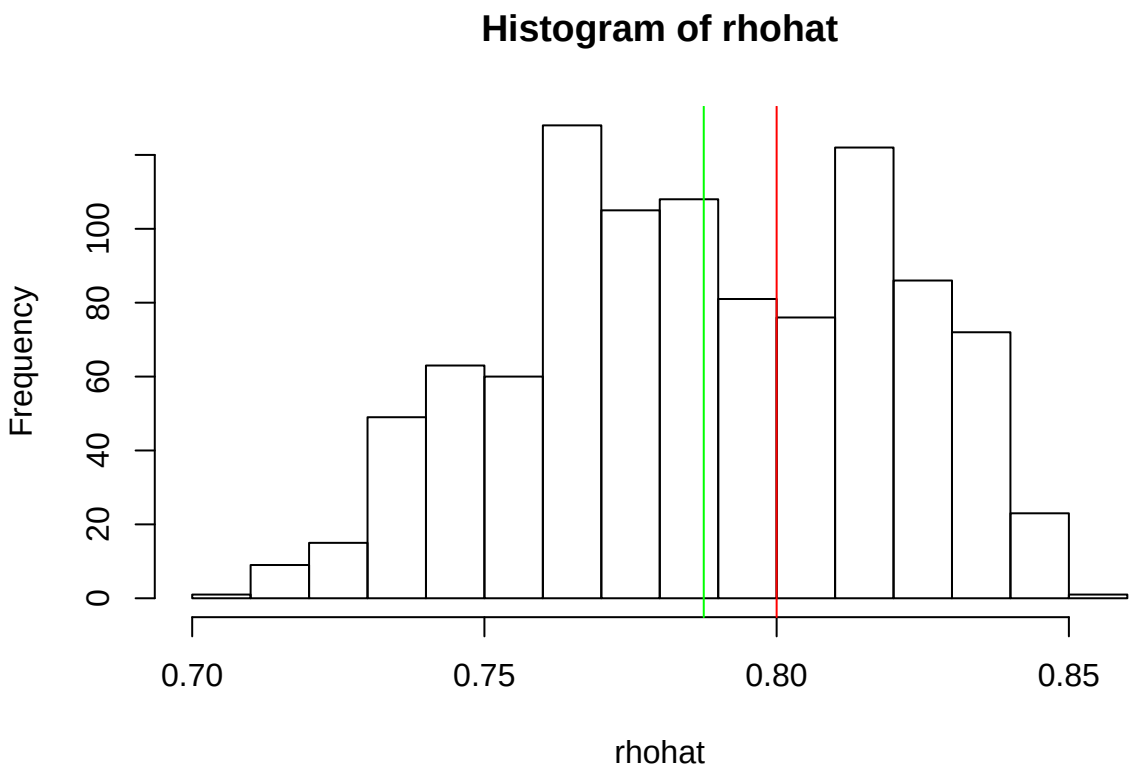

```
## [1]   0.8 200.0   0.5
## [1] -0.055729055  0.003188051  0.006293779
```



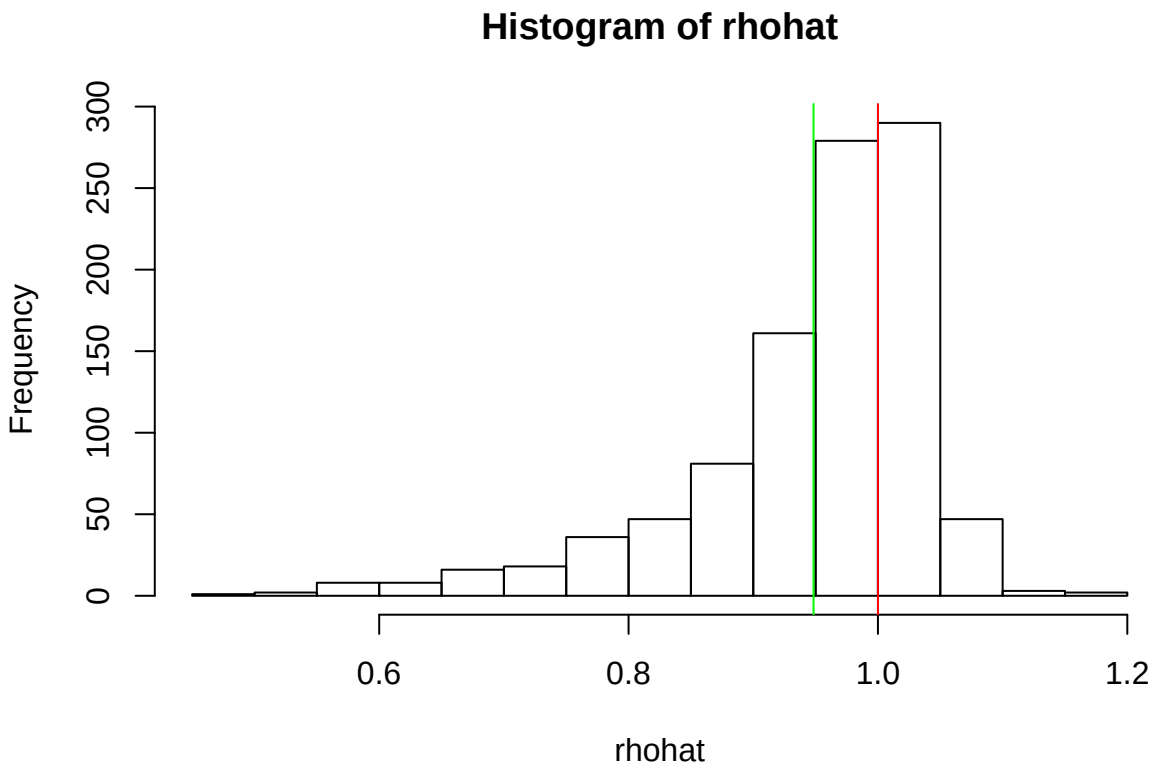

```
## [1]   0.8 200.0   1.0
## [1] -0.017858376  0.002118443  0.002437365
```



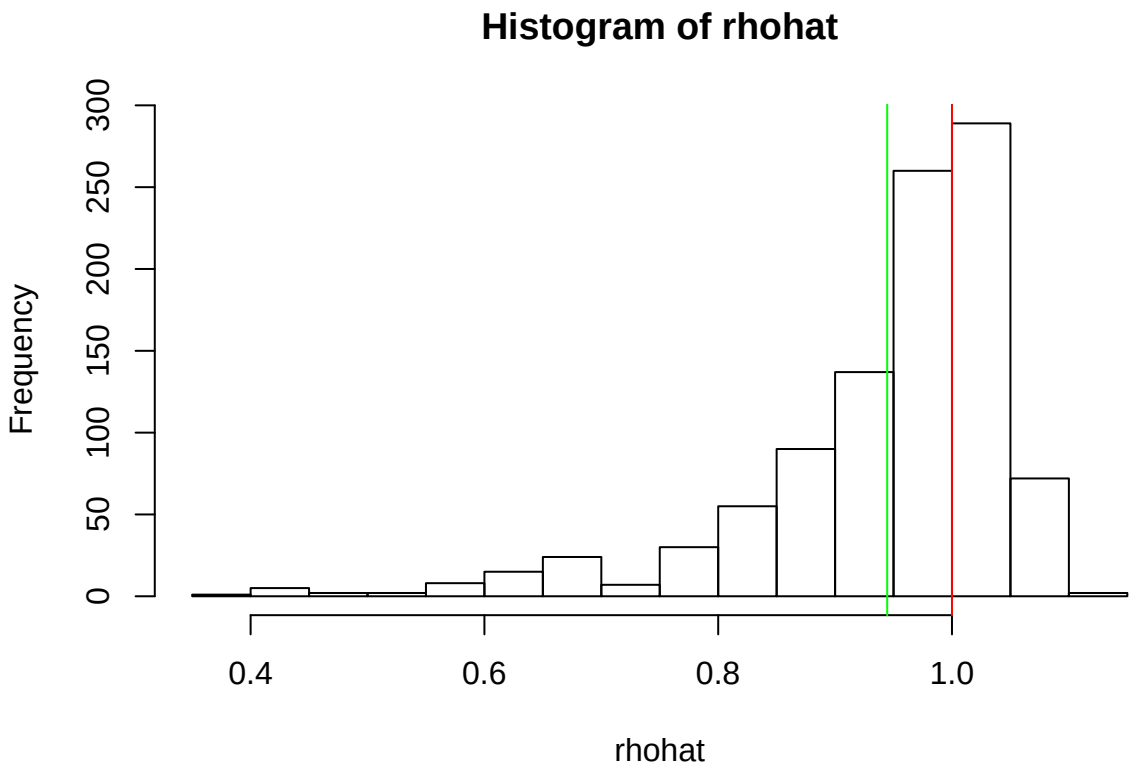

```
## [1]   0.8 200.0   2.0
## [1] -0.0124697711  0.0009917364  0.0011472316
```



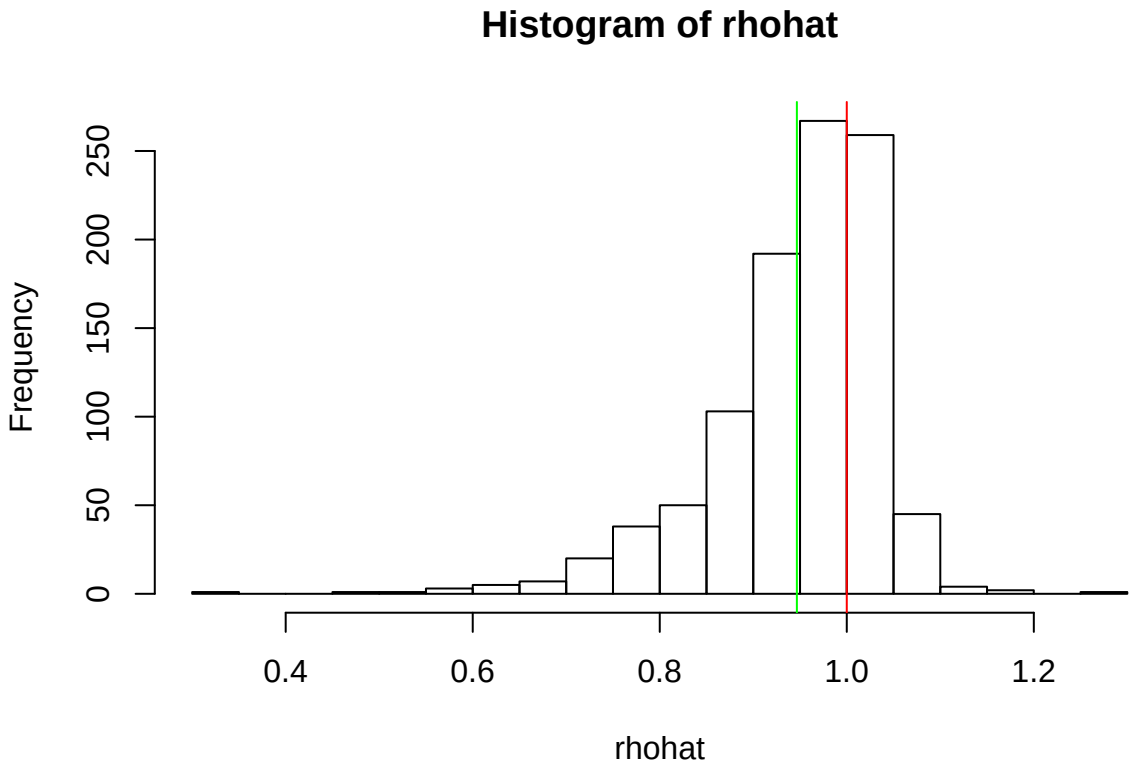

```
## [1]  1.0 25.0  0.5
## [1] -0.05165620  0.00969580  0.01236416
```



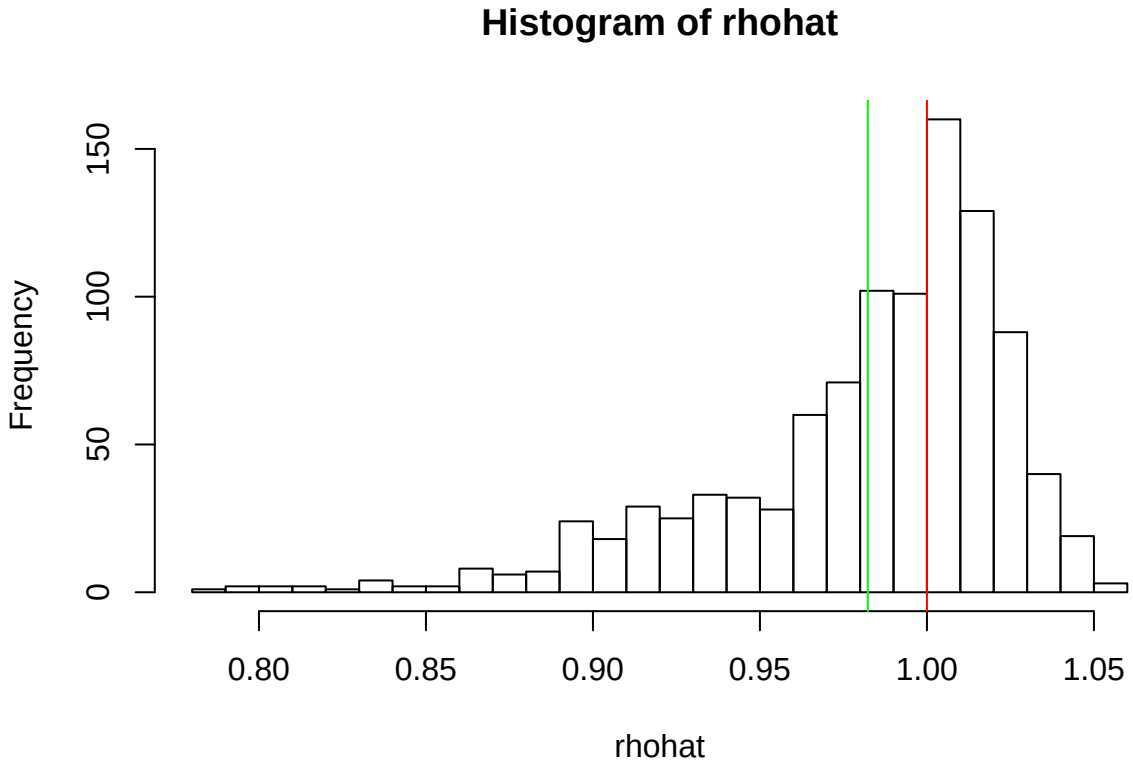

```
## [1]  1 25  1
## [1] -0.05541569  0.01261438  0.01568528
```



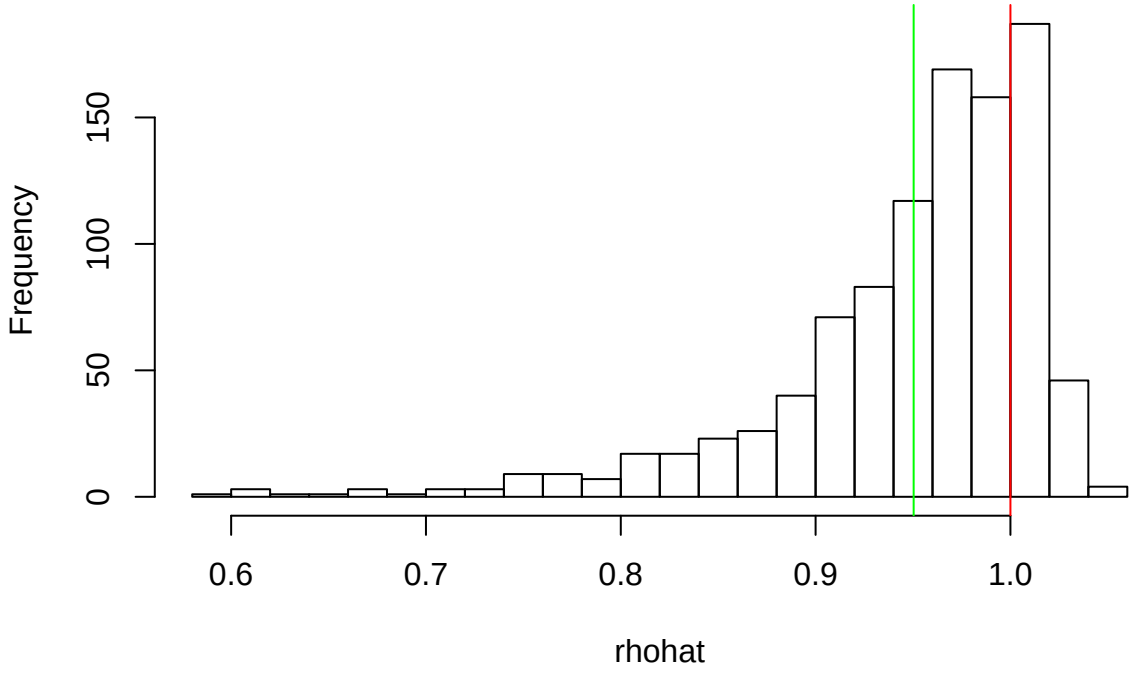

```
## [1]  1 25  2
## [1] -0.053370194  0.008436379  0.011284756
```



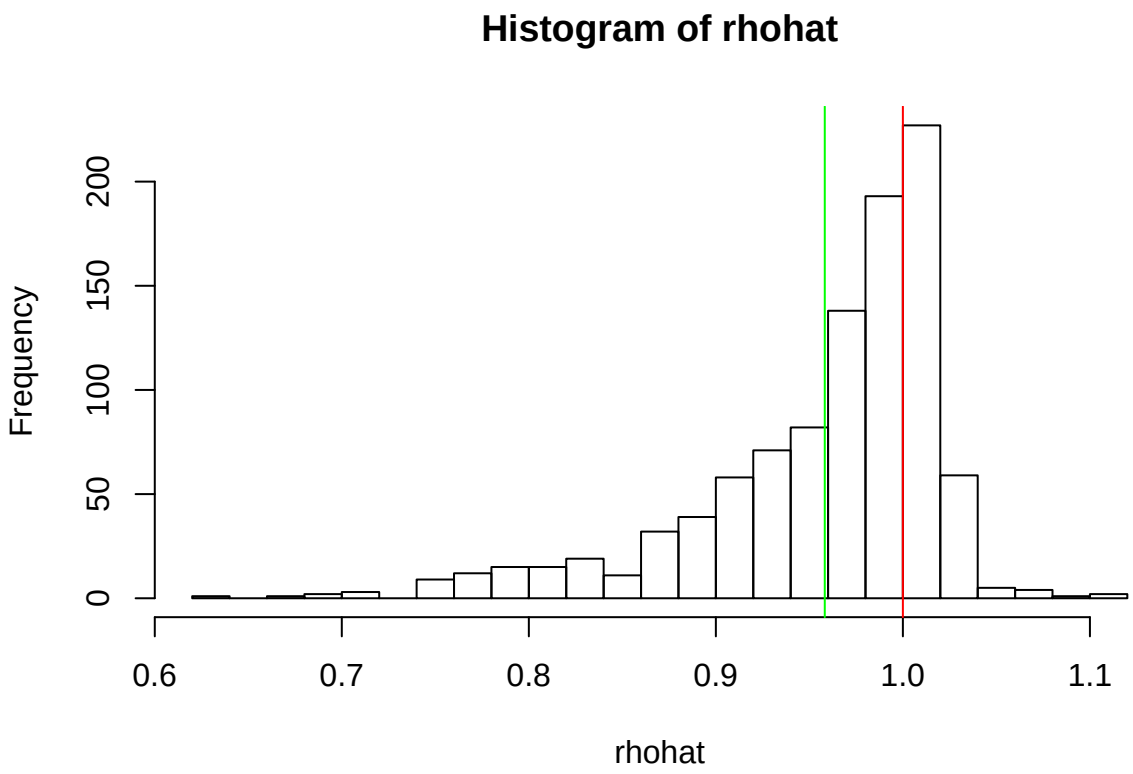

```
## [1]  1.0 50.0  0.5
## [1] -0.017689716  0.001920206  0.002233132
```



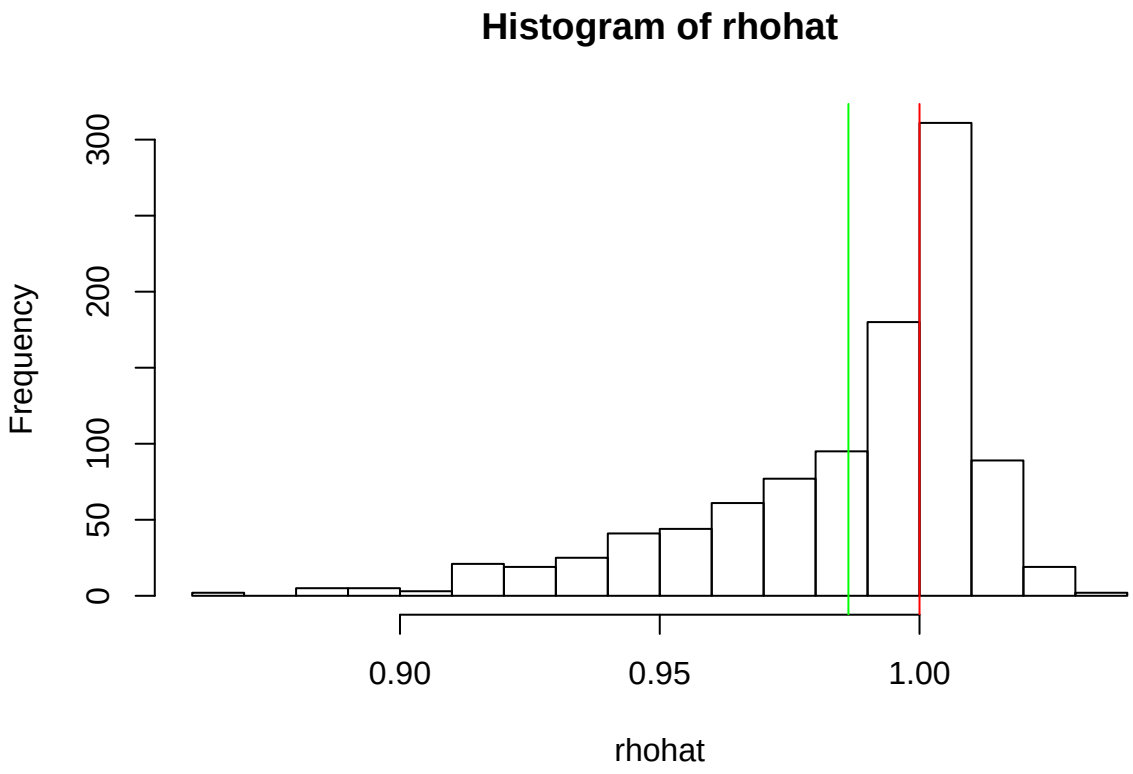

```
## [1]  1 50  1
## [1] -0.049672386  0.004661046  0.007128392
```



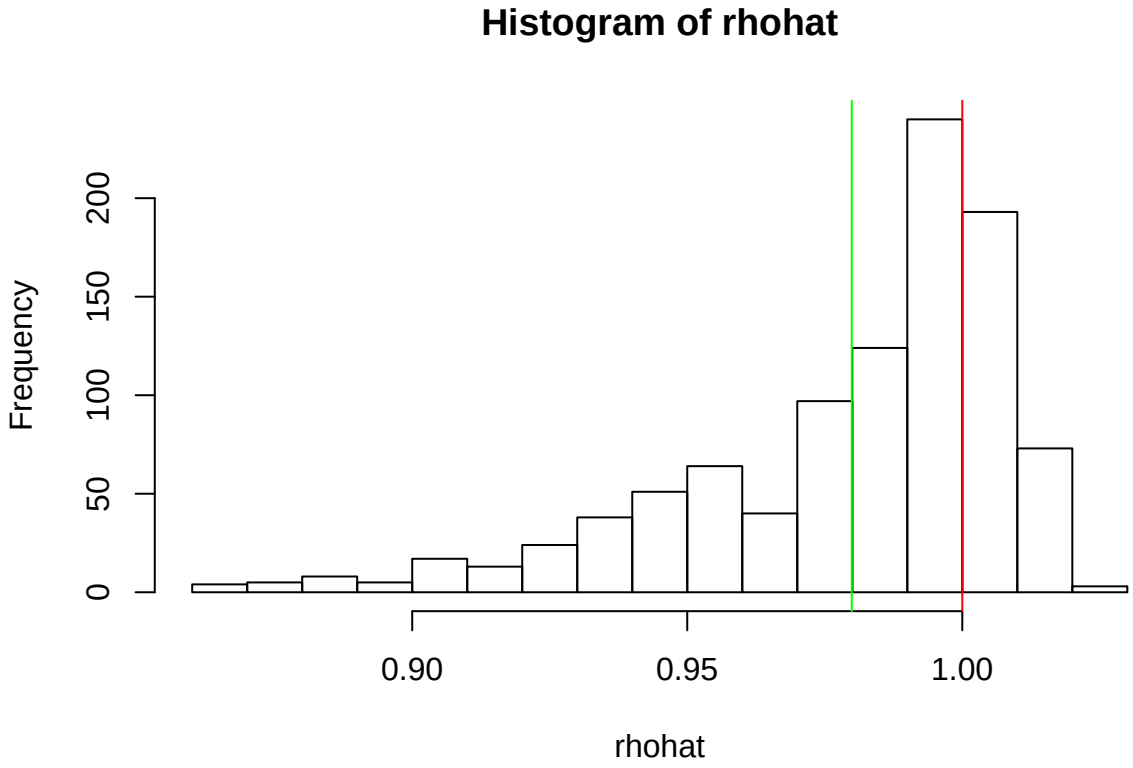

```
## [1]  1 50  2
## [1] -0.041769860  0.004441612  0.006186333
```



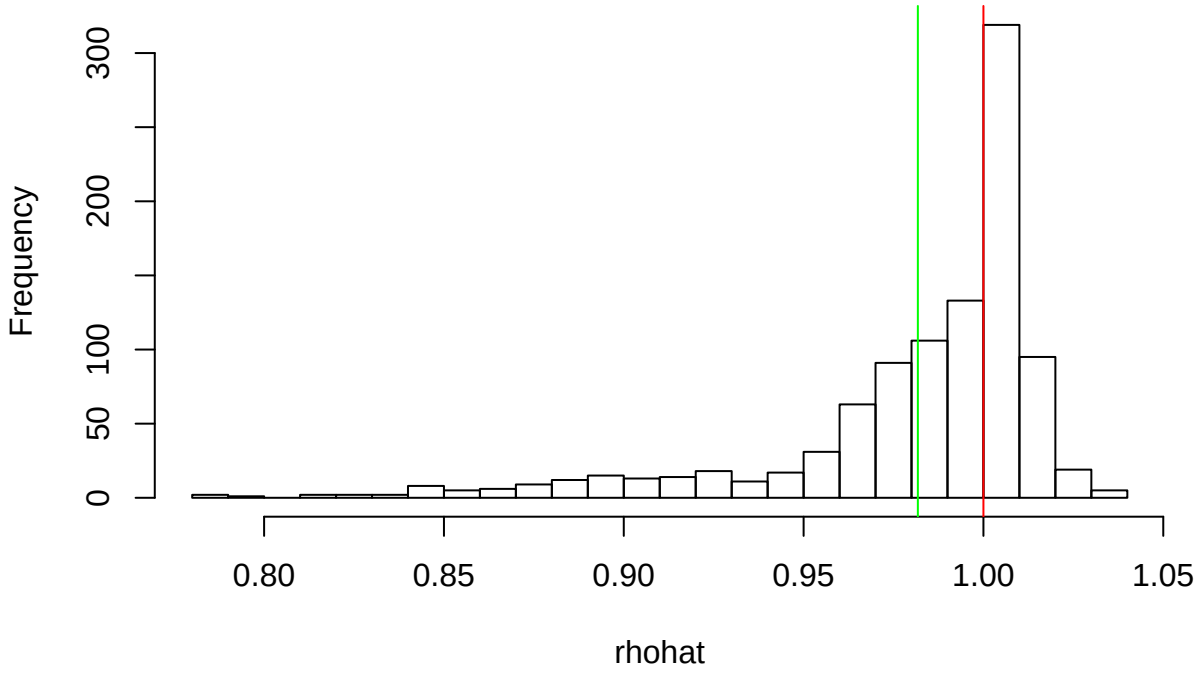

```
## [1]   1.0 100.0   0.5
## [1] -0.0136810791  0.0007644549  0.0009516268
```



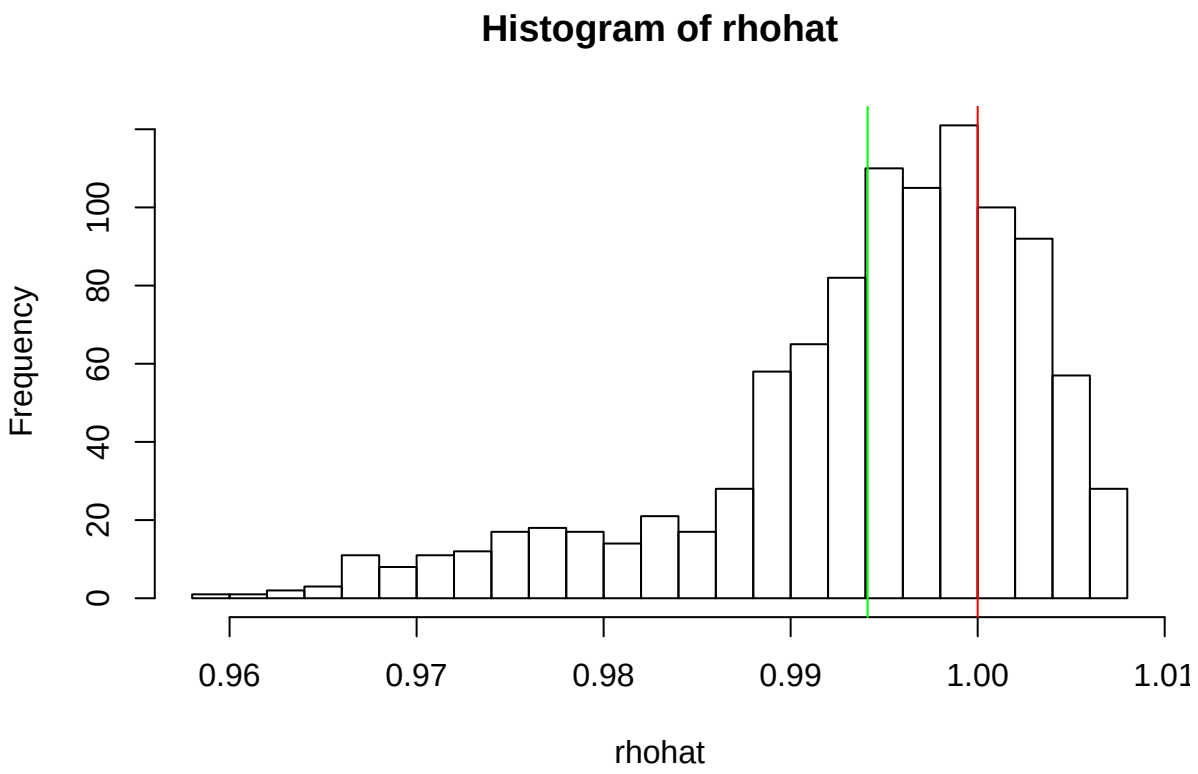

```
## [1]   1 100   1
## [1] -0.0200930108  0.0008886354  0.0012923645
```



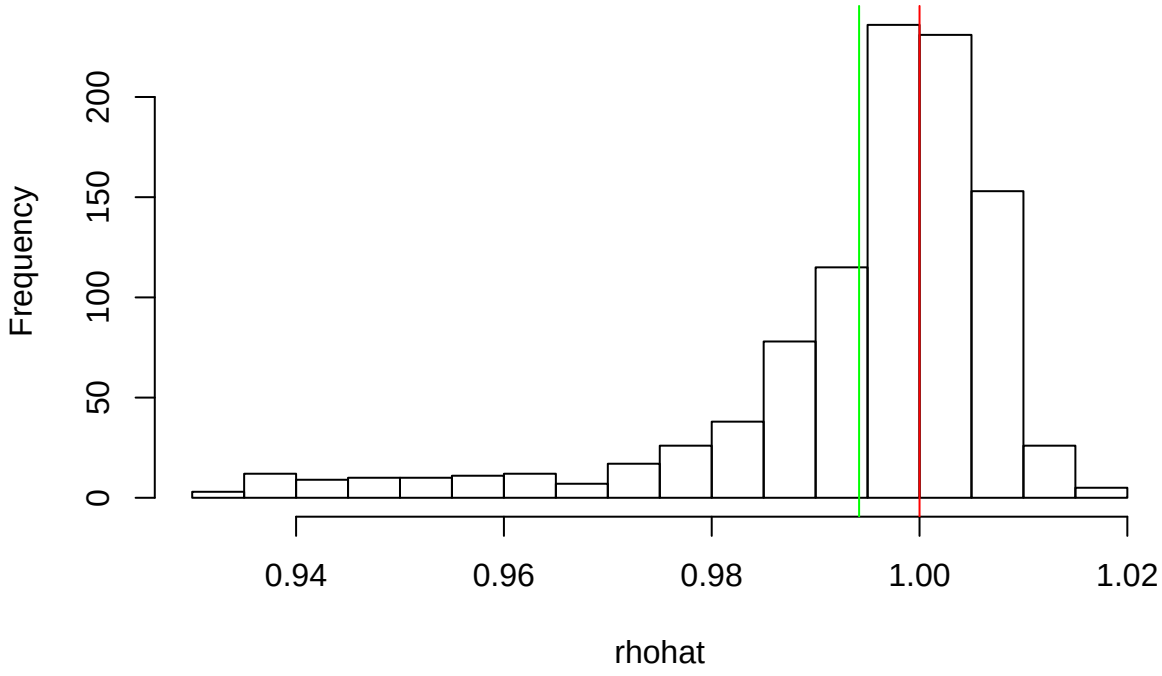

```
## [1]   1 100   2
## [1] -0.018238930  0.001533835  0.001866494
```



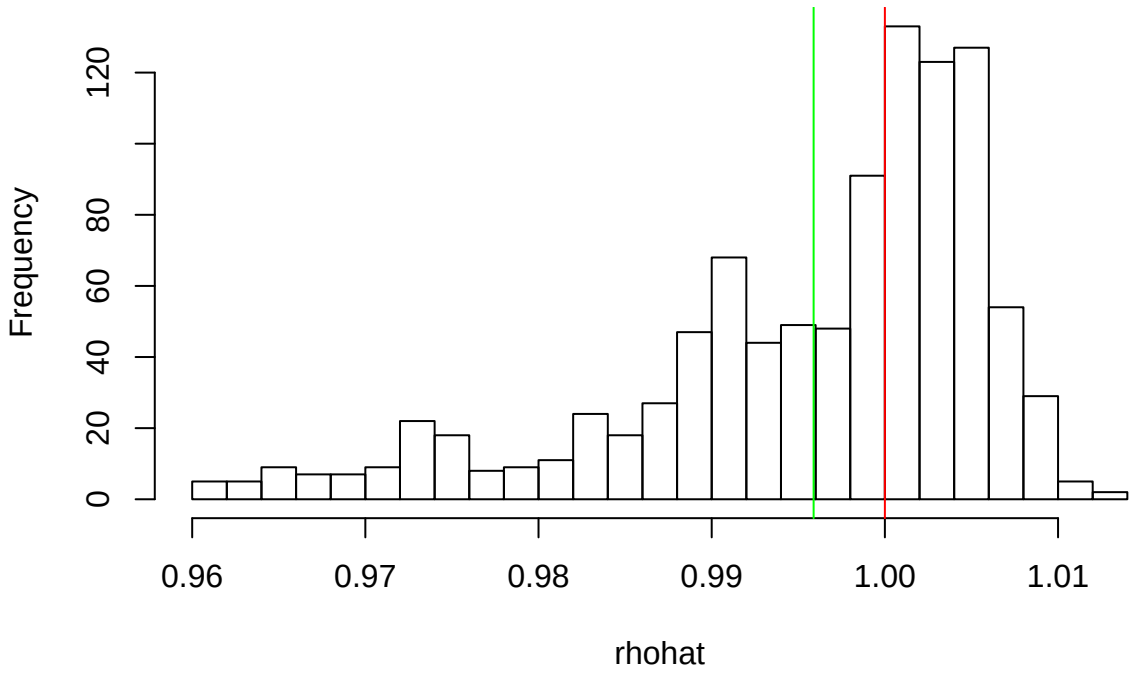

```
## [1]   1.0 200.0   0.5
## [1] -5.883382e-03  8.370974e-05  1.183239e-04
```




```
## [1]   1 200   1
## [1] -0.0058162861  0.0002218818  0.0002557109
```



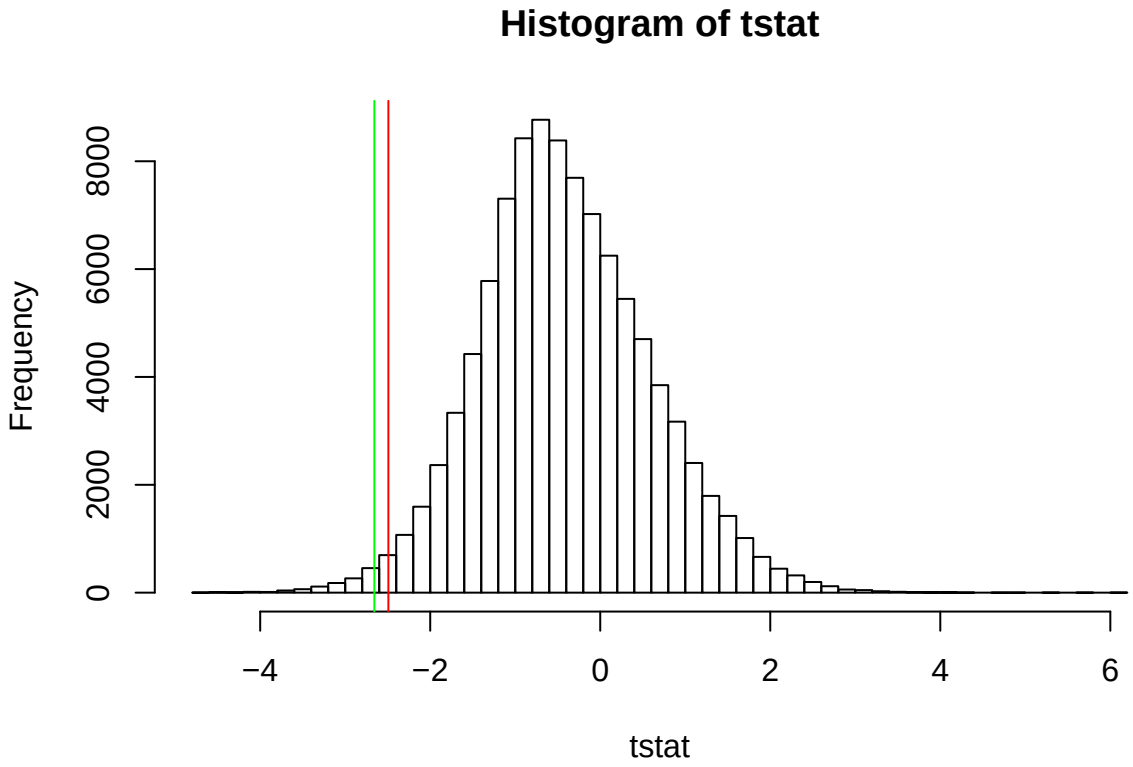

```
## [1]   1 200   2
## [1] -0.0041153016  0.0001105927  0.0001275284
```



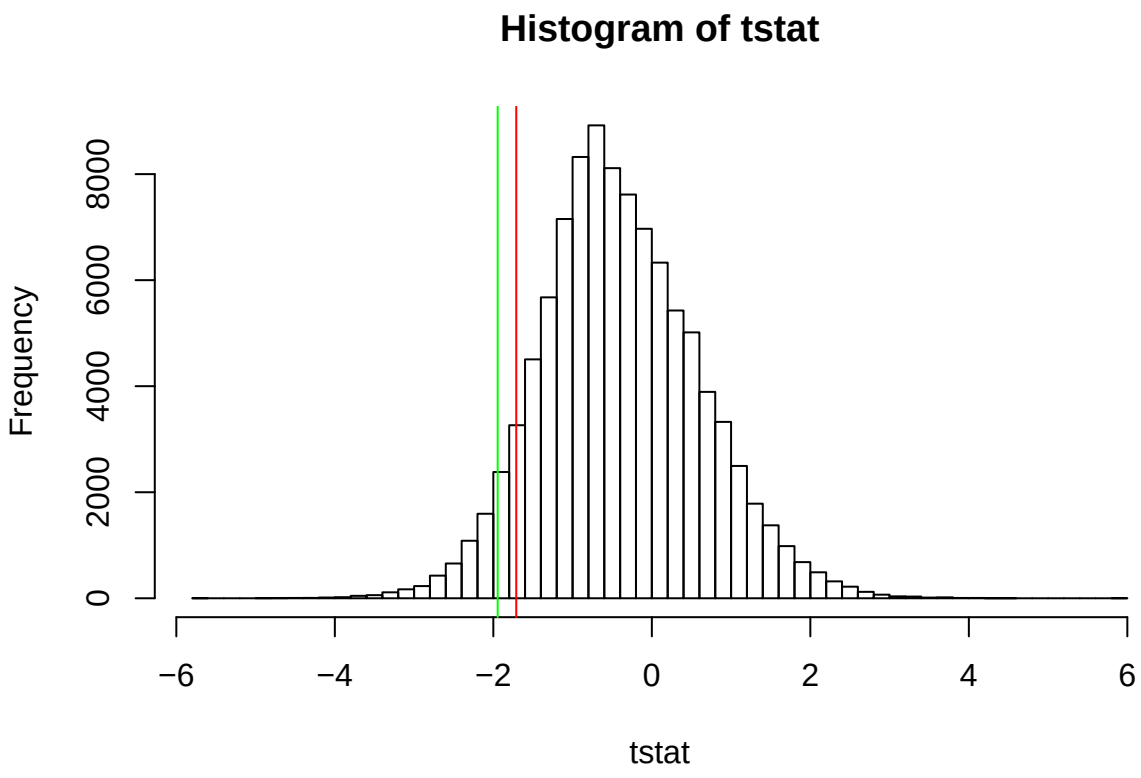

```
# non-stationary AR(1) model
# df test
S=99999
dftest=function(T,alpha,S){
  epsilon=matrix(rnorm(T+1*S),nrow = T+1,ncol=S,byrow = T)
  y=zeros(T+1,S)
  y[1,]=epsilon[1,]
  for (t in 1:T){
    y[t+1,]=y[t,]+epsilon[t+1,]
  }
  tstat=zeros(S,1)
  for (s in 1:S){
    ols=olsreg(t(t(y[2:(T+1),s])),t(t(y[1:T,s])))
    rhohat=ols[1]
    se=ols[2]
    wrong_t=ols[3]
    tstat[s]=(rhohat-1)/se
  }
  tsort=sort(tstat)
  cv_sim=tsort[round(alpha*(S+1))]
  cv_naive=qt(alpha,T-1)

  num_rej=sum(tstat<cv_naive)
  sz_naive=num_rej/S
  print(c(T,alpha,cv_sim,cv_naive,sz_naive))
  hist(tstat,breaks=50)
  abline(v=c(cv_sim,cv_naive),col=c("green","red"))
}
```

```
for (T in c(25,50,100,200)){
  for (alpha in c(0.01,0.05,0.1)){
    dftest(T,alpha,99999)
  }
}
```

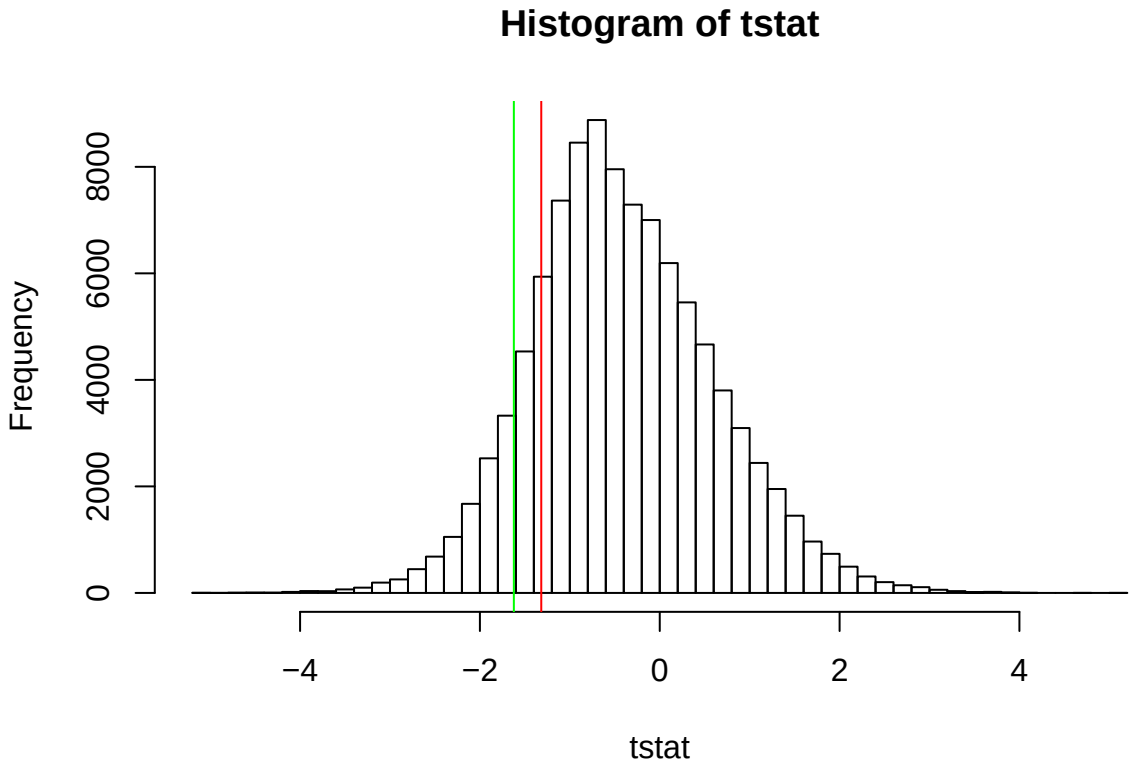
```
## [1] 25.00000000  0.01000000 -2.65706705 -2.49215947  0.01493015
```



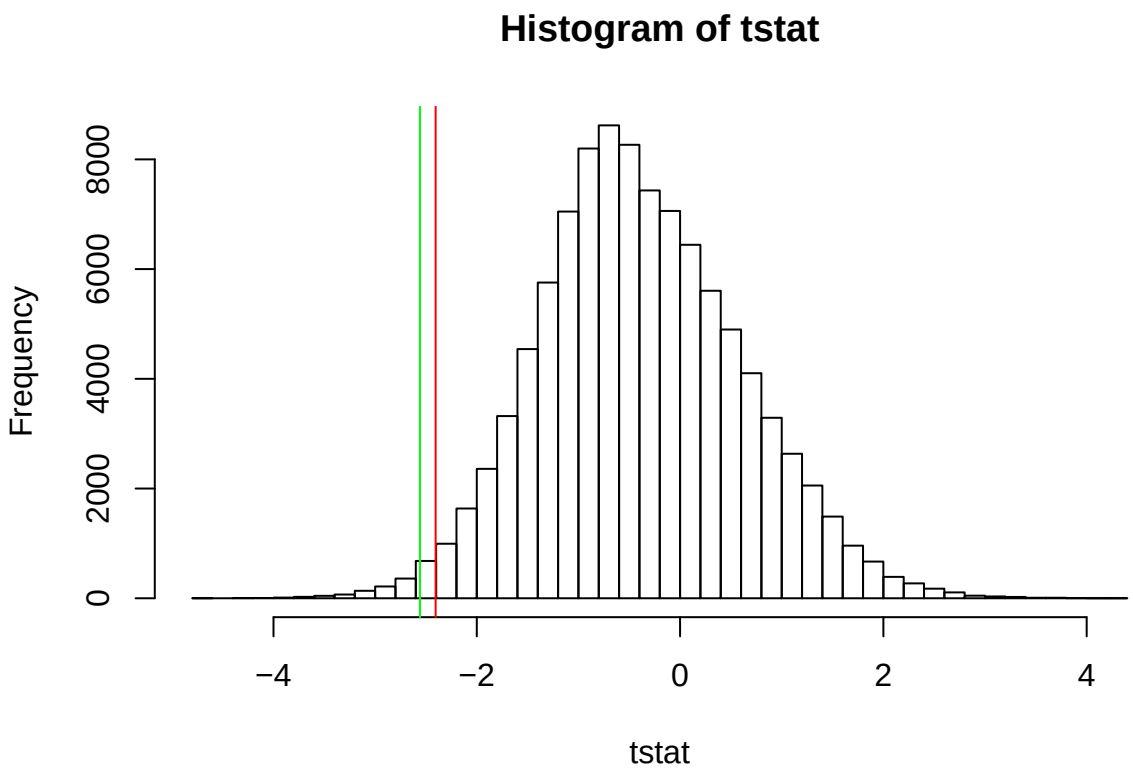

```
## [1] 25.00000000  0.05000000 -1.94540833 -1.71088208  0.08111081
```



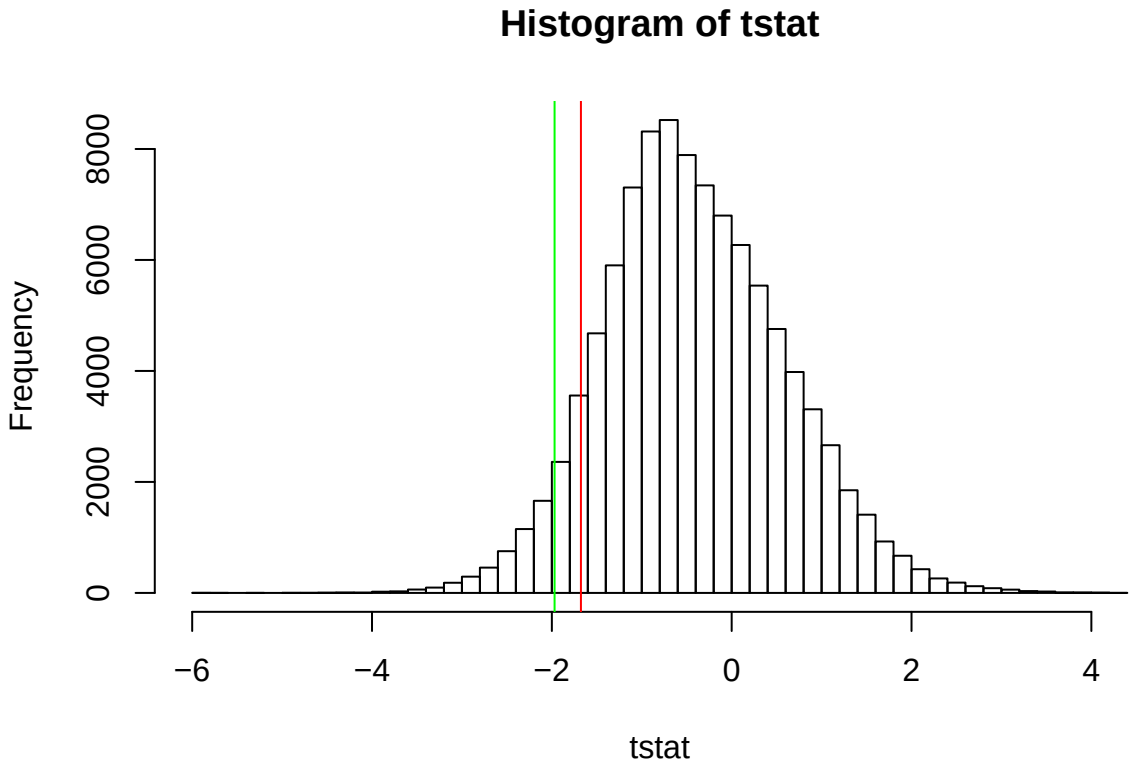

```
## [1] 25.0000000  0.1000000 -1.6226329 -1.3178359  0.1726517
```



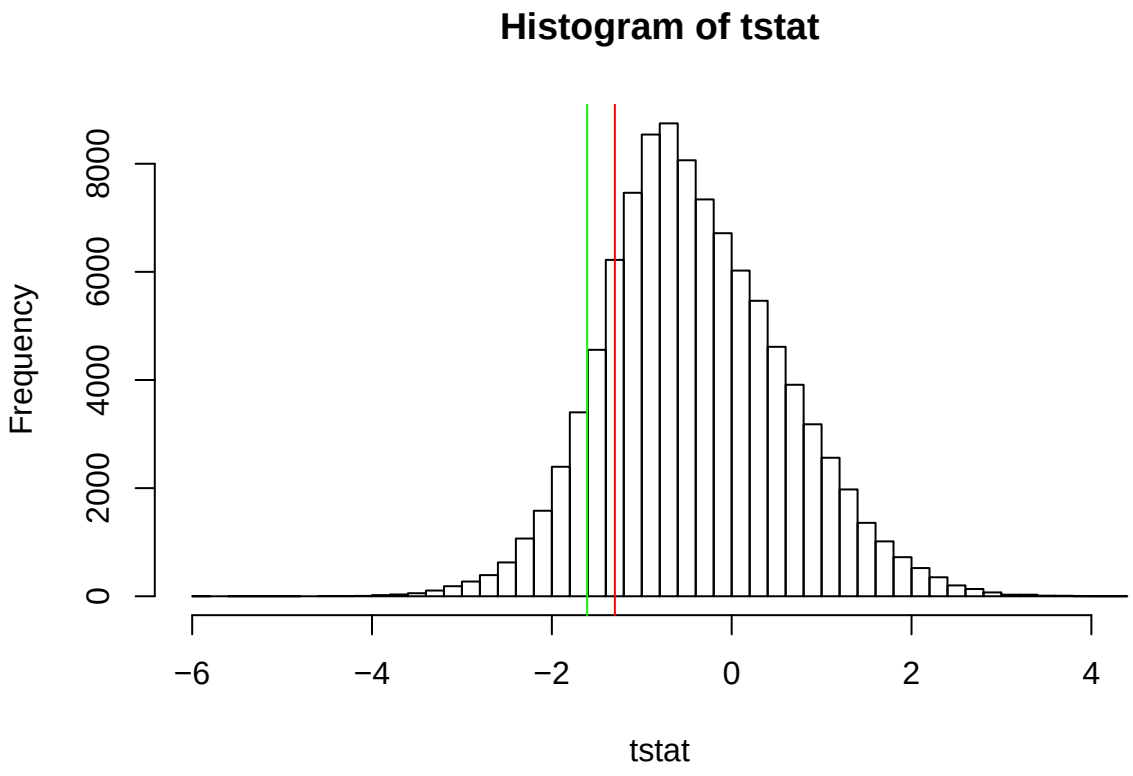

```
## [1] 50.00000000  0.01000000 -2.55894306 -2.40489176  0.01531015
```



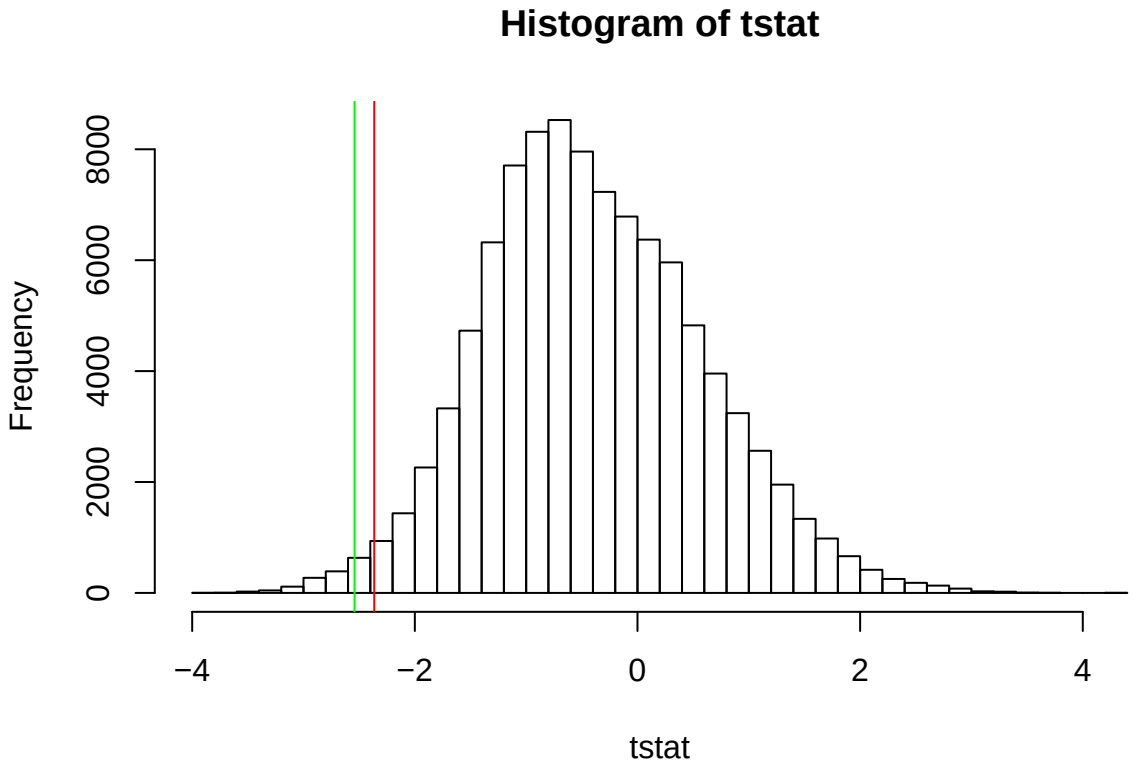

```
## [1] 50.00000000  0.05000000 -1.96978616 -1.67655089  0.09160092
```



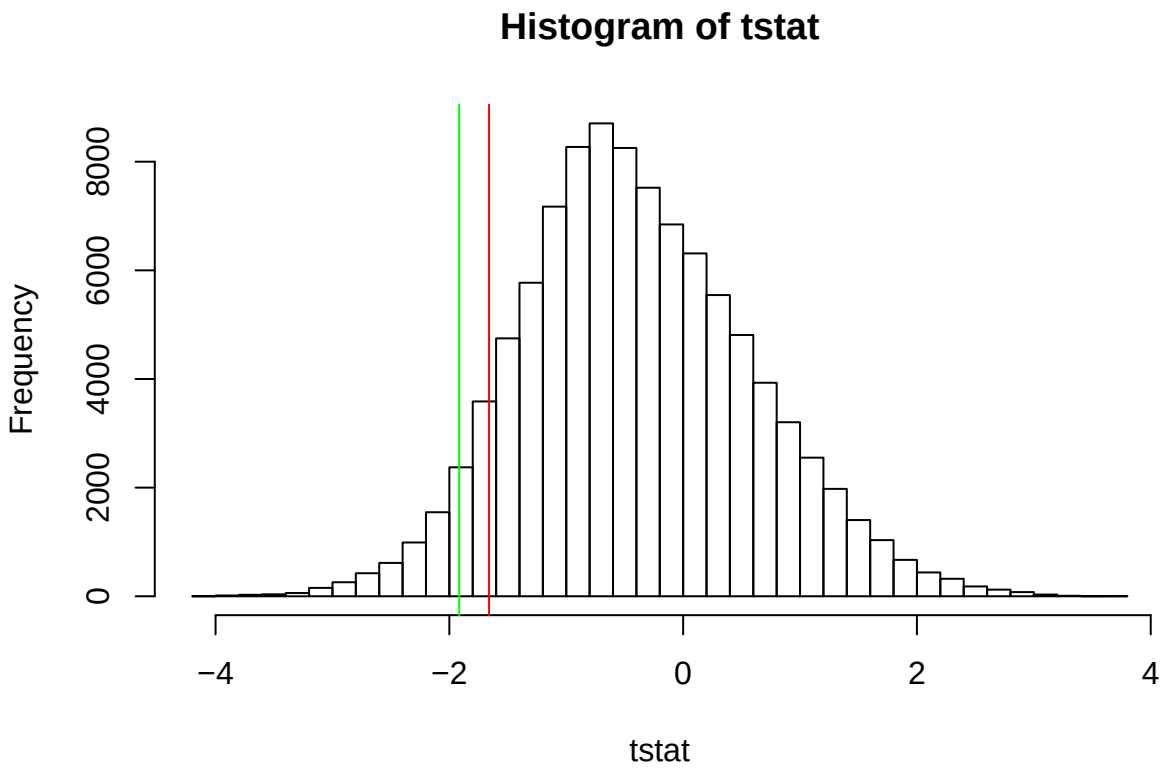

```
## [1] 50.0000000  0.1000000 -1.6078147 -1.2990688  0.1763418
```



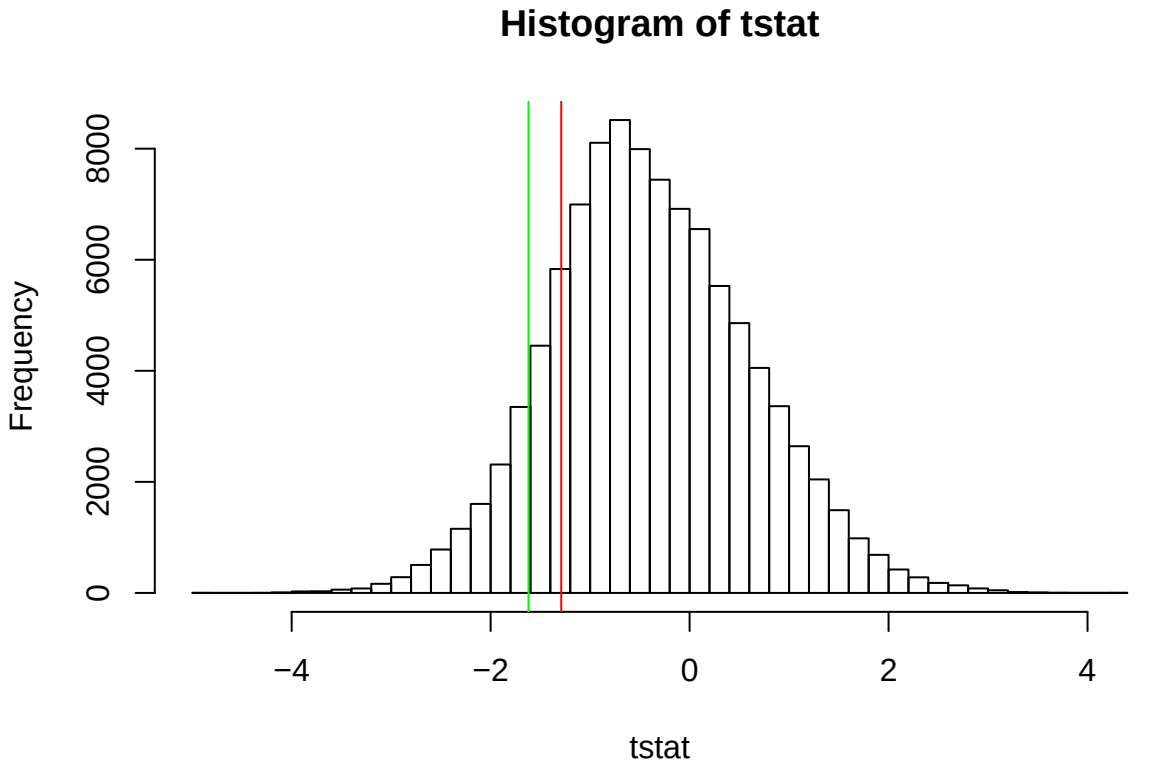

```
## [1] 100.00000000   0.01000000  -2.54171691  -2.36460586   0.01622016
```



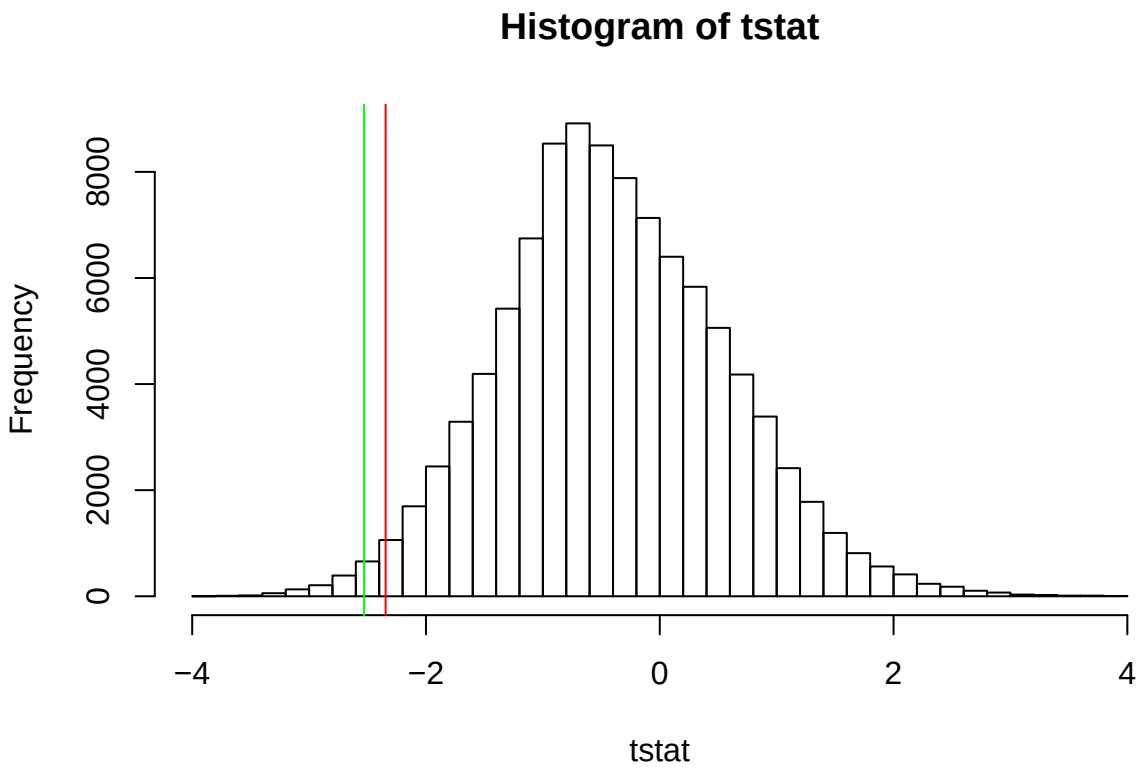

```
## [1] 100.00000000   0.05000000  -1.91572496  -1.66039116   0.08890089
```



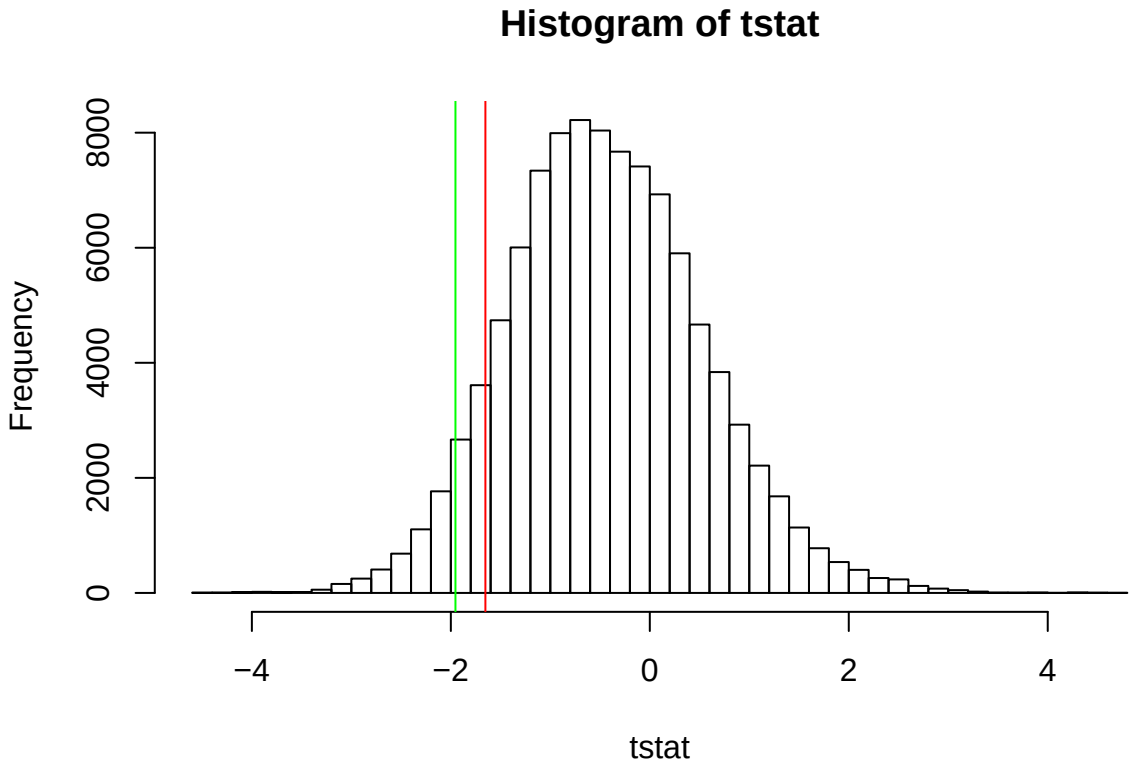

```
## [1] 100.0000000   0.1000000  -1.6194100  -1.2901614   0.1781818
```



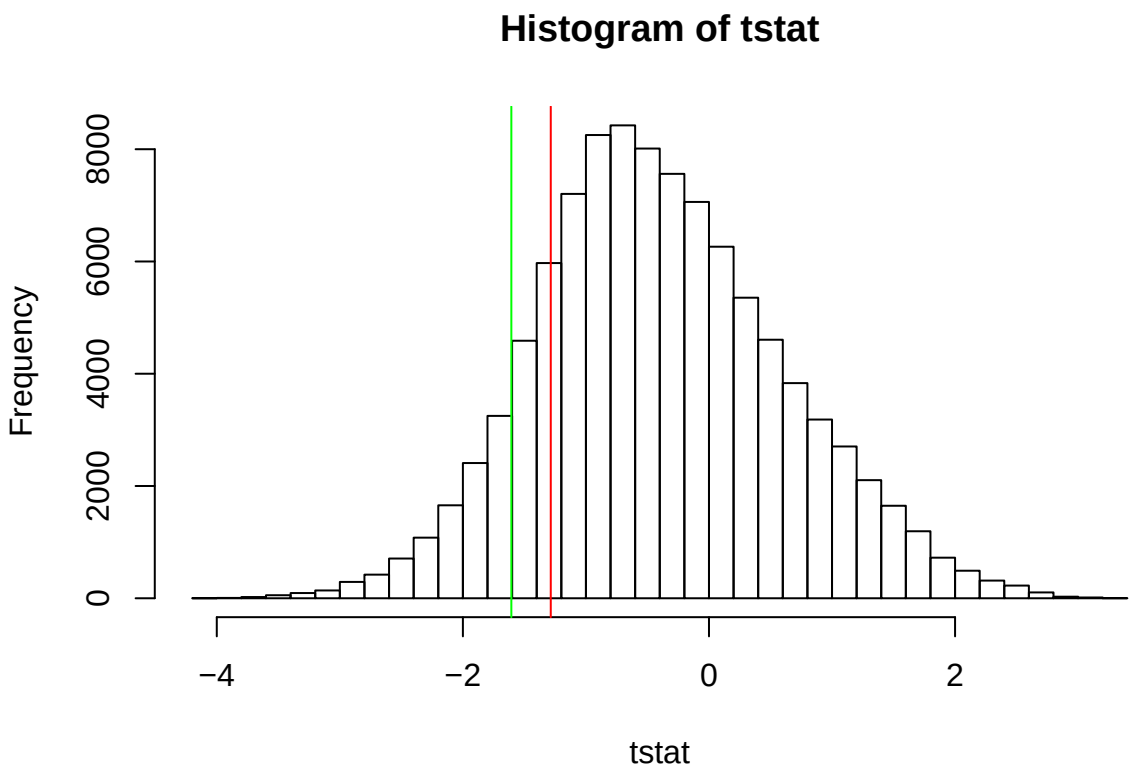

```
## [1] 200.00000000   0.01000000  -2.52905113  -2.34523223   0.01693017
```




```
## [1] 200.00000000   0.05000000  -1.95361928  -1.65254675   0.09770098
```




```
## [1] 200.0000000   0.1000000  -1.6062059  -1.2858202   0.1789818
```


Histogram of tstat
Frequency
0
2000
4000
6000
8000
−4
−2
0
2
tstat