Stat 425 HW5 Solutions

Fritz Scholz

1. Anatomy of an R function. Execute the following function with the given default arguments.

```
Using.get=function(N=5,dist="norm"){
   set.seed(26)
   x=get(paste("r",dist,sep=""))(N)
   x
}
```

Follow this with the command lines

```
  set.seed(26)
  rnorm(5)
and again
  rnorm(5)
```

To understand the 3rd line in `Using.get` execute the following commands in the given progression and describe your understanding of what is happening.

```
paste("r","norm")
paste("r","norm",sep="")
get(paste("r","norm"))
get(paste("r","norm",sep=""))
rnorm
set.seed(26)
rnorm(5)
set.seed(26)
get(paste("r","norm",sep=""))(5)
```

What is the usefulness of the argument `dist` in `Using.get`? Try the call

```
Using.get(N=5,dist="unif")
```

Possible values for `dist` are `unif, norm, lnorm, logis, exp, cauchy`. If you put an `r` before any of those strings you get the command for a random sample from that continuous distribution. For example, you get information on `logis` by searching in the web based help facility `help.start()` for `rlogis` or by executing `?rlogis` at the command line. All of these random sample generators have default arguments, but there are others (like `rf, rt, rweibull, rgamma`) that do not. If you use one of these latter random sample generating functions the above construct of `Using.get` would fail. To accommodate these the function `Using.get` would need some modification, but we will pass on that for now.

```
> Using.get()
[1] -2.1298417  1.1478961 -0.4895019  0.8263438 -0.4099352
>   set.seed(26)
>   rnorm(5)
[1] -2.1298417  1.1478961 -0.4895019  0.8263438 -0.4099352
# we got the same set of numbers, thus Using.get appears to do the same
# as rnorm(5) preceded by set.seed(26)
> rnorm(5)
[1]  0.14878788  0.12807120  0.91546624 -0.03993861  0.42013653
# here we got a different set of numbers because we did not reset the seet to 26.

>   paste("r","norm")
[1] "r norm" # here we have a space between r and norm
>   paste("r","norm",sep="")
[1] "rnorm" # here we have no space between r and norm
>   get(paste("r","norm"))
Error in get(paste("r", "norm")) : object ’r norm’ not found
# here R complains because there is no object with name ’r norm’ with
# a space in it.
>   get(paste("r","norm",sep=""))
function (n, mean = 0, sd = 1)
.Internal(rnorm(n, mean, sd))
<environment: namespace:stats>
# here R spits out the function body, which is the same as if
# I give rnorm on the comman line, see next.
>   rnorm
function (n, mean = 0, sd = 1)
.Internal(rnorm(n, mean, sd))
<environment: namespace:stats>
# the displayed result is exactly the same as before.
>   set.seed(26)
>   rnorm(5)
[1] -2.1298417  1.1478961 -0.4895019  0.8263438 -0.4099352
>   set.seed(26)
>   get(paste("r","norm",sep=""))(5)
[1] -2.1298417  1.1478961 -0.4895019  0.8263438 -0.4099352
# the previous two sequences of commands produce the same
# set of standard normal deviates.
```

The third line in `Using.get` does the following. It concatenates two strings without a space between them, i.e., ”r” and ”norm” to become ”rnorm”. Since I cannot refer to an object by ”rnorm” (”rnorm” is a character string and just a character value) I cannot call ”rnorm”(5). I want to call rnorm(5) without the quotes. That is what the function get does to ”rnorm”, i.e., strips the quotes and treats the remainder as an object (function) name.
The usefulness of the argument `dist` is that with the same function I can carry out simulations by sampling from different distribution, simply by giving a different value to `dist`. For example

```
> Using.get(dist="unif")
[1] 0.01659234 0.28947830 0.87449426 0.79992188 0.31224322
```

In order to enable other distributions which require parameter values, one could add a `parm` vector to the calling sequence of `Using.get` and then use `if` clauses to handle distributions differently by using parameter values `parm[1]`, `parm[2]`, `parm[3]` as needed. The following is such a version but it does not use the `get` construct. We pass the function object directly via dist (without quotes, with the r built a priori into the object name). It can handle up to three parameters and is easily extended to handle more if needed.

```
Using.dist=function (N = 5, dist = rnorm, parm) {
    np=length(parm)
    if(np==1){
    x=dist(N,parm[1])}
    if(np==2){
    x=dist(N,parm[1],parm[2])}
    if(np==3){
    x=dist(N,parm[1],parm[2],parm[3])}
    x
}
```

For example try the following two examples, the first one involving 3 parameters and the second involving just one parameter.

```
> Using.dist(5,dist=rf,parm=c(10,15,20))
[1] 0.8611638 1.6795521 2.6281027 3.2001922 3.0871473

> Using.dist(5,dist=rexp,parm=10)
[1] 0.35276456 0.08510604 0.02164305 0.12026024 0.10568541
```

2. Via simulation you are to investigate the sampling distributions of the Hodges-Lehmann estimator $\hat{\Delta} = \text{median}(Y_j - X_i)$ and of $\bar{\Delta} = \bar{Y} - \bar{X}$ in the shift model for various distributions F. Write a function `Delta.est.sim(Nsim=10000,m=15,n=15,dist="norm",dist.name="normal",Delta=2){...}` that uses a loop `for( i in 1:Nsim ){...}` to simulate samples `x` of size `m` from $F(x)$ (corresponding to `dist`, make use of what you learned from Problem 1.) and samples `y` of size `n` from $F(x-\Delta)$ (i.e., shift $\Delta = 2$ for the above default argument) and computes $\hat{\Delta}$ and $\bar{\Delta}$ for these two samples. Accumulate these estimates in respective vectors `Delta.hat` and `Delta.bar` of length `Nsim` each. While you develop `Delta.est.sim` use `Nsim=1000` but for the final plots use `Nsim=10000`.
You are to illustrate through plots (to be produced by `Delta.est.sim`) the following results from class, when they are known to be true and when they might be false.
The distributions of both estimators are symmetric around Δ when $m = n$ or when F is symmetric around some point μ. The symmetry of the simulated distribution `y=Delta.hat` (and similarly `y=Delta.bar`) around Δ is most effectively illustrated by using `x=y-Delta` and

```
qqplot(x[x>0],-x[x<0],pch=16,cex=.5)
abline(0,1)
```

Here `x[x>0]` is the positive part of the `x`-distribution while `-x[x<0]` is the negative part reflected around zero. If both parts have equal length then `qqplot` plots the smallest of one part against the smallest of the other, then the second smallest ones are plotted against each other, ..., and finally the largest ones are plotted against each other. If the two parts are of different length then the sorted larger vector is interpolated to give the same length as the shorter vector and these shorter vectors are plotted against each other according to the above scheme. When the distribution of `x` is symmetric around zero, then the plotted point pattern should align well with the main diagonal. The extremes in any data set typically show greater fluctuations. Thus one should be more forgiving for the large observation (the tail values) when judging closeness to the main diagonal. `Delta.est.sim` should create annotated example plots (indicating `dist.name, m, n` and estimator type) that show

1. symmetry when $m = n$ even when F is not symmetric.
2. asymmetry when $m \neq n$ and F is not symmetric (make m and n sufficiently different so that the asymmetry becomes obvious).
3. symmetry when $m \neq n$ and F is symmetric (m and n as different as in the previous illustration).

Do the above illustration for both the simulated $\hat{\Delta}$ and $\bar{\Delta}$ distribution as captured by `Delta.hat` and `Delta.bar`. Give the code for `Delta.est.sim` and these plots. If you write a function that produces several plots and you want to look at them at your own pace (also for cutting and pasting) you should add the following command after each plot is finished:

```
readline("hit return\n")
```

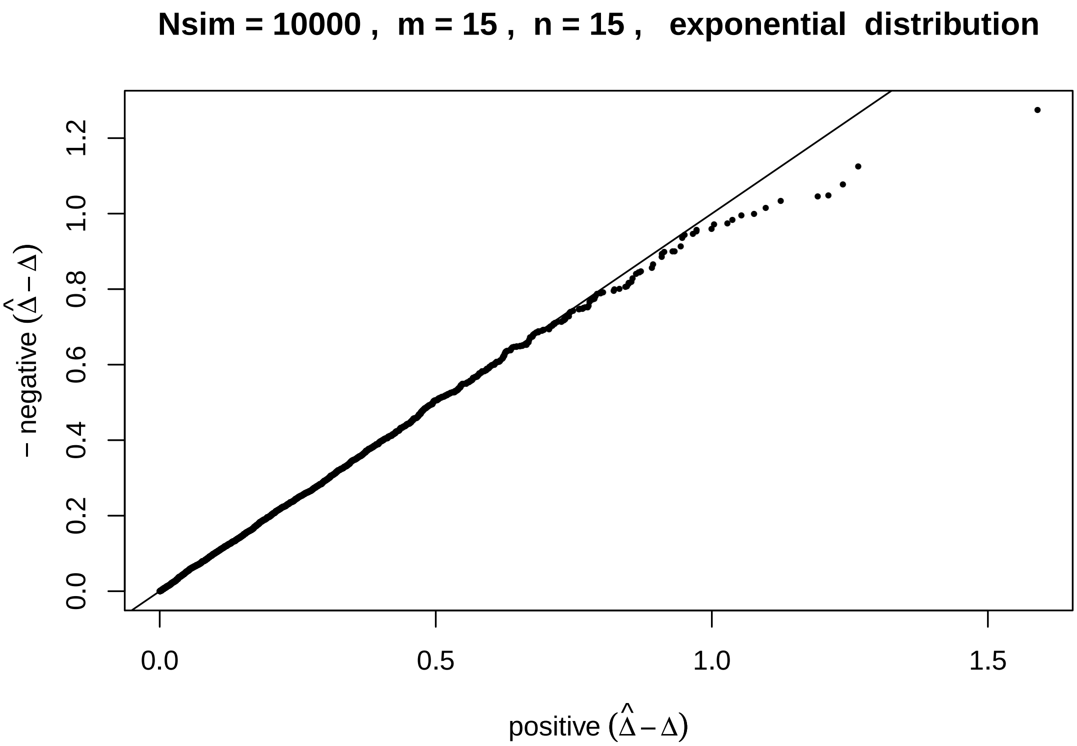
Optionally, for extra credit, illustrate the corresponding results or counter examples concerning unbiasedness and median unbiasedness. Here you would be looking at the mean of the generated distribution vectors, compare them with Δ and look at the proportion of the distributions $> \Delta$ and $< \Delta$.

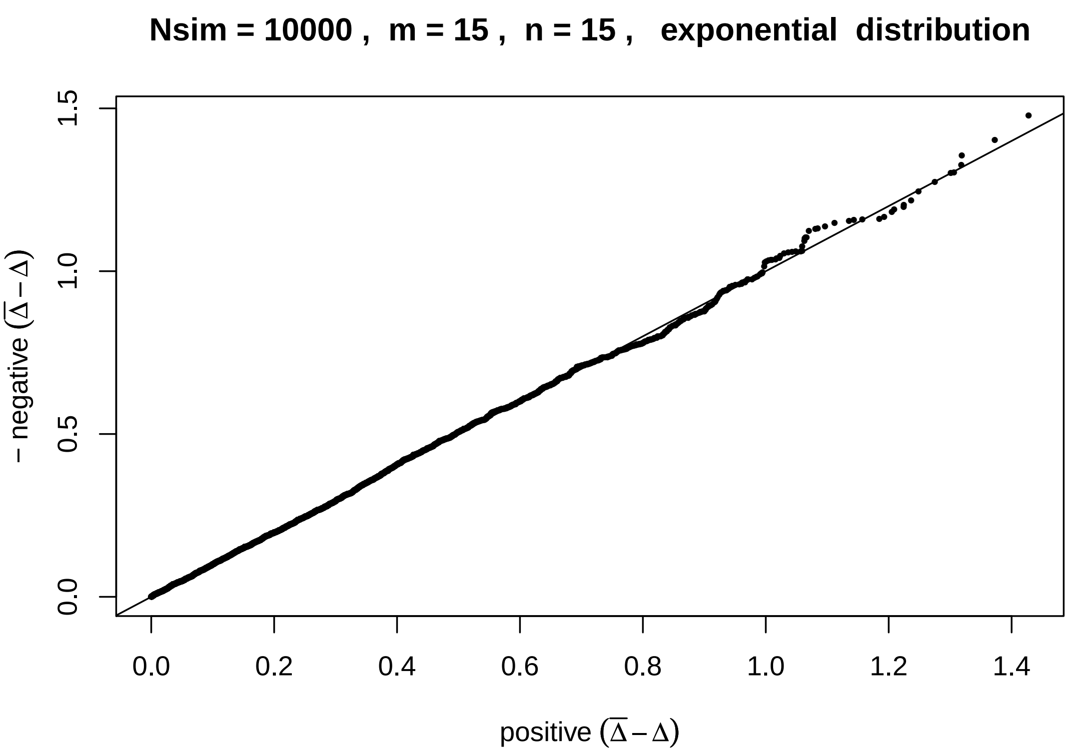
```
Delta.est.sim=function(m=15,n=15,Nsim=1000,dist=rnorm,parm=c(0,1),
                 dist.name="normal", Delta=2,PDF=F){
HL=rep(0,Nsim)
Delta.bar=HL
np=length(parm)
for(i in 1:Nsim){
if(np==1){
x=dist(m,parm[1])
y=dist(n,parm[1])+Delta}
if(np==2){
x=dist(m,parm[1],parm[2])
y=dist(n,parm[1],parm[2])+Delta}
if(np==3){
x=dist(m,parm[1],parm[2],parm[3])
y=dist(n,parm[1],parm[2],parm[3])+Delta}
HL[i]=median(outer(y,x,"-"))
Delta.bar[i]=mean(y)-mean(x)
}
if(PDF==T) pdf(file=paste("HLestimatesm",m,"n",n,dist.name,".pdf",sep=""),
  width=7,height=5)
x=HL-Delta
qqplot(x[x>0],-x[x<0],xlab=expression("positive"~(hat(Delta)-Delta)),
ylab="",pch=16,cex=.5)
mtext(expression("- negative"~(hat(Delta)-Delta)),2,2.5)
title(paste("Nsim =",Nsim,",  m =",m,",  n =",n,",  ",dist.name," distribution"))
abline(0,1)
if(PDF==T) dev.off()
readline("hit return\n")
if(PDF==T) pdf(file=paste("YbarXbarm",m,"n",n,dist.name,".pdf",sep=""),
  width=7,height=5)
x=Delta.bar-Delta
qqplot(x[x>0],-x[x<0],xlab=expression("positive"~(bar(Delta)-Delta)),
ylab="",pch=16,cex=.5)
mtext(expression("- negative"~(bar(Delta)-Delta)),2,2.5)
title(paste("Nsim =",Nsim,",  m =",m,",  n =",n,",  ",dist.name," distribution"))
abline(0,1)
if(PDF==T) dev.off()
# the lines below create the output for optional credit.
mean.HL=mean(HL)
mean.Delta.bar=mean(Delta.bar)
p.above.Delta.bar=mean(Delta.bar>Delta)
p.below.Delta.bar=mean(Delta.bar<Delta)
```

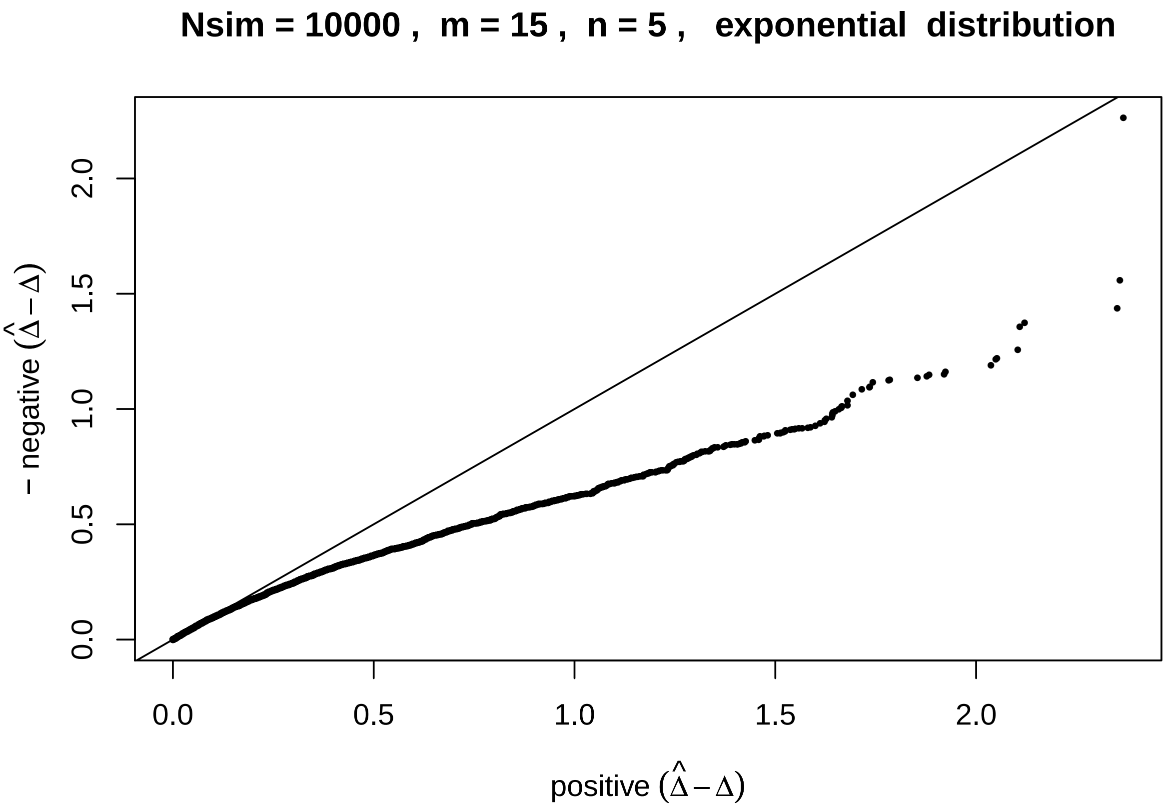
```
p.above.HL=mean(HL>Delta)
p.below.HL=mean(HL<Delta)
out=c(mean.Delta.bar,mean.HL,p.above.Delta.bar,p.below.Delta.bar,
p.above.HL,p.below.HL)
names(out)=c("mean.Delta.bar","mean.HL","p.above.Delta.bar","p.below.Delta.bar",
"p.above.HL","p.below.HL")
out
}
```

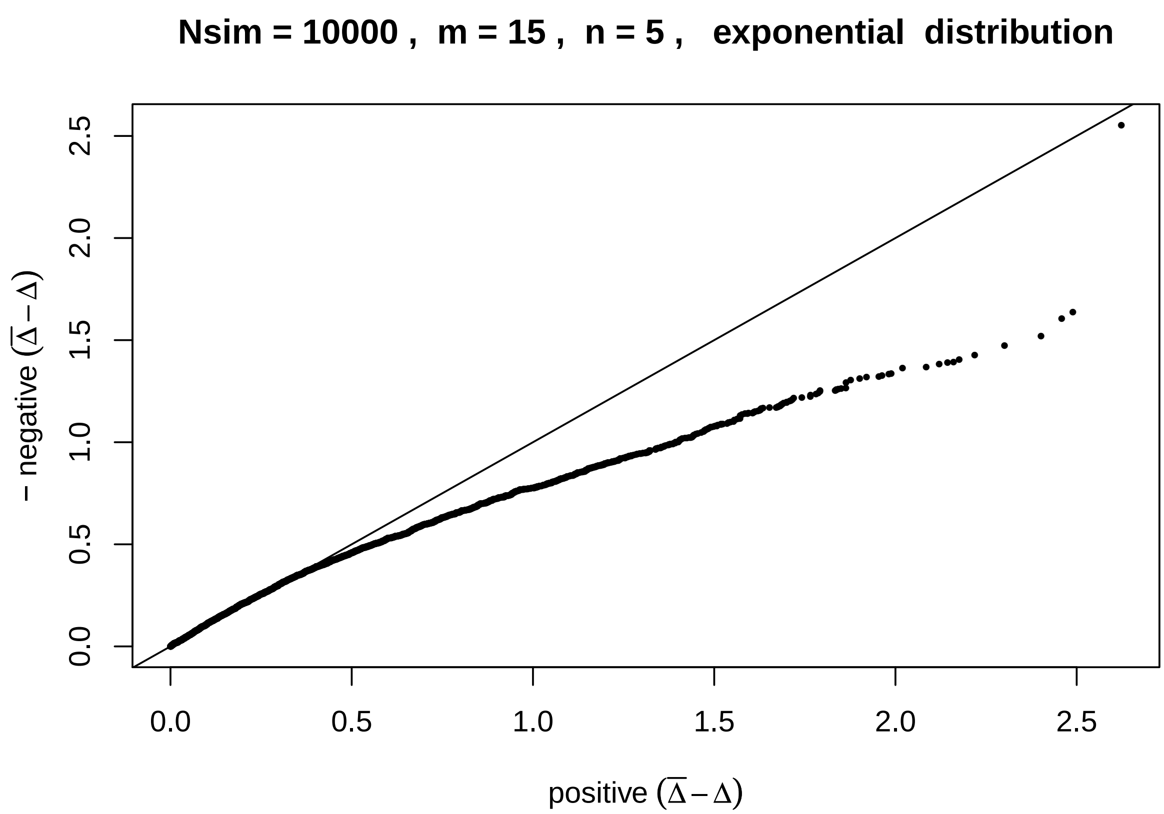
The desired plots follow, clearly illustrating the symmetry and asymmetry. Deviations from the main diagonal at the upper end are natural fluctuations. By repeating these plots several times you would learn to appreciate that.
The plots were generated by the following calls to `Delta.est.sim`, followed by the respective optional output.

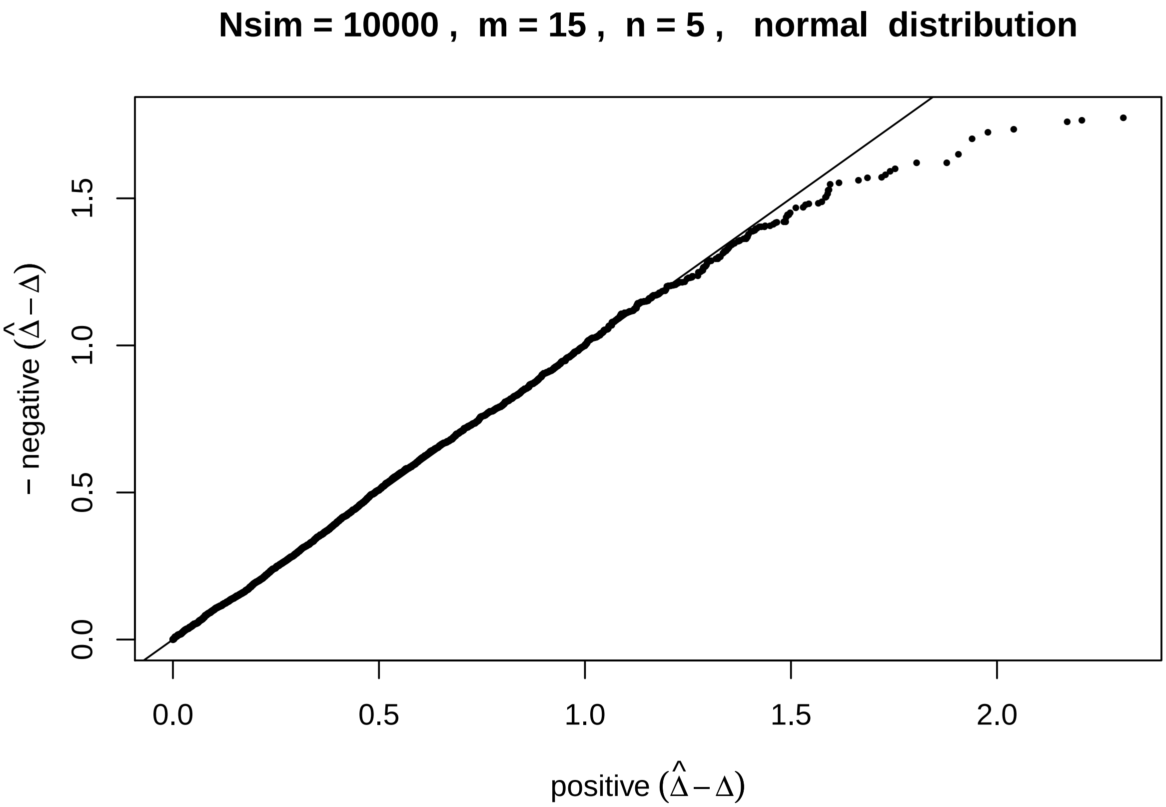
```
> Delta.est.sim(15,5,10000,rnorm,1,"normal",2,T)
hit return

   mean.Delta.bar           mean.HL p.above.Delta.bar p.below.Delta.bar
         2.008618          2.011785          0.509800          0.490200
       p.above.HL        p.below.HL
         0.513400          0.486600
m!=n distribution F is normal, symmetric:
Both estimators appear to be unbiased and median unbiased.
> Delta.est.sim(15,5,10000,rexp,1,"exponential",2,T)
hit return

   mean.Delta.bar           mean.HL p.above.Delta.bar p.below.Delta.bar
         1.997260          2.042722          0.461300          0.538700
       p.above.HL        p.below.HL
         0.495500          0.504500
m!=n distribution F is exponential, not symmetric:
Here Delta.bar is unbiased but not median unbiased,
HL appears to be median unbiased but not unbiased.
> Delta.est.sim(15,15,10000,rexp,1,"exponential",2,T)
hit return

   mean.Delta.bar           mean.HL p.above.Delta.bar p.below.Delta.bar
         1.998899          2.000587          0.497700          0.502300
       p.above.HL        p.below.HL
         0.499000          0.501000
illustrating both forms of unbiasedness for both estimators because m=n,
even though the exponential distribution is not symmetric
```

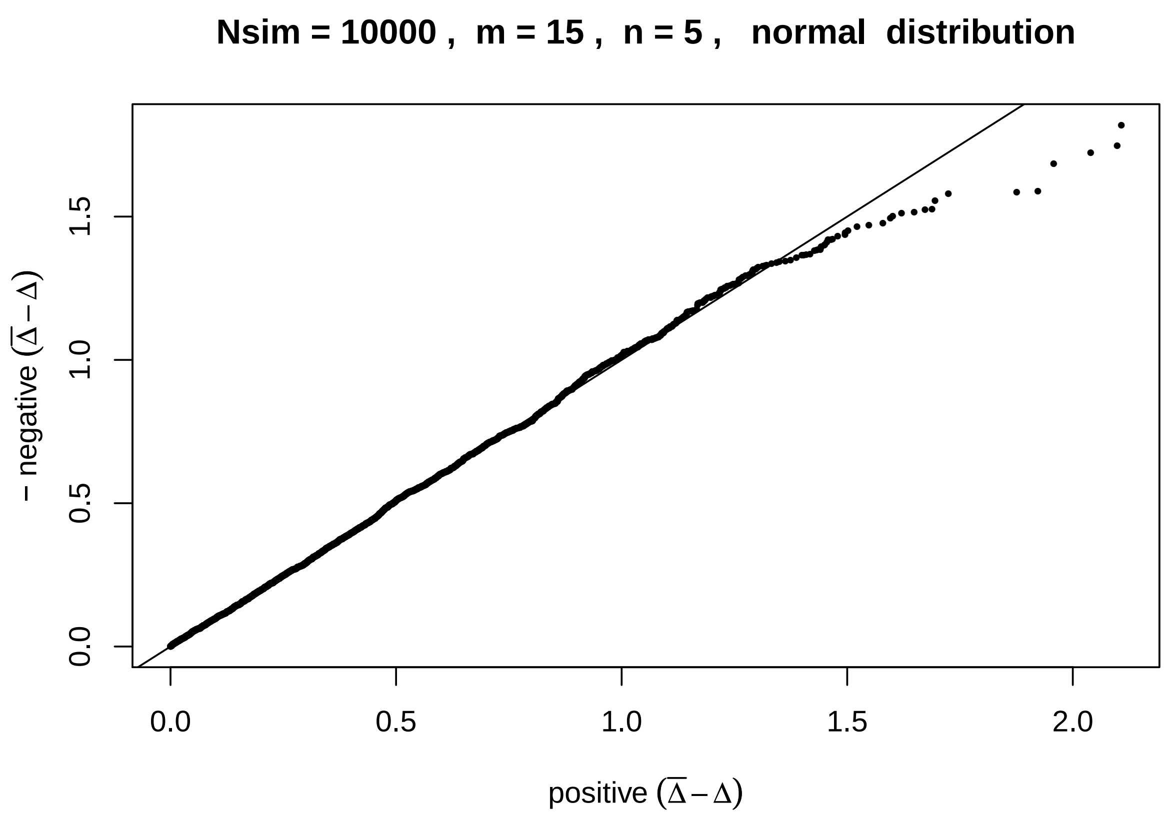

Nsim = 10000 , m = 15 , n = 15 , exponential distribution
− negative $(\hat{\Delta} - \Delta)$
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0.0
0.5
1.0
1.5
positive $(\hat{\Delta} - \Delta)$


Nsim = 10000 , m = 15 , n = 15 , exponential distribution
− negative $(\bar{\Delta} - \Delta)$
0.0
0.5
1.0
1.5
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
positive $(\bar{\Delta} - \Delta)$


Nsim = 10000 , m = 15 , n = 5 , exponential distribution
− negative $(\hat{\Delta} - \Delta)$
positive $(\hat{\Delta} - \Delta)$
0.0
0.5
1.0
1.5
2.0


Nsim = 10000 , m = 15 , n = 5 , exponential distribution
− negative $(\bar{\Delta} - \Delta)$
positive $(\bar{\Delta} - \Delta)$
0.0
0.5
1.0
1.5
2.0
2.5


Nsim = 10000 , m = 15 , n = 5 , normal distribution
− negative ($\hat{\Delta} - \Delta$)
0.0
0.5
1.0
1.5
0.0
0.5
1.0
1.5
2.0
positive ($\hat{\Delta} - \Delta$)


Nsim = 10000 , m = 15 , n = 5 , normal distribution
− negative ($\bar{\Delta} - \Delta$)
0.0
0.5
1.0
1.5
0.0
0.5
1.0
1.5
2.0
positive ($\bar{\Delta} - \Delta$)