Stat 425 HW3 Solution

Fritz Scholz

1. Describe in words what the following function does, in particular the distinction between `flag=T` and `flag=F`.

```
Initialize=function(flag=T,Nsim=10000){
if(flag==T){
   out=NULL}else{
   out=rep(0,Nsim)
}
for(i in 1:Nsim){
   out[i]=sum(rnorm(20))
}
}
```

Import this function into your R workspace (cut and paste should do). Time this function by executing the command

$$\texttt{system.time(Initialize(flag = F,Nsim = 10000))}$$

and do the same with `flag=T`. Repeat this with `Nsim=20000, 30000, 50000`. Plot the vector `y` of the respective execution times for `flag = T` against the `x` vector holding the corresponding `Nsim` values. You can add a second set of points with coordinate vectors `x1` and `y1` to an existing plot by `points(x1,y1)`. Thus you can plot the points corresponding to `flag=T` and `flag=F` on the same graph.

In another graph plot the square root (`sqrt(y)`) of the execution times obtained for `flag=T` against `x`. You can fit a line to a plot pattern created by plot(x,y) by doing out=lsfit(x,y), then out$coef gives you a vector of intercept and slope. The command abline(out) would add the fitted line to your plot. Only fit a line to those point patterns that look approximately linear.

Project how long you would have to wait to run each (`flag=T` and `flag=F`) for `Nsim=100000, 1000000`. What have you learned from this, in particular with respect to calculating estimated *p*-values?

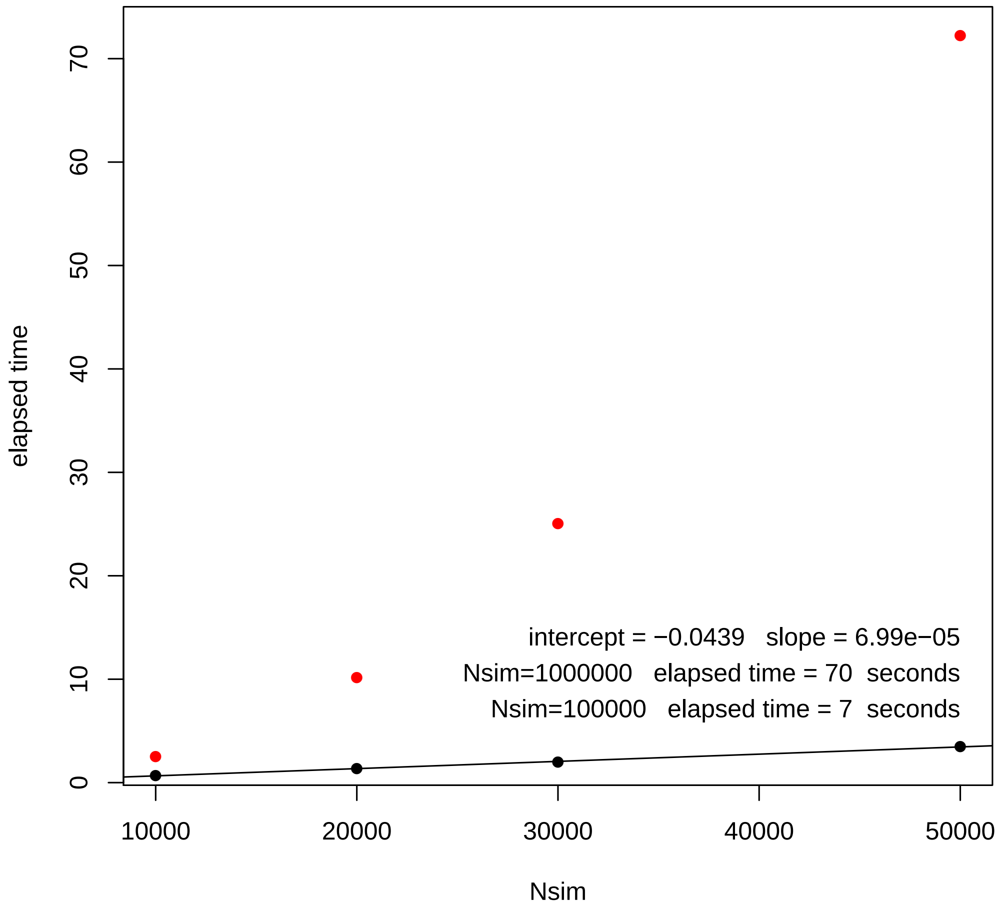
You can add R plots to your Word (or free Open Office Writer) document by activating the graphics window in R that shows the plot (i.e., click on it), go to File then Copy to the Clipboard, then choose as Metafile and then do a CTRL V in your Word (or free Open Office Writer) document.

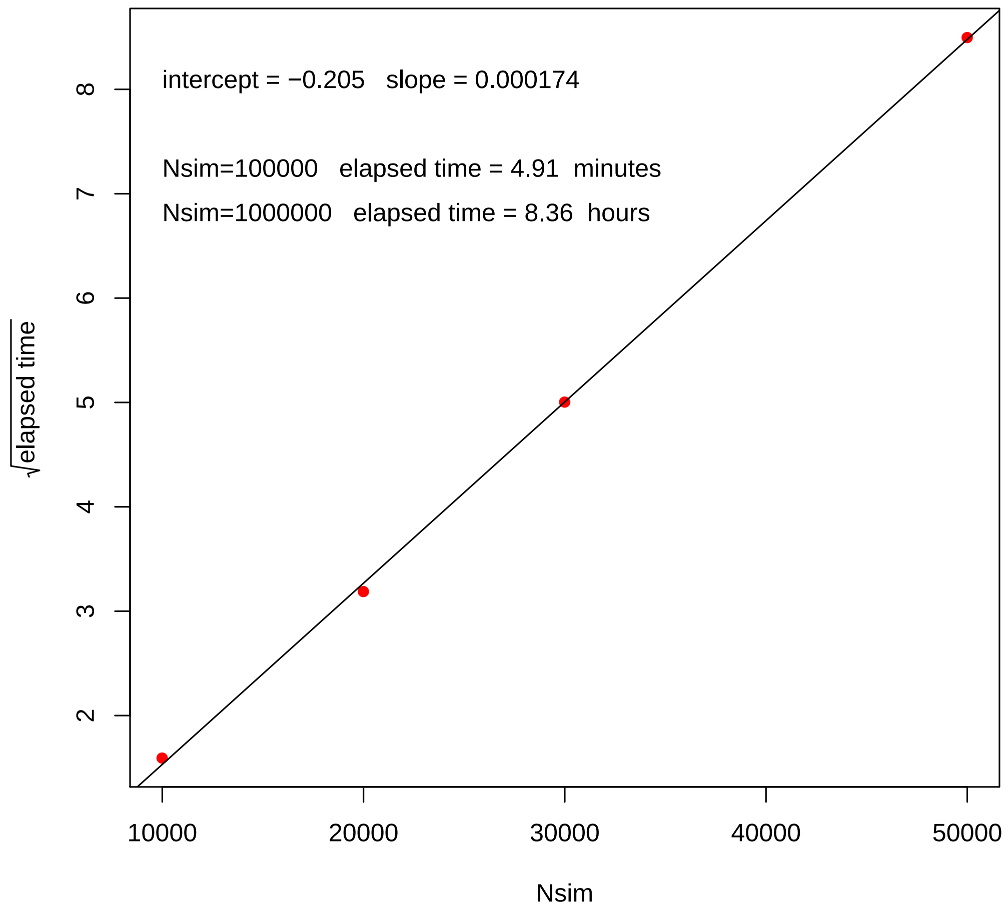
The function `Initialize` generates a vector `out` of length `Nsim` with independent sums of 20 standard normal random deviates for each position. It can do this two different ways depending on the input `flag`. If `flag=T` the vector out is initialzed as `out=NULL`, i.e., there is no predetermined space allocated for it. When `flag=F` the vector `out` is initialized to its full anticipated length `Nsim`.
The timing results on my laptop were as follows:

```
> system.time(Initialize(flag=F,Nsim=10000))
   user  system elapsed
  0.660   0.000   0.664
> system.time(Initialize(flag=F,Nsim=20000))
   user  system elapsed
  1.376   0.000   1.375
> system.time(Initialize(flag=F,Nsim=30000))
   user  system elapsed
  1.988   0.000   2.003
> system.time(Initialize(flag=F,Nsim=50000))
   user  system elapsed
  3.457   0.000   3.470
> system.time(Initialize(flag=T,Nsim=10000))
   user  system elapsed
  2.512   0.020   2.536
> system.time(Initialize(flag=T,Nsim=20000))
   user  system elapsed
  9.984   0.124  10.158
> system.time(Initialize(flag=T,Nsim=30000))
   user  system elapsed
 24.590   0.360  25.052
> system.time(Initialize(flag=T,Nsim=50000))
   user  system elapsed
 70.180   1.808  72.338
```

Of course, your elapsed time may be quite different.
The following plots are as requested and they also show the projections based on the fitted lines. These plots wer produced by

```
timing.plot=function(PDF=F){
if(PDF==T) pdf(file="timingF.pdf",width=7)
plot(Nsim,timeT,xlab="Nsim",ylab="elapsed time",pch=16,col="red")
points(Nsim,timeF,pch=16)
ls.out=lsfit(Nsim,timeF)
abline(ls.out)
text(max(Nsim),max(timeF)*4,paste("intercept =",
   signif(ls.out$coef[1],3),"  slope =",
signif(ls.out$coef[2],3)),adj=1)
text(max(Nsim),max(timeF)*2,paste("Nsim=100000", "  elapsed time =",
round(ls.out$coef[1]+ls.out$coef[2]*100000,0)," seconds"),adj=1)
text(max(Nsim),max(timeF)*3,paste("Nsim=1000000", "  elapsed time =",
round(ls.out$coef[1]+ls.out$coef[2]*1000000,0)," seconds"),adj=1)
if(PDF==T) dev.off()
readline("hit return\n")
if(PDF==T) pdf(file="timingT.pdf",width=7)
plot(Nsim,sqrt(timeT),xlab="Nsim",
  ylab=expression(sqrt("elapsed time")),pch=16,col="red")
ls.out=lsfit(Nsim,sqrt(timeT))
abline(ls.out)
text(min(Nsim),max(sqrt(timeT))*.95,paste("intercept =",
  signif(ls.out$coef[1],3),"  slope =",
  signif(ls.out$coef[2],3)),adj=0)
text(min(Nsim),max(sqrt(timeT))*.85,paste("Nsim=100000", "  elapsed time =",
round(((ls.out$coef[1]+ls.out$coef[2]*100000)^2)/60,2)," minutes"),adj=0)
text(min(Nsim),max(sqrt(timeT))*.8,paste("Nsim=1000000", "  elapsed time =",
round(((ls.out$coef[1]+ls.out$coef[2]*1000000)^2)/3600,2)," hours"),adj=0)
if(PDF==T) dev.off()
}
```





The simulations with preallocated memory space for the `out` vector appear to give elapsed times that are basically proportional to N_{sim} =`Nsim`.
In the simulation without allocated memory the elapsed time appears to grow quadratically with N_{sim}. One way to explain this is that each time we add a new element to `out` we have to reallocate memory for the previous $i-1$ elements and the new i^{th} element. This leads to

$$\sum_{i=1}^{N_{\text{sim}}} i = \frac{N_{\text{sim}}(N_{\text{sim}}+1)}{2}$$

processing steps, as opposed to allocating N_{sim} locations all at once, as done in the first scheme. Thus it is definitely of great advantage to allocate space for the simulation results upfront.

2. Write a function `Problem2=function(x,y,Nsim=30000)`{...} that computes the p-value for the two-sided Wilcoxon rank-sum test of the hypothesis of no treatment effect as indicated on slides 70-71 (Chapter 1). Write this as a function of two response vectors `x` and `y` (control and treatment) and of `Nsim` (see below).
Allow for possible ties and obtain an exact p-value by using `combn` as long as $\binom{m+n}{n}$ is not too large, say $\leq \texttt{Nsim} = 300000$. When the full enumeration exceeds `Nsim`, let the function estimate the p-value by sampling `Nsim` splits of the mid-rank vector and evaluating the treatment rank-sum each time.
In addition to the exact or estimated p-value this function should also produce the corresponding p-value obtained by normal approximation. Use the continuity correction if there are no ties, otherwise don't use it. You have no ties when `length(c(x,y))==length(unique(c(x,y))`.

Apply this test to the two vectors

```
x0=c(91, 94, 99, 99, 99, 100, 101, 102, 105, 106, 108)
```

```
y0=c(97, 102, 103, 104, 105, 107, 108, 111, 117)
```

Now run `wilcox.test(x0,y0)`. How do the results compare?

Repeat this with `wilcox.test(x1,y1)` and `Problem2(x1,y1,Nsim=300000)`, where

```
x1=c(94.1, 96.4, 101.55, 103.6, 100.51, 97.4, 110.69, 106.13, 101.61, 97.3, 100.33)
```

```
y1=c(104.59, 101.87, 105.84, 105.31, 108.72, 95.86, 102.35, 104.68, 102.21)
```

The code for `Problem2` follows. Note how the switch between exact and simulation evaluation is set.

```
Problem2=function(x,y,Nsim=300000){
Rz=rank(c(x,y))
n=length(y)
m=length(x)
N=m+n
Ws.star=sum(Rz[m+(1:n)])
D.obs=abs(Ws.star-(N+1)*n/2)
# the switch between full enumeration and simulation is
# set in the follwing if-block
if(choose(N,n)<=Nsim){
  out=combn(Rz,n,FUN=sum)
}else{
  out=rep(0,Nsim)
  for(i in 1:Nsim){
     out[i]=sum(sample(Rz,n,replace=F))
  }
}
D=abs(out-(N+1)*n/2)
p.val=mean(D>=D.obs)
mean.Wstar=n*(N+1)/2
var.Wstar=(n*(N-n)/N)*var(Rz)
# In the following if-block the normal approximation
# is adapted to the case with and without ties.
if( length(c(x,y))==length( unique(c(x,y)) ) ){
   p.val.norm=2*(1-pnorm((D.obs-.5)/sqrt(var.Wstar)))
}else{
   p.val.norm=2*(1-pnorm(D.obs/sqrt(var.Wstar)))
}
p=c(p.val,p.val.norm)
names(p)=c("p.val","p.val.norm")
p
}
```

The following shows the results of `Problem2` and `wilcox.test` for the two data sets. When there are no ties the results agree. When there are ties `wilcox.test` complains about not being able to compute exact *p*-values. By changing your code for `Problem2` so that it does a continuity correction in either case (ties or no ties) one can see that the *p*-value reported by `wilcox.test` uses a continuity correction. At least for our example here (`x0` and `y0`) that seems inappropriate.

```
> Problem2(x0,y0,Nsim=300000)
     p.val p.val.norm
0.03925935 0.03971414
> wilcox.test(x0,y0)

Wilcoxon rank sum test with continuity correction

data:  x0 and y0
W = 22.5, p-value = 0.04353
alternative hypothesis: true location shift is not equal to 0

Warning message:
In wilcox.test.default(x0, y0) : cannot compute exact p-value with ties
> Problem2(x1,y1,Nsim=300000)
     p.val p.val.norm
 0.1119433  0.1106121
>  wilcox.test(x1,y1)

Wilcoxon rank sum test

data:  x1 and y1
W = 28, p-value = 0.1119
alternative hypothesis: true location shift is not equal to 0
```