HW 461 1-15-14
R exercises on importance sampling. Copy and submit your R execution of this code.

1. What is P(Z > 10)? Simulating a large number of iid samples from N[0, 1] and taking the sample mean of indicator(Z > 10) will fail since few to none of the sample Z will exceed 10. If we instead take a large number of scores Z+10, around half will exceed 10 so we have large sample size and can obtain an unbiased estimator of P(Z>10) by taking the sample mean of indicator(Z+10 > 10) q(Z)/p(Z) where p is the probability density of N(10, 1) and q is the probability density of N(0, 1).

> q<-function(x){exp(-x^2 / 2)} # leave out root(2 pi) since it will cancel from q/p
> p<-function(x){exp(-(x-10)^2 / 2)} # verify q/p = exp(50-10^2*x)

> rnorm(2)
[1] -0.3501057  1.2940932

> mean(rnorm(1000000))
[1] 0.0008538581

> ck # see if ck name is being used by R
Error: object ‘ck’ not found

> ck<-function(x){ifelse(10<x,1,0)} # indicator of event x > 10.

> ck # R now knows ck.
function(x){ifelse(10<x,1,0)}

> ck(9.9999999999999999999999999999999)
[1] 0

> ck(10.0000000000000000000000000000001)
[1] 0

> ck(10.000000000000001)
[1] 1

> ck(10.000000000000001)
[1] 0

> mean(ck(rnorm(1000000))) # mean indicator of Z > 10 fails to est P(Z>10)
[1] 0

> fixck
Error: object ‘fixck’ not found
> fixck <- function(z) {ck(z+10)*q(z+10)/p(z+10)}

> mean(fixck(rnorm(1000000)))
[1] 7.577705e-24

> mean(fixck(rnorm(100000000)))
[1] 7.61949e-24

> q(10)/(sqrt(2*pi)*10)  # approximation P(Z > x) / (phi(x) / x) -> 1 as x -> infinity
[1] 7.694599e-24

> mean(fixck(rnorm(40000000)))  # simulation is not ~ known approximation?
[1] 7.619612e-24

> mean(fixck(rnorm(100000000)))  # still not getting known approximation?

# Here is Mathematica calculating approximation of P(Z > 10) to 25 places.

\[
\int_{10}^{\infty} \left(\frac{2}{\pi}\right)^{1/2} e^{-z^2/2} \, dz
\]

\[
\frac{1}{2} \text{Erfc}[5 \sqrt{2}]
\]

\[
\text{N}[%\,25]
\]

\[
7.619853024160526065973343 \times 10^{-24}
\]

Is R as accurate as Mathematica?

# Below is the above R code stripped of all prompts and results.
You can copy and paste it into R where the whole thing will execute when you hit return.

You’d best be careful about really large runs. If you let your computer run long enough it can eat ALL available memory (e.g. 16 Gb) and refuse to respond or close. If you copy commands while it is in that state you can corrupt what seem to be innocent text commands. That code will not run when pasted into R which may then refuse to respond or close although your temporary memory will not be affected.

Somehow R thinks it has unfinished business and won’t run that code. The condition cleared up for me when I shut R, reopened and typed the commands from scratch. The R commands I’ve included here have been retyped and should cause you no difficulty.

The Mathematica above has nothing to do with R. R can also give you an approximation of \( P(Z > 10) \). How? Do it and use the Mathematica calculation of \( P(Z > 10) \) shown above to find out how accurately R computes it.

```r
q <- function(x) {exp(-x^2 / 2)}  # leave out root(2 pi) since it will cancel from q/p
p <- function(x) {exp(-(x-10)^2 / 2)}  # verify q/p = exp(50-10*x)
rmnorm(2)
mean(rnorm(1000000))
ck
ck <- function(x) {ifelse(10 < x, 1, 0)}
ck
ck(9.99999999999999999999999999999999)
ck(10.00000000000000000000000000000001)
ck(10.00000000000000000000000000000001)  # 15 places
ck(10.00000000000000000000000000000001)  # 16 places
mean(ck(rnorm(1000000)))  # the mean indicator of Z > 10 fails
fixck
fixck <- function(z) {ck(z+10) * q(z+10) / p(z+10)}
mean(fixck(rnorm(1000000)))
mean(fixck(rnorm(1000000)))
q(10) / (sqrt(2*pi)*10)  # approximation \( P(Z > x) / \phi(x) / x \) -> 1 as x -> infinity
```
# THE COMMANDS THAT FOLLOW ARE DISCUSSED IN (2.) BELOW.

```r
meanck<-function(z){(z+10)*ck(z+10)*q(z+10)/p(z+10)}
zmillion<-rnorm(1000000)
mean(meanck(zmillion))/mean(fixck(zmillion))
x<- -log(runif(1000000))  # create vector of one million -log(U) ~ exponential(1)
c(mean(x), sd(x))  # create a vector of two elements mean, standard deviation of x
plot(density(x))
```

2. Continuation of exercise 1. To show the flexibility of these methods let’s look at the conditional mean of $Z$ conditional on the event $Z > 10$. Of course it is just the mean of the $N[0,1]$ tail divided by $P(Z > 10)$. Here it is in Mathematica.

![Mathematica code](https://via.placeholder.com/150)

# Here is what the importance sampling finds.

```r
> meanck<-function(z){(z+10)*ck(z+10)*q(z+10)/p(z+10)}
> zmillion<-rnorm(1000000)
> mean(meanck(zmillion))/mean(fixck(zmillion))  # IS estimate of $E(Z \mid Z > 10)$
[1] 10.09825
```

Here is what Mathematica gives

![Mathematica code](https://via.placeholder.com/150)

You have to be careful with Mathematica too. Look at this.
Continuation of exercise 1.

To show the flexibility of these methods let's look at the conditional mean of $Z$ conditional on the event $Z > 10$. Of course it is just the mean of the $N[0,1]$ tail divided by $P(Z > 10)$. Here it is in Mathematica.

```
# Here is what the importance sampling finds.

meanck<-function(z){(z+10)*ck(z+10)*q(z+10)/p(z+10)}
zmillion<-rnorm(1000000)
mean(meanck(zmillion))/mean(fixck(zmillion))

[1] 10.09825
```

Here is what Mathematica gives.

You have to be careful with Mathematica too. Look at this.

Inverse Probability Integral Transform will be used to generate one million iid random samples from the exponential density $\exp(-x)$, $\forall x > 0$ ($\text{mean} = \text{var} = 1$).

```
x<- -log(runif(1000000)) # create vector of one million -log(U) ~ exponential(1)
c(mean(x), sd(x)) # create a vector of two elements mean, standard dev of x
plot(density(x))
```

Explain why the density plot of the sample data has that hook at the left.