Class Example 1 – A Discrete Distribution

Define the discrete distribution with probability mass function,

$$p(k) = e^{-3}\frac{3}{k!}, \quad \text{for } k = 0, 1, 2, \dots$$

1. It can be shown that,

   $$\sum_{k=0}^{\infty} \frac{\lambda^k}{k!} = e^{\lambda}$$

   Use this to argue now that $p(\cdot)$ is a probability mass function.

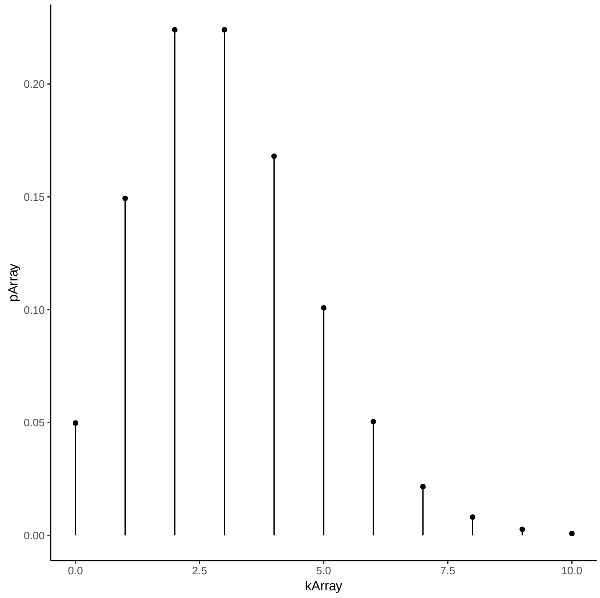
   Solution: It is clear that $p(k) \geq 0$ and further the probabilities add up to 1:

   $$\sum_{k=0}^{\infty} p(k) = e^{-3}\sum_{k=0}^{\infty}\frac{3^k}{k!} = e^{-3}e^{3} = 1.$$

2. Plot the probability mass function, for $k = 0, 1, 2, ..., 10$.

   Solution:

```
> library(tidyverse)
> kArray = seq(0:10)-1
> pArray = exp(-3)*3^kArray/factorial(kArray)
> Arrays = tibble(kArray,pArray)
>
> ggplot(Arrays,aes(x=kArray,y=pArray))+geom_point()+
+ geom_segment(aes(x=kArray, xend=kArray, y=0, yend=pArray)) +
+ theme_classic()
```



3. It is well known that the mean of this (Poisson distribution) is 3. Use R to compute,

   $$\sum_{k=0}^{10} kp(k) \approx \sum_{k=0}^{\infty} kp(k) = 3.$$

```
> sum(kArray*pArray)

[1] 2.996693
```

4. It is also well known that the variance is also 3 (for this type of distribution the mean equals the variance). Use R to compute,

$$\sum_{k=0}^{10}(k-3)^2 p(k) \approx 3$$

```
> sum((kArray-3)^2*pArray)

[1] 2.979679
```

Class Example 2 – A continuous distribution

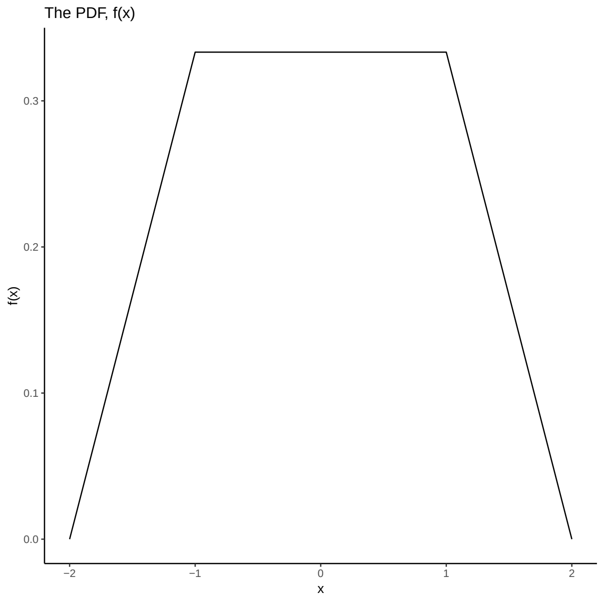
Consider a continuous distribution with probability density function (pdf) on $[2, 2]$,

$$f(x) = \begin{cases} \frac{2}{3} + \frac{1}{3}x, & x \in [-2, -1], \\ \frac{1}{3}, & x \in (-1, 1), \\ \frac{2}{3} - \frac{1}{3}x, & x \in [1, 2]. \end{cases}$$

Assume the random variable X is distributed according to $f(x)$

1. Plot the PDF of X in R

```
> library(tidyverse)
>
> f <- function(x){
+         if(-2 <= x && x <= -1) {
+                 return(2/3+1/3*x)
+         } else if(-1 < x && x < 1) {
+                 return(1/3)
+         } else if(1 <= x && x <= 2) {
+                 return(2/3-1/3*x)
+         } else {
+                 return(0)
+         }
+ }
>
> x=seq(-2,2,length=1001)
> y = sapply(x,f)
> plotArray = tibble(x,y)
>
> ggplot(plotArray,aes(x,y)) + geom_line()+
+ labs(title="The PDF, f(x)",x="x",y="f(x)") +
+ theme_classic()
```



2. Plot the CDF of X in R

   In order to do this we will numerical integrate the PDF. Remember that the density $f(x)$ has the following meaning for small δx:

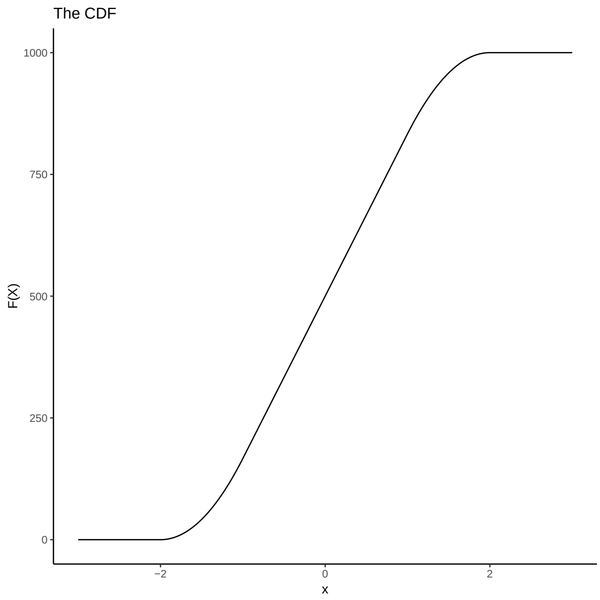
$$f(x)\delta x \approx P(X \in [x, x + \delta x])$$

   Hence

$$F(X) = P(X \leq x) = \int_{-2}^{x} f(u)du \approx \sum_{n=-2} x f(u)\delta x.$$

   We carry out the sum on the right in R. Here we choose $\delta x = 0 : 001$:

```
> F <- function(x) {
+         u = seq(-3,x,0.001)
+         sum(sapply(u,f))
+ }
>
> x= seq(-3,3,length=1001)
> y= sapply(x,F)
> plotArray = tibble(x,y)
>
> ggplot(plotArray,aes(x,y)) + geom_line() +
+ labs(title="The CDF",x="x",y="F(X)") +
+ theme_classic()
```



   Note that here we numerically integrated $f(x)$, but we can also do it analytically in this case. This would require integrating each part of $f(x)$ separately.

3. Looking at the distribution, what is it's mean?

   (i) Argue why the mean is 0 by looking at the PDF.

   (ii) Calculate the mean analytically.

Solution:
We see that $f(x) = f(-x)$ (it is a symmetric function). Now,

$$\int_{-2}^{2} xf(x)\ dx = \int_{-2}^{0} xf(x)\ dx + \int_{0}^{2} xf(x)\ dx$$

Now changin variable in the first integeral ($u = -x$) we get,

$$= \int_{0}^{2} (-u)f(-u)\ du + \int_{0}^{2} xf(x)\ dx$$

Now using the symmetry of f,

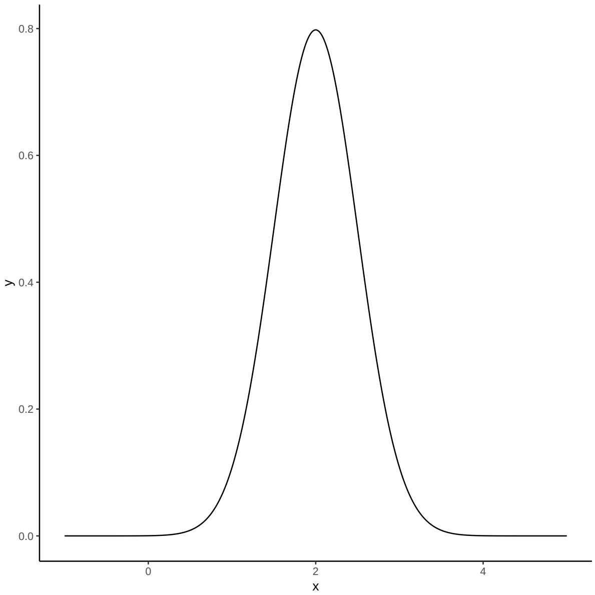
$$= -\int_{0}^{2} uf(u)\ du + \int_{0}^{2} xf(x)\ dx = 0$$

Class Example 1 – Common Families of Distributions

We now explore the Discrete Uniform, Binomial, Exponential and Normal distributions using R. For each such distribution, do the following:

(a) Create a variable representing the distributions with some parameters of your choice (within the valid parameter range). Here is an example of a Normal distribution with $\mu = 2$ and $\sigma = 0.5$

```
> xseq<-seq(-1,5,.001)
> normDist <- dnorm(xseq,2,0.5)
> ggplot(tibble(x=xseq,y=normDist),aes(x,y))+geom_line() +
+ theme_classic()
```



(b) Use `mean()` to obtain the mean value. Compare it to the value as calculated using the formula in the course material.

```
> c(mean(normDist),2)
```

```
[1] 0.1666389 2.0000000
```

The mean here is not the same as the theoretical value as it is calculated using a simple mean rather than a weighted mean. If you calculate the mean as follows, you get a more sensible answer

```
> sum(xseq*normDist)*0.001
```

```
[1] 2
```

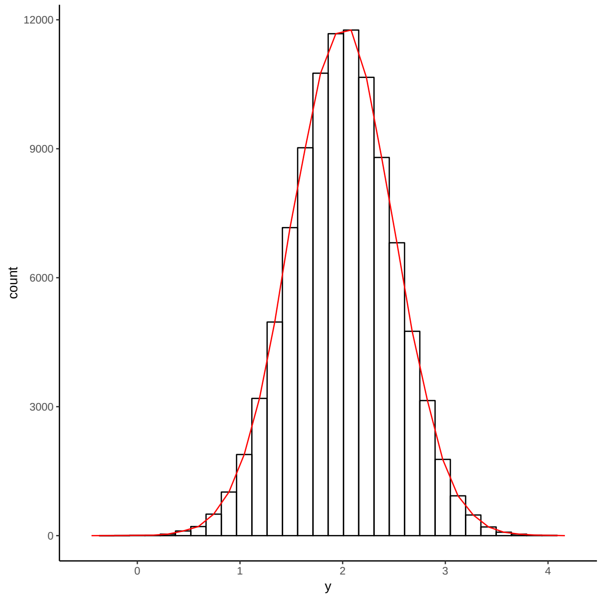
(c) Generate 100,000 random variables from the distribution and calculate the sample mean. Compare to (b) to ensure the values are close.

```
> randNorm = rnorm(10^5,2,0.5)
> c(mean(randNorm), sum(randNorm)/10^5)

[1] 2.000163 2.000163
```

Note that in the above the function `mean` performs the equivalent of `sum(...)/n`.

(d) Plot a histogram of the distribution taking care to present discrete and continuous distributions in an appropriate manner.

```
> #hist(randNorm)
> plotArray = tibble(y=randNorm)
> ggplot(plotArray,aes(y))+geom_histogram(colour="black",fill="NA")+geom_freqpoly(colou

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Solution: We now carry this out for all four families of distributions:

We first create an array of distributions. We then create a matrix of means for these distributions using three different methods.

We then plot the PMF/PDF of these four distributions over pre-defined domains, and finally we plot histograms of these four distributions, using a Monte Carlo sampling method.

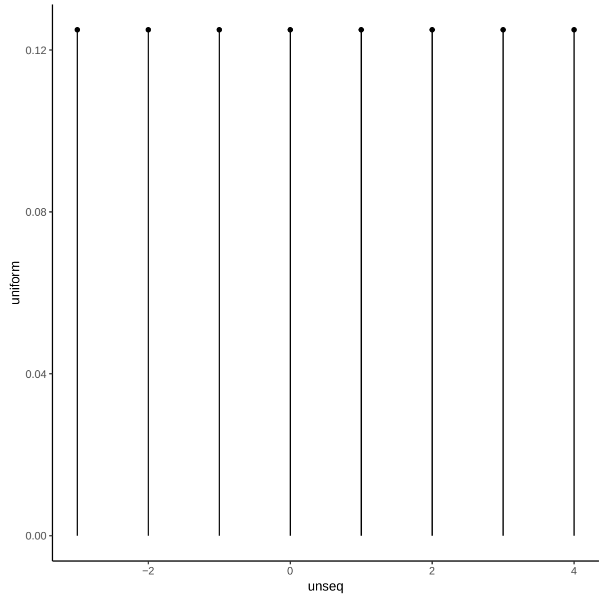
```
> # As Discrete Uniform does not exist in R
> dunifdisc<-function(x, min=0, max=1) ifelse(x>=min & x<=max & round(x)==x,
+  1/(max-min+1), 0)
> punifdisc<-function(q, min=0, max=1) ifelse(q<min, 0,
+  ifelse(q>=max, 1, (floor(q)-min+1)/(max-min+1)))
> qunifdisc<-function(p, min=0, max=1) floor(p*(max-min+1))
> runifdisc<-function(n, min=0, max=1) sample(min:max, n, replace=T)
```

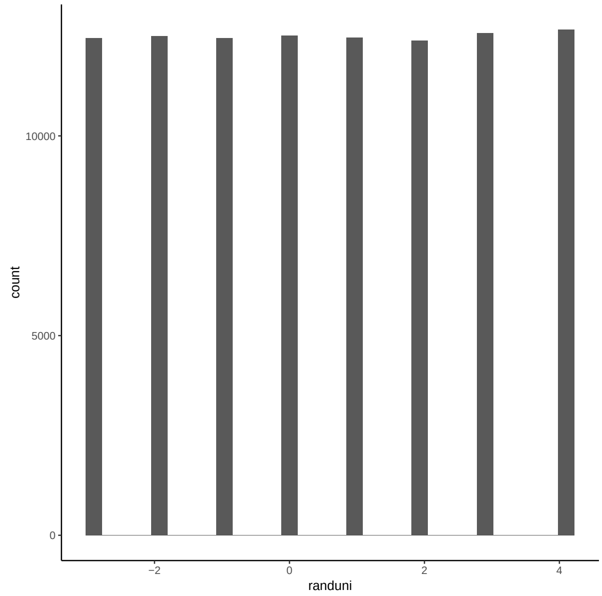
```
>
>
> unseq = seq(-3,4)
> biseq = seq(0,20)
> exseq = seq(0,20,0.001)
> noseq = seq(0,3,0.001)
>
> uniform<-dunifdisc(unseq,-3,4)
> binomial<-dbinom(biseq,20,0.25)
> exponential<-dexp(exseq,1/3)
> normal<-dnorm(noseq,1.5,0.25)
>
> meansFromFormula <- c((-3+4)/2,20*0.25,3,1.5)
> meansFromDistribution <- c(sum(unseq*uniform),sum(biseq*binomial),
+ sum(exseq*exponential)/1000,sum(noseq*normal)/1000)
>
> randuni = runifdisc(10^5,-3,4)
> randbin = rbinom(10^5,20,0.25)
> randexp = rexp(10^5,1/3)
> randnor = rnorm(10^5,1.5,0.25)
>
> meansFromMonteCarlo = c(mean(randuni),mean(randbin),mean(randexp),
+ mean(randnor))
>
> meansMatrix <- tibble(Distribution=
+ combine("DiscreteUniform(-3)","Binomial(20,0.25)","Exponential(3)",
+ "Normal(1.5,0.25)"), Formula=meansFromFormula,
+ FromDistribution=meansFromDistribution,MonteCarlo=meansFromMonteCarlo)
> kable(meansMatrix)
```

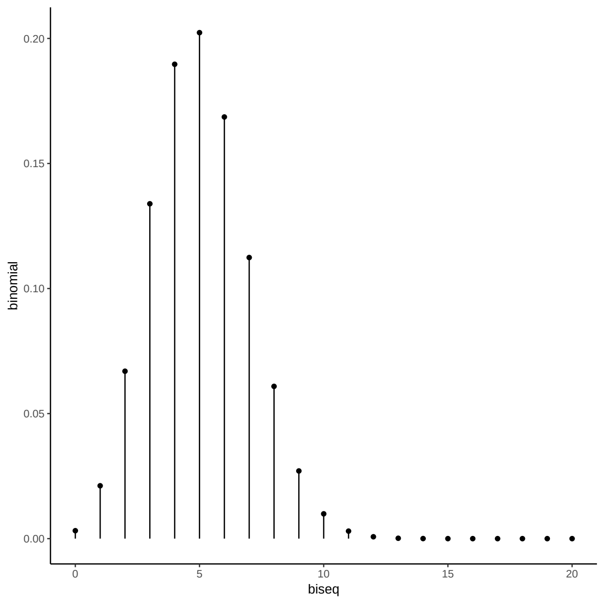
Distribution	Formula	FromDistribution	MonteCarlo
DiscreteUniform(-3)	0.5	0.500000	0.508190
Binomial(20,0.25)	5.0	5.000000	5.002590
Exponential(3)	3.0	2.970734	2.993150
Normal(1.5,0.25)	1.5	1.500000	1.500002

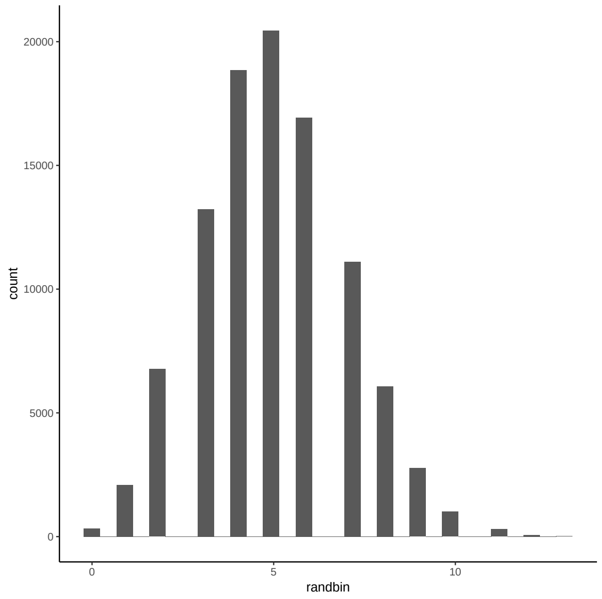
```
> ggplot(tibble(unseq,uniform),aes(x=unseq,y=uniform))+geom_point()+
+ geom_segment(aes(x=unseq,xend=unseq,y=0,yend=uniform))+
+ theme_classic()
> ggplot(tibble(randuni),aes(randuni))+geom_histogram() +
+ theme_classic()

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

> ggplot(tibble(biseq,binomial),aes(x=biseq,y=binomial))+
+ geom_point()+geom_segment(aes(x=biseq,xend=biseq,y=0,yend=binomial))+
+ theme_classic()
> ggplot(tibble(randbin),aes(randbin))+geom_histogram()+ theme_classic()

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

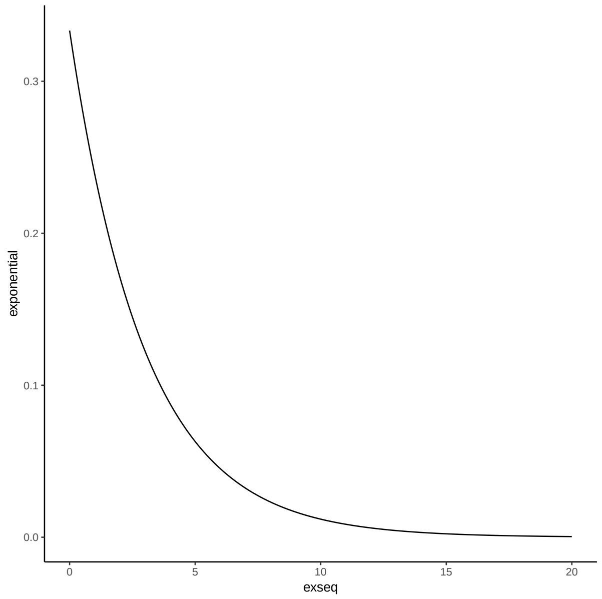
```
> ggplot(tibble(exseq,exponential),aes(exseq,exponential)) + geom_line()+
+ theme_classic()
> ggplot(tibble(randexp),aes(randexp))+geom_histogram()+ theme_classic()

 `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

> ggplot(tibble(noseq,normal),aes(noseq,normal)) + geom_line()+ theme_classic()
> ggplot(tibble(randnor),aes(randnor))+geom_histogram()+ theme_classic()

 `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

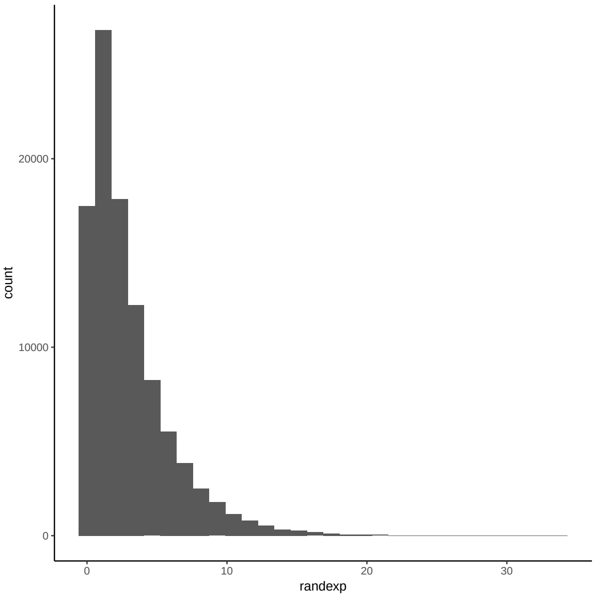


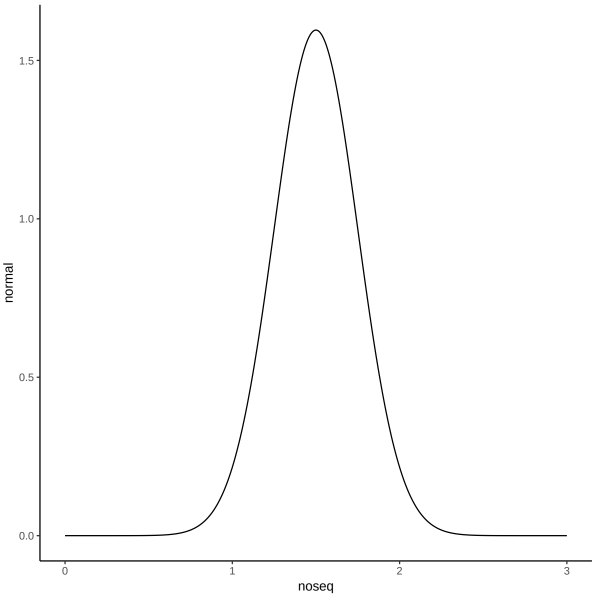


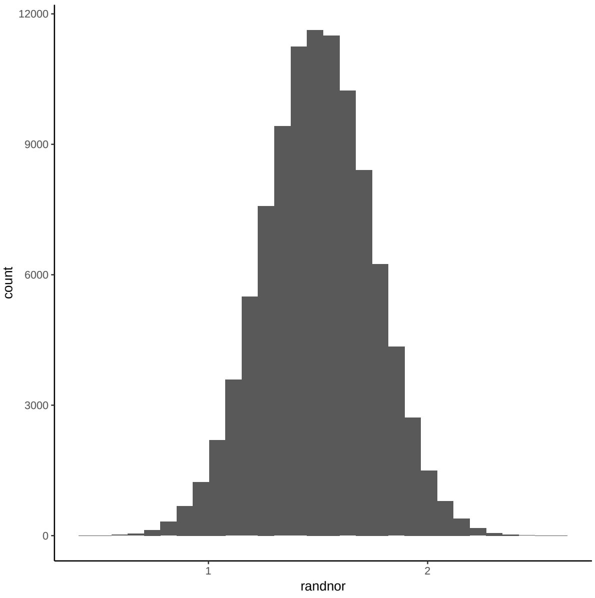

binomial
0.20
0.15
0.10
0.05
0.00
biseq
0
5
10
15
20


count
20000
15000
10000
5000
0
randbin
0
5
10


exponential
0.3
0.2
0.1
0.0
exseq
0
5
10
15
20


count
20000
10000
0
0
10
20
30
randexp


normal
1.5
1.0
0.5
0.0
0
1
2
3
noseq


count
12000
9000
6000
3000
0
1
2
randnor