Statistical Computing with R – MATH 6382[1,*]
Set 3 (Simulation)

Tamer Oraby

UTRGV

tamer.oraby@utrgv.edu

[1] Based on textbook.

[*] Last updated November 29, 2016

Generating a random number ...

from a general pmf

To generate random numbers from a discrete pmf, use `sample` in R
For example: To randomly generate 100 numbers of X with the pmf

$$f_X(x) = \begin{cases} .2 & \text{if } x = 2, \\ .6 & \text{if } x = 3, \\ .2 & \text{if } x = 5, \end{cases}$$

```
x<-sample(c(2,3,5),size=100,prob=c(.2,.6,.2),rep=TRUE)
> table(x)
x
 2  3  5
20 57 23
> table(x)/sum(table(x))
x
   2    3    5
0.20 0.57 0.23
```

Generating a random number from a common probability distribution

Generating a random number ...

from a common probability distribution

For any one of the common distributions (*dist*) like `binom, geom, nbinom, hyper, pois, unif, exp, gamma, beta, norm, lnorm, t, chisq, f, cauchy, weibull`

`r`*dist* generates random instances from *dist*

Use `help(rdist)` to find arguments for that function

Example:

```
> rnorm(3, mean = 0, sd = 1)
[1] 1.6362274 -1.7700555 0.4951518
```

Generating a random number ...

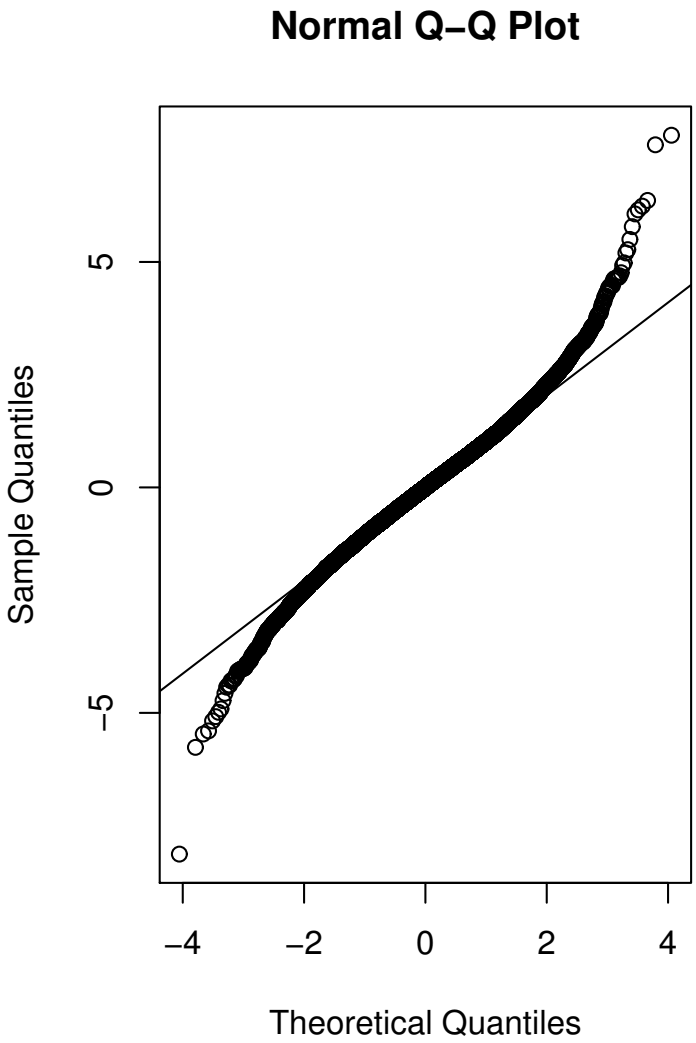
from a common probability distribution

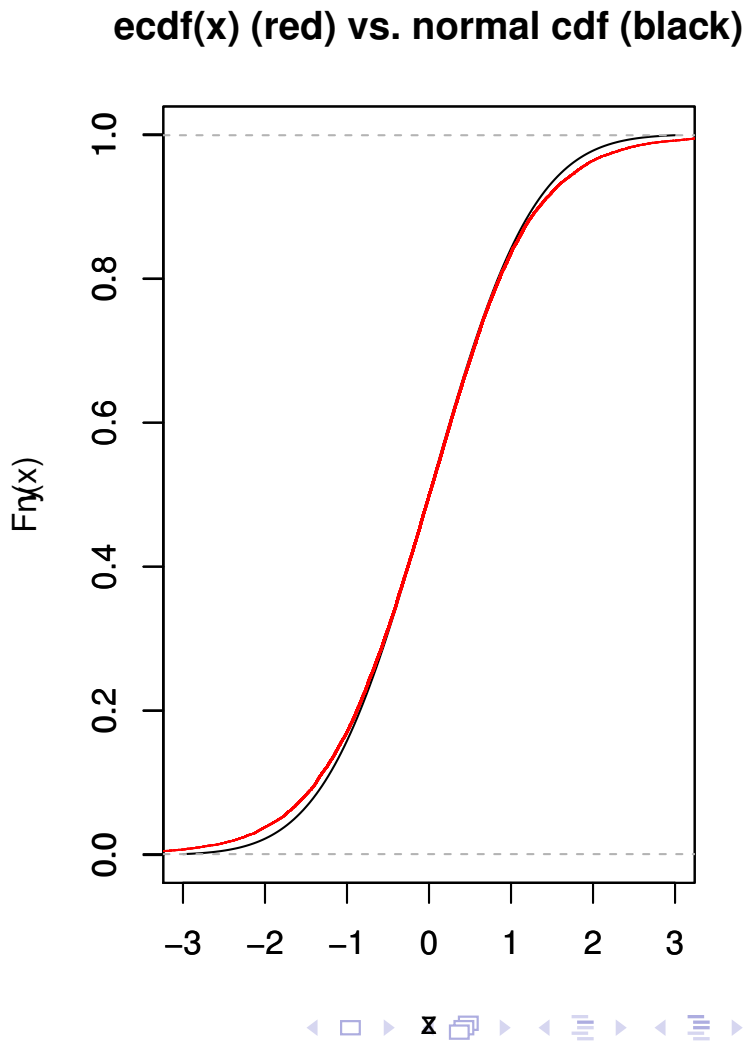
Example: Generate 20000 T random numbers with df=10

```
x <- rt(20000,df=10)
z<-seq(-3,3,.01)
y<-pnorm(z)
par(mfrow=c(1,2))
qqnorm(x);qqline(x)
plot(z,y,xlim=c(-3,3))
par(new=T)
plot.ecdf(x,xlim=c(-3,3),col="red",main="ecdf(x) vs.
normal cdf")
```

Generating a random number ...

from a common probability distribution





Generating a random number ...

from a common probability distribution

```
> x <- rt(20000,df=10)
> ks.test(x,"pnorm",0,1)
 One-sample Kolmogorov-Smirnov test
data:  x
D = 0.0159, p-value = 8.627e-05
alternative hypothesis:  two-sided
```

Generating a random number ...

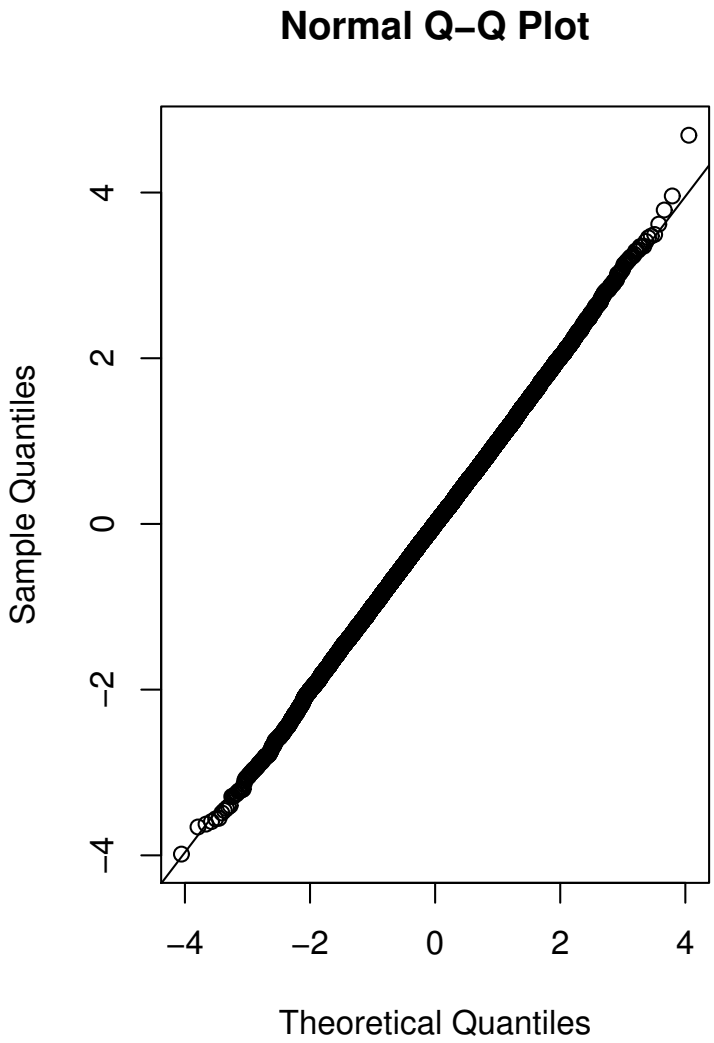
from a common probability distribution

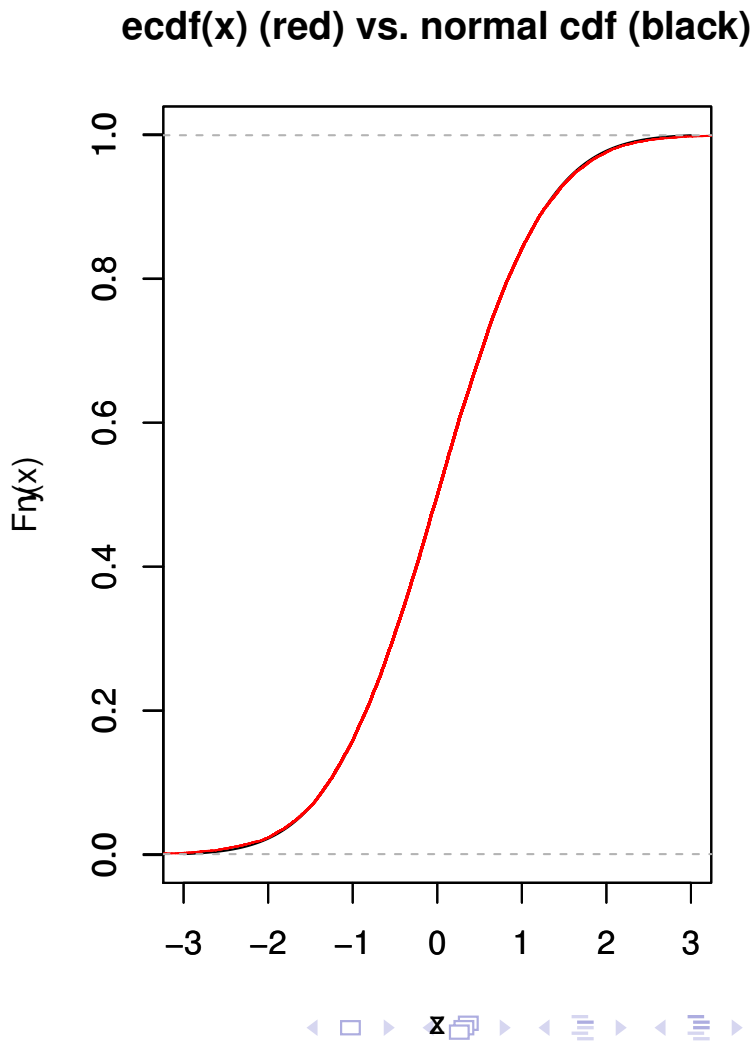
Example: Generate 20000 T random numbers with df=100

```
x <- rt(20000,df=100)
z<-seq(-3,3,.01)
y<-pnorm(z)
par(mfrow=c(1,2))
qqnorm(x);qqline(x)
plot(z,y,xlim=c(-3,3))
par(new=T)
plot.ecdf(x,xlim=c(-3,3),col="red",main="ecdf(x) vs.
normal cdf")
```

Generating a random number ...

from a common probability distribution





Generating a random number ...

from a common probability distribution

```
> ks.test(x,"pnorm",0,1)
 One-sample Kolmogorov-Smirnov test
data:  x
D = 0.0055, p-value = 0.5814
alternative hypothesis:  two-sided
```

Generating a random number ...

from a common probability distribution

```
> y<-ks.test(x,"pnorm",0,1)
> y$p
[1] 0.5813579
```

To fit the data to a t distribution

```
> library(MASS)
> fitdistr(x, "t")
        m             s             df
 0.002123809   0.996628933  74.293399393
 ( 0.007140446) ( 0.007546941) (31.368345433)
There were 23 warnings (use warnings() to see them)
```

Generating a random number from a general probability distribution

Generating a random number ...

from a general probability distribution

(1) Inverse Transform Method

Theorem

If X is a continuous random variable with cdf $F_X(x)$ then $U := F_X(X) \sim unif(0,1)$

Define a pseudo-inverse of F_X

$$F_X^{-1}(u) = \inf\{t : F_X(t) \geq u\} \text{ , 0<u<1}$$

since F_X is non-decreasing and right continuous.

$$\begin{aligned} F_{F_X^{-1}(U)}(x) = P(F_X^{-1}(U) \leq x) &= P(\inf\{t : F_X(t) \geq U\} \leq x) \\ &= P\left(F_X(\inf\{t : F_X(t) \geq U\}) \leq F_X(x)\right) \\ &\quad \text{since } F_X \text{ is non-decreasing} \\ &= P(U \leq F_X(x)) = F_U(F_X(x)) = F_X(x) \end{aligned}$$

Generating a random number ...

from a general probability distribution

(1) Inverse Transform Method

which means that

$$F_X^{-1}(U) \stackrel{\mathcal{D}}{=} X$$

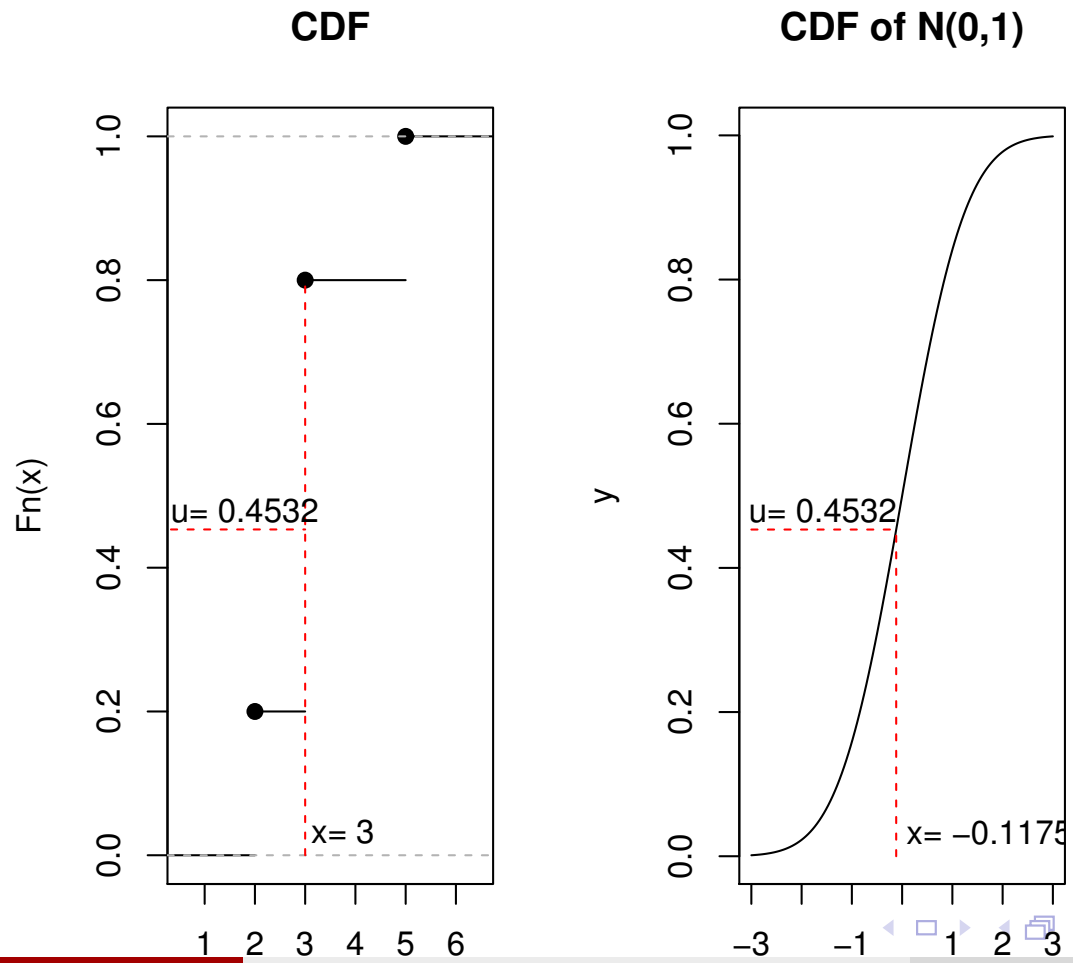
Algorithm:

1. Find F_X
2. Find F_X^{-1}
3. Generate a random number u from $unif(0,1)$
4. Return $x = F_X^{-1}(u) = \inf\{t : F_X(t) \geq u\}$

Generating a random number ...

from a general probability distribution

(1) Inverse Transform Method



Generating a random number ...

from a general probability distribution

```
> u<-runif(1,0,1)
> u
[1] .4532
par(mfrow=c(1,2))
plot.ecdf(c(2,2,3,3,3,3,3,3,5,5),main="CDF")
lines(c(0,3),c(.4532,.4532),xaxt='n',yaxt='n'
+,col="red",lty=2)
lines(c(3,3),c(0,.8),xaxt='n',yaxt='n'
+,col="red",lty=2)
text(1.8,.48,paste("u=",.4532))
text(3.75,.0325,paste("x=",3))
```

Generating a random number ...

from a general probability distribution

```
plot(z,y,xlim=c(-3,3), type="l",main="CDF of
N(0,1)")
lines(c(-3,qnorm(.4532)),c(.4532,.4532),xaxt='n'
+,yaxt='n',col="red",lty=2)
lines(c(qnorm(.4532),qnorm(.4532)),c(0,.4532)
+,xaxt='n',yaxt='n' ,col="red",lty=2)
text(-1.575,.48,paste("u=",.4532))
text(3.75,.0325,paste("x=",qnorm(.4532)))
```

Generating a random number ...

from a general probability distribution

(1) Inverse Transform Method

Example: Generate 1000 random numbers of X that has a pdf $f_X(x) = 3x^2$ for $0 < x < 1$.
Ans:

1. For $x \in (0, 1)$,

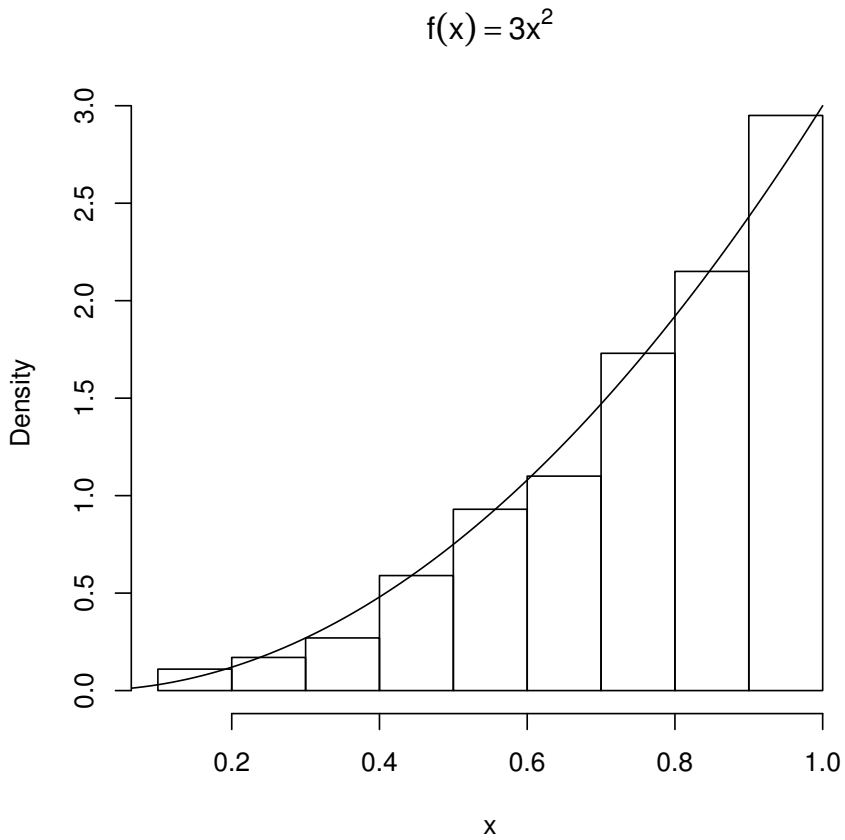
$$F_X(x) = \int_{-\infty}^{x} f_X(t)dt = \int_0^x 3t^2 dt = x^3$$

2. and so $F_X^{-1}(u) = \sqrt[3]{u}$
3.
```
u<-runif(1000)
x<-u^(1/3)
hist(x,prob=T,main=expression(f(x)==3*x^2))
y<-seq(0,1,.01)
lines(y,3*y^2)
```

Generating a random number ...

from a general probability distribution

(1) Inverse Transform Method



Generating a random number ...

from a general pmf

Example: To randomly generate 1 number of X with the pmf

$$f_X(x) = \begin{cases} .2 & \text{if } x = 2, \\ .6 & \text{if } x = 3, \\ .2 & \text{if } x = 5, \end{cases}$$

Ans:

$$F_X(x) = \begin{cases} 0 & \text{if } x < 2, \\ .2 & \text{if } 2 \leq x < 3, \\ .8 & \text{if } 3 \leq x < 5, \\ 1 & \text{if } 5 \leq x, \end{cases}$$

```
x<-c(2,3,5);probab<-c(.2,.6,.2)
CumF<-cumsum(probab)
u<-runif(1)
ind<-min(which(CumF>=u)) # or use which.min directly
x[ind]
```

Generating a random number ...

from a general pmf

Example: To randomly generate 100 number of X with the pmf

$$f_X(x) = \begin{cases} .2 & \text{if } x = 2, \\ .6 & \text{if } x = 3, \\ .2 & \text{if } x = 5, \end{cases}$$

```
n<-100
x<-c(2,3,5);probab<-c(.2,.6,.2)
CumF<-cumsum(probab)
u<-runif(n)
rx<-c()
for (i in 1:n){
rx[i]<-x[min(which(CumF>=u[i]))]}
```

But in a better way ...

Generating a random number ...

from a general pmf

Example: To randomly generate 100 number of X with the pmf

$$f_X(x) = \begin{cases} .2 & \text{if } x = 2, \\ .6 & \text{if } x = 3, \\ .2 & \text{if } x = 5, \end{cases}$$

```
n<-100
x<-c(2,3,5);probab<-c(.2,.6,.2)
CumF<-cumsum(probab)
u<-as.matrix(runif(n))
ind<-apply(u,1,function(y) min(which(CumF>=y)))
rx<-x[ind]
table(rx)/sum(table(rx)) # or similarity divide by n
rx
   2    3    5
0.15 0.63 0.22
```

Generating a random number ...

from a general pmf

Example: To randomly generate 100 number of X with the pmf

$$f(x) = \frac{6}{\pi^2}\frac{1}{x^2} \text{ for } x = 1, 2, \ldots$$

HW

Generating a random number ...

from a general probability distribution

(1) The Acceptance-Rejection Method

Theorem

If X and Y are random variables with pmf or pdf $f_X(x)$ and $f_Y(y)$ for which $\mathcal{S}_X \subset \mathcal{S}_Y$ and such that $\frac{f_X(k)}{f_Y(k)} \leq c$, where c is a constant, then

$$P\left(Y = k | U < \frac{f_X(Y)}{cf_Y(Y)}\right) = f_X(k)$$

where $U \sim unif(0,1)$

Note: 1) $U < \frac{f_X(Y)}{cf_Y(Y)}$ is called "Acceptance" condition.
2) Also, $c \geq 1$.

Generating a random number ...

from a general probability distribution

Proof: We will consider discrete case and you do continuous case.
By Law of Total Probability

$$
\begin{aligned}
P(\text{Acceptance}) &= P\left(U < \frac{f_X(Y)}{cf_Y(Y)}\right) \\
&= \sum_{k \in \mathcal{S}_Y} P\left(U < \frac{f_X(Y)}{cf_Y(Y)} | Y = k\right) P(Y = k) \\
&= \sum_{k \in \mathcal{S}_Y} F_U\left(\frac{f_X(k)}{cf_Y(k)}\right) f_Y(k) \\
&= \sum_{k \in \mathcal{S}_Y} \frac{f_X(k)}{cf_Y(k)} f_Y(k) \\
&= \frac{1}{c} \sum_{k \in \mathcal{S}_Y} f_X(k) = \frac{1}{c}
\end{aligned}
$$

Generating a random number ...

from a general probability distribution

(2) The Acceptance-Rejection Method

By Bayes' Theorem, (in the discrete case)

$$
\begin{aligned}
P\left(Y=k|U<\frac{f_X(Y)}{cf_Y(Y)}\right) &= \frac{P(U<\frac{f_X(Y)}{cf_Y(Y)}|Y=k)P(Y=k)}{P(\text{Acceptance})} \\
&= c\,P(U<\frac{f_X(k)}{cf_Y(k)}|Y=k)f_Y(k) \\
&= c\,F_U\left(\frac{f_X(k)}{cf_Y(k)}\right)f_Y(k) \\
&= c\,\frac{f_X(k)}{cf_Y(k)}f_Y(k)=f_X(k)
\end{aligned}
$$

Generating a random number ...

from a general probability distribution

(2) The Acceptance-Rejection Method

To generate a random number of X with support $\mathcal{S}_X$
Algorithm:

1. Find $f_Y(y)$ for which $\mathcal{S}_X \subset \mathcal{S}_Y$ and such that $\frac{f_X(k)}{f_Y(k)} \leq c$.
2. Generate a random number k of Y
3. Generate a random number u from $unif(0, 1)$
4. If Acceptance condition ($u < \frac{f_X(k)}{c f_Y(k)}$) is met, then return $x = k$, otherwise reject k and go to step 2.

Generating a random number ...

from a general probability distribution

(2) The Acceptance-Rejection Method

Note:

- Generating a random number of Y must be easy for the algorithm to work.
- The algorithm is a geometric random experiment in which rejection is a failure and acceptance is a success and

$$P(\text{success}) = P(\text{Acceptance}) = \frac{1}{c}.$$

Thus, the expected number of trials till the first success to generate one random number of X is c. Therefore, the smaller is the value of c, the faster the algorithm to generate the required sample.

Generating a random number ...

from a general pmf

Example: To randomly generate 100 number of X with the logarithmic pmf

$$f_X(x) = \frac{(1-e^{-\lambda})^x}{\lambda x} \text{ for } x = 1, 2, \ldots$$

with $\lambda > 0$.
Use $Y \sim geom(e^{-\lambda})$ in which

$$f_Y(y) = e^{-\lambda}(1-e^{-\lambda})^y \text{ for } y = 0, 1, 2, \ldots$$

Thus, for $y = 1, 2, \ldots$

$$\frac{f_X(y)}{f_Y(y)} = \frac{1}{y}\frac{e^{\lambda}}{\lambda} \leq \frac{e^{\lambda}}{\lambda}$$

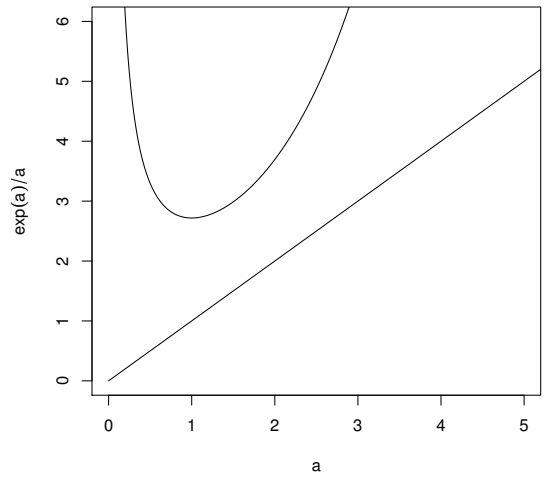
and so $c = \frac{e^{\lambda}}{\lambda}$ and the acceptance condition is $u < \frac{1}{y}$

Generating a random number ...

from a general pmf

Example: To randomly generate 100 number of X with the logarithmic pmf

```
a<-seq(0,6,.01)
plot(a,a,xlim=c(0,5),
+ type="l",ylab=expression(c=exp(a)/a))
lines(a,exp(a)/a,xlim=c(0,5))
```



Generating a random number ...

from a general pmf

Example: To randomly generate 100 number of X with the logarithmic pmf with $\lambda = 2$.
Since $c = \frac{e^2}{2} = 3.694528$ when we use $geom(e^{-2})$ as the proposal pmf then we expect that we need about 369 steps (369x2=738 random numbers) to generate 100 random numbers from $logarithmic(2)$.

```
n<-100
j<-1
x<-c()
while(length(x)<=n) {
k<-rgeom(1,exp(-2))
u<-runif(1)
if(k>=1 & u<1/k){ x<-c(x,k)}
j<-j+1 }
> j
[1] 396
```

and actually it used 396x2=792 random numbers

Generating a random number ...

from a general pmf

Remark: If $f(x) = af_0(x)f_1(x)$ is a target pdf with **unknown** normalization constant a and is given by the product of two functions. If $g(y) = bf_0(y)$ is a well-defined proposal pdf with normalization constant b and $f_1(x) \leq d$ then

$$1 = \int_{-\infty}^{\infty} f(x)dx \leq ad \int_{-\infty}^{\infty} f_0(x)dx = \frac{ad}{b}$$

Thus

$$\frac{f(x)}{g(x)} \leq \frac{ad}{b} := c$$

and so

$$\frac{1}{d} f_1(x) = \frac{f(x)}{cg(x)} \leq 1$$

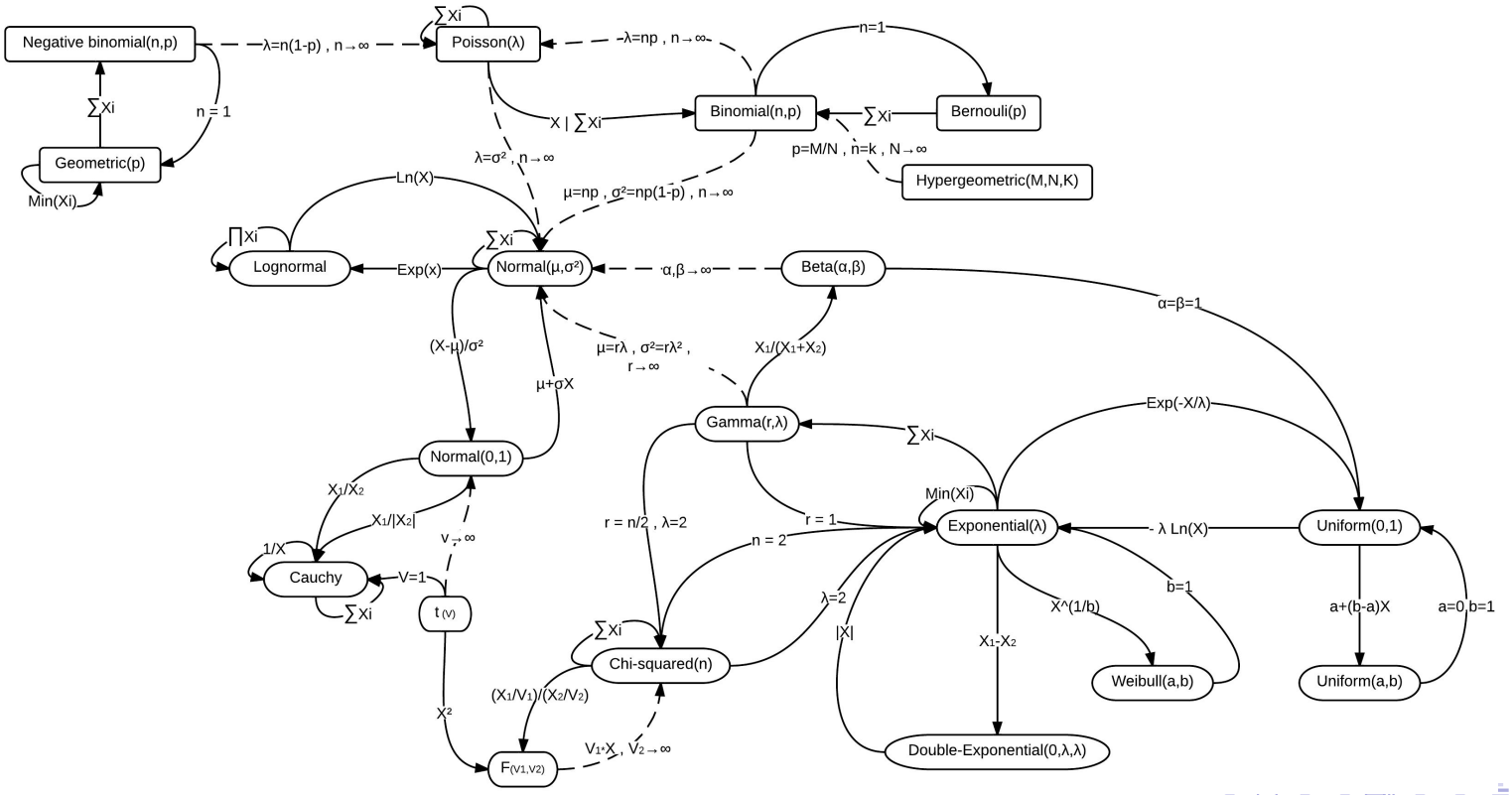
and the acceptance condition is $U < \frac{1}{d} f_1(Y)$ with Y generated by the proposal $g(y)$.

Generating a random number ...

from a general probability distribution

(3) Transformation Method

Using well-known relationships between random variables, we can generate random number from another one.



Generating a random number ...

from a general probability distribution

(3) Transformation Method

Example: If $X \sim gamma(\alpha, 1)$ and $Y \sim gamma(\beta, 1)$ are independent then

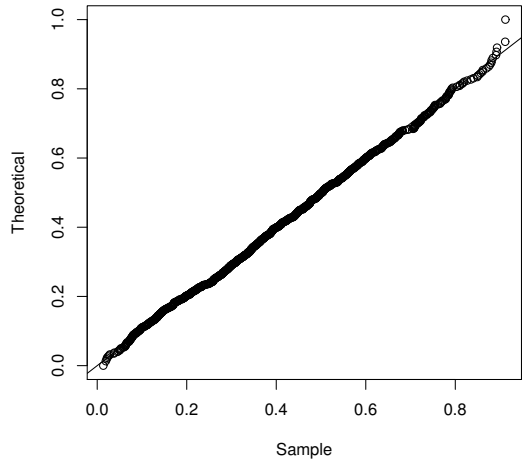
$$B = \frac{X}{X+Y} \sim beta(\alpha, \beta)$$

```
n<-1000
a<-2; b<-3
X<-rgamma(n,a,1)
Y<-rgamma(n,b,1)
B<-X/(X+Y)
```

Generating a random number ...

from a general probability distribution

```
qqplot(B,qbeta(seq(0,1,length=n),a,b)
+,xlab="Sample",ylab="Theoretical")
abline(0,1)
```




```
test<-ks.test(B,"pbeta",a,b)
test$p
[1] 0.4993225
```

Generating a random number from a convolution

Generating a random number ...

from a convolution of probability distributions

Let $X_1, X_2, \ldots, X_n$ be i.i.d.r.v. with cdf F_X
The distribution of $S_n = X_1 + X_2 + \cdots + X_n$ is the n-fold convolution $F_X^{*(n)}$
Recall:

$$(f * g)(x) = \int_{-\infty}^{x} f(y)g(x-y)dy$$

and $f^{*(n)} \equiv f * f * \cdots * f$ n-times.

Algorithm:

1. Generate one random number of each of $X_1, X_2, \ldots, X_n$
2. Add them up to generate one random number of S_n

Generating a random number ...

from a convolution of probability distributions

Example: Generate $n = 100$ numbers of *chisq*(15).
Let $X_1, X_2, \ldots, X_{15}$ be i.i.d.r.v. with *norm*(0, 1) then
$\chi^2_{15} = X_1^2 + X_2^2 + \cdots + X_{15}^2$ follows *chisq*(15).

```
n<-100
df<-15
x<-replicate(n,(rnorm(df))^2) # ncol=n and nrow=df
x<-colSums(x)
```

One can also produce the random matrix via

```
matrix(rnorm(df*n),df,n)
```

This algorithm requires generating 100 of each of the X's making a total of 1500 random number generations to produce 100 numbers of *Chi*(15).

Generating a random number from a mixture

Generating a random number ...

from a (discrete) mixture of probability distributions

Let $X_1, X_2, \ldots, X_K$ be r.v. with $X_i \sim F_{X_i}(\cdot|\theta_i)$
A discrete-mixture distribution of X is

$$F_X(\cdot|\theta) = p_1 F_{X_1}(\cdot|\theta_1) + p_2 F_{X_2}(\cdot|\theta_2) + \cdots + p_K F_{X_K}(\cdot|\theta_K)$$

where $p_i > 0$ and $p_1 + p_2 + \cdots + p_K = 1$
Algorithm:

1. Generate (using any algorithm) one random integer from the discrete probability distribution $f_P(i) = p_i$ for $i = 1, 2, \ldots, K$, say j
2. Generate (using any algorithm) one random number from F_{X_j}

Generating a random number ...

from a (discrete) mixture of probability distributions

Example: As a customer, you find $K = 6$ open cashiers with line lengths of 2, 3, 2, 1, 2, and 1. You are not sure which line to join so you select one of the six cashiers randomly (by casting a die in your brain). It takes a cashier a variable amount of time to serve a customer dependent on the size and type of the purchases and the cashier's skills. Based on statistical modeling, the store manager uses a model for the time to serve one customer given by exponential distribution with rates .3, .2, .5, .6, .2, .1 min^{-1} for the six cashiers, respectively. Analytically then by simulation, find the mean length of time you will take to finish and what is the probability to finish within 10 minutes? Assume that the length of time to serve two customers on the same cashier are independent.

Generating a random number ...

from a (discrete) mixture of probability distributions

Ans: By independence, if the time to serve a customer on cashier i is $exp(\lambda_i)$ then the time to serve r_i customers $X_i \sim gamma(r_i, \lambda_i)$. In this example $r_1 = 3$, $r_2 = 4$, $r_3 = 3$, $r_4 = 2$, $r_5 = 3$, and $r_6 = 2$ – in each of which we added you to the line – with rates $\lambda_1 = .3$, $\lambda_2 = .2$, $\lambda_3 = .5$, $\lambda_4 = .6$, $\lambda_5 = .2$, and $\lambda_6 = .1$. If X is the time for you to finish then X has a mixture of six gamma distributions with equal weights $\frac{1}{6}$. Thus,

$$F_X \equiv \frac{1}{6}F_{X_1} + \frac{1}{6}F_{X_2} + \cdots + \frac{1}{6}F_{X_6}$$

with mean
$\mathbf{E}(X) = \frac{1}{6}\sum_{i=1}^{6} \frac{r_i}{\lambda_i} = \frac{1}{6}(3/.3+4/.2+3/.5+2/.6+3/.2+2/.1) = 12.3889$
min and $F_X(10) = \frac{1}{6}F_{X_1}(10) + \frac{1}{6}F_{X_2}(10) + \cdots + \frac{1}{6}F_{X_6}(10) = .5275$

Generating a random number ...

from a (discrete) mixture of probability distributions

Ans: $F_X(10) = \frac{1}{6}F_{X_1}(10) + \frac{1}{6}F_{X_2}(10) + \cdots + \frac{1}{6}F_{X_6}(10) = .5275$ could be found using

```
r<-c(3,4,3,2,3,2)
lam<-c(.3,.2,.5,.6,.2,.1)
pcash<-pgamma(10,shape=r,rate=lam)
sum(pcash)/6
```

Generating a random number ...

from a (discrete) mixture of probability distributions

Ans: By simulation of n=1000 times for each one do

1. Sample one of the six digits $1, 2, \ldots, 6$; say it is j
2. Use $gamma(r_j, \lambda_j)$ to generate one number.

Note that the last step could be done as a convolution of r_j random numbers of $exp(\lambda_j)$, but let us keep it simple for now.

```
n<-1000
r<-c(3,4,3,2,3,2)
lam<-c(.3,.2,.5,.6,.2,.1)
K<-sample(1:6,n,rep=T)
x<-rgamma(n,shape=r[K],rate=lam[K])
mean(x)
[1] 12.27689
F<-ecdf(x)
F(10)
[1] 0.524
```

Generating a random number ...

from a (continuous) mixture of probability distributions

Let Y be r.v. with $Y \sim F_Y(\cdot|\theta, \lambda_1)$ and $\Theta \sim F_\Theta(\theta|\lambda_2)$
A continuous-mixture distribution of X is

$$F_X(\cdot|\lambda) = \int_{\mathbb{R}} F_Y(\cdot|\theta, \lambda_1)\, f_\Theta(\theta|\lambda_2)\, d\theta$$

where $f_\Theta(\theta|\lambda_2) > 0$ and $\int_{\mathbb{R}} f_\Theta(\theta|\lambda_2)\, d\theta = 1$
Algorithm:

1. Generate (using any algorithm) one random integer θ from the probability distribution $F_\Theta(\theta|\lambda_2)$, say θ_r
2. Generate (using any algorithm) one random number from $F_Y(\cdot|\theta_r, \lambda_1)$

Generating a random number ...

from a (continuous) mixture of probability distributions

Example: (Gamma-Poisson mixture)
Let $Y \sim pois(\lambda)$ and $\lambda \sim gamma(r, \beta)$. It is known analytically that the Gamma-Poisson mixture follows $nbiom(r, \frac{\beta}{1+\beta})$, since for each $x = 0, 1, \ldots$

$$
\begin{aligned}
f_X(x|r,\beta) &= \int_{\mathbb{R}} f_Y(x|\lambda)\, f_\Lambda(\lambda|r,\beta)\, d\lambda \\
&= \int_0^\infty \frac{\lambda^x}{x!} e^{-\lambda} \frac{\beta^r}{\Gamma(r)} \lambda^{r-1} e^{-\beta\lambda}\, d\lambda \\
&= \frac{\beta^r}{x!\Gamma(r)} \int_0^\infty \lambda^{x+r-1} e^{-(1+\beta)\lambda}\, d\lambda \\
&= \frac{\Gamma(x+r)}{x!\Gamma(r)} \frac{\beta^r}{(1+\beta)^{x+r}} \\
&= \frac{\Gamma(x+r)}{x!\Gamma(r)} \left(\frac{\beta}{1+\beta}\right)^r \left(\frac{1}{1+\beta}\right)^x
\end{aligned}
$$

Generating a random number ...

from a (continuous) mixture of probability distributions

Example: (Gamma-Poisson mixture)
If r is an integer, then for each $x = 0, 1, \ldots$

$$\begin{aligned} f_X(x|r,\beta) &= \frac{\Gamma(x+r)}{x!\Gamma(r)}\left(\frac{\beta}{1+\beta}\right)^r\left(\frac{1}{1+\beta}\right)^x \\ &= \frac{(x+r-1)!}{x!(r-1)!}\left(\frac{\beta}{1+\beta}\right)^r\left(\frac{1}{1+\beta}\right)^x \end{aligned}$$

which is $nbiom(r, \frac{\beta}{1+\beta})$. If $r > 0$ a real-number the it is called $polya(r, \mu)$ with $\beta = \frac{\mu}{r}$ which then has mean μ and variance $\mu + \frac{1}{r}\mu^2$. The parameter r (or its reciprocal) is called clustering, aggregation, heterogeneity, or over-dispersion parameter. As $r \to \infty$, $polya(r, \mu)$ approaches $pois(\mu)$.

Generating a random number ...

from a (continuous) mixture of probability distributions

Example: (Gamma-Poisson mixture)
Generate 100000 numbers from $Y \sim pois(\lambda)$ and $\lambda \sim gamma(.1, 10)$.
Algorithm:

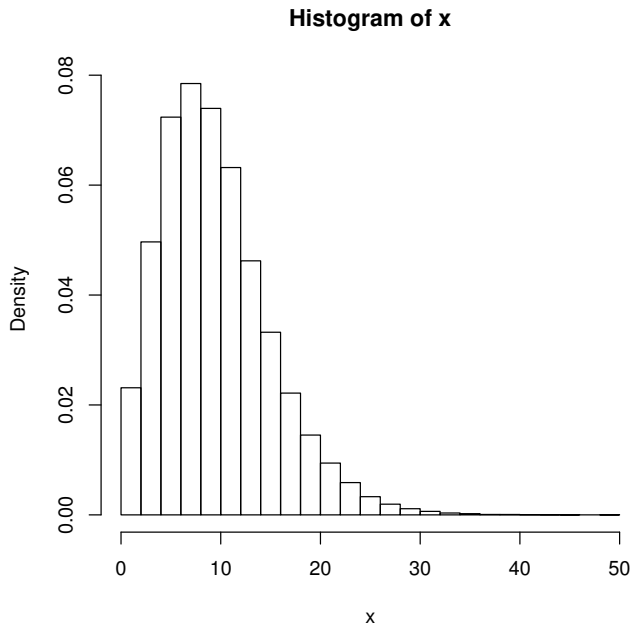
1. Generate $n = 100000$ random λ's using $rgamma(n, shape = 5, rate = .5)$
2. For each generated value of λ generate one? random number from $pois(\lambda)$

Generating a random number ...

from a (continuous) mixture of probability distributions

Example: (Gamma-Poisson mixture)

```
n<-100000;r<-5;beta<-.5
lambda<-rgamma(n,shape=r,rate=beta)
x<-rpois(n,lambda)
hist(x,prob=T)
```



Generating a random number ...

from data[1]

A method avoiding ecdf and good for vectors.
Thompson-Taylor Data-based Simulation: Given the data $x_1, x_2, \ldots, x_n$. Fix a value for the smoothing parameter m.
Algorithm:

1. Select one of the data points randomly, say x_j
2. Identify the closest m neighbors of x_j including itself, say $x_{j_1}, \ldots, x_{j_m}$, and calculate their average $\bar{x}_j = \frac{1}{m}\sum_{i=1}^{m} x_{j_i}$
3. Generate $u_1, \ldots, u_m$ from $unif(\frac{1}{m} - \sqrt{\frac{3(m-1)}{m^2}}, \frac{1}{m} + \sqrt{\frac{3(m-1)}{m^2}})$
4. Return $\bar{x}_j + \sum_{i=1}^{m} u_i(x_{j_i} - \bar{x}_j)$

[1] From "Computational Statistics" by Gentle p. 320

Simulating Multivariate Random Variables

Generating a random vector ...

from a multi-variate normal distribution

Use `mvrnorm(size,mean,covariance matrix)` after calling the library MASS.
Example: Generate n=100 random normal vectors of dimension 2 and means 2 and 3 and respective variances 25 and 16 with correlation .6.

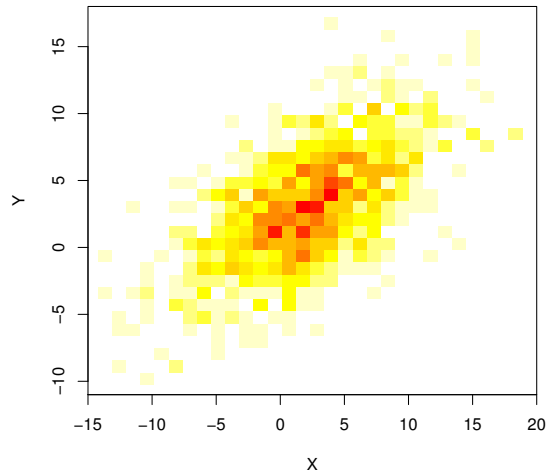
```
library(MASS)
n<-100
mu<-c(2,3)
sigma<-matrix(c(25,.6*sqrt(25)*sqrt(16)
,.6*sqrt(25)*sqrt(16),16),2)
x<-mvrnorm(n,mu,sigma)
```

Generating a random vector ...

from a multi-variate normal distribution

Install the package `gplots` once and for all.

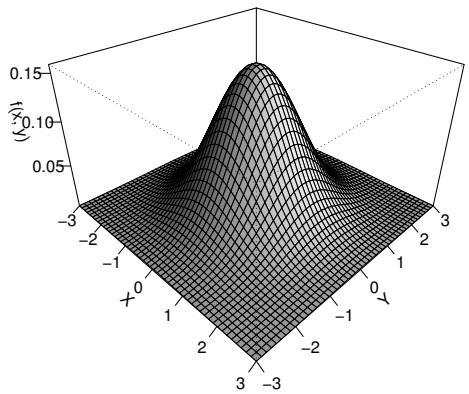
```
install.packages("gplots")
library(gplots)
hist2d(x,nbins=30,xlab=expression(X),
ylab=expression(Y)
,col=c("white",rev(heat.colors(30))),
xlim=c(-15,20),ylim=c(-11,18))
```



Generating a random vector ...

from a multi-variate normal distribution

```
f <- function(x,y) {
z <- (1/(2*pi)) * exp(-.5 * (x^2 + y^2))}
y <- x <- seq(-3, 3, length= 50)
z <- outer(x, y, f)
persp(x, y, z, theta = 45, phi = 30, expand = 0.6,
ltheta = 120, shade = 0.75, ticktype = "detailed",
xlab = "X", ylab = "Y", zlab = "f(x, y)")
```



Simulating Uniform Variable on the d-Sphere

Simulating Uniform Variable on the d-Sphere

To generate a uniform random point on a d-sphere, $(u_1, u_2, \ldots, u_d)$, the generate d independent $norm(0,1)$ random values $z = (z_1, z_2, \ldots, z_d)$ then normalize it by the ℓ_2 norm $\|z\|_2 = \sqrt{z_1^2 + z_2^2 + \cdots + z_d^2}$. So that $u_i = \frac{z_i}{\|z\|_2}$. Which makes $u_1^2 + u_2^2 + \cdots + u_d^2 = 1$.

To generate n points on a d-sphere

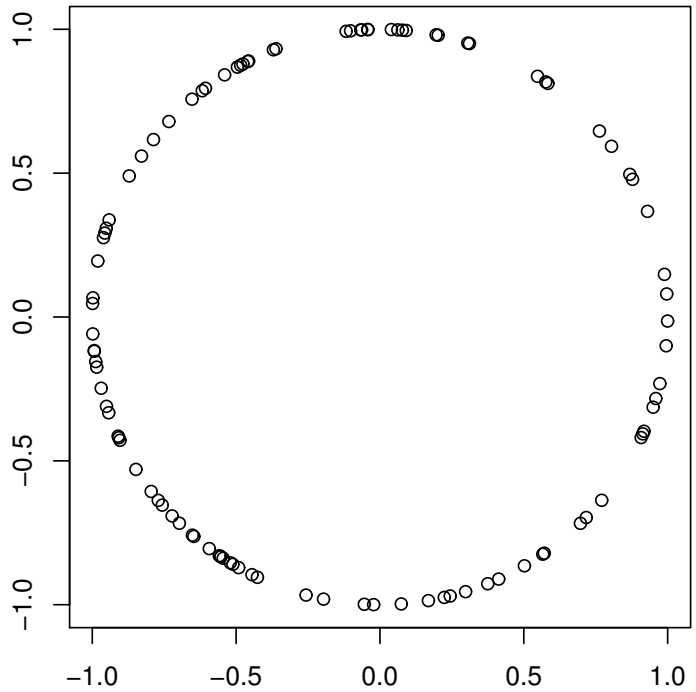
Algorithm:

1. Generate $n \times d$ random matrix of $norm(0,1)$
2. Divide each row by its ℓ_2 norm.
3. Each row is a d-dimensional vector of the coordinates of one point on the d-sphere.

Simulating Uniform Variable on the Circle

Example: Generate 100 uniform random point on a circle.

```
n<-100
d<-2
X<-matrix(rnorm(n*d),n)
X<-X/sqrt(rowSums(X*X))
par(pty="s")
plot(X,xlab="",ylab="")
par(pty="m")
```



Simulating Random Matrices

Simulating a random matrix

A random matrix X is a matrix whose entries are random variables. There are several types of random matrices

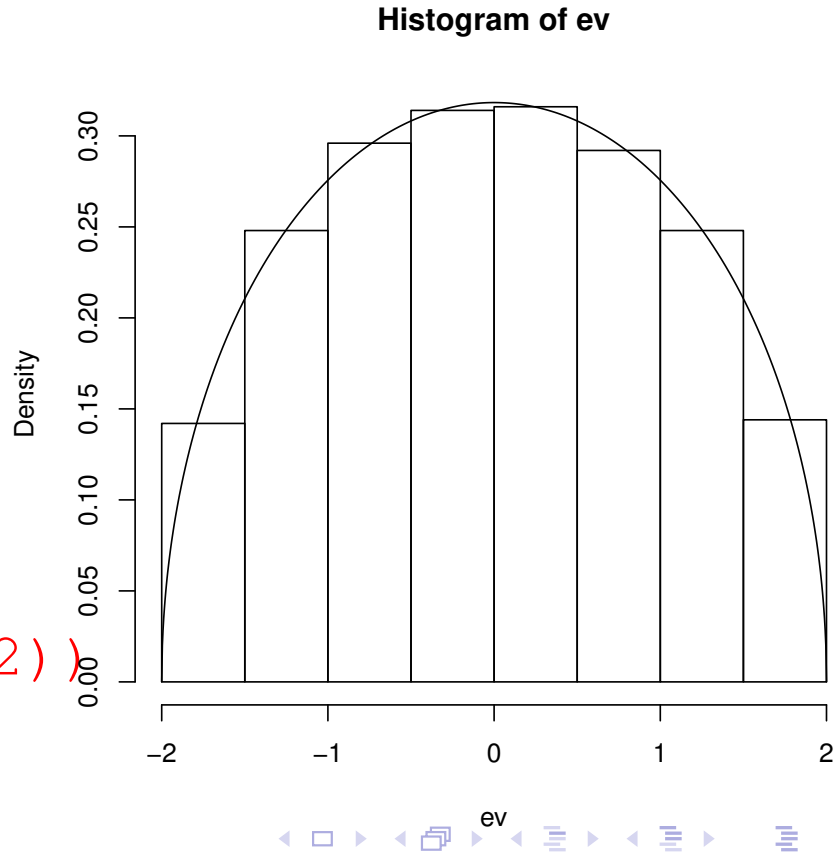
- Wigner matrix $W = \frac{1}{\sqrt{n}} X$ where X is symmetric with standard Gaussian entries. (We can drop Gaussian and get the same results.)

Simulating a random matrix

Example: Find the empirical distribution of the eigenvalues of 1000×1000 Wigner matrix. The pdf is the semi-circle law

$$f_Y(y) = \frac{1}{2\pi}\sqrt{4-y^2} \text{ for } y \in (-2,2)$$

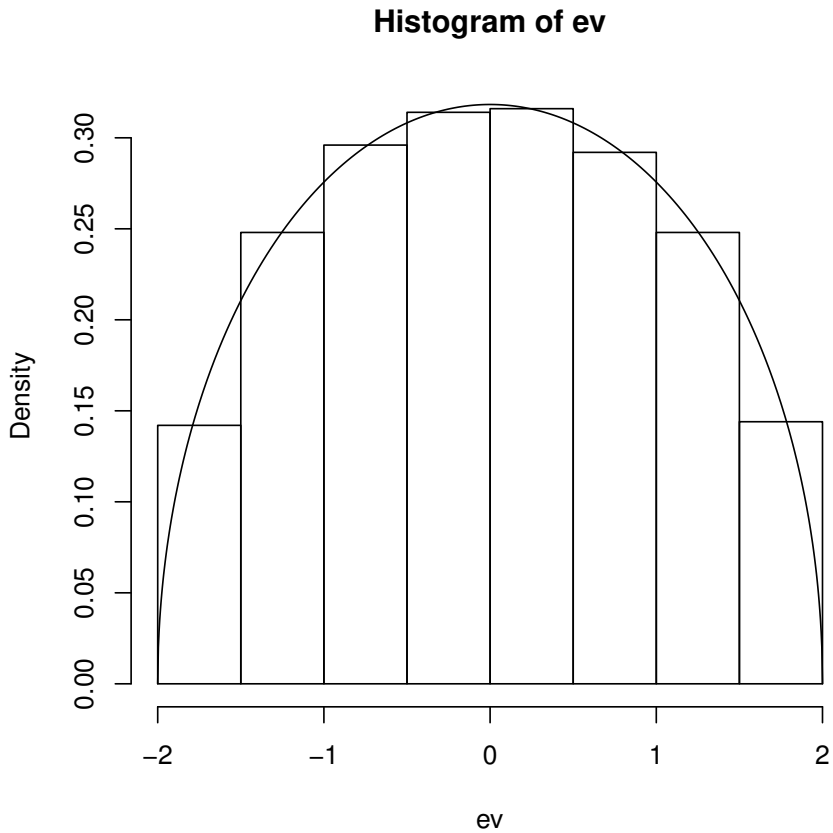
```
n<-1000
X<-matrix(rnorm(n*n),n)
X<-(X+t(X))-diag(diag(X))
X<-X/sqrt(2*n)
E<-eigen(X)
ev<-E$values
hist(ev,prob=T)
y<-seq(-2,2,length=1000)
lines(y,(1/(2*pi))*sqrt(4-y^2))
```



Simulating a random matrix

Example: Find the empirical distribution of the eigenvalues of 1000×1000 Wigner matrix. The pdf is the semi-circle law

$f_Y(y) = \frac{1}{2\pi}\sqrt{4-y^2}$ for $y \in (-2, 2)$
The even moments are the Catalan numbers $C_{2k} = \frac{1}{k+1}C_k^{2k}$ and the odd moments are zero.



Simulating a random matrix

- Wishart (or Sample Covariance) matrix with $df = n$: $S = \frac{1}{n} XX'$ with X is $p \times n$ matrix with standard Gaussian entries.

  Let us say one collected p samples of size n and centralized them so the averages are zero then

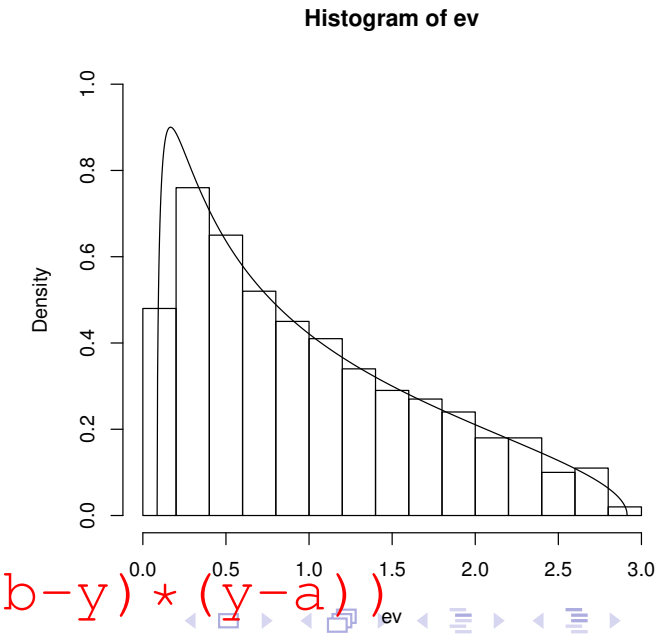
$$S_{i,j} = \frac{1}{n} \sum_{r=1}^{n} X_{i,r} X_{j,r}$$

  is the sample covariance between the two samples $(X_{i,1}, \ldots, X_{i,n})$ and $(X_{j,1}, \ldots, X_{j,n})$ and $S_{i,i}$ is the variance of $(X_{i,1}, \ldots, X_{i,n})$.

Simulating a random matrix

Example: Find the empirical distribution of the eigenvalues of a Wishart matrix with $n = 1000$ and $p = round(.5 * n * (1 - e^{-n}))$. Notice that $p/n \to \nu = .5$ as $n \to \infty$. The pdf is the Marchenko-Patur law $f_Y(y) = \frac{1}{2\pi y \nu}\sqrt{(b-y)(y-a)}$ for $y \in (a, b)$ where $a = (1 - \sqrt{y})^2$ and $b = (1 + \sqrt{y})^2$

```
n<-1000
p<-round(.5*n*(1-exp(-n)))
X<-matrix(rnorm(n*p),p,n)
X<-X%*%t(X)/n
E<-eigen(X)
ev<-E$values
hist(ev,prob=T,ylim=c(0,1))
y<-seq(0,max(ev),length=1000)
nu<-p/n
a<-(1-sqrt(nu))^2
b<-(1+sqrt(nu))^2
lines(y,(1/(2*pi*y*nu))*sqrt((b-y)*(y-a)))
```



Simulating a random matrix

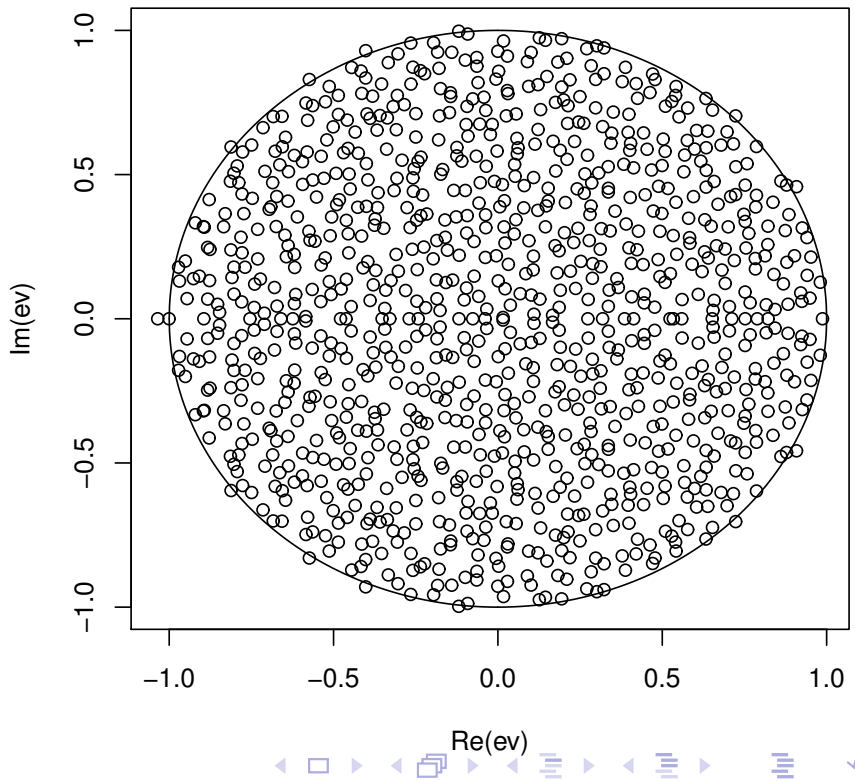
- A square X is $n \times n$ matrix with standard Gaussian entries.
- The distribution of the eigenvalues are uniformally distributed on the unit disc.

Simulating a random matrix

Example: Find the empirical distribution of the eigenvalues of a square matrix $X/\sqrt{n}$ with $n = 1000$. The limiting distribution is the Circular law

$$f_Y(x, y) = \frac{1}{\pi} \text{ with } x^2 + y^2 \leq 1$$

```
n<-1000
X<-matrix(rnorm(n*n),n)
X<-X/sqrt(n)
E<-eigen(X)
ev<-E$values
plot(Re(ev),Im(ev))
x<-seq(-1,1,length=2000)
lines(x,sqrt(1-x^2))
lines(x,-sqrt(1-x^2))
```



End of Set 3