Basic R commands

```
x <- 2
y <- 3
z <- x+y
z
```

```
#-----------------------------------------------------
# Binomial Distr
choose(10,3)
# The command below gives P(X = 4)
# where x ~ Bin( n = 12, p = 0.2)
dbinom(4, size = 12, prob = 0.2)
[1] 0.1328756
> pbinom(4, size = 12, prob = 0.2) # P(X <= 4)
[1] 0.9274445


#Poission dist
dpois(2, 3.5)
[1] 0.184959

ppois(2, 3.5)
[1] 0.3208472


#Geometric dist

dgeom(3, 0.4)
[1] 0.0864

> pgeom(3, 0.4)
[1] 0.8704
```

```
library(prob)
 tosscoin(1)
  toss1
1     H
2     T

tosscoin(2)
  toss1 toss2
1     H     H
2     T     H
3     H     T
4     T     T
```

The prob package accomplishes sampling from urns with the urnsamples function, which has arguments x, size, replace, and ordered. The argument x represents the urn from which sampling is to be done. The size argument tells how large the sample will be. The ordered and replace arguments are logical and specify how sampling will be performed.

```
> urnsamples(1:3, size = 2, replace = TRUE, ordered = TRUE)
  X1 X2
1  1  1
2  2  1
3  3  1
4  1  2
5  2  2
6  3  2
7  1  3
8  2  3
9  3  3

> urnsamples(1:3, size = 2, replace = F, ordered = TRUE)
  X1 X2
1  1  2
2  2  1
3  1  3
4  3  1
5  2  3
6  3  2

> urnsamples(1:3, size = 2, replace = F, ordered = F)
  X1 X2
1  1  2
2  1  3
3  2  3
```

```
> S <- c(2, 5, 7)
> urnsamples(S, size = 2, replace = TRUE, ordered = TRUE)
  X1 X2
1  2  2
2  5  2
3  7  2
4  2  5
5  5  5
6  7  5
7  2  7
8  5  7
9  7  7

> S <- c("A", "B", "C")
> urnsamples(S, size = 2, replace = TRUE, ordered = TRUE)
  X1 X2
1  A  A
2  B  A
3  C  A
4  A  B
5  B  B
6  C  B
7  A  C
8  B  C
9  C  C
>



#-----------------------------------------------------------
 rolldie(1)
  X1
1  1
2  2
3  3
4  4
5  5
6  6
```

```
nrow(rolldie(3))
[1] 216


subset(rolldie(3), X1+X2+X3 > 16) # X's must be uppercase
    X1 X2 X3
180  6  6  5
210  6  5  6
215  5  6  6
216  6  6  6
nA <- nrow(subset(rolldie(3), X1+X2+X3 > 16))

nA
[1] 4

nS <- nrow(rolldie(3))
 nS
[1] 216

p <- nA/nS
p
[1] 0.01851852

#----------------------------------------------------------
# Or
A <- subset(rolldie(3, makespace = T ), X1+X2+X3 > 16)
Prob(A)
[1] 0.01851852
```

Question: Roll two dice. What is the conditional probability that the two dice show the same number given that the sum of the two numbers is greater than or equal to 8?

```
S <- rolldie(2,makespace = T)

A <- subset(S, X1+X2 >=8)
A
   X1 X2      probs
12  6  2 0.02777778
17  5  3 0.02777778
18  6  3 0.02777778
22  4  4 0.02777778
23  5  4 0.02777778
24  6  4 0.02777778
27  3  5 0.02777778
28  4  5 0.02777778
29  5  5 0.02777778
30  6  5 0.02777778
32  2  6 0.02777778
33  3  6 0.02777778
34  4  6 0.02777778
35  5  6 0.02777778
36  6  6 0.02777778

B <- subset(S, X1==X2)
B
   X1 X2      probs
1   1  1 0.02777778
8   2  2 0.02777778
15  3  3 0.02777778
22  4  4 0.02777778
29  5  5 0.02777778
36  6  6 0.02777778
```

```
Prob(A)
[1] 0.4166667

Prob(B)
[1] 0.1666667

Prob(intersect(A,B))
[1] 0.08333333

Prob(A, given = B)
[1] 0.5

Prob(B, given = A)
[1] 0.2
```

```
S <- cards(makespace = T)
A <- subset(S, suit== "Heart")
B <- subset(S, rank %in% 7:9)

Prob(A)
[1] 0.25
Prob(B)
[1] 0.2307692
```

```
S
   rank    suit      probs
1     2    Club 0.01923077
2     3    Club 0.01923077
3     4    Club 0.01923077
4     5    Club 0.01923077
5     6    Club 0.01923077
6     7    Club 0.01923077
7     8    Club 0.01923077
8     9    Club 0.01923077
9    10    Club 0.01923077
10    J    Club 0.01923077
11    Q    Club 0.01923077
12    K    Club 0.01923077
13    A    Club 0.01923077
14    2 Diamond 0.01923077
15    3 Diamond 0.01923077
16    4 Diamond 0.01923077
17    5 Diamond 0.01923077
18    6 Diamond 0.01923077
19    7 Diamond 0.01923077
20    8 Diamond 0.01923077
21    9 Diamond 0.01923077
22   10 Diamond 0.01923077
23    J Diamond 0.01923077
24    Q Diamond 0.01923077
25    K Diamond 0.01923077
26    A Diamond 0.01923077
27    2   Heart 0.01923077
28    3   Heart 0.01923077
29    4   Heart 0.01923077
30    5   Heart 0.01923077
31    6   Heart 0.01923077
32    7   Heart 0.01923077
```

```
33    8   Heart 0.01923077
34    9   Heart 0.01923077
35   10   Heart 0.01923077
36    J   Heart 0.01923077
37    Q   Heart 0.01923077
38    K   Heart 0.01923077
39    A   Heart 0.01923077
40    2   Spade 0.01923077
41    3   Spade 0.01923077
42    4   Spade 0.01923077
43    5   Spade 0.01923077
44    6   Spade 0.01923077
45    7   Spade 0.01923077
46    8   Spade 0.01923077
47    9   Spade 0.01923077
48   10   Spade 0.01923077
49    J   Spade 0.01923077
50    Q   Spade 0.01923077
51    K   Spade 0.01923077
52    A   Spade 0.01923077

union(A,B)
   rank    suit      probs
6     7    Club 0.01923077
7     8    Club 0.01923077
8     9    Club 0.01923077
19    7 Diamond 0.01923077
20    8 Diamond 0.01923077
21    9 Diamond 0.01923077
27    2   Heart 0.01923077
28    3   Heart 0.01923077
29    4   Heart 0.01923077
30    5   Heart 0.01923077
31    6   Heart 0.01923077
32    7   Heart 0.01923077
33    8   Heart 0.01923077
34    9   Heart 0.01923077
35   10   Heart 0.01923077
36    J   Heart 0.01923077
37    Q   Heart 0.01923077
38    K   Heart 0.01923077
39    A   Heart 0.01923077
45    7   Spade 0.01923077
46    8   Spade 0.01923077
47    9   Spade 0.01923077
```

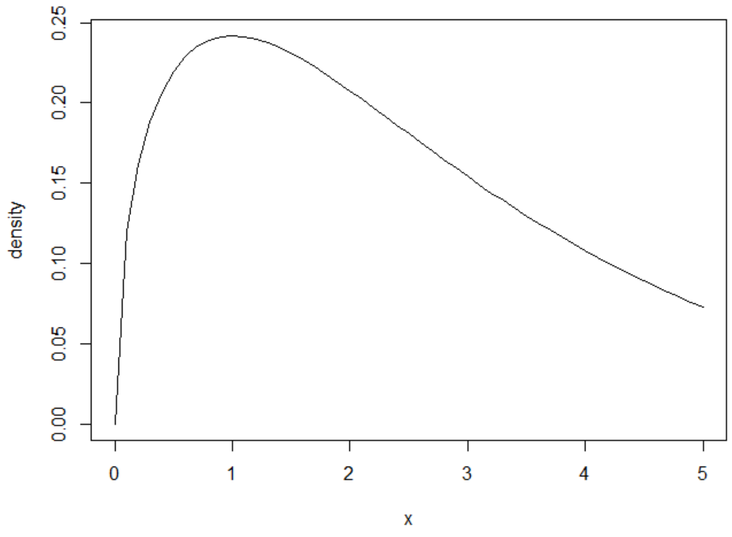
```
Prob(union(A,B))
[1] 0.4230769
>

intersect(A, B)
   rank  suit      probs
32    7 Heart 0.01923077
33    8 Heart 0.01923077
34    9 Heart 0.01923077

Prob(intersect(A,B))
[1] 0.05769231

AminusB <- setdiff(A,B)
AminusB
   rank  suit      probs
27    2 Heart 0.01923077
28    3 Heart 0.01923077
29    4 Heart 0.01923077
30    5 Heart 0.01923077
31    6 Heart 0.01923077
35   10 Heart 0.01923077
36    J Heart 0.01923077
37    Q Heart 0.01923077
38    K Heart 0.01923077
39    A Heart 0.01923077

Prob(AminusB)
[1] 0.1923077
```

Continuous distributions

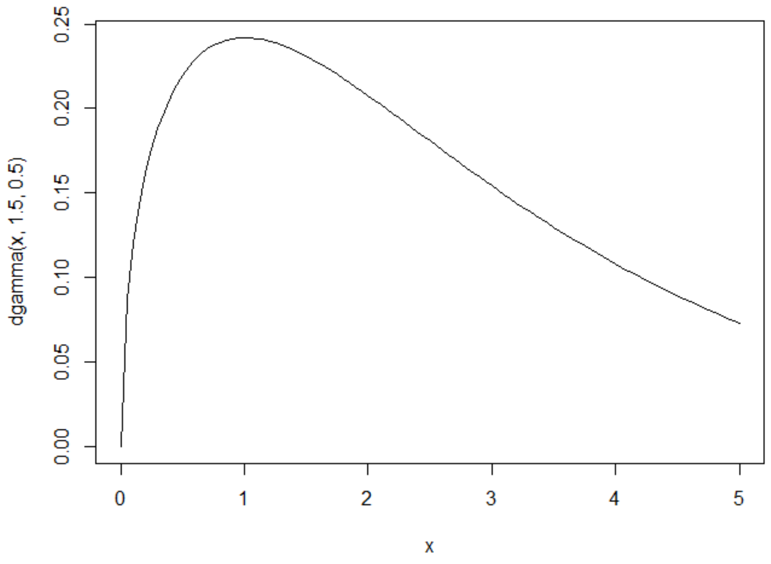
```
pnorm(1.96, 0, 1)
[1] 0.9750021

qnorm(0.975)
[1] 1.959964

# Gamma distr
# q=12, alpha = 1.5 (shape) , lambda = 0.5 (rate parameter)
pgamma(1.2, 1.5, 0.5)
#Note 1/rate is called the scale parameter
[1] 0.2469957

# Or you can integrate the densty

alpha <- 1.5
lambda <- 0.5
f <- function(x) (lambda^alpha)*(x^(alpha-
1))*(exp(-lambda*x))*(1/gamma(alpha))

integrate(f, 0, 1.2)
0.2469958 with absolute error < 0.00012


qgamma(0.247, 1.5, 0.5)
[1] 1.200018
```

Plotting the density curves

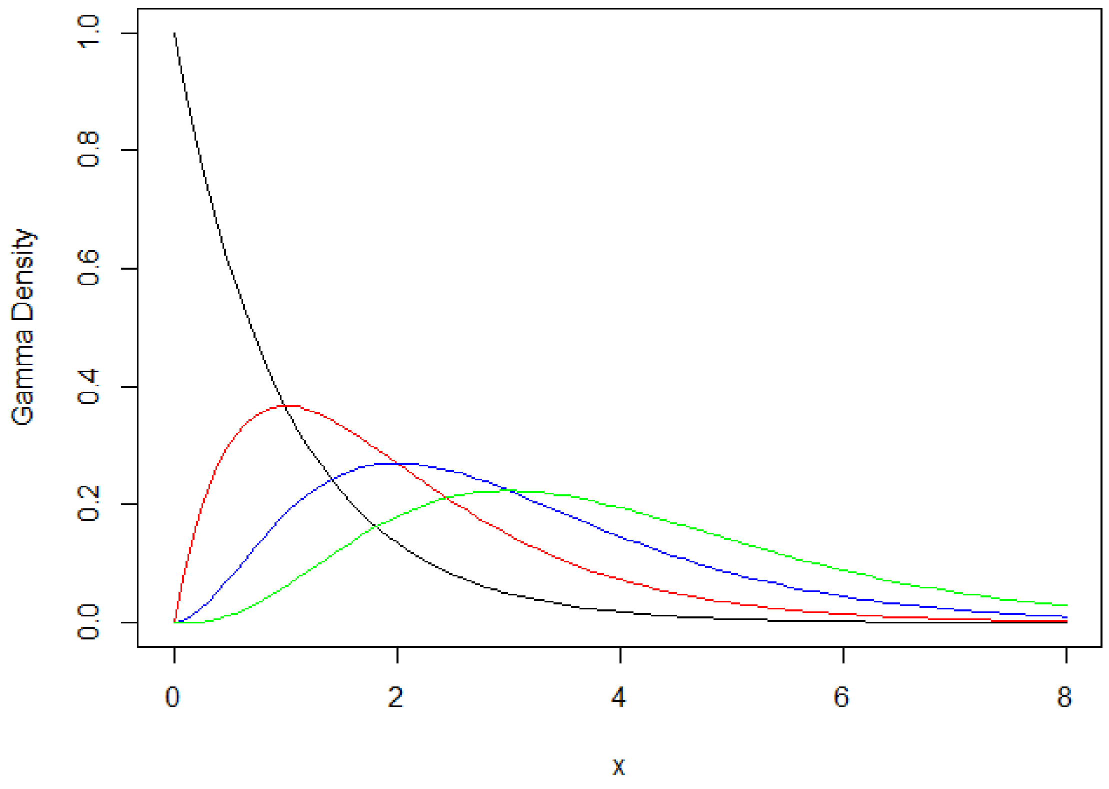
```
x <- seq(0,5,0.1)
density <- dgamma(x, 1.5, 0.5)
plot(x, density, type = "l")
```



```
#----------------------------------------------
# or
curve(dgamma(x, 1.5, 0.5),0,5.0)
```



```
#-------------------------------------------------
curve(dgamma(x,1,1),0,8, ylab = "Gamma Density")
curve(dgamma(x,2,1),0,8,add=TRUE,col="red")
curve(dgamma(x,3,1),0,8,add=TRUE,col="blue")
curve(dgamma(x,4,1),0,8,add=TRUE,col="green")
```



```
library(distr)
f <- function(x) 3*x^2
X <- AbscontDistribution(d = f, low1 = 0, up1 = 1)
p(X)(0.7)
[1] 0.3430001

p(X)(0.2)
[1] 0.008000034

p(X)(0.7) - p(X)(0.2)
[1] 0.3350001

# Can also integrate

integrate(f, 0.2, 0.7)
0.335 with absolute error < 3.7e-15
```

# R gives the same probability even if you drop the normalizing constant

```
f <- function(x) x^2
X <- AbscontDistribution(d = f, low1 = 0, up1 = 1)
p(X)(0.7)
[1] 0.3430001
```