Generating Random Numbers

- From any supported (univariate) distribution:

```
> rnorm(3)
[1]  2.6527311 -1.7975851 -0.4912465
> rnorm(3)
[1] -1.7124846 -0.5033372 -0.2457211
> rnorm(3,mean=5000,sd=50)
[1] 4934.942 5017.293 5000.024
> rgamma(1,shape=2,scale=6)
[1] 14.27924
> rpois(5,lambda=60)
[1] 58 57 65 66 61
>
```

- Uniform random numbers:
  - Uniform on real interval $[a, b]$,

```
> runif(5, min=-4, max=8)
[1]  4.824822  2.196409 -2.250274 -3.891445  5.604381
```

  - Uniform on $\{1, 2, 3, \ldots, n\}$,

```
> ceiling(runif(12, max=10))
 [1]  3 10  6  8  2  3  8  5 10 10  1  8
```

  - Uniform on $\{0, 1, 2, \ldots, n-1\}$,

```
> floor(runif(15, max=10))
 [1] 0 1 8 4 9 1 9 6 6 6 9 0 0 8 7
```

  - Uniform on $\{m, m+1, \ldots, n\}$,

```
> m <- -2
> n <- 3
> floor(runif(15, min=m, max=n+1))
 [1]  2  0  2 -1  1  2  3 -2  0  1  2 -2  2  0  3
>
```


Using Parameter Vectors

Some examples:

- Every fourth observation has a different parameter (using parameter "recycling"):

```
> rpois(10, lambda=c(1,1,1,100))
 [1]   0   0   1 101   2   2   0  96   1   0
>
```

- Mixture distribution: each observation independently drawn from $N(5, 1)$ with prob 0.2 and from $N(4, 3)$ with prob 0.8.

```
> (coin <- rbinom(20,size=1,prob=.2))
 [1] 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0
> ifelse(coin,5,4)
 [1] 4 5 4 4 4 4 5 4 4 4 5 4 4 4 4 4 5 4 4 4
> rnorm(20, mean=ifelse(coin,5,4), sd=ifelse(coin,1,3))
 [1]  6.7428421  3.3436197  2.8704465  3.6187790
[ . . . ]
[19]  1.6189685  6.4631923
>
```

- Generating a linear regression example:

```
> age <- floor(runif(10, min=5, max=13)) # ages 5-12
> gender <- factor(c("F","M")[rbinom(10,size=1,prob=.5)+1])
> height <- round(rnorm(10,
        mean = 60+6.5*age+5*(gender=="M"), sd=5))
> summary(lm(height ~ gender + age))
[ . . . ]
(Intercept)  68.6676     6.7869  10.118 1.98e-05 ***
genderM       1.6892     3.1540   0.536    0.609
age           5.7162     0.6883   8.305 7.17e-05 ***
[ . . . ]
Residual standard error: 4.586 on 7 degrees of freedom
[ . . . ]
```


Random Sampling

- Random permutation (i.e., perfect shuffle)

```
> sample(10)   # sample(1:10) has same effect
 [1]  6  9  2  7  5 10  4  3  1  8
> sample(c("Stan","Ruojia","Sam","Ahmed","Felicia"))
[1] "Stan"    "Sam"     "Ruojia"  "Felicia" "Ahmed"
> iris[sample(nrow(iris)),]
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
43           4.4         3.2          1.3         0.2     setosa
115          5.8         2.8          5.1         2.4  virginica
[ . . . 147 more randomly ordered records . . . ]
92           6.1         3.0          4.6         1.4 versicolor
```

- Random subsample (without replacement)

```
> dim(iris)
[1] 150   5
> iris.sample <- iris[sample(nrow(iris),size=30),]
> dim(iris.sample)
[1] 30  5
```

- Random subsample (with replacement)

```
> sample(10,size=5)               # without replacement
[1] 8 6 3 2 7
> sample(10,size=5,replace=T)     # with replacement
[1] 3 4 9 6 6
> sample(10, size=20)
Error in sample(x, size, replace, prob) : can't take a
sample larger than the population when replace = FALSE
> sample(10, size=20, replace=T)
 [1]  8  6  9  5  4  9  1  6  3  6  7  5  4  9  9  8  8  2  8 10
>
```


The Random Number Seed

```
> rnorm(5)
[1] -0.53931674  0.51708133  0.02273572  0.68687194 -0.16931581
> rnorm(5)
[1] -0.4644731 -1.4228934  0.1793277 -0.1587798  0.2894271
> set.seed(621)
> rnorm(5)
[1]  0.6832647 -0.8720841  1.4884790 -0.3136084 -1.0155666
> rnorm(5)
[1] -0.4917928 -2.4038981  0.2535542  0.4300435 -0.3850864
> set.seed(621)
> rnorm(5)
[1]  0.6832647 -0.8720841  1.4884790 -0.3136084 -1.0155666
> rnorm(5)
[1] -0.4917928 -2.4038981  0.2535542  0.4300435 -0.3850864
> set.seed(621)
> rnorm(10)
 [1]  0.6832647 -0.8720841  1.4884790 -0.3136084 -1.0155666
 [6] -0.4917928 -2.4038981  0.2535542  0.4300435 -0.3850864
>

> RNGkind()
[1] "Mersenne-Twister" "Inversion"
> .Random.seed
  [1]         403          20  -934327767  1112312283
[ . . . ]
[625] -1627563568  1404439705
>
```