Random numbers and random sampling

This worksheet is an interactive, guided module for learning the basics of generating a variety of different types of random numbers, and how to do random sampling in R.

It assumes you know how to read datafiles and produce a dataframe containing your data.

Let us begin by learning how to generate random numbers.
Two of the most useful types of random numbers are those that are: (1) uniformly distributed in some specified interval; and (2) normally distributed, with specified mean and SD.

Example: Shown below are several variants of R functions to generate both types of random numbers. The basic function for uniformly distributed numbers is `runif()`. Likewise, the function
for normally distributed numbers is `rnorm()`. </FONT>

Note that all the information following any "#" sign is just to explain what is going on. R ignores anything that follows a "#" sign.

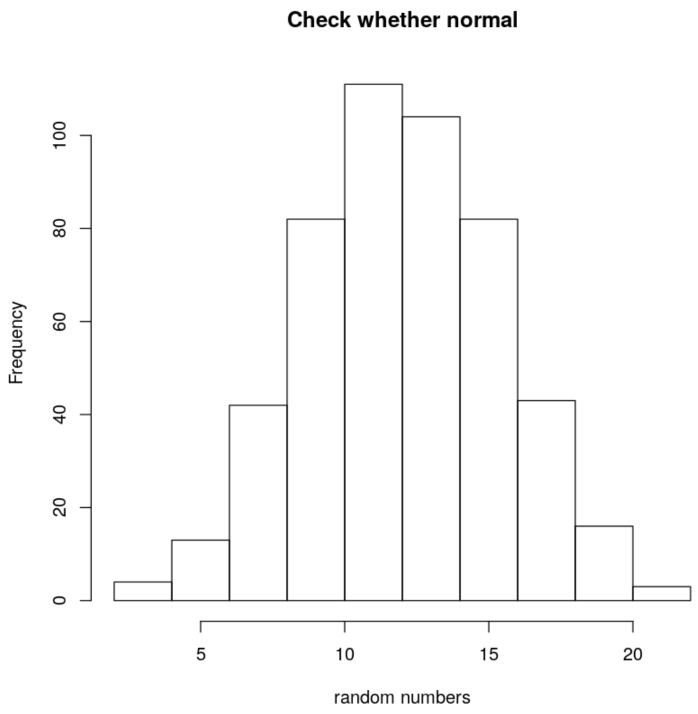
In [1]:
```
n = 3
# To generate n uniform random numbers between min and max:
runif (n, min=0, max=5)

# Note that the same command can be run by the following (less clear) short-h
and:
runif (3, 0, 5)

# If the min/max are also left out, it defaults to the range of 0 to 1:
runif (3)
```

```
3.4688546997495 · 2.57830733666196 · 0.74065915425308

1.59611364011653 · 4.34419165598229 · 2.68441868014634

0.787913508480415 · 0.621695661917329 · 0.153730960097164
```

In [2]:
```
n = 3
# To generate n normally distributed random numbers, use "rnorm":
rnorm (n, mean=12, sd=3.5)

# Again, can run the equivalent (less clear) short-hand:
rnorm (n, 12, 3.5)

# If mean/sd are left out, defaults to standard normal dist. with mean=0, sd=
1:
rnorm (n)
```

```
14.2106775244689 · 9.81123465000459 · 4.56381673206475

14.3605927080139 · 10.6440352691295 · 12.9928451561781

1.54390551109893 · 1.01453096368414 · -0.608159968281451
```

In [3]:
```
# To verify whether it is actually doing what we think, let's
# plot a histogram of 500 normally distributed random numbers:
x = rnorm (500, 12, 3.5)
hist(x, xlab="random numbers", main="Check whether normal")
```



How about discrete random numbers

Notice that all the above examples produce numbers that are continuously distributed across their range. Such numbers usually contain decimals, and rarely turn out to be nice, round numbers.

What if we wanted random integers, say, 12 of them, lying between -6 and 23? The function for doing that is `sample`, as shown in the following examples

In [4]:
```
# One way to  randomly pick integer numbers from a specified
# range is using the "sample" function.
#
# Example: Pick 12 numbers at random that lie between −6 and 23.
sample ( -6:23,  12, replace=TRUE)
```

```
12 · -2 · 0 · 5 · 12 · 4 · 4 · -5 · -2 · 6 · 6 · -5
```

In [5]:
```
# Note that "replace=TRUE" allows picking the same number
# more than once.  If you want all the numbers to
# be different, just leave out that option, like this
sample ( -6:23,  12)
```

```
1 · 13 · -4 · 17 · 16 · 21 · -3 · -1 · 5 · 22 · 7 · 10
```

The `sample` function can also be used to randomly pick from categorical variables. Some examples follow.

In [6]:
```
# Toss a coin 3 times and record the sequence of outcomes
sample( c("H", "T"), 3, replace=TRUE )

# Here is one way to do 10 trials of tossing a coin 3 times.
# There is, likely, a better way to do this. But this is what
# I know at the moment!
for ( i in 1:10 ){
#    print( sample( c("H", "T"), 3, replace=TRUE ) )
    show( sample( c("H", "T"), 3, replace=TRUE ) )
}
```

```
'T' · 'H' · 'T'

[1] "T" "H" "T"
[1] "H" "T" "H"
[1] "T" "H" "H"
[1] "H" "T" "T"
[1] "T" "T" "H"
[1] "T" "T" "T"
[1] "H" "H" "T"
[1] "H" "H" "H"
[1] "T" "T" "H"
[1] "T" "H" "T"
```

How to pick random samples from datafiles

Another very important use of the `sample` function
is to pick random samples from data sets. R provides relatively straightforward ways to pick simple random samples and stratified random samples from dataframes.

Example: The file "test_csv_file.csv" contains data on the employment status, work hours, age, etc., of a group of university students. The following examples show how to get an SRS and stratified random sample from this dataset.

In [9]:
```r
# Read data file and create a dataframe called "empdata"
empdata = read.csv(file="./test_csv_file.csv", header=TRUE, sep=",")
#empdata = read.csv(file="https://cs.earlham.edu/~pardhan/sage_and_r/test_csv
_file.csv", header=TRUE, sep=",")

# If we want to see the original data, uncomment next line
# empdata

# Pick a simple random sample of size 7 from this data
library(dplyr)
empdata %>%
    sample_n(7)
```

```
Attaching package: ‘dplyr’


The following objects are masked from ‘package:stats’:

    filter, lag


The following objects are masked from ‘package:base’:

    intersect, setdiff, setequal, union
```

A data.frame: 7 × 5

Response_id	Age	Gender	Employment.Status	Work.Hours
<int>	<int>	<fct>	<fct>	<dbl>
165345	21	Female	Unemployed	0.0
166389	38	Female	Part Time	20.0
165469	33	Female	Full Time	37.5
166216	33	Female	Full Time	37.5
165016	33	Female	Unemployed	0.0
164419	19	Female	Part Time	35.0
166397	33	Female	Full Time	37.5

```
In [10]: # Now let's try a stratified random sample based on employment status.
         # Let us pick 3 people from each stratum
         empdata %>%
             group_by(Employment.Status) %>%
             sample_n(3)
```

A grouped_df: 9 × 5

Response_id	Age	Gender	Employment.Status	Work.Hours
<int>	<int>	<fct>	<fct>	<dbl>
166397	33	Female	Full Time	37.5
166105	41	Male	Full Time	50.0
166415	37	Male	Full Time	40.0
164946	20	Female	Part Time	25.0
166391	18	Female	Part Time	25.0
166209	19	Female	Part Time	30.0
165638	32	Female	Unemployed	0.0
164685	19	Female	Unemployed	0.0
165016	33	Female	Unemployed	0.0

Exercise 1:

1. Generate 9 uniformly distributed random numbers in the range [2, 5].
2. Show that the "runif" function does, in fact, produce a uniform distribution of random numbers by plotting a histogram of 500 numbers in the range [2, 5].
3. Generate 50 normally distributed random numbers with mean=4.56 and SD=2. Plot a histogram showing your results.
4. Toss a fair coin 50 times (using R) and find the proportion of heads.

Exercise 2:

The file `https://cs.earlham.edu/~pardhan/sage_and_r/grades.csv` contains data on midterm scores and class year for a group of college students. Read the file and create a dataframe.

1. Pick a simple random sample of size 12 from this dataset.
2. Next, pick a stratified random sample containing 3 students from each class year.

For each type of sample, be sure to display the full data in the file for each of the selected individuals.

```
In [ ]:
```