STA 250: Statistics
Lab 1

This lab work is intended to be an introduction to the software **R.** What follows is a description of the basic functionalities of `R` , along with a series of tasks that you'd have to perform.

What is `R` ?

- An interpreter-based programming, graphics and statistics package.
- Free, stable, can be extended.
- Can easily perform standard statistical and numerical analysis.
- Can be programmed to handle non-standard cases.
- For complex tasks, it is often used as a first step to interface with `C` or `FORTRAN`.
- Almost all new statistical methodology is published with a ready to use package built with `R` .

Launching `R`

- On `UNIX` type workstations, simply type `R` at the prompt.
- On windows open through `Start -> All Programs -> R -> R 2.4.#`
- We will talk about the `UNIX` version, the Windows version is almost identical.
- Once launched `R` prints an opening message (see next page) and gives a command prompt `>`.

```
R version 2.4.1 (2006-12-18)
Copyright (C) 2006 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Entering Commands

- You can directly input commands at the prompt:

```
> 2 + 3
[1] 5
> options(digits = 3)
> rnorm(50)
 [1] -0.0156 -0.3881  1.0899  0.9864  0.1681  0.4718  2.4950 -0.2498  2.0989
[10] -0.5637  0.8308  1.2964  0.7397  0.4435  0.0214  0.4449  0.6374 -1.2289
[19]  0.2015 -0.5251  0.2335  1.0448 -0.0116  0.9199  1.0381 -1.1217  0.7467
[28] -0.2151 -1.0740  0.3098 -0.8013  1.3933 -1.0898 -0.4594 -0.1406 -0.4284
[37] -1.4735  1.7980  1.1085  0.8844  0.1370  0.4474  0.7443 -0.1297  0.4804
[46]  0.1587  1.0659  1.2216  0.0639 -0.0614
```

- If you hit enter befor completely entering a command, you will get a `+` prompt. You must complete the command or type `^C` (`Esc` in MSW) to continue.
- All arithmatic operations are represented via standard symbols (`+ - * /`) and have the usual order of precedence.
- All common functions are represented by their usual names.

Examples

```
> sin(pi/2)
[1] 1
> sqrt(10.34)
[1] 3.215587
> sqrt(1.44e2)
[1] 12
> log(1 + gamma(1))
[1] 0.6931472
> exp(-1/2)
[1] 0.6065307
> 1 + 2 * (log(cos(0.2)) + pi^2)
[1] 20.69894
```

As you can see, nested operations are performed with proper use of parantheses.

> TASK 1. The function call `dbinom(x, n, p)` evaluates the $\mathsf{Binomial}(n, p)$ pdf at the value x. Evaluate $\mathsf{Binomial}(309, 0.62)$ at $x = 193$.

`R` objects

- `R` is an objected oriented environment – everything in `R` is an object, named or unnamed.
- Objects created in the workspace can be saved before quitting `R` by choosing appropriate options after you type `q()`.
- Mainly two types of objects
  - Language objects: functions, expressions, formulas.

  - Data objects: all vectors of different size and type (mode).

- Main modes are: numeric, character (string), logical (TRUE/FALSE) and list (a collection of objects of various modes..).

Creating objects

- Vectors of length 1 can be created using the following "assignment" operator

```
> a <- 2.3
> b <- "StatComp"
> c <- TRUE
```

- General vectors are created using the concatenation function `c()` or the sequence operator `seq` or the repeat operator `rep`.

```
> x <- c(1, 2, -1)
> y <- seq(0, 2, 0.2)
> print(y)
 [1] 0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0
> z <- rep(1,times = 5)
> z
[1] 1 1 1 1 1
```

- An integer sequence with unit increment can be created by using the shortcut `:`.

```
> 1:10
 [1]  1  2  3  4  5  6  7  8  9 10
> -1:4
[1] -1  0  1  2  3  4
> -1:4 + 2
[1] 1 2 3 4 5 6
> -1:(4 + 2)
[1] -1  0  1  2  3  4  5  6
```

  Note the precedence of `:` over arithmetic operators.

> TASK 2. Create a vector `p` with elements $0.0, 0.01, 0.02, \cdots, 1.00$. What's the function of the command `p[42]`? Of `p[54:90]`? Of `p[-11]`?

- A matrix is a long vector stored as a rectangle.

```
> print(a <- matrix(1:12,nrow = 4,ncol = 3))
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
> (a <- matrix(1:12,4,3,byrow = TRUE))
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]   10   11   12
```

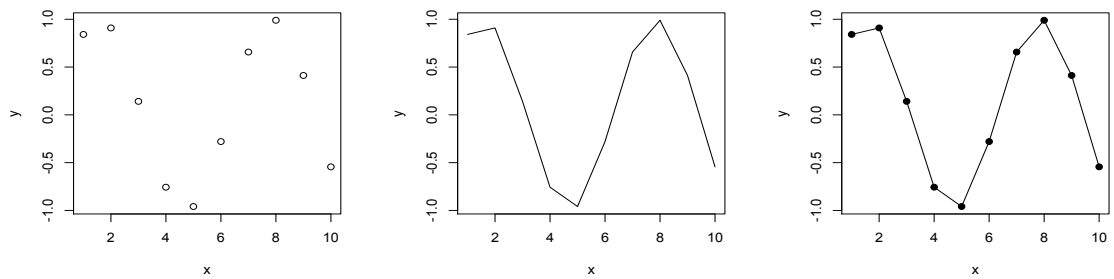
- Higher dimensional arrays can be created as an `array`.

> TASK 3. Store the first 100 elements of the vector `p` you created above as a 10×10 matrix `pp`, with the first 10 elements of `p` appearing on the first column of `pp`.
> What's the function of the command `pp[42]`? Of `pp[, 4]`? Of `pp[8, ]`? Of `pp[8, 4]`? Of `pp[,-1]`? Of `diag(pp)`?

Plots

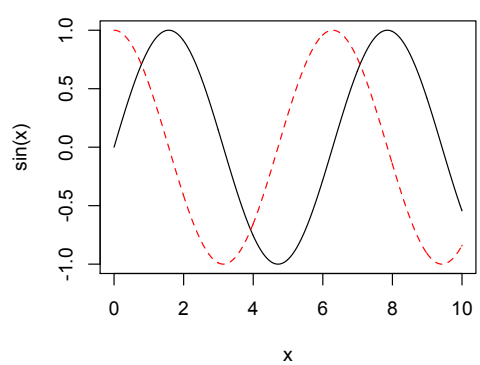
- `plot` is the generic function to plot variables. The default use is `plot(x, y)` which plots a vector `y` against another vector `x` provided they have the same length.

```
> x <- 1:10
> y <- sin(1:10)
> plot(x, y)
> plot(x, y, ty = "l")
> plot(x, y, ty = "o", pch = 19)
```



Lines can be added to an existing plot by using the `lines()` command.

```
> x <- seq(0, 1, 0.01)
> plot(x, sin(x), ty = "l")
> lines(x, cos(x), lty = 2, col = 2)
```



TASK 4. Make a plot of the $\mathsf{Binomial}(46, 0.083)$ pmf. What `x` should you use to satisfactorily represent the pmf? Add to your plot graphs of $\mathsf{Binomial}(46, 0.05)$ and $\mathsf{Binomial}(46, 0.1)$ pmfs. Use three different colors for the three pmfs.
Use the command `abilne(v = 6, lty = 2)` to add a dashed vertical line at $x = 6$. Which of the three pmfs assigns highest probability to $x = 6$?

Functions

A function is a special kind of `R` object that accepts objects as input arguments and produces other objects as outputs. The syntax of a function is as follows:

```
f <- function(args){
  body .....
  ...
  return(ans)
 }
```

For example

```
f <- function(x1, x2){
  f2 <- dunif(x2, 0.05, 0.1)
  f1.cond <- dbinom(x1, 46, x2)
  return(f2 * f1.cond)
 }
```

evaluates the joint pm/df $f(x_1, x_2) = f_2(x_2) f_{1|2}(x_1|x_2)$ for $X_1 =$ number of January borns and $X_2 =$ chance of being born in January with $f_2(x_2) = \mathsf{Uniform}(0.05, 0.1)$ and $f_{1|2}(x_1|x_2) = \mathsf{Binomial}(46, x_2)$.

TASK 5. Evaluate the above function at (6, 0.083), (6, 0.05), (6, 0.03), (6, 0.1) and (6, 0.15).

Getting help on `R` functions

Most stuff in `R` are done by calling in-built functions. These include mathematical functions such as `sin()` or `dbinom()` as well as utility functions `plot()`, `seq()` etc. To get help on a function, say `dnorm` you could use on of the following two commands

```
> help(dnorm)
> ?dnorm
```

You could also search for functions by keywords. For example with

```
> help.search("normal distribution")
```

produces a list of functional related to the normal distribution.

To learn more about `R`

For a more detailed introduction to `R` please refer to http://cran.r-project.org/doc/manuals/R-intro.html.