Getting Started

Statistics 427: R Programming

Module 1

2020

Introduction to R

What is R? R is a software package for scientific calculations, graphics, and statistical analyses. R is free and open source. Versions of R are available for Windows, Mac OS, and Linux/Unix operating systems. (download R here: https://www.r-project.org/) R is in the form of a command-driven programming language with vast statistical, mathematical, and graphical resources built in as pre-programmed functions: R has vector and matrix calculations built in (like Matlab). R features fantastic graphical displays of data. R has calculations (random numbers, percentiles, probabilities, etc.) built in for many different statistical distributions. R has special mathematical functions of all sorts built in (complex numbers, gamma & log-gamma function, numerical integration, optimization, etc.).

Use of R

R has grown an enormous, international group of users (critical mass). Lots of online forums for getting help. Many tutorial websites (often posted by university instructors for courses). Many instructional videos (YouTube especially). Many books. Online courses. Even some MOOCs. Vast numbers of contributed R routines, posted by users as functions, for producing the latest specialized statistical analyses (genetics, finance/investment, multivariate/social sciences, data mining, etc.)

History of R

Bell Laboratories (now Nokia Bell Laboratories) invented/discovered: radio astronomy, transistor, information theory, charge-coupled device, laser, cosmic microwave background Unix, C, C++ The S statistical programming language (John Chambers & others). S-PLUS MathSoft distributes commercial version of S, called S-PLUS (around 1995). S-PLUS spreads rapidly in the statistics world; features dazzling (at the time) graphics such as rotating 3D plots and painting of scatter plots, along with statistical programming for numerically intensive research.

Creation of R

R was created by Ross Ihaka and Robert Gentleman (later 1990s). Uses the syntax of S-PLUS & S (syntax can't be copyrighted). Free, open source, part of the GNU project. Currently developed by the R Development Core Team (John Chambers is a member). The R Journal is started in 2001. Many R users by 2005. "Data analytics" becomes big in the late 2000s; NYT features an article about R in 2009. Early reputation was that R was slow (in comparison to Matlab), but recent versions of R are substantially speeded up. (Microsoft has issued a free distribution of R that is reportedly quite fast) R is an interpreted programming language rather than a compiled language.

R competitors

S-PLUS (expensive). Matlab (expensive). Gauss (expensive). Octave (free, copies Matlab syntax, graphics not as extensive). Python (free interpreted programming language, with features that programmers like more

than R, but beginner level documentation is poor, and the library of user-contributed statistical routines is much smaller). SAS (expensive but has established itself in the 70s, 80s, and 90s as the most extensive and powerful statistical analysis and data management software).

Tutorial

In its most basic sense, `R` is an overgrown calculator. At the prompt (`>`), type `5+7` and hit `Enter`

```
5+7
```

```
[1] 12
```

The `[1]` means "Part 1" of the answer is 12. Many times answers can have many parts, so `R` prints part numbers along with the answers.

Operations I

`R` will always remember the order of operations, even if you do not. PEMDAS... please excuse my dear Aunt Sally. :-)

Basic arithmetic functions work rather intuitively, including use of negative numbers

```
# This is a comment in R, it will print in the console but will not be
# run as a command, which is good because my babbling will cause errors
5+-2
```

```
[1] 3
```

```
5*7
```

```
[1] 35
```

```
5/7
```

```
[1] 0.7142857
```

```
5^7 # exponentiation
```

```
[1] 78125
```

```
5+7*3-12/4-6 #PEMDAS
```

```
[1] 17
```

Operations II

Use parentheses to alter priorities of operations

```
(5^3+7)*3-(12/4-6)^2
```

```
[1] 387
```

Use multiple sets of parentheses, but be careful all left ones have a corresponding right.

```
(5+7)*3-(12/(4-6))
```

```
[1] 42
```

Storage I

R stores everything for you. You need to use the assignment operation to store your calculations. The assignment statement is the naming of `objects` (more on those later). R is case-sensitive, so you cannot use all caps if the name was created in lower-case.

```
fred=5+7; eve=4-2
fred-eve
```

```
[1] 10
```

```
fred
```

```
[1] 12
```

```
eve
```

```
[1] 2
```

Storage II

If you use the name for something different, then it will overwrite the original value

```
eve=9
eve
```

```
[1] 9
```

Assignment Statement

The '=' symbol in R (and many other programming languages) does not mean 'equals'. Rather it means 'calculate what is on the right and store results using the name on the left.'

Previous versions of R used the less than symbol '<' with a dash '-' (which when there is no space between those characters, creates an arrow `<-`) as an assignment statement.

```
eve <- 12 # the spaces are unnecessary but makes it easier to see
eve
```

```
[1] 12
```

Vectors

R works with whole "lists" of numbers (or characters). The use of the `c()` command will combine values into a vector (or list). The general form of `c()` is:

`c(...)`
`...`: objects to be concatenated (combined), separated by commas

There are a couple of other options if you need when creating vectors, most often just the list of items are needed, separated by commas. Do not use commas (or periods) at all for the three-digit separation, only use commas to separate the individual *elements* of the vector and decimals for actual fractional values.

```
x=c(3,-2,4.0,7,5,-1,0)
```

Vector logistics I

You can perform mathematical operations on vectors with other vectors or constants.

```
x=c(3,-2,4,7,5,-1,0)
y=4
x+y
```

```
[1]  7  2  8 11  9  3  4
```

```
x*y
```

```
[1] 12 -8 16 28 20 -4  0
```

```
x^y
```

```
[1]   81   16  256 2401  625    1    0
```

```
z=x+y; z # the semi-colon separated commands for processing
```

```
[1]  7  2  8 11  9  3  4
```

Vector Logistics II

If you make a mistake wile inputting, just type the line again (you can also hit the arrow up key to scroll through your last submitted commands). `R` will keep and store the newest version. Also, if a line is too long, you can continue it on the line by hitting the `Enter` key *at a place where the `R` command is obviously incomplete* (`R` is smart (not s-m-r-t)). `R` will respond with a different prompt that looks like a plus sign `+`. Just continue writing the `R` command at the new prompt and hit `Enter` when finished.

```
> fum=c(-1,1,
+ .5)
> fum
[1] -1.0  1.0  0.5
```

Vector Logistics III

If you are trying to create a vector of sequential values ($i = 1, 2, \ldots, 20$ or $j = A, B, C, \ldots, Z$), an easier way to do so without tying all the values is to use a colon `:`. The starting value goes on the left, the ending value on the right of the colon.

```
i=0:10 # a vector of all integers from 0 to 10
i
```

```
 [1]  0  1  2  3  4  5  6  7  8  9 10
```

```
j=10:0 # backwards!
j
```

```
 [1] 10  9  8  7  6  5  4  3  2  1  0
```

```
k=5:-5 # :-) negative numbers
k
```

```
 [1]  5  4  3  2  1  0 -1 -2 -3 -4 -5
```

Vector Logistics IV

```
l=0:20
# powers of 2 from 2^0 to 2^20
2^l
```

```
 [1]       1       2       4       8      16      32      64     128     256
[10]     512    1024    2048    4096    8192   16384   32768   65536  131072
[19]  262144  524288 1048576
```

Intro to basic graphs

R is great for computing but is also great at visualizations, even the most basic ones.

Interest I

Scenario: you have $1000 to save and decide on a CD that gets 5% annual interest, to be reinvested back into the total every year. From that we get:

$$P(t) = P_0(1+r)^t$$

$P(t)$: final amount after time t
P_0: starting amount
r: rate
t: time (years)

Interest II

Define variables below; time is 10 years.

```
t=0:10
p0=1000
r=.05
pt=p0*(1+r)^t
pt
```

```
 [1] 1000.000 1050.000 1102.500 1157.625 1215.506 1276.282 1340.096 1407.100
 [9] 1477.455 1551.328 1628.895
```

Note it only shows each year's totals.

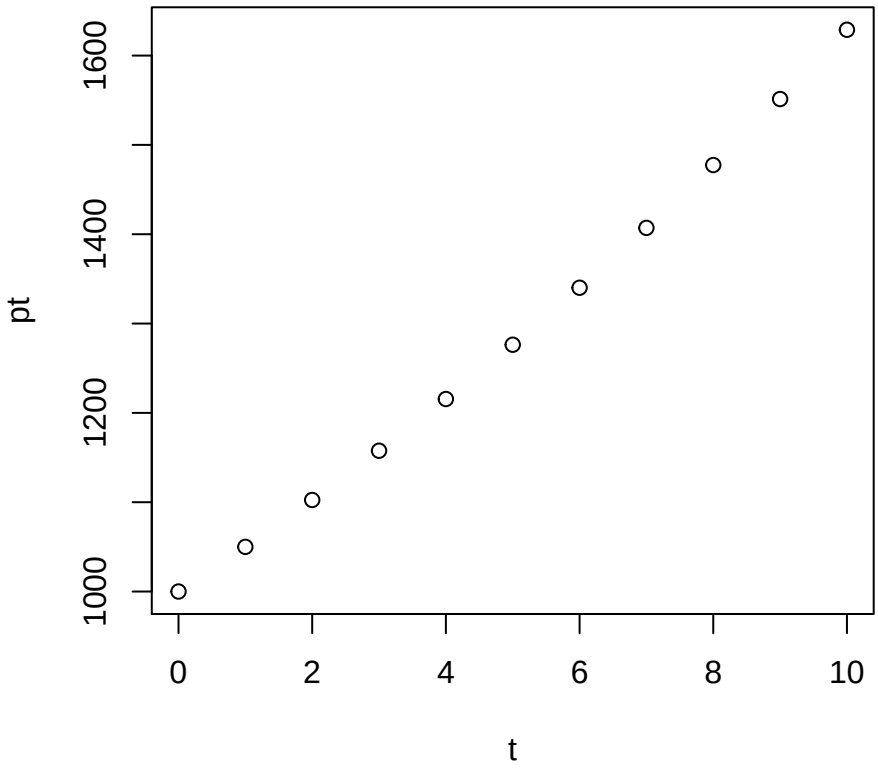
Plotting using `plot()`

General form of `plot(x,y,type='p',...)`:
`x`: the vector of x values
`y`: the vector of y values
`type=' '`: 'p' for points, 'l' for line, 'b' both, ...
`...`: more options (covered later in course)

The first plot

A scatterplot with time on x and values on y axes using points.

```
plot(t,pt,type='p')
```



Graphics Window

With that, a new window should have opened with the graph. In the Mac version of `R`, the graphics window should always display all the graphs created in your `R` session (until you close `R`). With Windows, you may have to tell the graphics window to `Record History` so that all plots created in the session can be accessed without having to make the graph again. I am unsure about the Linux version (sorry).

You can "Save As..." and save graphs as PDFs or copy and paste them into other documents. There are other mediums you can save them as but I will stop here.

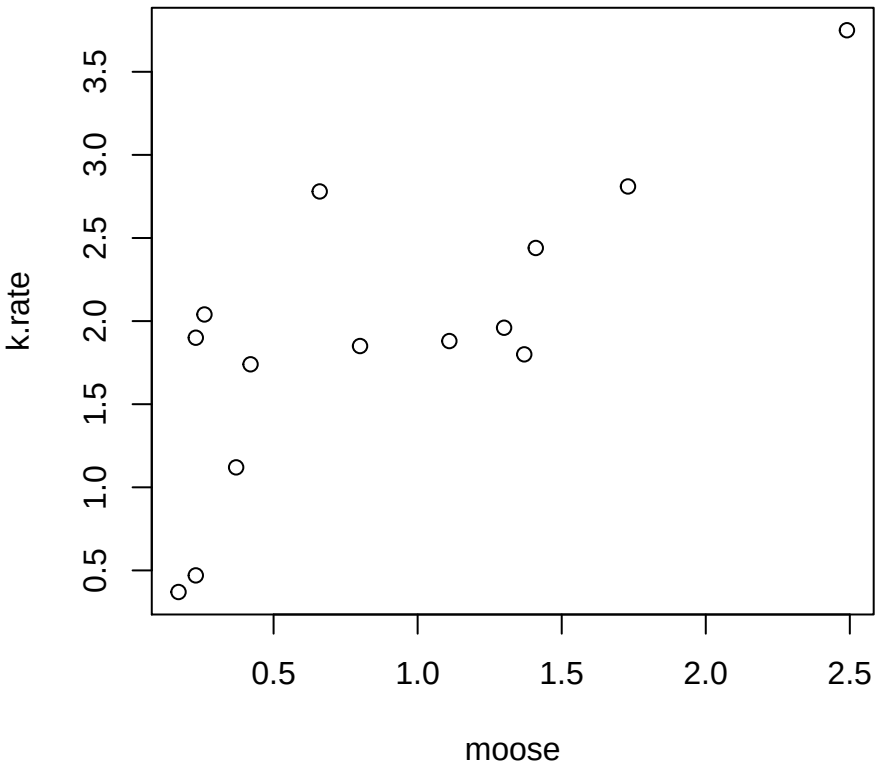
A Room With a Moose!

Real data here on moose density and wolves, specifically the kill rate defined as the average number of moose killed per wolf per 100 days. The moose density is the average number of moose per 1000 km^2.

This time we will enter the data using `c()`. In the future, I would most likely use a datafile and read it into `R` rather than type all values using `c()`. But here goes!

A Room With a Moose II

```
moose=c(.17,.23,.23,.26,.37,.42,.66,.8,1.11,
        1.3,1.37,1.41,1.73,2.49)
k.rate=c(.37,.47,1.9,2.04,1.12,1.74,2.78,1.85,
         1.88,1.96,1.8,2.44,2.81,3.75)
plot(moose,k.rate,type='p')
```



A Room With a Moose III

We are now going to look at the relationship between the feeding rate of an average wolf and the supply of moose. If you look at the graph, the average would be a line that starts at zero, increases steeply to 2, then increases at a decreasing rate to almost a flat horizontal line at the maximum. Ecologists have expressed this in a mathematical model, an equation that captures the essential expected relationship under "ideal" circumstances (only moose is varied, other factors are fixed).

$$k = \frac{am}{b+m}$$

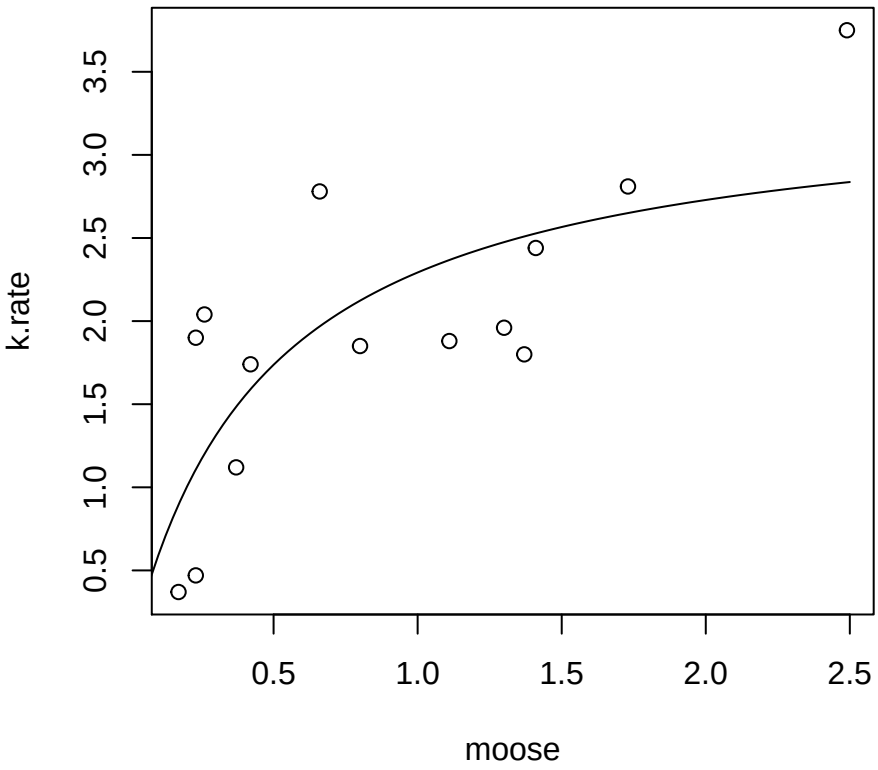
Where:
k: kill rate of an average predator
m: supply of prey
a , b: constants obtained from a curve fitting model (later!)

A Room With a Moose IV

Suppose we obtain the estimates and $a = 3.37$ and $b = 0.47$. We still need a vector for m. For those (to create a graph of the equation) we will use $m = 2.5(0:100)/100$ (moose density range from earlier data). `R` will compute the equation at all points and we can graph it.

A Room With a Moose V

```
m=2.5*(0:100)/100; a=3.37; b=.47
k=(a*m)/(b+m)
plot(moose,k.rate,type='p')
points(m,k,type='l') # adds points in line to existing graph
```


k.rate
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0.5
1.0
1.5
2.0
2.5
moose