Session 1

Ottar N. Bjornstad

2023-11-08

R Markdown

RStudio [https://www.rstudio.com] is an interface to the R language which is together with Python is the most widely used programming platform in many fields of computing. It is known for it's extremely broad statistical capabilities: cran.r-project.org serves out more than 10k add-on packages in addition to the R base... However in this workshop we will use it mainly for its raw numerical and computational power. To prep for the workshop, after installing Rstudio we need to install the **epimdr2** and **tinytex** packages either via the menu: "Tools -> Install packages'' or the console:

```
install.packages("epimdr2")
install.packages("tinytex")
```

Once a package has been installed, it is installed forever but needs to be attached to be used:

```
install.packages("epimdr2")
```

There are an enormous amount of online and printed resources for R. One usefull place is a search engine sole devoted to sifting through R related materials [https://rseek.org].

Six Things R may seem complicated, but the language itself is really only made up of six'ish things that is used to build objects.

Vectors are made forexample using the c() concatenate function... function are executed using round brackets:

```
x=c(1,5,3, 2, 23, 0)
x
```

```
## [1]  1  5  3  2 23  0
```

```
y=rep(1.25, 20)
y
```

```
##  [1] 1.25 1.25 1.25 1.25 1.25 1.25 1.25 1.25 1.25 1.25 1.25 1.25 1.25 1.25 1.25 1.25
## [16] 1.25 1.25 1.25 1.25 1.25
```

The [] indicate the element number for subsetting:

```
x[4]
```

```
## [1] 2
```

```
x[c(2,5)]
```

```
## [1]  5 23
```

Matrices are 2D arrays (there are higher D arrays also but we wont use them) which we subset by row column. or element:

```
z=matrix(x, nrow=3, byrow=TRUE)
z
```

```
##      [,1] [,2]
## [1,]    1    5
## [2,]    3    2
## [3,]   23    0
```

```
z[2,]
```

```
## [1] 3 2
```

```
z[,1]
```

```
## [1]  1  3 23
```

```
z[3,2]
```

```
## [1] 0
```

Data.frames are like simple spread-sheets. Easiest to generate by exporting a sheet to a comma separated file and then read in (note that names has to be in quotes otherwise R will try to evaluate it as an object), data.frames are accessed using $name or [,2]:

```
meas=read.csv("meas.csv", header=TRUE)
head(meas)
```

```
##   year week     time London    B
## 1   44    2 44.00000    180 1725
## 2   44    4 44.03846    271 1725
## 3   44    6 44.07692    423 1725
## 4   44    8 44.11538    465 1725
## 5   44   10 44.15385    523 1725
## 6   44   12 44.19231    649 1725
```

```
head(meas)$London
```

```
## [1] 180 271 423 465 523 649
```

```
head(meas)[,2]
```

```
## [1]  2  4  6  8 10 12
```

Lists are unstructured collection objects that can be accessed using $name or [[]] element number:

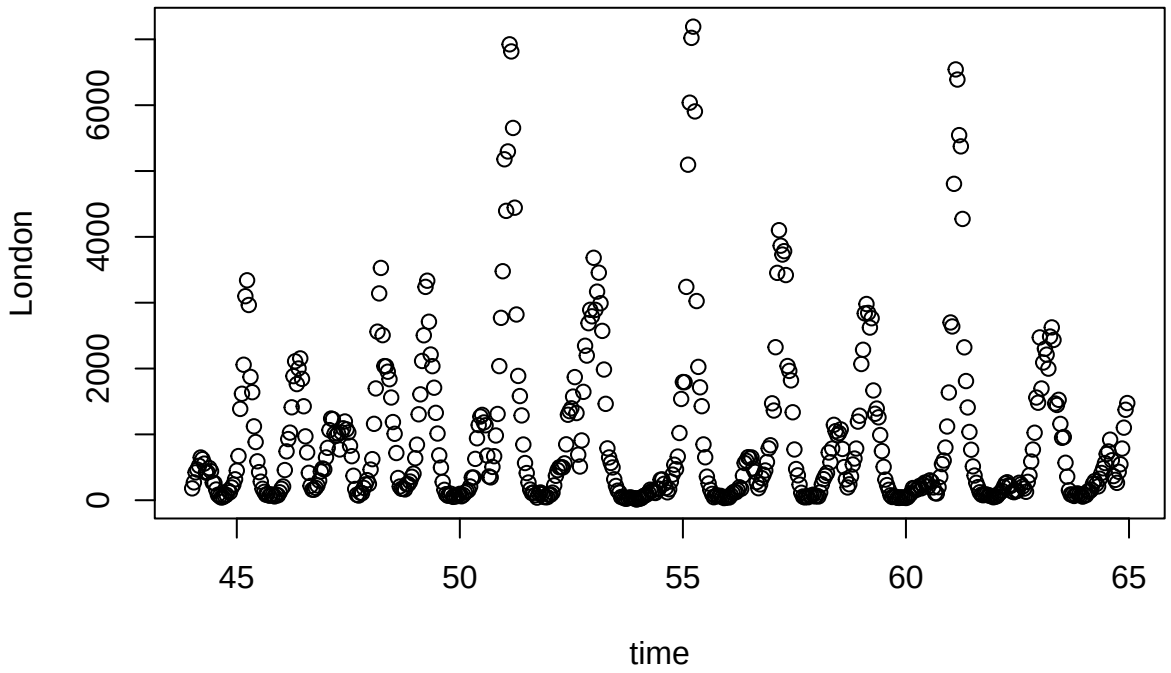
```
lst=list(a=x, b=y, pip=z)
lst$pip
```

```
##      [,1] [,2]
## [1,]    1    5
## [2,]    3    2
## [3,]   23    0
```

```
lst[[1]]
```

```
## [1]  1  5  3  2 23  0
```

Functions does things (when followed by () otherwise the function will just be displayed). We can for example use the plot() funtion to plot the measles data:



The ~ signifies a formula, i.e London as a function of time. We would get the same result from plot(x=meas$$time, y=meas$London). Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

We can also write our own functions which will be very important during the workshop:

```
myfn=function(arg){
ss=sum(arg)
return(ss)
}

myfn(y)
```

```
## [1] 25
```

Finally, loops are when we want to repeat something many times (we will visit more on this tomorrow):

```
for(i in 1:4){
  cat(x[1:i], "\n")
}
```

```
## 1
## 1 5
## 1 5 3
## 1 5 3 2
```

The cat() function simply prints to screen and the colon operator generates a vector of a:b whole numbers from a to b.