R recap

Mark Dunning; mark 'dot' dunning 'at' cruk.cam.ac.uk

Last modified: 06 Apr 2016

Pre-amble

In this session we will review some of the basic features of the R language, before proceeding more-complicated workflows required for the analysis of NGS, and other high-throughput data.

We recommend using the RStudio GUI for this course.

Getting help with R

R has an in-built help system. At the *console*, you can type `?` followed by the name of a function. This will bring-up the documentation for the function; which includes the expected inputs (*arguments*), the output you should expect from the function and some use-cases.

```
?mean
```

More-detailed information on particular packages is also available (see below)

R packages

The **Packages** tab in the bottom-right panel of RStudio lists all packages that you currently have installed. Clicking on a package name will show a list of functions that available once that package has been loaded. The `library` function is used to load a package and make it's functions / data available in your current R session. *You need to do this every time you load a new RStudio session.*

```
library(beadarray)
```

There are functions for installing packages within R. If your package is part of the main **CRAN** repository, you can use `install.packages`

We will be using the `wakefield` R package in this practical. To install it, we do.

```
install.packages("wakefield")
```

Bioconductor packages have their own install script, which you can download from the Bioconductor website

```
source("http://www.bioconductor.org/biocLite.R")
biocLite("affy")
```

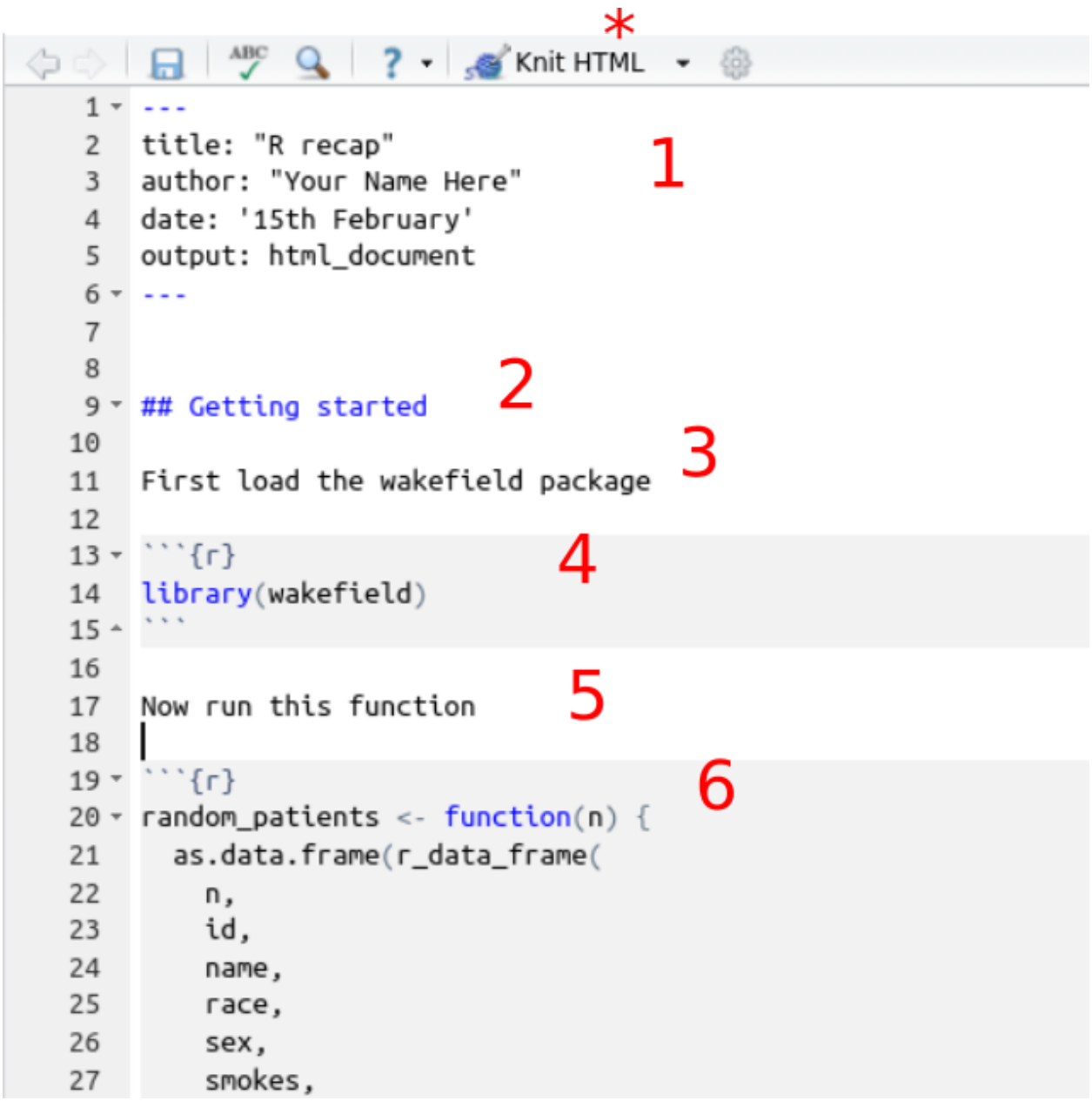
A package may have several *dependancies*; other R packages from which it uses functions or data types (re-using code from other packages is strongly-encouraged). If this is the case, the other R packages will be located and installed too.

So long as you stick with the same version of R, you won't need to repeat this install process.

About the R markdown format

Aside from teaching you about RNA-seq and ChIP-seq analysis, we also hope to teach you how to work in a reproducible manner. The first step in this process is to master the R markdown format.

Open the file `R-recap-template.Rmd` in Rstudio now. . . ..



1. Header information
2. Section heading
3. Plain text
4. R code to be run
5. Plain text
6. R code to be run

Each line of R code can be executed in the R console by placing the cursor on the line and pressing `CTRL + ENTER`. You can also highlight multiple lines of code. NB. You do not need to highlight to the backtick (“‘) symbols. Hitting the knit button (*) will run all R code in order and (providing there are no errors!) you will get a PDF or HTML document. The resultant document will contain all the plain text you wrote, the R code, and any outputs (including graphs, tables etc) that R produced. You can then distribute this document to have a reproducible account of your analysis.

How to use the template

- Change your name, add a title and date in the header section
- Add notes, explanations of code etc in the white space between code chunks. You can add new lines with `ENTER`. Clicking the `?` next to the `Knit HTML` button will give more information about how to format this text. You can introduce **bold** and *italics* for example.
- Some code chunks are left blank. These are for you to write the R code required to answer the questions
- You can try to knit the document at any point to see how it looks

The Practical

Getting started

We are going to explore some of the basic features of R using some patient data; the kind of data that we might encounter in the wild. However, rather than using real-life data we are going to make some up. There is a package called `wakefield` that is particularly convenient for this task.

```
library(wakefield)
```

Various patient characteristics can be generated. The following is a function that uses the package to create a *data frame* with various clinical characteristics. The number of patients we want to simulate is an argument.

Don’t worry about what the function does, you can just paste the following into the R console, or highlight it in the the markdown template and press `CTRL + ENTER` to run.

```
random_patients <- function(n) {
  as.data.frame(r_data_frame(
    n,
    id,
    name,
    race,
    sex,
    smokes,
    height,
    birth(random = TRUE, x = NULL, start = Sys.Date() - 365 * 45, k = 365*2,by = "1 days"),
    state,
    pet,
    grade_level(x=1:3),
    died,
    normal(name="Count"),
    date_stamp)
  )
}
```

We can now use the `random_patients` function to generate a data frame of fictitious patients

```
patients <- random_patients(100)
```

In Rstudio , you can view the contents of this data frame in a tab.

```
View(patients)
```

Q. What are the dimensions of the data frame?

Q. What columns are available?

*** HINT: see the `dim`, `ncol`, `nrow` and `colnames` functions

```
## [1] 100  13
```

```
##  [1] "ID"          "Name"        "Race"        "Sex"         "Smokes"
##  [6] "Height"      "Birth"       "State"       "Pet"         "Grade_Level"
## [11] "Died"        "Count"       "Date"
```

Q. Can you think of two ways to access the Names of the patients?

Q. What type of object is returned?

```
##   [1] "Britt"    "Martin"   "Young"    "Deon"     "Juan"     "Devon"
##   [7] "Adam"     "Cary"     "Yong"     "Clyde"    "Mary"     "Stacey"
##  [13] "Micah"    "Theo"     "Refugio"  "Toby"     "Chang"    "Connie"
##  [19] "Blair"    "Daniel"   "Paul"     "Devin"    "Joshua"   "Ira"
##  [25] "Casey"    "Oscar"    "Minh"     "Stacy"    "Kenneth"  "Julio"
##  [31] "Jude"     "Trinidad" "Cody"     "Laverne"  "Rudy"     "Larry"
##  [37] "Lonnie"   "Andrew"   "Jean"     "Odell"    "Andre"    "Timothy"
##  [43] "Cecil"    "Dee"      "John"     "Carlos"   "Wesley"   "Brandon"
##  [49] "Tracey"   "Johnnie"  "Ali"      "George"   "Carroll"  "Lewis"
##  [55] "Chris"    "Lawrence" "Otha"     "Sidney"   "Sydney"   "Royce"
##  [61] "James"    "Mario"    "Walter"   "Gary"     "Curtis"   "Lou"
##  [67] "Gail"     "Ronald"   "Frances"  "Keith"    "Steven"   "Carl"
##  [73] "Norman"   "Clair"    "Cleo"     "Carol"    "Ellis"    "Shannon"
##  [79] "Jamey"    "Rory"     "Dale"     "Marion"   "Leo"      "Peter"
##  [85] "Ollie"    "Michael"  "Billy"    "Morgan"   "Antonia"  "Corey"
##  [91] "Victor"   "Kim"      "Dean"     "Antonio"  "Shaun"    "Santos"
##  [97] "Leslie"   "Drew"     "Hollis"   "Lindsey"
```

```
##   [1] "Britt"    "Martin"   "Young"    "Deon"     "Juan"     "Devon"
##   [7] "Adam"     "Cary"     "Yong"     "Clyde"    "Mary"     "Stacey"
##  [13] "Micah"    "Theo"     "Refugio"  "Toby"     "Chang"    "Connie"
##  [19] "Blair"    "Daniel"   "Paul"     "Devin"    "Joshua"   "Ira"
##  [25] "Casey"    "Oscar"    "Minh"     "Stacy"    "Kenneth"  "Julio"
##  [31] "Jude"     "Trinidad" "Cody"     "Laverne"  "Rudy"     "Larry"
##  [37] "Lonnie"   "Andrew"   "Jean"     "Odell"    "Andre"    "Timothy"
##  [43] "Cecil"    "Dee"      "John"     "Carlos"   "Wesley"   "Brandon"
##  [49] "Tracey"   "Johnnie"  "Ali"      "George"   "Carroll"  "Lewis"
##  [55] "Chris"    "Lawrence" "Otha"     "Sidney"   "Sydney"   "Royce"
##  [61] "James"    "Mario"    "Walter"   "Gary"     "Curtis"   "Lou"
##  [67] "Gail"     "Ronald"   "Frances"  "Keith"    "Steven"   "Carl"
##  [73] "Norman"   "Clair"    "Cleo"     "Carol"    "Ellis"    "Shannon"
##  [79] "Jamey"    "Rory"     "Dale"     "Marion"   "Leo"      "Peter"
##  [85] "Ollie"    "Michael"  "Billy"    "Morgan"   "Antonia"  "Corey"
##  [91] "Victor"   "Kim"      "Dean"     "Antonio"  "Shaun"    "Santos"
##  [97] "Leslie"   "Drew"     "Hollis"   "Lindsey"
```

We can access the columns of a data frame by either

- knowing the column index
- knowing the column name

By column name is recommended, unless you can guarentee the columns will also be in the same order

TOP TIP: Use auto-complete with the key to get the name of the column correct

A vector (1-dimensional) is returned, the length of which is the same as the number of rows in the data frame. The vector could be stored as a variable and itself be subset or used in further calculations

```
peeps <- patients$Name
peeps
```

```
##   [1] "Britt"    "Martin"   "Young"    "Deon"     "Juan"     "Devon"
##   [7] "Adam"     "Cary"     "Yong"     "Clyde"    "Mary"     "Stacey"
##  [13] "Micah"    "Theo"     "Refugio"  "Toby"     "Chang"    "Connie"
##  [19] "Blair"    "Daniel"   "Paul"     "Devin"    "Joshua"   "Ira"
##  [25] "Casey"    "Oscar"    "Minh"     "Stacy"    "Kenneth"  "Julio"
##  [31] "Jude"     "Trinidad" "Cody"     "Laverne"  "Rudy"     "Larry"
##  [37] "Lonnie"   "Andrew"   "Jean"     "Odell"    "Andre"    "Timothy"
##  [43] "Cecil"    "Dee"      "John"     "Carlos"   "Wesley"   "Brandon"
##  [49] "Tracey"   "Johnnie"  "Ali"      "George"   "Carroll"  "Lewis"
##  [55] "Chris"    "Lawrence" "Otha"     "Sidney"   "Sydney"   "Royce"
##  [61] "James"    "Mario"    "Walter"   "Gary"     "Curtis"   "Lou"
##  [67] "Gail"     "Ronald"   "Frances"  "Keith"    "Steven"   "Carl"
##  [73] "Norman"   "Clair"    "Cleo"     "Carol"    "Ellis"    "Shannon"
##  [79] "Jamey"    "Rory"     "Dale"     "Marion"   "Leo"      "Peter"
##  [85] "Ollie"    "Michael"  "Billy"    "Morgan"   "Antonia"  "Corey"
##  [91] "Victor"   "Kim"      "Dean"     "Antonio"  "Shaun"    "Santos"
##  [97] "Leslie"   "Drew"     "Hollis"   "Lindsey"
```

```
length(peeps)
```

```
## [1] 100
```

```r
nchar(peeps)
```

```
##   [1] 5 6 5 4 4 5 4 4 4 5 4 6 5 4 7 4 5 6 5 6 4 5 6 3 5 5 4 5 7 5 4 8 4 7 4
##  [36] 5 6 6 4 5 5 7 5 3 4 6 6 7 6 7 3 6 7 5 5 8 4 6 6 5 5 5 6 4 6 3 4 6 7 5
##  [71] 6 4 6 5 4 5 5 7 5 4 4 6 3 5 5 7 5 6 7 5 6 3 4 7 5 6 6 4 6 7
```

```r
substr(peeps,1,3)
```

```
##   [1] "Bri" "Mar" "You" "Deo" "Jua" "Dev" "Ada" "Car" "Yon" "Cly" "Mar"
##  [12] "Sta" "Mic" "The" "Ref" "Tob" "Cha" "Con" "Bla" "Dan" "Pau" "Dev"
##  [23] "Jos" "Ira" "Cas" "Osc" "Min" "Sta" "Ken" "Jul" "Jud" "Tri" "Cod"
##  [34] "Lav" "Rud" "Lar" "Lon" "And" "Jea" "Ode" "And" "Tim" "Cec" "Dee"
##  [45] "Joh" "Car" "Wes" "Bra" "Tra" "Joh" "Ali" "Geo" "Car" "Lew" "Chr"
##  [56] "Law" "Oth" "Sid" "Syd" "Roy" "Jam" "Mar" "Wal" "Gar" "Cur" "Lou"
##  [67] "Gai" "Ron" "Fra" "Kei" "Ste" "Car" "Nor" "Cla" "Cle" "Car" "Ell"
##  [78] "Sha" "Jam" "Ror" "Dal" "Mar" "Leo" "Pet" "Oll" "Mic" "Bil" "Mor"
##  [89] "Ant" "Cor" "Vic" "Kim" "Dea" "Ant" "Sha" "San" "Les" "Dre" "Hol"
## [100] "Lin"
```

The `summary` function is a useful way of summarising the data containing in each column. It will give information about the *type* of data (remember, data frames can have a mixture of numeric and character columns) and also an appropriate summary. For numeric columns, it will report some stats about the distribution of the data. For categorical data, it will report the different *levels*.

```r
summary(patients)
```

```
##       ID                 Name                   Race        Sex
##  Length:100         Length:100         White    :55   Male  :50
##  Class :character   Class :character   Hispanic :22   Female:50
##  Mode  :character   Mode  :character   Black    :16
##                                        Asian    : 6
##                                        Native   : 1
##                                        Bi-Racial: 0
##                                        (Other)  : 0
##    Smokes            Height          Birth                   State
##  Mode :logical   Min.   :62.00   Min.   :1971-04-29   California: 9
##  FALSE:81        1st Qu.:66.00   1st Qu.:1971-09-03   New Jersey: 9
##  TRUE :19        Median :69.00   Median :1972-03-28   New York  : 8
##  NA's :0         Mean   :68.89   Mean   :1972-03-24   Florida   : 6
##                  3rd Qu.:71.00   3rd Qu.:1972-08-24   Texas     : 6
##                  Max.   :79.00   Max.   :1973-04-09   Georgia   : 5
##                                                       (Other)   :57
##     Pet     Grade_Level    Died             Count
##  Dog  :42   1:35        Mode :logical   Min.   :-2.22434
##  Cat  :16   2:38        FALSE:51        1st Qu.:-0.53078
##  None :36   3:27        TRUE :49        Median :-0.06120
##  Bird : 4               NA's :0         Mean   : 0.03244
##  Horse: 2                               3rd Qu.: 0.64184
##                                         Max.   : 2.45857
##
```

```
##       Date
##  Min.   :2015-05-06
##  1st Qu.:2015-07-06
##  Median :2015-10-06
##  Mean   :2015-10-08
##  3rd Qu.:2016-01-06
##  Max.   :2016-04-06
##
```

Q. Can you identify

which columns contain numerical data?

which columns contain categorical data?

which columns contain logical (`TRUE` or `FALSE`) values?

Subsetting

A data frame can be subset using square brackes `[]` placed after the name of the data frame. As a data frame is a two-dimensional object, you need a *row* and *column* index, or vector indices.

Q. Make sure you can understand the behaviour of the following commands

---

```
patients[1,2]
patients[2,1]
patients[c(1,2,3),1]
patients[c(1,2,3),c(1,2,3)]
```

Note that the data frame is not altered we are just seeing what a subset of the data looks like and not changing the underlying data. If we wanted to do this, we would need to create a new variale.

```
patients
```

Should we wish to see all rows, or all columns, we can neglect either the row or column index

---

Q. Make sure you can understand the behaviour of the following commands

---

```
patients[1,]
patients[,1]
patients[,c(1,2)]
```

`head` is commonly-used to give a snapshot of a data frame. Otherwise, you can use the `[row,column]` notation.

```
##    ID   Name     Race    Sex Smokes Height      Birth        State  Pet
## 1 001  Britt    Black   Male   TRUE     65 1973-04-06       Kansas None
## 2 002 Martin Hispanic Female  FALSE     73 1971-07-19   New Jersey  Dog
## 3 003  Young Hispanic   Male  FALSE     66 1972-02-10 Pennsylvania  Dog
## 4 004   Deon   Native Female   TRUE     66 1972-06-20      Florida  Dog
## 5 005   Juan    White Female  FALSE     73 1972-04-07    Wisconsin None
## 6 006  Devon    White Female  FALSE     79 1971-08-20     Arkansas  Dog
##   Grade_Level  Died       Count       Date
## 1           3 FALSE  0.01228378 2015-05-06
## 2           1  TRUE -2.22434373 2015-05-06
## 3           3  TRUE -0.41127709 2015-05-06
## 4           1  TRUE -1.37401244 2015-05-06
## 5           1 FALSE -1.16615907 2015-05-06
## 6           3 FALSE -1.30834218 2015-05-06
```

Rather than selecting rows based on their *numeric* index (as in the previous example) we can use what we call a *logical test*. This is a test that gives either a `TRUE` or `FALSE` result. When applied to subsetting, only rows with a `TRUE` result get returned.

For example we could compare the `Count` variable to zero. The result is a *vector* of `TRUE` or `FALSE`; one for each row in the data frame

```
patients$Count < 0
```

```
##   [1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
##  [12] FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE
##  [23] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
##  [34] FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
##  [45] FALSE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE
##  [56]  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
##  [67]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [78] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE
##  [89]  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE FALSE
## [100]  TRUE
```

This R code can be put inside the square brackets.

```
patients[patients$Count<0, ]
```

```
##    ID   Name     Race    Sex Smokes Height      Birth        State  Pet
## 2 002 Martin Hispanic Female  FALSE     73 1971-07-19   New Jersey  Dog
## 3 003  Young Hispanic   Male  FALSE     66 1972-02-10 Pennsylvania  Dog
## 4 004   Deon   Native Female   TRUE     66 1972-06-20      Florida  Dog
## 5 005   Juan    White Female  FALSE     73 1972-04-07    Wisconsin None
## 6 006  Devon    White Female  FALSE     79 1971-08-20     Arkansas  Dog
## 7 007   Adam Hispanic   Male  FALSE     68 1971-09-20      Wyoming  Dog
##   Grade_Level  Died      Count       Date
## 2           1  TRUE -2.2243437 2015-05-06
## 3           3  TRUE -0.4112771 2015-05-06
## 4           1  TRUE -1.3740124 2015-05-06
## 5           1 FALSE -1.1661591 2015-05-06
## 6           3 FALSE -1.3083422 2015-05-06
## 7           3  TRUE -1.6454587 2015-05-06
```

If we wanted to know about the patients that had died, we could do;

```
deceased <- patients[patients$Died == TRUE,]
```

```
deceased
```

```
##     ID   Name     Race    Sex Smokes Height      Birth        State  Pet
## 2  002 Martin Hispanic Female  FALSE     73 1971-07-19   New Jersey  Dog
## 3  003  Young Hispanic   Male  FALSE     66 1972-02-10 Pennsylvania  Dog
## 4  004   Deon   Native Female   TRUE     66 1972-06-20      Florida  Dog
## 7  007   Adam Hispanic   Male  FALSE     68 1971-09-20      Wyoming  Dog
## 11 011   Mary    White   Male  FALSE     72 1972-02-27      Montana None
## 12 012 Stacey Hispanic   Male  FALSE     68 1972-03-29     New York None
##    Grade_Level Died      Count       Date
## 2            1 TRUE -2.2243437 2015-05-06
## 3            3 TRUE -0.4112771 2015-05-06
## 4            1 TRUE -1.3740124 2015-05-06
## 7            3 TRUE -1.6454587 2015-05-06
## 11           1 TRUE -0.7416127 2015-05-06
## 12           2 TRUE  1.7317718 2015-06-06
```

In fact, this is equivalent

```
deceased <- patients[patients$Died,]
```

The test of equality `==` also works for text

```
patients[patients$Race == "White",]
```

```
##     ID    Name  Race    Sex Smokes Height      Birth          State  Pet
## 5  005    Juan White Female  FALSE     73 1972-04-07      Wisconsin None
## 6  006   Devon White Female  FALSE     79 1971-08-20       Arkansas  Dog
## 10 010   Clyde White   Male  FALSE     74 1972-07-07        Alabama None
## 11 011    Mary White   Male  FALSE     72 1972-02-27        Montana None
## 15 015 Refugio White Female  FALSE     74 1971-08-02 North Carolina  Dog
## 17 017   Chang White   Male   TRUE     68 1971-05-02          Texas None
##    Grade_Level  Died      Count       Date
```

```
## 5            1 FALSE -1.1661591 2015-05-06
## 6            3 FALSE -1.3083422 2015-05-06
## 10           3 FALSE  0.1924214 2015-05-06
## 11           1  TRUE -0.7416127 2015-05-06
## 15           3  TRUE  1.2337392 2015-06-06
## 17           1  TRUE -0.1831558 2015-07-06
```

Q. Can you create a data frame of dog owners?

```
##    ID   Name     Race    Sex Smokes Height      Birth        State Pet
## 2 002 Martin Hispanic Female  FALSE     73 1971-07-19   New Jersey Dog
## 3 003  Young Hispanic   Male  FALSE     66 1972-02-10 Pennsylvania Dog
## 4 004   Deon   Native Female   TRUE     66 1972-06-20      Florida Dog
## 6 006  Devon    White Female  FALSE     79 1971-08-20     Arkansas Dog
## 7 007   Adam Hispanic   Male  FALSE     68 1971-09-20      Wyoming Dog
## 8 008   Cary Hispanic   Male  FALSE     79 1973-02-01      Florida Dog
##   Grade_Level  Died      Count       Date
## 2           1  TRUE -2.2243437 2015-05-06
## 3           3  TRUE -0.4112771 2015-05-06
## 4           1  TRUE -1.3740124 2015-05-06
## 6           3 FALSE -1.3083422 2015-05-06
## 7           3  TRUE -1.6454587 2015-05-06
## 8           1 FALSE -0.6606125 2015-05-06
```

There are a couple of ways of testing for more than one text value. The first uses an *or* `|` statement. i.e. testing if the value of `Pet` is `Dog` *or* the value is `Cat`.

The `%in%` function is a convenient function for testing which items in a vector correspond to a defined set of values.

```r
patients[patients$Pet == "Dog" | patients$Pet == "Cat",]
patients[patients$Pet %in% c("Dog","Cat"),]
```

```
##    ID   Name     Race    Sex Smokes Height      Birth        State Pet
## 2 002 Martin Hispanic Female  FALSE     73 1971-07-19   New Jersey Dog
## 3 003  Young Hispanic   Male  FALSE     66 1972-02-10 Pennsylvania Dog
## 4 004   Deon   Native Female   TRUE     66 1972-06-20      Florida Dog
## 6 006  Devon    White Female  FALSE     79 1971-08-20     Arkansas Dog
## 7 007   Adam Hispanic   Male  FALSE     68 1971-09-20      Wyoming Dog
## 8 008   Cary Hispanic   Male  FALSE     79 1973-02-01      Florida Dog
##   Grade_Level  Died      Count       Date
## 2           1  TRUE -2.2243437 2015-05-06
## 3           3  TRUE -0.4112771 2015-05-06
## 4           1  TRUE -1.3740124 2015-05-06
## 6           3 FALSE -1.3083422 2015-05-06
## 7           3  TRUE -1.6454587 2015-05-06
## 8           1 FALSE -0.6606125 2015-05-06
```

```
##    ID   Name     Race    Sex Smokes Height      Birth        State Pet
## 2 002 Martin Hispanic Female  FALSE     73 1971-07-19   New Jersey Dog
## 3 003  Young Hispanic   Male  FALSE     66 1972-02-10 Pennsylvania Dog
## 4 004   Deon   Native Female   TRUE     66 1972-06-20      Florida Dog
## 6 006  Devon    White Female  FALSE     79 1971-08-20     Arkansas Dog
## 7 007   Adam Hispanic   Male  FALSE     68 1971-09-20      Wyoming Dog
## 8 008   Cary Hispanic   Male  FALSE     79 1973-02-01      Florida Dog
##   Grade_Level  Died      Count       Date
## 2           1  TRUE -2.2243437 2015-05-06
## 3           3  TRUE -0.4112771 2015-05-06
## 4           1  TRUE -1.3740124 2015-05-06
## 6           3 FALSE -1.3083422 2015-05-06
## 7           3  TRUE -1.6454587 2015-05-06
## 8           1 FALSE -0.6606125 2015-05-06
```

Similar to *or*, we can require that both tests are `TRUE` by using an *and* `&` operation. e.g. to look for white males.

```r
patients[patients$Race == "White" & patients$Sex =="Male",]
```

```r
head(patients[patients$Race == "White" & patients$Sex =="Male",])
```

```
##     ID  Name  Race  Sex Smokes Height      Birth        State  Pet
## 10 010 Clyde White Male  FALSE     74 1972-07-07      Alabama None
## 11 011  Mary White Male  FALSE     72 1972-02-27      Montana None
## 17 017 Chang White Male   TRUE     68 1971-05-02        Texas None
## 19 019 Blair White Male  FALSE     69 1971-11-25 Pennsylvania  Dog
## 24 024   Ira White Male  FALSE     68 1972-11-15      Indiana Bird
## 25 025 Casey White Male  FALSE     75 1971-05-08     New York  Dog
##    Grade_Level  Died       Count       Date
## 10           3 FALSE  0.19242136 2015-05-06
## 11           1  TRUE -0.74161272 2015-05-06
## 17           1  TRUE -0.18315577 2015-07-06
## 19           2  TRUE  0.72404776 2015-07-06
## 24           1 FALSE  1.97621969 2015-07-06
## 25           1  TRUE  0.09502301 2015-07-06
```

Q. Can you create a data frame of deceased patients with a 'count' < 0

```
##     ID   Name     Race    Sex Smokes Height      Birth        State  Pet
## 2  002 Martin Hispanic Female  FALSE     73 1971-07-19   New Jersey  Dog
## 3  003  Young Hispanic   Male  FALSE     66 1972-02-10 Pennsylvania  Dog
## 4  004   Deon   Native Female   TRUE     66 1972-06-20      Florida  Dog
## 7  007   Adam Hispanic   Male  FALSE     68 1971-09-20      Wyoming  Dog
## 11 011   Mary    White   Male  FALSE     72 1972-02-27      Montana None
## 13 013  Micah Hispanic   Male  FALSE     62 1972-12-31        Texas  Cat
##    Grade_Level Died      Count       Date
```

```
## 2            1 TRUE -2.2243437 2015-05-06
## 3            3 TRUE -0.4112771 2015-05-06
## 4            1 TRUE -1.3740124 2015-05-06
## 7            3 TRUE -1.6454587 2015-05-06
## 11           1 TRUE -0.7416127 2015-05-06
## 13           2 TRUE -0.2532091 2015-06-06
```

We can also use the negation operator `!` to find which entries are *not* equal to a particular value. For example, patients that do *not* own a dog can be found in the following way.

```
patients[patients$Pet != "Dog",]
```

```
##      ID     Name     Race    Sex Smokes Height      Birth          State
## 1   001    Britt    Black   Male   TRUE     65 1973-04-06         Kansas
## 5   005     Juan    White Female  FALSE     73 1972-04-07      Wisconsin
## 9   009     Yong    Black Female  FALSE     64 1971-12-15        Georgia
## 10  010    Clyde    White   Male  FALSE     74 1972-07-07        Alabama
## 11  011     Mary    White   Male  FALSE     72 1972-02-27        Montana
## 12  012   Stacey Hispanic   Male  FALSE     68 1972-03-29       New York
## 13  013    Micah Hispanic   Male  FALSE     62 1972-12-31          Texas
## 17  017    Chang    White   Male   TRUE     68 1971-05-02          Texas
## 20  020   Daniel Hispanic Female  FALSE     70 1971-12-13          Texas
## 22  022    Devin Hispanic Female  FALSE     67 1972-05-08   Pennsylvania
## 23  023   Joshua    White Female  FALSE     66 1972-07-10        Arizona
## 24  024      Ira    White   Male  FALSE     68 1972-11-15        Indiana
## 27  027     Minh    White   Male  FALSE     64 1972-10-24     California
## 28  028    Stacy    White   Male  FALSE     66 1972-10-29     New Jersey
## 29  029  Kenneth    Black Female  FALSE     69 1972-10-28        Georgia
## 30  030    Julio    White Female  FALSE     73 1972-08-30           Ohio
## 31  031     Jude Hispanic   Male  FALSE     75 1971-10-29        Georgia
## 33  033     Cody    White   Male  FALSE     68 1972-03-27        Georgia
## 34  034  Laverne    White Female  FALSE     68 1972-05-25      Wisconsin
## 36  036    Larry    Black Female   TRUE     71 1973-04-07      Louisiana
## 39  039     Jean    Black Female   TRUE     71 1973-04-05      Wisconsin
## 40  040    Odell Hispanic   Male  FALSE     67 1972-03-10     New Jersey
## 42  042  Timothy    White   Male  FALSE     75 1971-05-14        Florida
## 43  043    Cecil    White   Male   TRUE     70 1973-04-09           Ohio
## 45  045     John    Asian Female  FALSE     71 1971-06-04      Tennessee
## 46  046   Carlos    White Female  FALSE     69 1971-07-17   Rhode Island
## 47  047   Wesley    White   Male   TRUE     68 1972-04-23       Virginia
## 49  049   Tracey Hispanic Female  FALSE     67 1971-11-18      Tennessee
## 51  051      Ali    Black   Male  FALSE     66 1972-08-19     New Jersey
## 53  053  Carroll    White   Male  FALSE     63 1971-08-12       Michigan
## 54  054    Lewis    White   Male  FALSE     72 1972-06-21   Rhode Island
## 56  056 Lawrence    Black Female  FALSE     70 1972-09-17           Ohio
## 57  057     Otha    Black   Male  FALSE     62 1971-07-26       New York
## 58  058   Sidney Hispanic Female  FALSE     62 1972-05-29   Pennsylvania
## 62  062    Mario    White Female  FALSE     73 1972-05-01     California
## 63  063   Walter Hispanic Female   TRUE     65 1972-08-21          Texas
## 65  065   Curtis    Black Female  FALSE     71 1971-08-27 North Carolina
## 66  066      Lou    White Female   TRUE     70 1973-03-26  Massachusetts
## 67  067     Gail    White Female  FALSE     68 1971-06-13         Nevada
## 68  068   Ronald    Black   Male  FALSE     64 1971-06-26   Rhode Island
```

```
## 72  072     Carl    Asian Female   TRUE     70 1972-03-03     California
## 73  073   Norman    Black Female  FALSE     73 1971-12-16     New Jersey
## 75  075     Cleo    White   Male  FALSE     65 1972-08-03       Missouri
## 76  076    Carol    White   Male   TRUE     63 1971-12-18     California
## 77  077    Ellis Hispanic   Male  FALSE     71 1971-08-26        Indiana
## 79  079    Jamey    White Female  FALSE     66 1971-11-21 North Carolina
## 81  081     Dale    Asian Female   TRUE     69 1971-06-10       New York
## 82  082   Marion    White Female  FALSE     67 1972-04-29       New York
## 83  083      Leo    White Female  FALSE     68 1972-01-03           Ohio
## 84  084    Peter    White   Male  FALSE     68 1971-10-17        Arizona
## 85  085    Ollie    Asian   Male  FALSE     66 1972-07-14       Michigan
## 87  087    Billy    White   Male  FALSE     70 1971-08-19           Utah
## 89  089  Antonia    White   Male  FALSE     69 1972-02-02        Florida
## 90  090    Corey    White Female   TRUE     71 1971-09-11       Nebraska
## 91  091   Victor    White   Male  FALSE     77 1972-11-05     California
## 95  095    Shaun    Black Female  FALSE     72 1971-05-31        Arizona
## 98  098     Drew    Asian Female   TRUE     69 1972-05-07        Florida
## 100 100  Lindsey Hispanic   Male  FALSE     69 1971-07-06      Wisconsin
##       Pet Grade_Level  Died        Count       Date
## 1    None           3 FALSE  0.012283780 2015-05-06
## 5    None           1 FALSE -1.166159075 2015-05-06
## 9    Bird           2 FALSE -0.180908779 2015-05-06
## 10   None           3 FALSE  0.192421364 2015-05-06
## 11   None           1  TRUE -0.741612716 2015-05-06
## 12   None           2  TRUE  1.731771835 2015-06-06
## 13    Cat           2  TRUE -0.253209146 2015-06-06
## 17   None           1  TRUE -0.183155767 2015-07-06
## 20   Bird           2 FALSE -0.524699454 2015-07-06
## 22   None           1  TRUE  0.389477423 2015-07-06
## 23   None           2 FALSE  0.077667923 2015-07-06
## 24   Bird           1 FALSE  1.976219689 2015-07-06
## 27   None           3  TRUE  0.434665567 2015-08-06
## 28  Horse           2  TRUE  1.279391032 2015-08-06
## 29   None           1 FALSE  1.153771134 2015-08-06
## 30    Cat           3  TRUE  0.080266908 2015-08-06
## 31   None           1 FALSE -0.110386615 2015-08-06
## 33   None           2  TRUE  0.058897714 2015-08-06
## 34   Bird           2  TRUE  0.128140634 2015-08-06
## 36   None           2  TRUE  0.755164647 2015-08-06
## 39   None           2 FALSE -0.416769257 2015-08-06
## 40   None           2 FALSE -0.533597658 2015-09-06
## 42   None           1 FALSE  0.178015812 2015-09-06
## 43    Cat           2 FALSE -0.314486711 2015-09-06
## 45   None           2 FALSE  0.692157422 2015-09-06
## 46   None           2  TRUE -0.666023938 2015-09-06
## 47   None           2  TRUE -1.104948564 2015-09-06
## 49    Cat           3  TRUE -0.092381269 2015-10-06
## 51   None           1  TRUE  2.141152266 2015-10-06
## 53    Cat           3 FALSE -0.080056562 2015-10-06
## 54    Cat           1  TRUE  0.141241954 2015-10-06
## 56   None           3  TRUE -0.003055163 2015-10-06
## 57   None           3  TRUE -0.529835561 2015-10-06
## 58   None           2 FALSE -1.225654132 2015-11-06
## 62   None           1 FALSE  2.458571936 2015-11-06
```

```
## 63    Cat           1 FALSE  1.612507446 2015-11-06
## 65    Cat           1  TRUE -1.452302947 2015-11-06
## 66   None           2 FALSE  1.062928611 2015-12-06
## 67   None           1 FALSE -0.344224630 2015-12-06
## 68    Cat           1  TRUE -0.055319566 2015-12-06
## 72   None           3  TRUE -1.365200549 2015-12-06
## 73   None           2  TRUE -0.683187521 2015-12-06
## 75   None           1 FALSE -0.387245916 2016-01-06
## 76    Cat           3  TRUE -0.622979667 2016-01-06
## 77   None           3 FALSE -0.342490273 2016-01-06
## 79   None           1  TRUE  0.100516490 2016-02-06
## 81   None           1 FALSE  0.201379824 2016-02-06
## 82   None           2  TRUE  1.220997286 2016-02-06
## 83   None           1 FALSE  1.356711062 2016-02-06
## 84    Cat           1 FALSE -1.971550913 2016-02-06
## 85    Cat           3  TRUE -0.011059132 2016-02-06
## 87    Cat           1 FALSE -0.339175685 2016-03-06
## 89  Horse           2 FALSE -0.561276637 2016-03-06
## 90    Cat           2 FALSE -0.630353839 2016-03-06
## 91   None           3 FALSE  0.920329549 2016-03-06
## 95   None           2 FALSE -0.267360034 2016-04-06
## 98    Cat           3 FALSE -1.881609717 2016-04-06
## 100   Cat           1  TRUE -0.099847187 2016-04-06
```

Ordering and sorting

A vector can be returned in sorted form using the `sort` function.

```r
sort(peeps)
sort(patients$Count,decreasing = TRUE)
```

However, if we want to sort an entire data frame a different approach is needed. The trick is to use `order`. Rather than giving a sorted set of *values*, it will give sorted *indices*.

```r
patients[order(patients$Count),]
patients[order(patients$Sex),]
```

```
##     ID   Name     Race    Sex Smokes Height      Birth          State Pet
## 2  002 Martin Hispanic Female  FALSE     73 1971-07-19     New Jersey Dog
## 84 084  Peter    White   Male  FALSE     68 1971-10-17        Arizona Cat
## 98 098   Drew    Asian Female   TRUE     69 1972-05-07        Florida Cat
## 7  007   Adam Hispanic   Male  FALSE     68 1971-09-20        Wyoming Dog
## 41 041  Andre    Black   Male  FALSE     68 1972-02-08       Illinois Dog
## 65 065 Curtis    Black Female  FALSE     71 1971-08-27 North Carolina Cat
##    Grade_Level  Died     Count       Date
## 2            1  TRUE -2.224344 2015-05-06
## 84           1 FALSE -1.971551 2016-02-06
## 98           3 FALSE -1.881610 2016-04-06
## 7            3  TRUE -1.645459 2015-05-06
## 41           2 FALSE -1.616096 2015-09-06
## 65           1  TRUE -1.452303 2015-11-06
```

```
##     ID  Name     Race  Sex Smokes Height      Birth        State  Pet
## 1  001 Britt    Black Male   TRUE     65 1973-04-06       Kansas None
## 3  003 Young Hispanic Male  FALSE     66 1972-02-10 Pennsylvania  Dog
## 7  007  Adam Hispanic Male  FALSE     68 1971-09-20      Wyoming  Dog
## 8  008  Cary Hispanic Male  FALSE     79 1973-02-01      Florida  Dog
## 10 010 Clyde    White Male  FALSE     74 1972-07-07      Alabama None
## 11 011  Mary    White Male  FALSE     72 1972-02-27      Montana None
##    Grade_Level  Died       Count       Date
## 1            3 FALSE  0.01228378 2015-05-06
## 3            3  TRUE -0.41127709 2015-05-06
## 7            3  TRUE -1.64545873 2015-05-06
## 8            1 FALSE -0.66061246 2015-05-06
## 10           3 FALSE  0.19242136 2015-05-06
## 11           1  TRUE -0.74161272 2015-05-06
```

Q. Create a data frame where the patients are arranged in decreasing height order

---

```
##     ID   Name     Race    Sex Smokes Height      Birth      State  Pet
## 6  006  Devon    White Female  FALSE     79 1971-08-20   Arkansas  Dog
## 8  008   Cary Hispanic   Male  FALSE     79 1973-02-01    Florida  Dog
## 91 091 Victor    White   Male  FALSE     77 1972-11-05 California None
## 96 096 Santos    White   Male  FALSE     76 1971-11-12    Georgia  Dog
## 25 025  Casey    White   Male  FALSE     75 1971-05-08   New York  Dog
## 31 031   Jude Hispanic   Male  FALSE     75 1971-10-29    Georgia None
##    Grade_Level  Died       Count       Date
## 6            3 FALSE -1.30834218 2015-05-06
## 8            1 FALSE -0.66061246 2015-05-06
## 91           3 FALSE  0.92032955 2016-03-06
## 96           2  TRUE  0.47041682 2016-04-06
## 25           1  TRUE  0.09502301 2015-07-06
## 31           1 FALSE -0.11038661 2015-08-06
```

A final point on data frames is that we can export them out of R once we have done our data processing.

```r
countOrder <- patients[order(patients$Count),]
write.csv(countOrder, file="patientsOrderedByCount.csv")
```

Simple plotting

Various simple plots are supported in the *base* distribution of R (what you get automatically when you download R). In the course, we will show how some of these plots can be used to inform us about the quality of NGS data, and to visualise our results.

Plotting is discussed in greater length on our introductory R course and a useful reference is the Quick-R page.

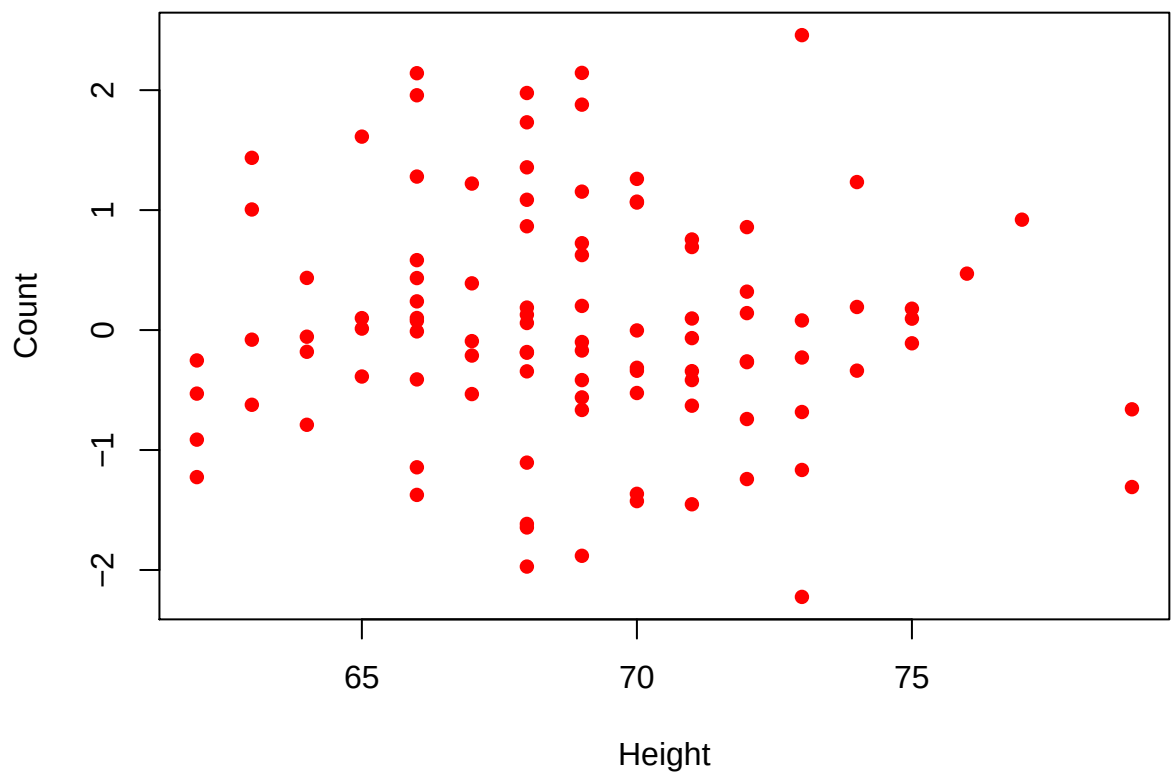
```r
hist(patients$Height)
```

```r
plot(patients$Height,patients$Count)
```

```r
barplot(table(patients$Race))
```

```r
boxplot(patients$Count ~ patients$Died)
```

Lots of customisations are possible to enhance the appaerance of our plots; colour, labels, axes, legends




```r
boxplot(patients$Count ~ patients$Died,col=c("red","yellow"))
```

Make the following plots

1. **Histogram of the Count variable**
2. **Barplot of the frequency of pet ownership**
3. **Boxplot of Height according to smoker / non-smoker**

...anything else that takes your fancy

Plots can be exported by the ***Plots*** tab in RStudio, or by calling the `pdf` or `png` functions which will write the plot to a file

```
png("myLittlePlot.png")
barplot(table(patients$Pet))
dev.off()
```

```
## pdf
##   2
```