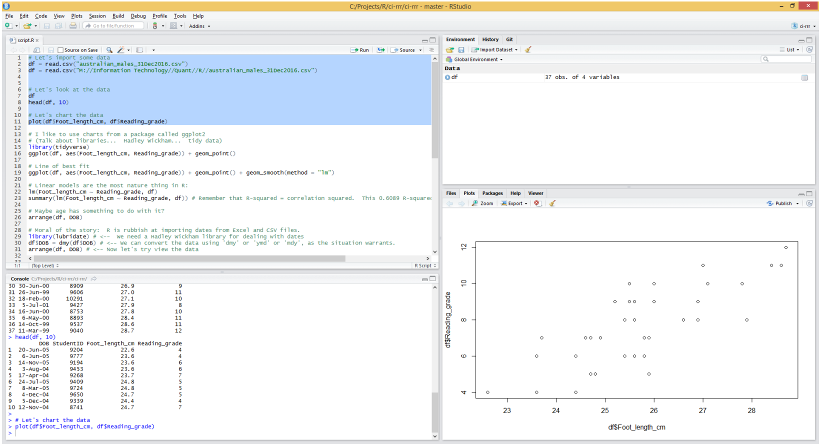
R workshop

Part I: The IDE



```
# Let's import some data
df = read.csv("australian_males_31Dec2016.csv")
df = read.csv("M://Information Technology//Quant//R//australian_males_31Dec2016.csv")


# Let's look at the data
df
head(df, 10)

# Let's chart the data
plot(df$Foot_length_cm, df$Reading_grade)

# I like to use charts from a package called ggplot2
# (Talk about libraries...  Hadley Wickham...  tidy data)
library(tidyverse)
ggplot(df, aes(Foot_length_cm, Reading_grade)) + geom_point()

# Line of best fit
ggplot(df, aes(Foot_length_cm, Reading_grade)) + geom_point() + geom_smooth(method = "lm")

# Linear models are the most nature thing in R:
lm(Foot_length_cm ~ Reading_grade, df)
summary(lm(Foot_length_cm ~ Reading_grade, df)) # Remember that R-squared = correlation squared.  This 0.6089 R-squared

# Maybe age has something to do with it?
arrange(df, DOB)

# Moral of the story:  R is rubbish at importing dates from Excel and CSV files.
library(lubridate) # <--  We need a Hadley Wickham library for dealing with dates
df$DOB = dmy(df$DOB) # <-- We can convert the data using 'dmy' or 'ymd' or 'mdy', as the situation warrants.
arrange(df, DOB) # <-- Now let's try view the data
```

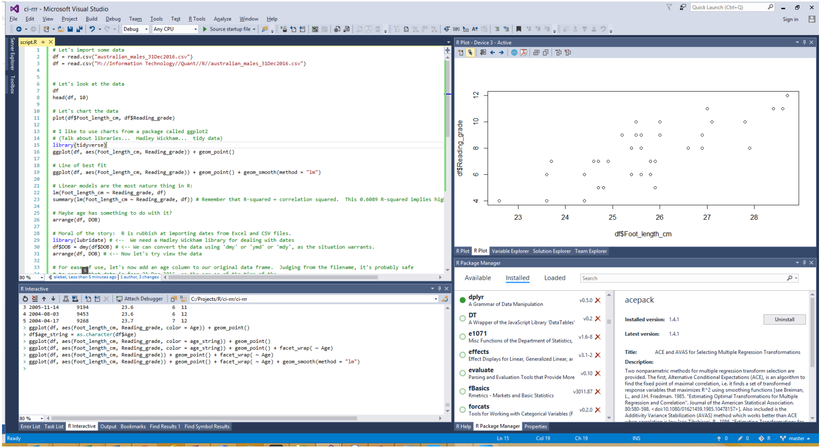
```
30 30-Jun-00      8909          26.9             9
31 26-Jun-99      9606          27.0            11
32 18-Feb-00     10291          27.1            10
33  5-Jul-01      9427          27.9             8
34 16-Jun-00      8753          27.8            10
35  6-May-00      8893          28.4            11
36 14-Oct-99      9537          28.6            11
37 11-Mar-99      9040          28.7            12
> head(df, 10)
         DOB StudentID Foot_length_cm Reading_grade
1  20-Jun-05      9204          22.6             4
2   6-Jun-05      9777          23.6             4
3  14-Nov-05      9194          23.6             6
4   3-Aug-04      9453          23.6             6
5  17-Apr-04      9268          23.7             7
6  24-Jul-05      9409          24.8             5
7   8-Mar-05      9724          24.8             5
8   4-Dec-04      9650          24.7             5
9   5-Dec-04      9339          24.4             4
10 12-Nov-04      8741          24.7             7
>
> # Let's chart the data
> plot(df$Foot_length_cm, df$Reading_grade)
> 
```

df   37 obs. of 4 variables

Figure 1: RStudio is the most popular IDE for working with R

Part I: The IDE

Visual Studio with R Tools



```
# Let's import some data
df = read.csv("australian_males_31Dec2016.csv")
df = read.csv("M://Information Technology//Quant//R//australian_males_31Dec2016.csv")


# Let's look at the data
df
head(df, 10)

# Let's chart the data
plot(df$Foot_length_cm, df$Reading_grade)

# I like to use charts from a package called ggplot2
# (Talk about libraries...  Hadley Wickham...  tidy data)
library(tidyverse)
ggplot(df, aes(Foot_length_cm, Reading_grade)) + geom_point()

# Line of best fit
ggplot(df, aes(Foot_length_cm, Reading_grade)) + geom_point() + geom_smooth(method = "lm")

# Linear models are the most nature thing in R:
lm(Foot_length_cm ~ Reading_grade, df)
summary(lm(Foot_length_cm ~ Reading_grade, df)) # Remember that R-squared = correlation squared.  This 0.6089 R-squared implies hig

# Maybe age has something to do with it?
arrange(df, DOB)

# Moral of the story:  R is rubbish at importing dates from Excel and CSV files.
library(lubridate) # <--  We need a Hadley Wickham library for dealing with dates
df$DOB = dmy(df$DOB) # <-- We can convert the data using 'dmy' or 'ymd' or 'mdy', as the situation warrants.
arrange(df, DOB) # <-- Now let's try view the data

# For ease of use, let's now add an age column to our original data frame.  Judging from the filename, it's probably safe
```

```
3 2005-11-14      9194          23.6          6  11
4 2004-08-03      9453          23.6          6  12
5 2004-04-17      9268          23.7          7  12
> ggplot(df, aes(Foot_length_cm, Reading_grade, color = Age)) + geom_point()
> df$age_string = as.character(df$Age)
> ggplot(df, aes(Foot_length_cm, Reading_grade, color = age_string)) + geom_point()
> ggplot(df, aes(Foot_length_cm, Reading_grade, color = age_string)) + geom_point() + facet_wrap( ~ Age)
> ggplot(df, aes(Foot_length_cm, Reading_grade)) + geom_point() + facet_wrap( ~ Age)
> ggplot(df, aes(Foot_length_cm, Reading_grade)) + geom_point() + facet_wrap( ~ Age) + geom_smooth(method = "lm")
>
```

● **dplyr** A Grammar of Data Manipulation v0.5.0

○ **DT** A Wrapper of the JavaScript Library 'DataTables' v0.2

○ **e1071** Misc Functions of the Department of Statistics, v1.6-8

○ **effects** Effect Displays for Linear, Generalized Linear, ar v3.1-2

○ **evaluate** Parsing and Evaluation Tools that Provide More v0.10

○ **fBasics** Rmetrics - Markets and Basic Statistics v3011.87

○ **forcats** Tools for Working with Categorical Variables (F v0.2.0

acepack

Installed version: 1.4.1 Uninstall

Latest version: 1.4.1

Title: ACE and AVAS for Selecting Multiple Regression Transformations

Description:

Two nonparametric methods for multiple regression transform selection are provided. The first, Alternative Conditional Expectations (ACE), is an algorithm to find the fixed point of maximal correlation, i.e. it finds a set of transformed response variables that maximizes R^2 using smoothing functions [see Breiman, L., and J.H. Friedman. 1985. "Estimating Optimal Transformations for Multiple Regression and Correlation". Journal of the American Statistical Association. 80:580-598. <doi:10.1080/01621459.1985.10478157>]. Also included is the Additivity Variance Stabilization (AVAS) method which works better than ACE

Figure 2: We can make VS look like RStudio

Part I: The IDE

Interactive R

Try

```
x = 2
x <- 3
4 -> x # note: nobody does this
y = sqrt(x)
```

and see how this affect *Variable Explorer* window.

Note: Can press up-arrow to repeat previous command

Part I: The IDE

Help

Try

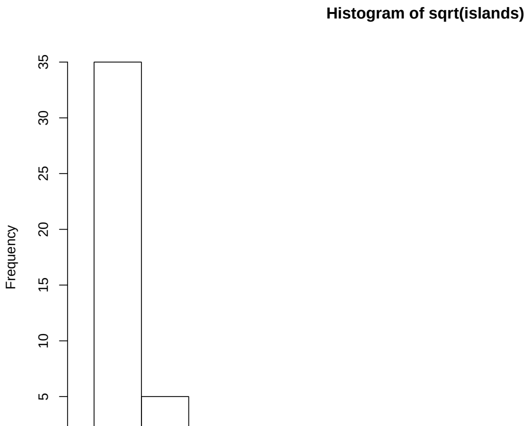
```
?sqrt
?lm
?hist
```

Part I: The IDE

Plots

Try the example from ?hist

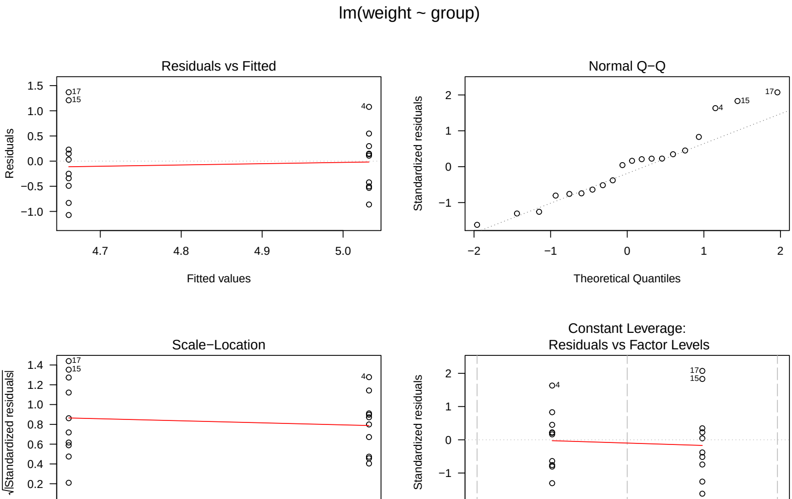
```r
hist(sqrt(islands), breaks = 12)
```



Part I: The IDE

Help examples

```r
example("lm")
```



Part II: Data Structures

Vectors

```r
v = c(10, 11, 12, 13, 14)
v # on its own should return the result
v[2]
v[-2]
v[3]
v[c(2, 3)]
v[ - c(2, 3)]
v == 12
v > 12
v %% 2
v %% 2 == 0
v - 10
which(v > 2)
v[which(v > 2)]
```

Part II: Data Structures

Vectors using seq() function and :

```r
y <- 1:4
pi:10
10:pi
seq(from = 2, to = 8, by = 2)
seq(from = 0, to = 1, length = 11)
```

Part II: Data Structures

Vectors using rep()

```r
rep(x = c(1, 2, 3, 4), times = 2)
rep(x = c(1, 2, 3, 4), each = 2)
rep(x = 1:4, each = 4)
rep(x = 1:4, times = 10)
rep(x = 1:4, times = 1:4)
```

Part II: Data Structures

Vectors arithmetic

```
c(1, 2, 3, 4) / 2
c(1, 2, 3, 4) / c(4, 3, 2, 1)
c(2, 4, 6, 8) / c(2, 3)
```

Part II: Data Structures

Vectors functions to try

```
length(v)
sum(v)
mean(v) # Also try median(v), prod(v), cumsum(v),
cumprod(v) # Can use cumprod(1 + r) to compute an index
#            from a return time series r.
names(v) = c("one", "two", "three", "four", "five")
v  # v is now indexed:  try v["one"]
c(v,v)
unique(c(v,v))
sort(c(v,v))
v %in% c(10, 12)
```

Part II: Data Structures

Lists

```
mylist <- list(c("Fred", "Bill", "Martha"),
               age = c(23,29,42,50))

mylist[[2]]
mylist[["age"]]
mylist$age # this should have an autocomplete
```

Part II: Data Structures

Matrices

```r
cells <- c(1,10,26,24,68,45)
rnames <- c("R1", "R2")
cnames <- c("C1", "C2", "C3")
mymatrix <- matrix(cells, nrow =2, ncol = 3,
       byrow=TRUE, dimnames=list(rnames, cnames))
```

I have yet to use these, but rest assured that matrix algebra works in R... Will be useful as we do more advanced portfolio optimisation work...

Part II: Data Structures

Dataframes

```r
data()
data(iris)
summary(iris)  # use Variable Explorer to view
```

Part III: Data Science Life cycle

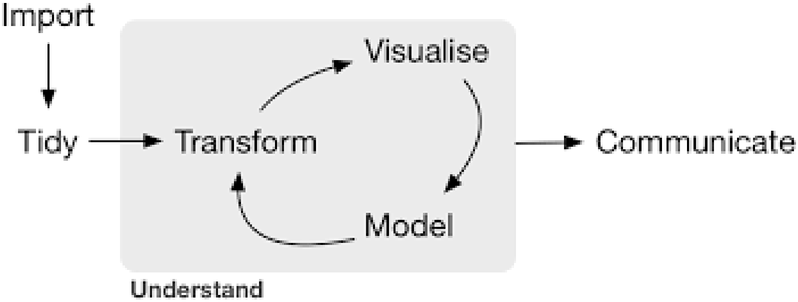


Figure 3: Data Science Workflow, by Hadley Wickham

Part IV: R at CI

Clone ci-factor_model from my GitHub account:

`https://github.com/lebelinoz/ci-factor_model`