http://courses.had.co.nz

R language & ecosystem

Hadley Wickham

Assistant Professor / Dobelman Family Junior Chair
Department of Statistics / Rice University

July 2012

What is R?

- R is a programming language
- R is statistical software
- R is an environment for interactive data analysis
- R is a community

http://www.inside-r.org/what-is-r

Programming language

```r
wtd.mean <- function(x, wt = rep(1, length(x))) {
  sum(x * wt) / sum(wt)
}

wtd.mean(1:10)
wtd.mean(1:10, 10:1)
```

What does this function return?

```
x <- 5
f <- function() {
  y <- 10
  c(x = x, y = y)
}
f()
```

What does this function return?

```
x <- 5
g <- function() {
  x <- 20
  y <- 10
  c(x = x, y = y)
}
g()
```

What does this function return?

```
x <- 5
h <- function() {
 y <- 10
 i <- function() {
  z <- 20
  c(x = x, y = y, z = z)
 }
 i()
}
h()
```

```
j <- function() {
  if (!exists(”a”)) {
    a <- 5
  } else {
    a <- a + 1
  }
  print(a)
}
```

What does this function return the first time you run it? The second time?

```
x <- 0
y <- 10
k <- function() {
  x <- 1
  function() {
    y <- 2
    x + y
  }
}

# What does k() return?
# What does k()() mean? What does it do?
# How does it work?
```

Functional heritage

First class functions & lexical scoping

Lazy evaluation of function arguments

Copy-on-modify = immutable objects + mutable bindings

OO based on generic functions

```
# First class functions and lexical scoping

slow_down <- function(f, seconds = 1) {
   function(...) {
     Sys.sleep(seconds)
     f(...)
   }
}

runif(1)
slow_runif <- slow_down(runif, 1)
slow_runif(1)
```

```
# Lazy evaluation of function arguments

add <- function(a, b, z) {
  a + b
}
add(10, 20, slow_down(runif, 1)(10))

# Most languages have this for boolean operators
# aka short circuiting

# Also allows syntactic manipulation:
add <- function(a, b) {
  cat(paste(”Adding”, deparse(substitute(b)), ”to”,
    deparse(substitute(a))), ”\n”)
  a + b
}
x <- 10
add(x, 15)
```

```r
# Immutable objects + mutable bindings =
# copy on modify

a <- list(a = 6, b = 10, c = 7)
b <- a

a$a <- 10
a$a
b$a

# Behind the scenes, any modification is implemented
# as the creation of a modified copy.  Above code
# translates to:
a <- modifyList(a, list(a = 10))
```

```
# There are a number of optimisations to reduce
# this copying. tracemem() helps to discover them
# (they have been increasing in recent versions)

x <- 1:10
tracemem(x)
x[5] <- 5L
x[11] <- 11L # Doesn’t count
x[5] <- 5    # Beware
attr(x, ”a”) <- 10
names(x) <- letters[1:10]

y <- as.list(x)
tracemem(y)
y$a <- 10
```

OO programming

Three OO systems: S3, S4, R5

S3 = ad hoc, single dispatch, naming conventions

S4 = formal & strict, multiple dispatch, based on CLOO/Dylan

Both generic function style, not message passing (**methods belong to functions, not classes**)

R5 = reference classes behave like classes from python, ruby, java etc

```
# Generic functions: specialise behaviour of a function,
# not of an object

mean <- function (x, ...) {
  UseMethod(”mean”, x)
}
mean.numeric <- function(x, ...) {
  sum(x) / length(x)
}
mean.data.frame <- function(x, ...) {
  sapply(x, mean, ...)
}
mean.matrix <- function(x, ...) {
  apply(x, 2, mean)
}
```

```
# No checks for object correctness, so easy to abuse

mod <- glm(log(mpg) ~ log(disp), data = mtcars)
class(mod)
class(mod) <- ”lm”
mod

class(mod) <- ”table”
mod

# But surprisingly, this doesn’t cause that
# many problems - instead of the language enforcing
# certain properties you need to do it yourself
```

Statistical software

Special features

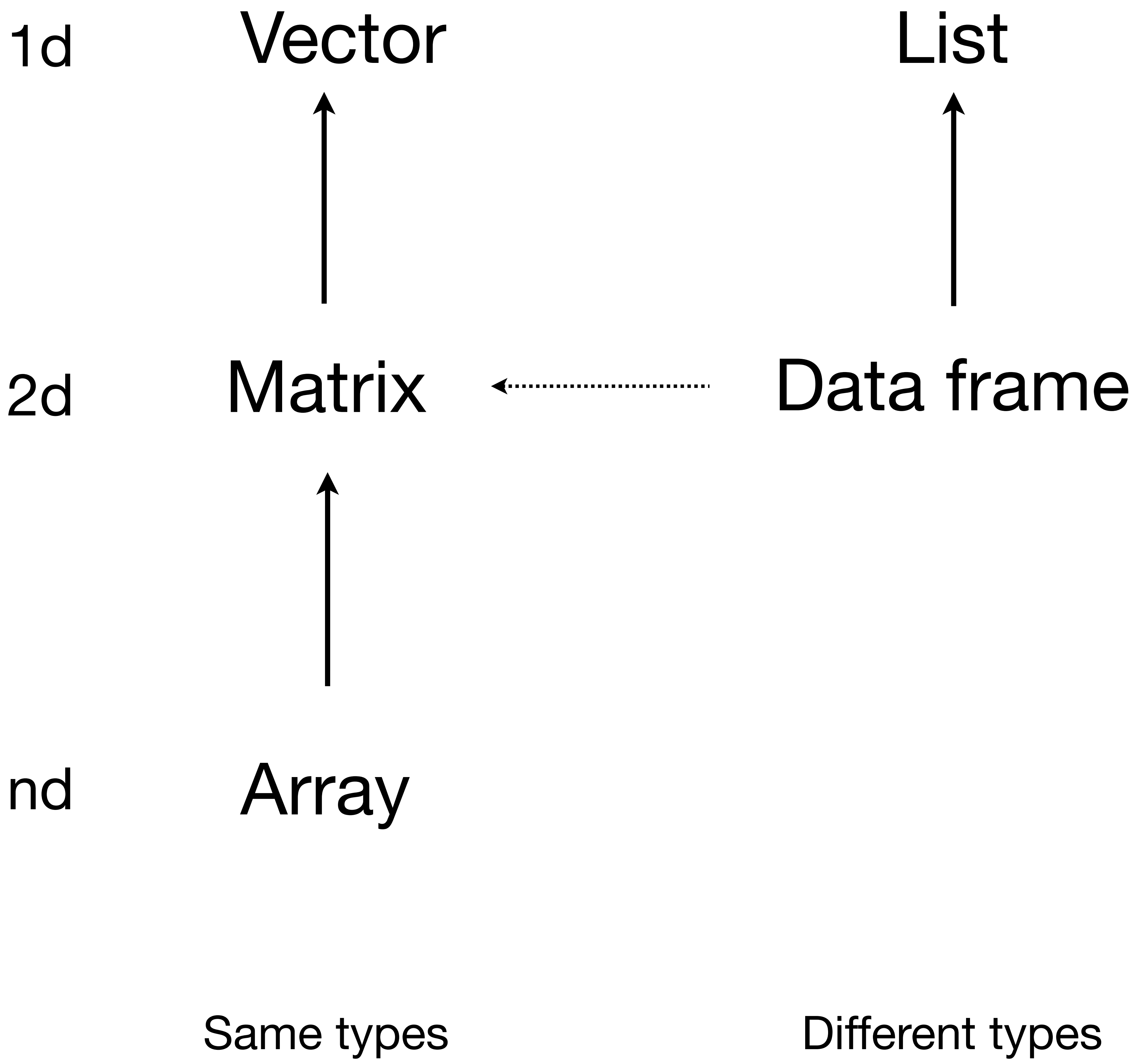
Vectorised computation

Data frames

Powerful indexing

Missing values

```r
# Vectorised computations
# Already seen an example in weighted.mean

sum(1:10 * 2) / sum(1:10)
1:10 * 10:1
1:10 * 2

# recycling expands length of shorter argument to
# length of longer (without warning if integer
# multiple - beware!)
# Data frames

library(ggplot2)
head(diamonds)
str(diamonds)

# A rectangular structure
# Each column has same type, but different
# columns may have different types
```



1d Vector List

2d Matrix Data frame

nd Array

Same types Different types

```
str()
```

```
# Indexing
# Mastering indexing/subsetting is critical for
# efficient R programming

diamonds[1:5, ]
diamonds[diamonds$x == diamonds$y, ]
diamonds[-(1:53900), c(”carat”, ”price”)]
```

Expression	Guess	Actual
`5 + NA`		
`NA / 2`		
`sum(c(5, NA))`		
`mean(c(5, NA)`		
`NA < 3`		
`NA == 3`		
`NA == NA`		

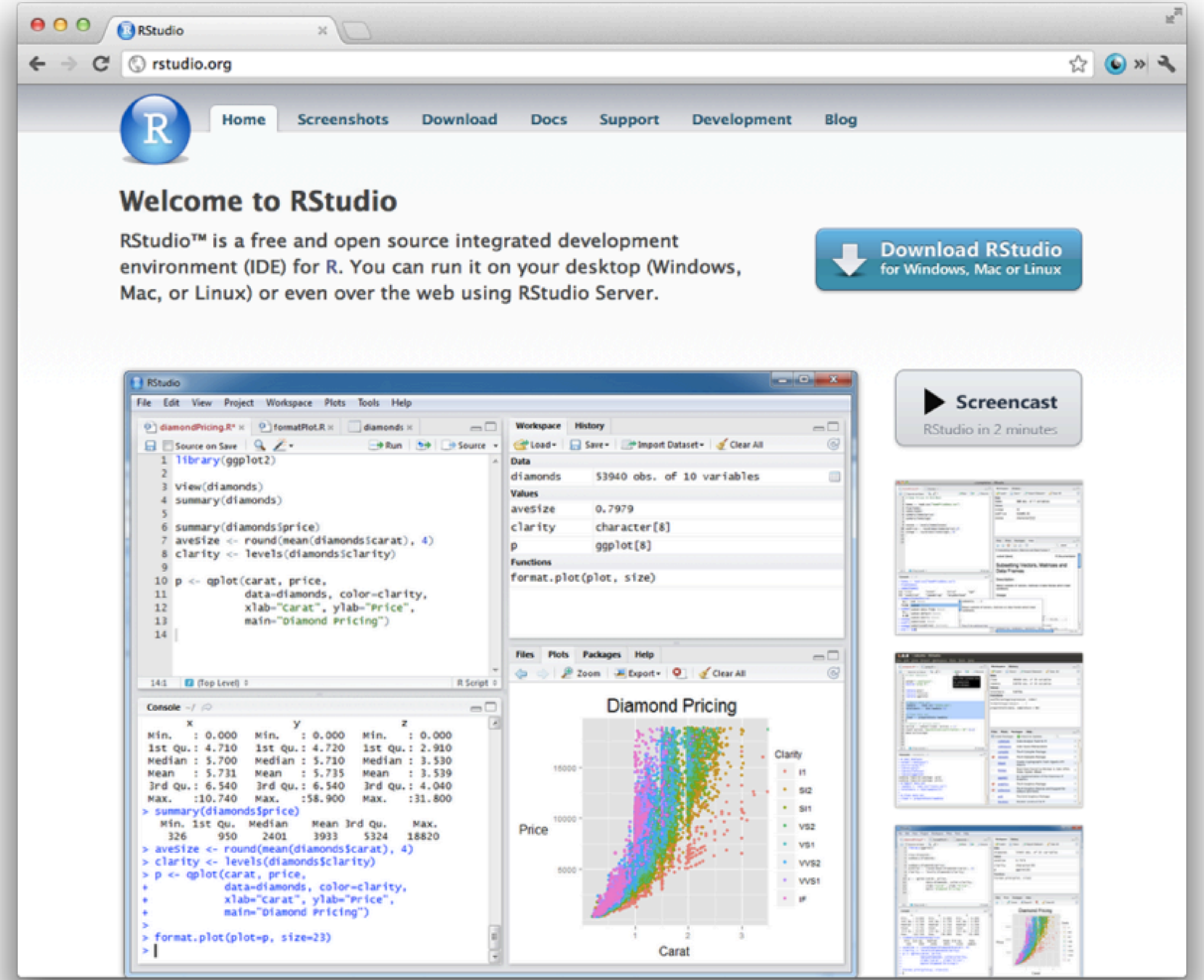
```
# Missing values: ternary logic, like SQL

NA == NA
# Is NA!

is.na(NA)
# Use is.na to check

# Default is to propagate missing values. Many
# functions have na.rm argument to remove them
```

Vocab

http://github.com/devtools/wiki/Vocabulary

Interactive environment

```
# Comprehensive built in help
?mean

# Heuristics minimise output when you don’t want
# to see it
a <- 10
a

(a <- 15)

# Use source to load complete files
```



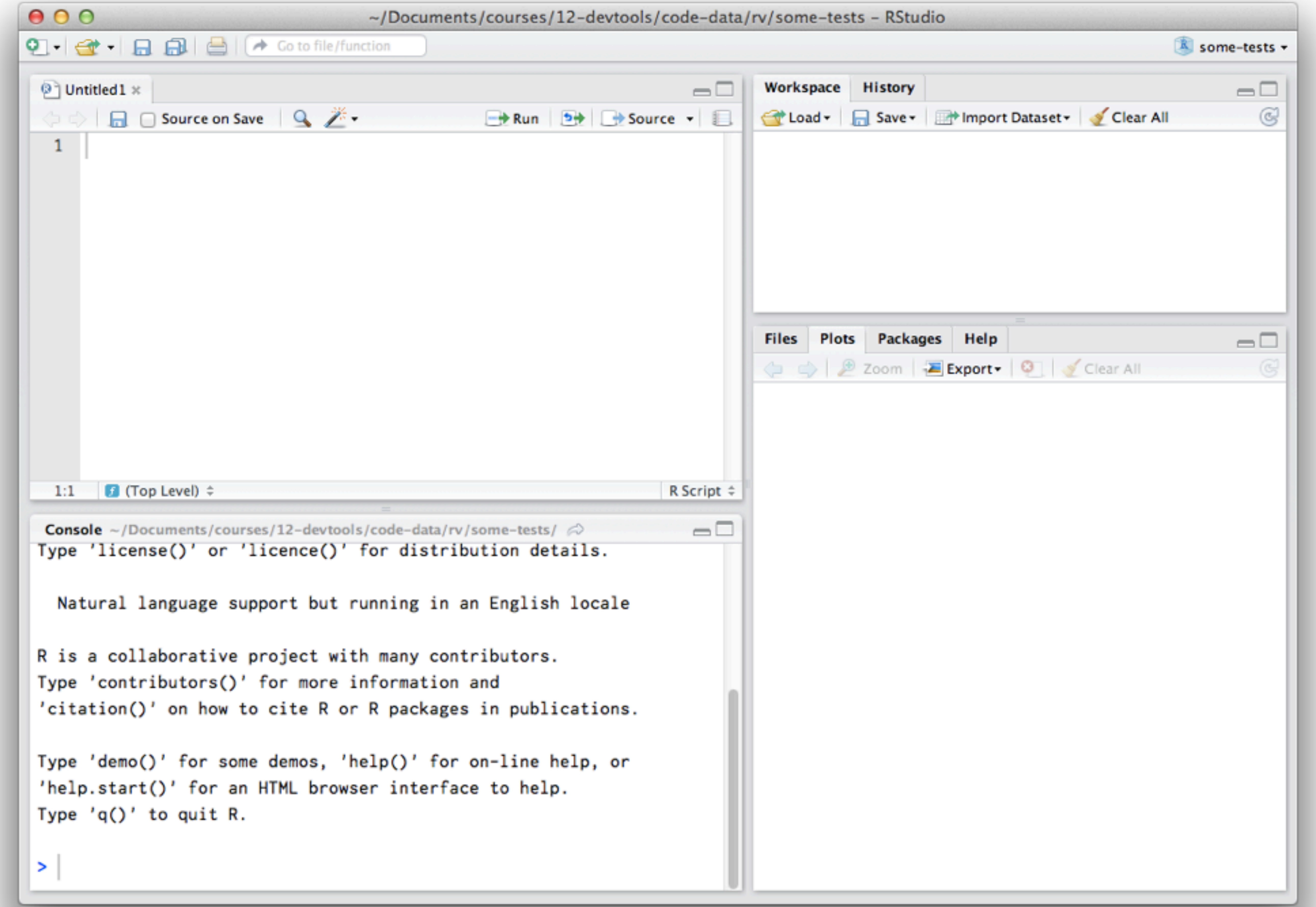
Welcome to RStudio

RStudio™ is a free and open source integrated development environment (IDE) for R. You can run it on your desktop (Windows, Mac, or Linux) or even over the web using RStudio Server.

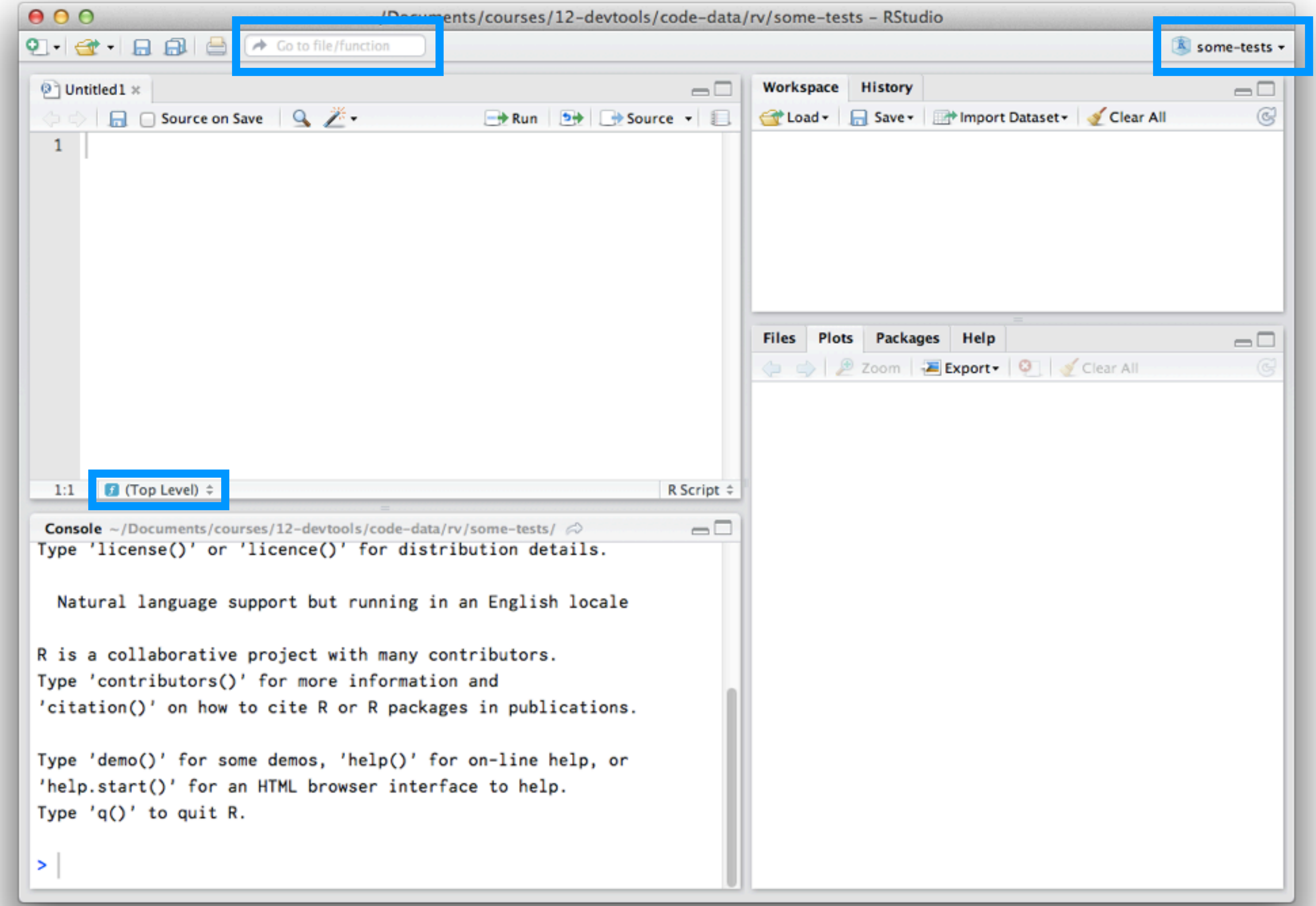
```
library(ggplot2)

View(diamonds)
summary(diamonds)

summary(diamonds$price)
aveSize <- round(mean(diamonds$carat), 4)
clarity <- levels(diamonds$clarity)

p <- qplot(carat, price,
           data=diamonds, color=clarity,
           xlab="Carat", ylab="Price",
           main="Diamond Pricing")
```

Data	
diamonds	53940 obs. of 10 variables
Values	
aveSize	0.7979
clarity	character[8]
p	ggplot[8]
Functions	
format.plot(plot, size)	

```
       x                y                z
 Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
 1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.910
 Median : 5.700   Median : 5.710   Median : 3.530
 Mean   : 5.731   Mean   : 5.735   Mean   : 3.539
 3rd Qu.: 6.540   3rd Qu.: 6.540   3rd Qu.: 4.040
 Max.   :10.740   Max.   :58.900   Max.   :31.800
> summary(diamonds$price)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    326     950    2401    3933    5324   18820
> aveSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+            data=diamonds, color=clarity,
+            xlab="Carat", ylab="Price",
+            main="Diamond Pricing")
+
> format.plot(plot=p, size=23)
> 
```



```
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.

> 
```



```
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.

>
```

Short cuts

In editor:

Command/ctrl + enter: send code to console

Ctrl + 2: move cursor to console

In console:

Up arrow: retrieve previous command

Ctrl + up arrow: search commands

Ctrl + 1: move cursor to editor

http://rstudio.org/docs/using/keyboard_shortcuts

Community

Community

Over 4,000 add on packages available from the community.

Finding the package you need can be hard: CRAN task views, http://rseek.org/, http://crantastic.org.

R-help mailing list can be prickly. Stackoverflow strong (http://stackoverflow.com/questions/tagged/r). #rstats on twitter.

Journals

The R Journal, *http://journal.r-project.org/*

The Journal of Statistical Software, *http://www.jstatsoft.org/*

Statistical computing and graphics newsletter, *http://stat-computing.org/newsletter/*