Outline

1. **Introduction to R and RStudio**
2. **Coding in R**
   - Simple calculations
   - Vectors
   - Data Frames
3. **Live R Session**
4. **Additional functionality and examples**
   - Help
   - Basic Operations and Functions

the tools...

R
www.r-project.org

The engine

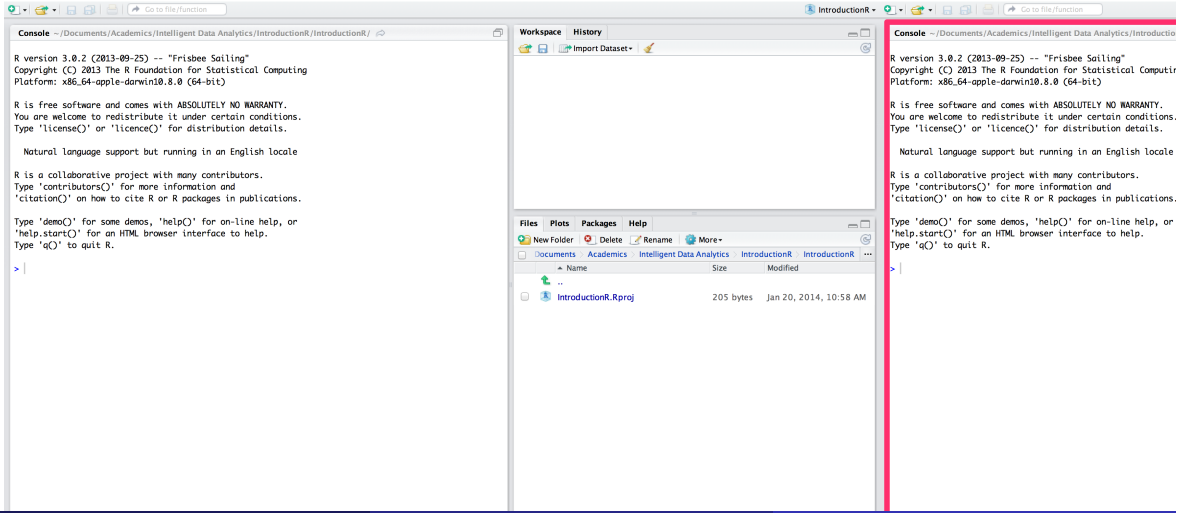
RStudio
www.rstudio.org

The pretty face

introduction to RStudio

RStudio is designed to support the iterative nature of R and data analysis in general.

RStudio


Go to file/function
IntroductionR
Console ~/Documents/Academics/Intelligent Data Analytics/IntroductionR/IntroductionR/
R version 3.0.2 (2013-09-25) -- "Frisbee Sailing"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin10.8.0 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
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.
Workspace
History
Import Dataset
Files
Plots
Packages
Help
New Folder
Delete
Rename
More
Documents
Academics
Intelligent Data Analytics
IntroductionR
IntroductionR
Name
Size
Modified
..
IntroductionR.Rproj
205 bytes
Jan 20, 2014, 10:58 AM

some advantages of RStudio

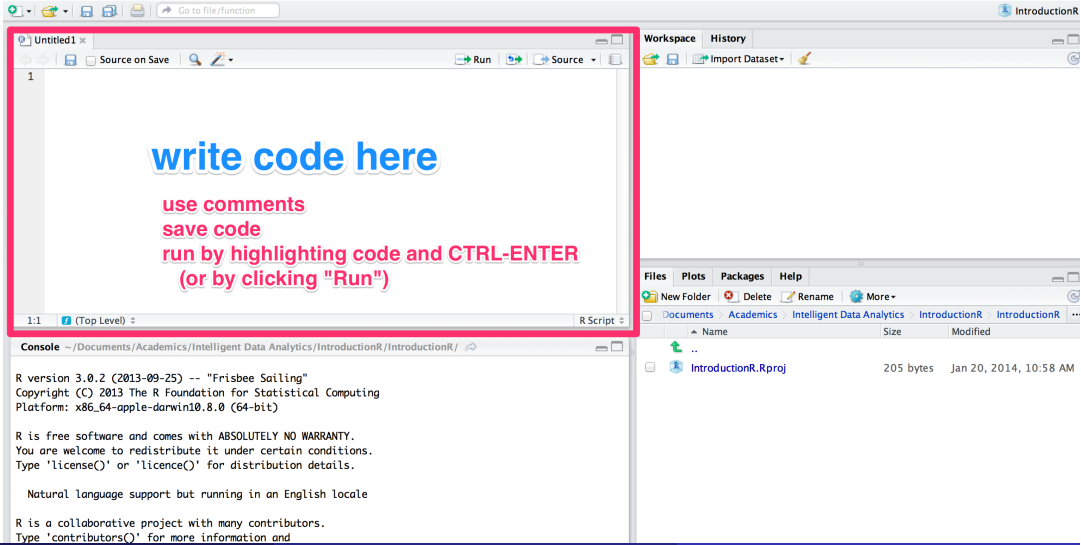
1. **Easy running and managing scripts**
   ctrl+enter runs selected code segment
2. **Easy access to help**
3. **Code completion**
   $< tab >$ autocompletes object names within the workspace.
4. **Command history view, search**
   - In "History" window, can search and filter
   - In console, Up and Down arrows cycle through recent commands; Ctrl-Up gives a list of recent commands for selection.
5. **Importing & exporting files**
6. **Project workspace management** (scripts, history, objects, data files)

some advantages of RStudio

Workspaces and directories

- Convenient to create, save, access **projects** in RStudio
- Convenient switching between multiple projects in R, each of which might be associated with several R code files and be in different locations.

RStudio: basic workflow



Outline

1. Introduction to R and RStudio
2. Coding in R
   - Simple calculations
   - Vectors
   - Data Frames
3. Live R Session
4. Additional functionality and examples
   - Help
   - Basic Operations and Functions

R as a calculator

Console Window

```
> 1 + 1
[1] 2

> (1 + 1) * 7
[1] 14

> log2(32)
[1] 5
```

```
> 9^3
[1] 729

> sqrt(2)
[1] 1.414214

> seq(0, 5, length=6)
[1] 0 1 2 3 4 5
```

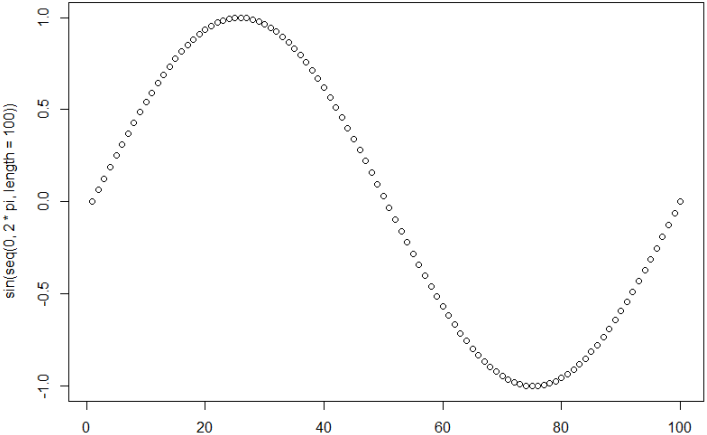
R as a calculator

Console Window

```
> seq(0, 5, length=15)
 [1] 0.0000000 0.3571429 0.7142857 1.0714286
 [5] 1.4285714 1.7857143 2.1428571 2.5000000
 [9] 2.8571429 3.2142857 3.5714286 3.9285714
[13] 4.2857143 4.6428571 5.0000000
```

R as a calculator

```
> plot(sin(seq(0, 2*pi, length=100)))
```



assigning variables

```
> x <- 1
> x
[1] 1

> y = 2
> y
[1] 2
```

- Use = or <− for assignment
- There are no scalars in R

assigning variables

```
> (x + y) * 3
[1] 9

> a <- (1+1==3)
> a
[1] FALSE
```

- Atomic data types in R:
  - numeric (integer, double, complex)
  - character
  - logical
  - function
- Can build vectors, matrices, arrays, lists, data frames, etc.

quick note

Naming Convention: Object Names

- Must start with a letter (A-Z or a-z)
- May contain alphanumerics, underscores, and/or periods
- Case-senstive: `mydata` is different from `MyData`

vectors, matrices and arrays

Definitions

vector: an ordered collection of data of the same type

matrix: a rectangular table of data of the same type

array: 3-,4-,n- . . . dimensional matrix

list: an ordered collection of data of arbitrary types

vectors

```
> x <- c(1.1,-2,5.7)
> x
[1]  1.1 -2.0 5.7

> y <- 1:5
> y
[1] 1 2 3 4 5

> length(x)
[1] 3
```

- `c` is a function to *combine* values into a list or vector
- Operator "`:`" implies a *series of integers between*
- `length` returns the length of a vector

math with vectors

```
> x <- c(0,1,2,3,4)
> y <- 1:5
> z <- 1:50
```

```
> x + y
[1] 1 3 5 7 9

> x * y
[1] 0 2 6 12 20

> x %*% y
     [,1]
[1,]   40
```

math with vectors

```
> x <- c(0,1,2,3,4)
> z <- 1:50
```

```
> x * z
```

```
 [1] 0 2 6 12 20 0 7 16 27 40 0
[12] 12 26 42 60 0 17 36 57 80 0 22
[23] 46 72 100 0 27 56 87 120 0 32 66
[34] 102 140 0 37 76 117 160 0 42 86 132
[45] 180 0 47 96 147 200
```

math with vectors

Typical operations on vectors include summary statistics (mean, var, sd, range, max, mad, . . .)

```
> y <- c(5,7,7,8,2,5,6,6,7,5,8,3,4)
```

```
> mean(y)
[1] 5.615385
```

```
> sd(y)
[1] 1.850156
```

```
> var(y)
[1] 3.423077
```

```
> mad(y)
[1] 1.4826
```

logical vectors

```
> y <- c(0,7,8,2,5,1)
> y
[1] 0 7 8 2 5 1

> y<6
[1] TRUE FALSE FALSE TRUE TRUE TRUE

> z <- y<6
> z
[1] TRUE FALSE FALSE TRUE TRUE TRUE
```

subscripts: obtaining parts of vectors

- Access elements of vectors by subscripts with `[]`:

```
> y[3]
```

- e.g. access third thru seventh elements of y:

```
> y[3:7]
```

- or, third, fifth, and ninth elements:

```
> y[c(3,5,9)]
```

- All elements *except for* the third:

```
> y[-3]
```

- Change an element of y:

```
y[3] <- 473
```

subscripts as logical variables

```
> y <- c(9,7,8,2,5,1)
```

- Logical condition to find a subset of the values in a vector:

```
> y[y<6]
[1] 2 5 1
```

- To know the values of z for which y>6:

```
> z[y>6]
```

- Element of y not multiples of three:

```
> y[y%%3 != 0]
```

data frames

Data frame

a list of vectors of equal length

- Data frames are used for storing data tables, very similar to a spreadsheet.
- Most of the time, when data is loaded, it will be organized in a data frame.

data frames

Example. **df** is a data frame containing three vectors n, s, b:

```
> n = c(2, 3, 5)
> s = c("aa", "bb", "cc")
> b = c(TRUE, FALSE, TRUE)
> df = data.frame(n, s, b)

  n  s     b
1 2 aa  TRUE
2 3 bb FALSE
3 5 cc  TRUE
```

accessing elements of a dataframe

df[row,column]

```
  n  s     b
1 2 aa  TRUE
2 3 bb FALSE
3 5 cc  TRUE
```

- ...the 1st row and 3rd column:

```
> df[1,3]
[1] TRUE
```

- ...the 3rd row of the 1st column:

```
> df[3,1]
[1] 5
```

- ...the 2nd row and 2nd column:

```
> df[2,2]
[1] bb
```

accessing elements of a dataframe

A better example

age	gender	disease
50	M	TRUE
43	M	FALSE
25	F	TRUE
18	M	TRUE
72	F	FALSE
65	M	TRUE

accessing elements of a dataframe

Accessing a complete column, e.g. to access the “age” variable:

```
> df$age
> df[,"age"]
> df[,1]

[1] 50 43 25 18 72 65
```

age	gender	disease
50	M	TRUE
43	M	FALSE
25	F	TRUE
18	M	TRUE
72	F	FALSE
65	M	TRUE

accessing elements of a dataframe

Accessing a full row:

```
> df[1,]

  age gender disease
1  50      M    TRUE
```

age	gender	disease
50	M	TRUE
43	M	FALSE
25	F	TRUE
18	M	TRUE
72	F	FALSE
65	M	TRUE

accessing elements of a dataframe

Accessing multiple rows:

```
> df[c(3,5),]

  age gender disease
3  25      F    TRUE
5  72      F   FALSE
```

age	gender	disease
50	M	TRUE
43	M	FALSE
25	F	TRUE
18	M	TRUE
72	F	FALSE
65	M	TRUE

accessing elements of a dataframe

Select all female records:

```
> df[df$gender=='F',]

  age gender disease
3  25      F    TRUE
5  72      F   FALSE
```

age	gender	disease
50	M	TRUE
43	M	FALSE
25	F	TRUE
18	M	TRUE
72	F	FALSE
65	M	TRUE

Outline

1. Introduction to R and RStudio
2. Coding in R
   - Simple calculations
   - Vectors
   - Data Frames
3. **Live R Session**
4. Additional functionality and examples
   - Help
   - Basic Operations and Functions

The R script from the coding session is available on the course website.

Outline

1. Introduction to R and RStudio
2. Coding in R
   - Simple calculations
   - Vectors
   - Data Frames
3. Live R Session
4. **Additional functionality and examples**
   - Help
   - Basic Operations and Functions

getting help

`?` is a function which returns documentation on functions
`??` is shorthand for searching the installed package documentation for a string

For example,

`?t.test` provides documentation from the `stats` package
`??t.test` lists all documentation from installed packages that contain the string “t.test”

`RSiteSearch("t.test")` searches the full R site documentation

```
?t.test
```

Environment History

Files Plots Packages Help Viewer

R: Student's t-Test sub

t.test {stats}

Student's t-Test

Description

Performs one and two sample t-tests on vectors of data.

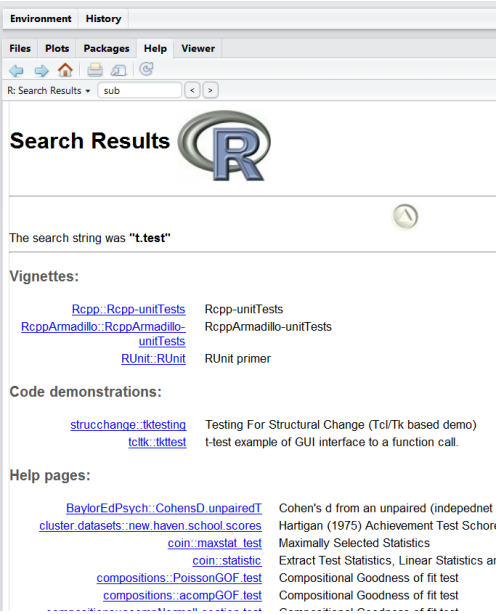
Usage

```
t.test(x, ...)

## Default S3 method:
t.test(x, y = NULL,
       alternative = c("two.sided", "less", "greater"),
       mu = 0, paired = FALSE, var.equal = FALSE,
       conf.level = 0.95, ...)

## S3 method for class 'formula'
t.test(formula, data, subset, na.action, ...)
```

Arguments

`x`	a (non-empty) numeric vector of data values.
`y`	an optional (non-empty) numeric vector of data values.
`alternative`	a character string specifying the alternative hypothesis, must be one of `"two.sided"` specify just the initial letter.
`mu`	a number indicating the true value of the mean (or difference in means if you are perf
`paired`	a logical indicating whether you want a paired t-test

```
??t.test
```


Environment
History
Files
Plots
Packages
Help
Viewer
R: Search Results
sub
Search Results
The search string was "t.test"
Vignettes:
Rcpp::Rcpp-unitTests
Rcpp-unitTests
RcppArmadillo::RcppArmadillo-unitTests
RcppArmadillo-unitTests
RUnit::RUnit
RUnit primer
Code demonstrations:
strucchange::tktesting
Testing For Structural Change (Tcl/Tk based demo)
tcltk::tkttest
t-test example of GUI interface to a function call.
Help pages:
BaylorEdPsych::CohensD.unpairedT
Cohen's d from an unpaired (indepednet
cluster.datasets::new.haven.school.scores
Hartigan (1975) Achievement Test Schor
coin::maxstat_test
Maximally Selected Statistics
coin::statistic
Extract Test Statistics, Linear Statistics ar
compositions::PoissonGOF.test
Compositional Goodness of fit test
compositions::acompGOF.test
Compositional Goodness of fit test

```
RSiteSearch("t.test")
```

arithmetic

`^` or `**`	exponentiation (e.g. 2**3 = 8)
`x%%y`	modulus (e.g. 5 %% 2 = 1)
`x%/%y`	integer division (e.g. 5%/%2 = 2)
`abs()`	absolute value
`ceiling()`	e.g. `ceiling(3.14) = 4`
`floor()`	`floor(3.14)=3`
`round()`	round to a specific number of digits, e.g. `round(3.14156,3)` = 3.142
`exp()`	`exp(x)` = e^x

example basic statistical functions

var()	variance
table()	creates a contingency table
range()	returns the largest and smallest values
summary()	calculates descriptive statistics
cor() and cor.test()	pearson, spearman, kendall-tau correlations
t.test()	t tests
chisq.test()	chi squared test
aov()	ANOVA

some utility functions

sample()	sample with or without replacement
rev()	reverses the order of elements
order()	sorts a data frame
unique()	removes duplicate elements
seq()	creates a sequence of numbers
rep()	replicates elements
cut()	divide continuous value into n-levels
quantile()	produces quantiles
sub()	substitutes character strings
paste()	to concatenate strings

sequences

Create sequences of values using the `seq()` function:

```
> seq(0,5,length=7)
[1] 0.000 0.833 1.667 2.500 3.333 4.167 5.000
```

basic sequence

```
> seq(2,5,0.5)
[1] 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

use start, stop, and stepsize

```
> seq(0.5,0,-0.1)
[1] 0.5 0.4 0.3 0.2 0.1 0.0
```

negative step values for decreasing values

replication

The `rep()` function replicates the first argument by the number of times specified in the second argument:

```
> rep("A",10)
[1] "A" "A" "A" "A" "A" "A" "A" "A" "A" "A"
```

Repeated series:

```
> rep(1:6,2)
[1] 1 2 3 4 5 6 1 2 3 4 5 6
```

To specify each repeat separately:

```
> rep(c(4,7,1,5),c(3,2,5,2))
[1] 4 4 4 7 7 1 1 1 1 1 5 5
```

character strings

Substitute strings with `sub()`

```
> a = "The dog ate my homework"
> sub("dog","cat",a)
[1] "The cat ate my homework"
```

Concatenate strings with `paste()`

```
> fname = "Joe"; lname ="Smith"
> paste(fname, lname)
[1] "Joe Smith"
```

sorting a data Frame

To sort a dataframe in R, use the `order()` function.
By default, sorting is ASCENDING.

Prepend the sorting variable by a minus sign to indicate DESCENDING order.

```
#sorting examples using mtcars dataset
data(mtcars)

#sort by mpg
newdata = mtcars[order(mtcars$mpg),]

#sort by mpg and cyl
newdata <- mtcars[order(mtcars$mpg, mtcars$cyl),]

#sort by mpg (ascending) and cyl (descending)
newdata <- mtcars[order(mtcars$mpg, -mtcars$cyl),]
```

creating new variables

Examples

Create new variable as function of other variables, e.g. a simple average:

```
df$avgValue <- (df$x1 + df$x2)/2
```

Or: create age categories using a logical condition:

```
df$agecat[df$age > 70] <- "Old"
df$agecat[df$age > 30 & df$age <= 70] <- "Middle"
df$agecat[df$age <= 30] <- "Young"
```

more complex variable creation

In this example, create a categorical variable based on the decile breaks of the `age` variable.

```
# determine the quartile points using quantile()
x<-quantile(df$age)

# use "x" and the cut() function together
df$ageBins <- cut(df$age, breaks = x,
                  include.lowest = T)
```