A short R refresher

Thibaut Jombart, Marie-Pauline Beugin

MRC Centre for Outbreak Analysis and Modelling
Imperial College London

Genetic data analysis with R
PR∼Statistics, Millport Field Station
16 Aug 2016

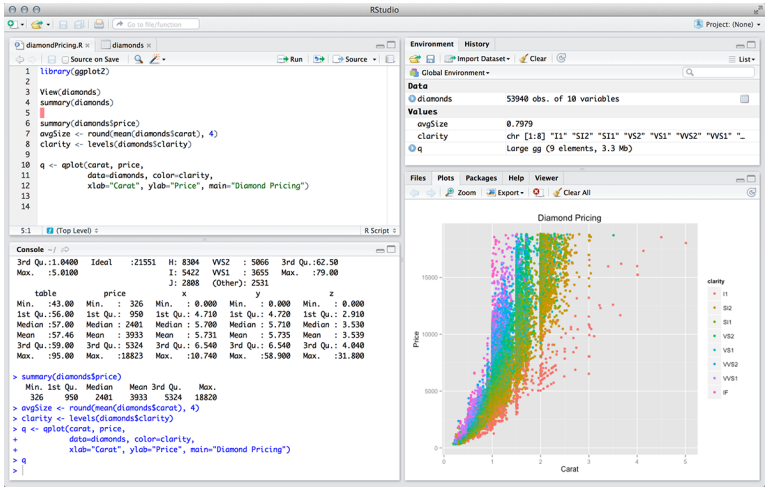
Installing R

Get latest version from: `https://www.r-project.org/`

We are using 3.3.1. To check your version:

```
> R.version.string

[1] "R version 3.3.1 (2016-06-21)"
```

Using R



Editor / GUI:

- Rstudio: `https://www.rstudio.com/`
- Emacs: `http://www.gnu.org/software/emacs/`
- Tinn-R: `https://sourceforge.net/projects/tinn-r/`
- ...

Installing development versions of packages

1. on Window$, install *Rtools*: `https://cran.r-project.org/bin/windows/Rtools/`
2. install the package *devtools*
3. install packages using `install_github`

For instance:

```
> devtools::install_github("thibautjombart/adegenet")
```

will install the current devel version of *adegenet*

Installing development versions of packages

1. on Window$, install *Rtools*: `https://cran.r-project.org/bin/windows/Rtools/`
2. install the package *devtools*
3. install packages using `install_github`

For instance:

```
> devtools::install_github("thibautjombart/adegenet")
```

will install the current devel version of *adegenet*

When you know the name of the beast

What does `t.test` do? How does it work?

```
> ?t.test
```

```
t.test                  package:stats                  R Documentation

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"),
...
```

When the beast is anonymous

You do not know the function's name, only some keywords; for instance:

```
> ??"Hardy Weinberg"
```

Package	Topic	Title
adegenet	HWE.test.genind	Hardy-Weinberg Equilibrium test for multilocus data
pegas	hw.test	Test of Hardy-Weinberg Equilibrium

```
> help.search("Hardy Weinberg")
```

Package	Topic	Title
adegenet	HWE.test.genind	Hardy-Weinberg Equilibrium test for multilocus data
pegas	hw.test	Test of Hardy-Weinberg Equilibrium

Other searching options

- `RSiteSearch()`: looks through manuals and ML archives
- `example()`: runs the example in the manual page (unless `dontrun`'ed)
- `apropos()`: when part of the name is known

```
> apropos("hw")

[1] "HWE.test.genind" "hw.test"
```

What does the beast eat?

What arguments does it accept? How do I specify them?

To list the arguments of a function:

```
> library(pegas)
> args(hw.test)

function (x, B = 1000, ...)
NULL
```

But even better (!):

```
> ?hw.test
```

In case of trouble

1. read the documentation
2. read the documentation
3. read the documentation

In case of trouble

1. read the documentation
2. read the documentation
3. read the documentation

In case of trouble

1. read the documentation
2. read the documentation
3. read the documentation

In case of trouble

1. read the documentation
2. read the documentation
3. read the documentation

If this is not enough, ask someone.

Where can you ask questions?

- general questions: R-help
  `http://stat.ethz.ch/mailman/listinfo/r-help`
- phylogenetics: R-sig-phylo
  `https://stat.ethz.ch/mailman/listinfo/r-sig-phylo`
- genetics: R-sig-phylo
  `https://stat.ethz.ch/mailman/listinfo/r-sig-genetics`
- *adegenet*: the adegenet forum
  `http://lists.r-forge.r-project.org/mailman/listinfo/adegenet-forum`

Where can you ask questions?

- general questions: R-help
  `http://stat.ethz.ch/mailman/listinfo/r-help`
- phylogenetics: R-sig-phylo
  `https://stat.ethz.ch/mailman/listinfo/r-sig-phylo`
- genetics: R-sig-phylo
  `https://stat.ethz.ch/mailman/listinfo/r-sig-genetics`
- *adegenet*: the adegenet forum
  `http://lists.r-forge.r-project.org/mailman/listinfo/adegenet-forum`

Where can you ask questions?

- general questions: R-help
  `http://stat.ethz.ch/mailman/listinfo/r-help`
- phylogenetics: R-sig-phylo
  `https://stat.ethz.ch/mailman/listinfo/r-sig-phylo`
- genetics: R-sig-phylo
  `https://stat.ethz.ch/mailman/listinfo/r-sig-genetics`
- *adegenet*: the adegenet forum
  `http://lists.r-forge.r-project.org/mailman/listinfo/adegenet-forum`

Where can you ask questions?

- general questions: R-help
  `http://stat.ethz.ch/mailman/listinfo/r-help`
- phylogenetics: R-sig-phylo
  `https://stat.ethz.ch/mailman/listinfo/r-sig-phylo`
- genetics: R-sig-phylo
  `https://stat.ethz.ch/mailman/listinfo/r-sig-genetics`
- *adegenet*: the adegenet forum
  `http://lists.r-forge.r-project.org/mailman/listinfo/adegenet-forum`

Asking questions on Github

Using github's **issue** system

- every github project has an 'issue' system for discussions
- mainly used for: questions, bug reports, feature requests, contributions
- accept *markdown* notations

Asking questions on Github

Using github's **issue** system

- every github project has an 'issue' system for discussions
- mainly used for: questions, bug reports, feature requests, contributions
- accept *markdown* notations

Asking questions on Github

Using github's **issue** system

- every github project has an 'issue' system for discussions
- mainly used for: questions, bug reports, feature requests, contributions
- accept *markdown* notations

Asking questions on Github

Using github's **issue** system

- for adegenet, go to:
  `https://github.com/thibautjombart/adegenet/issues`
- or just type in R: `adegenetIssues()`

Logicals operators 1/2

Basic logical operators

```
> a <- 1:10; a

 [1]  1  2  3  4  5  6  7  8  9 10

> a < 5

 [1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE

> a > 5

 [1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE

> a <= 5

 [1]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE

> a == 5

 [1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE

> a != 5

 [1]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
```

Using logicals 2/2

Logicals operators recycle their arguments

The shortest argument is repeated to match the length of the longer one.

```
> a <- 1:10
> b <- c(1,10)
> a == b

 [1]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
```

Same happens with arithmetic operators:

```
> a + b

 [1]  2 12  4 14  6 16  8 18 10 20
```

Logicals as numbers

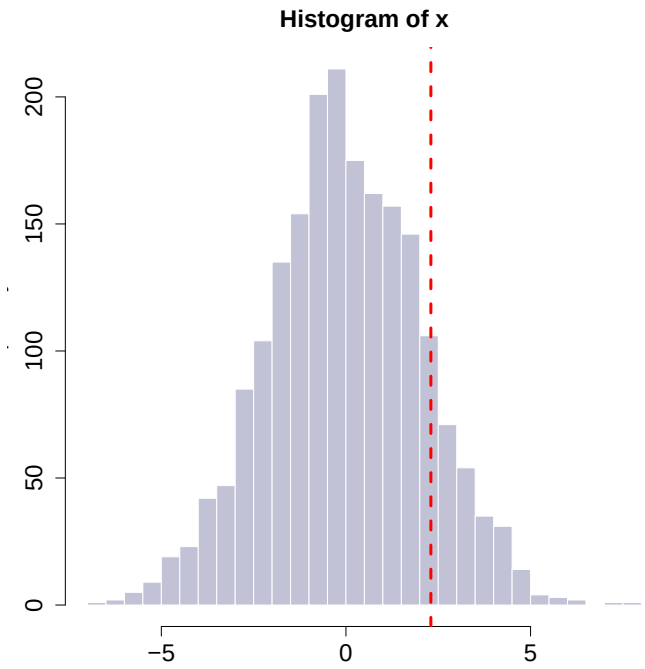
In short: TRUE=1, FALSE=0.

```
> a <- c(TRUE, FALSE, TRUE)
> sum(a)

[1] 2

> mean(a)

[1] 0.6666667

> !a

[1] FALSE  TRUE FALSE

> sum(!a)

[1] 1
```

Logicals as numbers: example

What is the p-value of $X = 2.3$ given the empirical distribution 'x'?

```
> x <- rnorm(1000, sd=2)
> hist(x, nclass=30, col="#c2c2d6",
+      border="white")
> abline(v = 2.3, lty=2, col="red")
```




```
> mean(x >= 2.3)

[1] 0.1295

> pnorm(2.3, 0, 2, lower.tail=FALSE)

[1] 0.1250719
```

Matching values

`which`: indices of `TRUE`s in a logical vector:

```
> which(c(TRUE, FALSE, FALSE, TRUE, FALSE))

[1] 1 4
```

`%in%`: which x is in y:

```
> c(1,5,11) %in% 0:10

[1]  TRUE  TRUE FALSE
```

`match`: positions of x matched in y:

```
> match(c(1,5,11), 0:10)

[1]  2  6 NA
```

Matching values

`which`: indices of `TRUE`s in a logical vector:

```
> which(c(TRUE, FALSE, FALSE, TRUE, FALSE))

[1] 1 4
```

`%in%`: which x is in y:

```
> c(1,5,11) %in% 0:10

[1]  TRUE  TRUE FALSE
```

`match`: positions of x matched in y:

```
> match(c(1,5,11), 0:10)

[1]  2  6 NA
```

Matching values

`which`: indices of `TRUE`s in a logical vector:

```
> which(c(TRUE, FALSE, FALSE, TRUE, FALSE))

[1] 1 4
```

`%in%`: which x is in y:

```
> c(1,5,11) %in% 0:10

[1]  TRUE  TRUE FALSE
```

`match`: positions of x matched in y:

```
> match(c(1,5,11), 0:10)

[1]  2  6 NA
```

Subsetting using indices

```
> x <- letters[1:10]
> x

 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"

> x[c(1,3,4)]

[1] "a" "c" "d"

> x[-c(1,3,4)]

[1] "b" "e" "f" "g" "h" "i" "j"
```

Subsetting using logicals

```
> set.seed(1)
> x <- sample(1:10)
> x

 [1]  3  4  5  7  2  8  9  6 10  1

> x > 5

 [1] FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE

> x[x > 5]

[1]  7  8  9  6 10
```

Subsetting using names

```
> names(x) <- letters[1:10]
> x

 a  b  c  d  e  f  g  h  i  j
 3  4  5  7  2  8  9  6 10  1

> x[c("a", "g", "a", "b", "c")]

a g a b c
3 9 3 4 5
```

Application: subsetting a genind object

```
> data(microbov)
> microbov


/// GENIND OBJECT /////////

 // 704 individuals; 30 loci; 373 alleles; size: 1.1 Mb

 // Basic content
   @tab:  704 x 373 matrix of allele counts
   @loc.n.all: number of alleles per locus (range: 5-22)
   @loc.fac: locus factor for the 373 columns of @tab
   @all.names: list of allele names for each locus
   @ploidy: ploidy of each individual  (range: 2-2)
   @type:  codom
   @call: genind(tab = truenames(microbov)$tab, pop = truenames(microbov)$pop)

 // Optional content
   @pop: population of each individual (group size range: 30-61)
   @other: a list containing: coun  breed  spe
```

Application: subsetting a genind object

Keep only Salers and Zebu, and loci with $\geq$ 10 alleles:

```
> ind.to.keep <- pop(microbov) %in% c("Salers","Zebu")
> loc.to.keep <- nAll(microbov) >= 10
> x <- microbov[i=ind.to.keep, loc=loc.to.keep]
> x


/// GENIND OBJECT /////////

 // 100 individuals; 22 loci; 310 alleles; size: 178.3 Kb

 // Basic content
   @tab:  100 x 310 matrix of allele counts
   @loc.n.all: number of alleles per locus (range: 10-22)
   @loc.fac: locus factor for the 310 columns of @tab
   @all.names: list of allele names for each locus
   @ploidy: ploidy of each individual  (range: 2-2)
   @type:  codom
   @call: .local(x = x, i = i, j = j, loc = ..1, drop = drop)

 // Optional content
   @pop: population of each individual (group size range: 50-50)
   @other: a list containing: coun  breed  spe
```

Is this 'refresher' over yet??

There's more!

- looping: `for`, `apply`, `lapply`, `sapply`, ...
- regular expressions: `grep`, `sub`, `gsub`, ...
- classes and methods: S3, S4, R6, ...
- integration of foreign languages: `.C`, `.Call`, *Rcpp*, ...
- advanced graphics: *ggplot2*, *ggvis*, JS plugins, ...

But ...

Is this 'refresher' over yet??

There's more!

- looping: `for`, `apply`, `lapply`, `sapply`, ...
- regular expressions: `grep`, `sub`, `gsub`, ...
- classes and methods: S3, S4, R6, ...
- integration of foreign languages: `.C`, `.Call`, *Rcpp*, ...
- advanced graphics: *ggplot2*, *ggvis*, JS plugins, ...

But ...

Is this 'refresher' over yet??

There's more!

- looping: `for`, `apply`, `lapply`, `sapply`, ...
- regular expressions: `grep`, `sub`, `gsub`, ...
- classes and methods: S3, S4, R6, ...
- integration of foreign languages: `.C`, `.Call`, *Rcpp*, ...
- advanced graphics: *ggplot2*, *ggvis*, JS plugins, ...

But ...

Is this 'refresher' over yet??

There's more!

- looping: `for`, `apply`, `lapply`, `sapply`, ...
- regular expressions: `grep`, `sub`, `gsub`, ...
- classes and methods: S3, S4, R6, ...
- integration of foreign languages: `.C`, `.Call`, *Rcpp*, ...
- advanced graphics: *ggplot2*, *ggvis*, JS plugins, ...

But ...

Is this 'refresher' over yet??

There's more!

- looping: `for`, `apply`, `lapply`, `sapply`, ...
- regular expressions: `grep`, `sub`, `gsub`, ...
- classes and methods: S3, S4, R6, ...
- integration of foreign languages: `.C`, `.Call`, *Rcpp*, ...
- advanced graphics: *ggplot2*, *ggvis*, JS plugins, ...

But ...

Is this 'refresher' over yet??

There's more!

- looping: `for`, `apply`, `lapply`, `sapply`, ...
- regular expressions: `grep`, `sub`, `gsub`, ...
- classes and methods: S3, S4, R6, ...
- integration of foreign languages: `.C`, `.Call`, *Rcpp*, ...
- advanced graphics: *ggplot2*, *ggvis*, JS plugins, ...

But ...

Is this 'refresher' over yet??

There's more!

- looping: `for`, `apply`, `lapply`, `sapply`, ...
- regular expressions: `grep`, `sub`, `gsub`, ...
- classes and methods: S3, S4, R6, ...
- integration of foreign languages: `.C`, `.Call`, *Rcpp*, ...
- advanced graphics: *ggplot2*, *ggvis*, JS plugins, ...

But ...

If you're seeing this, you need a break.