Introduction to *R*

Martin T. Morgan[1]

27-28 February 2014

[1] mtmorgan@fhcrc.org

R

- http://r-project.org
- Open-source, statistical programming language; widely used in academia, finance, pharma, ...
- Core language and base packages
- Interactive sessions, scripts
- $>$ 5000 contributed packages

```
## Two 'vectors'
x <- rnorm(1000)
y <- x + rnorm(1000, sd=.5)
## Integrated container
df <- data.frame(X=x, Y=y)
## Visualize
plot(Y ~ X, df)
## Regression; 'object'
fit <- lm(Y ~ X, df)
## Methods on the object
abline(fit) # regression line
anv <- anova(fit) # ANOVA ta-
ble
```

Programming *R*

1. Packages: loading, installing
2. Help
3. Scripts & reproducible research
4. Functions
5. Debugging and measuring performance

1. Packages

Already installed packages

```
library(parallel)
```

New packages from *repositories* such as CRAN and Bioconductor

- `biocLite()` to install, including *dependencies*
- Occasional problems when a package depends on third-party software installation

```
source("http://bioconductor.org/biocLite.R")
biocLite("IRanges")
library("IRanges")
```

Other repositories: `R-forge`, `github`, ...

Packages (cont.)

What packages are loaded?

```
head(search(), 3)
```

```
## [1] ".GlobalEnv"        "package:quantreg" "package:Spars
```

What functions are provided by a package?

```
help(package = "IRanges")
```

How does *R* find symbols, e.g,. `sin`?

- Look in `.GlobalEnv`, then proceed down search path
- Specify package with `base::sin`

2. Help

```
help.start()
? data.frame
? anova
? anova.lm  # anova generic, method for class lm
class ? DNAStringSet
method ? "alphabetFrequency,DNAStringSet"
vignette("GenomicRangesIntroduction", "GenomicRanges")
help(package = "Biostrings")
RShowDoc("R-intro")
```

3. Scripts, functions, and reproducible research

1. Write simple scripts of *R* code `my_analysis.R`
2. Implement common operations as functions.
3. 'Markdown' with *R* code embedded in surrounding text `my_analysis.Rmd`
4. Packages! – `package.skeleton()`
5. Version control!

4. Favorite functions

`dir`, `read.table`, `scan` List files; input data.

`c`, `factor`, `data.frame`, `matrix` Create vectors, etc.

`summary`, `table`, `xtabs` Summarize or cross-tabulate data.

`t.test`, `lm`, `anova` Compare two or several groups.

`dist`, `hclust`, `heatmap` Cluster data.

`plot` Plot data.

`ls`, `library` List objects; attach packages.

`lapply`, `sapply`, `mapply` Apply function to elements of lists.

`match`, `%in%` find elements of one vector in another.

`split`, `cut` Split or cut vectors.

`strsplit`, `grep`, `sub` Operate on character vectors.

`biocLite` Install a package from an on-line repository.

5. Debugging and measuring performance

Debugging

- `traceback()`: what went wrong?
- `debug()`: step through a function.
- `browser()`: insert a break-point in your own function / script. Help debug errors.

Debugging and measuring performance (cont.)

Performance

- `all.equal()`, `identical()` to compare values.
- `system.time()` to measure how long evaluation takes.
- *microbenchmark* to compare times for different functions
- `Rprof()` to summarize time in each function call, lineprof to profile each line of code