Data structures in R

Fundamentals of Computing and Data Display

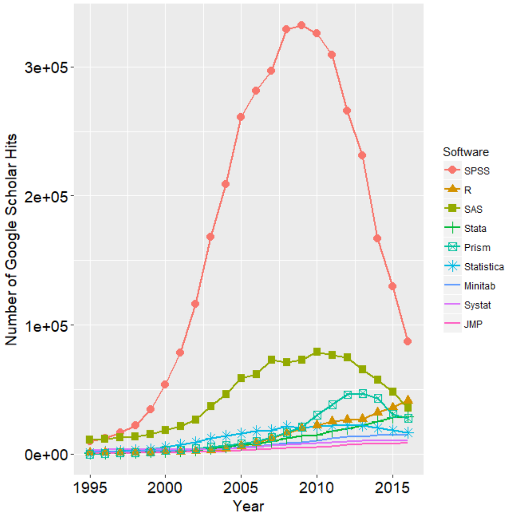
Christoph Kern Ruben Bach

c.kern@uni-mannheim.de

Outline

Introduction

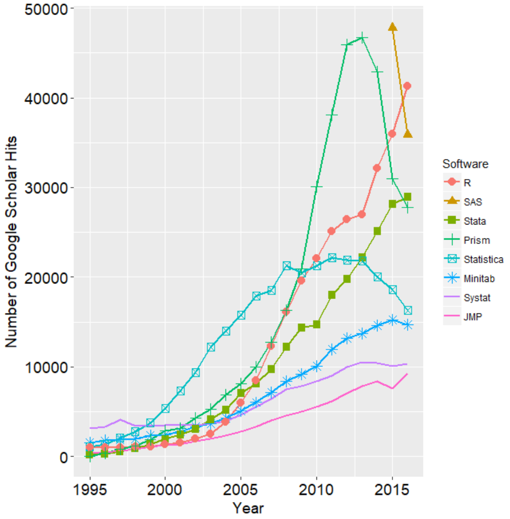
Figure: Number of scholarly articles for top six “classic” statistics packages[1]



[1] http://r4stats.com/articles/popularity/

Introduction

Figure: Number of scholarly articles for “classic” statistics packages, SPSS and SAS removed[2]



[2] `http://r4stats.com/articles/popularity/`

Introduction

Why use R?

- R is free (Open Source)
- Extensive statistical modeling capabilities (through ∼15,000 packages[3])
- Flexible programming of user functions
- Working with multiple objects (e.g. various datasets, model results)
- Active user community providing up-to-date functions & support

[3] https://cran.r-project.org/web/packages/

R Basics

R Overview

- Download R & RStudio
  - `http://www.r-project.org/`
  - `https://www.rstudio.com/`
- Version & citation
  - `sessionInfo()`
  - `citation()`
- File formats
  - .Rdata: Workspace
  - .R: Code
  - .Rhistory: (saved) code
  - .Rmd: R Markdown file

R Basics

Interacting with R

- RStudio
  - R Script Editor, R Console
- RGui (Windows)
  - Same as above, minus most features
- The shell/ terminal
  - Linux: Start R session in terminal: `R`
  - `https://www.statmethods.net/interface/batch.html`
- Text editor(s)
  - Various text editors support editing R scripts and interacting with R
  - e.g. `https://github.com/jalvesaq/Nvim-R`

R Basics

Setup

- List & change options: `options()`
- Set global options through .Rprofile
- Show, set working directory: `getwd()`, `setwd()`
- List files: `dir()`
- Find your files: `https://here.r-lib.org/`

Input

- Execute .R file: `source("file.R", echo = TRUE)`

Output

- Save code: `savehistory("file.Rhistory")`
- Save output: `sink("file.txt", split = TRUE)`
- `http://rmarkdown.rstudio.com/`

Functions

```
function(object[...], option = ...)
```

- Functions
  - `function` without `()`: Show code of function
  - `function()`: ~List corresponding elements
  - `function(...)`: Execute function as specified with `(arguments)`
- Syntax
  - R is case sensitive
  - Abbreviations of functions are not allowed
  - Comments begin with #
- Help
  - Function unknown: `help.search("topic")`
  - Function known: `help(function)`
  - Show similar functions: `fun...  + tab-key`
  - Show available options: `function( + tab-key`

Style guide

- `http://r-pkgs.had.co.nz/style.html`
    - File names: If files need to be run in sequence, prefix them with numbers
    - Object names: Use an underscore (_) to separate words
    - Spacing: Place spaces around all infix operators (=, +, -, <-, etc.) and after commas
    - Use <-, not =, for assignment
    - Comment your code (chunks)
    - ...

R file headers

- Title:
- Filename: 000_main.R
- Description:
- Author:
- Maintainer:
- DOI:
- Created: So Jan 20 04:34:27 2019 (+0100)
- Last-Updated: Mo Jan 28 17:36:09 2019 (+0100)
- Version:
- Dependencies: [Software (Version), Software (Version)...)
- Depends on: [File1, le2, ...]
- Data source: [name of dataset] (DOI)
- URL:
- Keywords:

Packages

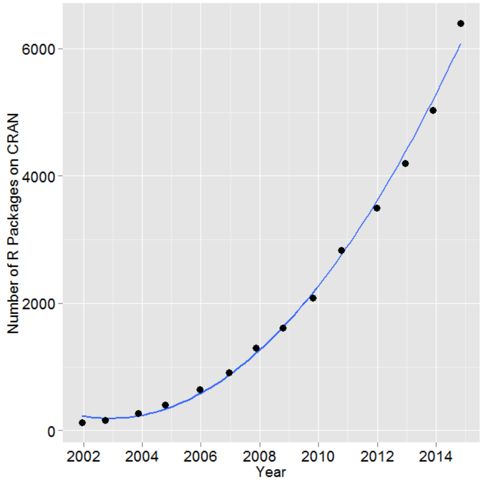
- Functions are included in (numerous) packages
- List installed packages: `library()`
- List loaded packages: `search()`
- Update packages: `update.packages()`
- Searching for packages
  - `available.packages()`
  - `http://cran.r-project.org/web/views/`
- Installing & loading packages
  1. `install.packages("package")`
  2. `library(package)`

Packages

base	recommended
base	boot
compiler	class
datasets	cluster
graphics	codetools
grDevices	foreign
grid	KernSmooth
methods	lattice
parallel	MASS
splines	Matrix
stats	mgcv
stats4	nlme
tcltk	nnet
tools	rpart
utils	spatial
	survival

Packages

Figure: Number of R packages on CRAN[4]



[4] `http://r4stats.com/articles/popularity/`

Data structures

Everything in R is an object...

- Data structures: `str(...)`, `class(...)`
  - vector (“atomic vectors”): A set of values / elements
  - factor: A set of named elements
  - data.frame: A two-dimensional set of vectors and/or factors
  - matrix: A two-dimensional set of objects with the same mode
  - array: A multidimensional matrix
  - list (“generic vectors”): A combinations of various objects
- Mode of an object: `mode(...)`
  - numeric
  - character
  - logical
  - ...
- Type of an object: `typeof(...)`
  - integer
  - double
  - character
  - logical
  - ...

Data structures

Working with objects

- Test whether object is of a certain type: `is.character()`, `is.integer()` ...
- Coercion into (other) types: `as.character()`, `as.integer()` ...
- Metadata of objects: `attributes()`
- Length, dimension of objects: `length()`, `dim()`

Object-oriented programming systems in R

- S3
  - Classes are attached to objects as attributes
  - Methods are implemented as generic functions
- S4

Objects & workspace

The workspace

- Contains all available objects...
  - Datasets
  - Subsets of data
  - Model results
  - ...
- List all objects: `ls()`
- Print content of an object: `object`
- **Create object**: `object <- ...`
- Structure of an object: `str(object)`
- Remove object: `rm(object)`

Working with data (Base R)

Data import

Table: Data import in R

	Base R	**readr**	**foreign**	**haven**
	`data.frame`	`tibble`	`data.frame`	`tibble`
Generic / Text Files	`read.table()`	`read_delim()`		
Comma-Separated	`read.csv()`	`read_csv()`		
	`read.csv2()`	`read_csv2()`		
Tab-Separated	`read.delim()`	`read_tsv()`		
	`read.delim2()`	`read_tsv2()`		
SPSS files			`read.spss()`	`read_sav()`
Stata files			`read.dta()`	`read_dta()`
SAS files			`read.ssd()`	`read_sas()`

Accessing data

Selecting variables and observations

- Working with indexes
  - Basic structure: `data[...,...]`
  - Selecting variables: `data[ ,1:3]`
  - Selecting variables: `data[ ,c(1,3,4)]`
  - Selecting observations: `data[1:10, ]`
- Selecting variables using $-notation
  - Basic structure: `data$var1`
  - Combine multiple variables with `data.frame()`
- ~~Selecting variables using `attach`~~

Accessing data

Table: Subsetting and output structure[5]

	Simplifying	Preserving
Vector	x[[1]]	x[1]
List	x[[1]]	x[1]
Factor	x[1:4, drop = T]	x[1:4]
Array	x[1,] or x[, 1]	x[1, , drop = F] or x[, 1, drop = F]
Data frame	x[, 1] or x[[1]]	x[, 1, drop = F] or x[1]

[5] http://adv-r.had.co.nz/Subsetting.html

Managing data

- Aggregating data
  - Collapse data frames: `aggregate()`
- Merging data
  - Merge data frames: `merge()`
  - Combine objects by rows or columns: `rbind()`, `cbind()`
- Subsetting data
  - Subset objects based on conditions: `subset()`
  - Split data based on a factor: `split()`
  - Draw random samples: `sample()`
- If ... do ... else ...
  - `ifelse(test, yes, no)`

Exploring data

Description I

- Dataviewer and -editor
  - View dataset: `View(...)`
  - Edit dataset: `fix(...)`
- Data overview
  - Data structure / dimension: `str(...)`
  - Attributes (e.g. labels): `attributes(...)`
  - Variable names: `names(data)`
  - List first, last observations: `head(...)`, `tail(...)`
  - Number of observations, variables: `nrow(...)`, `ncol(...)`

Exploring data

Description II

- Central tendency
  - Arithmetic mean: `mean(...)`
  - Median: `median(...)`
  - Quantiles: `quantile(...)`
- Dispersion
  - Variance: `var(...)`
  - Standard deviation: `sd(...)`
  - Interquartile range: `IQR(...)`
- Summary & table
  - mean, median, 25th & 75th quartiles, min, max: `summary(...)`
  - Tukey's five number summary: `fivenum(...)`
  - Frequencies: `table(...)`
  - Cross tabulation: `table(...,...)`
  - Proportions: `prop.table(...)`

(Functional) Programming

(Functional) Programming

Motivation

- Working with data often involves repeating some (customized) coding procedure
  - Copy-and-paste blocks of code is prone to errors
  - Repeated code is annoying to update
  - Purpose of code not clear at first sight
- Consider writing a function whenever there are three or more copies of the same block of code
  - “Do not repeat yourself” (DRY)

```
> # Compute Pearson’s second skewness coefficient for each variable
> 3*(mean(df$a) - median(df$a))/sqrt(var(df$a))
> 3*(mean(df$b) - median(df$b))/sqrt(var(df$b))
> 3*(mean(df$c) - median(df$c))/sqrt(var(df$c))
> 3*(mean(df$d) - median(df$d))/sqrt(var(df$d))
```

(Functional) Programming

For loops

- Automate iterations
  - Output: Initialize empty object
  - Sequence: Determine what to loop over
  - Body: Code that is run in each iteration
- Can often be avoided in R

```
> # Compute Pearson’s second skewness coefficients in a for loop
> output <- vector(‘‘double’’, length(df))
> for (i in seq_along(df)) {
>   output[i] <- 3*(mean(df[,i]) - median(df[,i])) / sqrt(var(df[,i]))
> }
```

(Functional) Programming

Functions

- Generalize code for repeated tasks
  - Name: Short and clear description of what function does
  - Arguments: Supply data and control instructions
  - Body: Code that is run given the arguments
- Should follow a consistent coding style

```
> # A function to compute Pearson's second skewness coefficient
> pearsons_skew2 <- function(x) {
>   3*(mean(x) - median(x)) / sqrt(var(x))
> }
```

(Functional) Programming

Functional Programming

- Functions are objects in R
- Composing functions can help reducing redundancy and duplication
  - Functionals: Functions that take functions as arguments
  - Function operators: Take functions as input and return functions as output

```
> # A simple functional
> col_summary <- function(df, fun) {
>   output <- vector("double", length(df))
>   for (i in seq_along(df)) {
>     output[i] <- fun(df[,i])
>   }
>   output
> }
> col_summary(df, mean)
> col_summary(df, pearsons_skew2)
```

Resources

- `help.start()`
- Books
    - Adler, J. (2012). *R in a Nutshell*. Sebastopol, CA: O'Reilly.
    - Crawley, M. J. (2007). *The R Book*. Chichester: Wiley.
- Reference Manuals
    - `https://cran.r-project.org/manuals.html`
- Online Learning
    - `https://www.rstudio.com/online-learning/#R`
    - `http://www.statmethods.net/`
- R vocabulary
    - `http://adv-r.had.co.nz/Vocabulary.html`
- Awesome R packages
    - `https://awesome-r.com/`