Modules in R

Sebastian Warnholz – INWT Statistics GmbH

July 7th, 2017 – UseR! – Brussels
## What we do!

### INWT Statistics GmbH

INWT Statistics specializes in intelligent data analysis and delivers solutions in the fields of online marketing, CRM, data management and business intelligence/reporting.

### Online Marketing
- Customer Journey Analysis
- Conversion Optimization
- Fraud Detection

### CRM
- Customer Lifetime Value
- Customer Segmentation
- Churn Management

### BI/Reporting
- Data Management
- Data Consolidation
- Dashboards

### Training

### Selected Customers:

- Cosmodirekt
- OTTO
- SIXT
- Tipico
- Webtrekk
- Zalando
Managing dependencies between several R-scripts
Motivation

Managing dependencies between several R-scripts

Options for organising functions in larger code bases
What are modules?

**Script**
- Interactive data analysis
- Everything ad-hoc
- Managing dependencies and names
- Organise helper functions

**Module**
- Local namespace
- Import + export
- Bundle related functions
- No Documentation
  Delivery inside packages

**Package**
- Documentation
- Dependencies
- Namespace (imports and exports)
- Share / deliver code
- One, global scope
One-file setup

```r
stats <- module({
  import("stats", "median")
  myMedian <- function(x) median(x, TRUE)
})

stats$myMedian(rnorm(10))
```
One-file setup

```r
stats <- module({
  import("stats", "median")
  myMedian <- function(x) median(x, TRUE)
})
stats$myMedian(rnorm(10))
```

Multiple-file setup

```r
helperStats.R
import("stats", "median")
myMedian <- function(x) ...
```

```r
analysis.R
stats <-
modules::use("helperStats.R")
stats$myMedian(rnorm(10))
```
Code example II

Dependencies in modules:

```r
stats <- module({
  import("stats", "median")
  export("myMedian")
  myMedian <- function(x) median(x, TRUE)
})
```

**import**: state dependencies explicitly

**export**: state what clients can use

**search path**: inside packages, the root is the packages namespace

Local search path:

- package: base
- package: stats
- module: stats

---

ImporT: STATEDEPENDENCIESExPLICITLY
exporT: STATEWHATCLIENTSCANUSE
SearcH paTH: INSIDEPACkAGES,THEROOTISTHEPACkAGES
NAMESPACE
Nested module definition:

```r
mathStuff <- module(
    add <- function(x, y) x + y
    subtract <- function(x, y) x - y
    usefulAbstractions <- module(
        add1 <- function(x) add(x, 1)
        add <- function(x) x + 1
    )
)

mathStuff$add(1, 2)
mathStuff$usefulAbstractions$add1(2)
```

Local search path:
An abstraction between functions and packages ...
When to use modules

Use cases

- In larger code bases
  - to bundle related functions
  - as units of abstraction
  - not as a substitute for packages

- Analysis distributed across several scripts
  - `funA` from `scriptA` has its own and local set of dependencies
  - using `import` has no side effects in the global environment

- Substitute for library

Alternatives

- Packages for Object Orientation (`methods`, `proto`, `R.oo`, `R6`)
- The package `import` as substitute for library

Try it...

```r
install.packages("modules")
vignette("modulesInR")
```
Thank You for Your Attention!

INWT Statistics GmbH
E-Mail: info@inwt-statistics.de
Internet: www.inwt-statistics.de
Tel.: +49 30 609857990