Data in R packages

Sophie Schmidt

Why should we want to put data in a package?

- example data for use cases
- distribute data along with a documentation for others to use
- is part of the service your package provides

Three ways to add data to a package

Three ways to add data to a package

- binary: use the folder data/
- parsed data, that's not available to the user: store it as R/sysdata.rda
- raw data, availabe for the user: inst/extdata

–> in package development, working with the source package, data/ is the usual choice

Exported data using /data

- save each object into an Rdata-file with the same name
- the `use_data()` function can take several objects, will create the data folder and write the objects as Rdata-files in there using the object names for file names.

```
x <- sample(1000)
usethis::use_data(x, mtcars)
```

–> leads to data/x.Rda and data/mtcars.Rda

- DESCRIPTION: `LazyData: true` –> data will be lazily loaded –> doesn't occupy memory until used
- is the default when using `usethis::create_package()`

raw data

- data included in the package is often a cleaned version of some raw data
- recommended: include the raw data + the code used to clean it in the source version of the package
- makes it easy to update and reproduce the package
- this code can go in a data-raw/ folder
- isn't needed in the bundled version of the package –> add it to .Rbuildignore.
- usethis wizardry does it all for you:

```
usethis::use_data_raw()
```

- input should be name of dataset –> a string in " "

Documenting data

Documenting data

- objects in data/ always have to be documented!
- similar to documenting functions
- can't write Roxygen documentation "into" the dataset
- instead: write it in an R-file in R/ with the same name as the data
- like function documentation: #', first paragraph = title, second paragraph = description

example documentation (from ggplot2):

```
#' Prices of 50,000 round cut diamonds.
#'
#' A dataset containing the prices and other attributes of almost 54,000
#' diamonds.
#'
#' @format A data frame with 53940 rows and 10 variables:
#' \describe{
#'   \item{price}{price, in US dollars}
#'   \item{carat}{weight of the diamond, in carats}
#'   ...
#' }
#' @source \url{http://www.diamondse.info/}
"diamonds"
```

- `@format` overview over dataset, description of variables and their units + `@source`: where you got the data from
- DON'T `@export` your data

internal data

- sometimes functions need "invisible" pre-computed data tables
- save these in R/sysdata.rda
- example: munsell uses R/sysdata.rda to store large tables of colour data
- `usethis::use_data()` to create this file with the argument `internal = TRUE`:

```
x <- sample(1000)
usethis::use_data(x, mtcars, internal = TRUE)
```

- code used to prepare this –> data-raw/
- Objects in R/sysdata.rda are not exported –> don't need to be documented

Raw data for the bundled package

- if you want to show e.g. how to load raw data –> inst/extdata
- all files in inst/ move up one level to the top-level directory when built
- to refer to files in inst/extdata (whether installed or not), use system.file()
- readr package uses inst/extdata to store delimited files for use in examples:

```
system.file("extdata", "mtcars.csv", package = "readr")
#> [1] "/Users/runner/work/_temp/Library/readr/extdata/mtcars.csv"
```

- by default, if the file does not exnist, system.file() does not return a error - it just returns the empty string:
- argument mustWork = TRUE –> error message if file doesn't exist

Exercise!

Exercise!

- create a small dataset, save it with `usethis::use_data()` and try `usethis::use_data_raw()`
- document the data in an R-file within the R-folder (same name as dataset!)
- remember to use `@format` with

```
\describe{
 \item {variable}{unit}
 }

# example data:
ceram <- data.frame(c("A","B","C"), c(10,5,2), c(10.5,2.6,3.4))
colnames(ceram) <-c("sites", "n_types", "ha")
```