Making an R package

Canhong Wen

Why develop your own packages?

R packages are an ideal way to package and distribute R code and data for re-use by others.

- **Community**: CRAN / packages part of R success
- **cross-platform / cross-OS**: packages are portable
- **gettting `R` code to colloborators**: distribution
- **reproducibility**: aided greatly by identifible package versions
- **version control**: learn about git (or svn)

Part I

Package structure

Software Prerequisites

There are two main prerequisites for building R packages:

1. GNU software development tools including a C/C++ compiler; and
2. LaTeX for building R manuals and vignettes.
3. Rtools (Windows)

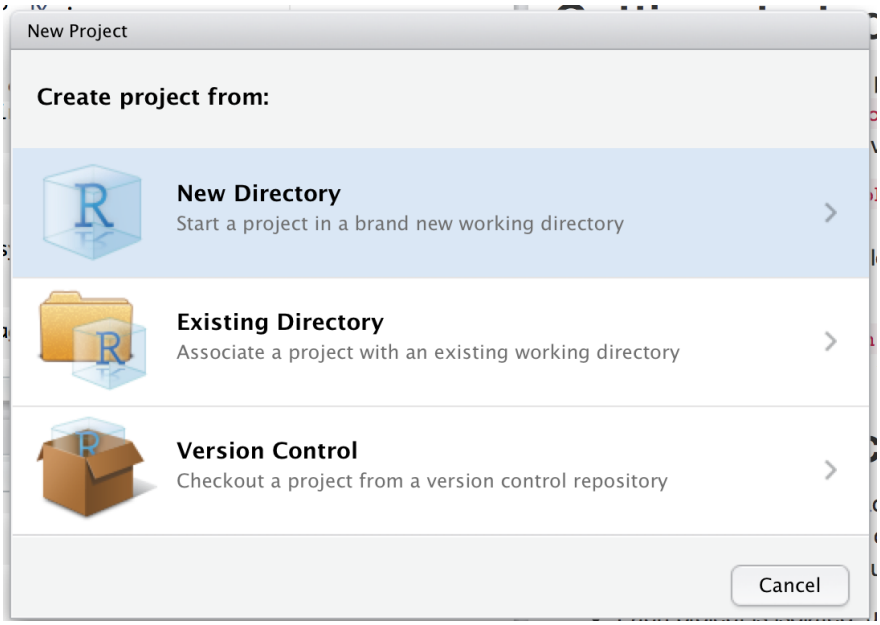
Reference: 1. R packages by Hadket wickham. See also at http://r-pkgs.had.co.nz/

Requirements for a name

There are three formal requirements: - the name can only consist of letters, numbers and periods, i.e., .; - it must start with a letter; - it cannot end with a period.

Strategies for creating a name

- Pick a unique name you can easily Google.
- Avoid using both upper and lower case letters.
- Find a word that evokes the problem and modify it so that it's unique:
  - plyr is generalisation of the apply family, and evokes pliers.
  - knitr (knit + r) is "neater"" than sweave (s + weave).
  - testdat tests that data has the correct format.
- Use abbreviations:
  - Rcpp = R + C++ (plus plus)
  - lvplot = letter value plots.
- Add an extra R:
  - stringr provides string tools.
  - tourr implements grand tours (a visualisation method).
  - gistr lets you programmatically create and modify GitHub gists.



Creating a New Package

1. Click File | New Project.
2. Choose "New Directory":

3.Then "R Package":

4.Then give your package a name and click "Create Project":

The package you just created

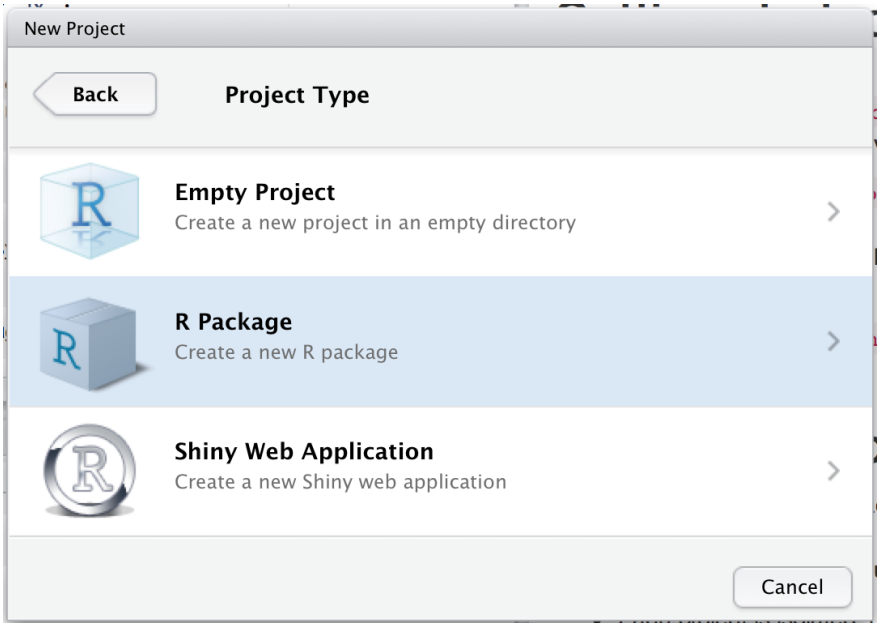
The smallest usable package, one with three components:

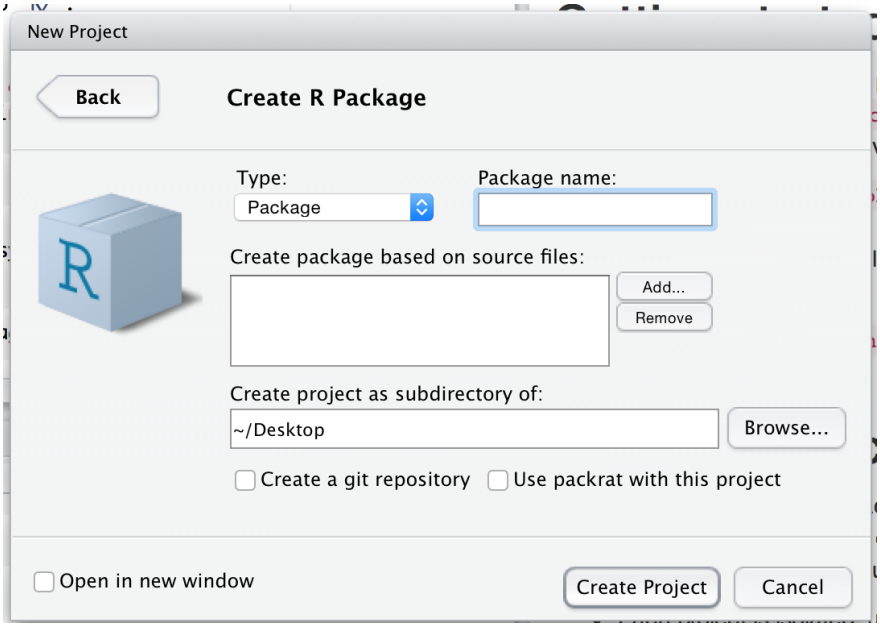
1. An R/ directory.
2. A basic DESCRIPTION file.
3. A basic NAMESPACE file.

It will also include an RStudio project file, pkgname.Rproj, that makes your package easy to use with RStudio.

Five states a package across its lifecycle:

1. Source
2. Bundled
3. Binary
4. Installed
5. In-memory


New Project
Back
Project Type
Empty Project
Create a new project in an empty directory
R Package
Create a new R package
Shiny Web Application
Create a new Shiny web application
Cancel


New Project
Back
Create R Package
Type:
Package
Package name:
Create package based on source files:
Add...
Remove
Create project as subdirectory of:
~/Desktop
Browse...
Create a git repository
Use packrat with this project
Open in new window
Create Project
Cancel

Source packages

- The development version of a package that lives on your computer.
- A source package is just a directory with components like R/, DESCRIPTION

Bundled packages

- A **bundled** package is a package that has been compressed into a single file.
- Package bundles in R use the extension .tar.gz.
- The main differences between an uncompressed bundle and a source package are:
  - Vignettes are built so that you get HTML and PDF output instead of Markdown or LaTeX input.
  - Your source package might contain temporary files used to save time during development, like compilation artefacts in src/. These are never found in a bundle.
  - Any files listed in .Rbuildignore are not included in the bundle.

- **.Rbuildignore** prevents files in the source package from appearing in the bundled package.
- It allows you to have additional directories in your source package that will not be included in the package bundle.
- A typical example

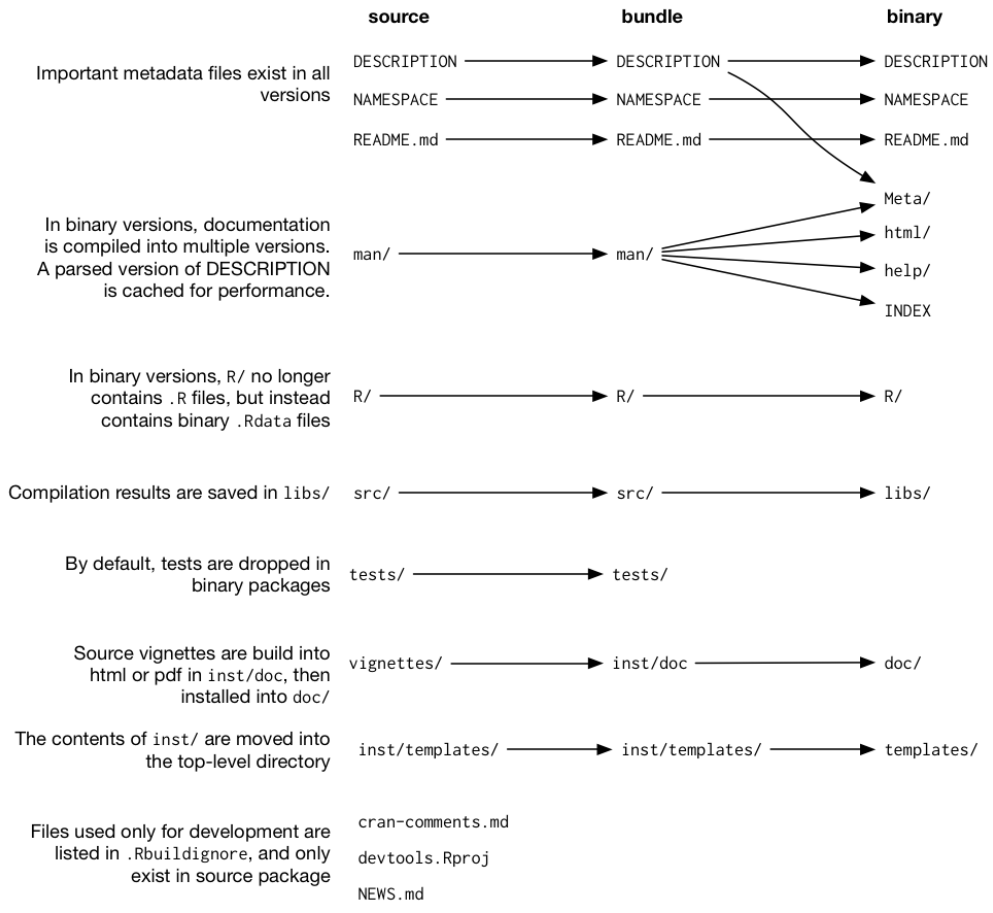
```
^.*\.Rproj$          # Automatically added by RStudio,
^\.Rproj\.user$      # used for temporary files.
^README\.Rmd$        # An Rmarkdown file used to generate README.md
^cran-comments\.md$ # Comments for CRAN submission
^NEWS\.md$           # A news file written in Markdown
^\.travis\.yml$      # Used for continuous integration testing with travis
```

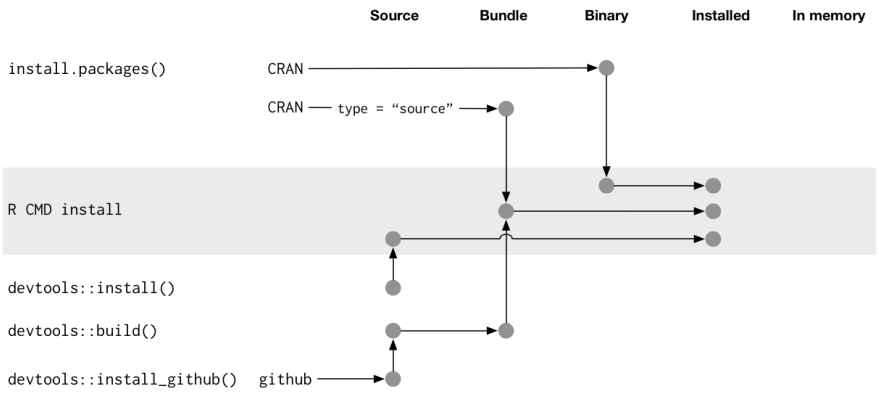
Binary packages

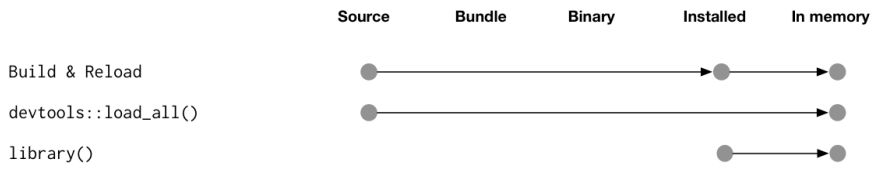
- If you want to distribute your package to an R user who doesn't have package development tools, you will need to make a **binary** package.
- Binary packages are platform specific:
  - Mac binary packages end in .tgz
  - Windows binary packages end in .zip

Installed packages

- An **installed** package is just a binary package that has been decompressed into a package library.
- The following diagram illustrates the many ways a package can be installed.
- An easier way: using **RStudio**


source
bundle
binary
Important metadata files exist in all versions
DESCRIPTION
NAMESPACE
README.md
In binary versions, documentation is compiled into multiple versions. A parsed version of DESCRIPTION is cached for performance.
man/
Meta/
html/
help/
INDEX
In binary versions, R/ no longer contains .R files, but instead contains binary .Rdata files
R/
Compilation results are saved in libs/
src/
libs/
By default, tests are dropped in binary packages
tests/
Source vignettes are build into html or pdf in inst/doc, then installed into doc/
vignettes/
inst/doc
doc/
The contents of inst/ are moved into the top-level directory
inst/templates/
templates/
Files used only for development are listed in .Rbuildignore, and only exist in source package
cran-comments.md
devtools.Rproj
NEWS.md


Source
Bundle
Binary
Installed
In memory
install.packages()
CRAN
CRAN
type = "source"
R CMD install
devtools::install()
devtools::build()
devtools::install_github()
github



In memory packages

- To use a package, you must load it into memory.
- To use it without providing the package name, you need to attach it to the search path.

```
# Automatically loads devtools
devtools::install()

# Loads and _attaches_ devtools to the search path
library(devtools) # or
require(devtools)
install()
```

- `library()` is not useful when you're developing a package because you have to install the package first.

What is a library?

- A library is simply a directory containing installed packages.
- Almost every one has at least two libraries:
  - one for packages you're installed
  - one for the packages that come with every R installation (like base, stats, etc).

```
.libPaths()
```

```
## [1] "C:/Users/wench/Documents/R/win-library/3.6"
## [2] "C:/Program Files/R/R-3.6.1/library"
```

```
lapply(.libPaths(), dir)[[2]]
```

```
##  [1] "base"          "boot"          "class"         "cluster"
##  [5] "codetools"     "compiler"      "datasets"      "foreign"
##  [9] "graphics"      "grDevices"     "grid"          "KernSmooth"
## [13] "lattice"       "MASS"          "Matrix"        "methods"
## [17] "mgcv"          "nlme"          "nnet"          "parallel"
## [21] "rpart"         "spatial"       "splines"       "stats"
## [25] "stats4"        "survival"      "tcltk"         "tools"
## [29] "translations"  "utils"
```

```
lapply(.libPaths(), dir)[[1]]
```

```
##   [1] "abind"                "AMIAS"                "arules"
##   [4] "arulesViz"            "askpass"              "assertthat"
##   [7] "backports"            "base64enc"            "BeSS"
##  [10] "BH"                   "BiocManager"          "BiocVersion"
##  [13] "bitops"               "bootstrap"            "brew"
##  [16] "broom"                "callr"                "car"
```

```
##  [19] "carData"             "caTools"             "cdcsis"
##  [22] "cellranger"          "cghAMIAS"            "cghFLasso"
##  [25] "cli"                 "clipr"               "clisymbols"
##  [28] "cmprsk"              "coin"                "colorspace"
##  [31] "commonmark"          "conquer"             "corpcor"
##  [34] "corrplot"            "covr"                "cowplot"
##  [37] "crayon"              "crosstalk"           "curl"
##  [40] "data.table"          "dendextend"          "DEoptimR"
##  [43] "desc"                "devtools"            "digest"
##  [46] "diptest"             "dlstats"             "DNAcopy"
##  [49] "doParallel"          "dplyr"               "DT"
##  [52] "ellipsis"            "Epi"                 "etm"
##  [55] "evaluate"            "fansi"               "farver"
##  [58] "fastmap"             "fdrtool"             "flexmix"
##  [61] "FNN"                 "forcats"             "foreach"
##  [64] "formatR"             "fpc"                 "freeknotsplines"
##  [67] "fs"                  "fusedBeSS"           "futile.logger"
##  [70] "futile.options"      "gclus"               "gdata"
##  [73] "GeneNet"             "generics"            "genlasso"
##  [76] "GGally"              "ggdendro"            "ggforce"
##  [79] "ggformula"           "ggplot2"             "ggpubr"
##  [82] "ggrepel"             "ggsci"               "ggsignif"
##  [85] "ggstance"            "gh"                  "git2r"
##  [88] "glmnet"              "glmulti"             "glue"
##  [91] "gplots"              "gridExtra"           "gtable"
##  [94] "gtools"              "gurobi"              "haven"
##  [97] "hdi"                 "hexbin"              "highr"
## [100] "hms"                 "htmltools"           "htmlwidgets"
## [103] "httpuv"              "httr"                "igraph"
## [106] "ini"                 "ISLR"                "iterators"
## [109] "jiebaR"              "jiebaRD"             "jpeg"
## [112] "jsonlite"            "kernlab"             "knitr"
## [115] "ks"                  "l0tf"                "labeling"
## [118] "lambda.r"            "lars"                "lassoshooting"
## [121] "later"               "latticeExtra"        "lazyeval"
## [124] "leaflet"             "leaflet.providers"   "libcoin"
## [127] "lifecycle"           "limSolve"            "linprog"
## [130] "lme4"                "lmtest"              "longitudinal"
## [133] "lpSolve"             "magick"              "magrittr"
## [136] "maptools"            "markdown"            "MatrixModels"
## [139] "matrixStats"         "mclust"              "memoise"
## [142] "mime"                "minqa"               "mixtools"
## [145] "modeltools"          "modSaRa"             "mosaic"
## [148] "mosaicCore"          "mosaicData"          "MrBeSS"
## [151] "MTE"                 "multcomp"            "multicool"
## [154] "munsell"             "mvtnorm"             "ncvreg"
## [157] "nlme"                "nloptr"              "numDeriv"
## [160] "openssl"             "openxlsx"            "ordinal"
## [163] "parcor"              "pbkrtest"            "pense"
## [166] "perry"               "PGEE"                "pillar"
## [169] "pkgA"                "pkgbuild"            "pkgconfig"
## [172] "pkgload"             "plogr"               "plotly"
## [175] "plyr"                "PMA"                 "png"
## [178] "polyclip"            "polynom"             "ppls"
```

```
## [181] "prabclus"            "praise"              "prettyunits"
## [184] "processx"            "progress"            "promises"
## [187] "pryr"                "ps"                  "purrr"
## [190] "qap"                 "quadprog"            "quantmod"
## [193] "quantreg"            "R.matlab"            "R.methodsS3"
## [196] "R.oo"                "R.utils"             "R6"
## [199] "randomForest"        "raster"              "rcmdcheck"
## [202] "RColorBrewer"        "Rcpp"                "RcppArmadillo"
## [205] "RcppEigen"           "readr"               "readxl"
## [208] "registry"            "rematch"             "rematch2"
## [211] "remotes"             "reshape"             "reshape2"
## [214] "rex"                 "rio"                 "rlang"
## [217] "rlmDataDriven"       "rmarkdown"           "robustbase"
## [220] "robustHD"            "robustsubsets"       "robustsubsets-master"
## [223] "roxygen2"            "rprojroot"           "rrpack"
## [226] "rrr"                 "RSAVS"               "rstatix"
## [229] "rstudioapi"          "rversions"           "sandwich"
## [232] "scales"              "scalreg"             "scatterplot3d"
## [235] "secure"              "segmented"           "SemiPar"
## [238] "seriation"           "sessioninfo"         "shiny"
## [241] "SIS"                 "slam"                "snow"
## [244] "snowfall"            "sourcetools"         "sp"
## [247] "SparseM"             "spikeslab"           "stringi"
## [250] "stringr"             "sys"                 "testthat"
## [253] "TH.data"             "tibble"              "tidyr"
## [256] "tidyselect"          "tinytex"             "tseries"
## [259] "TSP"                 "tsvd"                "TTR"
## [262] "tweenr"              "ucminf"              "usethis"
## [265] "utf8"                "vcd"                 "vctrs"
## [268] "venn"                "VennDiagram"         "viridis"
## [271] "viridisLite"         "visNetwork"          "wbs"
## [274] "whisker"             "withr"               "wordcloud"
## [277] "xfun"                "XML"                 "xml2"
## [280] "xopen"               "xtable"              "xts"
## [283] "yaml"                "zeallot"             "zip"
## [286] "zoo"
```

`library()` and `require()`

- The main difference between `library()` and `require()` is what happens if a package isn't found.
- While `library()` throws an error, `require()` prints a warning message and returns FALSE

```
library(blah)
```

```
require(blah)
```

```
## Loading required package: blah
```

```
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'blah'
```