Simple Generation of R packages under Windows

The process of building packages is described in the manual **“Writing R extensions” by Friedrich Leisch**, which can be downloaded from CRAN.
Additionally there are important descriptions in the file **readme.packages**, which can be found directly under R\rw1081.

The following description is for building simple packages without C or Fortran code under Windows versions 2000 or XP and the R version 1.8.1.

Shortcut

⇨ Clean the R workspace

```
> rm (list = ls( ))
```

⇨ Put the source code into the R workspace

⇨ Create the package source directory ‘pkgname’ using the function `package.skeleton`:

```
> package.skeleton ( “pkgname”, path=
”U:/target.directory”)
```

⇨ Edit its Description file and its documentation files (.Rd) in the man-subdirectory

⇨ Put the edited package source directory into R\rw1081\src\library

⇨ Run under DOS:
cd C:\R\rw1081\src\gnuwin32
make pkg-pkgname
the package ‘pkgname’ is built and installed in R\rw1081\library

⇨ check the package running
make pkgcheck-pkgname

under the gnuwin32 directory in DOS.

⇨ eliminate possible mistakes in the source directory and run ‘make-pkgname’ and ‘make pkgcheck-pkgname’ again.

⇨ build a zip file from the new package file ‘pkgname’ in R\rw1081\library.

1. Creating the source structure of the package

1.1 Clean the workspace

```
> rm (list = ls( ))
```

This will remove all objects from the search-list which were used in the current R session or saved in an earlier session. This is recommended for more complex R packages.

1.2 Package.skeleton ()

Put all the objects wanted in the package into the R workspace, for example by using copy and paste from txt-files. Data objects in the workspace will also be packaged.

The function package.skeleton ( ) creates a directory containing the basic structure of an R package:

README with a short description of the following steps for creating a package

the DESCRIPTION file of the package

the ‘man’ subdirectory, containing a help file (.Rd) for each object

the ‘R’ subdirectory with the code, each function in a separate R file

the ‘data’ subdirectory containing the datasets as Rda files

the ‘src’ folder for Fortran Code, …….

the function package.skeleton ( ) has the arguments:

name: directory name for your package

list: vector of names of R objects to put in the package

environment: if 'list' is omitted, the contents of this environment are packaged

path: path to put the package directories in

force: If 'FALSE' will not overwrite an existing directory

If the work space was cleaned as described above, package.skeleton ( ) can be used like:

```
> package.skeleton ( “name.of.directory”, path=
”U:/target.directory”)
```

If only a certain number of R objects in the workspace shall be packaged, these can be specified in the list statement as:

```
> package.skeleton ( “name.of.directory”, path=
”U:/target.directory”, list=c("function1", "dataset1",
"foo"))
```

After this, the target directory will contain the new directory with all the files needed for the package. Now you can start with description of the package, documentation of the single functions, data sets, organizing the help pages by filling the forms contained in the directory. The Rd-files can be opened for example with Wordpad.

1.3 Editing the DESCRIPTION file

The DESCRIPTION file has to contain basic information about package name, version, license, author, function in the following format:

```
Package: param
Version: 1.0
Title: Tools for Inference on Ratio of Means
Author: Frank Schaarschmidt
Description: For two samples of independent, continuous, Gaussian
      distributed data with equal variances. Sasabuchi Test and the
      Fieller Confidence interval can be calculated for Inference,
      non-inferiority and equivalence. Approximate sample size for
      Sasabuchitest can be calculated for one-sided inference, non-
inferiority and equivalence.
Maintainer: Frank Schaarschmidt <f.schaarschmidt@web.de>
License: GPL
```

Continuing lines have to start with tab or space. There are some additional, non-optional statements possible. The optional items of the DESCRIPTION file are checked in the procedure of package building. Take care that the package name in the DESCRIPTION file is the same as the directory where the package is stored in.
Take care, that the package name in the DESCRIPTION file and the later package name in the building process is the same.

1.4 Editing Rd files

The ‘man’ subdirectory contains a documentation file (Rd) for each packaged object. The Rd file for a function contains the mandatory items ‘name’, ‘alias’, ‘title’, ‘description’, ‘usage’ in the header and ‘keyword’ in the footer, and further optional items as ‘arguments’, ‘value’, ‘details’, ‘references’, ‘seealso’, ‘examples’ in the body of the file.

\name{} gives the name of the help-file

Under alias you can put names of all objects you want to document in the actual file:

\alias{foo}
\alias{foo.default}
\alias{foo.formula}
...

By filling the { }-brackets of the items

\title{}
\description{}
\usage{}
\arguments{}
\value{}
\details{}
\references{}
\seealso{}
\examples{}
a help page of the known structure is created.

The last item \keyword{} has to contain at least one of the standard keywords listed in the file R_HOME\doc\KEYWORDS.db.
The Statistics-part of the file KEYWORDS:

```
Statistics

      datagen      &     Functions for generating data sets
      distribution &     Probability Distributions and Random Numbers
      univar       &     simple univariate statistics  [!= S]
      htest        &     Statistical Inference
      models       &     Statistical Models
          & regression&   Regression
          & &nonlinear&   Non-linear Regression (only?)
      robust       &     Robust/Resistant Techniques
      design       &     Designed Experiments
      multivariate &     Multivariate Techniques
      ts           &     Time Series
      survival     &     Survival Analysis
      nonparametric&     Nonparametric Statistics [w/o 'smooth']
      smooth       &     Curve (and Surface) Smoothing
           & loess &     Loess Objects
      cluster      &     Clustering
      tree         &     Regression and Classification Trees
      survey       &     Complex survey samples
```

Further keywords can be added, which allow to find function names using the R command help.search (“ “). Only one keyword per line is allowed:

```
\keyword{ htest }
\keyword{ sample.size }
```

Help pages for data sets follow a similar structure, containing the items ‘name’, ‘alias’, ‘docType’, ‘title’, ‘description’, ‘usage’, ‘format’, ‘source’, ‘examples’ and ‘keyword’ where ‘doctype’ is already given as {data} and ‘keyword’ as {datasets}.

1.5 Adding single objects to a package source

If single objects like data sets containing examples shall be added to an existing source structure, the objects themselves and their Rd files have to be saved in the appropriate formats and placed into the appropriate subdirectory.

Saving of data sets can be done by the function save (). The names of the objects to be saved are given as the first argument. In the argument 'file' specify the directory where the objects shall be saved.
For example a data.frame 'rats' is saved as rda-file by

```
> save (rats, file="C:\R\rw1081\package\data\rats.rda")
```

R code can be saved using the function dump(), for example to save a function named 'fie.I' as R file:

```
> dump ("fie.I", file="U:/R/fie.I.R")
```

To build documentation files for single objects, the function prompt ( ) can be used. It creates an appropriate Rd file for the specified object. In the argument 'filename' the directory and filename can be specified, where the Rd file shall be saved.
For example:

```
> prompt(rat.weight, filename="C:/R/rat.weight.rd")
```

2. Building the package

2.1 Software

For building packages from source code, some additional tools need to be installed:

⇨ A package of Unix-tools can be downloaded as zip-file from:

http://www.stats.ox.ac.uk/pub/Rtools/tools.zip

or

http://www.murdoch-sutherland.com/Rtools/tools.zip

⇨ Also needed is a current version of Perl for Windows. It can be downloaded from:

http://activestate.com/Products/ActivePerl/

The current links for download of important tools can be found under Documentation following 'FAQs →R Windows FAQ →3 Packages→3.1 Can I install packages…' on a CRAN website, for example http://cran.at.r-project.org/

For transferring packages as zip-files, for example fil.zip can be used.

2.2 Installation of the additional tools

Both, the folder with the extracted Unix-tools and Perl should be installed at the same drive at the computer, most easily directly on C:\. Take care that all the paths including the path of R do not contain any spaces.
Then the path has to be set via the way Start → Einstellungen → Systemsteuerung → System → Erweitert → Umgebungsvariablen → Systemvariablen. There you can find the variable PATH.
Put

C:\…..\folderwithtools;C:\…\Perl\bin;

in the beginning of the path, before the settings of Windows.

Further remarks on possible difficulties on installation of the tools can be found in the file readme.packages. If the package contains C Code, the path to the additionally needed C-compiler should appear in the second position and the path to Perl in the third position.(For details see the file readme.packages in R\rw1081)

2.3 Creating the package

The directory ‘pkgname’ which was created using package.skeleton () and later on edited by you, should then be placed into the directory R\rw1081\src\library.

The next step has to be done under DOS, for example in the Command Prompt under Start → Programme → Accessories. Giving the commands

```
cd C:\R\rw1081\src\gnuwin32
make pkg-pkgname
```

will create the package and the helpfiles in a new directory called pkgname in the R\rw1081\library, where also the installed standard packages can be found.
This will contain additional files and subdirectories for the help and help.search-functions, INDEX and CONTENTS,…

This package can be used after loading via the R command library(pkgname).

2.4 Checking the package

In this step the package can be checked for mistakes in the documentation files as missing documentations for objects, syntax errors, incomplete or missing items in the Rd files, differences between the objects and their Rd files. Moreover, the R code is checked for syntax errors and whether methods have the same arguments as their generic functions. The examples given in the help files are run. For details see the manual ‘Writing R extensions’.
Checking can be done under DOS giving the commands

```
cd C:\R\rw1081\src\gnuwin32
make pkgcheck-pkgname
```

This is doing the checks and gives error messages. Additionally, a folder ‘check’ is created in the source directory of the package (in \rw1081\src\library) containing files of the error messages.
If there were error messages, the mistakes can be corrected in the original structure and the building procedure can be done again.

2.5 Transfer of packages

To transfer packages to other computers or to CRAN they can be converted into zip-format. If for example filzip 3.0 is installed, this can be done easily by using the right mouse button, then choosing ‘filzip’ and ‘Add to “pkgname”’. This will build a zip-file in R\rw1081\library containing the package directory. This can be transferred to other computers.

The zip-file can be installed in R via choosing ‘Packages’ → ‘Install package(s) from local zip files…’ and then browse to the place where the zip file is located and open it. This will install its content in R\rw1081\library. Then it can be loaded as usual with the R command library (pkgname).