R is Self Documenting

Paul E. Johnson[1] [2]

[1] Department of Political Science

[2] Center for Research Methods and Data Analysis, University of Kansas

2018

Outline

R packages are supposed to be ”self documenting”

- Ask your R (R Core Team, 2017) program which packages are already installed:

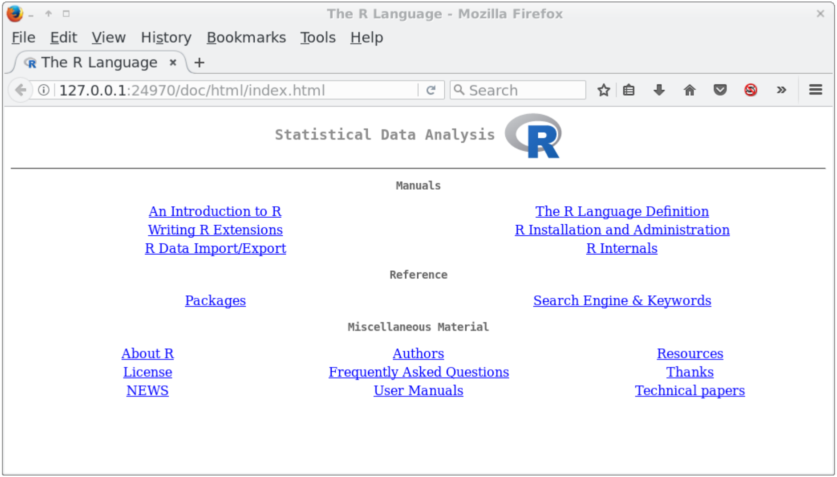
```
library()
```

- Each listed package has manual pages, examples, and usually more.
- Launch a Web browser overview of all of this

```
help.start()
```

Caution: Rstudio will block the browser from starting and will force the page into the small pane on the bottom right.

R provides books, help pages, and vignettes


The R Language - Mozilla Firefox
File Edit View History Bookmarks Tools Help
The R Language
127.0.0.1:24970/doc/html/index.html
Search

Statistical Data Analysis R

Manuals

An Introduction to R
Writing R Extensions
R Data Import/Export
The R Language Definition
R Installation and Administration
R Internals

Reference

Packages
Search Engine & Keywords

Miscellaneous Material

About R
License
NEWS
Authors
Frequently Asked Questions
User Manuals
Resources
Thanks
Technical papers

"An Introduction to R"

1. TOP LEFT: *An Introduction to R*. A Free Book!
2. "Writing R Extensions" and "The R Language Definition" are intended for R developers.
3. "Packages" shows a list of packages currently installed, with links to information about them
4. FAQ (Frequently Asked Questions) bottom center.
5. "User Manuals". A listing of the vignettes distributed with R's core packages.

Command Line Access to Help

- List *functions*, *datasets* and *vignettes* in a package.

```
help(package = "stats")
```

- Read the information on a particular function. The full form of the request would be

```
help(topic = "lm")
```

And you don't have to name the argument, the help function will guess what you mean so that same as

```
help("lm")
```

They noticed people would forget the quotes, so they enriched the help() function to guess the right thing if you leave them out

```
help(lm)
```

Command Line Access to Help ...

- But users said that's tedious, so they made a shortcut "?"

```
?lm
```

- The display of help documents can be delivered either as
  - "text": output inside the R console
  - "html": a web page in a browser
  - "pdf": a pdf document
- See what your system assumes:

```
options("help_type")
```

```
$help_type
NULL
```

- Can be configured
  - within the session, e.g.

```
options("help_type" = "text")
```

Command Line Access to Help ...

  - or as an argument to the help function

    ```
    help("ls", help_type = "text")
    ```

- Your chosen Editor/Graphical user environment will impose its preferences (RStudio interferes with this)

All Help Pages Follow the Same Format

- **Description**
- **Usage**
- **Arguments**: The named arguments
- **Details**: particulars author wants to mention
- **Value**: what you get back
- **Examples**: recommended, usually included.

Example of help

- For example, here's what I see for help on the linear model (lm) function.

```
lm                         package:stats                R
    Documentation

Fitting Linear Models

Description:

   ’lm’ is used to fit linear models. It can be
       used to carry out regression, single stratum
       analysis of variance and analysis of
       covariance (although ’aov’ may provide a
       more convenient interface for these).

Usage:
```

Example of help ...

```
  lm(formula, data, subset, weights, na.action,
     method = "qr", model = TRUE, x = FALSE, y =
     FALSE, qr = TRUE, singular.ok = TRUE,
     contrasts = NULL, offset, ...)

Arguments:

  formula: an object of class ’"formula"’ (or one
     that can be coerced to that class): a
     symbolic description of the model to be
     fitted. The details of model specification
     are given under ’Details’.
```

Example of help ...

```
data: an optional data frame, list or
    environment (or object coercible by
    'as.data.frame' to a data frame) containing
    the variables in the model. If not found in
    'data', the variables are taken from
    'environment(formula)', typically the
    environment from which 'lm' is called.

subset: an optional vector specifying a subset
    of observations to be used in the fitting
    process.
```

Example of help ...

```
weights: an optional vector of weights to be
    used in the fitting process. Should be
    ’NULL’ or a numeric vector. If non-NULL,
    weighted least squares is used with weights
    ’weights’ (that is, minimizing
    ’sum(w*e^2)’); otherwise ordinary least
    squares is used. See also ’Details’.

na.action: a function which indicates what
    should happen when the data contain ’NA’s.
    The default is set by the ’na.action’
    setting of ’options’, and is ’na.fail’ if
    that is unset. The ’factory-fresh’ default
    is ’na.omit’. Another possible value is
    ’NULL’, no action. Value ’na.exclude’ can be
    useful.
```

Example of help ...

```
method: the method to be used; for fitting,
     currently only ’method = "qr"’ is supported;
     ’method = "model.frame"’ returns the model
     frame (the same as with ’model = TRUE’, see
     below). model, x, y, qr: logicals. If ’TRUE’
     the corresponding components of the fit (the
     model frame, the model matrix, the response,
     the QR decomposition) are returned.

singular.ok: logical. If ’FALSE’ (the default in
     S but not in R) a singular fit is an error.

contrasts: an optional list. See the
     ’contrasts.arg’ of ’model.matrix.default’.

```

Example of help ...

```
  offset: this can be used to specify an _a
     priori_ known component to be included in
     the linear predictor during fitting. This
     should be 'NULL' or a numeric vector of
     length equal to the number of cases. One or
     more 'offset' terms can be included in the
     formula instead or as well, and if more than
     one are specified their sum is used. See
     'model.offset'.

  ...: additional arguments to be passed to the
     low level regression fitting functions (see
     below).

Details:
```

Example of help ...

```
Models for 'lm' are specified symbolically. A
    typical model has the form 'response $\sim$
    terms' where 'response' is the (numeric)
    response vector and 'terms' is a series of
    terms which specifies a linear predictor for
    'response'. A terms specification of the
    form 'first + second' indicates all the
    terms in 'first' together with all the terms
    in 'second' with duplicates removed. A
    specification of the form 'first:second'
    indicates the set of terms obtained by
    taking the interactions of all terms in
    'first' with all terms in 'second'. The
    specification 'first*second' indicates the
    _cross_ of 'first' and 'second'. This is the
    same as 'first + second + first:second'.
```

Example of help ...

```
 If the formula includes an 'offset', this is
    evaluated and subtracted from the response.
    If 'response' is a matrix a linear model is
    fitted separately by least-squares to each
    column of the matrix. See 'model.matrix' for
    some further details. The terms in the
    formula will be re-ordered so that main
    effects come first, followed by the
    interactions, all second-order, all
    third-order and so on: to avoid this pass a
    'terms' object as the formula (see 'aov' and
    'demo(glm.vr)' for an example).
```

Example of help ...

```
  A formula has an implied intercept term. To
     remove this use either 'y ∼ x - 1' or 'y ∼ 0
     + x'. See 'formula' for more details of
     allowed formulae. Non-'NULL' 'weights' can
     be used to indicate that different
     observations have different variances (with
     the values in 'weights' being inversely
     proportional to the variances); or
     equivalently, when the elements of 'weights'
     are positive integers w_i, that each
     response y_i is the mean of w_i unit-weight
     observations (including the case that there
     are w_i observations equal to y_i and the
     data have been summarized).
```

Example of help ...

```
  ’lm’ calls the lower level functions ’lm.fit’,
     etc, see below, for the actual numerical
     computations. For programming only, you may
     consider doing likewise.

  All of ’weights’, ’subset’ and ’offset’ are
     evaluated in the same way as variables in
     ’formula’, that is first in ’data’ and then
     in the environment of ’formula’.

Value:

  ’lm’ returns an object of ’class’ ’"lm"’ or for
     multiple responses of class ’c("mlm", "lm")’.

```

Example of help ...

```
The functions 'summary' and 'anova' are used to
    obtain and print a summary and analysis of
    variance table of the results. The generic
    accessor functions 'coefficients',
    'effects', 'fitted.values' and 'residuals'
    extract various useful features of the value
    returned by 'lm'.

An object of class '"lm"' is a list containing
    at least the following components:

coefficients: a named vector of coefficients
residuals: the residuals, that is response minus
    fitted values.
fitted.values: the fitted mean values.
rank: the numeric rank of the fitted linear
    model.
```

Example of help ...

```
weights: (only for weighted fits) the specified
    weights.
df.residual: the residual degrees of freedom.
call: the matched call.
terms: the 'terms' object used.
contrasts: (only where relevant) the contrasts
    used.
xlevels: (only where relevant) a record of the
    levels of the factors used in fitting.
offset: the offset used (missing if none were
    used).
y: if requested, the response used.
x: if requested, the model matrix used.
model: if requested (the default), the model
    frame used.
```

Example of help ...

```
  na.action: (where relevant) information returned
     by 'model.frame' on the special handling of
     'NA's.

  In addition, non-null fits will have components
     'assign', 'effects' and (unless not
     requested) 'qr' relating to the linear fit,
     for use by extractor functions such as
     'summary' and 'effects'.

Using time series:
```

Example of help ...

```
Considerable care is needed when using 'lm' with
     time series. Unless 'na.action = NULL', the
     time series attributes are stripped from the
     variables before the regression is done.
     (This is necessary as omitting 'NA's would
     invalidate the time series attributes, and
     if 'NA's are omitted in the middle of the
     series the result would no longer be a
     regular time series.) Even if the time
     series attributes are retained, they are not
     used to line up series, so that the time
     shift of a lagged or differenced regressor
     would be ignored. It is good practice to
     prepare a 'data' argument by
     'ts.intersect(..., dframe = TRUE)', then
     apply a suitable 'na.action' to that data
     frame and call 'lm' with 'na.action = NULL'
```

Example of help ...

```
     so that residuals and fitted values are time
     series.

Note:

  Offsets specified by 'offset' will not be
     included in predictions by 'predict.lm',
     whereas those specified by an offset term in
     the formula will be.

Author(s):

  The design was inspired by the S function of the
     same name described in Chambers (1992). The
     implementation of model formula by Ross
     Ihaka was based on Wilkinson & Rogers (1973).
```

Example of help ...

```
References:

  Chambers, J. M. (1992) _Linear models._ Chapter
     4 of _Statistical Models in S_ eds J. M.
     Chambers and T. J. Hastie, Wadsworth &
     Brooks/Cole.

  Wilkinson, G. N. and Rogers, C. E. (1973)
     Symbolic descriptions of factorial models
     for analysis of variance. _Applied
     Statistics_, *22*, 392-9.

See Also:

  'summary.lm' for summaries and 'anova.lm' for
     the ANOVA table; 'aov' for a different
     interface.
```

Example of help ...

```
The generic functions ’coef’, ’effects’,
    ’residuals’, ’fitted’, ’vcov’. ’predict.lm’
    (via ’predict’) for prediction, including
    confidence and prediction intervals;
    ’confint’ for confidence intervals of
    _parameters_.

’lm.influence’ for regression diagnostics, and
    ’glm’ for *generalized* linear models.

The underlying low level functions, ’lm.fit’ for
    plain, and ’lm.wfit’ for weighted regression
    fitting.
```

Example of help ...

```
  More 'lm()' examples are available e.g., in
     'anscombe', 'attitude', 'freeny',
     'LifeCycleSavings', 'longley', 'stackloss',
     'swiss'. 'biglm' in package 'biglm' for an
     alternative way to fit linear models to
     large datasets (especially those with many
     cases).

Examples:

  require(graphics)
   ## Annette Dobson (1990) "An Introduction to
      Generalized Linear Models".
   ## Page 9: Plant Weight Data.
  ctl <-
     c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14
```

Example of help ...

```
trt <-
   c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69
group <- gl(2,10,20, labels=c("Ctl","Trt"))
weight <- c(ctl, trt)
lm.D9 <- lm(weight $\sim$ group)
lm.D90 <- lm(weight $\sim$ group - 1) # omitting
   intercept
anova(lm.D9)
summary(lm.D90)
opar <- par(mfrow = c(2,2), oma = c(0, 0, 1.1,
   0))
plot(lm.D9, las = 1) # Residuals, Fitted, ...
par(opar)
## less simple examples in "See Also" above
```

How I read a help page

1. Look at the top to figure out
    1. what is this supposed to do? and
    2. what information do I need to give it?
2. Run the example to see if I can understand what it does
3. If still interested, go back to top
    1. Look more carefully at the arguments
    2. Study the return "**Value**"
    3. Look for the "**Details**" heading.

Run the Examples described on the help page

- Runs the entire example

```
> example(someFunction)
```

- If you use Emacs as your editor, there is a handy feature to run a help example line-by-line.
- One reason why I'm reluctant about “RStudio” is that access to help and examples is made more difficult.

Vignette: An essay with a package

- A vignette is a (hopefully) “more readable” discussion of a package’s features
- Some vignettes are quite excellent!
- Load the `rpart` package (which everybody does have because it is provided with R).

```
library(rpart)
```

- Vignettes are listed with the documentation. Toward the bottom of the help output for a package

```
help(package = "rpart")
```

```
Further information is available in the following vignettes in
     directory ’/usr/lib/R/library/rpart/doc’:

longintro: Introduction to Rpart (source, pdf)

usercode: User Written Split Functions (source, pdf)
```

Vignette: An essay with a package ...

- Clickable links to vignette in top of help.start() , after navigating to packages, and finding rpart
- loadable by name with the function vignette() . This is a package survey:

```
vignette("longintro")
```

Your system does not have ”help” for packages that are not installed

- `help` (or “?”) looks in your current session for functions in loaded packages.
- `help.search` looks in installed packages (“??” is shortcut).

  ```
  help.search("aov")
  ```

  ```
  ??aov
  ```

  Note, oddly, that quotation marks are needed within `help.search` but not with `??`
- `RSiteSearch(”aov”)` looks on the main R website for items items related to the aov function.

Reminder: When you ask for help, provide. . .

1. Calm down. Consider the possibility that you've corrupted the R session. Close R, re-start.
    1. Make sure no old failed session was reloaded. " `ls()` " should show no old objects.
    2. This will delete those objects " `rm(list = ls())` ".
    3. Then try again to run your code
2. If you do write to ask for help, don't forget `sessionInfo()` output.

```
sessionInfo()
```

```
R version 3.4.4 (2018-03-15)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 18.04 LTS

Matrix products: default
BLAS: /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.7.1
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.7.1

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
      LC_TIME=en_US.UTF-8
 [4] LC_COLLATE=en_US.UTF-8     LC_MONETARY=en_US.UTF-8
      LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                          LC_ADDRESS=C
```

Even Better: an MRE

3. MRE: Minimum Reproducible Example. The smallest set of code that reproduces the problem you are concerned about.

- Produce a small, clear example of the problem you are trying to solve.
- If you do that, the chances are good you will see what you were doing wrong (running commands out of order, depending on the wrong variables, etc).
- If you share the MRE to people when you ask for help, they are much more likely to take you seriously.

References

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Session

```
sessionInfo()
```

```
R version 3.4.4 (2018-03-15)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 18.04 LTS

Matrix products: default
BLAS: /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.7.1
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.7.1

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
     LC_TIME=en_US.UTF-8
 [4] LC_COLLATE=en_US.UTF-8     LC_MONETARY=en_US.UTF-8
     LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8
    LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  base

other attached packages:
[1] rpart_4.1-13
```

Session ...

```
loaded via a namespace (and not attached):
[1] compiler_3.4.4 tools_3.4.4
```