First R-02: R is Self Documenting

Paul E. Johnson¹ ²

¹Department of Political Science
²Center for Research Methods and Data Analysis, University of Kansas

2015
R packages are supposed to be "self documenting"

- Ask your system which packages are already installed:
  > library()

- Each listed package has manual pages, examples, and usually more.

- Launch a browser overview of all of this
  > help.start()
R provides books, help pages, and vignettes

- Result from `help.start()` appears in a web browser, if R can find one in your computer.

```
> help.start()
```
The Book at the top left: *An Introduction to R*. I learned R from that. You should too.

“Writing R Extensions” and “The R Language Definition” are intended for R developers.

Frequently Asked Questions.

“Packages” shows a list of packages currently installed, with links to information about them.

“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.
  
  ```r
  > help(package = "stats")
  ```

- Read the information on a particular function. The full form of the request would be
  
  ```r
  > help(topic = "lm")
  ```

  And you don’t have to name the argument, the help function will guess what you mean so that same as
  
  ```r
  > 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
  
  ```r
  > help(lm)
  ```
But users said that’s tedious, so they made a shortcut “?”

> ?lm
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:

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

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'.

**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.
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:**
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'.
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).
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).

'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")'.

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.
- weights: (only for weighted fits) the specified weights.
Example of help ...

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.
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:
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' so that residuals and fitted values are time series.

Note:
Example of help ...

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).

References:


Example of help ... 

See Also:

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

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.
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:

```r
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)
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
```
Example of help ...

```
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
All Help Pages Follow the Same Format

- General description
- Usage
- Definition of legal arguments
- Details: particulars author wants to mention
- Value: what you get back
- Example: recommended, usually included.
Run the Examples described on the help page

- Run the example from the command line
  ```r
  > example(someFunction)
  ```
- If you use Emacs as your editor, there is a handy feature to run a help example line-by-line.
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.
All of your packages are not loaded automatically

- As we saw in the `sessionInfo()`, just a few packages are loaded when you start R.
- Use the `library()` function to load them when you want to use them

```r
> library(rockchalk)
```

- But loading full package not absolutely necessary. Can access functions from a package with the "::" notation. Example

```r
> data(swiss)
> rockchalk::summarize(swiss)
```
All of your packages are not loaded automatically ...

<table>
<thead>
<tr>
<th>$\text{Numerics}$</th>
<th>Agriculture</th>
<th>Catholic</th>
<th>Education</th>
<th>Examination</th>
<th>Fertility</th>
<th>Infant. Mortality</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td>1.2</td>
<td>2.1</td>
<td>1.0</td>
<td>3</td>
<td>35</td>
<td>10.8</td>
</tr>
<tr>
<td>25%</td>
<td>35.9</td>
<td>5.2</td>
<td>6.0</td>
<td>12</td>
<td>65</td>
<td>18.1</td>
</tr>
<tr>
<td>50%</td>
<td>54.1</td>
<td>15.1</td>
<td>8.0</td>
<td>16</td>
<td>70</td>
<td>20.0</td>
</tr>
<tr>
<td>75%</td>
<td>67.7</td>
<td>93.1</td>
<td>12.0</td>
<td>22</td>
<td>78</td>
<td>21.7</td>
</tr>
<tr>
<td>100%</td>
<td>89.7</td>
<td>100.0</td>
<td>53.0</td>
<td>37</td>
<td>92</td>
<td>26.6</td>
</tr>
<tr>
<td>$\text{Mean}$</td>
<td>50.7</td>
<td>41.1</td>
<td>11.0</td>
<td>16</td>
<td>70</td>
<td>19.9</td>
</tr>
<tr>
<td>$\text{SD}$</td>
<td>22.7</td>
<td>41.7</td>
<td>9.6</td>
<td>8</td>
<td>12</td>
<td>2.9</td>
</tr>
<tr>
<td>$\text{Var}$</td>
<td>515.8</td>
<td>1739.3</td>
<td>92.5</td>
<td>64</td>
<td>156</td>
<td>8.5</td>
</tr>
<tr>
<td>$\text{NA's}$</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>$\text{N}$</td>
<td>47.0</td>
<td>47.0</td>
<td>47.0</td>
<td>47</td>
<td>47</td>
<td>47.0</td>
</tr>
</tbody>
</table>

$\text{Factors}$

NULL
A vignette is supposed to be a “human readable” discussion of a package’s features.

Some are quite excellent!

Vignettes are listed at the end of `help(package = "whatever")`.

Clickable links in `help.start()` are loadable by name with the function `vignette()`. This is a package survey:

```r
> vignette("rockchalk")
```

I urge everybody to read my opinions in

```r
> vignette("Rstyle")
```
Your system does not have "help" for packages that are not installed

- `help()` looks in your current session for functions in loaded packages
- `help.search()` looks in installed but unloaded packages (same as "??" shortcut)
- `RSiteSearch("someFunction")` looks on the main R website (helps sometimes).
Reminder: When you ask for help

1. Provide the output of `sessionInfo()`. For example, I see

```r
R version 3.1.2 (2014-10-31)
Platform: x86_64-pc-linux-gnu (64-bit)
locale:
[1] LC_CTYPE= en_US.UTF-8 LC_NUMERIC=C
  LC_TIME=en_US.UTF-8 LC_COLLATE=en_US.UTF-8
  LC_MONETARY=en_US.UTF-8
[6] LC_MESSAGES= en_US.UTF-8 LC_PAPER=en_US.UTF-8
  LC_NAME=C LC_ADDRESS=C
  LC_TELEPHONE=C

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

loaded via a namespace (and not attached):
[1] acepack_1.3-3.3  car_2.0-25  cluster_2.0.3
colorspace_1.2-6  digest_0.6.8
When you ask for help

2. Provide 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.
- Before asking for help, consider the possibility that you’ve corrupted the R session. Close R, re-start, and re-run your small working example.