# First R-02

Paul E. Johnson[1] [2]

[1]Department of Political Science

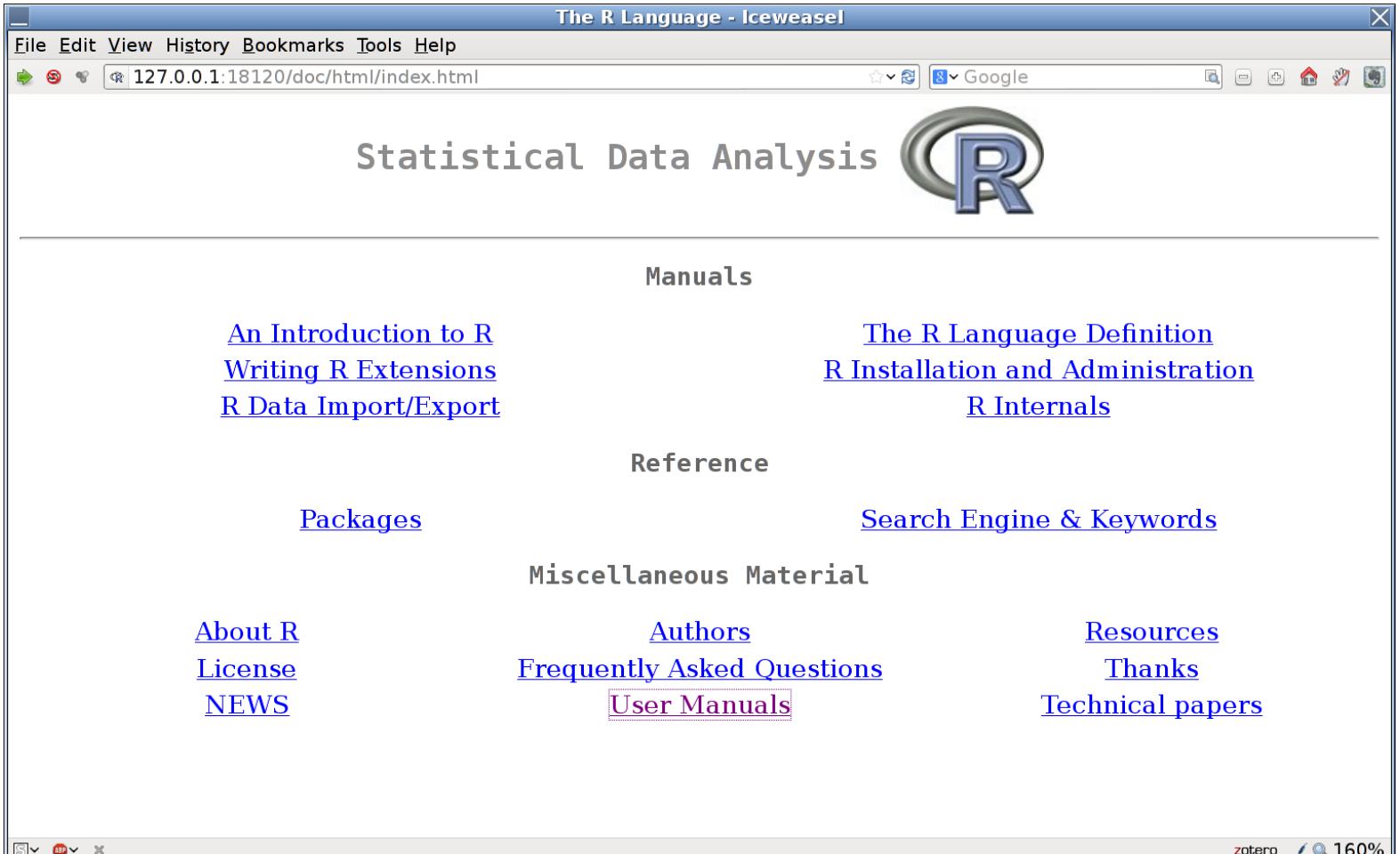
[2]Center for Research Methods and Data Analysis, University of Kansas

2013

# R provides books, help pages, and vignettes

- Get the big overview to documents with

```
> help.start()
```



# Help and Examples for functions

- You can point-and-click though that Web browser
- Or you can ask for same information at R command line. 2 equivalent methods to ask about a particular function "someFunction"

```
> ?someFunction
> help(someFunction)
```

Note: no quotation marks are necessary around the function's name.

# Example of help

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

```
> ?lm
```

```
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'.

  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).

'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'.

# Example of help ...

```
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:
```

# 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' 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:

  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.
```

# 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.
```

# 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
      )
  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
```

# Run the Examples described on the help page

- Scroll to the bottom of the help page. See the example usage? Gaze in wonder at it.
  Get out of the help page (hit "q") and then
- Run the example

```
> example(someFunction)
```

# How to read a help page

1. Get a general idea of what the function does
2. Go to the bottom for the example usage.
3. If still interested, go back to top
    1. Scan the arguments (the variables you specify to use the function)
    2. Look for the "Value" heading. That's a description of what you get back from the function
    3. Look for the "Details" heading.

# Packages: Installable Globs of Goodness

Explaining the next part requires a little detour from my theme.

- The CRAN repository offers 1000s of package, some good some bad
  - Our mirror of CRAN is `http://rweb.quant.ku.edu/cran`
- Other servers (Bioconductor, Omega) offer 100s of other packages
- R is provided with a set of "recommended packages"

# Package Survey

- See a giant list of packages that exist on CRAN

```
> giantList <- available.packages()
> row.names(giantList)
```

- Generally, I Web browse a CRAN server.
  - Mirror list on `http://r-project.org` (look left for CRAN link)
  - KU mirror: `http://rweb.quant.ku.edu/cran`
- All good KU students install the package "rockchalk" and look at its beautiful vignettes.

# However, your system does not have ”help” for packages that are not installed

- ?someFunction (help(“someFunction”) looks in your system for functions in packages that are both installed and loaded
- ??someFunction (help.search(“someFunction”)) looks in packages that are installed, even if not loaded
- RSiteSearch(“someFunction”) looks on the main R website (some wrinkles).

# Install and Update packages

- Install a package (example: "lme4") and all dependencies from a CRAN server

```
> install.packages(c("lme4"), dep = TRUE)
```

- Faster to specify a particular repository

```
> install.packages(c("lme4"), dep = TRUE, repos="http://
    /rweb.quant.ku.edu/cran")
```

- Check for updates

```
> update.packages(checkBuilt = TRUE)
```

Caution: the operating system has a list of places where it will install packages (Run ".libPaths()" to see it). If you are logged in as an "administrator", packages will be installed into the system libraries. Otherwise, they will be installed in a folder for only the current user.

# Interact with packages

- Packages installed are linked to the help.start() page
- Same: list all packages that are installed now in R:

```
> library()
```

- Read about a package, get a list of all functions & features

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

- Synonym: The "old way" I used to teach still works, but maybe not for long

```
> library(help = "stats")
```

# Load a package

- When R starts, it "attaches" some packages, so their functions are easily accessed. But not all.
- Make a package's functions immediately accessible.

```
> library(lme4)
```

- Synonym `require(lme4)`
- After that, we can use functions easily, for example, to read their help pages

```
> ?lmer
> ?glmer
```

# Package Namespaces: Using non-attached packages

- Without running library(), functions are still accessible with a namespace prefix

```
> lme4::glmer()
```

That's generally irrelevant to elementary R usage, but is becoming more noticeable in examples and help pages.

  - The "namespace" idea is increasingly popular in computer programming, part of an widespread emphasis on "disambiguation"

# These things called "vignettes"

- 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()`
- loadable by name with the function `vignette("rockchalk")`

# When you ask for help

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

```
> sessionInfo()
```

```
R version 3.0.0 (2013-04-03)
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=C
     LC_NAME=C                  LC_ADDRESS=C
     LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

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

loaded via a namespace (and not attached):
[1] tools_3.0.0
```

# When you ask for help

2. Provide the smallest set of code that reproduces the problem you are concerned about.

- It is tempting to just copy 100s of lines of disorganized code and hope somebody else will wade through it, but don't.
- Produce a small, clear example of the problem you are trying to solve.
- Never write to somebody and ask for help unless you close R, re-start, and re-produce the same problem with your clear example script.

# Help and Examples For functions