Managing Development
Writing the Manual

DataSHIELD manuals are generated using roxygen.

1. What is roxygen?

The roxygen package makes documenting code easier. It complements the standard way of creating manual pages in R (that is, by writing .Rd files into a man/ directory). However, roxygen generates these .Rd files automatically from comments written into the header of your .R scripts.

Process
(1) Write comments in the header of your R scripts.

(2) Within R, call `roxygenise()` to generate the manuals.

Install roxygen
roxygen is actually provided by the `roxygen2` package. Install in R with:

```
> install.packages('roxygen2')
```

2. roxygen header

By adding a specially formatted header into your R scripts, manual pages can be generated automatically by roxygen.

Each line of the roxygen header must start with #'

Roxygen comments use a simple markup for the different sections of the manual page.

Sections are identified by tags such as `@title`, `@description`, `@example`. (Note that to literally write an @ symbol you have use @@.)

2.1 Tags

`#' @title`
The title of the manual, which is displayed at the top of each manual page.

`#' @description`
A brief description of what the function does.

`#' @details`
If there is more to say about the function write your explanation in the details section. You can go into as much detail as you like and this can be a long description about how the function operates. The details section will appear after the list of the function's arguments.

`#' @param`
For each of the arguments that the function takes, include a separate `@param` tag that describes the argument and what kind of object is expected to be passed to it.

`#' @return`
For each object returned by the function, include a separate `@return` tag that describes the object.

`#' @author`
Add your name as an author. Names should be separated by a semicolon.

`#' @export`
The `@export` tag is used by roxygen to generate the `NAMESPACE` file. Include `@export` if the function needs to be available to users (i.e. it is not one of the 'internal' functions).

`#' @example`
Provide one or more examples of the function to demonstrate its usage and results. These should be examples that will work with the test data provided in the DataSHIELD testing VMs. The example should therefore give instructions for logging in to the VMs and doing any preparatory work necessary to demonstrate the function in question.

`#' @seealso`
Link to other useful resources. Use the formatting markup (below) to add weblinks (e.g. `\url{}`) and links to other manual pages (e.g. `\code{\link{function}}`)

`#' @aliases`
You can provide a list of (space separated) different keywords through which users can find the documentation when they use '?'.

`#' @concept`
Add extra keywords through which users can find the documentation when they use `help.search()`.

2.2 Formatting

You can add formatting to any text you write in the roxygen header using markup similar to LaTeX.

Text
```
\emph{italics}
\strong{bold}
\code{code}
\pkg{package_name}
\eqn{a + b}: inline eqution
\deqn{a + b}: display (block) equation
```

Numbered list:
```
#' \enumerate{
#'   \item First item
#'   \item Second item
#' }
```

Bullet list:
```
#' \itemize{
#'   \item First item
#'   \item Second item
#' }
```

Named list:
```
#' \describe{
#'   \item{One}{First item}
#'   \item{Two}{Second item}
#' }
```

Links
To other documentation:
`\code{\link{function}}`: function in this package
`\code{\link[MASS]{stats}}`: function in another package
`\link[=dest]{name}`: link to dest, but show name

To the web:
```
\url{http://datashield.ac.uk}
\href{http://datashield.ac.uk}{DataSHIELD Website}
\email{datashield-dev@@bristol.ac.uk}
```

2.3 Further info

For more information about roxygen, you can access a 'vignette' which is part of the package.

```
> vignette('roxygen2')
```

3. Using roxygen

3.1 Generating documentation from R

Run the `roxygenise()` function to regenerate documentation.

If you are working within the root of an R package folder, the you can run `roxygenise()` without any arguments. It will automatically take the headers from the scripts in the R/ folder and create the appropriate `NAMESPACE` and .Rd files in the proper places (i.e. .Rd files in a man/ folder).

Otherwise, you can pass the path to the package folder as an argument. For example, to generate the documentation for dsBaseClient:

```
> library(roxygen2)
>  roxygenise('path/to/dsBaseClient')
```

Alternatively, you can use the devtools package to call `roxygenise()` indirectly.

```
> library(devtools)
> devtools::document('path/to/dsBaseClient')
```

3.2 Generating documentation from Rstudio

The ability to use roxygen is integrated into Rstudio. You may want to edit some of the default settings:

From within Rstudio:
(1) Go to the 'Build' tab and select 'Configure Build Tools'.

(2) In the 'Project Options' window select 'Package' from the drop-down list of 'Package build tools'.

(3) Enable the option 'Generate documentation with Roxygen'.

(4) Enable all the options in the new window titled 'Roxygen options'

(5) Click the OK button on all windows to finish.

If at any time you want to regenerate the documentation manually, press Ctrl + Shift + D.

3.3 Committing changes

Making separate commits for code changes and documentation regeneration creates a slightly cleaner history. But equally, bundling together code with corresponding documentation changes is not problematic.