Customizing and extending Emacs Speaks Statistics

Frédéric Santos[1]

January 15, 2021

[1]frederic.santos@u-bordeaux.fr

A short tutorial for customizing ESS

Target audience

This tutorial is for R-users beginning to use ESS. We only assume that you have mild familiarity with Emacs, and that you know how to customize Emacs by filling your `.emacs` or `init.el` file.

Goals

- Helping newcomers and beginners to have a more friendly and efficient setup
- Helping users migrating from other IDEs to keep behaviours and habits they are used to
- Showcasing some other Emacs packages that play well with ESS and may improve your coding experience

These slides are a companion for the video tutorial available at . . .

A word about use-package

In this document, several pieces of Emacs Lisp code will be proposed so that you can use them in your init file.

It is assumed that you use use-package for your init file: the Emacs Lisp code can be adapted in a straightforward manner if you do not use it.

As a reminder, this is the minimal code to add in your init file so as to use use-package, once it has been installed:

```
;; Make sure that use-package is installed:
(unless (package-installed-p 'use-package)
  (package-refresh-contents)
  (package-install 'use-package))
;; Load use-package:
(eval-when-compile
  (require 'use-package))
```

Documentation popups I

`company-quickhelp` allows for documentation popups, e.g. to further describe function arguments.

Figure 1: Documentation popups with `company-quickhelp`.

Documentation popups II

The minimal elisp code to add to your init file is straightforward:

```
(use-package company-quickhelp
  :ensure t
  :config
  ;; Load company-quickhelp globally:
  (company-quickhelp-mode)
  ;; Time before display of documentation popup:
  (setq company-quickhelp-delay 0.3))
```

By default, the documentation popup is shown automatically. You can adjust the time before the popup shows up by customizing the variable company-quickhelp-delay.

Code snippets

`yasnippet` is an Emacs package allowing for the expansion of whole pieces of code you often use (*snippets*) from one given abbreviation.

Key features of `yasnippet`

- All code snippets are stored as plain-text files in one given directory, so that they are easy to share with other people, and can be easily version controlled.

- As a corollary, it is also easy to retrieve and use large collection of snippets already available online. For instance, Andrea Crotti maintains a great collection available at
  `https://github.com/AndreaCrotti/yasnippet-snippets`.

- Although we only demonstrate its use within ESS and R here, note that `yasnippet` is not an R-specific solution, and that you can use it for any other programming language.

Setting up yasnippet I

To set up yasnippet, proceed through the following steps:

1. Create a directory snippets/ at some convenient location, and add a subfolder ess-r-mode/ in this directory.
2. Add the minimal following code in your init file:

```
(use-package yasnippet
  :ensure t
  :config
  ;; Indicate the directory containing your snippets:
  (setq yas-snippet-dirs '("path/to/your/snippets"))
  ;; Load your snippets on startup:
  (yas-reload-all)
  ;; Turn on yasnippet (minor) mode when editing R files:
  (add-hook 'ess-r-mode-hook #'yas-minor-mode))
```

Setting up `yasnippet` II

3. You can now fill your `snippets/ess-r-mode/` directory with your
   own snippets. For instance, create a file `function` (without any
   extension) in this directory, with the following contents:

```
#name : function
#key : fun
# --
${1:name} <- function(${2:args}) {
    ${3:body}
}
```

   Each snippet has a unique `name`, and can be triggered by typing a
   given `key` (followed by `TAB`). As we will see later on, the present
   snippet allows for the expansion of a template for defining new R
   functions more easily. The `yasnippet` manual gives more details
   about the expected syntax to define your own code snippets:
   `http://joaotavora.github.io/yasnippet/`.

Setting up yasnippet III

④ Now your snippets directory should look like:

```
└── snippets
    └── ess-r-mode
        └── function
```

Feel free to add or retrieve (a lot!) more snippets, i.e. to add more template files within the ess-r-mode sub-directory.

Using `yasnippet` in an ESS[R] buffer

While you are editing an R source file with ESS, each snippet can be triggered by typing its key and then pressing TAB. You can then navigate through the placeholders of the expanded template by pressing TAB again.

For instance, with our previously defined snippet, typing `fun` followed by TAB will expand the full `function` template; you will then be able to specify easily a value for each of the three placeholders (the function's name, its `args` and `body`).

Note that `yasnippet` has a short video tutorial, available at https://www.youtube.com/watch?v=ZCGmZK4V7Sg.