1 General Information: README

This is the README file for the distribution of ESS version 18.10.2

ESS is a GNU Emacs and XEmacs mode for interactive statistical programming and data analysis. Languages supported: the S family (S, S-PLUS and R), SAS, BUGS/JAGS and Stata. ESS grew out of the desire for bug fixes and extensions to S-mode and SAS-mode as well as a consistent union of their features in one package.

Installation instructions are provided in sections for both Unix and Windows; see below.

The current development team is led by Martin Maechler since August 2004. Former project leader A.J. (Tony) Rossini (`rossini@blindglobe.net`) did the initial port to XEmacs and has been the primary coder. Martin Maechler (`maechler@stat.math.ethz.ch`) and Kurt Hornik (`Kurt.Hornik@R-project.org`) have assisted with the S family and XLispStat. Stephen Eglen (`stephen@gnu.org`) has worked mostly on R support. Richard M. Heiberger (`rmh@temple.edu`) has assisted with S/S-PLUS development for Windows. Richard and Rodney A. Sparapani (`rsparapa@mcw.edu`) have done much of the work improving SAS batch and interactive support. Rodney has also extended ESS to support BUGS/JAGS and has an interest in improving Stata support.

We are grateful to the previous developers of S-mode (Doug Bates, Ed Kademan, Frank Ritter, David M. Smith), SAS-mode (Tom Cook) and Stata-mode (Thomas Lumley).

1.1 License

1.2 Installation

ESS supports GNU Emacs versions 24.3 and newer.

ESS is most likely to work with current/recent versions of the following statistical packages: R/S-PLUS, SAS, Stata, OpenBUGS and JAGS.

To build the PDF documentation, you will need a version of TeX Live or texinfo that includes texi2dvi.

There are two main methods used for installing ESS. You may install from a third-party repository or from source code. Once you install it, you must also activate or load ESS in each Emacs session, though installation from a third-party repository likely takes care of that for you. See Section 1.5 [Activating and Loading ESS], page 2, for more details.

1.3 Installing from a third-party repository

ESS is packaged by many third party repositories. Many GNU/Linux distributions package it, usually with the name "emacs-ess" or similar.

ESS is also available through Milkypostmans Emacs Lisp Package Archive (MELPA), a popular repository for Emacs packages. Instructions on how to do so are found on MELPA's website (`https://melpa.org/`). MELPA also hosts MELPA-stable with stable ESS builds. You may choose between MELPA with the latest and greatest features (and bugs) or MELPA-stable, which may lag a bit behind but should be more stable.

After installing, users should make sure ESS is activated or loaded in each Emacs session. See Section 1.5 [Activating and Loading ESS], page 2. Depending on install method, this may be taken care of automatically.

1.4 Installing from source

Stable versions of ESS are available at the ESS web page (`https://ess.r-project.org`) as a .tgz file or .zip file. ESS releases are GPG-signed, you should check the signature by downloading the accompanying `.sig` file and doing:

```
gpg --verify ess-18.10.tgz.sig
```

Alternatively, you may download the git repository. ESS is currently hosted on Github: `https://github.com/emacs-ess/ESS`. `git clone https://github.com/emacs-ess/ESS.git` will download it to a new directory `ESS` in the current working directory.

We will refer to the location of the ESS source files as `/path/to/ESS/` hereafter.

After installing, users should make sure they activate or load ESS in each Emacs session, see Section 1.5 [Activating and Loading ESS], page 2,

Optionally, compile elisp files, build the documentation, and the autoloads:

```
cd /path/to/ESS/
make
```

Without this step the documentation, reference card, and autoloads will not be available. Uncompiled ESS will also run slower.

Optionally, you may make ESS available to all users of a machine by installing it site-wide. To do so, run `make install`. You might need administrative privileges:

```
make install
```

The files are installed into `/usr/share/emacs` directory. For this step to run correctly on macOS, you will need to adjust the `PREFIX` path in `Makeconf`. The necessary code and instructions are commented in that file.

1.5 Activating and Loading ESS

After installing ESS, you must activate or load it each Emacs session. ESS can be autoloaded, and if you used a third-party repository (such as your Linux distribution or MELPA) to install, you can likely skip this section and proceed directly to Section 1.6 [Check Installation], page 3, Otherwise, you may need to add the path to ESS to `load-path` with:

```
(add-to-list 'load-path "/path/to/ESS/lisp")
```

You then need to decide whether to take advantage of deferred loading (which will result in a faster Emacs startup time) or require ESS when Emacs is loaded. To autoload ESS when needed (note that if installed from source, you must have run `make`):

```
(load "ess-autoloads")
```

To require ESS on startup, you can either put

```
(require 'ess-site)
```

or

```
(require 'ess-r-mode)
```

In your configuration file, depending on whether you want all ESS features or only R related features.

1.6 Check Installation

Restart Emacs and check that ESS was loaded from a correct location with `M-x ess-version`.

1.7 Starting an ESS process

To start an S session on Unix or on Windows when you use the Cygwin bash shell, simply type *`M-x S RET`*.

To start an S session on Windows when you use the MSDOS prompt shell, simply type *`M-x S+6-msdos RET`*.

1.8 Current Features

- Languages Supported:
  - S family (R, S, and S+ AKA S-PLUS)
  - SAS
  - BUGS/JAGS
  - Stata
  - Julia
- Editing source code (S family, SAS, BUGS/JAGS, Stata, Julia)
  - Syntactic indentation and highlighting of source code
  - Partial evaluation of code
  - Loading and error-checking of code
  - Source code revision maintenance
  - Batch execution (SAS, BUGS/JAGS)
  - Use of imenu to provide links to appropriate functions
- Interacting with the process (S family, SAS, Stata, Julia)
  - Command-line editing
  - Searchable Command history

    - Command-line completion of S family object names and file names
    - Quick access to object lists and search lists
    - Transcript recording
    - Interface to the help system
- Transcript manipulation (S family, Stata)
    - Recording and saving transcript files
    - Manipulating and editing saved transcripts
    - Re-evaluating commands from transcript files
- Interaction with Help Pages and other Documentation (R)
    - Fast Navigation
    - Sending Examples to running ESS process.
    - Fast Transfer to Further Help Pages
- Help File Editing (R)
    - Syntactic indentation and highlighting of source code.
    - Sending Examples to running ESS process.
    - Previewing

1.9 New Features

Bug Fixes in 18.10.2:

- ESS[R] Fix namespace evaluation in non-installed packages. Evaluation is directed into GlobalEnv as originally intended.
- `Makefile` fixes, notably for `make install` and including full docs in the tarballs.

Bug Fixes in 18.10-1:

- New functions `ess-eval-line-visibly-and-step` (*C-c C-n* and `ess-eval-region-or-line-visibly-and-step` (*C-RET*) which behave as the old versions of `ess-eval-line-and-step` and `ess-eval-region-or-line-and-step`.

Changes and New Features in 18.10:

- This is the last release to support Emacs older than 25.1. Going forward, only GNU Emacs 25.1 and newer will be supported. Soon after this release, support for older Emacs versions will be dropped from the git master branch. Note that MELPA uses the git master branch to produce ESS snapshots, so if you are using Emacs < 25.1 from MELPA and are unable to upgrade, you should switch to MELPA-stable.
- ESS now displays the language dialect in the mode-line. So, for example, R buffers will now show ESS[R] rather than ESS[S].
- The ESS manual has been updated and revised.
- The ESS initialization process has been further streamlined. If you update the autoloads (which installation from `package-install` does), you should not need to `(require 'ess-site)` at all, as autoloads should automatically load ESS when it is needed (e.g. the first time an R buffer is opened). In order to defer loading your ESS config, you may want to do something like `(with-require-after-load "ess" <ess-config-here>)` in your Emacs init file. Users of the popular `use-package` Emacs

package can now do (use-package ess :defer t) to take advantage of this behavior. See Section "Activating and Loading ESS" in ess for more information on this feature.

- ESS now respects Emacs conventions for keybindings. This means that The C-c [letter] bindings have been removed. This affects C-c h, which was bound to ess-eval-line-and-step-invisibly in sas-mode-local-map; C-c f, which was bound to ess-insert-function-outline in ess-add-MM-keys; and C-c h, which was bound to ess-handy-commands in Rd-mode-map, ess-noweb-minor-mode-map, and ess-help-mode-map
- Functions ess-eval-line-and-step and ess-eval-region-or-line-and-step now behave consistently with other evaluation function inside a package.
- ESS[R]: ess-r-package-use-dir now works with any mode. This sets the working directory to the root of the current package including for example C or C++ files within /src).
- ESS[R]: Long + + prompts in the inferior no longer offset output.
- ESS[R]: New option strip for inferior-ess-replace-long+. This strips the entire + + sequence.
- ESS modes now inherit from prog-mode. In the next release, ESS modes will use define-derived-mode so that each mode will have (for example) its own hooks and keymaps.
- ESS[R]: Supports flymake in R buffers for Emacs 26 and newer. Users need to install the lintr package to use it. Customizable options include ess-use-flymake, ess-r-flymake-linters, and ess-r-flymake-lintr-cache.
- ESS[R]: Gained support for xref in Emacs 25+. See Section "Xref" in *The Gnu Emacs Reference Manual*
- ESS[R]: The startup screen is cleaner. It also displays the startup directory with an explicit setwd().
- ESS[R]: Changing the working directory is now always reflected in the process buffer.
- ESS[R]: Makevars files open with makefile-mode.
- New variable ess-write-to-dribble. This allows users to disable the dribble (*ESS*) buffer if they wish.
- All of the *-program-name variables have been renamed to *-program. Users who previously customized e.g. inferior-ess-R-program-name will need to update their customization to inferior-ess-R-program. These variables are treated as risky variables.
- ess-smart-S-assign was renamed to ess-insert-assign. It provides similar functionality but for any keybinding, not just '_'. For instance if you bind it to ';', repeated invokations cycle through between assignment and inserting ';'.
- C-c C-= is now bound to ess-cycle-assign by default. See the documentation for details. New user customization option ess-assign-list controls which assignment operators are cycled.
- ESS[R] In remote sessions, the ESSR package is now fetched from GitHub.
- Commands that send the region to the inferior process now deal with rectangular regions. See the documentation of ess-eval-region for details. This only works on Emacs 25.1 and newer.

- ESS[R]: Improvements to interacting with iESS in non-R files. Interaction with inferior process in non-R files within packages (for instance C or C++ files) has been improved. This is a work in progress.
- ESS[R]: Changing the working directory is now always reflected in the process buffer.
- ESS[JAGS]: *.jog and *.jmd files no longer automatically open in JAGS mode.

Many improvements to fontification:

- Improved customization for faces. ESS now provides custom faces for (nearly) all faces used and places face customization options into their own group. Users can customize these options using *`M-x customize-group RET ess-faces`*.
- Many new keywords were added to `ess-R-keywords` and `ess-R-modifiers`. See the documentation for details.
- ESS[R]: `in` is now only fontified when inside a `for` construct. This avoids spurious fontification, especially in the output buffer where 'in' is a commond English word.
- ESS: Font-lock keywords are now generated lazily.That means you can now add or remove keywords from variables like `ess-R-keywords` in your Emacs configuration file after loading ESS (i.e. in the `:config` section for `use-package` users).
- ESS[R]: Fontification of roxygen `@param` keywords now supports comma-separated parameters.
- ESS[R]: Certain keywords are only fontified if followed by a parenthesis. Function-like keywords such as `if ()` or `stop()` are no longer fontified as keyword if not followed by an opening parenthesis. The same holds for search path modifiers like `library()` or `require()`.
- ESS[R]: Fixed fontification toggling. Especially certain syntactic elements such as `%op%` operators and backquoted function definitions.
- ESS[R]: `ess-font-lock-toggle-keyword` can be called interactively. This command asks with completion for a font-lock group to toggle. This functionality is equivalent to the font-lock menu.

Notable bug fixes:

- `prettify-symbols-mode` no longer breaks indentation. This is accomplished by having the pretty symbols occupy the same number of characters as their non-pretty cousins. You may customize the new variable `ess-r-prettify-symbols` to control this behavior.
- ESS: Inferior process buffers are now always displayed on startup. Additionally, they don't hang Emacs on failures.

Obsolete libraries, functions, and variables:

- The `ess-r-args.el` library has been obsoleted and will be removed in the next release. Use `eldoc-mode` instead, which is on by default.
- Functions and options dealing with the smart assign key are obsolete. The following functions have been made obsolete and will be removed in the next release of ESS: `ess-smart-S-assign`, `ess-toggle-S-assign`, `ess-toggle-S-assign-key`, `ess-disable-smart-S-assign`.

The variable `ess-smart-S-assign-key` is now deprecated and will be removed in the next release. If you would like to continue using '_' for insterting assign in future releases, please bind `ess-insert-assign` in `ess-mode-map` the normal way.

- ESS[S]: Variable `ess-s-versions-list` is obsolete and ignored. Use `ess-s-versions` instead. You may pass arguments by starting the inferior process with the universal argument.

Changes and New Features in 17.11:

- The ESS initialisation process has been streamlined. You can now load the R and Stata modes independently from the rest of ESS. Just put `(require 'ess-r-mode)` or `(require 'ess-stata-mode)` in your init file. This is for experienced Emacs users as this requires setting up autoloads for `.R` files manually. We will keep maintaining `ess-site` for easy loading of all ESS features.
- Reloading and quitting the process is now more robust. If no process is attached, ESS now switches automatically to one (prompting you for selection if there are several running). Reloading and quitting will now work during a debug session or when R is prompting for input (for instance after a crash). Finally, the window configuration is saved and restored after reloading to prevent the buffer of the new process from capturing the cursor.
- ESS[R]: New command `ess-r-package-use-dir`. It sets the working directory of the current process to the current package directory.
- ESS[R] Lookup for references in inferior buffers has been improved. New variable `ess-r-package-source-roots` contains package sub-directories which are searched recursively during the file lookup point. Directories in `ess-tracebug-search-path` are now also searched recursively.
- ESS[R] Namespaced evaluation is now automatically enabled only in the `R/` directory. This way ESS will not attempt to update function definitions from a package if you are working from e.g. a test file.

Changes and New Features in 16.10:

- ESS[R]: Syntax highlighting is now more consistent. Backquoted names are not fontified as strings (since they really are identifiers). Furthermore they are now correctly recognised when they are function definitions or function calls.
- ESS[R]: Backquoted names and `%op%` operators are recognised as sexp. This is useful for code navigation, e.g. with *`C-M-f`* and *`C-M-b`*.
- ESS[R]: Integration of outline mode with roxygen examples fields. You can use outline mode's code folding commands to fold the examples field. This is especially nice to use with well documented packages with long examples set. Set `ess-roxy-fold-examples` to non-nil to automatically fold the examples field when you open a buffer.
- ESS[R]: New experimental feature: syntax highlighting in roxygen examples fields. This is turned off by default. Set `ess-roxy-fontify-examples` to non-nil to try it out.
- ESS[R]: New package development command `ess-r-devtools-ask` bound to *`C-c C-w C-a`*. It asks with completion for any devtools command that takes `pkg` as argument.
- ESS[R]: New command *`C-c C-e C-r`* to reload the inferior process. Currently only implemented for R. The R method runs `inferior-ess-r-reload-hook` on reloading.

- ESS[R]: `ess-r-package-mode` is now activated in non-file buffers as well.

Bug fixes in 16.10:

- ESS[R]: Fix broken (un)flagging for debugging inside packages
- ESS[R]: Fixes (and improvements) in Package development
- ESS[R]: Completion no longer produces `...=` inside `list( )`.
- ESS[R]: Better debugging and tracing in packages.
- ESS[R]: Better detection of symbols at point.
- ESS[R]: No more spurious warnings on deletion of temporary files.
- ESS[julia]: help and completion work (better)
- ESS[julia]: available via `ess-remote`

Changes and New Features in 16.04:

- ESS[R]: `developer` functionality has been refactored. The new user interface consists of a single command `ess-r-set-evaluation-env` bound by default to *`C-c C-t C-s`*. Once an evaluation environment has been set with, all subsequent ESS evaluation will source the code into that environment. By default, for file within R packages the evaluation environment is set to the package environment. Set `ess-r-package-auto-set-evaluation-env` to `nil` to disable this.
- ESS[R]: New `ess-r-package-mode` This development mode provides features to make package development easier. Currently, most of the commands are based on the `devtools` packages and are accessible with *`C-c C-w`* prefix. See the documentation of `ess-r-package-mode` function for all available commands. With *`C-u`* prefix each command asks for extra arguments to the underlying devtools function. This mode is automatically enabled in all files within R packages and is indicated with `[pkg:NAME]` in the mode-line.
- ESS[R]: Help lookup has been improved. It is now possible to get help for namespaced objects such as pkg::foobar. Furthermore, ESS recognizes more reliably when you change `options('html_type')`.
- ESS[R]: New specialized breakpoints for debugging magrittr pipes
- ESS: ESS now implements a simple message passing interface to communicate between ESS and inferior process.

Bug fixes in 16.04:

- ESS[R]: Roxygen blocks with backtics are now correctly filled
- ESS[R]: Don't skip breakpoints in magrittr's `debug_pipe`
- ESS[R]: Error highlighting now understands 'testthat' type errors
- ESS[Julia]: Added getwd and setwd generic commands

1.10 Reporting Bugs

Please send bug reports, suggestions etc. to `ESS-bugs@r-project.org`, or post them on our github issue tracker (`https://github.com/emacs-ess/ESS/issues`)

The easiest way to do this is within Emacs by typing

`M-x ess-submit-bug-report`

This also gives the maintainers valuable information about your installation which may help us to identify or even fix the bug.

If Emacs reports an error, backtraces can help us debug the problem. Type "M-x set-variable RET debug-on-error RET t RET". Then run the command that causes the error and you should see a *Backtrace* buffer containing debug information; send us that buffer.

Note that comments, suggestions, words of praise and large cash donations are also more than welcome.

1.11 Mailing Lists

There is a mailing list for discussions and announcements relating to ESS. Join the list by sending an e-mail with "subscribe ess-help" (or "help") in the body to `ess-help-request@r-project.org`; contributions to the list may be mailed to `ess-help@r-project.org`. Rest assured, this is a fairly low-volume mailing list.

The purposes of the mailing list include

- helping users of ESS to get along with it.
- discussing aspects of using ESS on Emacs and XEmacs.
- suggestions for improvements.
- announcements of new releases of ESS.
- posting small patches to ESS.

1.12 Authors

- A.J. Rossini (`mailto:blindglobe@gmail.com`)
- Richard M. Heiberger (`mailto:rmh@temple.edu`)
- Kurt Hornik (`mailto:Kurt.Hornik@R-project.org`)
- Martin Maechler (`mailto:maechler@stat.math.ethz.ch`)
- Rodney A. Sparapani (`mailto:rsparapa@mcw.edu`)
- Stephen Eglen (`mailto:stephen@gnu.org`)
- Sebastian P. Luque (`mailto:spluque@gmail.com`)
- Henning Redestig (`mailto:henning.red@googlemail.com`)
- Vitalie Spinu (`mailto:spinuvit@gmail.com`)
- Lionel Henry (`mailto:lionel.hry@gmail.com`)
- J. Alexander Branham (`mailto:alex.branham@gmail.com`)