

# Introduction to Org Mode for ESS users

part of the ESS intro series

Greg Minshall

April 6, 2021

# Org Mode

"If Emacs is the distribution, Org Mode is the entire desktop environment one runs on top of it."

Emacs Org Mode is an outline-structured file format and backing software that allows you to (among many other things)

- ▶ markup text via **asteriskizing**, equalizing, ~~plusizing~~, *slashizing*, tildeizing, underscorizing, etc.
- ▶ export a file, or a subtree of it, as, e.g., an .html or .pdf file
- ▶ take notes
- ▶ create agendas (items with date elements)
- ▶ organize your LIFE!
- ▶ do calculations – good support for math, L<sup>A</sup>T<sub>E</sub>X, etc.
- ▶ table support for storing information (including support for formulae)

# Babel – Org Mode support for programming

One can write source code in "source blocks". Source blocks have the following attributes:

- ▶ a name (optional)
- ▶ source code, one language per source block (though, if desired, multiple languages per file)
- ▶ "header arguments", parameters which define how a source block interacts with its environment

## "To a source block", you can

- ▶ **evaluate** it (`[C-c C-c]`<sup>1</sup>) to produce results, which, in turn, may be used as input to another source block. **NB:** there are security issues here: you will need to customize the Emacs variable `org-babel-load-languages`, and you will be prompted each time before a code block is evaluated
- ▶ **export** (`[C-c C-e]`) the code, its results, or both, with other parts of the .org file, into a .html, .pdf, or other format file
- ▶ **tangle** it (`[C-c C-v t]`), that is, write the source code itself into a separate file, that might be used as input to a compilation or some other packaging step, or be a stand alone script (Rscript, say).
- ▶ **edit** it, either in place in the Org buffer, or "stand alone" in an *OrgSrc* buffer (`[C-c ']`<sup>1</sup>), with the possibility of real-time syntax checking/linting, etc.

---

<sup>1</sup>with point in the source block

## R source blocks

In particular, one can have a source block with R code

```
#+begin_src R :results value
  "here, the last value executed is the result"
#+end_src
```

```
#+begin_src R :results output
  cat("here, the output is the result\n")
#+end_src
```

To evaluate an entire source block in an Org buffer, type  
[C-c C-c] or [C-c C-v e].

To edit this source block in an Org Src buffer, type [C-c ']. To  
close the OrgSrc buffer and return to the Org buffer, type [C-c ']  
again.

# Header arguments

**:noweb** allows code from another place in the org file to be inserted at a point in the current org file when evaluating or exporting.

**:var** allows results of other computations in an org file to be used as input to this source block

**:results** defines how results are collected after evaluation of the source block, and how (or whether) they are inserted into the org buffer

**:session** outlined on next slide

**:tangle** name of file to which to *tangle* the contents of this source block

## :session header argument

- ▶ Normally, Org Mode evaluates a source block by forking a new R subprocess, passing it the code (and any variables to be used as input), letting the code run, capturing the results, and then disposing of the R subprocess. This ensures that code runs in a fresh environment.
- ▶ Often, especially when doing statistical analysis, we would like to build up state over time, by executing various code blocks, and/or by poking in the environment.
- ▶ Also, we may want to debug our code in a convenient way.

For these reasons, the `:session` header argument is very useful. All source code blocks specifying the same session name in the `:session` header argument run in the same inferior ESS (`iESS`) process.

# A style of working

In Emacs, I find that I spend time bouncing between three buffers:

- ▶ the .org file, often doing minor in-line edits in a source block
- ▶ an OrgSrc edit buffer, for more major edits (and get font lock, etc.); this buffer comes and goes, as needed
- ▶ the R :session buffer to run code, examine results, and debug as needed



## Other resources

- ▶ the official [Quick Start](#) guide, a very good introduction to Org Mode
- ▶ the [Org Mode](#) web page
- ▶ the Org Mode [worg](#) site
- ▶ the other presentations in this [ESS intro series](#)
- ▶ the beamer slides for this tutorial are [here](#) (pdf)
- ▶ a [.org file](#) for experimenting (an HTML version [here](#) and a PDF one [here](#))

# Tutorials, Videos

There are some nice tutorials:

- ▶ a somewhat older one is from Erik Iverson, [org-mode-R-tutorial.org](http://org-mode-R-tutorial.org)
- ▶ a more recent one from Vikas Rawal, [orgpapers.org](http://orgpapers.org)

And, some videos, including:

- ▶ Rainer König's [screencasts](#) about Org Mode (also available as a [course on Udemy](#))
- ▶ DT ("Distro Tube"?) has yet another [introductory video](#).

# Farewell

Thank you for "attending" this tutorial. I hope it has given you a sense of Org Mode. This has been more of a teaser than an exhaustive introduction, but the resources we listed above should be enough to ease you into using R with Org Mode.