Using Emacs, Org-mode and R for Research Writing in Social Sciences: A Toolkit for Writing Reproducible Research Papers and Monographs

Vikas Rawal

Jawaharlal Nehru University, New Delhi

{E-mail: vikasrawal@gmail.com}

May 13, 2014

Contents

1 Introduction

This guide introduces an open-source toolkit for writing research papers and monographs. The main features of this toolkit centered around Emacs and Org-mode are:

- embedded R code in the document that allows for statistical results to be revised and reproduced,
- bibliographic citations from a personal bibliographic database,

- formatting using well defined styles with minimal markup,
- support for production of final output as pdf, odt, docx, html and many other formats.

1.1 Will this guide be useful for you?

This guide will be useful for you if you are writing a research paper, a dissertation or an academic book. It would be useful if your writing involves one or more of the following:

- Citing existing literature in your area
- Presenting results of statistical analyses (in tabular form and/or graphically)
- Using mathematical equations

Following this guide would need some investment of time but benefits far outweigh the investment you make.

1.2 What is our goal?

What are the most interesting features of the writing platform that you will set up using this guide?

- With easy style specifications that you provide, the document will be almost-entirely automatically formatted by the software.
  - Complicated LaTeX-style markup is a pain and Openoffice/MS-Word documents require too much manual formatting. Basic Org-mode mark-up is extremely simple, and can be mastered in very little time.
  - Org-mode can produce well formatted output in LaTeX, pdf, odt, docx, html and many other formats.
- Instead of including statistical results (tables, graphs, etc), we would embed appropriate R programs in the document, so that when the formatted output is produced, all programs are run to generate the results. Advantages of doing this are:
  - Any changes in the data being used can be accommodated just by publishing the document again.
  - Any modifications in statistical analysis are easily made by modifying the programs that are embedded in the file itself.
  - Anyone who has the org file, can reproduce your results. You can also extract all R programs from the org file and distribute those for reproduction of your results.
- The document will be integrated with a citation manager, so that bibliographic information will be pulled automatically from a central database to create a fully formatted bibliography.

- You will maintain a bibliographic database in BibTeX format, that you can build over time, adding bibliographic information for works that you cite.
- Many websites (including Google Scholar) provide bibliographic information directly in BibTeX format, and we will have integrated tools that will allow us to pull this information directly into our local database.

1.3 Acknowledgements

In my adaptation of Org, I have benefitted immensely from the great community of Orgers. The Org-mode manual, Worg, and archives of the Org-mode mailing list have been the most important resources. In addition, I have greatly benefited from solutions provided by various people to my specific queries on the Org-mode mailing list. What I present in this document is essentially a synthesis of solutions provided by various people, not only in the material posted on Orgmode website, but also in response to specific queries made by me. The community has been extremely generous in providing these.

I would particularly like to thank

- Carsten Dominik, the author of Org-mode
- Bastien Guerry, who has been a great maintainer of Org-mode, after Carsten passed on the baton to him.
- Nicolas Goaziou, who wrote the brilliant new exporter framework. The amount of code Nicolas has contributed to Org over the last two years or so is incredible . Nicolas very kindly responded to several of my queries.
- Eric Schulte, the main author of Babel, which gave Org mode the ability to execute code. I used to use Org-mode as a task manager and for taking notes. I discovered org-babel in the summer of 2010, when I was doing fieldwork in villages in eastern India. This discovery completely changed my work flow, and Org-mode became central to all my academic work.
- In addition to the above, Suvayu Ali, for responses to several of my queries on the mailing list.

2 Installing necessary software

This set up will work with any operating system. I have tested it on GNU/Linux and Mac OS-X, but it should work on Windows as well. For this setup, you need to install Emacs (Version 24 along with a few additional Emacs packages), Texlive, R (along with whatever additional R packages you want to use) and Pandoc.

2.1 Emacs

- GNU/Linux

  Emacs can be installed using package managers of all GNU/Linux distributions. Latest versions of most common distributions provide version 24. I strongly recommend using the latest version of Emacs.

- Mac OS-X

  For Mac OS-X, either use Aquamacs 3.0 or install Emacs using homebrew. The built-in Emacs on OS-X is an older version, and it would be a good idea to install the latest version instead.

- Microsoft Windows

  Download the latest version of Emacs from http://ftp.gnu.org/gnu/emacs/windows/, and install.

2.2 Texlive

- GNU/Linux

  Texlive can also be installed from package managers in most GNU/Linux distribution.

- Mac OS-X

  For OS-X, install MacTeX from http://www.tug.org/mactex/

- Microsoft Windows

  For Windows, download Texlive and follow instructions from https://www.tug.org/texlive/doc.html

2.3 R (http://www.r-project.org)

In this guide, I assume that you are familiar with R. I will not cover R programming in this guide.

For GNU/Linux, R can be installed from native package managers (look for r-base in debian and debian-based distributions). For Mac OS-X and Windows, download and see installation instructions at http://www.r-project.org

2.4 Pandoc (http://johnmacfarlane.net/pandoc/)

Pandoc is an extremely powerful converter, which can translate one markup to another. It supports conversion between many file formats, and supports "syntax for footnotes, tables, flexible ordered lists, definition lists, fenced code blocks, superscript, subscript, strikeout, title

blocks, automatic tables of contents, embedded LaTeX math, citations, and markdown inside HTML block elements." That is pretty much everything I use.

We shall use pandoc to convert our file from latex to odt/docx/html formats.

2.5 Customising emacs

I recommend using Eric Schulte's Emacs Starter Kit to take care of most of the customisation.[1]

To install the kit, go to http://eschulte.github.io/emacs24-starter-kit/#installation and follow the instructions.

Starting version 24, Emacs includes a package-manager. You can install/update add-on packages using the package manager. To use the package manager, press alt-x in emacs, and then type package-list-packages and press return. This would bring up a list of packages. Find ess, and with the cursor on it, mark it by pressing i. Similarly, find bibretrieve and mark it. Then press x to install them.

Org-mode should be pre-installed with Emacs. However, since Org-mode is under heavy development, and it is really a good idea to keep up with the latest version, it is better to clone it from the git repository of Org-mode, and update it regularly. You can keep org-mode under ~/.emacs.d/src/org and compile it.

I also recommend using, in addition, vikas-general.org, available from (https://raw.githubusercontent.com/vikasrawal/orgpaper/master/vikas-general.org). To use it, create a directory with your username under ~/.emacs.d/, git clone this repository, and move vikas-general.org to this directory the directory with your name that you have just created.

For any other personal customisation that you may need to do, you can create .org or .el files in this directory.

3 Emacs basics

GNU Emacs is an extensible platform. Although its primary function is as an editor, it can be extended to do almost anything that you would want your computer to do. Now, that really is not an overstatement. It is a worthwhile aim to slowly shift an increasing number of tasks you do on your computer to emacs-based solutions. For each major task you do on your computer, ask if it can be done using emacs. For almost everything, the answer is yes, and in most cases, emacs does it better than other software you are used to. Many emacs users have learnt emacs by shifting, one-by-one, to emacs for all major tasks that they do on the computer.

I am not going to give a detailed guide to use of emacs. A few tasks for which I use Emacs include

- File management (copying files, moving files, creating directories)

[1] Eric's Emacs Starter Kit is a beautiful illustration of power of Org-mode. It uses Org-mode source blocks to systematically document all Emacs customisation.

- Reading and writing e-mails
- Reading RSS feeds
- Calender, scheduler, planner
- Calculator
- Statistical work (by hooking Emacs to R)
- And, of course, as an editor (including for writing research papers)

In this guide, I will just provide a minimal set of basic commands in emacs to get you started. This is a minimal but a sufficient set to be able work. I expect that you would learn more commands as you start using emacs.

3.1 Notations

In emacs, a buffer is equivalent to a tab in a web browser. It is normal to have several buffers open at the same time. Each file opens in emacs as a buffer. Buffers could also have processes like R running in them. Emacs displays any messages for you in a separate buffer.

Most commands in emacs are given using the Control (ctrl) or the Meta (usually, alt) keys. Control key is usually referred to as `C-` and the Meta key as `M-`. So a command `C-c` means pressing Control and c together. Command `M-x` means pressing Meta and x together. Everything is case-sensitive. So `M-X` would mean, pressing Meta, Shift and x together. `C-c M-x l` would mean pressing C-c, release, then M-x, release, and then l.

3.2 Basic commands

Table 1 gives the commands that are the most important. This is a minimal set, commands that you should aim to learn as soon as possible. There are many more, which you will learn as you start using emacs.

All commands have a verbose version that can be used by pressing `M-x` and writing the command. For example, `M-x find-file` to open a file. All major commands are also mapped to a shortcut. For example, instead of typing `M-x find-file` to open a file, you can say `C-x C-f`. I remember shortcuts for commands that I use most frequently. For others, I use the verbose versions. Over time, one learns more shortcuts and starts using them instead of the verbose versions.

Table 1: Essential emacs commands

Description	Verbose command `M-x` followed by	Shortcut
Opening files, saving and closing		
Open a file	`find-file`	`C-x C-f`
Save the buffer/file	`save-buffer`	`C-x C-s`
Save as: prompts for a new filename and saves the buffer into it	`write-named-file`	`C-x C-w`
Save all buffers and quit emacs	`save-buffers-kill-emacs`	`C-x C-c`
Copy, Cut and Delete Commands		
Delete the rest of the current line	`kill-line`	`C-k`
To select text, press this at the beginning of the region and then take the cursor to the end	`set-mark-command`	`C-spacebar`
Cut the selected region	`kill-region`	`C-w`
Copy the selected region	`copy-region-as-kill`	`M-w`
Paste or insert at current cursor location	`yank`	`C-y`
Search Commands		
prompts for text string and then searches from the current cursor position forwards in the buffer	`isearch-forward`	`C-s`
Find-and-replace: replaces one string with another, one by one, asking for each occurrence of search string	`query-replace`	`M-%`
Find-and-replace: replaces all occurrences of one string with another	`replace-string`	
Other commands		
Divide a long sentence into multiple lines, each smaller than the maximum width specified	`fill-paragraph`	`M-q`
Window and Buffer Commands		
Switch to another buffer	`switch-to-buffer`	`C-x b`
List all buffers	`list-buffers`	`C-x C-b`
Split current window into two windows; each window can show same or different buffers	`double-window`	`C-x 2`
Remove the split	`zero-window`	`C-x 0`
When you have two or more windows, move the cursor to the next window	`other-window`	`C-x o`
Canceling and undoing		
Abort the command in progress	`keyboard-quit`	`C-g`
Undo	`undo`	`C-_`

4 Org-mode basics

4.1 Preamble

An Org file has a few special lines at the top that set up the environment. Following lines are an example of the minimal set of lines that we shall use.

```
#+TITLE: Reproducible Research Papers using Org-mode and R: A Guide
#+AUTHOR: Vikas Rawal
#+DATE: May 4, 2014
#+OPTIONS: toc:2 H:3 num:2
```

As you can see, each line starts with a keyword, and the values for this keyword are specified after the colon.

Table 2 gives details of a few major special lines that we shall use.

Table 2: Main special lines to be used at the top of an Org buffer

Keyword	Purpose
`#+TITLE`	To declare title of the paper
`#+AUTHOR`	To declare author/s of the paper
`#+DATE`	Sets the date. If blank, no date is used. If this keyword is omitted, current date is used.
`#+OPTIONS`	There are many options you can give. These are what I find the most important Multiple options can be separated by a space and specified on the same line. toc:nil (Do not include a Table of contents), toc:n (Include n levels of sections and sub-sections in Table of contents) H:2 (Treat top two levels of headlines as section levels, and anything below that as item list. Modify the number as appropriate) num:2 (Number top two levels of headlines. Modify the number as appropriate.)

In addition to these, we can use LaTeX specific options for formatting the pdf output, odt specific options for formatting the odt/docx output, and R specific options for setting up the R environment. These would also be specified using special lines at the top of the file. I shall provide details of some of these in the Sections where these topics are discussed.

4.2 Sections and headlines

After the special lines at the top comes the main body of the Org file.

The content in any Org file is organised in a hierarchy of headlines. Think of these headlines as sections of your paper.

A headline in Org starts with one or more stars (*) followed by a space. A single star denotes the main sections, double star denote the subsections, three stars denote sub-subsections, and so on. We shall use this to create the section structure of our document. You can create as many levels of sections as you need.

See the following example. Note that headlines are not numbered. We leave section numbering for org-mode to handle automatically.

```
#+TITLE: Reproducible Research Papers using Org-mode and R: A Guide
#+AUTHOR: Vikas Rawal
#+DATE: May 4, 2014
* Introduction
This is the first section. Add your content here.
* Literature review
** Is this an important issue
This is a sub-section under top-level section "Literature review" Now
indulgence dissimilar for his thoroughly has terminated. Agreement
offending commanded my an. Change wholly say why eldest period. Are
projection put celebrated particular unreserved joy unsatiable its. In
then dare good am rose bred or. On am in nearer square wanted.
** What are the major disputes in the literature
*** adulterated text
Instrument cultivated alteration any favourable expression law far
nor. Both new like tore but year. An from mean on with when sing pain.
Oh to as principles devonshire companions unsatiable an delightful.
The ourselves suffering the sincerity. Inhabit her manners adapted age
certain. Debating offended at branched striking be subjects.
*** Unadulterated prose
Announcing of invitation principles in. Cold in late or deal.
Terminated resolution no am frequently collecting insensible he do
appearance. Projection invitation affronting admiration if no on or.
It as instrument boisterous frequently apartments an in. Mr excellence
inquietude conviction is in unreserved particular. You fully seems
stand nay own point walls. Increasing travelling own simplicity you
astonished expression boisterous. Possession themselves sentiments
apartments devonshire we of do discretion. Enjoyment discourse ye
continued pronounce we necessary abilities.
* Methodology
This is the next top-level section. There are no sub-sections under this.
* Results
This is the third top-level section. Theere are sub-sections under this.
```

```
** Result 1
This is a sub-section under section Results.
** Result 2
This is another sub-section under section Results
* Conclusions
This is the next and final top-level section. There are  no sub-sections under it.
```

Org handles these headlines beautifully. With your cursor on the headline, pressing tab folds-in the contents of a headline. If you press tab on a folded headline, it opens to display the contents. If there are multiple levels of headlines, these open in stages as you repeat pressing the tab key.

When you are on a headline, pressing M-return creates a new headline at the same level (that is, with the same number of stars). Once you are on the new headline, a tab movies it to a lower level (that is, a star is added), and shift-tab moves it to a higher level (that is, a star is removed).

When I start writing a paper, I start with a tentative headline/section structure, and then start filling in the content under each headline, and modify the section structure, if needed, as the paper develops.

(Further reading, Headlines in Org manual)

4.3 Itemised lists

Following syntax produces unordered (bulleted) lists:

```
+ bullet
+ bullet
  - bullet2 1
  - bullet2 2
+ bullet
+ bullet
```

This is how this list shows up in the final document

- bullet
- bullet
  - bullet2 1
  - bullet2 2
- bullet
- bullet

Following syntax produces ordered/numbered lists:

```
1. Item 1
2. Item 2
  1) Item 2.1
  2) Item 2.2
     1) Item 2.2.1
3) Item 3
```

This is how the ordered list shows up in the final document.

1. Item 1
2. Item 2
   (a) Item 2.1
   (b) Item 2.2
      i. Item 2.2.1
3. Item 3

Note that:

- In unordered lists, `+` and `-` signs are interchangeable.
- Similarly, in ordered lists 1. and 1) are interchangeable.
- Levels of bullets and numbering are determined by indentation.
- Ordered and unordered lists can be mixed using numbers and bullets for different levels.
- If the cursor is on a line that is part of an itemised list, M-return inserts a new line with a bullet/number below the present line with the same level of indentation.

4.4 Inserting footnotes

- To insert a footnote at any point, use `C-c C-x f`
- To reorder and renumber footnotes after inserting a footnote in a text that already has some footnotes after the point where a new footnote is being inserted, use `C-u C-c C-x f S`

4.5 Tables

Sample code

We shall directly create only those tables in Org that present content not being produced through statistical analysis. For tables that are created through statistical analysis, we shall embed R programs rather than the tables themselves. This is discussed in Section 5 of this guide.

The following sample code produces a fully formated table, with a numbered title above the table and a name for cross-referencing the table from the text anywhere in the document.

```
#+NAME: table-yield
#+CAPTION: Average yields and average income, by State, India
  | State          | Average yield | Average income |
  |----------------+---------------+----------------|
  | Madhya Pradesh |           669 |          13000 |
  | Haryana        |           300 |          25000 |
  | Punjab         |           260 |          35000 |
```

See Table 3, for an illustration of how this table shows up in the final document.

Table 3: Average yields and average income, by State, India

State	Average yield	Average income
Madhya Pradesh	669	13000
Haryana	300	25000
Punjab	260	35000

Table editor

Org-mode has an in-built table editor, which is very simple to use.

- Tables in Org have columns separated using |.
- Once you create the first row by separating columns using |, pressing tabs takes you from the first column to the next. Org automatically aligns the columns.
- At the end of the row, pressing tab again, creates a new blank row. You can also create a new blank row by pressing return anywhere in the last row.
- For creating a horizontal line anywhere, type |- at the starting of the line, and press tab.
- Contents of each cell are aligned automatically by Org.
- To delete a row, use `C-k` (`M-x kill-line`).

Org provides various commands for manipulating design of tables. Table 4 provides the most important ones. Note that Table 4 is created using Org mode. It also gives you an idea of how the table would look eventually.

Table 4: Commands to manipulate tables in Org

Command	Description
`M-<left>`	Move the column left
`M-<right>`	Move the column right
`M-S-<left>`	Delete the current column
`M-S-<right>`	Insert a new column to the left of the cursor position
`M-<up>`	Move row up
`M-<down>`	Move row down
`M-S-<up>`	Delete the current row or horizontal line
`M-S-<down>`	Insert a new row above the current row

For more commands for manipulating tables, see this section of the Org manual. In particular, you may want to look at spreadsheet-like functions of the table editor.

One limitation of Org is lack of support for merging of cells in a Table.

Captions and cross-references

Please note the first two lines in the code for creating Table `table-yield-3`.

A line starting with `#+CAPTION:` placed just above a table adds a title to it. All Tables and Figures titles are automatically numbered.

For referring to these Tables from the text, we name each table in a line starting with `#+NAME:`. The table can then be referred to from anywhere in the text by usit commit –amend -m "New commit message" " Table `[[table-yield]]`. As an illustration, see the following sentence.

```
Tables [[table-yield]] and [[health-table]], and Figure
[[literacy-figure]], show the level of underdevelopment.
```

By default, all objects with captions are numbered, and names are used to anchor cross-references. When the formatted output is produced, all the references would be automatically converted to appropriate numbers. If new objects are inserted in the paper, numbering will be adjusted automatically when you create the formatted output.

4.6 Images

You can insert images in documents as follows

```
[[a.jpg]]
```

You should do this for images that you already have, and you just want to insert them in the document. For graphs produced by R, we shall embed the code instead, so that the graph is generated and inserted automatically.

4.7 Captions and cross-references

We would like to give a title to our tables and images. And we would like to be able to refer to them from the text. These are achieved by adding two lines above every table and image.

- A line starting with `#+CAPTION:` placed just above a table or a figure adds a title to it. All Tables and Figures titles are automatically numbered.
- For referring to these Tables and Figures in the text, we shall name each table and figure in a line starting with `#+NAME:` as below.

To illustrate, for inserting an image, with a caption and a name, this is what we shall do.

```
#+NAME: literacy-rate
#+CAPTION: Percentage of literate men and women, by country (per cent)
[[a.jpg]]
```

Similarly, a table will be inserted as follows.

```
#+NAME: literacy-rate-table
#+CAPTION: Percentage of literate men and women, by country (per cent)
| Country    | Men | Women |
|------------+-----+-------|
| India      |  75 |    43 |
| Bangladesh |  83 |    63 |
| Rwanda     |  77 |    60 |
```

,#+NAME: literacy-rate-table ,#+CAPTION: Percentage of literate men and women, by country (per cent)

Country	Men	Women
India	75	43
Bangladesh	83	63
Rwanda	77	60

To refer to the Table above in the text, write Table `[[literacy-rate-table]]`. As an illustration, see the following sentence.

```
Tables [[literacy-rate-table]] and [[health-table]], and Figure
[[literacy-figure]], show the level of underdevelopment.
```

By default, all objects with captions are numbered, and names are used to anchor cross-references. When the formatted output is produced, all the references would be automatically converted to appropriate numbers. If new objects are inserted in the paper, numbering will be adjusted automatically when you create the formatted output.

4.8 Formatting tables for LaTeX/PDF export

Column types

The default LaTeX `tabular` environment allows only a few column types. In particular, there is limited support in tabular environment for wrapping text in different types of columns. However, there are many other LaTeX environments for making tables, each with different advantages. I find `tabulary` the most useful for my needs.

Table 5 shows different types of columns available in tabulary package.

Table 5: Types of columns in LaTeX/tabulary package

Type	Description
l	Left aligned, no wrapping
L	Left aligned with wrapping
r	Right aligned, no wrapping
R	Right aligned with wrapping
c	Centre aligned, no wrapping
C	Centre aligned with wrapping
J	Justified and wrapped

A line of the following type needs to be inserted above an Org table to make it use `tabulary` environment instead of `tabular`.

```
#+attr_latex: :environment tabulary :width \textwidth :align L|llR
```

:width is used to specify the *maximum* total width of the table that the table can take [it may be specified as `\textwidth`, implying full text width, or in centimeters (like, 10cm) or in innces (like, 5in)]. Note that, in `tabulary`, the width is the maximum width of the whole table. If your columns do not need the entire width that you specify, the table turns out narrower than the width.

`:align` specifies how to render each columns by using one letter (l,L,r,R,c,C or J) for each column. The number of letters should exactly match the number of columns in your table. A `|` anywhere implies a vertical column (which should be used sparingly).

Notes below tables

LaTeX package `threeparttable` is used for including notes below the table. For using `threeparttable` you need to call the package. In addition, it is a good idea to include the following special line

for better formating of notes below the table

```
#+LATEX_HEADER: \renewcommand{\TPTminimum}{\linewidth}
```

The following code produces a table with notes below.

```
#+NAME: table-yield
#+CAPTION: Average yields and average income, by State, India
#+begin_table
#+begin_threeparttable
#+attr_latex: :environment tabulary :width \textwidth :align Lrr
  | State          | Average yield | Average income |
  |----------------+---------------+----------------|
  | Madhya Pradesh |           669 |          13000 |
  | Haryana        |           300 |          25000 |
  | Punjab         |           260 |          35000 |
#+begin_tablenotes
  \item[] \footnotesize Notes:
  \item[1] \footnotesize This table is very nice but this note is
  very long, so long that it goes wider than the table
  \item[2] \footnotesize This is a second note. But this is not
  very wide.
  \item[] \footnotesize Source: http://www.indianstatistics.org}
#+end_tablenotes
#+end_threeparttable
#+end_table
```

The notes use a little bit of direct LaTeX coding.

- `\item[]` ensures that each note is in a separate paragraph.
- `\footenotesize`, which is optional, renders the notes in a slightly smaller font.

5 Org-mode and R

5.1 Configuration

Following code in vikas-general.org enables Org to run different types of code. If you have installed vikas-general.org as specified in 2.5, these are already enabled.

I have included here the languages that I commonly use. See Org manual, if you would like to add any more.

```
(org-babel-do-load-languages
   'org-babel-load-languages
   '((R . t)
     (org . t)
     (ditaa . t)
     (latex . t)
     (dot . t)
     (emacs-lisp . t)
     (gnuplot . t)
     (screen . nil)
     (shell . t)
     (sql . nil)
     (sqlite . t)))
```

5.2 Special lines for R

Org allows you to run multiple R sessions simultaneously, if you are working on two documents side by side, and would like to keep statistical work for the two separately.

This is done by naming the R session which a particular Org file is linked to. All R code in this file would be run in the specified R session. You could have, at the same time, another R session, with a different name, being called by another Org buffer.

We can give a name to the R session (let us say, my-r-session) that our Org buffer should be linked to by adding the following line at the top (in the preamble, that is).

```
#+property: session my-r-session
```

5.3 Embedding R code in an Org document

Org uses ESS (emacs-speaks-statistics) to provide a fully functional, syntax-aware, development environment to write R code. R code is embedded into Org as a source block. The basic syntax is

```
#+NAME: name_of_code_block
#+BEGIN_SRC R <switches> <header-arguments>

  <Your R code goes here.>

#+END_SRC
```

This is how source blocks are created.

- First write the lines starting with `#+NAME`, `#+BEGIN_SRC` and `#+END_SRC`.

- Then with your cursor in between the `BEGIN_SRC` and the `END_SRC` lines, give the command C-c ' (that is, press Ctrl-C, release, and press ').
  - This would open a new buffer using ESS mode. If you type your code in this buffer, you will see that ESS is syntax-aware and nicely highlights R code.
  - ESS also allows you to run (evaluate) the code that you write, to test what your code is doing. Use `C-j` for evaluating a single line of code, `C-b` for evaluating the whole ess buffer, or `C-r` for a marked region within the ess buffer.
- Once you have finished writing a code block and tested it, press C-c ' again to come back to your Org buffer.
- In your Org buffer, with your cursor in a source-block, press `C-c C-c` to evaluate the whole code block and have the results included in your document.
- You can always edit your source code by opening a temporary ESS buffer using C-c'

5.4 Code blocks that read data and load functions for later use in the document without any immediate output

I normally have one or two code blocks that read the data I am going to use, call the libraries that I use, and define a few functions of my own that I plan to use. I want this code block to be evaluated, so that these data, libraries and functions become available in my R environment. But no output from such code blocks is expected to be included into the document.[2]

Code block `readdata-code` is an example of such a code block. Note `:results value silent` switch used in the `#+begin_src` line.

```
#+NAME: readdata-code
#+BEGIN_SRC R :results value silent

read.data("datafile1.csv",sep=",",header=T)->mydata1


#+END_SRC
```

5.5 Code blocks that produce results in the form of a table

Most of code blocks in my papers fall in this category. The code block may use data and functions made available by previous code blocks, read some new data and may load some new

[2]For libraries and functions that you need to call, it is even better to include them in a .Rprofile file in your working directory. These libraries and functions would then be called when R is started, and not each time you evaluate code blocks in your document.

functions. The code block does some statistical processing. The last command of the code block produces an object (for example, a data.frame) that is included in the document as a Table.

For example, the code block `bmi-table-code` below uses mydata1 read in the previous code block, reads a new dataset, and processes them to create a table that shows average BMI by country.

```
#+NAME: bmi-table-code
#+BEGIN_SRC R :results value :colnames yes :hline yes
aggregate(height~Country,data=mydata1,mean)->a1
read.data("datafile2.csv",sep=",",header=T)->mydata2
aggregate(weight~Country,data=mydata2,mean)->a2
merge(a1,a2,by="Country")->a1
a1$weight/a1$height->a1$BMI
subset(a1,select=c("Country","BMI"))
#+END_SRC
```

You can evaluate this code using C-c C-c. When you do that, it produces the output, and places it immediately below the code block. The results display the output of the code under a line that looks like below

```
#+RESULTS: bmi-table-code
```

Note that the results are tied to the code block using the name of the code block. Every time you go to the source code block and press `C-c C-c`, the code will be evaluated again and the results will be updated.

On top of the line starting with `#+RESULTS:`, we shall add two more lines, to give the table a title and a name. Note that both the code block and the result of the code block have separate names.

```
#+NAME: bmi-table-output
#+CAPTION: Average BMI, by country
#+RESULTS: bmi-table-code
```

Like any Org table, you can cross-refer to this table using `[[bmi-table-output]]`.

5.6 Code blocks that produce a graph to be included in the document

These code blocks can have a series of commands. The last command produces a graph that we would like to be included in the document.

The following code shows an example of a code block that produces a graph.

```
#+NAME: mygraph-code
#+BEGIN_SRC R :results output graphics :file bmi2.png :width 825 :height 1050 :fonts serif
```

```
#+END_SRC
```

As before, for creating your graph, you first write the `#+NAME`, `BEGIN_SRC` and the `END_SRC` lines, and then go into a temporary ESS buffer by using C-c '.

- Once in this temporary ESS buffer, you can write the R commands for making your graph.
- As you write, you can evaluate the commands using `C-j`, `C-r` and `C-b` and see what your output looks like.
- The output is displayed on your screen using the default graphic device used by R (X11, quartz or windows graphic device depending upon your operating system).
- Once you have finalised your graph, you press C-c ' and come back to the Org buffer.

Note that creation of the image file is left to appropriate switches in the `#+BEGIN_SRC` line. Org automatically chooses appropriate graphic device to produce the file. When you evaluate this code using `C-c C-c`, the results are displayed below the code block as follows.

```
#+RESULTS: mygraph-code
[[bmi2.png]]
```

Note that, taking the file name from our `#+BEGIN_SRC` line, a file called `bmi2.png` was automatically created and linked, so that the graph would be inserted in the document when you produce the formatted output.[3] Every time you evaluate the code using `C-c C-c`, the underlying image file containing the graph is overwritten by a new file.

As with the tables, we shall add a caption and a name to it as follows

```
#+NAME: my-bmi-graph
#+CAPTION: Average BMI, by Country
#+RESULTS: mygraph-code
[[gini.png]]
```

You can now refer to this graph in the text using `[[my-bmi-graph]]`.

6 Citations and Bibliographies using Org-mode

6.1 Building your database

We shall use a master bibliographic database to contain bibliographic records for the literature that we cite. The database, in Biber or BibTex format, is stored in a text file with .bib extension.

[3] Of various image formats, I find that png files are most versatile. png files support transparency, and are rendered well both on the web and in print. You can also specify jpeg or pdf files. pdf files for images work very well if you are only going to produce a pdf document.

In a BibTex/Biber database, each bibliographic record is given a unique key, which is used to cite it. Each record is classified as one among various categories of publications (journal article, book, etc.), and for the given publication type, the record specifies values for various fields (author, title, volume, publisher, etc). Biber recognises a wider variety of publication types and fields than BibTex, and is a better choice to use. Since it is compatible with BibTex, you can also add a BibTex record as it is as a Biber record.

To start with, it may be a good idea to use applications like JabRef (cross-platform, http://jabref.sourceforge.net/) or BibDesk (OS-X only, http://bibdesk.sourceforge.net/) to build your database. Eventually, you should use bibretrieve and RefTeX (http://www.gnu.org/software/auctex/reftex.html) from within Emacs to add entries to your database. org-ref.el provided by John Kitchin (https://github.com/jkitchin/jmax) has some useful functions.

Bibliographic information in BibTex/Biber format is available from many online sources, including Google Scholar. JabRef/BibDesk allow you to directly import BibTex citations from online databases rather than having to enter everything yourself. Of course, where the bibliographic information in BibTex/Biber format is not available from any existing database, you may have to enter the information yourself.

As a sample, my own bibliographic database is available from https://github.com/indianstatistics/bibliobase/blob/master/bibliobase.bib.

6.2 Using biblatex with Org

Setup

Using biblatex with Org requires some customisation of variables. This is already done for you if you have installed vikas-general.org.

The operative part in vikas-general.org is the following:

```
(setq org-latex-to-pdf-process
  '("pdflatex %f" "biber %b" "pdflatex %f" "pdflatex %f"))
```

Once this is done, every time you export the document to pdf via latex, it runs pdflatex, then runs Biber and then runs pdflatex twice again. This is necessary to get the citations in the pdf file.

In vikas-general.org, the package biblatex is loaded with following options:

```
("citestyle=authoryear-icomp,bibstyle=authoryear,hyperref=true,backref=true,
maxcitenames=3,url=true,backend=biber,natbib=true" "biblatex" t)
```

You may want to modify this if you want to change the citation and bibliography styles. If you want to do it differently in each document, you can remove this line from vikas-general.org, and add the following special line in your document.

```
#+LATEX_HEADER: \usepackage["citestyle=authoryear-icomp,bibstyle=authoryear, \
hyperref=true,backref=true,maxcitenames=3,url=true,backend=biber,natbib=true"] {biblatex}
```

Adding citations and bibliography in Latex/PDF export

The following special line, to be placed among other special lines at the top of the file, specifies the BibTex/Biber database that has the bibliographic records.

```
#+LATEX_HEADER: \addbibresource{filename.bib}
```

There are various commands that you can use for citations. These use different styles for citation. The general syntax of citation commands is:

`\command[<prenote>][<postnote>]{<Key>}`

Where `[<prenote>]` refers to any text you want before citation (for example "for more details, see") and `[<postnote>]` refers to any text you want after citation (for example, "Chapter 2").

The two most useful citation commands are `\parencite` (or `\citep`) and `\citet`. Their usage is illustrated in Table 6.

Table 6: Important citation commands in biblatex

Citation command	Output
`\parencite{jon90}`	(Jones et al., 1990)
`\parencite[chap. 2]{jon90}`	(Jones et al., 1990, chap. 2)
`\parencite[see][]{jon90}`	(see Jones et al., 1990)
`\parencite[see][chap. 2]{jon90}`	(see Jones et al., 1990, chap. 2)
`\parencite*{jon90}`	(Jones, Baker, and Williams, 1990)
`\citet{jon90}`	Jones et al. (1990)
`\citet[chap. 2]{jon90}`	Jones et al. (1990, chap. 2)
`\citet*{jon90}`	Jones, Baker, and Williams (1990)

To insert the bibliography, add the following line where you want to insert the bibliography (usually, at the end of your paper, but before the Footnotes)

```
\printbibliography
```

7 Producing a formatted LaTeX, pdf, odt, docx or html file

From Org, we can get a well-formatted document as a LaTeX, PDF, odt, docx or html file. To produce a formatted output, we shall use the built-in exporters provided with Org, and for some file types, use Pandoc for further conversion.

Built-in exporters can be called in Org using `C-c C-e` or `M-x org-export-dispatch`.

7.1 Creating LaTeX and/or PDF files

Use `C-c C-e` to call the Org export dispatcher.

- Press l to select LaTeX, and then chose one of the following options.
  - Press l again, if you just want to create a LaTeX file
  - Press p, if you want to create a pdf file. This will first create a latex file, then use pdflatex and Biber to create a pdf file.
  - Press o, if you want to create pdf and have it opened in the default pdf viewing application.

7.2 Creating odt or docx files

There is a built-in odt exporter in Org. While it works well for most situations, there are two components of the setup proposed here that it does not support. It does not support biblatex and it does not support LaTeX-specific solution we have for Notes under Tables and Images.[4]

Fortunately, Pandoc provides an excellent solution for converting LaTeX output to odt or docx documents. Pandoc supports all the LaTeX syntax that Org produces from our files, and you can get a very well formatted output.

Use `C-c C-e l l` to create a LaTeX file. Then, from the terminal, use Pandoc as follows to create an odt or a docx file.

```
pandoc --bibliography=biblidatabase.bib --filter pandoc-citeproc \
latexfile.tex -o outputfile.odt
```

```
pandoc --bibliography=biblidatabase.bib --filter pandoc-citeproc \
latexfile.tex -o outputfile.docx
```

If you want, you can use –template to specify an ott or a .dotx template file, so that the fonts and other formatting attributes are to your liking.

7.3 HTML

For html as well, there is a built-in exporter in Org. The built-in exporter is very good, and the way to go if you are planning to maintain a website using Org (as I do for `http://www.indianstatistics.org`).

The built-in exporter can support BibTex citations using ox-BibTex.el, which is including in Org, and will be loaded if you have installed vikas-general.org. You may need to install BibTex2html separately to make it work.

[4] Author of the odt exporter has chosen to develop the exporter outside Org-mode. He has developed a JabRef exporter to integrate citations into odt exports, but that is not a part of Org-mode and needs to be installed separately. In any case, since our toolkit primarily uses LaTeX, using Pandoc to create odt or docx files from LaTeX export works better.

However, ox-BibTex.el uses BibTex2html for converting citations and bibliography to html. BibTex2html provides limited support for citation and bibliography styles.

If you want full support for bibliography and citation styles, as well as for other LaTeX components like Table notes explained in this document, you can use Pandoc for converting LaTeX to html.

8 Additional tips and tricks

This section points some additional solutions that you may like to use. Some of these may come handy when you start using Org for documenting your research.

8.1 Emacs color-theme

It is good to use a color-theme that highlights text in Org well. This makes it easier to work in Org.

I use color theme caled leuven. The code for loading leuven is including in vikas-general.org but is commented out. If you wish to use it, remove ;; from the front of the relevant lines, and you will have Leuven.

8.2 Evaluating code during export

By default, Org evaluates source code at the time of exporting. If your code involves a lot of computation, this can slow down exporting.

In such cases, I do not evaluate all source blocks at the time of exporting. I evaluate them manually, one by one, using `C-c C-c`.

Insert following line right at the top. Note, this has to be the first line of the buffer.

```
# -*- mode: org; org-export-babel-evaluate: nil -*-
```

If your buffer has this line, the source code is not evaluated at the time of export, and whatever already exists in `#+RESULTS` block is exported.

8.3 Garamond font

I like to use garamond font. If you do too, add this special like at the top:

```
#+LaTeX_CLASS_OPTIONS: [garamond]
```

8.4 Margins

In LaTeX, package Geometry allows you to modify page margins. The following line in vikas-general.org sets the margins. You can tweak this to define the margins as you like.

```
("innermargin=1.5in,outermargin=1.25in,vmargin=1.25in" "geometry" t)
```

If you would like to do it for each document separately, remove the above line, and add the following special line at the top in your documents.

```
#+LaTeX_HEADER: \usepackage[innermargin=1.5in,outermargin=1.25in,vmargin=3cm]{geometry}
```

8.5 Line spacing

Use the following line at the top. Modify the number to whatever suits you.

```
#+LATEX_HEADER: \linespread{1.3}
```

8.6 Acknowledgements in footnote

When writing a research paper, it is common to put acknowledgements in a special footnote to names of authors. It is conventional to use * as the symbol for this footnote, and to keep this footnote out of the list of numbered footnotes that the paper may have.

This is achieved as follows.

- As illustrated in the example below, add acknowledgements in the special line that specifies authors of the paper.

```
#+AUTHOR: Vikas Rawal\footnote{Write your acknowledgements here...}
```

- Then, before your first headline, add the following text.

```
#+BEGIN_LaTeX
{% begin group
\renewcommand{\thefootnote}{\fnsymbol{footnote}}% set smybols
\setcounter{footnote}{0}% set footnote counter back to 0
}% end group
#+END_LaTeX
```

8.7 Restricting Location of Tables and Images in LaTeX export

LaTeX has a very sophisticated algorithm for determining the location of Tables and Images in a document. If, however, you want to add a restriction that the Tables and Images should not cross section boundaries, or a particular boundary, this can be done using command `\FloatBarier` provided by placeins package in LaTeX.

You can put any number of `\FloatBarrier` commands, each in a line by itself, in the document. Tables and Images before such a barrier will be placed before the barrier.

You can use the following special line at the top to restrict all Tables and Images within their own sections.

```
#+LATEX_HEADER: \usepackage[section]{placeins}
```

An extension to placeins package, extraplaceins can be used if you want to restrict the Tables and Images within subsections.[5]

8.8 Customising Biblatex style

I like to use authoryear bibliography style. However, I need some customisations.

- Use comma in place of period after Journal name.
- Remove "in: "
- For Journal articles, specify volume and issue number as volume(number)

The file biblatex.cfg included in this repository (https://raw.githubusercontent.com/vikasrawal/orgpaper/master/biblatex.cfg) takes care of these.

The following command tells you where is your texmf directory

```
kpsewhich -var-value=TEXMFHOME
```

I created `$TEXMFHOME/tex/latex/biblatex` and put the biblatex.cfg in that folder.

9 Important Resources

- Org-mode manual
- Worg
- Org-mode mailing list
- Emacs manual
- R website
- Pandoc
- E. Schulte, D. Davison, T. Dye, and C. Dominik. A multi-language computing environment for literate programming and reproducible research. Journal of Statistical Software, 46(3):1–24, 1 2012.http://www.jstatsoft.org/v46/i03
- Tutorial: Writing scientific papers for ACPD using emacs org-mode, http://draketo.de/english/emacs/writing-papers-in-org-mode-acpd
- Writing papers Using org-mode, http://nakkaya.com/2010/09/07/writing-papers-using-org-mode

[5] See http://lexfridman.com/blogs/research/2011/03/06/prevent-figures-from-floating-outside-sections-in-latex/