mp

Chris Dardis

Wednesday 27th May, 2015

Contents

1 Package Attributes

- **Commentary** :

Makes a pdf from the materials in the 'default-directory' (or will search up the directory tree for the appropriate file type).

These may include .R, .tex, .Rnw, .org and .bib files. Indexes, glossaries and table of contents are supported.

Place the folliwng in your init.el file:

(add-to-list 'load-path "~/path/to/directory") (require 'mp)

and, optionally, For HTML export of .org files:

(require 'htmlize)

Define f8 as prefix for code chunk in ESS, as there are 8 characters in the prefix:

(fset 'chunk1 "## ---- ") (add-hook 'ess-mode-hook (lambda () (local-set-key (kbd "<f8>") 'chunk1)))

There is also a simple function for producing .pdf's from an .el package file.

Function-local/temporary variables are named using the camelCase convention or, for single words, as e.g. variable1.

2 Groups

2.1 mp

This group consists of the elements of 'make-pdf'. This is a series of variables and functions to simplity the process of pdf creation using R, LaTeX and the intermediaries (entwiners) knitr, Sweave and Org ('org-mode').

2.2 mp-files

This group is part of 'mp'. These custom variables are whole files, which are stored as strings in elisp.

3 Functions

3.1 mp-mp *(&optional FILENAME)*

Documentation: 'Entwine' elements in a directory to produce and view a .pdf.

If no FILENAME is supplied, it will try to find the most recently modified of the following file types (in the order below) and pass this to the appropriate method.

There are two special cases, which are single character responses:

- If 'p' is entered, it will try to open the appropriate .pdf file directly.
- If 'c' is entered, it will call 'mp-restart''.
- .R –> 'mp-R-nw-or-org'
- .Rnw –> 'mp-nw-tex'
- .tex –> 'mp-latexmk'
- .org –> 'mp-org-tex'

- .el –> ‘mp-el-tex’
- .pdf –> ‘mp-view-pdf’

```
(interactive "F FILENAME: ")

(let
   (df1)
  (when
     (equal FILENAME "c")
   (mp-clean))
  (when FILENAME
   (setq FILENAME
        (file-name-nondirectory FILENAME)))
  (when
     (equal FILENAME "")
   (setq FILENAME nil))
  (when
     (and FILENAME
         (equal "p"
               (substring FILENAME
                        (-
                         (length FILENAME)
                         1)
                        (length FILENAME))))
   (when
      (buffer-file-name)
    (setq FILENAME
         (car
          (file-expand-wildcards
           (concat
            (file-name-sans-extension
             (file-name-nondirectory
              (buffer-file-name)))
            ".pdf")))))
   (unless
      (buffer-file-name)
    (set 'df1
        (directory-files default-directory nil ".*.pdf" t))
    (set 'df1
        (sort df1 'file-newer-than-file-p))
    (setq FILENAME
         (car df1))))
```

```
  (unless FILENAME
    (setq df1
         (directory-files default-directory nil "\\.R$\\|\\.Rnw$\\|\\.
            tex$\\|\\.org$\\|\\.el$\\|\\.pdf$"))
    (set 'df1
        (sort df1 'file-newer-than-file-p))
    (setq FILENAME
         (car df1)))
  (when buffer-file-name
    (save-buffer))
  (let
     (ext1)
    (set 'ext1
        (file-name-extension FILENAME))
    (cond
     ((string= ext1 'R)
      (mp-R-nw-or-org FILENAME))
     ((string= ext1 'Rnw)
      (mp-nw-tex FILENAME))
     ((string= ext1 'tex)
      (mp-latexmk FILENAME))
     ((string= ext1 'org)
      (mp-org-tex FILENAME))
     ((string= ext1 'el)
      (mp-el-tex FILENAME))
     ((string= ext1 'pdf)
      (mp-view-pdf FILENAME)))))
```

3.2 mp-clean *nil*

Documentation: Delete all files in the 'default-directory' with the following extensions: -.Rnw -.tex -.log -.org -.fls -.aux

```
(interactive)

(let
   (df1)
  (set 'df1
      (directory-files default-directory nil "\\.Rnw$\\|\\.tex$\\|\\.
         log$\\|\\.org$\\|\\.fls$\\|\\.aux$"))
  (mapc 'delete-file df1)
  (mp-mp))
```

3.3 mp-get-file *(&optional NAMESTEM EXTENSION)*

Documentation: Find the appropriate file based on the namestem and extension provided (as strings).

The function searches in the current 'default-directory'. If no matching file is found, it will search up the directory tree.

```
(interactive "s NAMESTEM: \ns EXTENSION: ")

(message "mp-get-file...")

(let
   (fileName df1)
  (setq fileName
      (buffer-name))
  (when
     (and
      (not fileName)
      (or
       (not
        (string= NAMESTEM ""))
       (not
        (equal nil NAMESTEM))
       (not
        (string= NAMESTEM
                 (file-name-sans-extension fileName)))))
   (setq fileName
        (member
         (concat NAMESTEM "." EXTENSION)
         (file-expand-wildcards
          (concat "*." EXTENSION)))))
  (unless fileName
   (set 'df1
       (file-expand-wildcards
        (concat "\\." EXTENSION)))
   (setq fileName
        (car
         (sort df1 'file-newer-than-file-p))))
  (unless fileName
   (set 'df1
       (locate-dominating-file default-directory
                               (lambda
                                 (x)
```

```
                            (directory-files x nil NAMESTEM))
                               ))
  (setq default-directory df1)
  (setq fileName
       (car
        (directory-files default-directory nil NAMESTEM))))
 (unless fileName
  (error
   (concat "No ." EXTENSION " file found in " default-directory)))
 (when
    (string-match-p
     (concat "\\." EXTENSION "$")
     (buffer-name))
  (save-buffer))
 (message "mp-get-file...done")
 (message fileName)
 fileName)
```

3.4 mp-R-nw-or-org *(&optional FILENAME)*

Documentation: Generate an .Rnw or .org file from an .R file with code chunks.

If no FILENAME is supplied, it will try to find the appropriate .R file in the curren directory with 'mp-get-file'.

If there is no .Rnw or .org file in the corresponding directory, it will generate one with 'mp-skeleton'.

If such a file already exists, it will update it with 'mp-update'.

It is called by 'mp-entwine'.

```
(interactive "F FILENAME: ")

(if FILENAME
   (setq FILENAME
        (file-name-nondirectory FILENAME))
  (setq FILENAME
       (mp-get-file "" "R")))

(message "mp-R-nw-or-org...")

(let
   (beg1 end1 elem1 elem2
```

```
      (listOfChunks 'nil))
(save-excursion
  (goto-char
   (point-min))
  (while
     (re-search-forward "## ----"
                        (point-max)
                        t)
    (save-excursion
      (set 'beg1
          (point))
      (move-end-of-line 1)
      (set 'end1
          (point))
      (set 'elem1
          (buffer-substring-no-properties beg1 end1)))
    (if
       (string= mp-entwiner "knitr")
       (set 'elem2 nil)
      (progn
       (move-beginning-of-line 2)
       (set 'beg1
           (point))
       (set 'end1
           (save-excursion
             (search-forward "## ----"
                             (point-max)
                             t)))
       (if end1
          (set 'end1
              (- end1
                 (length "## ----")))
        (set 'end1
            (point-max)))
       (set 'elem2
           (buffer-substring-no-properties beg1 end1))))
    (add-to-list 'listOfChunks
              (list elem1 elem2)
              t)))
(let
   ((fileStem
     (file-name-sans-extension
      (file-name-nondirectory FILENAME)))
```

```
  ext1)
(set 'ext1
    (if
       (string= "Org" mp-entwiner)
       ".org" ".Rnw"))
(set 'elem1
    (member
     (concat fileStem ext1)
     (file-expand-wildcards
      (concat "*" ext1))))
(if elem1
   (mp-update
    (concat fileStem ext1)
    listOfChunks)
  (mp-skeleton
   (concat fileStem ext1)
   listOfChunks))
(message "mp-R-nw-or-org...done")
(if
   (string= "Org" mp-entwiner)
   (mp-org-tex
    (concat fileStem ".org"))
  (mp-nw-tex
   (concat fileStem ".Rnw")))))
```

3.5 mp-skeleton *(FILENAME listOfChunks)*

Documentation: Generate an .Rnw or a .org file from a 'list' of chunks of 'R' code.

If no FILENAME is supplied, it will try to find the appropriate .R file in the curren directory with 'mp-get-file'.

This will read all 'chunks' (specified by '## —- chunkName') from the current .R file.

It makes a basic .Rnw or .org file from the chunks. The preamble for .Rnw files is 'mp-preamble'.

The chunkName is inserted above each chunk, with an optional prefix and suffix. This is ubsection{chunkName} by default; see 'mp-chunk-brackets'.

If 'mp-entwiner' is set to 'Sweave', packages 'Sweave' and 'lmodern' are also added.

If ‘mp-latex’ is set to ’xelatex’, package ’fontspec’ with font settings is added (see ‘mp-xetex-font’).

The preamble for .org files in ‘mp-org-latex-header’. The author is given by ‘user-full-name’, if available, otherwise by ‘user-login-name’.

The file will be saved with the same name as the associated .R file. Buffer options for Org export are set with ‘mp-ox-settings’.

This function may be called by ‘mp-R-nw-or-org’.

```
(interactive "F FILENAME: X listOfChunks: ")

(if FILENAME
   (setq FILENAME
        (file-name-nondirectory FILENAME))
  (setq FILENAME
       (mp-get-file "" "Rnw")))

(find-file FILENAME)

(let
   (preamble1 font1)
  (set 'preamble1 mp-preamble)
  (setq preamble1
       (assq-delete-all nil preamble1))
  (when
     (string= mp-latex "pdflatex")
   (set 'font1 mp-latex-font)
   (setq font1
        (rassq-delete-all nil font1)))
  (unless
     (string= mp-latex "pdflatex")
   (set 'font1 mp-xetex-font)
   (setq font1
        (rassq-delete-all nil font1)))
  (when
     (string= mp-entwiner "Org")
   (defun fun1
      (STRING)
     (let
        (list1)
      (set 'list1
          (split-string STRING "\n"))
      (mapc
```

```
      (lambda
        (x)
        (insert
         (format "#+LATEX_HEADER: %s \n" x)))
      list1)))
  (let
     (head1)
    (set 'head1
        (mapconcat
         (lambda
           (x)
           (when
              (equal 'org
                     (cadr x))
             (cddr x)))
         preamble1 "\n"))
    (fun1 head1)
    (set 'head1
        (mapconcat
         (lambda
           (x)
           (when
              (equal 'all
                     (cadr x))
             (cddr x)))
         preamble1 "\n"))
    (fun1 head1)
    (if
       (string= mp-latex "pdflatex")
       (mapc
        (lambda
          (x)
          (fun1
           (car x)))
        font1)
      (mapc
       (lambda
         (x)
         (fun1
          (car x)))
       font1))))
(when
   (not
```

```
    (string= mp-entwiner "Org"))
  (defun f1
     (KEY)
    (mapc
     (lambda
      (x)
      (when
         (equal KEY
                (cadr x))
        (insert
         (concat
          (cddr x)
          "\n"))))
     preamble1))
  (f1 'class)
  (when
     (string= mp-entwiner "knitr")
    (f1 'knitr))
  (when
     (string= mp-entwiner "Sweave")
    (f1 'sweave))
  (if
     (string= mp-latex "pdflatex")
     (mapc
      (lambda
        (x)
        (insert
         (car x)))
      font1)
    (mapc
     (lambda
       (x)
       (insert
        (car x)))
     font1))
  (when
     (string= mp-entwiner "knitr")
    (kill-whole-line 0))
  (f1 'all)
  (insert "\n%\n
     %-----------------------------------------\
     n%\n\\begin{document}\n%"))
(when
```

```
    (string= mp-entwiner "knitr")
  (insert mp-knitr-opts)
  (insert "\n% knitr read chunks\n<<readChunks, include=FALSE>>=\n")
  (insert
   (concat "read_chunk('" fileStem ".R')"))
  (insert "\n@\n"))
(insert
 (if
    (string= mp-entwiner "Org")
    (concat "#+TITLE: " fileStem)
   (concat "\\title{" fileStem "}")))
(insert "\n")
(let
   (author1)
  (set 'author1
       (if
          (equal user-full-name "")
          user-login-name user-full-name))
  (insert
   (if
      (string= mp-entwiner "Org")
      (concat "#+AUTHOR: " author1)
    (concat "\\author{" author1 "}"))))
(insert "\n\n")
(when
   (not
    (string= mp-entwiner "Org"))
  (insert "\n\\maketitle\n% page numbers appear top-right\n\\pagestyle{
     headings}\n\\tableofcontents\n\n\n"))
(let
   ((i 0)
    (elem1 nil)
    beg1 end1)
  (while
     (< i
        (length listOfChunks))
   (set 'elem1
        (nth i listOfChunks))
   (mp-insert-chunk elem1)
   (incf i)))
(when
   (not
    (string= mp-entwiner "Org"))
```

```
   (insert "\n% \n% \\bibliographystyle{plain}\n% \\bibliography{" fileStem "}\
       n% \\begin{thebibliography}{99}\n% \\bibitem{Smith1900}\n% John A. Smith
       and Terry Jones,\n% mph{An article}.\n% Journal of Things, \n% 1994.\n% \\
       end{thebibliography}\n%\n\\end{document}"))
  (write-region
   (point-min)
   (point-max)
   (buffer-name))
  (set-visited-file-name
   (buffer-name)
   t)
  (save-buffer)
  (when
     (string= mp-entwiner "Org")
    (mp-ox-settings)))
```

3.6 mp-insert-chunk *(CHUNK)*

Documentation: Insert a CHUNK into an existing .Rnw or .org file.

The CHUNK is given as a list, where the 'car' is the name of the chunk and the 'cdr' is the chunk contents.

The CHUNK is enclosed in by 'mp-chunk-brackets'. 'mp-Sweave-opts' or 'mp-org-header-args' will also be added as needed.

```
(set 'beg1
    (if
       (string= mp-entwiner "Org")
       (nth 2 mp-chunk-brackets)
      (nth 0 mp-chunk-brackets)))

(set 'end1
    (if
       (string= mp-entwiner "Org")
       (nth 3 mp-chunk-brackets)
      (nth 1 mp-chunk-brackets)))

(insert
 (concat beg1
       (prin1-to-string
        (first CHUNK)
        t)
       end1 "\n\n"))
```

```
(set 'beg1
    (if
       (string= mp-entwiner "Org")
       "#+NAME:" "<<"))

(set 'end1
    (if
       (string= mp-entwiner "Org")
       "" ">>="))

(insert
 (concat beg1
       (prin1-to-string
        (first CHUNK)
        t)))

(when
   (string= mp-entwiner "Sweave")
  (let
     (sweave1)
   (set 'sweave1 mp-Sweave-opts)
   (setq sweave1
        (rassq-delete-all nil sweave1))
   (insert
    (concat ","
          (mapconcat 'car sweave1 ","))))))

(insert
 (concat end1 "\n"))

(when
   (string= mp-entwiner "Org")
  (let
     (org1)
   (set 'org1 mp-org-header-args)
   (setq org1
        (assq-delete-all nil org1))
   (set 'end1
       (mapconcat
        (lambda
          (x)
          (concat
```

```
          (cadr x)
          " "
          (cddr x)))
       org1 " ")))
  (insert
   (concat "#+begin_src R " end1 "\n")))

(when
   (not
    (string= mp-entwiner "knitr"))
  (insert
   (prin1-to-string
    (second CHUNK)
    t)
   "\n"))

(set 'end1
    (if
       (string= mp-entwiner "Org")
       "#+end_src" "@"))

(insert
 (concat end1 "\n\n"))
```

3.7 mp-update *(&optional FILENAME listOfChunks)*

Documentation: Update an .Rnw or .org file with a 'list' of chunks.

If no FILENAME is supplied, it will try to find the appropriate .R file in the curren directory with 'mp-get-file'.

The 'list' should be in the form of a value-pair, indicating the name and contents of each chunk e.g. ('foo' 'barbarbar').

If 'mp-entwiner' is set to 'Sweave' or 'Org', this function is called by 'mp-R-nw-or-org'.

```
(interactive "F FILENAME: X listOfChunks: ")

(if FILENAME
   (setq FILENAME
        (file-name-nondirectory FILENAME))
  (if
     (string= mp-entwiner "Org")
     (setq FILENAME
```

```
          (mp-get-file "" "org"))
   (setq FILENAME
        (mp-get-file "" "Rnw"))))

(setq org-startup-folded nil)

(find-file FILENAME)

(message "mp-update...")

(let
   ((namesCurrChunks nil)
    prefix1 suffix1 beg1 end1 old1 chunk1 chunkName1
       chunkValue1)
  (set 'prefix1
      (if
         (string= "Org" mp-entwiner)
         "#[+]NAME: " "<<"))
  (set 'suffix1
      (if
         (string= "Org" mp-entwiner)
         "$" ".,"))
  (save-excursion
   (goto-char
    (point-min))
   (while
      (re-search-forward prefix1
                    (point-max)
                    t)
     (save-excursion
      (set 'beg1
          (point))
      (search-forward-regexp suffix1 nil t)
      (set 'end1
          (point))
      (set 'chunk1
          (buffer-substring-no-properties beg1 end1))
      (add-to-list 'namesCurrChunks chunk1 t))))
  (search-forward-regexp prefix1)
  (move-beginning-of-line -2)
  (let
     ((i 0)
      (elem1 nil))
```

```
   (while
      (< i
        (length listOfChunks))
     (set 'elem1
         (nth i listOfChunks))
     (unless
        (set 'old1
            (member
             (car elem1)
             namesCurrChunks))
      (mp-insert-chunk elem1))
     (when old1
      (set 'chunkName1
          (first elem1))
      (set 'chunkValue1
          (second elem1))
      (goto-char
       (point-min))
      (search-forward-regexp
       (concat prefix1 chunkName1)
       (point-max)
       t)
      (move-beginning-of-line
       (if
          (string= "Org" mp-entwiner)
          3 2))
      (set 'suffix1
          (if
             (string= "Org" mp-entwiner)
             "#+end_src" "@"))
      (insert
       (prin1-to-string chunkValue1 t))
      (set 'beg1
          (point))
      (search-forward suffix1 nil t)
      (end-of-line 0)
      (set 'end1
          (point))
      (delete-region beg1 end1)
      (move-beginning-of-line 4))
     (incf i))))

(save-buffer)
```

```
(when
   (string= mp-entwiner "Org")
  (mp-ox-settings))

(message "mp-update...done")
```

3.8 mp-ox-settings *nil*

Documentation: Set 'org-mode' export settings. All variables are set as buffer-only (see 'make-local-variable').

Adds support for 'R', 'latex' and 'emacs-lisp' to 'org-babel-load-languages'. Sets the following to 'nil': 'org-confirm-babel-evaluate' and 'org-latex-with-hyperref' and 'org-latex-table-caption-above'. Sets 'org-latex-listings' to 't'. Adds 'listings' and 'color' to 'org-latex-packages-alist'.

```
(interactive)

(require 'ox-latex)

(require 'ob-R)

(require 'ob-emacs-lisp)

(require 'ob-latex)

(set 'org-startup-folded nil)

(set 'org-latex-table-caption-above nil)

(org-babel-do-load-languages 'org-babel-load-languages
                      '((R . t)
                        (latex . t)
                        (emacs-lisp . t)))

(set
 (make-local-variable 'org-confirm-babel-evaluate)
 nil)

(set
 (make-local-variable 'org-latex-with-hyperref)
 nil)
```

```
(set
 (make-local-variable 'org-latex-listings)
 t)

(add-to-list 'org-latex-packages-alist
             '("" "listings"))

(add-to-list 'org-latex-packages-alist
             '("" "color"))
```

3.9 mp-org-tex *(&optional FILENAME)*

Documentation: Use an .org file to make a .tex (TEX) file.

If no FILENAME is supplied, it will try to find the appropriate .R file in the curren directory with 'mp-get-file'.

```
(interactive "F FILENAME: ")

(if FILENAME
   (setq FILENAME
        (file-name-nondirectory FILENAME))
  (setq FILENAME
       (mp-get-file "" "org")))

(mp-ox-settings)

(message "mp-org-tex...")

(find-file FILENAME)

(save-buffer)

(org-latex-export-to-latex)

(let
   ((fileStem
     (file-name-sans-extension
      (file-name-nondirectory FILENAME))))
  (message "mp-org-tex...done")
  (mp-latexmk
   (concat fileStem ".tex")))
```

3.10 mp-nw-tex *(&optional FILENAME)*

Documentation: Generate a .tex (TEX) file from an .Rnw file. Add 'mp-sweaveSty' and 'mp-upquoteSty' to the current directory if required.

If no FILENAME is supplied, it will try to find the appropriate .R file in the curren directory with 'mp-get-file'.

Once complete, 'mp-latex-pdf' will be run on the output.

```
(interactive "F FILENAME: ")

(if FILENAME
   (setq FILENAME
        (file-name-nondirectory FILENAME))
  (setq FILENAME
       (mp-get-file "" "Rnw")))

(message "mp-nw-tex...")

(unless
   (directory-files default-directory nil "upquote.sty")
  (when
     (assoc t mp-upquoteSty)
   (with-temp-file "upquote.sty"
     (insert
      (cdr
       (assoc t mp-upquoteSty))))))

(let
   ((fileStem
     (file-name-sans-extension
      (file-name-nondirectory FILENAME)))
    procRes defDir)
  (set 'defDir default-directory)
  (unless
     (directory-files default-directory nil
                  (concat fileStem ".Rnw"))
   (mp-R-nw-or-org
    (concat fileStem ".R")))
  (when
     (string= mp-entwiner "knitr")
   (with-temp-file "knit.R"
     (insert
      (concat "knitr::knit('" fileStem ".Rnw')")))
```

```
    (pop-to-buffer "*make-pdf*")
    (setq default-directory defDir)
    (goto-char
     (point-max))
    (set 'procRes
         (call-process "Rscript" nil t t "knit.R"))
    (unless
        (= procRes 0)
      (error
       (concat "Error with knitr"))))
  (when
      (string= mp-entwiner "Sweave")
    (when
        (assoc t mp-sweaveSty)
      (unless
          (directory-files default-directory nil "Sweave.sty")
        (with-temp-file "Sweave.sty"
          (insert
           (cdr
            (assoc t mp-sweaveSty))))))
    (pop-to-buffer "*make-pdf*")
    (setq default-directory defDir)
    (set 'procRes
         (call-process "R" nil t t "CMD" "Sweave"
                       (concat fileStem ".Rnw")))
    (unless
        (= procRes 0)
      (error
       (concat "Error with Sweave"))))
  (mp-latexmk
   (concat fileStem ".tex")))

(message "mp-nw-tex...done")
```

3.11 mp-latexmk *(&optional FILENAME)*

Documentation: Use a .tex (LaTeX, XeTeX) file to make a .pdf file, using the method given by 'mp-latex' and 'latexmk'. Add a 'latexmkrc' file to the 'default-directory' to help with this. If another 'latexmkrc' is on your path, the local copy will override this.

If no FILENAME is supplied, it will try to find the appropriate .tex file

in the curren directory with ‘mp-get-file’.

Once complete it will open the file with ‘mp-view-pdf’. It is called by the functions ‘mp-nw-tex’ and ‘mp-org-tex’.

See the manual for details: URL ‘http://ctan.mackichan.com/support/latexmk/latexmk.pdf’.

```
(interactive "F FILENAME: ")

(unless FILENAME
  (setq FILENAME
       (mp-get-file "" "tex")))

(message "mp-latexmk...")

(let
   (bibBuf extraArgs defDir
          (fileStem
           (file-name-sans-extension
            (file-name-nondirectory FILENAME))))
  (when
     (assoc t mp-latexmkrc)
    (unless
       (directory-files default-directory nil "^.latexmkrc$")
      (with-temp-file ".latexmkrc"
        (insert
         (cdr
          (assoc t mp-latexmkrc)))
        (insert "$pdf_mode = 1;\n$postscript_mode = $dvi_mode = 0; \n")
        (insert
         (concat "$pdflatex = '" mp-latex " %O %S ';\n")))))
  (when
     (set 'bibBuf
          (find-buffer-visiting
           (concat fileStem ".bib")))
    (with-current-buffer bibBuf
      (save-buffer)))
  (set 'extraArgs
       (mapconcat 'car mp-args-latex " "))
  (set 'defDir default-directory)
  (pop-to-buffer "*make-pdf*")
  (setq default-directory defDir)
  (goto-char
   (point-max))
```

```
  (insert "\n\n\n\nRUNNING LATEXMK\n\n")
  (lexical-let
     ((fileStem fileStem))
   (set-process-sentinel
    (start-process-shell-command "async-pdf" "*make-pdf*"
                                 (concat "latexmk " fileStem " "
                                    extraArgs))
    (lambda
      (process event)
      (message "mp-latexmk...done")
      (mp-highlight "*make-pdf*")
      (mp-view-pdf
       (concat fileStem ".pdf"))
      (when
         (not
          (string-match-p "finished" event))
        (error "Error in latexmk"))))))
```

3.12 mp-highlight *(BUFFER)*

Documentation: Highlight important words in output when generating .pdf files.

Runs in the current buffer.

```
(interactive "B Buffer: ")

(unless BUFFER
  (setq BUFFER
       (current-buffer)))

(pop-to-buffer BUFFER)

(defun fun1
    (REGEXP FACE)
  (save-excursion
    (while
       (re-search-forward REGEXP nil t)
     (set-text-properties
      (match-beginning 0)
      (match-end 0)
     FACE))))
```

```
(save-excursion
  (goto-char
   (point-min))
  (fun1 "^Run.*$"
       '(face highlight))
  (fun1 ".arning.*$"
       '(face holiday))
  (fun1 ".itation.*$"
       '(face holiday))
  (fun1 ".eference.*$"
       '(face holiday))
  (fun1 "Error.*$"
       '(face holiday))
  (fun1 "Fatal.*$"
       '(face holiday))
  (fun1 "ignored"
       '(face warning))
  (fun1 "..+?erfull.*$"
       '(face error))
  (fun1 "You can't.*$"
       '(face error)))
```

3.13 mp-view-pdf *(&optional FILENAME)*

Documentation: View FILENAME with 'mp-pdf-viewer'.

```
(interactive "F FILENAME: ")

(if FILENAME
   (setq FILENAME
        (file-name-nondirectory FILENAME))
  (setq FILENAME
       (mp-get-file "" "pdf")))

(start-process-shell-command "mp-view" nil
                      (concat mp-pdf-viewer " " FILENAME))
```

3.14 mp-el-tex *(&optional INCLUDESOURCE FILENAME)*

Documentation: Generate a .tex file from a .el file containing a package.

If INCLUDESOURCE is non nil, the source code for the functions and the default values of the variables in the package are included also.

The keywords to identify in the package preamble are given in 'mp-el-package-attributes'.

It is designed for packages which are contained completely in one file.

```
(interactive
 (list
  (y-or-n-p "Include source? ")
  (read-file-name "File name? ")))

(unless FILENAME
  (setq FILENAME
       (mp-get-file "" "el")))

(unless
   (or INCLUDESOURCE
      (equal INCLUDESOURCE ""))
  (setq INCLUDESOURCE t))

(let
   (preamble1 font1)
  (set 'preamble1 mp-preamble)
  (setq preamble1
       (assq-delete-all nil preamble1))
  (when
     (string= mp-latex "pdflatex")
   (set 'font1 mp-latex-font)
   (setq font1
        (rassq-delete-all nil font1)))
  (unless
     (string= mp-latex "pdflatex")
   (set 'font1 mp-xetex-font)
   (setq font1
        (rassq-delete-all nil font1)))
  (message "mp-el-tex...")
  (let*
     (beg1 end1 r1 elem1
          (l0 'nil)
          (l1 'nil)
          (fileStem
           (file-name-sans-extension
            (file-name-nondirectory FILENAME))))
```

```
(defun fun1
   (KEYWORD)
  (set 'r1
      (concat "^."
            (substring
             (symbol-name KEYWORD)
             1)
            ":?"))
  (save-excursion
    (set 'elem1
        (search-forward-regexp r1 nil t))
    (when elem1
      (set 'beg1
          (point))
      (end-of-line)
      (set 'end1
          (point))
      (set 'elem1
          (buffer-substring-no-properties beg1 end1))))
  (add-to-list 'l0
            (cons KEYWORD elem1)
            t))
(goto-char
 (point-min))
(save-excursion
  (goto-char
   (point-min))
  (set 'beg1
      (point))
  (forward-comment
   (buffer-size))
  (set 'end1
      (point))
  (set 'elem1
      (buffer-substring-no-properties beg1 end1)))
(when elem1
  (with-temp-buffer
    (insert elem1)
    (goto-char
     (point-min))
    (save-excursion
      (while
         (re-search-forward ";;" nil t)
```

```
        (replace-match "" nil nil)))
    (set 'case-fold-search nil)
    (mapc 'fun1 mp-package-attributes)
    (save-excursion
      (when
          (search-forward-regexp ";;;###autoload" nil t)
        (add-to-list 'l0
                     (cons 'autoload "TRUE")
                     t)))
    (search-forward-regexp "[cC]ommentary:?" nil t)
    (set 'beg1
        (point))
    (search-forward "Code" nil t)
    (beginning-of-line -2)
    (set 'end1
        (point))
    (set 'elem1
        (buffer-substring-no-properties beg1 end1))
    (set 'elem1
        (replace-regexp-in-string ";; " "\newline " elem1))
    (add-to-list 'l0
                 (cons 'Commentary elem1)
                 t)
    (set 'l0
        (rassq-delete-all nil l0)))
  l0)
(while
    (not
     (=
      (point)
      (point-max)))
  (forward-sexp)
  (save-excursion
    (set 'beg1
        (point))
    (set 'elem1
        (sexp-at-point))
    (add-to-list 'l1
                 (cons
                  (car elem1)
                  elem1)
                 t)))
(find-file
```

```
 (concat fileStem ".org"))
(erase-buffer)
(defun fun2
   (STRING)
  (let
     (list1)
    (set 'list1
        (split-string STRING "\n"))
    (mapc
     (lambda
       (x)
       (insert
        (format "#+LATEX_HEADER: %s \n" x)))
     list1)))
(let
   (head1)
  (set 'head1
      (mapconcat
       (lambda
         (x)
         (when
            (equal 'org
                   (cadr x))
           (cddr x)))
       preamble1 "\n"))
  (fun2 head1)
  (set 'head1
      (mapconcat
       (lambda
         (x)
         (when
            (equal 'all
                   (cadr x))
           (cddr x)))
       preamble1 "\n"))
  (fun2 head1))
(if
   (string= mp-latex "pdflatex")
   (mapc
    (lambda
      (x)
      (fun2
       (car x)))
```

```
      font1)
    (mapc
     (lambda
       (x)
       (fun2
        (car x)))
     font1))
  (insert "\n")
  (when l0
    (insert "\n\n")
    (insert "* Package Attributes\n\n")
    (mapc
     (lambda
       (x)
       (insert
        (concat " - *"
                (symbol-name
                 (car x))
                "*:"))
       (insert
        (concat
         (cdr x)
         "\n"))
       (insert "\n"))
     l0)
    (insert "\n"))
  (setq l1
        (sort l1 'equal))
  (when
      (assoc 'defgroup l1)
    (insert "\n* Groups\n\n")
    (mapc
     (lambda
       (x)
       (when
           (eq 'defgroup
               (car x))
         (insert
          (concat "** "
                  (symbol-name
                   (nth 1
                        (cdr x)))
                  "\n\n"))
```

```
        (insert
         (documentation-property
          (eval
           (cdr x))
          'group-documentation))
        (insert "\n\n")))
    l1)
   (set 'l1
        (assq-delete-all 'defgroup l1)))
 (when
     (assoc 'defun l1)
   (insert "\n* Functions\n\n")
   (mapc
    (lambda
      (x)
      (when
          (eq 'defun
              (car x))
        (insert
         (concat "** "
                 (symbol-name
                  (nth 1
                       (cdr x)))))
        (insert
         (concat " /"
                 (prin1-to-string
                  (help-function-arglist
                   (eval
                    (cdr x))))
                 "/ \n\n"))
        (insert
         (concat "*Documentation*: "
                 (documentation
                  (eval
                   (cdr x))
                  t)
                 "\n\n"))
        (when INCLUDESOURCE
          (insert "\n#+begin_src lisp \n")
          (set 'elem1
               (indirect-function
                (cdr x)))
          (pop elem1)
```

```
          (pop elem1)
          (pop elem1)
          (while elem1
            (set 'r1
                (pop elem1))
            (when
               (eq 'cons
                  (type-of r1))
              (insert
               (concat
                (pp r1)
                "\n"))))
          (insert "\n#+end_src \n\n"))))
    l1)
   (set 'l1
       (assq-delete-all 'defun l1)))
 (when
    (assoc 'defcustom l1)
   (insert "\n* Variables (customizable)\n\n")
   (mapc
    (lambda
      (x)
      (when
         (eq 'defcustom
            (car x))
        (insert
         (concat "** "
               (symbol-name
                (nth 1
                    (cdr x)))
               "\n\n"))
        (when
           (set 'elem1
               (documentation-property
                (eval
                 (cdr x))
                'variable-documentation t))
          (insert "*Documentation*: \n")
          (insert
           (concat elem1 "\n\n")))
        (when INCLUDESOURCE
          (defun fun3
             (STRING)
```

```
            (insert
             (format "\n%s" STRING)))
           (when
              (set 'elem1
                   (get
                    (nth 1
                         (cdr x))
                    'standard-value))
            (insert "*Standard value*:\n")
            (insert "#+begin_src lisp \n")
            (insert
             (format "%s" elem1))
            (insert "\n#+end_src \n"))
           (when
              (set 'elem1
                   (get
                    (nth 1
                         (cdr x))
                    'custom-type))
            (insert "*Type*:\n")
            (insert "#+begin_src TeX")
            (if
               (sequencep elem1)
               (mapcar
                (lambda
                  (x)
                  (fun3 x))
                elem1)
              (fun3 elem1))
            (insert "\n#+end_src \n"))
           (when
              (set 'elem1
                   (get
                    (nth 1
                         (cdr x))
                    'custom-options))
            (insert "*Options*:\n")
            (insert "#+begin_src lisp")
            (if
               (sequencep elem1)
               (mapcar
                (lambda
                  (x)
```

```
                (fun3 x))
              elem1)
            (fun3 elem1))
          (insert "\n#+end_src \n"))
        (when
           (set 'elem1
               (get
                (nth 1
                    (cdr x))
                'custom-links))
          (insert "*Links*:\n")
          (mapc
           (lambda
             (x1)
             (if
                (set 'r1
                    (member :tag x1))
                (insert
                 (concat "[["
                         (car
                          (last r1))
                         "]["
                         (cadr r1)
                         "]]\n\n"))
              (insert
               (concat "[[" car
                       (last x1)
                       "]]\n\n"))))
           elem1)))))
   l1)
  (set 'l1
       (assq-delete-all 'defcustom l1)))
(when
    (assoc 'defvar l1)
  (insert "\n* Variables\n")
  (mapc
   (lambda
     (x)
     (when
        (eq 'defvar
           (car x))
       (insert
        (concat "\n** "
```

```
                (symbol-name
                 (nth 1
                      (cdr x)))
                "\n"))
        (insert "\n")
        (when
           (set 'elem1
                (documentation-property
                 (eval
                  (cdr x))
                 'variable-documentation t))
          (insert
           (concat "*Documentation*:\n" elem1 "\n\n")))))
    l1)
   (setq l1
        (assq-delete-all 'defvar l1))
   (insert "\n"))
 (when l1
   (setq l1
        (sort l1 'equal))
   (insert "\n* Additional code\n")
   (mapc
    (lambda
      (x)
      (unless
         (equal nil
                (cdr x))
        (insert
         (concat "\n** *"
                 (prin1-to-string
                  (car x))
                 "*\n"))
        (insert
         (format "\n#+begin_src lisp\n%S\n#+end_src"
                 (cdr x)))))
    l1))
 (save-buffer)
 (mp-ox-settings)
 (message "mp-el-tex...done")
 (mp-org-tex
  (concat fileStem ".org"))))
```

4 Variables (customizable)

4.1 mp-entwiner

Documentation:
The method to 'entwine' files in a directory.
One of: 'knitr', 'Sweave' or 'Org' (see 'org-mode').
Standard value:

```
(knitr)
```

Type:

```
radio
(const :doc Default :value knitr)
(const :doc Sweave with .nw :value Sweave)
(const :doc Weave with .org :value Org)
```

4.2 mp-latex

Documentation:
This is the command for generating a .pdf from a .tex (TEX) file. Other options are also possible.
Standard value:

```
(pdflatex)
```

Type:

```
radio
(const :doc pdfTeX. The default. pdflatex)
(const :doc XeTeX. For use with OpenType fonts. xelatex)
(const :doc LuaTeX. For use with 'lua' scripts. lualatex)
```

4.3 mp-args-latex

Documentation:
Alist of command-line arguments to be added to 'mp-latex'.
If non-nil, the argument will be added.
Standard value:

```
((quote ((-interaction=nonstopmode . t))))
```

Type:

```
alist
:key-type
(choice :tag other (string :tag other))
:value-type
(boolean :tag Activate :value nil)
```

Options:

```
-interaction=nonstopmode
-shell-escape
-8bit
-interaction=errorstopmode
-enc
-etex
-mltex
-output-format=pdf
```

4.4 mp-pdf-viewer

Documentation:
This is the command line/ shell command to view a .pdf file.
It is used by the functions 'mp-latex-pdf' and 'mp-latexmk'.
The executable needs to be in your 'exec-path'.
Some useful URLs for downloads are given in the 'custmomize' help.
Standard value:

```
(evince)
```

Type:

```
radio
(const :doc Default. Cross platform. evince)
(const :doc Alternative for Windows. sumatrapdf)
(const :doc Good for Linux/Ubuntu. xpdf)
(const :doc Adobe Acrobat Reader. acrord32)
(string :tag Enter an alternative program here. )
```

Links: Adobe Acrobat Reader
xpdf
sumatra
evince

4.5 mp-preamble

Documentation:

This is the preamble for documents created with 'mp-skeleton-nw'.

It is an alist in the form (KEY1. (KEY2 . VALUE)). If KEY1 is non-nil, the list is included.

KEY2 is a symbol indicating when to include. This may depend on the value of 'mp-entwiner' (knitr, Sweave or Org). If KEY2 is 'class' or 'all' it will be always be included.

VALUE is a string.

The preamble inserted depends on the value of 'mp-entwiner'. For example, when 'mp-entwiner' is set to 'knitr' all elements of the list with KEY1 non-nil and KEY2='knitr' will be added to the preamble.

When editing this with 'customize', Use 'C-j' for carraige return.

See 'TEX-doc' i.e. (TEX-doc packageName) for more details on particular packages.

Standard value:

```
((quote ((t class . %
\documentclass{article}) (t knitr . %
% modified from default setup for knitr
%
\usepackage[]{graphicx}
\usepackage[]{color}
\usepackage{framed}
% recommended with 'knitr'
\usepackage{alltt}
\usepackage{mathtools}
\usepackage[sc]{mathpazo}
\usepackage{geometry}
\geometry{verbose, tmargin=2.5cm, bmargin=2.5cm,
 lmargin=2.5cm, rmargin=2.5cm}
\setcounter{secnumdepth}{2}
\setcounter{tocdepth}{2}
\usepackage{url}
\usepackage{hyperref}
\hypersetup{unicode=true, pdfusetitle}
\hypersetup{bookmarks=true, bookmarksnumbered=true}
\hypersetup{bookmarksopen=true, bookmarksopenlevel=2}
\hypersetup{breaklinks=false, pdfborder={0 0 1}}
\hypersetup{backref=false}
\hypersetup{colorlinks=true}
```

```
\definecolor{myDarkBlue}{rgb}{0, 0, 0.5}
\hypersetup{linkcolor=myDarkBlue}
\hypersetup{citecolor=myDarkBlue}
\hypersetup{pdfstartview={XYZ null null 1}}) (t sweave . %
% these are required for Sweave
\usepackage{Sweave}
\usepackage{lmodern}
% hyperref
\usepackage{color}
\usepackage{hyperref}
\hypersetup{colorlinks=true}
\definecolor{myDarkBlue}{rgb}{0, 0, 0.5}
\hypersetup{linkcolor=myDarkBlue}
\hypersetup{citecolor=myDarkBlue}
) (t org . %
% for multiple plots per .pdf
\usepackage{pdfpages}
% recommended for Org mode export
\definecolor{myGrey}{gray}{0.95}
\definecolor{myDarkBlue}{rgb}{0, 0, 0.5}
\definecolor{myOrange}{rgb}{0.85, 0.23, 0}
\definecolor{darkSkyBlue}{rgb}{0.0, 0.636, 0.85}
\definecolor{steelBlue}{rgb}{0.233, 0.433, 0.6}
\lstloadlanguages{[Auto]Lisp}
\lstloadlanguages{R}
\lstloadlanguages{[LaTeX]TeX}
\lstset{frame=single}
\lstset{framerule=0pt}
\lstset{backgroundcolor=\color{myGrey}}
\lstset{basicstyle=\ttfamily\small}
\lstset{columns=fullflexible}
\lstset{keywordstyle=\color{myOrange}}
\lstset{commentstyle=\color{steelBlue}}
\lstset{stringstyle=\color{darkSkyBlue}\sffamily}
\lstset{showstringspaces=false}
\lstset{breaklines=true}
\lstset{texcl=true}
\lstset{upquote=true}
\hypersetup{colorlinks=true}
\hypersetup{linkcolor=myDarkBlue}
\hypersetup{citecolor=myDarkBlue}) (t all . %
% other useful additions
%
```

```
% for rerunfilecheck:
% no need to rerun to get outlines right
\usepackage{bookmark}
% for nice tables
\usepackage{booktabs}
% for e.g. \formatdate
\usepackage{datetime}
% for SI units
\usepackage{siunitx}
\sisetup{per-mode=symbol}
% for chemical symbols
\usepackage[version=3]{mhchem}
% to use forced 'here'
% e.g. \begin{figure}[H]
\usepackage{float}
% for large numbers of floats
\usepackage{morefloats}
% to keep floats in same section
\usepackage[section]{placeins}
% for tables > 1 page
\usepackage{longtable}))))
```

Type:

```
alist
:tag


:key-type
(boolean :tag Activate :value nil)
:value-type
(cons :tag Cons-cell (choice :tag KEY2. Include in... (sexp
   :tag knitr :value knitr) (sexp :tag sweave :value sweave
   ) (sexp :tag org :value org) (sexp :tag all :value all)
   (sexp :tag class :value class)) (string :tag VALUE))
```

4.6 mp-knitr-opts

Documentation:

Default setup options for 'knitr'.

This is passed to 'chunks' in 'knitr' by 'mp-R-nw-or-org'. This list is not exhaustive.

Common options are given as vectors with a choice indicated by the index, in square brackets.

Standard value:

```
(
% knitr chunks
<<setup, include=FALSE>>=
library(knitr)
### Set global chunk options
opts_chunk$set(
   eval=TRUE,
   ## text results
   echo=TRUE,
   results=c('markup', 'asis', 'hold', 'hide')[1],
   collapse=FALSE,
   warning=TRUE, message=TRUE, error=TRUE,
   split=FALSE, include=TRUE, strip.white=TRUE,
   ## code decoration
   tidy=FALSE, prompt=FALSE, comment='##',
   highlight=TRUE, size='normalsize',
   background=c('#F7F7F7', colors()[479], c(0.1, 0.2, 0.3))
      [1],
   ## cache
   cache=FALSE,
   ## plots
   fig.path=c('figure', 'figure/minimal-'),
   fig.keep=c('high', 'none', 'all', 'first', 'last')[1],
   fig.align=c('center', 'left', 'right', 'default')[1],
   fig.show=c('hold', 'asis', 'animate', 'hide')[1],
   dev=c('pdf', 'png', 'tikz')[1],
   fig.width=7, fig.height=7, #inches
   fig.env=c('figure', 'marginfigure')[1],
   fig.pos=c('', 'h', 't', 'b', 'p', 'H')[1])
### Set R options
options(formatR.arrow=TRUE, width=60)
knit_hooks$set(inline = function(x) {
## if (is.numeric(x)) return(knitr:::format_sci(x, 'latex'))
   highr::hi_latex(x)
})
## uncomment below to change theme
## knit_theme$get()
## opts_knit$set(out.format='latex')
## thm1 <- knit_theme$get('acid')
```

```
## knit_theme$set(thm1)
@)
```

Type:

```
choice
(string :format %v :value )
```

Links: Code chunks and package options
Hooks - knitr documentation

4.7 mp-bib

Documentation:
Standard BibTeX entries. May be useful when editing .bib files.
When editing this with 'customize', Use 'C-j' for carraige return.
Standard value:

```
(
@Comment @Comment Example of a comment.
@Comment @Comment % can also be used, as with .tex files.
@Comment % An article from a magazine or a journal.
@Comment @article{Xarticle,
@Comment author = {Smith, John A. and Jones, Terry},
@Comment title = {Something about things},
@Comment journal = {The Journal of Things},
@Comment %volume = {},
@Comment %number = {},
@Comment %pages = {10--20},
@Comment %month = {},
@Comment %note = {},
@Comment year = {1900}
@Comment }
@Comment % alternative format using quotation marks
@Comment @article{Xarticle,
@Comment author = "",
@Comment title = "",
@Comment journal = "",
@Comment %volume = "",
@Comment %number = "",
@Comment %pages = "",
@Comment %month = "",
@Comment %note = "",
```

```
@Comment year = {XXXX},
@Comment }
@Comment % A published book.
@Comment @book{Xbook,
@Comment author = {},
@Comment title = {},
@Comment publisher = {},
@Comment %volume = {},
@Comment %number = {},
@Comment %series = {},
@Comment %address = {},
@Comment %edition = {},
@Comment %month = {},
@Comment %note = {},
@Comment year = {XXXX}
@Comment }
@Comment % A bound work without a named publisher or sponsor
    .
@Comment @booklet{Xbooklet,
@Comment %author = {},
@Comment title = {},
@Comment %howpublished = {},
@Comment %address = {},
@Comment %month = {},
@Comment %note = {},
@Comment year = {XXXX}
@Comment }
@Comment % An article in a conference proceedings.
@Comment % Same fields as to @inproceedings below.
@Comment @conference{Xconference,
@Comment author = {},
@Comment title = {},
@Comment booktitle = {},
@Comment %editor = {},
@Comment %volume = {},
@Comment %number = {},
@Comment %series = {},
@Comment %pages = {},
@Comment %address = {},
@Comment %month = {},
@Comment %publisher= {},
@Comment %note = {},
@Comment year = {XXXX}
```

```
@Comment }
@Comment % A section of a book *without* its own title.
@Comment @inbook{Xinbook,
@Comment author = {},
@Comment editor = {},
@Comment title = {},
@Comment chapter = {},
@Comment pages = {},
@Comment publisher= {},
@Comment %volume = {},
@Comment %number = {},
@Comment %series = {},
@Comment %type = {},
@Comment %address= {},
@Comment %edition= {},
@Comment %month = {},
@Comment %note = {},
@Comment year = {XXXX}
@Comment }
@Comment % A section of a book having its own title.
@Comment @incollection{Xincollection,
@Comment author = {},
@Comment title = {},
@Comment booktitle= {},
@Comment publisher= {},
@Comment %editor = {},
@Comment %volume = {},
@Comment %number = {},
@Comment %series = {},
@Comment %type = {},
@Comment %chapter= {},
@Comment %pages = {},
@Comment %address= {},
@Comment %edition= {},
@Comment %month = {},
@Comment %note = {},
@Comment year = {XXXX}
@Comment }
@Comment % An article in a conference proceedings.
@Comment @inproceedings{Xinproceedings,
@Comment author = {},
@Comment title = {},
@Comment booktitle = {},
```

```
@Comment %editor = {},
@Comment %volume = {},
@Comment %number = {},
@Comment %series = {},
@Comment %pages = {},
@Comment %address = {},
@Comment %organization = {},
@Comment %publisher = {},
@Comment %month = {},
@Comment %note = {},
@Comment year = {XXXX}
@Comment }
@Comment % Technical manual.
@Comment @manual{Xmanual,
@Comment title = {},
@Comment %author = {},
@Comment %organization = {},
@Comment %address = {},
@Comment %edition = {},
@Comment %month = {},
@Comment %note = {},
@Comment year = {XXXX}
@Comment }
@Comment % Masters thesis.
@Comment @mastersthesis{Xthesis,
@Comment author = {},
@Comment title = {},
@Comment school = {},
@Comment %type = {diploma thesis},
@Comment %address = {},
@Comment %month = {},
@Comment %note = {},
@Comment year = {XXXX}
@Comment }
@Comment % Miscellaneous.
@Comment @misc{Xmisc,
@Comment %author = {},
@Comment %title = {},
@Comment %howpublished = {},
@Comment %month = {},
@Comment %note = {},
@Comment %year = {XXXX}
@Comment }
```

```
@Comment % PhD thesis.
@Comment @phdthesis{Xphdthesis,
@Comment author = {},
@Comment title = {},
@Comment school = {},
@Comment %address = {},
@Comment %month = {},
@Comment %keywords = {},
@Comment %note = {},
@Comment year = {XXXX}
@Comment }
@Comment @proceedings{Xproceedings,
@Comment title = {},
@Comment %editor = {},
@Comment %volume = {},
@Comment %number = {},
@Comment %series = {},
@Comment %address = {},
@Comment %organization = {},
@Comment %publisher = {},
@Comment %month = {},
@Comment %note = {},
@Comment year = {XXXX}
@Comment }
@Comment % Technical report from educational,
@Comment % commercial or standardization institution.
@Comment @techreport{Xtreport,
@Comment author = {},
@Comment title = {},
@Comment institution = {},
@Comment %type = {},
@Comment %number = {},
@Comment %address = {},
@Comment %month = {},
@Comment %note = {},
@Comment year = {XXXX}
@Comment }
@Comment % An unpublished article, book, thesis, etc.
@Comment @unpublished{Xunpublished,
@Comment author = {},
@Comment title = {},
@Comment %year = {},
@Comment %month = {},
```

```
@Comment note = {}
@Comment }
)
```

Type:

```
choice
(string :format %v :value )
```

Links: Wikibooks - LaTeX/Bibliography Management

4.8 mp-Sweave-opts

Documentation:

These options will be added to all code 'chunks' generated with 'Sweave'. The value is used by 'mp-insert-chunk'.

Options are given as a list of (KEY . VALUE) pairs. Non-nil for the VALUE means the argument will be added.

See the manual for more details at URL 'http://www.statistik.lmu.de/~leisch/Sweave/Sweave-manual.pdf'.

Only one plot per chunk is supported by 'Sweave'. Manually changing to fig=TRUE in the .Rnw file generated is one simple approach to including plots.

Other possible options include:

- results=tex
  - if set to 'tex', results will be read as TeX
- echo=TRUE
  - if =FALSE, no R code is included in output
- fig=FALSE
  - if TRUE, a figure for graphics is included
- png=TRUE
  - if =FALSE, no .png graphics are generated
- strip.white=false
  - If =all, **all** blank lines are removed;

  - If =true, blank lines removed from top and bottom

- width=6
- height=6
  - inches, for figures
- print=FALSE
  - If =TRUE, wrap all expressions in print()
- EPS=TRUE
  - If =FALSE, no EPS figures are produced. EPS figures are required for LaTeX but not PDFLaTeX
- keep.source=FALSE
  - If =TRUE, do **not** deparse source before 'echo'ing i.e. include original source 'as-is'
- quiet=FALSE
  - If =TRUE, **all** progress messages are suppressed
- split=FALSE
  - If =TRUE, split over multiple files
- term=TRUE
  - If =FALSE, only output from print() and cat() is 'echo'ed

Standard value:

```
((quote ((results=verbatim . t) (results=tex) (echo=TRUE . t
   ) (fig=FALSE . t) (png=TRUE . t) (strip.white=false . t)
   )))
```

Type:

```
alist
:key-type
(choice :tag other (string :tag other))
:value-type
(boolean :tag Activate :value nil)
```

Options:

```
results=verbatim
results=tex
echo=TRUE
fig=FALSE
png=TRUE
strip.white=false
strip.white=all
strip.white=true
width=6
height=6
print=FALSE
EPS=TRUE
keep.source=FALSE
quiet=TRUE
```

4.9 mp-org-header-args

Documentation:

These options will be added to all code 'chunks' when 'mp-entwiner' is set to 'Org'. The value is used by 'mp-insert-chunk'.

Options are given as a list of (KEY1 . (KEY2 . VALUE)) pairs. Non-nil for KEY1 means the argument will be added.

For multiple plots per chunk, export the results to 'x.pdf' then in the .tex docuement change 'includegraphics' to 'includepdf' (from the LaTeX package 'pdfpages').

There is a link to help on the customize screen.

Standard value:

```
((quote ((t :session . *session*) (t :exports . both) (t :
   results . output) (t :results . verbatim) (t :results .
   code))))
```

Type:

```
alist
:key-type
(boolean :tag Activate :value nil)
:value-type
(choice :tag header (cons :tag :session (const :value :
   session) (string :value *session* :doc name of process
```

```
in which to run code; if absent then chunks will be run
in independent sessions)) (cons :tag :file (const :file)
 (string :value "f1.pdf" :doc Use double quotes around
name)) (cons :tag :exports (const :value :exports) (
choice :tag Export what? (const :value both :doc code
and results) (const :value code) (const :value results)
(const :value none))) (cons :tag :results, class = '
collection' (const :results) (choice :tag Collect what?
(const value :doc final value) (const output :doc all
ouput))) (cons :tag :results, class='type' (const :
results) (choice :tag Convert to... (const :value table
:doc table/ vector) (const :value list :doc org-mode
list) (const :value verbatim :doc verbatim/ scalar) (
const :value file :doc path/to/file) (const :value
graphics :doc need to specify :file also))) (cons :tag :
results, class='format' (const :results) (choice :tag
Wrap in... (const :value raw :doc nothing; insert as-is)
 (const :value org :doc BEGIN_SRC org block) (const :
value html :doc BEGIN_HTML block) (const :value latex :
doc BEGIN_LaTeX block) (const :value code :doc parsable
code block) (const :value pp :doc code block for pretty
printing; elisp, python, ruby) (const :value drawer :doc
 a RESULTS drawer))) (cons :tag :results, class = '
collection' (const :results) (choice :tag Collect what?
(const :value value :doc final value) (const :value
output :doc all ouput))) (cons :tag :results, class='
handling' (const :results) (choice :tag What happens (
const :value replace :doc overwrite) (const :value
silent :doc echo in minibuffer) (const :value append) (
const :value prepend))) (cons :tag :cache (const :value
:cache) (choice (const :value yes) (const :value no))) (
cons :tag other (string :other) (string other)))
```

Links: orgmode manual - header arguments

4.10 mp-latex-font

Documentation:

This variable is used by 'mp-skeleton-nw'.

It specifies the default font(s) to use when 'mp-latex' is set to 'pdflatex' (XeTeX) or 'lualatex'.

Some common options are given in the form:

- serif
- sans-serif
- monospace font

Standard value:

```
((quote ((%
% Palatino family
\usepackage[T1]{fontenc}
\usepackage{mathpazo}
\linespread{1.05}
\usepackage[scaled]{helvet}
\usepackage{courier} . t) (%
% Times family
\usepackage[T1]{fontenc}
\usepackage{mathptmx}
\usepackage[scaled=.90]{helvet}
\usepackage{courier}) (%
% Garamond family
\usepackage[T1]{fontenc}
\usepackage[urw-garamond]{mathdesign}
\usepackage{lmodern}
\usepackage{courier}
\linespread{1.0609}) (%
% KP (Kepler) family; displays math
\usepackage[T1]{fontenc}
\usepackage{kpfonts}) (%
% Nimbus family
\usepackage[T1]{fontenc}
\usepackage{tgtermes}
\usepackage[scale=.85]{tgheros}
\usepackage{tgcursor})))))
```

Type:

```
alist
:key-type
(string :tag value)
:value-type
(boolean :tag Activate :value nil)
```

Links: Font combinations per Org manual

4.11 mp-xetex-font

Documentation:

This variable is used by 'mp-skeleton-nw'.

It specifies the default font(s) to use when 'mp-latex' is set to 'xelatex' (XeTeX) or 'lualatex'.

- Some common commands include:
  - setmainfont
  - setsansfont
  - setmonofont
- Some common fonts and their TEX Gyre equivalents include:

Origin	TEX Gyre
Palatino	Pagella
Times	Termes
Helvetica	Heros
ITC Avant Garde Gothic	Adventor
New Century Schoolbook	Schola

- Common font options include:
  - Ligatures
    - Required/NoRequired,
    - Common/NoCommon,
    - Rare/Discretionary,
    - Historic, TEX
  - Letters

    Uppercase SmallCaps UppercaseSmallCaps
    PetiteCaps UppercasePetiteCaps Unicase
  - Numbers
    - Uppercase, Lowercase
    - Proportional, Monospaced
    - SlashedZero
    - Arabic

- Fractions
  - On, Alternate
- Style

Alternate	Italic	Swash
Historic	TitlingCaps	

Standard value:

```
((quote ((%
% Allows the use of OpenType fonts
% Needs to be placed after maths font packages
% particularly 'euler'
\usepackage{fontspec}
\defaultfontfeatures{Mapping=tex-text}
% Main text font
\setmainfont[Ligatures={Rare, TeX, NoCommon}, Numbers={
   Lowercase}]{Linux Libertine O}
\fontsize{12 pt}{16 pt}
\selectfont
 . t))))
```

Type:

```
alist
:key-type
(choice :tag other (string :tag other :format
Takes the form \setfont[options]{font} %v))
:value-type
(boolean :tag Activate :value nil)
```

Links: Free fonts at fontsquirrel
Fontspec documentation

4.12 mp-chunk-brackets

Documentation:

Brackets placed before and after each chunkName. Specify these in the form of a list of strings as follows: (openingForLaTeX closingForLaTeX openingForOrg closingForOrg)

This variable is used by 'mp-insert-chunk'.

The names are specified in the corresponding .R file by '## —- chunkName'.

This may be used to enclose chunkName in a TEX command e.g. 'subsection{'chunkName'}'. The corresponding Org headline or level prefix, '* ', is used when generating a skeleton .org file. Org uses up to 8 headline levels.

See also 'org-level-faces' and 'org-heading-components'.

Standard value:

```
((quote (\subsection{ } ** )))
```

Type:

```
choice
:tag

(list :tag subsection (string :tag LaTeX \subsection{) (
   string :tag LaTeX }) (string :tag org :value ** ) (
   string :tag org :value ))
(list :tag section (string :tag LaTeX :value \section{) (
   string :tag LaTeX :value }) (string :tag org :value * )
   (string :tag org :value ))
(list :tag subsubsection (string :tag LaTeX :value \
   subsubsection{) (string :tag LaTeX :value }) (string :
   tag org :value *** ) (string :tag org :value ))
(list :tag paragraph/list item (string :tag LaTeX :value \
   paragraph{) (string :tag LaTeX :value }) (string :tag
   org :value **** ) (string :tag org :value ))
(list :tag no section (string :tag LaTeX :value ) (string :
   tag LaTeX :value ) (string :tag org :value ) (string :
   tag org :value ))
(list :tag other (string :tag LaTeX :value prefix) (string :
   tag LaTeX :value suffix) (string :tag org :value prefix)
    (string :tag org :value suffix))
```

4.13 mp-sweaveSty

Documentation:

The file 'Sweave.sty'.

This is given in the form (KEY . VALUE) where VALUE is a string. If any KEY is t and if no such file is found in the 'default-directory', the first value which is t will be included in the working directory as 'Sweave.sty'.

It is essential to have 'Sweave.sty' on your path when 'mp-entwiner' is set to 'Sweave'. This variable is used by 'mp-nw-tex'.

When editing this with 'customize', Use 'C-j' for carraige return.
Standard value:

```
((quote ((t .
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{Sweave}{}
%
\RequirePackage{ifthen}
\newboolean{Sweave@gin}
\setboolean{Sweave@gin}{true}
\newboolean{Sweave@ae}
\setboolean{Sweave@ae}{true}
%
\DeclareOption{nogin}{\setboolean{Sweave@gin}{false}}
\DeclareOption{noae}{\setboolean{Sweave@ae}{false}}
\ProcessOptions
%
\RequirePackage{graphicx,fancyvrb}
\IfFileExists{upquote.sty}{\RequirePackage{upquote}}{}
%
\ifthenelse{\boolean{Sweave@gin}}{\setkeys{Gin}{width=0.8\
   textwidth}}{}%
\ifthenelse{\boolean{Sweave@ae}}{%
  \RequirePackage[T1]{fontenc}
  \RequirePackage{ae}
}{}%
%
\DefineVerbatimEnvironment{Sinput}{Verbatim}{fontshape=sl}
\DefineVerbatimEnvironment{Soutput}{Verbatim}{}
\DefineVerbatimEnvironment{Scode}{Verbatim}{fontshape=sl}
%
\ifdefined\Schunk%
  \message{\string Environment Schunk is already defined,
     stay with former definition}%
\else
  \newenvironment{Schunk}{}{}%
\fi
%
\newcommand{\Sconcordance}[1]{%
  \ifx\pdfoutput\undefined%
  \csname newcount\endcsname\pdfoutput\fi%
  \ifcase\pdfoutput\special{#1}%
  \else%
```

```
  \begingroup%
    \pdfcompresslevel=0%
    \immediate\pdfobj stream{#1}%
    \pdfcatalog{/SweaveConcordance \the\pdflastobj\space 0 R
       }%
  \endgroup%
 \fi}
))))
```

Type:

```
alist
:key-type
(boolean :tag Activate :value nil)
:value-type
(string :format %v :value )
```

4.14 mp-latexmkrc

Documentation:

The file '.latexmkrc'.

This is given in the form (KEY . VALUE) where VALUE is a string. If any KEY is t and if no such file is found in the 'default-directory', the first value which is t will be included in the working directory as '.latexmkrc'.

It is an initialization file or 'runcom'(commands) or for latexmk.

This variable is used by 'mp-latexmk'. It is recommended to have this on your path. It supports the use of glossaries, acronymns and indices amongst others. No support is provided for the (now deprecated) 'glossary' package.

A link to the source is available as a link in the custmomize help.

When editing this with 'customize', Use 'C-j' for carraige return.

Standard value:

```
((quote ((t .
# Custom dependency for 'glossaries' package
add_cus_dep('glo', 'gls', 0, 'makeglo2gls');
sub makeglo2gls{
 system("makeindex -s \"$_[0].ist\" -t \"$_[0].glg\" -o \"$_[0].gls\" \"$_[0].glo\" ");
}
# The' glossaries' package, with the [acronym] option,
# produces a .acn file when processed with (xe/pdf)latex and
```

```
# then makeindex to process the .acn into .acr and
# finally runs of (xe/pdf)latex to read in the .acr file.
add_cus_dep('acn', 'acr', 0, 'makeacn2acr');
sub makeacn2acr{
 system("makeindex -s \"$_[0].ist\" -t \"$_[0].alg\" -o \"$_[0].acr\" \"$_[0].acn\" ")
    ;
}
# Example of an added custom glossary type that is used
# in some of the 'glossaries' example files
# This is for the 'new glossary type' command
# \newglossary[nlg]{notation}{not}{ntn}{Notation}
add_cus_dep('ntn', 'not', 0, 'makentn2not');
sub makentn2not{
 system("makeindex -s \"$_[0].ist\" -t \"$_[0].nlg\" -o \"$_[0].not\" \"$_[0].ntn\" ")
    ;
}
# Dependencies for custom indexes using the 'index' package
 add_cus_dep('adx', 'and', 0, 'makeadx2and');
sub makeadx2and{
 system("makeindex -o \"$_[0].and\" \"$_[0].adx\" ");
}
add_cus_dep('ndx', 'nnd', 0, 'makendx2nnd');
sub makendx2nnd {
 system("makeindex -o \"$_[0].nnd\" \"$_[0].ndx\" ");
}
add_cus_dep('ldx', 'lnd', 0, 'makeldx2lnd');
sub makeldx2lnd{
 system("makeindex -o \"$_[0].lnd\" \"$_[0].ldx\" ");
}
# Custom dependency and function for 'nomencl' package
add_cus_dep('nlo', 'nls', 0, 'makenlo2nls');
sub makenlo2nls{
 system("makeindex -s nomencl.ist -o \"$_[0].nls\" \"$_[0].nlo\" ");
}
# Custom dependency and function(s) for 'epstopdf' package
# deletes an outdated pdf-image, and triggers a pdflatex-run
   :
# add_cus_dep( 'eps', 'pdf', 0, 'cus_dep_delete_dest' );
# FOR USERS OF epstopdf v1.5 and later ONLY:
# load it as \usepackage[update,prepend]{epstopdf}
# detects an outdated pdf-image, and triggers a pdflatex-run
# Custom dependecy to convert tif to png
# add_cus_dep('eps', 'pdf', 0, 'cus_dep_require_primary_run
```

```
    ');
# add_cus_dep('tif', 'png', 0, 'maketif2png');
# sub maketif2png{
# system("convert \"$_[0].tif\" \"$_[0].png\" ");
# }
))))
```

Type:

```
alist
:key-type
(boolean :tag Activate :value nil)
:value-type
(string :format %v :value )
```

Links: example: pdflatexmkrc

4.15 mp-upquoteSty

Documentation:

The file 'upquote.sty'.

This is given in the form (KEY . VALUE) where VALUE is a string. If any KEY is t and if no such file is found in the 'default-directory', the first value which is t will be included in the working directory as 'upquote.sty'.

This variable is used by 'mp-nw-tex'. This is favored by 'knitr' and 'Sweave' to improving code display.

When editing this with 'customize', Use 'C-j' for (carraige) return/ newline.

Standard value:

```
((quote ((t .
%% This is file `upquote.sty',
%% generated with the docstrip utility.
%%
%% The original source files were:
%% upquote.dtx (with options: `package')
%%
%% Copyright (C) 2000 by Michael A. Covington
%% Copyright (C) 2003 by Frank Mittelbach
%% Copyright (C) 2012 by Markus Kuhn (current maintainer)
%%
%% Released under the LaTeX Project Public License v1.3c or
   later
```

```
%% See http://www.latex-project.org/lppl.txt
%%
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{upquote}
  [2012/04/19 v1.3 upright-quote and grave-accent glyphs in
      verbatim]
\newcommand\upquote@cmtt{cmtt}
\newcommand\upquote@OTone{OT1}
\ifx\encodingdefault\upquote@OTone
 \ifx\ttdefault\upquote@cmtt\else\RequirePackage{textcomp}\
    fi
\else
 \RequirePackage{textcomp}
\fi
\begingroup
\catcode`'=\active
\catcode``=\active
\g@addto@macro\@noligs
  {\let'\textquotesingle
   \let`\textasciigrave
   \ifx\encodingdefault\upquote@OTone
   \ifx\ttdefault\upquote@cmtt
   \def'{\char13 }%
   \def`{\char18 }%
   \fi\fi}
\endgroup
\endinput
%% End of file `upquote.sty'.
))))
```

Type:

```
alist
:key-type
(boolean :tag Activate :value nil)
:value-type
(string :format %v :value )
```

4.16 mp-package-attributes

Documentation:

Keywords to search for in the initial comments of a package in an .el file.

These are used by ‘mp-el-tex’.

These are all read to the ‘end-of-line’.

Standard value:

```
((quote (Filename Copyright Author Maintainer Created
   Version Keywords Homepage Package-Version
   Package-Requires License URL Doc Keywords Compatibility)
   ))
```

Type:

```
sexp
```

5 Variables

5.1 mp-minor-mode-map

Documentation:

Defines the keymap for this minor mode.

The only keymap used by default is ’C-M-|’ for ‘mp-mp’.

6 Additional code

6.1 require

```
(require (quote org))
```

6.2 define-minor-mode

```
(define-minor-mode mp-mode "
Define mp mode to make .pdfs.

Mp mode is a global minor mode.

It's LIGHTER (displayed on the mode line) is ' mp '." t " mp "
   mp-minor-mode-map :group (quote mp) :global t :version
   0.1 (message "mp mode toggled"))
```

6.3 provide

```
(provide (quote mp))
```