Basecamp

Jeffrey Szymanski

Documentation for my general computing set up and operation.

Setup and Configuration

Good practice for base computing setup

- Bash
  - The default bashrc lives at *etc/skel*.bashrc.
  - Other programs such as singularity or conda may write directly to bashrc.
  - Otherwise, bashrc is not altered automatically, as through Org-mode tangling. Instead, additional code is sourced though a simple if else statement. I source my basecamp shell libraries as follows:
    []bash
    for file in "$HOME"/repos/basecamp/lib/*.sh; do if [-f "file"]; then source "$file" fi done
  - Use functions instead of aliases. Functions are more flexible, and can be debugged.

An opinionated and incomplete base computing configuration

- Base operating system

  My base operating system is linux Ubuntu using a long-term support (LTS) version.

- Bash

- Git repositories Git repos live in ${HOME}/repos.

  Git repos are modular using the git submodule structure.

  Git repos are pushed to GitHub.

  Git version control follows a simple trunk-based development strategy, with a single active branch: master. Major changes can be spun out into separate branches, but should be quickly merged back to master after changes are validated. Stable, validated versions are occasionally saved as git tags.

  - Repositories where I am the main or sole author
    Repository code is tangled from a single Org-mode file. Updates from others are through pull requests as tangling would overwrite any direct commits.

- Core directory structure – /

- Data governance
  - Security
  - Architecture
  - Lifecycle
  - Management
    * Backup

- Core applications core components (incomplete)
  - installed via apt
    * syncthing

    - i3
    - emacs
    - texlive
    - texlive
    - rclone
    - okular
    - libreoffice
    - minor
        - tree
- R
- R
    - [ ] convert python and R to argparse https://www.google.com/search?q=r+argparse+optparse&oq=r+argparse+optparse&gs_lcrp=EgZjaHJvbWUyBggAEEUYOdIBCDg2MDFqMGo0qAIAsAIA&sourceid=chrome-mobile&ie=UTF-8
- Python
    - [ ] convert python and R to argparse https://www.google.com/search?q=r+argparse+optparse&oq=r+argparse+optparse&gs_lcrp=EgZjaHJvbWUyBggAEEUYOdIBCDg2MDFqMGo0qAIAsAIA&sourceid=chrome-mobile&ie=UTF-8
- Emacs
    - Initialization
      When my Emacs loads, several configuration files and directories are references by symbolic link from git repositories to the default ~/.emacs.d directory:
        - init.el
        - public_emacs_config.el
        - private_emacs_config.el
        - public_yasnippets
        - private_yasnippets

      These are loaded as optional files and if not found, they will throw a warning on initialization. These files are generated by Org-mode tangle.
      The init.el file is compiled from the Org-mode header a simple wrapper for other Elisp files.
      My public customized Emacs initialization is compiled from the Org-mode header basecamp.org::Public configuration and tangles to ./config/public_emacs_config.el.
        - Style and good practice
          Use-package is my preferred package management system. By default, use-package loads are structured as:
          (use-package package2 :ensure t :init :config)
          Each use-package keyword will expect a list or needs to be removed. Other keywords exist (see documentation).
        - :ensure t will install any missing package
        - :init is evaluated before package loading
        - :config is evaluated after package loading
    - Org-mode
        - Policy, style, and good practice
        - Block structures, source code, and tangling
          Lowercase is preferred for all block notation, *e.g.*
          instead of
          At the file level, tangled code should reference it's location in orgmode files. tangle defaults to ./
    - Templating with YASnippets My public YASnippets are compiled from source code blocks across my Org-mode agenda files. These tangle into appropriate subdirectories of ./emacs/public_yasnippets. Private snippets are tangled into appropriate subdirectories of org/emacs/private_yasnippets.

General Style Guide Good Practice

- Prefer single-word file and directory names
- For multi-word file and directory names, prefer a dash (-) separator (*e.g.* a-longer-file-name.txt)

Conda

For repositories and projects with heavy use of Python and R, software should be managed through the Conda package manager.

My Basecamp Conda Environment

A basecamp conda environment is stored in this repository at basecamp_env.yaml.

Emacs

- file:///home/jeszyman/.emacs.d/backup-save-list/
- Emacs
  - Custom functions with names that would clash with existing function names from base emacs or packages shall be prefixed with "custom-"
  - Referencing external variables https://chatgpt.com/c/c9789fc7-81f8-4af4-96a2-82ffd94db68c

I install a more recent, non-apt emacs version from https://ftp.wayne.edu/gnu/emacs/ (v29.4 as of *[2024-07-05 Fri]*). []bash cd /tmp
wget https://mirrors.ocf.berkeley.edu/gnu/emacs/emacs-29.4.tar.xz
tar -xf /tmp/emacs-29.4.tar.xz
sudo apt-get install -y libwebp-dev xaw3dg libxpm4 libpng16-16 zlib1g libjpeg8 libtiff5 libgif7 librsvg2-2 librsvg2-dev libsqlite3-dev liblcms2-dev imagemagick libmagickwand-dev pkg-config libxaw7-dev libgpm-dev libgconf-2-4 libgconf2-dev libm17n-dev libotf-dev libxft-dev libsystemd-dev libjansson-dev libtree-sitter-dev libgtk-3-dev libwebkit2gtk-4.0-dev libacl1-dev
cd /tmp/emacs-29.4
./configure –prefix=/usr/local –with-xwidgets –with-imagemagick
make
sudo make install

Simple Emacs Lisp Tutorial

Alist
alist: association list, stores key-value pairs []common-lisp (setq my-alist '((key1 . value1) (key2 . value2) (key3 . value3)))
(assoc 'key2 my-alist) ;; Returns (key2 . value2) (cdr (assoc 'key2 my-alist)) ;; Returns value2
(setq my-alist (cons '(key4 . value5) my-alist)) ;; Adds (key4 . value4) to the front of the alist (setq my-alist (assoc-delete-all 'key2 my-alist)) ;; Removes the element with key 'key2'
[]common-lisp (setq my-alist '((key1 . value1) (key2 . value2) (key3 . value3)))
(prin1 (assoc 'key2 my-alist)) ;; Display (key2 . value2) (princ "") ;; Add a newline for readability (prin1 (cdr (assoc 'key2 my-alist))) ;; Display value2 (princ "") ;; Add a newline for readability
(setq my-alist (cons '(key4 . value4) my-alist)) ;; Adds (key4 . value4) to the front of the alist (prin1 my-alist) ;; Display the updated alist (princ "") ;; Add a newline for readability
(setq my-alist (assoc-delete-all 'key2 my-alist)) ;; Removes the element with key 'key2' (prin1 my-alist) ;; Display the final alist
progn: special form to evaluate sequence of expressions []common-lisp (progn (message "First") (message "Second"))

Org-mode

If you place an asterisk at the beginning of your search, Org-mode will search only headlines (and not entry text). E.g., to find all entries with "emacs" in the headline, you could type:
[]common-lisp C-c a s [+-]Word/Regexp ...: *+emacs
See also Emacs and Org-mode use in my LaTeX repository (GitHub, local buffer).

- Org File Standard Setup

Public configuration

for config rewrite https://chatgpt.com/c/e16840b5-355a-42e3-bd97-70693daa17c0 Visited files are backed up to ~/.emacs.d/back

- Compiled elisp file
- Org-mode
- Base Emacs
  - Load first []common-lisp (add-to-list 'exec-path "/usr/local/bin")

    []common-lisp ;(cua-selection-mode t) ;(setq mark-even-if-inactive t) ;; Keep mark active even when buffer is inactive ;(transient-mark-mode 1) ;; Enable transient-mark-mode for visual selection (scroll-bar-mode 'right) ;; Place scroll bar on the right side

    (setq org-export-backends '(ascii html latex odt icalendar md org)) ; This variable needs to be set before org.el is loaded.
  - Needed in –batch Code that needs to be tangled both here and into custom inits for batch export #+name need_in_batch []common-lisp (setq large-file-warning-threshold most-positive-fixnum) ; disable large file warning (setq-default cache-long-scans nil)
  - Appearance []common-lisp ; — General — ; ; ————————- ;

    (setq frame-background-mode 'dark) (setq inhibit-splash-screen t)

    ; — Windows — ; ; ————————- ;

    ;; Remove bars: (menu-bar-mode -1) (tool-bar-mode -1) (scroll-bar-mode -1) ; ; Fringe- Set finge color to background ;https://emacs.stackexchange.com/a/31944/11502 (set-face-attribute 'fringe nil :background nil)

    ; — Lines — ; ; ———————— ;

    ;; ;; Enable visual line mode (global-visual-line-mode 1) ;; ;; Line highlighting in all buffers (global-hl-line-mode t) ;; ;; Line numbers (global-display-line-numbers-mode 0) ;;; ;;; Disable line numbers by buffer (dolist (mode '(org-mode-hook term-mode-hook shell-mode-hook eshell-mode-hook)) (add-hook mode (lambda () (display-line-numbers-mode 0)))) ;; (setq-default indicate-empty-lines t)

    ; Do not wrap lines, but extend them off screen (setq default-truncate-lines nil)

    ;; no line numbers (setq global-linum-mode nil)

    ; — Syntax Highlighting — ; ; ——————————————- ;

    ;; When enabled, any matching parenthesis is highlighted (show-paren-mode) ;; ;; Enables highlighting of the region whenever the mark is active (transient-mark-mode 1)

    ; — Code — ; ; ————————- ;

    ;; Delimiters (use-package rainbow-delimiters :hook (prog-mode . rainbow-delimiters-mode))

    ; — Faces — ; ; ———————— ;

    ;; ?Fix broken face inheritance (let ((faces (face-list))) (dolist (face faces) (let ((inh (face-attribute face :inherit))) (when (not (memq inh faces)) (set-face-attribute face nil :inherit nil)))))

    ; — Text — ; ; ————————- ;

    ;https://emacs.stackexchange.com/questions/72483/how-to-define-consult-faces-generically-for-minibuffer-highlighting-that-fits-wi (global-hl-line-mode 1) (set-face-attribute 'highlight nil :background "294F6E")
  - Tramp []common-lisp (setq tramp-default-method "ssh")

    (defadvice tramp-completion-handle-file-name-all-completions (around dotemacs-completion-docker activate) "(tramp-completion-handle-file-name-all-completions ˝˝ ˝/docker:ṙeturns a list of active Docker container names, followed by colons." (if (equal (ad-get-arg 1) "/docker:") (let* ((dockernames-raw (shell-command-to-string "docker ps | awk '$NF != "NAMES"'"))(dockernames(cl-remove-if-not'(lambda(dockerline)(string-match" :" dockerline)) (split-string dockernames-raw ""))))) (setq ad-return-value dockernames)) ad-do-it))

    ; https://emacs.stackexchange.com/questions/29286/tramp-unable-to-open-some-files (setq tramp-copy-size-limit 10000000)

- Key bindings []common-lisp ; ASCII Arrows

  ; — ASCII Arrows — ; ; ———————— ;

  (global-set-key (kbd "C-<right>") (lambda () (interactive) (insert "Ž192"))) (global-set-key (kbd "C-<up>") (lambda () (interactive) (insert "Ž191")))

  ; — Disable Keys — ; ; ———————— ;

  ;; Minimize (global-unset-key (kbd "C-z")) ;; Print (global-unset-key (kbd "s-p"))

  (global-set-key (kbd "C-S-n") (lambda () (interactive) (next-line 10))) (global-set-key (kbd "C-S-p") (lambda () (interactive) (next-line -10)))

- On-save hooks and backup []common-lisp ;; Shorthand for save all buffers ;; https://stackoverflow.com/questions/15254414/how-to-silently-save-all-buffers-in-emacs (defun save-all () (interactive) (save-some-buffers t))

  ; — Saving And Backup — ; ; ———————————— ;

  ; Delete trailing whitespace on save (add-hook 'before-save-hook 'delete-trailing-whitespace)

  ;; Backup process upon save

  (setq vc-make-backup-files t) ; Allow old versions to be saved (setq delete-old-versions 20) ; Save 20 (setq backup-directory-alist '(("." . " /.emacs.d/backup-save-list"))) ; Save them here

  (setq auto-save-visited-mode t) ; Visited files will be auto-saved

  (setq auto-save-file-name-transforms `((".*" ,(concat user-emacs-directory "auto-save-list/") t)))

- Miscellaneous []common-lisp

  ; — Miscellaneous — ; ; ————————————- ;

  ;https://emacs.stackexchange.com/questions/62419/what-is-causing-emacs-remote-shell-to-be-slow-on-completion (defun my-shell-mode-setup-function () (when (and (fboundp 'company-mode) (file-remote-p default-directory)) (company-mode -1)))

  (add-hook 'shell-mode-hook 'my-shell-mode-setup-function)

  ;; delete the region when typing, just like as we expect nowadays. (delete-selection-mode t)

  (setq explicit-shell-file-name "/bin/bash")

  ;; Don't count two spaces after a period as the end of a sentence. (setq sentence-end-double-space nil)

  ;; don't check package signatures ;; https://emacs.stackexchange.com/questions/233/how-to-proceed-on-package-el-signature-check-failure (setq package-check-signature nil)

  ;; Avoid nesting exceeds max-lisp-eval-depth error ;; https://stackoverflow.com/questions/11807128/emacs-nesting-exceeds-max-lisp-eval-depth (setq max-lisp-eval-depth 1200)

  ;; allow remembering risky variables ;; https://emacs.stackexchange.com/questions/10983/remember-permission-to-execute-risky-local-variables (defun risky-local-variable-p (sym optional $_{ignored})nil$)

; Disable "buffer is read only" warning ;;https://emacs.stackexchange.com/questions/19742/is-there-a-way-to-disable-the-buffer-is-read-only-warning (defun my-command-error-function (data context caller) "Ignore the buffer-read-only signal; pass the rest to the default handler." (when (not (eq (car data) 'buffer-read-only)) (command-error-default-function data context caller)))

(setq command-error-function 'my-command-error-function)

; Follow symlinks in dired ;;https://emacs.stackexchange.com/questions/41286/follow-symlinked-directories-in-dired (setq find-file-visit-truename t)

(setq browse-url-browser-function 'browse-url-generic browse-url-generic-program "/usr/bin/brave-browser")

; y or n instead of yes or no (setopt use-short-answers t)

;; don't check package signatures ;; https://emacs.stackexchange.com/questions/233/how-to-proceed-on-package-el-signature-check-failure (setq package-check-signature nil)

;; Avoid nesting exceeds max-lisp-eval-depth error ;; https://stackoverflow.com/questions/11807128/emacs-nesting-exceeds-max-lisp-eval-depth (setq max-lisp-eval-depth 1200) ;;

;; normal c-c in ansi-term ;; https://emacs.stackexchange.com/questions/32491/normal-c-c-in-ansi-term (eval-after-load "term" '(progn (term-set-escape-char ?-c) (define-key term-raw-map (kbd "C-c") nil)))

(setq comint-scroll-to-bottom-on-output t)

;; allow kill hidden part of line ;; https://stackoverflow.com/questions/3281581/how-to-word-wrap-in-emacs (setq-default word-wrap t)

;; auto-refresh if source changes ;; https://stackoverflow.com/questions/1480572/how-to-have-emacs-auto-refresh-all-buffers-when-files-have-changed-on-disk (global-auto-revert-mode 1)

; — Frames And Windows — ; ; ———————— ;

(setq truncate-partial-width-windows nil) (setq split-window-preferred-function (quote split-window-sensibly))

; — Other — ; ; ————— ;

(setq require-final-newline nil) []common-lisp (defun toggle-theme () "Toggle between dark and light themes." (interactive) (if (custom-theme-enabled-p 'manoj-dark) (progn (disable-theme 'manoj-dark) (load-theme 'leuven t)) (progn (disable-theme 'leuven) (load-theme 'manoj-dark t))))

* (open-texdoc-in-background) []common-lisp (defun open-texdoc-in-background (docname) "Open a TEXDOC for DOCNAME in the background and close the terminal." (interactive "sEnter the name of the document: ") (let ((term-buffer (ansi-term "/bin/bash"))) (with-current-buffer term-buffer (term-send-raw-string (concat "texdoc " docname "")) (term-send-raw-string "sleep 2; exit") (set-process-sentinel (get-buffer-process term-buffer) (lambda (process signal) (when (or (string= signal "finished") (string= signal "exited")) (kill-buffer (process-buffer process))))) (bury-buffer))))

  https://chatgpt.com/c/7fac1ba4-f0a9-4d8f-8393-736e64e0ced1 acronym

- Make header regions read-only via tag []common-lisp (defun org-mark-readonly () (interactive) (let ((buf-mod (buffer-modified-p))) (org-map-entries (lambda () (org-mark-subtree) (add-text-properties (region-beginning) (region-end) '(read-only t))) "read$_{o}nly$")($unlessbuf - mod(set - buffer - modified - pnil)$))))

(defun org-remove-readonly () (interactive) (let ((buf-mod (buffer-modified-p))) (org-map-entries (lambda () (let* ((inhibit-read-only t)) (org-mark-subtree) (remove-text-properties (region-beginning) (region-end) '(read-only t)))) "read$_{o}nly$")($unlessbuf - mod(set - buffer - modified - pnil)$))))

(add-hook 'org-mode-hook 'org-mark-readonly)

- Protect text regions as read-only https://chatgpt.com/c/fe962d8c-eb34-42fe-b362-032a61d8b728 []common-lisp

  (defun make-region-read-only (start end) (interactive "*r") (let ((inhibit-read-only t)) (put-text-property start end 'read-only t) (put-text-property start end 'font-lock-face '(:background "8B0000"))))

  (defun make-region-read-write (start end) (interactive "*r") (let ((inhibit-read-only t)) (put-text-property start end 'read-only nil) (remove-text-properties start end '(font-lock-face nil))))

- Shell []common-lisp (defun dont-ask-to-kill-shell-buffer () "Don't ask for confirmation when killing *shell* buffer." (let ((buffer-name (buffer-name))) (when (string-equal buffer-name "*shell*") (setq kill-buffer-query-functions (delq 'process-kill-buffer-query-function kill-buffer-query-functions)))))

  (add-hook 'shell-mode-hook 'custom-dont-ask-to-kill-shell-buffer)

- cua-mode []common-lisp (cua-mode t)

- remove-blank-lines []common-lisp (defun remove-blank-lines () "Remove all blank lines (including lines with only whitespace) in the current buffer." (interactive) (save-excursion (goto-char (point-min)) (flush-lines "$^{[}$[: $space$:]]*")))

- Org-mode

  * Tags []common-lisp (setq org-tags-exclude-from-inheritance (list "alert" "biotool" "biopipe" "bimonthly" "block" "blk" "flat" "hierarchy" "include" "semimonthly" "purpose" "midGoal" "nearGoal" "focus" "project" "daily" "dinner" "kit" "maint" "manuscript" "mod" "monthly" "poster" "present" "prog" "report" "routine" "soln" "weekly" "write" "sci$_{r}ep$""$stretch$""$study$"))

  * .TODO []common-lisp (setq org-todo-keyword-faces (quote (("TODO" :background "red") ("NEXT" :foreground "black" :background "yellow"))))

    ;; keep TODO state timestamps in drawer (setq org-log-into-drawer t)

    ;; add done timestamp (setq org-log-done 'time)

    ;; enforce dependencies (setq org-enforce-todo-dependencies t)

    ;; priority levels (setq org-highest-priority 65) (setq org-lowest-priority 89) (setq org-default-priority 89)

* open in same frame []common-lisp (setq org-link-frame-setup '((vm . vm-visit-folder) (vm-imap . vm-visit-imap-folder) (gnus . org-gnus-no-new-news) (file . find-file) ;; Open files in the same frame (wl . wl)))
* Lists []common-lisp (setq org-cycle-include-plain-lists 'integrate) (setq org-list-indent-offset 0)
* Tables []common-lisp ;; https://emacs.stackexchange.com/questions/22210/auto-update-org-tables-before-each-export (add-hook 'before-save-hook 'org-table-recalculate-buffer-tables) []common-lisp (setq org-startup-align-all-tables t) (setq org-startup-shrink-all-tables t)
* Startup []common-lisp (setq org-startup-shrink-all-tables t) (setq org-startup-with-inline-images t)
* org-collector []common-lisp (require 'org-collector)
* Searching customizations
  - Create links to org tags http://endlessparentheses.com/use-org-mode-links-for-absolutely-anything.html Example: tag:rdata []common-lisp (defun endless/follow-tag-link (tag) "Display a list of TODO headlines with tag TAG. With prefix argument, also display headlines without a TODO keyword." (org-tags-view current-prefix-arg tag))
    (org-add-link-type "tag" 'endless/follow-tag-link)
* Set org-file-apps to use xdg-open for all file extensions

  []common-lisp ;(setq process-connection-type nil) this breaks *shell* (setq org-file-apps '((directory . "/usr/bin/gnome-terminal –working-directory=¨("
  .pdf
  '" . "setsid -w xdg-open ¨("
  .svg
  '" . "setsid -w xdg-open ¨("
  .yaml
  '" . "emacsclient -c ¨("
  .list
  '" . emacsclient) (auto-mode . emacsclient) (t . "setsid -w xdg-open ¨))
* org-image-actual-width When set as a list as below, 300 pixels will be the default, but another width can be specified through ATTR, e.g. #+ATTR_ORG: :width 800px []common-lisp (setq org-image-actual-width '(300))
* shk-fix-inline-images, reload inline images after code eval []common-lisp (defun shk-fix-inline-images () (when org-inline-image-overlays (org-redisplay-inline-images)))

  (with-eval-after-load 'org (add-hook 'org-babel-after-execute-hook 'shk-fix-inline-images))
* Source code
  - Declare Babel languages

    []common-lisp (org-babel-do-load-languages 'org-babel-load-languages '((ditaa . t) (dot .t) (emacs-lisp . t) (latex . t) (org . t) (python . t) (R . t) (shell . t) (sql .t) (sqlite . t)))

    https://claude.ai/chat/1017ebf5-da7e-40fb-b05d-2247281826b9

    []common-lisp (require 'ob-shell) (require 'yaml-mode)

    (defun org-babel-execute:yaml (body params) "Execute a block of YAML code with org-babel." (let ((temp-file (org-babel-temp-file "yaml-"))) (with-temp-file temp-file (insert body)) (org-babel-eval (format "cat

    (add-to-list 'org-src-lang-modes '("yaml" . yaml))
  - Default header arguments []bash (defun org-remove-properties-drawer () "Remove PROPERTIES drawer from tangled files." (save-excursion (goto-char (point-min)) (while (re-search-forward ":PROPERTIES:(?:.*)*?:END:"nilt)(

(add-hook 'org-babel-post-tangle-hook 'org-remove-properties-drawer)

(setq org-babel-default-header-args '((:results . "silent") (:eval . "no-export") (:exports . "none") (:tangle . "yes") (:cache . "yes") (:noweb . "yes") (:post-tangle . org-remove-properties-drawer)))
  - General []common-lisp

    (setq ;; Blocks inserted directly without additional formatting org-babel-inline-result-wrap ";; ;; Preserve language-specific indentation, aligns left org-src-preserve-indentation t ;; ;; Tab works like in major mode of lanuauge org-src-tab-acts-natively t ;; org-babel-python-command "python3" ;; org-confirm-babel-evaluate

nil ;; org-src-fontify-natively t) []common-lisp ;; disable confrmation for elisp execution of org src blocks (setq safe-local-variable-values '((org-confirm-elisp-link-function . nil)))

(setq org-hide-block-startup t)

- Toggle collapse blocks []common-lisp (defvar org-blocks-hidden nil)

  (defun org-toggle-blocks () (interactive) (if org-blocks-hidden (org-show-block-all) (org-hide-block-all)) (setq-local org-blocks-hidden (not org-blocks-hidden)))

- org-babel-min-lines-for-block-output When executing a source block in org mode with the output set to verbatim, it will sometimes wrap the results in an #begin_example block, and sometimes it uses : symbols at the beginning of the line. Prevented with org-babel-src-preserve-indentation

  https://emacs.stackexchange.com/questions/39390/force-org-to-use-instead-of-begin-example

  []common-lisp (setq org-babel-min-lines-for-block-output 1000)

- Change noweb wrapper symbols []common-lisp (setq org-babel-noweb-wrap-start "<" org-babel-noweb-wrap-end ">")

- (org-remove-properties-drawer) []common-lisp (defun org-remove-properties-drawer () "Remove PROPERTIES drawer from tangled files." (save-excursion (goto-char (point-min)) (while (re-search-forward ":PROPERTIES:(?:.*)*?:END:"nilt)(replace−match""))))

  (add-hook 'org-babel-post-tangle-hook 'org-remove-properties-drawer)

* Header views and cycling []common-lisp (setq org-show-hierarchy-above t)

  (setq org-fold-show-context-detail '((default . tree)))

* General []common-lisp

  (with-eval-after-load 'org (add-to-list 'org-modules 'org-habit))

  ;; Clock times in hours and minutes ;; (see https://stackoverflow.com/questions/22720526/set-clock-table-duration-format-for-emacs-org-mode (setq org-time-clocksum-format '(:hours "(setq org-duration-format (quote h:mm))

  (setq org-catch-invisible-edits 'error) (global-set-key (kbd "C-c l") 'org-store-link) (setq org-refile-targets '((org-agenda-files :maxlevel . 14))) (setq org-indirect-buffer-display 'current-window)

  (setq org-id-link-to-org-use-id 'use-existing) ;;https://stackoverflow.com/questions/28351465/emacs-orgmode-do-not-insert-line-between-headers (setf org-blank-before-new-entry '((heading . nil) (plain-list-item . nil)))

  (setq org-enforce-todo-checkbox-dependencies t) ;; don't adapt indentation to header level (setq org-adapt-indentation nil)

  (setq org-support-shift-select t) (setq org-src-window-setup 'current-window) (setq org-export-async-debug nil)

  (defun my-collapse-all-drawers (optional arg) (interactive "P") ;; "P" means that the function accepts a prefix argument and passes it as ARG (org-hide-drawer-all) (when arg ;; When ARG is non-nil (when called with C-u), execute 'org-cycle-global'. (org-cycle-global) (beginning-of-buffer)))

  (global-set-key (kbd "C-c d") 'my-collapse-all-drawers) ;; You might want to remove the hook if you don't want this function to run every time you open an org file (add-hook 'org-mode-hook 'my-collapse-all-drawers)

  []common-lisp ;; ensures that any file with the .org extension will automatically open in org-mode (add-to-list 'auto-mode-alist '("
  .org
  '" . org-mode))

  []common-lisp ;; Make heading regex include tags (setq org-heading-regexp "[[: *space* :]] *
  (
  +
)
  (? : +
  (.*?
)
)?[] * (: [[: *alnum* :]]@

  - org-blank-before-new-entry https://stackoverflow.com/questions/28351465/emacs-orgmode-do-not-in []common-lisp (setf org-blank-before-new-entry '((heading . never) (plain-list-item . never)))

- my-org-tree-to-indirect-buffer []common-lisp (defun my-org-tree-to-indirect-buffer (optional arg) "Open current org tree in indirect buffer, using one prefix argument. When called with two prefix arguments, ARG, run the original function without prefix argument." (interactive "P") (if (equal arg '(16)) ; 'C-u C-u' produces (16) (org-tree-to-indirect-buffer nil) ; original behavior (org-tree-to-indirect-buffer t)) ; one prefix argument (my-collapse-all-drawers)) (define-key org-mode-map (kbd "C-c C-x b") 'my-org-tree-to-indirect-buffer)

* Export []common-lisp ;; the below as nil fucks of export of inline code (setq org-export-babel-evaluate t) ;; https://emacs.stackexchange.com/questions/23982/cleanup-org-mode-export-intermediary-file/2400024000

  (setq-default cache-long-scans nil) (setq org-export-with-broken-links t) (setq org-export-allow-bind-keywords t)

  (setq org-export-with-sub-superscripts nil org-export-headline-levels 2 org-export-with-toc nil org-export-with-section-numbers nil org-export-with-tags nil org-export-with-todo-keywords nil)

  - LaTeX []common-lisp (require 'ox-latex)

    (customize-set-value 'org-latex-with-hyperref nil)

    (setq org-latex-logfiles-extensions (quote ("auto" "lof" "lot" "tex " "aux" "idx" "log" "out" "toc" "nav" "snm" "vrb" "dvi" "fdb$_latexmk$""blg""brf""fls""$entoc$""ps""spl""bbl")))

(add-to-list 'org-latex-packages-alist '("" "listings")) (add-to-list 'org-latex-packages-alist '("" "color")) (setq org-latex-caption-above nil)

(setq org-latex-remove-logfiles t)

(add-to-list 'org-latex-packages-alist '("" "listingsutf8")) (setq org-latex-src-block-backend 'minted)

(setq org-latex-pdf-process '("pdflatex -shell-escape -interaction nonstopmode -output-directory "bibtex "pdflatex -shell-escape -interaction nonstopmode -output-directory "pdflatex -shell-escape -interaction nonstopmode -output-directory

  - Empty latex class []common-lisp (with-eval-after-load 'ox-latex (add-to-list 'org-latex-classes '("empty" "
    documentclassarticle
    newcommand
    foobar [NO-DEFAULT-PACKAGES] [NO-PACKAGES]" ("
    section("
    subsection("
    subsubsection("
    paragraph("
    subparagraph

* Prevent blank lines inserted between headers []common-lisp (setf org-blank-before-new-entry '((heading . nil) (plain-list-item . nil)))

* iCalendar []common-lisp (setq org-icalendar-with-timestamps 'active) (setq org-icalendar-use-scheduled t) (setq org-icalendar-use-deadline nil) (setq org-icalendar-include-todo t) (setq org-icalendar-exclude-tags (list "noexport")) (setq org-icalendar-include-body '1) (setq org-icalendar-alarm-time '5) (setq org-icalendar-store-UID t) ;;Required for syncs (setq org-icalendar-timezone "America/Chicago") (setq org-agenda-default-appointment-duration 30) (setq org-icalendar-combined-agenda-file "/tmp/org.ics")

* Properties []common-lisp (setq org-use-property-inheritance t)

* (browse-org-table-urls-by-name)

  []common-lisp (defun browse-org-table-urls-by-name (table-name) "Browse URLs listed in an Org-mode table identified by TABLE-NAME.

  table-name Table name identified as +name:

  Example usage: (browse-org-table-urls-by-name the-table-name)"

  (interactive "sEnter table name: ") (let* ((element (org-element-map (org-element-parse-buffer) 'table (lambda (el) (when (string= (org-element-property :name el) table-name) el)) nil t))) (if (not element) (message "Table with name (let ((table-content (buffer-substring-no-properties (org-element-property :contents-begin element) (org-element-property :contents-end element)))) (with-temp-buffer (insert table-content) (goto-char (point-min)) (let ((urls (org-table-to-lisp))) (if (not urls) (message "No URLs found in the table with name (let ((first-url (car (car urls)))) (start-process "brave-browser" nil "brave-browser" "–new-window" first-url) (sit-for 2) ; Wait for the new window to open (dolist (url-row (cdr urls)) (start-process "brave-browser" nil "brave-browser" (car url-row)) (sit-for 0.5))) ; Add a delay of 0.5 seconds between each URL (message "Opened URLs from table with name

* Agenda []common-lisp (setq org-agenda-repeating-timestamp-show-all nil) (setq org-sort-agenda-notime-is-late nil) (setq org-agenda-start-on-weekday nil) (setq org-agenda-remove-tags t) (setq org-agenda-skip-scheduled-if-done t) (setq org-agenda-files (list " /repos/org/"))

  (define-key global-map "-ca" 'org-agenda) (setq org-agenda-skip-unavailable-files t)

  (setq org-agenda-use-tag-inheritance t) ;; http://stackoverflow.com/questions/36873727/make-org-agenda-full-screen (setq org-agenda-window-setup (quote only-window)) (setq org-agenda-todo-ignore-time-comparison-use-seconds t)

  ;;; Based on http://article.gmane.org/gmane.emacs.orgmode/41427 (defun my-skip-tag(tag) "Skip entries that are tagged TAG" (let* ((entry-tags (org-get-tags-at (point)))) (if (member tag entry-tags) (progn (outline-next-heading) (point)) nil)))

  []common-lisp ;; Needed for no y/n prompt at linked agenda execution (setq org-confirm-elisp-link-function nil)

- custom-command-error-function []common-lisp ;; https://emacs.stackexchange.com/questions/19742/is-there-a-way-to-disable-the-buffer-is-read-only-warning (defun custom-command-error-function (data context caller) "Ignore the buffer-read-only signal; pass the rest to the default handler." (when (not (eq (car data) 'buffer-read-only)) (command-error-default-function data context caller)))

  (setq command-error-function 'my-command-error-function)

- bibtex []common-lisp (setq reftex-default-bibliography '(" /repos/org/bib.bib"))

  ;; see org-ref for use of these variables

  (setq bibtex-completion-bibliography " /repos/org/bib.bib" bibtex-completion-library-path " /library" bibtex-completion-notes-path " /repo/org/notes")

  * get-bibtex-from-doi []common-lisp ;; Amazing bibtex from doi fetcher ;; https://www.anghyflawn.net/blog/2014/emacs-give-a-doi-get-a-bibtex-entry/ (defun get-bibtex-from-doi (doi) "Get a BibTeX entry from the DOI" (interactive "MDOI: ") (let ((url-mime-accept-string "text/bibliography;style=bibtex")) (with-current-buffer (url-retrieve-synchronously (format "http://dx.doi.org/(replace-regexp-in-string "http://dx.doi.org/" "" doi))) (switch-to-buffer (current-buffer)) (goto-char (point-max)) (setq bibtex-entry (buffer-substring (string-match "@" (buffer-string)) (point))) (kill-buffer (current-buffer)))) (insert (decode-coding-string bibtex-entry 'utf-8)) (bibtex-fill-entry))

    · dev

    · On *[2024-06-27 Thu]* I played around a little with also fetching abstract. The crossref metadata doesn't have that usually, although they built the functionality into their api. Instead, I can get abstract from pubmed like this:

      []common-lisp (require 'url) (require 'xml)

      (defun xml-node-to-string (node) "Convert an XML node to a string, handling nested elements." (cond ((stringp node) node) ((listp node) (let ((tag (car node)) (attrs (cadr node)) (content (cddr node))) (concat (mapconcat 'xml-node-to-string content "") (when (eq tag 'sup) " "))))))

      (defun get-pubmed-abstract (pmid) "Get abstract from PubMed using the given PubMed ID" (interactive "sPubMed ID: ") (let* ((url (format "https://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=pubmedid=(buffer (url-retrieve-synchronously url)) xml-data abstract-text) (with-current-buffer buffer (goto-char (point-min)) (search-forward "") (setq xml-data (xml-parse-region (point) (point-max))) (with-output-to-temp-buffer "*PubMed XML*" (print xml-data)) (let* ((pubmed-article (car (xml-get-children (car xml-data) 'PubmedArticle))) (medline-citation (car (xml-get-children pubmed-article 'MedlineCitation))) (article (car (xml-get-children medline-citation 'Article))) (abstract (car (xml-get-children article 'Abstract))) (abstract-texts (xml-get-children abstract 'AbstractText))) (setq abstract-text (mapconcat (lambda (a) (xml-node-to-string a)) abstract-texts " ")))) (if (not (string-empty-p abstract-text)) (progn (setq abstract-text (replace-regexp-in-string "
s-+" " " abstract-text)) (setq abstract-text (string-trim abstract-text)) abstract-text) (message "No abstract found for PubMed ID nil)))

      (defun insert-pubmed-abstract () "Insert a PubMed abstract" (interactive) (let* ((pmid (read-string "PubMed ID: ")) (abstract (get-pubmed-abstract pmid))) (when abstract (insert abstract))))

      In solid tumor oncology, circulating tumor DNA (ctDNA) is poised to transform care through accurate assessment of minimal residual disease (MRD) and therapeutic response monitoring. To overcome

the sparsity of ctDNA fragments in low tumor fraction (TF) settings and increase MRD sensitivity, we previously leveraged genome-wide mutational integration through plasma whole-genome sequencing (WGS). Here we now introduce MRD-EDGE, a machine-learning-guided WGS ctDNA single-nucleotide variant (SNV) and copy-number variant (CNV) detection platform designed to increase signal enrichment. MRD-EDGESNV uses deep learning and a ctDNA-specific feature space to increase SNV signal-to-noise enrichment in WGS by ~300× compared to previous WGS error suppression. MRD-EDGECNV also reduces the degree of aneuploidy needed for ultrasensitive CNV detection through WGS from 1 Gb to 200 Mb, vastly expanding its applicability within solid tumors. We harness the improved performance to identify MRD following surgery in multiple cancer types, track changes in TF in response to neoadjuvant immunotherapy in lung cancer and demonstrate ctDNA shedding in precancerous colorectal adenomas. Finally, the radical signal-to-noise enrichment in MRD-EDGESNV enables plasma-only (non-tumor-informed) disease monitoring in advanced melanoma and lung cancer, yielding clinically informative TF monitoring for patients on immune-checkpoint inhibition.

but i'm not able to get pubmed ID

and the ncbi eutils can get pmid from doi:

[]bash ;curl "https://api.ncbi.nlm.nih.gov/lit/ctxp/v1/pubmed/?format=pubmedid=10.1038/nature12373"

;curl -LH "Accept: application/json" "https://api.crossref.org/works/10.1038/nature12373"

;curl "https://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=pubmedid=23892778retmode=xml"

;curl "https://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=pubmedterm=10.1038/nature12373"

;curl "https://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=pubmedterm=10.1038/nature12373[doi]"

;DOI="10.1038/s41591-024-03040-4" ;ENCODED$_DOI$ =(echo "DOI"|$jq-sRr@uri$); $ESearch_URL$ = "$https://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=pubmedterm$ =ENCODED$_DOI$, ; $pmid$ = 38877116, ;

; Fetch the PubMed ID ;PMID=($curl-s$"ESearch$_URL$"|$grep-oP'(?<=<Id>)[^<]+'$); $echo$"$PubMedID$:PMID"

but i haven't build this into the retreival yet.

see also https://claude.ai/chat/cce7d404-eb1b-4424-8272-f46660f53612 https://github.com/jkitchin/org-ref/blob/master/doi-utils.el

- python.el https://github.com/gregsexton/ob-ipython/issues/28 []common-lisp

  (setq python-shell-completion-native-enable nil)

  (add-hook 'python-mode-hook (lambda () (setq indent-tabs-mode nil)))

  (setq python-indent-guess-indent-offset-verbose nil)

- Alpha key []common-lisp (global-set-key (kbd "C-x a") (lambda () (interactive) (insert "")))

- dev []common-lisp (set-language-environment "UTF-8") (set-default-coding-systems 'utf-8)

  * :plain link type https://chatgpt.com/c/6dfdb7af-d81c-4096-a89f-5a4b0455fe0f https://claude.ai/chat/c775f0eb-fa91-45b4-82d6-e1a0df8b5526 []common-lisp (defun org-plain-follow (id) "$Followaplainlinka$ $id-openidnil$))

(org-link-set-parameters "plain" :follow 'org-plain-follow :export 'org-plain-export :store 'org-store-link)

(defun org-plain-export (link description format) "$Exportaplainlink.Alwaysexportasplaintext.$" ($cond$(($eqformat'html$)($ordescriptionl

(provide 'ol-plain)

(with-eval-after-load 'org (require 'ol-plain))

- mark-whole-word []common-lisp ;; https://emacs.stackexchange.com/questions/35069/best-way-to-select-a-word (defun mark-whole-word (optional arg allow-extend) "Like 'mark-word', but selects whole words and skips over whitespace. If you use a negative prefix arg then select words backward. Otherwise select them forward.

  If cursor starts in the middle of word then select that whole word.

  If there is whitespace between the initial cursor position and the first word (in the selection direction), it is skipped (not selected).

If the command is repeated or the mark is active, select the next NUM words, where NUM is the numeric prefix argument. (Negative NUM selects backward.)" (interactive "P") (let ((num (prefix-numeric-value arg))) (unless (eq last-command this-command) (if (natnump num) (skip-syntax-forward "
s-") (skip-syntax-backward "
s-"))) (unless (or (eq last-command this-command) (if (natnump num) (looking-at "
b") (looking-back "
b"))) (if (natnump num) (left-word) (right-word))) (mark-word arg allow-extend)))

(global-set-key (kbd "C-c C-SPC") 'mark-whole-word)

- AUCTeX
  - AUCTeX reference header
  - Use-package []common-lisp (use-package tex :ensure auctex :config (setenv "PATH" (concat "/usr/local/texlive/2021/bin/x86$_6$4−*linux* : "(*getenv*"*PATH*"))) (*add* − *to* − *list*′*exec* − *path*"/*usr*/*local*/*texlive*/2021/*bin*/*x*86$_6$4 − *linux*") (*setqTeX* − *auto* − *savetTeX* − *save* − *querynilTeX* − *view* − *program* − *selection*′(((*output* − *dvihas* − *no* − *display* − *manager*)."*dvi2tty*") ((*output*−*dvistyle*−*pstricks*)."*dvipsandgv*") (*output*−*pdf*."*Okular*") (*output*−*dvi*."*xdvi*") (*output*−*pdf*."*Evince*") (*output* − *html*."*xdg* − *open*"))))
- Blacken
  - Use-package []common-lisp (use-package blacken :after elpy :hook (elpy-mode . blacken-mode))
- Company-mode
  - Company mode reference header
  - Use-package []common-lisp (use-package company :config (global-company-mode) (setq company-dabbrev-downcase nil)) ; Don't downcase by default
- Eglot
  - Use-package []common-lisp

    (use-package eglot :ensure t :init (add-hook 'sh-mode-hook 'eglot-ensure) (add-hook 'ess-r-mode-hook 'eglot-ensure) (add-hook 'python-mode-hook 'eglot-ensure) :config (add-to-list 'eglot-server-programs '(sh-mode . ("bash-language-server" "start"))) (add-to-list 'eglot-server-programs '(python-mode . ("pylsp"))) (add-to-list 'eglot-server-programs '(ess-r-mode . ("R" "–slave" "-e" "languageserver::run()"))))

    (with-eval-after-load 'eglot (define-key eglot-mode-map (kbd "C-c <tab>") 'company-complete))
- Essh []common-lisp (require 'essh) (defun essh-sh-hook () (define-key sh-mode-map "-c-r" 'pipe-region-to-shell) (define-key sh-mode-map "-c-b" 'pipe-buffer-to-shell) (define-key sh-mode-map "-c-j" 'pipe-line-to-shell) (define-key sh-mode-map "-c-n" 'pipe-line-to-shell-and-step) (define-key sh-mode-map "-c-f" 'pipe-function-to-shell) (define-key sh-mode-map "-c-d" 'shell-cd-current-directory)) (add-hook 'sh-mode-hook 'essh-sh-hook)

  (add-hook 'sh-mode-hook 'flycheck-mode)
- ESS
  - ESS reference header
  - Use-package []common-lisp (use-package ess :init (require 'ess-site) :config (setq ess-ask-for-ess-directory nil ess-help-own-frame 'one ess-indent-with-fancy-comments nil ess-use-auto-complete t ess-use-company t inferior-ess-own-frame t inferior-ess-same-window nil) (define-key ess-mode-map (kbd "C-c C-n") 'ess-eval-line-and-step) :mode (("/R/.*
    .q
    '" . R-mode) ("
    .[rR]
    '" . R-mode) ("
    .[rR]profile
    '" . R-mode) ("NAMESPACE
    '" . R-mode) ("CITATION

’" . R-mode) ("
.[Rr]out" . R-transcript-mode) ("
.Rd
’" . Rd-mode)))

  - Syntax highlighting []common-lisp (custom-set-variables ’(ess-R-font-lock-keywords (quote ((ess-R-fl-keyword:modifiers . t) (ess-R-fl-keyword:fun-defs . t) (ess-R-fl-keyword:keywords . t) (ess-R-fl-keyword:assign-ops . t) (ess-R-fl-keyword:constants . t) (ess-fl-keyword:fun-calls . t) (ess-fl-keyword:numbers . t) (ess-fl-keyword:operators . t) (ess-fl-keyword:delimiters . t) (ess-fl-keyword:= . t) (ess-R-fl-keyword:FT . t) (ess-R-fl-keyword:

- Elpy
  - Elpy reference header
  - Use-pacakge []common-lisp (use-package elpy :init (advice-add ’python-mode :before ’elpy-enable) :config (define-key elpy-mode-map (kbd "C-c C-n") ’elpy-shell-send-statement-and-step) (setenv "PATH" (concat " /miniconda3/bin:" (getenv "PATH"))) (setenv "WORKON$_{H}OME$"" $/miniconda3/envs$")($setqexec-path(append'$(" $/miniconda3/bin$")$exec-path$))($add-to-list'process-coding-system-alist'$("$python$".($utf-8.utf-8$)))($setqelpy-rpc-python-command$" $/miniconda3/bin/python$"))
- exec-path-from-shell
  - Ensures parts of Emacs inherit shell PATH when Emacs is runnng as a daemon
  - Use-package []common-lisp (use-package exec-path-from-shell :config (when (daemonp) (exec-path-from-shell-initialize)))
- flycheck
  - Use-package []common-lisp (use-package flycheck :hook (org-src-mode . my-org-mode-flycheck-hook) :config (defun my-org-mode-flycheck-hook () (when (derived-mode-p ’prog-mode) ;; Check if it’s a programming mode (flycheck-mode 1))))
- flyspell
  - ispell reference header
  - Use-package []common-lisp (use-package flyspell :config (setq ispell-personal-dictionary " /.aspell.en.pws"))
- Helm
  - Helm reference header
  - Use-package []common-lisp (use-package helm :config (global-set-key (kbd "C-x b") ’helm-mini) (global-set-key (kbd "C-s") ’helm-occur) (setq helm-completion-style ’emacs helm-move-to-line-cycle-in-source nil)) ;; allow C-n through different sections
- helm-org
  - Use-package []common-lisp (use-package helm-org :config (global-set-key (kbd "C-c j") ’helm-org-in-buffer-headings) (global-set-key (kbd "C-c w") ’helm-org-refile-locations) (setq org-outline-path-complete-in-steps nil org-refile-allow-creating-parent-nodes ’confirm org-refile-targets ’((org-agenda-files :maxlevel . 20)) org-refile-targets ’((org-agenda-files :maxlevel . 3)) org-refile-use-outline-path ’file)) (define-key global-map (kbd "C-c C-j") nil) (global-set-key (kbd "C-c C-j") ’helm-org-agenda-files-headings) (define-key global-map (kbd "C-")$'org-mark-ring-goto$)($global-set-key(kbd$"$C-cC-j$")$'helm-org-agenda-files-headings$)($setqhelm-org-ignore-autosavest$)

    []common-lisp (global-set-key (kbd "C-c C-j") ’helm-org-agenda-files-headings)

    (with-eval-after-load ’org (define-key org-mode-map (kbd "C-c C-j") ’helm-org-agenda-files-headings))
- helm-org-rifle

  - Use-package []common-lisp (use-package helm-org-rifle :config (setq helm-org-rifle-show-path nil helm-org-rifle-show-full-contents nil) (require 'helm) (global-set-key (kbd "C-c C-j") 'helm-org-agenda-files-headings))

- htmlize
  - Use-pacakge []common-lisp (use-package htmlize)

- ivy
  - Use-package []common-lisp (use-package ivy :diminish)

- marginalia
  - Use-package []common-lisp (use-package marginalia ;; Either bind 'marginalia-cycle' globally or only in the minibuffer :bind (("M-A" . marginalia-cycle) :map minibuffer-local-map ("M-A" . marginalia-cycle)) ;; The :init configuration is always executed (Not lazy!) :init (marginalia-mode))

- Native complete
  - Use-package []common-lisp (use-package native-complete)

- ob-async
  - Use-package []common-lisp (use-package ob-async)

- orderless
  - Use-package []common-lisp (use-package orderless :init ;; Configure a custom style dispatcher (see the Consult wiki) ;; (setq orderless-style-dispatchers '(+orderless-dispatch) ;; orderless-component-separator 'orderless-escapable-split-on-space) (setq completion-styles '(orderless basic) completion-category-defaults nil completion-category-overrides '((file (styles partial-completion)))))

- org-plus-contrib
  - Use-package []common-lisp (require 'org-checklist) (require 'ox-extra) (ox-extras-activate '(ignore-headlines))

- org-ref
  - Package notes
  - Use-package []common-lisp (use-package org-ref :init (require 'bibtex) (require 'org-ref-ivy) (require 'org-ref-bibtex) (require 'org-ref-pubmed) (require 'org-ref-scopus) (require 'org-ref-wos) :config (setq org-ref-default-bibliography '(" /repos/org/bib.bib") org-ref-pdf-directory " /library/")) []common-lisp (setq bibtex-completion-bibliography '(" /repos/org/bib.bib") bibtex-completion-library-path '(" /data/library/") bibtex-completion-additional-search-fields '(keywords) bibtex-completion-display-formats '((article . "=has-pdf=:1=has-note=:1 year :4author:36 title :∗journal:40") (inbook . "=has-pdf=:1=has-note=:1 year :4author:36 title :∗Chapterchapter:32") (incollection . "=has-pdf=:1=has-note=:1 year :4author:36 title :∗booktitle:40") (inproceedings . "=has-pdf=:1=has-note=:1 year :4author:36 title :∗booktitle:40") (t . "=has-pdf=:1=has-note=:1 year :4author:36 title :∗")) bibtex-completion-pdf-open-function (lambda (fpath) (call-process "open" nil 0 nil fpath)))

(define-key org-mode-map (kbd "C-c]") 'org-ref-insert-link)

(setq org-ref-show-broken-links nil)

(setq org-ref-bibliography-notes " /repo/org/notes" org-ref-default-bibliography '(" /repos/org/bib.bib") org-ref-pdf-directory " /library")

(require 'org-ref-ivy) (setq org-ref-insert-link-function 'org-ref-insert-link-hydra/body org-ref-insert-cite-function 'org-ref-cite-insert-ivy org-ref-insert-label-function 'org-ref-insert-label-link org-ref-insert-ref-function 'org-ref-insert-ref-link org-ref-cite-onclick-function (lambda () (org-ref-citation-hydra/body)))

- ox-pandoc

- Use-package []common-lisp

  (use-package ox-pandoc :after org :config (setq org-pandoc-options-for-docx '((standalone . nil))))

- savehist

  - Use-package []common-lisp (use-package savehist)

- snakemake-mode []common-lisp (use-package snakemake-mode) []common-lisp (defcustom snakemake-indent-field-offset nil "Offset for field indentation." :type 'integer)

  (defcustom snakemake-indent-value-offset nil "Offset for field values that the line below the field key." :type 'integer)

- Tree-sitter []common-lisp

  ;; Install and configure tree-sitter (use-package tree-sitter :ensure t)

  ;; Install and configure tree-sitter-langs (use-package tree-sitter-langs :ensure t :after tree-sitter :config (add-hook 'tree-sitter-after-on-hook 'tree-sitter-hl-mode))

  (global-tree-sitter-mode) (add-hook 'tree-sitter-after-on-hook 'tree-sitter-hl-mode)

  (defun disable-tree-sitter-for-org-mode () (when (eq major-mode 'org-mode) (tree-sitter-mode -1)))

  (add-hook 'tree-sitter-mode-hook 'disable-tree-sitter-for-org-mode)

- vertico

  - Use-package []common-lisp (use-package vertico)

  - Use 'consult-completion-in-region' if Vertico is enabled. []common-lisp

    ;; Otherwise use the default 'completion–in-region' function. (setq completion-in-region-function (lambda (rest args) (apply (if vertico-mode 'consult-completion-in-region 'completion–in-region) args)))

  - other []common-lisp ;; A few more useful configurations... (use-package emacs :init ;; Add prompt indicator to 'completing-read-multiple'. ;; We display [CRM<separator>], e.g., [CRM,] if the separator is a comma. (defun crm-indicator (args) (cons (format "[CRM(replace-regexp-in-string "
    '
*?

    |
*?

    '" "" crm-separator) (car args)) (cdr args))) (advice-add 'completing-read-multiple :filter-args 'crm-indicator)

    ;; Do not allow the cursor in the minibuffer prompt (setq minibuffer-prompt-properties '(read-only t cursor-intangible t face minibuffer-prompt)) (add-hook 'minibuffer-setup-hook 'cursor-intangible-mode)

    ;; Emacs 28: Hide commands in M-x which do not work in the current mode. ;; Vertico commands are hidden in normal buffers. ;; (setq read-extended-command-predicate ;; 'command-completion-default-include-p)

    ;; Enable recursive minibuffers (setq enable-recursive-minibuffers t))

    (define-key vertico-map (kbd "TAB") 'minibuffer-complete) (define-key vertico-map (kbd "C-n") 'vertico-next) (define-key vertico-map (kbd "C-p") 'vertico-previous)

    []common-lisp ;; Ensure you have these packages installed (use-package vertico :ensure t :init (vertico-mode))

    (use-package marginalia :ensure t :after vertico :init (marginalia-mode))

    (use-package orderless :ensure t :init ;; Customize completion styles to include orderless (setq completion-styles '(orderless basic)) ;; Optionally configure completion categories (setq completion-category-defaults nil) (setq completion-category-overrides '((file (styles basic partial-completion)))))

    (use-package savehist :ensure t :init (savehist-mode))

    (use-package consult :ensure t :bind (("C-x b" . consult-buffer) ("M-y" . consult-yank-pop) ("C-s" . consult-line) ("M-g g" . consult-goto-line) ("M-g M-g" . consult-goto-line) ("C-M-l" . consult-imenu) :map minibuffer-local-map ("M-r" . consult-history)) :init (setq register-preview-delay 0 register-preview-function 'consult-register-preview) ;;

Optionally configure preview (autoload 'consult-register-window "consult") (setq consult-register-window-function 'consult-register-window) ;; Optionally configure narrowing key (setq consult-narrow-key "<"))

;; Enable vertico-directory for better directory navigation (use-package vertico-directory :ensure nil :load-path "path/to/vertico-directory" :after vertico :bind (:map vertico-map ("RET" . vertico-directory-enter) ("DEL" . vertico-directory-delete-char) ("M-DEL" . vertico-directory-delete-word)))

;; Example configuration for more intuitive completion cycling (define-key vertico-map (kbd "TAB") 'minibuffer-complete) (define-key vertico-map (kbd "C-n") 'vertico-next) (define-key vertico-map (kbd "C-p") 'vertico-previous)

- vterm
  - Use-package []common-lisp

    (use-package vterm :bind* (:map vterm-mode-map ("C-z" . vterm-undo) ("C-v" . vterm-yank)) :init (add-hook 'vterm-mode-hook '(lambda () (setq-local cua-mode nil)))) :config (setq vterm-max-scrollback 100000) (custom-set-faces '(vterm-color-blue ((t (:foreground "477EFC" :background "477EFC")))))
  - Mult-vterm []common-lisp (use-package multi-vterm :ensure t)
- web-mode
  - Use-package []common-lisp (use-package web-mode :mode ("
    .phtml
    '" "
    .tpl
    .php
    '" "
    .[agj]sp
    '" "
    .as[cp]x
    '" "
    .erb
    '" "
    .mustache
    '" "
    .djhtml
    '" "
    .html?
    '"))
- YASnippets
  - Use-package []common-lisp (use-package yasnippet :init (setq yas-snippet-dirs '(" /.emacs.d/private$_{y}asnippets$"" $/.emacs.d/publ$ $config(yas-global-mode1); Enableeverywhere(define-keyyas-minor-mode-map(kbd" < C-tab >
    ")'yas-expand))

(when (file-exists-p " /repos/latex/latex$_{s}nippets$")($add-to-list'yas-snippet-dirs$" $/repos/latex/latex$_{s}nippets$"))

[]common-lisp (defun my-org-mode-hook () (setq-local yas-buffer-local-condition '(not (org-in-src-block-p t)))) (add-hook 'org-mode-hook 'my-org-mode-hook)

  - Prevent company mode during expansions []common-lisp (add-hook 'yas-before-expand-snippet-hook (lambda () (setq-local company-backends nil))) (add-hook 'yas-after-exit-snippet-hook (lambda () (kill-local-variable 'company-backends)))
  - Provide a setting to auto-expand snippets []common-lisp (setq require-final-newline nil) (defun yas-auto-expand () "Function to allow automatic expansion of snippets which contain a condition, auto."

    (when yas-minor-mode (let ((yas-buffer-local-condition '(require-snippet-condition . auto))) (yas-expand))))

    (defun my-yas-try-expanding-auto-snippets () (when yas-minor-mode (let ((yas-buffer-local-condition '(require-snippet-condition . auto))) (yas-expand))))

    (add-hook 'post-command-hook 'my-yas-try-expanding-auto-snippets)

- Load last []common-lisp

  (run-with-idle-timer 1 nil (lambda () (when (member "Hack" (font-family-list)) (set-face-attribute 'default nil :family "Hack" :height 114 :weight 'light) (message "Font set to Hack"))))

  (custom-set-faces '(default ((t (:family "Hack" :height 114 :weight light)))))

  (add-hook 'emacs-startup-hook (lambda () (load-theme 'manoj-dark t)))

- Ideas https://chatgpt.com/c/8e222105-a52a-4856-80d3-c37823f0efb1

Literate Programming with Emacs Org-mode

Generally comments should reside within the Org-mode structure and outside of code blocks. Tangling with a header argument :comments org will include both the header text and text between the header and code block. For example:
(property drawers are excluded from tangling with custom-org-remove-properties-drawer)

For example:

- This header will be a comment This text in org and below the header will be a comment

  []bash

  This text within the code block will be a comment

  ls

  This comment will not appear in the tangled code

  - Child headers are not comments unless they contain more code blocks

- Result:

  ls

YASnippets Style Guide and Good Practice

- – snippets are associated with a specific major mode, use of bash mode is discouraged
  - As snippets are associated with major modes, single word keywords are encouraged (e.g. function instead of bash.function)
  - snippets expand on tab
  - for modes with extensive snippet libraries, prefixes followed by a single period are preferred (e.g. mod.meeting)
  - prefixed snippets keywords are not likely to be confused with non-snippet terms, and they should expand without trigger
  - Snippet creation and storage
    - Snippets are created in org mode bash source code blocks with formatting as in the snippet
      - Each snippet resides under it's own terminal org mode header. The header consists of only the snippet keyword and tags including the :yas: tag (i.e. * <SNIPPET KEYWORD> :yas:)
      - The :yas: tag is reserved exclusively for yas block headers and does not have tag inheritance.
    - Snippets are stored in the main org repository under the directory./snippets which is symlinked to .emacs.d. Snippets are therefore under org version control.