Setting up neovim sor data science code development.

Github, neovim

Ronald (Ryy) Glenn Thomas

Table of contents

Photo by Nathan Waters on Unsplash

Introduction

Neovim (a fork of Vim) is a text editor with several advantages for data science code development. One of the main attractions is that its open source with a number of useful plugins to facilitate working on R, python, and julia code. Also, its modal system allows text manipulation at potentially far greater speed than conventional, mouse-centric, systems.

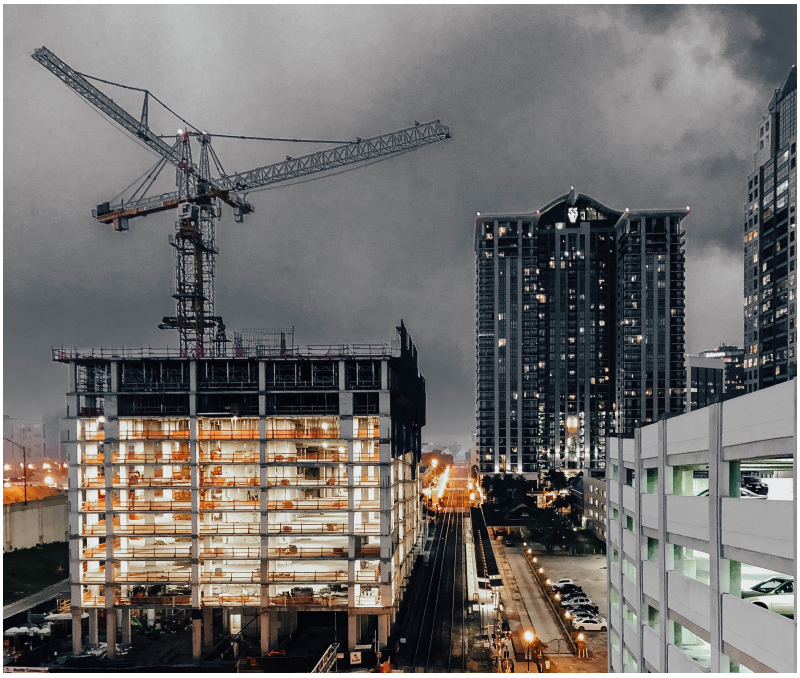

Figure 1: under construction

Our presentation here is for an macos environment. Appendix one contains required adjustments for a ubuntu linux environment.

Step one: Install the latest stable version of neovim.

The simplist approach is to use homebrew:

```
> brew install neovim neovim-qt
```

Set up convenience aliases in `zsh`.

```
> alias n = neovim-qt
> alias nt = neovim
```

(the `t` in `nt` is for `terminal`)

Step 2: Configure neovim

The standard location for `neovim` config files on "unix-like" systems is `~/.config/nvim`. The main config file is either init.vim (VimL) or init.lua (Lua). In this post we'll focus on lua based configuration.

Specifically, the following code block creates an `nvim` subdirectory under `~/.config` and initialize a configuration file `init.lua`.

```
> cd .config
> mkdir nvim
> cd nvim
> touch init.lua
```

install packer plugin manager

```
git clone --depth 1 https://github.com/wbthomason/packer.nvim\
~/.local/share/nvim/site/pack/packer/start/packer.nvim
```

Add following to init.lua to use `packer.nvim` to manage plugins (maximally minimal)

```
vim.g.mapleader = ","
vim.g.localleader = " "
require('vars')      -- Variables
require('opts')      -- Options
require('keys')      -- Keymaps
require('plug')      -- Plugins
vim.cmd([[
filetype plugin on
syntax enable
set clipboard+=unnamedplus
]])
vim.g.tex_flavor='latex'
vim.g.vimtex_view_method='skim'
vim.g.vimtex_quickfix_mode=0
vim.g.UltiSnipsSnippetDirectories = "['~/.vim/UltiSnips', 'UltiSnips']"
```

```
vim.g.UltiSnipsExpandTrigger="<tab>"
vim.g.UltiSnipsJumpForwardTrigger="<tab>"
vim.g.UltiSnipsJumpBackwardTrigger="<s-tab>"
vim.g.UltiSnipsEditSplit="vertical"
vim.g.UltiSnipsUsePythonVersion = 3

require('leap').add_default_mappings()
require'lspconfig'.r_language_server.setup{}
vim.opt.termguicolors = true
vim.cmd('colorscheme onedark')

vim.cmd([[
augroup rmarkdown
autocmd!
autocmd FileType rmd,r nnoremap <buffer> <CR>  :call SendLineToR("down")<CR>
autocmd FileType rmd,r nnoremap <buffer> <space>l :call b:SendChunkToR("silent","down")<cr>
autocmd FileType rmd,r nnoremap <buffer> <space>' :call RMakeRmd("default")<cr>
autocmd FileType rmd,r vmap <buffer> <CR> <localleader>sd
autocmd FileType rmd,r vmap <buffer> <s-CR> :call b:SendSelectionToR("silent","down")<cr>
autocmd FileType rmd,r noremap <space>r :call StartR("R")<cr>
autocmd FileType rmd,r noremap <space>s :call RAction("str")<cr>
autocmd FileType rmd,r noremap <space>i :call RAction("dim")<cr>
autocmd FileType rmd,r noremap <localleader>h :call RAction("head")<cr>
autocmd FileType rmd,r noremap <space>p :call RAction("print")<cr>
autocmd FileType rmd,r noremap <space>q :call RAction("length")<cr>
autocmd FileType rmd,r noremap <space>n :call RAction("nvim.names")<cr>
autocmd FileType rmd,r noremap <space>c :call RAction("cnt")<cr>
autocmd FileType rmd,r noremap <space>k :call b:PreviousRChunk()<cr>
autocmd FileType rmd,r noremap <space>j :call b:NextRChunk()<cr>
autocmd FileType rmd,r nnoremap <buffer> <space>t  :call b:RDSendLineAndInsertOutput("silent
augroup END
]])

-- COC install config
-- Some servers have issues with backup files, see #649.
vim.opt.backup = false
vim.opt.writebackup = false
-- Having longer updatetime (default is 4000 ms = 4 s) leads to noticeable
-- delays and poor user experience.
```

```
vim.opt.updatetime = 300
-- Always show the signcolumn, otherwise it would shift the text each time
-- diagnostics appear/become resolved.
vim.opt.signcolumn = "yes"

local keyset = vim.keymap.set
-- Auto complete
function _G.check_back_space()
    local col = vim.fn.col('.') - 1
    return col == 0 or vim.fn.getline('.'):sub(col, col):match('%s') ~= nil
end
local opts = {silent = true, noremap = true, expr = true, replace_keycodes = false}
keyset("i", "<TAB>", 'coc#pum#visible() ? coc#pum#next(1) : v:lua.check_back_space() ? "<TAB
keyset("i", "<S-TAB>", [[coc#pum#visible() ? coc#pum#prev(1) : "\<C-h>"]], opts)

-- Make <CR> to accept selected completion item or notify coc.nvim to format
-- <C-g>u breaks current undo, please make your own choice.
keyset("i", "<cr>", [[coc#pum#visible() ? coc#pum#confirm() : "\<C-g>u\<CR>\<c-r>=coc#on_ent

-- Use <c-j> to trigger snippets
keyset("i", "<c-j>", "<Plug>(coc-snippets-expand-jump)")
-- Use <c-space> to trigger completion.
keyset("i", "<c-space>", "coc#refresh()", {silent = true, expr = true})

-- Use `[g` and `]g` to navigate diagnostics
-- Use `:CocDiagnostics` to get all diagnostics of current buffer in location list.
keyset("n", "[g", "<Plug>(coc-diagnostic-prev)", {silent = true})
keyset("n", "]g", "<Plug>(coc-diagnostic-next)", {silent = true})


--vim.cmd([[
--inoremap <silent><expr> <TAB>
      --\ coc#pum#visible() ? coc#_select_confirm() :
      --\ coc#expandableOrJumpable() ? "\<C-r>=coc#rpc#request('doKeymap', ['snippets-expand
      --\ CheckBackSpace() ? "\<TAB>" :
      --\ coc#refresh()

--function! CheckBackSpace() abort
  --let col = col('.') - 1
```

```
  --return !col || getline('.')[col - 1]  =~# '\s'
--endfunction

--let g:coc_snippet_next = '<tab>'
--]])

vim.keymap.del({'x', 'o'}, 'x')
vim.keymap.del({'x', 'o'}, 'X')
11111
  vim.cmd('packadd packer.nvim')
  install_plugins = true
end

require('packer').startup(function(use)
  use 'wbthomason/packer.nvim'
  use 'joshdick/onedark.vim'

  if install_plugins then
    require('packer').sync()
  end
end)

if install_plugins then
  return
end

vim.opt.termguicolors = true
vim.cmd('colorscheme onedark')
```

plugins

```
require('packer').startup(function(use)
  use 'joshdick/onedark.vim'
use "wbthomason/packer.nvim"
use "nvim-lua/plenary.nvim"
use {'nvim-telescope/telescope-fzf-native.nvim', run = 'cmake -S. -Bbuild -DCMAKE_BUILD_TYPE
use "neovim/nvim-lspconfig"          -- Mind the semi-colons
```

```
use 'nvim-tree/nvim-web-devicons'
use {
        'nvim-treesitter/nvim-treesitter',
        run = function() require('nvim-treesitter.install').update({ with_sync = true }) end
    }
--use "honza/vim-snippets"
use {'neoclide/coc.nvim', branch = 'release'}
--use "ncm2/ncm2"
--use "roxma/nvim-yarp"
----use "ncm2/ncm2-bufword"
--use "ncm2/ncm2-path"
use "junegunn/vim-easy-align"
use {
  'nvim-telescope/telescope.nvim', tag = '0.1.0',
  requires = { {'nvim-lua/plenary.nvim'} }
}
use "junegunn/fzf"
use "rhysd/clever-f.vim"
use "ggandor/leap.nvim"
use "jalvesaq/Nvim-R"
use "tpope/vim-surround"
use "machakann/vim-highlightedyank"
--use "sirver/ultisnips"
use "junegunn/vim-peekaboo"
use "junegunn/goyo.vim"
use "junegunn/limelight.vim"
use "tpope/vim-unimpaired"
use "preservim/nerdcommenter"
use "vim-airline/vim-airline"
use "vim-airline/vim-airline-themes"
use "lervag/vimtex"      -- Use braces when passing options
    --use 'williamboman/nvim-lsp-installer' -- Automatically install LSPs
    --use 'jose-elias-alvarez/null-ls.nvim' -- Null ls is used for code formatting and pylin
    --use 'hrsh7th/nvim-cmp' -- Autocompletion plugin
    --use 'hrsh7th/cmp-nvim-lsp' -- Autocompletion with LSPs
end)
```

key maps

```
--[[ keys.lua ]]
local map = vim.api.nvim_set_keymap

-- remap the key used to leave insert mode
map('n', ':', ';', {noremap=true})
map('n', ';', ':', {noremap=true})
map('n', '<S-Space>','<C-u>', {noremap=true})
map('n', '<space><space>','<C-d>', {noremap=true})
map('n', '-','$', {noremap=true})
map('n', '<Space>f','vipgq', {noremap=true})
map('n', '<leader>v','edit ~/.vimrc<cr>', {noremap=true})
map('n', '<leader>a','ggVG', {noremap=true})
map('n', '<leader>S','let @x=@*', {noremap=true})
map('n', '<leader>t','tab split<cr>', {noremap=true})
map('n', '<leader>y','vert sb2<cr>', {noremap=true})
map('n', '<leader>u','UltiSnipsEdit<cr>', {noremap=true})
map('n', '<Leader>0','ls!<CR>:b<Space>', {noremap=true})
map('n', '<leader><leader>','<C-w>w', {noremap=true})
map('n', '<leader>1','<C-w>:b1<cr>', {noremap=true})
map('n', '<leader>2','<C-w>:b2<cr>', {noremap=true})
map('n', '<leader>3','<C-w>:b3<cr>', {noremap=true})
--map('i',  '--','_', {noremap=true})
--map('t',  '--','_', {noremap=true})
map('t',  'ZZ', "q('no')<CR>", {noremap=true})
map('t',  'ZQ', "q('no')<CR>", {noremap=true})
map('v',  '-', '$', {noremap=true})
map('t',  '<leader>0','<C-w>:ls!<cr>:b<Space>', {noremap=true})
map('i',  '<Esc>', '<Esc>`^', {noremap=true})
map('t',  '<leader><leader>', '<C-w>w', {noremap=true})
local builtin = require('telescope.builtin')
vim.keymap.set('n', '<leader>ff', builtin.find_files, {})
vim.keymap.set('n', '<leader>fg', builtin.live_grep, {})
vim.keymap.set('n', '<leader>fb', builtin.buffers, {})
vim.keymap.set('n', '<leader>fh', builtin.help_tags, {})
```

key opts

```
--[[ opts.lua ]]
local opt = vim.opt
opt.relativenumber = true
opt.clipboard='unnamed'
opt.textwidth=80
opt.cursorline = true
opt.hlsearch   = true
opt.splitright= true
opt.hidden   = true
opt.incsearch    = true
opt.showmatch= true
opt.ignorecase= true
opt.smartcase= true
-- opt.background='light'
-- opt.gdefault
-- [[ Context ]]
opt.colorcolumn = '80'          -- str:  Show col for max line length
opt.number = true               -- bool: Show line numbers
-- opt.relativenumber = true       -- bool: Show relative line numbers
opt.scrolloff = 4               -- int:  Min num lines of context
opt.signcolumn = "yes"          -- str:  Show the sign column

-- [[ Filetypes ]]
opt.encoding = 'utf8'           -- str:  String encoding to use
opt.fileencoding = 'utf8'       -- str:  File encoding to use

-- [[ Theme ]]
opt.syntax = "ON"               -- str:  Allow syntax highlighting
opt.termguicolors = true        -- bool: If term supports ui color then enable

-- [[ Whitespace ]]
opt.expandtab = true            -- bool: Use spaces instead of tabs
opt.shiftwidth = 4              -- num:  Size of an indent
opt.softtabstop = 4             -- num:  Number of spaces tabs count for in insert mode
opt.tabstop = 4                 -- num:  Number of spaces tabs count for

-- [[ Splits ]]
```

```
opt.splitright = true            -- bool: Place new window to right of current one
opt.splitbelow = true            -- bool: Place new window below the current one
```

key vars

```
--[[ vars.lua ]]

local g = vim.g
g.t_co = 256
g.background = "dark"
```