Setting up a neovim environment for data science code development

A neovim IDE for R, Python, and Julia

Ronald (Ryy) Glenn Thomas

2024-11-09

Table of contents

1 Introduction

Neovim (a fork of Vim) is a text editor that has several advantages for data science code development. One of its main

Photo by Nathan Waters on Unsplash

attractions (besides being open source) is that it has a number of useful plugins to facilitate working on R, python, and julia code. Also, its modal, keyboard-centric system allows text and code manipulation at potentially far greater speed than conventional, mouse-centric, systems.

In this post we describe a minimal, yet functional setup to allow a quick start with neovim. We also describe a more extensive setup utilizing several of the neovim-only plugins that together allow neovim to provide IDE style code editing and REPL interaction for the three primary data science coding tools: R, Python, and Julia.

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

- add reference for nvim speed *

2 Install neovim.

With minimal effort we can install both the terminal and GUI versions of neovim. The simplist approach is to use homebrew:

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

Suggest set up convenience aliases in `zsh`.

```
> alias ng = "nvim-qt"
> alias nt = "nvim"
```

(mnemonic: the `t` in `nt` is for terminal, the `g` in `ng` is for GUI)

3 Configure neovim

The standard location for `neovim` configuration files on "unix-like" systems is `~/.config/nvim`. The main config file is either

init.vim (vimscript) or init.lua (Lua). In this post we'll focus on lua based configuration.

Below is the file hierarchy we'll construct. In fact all the code in the lua subdirectory could be bundled into the `init.lua` file, but this approach is clearer and cleaner.

```
.
|-- ginit.vim
|-- init.lua
|-- lazy-lock.json
|-- lua
|   |-- basics.lua
|   |-- leap-config.lua
|   |-- nvim-R-config.lua
|   |-- nvim-cmp-config.lua
|   |-- nvim-telescope-config.lua
|   |-- nvim-tree-config.lua
|   `-- treesitter-config.lua
|-- my_snippets
|   |-- all.snippets
|   |-- tex
|   |-- R
|   |-- python
|   |-- julia
|   |-- giles.tex.snipppets
|   |-- mail.snippets
|   |-- r.snippets
|   |-- rmd.snippets
|   |-- snippets.snippets
|   |-- tex.snippets
|   |-- text.snippets
|   `-- txt.snippets
|-- spell
|   |-- en.utf-8.add
|   `-- en.utf-8.add.spl
```

Neovim on its own is useful but limited. We can add functionality using plugins. To install one or more plugins we'll need a plugin manager. There are several to choose from. We'll use the `Lazy` plugin manager for this post. To install the `Lazy` issue the following command at the shell prompt:

```
> git clone https://github.com/folke/lazy.nvim.git \
   ~/.local/share/nvim/lazy/lazy.nvim
```

Add the following code to `init.lua` list the plugins needed to be installed from `github` and "feed" them to `Lazy` for installation.

`Nvim-R`, `Leap`, `UltiSnips`, and `vimtex` need additional configuration. The required code is contained in bespoke files under the `lua` directory.

```
vim.g.mapleader = ","
vim.g.maplocalleader = " "
vim.opt.rtp:prepend("~/.local/share/nvim/lazy/lazy.nvim")
require('plugins')
require('basics')
require('nvim-tree-config')
require('nvim-R-config')
require('nvim-telescope-config')
require('leap').add_default_mappings()
require('leap-config')
require('lualine').setup()
```

List of plugins

```
require('lazy').setup({
--
--minimal data science setup
--
'jalvesaq/Nvim-R',
'lervag/vimtex',
'SirVer/ultisnips',
'jalvesaq/vimcmdline',
--
```

```
--optional utilities
--
"nvim-lualine/lualine.nvim",
"bluz71/vim-moonfly-colors",
'junegunn/vim-peekaboo',
'tpope/vim-commentary',
'francoiscabrol/ranger.vim',
'machakann/vim-highlightedyank',
'tpope/vim-surround',
'ggandor/leap.nvim',
--
--neovim specific
'nvim-lua/plenary.nvim',
'nvim-tree/nvim-web-devicons',
'nvim-tree/nvim-tree.lua',
'nvim-telescope/telescope.nvim',
'nvim-treesitter/nvim-treesitter',
'neovim/nvim-lspconfig',
})
```

4 plugin discussions

5 Setuo Basics

```
local map = vim.keymap.set
local opts = {noremap = true}
vim.cmd([[
"    paste registers into terminal
tnoremap <expr> <C-R> '<C-\><C-N>"'.nr2char(getchar()).'pi'
set background=dark
colorscheme moonfly
let $FZF_DEFAULT_COMMAND = 'rg --files --hidden'
set completeopt=menu,menuone,noinsert,noselect
set number relativenumber
```

```
set textwidth=80
set cursorline
set clipboard=unnamed
set iskeyword-=_
set hlsearch
set splitright
set hidden
set incsearch
set noswapfile
set showmatch
set ignorecase
set smartcase
set gdefault
filetype plugin on
set encoding=utf-8
set nobackup
set nowritebackup
set updatetime=300
set signcolumn=yes
set colorcolumn=80
set timeoutlen=1000 ttimeoutlen=10
let g:UltiSnipsSnippetDirectories = ['~/.config/nvim/my_snippets']
let g:UltiSnipsExpandTrigger="<tab>"
let g:UltiSnipsJumpForwardTrigger="<c-j>"
let g:UltiSnipsJumpBackwardTrigger="<c-k>"
nnoremap <leader>U <Cmd>call UltiSnips#RefreshSnippets()<CR>

autocmd BufWinEnter,WinEnter term://* startinsert
"autocmd TermOpen * exec "normal! i"
]])
map('n', ':', ';', opts)
map('n', ';', ':', opts)
map('n', '<leader>u',':UltiSnipsEdit<cr>', opts)
map('n', '<leader>U','<Cmd>call UltiSnips#RefreshSnippets()<cr>', opts)
map('n', '<localleader><localleader>','<C-d>', opts)
map('n', '-','$', opts)
map('n', '<leader>w','vipgq', opts)
map('n', '<leader>v',':edit ~/.config/nvim/init.lua<cr>', opts)
map('n', '<leader>n',':edit ~/.config/nvim/lua/basics.lua<cr>', opts)
map('n', '<leader>a','ggVG', opts)
```

```
map('n', '<leader>t',':tab split<cr>', opts)
map('n', '<leader>y',':vert sb3<cr>', opts)
map('n', '<leader>0',':ls!<CR>:b<Space>', opts)
map('n', '<leader><leader>','<C-w>w', opts)
map('n', '<leader>1','<C-w>:b1<cr>', opts)
map('n', '<leader>2','<C-w>:b2<cr>', opts)
map('n', '<leader>3','<C-w>:b3<cr>', opts)
map('t',  'ZZ', "q('yes')<CR>", opts)
map('t',  'ZQ', "q('no')<CR>", opts)
map('v',  '-', '$', opts)
map('t',  '<leader>0','<C-\\><C-n><C-w>:ls!<cr>:b<Space>', opts)
map('t',  '<Escape>','<C-\\><C-n>', opts)
map('t',  ',,','<C-\\><C-n><C-w>w', opts)
map('i',  '<Esc>', '<Esc>`^', opts)
```

6 Set up R

```
vim.cmd([[
iabb <buffer> x %>%
iabb <buffer> z %in%
let R_auto_start = 2
let R_enable_comment = 1
let R_hl_term = 0
let R_clear_line = 1
let R_pdfviewer = "zathura"
let R_assign = 2
let R_latexcmd = ['xelatex']
augroup rmarkdown
autocmd!
autocmd FileType rmd,r nnoremap <buffer> <CR>  :call SendLineToR("down")<CR>
autocmd FileType rmd,r nnoremap <buffer> <space>' :call RMakeRmd("default")<cr>
autocmd FileType rmd,r noremap <space>i :call RAction("dim")<cr>
```

```
autocmd FileType rmd,r noremap <space>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 vmap <buffer> <CR> <localleader>sd
autocmd FileType rmd,r nmap <buffer> <space>j <localleader>gn
autocmd FileType rmd,r nmap <buffer> <space>k <localleader>gN
autocmd FileType rmd,r nmap <buffer> <space>l <localleader>cd
augroup END
]])
```

7 Appendix Ubuntu tweaks

8 Appendix, Using multiple configs:

Switching Configs in Neovim • Michael Uloth

Initial bare cofig:

julia .config/test_nvim NVIM_APPNAME=test_nvim nvim