Neuropsyentist

Release 0.0.1

Nov 22, 2020

Setting up your Mac

CHAPTER 1

XCode

XCode is an Integrated Development Environment (IDE) for MacOS.

- What's an IDE? An IDE helps programmers code.
- You can write and edit code, execute commands, and debug.

XCode contains a suite of software development tools made by Apple.XCode provides access to a compiler, text editor and other tools.

1.1 Installation

Open the terminal and enter this command:

```
xcode-select --install
```

- Here is an alternative link for the direct download from Apple.

1.2 XQuartz

XQuartz is an open-source application that connects MacOS graphical frameworks to GUIs.

- XQuartz makes it possible to use GUIs.

1.3 Installation

To install XQuartz, go to the XQuartz site, download the .dmg, and add it to applications.

CHAPTER 2

Package Managers

Homebrew, Anaconda, and Pip are all package installer tools.

2.1 Installation

Command:

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/
↪install.sh)"
```

To install anaconda, go to the Anaconda site, download the 64-Bit Graphical Installer OR the 64-Bit Command Line Installer.

2.2 Need to Know

Why do we need a package manager tool?

- Softwares come with dependencies.
- Package manager tools allow us to download software packages with everything they need.
- They also make sure that packages play nice.
  - They analyze the current environment (all of the existing packages) and install compatible dependencies.

What's the difference between homebrew vs anaconda vs pip?

- Homebrew is a package manager tool for macOS and Linux.
- Anaconda distributes most of the popular scientific computing packages.
  - conda-forge: to install more packages
- Pip is a package manager specific to the Python Package Index (PyPI).

  - It installs all Python package dependencies as they are (regardless of whether they are compatible with existing packages).

Which one should I use to install packages?

- It depends on what kind of package you want to install.
- If mac application, use brew.
- If python package, first try conda then conda-forge. If not, use pip.
  - Remember, anaconda manages dependencies and versions to keep packages compatible.

Installation Paths

- Homebrew is installed in /usr/local.
  - Every time you open up the terminal, it will have access to our /usr/local/bin directory.
  - Installing a package using "brew" will install package in its own directory and then symlink it into /usr/local.
- Anaconda is installed in ~/opt directory when installing with the Graphical Installer.

2.3 Bread and Butter Commands

- Find package:

```
brew search <package>
```

```
conda search <package>
```

- Install:

```
brew install <package>
```

```
conda install <package>
```

- Remove:

```
brew uninstall <package>
```

```
conda remove <package>
```

2.4 Upgrade

```
brew upgrade
```

```
conda upgrade conda
```

2.5 Cheat Sheets

Homebrew Conda Pip

CHAPTER 3

VScode

Visual Studio Code is an open-source code editor made by Microsoft.

- VScode comes with many tools: an integrated terminal, Intellisense (code completion), and Git integration.
- Extensions transform VScode into a powerhouse for using multiple languages, debuggers, themes, etc.
- Cheat Sheet

3.1 Installation

- Direct download or to install using brew:

```
brew cask install visual-studio-code
```

3.2 Bread and Butter Commands

- Launch VS code from the current directory of the terminal:

```
code .
```

- Open command palette:

```
cmd + shift + p
```

- Open integrated terminal:

```
ctrl + ~
```

3.3 Must-have Extensions

- Python: `ms-python.python`
  - Essential for developing anything in Python.
  - Features:
    - Intellisense = auto-complete
    - Linting = enforces a style guide
    - Jupyter Notebooks = execute code in segments/cells
- Community Material Theme: `equinusocio.vsc-community-material-theme`
  - Fit your style using preset themes or make your own!

3.4 Honorable Mentions

- Live Server: `ritwickdey.liveserver`
  - Start up a local, hot-reloadable server for front-end development work
- Live Share: `ms-vsliveshare.vsliveshare`
  - Enables collaboration in real-time without being co-located
- Markdown Linter: `davidanson.vscode-markdownlint`
  - Best styling practices
- Git Lens: `eamodio.gitlens`
  - Advanced git tracking (see who edited what and when)
- Code Spell Checker: `streetsidesoftware.code-spell-checker`
  - Basic spell checker
- Remote Development: `ms-vscode-remote.vscode-remote-extensionpack`
  - Open folders in a container
- Docker: `ms-azuretools.vscode-docker`
  - Manage docker containers.
- Anaconda Extension: `ms-python.anaconda-extension-pack`
  - Manage anaconda settings and envs.

CHAPTER 4

Python

Powerful programming language, well-maintained by development team.

- Versatile! Can be used as a simple calculator, to organize and manipulate data structures, visualize data, for machine learning, and even manipulate MRI data.
- Macs come pre-installed with python 2.7 but most packages will likely require python 3(+).
- To take advantage of the robustness of python, recommend installing libraries.
  - Library = package that uses python to perform pre-set functions
- Some libraries are built upon others–they add functionality to already existing functions.
- Python documentation.
- Resources for learning how to use Python for data science.

4.1 Installation

To install the latest version of Python, go to Python for Mac, download the macOS 64-bit installer, and follow prompts.

4.2 Bread and Butter Libraries

Data Organization & Processing

- Pandas: organize, manipulate, plot single column series data, and data frames tables
- NumPy: short for Numerical Python, perform operations on arrays and matrices
- SciPy: conduct linear algebra, integration, optimization, and statistics

Data Visualization

- Matplotlib: plotting, making pretty figures (line graphs, scatterplots, histograms)
- Seaborn: extends Matplotlib for making figures to visualize statistical tests and models (correlation matrices, connectivity matrices, distributions)
- Plotly: web-based tool; graph plotting library for creating and displaying figures, hovering over to show details.
- Bokeh: for interactive and scalable visualizations through web-browsers (aka cool presentations).

Machine Learning/AI

- Scikit-Learn: group of packages in the SciPy stack
  - Machine learning and data mining. Think classification, clustering, regression, dimensionality reduction, model selection.
- TensorFlow: artificial intelligence library for creating large-scale neural networks (machine learning + deep learning)
- Keras: neural network library
  - TensorFlow's high-level API for building and training Deep Neural Network code
  - Statistical modeling but with images + text

CHAPTER 5

Git

Git is a version control system.

- What's that? A version control system records changes to a file.
- If someone changes a file (i.e. writes to it or deletes from it), git records the differences between the new version and the old version, and maintains a history. Multiple people can collaborate on a project and each person's changes are documented. Fosters collaboration! Documentation

5.1 Installation

Before installing check to see whether you have git:

```
git --version
```

To install:

```
brew install git
```

5.2 Need to Know

- MacOS comes with git preinstalled in /usr/bin/git
- How to work with others?
  - Git allows you to create branches.
  - Branches can serve as separate workflows out of the same parent version which you can then merge back with main to update the parent code.
- Found an error? Fantastic. Git lets you revert back to a previous version.
- Git Cheat Sheet

5.3 Necessary configuration:

- User name and emails can be set for a single repository and globally.

```
git config --global user.name "Name"
git config --global user.email email@email.com
```

5.4 Initial Set-Up

Either pull/clone a remote repository existing on Git to your local machine OR take an existing directory on your local and initialize it with Git.

To clone an existing remote repository in the current directory:

```
git clone https://github.com/user-name/repo-name.git
```

To initialize a local repository:

1. Configure the directory to be a .git directory:

```
git init
```

2. Add a README file to the files currently tracked by git:

```
git add README.md
```

3. Commit the added files so they are now tracked:

```
git commit -m "first commit"
```

4. Rename the branch to main:

```
git branch -M main
```

5. Add the remote (i.e. Github) tracking:

```
git remote add origin https://github.com/user-name/testrepo.git
```

6. Push the staged commit to the remote branch main (-u flag is set-upstream)

```
git push -u origin main
```

5.5 Bread and Butter Commands

- To see all modified files and their status (changed, addded, deleted, commited):

```
git status
```

- To pull the remote repository down to local machine:

```
git pull <remote repository>
```

- To add specific files/directories to be commited.

```
git add <files>
```

- To add all the files to be commited.

```
git add .
```

- To stage your changes before publishing them:
  - The -m flag adds a message describing the changes.
  - These messages are permanent and will be kept in the history of your repository forever!

```
git commit -m "<message>"
```

- To publish your changes to your remote repository for others to use, see, and change:

```
git push <remote repository>
```

- To create a new branch and move into it:
  - -b flag creates the branch.

```
git checkout -b <branch name>
git checkout <branch name>
```

- To list all branches:

```
git branch -a
```

5.6 Best practices

- Pull Before Push Always pull from the branch you are working on to make sure that you have the most recent version of the remote (i.e., Github) repository.
  - This lets you deal with merge conflicts locally on your machine rather than in your remote repository.
- Push Only Working Code (POWC) Only push functional code to your remote repository so that when others checkout your branch, the code works.
  - Main and develop should generally not be pushed directly to.
  - Merge your code into a develop branch instead and have someone else ensure it works before changing the parent code (main).

CHAPTER 6

Virtualenvs

Tool to manage and isolate project dependencies. Documentation

- Reproducibility crisis no more!
  - Have all of the packages you need for a specific project in the exact version needed.
  - Share your env including all packages and their dependencies.

6.1 Bread and Butter Commands

To create a virtual env:

```
conda create --name <env_name> python=3.8 <package>
```

To activate a virtual env:

```
conda activate <env_name>
```

Install a package to the virtual env:

```
conda install numpy
```

To view a list of your envs:

```
conda env list
```

To exit the env:

```
conda deactivate
```

To export your env:

```
conda activate <env_name>
conda env export > environment.yml
```

To duplicate an existing env:

```
conda install -n <env_name> environment.yml
conda activate <env_name>
```

6.2 Need to Know

- Virtual envs will live in your home dir /opt/anaconda3/envs/.

CHAPTER 7

Customization

7.1 Oh My Zsh

- Zsh is a terminal built on the same shell as bash and is now the default on Macs.
- Oh My Zsh is a framework that wraps around the zsh terminal and allows use of plugins and themes.

7.2 Installation

To install Oh My Zsh:

```
sh -c "$(curl -fsSL https://raw.github.com/ohmyzsh/ohmyzsh/master/tools/install.sh)"
```

To install Powerlevel10k theme:

```
git clone --depth=1 https://github.com/romkatv/powerlevel10k.git ${ZSH_CUSTOM:-$HOME/.
↪oh-my-zsh/custom}/themes/powerlevel10k
```

To install plugins for the terminal:

- Auto suggestions

  ```
  git clone https://github.com/zsh-users/zsh-autosuggestions ${ZSH_CUSTOM:-~/.oh-my-
  ↪zsh/custom}/plugins/zsh-autosuggestions
  ```

- Auto highlighting commands

  ```
  git clone https://github.com/zsh-users/zsh-syntax-highlighting.git ${ZSH_CUSTOM:-~
  ↪/.oh-my-zsh/custom}/plugins/zsh-syntax-highlighting
  ```

- Auto completions

```
git clone https://github.com/zsh-users/zsh-completions ${ZSH_CUSTOM:=~/.oh-my-zsh/
↪custom}/plugins/zsh-completions
```

7.3 P10k fonts

Though not necessary, the Powerlevel10k theme has recommended fonts. Install them here.

7.4 Updating .zshrc

All of the configuration settings for OMZ will be in the ~/.zshrc file.

- To update, in the terminal enter:

```
nano ~/.zshrc
```

- To change the zsh_theme:

```
ZSH_THEME="powerlevel10k/powerlevel10k"
```

- Add this into the plugins section:

```
  plugins=(
          git
          zsh-autosuggestions
          zsh-syntax-highlighting
          zsh-completions
  )
```

- This is added to bash_profile when anaconda is installed.

```
# >>> conda initialize >>>
# !! Contents within this block are managed by 'conda init' !!
__conda_setup="$('/opt/anaconda3/bin/conda' 'shell.bash' 'hook' 2> /dev/null)"
if [ $? -eq 0 ]; then
    eval "$__conda_setup"
else
    if [ -f "/opt/anaconda3/etc/profile.d/conda.sh" ]; then
        . "/opt/anaconda3/etc/profile.d/conda.sh"
    else
        export PATH="/opt/anaconda3/bin:$PATH"
    fi
fi
unset __conda_setup
# <<< conda initialize <<<
```

- Close and re-open Terminal to go through powerlevel10k installation.

To reconfigure the powerlevel10k theme:

```
p10k configure
```

CHAPTER 8

Indices and tables

- genindex
- modindex
- search