Distributed Computing

This brief tutorial is designed to get you up-to-speed with regard to handling large datasets in Python and R. We'll be using Amazon Web Services for this tutorial.

Since the readers of this document have varying levels of experience with Linux, I've tried to explain this in some depth. Many terms which may be unfamiliar to you are linked to expository pages; if you are unfamiliar with a linked term, I strongly urge you to spend a minute or two reading at least the introduction of the linked page.

Getting started with Amazon Web Services

- Register an account on the AWS website. You'll have to enter your billing info, but we'll be working with a cheap (under $0.02/hour) instance throughout this demonstration for which Amazon provides 31 free days of usage each month.

- At the top-right of AWS, make sure that the "US West (Oregon)" region is selected. You'll have the fastest connection to the N. California server, but speeds to Oregon are unnoticeably slower and AWS is substantially cheaper in that region due to high demand for N. California servers.

- Click on "Services" in the top left corner and go to "Console Home", where you'll see an overview of all the services which AWS offers. Click on the top option in the "Compute" section, labeled "EC2". This stands for "Elastic Compute Cloud".

- Click on "Launch Instance" to make your first EC2 instance!

    - *Step 1:* You can choose from a wide variety of preconfigured operating systems. Scroll through the list to get a sense of the options available to you, then scroll back to the top and select "Amazon Linux AMI", which comes configured with a wide variety of programming-related tools.

– *Step 2:* Select the `t2.micro` instance type, which should be selected by default and is labeled as "Free tier eligible".

– *Step 3:* Read the information tooltip (hover over the "i" icon) for each option. For the "Subnet" option, select `us-west-2a`. (The subnet labels are randomized for each user, so that the one labeled `us-west-2a` doesn't get a disproportionate number of people.)

– *Step 4:* Briefly read about the different types of EBS storage available. (You can think of Amazon's EBS service as being analogous to a choice of hard drive.) Change the size of the "Root" volume, where your operating system files will reside, to "10 GiB", and select the General Purpose SSD (`gp2`) type.

– *Step 5:* If you have a lot of instances, you can add tags to them for organization. Skip this step.

– *Step 6:* Configuring the security group is important; otherwise, you won't be able to connect to your server properly. Select "Create a **new** security group" and give it any name or description you want. The default rule for "SSH" allows you to connect to your server via the SSH protocol from any IP address (because the "Source" is set to "Anywhere"). We would like to allow web browsers to connect to the server, so click "Add Rule", select "[HTTP](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol" for "Type", and select "Anywhere" for "Source".

– *Step 7:* Review the information and click "Launch". If you haven't used Amazon Web Services before, it will prompt you to create an authentication key. Do so and download the key file.

Congratulations: you've launched your first AWS server!

Amazon EC2 instances are billed on a *per-hour* basis, with different types of servers having different costs. You can turn instances on and off from the AWS web interface. If you'll be using an AWS instance consistently over a long period of time, it's possible to buy a reserved instance, where you pay a large upfront cost for a lower overall cost.[1] However, we won't be doing that at the moment, although you can spend a couple minutes looking around in the "Reserved Instances" tab to get a sense for how it works.

---

[1]The reservation is *automatically applied* to any eligible instances and is not its own separate instance. As soon as you buy one, no additional action is required; it's purely a change in how your running instances are billed.

Logging on to your server

- Follow Amazon's official guide on how to connect to your server. You will need the key file which you previously created and downloaded.

- Following the same instructions, transfer a small test file (anything will do) to your home directory, located at `/home/ec2-user/`.

Next, we'll set up the server with some standard programming utilities.

- Software can be installed with the `yum` command. To search for packages, type `yum search "x"`, where `x` is the string to search for in package names and descriptions. (For example, try `yum search "emacs"`.) The part of the package name before the period (".") is the *name* of the package, and you can install a package with the command `sudo yum install <pkg>` (where `<pkg>` is the name of the package and does not include the `<>` brackets).

- Python comes preinstalled on the Amazon Linux AMI. Type `python --version` to find out which version of Python is installed.

- We would like to install Python 3, not Python 2. Search for packages with `python3` in their name and install the main Python 3 package as well as the `-devel` addon package. To figure out the command associated with Python 3, type `python` and then press Tab to see the autocomplete options. Run the correct command with the `--version` flag to verify that the correct version of Python has been installed.

- Briefly read about virtual environments in Python. It's easiest to work with Python on a server within a virtual environment. To create one, first determine the *path* of your Python 3 executable with the `which` command (try `which python` for an illustration of how it works). Next, type `virtualenv -p <path> ~/venv`, where `<path>` is the path to your Python 3 executable.

- Now that your virtual environment is set up, you can access the Python executable associated with your virtual environment by typing `~/venv/bin/python`. Call it with the `--version` flag to verify that you successfully created a virtual environment with Python 3. Next, Python packages can be installed with by calling `~/venv/bin/pip install <pkg>`, where `<pkg>` is the name of the Python package to install.

You can run Python scripts by calling `<exec> script.py`, where `<exec>` denotes the path to a Python executable, but you can also play around in the Python interpreter itself by just running `<exec>`.

- If you haven't tried using the Python interpreter before, then do so. You can exit the interpreter with `quit()`.

Using Python with AWS

We will use Python to perform some natural language processing on a dataset of Amazon reviews. Although the specific dataset we will work with is not very large, these methods generalize well to much larger datasets.

- Read the description of the Amazon review dataset, including the definition of a "5-core" subset of the data.

- Change the directory to /home/ec2-user/ with the `cd` command and download the 5-core subset of video game review data by running `wget url_of_data`. (*Note:* You can usually paste into a Linux command line with `Shift+Insert`.) After it's done, you can display the contents of the directory with `ls` to verify that it downloaded correctly.

- The data comes in a compressed `.gz` format. Use `gunzip <file>` on the file to decompress it, and then remove the compressed file with `rm <file>`. The extracted data is in a `.json` format. Use `head -n 1 <file>` to view the first line of the file and see what the format looks like.

Getting a subset of the data

When you work with extremely large datasets, it's often the case that you want to take a *subset* of the data when performing exploratory analyses (so you get results faster), and only run full analyses on the entire dataset when you have a better idea of what methods work best.

We'll take a random 10% subset of Amazon video game reviews.

- Use `wc -l <file>` to count the number of lines in the file.

You can use `nano <file>` to open a simple text editor suitable for writing a short Python script. If you prefer, you can install and use Emacs, Vim, etc. In the nano editor, you can press `Ctrl+O` to save a a file, `Ctrl+W` to search, and `Ctrl+X` to exit.

- Write a Python script following these specifications:

  - Use the `random.sample()` function from the `random` library to sample 10% numbers from 0 to $L - 1$, where L is the number of lines in the JSON file.

  - You can use the following Python code to iterate over the lines of a file without reading the entire file into memory simultaneously:

    ```python
    f = open('file.json')
    for i, line in enumerate(f):
        if i == 25:
            # This is the 26th line
    ```

```
        elif i == 30:
            # This is the 31st line
        elif i > 100:
            # Exit the loop
            break
    f.close()
```

Referring to the Python documentation, especially the section on file input and output, write code which iterates through the lines of the data file.

– For the *i*th line, check if $i - 1$ is among the subset of line numbers previously generated; if so, write the *i*th line to subset.json. (You should open subset.json once at the beginning of the file and close it at the end instead of reopening the file every time you want to write a line to it.) Make sure to separate each written line from the next with a '\n' (newline character).

– The code may take some time to run. If you like, you can print out the line number every 1000 lines for an indication of the script's progress.

When calling the Python script, remember to call the executable corresponding to your installation of Python 3 in the virtual environment.

Next, we'll extract just the text of the reviews in preparation for doing some natural language processing with Python.

• Write a Python script which reads through the lines of subset.json. For each line, call json.loads() on it (from the json library) to turn the line into a dictionary, and then write the 'reviewText' value of the dictionary to vidreviews.txt. (Make sure to separate each line from the next with a space or newline.)

For an indication of what's happening, you can run the following Python code directly in the console:

```
f = open('subset.json')
l = f.readline()
f.close()
import json
d = json.loads(l)
print(d)
print(d['reviewText'])
```

Running **vec2pca**

Finally, you will be running the vec2pca algorithm on your server to analyze the subset of Amazon reviews.

- Read a short description of the vec2pca algorithm.

- In order to compile some of the Python packages necessary, you will need to yum install the gcc, gcc-c++, blas-devel, lapack-devel, atlas-devel, and gfortran packages. While installing them, read the introductions to their linked Wikipedia pages so you know what you're installing![2]

- Since the t2.micro EC2 instance doesn't have enough RAM to properly install the necessary packages, you'll have to enable swap space. Run these commands to do so:

```
sudo /bin/dd if=/dev/zero of=/var/swap.1 bs=1M count=1024
sudo /sbin/mkswap /var/swap.1
sudo chmod 0600 /var/swap.1
sudo /sbin/swapon /var/swap.1
```

- Using the version of pip associated with your virtual environment, install the following libraries: numpy, scipy, sklearn, gensim, nltk, beautifulsoup4, plac, and pandas. You will need to run three separate calls to pip: one to install numpy, another to install scipy, and a last one to install everything else (you can include them all in the same line). This is because some libraries are dependent upon others.

  These packages may take some time to install. This is a good time to go back and look at any explanatory pages about various packages and utilities which you would like to learn more about.

- The nltk library will need some configuration. Open a command-line session of Python, import the nltk library, and run nltk.download(). Enter d (for "Download") and then enter punkt. Afterwards, use q followed by quit() to return to the shell.

- Install the git package (using yum). Go to the Github page for vec2pca and click on the green "Clone or download" button. Copy the URL and run git clone <url> to download the Github repository's files.

- Finally, cd into the vec2pca directory and run ~/venv/bin/python vec2pca.py <input> <output>, where <input> denotes your input file of review text and <output> is a file to which the algorithm's output will be written. Give the output file a .html extension.

Viewing the results of `vec2pca`

vec2pca gives us results in HTML format, which web browsers know how to handle. We'll set up a webserver on our EC2 instance so we can see the results

---

[2]In the future, you can do sudo yum groupinstall "Development Tools", which will install a lot of unnecessary packages but will probably give you all the general-purpose compilation-required packages you'll need.

online.

- Install `httpd`, which is a standard, commonly-used web server software package. To start the HTTP server, run `sudo /etc/init.d/httpd start`. We want the HTTP server to automatically start whenever the server is rebooted, so run `sudo chkconfig --levels 3 httpd on`.

- In the Amazon AWS EC2 Instances page, click on your instance. In the "Description" tab below, copy the "Public IP" and paste it into your web browser's URL bar.

Each time we restart the server, its public IP will change, which isn't good for a website! Fortunately, Amazon's Elastic IP service allows us to associate a specific, unchanging IP with our server.

- In the EC2 overview page, click on "Elastic IPs" on the left (under "Network & Security"). Click on "Allocate New Address" and then "Yes, Allocate". You'll see a single entry show up in the table of Elastic IPs. Select it, click "Actions", and select "Associate Address". In the popup window, select your EC2 instance and then click "Associate". When you return to the "Instances" page, you'll see that the "Elastic IP" field, directly underneath "Public IP", is now filled in. Verify that it works by navigating to the Elastic IP of your instance in your web browser.

- The webserver's files should go in `/var/www/html`. Use the `cp` utility to copy the output of `vec2pca` into a file called `index.html` located in the aforementioned directory. (You may need to prefix `cp` with `sudo`, because the `/var` directory requires elevated privileges to modify.)

- Navigate to the Elastic IP and view the results! (One extreme end of each principal component is located at the top of the table and the other end is at the bottom.) Compare your results with the results of those around you, since you all took different subsets of the same dataset. How much variation is there?

Using RStudio in a web browser

We can run RStudio in a web browser! For your `t2.micro` instance, it won't be much better than your own computer, but we'll go through the process so you know how to set it up for yourself.

- Run the two commands RStudio Server installation instructions for "Red-Hat/CentOS 6 and 7" under the "64bit" section.

You can now use the `rstudio-server` command to start and stop RStudio Server.

- Type `rstudio-server` on its own to view a list of commands.

- Use `rstudio-server status` to check the status of RStudio Server.

- Use `sudo rstudio-server start` and `sudo rstudio-server stop` to start and stop RStudio Server, verifying that its status is changing correctly with the `status` command. Can you start and stop the server without elevating privileges with `sudo`?

- Read RStudio Server: Getting Started and try to nagivate to `http://<Elastic IP>:8787`. Oops – we forgot to allow connections through port 8787 in our security group! In the EC2 dashboard, select "Security Groups" on the left, select the security group associated with your EC2 instance, click "Actions" and then "Edit inbound rules", and add a new rule of type "Custom TCP Rule". Set the "Port Range" to 8787 and the "Source" to "Anywhere".

- Try accessing port 8787 again through your web browser. Oops – we have to log in, but how? Create a new user with `sudo adduser <username>` and set its password with `sudo passwd <username>`, and then try logging into RStudio Server.

- Play around with the web interface a little bit and compare its performance to the speed of RStudio on your own computer!