Maximising the value of data
Through software publishing

Ian Mulvany
Head of product innovation SAGE Publishing
Learn Data Science Online. Courses by Social Scientists, for Social Scientists.

UPCOMING COURSES

SAGE Campus courses have been developed with a social science academic expert involved at every stage of design, authoring, and development. A key feature of our courses is the learning support we offer - we make sure that you have support and guidance from experts throughout, giving you the best possible online learning experience.
Acquire Data
Traceback (most recent call last):
  File "NSF-to-es.py", line 7, in <module>
    from commonGrant import CommonGrantInfo
ImportError: No module named commonGrant
http://www.nature.com/encode/#/threads
The data and code remain separate from the paper.

Hosting is $$$

Complex instructions

Large artefacts

Cloud Instance

Install an Amazon EC2 instance to examine the figures in the cloud.

1. If you do not already have an AWS account, register with Amazon.
2. Follow the detailed instructions (how to get your access key and secret key.
3. Go to CloudShell and enter a name for your cluster, the password for the interface, and your keys, and submit the form. The request may take a moment to complete.
4. After saving the private key, navigate to the address presented on the page. This is the public web address for your instance, and may not respond immediately depending on how fast the instance boots. It should initially say "No Instance Running," but click the link to go to the Cloud Console.
5. When the CloudShell interface is accessible (using the password you defined previously), you should see the "Initial Cluster Configuration" window. Click "Show more startup options" and enter the parameters:

```
-cm=cdw62009-563d3b0046632151.0333d004/vm/cluster/2013-09-05-14-59/" (no spaces, but do include everything else) in the "cluster name" cluster startup box. Click the configure button.
```
6. CloudShell will now customize your instance as a clone of the BN2120 machine. This process may take a minute or two, but is finished when the log shows "Some running system is installed and all applications are [green] as indicated by the interface.
7. Now, simply ssh into your instance as the 'figuris' user and you're good to go. The command should look something like this: ssh clouddmn_key[.].pl
8. Remember to terminate your instance when finished to avoid incurring additional costs. You can always start a new instance with the same cluster name to get back to where you were, until you terminate and delete (via the cloud console).

Downloadable Virtual Machine

1. Download VirtualBox.
2. Download the Virtual Machine BN2120.035.1GB (MB2120.035).
3. Import the VM into VirtualBox.
   1. First, make a backup copy of the downloaded .ova file. If something goes wrong you can always make a new copy.
   2. Import the VM into VirtualBox by either starting the downloaded .ova file directly or by launching VirtualBox and navigating to File -> Import Appliance and opening the file.
4. This will display the Appliance Import Settings window. Click the Import button.
5. It may take several minutes for VirtualBox to import the VM. Once it is done, a new VM will appear in the left pane in the "powered off" state.
6. The default settings should be mostly appropriate, but note that the VM is running in VMware Fusion. You can find this for your virtual machine in Settings -> System -> Acceleration.
7. Start your new BN2120 VM.
8. Log in as the user "figuris" with the password "PP46BB204" (case sensitive).

At this point, you're ready to go (see note below). From the figuris user home directory, relevant directories are:

- bin - some executables and scripts in common between different analysis.
- common/data - data in common between some analysis. (Structure is the same as the public ftp archive as given in the supplementary info).
- figures - R versions of the code for each of the 10 figures in the BN2120 integration paper. Directories are numbered as for the final figures.
- lib - R libraries used in common
- manuscript - copies of the submitted version of the manuscript, supplementary information and figures.
- R - required R packages
- supplementary - code for supplementary information and tables, plus some code dropped during its review process.
- tables - code for the generation of main paper table 1. Other tables are in supplementary.

Overall to run the code for a figure or supplementary info, you should cd into the relevant directory, where there should be a README file with step by step instructions for running the code. In some cases, because the analysis is the result of a long and complex pipeline, the code here may run on an intermediate result. Also in some cases the full analysis requires large scale analysis of many datasets which are typically implemented on a compute farm. In these cases we have provided a subset of the analysis as an example, which can then be scaled up if the user chooses to do so.

Note: There is an issue with the currently available virtual machine preventing one of the figures from running. The download available here will be fixed shortly, but to fix this on your copy of the VM you should be able to execute the following commands:

```
  figuris@figuris-vm:--> cd /mnt/galaxyData
  figuris@figuris-vm:/mnt/galaxyData> sudo rm -r encode_files
  figuris@figuris-vm:/mnt/galaxyData> sudo rm -r encode_files
```
Physical Hardware (Host)

Virtual Machine Approach

- VM system (Hypervisor)
- Operating System
- Physical Hardware (Host)

- App
- OS
- Guest OS

OS

App

Container / Docker Approach

- Docker Engine
- Operating System
- Physical Hardware (Host)

- App 1
- App 2
- App 3

- Shared Dependencies
- Shared Libraries

LOTS O’SPACE!!
<table>
<thead>
<tr>
<th>Hosts per box</th>
<th>Virtual Machines</th>
<th>Containers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tens</td>
<td>Thousands</td>
</tr>
<tr>
<td>Startup time</td>
<td>Minutes</td>
<td>Fractions of a second</td>
</tr>
</tbody>
</table>
UPCOMING COURSES

SAGE Campus courses have been developed with a social science academic expert involved at every stage of design, authoring, and development. A key feature of our courses is the learning support we offer - we make sure that you have support and guidance from experts throughout, giving you the best possible online learning experience.

Browse courses

Interactive Visualization with R

Introduction to Applied Data Science Methods

Fundamentals of Quantitative Text Analysis

Practical Data Management with R

Introduction to Python

Introduction to Applied Data Science Methods for Social Scientists

Price: $399

Instructors: Dr. Claudia von Vacano, D-Lab, University of California, Berkeley and Christopher Hench, D-Lab, University of California, Berkeley

READ FULL COURSE INFORMATION
Module 10: Machine Learning

Notebook 1: Data Preprocessing for Machine Learning

Dataset Description
In this notebook, we're going to preprocess the US Census dataset to make it usable for machine learning. Throughout this notebook, if you need to access the documentation describing the dataset, follow the link here.

Loading the data
Let's load the CSV file containing the US Census data using pandas.

```python
In [ ]:
import pandas as pd
import matplotlib
%matplotlib inline

infile = '...'/'data/CENSUS/2015_ACS_sf_oak_hay_SUBSET.csv'

df = pd.read_csv(infile)
```

Exploring the data
What do the data look like? Recall that this is American Community Survey datathat contains a wide variety of demographic information. You can get dense summary statistics with basic functions in pandas. Use `df.head()` to see the first few rows of the data:

```python
In [ ]:
df.head()
```
Let's be clear exactly what we're trying to achieve. We're looking for a Python regular expression which, when used with the `re.search` function, will match all of the winners but none of the losers. We can define the function `mistakes` to return a set of misclassifications; if `mistakes` is empty then the regular expression is verified correct:

```python
def mistakes(regex, winners, losers):
    """The set of mistakes made by this regex in classifying winners and losers."
    return {"Should have matched": " + W
            for W in winners if not re.search(regex, W) | "Should not have matched": " + L
            for L in losers if re.search(regex, L)}

def verify(regex, winners, losers):
    assert not mistakes(regex, winners, losers)
    return True
```

Let's check the xkcd regex:

```python
xkcd = "b([-nrt][coyle][mtg][a][iso][nmll][ae]d[lev][sh][lnd]i][pol]ls"

mistakes(xkcd, winners, losers)
```
Google

\(~33000\text{/s}\)
Summary

can solve the problem of software dependencies

allows “live coding” to be delivered over the web

can wrap code, data and environment together

growing ecosystem of tools to support containers

requires a level of infrastructural sophistication