World income inequality project

BASH workshop

Speaker: Dr. Shruti Kulkarni

Description of the project

Introduction

- The World Income Inequality Database (WIID) presents information on income inequality for developed, developing, and transition countries. It provides the most comprehensive set of income inequality statistics available and can be downloaded for free.*
- The latest version of the WIID, released in May 2020, covers 200 countries (including historical entities), with over 11,000 data points in total.
- From this data, using Bash (csvkit, sed, awk, grep,etc.) we will explore different features.
- Finally find an interesting fact that despite of 11000 + data points, we have only 1.29 % of the datapoints are from the Non-EU and poor countries.

Attributes

This simple dataset contains the following fields cut from the original WIID dataset:

• Country – country name.

• EU membership- Whether the country is a current EU member or non-EU.

• Gini coefficient - Gini coefficient as reported by the source (in most cases based on microdata).

• Income group - World Bank classification by country income.

• GDP- Gross domestic product (GDP) converted to 2017 US$ in per capita terms, integrated series.

• Population - Population of countries from the UN population prospects.

For User guide, data sources/methodology - read here.

*UNU-WIDER, World Income Inequality Database (WIID). Version 31 May 2021. https://doi.org/10.35188/UNU-WIDER/WIID-310521

Installing CSV kit

Instructions

- Head command is used to visualize the initial few lines of the dataset or text files.
- However, as you see the output of head command for our dataset is not very interesting in presentation.
- Therefore, to modify and work around with csv files we will install csvkit tool.
- csvkit is a command line tool for working with CSVs files.
- Install csvkit by entering the following commands in the terminal*

$sudo apt update

$sudo apt install csvkit

```
sk@DESKTOP-7TPU1IA:~$ head IEdataset.csv
id,country,2-digit country code in ISO 3166-1 alpha-2 format,year,gini,gdp,population
1,Afghanistan,AF,2008,29,1484,27722282
2,Afghanistan,AF,2012,33,2075,31161378
3,Afghanistan,AF,2017,31,2058,36296108
4,Albania,AL,1996,27.01,5011,3098699
5,Albania,AL,2002,31.74,6895,3126183
6,Albania,AL,2005,30.6,8208,3086810
7,Albania,AL,2008,29.98,10119,3002683
8,Albania,AL,2012,28.96,11462,2914091
9,Albania,AL,2014,34.6,11828,2896307
sk@DESKTOP-7TPU1IA:~$ sudo apt install csvkit
[sudo] password for sk:
Reading package lists... Done
Building dependency tree
Reading state information... Done
csvkit is already the newest version (1.0.2-2).
0 upgraded, 0 newly installed, 0 to remove and 119 not upgraded.
```

*Reference: csvkit documentation: https://csvkit.readthedocs.io/en/latest/

Visualizing the head and the tail of the data

Syntax

- $head <filename> | csvlookup #will display first 10 lines of data.
- $head –n 25 <filename> | csvlookup #will display first 25 lines of data.
- Now let's visualize our world income inequality dataset:
- $head IEdataset.csv | csvlookup
- To visualize first 25 lines of data:
- $head –n 25 IEdataset.csv | csvlookup
- Tail command is similar to the head command in syntax and the only difference is that it displays the bottom ten rows of the data.

```
sk@DESKTOP-7TPU1IA:~$ head IEdataset.csv | csvlook
| id | country     | year | gini | gdp   | population | eu_membership | oecd    | incomegroup         |
| -- | ----------- | ---- | ---- | ----- | ---------- | ------------- | ------- | ------------------- |
|  1 | Afghanistan | 2008 | 29.0 | 1484  | 27722282   | Non-EU        | Non-OECD | Low income         |
|  2 | Afghanistan | 2012 | 33.0 | 2075  | 31161378   | Non-EU        | Non-OECD | Low income         |
|  3 | Afghanistan | 2017 | 31.0 | 2058  | 36296108   | Non-EU        | Non-OECD | Low income         |
|  4 | Albania     | 1996 | 27.1 | 5011  | 3098699    | Non-EU        | Non-OECD | Upper middle income |
|  5 | Albania     | 2002 | 31.7 | 6895  | 3126183    | Non-EU        | Non-OECD | Upper middle income |
|  6 | Albania     | 2005 | 30.6 | 8208  | 3086810    | Non-EU        | Non-OECD | Upper middle income |
|  7 | Albania     | 2008 | 29.9 | 10119 | 3002683    | Non-EU        | Non-OECD | Upper middle income |
|  8 | Albania     | 2012 | 28.9 | 11462 | 2914091    | Non-EU        | Non-OECD | Upper middle income |
|  9 | Albania     | 2014 | 34.6 | 11828 | 2896307    | Non-EU        | Non-OECD | Upper middle income |
```

```
sk@DESKTOP-7TPU1IA:~$ tail IEdataset.csv | csvlook
| 21244 | Slovakia | 2019 | 25.66 | 32730 | 5457012 | EU | OECD | High income |
| ----- | -------- | ---- | ----- | ----- | ------- | -- | ---- | ----------- |
| 21245 | Slovakia | 2019 | 33.17 | 32730 | 5457012 | EU | OECD | High income |
| 21246 | Slovakia | 2019 | 25.0  | 32730 | 5457012 | EU | OECD | High income |
| 21247 | Slovakia | 2019 | 22.75 | 32730 | 5457012 | EU | OECD | High income |
| 21248 | Slovakia | 2019 | 23.69 | 32730 | 5457012 | EU | OECD | High income |
| 21249 | Slovakia | 2019 | 30.68 | 32730 | 5457012 | EU | OECD | High income |
| 21250 | Slovakia | 2019 | 26.39 | 32730 | 5457012 | EU | OECD | High income |
| 21251 | Slovakia | 2019 | 24.83 | 32730 | 5457012 | EU | OECD | High income |
| 21252 | Slovakia | 2019 | 25.69 | 32730 | 5457012 | EU | OECD | High income |
| 21253 | Slovakia | 2019 | 33.17 | 32730 | 5457012 | EU | OECD | High income |
```

Sorting alphabetically with sort vs csvsort

- SORT command sorts the words in a file alphabetically.
- For example, let's say we want to sort words from a text file 'words.txt' alphabetically, the syntax is:
- sort words.txt or sort <filename.txt>
- Syntax to sort by column using sort command:
- $sort -t"," -k2 -r -u <filename>.csv

Delimeter: here it is comma

Column number: here it is 2nd column

Sorting order: -r is to specify reverse alphabetical sorting.

-u is to specify unique words.

- As you can see, it is pretty annoying to type the entire command without the csv tool.
- More hassle free approach is to use csvsort:
- $csvsort –c2 –r <filename>.csv

Column number: here it is 2nd column

Sorting order: -r is to specify reverse alphabetical sorting.

```
sk@DESKTOP-7TPU1IA:~$ sort -t"," -k2 -r -u IEdataset.csv > newdataIE.csv
sk@DESKTOP-7TPU1IA:~$ head new
newdataIE.csv  newthesis.tex
sk@DESKTOP-7TPU1IA:~$ head newdataIE.csv
id,country,2-digit country code in ISO 3166-1 alpha-2 format,year,gini,gdp,population
20792,Zimbabwe,ZW,2017,44.34,3028,14236599
20791,Zimbabwe,ZW,2011,43.15,2556,12894323
20788,Zimbabwe,ZW,2011,42.3,2556,12894323
20790,Zimbabwe,ZW,2011,39,2556,12894323
20789,Zimbabwe,ZW,2011,37,2556,12894323
20786,Zimbabwe,ZW,1995,74.6,3226,11410721
20787,Zimbabwe,ZW,1995,70.3,3226,11410721
20785,Zimbabwe,ZW,1990,56.8,3324,10432409
20782,Zimbabwe,ZW,1969,66.3,2664,5111326
```

```
sk@DESKTOP-7TPU1IA:~$ csvsort -c2  -r IEdataset.csv | head -n 5
id,country,year,gini,gdp,population,eu_membership,oecd,incomegroup
20781,Zimbabwe,1945,46,,,Non-EU,Non-OECD,Lower middle income
20782,Zimbabwe,1969,66.3,2664,5111326,Non-EU,Non-OECD,Lower middle income
20783,Zimbabwe,1969,62.9,2664,5111326,Non-EU,Non-OECD,Lower middle income
20784,Zimbabwe,1969,62.3,2664,5111326,Non-EU,Non-OECD,Lower middle income
sk@DESKTOP-7TPU1IA:~$ csvsort -c2  -r IEdataset.csv | head -n 5 | csvlook
|     id | country  |  year | gini |   gdp | population | eu_membership | oecd     | incomegroup          |
| ------ | -------- | ----- | ---- | ----- | ---------- | ------------- | -------- | -------------------- |
| 20,781 | Zimbabwe | 1,945 | 46.0 |       |            | Non-EU        | Non-OECD | Lower middle income  |
| 20,782 | Zimbabwe | 1,969 | 66.3 | 2,664 |  5,111,326 | Non-EU        | Non-OECD | Lower middle income  |
| 20,783 | Zimbabwe | 1,969 | 62.9 | 2,664 |  5,111,326 | Non-EU        | Non-OECD | Lower middle income  |
| 20,784 | Zimbabwe | 1,969 | 62.3 | 2,664 |  5,111,326 | Non-EU        | Non-OECD | Lower middle income  |
```

Using csvcut and sort by unique

Instructions

- Let's say we are interested only in column 7 to 9
- We will use csvcut command to to divide a file into parts and write the selected columns to the output.
- $csvcut -c7-9 IEdataset.csv | head –n 5 | csvlook

- Now lets say we want to find out –

How many entries belong to the EU or non-EU?

- An interesting command "uniq –c" removes duplicate entries.
- Piping with cut and sort commands to the uniq command, we can see the amazing result.
- $ csvcut -c7-9 IEdataset.csv | sort | uniq -c
- Now, we can see just by the overview of the output that maximum i.e. 8116 data entries belong EU, OECD as well as high income countries.

```
sk@DESKTOP-7TPU1IA:~$ csvcut -c7-9 IEdataset.csv | head -n 5 | csvlook
| eu_membership | oecd     | incomegroup         |
| ------------- | -------- | ------------------- |
| Non-EU        | Non-OECD | Low income          |
| Non-EU        | Non-OECD | Low income          |
| Non-EU        | Non-OECD | Low income          |
| Non-EU        | Non-OECD | Upper middle income |
sk@DESKTOP-7TPU1IA:~$ csvcut -c7-9 IEdataset.csv | head -n 5 | csvlook | sort | uniq -c
      1 | ------------- | -------- | ------------------- |
      3 | Non-EU        | Non-OECD | Low income          |
      1 | Non-EU        | Non-OECD | Upper middle income |
      1 | eu_membership | oecd     | incomegroup         |
sk@DESKTOP-7TPU1IA:~$ csvcut -c7-9 IEdataset.csv | sort | uniq -c
    659 EU,Non-OECD,High income
    237 EU,Non-OECD,Upper middle income
   8116 EU,OECD,High income
   1028 Non-EU,Non-OECD,High income
    267 Non-EU,Non-OECD,Low income
   2049 Non-EU,Non-OECD,Lower middle income
   3708 Non-EU,Non-OECD,Upper middle income
   3868 Non-EU,OECD,High income
    743 Non-EU,OECD,Upper middle income
      1 eu_membership,oecd,incomegroup
```

Text processing - GREP

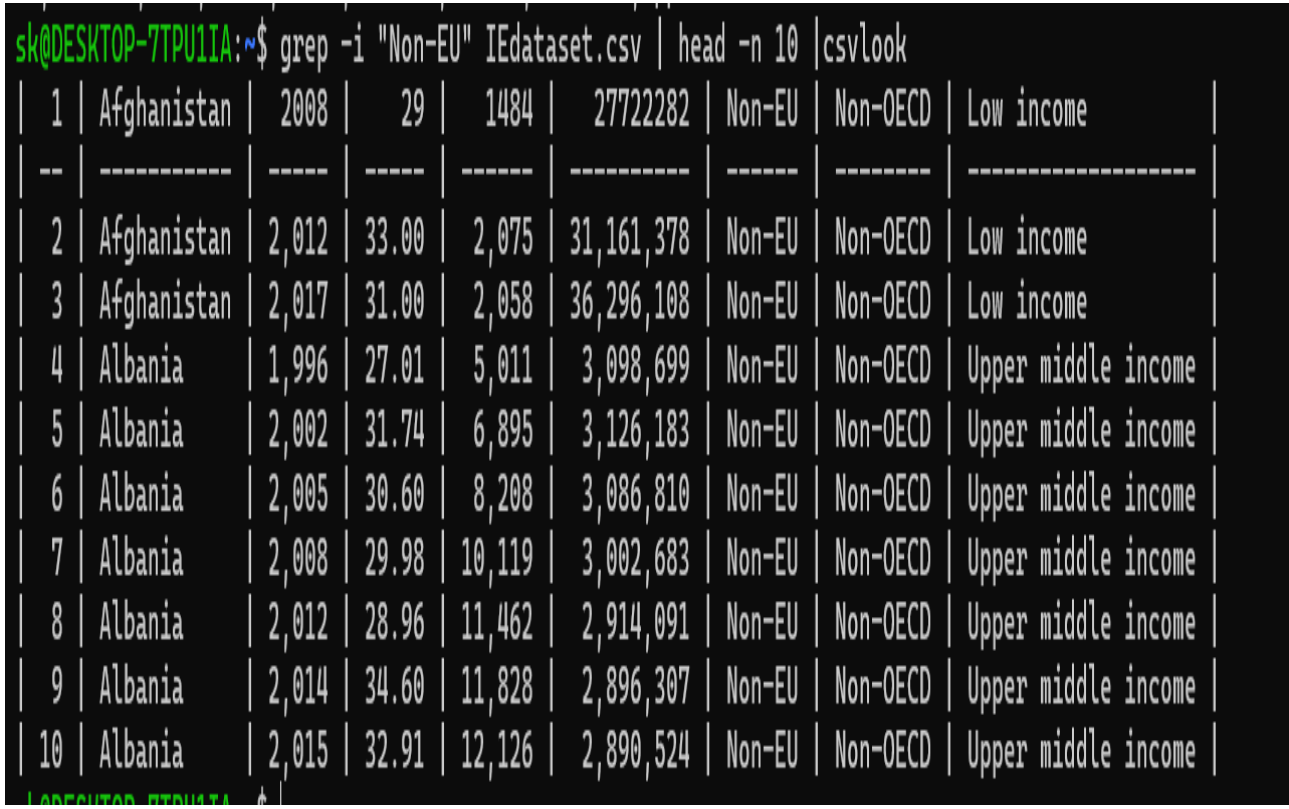
GREP

- The command grep is a small utility for searching plain-text data sets for lines matching a regular expression.
- Its name comes from the Globally search a Regular Expression and Print.
- Grep searches through all the lines but only returns those which matches the pattern phrase.
- To list all the lines in the dataset that contains the word phrase “Non-EU” we will use ‘grep’ as follows:

  $grep -i "Non-EU" IEdataset.csv | head -n 10 |csvlook

- -ignore-case: hyphen i allows us to ignore case sensitive distinctions in patterns and input data, so that characters that differ only in case match each other.

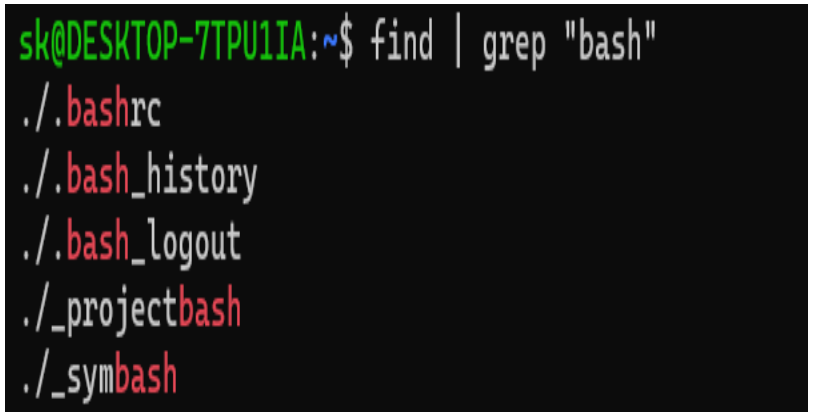
Syntax



Piping find + grep

- Find command find find is a command-line utility that searches one or more directory trees of a file system, locates files based on some user-specified criteria.
- The following example will print out the files that find returns that contain the text “bash":

$ find | grep “bash“



Text processing and manipulation - SED

SED

- SED - **Stream Editor Stream EDitor (SED)** is an important text-processing utilities on GNU/Linux.
- It uses a simple programming language and is capable of solving complex text processing tasks with few lines of code.
- SED can be used in many different ways, such as:

• Text substitution,

• Selective printing of text files,

• In-a-place editing of text files,

• Non-interactive editing of text files, and many more.

Basic syntax

- SED basic syntax:
- ‘s/regular-expression/replacement text/{flags}’
- In the example below, we have used g as a flag, which means “replace all matches” (global replacement):

```
sk@DESKTOP-7TPU1IA:~$ touch datafile.txt | echo "Brussels is a big city!" > datafile.txt
sk@DESKTOP-7TPU1IA:~$ cat datafile.txt
Brussels is a big city!
sk@DESKTOP-7TPU1IA:~$ sed -e 's/big/small/g' datafile.txt
Brussels is a small city!
sk@DESKTOP-7TPU1IA:~$
```

Awk!

What is awk?

- Awk is a record processing tool written by Aho, Kernighan, and Weinberger in 1977. Its name is an acronym of their name
- Awk is part of the Portable Operating System Interface (POSIX). This means it's already on your MacBook and your Linux server.
- he following command will read the content of the IEdataset.csv file and check the 3rd field value in each line.
- If the value is empty, then it will print an error message with the line number.
- $ awk '{ if ($2 == “") print "field is missing in line " NR }' IEdataset.csv
- P.S. The $2 value here does not really represent empty numerical values in the data but a missing field on the 2nd position of each column. While awk is amazing for summarizing text data , it (amazingly) has poor CSV support.

Syntax and finding missing fields with awk

Awk creates a variable for each field (column) in a record (line) ($1, $2 … $NF). $0 refers to the whole record. And $NF refers to the last field of the column.
You can print out fields like this:

```
sk@DESKTOP-7TPU1IA:~$  echo "one two three" | awk '{ print $1 }'
one
sk@DESKTOP-7TPU1IA:~$  echo "one two three" | awk '{ print $2 }'
two
sk@DESKTOP-7TPU1IA:~$  echo "one two three" | awk '{ print $3 }'
three
```

```
sk@DESKTOP-7TPU1IA:~$ awk '{ if ($2 == "") print "Field is missing in line " NR }' IEdataset.csv
Field is missing in line 1
```

Data analysis: Finding percentage % of non-EU countries in the data

- A wc command is used for counting number of words in a given input file. However, it is often used with hyphen –l to count the number of lines in a given input.
- First, we will find out total number of lines in our dataset.
- After that we will find out total number of rows belonging to Non-EU and low-income countries by using the grep command.
- The percentage is calculated using bc command.
- bc command- It is an arbitrary precision calculator language and bc-l, defines the standard math library.
- As we see from our output - we have only 1.29 % of the data from Non-EU and poor countries.
- Please note that grep command does not pick up unique entries in the data, therefore please use this example as a representative for non-precision calculation only.

```
sk@DESKTOP-7TPU1IA:~$ wc -l IEdataset.csv
20676 IEdataset.csv
sk@DESKTOP-7TPU1IA:~$ grep -i "Non-EU" IEdataset.csv | grep -i "low income" IEdataset.csv | wc -l
267
sk@DESKTOP-7TPU1IA:~$ bc -l <<< "267/20676 *100"
1.29135229251305861800
```

"<<<" triple braces are a bit advanced feature. Alternatively, you may want to use echo pipe it to bc command.

```
sk@DESKTOP-7TPU1IA:~$ echo  "267/20676 *100"|bc -l
1.29135229251305861800
```

Using csvstat- statistics without code

One-liner trick:

- Sorting the data in command line may seem inconvenient as the data complexity increases.
- However, there are powerful one-liners that let you work around data without the need to open another application.
- Syntax:
- $csvstat <filename.csv>

```
sk@DESKTOP-7TPU1IA:~$ csvstat IEdataset.csv
  1. "id"

        Type of data:          Number
        Contains null values:  False
        Unique values:         20675
        Smallest value:        1
        Largest value:         21253
        Sum:                   219721036
        Mean:                  10627.377799
        Median:                10638
        StDev:                 6152.187589
        Most common values:    1 (1x)
                               2 (1x)

  5. "gini"

        Type of data:          Number
        Contains null values:  True (excluded from calculations)
        Unique values:         3399
        Smallest value:        12.1
        Largest value:         78.6
        Sum:                   767701.63
        Mean:                  37.281548
        Median:                35.57
        StDev:                 9.416913
        Most common values:    None (83x)
                               33 (77x)
                               35 (72x)
                               29 (70x)
                               34 (67x)

  6. "gdp"

        Type of data:          Number
        Contains null values:  True (excluded from calculations)
        Unique values:         3505
        Smallest value:        385
        Largest value:         191637
        Sum:                   578543446
        Mean:                  28111.926433
        Median:                26413
        StDev:                 19478.432493
        Most common values:    None (95x)
```

Forward thoughts

Known limitations

- This project can be used as an example of a text/data processing pipeline using regular unix command line tools.
- It should be noted that there are limitations to the command line tools that can properly handle CSV data (such as handling multi-character delimiters).
- However, sometimes command line tools can outperform much more popular tools like python (in terms of speed).

Further tools and readings-

- *CSVquote - Smart and simple CSV processing on the command line*
- XSV - xsv is a command line program for indexing, slicing, analyzing, splitting and joining CSV files.