zacheller@home:~/blog$

posts about whoami contact

./missing-semester - Data Wrangling - Exercises

22 Oct 2020

Course located at: missing.csail.mit.edu

Exercises

1. Take this short interactive regex tutorial.

2. Find the number of words (in /usr/share/dict/words) that contain at least three a’s and don’t have a ’s ending. What are the three most common last two letters of those words? sed’s y command, or the tr program, may help you with case insensitivity. How many of those two-letter combinations are there? And for a challenge: which combinations do not occur?

```
$ sudo apt install wamerican-small

# Find the number of words that contain at least three `a`s and don’ t have
$ cat /usr/share/dict/words | grep -E ″.*a.*a.*a.*[^’s]$″ | wc -l
117

# What are the three most common last two letters of those words?
$ cat /usr/share/dict/words | tr ″[:upper:]″ ″[:lower:]″ | grep -E ″(a.*){3,
al,ly,on

# How many two-letter combinations are there?
$ cat /usr/share/dict/words | tr ″[:upper:]″ ″[:lower:]″ | grep -E ″(a.*){3,
31

# And for a challenge: which combinations do not occur?
$ cat /usr/share/dict/words | tr ″[:upper:]″ ″[:lower:]″ | grep -E ″(a.*){3,
$ printf ″%s\n″ {a..z}{a..z} > all_two-letter
$ comm occured all_two-letter -3 | awk '{print $1}' | paste -sd,
aa,ab,ad,af,ag,ah,ai,aj,ak,ao,ap,aq,as,at,au,av,aw,ax,az,ba,bb,bc,bd,be,bf,b
```

3. To do in-place substitution it is quite tempting to do something like `sed s/REGEX/SUBSTITUTION/ input.txt > input.txt`. However this is a bad idea, why? Is this particular to `sed`? Use `man sed` to find out how to accomplish this.

   The processes in a pipeline are all started up in parallel and will truncate the input file before the process at the head of the pipeline has finished. This is not particular to `sed`. The `-i` option streams the edited content into a new file and then renames it behind the scenes.

4. Find your average, median, and max system boot time over the last ten boots. Use journalctl on Linux and log show on macOS, and look for log timestamps near the beginning and end of each boot.

```
$ sudo apt install r-base

$ journalctl | grep -e "userspace" | head -n 10 | sed -E 's/^.*= (.*)s\./\1/
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  11.44   12.64   13.26   18.11   15.25   59.60
```

5. Look for boot messages that are not shared between your past three reboots.

```
$ touch uniq_messages
$ journalctl -b | tail -n +2 | sed -E 's/^.*kali (.*)$/\1/' | sort | uniq |
$ journalctl -b -1 | tail -n +2 | sed -E 's/^.*kali (.*)$/\1/' | sort | uniq
$ journalctl -b -2 | tail -n +2 | sed -E 's/^.*kali (.*)$/\1/' | sort | uniq

# Shared
$ cat uniq_messages | sort | uniq -c | awk '{print $1}' | grep 3 | wc -l
658
# Not Shared
$ cat uniq_messages | sort | uniq -c | awk '{print $1}' | grep -v 3 | wc -l
7408

# All 7408 unshared lines
$ cat uniq_messages | sort | uniq -c | sort -n |awk '{$1=$1};1'| sed -nE 's/
```

6. Find an online data set. Fetch it using curl and extract out just two columns of numerical data. If you're fetching HTML data, pup might be helpful. For JSON data, try jq. Find the min and max of one column in a single command, and the sum of the difference between the two columns in another.

```
# India Historical Population Since 1960
$ wget -O india_historical_pop http://api.worldbank.org/v2/countries/IND/ind

# Min and Max for Population and Years
$ a=($(cat india_historical_pop | xq . | jq 'map(.["wb:data"][]["wb:value"])
min=450547679, max=1366417754
$ a=($(cat india_historical_pop | xq . | jq 'map(.["wb:data"][]["wb:date"])'
[1]}"
min=1960, max=2020

# Sum each column and find the difference
$ cat india_historical_pop | xq . | jq 'map(.["wb:data"][]["wb:value"])' | g
52835203044
$ cat india_historical_pop | xq . | jq 'map(.["wb:data"][]["wb:date"])' | gr
0-9]* | paste -sd+ | bc -l
121390
$ a=$(cat india_historical_pop | xq . | jq 'map(.["wb:data"][]["wb:value"])'
52835081654
```

Zach Heller - 2021