Katharina Gschwind, Sonja Lindberg
gschwind, sonj
Link to repo:
Commit hash: `git log | head -n 1`

Q1
Wrangler script saved to github repo: see wrangler-synsets.py

Katharina Gschwind, Sonja Lindberg
gschwind, sonj
Link to repo:
Commit hash: `git log | head -n 1`

Q2

unique words: 118294

Since our command will include our header “Term” which is not in our dictionary.

To count the number of unique words:

$ cat synsets-clean.txt | awk -F"\",\"" '{print substr($1, 2)}' | sort | uniq | wc -l

Katharina Gschwind, Sonja Lindberg
gschwind, sonj
Link to repo:
Commit hash: `git log | head -n 1`

Q3
How often each country has won the world cup, all those not listed have not won it.

BRA 5

ITA 4

GER 3

ARG 2

URU 2

FRA 1

ENG 1

ESP 1

The Wrangler Transforms Used: wrangler-worldcup.py

To clean the dataset I first changed wrangler's default command to split data on newlines into rows to have it split rows on '-' instead. This allowed me to group a country's data together. I then split on '|', dropped the column of data containing total times the country has placed. I changed the headers of each column to represent the 'place' of the years listed within it for each country. Then I folded columns '1', '2', '3', and '4' on their headers. This got us rows in the format of country|place|years country placed in place. From here, we split our 'years' column into multiple rows, one for each year, and deleted rows where '0' was listed as the year as in the country never placed with that number.

Code to count # wins: python script, run as a python file. Using a special worldcup.csv file because datawrangler kept saving my data with extra new line characters, making it unusable.

```
from collections import Counter
import pandas as pd
import csv
with open('worldcup.csv') as worldcup:
        readWorldcup = csv.reader(worldcup, delimiter = ",")
        # print(lines)

        country_wins = {}
        for row in readWorldcup:
                if row[1] == "1" and row[2]!= “0”:
                        if row[0] in country_wins.keys():
                                country_wins[row[0]] += 1
                        else:
```

Katharina Gschwind, Sonja Lindberg
gschwind, sonj
Link to repo:
Commit hash: `git log | head -n 1`

```
                country_wins[row[0]] = 1

    df = pd.DataFrame.from_dict(country_wins, orient='index')
    print(df)
```

Q4

Clean synsets.txt from the command line

In this command, we read in all the data from the synsets.txt file, then we split our Terms column's fields into multiple rows with their appropriate definitions on the spaces within the Terms fields. Then we loop through the resulting text and split our definitions on ';', and add new rows for each of the definitions with their appropriate and now expanded Terms fields. We finally pipe everything into a text file.

```
$ cat data/synsets.txt  | awk -F',' '{$1 = ""; \
split($2,all_words, " ") \
;\
for (word in all_words) \
{print all_words[word]","$3}; \
}' | awk -F',' '{split($2, all_defs, ";"); for(def in all_defs){ print $1","all_defs[def]}}' >
synsets_clean_awk.txt
```

Clean worldcup.txt from the command line

What we do here is first read in the semi clean worldcup data, remove all newlines, then resplit the data into lines on occurences of '|-', which occur at the end of a country's set of information. That way, we have one row per country, with fields delimited by '|' representing the years in which a country finished first, second, third or fourth. With 0 values in these fields representing no wins of this kind. From here, we loop through all fields of every row, and construct our final rows of country, year, place if the year value is not zero. Finally we write everything to an output file

```
$ cat data/worldcup-semiclean.txt | sed -n -e 'H;${x;s/\n//g;s/^,//;p;}' | sed -e 's/|-/\

/g' | awk -F'|' '{if ($2 != "0"){split($2, fp_yrs, ","); for (i in fp_yrs){print $1","fp_yrs[i]",1"}}; if ($3!=

"0"){split($3, sp_yrs, ","); for (i in sp_yrs){print $1","sp_yrs[i]",2"}}; if ($4 != "0"){split($4, tp_yrs,

","); for (i in tp_yrs){print $1","tp_yrs[i]",3"}}; if ($5 != "0"){split($5, fthp_yrs, ","); for (i in

fthp_yrs){print $1","fthp_yrs[i]",4"}} }' > worldcup-clean-sed.txt
```

Katharina Gschwind, Sonja Lindberg
gschwind, sonj
Link to repo:
Commit hash: `git log | head -n 1`

Katharina Gschwind, Sonja Lindberg
gschwind, sonj
Link to repo:
Commit hash: `git log | head -n 1`

Q5

Using data wrangler is a more user-friendly way to clean our datasets, since the gui makes it more abstract and human-legible. It's more accessible to the general public. On the flip side, however, the command line tools give you more fine grained control over data manipulation, and are faster to use once you've gotten to know them. The payoff is in the overhead in time it takes to learn how to use them. Sed and awk let you do a lot of powerful stuff very quickly that data wrangler would need multiple commands to accomplish.

Katharina Gschwind, Sonja Lindberg
gschwind, sonj
Link to repo:
Commit hash: `git log | head -n 1`

Q6

WMBR cleaning

$ cat data/wmbr.txt |sed -n '/Date:.*/ {N;N;N;N;N;N; s/\n//g; p;}' | sed 's/Date://g; s/Artist:/|/g; s/Song:/|/g; s/Album:/|/g; s/Label:/|/g; s/Show:/|/g; s/DJ:/|/g' > wmbr_cleaned_awk.txt

Lizzo cleaning

Using wrangler, named wrangler-lizzo.py in my github repo.

Top2018.csv already cleaned, no need to clean.

Katharina Gschwind, Sonja Lindberg
gschwind, sonj
Link to repo:
Commit hash: `git log | head -n 1`

Q7

Code

```
$ cat wmbr_cleaned_awk.txt | awk -F"|" '{if (index(tolower($3), "live") != 0 || index(tolower($4), "live") != 0) {print $2}}' | sort | uniq | sort --ignore-case
```

Artists:

amiina

Brendan Little

Gary War

lindefelt

Peter Galperin

Thingy

White Hills

Katharina Gschwind, Sonja Lindberg
gschwind, sonj
Link to repo:
Commit hash: `git log | head -n 1`

Q8

$ cat wmbr_cleaned_awk.txt | awk -F "|" '$4 ~ /Stranger Things/ {print $7}'|sort | uniq -c | sort -n -r

#times DJ
3 DJ Lipika
2 Sara Achour
2 L-Train
2 Brian Sennett
1 TJ Connelly
1 Lisa

Katharina Gschwind, Sonja Lindberg
gschwind, sonj
Link to repo:
Commit hash: `git log | head -n 1`

Q9

```
$ cat wmbr_cleaned_awk.txt | awk -F "|" '$1 ~ /2018/ || $1 ~ /2017/ || $1 ~ /2019/ {print $1","$2}'| awk -F',' '{$1= ""; print $0}' | sed 's/Billie Ellish/Billie Eilish/g' | sed 's/billie eilish/Billie Eilish/g' | sort -n | uniq -c |awk '{all_but_first_two = ""; for (i=3; i<= NF; i++){all_but_first_two = all_but_first_two" "$i}; if( all_but_first_two !~ /Billie/){ all_but_first_two = "Other"; $3 = all_but_first_two; for (i = 4; i <= NF; i++){$i = ""}} print $0}' | awk '{ $2 = $2$3; $3 = ""; $4= ""; print $0}' | awk '{arr[$2]+=$1} END {for (i in arr) {print i,arr[i]}}' | awk '{print $0}' | sort | awk '{if ($1 ~ /Billie/) { getline counterpart; $1 = substr($1, 1, 4); split(counterpart, cparr, " "); print $1" "$2/(cparr[2] + $2)}}' | sort -n -r
```

Year Ratio
2019 0.189655
2018 0.22449
2017 0.266667

Katharina Gschwind, Sonja Lindberg
gschwind, sonj
Link to repo:
Commit hash: `git log | head -n 1`

Q10

To get the years Lizzo was on talk shows:

```
$ cat lizzo_cleaned.txt | awk -F"\",\"" '{if($2 ~ /Show/) {print substr($1,2)}}' | uniq
```

2014

2015

2019

Knowing our three applicable years, and seeing that there are few enough that manually coding for them is fastest, we find our songs:

```
$ cat wmbr_cleaned_awk.txt | awk -F"|" '{ if (($1 ~ /2014/ || $1 ~ /2015/ || $1 ~ /2019/) && $2 ~
/Lizzo/) {print $3}}' |  sort | uniq -c | sort -k1,1r -k2,2
```

```
  9 Juice
  3 Soulmate
  3 Tempo (feat. Missy Elliott)
  2 Boys
  2 Crybaby
  2 Cuz I Love You
  2 Good As Hell
  2 Humanize
  2 Lingerie
  2 Water Me
  1 Fitness
  1 Jerome
  1 Juice - Breakbot Mix
  1 Like A Girl
  1 Torn Apart, Pt. II (Bastille vs. GRADES vs. Lizzo)
  1 Truth Hurts
  1 Worship
```

Katharina Gschwind, Sonja Lindberg
gschwind, sonj
Link to repo:
Commit hash: `git log | head -n 1`

Q11

Walk It Talk It by Migos

Was the most danceable track per our criteria.

Open python3 in the terminal in the container and run the following code.

```
import pandas as pd

wmbr = pd.read_csv("wmbr_cleaned_awk.txt", delimiter="|")
top18 = pd.read_csv("data/top2018.csv")
set_combos = pd.unique(wmbr["Artist"])

for i in range(len(set_combos)):
        set_combos[i] = set_combos[i].replace(" and ", ", ")
        set_combos[i] = set_combos[i].replace(" & ", ", ")

set_combos = pd.unique(set_combos)

artists = []
for combo in set_combos:
        tmp = combo.split(", ")
        for val in tmp:
                if val not in artists:
                        artists.append(val)

d_artists = {'Artist':artists}
df = pd.DataFrame(data=d_artists)

joined = top18.merge(right=df,how='inner', on="artists")

joined = joined[["artists", "danceability", "name"]]
joined.sort_values(by="danceability", ascending=False)[:1][["artists", "name"]]
```

Katharina Gschwind, Sonja Lindberg
gschwind, sonj
Link to repo:
Commit hash: `git log | head -n 1`

Q12

I didn't have enough knowledge of UNIX tools to get started with this lab, such that I ended up spending ~10 hours on this pset and ~3 of those hours on googling and reading about awk and sed. Some more intro material in the pset would have been helpful and would have saved me a lot of looking up. The valuable parts of the lab were getting to choose our own method for approaching questions, this let us figure out what we value vs. already know how to do. The pace of lectures so far is good, but the psets have been taking an extremely long time!