You will be using data from the Observed Antibody Space (OAS) database. You'll need a username/password to view it:

User: opig

Password: khazir

Have a look around - I've sent you the OAS paper which will contain more details. Once you've seen enough, you can start playing with some data! Go to the following page and download the 'data unit':

http://antibodymap.org/dataunit?unit=Vander_Heiden_2017/Vander_Heiden_2017_Heavy_HD09_IGHG_HD09_Unsorted_Bcells_age31_healthy_iglblastn_igblastn_IGHG.json.gz

This file contains the details of 14460 antibody heavy chain sequences. You should be able to just open the file with a text editor and take a look - in a terminal, navigate to wherever you saved the file, and type:

```
vim Vander_Heiden_2017_Heavy_HD09_IGHG_HD09_Unsorted_Bcells_age31_healthy_iglblastn_igblastn_IGHG.json.gz
```

The first line of the file contains a dictionary of metadata, i.e. details about this particular set of sequences. Each remaining line contains a dictionary of sequence data. More information about exactly what's inside can be found here:

http://antibodymap.org/documentation

To close vim, type '`:q`' and press enter. Now, to play with the data in Python, type the command '`ipython`' into the terminal and press enter. You should now find yourself with a Python shell, into which you can type commands etc. interactively. I've quickly written a bit of code that should read in the data for you (make sure you're in the same directory as your file):

```
import gzip
import json

oas_file = "Vander_Heiden_2017_Heavy_HD09_IGHG_HD09_Unsorted_Bcells_age31_healthy_iglblastn_igblastn_IGHG.json.gz"

meta_line = True
sequence_data = []
for line in gzip.open(oas_file,'rb'):
    if meta_line == True:
        metadata = json.loads(line)
        meta_line=False
        continue

    #Parse actual sequence data.
    basic_data = json.loads(line)
    sequence_data.append(basic_data)

    #IMGT-Numbered sequence.
    d = json.loads(basic_data['data'])
    sequence_data[-1]['data'] = d
```

You should now have a dictionary called 'metadata', and a list of dictionaries called 'sequence_data'. Again, have a play around and figure out what is stored where. Also since Python is quite new to you, it may be worth going through the above code and working out what's going on! There's more information about parsing OAS data here which may help:

http://www.blopig.com/blog/2018/06/how-to-parse-oas-data/

Now we can start playing with the data properly! To begin with, have a go at the following:

- There is one dictionary per unique sequence in 'sequence_data'. How many different sequences are there in this data unit?
- On the OAS website it said there were 14460 redundant sequences ('redundant' means counting duplicates). Verify this is true. (Hint: each sequence has a 'redundancy' value that states how many times it occurred in this data)
- How many of the sequences are thought to contain errors? Count how many have zero errors, 1 error, 2 errors, etc.
- Which CDR H3 sequence is the most common?
- More challenging but related to what we'll be doing - the numbering for a sequence is given in its dictionary (i.e. in 'sequence_data[0]['data']'). This is split into regions; the four framework regions ('fwh1' etc.) and the CDR regions ('cdrh1' etc.). The ordering of these regions along the sequence is fwh1-cdrh1-fwh2-cdrh2-fwh3-cdrh3-fwh4. fwh1 always contains positions 1-26, cdrh1 is always 27-38, and so on. For each position, calculate the amino acid usage as a percentage. For example, for position 1, glutamine (Q) occurs 2949 times, glutamic acid (E) occurs 1960 times, aspartic acid (D) occurs 51 times and histidine (H) occurs 30 times, making their percentage usages 53.6%, 35.6%, 0.9% and 0.5% respectively (position 1 does not appear at all for 511 sequences). See if you can write out the 'consensus' sequence, i.e. the most common amino acid for each position. To begin with, just consider each unique sequence once, then see if you can re-calculate this taking into account how many times each sequence was seen in the dataset.