Big Data Exercise - due 10/21 *by Charlotte*

Exercise 1

The data we'll be working with can be found in the file ghcnd_daily.tar.gz. It includes daily weather data from thousands of weather stations around the work over many decades.

Begin by unzipping the file and checking it's size – it should come out to be about 4gb, but will expand to about 12 gb in RAM, which means there's just no way most students (who usually have, at most, 16gb of RAM) can import this dataset into pandas and manipulate it directly.

(Note: what we're doing can be applied to much bigger datasets, but they sometimes takes hours to work with, so we're working with data that's just a little big so we can get exercises done in reasonable time).

- The info says 3.97 GB on disk

Exercise 2

To pick your stations, we'll need to open the ghcnd-stations.txt file in the directory you've downloaded. It includes both station codes (which is what we'll find in the ghcnd_daily.csv data, as well as the name and location of each station).

Stations picked:

- GME00111445 # Berlin Tempelhof
- GME00130534 # Worms, hometown

Reading in the stations file:

In [ ]:

```
import pandas as pd
headings = ["ID", "Latitude", "Longitude", "Element", "first_year", "this_y
colspecs = [(1, 11), (13, 20), (22, 30), (32, 35), (37, 40), (42, 45)]
info = pd.read_fwf("ghcnd-stations.txt", colspecs=colspecs, header=None, na
```

In [ ]:

```
info.sample(20)
```

Out[ ]:

	ID	Latitude	Longitude	Element	first_year	this_year
75624	S1NYER0123	43.0455	-78.7215	177	NY	LAR
58540	S1COKW0014	38.5437	102.1624	221	CO	OWN
20370	R00B5-0020	20.5700	-48.5700	520	NaN	ARR
64068	S1ILMCH013	42.3245	-88.3946	292	IL	ULL
71494	S1NCBC0078	35.6856	-82.5173	704	NC	EAV
82652	S1TXHRR018	29.7276	-95.3846	14	TX	OUS
80538	S1TNHR0002	35.4122	-88.8899	129	TN	EDO
23772	A001013051	48.8667	123.5000	45	BC	ANG
92745	SC00137738	42.2333	-96.2333	326	IA	LOA
11892	SN00061362	33.2422	151.4714	30	NaN	ARN
92091	SC00121869	38.8725	-86.8350	222	IN	RAN
16682	SN00091283	41.2292	147.1225	105	NaN	ARO
17576	F1BI000001	25.7371	-79.2908	7	BH	LIC
13345	SN00070244	35.3833	149.0833	670	NaN	ORR
114525	A005688170	24.6200	17.9700	110	NaN	ARI
21620	R00F4-0250	24.2800	-47.9500	30	NaN	IBE
56438	S1CASK0011	41.2988	122.3031	95	CA	OUN
65614	S1INTP0029	40.4483	-87.0048	195	IN	EST
24730	A00110EF57	49.3167	123.0500	111	BC	VA
96848	SC00238585	36.9333	-92.2667	999	MO	ANZ

Exercise 3

Now that we something about the observations we want to work with, we can now turn to our actual weather data. Let's start with the fun part. SAVE YOUR NOTEBOOK AND ANY OTHER OPEN FILES!. Then just try and import the data (ghcnd_daily.csv) while watching your Activity Monitor (Mac) or Resource Monitor (Windows) to see what happens.

In [ ]:

```
# wetter = pd.read_csv("ghcnd_daily.csv")
print("Killed kernel, didn't manage.")
```

```
Killed kernel, didn't manage.
```

Exercise 4

Now that we know that we can't work with this directly, it's good with these big datasets to just import ~200 lines so you can get a feel for the data. So load just 200 lines of ghcnd_daily.csv.

Reading in the first 200 rows:

In [ ]:
```
wetter_200 = pd.read_csv("ghcnd_daily.csv", nrows=200, dtype=object)
wetter_200.set_index("id")
```

Out[ ]:

	year	month	element	value1	mflag1	qflag1	sflag1	value2	mflag2	qflag
id										
ACW00011604	1949	1	TMAX	289	NaN	NaN	X	289	NaN	Na
ACW00011604	1949	2	TMAX	267	NaN	NaN	X	278	NaN	Na
ACW00011604	1949	3	TMAX	272	NaN	NaN	X	289	NaN	Na
ACW00011604	1949	4	TMAX	278	NaN	NaN	X	283	NaN	Na
ACW00011604	1949	5	TMAX	283	NaN	NaN	X	283	NaN	Na
...	...	...	...	...	...	...	...	...	...	
AE000041196	1981	9	TMAX	-9999	NaN	NaN	NaN	-9999	NaN	Na
AE000041196	1981	10	TMAX	-9999	NaN	NaN	NaN	350	NaN	Na
AE000041196	1981	11	TMAX	330	NaN	NaN	I	310	NaN	Na
AE000041196	1981	12	TMAX	270	NaN	NaN	I	290	NaN	Na
AE000041196	1982	1	TMAX	245	NaN	NaN	I	230	NaN	Na

200 rows × 127 columns

In [ ]:
```
print(wetter_200['id'].isin(['GME00111445', 'GME00130534', 'USC00050848']).
```

```
False
```

As I find my chosen stations and Nick's station are not yet included in that chunk.

Exercise 5

Once you have a sense of the data, write code to chunk your data: i.e. code that reads in all blocks of the data that will fit in ram, keeps only the observations for the weather stations you've selected to focus on, and throws away everything else.

In addition to your own 4 weather stations, please also include station USC00050848 (a weather station from near my home!) so you can generate results that we can all compare (to check for accuracy).

In [ ]:

```
chunkies = []
chosen_stations = ['GME00111445', 'GME00130534', 'USC00050848']
for df in pd.read_csv('ghcnd_daily.csv', iterator=True, chunksize=2000000,
    if df['id'].isin(['GME00111445', 'GME00130534', 'USC00050848']).any() =
        chunkies.append(df)
```

In [ ]:

```
print("Combine all chunks where mine and Nick's weather stations are incluc
wetter_all = pd.concat(chunkies)
```

```
Combine all chunks where mine and Nick's weather stations are included
```

Exercise 6

Now, for each weather station, figure out the earliest year with data. Keep USC00050848 and the one weather station for each member of your team with the best data (i.e. each member of your pair should have picked two weather stations: keep the one from each pair with the best data).

In [ ]:

```
print("Subsetting the dataset for the relevant stations")
wetter_mine = wetter_all.loc[(wetter_all['id'] == "GME00111445") | (wetter_
```

```
Subsetting the dataset for the relevant stations
```

In [ ]:

```
wetter_mine
```

Out[ ]:

	id	year	month	element	value1	mflag1	qflag1	sflag1	value2	mfl
2971391	GME00111445	1931	1	TMAX	NaN	NaN	NaN	NaN	50.0	I
2971392	GME00111445	1931	2	TMAX	11.0	NaN	NaN	S	-11.0	I
2971393	GME00111445	1931	3	TMAX	28.0	NaN	NaN	S	22.0	I
2971394	GME00111445	1931	4	TMAX	78.0	NaN	NaN	S	61.0	I
2971395	GME00111445	1931	5	TMAX	139.0	NaN	NaN	S	178.0	I
...	...	...	...	...	...	...	...	...	...	
7987013	USC00050848	2019	5	TMAX	72.0	NaN	NaN	7	150.0	I
7987014	USC00050848	2019	6	TMAX	233.0	NaN	NaN	7	244.0	I
7987015	USC00050848	2019	7	TMAX	289.0	NaN	NaN	H	289.0	I
7987016	USC00050848	2019	8	TMAX	283.0	NaN	NaN	H	311.0	I
7987017	USC00050848	2019	9	TMAX	367.0	NaN	NaN	H	378.0	I

3293 rows × 129 columns

Exercise 7

Now calculate the average max temp for each weather station / month in the data.

In [ ]:

```
wetter_mine.filter(like='value')
```

Out[ ]:

	value1	value2	value3	value4	value5	value6	value7	value8	value9	value10
2971391	NaN	50.0	28.0	NaN	39.0	0.0	0.0	11.0	-22.0	0.0
2971392	11.0	-11.0	-11.0	-28.0	-50.0	-39.0	-61.0	NaN	-22.0	39.0
2971393	28.0	22.0	39.0	11.0	22.0	-22.0	-11.0	NaN	-22.0	-11.0
2971394	78.0	61.0	89.0	150.0	111.0	100.0	100.0	89.0	122.0	NaN
2971395	139.0	178.0	222.0	200.0	189.0	222.0	250.0	128.0	122.0	128.0
...	...	...	...	...	...	...	...	...	...	...
7987013	72.0	150.0	200.0	211.0	239.0	217.0	161.0	94.0	39.0	111.0
7987014	233.0	244.0	250.0	272.0	250.0	278.0	289.0	272.0	200.0	250.0
7987015	289.0	289.0	317.0	294.0	267.0	267.0	300.0	289.0	317.0	300.0
7987016	283.0	311.0	356.0	339.0	NaN	350.0	306.0	NaN	356.0	350.0
7987017	367.0	378.0	333.0	294.0	NaN	278.0	372.0	294.0	289.0	278.0

3293 rows × 31 columns

Change missing values coded as -9999 to NaN so that they don't skew the means and calculate the means per month:

In [ ]:
```
import numpy as np
import pandas as pd
wetter_mine = wetter_mine.replace(-9999, np.nan)
wetter_mine["mean_per_month"] = wetter_mine.filter(like='value').mean(axis=
```

In [ ]:
```
wetter_mine["mean_per_month"] = pd.to_numeric(wetter_mine["mean_per_month"]
```

Exercise 8

Now for each weather station, generate a separate plot of the daily temperatures over time. Subsetting the datsets for the specific stations and cutting them down to the respective values:

In [ ]:
```
wetter_Berlin = wetter_mine.loc[wetter_mine['id'] == "GME00111445"]
wetter_Worms = wetter_mine.loc[wetter_mine['id'] == "GME00130534"]
wetter_Nick = wetter_mine.loc[wetter_mine['id'] == "USC00050848"]
```

In [ ]:
```
wetter_Berlin = wetter_Berlin.loc[:, (wetter_Berlin.columns == 'year') | (w
wetter_Worms = wetter_Worms.loc[:, (wetter_Worms.columns == 'year') | (wett
wetter_Nick = wetter_Nick.loc[:, (wetter_Nick.columns == 'year') | (wetter_
```

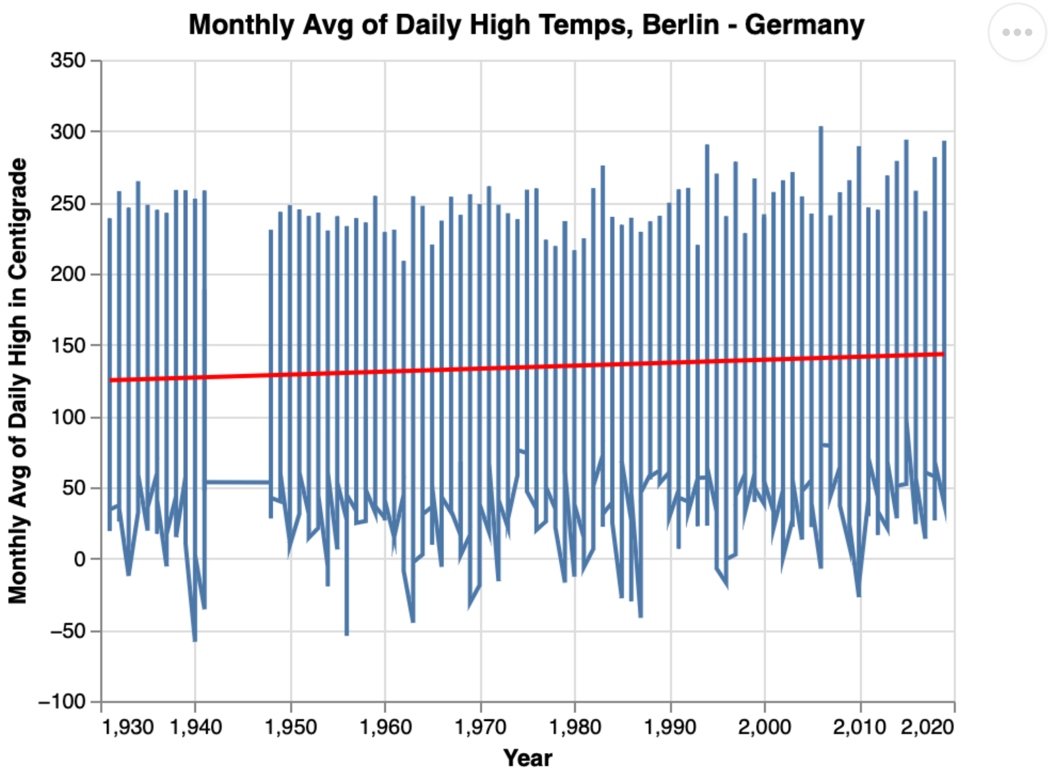
Graphing Berlin Chart:

In [ ]:
```
import altair as alt
base = (alt.Chart(wetter_Berlin, title="Monthly Avg of Daily High Temps, Be
    x=alt.X("year", title="Year", scale=alt.Scale(zero=False)),
    y=alt.Y("mean_per_month", title="Monthly Avg of Daily High in Centigrad
    )
)

fit = base.transform_regression(
        "year",    "mean_per_month",
    ).mark_line(color="red")


base + fit
```

Out[ ]:



What is up between 1940 and 1950 (Second World War II)? Checking whether I have data available:

In [ ]:

```
wetter_Berlin.loc[(wetter_Berlin['year'] < 1950) | wetter_Berlin['year'] >
```

Out[ ]:

year	mean_per_month

No data available, which though makes sense, seeing as it was the Second World War and Germany most likely lost a lot of its stations.

Graphing Worms chart:

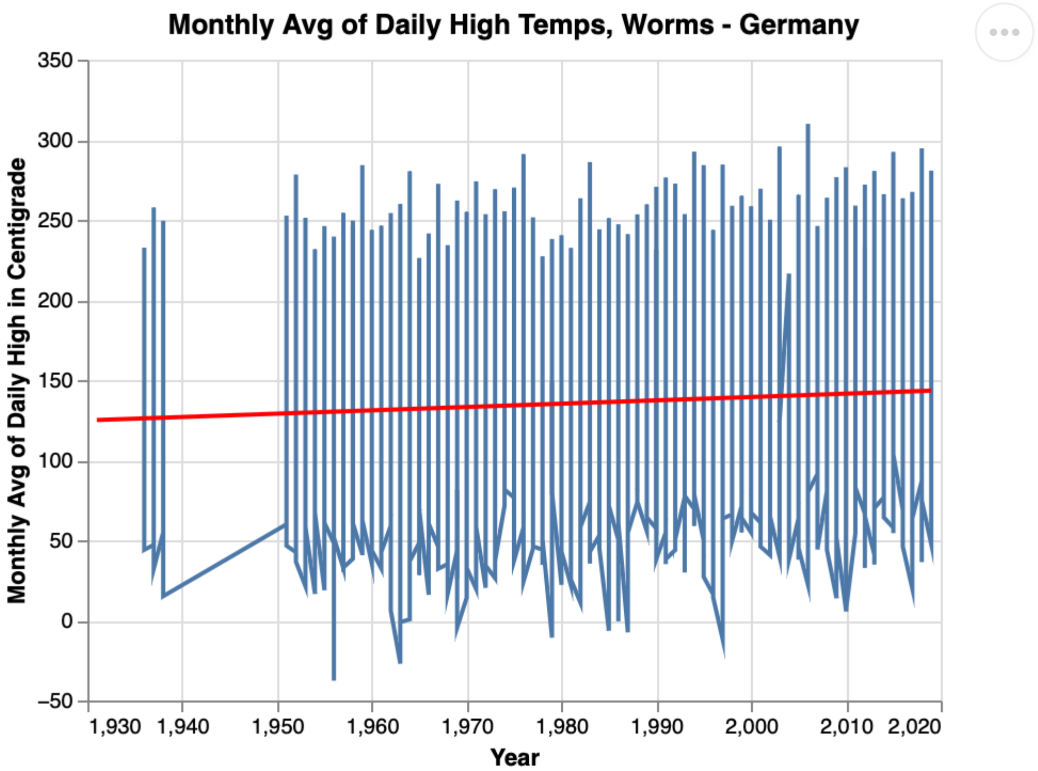
In [ ]:

```
base1 = (alt.Chart(wetter_Worms, title="Monthly Avg of Daily High Temps, Wo
    x=alt.X("year", title="Year", scale=alt.Scale(zero=False)),
    y=alt.Y("mean_per_month", title="Monthly Avg of Daily High in Centigrad
    )
)

fit1 = base.transform_regression(
        "year",    "mean_per_month",
    ).mark_line(color="red")


base1 + fit0
```

Out[ ]:



Same as with Berlin regarding missing data between 1940 and 1950.

Graphing Nick's station's chart:

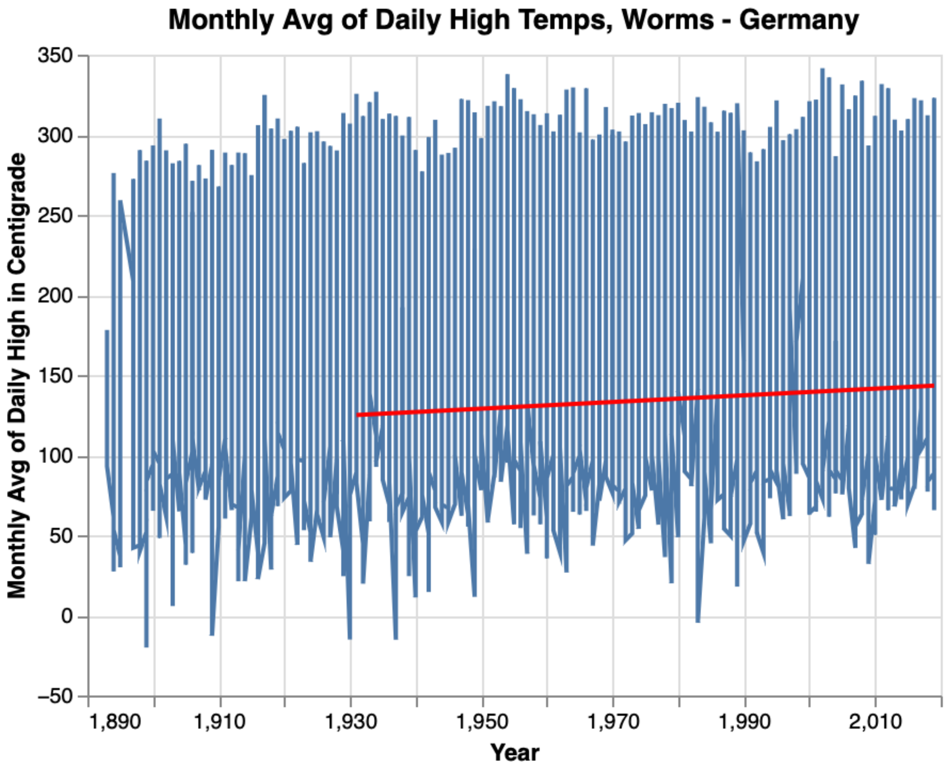
In [ ]:

```
base2 = (alt.Chart(wetter_Nick, title="Monthly Avg of Daily High Temps, Wor
    x=alt.X("year", title="Year", scale=alt.Scale(zero=False)),
    y=alt.Y("mean_per_month", title="Monthly Avg of Daily High in Centigrac
    )
)

fit2 = base.transform_regression(
        "year",    "mean_per_month",
    ).mark_line(color="red")

base2 + fit2
```

Out[ ]:


Monthly Avg of Daily High Temps, Worms - Germany
Monthly Avg of Daily High in Centigrade
350
300
250
200
150
100
50
0
−50
1,890
1,910
1,930
1,950
1,970
1,990
2,010
Year