Unit 4 - Noisy data

In [1]:
```
import pandas as pd
import numpy as np
```

1. Histograms

Gym example

Taken from https://data4ml.com/plot-histogram-python-pandas/

In [2]:
```
# 20 CM Dress
height = 160.0
sample = 20
np.random.seed(0)
height_2 = np.random.normal(mu, sigma, sample).astype(int)
```

In [3]:
```
# 20 CM Pants
sigma = 20.0
sample = 20
np.random.seed(1)
height_3 = np.random.normal(mu, sigma, sample).astype(int)
```

In [4]:
```
gym = pd.DataFrame({'height_2': height_2, 'height_3': height_3})
gym
```

Out[4]:

	height_2	height_3
0	176	185
1	155	172
2	172	172
3	170	169
4	171	181
5	163	162
6	172	166

To begin with - why a histogram?

This is what happens if we just plot lines:

In [5]:
```
gym.plot()
```

In [6]:
```
%matplotlib=
```

(ignore, it's very messy)

Different histograms:

In [7]:
```
gym.hist('height_2')
```

Out[7]:
```
array([[<AxesSubplot:title={'center':'height_2'}>]], dtype=object)
```

In [8]:
```
gym.hist('height_3')
```

Out[8]:
```
array([[<AxesSubplot:title={'center':'height_3'}>]], dtype=object)
```

In [9]:
```
gym.hist(bins=5)
```

Out[9]:
```
array([[<AxesSubplot:title={'center':'height_2'}>,
        <AxesSubplot:title={'center':'height_3'}>]], dtype=object)
```

In [10]:
```
gym.hist(bins=5, sharey=True)
```

Out[10]:
```
array([[<AxesSubplot:title={'center':'height_2'}>,
        <AxesSubplot:title={'center':'height_3'}>]], dtype=object)
```

In [11]:
```
gym.hist(bins=5, sharey=True, legend=True)
```

Out[11]:
```
array([[<AxesSubplot:title={'center':'Frequency'}>]]
```

Back to our vaccinations data:

In [12]: `!wget -O ./data/vaccinations.csv https://ourworldindata.org/grapher/covid-19-data-owidcom/public/data/vaccinations/vaccinations.csv`

Out[12]:
```
--2022-01-19 05:24:42--  https://ourworldindata.org/grapher/covid-19-data-owidcom/public/data/vaccinations/vaccinations.csv
Resolving ourworldindata.org (ourworldindata.org)... 104.26.13.205, 104.26.12.205, 172.67.70.149
Connecting to ourworldindata.org (ourworldindata.org)|104.26.13.205|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/plain]
Saving to: ./data/vaccinations.csv

./data/vaccinations.csv     [ <=>                ]  12.57M  --.-KB/s    in 0.1s

2022-01-19 05:24:42 (94.1 MB/s) - ./data/vaccinations.csv saved [13181000]
```

In [13]: `!ls -lh ./data/vaccinations.csv`

Out[13]:
```
-rw-r--r-- 1 root root 13M Jan 19 05:24 ./data/vaccinations.csv
```

In [14]:
```
df = pd.read_csv("./data/vaccinations.csv")
df.head(3)
```

Out[14]:

	location	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	total_boosters	daily_vaccinations_raw	daily_vacc
0	Afghanistan	AFG	2021-02-22	NaN	NaN	NaN	NaN	NaN	NaN
1	Afghanistan	AFG	2021-02-23	NaN	NaN	NaN	NaN	NaN	NaN
2	Afghanistan	AFG	2021-02-24	NaN	NaN	NaN	NaN	NaN	NaN

Look at data for different countries

Use `groupby()` to group according to location

Attempt #1 - what is wrong?

In [15]:
```
df.groupby('location')['total_vaccinations','daily_vaccinations','people_fully_vaccinated','people_fully_
```

Out[15]:

	total_vaccinations	daily_vaccinations	people_fully_vaccinated	people_fully_vaccinated_per_hundred
location				
Afghanistan	5.155227e+06	72098.0	4.207845e+05	1.08
Africa	1.495724e+08	1012754.0	5.792854e+07	4.72
Albania	1.083944e+06	1719.0	7.140202e+05	26.25
Algeria	9.989663e+06	25857.0	4.174623e+06	9.36
Andorra	9.343035e+04	176.0	4.161353e+04	54.08
Wallis and Futuna	1.021010e+04	347.0	5.010460e+03	45.21
World	6.137924e+09	43379850.0	2.562763e+09	32.96
Yemen	2.229845e+05	10240.0	1.490806e+05	0.55
Zambia	6.700618e+06	138143.0	2.919407e+05	1.54
Zimbabwe	5.218672e+06	75093.0	2.188626e+06	14.50

234 rows × 4 columns

Attempt #2

Is this better?

In [16]:
```
df.groupby('location')['total_vaccinations','daily_vaccinations','people_fully_vaccinated','p
```

Out[16]:

	total_vaccinations	daily_vaccinations	people_fully_vaccinated	people_fully_vaccinated_per_hundred
location				
Afghanistan	5.155227e+06	72098.0	4.207845e+05	1.08
Africa	1.495724e+08	1012754.0	5.792854e+07	4.72
Albania	1.083944e+06	1719.0	7.140202e+05	26.25
Algeria	9.989663e+06	25857.0	4.174623e+06	9.36
Andorra	9.343035e+04	176.0	4.161353e+04	54.08
Wallis and Futuna	NaN	NaN	NaN	NaN
World	6.137924e+09	43379850.0	2.562763e+09	32.96
Yemen	NaN	NaN	NaN	NaN
Zambia	NaN	NaN	NaN	NaN
Zimbabwe	NaN	NaN	NaN	NaN

234 rows × 4 columns

Attempt #3 - change the 0 from strong to int, and finally, it works :)

In [17]:
```
df.fillna(0).groupby('location')['total_vaccinations','daily_vaccinations','people_fully_vaccinated','p
```

Out[17]:

	total_vaccinations	daily_vaccinations	people_fully_vaccinated	people_fully_vaccinated_per_hundred
location				
Afghanistan	5.155227e+06	72098.0	4.207845e+05	1.08
Africa	1.495724e+08	1012754.0	5.792854e+07	4.72
Albania	1.083944e+06	1719.0	7.140202e+05	26.25
Algeria	9.989663e+06	25857.0	4.174623e+06	9.36
Andorra	9.343035e+04	176.0	4.161353e+04	54.08
Wallis and Futuna	0.000000e+00	0.0	0.000000e+00	0.00
World	6.137924e+09	43379850.0	2.562763e+09	32.96
Yemen	2.229845e+05	10240.0	1.490806e+05	0.55
Zambia	6.700618e+06	138143.0	2.919407e+05	1.54
Zimbabwe	5.218672e+06	75093.0	2.188626e+06	14.50

234 rows × 4 columns

The `world` row shouldn't be there. Remove it using `.drop()` and `inplace`:

In [18]: `df.drop(max(df.location), inplace=True)`

Your turn:

>What do you think `.index` does? Why is it there? How can you find out?

Before we continue, just group this by one to see a dataframe, will be easier

In [19]:
```
grouped_df = df.fillna(0).groupby('location')['total_vaccinations','daily_vaccinations','people_fully_va
grouped_df.head(3)
```

Out[19]:

	total_vaccinations	daily_vaccinations	people_fully_vaccinated	people_fully_vaccinated_per_hundred
location				
Wales	4149020.0	13151.0	2216010.0	69.96
Wallis and Futuna	10210.0	347.0	5010.0	45.21
Yemen	223984.0	10240.0	149080.0	0.55

sort the values using 'sort_values()'

In [20]:
```
grouped_df.sort_values('people_fully_vaccinated_per_hundred', ascending = False).head(10)
```

Out[20]:

	total_vaccinations	daily_vaccinations	people_fully_vaccinated	people_fully_vaccinated_per_hundred
location				
Gibraltar	79010.0	1068.0	39463.0	117.13
Pitcairn	963	1.0	47.0	100.00
Portugal	15096050.0	139618.0	8602244.0	84.60
Cayman Islands	107000.0	1004.0	34800.0	82.34
United Arab Emirates	19843232.0	1312112.0	9140120.0	81.77
Malta	874184.0	7107.0	419919.0	81.61
Ireland	5111553	15463.0	2724812.0	55.23
Spain	44766812.0	379710.0	26337117.0	72.72
Singapore	5236575.0	87359.0	4566229.0	77.44
Israel	4650668.0	37464.0	2379078.0	75.70

Histogram according to all values:

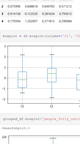
In [21]:
```
grouped_df.hist(bins=5)
```

Out[21]:
```
array([[<AxesSubplot:title={'center':'total_vaccinations'}>,
        <AxesSubplot:title={'center':'daily_vaccinations'}>],
       [<AxesSubplot:title={'center':'people_fully_vaccinated'}>,
        <AxesSubplot:title={'center':'people_fully_vaccinated_per_hundred'}>],
       dtype=object)
```

Histogram according to people_fully_vaccinated_per_hundred

In [22]:
```
grouped_df.hist('people_fully_vaccinated_per_hundred', bins=5)
```

Out[22]:
```
array([[<AxesSubplot:title={'center':'people_fully_vaccinated_per_hundred'}>]],
       dtype=object)
```



Remove rows with 0's

Note that this is different than changing values to 0's

In [23]:
```
grouped_df.drop(grouped_df.grouped_df.people_fully_vaccinated_per_hundred == 0.0).index, inplace=True
grouped_df.hist('people_fully_vaccinated_per_hundred',bins=5)
```

Out[23]:
```
array([[<AxesSubplot:title={'center':'people_fully_vaccinated_per_hundred'}>]],
       dtype=object)
```


Your turn:

Do the same, but for another column

In [24]:

In [25]:

Functions covered in this unit:

`std()` - standard deviation

`.hist(bins,sharey,legend)` - the data and the number of bins

`.plot()` - simple line plot

`.hist(bins,sharey,alpha)` - the data, the number of bins and the transparency (default is 10 bins, all data and not transparent)

`.plot.hist()` - histograms on same chart

`.drop()` - removes unwanted rows or columns

`.index()` - index of the selected rows

`sort_values()` - self explanatory. It just sorts values.

2. Standard deviation

An old example showing that the more the data is spread, the higher the std

In [26]:
```
df = pd.DataFrame({"height": [161, 156, 172],
                   "weight": [ 65, 85, 89],
                   "age": [35,25,33]})
df
```

Out[26]:

	height	weight	age
0	161	65	35
1	156	85	25
2	172	89	33

In [27]:
```
df.mean()
```

Out[27]:
```
height    163.333333
weight     79.666667
age        31.333333
dtype: float64
```

In [28]:
```
df.std()
```

Out[28]:
```
height    8.145251
weight   12.583554
age       5.507571
dtype: float64
```

3. Boxplots

In [29]:
```
np.random.seed(100)
df = pd.DataFrame(np.random.rand(25,4),
                  columns=['C1', 'C2', 'C3', 'C4'])
df.head()
```

Out[29]:

	C1	C2	C3	C4
0	0.912291	0.965753	0.092627	0.733782
1	0.210851	0.749820	0.253460	0.678446
2	0.273446	0.844618	0.240755	0.573123
3	0.914168	0.125228	0.239104	0.795612
4	0.179546	0.200867	0.271915	0.200968

In [30]:
```
df.plot(kind='boxplot')
```

Out[30]:
```
<AxesSubplot:>
```


In [31]:
```
df.plot.hist(bins=5,sharey=True,alpha=0.5)
```

Out[31]:
```
<AxesSubplot:>
```


In [32]:
```
%matplotlib=
```

In [33]:
```
grouped_df.hist('people_fully_vaccinated_per_hundred')
```

Out[33]:
```
array[[<AxesSubplot:title={'center':'people_fully_vaccinated_per_hundred'}>]]
```


In [34]:
```
grouped_df.sort_values('people_fully_vaccinated_per_hundred')
```

Out[34]:

	total_vaccinations	daily_vaccinations	people_fully_vaccinated	people_fully_vaccinated_per_hundred
location				
Democratic Republic of Congo	1049063.0	2509.0	375313.0	0.44
Yemen	223984.0	10240.0	149080.0	0.55
Chad	123612.0	4723.0	246393.0	0.15
Turkmenistan	41993.0	0.0	5713.0	0.10
Haiti	67345.0	1503.0	18987.0	0.16
...	...	...	...	...
United Arab Emirates	19843232.0	1312112.0	9140120.0	81.77
Cayman Islands	107000.0	1004.0	34800.0	82.34
Portugal	15096050.0	139618.0	8602244.0	84.60
Pitcairn	963	1.0	47.0	100.00
Gibraltar	79010.0	1068.0	39463.0	117.13

232 rows × 4 columns

This doesn't seem to fit https://ourworldindata.org/grapher/covid-vaccination-doses-per-capita

Is this an error???