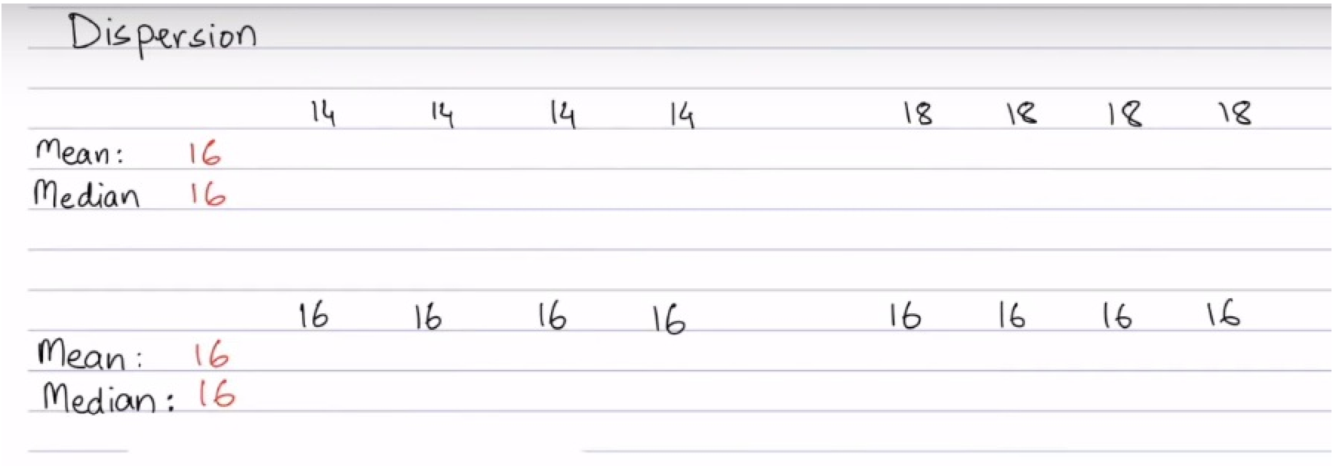
LECTURE 6

ISSUES WITH MEAN AND MEDIAN



SUPPOSE U HAVE 2 DATASETS IF U TELL SOMEONE MEAN AND MEDIAN THEY WILL THINK THAT THEY ARE THE SAME DATA WHICH IS NOT TRUE IN THIS CASE AS ONE HAS EXTREME VALUES AND THE OTHER HAS SAME

SO WE HAVE TO FIND THE SPREAD IN DATA WHICH IS CALLED DISPERSION.

SAMPLE:

WE ESTIMATE THE REST OF THE DATA BASED ON THE SMALLEST PART/SAMPLE

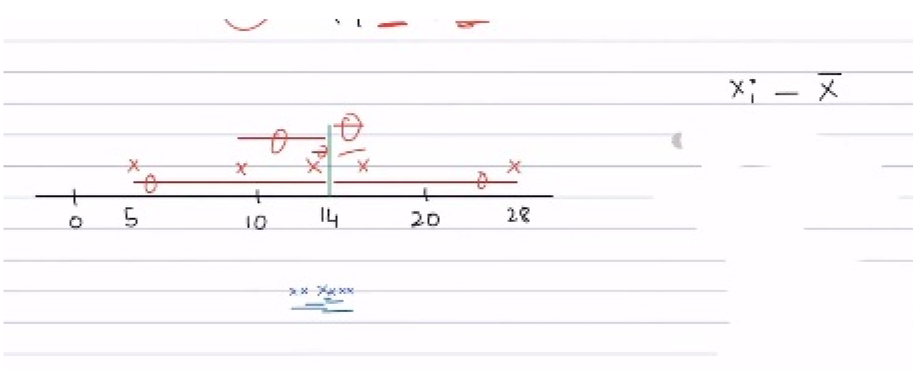
here we have to find the spread from the mean

dispersion is avg distance of values from the mean

$$\bar{x} = \Sigma xi / N$$

formula of mean

we have to find the points distance from the mean so in our mind it comes xi - $\bar{x}$ but there is a problem in it

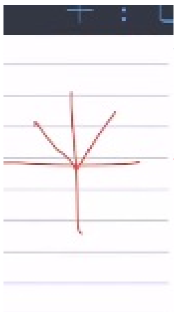


which is that distance cant be negative so changing the direction does not change the distance value from positive to negative

so one method is to take the absolute

$| x_i - \bar{x} |$

BUT AGAIN WE HAVE SOME PROBLEMS WITH ABSOLUTE



WHICH IS ABSOLUTE IS NOT DIFFERENTIABLE AT MINIMUM POINT

SO INSTEAD OF ABSOLUTE WE TAKE THE SQUARE OF THE VALUE WHICH MAKE IT POSITIVE

Now we are doing ***avg squared distance from the mean → variance***

$$\sigma^2 = \Sigma(x_i - \bar{x})^2 \ / \ N$$

Problem: Original data units : m

variance units : m^2

now the problem is that the original data units is in m while the variance units is in square

solution : square root

σ which is standard deviation (square root of the variance and its units will be that of the values)

Standard Deviation:

"Square root of

average square distances of values from the mean"

"How far away are values from the mean..."

we use numpy to generate random values(generating small data for understanding)

np.random.uniform

low → lowest value high → highest value size → no of values

```
import numpy as np
```

```
import seaborn as sns
sns.set(color_codes=True)
sns.set_style("white")     # See more styling options here: https://seaborn.pydata.or
```

```
np.random.uniform(low=0.0, high=1.0)
```

```
np.random.uniform(low=0.0, high=1.0, size = 10)
# All values equally likely. Sort of like     [  0.0   0.2   0.4    0.6    0.8   1.0
```

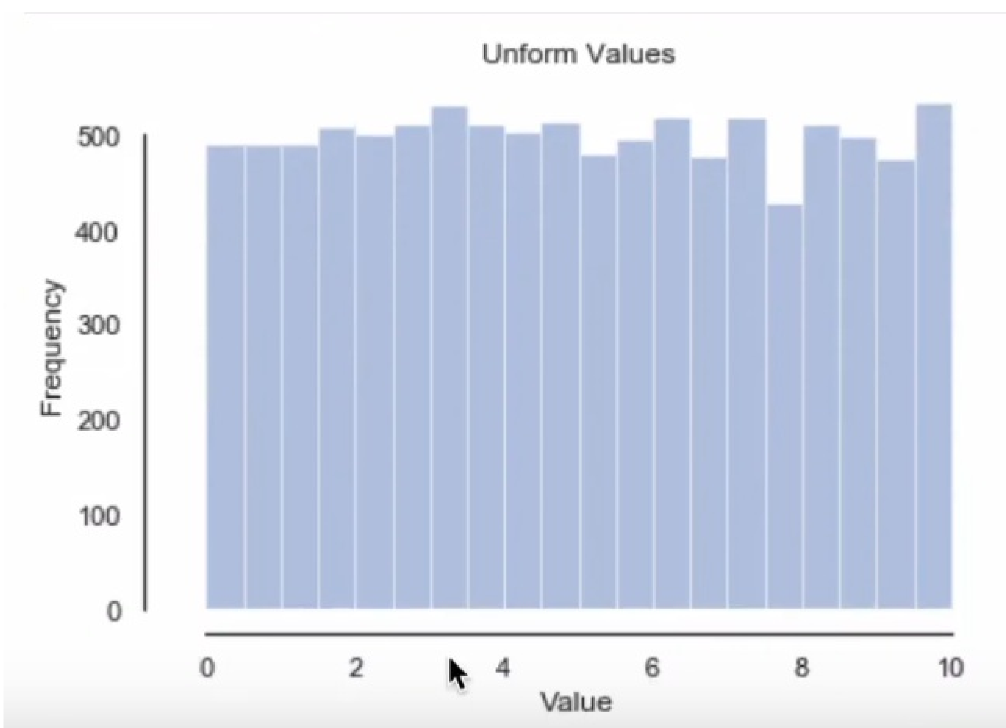
Generating a LOT of numbers

```
num_samples = 10000    # get rid of 'magic numbers'
```

```
uniform_vals = np.random.uniform(low=0.0, high=10.0, size = num_samples)
```

```
sns.distplot(uniform_vals, bins=20, kde=False)
plt.ylabel('Frequency')
plt.xlabel('Value');
plt.title("Unform Values")
sns.despine(offset=10, trim=True);  # move axes away
plt.show()
```

plotting



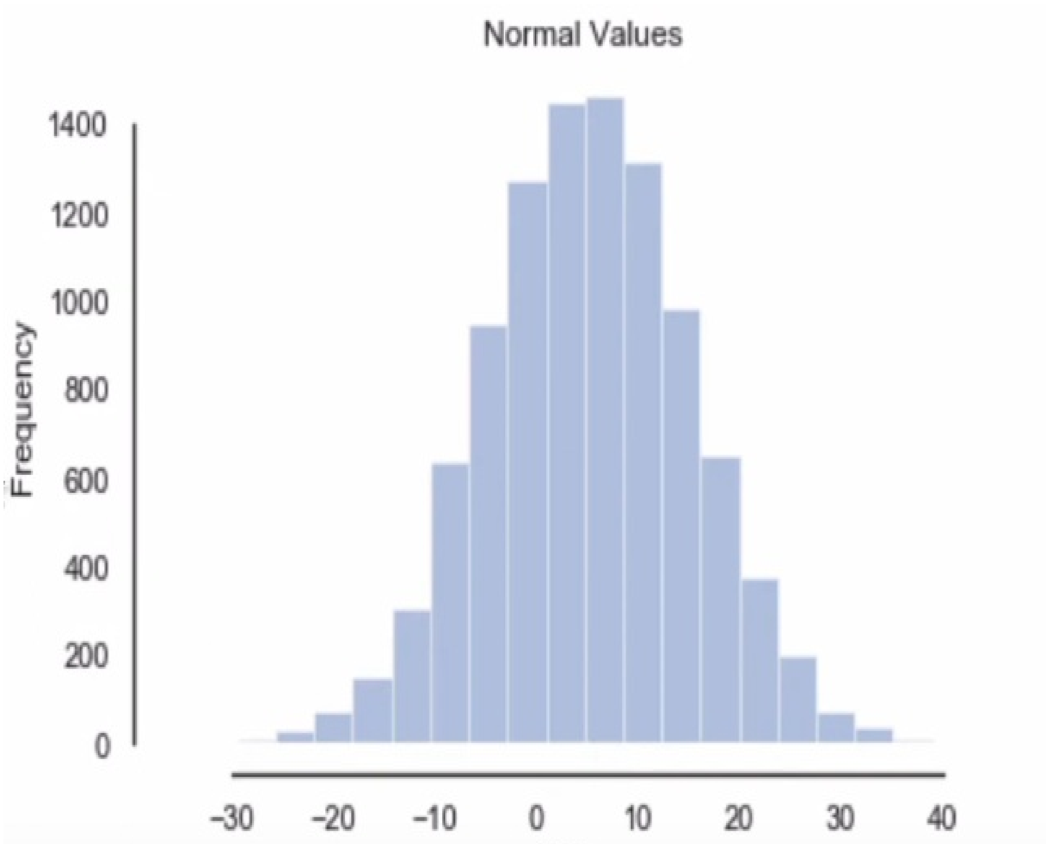
Now we are genrating another data but here we are genarating more values near the position 5 for this we have use np.random.normal and set the loc to 5

Value

```
[ ]: normal_vals = np.random.normal(loc=5.0, scale=10.0, size = num_samples)
     # More values closer to `loc`    ... Sorf of like [ 1 5 5 5 5 7 ]
```

```
[ ]: sns.distplot(normal_vals, bins=20, kde=False)
     plt.ylabel('Frequency')
     plt.xlabel('Value');
     plt.title("Normal Values")
     sns.despine(offset=10, trim=True);  # move axes away
     plt.show()
```

As we can see most values are near the 5



Mean of both samples

```
print("Uniform vals mean:", np.mean(uniform_vals))
print("Normal vals mean: ", np.mean(normal_vals))

Uniform vals mean: 4.986530741552067
Normal vals mean:  5.021687174353235
```

As mean was almost same for both samples but from variance we can see that there is massive difference which tells that the spread is higher in normal values.

```
print("Uniform vals variance:", np.var(uniform_vals))
print("Normal vals variance: ", np.var(normal_vals))
```

```
Uniform vals variance: 8.298200318710768
Normal vals variance:  101.43193559335936
```

Std for seeing the original

```
print("Uniform vals sd:", np.std(uniform_vals))
print("Normal vals sd: ", np.std(normal_vals))
```

```
Uniform vals sd: 2.8806597019972298
Normal vals sd:  10.071491229870547
```

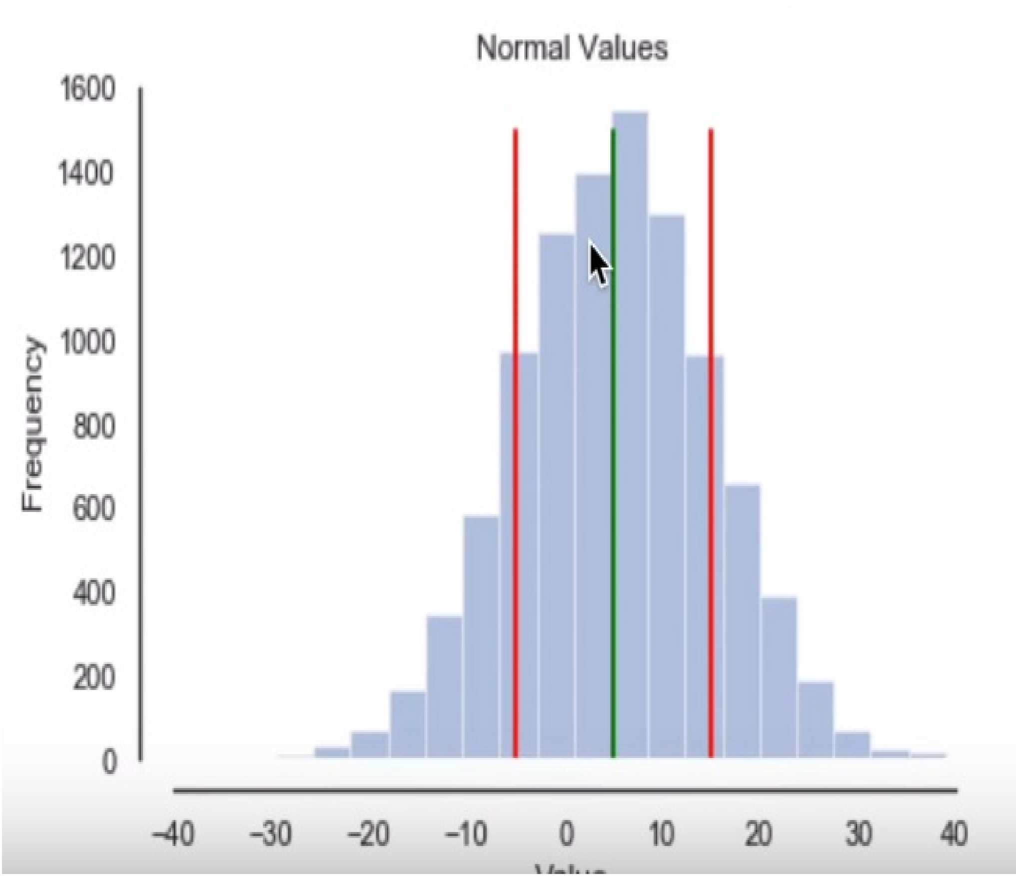
now taking another sample

graph: green → mean red → mean + std *(left)* and mean – std *(right)*

```
normal_vals = np.random.normal(loc=5.0, scale=10.0, size = num_samples)

nv_mean = np.mean(normal_vals)
nv_sd = np.std(normal_vals)
```

```
sns.distplot(normal_vals, bins=20, kde=False)
# plt.xlim(-60, 60)

plt.ylabel('Frequency')
plt.xlabel('Value');
plt.title("Normal Values")
sns.despine(offset=10, trim=True);  # move axes away

# plot the SD line
x_c, y_c = ([nv_mean, nv_mean], [0, 1500])
plt.plot(x_c, y_c, color='green', linewidth=2)

x_c, y_c = ([nv_mean + nv_sd, nv_mean + nv_sd], [0, 1500])
plt.plot(x_c, y_c, color='red', linewidth=2)


x_c, y_c = ([nv_mean - nv_sd, nv_mean - nv_sd], [0, 1500])
plt.plot(x_c, y_c, color='red', linewidth=2)
```

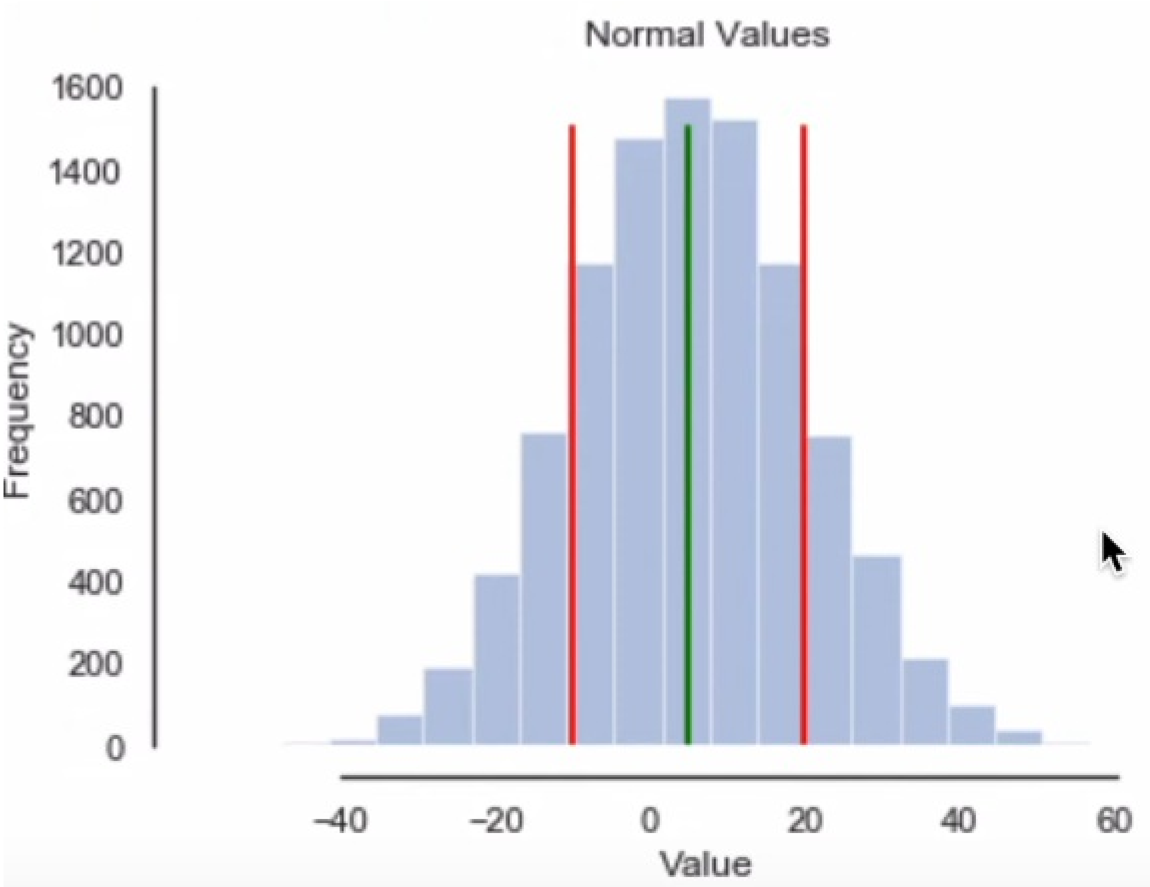
These values lie in one standard deviation of the mean(red to red)



Now change the scale to 15 which means the dispersion

```
[36]: normal_vals = np.random.normal(loc=5.0, scale=10.0, size = num_samples)

      nv_mean = np.mean(normal_vals)
      nv_sd = np.std(normal_vals)
```

```
[37]: sns.distplot(normal_vals, bins=20, kde=False)
      # plt.xlim(-60, 60)

      plt.ylabel('Frequency')
      plt.xlabel('Value');
      plt.title("Normal Values")
      sns.despine(offset=10, trim=True);  # move axes away

      # plot the SD line
      x_c, y_c = ([nv_mean, nv_mean], [0, 1500])
      plt.plot(x_c, y_c, color='green', linewidth=2)

      x_c, y_c = ([nv_mean + nv_sd, nv_mean + nv_sd], [0, 1500])
      plt.plot(x_c, y_c, color='red', linewidth=2)


      x_c, y_c = ([nv_mean - nv_sd, nv_mean - nv_sd], [0, 1500])
```



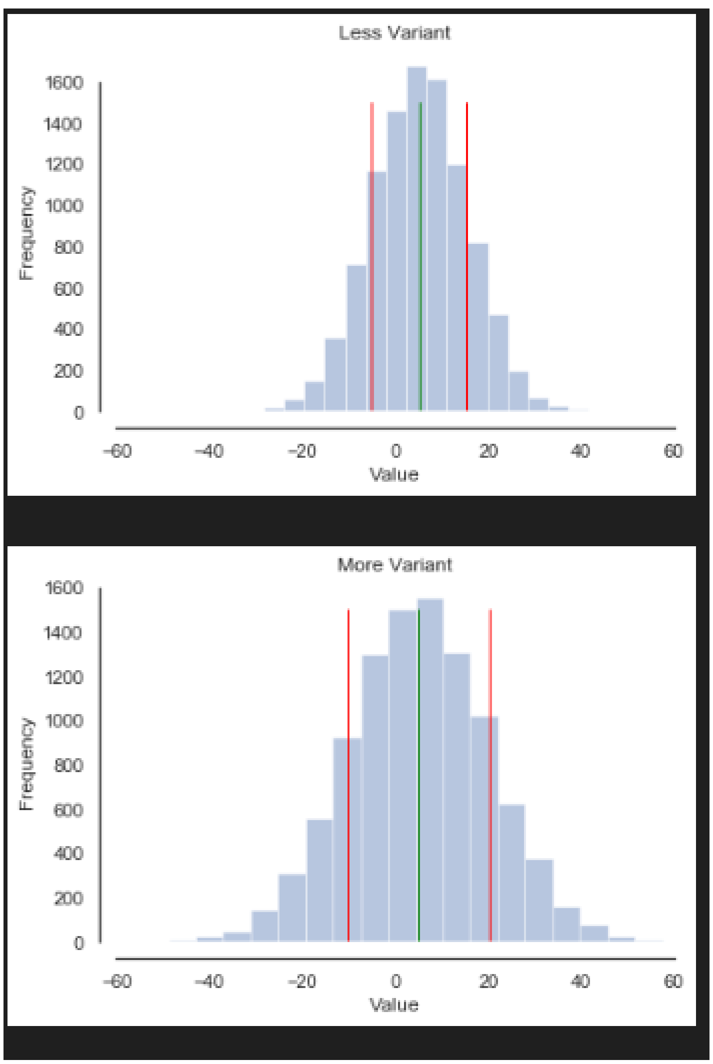
difference between the two that its max is 60 while the above one was 40

units are different

but when u can give ur code to someone will not look at the units so for this first set the limits

plt.xlim(-60,60)

```
# Let's put both together
normal_vals = np.random.normal(loc=5.0, scale=10.0, size = num_samples)
normal_vals_2 = np.random.normal(loc=5.0, scale=15.0, size = num_samples)

def plot_dist(vals, label):
    nv_mean = np.mean(vals)
    nv_sd = np.std(vals)

    sns.distplot(vals, bins=20, kde=False)

    plt.xlim(-60, 60)

    plt.ylabel('Frequency')
    plt.xlabel('Value');
    plt.title(label)
    sns.despine(offset=10, trim=True);  # move axes away

    # plot the SD line
    x_c, y_c = ([nv_mean, nv_mean], [0, 1500])
    plt.plot(x_c, y_c, color='green', linewidth=1)

    x_c, y_c = ([nv_mean + nv_sd, nv_mean + nv_sd], [0, 1500])
    plt.plot(x_c, y_c, color='red', linewidth=1)
```



Now we work with real world data

library used → sklearn

from sklearn.datasets.california_housing import fetch_california_housing

description:

```
houses = fetch_california_housing()
```

```
print(houses.DESCR)
```

```
. _california_housing_dataset:

alifornia Housing dataset
-------------------------

*Data Set Characteristics:**

    :Number of Instances: 20640

    :Number of Attributes: 8 numeric, predictive attributes and the target

    :Attribute Information:
        - MedInc        median income in block
        - HouseAge      median house age in block
        - AveRooms      average number of rooms
        - AveBedrms     average number of bedrooms
        - Population    block population
        - AveOccup      average house occupancy
        - Latitude      house block latitude
        - Longitude     house block longitude

    :Missing Attribute Values: None

his dataset was obtained from the StatLib repository.
ttp://lib.stat.cmu.edu/datasets/
```

list comprehension to find the med-inc of each house

```
[ x for x in  houses.data[:10]  ] #arrays of each house data
```

```
[array([   8.3252    ,   41.        ,    6.98412698,    1.02380952,
        322.        ,    2.55555556,   37.88      , -122.23      ]),
 array([ 8.30140000e+00,  2.10000000e+01,  6.23813708e+00,  9.71880492e-01,
         2.40100000e+03,  2.10984183e+00,  3.78600000e+01, -1.22220000e+02]),
 array([   7.2574    ,   52.        ,    8.28813559,    1.07344633,
        496.        ,    2.80225989,   37.85      , -122.24      ]),
 array([   5.6431    ,   52.        ,    5.8173516 ,    1.07305936,
        558.        ,    2.54794521,   37.85      , -122.25      ]),
 array([   3.8462    ,   52.        ,    6.28185328,    1.08108108,
        565.        ,    2.18146718,   37.85      , -122.25      ]),
 array([   4.0368    ,   52.        ,    4.76165803,    1.10362694,
        413.        ,    2.13989637,   37.85      , -122.25      ]),
 array([ 3.65910000e+00,  5.20000000e+01,  4.93190661e+00,  9.51361868e-01,
         1.09400000e+03,  2.12840467e+00,  3.78400000e+01, -1.22250000e+02]),
 array([ 3.12000000e+00,  5.20000000e+01,  4.79752705e+00,  1.06182380e+00,
         1.15700000e+03,  1.78825348e+00,  3.78400000e+01, -1.22250000e+02]),
 array([ 2.08040000e+00,  4.20000000e+01,  4.29411765e+00,  1.11764706e+00,
         1.20600000e+03,  2.02689076e+00,  3.78400000e+01, -1.22260000e+02]),
 array([ 3.69120000e+00,  5.20000000e+01,  4.97058824e+00,  9.90196078e-01,
         1.55100000e+03,  2.17226891e+00,  3.78400000e+01, -1.22250000e+02])]
```

```
med_incs = [   x[0] for x in  houses.data     ]   # list comprehension  #first element of each house data
```

Mean and median

```
np.mean(med_incs)
```

```
3.8706710029069766
```

```
np.var(med_incs)
```

```
3.609147689697444
```

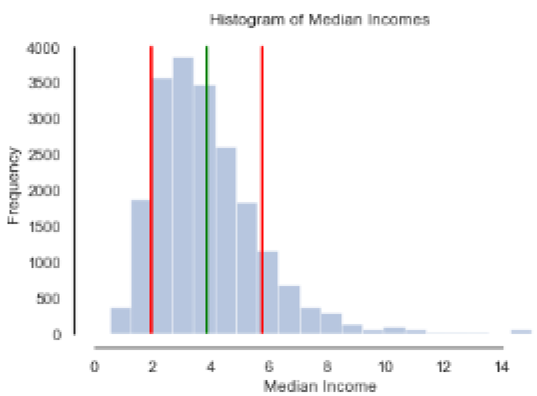
plotting

```
med_incs = [   x[0] for x in  houses.data     ]
mean = np.mean(med_incs)
sd = np.std(med_incs)


sns.distplot(med_incs, bins=20, kde=False)

# plt.xlim(-60, 60)

plt.ylabel('Frequency')
plt.xlabel('Median Income');
plt.title("Histogram of Median Incomes")
sns.despine(offset=10, trim=True);  # move axes away

# plot the SD line
x_c, y_c = ([mean, mean], [0, 4000])
plt.plot(x_c, y_c, color='green', linewidth=2)

x_c, y_c = ([mean + sd, mean + sd], [0, 4000])
plt.plot(x_c, y_c, color='red', linewidth=2)


x_c, y_c = ([mean - sd, mean - sd], [0, 4000])
plt.plot(x_c, y_c, color='red', linewidth=2)

plt.show()
```

this graph is skewed we have find from one dispersion value that how far the values goes but we havn't find how far the value goes from one side to the mean

how far the values goes to the right from the mean and to the left from the mean

here dispersion is ***unbalanced(a-symmetric)*** *so the graph is skewed*



how large is SD?
DEPENDS ON MEAN

Is 10.7 large? How about 1094?

— Depends on the mean ...

$$\frac{10.7}{100} * 100 = 10.7$$

$$\frac{1094}{100,000} \times 100 = 1.09$$

Coefficient of variation : $\frac{\sigma}{\bar{x}} * 100$

" Relative measure of deviation."