Practical_Statistics

August 22, 2019

Coursebook: Inferential Statistics - Part 1 of *Python Fundamental Course* - Course Length: 9 Hours - Last Updated: July 2019

- Developed by Algoritma's product division and instructors team

1 Background

The coursebook is part of the **Python Fundamental Course** prepared by Algoritma. The coursebook is intended for a restricted audience only, i.e. the individuals and organizations having received this coursebook directly from the training organization. It may not be reproduced, distributed, translated or adapted in any form outside these individuals and organizations without permission.

Algoritma is a data science education center based in Jakarta. We organize workshops and training programs to help working professionals and students gain mastery in various data science sub-fields: data visualization, machine learning, data modeling, statistical inference etc.

1.1 Training Objectives

On the first section of this course is to pave the statistical foundation for more advance implementation and machine learning implementation. There will be 2 main objectives:

- **Descriptive Statistics**
- Understanding 5 number summary
- Central tendency measure
- Variability measure
- Z Score and Central limit theorem
- **Inferential Statistics**
- Probability density function
- Confidence intervals
- Hypothesis test
- Power analysis

At the end of this course, we'll be working with Learn by Building module as your graded assignment. You'll be working in groups and design an A/B testing of your choice!

2 Descriptive Statistics

Statisticians and data scientists use descriptive statistics to summarize and describe a large number of measurements. Many times, this task is accompanied with graphs and plots that help describe the numerical summary of data. When data science is applied in the business context, an example of descriptive statistic is the average number of transactions per month. Another example is the percentage of e-commerce transactions with a voucher code applied. The simple rule is that descriptive statistics do not involve generalizing beyond the data we have obtained, and are merely descriptive of what we have at hand. The branch of statistics that deal with drawing inferences about the larger population is called inferential statistics.

In describing data, we are typically concerned with the task of quantifying and comparing central tendency, variability, and the shape of our data. In this coursebook, we'll be working on with Tokopedia listing item in handphones section:

[96]:
```
import pandas as pd

hp = pd.read_csv('tokped_hp.csv', encoding='latin8')
hp.dtypes
```

[96]:
```
product_title      object
location           object
account_seller     object
price             float64
review              int64
installment       float64
rating            float64
brand_hp           object
dtype: object
```

The dataset consist of several listing information such as product name, price, review, rating, and brand. Let's explore some of them first to get a good sense from our data:

[97]:
```
hp.brand_hp.value_counts().head(10)
```

[97]:
```
xiaomi        1404
samsung       1095
apple          559
oppo           441
lain-lain      306
vivo           243
nokia          209
asus           178
realme         115
sony           107
Name: brand_hp, dtype: int64
```

The dataset consist of some well known handphone brand, as we can see most of the data is dominated by Xiaomi and Samsung brand. It is also corresponds to Indonesian market interest. In the next sections, as we go through this module, we are going to explore this specific dataset and see how an Indonesian mobile sales landscape can be extracted using this dataset.

2.1 Measures of Central Tendency

Often times in the exploratory data analysis phase, we want to get a sense of what the **most representative score** of a particular measurement is. We often simplify this idea by referring to it as the "average", but there are in fact, three measures of central tendency that you need to have in your statistical toolset.

The most popular measure of central tendency is the **mean**, which is sometimes represented as $\bar{x}$ when computed on a **sample** and represented as μ when computed on a population. Mean is really the sum of all your measurements, divided by the number of measurements, and works best on data that has an even distribution or a normal distribution (don't worry if the idea of a normal distribution isn't clear - we'll get to that in a while!).

Say for example, we are want to see the mean price of Apple phones. Luckily, pandas data frame came with a function that works well with a Series object:

```
[98]: 'IDR {:,.2f}'.format(hp[hp.brand_hp == 'apple'].price.mean())
```

```
[98]: 'IDR 5,677,991.95'
```

The **median** is the point of value that cuts the distribution into two equal halves such that 50% of the observations are below it. To find this value, we would order the observations and find the middle value that separates the distribution into two equal halves.

```
[99]: 'IDR {:,.2f}'.format(hp[hp.brand_hp == 'apple'].price.median())
```

```
[99]: 'IDR 3,999,000.00'
```

For data with odd number of observations, the median is the middle value but for data with an even number of observations we would instead use the average of the two middle scores:

```
[100]: odds_price = hp[hp.brand_hp == 'apple'].iloc[0:9,:]
       odds_price.price.median()
```

```
[100]: 3300000.0
```

```
[101]: odds_price.sort_values('price').price.tolist()[4]
```

```
[101]: 3300000.0
```

```
[102]: odds_price = hp[hp.brand_hp == 'apple'].iloc[0:10,:]
       odds_price.price.median()
```

```
[102]: 2925000.0
```

```
[103]: fourth = odds_price.sort_values('price').price.tolist()[4]
       fifth = odds_price.sort_values('price').price.tolist()[5]

       (fourth + fifth)/2
```

```
[103]: 2925000.0
```

We need to be cautious when applying the mean on data with a skewed distribution because the mean may not be the best candidate for a most representative score compared to other measures of central tendency. Let's take the first 99 rows of our apple phones:

```
[104]: apple_sample = hp[hp.brand_hp == 'apple'].iloc[0:99,:]
       apple_sample.describe()
```

[104]:
```
              price      review    installment     rating
count  9.900000e+01   99.000000      99.000000  99.000000
mean   4.753343e+06   94.494949  198056.323232   4.818182
std    3.653102e+06   87.872864  152212.612375   0.153439
min    3.950000e+05    6.000000   16459.000000   4.300000
25%    2.449500e+06   36.500000  102062.500000   4.700000
50%    3.500000e+06   68.000000  145834.000000   4.800000
75%    5.350000e+06  127.500000  222917.000000   4.950000
max    1.720000e+07  665.000000  716667.000000   5.000000
```

And then imagine if there's a new product listing selling an Apple iPhone XS Max 512GB priced at IDR 27 Mil. Let's simulate that and calculate the sample's mean:

[105]:
```
new_product = {
    'product_title': ['Apple iPhone XS Max 512 GB'],
    'price': [27000000],
    'brand_hp': ['apple']
}

new_row = pd.DataFrame(data=new_product)
apple_sample = apple_sample.append(new_row, sort=False)
```

Now calculating the central tendency:

[106]:
```
'{:,.2f}'.format(apple_sample.price.mean())
```

[106]:
```
'4,975,810.00'
```

[107]:
```
'{:,.2f}'.format(apple_sample.price.median())
```

[107]:
```
'3,500,000.00'
```

The mean puts that figure at about almost 1.5 times higher and is not truly representative of the actual price. While as we can see in the initial `describe()` that only 25% of our data has price more than 5.3 Mil, the mean value of price would have believe that the average price of our Apple phones was in fact is around 4.9 Mil.

The median in this case is a better measure of centrality because it is not sensitive to the outlier data.

If we are in fact, *required* to compute the mean on data with skewed distribution, another technique to reduce the influence of outlier data is to use a slight variation of the mean, called the Trimmed Mean. The trimmed mean removes a small designated percentage of the largest and smallest values before computing the mean. While pandas data frame does not support trimmed mean, we can use `scipy` package for `trim_mean()`:

[108]:
```
from scipy.stats import trim_mean

'{:,.2f}'.format(trim_mean(apple_sample.price, .1))
```

[108]:
```
'4,165,725.00'
```

The second parameters would trimmed 10% of the leftmost and rightmost values after sorting, and this figure is slightly beliavable than the previous one.

When there are discreet values for a variable, the **mode** refers to the value that occurs most frequently. This statistic is rarely used in practice, but pandas data frame also come in with the function:

[109]:
```
hp[hp.brand_hp == 'apple'].location.mode()
```

[109]:
```
0    Jakarta
dtype: object
```

2.2 Measures of Spread

Measures of spread measures the extent to which **value in a distribution differ from each other.** In practice, it is far easier to compute the distance between the values to their mean and when we square each one of these distances and add them all up the average1 of that result is known as **variance**. Taking the square root of the variance will result in the **standard deviation**. Just like the mean, standard deviation is the "expected value" of how far the scores deviate from the mean.

To change the scenery a bit, let's take a look at Xiaomi brand from our dataset:

[110]:
```
hp[hp.brand_hp == 'xiaomi'].price.var()
```

[110]:
```
1269429778891.8962
```

And taking the square root of variance yields the standard deviation:

[111]:
```
'{:,.2f}'.format(hp[hp.brand_hp == 'xiaomi'].price.std())
```

[111]:
```
'1,126,689.74'
```

As stated before, think of standard deviation as the "expected standard deviation from the mean", means that if we were to calculate its mean, we expect positive and negative deviation around 1.1 Mil from the mean to be "normal".

[112]:
```
'{:,.2f}'.format(hp[hp.brand_hp == 'xiaomi'].price.mean())
```

[112]:
```
'1,951,973.21'
```

Variance and standard deviation are always positive when the values are not identical. When there's no variability, the variance is 0. Because variance and standard deviation are sensitive to every value, they may not be the most "representative" measurement for skewed data.

Other measurements of the spread are the **range** and the **interquartile range**. The range is the distance from our smallest measurement to the largest one:

[113]:
```
top = hp[hp.brand_hp == 'xiaomi'].price.max()
bottom = hp[hp.brand_hp == 'xiaomi'].price.min()

'{:,.2f}'.format(top-bottom)
```

[113]:
```
'9,570,000.00'
```

The interquartile range is the range computed for the middle 50% of the distribution:

[114]:
```
third_quantile = hp[hp.brand_hp == 'xiaomi'].price.quantile(0.75)
first_quantile = hp[hp.brand_hp == 'xiaomi'].price.quantile(0.25)

'{:,.2f}'.format(third_quantile-first_quantile)
```

[114]:
```
'1,126,000.00'
```

Recall how we can call `describe()` to create a summarized numerical information for each column? It gives us a quick access to what we have learned in this section:

```
[115]: hp[hp.brand_hp == 'xiaomi'].price.describe()
```

```
[115]: count    1.404000e+03
       mean     1.951973e+06
       std      1.126690e+06
       min      5.000000e+03
       25%      1.199000e+06
       50%      1.636000e+06
       75%      2.325000e+06
       max      9.575000e+06
       Name: price, dtype: float64
```

Discussion:
Which financial assets has more votality in their annual price?

```
[116]: price_coins = [1.4, 0.4, 0.8, 1.1, 1.8, 2.2, 2.3, 1.2]
       price_oil = [1.6, 1.2, 1.9, 0.8, 0.6, 1.5, 2.1, 1.5]
```

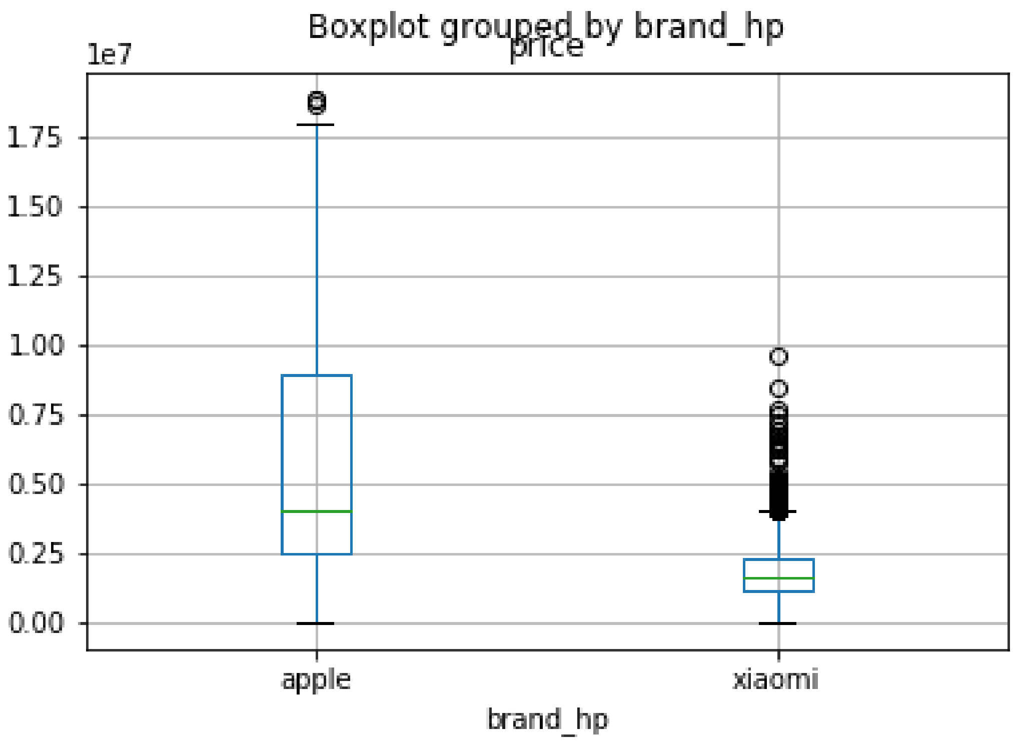
The primary measure of votality used by stock traders and financial analysts is standard deviation, and recall that this metric reflects the average amount of an item's price over a period of time. While the price for our fictional "oil" asset and "coins" asset averaged out to be USD 1.4 over time, which of these two present a higher votality than the other?

A common way to quickly inspect a data is by using visualization techniques. Let's recall back to last week course using Bokeh. The following codes is used to create a *box plot*. This plot provides you a five number summary and is really useful for you to understand the distribution. It contains a several components:
- First quantile
- Median
- Third quantile
- Inner and outer fence
- Outliers

Let's try to use pandas built-in `boxplot()` function:

```
[117]: brands = ['apple', 'xiaomi']
       xiaomi_apple = hp[hp.brand_hp.isin(brands)]

       xiaomi_apple.boxplot(column='price', by='brand_hp')
```

```
[117]: <matplotlib.axes._subplots.AxesSubplot at 0x211466382b0>
```



Discussion:

How do you interpret the above boxplot? Can you say whether or not the two brands has different pricing standard?

2.3 Covariance and Correlation

When we have two samples, X and Y, of the same size, then the **covariance** is an estimate of how **variation in X is related to the variation in Y**. Covariance measures how two variables *covary* and is represented as:

$Cov(X, Y) = \frac{1}{n-1} \sum_{i=1}^{n} (X_i - \mu_X)(Y_i - \mu_Y)$

[118]:
```
hp.cov()
```

[118]:
```
                    price        review   installment        rating
price        5.851683e+12 -3.626288e+07  2.438004e+11  66207.455167
review      -3.626288e+07  3.081258e+04 -1.511326e+06     -1.744481
installment  2.438004e+11 -1.511326e+06  1.015783e+10   2759.214768
rating       6.620746e+04 -1.744481e+00  2.759215e+03      0.035065
```

Getting a negative covariance means that smaller X tends to be associated with larger Y (and vice versa). The covariance of any variable with itself is its variance2. Notice also that $Cov(X, Y) = Cov(Y, X)$.

Correlation, unlike covariance, is not sensitive to the units in which our variables X and Y are measured and hence more useful for determining how strong the relationship is between variables:

[119]:
```
hp.corr()
```

[119]:
```
                price    review  installment    rating
price        1.000000 -0.085400     0.999985  0.146161
review      -0.085400  1.000000    -0.085427 -0.053072
installment  0.999985 -0.085427     1.000000  0.146201
rating       0.146161 -0.053072     0.146201  1.000000
```

Some facts about correlation:
- $Cor(X,Y) == Cor(Y,X) $ - $-1 <= Cor(X,Y) <= 1 $ - $Cor(X, Y)$ is 1 or -1 only when the X and Y observations fall perfectly on a positive or negatively sloped line
- $Cor(X, Y) = 0$ implies no linear relationship

Say you had a rather large amount of variables in your hand. While python can handle large numbers well, a visualization technique called correlation plot can save you some time in inspecting your data. First let's create the dictionary for each correlation value per pairs:

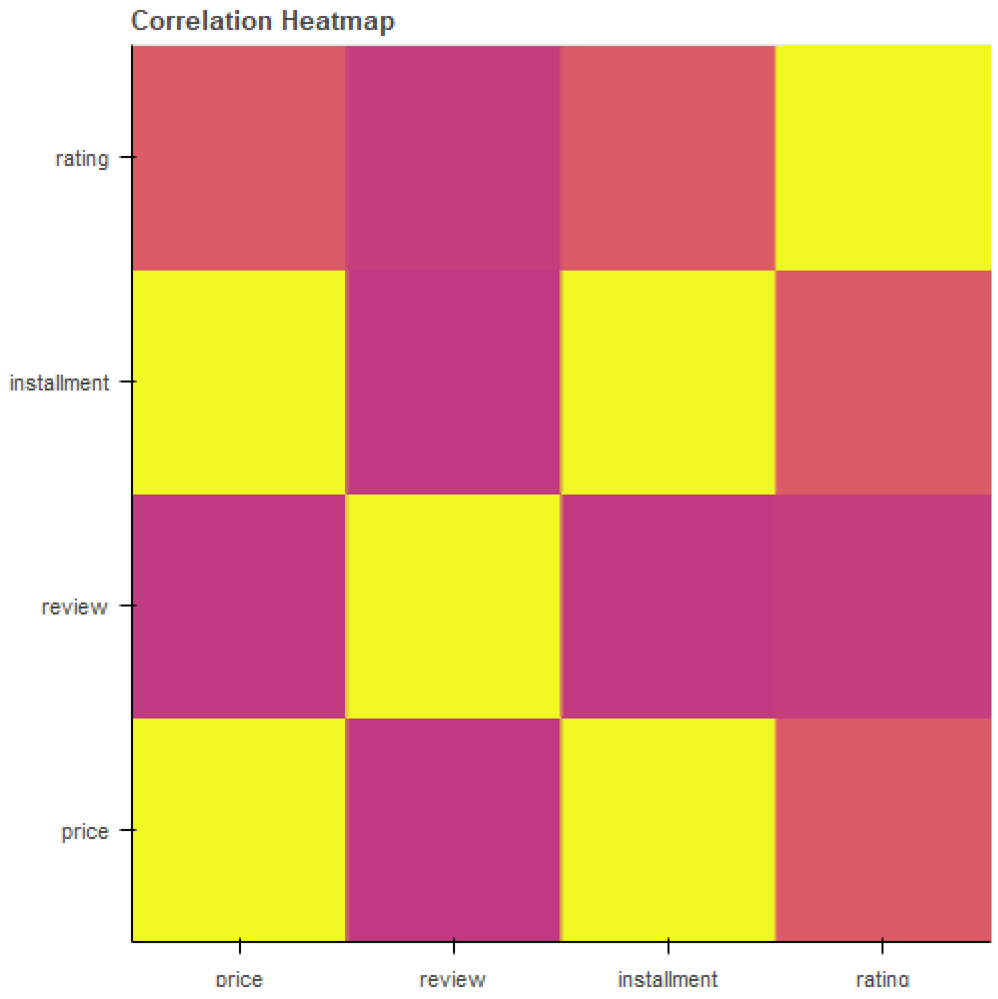
[120]:
```
corr_table = hp.corr()

val = []
x = []
y = []

for index,row in corr_table.iterrows():
    i = row.index.tolist()
    for e in range(0,len(row)):
        val.append(row[e])
        y.append(index)
        x.append(row.index.tolist()[e])

rect_dict = dict(x=x, y=y, corr=val)
```

Since we'll be using `bokeh` let's import the package and prepare our Notebook:

[121]:
```
from bokeh.io import show, output_notebook

output_notebook()
```

Next, use a `rect()` glyph to create each box. The plot will provide you with quick inspection in determining a higher correlated variables (yellow), and a lower one (purple):

[122]:
```
from bokeh.transform import linear_cmap
from bokeh.palettes import all_palettes
from bokeh.plotting import figure
from bokeh.models import ColumnDataSource

p = figure(title="Correlation Heatmap",
           x_range=corr_table.columns.tolist(),
           y_range=corr_table.columns.tolist(),
           toolbar_location=None,
           tools='hover, save',
           tooltips=[('Correlation','@corr{0.0000}')],
           plot_width=500,
           plot_height=500)

p.rect('x', 'y',
       source=ColumnDataSource(rect_dict),
       color=linear_cmap('corr', all_palettes['Plasma'][256], -1, 1),
       width=1,
       height=1)

show(p)
```

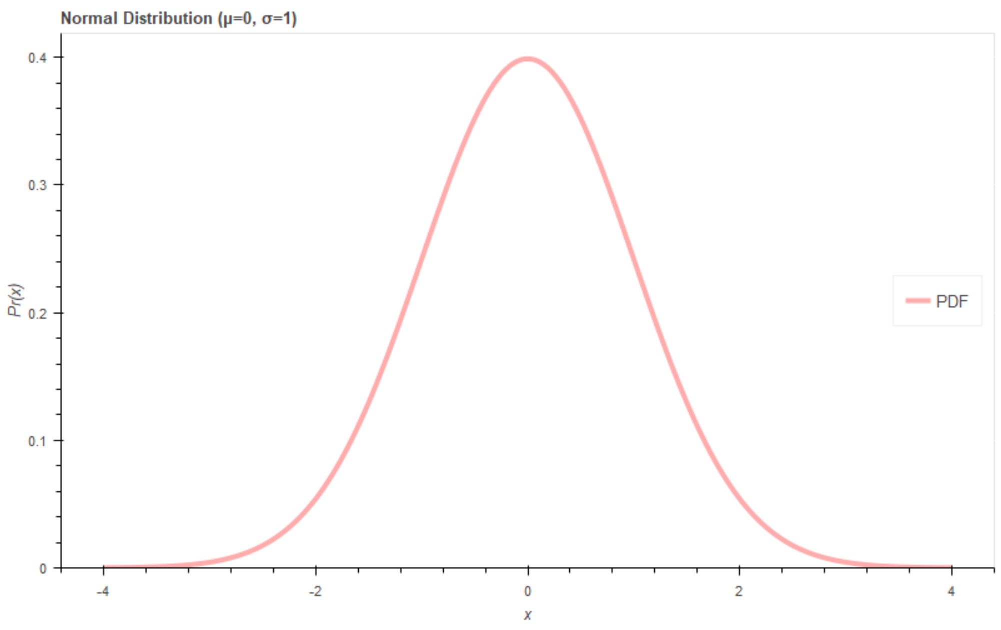


2.4 The Normal Distribution

Another way of studying the central tendency and spread of data is through a curve: a curve is often used to represent a distribution and the most famous of all distributions is the normal curve.

A normal distribution with a mean of 0 and standard deviation of 1 is called a **standard normal curve** and can be plotted with while specifying the limits for our x-axis:

[123]:
```
import numpy as np
from scipy.stats import norm
import scipy

def density_plot(title, x, pdf):
    p = figure(title=title,
               tools='hover',
               toolbar_location=None,
               tooltips=[('Probability','@pdf{0.00000}'),('Sigma','@x{0.
 ↪00000}')],
               plot_width=800,
               plot_height=500)

    di = dict(x=x, pdf=pdf)

    p.line('x', 'pdf', line_color="#ff8888", line_width=4, alpha=0.7,␣
 ↪legend="PDF", source=ColumnDataSource(di))

    p.y_range.start = 0
    p.legend.location = "center_right"
    p.xaxis.axis_label = 'x'
    p.yaxis.axis_label = 'Pr(x)'
    p.grid.grid_line_color="white"
    return p


mu, sigma = 0, 1

x = np.linspace(-4, 4, 1000)
pdf = norm.pdf(x)

p = density_plot(f"Normal Distribution (={mu}, ={sigma})", x, pdf)

show(p)
```



When a measurement follows a standard normal distribution, then the assumptions of a normal distribution can be applied to the data and these assumptions can be completely specified by two parameters, which are the mean and standard deviation. The empirical rule of a standard normal gives us the following:

- 68% of data will fall within 1 standard deviation of the mean
- 95% of data will fall within 2 standard deviations of the mean
- 99.7% of data will fall within 3 standard deviations of the mean

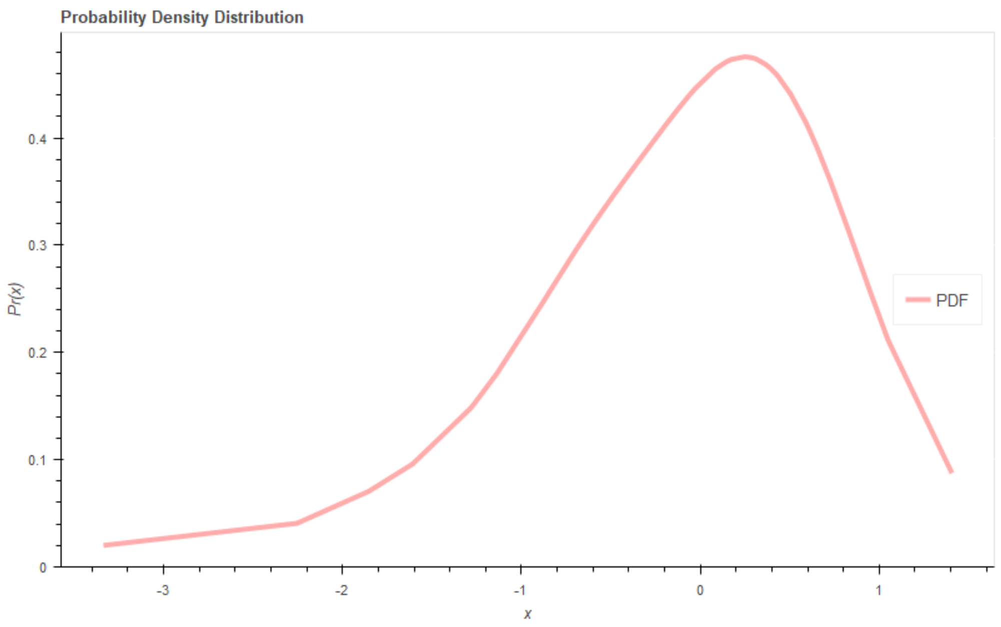
Scroll back to the normal curve we plotted above, observe:
- It is perfectly symmetrical
- It is unimodal (has only a single mode)
- Area under curve is 1

One relating idea that gives the normal distribution such significance is known as the **Central limit theorem**: it says that when we have many independent variables generated by all kinds of distributions, the aggregate of those variables will tend toward a normal distribution assuming of course the lack of any extraordinary intervention. This universality is observed across different domains making the normal distribution a core centerpiece in applied statistics and mathematics.

As an exercise, I'd like you to generate 50 random numbers using `numpy` indicating mean of 0 and standard deviation of 1. Now use the `density()` function I created and create a plot based on the generated 50 random numbers:

[124]:
```
from scipy import stats

# Careful on more than 50,000 samples will exhaust your machine
n = 50
x = np.random.randn(n)
x.sort()

# Generate pdf estimation using gaussian kernel
```

```
pdf = stats.gaussian_kde(x)
pd = np.reshape(pdf(x).T, len(x))

p = density_plot("Probability Density Distribution", x, pd)

show(p)
```



Supposed you were to change 10 to 100, and then to 1000 and even 10,000, what did you observe? The key takeaway here is that as the number of sample approach infinity this plot will eventually converge in distribution to the standard normal.

3 Inferential Statistics

3.1 Probability Density Function and Probability Mass Function

When we're thinking about continuous random variables (blood sugar level, height, rainfall amount), it is important to realize that this variable has an uncountable number of possible values, even between two real intervals. The resulting probability distribution of the variable can be described by a probability density, where the probability is found by taking the area under the curve.

Add normal distribution pdf here

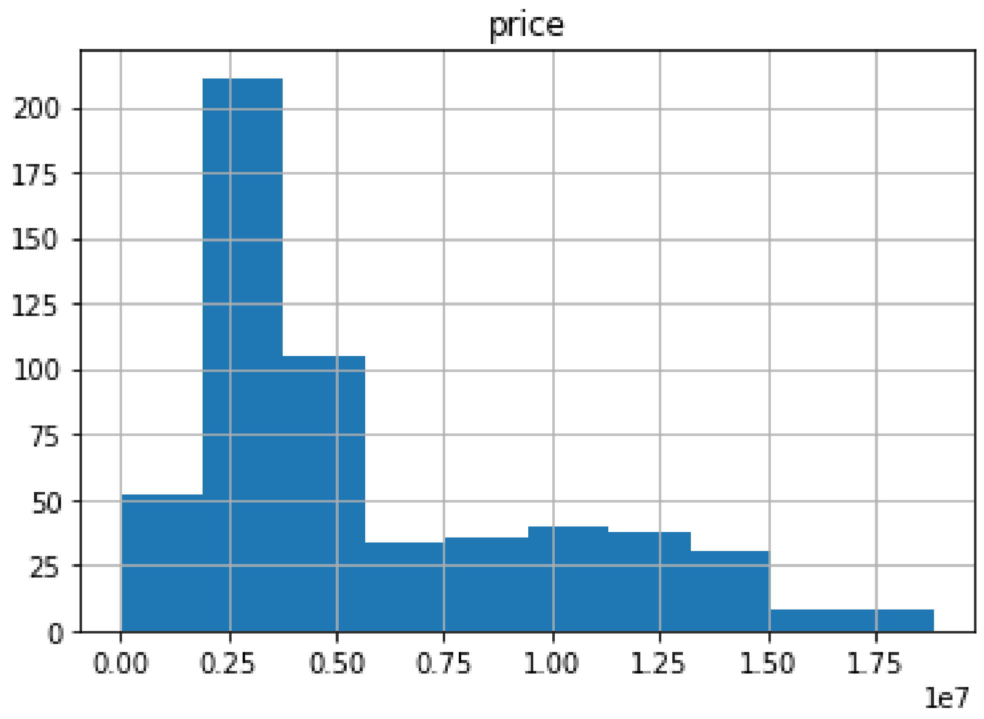
Discrete random variables (number of player injury, amount of defaulted loans, travel bookings per customer), on the other hand, can be described using a probability mass function, which maps each value of the random variable to a probability:
- p(0 bookings) = 0.28
- p(1 booking) = 0.09
- . . .
- p(6 bookings) = 0.004

Because they are probabilities, these individual probabilities have to sum up to 1.

Before going into probability density function and mass function, it is important to understand that the function is estimated by binning the distribution values. In pandas DataFrame, we also have hist() function to create a quick visual inspection using histogram:

[125]:
```
hp[hp.brand_hp == 'apple'].hist(column='price')
```

[125]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x00000211464C46D8>]],
      dtype=object)
```

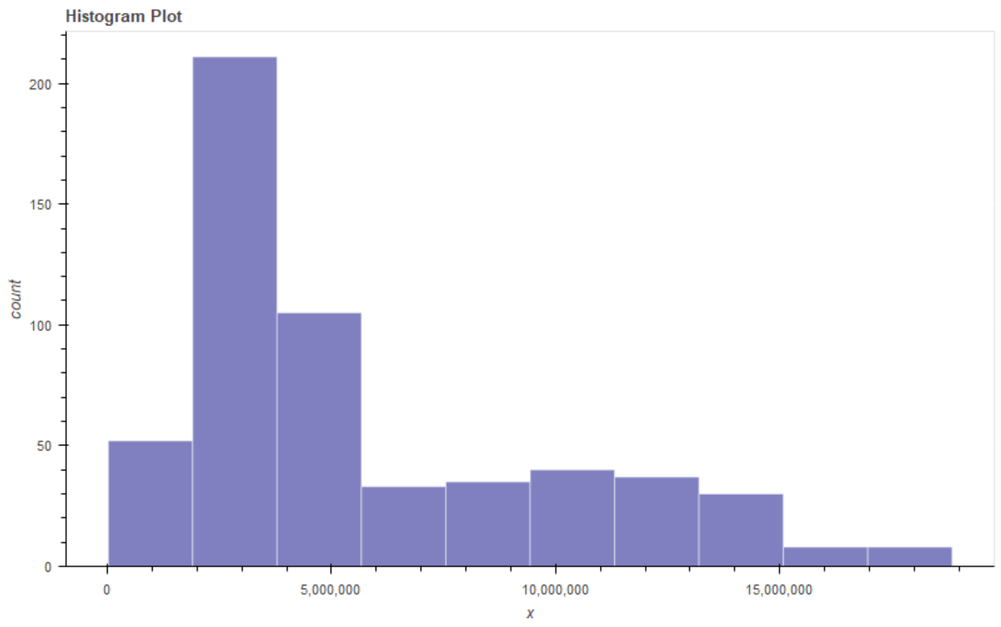


Other approach to make a prettier histogram is to create one using bokeh:

[126]:
```
from bokeh.models.formatters import NumeralTickFormatter

def histogram_plot(title, hist, edges):
    p = figure(title=title,
               tools='',
               toolbar_location=None,
               plot_width=800,
               plot_height=500)

    p.quad(top=hist, bottom=0, left=edges[:-1], right=edges[1:],
           fill_color="navy", line_color="white", alpha=0.5)

    p.y_range.start = 0
    p.xaxis.axis_label = 'x'
    p.yaxis.axis_label = 'count'
    p.grid.grid_line_color="white"
    p.xaxis.formatter = NumeralTickFormatter(format="0,0")

    return p
```

Then generate a random numbers and use `np.histogram()` to generate the counts and binning ranges:

[127]:
```
n = 50
x = hp[hp.brand_hp == 'apple'].price
hist, edges = np.histogram(x, bins=10)

p = histogram_plot("Histogram Plot", hist, edges)

show(p)
```



Dive Deeper:

Using our previous `hp` dataset, can you inspect visually, between the top 10 most listed brands, is it safe to assume that each brand has it own market segment?

Tips: A brand that has single market segment will show a unimodal shapes in our histogram

[128]:
```
# Your code here
```

3.2 Fundamentals of Hypothesis Testing

Going back to the standard normal distribution - you may be asking by now how any of what you're learning in the past few chapters are useful. To answer the question, I feel it is only appropriate we solidify these intuition with a few concrete examples. Consider the following scenario:

The height of men in Indonesia is normally distributed with a mean of 160cm and a standard deviation of 7cm. What is the probability of a randomly selected man being taller than 175cm?

[129]:
```
import scipy.stats as st

Z = (175-160)/7
```

```
p = st.norm.cdf(Z)
print(1-p)
```

```
0.016062285603828275
```

The solution: 175cm is 15cm above the mean, and dividing that by the standard deviation of 7cm, we get 2.143. We refer to this as the **z-score**. The probability of an Indonesian men being taller than 175cm is P(Z > 2.143) = 0.161.

Discussion:

Supposed we have a population mean of 180cm and standard deviation of 8cm, what is the probability that a randomly picked person being shorter than 174cm?

[130]:
```
# Your answer here
```

The Z-scores we used above is useful when relating different measurement distributions to each acting as a common denominator. Essentially, a z-score gives us a "standardized" unit that measure how many standard deviations is a particular statistic away from the mean. This property, as we'll see, is paramount to many statistical hypothesis tests, performance evaluation (more of that when we get to the Machine Learning courses), and in the construction of confidence or prediction intervals.

3.2.1 Standard Error and Confidence Intervals

Standard error can be thought as a related concept to standard deviation. It is a measure that estimates how close a calculated mean (sample) is likely to be to the true mean of that population; It is calculated by dividing the standard deviation of the sample data by the square root of the number of observations:

[131]:
```
import math

(hp.price.std())/ math.sqrt(hp.price.size)
```

[131]:
```
34487.19271046868
```

[132]:
```
from scipy import stats

stats.sem(hp.price)
```

[132]:
```
34487.19271046868
```

On a normal distribution (we'll get to this in a moment), 67% of the time the true population mean will lie within the range of +- 1 SE. In other words, if we can establish a normal distribution, we can theorize about the true value of the population mean with a range. Because of the formula:

$SE = \frac{\sigma}{\sqrt{n}}$

The larger our sample size n gets, the smaller SE will be: and hence we are less uncertain about our estimation of the true population mean.

We often begin our estimation with a point estimate, using for example the sample mean $\bar{x}$ as a point estimate of the population mean μ. We can then construct confidence intervals around our point estimates so we have an interval that may contain the true value of the parameter. When statisticians say a "95% confidence interval", what they mean is that if we create 100 confidence intervals of the same size from a given population, we expect to find the true parameter (let's say the population's net savings per household) in 95 of them.

We construct a confidence interval by taking the point estimate +/- margin of error, where margin of error is computed as:

$E = Z_{a/2} \times SE$

Because confidence intervals are two-sided the level of significance we chose (alpha) has to be divided into halves. When we compute by finding the z-score associated with a value of 2.5% (0.025) on each end, we end up looking at the middle 95% of the area under the curve. We can use `ppf()` from `scipy` package to help us find the z-score associated with a 95% confidence interval):

[133]:
```python
from scipy.stats import norm

norm.ppf(.975)
```

[133]:
```
1.959963984540054
```

Again, let's put together a scenario to make all of this more concrete. Say we want to know the average annual dividend payout in a particular industry, and had known through an earlier study that this figure resembles a normal distribution with a population standard deviation of 2.4%. We looked at the public books of these 81 companies in said industry and attain a sample mean of 11.8% (that is, the average company from this group of 81 companies pay 11.8% of profit to their shareholders annually). We want to construct a 95% confidence interval for the μ, the mean dividend payout.

[134]:
```python
# Your code here
```

Solution: - Z-score associated with a 95% confidence interval is 1.96
- The standard error of the mean (SE) is $\frac{2.4}{\sqrt{81}} = 0.267$
- The margin of error (E) is $1.96 \times SE = 0.524$
- The confidence interval is 11.8% +- 0.524%

And so we can say that the 95% confidence interval for the mean dividend payout in this industry is (11.28%, 12.32%). We can be 95% confident that this interval will contain the mean dividend payout for this particular industry.

3.2.2 p-value

In your day to day data science work, you will often be required to explain the model's reliability and uncertainty, and the formal process of such is specified as something called the **test of significance**. Statistical significance often reference the **p-value**, a measure of the probability of obtaining a result equal to or more extreme than what was actually observed, assuming the null hypothesis is true.

Imagine a scenario where you're assigned to consult on Quicker, a startup that simplify and automate government grants application for newly incorporated startups. Through public announcements and official records, you find that the average duration for a newly incorporated startup to get its first government grant or financial funding is 215 days (with a population standard variance of 24 days, again through official records). Of the 35 entrepreneurs using Quicker platform, the average time is 178 days.

Discussion: Does this observation (178 days) deviate away from the population enough for it to be statistically significant? Use a 95% confidence interval

$H_0 : \mu = \mu_0$

$H_A : \mu < \mu_0$

The null hypothesis is that our new mean equals to the original mean, whereas the alternative hypothesis states that the mean of Quicker customers **is lower** than the original mean.

[135]:
```
# Your code here
```

Solution:
Recall what we've learned about the z-score, we can compute the p-value as follow:

[136]:
```
Z = (178-215)/24
p = st.norm.cdf(Z)

'{:,.4f}'.format(p)
```

[136]:
```
'0.0616'
```

While we can reject the null hypothesis at the 95% confidence level if our p-value is <= alpha (0.05), this is not the case here as our p-value is actually 0.0616. In this scenario, we fail to reject the null hypothesis and fail to proof that the Quicker platform did in fact led to a statistically significant reduction in government funding time.

3.2.3 T-Test

Generally, z-tests are used when we have a large enough sample size (rule of thumb is n >= 30) and when the population standard deviation is known. If the above conditions aren't met, we can instead use a statistical measurement known as the Student's t-test. Say grant automation platform has 10 users so far and that the population standard deviation (sigma) is not known.

[137]:
```
times = [184, 181, 230, 169, 158, 204, 220, 197, 219, 223]
```

To perform a t-test we can use groups of t-test functions from `scipy.stats`. Here we're interested in finding out whether we can reject the null hypothesis (that says there is no difference in government funding time for Quicker startups and other startups) in favor of the alternative hypothesis (that Quicker startup spend less time on average to acquire their first grant):

[138]:
```
from scipy import stats

stats.ttest_1samp(times, 215)
```

[138]:
```
Ttest_1sampResult(statistic=-2.104098191307367, pvalue=0.06468659586101468)
```

Since we use `ttest_1samp()`, it calculates the p-value to proof that the expected value (mean) of a sample of independent observations a is equal to the given population mean. The p-value is not smaller than 0.05 means we fail to reject the null hypothesis that the 2 population is equal.

There are multiple variation of the t-test function in scipy, namely a common one:
- `ttest_ind()`: Used two samples for the null hypothesis that 2 independent samples have identical average (expected) values
- `ttest_ind_from_stats()`: Used two samples mean, standard deviation, and number of observation for the null hypothesis that two independent samples have identical average (expected) values.
- `ttest_rel()`: Used 2 repeated samples that the samples have identical average (expected) values

Dive Deeper:

Recall our `hp` dataset, can you proof that Samsung phones that are sold in Batam are indeed significantly cheaper than the one sold in other cities?

[139]:
```
# Your code here
```

3.2.4 Power Analysis

In designing a hypothesis testing, or popularly known as A/B testing, one of its critical part is: determining the size of your sample. While significance test is widely used in the industry, namely a few:
- Testing new features that creates more engagement
- Selecting effective marketing strategy to generate more conversion
- Finding promotion best practice to increase revenue

The cost of evaluating these experiments is expensive for both time and resource. The role of data scientist, is to make sure the proposed design of an A/B testing has considered this resource constraint and also generate a conclusive result.

In some cases, drawing conclusion can lead to a fatal error due to wrong choice of sampling decision. Now let me introduce you to **2 types of error**:
- Type 1 error
- Type 2 error

In hypothesis test context, type 1 error occur when we reject a true null hypothesis while type 2 error occur when we fail to reject a false null hypothesis (a similar concept later on machine learning course!). Now what do we do with this information? Well a **higher statistical power** will result in a lower probability of failing to reject a false null hypothesis. In practice, a lower power will lead to wrong conclusion as stated in the following formula:

$Power = P(rejectH_0|H_ATrue)$
$Power = 1 - P(Error_2)$

Other related concept is **effect size**, that measure how strong a relationship between two variables. A somehow similar concept that we have learned is correlation. It is in fact not that much of a difference. In statistics analysis, the effect size is usually measured in three ways: (1) standardized mean difference, (2) odd ratio, (3) correlation coefficient.

Since we have learned about correlation coefficient, we can try to understand the idea of how does an effect size should be estimated.

Consider this:

You are designing an experiment for Tokopedia new homepage, you'd like to see if a certain promotional banner is placed in the second slot would lead to an ineffective campaign. In this case, say we accept the probability of type 2 error to be 15% and a significance level of 5%. The expected effect size is set to be 60%, means that you expect that the difference of redeemed promotion when put in the second slot would lead to 60% effect size. Try the below code and see what is the required sample size:

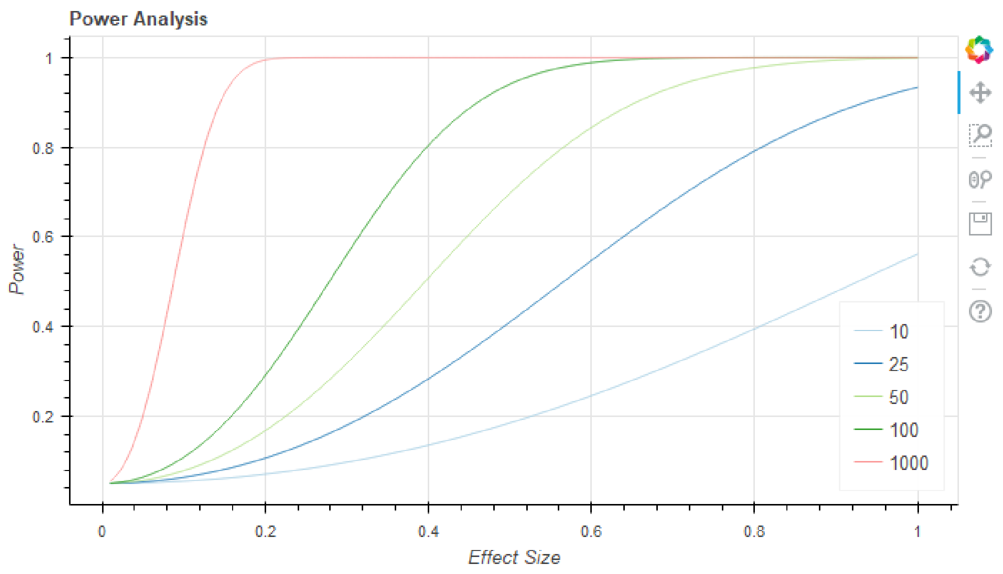
[140]:
```python
from statsmodels.stats.power import TTestIndPower

effect_size = 0.8
alpha = 0.05
power = 0.8

power_analysis = TTestIndPower()
sample_size = power_analysis.solve_power(effect_size = effect_size,
                                         power = power,
                                         alpha = alpha)

int(sample_size)
```

[140]: 25

Most times, you would generate a visualization to analyze the trade-off in selecting the parameters. Let's take a look at the following plot:

[141]:
```
nobs = [10, 25, 50, 100, 1000]
alpha = 0.05
effect_size = np.linspace(0.01, 1, 1000)
```

[142]:
```
from bokeh.transform import factor_cmap

p = figure(title='Power Analysis',
           plot_width=700,
           plot_height=400)

for n in nobs:
    power_line = []
    for i in effect_size:
        power = power_analysis.solve_power(effect_size=i,
                                           nobs1=n,
                                           alpha=alpha)
        power_line.append(power)
    p.line(effect_size, power_line,␣
 ↪color=all_palettes['Paired'][len(nobs)][nobs.index(n)], legend=str(n))


p.legend.location = 'bottom_right'
p.xaxis.axis_label = 'Effect Size'
p.yaxis.axis_label = 'Power'
show(p)
```



Looking at the above plot will give you an idea of how would the Power and sample size varies given different set of parameters. The information you gain in this analysis needs to correspond with the cost constraints of your experiment and the risk that came with it.

Discussion:

The plot above was created to understand the trade-off between power and effect size given a set of number of observations. The alpha, however, was assumed to be 5%. There are a common best practice we can use as the rule of thumbs on how to pick a value: for example 5% for significance level, and 80% for Cohen's d effect size measure.

Often times, it's always a good practice to see the plot in different axes such as power compared to number of observation given a set of acceptable effect size. Can you create the plot?