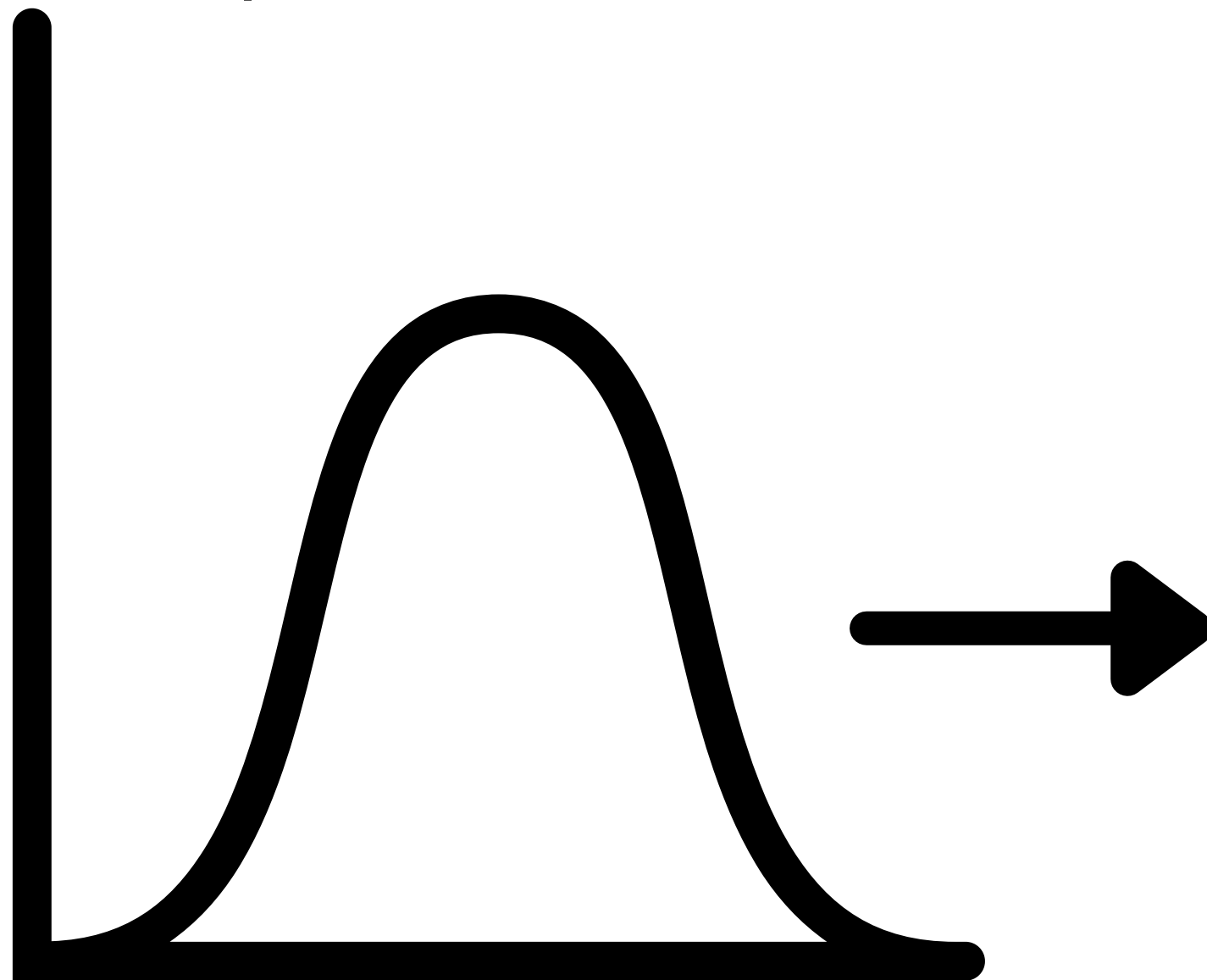


FGV QUANT

Kolmogorov - Smirnov Test

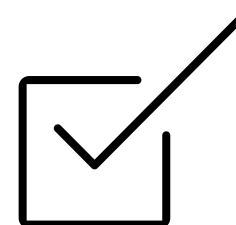
Um grande número de fenômenos naturais apresenta sua distribuição de probabilidade semelhante a uma curva de sino, nomeada normal



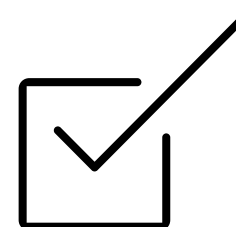
A função que descreve a curva pode ser descrita como:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{(x-\mu)}{\sigma}\right)^2}$$

onde x representa a média da amostra, sigma representa o desvio da mesma e μ a esperança matemática



O grupo se questionou se os retornos das ações **seguem uma distribuição normal** de probabilidade.

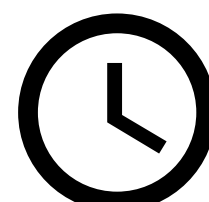


Para isso realizamos uma análise utilizando o teste estatístico de **Kolmogorov Smirnov**

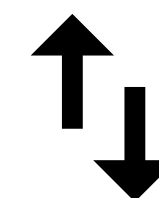
Parâmetros Iniciais



Empresas brasileiras



2017 - 2021



Retornos diários

99%

Grau de confiança

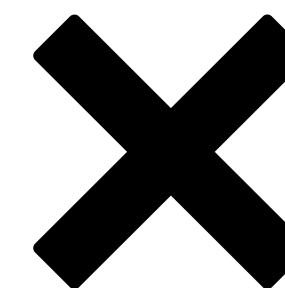
Diversos modelos financeiros assumem que os retornos das ações seguem uma distribuição normal. entre eles o **Value At Risk**, **Expected Shortfall**, ambos como medida de risco. **Modelo CAPM** para precificação do custo de capital.o **Beta** que mede o grau de relação entre um ativo e um índice

Consequências



retornos seguem uma distribuição normal

- A precificação de risco do mercado está correta
- A precificação do custo de capital da empresa está correta
- O grau de significância do Beta é conhecido e as equações derivadas do Beta tem relevância estatística conhecida



retornos não seguem uma distribuição normal

- A precificação de risco do mercado está incorreta
- A precificação do custo de capital da empresa está incorreta
- O grau de significância do Beta é conhecido e as equações derivadas do Beta não tem relevância estatística conhecida

1. Importar dados de **fechamento** histórico de certo ativo
2. Calcular **retornos diários**
3. Classificar do **menor para o maior** retorno e os ranquear
4. Calcular a **média, desvio padrão e número de observações** dos retornos
5. Calcular distribuição cumulativa **empírica** (rank / observações)
6. Calcular a **distribuição normal cumulativa** de cada retorno, usando a média e desvio
7. Calcular as **diferenças absolutas** entre os valores do passo 5 e 6
8. Encontrar o **Supremum** (máximo valor do passo 7)
9. Calcular **valor observado**: supremum * raiz(observações)
10. Calcular **valor Crítico**: $\text{raiz}(-\log(\alpha/2) * 0.5)$, sendo alfa: 100% - grau de confiança
11. Se o valor no passo 9 foi maior que o valor no passo 10 os retornos não são **normalmente distribuídos**, caso contrário, são.



1.0 Import Libraries

Import Relevant Libraries like Yfinance to download stock data, pandas to visualize dataframes and manipulate them, matplotlib to plot some distributions and values, and scipy stats to help us to calculate the normal distribution faster and numpy to log things

```
] import yfinance as yf
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import norm
import numpy as np
```

2.0 Download Stock Data

Using the Yahoo Finance Library to download stock data and calculate the returns in a new column

```
df=yf.download("^BVSP",start="2020-01-01",end="2020-12-12",progress=False)
df["Returns"]=df["Adj Close"].pct_change()
df
```

	Open	High	Low	Close	Adj Close	Volume	Returns
Date							
2020-01-02	115652.0	118573.0	115649.0	118573.0	118573.0	5162700	NaN
2020-01-03	118564.0	118792.0	117341.0	117707.0	117707.0	6834500	-0.007304
2020-01-06	117707.0	117707.0	116269.0	116878.0	116878.0	6570000	-0.007043
2020-01-07	116872.0	117076.0	115965.0	116662.0	116662.0	4854100	-0.001848
2020-01-08	116667.0	117335.0	115693.0	116247.0	116247.0	5910500	-0.003557

Exemplo capturando os dados do Ibovespa no ano de 2020



3.0 Sort & Rank Returns

Sort the Value by the minimum return to the maximum return and add a column with the Ranks

```
|: df=df.sort_values(by=['Returns'])
df=df.reset_index() # Rank
df.index=df.index+1
df=df.dropna()
df
```

```
|:
      Date    Open    High    Low    Close  Adj Close  Volume  Returns
1  2020-03-12  85103.0  85103.0  68488.0  72583.0    72583.0  12008700  -0.147797
2  2020-03-16  82565.0  82565.0  70855.0  71168.0    71168.0  12847800  -0.139215
3  2020-03-09  97982.0  97982.0  85880.0  86067.0    86067.0  14645500  -0.121738
4  2020-03-18  74576.0  74576.0  63547.0  66895.0    66895.0  16751500  -0.103488
5  2020-03-11  92202.0  92202.0  80796.0  85171.0    85171.0  11786900  -0.076377
```

4.0 Mean & Std of Returns

Calculating Mean, Standard Deviation and number of observations in our dataframe

```
mean=df["Returns"].mean()
std=df["Returns"].std()
number_observations=len(df["Adj Close"])
print(f"Mean: {(mean*100):.4f}% ")
print(f"Standard Deviation: {(std*100):.4f}% ")
print(f"Number of Observations: {number_observations}")
```

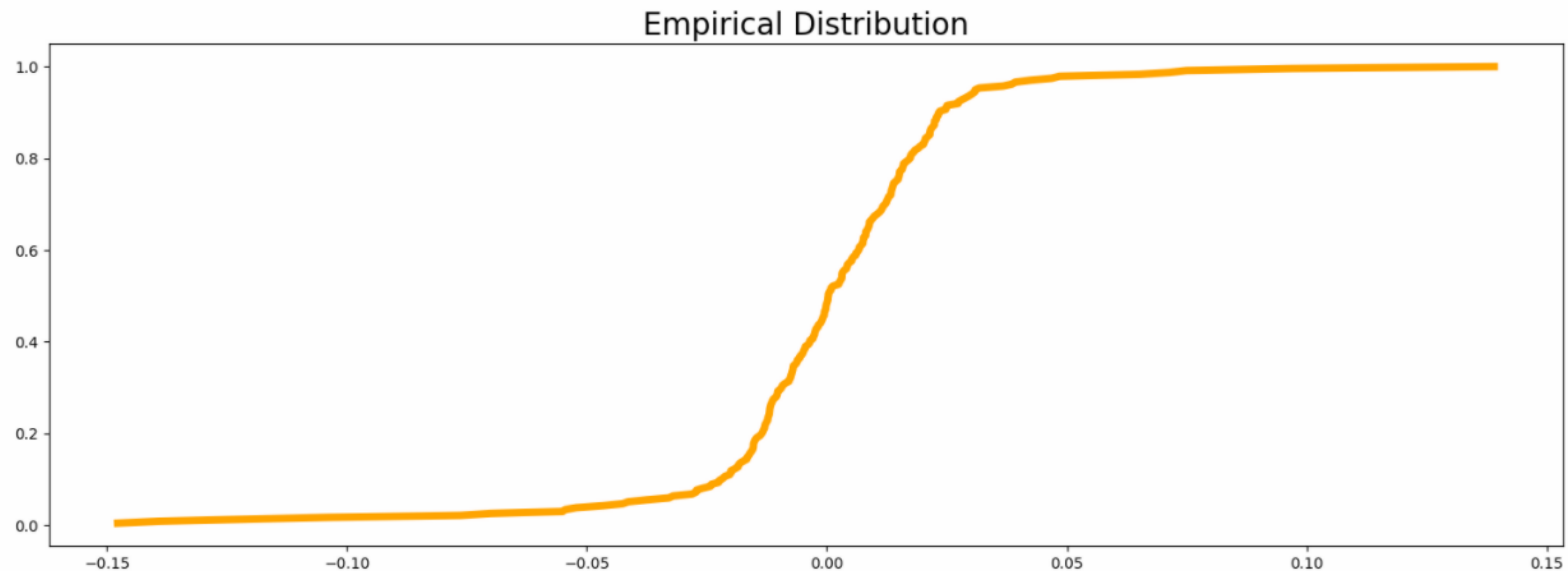
```
Mean: 0.0303%
Standard Deviation: 2.8752%
Number of Observations: 236
```

5.0 Empirical Distribution

Plotting what is the expected formation of the distribution to be consider normally distributed

```
plt.style.use("default")
plt.figure(figsize=(18,6))
plt.title("Empirical Distribution",fontsize=20)
df["Empirical_Distribution"]=df.index/number_observations
plt.plot(df["Returns"],df["Empirical_Distribution"],color="Orange",linewidth=5)
```

```
[<matplotlib.lines.Line2D at 0x299f1b83ac0>]
```

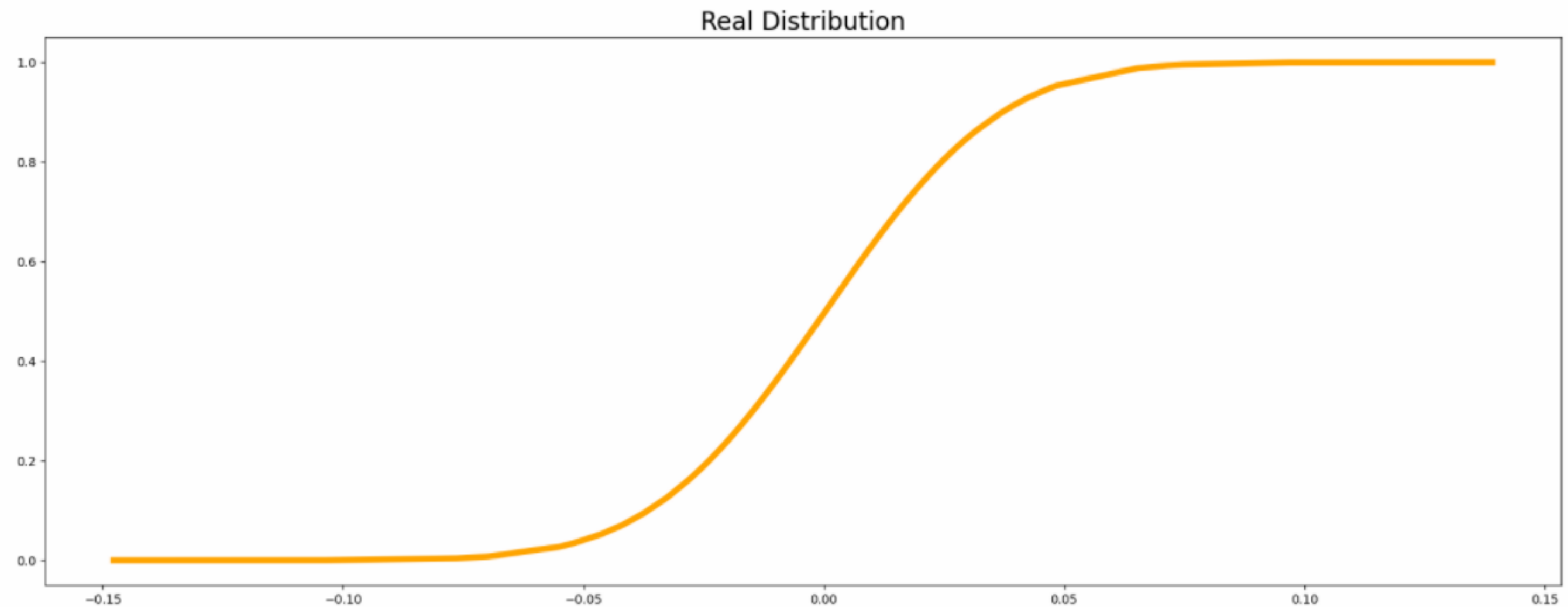


6.0 Real Distribution

Plotting what is the Real formation of the distribution

```
df["Real_Distribution"]=norm.cdf(df["Returns"],mean,std) # Cumulative Normal Distribution  
plt.style.use("default")  
plt.figure(figsize=(22,8))  
plt.title("Real Distribution",fontsize=20)  
plt.plot(df["Returns"],df["Real_Distribution"],color="orange",linewidth=5)
```

```
[<matplotlib.lines.Line2D at 0x299f1bf30d0>]
```

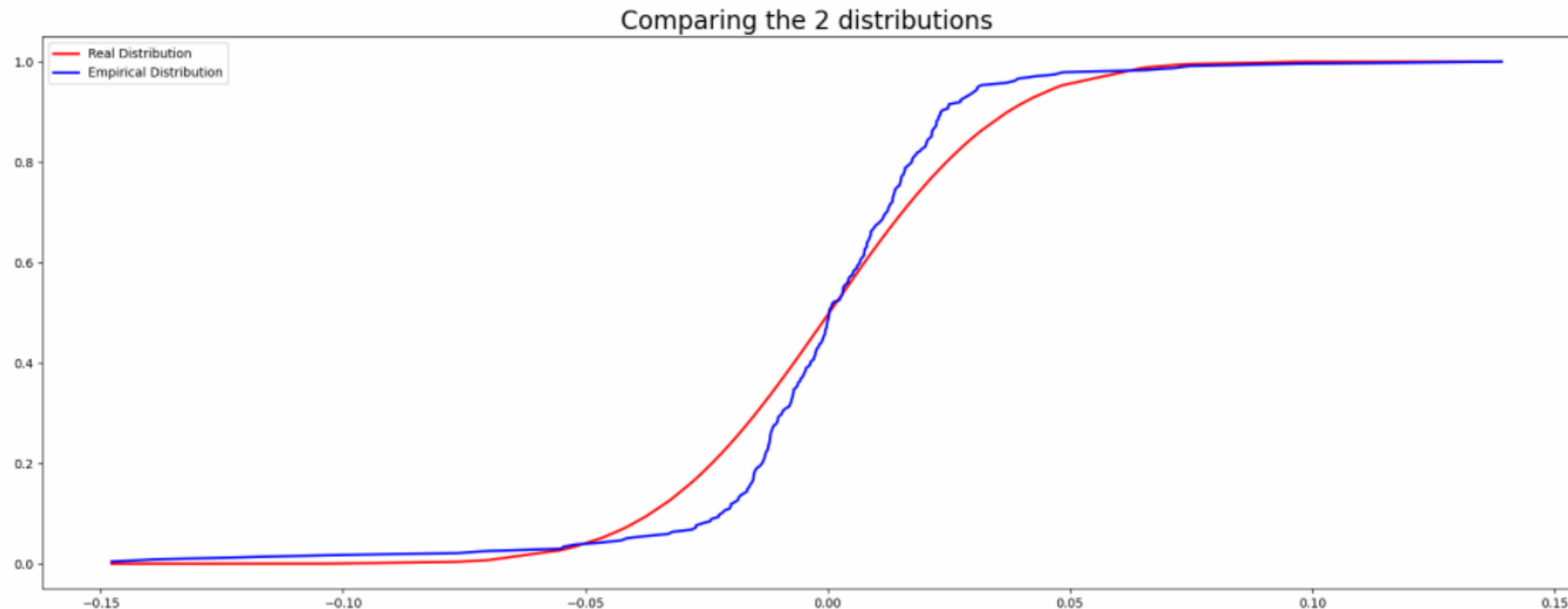


7.0 Plot Real & Empirical Distributions

Comparing the real distribution find by the values downloaded by the empirical distribution calculated by the "expected" curve formation following the Normal Distribution

```
plt.figure(figsize=(22,8))  
plt.title("Comparing the 2 distributions",fontsize=20)  
plt.plot(df["Returns"],df["Real_Distribution"],color="red",linewidth=2,label="Real Distribution")  
plt.plot(df["Returns"],df["Empirical_Distribution"],color="blue",linewidth=2,label="Empirical Distribution")  
plt.legend()
```

<matplotlib.legend.Legend at 0x299f1c173d0>



8.0 Differences between Empirical & Real Distribution

Calculating the absolute differences between empirical and real distribution

```
df["Difference"] = abs(df["Empirical_Distribution"] - df["Real_Distribution"])
df
```

	Date	Open	High	Low	Close	Adj Close	Volume	Returns	Empirical_Distribution	Real_Distribution	Difference
1	2020-03-12	85103.0	85103.0	68488.0	72583.0	72583.0	12008700	-0.147797	0.004237	1.295873e-07	4.237159e-03
2	2020-03-16	82565.0	82565.0	70855.0	71168.0	71168.0	12847800	-0.139215	0.008475	6.096877e-07	8.473967e-03
3	2020-03-09	97982.0	97982.0	85880.0	86067.0	86067.0	14645500	-0.121738	0.012712	1.094767e-05	1.270092e-02
4	2020-03-18	74576.0	74576.0	63547.0	66895.0	66895.0	16751500	-0.103488	0.016949	1.531611e-04	1.679599e-02
5	2020-03-11	92202.0	92202.0	80796.0	85171.0	85171.0	11786900	-0.076377	0.021186	3.827270e-03	1.735917e-02

9.0 Supremum

Supremum is the max absolute difference between the real and the empirical distributions

```
supremum = max(df["Difference"])
print(f"Supremum: {supremum*100:.4f}%")
```

Supremum: 13.3649%

10.0 Kolmogorov-Smirnov Statistic

Using the Kolmogorov-Smirnov Statistic too find the observed value

```
: observed_value = supremum * np.sqrt((number_observations))
observed_value
```

```
: 2.0531540429992585
```

11.0 Results Analysis

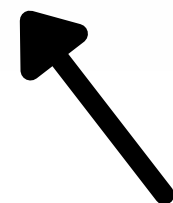
The confidence used in the test is 1%, so I want to test if the observed value found above is significant to prove that my distribution is normal or not

```
confidence=0.01
critical_value=np.sqrt(-np.log(confidence/2)*0.5)
critical_value
```



1.6276236307187293

```
if observed_value>critical_value:
    print("Not Normally Distributed")
if observed_value<=critical_value:
    print("Normally Distributed!")
p_value=np.exp(-supremum**2*number_observations)
print(f"P-Value: {p_value:.7f}")
```

Not Normally Distributed
P-Value: 0.0147658



No exemplo do Ibovespa no ano de 2020 percebe-se que os retornos não seguem uma distribuição normal

 Normalmente distribuidos
 Não normalmente distribuido



PETR4



VALE3



ITUB4



BBDC4



ABEV3



IBOVESPA



✓ Normalmente distribuídos

✗ Não normalmente distribuído



PETR4



VALE3



ITUB4



BBDC4



ambev

ABEV3



IBOVESPA



✓ Normalmente distribuídos

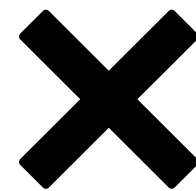
✗ Não normalmente distribuído



PETR4



VALE3



ITUB4



BBDC4



ambev

ABEV3



IBOVESPA



✓ Normalmente distribuidos
✗ Não normalmente distribuido



PETR4



VALE3



ITUB4



BBDC4



ABEV3



IBOVESPA



✓ Normalmente distribuídos

✗ Não normalmente distribuído



PETR4



VALE3



ITUB4



BBDC4



ambev

ABEV3



IBOVESPA

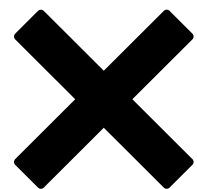


✓ Normalmente distribuídos

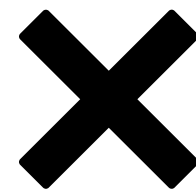
✗ Não normalmente distribuído



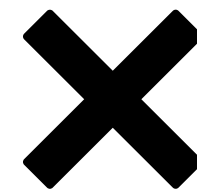
PETR4



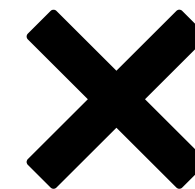
VALE3



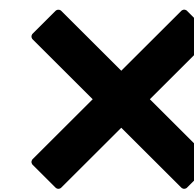
ITUB4



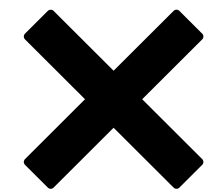
BBDC4



ABEV3



IBOVESPA

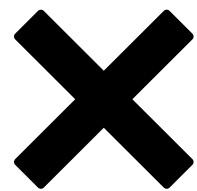


✓ Normalmente distribuídos

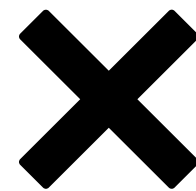
✗ Não normalmente distribuído



PETR4



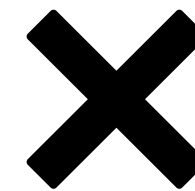
VALE3



ITUB4

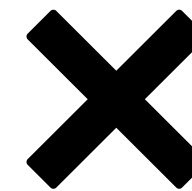


BBDC4

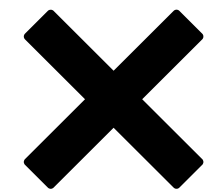


ambev

ABEV3



IBOVESPA

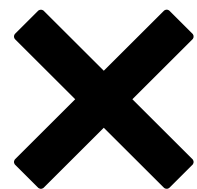


✓ Normalmente distribuídos

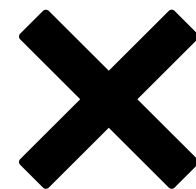
✗ Não normalmente distribuído



PETR4



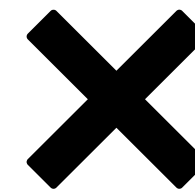
VALE3



ITUB4

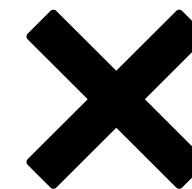


BBDC4

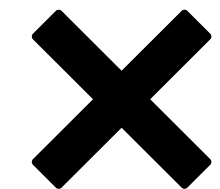


ambev

ABEV3



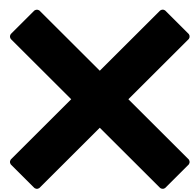
IBOVESPA



✓ Normalmente distribuidos
✗ Não normalmente distribuido



PETR4



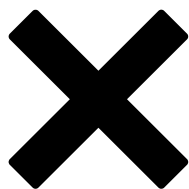
VALE3



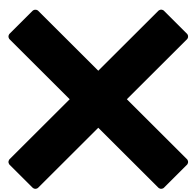
ITUB4



BBDC4



ABEV3



IBOVESPA



Ao se analisar o intervalo anual dos ativos a grande maioria se comporta de maneira normal na maioria dos casos, entretanto em anos atípicos com o de 2020 os retornos dos ativos tendem a se afastar de uma distribuição normal no ano. Ao se analisar intervalos maiores percebe-se que a grande maioria dos casos apresentou retornos que diferem de uma curva normal.