Skewness Demonstration

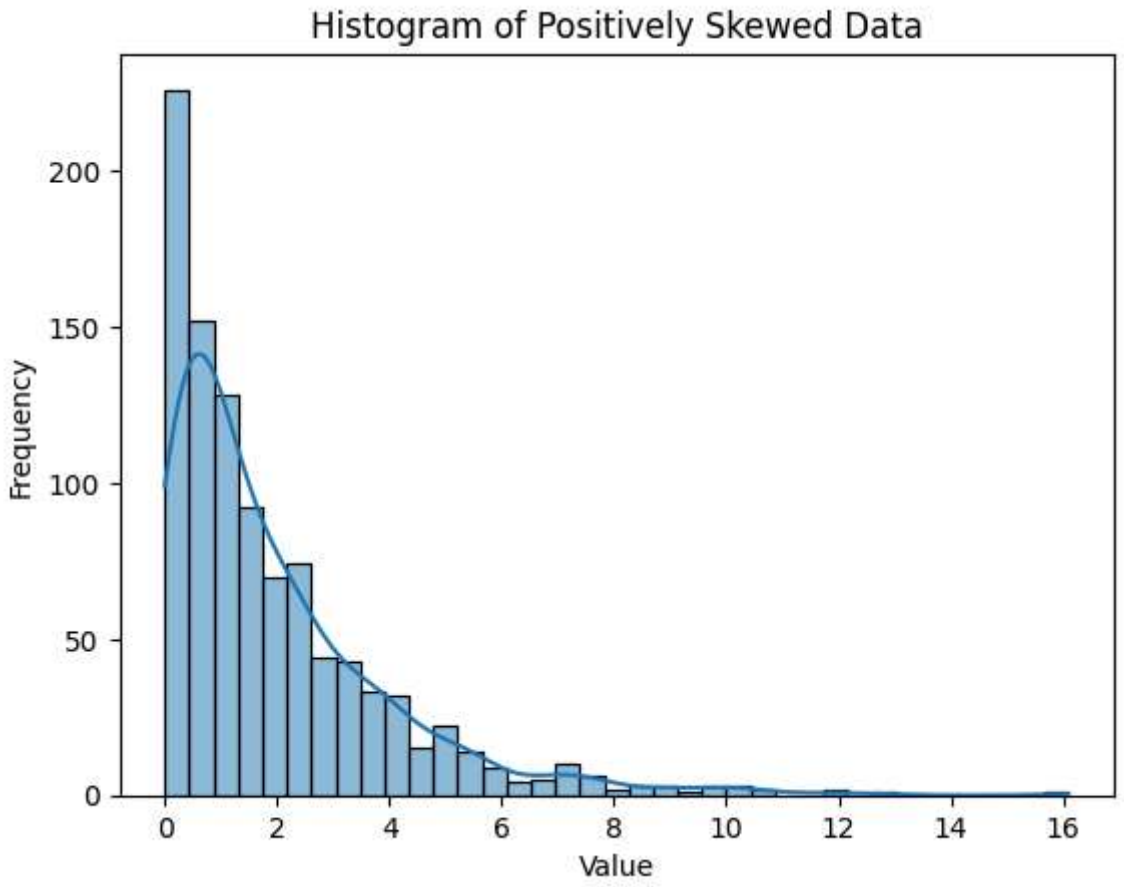
Definition Skewness measures the deviation of a dataset distribution compared to the normal distribution. A normal distribution would have 0 skewness. The skewness of a distribution is defined as: Skewness = 3X-μ σ

Where: X is the mean, μ is the median, and σ is the standard deviation.

Description Skewness indicates the asymmetry of the data distribution. Positive skewness means a longer or fatter tail on the right side, and negative skewness means a longer or fatter tail on the left side.

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import skew
import pandas as pd
import yfinance as yf
import scipy.stats as stats
```

```
# Generating positively skewed data
data = np.random.exponential(scale=2, size=1000)

# Plotting the histogram
sns.histplot(data, kde=True)
plt.title('Histogram of Positively Skewed Data')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```



This histogram has a KDE (Kernel Density Estimate) overlay showing the positively skewed distribution.

```
# Calculating and printing skewness
print("Skewness:", skew(data))
```

```
Skewness: 2.7080694409667
```

As expected after seeing the histogram, the skewness of the dataset is significantly higher than 0.

Diagnosis

There are 2 quick ways to recognize or test for skewness in a dataset, the first would be by checking the data visually by creating a histogram or box plot of the dataset. The second one would be by performing statistical tests by calculating the skewness coefficient using scipy.stats.skew(data) in Python. A skewness coefficient significantly lower or higher than zero indicates skewness.

Damage

Skewed data can cause several issues in statistical modelling and data analysis: Biased Parameter Estimates: Many statistical models assume normality, and skewness can lead to biased parameter estimates and predictions. Misleading Descriptive Statistics: Measures like the mean may not accurately represent the central tendency in skewed distributions, leading to incorrect conclusions.

Directions

We can perform transformations to reduce skewness like log, square root or box-cox transformations. Or we can use robust statistical measures like using the median instead of the mean as it is less affected by skewness and employing robust regression

techniques that are less sensitive to skewed data, such as quantile regression. By implementing these strategies, we can mitigate the impact of skewness on our model and data analysis, leading to more reliable and interpretable results.

Kurtosis Demonstration

To carry out an analysis of kurtosis, real data of Apple Inc. stock daily stock was extracted from yahoo finance for a period of 52 months (1st Jan 2020 to 1st May 2024)

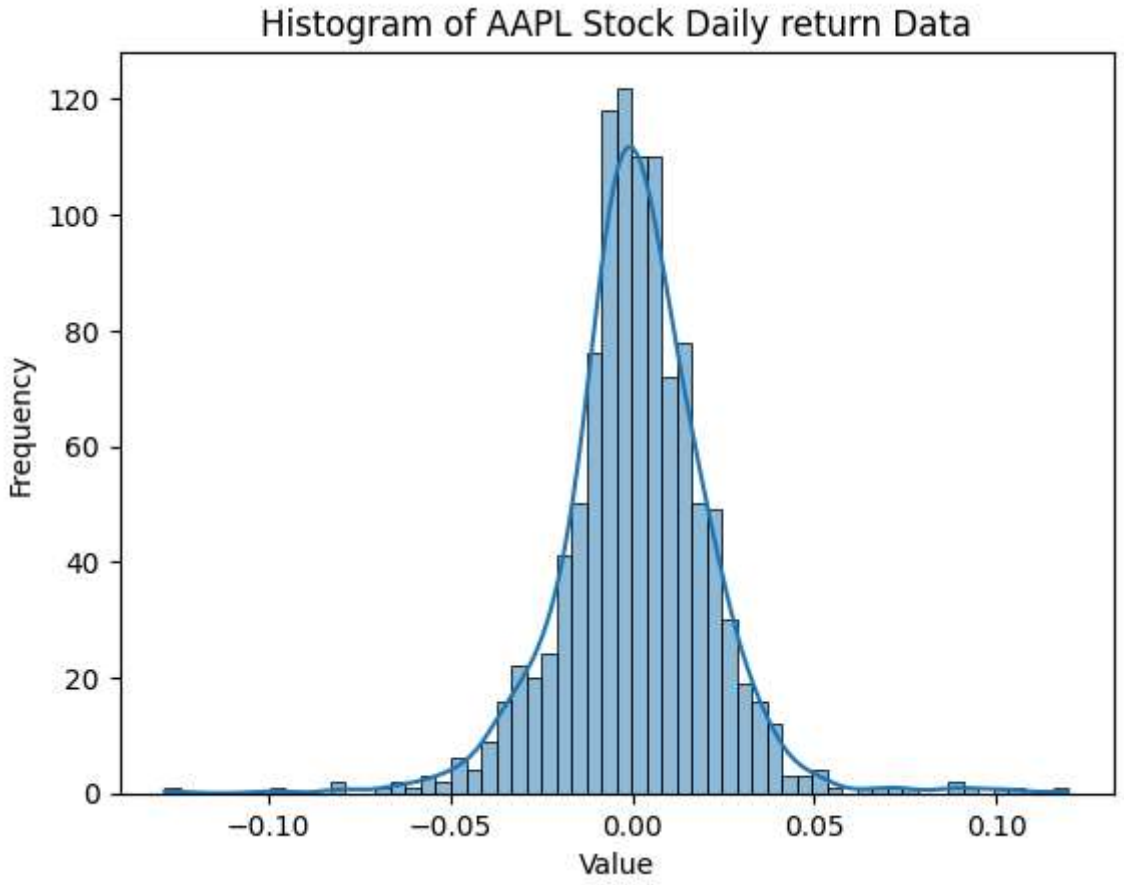
```
#Importation of Real Data for Analysis
ticker = "AAPL"
start_date = "2020-01-01"
end_date = "2024-05-01"
Data = yf.download(ticker, start = start_date, end = end_date)
```

```
[*********************100%%**********************]  1 of 1 completed
```

```
#Computation of Kurtosis

Data["Daily Return"] = Data["Adj Close"].pct_change().dropna()

kurtosis_value = stats.kurtosis(Data["Daily Return"].dropna())

print(f"Kurtosis of {ticker} daily returns: {kurtosis_value}")
```

```
Kurtosis of AAPL daily returns: 5.000449730157122
```

```
#Visualisation of kurtosis data
sns.histplot(Data["Daily Return"], kde = True)
plt.title(f"Histogram of {ticker} Stock Daily return Data")
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```


Histogram of AAPL Stock Daily return Data
Frequency
Value
120
100
80
60
40
20
0
−0.10
−0.05
0.00
0.05
0.10

Diagnosis

To determine if kurtosis exists in a dataset, calculate the fourth standardized moment using the formula above. A kurtosis value greater than 3 indicates a leptokurtic distribution (heavy tails and a sharp peak), while a value less than 3 indicates a platykurtic distribution (light tails and a flatter peak) (DeCarlo, 1997).

Damage

Ignoring kurtosis can lead to underestimating the probability of extreme events, which can have severe consequences in fields such as finance, where risk assessment is crucial (Xiong & Idzorek, 2011).

Directions

To address kurtosis in data analysis, it's best we use robust statistical methods that are less sensitive to outliers, such as the median and interquartile range instead of the mean and standard deviation. Additionally, using fat-tailed distributions, such as the Student's t-distribution or the generalized extreme value distribution, can better capture the presence of kurtosis in the data (Westfall, 2014).

Over-reliance on the Gaussian distribution

Definition: A Gaussian distribution, also known as a normal distribution, is defined by the probability density function (PDF):

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

where μ is the mean, σ is the standard deviation, and x represents the variable.

Description: Over-reliance on the Gaussian distribution in finance means assuming that asset returns follow a normal distribution, which often underestimates the probability of extreme events (fat tails) and leads to mispricing of risk and poor risk management.

Let's demonstrate this concept with the S&P 500 data.

```
!pip install copulas
```

```
Collecting copulas
  Downloading copulas-0.11.0-py3-none-any.whl (51 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 51.9/51.9 kB 939.9 kB/s eta 0:00:00
Requirement already satisfied: plotly>=5.10.0 in /usr/local/lib/python3.10/dist-packages (from copulas) (5.15.0)
Requirement already satisfied: pandas>=1.3.4 in /usr/local/lib/python3.10/dist-packages (from copulas) (2.0.3)
Requirement already satisfied: numpy>=1.23.3 in /usr/local/lib/python3.10/dist-packages (from copulas) (1.25.2)
Requirement already satisfied: scipy>=1.9.2 in /usr/local/lib/python3.10/dist-packages (from copulas) (1.11.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.3.4->c
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.3.4->copulas) (2
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.3.4->copulas)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from plotly>=5.10.0->copulas
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from plotly>=5.10.0->copulas) (24.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas
Installing collected packages: copulas
```

```
Successfully installed copulas-0.11.0
```

```
import yfinance as yf
import pandas as pd
import numpy as np
import seaborn as sns
import statsmodels.api as sm
import plotly.express as px
import plotly.figure_factory as ff
from copulas.multivariate import GaussianMultivariate
import matplotlib.pyplot as plt
from scipy.stats import t, norm, gamma, beta
import datetime
```

Real data of SP500. stock daily stock was extracted from yahoo finance for a period of 57 months (1st Jan 2018 to 1st Sept 2022)

```
start = '2018-01-01'
end = '2022-09-01'

# Pull data from Yahoo Finance using yfinance
data = yf.download("^GSPC", start=start, end=end)
data_set = data[["Adj Close"]].rename(columns={"Adj Close": "SP500"})

# Verify the data
print(data_set.head())
```

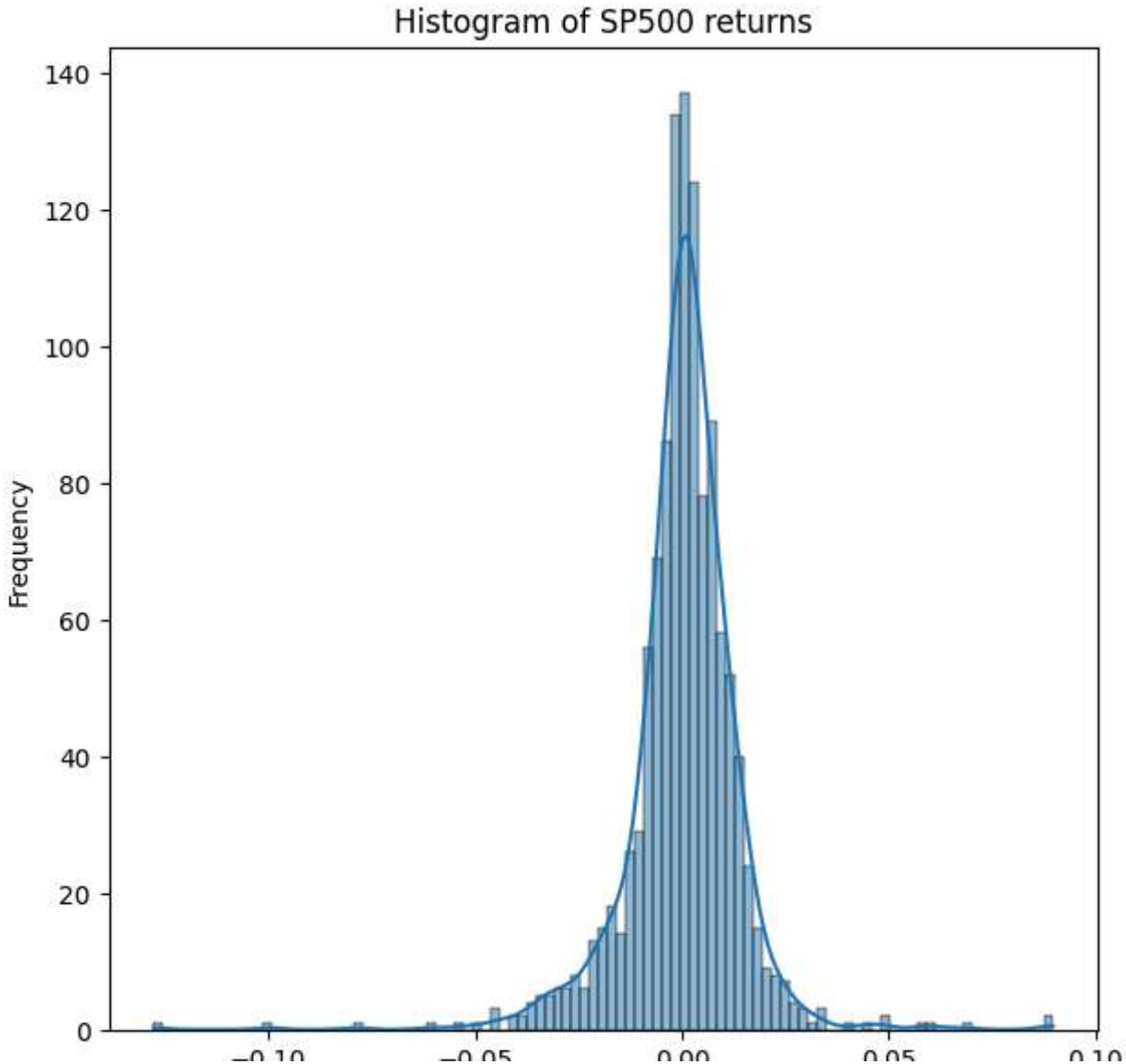
```
[*********************100%%**********************]  1 of 1 completed
                SP500
Date
```

```
2018-01-02  2695.810059
2018-01-03  2713.060059
2018-01-04  2723.989990
2018-01-05  2743.149902
2018-01-08  2747.709961
```

```
# Calculate daily returns
data_set["SP500_R"] = np.log(data_set["SP500"]) - np.log(data_set["SP500"].shift(1))
data_set = data_set.dropna()
y = data_set["SP500_R"]

# Verify the returns data
print(y.head())
```

```
Date
2018-01-03    0.006378
2018-01-04    0.004021
2018-01-05    0.007009
2018-01-08    0.001661
2018-01-09    0.001302
Name: SP500_R, dtype: float64
```

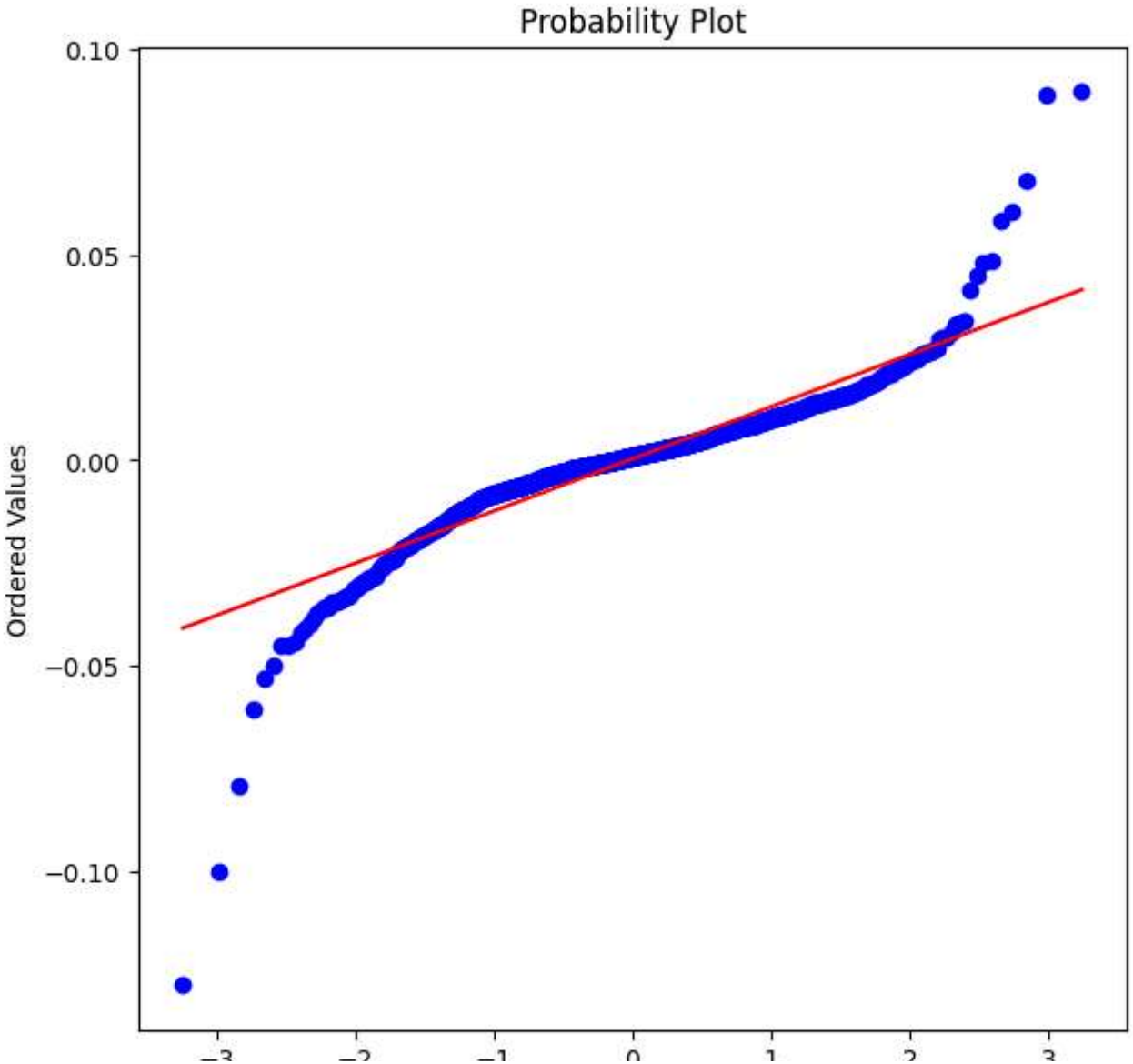
```
# Plot histogram of SP500 returns
plt.figure(figsize=(7, 7))
plt.xlabel("Returns")
plt.ylabel("Frequency")
plt.title("Histogram of SP500 returns")
sns.histplot(y, kde=True)
plt.show()
```

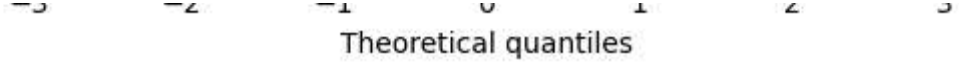

Histogram of SP500 returns
Frequency
140
120
100
80
60
40
20
0
−0.10
−0.05
0.00
0.05
0.10

−0.10 −0.05 0.00 0.05 0.10

Returns

```
#Generate a Q-Q plot to assess the normality of the returns.
import pylab
from scipy import stats
plt.figure(figsize=(7, 7))
stats.probplot(y, dist="norm", plot=pylab)
pylab.show()
```


Probability Plot
Ordered Values
0.10
0.05
0.00
−0.05
−0.10
−3
−2
−1
0
1
2
3



Here our SP500 returns do seam to be normally distributed. But those tails do cause some concern and hence we need to investigate further.

```
# Perform Shapiro-Wilk normality test
shapiro_test = stats.shapiro(y)
print(f'Shapiro-Wilk test statistic: {shapiro_test.statistic}, p-value: {shapiro_test.pvalue}')

shapiro_test = stats.shapiro(data_set["SP500_R"][1:])
shapiro_test
```

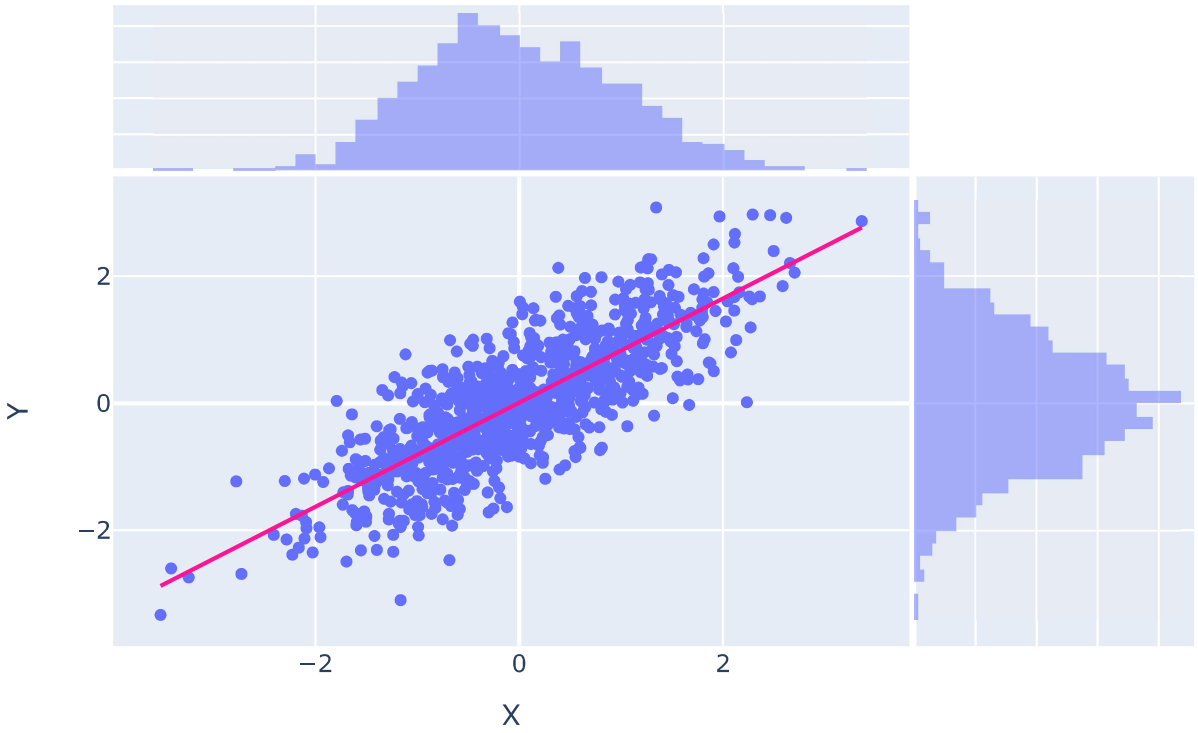
```
Shapiro-Wilk test statistic: 0.8588029742240906, p-value: 3.537216491384221e-31
ShapiroResult(statistic=0.8588845133781433, pvalue=3.6957676396208713e-31)
```

Our Shapiro test failed and hence our data is not normally distributed. Meaning our next option here could be to use the Box and Cox method of transforming non-normal data to normal data. But that does require more work

```
np.random.seed(seed=5)
mean = [0,0]
rho = 0.8
cov = [[1,rho],[rho,1]] # diagonal covariance, points lie on x or y-axis

norm_1,norm_2 = np.random.multivariate_normal(mean,cov,1000).T
unif_1 = norm.cdf(norm_1)
unif_2 = norm.cdf(norm_2)
norm_data = pd.concat([pd.DataFrame(norm_1), pd.DataFrame(norm_2)], axis=1)
norm_data.columns = ['X', 'Y']
norm_data.corr()
```

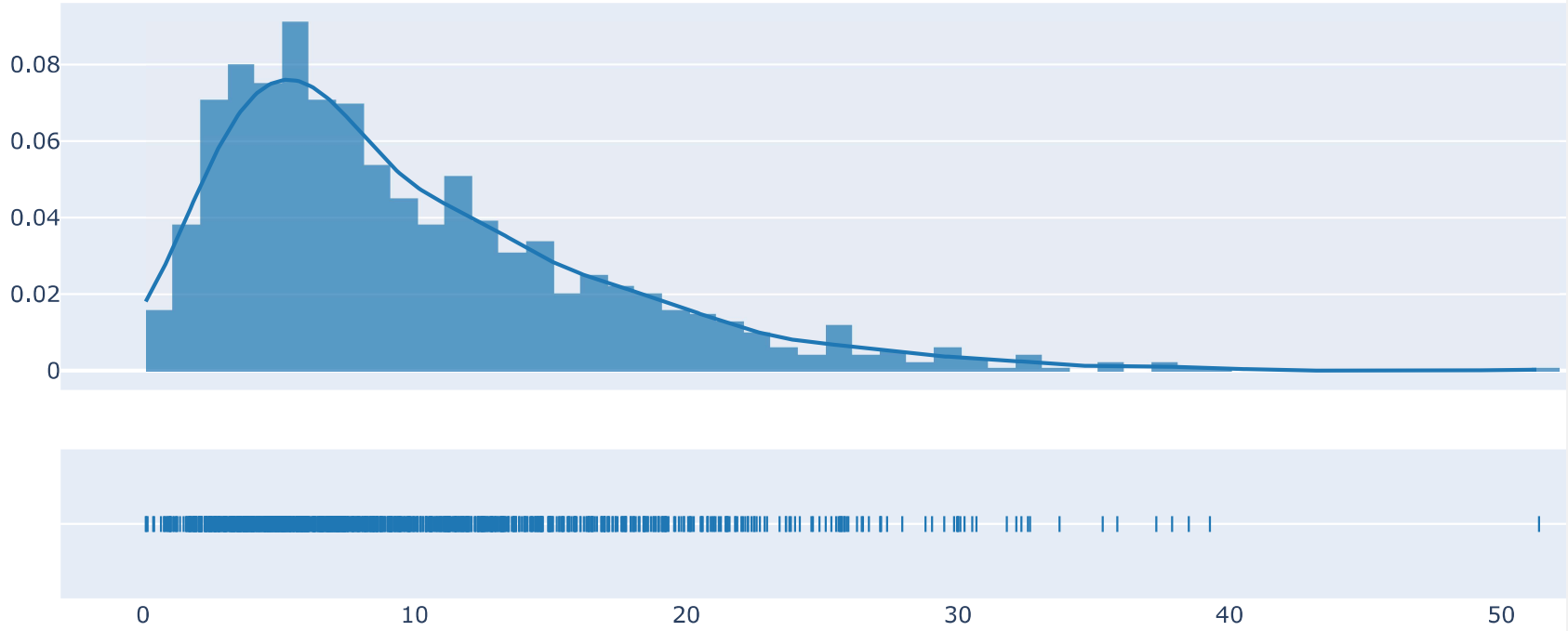
	X	Y
X	1.000000	0.803551
Y	0.803551	1.000000

```
fig = px.scatter(norm_data, x = 'X', y='Y', width=700, height=500, trendline='ols', trendline_color_override='DeepPink', ma
fig.show()
```

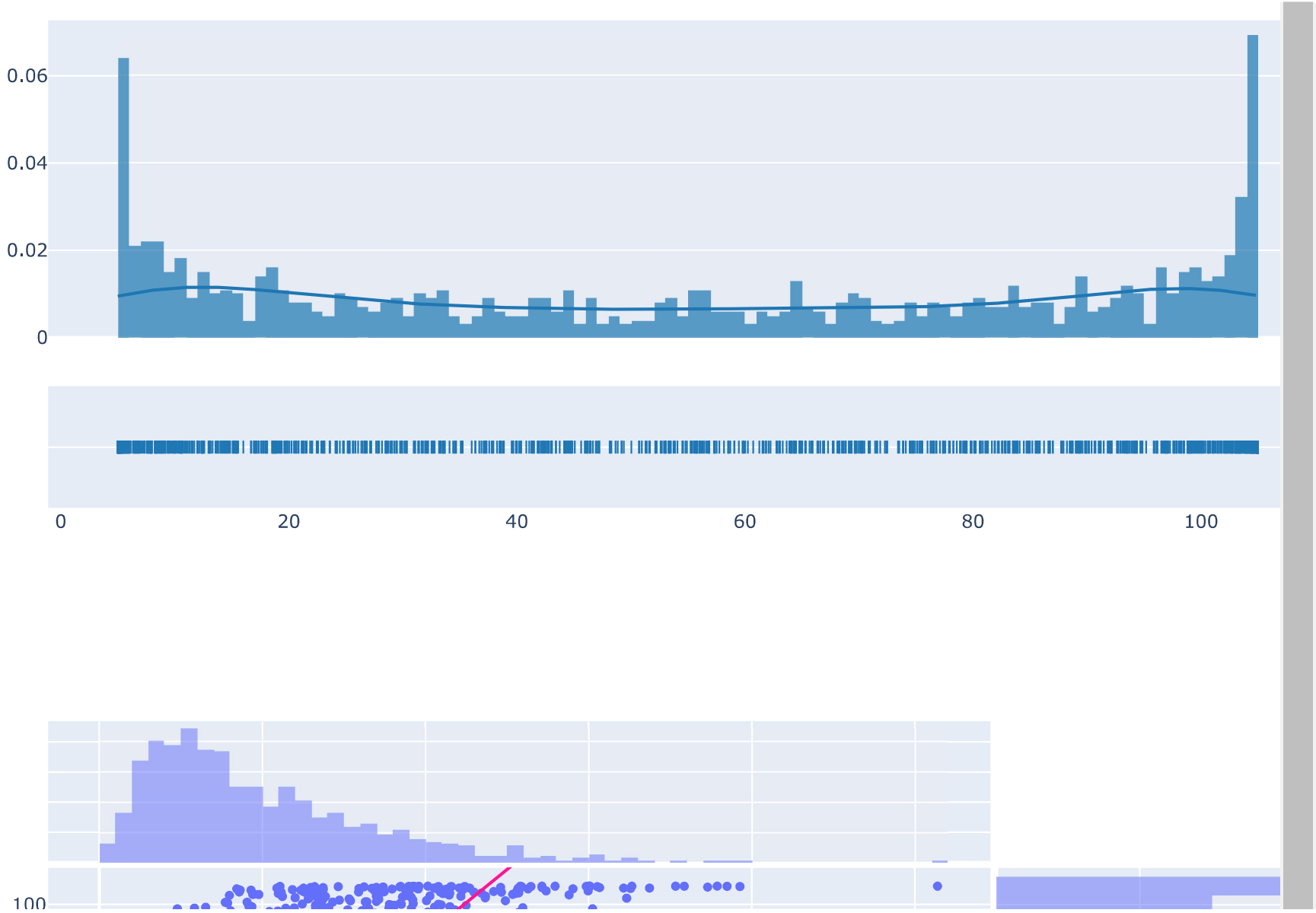

Bi-Variate Normal
2
0
−2
Y
−2
0
2
X

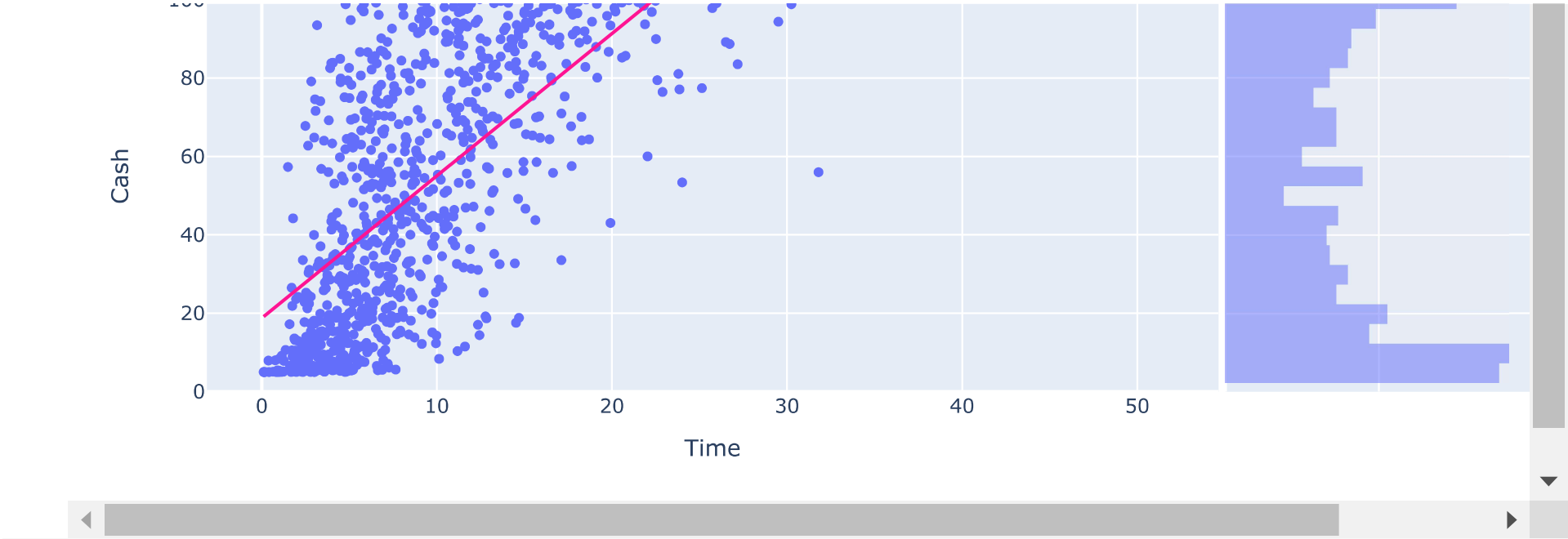
```
# Generate non-normal data using Gamma and Beta distributions
website_time = pd.DataFrame(gamma.ppf(unif_1, a=2, scale=5))
website_spend = pd.DataFrame(beta.ppf(unif_2, a=0.5, b=0.5, loc=5, scale=100))

join_time_spend = pd.concat([website_time, website_spend], axis=1)
join_time_spend.columns = ['Time', 'Cash']

# Plot distribution of time spent on website
gamma_dist = ff.create_distplot([website_time.values.reshape(-1)], group_labels=[' '])
gamma_dist.update_layout(showlegend=False, title_text='Time Spent on Website', width=1000, height=500)
gamma_dist.show()

# Plot distribution of dollars spent on website
t_dist = ff.create_distplot([website_spend.values.reshape(-1)], group_labels=[' '])
t_dist.update_layout(showlegend=False, title_text='Dollars Spent on Website', width=1000, height=500)
t_dist.show()

# Scatter plot of time spent on website vs dollars spent
fig = px.scatter(join_time_spend, x='Time', y='Cash', width=1000, height=500, range_y=[0, 110], trendline='ols', trendline_
fig.show()
```

Time Spent on Website

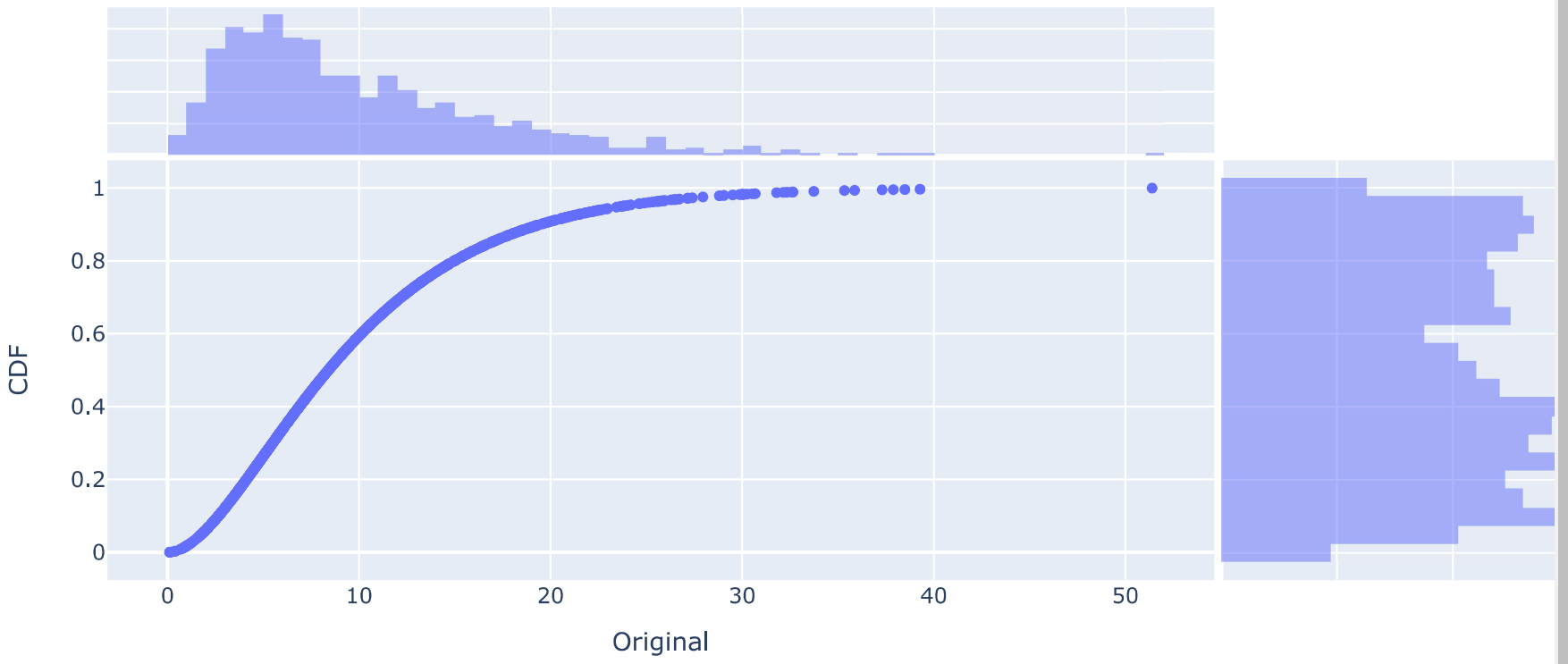


Dollars Spent on Website

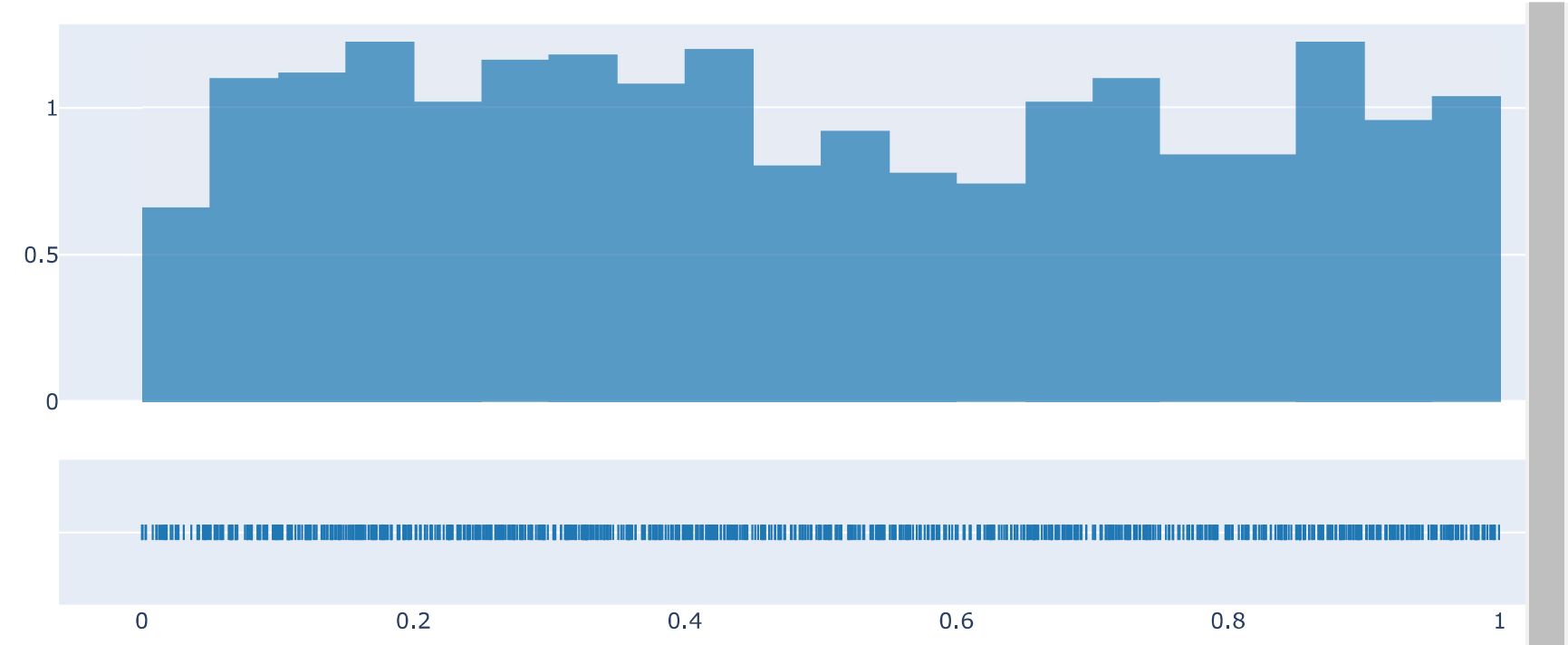

0.06
0.04
0.02
0
0
20
40
60
80
100
100


Cash
Time
0
20
40
60
80
0
10
20
30
40
50

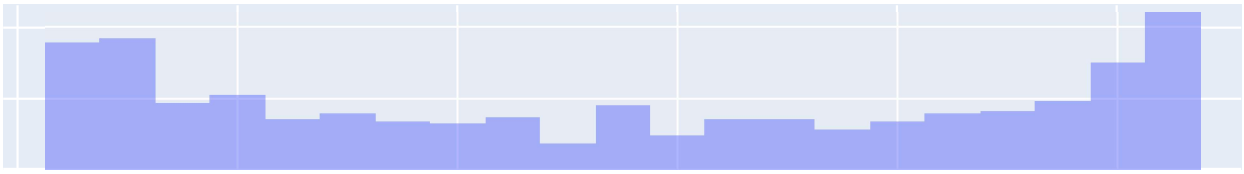
```
# Transform non-normal data using CDF
time_cdf = pd.DataFrame(gamma.cdf(website_time.values, a=2, scale=5))
time_cdf_vs_original = pd.concat([time_cdf, website_time], axis=1)
time_cdf_vs_original.columns = ['CDF', 'Original']

time_cdf_vs_original_plot = px.scatter(time_cdf_vs_original, x='Original', y='CDF', width=1000, height=500, title='Gamma Cu
time_cdf_vs_original_plot.show()

# Plot transformed time data
time_cdf_plot = ff.create_distplot([time_cdf.values.reshape(-1)], group_labels=[' '], show_curve=False, bin_size=0.05)
time_cdf_plot.update_layout(showlegend=False, title_text='Uniform distribution of time spent on site', width=1000, height=5
time_cdf_plot.show()

# Transform dollars spent data using CDF
dollar_cdf = pd.DataFrame(beta.cdf(website_spend.values, a=0.5, b=0.5, loc=5, scale=100))
dollar_cdf_vs_original = pd.concat([dollar_cdf, website_spend], axis=1)
dollar_cdf_vs_original.columns = ['CDF', 'Original']

dollar_cdf_vs_original_plot = px.scatter(dollar_cdf_vs_original, x='Original', y='CDF', width=1000, height=500, title='Beta
dollar_cdf_vs_original_plot.show()
```

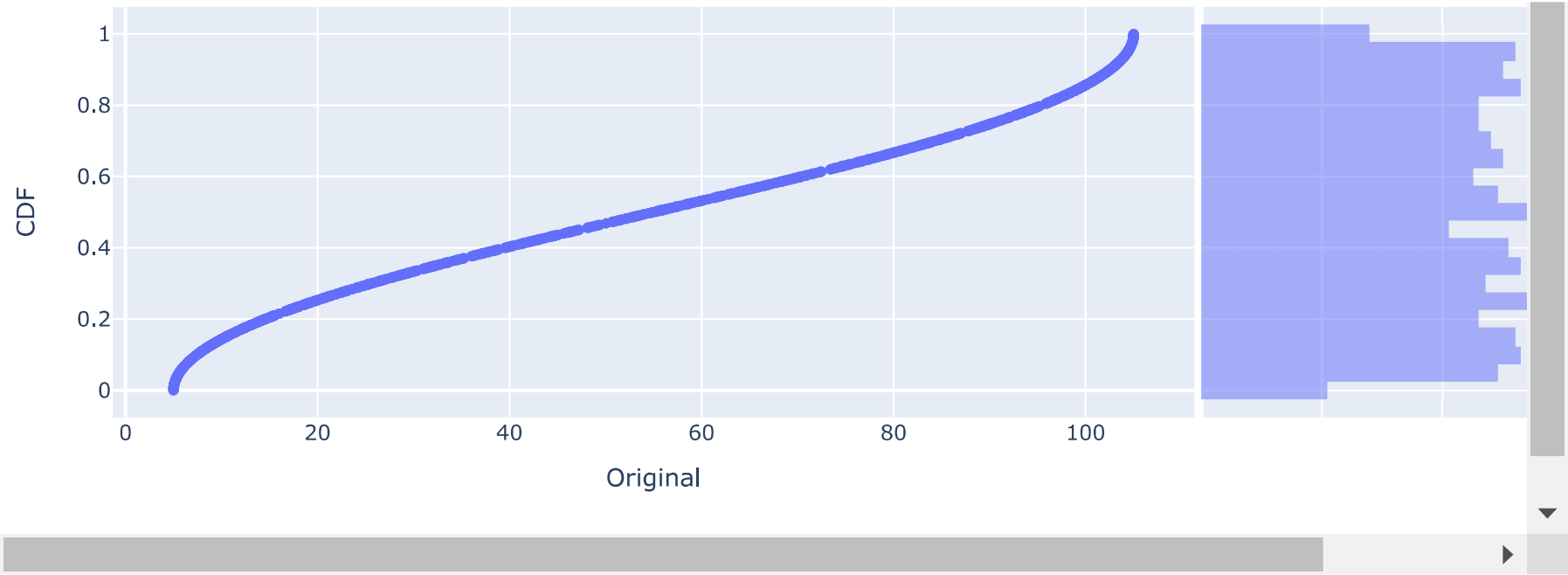


Uniform distribution of time spent on site

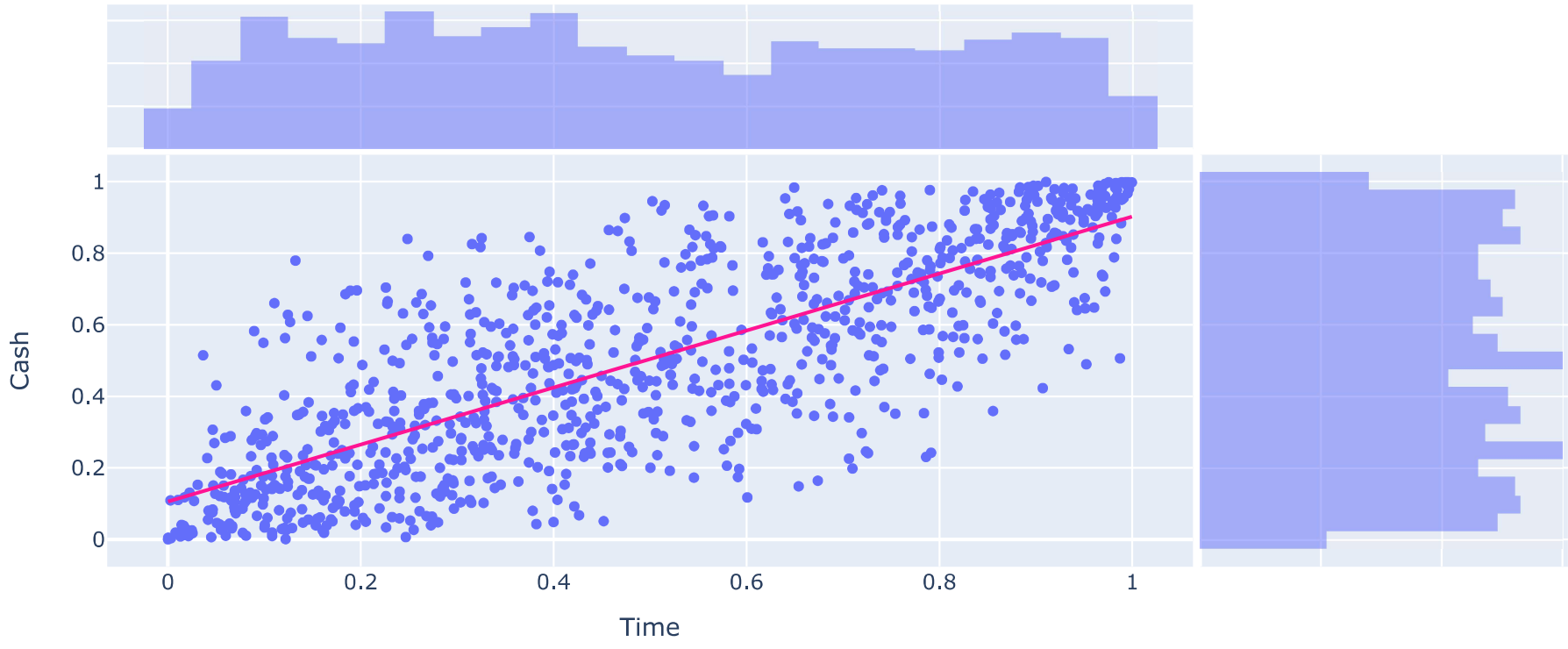


Beta Cumulative Distribution Function for Dollars Spent






```
# Combine transformed data for analysis
join_time_spend_uniform = pd.concat([time_cdf, dollar_cdf], axis=1)
join_time_spend_uniform.columns = ['Time', 'Cash']

# Plot transformed data in uniform space
time_v_money_uniform = px.scatter(join_time_spend_uniform, x='Time', y='Cash', width=1000, height=500, marginal_x='histogra
time_v_money_uniform.show()
```



Now if we are to hover our mouse on the red line in the first scatter plot we get an R-squared value of about 0.52. Now if we are to do the same here we get an R-squared of about 0.62. This means by transforming our non-normal data we have managed to fined a line

of best fit and form some sort of relationship in our example.

Diagnosis

To recognize if data deviates from normality:

Visual Inspection: Use histograms and Q-Q plots to visually inspect the shape and tail behavior of the data. Statistical Tests: Perform tests like the Shapiro-Wilk test, Kolmogorov-Smirnov test, or Anderson-Darling test for more formal diagnosis.

Damage:

Relying too heavily on the Gaussian distribution can lead to:

Underestimation of Risk: Ignoring fat tails can result in severe underestimation of the likelihood and impact of extreme market movements. Mispricing of Derivatives: Incorrect assumptions about the distribution of returns can lead to mispricing and ineffective hedging strategies. Poor Risk Management: Overestimating the effectiveness of diversification and risk models that rely on normality can result in inadequate risk mitigation.

Directions

To address issues arising from non-normal data distributions, consider the following methodologies:

Non-Gaussian Models:

Use distributions like t-distribution, GARCH models, or EVT (Extreme Value Theory) to model fat tails and volatility clustering. Copulas: Employ copula functions to model and simulate the dependency structure between multiple financial variables without assuming normality.

Bootstrap Methods:

Use bootstrap resampling techniques to estimate the distribution of returns and risk measures.

By acknowledging and addressing the limitations of the Gaussian assumption, more robust and accurate financial models can be developed.

Non-stationary

Definition

A time series is said to be non-stationary if its statistical properties, such as mean, variance, and autocorrelation, change over time. Mathematically, a time series $\{Xt\}$ is non-stationary if: $E[Xt] \neq$ constant Var$(Xt) \neq$ constant Cov$(Xt,Xt+k)$ depends on t and k

Description