statistical_analysis

June 4, 2025

1 Análise Estatística de Retornos Diários

Este notebook calcula e visualiza estatísticas descritivas dos retornos diários de ações negociadas na B3, utilizando a API do Yahoo Finance.

[1]:
```
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import skew, kurtosis, norm

sns.set(style="whitegrid")
plt.rcParams["figure.figsize"] = (12,6)
```

1.1 Download dos Dados e Cálculo dos Retornos

[48]:
```
def fetch_and_prepare_data(ticker: str, start="2010-01-01", end=None):
    df = yf.download(ticker, start=start, end=end)
    df = df[["Close"]].rename(columns={"Close": "adjusted_close"})
    df["daily_return"] = df["adjusted_close"].pct_change()
    return df.dropna()

ticker = "BBDC3.SA"
df = fecth_and_prepare_data(ticker)
```

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

Price	adjusted_close	daily_return
Ticker	BBDC3.SA	
Date		
2010-01-04	4.516166	NaN
2010-01-05	4.453969	-0.013772
2010-01-06	4.427732	-0.005891
2010-01-07	4.417532	-0.002303
2010-01-08	4.423359	0.001319
…	…	…
2025-05-29	13.882897	-0.009972
2025-05-30	13.962799	0.005755

```
2025-06-02      14.002750     0.002861
2025-06-03      14.180000     0.012658
2025-06-04      14.180000     0.000000

[3829 rows x 2 columns]
```

1.2 Estatísticas Descritivas dos Retornos Diários

[49]:
```
stats = df["daily_return"].describe()
print(f"  Estatísticas Descritivas de Retornos Diários para {ticker}:")
display(stats)

mean = df["daily_return"].mean()
std = df["daily_return"].std()
skewness = skew(df["daily_return"])
kurt = kurtosis(df["daily_return"])

print(f"\nMédia: {mean:.4%}")
print(f"Desvio Padrão: {std:.4%}")
print(f"Assimetria (Skewness): {skewness:.4f}")
print(f"Curtose: {kurt:.4f}")
```

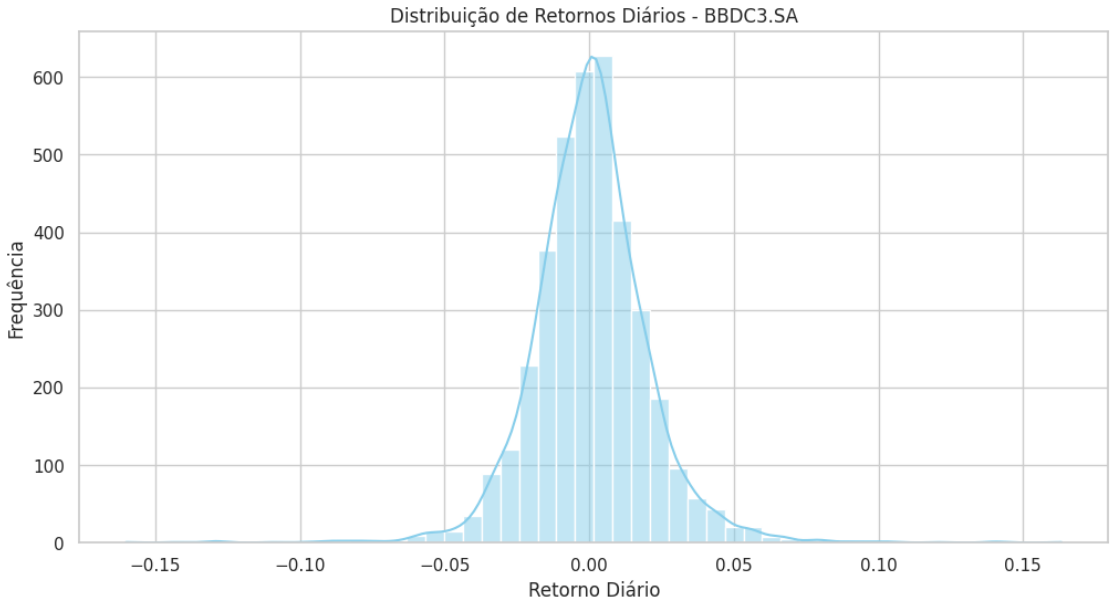
```
  Estatísticas Descritivas de Retornos Diários para BBDC3.SA:

count    3828.000000
mean        0.000508
std         0.020456
min        -0.160126
25%        -0.010609
50%         0.000000
75%         0.011068
max         0.163276
Name: daily_return, dtype: float64


Média: 0.0508%
Desvio Padrão: 2.0456%
Assimetria (Skewness): 0.0670
Curtose: 6.9206
```

1.3 Distribuição dos Retornos Diários

[50]:
```
sns.histplot(df["daily_return"], bins=50, kde=True, color="skyblue")
plt.title(f"Distribuição de Retornos Diários - {ticker}")
plt.xlabel("Retorno Diário")
plt.ylabel("Frequência")
plt.show()
```



[51]:
```
annual_volatility = df["daily_return"].std() * np.sqrt(252)
confidence_level = 0.95
VaR_95 = np.percentile(df["daily_return"], (1 -  confidence_level) * 100)
VaR_parametric = norm.ppf(1 - confidence_level, df["daily_return"].mean(),␣
 ↪df["daily_return"].std())

print(f"  Volatilidade Anualizada: {annual_volatility:.2%}")
print(f"  VaR Histórico (95%): {VaR_95:.2%}")
print(f"  VaR Paramétrico Normal (95%): {VaR_parametric:.2%}")
```

```
 Volatilidade Anualizada: 32.47%
 VaR Histórico (95%): -2.99%
 VaR Paramétrico Normal (95%): -3.31%
```

1.4 Comparação entre múltiplos ativos

[64]:
```
tickers = ["BBDC3.SA", "ITUB4.SA", "BBAS3.SA"]
returns = {}
all_data = {}

for t in tickers:
    df_temp = fetch_and_prepare_data(t)
    all_data[t] = df_temp
    returns[t] = df_temp["daily_return"]

returns_df = pd.DataFrame(returns).dropna()
display(returns_df.head())
```

```
display(returns_df.describe().T[["mean", "std"]].rename(columns={"mean":␣
 ↪"Média", "std": "Desvio Padrão"}))
```

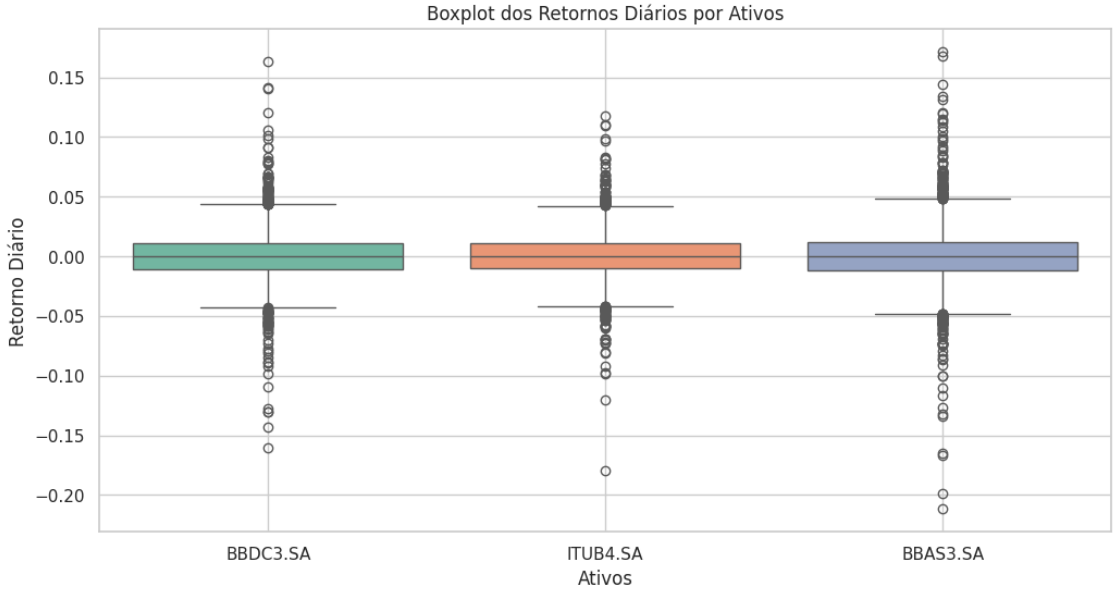
```
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
[**********************100%**********************]  1 of 1 completed
```

Date	BBDC3.SA	ITUB4.SA	BBAS3.SA
2010-01-05	-0.013773	0.006481	-0.010034
2010-01-06	-0.005891	-0.008668	0.001352
2010-01-07	-0.002304	-0.010242	0.000337
2010-01-08	0.001319	-0.015144	0.005734
2010-01-11	-0.003294	-0.008713	0.007712

	Média	Desvio Padrão
BBDC3.SA	0.000508	0.020456
ITUB4.SA	0.000480	0.019129
BBAS3.SA	0.000670	0.024182

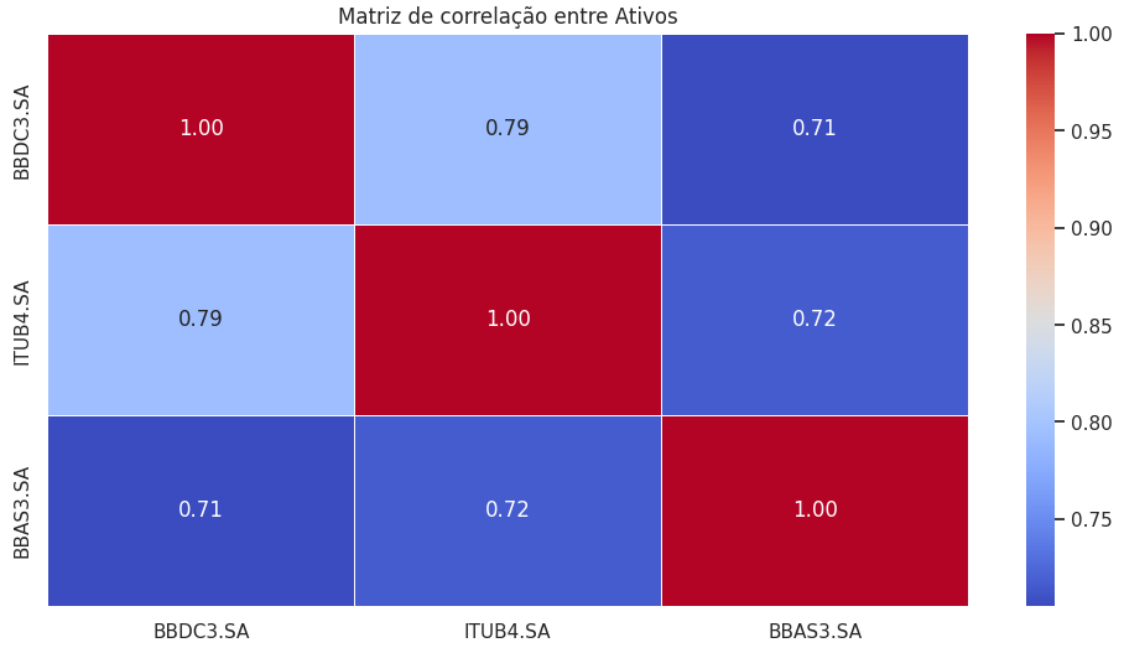
1.5 Boxplot Comparativo dos Retornos

[53]:
```
return_melted = returns_df.reset_index().melt(id_vars="Date", var_name="Ativo",␣
 ↪value_name="Retorno")
sns.boxplot(data=return_melted, x="Ativo", y="Retorno",hue="Ativo",␣
 ↪palette="Set2", legend=False)
plt.title("Boxplot dos Retornos Diários por Ativos")
plt.xlabel("Ativos")
plt.ylabel("Retorno Diário")
plt.grid(True)
plt.show()
```



1.6 Correlação entre os Ativos

[54]:
```
correlation_matrix = returns_df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f",␣
↪linewidths=0.5)
plt.title("Matriz de correlação entre Ativos")
plt.show()
```

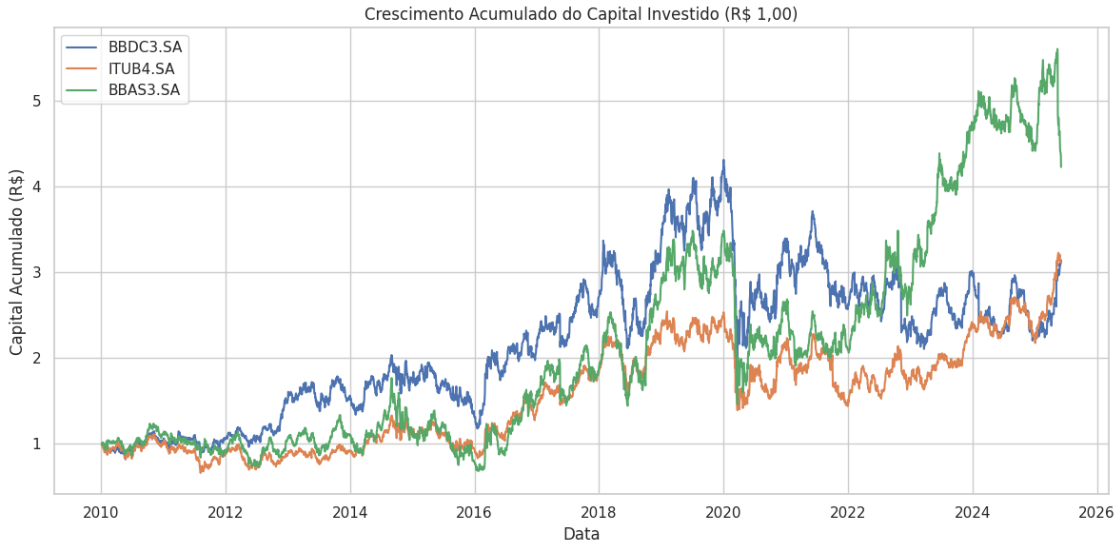


1.7 Crescimento Acumulado do Capital Investido

[65]:
```
plt.figure(figsize=(12,6))

for ticker, df in all_data.items():
    cumulative_growth = (1 + df["daily_return"]).cumprod()
    plt.plot(cumulative_growth, label=ticker)

plt.title("Crescimento Acumulado do Capital Investido (R$ 1,00)")
plt.xlabel("Data")
plt.ylabel("Capital Acumulado (R$)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



1.8 Interpretação:

O gráfico acima simula como R$ 1,00 investido no início do período teria evoluído ao longo do tempo em cada um dos ativos analisados, com base nos retornos diários calculados.

Ele representa o crescimento acumulado do capital, **assumindo que todos os lucros (ou prejuízos) foram reinvestidos diariamente** — ou seja, mostra o efeito dos juros compostos ao longo do tempo.

Através desse gráfico, é possível:

- Comparar o desempenho relativo de cada ação durante o período;
- Identificar qual ativo teve o maior retorno acumulado;
- Visualizar momentos de valorização ou queda do investimento;
- Observar a consistência ou volatilidade no crescimento do capital.

[ ]: