**Aluno: Davi Braga do Amaral  
Matrícula: 17/0079201**

Detalhamento do Código  
  
1. Importação e Setup

import os, numpy as np, pandas as pd, matplotlib.pyplot as plt

from scipy import stats

np.random.seed(42)

plt.rcParams['figure.figsize'] = (8, 5)

* Carrega bibliotecas (arquivo, cálculo numérico, dataframes, gráficos e estatística).
* Fixa a **seed** (reprodutibilidade) e define tamanho padrão dos gráficos.

2. Configuração  
  
EXCEL\_PATH = "dados\_atividade1.xlsx"

# (opção de ler do Google Drive está comentada)

PRE\_DAYS = 5

POST\_DAYS = 5

EST\_WINDOW = 120 # janela de estimação do market model

GAP = 21 # folga antes do evento para evitar contaminação

* Onde encontra-se a base de dados.
* **Janela do evento**: do dia -5 ao +5 em torno do Copom.
* **Janela de estimação** do *market model* (α, β) usa 120 pregões **bem antes** do evento; o **GAP** de 21 dias evita vazar informação do período “colado” no evento.

3. Leitura da Base de Dados

def read\_excel\_or\_upload(path=EXCEL\_PATH):

if os.path.exists(path): ...

else: abre files.upload() no Colab ...

df = read\_excel\_or\_upload(EXCEL\_PATH)

df = df.rename(columns={c: c.strip() for c in df.columns})

assert {'Data','IBOV','IFNC'}.issubset(df.columns)

df['Data'] = pd.to\_datetime(df['Data'])

df = df.sort\_values('Data').reset\_index(drop=True)

display(df.head(), df.tail()); df.info()

* Tenta ler o arquivo do caminho; se não existir, abre o seletor para ser enviado.
* Normaliza nomes das colunas, garante que existem Data, IBOV, IFNC.
* Converte Data para datetime e ordena cronologicamente.
* Mostra uma prévia (cabeça/cauda) e os tipos.

Observação no código: ele assume que IBOV e IFNC já são retornos diários. Se fossem preços, seria necessário fazer pct\_change() antes.

4. Definição dos Eventos:

events = pd.DataFrame({

'governo': [...],

'data\_evento': [...],

'selic': [...],

'obs': ['Maior Selic' ou 'Menor Selic']

})

display(events)

* Tabela com 8 eventos (as datas de maior e menor Selic em cada governo), nível da Selic e um rótulo (“Maior/ Menor”).
* É a lista de choques que vamos analisar.

5. Série Indexada por Data

ret = df[['Data','IBOV','IFNC']].copy().set\_index('Data')

* Cria um dataframe indexado por data para facilitar recortes por janelas.

6. Funções auxiliares:

def nearest\_trading\_day(index, target\_date):

# se evento cai em fim de semana/feriado, pega o pregão mais próximo

* Ajuste para o dia do estudo.

def get\_event\_window(ret\_df, event\_date, pre, post):

ed = nearest\_trading\_day(...)

# pega as datas do índice de t=-pre até t=+post

# retorna: data ajustada do evento (ed), o índice de datas (idx) e o vetor relativo t=[-pre..+post]

* Construção da janela do evento.
* Se a janela estoura o começo/fim da série, retorna None (evento é pulado).

def estimate\_market\_model(ret\_df, event\_date, est\_window, gap):

# janela de estimação: [event - gap - est\_window, event - gap]

# OLS: R\_IFNC = α + β \* R\_IBOV

X = [1, IBOV]; y = IFNC

alpha, beta = np.linalg.lstsq(X, y, rcond=None)[0]

return alpha, beta

* Regressão do market model
* Estima **α** (intercepto) e **β** (sensibilidade ao IBOV) **bem antes** do evento (por causa do GAP).
* Se faltar dado, retorna None.

7. Cálculo de AR e CAR por evento.

mu\_ifnc = ret['IFNC'].mean() # média histórica (p/ mean-adjusted)

rows = []

for cada evento em events:

ed, idx, rel = get\_event\_window(...)

win = ret.loc[idx] # sub-série na janela

# Mean-adjusted (azul)

exp\_ma = média(IFNC) em todos os dias da janela

AR\_ma = IFNC\_observado - exp\_ma

CAR\_ma = cumsum(AR\_ma) # acumulado

# Market model (laranja)

(a, b) = estimate\_market\_model(...) (se falhar, cai no exp\_ma)

exp\_mm = a + b \* IBOV

AR\_mm = IFNC - exp\_mm

CAR\_mm = cumsum(AR\_mm)

# empilha os resultados deste evento

rows.append(DataFrame(... 't', 'AR\_ma','CAR\_ma','AR\_mm','CAR\_mm'))

ev\_df = concat(rows)

display(ev\_df.head())

* AR (*Abnormal Return*) é o desvio do retorno observado em relação ao esperado.
  + Mean-adjusted: esperado = média histórica do IFNC.
  + Market model: esperado = α + β·IBOV (estimado na janela de estimação).
* CAR é o acumulado dos ARs na janela (soma dia a dia).
* ev\_df guarda tudo: para cada evento e cada t (−5..+5), você tem AR/CAR nas duas abordagens.

8. Agregação (AR/CAR) e teste-t.

def aggregate\_and\_ttest(df, ar\_col, car\_col):

agg = df.groupby('t').agg(

AAR = média dos ARs no dia t (sobre todos os eventos),

CAAR = média dos CARs no dia t,

N = nº de eventos válidos naquele t,

SD\_AR = desvio-padrão dos ARs naquele t

)

# Estatística t para H0: AAR\_t = 0

t\_stat = AAR / (SD\_AR / sqrt(N))

p\_value = 2 \* (1 - CDF\_t(|t\_stat|; df=N-1))

* Roda duas vezes:
  + agg\_ma para AR/CAR mean-adjusted,
  + agg\_mm para AR/CAR market model.
* Saída: tabelas com t, AAR, CAAR, N, SD\_AR, t\_stat, p\_value.

Interpretação rápida:

* AAR(t) positivo e p < 0,05 →  retorno anormal médio significativamente positivo no dia t.
*  CAAR mostra efeito acumulado na janela.

9. Gráficos:

# AAR: duas linhas (azul: mean-adjusted; laranja: market model), eixo x = t

# CAAR: idem, mas acumulado

plt.axvline(0, '--') marca o dia do evento

* AAR ressalta o impacto pontual (dia a dia).
* CAAR mostra o impacto acumulado ao redor do evento.
* A linha vertical em t=0 é o anúncio do Copom.

10. Sumário

def pick\_rows(agg, ts=[0,1,2]): ...

display(pick\_rows(agg\_ma)); display(pick\_rows(agg\_mm))

* Extrai uma mini-tabela com AAR, p-valor e CAAR nos dias mais importantes para sua discussão:

t=0 (dia do anúncio),

t=+1 (reação no dia seguinte),

t=+2 (persistência).

11. Comparação por tipo de evento:

def aggregate\_by\_obs(df, obs\_value, ...):

# repete a agregação só para o subgrupo

agg\_ma\_maior = aggregate\_by\_obs(ev\_df, 'Maior Selic', ...)

agg\_ma\_menor = aggregate\_by\_obs(ev\_df, 'Menor Selic', ...)

# plota CAAR de cada grupo

* Separa os eventos rotulados como “Maior Selic” vs “Menor Selic” e compara os CAARs.