perplexity

Geração de datas sintéticas com NumPy & Pandas para protótipos em Plotly + Dash

Take-away
Pandas e NumPy oferecem um "arsenal" completo para simular calendários, séries temporais e eventos aleatórios – de sequências regulares por minuto a feriados bancários – permitindo criar cenários realistas para gráficos e dashboards sem depender de bases reais.

1. Fundamentos em uma tabela

Abordagem	Funções–chave	Exemplos de uso	Vantagens	Observações
Sequência regular	`pd.date_range()`	Horários a cada 15 min, dias, meses	Simples e rápido	Freqs (`'15T'`, `'H'`, `'MS'`, …) [1]
Dias úteis	`pd.bdate_range()`	Calendário de pregão/expediente	Exclui fins-de-semana sem esforço [2] [3]	Aceita feriados personalizados
Calendário de negócios avançado	`np.busdaycalendar`, `np.is_busday()`	Restrição por feriados nacionais	Controle fino dia-a-dia [2]	Trabalha em arrays NumPy
Séries com deslocamentos	`pd.offsets.*` (`MonthEnd`, `QuarterBegin`…)	Último dia útil do mês, 3.º dia útil etc.	Datas "inteligentes"	Combinável com `+`/`-` (ex.: `+ pd.DateOffset(days=2)`) [4]
Datas aleatórias uniformes	`np.random.randint`, `astype('datetime64[ms]')`	Simular carimbos de IoT	Ultra-rápido	Lembre-se de fixar semente (`np.random.seed`) [5]
Datas aleatórias em dias úteis	`np.random.choice(pd.bdate_range(...))`	Movimentação bancária	Garante business days [3]	
Amostras ponderadas	`np.random.choice(..., p=pesos)`	25% logo após o amanhecer etc. [6]	Controla sazonalidade	
Time-zones	`.tz_localize()` / `.tz_convert()`	Dados globais	Evita erros de fuso [7]	
Multinível (painéis)	`pd.MultiIndex.from_product([datas, categorias])`	Métricas por filial X data	Fácil de pivotar	

2. Construindo séries-base

```
import pandas as pd
import numpy as np

# 2.1 Datas regulares por minuto em 24 h
idx_min = pd.date_range('2025-01-01', periods=1440, freq='T')

# 2.2 Datas úteis de um semestre
idx_biz = pd.bdate_range('2025-01-01', '2025-06-30')

# 2.3 Calendário customizado (exclui feriados nacionais)
br_holidays = ['2025-04-21', '2025-05-01']
cal = np.busdaycalendar(holidays=br_holidays)
```

```
mask = np.is_busday(idx_biz.values.astype('datetime64[D]'), busdaycal=cal)
idx_biz_no_hol = idx_biz[mask]
```

`pd.date_range` aceita qualquer **string de frequência**: minutos (`T`), horas (`H`), dias úteis (`B`), finais de mês (`BM`, `M`), início de trimestre (`QS-JAN`) etc.[1]

3. Injetando aleatoriedade

3.1 Amostra aleatória uniforme de timestamps

```
np.random.seed(42)                    # reprodutibilidade[^1_60]
n = 10_000
epoch_start = np.datetime64('2025-01-01T00:00')
rand_ns = np.random.randint(0, 86_400*1e9, n)   # até 24 h em nanos
ts = epoch_start + rand_ns.astype('timedelta64[ns]')
```

3.2 Aleatório restrito a dias úteis

```
rng = pd.bdate_range('2025-01-01', '2025-01-31')
biz_rand = np.random.choice(rng, size=500)
```

3.3 Probabilidade sazonal (25% próximo do nascer/por-do-sol)[6]

```
def random_solar(sunrise='07:00', sunset='18:00', size=1000):
    sunrise = pd.to_timedelta(sunrise)
    sunset  = pd.to_timedelta(sunset)
    span    = (sunset - sunrise).seconds
    r       = np.random.rand(size)
    secs = np.where(
        r < .25,                      # 25 %: 1ª hora
        np.random.randint(0, 3600, size),
        np.where(
            r > .75,                  # 25 %: última hora
            span - np.random.randint(0, 3600, size),
            np.random.randint(3600, span-3600, size)  # 50 % meio
        )
    )
    return sunrise + pd.to_timedelta(secs, unit='s')

times = random_solar(size=10000)
```

4. Combinando múltiplas dimensões

```
dates = pd.date_range('2025-01', '2025-12', freq='D')
categorias = ['Centro-Sul', 'Nordeste', 'Sul']
produtos = ['Premium', 'Standard']

mux = pd.MultiIndex.from_product([dates, categorias, produtos],
                                 names=['data', 'região', 'plano'])

df = pd.DataFrame({
    'vendas': np.random.poisson(lam=20, size=len(mux)),
    'ticket_médio': np.round(np.random.normal(120, 30, size=len(mux)), 2)
}, index=mux)
```

Com `MultiIndex` fica fácil pivotar para mapas de calor ou gráficos de barras empilhadas em Plotly.

5. Cenários de demonstração para Dash

Cenário	Como gerar	Insight a mostrar
Realtime dashboard	`pd.date_range(…, freq='S')` + ruído de sensor	Gráfico streaming de linhas
Comparativo sazonal	Séries ano-sobre-ano com `.groupby(data.dt.month)`	Área acumulada por mês
"Gap analysis"	Falhas deliberadas usando `df.sample(frac=.1).drop(...)`	Heatmap de missing + interpolação [8]
Forecast fake	Tendência linear + salto aleatório (`np.random.normal`)	Forecast vs. Real em linha pontilhada
Multi-timezone	`.tz_localize('UTC').tz_convert(['America/Sao_Paulo', …])`	Small-multiples fuso horário

6. Dicas finais

1. **Use offsets** para criar datas "inteligentes" (ex.: faturamento dois dias úteis após o fechamento):

   ```
   fecho = pd.Timestamp('2025-03-31') + pd.offsets.BDay(2)
   ```

2. **Interpolação / reamostragem**: gere em frequência fina e depois `resample('H').mean()` para dashboards detalhados.
3. **Copy-on-write** do Pandas 2.0 evita cópias extras – bom para grandes simulações [9].
4. Fixe *seed* ao demonstrar notebooks: `np.random.seed(0)` garante que o gráfico no slide sempre bata [5].

Com esses recursos você consegue montar desde um *mock* de telemetria até um painel executivo multi-região, tudo sem dados sensíveis – apenas criatividade na combinação de `pd.date_range`, `numpy.random` e alguns offsets!

⁂

1. https://www.reddit.com/r/pystats/comments/2znixc/why_does_pandasdate_range_generate_timestamps/
2. https://www.reddit.com/r/learnpython/comments/klv9tw/trade_day_of_year/
3. https://www.reddit.com/r/learnpython/comments/dxlkxs/my_pandas_implementation_for_creating_test_data/
4. https://www.reddit.com/r/learnpython/comments/z9utzo/how_do_i_create_a_custom_daterange_frequency_with/
5. https://www.reddit.com/r/learnpython/comments/s678b0/explain_randomseed_like_im_five/
6. https://www.reddit.com/r/learnpython/comments/11xds2o/weight_random_choice/
7. https://www.reddit.com/r/learnpython/comments/11juc5y/creating_a_timezone_aware_datetime64_instance/
8. https://www.reddit.com/r/learnpython/comments/10dgot9/merging_data_frames_to_fill_missing_values/
9. https://www.reddit.com/r/Python/comments/11fio85/we_are_the_developers_behind_pandas_currently/