perplexity

# Gerando datas sintéticas com NumPy e pandas para dashboards avançados em Plotly & Dash

**Principais ideias**

1. pandas e NumPy oferecem geradores de datas de alta-performance (`date_range`, `bdate_range`, `period_range`, `arange`, `datetime64`).
2. Combinar frequências, sazonalidades, feriados, fusos e ruído cria séries realistas para IoT, vendas, finanças ou saúde.
3. Plotly interpreta objetos DatetimeIndex, arrays `datetime64` e strings ISO automaticamente, simplificando eixos de tempo interativos[1].
4. Dash conecta esses dados a filtros (DatePicker, RangeSlider) e callbacks reativos, formando painéis prontos para produção[2] [3].

## 1 | Catálogo rápido de geradores de datas

| Biblioteca | Função | O que faz | Exemplo rápido |
|---|---|---|---|
| pandas | `date_range()` | Grade fixa; suporta todos os *offset aliases* (“D”, “H”, “M”, “QS”…)[4] | `pd.date_range('2024-01-01','2024-12-31',freq='W-MON')` |
| pandas | `bdate_range()` | Só dias úteis[5] | `pd.bdate_range('2025-01-01',periods=22)` |
| pandas | `period_range()` | Índice periódico (sem horas)[5] | `pd.period_range('2020Q1',periods=12,freq='Q')` |
| pandas | `timedelta_range()` | Range de durações | `pd.timedelta_range('0s',periods=100,freq='15s')` |
| NumPy | `np.arange(dtype='datetime64[...]')` | Range rápido de datas[6] | `np.arange('2025-01','2025-06',dtype='datetime64[D]')` |
| NumPy | escolha aleatória | Amostrar timestamps[6] | `np.random.choice(dt_array,1000)` |
| Faker | `fake.date_between()` | Datas randômicas com restrições[7] | `fake.date_between('-2y','today')` |
| ## 2 | Estratégias para séries sintéticas | | |

### 2.1 Frequência fixa clássica

```
dias = pd.date_range('2022-01-01', '2024-12-31', freq='D')
horas = pd.date_range('2025-03-01', periods=168, freq='H')  # 1 semana
```

### 2.2 Calendário de negócios ou fusos

```
uteis_sp = pd.bdate_range('2025-01-01', '2025-12-31', freq='C', tz='America/Sao_Paulo')
```

### 2.3 Sazonalidade + tendência + ruído

Guia resumido do artigo Index.dev[8] e do manual NumPy[6]:

```
def serie_sazonal(n, start='2023-01-01', freq='D'):
    idx   = pd.date_range(start, periods=n, freq=freq)
    trend = np.linspace(0, 5, n)                              # tendência
    saz   = 10*np.sin(np.linspace(0, 2*np.pi, n))             # padrão anual
    ruido = np.random.normal(0, 1.5, n)                       # ruído
    return pd.Series(trend + saz + ruido, index=idx, name='valor')
ts = serie_sazonal(730)
```

### 2.4 Processos estocásticos (Random Walk)

```
walk = ts.cumsum()                                            # random walk diário[^1_4]
```

### 2.5 Datas irregulares e picos

```
# Selecionar 5 000 timestamps aleatórios em 2 anos
base = np.arange('2024-01-01','2026-01-01', dtype='datetime64[m]')
irreg = np.random.choice(base, 5000, replace=False)
```

### 2.6 Bibliotecas específicas de *time-series GANs*

- **DoppelGANger** – gera múltiplas variáveis temporais preservando correlações[9].
- **DeepEcho / SDV** – séries multivariadas com dependências complexas[10].

### 3 | Exemplo completo: vendas sintéticas multiloja

```
import numpy as np, pandas as pd
np.random.seed(42)

stores  = ['Centro','Norte','Sul','Oeste']
base    = pd.date_range('2023-01-01','2024-12-31',freq='D')
n_trans = 10_000

def sample_timestamp():
    d   = np.random.choice(base)
    h   = np.random.choice(range(8,23), p=[.05,.08,.12,.15,.15,.15,.10,.05,.03,.02,.02,.03,.02,.02,.03])
    m   = np.random.randint(0,60)
    return pd.Timestamp(d)+pd.Timedelta(hours=h, minutes=m)

df = pd.DataFrame({
    'timestamp'  : [sample_timestamp() for _ in range(n_trans)],
    'store'      : np.random.choice(stores, n_trans),
    'sale_value' : np.random.lognormal(mean=3, sigma=.8, size=n_trans),
    'items_sold' : np.random.poisson(lam=2, size=n_trans)+1
})
```

A estrutura permite:

- **séries por loja**, **heatmaps hora×dia**, **acumulados mensais** etc.

## 4 | Visualizando no Plotly

### 4.1 Linha diária acumulada

```
import plotly.express as px
daily = (df.set_index('timestamp')
           .resample('D')['sale_value']
           .sum()
           .reset_index())
fig = px.line(daily, x='timestamp', y='sale_value',
              title='Receita diária')
fig.update_xaxes(rangeslider_visible=True)   # range-slider nativo[^1_21]
fig.show()
```

### 4.2 Heatmap hora × dia da semana

```
df['dow']  = df['timestamp'].dt.day_name()
df['hour'] = df['timestamp'].dt.hour
pivot     = (df.pivot_table(index='dow', columns='hour',
                            values='sale_value', aggfunc='sum')
                .reindex(['Monday','Tuesday','Wednesday',
                          'Thursday','Friday','Saturday','Sunday']))
fig = px.imshow(pivot, aspect='auto', color_continuous_scale='Turbo')
```

Plotly detecta automaticamente objetos datetime[1]; evite converter para string, salvo se a performance for crítica (render 10× mais rápido em alguns casos[11]).

## 5 | Montando um painel Dash reativo

```
import dash, dash_core_components as dcc, dash_html_components as html
from dash.dependencies import Input, Output
import plotly.express as px

app = dash.Dash(__name__)
app.layout = html.Div([
    dcc.DatePickerRange(id='picker',
        start_date=df.timestamp.min(),
        end_date=df.timestamp.max()),
    dcc.Dropdown(id='store', multi=True,
        options=[{'label':s,'value':s} for s in df.store.unique()],
        value=df.store.unique().tolist()),
    dcc.Graph(id='graph')
])

@app.callback(Output('graph','figure'),
              [Input('picker','start_date'),
               Input('picker','end_date'),
               Input('store','value')])
def update(s,e,stores):
    mask = (df.timestamp.between(s,e)) & (df.store.isin(stores))
    cur  = (df[mask].set_index('timestamp')
              .resample('D')['sale_value'].sum().reset_index())
    return px.line(cur, x='timestamp', y='sale_value')

# Hot-reload mantém o date default sempre atualizado[^1_27]
if __name__ == '__main__':
    app.run_server(debug=True)
```

## 6 | Boas práticas e combinações úteis

1. **Reprodutibilidade** `np.random.seed(…)` antes de gerar dados[8].
2. **Aliases de frequência** `'15T'` (minutos), `'MS'` (month-start), `'QS'` (quarter-start), `'BH'` (business hour)[12].
3. **Fusos & horário de verão** `tz='America/Sao_Paulo'` e posterior `tz_convert` quando necessário[6].
4. **Datas futuras/retroativas com Faker** `fake.date_time_between('-30d','+30d')` para testes de inputs de usuário[13].
5. **Multiprocessamento** para séries muito longas, gere blocos e concatene (evita grandes vetores únicos).
6. **Privacidade** quando seus dashboards derivam de dados sensíveis (saúde/finanças), avalie SDV/DeepEcho ou GANs como DoppelGANger[9] [10] [14].

### Onde aplicar

- **IoT** leitura horária de sensores com falhas aleatórias.
- **E-commerce** transações com picos em Black Friday e horários nobres.
- **Finanças** cotações em random-walk ou séries GBM para back-testes[15].
- **Saúde** sinais vitais minuto-a-minuto; preservar correlações entre múltiplos sinais via SDV[16] [10].

Gere, explore e conecte-os ao ecossistema Plotly/Dash para painéis interativos de alto impacto – tudo sem depender de bases reais.

⁂


1. https://plotly.com/python/time-series/
2. https://community.plotly.com/t/ploting-time-series-data/5265
3. https://community.plotly.com/t/dash-python-changing-default-date-dynamically-everyday/27823
4. https://pandas.pydata.org/docs/reference/api/pandas.date_range.html
5. https://towardsdatascience.com/dealing-with-dates-in-pythons-dataframe-part-1-date-series-creation-f4a800db9ae/
6. https://numpy.org/doc/stable/reference/arrays.datetime.html
7. https://www.unimedia.tech/mastering-python-faker-date_between-function-for-realistic-test-data/
8. https://www.index.dev/blog/generate-time-series-data-python
9. https://www.kdnuggets.com/2022/06/generate-synthetic-timeseries-data-opensource-tools.html
10. https://github.com/sdv-dev/DeepEcho
11. https://discuss.streamlit.io/t/plotly-chart-performance-with-datetime-x-axis/80935
12. https://pandas.pydata.org/docs/user_guide/timeseries.html
13. https://faker.readthedocs.io/en/master/providers/faker.providers.date_time.html
14. http://arxiv.org/pdf/2401.00081.pdf
15. https://towardsdatascience.com/generating-synthetic-time-series-data-with-random-walks-8701bb9a56a8/
16. https://github.com/Jeremy-Harper/Synthetic-Data-Replica-for-Healthcare