Visualização histórica

Iniciamos importando as bibliotecas necessárias:

In [ ]:

```
# Import lybraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Vega lite for interactive plots
import altair as alt
from vega_datasets import data
import altair.vegalite.v3 as v3
```

Agora, criamos um `dataframe` com os dados (como os dados originais não foram encontrados, foram extraídos os valores aproximados extraídos da visualização original):

In [ ]:

```
data = {
        "percent": [
                100.0, 101.0, 100.0, 102.0, 101.5, 103.0, 101.5,
                102.0, 101.5, 102.0, 100.0, 100.5, 98.5, 99.0,
                97.5, 98.0, 97.5, 98.0, 96.5, 97.0, 94.5],
        "date": pd.date_range(start='1914-07-01',
                  end='1916-12-31',
                  periods=21)
}
data = pd.DataFrame(data)
data
```

Out[ ]:

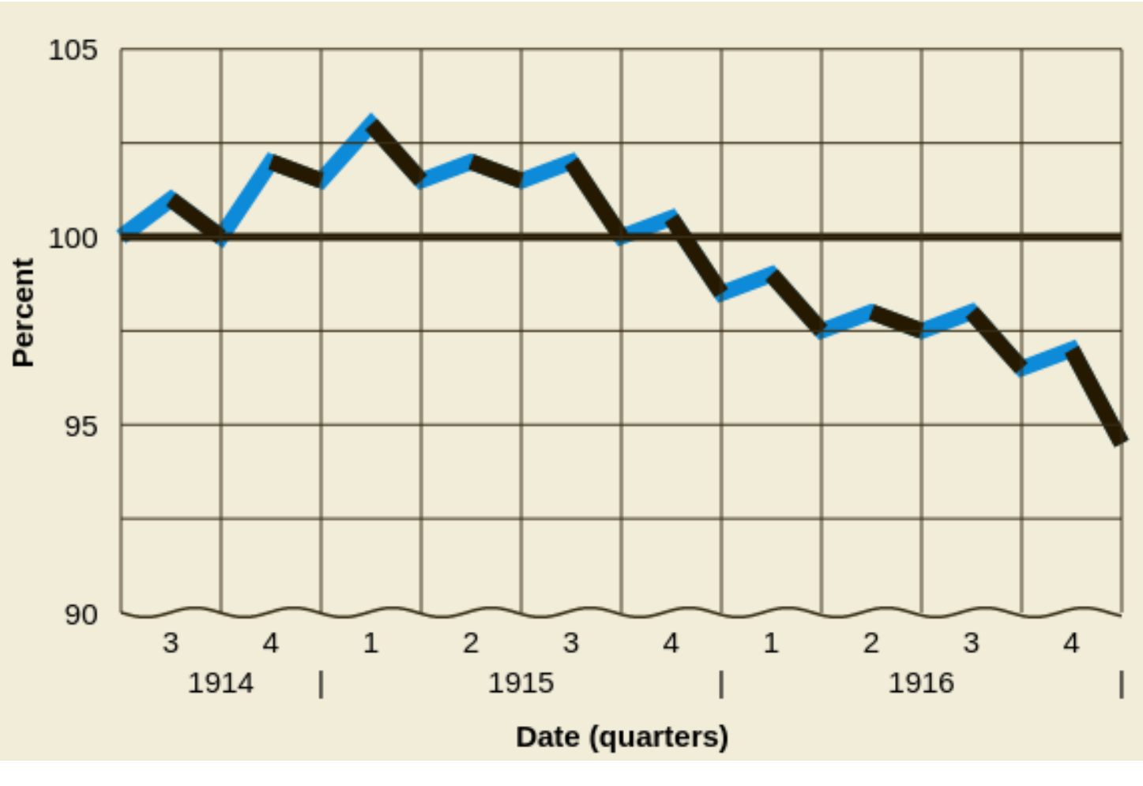
	percent	date
0	100.0	1914-07-01 00:00:00
1	101.0	1914-08-15 16:48:00
2	100.0	1914-09-30 09:36:00
3	102.0	1914-11-15 02:24:00
4	101.5	1914-12-30 19:12:00
5	103.0	1915-02-14 12:00:00
6	101.5	1915-04-01 04:48:00
7	102.0	1915-05-16 21:36:00
8	101.5	1915-07-01 14:24:00
9	102.0	1915-08-16 07:12:00
10	100.0	1915-10-01 00:00:00
11	100.5	1915-11-15 16:48:00
12	98.5	1915-12-31 09:36:00
13	99.0	1916-02-15 02:24:00
14	97.5	1916-03-31 19:12:00
15	98.0	1916-05-16 12:00:00
16	97.5	1916-07-01 04:48:00
17	98.0	1916-08-15 21:36:00
18	96.5	1916-09-30 14:24:00
19	97.0	1916-11-15 07:12:00
20	94.5	1916-12-31 00:00:00

Com isso, podemos fazer o gráfico. Devemos ressaltar que para gerar um resultado parecido com o original, foi necessário fazer várias adaptações: o *grid* do gráfico preciso ser removido (dos valores padrões) pois o Altair não deixa defini-lo precisamente, ele cria modificações como julga melhor para o resultado, deixando diferente do desejado; a linha ondulada embaixo do gráfico também foi feita "manualmente" (com uma função senoidal, gerando os dados dos pontos da senoide e plotando um gráfico de linha); as *labels* do eixo `x` também foram feitas manualmente, pois criar labels multilinha é um ainda problema conhecido e não resolvido do Altair e, além disso, acontecia o mesmo problema do *grid*. Segue abaixo o código para atingir o resultado do gráfico:

In [ ]:

```
# Set palette for plots (extracted with Adobe Color)
blue = "#0D8BD9"
dark_black = "#261901"
light_black = "#382E16"
light_brown = "#F2EDD9"


# Plot graph of data (all blue)
g = alt.Chart(data).mark_line(
    size = 8,
    color = blue
).encode(
    x = alt.X(
        "date:T",
        title="Date (quarters)"
    ),
    y = alt.Y(
        "percent",
        scale = alt.Scale(domain = (90, 105)),
        title="Percent"
    )
)

# Plot graph of data (black lines for descending)
gs = []
for i in range(1, 20, 2):
    # Plot each black line
    gs.append(
        alt.Chart(data[:][i:i+2]).mark_line(
            size = 8,
            color = dark_black
        ).encode(
            x = alt.X("date"),
            y = alt.Y(
                "percent",
                scale=alt.Scale(domain = (90, 105))
            )
        )
    )
# Combine all graphs
for i in range(len(gs)):
    g = g + gs[i]

# Plot line for 100%
yrule = (
    alt.Chart().mark_rule(
        size = 4,
        color = dark_black
    ).encode(y = alt.datum(100))
)

# Transform data to get the DateTime points
alt_date = []
for i in range(0, 21):
    data_i = pd.to_datetime(data["date"][i])
    alt_date.append(data_i)
data["alt_date"] = alt_date
data["y_min"] = 90
data["y_max"] = 105

# Plot vertical lines (grid)
x_lines_ = alt.Chart(data[:][0:21:2]).encode(
    alt.X('alt_date:T')
)
x_lines_ = x_lines_.mark_rule(color = light_black).encode(
    alt.Y('y_min:Q'),
    alt.Y2('y_max:Q')
)

# Plot horizontal lines (grid)
y_lines = []
for i in range(5, 33, 5):
    y_lines.append(
        alt.Chart().mark_rule(
            size = 1,
            color = light_black
        ).encode(y = alt.datum(90 + (i / 2)))
    )
y_lines_ = y_lines[0]
for i in range(1, len(y_lines)):
    y_lines_ = y_lines_ + y_lines[i]

# Generate data for waved line for bottom of graph (with 1000 points)
x = np.arange(1001) / 50
x_temp = pd.date_range(start = '1914-07-01',
                       end = '1916-12-31',
                       periods = 1001)
source = pd.DataFrame({
    'x': x_temp,
    'f(x)': - np.sin((10 / np.pi) * x) / 8 + 90
})

# Plot waved line for bottom of graph
sin_line = alt.Chart(source).mark_line(
    color = light_black,
    size = 1.5
).encode(
    x='x',
    y='f(x)',
)

# Plot text for x axis of graph (multiline labels)
texts = []
for i in range(1, 11):
    texts.append( # Quarters
        alt.Chart().mark_text(
            x = 50*i - 25,
            y = 315,
            text = f"{(i + 1) % 4 + 1}",
            color = "black",
            size=15
        ).encode()
    )
texts_ = texts[0]
for i in range(1, len(texts)):
    texts_ = texts_ + texts[i]
texts_ += alt.Chart().mark_text( # First year
    x = 50,
    y = 335,
    text = f"{1914}",
    color = "black",
    size = 15
).encode()
texts_ += alt.Chart().mark_text( # Fist separation of years
    x = 100,
    y = 335,
    text = f"|",
    color = "black",
    size = 15
).encode()
for i in range(1, 3):
    texts_ += alt.Chart().mark_text( # Two last years
        x = 200 * i,
        y = 335,
        text = f"{1914 + i}",
        color = "black",
        size = 15
    ).encode()
    texts_ += alt.Chart().mark_text( # Two last separation of years
        x = 200 * i + 100,
        y = 335,
        text = f"|",
        color="black",
        size=15
    ).encode()


# Combine all graphs
result = g + yrule + y_lines_ + x_lines_ + sin_line + texts_

# Configure the chart
result = result.configure(background = light_brown)  # "Canvas" background color
result = result.configure_view(
    continuousHeight = 300,     # Height of chart
    continuousWidth = 500,      # Width of chart
    strokeOpacity = 0,          # Remove border
    fill = light_brown          # Background color of chart
)
result = result.configure_axis(
    grid = False,         # Remove grid
    domain = False,       # Remove axis
    titleFontSize = 15    # Font size of axis title
)
result = result.configure_axisX(
    ticks = False,     # Remove ticks
    labels = False,    # Remove labels
    offset = 50,       # Move axis to the bottom of the chart
)
result = result.configure_axisY(
    tickCount = 6,        # Number of ticks
    tickMinStep = 5,      # Minimum step between ticks
    labelFontSize = 15,   # Font size of labels
    ticks = False,        # Remove ticks
    offset = 10           # Move axis to the left of the chart
)

# result.save("chart.html") # Save chart to basic html file (for debugging)

result # Show chart
```

Out[ ]:



Observação: Não foi possível exibir o gráfico no notebook Python ao longo do desenvolvimento do trabalho, mas após a apresentação em aula, com o conselho do Rodrigo Gomes Hutz Pintucci, instalando a extensão do VSCode chamada "Markdown Preview Enhanced" foi possível visualizá-lo.

Com isso, vamos fazer de forma análoga ao primeiro trabalho, e criar um documento HTML (melhor editado que o modelo padrão do Altair) com o resultado.

Assim, fazemos um *template* HTML para inserir os gráficos:

In [ ]:

```
html_template = """
<!DOCTYPE html>
<html>
<head>
  <script src="https://cdn.jsdelivr.net/npm/vega@{vega_version}"></script>
  <script src="https://cdn.jsdelivr.net/npm/vega-lite@{vegalite_version}"></script>
  <script src="https://cdn.jsdelivr.net/npm/vega-embed@{vegaembed_version}"></script>
  <style>
    {style}
  </style>
</head>
<body>

<h1>Exemplo histórico</h1>
<h2>Introdução</h2>
<p>{introduction}</p>

<div id="historic_graph"></div>

<br>

<h2>Considerações finais</h2>
<p>{comments}</p>

<br>

<h2>Observações</h2>
<p>{notes}</p>

<br>

<h2>Bibliografia</h2>
{bibliography}

<br>


<script type="text/javascript">
  vegaEmbed('#historic_graph', {graph}).catch(console.error);
</script>
</body>
</html>
"""
```

Criamos uma folha de estilo:

In [ ]:

```
style = """ body {
                background-color: #121212;
                margin-left: 20%;
                margin-right: 20%;
                text-align: justify;
                font-family: Arial, Helvetica, sans-serif;
            }
            h1 {
                color: #07D9B2;
            }
            h2 {
                color: #F2E7FE;
            }
            p {
                color: #E2E2E2;
            }
            #historic_graph {
                display: flex;
                justify-content: center;
            }
            .marks {
                display: block;
                margin-left: auto;
                margin-right: auto;
                width: 50%;
            }
            """
```

Anotamos todas as considerações (comentários textuais):

In [ ]:

```
introduction = """Para este trabalho, foi escolhido reproduzir um gráfico de Willard
Cope Brinton, o qual é possível ver em seu livro Graphic Presentation, de 1880, na
<a href="https://archive.org/details/graphicpresentat00brinrich/page/302/mode/2up"
target="_blank">página 303</a>. O gráfico em questão é um gráfico de linhas, como é
possível ver no livro, ele representa um método desenvolvido em Washington D. C.
para acompanhar a construção naval britânica, onde a linha azul mostra o que foi
construído no trimestre, e a linha preta o que foi afundado ao fim do mesmo trimestre.
Além disso, também podemos extrair do livro que cem por cento equivalia a quantidade
de navios a vapor britânicos no início da guerra.</p>
<p>Após essa contextualização, devemos explicar qual é a importância histórica desse
gráfico: a linha ondulada no eixo X. Nesse sentido, podemos ver que o gráfico não
começa com a escala do eixo Y em zero, mas sim em 90 e, por esse motivo, o autor
decidiu chamar a atenção do leitor a esse fato reproduzindo a linha ondulada que vemos
no eixo X, simulando um papel rasgado, o que indica que haviam mais dados abaixo daquela
linha e que devemos nos atentar a isso.</p>
<p> Nesse contexto, sabemos que hoje em dia é considerado má prática marcar as linhas de
grade que não sejam as dos eixos X e Y (em 0) de forma diferente das demais, pois não
devemos dar a entender que as linhas de grade têm importâncias diferentes de acordo com a
marca gráfica. Além disso, não é recomendado começar com a escala diferente de 0, pois com
isso podemos estar omitindo dados relevantes, como por exemplo em um gráfico de barras, se
começarmos a escala em outra posição podemos causar a impressão errada na diferença entre
os dados das diferentes barras. Entretanto, vemos que no caso onde desejamos recuperar dados
de valores absolutos e compará-los (como no exemplo do gráfico de barras), realmente isso
pode ser uma má prática, entretanto quando se deseja apenas mostrar variações nos valores,
pode ser realmente útil e em certos casos necessário omitir parte do "domínio" dos valores
como é feito no gráfico de Brinton, e dessa maneira o método que ele utilizou para indicar
que isso foi feito é muito interessante e pode ser bem vindo.</p>
<p>Dito isso, podemos ver abaixo o gráfico reproduzido em Python, utilizando a biblioteca
Altair:"""
```

In [ ]:

```
comments = """Como é possível ver, o resultado final atingido ficou muito parecido com o
gráfico original, sendo bem satisfatório. Entretanto, o processo de criação do gráfico
fica omitido ao só ver o resultado final. Nesse sentido, é importante ressaltar que ao
longo do processo diversos ajustes foram feitos, como: o <i>grid</i> do gráfico preciso ser
removido (dos valores padrões) pois o Altair não deixa defini-lo precisamente, ele cria
modificações como julga melhor para o resultado, deixando diferente do desejado; a linha
ondulada embaixo do gráfico também foi feita "manualmente" (com uma função senoidal,
gerando os dados dos pontos da senoide e plotando um gráfico de linha); as <i>labels</i>
do eixo X também foram feitas "manualmente", pois criar labels multilinha é um ainda
problema conhecido e não resolvido do Altair e, além disso, acontecia o mesmo problema
do <i>grid</i>. Ainda buscando reproduzir o gráfico da forma mais fiel possível, a paleta
de cores foi extraída da imagem original utilizando o <a
    href="https://color.adobe.com/pt/create/image"
        target="_blank">Adobe Color</a> (sabemos que a cor do fundo se deve a cor da página
onde foi feito o gráfico, mas mesmo assim a cor dela também foi considerada na paleta).</p>
<p> Mesmo assim, podemos notar algumas diferenças do gráfico original, como a
alternância entre a linha azul e preta, pois foi necessário fazer toda a linha azul e
após isso sobrepor com as linhas pretas a cada vez que os dados indicavam uma descida.
Também podemos ver que foram modificadas algumas anotações textuais, mas isso foi feito
propositalmente, pois não era o mais importante a ser destacado nessa representação
gráfica, por esse motivo a modificação foi feita para ser mais legível (foram alteradas
as posições dos títulos das <i>labels</i> e removida a anotação de "100%", reforçando a linha
horizontal que já estava presente no gráfico).</p>
<p>Dessa forma, embora tenha sido complicado reproduzir o gráfico, o processo para chegar ao
resultado final foi importante para entender melhor o funcionamento da biblioteca utilizada
para a criação dos gráficos, compreendendo suas limitações e descobrindo meios de contorná-las.
Por fim, novamente, esta é mais uma experiência que nos faz dar valor a documentação da
biblioteca utilizada, que por mais que ela não seja suficiente para resolver todos os problemas
(até porque nem todos os problemas têm solução "da forma esperada" - de maneira que escala bem
com outros dados), ela ainda é uma grande ferramenta de auxílio para o desenvolvimento dos
projetos."""
```

In [ ]:

```
notes = """O código para gerar os gráficos e esta página HTML foram feitos em um notebook python,
que pode ser encontrado em um documento separado. Tanto essa página HTML quanto o notebook python
estão disponíveis no <a
                        href="https://github.com/BrunoFornaro/Visualiza-o-da-Informa-o---Exemplo-Hist-rico"
                        target="_blank">repositório do GitHub</a>."""
```

In [ ]:

```
bibliography = """
<p>ALTAIR. Vega-Altair: Declarative Visualization in Python. Disponível em: <a
                        href="https://altair-viz.github.io/index.html"
                        target="_blank">altair-viz.github.io/</a>. Acessado em 8 de setembro de 2022.</p>
<p>VANDERPLAS, Jake. Multiple Charts in one HTML. Disponível em: <a
                        href="https://github.com/altair-viz/altair/issues/1422"
                        target="_blank">github.com/</a>. Acessado em 1 de outubro de 2022.</p>
<p>MATERIAL. Dark theme. Disponível em: <a
                        href="https://material.io/design/color/dark-theme.html"
                        target="_blank">material.io/</a>. Acessado em 8 de outubro de 2022.</p>
<p>ADOBE. Adobe Color. Disponível em: <a
                        href="https://color.adobe.com/pt/create/image"
                        target="_blank">color.adobe.com/pt/create/</a>. Acessado em 8 de outubro de 2022.</p>
<p>ALTAIR. Top-Level Chart Configuration. Disponível em: <a
                        href="https://altair-viz.github.io/user_guide/configuration.html#config-axis"
                        target="_blank">altair-viz.github.io/</a>. Acessado em 8 de outubro de 2022.</p>
<p>ANDREWS, R. J.. Design secrets we can learn from historic visualizations. Disponível em: <a
                        href="https://www.tableau.com/about/blog/2019/6/design-secrets-historic-visualization"
                        target="_blank">www.tableau.com/</a>. Acessado em 8 de outubro de 2022.</p>
<p>ANDREWS, R. J.. Tear Up Your Baseline. Disponível em: <a
                        href="https://medium.com/nightingale/tear-up-your-baseline-b6b68a2a60f1"
                        target="_blank">medium.com/</a>. Acessado em 8 de outubro de 2022.</p>
<p>BRINTON, Willard Cope. Graphic presentation. Disponível em: <a
                        href="https://archive.org/details/graphicpresentat00brinrich/page/302/mode/2up"
                        target="_blank">archive.org/</a>. Acessado em 8 de outubro de 2022.</p>
"""
```

Por fim, podemos unir tudo escrevendo um documento HMTL com as nossas variáveis criadas:

In [ ]:

```
with open('results.html', 'w') as f:
    f.write(html_template.format(
        vega_version = alt.VEGA_VERSION,
        vegalite_version = alt.VEGALITE_VERSION,
        vegaembed_version = alt.VEGAEMBED_VERSION,
        graph = result.to_json(indent=None),
        style = style,
        introduction = introduction,
        comments = comments,
        notes = notes,
        bibliography = bibliography
    ))
```