Múltiplos Gráficos

Bibliotecas necessárias

Primeiramente, precisamos importar as bibliotecas necessárias.

```
# Import libraries
import pandas as pd
import numpy as np
import altair as alt
```

Criando os gráficos

Desenvolvimento inicial

Começando importando os dados de um arquivo `CSV`, que contém informações "relevantes" sobre os gastos mensais de estudantes universitários (em dólar). Além disso, fazemos um tratamento inicial nos dados para ser mais fácil de trabalhar com eles futuramente.

```
# Read the data
# https://www.kaggle.com/datasets/shariful07/nice-work-thanks-for-share?resource=download
df = pd.read_csv("University Students Monthly Expenses.csv", encoding="UTF-8")

# Column names of non-numeric data
columns_with_nulls = [
       'Living',
       'Part_time_job',
       'Transporting',
       'Smoking',
       'Drinks',
       'Cosmetics_&_Self-care',
       'Monthly_Subscription'
]

# Replace the null values in non-numeric columns with "Null"
for column in columns_with_nulls:
       df[column] = df[column].fillna("Null")

# Change the column names
df = df.rename(columns={
       "Study_year": "Study Year",
       "Part_time_job": "Part time job",
       "Games_&_Hobbies": "Hobbies",
       "Cosmetics_&_Self-care": "Cosmetics & Self-care",
       "Monthly_Subscription": "Monthly Subscription",
       "Monthly_expenses_$": "Monthly Expenses"
})

# Print the data (first 5 rows)
df.head()
```

	Gender	Age	Study Year	Living	Scholarship	Part time job	Transporting	Smoking	Drinks	Hobbies	Cosmetics & Self-care	Monthly Subscription	Monthly Expenses
0	Female	21	2.0	Home	No	No	No	No	No	No	Yes	No	150.0
1	Male	25	3.0	Hostel	No	Yes	Motorcycle	No	No	Yes	Yes	Yes	220.0
2	Male	23	2.0	Home	Yes	No	No	No	No	No	No	Null	180.0
3	Male	19	3.0	Hostel	No	No	Motorcycle	No	No	Yes	Yes	Yes	200.0
4	Female	19	2.0	Home	No	No	Motorcycle	No	No	No	Yes	No	300.0

Com os dados prontos, podemos fazer os gráficos (salvaremos inicialmente em objetos contendo a informação dos gráficos, antes de plotar).

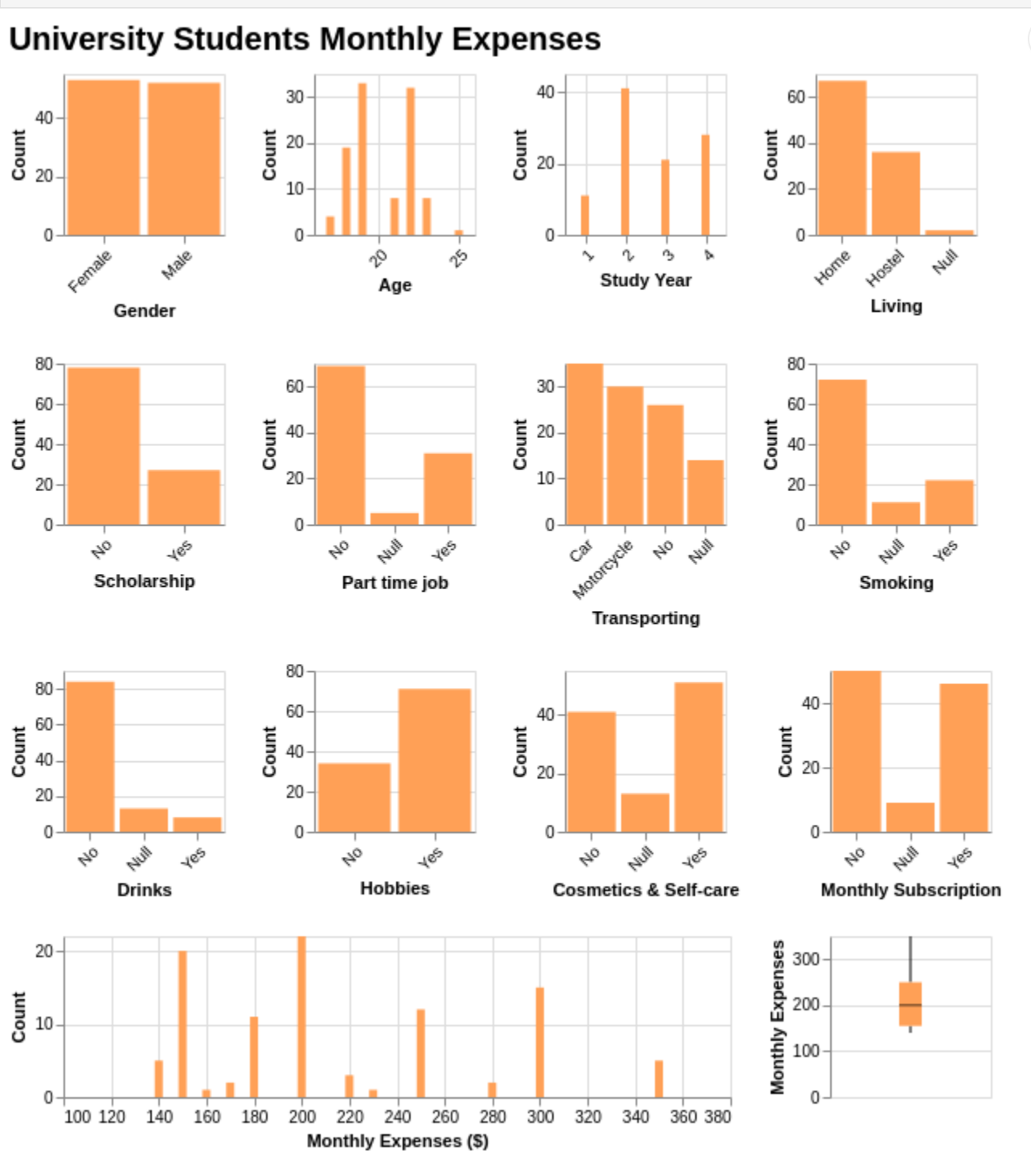
```
# Pallette
orange = "#FFA056"
black = "black"

# Initial variables
multi = [] # List of objects for interactive selection
charts = [] # List of charts
# Index of the column "Monthly Expenses"
idx = list(df.columns).index("Monthly Expenses")

# Create the charts
for column in df.columns:
    # Create the charts for each column in the dataframe except the last one
    if column != "Monthly Expenses":
        # Create and save the object for interactive selection for each chart
        multi.append(alt.selection_multi(encodings = ['x']))
        # Create the chart and save it in the charts list
        charts.append(
            alt.Chart(
                df # Data
            ).mark_bar(
            ).encode(
                x = alt.X( # X-axis
                    column,
                    axis = alt.Axis(labelAngle = -45)
                ),
                y = alt.Y( # Y-axis
                    "count()",
                    title = "Count"
                ),
                tooltip = ["count()"], # Tooltip
                color = alt.condition( # Color
                    multi[-1], # Object for interactive selection
                    alt.value(orange), # Selected
                    alt.value("lightgray") # Not selected
                )
            ).properties( # Properties (size of the chart)
                height = 100,
                width = 100
            )
        )
        # Add the interactive selection to the chart
        charts[-1] = charts[-1].add_selection(multi[-1])
    # Create the chart for the last column in the dataframe (principal chart)
    else:
        # Create the chart and save it in the list of charts
        charts.append(
            alt.Chart(
                df # Data
            ).mark_bar(
            ).encode(
                x = alt.X( # X-axis
                    column,
                    scale = alt.Scale(domain = (110, 360)),
                    title = "Monthly Expenses ($)"
                ),
                y = alt.Y( # Y-axis
                    "count()",
                    scale = alt.Scale(domain = (0, 21)),
                    title = "Count"
                ),
                tooltip = ["count()"] # Tooltip
            ).properties( # Properties (size of the chart)
                height = 100,
                width = 415
            )
        )

# Create the boxplot of the Monthly Expenses
boxplot = alt.Chart(
    df # Data
).mark_boxplot( # Boxplot
    median = {'color': 'black'}, # Color of the median line
    outliers = {'size': 5} # Size of the outliers points
).encode(
    y = alt.Y( # Y-axis
        "Monthly Expenses",
        scale = alt.Scale(domain=(0, 350))
        ),
).properties( # Properties (size of the chart)
    height = 100,
    width = 100
)

# Adjust interactivity
for _multi in multi:
    # Add the response of interactive selection to the histogram of the Monthly
    # Expenses
    charts[idx] = charts[idx].transform_filter(
        _multi
    )
    # Add the response of interactive selection to the boxplot
    boxplot = boxplot.transform_filter(
        _multi
    )
```

Agora, vamos compor os gráficos, utilizando a função de concatenação horizontal (`hconcat()` ou `|`) e concatenação vertical (`vconcat()` ou `&`) do Altair. A ferramenta dispões de outras maneiras de compor múltiplos gráficos, mas foi escolhida dessa maneira pela versatilidade que ela oferece, sendo possível escolher a posição dos gráficos de forma simplificada e personalizá-los um a um (até certo limite) antes de uní-los.

```
# Create the multi-chart
result = \
    (charts[0] | charts[1] | charts[2] | charts[3]) & \
    (charts[4] | charts[5] | charts[6] | charts[7]) & \
    (charts[8] | charts[9] | charts[10] | charts[11]) & \
    (charts[12] | boxplot)
```

Por fim, podemos fazer alguns ajustes finais e plotar o resultado.

```
# Configure the result
result = result.configure_mark(color = orange)
result = result.configure_axis(
    labelColor = black,
    titleColor = black
)
result = result.configure_title(color = black)
result = result.properties(
    title=alt.TitleParams(
        text="University Students Monthly Expenses",
        offset=10,
        fontSize=20
    )
)

result
```

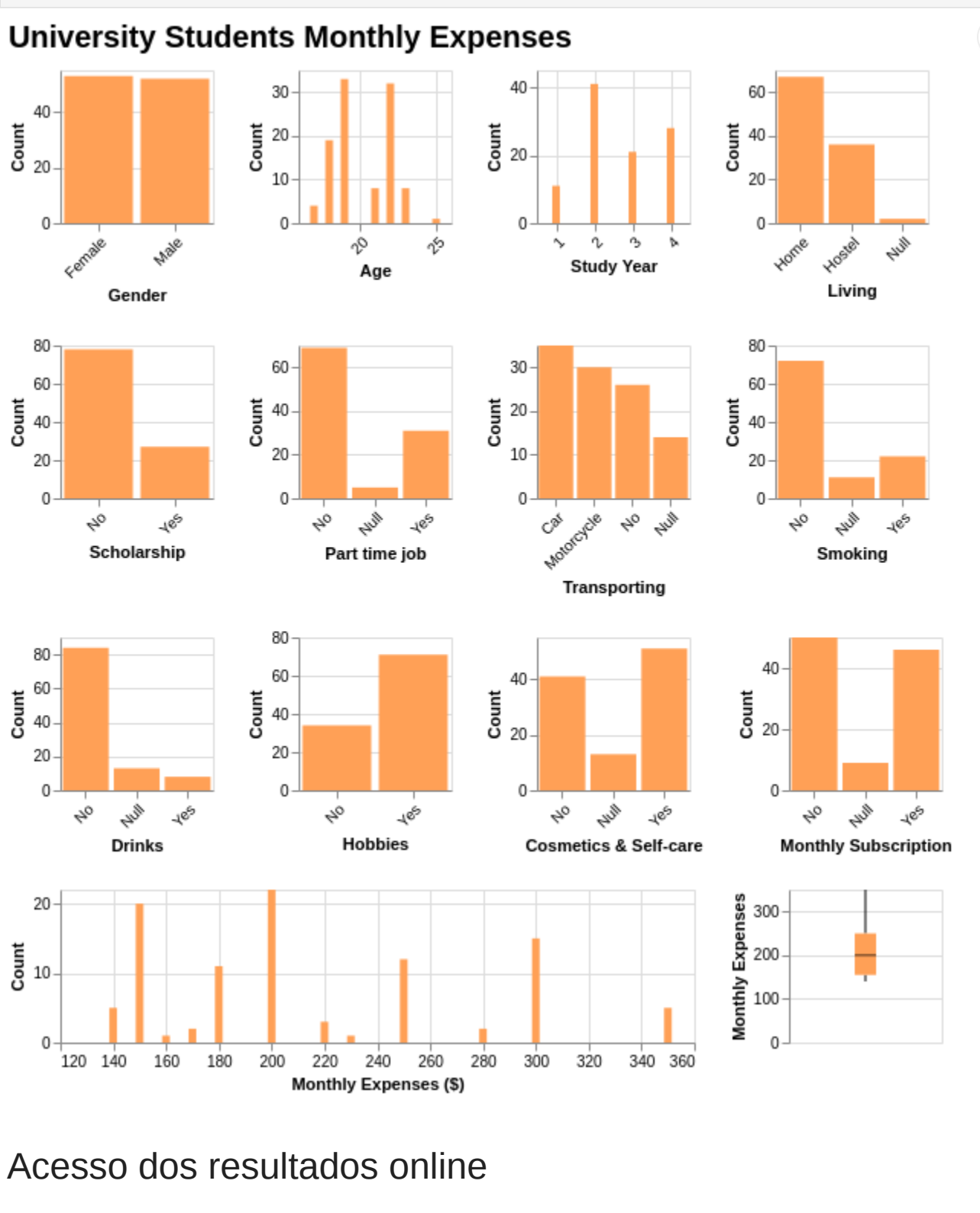


Devemos ressaltar que caso o leitor esteja vendo a versão em HTML (ou até mesmo executando as células do notebook) é possível interagir com o gráfico acima, clicando nas barras dos 12 primeiros gráficos de forma a filtrar os 2 últimos. Dessa maneira, é possível gerar múltiplos filtros utilizando os diferentes gráficos, e caso seja desejado selecionar mais de uma barra de um mesmo gráfico, é possível segurar a tecla `shift` enquanto seleciona as barras para que a anterior não seja desselecionada (também é possível remover as seleções com duplo clique).

Melhorando os resultados

Acima, vimos uma dashboard interativa na qual os últimos gráficos recebem filtros conforme interagimos com os outros. Após fazer isso, pode-se notar que os demais gráficos permanecem representando as contagens dos dados iniciais, não filtrados, e parece razoável que eles também respondam aos filtros realizados, de forma que todos os gráficos correspondem ao mesmo tempo ao mesmo conjunto de dados (filtrados). Dessa forma, realizamos poucas modificações nos códigos que já vimos assim, para produzir um novo resultado final, como desejado. Segue abaixo o código e o resultado (dessa vez o código está todo junto em uma única célula do notebook, visto que já explicamos melhor o passo a passo anteriormente):

```
# Pallette
orange = "#FFA056"
black = "black"

# Initial variables
multi = [] # List of objects for interactive selection
charts = [] # List of charts
# Index of the column "Monthly Expenses"
idx = list(df.columns).index("Monthly Expenses")

# Create the charts
for column in df.columns:
    # Create and save the object for interactive selection for each chart
    multi.append(alt.selection_multi(encodings = ['x']))
    # Create the chart and save it in the charts list
    charts.append(
        alt.Chart(
            df # Data
        ).mark_bar(
        ).encode(
            x = alt.X( # X-axis
                column,
                axis = alt.Axis(labelAngle =
                    -45 if column != "Monthly Expenses" else 0
                ),
                title = column if column != "Monthly Expenses" else column + " ($)",
            ),
            y = alt.Y( # Y-axis
                "count()",
                title = "Count"
            ),
            tooltip = ["count()"], # Tooltip
            color = alt.condition( # Color
                multi[-1], # Object for interactive selection
                alt.value(orange), # Selected
                alt.value("lightgray") # Not selected
            )
        )
    )
    # Configure the size of the charts
    if column != "Monthly Expenses":
        charts[-1] = charts[-1].properties( # Properties (size of the chart)
            height = 100,
            width = 100
        )
    else:
        charts[-1] = charts[-1].properties( # Properties (size of the chart)
            height = 100,
            width = 415
        )

    # Add the interactive selection to the chart
    charts[-1] = charts[-1].add_selection(multi[-1])

# Create the boxplot of the Monthly Expenses
boxplot = alt.Chart(
    df # Data
).mark_boxplot( # Boxplot
    median = {'color': 'black'}, # Color of the median line
    outliers = {'size': 5} # Size of the outliers points
).encode(
    y = alt.Y( # Y-axis
        "Monthly Expenses",
        scale = alt.Scale(domain=(0, 350))
        ),
).properties( # Properties (size of the chart)
    height = 100,
    width = 100
)

# Adjust interactivity
for _multi in range(len(multi)):
    # Add the response of interactive selection to the bar charts
    for chart in range(len(charts)):
        # Don't add the interative filter to the same chart of the interactive
        # selection
        if chart != _multi:
            # Add filter to the chart
            charts[chart] = charts[chart].transform_filter(
                multi[_multi]
            )
    # Add the response of interactive selection to the boxplot chart (filter)
    boxplot = boxplot.transform_filter(
        multi[_multi]
    )

# Create the multi-chart
result = \
    (charts[0] | charts[1] | charts[2] | charts[3]) & \
    (charts[4] | charts[5] | charts[6] | charts[7]) & \
    (charts[8] | charts[9] | charts[10] | charts[11]) & \
    (charts[12] | boxplot)


# Configure the result
result = result.configure_mark(color = orange)
result = result.configure_axis(
    labelColor = black,
    titleColor = black
)
result = result.configure_title(color = black)
result = result.properties(
    title=alt.TitleParams(
        text="University Students Monthly Expenses",
        offset=10,
        fontSize=20
    )
)

result
```



Acesso dos resultados online

Caso o leitor esteja vendo a versão em PDF, é possível acessar o repositório com esse notebook (e também a versão HTML e em PDF) no repositório: https://github.com/BrunoFornaro/Visualizacao-da-Informacao---Multiplos-Graficos. Além disso, é possível ver online o HTML com o gráfico interativo em: https://brunofornaro.github.io/Visualizacao-da-Informacao---Multiplos-Graficos/multiple_graphs.html.

Conclusões Finais

Para esse trabalho, primeiramente, foi necessário escolher uma base de dados na qual nos permitisse obter um resultado interessante com múltiplos gráficos. Para isso, o primeiro desafio foi conseguir encontrar uma base de dados com essas características, pois como os dados eram dispostos seria muito importante. Uma das primeiras tentativas, com outra base de dados aberta (sobre Airbnb em Nova York), foi de fazer múltiplos `scatter plots`, assim como muitas bibliotecas de gráficos implementam por padrão. O Altair tem uma funcionalidade para plotar múltiplos gráficos dessa forma, onde podemos especificar quais dados desejamos que sejam comparados em cada célula da matriz de gráficos. Porém, esse recurso é um pouco limitante, e o resultado obtido com ele não foi tão satisfatório, ainda mais pois já havia a intenção de além de trabalhar com múltiplos gráficos, também implementar interatividade entre eles.

Dessa maneira, para chegar no resultado obtido, foi utilizada uma base de dados sobre os gastos de estudantes universitários, que têm diversos dados categóricos e outros numéricos discretos, que nos possibilitou fazer gráficos de contagens (e histogramas) para cada dado. Com isso, pudemos unir esses gráficos com as funções de concatenação horizontal (`|`) e vertical (`&`) do Altair, facilmente. Além disso, plotamos juntamente com um histograma e com um boxplot dos gastos dos estudantes e adicionamos interatividade de tal forma que é possível selecionar as barras dos gráficos de barras de forma a filtrar os dados de todos os gráficos (embora que, num primeiro momento, os últimos dois tenham sido considerados os "principais"). Nesse contexto, pudemos ver que a base de dados utilizada é "pequena" (com apenas 105 linhas), o que pode comprometer um pouco o resultado final (pelos gráficos ficarem com ainda menos dados a cada filtro, cada vez mais simples e vazios), mas ainda assim é suficiente para vermos como a implementação está funcionando e poderia facilmente escalar para outros dados de forma semelhante.

Dessa forma, ao longo deste trabalho foi possível desenvolver ainda mais o conhecimento sobre a biblioteca Altair e utilizar melhor algumas das suas funcionalidades que ainda não haviam sido bem exploradas. Nesse mesmo sentido, enquanto eram exploradas outras bibliotecas, foi possível perceber uma limitação da biblioteca com bases de dados muito grandes. Por padrão, a Altair limita a quantidade de linhas da base de dados utilizada para 5000, pois os dados são inseridos no resultado final do notebook (ou HTML), de forma que ao aumentar a quantidade de dados o tamanho do arquivo também fica muito grande. Entretanto, é possível contornar isso ao desativar o erro que isso gera, e também importando os dados para o documento de outra forma, sem "embutí-los", como, por exemplo, fazendo o Altair ler um arquivos JSON externo (que ele mesmo pode gerar a partir do `dataframe` do `pandas`), ou até mesmo ler os dados a partir de uma URL. De toda forma, foi preferível para esse caso, que não era necessário trabalhar com muitos dados realmente, utilizar uma base de dados menor, pois seria mais fácil para trabalhar com ela, sem comprometer a velocidade para plotar os gráficos e eles responderem a interatividade, além de evitar gerar arquivos muito grandes.

Referências bibliográficas

ADOBE. Adobe Color. Disponível em: color.adobe.com/pt/create/. Acessado em 22 de outubro de 2022.

ALTAIR. Top-Level Chart Configuration. Disponível em: altair-viz.github.io/. Acessado em 22 de outubro de 2022.

ISLAM, Shariful. University Students Monthly Expenses. Disponível em: www.kaggle.com/. Acessado em 22 de outubro de 2022.

ALTAIR. Interactivity and Selections. Disponível em: altair-viz.github.io/. Acessado em 22 de outubro de 2022.