In [ ]:

```python
#import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import geopandas as gpd
from geobr import read_state
import matplotlib.colors as mcolors
import matplotlib.ticker as ticker

#format float numbers to not use scientific notation
pd.options.display.float_format = '{0:.2f}'.format
```

In [ ]:

```python
#import data from .csv to pandas dataframe
df_data = pd.read_csv(r'C:\Users\Mariano\Documents\Aprendizaje Data Science\Electrical
```

In [ ]:

```python
#see some information about the dataframe
df_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25909 entries, 0 to 25908
Data columns (total 23 columns):
 #   Column                        Non-Null Count  Dtype 
---  ------                        --------------  ----- 
 0   DatGeracaoConjuntoDados       25909 non-null  object
 1   NomEmpreendimento             25909 non-null  object
 2   IdeNucleoCEG                  25909 non-null  int64 
 3   CodCEG                        25909 non-null  object
 4   SigUFPrincipal                25909 non-null  object
 5   SigTipoGeracao                25909 non-null  object
 6   DscFaseUsina                  25909 non-null  object
 7   DscOrigemCombustivel          25909 non-null  object
 8   DscFonteCombustivel           25909 non-null  object
 9   DscTipoOutorga                25909 non-null  object
 10  NomFonteCombustivel           25909 non-null  object
 11  DatEntradaOperacao            25909 non-null  object
 12  MdaPotenciaOutorgadaKw        25909 non-null  object
 13  MdaPotenciaFiscalizadaKw      25909 non-null  int64 
 14  MdaGarantiaFisicaKw           25909 non-null  object
 15  IdcGeracaoQualificada         9951 non-null   object
 16  NumCoordNEmpreendimento       25909 non-null  object
 17  NumCoordEEmpreendimento       25909 non-null  object
 18  DatInicioVigencia             6572 non-null   object
 19  DatFimVigencia                6562 non-null   object
 20  DscPropriRegimePariticipacao  25909 non-null  object
 21  DscSubBacia                   1445 non-null   object
 22  DscMuninicpios                25909 non-null  object
dtypes: int64(2), object(21)
memory usage: 4.5+ MB
```

In [ ]:

```python
#count null values for each column
df_data.isnull().sum()
```

Out[ ]:

```
DatGeracaoConjuntoDados              0
NomEmpreendimento                    0
IdeNucleoCEG                         0
CodCEG                               0
SigUFPrincipal                       0
SigTipoGeracao                       0
DscFaseUsina                         0
DscOrigemCombustivel                 0
DscFonteCombustivel                  0
DscTipoOutorga                       0
NomFonteCombustivel                  0
DatEntradaOperacao                   0
MdaPotenciaOutorgadaKw               0
MdaPotenciaFiscalizadaKw             0
MdaGarantiaFisicaKw                  0
IdcGeracaoQualificada            15958
NumCoordNEmpreendimento              0
NumCoordEEmpreendimento              0
DatInicioVigencia                19337
DatFimVigencia                   19347
DscPropriRegimePariticipacao         0
DscSubBacia                      24464
DscMuninicpios                       0
dtype: int64
```

In [ ]:

```python
#search for duplicates rows
df_data[df_data.duplicated(keep = False)]
```

Out[ ]:

DatGeracaoConjuntoDados	NomEmpreendimento	IdeNucleoCEG	CodCEG	SigUFPrincipal	SigTipoGe

0 rows × 23 columns

The data frame has 25909 rows with no duplicated rows. There is some null values in some columns, te interesting columns to analyze are "DatInicioVigencia" and "DatFimVigencia"

In [ ]:

```python
#set interesting columns format from Object to DateTime
df_data['DatEntradaOperacao'] = pd.to_datetime(df_data['DatEntradaOperacao'])
df_data['DatInicioVigencia'] = pd.to_datetime(df_data['DatInicioVigencia'])
df_data['DatFimVigencia'] = pd.to_datetime(df_data['DatFimVigencia'])

#set interesting columns format from Object to float
#defining function to change comma decima separator to dot decimal separator
def comma_to_dot(x):
    try:
        return float(x.replace(',', '.'))
    except ValueError:
        return print('The comma_to_dot function has an error')

#use previous function to change the decimal separator from comma to dot
df_data['MdaPotenciaOutorgadaKw'] = df_data['MdaPotenciaOutorgadaKw'].apply(comma_to_d
df_data['MdaPotenciaFiscalizadaKw'] = df_data['MdaPotenciaFiscalizadaKw'].astype('floa
df_data['MdaGarantiaFisicaKw'] = df_data['MdaGarantiaFisicaKw'].apply(comma_to_dot).as

#check the change of columns type
df_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25909 entries, 0 to 25908
Data columns (total 23 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   DatGeracaoConjuntoDados        25909 non-null  object
 1   NomEmpreendimento              25909 non-null  object
 2   IdeNucleoCEG                   25909 non-null  int64
 3   CodCEG                         25909 non-null  object
 4   SigUFPrincipal                 25909 non-null  object
 5   SigTipoGeracao                 25909 non-null  object
 6   DscFaseUsina                   25909 non-null  object
 7   DscOrigemCombustivel           25909 non-null  object
 8   DscFonteCombustivel            25909 non-null  object
 9   DscTipoOutorga                 25909 non-null  object
 10  NomFonteCombustivel            25909 non-null  object
 11  DatEntradaOperacao             25909 non-null  datetime64[ns]
 12  MdaPotenciaOutorgadaKw         25909 non-null  float64
 13  MdaPotenciaFiscalizadaKw       25909 non-null  float64
 14  MdaGarantiaFisicaKw            25909 non-null  float64
 15  IdcGeracaoQualificada          9951 non-null   object
 16  NumCoordNEmpreendimento        25909 non-null  object
 17  NumCoordEEmpreendimento        25909 non-null  object
 18  DatInicioVigencia              6572 non-null   datetime64[ns]
 19  DatFimVigencia                 6562 non-null   datetime64[ns]
 20  DscPropriRegimePariticipacao   25909 non-null  object
 21  DscSubBacia                    1445 non-null   object
 22  DscMuninicpios                 25909 non-null  object
dtypes: datetime64[ns](3), float64(3), int64(1), object(16)
memory usage: 4.5+ MB
```

The datetime columns are: DatEntradaOperacao: the beginning of operation time of the first generator unit. DatInicioVigencia: the beginning of the permit for generate electricity for the generator. DatFimVigencia: the end of the permit for generate electricity for generator.

In [ ]:

```python
"""compare the columns DatInicioVigencia and DatEntradaOperacao
to see if the beginning of one of them is projected for the future"""
#creation of auxiliary variables
aux_datacheck = 0
aux_datacheck1 = []
aux_datacheck2 = 0
for i in range(0, len(df_data['DatEntradaOperacao'])):
    if df_data['DatEntradaOperacao'][i] > pd.to_datetime('today'):
        aux_datacheck += 1
        aux_datacheck1.append(df_data['DatEntradaOperacao', 'DatInicioVigencia'][i])

print(aux_datacheck1)
```

```
[]
```

There is no Generator that begins to generate in a future date, so we can consider every generator for our analysis.

In [ ]:

```python
#check basic descriptive statistics for electrical power columns of generators
df_data[['MdaPotenciaOutorgadaKw', 'MdaPotenciaFiscalizadaKw', 'MdaGarantiaFisicaKw']]
```

Out[ ]:

	MdaPotenciaOutorgadaKw	MdaPotenciaFiscalizadaKw	MdaGarantiaFisicaKw
count	25909.00	25909.00	25909.00
mean	14336.92	7675.40	3706.45
std	122436.33	120237.58	74096.98
min	0.16	0.00	0.00
25%	1.00	1.00	0.00
50%	1.38	1.00	0.00
75%	4352.00	1.00	0.00
max	11233100.00	11233100.00	7750800.00

The columns of interest of electric power of every generator are: MdaPotenciaOutorgadaKw: installed electric power MdaPotenciaFiscalizadaKw: declarated available electric power

In [ ]:

```python
#check unique values of object data type columns:
for col in df_data.select_dtypes(include=['object']).columns:
    print(f"\nUnique values in {col}:")
    print(df_data[col].unique())
```

```
Unique values in DatGeracaoConjuntoDados:
['2024-02-01']

Unique values in NomEmpreendimento:
['E' 'F' 'G' ... 'Vsc Industria E Comercio De Embalagens Plasticas'
 'Sueli Xavier Rodrigues' 'Supermercado Guedes']

Unique values in CodCEG:
['PCH.PH.MG.000008-6.1' 'PCH.PH.MG.000009-4.1' 'PCH.PH.MG.000010-8.1' ...
 'UFV.RS.SC.073873-5.1' 'UFV.RS.GO.073874-3.1' 'UFV.RS.PB.073970-7.1']

Unique values in SigUFPrincipal:
['MG' 'RS' 'SC' 'TO' 'RR' 'MT' 'SP' 'ES' 'RO' 'AM' 'RJ' 'PR' 'CE' 'BA'
 'MA' 'PI' 'AL' 'GO' 'PB' 'AP' 'MS' 'PE' 'PA' 'DF' 'SE' 'RN' 'AC']

Unique values in SigTipoGeracao:
['PCH' 'UHE' 'CGH' 'UTE' 'UTN' 'EOL' 'UFV']

Unique values in DscFaseUsina:
['Operação' 'Construção não iniciada' 'Construção']

Unique values in DscOrigemCombustivel:
['Hídrica' 'Fóssil' 'Biomassa' 'Nuclear' 'Eólica' 'Solar']

Unique values in DscFonteCombustivel:
['Potencial hidráulico' 'Carvão mineral' 'Petróleo' 'Agroindustriais'
 'Gás natural' 'Urânio' 'Floresta' 'Resíduos sólidos urbanos'
 'Cinética do vento' 'Radiação solar' 'Outros Fósseis' 'Resíduos animais'
 'Biocombustíveis líquidos']

Unique values in DscTipoOutorga:
['Autorização' 'Concessão' 'Registro']

Unique values in NomFonteCombustivel:
['Potencial hidráulico' 'Gás de Alto Forno - CM' 'Óleo Diesel'
 'Bagaço de Cana de Açúcar' 'Gás Natural' 'Urânio' 'Licor Negro'
 'Óleo Combustível' 'Calor de Processo - CM'
 'Outros Energéticos de Petróleo' 'Carvão Mineral' 'Resíduos Florestais'
 'Calor de Processo - GN' 'Gás de Refinaria' 'Biogás - RU'
 'Cinética do vento' 'Lenha' 'Casca de Arroz' 'Radiação solar'
 'Carvão Vegetal' 'Gás de Alto Forno - PE' 'Gás de Alto Forno - Biomassa'
 'Calor de Processo - OF' 'Biogás - RA' 'Capim Elefante' 'Óleos vegetais'
 'Biogás-AGR' 'Resíduos Sólidos Urbanos - RU' 'Etanol' 'Carvão - RU']

Unique values in IdcGeracaoQualificada:
['Não' 'Sim' nan]

Unique values in NumCoordNEmpreendimento:
['-20,12479858' '-20,13187300' '-20,13754468' ... '-20,03777778'
 '-3,05509444' '-16,80438889']

Unique values in NumCoordEEmpreendimento:
['-43,87020250' '-43,87693500' '-43,89192620' ... '-44,24694444'
 '-60,02339722' '-49,20592500']

Unique values in DscPropriRegimePariticipacao:
['100% para ANGLOGOLD ASHANTI CÓRREGO DO SÍTIO MINERAÇÃO S.A. (APE)'
 '100% para COMPANHIA ENERGÉTICA RIO DAS ANTAS (PIE)'
 '100% para Cooperativa de Geração de Energia e Desenvolvimento (REG)' ...
 '100% para VSC INDUSTRIA E COMERCIO DE EMBALAGENS PLASTICAS LTDA (REG)'
 '100% para SUELI XAVIER RODRIGUES (REG)'
 '100% para LUIZ GUEDES SOBRINHO LTDA (REG)']

Unique values in DscSubBacia:
[' 41 - Das Velhas - Sao Francisco' ' 86 - Taquari'
 ' 72 - Uruguai, Inhanduva, Peixe e outros'
 ' 73 - Uruguai, Chapeco,Passo Fundo e outros' nan
 ' 21 - Tocantins, entre os rios Preto e Parana'
 ' 26 - Araguaia, trecho da ilha Bananal' ' 61 - Grande' ' 56 - Doce'
 ' 81 - Ribeira do Iguape' ' 57 - Litoraneas do Espirito Santo'
 ' 15 - Madeira' ' 24 - Alto Araguaia e rio Claro' ' 83 - Itajai'
 ' 16 - Amazonas, entre os rios Madeira e Trombetas' ' 66 - Alto Paraguai'
 ' 62 - Parana, Tiete e outros' ' 58 - Paraiba do Sul' ' 17 - Tapajos'
 ' 64 - Parana, Paranapanema, Amambai e outros' ' 18 - Xingu e Paru'
 ' 22 - Tocantins, entre os rios Parana e do Sono' ' 60 - Paranaiba'
 ' 65 - Parana, Iguacu' ' 34 - Parnaiba'
 ' 82 - Cachoeira, Sao Joao e outros'
 ' 40 - Alto Sao Francisco, ate Tres Marias'
 ' 87 - Camaqua, Jacui, lagoa dos Patos e outros'
 ' 74 - Uruguai, Varzea, Turvo e outros' ' 39 - Paraiba e outros'
 ' 84 - Tubarao, Capivari e outros'
 ' 75 - Uruguai, Ijui, piratinim e outros' ' 85 - Alto Jacui'
 ' 71 - Canoas' ' 59 - Litoraneas do Rio de Janeiro'
 ' 30 - Oiapoque e outros' ' 63 - Parana, Verde , Peixe, e outros'
 ' 46 - Grande e outros - Sao Francisco' ' 76 - Ibicui'
 ' 80 - Litoraneas de Sao Paulo' ' 54 - Jequitinhonha'
 ' 49 - Sao Francisco, a jusante do Pajeu' ' 28 - Baixo Araguaia'
 ' 53 - Pardo, Cachoeira e outros'
 ' 45 - Corrente e outros - Sao Francisco'
 ' 42 - Paracatu e outros - Sao Francisco'
 ' 55 - Sao mateus, Itanhem e outros' ' 20 - Alto Tocantins e rio Preto'
 ' 47 - Salitre e outros - Sao Francisco'
 ' 29 - Tocantins, entre o rio Araguaia e a foz'
 ' 19 - Amazonas, entre o rio Xingu e a foz' ' 52 - Contas'
 ' 50 - Itapicuru, Vaza Barris e outros' ' 37 - Piranhas, Acu e outros'
 ' 70 - Pelotas' ' 39 - Litoraneas de Pernambuco e Alagoas'
 ' 51 - Jequirica, Paraguacu e outros'
 ' 23 - Tocantins, entre os rios do Sono e Araguaia'
 ' 43 - Urucuia - Sao Francisco' ' 32 - Litoraneas do Para e Maranhao'
 ' 31 - Guama e outros']

Unique values in DscMuninicpios:
['Nova Lima - MG' 'Bento Gonçalves - RS, Cotiporã - RS'
 'Floriano Peixoto - RS' ... 'Pendências - RN' 'Cascavel - CE'
 'Patos - PB']
```

The interesting columns from these analysis and taking into consideration the description provided by the ANEL are: DscFaseUsina: current status of operation of the power plant (operative, projected, in construction).

SigUFPrincipal: Location in state of brazil (abbreviated).

SigTipoGeracao: type of electric generator.

DscOrigemCombustivel: type of fuel used to generate electricity.

DscFonteCombustivel: source of the fuel used to generate electricity.

NumCoordEEmprendimento: geographic coordinates of the electric power plant.

NumCoordNEmprendimento: geographic coordinates of the electric power plant.

Create Dataframes for graphs:

In [ ]:

```
#import geoespatial data for states of brazil
brazilian_states = read_state(code_state = 'all')
#print(brazilian_states)
```

In [ ]:

```
#dataframes for pie/bar chart Electric Power vs Fuel Type
type_fuel_op = df_data[df_data['DscFaseUsina']== 'Operação'].groupby('DscOrigemCombust
type_fuel_proj = df_data[df_data['DscFaseUsina']== 'Construção não iniciada'].groupby(
type_fuel_constr = df_data[df_data['DscFaseUsina']== 'Construção'].groupby('DscOrigemC

#dataframes for pie/bar chart Electric Power vs Generator Type
type_generator_op = df_data[df_data['DscFaseUsina']== 'Operação'].groupby('SigTipoGera
type_generator_proj = df_data[df_data['DscFaseUsina']== 'Construção não iniciada'].grc
type_generator_constr = df_data[df_data['DscFaseUsina']== 'Construção'].groupby('SigTi

#dataframes for bar chart and choropleth map Electric Power vs State of Brazil
#first groupby, then merge with geodata of brazil, last drop unused columns
states_power_op = df_data[df_data['DscFaseUsina']== 'Operação'].groupby('SigUFPrincipa
states_power_op = pd.merge(brazilian_states, states_power_op, right_on = 'SigUFPrincip
states_power_op = states_power_op.drop(columns=['code_region', 'abbrev_state', 'code_s

states_power_proj = df_data[df_data['DscFaseUsina']== 'Construção não iniciada'].group
states_power_proj = pd.merge(brazilian_states, states_power_proj, right_on = 'SigUFPri
states_power_proj = states_power_proj.drop(columns =['code_region', 'abbrev_state', 'c

states_power_constr = df_data[df_data['DscFaseUsina']== 'Construção'].groupby('SigUFPr
states_power_constr = pd.merge(brazilian_states, states_power_constr, right_on = 'SigU
states_power_constr = states_power_constr.drop(columns = ['code_state', 'abbrev_state'

#adicional data
total_power_op = type_fuel_op['MdaPotenciaOutorgadaKw'].sum()
total_power_proj = type_fuel_proj['MdaPotenciaOutorgadaKw'].sum()
total_power_constr = type_fuel_constr['MdaPotenciaOutorgadaKw'].sum()
```

Usefull function for the graphs:

In [ ]:

```
#set style for data visualization with matplotlib
plt.style.use('fivethirtyeight')
#function to add label to the top of each bar
def add_label_top_bar(ax, spacing=5):
    """Add labels to the end of each bar in a bar chart.

    Arguments:
        ax (matplotlib.axes.Axes): The matplotlib object containing the axes of the pl
        spacing (int): The distance between the labels and the bars.
    """
    # For each bar: Place a label
    for bar in ax.patches:
        # Get X and Y coordinates of the top center of bar for label.
        y_value = bar.get_height()
        x_value = bar.get_x() + bar.get_width() / 2
        # Number of points between bar and label. Change to your liking.
        space = spacing
        # Vertical alignment for positive values
        va = 'bottom'

        # If value of bar is negative: Place label below bar
        if y_value < 0:
            # Invert space to place label below
            space *= -1
            # Vertically align label at top
            va = 'top'


        # Use Y value as label and format number with commas and no decimal
        label = "{:,}".format(int(y_value))

        # Create annotation
        ax.annotate(
            label,                      # Use `label` as label
            (x_value, y_value),         # Place label at end of the bar
            xytext=(0, space),          # Vertically shift label by `space`
            textcoords="offset points", # Interpret `xytext` as offset in points
            ha='center',                # Horizontally center label
            va=va)

#function to generate colors for categories, to mantain the same color for each catego
def generate_color_dict(categories, colormap='Paired'):
    """
    Generate a color dictionary for given categories using a specified colormap.

    :param categories: List of category names
    :param colormap: Name of the colormap to use (default is 'Paired')
    :return: Dictionary mapping categories to colors
    """
    cmap = plt.get_cmap(colormap)
    colors = cmap(np.linspace(0, 1, len(categories)))
    return dict(zip(categories, [mcolors.rgb2hex(color) for color in colors]))

#function to get the color for the especified categories
def get_color(color_dict, categories):
    return [color_dict[category] for category in categories if category in color_dict]
```

In [ ]:

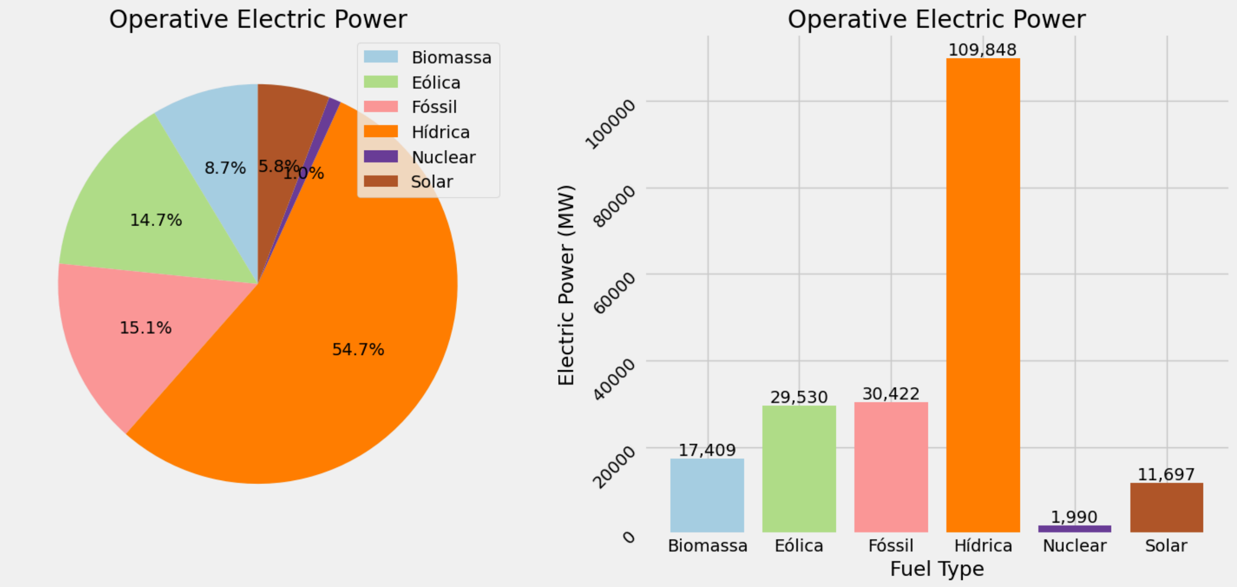
```
#define colors for graphs by fuel type
colors_type_fuel_dict = generate_color_dict(type_fuel_op['DscOrigemCombustivel'].uniqu
#define colors for graphs by generator type
colors_type_generator_dict = generate_color_dict(type_generator_op['SigTipoGeracao'].u
```

Pie chart and bar chart of Electric Power by Fuel Type:

In [ ]:

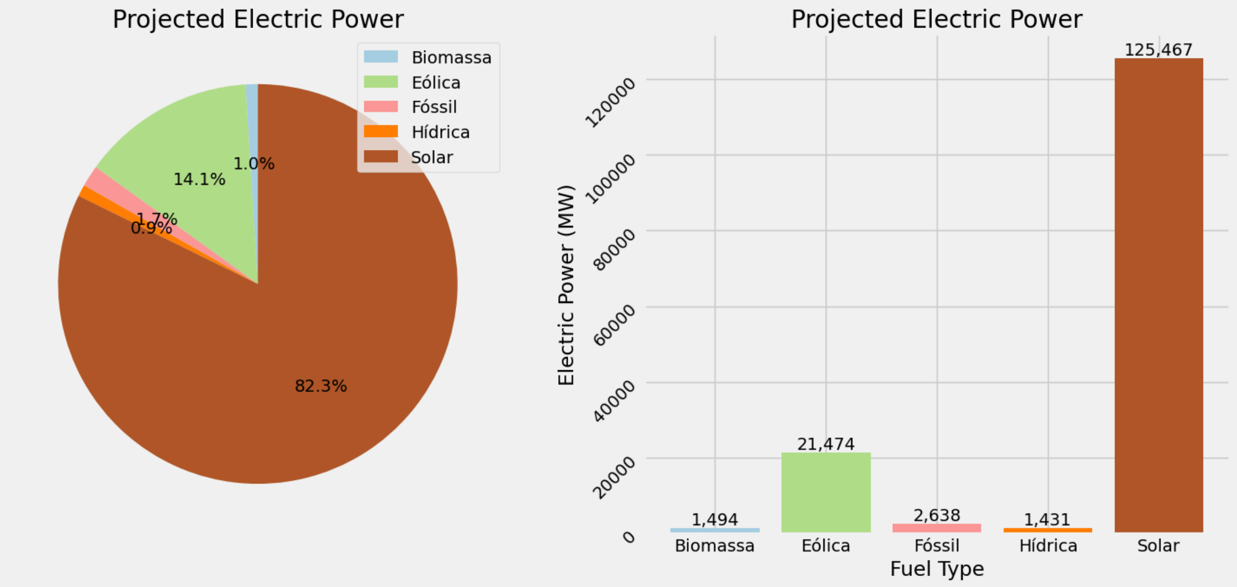
```
#bar and pie chart of Operative Electric Power by Fuel Type
#define the array for subplots
fig, ax= plt.subplots(1,2, figsize=(15,7))

#Pie Chart
wedges, texts, autotexts = ax[0].pie(
    type_fuel_op['MdaPotenciaOutorgadaKw'],
    autopct='%1.1f%%',
    colors=get_color(colors_type_fuel_dict, type_fuel_op['DscOrigemCombustivel']),
    startangle=90)
```

```
ax[0].set_title('Operative Electric Power')
ax[0].legend(type_fuel_op['DscOrigemCombustivel'])

#Bar chart
fuel_type_bar = ax[1].bar(
    type_fuel_op['DscOrigemCombustivel'],
    type_fuel_op['MdaPotenciaOutorgadaKw']/1000,
    color=get_color(colors_type_fuel_dict, type_fuel_op['DscOrigemCombustivel']))

ax[1].ticklabel_format(axis='y', style='plain')
ax[1].set_title('Operative Electric Power')
ax[1].set_ylabel('Electric Power (MW)')
ax[1].set_xlabel('Fuel Type')
#rotate y-axis label for better readability
plt.setp(ax[1].get_yticklabels(), rotation=45, ha='right')

#add value to top of each bar
add_label_top_bar(ax[1],0)

#print graph
plt.tight_layout()
plt.show()
```



In [ ]:

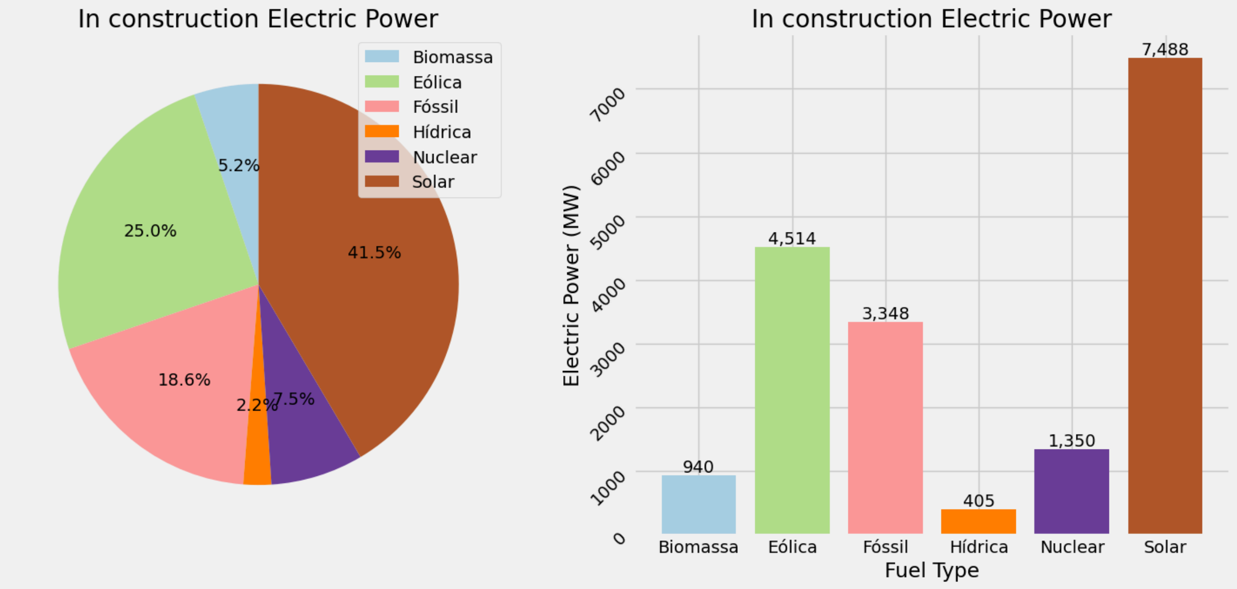
```
#bar and pie chart of Projected Electric Power by Fuel Type
#define the array for subplots
fig, ax= plt.subplots(1,2, figsize=(15,7))
#define colors for graphs
#colors= plt.cm.Set3(np.linspace(0, 1, len(type_fuel_proj['DscOrigemCombustivel'])))
#Pie Chart
wedges, texts, autotexts = ax[0].pie(type_fuel_proj['MdaPotenciaOutorgadaKw'],
                                     autopct='%1.1f%%',
                                     colors=get_color(colors_type_fuel_dict, type_fuel
                                     startangle=90)
ax[0].set_title('Projected Electric Power')
ax[0].legend(type_fuel_proj['DscOrigemCombustivel'])

#Bar chart
fuel_type_bar = ax[1].bar(
    type_fuel_proj['DscOrigemCombustivel'],
    type_fuel_proj['MdaPotenciaOutorgadaKw']/1000,
    color=get_color(colors_type_fuel_dict, type_fuel_proj['DscOrigemCombustivel']))

ax[1].ticklabel_format(axis='y', style='plain')
ax[1].set_title('Projected Electric Power')
ax[1].set_ylabel('Electric Power (MW)')
ax[1].set_xlabel('Fuel Type')
#rotate y-axis label for better readability
plt.setp(ax[1].get_yticklabels(), rotation=45, ha='right')

#add value to top of each bar
add_label_top_bar(ax[1],0)

#print graph
plt.tight_layout()
plt.show()
```



In [ ]:

```
#bar and pie chart of Projected Electric Power by Fuel Type
#define the array for subplots
fig, ax= plt.subplots(1,2, figsize=(15,7))
#define colors for graphs
colors= plt.cm.Set3(np.linspace(0, 1, len(type_fuel_constr['DscOrigemCombustivel'])))
#Pie Chart
wedges, texts, autotexts = ax[0].pie(
    type_fuel_constr['MdaPotenciaOutorgadaKw'],
    autopct='%1.1f%%',
    colors=get_color(colors_type_fuel_dict, type_fuel_constr['DscOrigemCombustivel']),
    startangle=90)

ax[0].set_title('In construction Electric Power')
ax[0].legend(type_fuel_constr['DscOrigemCombustivel'])

#Bar chart
fuel_type_bar = ax[1].bar(
    type_fuel_constr['DscOrigemCombustivel'],
    type_fuel_constr['MdaPotenciaOutorgadaKw']/1000,
    color=get_color(colors_type_fuel_dict, type_fuel_constr['DscOrigemCombustivel']))

ax[1].ticklabel_format(axis='y', style='plain')
ax[1].set_title('In construction Electric Power')
ax[1].set_ylabel('Electric Power (MW)')
ax[1].set_xlabel('Fuel Type')
```

```
#rotate y-axis label for better readability
plt.setp(ax[1].get_yticklabels(), rotation=45, ha='right')

#add value to top of each bar
add_label_top_bar(ax[1],0)

#print graph
plt.tight_layout()
plt.show()
```

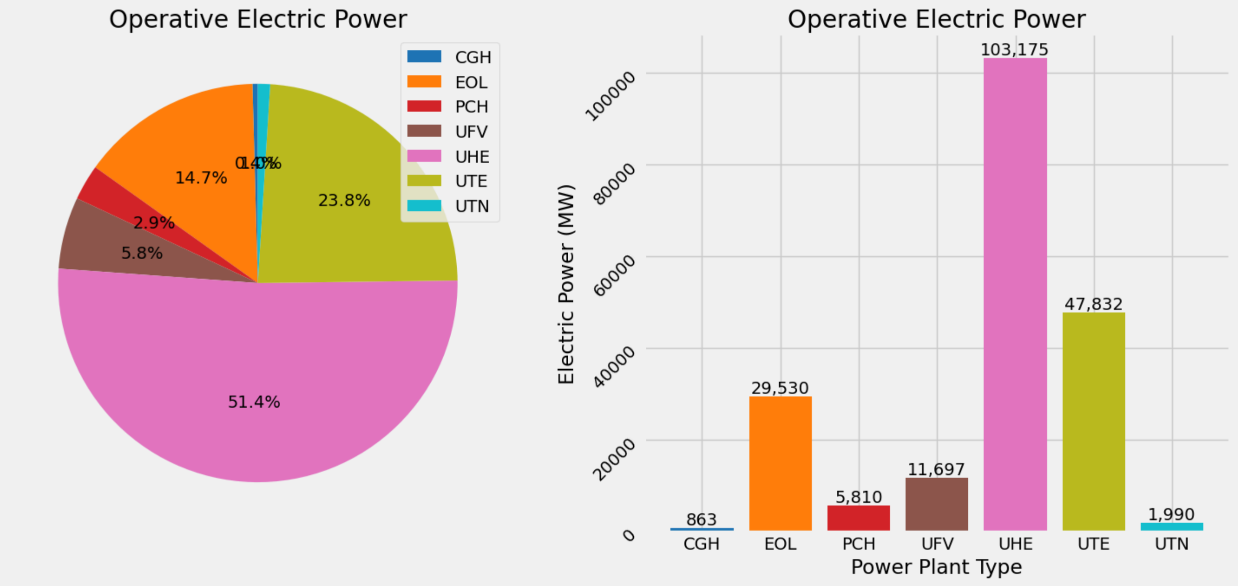


Pie chart and bar chart of Electric Power by Generator Power Plant Type:

In [ ]:

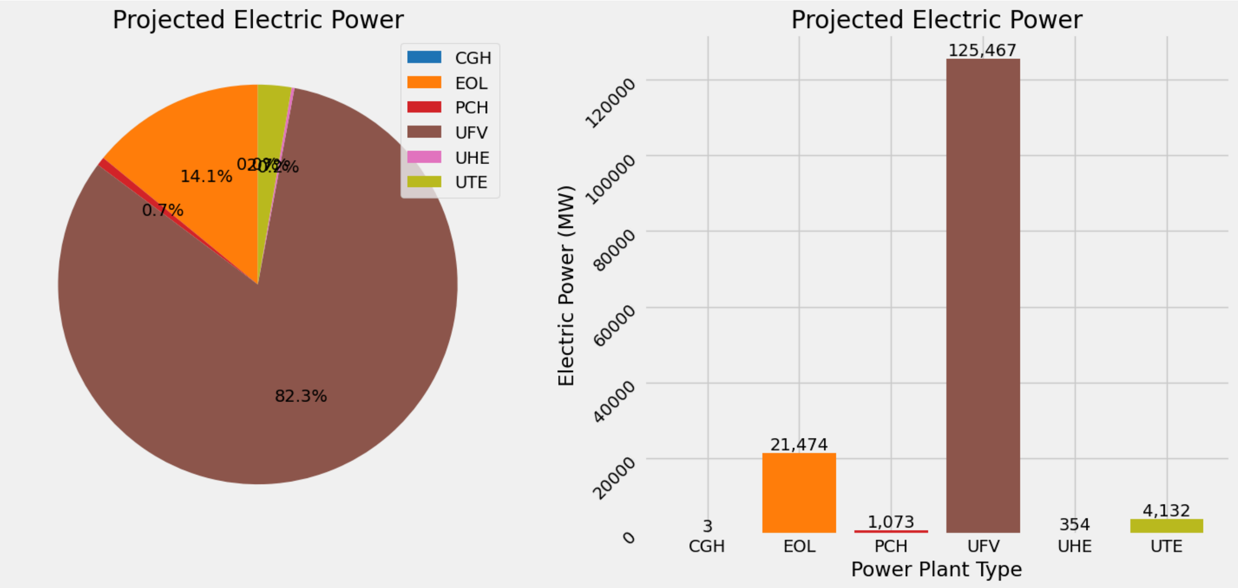
```
#bar and pie chart of Operative Electric Power by Power Plant Generator Type
#define the array for subplots
fig, ax= plt.subplots(1,2, figsize=(15,7))
#define colors for graphs
colors= plt.cm.Paired(np.linspace(0, 1, len(type_generator_op['SigTipoGeracao'])))
#Pie Chart
wedges, texts, autotexts = ax[0].pie(
    type_generator_op['MdaPotenciaOutorgadaKw'],
    autopct='%1.1f%%',
    colors=get_color(colors_type_generator_dict, type_generator_op['SigTipoGeracao']),
    startangle=90)

ax[0].set_title('Operative Electric Power')
ax[0].legend(type_generator_op['SigTipoGeracao'])

#Bar chart
fuel_type_bar = ax[1].bar(
    type_generator_op['SigTipoGeracao'],
    type_generator_op['MdaPotenciaOutorgadaKw']/1000,
    color=get_color(colors_type_generator_dict, type_generator_op['SigTipoGeracao']))

ax[1].ticklabel_format(axis='y', style='plain')
ax[1].set_title('Operative Electric Power')
ax[1].set_ylabel('Electric Power (MW)')
ax[1].set_xlabel('Power Plant Type')
#rotate y-axis label for better readability
plt.setp(ax[1].get_yticklabels(), rotation=45, ha='right')
```

```
#add value to top of each bar
add_label_top_bar(ax[1],0)

#print graph
plt.tight_layout()
plt.show()
```



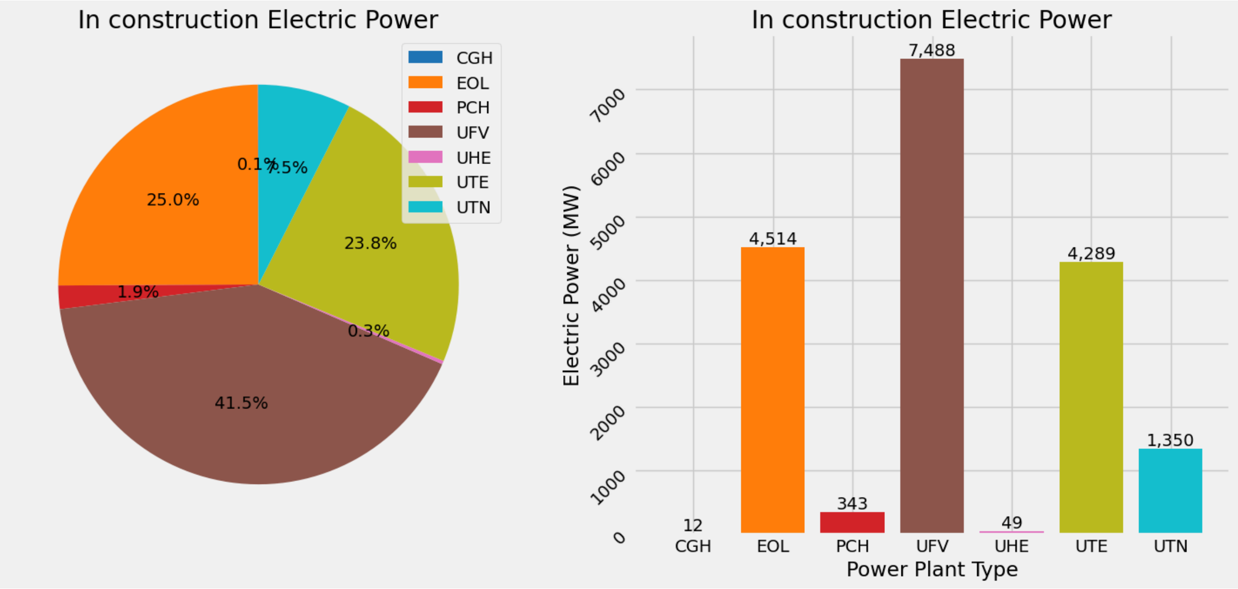
In [ ]:

```
#bar and pie chart of Projected Electric Power by Power Plant Generator Type
#define the array for subplots
fig, ax= plt.subplots(1,2, figsize=(15,7))

#Pie Chart
wedges, texts, autotexts = ax[0].pie(
    type_generator_proj['MdaPotenciaOutorgadaKw'],
    autopct='%1.1f%%',
    colors=get_color(colors_type_generator_dict, type_generator_proj['SigTipoGeracao']
    startangle=90)

ax[0].set_title('Projected Electric Power')
ax[0].legend(type_generator_proj['SigTipoGeracao'])

#Bar chart
fuel_type_bar = ax[1].bar(
    type_generator_proj['SigTipoGeracao'],
    type_generator_proj['MdaPotenciaOutorgadaKw']/1000,
    color=get_color(colors_type_generator_dict, type_generator_proj['SigTipoGeracao'])

ax[1].ticklabel_format(axis='y', style='plain')
ax[1].set_title('Projected Electric Power')
ax[1].set_ylabel('Electric Power (MW)')
ax[1].set_xlabel('Power Plant Type')
#rotate y-axis label for better readability
plt.setp(ax[1].get_yticklabels(), rotation=45, ha='right')

#add value to top of each bar
add_label_top_bar(ax[1],0)

#print graph
plt.tight_layout()
plt.show()
```



In [ ]:

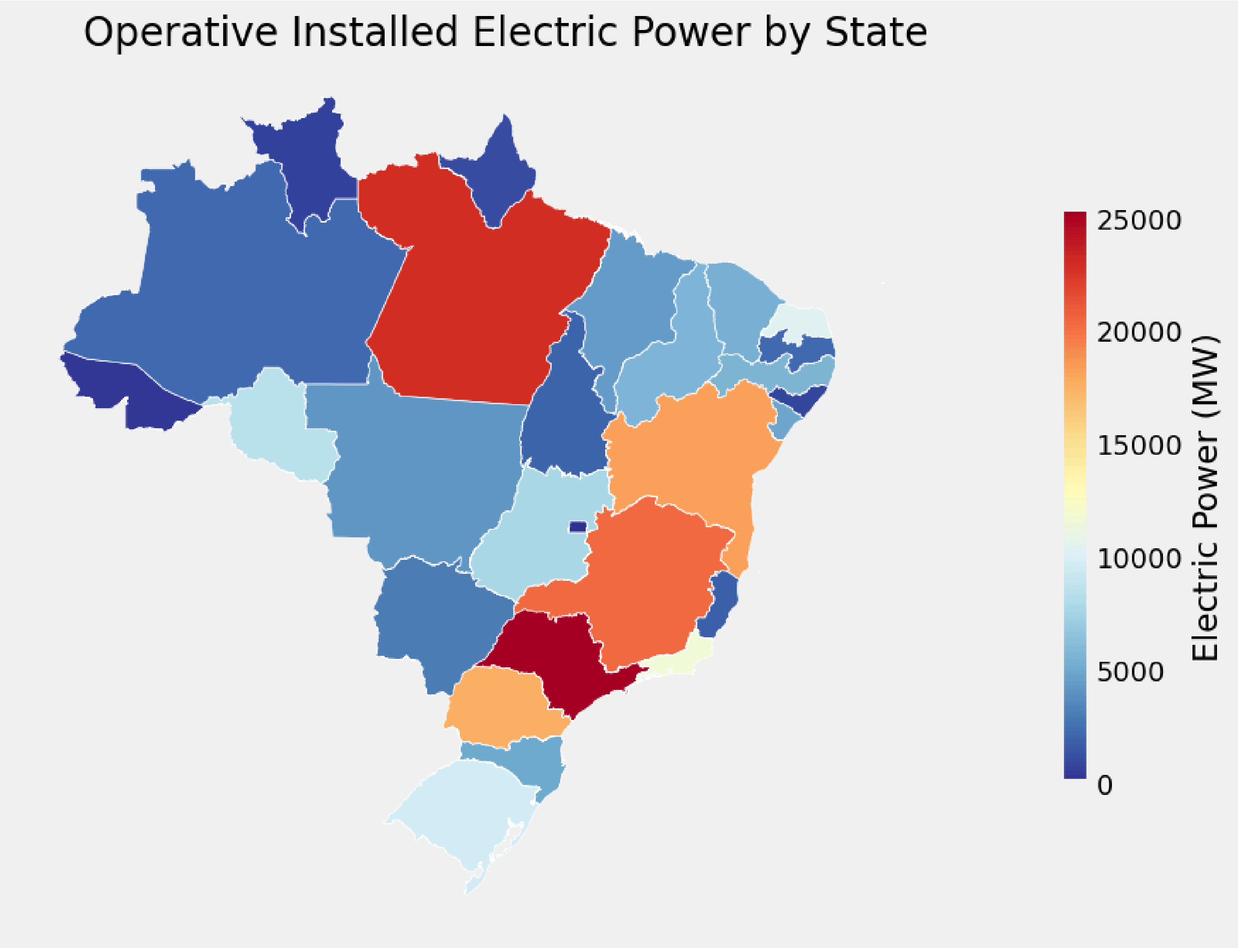
```
#bar and pie chart of In construction Electric Power by Power Plant Generator Type
#define the array for subplots
fig, ax= plt.subplots(1,2, figsize=(15,7))

#Pie Chart
wedges, texts, autotexts = ax[0].pie(
    type_generator_constr['MdaPotenciaOutorgadaKw'],
    autopct='%1.1f%%',
    colors=get_color(colors_type_generator_dict, type_generator_constr['SigTipoGeracac
    startangle=90)

ax[0].set_title('In construction Electric Power')
ax[0].legend(type_generator_constr['SigTipoGeracao'])

#Bar chart
fuel_type_bar = ax[1].bar(
    type_generator_constr['SigTipoGeracao'],
    type_generator_constr['MdaPotenciaOutorgadaKw']/1000,
    color=get_color(colors_type_generator_dict, type_generator_constr['SigTipoGeracao'

ax[1].ticklabel_format(axis='y', style='plain')
ax[1].set_title('In construction Electric Power')
ax[1].set_ylabel('Electric Power (MW)')
ax[1].set_xlabel('Power Plant Type')
#rotate y-axis label for better readability
plt.setp(ax[1].get_yticklabels(), rotation=45, ha='right')

#add value to top of each bar
add_label_top_bar(ax[1],0)

#print graph
plt.tight_layout()
plt.show()
```



Choropleth map of Electric Power generated by state:

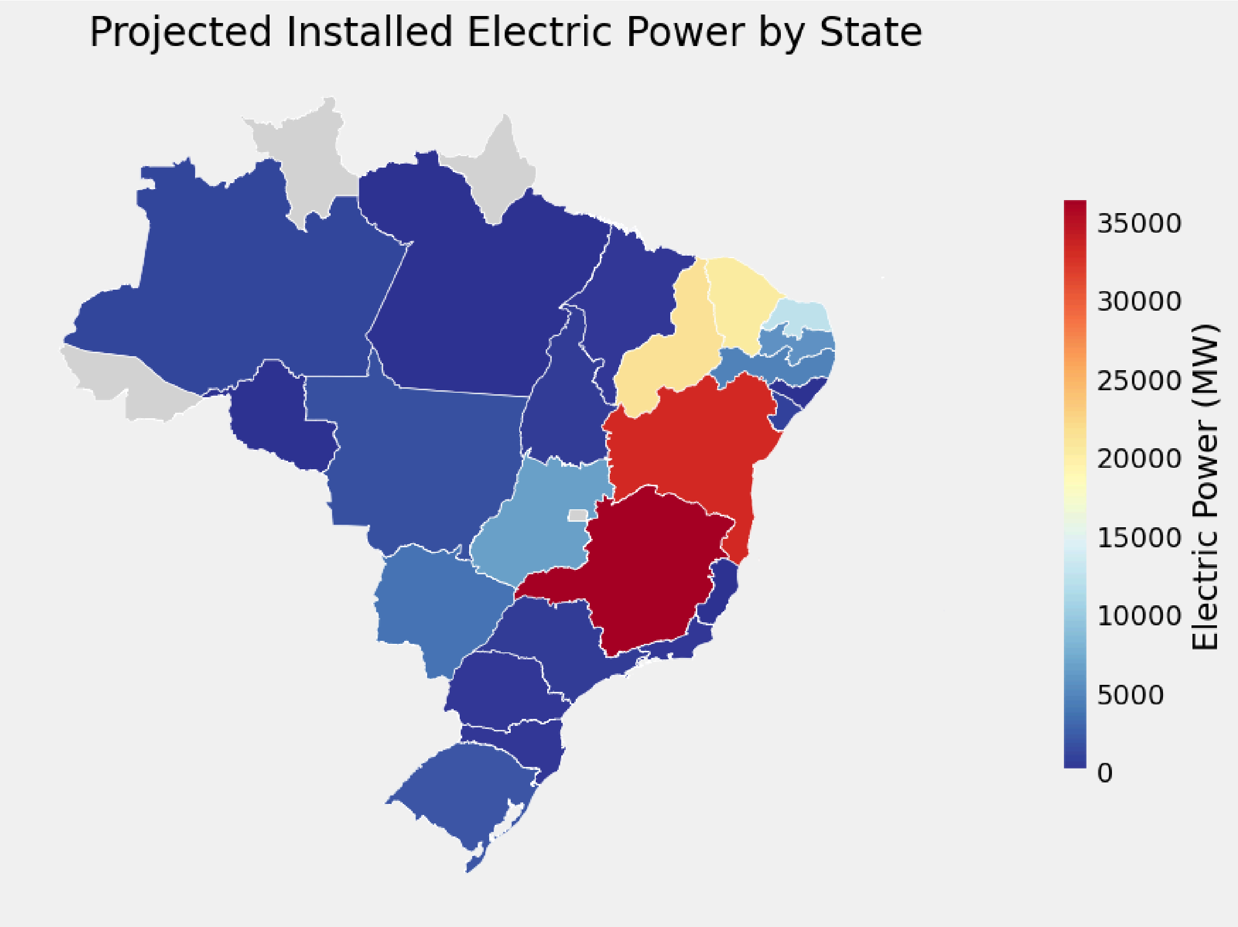
In [ ]:

```
#choropleth map of Operative Electric Power by State
#create figures and axes for matplotlib
fig, ax = plt.subplots(figsize=(10, 10))

#plot map
states_power_op.plot(
    column=states_power_op['MdaPotenciaOutorgadaKw']/1000,
    cmap="RdYlBu_r",
    vmin = 0,
    linewidth=0.5,
    edgecolor='1',
    legend=True,
    legend_kwds={
        "label": "Electric Power (MW)",
        "orientation": "vertical",
        "shrink": 0.5,},
    missing_kwds={
            "color": "lightgrey",
            "label": "Missing values",},
    ax=ax,
)
#set title
ax.set_title("Operative Installed Electric Power by State")
ax.axis("off")
plt.show()
```



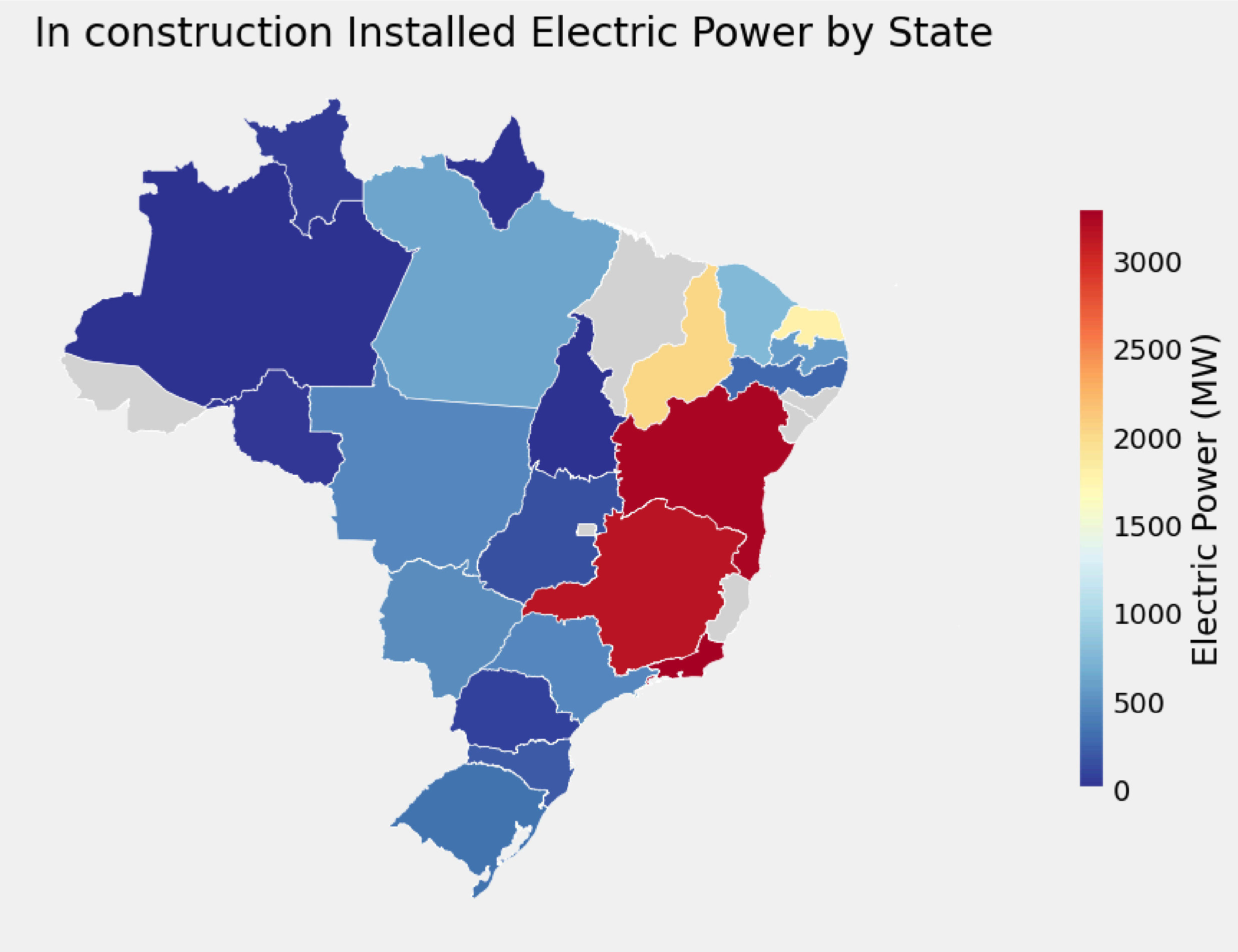
In [ ]:

```
#choropleth map of Projected Electric Power by State
#create figures and axes for matplotlib
fig, ax = plt.subplots(figsize=(10, 10))
#plot map
states_power_proj.plot(
    column=states_power_proj['MdaPotenciaOutorgadaKw']/1000,
    cmap="RdYlBu_r",
    vmin = 0,
    linewidth=0.5,
    edgecolor='1',
    legend=True,
    legend_kwds={
        "label": "Electric Power (MW)",
        "orientation": "vertical",
        "shrink": 0.5,},
    missing_kwds={
            "color": "lightgrey",
            "label": "Missing values",},
    ax=ax,
)
#set title
ax.set_title("Projected Installed Electric Power by State")
ax.axis("off")
plt.show()
```



In [ ]:

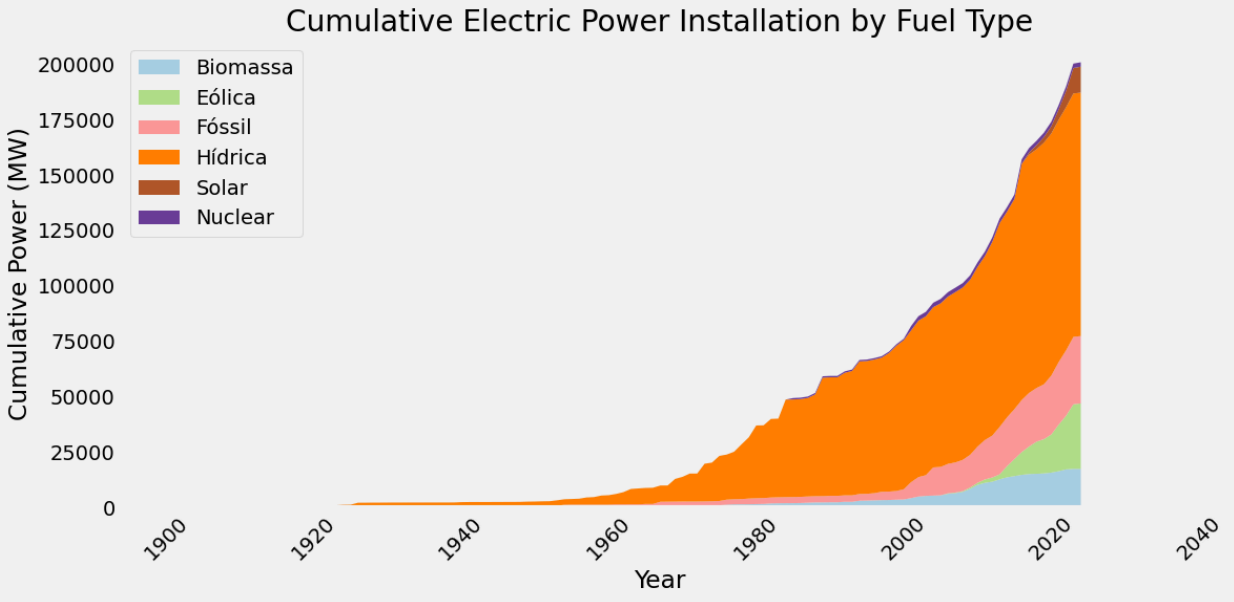
```
#choropleth map of In construction Electric Power by State
#create figures and axes for matplotlib
fig, ax = plt.subplots(figsize=(10, 10))
#plot map
states_power_constr.plot(
    column=states_power_constr['MdaPotenciaOutorgadaKw']/1000,
    cmap="RdYlBu_r",
    linewidth=0.5,
    edgecolor='1',
    vmin = 0,
    legend=True,
    legend_kwds={
        "label": "Electric Power (MW)",
        "orientation": "vertical",
        "shrink": 0.5,},
    missing_kwds={
            "color": "lightgrey",
            "label": "Missing values",},
    ax=ax,
)
#set title
ax.set_title("In construction Installed Electric Power by State")
ax.axis("off")
plt.show()
```



Historical line graph of evolution of installed electric power capacity in brazil:

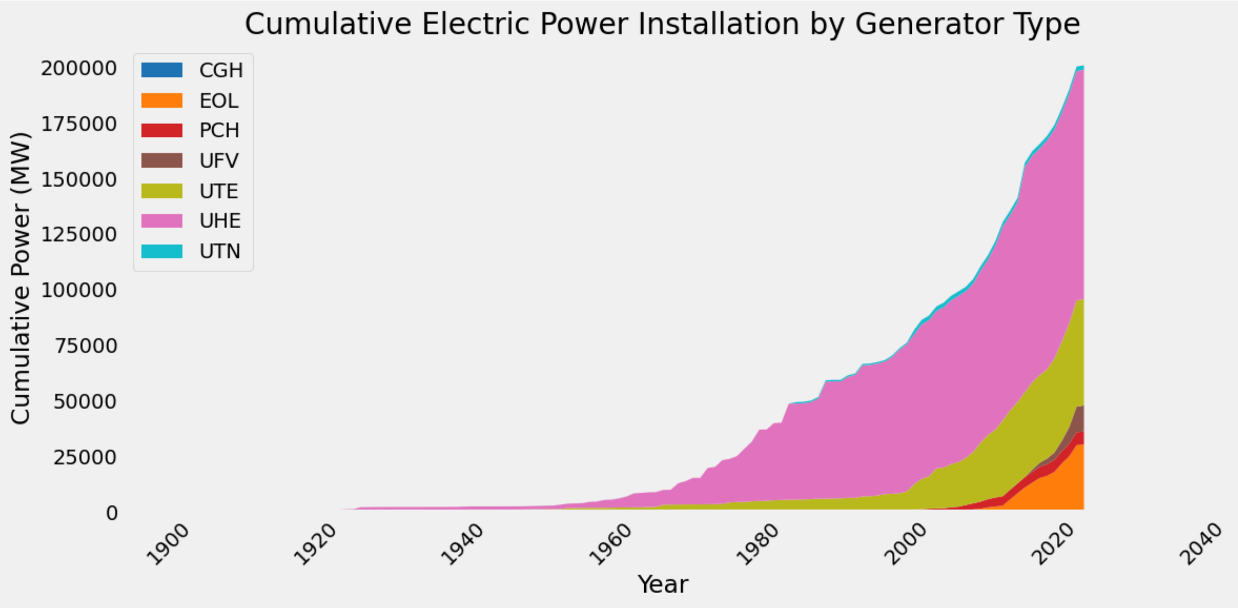
```
In [ ]: #create dataframe for graph of historical acumulative
        historical_electric_power = df_data[['DatEntradaOperacao', 'MdaPotenciaOutorgadaKw', '
        #get the year of the datetime column to do a groupby
        historical_electric_power = historical_electric_power.sort_values('DatEntradaOperacao'
        historical_electric_power['Year'] = historical_electric_power['DatEntradaOperacao'].dt

        #historical acumulated values by fuel type
        historical_electric_power_type_fuel = historical_electric_power[historical_electric_pc
        historical_electric_power_type_fuel['MdaPotenciaOutorgadaKw'] = historical_electric_pc

        # List of fuel types
        fuel_types = historical_electric_power_type_fuel['DscOrigemCombustivel'].unique()

        # Pivot the dataframe to have years as index and fuel types as columns
        df_pivot = historical_electric_power_type_fuel.pivot(index='Year', columns='DscOrigem(

        # Fill NaN values with 0 (for years where no power was added for a fuel type)
        df_pivot = df_pivot.fillna(0)

        # Ensure all fuel types are present, add missing ones with 0 if necessary
        for fuel in fuel_types:
            if fuel not in df_pivot.columns:
                df_pivot[fuel] = 0

        # Calculate cumulative sum for each fuel type
        df_cumsum = df_pivot.cumsum()
```

```
# Create the stacked area chart
fig, ax = plt.subplots(figsize=(12, 6))

# Define a color palette
#colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b']

# Plot stacked area chart
ax.stackplot(df_cumsum.index,
             [df_cumsum[fuel] for fuel in fuel_types],
             labels=fuel_types,
             colors=get_color(colors_type_fuel_dict, fuel_types))

# Customize the chart
ax.set_title('Cumulative Electric Power Installation by Fuel Type')
ax.set_xlabel('Year')
ax.set_ylabel('Cumulative Power (MW)')
ax.legend(loc='upper left')

# Set x-axis ticks to show every 20 years
ax.set_xticks(range(1900, 2041, 20))
ax.set_xticklabels(range(1900, 2041, 20), rotation=45)

# Add grid lines
ax.grid(False)

# Tight layout to prevent clipping of labels
plt.tight_layout()

# Show the plot
plt.show()
```



In [ ]:

```
#historical acumulated values by generator type
historical_electric_power_type_generator = historical_electric_power[historical_electr
historical_electric_power_type_generator['MdaPotenciaOutorgadaKw'] = historical_electr

# List of generator types
generator_types = historical_electric_power_type_generator['SigTipoGeracao'].unique()

# Pivot the dataframe to have years as index and generator types as columns
df_pivot_generator = historical_electric_power_type_generator.pivot(index='Year', colu
# Fill NaN values with 0 (for years where no power was added for a generator type)
df_pivot_generator = df_pivot_generator.fillna(0)

# Ensure all generator types are present, add missing ones with 0 if necessary
for generator in generator_types:
    if generator not in df_pivot_generator.columns:
        df_pivot_generator[generator] = 0

# Calculate cumulative sum for each generator type
df_cumsum_generator = df_pivot_generator.cumsum()

# Create the stacked area chart
fig, ax = plt.subplots(figsize=(12, 6))

# Define a color palette
#colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b']

# Plot stacked area chart
ax.stackplot(df_cumsum_generator.index,
             [df_cumsum_generator[generator] for generator in generator_types],
             labels=generator_types,
             colors=get_color(colors_type_generator_dict, generator_types))

# Customize the chart
ax.set_title('Cumulative Electric Power Installation by Generator Type')
ax.set_xlabel('Year')
ax.set_ylabel('Cumulative Power (MW)')
ax.legend(loc='upper left')

# Set x-axis ticks to show every 20 years
ax.set_xticks(range(1900, 2041, 20))
ax.set_xticklabels(range(1900, 2041, 20), rotation=45)

# Add grid lines
ax.grid(False)

# Tight layout to prevent clipping of labels
plt.tight_layout()

# Show the plot
plt.show()
```


Cumulative Electric Power Installation by Generator Type
CGH
EOL
PCH
UFV
UTE
UHE
UTN
Cumulative Power (MW)
200000
175000
150000
125000
100000
75000
50000
25000
0
1900
1920
1940
1960
1980
2000
2020
2040
Year

In [ ]: