Contagem de pixels

```
!pip install rasterio
!pip install shapely
!pip install pyproj
```

```
Collecting rasterio
  Downloading rasterio-1.4.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (9.1 kB)
Collecting affine (from rasterio)
  Downloading affine-2.4.0-py3-none-any.whl.metadata (4.0 kB)
Requirement already satisfied: attrs in /usr/local/lib/python3.11/dist-packages (from rasterio) (25.3.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from rasterio) (2025.4.26)
Requirement already satisfied: click>=4.0 in /usr/local/lib/python3.11/dist-packages (from rasterio) (8.2.1)
Collecting cligj>=0.5 (from rasterio)
  Downloading cligj-0.7.2-py3-none-any.whl.metadata (5.0 kB)
Requirement already satisfied: numpy>=1.24 in /usr/local/lib/python3.11/dist-packages (from rasterio) (2.0.2)
Collecting click-plugins (from rasterio)
  Downloading click_plugins-1.1.1-py2.py3-none-any.whl.metadata (6.4 kB)
Requirement already satisfied: pyparsing in /usr/local/lib/python3.11/dist-packages (from rasterio) (3.2.3)
Downloading rasterio-1.4.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (22.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 22.2/22.2 MB 76.7 MB/s eta 0:00:00
Downloading cligj-0.7.2-py3-none-any.whl (7.1 kB)
Downloading affine-2.4.0-py3-none-any.whl (15 kB)
Downloading click_plugins-1.1.1-py2.py3-none-any.whl (7.5 kB)
Installing collected packages: cligj, click-plugins, affine, rasterio
Successfully installed affine-2.4.0 click-plugins-1.1.1 cligj-0.7.2 rasterio-1.4.3
Requirement already satisfied: shapely in /usr/local/lib/python3.11/dist-packages (2.1.1)
Requirement already satisfied: numpy>=1.21 in /usr/local/lib/python3.11/dist-packages (from shapely) (2.0.2)
Requirement already satisfied: pyproj in /usr/local/lib/python3.11/dist-packages (3.7.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from pyproj) (2025.4.26)
```

```
import rasterio
import os
import pandas as pd
from shapely.geometry import Polygon, mapping
from rasterio.features import geometry_mask
import numpy as np
from pyproj import Transformer
from google.colab import drive
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import geopandas as gpd
from scipy.stats import linregress
from rasterio.plot import show
from rasterio.mask import mask
from glob import glob
from PIL import Image
```

```
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

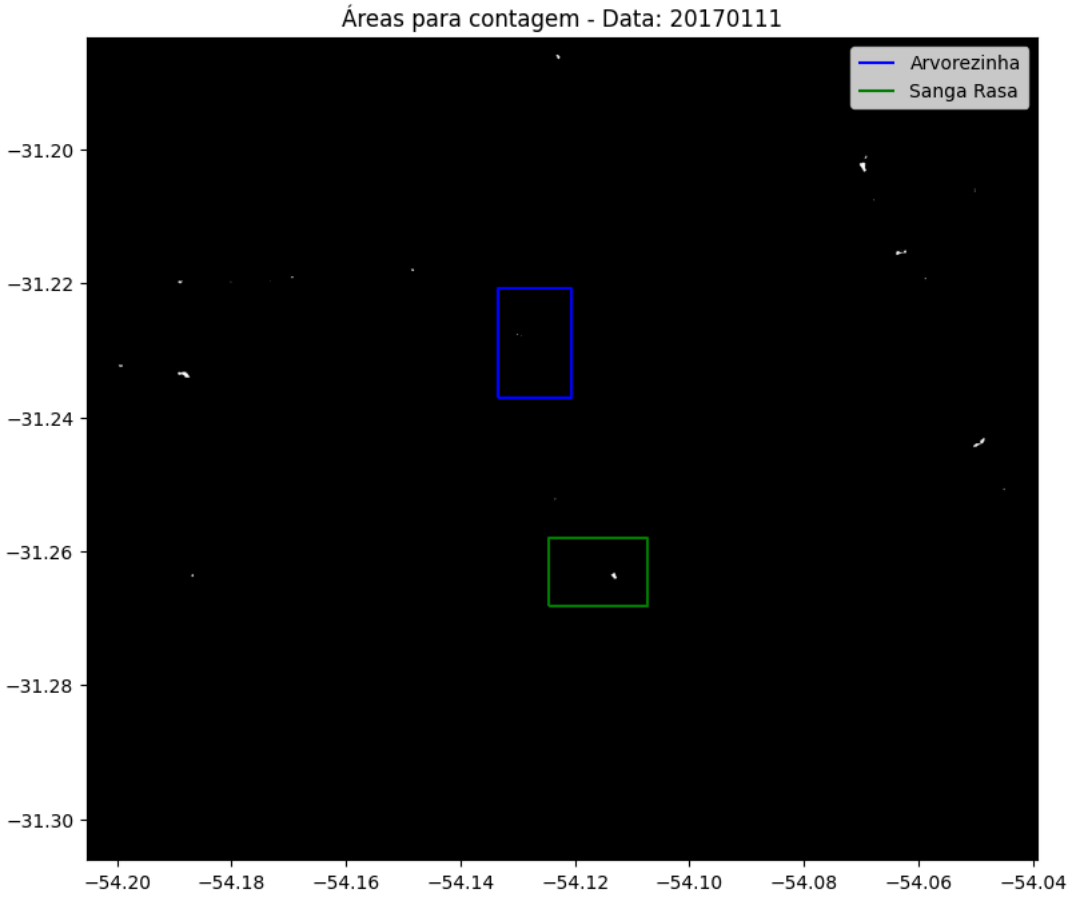
```
csv_paths = {
    2017: '/content/drive/My Drive/SENTINEL_DATA/sentinel2/2017/output/contagem_pixels.csv',
    2018: '/content/drive/My Drive/SENTINEL_DATA/sentinel2/2018/output/contagem_pixels.csv',
    2019: '/content/drive/My Drive/GEE_Folder_Raw_40_Perc_2019/output/contagem_pixels.csv',
    2020: '/content/drive/My Drive/GEE_Folder_Raw_40_Perc_2020/output/contagem_pixels.csv',
    2021: '/content/drive/My Drive/GEE_Folder_Raw_40_Perc_2021/output/contagem_pixels.csv',
    2022: '/content/drive/My Drive/GEE_Folder_Raw_40_Perc_2022/output/contagem_pixels.csv',
    2023: '/content/drive/My Drive/GEE_Folder_Raw_40_Perc_2023/output/contagem_pixels.csv',
    2024: '/content/drive/My Drive/GEE_Folder_Raw_40_Perc_2024/output/contagem_pixels.csv',
    2025: '/content/drive/My Drive/GEE_Folder_Raw_40_Perc_2025/output/contagem_pixels.csv'
}
```

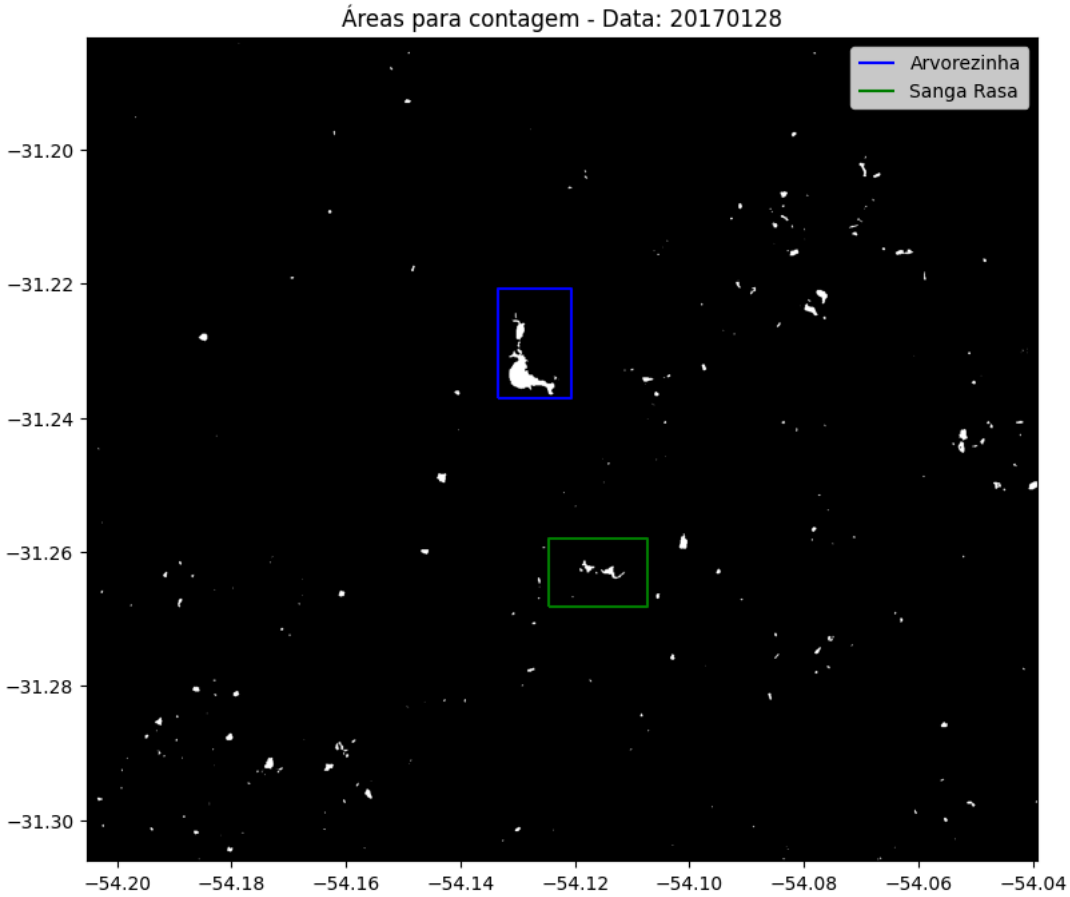
```
output_folders = {
    2017: '/content/drive/My Drive/SENTINEL_DATA/sentinel2/2017/output/',
    2018: '/content/drive/My Drive/SENTINEL_DATA/sentinel2/2018/output/',
    2019: '/content/drive/My Drive/GEE_Folder_Raw_40_Perc_2019/output/',
    2020: '/content/drive/My Drive/GEE_Folder_Raw_40_Perc_2020/output/',
    2021: '/content/drive/My Drive/GEE_Folder_Raw_40_Perc_2021/output/',
    2022: '/content/drive/My Drive/GEE_Folder_Raw_40_Perc_2022/output/',
    2023: '/content/drive/My Drive/GEE_Folder_Raw_40_Perc_2023/output/',
    2024: '/content/drive/My Drive/GEE_Folder_Raw_40_Perc_2024/output/',
    2025: '/content/drive/My Drive/GEE_Folder_Raw_40_Perc_2025/output/',
}
```

```
def count_pixels(path_tif):
    with rasterio.open(path_tif) as img:
```

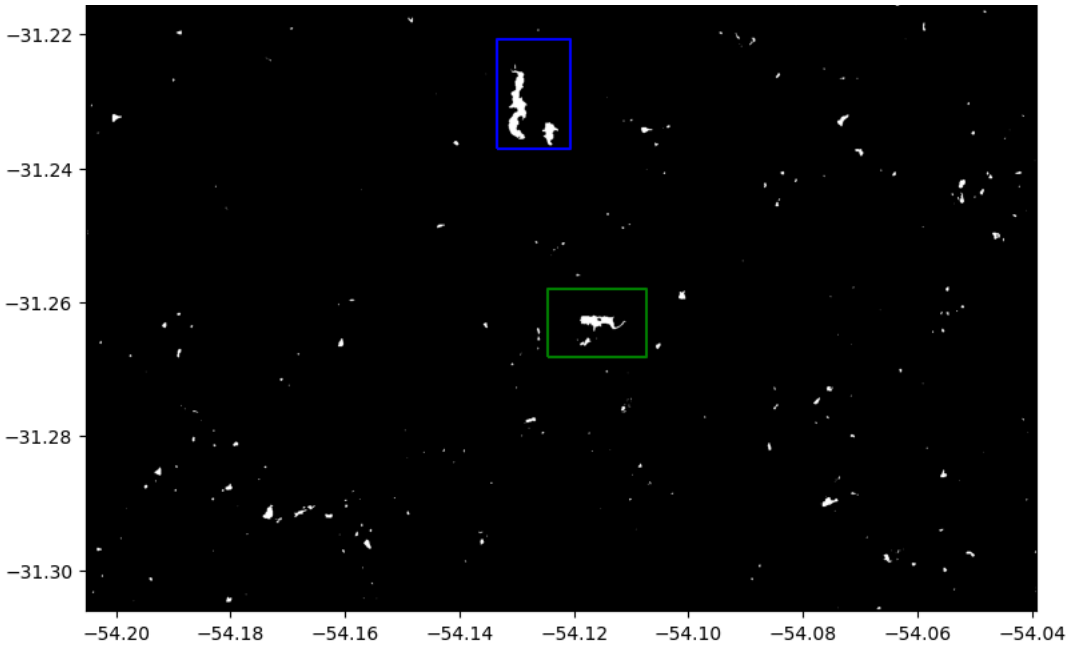
```
            #print(img.crs)
            data = img.read(1)
            transform = img.transform
            crs_img = img.crs  # sistema de coordenadas do .tif

            # reprojeta de EPSG:4326 (lat/lon) para o crs do .tif
            transformer = Transformer.from_crs("EPSG:4326", crs_img, always_xy=True)

            def reproject_polygon(coords):
                return Polygon([transformer.transform(x, y) for x, y in coords])

            # cordenadas provenientes do google earth engine
            arvorezinha_coords = [
                (-54.13341114332514, -31.23703975708336),
                (-54.12070820143061, -31.23703975708336),
                (-54.12070820143061, -31.22081967345758),
                (-54.13341114332514, -31.22081967345758)
            ]

            sanga_rasa_coords = [
                (-54.124553222478525, -31.268190891664986),
                (-54.10743000012745, -31.268190891664986),
                (-54.10743000012745, -31.257992809414805),
                (-54.124553222478525, -31.257992809414805)
            ]

            # reprojeta as coordenadas dos poligonos
            arvorezinha_poly = reproject_polygon(arvorezinha_coords)
            sanga_rasa_poly = reproject_polygon(sanga_rasa_coords)

            # cria mascaras
            mask1 = geometry_mask([mapping(arvorezinha_poly)], transform=transform, invert=True, out_shape=data.shape)
            mask2 = geometry_mask([mapping(sanga_rasa_poly)], transform=transform, invert=True, out_shape=data.shape)

            # 1 na watnet conta agua e 0 na unet (data == 1)
            # 0 na watnet conta terra e 1 na unet
            arvorezinha_count = np.sum((data == 1) & mask1)
            sanga_rasa_count = np.sum((data == 1) & mask2)

            return arvorezinha_count, sanga_rasa_count


    def count_pixels_aux(img_path):
      result = []

      # itera arquivos .tif
      for arq in sorted(os.listdir(img_path)):
          if arq.endswith(".tif"):
              path = os.path.join(img_path, arq)
              data_str = os.path.splitext(arq)[0]  # remove ".tif" para pegar a data
              try:
                  arvorezinha, sanga_rasa = count_pixels(path)
                  result.append({"data": data_str, "Arvorezinha": arvorezinha, "Sanga_Rasa": sanga_rasa})
              except Exception as e:
                  print(f"Erro ao processar {arq}: {e}")

      # cria o DataFrame e salva em .csv
      df = pd.DataFrame(result)

      output_path = os.path.join(img_path, "contagem_pixels.csv")
      df.to_csv(output_path, index=False)


    count_pixels_aux(output_folders[2017])
    count_pixels_aux(output_folders[2018])


    count_pixels_aux(output_folders[2019])
    count_pixels_aux(output_folders[2020])
    count_pixels_aux(output_folders[2021])
    count_pixels_aux(output_folders[2022])
    count_pixels_aux(output_folders[2023])
    count_pixels_aux(output_folders[2024])
    count_pixels_aux(output_folders[2025])


    def show_areas(img_path):
        # coordenadas originais (EPSG:4326)
        arvorezinha_coords = [
                (-54.13341114332514, -31.23703975708336),
                (-54.12070820143061, -31.23703975708336),
                (-54.12070820143061, -31.22081967345758),
                (-54.13341114332514, -31.22081967345758)
```

```
        ]

        sanga_rasa_coords = [
                (-54.124553222478525, -31.268190891664986),
                (-54.10743000012745, -31.268190891664986),
                (-54.10743000012745, -31.257992809414805),
                (-54.124553222478525, -31.257992809414805)
        ]

        for arq in sorted(os.listdir(img_path)):
            if arq.endswith(".tif"):
                path = os.path.join(img_path, arq)
                data_str = os.path.splitext(arq)[0]
                try:
                    with rasterio.open(path) as img:
                        data = img.read(1)
                        crs_img = img.crs
                        transform = img.transform

                        transformer = Transformer.from_crs("EPSG:4326", crs_img, always_xy=True)
                        arvorezinha_poly = Polygon([transformer.transform(x, y) for x, y in arvorezinha_coords])
                        sanga_rasa_poly = Polygon([transformer.transform(x, y) for x, y in sanga_rasa_coords])

                        arvorezinha_gdf = gpd.GeoDataFrame(geometry=[arvorezinha_poly], crs=crs_img)
                        sanga_rasa_gdf = gpd.GeoDataFrame(geometry=[sanga_rasa_poly], crs=crs_img)

                        fig, ax = plt.subplots(figsize=(10, 8))
                        show(data, transform=transform, ax=ax, cmap='gray')
                        arvorezinha_gdf.boundary.plot(ax=ax, color='blue', label='Arvorezinha')
                        sanga_rasa_gdf.boundary.plot(ax=ax, color='green', label='Sanga Rasa')
                        plt.title(f"Áreas para contagem - Data: {data_str}")
                        plt.legend()
                        plt.show()
                except Exception as e:
                    print(f"Erro ao processar {arq}: {e}")


show_areas(output_folders[2017])
#show_areas(output_folders[2023])
```

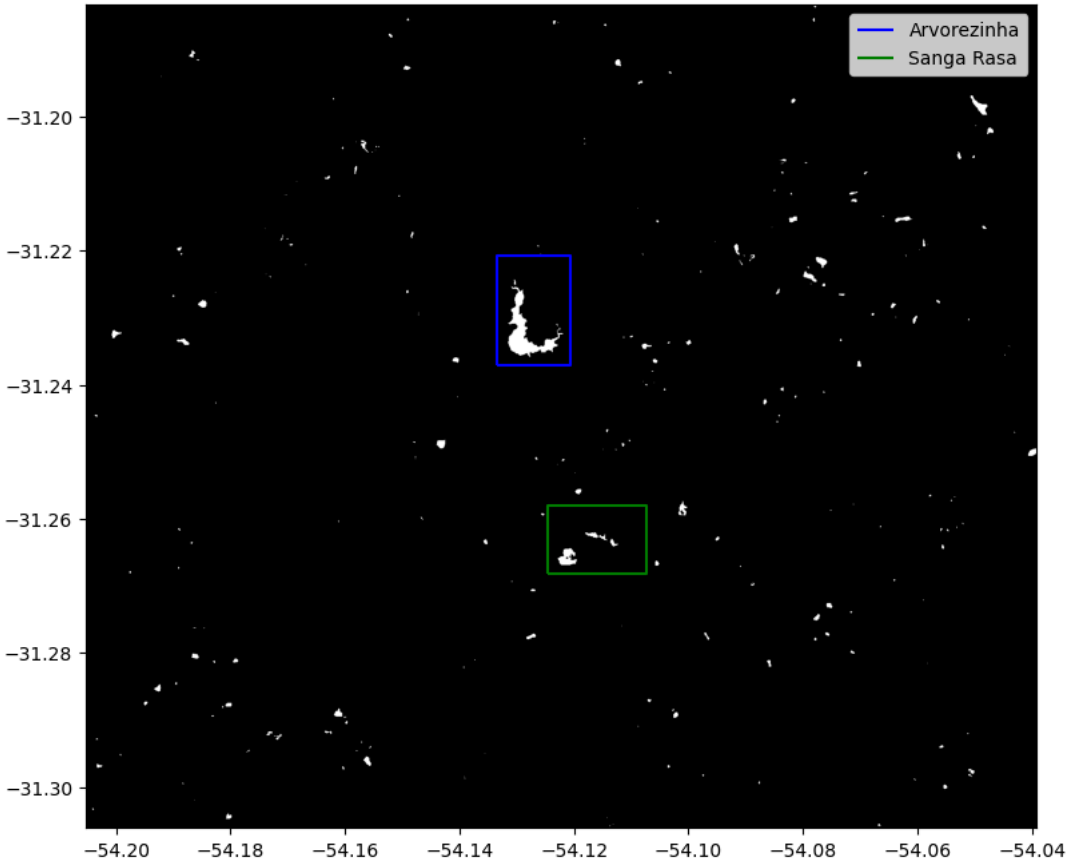

Áreas para contagem - Data: 20170111
Arvorezinha
Sanga Rasa
−31.20
−31.22
−31.24
−31.26
−31.28
−31.30
−54.20
−54.18
−54.16
−54.14
−54.12
−54.10
−54.08
−54.06
−54.04


Áreas para contagem - Data: 20170128
Arvorezinha
Sanga Rasa
−31.20
−31.22
−31.24
−31.26
−31.28
−31.30
−54.20
−54.18
−54.16
−54.14
−54.12
−54.10
−54.08
−54.06
−54.04

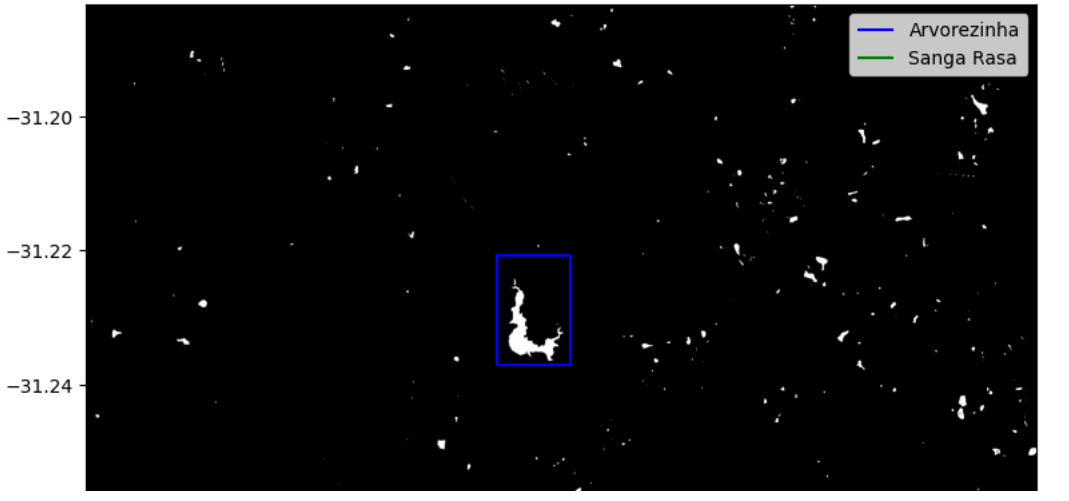

Áreas para contagem - Data: 20170207
Arvorezinha
Sanga Rasa
−31.20

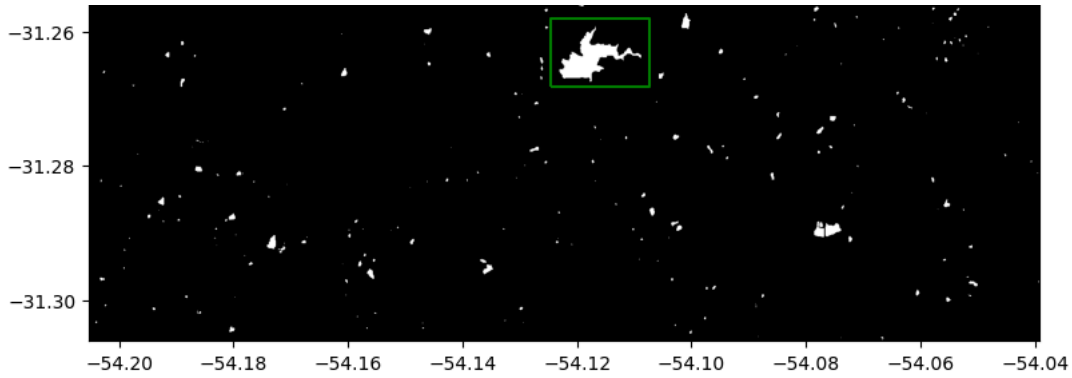


Áreas para contagem - Data: 20170302

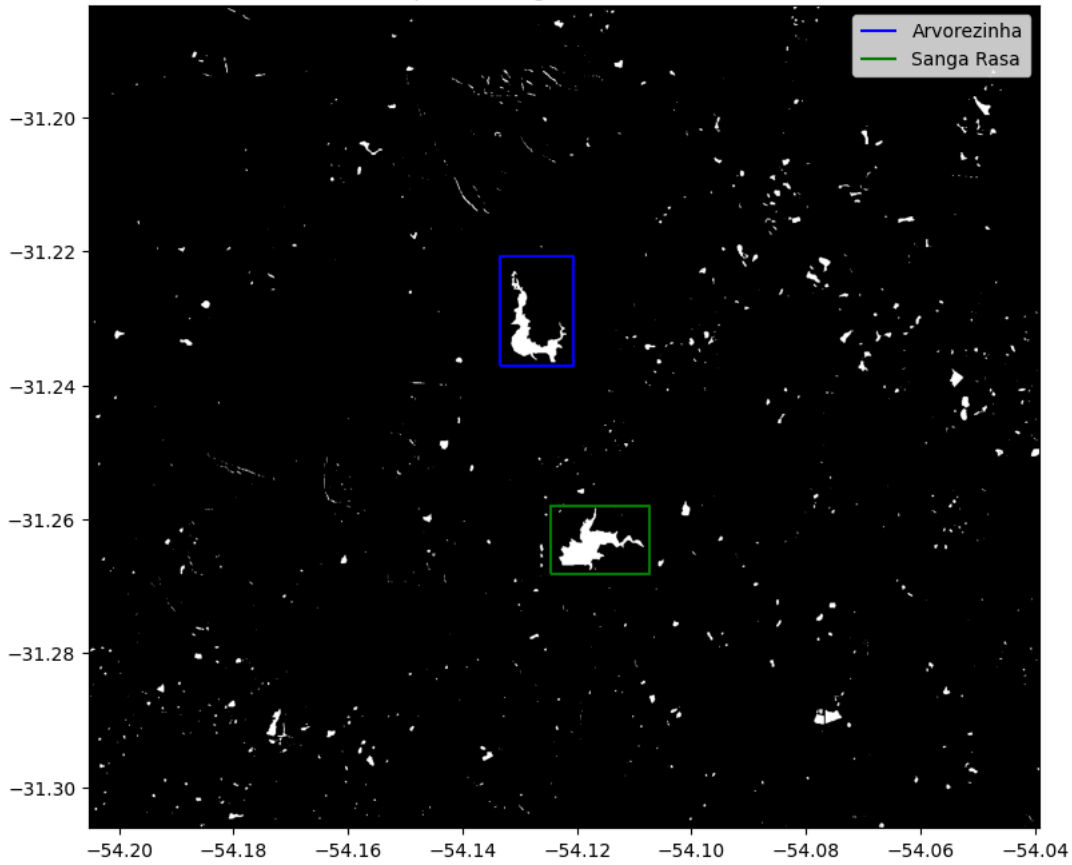


Áreas para contagem - Data: 20170329

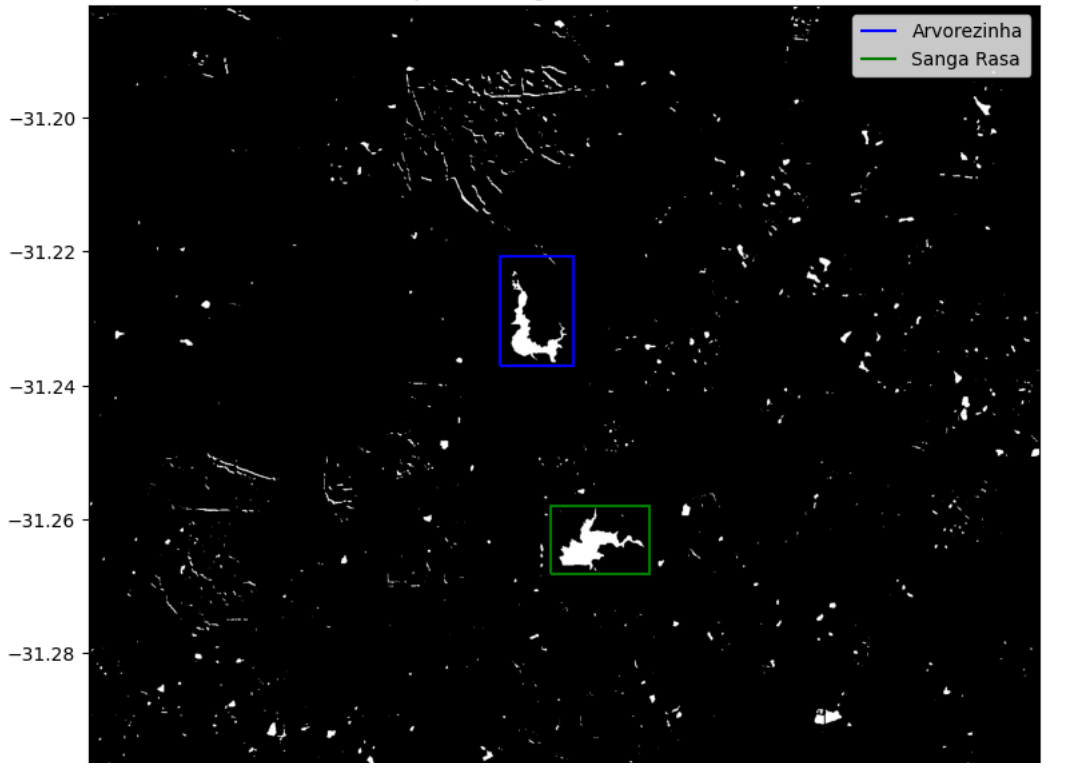


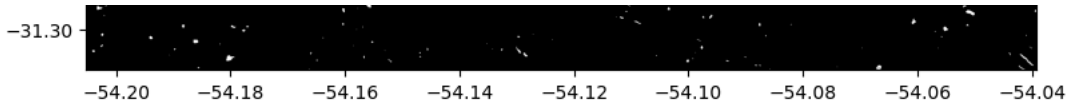


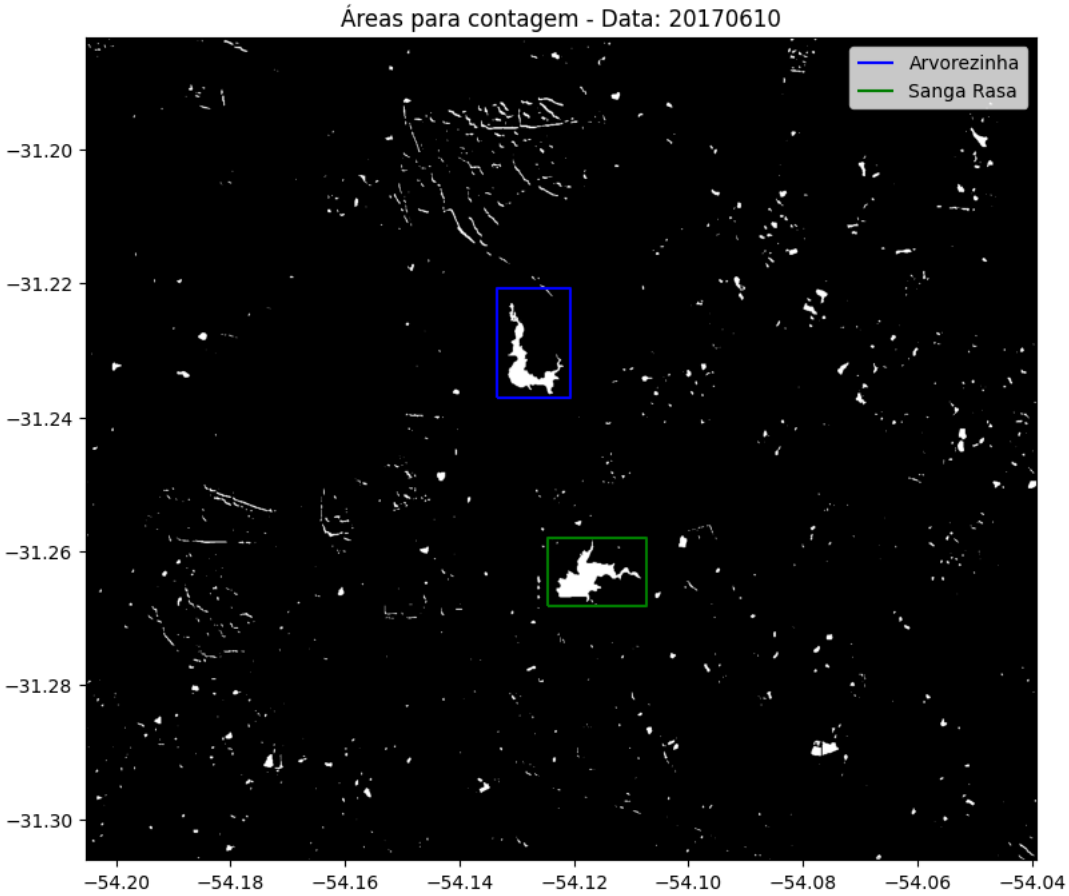
Áreas para contagem - Data: 20170411

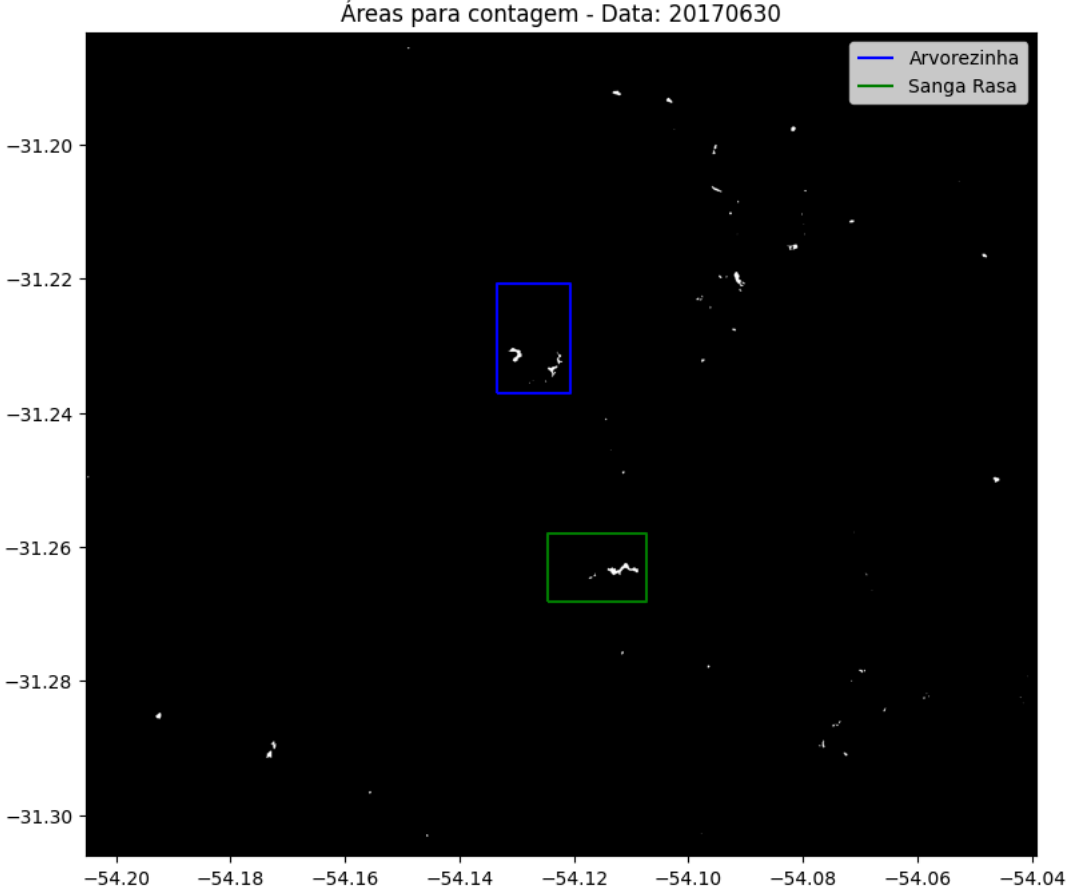


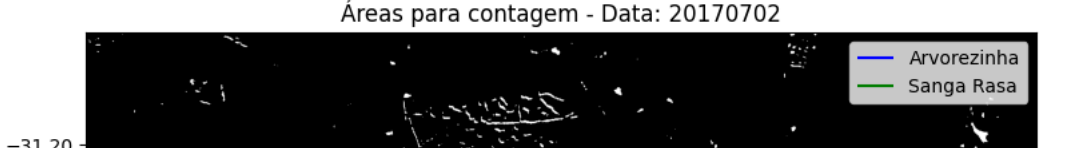
Áreas para contagem - Data: 20170521

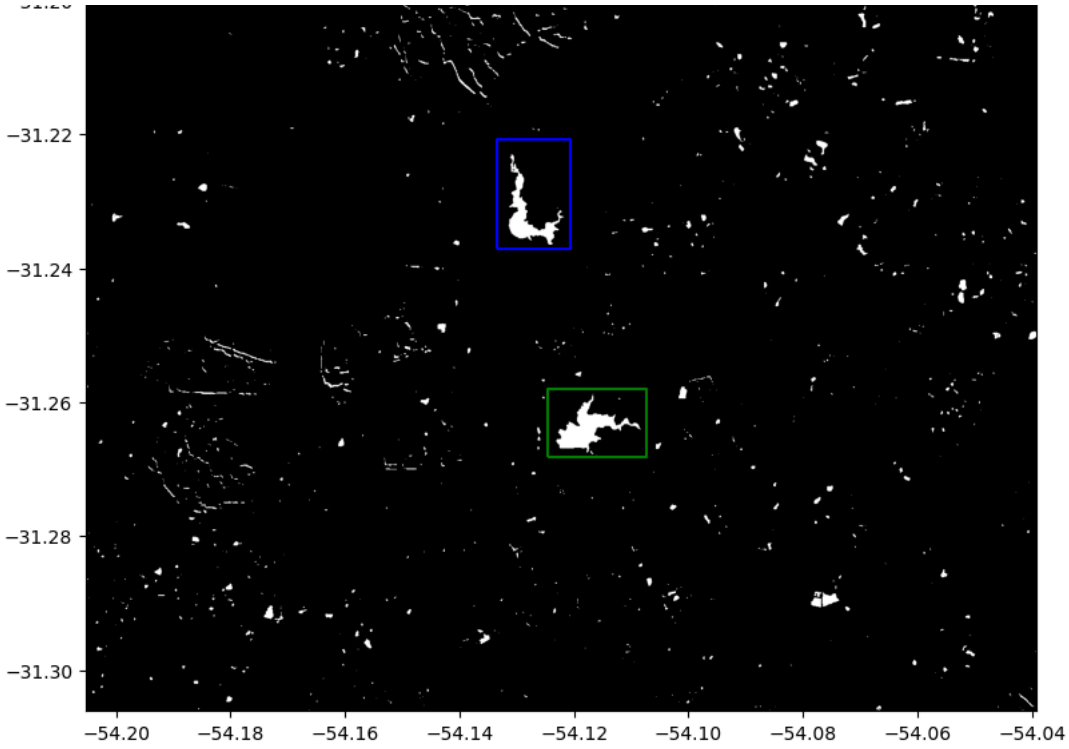



−31.30
−54.20
−54.18
−54.16
−54.14
−54.12
−54.10
−54.08
−54.06
−54.04

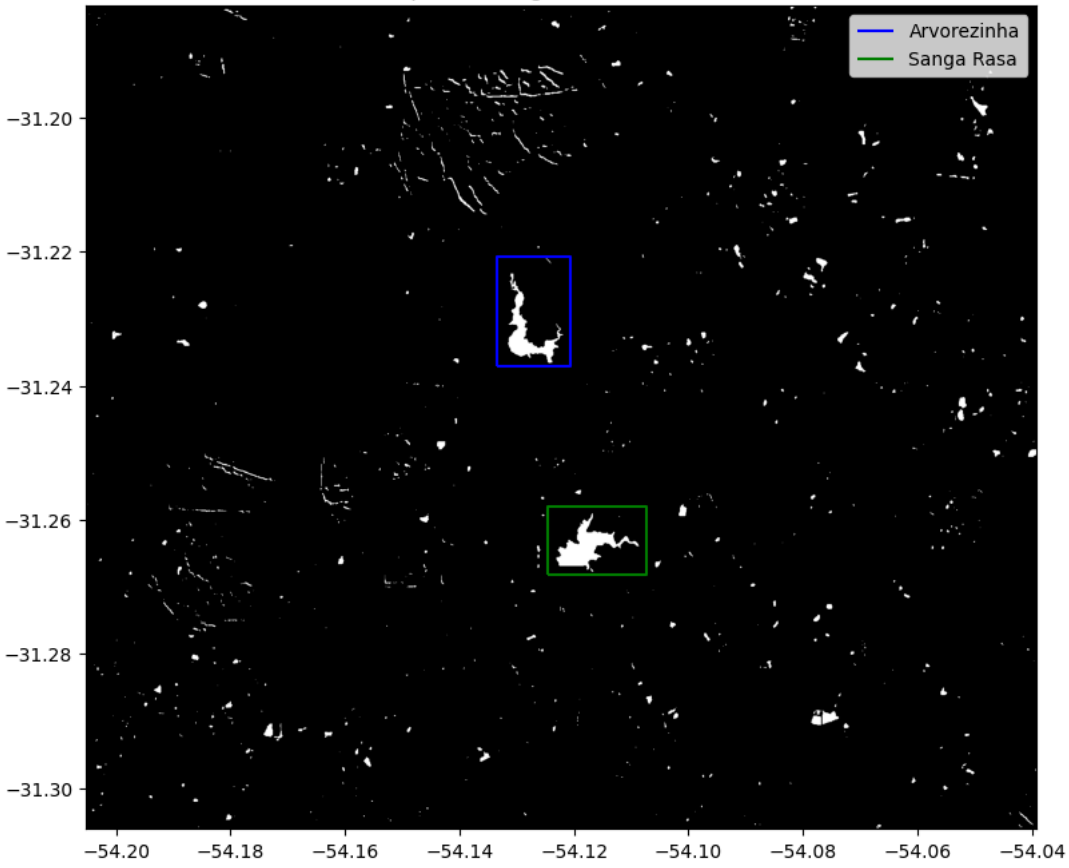

Áreas para contagem - Data: 20170610
Arvorezinha
Sanga Rasa
−31.20
−31.22
−31.24
−31.26
−31.28
−31.30
−54.20
−54.18
−54.16
−54.14
−54.12
−54.10
−54.08
−54.06
−54.04


Áreas para contagem - Data: 20170630
Arvorezinha
Sanga Rasa
−31.20
−31.22
−31.24
−31.26
−31.28
−31.30
−54.20
−54.18
−54.16
−54.14
−54.12
−54.10
−54.08
−54.06
−54.04

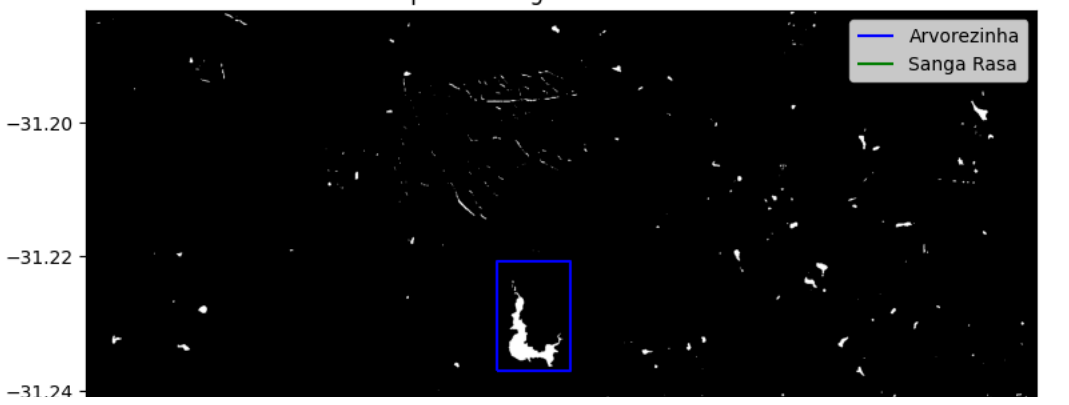

Áreas para contagem - Data: 20170702
Arvorezinha
Sanga Rasa

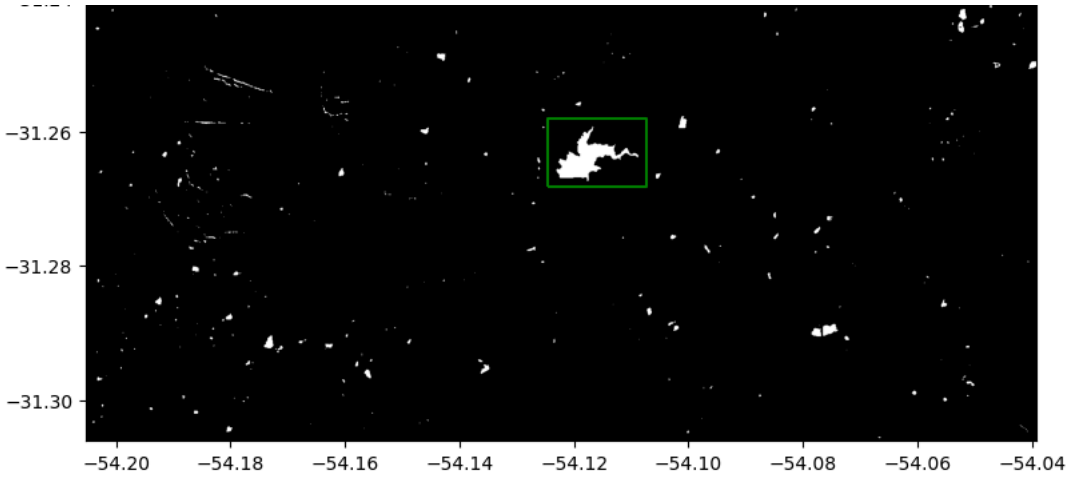


Áreas para contagem - Data: 20170720

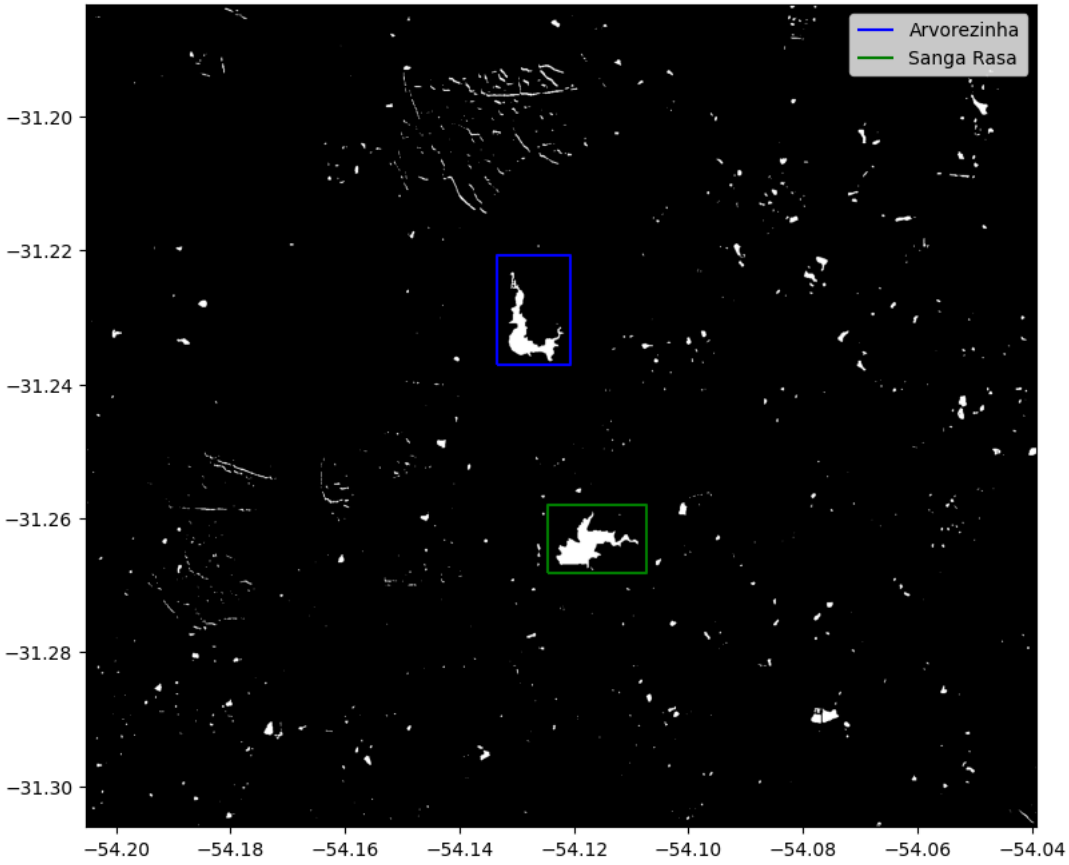


Áreas para contagem - Data: 20170722

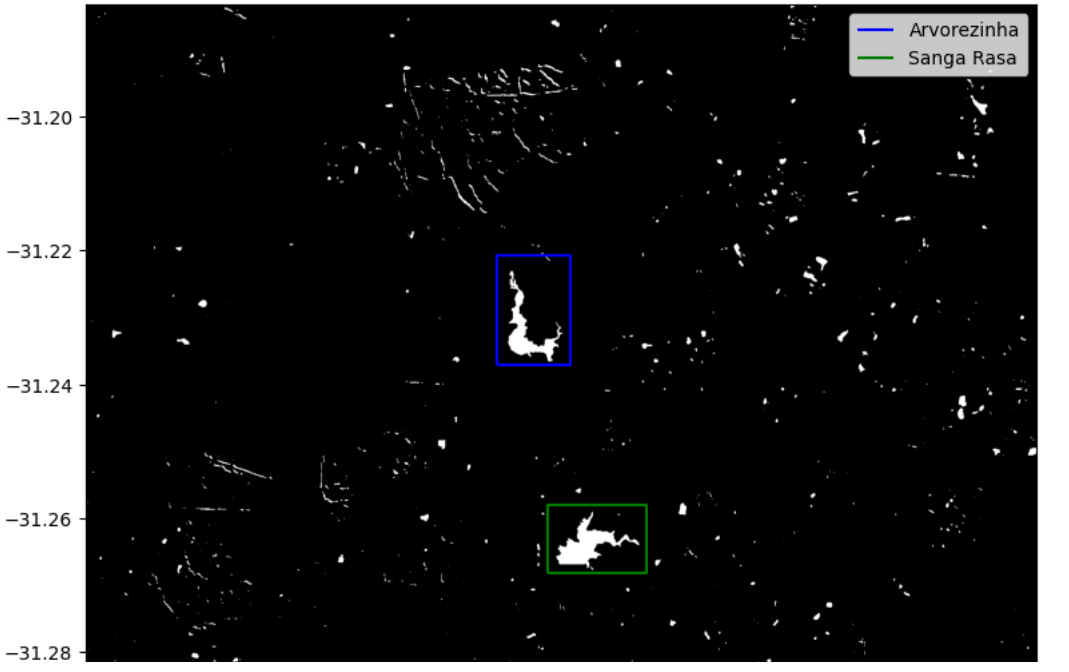


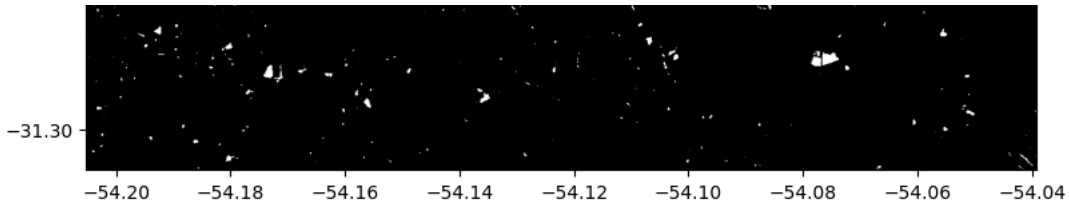


Áreas para contagem - Data: 20170727

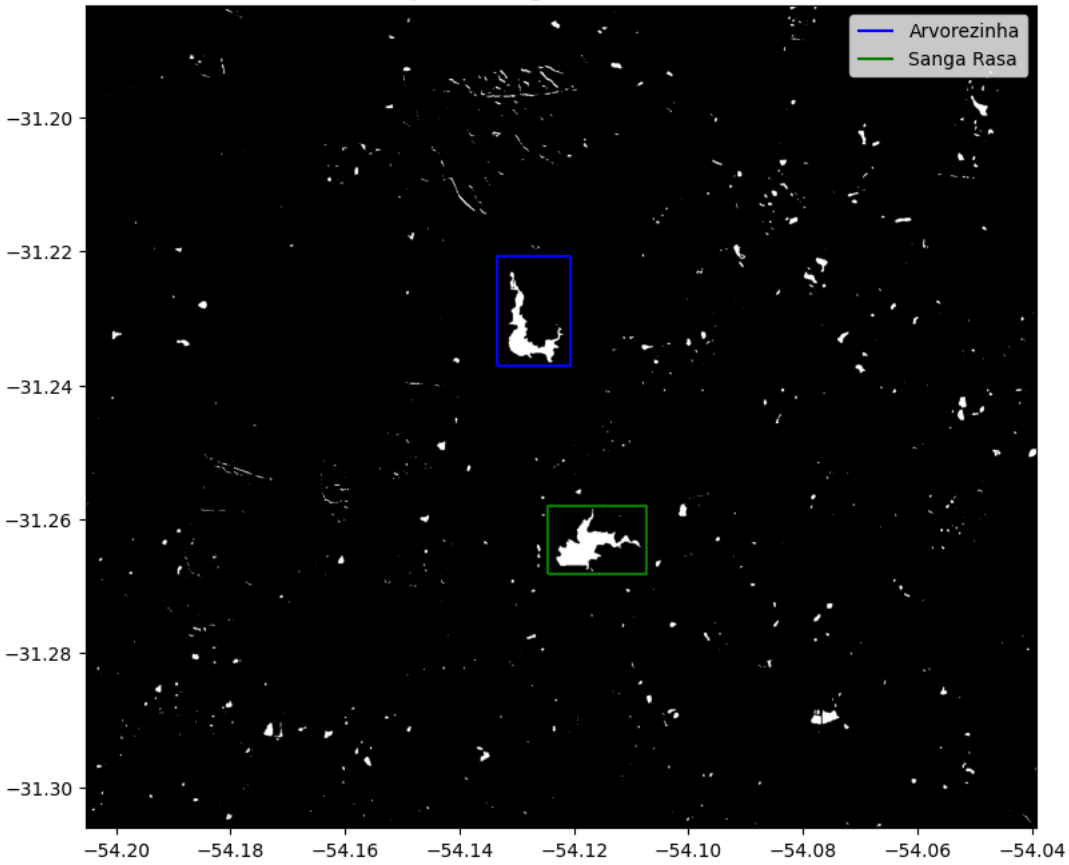


Áreas para contagem - Data: 20170804

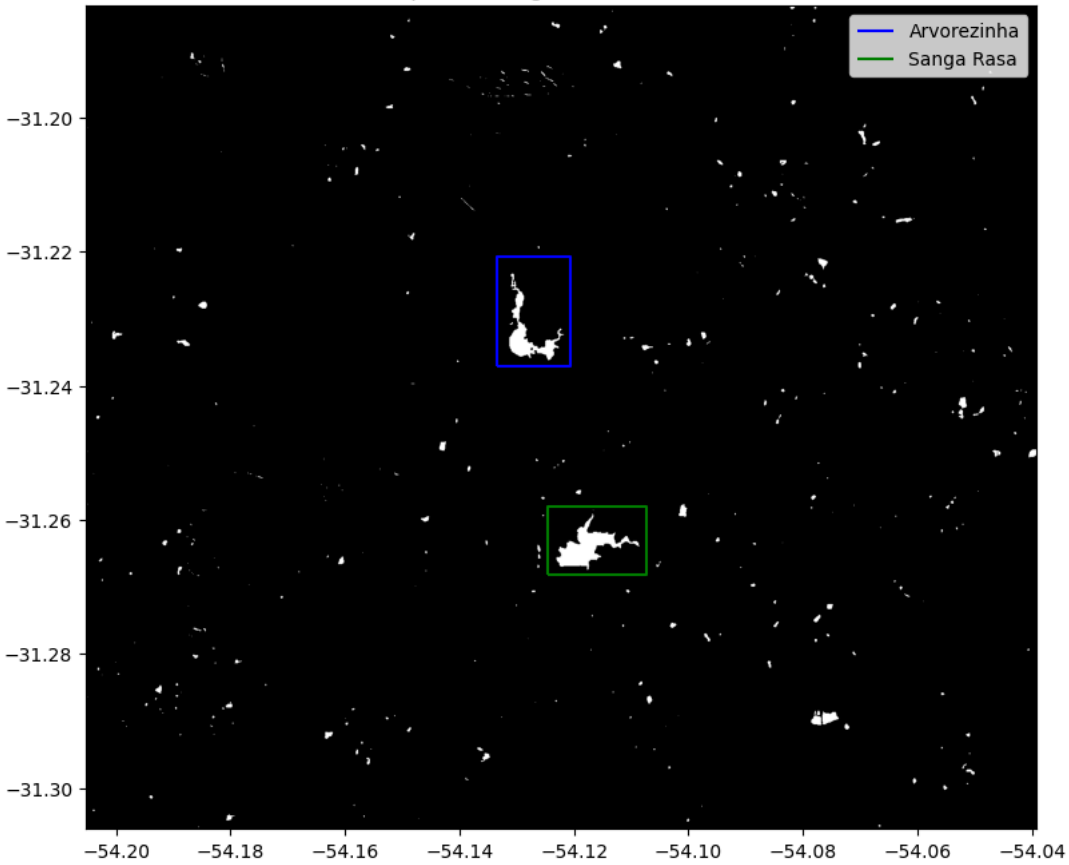




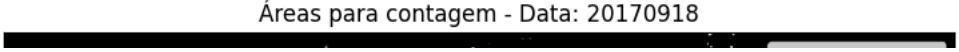
Áreas para contagem - Data: 20170816

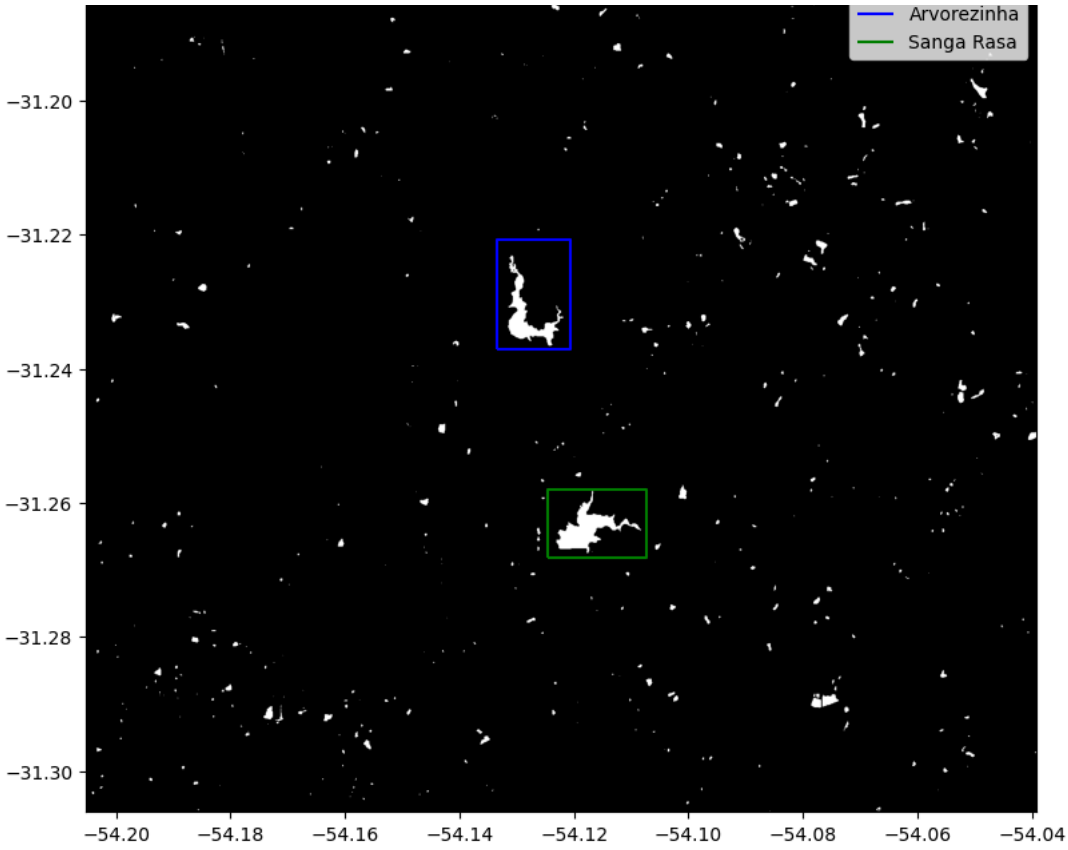


Áreas para contagem - Data: 20170831

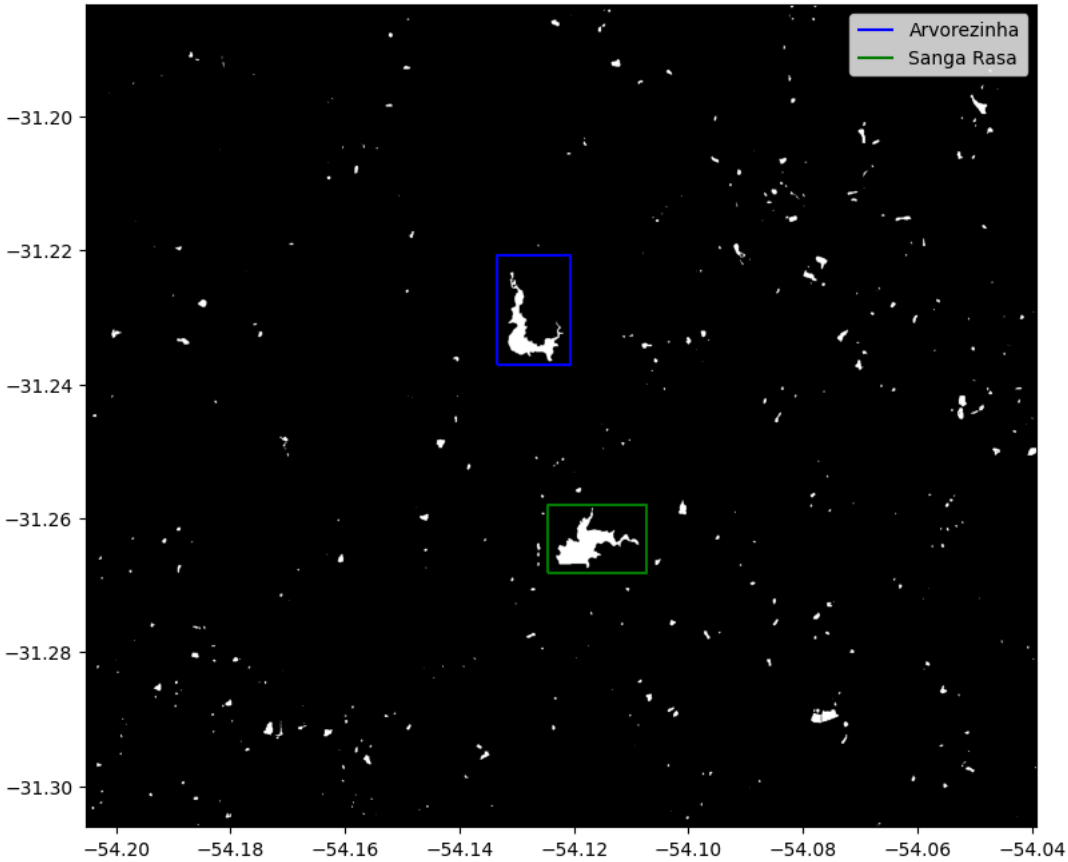


Áreas para contagem - Data: 20170918

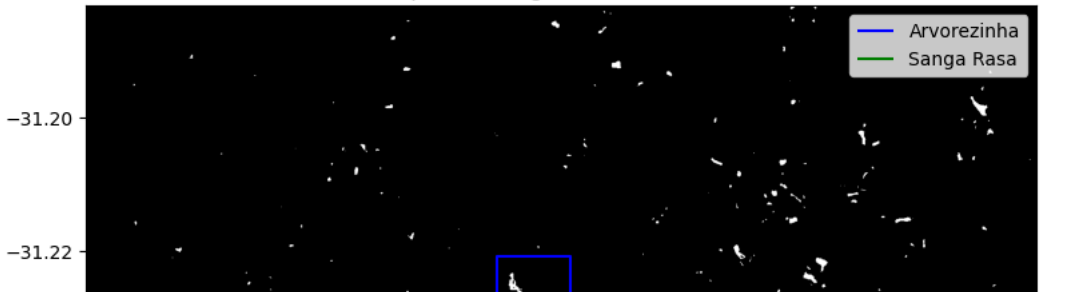


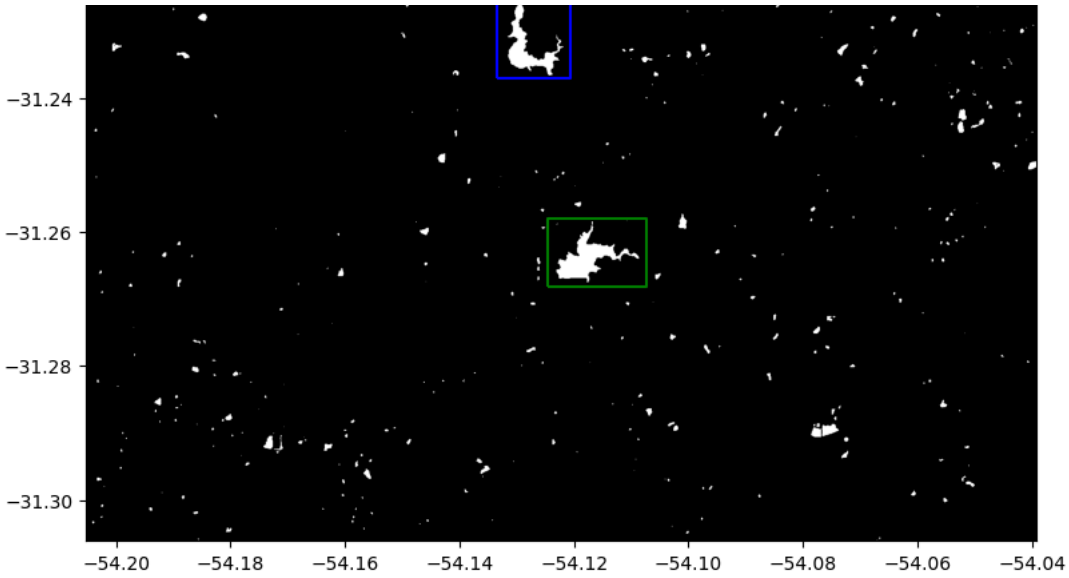


Áreas para contagem - Data: 20171003

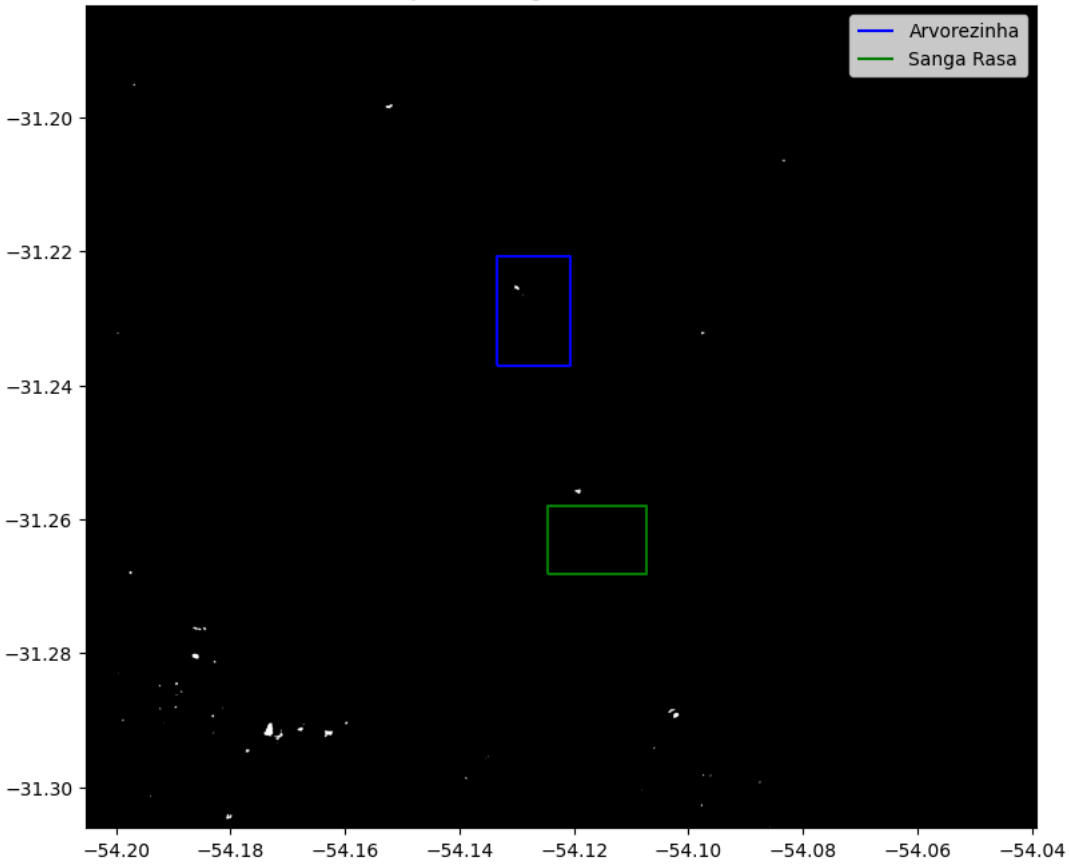


Áreas para contagem - Data: 20171023

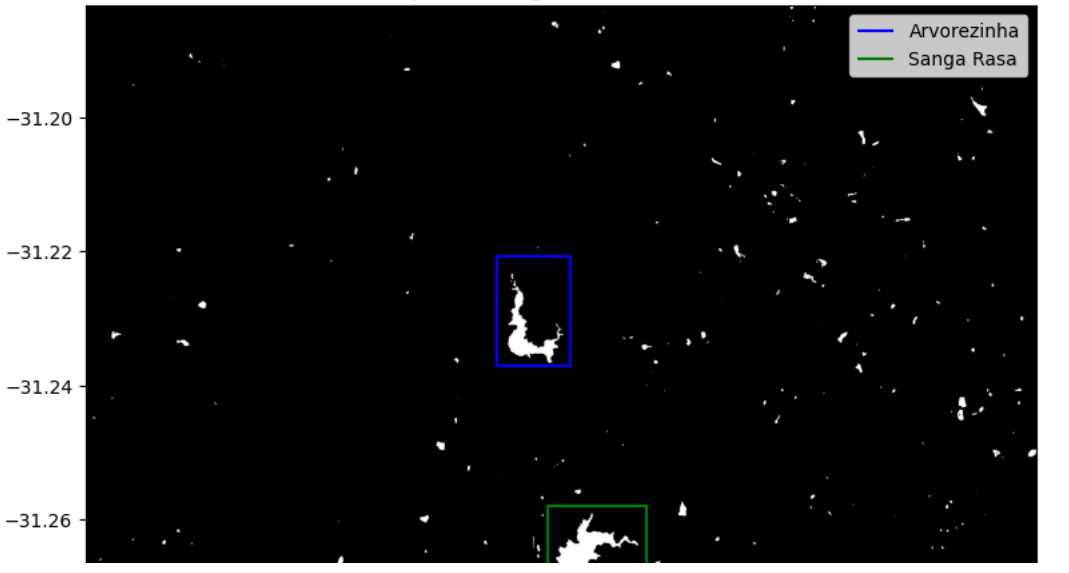


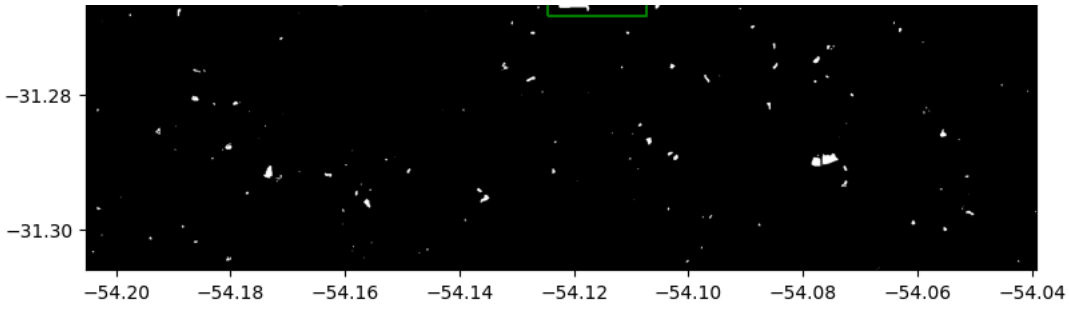


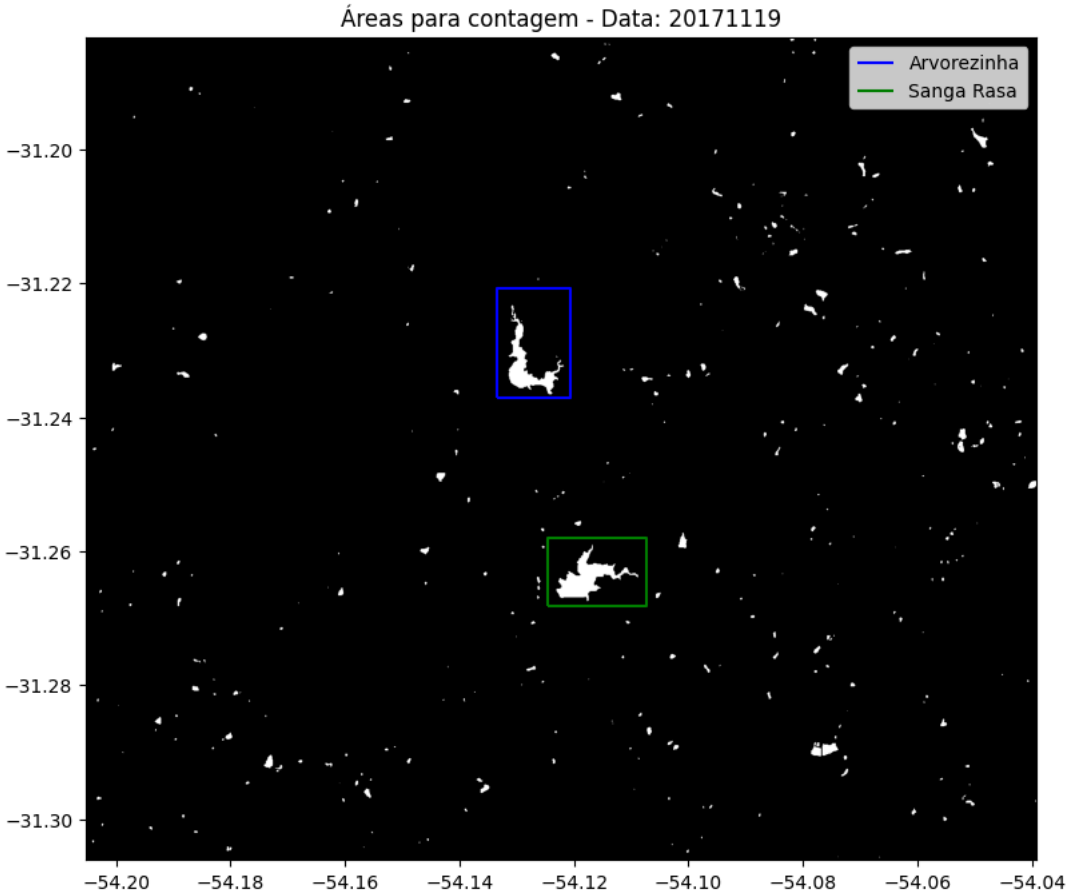
Áreas para contagem - Data: 20171104

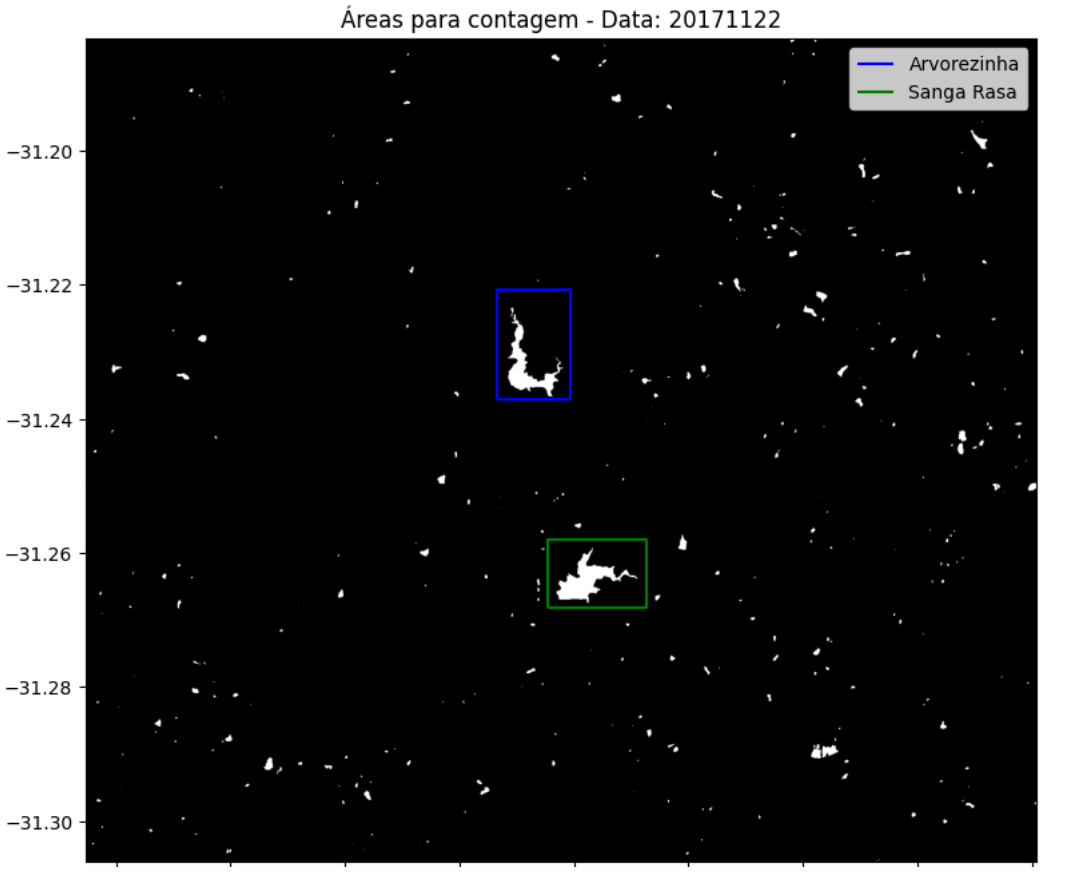


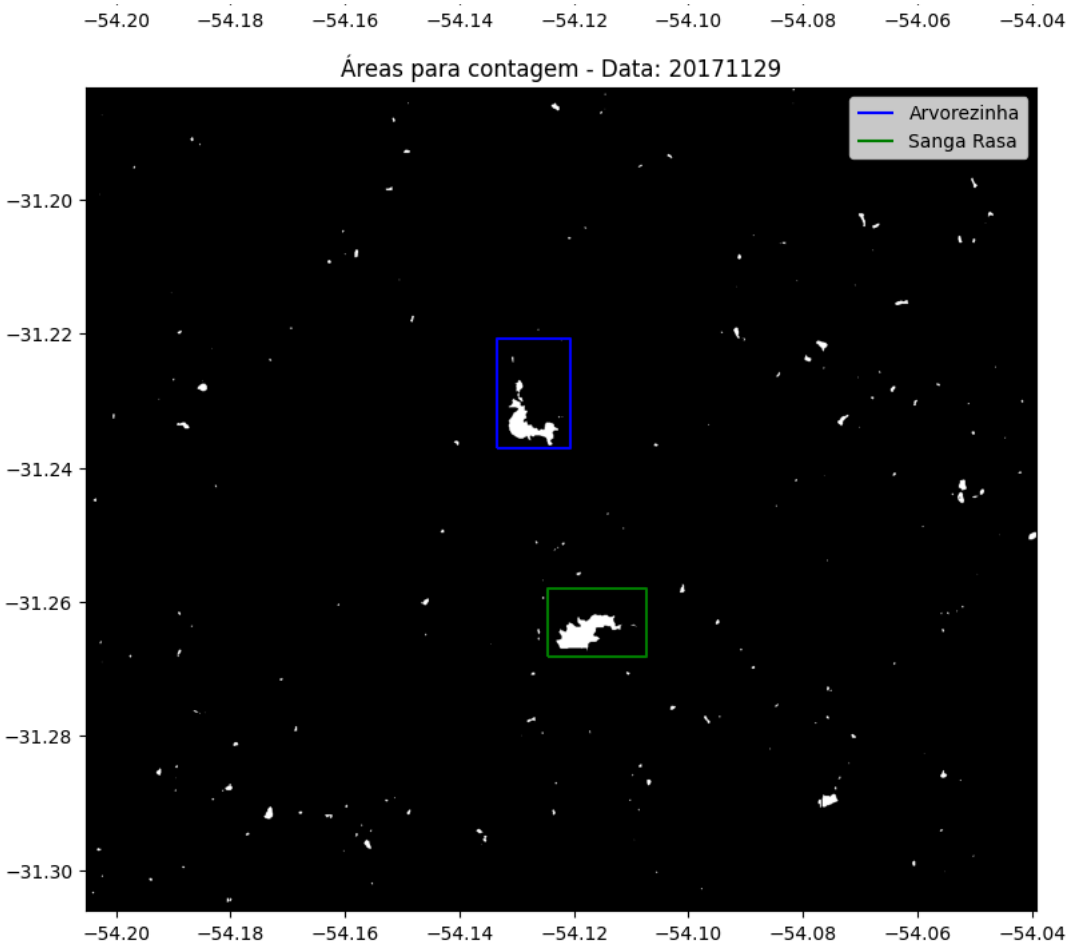
Áreas para contagem - Data: 20171112

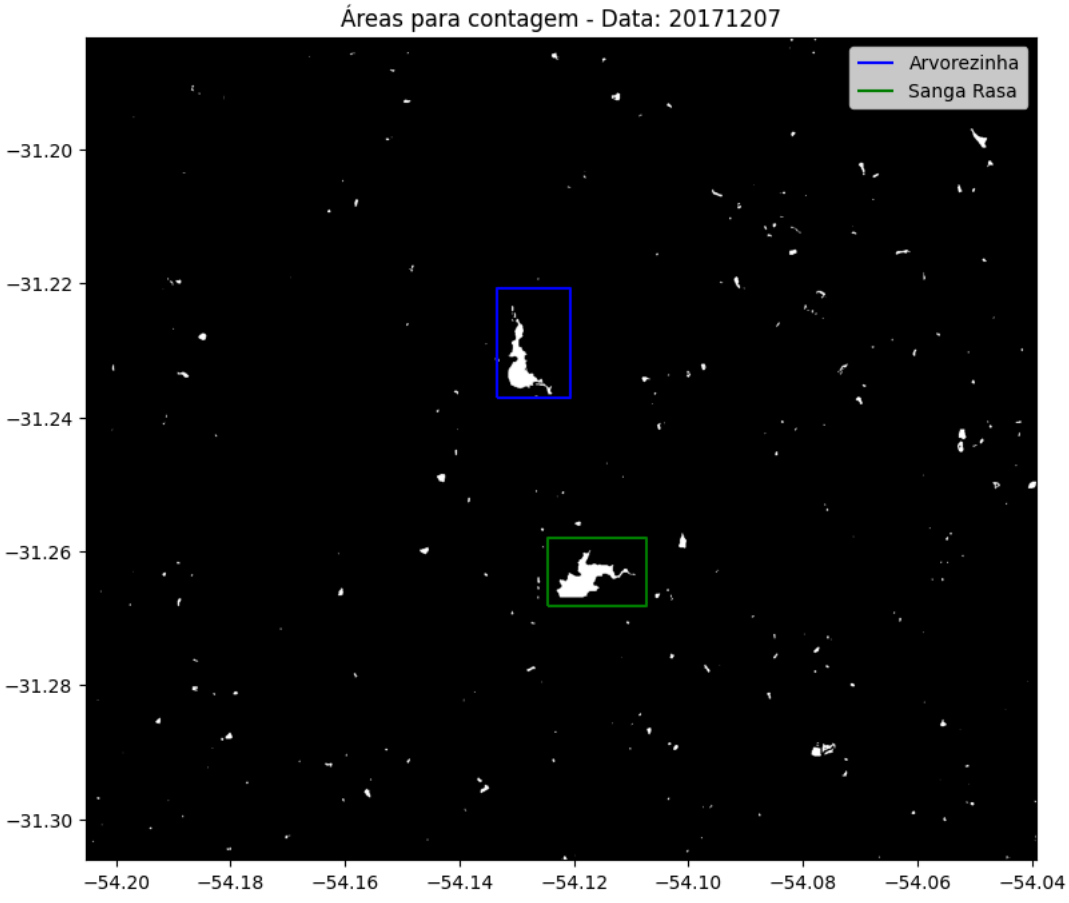



−31.28
−31.30
−54.20
−54.18
−54.16
−54.14
−54.12
−54.10
−54.08
−54.06
−54.04

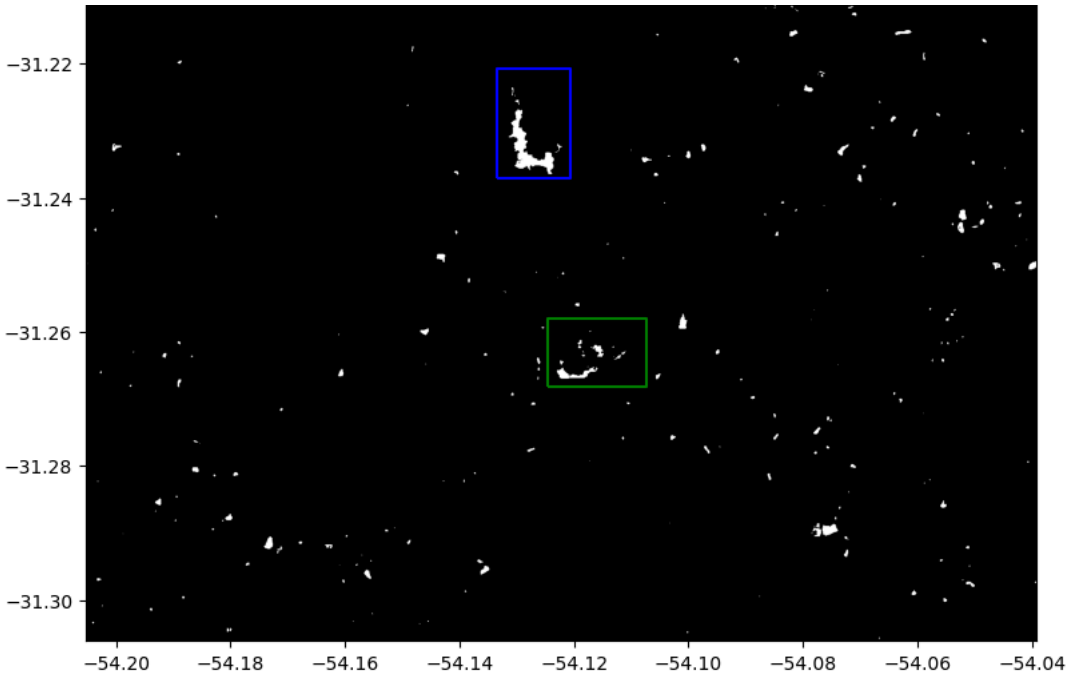

Áreas para contagem - Data: 20171119
Arvorezinha
Sanga Rasa
−31.20
−31.22
−31.24
−31.26
−31.28
−31.30
−54.20
−54.18
−54.16
−54.14
−54.12
−54.10
−54.08
−54.06
−54.04


Áreas para contagem - Data: 20171122
Arvorezinha
Sanga Rasa
−31.20
−31.22
−31.24
−31.26
−31.28
−31.30

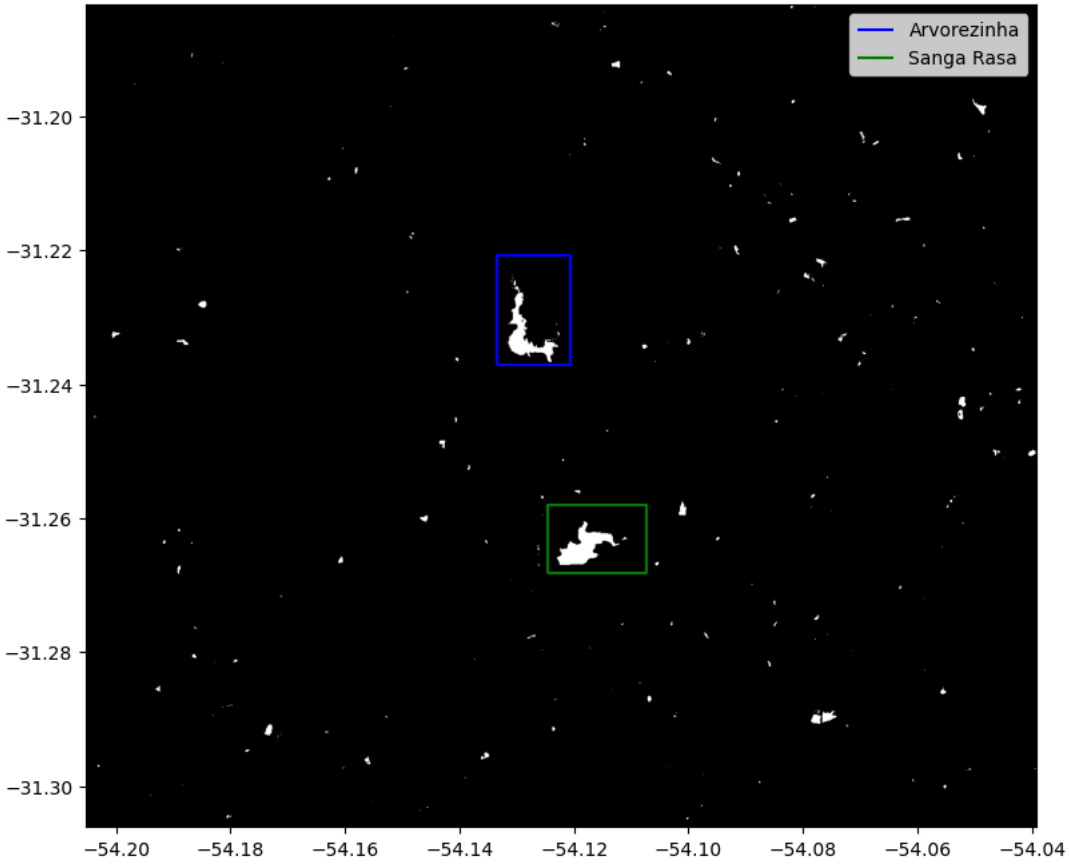

Áreas para contagem - Data: 20171129
Arvorezinha
Sanga Rasa
−54.20
−54.18
−54.16
−54.14
−54.12
−54.10
−54.08
−54.06
−54.04
−31.20
−31.22
−31.24
−31.26
−31.28
−31.30


Áreas para contagem - Data: 20171207
Arvorezinha
Sanga Rasa
−54.20
−54.18
−54.16
−54.14
−54.12
−54.10
−54.08
−54.06
−54.04
−31.20
−31.22
−31.24
−31.26
−31.28
−31.30

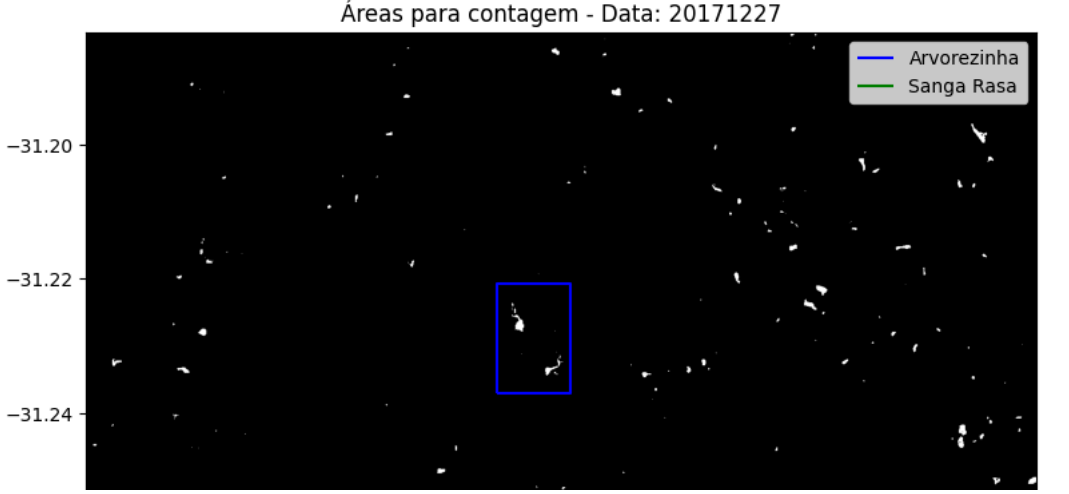

Áreas para contagem - Data: 20171209
Arvorezinha
Sanga Rasa
−31.20

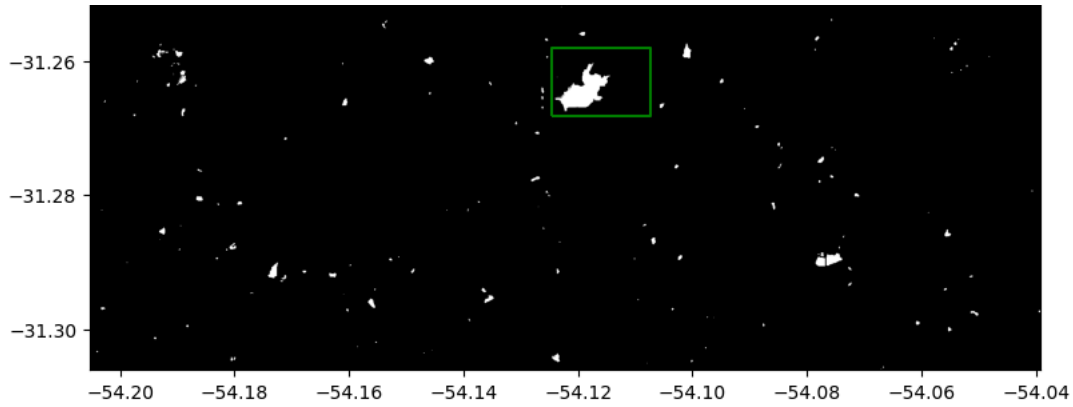


Áreas para contagem - Data: 20171212



Áreas para contagem - Data: 20171227



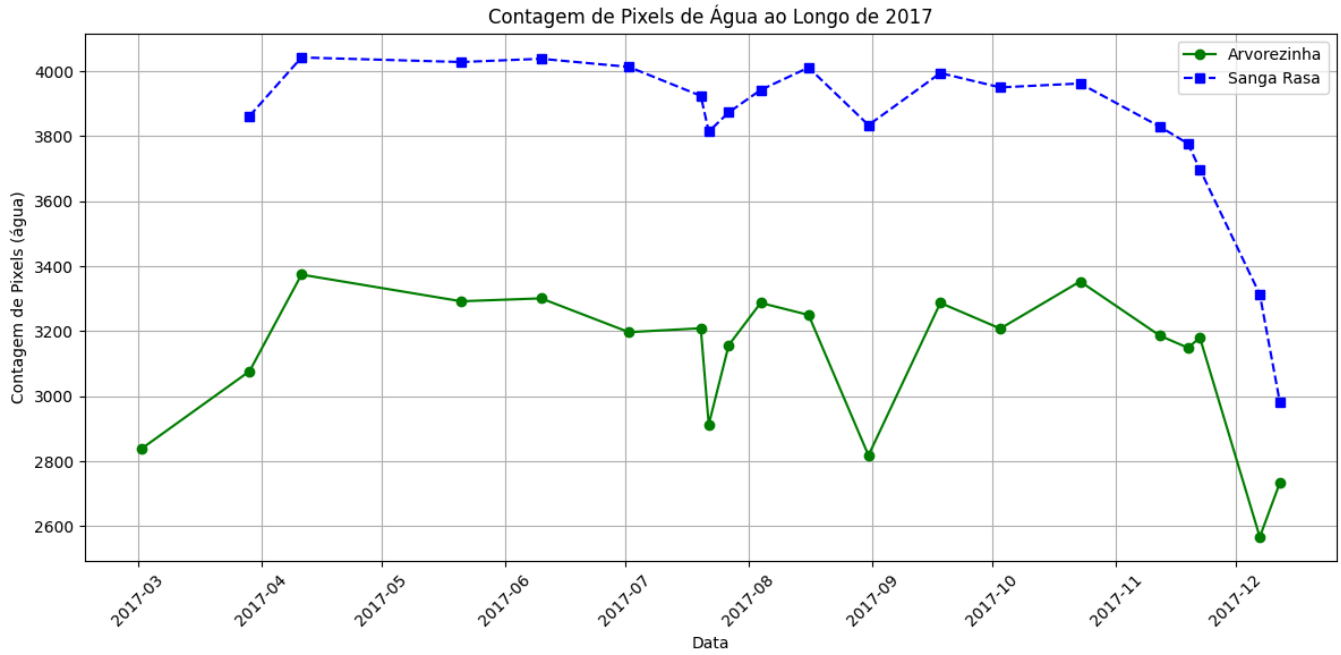

−31.26
−31.28
−31.30
−54.20
−54.18
−54.16
−54.14
−54.12
−54.10
−54.08
−54.06
−54.04

Análise da contagem de pixels

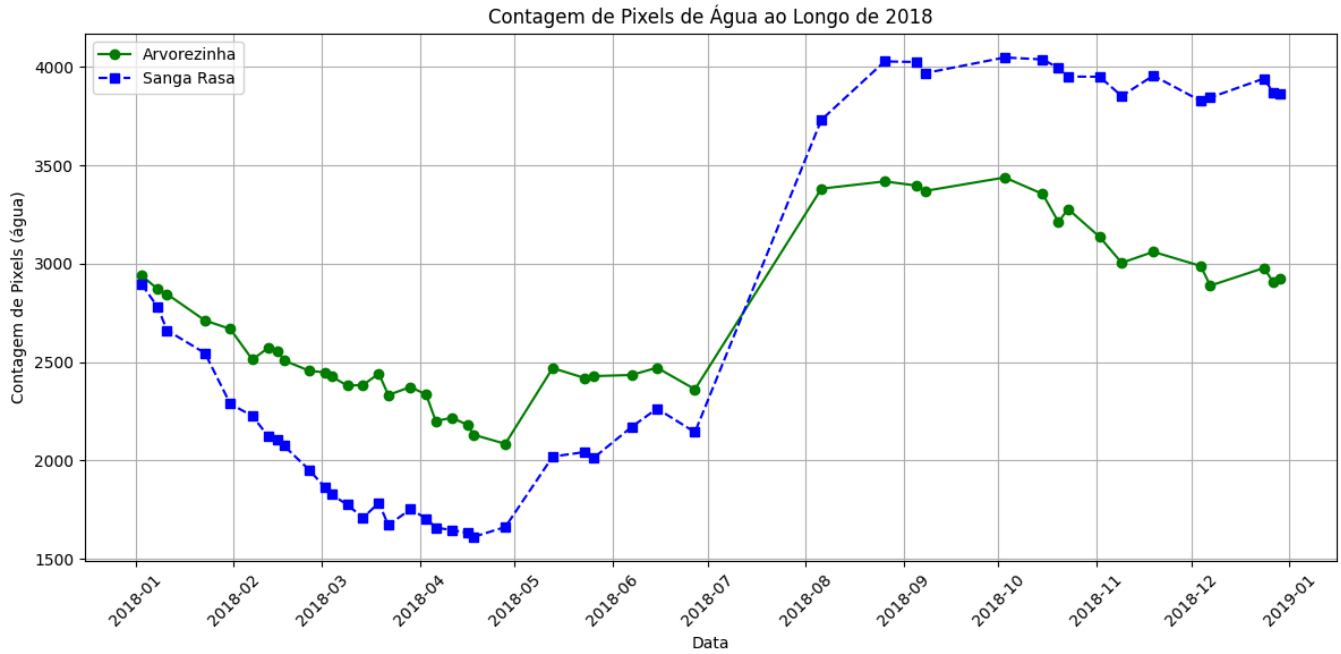
```
''''def plot_count_pixels(csv_file_path, year):
    df = pd.read_csv(csv_file_path)

    # converte a coluna 'data' para datetime
    df['data'] = pd.to_datetime(df['data'])
    df.replace("?", np.nan, inplace=True)

    df[['Arvorezinha', 'Sanga_Rasa']] = df[['Arvorezinha', 'Sanga_Rasa']].apply(pd.to_numeric)

    # interpolação pra grantir continuidade das linhas no grafico
    df[['Arvorezinha', 'Sanga_Rasa']] = df[['Arvorezinha', 'Sanga_Rasa']].interpolate(method='linear')

    df = df.sort_values('data')

    plt.figure(figsize=(12, 6))
    plt.plot(df['data'], df['Arvorezinha'], marker='o', linestyle='-', label='Arvorezinha', color='green')
    plt.plot(df['data'], df['Sanga_Rasa'], marker='s', linestyle='--', label='Sanga Rasa', color='blue')

    # título e eixos
    plt.title(f'Contagem de Pixels de Água ao Longo de {year}')
    plt.xlabel('Data')
    plt.ylabel('Contagem de Pixels (água)')

    # grade e legenda
    plt.grid(True)
    plt.legend()

    plt.gca().xaxis.set_major_locator(mdates.MonthLocator())
    plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))

    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()''''


def plot_count_pixels(csv_file_path, year):
    # tenta ler o arquivo, se não achar, avisa e para a execução da função
    try:
        # lê o csv, garantindo que a coluna 'data' seja lida como texto (string)
        # isso resolve a ambiguidade entre os formatos YYYYMMDD e YYYY-MM-DD
        df = pd.read_csv(csv_file_path, dtype={'data': str})
    except FileNotFoundError:
        print(f"ARQUIVO NÃO ENCONTRADO. Arquivo: {csv_file_path}")
        return

    # troca os '?' por NaN, um valor que o pandas entende como ausente
    df.replace("?", np.nan, inplace=True)

    # converte a coluna 'data' para datetime
    # o pandas se vira pra converter os diferentes formatos de data que são texto
    df['data'] = pd.to_datetime(df['data'])

    # transforma as colunas de interesse em tipo numérico
    # o 'coerce' força erros de conversão a virarem NaN, o que evita que o script quebre
    df[['Arvorezinha', 'Sanga_Rasa']] = df[['Arvorezinha', 'Sanga_Rasa']].apply(pd.to_numeric, errors='coerce')

    # interpola os valores ausentes pra linha do gráfico não ter falhas
    df[['Arvorezinha', 'Sanga_Rasa']] = df[['Arvorezinha', 'Sanga_Rasa']].interpolate(method='linear')

    # garante que as datas estejam em ordem cronológica
    df = df.sort_values('data')

    # aqui começa a criação do gráfico
    plt.figure(figsize=(12, 6))
    plt.plot(df['data'], df['Arvorezinha'], marker='o', linestyle='-', label='Arvorezinha', color='green')
    plt.plot(df['data'], df['Sanga_Rasa'], marker='s', linestyle='--', label='Sanga Rasa', color='blue')

    # título e nome dos eixos
    plt.title(f'Contagem de Pixels de Água ao Longo de {year}')
    plt.xlabel('Data')
    plt.ylabel('Contagem de Pixels (água)')

    # grade e legenda pra ficar mais fácil de ler
    plt.grid(True)
    plt.legend()

    # ajusta como as datas são mostradas no eixo x (um marcador por mês)
    plt.gca().xaxis.set_major_locator(mdates.MonthLocator())
```

```
    plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m'))

    # gira os nomes das datas pra não ficarem sobrepostos
    plt.xticks(rotation=45)
    plt.tight_layout() # ajusta o gráfico pra tudo caber na imagem
    plt.show()
```

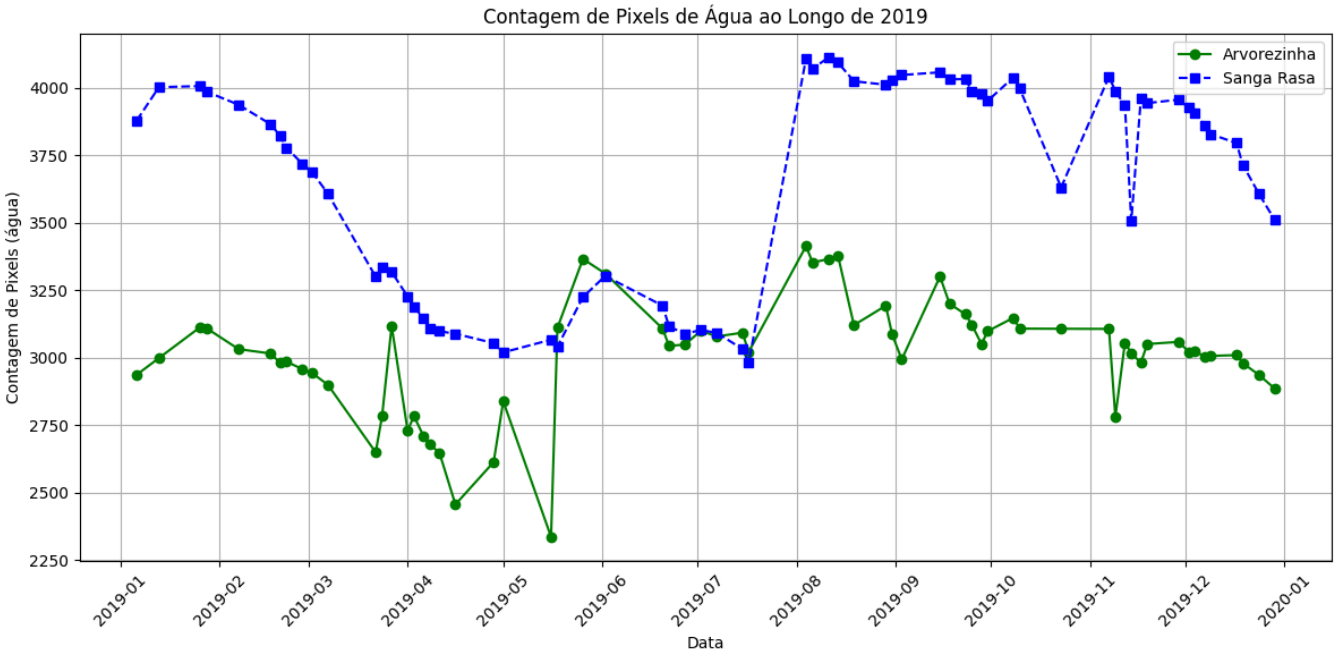
```
plot_count_pixels(csv_paths[2017], '2017')
```



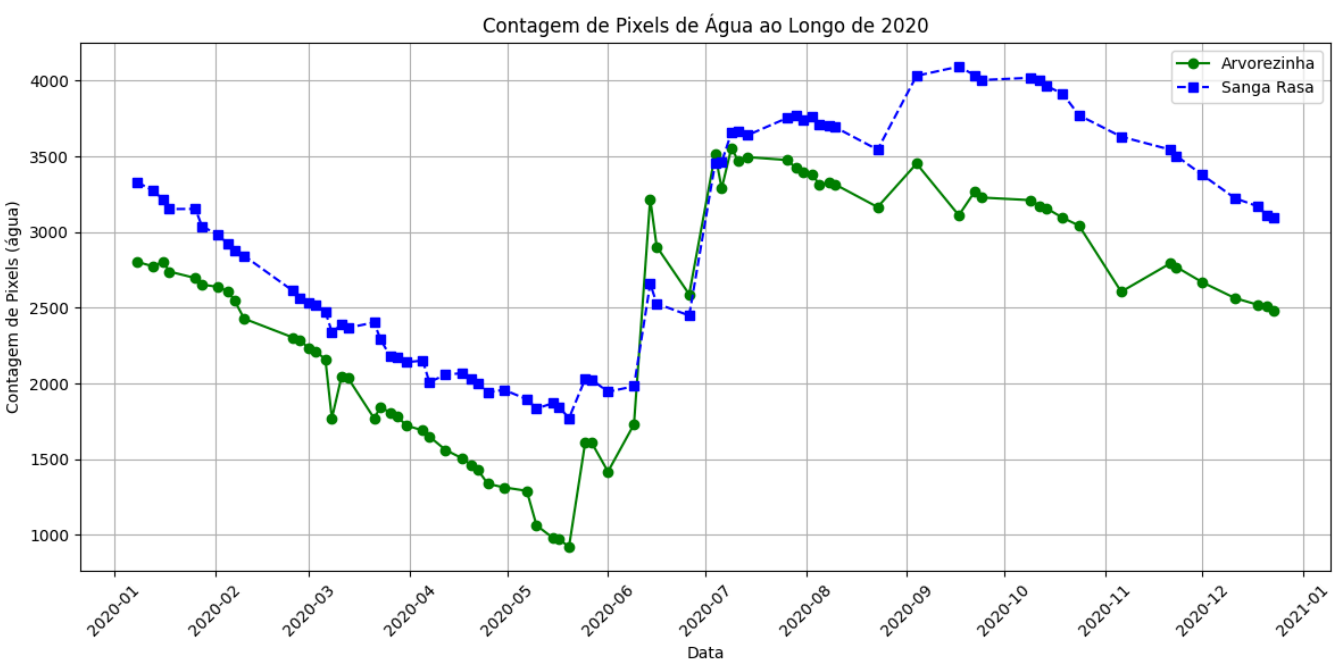

```
plot_count_pixels(csv_paths[2018], '2018')
```



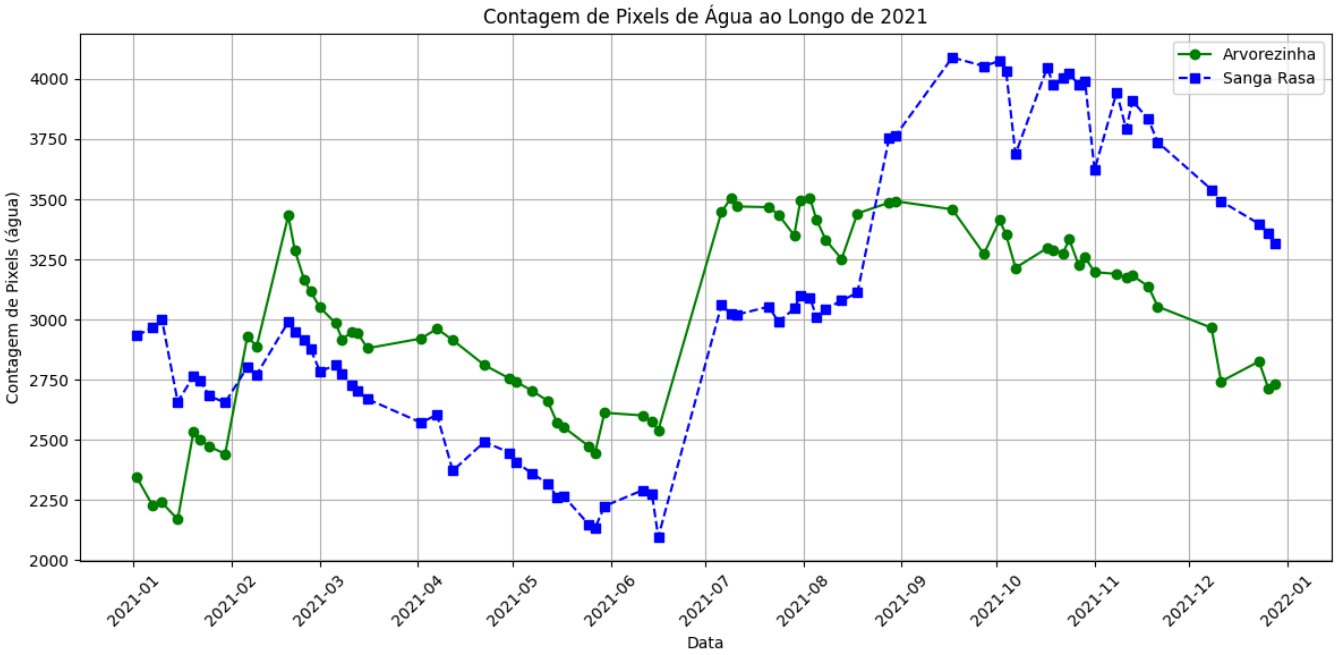

```
plot_count_pixels(csv_paths[2019], '2019')
```



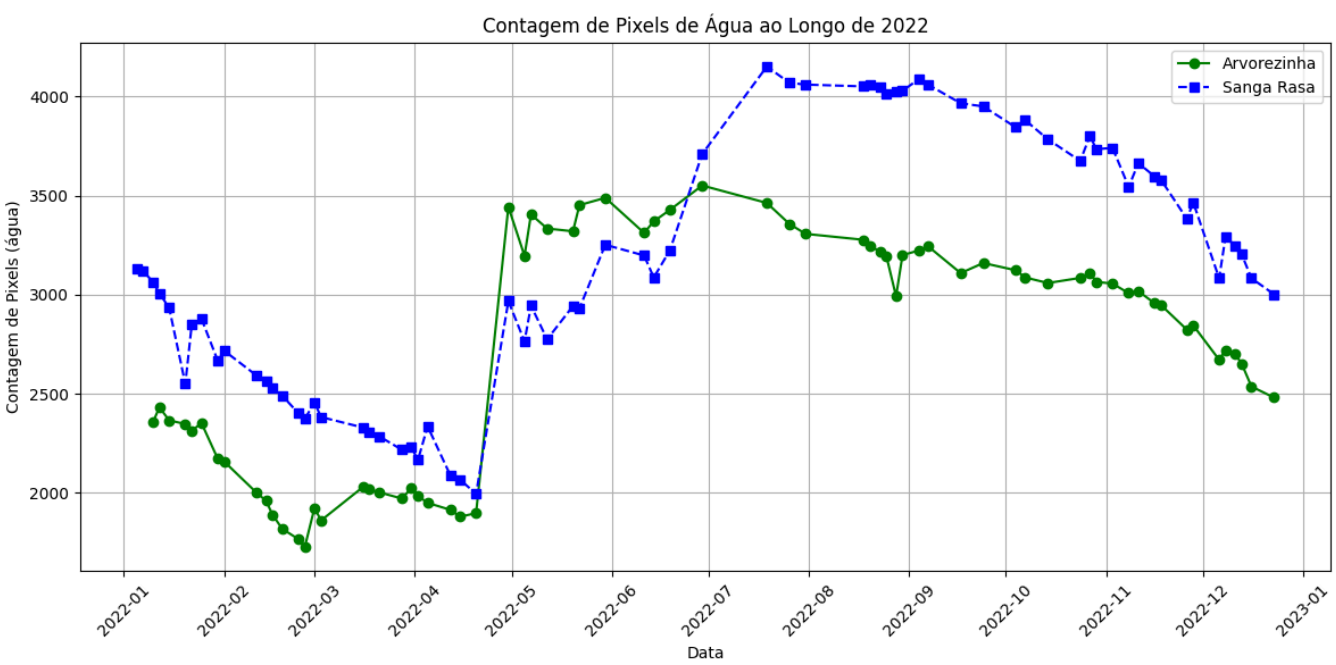

```
plot_count_pixels(csv_paths[2020], '2020')
```



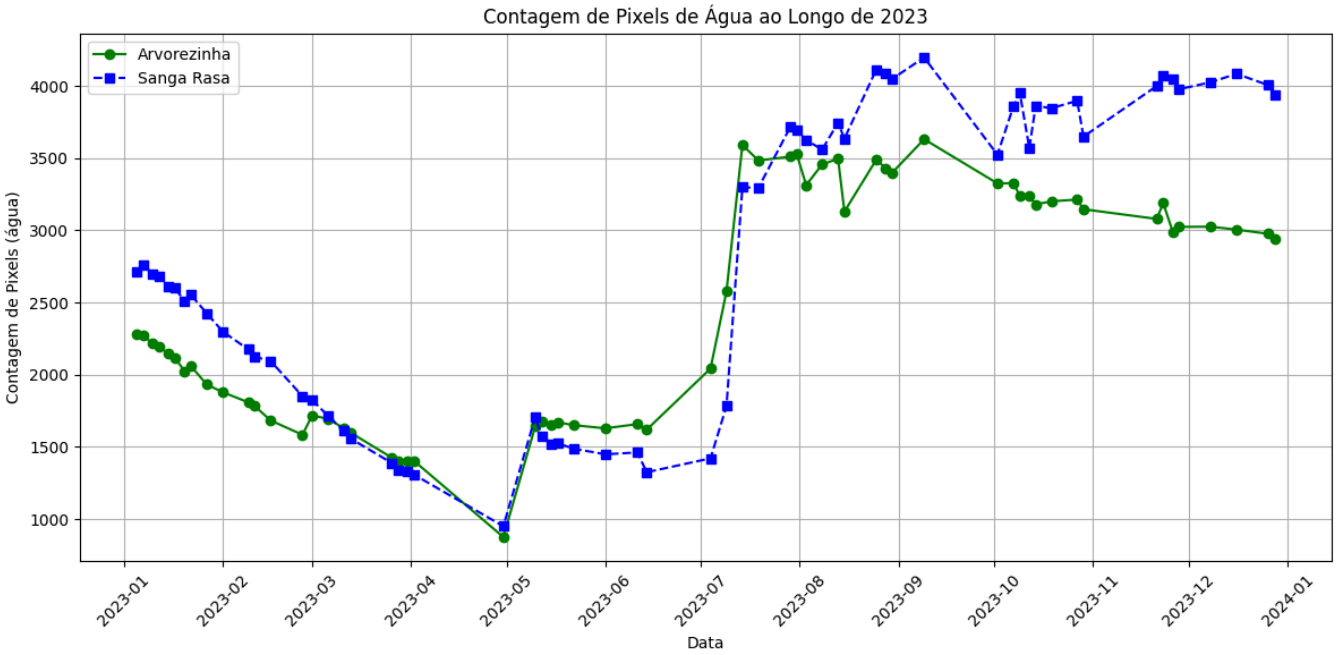

```
plot_count_pixels(csv_paths[2021], '2021')
```



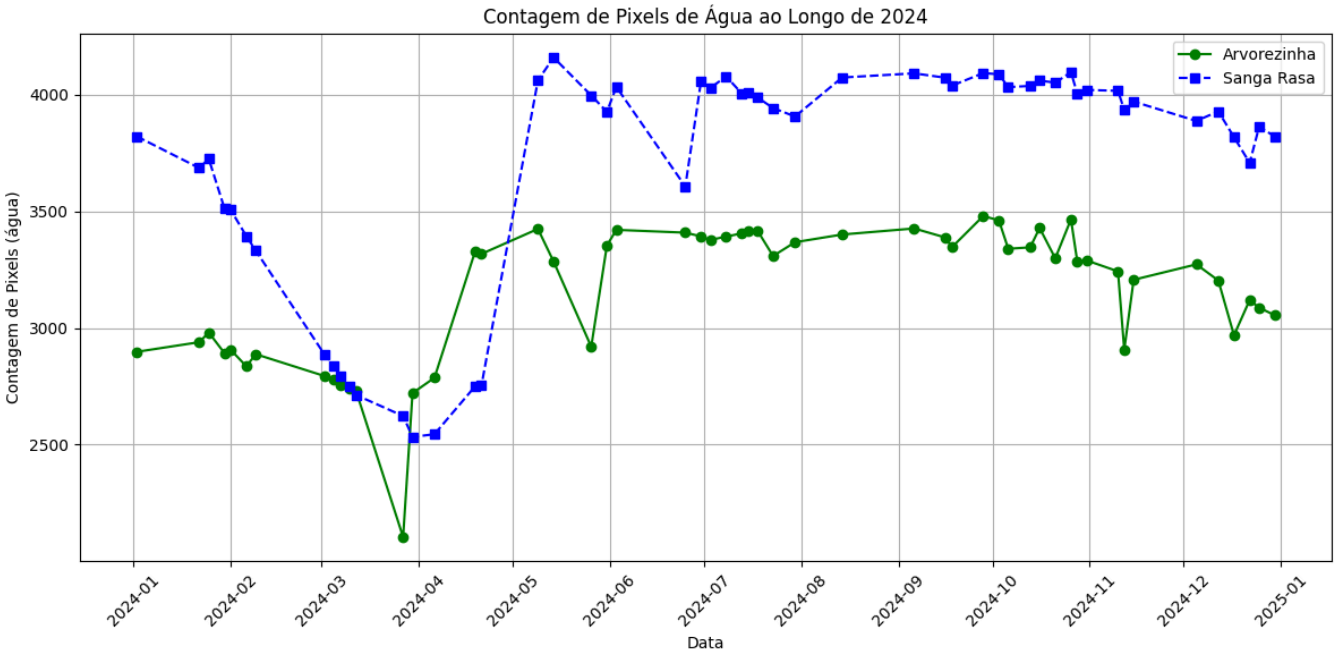

```
plot_count_pixels(csv_paths[2022], '2022')
```



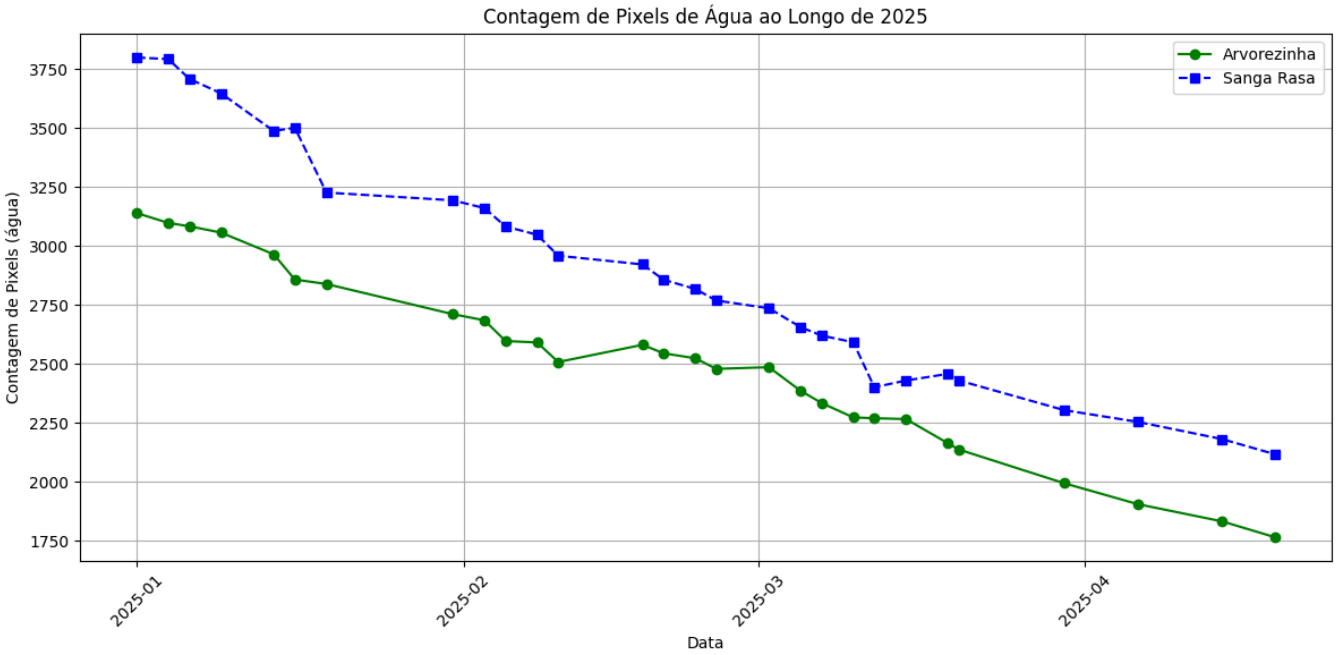

```
plot_count_pixels(csv_paths[2023], '2023')
```




```
plot_count_pixels(csv_paths[2024], '2024')
```

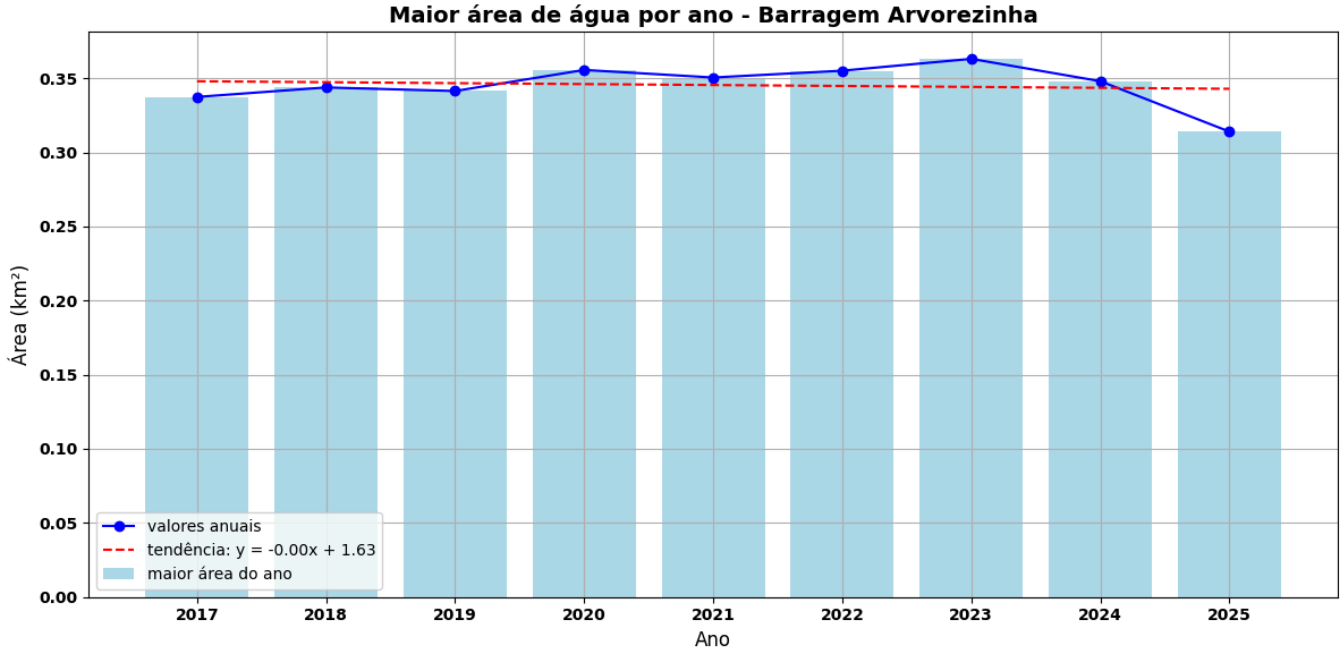


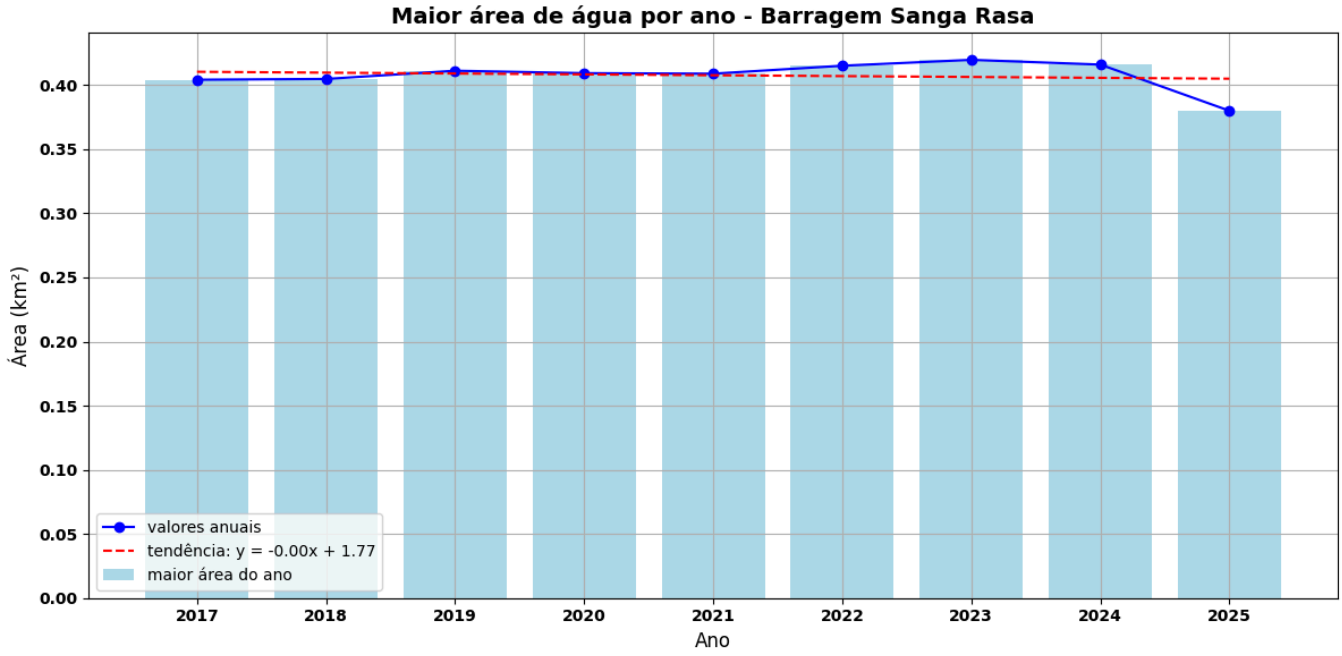

```
plot_count_pixels(csv_paths[2025], '2025')
```




```
# listas para armazenar os dados processados de cada barragem
arvorezinha_data = []
sanga_rasa_data = []

# processa cada arquivo csv listado em csv_paths
for year, path in csv_paths.items():
    # tenta ler o arquivo; se não encontrar, avisa e pula para o próximo ano
    try:
        # lê o csv, forçando a coluna 'data' a ser lida como texto (string)
        # isso ajuda o pandas a entender melhor os formatos YYYYMMDD e YYYY-MM-DD
        df = pd.read_csv(path, dtype={'data': str})
    except FileNotFoundError:
        print(f"ARQUIVO NÃO ENCONTRADO. Arquivo: {path}, pulando o ano {year}")
        continue # vai para a próxima iteração do loop

    # substitui "?" por NaN (Not a Number) para o pandas lidar melhor com dados ausentes
    df.replace("?", np.nan, inplace=True)

    # converte a coluna 'data' para o formato datetime
    # o pandas é bom em adivinhar o formato correto quando a coluna é texto
    df['data'] = pd.to_datetime(df['data']) # convertendo a coluna 'data' original

    # converte colunas de contagem de pixels para tipo numérico
    # 'errors='coerce'' faz com que qualquer valor não conversível vire NaN
    df[['Arvorezinha', 'Sanga_Rasa']] = df[['Arvorezinha', 'Sanga_Rasa']].apply(pd.to_numeric, errors='coerce')

    # interpola valores NaN (ausentes) para garantir a continuidade dos dados
    # isso ajuda a preencher "buracos" nas séries temporais de forma linear
    df[['Arvorezinha', 'Sanga_Rasa']] = df[['Arvorezinha', 'Sanga_Rasa']].interpolate(method='linear')

    # obtém o maior valor de pixel de água por barragem no ano
    max_arvorezinha = df['Arvorezinha'].max()
    max_sanga_rasa = df['Sanga_Rasa'].max()

    # converte o valor de pixel em área (m² → km²)
    # cada pixel representa 10m x 10m = 100 m². para km², dividimos por 1.000.000
    area_arvorezinha = (max_arvorezinha * 100) / 1_000_000 if pd.notna(max_arvorezinha) else np.nan
    area_sanga_rasa = (max_sanga_rasa * 100) / 1_000_000 if pd.notna(max_sanga_rasa) else np.nan

    # armazena os dados calculados para o ano atual
    arvorezinha_data.append({'year': year, 'area': area_arvorezinha})
    sanga_rasa_data.append({'year': year, 'area': area_sanga_rasa})

# cria dataframes para cada barragem a partir das listas populadas
df_arvorezinha = pd.DataFrame(arvorezinha_data)
df_sanga_rasa = pd.DataFrame(sanga_rasa_data)

# função para plotar os gráficos de área ao longo dos anos
def plot_water_area(df, title):
```

```
        plt.figure(figsize=(12, 6)) # define o tamanho da figura do gráfico

        # barras para mostrar a maior área de água registrada no ano
        plt.bar(df['year'], df['area'], color='lightblue', label='maior área do ano')
        # linha conectando os pontos anuais
        plt.plot(df['year'], df['area'], marker='o', linestyle='-', color='blue', label='valores anuais')

        # calcula a regressão linear para mostrar uma linha de tendência
        # 'df.dropna(subset=['year', 'area'])' garante que não haja NaNs na regressão
        df_cleaned = df.dropna(subset=['year', 'area'])
        if len(df_cleaned) >= 2: # precisa de pelo menos 2 pontos para uma linha
            slope, intercept, _, _, _ = linregress(df_cleaned['year'], df_cleaned['area'])
            trend_line = slope * df_cleaned['year'] + intercept
            plt.plot(df_cleaned['year'], trend_line, color='red', linestyle='--', label=f'tendência: y = {slope:.2f}x + {interce
        else:
            print(f"não há dados suficientes para calcular a linha de tendência para {title}")


        # título e nomes dos eixos, com um toque de estilo
        plt.title(f'Maior área de água por ano - {title}', fontsize=14, fontweight='bold')
        plt.xlabel('Ano', fontsize=12)
        plt.ylabel('Área (km²)', fontsize=12)
        plt.xticks(df['year'].unique(), fontweight='bold') # garante que todos os anos sejam mostrados e em negrito
        plt.yticks(fontweight='bold') # valores do eixo y em negrito

        plt.grid(True) # adiciona uma grade pra facilitar a leitura
        plt.legend() # mostra a legenda do gráfico
        plt.tight_layout() # ajusta o layout pra tudo caber direitinho
        plt.show() # exibe o gráfico

    # gera e exibe os gráficos para cada barragem
    plot_water_area(df_arvorezinha, 'Barragem Arvorezinha')
    plot_water_area(df_sanga_rasa, 'Barragem Sanga Rasa')
```

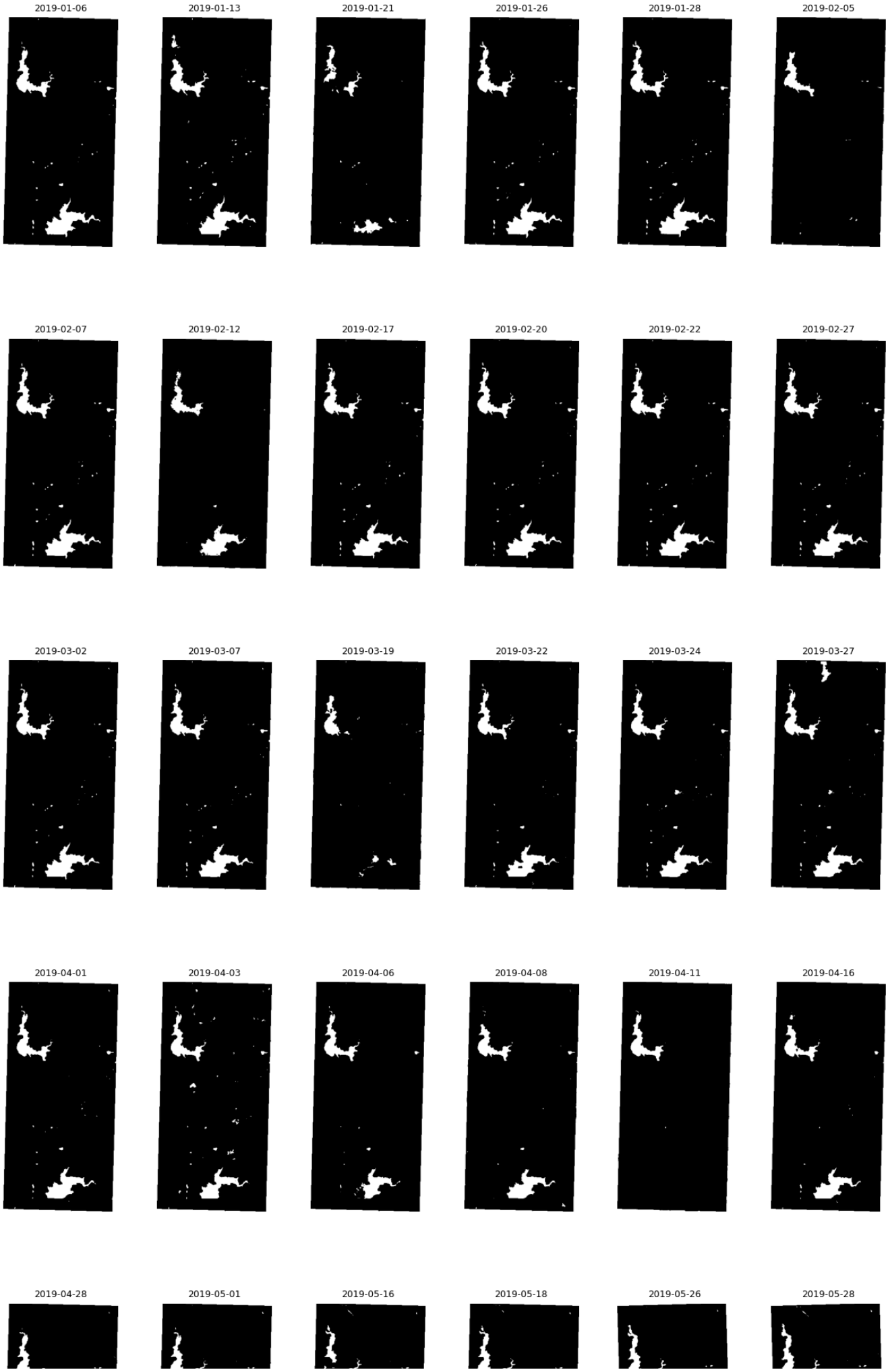


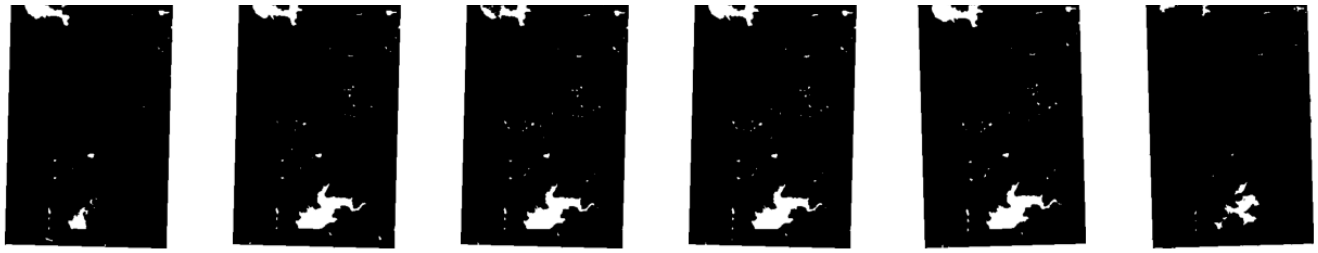


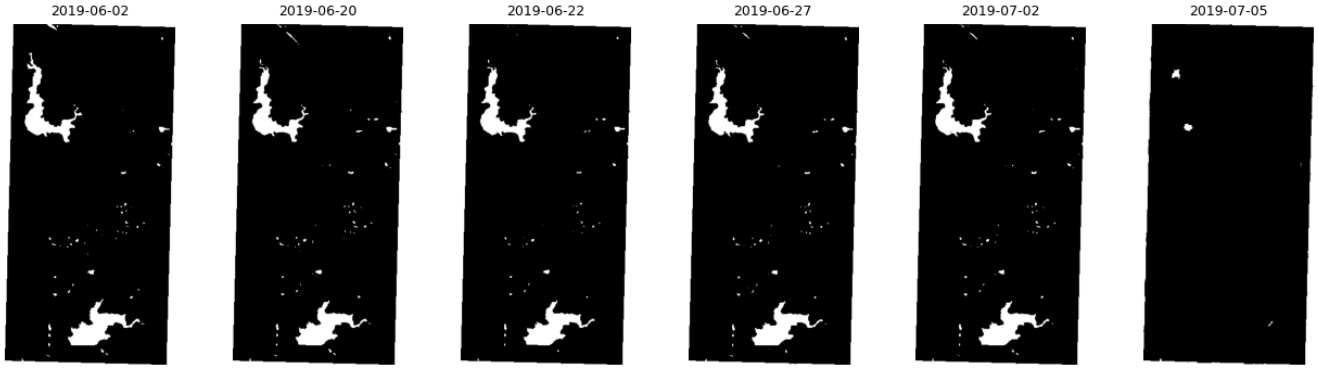

```
from glob import glob


# função para exibir as imagens .tif de uma pasta
def plot_tif_subplots(folder_path, year):
    # encontra todos os arquivos .tif
    tif_files = sorted(glob(os.path.join(folder_path, '*.tif')))

    if not tif_files:
        print(f"Nenhuma imagem .tif encontrada para o ano {year}")
        return

    num_images = len(tif_files)
    cols = 6
    rows = (num_images + cols - 1) // cols

    # cria a figura com subplots
    fig, axs = plt.subplots(rows, cols, figsize=(16, 5 * rows))
```

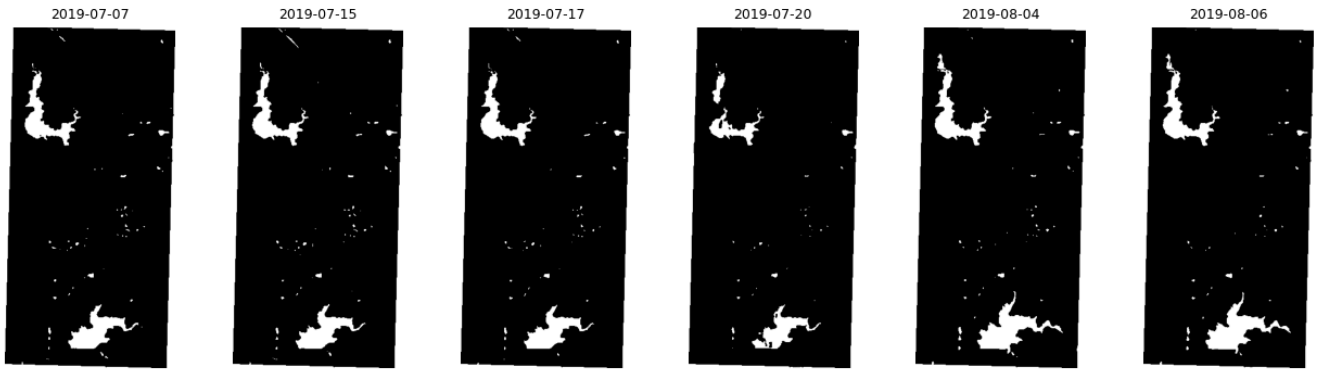
```
        axs = axs.flatten()

        for i, tif_path in enumerate(tif_files):
            with rasterio.open(tif_path) as src:
                img = src.read(1)
            axs[i].imshow(img, cmap='gray')
            axs[i].set_title(os.path.basename(tif_path).replace('.tif', ''), fontsize=9)
            axs[i].axis('off')

        # desativa eixos de subplots vazios
        for j in range(i + 1, len(axs)):
            axs[j].axis('off')

        fig.suptitle(f'Imagens da barragem em {year}', fontsize=16, fontweight='bold')
        plt.subplots_adjust(wspace=0.1, hspace=0.4, top=0.97)
        plt.show()

    # executa para cada pasta
    for year, folder in output_folders.items():
        plot_tif_subplots(folder, year)
```

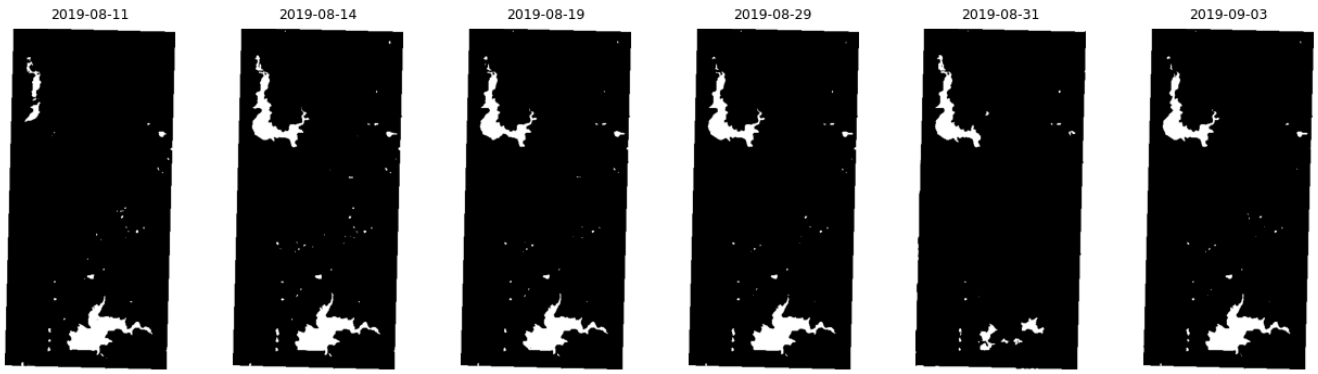
Imagens da barragem em 2019

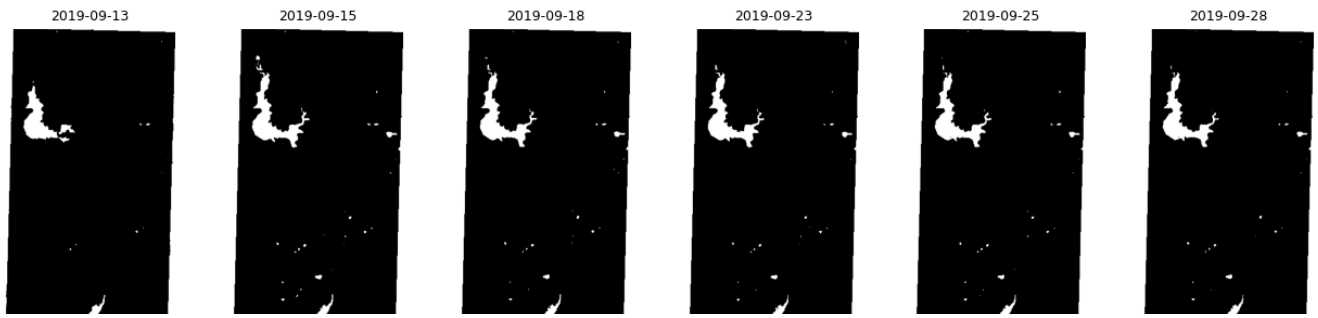


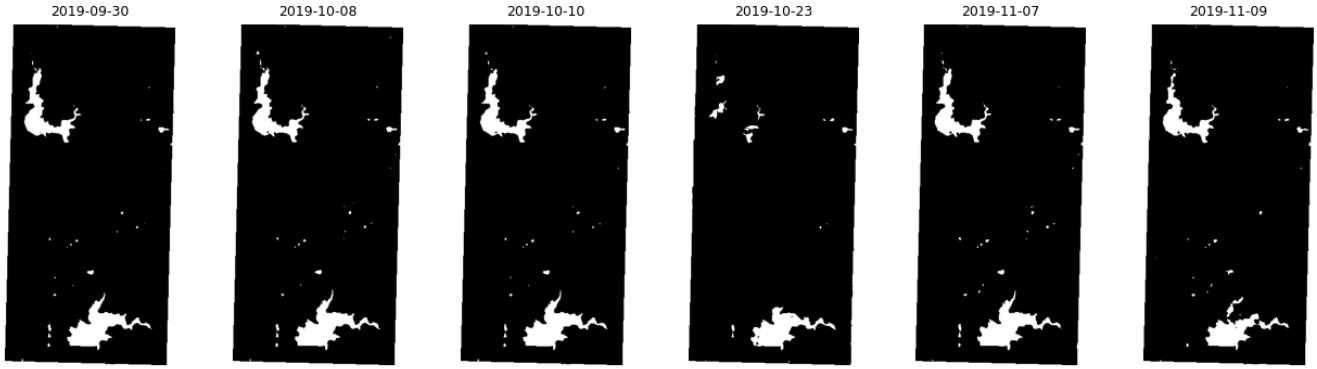


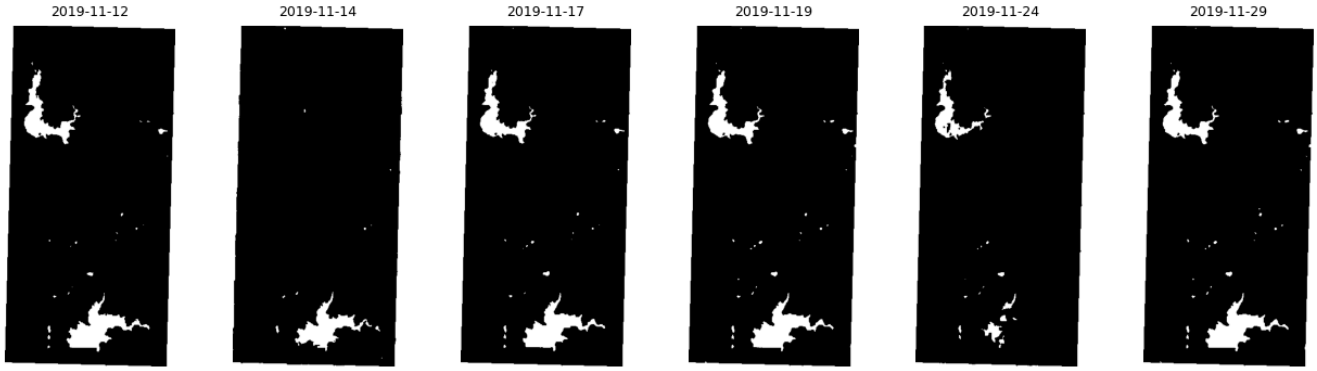

2019-06-02
2019-06-20
2019-06-22
2019-06-27
2019-07-02
2019-07-05

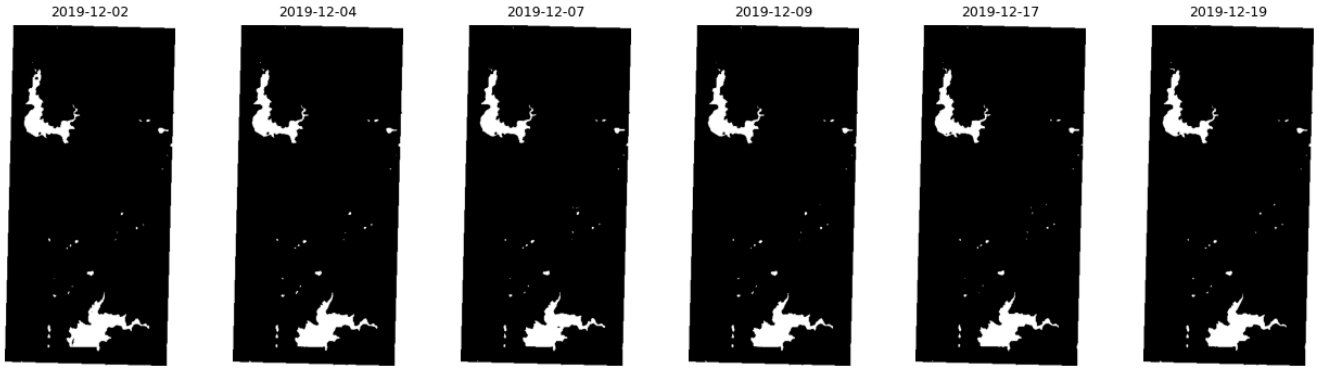

2019-07-07
2019-07-15
2019-07-17
2019-07-20
2019-08-04
2019-08-06

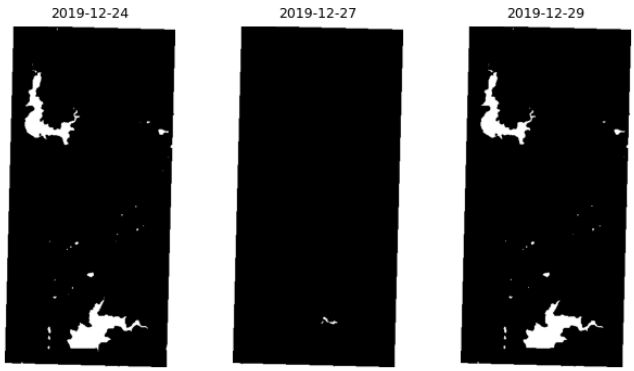

2019-08-11
2019-08-14
2019-08-19
2019-08-29
2019-08-31
2019-09-03


2019-09-13
2019-09-15
2019-09-18
2019-09-23
2019-09-25
2019-09-28









Imagens da barragem em 2020