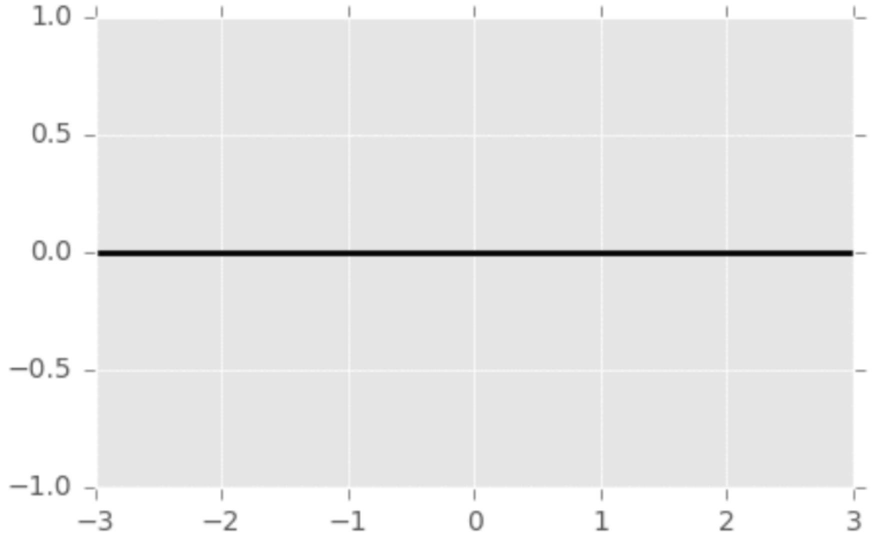
Spatial Data Analysis with matplotlib Animations

Advanced Topics in Databases

Spatial Data Animation

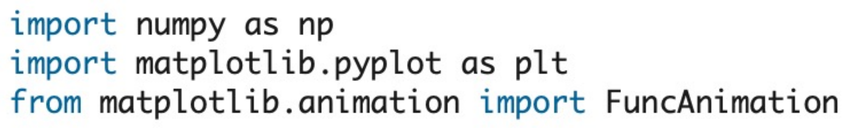
- Information exploration is an important task and the development of good visualization tools is essential;
- In many cases, visualization becomes complex when the information has more than two dimensions, as happens in the data we have used (longitude, latitude, and temporal dimension);
- In many examples, the possibility of animating data makes analysis tasks easier and better.

Animations with matplotlib (lines)



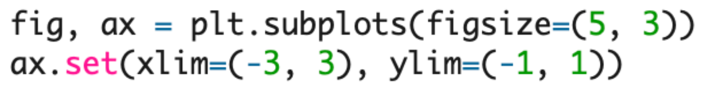
Step-by-step animation

Step 1: Import the necessary modules



```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
```

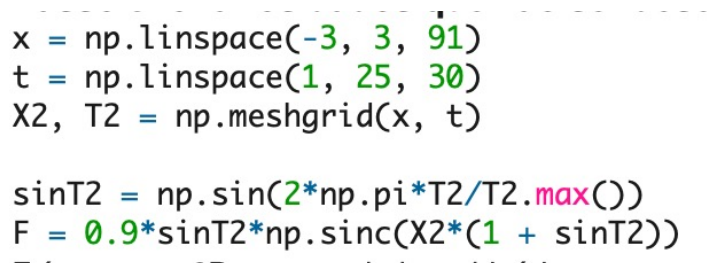
Step 2: Define the plot area



```python
fig, ax = plt.subplots(figsize=(5, 3))
ax.set(xlim=(-3, 3), ylim=(-1, 1))
```

The first line defines the figure and its axes. The second line sets the limits for the axes.

Step 3: Create the data that will be plotted



```python
x = np.linspace(-3, 3, 91)
t = np.linspace(1, 25, 30)
X2, T2 = np.meshgrid(x, t)

sinT2 = np.sin(2*np.pi*T2/T2.max())
F = 0.9*sinT2*np.sinc(X2*(1 + sinT2))
```

F is a 2D array that has arbitrary data to be drawn. In this case, they are not arbitrary, being defined to produce a perfect cyclic graph.

Step-by-step animation

Step 4: plot the first line

```
line = ax.plot(x, F[0, :], color='k', lw=2)[0]
```

This line of code defines a first line with the desired attributes (color and line thickness). Notice the [0] in end. This is necessary because the plot command returns a list of lines. Here we only intend to draw a single line, hence we index only the first (i.e., zeroth) in the list of lines.

Step 5: create a function to update the line

```
def animate(i):
    line.set_ydata(F[i, :])
```

This function takes one argument. The single command of this function changes the y-coordinates of the line. We'll see later commands to change other things.

Step-by-step animation

Step 6: call the FuncAnimation and show

```
anim = FuncAnimation(
    fig, animate, interval=100, frames=len(t)-1)

plt.draw()
plt.show()
```

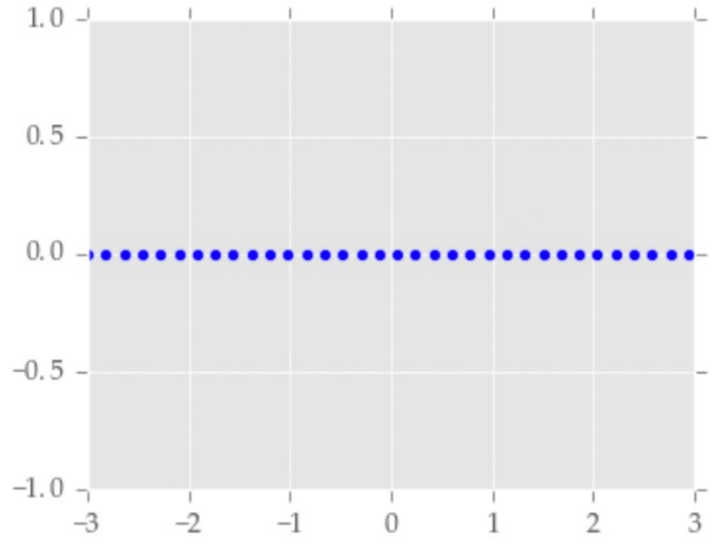
FuncAnimation receives two arguments: the figure object ‘fig’ and the animation function ‘animate’. The interval argument is optional and defines the interval between frames in milliseconds. The argument of frames is only necessary if we export the animation. The last two lines are necessary or not depending on
if the animation is to be exported.

Optional step: export the animation

To export the animation, for example to a video in mpeg format, simply run the command:

```
anim.save('filename.mp4')
```

Scatter Animation



Complete scatter code

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

def animate(i):
    y_i = F[i]
    scat.set_offsets(np.c_[x, y_i])


fig, ax = plt.subplots(figsize=(5, 3))
ax.set(xlim=(-3, 3), ylim=(-1, 1))

x = np.linspace(-3, 3, 9)
t = np.linspace(1, 25, 30)
X2, T2 = np.meshgrid(x, t)

sinT2 = np.sin(2*np.pi*T2/T2.max())
F = 0.9*sinT2*np.sinc(X2*(1 + sinT2))

scat = ax.scatter(x, F[0])

anim = FuncAnimation(
    fig, animate, interval=100, frames=len(t)-1)

plt.draw()
plt.show()
```

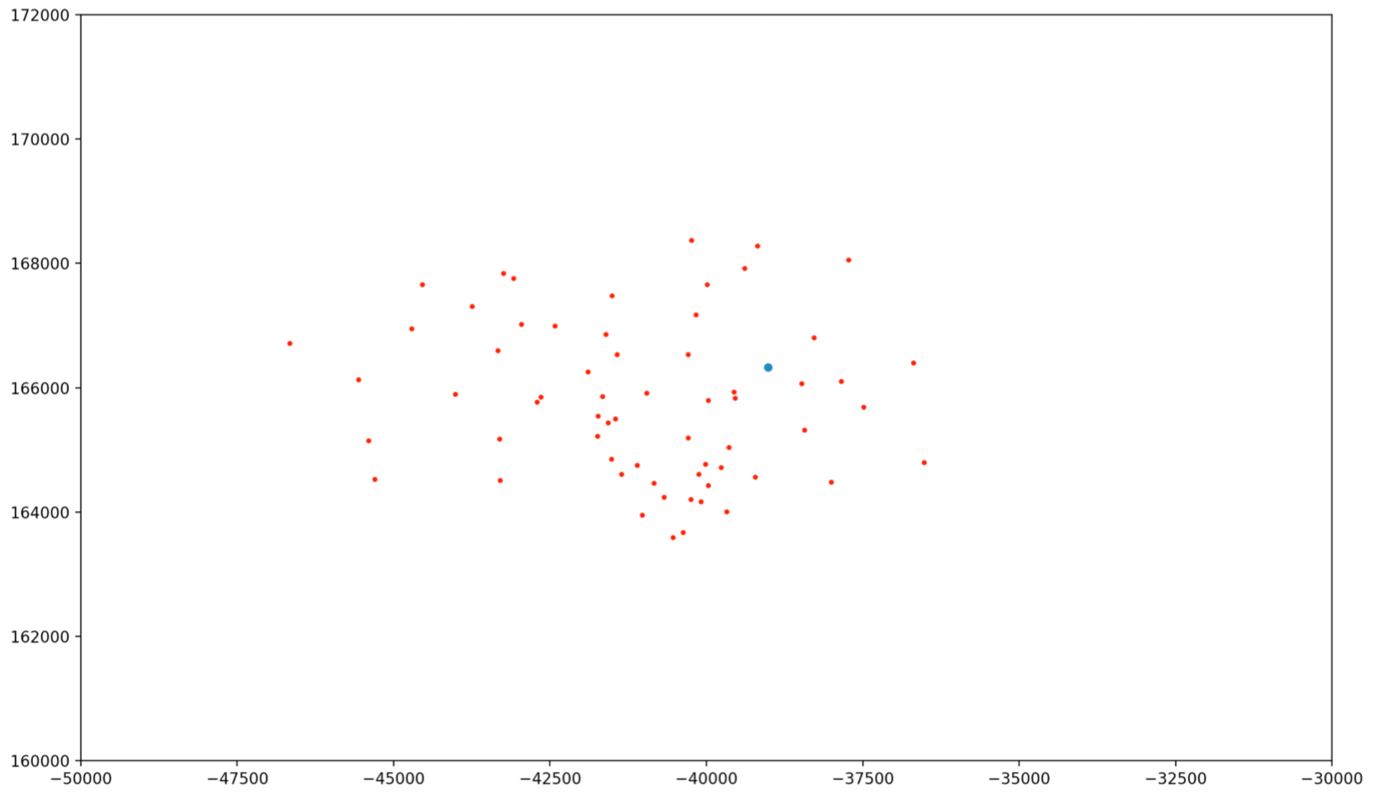
Animation of the trajectories of taxis

```
import numpy as np
import matplotlib.pyplot as plt
import psycopg2
import math
from matplotlib.animation import FuncAnimation

def animate(i):
    scat.set_offsets([x[i],y[i]])

def linestring_to_points(line_string):
    xs, ys = [],[]
    points = line_string[11:-1].split(',')
    for point in points:
        (x,y) = point.split()
        xs.append(float(x))
        ys.append(float(y))
    return xs,ys

scale=1/60000
conn = psycopg2.connect("dbname=michelferreira user=michelferreira")
cursor_psql = conn.cursor()


(continua)…
```

Animation of the trajectories of taxis

```
(continuação)…

xs_min, xs_max, ys_min, ys_max = -50000, -30000, 160000, 172000
width_in_inches = (xs_max-xs_min)/0.0254*1.1
height_in_inches = (ys_max-ys_min)/0.0254*1.1

fig, ax = plt.subplots(figsize=(width_in_inches*scale, height_in_inches*scale))
ax.set(xlim=(xs_min, xs_max), ylim=(ys_min, ys_max))

sql = """select st_astext(proj_track) from tracks where taxi='20000333' order by ts"""
cursor_psql.execute(sql)
results = cursor_psql.fetchall()

x, y = [], []
for row in results:
    xs, ys = linestring_to_points(row[0])
    x = x + xs
    y = y + ys

scat = ax.scatter(x[0],y[0],s=10)
anim = FuncAnimation(
    fig, animate, interval=1, frames=len(y)-1)

plt.draw()
plt.show()
conn.close()
```

Animation of the trajectories of taxis


172000
170000
168000
166000
164000
162000
160000
−50000
−47500
−45000
−42500
−40000
−37500
−35000
−32500
−30000