Pandas: Plots

Dr. Cornelis Stal

April 28, 2022

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

1 Data uit een `DataFrame` grafisch voorstellen met een plot

> Deze tutorial is een vertaling van de *Pandas Tutorial* op https://pandas.pydata.org/pandas-docs/stable/getting_started/.
>
> **Data:** voor deze tutorial zullen we gebruikmaken van de jaarlijkse vastgoedcijfers die bijgehouden en beschikbaar gemaakt worden door Statbel via deze link. We richten ons hierbij meer bepaald op de cijfers van verkoop van onroerende goederen (N) per jaar 2010-2021 voor de individuele gemeenten.

In deze tutorial zullen we verschillende methoden behandelen om de data grafisch voor te stellen met enkele plots. Hiervoor maken we gebruik van de `matplotlib`-bibliotheek.

Hier hier om de data te downloaden.

1.1 Voorbereiden van de data

In plaats van de data als een offline bestand te importeren in een `pandas DataFrame`, zullen we het MS Excel-bestand rechtstreeks van het web downloaden en importeren.

[ ]:
```
statbel = 'https://statbel.fgov.be/sites/default/files/files/\
    documents/Bouwen%20%26%20wonen/2.1%20Vastgoedprijzen/NL_immo_jaar.xlsx'

vastgoed = pd.read_excel(statbel, sheet_name='Per gemeente', skiprows=2,
    usecols=['refnis', 'lokaliteit', 'jaar', 'aantal transacties',
    'mediaan prijs(€)', 'eerste kwartiel prijs(€)','derde kwartiel prijs(€)'])
vastgoed.head()
```

Het loont de moeite om de databron eerst eens grondig te bekijken. Merk bijvoorbeeld op dat de te importeren data zich bevindt in het tabblad `Per gemeente`, en dat er een aantal regels moeten worden overgeslagen. Dit resulteert echter in een dubbelzinnigheid van de kolomhoofdingen, aangezien verschillende variabelen gegeven worden voor verschillende typen vastgoed. Vermits we hier enkel geïntereseerd zijn in de groep 'Alle huizen met 2, 3, 4 of meer gevels (excl. appartementen)', zullen we enkel de eerste reeks kolomen inlezen.

De namen van de verschillende kolommen zijn verre van ideaal (speciale tekens, spaties, lange

namen, …). Voordat we verder gaan met deze demo zullen we deze namen veranderen. Volledigheidshalve converteren we ook de NIS-code van een geheel getal naar een tekstveld:

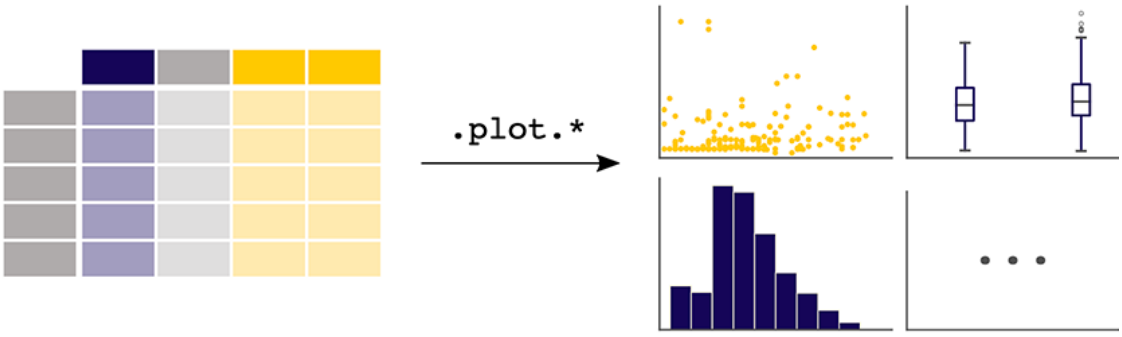
[ ]:
```
vastgoed = vastgoed.rename(
    columns={'refnis': 'NIS', 'lokaliteit': 'NAAM', 'jaar': 'JAAR',
    'aantal transacties': 'AANTAL', 'mediaan prijs(€)': 'MEDIAAN',
    'eerste kwartiel prijs(€)': 'Q1', 'derde kwartiel prijs(€)': 'Q3'},
    errors="raise"
)
vastgoed['NIS'] = vastgoed['NIS'].astype(str)
vastgoed.head()
```

De jaartallen converteren we vervolgens van gehele getallen naar het `Timestamp`-datatype met behulp van de `pd.to_datetime()`-methode. Deze resulteert in een datum (eerste dag van ieder jaar), waardoor we aanvullende de jaren nog moeten extraheren met de `pd.DatetimeIndex()`-methode. Tot slot stellen we deze jaartallen in als nieuw index-veld door middel van de `set_index()`-methode.

[ ]:
```
vastgoed['JAAR'] = pd.to_datetime(vastgoed['JAAR'], format='%Y')
vastgoed['JAAR'] = pd.DatetimeIndex(vastgoed['JAAR']).year
vastgoed = vastgoed.set_index('JAAR')
vastgoed.info()
```

> **Opmerking:** deze laatste stappen hadden we in principe ook al uit kunnen voeren bij het importeren. De `parse_date`-attribuut zou de jaartallen rechtstreeks omgezet hebben van gehele getallen naar `Timestamp`-objecten. Met de `index_col`-attribuut hadden we het volgnummer van de kolom met indices mee kunnen geven.

1.2 Dataplots maken met `pandas` en `matplotlib`



Het plotten van alle data tegelijk geeft vanzelfsprekend niet het beste resultaat:

[ ]:
```
vastgoed.plot()
```

Met een `DataFrame` zal `pandas` namelijk voor iedere kolom met numerieke data standaard een aparte lijn aanmaken binnen een plot. In ons geval zal het dus een goed idee zijn om te werken met een selectie. In onderstaand voorbeeld geven we de mediaanprijs, Q1- en Q3 prijzen voor de stad Gent:

[ ]:
```
vastgoed.loc[vastgoed['NAAM'] == 'GENT',
    ['MEDIAAN', 'Q1', 'Q3']].plot(
    title='Vstgoedprijzen in Gent (in €)'
)
```

We gebruiken de selectie-methodes zoals betroken in de vorige tutorial om data uit een specifieke kolom te plotten. Op basis van bovenstaande code kunnen we al opmaken dat de `plot()`-methode werkt op zowel `DataFrame`- als `Series`-objecten.

Door gebruik te maken van de combinatie van een aantal interessante functies en methodes kunnen we de mediaanprijzen per jaar vergelijken tussen Oost-Vlaanderen en Limburg:

[ ]:
```
x = pd.DataFrame()

x['Limburg'] = vastgoed.loc[vastgoed["NIS"].str.startswith('7'),
    'MEDIAAN'].groupby('JAAR').median()
x['Oost-Vlaanderen'] = vastgoed.loc[vastgoed["NIS"].str.
    startswith('4'), 'MEDIAAN'].groupby('JAAR').median()

ax = x.plot.scatter(x="Limburg", y="Oost-Vlaanderen", alpha=0.5)

for k, v in x.iterrows():
    ax.annotate(k, v)

import numpy as np
lims = [
    np.min([ax.get_xlim(), ax.get_ylim()]),
    np.max([ax.get_xlim(), ax.get_ylim()])
]

ax.plot(lims, lims, 'k-', alpha=0.75, zorder=0)
ax.set_aspect('equal')
ax.set_xlim(lims)
ax.set_ylim(lims)
```

De volgende stukken code leveren de bovenstaande grafiek:

- `loc[vastgoed["NIS"].str.startswith('7'), 'MEDIAAN']`: selecteert alle rijen de kolom `MEDIAAN` in de `DataFrame` waarvoor het veld `NIS` start met een waarde `7` (= alle gemeenten in Limburg
- `groupby('JAAR')`: groepeer alle waarden in functie van iedere unieke waarde in de kolom `JAAR`
- `median()`: in combinatie met de `groupby('JAAR')`-methode wordt hier de mediaanwaarde per jaar berekend
- `x['Limburg'] = ...`: het resultaat wordt opgeslagen in een `Series`-object, dat aan een nieuwe kolom `Limburg` wordt toegevoegd in de `DataFrame x`
- `x['Oost-Vlaanderen'] = ...`: voor Oost-Vlaanderen doen we nog eens hetzelfde, maar dan geselecteerd op de NIS-codes startend met een waarde `7`
- `ax = x.plot.scatter(x="Limburg", y="Oost-Vlaanderen", alpha=0.5)`: de jaarlijkse

mediaanwaarden worden voor Limburg en Oost-Vlaanderen tegen elkaar uitgezet
- `for k, v in x.iterrows(): ax.annotate(k, v)`: voeg aan ieder punt een label toe voor het jaar (hier de index)
- Laatste deel van de code: een `x=y` lijn toevoegen volgens de limieten van de grafiek

Naast de `line`-plot die standaard gebruikt wordt bij het aanroepen van de `plot()`-functie, kunnen we dus ook scatterplots maken met de `plot.scatter()`-methode. Er bestaan echter nog meer methodes om verschillende grafieken aan te maken. We kunnen deze als volgt bekijken:

[ ]:
```
[
    graphType
    for graphType in dir(vastgoed.plot)
    if not graphType.startswith("_")
]
```

> **Opmerking:** in veel programmeeromgevingen, maar ook in IPython en Jupyter Notebook kan de TAB-knop gebruikt worden om een overzicht te krijgen van de beschikbare eigenschappen en methodes die bij een object horen. Bekijk maar eens wat er gebeurt na het intypen van `x.plot.`, gevolgd door een TAB…

Een van de beschikbare opties is `DataFrame.plot.box()`wat uiteraard verwijst naar het aanmaken van een boxplot. De `box()`-methode kan bijvoorbeeld worden toegepast op de mediaanprijzen voor een gegeven jaar en voor een gegeven provincie:

[ ]:
```
limburg = pd.DataFrame()
limburg['MEDIAAN'] = vastgoed.loc[(vastgoed["NIS"].str.startswith('7') &
    (vastgoed.index == 2020)), 'MEDIAAN']
limburg = limburg.rename(columns={'MEDIAAN': 'LIMBURG'})

oost = pd.DataFrame()
oost['MEDIAAN'] = vastgoed.loc[(vastgoed["NIS"].str.startswith('4') &
    (vastgoed.index == 2020)), 'MEDIAAN']
oost = oost.rename(columns={'MEDIAAN': 'OOST-VLAANDEREN'})

x = limburg.merge(oost,how='left', left_on='JAAR', right_on='JAAR')
x.plot.box()
```

> **Gebruikshandleiding:** in deze tutorial beperken we ons tot een aantal standaardgrafiekjes. Voor meer informatie over meer geavanceerde plots verwijzen we door naar de sectie over ondersteunde plotstijlen in de gebruikshandleiding.

We plotten nu een aantal individuele kolommen in aparte `subplot`.

[ ]:
```
limburg = pd.DataFrame()
limburg['MEDIAAN'] = vastgoed.loc[vastgoed["NIS"].str.startswith('7'),
    'MEDIAAN']
limburg = limburg.rename(columns={'MEDIAAN': 'LIMBURG'})

oost = pd.DataFrame()
oost['MEDIAAN'] = vastgoed.loc[vastgoed["NIS"].str.startswith('4'), 'MEDIAAN']
```

```
oost = oost.rename(columns={'MEDIAAN': 'OOST-VLAANDEREN'})

x = limburg.merge(oost,how='left', left_on='JAAR', right_on='JAAR')
x = x.groupby(x.index).median()
axs = x.plot.area(figsize=(12, 4), subplots=True)
```

Om data uit indivuele kolommen in aparte subplots te projecteren maken we gebruik van het `subplots`-argument binnen de `plot()`-functie. Deze ingebouwde optie is beschikbaar binnen iedere iedere `plot()`-functie binnen `pandas`. Het loont zeker de moeite de mogelijkheden van deze optie nader te bestuderen.

> **Gebruikshandleiding:** naast het `subplots`-argument worden meerdere opties behandeld in de sectie over het opmaken van plots in de gebruikshandleiding.

Als we de totale waarde van het verkochte vastgoed kunnen schatten aan de hand van de mediaan vermenigvuldigd met het aantal objecten (sterke vereenvoudiging), dan kunnen we de jaarlijkse waarde van de gehele Limburgse en Oost-Vlaamse vastgoedmarkt als volgt grafisch voorstellen:

[ ]:
```
waarde = vastgoed[['MEDIAAN', 'AANTAL', 'NIS']]
waarde['WAARDE'] = waarde['MEDIAAN'] * waarde['AANTAL'] / 1000000

limburg = pd.DataFrame()
limburg = waarde.loc[waarde["NIS"].str.startswith('7'), ['WAARDE']]
limburg = limburg.rename(columns={'MEDIAAN': 'LIMBURG'})

oost = pd.DataFrame()
oost = waarde.loc[waarde["NIS"].str.startswith('4'), ['WAARDE']]
oost = limburg.rename(columns={'MEDIAAN': 'OOST-VLAANDEREN'})

x = limburg.merge(oost,how='left', left_on='JAAR', right_on='JAAR')
x = x.groupby(x.index).sum()

fig, axs = plt.subplots(figsize=(12, 4))
x.plot.area(ax=axs)
axs.set_ylabel("Vastgoedwaarde (in M€)")
fig.savefig("TotaleVastgoedwaarde.png")
```

Iedere `plot()`-functie binnen `pandas` zal resulteren in een `matplotlib`-object. `matplotlib` maakt het mogelijk om grafieken geheel naar eigen wens vorm te geven en wordt daarom ook heel veel gebruikt voor datavisualisaties. Een andere veelgebruikte bibliotheek is `seaborn`. In bovenstaande voorbeeld maken we hier dankbaar gebruik van door bijvoorbeeld een as-label toe te voegen en de grafiek direct te exporteren als een png-bestand.

1.3 Te onthouden:

- De `.plot.*`-methodes zijn beschikbaar voor zowel `DataFrame`- als `Series`-objecten;
- Standaard zal iedere kolom geprojecteerd worden als een individueel element in de grafiek (lijn, boxplot,…);
- Iedere plot aangemaakt door `pandas` is een `Matplotlib`-object.

Gebruikshandleiding: een compleet overzicht over het plotten van data kan gevonden worden in de sectie over data visualisatie in de handleiding.