Pandas: Data lezen en schrijven

Dr. Cornelis Stal

April 29, 2022

[1]:
```
import pandas as pd
```

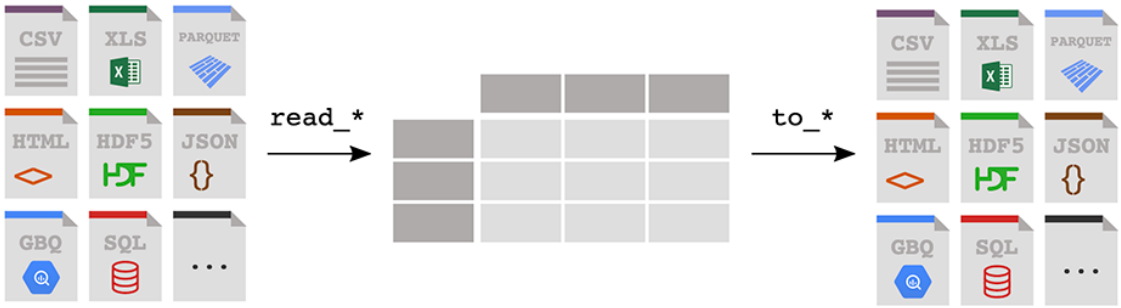
1 Data uit tabellen lezen en schrijven

Deze tutorial is een vertaling van de *Pandas Tutorial* op https://pandas.pydata.org/pandas-docs/stable/getting_started/.

Data: voor deze tutorial maken we gebruik van de combinatie van de Vlaamse gemeentegrenzen, waarvoor enkel de attributen geëxporteerd zijn uit de gehele dataset. De oorspronkelijke data zijn te vinden op geopunt. Aan iedere gemeente is eveneens het aantal inwoners toegekend, zoals beschikbaar gesteld door StatBel. De data bestaat uit de volgende kolommen:

- `OIDN`: versie identificator (geheel getal);
- `UIDN`: Identificator van de verschijningsvorm (geheel getal);
- `VERSDATUM`: Versiedatum van het object (staat overal op 01.01.1900, Datum);
- `TERRID`: Uniek volgnummer dat een Arrondissement identificeert (Geheel getal);
- `DATPUBLBS`: Datum van de publicatie in het Belgische Staatsblad (Datum);
- `NUMAC`: NUMAC-code van de publicatie in het Belgische Staatsblad (Geheel getal);
- `NISCODE`: Code die door het Nationaal Instituut voor Statistiek (NIS) aan een gemeente is toegekend (Tekst);
- `NAAM`: Vastgestelde naam van een gemeente volgens betreffende KB's (Tekst);
- `LENGTE`: Totale lengte (in m) van de omtreklijn(en) van de polygoon, berekend op basis van de geometrie van de polygoon (Decimaal getal);
- `OPPERVL`: Totale oppervlakte (in m2) van de polygoon, berekend op basis van de geometrie van de polygoon (Decimaal getal);
- `INWONERS`: Aantal inwoners volgens StatBel (Geheel getal);
- `OPPERVL_HA`: Totale oppervlakte (in hectare) van de polygoon, berekend op basis van de geometrie van de polygoon (Decimaal getal).

Klik hier hier om de data te downloaden.



Laten we starten met de data in te lezen:

[2]:
```
refgem = pd.read_csv("data/refgem.csv")
```

`pandas` beschikt over de `read_csv()`-function om data opgeslagen in een csv-bestand te importeren in een `pandas DataFrame`-object. `pandas` ondersteunt standaard verschillende andere bestandsformaten en databronnen, zoals csv, xlsx, sql, json, parquet, … Deze bestanden kunnen met de bijbehorende functies geïmporteerd worden, waarbij de naam van de methode telkens start met de prefix `read_*`.

Het valt sterk aan te bevelen om te controleren of de ingelezen data daadwerkelijk correct geïmporteerd zijn. Als we de `DataFrame` rechtstreeks aanspreken met de `head()`-methode, zullen de eerste 5 rijen uit de tabel standaard worden voorgesteld:

[3]:
```
refgem.head()
```

[3]:

	OIDN	UIDN	VERSDATUM	TERRID	DATPUBLBS	NUMAC	NISCODE	\
0	278	278	1900/01/01	1	1976/01/23	1.975123e+09	13001	
1	209	209	1900/01/01	2	1982/12/29	1.982002e+09	13031	
2	67	67	1900/01/01	3	1831/02/07	NaN	13025	
3	53	53	1900/01/01	4	1976/01/23	1.975123e+09	72037	
4	102	102	1900/01/01	5	1982/12/29	1.982002e+09	13036	

	NAAM	LENGTE	OPPERVL	INWONERS	OPPERVL_HA
0	Arendonk	32278.608018	5.501111e+07	13207	5501.111
1	Oud-Turnhout	33317.260657	3.917257e+07	14201	3917.257
2	Mol	77705.294504	1.144999e+08	37021	11449.991
3	Hamont-Achel	31929.641885	4.373537e+07	14337	4373.537
4	Retie	35339.133575	4.865207e+07	11582	4865.207

> **Opmerking:** `pandas` zal automatisch een identifier toekennen aan iedere rij. Zoals we gezien hebben met onder andere `list`-objecten in Python, start deze identifier met een waarde 0 en zal deze voor iedere rij met een waarde 1 toenemen.

Het aantal weer te geven rijen kunnen we aanpassen door dit gewenste aantal rijen mee te geven als attribuut aan de `head()`-methode:

[4]:
```
refgem.head(8)
```

[4]:

	OIDN	UIDN	VERSDATUM	TERRID	DATPUBLBS	NUMAC	NISCODE	\
0	278	278	1900/01/01	1	1976/01/23	1.975123e+09	13001	
1	209	209	1900/01/01	2	1982/12/29	1.982002e+09	13031	
2	67	67	1900/01/01	3	1831/02/07	NaN	13025	
3	53	53	1900/01/01	4	1976/01/23	1.975123e+09	72037	
4	102	102	1900/01/01	5	1982/12/29	1.982002e+09	13036	
5	77	318	1900/01/01	7	1843/10/09	NaN	72020	
6	251	251	1900/01/01	8	1831/02/07	NaN	13006	
7	293	293	1900/01/01	10	1982/07/17	1.982001e+09	72003	

	NAAM	LENGTE	OPPERVL	INWONERS	OPPERVL_HA
0	Arendonk	32278.608018	5.501111e+07	13207	5501.111
1	Oud-Turnhout	33317.260657	3.917257e+07	14201	3917.257
2	Mol	77705.294504	1.144999e+08	37021	11449.991
3	Hamont-Achel	31929.641885	4.373537e+07	14337	4373.537
4	Retie	35339.133575	4.865207e+07	11582	4865.207
5	Lommel	44857.532030	1.022498e+08	34255	10224.981
6	Dessel	27881.960842	2.749438e+07	9659	2749.438
7	Bocholt	41742.768306	5.927278e+07	13253	5927.278

De laatste 3 rijen kunnen we voorstellen met de `tail()`-methode. Deze zal zonder argumenten opnieuw 5 rijen teruggeven, geteld vanaf de laatste rij. De waarde `3` geven we als attribuut mee:

[5]:
```
refgem.tail(3)
```

[5]:

	OIDN	UIDN	VERSDATUM	TERRID	DATPUBLBS	NUMAC	NISCODE	\
297	315	315	1900/01/01	368	2018/06/01	2.018012e+09	44083	
298	311	311	1900/01/01	369	2018/06/01	2.018012e+09	12041	
299	312	312	1900/01/01	370	2018/06/01	2.018012e+09	44085	

	NAAM	LENGTE	OPPERVL	INWONERS	OPPERVL_HA
297	Deinze	82273.711221	1.280527e+08	43922	12805.275
298	Puurs-Sint-Amands	39898.949253	4.930680e+07	26208	4930.680
299	Lievegem	46791.029184	8.077192e+07	26441	8077.192

Het valt ook aan te bevelen om te controleren of **pandas** de data in de verschillende kolommen op de juiste manier geïnterterpreteerd heeft. We moeten met andere woorden nagaan of de juiste datatypes zijn toegekend aan de verschillende kolommen. Hiervoor gebruiken we de **dtypes** attribuut van de `DataFrame`:

[6]:
```
refgem.dtypes
```

[6]:
```
OIDN            int64
UIDN            int64
VERSDATUM      object
TERRID          int64
DATPUBLBS      object
NUMAC         float64
```

```
NISCODE         int64
NAAM           object
LENGTE        float64
OPPERVL       float64
INWONERS        int64
OPPERVL_HA    float64
dtype: object
```

Voor iedere kolom wordt het corresponderende datatype weergegeven. De datatypes in deze `DataFrame` bestaan uit gehele getallen ('integers', `int64`), decimale getallen ('floats', `float64`) en tekst ('strings', `object`).

> **Opmerking:** voor het opvragen van de `dtypes` worden geen ronde haakjes gebruikt. `dtypes` is een attribuut van een `DataFrame`- en `Series`-object. Attributen van een `DataFrame`- of `Series`-object vereisten geen ronde haakjes. Attributen representeren een bepaalde eigenschap van een `DataFrame`/`Series` terwijl voor een methode wel ronde haakjes vereist zijn. We hebben dit zojuist gedemonstreerd aan de hand van de `head()`- en `tail()`-methode.

Laten we de `DataFrame` nu eens exporteren naar een MS Excel bestand:

[9]:
```
refgem.to_excel("data/refgem.xlsx", sheet_name="refgem", index=False)
```

> **Opmerking:** indien er een foutmelding verschijnt (`ModuleNotFoundError: No module named 'openpyxl'`), dient een aanvullende bibliotheek geïnstalleerd te worden. Gebruik hiervoor:
>
> ```
> conda install -c anaconda openpyxl
> ```
>
> of
>
> ```
> pip install openpyxl
> ```

Waar we `read_*`-functies gebruiken om data in te lezen in een `pandas DataFrame`, gebruiken we `to_*`-methodes om `DataFrames` weg te schrijven naar een nieuw bestand. De `to_excel()`-methode staat de data op als een MS Excel-bestand. In bovenstaande voorbeeld wordt de naam van het tabblad (`sheet_name`) veranderd naar `refgem` in plaats van de standaard naam `Sheet1`. De rij-indices worden genegeerd dankzij de instelling `index=False`. Standaard worden de indices echter wel mee geëxporteerd.

De equivalente functie om een MS Excel-bestand in te lezen is de `read_excel()`-methode. Deze resulteert opnieuw in een nieuwe `DataFrame`. De instelling van `sheet_name` is voor dit voorbeeld niet vereist, maar geven we volledigheidshalve wel mee:

[ ]:
```
refgem = pd.read_excel("refgem.xlsx", sheet_name="refgem")
refgem.head()
```

Laten we nu eens kijken naar wat technische informatie over de geïmporteerde data:

[ ]:
```
refgem.info()
```

De `info()`-mothode geeft ons technische informatie over een `DataFrame`. Onderstaand overzicht geeft meer details over de interpretatie van deze output:

- We hebben inderdaad te maken met een `DataFrame`
- Er zijn 300 entiteiten aanwezig, dus 300 rijen
- Iedere tij heeft een rij-label (de `index`) met waarden varierend van 0 tto 299
- De tabel heeft 12 kolommen. De meeste kolommen beschikken voor iedere rij over een waarde (alle 300 waarden zijn `non-null`). In de kolom `NUMAC` ontbreken echter enkele waarden
- De kolommen `VERSDATUM`, `DATPUBLBS` en `NAAM` beschikken over tekst ('string'). De andere kolommen beschikken ofwel over gehele getallen ('integer') of over decimale getallen ('float')
- Er zijn 4 kolommen met het `dtype float64`, 5 kolommen van het `dtype integer64` en 3 kolommen van het `dtype object`
- De huidige `DataFrame` heeft ongeveer 28.2 Kb aan RAM-geheugen ingenomen.

1.1 Te onthouden:

- `pandas` ondersteunt verschillende bestandsformaten. Om data te importeren gebruiken we een van de verschillende `read_*`-functies;
- Het exporteren van een `DataFrame` wordt verzorgd door een van de verschillenden `to_*`-methodes;
- De `head()`-, `tail()`- en `info()`-methodes en de `dtypes` attribuut worden gebruikt om snel inzicht te krijgen in de geïmporteerde data

  Gebruikshandleiding: een meer uitgebreide uiteenzetting over de mogelijkheden om data vanuit `pandas` te importeren en te exporteren wordt gegeven in de sectie reader and writer functions.