Pandas: Subsets

Dr. Cornelis Stal

April 28, 2022

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

1 Het selecteren van data uit `DataFrames` ('subsets')

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 om de data te downloaden.

Ter voorbereiding importeren we opnieuw deze data:

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

1.1 Selecties maken

In de veronderstelling dat deze data opnieuw correct zijn ingeladen zullen we nog eens kijken naar de oppervlaktes in hectare van de gemeentes:

[ ]:
```
oppervlakte = refgem["OPPERVL_HA"]
oppervlakte.head()
```

Om één enkele kolom te selecteren gebruiken we vierkante haakjes (`[]`), met daarin de naam van de gewenste kolom.

Iedere individuele kolom in een `DataFrame`-object is een `Series`-object. Wanneer we dus één enkele kolom selecteren, zal het geretourneerde object van het type `pandas Series` zijn. We kunnen dit controleren door het datatype van de output op te vragen:

[ ]:
```
type(refgem["OPPERVL_HA"])
```

Laten we eens kijken naar de vorm ('shape') van dit object:

[ ]:
```
refgem["OPPERVL_HA"].shape
```

`DataFrame.shape` is een attribuut (herinner de tutorial over het lezen en schrijven van data, waarbij we het verschil tussen methodes en attributen toegelicht hebben) van een `pandas Series`- en `DataFrame`-object. Het resultaat van de `shape`-attribuut is een `tupple` met het aantal rijen en kolommen: (`nrows, ncolumns`). Een `pandas Series`-object is een 1-deimensioneel object, waardoor enkel het aantal rijen teruggegeven wordt.

Als we nu eens kijken naar zowel de oppervlakte in hectare en en aantal inwoners per gemeente.

[ ]:
```
oppervlakte_inwoners = refgem[["OPPERVL_HA", "INWONERS"]]
oppervlakte_inwoners.head()
```

Om meerdere kolommen te selecteren gebruiken we een lijst met kolomnamen binnen de vierkante haakjes `[]`.

> **Opmerking:** de vierkante haakjes aan de binnenkant van deze attribuut definieren een Pythin `list` met kolomnamen. De buitenste vierkante haakjes worden gebruikt om de data uit een `pandas DataFrame` te selecteren, zoals geïllustreerd in het vorige voorbeeld.

Het datatype van het geretourneerde object is nu een `DataFrame`:

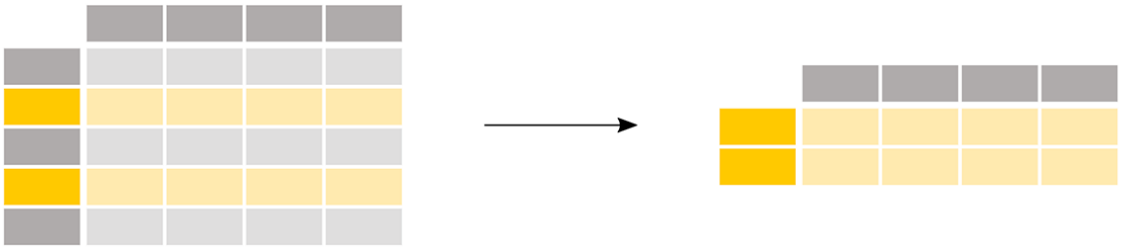
[ ]:
```
type(refgem[["OPPERVL_HA", "INWONERS"]])
```

[ ]:
```
refgem[["OPPERVL_HA", "INWONERS"]].shape
```

Deze selectie resulteert in een `DataFrame` met 300 rijen en 2 kolommen. with 891 rows and 2 columns. We moeten onthouden dat een `DataFrame` altijd een 2-deimensioneel object is met zowel een rij-dimensie en een kolom-dimensie.

> **Gebruikshandleiding:** voor meer informatie over data-indexering verwijzen we naar de sectie in de handleiding over 'indexing and selecting data'.

1.2 Data filteren uit specifieke rijen binnen een `DataFrame`



Laten we eens kijken naar alle Vlaamse gemeenten met meer van 50.000 inwoners:

[ ]:
```
above_50k = refgem[refgem["INWONERS"] > 50000]
above_50k
```

Om rijen te selecteren op basis van een conditionele expressie maken we gebruik van een gewenste voorwaarde binnen vierkante haakjes `[]`.

De conditionele expressie binnen de 'selectie-haakjes' `refgem["INWONERS"] > 50000` gaat na voor welke rijen het aantal in de kolom `INWONERS` groter is dan het getal `50000`:

[ ]:
```
refgem["INWONERS"] > 50000
```

Het resultaat van een confitionele expressie (`>`, maar `==`, `!=`, `<`, `<=`,... zouden ook werken) bestaat uit een `pandas Series`-object met booleaanse waarden (namelijk `True` (waar) or `False` (onwaar)). Het aantal elementen in dit `Series`-object is gelijk aan het aantal rijen in de oorspronkelijke `DataFrame`. Een dergelijke `Series` met booleaanse waarden kan worden gebruikt om de oorspronkelijke `DataFrame` te filteren door dit `Series`-object tussen vierkante 'selectiehaakjes' `[]` te plaatsen. Enkel de rijen waarvoor de corresponderende waarde gelijk is aan `True` zullen worden geselecteerd.

We wisten reeds dat de `DataFrame` met Vlaamse gemeenten beschikt over 300 entiteiten. Laten we nu eens kijken naar het aantal entiteiten waarvoor de conditie geldig is, dus waar het aantal inwoners groter is dan 50.000. We gebruiken hiervoor opnieuw de `shape`-attribuut, maar nu voor het `DataFrame`-object `above_50k`:

[ ]:
```
above_50k.shape
```

Vlaanderen heeft blijkbaar 12 gemeenten waarvan het aantal inwoners groter is dan 50.000.

We zoomen nu in op de gemeentes Gent, Merelbeke en Melle:

```
[ ]: refgem_genteo = refgem[refgem["NAAM"].isin(
         ['Gent', 'Merelbeke', 'Melle'])
     ]
     refgem_genteo.head()
```

Zoals bij de zojuist besproken conditionele expressie zal de `isin()`-conditionele functie een `Series`-object retourneren. Opnieuw omvat dit object de waarde `True` voor iedere rij waarvoor de waarde in de lijst overeenkomt met de waarde in de aangegeven kolom. Om de rijen te filteren op basis van een dergelijke functie wordt deze conditionele functie ook weer tussen vierkante 'selectiehaakjes' `[]` geplaatst. In bovenstaande voorbeeld `refgem["NAAM"].isin(['Gent', 'Merelbeke', 'Melle'])` worden alle rijen geselecteerd waarvoor het veld `NAAM` overeenkomt met `Gent`, `Merelbeke` of `Melle`.

Het resultaat van deze `isin()`-methode kunnen we ook verkrijgen door de 3 condisies te combineren met de `|` (of, 'or')-operator:

```
[ ]: refgem_genteo = refgem[(refgem["NAAM"] == 'Gent') |
         (refgem["NAAM"] == 'Merelbeke') |
         (refgem["NAAM"] == 'Melle')
     ]
     refgem_genteo.head()
```

We gaan nu eens kijken naar middelgrote steden waarbij het aantal inwoners gelegen is tussen 50.000 en 100.000:

```
[ ]: refgem_medium = refgem[
         refgem['INWONERS'].between(50000, 100000)
     ]
     refgem_medium.head()
```

In bovenstaande voorbeeld hebben we de `between()`-methode gebruikt om te onderzoeken voor welke entiteiten het veld `INWONERS` een waarde heeft die gelegen is tussen de 50000 inwoners en 100000 inwoners. Deze expressie kunnen we ook uitschrijven met de `&` (en, 'and') operator:

```
[ ]: refgem_medium = refgem[
         (refgem['INWONERS'] > 50000) &
         (refgem['INWONERS'] < 100000)
     ]
     refgem_medium.head()
```

> **Opmerking:** wanneer meerdere conditionele voorwaarde gecombineerd worden zal iedere conditie tussen ronde haakjes `()` geplaatst moeten worden. Daarnaast kunnen we geen gebruik maken van `or` en `and` expressies, maar gebruiken we voor or de operator `|` en voor `and` de operator `&`.
>
> **Gebruikshandleiding:** we verwijzen naar de secties in de handleiding voor meer informatie over 'boolean indexing en over de `isin()`-functie.

We gaan eens kijken voor welke gemeenten er een NUMAC-waarde aanwezig is:

```
[ ]: numac_no_na = refgem[refgem['NUMAC'].notna()]
     numac_no_na.head()
```

De notna()-conditionele functie geeft een waarde True terug voor iedere rij waarvoor de waarde van een bepaalde kolom niet gelijk is aan een Null-waarde. Deze functie kan gebruikt worden binnen vierkande 'selectiehaakjes' om data uit de tabel te filteren.

Omgekeerd kunnen we de isnan()-methode gebruiken om te achterhalen voor welke rijen een Null-waarde aanwezig is:

```
[ ]: numac_na = refgem[refgem["NUMAC"].isna()]
     numac_na.shape
```

Blijkbaar zijn er 31 gemeenten waarvoor geen NUMAC-waarde beschikbaar is.

> **Gebruikshandleiding:** voor meer informatie over functies die betrekking hebben op afwezige data ('missing values') verwijzen we naar de sectie in de handleiding over 'handling missing data'.

1.3 Specifieke rijen en kolommen selecteren uit een DataFrame

Zojuist hebben we het aantal inwoners verkregen van individuele gemeentes met behulp van refgem['INWONERS'] en hebben we alle rijen verkregen met gemeenten met meer dan 50.000 inwoners door middel van refgem[refgem["INWONERS"] > 50000]. Laten we nu eens enkel de gemeentenamen en bijbehorende aantal inwoners opvragen van alle gemeenten waarvan het aantal inwoners groter is dan 50.000:

```
[ ]: large_names = refgem.loc[
         refgem["INWONERS"] > 50000, ["NAAM", "INWONERS"]
     ]
     large_names
```

In dit geval wordt er een subset aangemaakt van zowel rijen als kolommen binnen één operatie. Het simpele gebruik van vierkante 'selectiehaakjes' [] volstaat niet meer voor een dergelijke operatie. De loc- en iloc-operatoren zijn in dit geval vereist voorafgaand aan de vierkante 'selectiehaakjes' []. Bij het gebruik van de loc- en iloc-operatoren correspondeert het gedeelte voor de komma met de gewenste rijen. Het deel achter de komma correspondeert met de gewenste kolom of kolommen.

De `loc`-operator schrijven we voorafgaand aan de vierkante 'selectiehaakjes' `[]` wanneer we gebruik maken van verwijzingen naar kolomnamen, verwijzingen naar rijlabels or conditionele expressies.

Zowel het gedeelte voor als na de komma kan bestaan uit slechts één label of lijst van labels, een deel van een lijst, een conditionele expressie of een dubbele punt `:`. De dubbele punt `:` geeft aan dat we alle rijen of kolommen wensen te selecteren. Als we slechts geïnteresseerd zijn in één kolom, volstaat het om slechts de kolomnaam mee te geven als attribuut.

We kunnen op deze manier kijken naar de rijen 10 tot 15 en de kolommen 7 tot 10:

[ ]:
```
refgem.iloc[9:14, 6:9]
```

Wanneer we geïnteresseerd zijn in specifieke rijen en/of kolommen in een `DataFrame` op basis van de specifieke positie binnen een tabel maken we gebruik van de `iloc`-operator, gevolgd door vierkante 'selectiehaakjes' `[]`. Opnieuw wordt een subset aangemaakt bestaande uit een aantal rijen en kolommen aangemaakt door middel van slechts één operatie. Ook hier volstaat het gebruik van vierkante 'selectiehaakjes' `[]` niet om een dergelijke operatie uit te voeren.

Selecties laten toe om nieuwe waarden toe te kennen aan een (verzameling) velden. In onderstaande voorbeeld zullen we bijvoorbeeld de `Null`-waarden voor de kolom `NUMAC` vervangen door de tekst `Niet beschikbaar`:

[ ]:
```
refgem.loc[refgem["NUMAC"].isna(), "NUMAC"] = "Niet beschikbaar"
refgem[refgem["NUMAC"] == "Niet beschikbaar"].head()
```

> **Gebruikshandleiding:** we verwijzen naar de sectie 'different choices' in de handleiding voor met informatie over het gebruik van `loc` and `iloc`.

1.4 Te onthouden:

- Voor het aanmaken van subsets aan de hand van selecties gebruiken we vierkante 'selectiehaakjes' `[]`;
- Binnen deze haakjes kunnen we melding maken van een enkele rij of kolom, maar ook een lijst van rijen/kolommen, een lijst van rij/kolomlabels, een deel van een label, een conditionele expressie of een dubbele punt;
- Met de `loc`-operator kunnen we specifieke rijen en kolommen selecteren aan de hand van de corresponderende namen;
- Met de `iloc`-operator kunnen we specifieke rijen en kolommen selecteren aan de hand van de posities binnen een tabel;
- Met de `loc`- en `iloc`-operator kunnen we nieuwe waarden toekennen aan velden.

> **Gebruikshandleiding:** voor een volledig overzicht over indexering verwijzen we naar de corresponderende sectie over 'indexing and selecting data' in de handleiding.