Pandas: Statistieken

Dr. Cornelis Stal

April 28, 2022

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

1 Statistieken berekenen

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.

Hier hier om de data te downloaden.

[ ]:
```
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()
```

1.1 Samenvattende statistieken

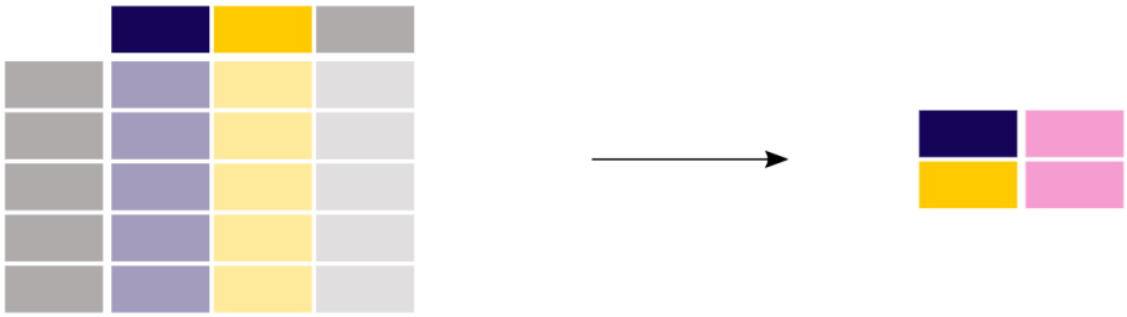
Met `pandas` kunnen eenvoudig statistieken berekenen, zoals de gemiddelde mediaanprijs van residentieel vastgoed in Vlaanderen:

```
[ ]: vastgoed["mediaan prijs(€)"].mean()
```

Bovenstaande waarde zou beter voorafgegaan worden door een `€`-teken, en afgerond op twee cijfers na de komma:

```
[ ]: median = vastgoed["mediaan prijs(€)"].mean()
     print('€%.2f' % median)
```

Verschillende statistieken kunnen worden berekend op kolommen met numerieke waarden. Over het algemeen genomen worden ontbrekende data genegeerd en worden operaties voor een hele rij.



Statistische operatoren kunnen tegelijkertijd uitgevoerd worden op meerdere kolommen:

```
[ ]: vastgoed[["aantal transacties", "mediaan prijs(€)"]].max()
```

Met bovenstaande code hebben we het maximaal aantal transacties, alsook de maximale mediaanprijs achterhaald uit twee kolommen uit de `DataFrame`. Hoewel de hier een lijst met kolomhoofdingen hebben meegegeven (zie de tutorial over subsets om te zien hoe dit functioneert). De statistieken worden echter berekend voor beide kolommen afzonderlijk.

Samenvattende statistieken kunnen dus berekend worden voor meerdere kolommen tegelijkertijd. Een interessante samenvatting uit de eerste tutorial wordt verkregen met de `describe()`-functie:

```
[ ]: vastgoed[["aantal transacties", "mediaan prijs(€)"]].describe()
```

In plaats van alle basisstatistieken te krijgen met de `describe()`-functie, kunnnen we ook zelf bepalen welke specifieke statistieken we willen terugkrijgen. Hiervoor definiëren we een `dict`-object, waarbij de sleutel ('key') verwijst naar een bepaalde kolomnaam. Als waarde ('value') krijgt de `dict` een lijst mee met vermelding van alle gewenste statistieken. Deze `dict` wordt tot slot als attribuut meegegeven aan de `DataFrame.agg()`-methode:

```
[ ]: vastgoed.agg({
         "aantal transacties": ["min", "max", "median", "mean"],
         "mediaan prijs(€)": ["min", "max", "median", "skew", "kurtosis"],
     })
```

> **Gebruikshandleiding:** details over omschrijvende statistieken worden gegeven in de gelijknamige sectie in de handleiding.

1.2 Samenvattende statistieken op basis van groepen

Het berekenen van statistieken op basis van groepen illustreren we aan de hand van de gemiddelde mediaanprijs van residentieel vastgoed in Vlaanderen per jaar:

[ ]:
```
vastgoed[["jaar", "mediaan prijs(€)"]].groupby("jaar").mean()
```

Vermits we geïnteresseerd zijn in de gemiddelde mediaanprijs per jaar selecteren we eerst de corresponderende twee kolommen uit de `DataFrame` via `vastgoed[["jaar", "mediaan prijs(€)"]]`. Vervolgens wordt op deze selectie de `groupby()`-methode uitgevoerd op de kolom `jaar`, waardoor groepen of categoriën ontstaan voor ieder uniek jaar. Het gemiddelde is tot slot berekend voor iedere categorie en als resultaat teruggegeven.

Het berekenen van eenn bepaalde statistiek (zoals het `mean()` mediaanprijs) voor iedere categorie in een kolom (zoals de waarden in de kolom `jaar`) is een veelgebruikte techniek. De `groupby`-methode wordt toegepast om dit soort berekeningen uit te voeren. Meer algemeen hebben we hier het `split-apply-combine`-patroon toegepast:

- **Split**: de data opsplitsen in groepen of categorieen
- **Apply**: een berekening toepassen op iedere afzonderlijke categorie
- **Combine**: het resultaat combineren in een nieuwe datastructuur

De 'apply'- (toepassen) en 'combine'-stappe worden normaal gesproken samen uitgevoerd binnen `pandas`.

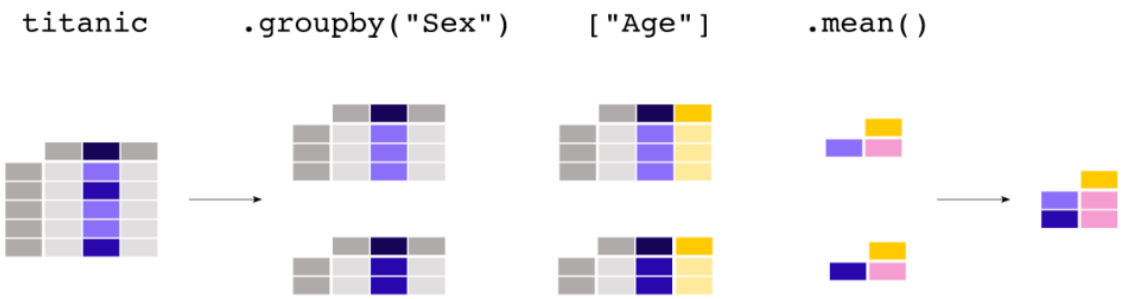
In het voorgaande voorbeeld hebben we eerst expliciet 2 kolommen geselecteerd. Hadden we deze selectie niet eerst niet uitgevoerd, dan zou de `mean()`-methode simpelweg op alle kolommen met nummerieke data toegepast:

[ ]:
```
vastgoed.groupby("jaar").mean()
```

Het is niet echt zinvol om de gemiddelde waarde te berekenen voor de kolom `refnis`.

Indien we opnieuw enkel geïnteresseerd zijn in de gemiddelde mediaanprijs per jaar, kunnnen we ook opnieuw kolommen selecteren met behulp van vierkante 'selectiehaakjes' (`[]`). Ook hiermee kunnen we numerieke waarden groeperen:

[ ]:
```
vastgoed.groupby("jaar")["mediaan prijs(€)"].mean()
```



Opmerking: De kolom `refnis` bevat numerieke data, maar betreft eigenlijk een unieke code die wordt toegewezen aan een bepaalde gemeente. Het berekenen van statistieken op deze kolom heeft dus weinig zijn. Hetzelfde zou in dit geval van toepassing zijn op de kolom `jaar`. `pandas` biedt de mogelijkheid om data om te zetten in expliciete `Categorical`-datatype, waardoor we op gepaste wijze met dergelijke data kunnen werken. Meer informatie hierover kan teruggevonden worden in de sectie over categorische data.

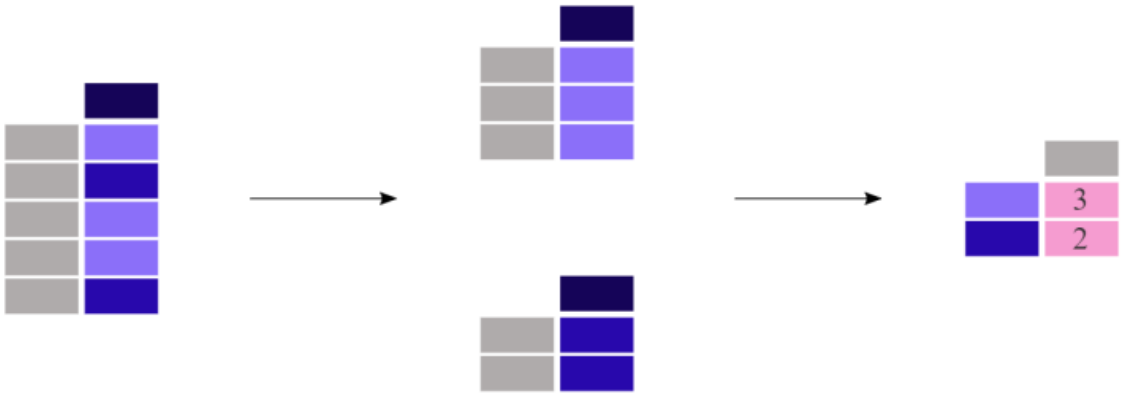
We kunnen data groeperen op basis van meerdere kolommen:

[ ]:
```
vastgoed.groupby(["lokaliteit", "jaar"])["mediaan prijs(€)"].mean()
```

In bovenstaand voorbeeld hebben we de `groupby()`-methode toegepast op de kolommen `lokaliteit` en `jaar`. We hebben hiervoor een lijst met kolomhoofden meegegeven als attribuut aan de `groupby()`-methode.

Gebruikshandleiding: een volledige omschrijving over het *split-apply-combine*-patroon kan teruggevonden worden in de sectie over groeperings operatoren in de handleiding.

1.3 CAantal rijen per categorie berekenen



Laten we eens kijken naar het aantal entiteiten binnen iedere individuele groep:

[ ]:
```
vastgoed["jaar"].value_counts()
```

De `value_counts()`-methode berekent het aantal elementen binnen iedere categorie in een kolom. Deze functie is feitelijk een combinatie van drie operaties, namelijk het selecteren van een een

kolom, het groeperen van alle unieke waarden binnen deze kolom. Tot slot het tellen van het aantal elementen binnen iedere groep:

[ ]:
```
vastgoed.groupby("jaar")["jaar"].count()
```

Opmerking: zowel de `size()`- als de `count()`-methode kunnen worden gecombineerd met binnen de `groupby()`-functie. In tegenstelling tot de `size()`-methode, waar `NaN` in rekening worden genomen en simpelweg het aantal rijen (de grootte van te tabel) wordt gegeven, zal de `count()`-methode ontbrekende data negeren. Bij de `value_counts()`-methode kan het `dropna`-argument gebruikt worden om `NaN`-waarden al dan niet in rekening te brengen.

Gebruikshandleiding: de handleiding beschikt over een (verouderde) versie over de `value_counts()`-methode. Meer bepaald verwijzen we hiervoor door naar de sectie over discretisatie.

1.4 Te onthouden:

- Samenvattende statistieken kunnen worden berekend voor gehele rijen en kolommen;
- De `groupby()`-methode introduceert de kracht van de *split-apply-combine* -benadering;
- De `value_counts()`-methode is een handige functie om het aantal elementen binnne iedere categorie te bepalen.

Gebruikshandleiding: een volledige omschrijving van de *split-apply-combine*-benadering wordt gepresenteerd in de sectie over operaties om data te groeperen in de handleiding.