Data Science 1

Extra Folien - Pandas GroupBy

Prof. Dr. Christian Bockermann

Hochschule Bochum

Wintersemester 2022/2023

DataFrames bieten Funktionen für Aggregate

Wir betrachten den folgenden `DataFrame`:

Nr	Kategorie	Einkäufe	Produkte
2	Young	1	3
1	MidAge	3	7
5	MidAge	2	5
3	Young	1	4
6	Old	2	6
4	Old	1	8

Mögliche Fragestellungen:

- Wie war die Gesamtzahl an Einkäufen?
- Wie ist die gesamte Anzahl verkaufter Produkte?

Einfaches Aggregat – die Summe: `sum`

```
customers = DataFrame(...) # siehe vorherige Folie

customers.sum()
```

`DataFrame.sum()` liefert ein `Series` Objekt zurück:

Nr	21
Kategorie	...
Einkäufe	10
Produkte	33

Die Zeilen enthalten den Spaltennamen und die Summe, die Zeile “Kategorie” enthält an der Stelle “...” die Werte als String

Häufige Fragestellung: Wie ist die Summe pro Kategorie?

Pandas DataFrames unterstützen Aufteilung/Gruppierung nach Spalten/Werten:

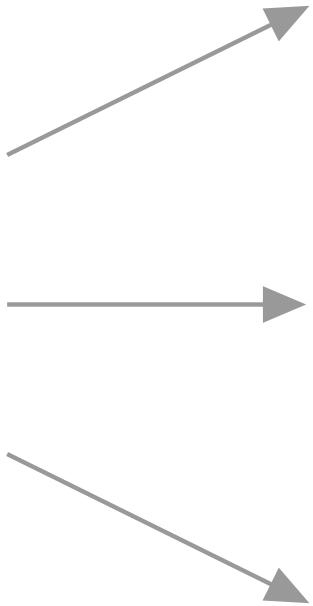
```
gruppiert = customers.groupby("Kategorie")

type(gruppiert)
# gruppiert ist vom Typ DataFrameGroupBy
```

Attribut `.groups` enthält die Gruppen und zugehörigen Zeilennummern:

```
print(gruppiert.groups)
#
# {'MidAge': [1, 2], 'Old': [4, 5], 'Young': [0, 3]}
```

Jede Gruppe repräsentiert einen eigenen DataFrame

Nr	Kategorie	Einkäufe	Produkte
2	Young	1	3
1	MidAge	3	7
5	MidAge	2	5
3	Young	1	4
6	Old	2	6
4	Old	1	8



Nr	Kategorie	Einkäufe	Produkte
2	Young	1	3
3	Young	1	4

Nr	Kategorie	Einkäufe	Produkte
1	MidAge	3	7
5	MidAge	2	5

Nr	Kategorie	Einkäufe	Produkte
6	Old	2	6
4	Old	1	8

Aggregate auf DataFrameGroupBy liefern DataFrame

Auf dem Grouping lassen sich Funktionen wie `sum()` auf alle Gruppen anwenden:

```
# Kategorie hat die Werte 'Yound', 'MidAge' und 'Old'
gruppiert = customers.groupby("Kategorie")
gruppiert.sum()
```

Das Ergebnis ist ein `DataFrame`:

	Nr	Einkäufe	Produkte
MidAge	**6**	**5**	**12**
Old	**10**	**3**	**14**
Young	**5**	**2**	**7**

Einfache Aggregat-Funktionen

Einfache Aggregate z.B. über `count()` und `mean()`

```
grps = customers.groupby("Kategory")
grps.count()
```

	Nr	Einkäufe	Produkte
MidAge	2	2	2
Old	2	2	2
Young	2	2	2

```
grps = customers.groupby("Kategory")
grps.mean()
```

	Nr	Einkäufe	Produkte
MidAge	3	2.500	6
Old	5	1.500	7
Young	2.500	1	3.500

Einfache Aggregat-Funktionen

Einfache Aggregate z.B. über `count()` und `mean()`

```
grps = customers.groupby("Kategory")
grps.count()
```

	Nr	Einkäufe	Produkte
MidAge	2	2	2
Old	2	2	2
Young	2	2	2

```
grps = customers.groupby("Kategory")
grps.mean()
```

	Nr	Einkäufe	Produkte
MidAge	3	2.500	6
Old	5	1.500	7
Young	2.500	1	3.500

`mean()` ist dabei durch `sum()/count()` definiert

Benutzerdefiniert Aggregatsfunktionen

Mit `apply()` lassen sich Funktionen auf alle Gruppen anwenden:

```
gruppen = customers.groupby("Kategorie")

# gruppen enthaelt jetzt quasi die
# Teil-DataFrames df0, df1, df2
#
# dfX.name ist jeweils der Wert aus "Kategorie"

gruppen.apply(lambda df: df.name[0:1])
```

Benutzerdefiniert Aggregatsfunktionen

Mit `apply()` lassen sich Funktionen auf alle Gruppen anwenden:

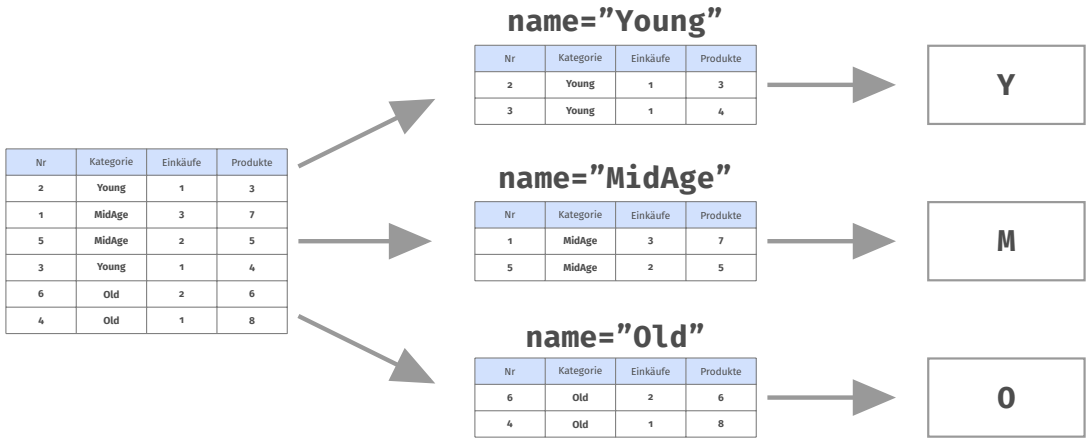
```
gruppen = customers.groupby("Kategorie")

# gruppen enthaelt jetzt quasi die
# Teil-DataFrames df0, df1, df2
#
# dfX.name ist jeweils der Wert aus "Kategorie"

gruppen.apply(lambda df: df.name[0:1])
```

Was passiert dabei?

Die Funktion

```
lambda df : df.name[0:1]
```

entnimmt aus einem DataFrame Namen den ersten Buchstaben

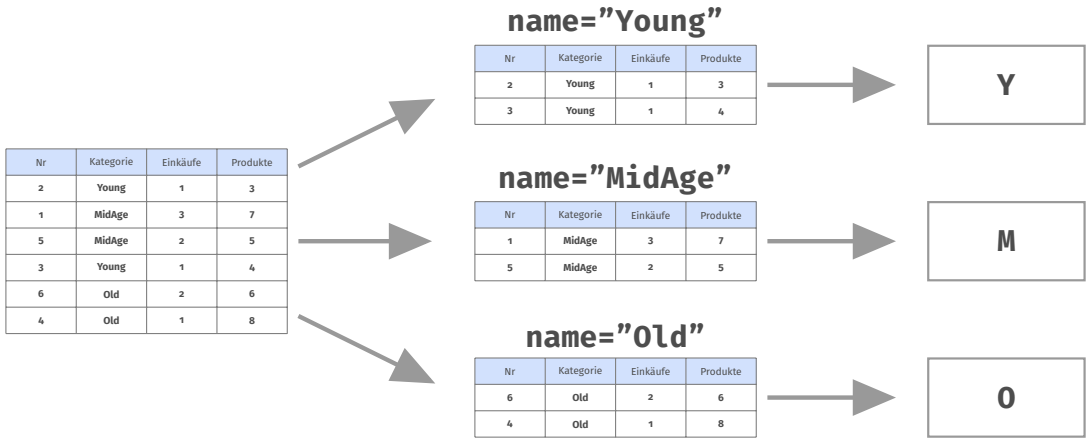


Nr	Kategorie	Einkäufe	Produkte
2	Young	1	3
1	MidAge	3	7
5	MidAge	2	5
3	Young	1	4
6	Old	2	6
4	Old	1	8

Die Funktion

```
lambda df : df.name[0:1]
```

entnimmt aus einem DataFrame Namen den ersten Buchstaben



Nr	Kategorie	Einkäufe	Produkte
2	Young	1	3
1	MidAge	3	7
5	MidAge	2	5
3	Young	1	4
6	Old	2	6
4	Old	1	8

Das es pro Gruppe nur ein Wert ist, macht Pandas daraus ein `Series` Objekt

Das Ergebnis

`Series` Objekt mit einem Index pro Gruppe und dem Wert aus der `lambda`-Funktion

MidAge	**M**
Old	**O**
Young	**Y**

Mehrdimensionale Aggregate

Betrachten wir die Funktion `myFun`:

```
def myFun(df):
  initial = df.name[0:1]  # erster Buchstabe
  size = len(df)          # Anzahl Zeilen

  row = [initial, size]   # Ergebnis-Zeile
  names = ['Initial', 'Groesse']

  return pd.Series(row, index=names)
```

Die Funktion berechnet für einen DataFrame df den ersten Buchstaben des Namens und die Größe
Das Ergebnis ist ein `Series` Objekt mit dem angegebenen Index

Mehrdimensionale Aggregate - `myFun`

Das Ergebnis ist ein DataFrame mit den berechneten `Series` Objekten als Zeilen:

	Initial	Groesse
MidAge	**M**	**2**
Old	**O**	**2**
Young	**Y**	**2**