DataCraft

4
Aggregation

kumulative Funktionen

DataCraft

- sum(), mean(), count() … sind Aggregatfunktionen, die zeilen- oder spaltenweise funktionieren
- **cumsum()** summiert alle Zeilen/Spalten bis zur aktuellen Zeile/Spalte. **cumprod()** multipliziert die Werte.
- **cummax()**, **cummin()** findet das Maximum bzw. Minimum bis zur aktuellen Spalte

Monat	Umsatz (in Mio. €)	kum. Umsatz (in Mio. €)	bester Umsatz YTD
Jan	2,34	2,34	2,34
Feb	2,50	4,84	2,50
Mrz	1,82	6,66	2,50
Apr	1,97	8,63	2,50
Mai	2,44	11,07	2,50
Jun	2,89	13,96	2,89
Jul	3,21	17,17	3,21
Aug	3,03	20,20	3,21
Sep	2,76	22,96	3,21
Okt	2,53	25,49	3,21
Nov	2,40	27,89	3,21
Dez	2,18	30,07	3,21

Die Funktion describe

Die Funktion **describe()** fasst wichtige Statistiken zusammen. Parameter include='all', um alle Spalten anzuzeigen.

Für Zahlen

count	Anzahl (nicht NaN)
mean	Mittelwert / Durchschnitt
std	Standardabweichung
min	Minimum
25%	25%-Quantil
50%	50%-Quantil / Median
75%	75%-Quantil
max	Maximum

Für alle anderen Datentypen

count	Anzahl (nicht NaN)
unique	eindeutige Werte
top	häufigster Wert
freq	Anzahl des häufigsten Werts

pandas-docs

deskriptive Statistik beschreibt die Verteilung eines Datensatzes

- **Mittelwert** = Durchschnitt = "Mitte der Daten"
- **Standardabweichung** = Streuung um den Mittelwert

x	y	z
1	1	1
2	1	2
3	1	1
4	1	2
5	1	1
6	1	2
7	1	1
8	100	2

mean(x) = 4.5
mean(y) = 13.36
mean(z) = 1.5

std(x) = 2.29
std(y) = 32.74
std(z) = 0.5

Exkurs: deskriptive Statistik - Quantile

DataCraft

- **Median** = Der Wert, bei dem die Hälfte der Daten kleiner und die andere Hälfte größer ist
- **25%-Quantil** = Der Wert, bei dem 25% der Daten kleiner und 75% größer sind
- **x%-Quantil** = Der Wert, bei dem x% der Daten kleiner und 1-x% größer sind
- Es gibt mehrere Möglichkeiten, wenn der Wert nicht enthalten ist. Z.B. Mittelwert der beiden mittleren Werte.

x	y	z
1	1	1
2	1	2
3	1	1
4	1	2
5	1	1
6	1	2
7	1	1
8	100	2

	x	y	z
Median	4.5	1	1.5
25%-Quantil	2.75	1	1
75%-Quantil	6.25	1	2

Aggregation

DataCraft

Eine der häufigsten Aufgaben in der Datenanalyse ist die Aggregation von Werten nach Gruppen, z.B. tägliche Kennzahlen je Produktkategorie

Date	Customer_ID	Transaction_ID	SKU_Category	SKU	Quantity	Sales_Amount
02/01/2016	2547	1	X52	0EM7L	1.0	3.13
02/01/2016	822	2	2ML	68BRQ	1.0	5.46
02/01/2016	3686	3	0H2	CZUZX	1.0	6.35
02/01/2016	3719	4	0H2	549KK	1.0	5.59
02/01/2016	9200	5	0H2	K8EHH	1.0	6.88
02/01/2016	5010	6	JPI	GVBRC	1.0	10.77
02/01/2016	1666	7	XG4	AHAE7	1.0	3.65
02/01/2016	1666	7	FEW	AHZNS	1.0	8.21
02/01/2016	1253	8	0H2	9STQJ	1.0	8.25
02/01/2016	5541	9	N5F	7IE9S	1.0	8.18

		Sales_Amount
Date	SKU_Category	
01/02/2016	01F	108.76
	0H2	71.92
	0KX	10.76
	0WT	16.18
	1EO	220.80

Die pandas-Funktionen groupby und agg

DataCraft

- Die Funktion **groupby()** bereitet die Gruppierung vor, so dass dann eine Aggregationsfunktion angewendet werden kann

```
transactions.groupby("Date")["Sales_Amount"].sum()
```

- Die Aggregation kann mehrere Gruppen umfassen

```
transactions.groupby(["Date","SKU_Category"])["Sales_Amount"].sum()
```

- Statt direkt eine Aggregationsfunktion aufzurufen, kann mit der Funktion **agg()** die Funktion als String übergeben werden

```
transactions.groupby(["Date","SKU_Category"])["Sales_Amount"].agg("sum")
```

```
transactions.groupby(["Date","SKU_Category"])["Sales_Amount"].agg(["sum"
,"min","max"])
```

- Aggregat-Funktionen: count, sum, mean, median, min, max, mode, std, var
- Die Gruppierung kann auch alleine durchgeführt werden. Dann wird ein DataFrameGroupBy Objekt zurückgegeben
- Einige Attribute und Funktionen
  - ngroups gibt die Anzahl Gruppen zurück
  - groups gibt die Gruppen mit den zugehörigen Indizes zurück
  - groups.keys() gibt die Bezeichnung der Gruppen zurück
  - get_group('Gruppenname') gibt den DataFrame, eingeschränkt auf die Gruppe, zurück

DataCraft

Daten können in zwei Formen vorliegen (+ Mischformen): Im Quer- oder Längsformat

	Eins	Zwei	Drei
Hamburg	0	1	2
Berlin	3	4	5

		Zahlen
Hamburg	Eins	0
Hamburg	Zwei	1
Hamburg	Drei	2
Berlin	Eins	3
Berlin	Zwei	4
Berlin	Drei	5

Multiindex

- Umwandlung eines DataFrames in eine Serie mit **stack()**. Umkehrung mit **unstack()**. Die Indizes sind entscheidend

- Durch Angabe des Indexnamen kann auch nur ein Teil umgewandelt werden: `unstack('Stadt')`

Zahlen

Stadt	Eins	Zwei	Drei
Hamburg	0	1	2
Berlin	3	4	5

Index

stack

unstack

Stadt	Zahlen	
Hamburg	Eins	0
Hamburg	Zwei	1
Hamburg	Drei	2
Berlin	Eins	3
Berlin	Zwei	4
Berlin	Drei	5

Multiindex

- Um Spalten statt Indizes zu verwenden, müsste der Index mit **reindex()** und **reset_index()** geeignet gesetzt werden

- Die Funktionen **pivot()** und **melt()** machen es etwas einfacher. Pivot wandelt einen Längs- in einen Querdatensatz um, melt macht das umgekehrte.

```
orders_wide = orders.pivot(index="Order",columns="Product",
                           values="Quantity")
orders = orders_wide.melt(id_vars = ["Order"],
                          value_name="Quantity")
```