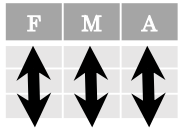
Data Wrangling mit pandas Merkblatt

http://pandas.pydata.org

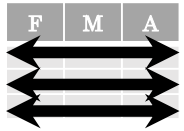
Ordentliche Daten (Tidy Data) – Eine Grundlage für Wrangling in pandas

In einem ordentlichen Datensatz:



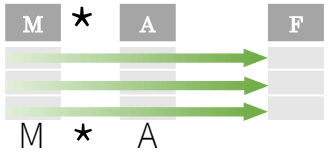
Jede **Variable** wird in einer eigenen **Spalte** gespeichert.

&



Jede **Beobachtung** wird in einer eigenen **Zeile** gespeichert.

Ordentliche Daten ergänzen pandas vektorisierte Operationen (vectorized operations). pandas speichert automatisch Beobachtungen, wenn du Variablen veränderst. Kein anderes Datenformat funktioniert mit pandas so intuitiv.



Syntax – DataFrames

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

```
df = pd.DataFrame(
       {"a" : [4 ,5, 6],
        "b" : [7, 8, 9],
        "c" : [10, 11, 12]},
      index = [1, 2, 3])
```

Spezifiziere Werte für jede Spalte.

```
df = pd.DataFrame(
    [[4, 7, 10],
     [5, 8, 11],
     [6, 9, 12]],
    index=[1, 2, 3],
    columns=['a', 'b', 'c'])
```

Spezifiziere Werte für jede Zeile.

		a	b	c
n	v			
d	1	4	7	10
	2	5	8	11
e	2	6	9	12

```
df = pd.DataFrame(
       {"a" : [4 ,5, 6],
        "b" : [7, 8, 9],
        "c" : [10, 11, 12]},   index =
pd.MultiIndex.from_tuples(
       [('d',1),('d',2),('e',2)],
         names=['n','v'])))
```

Erstelle DataFrame mit einem MultiIndex.

Verkettung von Methoden

Die meisten Methoden von pandas geben ein DataFrame zurück, damit auf das Ergebnis weitere Methoden von pandas angewendet werden können. Dies verbessert die Lesbarkeit des Codes.

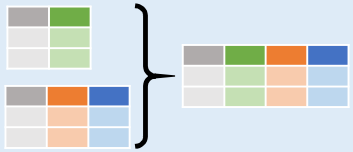
```
df = (pd.melt(df)
    .rename(columns={
          'variable' : 'var',
          'value' : 'val'})
    .query('val >= 200')
  )
```

Daten umformen (Reshaping) – Layout eines Datensatzes verändern

pd.melt(df)
Spalten in Zeilen zusammenfassen.

df.pivot(columns='var', values='val')
Zeilen zu Spalten erweitern.

pd.concat([df1,df2])
Zeilen eines DataFrames anhängen.



pd.concat([df1,df2], axis=1)
Spalten eines DataFrames anhängen.

df.sort_values('mpg')
Zeilen nach Werten einer Spalte ordnen (vom niedrigsten zum höchsten).

df.sort_values('mpg',ascending=False)
Zeilen nach Werten einer Spalte ordnen (vom höchsten zum niedrigsten).

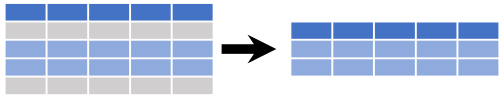
df.rename(columns={'y':'year'})
Spalten eines DataFrames umbenennen.

df.sort_index()
Index eines DataFrames sortieren.

df.reset_index()
Index eines DataFrames zurücksetzen auf Zeilennummern, Index zu Spalte machen.

df.drop(columns=['Length','Height'])
Spalten aus dem DataFrame weglassen.

Untermenge Beobachtungen (Zeilen)



df[df.Length > 7]
Zeilen, die logische Kriterien erfüllen, herausziehen.

df.drop_duplicates()
Doppelte Zeilen entfernen (nur Spalten beachten).

df.head(n)
Erste n Zeilen auswählen.

df.tail(n)
Letzte n Zeilen auswählen.

df.sample(frac=0.5)
Zufälligen Anteil der Zeilen auswählen.

df.sample(n=10)
Zufällig n Zeilen auswählen.

df.iloc[10:20]
Zeilen nach Position auswählen.

df.nlargest(n, 'value')
Oberste n Einträge auswählen und ordnen.

df.nsmallest(n, 'value')
Unterste n Einträge auswählen und sortieren.

Logik in Python (und pandas)			
<	Kleiner als	!=	Ungleich
>	Größer als	`df.column.isin(values)`	Zugehörigkeit zu einer Gruppe
==	Gleich	`pd.isnull(obj)`	Ist NaN
<=	Kleiner oder gleich	`pd.notnull(obj)`	Ist nicht NaN
>=	Größer oder gleich	`&,|,~,^,df.any(),df.all()`	Logisches und, oder, nicht, xor, ein, alle

Untermenge Variablen (Spalten)

df[['width','length','species']]
Mehrere Spalten mit bestimmten Namen auswählen.

df['width'] *or* **df.width**
Einzelne Spalte mit bestimmtem Namen auswählen.

df.filter(regex='*regex*')
Spalten, deren Namen regular expressions entsprechen, auswählen.

regex (Regular Expressions) Beispiele	
`'\.'`	Entspricht Strings, die einen Punkt "." enthalten.
`'Length$'`	Entspricht Strings, die mit dem Wort "Length" enden.
`'^Sepal'`	Entspricht Strings, die mit dem Wort "Sepal" beginnen.
`'^x[1-5]$'`	Entspricht Strings, die mit "x" beginnen und mit 1,2,3,4,5 enden.
`''^(?!Species$).*'`	Entspricht Strings, außer dem String "Species".

df.loc[:,'x2':'x4']
Alle Spalten zwischen x2 und x4 (inklusive) auswählen.

df.iloc[:,[1,2,5]]
Spalten an den Positionen 1,2 und 5 auswählen (erste Spalte ist 0).

df.loc[df['a'] > 10, ['a','c']]
Spalten, die logische Bedingung erfüllen, auswählen, nur bestimmte.

http://pandas.pydata.org/
Dieses cheat sheet wurde inspiriert vom Rstudio Data Wrangling Cheatsheet (https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf)
von Irv Lustig, Princeton Consultants
Für euch übersetzt und überarbeitet von: Datamics.com

Daten zusammenfassen

df['w'].value_counts()
Anzahl der Zeilen zählen mit eigenen Werten in allen Variablen.

len(df)
Anzahl der Zeilen im DataFrame.

df['w'].nunique()
Anzahl der eindeutigen Werte in einer Spalte.

df.describe()
Einfache beschreibende Statistik für jede Spalte (oder GroubBy).

pandas bietet eine große Anzahl von **zusammenfassenden Funktionen**, die auf verschiedenen pandas Objekten (DataFrame Spalten, Reihen, GroupBy, Expanding und Rolling (s.u.)) operieren und einzelne Werte für jede der Gruppen produzieren. Bei Anwendung auf ein DataFrame wird das Ergebnis als pandas Reihe für jede Spalte zurückgegeben.

Beispiele:

sum()
Summiert Werte jedes Objekts.

count()
Zählt alle nicht-NA/null Werte jedes Objekts.

median()
Median jedes Objekts.

quantile([0.25,0.75])
Quantile jedes Objekts

apply(*function*)
Funktion auf jedes Objekt anwenden.

min()
Kleinster Wert jedes Objekts.

max()
Größter Wert jedes Objekts.

mean()
Mittelwert jedes Objekts.

var()
Varianz jedes Objekts.

std()
Standardabweichung jedes Objekts.

Daten gruppieren

df.groupby(by="col")
Rückgabe eines GroupBy Objekts, gruppiert nach Werten in der Spalte "col"

df.groupby(level="ind")
Rückgabe eines GroupBy Objekts, gruppiert nach Werten des Indexlevels "ind".

Alle oben aufgeführten zusammenfassenden Funktionen können auf eine Gruppe angewendet warden. Zusätzliche GroupBy-Funktionen:

size()
Größe jeder Gruppe.

agg(*function*)
Gruppe anhand Funktion zusammenstellen.

Umgang mit fehlenden Daten

df.dropna()
Zeilen mit NA/null-Werten in einer Spalte weglassen.

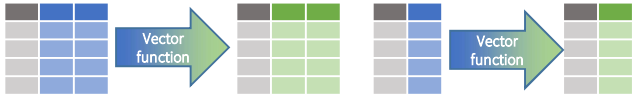
df.fillna(value)
Alle NA/null-Werte durch einen Wert ersetzen.

Neue Spalten erstellen

df.assign(Area=lambda df: df.Length*df.Height)
Berechnen und Hinzufügen einer oder mehrerer neuer Spalten.

df['Volume'] = df.Length*df.Height*df.Depth
Einzelne Spalte hinzufügen.

pd.qcut(df.col, n, labels=False)
Spalten in n Behälter (buckets) zusammenfassen.



pandas bietet eine große Anzahl an Vektorfunktionen, die auf allen Spalten eines DataFrames oder einer ausgewählten Spalte (einer pandas Reihe) operieren. Sie produzieren Vektoren von Werten für jede der Spalten oder eine einzige Reihe für die einzelnen Reihen.

Beispiele:

max(axis=1)
Elementweises Maximum.

clip(lower=-10,upper=10)
Werte an geg. Schranke kappen.

min(axis=1)
Elementweises Minimum.

abs()
Absolutwert.

Die Beispiele unten können ebenfalls auf Gruppen angewendet werden. In diesem Fall wird die Funktion auf die Basis der Gruppe angewendet und die zurückgegebenen Vektoren haben die Länge des ursprünglichen DataFrames.

shift(1)
Mit um 1 verschobenen Werten kopieren.

rank(method='dense')
Ohne Lücken ordnen.

rank(method='min')
Ordnen. Bei Gleichstand bekommen beide den niedrigeren Wert.

rank(pct=True)
Skaliert auf Intervall [0,1] ordnen.

rank(method='first')
Ordnen. Bei Gleichstand bekommen beide den ersten Wert.

shift(-1)
Mit um -1 verschobenen Werten kopieren.

cumsum()
Gesamte Summe.

cummax()
Gesamtes Maximum.

cummin()
Gesamtes Minimum.

cumprod()
Gesamtes Produkt.

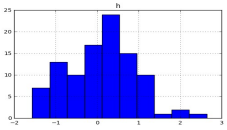
Fenster

df.expanding()
Gibt ein Expanding Objekt zurück, welches die Verwendung von zusammenfassenden Funktionen auf die gesamten Daten ermöglicht.

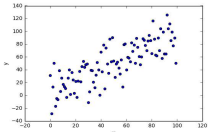
df.rolling(n)
Gibt ein Rolling Objekt zurück, welches die Verwendung von zusammenfassenden Funktionen auf Fenster der Länge n ermöglicht.

Plotting

df.plot.hist()
Histogramm für jede Spalte.



df.plot.scatter(x='w',y='h')
Streudiagramm mit Punktepaaren.



Datensätze kombinieren

adf

x1	x2
A	1
B	2
C	3

+

bdf

x1	x3
A	T
B	F
D	T

=

Standard Verknüpfungen

x1	x2	x3
A	1	T
B	2	F
C	3	NaN

pd.merge(adf, bdf, how='left', on='x1')
Passende Zeilen von bdf in adf einfügen.

x1	x2	x3
A	1.0	T
B	2.0	F
D	NaN	T

pd.merge(adf, bdf, how='right', on='x1')
Passende Zeilen von adf in bdf einfügen.

x1	x2	x3
A	1	T
B	2	F

pd.merge(adf, bdf, how='inner', on='x1')
Daten vereinen. Nur Zeilen behalten, in denen bei beiden Sätzen Werte stehen.

x1	x2	x3
A	1	T
B	2	F
C	3	NaN
D	NaN	T

pd.merge(adf, bdf, how='outer', on='x1')
Daten vereinen. Alle Werte in allen Zeilen behalten.

Verknüpfungen filtern

x1	x2
A	1
B	2

adf[adf.x1.isin(bdf.x1)]
Alle Zeilen in adf, die eine Entsprechung in bdf haben.

x1	x2
C	3

adf[~adf.x1.isin(bdf.x1)]
Alle Zeilen in adf, die keine Entsprechung in bdf haben.

ydf

x1	x2
A	1
B	2
C	3

+

zdf

x1	x2
B	2
C	3
D	4

=

Datensatz-ähnliche Operationen

x1	x2
B	2
C	3

pd.merge(ydf, zdf)
Zeilen, die in ydf und in zdf vorkommen (Schnittmenge).

x1	x2
A	1
B	2
C	3
D	4

pd.merge(ydf, zdf, how='outer')
Zeilen, die in ydf oder zdf vorkommen (Vereinigung).

x1	x2
A	1

pd.merge(ydf, zdf, how='outer', indicator=True)
.query('_merge == "left_only"')
.drop(columns=['_merge'])
Zeilen, die in ydf, aber nicht in zdf vorkommen (ydf ohne zdf; "Setdiff").

http://pandas.pydata.org/
Dieses cheat sheet wurde inspiriert vom Rstudio Data Wrangling Cheatsheet (https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf) von Irv Lustig, Princeton Consultants
Für euch übersetzt und überarbeitet von: Datamics.com