Long und Wide Format - Pivot und Melt

by Woche 14

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

In diesem Kapitel werden wir den Unterschied zwischen Daten im *Long* und *Wide* Format kennenlernen und wie wir zwischen den beiden Formaten hin und her konvertieren können.

Long und Wide Format

Von Anfang an muss klar sein, dass exakt dieselben Daten/Informationen im *Long* und *Wide* Format gespeichert werden können. Der Unterschied liegt ausschließlich in der Struktur der Daten. Am schnellsten kann dies anhand eines Beispiels verstanden werden. das folgende Bild zeigt die gleichen Daten im *Long* und *Wide* Format, wobei `Team` die Index-Spalte ist.

Wide Format

Team	Points	Assists	Rebounds
A	88	12	22
B	91	17	28
C	99	24	30
D	94	28	31

Long Format

Team	Variable	Value
A	Points	88
A	Assists	12
A	Rebounds	22
B	Points	91
B	Assists	17
B	Rebounds	28
C	Points	99
C	Assists	24
C	Rebounds	30
D	Points	94
D	Assists	28
D	Rebounds	31

Quelle: Statology

Das **Wide** Format ist das Format, in dem die meisten Menschen intuitiv ihre Daten speichern. In diesem Format sind die Spalten die Variablen und die Zeilen die Beobachtungen. Es gibt demnach mehr Spalten und die Tabelle ist im Vergleich breiter (*wide*).

Beim Verarbeiten von Daten ist es aber häufig so, dass wir die Daten im **Long** Format benötigen. In diesem Format sind die Variablen in einer Spalte und die Werte in einer anderen Spalte. Es demnach mehr Zeilen und die Tabelle ist im Vergleich länger (*long*).

Wir hatten beispielsweise in Kapitel 6.1 die Noten der beiden Personen im Long Format. Prinzipiell hätten wir die Daten ja auch im Wide Format speichern können, indem wir die Noten der beiden Personen in zwei Spalten speichern. Allerdings waren unsere Daten dann doch etwas untypisch, da es keine gemeinsame ID-Spalte gibt wie oben in der Abbildung “Team”. Solche Spalte wollen wir deshalb hier hinzufügen und sie

“Klassenarbeit” nennen. Außerdem nehmen wir drei Personen und dafür weniger Noten pro Person

```
df_noten_long = pd.DataFrame({
  'Klassenarbeit': ['Deutsch', 'Mathe', 'Geschichte', 'Deutsch', 'Mathe',
'Geschichte', 'Deutsch', 'Mathe', 'Geschichte'],
  'Person': ['A', 'A', 'A', 'B', 'B', 'B', 'C', 'C', 'C'],
  'Note': [2, 3, 3, 2, 2, 2, 3, 2, 1],
})

df_noten_wide = pd.DataFrame({
  'Klassenarbeit': ['Deutsch', 'Mathe', 'Geschichte'],
  'Note_A': [2, 3, 3],
  'Note_B': [2, 2, 2],
  'Note_C': [3, 2, 1],
})
```

```
df_noten_wide
```

```
  Klassenarbeit  Note_A  Note_B  Note_C
0       Deutsch       2       2       3
1         Mathe       3       2       2
2    Geschichte       3       2       1
```

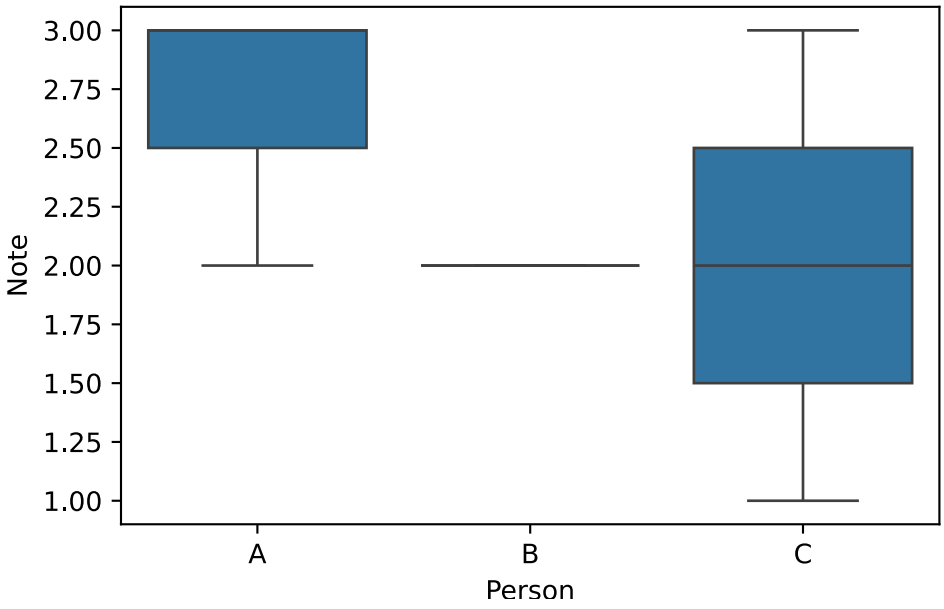
```
df_noten_long
```

```
  Klassenarbeit Person  Note
0       Deutsch      A     2
1         Mathe      A     3
2    Geschichte      A     3
3       Deutsch      B     2
4         Mathe      B     2
5    Geschichte      B     2
6       Deutsch      C     3
7         Mathe      C     2
8    Geschichte      C     1
```

Ein Beispiel wofür das Long Format besser geeignet ist, ist auch direkt das Erzeugen der uns bereits bekannten Abbildung:

```
# ???
```

```
plt.figure()
sns.boxplot(
  x='Person',
  y='Note',
  data=df_noten_long
)
plt.show()
```



Andererseits gibt es auch Situationen, in denen man das Wide Format benötigt. Es geht nicht so sehr darum welches Format besser ist, sondern darum, dass man in der Lage sein sollte schnell zwischen den beiden Formaten zu konvertieren.

Konvertierung

Die Konvertierung von Wide nach Long und umgekehrt wird in Pandas mit den Funktionen `pivot` und `melt` durchgeführt. Diese Begriffe sind auch in anderen Programmiersprachen bekannt.

Hier ist eine großartige Darstellung des Prinzips, allerdings muss der im Gif gezeigte Code ignoriert werden, da es R Code ist.

Melt: Wide zu Long

Mit `melt` konvertieren wir also Daten von Wide nach Long. Das bedeutet, dass wir - abgesehen von den Index-Spalten - eine beliebige Anzahl von Spalten in dann nur noch zwei Spalten konvertieren.

Die Funktion `melt` wird auf einem DataFrame aufgerufen und benötigt mindestens zwei Argumente: `id_vars` und `value_vars`. `id_vars` sind die Spalten, die als Index verwendet werden, also vor und nach der Konvertierung den einzelnen Wert idetifizieren. `value_vars` sind die Spalten, die in die Long-Form konvertiert werden, also nach der Konvertierung nur noch in einer Spalte vorliegen sollen.

Standardmäßig werden die neuen beiden Spalten `variable` und `value` genannt. Diese Namen können aber auch geändert werden, indem wir zusätzlich noch die zwei Argumente `var_name` und `value_name` übergeben. Wie die Namen schon andeuten, wird `var_name` dann statt `variable` und `value_name` statt `value` verwendet.

```
df_noten_wide.melt(
  id_vars='Klassenarbeit',
  value_vars=['Note_A', 'Note_B', 'Note_C']


)
```

```
  Klassenarbeit variable  value
0       Deutsch   Note_A      2
1         Mathe   Note_A      3
2    Geschichte   Note_A      3
3       Deutsch   Note_B      2
4         Mathe   Note_B      2
5    Geschichte   Note_B      2
6       Deutsch   Note_C      3
7         Mathe   Note_C      2
8    Geschichte   Note_C      1
```

```
df_noten_wide.melt(
  id_vars='Klassenarbeit',
  value_vars=['Note_A', 'Note_B', 'Note_C'],
  var_name='Person',
  value_name='Note'
)
```

```
  Klassenarbeit  Person  Note
0       Deutsch  Note_A     2
1         Mathe  Note_A     3
2    Geschichte  Note_A     3
3       Deutsch  Note_B     2
4         Mathe  Note_B     2
5    Geschichte  Note_B     2
6       Deutsch  Note_C     3
```

```
7         Mathe  Note_C     2
8    Geschichte  Note_C     1
```

Pivot: Long zu Wide

Mit `pivot` konvertieren wir also Daten von Long nach Wide. Das bedeutet, dass wir - abgesehen von den Index-Spalten - aus zwei Spalten in mehrere Spalten konvertieren.

Die Funktion `pivot` wird auf einem DataFrame aufgerufen und benötigt mindestens drei Argumente entgegen: `index`, `columns` und `values`. `index` ist die Spalte, die als Index verwendet wird, `columns` ist die Spalte, deren Inhalt dann die mehreren Spalten im Wide Format definiert und `values` ist die Spalte, die die zugehörigen Werte enthält.

```
df_noten_long.pivot(
  index='Klassenarbeit',
  columns='Person',
  values='Note'
)
```

```
Person         A  B  C
Klassenarbeit
Deutsch        2  2  3
Geschichte     3  2  1
Mathe          3  2  2
```

> **Anmerkung zu Standartoutput von `pivot`**
>
> Der oben gezeigte Ansatz mit `pivot` hat prinzipiell getan was er sollte. Allerdings kann man bei genauerem Hinsehen feststellen, dass die Spaltenüberschriften in mehreren Zeilen angezeigt werden. Das liegt daran, dass die Spaltenüberschriften ein MultiIndex sind. Das mag in manchen Situationen hilfreich sein, ist meines Erachtens aber dennoch eher ein unpraktisches Standardverhalten. Oft benötige ich die zum-Wide-Format-konvertierten Daten nämlich als ganz normalen DataFrame und hänge deshalb recht häufig `reset_index()` und `rename_axis(None, axis=1)` an, sodass man schlichtweg das erhält, was einige von euch hier vielleicht auch erwartet haben:
>
> ```
> df_noten_long.pivot(
>   index='Klassenarbeit',
>   columns='Person',
>   values='Note'
> ).reset_index().rename_axis(None, axis=1)
> ```
>
> ```
>   Klassenarbeit  A  B  C
> 0       Deutsch  2  2  3
> 1    Geschichte  3  2  1
> 2         Mathe  3  2  2
> ```
>
> Einfach ausgedrückt bewirtk `reset_index()`, dass der MultiIndex aufgelöst wird, sodass es wieder nur eine normale Index-Spalte gibt. Dann wird mit `rename_axis(None, axis=1)` bewirkt, dass die Spaltenüberschrift der Index-Spalte entfernt wird.

Es sei angemerkt, dass es noch die artverwandten Funktionen `stack` und `unstack` gibt, auf die hier - abgesehen von den *weiteren Ressourcen* unten - aber nicht genauer eingegangen wird.

Ein praktisches Beispiel

Um die Konvertierung von Wide nach Long und umgekehrt zu verdeutlichen, betrachten wir ein Beispiel mit Verkaufsdaten wie es auch in der Praxis vorkommen könnte. Wir haben Daten von vier Filialen und zwei Produkten, die in den Monaten Januar und Februar verkauft wurden und die Daten liegen in eingem ungünstigen Format vor. Das Zielformat ist direkt daneben gezeigt.

Ausgangsformat

```
df_start = pd.DataFrame({
  'Filiale': ['N', 'O', 'S', 'W'],
  'P_A_Jan': [150, 250, 200, 300],
  'P_A_Feb': [160, 260, 210, 310],
  'P_B_Jan': [100, 200, 150, 250],
  'P_B_Feb': [110, 210, 160, 260],
})

df_start
```

```
  Filiale  P_A_Jan  P_A_Feb  P_B_Jan  P_B_Feb
0       N      150      160      100      110
1       O      250      260      200      210
2       S      200      210      150      160
3       W      300      310      250      260
```

Zielformat

```
df_ziel = pd.DataFrame({
  'Filiale': np.repeat(['N', 'O', 'S', 'W'], 2),
  'Produkt': np.tile(['A', 'B'], 4),
  'Jan': [150, 100, 250, 200, 200, 150, 300, 250],
  'Feb': [160, 110, 260, 210, 210, 160, 310, 260]
})


df_ziel
```

```
  Filiale Produkt  Jan  Feb
0       N       A  150  160
1       N       B  100  110
2       O       A  250  260
3       O       B  200  210
4       S       A  200  210
5       S       B  150  160
6       W       A  300  310
7       W       B  250  260
```

Um vom Ausgangsformat zum Zielformat zu gelangen, können wir folgende Schritte durchführen: Zuerst konvertieren wir die Daten von Wide nach Long:

```
df1 = df_start.melt(
    id_vars='Filiale',
    value_vars=['P_A_Jan', 'P_A_Feb', 'P_B_Jan', 'P_B_Feb'],
    var_name='Produkt_Monat',
```

```
    value_name='Verkauf'
)

df1
```

```
   Filiale Produkt_Monat  Verkauf
0        N       P_A_Jan      150
1        O       P_A_Jan      250
2        S       P_A_Jan      200
3        W       P_A_Jan      300
4        N       P_A_Feb      160
5        O       P_A_Feb      260
6        S       P_A_Feb      210
7        W       P_A_Feb      310
8        N       P_B_Jan      100
9        O       P_B_Jan      200
10       S       P_B_Jan      150
11       W       P_B_Jan      250
12       N       P_B_Feb      110
13       O       P_B_Feb      210
14       S       P_B_Feb      160
15       W       P_B_Feb      260
```

Noch eleganter wäre es übrigens, wenn wir die Spalten für `value_vars=` automatisch extrahiert hätten. Spätestens wenn wir nicht nur von 2 Monaten und/oder 2 Produkten ausgehen, wird klar, dass man diese Liste nicht manuell erzeugen möchte. Um diese Spaltennamen automatisch zu extrahieren, hätten wir beispielsweise `produkt_spalten = [spalte for spalte in df_start.columns if spalte.startswith('P_')]` verwenden können und dann `value_vars=produkt_spalten` übergeben können.

Als nächstes müssen wir die Spalte `Produkt_Monat` trennen, sodass wir zwei separate Spalten `Produkt` und `Monat` vorliegen haben. Das können wir mit der Methode `str.split('_', expand=True)` durchführen, sodass das _ Symbol den Trennpunkt festlegt und `expand=True` dafür sorgt, dass erzeugten Spalten nicht in einer Liste mit je zwei Elementen, sondern als zwei separate Spalten vorliegen. Es gibt allerdings das Problem, dass ja alle Werte in der Spalte `Produkt_Monat` zwei Mal das _ Symbol enthalten. Da wir den Präfix `P_` aber sowieso nicht in den Werten haben wollen, können wir diesen einfach zuerst mit `str.removeprefix()`entfernen.

```
# Entferne Präfix in Produkt_Monat
df1['Produkt_Monat'] = df1['Produkt_Monat'].str.removeprefix('P_')
df1a = df1.copy()

# Trenne Produkt_Monat in zwei neue Spalten
```

```
df1[['Produkt', 'Monat']] = df1['Produkt_Monat'].str.split('_', expand=True)
df1b = df1.copy()

# Entferne Produkt_Monat
df1 = df1.drop(columns=['Produkt_Monat'])
```

```
# Nach Entfernen Präfix
df1a
```

```
   Filiale Produkt_Monat  Verkauf
0        N         A_Jan      150
1        O         A_Jan      250
2        S         A_Jan      200
3        W         A_Jan      300
4        N         A_Feb      160
5        O         A_Feb      260
6        S         A_Feb      210
7        W         A_Feb      310
8        N         B_Jan      100
9        O         B_Jan      200
10       S         B_Jan      150
11       W         B_Jan      250
12       N         B_Feb      110
13       O         B_Feb      210
14       S         B_Feb      160
15       W         B_Feb      260
```

```
# Nach Trennen in Produkt & Monat
df1b
```

```
   Filiale Produkt_Monat  Verkauf Produkt Monat
0        N         A_Jan      150       A   Jan
1        O         A_Jan      250       A   Jan
2        S         A_Jan      200       A   Jan
3        W         A_Jan      300       A   Jan
4        N         A_Feb      160       A   Feb
5        O         A_Feb      260       A   Feb
6        S         A_Feb      210       A   Feb
7        W         A_Feb      310       A   Feb
8        N         B_Jan      100       B   Jan
9        O         B_Jan      200       B   Jan
10       S         B_Jan      150       B   Jan
11       W         B_Jan      250       B   Jan
12       N         B_Feb      110       B   Feb
```

```
13       O         B_Feb      210       B   Feb
14       S         B_Feb      160       B   Feb
15       W         B_Feb      260       B   Feb
```

```
# Nach Entfernen von Produkt_Monat
df1
```

```
   Filiale  Verkauf Produkt Monat
0        N      150       A   Jan
1        O      250       A   Jan
2        S      200       A   Jan
3        W      300       A   Jan
4        N      160       A   Feb
5        O      260       A   Feb
6        S      210       A   Feb
7        W      310       A   Feb
8        N      100       B   Jan
9        O      200       B   Jan
10       S      150       B   Jan
11       W      250       B   Jan
12       N      110       B   Feb
13       O      210       B   Feb
14       S      160       B   Feb
15       W      260       B   Feb
```

Schließlich können wir nun die Daten von Long nach Wide konvertieren um das Zielformat zu erhalten. Streng genommen sind die Monatsspalten zwar noch nicht richtig (sondern wurde automatisch alphabetisch) sortiert, aber den Schritt sparen wir uns hier.

```
df1.pivot(
    index=['Filiale', 'Produkt'],
    columns='Monat',
    values='Verkauf'
).reset_index().rename_axis(None, axis=1)
```

```
  Filiale Produkt  Feb  Jan
0       N       A  160  150
1       N       B  110  100
2       O       A  260  250
3       O       B  210  200
4       S       A  210  200
5       S       B  160  150
6       W       A  310  300
7       W       B  260  250
```

Transponieren

Schließlich passt es in diesem Kapitel auch gut noch das Transponieren von DataFrames zu erwähnen. Transponieren ist einfacher und schneller erklärt als die Konvertierung zwischen Long und Wide, da lediglich die Zeilen und Spalten vertauscht werden. Das kann mit der Methode `.transpose()` durchgeführt werden. Hier ein Beispiel:

```
df_noten_wide
```

```
  Klassenarbeit  Note_A  Note_B  Note_C
0       Deutsch       2       2       3
1         Mathe       3       2       2
2    Geschichte       3       2       1
```

```
df_noten_wide.transpose()
```

```
                    0      1           2
Klassenarbeit  Deutsch  Mathe  Geschichte
Note_A               2      3           3
Note_B               2      2           2
Note_C               3      2           1
```

Transponieren funktioniert allerdings besser, wenn es eine “richtige” Index-Spalte gibt:

```
df_noten_wide.set_index('Klassenarbeit')
```

```
               Note_A  Note_B  Note_C
Klassenarbeit
Deutsch             2       2       3
Mathe               3       2       2
Geschichte          3       2       1
```

```
df_noten_wide.set_index('Klassenarbeit').transpose()
```

```
Klassenarbeit  Deutsch  Mathe  Geschichte
Note_A               2      3           3
Note_B               2      2           2
Note_C               3      2           1
```

Weitere Ressourcen

- Stack, Unstack, Melt, Pivot - Pandas
- How to Reshape Dataframes | Pivot, Stack, Melt and More

Übungen

Bringe folgenden DataFrame in ein Format mit Spalten für `Stadt`, `Jahr` und `Temperatur`. Konvertiere ihn danach wieder zurück in das Ursprungsformat.

```
df = pd.DataFrame({
    'Stadt': ['Berlin', 'München', 'Hamburg', 'Köln'],
    'Temp_2020': [10.5, 11.0, 9.5, 10.0],
    'Temp_2021': [11.2, 11.5, 10.0, 10.8]
})
```

- (A) Geschafft

Bringe diesen DataFrame…

```
df = pd.DataFrame({
    'Produkt': ['Produkt_A', 'Produkt_B'],
    'Region': ['Nord', 'Süd'],
    'Q1_2020': [150, 200],
    'Q2_2020': [180, 210],
    'Q1_2021': [160, 190],
    'Q2_2021': [170, 220]
})
```

…in dieses Format:

```
     Produkt Region  Jahr Quartal  Verkaufszahlen
0  Produkt_A   Nord  2020      Q1             150
1  Produkt_B    Süd  2020      Q1             200
2  Produkt_A   Nord  2020      Q2             180
3  Produkt_B    Süd  2020      Q2             210
4  Produkt_A   Nord  2021      Q1             160
5  Produkt_B    Süd  2021      Q1             190
6  Produkt_A   Nord  2021      Q2             170
7  Produkt_B    Süd  2021      Q2             220
```

- (A) Geschafft