Business Analytics

02 | Einführung in Python & Ausblick

Prof. Dr. Felix Zeidler | FH Bielefeld | SoSe 2023

Inhaltsverzeichnis

Lernziele der heutigen Veranstaltung

Nach der heutigen Veranstaltung sollten Sie in der Lage sein

- erste Schritte in Python zu machen, d.h. kleine Code-Beispiele selber zu programmieren
- einen ersten Überblick über die Möglichkeiten von Python zu bekommen
- erste Schritte in der Datenanalyse mit Python zu machen

(1) Kurze Einführung in Python

Grundlagen der Programmierung

Einige der wichtigsten Grundlagen der Programmierung werden in diesem Kapitel **wiederholt**.

- Einfache Arithmetik
- Variablen und Zuweisung
- Datentypen
- Fehlermeldungen
- Importieren von Modulen und Bibliotheken

> **Warum:** notwendige Grundlagen, um mit Python arbeiten zu können

Einfache Arithmetik

Python als Taschenrechner:

- einfache Berechnungen können mit Python durchgeführt werden
- bekannte arithmetische Operatoren können verwendet werden

`+` = Addition

`-` = Subtraktion

`*` = Multiplikation

`/` = Division

`**` = Potenzieren (**!** die Schreibweise für \(3^4\) in Python lautet `3**4`)

Einfache Arithmetik (cont'd)

Beispiel

```
1 # Beispiel für Berechnung in Python
2 1 + (5 * 12) / 3 - 12
```

```
9.0
```

Wie?:

- Eingabe des Ausdrucks in eine Jupyter-Zelle
- Zelle ausführen via
  - `STRG` + `ENTER` (führt Zelle aus) oder
  - `SHIFT` + `ENTER` (führt Zelle aus und springt in nächste Zelle) ausführen

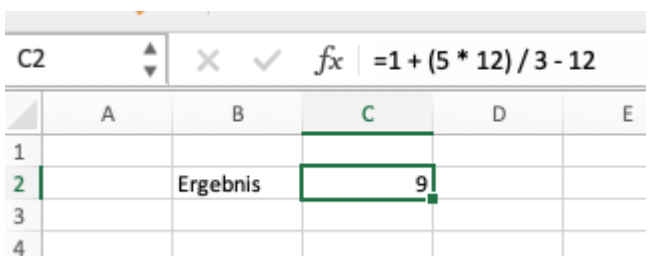
Variablen und Zuweisungen: Warum?

Warum Variablen?

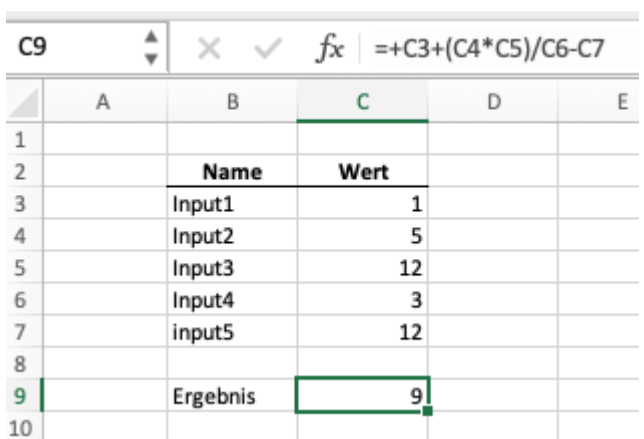
- Wiederverwendbarkeit: Variablen ermöglichen effizientes und modulares Programmieren.
- Lesbarkeit: Variablen verbessern die Verständlichkeit des Codes.

Analogie Excel

Schlechtes Beispiel



Besseres Beispiel



Variablen und Zuweisungen: Wie?

Wir können uns Variablen wie Schubladen vorstellen. Diese Schubladen beinhalten folgende Informationen:

- Name der Schublade (Variablenname)
- Inhalt der Schublade (Wert)
- Art der Schublade (Datentyp)

In Python werden Variablen mit dem =-Operator zugewiesen.

```
<variable> = <wert>
```

Variablen und Zuweisungen: wie? (cont'd)

Beispiele in Python:

```
1  zahl1 = 42
2  zahl2 = 3.2
3  satz = "Deutscher Meister wird nur der BVB"
```

Variablenhnamen und Werte

- müssen explizit angegeben (oder berechnet werden)
- `zahl1`, `zahl2` und `satz` sind die Variablennamen
- `42`, `3.2` und `"Deutscher Meister wird nur der BVB"` sind die Werte

Variablen und Zuweisungen: Datentyp(en)

Datentypen

- geben an, welche Art von Informationen in einer Variablen gespeichert wurde oder werden kann
- werden automatisch erkannt und müssen nicht explizit angegeben werden

Datentyp	Beispiel	Beschreibung
`int`	`42`	Ganze Zahl
`float`	`3.2`	Gleitkommazahl
`str`	`"Deutscher Meister wird nur der BVB"`	Text

Variablen und Zuweisungen: Datentyp(en)

- Wir können den Typ einer Variablen mit dem Befehl `type()` herausfinden.
- Wir können also z.B. den Typ der Variablen `zahl1` mit dem Befehl `type(zahl1)` herausfinden.

```
1  zahl1 = 42
2  type(zahl1)
```

```
int
```

Variablen und Zuweisungen: Datentyp(en)

Beispiele weiterer Datentypen in Python

```python
# Boolean
wahr = True
falsch = False

# Liste
liste = [1, 2, 3, 4, 5]

# Dictionary
dictionary = {"a": 1, "b": 2, "c": 3}
```

Variablen und Zuweisungen: arbeiten mit Variablen

Wir können nun das Konzept der Variablenzuweisung verwenden, um Berechnungen durchzuführen.

```
1 cash_flow = 100
2 zinssatz = 0.05
3 kapitalwert = cash_flow / (1 + zinssatz)
4 kapitalwert
```

```
95.23809523809524
```

Aufgabe 1

Ein Unternehmen hat ein Kapital von 10.000 Euro auf einem Sparkonto angelegt. Die Bank zahlt einen jährlichen Zinssatz von 3,5%. Berechne den Zinsertrag nach einem Jahr und das Endguthaben nach einem Jahr.

Anweisungen:

- Erstelle Variablen für das Startkapital, den Zinssatz und die Anzahl der Jahre.
- Berechne den Zinsertrag nach einem Jahr.
- Berechne das Endguthaben nach einem Jahr.
- Gib die Ergebnisse mit aussagekräftigen Texten aus (z.B. “Der Zinsertrag nach einem Jahr beträgt 350 Euro.”)
  (Hinweis: nutzen Sie die Funktion `print()`)

Aufgabe 1: Lösung

▶ Code

Fähigkeiten von Variablen abhängig von Datentyp

Warum ist Datentyp wichtig?

- jeder Datentyp hat unterschiedliche Fähigkeiten
- zwei Gründe, weshalb wir Fähigkeiten einer Variable kennen sollten:
  1. Wir sollten die Fähigkeiten einer Variable kennen, um **Fehler zu vermeiden**.
  2. Wir können die Fähigkeiten einer Variable nutzen, **um Aufgaben zu lösen**.

Beispiel: Fehler vermeiden

Berechnung mit Zahlen

```
1 zahl1 = 2
2 zahl2 = 3
3 zahl1 + zahl2
```

```
5
```

Berechnung mit Text

```
1 text1 = "2"
2 text2 = "3"
3 text1 + text2
```

```
'23'
```

Wichtig: Wissen über Datentypen ist wichtig, um fehlerhafte Berechnungen/Programme zu vermeiden

Beispiel: Fähigkeiten nutzen

- Datentyp `str` kann Operation `+` nutzen
- Aber: nicht im mathematischen Sinne, sondern um Texte zu verbinden

```
1 vorname = "Max"
2 nachname = "Mustermann"
3 anrede = "Lieber Herr " + vorname + " " + nachname
4 anrede
```

```
'Lieber Herr Max Mustermann'
```

Wichtig: Wissen über Datentypen ist wichtig, um spezifische Fähigkeiten der Datentypen nutzen zu können
- `help()`-Funktion kann helfen, die Fähigkeiten einer Variable herauszufinden
- `dir()`-Funktion kann helfen, die Fähigkeiten einer Variable herauszufinden

Fehlermeldungen

- Fehlermeldungen werden automatisch generiert, wenn der Code nicht ausgeführt werden kann
- Fehlermeldungen sind eine wichtige Quelle von Informationen, um Fehler zu finden und zu beheben

Beispiel:

```
1 vorname = "Max"
2 nachname = "Mustermann"
3 anrede = "Lieber Herr " vorname + " " + nachname
4 anrede
```

Fehlermeldungen: typische Fehlermeldungen

Fehlermeldung	Beschreibung der Art des Fehlers	kurzes Beispiel
SyntaxError: EOL while scanning string literal	Fehler bei der Syntax, in dem ein String nicht richtig abgeschlossen wurde.	“Max Muster
NameError: name ‘y’ is not defined	Fehler bei der Verwendung einer Variable, die nicht definiert wurde.	x = y + 2
TypeError: unsupported operand type(s) for +: ‘int’ and ‘str’	Fehler beim Verwenden eines Operators mit ungültigen Typen.	2 + “Text”
IndexError: list index out of range	Fehler beim Zugriff auf ein Element einer Liste, das nicht existiert.	meine_liste[5]
KeyError: ‘key’	Fehler beim Zugriff auf einen Schlüssel in einem Dictionary, der nicht existiert.	mein_dictionary[‘alter’]

Module und Bibliotheken: warum?

Warum brauchen wir Module und Bibliotheken?

- Umfang von Python ist begrenzt
- Umfang von Python kann durch **Module** und **Bibliotheken** erweitert werden
- Module und Bibliotheken sind **Python-Dateien** mit zusätzlichen Funktionen
- wir können uns diese wie Add-ins vorstellen, die wir in Excel installieren können

Importieren von Modulen und Bibliotheken

Wie können wir Module und Bibliotheken nutzen?

- Module und Bibliotheken müssen zunächst **installiert** werden
  (Hinweis: die von uns genutzten Module sollten bereits via Anaconda installiert sein)
- sind die Module und Bibliotheken installiert, können wir sie **importieren**

Der Import von Modulen erfolgt typischerweise über drei Varianten:

1. Importieren des gesamten Moduls
2. Importieren einer Funktion aus einem Modul
3. Importieren des gesamten Moduls unter einem anderen Namen

Importieren des gesamten Moduls

Beispiel 1: Importieren des gesamten Moduls

```
1 import statistics
2
3 daten = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
4 avg = statistics.mean(daten)
5 avg
```

Wir haben hier das gesamte Modul importiert und können nun **alle** Fähigkeiten, die dieses Modul zur Verfügung stellt, nutzen. In diesem Fall haben wir die Funktion `mean()` aus dem Modul `statistics` genutzt, um den Mittelwert der Daten zu berechnen.

Importieren einzelner Fähigkeiten aus einem Modul

Beispiel 2: Importieren einer Funktion aus einem Modul

```
1 from statistics import mean
2
3 daten = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
4 avg = mean(daten)
5 avg
```

> Wir haben hier nur die Funktion `mean()` aus dem Modul `statistics` importiert. Dies hat den Vorteil, dass wir nicht mehr den Namen des Moduls angeben müssen, wenn wir die Funktion nutzen wollen. Jedoch können wir auch **nur** die Funktion `mean()` aus dem Modul `statistics` nutzen, alle anderen Funktionen sind nicht verfügbar.

Importieren eines gesamten Moduls unter einem anderen Namen

Beispiel 3: Importieren des gesamten Moduls unter einem anderen Namen

```
1 import statistics as stat
2
3 daten = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
4 avg = stat.mean(daten)
5 avg
```

Wir haben hier das gesamte Modul `statistics` importiert und ihm den Namen `stat` gegeben. Dies hat den Vorteil, dass wir nicht mehr den Namen des Moduls angeben müssen, wenn wir die Funktion nutzen wollen. Jedoch können wir auch **alle** Funktionen aus dem Modul `statistics` nutzen, auch wenn wir diese nicht explizit importiert haben.

Aufgabe

- Importieren Sie die `pandas`-Bibliothek
- Lesen Sie die Datei `sales.xlsx` ein und zeigen Sie das daraus resultierende Dataframe an (Hinweis: `.read_...`-Funktionen sind in der `pandas`-Bibliothek enthalten)
- Ermitteln Sie die Anzahl der Zeilen im Dataframe (Hinweis: checken Sie die Fähigkeiten des Datentypen `DataFrame`)

Aufgabe

Lösung - Einlesen und anzeigen:

▶ Code

Lösung - Anzahl der Zeilen:

▶ Code

(2) Fallstudie: Fashion Avenue GmbH

Zweck der Fallstudie

Die Fallstudie soll Ihnen einen Eindruck davon vermitteln, was Sie in diesem Kurs erwartet. Diese Fallstudie verfolgt somit zwei Ziele:

1. Anwendungsbeispiele für die einzelnen Schritte des Analyseprozesses geben und
2. Programmierbeispiele für die einzelnen Schritte des Analyseprozesses vorzustellen.

Hinweis

- In dieser Fallstudie werden Dinge vorgegriffen, die Sie noch nicht gelernt haben.
- Ziel der Fallstudie ist es ausdrücklich **nicht**, dass Sie jede Code-Zeile nachvollziehen können.
- Vielmehr sollen Sie einen Eindruck davon vermittelt bekommen, was Sie in diesem Kurs erwartet bzw. was Sie am Ende in der Lage sind selber umzusetzen.

Fallstudie

Ausgangssituation

Situationsbeschreibung:

- Fashion Avenue GmbH: Einzelhandelsunternehmen in Bielefeld, spezialisiert auf Bekleidung und Accessoires für Frauen, Männer und Kinder.
- Gründerin Anna Meyer: Erfahrene Einzelhändlerin, die hochwertige und modische Produkte zu erschwinglichen Preisen anbietet.
- Engagiertes Verkaufsteam: Kombination aus erfahrenen Verkäufern und motivierten, jungen Mitarbeitern, bekannt für hohe Kundenorientierung und ausgezeichneten Service.
- Aktuelle Herausforderung: Rückläufige Verkäufe

> **Aufgabe:**
>
> Datensammlung, -aufbereitung und -analyse zur Ermittlung der Ursachen und Entwicklung von Lösungsansätzen.

Fallstudie

Strukturierung der Analyse entlang des Analyseprozesses

Der Analyseprozess besteht aus folgenden Schritten, die wir im Folgenden durchlaufen werden:

1. Problemstellung
2. Einlesen der Daten
3. Aufbereitung der Daten
4. Transformation der Daten
5. Visualisierung der Daten
6. Modellierung der Daten
7. Kommunikation der Ergebnisse

Fallstudie

Schritt 1: Problemstellung

Was ist die konkrete Problemstellung?

> **Was ist der Grund für den Rückgang des Umsatzes?**

Schritt 2: Einlesen der Daten

Grundsätzlich:

- Datenbedarfsanalyse: Identifizierung relevanter Variablen.
- Datenverfügbarkeit prüfen: Intern, extern, generieren oder erheben.
- Voranalysen für effiziente und zielführende Datenanalyse.

Unsere Fallstudie:

- Transaktionsdaten aus ERP-System, CSV-Datei (`transactions_fashion_avenue.csv`)
- Nutzung von Pandas: Einlesen, Verarbeiten, Manipulieren von Daten in Python.

Schritt 2: Einlesen der Daten (cont'd)

Ablauf:

- importieren der Bibliothek `pandas` mit dem Namen `pd`
- Wir lesen die Daten nun mit der Funktion `read_csv()`[1] ein und weisen die Daten der Variable `df` zu.
- Danach geben wir - mit der Funktion `head()`- die ersten fünf Zeilen der Tabelle aus, um einen ersten Überblick über die Daten zu erhalten.

```
1 # Importieren der Bibliothek pandas
2 import pandas as pd
3
4 # Einlesen der Daten (hier: csv-Datei)
5 df = pd.read_csv("_data/transactions_fashion_avenue.csv")
6
7 # Ausgabe der ersten fünf Zeilen der Tabelle
8 df.head()
```

1. Hinweis: die Datei ist in unserem Fall im Ordner `_data`

Schritt 2: Einlesen der Daten (cont'd)

Auszug aus den Daten:

	Kundenname	Alter	Zahlungsmethode	ø Preis	Menge	Datum	Uhrzeit
0	Gislinde Börner	29	EC-Karte	14.99	4	01.01.2022	10:24:44
1	Pierre Ullmann	28	Bar	17.99	4	01.01.2022	10:55:09
2	Rainer Birnbaum	57	Bar	43.99	4	01.01.2022	11:25:31
3	Ekaterina Binner	47	Kreditkarte	36.99	3	01.01.2022	11:30:28
4	Prof. Mandy Riehl	59	Bar	35.99	4	01.01.2022	11:46:50
...	...	...	...	...	...	...	...
9995	Dr. Randolf Renner B.Sc.	27	Kreditkarte	24.99	4	30.12.2022	18:36:02
9996	Harry Davids	27	Kreditkarte	14.99	3	30.12.2022	18:41:31
9997	Univ.Prof. Gunhild Hornich B.A.	28	Kreditkarte	16.99	4	30.12.2022	19:19:53
9998	Christof Wernecke	19	Bar	21.99	3	30.12.2022	19:21:06
9999	Prof. Josephine Stiebitz	25	EC-Karte	12.99	6	31.12.2022	12:38:00

10000 rows × 7 columns

Fallstudie

Schritt 3: Aufbereitung der Daten

Grundsätzlich:

- das Entfernen von fehlenden Werten,
- das Korrigieren von offensichtlich fehlerhaften Werte,
- das Umwandeln von Datentypen,
- das Umbennen von Spalten etc.

Unsere Fallstudie:

- Was fällt ihnen auf?
- Was müsste man noch tun, um die Daten für die weitere Analyse aufzubereiten?

Schritt 3: Feststellen, ob etwas fehlt

- wir verwenden die Funktion `info()`
- Funktion gibt Information über z.B.
  - die Anzahl der Zeilen,
  - die Anzahl der Spalten,
  - den Datentyp der Spalten,
  - die Anzahl der nicht-leeren Werte etc.

Schritt 3: Feststellen, ob etwas fehlt

```
1 # Ausgabe von Informationen über die Tabelle
2 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Kundenname       10000 non-null  object
 1   Alter            10000 non-null  int64
 2   Zahlungsmethode  10000 non-null  object
 3   ø Preis          10000 non-null  float64
 4   Menge            10000 non-null  int64
 5   Datum            10000 non-null  object
 6   Uhrzeit          10000 non-null  object
dtypes: float64(1), int64(2), object(4)
memory usage: 547.0+ KB
```

Schritt 3: Aufbereitung der Daten

Was ändern wir?

- Umbenennen der Spalte `ø Preis`
- Ändern des Datentyps der Spalte `Datum`

Python-Code:

```python
# Umbenennen der Spalte "ø Preis" in "Preis"
df = df.rename(columns={"ø Preis": "Preis"})

# Ändern des Datentyps der Spalten "Datum" und "Uhrzeit"
df = df.astype({"Datum": "datetime64[ns]"})
```

Schritt 3: Aufbereitung der Daten (cont'd)

Wir können nun nochmals die Funktion `info()` verwenden, um zu überprüfen, ob die Änderungen korrekt durchgeführt wurden.

```
1 # Ausgabe von Informationen über die Tabelle
2 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Kundenname       10000 non-null  object
 1   Alter            10000 non-null  int64
 2   Zahlungsmethode  10000 non-null  object
 3   Preis            10000 non-null  float64
 4   Menge            10000 non-null  int64
 5   Datum            10000 non-null  datetime64[ns]
 6   Uhrzeit          10000 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(2), object(3)
memory usage: 547.0+ KB
```

Schritt 4 und 5: Aufbereiteter Datensatz

```
# Ausgabe der ersten 5 Zeilen
df.head()
```

	Kundenname	Alter	Zahlungsmethode	Preis	Menge	Datum	Uhrzeit
0	Gislinde Börner	29	EC-Karte	14.99	4	2022-01-01	10:24:44
1	Pierre Ullmann	28	Bar	17.99	4	2022-01-01	10:55:09
2	Rainer Birnbaum	57	Bar	43.99	4	2022-01-01	11:25:31
3	Ekaterina Binner	47	Kreditkarte	36.99	3	2022-01-01	11:30:28
4	Prof. Mandy Riehl	59	Bar	35.99	4	2022-01-01	11:46:50

Fallstudie

Schritt 4 und 5: Erkenntnisse aus den Daten gewinnen

Explorative Analyse:

- Die Schritte 4 (Transformieren) und 5 (Visualisieren) bezeichnet man auch als explorative Analyse.[1]
- Hypothesen-basiertes Vorgehen, welches mittels **einfacher statistischer Methoden** (z.B. Durchschnitte über Gruppen bilden) überprüft wird
- Vorgehen ist oft iterativ

konkrete Nächste Schritte: was sollten ihrer Meinung nach nächsten Schritte sein?

- Ausgangslage validieren / überprüfen: gibt es überhaupt einen Rückgang des Umsatzes?
- die Ursache für den Umsatzrückgang finden.

1. Hinweis: den Schritt der Modellierung lassen wir an dieser Stelle noch aus

Schritt 4 und 5: Transformieren der Daten

Transformieren der Daten:

- Spalte `Umsatz` erzeugen, da Umsatz nicht explizit im Datensatz.
- Datumsinformationen (Woche, Monat, Tag etc.) generieren

Python-Code:

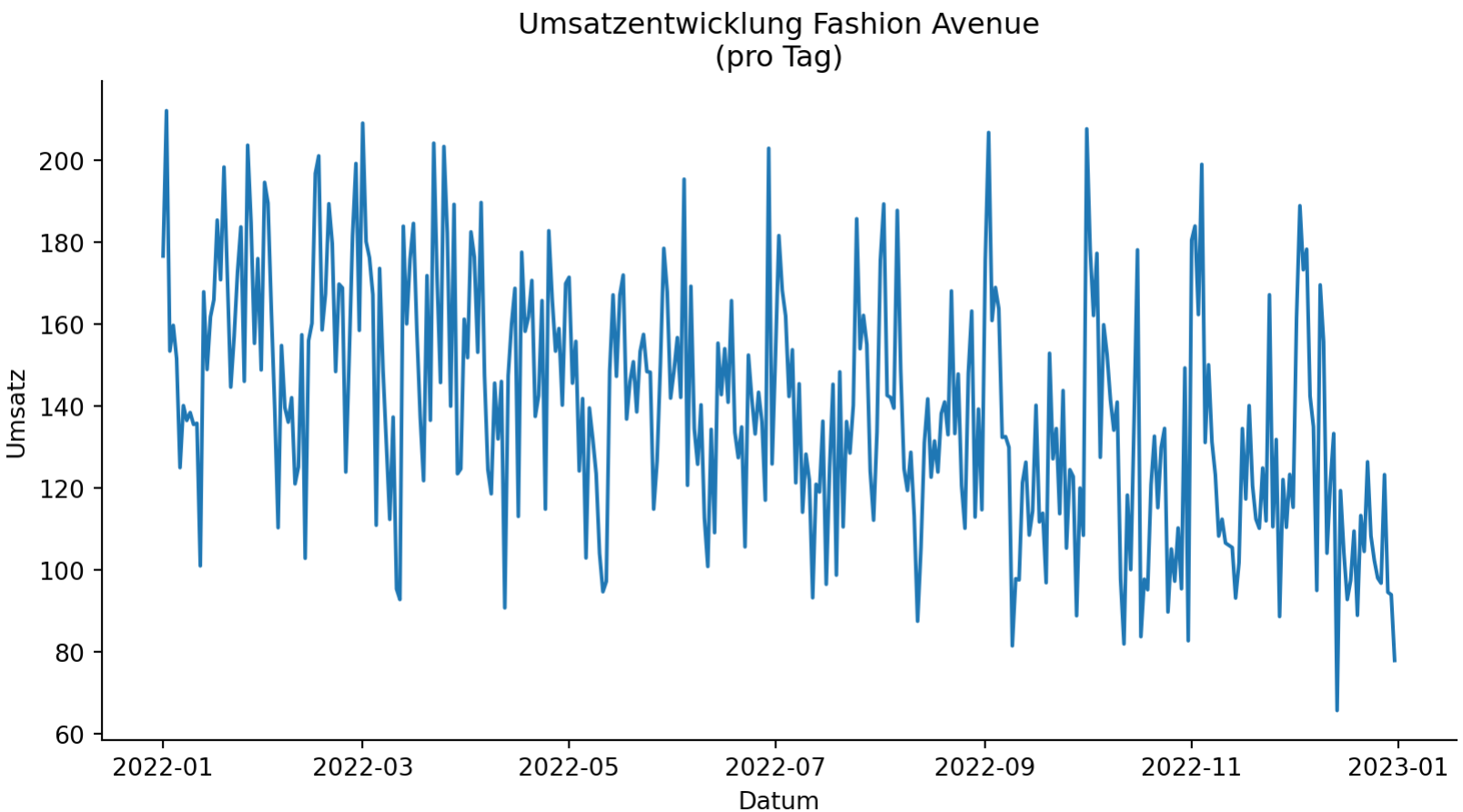
```
# Hinzufügen einer neuen Spalte "Umsatz", "Wochentag" und "Monat"
df = df.assign(Umsatz = df["Menge"] * df["Preis"],
               Monat = df["Datum"].dt.month,
               Woche = df["Datum"].dt.week,
               Wochentag = df["Datum"].dt.day_name())
```

Schritt 4 und 5: Transformierter Datensatz

	Kundenname	Alter	Zahlungsmethode	Preis	Menge	Datum	Uhrzeit	Umsatz	Monat	Woche	Wochentag
0	Gislinde Börner	29	EC-Karte	14.99	4	2022-01-01	10:24:44	59.96	1	52	Saturday
1	Pierre Ullmann	28	Bar	17.99	4	2022-01-01	10:55:09	71.96	1	52	Saturday
2	Rainer Birnbaum	57	Bar	43.99	4	2022-01-01	11:25:31	175.96	1	52	Saturday
3	Ekaterina Binner	47	Kreditkarte	36.99	3	2022-01-01	11:30:28	110.97	1	52	Saturday
4	Prof. Mandy Riehl	59	Bar	35.99	4	2022-01-01	11:46:50	143.96	1	52	Saturday

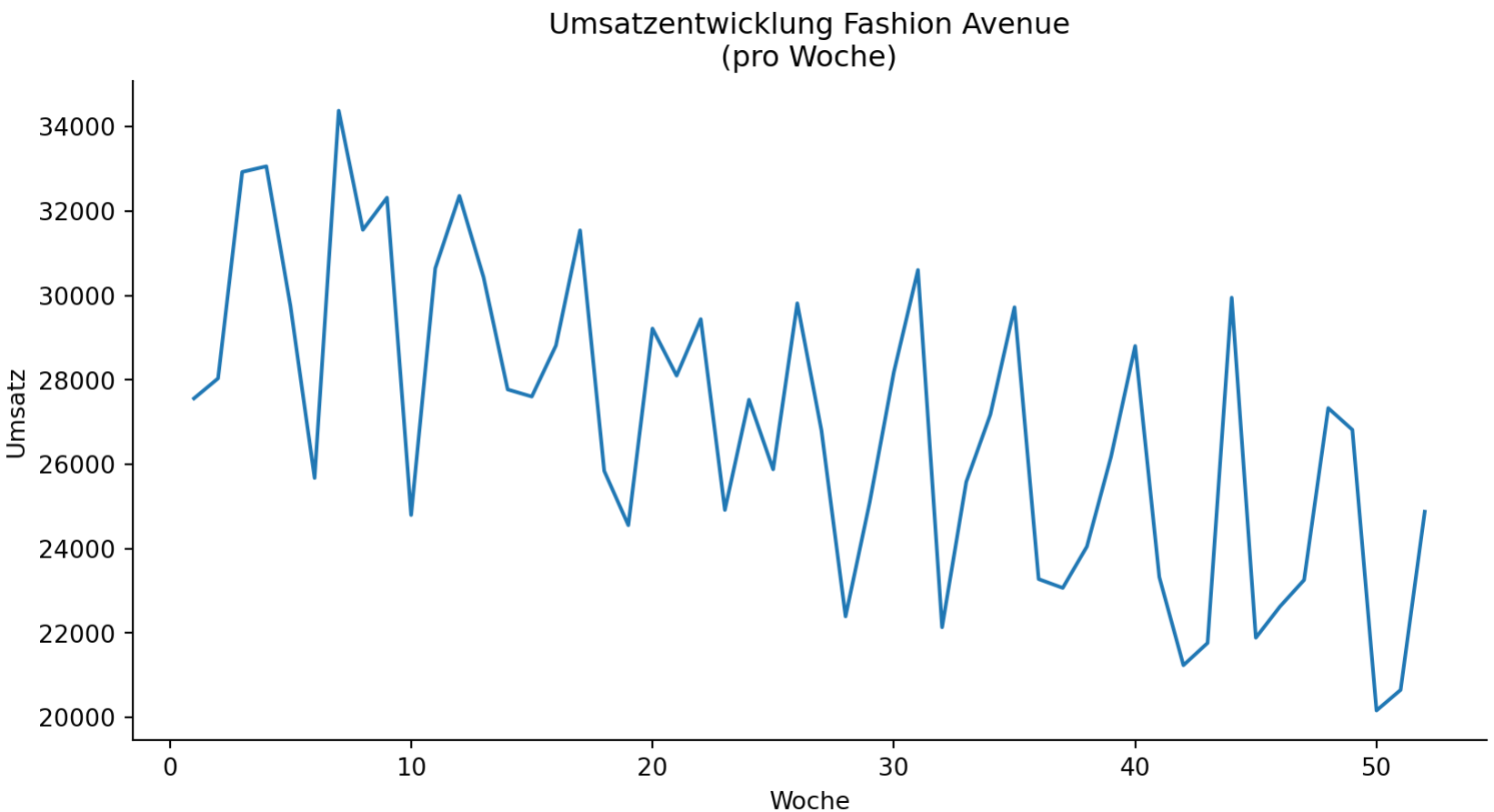
Schritt 4 und 5: Umsatzentwicklung pro Tag

Plot Code



Schritt 4 und 5: Umsatzentwicklung pro Woche

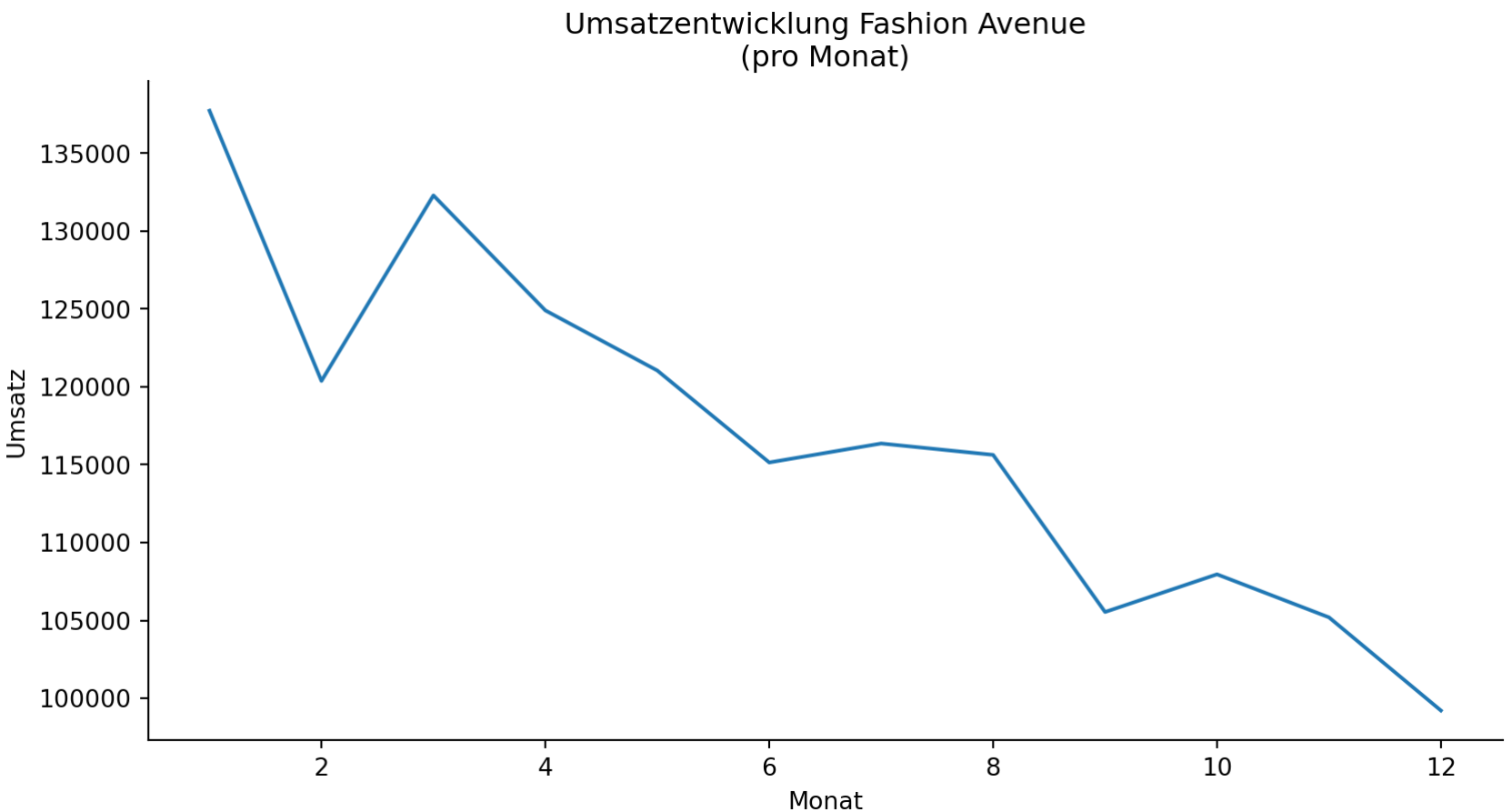
Plot | Code



Fallstudie

Schritt 4 und 5: Umsatzentwicklung pro Monat

Plot Code



Fallstudie

Schritt 4 und 5: Hypothesenbildung

Nachdem wir die Problembeschreibung validiert haben, können wir nun mit der Analyse der Ursachen beginnen.

Lassen Sie uns dafür zunächst überlegen, was mögliche Ursachen für den Umsatzrückgang sein könnten.

Was könnten mögliche Ursachen für den Umsatzrückgang sein?

Fallstudie

Schritt 4 und 5: Hypothesenbildung

Hier ein paar Beispiele die uns spontan in den Sinn kommen könnten[1]:

1. die Preise sind gesunken

1. die Nachfrage ist gesunken

3. das Zahlungsverhalten der Kunden hat sich verändert

4. die Kundenstruktur hat sich verändert

1. Hinweis: an dieser Stelle ist es wichtig, dass Sie sich nicht auf eine einzige Ursache festlegen. Es können auch mehrere oder andere Ursachen für den Umsatzrückgang verantwortlich sein. Wichtig ist, dass Sie sich auf die Ursachen konzentrieren, die Sie im Rahmen der Analyse überprüfen können.

Fallstudie

Hypothese 1: Vorbereitung

Ursache 1: Preise sind gesunken

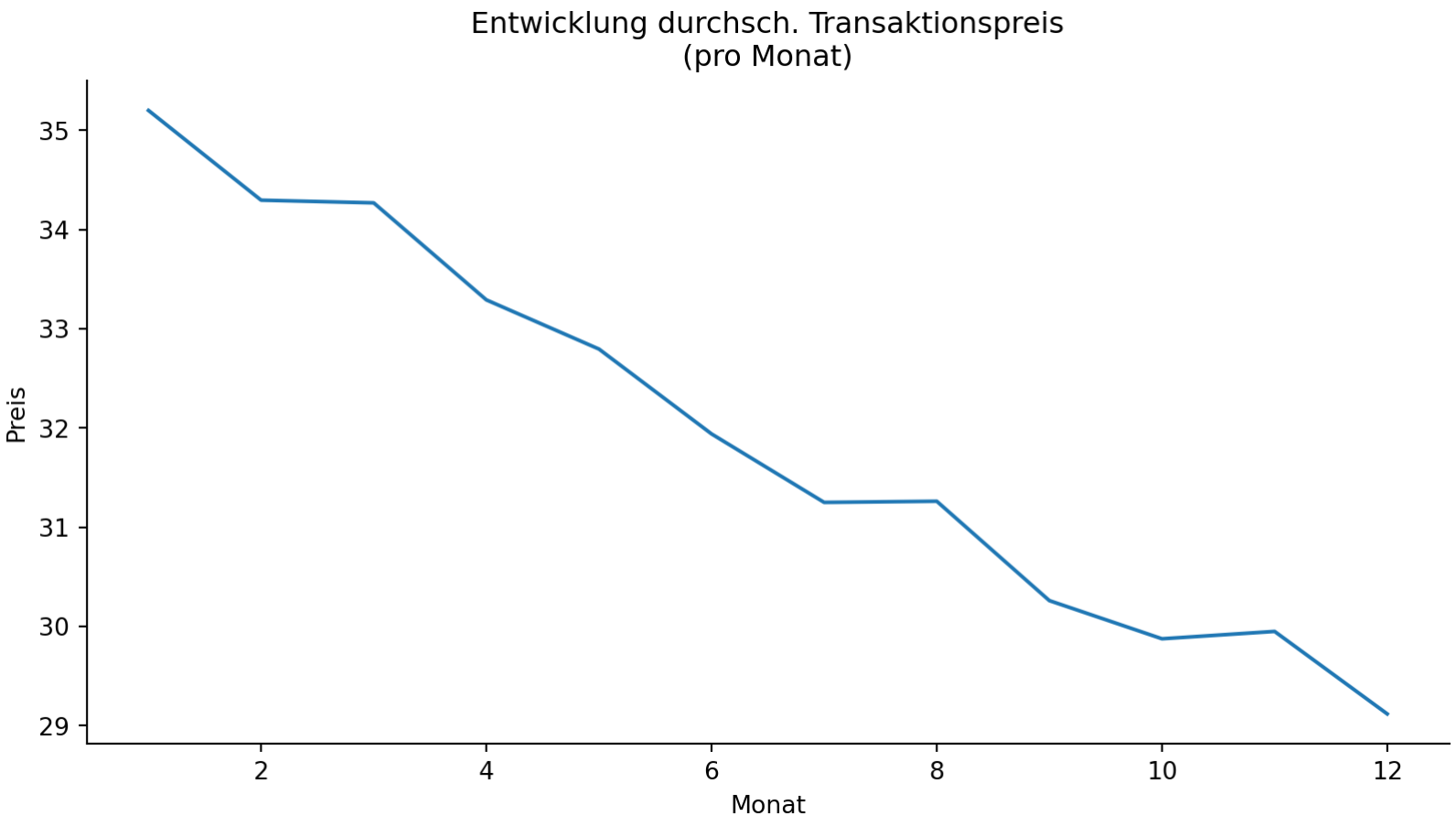
- Eine mögliche Ursache für den Umsatzrückgang ist, dass die Kunden je Kauf (d.h. in unserem Fall je Transaktion bzw. je Zeile im Datensatz) weniger Geld ausgeben.

Wie können wir dies überprüfen?

> **Vorgehen**
>
> - Daten (z.B. monatlich) aggregieren
> - je Monat durchschnittlichen Preis berechnen
> - Preisentwicklung über die Zeit analysieren

Hypothese 1: Analyse

```
1 preis_pro_monat = df.groupby("Monat").agg({"Preis": "mean"})
2 line = sns.lineplot(data=preis_pro_monat, x=preis_pro_monat.index, y="Preis")
3 line.set_title("Entwicklung durchsch. Transaktionspreis\n(pro Monat)");
```



Fallstudie

Hypothese 2: Vorbereitung

Ursache 2: Nachfrage ist gesunken

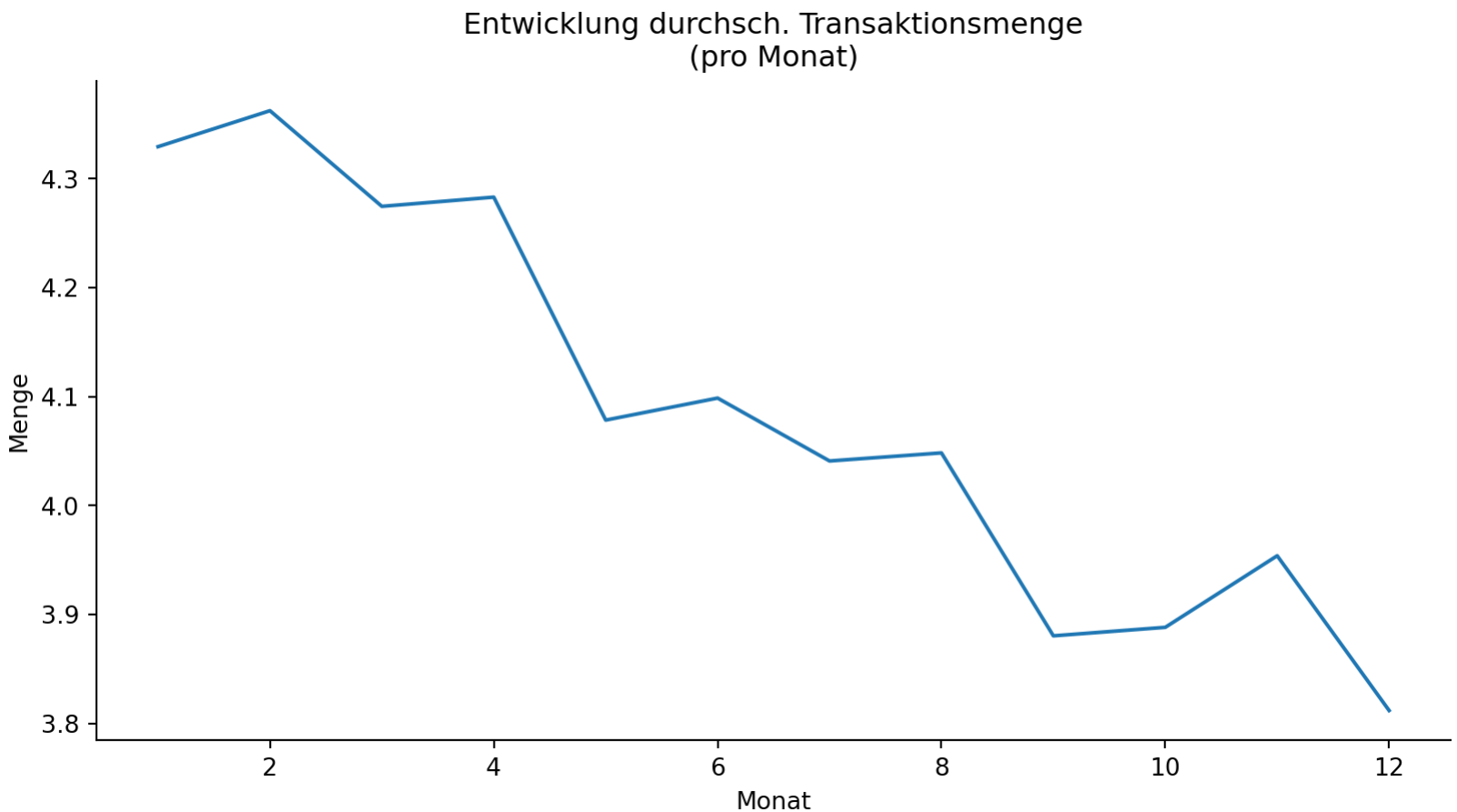
- Nachfrage nach Produkten des Unternehmens ist gesunken

Wie können wir dies überprüfen?

> **Vorgehen**
>
> - Daten (z.B. monatlich) aggregieren
> - je Monat durchschnittlichen Menge berechnen
> - Mengenentwicklung über die Zeit analysieren

Hypothese 2: Analyse

```
1 preis_pro_monat = df.groupby("Monat").agg({"Menge": "mean"})
2 line = sns.lineplot(data=preis_pro_monat, x=preis_pro_monat.index, y="Menge")
3 line.set_title("Entwicklung durchsch. Transaktionsmenge\n(pro Monat)");
```



Fallstudie

Hypothese 1 & 2: Zwischenfazit

Was wir bisher wissen

- Analyse zeigt: durchschnittliche Preise und Menge je Transaktion gesunken
- anders formuliert: Kunden kaufen günstigere, kleinere Mengen
- Warum Kunden weniger und günstiger kaufen nicht bekannt
- Ursachen 3 und 4 noch zu untersuchen

Fallstudie

Hypothese 3: Vorbereitung

Ursache 2: Zahlungsverhalten der Kunden hat sich verändert

- Datensatz enthält kaum Informationen über das Zahlungsverhalten der Kunden
- nur Information über die Zahlungsmethode verfügbar

Wie können wir dies überprüfen?

> **Vorgehen**
>
> - Gesamtumsätze und Anzahl der Transaktionen pro Zahlungsmethode aggregieren

Hypothese 3: Analyse

```
# Berechnen der Gesamtumsätze und Anzahl an Transaktionen je Zahlungsmethode
umsatz_je_zahlungsmethode = df.groupby("Zahlungsmethode").agg({"Umsatz": ["sum", "count"]})
umsatz_je_zahlungsmethode
```

	Umsatz	
	sum	count
Zahlungsmethode		
Bar	476976.95	3370
EC-Karte	477491.92	3413
Kreditkarte	446887.50	3217

Fallstudie

Hypothese 3: Analyse (cont'd)

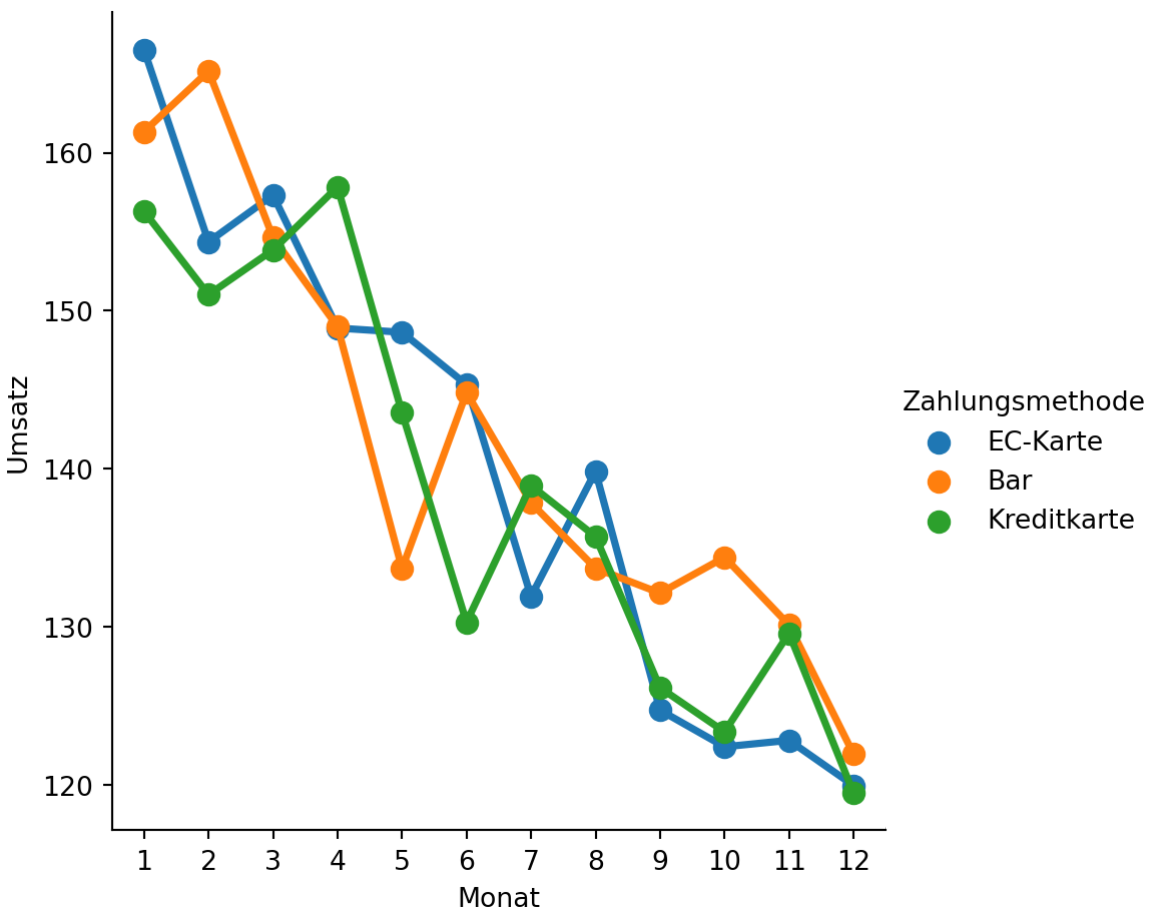
- Zahlungsmethoden unterscheiden sich scheinbar nicht
- **aber:** wir wissen nicht, ob Zahlungsmethoden über die Zeit gleichmäßig verteilt waren

Wie können wir dies überprüfen?

> **Vorgehen**
>
> - Umsätze je Monat und je Zahlungsmethode berechnen
> - Visualisierung der Ergebnisse zur Überprüfung der Annahme

Hypothese 3: Analyse (cont'd)

Plot | Code



Fallstudie

Hypothese 4: Vorbereitung

Ursache 4: Kundenstruktur hat sich verändert

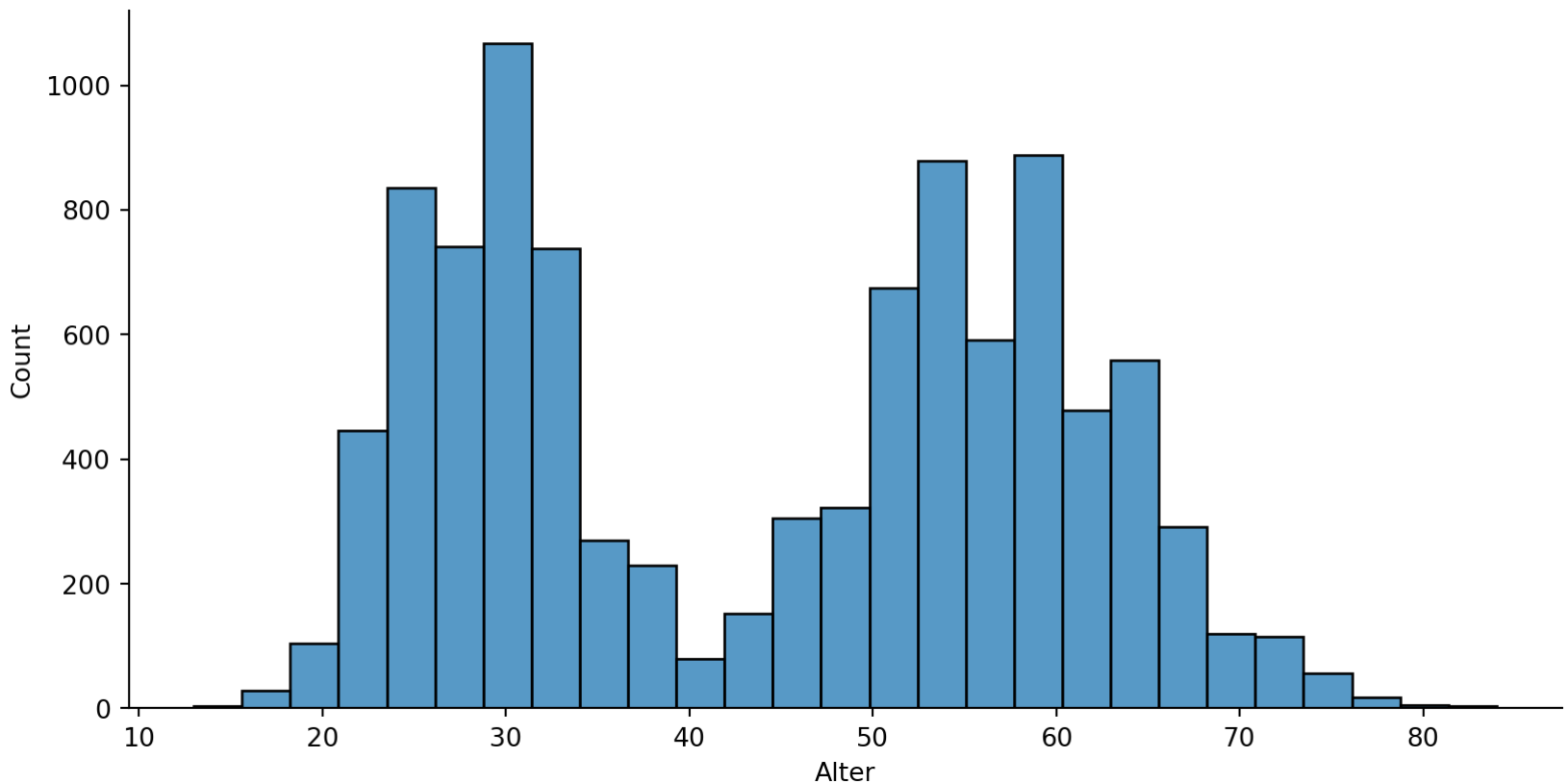
- Datensatz enthält kaum Informationen zur Kundenstruktur

Wie können wir dies überprüfen?

> **Vorgehen**
>
> - Verteilung des Kundenalters visualisieren (um Alterstruktur zu verstehen)
> - Verteilung des Kundenalters über die Zeit visualisieren (um Veränderungen zu erkennen)

Hypothese 4: Analyse

Plot Code



Fallstudie

Hypothese 4: Analyse (cont'd)

Beobachtung: Kundenalter ist nicht gleichmäßig verteilt

- Verteilung mit zwei Spitzen
- erste Spitze: ca. **25 Jahre**
- zweite Spite: ca. **55 Jahre**
- weitere Analyse sinnvoll

Was wäre eine sinnvolle Folgeanalyse?

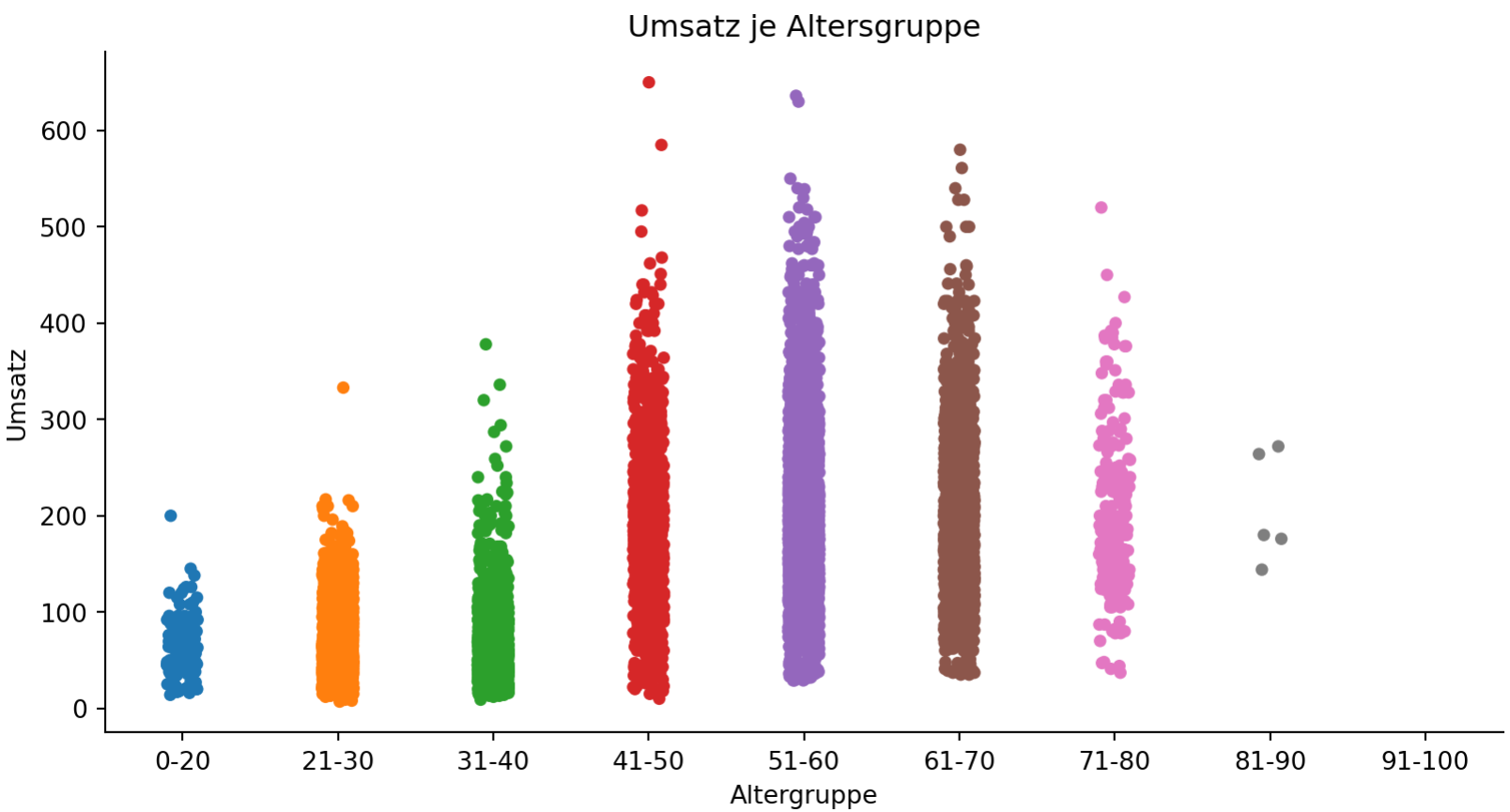
Hypothese 4: Analyse (cont'd)

Vorgehen

- Erstellen einer neuen Spalte `Altersgruppe` (z.B. 0-20, 21-30, 31-40, 41-50, 51-60, 61-70, 71-80, 81-90, 91-100)
- Analyse des Umsatzes je `Altersgruppe`

Hypothese 4: Analyse (cont'd)

Plot | Code



Fallstudie

Hypothese 1 bis 4: Zwischenfazit

Zwischenfazit:

- Umsatz und Anzahl der Transaktionen sind rückläufig
- Kunden kaufen weniger und günstiger
- Ältere Kunden haben höheren Umsatz als jüngere Kunden

Was für Folgeanalyse wäre sinnvoll?

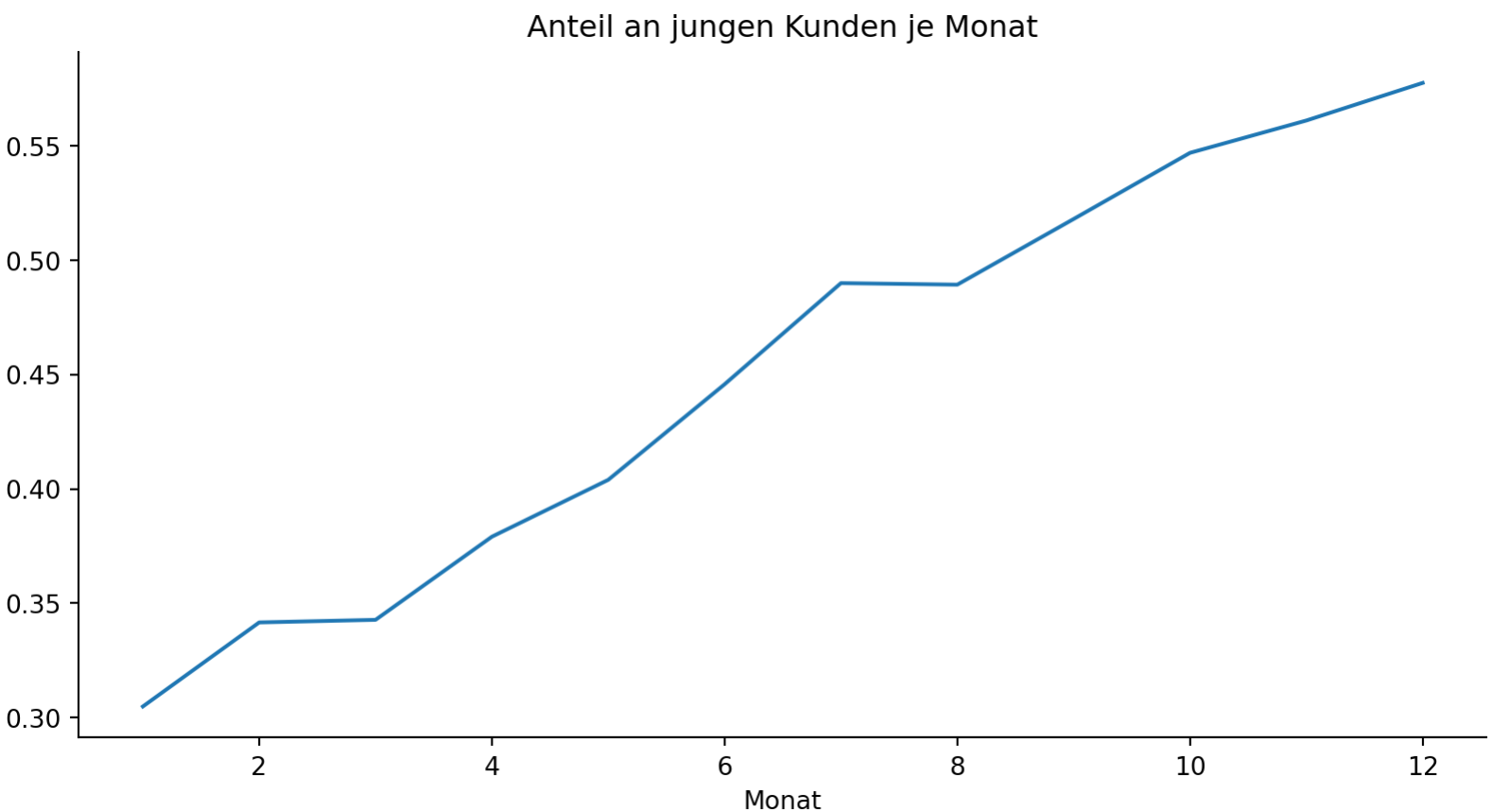
Analyse

- Entwicklung des Umsatzes je Altersgruppe über die Zeit
- Zur Vereinfachung: zwei Kundengruppen (ältere und jüngere Kunden) untersuchen
- alte Kunden: > 40 Jahre
- junge Kunden: < 40 Jahre

Fallstudie

Hypothese 4: Kundenstruktur über die Zeit

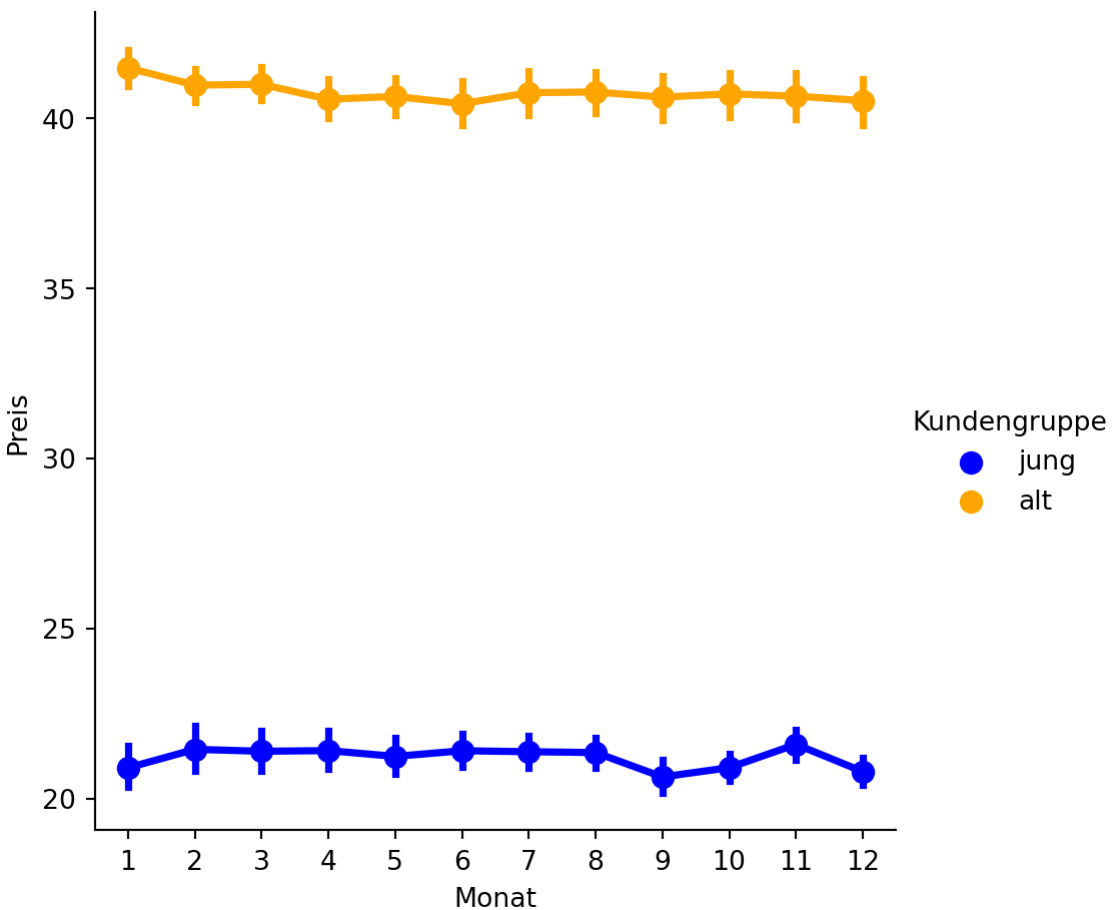
Plot | Data | Code



Fallstudie

Hypothese 4: Kaufverhalten unverändert

Preis | Menge | Code



Fazit

Lassen Sie uns ein Zwischenfazit ziehen und bisherige Erkenntnisse zusammenfassen:

- Problem definiert: “Grund für Umsatzrückgang”
- Daten geladen, bereinigt (bereits sauber)
- Problembeschreibung bestätigt (Umsatzrückgang)
- Vier Hypothesen aufgestellt, überprüft:
  - Preise gesunken
  - Nachfrage gesunken
  - Zahlungsverhalten verändert
  - Kundenstruktur verändert
- Hypothesen mit Transformationen, Visualisierungen geprüft
- Preise, Mengen gesunken (Hypothesen 1 und 2)
- Kundenstruktur verändert: mehr “unter 40” Kunden, weniger/günstiger kaufen
- Zahlungsverhalten keinen Einfluss

> **Fazit:**
>
> Umsatzrückgang durch veränderte Kundenstruktur (mehr “unter 40” Kunden, weniger/günstiger kaufen).

Fallstudie

Ausblick: Diagnostische Analyse

Plot | Auswertung | Code