DHBW Lörrach

Business Intelligence

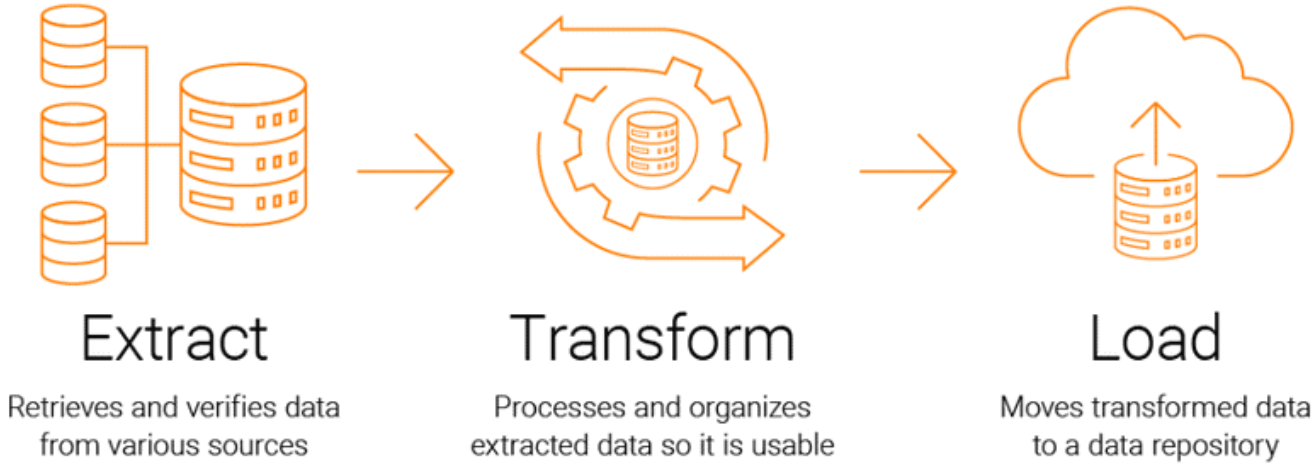
Vert. Prof. Dr. Aikaterini Nakou

ETL-Prozess

Extract: Daten aus unterschiedlichen Quellsystemen wie Datenbanken, Cloud-Diensten, APIs, Excel-Dateien usw. extrahieren.

Transform: Daten bereinigen, anreichern, normalisieren, filtern, aggregieren und formatieren, sodass sie für analytische Zwecke geeignet sind.

Load: Die transformierten Daten in ein Zielsystem laden, üblicherweise in ein Data Warehouse oder eine Datenbank.



ETL-Prozess in R

Extraktion von Daten:

Datenextraktion ist der erste Schritt im ETL-Prozess. R bietet mehrere Möglichkeiten, Daten aus verschiedenen Quellen zu extrahieren:

Extraktion aus CSV-Dateien
R bietet die Funktion read.csv(), um Daten aus CSV-Dateien zu extrahieren:

```
data <- read.csv("pfad_zur_datei.csv")
```

Extraktion aus JSON
Das jsonlite Paket wird verwendet, um JSON-Daten zu lesen:

```
library(jsonlite)
data <- fromJSON("pfad_zur_datei.json")
```

ETL-Prozess in R

Extraktion von Daten:

Datenbankanbindung
Mit DBI und RMySQL (oder anderen DB-Treibern) können Daten aus Datenbanken extrahiert werden:

```
library(DBI)
con <- dbConnect(RMySQL::MySQL(), dbname = "datenbank_name", host = "host",
user = "user", password = "password")
data <- dbGetQuery(con, "SELECT * FROM tabelle")
```

Web-Datenextraktion (APIs)
Mit httr können wir REST-APIs ansprechen:

```
library(httr)
response <- GET("https://api.example.com/data")
data <- content(response, as = "text")
```

DHBW Lörrach

ETL-Prozess in R

Extraktion von Daten - Beispiel:

In dieser Vorlesung werden wir detailliert erläutern, wie wir Daten aus zwei verschiedenen Quellen extrahieren:

CSV-Datei: Enthält Produktionskosten-Daten (productionscosts.csv)
SQLite-Datenbank: Enthält Verkaufsdaten (sales.sqlite)

Anschließend werden wir diese beiden Datensätze zusammenführen (mergen), um eine einheitliche Datenbasis für die Transformation und Analyse zu erstellen.

ETL-Prozess in R

Extraktion von Daten aus der CSV-Datei:

Wir beginnen mit dem Einlesen der Produktionskosten-Daten aus der CSV-Datei. Dazu verwenden wir die Funktion read.csv()

```
# Einlesen der Produktionskosten-Daten aus der CSV-Datei
production_costs_data <- read.csv("productioncosts.csv", header = TRUE, sep= ",", dec=".")

# Einen Blick auf die ersten Zeilen des CSV-Datensatzes werfen
head(production_costs_data)
```

ETL-Prozess in R

Extraktion von Daten aus der SQLite-Datenbank:

Wir extrahieren die Verkaufsdaten aus der SQLite-Datenbank. Dafür verwenden wir das Paket RSQLite, die speziell für den Zugriff auf SQLite-Datenbanken in R entwickelt wurden.

1. Verbindung zur SQLite-Datenbank herstellen

```
# Verbindung zur SQLite-Datenbank herstellen
con <- dbConnect(SQLite(), dbname = "sales_database.sqlite")
```

2. Extraktion der Verkaufsdaten

```
# Extrahieren der Verkaufsdaten aus der Datenbanktabelle 'sales'
sales_data <- dbReadTable(con, "sales")
```

3. Schließen der Verbindung zur Datenbank

```
# Schließen der Verbindung zur Datenbank
dbDisconnect(con)
```

DHBW Lörrach

ETL-Prozess in R

Zusammenführen (Merging) der Daten aus CSV und SQLite-Datenbank:

Wir führen die Daten aus den beiden Quellen (productionscosts.csv und sales_database.sqlite) zusammen, um eine einheitliche Datentabelle zu erstellen. Hierfür verwenden wir die dplyr-Funktion left_join().

```
# Zusammenführen der Produktionskosten- und Verkaufsdaten
merged_data <- production_costs_data %>%
  left_join(sales_data, by = c("Date", "Country"))

# Überprüfen der ersten Zeilen der zusammengeführten Daten
head(merged_data)
```

Erklärung:

- left_join(): Wir verwenden left_join(), um sicherzustellen, dass alle Zeilen aus den Produktionskosten-Daten (productionscosts.csv) erhalten bleiben, selbst wenn es keine übereinstimmenden Zeilen in den Verkaufsdaten (sales.sqlite) gibt.

- Die Verknüpfung erfolgt auf Basis der gemeinsamen Spalten Date und Country.

ETL-Prozess in R

Daten-Transformation:

Die Transformation von Daten ist der Prozess, durch den Rohdaten in ein nützliches Format für die Datenanalyse umgewandelt werden. Dies umfasst:

- **Datenbereinigung**: Entfernen oder Korrigieren fehlerhafter Daten.
- **Umformung**: Anpassen der Datenstruktur, um sie besser analysieren zu können.
- **Filterung**: Auswahl nur relevante Informationen für unsere Analyse.
- **Erstellung neuer Features**: Berechnung neuer Kennzahlen aus den vorhandenen Daten.

ETL-Prozess in R

Daten-Transformation - Datenbereinigung:

Umgang mit fehlenden Werten

Zunächst überprüfen wir, ob in den Daten fehlende Werte vorhanden sind, und entscheiden dann, wie wir diese behandeln.

```
31  # Überprüfung auf fehlende Werte in jedem Feld des zusammengeführten Datensatzes
32  colSums(is.na(merged_data))
```

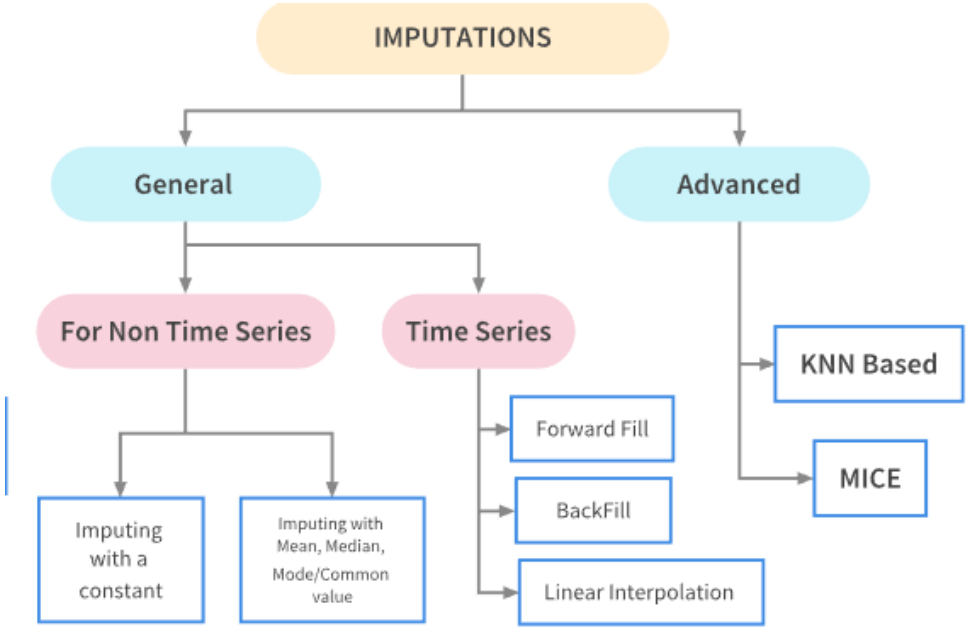
ETL-Prozess in R

Daten-Transformation - Datenbereinigung:

Umgang mit fehlenden Werten

Es gibt eine Vielzahl von Möglichkeiten, wie fehlende Werte je nach Art des Problems und der Daten unterstellt werden können:

- **Ersetzen mit Durchschnittswerten**: Besonders nützlich bei numerischen Daten.
- **Löschen von Zeilen**: Wenn nur wenige Zeilen betroffen sind, kann es sinnvoll sein, diese zu entfernen.
- **Vorherige/nachfolgende Werte verwenden**: Insbesondere bei Zeitreihendaten, wo man die Lücken durch den vorherigen oder nachfolgenden Wert schließen kann.



ETL-Prozess in R

Daten-Transformation - Umformung:

Datumsumwandlung

Die korrekte Formatierung von Datumsangaben ist entscheidend, insbesondere wenn wir später Zeitreihenanalysen durchführen wollen.

```
40  # Umwandlung der Date-Spalte in das Datumsformat
41  merged_data$Date <- as.Date(merged_data$Date, format = "%Y-%m-%d")
42
```

ETL-Prozess in R

Daten-Transformation - Filterung:

Filterung nach einem bestimmten Jahr

Wenn wir Daten nur für ein bestimmtes Jahr analysieren möchten, beispielsweise das Jahr 2020, verwenden wir folgende Syntax:

```
# Filterung der Daten für das Jahr 2020
filtered_data_2020 <- merged_data %>%
  filter(Year == 2020)

# Anzeige der gefilterten Daten
head(filtered_data_2020)
```

Filterung für einen bestimmten Land
Wir können auch ein bestimmte Land auswählen, z. B. alle Daten aus Deutschland:

```
# Filterung der Daten für Germany
filtered_data_germany <- merged_data %>%
  filter(Country == "Germany")
```

ETL-Prozess in R

Daten-Transformation - Filterung:

Filterung nach hohem Verkaufsumsatz

Wir möchten nur Datensätze betrachten, bei denen der Verkaufsumsatz (Sales_Revenue) höher als 10000 ist:

```
# Filterung der Daten für Verkaufsumsatz größer als 1000
high_revenue_data <- merged_data %>%
  filter(Sales_Revenue > 10000)
```

Kombinierte Filterung

In der Praxis müssen oft mehrere Filterkriterien gleichzeitig angewendet werden.

```
# Kombinierte Filterung: Daten aus Deutschland und Verkaufsumsatz größer als 10000
filtered_combined <- merged_data %>%
  filter(Country == "Germany" & Sales_Revenue > 10000)
```

ETL-Prozess in R

Daten-Transformation - Erstellung neuer Features:

Berechnung des Gewinns

Der Gewinn kann als Differenz zwischen Sales_Revenue und Production_Costs berechnet werden:

```
# Berechnung des Gewinns
merged_data$Profit <- merged_data$Sales_Revenue - merged_data$Production_Costs
```

Beschwerdequote

Die Beschwerdequote zeigt an, wie viele Beschwerden pro aktivem Benutzer geöffnet wurden:

```
# Berechnung der Beschwerdequote
merged_data$Complaint_Rate <- merged_data$Opened_Complaints / merged_data$Active_Users
```

ETL-Prozess in R

Datenladen:

SQLite-Datenbank

```r
# Verbindung zur SQLite-Datenbank herstellen (oder erstellen, falls nicht vorhanden)
output <- "output.sqlite"
con <- dbConnect(SQLite(), dbname = output)

# Daten in die Datenbank schreiben (Überschreiben, falls die Tabelle bereits existiert)
dbWriteTable(con, "merged_data", merged_data, overwrite = TRUE, row.names = FALSE)

# Schließen der Verbindung zur Datenbank
dbDisconnect(con)
```

CSV-Datei

```r
# Definieren des Pfades zur CSV-Datei
csv_file_path <- "merged_data.csv"

# Daten in eine CSV-Datei schreiben
write.csv(merged_data, file = csv_file_path, row.names = FALSE)
```

Datenvisualisierung

Bei der Datenvisualisierung geht es darum:

- Informationen in Form von Zahlen und Daten mit Hilfe von grafischen Mitteln aufzubereiten
- um rasch und mit wenig Vorkenntnissen Muster, Trends, Beziehungen und Ausreißer erkennbar zu machen.

Die gängigsten grafischen Hilfsmittel für Datenvisualisierung sind u.a. **Diagramme, Graphen, Karten, Tabellen**, **Infografiken** und **Dashboards**.

DHBW Lörrach

Warum Datenvisualisierung

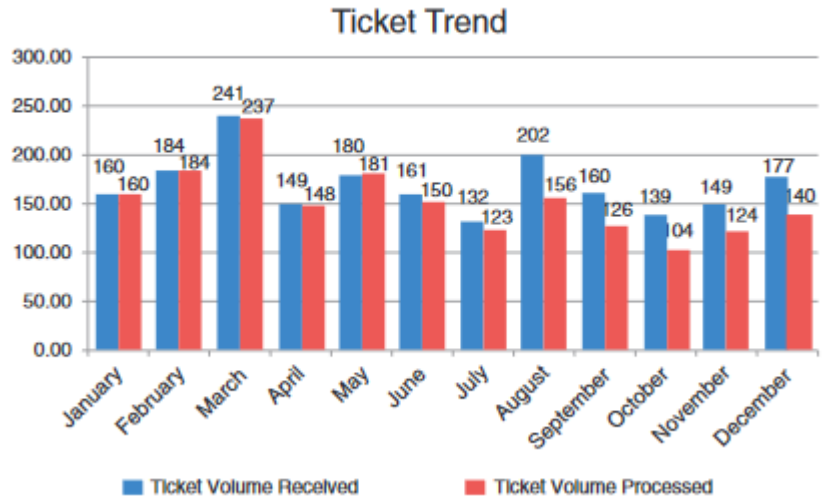
- Menschen sind in der Lage, extrem schnell aus komplexen visuellen Szenen wichtige Informationen herauszulesen
- Im Gegensatz zu verbalen Informationen, verarbeitet der Gehirn visuelle Informationen parallel
- Der Mensch erlernt visuelle Informationen schneller und erinnert sich an diese besser als verbale.

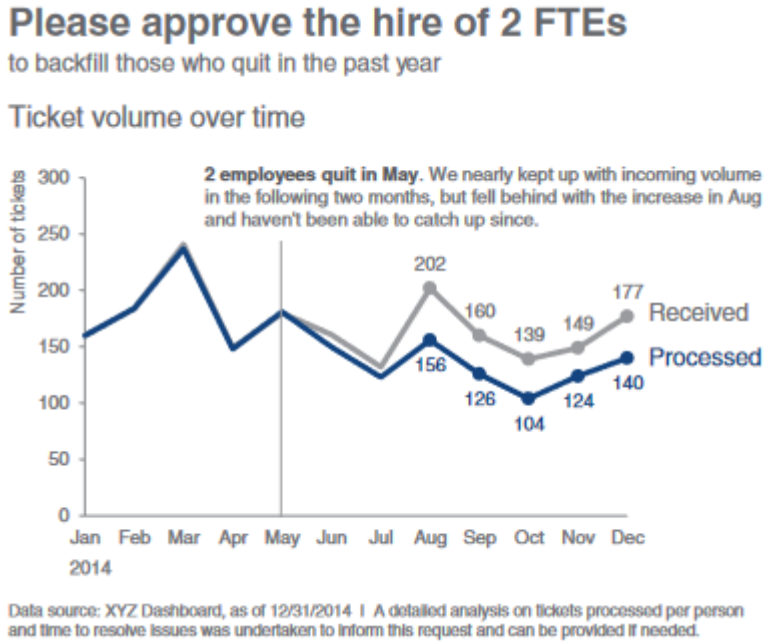
Arten der Datenvisualisierung

- Visualisierungen, die die Exploration der Daten bzw. die Verifikation der Datenanalyse dienen

- Visualisierungen, die die Präsentationen der Daten dienen → „Storytelling“ mit Daten

Schlechte vs Gute Grafiken





Prinzipien der Datenvisualisierung

- Den Kontext verstehen
- Auswahl der geeignete visuelle Darstellung
- Beseitigung der Unordnung
- Richten der Aufmerksamkeit dorthin, wo es gewollt ist
- Richtige Designing
- Storytelling

Kontext verstehen

Frage1: Welche Geschichte möchten Sie erzählen?

Frage 2: Was ist Ihre Zielgruppe?

Frage 3: Möchten Sie bestimmte Trends analysieren?

Frage 4: Soll die Zusammensetzung von Daten präsentiert werden?

Frage 5: Sollen Daten miteinander verglichen werden?

Frage 6: Ist ein bestimmter Zeitrahmen zu betrachten?

Auswahl der geeignete visuelle Darstellung

Simple text

Scatterplot

	A	B	C
Category 1	15%	22%	42%
Category 2	40%	36%	20%
Category 3	35%	17%	34%
Category 4	30%	29%	26%
Category 5	55%	30%	58%
Category 6	11%	25%	49%

Table

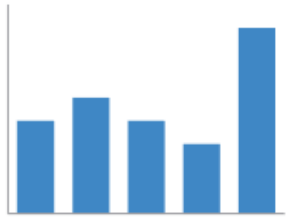
Line

	A	B	C
Category 1	15%	22%	42%
Category 2	40%	36%	20%
Category 3	35%	17%	34%
Category 4	30%	29%	26%
Category 5	55%	30%	58%
Category 6	11%	25%	49%

Heatmap

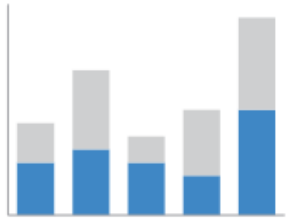
Slopegraph

FIGURE 2.1 The visuals I use most

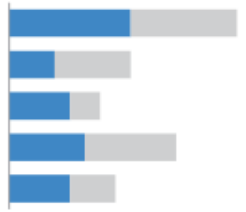


Vertical bar

Horizontal bar

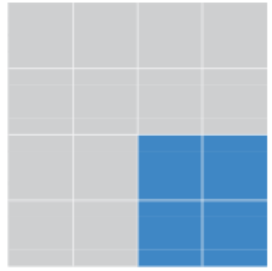


Stacked vertical bar

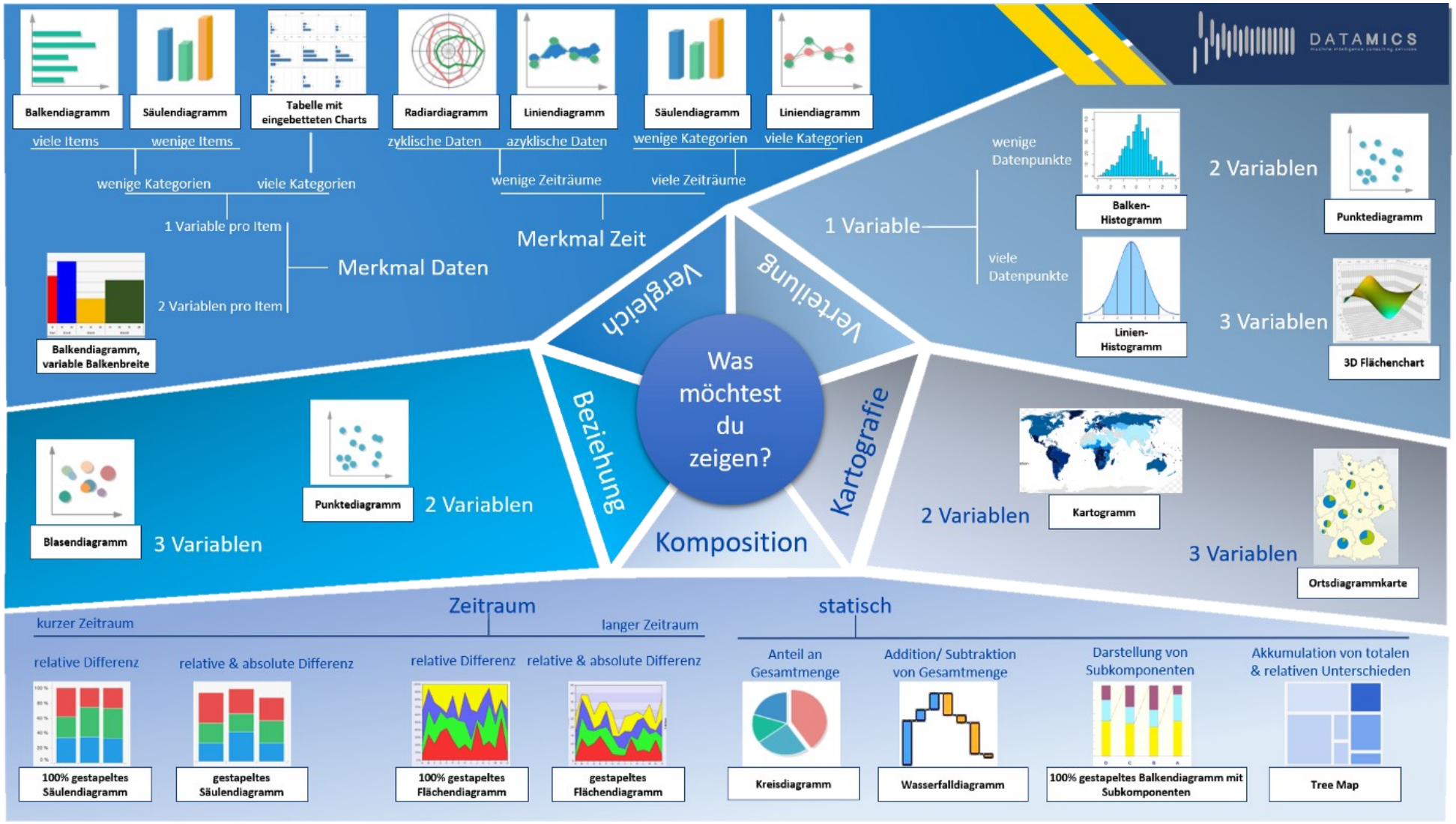


Stacked horizontal bar

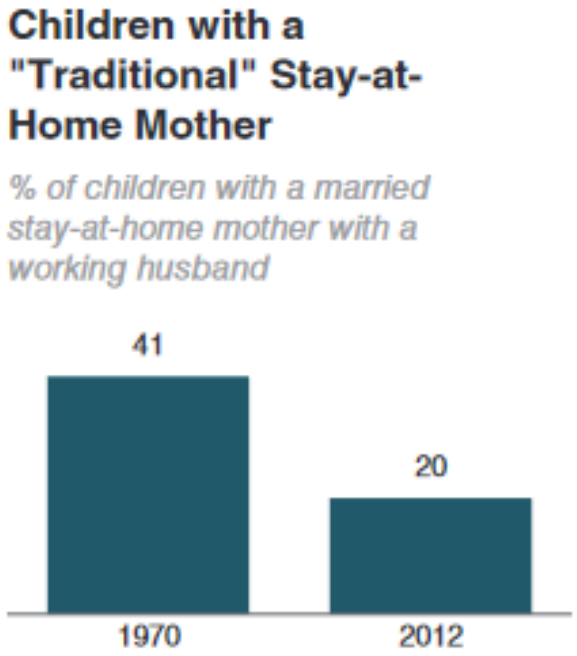
Waterfall



Square area


DATAMICS
Was möchtest du zeigen?
Merkmal Daten
Merkmal Zeit
Vergleich
Verteilung
Beziehung
Kartografie
Komposition
Balkendiagramm
Säulendiagramm
Tabelle mit eingebetteten Charts
Radiardiagramm
Liniendiagramm
Säulendiagramm
Liniendiagramm
viele Items
wenige Items
zyklische Daten
azyklische Daten
wenige Kategorien
viele Kategorien
wenige Kategorien
viele Kategorien
wenige Zeiträume
viele Zeiträume
1 Variable pro Item
2 Variablen pro Item
Balkendiagramm, variable Balkenbreite
1 Variable
wenige Datenpunkte
viele Datenpunkte
Balken-Histogramm
Linien-Histogramm
2 Variablen
3 Variablen
Punktediagramm
3D Flächenchart
Blasendiagramm
Punktediagramm
3 Variablen
2 Variablen
2 Variablen
Kartogramm
3 Variablen
Ortsdiagrammkarte
Zeitraum
kurzer Zeitraum
langer Zeitraum
relative Differenz
relative & absolute Differenz
relative Differenz
relative & absolute Differenz
100% gestapeltes Säulendiagramm
gestapeltes Säulendiagramm
100% gestapeltes Flächendiagramm
gestapeltes Flächendiagramm
statisch
Anteil an Gesamtmenge
Addition/ Subtraktion von Gesamtmenge
Darstellung von Subkomponenten
Akkumulation von totalen & relativen Unterschieden
Kreisdiagramm
Wasserfalldiagramm
100% gestapeltes Balkendiagramm mit Subkomponenten
Tree Map

Auswahl der geeignete visuelle Darstellung



Beseitigung der Unordnung

- **Kognitive Belastung** ist die geistige Anstrengung, die erforderlich ist, um neue Informationen zu lernen.

- Als Designer von Informationen müssen wir wie wir die **Gehirnleistung** unseres Publikums nutzen und **nicht ausnutzen!**

Beseitigung der Unordnung

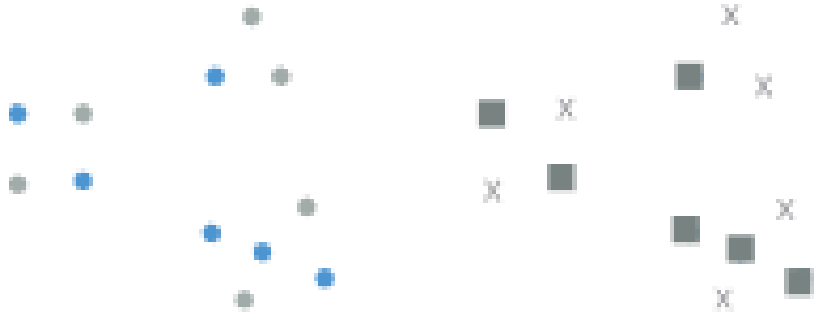
Gestaltprinzipien der visuellen Wahrnehmung:

- Nähe
- Ähnlichkeit
- Einschluss
- Abgeschlossenheit
- Kontinuität
- Verbindung

Beseitigung der Unordnung

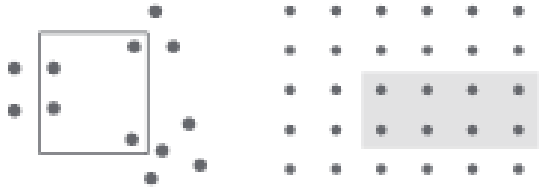
Nähe

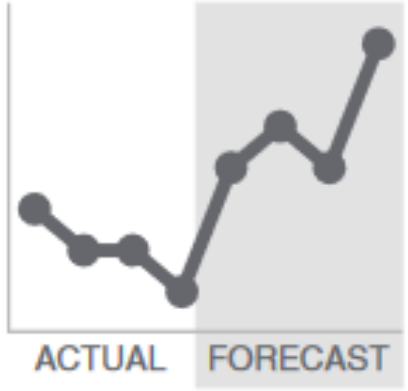
Ähnlichkeit



Beseitigung der Unordnung

Einschluss

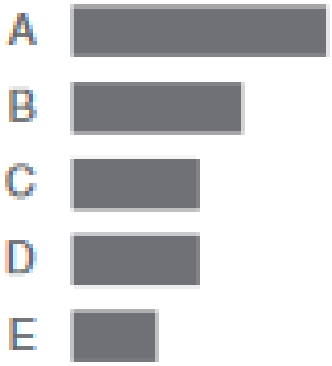




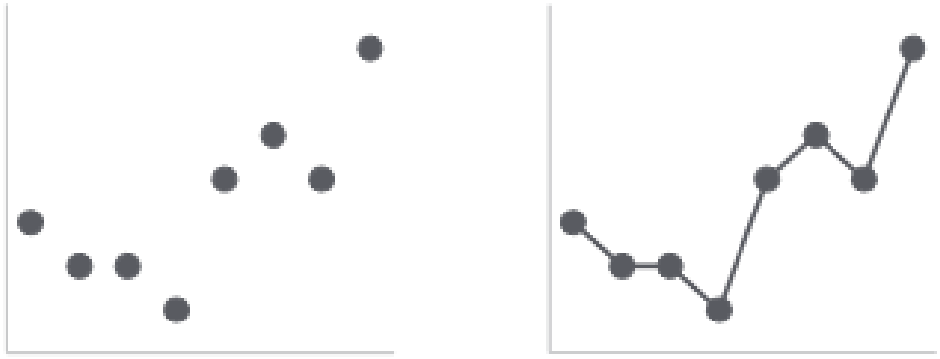
Abgeschlossenheit

Beseitigung der Unordnung

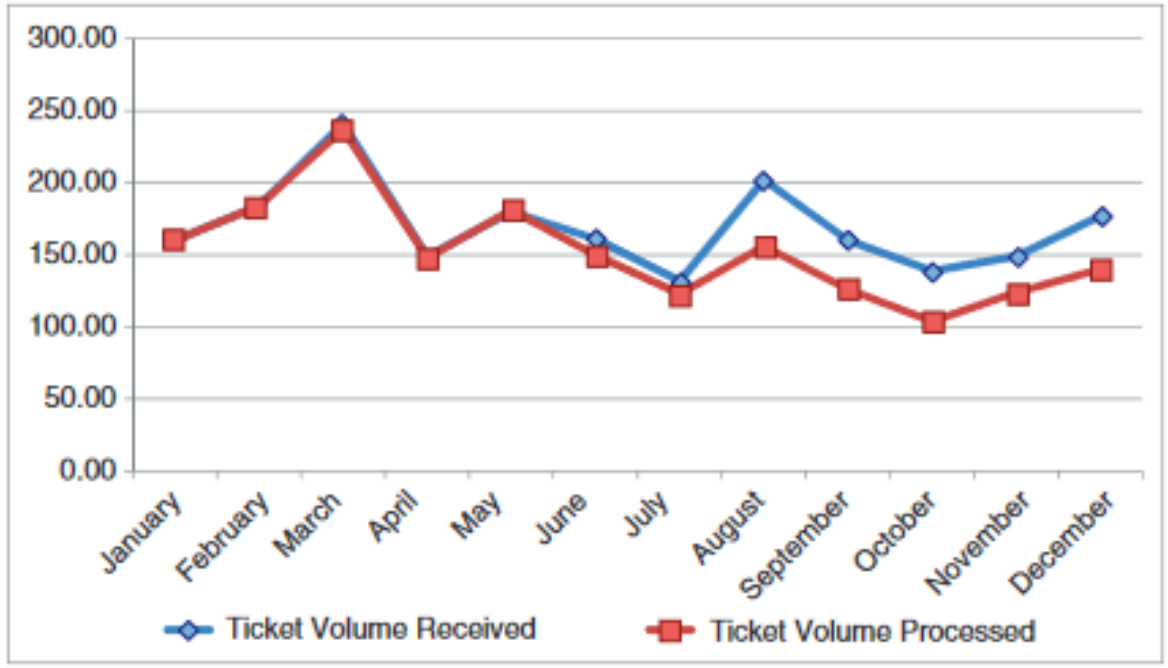
Kontinuität



Verbindung

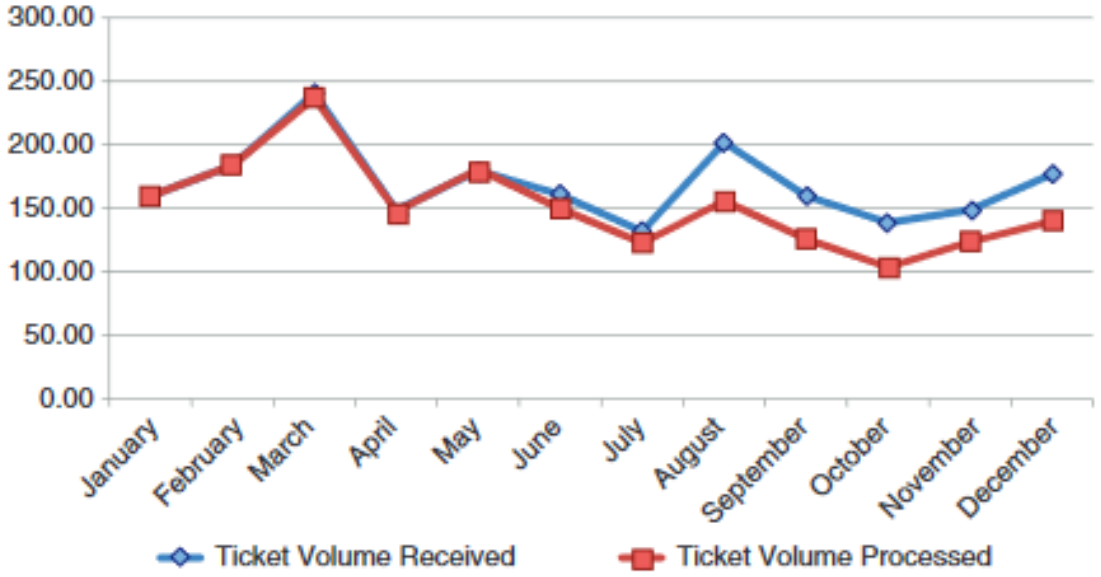


Beseitigung der Unordnung

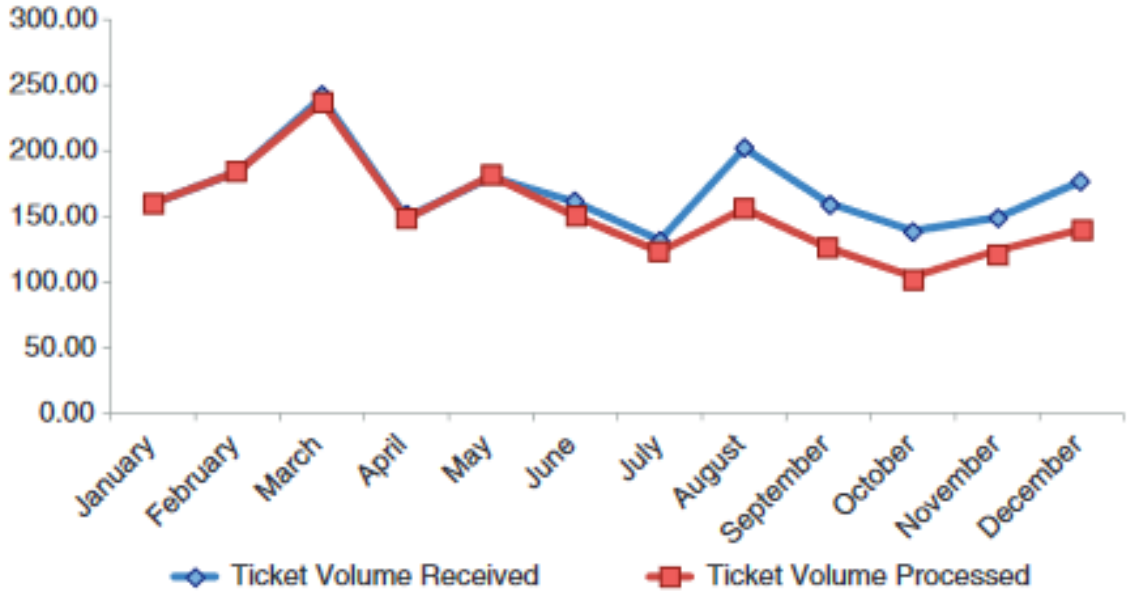


DHBW Lörrach

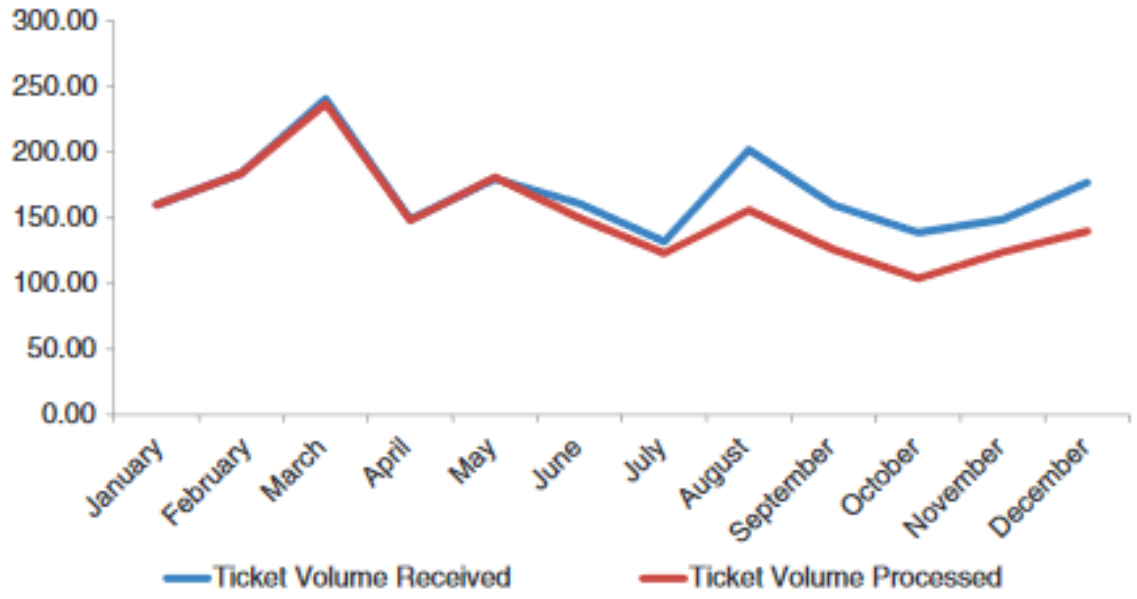
Beseitigung der Unordnung



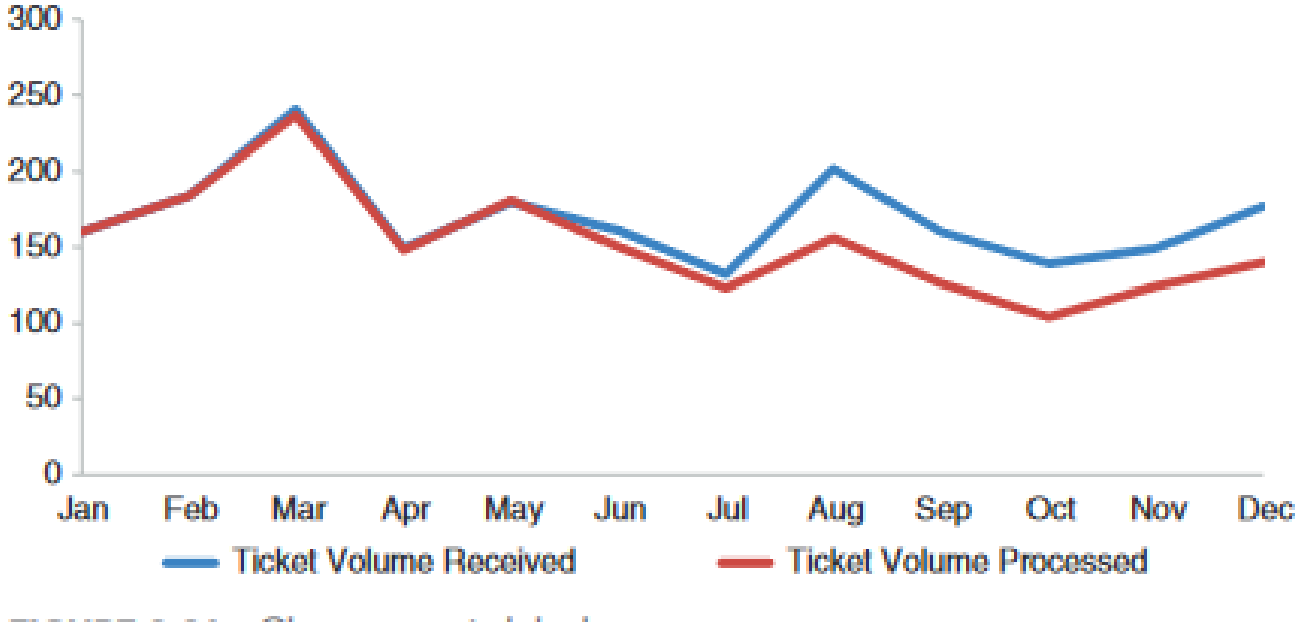
Beseitigung der Unordnung



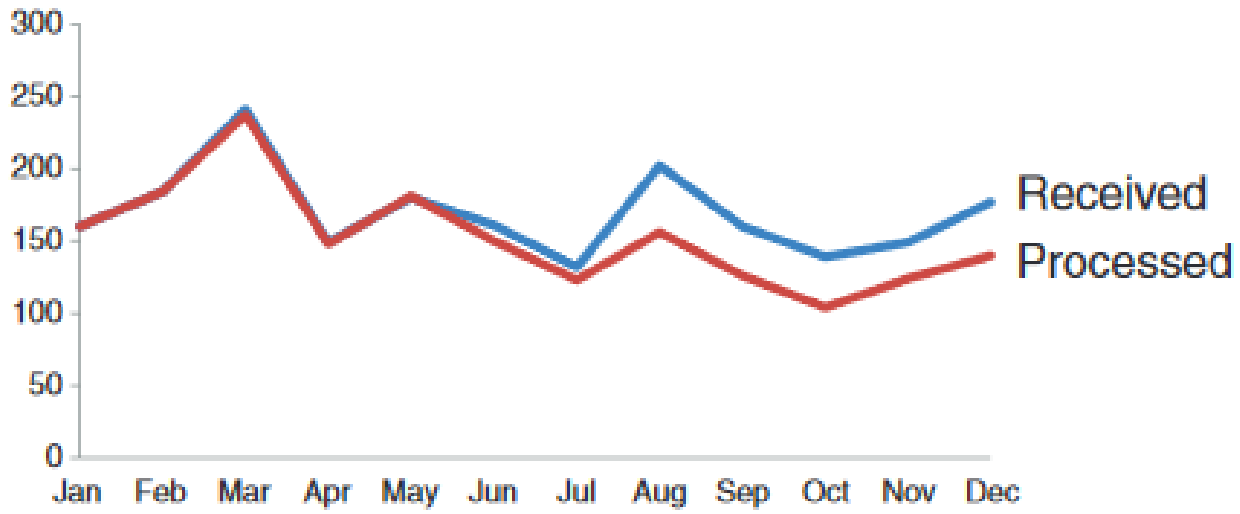
Beseitigung der Unordnung



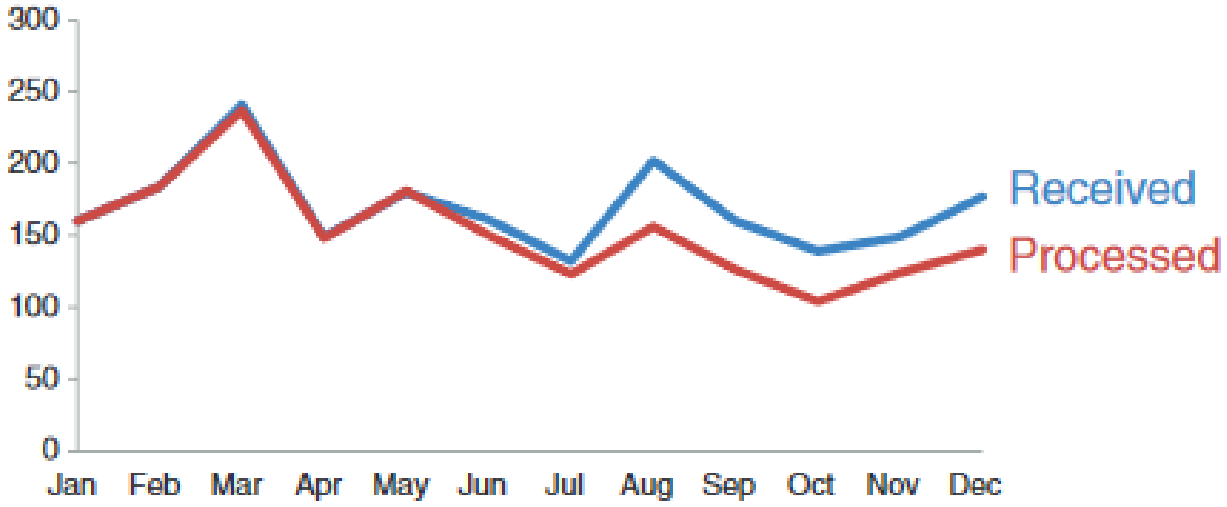
Beseitigung der Unordnung



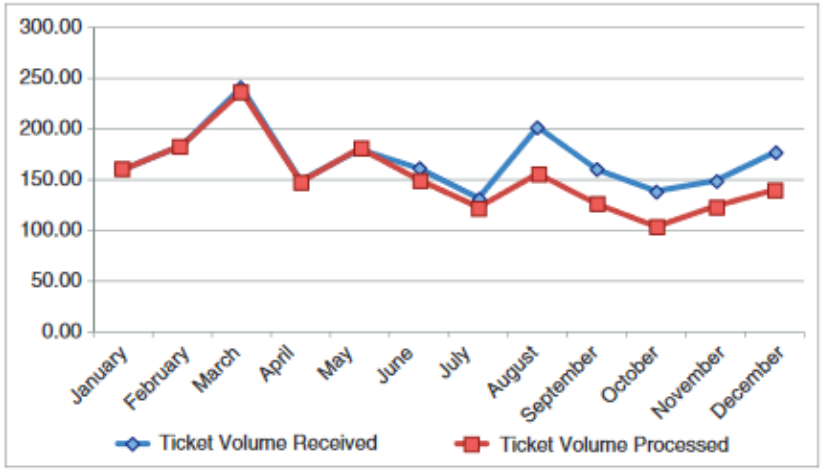
Beseitigung der Unordnung

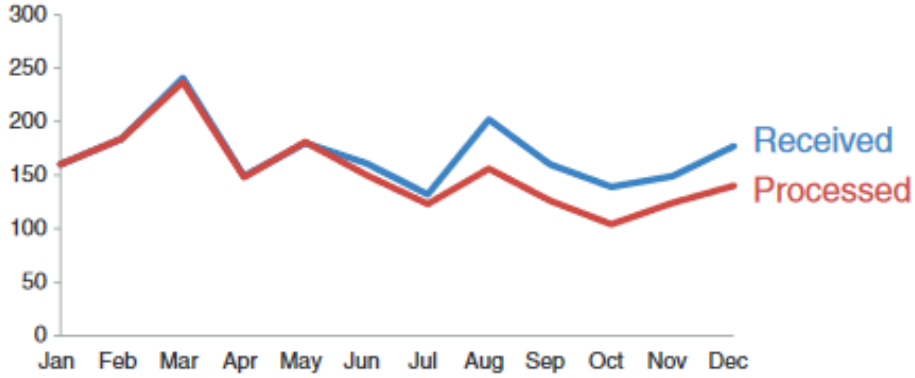


Beseitigung der Unordnung



Beseitigung der Unordnung





Fokussierung der Aufmerksamkeit

Sehen und Gedächtnis → präattentive Attribute → Lenkung der Aufmerksamkeit des Publikums, wo wir sie haben möchten!

Präattentive Attribute

- Farbe
- Größe
- Position

Fokussierung der Aufmerksamkeit

Präattentives Attribute: Farbe

Fokussierung der Aufmerksamkeit

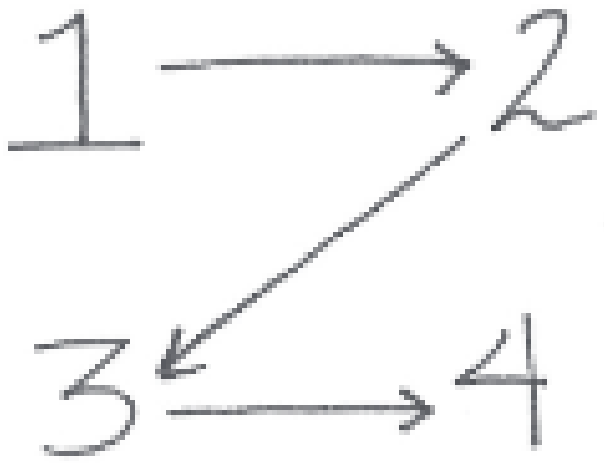
Präattentives Attribute: Farbe

Fokussierung der Aufmerksamkeit

Präattentives Attribute:Farbe

Fokussierung der Aufmerksamkeit

Präattentives Attribute: Position



Richtige Designe

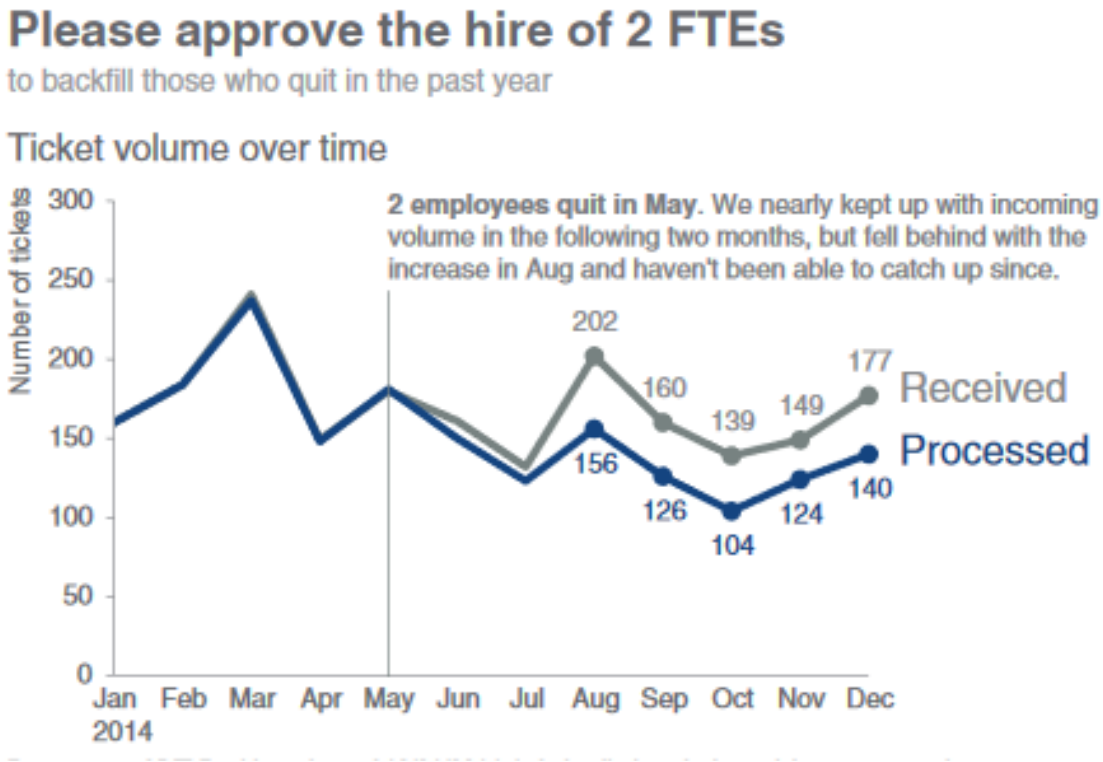
Folge das Sprichwort der Produktdesign:

Die Form folgt der Funktion

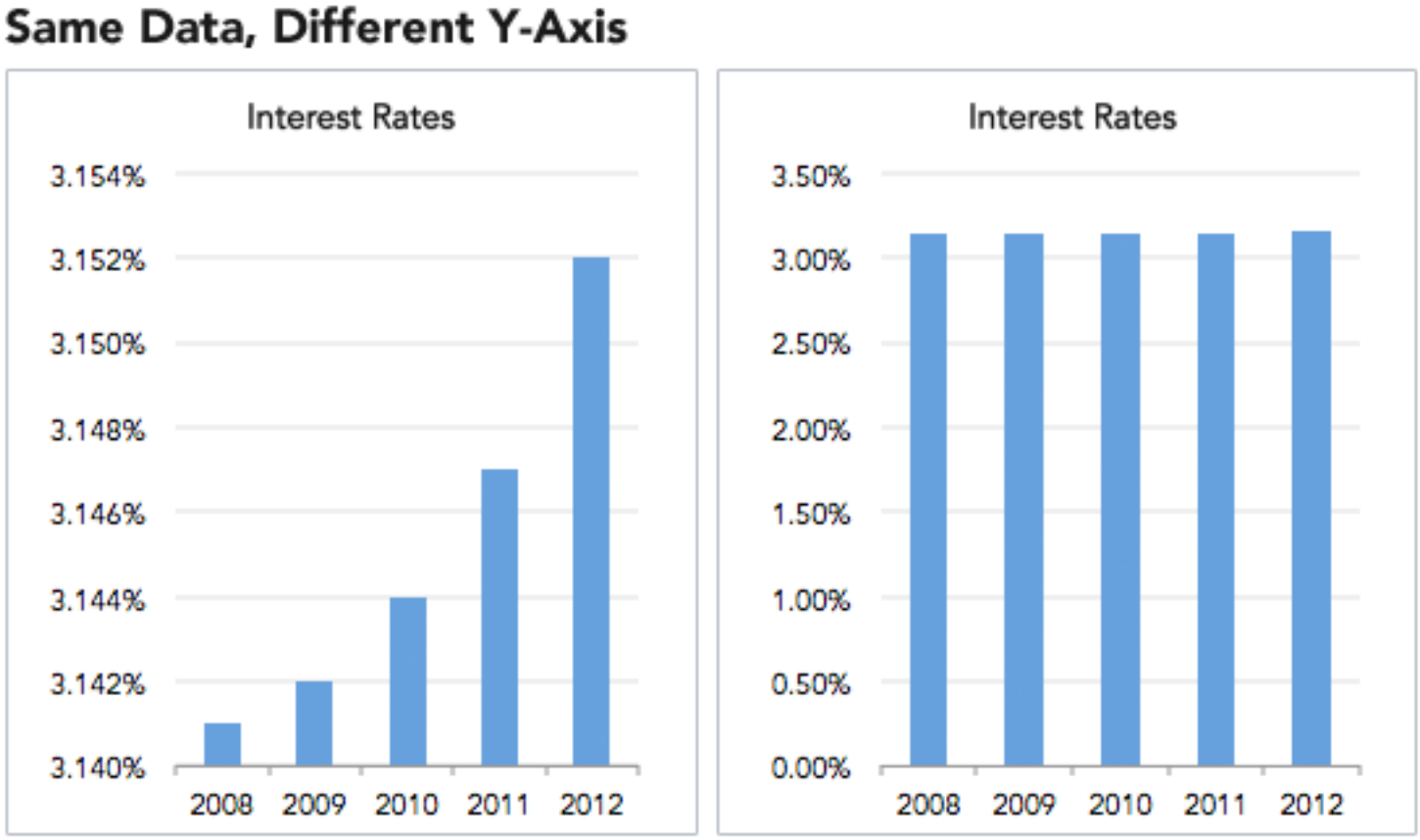
Also überlegen was unser Publikum mit den Daten machen soll (Funktion)
und dann eine Visualisierung (Form) erstellen!

Richtige Designe

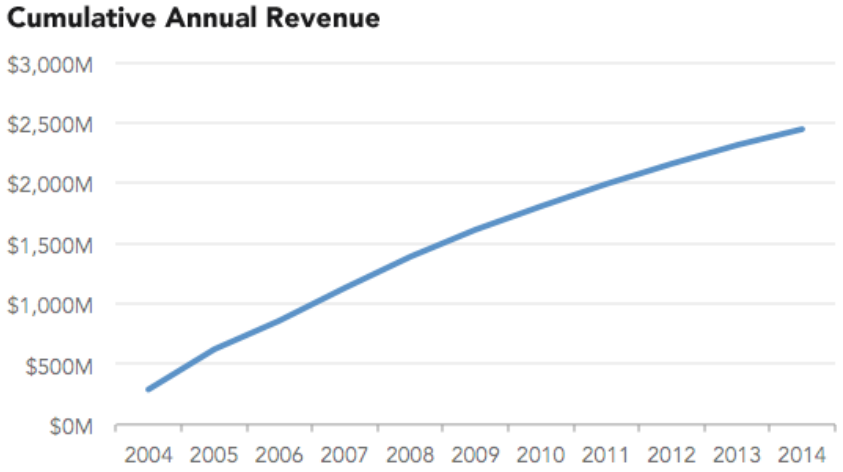
Zugänglichkeit der Informationen!

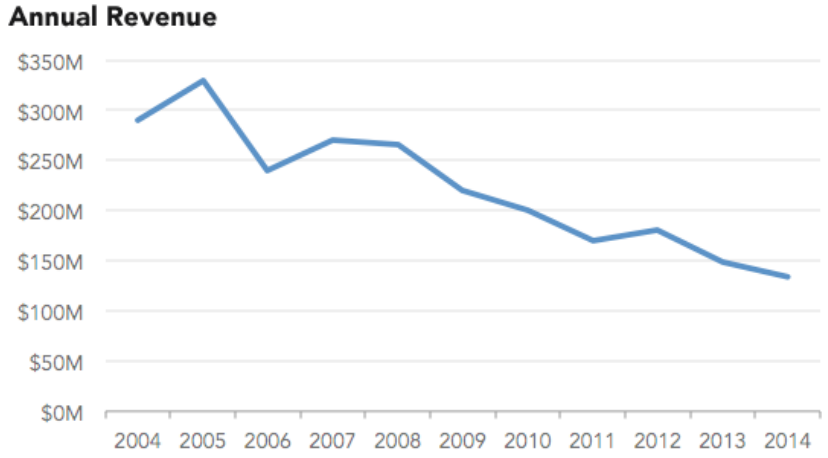


Manipulation durch Datenvisualisierung



Manipulation durch Datenvisualisierung

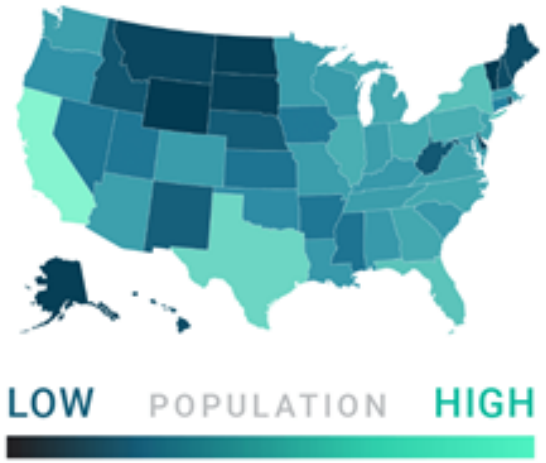




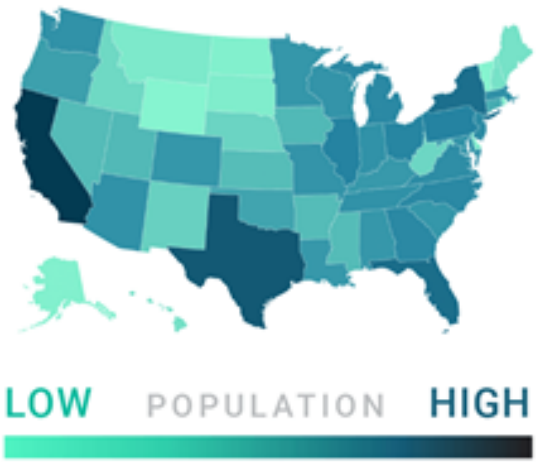
DHBW Lörrach

Manipulation durch Datenvisualisierung

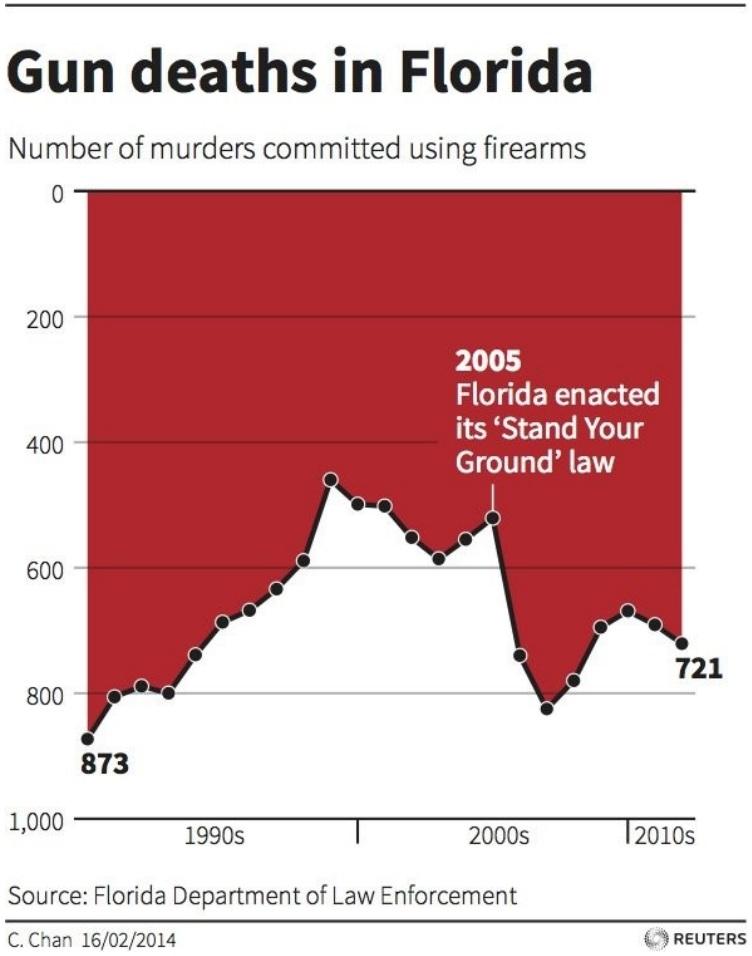
NOT IDEAL



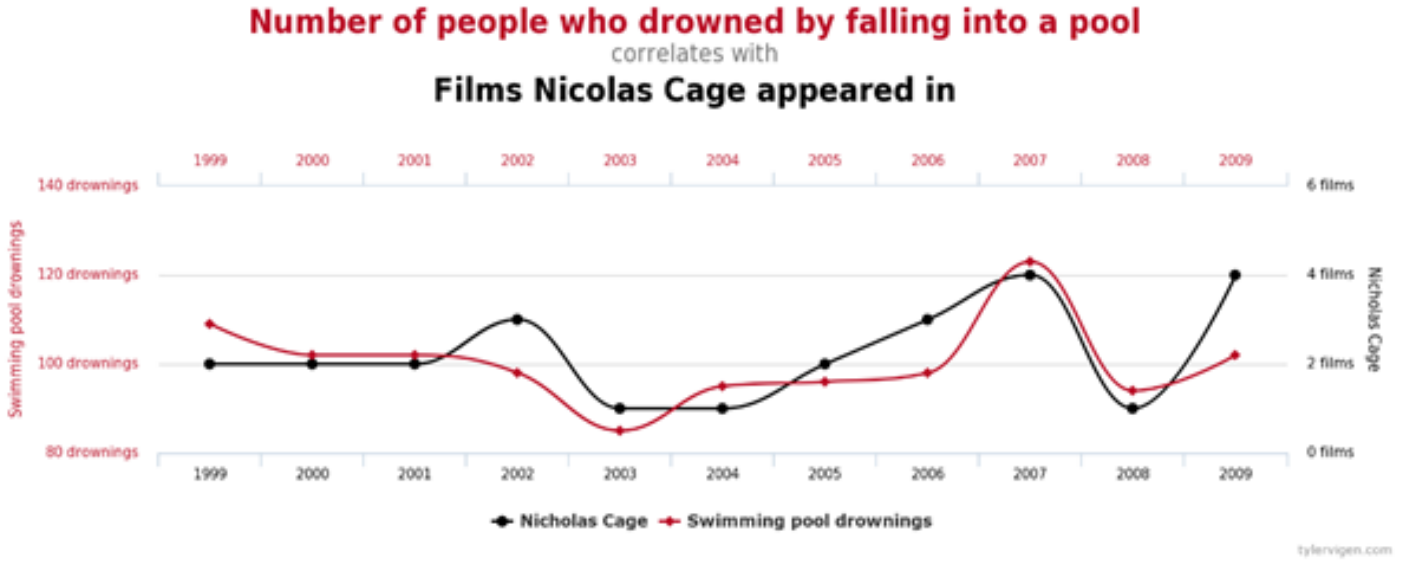
BETTER



Manipulation durch Datenvisualisierung



Manipulation durch Datenvisualisierung



DHBW Lörrach

Pipeline der Datenvisualisierung

Von Merkmal zu Variablen

Merkmal: Isolierte Eigenschaft eines größeren Ganzen, z.B. Intelligenz, Farbe, Einkommen

Ausprägung: Zustand des Merkmals, z.B. IQ =115, Farbe = Rot, Einkommen = hoch

Eine **Variable** wird definiert, indem den Ausprägungen des Merkmals **Zahlen** zugeordnet werden.

Diese Zahlen heißen **Realisationen** oder **Werte.**

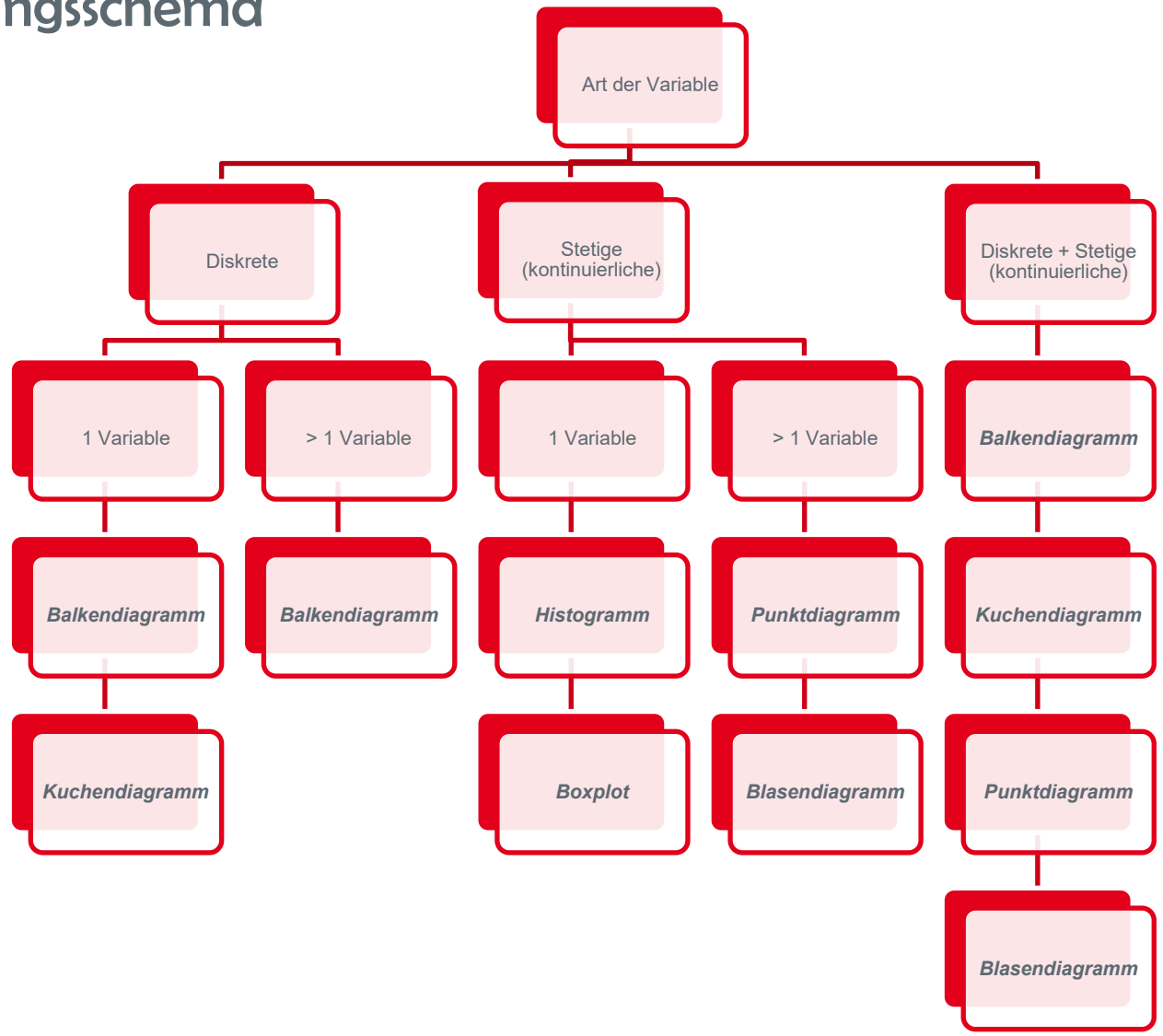
Variablen

Eine **diskrete** Variable besitzt zumeist endlich viele und feste Werte, die man über Ganzzahlen beschreiben kann:

- **Dichtome** Variablen haben genau zwei diskrete Werte
- **Polytome** Variablen haben mehr als zwei diskrete Werte

Eine **stetige (kontinuierliche)** Variable kann (unendlich viele) beliebige Werte annehmen, die man über reelle Zahlen beschreibt

Datenvisualisierungsschema



Plotly

Warum Plotly

- Interaktive ope-source Framework für Datenvisualisierung
- kann verschiedene Arten von Diagrammen und Grafiken leicht (low-code) erstellen
- optisch tolles Design
- unendliche Anpassungsmöglichkeiten der Diagramme
- Großes Community

Infos und Beispiele

R: *https://plotly.com/r*

DHBW Lörrach

Syntax für alle Plotly Grafiken

```
plot_ly(data = <Daten>,
        x = ~<x-Achsenvariable>,
        y = ~<y-Achsenvariable>,
        z = ~<z-Achsenvariable>,
        type = '<Diagrammtyp>',
        mode = '<Modus>',
        color = ~<Farbvariable>,
        colors = <Farbschema>,
        symbol = ~<Symbolvariable>,
        size = <Punktgröße>,
        text = ~<Tooltiptext>,
        hoverinfo = '<Hover-Info>',
        marker = list(<Marker-Einstellungen>),
        line = list(<Linien-Einstellungen>),
        name = '<Legendeneintrag>',
        opacity = <Transparenz>,
        showlegend = <TRUE/FALSE>)
```

DHBW Lörrach

Syntax für alle Plotly Grafiken

- `data`: Das Datenset, das zur Erstellung des Diagramms verwendet wird.
- `x` und `y`: Variablen für die x- und y-Achsen. Sie sind in fast allen Diagrammtypen erforderlich.
- `z`: Wird für 3D-Diagramme benötigt, um die z-Achsenwerte anzugeben.
- `type`: Bestimmt den Typ des Diagramms. Einige gängige Typen sind:
  - `'scatter'` (Streudiagramm oder Linienplot)
  - `'bar'` (Balkendiagramm)
  - `'box'` (Boxplot)
  - `'histogram'` (Histogramm)
  - `'scatter3d'` (3D-Streudiagramm)
  - `'surface'` (3D-Oberflächendiagramm)
  - `'pie'` (Tortendiagramm)
  - `'heatmap'` (Heatmap)
  - `'line'` (Linienplot)
  - `'area'` (Flächendiagramm)
  - `'bubble'` (Blasendiagramm)

DHBW Lörrach

Syntax für alle Plotly Grafiken

- `mode` : Definiert die Darstellungsweise bei Streudiagrammen:
  - `'markers'` für Punkte
  - `'lines'` für Linien
  - `'lines+markers'` für eine Kombination aus beidem
- `color` : Variable, die verwendet wird, um die Farbe der Datenpunkte oder Balken zu bestimmen.
- `colors` : Gibt das Farbschema an, das für das Diagramm verwendet wird, wie z. B. `'Viridis'` oder `'Blues'` .
- `symbol` : Variable, die zur Festlegung der Symbolform in Streudiagrammen verwendet wird.
- `size` : Bestimmt die Größe der Marker in Streudiagrammen.
- `text` : Text, der angezeigt wird, wenn der Benutzer über einen Punkt fährt (Tooltip).
- `hoverinfo` : Steuerung, welche Informationen im Tooltip angezeigt werden, z. B. `'x+y+text'` .

DHBW Lörrach

Syntax für alle Plotly Grafiken

- `marker` : Liste zur Anpassung der Marker (Datenpunkte), wie:
  - `color` : Farbe der Marker
  - `size` : Größe der Marker
  - `line` : Randfarbe und -breite der Marker
- `line` : Liste zur Anpassung der Linien, wie:
  - `color` : Farbe der Linie
  - `width` : Breite der Linie
  - `dash` : Stil der Linie (z. B. `'solid'` , `'dash'` , `'dot'`)
- `name` : Text, der als Bezeichnung in der Legende des Diagramms angezeigt wird.
- `opacity` : Wert zwischen 0 und 1, der die Transparenz des Diagramms steuert (1 = vollständig sichtbar, 0 = vollständig transparent).
- `showlegend` : Steuert, ob die Legende im Diagramm angezeigt wird (`TRUE` oder `FALSE`).

Syntax für alle Plotly Grafiken

DHBW Lörrach

```
layout(
  title = '<Diagrammtitel>',
  xaxis = list(
    title = '<x-Achsentitel>',
    showgrid = <TRUE/FALSE>,
    zeroline = <TRUE/FALSE>,
    showline = <TRUE/FALSE>
  ),
  yaxis = list(
    title = '<y-Achsentitel>',
    showgrid = <TRUE/FALSE>,
    zeroline = <TRUE/FALSE>,
    showline = <TRUE/FALSE>
  ),
  legend = list(
    title = list(text = '<Legendentitel>'),
    x = <x-Position>,
    y = <y-Position>,
    orientation = '<horizontal/vertical>'
  ),
  plot_bgcolor = '<Diagrammhintergrundfarbe>',
  paper_bgcolor = '<Hintergrundfarbe>',
  showlegend = <TRUE/FALSE>,
  barmode = '<stack/group>'
)
```

Syntax für alle Plotly Grafiken

DHBW Lörrach

- `title`: Legt den Titel des Diagramms fest. Sie können auch zusätzliche Formatierungen wie Schriftgröße und Farbe definieren.
- `xaxis` und `yaxis`: Listen von Parametern zur Anpassung der Achsen.
  - `title`: Titel der Achse.
  - `showgrid`: Steuerung, ob das Gitter angezeigt wird (`TRUE` oder `FALSE`).
  - `zeroline`: Anzeige der Nulllinie, die die Nullposition markiert.
  - `showline`: Anzeige einer Linie entlang der Achse.
- `legend`: Anpassung der Legende des Diagramms.
  - `title`: Titel der Legende.
  - `x` und `y`: Position der Legende im Diagramm (Koordinaten zwischen 0 und 1).
  - `orientation`: Ausrichtung der Legende (`horizontal` oder `vertical`).

Syntax für alle Plotly Grafiken

DHBW Lörrach

- `plot_bgcolor`: Hintergrundfarbe des Diagrammbereichs.
- `paper_bgcolor`: Hintergrundfarbe des gesamten Diagrammpapiers.
- `showlegend`: Steuerung, ob die Legende angezeigt wird (`TRUE` oder `FALSE`).
- `barmode`: Definiert die Art der Anordnung der Balken in einem Balkendiagramm.
  - Werte: `'stack'` für gestapelte Balken oder `'group'` für gruppierte Balken.

DHBW Lörrach

Vielen Dank!