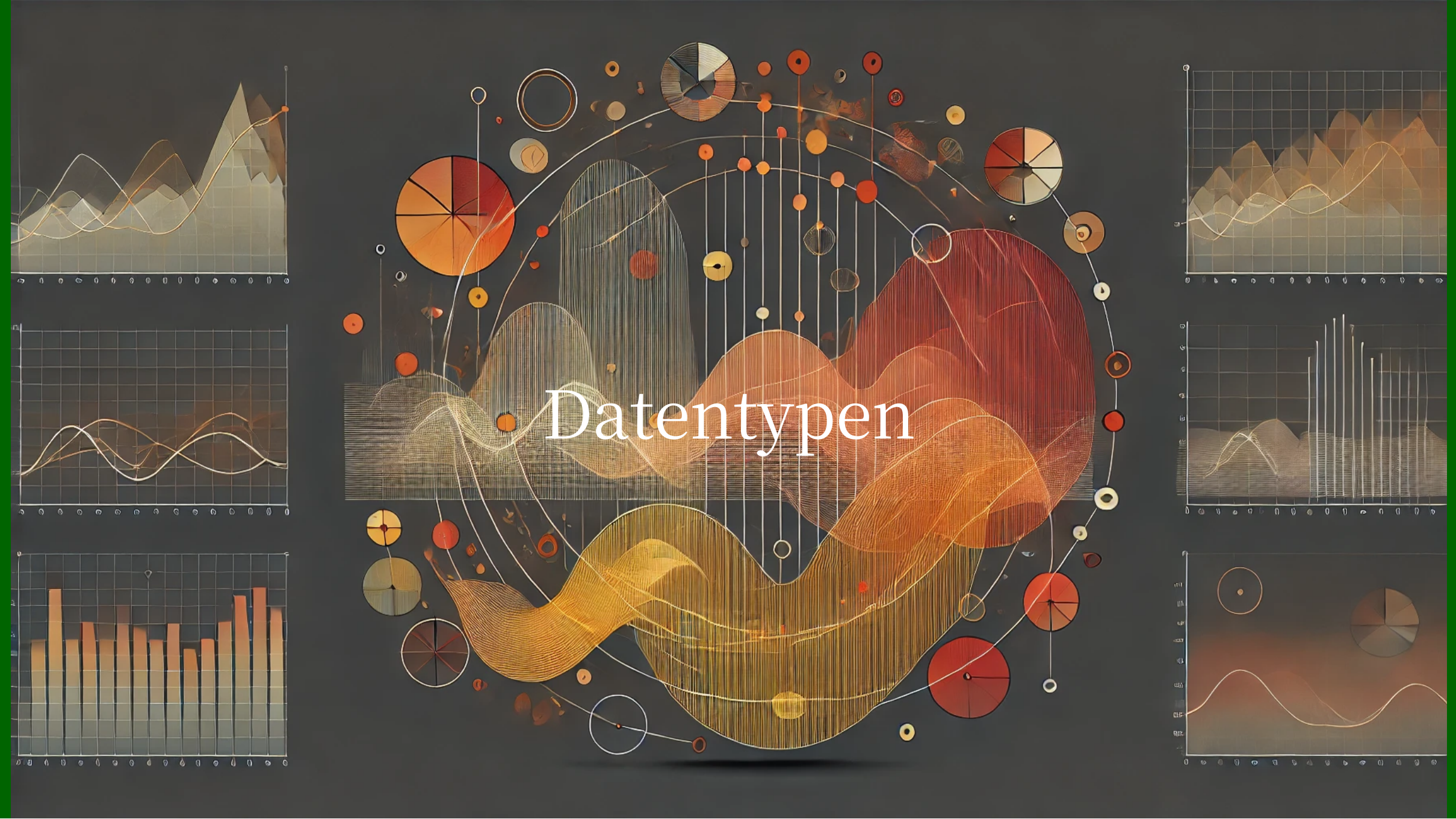
Datentypen

Verschiedene Datentypen in R

R unterscheidet fünf verschiedene, sogenannte atomare Datentypen.

- Buchstaben, z.B. Namen (sogenannte "Character")
- Ganzzahlige Werte ("Integer")
- Numerische Werte ("Numeric")
- Faktor Variablen ("Factor")
- Boolsche Variablen ("Boolean"), d.h. logische Werte wie z.B. "WAHR"

Die Funktion `class()` ist hilfreich wenn Sie den Datentyp einer Variable herausfinden wollen.

Die `class` Funktion

```
a <- 5
b <- "Bruttoinlandsprodukt"
class(a)
```

```
[1] "numeric"
```

```
class(b)
```

```
[1] "character"
```

- R speichert grundsätzlich alle Zahlen als numerische Variablen
- Wenn speziell "Integers" gespeichert werden sollen wird dies mit angehängtem `L` erreicht (d.h. `a <- 5L`).

Data Frame

- In der Praxis sind in Datensätzen unterschiedliche Datentypen vermischt
- Werden in R als sogenannte Data Frames abgespeichert
  - Diese sind auf keinen Datentyp festgelegt
  - Zweidimensionale Tabelle
- Beispielhaft am Datensatz `gapminder` aus dem `gapminder` Paket
  - Herunterladen und einlesen mittels:

```
#install.packages("gapminder")
library(gapminder)
data("gapminder")
class(gapminder)
```

```
[1] "tbl_df"     "tbl"        "data.frame"
```

Erster Blick auf die Daten

- `class` Funktion -> Um welchen Datentyp handelt es sich?
- `glimpse` Funktion -> Übersicht über die wichtigsten Eigenschaften des Datensatzes:
  - Anzahl der Beobachtungen und Variablen
  - Name jeder Variablen
  - Kodierung jeder Variablen
  - Auszug der ersten Einträge pro Variable
- `str` Funktion -> base R Variante von `glimpse`
- Nähere Infos zum Datensatz und den Variablen über Hilfe-Seiten (`?gapminder`)

Erster Blick auf die Daten

- Alternativen zu `str` sind die Funktionen `head()` und in RStudio `View`

```
head(gapminder)
```

```
# A tibble: 6 × 6
  country     continent  year lifeExp      pop gdpPercap
  <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
1 Afghanistan Asia       1952    28.8  8425333      779.
2 Afghanistan Asia       1957    30.3  9240934      821.
3 Afghanistan Asia       1962    32.0 10267083      853.
4 Afghanistan Asia       1967    34.0 11537966      836.
5 Afghanistan Asia       1972    36.1 13079460      740.
6 Afghanistan Asia       1977    38.4 14880372      786.
```

```
View(gapminder)
```

- `dim` Anzahl der Variablen und Beobachtungen
- `attributes` gibt Spalten- und Reihennamen aus, sowie den Datentyp

Variablen im Datensatz

- `names` zeigt die Namen der Variablen im Datensatz
- Um eine bestimmte Variable auszuwählen nutzen wir folgende Struktur: `Datensatzname$Variablenname`
- Hier ein Beispiel:

```
names(gapminder)
gapminder$country
```

Ausgabe: Vektor aller in der Variablen enthaltenen Werte

- Sortierung gleich wie im Data Frame

Vektoren: Numerische, alphabetische und logische Informationen

Numerische und logische Vektoren

- Durch `gapminder$lifeExp` wird ein Zahlenvektor ausgegeben
- Jede Spalte eines Data Frame ist ein Vektor
- Anzahl der Einträge über Funktion `length` bestimmen

```
length(gapminder$lifeExp)
```

```
[1] 1704
```

Vektoren können nicht nur numerisch oder alphabetisch sein, sondern auch logische Werte enthalten.

- Logische Vektoren entweder WAHR (TRUE) oder FALSCH (FALSE)

```
z <- 3 == 2
z
```

```
[1] FALSE
```

```
class(z)
```

```
[1] "logical"
```

Relationale Operatoren

Relationale Operatoren prüfen ob bestimmte Beziehungen gelten:

- Der *relationale Operator* == im vorherigen Beispiel evaluiert ob 3 gleich 2 ist.
- Alle weiteren *relationalen Operatoren* finden Sie in R mit Hilfe der folgenden Suche:

```
?Comparison
```

Bitte beachten: Der Befehl = weist einer Variablen einen bestimmten Wert zu. Bitte **vermeiden** Sie die Zuordnung mittels **=** und verwenden Sie <– um einer Variablen einen bestimmten Wert zuzuordnen.

Sehen Sie hierzu auch den Coding Style Guide von Google und im tidyverse

Relationale Operatoren in R

Operator	Definition
`<`	kleiner als
`<=`	kleiner als oder gleich
`>`	größer als
`>=`	größer als oder gleich
`==`	exakt gleich
`!=`	ungleich
`x & y`	`x` UND `y`
`x|y`	`x` ODER `y`
`is.na(x)`	teste ob `x` ist gleich `NA`
`!is.na(x)`	teste ob `x` ist nicht gleich `NA`
`x %in% y`	teste ob `x` in `y` ist
`!(x %in% y)`	teste ob `x` nicht in `y` ist
`!x`	nicht `x`

Vektoren erzeugen

In R selbst lassen sich folgendermaßen Vektoren erzeugen:

```
vektor <- c(15, 20, 25) # numerischer Vektor
vektor
```

```
[1] 15 20 25
```

```
länder <- c("Deutschland","Schweden","Südafrika") # alphabetischer Vektor
```

- `c()` steht für concatenate, d.h. *verbinden*

Vektoren benennen

Weiterhin können die Einträge von Vektoren benannt werden:

```
länder <- c("Deutschland" = 15, "Schweden" = 20, "Südafrika" = 25)
class(länder)
```

```
[1] "numeric"
```

```
names(länder)
```

```
[1] "Deutschland" "Schweden"    "Südafrika"
```

Auch bestehende Vektoren können benannt werden:

```
vektor <- c(15, 20, 25)
länder <- c("Deutschland","Schweden","Südafrika")
names(vektor) <- länder
vektor
```

```
Deutschland    Schweden   Südafrika
         15          20          25
```

Reihen

Eine weitere wichtige Möglichkeit Vektoren zu generieren sind Reihen:

```
seq(1, 10)
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

```
1:10
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

- Erstes Argument: Start der Reihe
- Zweites Argument: Ende der Reihe
- Default: Schritte erhöhen sich um 1
- Möglichkeit die Größe der Schritte zu definieren:

```
seq(1, 10, 0.5)
```

```
 [1]  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5  6.0  6.5  7.0  7.5  8.0
[16]  8.5  9.0  9.5 10.0
```

Die Unterteilung

- Einzelne Elemente eines Vektors mit quadratischen Klammern ansteuern

```
vektor[2]
```

```
Schweden
      20
```

- Mehrere Elemente durch verketteten Index

```
vektor[c(1,3)]
```

```
Deutschland   Südafrika
         15          25
```

Die Unterteilung

- Mehrere aufeinander folgende Elemente

```
vektor[1:2]
```

```
Deutschland    Schweden
         15          20
```

- Benannte Elemente

```
vektor["Deutschland"]
```

```
Deutschland
         15
```

Typumwandlung

- Wichtig zu verstehen, wie R Datentypen intern behandelt
- Bevor R eine Fehlermeldung ausgibt stellt es Vermutungen an, was mit der Eingabe gemeint war

Beispiel:

Ein Vektor kann nur gleiche Datentypen beinhalten, bspw. nur numerische oder nur alphabetische Einträge.

Somit sollte folgende Eingabe eine Fehlermeldung produzieren:

```
x <- c(1, "Schweden", 3)
```

- Weder Warn- noch Fehlermeldung

Typumwandlung

- Die Funktion `class` gibt Aufschluss warum

```
x
```

```
[1] "1"         "Schweden" "3"
```

```
class(x)
```

```
[1] "character"
```

- Augen auf bei der Programmierung!
  - Unbeabsichtigte Typumwandlungen können zu Fehlern führen
  - Sehr mühsam herauszufinden
  - Fallen oft sehr spät auf

Typumwandlung erzwingen

In R kann die Typumwandlung auch erzwungen werden:

```
x <- 1:5
y <- as.character(x)
y
```

```
[1] "1" "2" "3" "4" "5"
```

Mit `as.numeric` können Sie diese Umwandlung in die andere Richtung erzwingen.

```
as.numeric(y)
```

```
[1] 1 2 3 4 5
```

Wichtig: In einigen Datensätzen werden Zahlen als Buchstaben gespeichert, hier kann die Funktion `as.numeric` sehr hilfreich sein.

> Welche Klasse hat der folgende Vektor `test <- c(12, "Kanada", TRUE)`?

Nicht verfügbare Werte (NA)

Falls R keine Vermutung hat:

- Datenpunkt wird als `NA` deklariert.
- `NA` steht für "not available"

```
x <- c("1", "b", "3")
as.numeric(x)
```

```
[1]  1 NA  3
```

- Hier gibt es eine Warnmeldung

In echten Datensätzen gibt es oft einige `NA`!

Faktorvariablen

Faktorvariablen scheinen aus Buchstaben zu bestehen, werden von R intern jedoch als "Integers" behandelt.

Pros:

- ✚ Sind in einigen Analysen notwendig um kategorische Daten darzustellen (Regressionen, Grafiken etc.)
- ✚ Sind selbsterklärend durch Labels und damit nachvollziehbar
- ✚ Kompakte Form um Datensatz darzustellen

Contra:

- ✚ Sorgen oft für Verwirrung und werden mit *Characters* verwechselt, verhalten sich jedoch ganz anders wie diese

Faktorvariablen

Die Variable "continent" ist beispielsweise als Faktorvariable abgespeichert:

```
class(gapminder$continent)
```

```
[1] "factor"
```

```
is.factor(gapminder$continent)
```

```
[1] TRUE
```

```
levels(gapminder$continent)
```

```
[1] "Africa"   "Americas" "Asia"     "Europe"   "Oceania"
```

```
table(gapminder$continent)
```

```
  Africa Americas     Asia   Europe  Oceania 
     624      300      396      360       24 
```

Basislevel bei Faktorvariablen

Wenn Faktorvariablen in Regressionen verwendet werden, dann wird oft ein sogenanntes Basislevel herangezogen. In der Regression werden dann die Ergebnisse für die anderen Gruppen in der Faktorvariablen immer im Vergleich zu diesem Basislevel berechnet.

- Wenn Sie nun lieber eine andere Gruppe als Basislevel auswählen möchten (hier die Europ)a, so können Sie dies folgendermaßen machen:

```
gapminder$continent <- fct_relevel(gapminder$continent, "Europe")
levels(gapminder$continent)
```

```
[1] "Europe"   "Africa"   "Americas" "Asia"     "Oceania"
```

```
table(gapminder$continent)
```

```
  Europe   Africa Americas     Asia  Oceania 
     360      624      300      396       24 
```

Das Paket `forcats`, aus welchem die Funktion `fct_relevel` stammt, hilft bei Faktorvariablen ungemein weiter