Data Wrangling 2

Data Tidying

Daniela Palleschi

2023-06-06

Inhaltsverzeichnis

Wiederholung

Letzte Woche haben wir gelernt, wie man…

- Code-Chunk-Optionen verwendet
- formatierte Tabellen druckt
- Abbildungsunterschriften hinzufügt
- die Größe von Abbildungen kontrolliert
- Querverweise erstellet

Die Übungen der letzten Woche

- Ich hatte einige Tippfehler in meiner Anleitung :(
  - `include: false` sollte `message: false` sein
  - `fig-out: 6` sollte `fig-width: 6` sein
- es gab einige allgemeine Probleme
  - z.B. das Setzen von globalen Optionen
  - `label`-Formatierung
  - fehlende Querverweise

Heutige Ziele

Heute werden wir…

- lernen wir etwas über breite und lange Daten
- breite Daten länger machen
- lange Daten breiter machen

Lust auf mehr?

- Ch. 6 (Data tidying) in Wickham et al. (o. J.)
- Ch. 8 (Data Tidying) in Nordmann & DeBruine (2022)

1 Einrichtung

- Pakete: `tidyverse`, `here`

```
pacman::p_load(tidyverse,
               here)
```

- Daten (in `daten` Ordner):
  - `table1.csv`
  - `df_billboard.csv` (neu)
  - `cms_patient_experience.csv` (neu)

```
df_tb <- read_csv(here("daten", "table1.csv"))
df_billboard <- read_csv(here("daten", "df_billboard.csv"))
df_cms <- read_csv(here("daten", "cms_patient_experience.csv"))
```

2 Aufgeräumter Arbeitsablauf (Tidy workflow)

- wir haben gelernt, wie man Daten importiert (`readr::read_csv`), transformiert (`dplyr` Paket) und visualisiert (`ggplot` Paket)
- Bisher haben wir immer mit aufgeräumten Daten gearbeitet
  - so dass wir den Schritt "tidy" nicht durchführen mussten

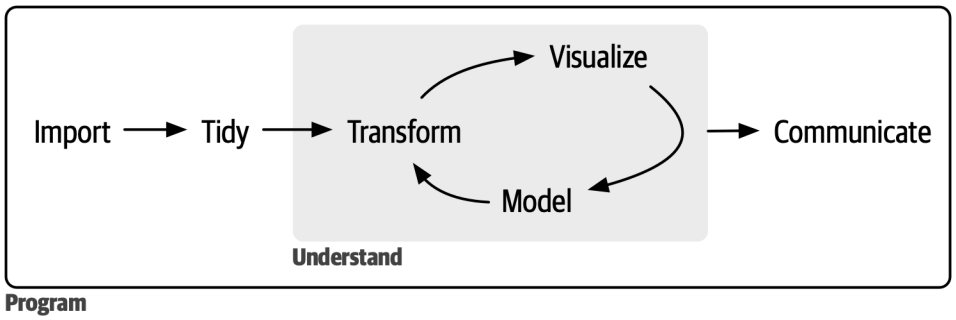

Abbildung 1: Image source: Wickham et al. (o. J.) (all rights reserved)

3 Aufgeräumte Daten (tidy data)

- dieselben Daten können auf verschiedene Weise dargestellt werden
- Die folgenden Datensätze zeigen alle dieselben Werte für vier Variablen: `country`, `year`, `popuplation`, und `cases` (Anzahl) der Tuberkulosefälle
  - jeder Datensatz ordnet die Werte anders an

Tabelle 1: Tabelle 1

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

Tabelle 2: Tabelle 2

country	year	type	count
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

Tabelle 3: Tabelle 3

country	year	rate
Afghanistan	1999	745/19987071
Afghanistan	2000	2666/20595360
Brazil	1999	37737/172006362
Brazil	2000	80488/174504898
China	1999	212258/1272915272
China	2000	213766/1280428583

- was ist am einfachsten zu lesen?

Drei Regeln für aufgeräumte Daten:

1. Jede Variable ist eine Spalte, jede Spalte ist eine Variable
2. Jede Beobachtung ist eine Zeile, jede Zeile ist eine Beobachtung
3. Jeder Wert ist eine Zelle, jede Zelle ist ein Einzelwert

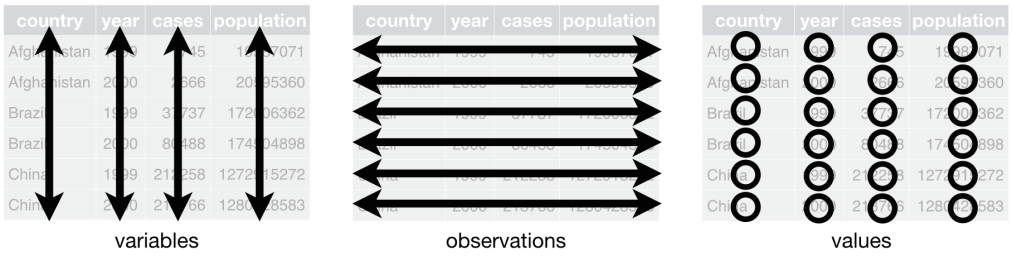

Abbildung 2: Image source: Wickham et al. (o. J.) (all rights reserved)

- was als Beobachtung oder als Variable gilt, hängt oft von der jeweiligen Aufgabe ab

3.1 Why tidy data?

> "**Happy families** are all alike; every **unhappy family** is unhappy in its own way."
> — Leo Tolstoy

> "**Tidy datasets** are all alike, but every **untidy dataset** is untidy in its own way."
> — Hadley Wickham

- Die Bereinigung der Daten erfordert im Vorfeld einige Arbeit, ist aber auf lange Sicht hilfreich
- wenn wir einmal aufgeräumte Daten haben, werden wir weniger Zeit damit verbringen, die Daten in die richtige Form zu bringen, um das zu tun, was wir wollen

---

Die Arbeit mit aufgeräumten Daten hat zwei wesentliche Vorteile:

1. Die Arbeit mit einer konsistenten Datenstruktur ermöglicht es uns, Konventionen zu übernehmen
    - Da aufgeräumte Daten die allgemein anerkannte Datenstruktur sind, basieren die Konventionen auf der Annahme dieser Struktur.
    - Werkzeuge haben also eine zugrundeliegende Einheitlichkeit
2. Die vektorisierte Natur von R kann glänzen
    - Die meisten eingebauten R-Funktionen arbeiten mit *Vektorwerten*.
    - Alle Pakete im `tidyverse` sind für die Arbeit mit Tidy-Daten konzipiert

---

Vectors

Vektoren sind der grundlegendste Datenobjekttyp in R. Ein Vektor enthält Daten desselben Typs und ist im Wesentlichen eine Liste. Wir können einen Vektor zum Beispiel mit der Funktion `c()` erstellen.

```
vector1 <- c(2, 3, 4, 6, 7)
vector2 <- c(2, 3, 4, 6, "c")
```

`vector1` wird numerische Werte enthalten, da alle Elemente Zahlen sind. `vector2` enthält alle Zeichenwerte (d.h. Text), da es ein einziges eindeutiges Zeichenelement (`c()`) gibt. R liest also alle Elemente als Zeichentyp. Wir können einen Datenrahmen aus Vektoren gleicher Länge erstellen, indem wir z. B. die Funktion `tibble()` verwenden.

```
tibble(vector1,vector2)
```

```
# A tibble: 5 x 2
```

```
  vector1 vector2
    <dbl> <chr>
1       2 2
2       3 3
3       4 4
4       6 6
5       7 c
```

- die meisten Daten "in the wild" sind unordentlich
  - Die Daten werden häufig zunächst für ein anderes Ziel als die Analyse organisiert
    - Dieses Ziel ist in der Regel die Dateneingabe: Wir wollen unsere Beobachtungen zunächst einfach dokumentieren können
  - die meisten Menschen sind nicht mit den Grundsätzen ordentlicher Daten vertraut, und erst nachdem sie *viel* Zeit mit Daten verbracht haben, wird klar, warum ordentliche Daten notwendig sind
- Dies bedeutet, dass die meisten *echten* Analysen zumindest ein gewisses Maß an Aufräumen erfordern.

Aufgabe 3.1: Tidy data

Beispiel 3.1.

1. Geht zurück zu den Tabellen 1-3. Beschreibt für jede Tabelle, was jede Beobachtung und jede Spalte darstellt.
2. Überlegt euch, wie ihr die Rate für Tabelle 1 berechnen würdet. Ihr braucht nur ein einziges Verb, das:
   - eine neue Variable (nennt sie `rate`) erzeugt, die enthält:
     - die Anzahl der TB-`cases` pro Land und Jahr, geteilt durch
     - die entsprechende `population` pro Land und Jahr,
     - multipliziert mit `10000`
   - Hinweis: Welches `dplyr`-Verb erzeugt neue Variablen? (Schaue in Woche 5 nach)
3. Schaut euch die Tabellen 2 und 3 an. Wäre es so einfach gewesen, die "Rate" mit diesen Datenstrukturen zu berechnen?

3.2 Daten aufräumen

- Umformung
  - z.B. von breiten zu langen Daten
- Ergebnis:
  - jede Spalte = eine Variable
  - jede Zeile = eine Beobachtung

3.2.1 Datenaufräumung mit dem `tidyverse`

- mit dem `tidyr`-Paket
  - `pivot_longer()`: macht breite Daten länger
  - `pivot_wider()`: lange Daten breiter machen
- wir müssen oft zwischen diesen Formaten konvertieren, um verschiedene Arten von Zusammenfassungen oder Visualisierungen zu erstellen

---

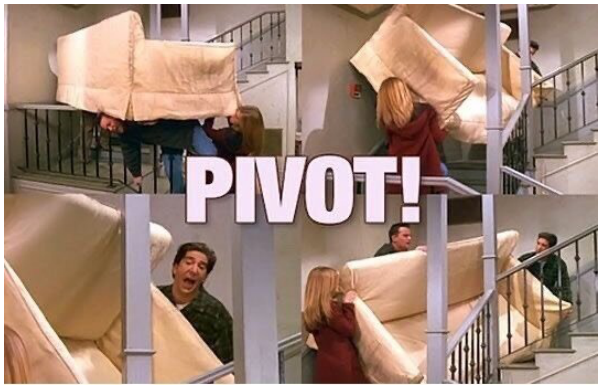

Abbildung 3: die berühmteste Verwendung des Wortes Pivot (zumindest für Millenials)

3.2.2 Breite versus lange Daten

- Breite Daten: alle Beobachtungen zu einer Sache stehen in derselben Zeile
  - breite Daten sind *normalerweise* nicht 'tidy'
- lange Daten: jede Beobachtung steht in einer eigenen Zeile
  - lange Daten sind *in der Regel* 'tidy'

Tabelle 4: df_billboard rank of songs in the year 2000

artist	track	date_entered	wk1	w
2 Pac	Baby Don't Cry (Keep...	2000-02-26	87	
2Ge+her	The Hardest Part Of ...	2000-09-02	91	
3 Doors Down	Kryptonite	2000-04-08	81	
3 Doors Down	Loser	2000-10-21	76	
504 Boyz	Wobble Wobble	2000-04-15	57	
98^0	Give Me Just One Nig...	2000-08-19	51	
A*Teens	Dancing Queen	2000-07-08	97	
Aaliyah	I Don't Wanna	2000-01-29	84	
Aaliyah	Try Again	2000-03-18	59	
Adams, Yolanda	Open My Heart	2000-08-26	76	

4 Verlängern von Daten: `df_billboard`

- in dem Datensatz `billboard.csv`
  - jede Zeile ist ein Lied
  - die ersten drei Spalten sind Variablen, die den Song beschreiben (`artist`, `track`, `date_entered`)
  - wir haben 76 Spalten (`wk1-wk76`), die den Rang des Liedes in dieser Woche beschreiben
    - Die Spaltennamen sind eine Variable (die `week`), und die Zellwerte sind eine andere Variable (der `rank`)

```
df_billboard %>%
  head(n = 10) %>%
  knitr::kable() %>%
  kableExtra::kable_styling(font_size = 20)
```

- Sind die Daten in Tabelle 4 aufgeräumt?

- Sind diese Daten zu breit oder zu lang?
- Wie können wir diese Daten bereinigen?

4.1 `pivot_longer()`

- `pivot_longer()`
  - wandelt eine breite Datentabelle in ein längeres Format um, indem es die Überschriften der angegebenen Spalten in die Werte neuer Spalten umwandelt
  - und die Werte dieser Spalten in einer neuen komprimierten Spalte kombiniert

```
df_billboard_tidy <- df_billboard %>%
  pivot_longer(
    cols = starts_with("wk"),
    names_to = "week",
    values_to = "rank"
  )
```

- `col =` gibt an, welche Spalten gedreht werden müssen (d.h. welche *nicht* Variablen sind)
  - hat die gleiche Syntax wie `select()`, wir könnten also z.B. `starts_with("")` verwenden
- `names_to =` benennt die Variable, die in den aktuellen Spaltennamen gespeichert ist, hier ist es `week`
- `values_to =` benennt die in den Zellwerten gespeicherte Variable, die wir `rank` nennen
- N.B., wir mussten `week` und `rank` in Anführungszeichen setzen, weil sie noch keine Variablennamen sind

4.1.1 Entfernen fehlender Werte (`NA`)

- wir haben ein paar `NA` (fehlende) Werte; dies geschah, wenn ein Song nicht in den Top 100 war
  - Mit dem Argument `values_drop_na = TRUE/FALSE` werden geschwenkte Zeilen gelöscht, die keinen Wert für die neue Variable haben.

```
df_billboard_tidy <- df_billboard %>%
  pivot_longer(
    cols = starts_with("wk"),
    names_to = "week",
    values_to = "rank",
    values_drop_na = TRUE
```

Tabelle 5: A pivoted version of `df_billboard` (first 10 rows)

artist	track	date_entered	week	rank
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk1	87
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk2	82
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk3	72
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk4	77
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk5	87
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk6	94
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk7	99
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk8	NA
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk9	NA
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk10	NA

```
  )
```

- Wie viele Zeilen weniger haben wir jetzt?

4.1.2 Parsing Zahlen

- die Daten sind jetzt ordentlich, aber `week` enthält immer noch `wk` in den Werten
- Das Paket `readr` hat eine praktische Funktion: `parse_number()` extrahiert die erste Zahl aus einer Zeichenkette und ignoriert alle anderen Texte
  - Wie könnten wir `parse_number()` benutzen, um die Variable `week` zu *ändern*?

```
df_billboard_tidy <- df_billboard_tidy %>%
  mutate(week = parse_number(week))
```

4.1.3 Arbeiten mit Datumsangaben

- Das `tidyverse` hat auch ein Paket, das die Arbeit mit Datumsangaben erleichtert: `lubridate`

Tabelle 6: Tidy `df_billboard` data with dropped `NA` values

artist	track	date_entered	week	rank
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk1	87
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk2	82
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk3	72
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk4	77
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk5	87
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk6	94
2 Pac	Baby Don't Cry (Keep...	2000-02-26	wk7	99
2Ge+her	The Hardest Part Of ...	2000-09-02	wk1	91
2Ge+her	The Hardest Part Of ...	2000-09-02	wk2	87
2Ge+her	The Hardest Part Of ...	2000-09-02	wk3	92

Tabelle 7: Tidy `df_billboard` data with dropped `NA` values

artist	track	date_entered	week	rank
2 Pac	Baby Don't Cry (Keep...	2000-02-26	1	87
2 Pac	Baby Don't Cry (Keep...	2000-02-26	2	82
2 Pac	Baby Don't Cry (Keep...	2000-02-26	3	72
2 Pac	Baby Don't Cry (Keep...	2000-02-26	4	77
2 Pac	Baby Don't Cry (Keep...	2000-02-26	5	87
2 Pac	Baby Don't Cry (Keep...	2000-02-26	6	94

- die Variable `date_entered` hat das Format Jahr-Monat-Tag (`ymd`)
- Wir können `lubridate` Verben verwenden, um das `year`, den `month` und den `day` zu extrahieren

```
df_billboard_tidy <- df_billboard_tidy %>%
  mutate(
    month = month(date_entered, label = F),
    day = day(ymd(date_entered)),
    year = year(ymd(date_entered))
  )
```

4.1.4 Plotten unserer aufgeräumten Daten

- Jetzt haben wir die `week`-Daten in einer Variablen und die `rank` Daten in einer anderen Variablen
- Versuchen wir nun, die Ränge der Plakate nach Woche darzustellen

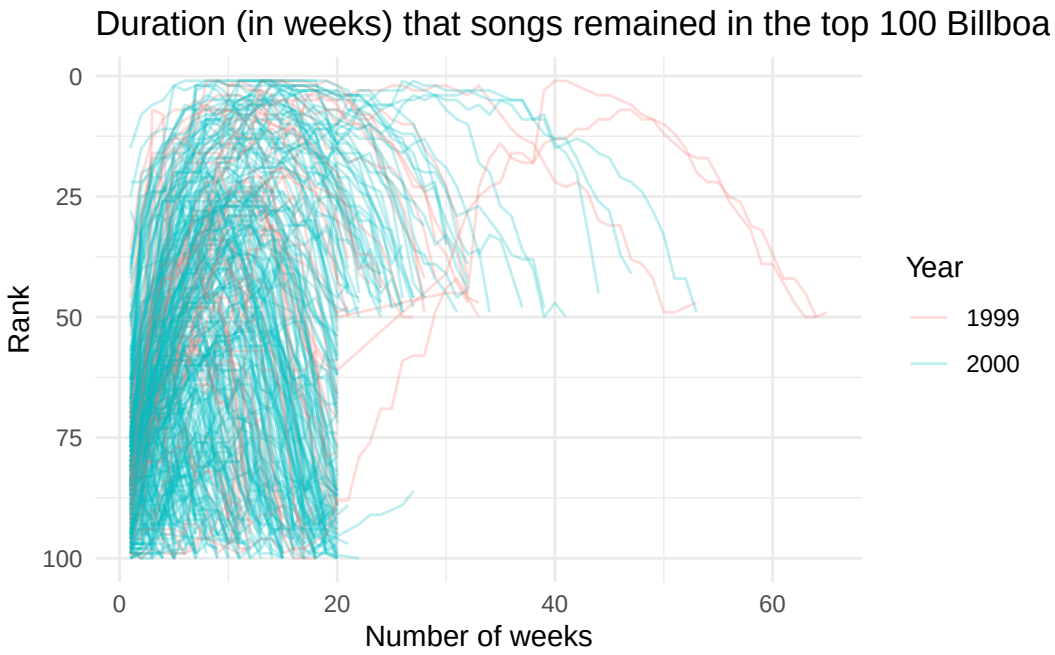

Abbildung 4: Billboard ranks by number of weeks for songs that were in the top 100 in the year 2000

- Wie viele Wochen war ein Lied am längsten in den Top 100?
- Warum ist 100 unten und 0 oben?

- Warum ist das untere rechte Viertel leer? Was bedeutet das?

Aufgabe 3.1: Tidy data

Beispiel 4.1.

1. Erstellt Abbildung 4 neu.

5 Widening data: `df_cms`

- `pivot_wider()` macht Datensätze *breiter*, indem es Spalten vergrößert und Zeilen verkleinert
  - Dies ist hilfreich, wenn eine Beobachtung über mehrere Zeilen verteilt ist.
  - Diese Art von Daten ist in der freien Wildbahn nicht sehr häufig, aber in Regierungsdaten ist sie ziemlich verbreitet.

- der Datensatz `cms_patient_experience.csv` enthält Daten über Patientenerfahrungen von den Centers of Medicare and Medicaid Services
- die untersuchte Kerneinheit ist eine Organisation (gespeichert in `org_pac_id` und `org_nm`), aber jede Organisation (d.h. Beobachtung) nimmt 6 Zeilen ein
  - eine Zeile für jede Maßnahme (d. h. Variable)
  - Wir wollen also, dass diese Variablen in Spalten dargestellt werden

5.1 `pivot_wider()`

- nimmt lange Daten und macht sie breiter
- benötigt einige Argumente:
  - `id_cols`: identifizierende Spalten
  - `names_from`: wie soll die neue Spalte heißen, die die bisherigen Spaltennamen enthält?
  - `names_prefix`: Präfix für die neuen Spaltennamen
  - `values_from`: neue Spaltenwerte

---

org_pac_id	org_nm	measure_cd
0446157747	USC CARE MEDICAL GROUP INC	CAHPS_GE
0446157747	USC CARE MEDICAL GROUP INC	CAHPS_GE
0446157747	USC CARE MEDICAL GROUP INC	CAHPS_GE
0446157747	USC CARE MEDICAL GROUP INC	CAHPS_GE
0446157747	USC CARE MEDICAL GROUP INC	CAHPS_GE
0446157747	USC CARE MEDICAL GROUP INC	CAHPS_GE

- Wie können wir für jede `org`,
  - die sechs Werte aus `measure_cd` nehmen und sechs neue Variablen `names` daraus erstellen,
  - und die `values` aus `prf_rate` nehmen?
- das Ergebnis sollte wie Tabelle 8 aussehen

```
cms_patient_experience %>%
  pivot_wider(
    id_cols = starts_with("org"),
    names_from = measure_cd,
    values_from = prf_rate
  ) %>%
  head() %>%
  knitr::kable() %>%
  kableExtra::kable_styling(font_size = 20)
```

- Tabelle 9 zeigt die ersten 6 Zeilen des Originaldatensatzes
- Tabelle 10 zeigt die ersten 6 Zeilen des erweiterten Datensatzes
- Wie werden die Daten aus Tabelle 9 in Tabelle 10 dargestellt?

6 Aufgabe

1. Durchsucht die Titel. Wählt einen Titel, der euch gefällt (oder einen zufälligen Titel), und dann

Tabelle 8: Wider `df_cms` data (measure variable to columns with rating as values)

org_pac_id	org_nm
0446157747	USC CARE MEDICAL GROUP INC
0446162697	ASSOCIATION OF UNIVERSITY PHYSICIANS
0547164295	BEAVER MEDICAL GROUP PC
0749333730	CAPE PHYSICIANS ASSOCIATES PA
0840104360	ALLIANCE PHYSICIANS INC
0840109864	REX HOSPITAL INC

Tabelle 9: Original `cms_patient_experience.csv` format

org_pac_id	org_nm	measure_cd
0446157747	USC CARE MEDICAL GROUP INC	CAHPS_GI
0446157747	USC CARE MEDICAL GROUP INC	CAHPS_GI
0446157747	USC CARE MEDICAL GROUP INC	CAHPS_GI
0446157747	USC CARE MEDICAL GROUP INC	CAHPS_GI
0446157747	USC CARE MEDICAL GROUP INC	CAHPS_GI
0446157747	USC CARE MEDICAL GROUP INC	CAHPS_GI

Tabelle 10: Widened `cms_patient_experience.csv` format

org_pac_id	org_nm
0446157747	USC CARE MEDICAL GROUP INC
0446162697	ASSOCIATION OF UNIVERSITY PHYSICIANS
0547164295	BEAVER MEDICAL GROUP PC
0749333730	CAPE PHYSICIANS ASSOCIATES PA
0840104360	ALLIANCE PHYSICIANS INC
0840109864	REX HOSPITAL INC

- filtert die Daten so, dass sie nur diesen Titel enthalten, und dann
- Erstellt ein Liniendiagramm der Zeit des Liedes in den Top 100 Billboard
- Bezieht sich auf das Diagramm und beschreibt den Trend des Liedes (dazu solltet euch die Daten des Liedes ansehen).

Ein Beispiel: *Abbildung 5 zeigt den Trend von 'Say My Name' von Destiny's Child, das am 25. Dezember 1999 (`date_entered`) auf Platz 83 (`rank` für `wk1`) in die Charts einstieg und 32 Wochen lang in den Top 100 blieb (z.b., `max(week)`).*

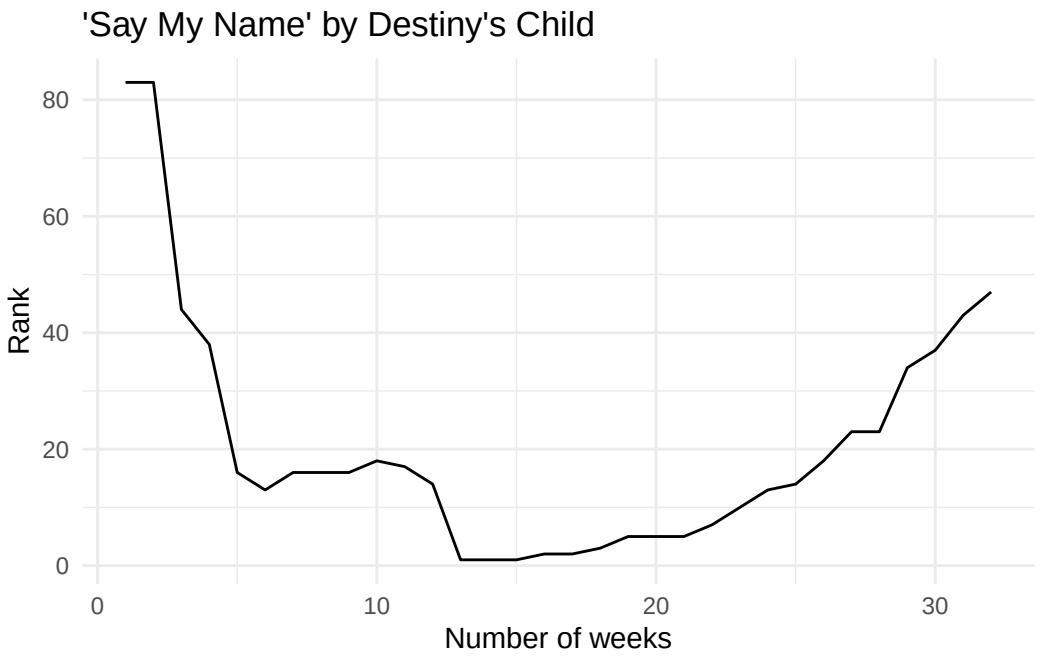

Abbildung 5: Example linegraph for 'Say My Name' by Destiny's Child

---

2. Lädt den Datensatz `biondo_etal_2021_tidy.csv` (Teilmenge der Daten von Biondo et al. (2022)) und speichert ihn als `df_biondo`

- druckt eine formatierte Tabelle (mit `knitr::kable()`) mit einem `Label` und einer Beschriftung des `Kopfes()` der Daten

---

3. Verlängere die Daten so, dass die Spalten `rt` und `tt` in einer Spalte stehen:
    - Die Variable `Namen` sollte in eine neue Variable mit dem Namen `Maß` übergehen.

    - Die Variable "Werte" geht in eine neue Variable "ms" (für Millisekunden) über.
    - Speichern der verlängerten Daten als `df_biondo_long`

- druckt eine formatierte Tabelle (mit `knitr::kable()`) mit einem `Label` und einer Beschriftung

---

4. Erweitert `df_biondo_long` so, dass die Spalten `rt` und `tt` wieder in ihren eigenen Spalten sind

    - die `id_cols` sollten `subj` und `item` sein
    - die Variable `Namen` sollte `von` `Maß` kommen
    - die Variable `values` sollte von `ms` (für Millisekunden) stammen
    - die verlängerten Daten als `df_biondo_wide` speichern
    - eine formatierte Tabelle (mit `knitr::kable()`) mit einem `Label` und einer Beschriftung ausgeben
    - Tipp: `df_biondo_wide` sollte genau dasselbe sein wie `df_biondo`.

Heutige Ziele

Heute haben wir...

- etwas über breite und lange Daten lernen
- breite Daten länger machen
- lange Daten breiter machen

Session Info

Hergestellt mit R version 4.3.0 (2023-04-21) (Already Tomorrow) und RStudioversion 2023.3.0.386 (Cherry Blossom).

```
sessionInfo()
```

```
R version 4.3.0 (2023-04-21)
Platform: aarch64-apple-darwin20 (64-bit)
Running under: macOS Ventura 13.2.1

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
```

```
LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib;  l

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: Europe/Berlin
tzcode source: internal

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] here_1.0.1      lubridate_1.9.2 forcats_1.0.0   stringr_1.5.0
 [5] dplyr_1.1.2     purrr_1.0.1     readr_2.1.4     tidyr_1.3.0
 [9] tibble_3.2.1    ggplot2_3.4.2   tidyverse_2.0.0

loaded via a namespace (and not attached):
 [1] utf8_1.2.3           generics_0.1.3       xml2_1.3.4
 [4] stringi_1.7.12       hms_1.1.3            digest_0.6.31
 [7] magrittr_2.0.3       evaluate_0.21        grid_4.3.0
[10] timechange_0.2.0     fastmap_1.1.1        rprojroot_2.0.3
[13] jsonlite_1.8.4       httr_1.4.6           rvest_1.0.3
[16] fansi_1.0.4          viridisLite_0.4.2    scales_1.2.1
[19] cli_3.6.1            rlang_1.1.1          crayon_1.5.2
[22] bit64_4.0.5          munsell_0.5.0        withr_2.5.0
[25] yaml_2.3.7           tools_4.3.0          parallel_4.3.0
[28] tzdb_0.4.0           colorspace_2.1-0     webshot_0.5.4
[31] pacman_0.5.1         kableExtra_1.3.4.9000 png_0.1-8
[34] vctrs_0.6.2          R6_2.5.1             lifecycle_1.0.3
[37] magick_2.7.4         bit_4.0.5            vroom_1.6.3
[40] pkgconfig_2.0.3      pillar_1.9.0         gtable_0.3.3
[43] glue_1.6.2           Rcpp_1.0.10          systemfonts_1.0.4
[46] xfun_0.39            tidyselect_1.2.0     rstudioapi_0.14
[49] knitr_1.42           farver_2.1.1         htmltools_0.5.5
[52] labeling_0.4.2       svglite_2.1.1        rmarkdown_2.21
[55] compiler_4.3.0
```

Literaturverzeichnis

Biondo, N., Soilemezidi, M., & Mancini, S. (2022). Yesterday Is History, Tomorrow Is a Mystery: An Eye-Tracking Investigation of the Processing of Past and Future Time Reference during Sentence Reading. *Journal of Experimental Psychology: Learning,*

Memory, and Cognition, *48*(7), 1001–1018. https://doi.org/10.1037/xlm0001053

Nordmann, E., & DeBruine, L. (2022). *Applied Data Skills* (Version 2.0). Zenodo. https://doi.org/10.5281/zenodo.6365078

Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. (o. J.). *R for Data Science* (2. Aufl.). https://r4ds.hadley.nz/