

ana1-UG-Rang-Häufigkeit (ana1-UG-rank-freq)

Analyse der Unigramme nach ihrer Rang-Häufigkeit. Unterteilung in drei Programme:

- Einfache Analyse der Rang-Häufigkeit
- Vergleich der Rang-Häufigkeitsverteilung mit der Mandelbrot-Zipf-Kurve
- Vergleich von zwei Datensets bzgl. der Rang-Häufigkeit ihrer Unigramme

ana1a-UG-rank-freq.R

Dieses Programm

(A) Nutzt das Programm 4a-combined-data.R um alle gewünschten annotierten Stücke¹ zu einer grossen Tabelle zu kombinieren und diese zu bereinigen (siehe 4-Bericht)

(B) Kann nach Tongeschlecht der lokalen Tonartabschnitte filtern durch:

```
59 oval_harmonic_tab <- subset(final_combined_data, localkey_is_minor == 0)
```

(C) Erstellt anschliessend ein Dataframe (chord_freq_df), welches jedem Akkord (chord) seine absolute Häufigkeit (absolute_frequency), den Häufigkeitsrang (rank) und die relative Häufigkeit (relative_frequency) zuordnet.

(D) Mit dem Paket 'ggplot2' lassen sich Rang-Häufigkeits Diagramme ausgeben. Sinnvollerweise sollten hier beide Achsen mit \log_{10} skaliert werden. Die Grenzen werden am besten auf der y-Achse bei 0.00005 – 0.18 und auf der x-Achse bei 1 – 1000 gewählt. (Abb. 1 und 2)

ana1b-UG-rank-freq.R

Dieses Programm ist gleich zu ana1a-UG-rank-freq.R (A-C) und erlaubt es zusätzlich die erhaltenen Werte im Dataframe (chord_freq_df) mit der Mandelbrot-Zipf-Kurve zu vergleichen (C2).

Die Mandelbrot-Zipf-Kurve wird definiert als: $\tilde{f}(r) = \frac{a}{(b+r)^c}$

Mit der Funktion 'nls' aus dem Paket 'stats' eine Optimierung der Variablen a , b und c durchgeführt über der Methode der kleinsten Quadrate. Dies ermöglicht eine Maximierung des Bestimmtheitsmass R^2 . Dies folgendermassen:

```
59 zipf_fit <- nls(chord_freq_df$relative_frequency ~ zipf_function(chord_freq_df$rank, a, b, c),
60               data = chord_freq_df,
61               start = initial_params)
```

Führt man die beschriebene Optimierung durch mit den Dur-Abschnitten aller Beethoven-sonaten erhält man:

```
> summary(zipf_fit)
Parameters:
  Estimate Std. Error t value Pr(>|t|)
a 0.52530 0.03474 15.12 <2e-16 ***
b 1.67828 0.09469 17.72 <2e-16 ***
c 1.33423 0.02286 58.38 <2e-16 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 0.0009972 on 837 degrees of freedom
Number of iterations to convergence: 7
Achieved convergence tolerance: 1.993e-06

> print(paste("R^2 =", r_quad))
[1] "R^2 = 0.980712560621019"
```

Führt man sie durch mit den Dur-Abschnitten aller Beethovenstreicherquartette erhält man:

```
> summary(zipf_fit)
Parameters:
  Estimate Std. Error t value Pr(>|t|)
a 0.37397 0.01985 18.84 <2e-16 ***
b 0.94534 0.06851 13.80 <2e-16 ***
c 1.22358 0.01968 62.16 <2e-16 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 0.001151 on 789 degrees of freedom
Number of iterations to convergence: 8
Achieved convergence tolerance: 1.652e-06

> print(paste("R^2 =", r_quad))
[1] "R^2 = 0.980323671672286"
```

Es lassen sich analog zu ana1a-UG-rank-freq.R Rang-Häufigkeits Diagramme ausgeben (D).

ana1c-UG-rank-freq-Compare.R

Dieses Programm ist ähnlich zu ana1a-UG-rank-freq.R wobei es ermöglicht die Rang-Häufigkeits Verteilungen von zwei Datensätzen zu vergleichen...

ana1c-UG-rank-freq- Compare-three.R

Dieses Programm ist ähnlich zu ana1c-UG-rank-freq-Compare.R wobei es ermöglicht die Rang-Häufigkeits Verteilungen von drei Datensätzen zu vergleichen. Ein Datensatz wird als Basisdatensatz (base) festgelegt, die anderen beiden als Vergleichsdatsatz 1 (dev1) und Vergleichsdatsatz 2 (dev2). Es besteht die Möglichkeit nach Tongeschlecht der lokalen Tonart zu filtern:

```
41 tongeschlecht <- 0
..
84 oval_harmonic_tab <- subset(final_combined_data, localkey_is_minor == tongeschlecht)
```

Es werde für jeden Datensatz Rang-Häufigkeits Dataframes und alle diese Dataframes dann in einem grossen kombiniert (chord_freq_df). Dabei wird sich am Basisdatensatz (base) orientiert. Entsprechend werden für die Kombination jeweilige Zeilen angefügt. Weiter werden auch die Deltas der relativen Häufigkeiten zwischen dem Basisdatensatz (base) und den Vergleichsdatsätzen (dev1 und dev2) berechnet.

Für die Darstellung wird wieder 'ggplot2' verwendet. Hierbei sollen jedoch nur die Rang-Häufigkeits-Werte in \log_{10} Skalierung dargestellt werden. Daher werden zwei Achsen erstellt: Die "Standard"-Achse und die skalierte Achse. Für die Skalierung wird daher die Funktion definiert:

$$\xi: \mathbb{R}^+ \rightarrow \mathbb{R}^+ \text{ mit } x \mapsto \frac{\log_{10}(x)+2.5}{6}.$$

$$\text{Umkehrfunktion: } \xi^{-1}: \mathbb{R}^+ \rightarrow \mathbb{R}^+ \text{ mit } x \mapsto 10^{6x-2.5}.$$

Da in R, die Werte nur bzgl. "Standard"-Achse dargestellt werden können (obwohl sich eine zweite, skalierte Achse anzeigen lässt) müssen die Werte, die bzgl. skalierten Achse angezeigt werden sollen, durch die Umkehrfunktion. Dazu werden weitere Spalten erstellt:

```
262 xi <- function(x) {
263   return((log10(x) + 2.5) / 6)
264 }
265
266 xi_minus <- function(x) {
267   return(10^( 6x - 2.5 ))
268 }
..
275 chord_freq_df$delta_relative_frequency_dev1_xi_minus <-
      xi_minus(chord_freq_df$delta_relative_frequency_dev1)
..
279 chord_freq_df$delta_relative_frequency_dev2_xi_minus <-
      xi_minus(chord_freq_df$delta_relative_frequency_dev2)
```

Wir wollen sicherstellen, dass alle Werte grösser 0 sind, damit wir keine Probleme mit der logarithmischen Achse bekommen: Wir sehen $\xi^{-1}(x) = 10^{6x-2.5} > 0$ gilt aber $\forall x \in \mathbb{R}$.

¹ Die Tabellen müssen, damit sie von den Skripten erkannt und eingelesen werden können, wie folgt benannt werden:

[KOM]-[A][xx]-M[x].tsv

wobei KOM für das Komponistenkürzel steht (LVB für L. v. Beethoven, WAM für W. A. Mozart), A für die Stückart (S für Sonate, Q für Streicherquartett) mit der üblichen Nummerierung (xx aus 00-99) und M für den Satz mit Nummer (x aus 0-9)

Die Datei zum dritten Satz der ersten Beethovensonate heisst also: LVB-S01-M3.tsv