07
Multiple Lineare Regression

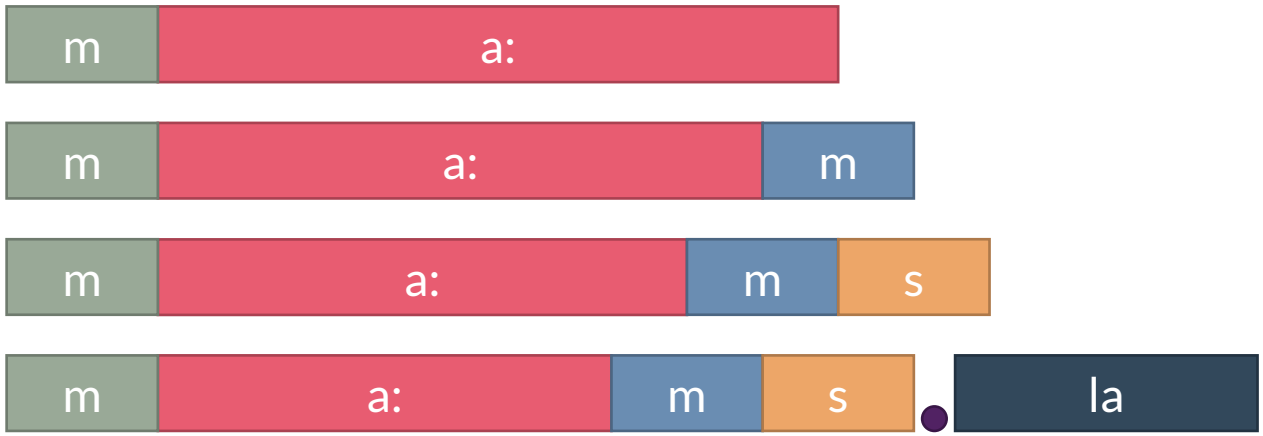
Dominic Schmitz & Janina Esser

Beispieldaten

- Für die folgenden Beispiele werden wir Daten folgender Studie nutzen:

Compensatory Vowel Shortening in German[1]

- Stressed Vowels sind kürzer je nachdem wie viele Konsonanten ihnen folgen:



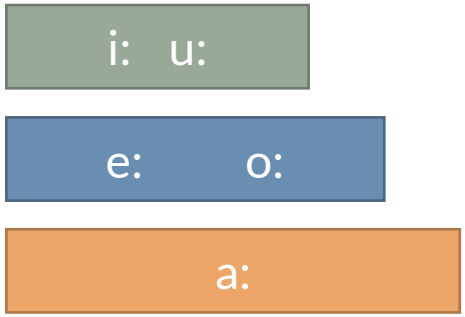
[1] Schmitz, D., Cho, H.-E., & Niemann, H. (2018). Vowel shortening in German as a function of syllable structure. Proceedings 13. Phonetik Und Phonologie Tagung (P&P13), 181–184.

Beispieldaten

- Für die folgenden Beispiele werden wir Daten folgender Studie nutzen:

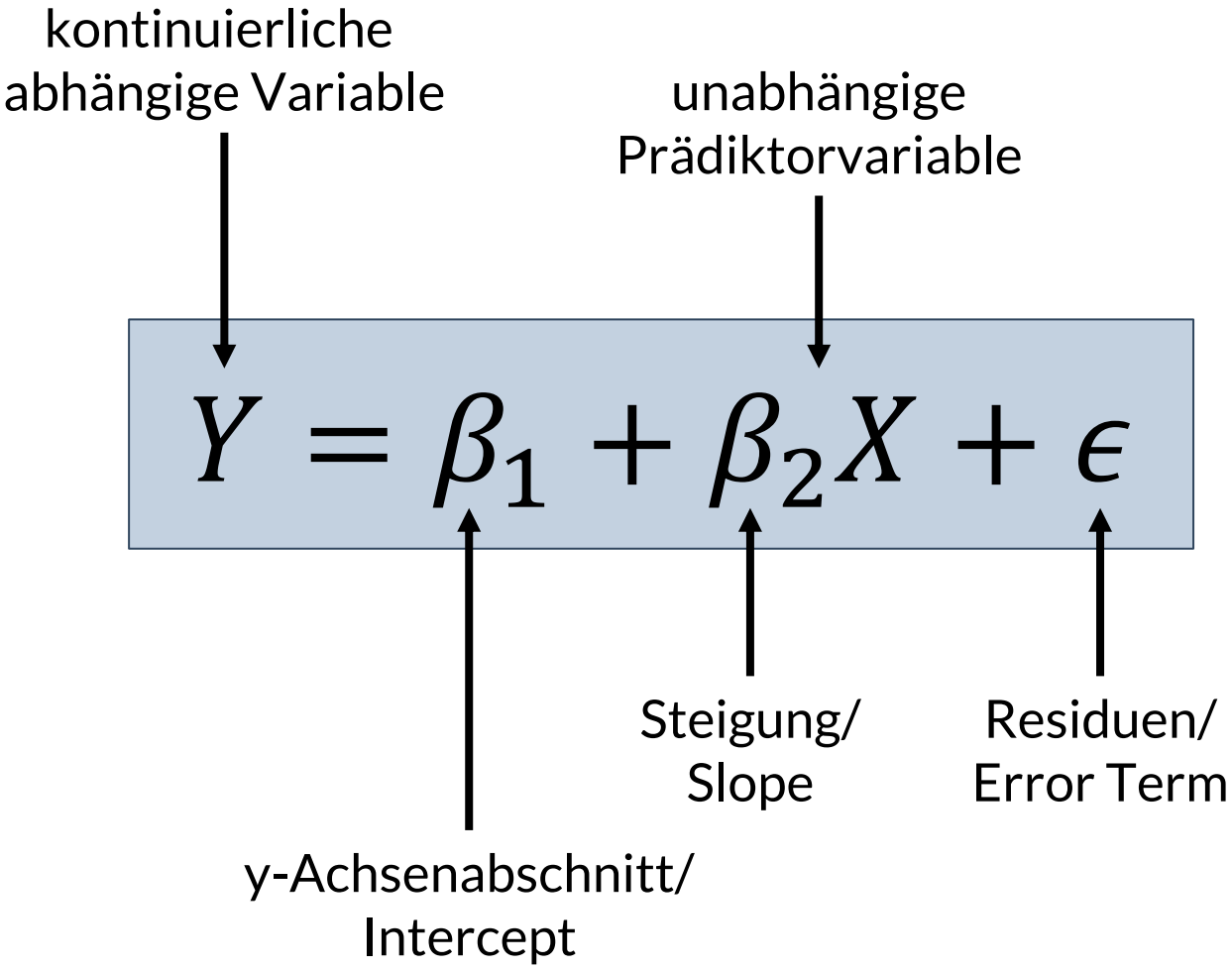
Compensatory Vowel Shortening in German[1]

- Unabhängig von diesem Vowel Shortening gilt, dass offene Vokale länger sind als halb-offene Vokale, und halb-offene Vokale sind länger als geschlossene Vokale:

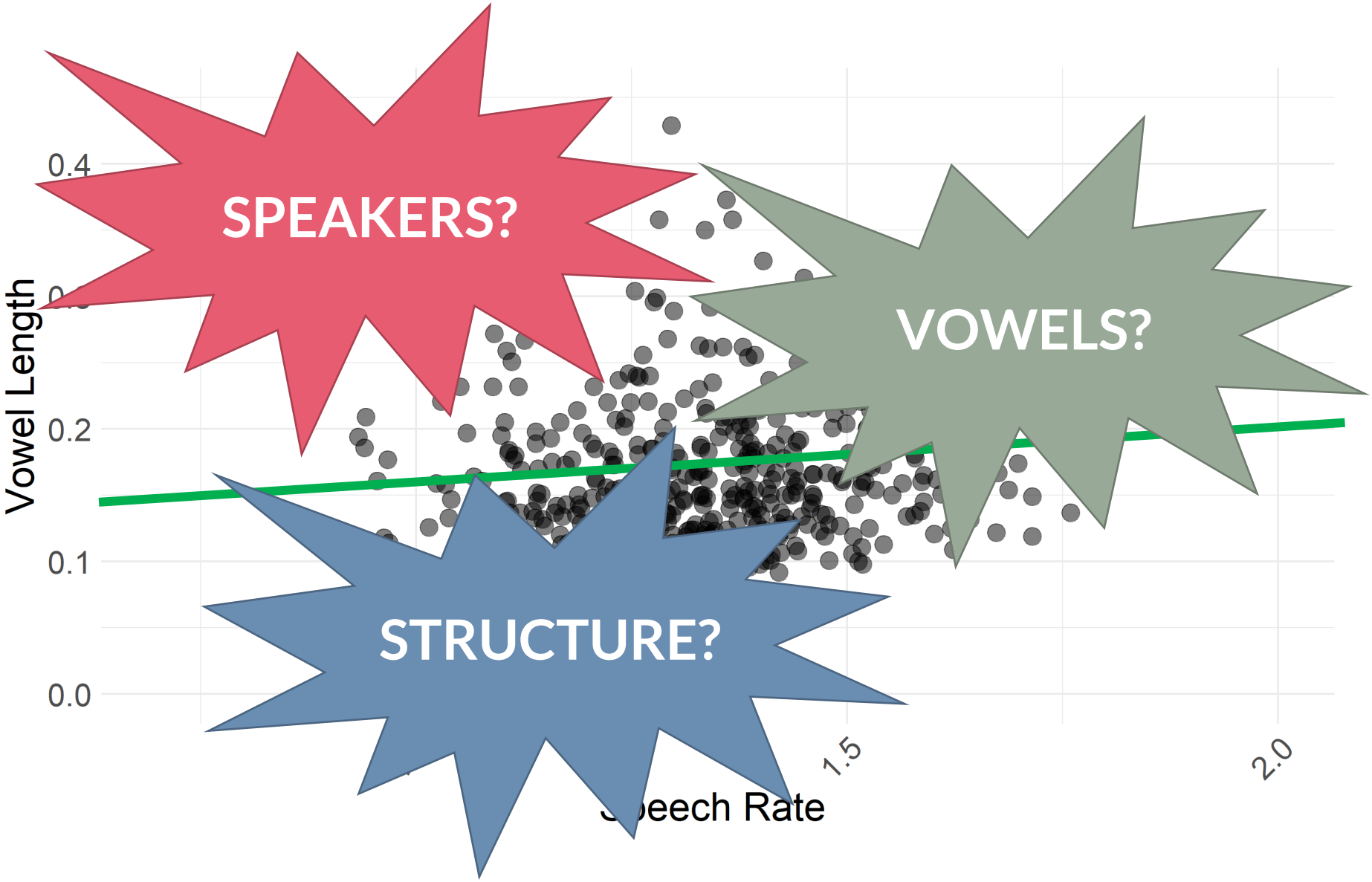


[1] Schmitz, D., Cho, H.-E., & Niemann, H. (2018). Vowel shortening in German as a function of syllable structure. Proceedings 13. Phonetik Und Phonologie Tagung (P&P13), 181–184.

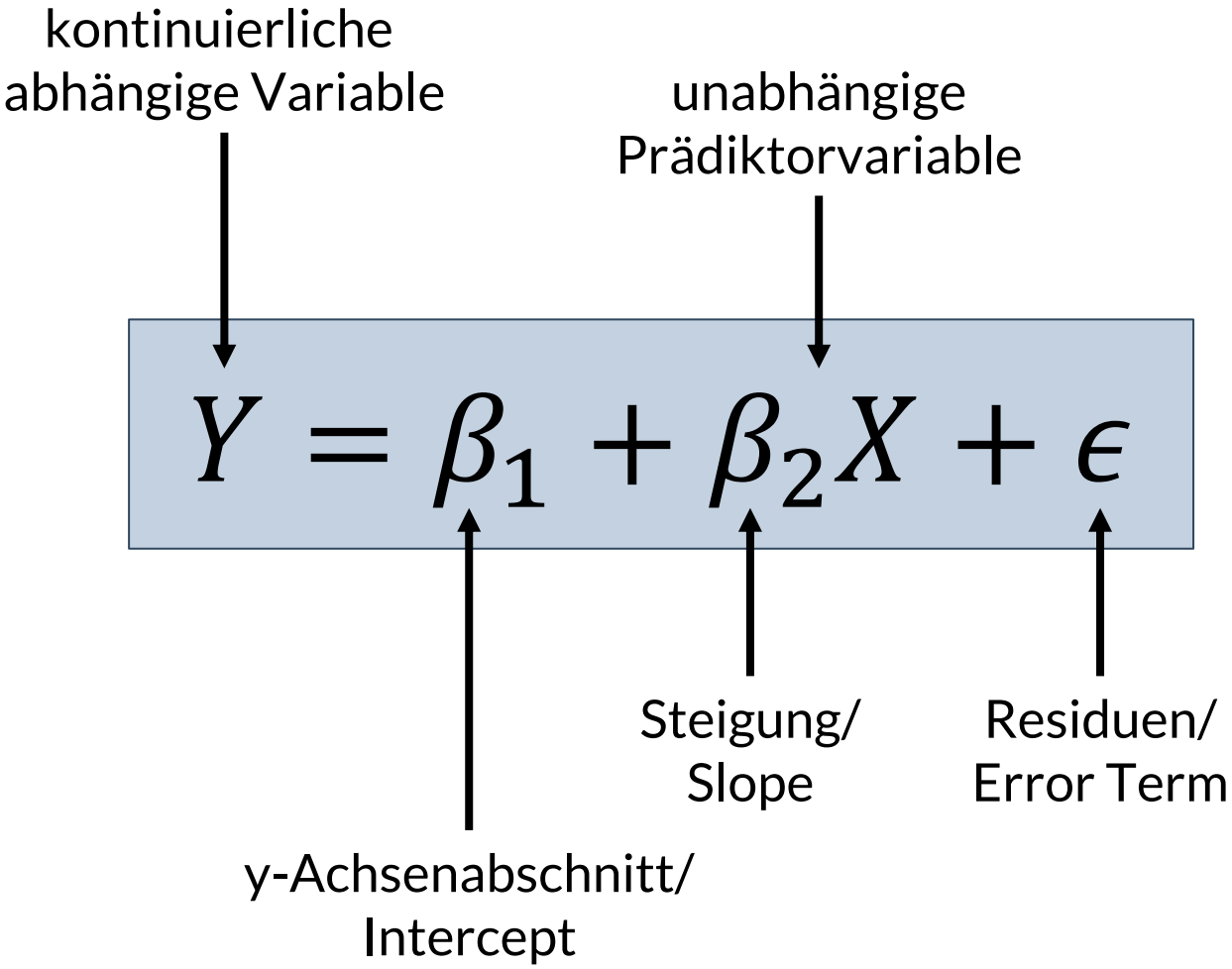
Einfache Lineare Regression



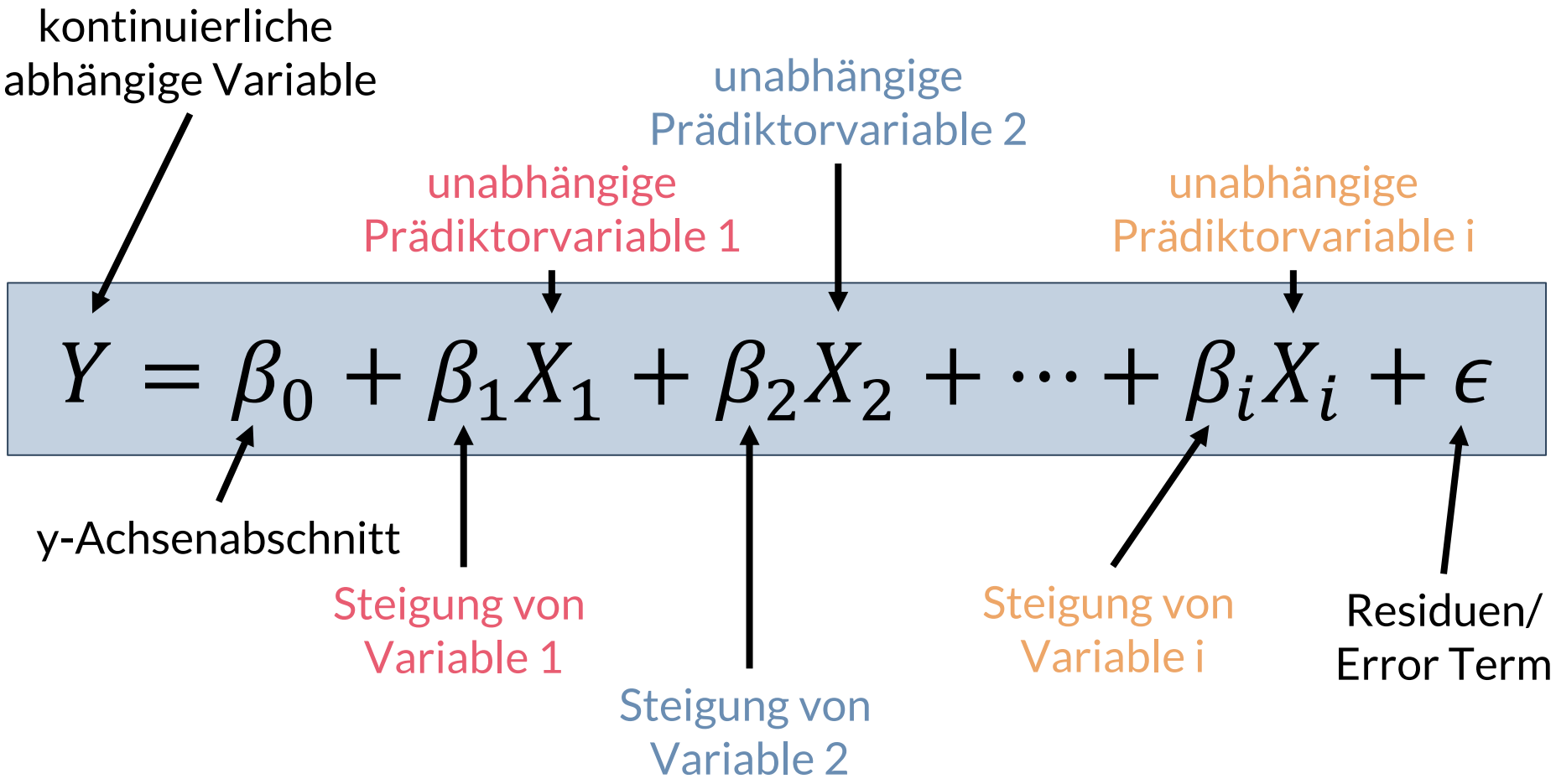
(Zu) Einfache Lineare Regression



Einfache Lineare Regression



Multiple Lineare Regression: Formel



Multiple Lineare Regression in R

- Mehr Variablen = mehr Zeitaufwand
- Typische Schritte bei Multipler Linearer Regression sind
  1. Verteilung der abhängigen Variable überprüfen
  2. „volles“ Modell erstellen
  3. „bestes“ Modell finden
  4. Annahmen überprüfen
  5. Modell interpretieren

1. Verteilung der abhängigen Variable

- Wie wir bereits wissen, nutzen wir hierzu den **Shapiro-Wilk Test**
- Die abhängige Variable in unserem Beispiel, `duration`, ist nicht normalverteilt
- Daher nutzen wir wieder eine **log-transformierte Version** der Variable, `durationLog`

2. „Volles“ Modell

- Unsere abhängige Variable ist `durationLog`
- Als nächstes müssen wir die **unabhängigen Variablen** identifizieren, die wir nutzen möchten
- In diesem Beispiel sind es die folgenden Variablen:
  - `structure` = coda structure
  - `vowel` = vowel quality
  - `rate` = speech rate
  - `number` = slide number during experiment

2. „Volles“ Modell

- Erstellen des „vollen“ Modells:

```
model = lm(durationLog ~ structure +
                         vowel +
                         rate +
                         number,
                       data)
```

3. „Bestes“ Modell

- Theoretisch müssten wir nun **alle möglichen Variabel-Kombinationen** testen um das “beste” Modell zu finden
- Allerdings ist dieser Vorgang manuell durchgeführt fehleranfällig und zeitaufwendig (und macht wirklich keinen Spaß)
- Zum Glück gibt es eine Funktion, die diesen Schritt übernimmt:

```
step(model)
```

3. „Bestes“ Modell

```
> step(model)
```

3. „Bestes“ Modell

```
> step(model)

Start:  AIC=-1167.31

durationLog ~ structure + vowel + rate + number
```

Akaike Information Criterion
Je niedriger, desto besser der Fit

3. „Bestes“ Modell

Akaike Information Criterion
je niedriger, desto besser der Fit

```
> step(model)
Start:  AIC=-1167.31
durationLog ~ structure + vowel + rate + number

            Df  Sum of Sq      RSS      AIC
- number     1     0.0536   31.839 -1168.55
<none>                     31.786 -1167.31
- rate       1     0.8500   32.636 -1157.48
- vowel      4     3.4109   35.197 -1129.64
- structure  2    14.9708   46.756  -998.41
```

ein Model ohne number

ein Model ohne vowel

3. „Bestes“ Modell

```
Step:  AIC=-1168.55
durationLog ~ structure + vowel + rate
```

bestes gefundenes Model und sein AIC-Wert

```
             Df Sum of Sq    RSS      AIC
<none>                    31.839 -1168.55
- rate        1    0.8416 32.681 -1158.86
- vowel       4    3.4070 35.246 -1131.01
- structure   2   14.9881 46.827  -999.73
```

zusätzlicher Beweis dafür, dass eine weitere Reduzierung den Fit des Modells nicht verbessert

3. „Bestes“ Modell

bestes gefundenes Model und seine Struktur

```
Call:
lm(formula = durationLog ~ structure + vowel + rate, data = data)

Coefficients:
    (Intercept)    structureopen  structuresingle           vowele
        -1.5062           0.4340           0.1219          -0.1441
         voweli           vowelo           vowelu             rate
        -0.2374          -0.1229          -0.2365          -0.2532
```

Koeffizienten des Models – mehr dazu bei Schritt 5

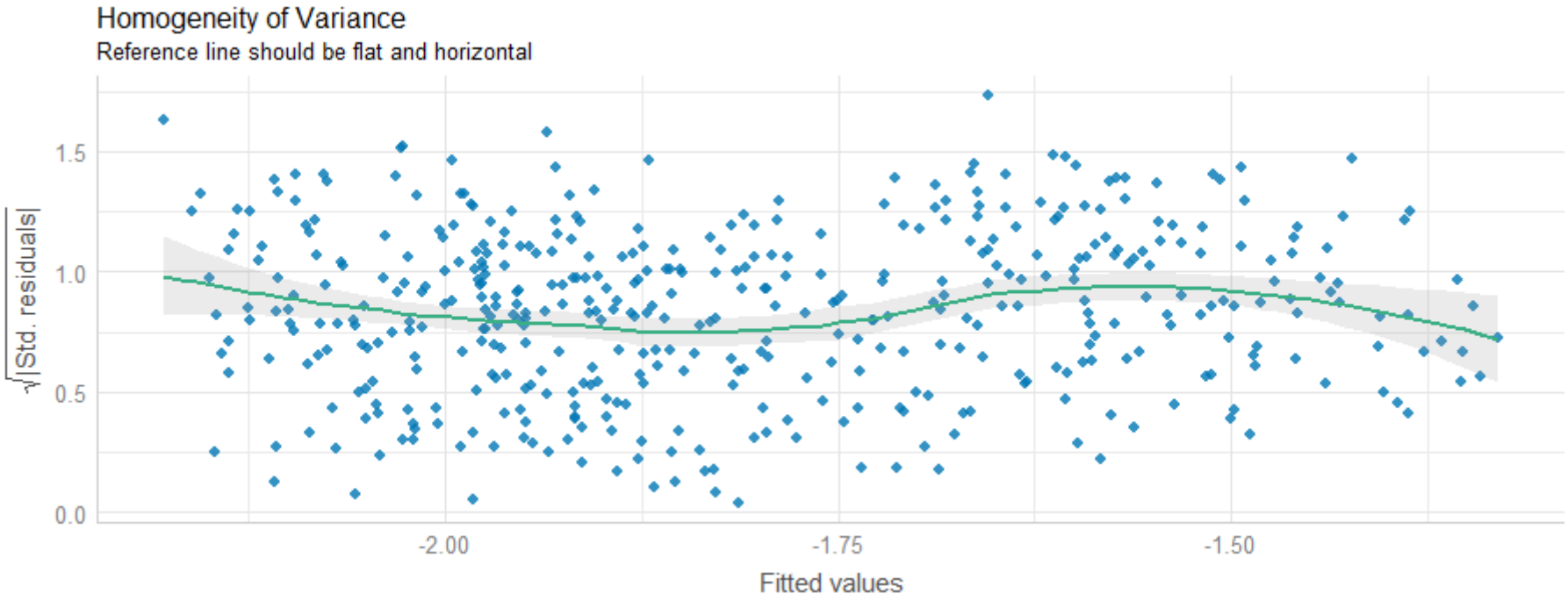
4. Annahmen überprüfen

- Multiple Lineare Regression folgt den gleichen Annahmen, denen auch Simple Lineare Regression folgt
  - Linearität / Linearity
  - Homoskedastizität / Homoscedasticity
  - Normalität / Normality
  - Unabhängigkeit / Independence

4. Annahmen überprüfen

- Annahme: Linearität

  Die Beziehung zwischen X und dem Mittelwert von Y ist linear.

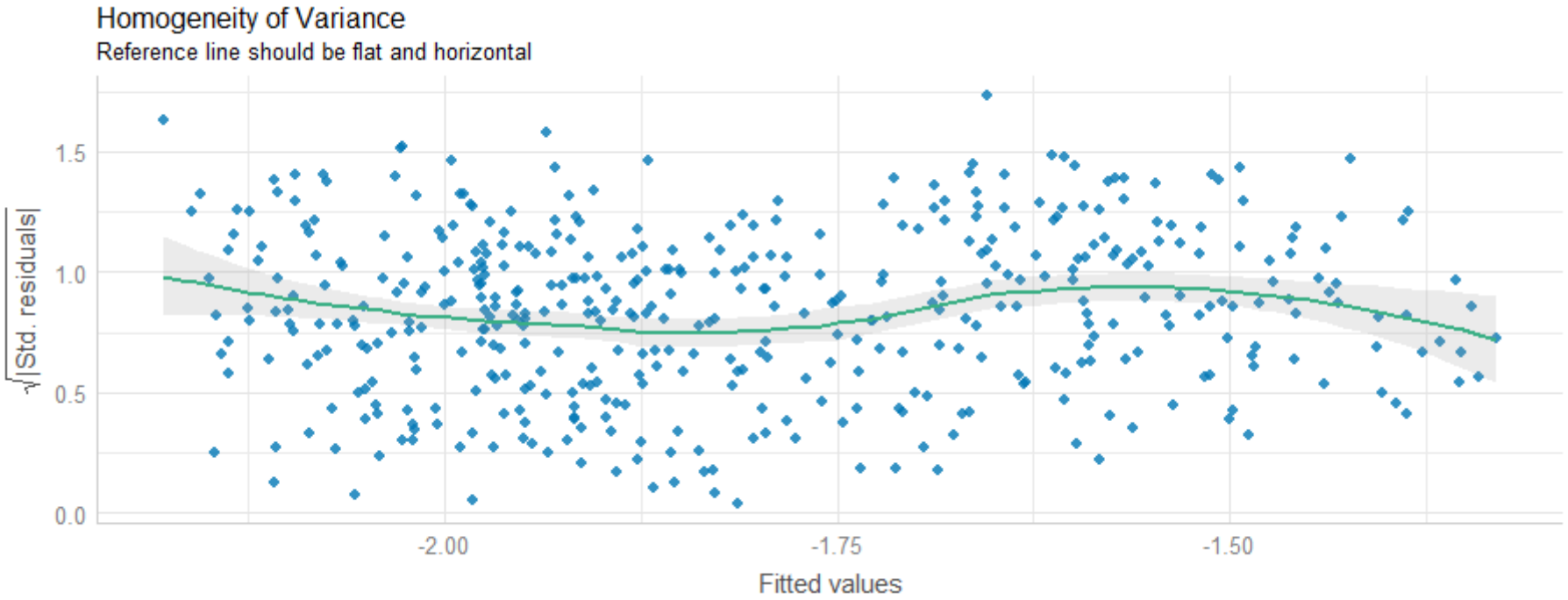


- Die Linie sollte horizontal und flach verlaufen.

4. Annahmen überprüfen

- Annahme: Homoskedastizität

  Die Varianz der Residuen ist für jeden Wert von X gleich.

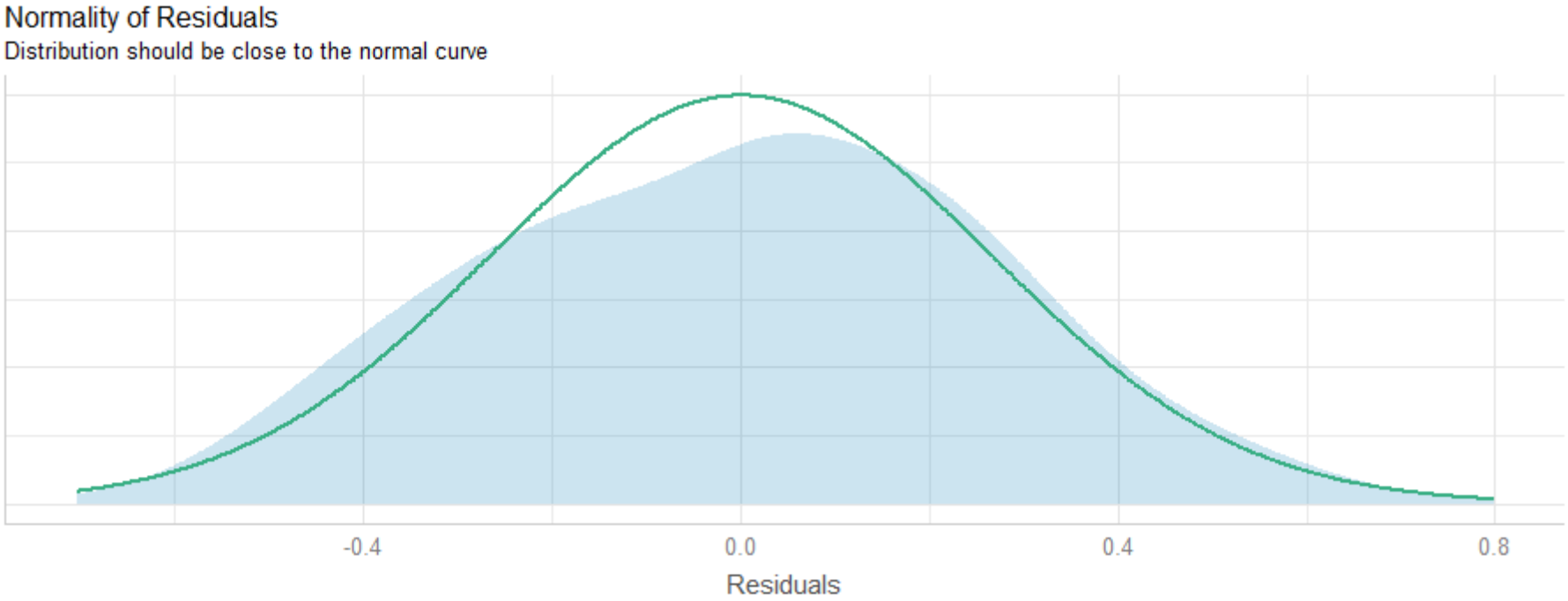


- Die Daten sollten gleichmäßig über die Linie verteilt sein und keine offensichtlichen Muster aufweisen.

4. Annahmen überprüfen

- Annahme: Normalität

  Für jeden festen Wert von X ist Y normalverteilt.



- Die Verteilung der Residuen eines linearen Modells sollte einer Normalverteilung folgen.

5. Interpretation

- Generell sind wir an zwei Dingen interessiert:
    1. den ***p*-Werten** der einzelnen Prädiktoren
    2. den **Effekten** der einzelnen Prädiktoren

5. Interpretation – *p*-Werte

1. Mit der `anova()` Funktion erhalten wir *p*-Werte

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	
structure	2	15.131	7.5654	104.4874	< 2.2e-16	***
vowel	4	3.507	0.8767	12.1079	2.41e-09	***
rate	1	0.842	0.8416	11.6241	0.0007112	***
Residuals	439	31.786	0.0724			

5. Interpretation – Effekte

2. Mit der `summary()` Funktion können wir einen Blick auf die einzelnen Effekte der Prädiktoren werfen

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	-1.506	0.105	-14.364	0.000	***
structureopen	0.434	0.031	13.947	0.000	***
structuresingle	0.122	0.031	3.910	0.000	***
vowele	-0.144	0.040	-3.572	0.000	***
voweli	-0.237	0.040	-5.883	0.000	***
vowelo	-0.123	0.040	-3.048	0.002	**
vowelu	-0.237	0.040	-5.864	0.000	***
rate	-0.253	0.074	-3.410	0.001	***

5. Interpretation – Effekte

2. Mit der `summary()` Funktion können wir einen Blick auf die einzelnen Effekte der Prädiktoren werfen

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	-1.506	0.105	-14.364	0.000	***
structureopen					***
structuresingle					***
vowele					***
voweli					***
vowelo					**
vowelu	-0.237	0.040	-5.864	0.000	***
rate	-0.253	0.074	-3.410	0.001	***

enthält die 'baseline' Werte aller Faktoren, d.h.

structure:double
vowel:a

& den 'Startpunkt' der numerischen Prädiktor(en)

5. Interpretation – Effekte

2. Mit der `summary()` Funktion können wir einen Blick auf die einzelnen Effekte der Prädiktoren werfen

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	-1.506	0.105	-14.364	0.000	***
structureopen	0.434	0.031	13.947	0.000	***
structuresingle				00	***
vowele				00	***
voweli	-0.237	0.040	-5.883	0.000	***
vowelo	-0.123	0.040	-3.048	0.002	**
vowelu	-0.237	0.040	-5.864	0.000	***
rate	-0.253	0.074	-3.410	0.001	***

structure:double + vowel:a + rate:start

geschätzter Durchschnitt von `durationLog`

5. Interpretation – Effekte

2. Mit der `summary()` Funktion können wir einen Blick auf die einzelnen Effekte der Prädiktoren werfen

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	-1.506	0.105	-14.364	0.000	***
structureopen					***
structuresingle					***
vowele					***
voweli	-0.237	0.040	-5.883	0.000	***
vowelo	-0.123	0.040	-3.048	0.002	**
vowelu	-0.237	0.040	-5.864	0.000	***
rate	-0.253	0.074	-3.410	0.001	***

structure:double + vowel:a + rate:start

Standardfehler des geschätzten Durchschnitts

5. Interpretation – Effekte

2. Mit der `summary()` Funktion können wir einen Blick auf die einzelnen Effekte der Prädiktoren werfen

structure:double + vowel:a + rate:start

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	-1.506	0.105	-14.364	0.000	***
structureopen	0.434	0.031	13.947	0.000	***
structuresingle	0.122	0.031	3.910	0.000	***
vowele				0	***
voweli				0	***
vowelo				2	**
vowelu				0	***
rate				0.001	***

um den geschätzten Durchschnitt von `durationLog` in structure:single Wörtern zu berechnen, wird der Schätzwert von `structure:single` zum Estimate-Schätzwert addiert, d.h.

$-1.506 + 0.122 = -1.384$

5. Interpretation – Effekte

2. Mit der `summary()` Funktion können wir einen Blick auf die einzelnen Effekte der Prädiktoren werfen

structure:double + vowel:a + rate:start

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	-1.506	0.105	-14.364	0.000	***
structureopen	0.434	0.031	13.947	0.000	***
structuresingle	0.122	0.031	3.910	0.000	***
vowele	-0.144	0.040	-3.572	0.000	***
voweli	-0.237	0.040	-5.883	0.000	***
vowelo					
vowelu					
rate					

um den geschätzten Durchschnitt von `durationLog` in structure:single Wörtern mit `vowel:i` zu berechnen, wird ebenfalls die Summe gebildet, d.h.

$$-1.506 + 0.122 - 0.237 = -1.621$$

5. Interpretation – Effekte

2. Mit der `summary()` Funktion können wir einen Blick auf die einzelnen Effekte der Prädiktoren werfen

structure:double + vowel:a + rate:start

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	-1.506				***
structureopen	0.434				***
structuresingle	0.122				***
vowele	-0.144				***
voweli	-0.237				***
vowelo	-0.123				**
vowelu	-0.237	0.040	-5.864	0.000	***
rate	-0.253	0.074	-3.410	0.001	***

`durationLog` ist

- signifikant höher in Silben ohne Coda
- signifikant höher in Silben mit einfacher Coda

im Vergleich zu Silben mit komplexer Coda

5. Interpretation – Effekte

2. Mit der `summary()` Funktion können wir einen Blick auf die einzelnen Effekte der Prädiktoren werfen

structure:double + vowel:a + rate:start

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	-1.506	0.105	-14.364	0.000	***
structureopen	0.434	0.031	13.947	0.000	***
structuresingle	0.122	0.031	3.910	0.000	***
vowele	-0.144			0	***
voweli	-0.237			0	***
vowelo	-0.123			2	**
vowelu	-0.237			0	***
rate	-0.253	0.074	-3.410	0.001	***

`durationLog` ist

- signifikant niedriger in Silben mit /e, i, o, u/

als in Silben mit /a/

5. Interpretation – Effekte

2. Mit der `summary()` Funktion können wir einen Blick auf die einzelnen Effekte der Prädiktoren werfen

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	-1.506	0.105	-14.364	0.000	***
structureopen	0.434	0.031	13.947	0.000	***
structuresingle	0.122	0.031	3.910	0.000	***
vowele	-0.144	0.040	-3.572	0.000	***
voweli	-0.237	0.040	-5.883	0.000	***
vowelo	-0.122	0.040	-3.048	0.002	**
vowelu	-0.237	0.040	-5.864	0.000	***
rate	-0.253	0.074	-3.410	0.001	***

je höher die Sprechgeschwindigkeit, desto niedriger `durationLog`

5. Interpretation – Effekte

2. Mit der `summary()` Funktion können wir einen Blick auf die einzelnen Effekte der Prädiktoren werfen

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	-1.506	0.105	-14.364	0.000	***
structureopen	0.434	0.031	13.947	0.000	***
structuresingle	0.122	0.031	3.910	0.000	***
vowele	-0.144	0.040	-3.572	0.000	***
voweli	-0.237	0.040	-5.883	0.000	***
vowelo	-0.123	0.040	-3.048	0.002	**
vowelu	-0.237	0.040	-5.864	0.000	***
rate	-0.253	0.074	-3.410	0.001	***

5. Interpretation – Effekte

Der s.g. **Tukey-Contrast** zeigt uns die Unterschiede innerhalb eines kategorischen Prädiktors

```
> tukey(model = mdl_fin, predictor = structure)

                     Estimate Std. Error t value Pr(>|t|)
open - double == 0    0.43395    0.03112   13.95  < 1e-04 ***
single - double == 0  0.12186    0.03117    3.91  0.00031 ***
single - open == 0   -0.31209    0.03111  -10.03  < 1e-04 ***
```