

# Multiple Lineare Regression

Richard Hunger

24. Oktober 2024

# Übersicht

- Generalisierung der bivariaten Regression
- Durchführung in R
- Prüfung der Modellannahmen
- Bewertung der Modellgüte
- Ausreißer- & Einflussdiagnostik
- Prädiktortransformationen
- Variablenelektion
- Vorhersage (Prädiktion)
- Ergebnisbericht
- Visualisierungen
- Interaktionen
- Übungen

# Motivation

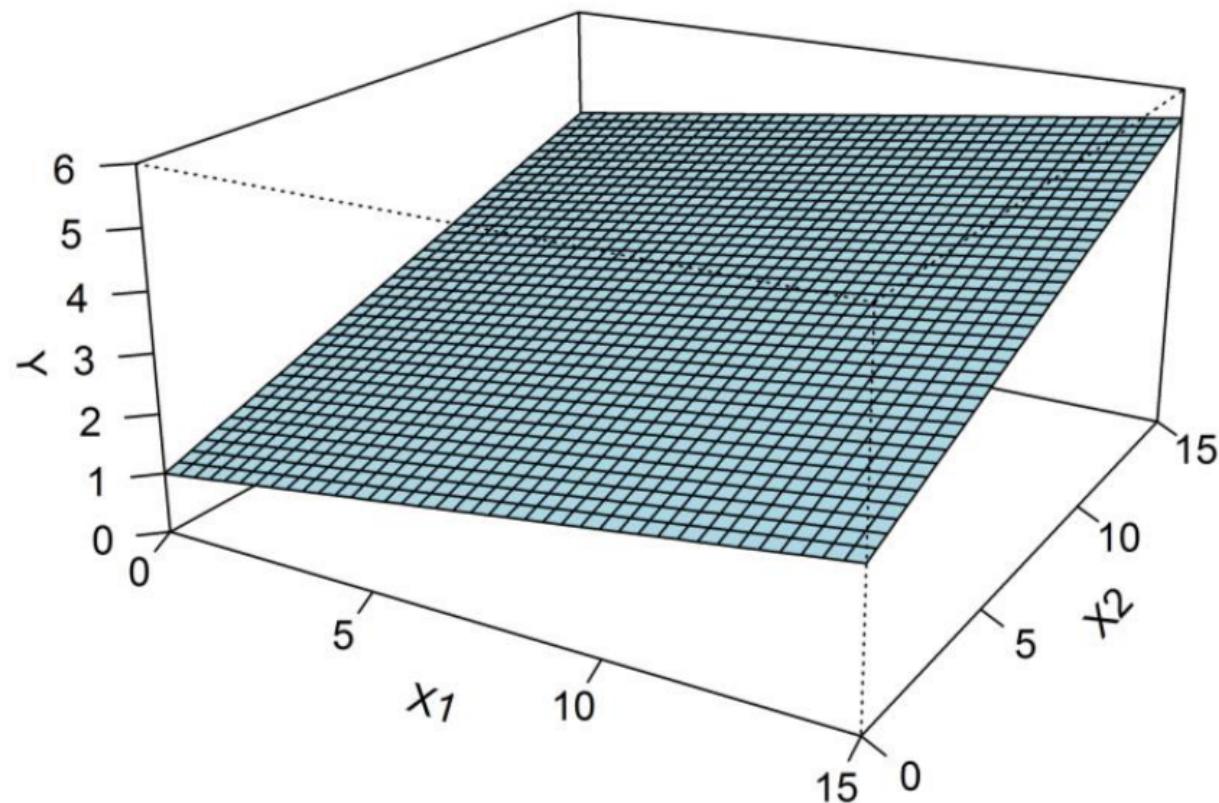
- idR sind mehrere Prädiktoren für ein Kriterium relevant
- selten monokausale Zusammenhänge
- gleichzeitige Berücksichtigung verschiedener Einflussfaktoren

# multiple lineare Regression

- *multipel* mehrere UV gleichzeitig:
- Gleichung:  $Y = b_0 + b_1 X_1 + b_2 X_2 + b_3 X_3 + \dots + b_k X_k$
- Beispiele:
  - Rauchverhalten, sozio-ökonomischer Status, Geschlecht, ... → Lebenserwartung
  - Alter, Bildung, Einkommen, Guthaben, Geschlecht, Ethnie ... → Kreditausfall
- $k$  ... Anzahl der Prädiktoren
- (nichtlineare) Transformationen der Prädiktoren möglich
  - zB.  $X^2$ ,  $\log(X)$ ,  $\sqrt{X}$ ,  $\frac{1}{X}$ , ...
- kategoriale Prädiktoren möglich
- multiplikative Verknüpfung von Prädiktoren möglich (Interaktion)

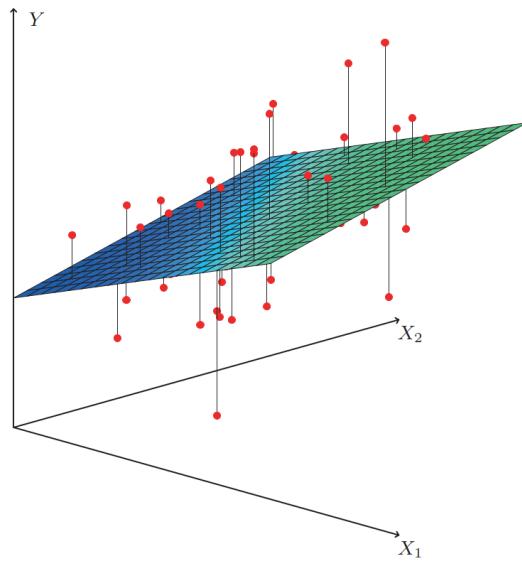
# multiple lineare Regression

- Aus der Regressiongrade wird eine Regressionsebene
- für den 2-Prädiktor-Fall:  $y = 1 + 0.1 \times X_1 + 0.2 \times X_2$



# Residuen

- Visualisierung:



- Residualvarianz, Residual Standard Error (RSE)<sup>2</sup>:

- einfache OLS:  $\sigma_{\epsilon}^2 = \frac{1}{N-2} \sum ((y - \hat{y})^2)$
- multiple OLS:  $\sigma_{\epsilon}^2 = \frac{1}{N-k-1} \sum ((y - \hat{y})^2)$

# Beispieldatensatz

- Erweiterung des Eingangsbeispiels um weitere Variablen

```
• df_cars <-  
  mtcars %>%  
  mutate(lkm = 235.2145/mpg) %>%  
  mutate(Auto = row.names(.)) %>%  
  select(-mpg)
```

# Prädiktor-Identifikation

- Welche Variablen eignen sich (noch) für die Vorhersage des Kraftstoffverbrauchs?
- bivariate Korrelationen:

```
korrr_matrix <-
  df_cars %>%
  select(where(is.numeric)) %>%
  cor()

korrr_matrix[c(11, 1:5), c(11, 1:5)] %>%
  flex()
```

	Var2					
	Ikm	cyl	disp	hp	drat	wt
Var1						
Ikm	1.0000000	0.8137493	0.8798217	0.7629477	-0.6380538	0.8898927
cyl	0.8137493	1.0000000	0.9020329	0.8324475	-0.6999381	0.7824958
disp	0.8798217	0.9020329	1.0000000	0.7909486	-0.7102139	0.8879799
hp	0.7629477	0.8324475	0.7909486	1.0000000	-0.4487591	0.6587479
drat	-0.6380538	-0.6999381	-0.7102139	-0.4487591	1.0000000	-0.7124406
wt	0.8898927	0.7824958	0.8879799	0.6587479	-0.7124406	1.0000000

# Prädiktor-Identifikation

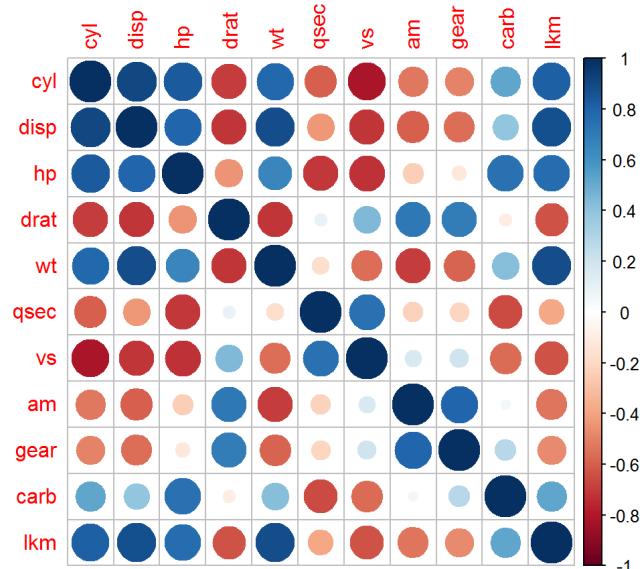
- Korrelationsplot:

```
df_cars %>%
  select(-Auto) %>%
  GGally::ggpairs()
```

# Prädiktor-Identifikation

- Korrelationsplot:

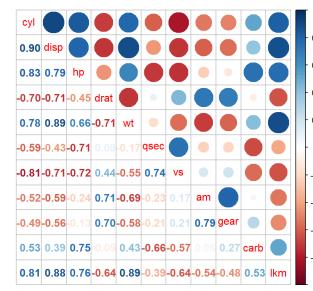
```
korrr_matrix %>%
  corrplot::corrplot()
```



# Prädiktor-Identifikation

- verbesserter Korrelationsplot:

```
corr_matrix %>%
  corrplot::corrplot.mixed()
```



- Welche weiteren Variablen als Prädiktor?
  - hp ... Leistung
  - wt ... Gewicht
  - disp ... Hubraum
  - cyl ... Zylinderzahl

# Berechnung:

- Erweiterung der Modellformel in R:

```
lm_obj2 <- lm(lkm ~ hp + disp + cyl + wt, data = df_cars)
tidy(lm_obj2)
```

- ↗ Ist `cyl` korrekt berücksichtigt?
- Gewicht in lbs?
- `wt` in Tonnen umrechnen!

```
df_cars <-
  mtcars %>%
  mutate(lkm  = 235.2145/mpg) %>%
  mutate(wt   = wt*0.45359237) %>%
  mutate(cyl  = factor(cyl)) %>%
  mutate(Auto = row.names(.)) %>%
  select(-mpg)

lm_obj2 <- lm(lkm ~ hp + disp + cyl + wt, data = df_cars)
```

# Modellergebnisse

- Modellergebnisse ausgeben:

```
lm_obj2  
summary(lm_obj2)  
broom::tidy(lm_obj2)
```

term	estimate	std.error	statistic	p.value
(Intercept)	2.24	1.39	1.61	p = 0.120
hp	0.01	0.01	1.72	p = 0.097
disp	0.01	0.01	0.70	p = 0.492
cyl6	0.08	0.96	0.08	p = 0.935
cyl8	0.02	1.84	0.01	p = 0.990
wt	4.87	1.52	3.20	p = 0.004

- Regressionsgleichung:

$$\begin{aligned} \text{Verbrauch} &= 2.24 + 0.014 \times PS + 0.006 \times \text{Hubraum} + 4.87 \times \text{Gewicht} \\ &\quad + 0.08 \times D_{cyl=6} + 0.02 \times D_{cyl=8} \end{aligned}$$

- Interpretation:

- Wirkung der Variablen  $X_j$  bei gleichzeitiger Kontrolle der Effekte der anderen Prädiktoren

# Voraussetzungen

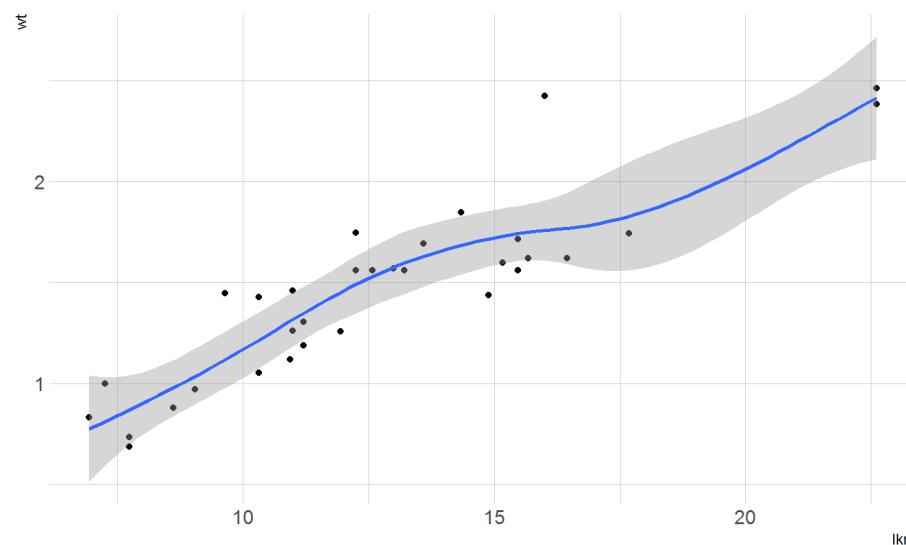
- gleiche Modellannahmen wie bei der einfachen linearen Regression:
  - Linearität
  - Normalverteilung der Residuen
  - keine Autokorrelation
  - Homoskedastizität
- mindestens so viele Beobachtungen wie Prädiktoren
  - $N > (k + 1)$
  - notwendig für Parameterschätzung und Signifikanztests
- keine Multikollinearität

# Modellannahmen

## 1. Linearität

- Grundannahme auch bei *multipler* Regression: lineare Beziehung der Prädiktoren zum Kriterium
- graphisch:
  - bivariate Streudiagramme zwischen Y und einzelnen Prädiktoren:
  - mittel ggplot:

```
df_cars %>%
  ggplot(aes(lkm, wt)) +
  geom_point() +
  geom_smooth(method = 'loess', formula = 'y ~ x')
```

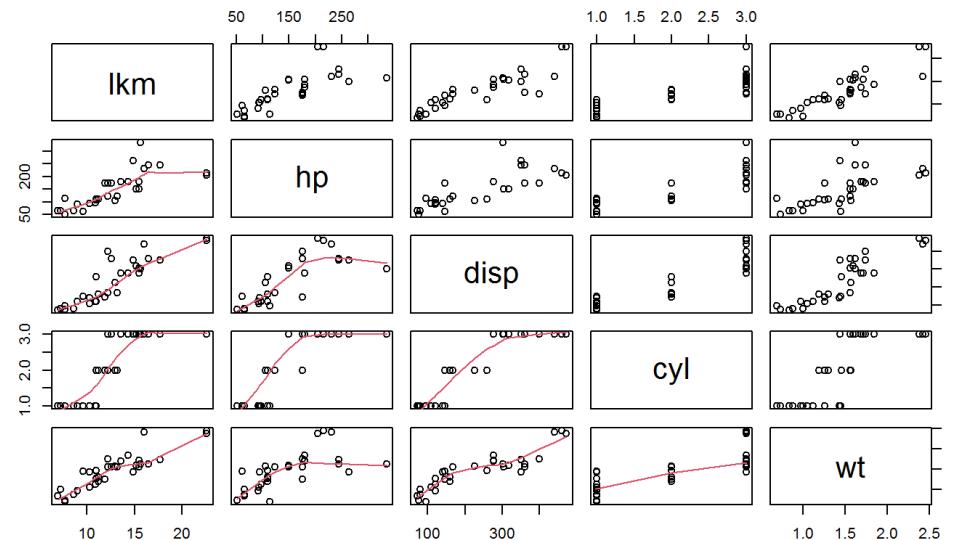


# Modellannahmen

## 1. Linearität

- bei Vielzahl an Variablen umständlich
- Basis Funktion von R `pairs()`

```
df_cars %>%
  select(lkm, hp, disp, cyl, wt) %>%
  pairs(lower.panel = panel.smooth)
```

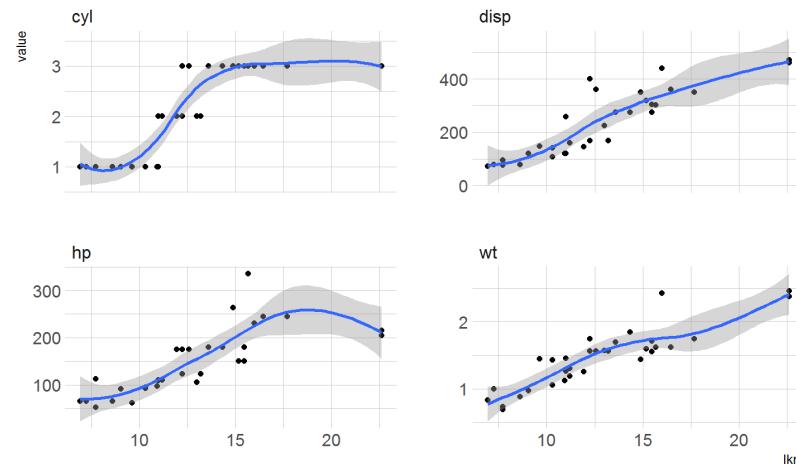


# Modellannahmen

## 1. Linearität

- mit etwas Datenumformung auch als ggplot:

```
df_cars %>%
  select(all.vars(lm_obj2$terms)) %>%
  mutate_all(as.numeric) %>%
  pivot_longer(-lkm) %>%
  ggplot(aes(lkm, value)) +
  geom_point() +
  facet_wrap(. ~ name, scales = 'free_y') +
  geom_smooth(method = 'loess', formula = 'y ~ x')
```



# Modellannahmen

## 1. Linearität

- in einem Befehl, verbesserte `pairs()`-Version:

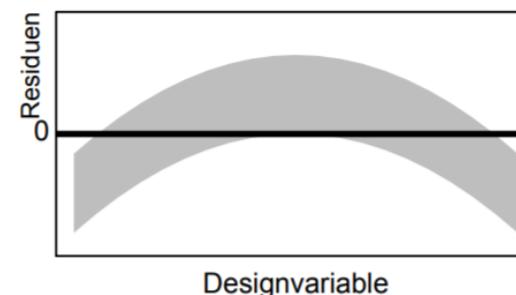
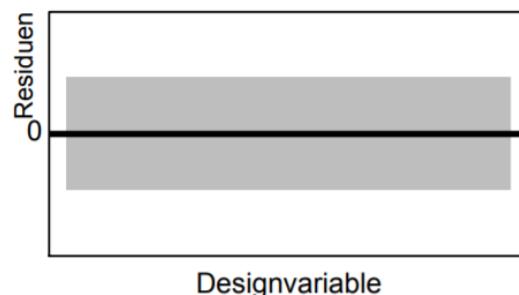
```
df_cars %>%
  select(all.vars(lm_obj2$terms)) %>%
  GGally::ggpairs(lower = list(continuous =
    GGally::wrap("smooth", method = "loess")))
```

```
## plot: [1, 1] [=====>-----]
## plot: [1, 2] [======>-----]
## plot: [1, 3] [==========>-----]
## plot: [1, 4] [===========>-----]
## plot: [1, 5] [================>-----]
## plot: [2, 1] [=====================>-----]
## plot: [2, 2] [=====================>-----]
## plot: [2, 3] [=====================>-----]
## plot: [2, 4] [=====================>-----]
## plot: [2, 5] [=====================>-----]
## plot: [3, 1] [=====================>-----]
## plot: [3, 2] [=====================>-----]
## plot: [3, 3] [=====================>-----]
## plot: [3, 4] [=====================>-----]
## plot: [3, 5] [=====================>-----]
## plot: [4, 1] [=====================>-----]
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`. plot: [4, 2]
## [=========================================>-----]
## using `bins = 30`. Pick better value with `binwidth`. plot: [4, 3]
## [====================================>-----]
## using `bins = 30`. Pick better value with `binwidth`. plot: [4, 4]
## [====================================>-----]
## [===============================>-----]
## [==========================>-----]
```

# Modellannahmen

## 1. Linearität

- Residuenplots wieder Mittel der Wahl zur Prüfung des Gesamtmodells
- einfaches Streudiagramm:
  - x-Achse: nicht mehr  $x_i$ , sondern  $\hat{y}_i$
  - y-Achse:  $\epsilon_i$
- Plot:



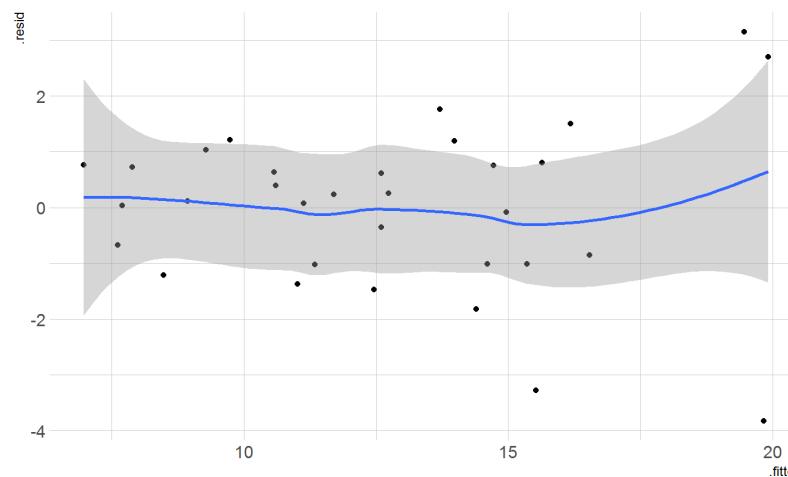
- die Residuen sollten über den gesamten x-Bereich gleichmäßig über- bzw. unterhalb der 0-Linie streuen (linke Abb)
- dabei sollte kein systematisches Muster erkennbar sein (z.B. rechte Abb)

# Modellannahmen

## 1. Linearität

- in R: Residuenplots

```
augment(lm_obj2) %>%
  ggplot(aes(.fitted, .resid)) +
  geom_point() +
  geom_smooth(method = 'loess', formula = 'y ~ x')
```

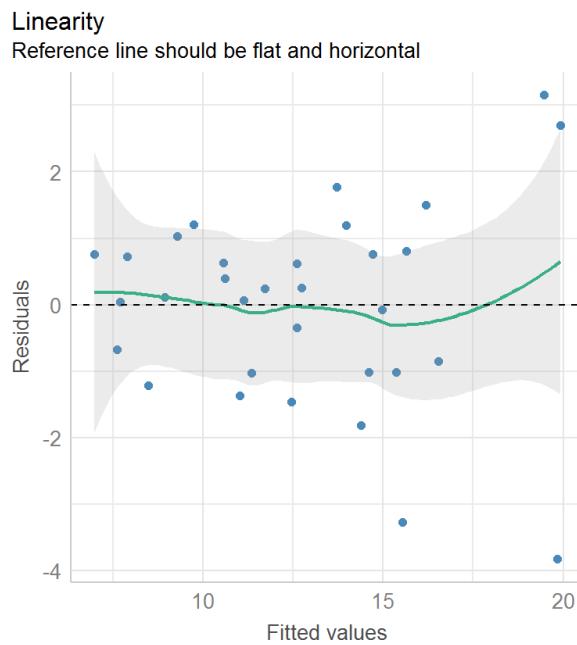


# Modellannahmen

## 1. Lineare Beziehung

- Die Visualisierung mit einer Funktion:

```
performance::check_model(lm_obj2, check = 'linearity')
```



# Modellannahmen

## 1. Linearität

- inferenzsstatistisch: **Rainbow-Test**

```
lmtest::raintest(lm_obj2, fraction = .5) %>%
  broom::tidy() %>%
  flex()
```

```
## Multiple parameters; naming those columns df1, df2
```

df1	df2	statistic	p.value	method
16	10	0.89	p = 0.596	Rainbow test

# Modellannahmen

## 1. Linearität

- Zusammenfassung:
  - Residuenplot
  - bei nicht-linearität: bivariate Streudiagramme inspizieren
  - ggf. *linearisierende Transformationen* des Prädiktors X:  $\sqrt{X}$ ;  $\log X$ ;  $X^2$ ;  $e^X$ ;  $\frac{1}{X}$ ;  $e^{-X}$
  - ggf. Regression um Polynome erweitern ( $X^2$ ,  $X^3$ , ...), insbes. bei Homoskedastizität
  - ggf. transformation des Kriteriums (zB.  $\log(Y)$ , ...), insbes. bei nicht-NV Residuen

# Modellannahmen

## 2. NV der Residuen

- Der Mittelwert der Residuen beträgt 0 (*immer*)
- die Fehler sollen dabei aber gleichmäßig um die 0 streuen
- Die Methoden zur Prüfung sind identisch zur einfachen lin. Regression:
  - Verteilung/Dichteplot der Residuen
  - QQ-Plot
  - Shapiro-Wilk-Test

# Modellannahmen

## 2. NV der Residuen

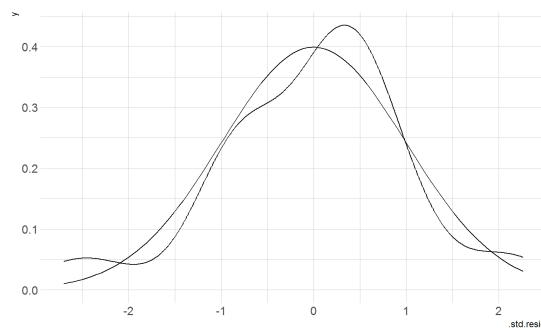
- Prüfen Sie unser Regressionsmodell hinsichtlich der NV der (standardisierten) Residuen:
  - Verteilung/Dichteplot der Residuen
  - QQ-Plot
  - Shapiro-Wilk-Test

# Modellannahmen

## 2. NV der Residuen

- Verteilung/Dichteplot der Residuen
- Dichteplot:

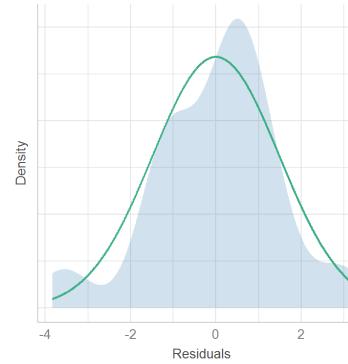
```
augment(lm_obj2) %>%
  ggplot(aes(.std.resid)) +
  geom_density() +
  stat_function(fun = dnorm)
```



- NV-Prüfung:

```
performance::check_model(lm_obj2, check = 'normality')
```

Normality of Residuals  
Distribution should be close to the normal curve

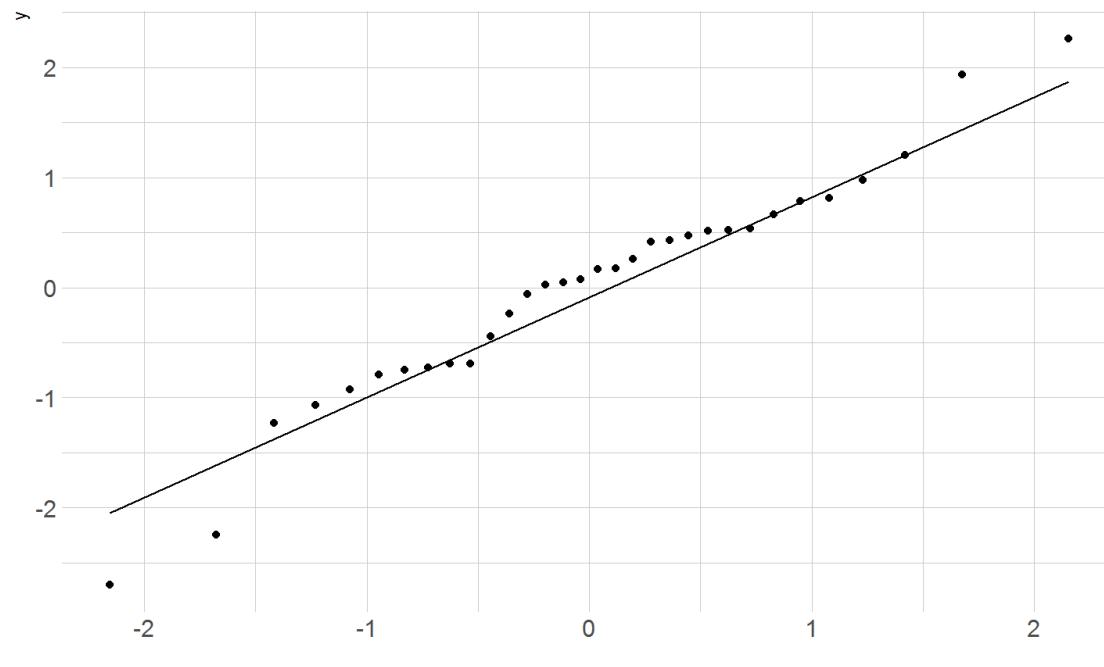


# Modellannahmen

## 2. NV der Residuen

- QQ-plot der Residuen

```
augment(lm_obj2) %>% ggplot(aes(sample = .std.resid)) + geom_qq() + geom_qq_line()
```



# Modellannahmen

## 2. NV der Residuen

- NV-Tests der Residuen

```
• augment(lm_obj2) %>%  
  shapiro_test(.std.resid) %>%  
  flex()
```

variable	statistic	p
.std.resid	0.97	0.38

```
• performance::check_normality(lm_obj2)
```

```
## OK: residuals appear as normally distributed (p = 0.380)
```

# Modellannahmen

## 3. Unabh. der Residuen

- Berechnung der Autokorrelation:

- `residuals(lm_obj2) %>% acf(plot = F)`

- Plotten der Autokorrelation

- `residuals(lm_obj2) %>% acf()`

- “blaue Linien” ... Signifikanzschwelle

- Durbin-Watson-Test:

```
lmtest::dwtest(lm_obj2) %>%
  broom::tidy() %>%
  flex()
```

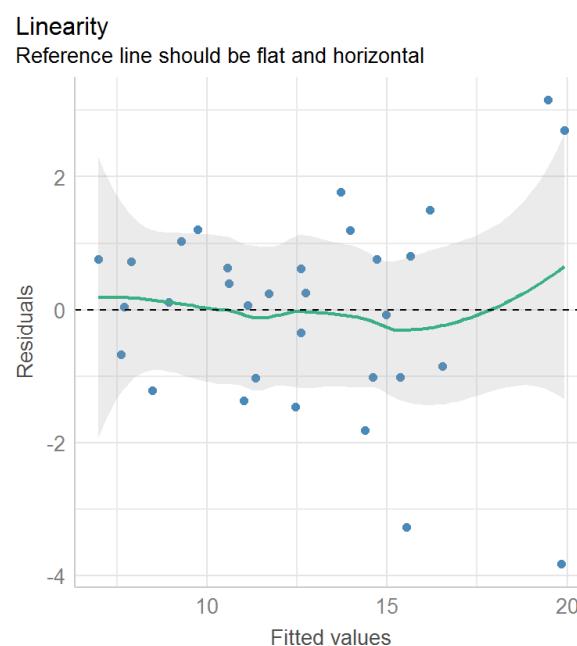
statistic	p.value	method	alternative
1.92	p = 0.249	Durbin-Watson test	true autocorrelation is greater than 0

# Modellannahmen

## 4. Homoskedastizität

- Prinzipiell auch wieder in Residuenplot erkennbar:

```
performance::check_model(lm_obj2, check = 'linearity')
```



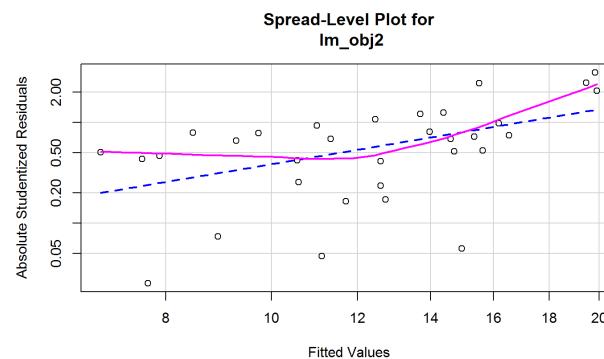
- tendenziell Trichterform erkennbar:

# Modellannahmen

## 4. Homoskedastizität

- *Spread & Level-Plot*
- in Kurzform:

```
sldat <- car::spreadLevelPlot(lm_obj2)
```



- ⚠ Hinweis auf Heteroskedastizität

# Modellannahmen

## 4. Homoskedastizität

- *Breusch-Pagan-Heteroskedastizitätstest*
- in Kurzform:

```
lmtest::bptest(lm_obj2) %>% broom::tidy() %>% flex()
```

statistic	p.value	parameter	method
22.39	p < .001	5	studentized Breusch-Pagan test

- ↗ Trichterform, sig. BP-Test, SL-Plot Anstieg -> Heteroskedastizität

# Modellannahmen

## 4. Homoskedastizität

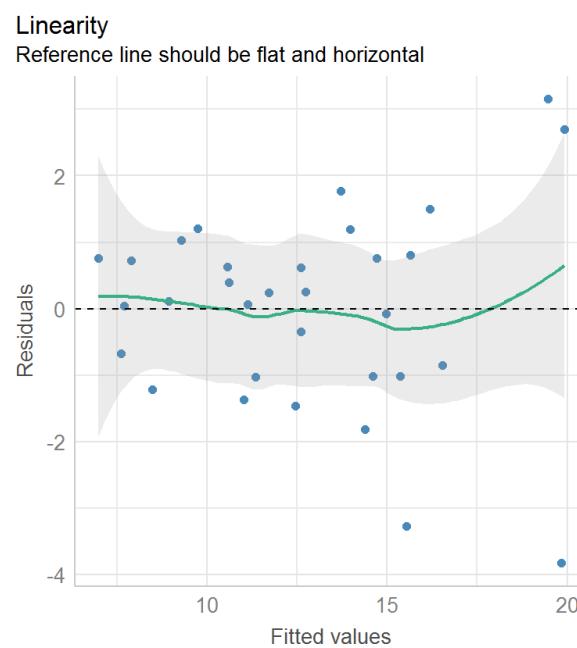
- Gegenmaßnahmen:
  - einzelne Ausreißer eliminieren
  - fehlende(n) Prädiktor(en) ergänzen
  - Transformation des Kriteriums (S-L-Plot Anstieg)
  - HCC-Schätzer verwenden
-  Führen Sie, sofern möglich, jeweils einzeln die Gegenmaßnahmen zur Behebung der Heteroskedastizität durch! Konnte der Modelldefekt dadurch behoben werden? Wie wirken sich die Modifikationen auf die Modellgüte aus?

# Modellannahmen

## 4. Homoskedastizität

- einzelne Ausreißer eliminieren (ab  $| \text{Residuum} | > 2-3$ )
- Identifikation

```
# Identifikation  
performance::check_model(lm_obj2, check = 'linearity')
```



# Modellannahmen

## 4. Homoskedastizität

```
# tmp nach .fitted sortieren
# vgl mit Resduenplot
tmp <- augment(lm_obj2)

# 4 Fälle filtern
lm_h1 <-
  augment(lm_obj2) %>%
  filter(abs(.resid) < 2.5) %>%
  lm(lkm ~ hp + disp + cyl + wt, data = .) # Trick17: lm_obj2[["terms"]]

glance(lm_h1) %>% flex()
```

r.squared	adj.r.squared	sigma	statistic	p.value	df	logLik	AIC	BIC	deviance	df.residual	nobs
0.9088	0.88	1.05	39.31	p < .001	5	-37.63	89.25	98.58	24.09	22	28

```
glance(lm_obj2) %>% flex()
```

r.squared	adj.r.squared	sigma	statistic	p.value	df	logLik	AIC	BIC	deviance	df.residual	nobs
0.85082	0.82	1.62	29.89	p < .001	5	-57.6	129.19	139.45	68.55	26	32

# Modellannahmen

## 4. Homoskedastizität

- Transformation des Kriteriums (S-L-Plot Anstieg)

```
# Anstieg in S-L-Plot
tmp <- car::slp(lm_obj2)

# Transformation
lm_h2 <-
  df_cars %>%
  mutate(lkm = lkm^tmp$PowerTransformation) %>%
  lm(lkm ~ hp + disp + cyl + wt, data = .)

performance::check_model(lm_h2, check = 'linearity')
car::slp(lm_h2)
lmtest::bptest(lm_h2)
glance(lm_h2)
```

# Modellannahmen

## 4. Homoskedastizität

- HCC-Schätzer

```
• lm_h3 <-  
  lmtest::coeftest(  
    lm_obj2,  
    vcov = sandwich::vcovHC(lm_obj2, type = "HC3"))  
  
tidy(lm_h3, conf.int = T)
```

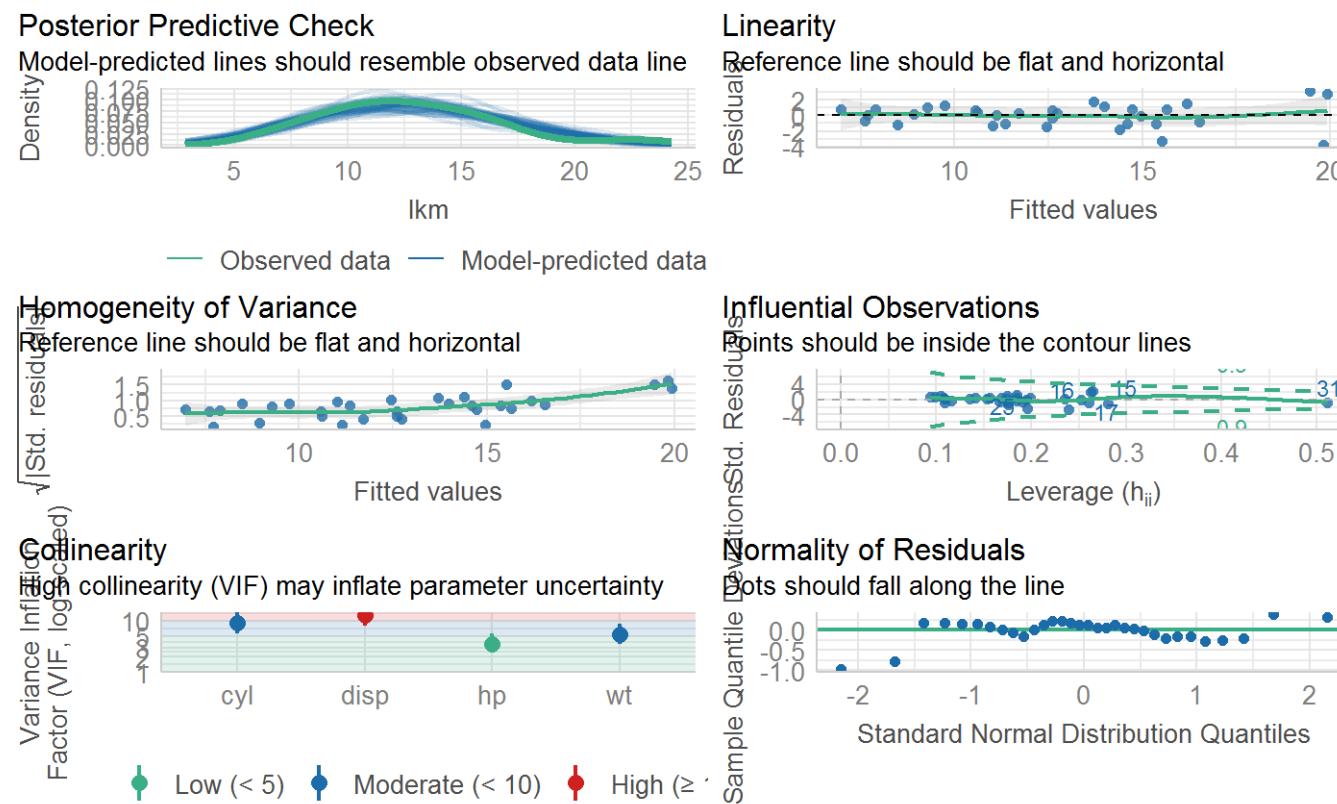
- ⚡ Schätzt nur die Standardfehler neu, nicht die RK und das Modell!

# Modellannahmen

## All-in-One

- visuelle Modellinspektion zusammen:

```
• performance::check_model(lm_obj2)
```



Chill...



# Modellprüfung

- auch beim multiplen Regressionsmodell gibt es die Möglichkeit zur Quadratsummenzerlegung
- dazu wieder die ANOVA-Tabelle der Regression:

```
anova(lm_obj2) %>%
  rownames_to_column('Term') %>%
  flex()
```

Term	Df	Sum Sq	Mean Sq	F value	Pr(>F)
hp	1	269.31	269.31	102.14	0.00
disp	1	94.39	94.39	35.80	0.00
cyl	2	3.41	1.70	0.65	0.53
wt	1	27.01	27.01	10.24	0.00
Residuals	26	68.55	2.64		

- Die Gesamtsumme der SQ bleibt gleich mit  $SQ_{Total} = 462.7$
- Die Gesamtsumme der Fehler ist aber deutlich kleiner  $SQ_{Residuen} = 68.6$ 
  - vgl. auch `glance(lm_obj2)$deviance`
- **⚠** Die Ergebnisse zu den Einzeltermen nicht interpretieren!

# Modellprüfung

- Signifikanztests der Regressionskoeffizienten nicht direkt anwendbar
- Problem Alpha-Fehler Inflation
- bei bswp. 5 Koeffizienten liegt Alpha-Fehler schon bei 22.6% ( $1 - 0.95^5$ )
- Lösung globaler F-Test:
- $$F = \frac{SQ_{Regression}/k}{SQ_{Residenz}/(n-k-1)}$$
- in *einem einzigen* Test wird geprüft ob:
  - $H_0 : \beta_1 = \beta_2 = \beta_3 = \dots = \beta_k = 0$
  - $H_A : \text{irgendein } \beta \neq 0$

# Modellprüfung

- Führen Sie die Regression erneut durch, aber ändern Sie die Reihenfolge der Prädiktoren im Modell. Vergleichen Sie die Modelle hinsichtlich der Regressionskoeffizienten und der ANOVA-Tabelle. Was fällt Ihnen auf?

# Modellprüfung

- Bei der Regressions-ANOVA ist lediglich die SQ-Summe aller Prädiktoren mit der Residualsumme zu vergleichen
- Die Mittlere Quadratsumme ergibt sich aus:
  - $MSQ_{Regression} = \sum SQ_{Prediktoren} / \sum df_{Praediktoren}$
- im Beispiel:
  - $SQ_{Regression} = 394.1$
  - $df_{Regression} = 5$
  - $MSQ_{Regression} = 78.8$
  - $MSQ_{Residuen} = 2.6$
  - $F = 78.8 / 2.6 = 30.3$
  - `pf(29.895, 5, 31, lower.tail = F)`
  - `glance(lm_obj)`

# Modellprüfung

- entspricht Prüfung gegen das **Nullmodell**
  - auch *Intercept-Only-Modell*
  - Modell (Vorhersage) nur mit Mittelwert
  - in R:

```
lm_obj_null <- lm(lkm ~ 1, data = df_cars)
tidy(lm_obj_null) %>% flex()
```

term	estimate	std.error	statistic	p.value
(Intercept)	12.76	0.68	18.68	p < .001

- Vergleich mittels ANOVA:

```
anova(lm_obj_null, lm_obj2) %>% tidy() %>% flex()
```

term	df.residual	rss	df	sumsq	statistic	p.value
lkm ~ 1	31	462.66				
lkm ~ hp + disp + cyl + wt	26	68.55	5	394.11	29.89	p < .001

# Modellgüte

## Übersicht

- diverse Maßzahlen und Kriterien
  - *Bestimmtheitsmaß* ( $R^2$ )
  - *Residual Sum of Squares* (RSS)
  - *Mean Squared Error* (MSE)
  - *Root Mean Squared Error* (RMSE)
  - *Residual Standard Error* (RSE)

# Modellgüte

$R^2$

- zunehmende Anzahl Prädiktoren ( $k$ ) führt *immer* zu Erhöhung von  $R^2$
- unabhängig von tatsächlichem Zusammenhang zwischen UVs und AV
- klassisches **Bestimmtheitsmaß** nicht mehr anwendbar
- gewisse “Zufallszusammenhänge” bestehen immer

# Modellgüte

$R^2$

- Beispiel :

```
set.seed(4)
tibble(
  x1 = rnorm(100),
  x2 = rnorm(100),
  x3 = rnorm(100),
  x4 = rnorm(100),
  x5 = rnorm(100),
  y   = rnorm(100, 5, 2)
) %>%
  lm(y ~ ., data = .) %>%
  tidy() %>%
  flex()
```

term	estimate	std.error	statistic	p.value
(Intercept)	4.96	0.19	26.20	p < .001
x1	0.19	0.21	0.92	p = 0.360
x2	-0.35	0.19	-1.83	p = 0.070
x3	-0.02	0.20	-0.10	p = 0.921
x4	0.40	0.20	2.04	p = 0.044
x5	0.18	0.19	0.95	p = 0.344

# Modellgüte

$$R^2_{adj}$$

- klassisches **Bestimmtheitsmaß** nicht mehr anwendbar
- Lösung: "Strafterm" für Prädiktorzahl
- $\Rightarrow$  adjustiertes / korrigiertes  $R^2$
- Formel:  $R^2_{adj} = R^2 - \frac{k \times (1-R^2)}{n-k-1}$
- in R: `glance(lm_obj2)`

# Modellgüte

## RSS & MSE

- RSS: Summe der Abweichungsquadrate
  - s.o. ANOVA der Regression
  - `glance(lm_obj2)`, 68.6
  - `glance(lm_obj)`, 193.3
- MSE: Mittelwert der Abweichungsquadrate
  - $68.6/32 = 2.14$
  - $193.3/32 = 6.04$

# Modellgüte

## RSE

- wie bei einfacher linearer Regression
  - $\epsilon \sim \mathcal{N}(0, \sigma)$
  - interpretierbar als die durchschnittliche Abweichung der Beobachtungen von der Schätzung (Regressionsgrade)
  - `sigma(lm_obj2)`
- 
- Verbesserung ggü. einfacher Regression ( $\sigma_\epsilon = 2.54$ )
  - Präzisere Schätzung um  $(1 - 1.62/2.54) * 100 = 36.2\%$
  - Problem:
    - Abhängig von Maßeinheit (vgl m vs. mm)

# Modellgüte & -genauigkeit

## relativer RSE

- "durchschnittlicher" Fehler im Verhältnis zum durchschnittlichen Kriteriumswert
- `sigma(lm_obj2) / mean(df_cars$1km)`
- 12.7%
- vgl. einfache Regression: 20% relativer RSE

# Modellgüte

## Zusammenfassung in R

-  `broom::glance(lm_obj2)`
- viele wichtige Gütemaße sind in der `broom::glance()`-Funktion vereint:
  - `r.squared` ... Bestimmtheitsmaß / Determinationskoeffizient ( $R^2$ )
  - `adj.r.squared` ... um Parameterzahl korrig. Bestimmtheitsmaß ( $R^2$ )
  - `sigma` ... Std. Abweichung der Residuen ( $\epsilon \sim \mathcal{N}(0, \hat{\sigma}_\epsilon)$ )
  - `statistic` ... F-Wert der ANOVA
  - `p.value` ... p-Wert der ANOVA
  - `deviance` ... Summe der Abweichungsquadrate ( $SQ_{Residuum}$ )

# Ausreißerdiagnostik

## Residuen

- bisher ...
- **einfache Residuen**
  - $\epsilon_i = y_i - \hat{y}_i$
  - abh. von Einheit des Kriteriums
- **standardisierte Residuen**
  - $\epsilon_{i, std} = \epsilon_i / \hat{SE}_\epsilon$
  - Division des Residuums durch seinen Standardfehler
  - abh. von Einheit des Kriteriums
  - kritisch:  $|Werte| > 2-3$
  - enthalten in `broom::augment() : .std.resid` bzw `rstandard()`

# Ausreißerdiagnostik

## Residuen

- studentisierte Residuen
  - Division des Residuums durch den individuellen Standardfehler des Residuums des jeweiligen Falls  $\epsilon_{i,stu} = \epsilon_i / \hat{SE}_{\epsilon_i}$
  - geschätzter SE für jeden einzelnen Fall
  - dieser variiert je nach der Distanz zwischen den Werten der UV des Falles und dem Mittelwert der UV
  - in R: `rstudent(lm_obj2)`

# Ausreißerdiagnostik

## Residuen

- standardisierte vs. studentisierte Residuen:
- Berechnen Sie die statistischen Kennwerte der beiden Residuenarten. Vergleichen Sie diese.
- Plotten Sie die beiden Residuenarten gegeneinander in einem Streudiagramm.

# Ausreißerdiagnostik

## Residuen

- standardisierte vs. studentisierte Residuen
- statistische Kennwerte:

```
tibble(  
  stand = rstandard(lm_obj2),  
  stud = rstudent(lm_obj2)) %>%  
  get_summary_stats() %>%  
  flex()
```

variable	n	min	max	median	q1	q3	iqr	mad	mean	sd	se	ci
stand	32	-2.70	2.26	0.12	-0.70	0.52	1.23	0.91	-0.01	1.04	0.18	0.37
stud	32	-3.13	2.48	0.12	-0.69	0.52	1.21	0.90	-0.02	1.11	0.20	0.40

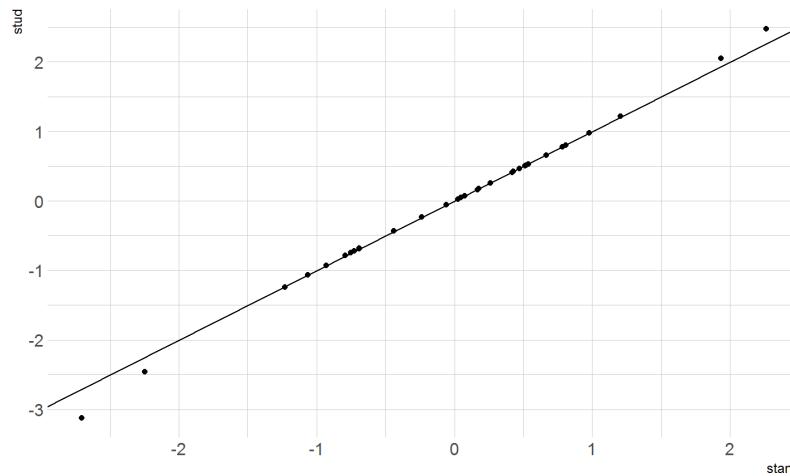
- Standardisiert: MW = 0, SD = 1

# Ausreißerdiagnostik

## Residuen

- Plot:

```
tibble(
  stand = rstandard(lm_obj2),
  stud = rstudent(lm_obj2)) %>%
  ggplot(aes(stand, stud)) +
  geom_point() +
  geom_abline(slope = 1)
```

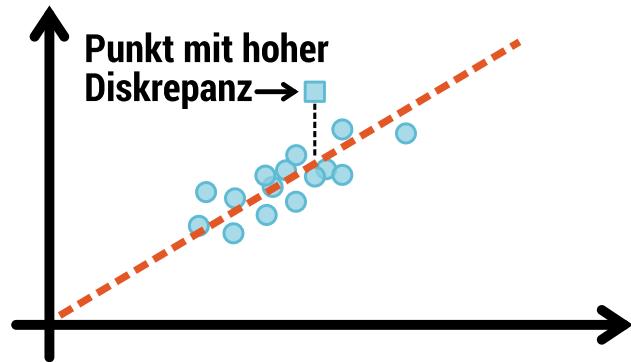


- im mittleren Bereich praktisch identisch
- studentisierte R. extremer an den Verteilungsenden

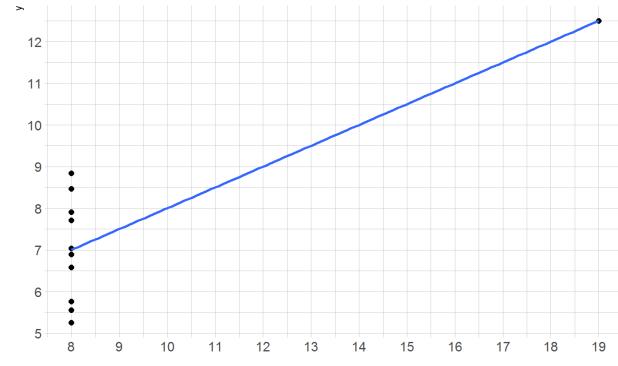
# Ausreißerdiagnostik

## Residuen

- klar identifizierbar:



- aber so (Anscombe Set 4)?



# Ausreißerdiagnostik

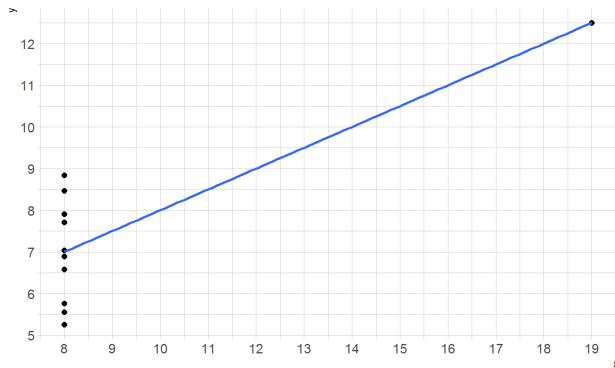
## Residuen

- **ausgeschlossene** Residuen
  - das Residuum, wenn der jeweilige Fall bei der Modellbildung nicht berücksichtigt wurde
  - besonders relevant für Fälle mit starkem Einfluss auf das Modell
  - *ausgeschlossene* Version für alle o.g. Residuenarten
  - "Leave-One-Out-Deletion"
- ✓ studentisierte ausgeschlossene Residuen
  - Die Differenz zwischen dem studentisierten ausgeschlossenen Residuum und dem studentisierten Residuum gibt an, welchen Unterschied die Entfernung eines Falles für dessen eigene Vorhersage bewirkt.
  - in R: `rstandard(lm_obj2, type = 'predictive')`

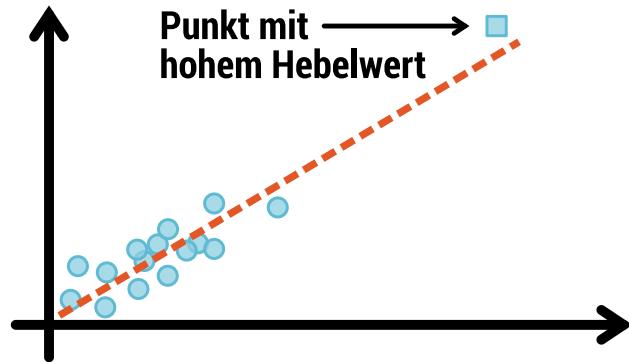
# Ausreißerdiagnostik

## Residuen

- aber so (Anscombe Set 4):



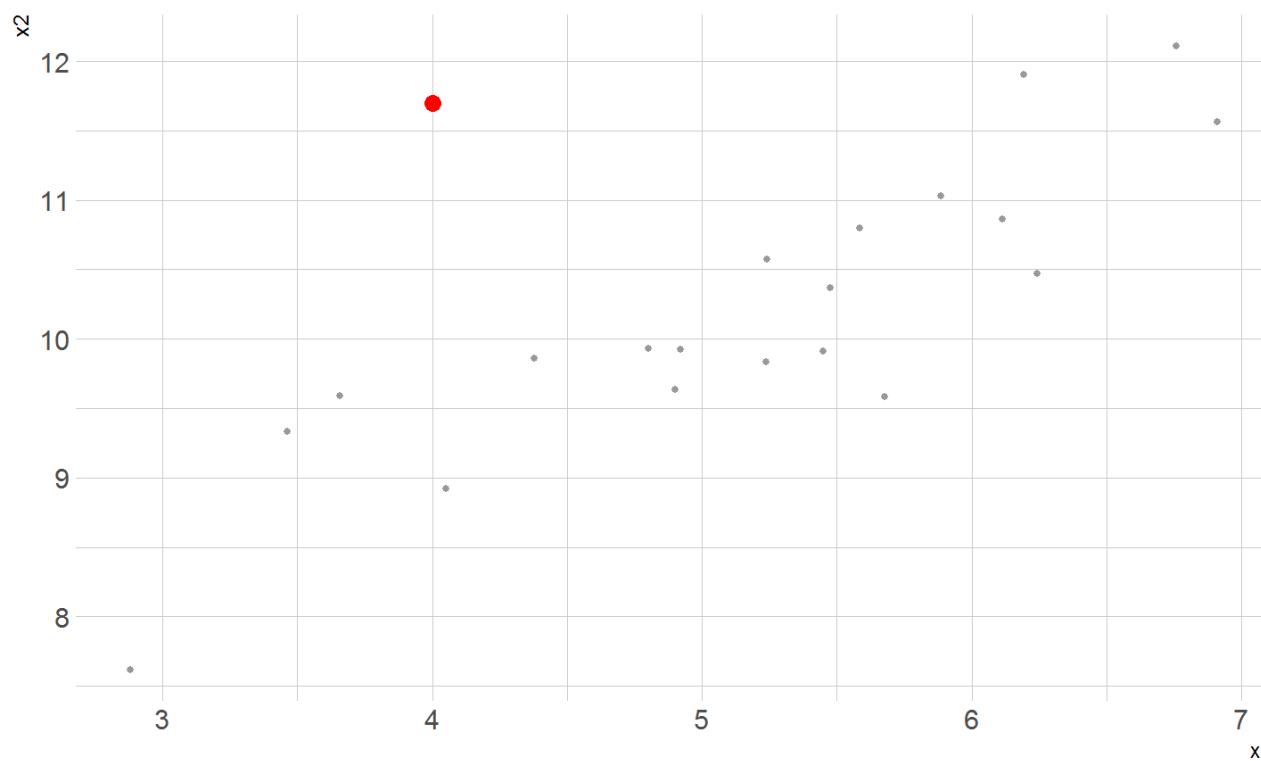
- ungewöhnlicher Prädiktorwert:



# Einflussdiagnostik

## Mahalanobis

- einfache, jeweils univariate Analyse der Prädiktoren?!
- funktioniert nicht:



- Lösung: spezielles (multivariates) Distanzmaß

# Einflussdiagnostik

## Mahalanobis

- Mahalanobis-Abstand
- Auffinden von Fällen mit ungewöhnlichen Wertekombinationen in den UVs
- ggf. mit großem Einfluss auf die Modellschätzung
  - Distanz zwischen zwei Punkten (im mehrdimensionalen Raum)
  - Abstand zweier Punkte in Standardabweichungen
- im Einvariablen-Fall: Abstand Beobachtung zu Mittelwert / Standardabweichung (vgl. z-Standardisierung)

# Einflussdiagnostik

## Mahalanobis

- Erstellen Sie einen normalverteilte Zufallsvariable mit n = 100 (als tibble Variable). Berechnen Sie den Malahanobis-Abstand von Hand.
- `dat <- tibble(x = rnorm(n = 100))`
- `mean(dat$x)`
- `sd(dat$x)`
- `dat %>% mutate(mal_dis = (x-mean(dat$x)) / sd(dat$x))`
- Ergänzen Sie den Datenpunkt x = 3.6 von Hand. Führen Sie die Auswertung noch einmal durch. Wie groß ist der Mahalanobis-Abstand von diesem Wert?

# Einflussdiagnostik

## Mahalanobis

- Mahalanobis-Abstand
- die quadrierten Abstände ( $D^2$ ) sind  $\chi^2$ -verteilt
- mit  $df = \text{Anzahl der Variablen bzw. Dimensionen}$
- entsprechend kann damit ein kritischer Grenzwert bestimmt werden
- dabei üblich ist  $\alpha = 0.01$
- in R: `qchisq(0.999, df = 1)` -> 12.3
- 2 Variablen: 13.8
- 3 Variablen: 16.3
- die Wahrscheinlichkeit eines bestimmten  $D^2$ -Wertes kann entsprechend mit `pchisq(q = ..., df = 1, lower.tail = F)` bestimmt werden.

# Einflussdiagnostik

## Mahalanobis

- Ergänzen Sie das vorangehende Beispiel um die Berechnung der Signifikanz hinsichtlich der Abweichung. Beachten Sie die Quadrierung der Distanz.

```
• set.seed(1)
dat <- tibble(x = rnorm(n = 100))
dat %>%
  #add_row(x = 3.6) %>%
  mutate(mal_dis = (x-mean(dat$x))/sd(dat$x)) %>%
  mutate(mal_dis_sq = mal_dis^2) %>%
  mutate(p_wert = pchisq(q = mal_dis_sq, df = 1, lower.tail = F)) %>%
  flex()
```

x	mal_dis	mal_dis_sq	p_wert
-0.63	-0.82	0.67	p = 0.413
0.18	0.08	0.01	p = 0.934
-0.84	-1.05	1.11	p = 0.293
1.60	1.65	2.74	p = 0.098
0.33	0.25	0.06	p = 0.806
-0.82	-1.03	1.07	p = 0.301
0.49	0.42	0.18	p = 0.673
0.74	0.70	0.49	p = 0.483
0.58	0.52	0.27	p = 0.603
-0.31	-0.46	0.21	p = 0.645
1.51	1.56	2.44	p = 0.118
0.39	0.31	0.10	p = 0.754
-0.62	-0.81	0.66	p = 0.416
-2.21	-2.59	6.69	p = 0.010
1.12	1.13	1.28	p = 0.258
-0.04	-0.17	0.03	p = 0.864
-0.02	-0.14	0.02	p = 0.889
0.94	0.93	0.86	p = 0.353
0.82	0.79	0.63	p = 0.428
0.59	0.54	0.29	p = 0.589
0.92	0.90	0.81	p = 0.367
0.78	0.75	0.56	p = 0.454
0.07	0.04	0.00	p = 0.970

# Einflussdiagnostik

## Mahalanobis

- in R:
  - `rstatix::mahalanobis_distance(df_cars)`
  - berechnet direkt die *quadrierten* Distanzen
  - Ergebnis der inferenzstatistischen Prüfung
- Im Mehrvariable Fall werden die Distanzen zum Zentroid (der multivariaten Verteilung) bestimmt.
- Wenden Sie die Funktion auf das Beispiel an.

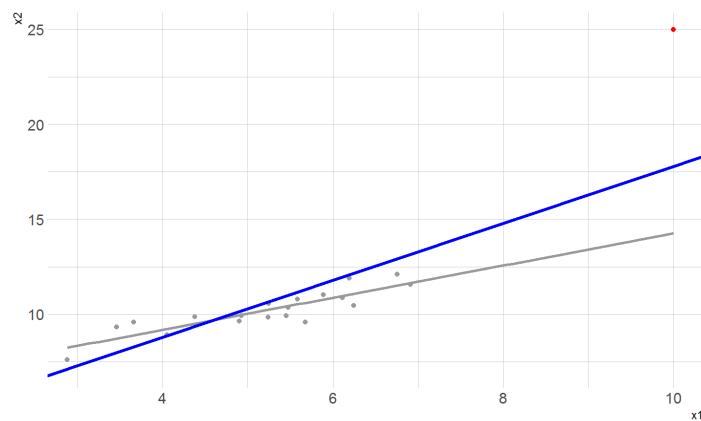
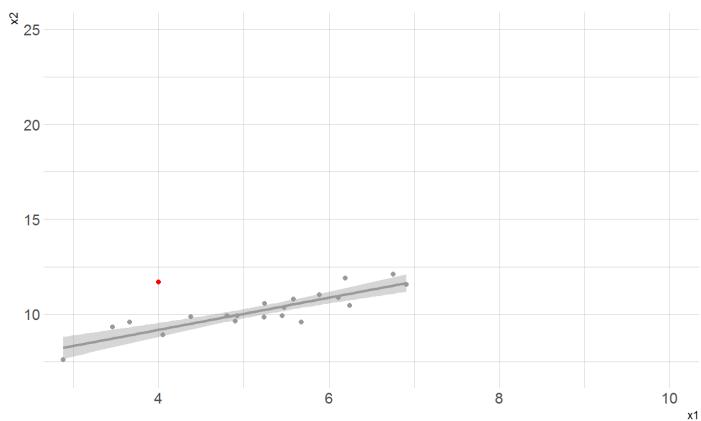
```
• dat %>%
  add_row(x = 3.6) %>%
  mahalanobis_distance()
```

# Einflussdiagnostik

## Mahalanobis

- eine große (Mahalanobis-) Distanz allein ist nicht zwingend mit einer Verzerrung des Regressionsmodells verbunden

- 



# Einflussdiagnostik

## Hebelwert

- zentrierte Hebelwerte  $h$
- Maß für den Einfluss eines Punktes auf die Anpassung der Regression
- Werte zwischen 0 und 1 bzw.  $1 - \frac{1}{N}$ 
  - 0 ... keinen Einfluss auf die Vorhersage
  - 1 ... Vorhersage wird vollständig durch diesen einen Wert bestimmt
- *leverage*

# Einflussdiagnostik

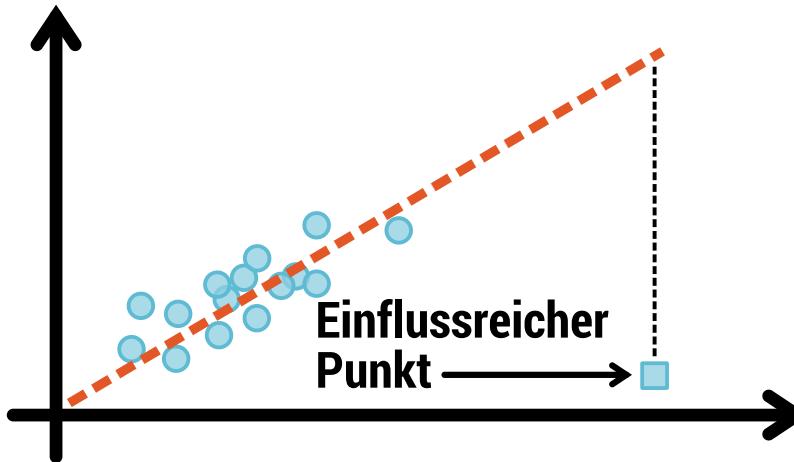
## Hebelwert

- direkte Beziehung zu Mahalanobis-Distanz:
  - $D^2 = (n - 1)(h_{ii} - \frac{1}{n})$
- in R:
  - `hatvalues(lm_obj)`
  - `olsrr::ols_leverage(lm_obj)`
  - `broom::augment, .hat`
- krit. Grenzwerte:
- Prädiktoren k und der Anzahl der Fälle n.
  - 0.2 (Huber, 1981)
  - $(2*k)/n$ , wenn  $n-p > 50$  (Igo, 2010)
  - $(3*k)/n$ , wenn  $k > 6$  und  $n-k > 12$  (Velleman & Welsch, 1981)
  - $(3*(k+1))/n$ , (Frees, 2010)

# Einflussdiagnostik

## Cooks D

- einflussreiche Fälle
- Kombination aus:
  - großes Residuum
  - hoher Hebelwert
- 



- Maßzahl für die Stärke der Änderung des Modells bei Ausschluss des jeweiligen Falls
- je größer der Wert, umso stärker der Einfluss des Falls auf das Regressionsmodell
- Kennzahl: **Cook Distanz** (Cook's D)

# Einflussdiagnostik

## Cooks D

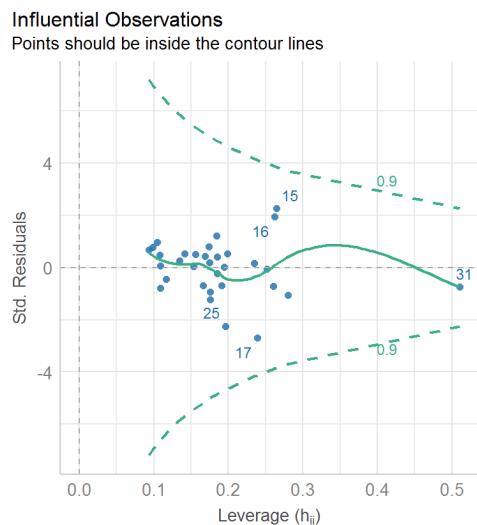
- in R:
  - `cooks.distance(lm_obj2)`
  - `broom::augment(lm_obj2)`, *.cooksD*
- kritische Werte:
  - 1 (Weisberg, 1985)
  - 50%-Quantil der F-Verteilung (`qf(0.5, 5, 26)`)
  - $4/N$

# Einflussdiagnostik

## Cooks D

- in R:

```
- performance::check_outliers(lm_obj2)  
- performance::check_model(lm_obj2, check = 'outliers')
```



```
- olsrr::ols_plot_cooksd_chart(lm_obj2)
```

# Einflussdiagnostik

## Cooks D

- im Fall von Ausreißern
  - Kontrolle der Werte (Plausibilität, Eingabefehler)
  - Ausschluss der Fälle
  - Modellrespezifikation
  - ✓ Sensitivitätsanalyse
  - Signifikanztests mit robusten Standardfehlern (HC4)

# Einflussdiagnostik

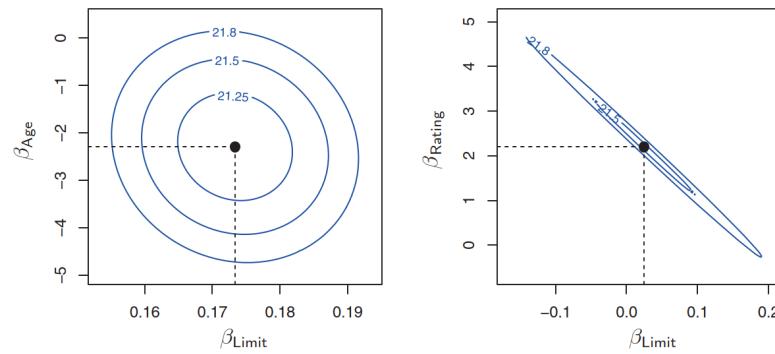
## DFFits, DFBetas

- weitere Maße:
  - DFFits
  - DFBetas
- Auswirkung eines Fallausschlusses auf Modellgüte (DFFits) und einzelne Koeffizienten (DFBetas)
- in R:

```
olsrr::ols_plot_dffits(lm_obj2)
olsrr::ols_plot_dfbetas(lm_obj2)
```

# Multikollinearität

- Multikollinearität ... Prädiktoren sind untereinander hoch korreliert
- Konsequenzen:
  - Regressionskoeffizienten können nicht mehr zuverlässig geschätzt werden
  - Standardschätzfehler der Koeffizienten nimmt massiv zu (Inflation)
- Auswirkung, graphisch:



# Multikollinearität

- Bewertung mittels Varianzinflationsfaktor:
  - Berechnung von  $k$  Regressionsmodellen
  - jeweils eine Prädiktorvariable  $X_k$  wird durch alle anderen X vorhergesagt
  - $\text{VIF} = \frac{1}{1-R_j^2}$
- Berechnug von Hand:
  - `lm(hp ~ disp + cyl + wt, data = df_cars) %>% glance()`
  - $1/(1-0.717) = 3.53$
- Bewertung:
  - VIF < 5 ... unproblematisch
  - VIF < 10 ... moderat
  - VIF > 10 ... kritisch

# Multikollinearität

- in R:

```
performance::check_collinearity(lm_obj2) %>% flex()
```

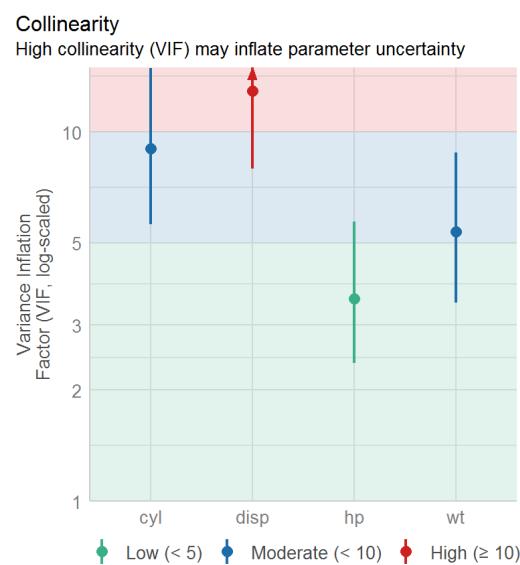
Term	VIF	VIF_CI_low	VIF_CI_high	SE_factor	Tolerance	Tolerance_CI_low	Tolerance_CI_high
hp	3.53	2.36	5.71	1.88	0.28	0.18	0.42
disp	12.90	7.94	21.42	3.59	0.08	0.05	0.13
cyl	8.99	5.61	14.86	3.00	0.11	0.07	0.18
wt	5.37	3.45	8.78	2.32	0.19	0.11	0.29

- Toleranz =  $1/VIF$
- `SE_factor = sqrt(VIF)`
- `cor(df_cars$hp, df_cars$disp)`

# Multikollinearität

- in R:

```
performance::check_model(lm_obj2, check = 'vif')
```



# Multikollinearität

- VIF ist problematisch bei kategorialen Prädiktoren
  - VIF abhängig von Referenzkategorie
- modifizierter VIF: Generalisierter VIF
- in R:

```
car::vif(lm_obj2) %>% flex()
```

Var1	Var2		
	GVIF	Df	GVIF^(1/(2*Df))
hp	3.531254	1	1.879163
disp	12.900887	1	3.591780
cyl	8.993015	2	1.731715
wt	5.368271	1	2.316953

- `GVIF` ... entspricht klassischem VIF
- `GVIF^(1 / (2*Df))` ... adjustierter VIF
- `GVIF^(1 / (2*Df))` quadrieren für obigen Bewertungsmaßstab

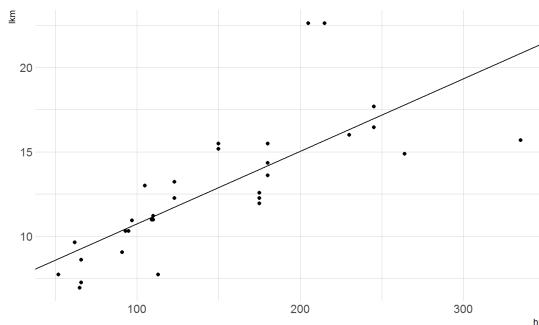
# Prädiktortransformationen

- Auch Transformationen von Prädiktoren möglich
- bisher: Zur Linearisierung der Beziehung zwischen Y und X
  - Vgl. Einfache Lineare Regression, Modellannahmen
  - $\sqrt{X}$ ;  $\log X$ ;  $X^2$ ;  $e^X$ ;  $\frac{1}{X}$ ;  $e^{-X}$
  - Originalprädiktorvariable wird ersetzt
- Aber transformierte Variable auch als *zusätzlichen* Prädiktor:

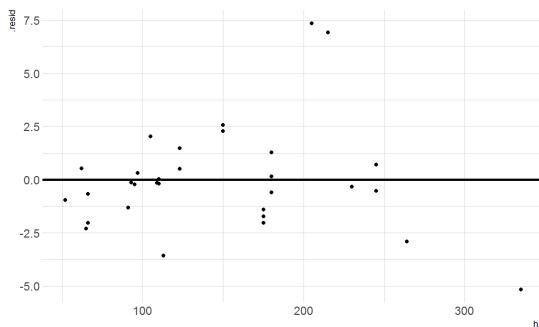
# Prädiktortransformationen

## Beispiel

- nochmal die einfache Regression von Verbrauch auf Hubraum:



- und die Modellresiduen:



- sehen Sie auch das Muster?!
- leicht kurvilineare Beziehung

- Koeffizienten:

```
lm(lkm ~ hp, data = df_cars) %>%
  tidy() %>%
  flex()
```

term	estimate	std.error	statistic	p.value
(Intercept)	6.45	1.07	6.01	p < .001
hp	0.04	0.01	6.46	p < .001

- Modellgüte:

```
lm(lkm ~ hp, data = df_cars) %>%
  glance() %>%
  flex()
```

r.squared	adj.r.squared	sigma	statistic	p.value	df	logLik	AIC	BIC	deviance
0.58	0.57		2.54	41.79	p < .001	1	-74.19	154.37	158.77

# Prädiktortransformationen

## Polynom

- Modellierung mit *zusätzlichem* quadratischem Term!
- Polynom 2. Grades:

```
# bisher:  
m1 <- lm(lkm ~ hp, data = df_cars)  
  
# neu:  
m2 <-  
  df_cars %>%  
  mutate(hp_sq = hp^2) %>%  
  lm(lkm ~ hp + hp_sq, data = .)
```

- Ergebnis:

```
bind_rows(glance(m1), glance(m2), .id = 'Modell') %>% flex()
```

Modell	r.squared	adj.r.squared	sigma	statistic	p.value	df	logLik	AIC	BIC	deviance	df.residual	nobs
1	0.58	0.57	2.54	41.79	p < .001	1	-74.19	154.37	158.77	193.35	30	32
2	0.68	0.66	2.26	30.97	p < .001	2	-69.86	147.72	153.59	147.55	29	32

# Prädiktortransformationen

## Beispiel

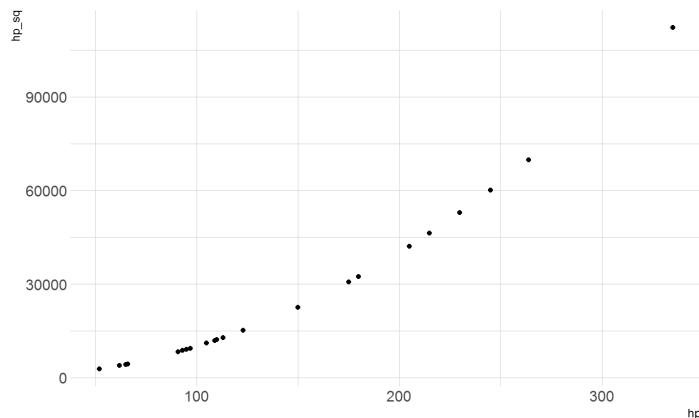
- Was ist das Problem?
- Korrelation der Prädiktoren:

```
df_cars %>%
  mutate(hp_sq = hp^2) %>%
  cor_test(hp, hp_sq) %>%
  select(3:5) %>%
  flex()
```

cor	statistic	p
0.97	23.06	0

- im Streudiagramm:

```
df_cars %>%
  mutate(hp_sq = hp^2) %>%
  ggplot(aes(hp, hp_sq)) +
  geom_point()
```



- hohe Multikollinearität!
- VIF:

```
car::vif(m2)
```

```
##          hp      hp_sq
## 18.72562 18.72562
```

# Prädiktortransformationen

## orthogonale Polynome

- statt einfachem polynomialem Term: *orthogonaler polynomialer Term*
- quadratischen Term mit `poly()` erstellen

```
poly(df_cars$hp, 2) %>%  
  head(5) %>%  
  flex()
```

Var1	Var2	
	1	2
A	-0.09610551	-0.04231779
B	-0.09610551	-0.04231779
C	-0.14063822	0.03477520
D	-0.09610551	-0.04231779
E	0.07416660	-0.16657799

- ?

# Prädiktortransformationen

## orthogonale Polynome

- im Detail:

```
df_ortho_pol <-  
  df_cars %>%  
  transmute(  
    hp_1 = hp,  
    hp_2 = hp^2,  
    op_hp_1 = poly(hp, 2) [,1],  
    op_hp_2 = poly(hp, 2) [,2])
```

- Korr.matrix:

```
df_ortho_pol %>%  
  cor_mat() %>%  
  flex_align(content = 'center')
```

rowname	hp_1	hp_2	op_hp_1	op_hp_2
hp_1	1.00	0.97	1.00	0.00
hp_2	0.97	1.00	0.97	0.23
op_hp_1	1.00	0.97	1.00	0.00
op_hp_2	0.00	0.23	0.00	1.00

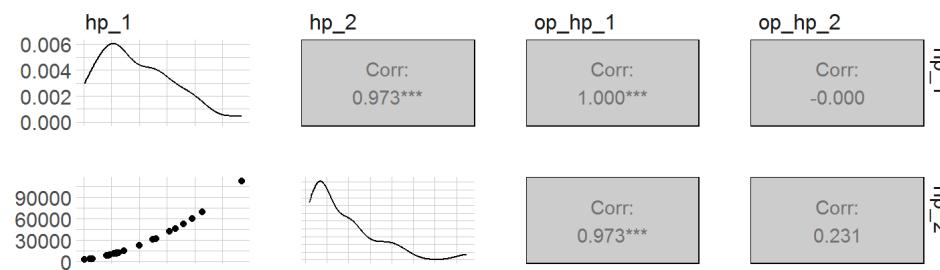
# Prädiktortransformationen

## orthogonale Polynome

- Zusammenhänge zwischen den Termen:

```
GGally::ggpairs(df_ortho_pol)
```

```
##  plot: [1, 1] [=====>-----  
##  plot: [1, 2] [=====>-----  
##  plot: [1, 3] [=====>-----  
##  plot: [1, 4] [=====>-----  
##  plot: [2, 1] [=====>-----  
##  plot: [2, 2] [=====>-----  
##  plot: [2, 3] [=====>-----  
##  plot: [2, 4] [=====>-----  
##  plot: [3, 1] [=====>-----  
##  plot: [3, 2] [=====>-----  
##  plot: [3, 3] [=====>-----  
##  plot: [3, 4] [=====>-----  
##  plot: [4, 1] [=====>-----  
##  plot: [4, 2] [=====>-----  
##  plot: [4, 3] [=====>-----  
##  plot: [4, 4] [=====>-----  
##
```



# Prädiktortransformationen

## orthogonale Polynome

- Regressionsmodelle im Vergleich:

```
# klassisch
m1 <- lm(lkm ~ hp, data = df_cars)

# kollineares Polynom
m2 <-
  df_cars %>%
  mutate(hp_sq = hp^2) %>%
  lm(lkm ~ hp + hp_sq, data = .)

# orthogonales Polynom
m3 <-
  df_cars %>%
  mutate(
    op_hp_1 = poly(hp, 2)[,1],
    op_hp_2 = poly(hp, 2)[,2]) %>%
  lm(lkm ~ op_hp_1 + op_hp_2, data = .)

# Kurzform:
m3 <- lm(lkm ~ poly(hp, 2), data = df_cars)
```

# Prädiktortransformationen

## orthogonale Polynome

- Ergebnis, Modellgüte:

```
bind_rows(  
  glance(m1),  
  glance(m2),  
  glance(m3),  
  .id = 'Modell'  
) %>%  
  flex()
```

Modell	r.squared	adj.r.squared	sigma	statistic	p.value	df	logLik	AIC	BIC	deviance	df.residual	nobs
1	0.58	0.57	2.54	41.79	p < .001	1	-74.19	154.37	158.77	193.35	30	32
2	0.68	0.66	2.26	30.97	p < .001	2	-69.86	147.72	153.59	147.55	29	32
3	0.68	0.66	2.26	30.97	p < .001	2	-69.86	147.72	153.59	147.55	29	32

- Modellgüten von m2 und m3 identisch

# Prädiktortransformationen

## orthogonale Polynome

- Ergebnis, Koeffizienten:

```
bind_rows(
  tidy(m1),
  tidy(m2),
  tidy(m3),
  .id = 'Modell'
) %>%
  mutate(se_est_ratio = std.error/estimate) %>%
  as_grouped_data('Modell') %>%
  flex()
```

Modell	term	estimate	std.error	statistic	p.value	se_est_ratio
1	(Intercept)	6.45	1.07	6.01	p < .001	0.17
	hp	0.04	0.01	6.46	p < .001	0.15
2	(Intercept)	1.14	2.01	0.57	p = 0.573	1.75
	hp	0.12	0.03	4.60	p < .001	0.22
	hp_sq	0.00	0.00	-3.00	p = 0.005	-0.33
3	(Intercept)	12.76	0.40	31.99	p < .001	0.03
	poly(hp, 2)1	16.41	2.26	7.28	p < .001	0.14
	poly(hp, 2)2	-6.77	2.26	-3.00	p = 0.005	-0.33

- in m3 kleinere Standardfehler (-verhältnisse)

# Modellprüfung

## Modellvergleich

- aber ist das Modell mit quadratischem Term auch *statistisch signifikant* besser?
- bisher: Prüfung der Modells gegen das **Nullmodell**
- mittels `anova()`
- in R:

```
lm_obj_null <- lm(lkm ~ 1, data = df_cars)
anova(lm_obj_null, lm_obj2) %>% tidy() %>% flex()
```

term	df.residual	rss	df	sumsq	statistic	p.value
lkm ~ 1	31	462.66				
lkm ~ hp + disp + cyl + wt	26	68.55	5	394.11	29.89	p < .001

- Werte entsprechen der Ausgabe von `glance()`

# Modellprüfung

## Modellvergleich

- Erweiterung:
  - Modelle gegeneinander prüfen:

```
anova(m1, m3) %>% tidy() %>% flex()
```

term	df.residual	rss	df	sumsq	statistic	p.value
lkm ~ hp	30	193.35				
lkm ~ poly(hp, 2)	29	147.55	1	45.8	9	p = 0.005

- F-Test:  $F = \frac{(RSS_1 - RSS_2)/(df_1 - df_2)}{RSS_2/df_2}$
- $((193.4 - 147.6) / (30 - 29)) / (147.6/29) = 45.8 / (147/29) = 9.0$
- `pf(9, 1, 29, lower = F)`
- $p < 0.05$ , Modell mit quadratischem Term sig. besser!

# Modellprüfung

## Modellvergleich

- weniger formaler Vergleich:
- Vergleich der logLik, AIC, BIC-Werte

```
bind_rows(  
  glance(m1),  
  glance(m2),  
 .id = 'Modell') %>%  
 flex()
```

Modell	r.squared	adj.r.squared	sigma	statistic	p.value	df	logLik	AIC	BIC	deviance	df.residual	nobs
1	0.58	0.57	2.54	41.79	p < .001	1	-74.19	154.37	158.77	193.35	30	32
2	0.68	0.66	2.26	30.97	p < .001	2	-69.86	147.72	153.59	147.55	29	32

- Werte näher an 0 zeigen bessere Modellgüte
- alle drei Werte nur für internen Vergleich geeignet (gleiche Daten)

# Modellprüfung

## Informationskriterien

- AIC = Akaike Information Criterion
- BIC = Bayes'sches Information Criterion
- Setzen Anpassungsgüte des Modells ins Verhältnis zur Parameterzahl
- *Occrams Razor, parsimony, KISS-Prinzip*
  - Parameterzahl wird "strafend" berücksichtigt
  - kleinere Werte zeigen besseres Modell an
- $AIC = -2 \times \ell(\hat{\theta}_{ML}) + 2 \times p$
- $BIC = -2 \times \ell(\hat{\theta}_{ML}) + \ln(n) \times p$
- Unterschied AIC zu BIC
  - ab n=8 straft das BIC zusätzliche Parameter schärfer als das AIC
- für Berechnung: `logLik(m1)`
  - AIC: `-2 * -74.18648 + 2 * 3`
  - BIC: `-2 * -74.18648 + log(32) * 3`

# Formelsyntax in R

- R bietet eine umfangreiche Formelnotation für statistische Modelle

Operator	Beispiel	Beschreibung
+	<code>y ~ x1 + x2</code>	Einfluss einzelner Variablen
-	<code>y ~ . - x1</code>	Variable aus Formel löschen
*	<code>y ~ x1 * x2</code>	Interaktion und direkter Effekte
:	<code>y ~ x1 : x2</code>	nur Interaktion
^	<code>y ~ (x1 + x2 + x3)^3</code>	alle möglichen Interaktionen und direkter Einfluss
-1 / +0	<code>y ~ x1 + 0</code>	kein Intercept

- Transformationen sind direkt in Formel möglich

- `y ~ log(x1) + sqrt(x2)`
- `y ~ poly(x, 2)`
- `y ~ exp(x)`

# Formelsyntax in R

- Vgl das Beispiel mit dem quadratischen Prädiktor:

```
lm(lkm ~ hp, data = df_cars)
lm(lkm ~ hp + hp^2, data = df_cars) # geht nicht :(
```

- ⚠ Genannte Operatoren haben in der Formelsyntax eine besondere Bedeutung
  - Inhibit-Operator ermöglicht “klassische” Interpretation der Operatoren
  - `I()`
  - `y ~ x1 + I(x2^2)`
- im Beispiel des quadratischen Prädiktors:

```
# so gehts!
lm(lkm ~ hp + I(hp^2), data = df_cars)
```

# Formelsyntax in R

- `model.matrix()` zeigt die intern für die Modellierung verwendeten Variablen (nur UV)
- `model.frame()` zeigt die intern für die Modellierung verwendeten Variablen (inkl. AV)
- Vergleichen Sie die Ausgaben der beiden Funktionen.
- Vergleichen Sie die Modelle der folgenden Formeln:
  - `lm(lkm ~ hp , data = df_cars)`
  - `lm(lkm ~ hp -1 , data = df_cars)`
  - `lm(lkm ~ hp+disp , data = df_cars)`
  - `lm(lkm ~ hp*disp , data = df_cars)`
  - `lm(lkm ~ hp:disp , data = df_cars)`
  - `lm(lkm ~ hp+disp+hp:disp , data = df_cars)`
  - `lm(lkm ~ hp+log(disp) , data = df_cars)`
  - `lm(lkm ~ 2*hp , data = df_cars)`
  - `lm(lkm ~ I(2*hp) , data = df_cars)`
  - `lm(lkm ~ hp+hp^2 , data = df_cars)`
  - `lm(lkm ~ hp+I(hp^2) , data = df_cars)`
  - `lm(lkm ~ I((hp+disp)^2) , data = df_cars)`

# Variablenelektion

- bisher: bivariate Streudiagramme / Korrelationen

```
df_cars %>%
  select(-Auto) %>%
  GGally::ggpairs()
```

# Variablenelektion

- einfach mal alle probieren?!
- *Best Subset Regression*
- `ols_step_best_subset(lm(lkm ~ ., data= df_cars))`
- so einfach gehts nicht :)
- `subsets <- ols_step_best_subset(lm(lkm ~ . - Auto, data= df_cars))`

# Variablenelektion

- primäre Auswahlkriterien:
  - adjustiertes R<sup>2</sup>: höchster Wert
  - AIC: niedrigster Wert
  - Mallows Cp-Statistik: Wert am nächsten an p+1 liegt
- Problem: Anzahl zu prüfender Modelle steigt exponentiell  $2^p$ 
  - 10 Prädiktoren = 1024 Modelle
  - 11 Prädiktoren = 2048 Modelle
  - 20 Prädiktoren = ~ 1 Mio Modelle

# Variablenelektion

## Schrittweise Verfahren

- 2 grundlegende Verfahren
- **Vorwärts-Selektion**
  - beginn mit Nullmodell
  - alle Prädiktoren werden einzeln getestet
  - Prädiktor mit "bestem" Wert wird aufgenommen
  - als Kriterium verschiedene Maße: AIC, adj R<sup>2</sup>, p-Wert
  - im nächsten Schritt werden wiederrum alle übrigen Prädiktoren einzeln getestet
  - Fortsetzung bis Grenzwert nicht mehr erreicht wird

# Variablenelektion

## Vorwärts-Selektion

- in R:
- `olsrr::ols_step_forward_p()`
- `olsrr::ols_step_forward_****()`
- ```
fwd_sel <- ols_step_forward_p(lm(lkm ~ . - Auto, data= df_cars), details = T)
fwd_sel
plot(fwd_sel)
```
- `hierarchical`-Argument
- `include`-Argument
- `exclude`-Argument

# Variablenelektion

## Rückwärts-Selektion

- Rückwärts-Selektion
  - beginn mit Modell mit allen Prädiktoren
  - derjenige Prädiktor mit dem "schlechtesten" Kriterienwert wird entfernt
  - z.B. der Prädiktor mit dem höchsten p-Wert
  - neues Modell unter Ausschluss des Prädiktors berechnen
  - so lange Fortsetzen bis Grenzwert nicht mehr überschritten wird

# Variablenelektion

## Rückwärts-Selektion

- in R:
- `olsrr::ols_step_backward_p()`
- `olsrr::ols_step_backward_****()`

```
bwd_sel <- ols_step_backward_p(lm(lkm ~ . - Auto, data= df_cars), details = T)
bwd_sel
plot(bwd_sel)
```

# Variablenelektion

## Gemischte-Selektion

- Mischung aus Vorwärts- und Rückwärts-Selektion
- Prädiktoren können auch wieder ausgeschlossen werden

# Variablenelektion

## Gemischte-Selektion

- in R:
- `olsrr::ols_step_both_p()`
- `olsrr::ols_step_both_****()`
- ```
both_sel <- ols_step_both_p(lm(lkm ~ . - Auto, data= df_cars), details = T)
both_sel
plot(both_sel)
```

# Prädiktion

## predict()

- neues Fahrzeug, mit: hp = 162, disp = 158, cyl = 6, wt = 1.4
- Einsetzen neuer Werte in Regressionsgleichung
- Regressionskoeffizienten:

term	estimate	std.error	statistic	p.value
(Intercept)	2.24	1.39	1.61	p = 0.120
hp	0.01	0.01	1.72	p = 0.097
disp	0.01	0.01	0.70	p = 0.492
cyl6	0.08	0.96	0.08	p = 0.935
cyl8	0.02	1.84	0.01	p = 0.990
wt	4.87	1.52	3.20	p = 0.004

-  Berechnen Sie den (geschätzten) Kraftstoffverbrauch dieses Fahrzeugs!
- $2.24 + 0.0137 \cdot 162 + 0.0059 \cdot 158 + 0.0786 \cdot 1 + 4.87 \cdot 1.48 = 12.68$

# Prädiktion

## predict()

- in R mittels `predict()` und `newdata`-Argument

```
tribble(  
  ~hp, ~disp, ~cyl, ~wt,  
  162, 158, 6, 1.48,  
) %>%  
  mutate(cyl = factor(cyl)) %>% #als Faktor!  
  predict(lm_obj2, newdata = .)
```

```
##      1  
## 12.68939
```

- prinzipiell auch gleich mehrere Vorhersagen möglich

# Prädiktion

## augment()

- stattdessen mit `augment()`:
- gibt zusätzlich noch die Prädiktorwerte mit aus

```
tribble(  
  ~hp, ~disp, ~cyl, ~wt,  
  162, 158, 6, 1.48,  
  mean(df_cars$hp), mean(df_cars$disp), 8, mean(df_cars$wt),  
 ) %>%  
  mutate(cyl = factor(cyl)) %>% #als Faktor!  
  augment(lm_obj2, newdata = .) %>%  
  flex()
```

hp	disp	cyl	wt	.fitted
162.00	158.00	6	1.48	12.69
146.69	230.72	8	1.46	12.75

# Prädiktion

## KI

- zusätzlich mit Konfidenzintervall

```
tribble(
  ~hp, ~disp, ~cyl, ~wt,
  162, 158, 6, 1.48,
  mean(df_cars$hp), mean(df_cars$disp), 8, mean(df_cars$wt),
) %>%
  mutate(cyl = factor(cyl)) %>% #als Faktor!
  augment(lm_obj2,
    newdata = .,
    se_fit = T,
    interval = 'confidence',
    conf.level = .95
  ) %>%
  flex()
```

hp	disp	cyl	wt	.fitted	.lower	.upper	.se.fit
162.00	158.00	6	1.48	12.69	11.12	14.26	0.76
146.69	230.72	8	1.46	12.75	10.78	14.73	0.96

# Ergebnisbericht

- Und was nun berichten?
- Und wie?
- Vielzahl an Möglichkeiten!
  - `jtools::summ(lm_obj2)`
  - `gtsummary::tbl_regression(lm_obj2)`
  - `stargazer::stargazer(lm_obj2, type = 'text')`
  - `sjPlot::tab_model(lm_obj2)`
- mit jeweils vielfältigen zusätzlichen Optionen und Ergänzungen
- je nach Zielgruppe / Dateiformat

# Ergebnisbericht

- Vergabe von Labels (am Beginn des Skripts, vor den Analysen)

```
df_cars_lbld <-
  df_cars %>%
  labelled::set_variable_labels(
    lkm   = 'Verbrauch [l/100 km]',
    cyl   = 'Zylinderzahl',
    wt    = 'Gewicht [t]',
    hp    = 'Leistung [PS]',
    disp  = 'Hubraum [ccm]'
)
lm_obj <- lm(lkm ~ hp, data = df_cars_lbld)
lm_obj2 <- lm(lkm ~ hp + cyl + wt + disp, data = df_cars_lbld)
```

# Ergebnisbericht

- am gebräuchlichsten Word
- Kombination von `gtsummary`, zum Erstellen der Tabellen
- und `flextable` zur Konvertierung ins docx-Format

```
lm_obj2 %>%
  gtsummary::tbl_regression() %>%
  gtsummary::add_global_p(keep = T) %>%
  gtsummary::add_n() %>%
  gtsummary::add_vif() %>%
  gtsummary::add_glance_source_note() %>%
  gtsummary::as_flex_table()
```

# Ergebnisbericht

- auch mehrere Modelle im Vergleich

```
tbl_merge(  
  list(  
    tbl_regression(lm_obj),  
    tbl_regression(lm_obj2)),  
  tab_spanner = c('**Modell 1**', '**Modell 2**'))  
)
```

# Ergebnisbericht

- für LaTeX / PDF Berichte:

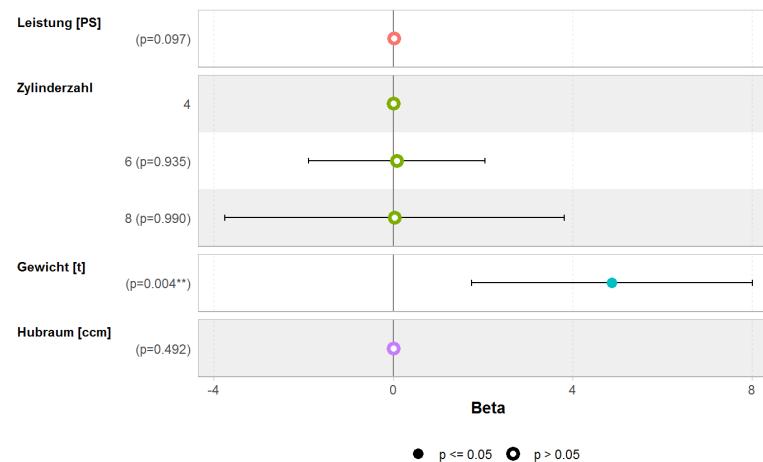
```
stargazer::stargazer(  
  lm_obj, lm_obj2,  
  type = "text",  
  ci = T,  
  title = "Linear Regression Results",  
  dep.var.caption = "Dependent variable")
```

# Visualisierung

## Koeffizienten

- ergänzend zu Tabellen
- Visualisierung der Koeffizienten:

```
ggstats:::ggcoef_model(lm_obj2)
```

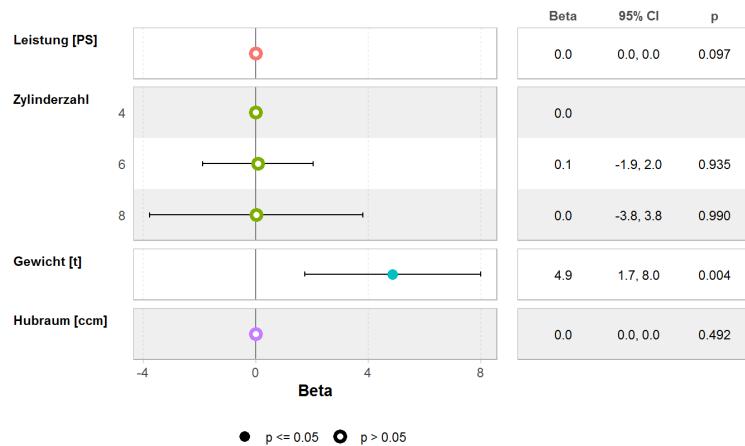


# Visualisierung

## Koeffizienten

- Visualisierung der Koeffizienten zusätzlich mit Werten:

```
ggstats:::ggcoef_table(lm_obj2)
```

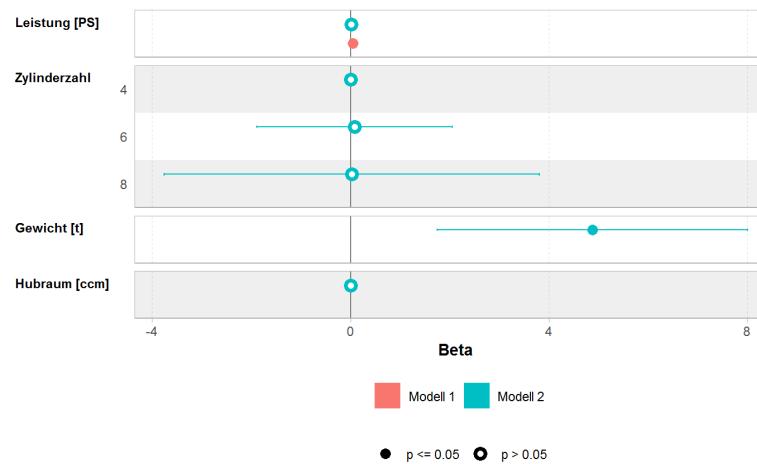


# Visualisierung

## Koeffizienten

- Visualisierung der Koeffizienten von Modellen im Vergleich:

```
ggstats:::ggcoef_compare(list(  
  "Modell 1" = lm_obj,  
  "Modell 2" = lm_obj2))
```

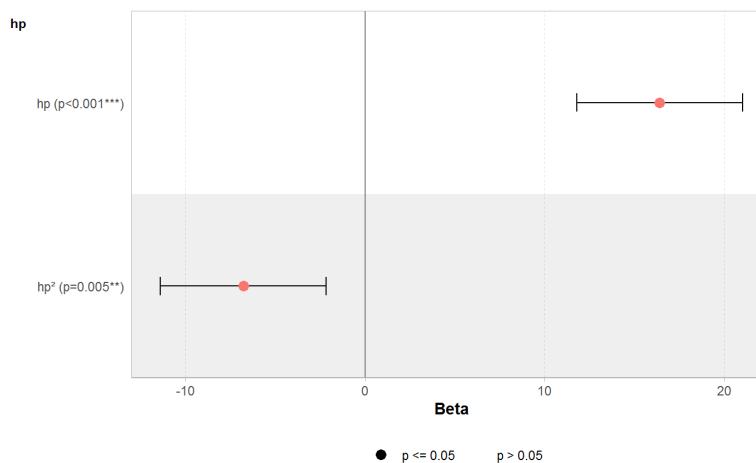


# Visualisierung

## Koeffizienten

- auch orthogonale Polynome werden erkannt

```
ggstats:::ggcoef_model(m3)
```



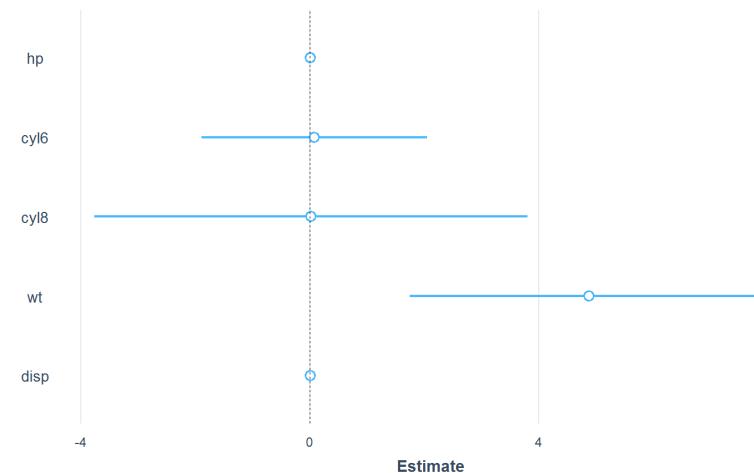
- viele weitere Optionen verfügbar: `vignette("ggcoef_model")`

# Visualisierung

## Koeffizienten

- Visualisierung mit dem `jtools`-Package

```
jtools::plot_coefs(lm_obj2)
```

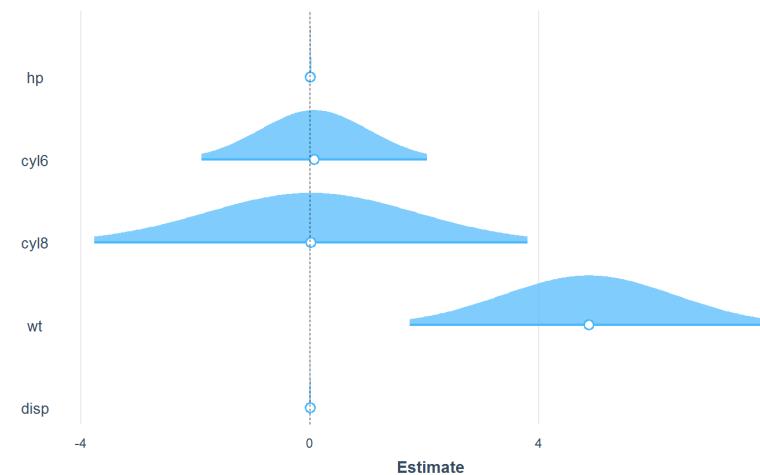


# Visualisierung

## Koeffizienten

- Visualisierung mit Unsicherheit des Schätzers

```
jtools::plot_coefs(  
  lm_obj2,  
  plot.distributions = T,  
  rescale.distributions = T)
```

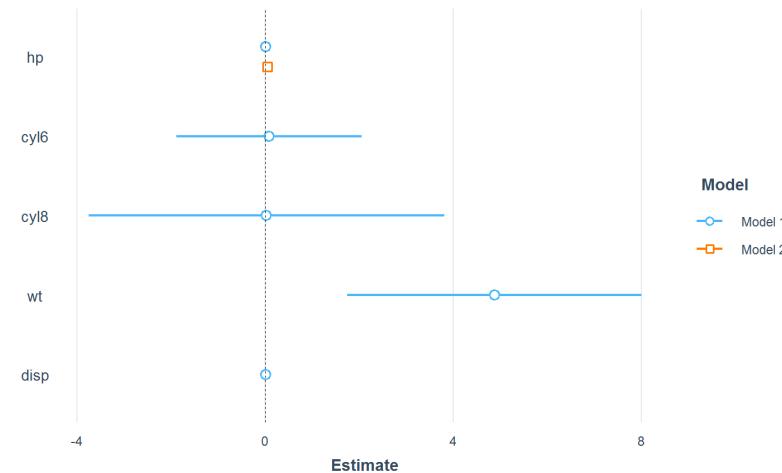


# Visualisierung

## Koeffizienten

- Visualisierung von Modellvergleichen

```
jtools::plot_coefs(  
  lm_obj2, lm_obj)
```

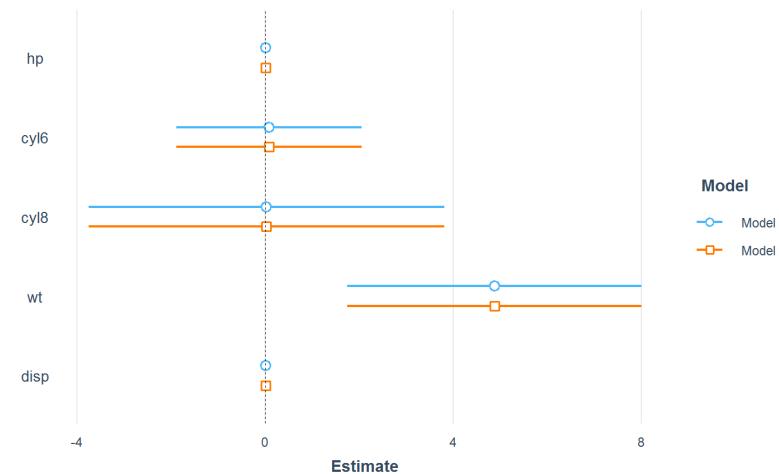


# Visualisierung

## Koeffizienten

- Visualisierung von HCC-Schätzern

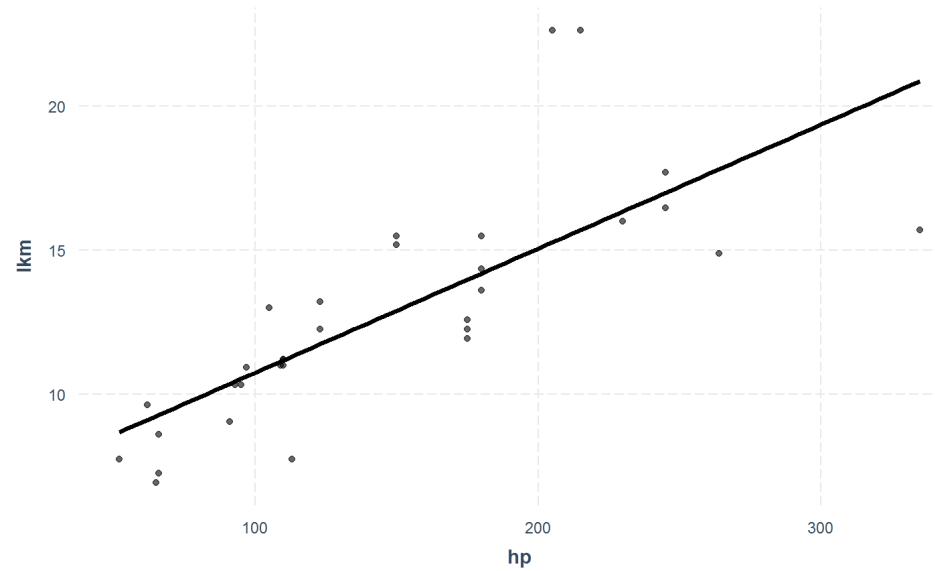
```
jtools::plot_coefs(  
  lm_obj2, lm_obj2,  
  robust = list(F, 'HC3'))
```



# Visualisierung der Effekte

- Visualisierung der Vorhersagen für verschiedene Prädiktorwerte
- für die einfache lineare Regression:

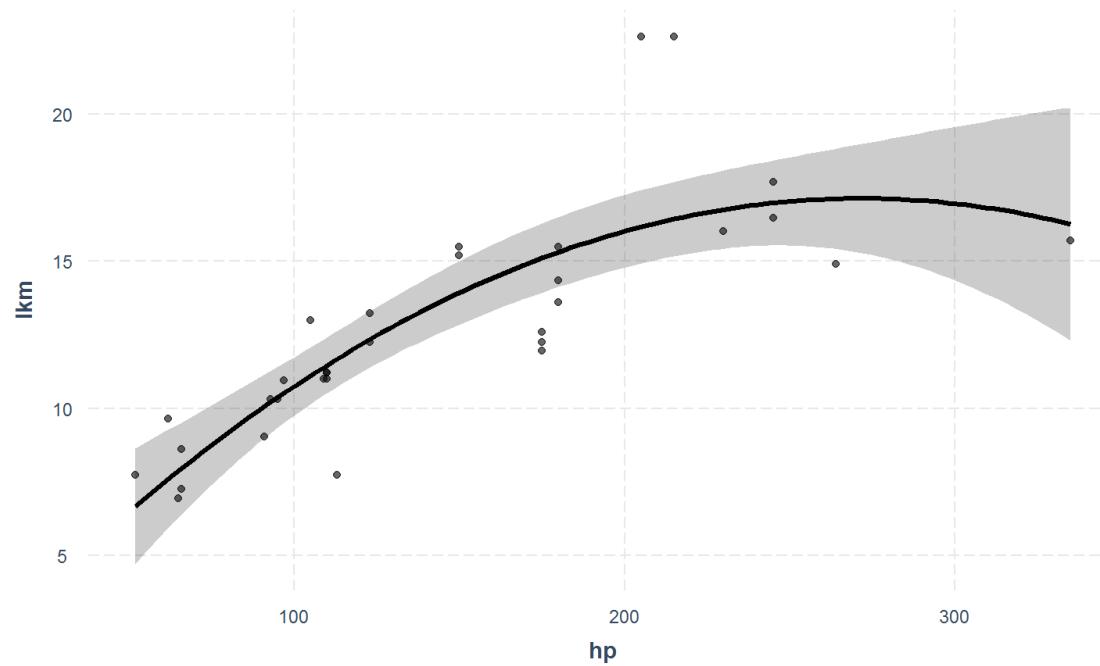
```
jtools::effect_plot(lm_obj, pred = 'hp', plot.points = T, data = df_cars)
```



# Visualisierung der Effekte

- Visualisierung der Vorhersagen für verschiedene Prädiktorwerte
- für die einfache lineare Regression:

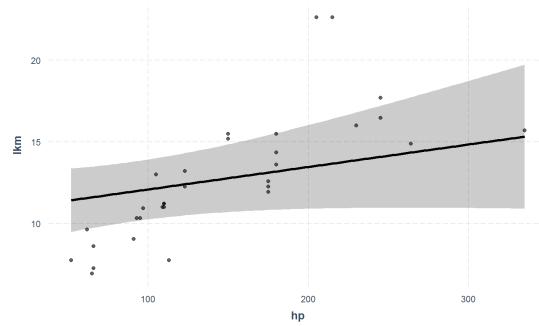
```
jtools::effect_plot(m3, pred = 'hp',  
  plot.points = T, data = df_cars, interval = T)
```



# Visualisierung der Effekte

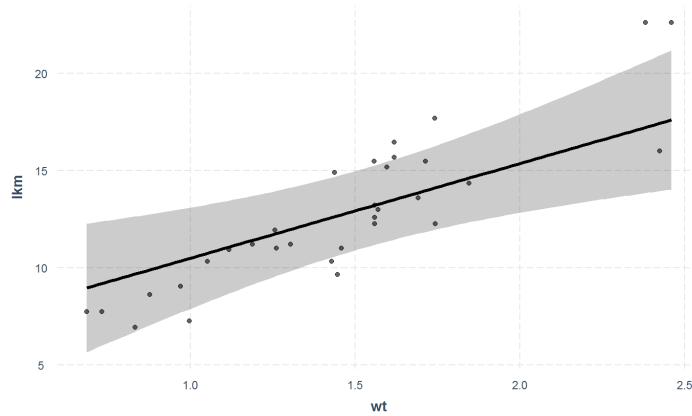
- für komplexere Modelle
- Prädiktor `hp`

```
jtools::effect_plot(lm_obj2, pred = 'hp',  
                    plot.points = T, interval = T)
```



- Prädiktor `wt`

```
jtools::effect_plot(lm_obj2, pred = 'wt',  
                    plot.points = T, interval = T)
```



- Effekte des Prädiktors auf die vorhergesagten Werte
- die Einflüsse der weiteren Variablen werden dabei *konstant gehalten*

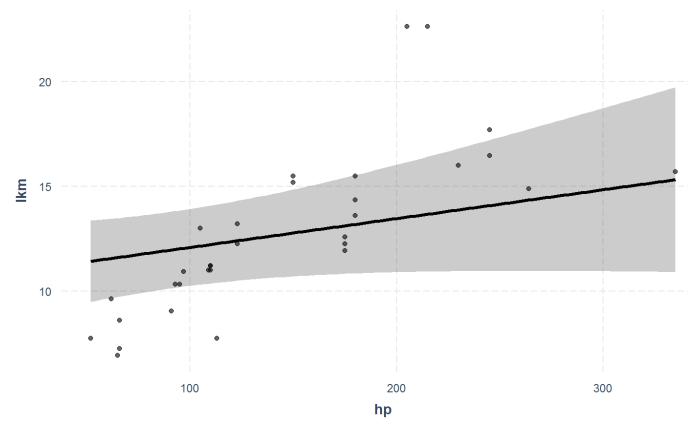
# Visualisierung der Effekte

- *konstant halten*
  - Werte der anderen Prädiktoren werden nicht variiert
  - Wirkung des Prädiktors wenn *um die Einflüsse der anderen Variablen kontrolliert wird*
  - ein fester Wert für die anderen Prädiktoren wird verwendet
- feste Werte:
  - kontinuierliche Prädiktoren: Mittelwert
  - Faktoren: Referenzkategorie
  - logische Variablen: auf FALSE

# Visualisierung der Effekte

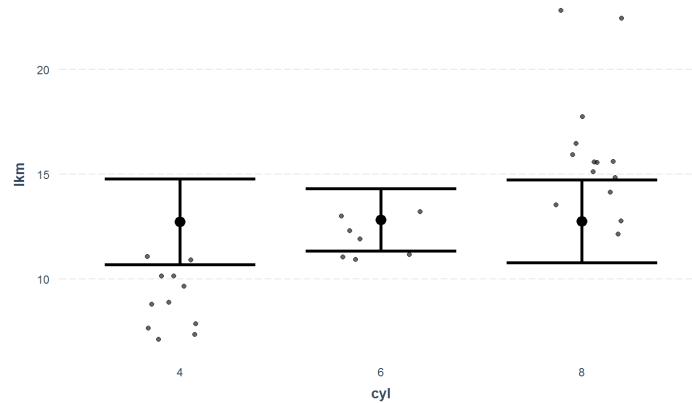
- für komplexere Modelle:
- hp

```
jtools::effect_plot(  
  lm_obj2, pred = 'hp',  
  plot.points = T, interval = T)
```



- auch kategoriale Prädiktoren werden korrekt interpretiert hp

```
jtools::effect_plot(  
  lm_obj2, pred = 'cyl',  
  plot.points = T, jitter = .2)
```



- das sieht nicht gut aus!
- ist das Modell wirklich so schlecht?

# Visualisierung der Effekte

## partielle Residuen

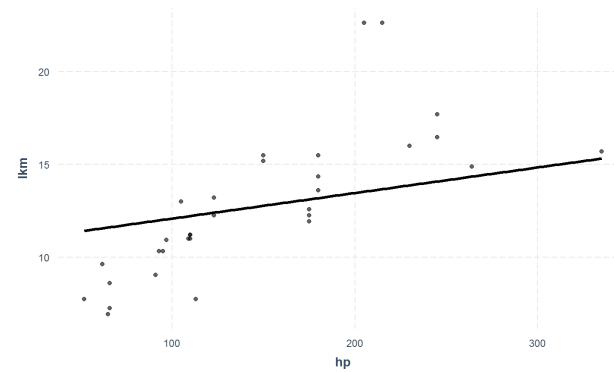
- 😊 Nein, es ist besser.
- in obigen Plots sind die Effekte der weiteren Prädiktoren nicht berücksichtigt
- bei der einfachen Regression war das noch kein Problem
- Lösung: *partielle Residuenplots*
  - die Residuen die “übrig bleiben” nach Kontrolle der Einflüsse aller anderer Prädiktoren
  - $\hat{\epsilon}_{x_j} = y - \beta_0 - \beta_1 \times X_1 - \beta_2 \times X_2 \dots - \beta_{j-1} \times X_{j-1}$

# Visualisierung der Effekte

## partielle Residuen

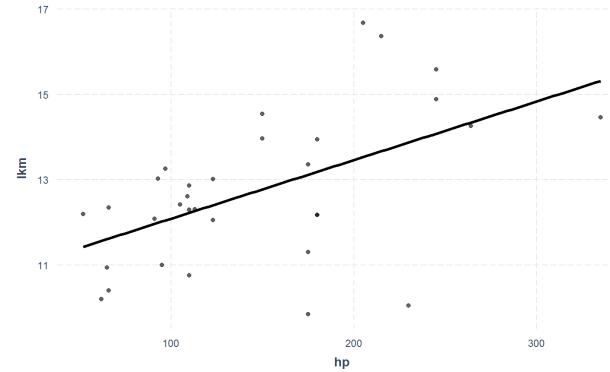
- einfache Residuen

```
jtools::effect_plot(lm_obj2,
                     pred = 'hp',
                     data = df_cars,
                     plot.points = T,
                     partial.residuals = F)
```



- partielle Residuen:

```
jtools::effect_plot(lm_obj2,
                     pred = 'hp',
                     data = df_cars,
                     plot.points = T,
                     partial.residuals = T)
```

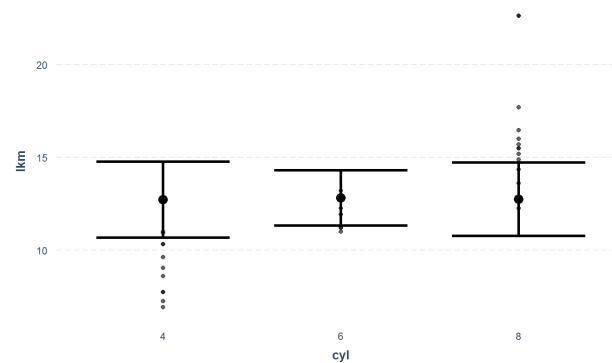


# Visualisierung der Effekte

## partielle Residuen

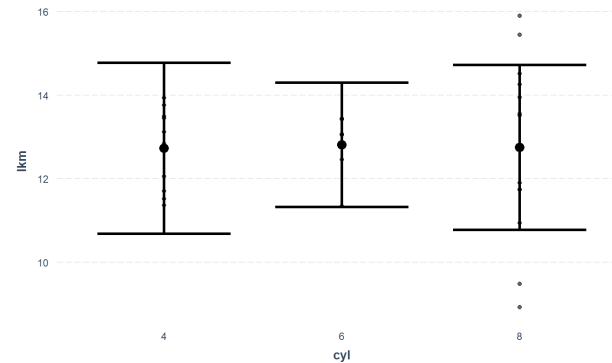
- einfache Residuen

```
jtools::effect_plot(lm_obj2,  
                    pred = 'cyl',  
                    data = df_cars,  
                    plot.points = T,  
                    partial.residuals = F)
```



- partielle Residuen:

```
jtools::effect_plot(lm_obj2,  
                    pred = 'cyl',  
                    data = df_cars,  
                    plot.points = T,  
                    partial.residuals = T)
```



# Interaktionen

- bisher...
  - Effekte einer Variable unabhängig von den Werten anderer Variablen
  - ggf. bestehen aber Wechselwirkungen bzw. Unterschiede in der Wirkung je nach Werten eines anderen Prädiktors
- 2-Variablen-Modell:
  - $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$
  - Erhöhung von  $X_1$  um eine Einheit, dann erhöht sich  $Y$  um  $\beta_1$ -Einheiten
  - Dieser Effekt ist unabhängig vom Wert  $X_2$
- Erweiterung um Interaktion:
  - $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon$
  - rein additive Wirkung wird aufgehoben

# Interaktionen

- Umstellen:
  - $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon$
  - $Y = \beta_0 + (\beta_1 + \beta_3 X_2) X_1 + \beta_2 X_2 + \epsilon$
  - $Y = \beta_0 + \tilde{\beta}_1 X_1 + \beta_2 X_2 + \epsilon$
  - mit  $\tilde{\beta}_1 = (\beta_1 + \beta_3 X_2)$
  - und  $\tilde{\beta}_1$  ändert sich mit dem Wert von  $X_2$
  - somit ändert sich auch der Effekt von  $X_1$  auf  $Y$
- Beispiel:
  - Produktionseinheiten ( $Y$ ) in Abh. von Prod.linien und Arbeitern
  - $Einheiten = 1.2 + 3.4 \times Linien + 0.22 \times Arbeiter + 1.4 \times (Linien \times Arbeiter)$
  - $Einheiten = 1.2 + (3.4 + 1.4 \times Arbeiter) \times Linien + 0.22 \times Arbeiter$
  - 1 zusätzliche Linie erhöht Einheiten um  $(3.4 + 1.4 \times Arbeiter)$

# Interaktionen

- *hierarchisches Prinzip*
  - bei sig. Interaktion bleiben Haupteffekte unabhängig von Signifikanz im Modell!
- Interaktion zwischen
  - zwei kontinuierlichen Prädiktoren
  - zwei kategorialen Prädiktoren
  - kategorialem und kontinuierlichem Prädiktor

# Interaktionen

- Modell nur mit Haupteffekten:

```
i1 <- lm(lkm ~ hp + factor(am), data = df_cars)
```

- Visualisieren 1:

```
sjPlot::plot_model(i1, terms = 'hp', type = 'pred')
sjPlot::plot_model(i1, terms = 'am', type = 'pred')
```

- Visualisieren 2:

```
sjPlot::plot_model(i1, terms = c('hp', 'am'), type = 'pred')
sjPlot::plot_model(i1, terms = c('am', 'hp'), type = 'pred')
```

# Interaktionen

- Modell mit Interaktion:

```
i2 <- lm(lkm ~ hp * factor(am), data = df_cars)
```

- Visualisieren 1:

```
sjPlot::plot_model(i1, terms = 'hp', type = 'pred')
sjPlot::plot_model(i2, terms = 'am', type = 'pred')
```

- Visualisieren 2:

```
sjPlot::plot_model(i2, terms = c('hp', 'am'), type = 'pred')
sjPlot::plot_model(i2, terms = c('am', 'hp'), type = 'pred')
```

# Fragen 1

Angenommen, wir haben einen Datensatz mit fünf Prädiktoren:  $X_1$  = Notenpunkte,  $X_2$  = IQ,  $X_3$  = Geschlecht (1 für weiblich und 0 für männlich),  $X_4$  = Interaktion zwischen Notenpunkten und IQ, und  $X_5$  = Interaktion zwischen Notenpunkten und Geschlecht. Die Antwort ist das Anfangsgehalt nach dem Abschluss (in Tausend Euro). Nehmen wir an, wir verwenden die Methode der kleinsten Quadrate, um das Modell zu finden, und erhalten  $b_0 = 50$ ,  $b_1 = 20$ ,  $b_2 = 0,07$ ,  $b_3 = 35$ ,  $b_4 = 0,01$ ,  $b_5 = -10$ .

- Welche Antwort ist richtig und warum?
  1. Bei einem fixierten Wert von IQ und Notenpunkten verdienen Männer im Durchschnitt mehr als Frauen.
  2. Bei einem fixen Wert von IQ und Notenpunkten verdienen Männer im Durchschnitt mehr als Frauen, vorausgesetzt, die Notenpunkte sind hoch genug.
  3. Bei einem fixen Wert von IQ und Notenpunkten verdienen Frauen im Durchschnitt mehr als Männer, vorausgesetzt, die Notenpunkte sind hoch genug.
- Sagen Sie das Gehalt einer Frau mit einem IQ von 110 und einem Notenpunktewert von 4,0 voraus.
- Richtig oder falsch: Da der Koeffizient für den Interaktionsterm Notenpunkte/IQ sehr klein ist, gibt es nur sehr wenige Hinweise auf einen Interaktionseffekt. Begründen Sie Ihre Antwort.

# Übung 1

## Kraftstoffverbrauch

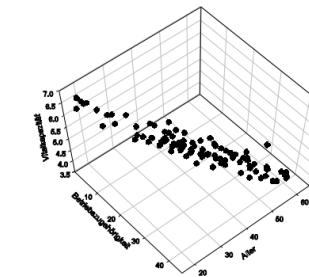
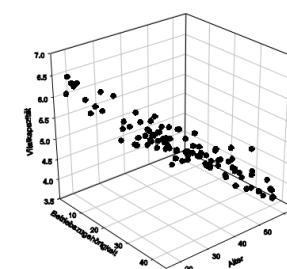
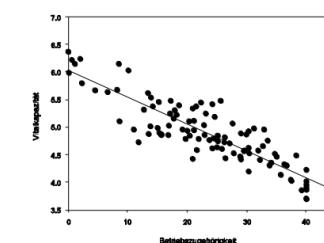
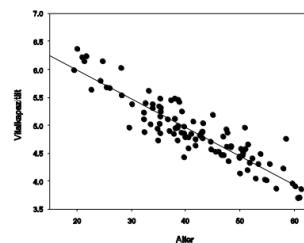
- Ein kleiner Wettbewerb: Verwenden sie den oben besprochenen *df\_cars*-Datensatz und versuchen Sie die Vorhersage des Kraftstoffverbrauchs weiter zu verbessern. Prüfen Sie dabei auch ob die Verwendung von Plynomen oder Interaktionen als Prädiktoren geeignet ist. Bedenken Sie auch die Überanpassung des Modells an die Daten bei der Modellbewertung.

# Übung 2

## Vitalkapazität

- Bei 100 Mitarbeitern eines Chemieunternehmens wurde untersucht, ob ein Zusammenhang zwischen deren Lungen-Vitalkapazität (VK, in Liter) und der Betriebszugehörigkeit (in Jahren) besteht. In der folgenden Tabelle sind die Resultate der einfachen und multiplen Regressionsanalysen unter Adjustierung für das Lebensalter (in Jahren) aufgeführt.

Einflussgröße/Parameter	einfache Analyse		multiple Analyse
	b (p-Wert)	b (p-Wert)	b (p-Wert)
Betriebszugehörigkeit	-0.048 (<0.001)	---	-0.006 (0.452)
Lebensalter	---	-0.051 (<0.001)	-0.046 (<0.001)
Achsenabschnitt	6.020 (<0.001)	7.023 (<0.001)	6.924 (<0.001)
Bestimmtheitsmaß	0.762	0.822	0.823



- Stellen Sie für die beiden einfachen Regressionsanalysen und für die multiple Regressionsanalyse die resultierenden Modellgleichungen auf. Wie sind die Resultate dieser linearen Regressionsanalysen zu interpretieren?

# Übung 3

## Simulation

In dieser Übung werden Sie einige simulierte Daten erstellen und einfache lineare Regressionsmodelle darauf anwenden. Stellen Sie sicher, dass Sie `set.seed(1)` verwenden um konsistente Ergebnisse sicherzustellen.

1. Erstellen Sie mit der Funktion `rnorm()` einen Vektor  $x$ , der 100 Beobachtungen enthält, die aus einer  $N(0, 1)$ -Verteilung gezogen wurden. Dies stellt ein Merkmal  $X$  dar.
2. Erstellen Sie mit der Funktion `rnorm()` einen Vektor  $\text{eps}$ , der 100 Beobachtungen enthält, die aus einer  $N(0, 0.25)$ -Verteilung gezogen wurden, d. h. einer Normalverteilung mit Mittelwert Null und Varianz 0.25.
3. Erzeugen Sie mit Hilfe von  $x$  und  $\text{eps}$  einen Vektor  $y$  nach dem Modell  $Y = -1 + 0.5 \times X + \epsilon$ . Wie groß ist die Länge des Vektors  $y$ ? Welches sind die Werte von  $b_0$  und  $b_1$  in diesem linearen Modell?
4. Erstellen Sie ein Streudiagramm, das die Beziehung zwischen  $x$  und  $y$  darstellt. Kommentieren Sie Ihre Beobachtungen.
5. Passen Sie ein lineares Regressionsmodell an, um  $y$  anhand von  $x$  vorherzusagen. Kommentieren Sie das erhaltene Modell. Wie verhalten sich  $b_0$  und  $b_1$  im Vergleich zu  $\beta_0$  und  $\beta_1$ ?

# Übung 3

## Simulation (Forts.)

6. Stellen Sie die Linie der kleinsten Quadrate in dem in 4. erstellten Streudiagramm dar. Zeichnen Sie die Regressionslinie der Population in einer anderen Farbe in das Diagramm ein.
7. Erstellen Sie nun ein polynomiales Regressionsmodell, das  $y$  anhand von  $x$  und  $x^2$  vorhersagt. Gibt es Hinweise darauf, dass der quadratische Term das Modell verbessert? Erläutern Sie Ihre Antwort.
8. Wiederholen Sie 1-6, nachdem Sie den Prozess der Datenerzeugung so verändert haben, dass die Daten *weniger* Rauschen enthalten. Das Modell sollte das gleiche bleiben. Sie können dies erreichen, indem Sie die Varianz der Normalverteilung *verringern*, die zur Erzeugung des Fehlerterms  $\epsilon$  in 2 verwendet wurde. Beschreiben Sie Ihre Ergebnisse.
9. Wiederholen Sie 1-6, nachdem Sie den Prozess der Datenerzeugung so verändert haben, dass die Daten *mehr* Rauschen enthalten. Das Modell sollte das gleiche bleiben. Sie können dies erreichen, indem Sie die Varianz der Normalverteilung *erhöhen*, die zur Erzeugung des Fehlerterms  $\epsilon$  in 2 verwendet wurde. Beschreiben Sie Ihre Ergebnisse.
10. Wie groß sind die Konfidenzintervalle für  $\beta_0$  und  $\beta_1$  auf der Grundlage des ursprünglichen Datensatzes, des verrauschten Datensatzes und des weniger verrauschten Datensatzes? Kommentieren Sie Ihre Ergebnisse.

# Übung 4

## Werbeausgaben

- Laden Sie den Datensatz zum Werbeausgaben und Verkaufszahlen:

```
df_ads <- rio:::import("https://raw.githubusercontent.com/Statistican/Datasets/main/Werbung.csv")
```

- Beantworten Sie folgende Fragen:
  - Besteht ein Zusammenhang zwischen dem Werbeetat und den Verkaufszahlen?
  - Wie stark ist der Zusammenhang?
  - Welche Medien tragen zum Verkauf bei?
  - Wie groß ist die Werbewirkung einzelner Medien?
  - Wie präzise können zukünftige Verkaufszahlen vorhergesagt werden?
  - Gibt es Synergieeffekte bei Werbung in verschiedenen Medien?

# Übung 5

## Kindersitze

- Laden Sie folgenden Datensatz:

```
df_seat <- rio::import("https://github.com/Statistican/Datasets/raw/main/Kindersitze.RData", trust = T)
```

- Schauen Sie sich den Datensatz an. Was fällt Ihnen auf?
- Passen Sie ein multiples Regressionsmodell an, um die Verkäufe mit Hilfe von Preis, Urban und USA vorherzusagen (Modell 1).
- Interpretieren Sie die einzelnen Koeffizienten des Modells. Achtung, einige der Variablen im Modell sind qualitativ!
- Schreiben Sie das Modell in Form von Gleichungen auf, wobei Sie darauf achten, die qualitativen Variablen richtig zu behandeln.
- Für welche der Prädiktoren können Sie die Nullhypothese ablehnen?
- Auf der Grundlage der vorherigen Frage: Erstellen Sie ein kleineres Modell, das nur die Prädiktoren verwendet, für die ein Zusammenhang mit den Verkaufszahlen nachgewiesen ist (Modell 2).
- Wie gut sind die Vorhersagen der Modelle 1 und 2 im Vgl zu den beobachteten Daten?
- Ermitteln Sie unter Verwendung des Modells 2 95%-ige Konfidenzintervalle für den/die Koeffizienten.
- Gibt es Anzeichen für Ausreißer oder Beobachtungen mit hoher Hebelwirkung in Modell 2?

# Übung 6

## Boston-Housing Daten

- Dieses Problem bezieht sich auf den Boston-Housing Datensatz (`df_bost <- MASS::Boston`). Ziel ist die Pro-Kopf-Verbrechensrate anhand der anderen Variablen in diesem Datensatz vorherzusagen.
1. Erstellen Sie für jeden Prädiktor ein einfaches lineares Regressionsmodell zur Vorhersage der Antwort. Beschreiben Sie Ihre Ergebnisse. In welchem der Modelle besteht ein statistisch signifikanter Zusammenhang zwischen dem Prädiktor und der Antwort? Erstellen Sie einige Diagramme, um Ihre Behauptungen zu untermauern.
  2. Passen Sie ein multiples Regressionsmodell an, um die Antwort unter Verwendung aller Prädiktoren vorherzusagen. Beschreiben Sie Ihre Ergebnisse. Für welche Prädiktoren kann die Nullhypothese  $H_0 : \beta_j = 0$  zurückgewiesen werden?
  3. Vergleichen Sie die Ergebnisse aus 1. mit denen aus 2.! Erstellen Sie dazu ein Diagramm, dass die univariaten Regressionskoeffizienten aus 1 auf der x-Achse und die multiplen Regressionskoeffizienten aus 2 auf der y-Achse darstellt. Das heißt, jeder Prädiktor wird als ein einzelner Punkt in der Grafik dargestellt. Sein Koeffizient in einem einfachen linearen Regressionsmodell wird auf der x-Achse und sein Koeffizientenschätzer im multiplen linearen Regressionsmodell auf der y-Achse dargestellt.
  4. Gibt es Hinweise auf einen nicht-linearen Zusammenhang zwischen einem der Prädiktoren und dem Regressand? Um diese Frage zu beantworten, erstellen Sie für jeden einzelnen Prädiktor X ein Modell der Form  $Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \epsilon$ .