Über die Koeffizienten einer linearen Regression

Norman Markgraf

2021-06-09

Bei einer *einfachen Regression* versuchen wir zu gegebenen Datenpunkten $(x_1, y_1), ...(x_n, y_n)$ eine *möglichst passende* Funktion $g(x)$ zu finden, so dass

$$y_i = g(x_i) + e_i$$

gilt. Dabei tollerieren wir eine (kleine) Abweichung e_i.

Bei einer *einfachen* ***linearen*** *Regression* gehen wir davon aus, dass die Datenpunkte (im wesendlichen) auf einer Geraden

$$y_i = \beta_0 + \beta_1 \cdot x_i + e_i$$

liegen.

Unsere Aufgabe besteht nun darin die Parameter β_0 (y-Achenabschnitt) und β_1 (Steigung) an hand der Datenpunkte zu schätzen. Schätzungen tragen ein Dacht ($\hat{.}$) um sie von den (in der Regel unbekannten) Parametern zu unterscheiden.

Wir suchen somit nach $\hat{\beta} = \left(\hat{\beta}_0, \hat{\beta}_1\right)$, so dass die Gerade $\hat{\beta}_0 + \hat{\beta}_1 \cdot x$ zu gegebenem x_i eine möglichst gute Schätzung von y_i (genannt $\hat{y}_i$) bestimmt:

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 \cdot x_i$$

Die Abweichung $\hat{e}_i$ unserer Schätzung $\hat{y}_i$ von dem gegebenen Wert y_i ist somit:

$$\hat{e}_i = \hat{y}_i - y_i = \hat{\beta}_0 + \hat{\beta}_1 \cdot x_i - y_i$$

Wenn wir diese Abweichung über alle i minimieren, finden wir unser $\hat{\beta}$.

Doch das wirft eine Frage auf: *Wie genau messen wir* ***kleinsten Fehler*** *der* $\hat{e}_i$*?*

Wir betrachten zunächste drei einfache Ideen:

1. Idee: Summe der Abweichungen
2. Idee: Summe der absoluten Abweichungen

3. Idee: Summe der quadratischen Abweichungen

Wir betrachten die drei Ideen in umgekehrter Reihenfolge:

3. Idee: Summe der quadratischen Abweichungen

Wir bezeichnen mit

$$QS = QS(\hat{\beta}) = QS(\hat{\beta}_0, \hat{\beta}_1) = \sum_{i=1}^{n} \hat{e}_i^{\,2} = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 = \sum_{i=1}^{n} \left(\hat{\beta}_0 + \hat{\beta}_1 \cdot x_i - y_i\right)^2$$

die **Q**uadrat-**S**umme der Abweichungen.

Gesucht wird $\hat{\beta} = \left(\hat{\beta}_0,\, \hat{\beta}_1\right)$, so das QS minimiert wird.

Dieses klassische Minimierungsproblem hat eine (exakte) mathematisch-algebraisch Lösung in dem wir die Nullstelle der ersten partiellen Ableitung von QS nach $\hat{\beta}_0$ bzw. $\hat{\beta}_1$ bestimmen. Neben dieser notwendigen Bedindung, müssen wir formal noch die dann hinreichende Bedinung auf ein Minimum prüfen.

Aber wir beginnen mit den notwenigdigen Bedingungen:

Schätzen des y-Achenabschnitts $\hat{\beta}_0$

Es ist:

$$\begin{aligned}
\frac{\partial}{\partial \hat{\beta}_0} QS &= 2 \cdot \sum_{i=1}^{n} \left(\hat{\beta}_0 + \hat{\beta}_1 \cdot x_i - y_i\right) \cdot 1 \\
&= 2 \cdot \left(\sum_{i=1}^{n} \hat{\beta}_0 + \sum_{i=1}^{n} \hat{\beta}_1 \cdot x_i - \sum_{i=1}^{n} y_i\right) \\
&= 2 \cdot \left(n \cdot \hat{\beta}_0 + \hat{\beta}_1 \cdot \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} y_i\right)
\end{aligned}$$

Für die notwendige Bedingung eines Minimums, setzen wir den Ausdruck gleich Null und erhalten:

$$\begin{aligned}
0 &= \frac{\partial}{\partial \hat{\beta}_0} QS \\
&= 2 \cdot \left(n \cdot \hat{\beta}_0 + \hat{\beta}_1 \cdot \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} y_i\right) \\
&= n \cdot \hat{\beta}_0 + \hat{\beta}_1 \cdot \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} y_i
\end{aligned}$$

Wir stellen nach $\hat{\beta}_0$ um:

$$
\begin{aligned}
-n \cdot \hat{\beta}_0 &= \hat{\beta}_1 \cdot \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} y_i \\
\hat{\beta}_0 &= \frac{1}{n} \cdot \sum_{i=1}^{n} y_i - \hat{\beta}_1 \cdot \frac{1}{n} \cdot \sum_{i=1}^{n} x_i \\
&= \bar{y} - \hat{\beta}_1 \cdot \bar{x}
\end{aligned}
$$

Wenn wir $\hat{\beta}_1$ bestimmen, so haben wir auch $\hat{\beta}_0$.

Schätzen der Steigung $\hat{\beta}_1$

Es ist:

$$
\begin{aligned}
\frac{\partial}{\partial \hat{\beta}_1} QS &= 2 \cdot \sum_{i=1}^{n} \left(\hat{\beta}_0 + \hat{\beta}_1 \cdot x_i - y_i \right) \cdot x_i \\
&= 2 \cdot \left(\sum_{i=1}^{n} \hat{\beta}_0 \cdot x_i + \sum_{i=1}^{n} \hat{\beta}_1 \cdot x_i \cdot x_i - \sum_{i=1}^{n} y_i \cdot x_i \right) \\
&= 2 \cdot \left(\hat{\beta}_0 \cdot \sum_{i=1}^{n} x_i + \hat{\beta}_1 \cdot \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} y_i \cdot x_i \right)
\end{aligned}
$$

Wir ersetzen nun $\hat{\beta}_0$ durch $\bar{y} - \hat{\beta}_1 \cdot \bar{x}$ und erhalten:

$$
\begin{aligned}
\frac{\partial}{\partial \hat{\beta}_1} QS &= 2 \cdot \left(\left(\bar{y} - \hat{\beta}_1 \cdot \bar{x} \right) \cdot \sum_{i=1}^{n} x_i + \hat{\beta}_1 \cdot \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} y_i \cdot x_i \right) \\
&= 2 \cdot \left(\bar{y} \cdot \sum_{i=1}^{n} x_i - \hat{\beta}_1 \cdot \bar{x} \cdot \sum_{i=1}^{n} x_i + \hat{\beta}_1 \cdot \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} y_i \cdot x_i \right)
\end{aligned}
$$

Wir setzen nun wieder den Ausdruck gleich Null:

$$
\begin{aligned}
0 &= 2 \cdot \left(\bar{y} \cdot \sum_{i=1}^{n} x_i - \hat{\beta}_1 \cdot \bar{x} \cdot \sum_{i=1}^{n} x_i + \hat{\beta}_1 \cdot \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} y_i \cdot x_i \right) \\
&= \bar{y} \cdot \sum_{i=1}^{n} x_i - \hat{\beta}_1 \cdot \bar{x} \cdot \sum_{i=1}^{n} x_i + \hat{\beta}_1 \cdot \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} y_i \cdot x_i \\
&= \frac{1}{n} \cdot \bar{y} \cdot \sum_{i=1}^{n} x_i - \frac{1}{n} \cdot \hat{\beta}_1 \cdot \bar{x} \cdot \sum_{i=1}^{n} x_i + \frac{1}{n} \cdot \hat{\beta}_1 \cdot \sum_{i=1}^{n} x_i^2 - \frac{1}{n} \cdot \sum_{i=1}^{n} y_i \cdot x_i \\
&= \bar{y} \cdot \frac{1}{n} \cdot \sum_{i=1}^{n} x_i - \hat{\beta}_1 \cdot \bar{x} \cdot \frac{1}{n} \cdot \sum_{i=1}^{n} x_i + \frac{1}{n} \cdot \hat{\beta}_1 \cdot \sum_{i=1}^{n} x_i^2 - \frac{1}{n} \cdot \sum_{i=1}^{n} y_i \cdot x_i \\
&= \bar{y} \cdot \bar{x} - \hat{\beta}_1 \cdot \bar{x} \cdot \bar{x} + \hat{\beta}_1 \cdot \frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2 - \frac{1}{n} \cdot \sum_{i=1}^{n} y_i \cdot x_i \\
&= \bar{y} \cdot \bar{x} - \hat{\beta}_1 \cdot \bar{x}^2 + \hat{\beta}_1 \cdot \frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2 - \frac{1}{n} \cdot \sum_{i=1}^{n} y_i \cdot x_i
\end{aligned}
$$

Wir stellen nach $\hat{\beta}_1$ um:

$$\begin{aligned}
\hat{\beta}_1 \cdot \bar{x}^2 - \hat{\beta}_1 \cdot \frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2 &= \bar{y} \cdot \bar{x} - \frac{1}{n} \cdot \sum_{i=1}^{n} y_i \cdot x_i \\
\hat{\beta}_1 \cdot \left(\bar{x}^2 - \frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2 \right) &= \bar{y} \cdot \bar{x} - \frac{1}{n} \cdot \sum_{i=1}^{n} y_i \cdot x_i \\
\hat{\beta}_1 &= \frac{\bar{y} \cdot \bar{x} - \frac{1}{n} \cdot \sum\limits_{i=1}^{n} y_i \cdot x_i}{\bar{x}^2 - \frac{1}{n} \cdot \sum\limits_{i=1}^{n} x_i^2}
\end{aligned}$$

Mit Hilfe des *Verschiebesatzes von Steiner* erhalten wir:

$$\begin{aligned}
\hat{\beta}_1 &= \frac{\frac{1}{n} \cdot \sum\limits_{i=1}^{n} (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\frac{1}{n} \cdot \sum\limits_{i=1}^{n} (x_i - \bar{x})^2} \\
&= \frac{\sigma_{x,y}}{\sigma_x^2}
\end{aligned}$$

Beachte: *Da* $\frac{1}{n}$ *sowohl im Zähler als auch im Nenner steht können wir ihr kürzen und mit* $\frac{1}{n-1}$ *erweitern.*

Somit gilt ebenfalls:

$$\begin{aligned}
\hat{\beta}_1 &= \frac{\frac{1}{n} \cdot \sum\limits_{i=1}^{n} (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\frac{1}{n} \cdot \sum\limits_{i=1}^{n} (x_i - \bar{x})^2} = \frac{\frac{1}{n-1} \cdot \sum\limits_{i=1}^{n} (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\frac{1}{n-1} \cdot \sum\limits_{i=1}^{n} (x_i - \bar{x})^2} \\
&= \frac{s_{x,y}}{s_x^2}
\end{aligned}$$

Wir können zur Berechnung sowohl die Kovarianz der Grundgesamtheit $\sigma_{x,y}$ und die Varianz σ_x^2 von x, als auch deren Schätzer $s_{x,y}$ und s_x^2 verwendet werden!

Diese Methode nennt sich **Methode der kleinsten Quadrate** (engl. *ordenary least square method*) und wir sprechen dann auch von den **Kleinste-Quadrate-Schätzern** (oder kurz **KQ-Schätzer** bzw. **OLS-Schätzer**) $\hat{\beta}_0$ und $\hat{\beta}_1$.

Erweitern wir den Ausdruck mit Standardabweichung σ_y bzw. s_y, so erhalten wir:

$$
\begin{aligned}
\hat{\beta}_1 &= \frac{\sigma_{x,y}}{\sigma_x^2} \cdot \frac{\sigma_y}{\sigma_y} \\
&= \frac{\sigma_{x,y}}{\sigma_x \cdot \sigma_x} \cdot \frac{\sigma_y}{\sigma_y} \\
&= \frac{\sigma_{x,y}}{\sigma_x \cdot \sigma_y} \cdot \frac{\sigma_y}{\sigma_x} \\
&= \rho_{x,y} \cdot \frac{\sigma_y}{\sigma_x}
\end{aligned}
$$

und analog für die Schätzer:

$$
\begin{aligned}
\hat{\beta}_1 &= \frac{s_{x,y}}{s_x^2} \cdot \frac{s_y}{s_y} \\
&= \frac{s_{x,y}}{s_x \cdot s_x} \cdot \frac{s_y}{s_y} \\
&= \frac{s_{x,y}}{s_x \cdot s_y} \cdot \frac{s_y}{s_x} \\
&= r_{x,y} \cdot \frac{s_y}{s_x}
\end{aligned}
$$

Die Steigung $\hat{\beta}_1$ hat somit eine direkte Beziehung mit dem *Korrelationskoeffizenten* ρ (der Grundgesamtheit) bzw. r (der Stichprobe).

Für eine Berechnung in **R** heißt dies: wir können die Regressionskoeffizienten $\hat{\beta}_0$ und $\hat{\beta}_1$ direkt algebraisch ausrechnen, wenn wir

a) die Standardabweichungen von x und y und den Korrelationskoeffizienten oder

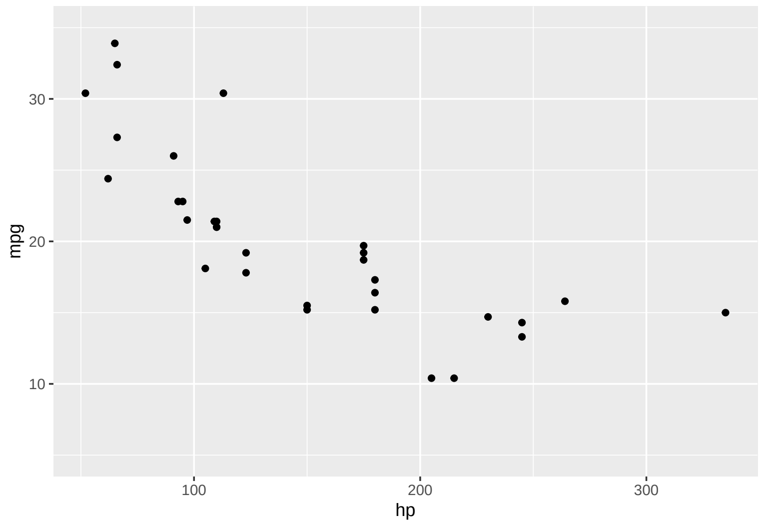
b) die Varianz von x und Kovarianz von x und y

haben.

Ein Beispiel in R:

Auf Grundlage der Datentabelle *mtcars* wollen wir Prüfen wie ein linearer Zusammenhang zwischen dem Verbrauch (in Meilen pro Gallone *mpg*) und der Leistung (Pferdestärke *hp*) modelliert werden kann.

```
library(mosaic)
# Wir nehmen eie Datentabelle 'mtcars':
dt <- mtcars
# und vergleichen Verbrauch (mpg, miles per gallon) mit der Pferdestärke (hp)
# Mit Hilfe eines Streudiagramms
gf_point(mpg ~ hp, data = dt) %>%
  gf_lims(y = c(5,35))
```



Berechnen wir zunächst die Mittelwerte von x (also 'hp') und y (also 'mpg')

```
mean_hp = mean(~ hp, data = dt)
mean_mpg = mean(~ mpg, data = dt)
mean_hp
```

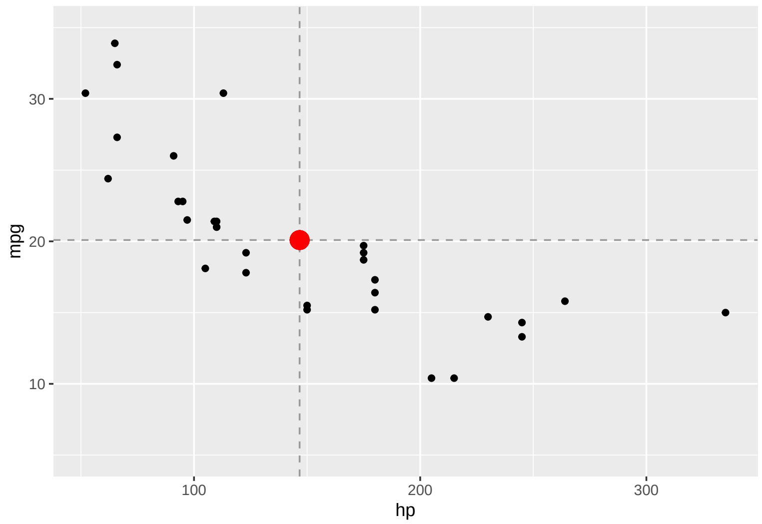
```
## [1] 146.6875
```

```
mean_mpg
```

```
## [1] 20.09062
```

und zeichnen die Punkt $(\bar{x}, \bar{y}) = (146.69, 20.09)$ in unser Streudiagramm ein:

```
gf_point(mpg ~ hp, data = dt) %>%
  gf_hline(yintercept = ~ mean_mpg, color = "grey60", linetype = "dashed") %>%
  gf_vline(xintercept = ~ mean_hp, color = "grey60", linetype = "dashed") %>%
  gf_point(mean_mpg ~ mean_hp, color = "red", size = 5, alpha = 0.2) %>%
  gf_lims(y = c(5,35))
```

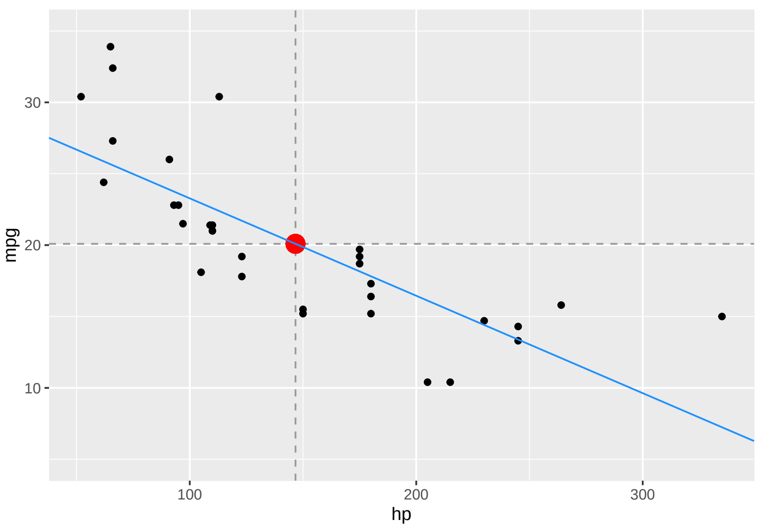


Berechnen wir nun die Schätzwerte für die Regressiongerade

```
beta_1 = cov(mpg ~ hp, data = dt) / var(~ hp, data = dt)
beta_0 = mean_mpg - beta_1 * mean_hp
```

und zeichnen diese ebenfalls in unser Streudiagramm ein:

```
gf_point(mpg ~ hp, data = dt) %>%
  gf_hline(yintercept = ~ mean_mpg, color = "grey60", linetype = "dashed") %>%
  gf_vline(xintercept = ~ mean_hp, color = "grey60", linetype = "dashed") %>%
  gf_point(mean_mpg ~ mean_hp, color = "red", size = 5, alpha = 0.2) %>%
  gf_abline(slope = ~ beta_1, intercept = ~beta_0, color = "dodgerblue") %>%
  gf_lims(y = c(5,35))
```



Die Funktionsvorschrift für die (blaue) Regressionsgerade lautet:

$$
\begin{aligned}
\hat{y} &= \hat{\beta}_0 + \hat{\beta}_1 \cdot x \\
&\approx 30.0988605 - 0.0682283 \cdot x \\
&\approx 30.099 - 0.068 \cdot x
\end{aligned}
$$

Studentisieren – einmal hin und einmal zurück

Was passiert eigentlich, wenn wir unsere x und y Werte studentisieren (aka standardisieren oder z-transformieren)?

Zur Erinnerung, studentisieren geht so:

$$x^{stud} = \frac{x - \bar{x}}{s_x}$$

In **R** können wir das mit der Funktion 'zscore' wie folgt:

```
dt$hp_stud <- zscore(dt$hp)
dt$mpg_stud<- zscore(dt$mpg)
```

Natürlich sind die Mittelwerte nun Null und die Standardabweichungen Eins:

```
favstats( ~ hp_stud, data = dt)
```

```
##        min         Q1     median        Q3      max         mean sd  n missing
##  -1.381032 -0.7319924 -0.3454858 0.4858679 2.746567 1.040834e-17  1 32       0
```

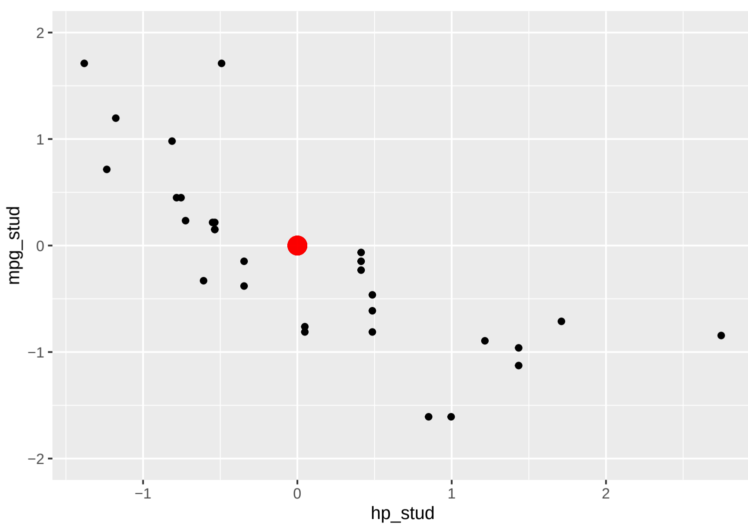
```
favstats( ~ mpg_stud, data = dt)
```

```
##        min         Q1     median        Q3      max         mean sd  n missing
##  -1.607883 -0.7741273 -0.1477738 0.4495434 2.291272 7.112366e-17  1 32       0
```

Der Grund für die kleinen Abweichungen von der Null beim MIttelwert sind Rundungsfehler, die der Computer macht!

Schauen wir uns nun das Streudiagramm an, zusammen mit dem Mittelpunkt $(0, 0)$

```
gf_point(mpg_stud ~ hp_stud, data = dt) %>%
  gf_point(0 ~ 0, color = "red", size = 5, alpha = 0.2) %>%
  gf_lims(y = c(-2, 2))
```

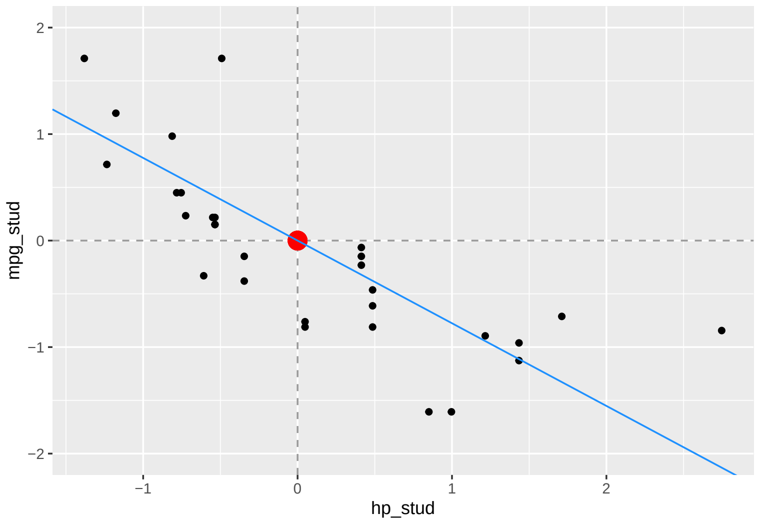


Auch wenn die Skalierungen sich geändert haben, die Diagramme sind sehr ähnlich.

Bestimmen wir die Koeffizienten der Regressionsgerade

```
beta_stud_1 = cov(mpg_stud ~ hp_stud, data = dt)
beta_stud_0 = 0 - beta_stud_1 * 0
```

und setzen sie in das Steudiagramm ein:



Wir können das studentisierte Problem auch wieder auf unser ursprüngliches zurück rechnen. Die Regressionsgerade im studentisierten Problem lautet:

$$
\begin{aligned}
\hat{y}^{stud} &= \hat{\beta}_0^{stud} + \hat{\beta}_1^{stud} \cdot x^{stud} \\
&\approx 30.0988605 - 0.7761684 \cdot x \\
&\approx 0 - 0.776 \cdot x^{stud}
\end{aligned}
$$

Rechnen wir nun mittels der Formel

$$\hat{\beta}_1 = \hat{\beta}_1^{stud} \cdot \frac{s_y}{s_x}$$

die Steigung um, so erhalten wir:

```
b1 <- beta_stud_1 * sd(dt$mpg) / sd(dt$hp)
b1
```

```
## [1] -0.06822828
```

Und setzen wir das in unsere Gleichung zur Bestimmung von $\hat{\beta}_0$ ein:

```
b0 <- mean(dt$mpg) - b1 * mean(dt$hp)
b0
```

```
## [1] 30.09886
```

so erhalten wir die Schätzwerte des ursprünglichen Problem.

Ein anderer Weg um die Regressionskoeffizenten zu bestimmen...

Gehen wir das Problem noch einmal neu an. Wir suchen $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1)$ welches $QS(\hat{\beta}) = QS(\hat{\beta}_0, \hat{\beta}_1) = \sum_{i=1}^{n} \left(\hat{\beta}_0 + \hat{\beta}_1 \cdot x_i - y_i\right)^2$ minimiert.

Statt es direkt, wie oben durch Nullsetzen der partiellen Ableitungen, zu bestimmen, wählen wir nun einen mathematisch-*numerischen* Ansatz und wollen $\hat{\beta} \in \mathbf{R}^2$ als *Optimierungsproblem* mit Hilfe des *Gradientenverfahrens* lösen.

Beim Gradientenverfahren wird versucht, ausgehend von einem Startwert $\hat{\beta}^0 \in \mathbf{R}^2$, gemäß der Iterationsvorschrift

$$\hat{\beta}^{k+1} = \hat{\beta}^k + \alpha^k \cdot d^k$$

für alle $k = 0, 1, ...$ eine Näherungslösung für $\hat{\beta}$ zu finden. Dabei ist $\alpha^k > 0$ eine *positive Schrittweite* und $d^k \in \mathbf{R}^n$ eine *Abstiegsrichtung*, welche wir in jedem Iterationsschritt k so bestimmen, dass die Folge $\hat{\beta}^k$ zu einem stationären Punkt, unserer Näherungslösung, konvergiert.

Im einfachsten Fall, dem **Verfahren des steilsten Abstieges**, wird der Abstiegsvektor d^k aus dem Gradienten ∇QS wie folgt bestimmt:

$$d^k = -\nabla QS\left(\hat{\beta}^k\right)$$

Wegen

$$\frac{\partial}{\partial \hat{\beta}_0} QS = 2 \cdot \left(n \cdot \hat{\beta}_0 + \hat{\beta}_1 \cdot \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} y_i \right)$$

und

$$\frac{\partial}{\partial \hat{\beta}_1} QS = 2 \cdot \left(\bar{y} \cdot \sum_{i=1}^{n} x_i - \hat{\beta}_1 \cdot \bar{x} \cdot \sum_{i=1}^{n} x_i + \hat{\beta}_1 \cdot \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} y_i \cdot x_i \right)$$

gilt:

$$\nabla QS(\hat{\beta}) = \nabla QS(\hat{\beta}_0, \hat{\beta}_1) = 2 \cdot \begin{pmatrix} n \cdot \hat{\beta}_0 + \hat{\beta}_1 \cdot \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} y_i \\ \bar{y} \cdot \sum_{i=1}^{n} x_i - \hat{\beta}_1 \cdot \bar{x} \cdot \sum_{i=1}^{n} x_i + \hat{\beta}_1 \cdot \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} y_i \cdot x_i \end{pmatrix}$$

Wir wollen hier von Anfang an mit den studentisierten Werten arbeiten, weil diese numerisch viele Vorteile haben. Darum vereinfachen sich die beiden partiellen Ableitungen noch einmal zu:

$$\frac{\partial}{\partial \hat{\beta}_0} QS = 2 \cdot \left(\hat{\beta}_1 \cdot \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} y_i \right)$$

und

$$\frac{\partial}{\partial \hat{\beta}_1} QS = 2 \cdot \left(\hat{\beta}_1 \cdot \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} y_i \cdot x_i \right)$$

Somit gilt:

$$\nabla QS(\hat{\beta}) = \nabla QS(\hat{\beta}_0, \hat{\beta}_1) = 2 \cdot \begin{pmatrix} \hat{\beta}_1 \cdot \sum\limits_{i=1}^{n} x_i - \sum\limits_{i=1}^{n} y_i \\ \hat{\beta}_1 \cdot \sum\limits_{i=1}^{n} x_i^2 - \sum\limits_{i=1}^{n} y_i \cdot x_i \end{pmatrix}$$

Weil im studentisierten Fall $\bar{x} = \bar{y} = 0$ gilt, können wir wegen $n \cdot \bar{x} = \sum\limits_{i=1}^{n} x_i$, $n \cdot \bar{y} = \sum\limits_{i=1}^{n} y_i$, $(n-1) \cdot s_x^2 = \sum\limits_{i=1}^{n} x_i^2$ und $(n-1) \cdot \mathrm{cov}_{x,y} = \sum\limits_{i=1}^{n} y_i \cdot x_i$ den Term ∇QS weiter vereinfachen:

$$\begin{aligned} \nabla QS(\hat{\beta}) = \nabla QS(\hat{\beta}_0, \hat{\beta}_1) &= 2 \cdot \begin{pmatrix} \hat{\beta}_1 \cdot n \cdot \bar{x} - n \cdot \bar{y} \\ \hat{\beta}_1 \cdot \sum\limits_{i=1}^{n} x_i^2 - \sum\limits_{i=1}^{n} y_i \cdot x_i \end{pmatrix} \\ &= 2 \cdot \begin{pmatrix} 0 \\ (n-1) \cdot \left(\hat{\beta}_1 \cdot s_x^2 - \mathrm{cov}_{x,y}\right) \end{pmatrix} \\ &= 2 \cdot (n-1) \cdot \begin{pmatrix} 0 \\ \hat{\beta}_1 \cdot s_x^2 - \mathrm{cov}_{x,y} \end{pmatrix} \end{aligned}$$

Um die Varianz und die Kovarianz nicht jedesmal neu zu berechnen, speichern wir die Ergebnisse vorab. Ebenso, damit der Quellcode kürzer wird, speichern wir in x und y die studentisierten Werte von *hp* und *mpg*:

```
# Vorbereitungen
sd_x <- var(dt$hp_stud)
cov_xy <- cov(mpg_stud ~ hp_stud, data = dt)

n <- length(dt$hp_stud)

x <- dt$hp_stud
y <- dt$mpg_stud
```

Nun erstellen wir die QS und ∇QS Funktionen: Wir definieren diese Funktion wie folgt in **R**:

```
qs <- function(b_0, b_1) {
  sum((b_1 * x - y)**2)
}

nabla_qs <- function(b_0, b_1) {
  c(0,
    2 * (n - 1) * (b_1 * sd_x - cov_xy)
  )
}
```

Die Schrittweite *alpha* bestimmen wir mit Hilfe der *Armijo-Bedingung* und der *Backtracking Liniensuche*: Diese formalisiert das Konzept “genügend” in der geforderten Verringerung des

Funktionswertes. Die Bedingung $f(x^k + \alpha d^k) < f(x^k)$ wird modifiziert zu

$$f(x^k + \alpha d^k) \leq f(x^k) + \sigma\alpha \left(\nabla f(x^k)\right)^T d^k,$$

mit $\sigma \in (0, 1)$. Die Armijo-Bedingung umgeht Konvergenzprobleme der einfachen Bedingung, indem sie fordert, dass die Verringerung zumindest proportional zur Schrittweite und zur Richtungsableitung $\left(\nabla f(x^k)\right)^T d^k$ ist, mit Hilfe der Proportionalitätskonstante σ. In der Praxis werden oft sehr kleine Werte verwendet, z.B. $\sigma = 0.0001$.

Die *Backtracking-Liniensuche* verringert die Schrittweite wiederholt um den Faktor ρ (`rho`) , bis die Armijo-Bedingung erfüllt ist. Sie terminiert garantiert nach einer endlichen Anzahl von Schritten. Wesshalb wir sie hier einsetzen:

```
alpha_k <- function(b_0, b_1, d_k, alpha = 1, sigma = 0.0001, rho = 0.5) {
  d_0 <- d_k[1]
  d_1 <- d_k[2]
  nabla <- nabla_qs(b_0, b_1)
  n_0 <- nabla[1]
  n_1 <- nabla[2]

  lhs <- qs(b_0 + alpha*d_0, b_1 + alpha*d_1)
  rhs <- qs(b_0, b_1) + sigma*alpha*(n_0*d_0 + n_1*d_1)

  while (lhs > rhs) {
    alpha <- rho * alpha
    lhs <- qs(b_0 + alpha*d_0, b_1 + alpha*d_1)
    rhs <- qs(b_0, b_1) + sigma*alpha*(n_0*d_0 + n_1*d_1)
  }
  return(alpha)
}
```

Ein paar Einstellungen vorab:

```
# maximale Anzahl an Iterationen
max_iter <- 1000
iter <- 0

# Genauigkeit
eps <- 10**-6

# Startwerte
b_0 <- 0
b_1 <- -1
```

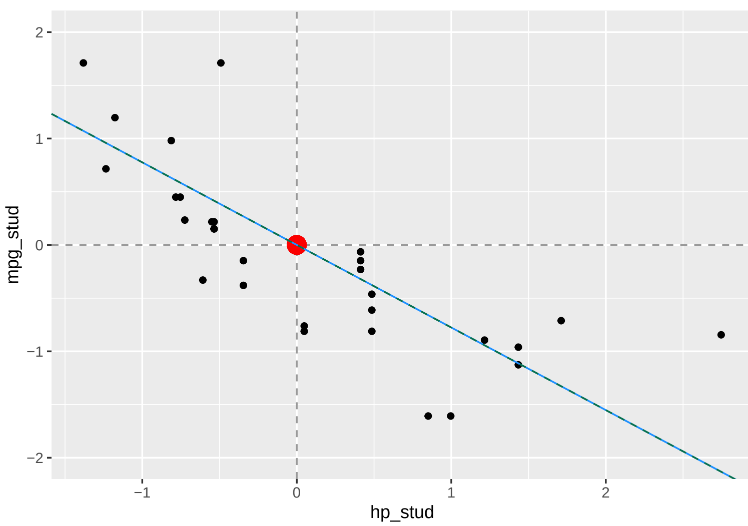
Für eine vorgegebene Genauigkeit ‘*eps*‘ $= 10^{-6}$, den Startwerten $\hat{\beta}_0^0 = 30.0988605$ und $\hat{\beta}_1^0 = -0.0682283$ können wir somit das Verfahren starten:

```
while (TRUE) {
  iter <- iter + 1

  d_k <- -nabla_qs(b_0, b_1)

  ad_ <- alpha_k(b_0, b_1, d_k) * d_k

  x0 <- b_0 + ad_[1]
  x1 <- b_1 + ad_[2]

  if ((abs(b_0 - x0) < eps) & (abs(b_1 - x1) < eps) | (iter > max_iter)) {
    break
  }
  b_0 <- x0
  b_1 <- x1
}
```

Wir haben somit mit 203 Iterationsschritten das folgende Ergebnisse für die Regressionskoeffizienten:

$$\hat{\beta}_0^{stud} = 0` \qquad \hat{\beta}_1 stud = -0.7761689$$

Betrachten wir die daraus erstellte Regressionsgerade:



Um die ursprünglichen Regressionskoeffizenten zu erhalten müssen wir zurück rechnen:

```
b1 <- b_1 * sd(dt$mpg) / sd(dt$hp)
b0 <- mean(dt$mpg) -  b1 * mean(dt$hp)
```

Die Geradengleichung für unser ursprüngliches Problem lautet somit:

$$
\begin{aligned}
\hat{y} &= \hat{\beta}_0 + \hat{\beta}_1 \cdot x \\
&\approx 30.0988668 - 0.0682283 \cdot x \\
&\approx 30.099 - 0.068 \cdot x
\end{aligned}
$$

Die R Funktion `optim`

In **R** gibt es bessere Optimierungsmethoden, als die hier verwendete. Zum Beispiel kännen wir die Funktion `optim` verwenden. Die Funktion `optim` benötigt die zu optimierende $f(x)$ und ggf. die Gradientenfunkt $gf(x)$ sowie einen Startpunkt x^0:

```r
f <- function(beta) {
  qs(beta[1], beta[2])
}

grf <- function(beta) {
  nabla_qs(beta[1], beta[2])
}

# Der eigentliche Aufruf von optim:
ergb <- optim(c(0,-1), f, grf)

# Auslesen der Schätzer aus dem Ergbnis:
optim_beta_0 <- ergb$par[1]
optim_beta_1 <- ergb$par[2]
```

Wir erhalten somit für das studentisierte Problem die Gerade:

$$
\begin{aligned}
\hat{y}^{stud} &= \hat{\beta}_0^{stud} + \hat{\beta}_1^{stud} \cdot x^{stud} \\
&\approx 0.1505859 - 0.7761719 \cdot x^{stud} \\
&\approx 0.151 - 0.776 \cdot x^{stud}
\end{aligned}
$$

Für das ursprüngliche Problem rechnen wir mittels

```r
optim_b1 <- optim_beta_1 * sd(dt$mpg) / sd(dt$hp)
optim_b0 <- mean(dt$mpg) -  optim_b1 * mean(dt$hp)
```

um und erhalten:

$$
\begin{aligned}
\hat{y} &= \hat{\beta}_0 + \hat{\beta}_1 \cdot x \\
&\approx 30.0989057 - 0.0682286 \cdot x \\
&\approx 30.099 - 0.068 \cdot x
\end{aligned}
$$

2. Idee: Summe der absoluten Abweichungen

Wir ändern nun die Abweichungsmessfunktion von der *Q*uadrat-*S*umme hin zu den **A**bsolut-*S*ummen:

$$AS = AS(\hat{\beta}) = AS(\hat{\beta}_0, \hat{\beta}_1) = \sum_{i=1}^{n} |\hat{y}_i - y_i|$$

Auch hier wollen wir mit den studentisierten Daten arbeiten und stellen die Funktion der *A*bsolut-*S*ummen auf:

```
# Absolute Abweichungssummen
as <- function(b_0, b_1) {
  return(sum(abs(b_0 + b_1 * x - y)))
}
```

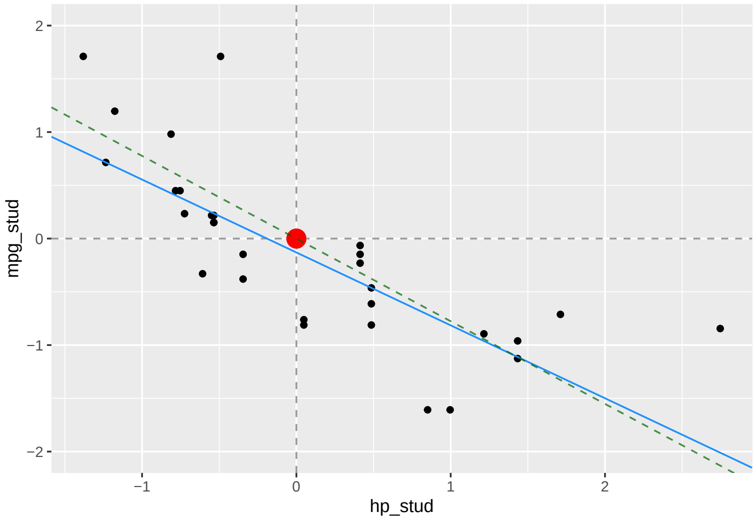
Danach konstuieren wir die zu optimierende Funktion f:

```
# Zu optimierende Funktion
f <- function(beta) {
  as(beta[1], beta[2])
}
```

Diesmal nutzen wir `optim` ohne eine Gradientenfunktion:

```
ergb <- optim(c(0,-1), f)

# Schätzer auslesen
opti_as_beta_0 <- ergb$par[1]
opti_as_beta_1 <- ergb$par[2]
```

Schauen wir uns nun die so erhaltene Gerade im Vergleich mit der 'normalen' Regressionsgerade an:

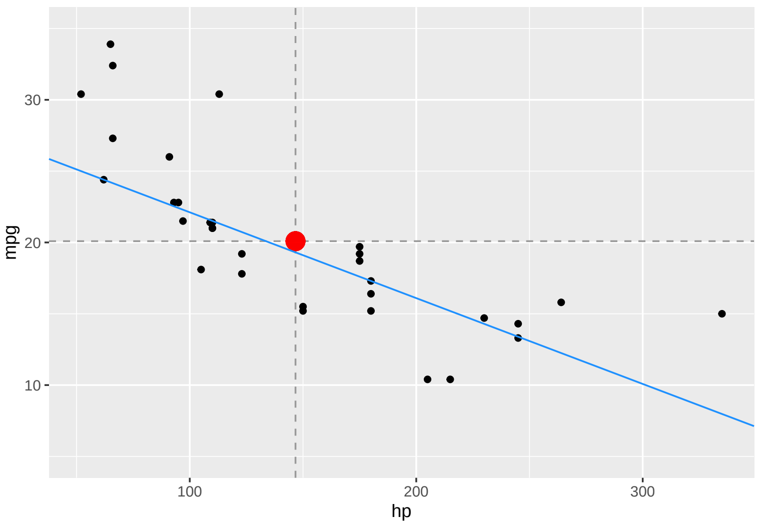


In grün und gestrichelt sehen wir die Gerade aus der *Idee der quadratsichen Abweichungssummen*, in blau die aus der *Idee der absoluten Abweichungssummen*.

Für unser ursprüngliches Problem rechnen wir um:

```
# Umrechnen in die urspüngliche Fragestellung
as_b1 <- opti_as_beta_1 * sd(dt$mpg) / sd(dt$hp)
as_b0 <- (mean(dt$mpg) - as_b1 * mean(dt$hp)) + opti_as_beta_0 * sd(dt$mpg)
```

Und die dazu gehörige Darstellung:



Die Methode nennt sich: **Median-Regression**. U.a. mit dem R-Paket *quantreg* lässt sich dies auch unmittelbar umsetzen:

```
library(quantreg)
ergmedianreg <- rq(mpg ~ hp, data = dt)
coef(ergmedianreg)
```

```
## (Intercept)          hp
## 28.13050847 -0.06016949
```

Als Vergleich können wir uns die Quadratsumme QS und Absulutsumme AS der beiden Modelle einmal ansehen:

```
# Quadratsummen:
qs(b0, b1)
```

```
## [1] 27.861
```

```
qs(as_b0, as_b1)
```

```
## [1] 28.21673
```

```
# Absolutsummen:
as(b0, b1)
```

```
## [1] 963.1637
```

```r
as(as_b0, as_b1)
```

```
## [1] 900.1763
```

1. Idee: Summe der Abweichungen

Wenn wir die Summe der Abweichungen $\sum\limits_{i=1}^{n} \hat{e}_i$ minimieren wollen, dann ist es sinnvoll den Betrag davon zu minimieren. Wir suchen also die Schätzer $\hat{\beta}_0$ und $\hat{\beta}_1$, so dass der Ausdruck

$$\left|\sum_{i=1}^{n} \hat{e}_i\right| = \left|\sum_{i=1}^{n}(\hat{\beta}_0 + \hat{\beta}_1 \cdot x_i - y_i)\right|$$

minimal ist.

Wegen:

$$\begin{aligned}
\sum_{i=1}^{n}(\hat{\beta}_0 + \hat{\beta}_1 \cdot x_i - y_i) &= \sum_{i=1}^{n}\hat{\beta}_0 + \sum_{i=1}^{n}\hat{\beta}_1 \cdot x_i - \sum_{i=1}^{n} y_i \\
&= n \cdot \hat{\beta}_0 + \hat{\beta}_1 \cdot \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} y_i \\
&= n \cdot \hat{\beta}_0 + \hat{\beta}_1 \cdot n \cdot \bar{x} - n \cdot \bar{y} \\
&= n \cdot \left(\hat{\beta}_0 + \hat{\beta}_1 \cdot \bar{x} - \bar{y}\right) \\
&= n \cdot \left(\hat{\beta}_0 - \bar{y} + \hat{\beta}_1 \cdot \bar{x}\right)
\end{aligned}$$

können wir das absolute Mininum bei $hat\beta_0 - \bar{y} = 0$ und $\hat{\beta}_1 \cdot \bar{x} = 0$ erreichen, was zur Lösung $\hat{\beta}_0 = \bar{y}$ und $\hat{\beta}_1 = 0$ führt. Dies ist unser *Nullmodel* in dem die x_i keinen Einfluss auf die y_i haben und wir daher pauschal die y_i mit $\hat{y}_i = \bar{y}$, also dem Mittelwert der y_i abschätzen.