Das Cobweb Modell

Claudius Gräbner

4/14/2020

Inhaltsverzeichnis

1 Einführung

Dieses Dokument beschreibt die Grundstruktur des Cobweb oder Spinnenweben-Modells. Neben der Einführung in das Modell soll dabei auch die Implementierung in R vermittelt werden. Das in diesem Text eingeführte Modell bildet auch die Grundlage für die Shiny-App, welche die Vorlesung ergänzt.

Die folgenden R-Pakete werden dabei im Text verwendet:

```
library(tidyverse)
library(latex2exp)
library(ggpubr)
col_1 <- "#006600"
col_2 <- "#800000"
col_3 <- "#004c93"
col_4 <- "1a171b"
```

2 Theorie

Die Kernidee des Cobweb-Modells ist die verzögerte Reaktion von Produzenten auf Marktsignale. Während der Name auch die Arbeiten von Nicholas Kaldor (1934), der das Modell vor allem für Agrarmärkte für geeignet hielt, zurückgeht, können die theoretischen Ursprünge noch deutlich weiter zurückverfolgt werden (siehe z.B. Ezekiel 1938). Während die Konsumenten ihre Zahlungsbereitschaft nach dem aktuellen Angebot ausrichten entscheiden Anbieter zwar ebenfalls auf Basis der aktuellen Preise, können ihr Angebot aber erst für die nächste Periode anpassen. In Agrarmärkten mag diese Annahme durchaus sinnvoll erscheinen, da Anbieter hier ihre Produktionentscheidung deutlich früher treffen müssen als wenn sie die angebauten Waren tatsächlich verkaufen können.[1]

[1] Ein alternative Interpretation ist, dass wir es mit einem einfachen Prozess der *Erwartungsbildung* zu tun haben: die Anbieter treffen ihre Produktionsentscheidung auf Basis des erwarteten Preises in Periode t, p_t^e, wobei im Cobweb Modell die simplistische Erwartung $p_t^e = p_{t-1}$ angenommen wird.

Entsprehend der Grundidee ist die angebotene Menge eine Funktion der Preise der Vorperiode. Sei $S(p)$ die Angebots- und $D(p)$ die Nachfragefunktion dann gilt für die angebotene Menge:

$$Q_{s,t} = S(p_{t-1})$$

und für die nachgefrage Menge:

$$Q_{d,t} = D(p_t)$$

.

Wir betrachten hier den Fall linearer Angebots- und Nachfragefunktionen:

$$S(p_{t-1}) = \alpha + \beta p_{t-1}$$

und

$$D(p_t) = \gamma + \delta p_t$$

wobei $\beta, \gamma > 0$ und $\alpha, \delta < 0$.

Berechnen wir zunächst den hypothetischen Gleichgewichtspreis p^*. Im Gleichgewicht muss gelten, dass $D(p^*) = S(p^*)$ und somit:

$$\alpha + \beta p^* \stackrel{!}{=} \gamma + \delta p^*$$

Durch einige Umformungen kann p^* nun bestimmt werden:

$$\begin{aligned}
\alpha + \beta p^* &= \gamma \delta p^* \\
\beta p^* - \delta p^* &= \gamma - \alpha \\
p^* &= \frac{\gamma - \alpha}{\beta - \delta}
\end{aligned}$$

In solchen linearen Systemen gilt also $p^* = \frac{\gamma-\alpha}{\beta-\delta}$ (da $\alpha, \delta < 0$ ist $p^* > 0$). Die dazugehörige Gleichgewichtsmenge ist

$$Q^* = \alpha + \beta p^* = \gamma + \delta p^*.$$

Interessanter sind im Cobweb Modell aber die *Anpassungsdynamiken* wenn wir mit einem Nicht-Gleichgewichtspreis beginnen. In anderen Worten: was passiert, wenn der durch die Anbieter angenommene Preis, auf Basis dessen sie ihr Angebot für die Anfangsperiode vorbereiten, nicht gleich dem Gleichgewichtspreis ist? Da sie zu diesem Zeitpunkt keinerlei Feedback durch die Nachfrageseite erhalten haben ist dies nicht unplausibel. Die Frage ist dann: konvergiert der Marktpreis zum theoretischen Gleichgewichtspreis oder nicht?

An dieser Stelle wollen wir uns dieser Frage analytisch nähern. Die etwas intuitiveren Simulationen in R finden Sie im nächsten Abschnitt. Betrachten Sie die folgenden Ausführungen dieses Abschnitts daher als optional.

Wenn wir annehmen, dass der Markt in jeder Periode geräumt wird gilt:

$$\alpha + \beta p_{t-1} = \gamma + \delta p_t = Q_t$$

wobei hier $Q_t \neq Q^*$ solange $p_t \neq p^*$. Entsprechend betrachten wir die *Differenz* von Q_t zu Q^* indem wir die letzte Gleichung von den hypothetischen Gleichgewichtsgrößen substrahieren. Wir erhalten:

$$
\begin{aligned}
\alpha + \beta p_{t-1} - (\alpha + \beta p^*) &= \gamma + \delta p_t - (\gamma + \delta p^*) \\
\alpha + \beta p_{t-1} - \alpha - \beta p^* &= \gamma + \delta p_t - \gamma - \delta p^* \\
\beta p_{t-1} - \beta p^* &= \delta p_t - \delta p^* \\
\beta(p^* - p_{t-1}) &= \delta(p^* - p_t) \\
\frac{\beta}{\delta}(p^* - t_{t-1}) &= (p^* - p_t)
\end{aligned}
$$

Wir definieren nun $\mathcal{A} = \frac{\beta}{\delta}$, $\hat{p}_{t-1} = p^* - p_{t-1}$ und $\hat{p}_t = p^* - p_t$ und bekommen:

$$\hat{p}_t = \mathcal{A}\hat{p}_{t-1}$$

Dieser Ausdruck gibt uns die Abweichung vom Gleichgewichtspreis in t ($\hat{p}_t$) als eine Funktion der Abweichung in der Vorperiode. Für beliebig viele Zeitschritte t gilt dabei:

$$\hat{p}_t = \hat{p}_0 \mathcal{A}^t$$

Nun können wir folgende drei Fälle auf Basis von $\mathcal{A}$ unterscheiden, wobei zu bedenken ist, dass durch $\mathcal{A} = \frac{\beta}{\delta}$ das Verhältnis der Preiselastizitäten des Angebots ($\epsilon_S = \beta$) und der Nachfrage ($\epsilon_D = \delta$) angegeben wird:

- Konvergenz zum Gleichgewichtspreis wenn $|\mathcal{A}| < 1$, also das Angebot elastischer ist als die Nachfrage
- Explodierende Dynamiken weg vom Gleichgewichtspreis wenn $|\mathcal{A}| > 1$, also die Nachfrage elastischer ist als das Angebot
- Dauerhafte Oszillation um den Gleichgewichtspreis wenn $|\mathcal{A}| = 1$, Angebot und Nachfrage also gleich elastisch sind

Dieses Ergebnis wird im Folgenden durch eine Simulation intuitiv dargestellt.

Vorher leiten wir aber noch ein später hilfreiches Zwischenresultat her, mit dem wir den Preis in einer beliebigen Zeitperiode berechnen können. Wir wissen von oben, dass:

$$\hat{p}_t = \hat{p}_0 \mathcal{A}^t$$

Da $\hat{p}_t = p^* - p_t$ gilt:

$$
\begin{aligned}
p^* - p_t &= \hat{p}_0 \mathcal{A}^t \\
-p_t &= \hat{p}_0 \mathcal{A}^t - p^*
\end{aligned}
$$

$$p_t = p^* - \mathcal{A}^t \hat{p}_0$$

Diese Formel wird später bei der Simulation hilfreich.

3 Implementierung in R

3.1 Kernmechanismen

In diesem Beispiel arbeiten wir wie oben mit klassischen linearen Angebots- und Nachfragekurven. Beachten Sie dabei, dass die Steigungsparameter hier gleichzeitig die Preiselastizitäten von Angebot und Nachfrage darstellen.

```
supply_func <- function(p, intercept_s, slope_s){
  intercept_s + slope_s * p
}

demand_func <- function(p, intercept_d, slope_d){
  intercept_d + slope_d * p
}
```

Zudem definieren wir auch noch inverse Angebots- und Nachfragefunktionen, die uns für eine gegebene Menge den entsprechenden Preis angeben:

```
supply_func_inv <- function(q, intercept_s, slope_s){
  (q - intercept_s) / slope_s
}

demand_func_inv <- function(q, intercept_d, slope_d){
  (q-intercept_d) / slope_d
}
```

Betrachten wir ein Beispiel Angebots-Nachfrage-Diagramm mit folgenden Parametern:

```
intercept_angebot <- -0.5
intercept_nachfrage <- 5.0
slope_angebot <- 0.6
slope_nachfrage <- -0.7
```

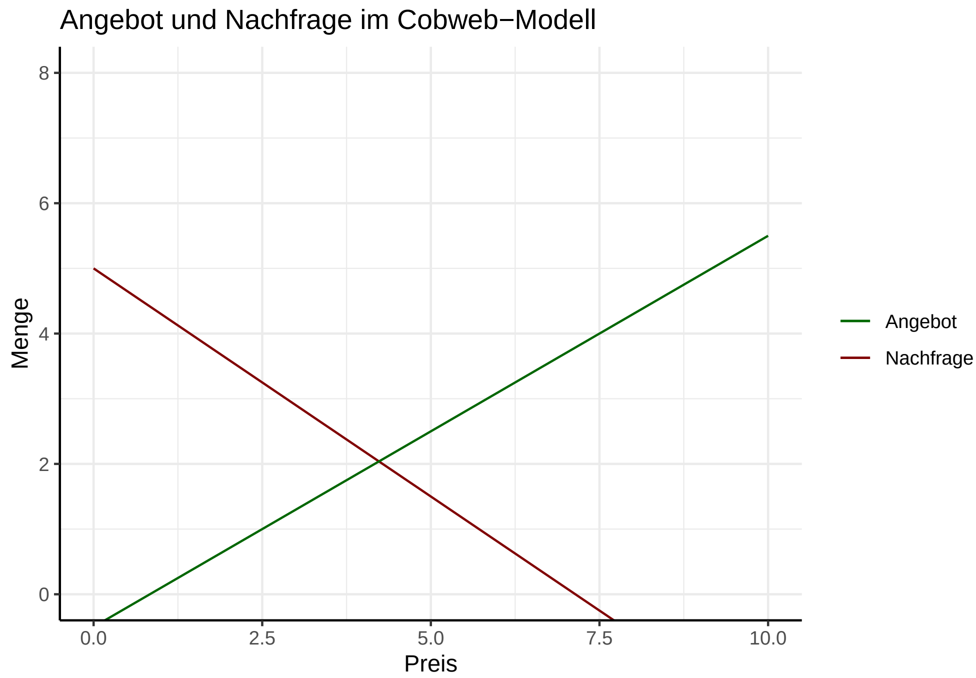
Wir definieren vorher noch eine Hilfsfunktion zum plotten:

```
make_supply_demand_plot <- function(price_range,
                                    intercept_nachfrage, intercept_angebot,
                                    slope_nachfrage, slope_angebot){
  demand <-demand_func(
    price_range,
    intercept_d = intercept_nachfrage,
    slope_d = slope_nachfrage)

  supply <-supply_func(price_range,
                       intercept_s = intercept_angebot,
                       slope_s = slope_angebot)

  demand_supply_data <- tibble(
    Nachfrage=demand, Angebot=supply, Preis=price_range, Menge=price_range)

  ggplot(data=demand_supply_data,
         mapping=aes(x=Preis)) +
    geom_line(aes(x=Preis, y=Nachfrage, color="Nachfrage")) +
    geom_line(aes(x=Preis, y=Angebot, color="Angebot")) +
    xlab("Preis") + ylab("Menge") +
```

```
    coord_cartesian(ylim = c(0, 8)) +
    scale_color_manual(values = c(col_1, col_2)) +
    ggtitle("Angebot und Nachfrage im Cobweb-Modell") +
    theme_bw() +
    theme(panel.border = element_blank(),
          axis.line = element_line(),
          legend.title = element_blank())
  }
```

Nun können wir derlei Abbildungen leichter erstellen:

```
make_supply_demand_plot(seq(0, 10, 0.1),
                        intercept_nachfrage, intercept_angebot,
                        slope_nachfrage, slope_angebot)
```



Strikt genommen sehen wir hier nur die *langfristige* Angebotskurve. Denn aus den oben beschriebenen Gleichungen ergibt sich ja, dass das Angebot im Bezug auf p_t *vollkommen unelastisch* ist. Die Anbieter können auf Preisänderungen in der aktuellen Periode in diesem Modell ja gar nicht reagieren. Erst in der nächsten Runde ist eine Reaktion möglich. Daher wäre, wenn wir die x-Achse in der Abbildung als p_t interpretieren eine vertikale Angebotskurve passender. Wir können das Diagram also entweder als Beschreibung der langfristigen Situation im Gleichgewicht betrachten, oder aber - besser - wir beachten, dass für das Angebot die x-Achse den Preis in $t-1$, die die Nachfrage aber den Preis in t angibt. Letztere Interpretation macht es weiter unten leichter die Anpassungsdynamiken zu visualisieren.

Um die Preisdynamiken zu visualisieren schreiben wir eine Funktion, die p_{t+1} aus p_t berechnet. Dazu verwenden wir das in Abschnitt 2 hergeleitete Resultat $p_t = p^* - \mathcal{A}^t \hat{p}_0$:

```
price_func <- function(p_0, intercept_demand, slope_demand,
                       intercept_supply, slope_supply,
                       timesteps){
  p_eq <- (intercept_supply-intercept_demand)/(slope_demand-slope_supply)
  p_t <- p_eq - (slope_supply/slope_demand)**timesteps * (p_eq-p_0)
  return(p_t)
}
```

Nun können wir recht einfach unterschiedliche Fälle vergleichen. Man wird sehen, dass die Konvergenzeigenschaften von den relativen Steigungen der Angebots- und Nachfragefunktion abhängen, also des relativen Preiselastizitäten, so wie im zweiten Abschnitt analytisch hergeleitet.

```
p_init <- 2.5
intercept_angebot <- -0.5
intercept_nachfrage <- 5.0
slope_angebot <- 0.3
slope_nachfrage_small <- -0.2
slope_nachfrage_med <- -0.3
slope_nachfrage_large <- -0.4

t_input <- seq(0, 20)

p_dynamics_d_greater <- price_func(
  p_init, intercept_nachfrage, slope_nachfrage_large,
  intercept_angebot, slope_angebot, t_input)

p_dynamics_d_smaller <- price_func(
  p_init, intercept_nachfrage, slope_nachfrage_small,
  intercept_angebot, slope_angebot, t_input)

p_dynamics_equal <- price_func(
  p_init, intercept_nachfrage, slope_nachfrage_med,
  intercept_angebot, slope_angebot, t_input)

demand_supply_dynamics <- tibble(
  Zeit=t_input,
  Preisdynamik1=p_dynamics_d_greater,
  Preisdynamik2=p_dynamics_d_smaller,
  Preisdynamik3=p_dynamics_equal)
```

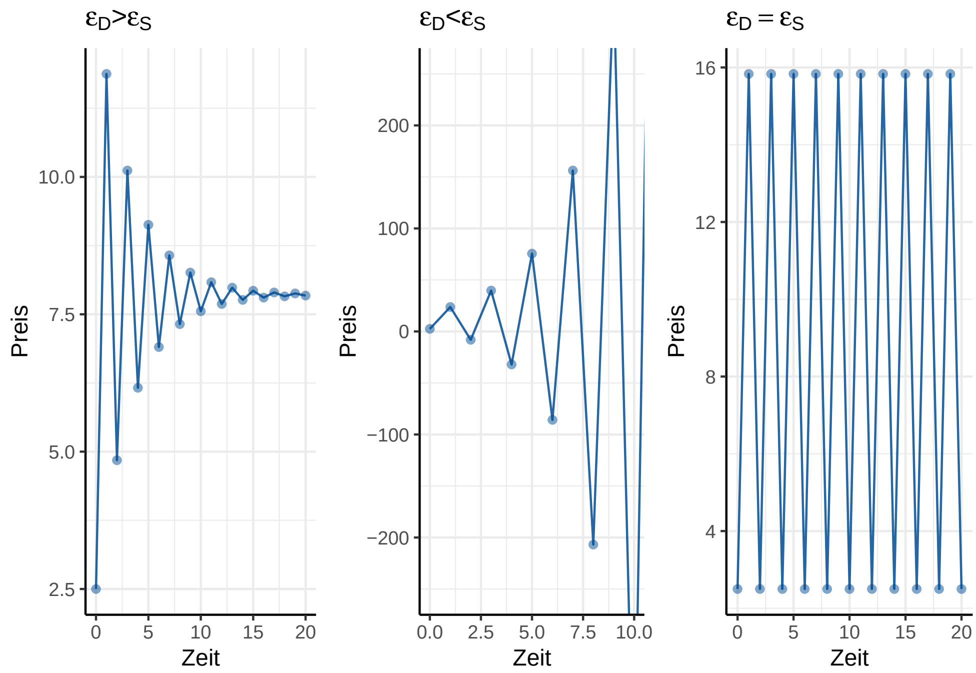
Der Einfachheit halber definieren wir folgende Hilfsfunktion für die dynamische Abbildung:

```
get_dyn_ggplot <- function(data, y_val){
  ggplot(data, aes_string(x="Zeit", y=y_val)) +
  geom_line(color=col_3, alpha=0.85) +
  geom_point(color=col_3, alpha=0.5) +
  ylab("Preis") +
  theme_bw() +
  theme(panel.border = element_blank(),
                           axis.line = element_line(),
                           legend.title = element_blank())
}
```

Diese Funktion verwenden wir für die dynamischen Plots:

```
p1 <- get_dyn_ggplot(demand_supply_dynamics, "Preisdynamik1") +
  ggtitle(TeX("$\\epsilon_D>\\epsilon_S$"))

p2 <- get_dyn_ggplot(demand_supply_dynamics, "Preisdynamik2") +
  ggtitle(TeX("$\\epsilon_D<\\epsilon_S$")) +
  coord_cartesian(xlim = c(0, 10), ylim = c(-250, 250))

p3 <- get_dyn_ggplot(demand_supply_dynamics, "Preisdynamik3") +
  ggtitle(TeX("$\\epsilon_D=\\epsilon_S$"))
```

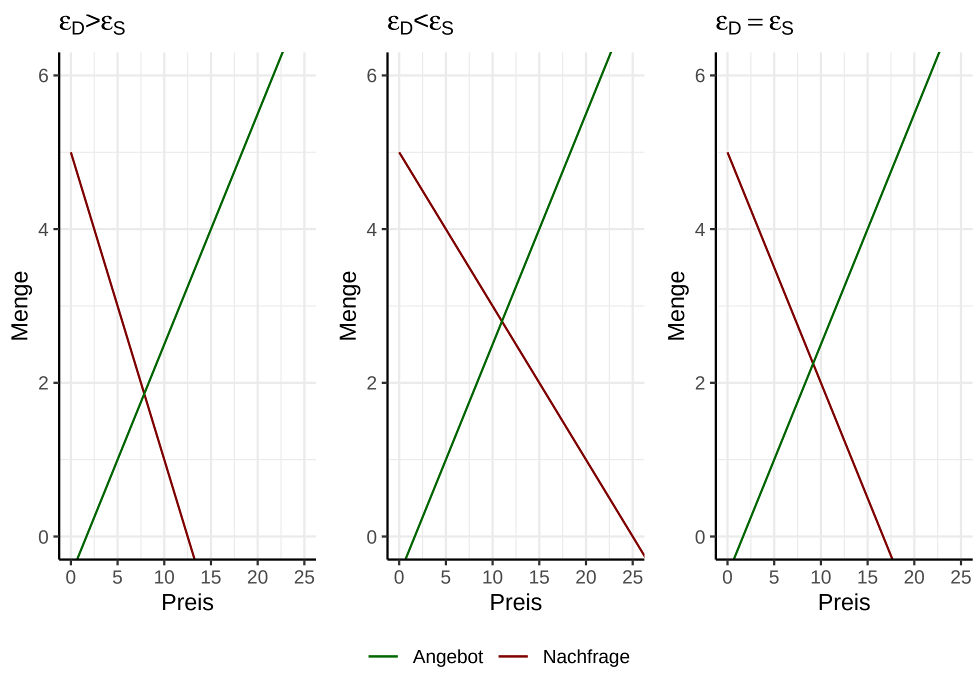
```r
p_full <- ggarrange(p1, p2, p3, ncol = 3)
p_full
```



Zur besseren Darstellung ergänzen wir noch die klassische Angebots-Nachfrage-Darstellung:

```r
p_range <- seq(0, 40, 0.1)
af1 <- make_supply_demand_plot(p_range, intercept_nachfrage, intercept_angebot,
                               slope_nachfrage_large, slope_angebot) +
  coord_cartesian(xlim = c(0, 25), ylim = c(0,6))  +
  ggtitle(TeX("$\\epsilon_D>\\epsilon_S$"))
af2 <- make_supply_demand_plot(p_range, intercept_nachfrage, intercept_angebot,
                               slope_nachfrage_small, slope_angebot) +
  coord_cartesian(xlim = c(0, 25), ylim = c(0,6)) +
  ggtitle(TeX("$\\epsilon_D<\\epsilon_S$"))
af3 <- make_supply_demand_plot(p_range, intercept_nachfrage, intercept_angebot,
                               slope_nachfrage_med, slope_angebot) +
  coord_cartesian(xlim = c(0, 25), ylim = c(0,6)) +
  ggtitle(TeX("$\\epsilon_D=\\epsilon_S$"))
af_full <- ggarrange(af1, af2, af3, ncol = 3, common.legend = T, legend = "bottom")
```

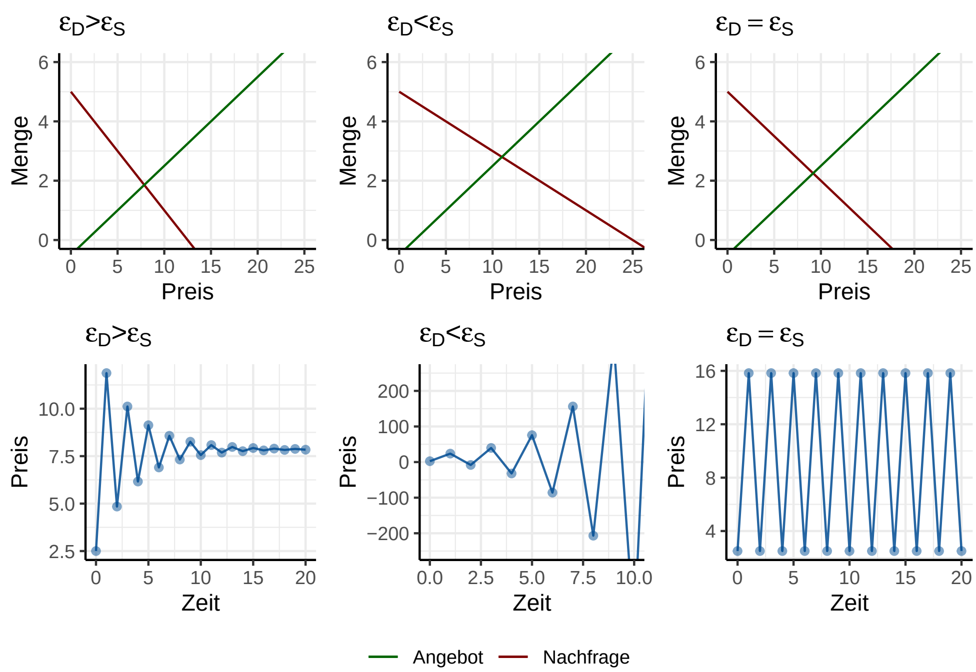
```r
af_full
```



Und produzieren folgenden Überblickplot:

```
p_complete <- ggarrange(
    af1, af2, af3,
    p1, p2, p3,
  ncol = 3, nrow = 2, common.legend = T, legend = "bottom")
```

```
p_complete
```



3.2 Zusammenfassende Beschreibung des Anpassungsalgorithmus

Im Folgenden soll noch einmal zusammengefasst werden wie der Anpassungsprozess im Cobweb-Modell stattfindet. Dazu gehen wir davon aus, dass $p_0 \neq p^*$ und nehmen folgende Beispielparameter an:

```
p_init <- 2.5
intercept_angebot <- -0.5
```

```r
intercept_nachfrage <- 5.0
slope_angebot <- 0.6
slope_nachfrage <- -0.7
```

1. Als erstes müssen die Anbieter entscheiden, welche Menge Sie produzieren möchten. Diese Menge bestimmt sich nach p_0, der aber nicht Ergebnis eines Aushandlungsprozesses zwischen Angebot und Nachfrage ist, sondern von der Angebotsseite angenommen wurde. Denn bislang haben ja noch keine Marktinteraktionen stattgefunden, aber die Produzenten müssen ja dennoch bereits entscheiden wie viel sie produzieren sollen. Alternativ können wir annehmen, dass p_0 der Gleichgewichtspreis aus vergangenen Interaktionen ist, seitdem aber ein exogener Angebots- oder Nachfrageschock eingetreten ist. In jedem Fall ist diese angebotene Menge bei unserer Parametrisierung gegeben durch:

```r
q_s_1 <- supply_func(p_init, intercept_angebot, slope_angebot)
q_s_1
```

```
#> [1] 1
```

da

$$-0.5 + 0.6 \cdot 2.5 = 1.0$$

2. Nun sind wir in der eigentlichen ersten Runde. Die Nachfrager würden bei p_0 gerne die folgende Menge des Gutes erwerben:

```r
q_d_1_hyp <- demand_func(p_init, intercept_nachfrage, slope_nachfrage)
q_d_1_hyp
```

```
#> [1] 3.25
```

da

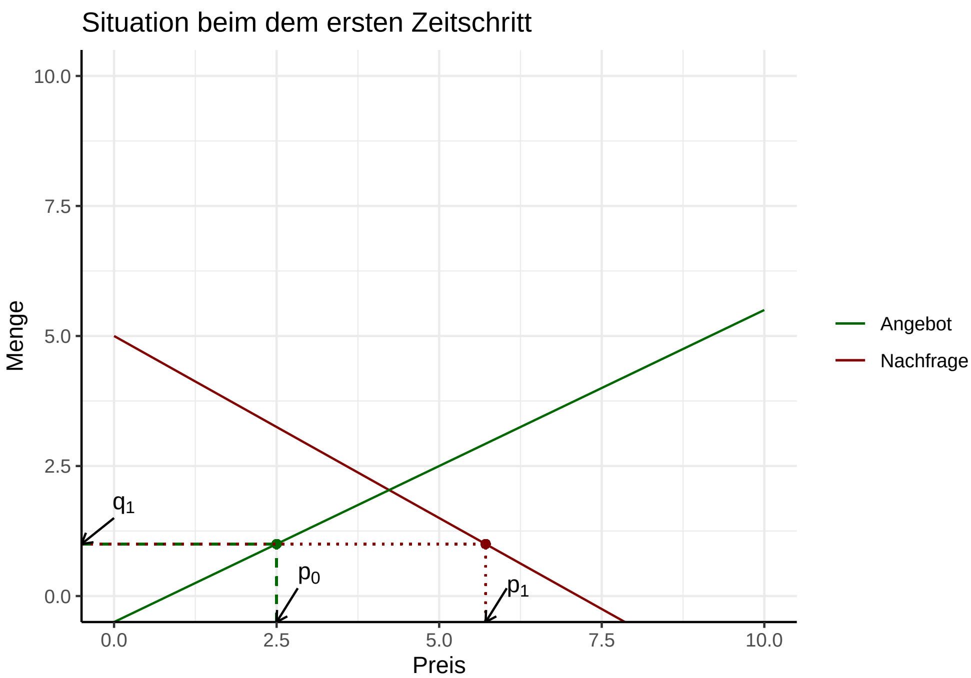
$$5.0 - 0.4 \cdot 2.5 = 4$$

Es gibt also ein Unterangebot! In dieser Situation kann nur 1 Einheit gehandelt werden! Zu welchem Preis wird diese Menge gehandelt? Dazu betrachten wir die inverse Nachfragefunktion:

```r
p_d_1 <- demand_func_inv(q_s_1, intercept_nachfrage, slope_nachfrage)
p_d_1
```

```
#> [1] 5.714286
```

Hier findet der Handel jedoch zu einem Preis von $p_1 = 5.71$ statt, da in der aktuellen Situation nur die kaufkräftigsten Nachfrager zum Zuge kommen und der markträumende Preis für ein Angebot von 1 Einheit 5.71 beträgt. Diese Situation ist in folgender Abbildung dargestellt:



3. Die Produzenten entscheiden jetzt was sie nächste Runde produzieren sollen. Da dieses mal der Marktpreis 5.71 betrug und das Angebot in $t+1$ auf Basis von p_t gebildet wird, berechnen wir die Angebotsmenge in t_2 folgendermaßen:

```
q_s_2 <- supply_func(p_d_1, intercept_angebot, slope_angebot)
q_s_2
```

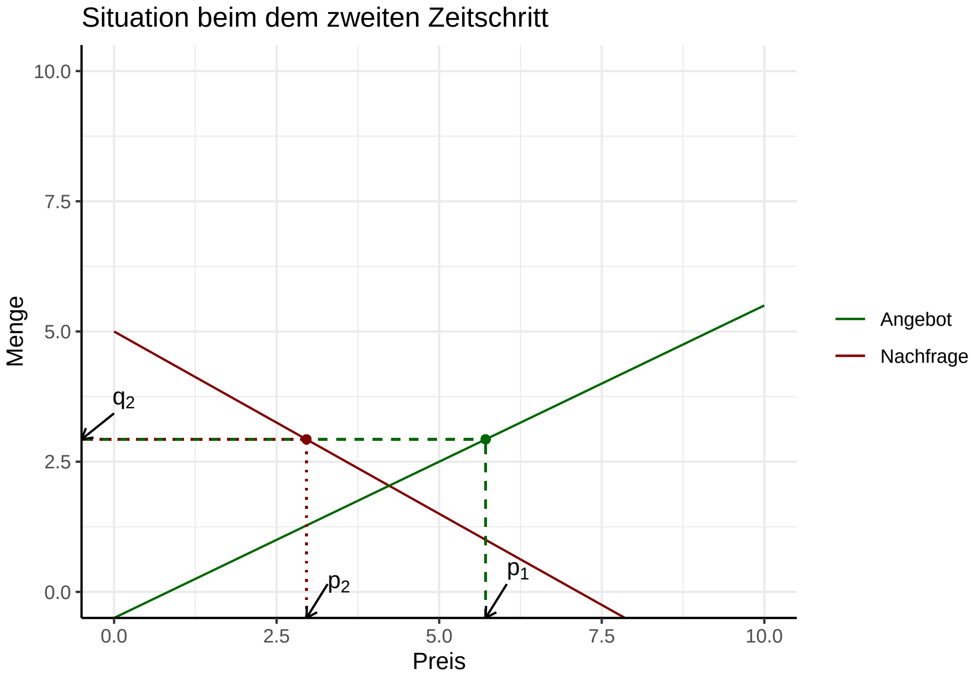
```
#> [1] 2.928571
```

4. Die Produzenten haben sich entschieden $q_2 = 2.93$ Einheiten anzubieten. Was ist der Preis, bei dem der Markt in diesem Fall geräumt wird? Dazu betrachten wir den Preis, den die Nachfrager bereit sind, für eine solche Angebotsmenge zu zahlen:

```
p_d_2 <- demand_func_inv(q_s_2, intercept_nachfrage, slope_nachfrage)
p_d_2
```

```
#> [1] 2.959184
```

Es kommt also zu Handel zum Preis von 2.96. Grafisch sieht die Sache so aus:



5. Nun entscheiden die Anbieter wieder welche Menge in der nächsten Runde anzubieten ist. Dafür legen sie diesmal natürlich $p = 2.96$ zugrunde, sodass sich die folgende Angebotsmenge ergibt:

```
q_s_3 <- supply_func(p_d_2, intercept_angebot, slope_angebot)
q_s_3
```

```
#> [1] 1.27551
```

6. So geht es nun immer weiter bis wir irgendwann bei einerm Marktpreis von $p = 4.23$ und einer Handelsmenge von $q = 2.04$ landen. Wir sehen, dass sich diese Preis-Mengen-Kombination *selbst reproduziert*:

```
p_eq <- (intercept_angebot-intercept_nachfrage)/(slope_nachfrage-slope_angebot)
q_d <- demand_func(p_eq, intercept_nachfrage, slope_nachfrage)
q_s <- supply_func(p_eq, intercept_angebot, slope_angebot)
print(paste0("Nachfrage bei p*: ", q_d))
```

```
#> [1] "Nachfrage bei p*: 2.03846153846154"
```

```
print(paste0("Angebot bei p*: ", q_s))
```

```
#> [1] "Angebot bei p*: 2.03846153846154"
```

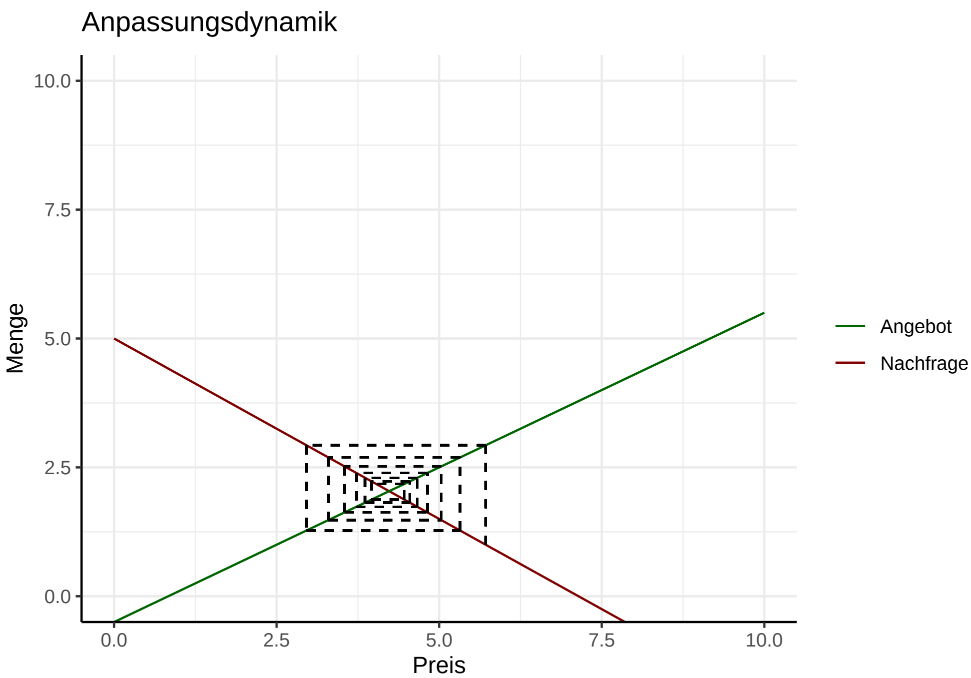
```
p_d <- demand_func_inv(q_s, intercept_nachfrage, slope_nachfrage)
p_s <- supply_func_inv(q_s, intercept_angebot, slope_angebot)
print(paste0("Preis bei dem q* nachgefragt wird: ", p_d))
```

```
#> [1] "Preis bei dem q* nachgefragt wird: 4.23076923076923"
```

```
print(paste0("Preis bei dem q* angeboten wird: ", p_s))
```

```
#> [1] "Preis bei dem q* angeboten wird: 4.23076923076923"
```

Das bedeutet, dass wir in unserem Fall schlussendlich bei der *Gleichgewichtsmenge* und dem *Gleichgewichtspreis* angekommen sind. Dies ist der Fall, da wir es mit einer Situation zu tun haben, in der die Nachfrage preis-elastischer ist als das Angebot. Die folgende Abbildung illustriert hier noch einmal den gesamten Prozess:



In anderen Konstellationen wäre eine solche Konvergenz unter Umständen nicht zu erwarten. Sie können die oben diskutierten Beispiele dabei gerne mit konkreten Zahlenbeispielen selbst durchrechnen, bzw. mit der Shiny-App experimentieren.