6 Markov Chain Monte Carlo

6.1 Einführung

Von den Herausgebern der Zeitschrift *Computing in Science and Engineering* wurden im Jahre 2000 zehn Algorithmen ausgewählt, die ihrer Ansicht nach die grösste Bedeutung für die Wissenschaft und Technik im 20. Jahrhundert hatten. Einer davon ist die *Markov-Chain-Monte-Carlo-Methode (MCMC)*, die im Folgenden kurz erläutert wird. Wir werden uns zuerst nur mit dem Fall einer diskreten Zufallsvariable mit endlich vielen Werten beschäftigen, und dann die Methode auf abzählbare Wertebereiche und für stetige Zufallsvariablen (auch in höheren Dimensionen) übertragen.

Häufig benötigt man Zufallszahlen, die aus einer bestimmten Verteilung stammen. Zum Beispiel möchte man oft Erwartungswerte einer Funktion $g(X)$ von Zufallsvariablen berechnen. Für eine Verteilung $\vec{\pi}$ auf der endlichen Menge $S = \{1, \ldots, N\}$ möchte man gerne

$$E(g(X)) = \sum_{i=1}^{N} g(i)\pi_i$$

berechnen, wobei X eine Zufallsvariable mit Verteilung $\vec{\pi}$ ist. Kann man das Problem nicht analytisch lösen (was bei Verteilungen auf endlichen Mengen meistens leicht ist), so kann man aus der Verteilung $\vec{\pi}$ vielleicht eine unabhängig identisch verteilte Stichprobe $X_1, X_2, \ldots, X_n$ ziehen. Wegen des *(starken) Gesetzes der grossen Zahlen*

$$\lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} g(X_k) = E(g(X))$$

konvergiert das arithmetische Mittel der X_k mit Wahrscheinlichkeit 1 gegen $E(g(X))$, wenn die Stichprobe nur gross genug ist.

Die Erzeugung von iid-verteilten Zufallszahlen ist also extrem wichtig. Nun ist es allerdings gar nicht so einfach (effektiv) die benötigten Zufallszahlen aus einer beliebigen Verteilung zu ziehen. Für einfache Verteilungen kommt man mit uniform verteilten Zufallszahlen und Transformationen weiter – aber eben nicht für beliebige Verteilungen. Eine weitere Möglichkeit ist das sogenannte *Rejection Sampling*, welches allerdings für grössere Probleme nicht mehr praktisch anwendbar ist.

Die Idee ist es nun, eine Markov-Kette durch Definition der Übergangsmatrix so zu erzeugen, dass diese asymptotisch der gewünschten Verteilung π_i folgt. Wie wir in (2.8) gezeigt haben, sind für aperiodische und irreduzible Kettern die Aufenthaltsdauern in den Zuständen entlang einer Trajektorie (Längsschnitt) für lange Zeiten gemäss der (in diesem

Falle eindeutigen) stationären Verteilung π^* verteilt. Die Zufallszahlen $x_1, \ldots, x_n$ sind also die Trajektorie der Markov-Kette. Diese Zufallszahlen sind natürlich nicht unabhängig, aber für die Schätzung von Erwartungswerten macht das nichts aus.

Wir müssen uns also eine Übergangsmatrix geeignet konstruieren. Gesucht ist also eine aperiodische, irreduzible Markov-Kette mit stationärer Verteilung $\vec{\pi^*}$. In diesem Fall konvergieren die $\vec{\pi}(t)$ unabhängig vom Startzustand gegen $\vec{\pi}$, wobei allerdings die Geschwindigkeit, mit der dies geschieht, in der Regel schon von der Startverteilung abhängt. Gesucht ist also eine Übergangsmatrix $\mathbf{P}$ mit

$$\vec{\pi} \cdot \mathbf{P} = \vec{\pi}, \tag{6.1}$$

so dass die zugehörige Markov-Kette aperiodisch und irreduzibel ist. Wir können sogar eine Markov-Kette konstruieren, die die sogenannte *Detailed-Balance-Bedingung* erfüllt. Diese verlangt, das für alle $i, j \in S$ gilt:

$$\pi_i p_{ij} = \pi_j p_{ji}, \tag{6.2}$$

Diese Bedingung bedeutet also, dass (nach der Einschwingphase) zu jedem Zeitpunkt und für alle i und j die (unbedingte) Wahrscheinlichkeit, dass ein Sprung von i nach j stattfindet, genau so gross ist wie die Wahrscheinlichkeit für einen Sprung von j nach i. Bedingung (6.2) impliziert (6.1), denn für festes j erhält man bei Summation von (6.2) über alle i

$$\sum_{i=1}^{N} \pi_i p_{ij} = \sum_{i=1}^{N} \pi_j p_{ji},$$

bzw.

$$\pi_j = \pi_j \sum_{i=1}^{N} p_{ji},$$

und damit (nach Zusammenfassen in einem Vektor für alle j):

$$\vec{\pi} \cdot \mathbf{P} = \vec{\pi}.$$

Kann man also eine Matrix $\mathbf{P}$ finden, so dass (6.2) erfüllt ist, so hat man eine Markov-Kette mit der gewünschten stationären Verteilung. Es gibt im Allgemeinen viele Möglichkeiten, eine Übergangsmatrix $\mathbf{P}$ zu konstruieren, so dass (6.2) gilt.[1]

Wir betrachten hier einen zweistufigen Prozess, da eine direkte Konstruktion geeigneter p_{ij} schwierig ist. Ausgehend vom Zustand i wird zunächst mit der Wahrscheinlichkeit q_{ij} ein neuer Zustand j *vorgeschlagen*. Dieser neue Zustand wird nun mit der Wahrscheinlichkeit

[1] Allerdings hängen viele Eigenschaften des Verfahrens von der Wahl ab, so z.B. die Geschwindigkeit, mit der die stationäre Verteilung erreicht wird. Wir gehen nicht weiter darauf ein.

α_{ij} *angenommen* – wird er nicht angenommen, bleibt die Kette zum nächsten Zeitpunkt im bisherigen Zustand. Der Trick ist nun, die α_{ij} so zu wählen, dass gilt:

$$p_{ij} = q_{ij} \cdot \alpha_{ij}, \tag{6.3}$$

Die *Vorschlagsmatrix* $\mathbf{Q}$ ist im Allgemeinen vorgegeben, so dass wir *Akzeptanzwahrscheinlichkeit* α_{ij} so bestimmen müssen, dass die Detailed-Balance-Bedingung (6.2) erfüllt ist. Die einfachste und älteste Wahl von α_{ij} führt zum sogenannten *Metropolis-Algorithmus*:

$$\alpha_{ij} = \min\left\{1, \frac{\pi_j q_{ji}}{\pi_i q_{ij}}\right\}. \tag{6.4}$$

Dieses Verfahren wurde erstmals 1953 von den Physikern Metropolis, Rosenbluth und Teller und deren Ehefrauen veröffentlicht. Hierbei ist zu beachten, dass in (6.4) zwar π_j und π_i auftauchen, man aber nur ihren Quotienten benötigt. Das ist dann von grossem Vorteil, wenn man π nur bis auf eine Konstante kennt, die sich dann rauskürzt. Dies ist zum Beispiel in der Bayes-Statistik häufig der Fall, siehe Abschnitt 6.2.

Man erhält also folgenden Algorithmus (Metropolis-Algorithmus):

1. Initialisiere x_0 beliebig mit einem der Zustände aus S.
2. Generiere ausgehend von x_t einen Vorschlag x^* aus der Markov-Kette mit Übergangsmatrix $\mathbf{Q}$.
3. Setze mit Wahrscheinlichkeit $\min\left\{1, \frac{\pi_{x^*} q_{x^*,x_t}}{\pi_{x_t} q_{x_t,x^*}}\right\}$ $x_{t+1} = x^*$, ansonsten $x_{t+1} = x_t$.
4. Wiederhole Schritte 2 und 3, bis die Realisation der Kette die gewünschte Länge T hat.

Häufig kann man $\mathbf{Q}$ als symmetrische Matrix mit $q_{ij} = q_{ji}$ wählen, d.h. für die zugehörige Markov-Kette ist die bedingte Wahrscheinlichkeit für einen Sprung nach j bei aktuellem Aufenthalt in i gleich der bedingte Wahrscheinlichkeit für einen Sprung nach i bei aktuellem Aufenthalt in j. In diesem Fall vereinfacht sich (6.4) zu

$$\alpha_{ij} = \min\left\{1, \frac{\pi_j}{\pi_i}\right\}.$$

Der oben angegebene Algorithmus lässt sich sehr einfach auch auf den Fall von diskreten Verteilungen mit abzählbar unendlichem Zustandsraum oder stetigen Verteilungen (im letzteren Fall benutzt man Dichten statt Wahrscheinlichkeiten) übertragen.

Im Falle von stetigen Zufallsvariablen ist der Zustand x_t nun kontinuierlich oder sogar ein mehrdimensionaler Vektor. Am Prinzip des Metropolis-Algorithmus ändert sich wenig. Nur im Schritt 2 wird ausgehend von einem Zustand x_t, der nun kontinuierlich oder sogar mehrdimensional sein kann, ein neuer Vorschlagszustand x^* bestimmt. Dies geschieht zum Beispiel durch Ziehung von x^* aus einer Normalverteilung $x^* \sim N(x_t - x^*, \sigma^2)$. Die Wahl des Parameters σ^2 bestimmt die Geschwindigkeit, mit der die stationäre Verteilung

erreicht wird. Auch hier gehen wir nicht weiter auf solche technischen Details ein. Die Vorschlagsmatrix **Q** ist also zu einer *Vorschlagsdichte* $Q(x, x^*)$ geworden, die im Falle der Normalverteilung wie gerade beschrieben symetrisch ist $Q(x, x^*) = Q(x^*, x)$ und sich somit in (6.4) ebenfalls rauskürzt. Der Rest des Algorithmus bleibt gleich.

Beispiel Wir möchten Zufallszahlen erzeugen, deren Dichte im Intervall $[0, \pi]$ die Form der Sinusfunktion hat und die sonst 0 ist. Wir können den Metropolis-Algorithmus, wie folgt in `R` implementieren:

```
set.seed(0)
unnorm_density <- function(x) {
  ifelse (x < 0 | x > pi, 0, sin(x))
}

simu_mcmc <- function(steps, sd=0.5) {
  mcmc <- rep(NA, steps)
  # Step 1 (Initialisierung)
  x <- 0.2
  for (i in 1:length(mcmc)) {
    # Step 2 (Vorschlag)
    x_star <- rnorm(1, mean=x, sd=sd)
    # Step 3 (Vorschlagsdichte Q kürzt sich raus)
    alpha <- min(1, unnorm_density(x_star)/unnorm_density(x))
    x <- sample(c(x_star, x), prob = c(alpha, 1-alpha), size=1)
    mcmc[i] <- x #Wir merken uns den Zustand
  }
  mcmc
}
```

Man beachte, dass man die Normierungskonstante der Dichte nicht kennen muss. D.h. für den MCMC-Algorithmus müssen wir das Intergral $\int_0^\pi \sin(x)\, dx = 2$ nicht berechnen. Das mag hier trivial sein, für komplizierte mehrdimensionale Dichten ist die Berechnung des Integrals oft analytisch nicht möglich und auch numerisch anspruchsvoll. In der Abbildung (6.1) ist das Ergebnis des Algorithmus dargestellt. Auf der linken Seite, sieht man 3 kurze Markovketten die mit unterschiedlichen Parametern `sd` der Vorschlagsdichte realisiert wurden. Für `sd`=0.01 liegt der vorgeschlagene Zustand $x^*(t)$ meist in der Nähe des alten Zustandes, und so dauert es eine Weile bis die Kette den Anfangszustand verlassen hat. Ausserdem sind die $x^(t)$ mit dieser Wahl von `sd` stark korreliert und man erzeugt nicht wirklich unabhängige Zufallszahlen. Man darf *sd* nun allerdings auch nicht einfach zu gross machen. Wählt man `sd` = 500 so landet fast jeder Vorschlag in einem Bereich in dem die Dichte 0 ist und somit wird der neue Zustand im Schritt 3 zu selten angenommen. Eine geeignetete Wahl ist hier `sd`=0.5. Ein Histogramm der besuchten Zuständer einer Markov-Kette der Länge 10000 ist auf der rechten Seite von (6.1) dargestellt. Wie man sieht, stammen die Zufallszahlen tatsächlich aus der gewünschten Verteilung.

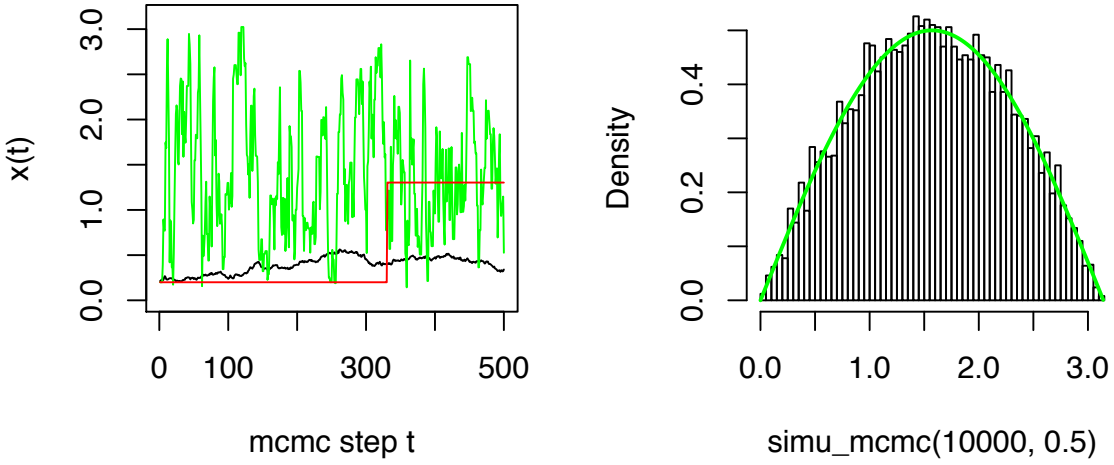

Abbildung 6.1: MCMC für das Beispiel. Links: 3 Realisationen der ersten 500 Schritte einer MCMC-Simulation, die mit unterschiedlichen Breiten ($sd = 0.01, 0.5, 500$ in schwarz, grün und rot) der Vorschlagsverteilung simuliert wurden. Rechts: Histogramm der besuchten Zustände sowie die theoretische Dichte

6.2 Markov Chain Monte Carlo in der Bayes-Statistik

Wir wollen nun zeigen, wie man die MCMC-Methode verwenden kann, um A-Posteriori-Wahrscheinlichkeiten $P(\Theta = \theta_i | X = x)$ zu bestimmen. Wir erinneren uns: In der bayesianischen Statistik sind die Parameter Zufallsvariablen und wir sind an der Wahrscheinlichkeitsfunktion (oder Wahrscheinlichskeitsdichte) $P(\Theta|X = x)$ interesiert, also der Wahrscheinlichkeit, dass ein Parameter bei gegebenen Daten und entsprechendem Vorwissen einen bestimmten Wert hat. Bis zur Anwendung des MCMC-Verfahrens konnte die bayesianische Statstik nur auf einfache Probleme angewendet werden. Beginnend in den Achtizerjahren des letzten Jahrhunderts wurde dann, auch aufgrund der immer einfacheren Verfügbarkeit von Computern, die MCMC-Methode vermehrt in der bayesianische Statistik angewendet. Man spricht in diesem Zusammenhang auch von der MCMC-Revolution.

Wir betrachten folgende Situation: Wir möchten einen eindimensionalen Parameter Θ, der die diskreten Werte $\theta_1, \ldots, \theta_N$ annehmen kann. Unser Vorwissen beschrieben wir durch eine A-Priori-Verteilung, d.h. wir kennen für alle θ_i

$$P(\Theta = \theta_i).$$

In einem Experiment erheben wir eine Beobachtung X, deren (diskrete) Verteilung von Θ

abhängt[2]. Wir kennen also ausserdem für jedes x und jedes θ_i

$$P(X = x|\Theta = \theta_i).$$

Als Funktion von θ_i betrachtet, ist dies die *Likelihood*.

In der Bayes-Statistik wollen wir nun die Information aus der Beobachtung (oder den Beobachtungen) mit der A-Priori-Verteilung zur *A-Posteriori-Verteilung* kombinieren, also der bedingten Verteilung des Parameters gegeben die Daten. Nach dem Satz von Bayes ergibt sich:

$$P(\Theta = \theta_i|X = x) = \frac{P(X = x|\Theta = \theta_i)P(\Theta = \theta_i)}{P(X = x)} = \frac{P(X = x|\Theta = \theta_i)P(\Theta = \theta_i)}{\sum_{j=0}^{n} P(X = x|\Theta = \theta_j)}$$

In der Regel ist die Likelihood $P(X = x|\Theta = \Theta_i)$ bekannt bzw. einfach zu berechnen, während $P(X = x) = \sum_{j=0}^{n} P(X = x|\Theta = \theta_j)$ schwerer zu bestimmen ist. Andererseits ist $P(X = x)$ eine Normierungskonstante, die zwar von den Daten, aber nicht von Θ abhängt. Sie sorgt nur dafür, dass die A-Posteriori-Verteilung tatsächlich eine Wahrscheinlichkeitsverteilung ist – die Form wird von ihr nicht beeinflusst. Obwohl $P(\Theta = \theta_i|X = x)$ also schwierg vollständig zu berechnen ist, ist die Berechnung bis auf eine Konstante relativ einfach. Man kann nun – wie im letzten Abschnitt gesehen – aus nur bis auf eine Konstante bekannten Verteilungen Stichproben mit Hilfe der MCMC-Methode simulieren, indem man eine Markov-Kette konstruiert, deren Zustandsraum der Parameterraum $\{\theta_1, \theta_1, \ldots, \theta_N\}$ ist und deren asymptotische Verteilung genau der A-Posteriori-Verteilung entspricht. Man beachte, dass selbst wenn man die A-Posteriori-Verteilung $P(\Theta = \theta_i|X = x)$ einfach berechenen könnte, die Ziehnung von Zufallszahlen aus einer Verteilung immer noch ein Problem darstellen würde. Die MCMC-Methode löst das Problem mit der Konstante und der Ziehung von Zufallsvariablen aus einer Verteilung.

Ziel ist also die Simulation einer Stichprobe aus der Verteilung $\vec{\pi}$ mit

$$\pi_i = P(\Theta = \theta_i|X = x) = \frac{P(X = x|\Theta = \theta_i)P(\Theta = \theta_i)}{P(X = x)}.$$

Wir können nun, wenn die Vorschlagswahrscheinlichkeiten q_{ij} gegeben sind, die folgenden Akzeptanzwahrscheinlichkeiten wählen:

$$\begin{aligned}
\alpha_{ij} &= \min\left\{1, \frac{\pi_j q_{ji}}{\pi_i q_{ij}}\right\} \\
&= \min\left\{1, \frac{\frac{P(X=x|\Theta=\theta_j)P(\Theta=\theta_j)}{P(X=x)} q_{ji}}{\frac{P(X=x|\Theta=\theta_i)P(\Theta=\theta_i)}{P(X=x)} q_{ij}}\right\} \\
&= \min\left\{1, \frac{P(X = x|\Theta = \theta_j)P(\Theta = \theta_j)q_{ji}}{P(X = x|\Theta = \theta_i)P(\Theta = \theta_i)q_{ij}}\right\}
\end{aligned} \tag{6.5}$$

[2]Der Fall einer Stichprobe $X_1, X_2, \ldots, X_n$ geht analog.

Um das Prinzip zu verstehen, wollen wir nun einfaches Beispiel simulieren. Ihre Stärke zeigt die MCMC-Methode allerdings erst bei komplexeren Problemen, bei denen viele Parameter zu bestimmen sind, die sogar komplizierte hierachische Abhängigkeiten haben könnnen. Auch wird man MCMC-Methoden nicht mehr selbst programmieren, sondern eines der spezielle Pakete für MCMC-Methoden verwenden, zum Beispiel `WinBUGS`, `JAGS` oder neuerdings `Stan`.

Beispiel Sie finden eine Münze, von der Sie annehmen, dass diese manipuliert ist und eher Kopf zeigt. Sie haben die Münze 4 mal geworfen und zweimal ist Kopf gefallen. Der Parameter der Verteilung Θ ist die Wahrscheinlichkeit, dass bei einem Münzwurf Kopf fällt, d.h. gegeben Θ ist die Anzahl Kopf Binomialverteilt mit $n = 4$ und Trefferwahrscheinlichkeit Θ. Aus irgendeinem seltsamen Grund[3] kann Θ nur die 101 Werte $\Theta = 0, 0.01, \ldots, 0.99, 1.00$ annehmen. Wir beschreiben dies durch eine Markov-Kette mit den Zuständen 1 bis 101 und verwenden eine symmetrische Vorschlagsmatrix Q nach folgendem Schema: Wir springen mit der Wahrscheinlichkeit von 0.5 in benachtbarte Zustände und verweilen mit der Wahrscheinlichkeit von 0.5 an den Rändern. Wegen der Symmetrie von Q ($q_{ij} = q_{ji}$) kürzt sich q_{ij} in (6.5).

Da wir annehmen, dass die Münze eher Kopf zeigt, nehmen wir eine A-Priori-Verteilung an, die das Maximum bei 0.8 hat (Abbildung 6.2 (links)):

$$P(\Theta = \theta) = \begin{cases} c_{prior} \cdot \theta & (\theta \leq 0.8) \\ c_{prior} \cdot (0.8 + (0.8 - \theta)) & (\theta > 0.8) \end{cases}.$$

Dies ist strengenommen keine Wahrscheinlichkeitsverteilung, da sie nicht auf 1 normiert ist. Da sich c_{prior} aber in (6.5) sowieso rauskürzt, müssen wir sie auch gar nicht normieren.

Die Likelihood, also die Wahrscheinlichkeit $P(2 \text{ mal Kopf bei } 4 \text{ Würfen}|\Theta_i)$ ergibt sich aus der Binomialverteilung.

Ausgehend von einem beliebigen Zustand $\Theta_i = 0.5$ simulieren wir eine Markov-Kette, die wir in der Variable `mcmc` speichern, 1 bedeutet $\Theta_i = 0.0$ und 101 $\Theta_i = 1.0$. Eine Trajektorie ist in der Mitte der Abbildung 6.2 dargestellt.

```
# Wir halten es für möglich, dass Kopf häufiger vorkommt und nehmen
# folgende A-Priori-Verteilung an
prior <- function(theta) {
  ifelse(theta < 0.8,  theta, 0.8 + (0.8-theta))
}

# p(X | Theta) = Wahrscheinlichkeit, dass man Kopf wirft
likelihood <- function(theta) dbinom(x = 2, size = 4, prob = theta)

theta <- 0.5 # Wir starten in einem beliebigen Zustand
```

[3]Der wahre Grund ist, dass wir in StoP nur diskrete Zustände betrachten. Die Verallgemeinerung auf kontinuierliche Zustände ist allerdings einfach, wenn man das Prinzip verstanden hat.

```
mcmc <- rep(0, 100000)
mcmc[1] <- 100 * theta + 1 # von 1 bis 101
pold <- prior(theta) * likelihood(theta)
for (t in 2:length(mcmc)){
  if (mcmc[t-1] == 1) { # Vorschlag eines neuen Zustandes
    mcnew <- sample(1:2,1)
  } else if (mcmc[t-1] == 101) {
    mcnew <- sample(100:101,1)
  } else {
    mcnew <- sample(c(mcmc[t-1]-1,mcmc[t-1]+1),1)
  }
  theta_new <- (mcnew - 1) / 100
  pnew <- prior(theta_new) * likelihood(theta_new)
  alpha_ij <- pnew / pold
  if (runif(1) < alpha_ij) { # Annahmew'keit min(alpha_ij, 1)
    pold <- pnew
    mcmc[t] <- mcnew
  } else{
    mcmc[t] <- mcmc[t-1]
  }
}
```

In Abb. 6.2 ist auf der rechten Seite das Histogramm der Variablen `mcmc` dargestelt, es schätzt die A-Posteriori-Verteilung $P(\Theta = \theta_i|2$ mal Kopf bei 4 Würfen). Auf Grund der A-Priori-Verteilung ist die A-Posteriori-Verteilung nicht symmetrisch, obwohl gleich viel Kopf wie Zahl gefallen ist. Der Median der A-Posteriori-Verteilung wird oft als Punktschätzer für den Parameter verwendet. Er ist 61 und entspricht somit einem Wert von $\Theta = 0.6$.

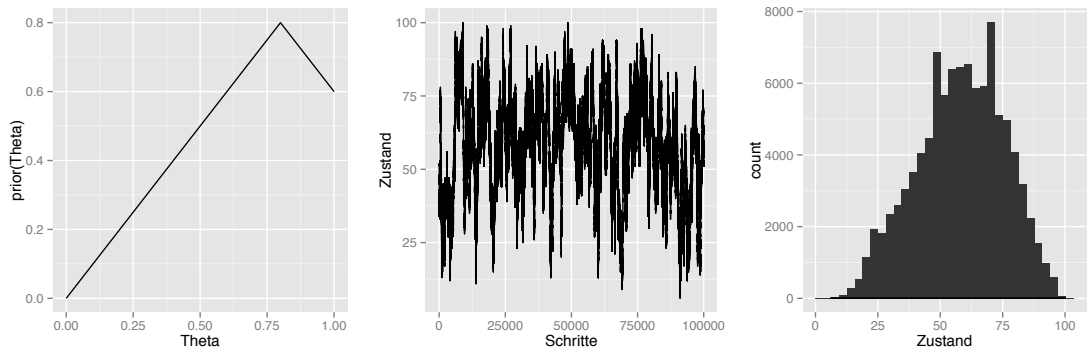

Abbildung 6.2: MCMC für das Beispiel: (Unnormierte) A-Priori-Verteilung (links), Trajektorie einer Realisation (mitte) und Histogramm der besuchten Zustände (rechts)