Hausübung 2

Philipp Lintl

16 Juni 2017

Posteriori und Full Conditionals

Es werden wie schon in Hausübung 1 die Kaiserschnitt Daten verwendet. Diesmal wird ein sogenanntes Überdispersionsmodell angenommen:

$$\begin{aligned} X_i &\sim Poi(\lambda_i) \\ log(\lambda_i) &= \eta_i \\ \eta_i &\sim N(\mu, \tau^{-1}) \\ p(\mu) &\propto const. \\ \tau &\sim Ga(a, b) \end{aligned}$$

Wie schon in der vorherigen Hausübung handelt es sich hierbei um eine hierarchische Struktur, was folgende Posteriori ergibt:

$$\begin{aligned} p(\lambda, \eta, \mu, \tau | x) &\propto p(x|\lambda) * p(\lambda) \\ &\propto p(x|\lambda) * p(\lambda|\mu, \tau) * p(\mu, \tau) \\ &\propto p(x|\lambda) * p(\lambda|\mu, \tau) * p(\mu) * p(\tau) \\ &\propto \left[\prod_{i=1}^{N} p(x_i|\lambda_i) * p(\lambda_i|\mu, \tau)\right] * p(\mu) * p(\tau) \end{aligned}$$

Einsetzen von $\lambda_i = exp(\eta_i)$ ergibt die einzelnen Teilterme aus der Angabe:
$p(x_i|\lambda_i) = \frac{exp(\eta_i)^{x_i}}{x_i!} * exp(-exp(\eta_i))$

$p(\lambda_i|\mu, \tau) = \sqrt{\frac{\tau}{2\pi}} \cdot exp\left(-\frac{\tau}{2} \cdot (\eta_i - \mu)^2\right)$

$p(\tau) = \frac{b^a}{\Gamma(a)} \cdot \tau^{a-1} \cdot exp(-b\tau)$ Diese in die obere Form von $p(\lambda, \eta, \mu, \tau|x)$ eingesetzt ergibt:

$$\begin{aligned} &\propto \left[\prod_{i=1}^{N} \frac{exp(\eta_i)^{x_i}}{x_i!} \cdot exp(-exp(\eta_i)) \cdot \sqrt{\frac{\tau}{2\pi}} \cdot exp\left(-\frac{\tau}{2} \cdot (\eta_i - \mu)^2\right)\right] \cdot p(\mu) \cdot \frac{b^a}{\Gamma(a)} \cdot \tau^{a-1} \cdot exp(-b\tau) \\ &\propto \left[\prod_{i=1}^{N} exp(\eta_i \cdot x_i) \cdot exp(-exp(\eta_i)) \cdot \tau^{\frac{1}{2}} \cdot exp\left(-\frac{\tau}{2}(\eta_i - \mu)^2\right)\right] \cdot \tau^{a-1} \cdot exp(-b\tau) \\ &\propto \left[\prod_{i=1}^{N} exp\left(\eta_i \cdot x_i - exp(\eta_i) - \frac{\tau}{2}(\eta_i - \mu)^2\right)\right] \cdot \tau^{(a+\frac{n}{2})-1} \cdot exp(-b\tau) \end{aligned}$$

Dies ist die endgültige gemeinsame Posteriori, aus der sich nun die gesuchten Full Conditionals herleiten lassen:

1)

$$\begin{aligned} p(\mu|\cdot) &\propto \prod_{i=1}^{N} exp\left(-\frac{\tau}{2}(\eta_i - \mu)^2\right) \\ &\propto exp\left(-\frac{\tau}{2}\sum_{i=1}^{N}(\eta_i - \mu)^2\right) \\ &\propto exp\left(-\frac{\tau}{2}\left[\sum_{i=1}^{N}\eta_i^2 - 2\mu\sum_{i=1}^{N}\eta_i + n\mu^2\right]\right) \\ &\propto exp\left(-\frac{\tau \cdot n}{2}\mu^2 + \mu\tau\sum_{i=1}^{N}\eta_i\right) \end{aligned}$$

Dies entspricht der kanonischen Form des Kerns einer Normalverteilung:

$$Präzision : t := \tau \cdot n; Parameterm = \tau \cdot \sum_{i=1}^{n}\eta_i$$

Daraus ergibt sich nach Skript aus dem Einführungskurs:

$$\mu|\cdot \sim N\left(\frac{m}{t}, \frac{1}{t}\right)$$

$$Mit : \frac{m}{t} = \frac{1}{n}\sum_{i=1}^{n}\eta_i; und : \frac{1}{t} = (nt)^{-1}$$

Somit ist die Full Conditional für μ wie in der Angabe gefordert, eine Normalverteilung:

$$\mu|\cdot \sim N\left(\frac{1}{n}\sum_{i=1}^{n}\eta_i, (nt)^{-1}\right)$$

2) Die Full Conditional für *tau* ergibt sich zu:

$$\begin{aligned} &\propto \exp\left(\sum_{i=1}^{n} -\frac{\tau}{2}(\eta_i - \mu)^2\right) \cdot \tau^{(a+\frac{n}{2})-1} \cdot exp(-b\tau) \\ &\propto \exp\left(\sum_{i=1}^{n} -\frac{\tau}{2}(\eta_i - \mu)^2\right) \cdot \tau^{(a+\frac{n}{2})-1} \cdot exp(-b\tau) \\ &\propto \exp\left(\sum_{i=1}^{n} -\frac{\tau}{2}(\eta_i - \mu)^2 - b\tau\right) \cdot \tau^{(a+\frac{n}{2})-1} \\ &\propto \tau^{(a+\frac{n}{2})-1} \exp\left(-\tau\left(\frac{1}{2}\sum_{i=1}^{n}(\eta_i - \mu)^2 + b\right)\right) \end{aligned}$$

Dies entspricht wie in der Angabe gefordert dem Kern einer Gammaverteilung, somit ist Gamma also folgendermaßen verteilt:

$$\tau|\cdot \sim Ga\left(a + \frac{n}{2}, b + \frac{1}{2}\sum_{i=1}^{n}(\eta_i - \mu)^2\right)$$

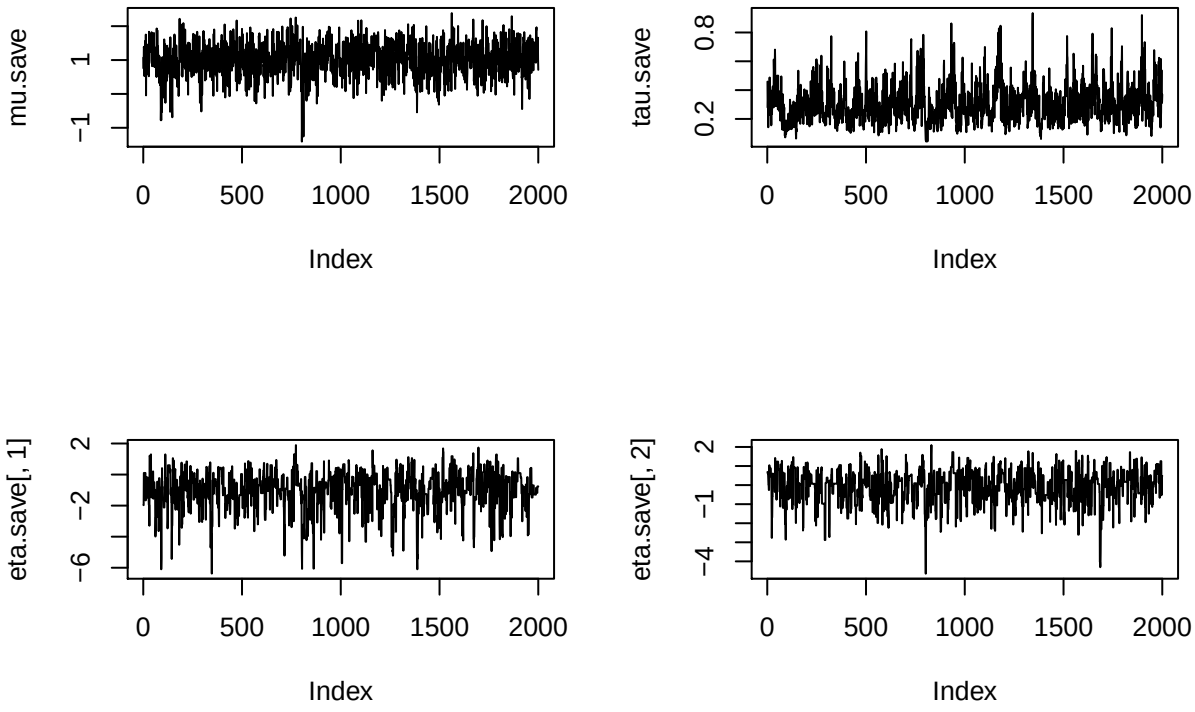
3) Nun fehlt noch die Full Conditional von η_i:

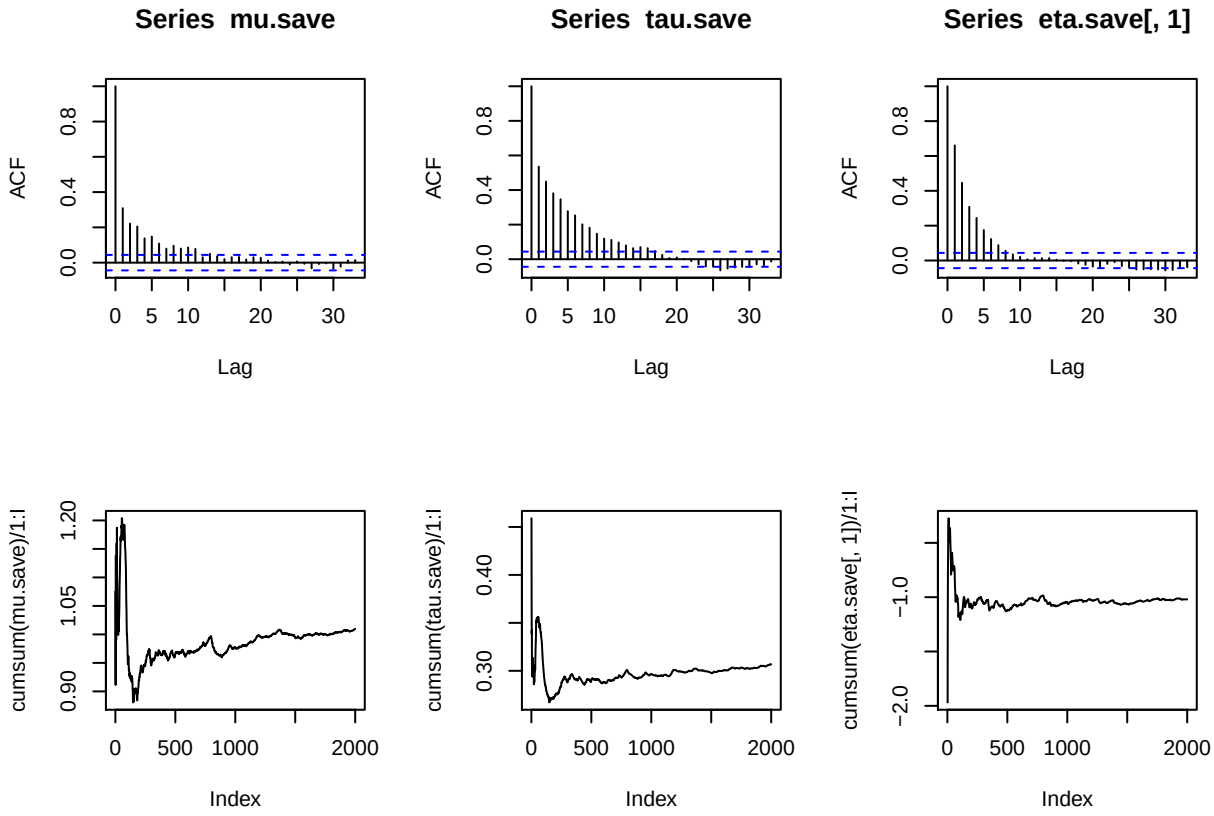
$$p(\eta_i|x_i,\mu,\tau) \propto exp\left(\eta_i \cdot x_i - exp(\eta_i) - \frac{\tau}{2}(\eta_i-\mu)^2\right)$$
$$\propto exp\left(x_i\eta_i - \frac{\tau}{2}(\eta_i-\mu)^2\right) exp(-exp(\eta_i))$$

Dies entspricht genau der Form, die in der Angabe gefordert war.

MCMC

Nun sollte aus der gemeinsamen Posteriori mit (a=1,b=0,001) gezogen werden. Es wurde wie gefordert das vorgegebene Code-Gerüst verwendet. Zum Tuning:
Anzahl Iterationen
Begonnen habe ich mit 2000 Iterationen. Außerdem war tuning=66 gesetzt und burnin auf 0, um anschließend genauer untersuchen zu können, welcher Burnin Sinn macht. Nun also der Start mit 2000 Iterationen, burnin=0, Akzeptanzfenster: 0.15-0.5 (Nach Herumprobieren darauf festgelegt).






```
## $mpsrf
## [1] 1.135897
```

Dieser Wert aus der Gelman Rubin Diagnostik zeigt an, dass 2000 Iterationen bei einem Burnin von 0 nicht genügen. Laut Angabe sollen $\mu, \tau, exp(\mu)$ geschätzt werden, also bietet sich die Raftery Diagnostik an, welche uns für ein gewünschtes Quantil (95%) den benötigten Burnin sowie die Mindestanzahl an Iterationen angibt. Als Pilotkette dient die zuvor berechnete zu 2000 Iterationen:

```
raftery.diag(data=mcmc(cbind(mu.save,tau.save, exp(mu.save))), q=0.95,r=0.01)
```

```
##
## Quantile (q) = 0.95
## Accuracy (r) = +/- 0.01
## Probability (s) = 0.95
##
##           Burn-in  Total Lower bound  Dependence
##           (M)      (N)   (Nmin)       factor (I)
##  mu.save  1        1827  1825         1.00
##  tau.save 8        4328  1825         2.37
##           1        1827  1825         1.00
```

```
# Es ergibt sich bei Toleranz r=0.01 ein N von ~ 5000. Burnin scheint sehr klein zu sein.
# Im Vergleich dazu, wenn eta ebenfalls zu schätzen wäre:
raftery.diag(data=mcmc(cbind(mu.save,tau.save,eta.save)), q=0.95,r=0.01)
```

```
##
## Quantile (q) = 0.95
## Accuracy (r) = +/- 0.01
```

```
## Probability (s) = 0.95
##
##           Burn-in  Total Lower bound  Dependence
##           (M)      (N)   (Nmin)       factor (I)
##  mu.save  1        1827  1825         1.00
##  tau.save 8        4328  1825         2.37
##           12       6281  1825         3.44
##           12       5985  1825         3.28
##           13       7081  1825         3.88
##           14       7017  1825         3.84
##           14       7335  1825         4.02
##           13       6648  1825         3.64
##           13       6859  1825         3.76
##           12       6307  1825         3.46
##           11       5807  1825         3.18
##           11       5477  1825         3.00
##           20       10582 1825         5.80
##           13       6648  1825         3.64
##           14       6934  1825         3.80
##           12       6253  1825         3.43
##           15       7701  1825         4.22
##           12       6070  1825         3.33
##           14       7563  1825         4.14
##           12       6070  1825         3.33
##           11       5566  1825         3.05
##           15       7947  1825         4.35
##           12       6224  1825         3.41
##           15       7825  1825         4.29
##           13       6551  1825         3.59
##           14       7404  1825         4.06
```

```
# Dafür wären die Anzahl benötigter Iterationen viel höher (~12000).
# Burnin jedoch auch hierfür sehr klein. Deshalb Wahl des Burnins auf 25.
```

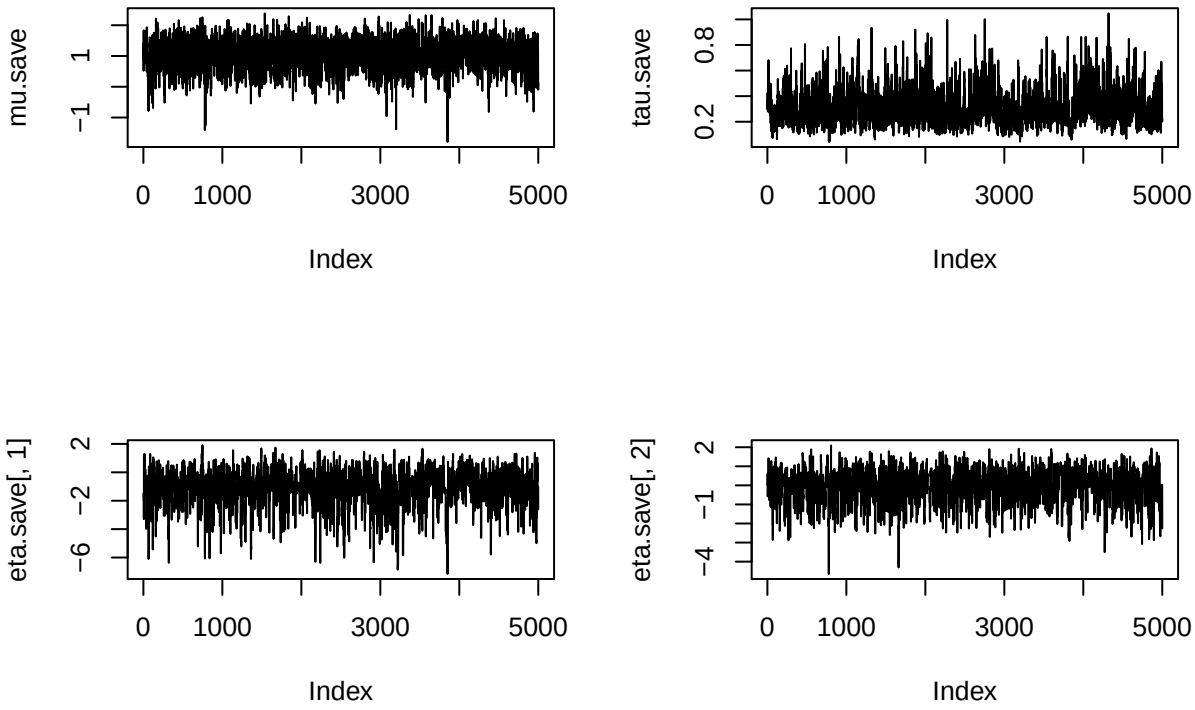
Aufgrund dieser Ergebnisse wähle ich nun 5000 Iterationen und einen Burnin von 25.

```
I <- 5000 # Anzahl Iterationen
mu.save<-rep(NA,I) # Hier wird mu gespeichert
tau.save<-rep(NA,I) # dito tau
eta.save<-array(NA,c(I,n)) # dito eta
mu <- log(mean(data)) # Startwert für  mu
tau <- 1/var(log(data+.1)) # Startwert tau (Präzision)
eta <- log(data+.1) # Startwert eta Vektor; +.1 um -Inf zu verhindern
a=1 # Priori-Parameter
b=0.001
# Berechnung der log Full Conditional von eta
log.fc.eta <- function(eta,data,tau,mu)
{
  return(data*eta-(tau/2)*(eta-mu)^2-exp(eta))
}
count <- 0                  # Zählt Tuning Durchführungen
rw.sd <- rep(.1,n)
akzeptanz <- rep(0,n)
do.tuning <- rep(TRUE,n)
tuning=66
```

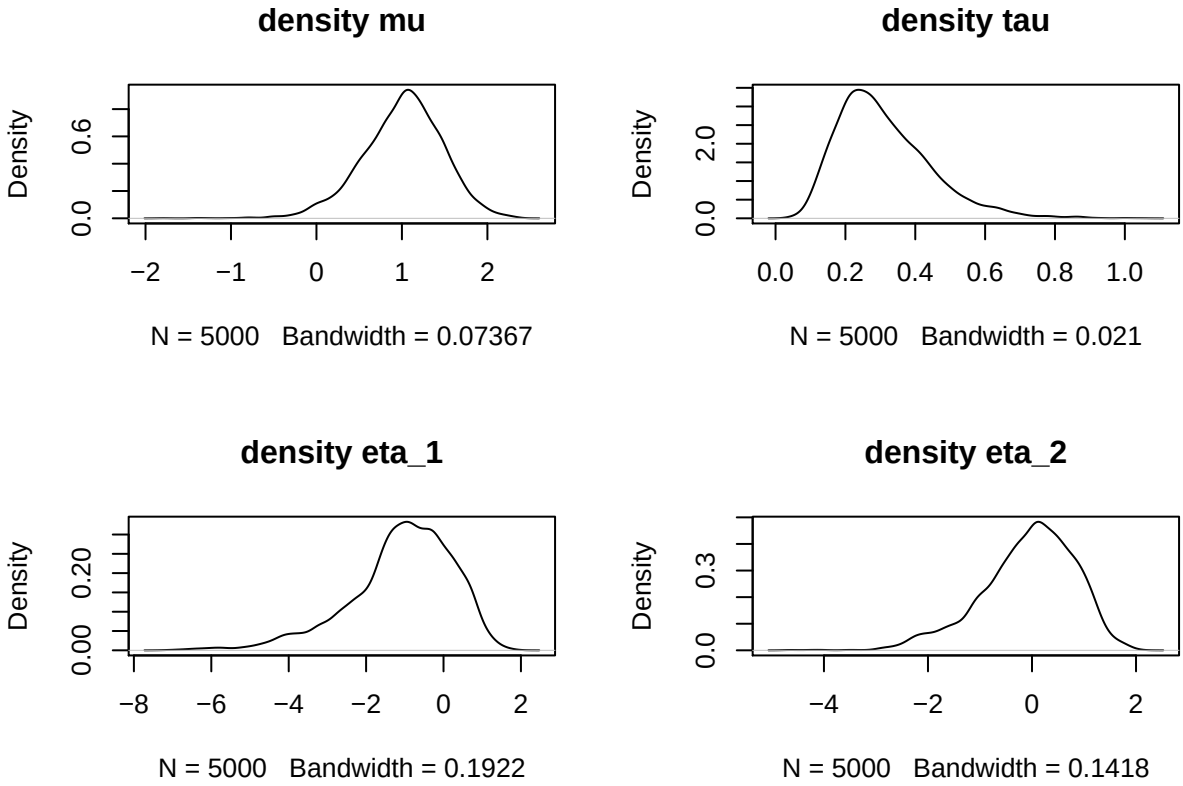
```
burnin=25
i=0
set.seed(42)
while (i<(I+burnin)) # Insgesamt I + burnin Iterationen,
                     # von denen die bis burnin nicht gespeichert werden
{
  i=i+1
  #cat("Iteration ",i,"\n")
  # Ziehe mu und tau mittels Gibbs-Steps: Also aus Full Conditionals,
  #die aus erstem Aufgabenteil bekannt sind.
  mu  <-  rnorm(1,mean=mean(eta),sd=sqrt(1/(tau*n)))
  tau <-  rgamma(1,shape=a+n/2, rate=b+(1/2)*sum((eta-mu)^2))

  # Ziehe eta's mittels random walk proposal
  etastern <- rnorm(n,eta,rw.sd)
  for (j in 1:n)
  {
    logalpha<-log.fc.eta(etastern[j],data[j],tau,mu)-log.fc.eta(eta[j],data[j],tau,mu)
    alpha<-exp(logalpha)
    if(runif(1)<alpha)
    {
      eta[j]<-etastern[j]
      akzeptanz[j]=akzeptanz[j]+1
    }
  }
  # Tuning
  # Ähnlich zum Vorlesungsbeispiel, nur multidimensional
  if(i==tuning)if(any(do.tuning==TRUE)){
    ak.rate <- akzeptanz/tuning
    # Checkt n Spalten von Eta, ob Tuning nötig ist
    # und speichert dies im do.tuning Vektor
    for(j in 1:n){
      if((ak.rate[j]>.5)||(ak.rate[j]<.15)){
        do.tuning[j]<-TRUE
      }
      else{
        do.tuning[j]<-FALSE
      }
      if(do.tuning[j])
      {
        if (ak.rate[j]>.5){
          rw.sd[j] <-rw.sd[j] * ak.rate[j] * 2
          eta.save[1,j]<-eta.save[tuning,j]
        }
        if (ak.rate[j]<.15){
          rw.sd[j] <-rw.sd[j] * ak.rate[j]
          eta.save[1,j]<-eta.save[tuning,j]
        }
      }
    }
    # Setzt Iterationen auf 0, da Tuning notwendig war.
    i=0
    # Setzt akzeptierte Ziehungen wieder auf 0, da Tuning notwendig war.
```

```
    akzeptanz <- rep(0,n)
    count <- count+1
  }
  # Nach dem Burn-in speichern wir die Ergebnisse ab
  if(i>burnin)
  {
    mu.save[i-burnin]<-mu
    tau.save[i-burnin]<-tau
    eta.save[i-burnin,]<-eta
  }
}
```

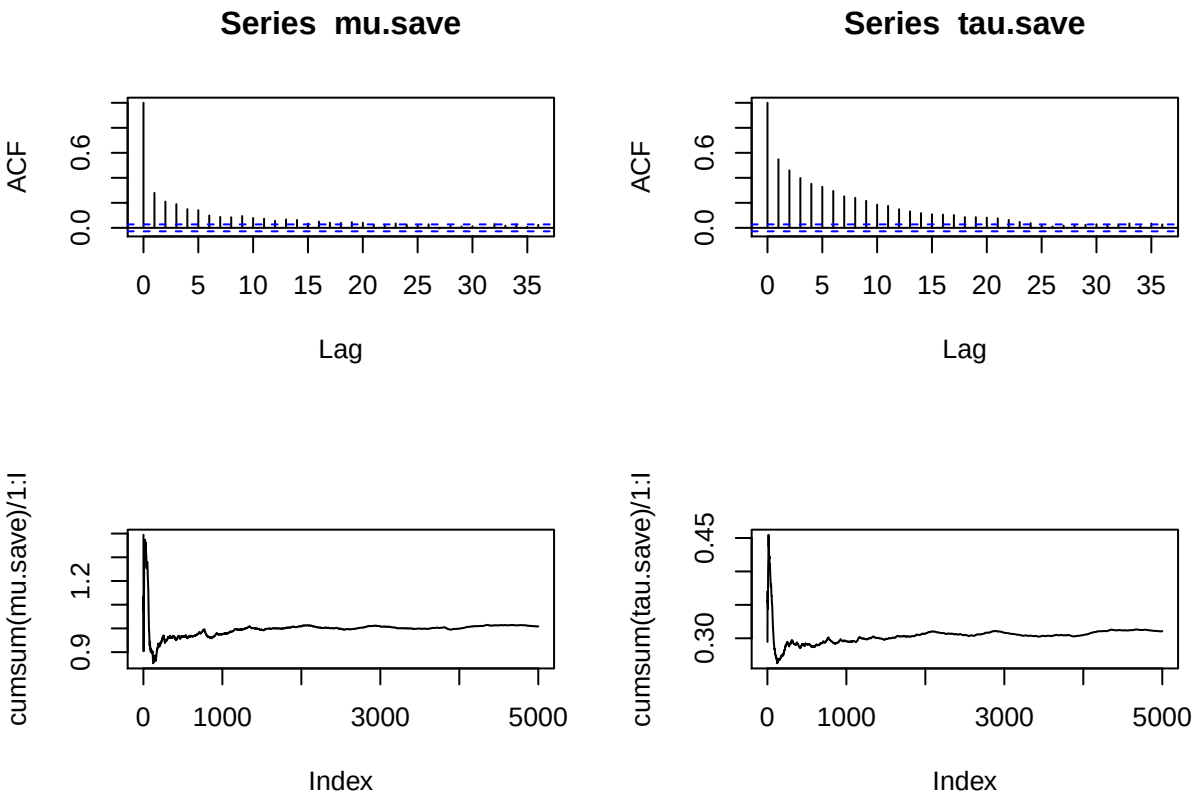
Diagnostik:
Plots von mu, tau und eta_1; eta_2 zur ersten Veranschaulichung der Ziehungen:



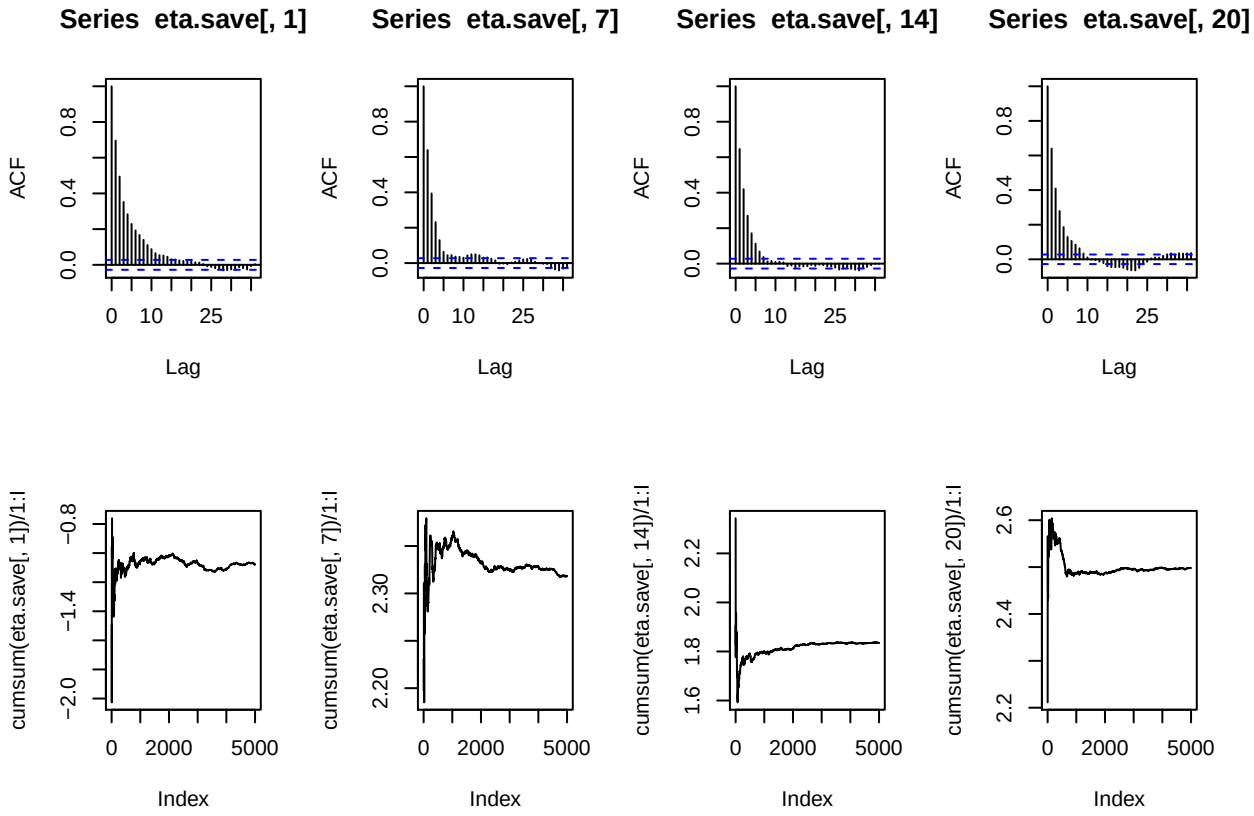
Dichten der gezogenen Parameter:



Laufende Mittelwerte von mu, tau inklusive entsprechender ACFS:



Veranschaulichung ein paar gezogener Eta´s



Diagnostik durch Gelman-Rubin: R_hat muss <1 sein:

```
chain.1 <- mcmc(cbind(mu.save[1:1000], tau.save[1:1000], eta.save[1:1000,]))
chain.2 <- mcmc(cbind(mu.save[1001:2000], tau.save[1001:2000], eta.save[1001:2000,]))
chain.3 <- mcmc(cbind(mu.save[2001:3000], tau.save[2001:3000], eta.save[2001:3000,]))
chain.4 <- mcmc(cbind(mu.save[3001:4000], tau.save[3001:4000], eta.save[3001:4000,]))
chain.5 <- mcmc(cbind(mu.save[4001:5000], tau.save[4001:5000], eta.save[4001:5000,]))
diagnostik <- gelman.diag(mcmc.list(chain.1,chain.2,chain.3,chain.4,chain.5))
diagnostik[2]
```

```
## $mpsrf
## [1] 1.068238
```

```
# Ist jetzt < 1.1
```

Man kann nun noch die Heidelberg und Welch Diagnostik zu Rateziehen, ob auch wirklich alle Ketten konvergieren:

```
heidel.diag(mcmc(cbind(mu.save,tau.save, eta.save)))
```

```
##
##          Stationarity start     p-value
##          test         iteration
## mu.save  passed          1      0.6386
## tau.save passed          1      0.3950
##          passed          1      0.8006
##          passed          1      0.5326
##          passed          1      0.2315
##          passed          1      0.8362
```

```
##          passed        501      0.2069
##          passed          1      0.5015
##          passed          1      0.1027
##          passed       1001      0.0965
##          passed          1      0.7422
##          passed          1      0.4317
##          passed          1      0.1336
##          passed       1001      0.0978
##          passed          1      0.4190
##          passed          1      0.3041
##          passed          1      0.2438
##          passed          1      0.8904
##          passed          1      0.7915
##          passed          1      0.2275
##          passed          1      0.9737
##          passed          1      0.6567
##          passed          1      0.4029
##          passed          1      0.8447
##          passed          1      0.2401
##          passed          1      0.3539
##
##          Halfwidth Mean    Halfwidth
##          test
## mu.save  passed     1.0086 0.02748
## tau.save passed     0.3106 0.01202
##          passed    -1.0773 0.09509
##          failed    -0.0669 0.05633
##          passed     2.7681 0.01435
##          passed    -1.1194 0.09009
##          passed    -1.0759 0.09180
##          passed     0.6110 0.03991
##          passed     2.3182 0.01768
##          passed     2.7629 0.01649
##          passed     3.3647 0.01039
##          passed     1.2351 0.03232
##          passed     1.2414 0.03058
##          passed     3.4281 0.01228
##          passed     1.2360 0.03144
##          passed     1.8342 0.02297
##          passed     4.4506 0.00626
##          passed    -1.1014 0.09722
##          passed    -1.1002 0.10253
##          passed    -1.0977 0.09108
##          passed     2.2081 0.01888
##          passed     2.4980 0.01713
##          passed     0.9383 0.03601
##          passed    -1.1108 0.10356
##          passed    -1.1037 0.09283
##          passed     2.0923 0.02227
```

Bis auf η_4 konvergieren alle Ketten. Nun noch zu den geforderten Schätzungen:

```
# Wegen Monte Carlo Prinzip entspricht der Mittelwert
# der Ziehungen dem Erwartungswert der Parameter
mean(mu.save)           # Erwartungswert mu
```

```
## [1] 1.008603
```

```
mean(tau.save)          # Erwartungswert tau
```

```
## [1] 0.3105695
```

```
mean(exp(mu.save))      # Erwartungswert exp(mu)
```

```
## [1] 3.04082
```

```
# 95%-Kredibilitätsintervalle
quantile(tau.save, probs=c(0.025, 0.975))       # 95% KI für tau
```

```
##      2.5%     97.5%
## 0.1166557 0.6342711
```

```
quantile(mu.save, probs=c(0.025, 0.975))        # 95% KI für mu
```

```
##        2.5%       97.5%
## 0.003314703 1.870557470
```

```
quantile(exp(mu.save), probs=c(0.025, 0.975))   # 95% KI für exp(mu)
```

```
##     2.5%    97.5%
## 1.003320 6.491915
```

STAN

```
library(rstan)
library(ggplot2)
```

```
# Vorgehen genau wie in den Vorlesungsfolien
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
# Settings für das Modell
model_poisson_ueberdisp <- "
data {
  # Anzahl an Beobachtungen
  int<lower=0> n;
  # Beobachtungswerte
  int<lower=0> x[n];
  # Prioriparameter
  real<lower=0> a;
  # Prioriparameter
  real<lower=0> b;
}
parameters {
  # eta der Laenge n, da insgesamt n etas gezogen werden
  vector[n] eta;
  # mu
  real mu;
  # tau
  real<lower=0> tau;
}
  # Modell aus erstem Aufgabenteil bekannt durch Full Conditionals:
model {
```

```
  for(i in 1:n)
  x[i] ~ poisson(exp(eta[i]));
  for(i in 1:n)
  eta[i] ~ normal(mu, sqrt(1/tau));
  tau ~ gamma(a,b);
}
"
stan_data <- list(n = length(data) , x = data, a = a, b = b)
```

Um herauszufinden, welcher Burnin nötig ist, starte ich diesen bei 4 (warmup=4; Hier komplett zufällig gewählt, jedoch aus MCMC bekannt, dass es klein ist). Burnin=0 hat viel zu hohe R_hat Werte geliefert, weshalb es insgesamt schon sinnvoll ist, einen Burnin zu verwenden.

```
model_1_b4 <- stan(model_code = model_poisson_ueberdisp, model_name = 'Poisson_Ueberdispersionsmodell',
                   data = stan_data, iter = 1000, chains = 4, warmup = 4, seed =  1)
print(model_1_b4)
```

R_hat ist bei manchen Parametern deutlich zu hoch, außerdem Warnmeldung: "divergent transitions after warmup". Deshalb erhöhe ich als nächstes den Warmup auf 10

```
model_1_b10 <- stan(model_code = model_poisson_ueberdisp, model_name = 'Poisson_Ueberdispersionsmodell'
                    data = stan_data, iter = 1000, chains = 4, warmup = 10, seed =  42)
print(model_1_b10)
```

```
## Inference for Stan model: Poisson_Ueberdispersionsmodell.
## 4 chains, each with iter=1000; warmup=10; thin=1;
## post-warmup draws per chain=990, total post-warmup draws=3960.
##
##           mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## eta[1]   -1.15    0.04 1.36  -4.30  -1.93  -0.95  -0.20   0.91  1313    1
## eta[2]   -0.06    0.01 0.91  -2.09  -0.58   0.04   0.59   1.36  3960    1
## eta[3]    2.78    0.00 0.25   2.27   2.62   2.79   2.94   3.25  3960    1
## eta[4]   -1.15    0.03 1.37  -4.60  -1.90  -0.94  -0.18   0.92  1636    1
## eta[5]   -1.17    0.03 1.32  -4.17  -1.91  -0.99  -0.22   0.87  1761    1
## eta[6]    0.55    0.01 0.70  -1.00   0.14   0.61   1.05   1.74  3960    1
## eta[7]    2.32    0.00 0.31   1.68   2.12   2.34   2.54   2.89  3960    1
## eta[8]    2.77    0.00 0.25   2.26   2.60   2.77   2.94   3.23  3960    1
## eta[9]    3.37    0.00 0.19   2.98   3.25   3.37   3.49   3.72  3859    1
## eta[10]   1.25    0.01 0.51   0.14   0.95   1.30   1.60   2.14  3960    1
## eta[11]   1.25    0.01 0.52   0.12   0.93   1.29   1.61   2.16  3960    1
## eta[12]   3.43    0.00 0.18   3.06   3.31   3.43   3.55   3.76  3960    1
## eta[13]   1.25    0.01 0.51   0.14   0.93   1.29   1.61   2.12  3960    1
## eta[14]   1.84    0.01 0.39   0.98   1.60   1.87   2.10   2.55  3960    1
## eta[15]   4.45    0.00 0.11   4.24   4.38   4.45   4.52   4.65  3566    1
## eta[16]  -1.21    0.04 1.46  -4.48  -1.99  -0.99  -0.21   0.90  1162    1
## eta[17]  -1.18    0.03 1.34  -4.23  -1.90  -0.99  -0.23   0.90  1577    1
## eta[18]  -1.18    0.03 1.39  -4.53  -1.95  -0.96  -0.20   0.91  1589    1
## eta[19]   2.21    0.01 0.32   1.57   2.00   2.23   2.44   2.79  3960    1
## eta[20]   2.49    0.00 0.29   1.88   2.30   2.50   2.69   3.00  3960    1
## eta[21]   0.94    0.01 0.59  -0.39   0.58   1.00   1.36   1.91  3960    1
## eta[22]  -1.16    0.03 1.33  -4.21  -1.88  -0.97  -0.22   0.88  1471    1
## eta[23]  -1.17    0.04 1.42  -4.47  -1.92  -0.96  -0.19   0.93  1326    1
## eta[24]   2.10    0.01 0.34   1.37   1.89   2.12   2.34   2.71  3960    1
## mu        0.98    0.01 0.49  -0.04   0.70   1.01   1.31   1.85  2068    1
## tau       0.30    0.00 0.13   0.11   0.21   0.28   0.37   0.63  1138    1
```

```
## lp__    541.74    0.14 4.23 532.19 539.11 542.20 544.74 548.80   935    1
##
## Samples were drawn using NUTS(diag_e) at Sun Jun 18 10:20:23 2017.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Diesmal ist jedes R_hat=1, also konvergiert jede Kette. Außerdem keine Warnmeldungen, deshalb verwende ich ab jetzt burnin von 10. Aus Interesse habe ich das ganze für einen deutlich höheren Burnin versucht:

```
model_1_b100 <- stan(model_code = model_poisson_ueberdisp, model_name = 'Poisson_Ueberdispersionsmodell
                     data = stan_data, iter = 1000, chains = 4, warmup = 100, seed =100)
# print(model_1_b100)
```

An der Konvergenz hat dies Allerdings nicht viel verändert, was andeutet, dass ein burnin von 10 anscheinend wirklich ausreicht.

```
coda::gelman.diag(As.mcmc.list(model_1_b10))[2]
```

```
## $mpsrf
## [1] 1.01669
```

Rubin Gelman Diagnostik bestätigt dies auch, also Multivariate psrf=1.02 und damit < 1.1.
In der Angabe war gefragt, wie viele Iterationen man "braucht", deshalb noch ein paar Tests des Modells mit niedrigeren Iterationswerten und Analyse, ob diese ausreichen.

```
model_2_b10 <- stan(model_code = model_poisson_ueberdisp, model_name = 'Poisson_Ueberdispersionsmodell'
                    data = stan_data, iter = 750, chains = 4, warmup = 10, seed =  42)
# print(model_2_b10)
model_3_b10 <- stan(model_code = model_poisson_ueberdisp, model_name = 'Poisson_Ueberdispersionsmodell'
                    data = stan_data, iter = 500, chains = 4, warmup = 10, seed =  42)
# print(model_3_b10)
model_4_b10 <- stan(model_code = model_poisson_ueberdisp, model_name = 'Poisson_Ueberdispersionsmodell'
                    data = stan_data, iter = 250, chains = 4, warmup = 10, seed =  42)
# print(model_4_b10)
coda::gelman.diag(As.mcmc.list(model_4_b10))[2]
```

```
## $mpsrf
## [1] 1.076713
```

Bei allen sind die R_hat Werte in Ordnung und sogar für einen Iterationswert von 250 deutet die Rubin Gelman Diagnostik noch auf Konvergenz hin. Ein weiterer Test für 150 Iterationen pro Kette deutet jedoch an, dass nicht viel weniger verwendet werden sollten.

```
model_5_b10 <- stan(model_code = model_poisson_ueberdisp, model_name = 'Poisson_Ueberdispersionsmodell'
                    data = stan_data, iter = 150, chains = 4, warmup = 10, seed =  42)
# print(model_5_b10)
coda::gelman.diag(As.mcmc.list(model_5_b10))[2]
```

```
## $mpsrf
## [1] 1.180742
```

Erstmals R_hat zu groß, deshalb iter=250 als ausreichend verwenden. Nun zu den geforderten Schätzungen, welche direkt bis auf exp(mu) aus dem Output auslesbar sind:

```
model_4_b10
```

```
## Inference for Stan model: Poisson_Ueberdispersionsmodell.
## 4 chains, each with iter=250; warmup=10; thin=1;
## post-warmup draws per chain=240, total post-warmup draws=960.
```

```
##
##           mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## eta[1]   -1.14    0.07 1.33  -4.01  -1.91  -0.90  -0.24   0.90   359 1.01
## eta[2]   -0.11    0.05 0.98  -2.36  -0.64   0.01   0.58   1.32   456 1.01
## eta[3]    2.78    0.01 0.24   2.27   2.62   2.80   2.95   3.23   934 1.00
## eta[4]   -1.18    0.07 1.35  -4.62  -1.93  -1.02  -0.21   0.94   420 1.01
## eta[5]   -1.21    0.07 1.35  -4.38  -2.00  -1.02  -0.24   0.86   394 1.00
## eta[6]    0.53    0.02 0.74  -1.08   0.11   0.58   1.05   1.79   960 1.00
## eta[7]    2.31    0.01 0.32   1.63   2.11   2.32   2.54   2.85   960 1.00
## eta[8]    2.78    0.01 0.25   2.26   2.63   2.79   2.95   3.26   960 1.00
## eta[9]    3.37    0.01 0.18   3.00   3.26   3.37   3.48   3.68   899 1.00
## eta[10]   1.28    0.02 0.49   0.20   1.00   1.31   1.61   2.16   960 1.00
## eta[11]   1.23    0.02 0.52   0.15   0.91   1.28   1.57   2.13   960 1.00
## eta[12]   3.43    0.01 0.18   3.07   3.32   3.44   3.55   3.76   960 1.00
## eta[13]   1.25    0.02 0.50   0.10   0.92   1.28   1.61   2.13   960 1.00
## eta[14]   1.86    0.01 0.37   1.07   1.64   1.89   2.10   2.54   960 1.00
## eta[15]   4.45    0.00 0.11   4.24   4.38   4.45   4.52   4.63   907 1.00
## eta[16]  -1.25    0.07 1.38  -4.33  -2.08  -1.08  -0.30   0.88   375 1.01
## eta[17]  -1.25    0.07 1.41  -4.35  -2.06  -1.02  -0.22   0.82   374 1.01
## eta[18]  -1.19    0.08 1.42  -4.69  -1.86  -0.99  -0.16   0.85   354 1.01
## eta[19]   2.21    0.01 0.34   1.49   2.00   2.23   2.44   2.82   960 1.00
## eta[20]   2.49    0.01 0.27   1.91   2.32   2.51   2.68   3.00   960 1.00
## eta[21]   0.94    0.02 0.58  -0.38   0.57   1.00   1.35   1.89   960 1.00
## eta[22]  -1.23    0.07 1.37  -4.32  -1.93  -1.07  -0.24   0.86   365 1.01
## eta[23]  -1.20    0.08 1.41  -4.67  -1.94  -0.92  -0.24   0.92   283 1.01
## eta[24]   2.09    0.01 0.36   1.36   1.86   2.11   2.34   2.71   960 1.00
## mu        0.96    0.02 0.49  -0.06   0.67   0.99   1.29   1.81   508 1.00
## tau       0.30    0.01 0.13   0.10   0.21   0.28   0.36   0.63   220 1.02
## lp__    541.59    0.30 4.33 531.63 538.98 542.26 544.66 548.60   206 1.02
##
## Samples were drawn using NUTS(diag_e) at Sun Jun 18 10:20:45 2017.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
# Um auf exp(mu) zu kommen benötigt man alle gezogenen Werte:
samples_1 <- extract(model_4_b10, permuted = TRUE)
mean(exp(samples_1$mu))
```

```
## [1] 2.904947
```

```
quantile(exp(samples_1$mu),  probs = c(0.025, 0.975))
```

```
##      2.5%     97.5%
## 0.9459252 6.1001007
```

Vergleicht man diese mit den Werten aus dem MCMC Teil, liegen diese in einem ähnlichen Bereich. Da allerdings im MCMC insgesamt 5000 Iterationen verwendet wurden will ich noch einmal sehen, ob vielleicht das STAN-Modell mir iter=1000 den Schätzungen aus dem MCMC Teil näher kommt:

```
model_1_b10 <- stan(model_code = model_poisson_ueberdisp, model_name = 'Poisson_Ueberdispersionsmodell'
                    data = stan_data, iter = 1000, chains = 4, warmup = 10, seed =  42)
print(model_1_b10)
```

```
## Inference for Stan model: Poisson_Ueberdispersionsmodell.
## 4 chains, each with iter=1000; warmup=10; thin=1;
## post-warmup draws per chain=990, total post-warmup draws=3960.
```

```
##
##           mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## eta[1]   -1.15    0.04 1.36  -4.30  -1.93  -0.95  -0.20   0.91  1313    1
## eta[2]   -0.06    0.01 0.91  -2.09  -0.58   0.04   0.59   1.36  3960    1
## eta[3]    2.78    0.00 0.25   2.27   2.62   2.79   2.94   3.25  3960    1
## eta[4]   -1.15    0.03 1.37  -4.60  -1.90  -0.94  -0.18   0.92  1636    1
## eta[5]   -1.17    0.03 1.32  -4.17  -1.91  -0.99  -0.22   0.87  1761    1
## eta[6]    0.55    0.01 0.70  -1.00   0.14   0.61   1.05   1.74  3960    1
## eta[7]    2.32    0.00 0.31   1.68   2.12   2.34   2.54   2.89  3960    1
## eta[8]    2.77    0.00 0.25   2.26   2.60   2.77   2.94   3.23  3960    1
## eta[9]    3.37    0.00 0.19   2.98   3.25   3.37   3.49   3.72  3859    1
## eta[10]   1.25    0.01 0.51   0.14   0.95   1.30   1.60   2.14  3960    1
## eta[11]   1.25    0.01 0.52   0.12   0.93   1.29   1.61   2.16  3960    1
## eta[12]   3.43    0.00 0.18   3.06   3.31   3.43   3.55   3.76  3960    1
## eta[13]   1.25    0.01 0.51   0.14   0.93   1.29   1.61   2.12  3960    1
## eta[14]   1.84    0.01 0.39   0.98   1.60   1.87   2.10   2.55  3960    1
## eta[15]   4.45    0.00 0.11   4.24   4.38   4.45   4.52   4.65  3566    1
## eta[16]  -1.21    0.04 1.46  -4.48  -1.99  -0.99  -0.21   0.90  1162    1
## eta[17]  -1.18    0.03 1.34  -4.23  -1.90  -0.99  -0.23   0.90  1577    1
## eta[18]  -1.18    0.03 1.39  -4.53  -1.95  -0.96  -0.20   0.91  1589    1
## eta[19]   2.21    0.01 0.32   1.57   2.00   2.23   2.44   2.79  3960    1
## eta[20]   2.49    0.00 0.29   1.88   2.30   2.50   2.69   3.00  3960    1
## eta[21]   0.94    0.01 0.59  -0.39   0.58   1.00   1.36   1.91  3960    1
## eta[22]  -1.16    0.03 1.33  -4.21  -1.88  -0.97  -0.22   0.88  1471    1
## eta[23]  -1.17    0.04 1.42  -4.47  -1.92  -0.96  -0.19   0.93  1326    1
## eta[24]   2.10    0.01 0.34   1.37   1.89   2.12   2.34   2.71  3960    1
## mu        0.98    0.01 0.49  -0.04   0.70   1.01   1.31   1.85  2068    1
## tau       0.30    0.00 0.13   0.11   0.21   0.28   0.37   0.63  1138    1
## lp__    541.74    0.14 4.23 532.19 539.11 542.20 544.74 548.80   935    1
##
## Samples were drawn using NUTS(diag_e) at Sun Jun 18 10:20:55 2017.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
samples_2 <- extract(model_1_b10, permuted = TRUE)
mean(exp(samples_2$mu))
```

```
## [1] 2.978569
```

```
quantile(exp(samples_2$mu),  probs = c(0.025, 0.975))
```

```
##      2.5%     97.5%
## 0.9590271 6.3768465
```

Das bringt die Schätzungen aus dem STAN Modell dem aus dem MCMC Teil näher. Da wir aber dort vielleicht auch weniger Iterationen 'gebraucht' hätten, kann iter=250 schon verwendet werden. Wie in den Vorlesungsfolien könnte man auch noch die Ketten visualisieren (ACFS,Density), worauf ich aber aus Platzgründen verzichten will.