Project Model

Lynn Fan

4/26/2020

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(dplyr)
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 3.0-2
```

```
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following object is masked from 'package:purrr':
##
##     some
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
library(bayesm)
library(ggplot2)
library(R2admb)
library(glmmADMB)
```

```
##
## Attaching package: 'glmmADMB'
```

```
## The following object is masked from 'package:MASS':
##
##     stepAIC
```

```
## The following object is masked from 'package:stats':
##
##     step
```

```
library(lme4)
```

```
## Registered S3 methods overwritten by 'lme4':
##   method                          from
##   cooks.distance.influence.merMod car
##   influence.merMod                car
##   dfbeta.influence.merMod         car
##   dfbetas.influence.merMod        car
```

```
##
## Attaching package: 'lme4'
```

```
## The following object is masked from 'package:glmmADMB':
##
##     VarCorr
```

```
library(tidyr)
library(mcmc)
library(dplyr)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following objects are masked from 'package:data.table':
##
##     dcast, melt
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library(bayesplot)
```

```
## This is bayesplot version 1.7.1
```

```
## - Online documentation and vignettes at mc-stan.org/bayesplot
```

```
## - bayesplot theme set to bayesplot::theme_default()
##    * Does _not_ affect other ggplot2 plots
##    * See ?bayesplot_theme_set for details on theme setting
```

```
library(varhandle)
library(loo)
```

```
## This is loo version 2.2.0
## - Online documentation and vignettes at mc-stan.org/loo
## - As of v2.0.0 loo defaults to 1 core but we recommend using as many as possible. Use the 'cores' ar
```

Data Cleanup

```
data <- read.table("rawdata.txt",
               col.names=c('stops', 'pop', 'past.arrests', 'precinct', 'eth', 'crime'),
               fill=FALSE,
               strip.white=TRUE)
```

Exploratory Data Analysis

```
r <- c(mean(data$stops), var(data$stops))
c(mean=r[1], var=r[2], ratio=r[2]/r[1])
```

```
##       mean        var      ratio
##   146.0222 47254.9317   323.6147
```

Overdispersed, so we should do Negative Binomial instead of Poisson.

```
png('ran_effect.png')
data %>%
  group_by(precinct) %>%
  ggplot(., mapping = aes(x = as.factor(precinct), y = stops)) +
    geom_boxplot() + theme(axis.text.x = element_text(angle = 90, hjust=1)) +
  labs(title="number of stops by precincts", x="precincts")
dev.off()
```

```
## pdf
##   2
```

```
stops<-data$stops ; ethi<-as.factor(data$eth) ; precinct<-as.factor(data$precinct);arrest=data$past.arr
overdisp_fun <- function(model) {
    rdf <- df.residual(model)
    rp <- residuals(model,type="pearson")
    Pearson.chisq <- sum(rp^2)
    prat <- Pearson.chisq/rdf
    pval <- pchisq(Pearson.chisq, df=rdf, lower.tail=FALSE)
    c(chisq=Pearson.chisq,ratio=prat,rdf=rdf,p=pval)
}
# Poisson with random effects
fit.poi <- glmer(stops~1+ethi+(1|precinct),family = poisson(link = "log"), nAGQ = 100)
summary(fit.poi)
```

```
## Generalized linear mixed model fit by maximum likelihood (Adaptive
##   Gauss-Hermite Quadrature, nAGQ = 100) [glmerMod]
```

```
##  Family: poisson  ( log )
## Formula: stops ~ 1 + ethi + (1 | precinct)
##
##      AIC      BIC   logLik deviance df.resid
## 113922.4 113941.6 -56957.2 113914.4      896
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -20.035  -7.453  -3.245   3.249  77.185
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  precinct (Intercept) 0.3368   0.5803
## Number of obs: 900, groups:  precinct, 75
##
## Fixed effects:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.301655   0.067142   78.96   <2e-16 ***
## ethi2       -0.447714   0.006061  -73.87   <2e-16 ***
## ethi3       -1.414280   0.008558 -165.26   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##       (Intr) ethi2
## ethi2 -0.035
## ethi3 -0.025  0.276
```

```
overdisp_fun(fit.poi)
```

```
##       chisq       ratio         rdf           p
## 144223.2372    160.9634    896.0000      0.0000
```

```
# Negative Binomial
fit.nb <- glmer.nb(stops~1+ethi+(1|precinct), verbose=TRUE)
```

```
## theta.ml: iter 0 'theta = 0.610130'

## theta.ml: iter1 theta =0.810456

## theta.ml: iter2 theta =0.914262

## theta.ml: iter3 theta =0.931802

## theta.ml: iter4 theta =0.932198

## theta.ml: iter5 theta =0.932198

## th := est_theta(glmer(..)) = 0.9321982 --> dev.= -2*logLik(.) = 10427.73

## Warning in glmer.nb(stops ~ 1 + ethi + (1 | precinct), verbose = TRUE): no
## 'data = *' in glmer.nb() call ... Not much is guaranteed

##  1: th=   0.4591337260, dev=10641.90596216, beta[1]=    5.42176950
##  2: th=    1.892680533, dev=10727.25392378, beta[1]=    5.41299236

## boundary (singular) fit: see ?isSingular

##  3: th=   0.1913211266, dev=11357.19917246, beta[1]=    5.44992947
##  4: th=   0.8616480080, dev=10430.74019060, beta[1]=    5.40699784
```

```
##  5: th=   0.8779870295, dev=10429.47790274, beta[1]=    5.40706944
##  6: th=   0.9433502167, dev=10427.80377650, beta[1]=    5.40742909
##  7: th=    1.230782015, dev=10469.02080218, beta[1]=    5.40958304
##  8: th=   0.9322493059, dev=10427.72947553, beta[1]=    5.40736521
##  9: th=   0.9318378247, dev=10427.72942708, beta[1]=    5.40735768
## 10: th=   0.9319417111, dev=10427.72942344, beta[1]=    5.40736519
## 11: th=   0.9319250090, dev=10427.72942319, beta[1]=    5.40736519
## 12: th=   0.9319094760, dev=10427.72942324, beta[1]=    5.40736012
## 13: th=   0.9319250090, dev=10427.72942319, beta[1]=    5.40736372
```

```
summary(fit.nb)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: Negative Binomial(0.9319)  ( log )
## Formula: stops ~ 1 + ethi + (1 | precinct)
##
##      AIC      BIC   logLik deviance df.resid
##  10437.7  10461.7  -5213.9  10427.7      895
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -0.9623 -0.6824 -0.3317  0.2750  6.7019
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  precinct (Intercept) 0.1942   0.4407
## Number of obs: 900, groups:  precinct, 75
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.40736    0.08201  65.935  < 2e-16 ***
## ethi2       -0.56572    0.09288  -6.091 1.12e-09 ***
## ethi3       -1.52446    0.09599 -15.881  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##       (Intr) ethi2
## ethi2 -0.568
## ethi3 -0.575  0.527
```

```
overdisp_fun(fit.nb)
```

```
##        chisq        ratio          rdf            p
## 1.047986e+03 1.170934e+00 8.950000e+02 2.867296e-04
```

```
n <- nrow(data)
precinct.number <- unique(data$precinct)
n.precinct <- length(precinct.number)
precincts <- rep(NA,n)
pblack <- rep(NA,n.precinct)
for (i in 1:n.precinct) {
  temp <- data[data$precinct==i,]
  blackpop <- temp[temp$eth==1,]$pop[1]
  totalpop <- temp[temp$eth==1,]$pop[1]+temp[temp$eth==2,]$pop[1]+temp[temp$eth==3,]$pop[1]
```

```
  pblack[i]<-blackpop/totalpop
}
precinct.category <- ifelse (pblack < .1, 1, ifelse (pblack < .4, 2, 3))
arrests <- data$past.arrests
dcjs <- log(arrests*15/12)
dcjs[which(!is.finite(dcjs))] <- 0
crime <- data$crime
pop <- data$pop
stop_df <- as.data.frame (cbind(stops, ethi, precinct, crime, precinct.category, arrests, dcjs))
stop_df$ethi <- as.factor(ethi)
# Multilevel analysis of NYC police stops

# lmer() fits
M1 <- as.list (rep (NA, 12))
M2 <- as.list (rep (NA, 12))
index <- 0
for (j in 1:3){
  for (k in 1:4){
    index <- index + 1
    ok <- precinct.category==j & crime==k
    M1[[index]] <- glmer(stops~1+dcjs+ethi+(1|precinct), #Poisson with random effect
     family=poisson(link="log"), subset=ok, data=stop_df)
    #Negative Binomial
    M2[[index]] <- glmer.nb(stops~1+dcjs+ethi+(1|precinct), verbose=TRUE,subset=ok,data=stop_df,nAGQ=0)
  }
}
```

```
## theta.ml: iter 0 'theta = 16.005300'
## theta.ml: iter1 theta =28.0078
## theta.ml: iter2 theta =47.4186
## theta.ml: iter3 theta =77.8122
## theta.ml: iter4 theta =124.182
## theta.ml: iter5 theta =193.545
## theta.ml: iter6 theta =296.19
## theta.ml: iter7 theta =448.003
## theta.ml: iter8 theta =674.091
## theta.ml: iter9 theta =1013.84
## theta.ml: iter10 theta =1527.57
## theta.ml: iter11 theta =2305.62
## theta.ml: iter12 theta =3481.94
## theta.ml: iter13 theta =5255.63
## theta.ml: iter14 theta =7924.1
## theta.ml: iter15 theta =11933
## theta.ml: iter16 theta =17951
## theta.ml: iter17 theta =26981.3
```

```
## theta.ml: iter18 theta =40529

## theta.ml: iter19 theta =60852.1

## Warning in theta.ml(Y, mu, weights = object@resp$weights, limit = limit, :
## iteration limit reached

## th := est_theta(glmer(..)) = 60852.13 --> dev.= -2*logLik(.) = 682.4184
##  1: th=    29971.37919, dev=  682.34334344, beta[1]=   -1.11567969
##  2: th=    123550.5970, dev=  682.45547003, beta[1]=   -1.11659032
##  3: th=    12489.08043, dev=  682.13828787, beta[1]=   -1.11400540
##  4: th=    7270.572483, dev=  681.88975432, beta[1]=   -1.11195732
##  5: th=    5204.202615, dev=  681.65736979, beta[1]=   -1.11002498
##  6: th=    4232.595389, dev=  681.47230657, beta[1]=   -1.10847403
##  7: th=    3725.115859, dev=  681.33872254, beta[1]=   -1.10734785
##  8: th=    3442.383982, dev=  681.24791369, beta[1]=   -1.10657913
##  9: th=    3278.482087, dev=  681.18840690, beta[1]=   -1.10607408
## 10: th=    3181.111120, dev=  681.15027960, beta[1]=   -1.10574983
## 11: th=    3122.384037, dev=  681.12618599, beta[1]=   -1.10554475
## 12: th=    3086.632072, dev=  681.11108966, beta[1]=   -1.10541618
## 13: th=    3064.741125, dev=  681.10168025, beta[1]=   -1.10533587
## 14: th=    3051.289474, dev=  681.09583443, beta[1]=   -1.10528614
## 15: th=    3043.005434, dev=  681.09220981, beta[1]=   -1.10525527
## 16: th=    3037.896864, dev=  681.08996519, beta[1]=   -1.10523601
## 17: th=    3034.743884, dev=  681.08857623, beta[1]=   -1.10522416
## 18: th=    3032.796871, dev=  681.08771715, beta[1]=   -1.10521696
## 19: th=    3031.594176, dev=  681.08718596, beta[1]=   -1.10521213
## 20: th=    3030.851108, dev=  681.08685757, beta[1]=   -1.10520955
## 21: th=    3030.391957, dev=  681.08665458, beta[1]=   -1.10520787
## 22: th=    3030.108222, dev=  681.08652911, beta[1]=   -1.10520679
## 23: th=    3029.932877, dev=  681.08645156, beta[1]=   -1.10520608
## 24: th=    3029.824513, dev=  681.08640363, beta[1]=   -1.10520562
## 25: th=    3029.757542, dev=  681.08637401, beta[1]=   -1.10520548
## 26: th=    3029.706684, dev=  681.08635151, beta[1]=   -1.10520530
## 27: th=    3029.706684, dev=  681.08635151, beta[1]=   -1.10520530

## theta.ml: iter 0 'theta = 6.681820'

## theta.ml: iter1 theta =11.5403

## theta.ml: iter2 theta =19.4024

## theta.ml: iter3 theta =31.716

## theta.ml: iter4 theta =50.4175

## theta.ml: iter5 theta =78.2957

## theta.ml: iter6 theta =120.251

## theta.ml: iter7 theta =187.13

## theta.ml: iter8 theta =306.843

## theta.ml: iter9 theta =546.648

## theta.ml: iter10 theta =996.849

## theta.ml: iter11 theta =1718.91

## theta.ml: iter12 theta =2810.2
```

```
## theta.ml: iter13 theta =4440.42

## theta.ml: iter14 theta =6873.08

## theta.ml: iter15 theta =10507.7

## theta.ml: iter16 theta =15945.9

## theta.ml: iter17 theta =24091.6

## theta.ml: iter18 theta =36300.9

## theta.ml: iter19 theta =54608

## Warning in theta.ml(Y, mu, weights = object@resp$weights, limit = limit, :
## iteration limit reached

## th := est_theta(glmer(..)) = 54608.02 --> dev.= -2*logLik(.) = 857.4893
##  1: th=    26895.98211, dev=  857.25633627, beta[1]=    0.29984549
##  2: th=    110872.9306, dev=  857.60426731, beta[1]=    0.29849855
##  3: th=    11207.56177, dev=  856.61937263, beta[1]=    0.30234592
##  4: th=    6524.530825, dev=  855.84625294, beta[1]=    0.30543582
##  5: th=    4670.193504, dev=  855.12234114, beta[1]=    0.30838083
##  6: th=    3798.283993, dev=  854.54516613, beta[1]=    0.31076280
##  7: th=    3342.877511, dev=  854.12818438, beta[1]=    0.31250154
##  8: th=    3089.157072, dev=  853.84455638, beta[1]=    0.31369225
##  9: th=    2942.073335, dev=  853.65862257, beta[1]=    0.31447655
## 10: th=    2854.693713, dev=  853.53946054, beta[1]=    0.31498056
## 11: th=    2801.992682, dev=  853.46414691, beta[1]=    0.31529966
## 12: th=    2769.909267, dev=  853.41695289, beta[1]=    0.31549986
## 13: th=    2750.264575, dev=  853.38753545, beta[1]=    0.31562475
## 14: th=    2738.193213, dev=  853.36925840, beta[1]=    0.31570234
## 15: th=    2730.759207, dev=  853.35792572, beta[1]=    0.31575052
## 16: th=    2726.174834, dev=  853.35090763, beta[1]=    0.31578030
## 17: th=    2723.345385, dev=  853.34656482, beta[1]=    0.31579885
## 18: th=    2721.598157, dev=  853.34387875, beta[1]=    0.31581022
## 19: th=    2720.518872, dev=  853.34221788, beta[1]=    0.31581725
## 20: th=    2719.852051, dev=  853.34119110, beta[1]=    0.31582167
## 21: th=    2719.440014, dev=  853.34055641, beta[1]=    0.31582428
## 22: th=    2719.185393, dev=  853.34016410, beta[1]=    0.31582595
## 23: th=    2719.028040, dev=  853.33992162, beta[1]=    0.31582692
## 24: th=    2718.930796, dev=  853.33977176, beta[1]=    0.31582765
## 25: th=    2718.870697, dev=  853.33967913, beta[1]=    0.31582800
## 26: th=    2718.825062, dev=  853.33960880, beta[1]=    0.31582826
## 27: th=    2718.825062, dev=  853.33960880, beta[1]=    0.31582836

## theta.ml: iter 0 'theta = 0.220793'

## theta.ml: iter1 theta =0.40472

## theta.ml: iter2 theta =0.734929

## theta.ml: iter3 theta =1.33002

## theta.ml: iter4 theta =2.40162

## theta.ml: iter5 theta =4.301

## theta.ml: iter6 theta =7.55052

## theta.ml: iter7 theta =12.8182
```

```
## theta.ml: iter8 theta =20.8427

## theta.ml: iter9 theta =32.4058

## theta.ml: iter10 theta =48.3966

## theta.ml: iter11 theta =69.8888

## theta.ml: iter12 theta =98.1217

## theta.ml: iter13 theta =134.231

## theta.ml: iter14 theta =178.474

## theta.ml: iter15 theta =228.683

## theta.ml: iter16 theta =278.121

## theta.ml: iter17 theta =315.042

## theta.ml: iter18 theta =330.74

## theta.ml: iter19 theta =332.918

## Warning in theta.ml(Y, mu, weights = object@resp$weights, limit = limit, :
## iteration limit reached

## th := est_theta(glmer(..)) = 332.9184 --> dev.= -2*logLik(.) = 804.8207
##  1: th=    163.9716327, dev=  793.66688602, beta[1]=   -1.36057114
##  2: th=    675.9379666, dev=  811.88139027, beta[1]=   -1.48091290
##  3: th=    68.32701610, dev=  774.65129909, beta[1]=   -1.22302681
##  4: th=    39.77686956, dev=  761.83626636, beta[1]=   -1.10736785
##  5: th=    28.47188293, dev=  754.40677891, beta[1]=   -1.02119653
##  6: th=    23.15627759, dev=  750.21586695, beta[1]=   -0.96100827
##  7: th=    20.37988727, dev=  747.82200424, beta[1]=   -0.92066545
##  8: th=    18.83307799, dev=  746.42608120, beta[1]=   -0.89436727
##  9: th=    17.93638047, dev=  745.59702917, beta[1]=   -0.87754518
## 10: th=    17.40366970, dev=  745.09788146, beta[1]=   -0.86691862
## 11: th=    17.08237733, dev=  744.79452759, beta[1]=   -0.86026004
## 12: th=    16.88678045, dev=  744.60902468, beta[1]=   -0.85610925
## 13: th=    16.76701639, dev=  744.49513807, beta[1]=   -0.85353014
## 14: th=    16.69342320, dev=  744.42504372, beta[1]=   -0.85193087
## 15: th=    16.64810171, dev=  744.38183456, beta[1]=   -0.85094039
## 16: th=    16.62015303, dev=  744.35517250, beta[1]=   -0.85032747
## 17: th=    16.60290326, dev=  744.33871074, beta[1]=   -0.84994841
## 18: th=    16.59225127, dev=  744.32854304, beta[1]=   -0.84971400
## 19: th=    16.58567140, dev=  744.32226144, beta[1]=   -0.84956905
## 20: th=    16.58160612, dev=  744.31838012, beta[1]=   -0.84947951
## 21: th=    16.57909413, dev=  744.31598166, beta[1]=   -0.84942417
## 22: th=    16.57754183, dev=  744.31449946, beta[1]=   -0.84938992
## 23: th=    16.57658253, dev=  744.31358348, beta[1]=   -0.84936873
## 24: th=    16.57598967, dev=  744.31301739, beta[1]=   -0.84935561
## 25: th=    16.57562328, dev=  744.31266754, beta[1]=   -0.84934761
## 26: th=    16.57534633, dev=  744.31240303, beta[1]=   -0.84934148
## 27: th=    16.57534633, dev=  744.31240318, beta[1]=   -0.84934146

## theta.ml: iter 0 'theta = 6.367940'

## theta.ml: iter1 theta =10.2142

## theta.ml: iter2 theta =14.9763
```

```
## theta.ml: iter3 theta =19.5793

## theta.ml: iter4 theta =22.4725

## theta.ml: iter5 theta =23.2757

## theta.ml: iter6 theta =23.3238

## theta.ml: iter7 theta =23.324

## theta.ml: iter8 theta =23.324

## th := est_theta(glmer(..)) = 23.32396 --> dev.= -2*logLik(.) = 700.1186
##  1: th=    11.48770215, dev=  683.30365569, beta[1]=   -1.55019502
##  2: th=    47.35559379, dev=  726.12403723, beta[1]=   -2.00100782
##  3: th=    4.786928061, dev=  674.43384558, beta[1]=   -0.92033123
##  4: th=    4.161655243, dev=  673.90290301, beta[1]=   -0.77880802
##  5: th=    3.294099069, dev=  673.36267368, beta[1]=   -0.50869929
##  6: th=    2.211951594, dev=  673.06472999, beta[1]=    0.06149685
##  7: th=    2.320348258, dev=  673.07013702, beta[1]=   -0.01355320
##  8: th=    2.195955918, dev=  673.06451803, beta[1]=    0.07332785
##  9: th=    2.182882497, dev=  673.06437617, beta[1]=    0.08497234
## 10: th=    1.715256416, dev=  673.25491377, beta[1]=    0.47866658
## 11: th=    2.127157847, dev=  673.06722691, beta[1]=    0.12418215
## 12: th=    2.182846090, dev=  673.06475025, beta[1]=    0.08269641
## 13: th=    2.187866887, dev=  673.06418670, beta[1]=    0.08312717
## 14: th=    2.188511184, dev=  673.06436344, beta[1]=    0.08482379
## 15: th=    2.185715388, dev=  673.06490800, beta[1]=    0.08427020
## 16: th=    2.187044838, dev=  673.06435549, beta[1]=    0.08561931
## 17: th=    2.187769604, dev=  673.06496702, beta[1]=    0.08309039
## 18: th=    2.188112964, dev=  673.06432098, beta[1]=    0.08463423
## 19: th=    2.187960877, dev=  673.06492579, beta[1]=    0.08290310
## 20: th=    2.187829728, dev=  673.06430183, beta[1]=    0.08459129
## 21: th=    2.187903377, dev=  673.06491781, beta[1]=    0.08293394
## 22: th=    2.187866887, dev=  673.06430501, beta[1]=    0.08460476

## theta.ml: iter 0 'theta = 10.388200'

## theta.ml: iter1 theta =16.8575

## theta.ml: iter2 theta =25.8566

## theta.ml: iter3 theta =36.8388

## theta.ml: iter4 theta =47.7094

## theta.ml: iter5 theta =55.1196

## theta.ml: iter6 theta =57.5782

## theta.ml: iter7 theta =57.7849

## theta.ml: iter8 theta =57.7862

## theta.ml: iter9 theta =57.7862

## th := est_theta(glmer(..)) = 57.78624 --> dev.= -2*logLik(.) = 1101.886
##  1: th=    28.46134463, dev= 1076.65990376, beta[1]=   -0.32799663
##  2: th=    117.3258026, dev= 1135.11708624, beta[1]=   -0.30120361
##  3: th=    11.85984869, dev= 1062.16682109, beta[1]=   -0.31415665
##  4: th=    8.527038855, dev= 1061.50877375, beta[1]=   -0.28844948
##  5: th=    9.259513575, dev= 1061.44895202, beta[1]=   -0.29651938
```

```
##  6: th=    9.184047021, dev= 1061.44796116, beta[1]=   -0.29528671
##  7: th=    9.172907631, dev= 1061.44849085, beta[1]=   -0.29501177
##  8: th=    9.212297961, dev= 1061.44846146, beta[1]=   -0.29618614
##  9: th=    9.194827680, dev= 1061.44807621, beta[1]=   -0.29526042
## 10: th=    9.176250094, dev= 1061.44850914, beta[1]=   -0.29504295
## 11: th=    9.188163373, dev= 1061.44846848, beta[1]=   -0.29614064
## 12: th=    9.181068078, dev= 1061.44804748, beta[1]=   -0.29512568
## 13: th=    9.187582899, dev= 1061.44855596, beta[1]=   -0.29515112
## 14: th=    9.185397446, dev= 1061.44853247, beta[1]=   -0.29513124
## 15: th=    9.182909052, dev= 1061.44853936, beta[1]=   -0.29510605
## 16: th=    9.184562814, dev= 1061.44854875, beta[1]=   -0.29512165
## 17: th=    9.183612339, dev= 1061.44854254, beta[1]=   -0.29511272
## 18: th=    9.184244033, dev= 1061.44854612, beta[1]=   -0.29511868
## 19: th=    9.183880985, dev= 1061.44854375, beta[1]=   -0.29511527
## 20: th=    9.184047021, dev= 1061.44854510, beta[1]=   -0.29511681

## theta.ml: iter 0 'theta = 6.596650'

## theta.ml: iter1 theta =10.154

## theta.ml: iter2 theta =14.2736

## theta.ml: iter3 theta =17.9897

## theta.ml: iter4 theta =20.1156

## theta.ml: iter5 theta =20.6171

## theta.ml: iter6 theta =20.6394

## theta.ml: iter7 theta =20.6395

## th := est_theta(glmer(..)) = 20.63946 --> dev.= -2*logLik(.) = 1046.116
##  1: th=    10.16551473, dev= 1032.56918268, beta[1]=    0.83367738
##  2: th=    41.90515909, dev= 1078.43202850, beta[1]=    0.85008532
##  3: th=    4.235972266, dev= 1038.10567900, beta[1]=    0.91363082
##  4: th=    7.913392892, dev= 1031.92376260, beta[1]=    0.84632097
##  5: th=    8.193297459, dev= 1031.89631365, beta[1]=    0.84425264
##  6: th=    8.258400227, dev= 1031.89534533, beta[1]=    0.84358415
##  7: th=    8.258026200, dev= 1031.89484104, beta[1]=    0.84399785
##  8: th=    8.226136212, dev= 1031.89559318, beta[1]=    0.84429406
##  9: th=    8.245830742, dev= 1031.89532427, beta[1]=    0.84385043
## 10: th=    8.252105510, dev= 1031.89513355, beta[1]=    0.84410892
## 11: th=    8.255764196, dev= 1031.89528017, beta[1]=    0.84373461
## 12: th=    8.257162118, dev= 1031.89503049, beta[1]=    0.84407395
## 13: th=    8.257696139, dev= 1031.89527813, beta[1]=    0.84371074
## 14: th=    8.258169063, dev= 1031.89500977, beta[1]=    0.84406702
## 15: th=    8.257888307, dev= 1031.89527853, beta[1]=    0.84370671
## 16: th=    8.258026200, dev= 1031.89500466, beta[1]=    0.84406822

## theta.ml: iter 0 'theta = 20.965100'

## theta.ml: iter1 theta =30.2205

## theta.ml: iter2 theta =38.5708

## theta.ml: iter3 theta =43.1268

## theta.ml: iter4 theta =44.0821

## theta.ml: iter5 theta =44.1158
```

```
## theta.ml: iter6 theta =44.1159
## th := est_theta(glmer(..)) = 44.11587 --> dev.= -2*logLik(.) = 1062.737
##  1: th=    21.72830369, dev= 1046.72300853, beta[1]=    0.13888976
##  2: th=    89.57028221, dev= 1094.54045587, beta[1]=    0.00986151
##  3: th=    9.054189019, dev= 1046.80514749, beta[1]=    0.31916985
##  4: th=    14.07074337, dev= 1044.37062887, beta[1]=    0.21323946
##  5: th=    14.07889854, dev= 1044.37013401, beta[1]=    0.21361024
##  6: th=    14.49714760, dev= 1044.37081383, beta[1]=    0.20774950
##  7: th=    14.23721193, dev= 1044.36778289, beta[1]=    0.21135176
##  8: th=    14.18505140, dev= 1044.36829791, beta[1]=    0.21209671
##  9: th=    14.33594381, dev= 1044.36801940, beta[1]=    0.20997329
## 10: th=    14.27167994, dev= 1044.36774684, beta[1]=    0.21086984
## 11: th=    14.26415613, dev= 1044.36777760, beta[1]=    0.21097707
## 12: th=    14.29619248, dev= 1044.36779579, beta[1]=    0.21052728
## 13: th=    14.28103793, dev= 1044.36776113, beta[1]=    0.21073917
## 14: th=    14.27406123, dev= 1044.36776546, beta[1]=    0.21083754
## 15: th=    14.26880563, dev= 1044.36777116, beta[1]=    0.21091160
## 16: th=    14.27058198, dev= 1044.36777163, beta[1]=    0.21088676
## 17: th=    14.27258946, dev= 1044.36776984, beta[1]=    0.21085847
## 18: th=    14.27126055, dev= 1044.36776982, beta[1]=    0.21087711
## 19: th=    14.27202734, dev= 1044.36776986, beta[1]=    0.21086637
## 20: th=    14.27167994, dev= 1044.36776978, beta[1]=    0.21087124
## theta.ml: iter 0 'theta = 6.752810'
## theta.ml: iter1 theta =10.2475
## theta.ml: iter2 theta =14.3831
## theta.ml: iter3 theta =18.3071
## theta.ml: iter4 theta =20.7815
## theta.ml: iter5 theta =21.4835
## theta.ml: iter6 theta =21.5273
## theta.ml: iter7 theta =21.5275
## theta.ml: iter8 theta =21.5275
## th := est_theta(glmer(..)) = 21.52747 --> dev.= -2*logLik(.) = 977.3329
##  1: th=    10.60288085, dev=  953.69206997, beta[1]=   -1.21703349
##  2: th=    43.70810732, dev= 1007.99396796, beta[1]=   -1.90055665
##  3: th=    4.418222826, dev=  938.44235614, beta[1]=   -0.50479347
##  4: th=    2.635552354, dev=  937.94710794, beta[1]=   -0.09224963
##  5: th=    3.276860535, dev=  937.31322161, beta[1]=   -0.26078825
##  6: th=    3.288377013, dev=  937.31343601, beta[1]=   -0.26346951
##  7: th=    3.275152156, dev=  937.31273253, beta[1]=   -0.26017971
##  8: th=    3.014319145, dev=  937.40517457, beta[1]=   -0.19541395
##  9: th=    3.172959779, dev=  937.32700956, beta[1]=   -0.23540413
## 10: th=    3.235735307, dev=  937.31559997, beta[1]=   -0.25047137
## 11: th=    3.260039899, dev=  937.31366821, beta[1]=   -0.25640905
## 12: th=    3.269371536, dev=  937.31335397, beta[1]=   -0.25887261
## 13: th=    3.273281873, dev=  937.31294200, beta[1]=   -0.25746460
## 14: th=    3.274719765, dev=  937.31330650, beta[1]=   -0.25999352
## 15: th=    3.275804593, dev=  937.31289261, beta[1]=   -0.25830130
## 16: th=    3.275401350, dev=  937.31329606, beta[1]=   -0.26017278
```

```
## 17: th=    3.274986991, dev=  937.31286859, beta[1]=   -0.25830432
## 18: th=    3.275247337, dev=  937.31329503, beta[1]=   -0.26013869
## 19: th=    3.275089067, dev=  937.31287261, beta[1]=   -0.25829680
## 20: th=    3.275152156, dev=  937.31329556, beta[1]=   -0.26011557

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge with max|grad| = 0.00120311
## (tol = 0.001, component 1)

## theta.ml: iter 0 'theta = 16.111700'

## theta.ml: iter1 theta =25.2129

## theta.ml: iter2 theta =36.2259

## theta.ml: iter3 theta =46.9926

## theta.ml: iter4 theta =54.2758

## theta.ml: iter5 theta =56.6948

## theta.ml: iter6 theta =56.9003

## theta.ml: iter7 theta =56.9017

## theta.ml: iter8 theta =56.9017

## th := est_theta(glmer(..)) = 56.90167 --> dev.= -2*logLik(.) = 611.3818
##  1: th=    28.02566505, dev=  597.45513844, beta[1]=    0.44007628
##  2: th=    115.5298068, dev=  629.91508469, beta[1]=    0.51041861
##  3: th=    11.67830091, dev=  589.36084979, beta[1]=    0.38463594
##  4: th=    8.334165173, dev=  588.56246706, beta[1]=    0.36796143
##  5: th=    8.008263360, dev=  588.52963855, beta[1]=    0.36651198
##  6: th=    7.387894954, dev=  588.49514036, beta[1]=    0.36398030
##  7: th=    5.122904000, dev=  588.69601094, beta[1]=    0.36491459
##  8: th=    7.205961305, dev=  588.49282716, beta[1]=    0.36329089
##  9: th=    7.190509344, dev=  588.49206625, beta[1]=    0.36344022
## 10: th=    6.317098758, dev=  588.52878948, beta[1]=    0.36128127
## 11: th=    6.843481963, dev=  588.49750319, beta[1]=    0.36220359
## 12: th=    7.055926682, dev=  588.49317859, beta[1]=    0.36285723
## 13: th=    7.133581301, dev=  588.49270358, beta[1]=    0.36317584
## 14: th=    7.168660809, dev=  588.49264247, beta[1]=    0.36328770
## 15: th=    7.182156097, dev=  588.49226917, beta[1]=    0.36388077
## 16: th=    7.196407553, dev=  588.49277460, beta[1]=    0.36338139
## 17: th=    7.187317541, dev=  588.49211983, beta[1]=    0.36362277
## 18: th=    7.192761688, dev=  588.49270761, beta[1]=    0.36331864
## 19: th=    7.189290016, dev=  588.49213634, beta[1]=    0.36364824
## 20: th=    7.191369579, dev=  588.49270007, beta[1]=    0.36329201
## 21: th=    7.190043578, dev=  588.49214134, beta[1]=    0.36365502
## 22: th=    7.190837912, dev=  588.49269727, beta[1]=    0.36327881
## 23: th=    7.190331433, dev=  588.49214200, beta[1]=    0.36365442
## 24: th=    7.190634844, dev=  588.49269614, beta[1]=    0.36327060
## 25: th=    7.190509344, dev=  588.49214119, beta[1]=    0.36365230

## theta.ml: iter 0 'theta = 13.718200'

## theta.ml: iter1 theta =17.6097

## theta.ml: iter2 theta =19.7313

## theta.ml: iter3 theta =20.1764
```

```
## theta.ml: iter4 theta =20.1922

## theta.ml: iter5 theta =20.1922

## th := est_theta(glmer(..)) = 20.19224 --> dev.= -2*logLik(.) = 712.9688
##  1: th=    9.945242466, dev=  696.19340019, beta[1]=    0.79757612
##  2: th=    40.99713383, dev=  743.29221173, beta[1]=    0.88255157
##  3: th=    4.144184766, dev=  688.79184957, beta[1]=    0.97308528
##  4: th=    4.343495158, dev=  688.84546975, beta[1]=    0.95296384
##  5: th=    3.977064136, dev=  688.79192724, beta[1]=    0.99173163
##  6: th=    4.060059860, dev=  688.78623333, beta[1]=    0.98226960
##  7: th=    4.059992108, dev=  688.78621592, beta[1]=    0.98257076
##  8: th=    4.028114276, dev=  688.78704280, beta[1]=    0.98577741
##  9: th=    4.045481528, dev=  688.78638484, beta[1]=    0.98394789
## 10: th=    4.053045746, dev=  688.78628692, beta[1]=    0.98335515
## 11: th=    4.056651510, dev=  688.78622994, beta[1]=    0.98266104
## 12: th=    4.058348830, dev=  688.78622478, beta[1]=    0.98275580
## 13: th=    4.059364353, dev=  688.78622411, beta[1]=    0.98224442
## 14: th=    4.059752316, dev=  688.78621173, beta[1]=    0.98259644
## 15: th=    4.059820063, dev=  688.78622662, beta[1]=    0.98210110
## 16: th=    4.059684569, dev=  688.78620464, beta[1]=    0.98258663
## 17: th=    4.059562254, dev=  688.78622796, beta[1]=    0.98209448
## 18: th=    4.059684569, dev=  688.78620414, beta[1]=    0.98257572

## theta.ml: iter 0 'theta = 0.919633'

## theta.ml: iter1 theta =1.63491

## theta.ml: iter2 theta =2.84507

## theta.ml: iter3 theta =4.74288

## theta.ml: iter4 theta =7.36666

## theta.ml: iter5 theta =10.3314

## theta.ml: iter6 theta =12.6999

## theta.ml: iter7 theta =13.694

## theta.ml: iter8 theta =13.8187

## theta.ml: iter9 theta =13.8204

## theta.ml: iter10 theta =13.8204

## th := est_theta(glmer(..)) = 13.82042 --> dev.= -2*logLik(.) = 677.5277
##  1: th=    6.806945644, dev=  660.96639966, beta[1]=   -0.30578387
##  2: th=    28.06017677, dev=  705.49337239, beta[1]=   -0.92819274
##  3: th=    2.836455776, dev=  653.88960863, beta[1]=    0.14945489
##  4: th=    2.955181188, dev=  653.95226511, beta[1]=    0.12807090
##  5: th=    2.626949205, dev=  653.82612502, beta[1]=    0.19036549
##  6: th=    1.574775274, dev=  654.30649432, beta[1]=    0.53531692
##  7: th=    2.546399903, dev=  653.81850635, beta[1]=    0.20733036
##  8: th=    2.528451733, dev=  653.81795861, beta[1]=    0.21137801
##  9: th=    2.515054110, dev=  653.81794152, beta[1]=    0.21428738
## 10: th=    2.521067036, dev=  653.81806613, beta[1]=    0.21305849
## 11: th=    2.103210121, dev=  653.94078214, beta[1]=    0.32038437
## 12: th=    2.348994873, dev=  653.83993617, beta[1]=    0.25278831
## 13: th=    2.450282794, dev=  653.82185801, beta[1]=    0.22873038
## 14: th=    2.490113996, dev=  653.81879465, beta[1]=    0.21967219
```

```
## 15: th=    2.505498485, dev=  653.81828080, beta[1]=    0.21647382
## 16: th=    2.511399892, dev=  653.81814967, beta[1]=    0.21526943
## 17: th=    2.517349149, dev=  653.81809332, beta[1]=    0.21403221
## 18: th=    2.513657695, dev=  653.81789968, beta[1]=    0.21488734
## 19: th=    2.512795052, dev=  653.81796379, beta[1]=    0.21549814
## 20: th=    2.514190987, dev=  653.81812324, beta[1]=    0.21472603
## 21: th=    2.513328160, dev=  653.81794410, beta[1]=    0.21529933
## 22: th=    2.513861381, dev=  653.81811434, beta[1]=    0.21468994
## 23: th=    2.513531819, dev=  653.81793506, beta[1]=    0.21520907
## 24: th=    2.513735495, dev=  653.81811177, beta[1]=    0.21463267
## 25: th=    2.513609614, dev=  653.81792585, beta[1]=    0.21513224
## 26: th=    2.513657695, dev=  653.81801888, beta[1]=    0.21565268

## theta.ml: iter 0 'theta = 12.159800'

## theta.ml: iter1 theta =20.3462

## theta.ml: iter2 theta =32.4899

## theta.ml: iter3 theta =49.3889

## theta.ml: iter4 theta =71.2616

## theta.ml: iter5 theta =96.9624

## theta.ml: iter6 theta =122.869

## theta.ml: iter7 theta =142.586

## theta.ml: iter8 theta =151.148

## theta.ml: iter9 theta =152.374

## theta.ml: iter10 theta =152.396

## theta.ml: iter11 theta =152.396

## th := est_theta(glmer(..)) = 152.396 --> dev.= -2*logLik(.) = 580.8714
##  1: th=    75.05928028, dev=  571.18259533, beta[1]=   -1.45650983
##  2: th=    309.4158206, dev=  589.47395485, beta[1]=   -1.76785967
##  3: th=    31.27721892, dev=  561.10588599, beta[1]=   -1.14899190
##  4: th=    18.20816901, dev=  557.13717361, beta[1]=   -0.89358823
##  5: th=    13.03322414, dev=  555.62209228, beta[1]=   -0.70741421
##  6: th=    10.59996477, dev=  555.02206979, beta[1]=   -0.58293664
##  7: th=    9.329050675, dev=  554.77343041, beta[1]=   -0.50318417
##  8: th=    8.620986791, dev=  554.66513248, beta[1]=   -0.45304433
##  9: th=    8.210516581, dev=  554.61526958, beta[1]=   -0.42184684
## 10: th=    7.966664116, dev=  554.58992666, beta[1]=   -0.40566232
## 11: th=    7.819590050, dev=  554.57829409, beta[1]=   -0.39042135
## 12: th=    7.730054072, dev=  554.57028229, beta[1]=   -0.38217087
## 13: th=    7.675231147, dev=  554.56655012, beta[1]=   -0.37578565
## 14: th=    7.641543299, dev=  554.56470862, beta[1]=   -0.37055358
## 15: th=    7.620797036, dev=  554.56405738, beta[1]=   -0.37370397
## 16: th=    7.594923995, dev=  554.56131504, beta[1]=   -0.36893406
## 17: th=    7.604796233, dev=  554.56302797, beta[1]=   -0.37262138
## 18: th=    7.598693340, dev=  554.56159403, beta[1]=   -0.36891691
## 19: th=    7.594541774, dev=  554.56238891, beta[1]=   -0.37184112
## 20: th=    7.596755137, dev=  554.56149736, beta[1]=   -0.36859002
## 21: th=    7.595623377, dev=  554.56245616, beta[1]=   -0.37167491
## 22: th=    7.595191127, dev=  554.56142687, beta[1]=   -0.36828070
```

```
## 23: th=    7.594777997, dev=  554.56240578, beta[1]=   -0.37149696
## 24: th=    7.595050807, dev=  554.56143835, beta[1]=   -0.36814204
## 25: th=    7.594923995, dev=  554.56241636, beta[1]=   -0.37150853
```

```
M1[1]
```

```
## [[1]]
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: poisson  ( log )
## Formula: stops ~ 1 + dcjs + ethi + (1 | precinct)
##    Data: stop_df
##  Subset: ok
##       AIC       BIC    logLik  deviance  df.resid
##  692.4676  703.6381 -341.2338  682.4676        64
## Random effects:
##  Groups   Name        Std.Dev.
##  precinct (Intercept) 0.5812
## Number of obs: 69, groups:  precinct, 51
## Fixed Effects:
## (Intercept)         dcjs        ethi2        ethi3
##     -1.1428       1.0574      -0.2087      -0.7308
```

```
M2[1]
```

```
## [[1]]
## Generalized linear mixed model fit by maximum likelihood (Adaptive
##   Gauss-Hermite Quadrature, nAGQ = 0) [glmerMod]
##  Family: Negative Binomial(3029.707)  ( log )
## Formula: stops ~ 1 + dcjs + ethi + (1 | precinct)
##    Data: stop_df
##  Subset: ok
##       AIC       BIC    logLik  deviance  df.resid
##  693.0864  706.4910 -340.5432  681.0864        63
## Random effects:
##  Groups   Name        Std.Dev.
##  precinct (Intercept) 0.5806
## Number of obs: 69, groups:  precinct, 51
## Fixed Effects:
## (Intercept)         dcjs        ethi2        ethi3
##     -1.1052       1.0529      -0.2125      -0.7315
```

```
anova(M1[[2]],M2[[2]])
```

```
## Data: stop_df
## Subset: ok
## Models:
## M1[[2]]: stops ~ 1 + dcjs + ethi + (1 | precinct)
## M2[[2]]: stops ~ 1 + dcjs + ethi + (1 | precinct)
##         Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
## M1[[2]]  5 867.71 878.88 -428.86   857.71
## M2[[2]]  6 865.34 878.74 -426.67   853.34 4.3706      1    0.03656 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

negative binomial with overdispersion effect

how do we know proposal distribution? MVN Poisson GLM with random effects If NB: link function for negative binomial, try a bayesian glm package, see the parametization, and change the link for negative binomial, the r parameter: you need to sample both from beta and r for posterior sampling

prior for beta: MVN or whatever in the package prior for r: uninformative uniform distribution

```
stop_clean <- as.data.frame(cbind(stop_df$stops,stop_df$precinct.category,stop_df$crime,stop_df$dcjs,st
colnames(stop_clean) <- c("stops","precinct.category","crime","dcjs","arrests","black","hispanic","whit
```

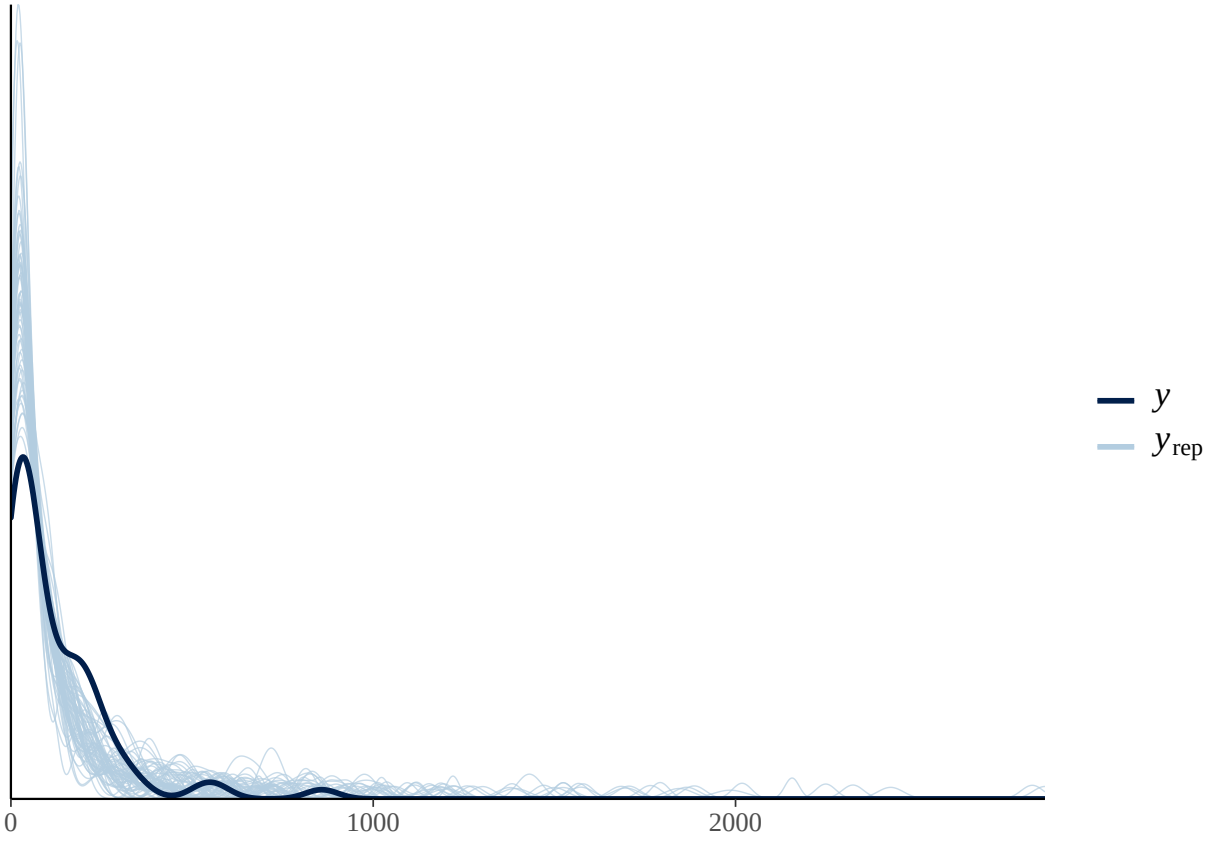
```
## Using default s_alpha = 2.93
## Using default s_beta = 2.93/sqrt(nvar)
##
## Starting Random Walk Metropolis Sampler for Negative Binomial Regression
##    69  obs;  4  covariates (including intercept);
## Prior Parameters:
## betabar
## [1] 0 0 0 0
## A
##      [,1] [,2] [,3] [,4]
## [1,] 0.01 0.00 0.00 0.00
## [2,] 0.00 0.01 0.00 0.00
## [3,] 0.00 0.00 0.01 0.00
## [4,] 0.00 0.00 0.00 0.01
## a
## [1] 0.5
## b
## [1] 0.1
##
## MCMC Parms:
## R=  1000  keep=  1  nprint=  100
## s_alpha =  2.38
## s_beta =  1.19
##
##  Initializing RW Increment Covariance Matrix...
## beta_mle =  -0.9053304 1.028122 -0.2451014 -1.112769
## alpha_mle =  5.101863
##  MCMC Iteration (est time to end - min)
##  100 (0.0)
##  200 (0.0)
##  300 (0.0)
##  400 (0.0)
##  500 (0.0)
##  600 (0.0)
##  700 (0.0)
##  800 (0.0)
##  900 (0.0)
##  1000 (0.0)
##  Total Time Elapsed: 0.00

## Summary of alpha/beta draw

## Summary of Posterior Marginal Distributions
## Moments
##   tvalues mean std dev num se rel eff sam size
## 1       5  4.1     1.6   0.27      26       33
##
```

```
## Quantiles
##   tvalues 2.5%   5% 50% 95% 97.5%
## 1       5 0.26 0.47 4.4 6.2   6.6
##    based on 900 valid draws (burn-in=100)
```

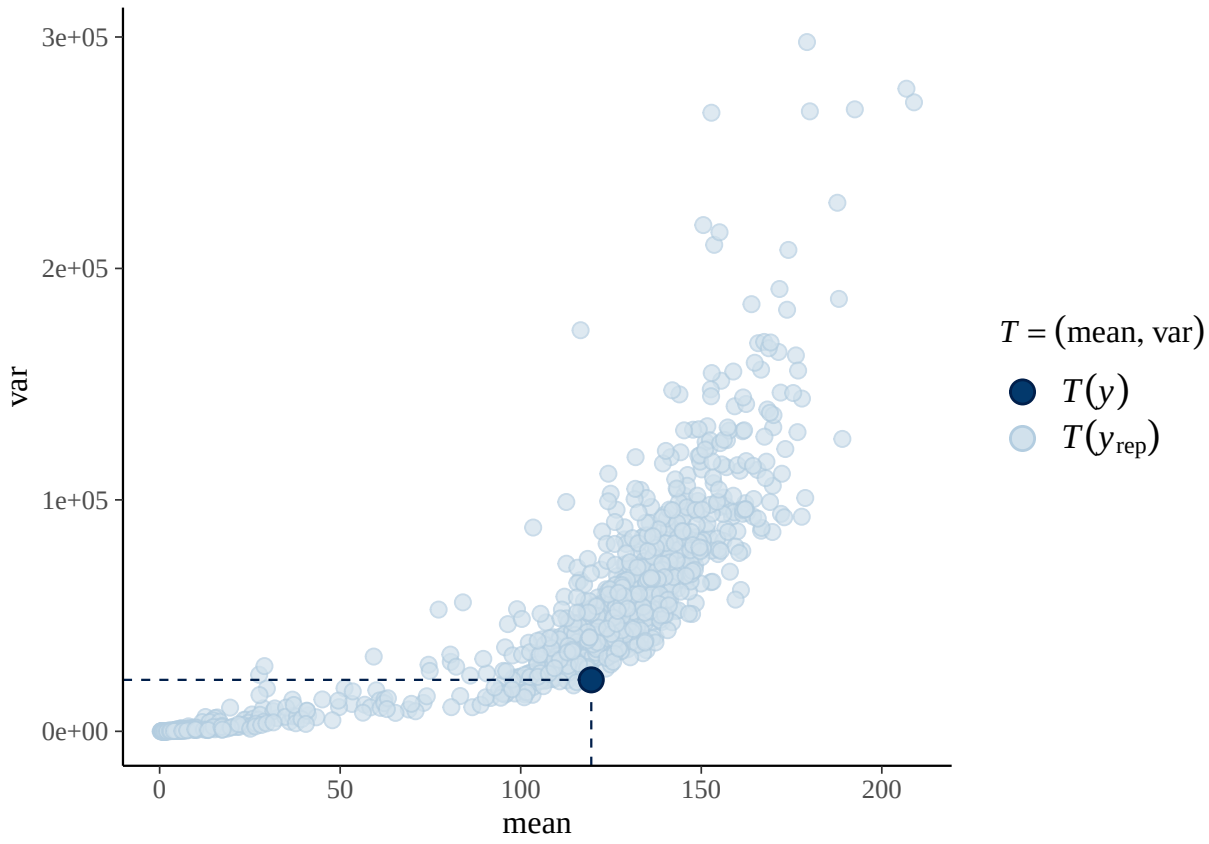
```
## Summary of Posterior Marginal Distributions
## Moments
##   tvalues  mean std dev num se rel eff sam size
## 1   -1.11 -0.91    0.63  0.079      14       60
## 2    1.05  1.01    0.13  0.021      25       36
## 3   -0.21 -0.27    0.22  0.029      16       56
## 4   -0.73 -1.11    0.23  0.025      11       82
##
## Quantiles
##   tvalues  2.5%    5%   50%    95% 97.5%
## 1   -1.11 -1.96 -1.84 -0.92  0.211  0.55
## 2    1.05  0.63  0.78  1.02  1.166  1.19
## 3   -0.21 -0.77 -0.60 -0.26  0.075  0.13
## 4   -0.73 -1.54 -1.46 -1.09 -0.761 -0.65
##    based on 900 valid draws (burn-in=100)
```

```
## [1] 0.435
```

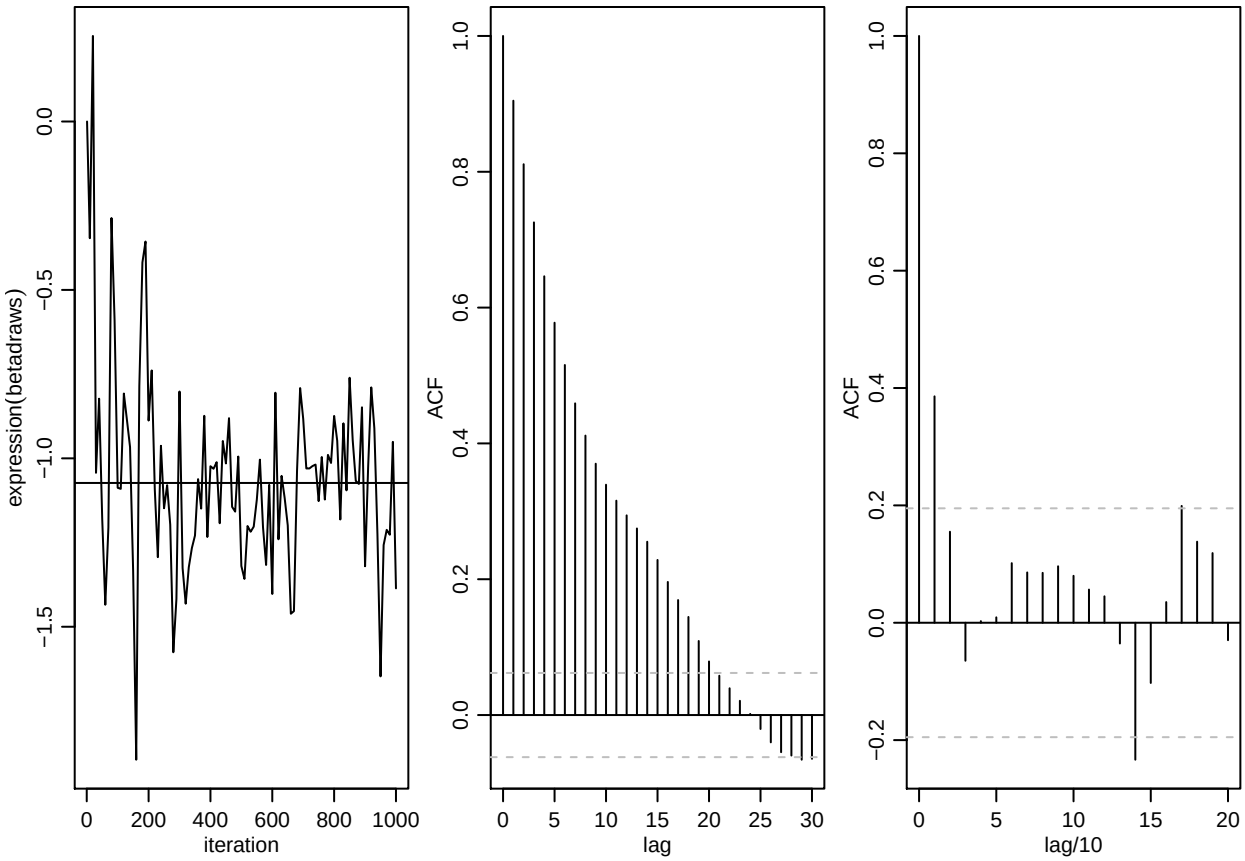
```
betadraws <- out$betadraw #posterior beta
alphadraws <- out$alphadraw #posterior alpha
z.mcmc <- NULL
# posterior predictive
for(i in 1:nrow(betadraws)){
  z <- simnegbin(X,betadraws[i,],alphadraws[i]) #sampling from the posterior
  z.mcmc <- rbind(z.mcmc, z)
}
ppc_dens_overlay(data1$stops, z.mcmc[940:1000,])
```




```
ppc_stat_2d(data1$stops, z.mcmc, stat = c("mean", "var"))
```




```
#### Figure 10.5 (traceplots & ACF)
par(mar=c(2.75,2.75,.5,.5),mgp=c(1.7,.7,0))
par(mfrow=c(1,3))
blabs<-c(expression(betadraws[,1]),expression(betadraws[,2]),expression(betadraws[,3],expression(betadra
thin<-c(1,(1:100)*(R/100))
j<-4
plot(thin,betadraws[thin,j],type="l",xlab="iteration",ylab=blabs[j])
abline(h=mean(betadraws[,j]) )

acf(betadraws[,j],ci.col="gray",xlab="lag")
acf(betadraws[thin,j],xlab="lag/10",ci.col="gray") #ACF of thinned chain
```



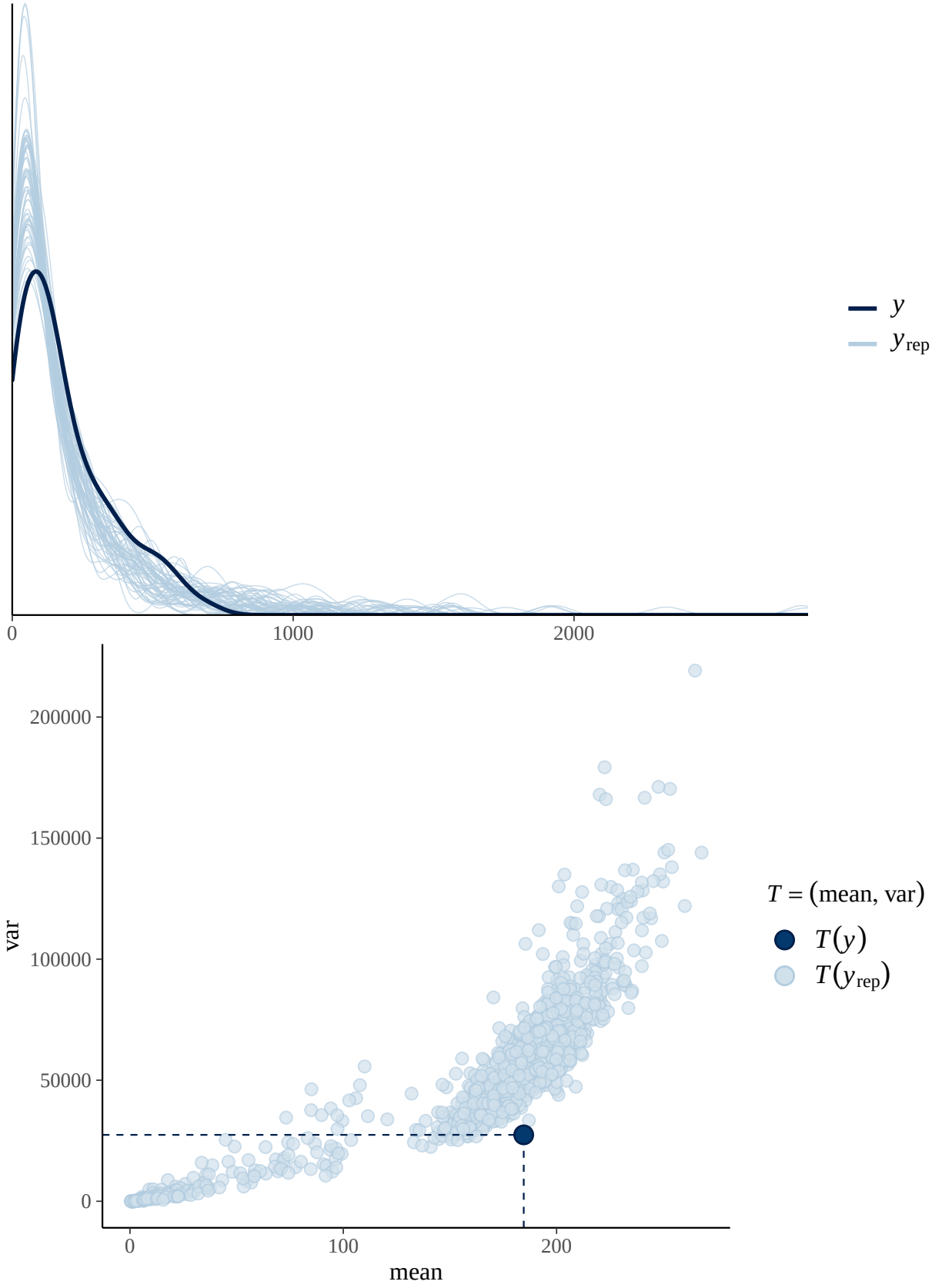

```
## Using default s_alpha = 2.93
## Using default s_beta = 2.93/sqrt(nvar)
##
## Starting Random Walk Metropolis Sampler for Negative Binomial Regression
##    96  obs;  4  covariates (including intercept);
## Prior Parameters:
## betabar
## [1] 0 0 0 0
## A
##      [,1] [,2] [,3] [,4]
## [1,] 0.01 0.00 0.00 0.00
## [2,] 0.00 0.01 0.00 0.00
## [3,] 0.00 0.00 0.01 0.00
## [4,] 0.00 0.00 0.00 0.01
## a
## [1] 0.5
## b
## [1] 0.1
##
## MCMC Parms:
## R=  1000  keep=  1  nprint=  100
## s_alpha =  2.38
## s_beta =  1.19
##
##  Initializing RW Increment Covariance Matrix...
## beta_mle =  -0.5819138 0.9908039 -0.366685 -1.165407
```

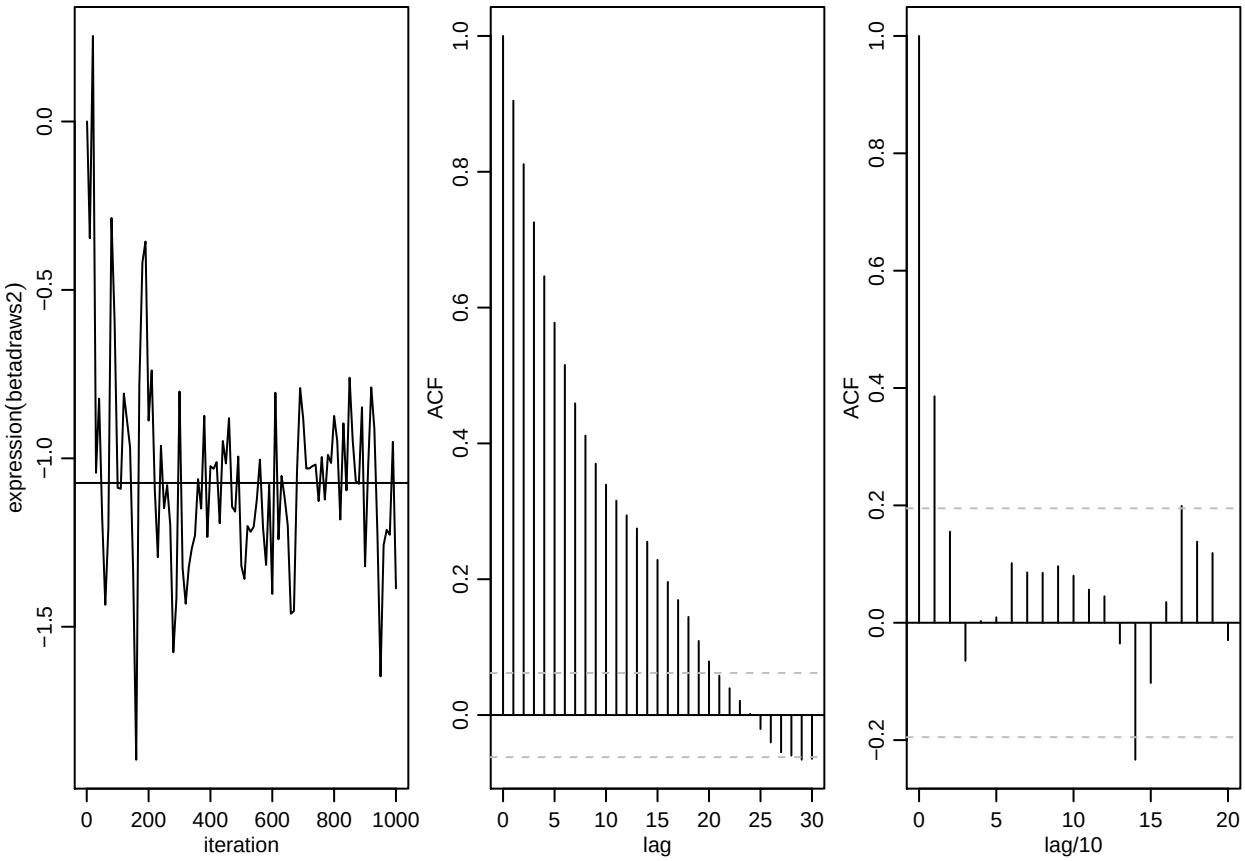
```
## alpha_mle =  4.576123
##  MCMC Iteration (est time to end - min)
##  100 (0.0)
##  200 (0.0)
##  300 (0.0)
##  400 (0.0)
##  500 (0.0)
##  600 (0.0)
##  700 (0.0)
##  800 (0.0)
##  900 (0.0)
##  1000 (0.0)
##  Total Time Elapsed: 0.00
```

```
## Summary of alpha/beta draw
```

```
## Summary of Posterior Marginal Distributions
## Moments
##   tvalues mean std dev num se rel eff sam size
## 1       5  3.6     1.6   0.31      36       25
##
## Quantiles
##   tvalues  2.5%   5% 50% 95% 97.5%
## 1       5 0.072 0.12 4.1 5.2   5.5
##    based on 900 valid draws (burn-in=100)
```

```
## Summary of Posterior Marginal Distributions
## Moments
##   tvalues  mean std dev num se rel eff sam size
## 1   -0.30 -0.51    0.73  0.117      23       39
## 2    0.94  0.93    0.19  0.036      32       28
## 3   -0.37 -0.34    0.23  0.038      24       36
## 4   -1.04 -1.29    0.49  0.089      30       29
##
## Quantiles
##   tvalues  2.5%    5%   50%    95%  97.5%
## 1   -0.30 -2.03 -1.29 -0.60  0.702  1.694
## 2    0.94  0.24  0.48  0.99  1.079  1.113
## 3   -0.37 -1.08 -0.55 -0.34 -0.025  0.072
## 4   -1.04 -3.03 -2.30 -1.18 -0.840 -0.508
##    based on 900 valid draws (burn-in=100)
```


0
1000
2000
y
y_rep
200000
150000
100000
50000
0
var
0
100
200
mean
T = (mean, var)
T(y)
T(y_rep)



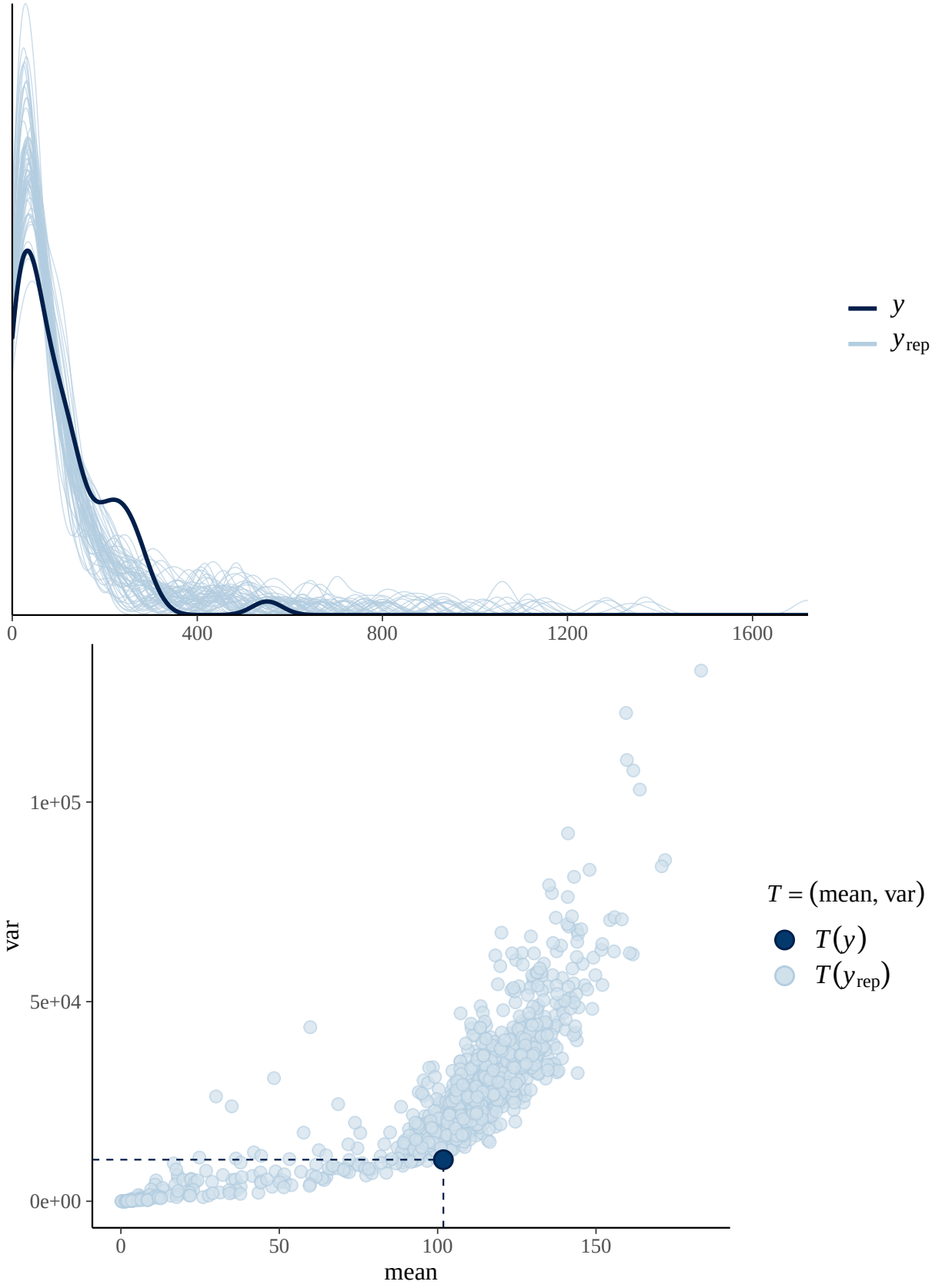

```
## Using default s_alpha = 2.93
## Using default s_beta = 2.93/sqrt(nvar)
##
## Starting Random Walk Metropolis Sampler for Negative Binomial Regression
##    60  obs;  4  covariates (including intercept);
## Prior Parameters:
## betabar
## [1] 0 0 0 0
## A
##      [,1] [,2] [,3] [,4]
## [1,] 0.01 0.00 0.00 0.00
## [2,] 0.00 0.01 0.00 0.00
## [3,] 0.00 0.00 0.01 0.00
## [4,] 0.00 0.00 0.00 0.01
## a
## [1] 0.5
## b
## [1] 0.1
##
## MCMC Parms:
## R=  1000  keep=  1  nprint=  100
## s_alpha =  2.38
## s_beta =  1.19
##
##  Initializing RW Increment Covariance Matrix...
## beta_mle =  0.310728 0.8832016 -0.6048846 -1.086839
```

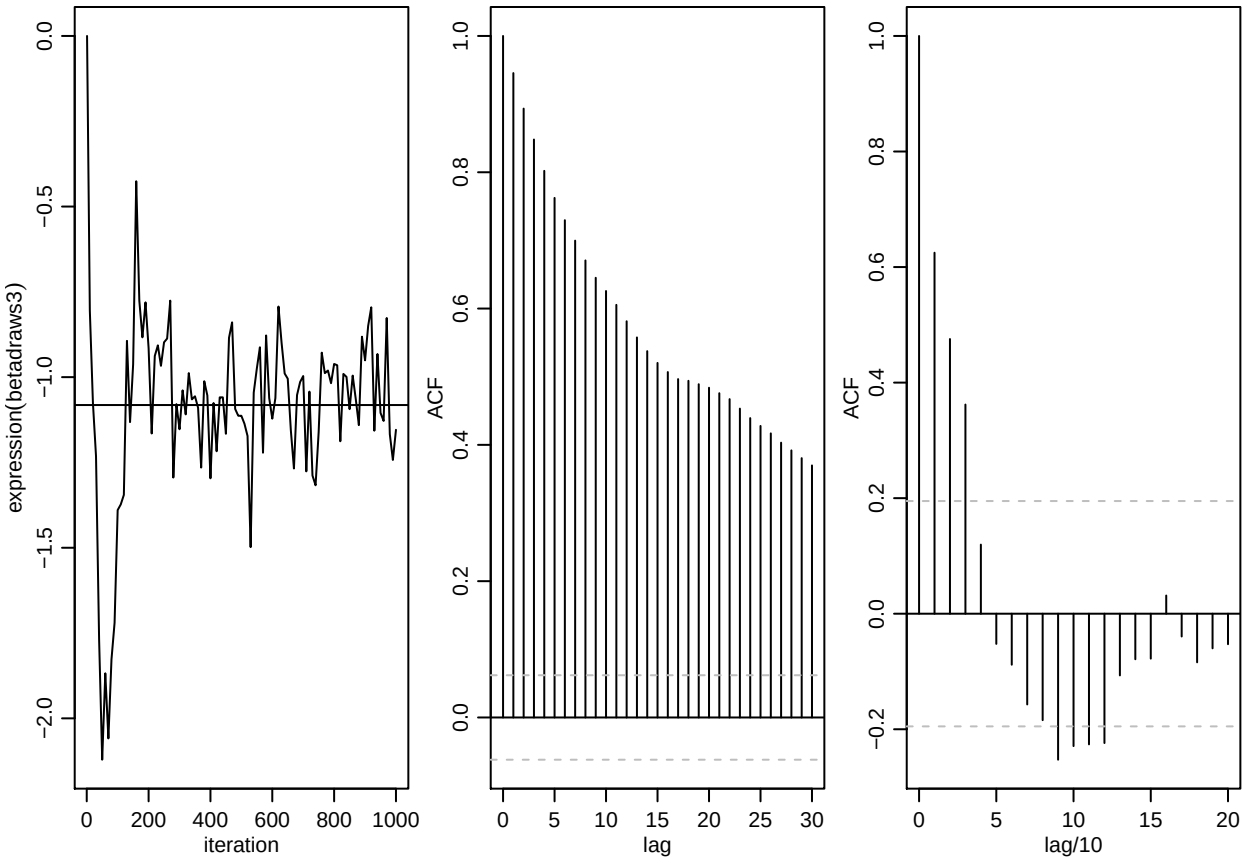
```
## alpha_mle =  5.995455
##  MCMC Iteration (est time to end - min)
##  100 (0.0)
##  200 (0.0)
##  300 (0.0)
##  400 (0.0)
##  500 (0.0)
##  600 (0.0)
##  700 (0.0)
##  800 (0.0)
##  900 (0.0)
##  1000 (0.0)
##  Total Time Elapsed: 0.00
```

```
## Summary of alpha/beta draw
```

```
## Summary of Posterior Marginal Distributions
## Moments
##   tvalues mean std dev num se rel eff sam size
## 1       5  4.9     1.6   0.25      22       41
##
## Quantiles
##   tvalues 2.5%  5% 50% 95% 97.5%
## 1       5 0.51 1.2   5 7.3   7.8
##    based on 900 valid draws (burn-in=100)
```

```
## Summary of Posterior Marginal Distributions
## Moments
##   tvalues  mean std dev num se rel eff sam size
## 1    0.36  0.27   0.413  0.046    11.4       75
## 2    0.86  0.87   0.091  0.014    20.4       43
## 3   -0.56 -0.58   0.189  0.018     8.5      100
## 4   -1.08 -1.03   0.175  0.022    13.7       64
##
## Quantiles
##   tvalues  2.5%    5%   50%   95% 97.5%
## 1    0.36 -0.47 -0.39  0.27  0.96  1.03
## 2    0.86  0.65  0.69  0.89  0.99  1.01
## 3   -0.56 -0.96 -0.90 -0.60 -0.27 -0.22
## 4   -1.08 -1.37 -1.30 -1.05 -0.76 -0.69
##    based on 900 valid draws (burn-in=100)
```


0
400
800
1200
1600
y
y_rep
1e+05
5e+04
0e+00
var
0
50
100
150
mean
T = (mean, var)
T(y)
T(y_rep)



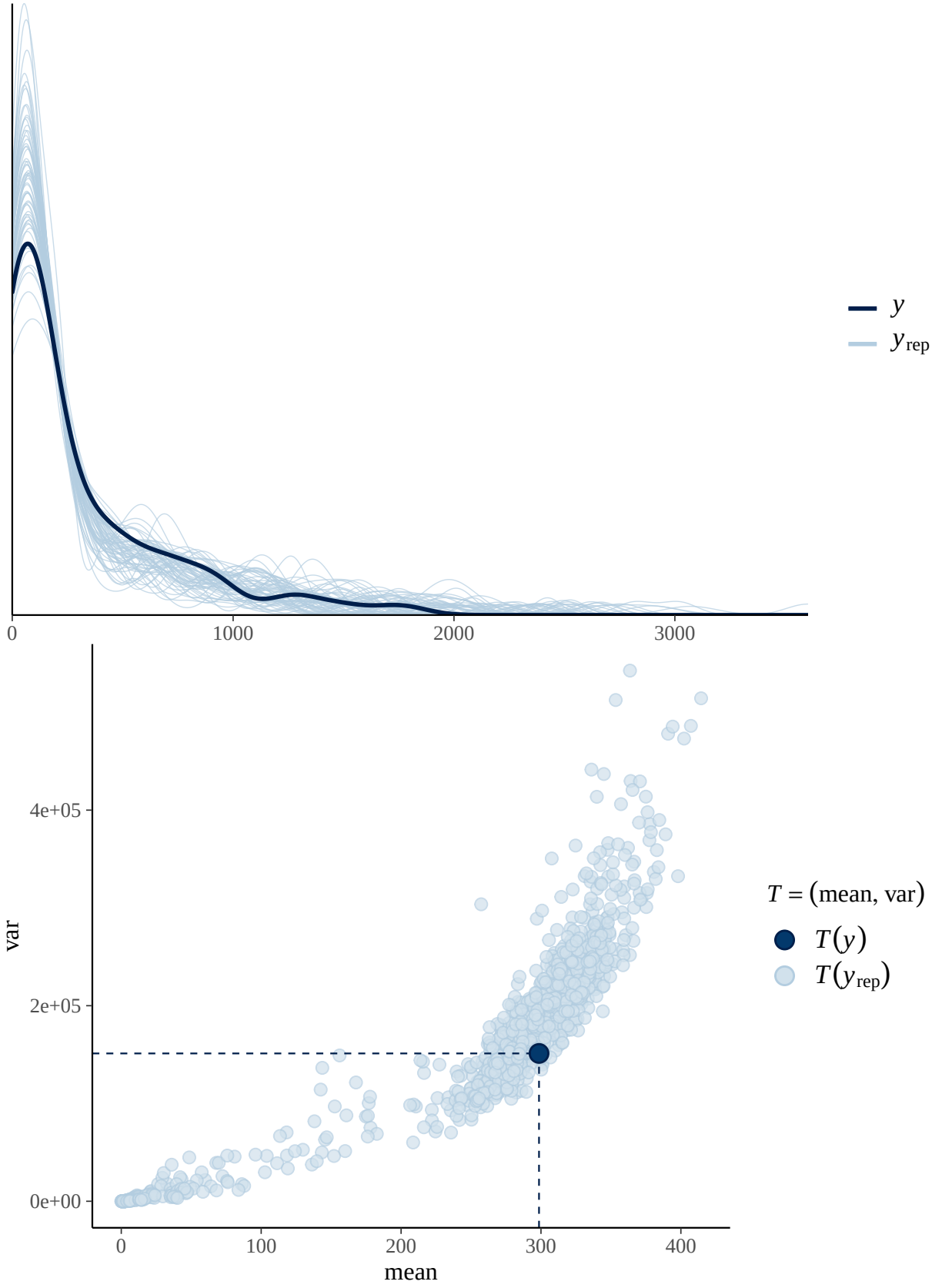

```
## Using default s_alpha = 2.93
## Using default s_beta = 2.93/sqrt(nvar)
##
## Starting Random Walk Metropolis Sampler for Negative Binomial Regression
##    69  obs;  4  covariates (including intercept);
## Prior Parameters:
## betabar
## [1] 0 0 0 0
## A
##      [,1] [,2] [,3] [,4]
## [1,] 0.01 0.00 0.00 0.00
## [2,] 0.00 0.01 0.00 0.00
## [3,] 0.00 0.00 0.01 0.00
## [4,] 0.00 0.00 0.00 0.01
## a
## [1] 0.5
## b
## [1] 0.1
##
## MCMC Parms:
## R=  1000  keep=  1  nprint=  100
## s_alpha =  2.38
## s_beta =  1.19
##
##  Initializing RW Increment Covariance Matrix...
## beta_mle =  0.6551693 0.9996903 -0.2505309 -0.9235909
```

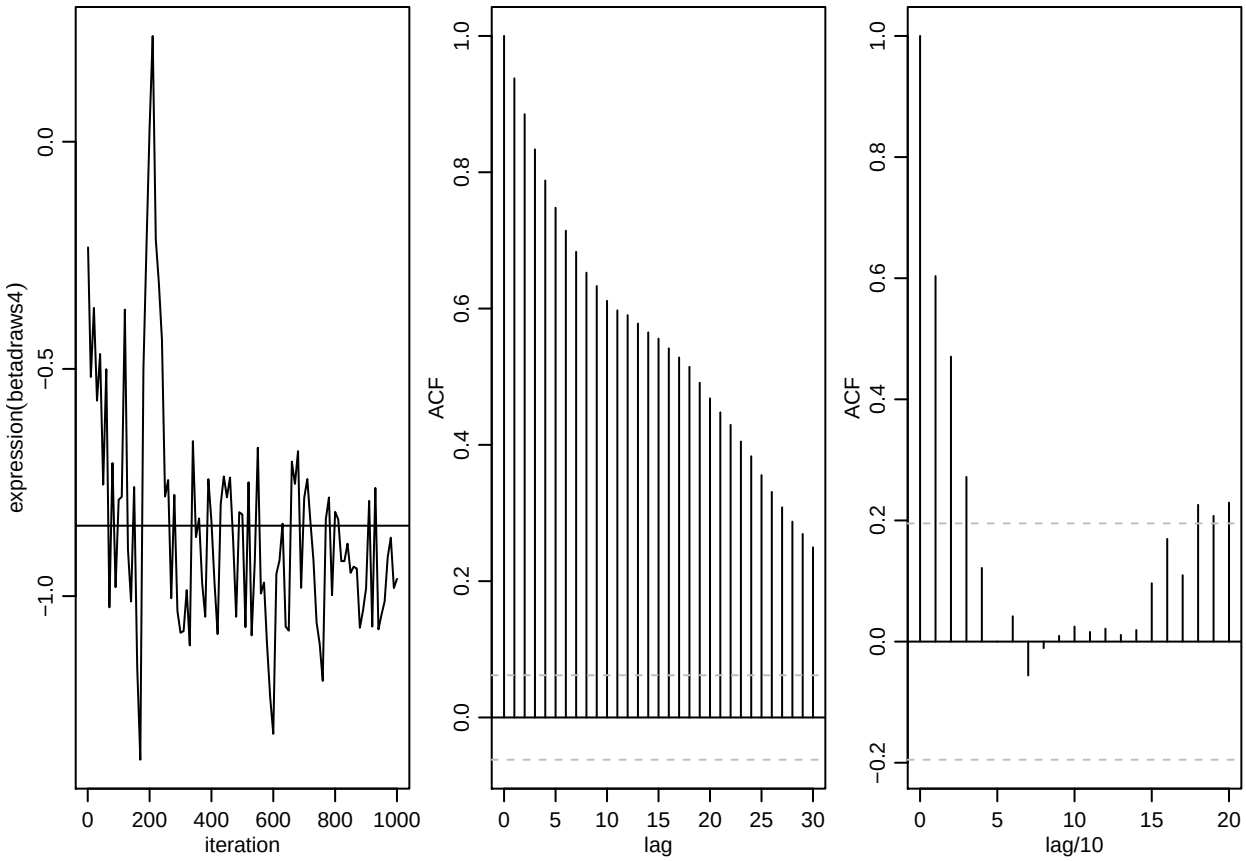
```
## alpha_mle =  6.650405
##  MCMC Iteration (est time to end - min)
##  100 (0.0)
##  200 (0.0)
##  300 (0.0)
##  400 (0.0)
##  500 (0.0)
##  600 (0.0)
##  700 (0.0)
##  800 (0.0)
##  900 (0.0)
##  1000 (0.0)
##  Total Time Elapsed: 0.00
```

```
## Summary of alpha/beta draw
```

```
## Summary of Posterior Marginal Distributions
## Moments
##   tvalues mean std dev num se rel eff sam size
## 1       5  5.3     2.2   0.42      32       28
##
## Quantiles
##   tvalues  2.5%   5% 50% 95% 97.5%
## 1       5 0.068 0.12 5.7   8   8.6
##    based on 900 valid draws (burn-in=100)
```

```
## Summary of Posterior Marginal Distributions
## Moments
##   tvalues  mean std dev num se rel eff sam size
## 1    0.32  0.60    0.34  0.044      15       56
## 2    1.05  0.96    0.12  0.023      31       29
## 3   -0.13 -0.31    0.21  0.036      26       35
## 4   -0.75 -0.87    0.26  0.043      25       35
##
## Quantiles
##   tvalues  2.5%     5%   50%    95%  97.5%
## 1    0.32 -0.33  0.021  0.63  1.043  1.121
## 2    1.05  0.58  0.640  0.99  1.089  1.110
## 3   -0.13 -0.98 -0.774 -0.28 -0.063 -0.024
## 4   -0.75 -1.22 -1.171 -0.92 -0.288 -0.060
##    based on 900 valid draws (burn-in=100)
```


0
1000
2000
3000
y
y_{rep}
4e+05
2e+05
0e+00
var
0
100
200
300
400
mean
$T = (\text{mean}, \text{var})$
$T(y)$
$T(y_{\text{rep}})$



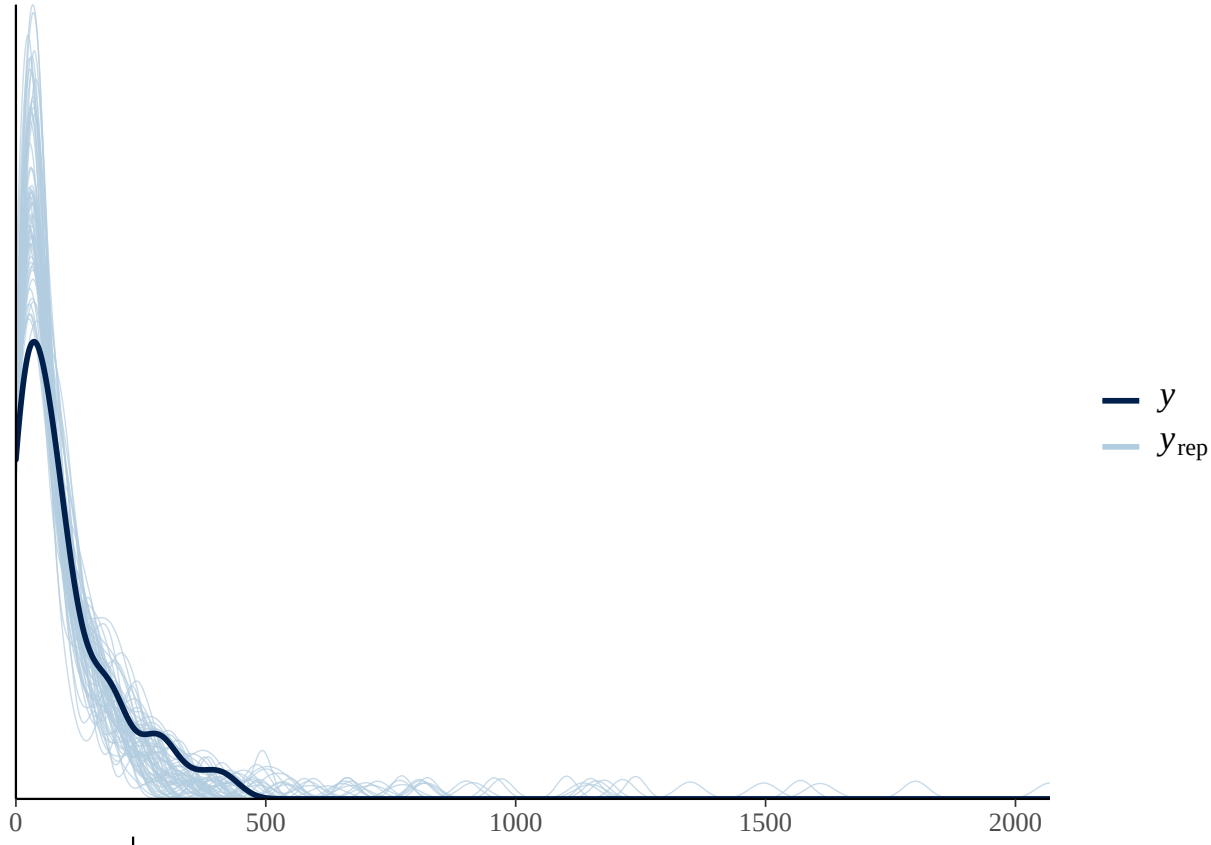

```
## Using default s_alpha = 2.93
## Using default s_beta = 2.93/sqrt(nvar)
##
## Starting Random Walk Metropolis Sampler for Negative Binomial Regression
##    69  obs;  4  covariates (including intercept);
## Prior Parameters:
## betabar
## [1] 0 0 0 0
## A
##      [,1] [,2] [,3] [,4]
## [1,] 0.01 0.00 0.00 0.00
## [2,] 0.00 0.01 0.00 0.00
## [3,] 0.00 0.00 0.01 0.00
## [4,] 0.00 0.00 0.00 0.01
## a
## [1] 0.5
## b
## [1] 0.1
##
## MCMC Parms:
## R=  1000  keep=  1  nprint=  100
## s_alpha =  2.38
## s_beta =  1.19
##
##  Initializing RW Increment Covariance Matrix...
## beta_mle =  -0.3607132 0.8890698 -0.02683061 -0.2347519
```

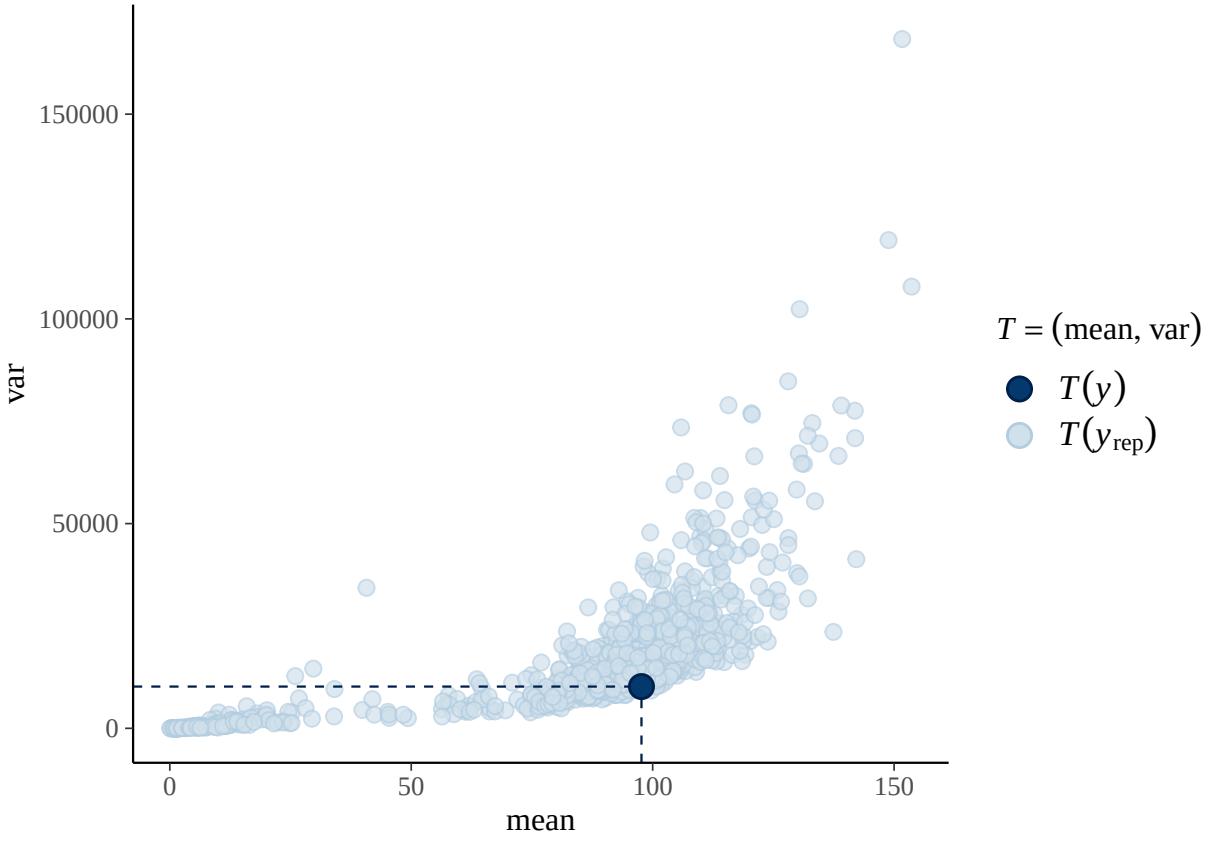
```
## alpha_mle =  4.596502
##  MCMC Iteration (est time to end - min)
##  100 (0.0)
##  200 (0.0)
##  300 (0.0)
##  400 (0.0)
##  500 (0.0)
##  600 (0.0)
##  700 (0.0)
##  800 (0.0)
##  900 (0.0)
##  1000 (0.0)
##  Total Time Elapsed: 0.00
```

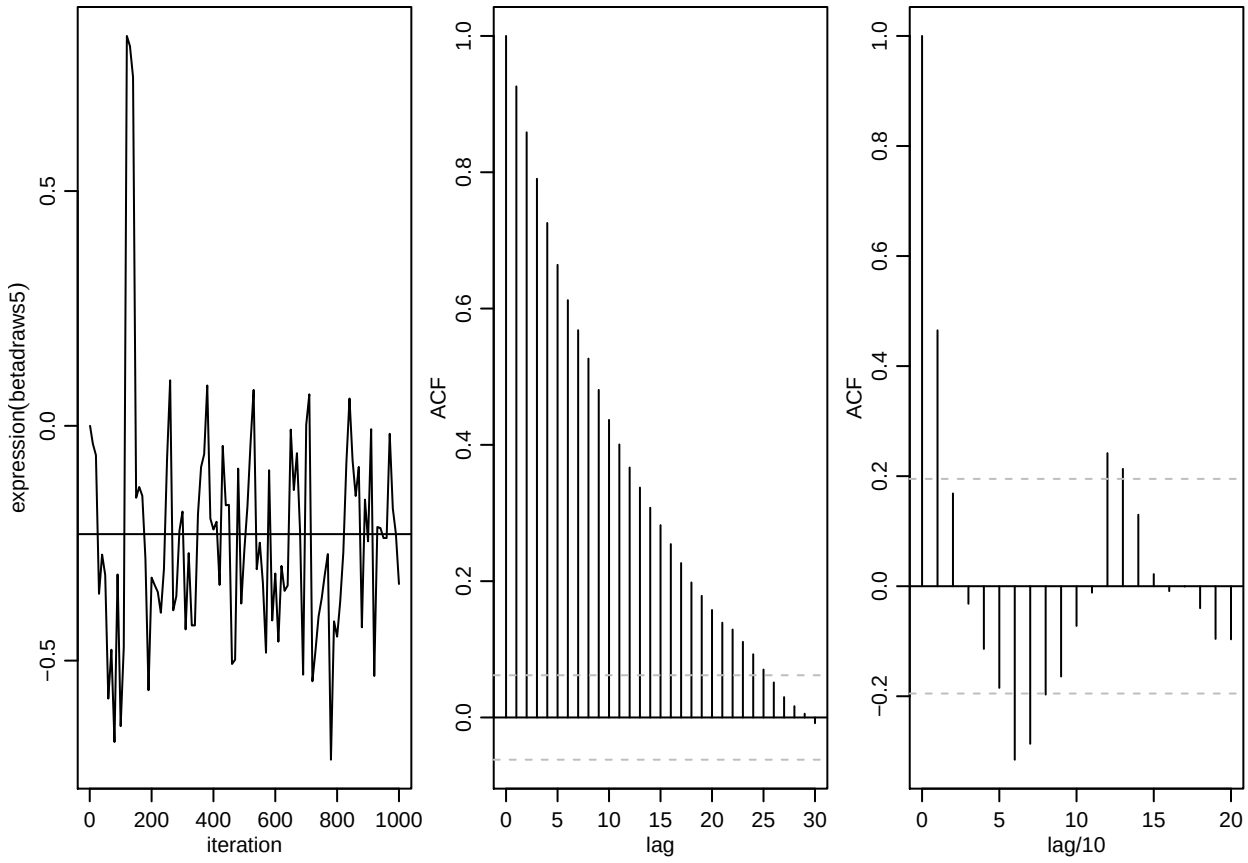
```
## Summary of alpha/beta draw
```

```
## Summary of Posterior Marginal Distributions
## Moments
##   tvalues mean std dev num se rel eff sam size
## 1       5  3.8     1.2   0.18      23       39
##
## Quantiles
##   tvalues 2.5%   5% 50% 95% 97.5%
## 1       5 0.25 0.82 3.9 5.4   5.7
##    based on 900 valid draws (burn-in=100)
```

```
## Summary of Posterior Marginal Distributions
## Moments
##   tvalues    mean std dev num se rel eff sam size
## 1 -0.8493 -0.3596   0.431  0.047      11       82
## 2  0.9616  0.8710   0.078  0.011      19       45
## 3  0.1663 -0.0062   0.262  0.037      18       50
## 4  0.0042 -0.2215   0.253  0.034      17       53
##
## Quantiles
##   tvalues  2.5%    5%    50%   95% 97.5%
## 1 -0.8493 -1.23 -0.99 -0.351 0.273  0.40
## 2  0.9616  0.69  0.75  0.880 0.975  0.98
## 3  0.1663 -0.38 -0.33 -0.028 0.457  0.84
## 4  0.0042 -0.62 -0.53 -0.239 0.097  0.59
##    based on 900 valid draws (burn-in=100)
```


0
500
1000
1500
2000
y
y rep


150000
100000
50000
0
var
0
50
100
150
mean
T = (mean, var)
T(y)
T(y rep)



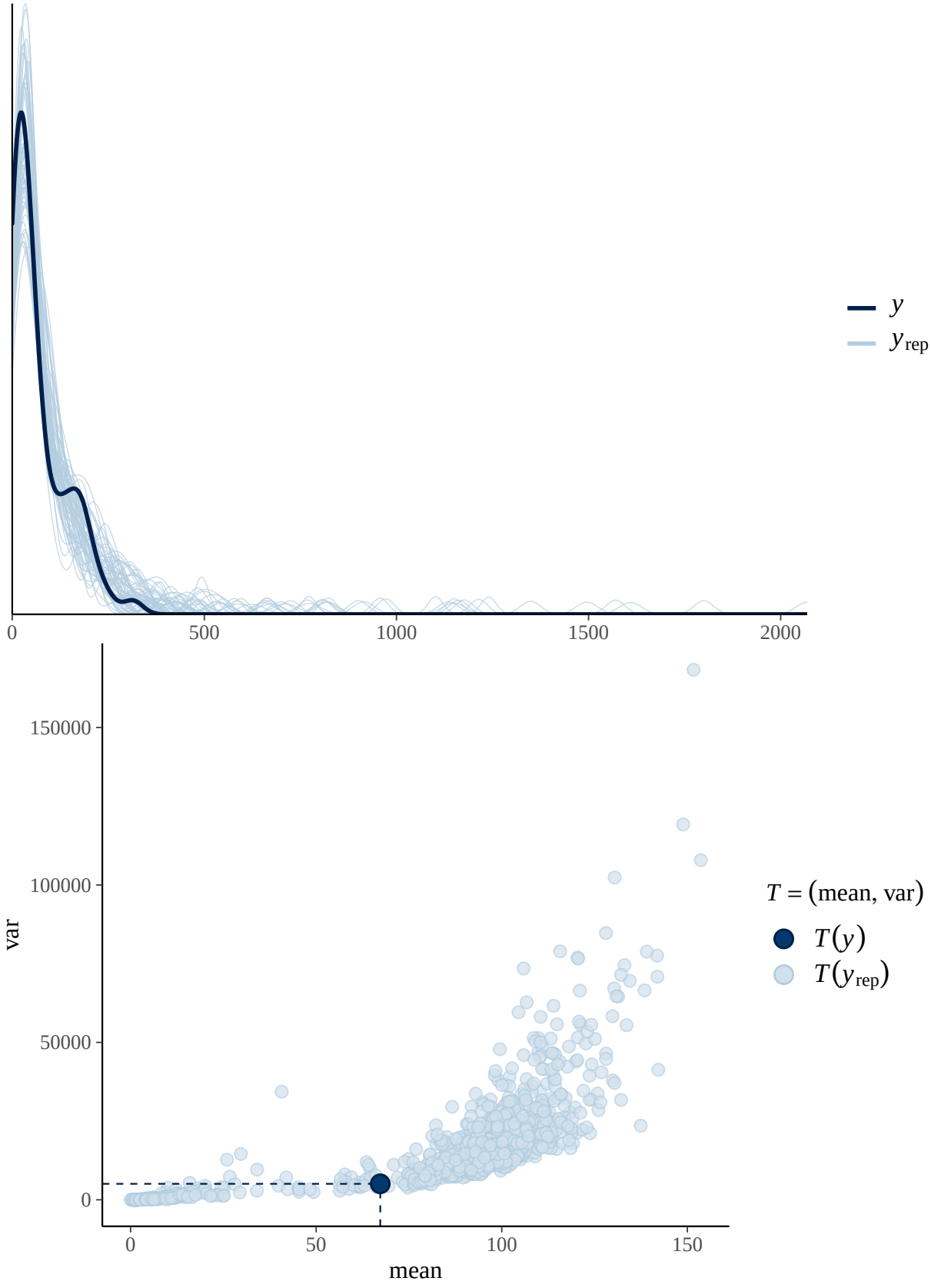

```
## Using default s_alpha = 2.93
## Using default s_beta = 2.93/sqrt(nvar)
##
## Starting Random Walk Metropolis Sampler for Negative Binomial Regression
##    69  obs;  4  covariates (including intercept);
## Prior Parameters:
## betabar
## [1] 0 0 0 0
## A
##      [,1] [,2] [,3] [,4]
## [1,] 0.01 0.00 0.00 0.00
## [2,] 0.00 0.01 0.00 0.00
## [3,] 0.00 0.00 0.01 0.00
## [4,] 0.00 0.00 0.00 0.01
## a
## [1] 0.5
## b
## [1] 0.1
##
## MCMC Parms:
## R=  1000  keep=  1  nprint=  100
## s_alpha =  2.38
## s_beta =  1.19
##
##  Initializing RW Increment Covariance Matrix...
## beta_mle =  0.4576326 0.5854973 0.3810144 -0.2822934
```

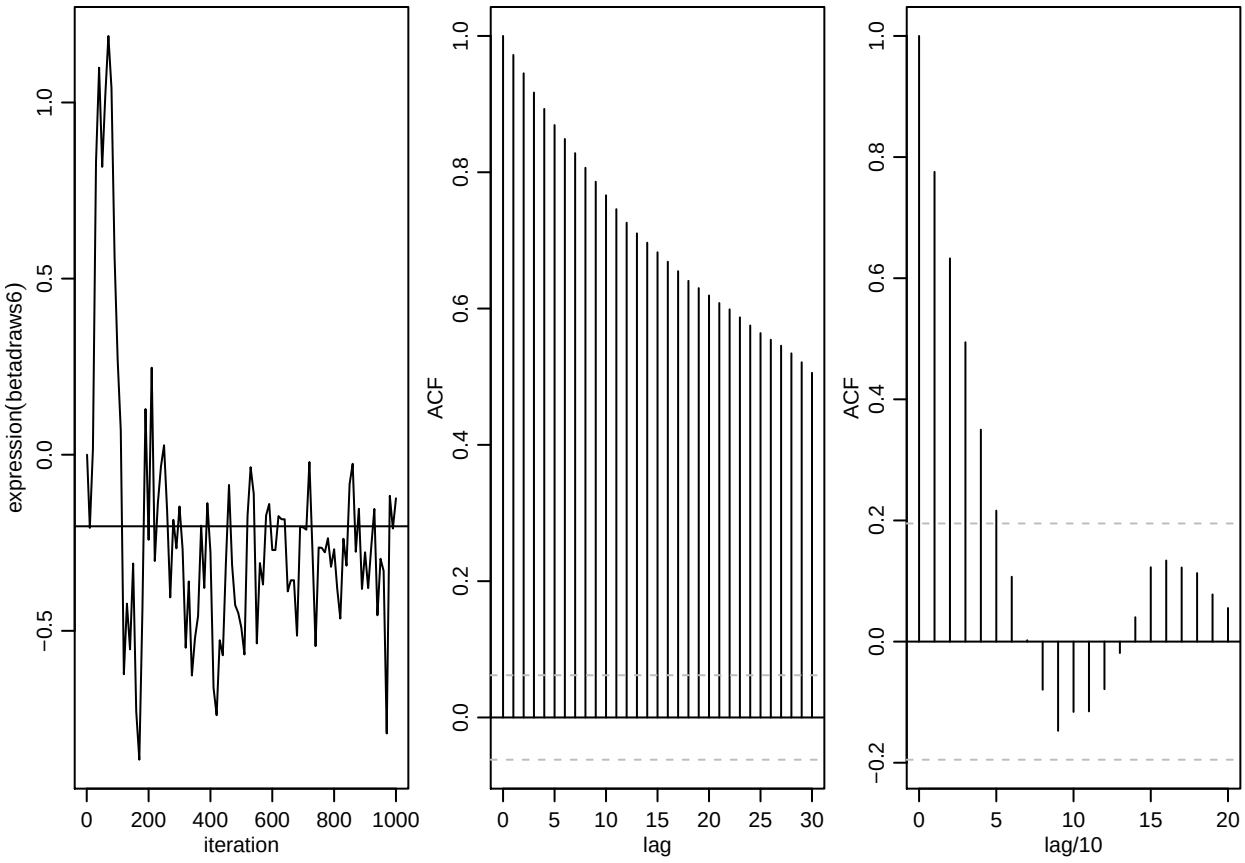
```
## alpha_mle =  5.867237
##  MCMC Iteration (est time to end - min)
##  100 (0.0)
##  200 (0.0)
##  300 (0.0)
##  400 (0.0)
##  500 (0.0)
##  600 (0.0)
##  700 (0.0)
##  800 (0.0)
##  900 (0.0)
##  1000 (0.0)
##  Total Time Elapsed: 0.00
```

```
## Summary of alpha/beta draw
```

```
## Summary of Posterior Marginal Distributions
## Moments
##   tvalues mean std dev num se rel eff sam size
## 1       5  4.8     1.8   0.31      26       33
##
## Quantiles
##   tvalues 2.5%   5% 50% 95% 97.5%
## 1       5  0.3 0.61   5 7.3   7.9
##    based on 900 valid draws (burn-in=100)
```

```
## Summary of Posterior Marginal Distributions
## Moments
##   tvalues  mean std dev num se rel eff sam size
## 1   0.085  0.36   0.528 0.0789      20       43
## 2   0.679  0.58   0.053 0.0058      11       82
## 3   0.155  0.35   0.245 0.0359      19       45
## 4  -0.383 -0.30   0.211 0.0238      11       75
##
## Quantiles
##   tvalues  2.5%    5%   50%     95% 97.5%
## 1   0.085 -0.97 -0.59  0.46 0.95345  1.10
## 2   0.679  0.48  0.50  0.58 0.67975  0.69
## 3   0.155 -0.27 -0.14  0.37 0.62771  0.74
## 4  -0.383 -0.73 -0.66 -0.29 0.00078  0.12
##    based on 900 valid draws (burn-in=100)
```


0
500
1000
1500
2000
y
y rep
150000
100000
50000
0
var
0
50
100
150
mean
T = (mean, var)
T(y)
T(y rep)



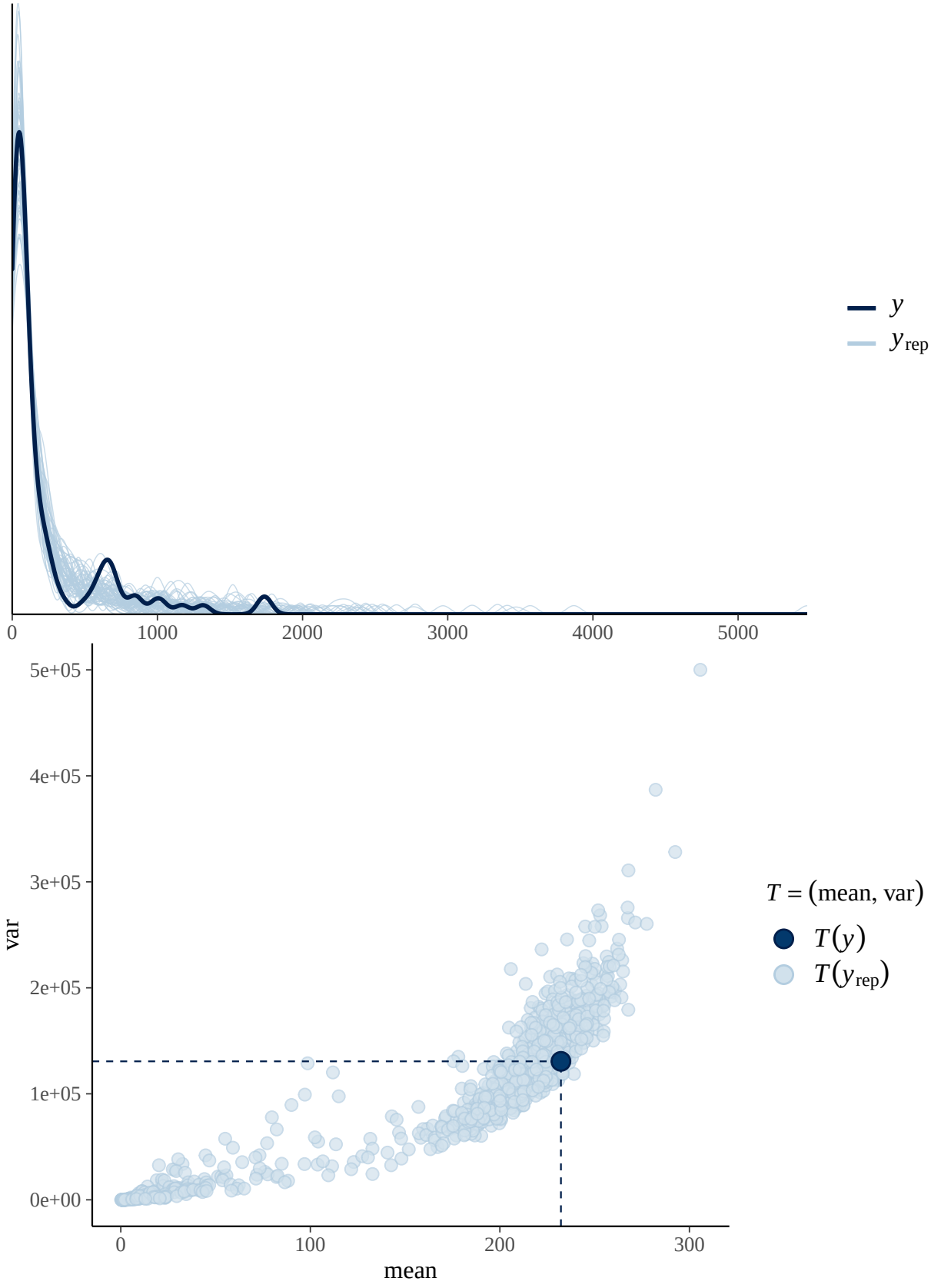

```
## Using default s_alpha = 2.93
## Using default s_beta = 2.93/sqrt(nvar)
##
## Starting Random Walk Metropolis Sampler for Negative Binomial Regression
##    96  obs;  4  covariates (including intercept);
## Prior Parameters:
## betabar
## [1] 0 0 0 0
## A
##      [,1] [,2] [,3] [,4]
## [1,] 0.01 0.00 0.00 0.00
## [2,] 0.00 0.01 0.00 0.00
## [3,] 0.00 0.00 0.01 0.00
## [4,] 0.00 0.00 0.00 0.01
## a
## [1] 0.5
## b
## [1] 0.1
##
## MCMC Parms:
## R=  1000  keep=  1  nprint=  100
## s_alpha =  2.38
## s_beta =  1.19
##
##  Initializing RW Increment Covariance Matrix...
## beta_mle =  0.9277311 0.9371275 -0.07766996 -0.6674303
```

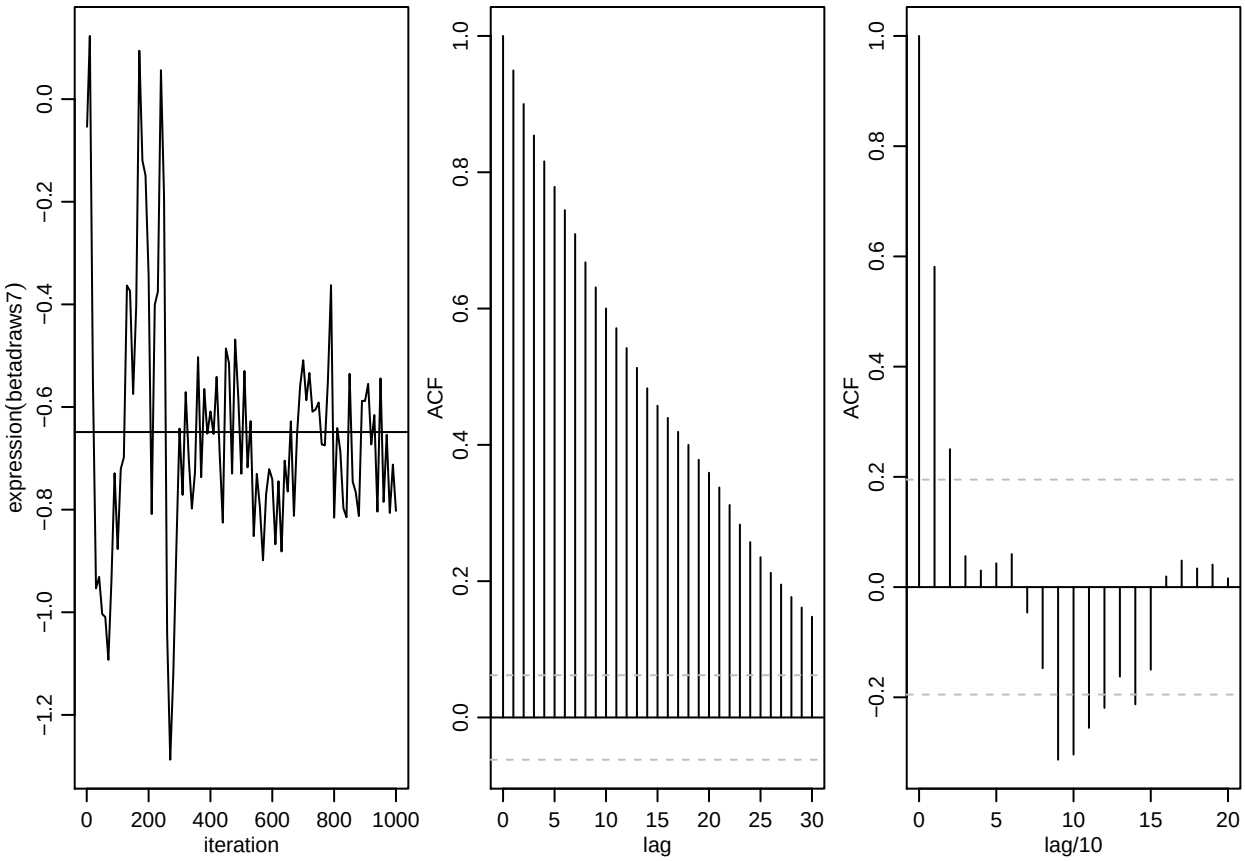
```
## alpha_mle =  6.341949
##  MCMC Iteration (est time to end - min)
##  100 (0.0)
##  200 (0.0)
##  300 (0.0)
##  400 (0.0)
##  500 (0.0)
##  600 (0.0)
##  700 (0.0)
##  800 (0.0)
##  900 (0.0)
##  1000 (0.0)
##  Total Time Elapsed: 0.00
```

```
## Summary of alpha/beta draw
```

```
## Summary of Posterior Marginal Distributions
## Moments
##   tvalues mean std dev num se rel eff sam size
## 1       5  4.8     2.4   0.48      37       24
##
## Quantiles
##   tvalues  2.5%    5% 50% 95% 97.5%
## 1       5 0.047 0.061 5.6 7.5   7.9
##    based on 900 valid draws (burn-in=100)
```

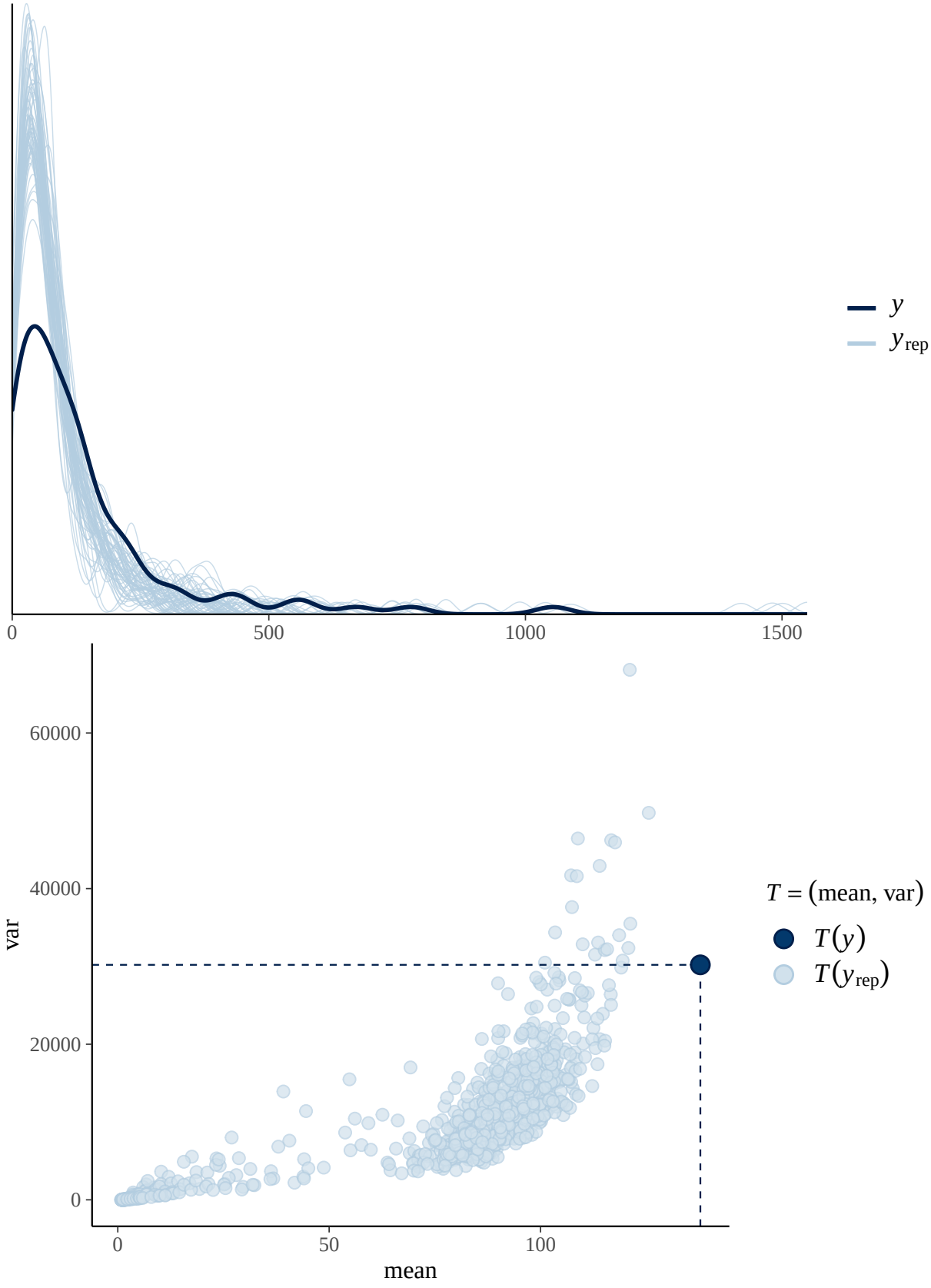
```
## Summary of Posterior Marginal Distributions
## Moments
##    tvalues   mean std dev num se rel eff sam size
## 1  8.4e-01  0.582   0.882  0.172      34       26
## 2  9.3e-01  0.922   0.073  0.010      18       50
## 3 -8.7e-05 -0.088   0.187  0.025      16       53
## 4 -5.2e-01 -0.636   0.223  0.036      23       39
##
## Quantiles
##    tvalues  2.5%    5%    50%   95%  97.5%
## 1  8.4e-01 -1.99 -1.72  0.836  1.35  1.509
## 2  9.3e-01  0.72  0.76  0.931  1.02  1.035
## 3 -8.7e-05 -0.42 -0.37 -0.092  0.26  0.336
## 4 -5.2e-01 -1.02 -0.90 -0.655 -0.18 -0.078
##    based on 900 valid draws (burn-in=100)
```

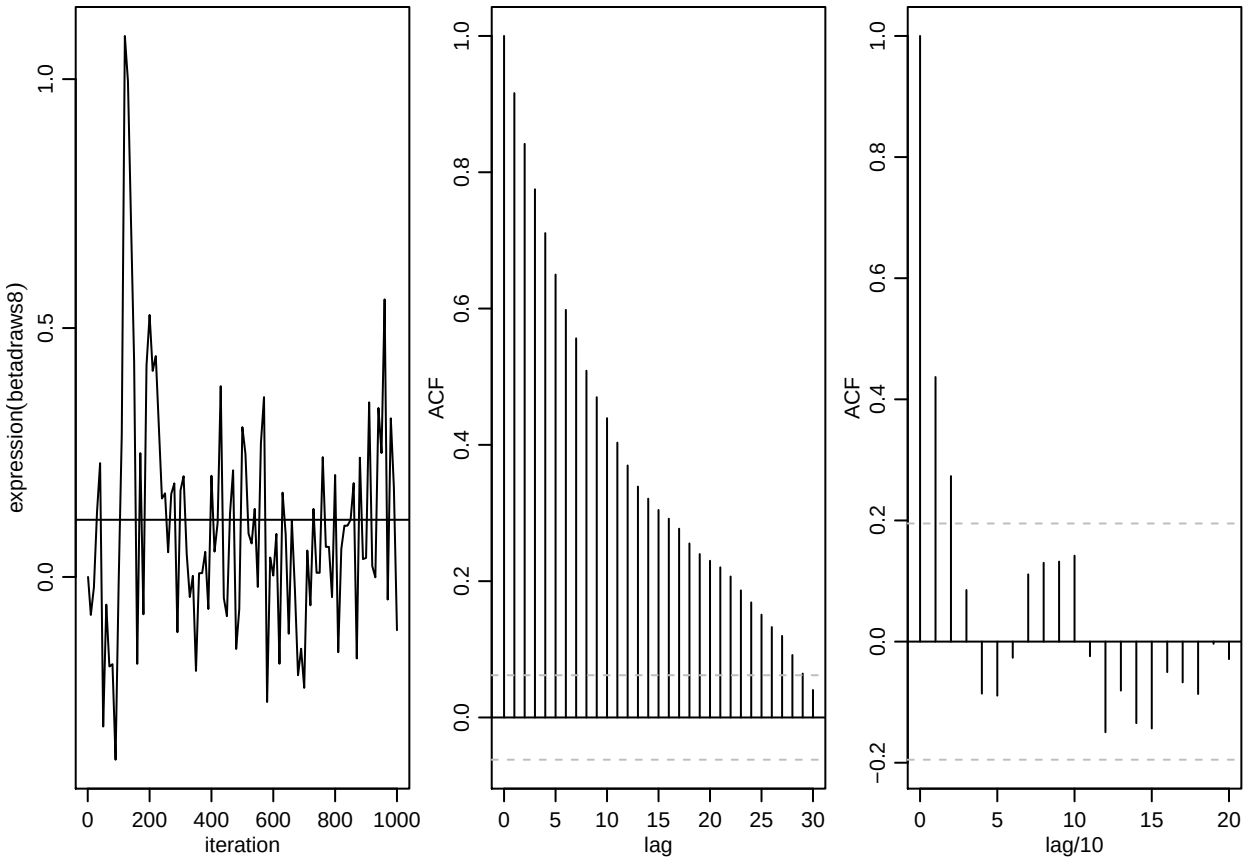

y
y_rep
0
1000
2000
3000
4000
5000
5e+05
4e+05
3e+05
2e+05
1e+05
0e+00
var
0
100
200
300
mean
T = (mean, var)
T(y)
T(y_rep)




```
## Using default s_alpha = 2.93
## Using default s_beta = 2.93/sqrt(nvar)
##
## Starting Random Walk Metropolis Sampler for Negative Binomial Regression
##    96  obs;  4  covariates (including intercept);
## Prior Parameters:
## betabar
## [1] 0 0 0 0
## A
##      [,1] [,2] [,3] [,4]
## [1,] 0.01 0.00 0.00 0.00
## [2,] 0.00 0.01 0.00 0.00
## [3,] 0.00 0.00 0.01 0.00
## [4,] 0.00 0.00 0.00 0.01
## a
## [1] 0.5
## b
## [1] 0.1
##
## MCMC Parms:
## R=  1000  keep=  1  nprint=  100
## s_alpha =  2.38
## s_beta =  1.19
##
##  Initializing RW Increment Covariance Matrix...
## beta_mle =  0.171047 0.7535497 0.6024082 0.1326003
```

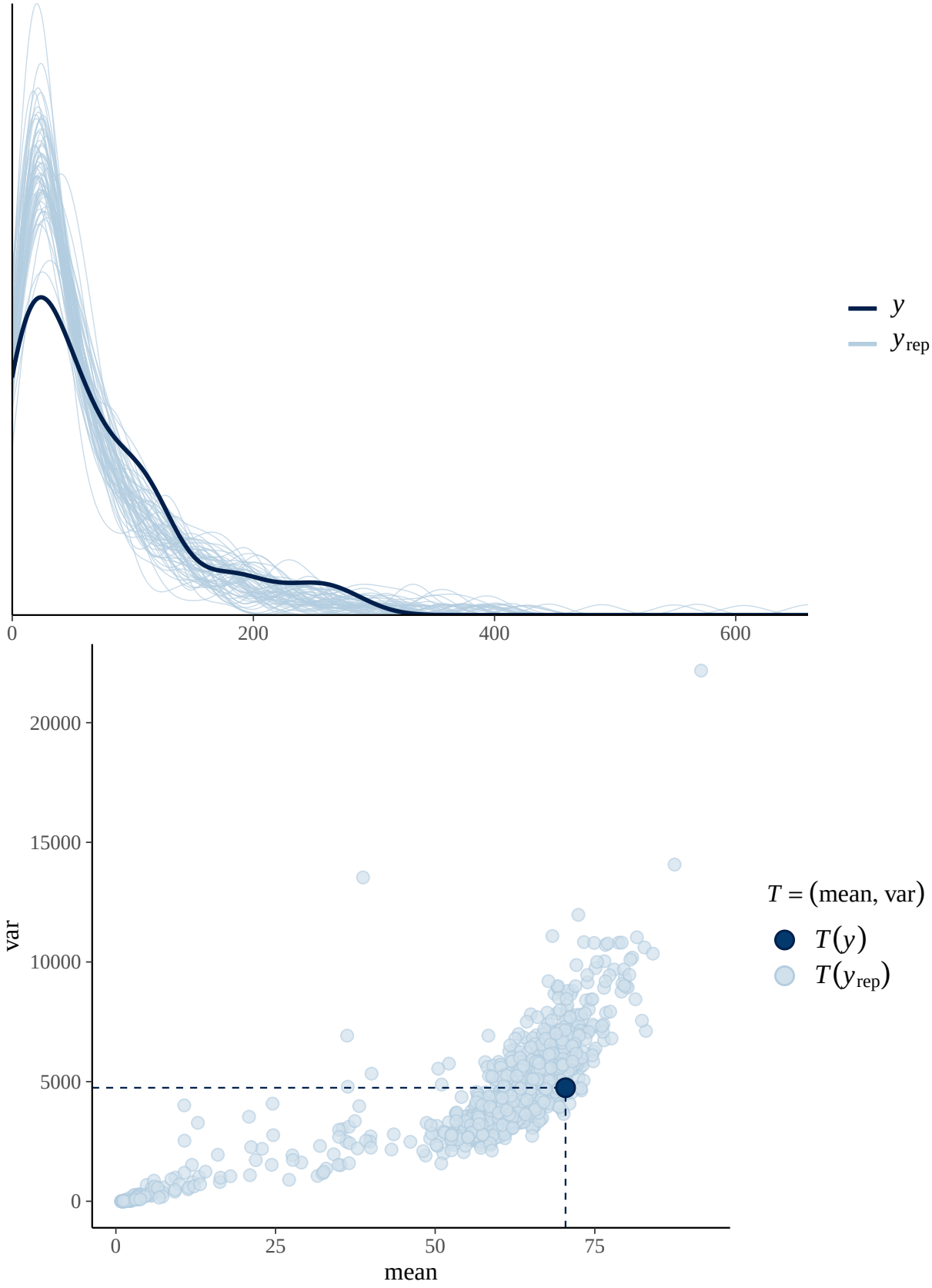
```
## alpha_mle =  4.706306
##  MCMC Iteration (est time to end - min)
##  100 (0.0)
##  200 (0.0)
##  300 (0.0)
##  400 (0.0)
##  500 (0.0)
##  600 (0.0)
##  700 (0.0)
##  800 (0.0)
##  900 (0.0)
##  1000 (0.0)
##  Total Time Elapsed: 0.00

## Summary of alpha/beta draw

## Summary of Posterior Marginal Distributions
## Moments
##   tvalues mean std dev num se rel eff sam size
## 1       5    4     1.5   0.27      30       30
##
## Quantiles
##   tvalues  2.5%   5% 50% 95% 97.5%
## 1       5 0.095 0.18 4.3 5.8   6.2
##    based on 900 valid draws (burn-in=100)

## Summary of Posterior Marginal Distributions
## Moments
##   tvalues  mean std dev num se rel eff sam size
## 1    0.21 0.019   0.684 0.1235      29       30
## 2    0.76 0.751   0.065 0.0074      12       75
## 3    0.48 0.597   0.189 0.0268      18       47
## 4    0.20 0.133   0.223 0.0322      19       47
##
## Quantiles
##   tvalues  2.5%    5%  50%  95% 97.5%
## 1    0.21 -2.24 -1.55 0.15 0.69  0.83
## 2    0.76  0.61  0.65 0.75 0.85  0.89
## 3    0.48  0.27  0.30 0.59 0.95  1.03
## 4    0.20 -0.17 -0.14 0.10 0.56  0.76
##    based on 900 valid draws (burn-in=100)
```

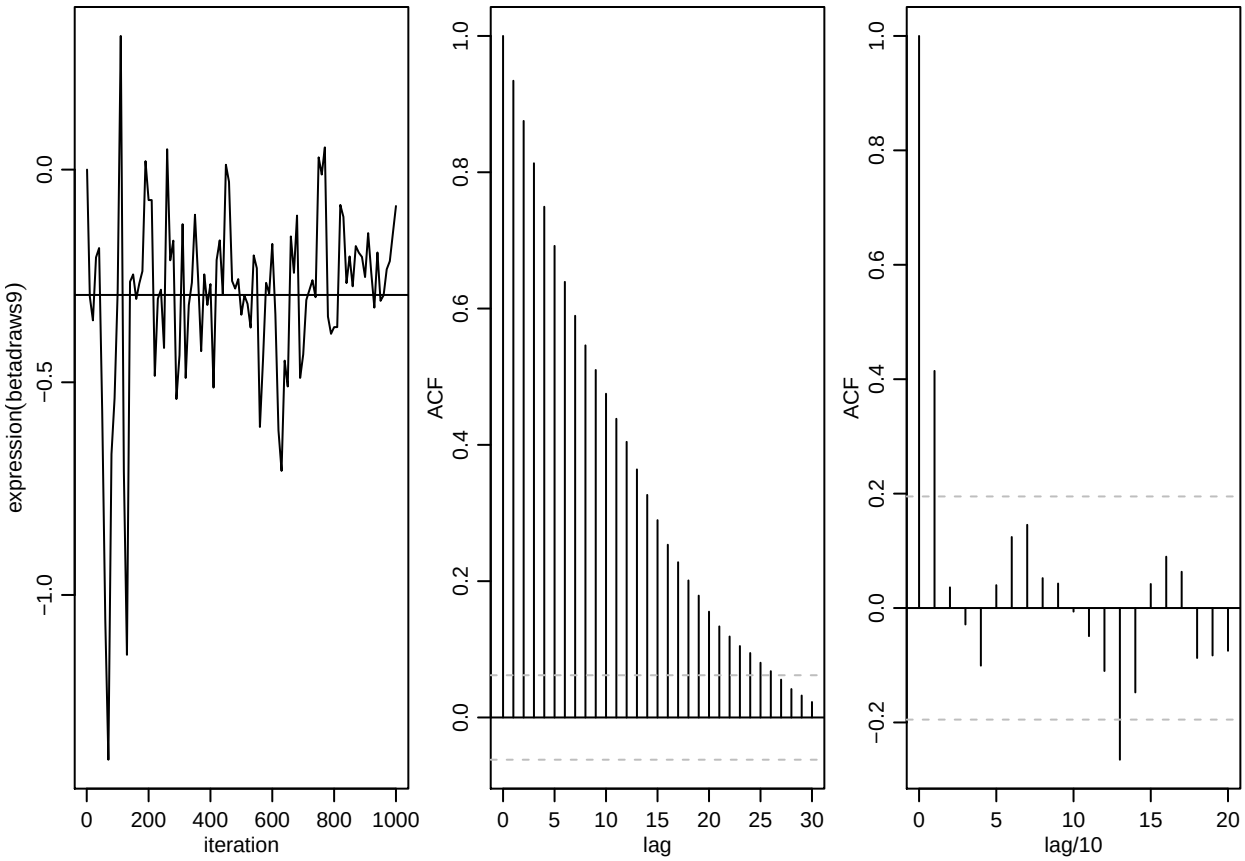

0
500
1000
1500
y
y_rep
60000
40000
20000
0
var
0
50
100
mean
T = (mean, var)
T(y)
T(y_rep)




```
## Using default s_alpha = 2.93
## Using default s_beta = 2.93/sqrt(nvar)
##
## Starting Random Walk Metropolis Sampler for Negative Binomial Regression
##    96  obs;  4  covariates (including intercept);
## Prior Parameters:
## betabar
## [1] 0 0 0 0
## A
##      [,1] [,2] [,3] [,4]
## [1,] 0.01 0.00 0.00 0.00
## [2,] 0.00 0.01 0.00 0.00
## [3,] 0.00 0.00 0.01 0.00
## [4,] 0.00 0.00 0.00 0.01
## a
## [1] 0.5
## b
## [1] 0.1
##
## MCMC Parms:
## R=  1000  keep=  1  nprint=  100
## s_alpha =  2.38
## s_beta =  1.19
##
##  Initializing RW Increment Covariance Matrix...
## beta_mle =  0.1915881 0.6428964 0.07613809 -0.2388481
```

```
## alpha_mle =  4.369725
##  MCMC Iteration (est time to end - min)
##  100 (0.0)
##  200 (0.0)
##  300 (0.0)
##  400 (0.0)
##  500 (0.0)
##  600 (0.0)
##  700 (0.0)
##  800 (0.0)
##  900 (0.0)
##  1000 (0.0)
##  Total Time Elapsed: 0.00

## Summary of alpha/beta draw

## Summary of Posterior Marginal Distributions
## Moments
##   tvalues mean std dev num se rel eff sam size
## 1       5  3.9    0.99   0.15      20       43
##
## Quantiles
##   tvalues 2.5%  5% 50% 95% 97.5%
## 1       5 0.77 1.6   4 5.2   5.4
##    based on 900 valid draws (burn-in=100)

## Summary of Posterior Marginal Distributions
## Moments
##   tvalues   mean std dev num se rel eff sam size
## 1  -0.260  0.136   0.310 0.0399      15       60
## 2   0.720  0.645   0.045 0.0051      11       75
## 3   0.093  0.056   0.152 0.0208      17       53
## 4  -0.130 -0.265   0.190 0.0220      12       69
##
## Quantiles
##   tvalues  2.5%    5%    50%   95% 97.5%
## 1  -0.260 -0.57 -0.39  0.144 0.636 0.697
## 2   0.720  0.56  0.58  0.645 0.714 0.729
## 3   0.093 -0.30 -0.19  0.076 0.256 0.282
## 4  -0.130 -0.66 -0.55 -0.260 0.016 0.048
##    based on 900 valid draws (burn-in=100)
```


0
200
400
600
y
y_rep
20000
15000
10000
5000
0
var
0
25
50
75
mean
T = (mean, var)
T(y)
T(y_rep)



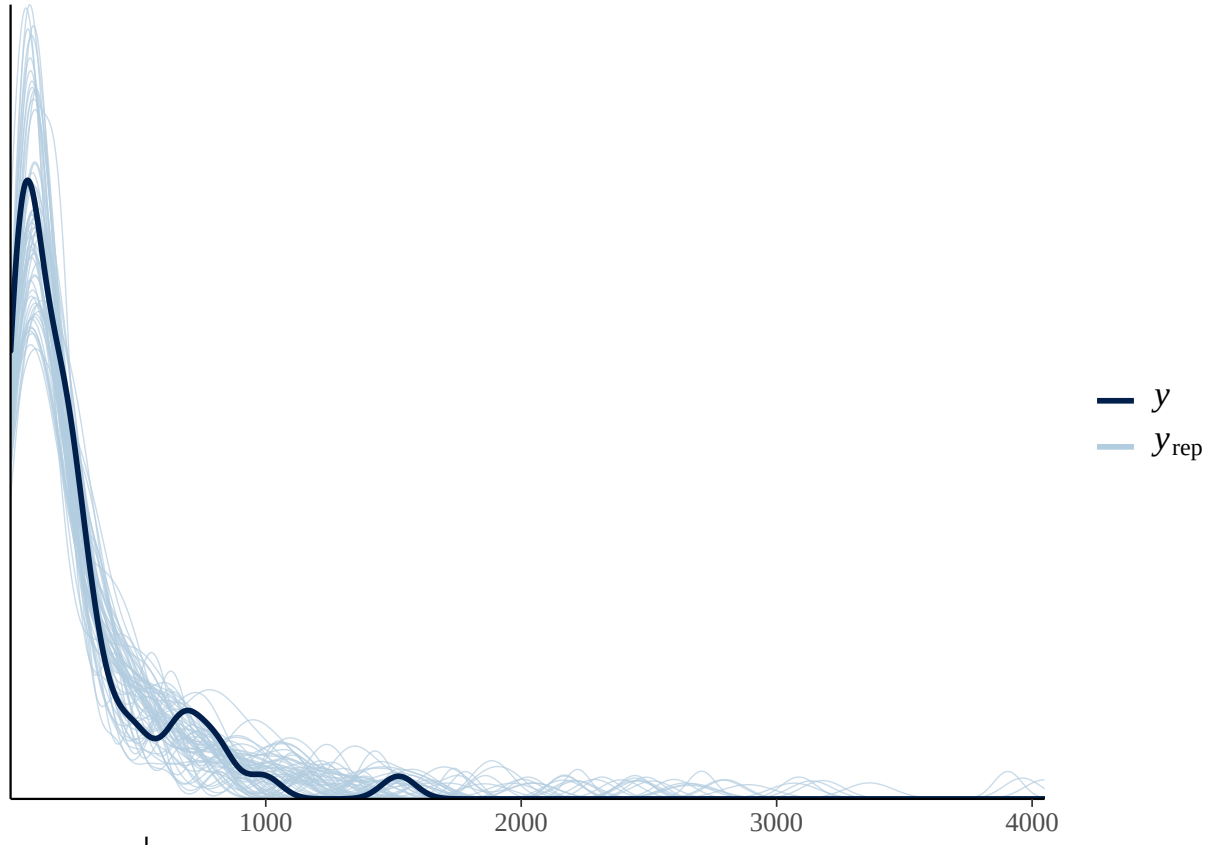

```
## Using default s_alpha = 2.93
## Using default s_beta = 2.93/sqrt(nvar)
##
## Starting Random Walk Metropolis Sampler for Negative Binomial Regression
##    60  obs;  4  covariates (including intercept);
## Prior Parameters:
## betabar
## [1] 0 0 0 0
## A
##      [,1] [,2] [,3] [,4]
## [1,] 0.01 0.00 0.00 0.00
## [2,] 0.00 0.01 0.00 0.00
## [3,] 0.00 0.00 0.01 0.00
## [4,] 0.00 0.00 0.00 0.01
## a
## [1] 0.5
## b
## [1] 0.1
##
## MCMC Parms:
## R=  1000  keep=  1  nprint=  100
## s_alpha =  2.38
## s_beta =  1.19
##
##  Initializing RW Increment Covariance Matrix...
## beta_mle =  0.4841367 1.050877 0.001844416 -0.5463842
```

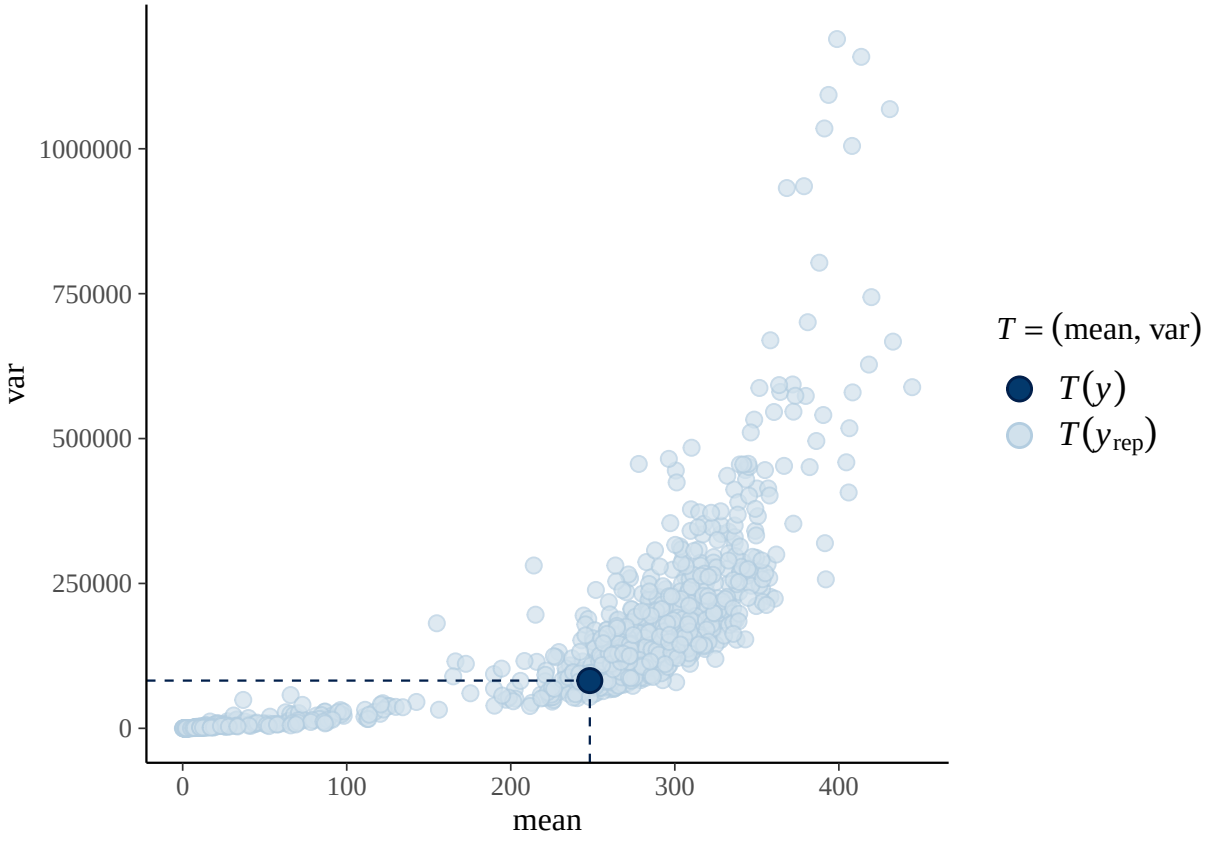
```
## alpha_mle =  4.266917
##  MCMC Iteration (est time to end - min)
##  100 (0.0)
##  200 (0.0)
##  300 (0.0)
##  400 (0.0)
##  500 (0.0)
##  600 (0.0)
##  700 (0.0)
##  800 (0.0)
##  900 (0.0)
##  1000 (0.0)
##  Total Time Elapsed: 0.00
```

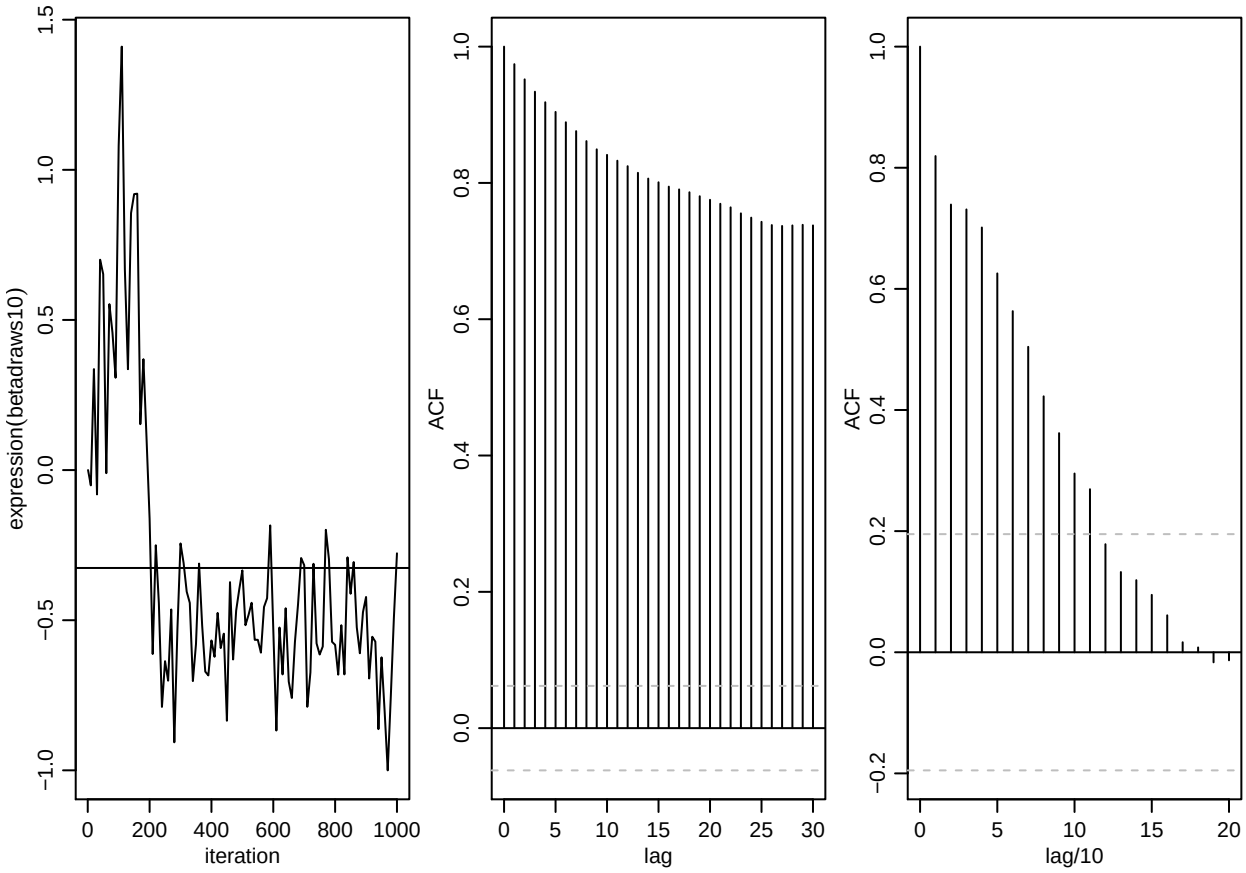
```
## Summary of alpha/beta draw
```

```
## Summary of Posterior Marginal Distributions
## Moments
##   tvalues mean std dev num se rel eff sam size
## 1       5  3.5     1.4   0.26      31       28
##
## Quantiles
##   tvalues  2.5%   5% 50% 95% 97.5%
## 1       5 0.085 0.16 3.8 5.1   5.4
##    based on 900 valid draws (burn-in=100)
```

```
## Summary of Posterior Marginal Distributions
## Moments
##   tvalues    mean std dev num se rel eff sam size
## 1   0.983  0.5927    0.45  0.053      13       69
## 2   0.940  0.9783    0.17  0.033      33       27
## 3  -0.096 -0.0099    0.18  0.021      12       69
## 4  -0.605 -0.4084    0.42  0.078      31       29
##
## Quantiles
##   tvalues  2.5%    5%     50%  95% 97.5%
## 1   0.983 -0.26 -0.14  0.5916 1.39  1.57
## 2   0.940  0.46  0.56  1.0174 1.16  1.21
## 3  -0.096 -0.38 -0.30 -0.0056 0.26  0.34
## 4  -0.605 -0.86 -0.79 -0.5243 0.71  0.96
##    based on 900 valid draws (burn-in=100)
```


1000
2000
3000
4000
y
y rep


1000000
750000
500000
250000
0
var
0
100
200
300
400
mean
T = (mean, var)
T(y)
T(y rep)



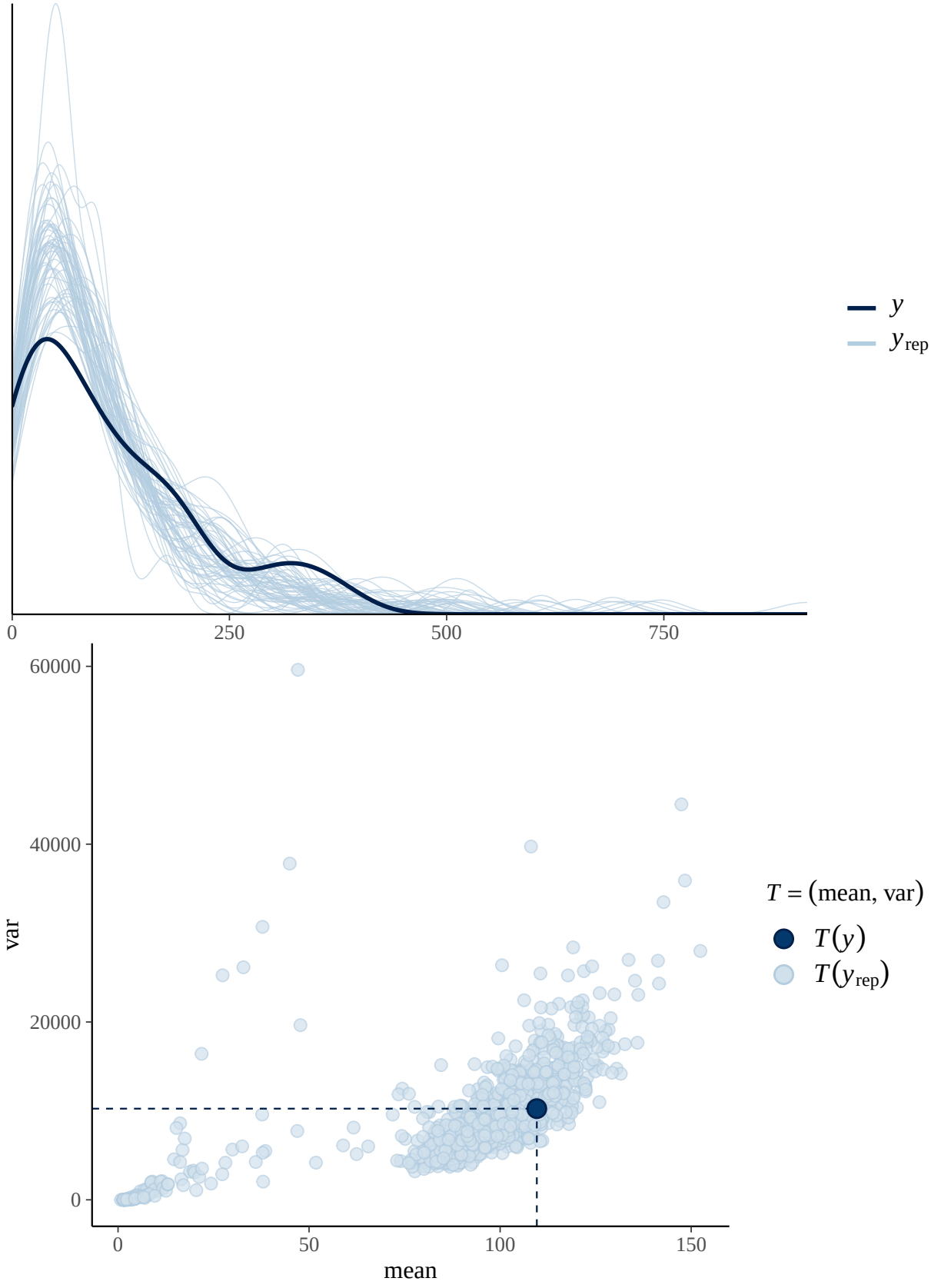

```
## Using default s_alpha = 2.93
## Using default s_beta = 2.93/sqrt(nvar)
##
## Starting Random Walk Metropolis Sampler for Negative Binomial Regression
##    60  obs;  4  covariates (including intercept);
## Prior Parameters:
## betabar
## [1] 0 0 0 0
## A
##      [,1] [,2] [,3] [,4]
## [1,] 0.01 0.00 0.00 0.00
## [2,] 0.00 0.01 0.00 0.00
## [3,] 0.00 0.00 0.01 0.00
## [4,] 0.00 0.00 0.00 0.01
## a
## [1] 0.5
## b
## [1] 0.1
##
## MCMC Parms:
## R=  1000  keep=  1  nprint=  100
## s_alpha =  2.38
## s_beta =  1.19
##
##  Initializing RW Increment Covariance Matrix...
## beta_mle =  -0.1683736 0.8602028 0.1906381 0.3314953
```

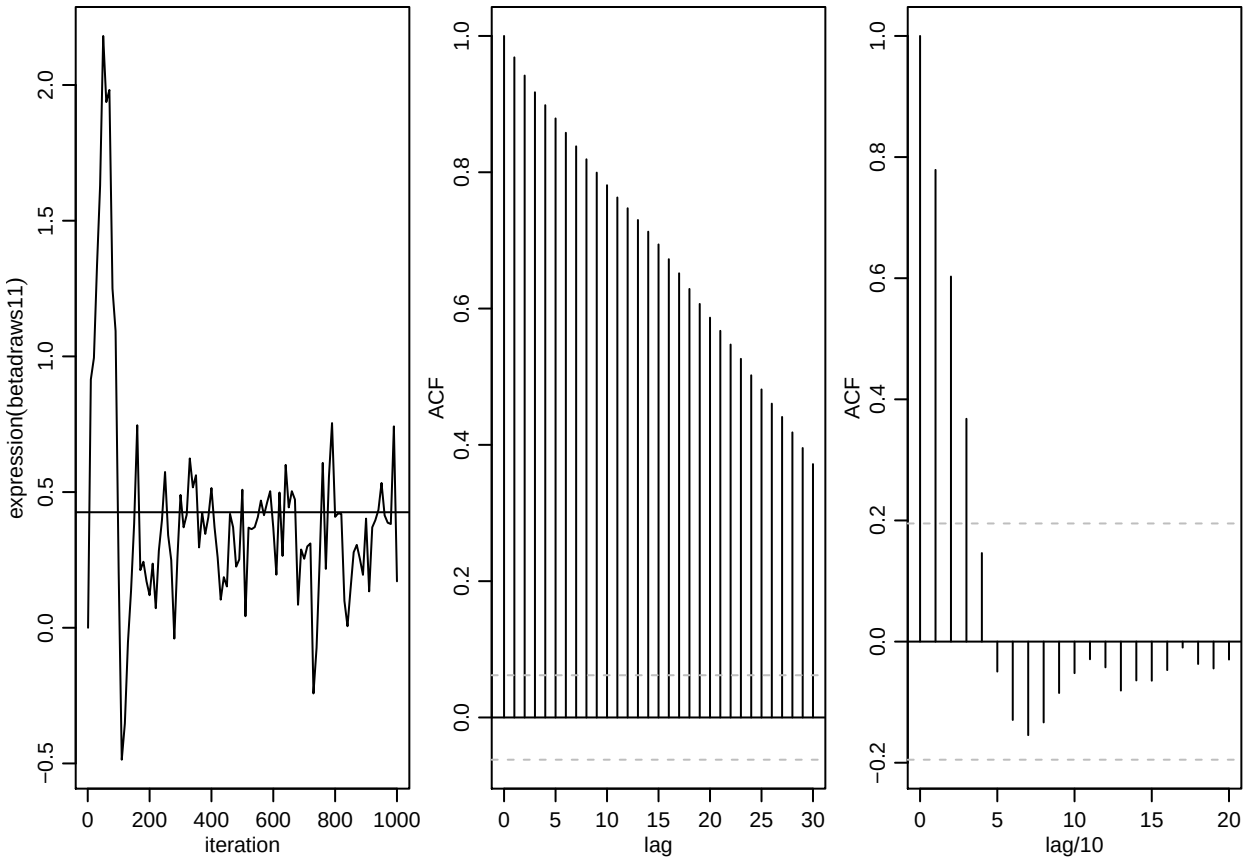
```
## alpha_mle =  4.650683
##  MCMC Iteration (est time to end - min)
##  100 (0.0)
##  200 (0.0)
##  300 (0.0)
##  400 (0.0)
##  500 (0.0)
##  600 (0.0)
##  700 (0.0)
##  800 (0.0)
##  900 (0.0)
##  1000 (0.0)
##  Total Time Elapsed: 0.00
```

```
## Summary of alpha/beta draw
```

```
## Summary of Posterior Marginal Distributions
## Moments
##   tvalues mean std dev num se rel eff sam size
## 1       5  4.1       1   0.11      12       75
##
## Quantiles
##   tvalues 2.5%  5% 50% 95% 97.5%
## 1       5  1.5 2.8 4.1 5.8   6.2
##    based on 900 valid draws (burn-in=100)
```

```
## Summary of Posterior Marginal Distributions
## Moments
##   tvalues  mean std dev num se rel eff sam size
## 1    0.22 -0.23    0.41 0.0539    15.2       56
## 2    0.78  0.87    0.07 0.0089    14.8       60
## 3    0.15  0.21    0.16 0.0162     8.8      100
## 4    0.38  0.32    0.21 0.0285    16.1       53
##
## Quantiles
##   tvalues  2.5%     5%   50%  95% 97.5%
## 1    0.22 -1.18 -0.854 -0.22 0.40  0.51
## 2    0.78  0.74  0.763  0.86 0.98  1.02
## 3    0.15 -0.11 -0.051  0.21 0.48  0.54
## 4    0.38 -0.14 -0.054  0.35 0.61  0.65
##    based on 900 valid draws (burn-in=100)
```


0
250
500
750
60000
40000
20000
0
var
0
50
100
150
mean
y
y rep
T = (mean, var)
T(y)
T(y rep)



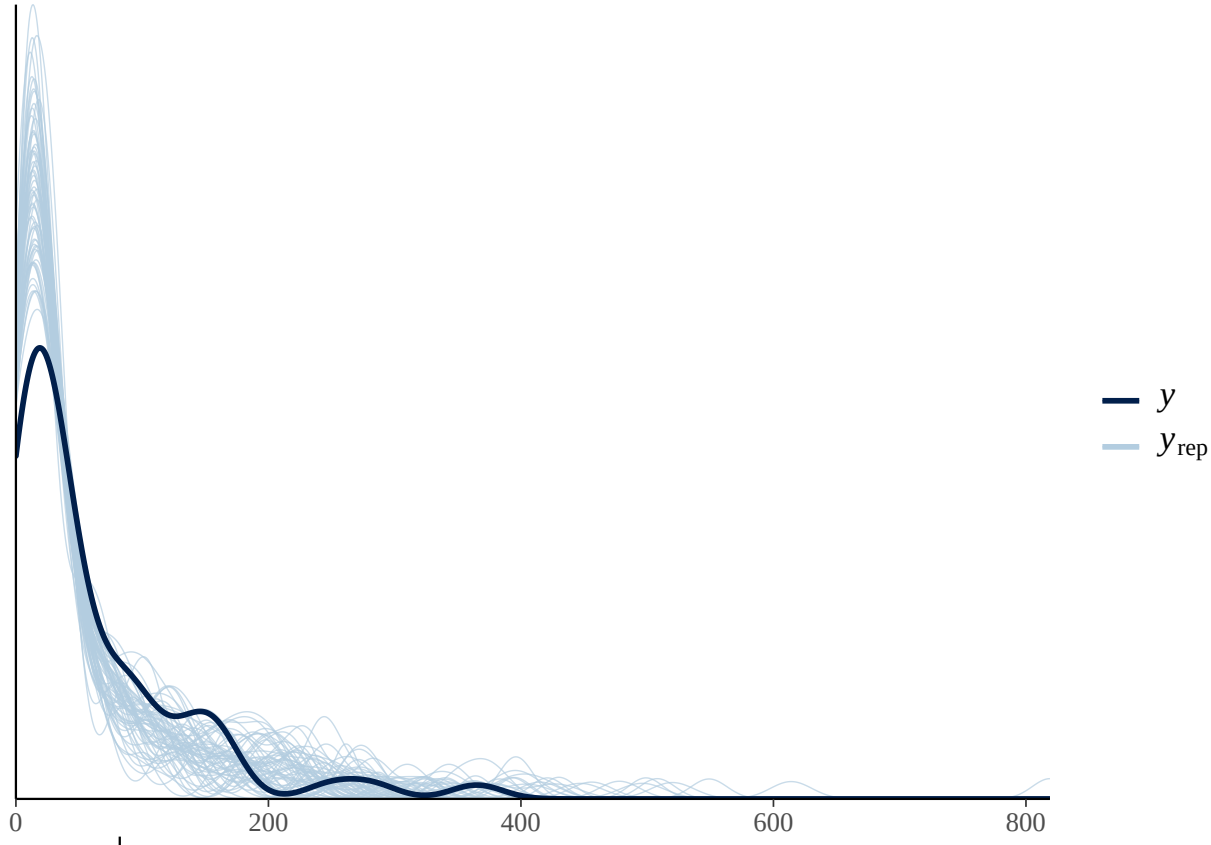

```
## Using default s_alpha = 2.93
## Using default s_beta = 2.93/sqrt(nvar)
##
## Starting Random Walk Metropolis Sampler for Negative Binomial Regression
##    60  obs;  4  covariates (including intercept);
## Prior Parameters:
## betabar
## [1] 0 0 0 0
## A
##      [,1] [,2] [,3] [,4]
## [1,] 0.01 0.00 0.00 0.00
## [2,] 0.00 0.01 0.00 0.00
## [3,] 0.00 0.00 0.01 0.00
## [4,] 0.00 0.00 0.00 0.01
## a
## [1] 0.5
## b
## [1] 0.1
##
## MCMC Parms:
## R=  1000  keep=  1  nprint=  100
## s_alpha =  2.38
## s_beta =  1.19
##
##  Initializing RW Increment Covariance Matrix...
## beta_mle =  -0.2179628 0.7337058 -0.3181977 -0.8984002
```

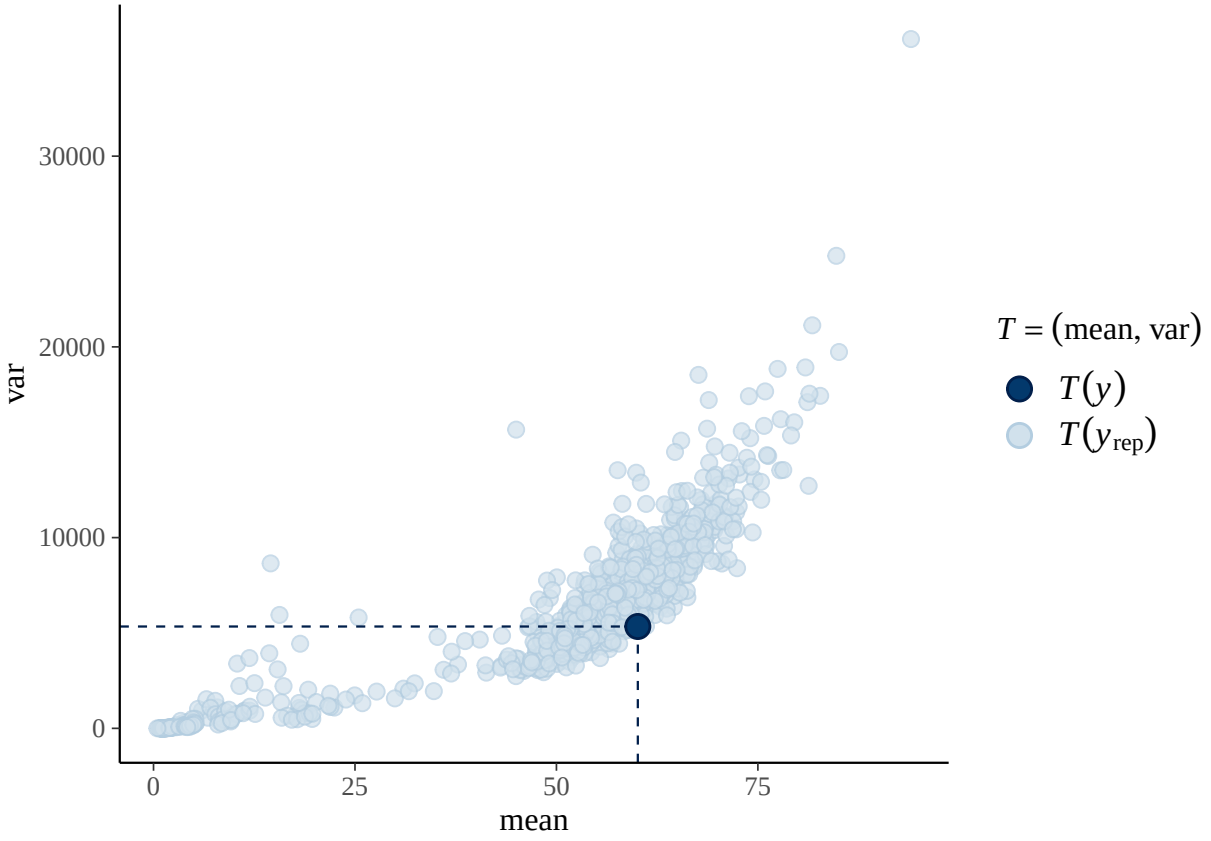
```
## alpha_mle =  7.57096
##  MCMC Iteration (est time to end - min)
##  100 (0.0)
##  200 (0.0)
##  300 (0.0)
##  400 (0.0)
##  500 (0.0)
##  600 (0.0)
##  700 (0.0)
##  800 (0.0)
##  900 (0.0)
##  1000 (0.0)
##  Total Time Elapsed: 0.00
```

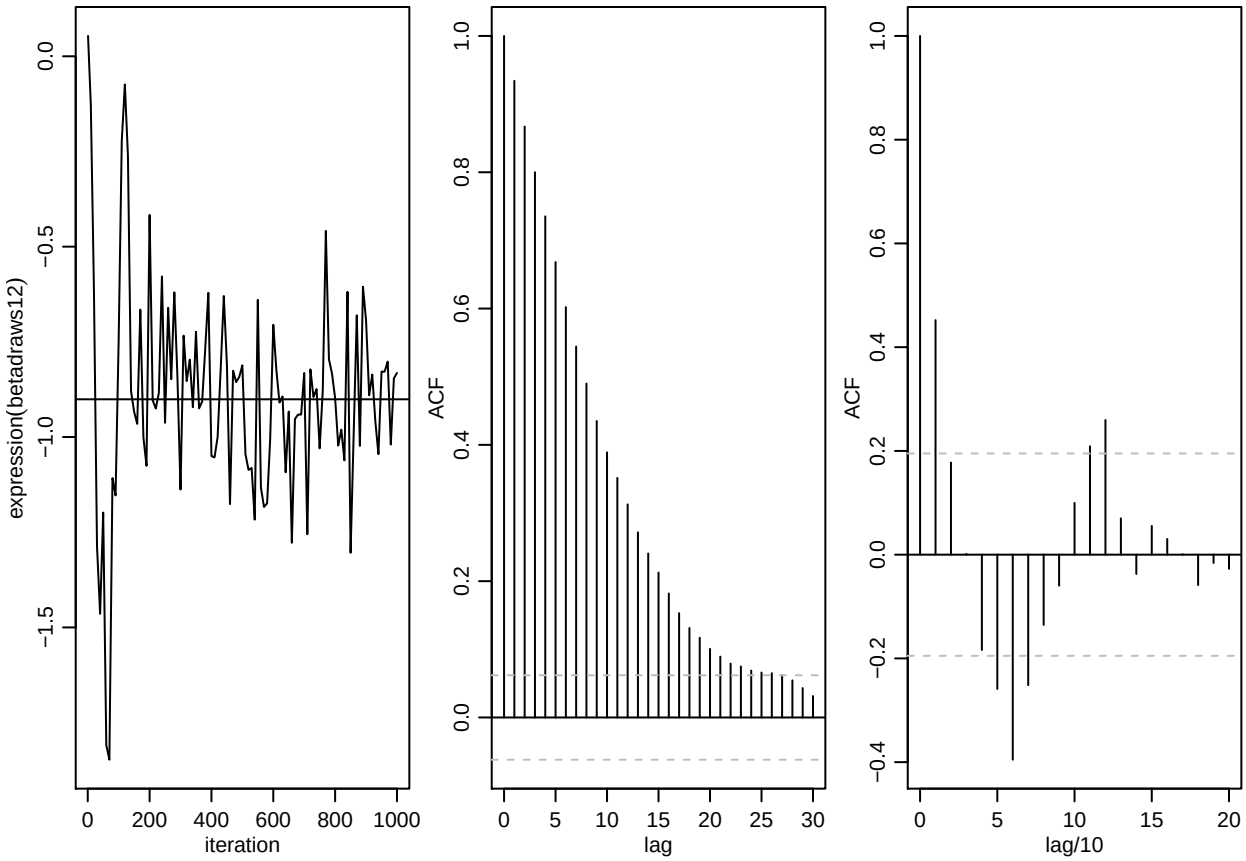
```
## Summary of alpha/beta draw
```

```
## Summary of Posterior Marginal Distributions
## Moments
##   tvalues mean std dev num se rel eff sam size
## 1       5    6     2.2   0.33      21       41
##
## Quantiles
##   tvalues 2.5%  5% 50% 95% 97.5%
## 1       5 0.29 1.2 6.1 9.4   9.9
##    based on 900 valid draws (burn-in=100)
```

```
## Summary of Posterior Marginal Distributions
## Moments
##   tvalues  mean std dev num se rel eff sam size
## 1  -0.372 -0.32   0.464 0.0545    12.4       69
## 2   0.742  0.73   0.068 0.0099    18.9       47
## 3  -0.093 -0.29   0.181 0.0182     9.1       90
## 4  -0.594 -0.88   0.206 0.0265    14.9       60
##
## Quantiles
##   tvalues  2.5%    5%   50%    95%  97.5%
## 1  -0.372 -1.28 -1.02 -0.30  0.320  0.592
## 2   0.742  0.54  0.63  0.74  0.833  0.853
## 3  -0.093 -0.67 -0.60 -0.28 -0.015  0.085
## 4  -0.594 -1.26 -1.21 -0.89 -0.524 -0.327
##    based on 900 valid draws (burn-in=100)
```


0
200
400
600
800
y
y_{rep}


30000
20000
10000
0
var
0
25
50
75
mean
$T = (\text{mean, var})$
$T(y)$
$T(y_{\text{rep}})$


expression(betadraws12)
0.0
−0.5
−1.0
−1.5
0
200
400
600
800
1000
iteration
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
lag
ACF
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
0
5
10
15
20
lag/10