Computational Statistics & Probability

Problem Set 4 - HMC and Generalized Linear Models
Due: 23:59:59 7.dec.2022

Fall 2022

Instructions

Assignments must be submitted through Canvas. See the course Canvas page for policies covering collaboration, acceptable file formats (`.Rmd` & `.pdf`), and late submissions. Completed assignments must include executable code (`.Rmd`) and a corresponding knitted markdown file (`pdf`). An R Markdown cheat sheet is available.

Note: *You must have* `rstan` *installed to complete this assignment.*

1. Log-odds

a) If an event has probability 0.3, what are the log-odds of this event?

```
p <- 0.3
log( p / (1- p ) )
```

```
## [1] -0.8472979
```

b) If an event has log-odds of 1, what is the probability of that event?

```
# use `logistic()`
logistic( 1 )
```

```
## [1] 0.7310586
```

c) If a logistic regression coefficient has value -0.70, what does this imply about the proportional change in odds of the outcome? Briefly explain your answer.

```
# use `exp()`
exp( -0.70 )
```

```
## [1] 0.4965853
```

```
# This means that each unit change in the predictor variable multiplies the odds
# of the event by roughly 1/2, or exp(-0.70) = 0.4965853.
```

2. HMC, Interactions and Robust Priors

Recall the interaction model `m8.3`, which is a varying-slope regression model assessing the effect of a country being inside or outside Africa on relationship between the ruggedness of its terrain and its GDP.

```
m8.3 <- quap(
  alist(
    log_gdp_std ~ dnorm(mu, sigma),
    mu <- a[cid] + b[cid]*(rugged_std - 0.215) ,
    a[cid] ~ dnorm(1, 0.1),
    b[cid] ~ dnorm(0, 0.3),
    sigma ~ dexp(1)
  ), data = dd)  # See R code 9.11 to prepare dd
```

```
library(rethinking)
data(rugged)
d <- rugged
d$log_gdp <- log(d$rgdppc_2000)
dd <- d[ complete.cases(d$rgdppc_2000) , ]
dd$log_gdp_std <- dd$log_gdp / mean(dd$log_gdp)
dd$rugged_std <- dd$rugged / max(dd$rugged)
dd$cid <- ifelse( dd$cont_africa==1, 1, 2 )
m8.3 <- quap(
  alist(
    log_gdp_std ~ dnorm(mu, sigma),
    mu <- a[cid] + b[cid]*(rugged_std - 0.215) ,
    a[cid] ~ dnorm(1, 0.1),
    b[cid] ~ dnorm(0, 0.3),
    sigma ~ dexp(1)
  ), data = dd)
```

a) Now fit this same model using Hamiltonian Monte Carlo (HMC). The code to do this is in the book, beginning with `R code 9.13`. You should use the `ulam` convenience function provided by the `rethinking` package.

```
dat_slim <- list(
  log_gdp_std = dd$log_gdp_std,
  rugged_std = dd$rugged_std,
  cid = as.integer( dd$cid )
)

m9.1 <- ulam(
  alist(
    log_gdp_std ~ dnorm(mu, sigma),
    mu <- a[cid] + b[cid]*(rugged_std - 0.215) ,
    a[cid] ~ dnorm(1, 0.1),
    b[cid] ~ dnorm(0, 0.3),
    sigma ~ dexp(1)
  ), data=dat_slim, chains=4)
```

```
show( m9.1 )
```

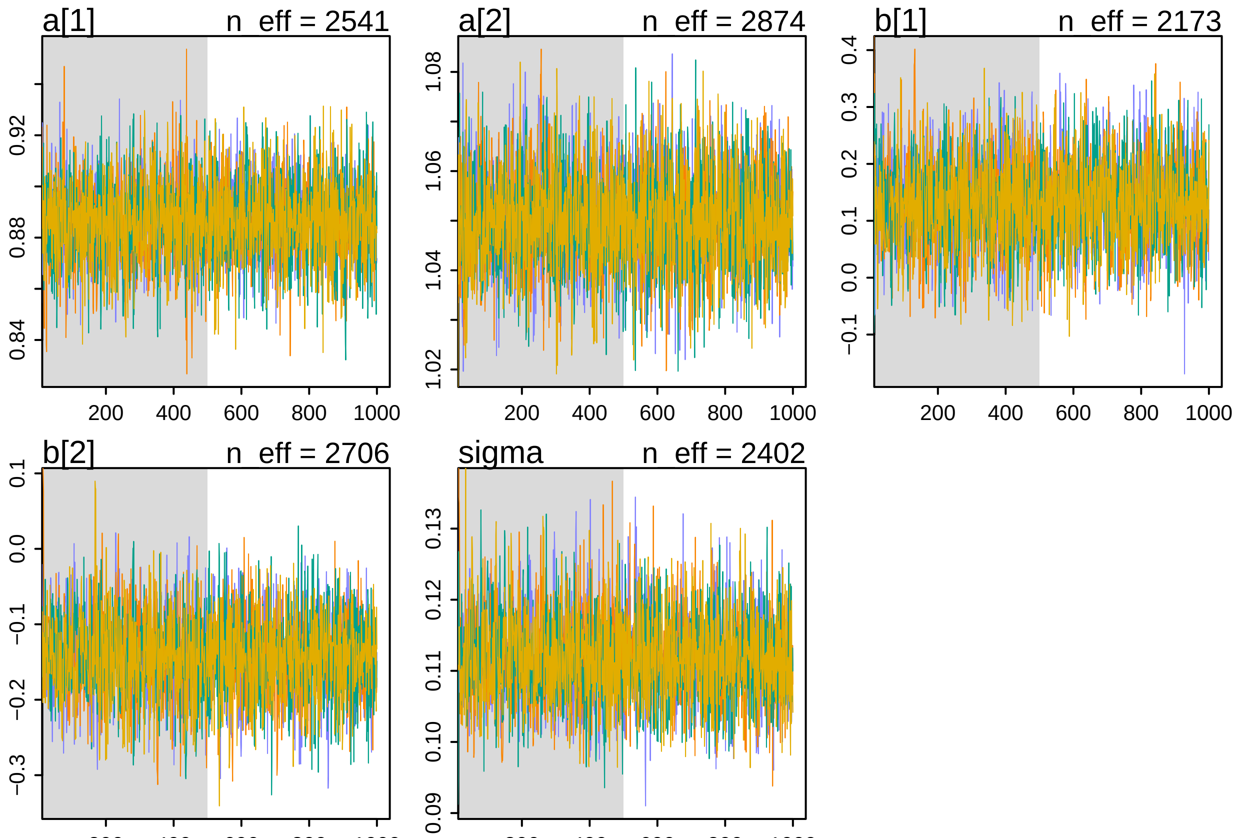
```
## Hamiltonian Monte Carlo approximation
## 2000 samples from 4 chains
##
## Sampling durations (seconds):
##         warmup sample total
## chain:1   0.06   0.04  0.10
## chain:2   0.06   0.04  0.10
## chain:3   0.07   0.04  0.11
## chain:4   0.06   0.04  0.10
##
## Formula:
## log_gdp_std ~ dnorm(mu, sigma)
## mu <- a[cid] + b[cid] * (rugged_std - 0.215)
## a[cid] ~ dnorm(1, 0.1)
## b[cid] ~ dnorm(0, 0.3)
## sigma ~ dexp(1)
```

```
precis( m9.1, depth=2 )
```

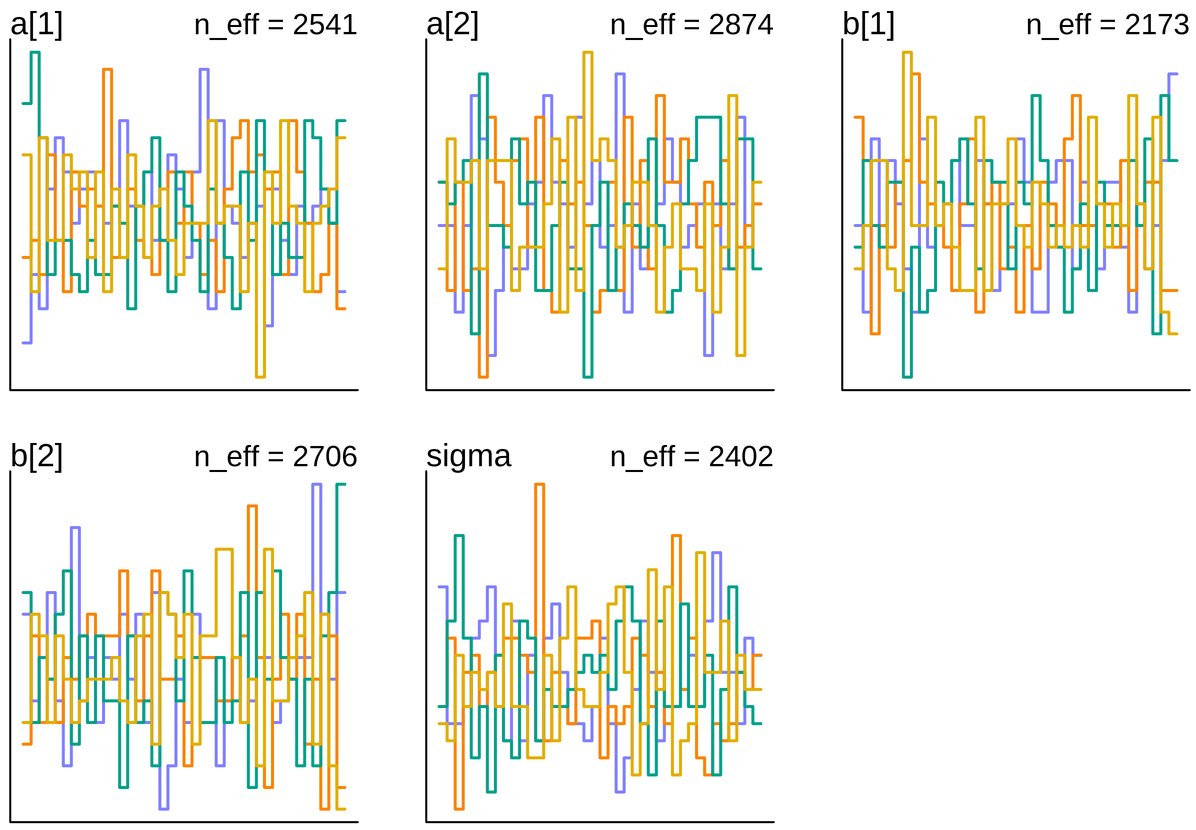
```
##              mean          sd        5.5%       94.5%    n_eff     Rhat4
## a[1]    0.8865946 0.015897560  0.86026301  0.91269809 2541.480 0.9985800
## a[2]    1.0502974 0.010107657  1.03431945  1.06675110 2874.485 0.9993111
## b[1]    0.1328096 0.075558031  0.01112909  0.25882495 2172.861 0.9992068
## b[2]   -0.1446571 0.055629771 -0.23601964 -0.05674836 2706.148 0.9995358
## sigma   0.1115347 0.006263894  0.10191389  0.12196116 2402.418 0.9992819
```

b) Check your chains with traceplots and tankplots. Interpret these graphs to explain why, or why not, your HMC model is suitable for inference.

```
traceplot( m9.1, window = c(50,1000))

trankplot( m9.1 )
```



```
# DISCUSSION: We include here the trace plot, which a window that omits the first
# 49 samples with `window = c(50,100).  We want to check these plots for stationarity,
# (are the paths of the chains at the same height; that is, is there a stable central
# tendency that each chain obeys), mixture (do the paths quickly move to explore the
# full region), and convergence (do multiple, independent chains stick around the
# same region with high probability).

# Next, we include a trace rank (trank) plot, which takes all samples for
# each parameter and ranks them.  When the chains explore the same space
# efficiently, the histograms are similar to one another and largely overlap.
# Which is what we see.  You should always check the health of your chains
# when running HMC.
```

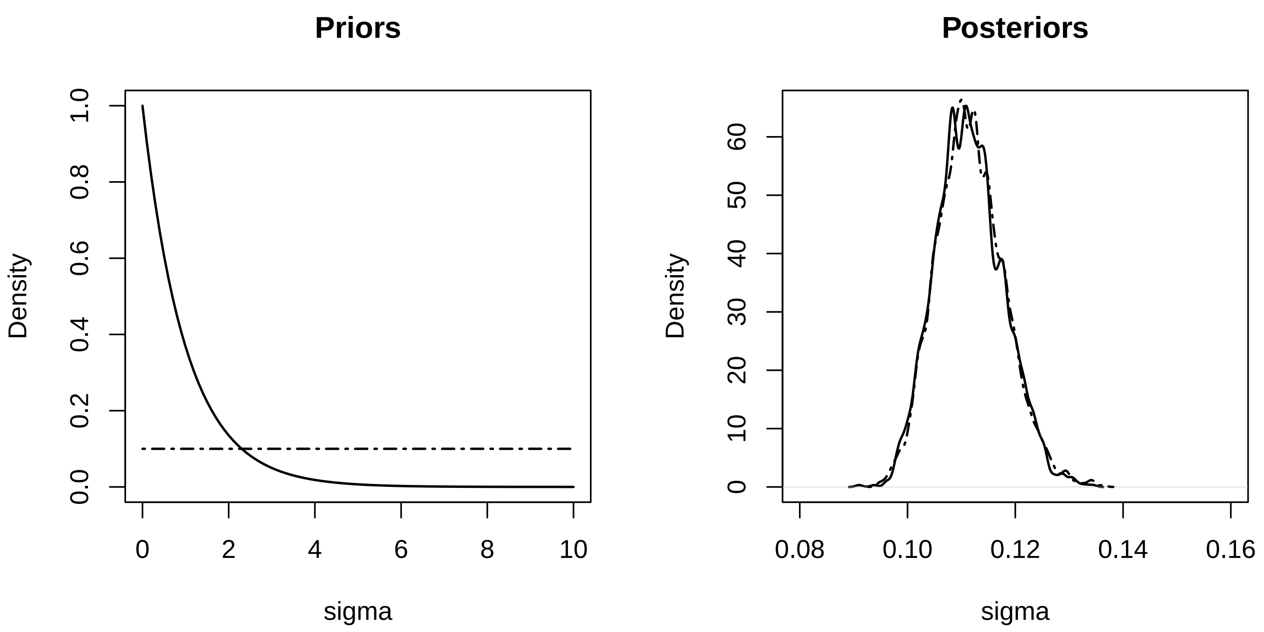


c) Now fit your HMC model with a flat prior for sigma, `sigma ~ dunif(0,1)`. What effect does this prior have on your posterior distribution? Explain your answer.

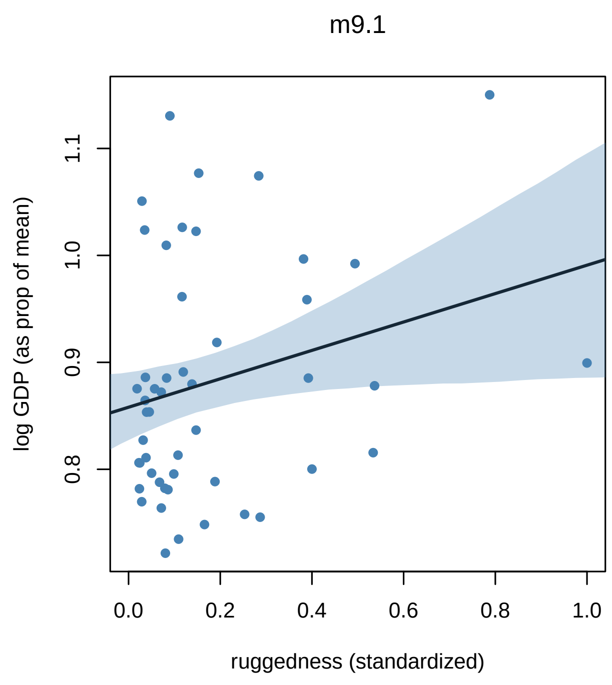
```
m9.1_unif <- ulam(
  alist(
    log_gdp_std ~ dnorm(mu, sigma),
    mu <- a[cid] + b[cid]*(rugged_std - 0.215) ,
    a[cid] ~ dnorm(1, 0.1),
    b[cid] ~ dnorm(0, 0.3),
    sigma ~ dunif(0,1)
  ), data=dat_slim, chains=4)
precis( m9.1_unif )

sigma <- extract.samples(m9.1,pars="sigma")
sigma_unif <- extract.samples(m9.1_unif,pars="sigma")
```

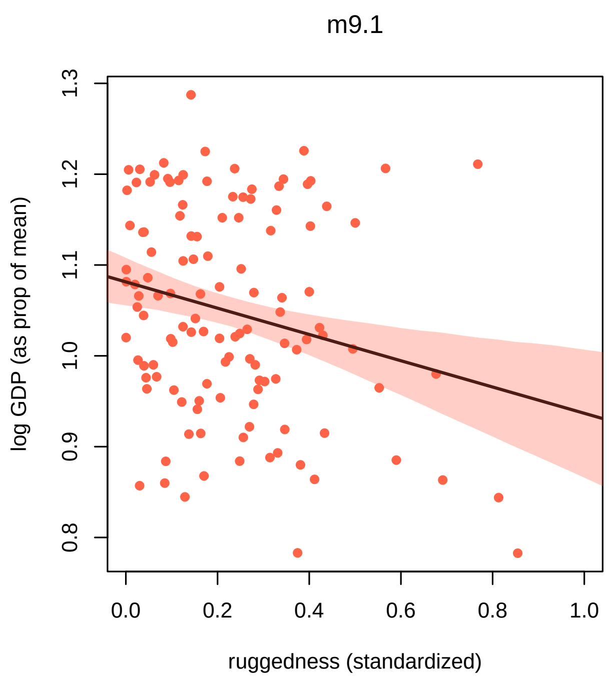
```
# Below is a comparison of prior distributions to posterior distributions for
# sigma. The left plot displays the original model m9.1 with an informed prior
# (solid) and m9.1_unif with an uninformed uniform prior (dashed) on sigma.

par(mfrow=c(1,2))
# priors
curve( dexp(x,1) , from=0 , to=10 , main="Priors",
    xlab="sigma" , ylab="Density" , ylim=c(0,1), lwd=1.5 )
curve( dunif(x, 0, 10) , add=TRUE,   lwd=1.5, lty=6)
```

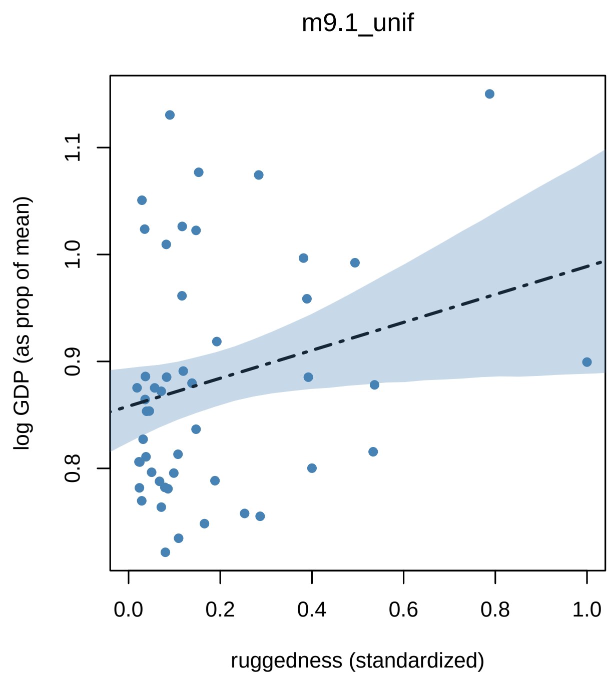
```
# posterior
dens( sigma[[1]] , xlab="sigma", xlim=c(0.08, 0.16), main="Posteriors", lwd=1.5)
dens( sigma_unif[[1]] , add=TRUE, lwd=1.5, lty=6)
```

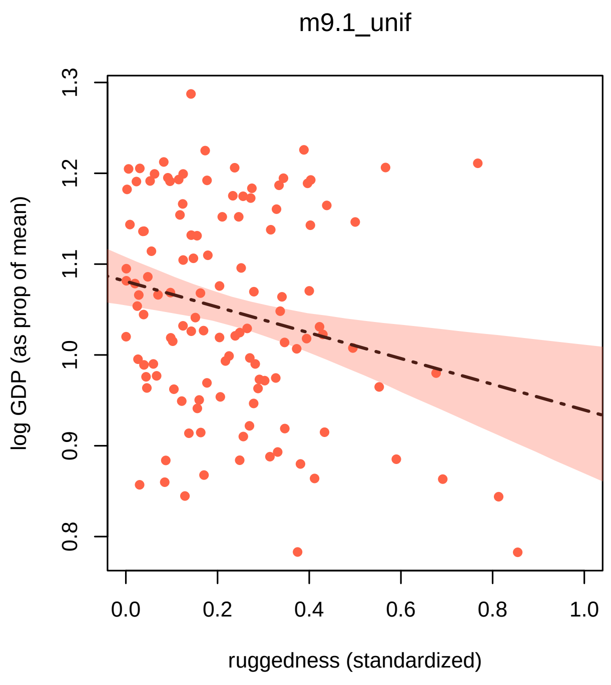


```
# DISCUSSION: Despite the difference between the informed, regularizing exponential
# prior (solid) and the very weak uniform prior (dashed), the posterior distributions
# for sigma are practically identical.

# Indeed, we see no effective difference between the posterior distributions:

# Plot guide:
# TOP Row: Baseline model m9.1, African (blue) and Non-African (red)
# Bottom Row: Alternative m9.1_unif, African (blue) and Non-African (red)
```









d) Now fit your model with the log normal prior `b[cid] ~ dlnorm(0,1)` for b. What effect does this prior have on your posterior distribution? Explain your answer.

```
m9.1_lnorm <- ulam(
  alist(
    log_gdp_std ~ dnorm(mu, sigma),
    mu <- a[cid] + b[cid]*(rugged_std - 0.215) ,
    a[cid] ~ dnorm(1, 0.1),
    b[cid] ~ dlnorm(0, 1),
    sigma ~ dexp(1)
  ), data=dat_slim, chains=4)

b <- extract.samples(m9.1,pars="b")
b <- b[[1]]
b_lnorm <- extract.samples(m9.1_lnorm,pars="b")
b_lnorm <- b_lnorm[[1]]
```

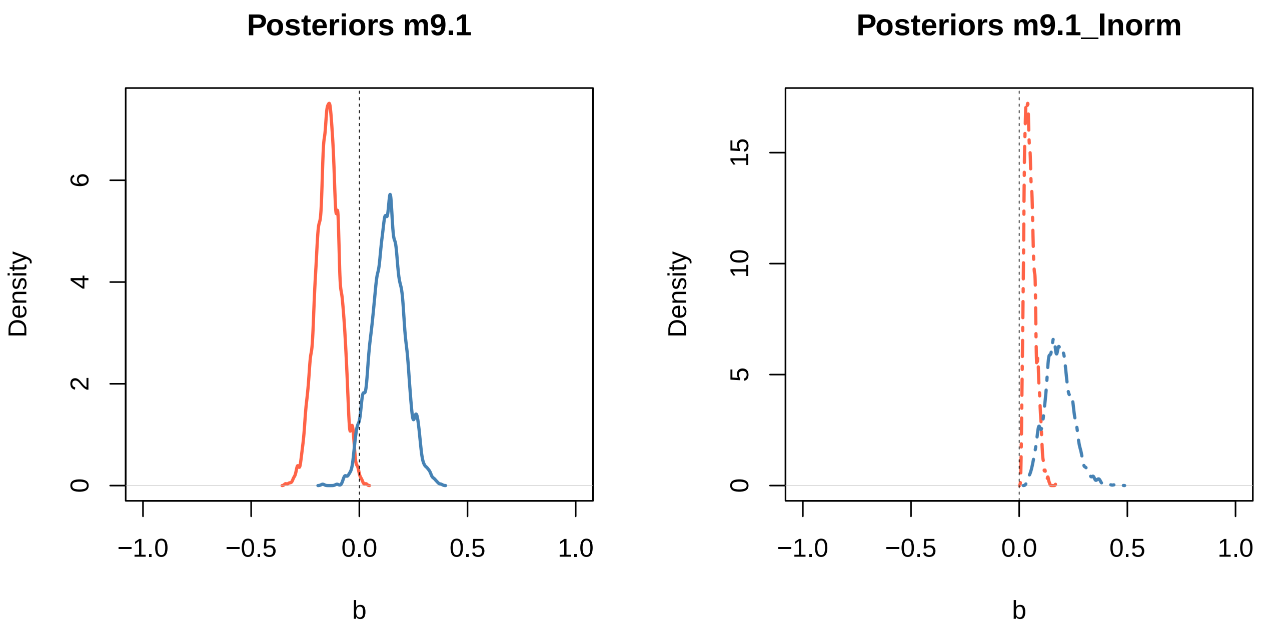
```
# DISCUSSION: The change of the variable slope parameter prior `b` from a normal
# distribution, which allows positive and negative slopes, to a log-normal, which
# rules out negative slopes, has the effect of misfiting a positive slope to
# the non-African countries when it should be negative
precis( m9.1, depth= 2)
```

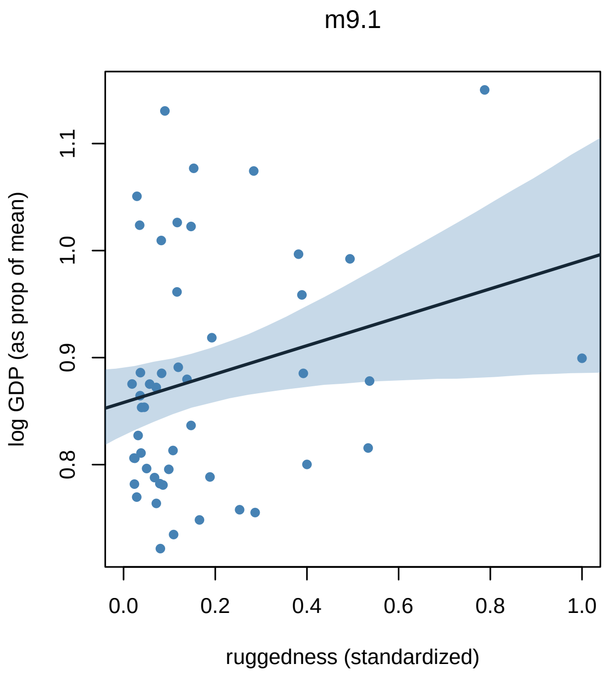
```
##              mean          sd        5.5%       94.5%    n_eff     Rhat4
## a[1]    0.8865946 0.015897560  0.86026301  0.91269809 2541.480 0.9985800
## a[2]    1.0502974 0.010107657  1.03431945  1.06675110 2874.485 0.9993111
## b[1]    0.1328096 0.075558031  0.01112909  0.25882495 2172.861 0.9992068
## b[2]   -0.1446571 0.055629771 -0.23601964 -0.05674836 2706.148 0.9995358
## sigma   0.1115347 0.006263894  0.10191389  0.12196116 2402.418 0.9992819
```

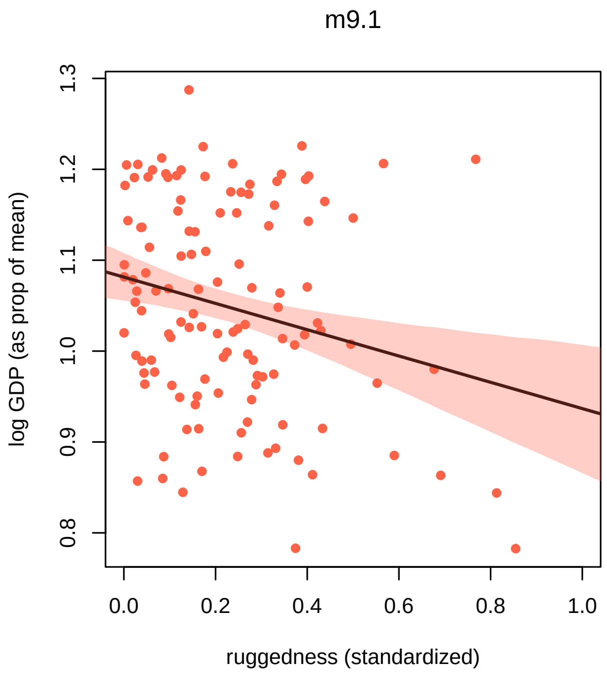
```
precis( m9.1_unif, depth=2)
```

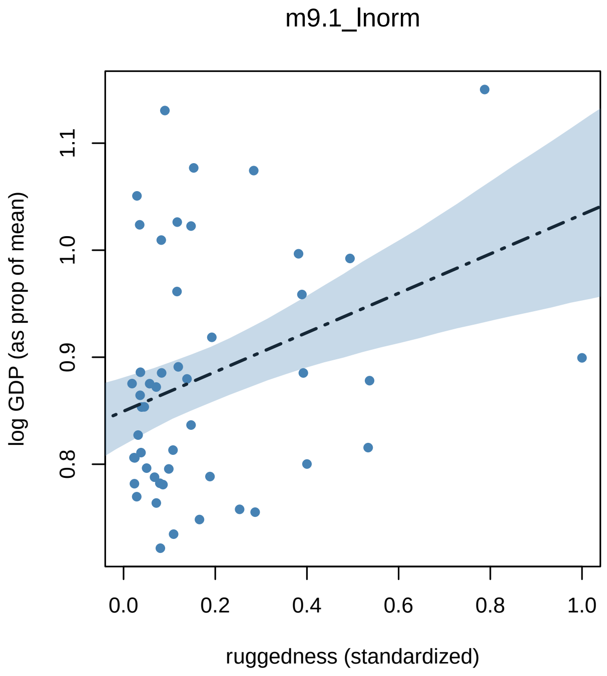
```
##              mean          sd         5.5%      94.5%    n_eff     Rhat4
## a[1]    0.8862651 0.015795512  0.861259430  0.9115735 2995.977 0.9989617
## a[2]    1.0505513 0.010072209  1.034530000  1.0665966 3299.377 0.9993064
## b[1]    0.1306254 0.077108492  0.009004215  0.2507081 2510.780 0.9994029
## b[2]   -0.1416425 0.057174813 -0.228306445 -0.0526917 2577.316 1.0001769
## sigma   0.1117303 0.006347419  0.102080560  0.1222537 2391.314 0.9993651
```

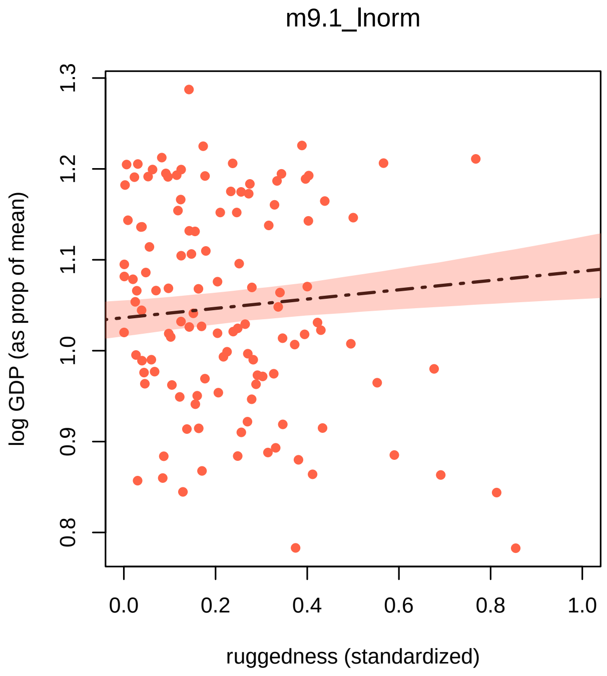
```
# This difference is visualized in terms of the posterior distributions for `b`,
# followed by a comparison of the baseline regression models of m9.1 to those
# of model m9.1_lnorm.

par(mfrow=c(1,2))
dens( b[,2] , xlab="b", xlim=c(-1, 1), main="Posteriors m9.1", col="tomato", lwd=2)
dens( b[,1] , add=TRUE, col="steelblue", lwd=2)
abline(v=0, lty=2, lwd=0.5)

dens( b_lnorm[,2] , xlab="b", xlim=c(-1, 1), main="Posteriors m9.1_lnorm",
      col="tomato", lwd=2, lty=6)
dens( b_lnorm[,1] , add=TRUE, col="steelblue", lwd=2, lty=6)
abline(v=0, lty=2, lwd=0.5)
```


Posteriors m9.1
Posteriors m9.1_lnorm
Density
b
−1.0
−0.5
0.0
0.5
1.0
0
2
4
6
5
10
15









3. Binomial Regression

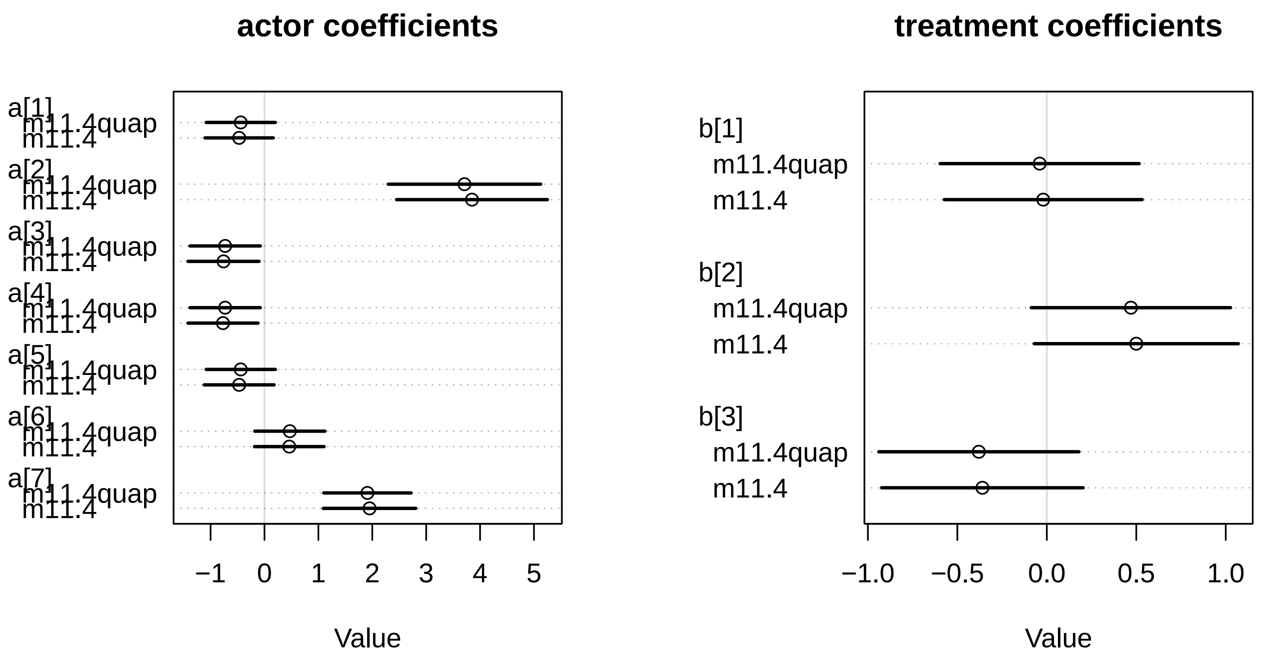
We started the course sampling marbles from a bucket to estimate its contents and tossing a globe to estimate the proportion of its surface covered in water. Each made use of the binomial distribution and was ideal to introduce the fundamentals of Bayesian inference. Nevertheless, *Binomial regression* – which is *any* type of GLM using a binomial mean-variance relationship – introduces complications that we needed to postpone until now.

Return to the prosocial chimpanzee experiment in section §11.1 of the textbook, and the HMC model that features individual chimpanzee (`actor`) parameters `actor` and individual `treatment` parameters:

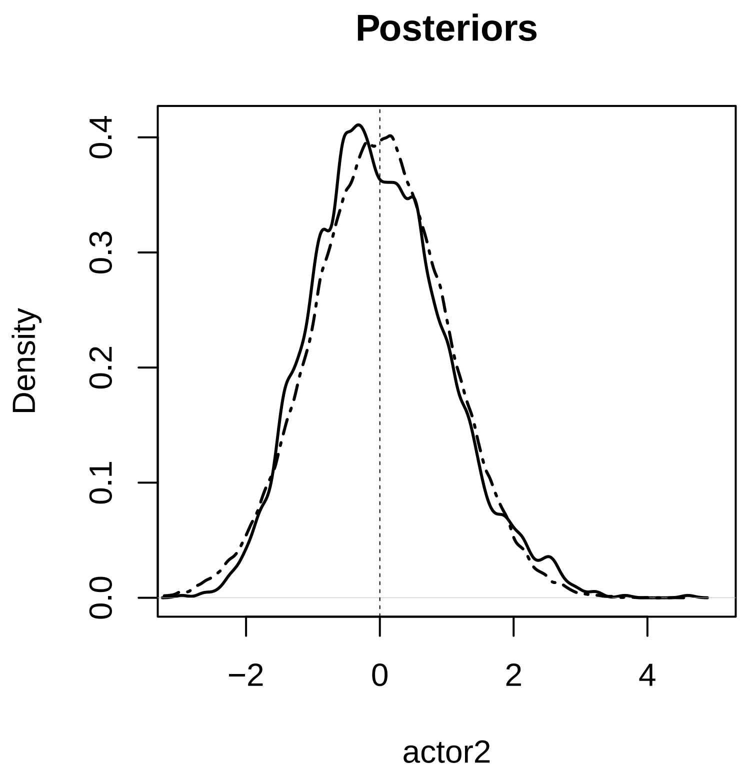
a) Compare `m11.4` to a Laplacian quadratic approximate posterior distribution, constructed using `quap()`, that also includes individual parameters for `actor` and `treatment`. What are the differences and similarities between the two approximate posteriors? Explain your answer.

```r
data(chimpanzees)
dat <- chimpanzees
dat$treatment <- 1 + dat$prosoc_left + 2*dat$condition
dat_list <- list(
  pulled_left = dat$pulled_left,
  actor = dat$actor,
  treatment = as.integer(dat$treatment) )

# Original HMC Model
m11.4 <- ulam(
  alist(
    pulled_left ~ dbinom( 1, p ),
    logit(p) <- a[actor] + b[treatment] ,
    a[actor] ~ dnorm( 0 , 1.5 ),
    b[treatment] ~ dnorm( 0, 0.5 )
  ), data = dat_list, chains=4, log_lik =TRUE)

# Quadratic Approximation Model
m11.4quap <- quap(
  alist(
    pulled_left ~ dbinom( 1, p ),
    logit(p) <- a[actor] + b[treatment] ,
    a[actor] ~ dnorm( 0 , 1.5 ),
    b[treatment] ~ dnorm( 0, 0.5 )
  ), data = dat)
```

```r
#plot( precis( m11.4, depth=2, pars="a"))

# A quick inspection of the parameters of each reveals that the estimates are
# identical, except for actor 2 `a[2]`, where quadratic approximation gives a
# slightly lower estimate of uncertainty on handedness than the HMC model.  On
# closer inspection, we see a tendency of all quap estimates of the actors'
# handedness to revert very slightly to the mean.  To be clear, these differences
# are dominated by SE for making any strong claim about the parameter values. But,
# this overall pattern does suggest a systematic difference in the numerical
# approximation methods (Laplace vs HMC), a difference we can investigate.

# That difference is most pronounced for actor 2. So let's focus there.

par(mfrow=c(1,2))
plot( coeftab( m11.4, m11.4quap), pars=c("a[1]", "a[2]", "a[3]", "a[4]", "a[5]", "a[6]", "a[7]" ), main
plot( coeftab( m11.4, m11.4quap), pars=c("b[1]", "b[2]", "b[3]" ), main="treatment coefficients")
```



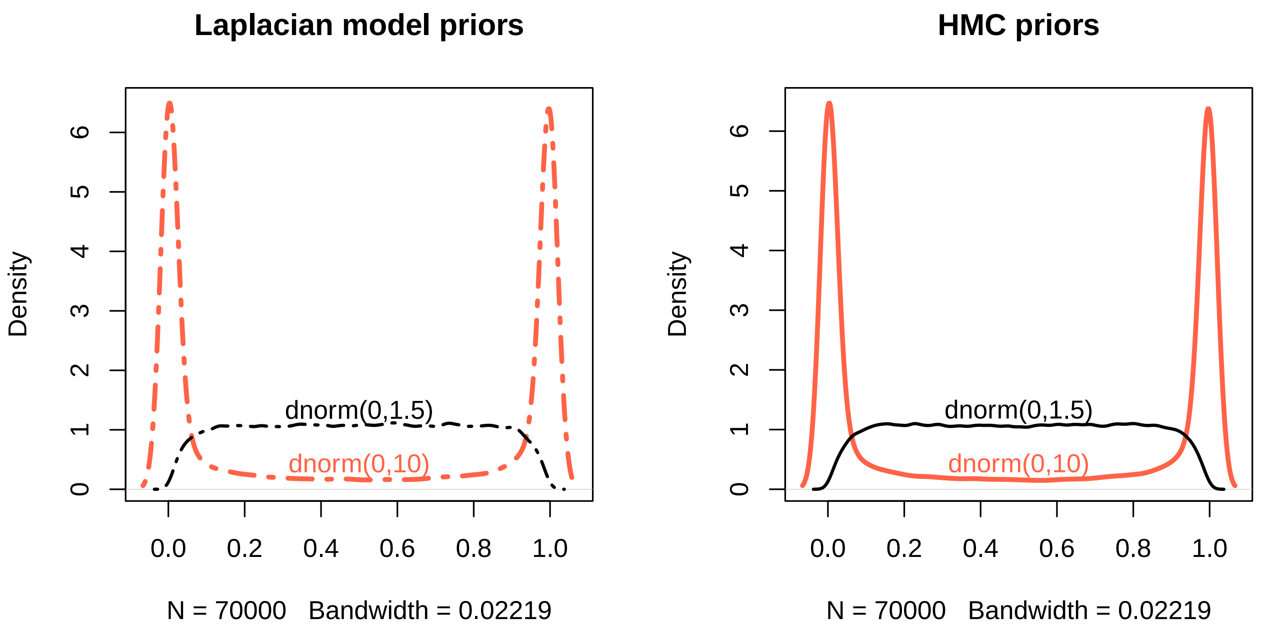
```r
# We extract samples from each posterior for the second actor, `a[2]`, and plot
# samples from each posterior distribution, m11.4 (solid) and m11.4quap (dashed)
actor2 <- extract.samples(m11.4, pars="a[2]")
actor2 <- standardize(actor2[[1]])
actor2quap <- extract.samples(m11.4quap, pars="a[2]")
actor2quap <- standardize(actor2quap[[1]])

dens( actor2 , xlab="actor2", xlim=c(-3, 5), main="Posteriors", lwd=1.5)
dens( actor2quap , add=TRUE, lwd=1.5, lty=6)
abline(v=0, lty=2, lwd=0.5)
# We can see the slight difference in each models MAP estimates.
```



b) Change the prior on the variable intercept to `dnorm( 0 , 10)` and estimate the posterior distribution with both `ulam()` and `quap()`. Do the differences between the two estimatations increase, decrease, or stay the same? Explain your answer.

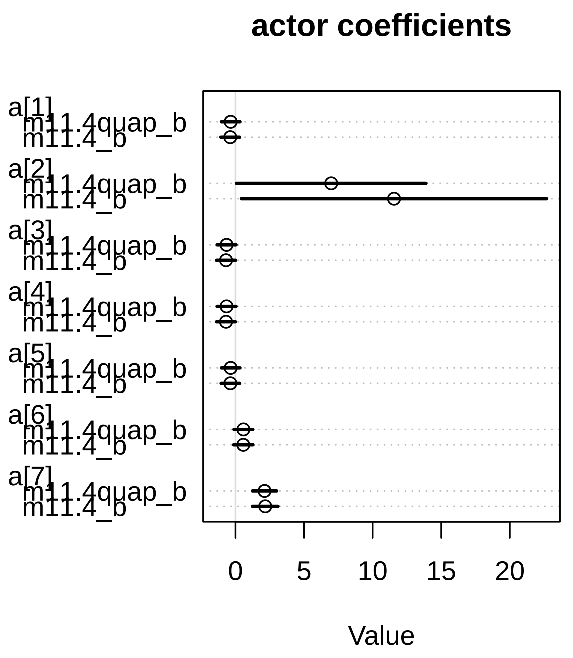
```
# Modified HMC Model
m11.4_b <- ulam(
  alist(
    pulled_left ~ dbinom( 1, p ),
    logit(p) <- a[actor] + b[treatment] ,
    a[actor] ~ dnorm( 0 , 10 ),
    b[treatment] ~ dnorm( 0, 0.5 )
  ), data = dat_list, chains=4, log_lik =TRUE)

# Modified Quadratic Approximation Model
m11.4quap_b <- quap(
  alist(
    pulled_left ~ dbinom( 1, p ),
    logit(p) <- a[actor] + b[treatment] ,
    a[actor] ~ dnorm( 0 , 10 ),
    b[treatment] ~ dnorm( 0, 0.5 )
  ), data = dat)

# extract priors Laplacian models
prior_crazy <- extract.prior(m11.4quap_b, n=10000)
prior_sane <- extract.prior(m11.4quap, n=10000)
# extract priors HMC models
prior_crazy2 <- extract.prior(m11.4_b, n=10000)
prior_sane2 <- extract.prior(m11.4, n=10000)
```

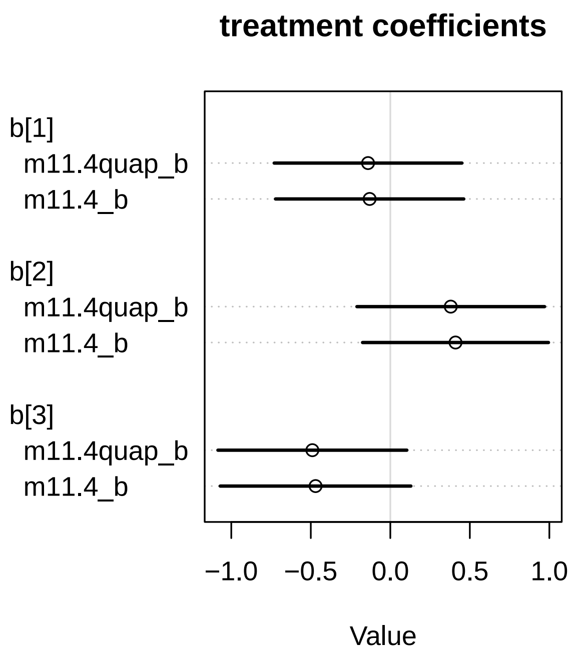
```r
# The first thing to recognize is that `dnorm(0, 10)` is a *stronger prior* than
# `dnorm(0,1.5)`. The reason is that this "flat prior" is on the parameter scale,
# but once you convert it to the outcome scale you see that the prior expresses
# that every chimpanzee is expected to always pull the left lever or never pull
# the left lever:

par(mfrow=c(1,2))
# Laplacian models
dens(inv_logit(prior_crazy$a), lty=6, lwd=3, col="tomato", main="Laplacian model priors")
dens(inv_logit(prior_sane$a), add=TRUE, lty=6, lwd=2)
text(0.5, 0.4, "dnorm(0,10)",  col="tomato")
text(0.5, 1.3, "dnorm(0,1.5)")
# HMC models
dens(inv_logit(prior_crazy2$a),  lwd=3, col="tomato", main="HMC priors")
dens(inv_logit(prior_sane2$a), add=TRUE, lwd=2)
text(0.5, 0.4, "dnorm(0,10)", col="tomato")
text(0.5, 1.3, "dnorm(0,1.5)")
```



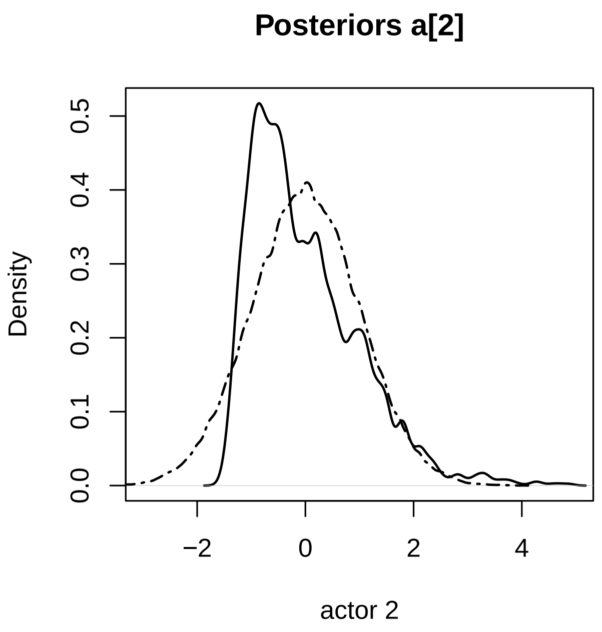
```r
# We plot the pair of priors for the Laplacian (quap) model and HMC (ulam) model
# simply to show, or remind you, that the priors are the same: neither are mediated
# by the model.  The crazy prior `dnorm(0,10)` is plotted in red, the sensible
# prior `dnorm(0,1.5)` in black.
```

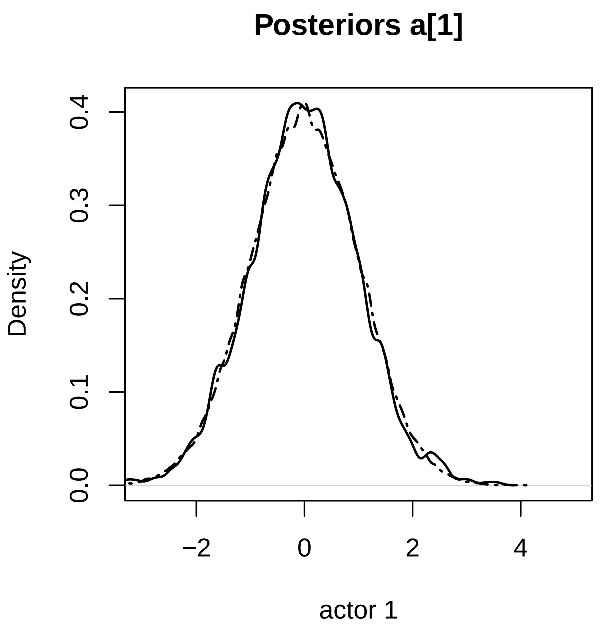
```r
# What effects if any does this strong prior have? A quick inspection of the
# parameters of each confirms that the estimates remain identical, except for
# actor 2 `a[2]`, where quadratic approximation gives a an even lower estimate of
# uncertainty on handedness:

par(mfrow=c(1,2))
plot( coeftab( m11.4_b, m11.4quap_b), pars=c("a[1]", "a[2]", "a[3]", "a[4]", "a[5]", "a[6]", "a[7]" ),
plot( coeftab( m11.4_b, m11.4quap_b), pars=c("b[1]", "b[2]", "b[3]" ), main="treatment coefficients")
```





```
# Let us extract samples from each posterior for the second actor, `a[2]`, and plot
# samples from each posterior distribution, m11.4 (solid) and m11.4quap (dashed)
actor2_b <- extract.samples(m11.4_b, pars="a[2]")
actor2_b <- standardize(actor2_b[[1]])
actor2quap_b <- extract.samples(m11.4quap_b, pars="a[2]")
actor2quap_b <- standardize(actor2quap_b[[1]])

# For comparison, consider actor 1
actor1_b <- extract.samples(m11.4_b, pars="a[1]")
actor1_b <- standardize(actor1_b[[1]])
actor1quap_b <- extract.samples(m11.4quap_b, pars="a[1]")
actor1quap_b <- standardize(actor1quap_b[[1]])

# PLOT actor 2
dens( actor2_b , xlab="actor 2", xlim=c(-3, 5), main="Posteriors a[2]", lwd=1.5)
dens( actor2quap_b , add=TRUE, lwd=1.5, lty=6)
#abline(v=0, lty=2, lwd=0.5)

# PLOT actor 1
dens( actor1_b , xlab="actor 1", xlim=c(-3, 5), main="Posteriors a[1]", lwd=1.5)
dens( actor1quap_b , add=TRUE, lwd=1.5, lty=6)
```





```
# DISCUSSION: There are a couple things going on here.  First, the data dominates
# the strong prior for all actors except the one, a[2], which only pulls the left
# lever.  As there is a lot more mass in the prior `dnorm(0,10)` on this possibility,
# than the relatively flat (in outcome space) prior `dnorm(0,1.5)`, the ulam variable-
# effects model responds to this in the posterior.

# The strong prior does not have the same effect on the Laplacian varrying effects
# model, however. Why? Quadratic approximation presumes that the posterior distribution
# is (multivariate) Gaussian, which enforces symmetric uncertainty around the
# posterior mode on each dimension. But, uncertainty within logistic regression
# is sometimes asymmetric, which is what we see in the case of handedness of
# actor 2. The HMC model can respond to the strong prior, `dnorm(0,10)`, pushing
# the consequences of that strong assumption -- namely that chimps always pull
# the right or always pull the left, and your uncertainty is simply which hand
# it will be -- through to your posterior distribution.  Actor 2 accords with your
# strong prior:  the evidence suggests he's a Lefty.

# What is the effect on the posterior? As values of x approach infinity, the logistic
# function returns an estimation of the probability that actor 2 will pull the left
# lever that approaches 1.  The long tail of the m11.4 posterior for `a[2]` represents
# the one-sided infinity of large, positive values that are consistent with the data
# (i.e., of observing only left-handed pulls from N pulls). In the small world of
# m11.4, the strong `dnorm(0,10) prior simply inflates this tale to emphasize your
# uncertainty of how close to 1 your estimate should be given N observations of actor
# 2.  In the large world, the assumption that `dnorm(0,10)` encodes
# is unreasonable if not crazy.

# Lastly, notice that this crazy prior only impacts estimates concerning actor 2's
# handedness. To see this, compare the asymmetric uncertainty regarding the prospect
# of actor 2 ever pulling the right handle to the posterior for actor 1, where the
# uncertainty in the posterior distribution of is approximately Gaussian. Hence, the
# quadratic approximate posterior of m11.4quap accords with the full Bayesian
# distribution of m11.4 everywhere else, despite of the crazy `dnorm(0,10)` prior.
```