Code

Philipp Grimm (PWG15)

5/8/2018

########################  
# File: analyze\_data.R  
# Purpose: to conduct core sampling and analysis  
# Output: console, graphs  
########################  
  
###############  
# A. Setup   
###############  
  
rm(list=ls())  
  
# Load packages and helper functions  
source("helper\_functions.R")

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

##   
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':  
##   
## date

## Loading required package: coda

## Loading required package: MASS

##   
## Attaching package: 'MASS'

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

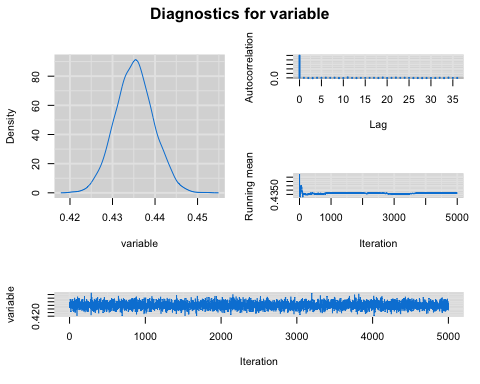
## ##  
## ## Markov Chain Monte Carlo Package (MCMCpack)

## ## Copyright (C) 2003-2018 Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park

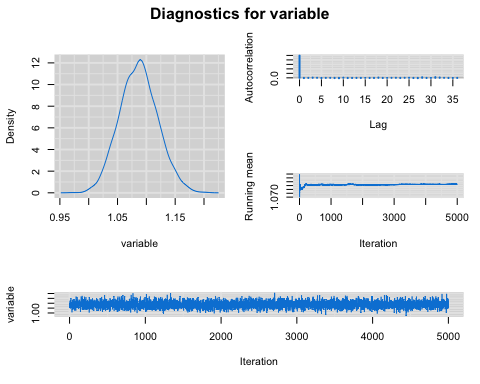
## ##  
## ## Support provided by the U.S. National Science Foundation

## ## (Grants SES-0350646 and SES-0350613)  
## ##

# Load data  
tweet\_storms <- readRDS("tweet\_storms.rds")  
  
all\_storms <- tweet\_storms %>% pull(days\_elapsed)  
pre\_election <- tweet\_storms %>% filter(!post\_election) %>% pull(days\_elapsed)  
post\_election <- tweet\_storms %>% filter(post\_election) %>% pull(days\_elapsed)  
  
###############  
# B. Examine different posteriors   
# (always source sampler file before runs, since posteriors overwrite each other)   
###############  
  
# LOG-NORMAL #############################  
source("lognormal\_sampler.R")  
  
# NON-INFORMATIVE  
# 1. No tuning necessary  
# 2. Sample   
lognorm\_samples\_noninf <- lognormSamp(data=all\_storms, B=10000,   
 a=1000, b=1000,   
 c=1, d=1000)  
  
# 3. Check for convergence  
mcmcplot2(lognorm\_samples\_noninf$mu)



mcmcplot2(lognorm\_samples\_noninf$sig2)



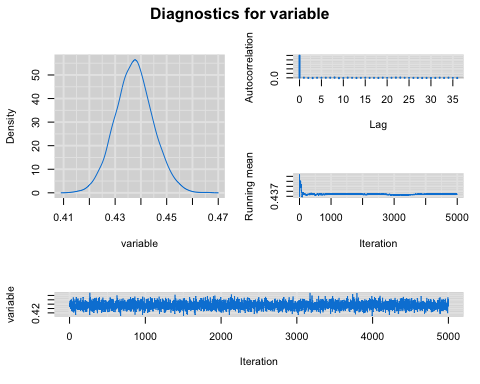
# 4. Grab Log-normal parameters  
lapply(lognorm\_samples\_noninf, mean)

## $mu  
## [1] 0.4351605  
##   
## $sig2  
## [1] 1.086727

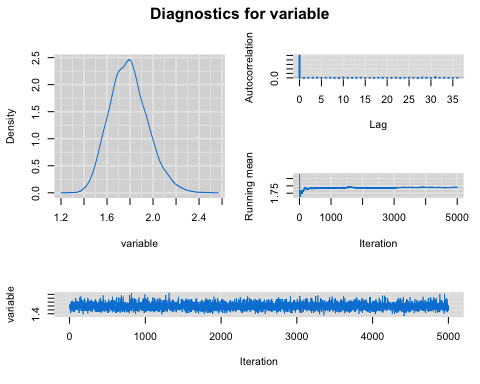
lapply(lognorm\_samples\_noninf, median)

## $mu  
## [1] 0.43514  
##   
## $sig2  
## [1] 1.086738

# INFORMATIVE  
# 1. No tuning necessary  
# 2. Sample   
lognorm\_samples\_inf <- lognormSamp(data=all\_storms, B=10000,   
 a=2, b=1,   
 c=0.752763, d=1)  
  
# 3. Check for convergence  
mcmcplot2(lognorm\_samples\_inf$mu)



mcmcplot2(lognorm\_samples\_inf$sig2)



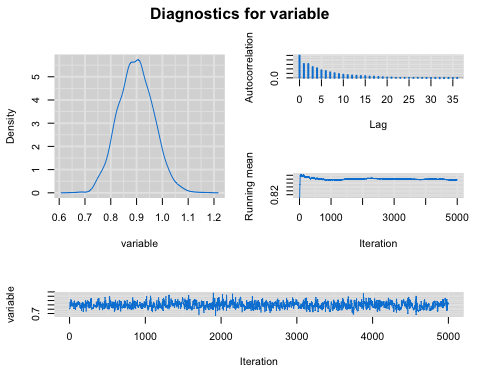
# 4. Grab Log-normal parameters  
lapply(lognorm\_samples\_inf, mean)

## $mu  
## [1] 0.437406  
##   
## $sig2  
## [1] 1.789291

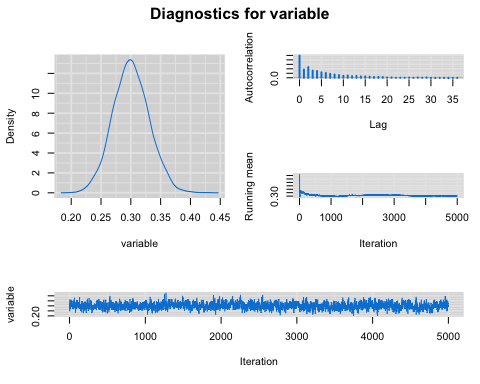
lapply(lognorm\_samples\_inf, median)

## $mu  
## [1] 0.4373398  
##   
## $sig2  
## [1] 1.782726

# GAMMA #############################  
source("gamma\_sampler.R")  
  
# NON-INFORMATIVE  
# 1. Tune with Gamma proposal  
# tune\_acceptance\_rate(  
# a\_vals = seq(60, 70, by=1),  
# b\_vals = seq(60, 70, by=1),  
# sampler = gammaSamp,  
# data = all\_storms,  
# B = 10000,  
# p=1, q=0, r=0, s=0,  
# alpha\_start = 1, beta\_start = 1) # settle on Gamma(60, 63) with ~43% acceptance  
  
# 2. Sample   
gamma\_samples\_noninf <- gammaSamp(data = all\_storms, B = 10000,  
 a1 = 60, b1 = 63,  
 p=1, q=0, r=0, s=0,  
 alpha\_start = 1, beta\_start = 1)  
  
# 3. Check for convergence  
mcmcplot2(gamma\_samples\_noninf$alpha)



mcmcplot2(gamma\_samples\_noninf$beta)



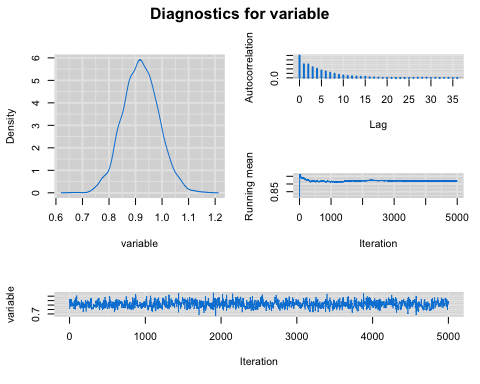
# 4. Grab Gamma parameters  
lapply(gamma\_samples\_noninf, mean)

## $ar  
## [1] 0.4314  
##   
## $alpha  
## [1] 0.8972315  
##   
## $beta  
## [1] 0.3004765

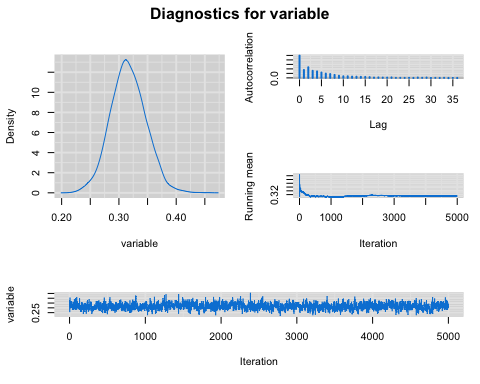
lapply(gamma\_samples\_noninf, median)

## $ar  
## [1] 0  
##   
## $alpha  
## [1] 0.8967528  
##   
## $beta  
## [1] 0.3000792

# INFORMATIVE  
# 1. Tune with Gamma proposal  
# Not necessary  
  
# 2. Sample   
gamma\_samples\_inf <- gammaSamp(data = all\_storms, B = 10000,  
 a1 = 64, b1 = 67,  
 p=10, q=10, r=10, s=10,  
 alpha\_start = 1, beta\_start = 1)  
  
# 3. Check for convergence  
mcmcplot2(gamma\_samples\_inf$alpha)



mcmcplot2(gamma\_samples\_inf$beta)



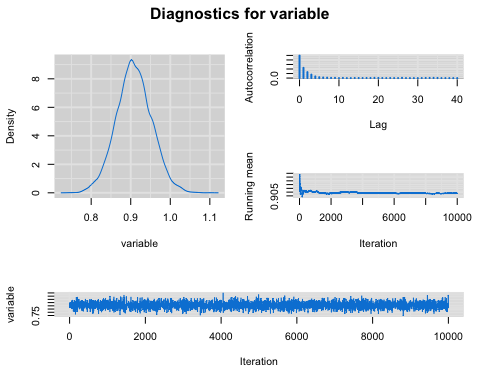
# 4. Grab Gamma parameters  
lapply(gamma\_samples\_inf, mean)

## $ar  
## [1] 0.4524  
##   
## $alpha  
## [1] 0.9197122  
##   
## $beta  
## [1] 0.3163077

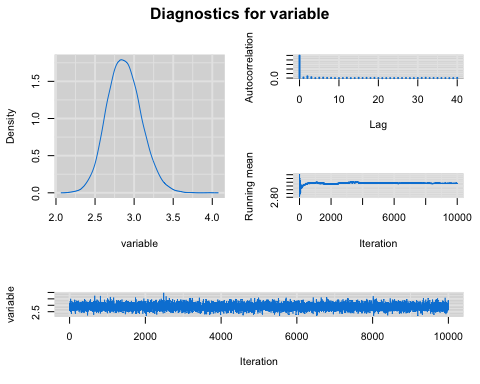
lapply(gamma\_samples\_inf, median)

## $ar  
## [1] 0  
##   
## $alpha  
## [1] 0.9183915  
##   
## $beta  
## [1] 0.315243

# WEIBULL #############################  
source("weibull\_sampler.R")  
  
# 1. Tune with Gamma proposal  
# tune\_acceptance\_rate(  
# a\_vals = seq(60, 70, by=1),  
# b\_vals = seq(60, 70, by=1),  
# sampler = weibullSamp,  
# data = all\_storms,  
# B = 20000,  
# theta\_start = 1, lambda\_start = 1) # settle on Gamma(60, 67) with ~44% acceptance  
  
# 2. Sample   
weibull\_samples <- weibullSamp(  
 seed = 1, data = all\_storms, B = 20000,  
 a1 = 60, b1 = 67,  
 theta\_start = 0.5, lambda\_start = 1)  
  
# 3. Check for convergence  
mcmcplot2(weibull\_samples$theta)



mcmcplot2(weibull\_samples$lambda)



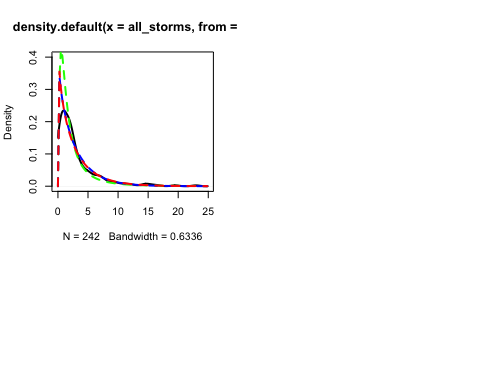
# 4. Grab Weibull parameters  
lapply(weibull\_samples, mean)

## $ar  
## [1] 0.4407  
##   
## $theta  
## [1] 0.9083159  
##   
## $lambda  
## [1] 2.873532

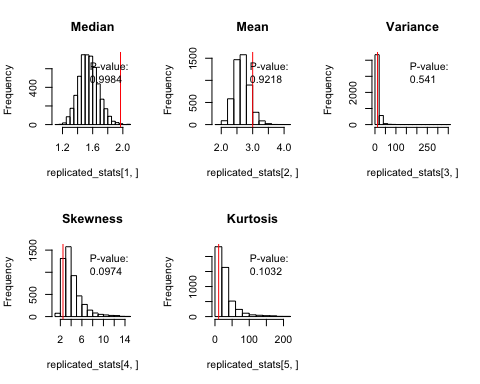
lapply(weibull\_samples, median)

## $ar  
## [1] 0  
##   
## $theta  
## [1] 0.9067073  
##   
## $lambda  
## [1] 2.865898

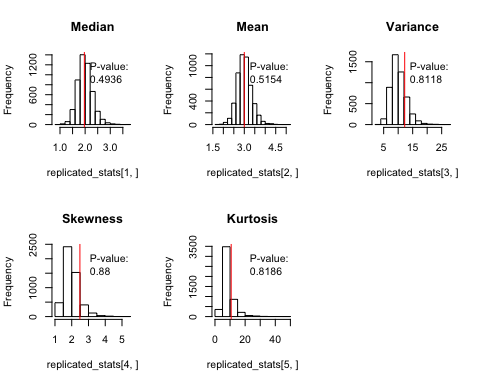
# OPTIONAL EXHAUSTIVE CONVERGENCE CHECK CODE  
# # X. Run multiple chains and check for convergence  
# chain1 <- weibullSamp(seed = 1, data = all\_storms, B = 20000,  
# a1 = 60, b1 = 67,  
# theta\_start = 0.5, lambda\_start = 1)  
# chain2 <- weibullSamp(seed = 2, data = all\_storms, B = 20000,  
# a1 = 60, b1 = 67,  
# theta\_start = 0.75, lambda\_start = 2)  
# chain3 <- weibullSamp(seed = 3, data = all\_storms, B = 20000,  
# a1 = 60, b1 = 67,  
# theta\_start = 1, lambda\_start = 3)  
# chain4 <- weibullSamp(seed = 4, data = all\_storms, B = 20000,  
# a1 = 60, b1 = 67,  
# theta\_start = 1.5, lambda\_start = 4)  
#   
# gelman\_rubin(chain1$theta, chain2$theta, chain3$theta, chain4$theta)  
# gelman\_rubin(chain1$lambda, chain2$lambda, chain3$lambda, chain4$lambda)  
#   
# # If desired: convergence plots for every chain variable  
# # mcmcplot2(chain1$theta) # etc.  
#   
# # Y. Determine optimal thinning (10 seems best)  
# show\_thinning\_options(chain1$theta)  
# show\_thinning\_options(chain2$theta)  
# show\_thinning\_options(chain3$theta)  
# show\_thinning\_options(chain4$theta)  
#   
# show\_thinning\_options(chain1$lambda)  
# show\_thinning\_options(chain2$lambda)  
# show\_thinning\_options(chain3$lambda)  
# show\_thinning\_options(chain4$lambda)  
  
  
###############  
# C. Decide on best likelihood (probably tie between weibull, gamma)  
###############  
  
# Graphical comparison to empirical density  
plot(density(all\_storms, from=0), lwd=2, ylim=c(0, 0.4))  
curve(dlnorm(x, 0.4351, sqrt(1.086)), col="green", lwd=2, lty=2, add=TRUE)  
curve(dgamma(x, 0.9197, 0.3163), col="blue", lwd=2, lty=2, add=TRUE)  
curve(dweibull(x, 0.9083, 2.8735), col="red", lwd=2, lty=2, add=TRUE)  
  
# Show replicated moment distribution with original data  
show\_replicate\_analysis(all\_storms, rlnorm, lognorm\_samples\_noninf$mu, sqrt(lognorm\_samples\_noninf$sig2)) # remember sqrt!



show\_replicate\_analysis(all\_storms, rgamma, gamma\_samples\_noninf$alpha, gamma\_samples\_noninf$beta)



show\_replicate\_analysis(all\_storms, rweibull, weibull\_samples$theta, weibull\_samples$lambda)



###############  
# D. Compare credible intervals of pre- and post-election   
# (there is a difference, but not stat significant because intervals overlap)  
###############  
  
# Use Weibull because paramters have interesting interpretation  
source("weibull\_sampler.R")  
  
weibull\_samples\_pre <- weibullSamp(  
 seed = 1, data = pre\_election, B = 20000,  
 a1 = 60, b1 = 67,  
 theta\_start = 0.5, lambda\_start = 1)  
  
weibull\_samples\_post <- weibullSamp(  
 seed = 1, data = post\_election, B = 20000,  
 a1 = 60, b1 = 67,  
 theta\_start = 0.5, lambda\_start = 1)  
  
posterior\_summary(weibull\_samples\_pre$theta)

## $mean  
## [1] 0.8378507  
##   
## $quantiles  
## 2.5% 50% 97.5%   
## 0.7066770 0.8374230 0.9707844

posterior\_summary(weibull\_samples\_post$theta)

## $mean  
## [1] 0.9958722  
##   
## $quantiles  
## 2.5% 50% 97.5%   
## 0.8705424 0.9951131 1.1280295

posterior\_summary(weibull\_samples\_pre$lambda)

## $mean  
## [1] 3.41187  
##   
## $quantiles  
## 2.5% 50% 97.5%   
## 2.625006 3.385823 4.357510

posterior\_summary(weibull\_samples\_post$lambda)

## $mean  
## [1] 2.536561  
##   
## $quantiles  
## 2.5% 50% 97.5%   
## 2.117619 2.526137 3.005464

# Look at median of distribution via parameters  
posterior\_summary(weibull\_samples\_pre$lambda \* log(2)^(1/weibull\_samples\_pre$theta))

## $mean  
## [1] 2.199436  
##   
## $quantiles  
## 2.5% 50% 97.5%   
## 1.647949 2.183678 2.847735

posterior\_summary(weibull\_samples\_post$lambda \* log(2)^(1/weibull\_samples\_post$theta))

## $mean  
## [1] 1.754006  
##   
## $quantiles  
## 2.5% 50% 97.5%   
## 1.438488 1.747281 2.106025

###############  
# E. How well does pre-election model/data do for post-election realized data?  
###############  
  
weibull\_samples\_pre <- weibullSamp(  
 seed = 1, data = pre\_election, B = 20000,  
 a1 = 60, b1 = 67,  
 theta\_start = 0.5, lambda\_start = 1)  
  
posterior\_summary(weibull\_samples\_pre$theta)

## $mean  
## [1] 0.8378507  
##   
## $quantiles  
## 2.5% 50% 97.5%   
## 0.7066770 0.8374230 0.9707844

posterior\_summary(weibull\_samples\_pre$lambda)

## $mean  
## [1] 3.41187  
##   
## $quantiles  
## 2.5% 50% 97.5%   
## 2.625006 3.385823 4.357510

plot(density(post\_election, from=0), lwd=2, ylim=c(0, 0.4))  
curve(dweibull(x, 0.8378, 3.4118), add=TRUE, col="red", lty=2, lwd=2)

