Bayesian modeling in Stan

Jason Doll, PhD

Quantitative Fisheries Center

$$P(\mathrm{H}|D) = \frac{P(D|\mathrm{H}) * P(\mathrm{H})}{P(D)}$$

Workshop Outline

Morning

- Bayesian basics
- Stan background, syntax, and coding best practices
- Simple examples
  - Normal distribution
  - Linear regression
- Common errors and techniques to address them

Afternoon

- Addressing common errors, cont.
- ShinyStan
- Mixed effects
- Model Selection

Learning Outcomes

- Learn Stan coding syntax
- Learn basic parameter estimation using Bayesian inference with Stan
- Learn ways to address common errors
- Learn how to do model selection/comparison using Stan

Workshop Structure

- Lecture and Exercises
- All *.R files provided
- Data will either be simulated in *.R file or using base R data files
- You will create all *.stan files

Prerequisites

- R
- RStudio
- R packages
  - rstan
  - shinystan
  - Loo
  - ggplot2
- Git
- Or .zip folder from D2L

https://github.com/jcdoll79/IntroStan.git

Download GitHub files
Install software

FEATURE

Introduction to Bayesian Modeling and Inference for Fisheries Scientists

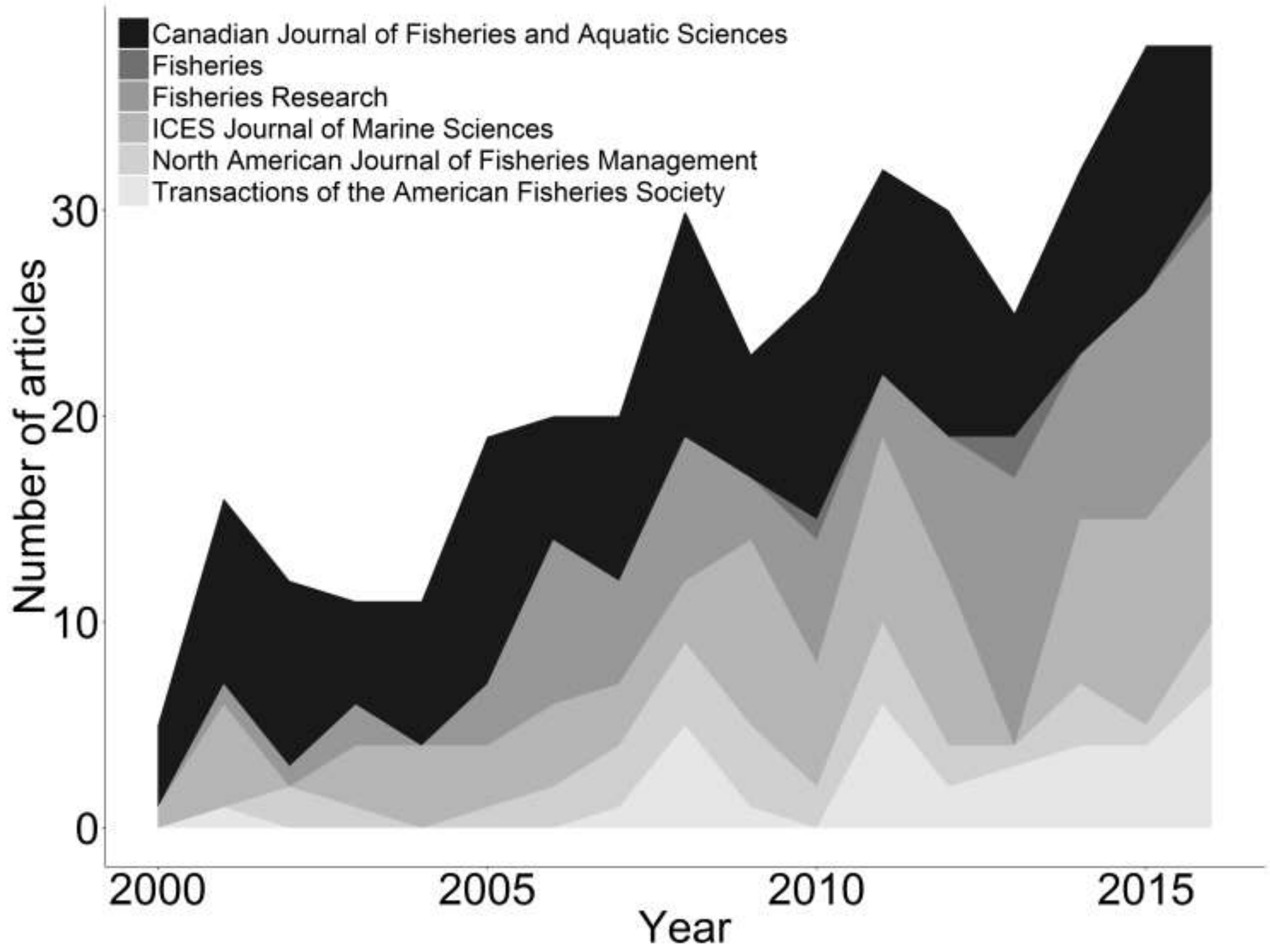
Jason C. Doll | Quantitative Fisheries Center, Department of Fisheries and Wildlife, Michigan State University, 375 Wilson Road, Room 100, East Lansing, MI 48824. E-mail: dolljas1@msu.edu

Stephen J. Jacquemin | Department of Biological Sciences, Wright State University–Lake Campus, Celina, OH

Photo credit: DaveAlan/Getty Images and Aleksandar Velasevic/iStockphoto (background image)

© 2018 American Fisheries Society
DOI: 10.1002/fsh.10038

Keyword search: “Bayesian”



Frequentist vs. Bayes

- Frequentist
  - Traditional inference methods
  - Views probability as frequency of occurrence
  - Parameters are fixed
  - *P*(D|H)
  - Asks: what is the probability of getting these data based on a hypothesis?
  - P-values, H_0 testing and H_1
  - Inference from point estimates
  - 95% confidence intervals

"Only a Sith deals in absolutes"
-Obi-Wan Kenobi,
a long time ago

Frequentist vs. Bayes

- Bayesian
  - Views probability as a measure of belief
  - Parameters are random
  - Probability statement conditional on data and prior
  - $P(\mathrm{H}|\mathrm{D})$
  - Asks: what is the probability of this hypothesis based on the data, model, and prior?
  - Inference from posterior distributions
  - Bayesian 95% credible intervals

Advantages of being Bayesian

- Conclusions conditional on the data
- Easily handle latent variables and functions of latent variables
- Incorporate prior, expert, knowledge
- Incorporate pilot study results
- Conceptually more intuitive
- Express your findings in terms of probabilities, easier for non-scientists to understand

What is Bayesian inference?

- A different approach to ALL statistical inference problems
  - i.e., not just another tool in the trunk like: maximum likelihood, SEM, survival analysis, multivariate analysis

What is Bayesian inference?

- Uses probability theory to quantify the strength of arguments
  - e.g., There is a 95% probability that the true mean is between 6 and 10

What is Bayesian inference?

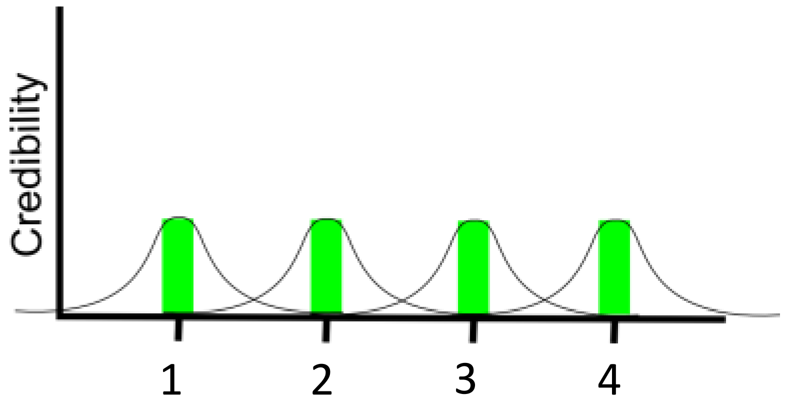
- Re-allocating belief about an event or parameter

Re-allocating belief

- Suppose you submit your first manuscript to a journal.
- The journal has a 25% acceptance rate.
- Your manuscript gets accepted!
- Now what is your assessment of the probability of your next submission to that journal on a related topic being accepted?
- 100%? 25%? 62.5%? Why?

Re-allocating belief

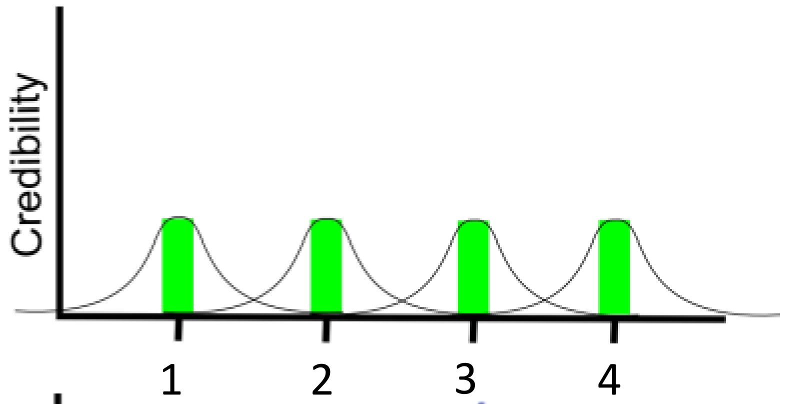
Prior

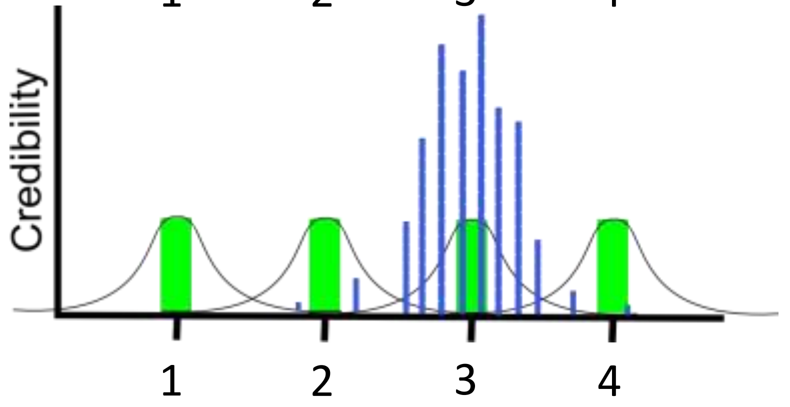


Re-allocating belief

Prior

Data



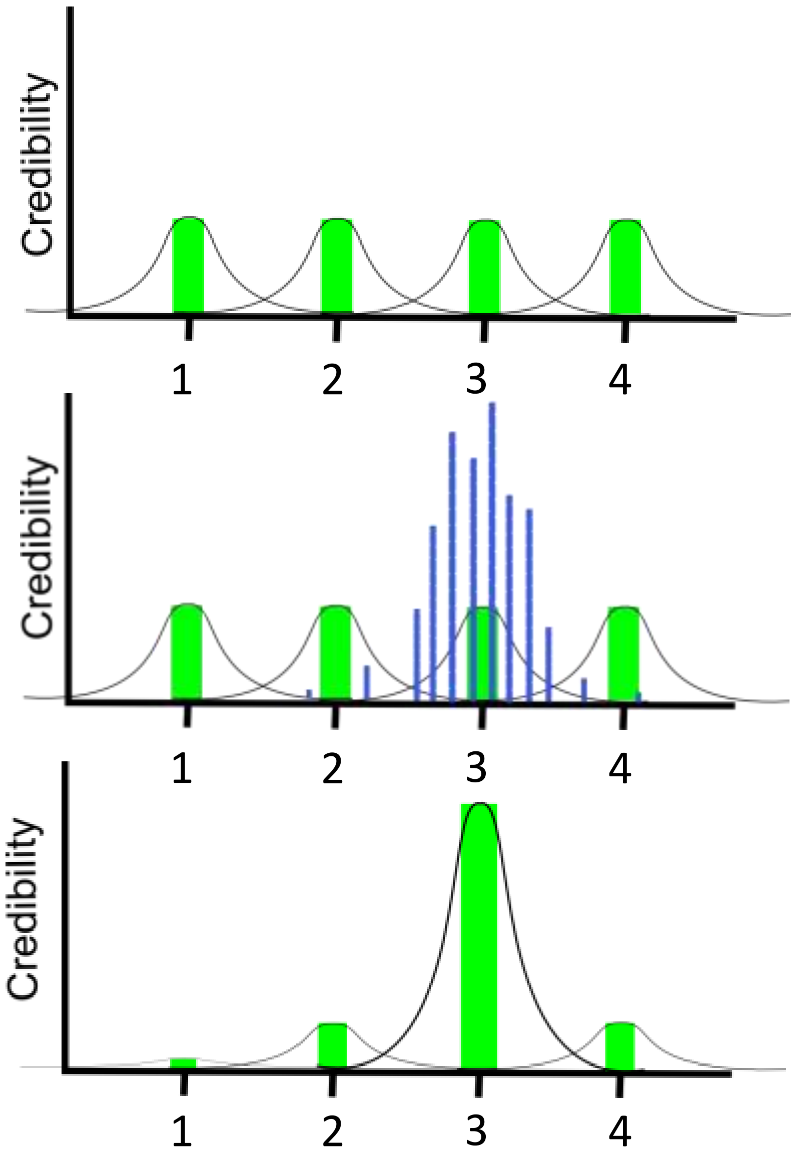


Re-allocating belief

Prior

Data

Posterior



How Bayesian inference works

- We have Y
  - The Data
- We want to know θ
  - Parameters of the model
- We assume Y produced by θ
  - “Data Generation Mechanism”
- We build a mathematical ***model*** of the data generation mechanism

Model

- Definitions
  - Y = Data
  - θ = Parameters of the model
  - X = Covariates
- The model is assumed to produce the data

Likelihood

- Model answers the question:
  - What is the probability of obtaining the data given some parameters

    $P(\mathrm{D}|\theta)$
- This is the likelihood
- We want to use this to learn about the parameters, θ
- We want to know

  $P(\theta|\mathrm{D})$
- We need to invert the probability!

Enter Bayes' Theorem

$$P(\theta|D) = \left(\frac{P(D|\theta) * P(\theta)}{P(D)}\right)$$

- Allows us to invert the probability
- But what are $P(\theta)$ and $P(D)$?

P(θ)

$$P(\theta|D) = \left(\frac{P(D|\theta) * P(\theta)}{P(D)}\right)$$

- *P*(θ) is a measure of our prior belief about the parameters
- Not conditioned on the data
- Interpreted as the probability before we see the data
- Called the ***prior distribution***
- Before we see data we have some idea of what the values might be
- Can be non-informative (e.g., ± 1,000)
- Can be informative (e.g., ↑1 mm = ↑ 0.5 – 1.5 g)

$P(D)$

$$P(\theta|D) = \left(\frac{P(D|\theta) * P(\theta)}{P(D)}\right)$$

- Unconditional distribution of the data
- Marginal distribution of the data
- Only depends on D, which is constant
  - P(D) is a constant

Bayes' Theorem

$$P(\theta|D) = \left(\frac{P(D|\theta) * P(\theta)}{P(D)}\right)$$

- With *P*(D) constant all we need to estimate *P*(θ|D) are
  - *P*(D|θ) and *P*(θ)
- Bayes' Theorem becomes

$$P(\theta|D) \propto P(D|\theta) * P(\theta)$$

- *P*(θ|D) is called the ***posterior distribution***

Bayesian inference cont.

- Quantify and propagating uncertainty
  - Defined as probability
  - From Prior → Posterior
- This was completed by integration
  - Difficult to do with high dimensional parameter space.
- Enter MCMC!

MCMC

- Markov Chain Monte Carlo (Gelfand & Smith 1991)
  - Revolutionized Bayesian inference
  - Constructs Markov chains
    - Series of steps where the next step is only conditional on the current step and nothing else.
  - MCMC simulation approximates the true posterior density, $p(\theta|y)$, using a bag of samples drawn from the density

MCMC

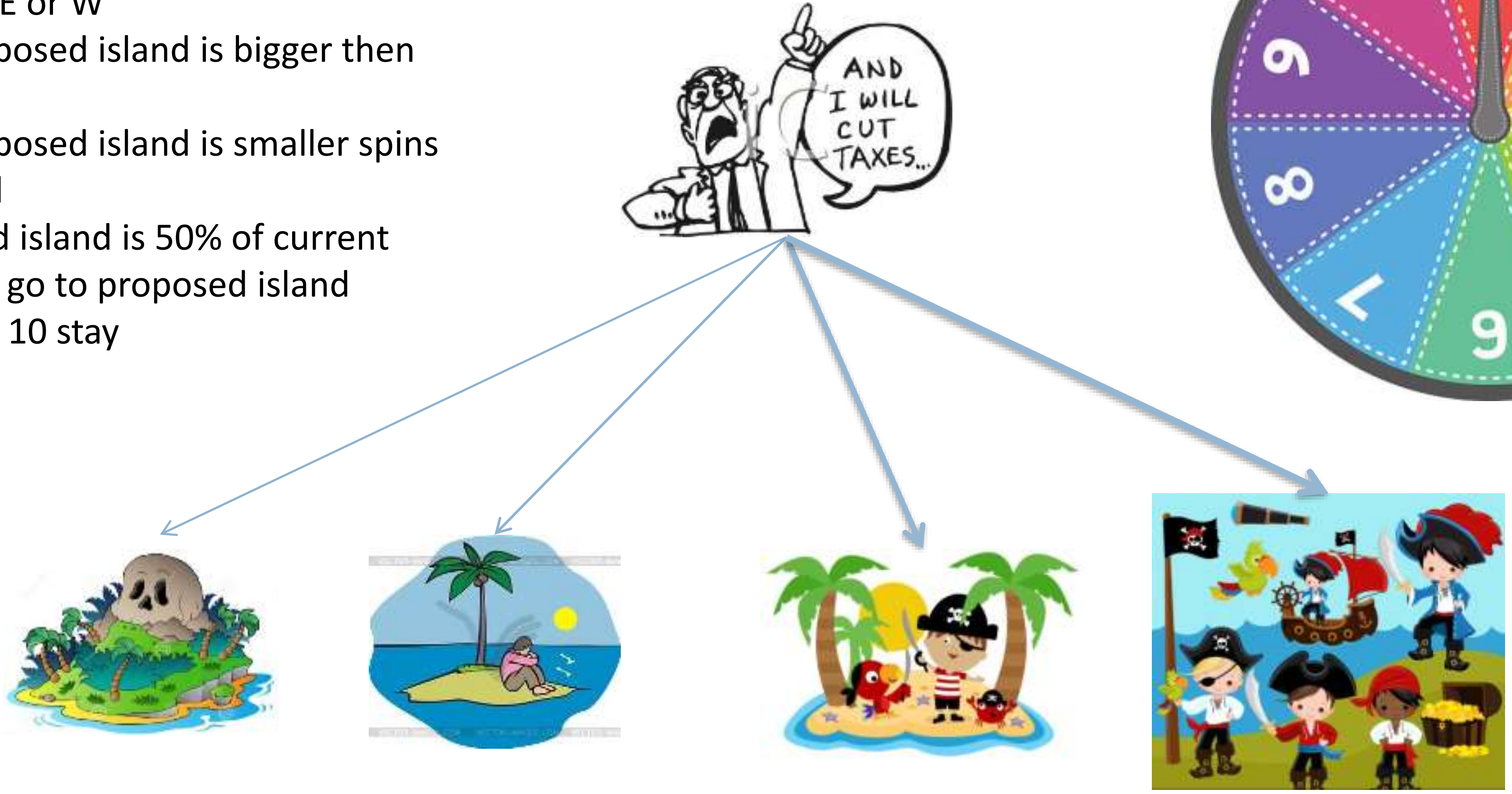
- An iterative procedure that makes a probabilistic update to θ^{i+1} given the current state of the chain θ^{i}

```
$\theta^{0} \leftarrow x$
For i = 1 to M
        $\theta^{i} = f(\theta^{i-1})$
Next i
```

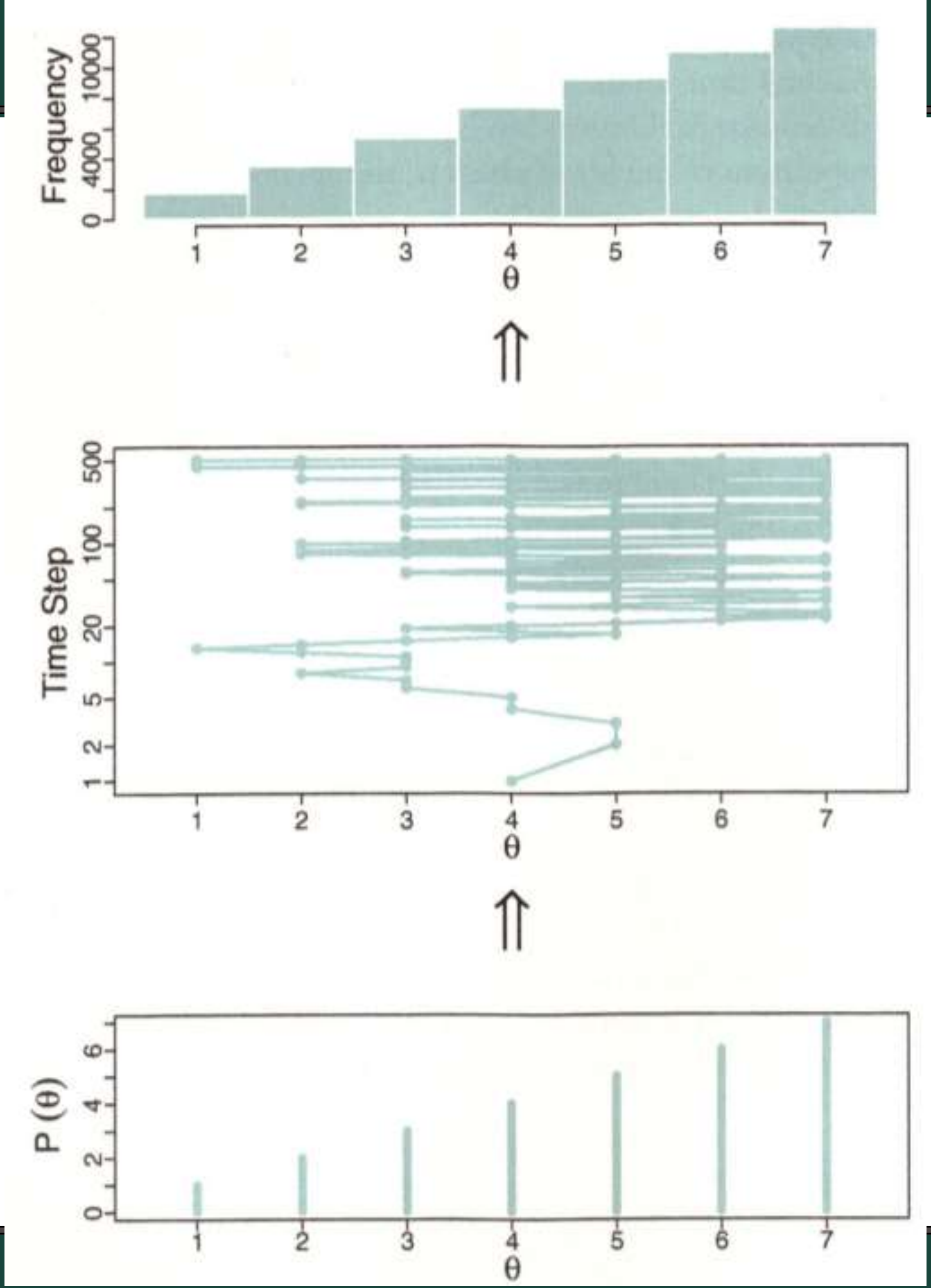
- The update, f(), is made so that the distribution of $p(\theta^{i}) \rightarrow p(\theta | y)$, as $i \rightarrow \infty$ for any starting value.

Metropolis-Hastings

1. Flip coin – E or W
   - If proposed island is bigger then goes
   - If proposed island is smaller spins wheel
2. If proposed island is 50% of current
   - 1 to 5 go to proposed island
   - 5.1 to 10 stay



Example from Kruschke 2015



Example from Kruschke 2015

Bayesian workflow

- Determine probability model
- Identify prior probability distributions (informative is better!)
- Gather data
- Generate posterior distribution
- Check convergence
- Draw inference

Break?

Stan

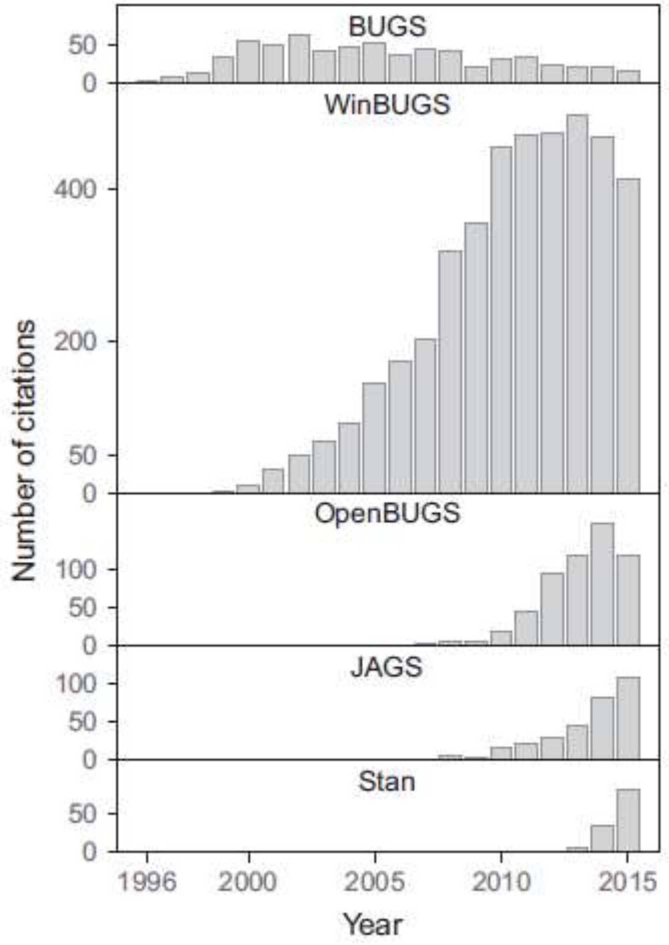


Fig. 1. Citation patterns of Stan and the BUGS family of Bayesian software platforms, for all journals in all fields. Data are from ISI Web of Science Core Collection. The *y*-axis units are the same, despite variable ranges.

Monnahan et al. 2017

What is Stan?

- "Stan is a package for obtaining Bayesian inference using the No-U-Turn sampler, a variant of the Hamiltonian Monte Carlo."
- http://mc-stan.org/
- Created to reconstruct climate based on tree-ring data
  - JAGS/BUGS would not converge after hundreds of thousands of iterations
  - Hamiltonian Monte Carlo converged with a few hundred

Schofield, M.R., R.J. Barker, A. Gelman, E.R. Cook, K.R. Briffa. 2016. A model-based approach to climate reconstruction using tree-ring data. Journal of the American Statistical Association. 111(513):93-106.

What is Stan?

- Very active developers and user base
- Coded in C++ and runs on all major platforms
- Can be ran via command-line terminal
- Not very similar to BUGS, more procedural
  - Order matters
- Fast – compile to execute file
- Easy to use with Stan interfaces
- Easy to run in parallel (`cores=3`)

Stan interfaces and platforms

- RStan v2.17.3 (R)
- PyStan (Python)
- CmdStan (command-line terminal)
- MatlabStan (MATLAB)
- Stan.jl (Julia)
- StataStan (Stata)
- MathematicaStan (Mathematica)

Mac OS

Windows 10

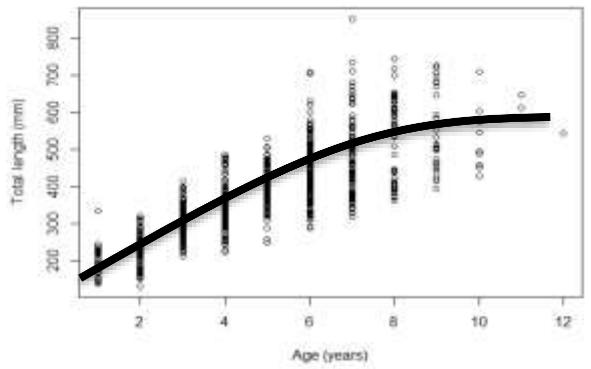
No-U-Turn sampler

- Extension of Hamiltonian Monte Carlo.
- Not sensitive to correlated parameters (like many MCMC methods).
- Generally converges more quickly than Metropolis or Gibbs.
- Details beyond the scope of this presentation.

Monnahan, C.C., J.T. Thorson, and T.A. Branch. 2017. Faster estimation of Bayesian models in ecology using Hamiltonian Monte Carlo. Methods in Ecology and Evolution 8:339-348.

Why use Stan?

- More efficient sampler
- Nonlinear model
- N = 60
- Iter = 10,000, warmup = 1,000, thin = 3, chains = 3
- BGR/Rhat < 1.10



MCMC Efficiency = Effective Sample Size / Computation time (sec)

	Stan	JAGS
Mean ESS/sec	499	20
Min ESS/sec	434	20
Max ESS/sec	612	21

Posterior Inference

- Coda-like posterior summary
  - Posterior mean, SD, quantiles, etc.
  - Gelman/Rubin statistics for convergence diagnostics
  - Multi-chain effective sample size estimation
- ShinyStan for convergence diagnostics
- Model comparison with WAIC

RStan Workflow

- Write program in a .stan file
- Package everything up (.stan file, data list, initialization list, etc.)
- Submit to Stan in R
  - Checks that program is syntactically valid
  - Writes C++ source file to disk
- C++ complier creates binary file from source
- You execute the binary file from R
- You analyze posterior distribution

Stan language

- Variables must be declared with types

Data Types

Description	Example
Primitive types:	
integer	`int N;`
continuous	`real TL[N];`
Matrix	`matrix[20,10] F;`
Vector	`vector[G] Linf;`
Bounded data types	`vector<lower=0, upper=2000>[G] Linf;`
Arrays of any dimensionality	`real N_mo[Y,A,M];`

Stan language

- Variables must be declared with types
- Statements are terminated with `;`
- `Use` = for assignments, `<-` is depreciated
- Can include files within a file, `#include model1.stan`
- `// for comment on one line`
- `/* comment here for multiple lines */`
- Space, tab, carriage return, line feed all treated as whitespace
- `print(…);`

Distributions (partial list)

Discrete	Continuous
Bernoulli	Normal
Binomial	Student-*t*
Beta-Binomial	Double exponential
Categorical	Lognormal
Negative Binomial	Beta
Poisson	Gamma
Multinomial	Multivariate normal

Basic Program Blocks

- data
  - Declare data types, sizes, and constraints
- parameters
  - Declare parameter types, sizes, and constraints
- model
  - Statements defining posterior density
  - Prior distributions and likelihood

Other Program Blocks

- functions
  - Declare and define user specified functions
- transformed data
  - Declare and define transformation of data variables
- transformed parameters
  - Declare and define transformation of parameters
- generated quantities
  - Declare and define generated quantity variables, pseudo-random number generators, and WAIC

Block order

```
functions{ }
data{ }
transformed data{ }
parameters{ }
transformed parameters{ }
model{ }
generated quantities{ }
```

Variable Scope

- Variables declared in each block have scope over all *subsequent* statements.
- Ex: Declare variable in transformed parameter block
  - Can be used in model or generated quantities block
  - Can't be used in parameter, transformed data or data block
- Exception!
  - Variables declared in model block ONLY valid in model block

What we have covered so far

- We remembered how cool Bayesian inference is
- Stan is efficient and awesome
- Stan uses common programming conventions (e.g., ; and =)
- Stan data types
- Stan probability distributions
- Stan requires "Blocks" in a specific order

Format of Exercises

A. Describe data and model
B. Review R code (provided)
C. Review Stan code
D. You program the *.stan file following the screen or handout
E. Enjoy your results!

(A) Exercise 1: Mean and SD

- We will randomly generate data with known parameters
- Construct a Bayesian model to describe the mean and SD
- Estimate mean and SD using Stan

(A) Estimating mean and SD

Model: $\mathrm{y}_i = \mu + \varepsilon_i$

$\varepsilon_i \sim \mathrm{Normal}(0, \sigma)$

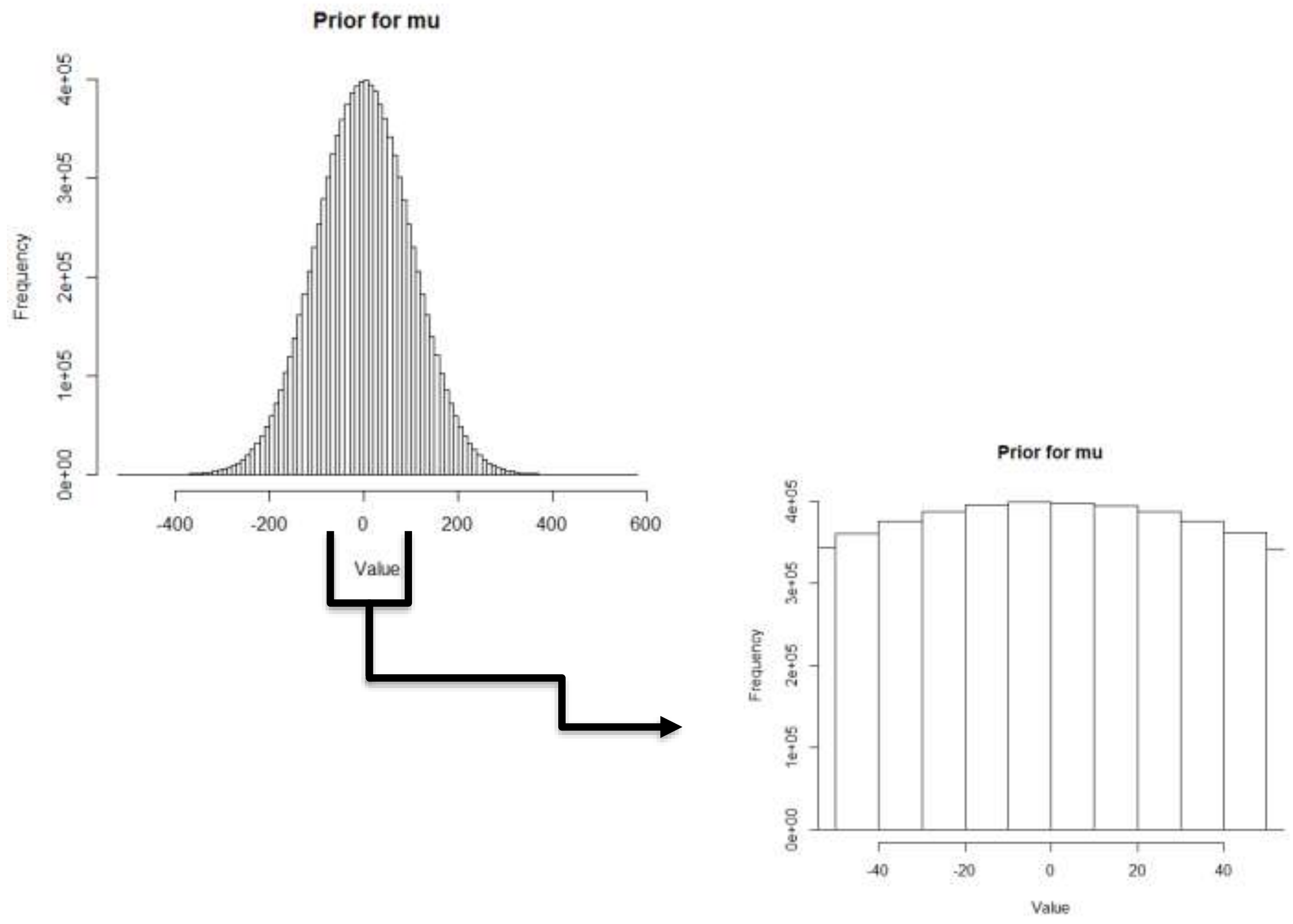
Priors: $\mu \sim \mathrm{Normal}(0{,}100)$

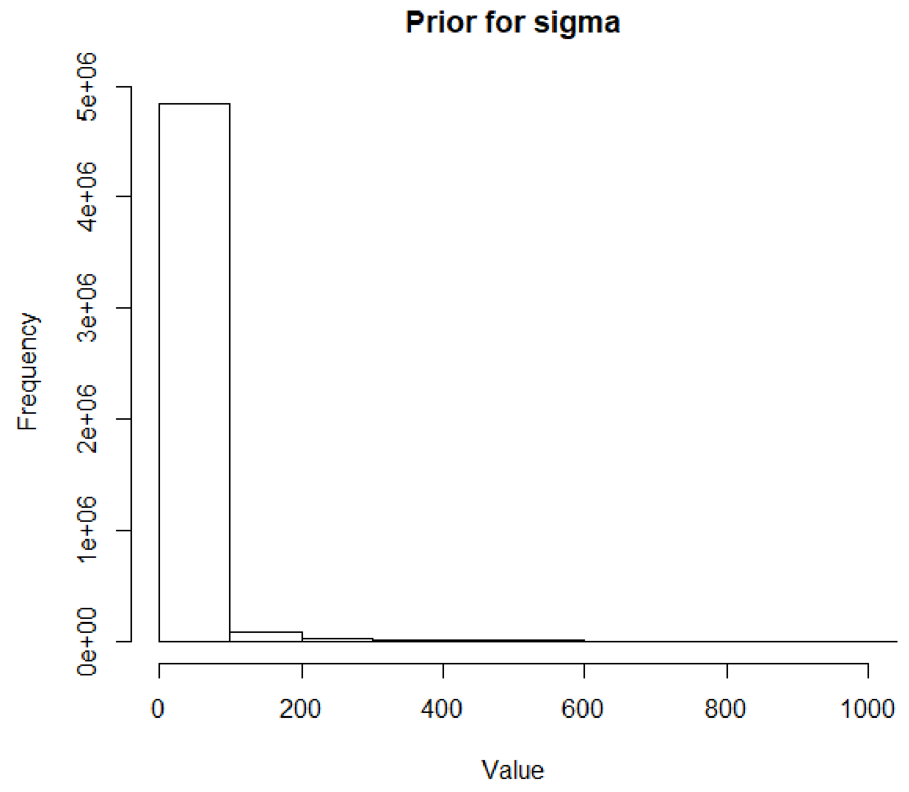
$\sigma \sim \mathrm{half} - \mathrm{cauchy}(0{,}5)$

- Model and priors

$\mu \sim \text{Normal}(0,100)$

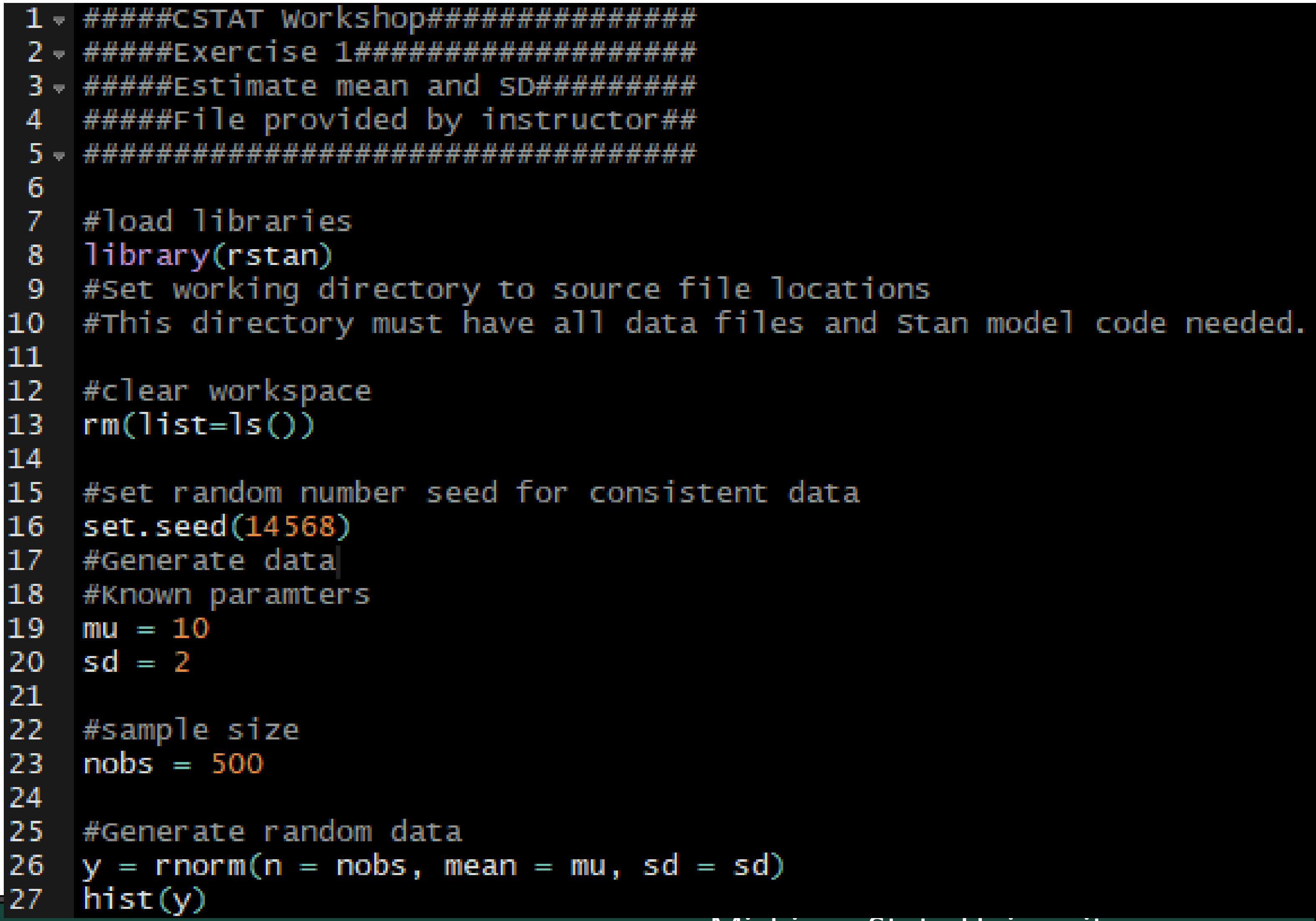
$\sigma \sim \text{half} - \text{cauchy}(0,5)$





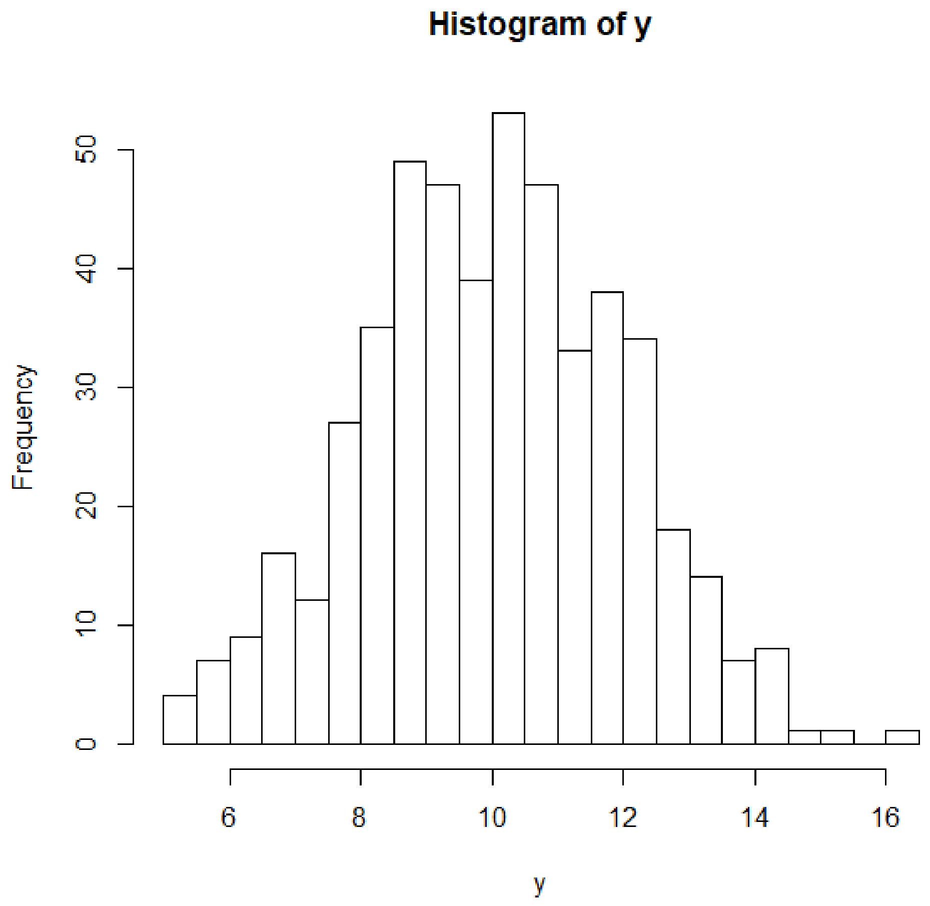
(B) Exercise 1: Mean and SD

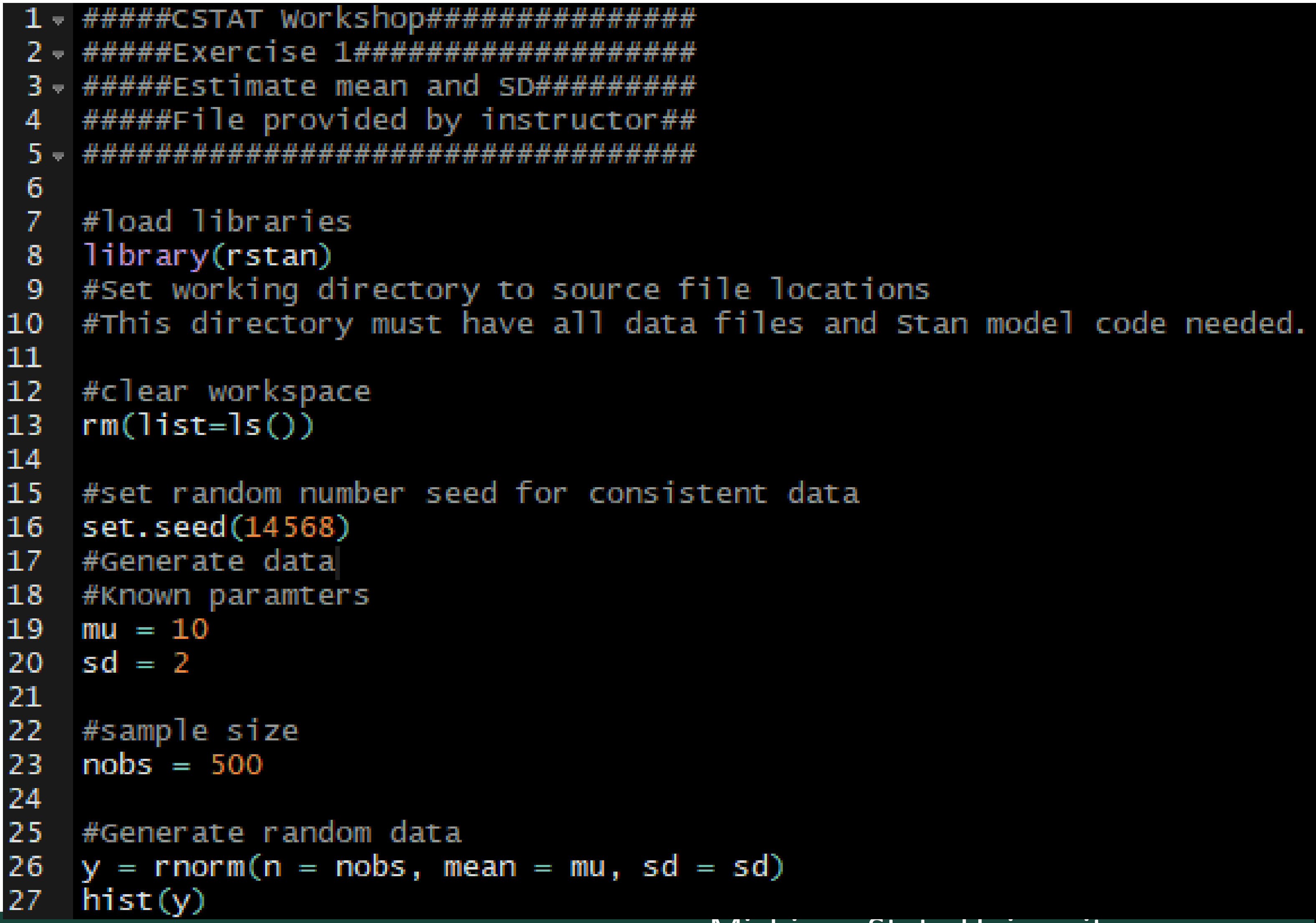
1. Open "Ex1_mean.R"
2. Load library
3. Set working directory
4. Clear workspace
5. Generate or load data



```
#####CSTAT Workshop###############
#####Exercise 1###################
#####Estimate mean and SD#########
#####File provided by instructor##
##################################

#load libraries
library(rstan)
#Set working directory to source file locations
#This directory must have all data files and Stan model code needed.

#clear workspace
rm(list=ls())

#set random number seed for consistent data
set.seed(14568)
#Generate data
#Known paramters
mu = 10
sd = 2

#sample size
nobs = 500

#Generate random data
y = rnorm(n = nobs, mean = mu, sd = sd)
hist(y)
```

(B) Exercise 1: Mean and SD





```
#####CSTAT Workshop###############
#####Exercise 1###################
#####Estimate mean and SD#########
#####File provided by instructor##
##################################

#load libraries
library(rstan)
#Set working directory to source file locations
#This directory must have all data files and Stan model code needed.

#clear workspace
rm(list=ls())

#set random number seed for consistent data
set.seed(14568)
#Generate data
#Known paramters
mu = 10
sd = 2

#sample size
nobs = 500

#Generate random data
y = rnorm(n = nobs, mean = mu, sd = sd)
hist(y)
```

(A) Exercise 1: Mean and SD

Model: $y_i = \mu + \varepsilon_i$

$$\varepsilon_i = \mathrm{Normal}(0, \sigma)$$

Priors: $\mu = \mathrm{Normal}(0{,}100)$

$$\sigma = \mathrm{half} - \mathrm{cauchy}(0{,}5)$$

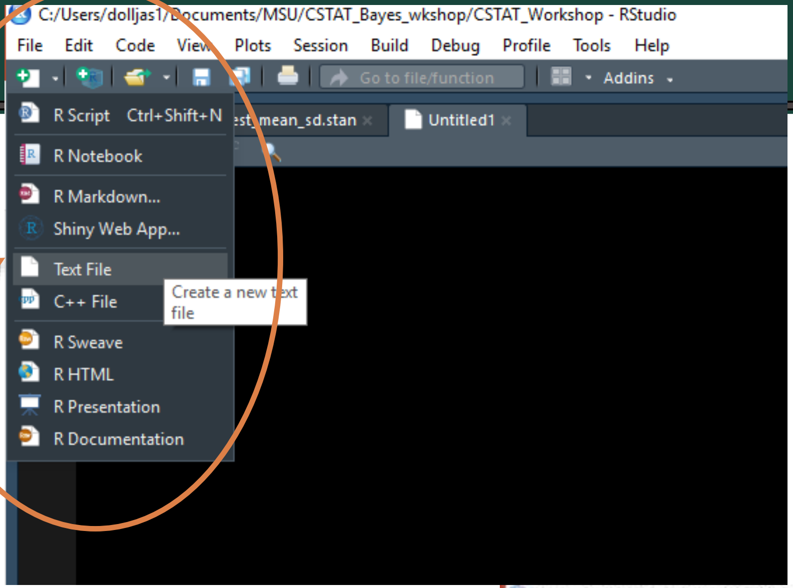
- Model and priors

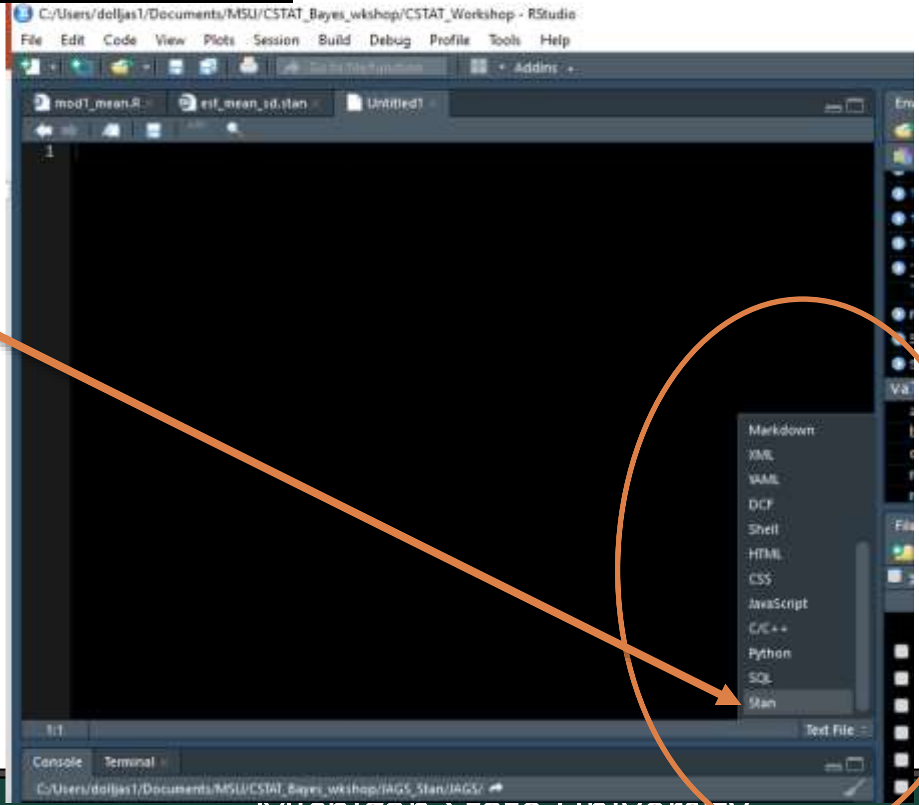
(C) Exercise 1

- Create model file

(C) Exercise 1

- Create new text file
- Change “Text File” to “Stan”




C:/Users/dolljas1/Documents/MSU/CSTAT_Bayes_wkshop/CSTAT_Workshop - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Addins
mod1_mean.R
est_mean_sd.stan
Untitled1
Markdown
XML
YAML
DCF
Shell
HTML
CSS
JavaScript
C/C++
Python
SQL
Stan
Text File
Console
Terminal
C:/Users/dolljas1/Documents/MSU/CSTAT_Bayes_wkshop/JAGS_Stan/JAGS/

(C) Exercise 1: Mean and SD

```
data{
  int<lower=0> n;   //number of observations
  vector[n] y;      //observations as a vector
}
parameters{
  real<lower=0> sigma;    //standard deviation
  real mu;                //mean
}
model{
  //reference priors
  sigma ~ cauchy(0,5);
  mu ~ normal(0,100);

  //likelihood, loop through number of observations
  for(i in 1:n){
    y[i] ~ normal(mu, sigma);
  }
}
```

(D) Exercise 1: data block

```
data{
  int<lower=0> n;  //number of observations
  vector[n] y;     //observations as a vector
}
```

(D) Exercise 1: parameter block

```
parameters{
   real<lower=0> sigma;  //standard deviation
   real mu;                //mean
}
```

(D) Exercise 1: model block

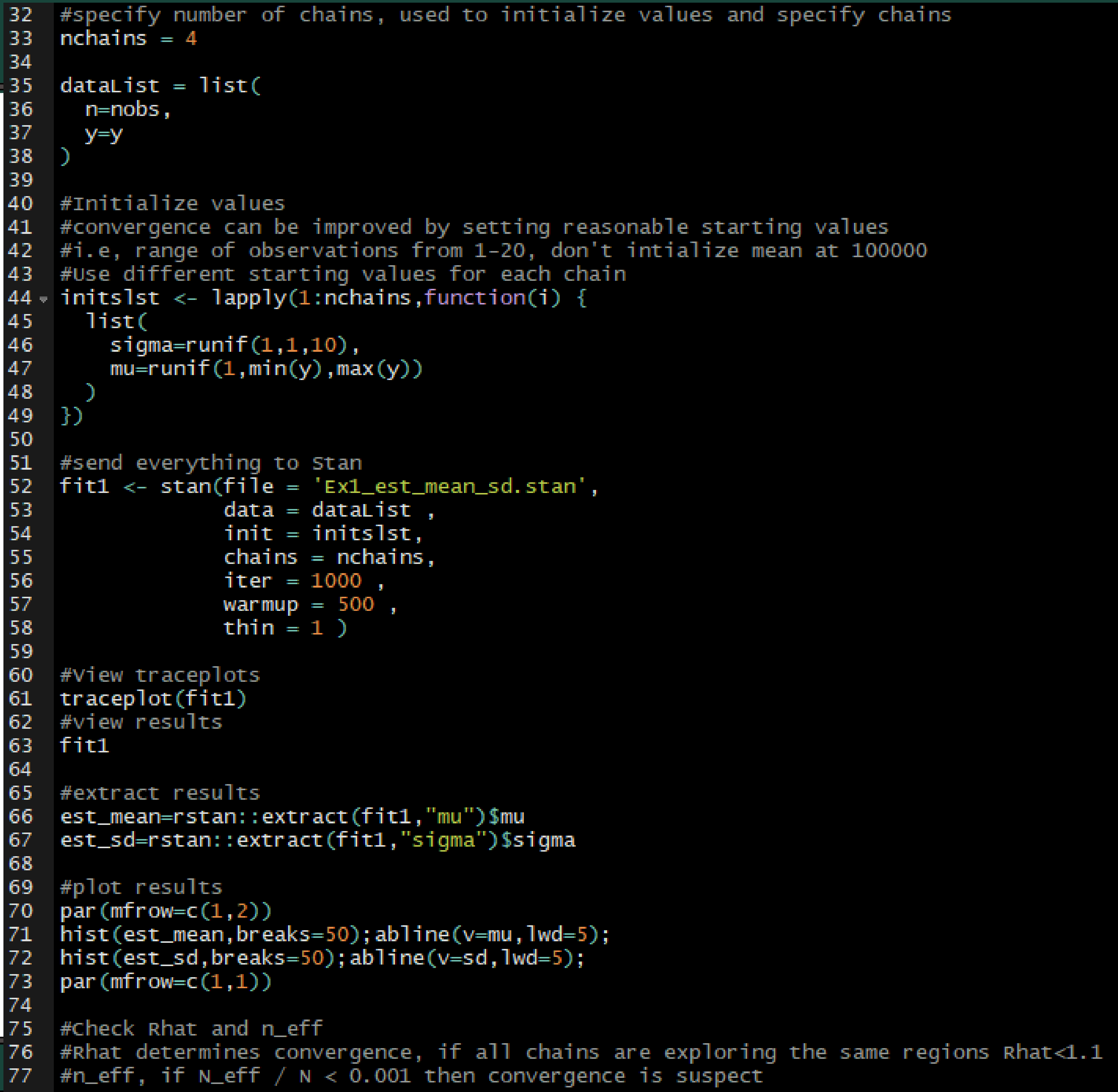
```
model{
  //reference priors
  sigma ~ cauchy(0,5);
  mu ~ normal(0,100);

  //likelihood, loop through observations
  for(i in 1:n){
    y[i] ~ normal(mu, sigma);
  }
}
```

(D) Exercise 1: Mean and SD

- Save Stan code file as “Ex1_est_mean_sd.stan”
- Return to “Ex1_mean.R”

(D) Exercise 1: Mean and SD

6. Specify chains
7. Create data list
8. Create initialization list
9. Send to Stan
10. Misc. functions to explore Stan Output



```
#specify number of chains, used to initialize values and specify chains
nchains = 4

dataList = list(
  n=nobs,
  y=y
)

#Initialize values
#convergence can be improved by setting reasonable starting values
#i.e, range of observations from 1-20, don't intialize mean at 100000
#Use different starting values for each chain
initslst <- lapply(1:nchains,function(i) {
  list(
    sigma=runif(1,1,10),
    mu=runif(1,min(y),max(y))
  )
})

#send everything to Stan
fit1 <- stan(file = 'Ex1_est_mean_sd.stan',
             data = dataList ,
             init = initslst,
             chains = nchains,
             iter = 1000 ,
             warmup = 500 ,
             thin = 1 )

#View traceplots
traceplot(fit1)
#view results
fit1

#extract results
est_mean=rstan::extract(fit1,"mu")$mu
est_sd=rstan::extract(fit1,"sigma")$sigma

#plot results
par(mfrow=c(1,2))
hist(est_mean,breaks=50);abline(v=mu,lwd=5);
hist(est_sd,breaks=50);abline(v=sd,lwd=5);
par(mfrow=c(1,1))

#Check Rhat and n_eff
#Rhat determines convergence, if all chains are exploring the same regions Rhat<1.1
#n_eff, if N_eff / N < 0.001 then convergence is suspect
```

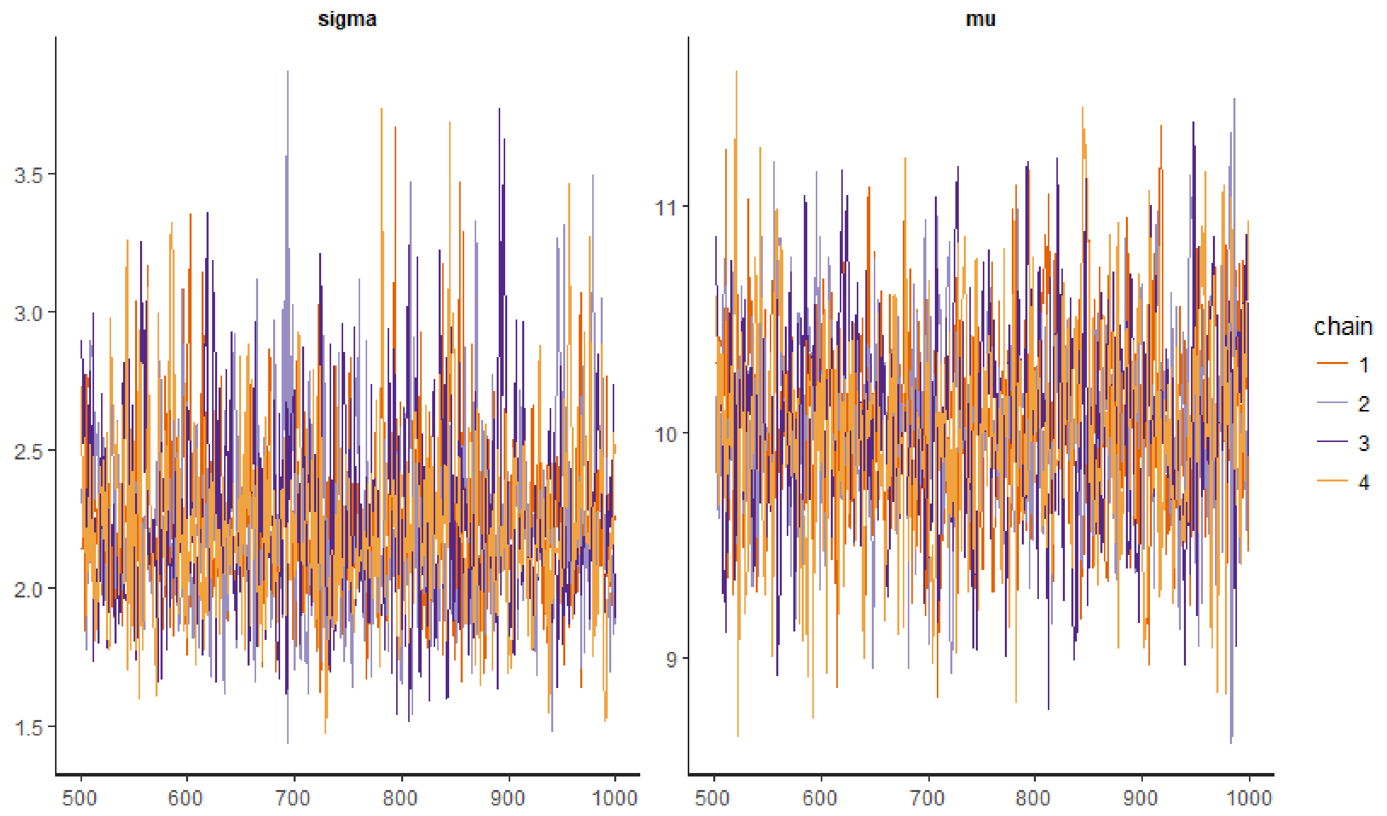
(E) Exercise 1: Mean and SD

- Save Stan code file as “Ex1_est_mean_sd.stan”
- Return to “Ex1_mean.R”
- Run lines 1-58
- Stan compiles program (takes ~30-60 second)
- Compare results to known values

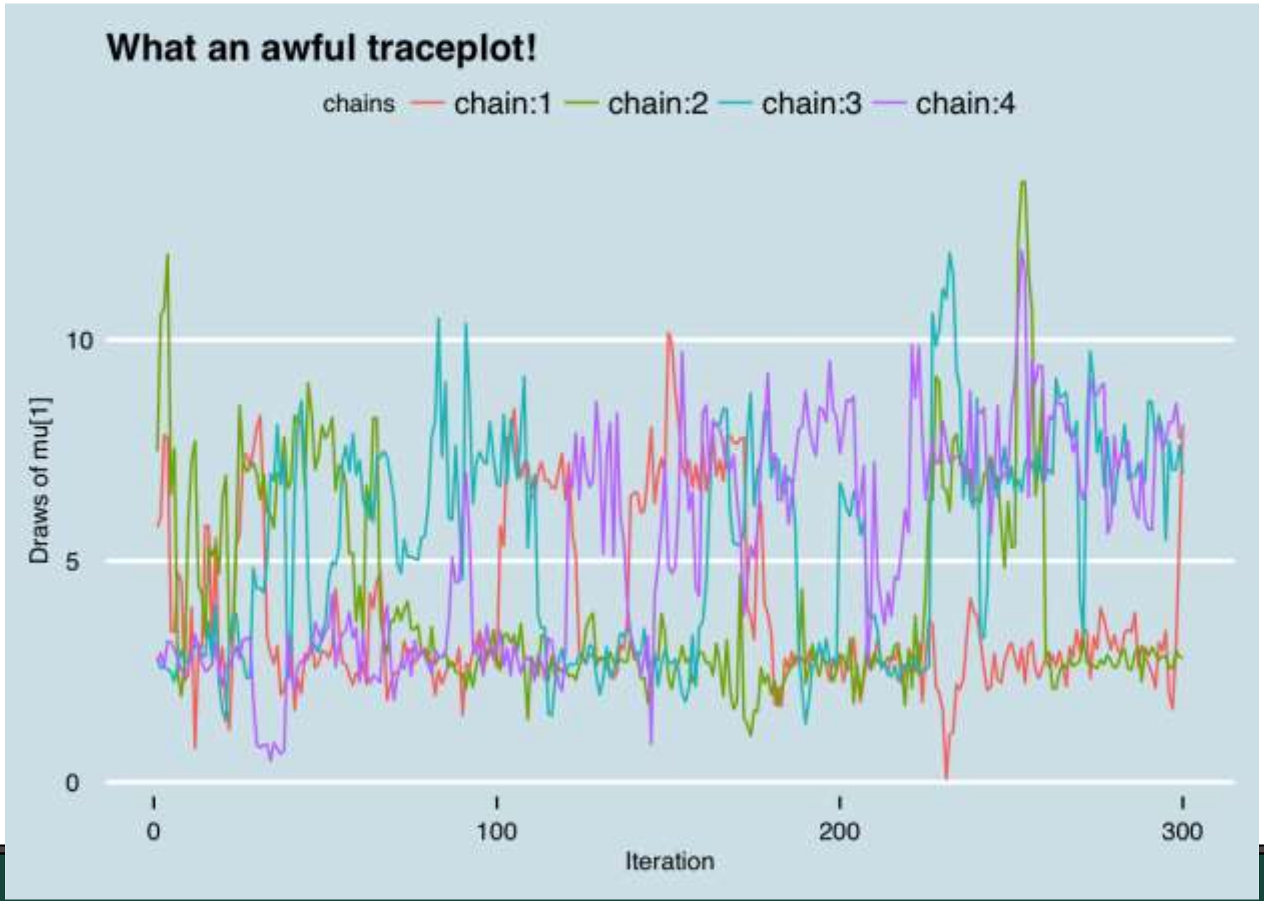
(E) Exercise 1: Mean and SD

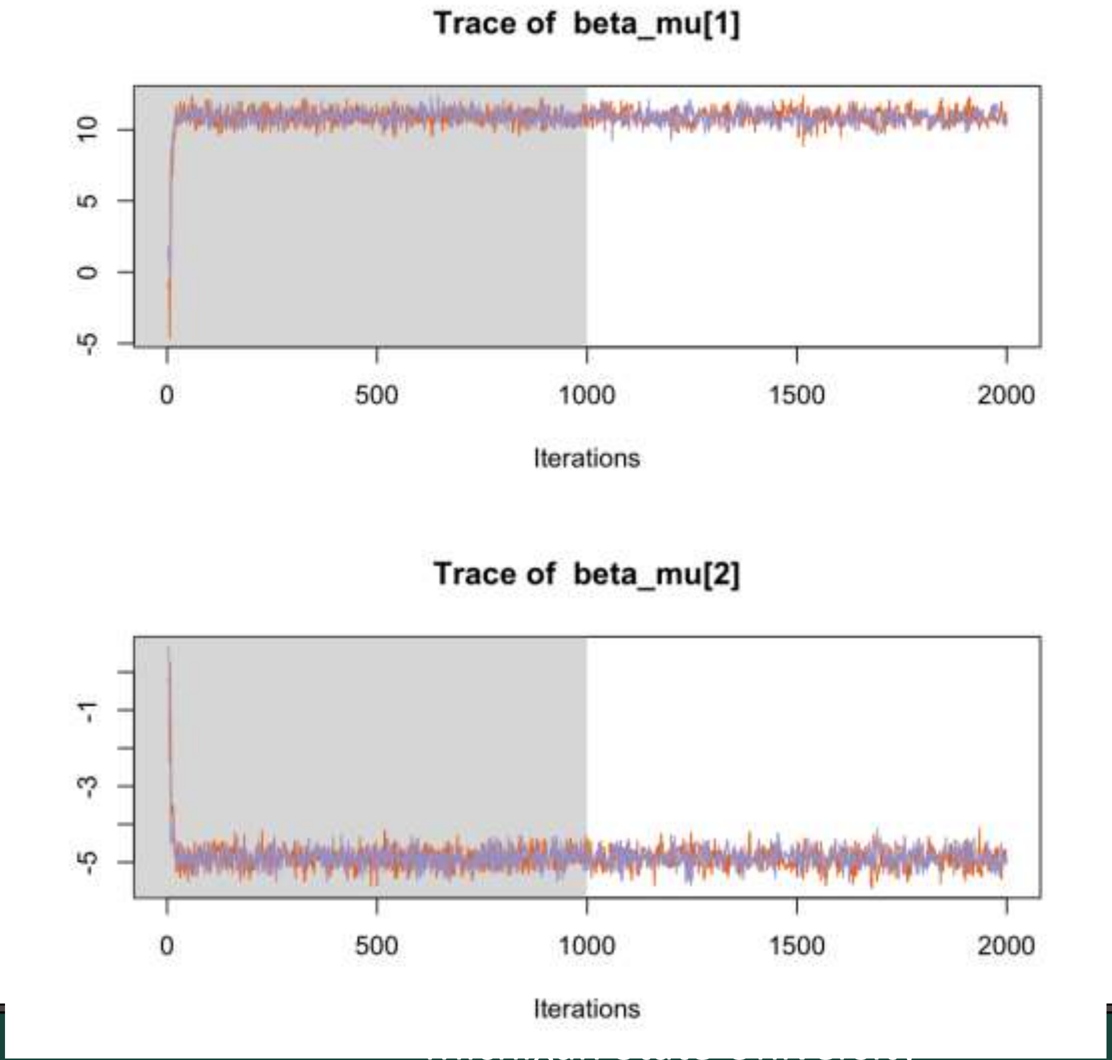
- Review output
  - Compiler warnings
  - Leapfrog steps per transition
- Check Convergence
  - Trace plots
  - Effective Sample Size
  - R-hat

(E) `traceplot(fit1)`

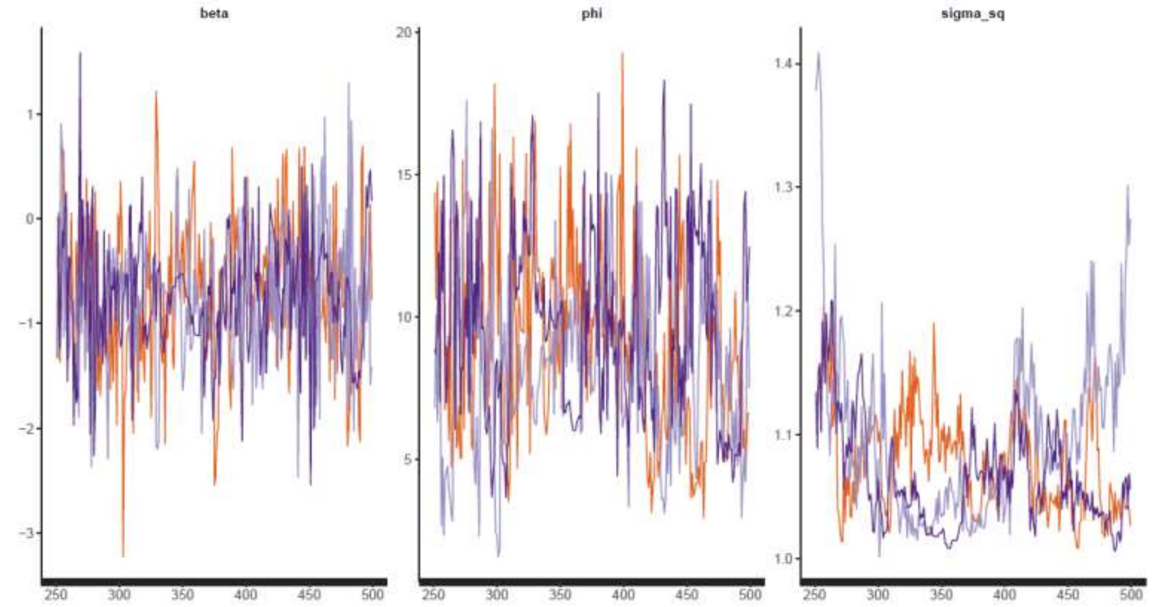


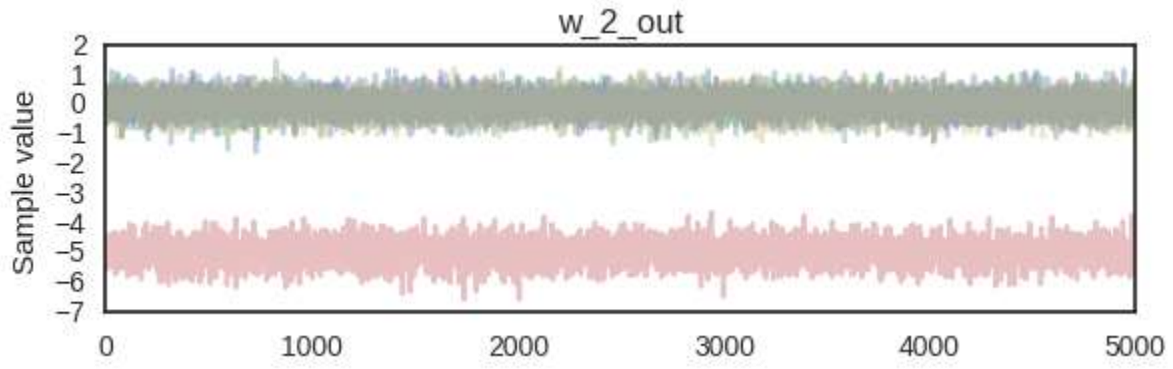
Example of poorly mixing traceplots



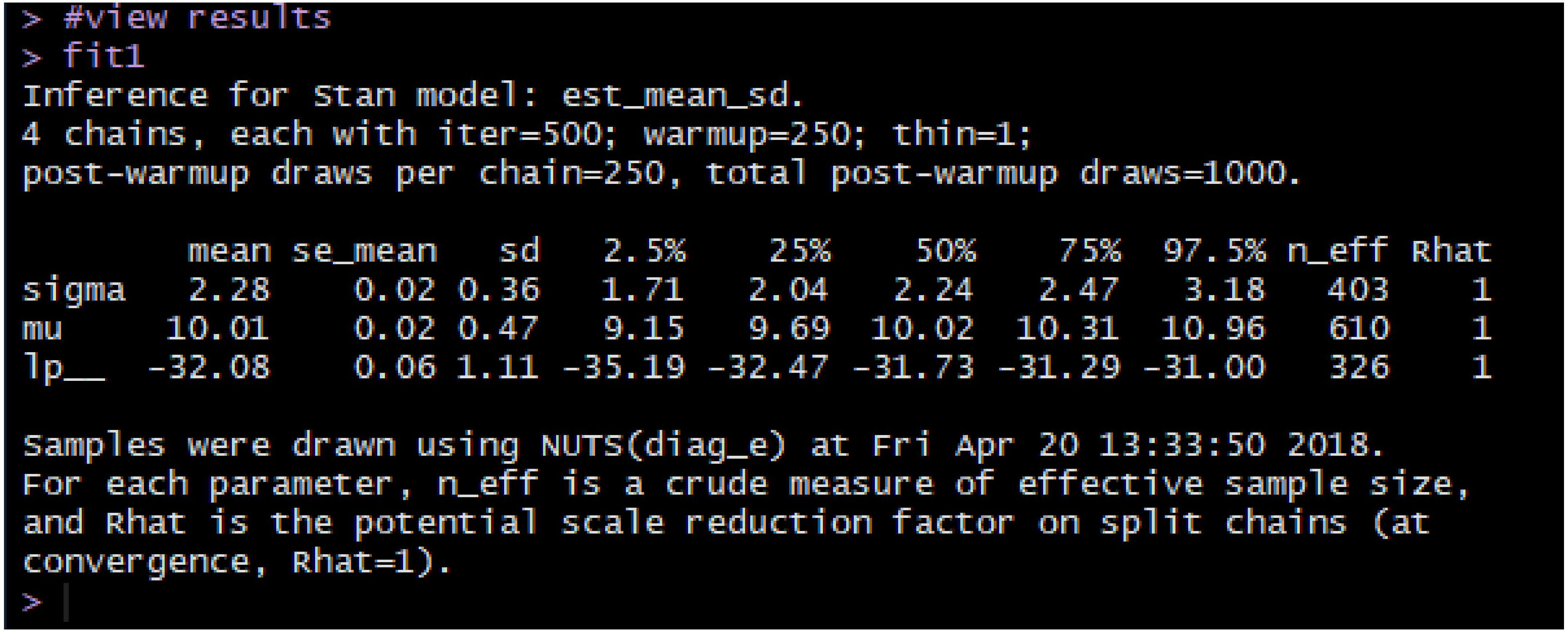


Example of poorly mixing traceplots





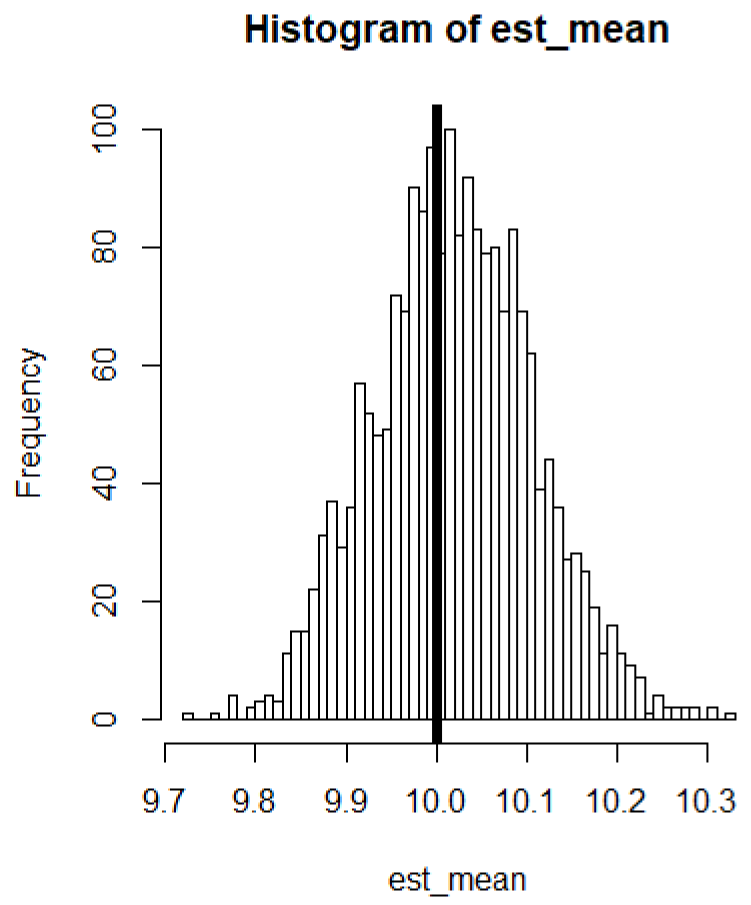
(E) fit1

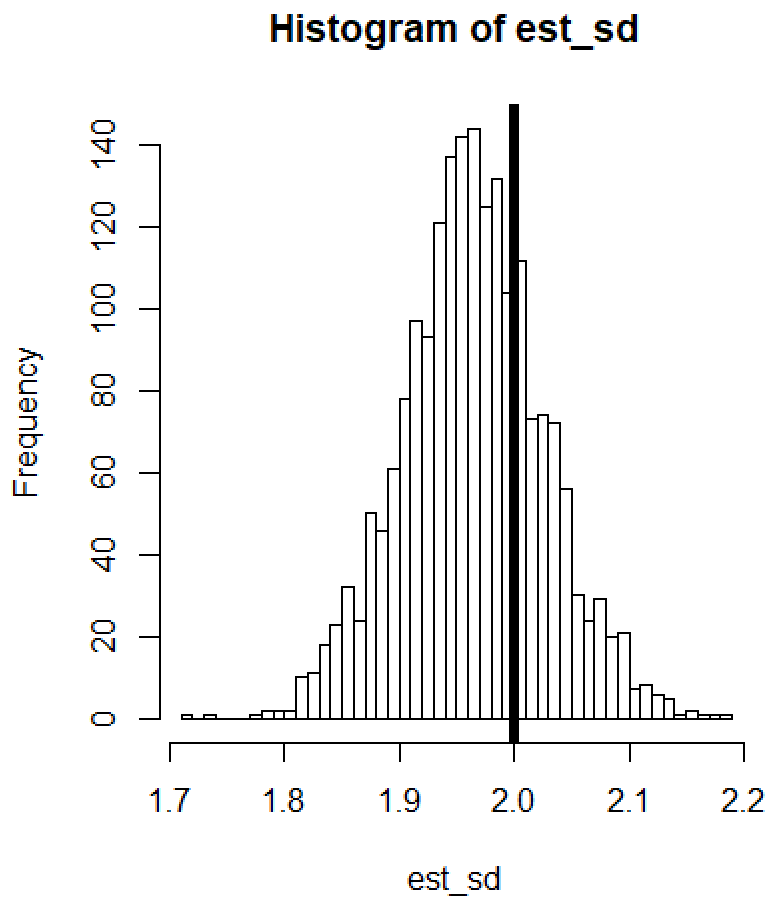


```
> #view results
> fit1
Inference for Stan model: est_mean_sd.
4 chains, each with iter=500; warmup=250; thin=1;
post-warmup draws per chain=250, total post-warmup draws=1000.

        mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
sigma   2.28    0.02 0.36   1.71   2.04   2.24   2.47   3.18   403    1
mu     10.01    0.02 0.47   9.15   9.69  10.02  10.31  10.96   610    1
lp__  -32.08    0.06 1.11 -35.19 -32.47 -31.73 -31.29 -31.00   326    1

Samples were drawn using NUTS(diag_e) at Fri Apr 20 13:33:50 2018.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
>
```

if N_eff / N < 0.001 then convergence is suspect

(E) Compare to known





Questions?

Exercise 2: Linear Regression

A. Describe data and model
B. Review R code (provided)
C. Review Stan code
D. You program the *.stan file following the screen or handout
E. Enjoy your results!

(A) Exercise 2: Linear Regression

- We will randomly generate data with known parameters
- Construct a Bayesian model to describe the regression parameters
- Estimate regression parameters using Stan

(A) Exercise 2: Linear Regression

Model: $$\mathrm{Y}_i = \alpha + \beta \mathrm{X}_i + \varepsilon_i$$

$$\varepsilon_i \sim \mathrm{Normal}(0, \sigma)$$

Priors: $$\alpha \sim \mathrm{Normal}(0{,}100)$$

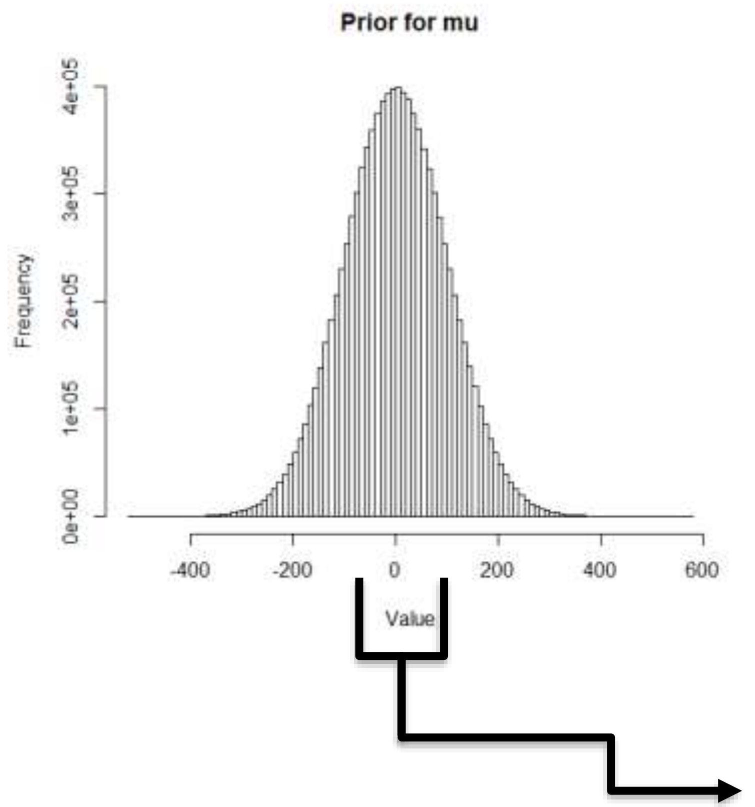
$$\beta \sim \mathrm{Normal}(0{,}100)$$

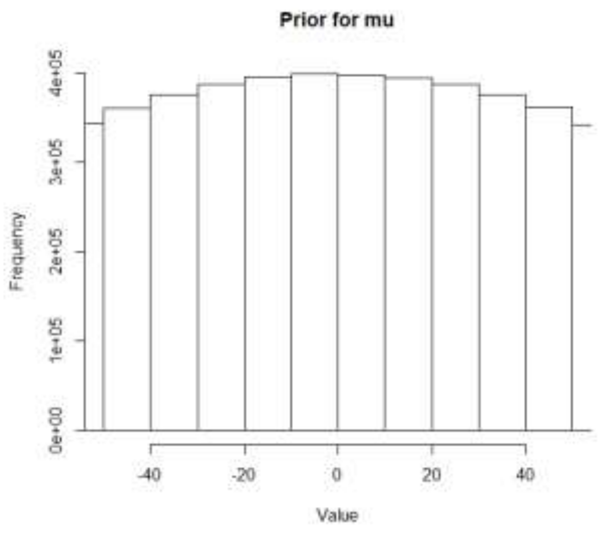
$$\sigma \sim \mathrm{half} - \mathrm{cauchy}(0{,}5)$$

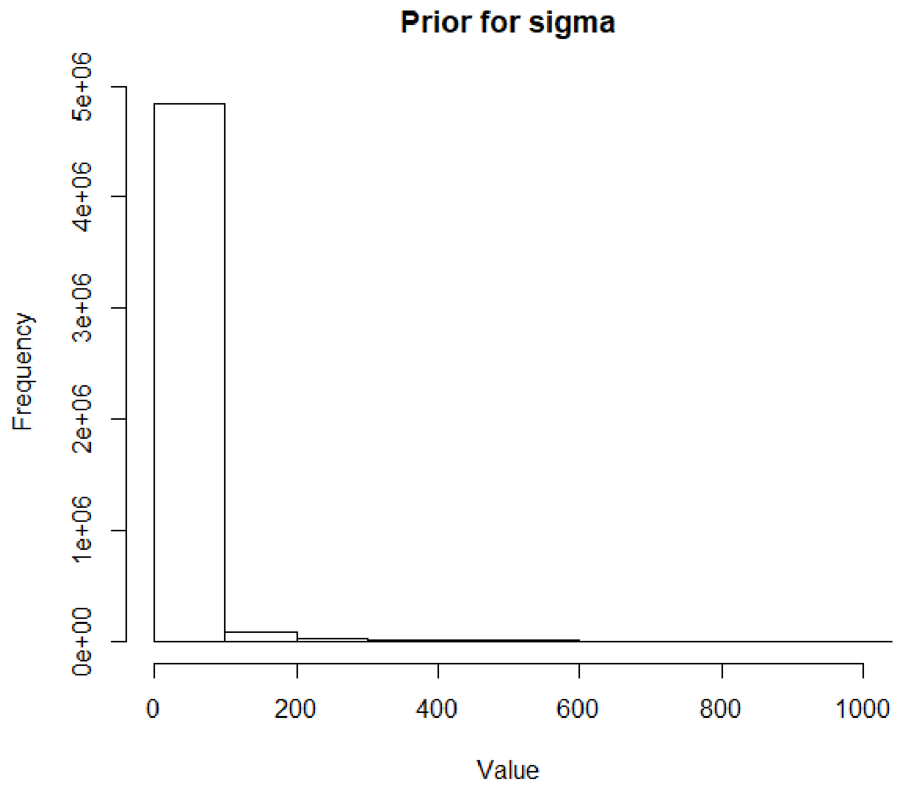
- Model and priors

$$\mu = \text{Normal}(0{,}100)$$

$$\sigma = \text{half} - \text{cauchy}(0{,}5)$$

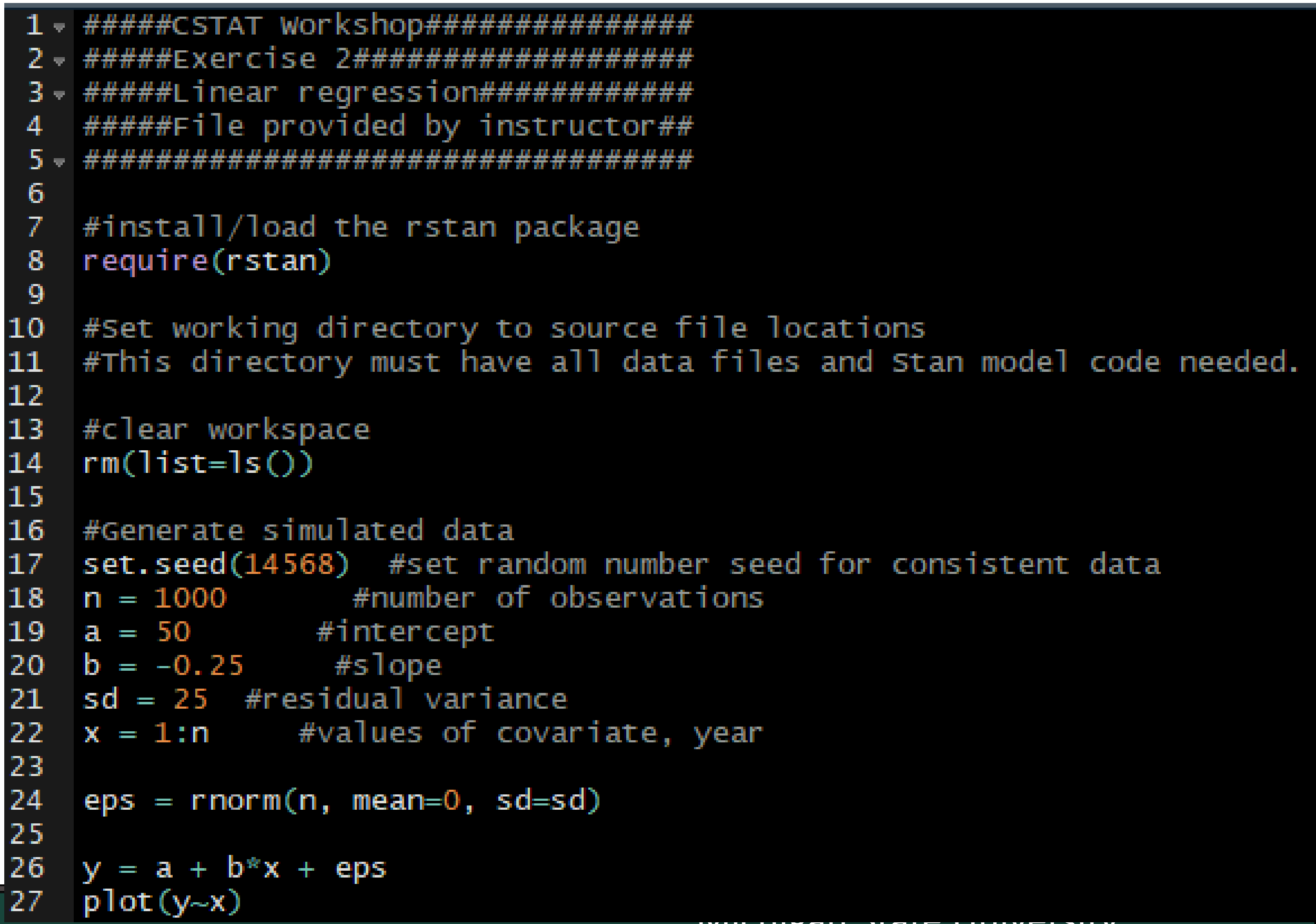






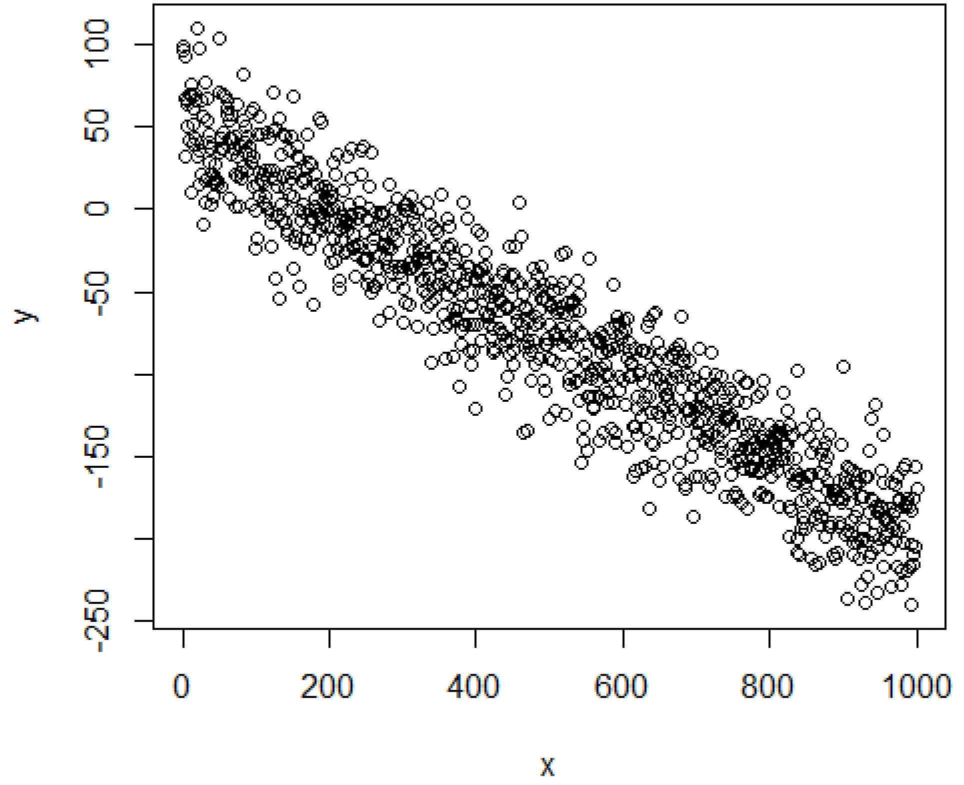
(B) Exercise 2: Linear Regression

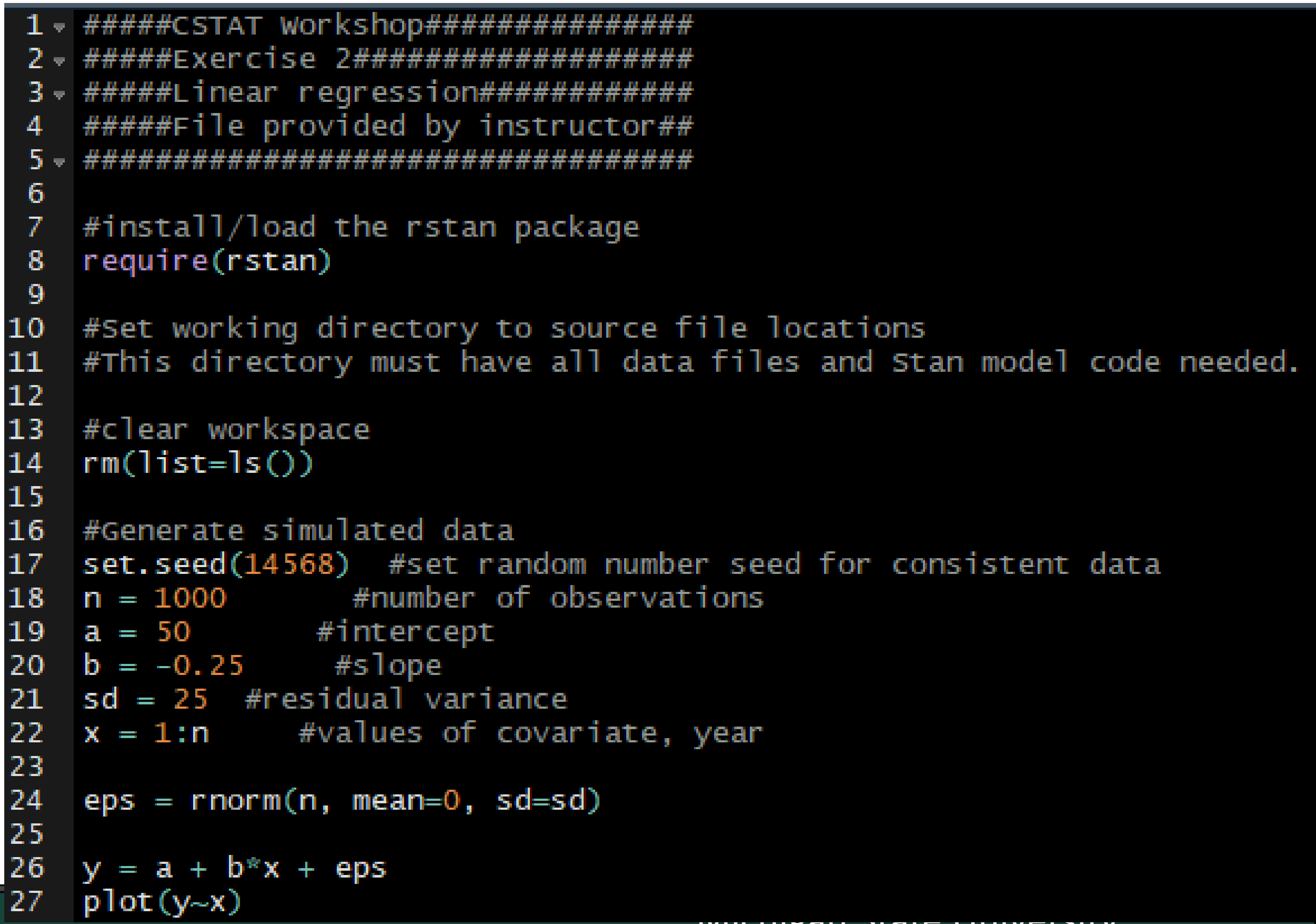
1. Open "Ex2_LReg.R"
2. Load library
3. Set working directory
4. Clear workspace
5. Generate or load data



```
#####CSTAT Workshop##############
#####Exercise 2##################
#####Linear regression###########
#####File provided by instructor##
#################################

#install/load the rstan package
require(rstan)

#Set working directory to source file locations
#This directory must have all data files and Stan model code needed.

#clear workspace
rm(list=ls())

#Generate simulated data
set.seed(14568)  #set random number seed for consistent data
n = 1000       #number of observations
a = 50       #intercept
b = -0.25     #slope
sd = 25  #residual variance
x = 1:n     #values of covariate, year

eps = rnorm(n, mean=0, sd=sd)

y = a + b*x + eps
plot(y~x)
```

(B) Exercise 2: Linear Regression





```
#####CSTAT Workshop##############
#####Exercise 2##################
#####Linear regression###########
#####File provided by instructor##
##################################

#install/load the rstan package
require(rstan)

#Set working directory to source file locations
#This directory must have all data files and Stan model code needed.

#clear workspace
rm(list=ls())

#Generate simulated data
set.seed(14568)  #set random number seed for consistent data
n = 1000       #number of observations
a = 50       #intercept
b = -0.25     #slope
sd = 25  #residual variance
x = 1:n     #values of covariate, year

eps = rnorm(n, mean=0, sd=sd)

y = a + b*x + eps
plot(y~x)
```

(A) Exercise 2: Linear Regression

Model: $$\mathrm{Y}_i = \alpha + \beta \mathrm{X}_i + \varepsilon_i$$

$$\varepsilon_i \sim \mathrm{Normal}(0, \sigma)$$

Priors: $$\alpha \sim \mathrm{Normal}(0{,}100)$$

$$\beta \sim \mathrm{Normal}(0{,}100)$$

$$\sigma \sim \mathrm{half} - \mathrm{cauchy}(0{,}5)$$

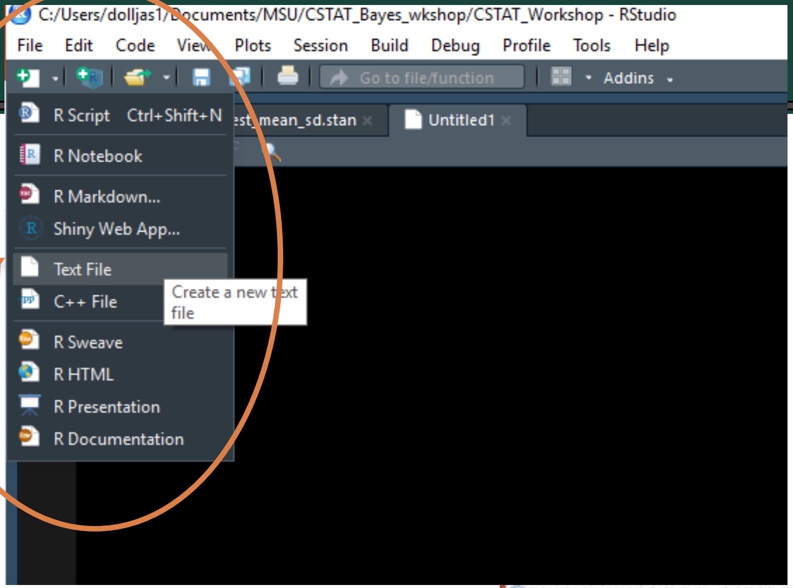
- Model and priors

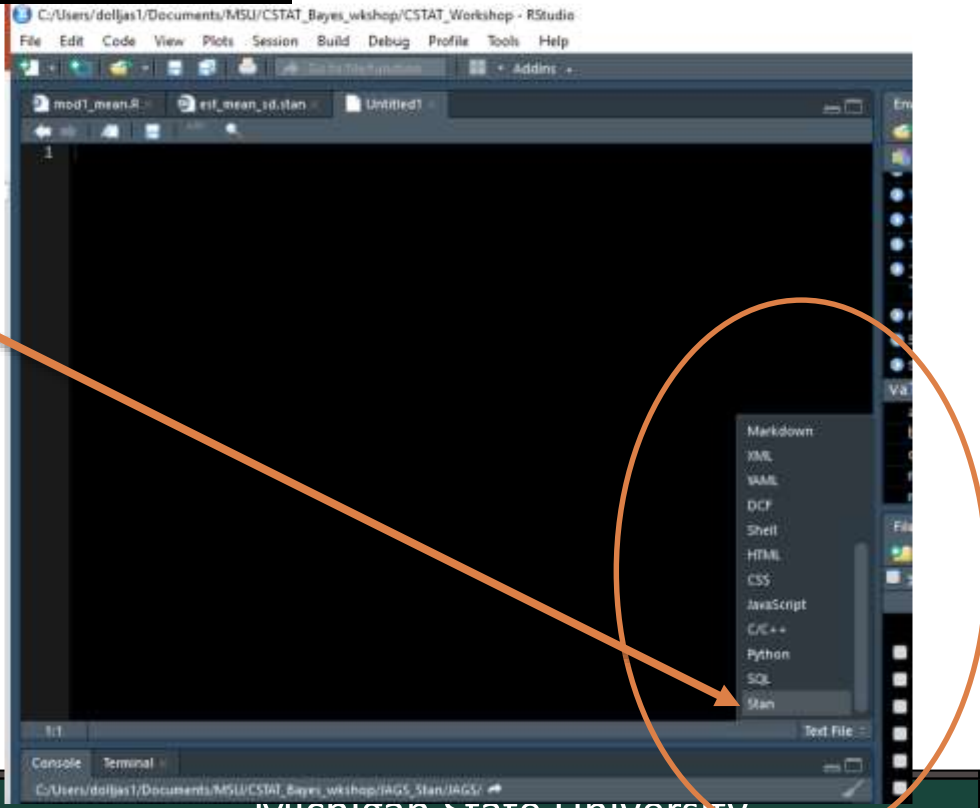
(C) Exercise 2: Linear Regression

- Create .stan file
- Try on your own following the handout

(C) Exercise 2

- Create new text file
- Change “Text File” to “Stan”




C:/Users/dolljas1/Documents/MSU/CSTAT_Bayes_wkshop/CSTAT_Workshop - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Addins
mod1_mean.R
est_mean_sd.stan
Untitled1
Markdown
XML
YAML
DCF
Shell
HTML
CSS
JavaScript
C/C++
Python
SQL
Stan
Text File
Console
Terminal
C:/Users/dolljas1/Documents/MSU/CSTAT_Bayes_wkshop/JAGS_Stan/JAGS/

(C) Exercise 2: Linear Regression

```
data{
  int<lower=0> n;  //number of observations
  vector[n] x;     //observed x values, predictors
  vector[n] y;     //observed y values, response
}
parameters{
 real<lower=0> sigma; //standard deviation
  real alpha;          //y-intercept
  real beta;           //slope
}
model{
 //reference priors
  alpha ~ normal(0,100);
  beta ~ normal(0,100);
  sigma ~ cauchy(0,5);

  //likelihood, loop through number of observations
  for(i in 1:n){
    y[i] ~ normal(alpha + beta * x[i], sigma);
  }
}
```

(D) Exercise 2: data block

```
data{
  int<lower=0> n;  //number of observations
  vector[n] x;     //observed x values, predictors
  vector[n] y;     //observed y values, response
}
```

(D) Exercise 2: parameter block

```
parameters{
  real<lower=0> sigma; //standard deviation
  real alpha;          //y-intercept
  real beta;           //slope
}
```

(D) Exercise 2: model block

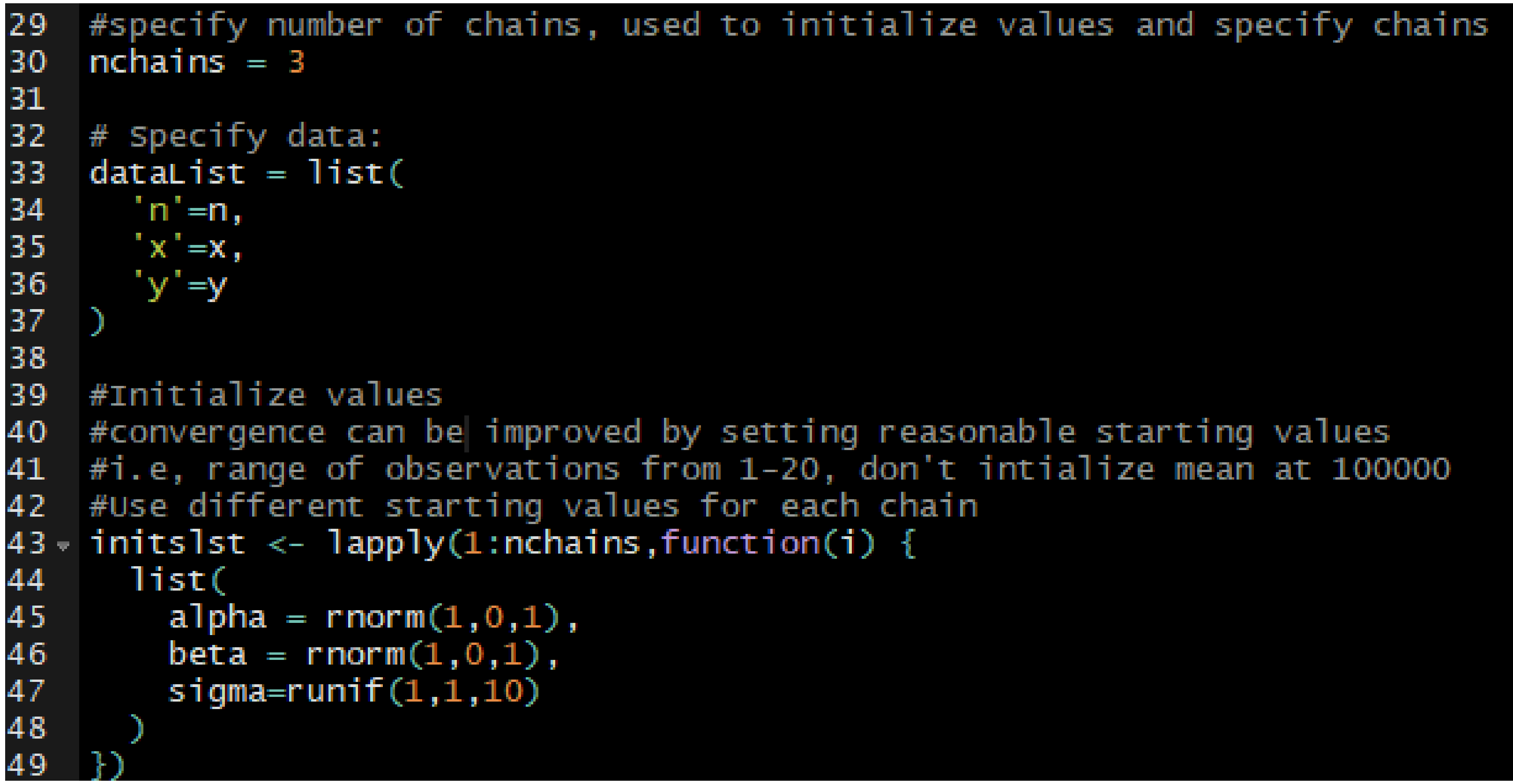
```
model{
  //reference priors
  alpha ~ normal(0,100);
  beta ~ normal(0,100);
  sigma ~ cauchy(0,5);

  //likelihood, loop through number of observations
  for(i in 1:n){
    y[i] ~ normal(alpha + beta * x[i], sigma);
  }
}
```

(D) Run model

- Save Stan code file as “Ex2_est_lr.stan”
- Return to “Ex2_LReg.R”

(D) Exercise 2: Linear Regression

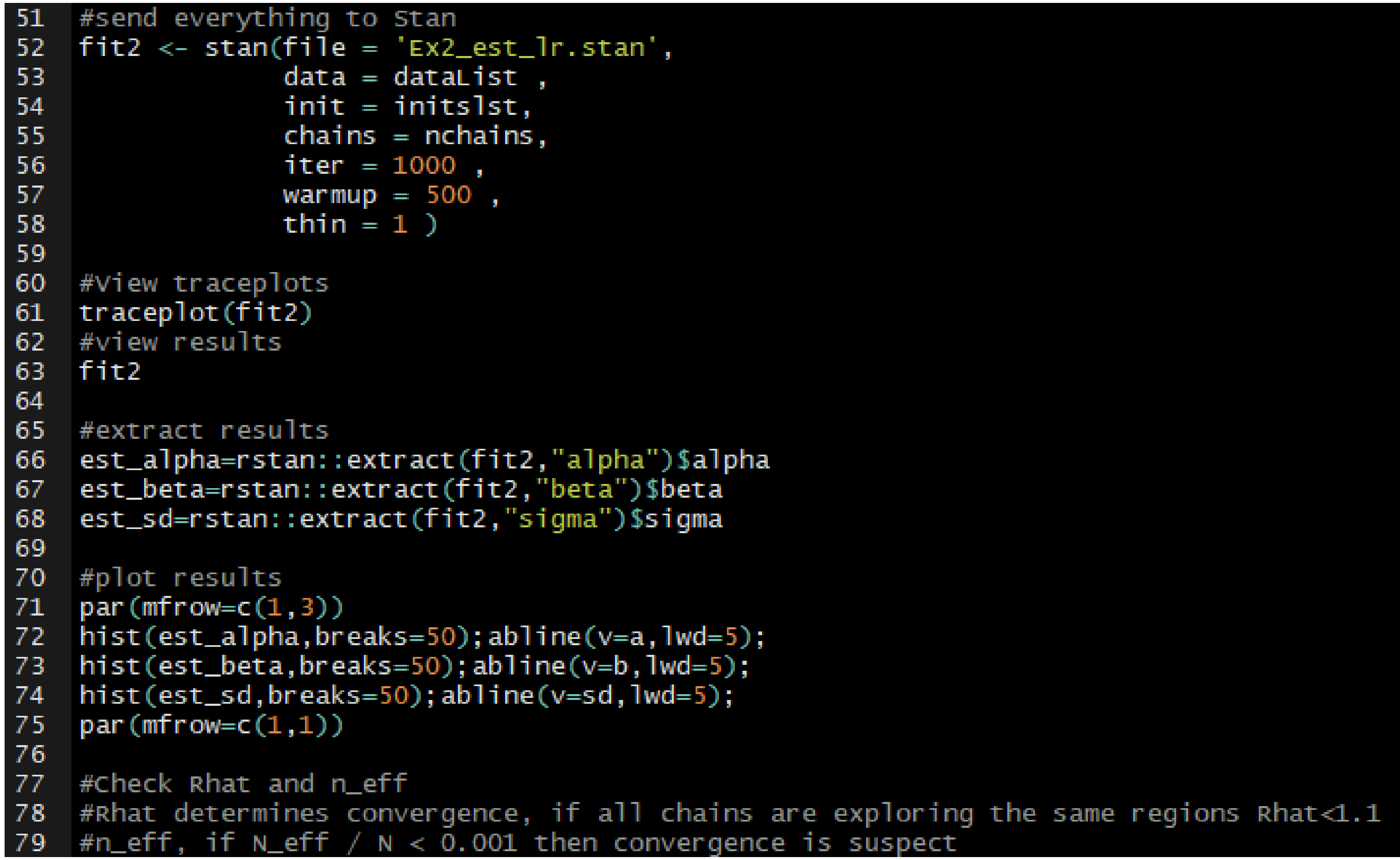
6. Specify chains
7. Create data list
8. Create initialization list



```
#specify number of chains, used to initialize values and specify chains
nchains = 3

# Specify data:
dataList = list(
  'n'=n,
  'x'=x,
  'y'=y
)

#Initialize values
#convergence can be improved by setting reasonable starting values
#i.e, range of observations from 1-20, don't intialize mean at 100000
#Use different starting values for each chain
initslst <- lapply(1:nchains,function(i) {
  list(
    alpha = rnorm(1,0,1),
    beta = rnorm(1,0,1),
    sigma=runif(1,1,10)
  )
})
```

(D) Exercise 2: Linear Regression

9. Send to Stan
10. Misc. functions to explore Stan Output



```
#send everything to Stan
fit2 <- stan(file = 'Ex2_est_lr.stan',
             data = dataList ,
             init = initslst,
             chains = nchains,
             iter = 1000 ,
             warmup = 500 ,
             thin = 1 )

#View traceplots
traceplot(fit2)
#view results
fit2

#extract results
est_alpha=rstan::extract(fit2,"alpha")$alpha
est_beta=rstan::extract(fit2,"beta")$beta
est_sd=rstan::extract(fit2,"sigma")$sigma

#plot results
par(mfrow=c(1,3))
hist(est_alpha,breaks=50);abline(v=a,lwd=5);
hist(est_beta,breaks=50);abline(v=b,lwd=5);
hist(est_sd,breaks=50);abline(v=sd,lwd=5);
par(mfrow=c(1,1))

#Check Rhat and n_eff
#Rhat determines convergence, if all chains are exploring the same regions Rhat<1.1
#n_eff, if N_eff / N < 0.001 then convergence is suspect
```

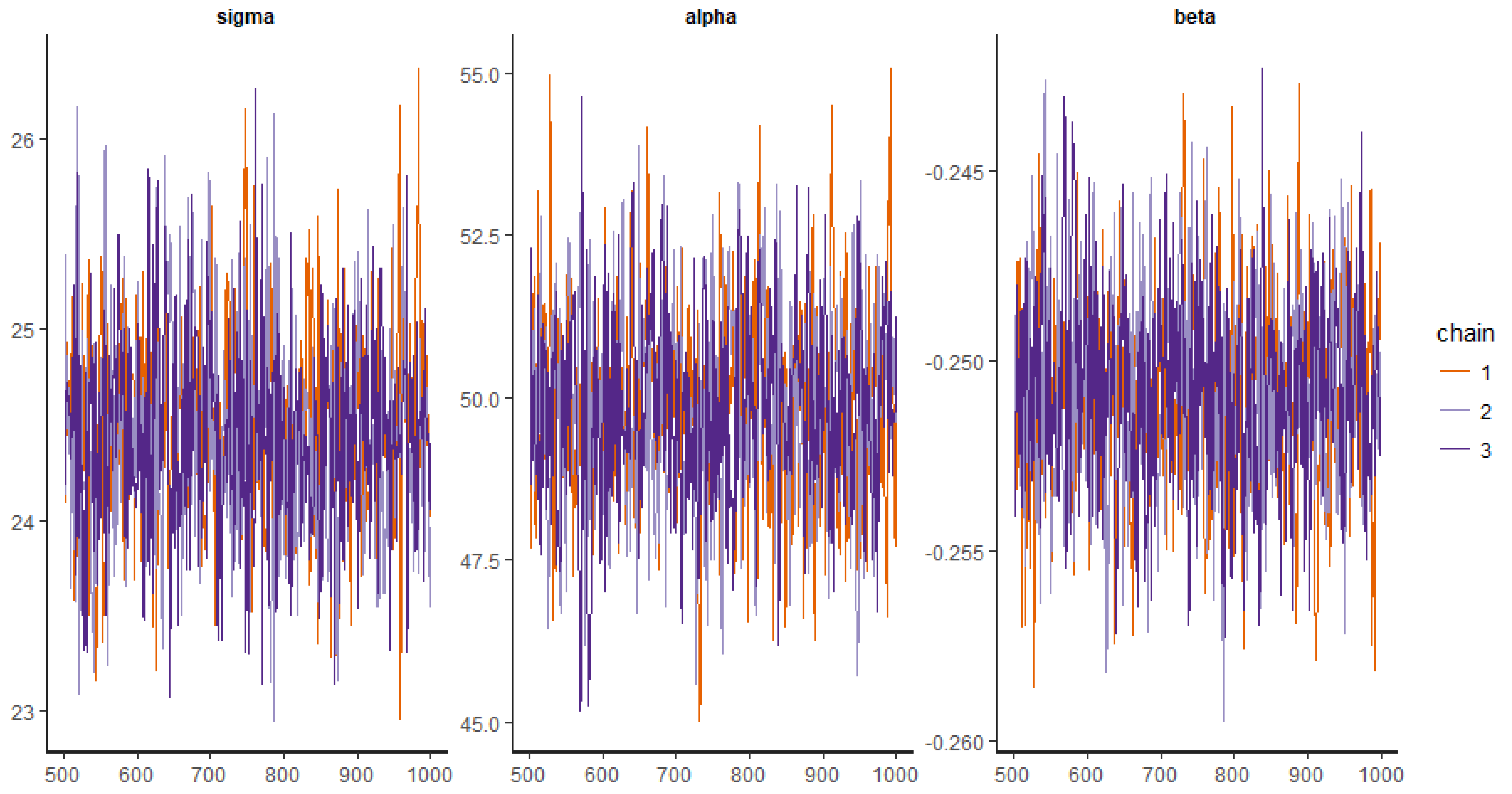
(E) Run model

- Save Stan code file as “Ex2_est_lr.stan”
- Return to “mod1_mean.R”
- Run lines 1-58
- Stan compiles program (takes ~30-60 second)
- Compare results to known values

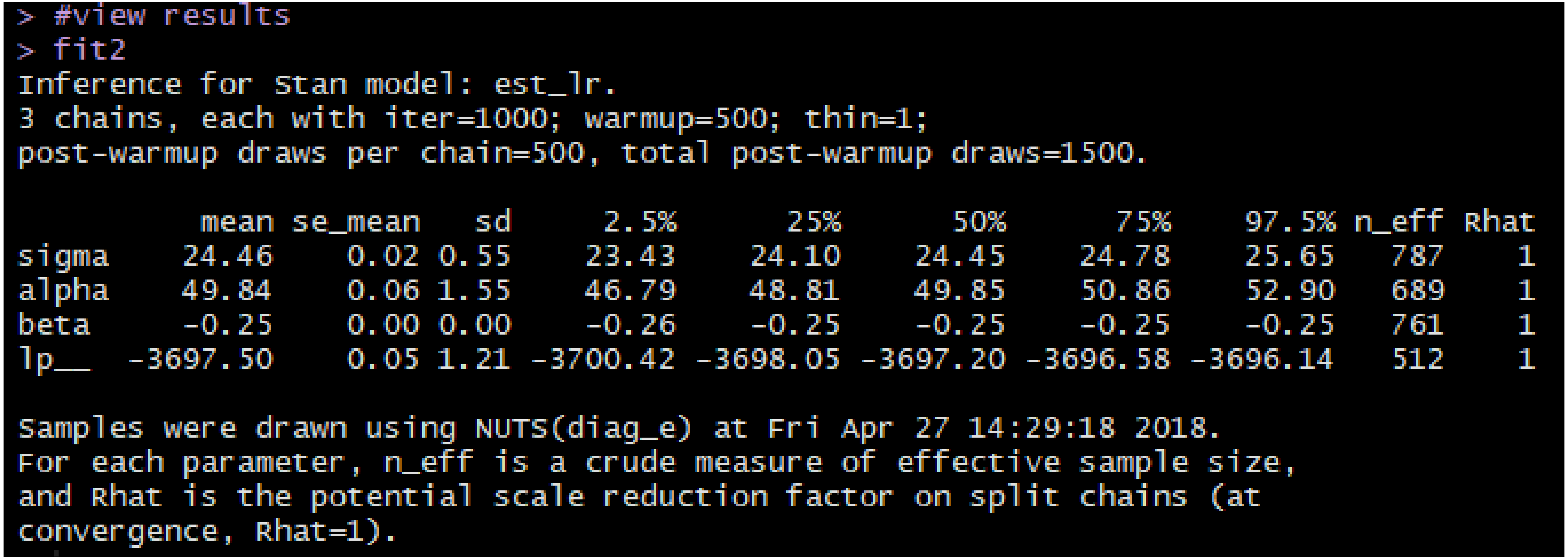
(E) Exercise 2: Linear Regression

- Review output
- Check Convergence
  - Trace plots
  - Effective Sample Size
  - R-hat

(E) `traceplot(fit1)`



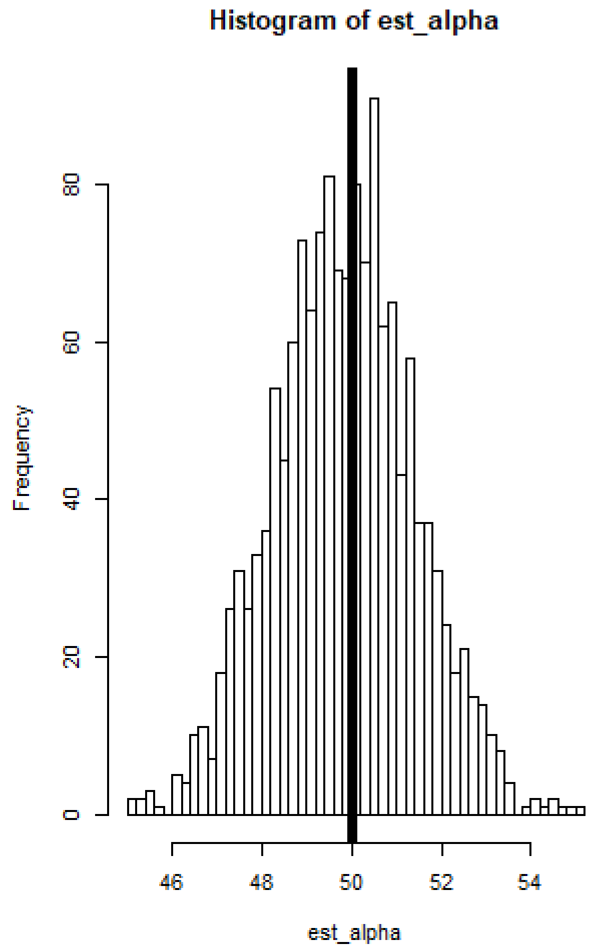
(E) fit1

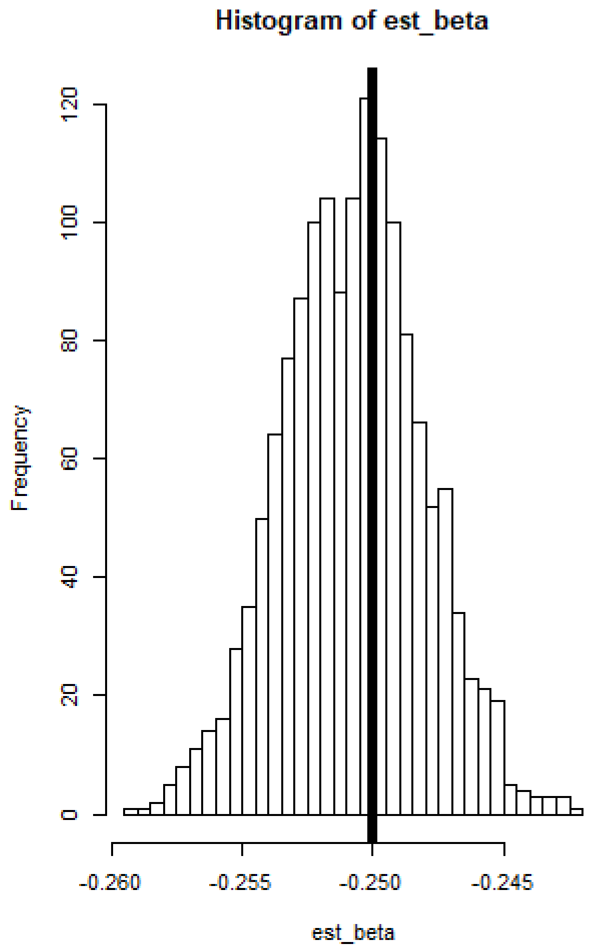


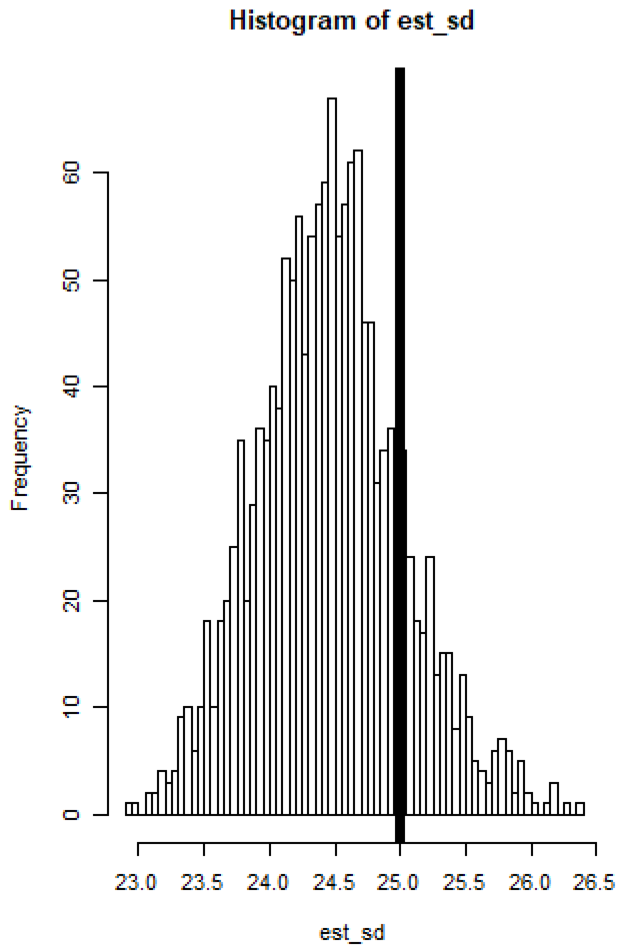
```
> #view results
> fit2
Inference for Stan model: est_lr.
3 chains, each with iter=1000; warmup=500; thin=1;
post-warmup draws per chain=500, total post-warmup draws=1500.

         mean se_mean   sd     2.5%      25%      50%      75%    97.5% n_eff Rhat
sigma   24.46    0.02 0.55    23.43    24.10    24.45    24.78    25.65   787    1
alpha   49.84    0.06 1.55    46.79    48.81    49.85    50.86    52.90   689    1
beta    -0.25    0.00 0.00    -0.26    -0.25    -0.25    -0.25    -0.25   761    1
lp__ -3697.50    0.05 1.21 -3700.42 -3698.05 -3697.20 -3696.58 -3696.14   512    1

Samples were drawn using NUTS(diag_e) at Fri Apr 27 14:29:18 2018.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

if N_eff / N < 0.001 then convergence is suspect

(E) Compare to known







Questions?

Stan tips

- NEVER use default uniform prior
- Weakly informative prior when possible
  - e.g., normal(0,1000)
  - Do you really think values > 4,000 are reasonable?
  - Maybe normal(0,10)?
- Complex model
  - Try super-constrained priors first
  - e.g., normal(4,0.1)

Common Stan warnings

- Ignore compiler warnings (see examples)

Common Stan warnings

- Ignore compiler warnings (see examples)
- Maximum treedepth exceeded

Max treedepth

- No-U-Turn sampler automatically tunes step size and number of steps
- Step size determined by a tree building algorithm
- When max treedepth is reached, the algorithm stops
- If max treedepth too low, more iterations required to explore posterior
- If max treedepth too high, fewer iterations but increases time between iterations (much less efficient)

Max treedepth: What to do?

- Exceeding max treedepth is an **efficiency concern**
- Increase max treedepth
- Reparametrize your model
- Taking too long? Send to HPCC

Some Stan warnings

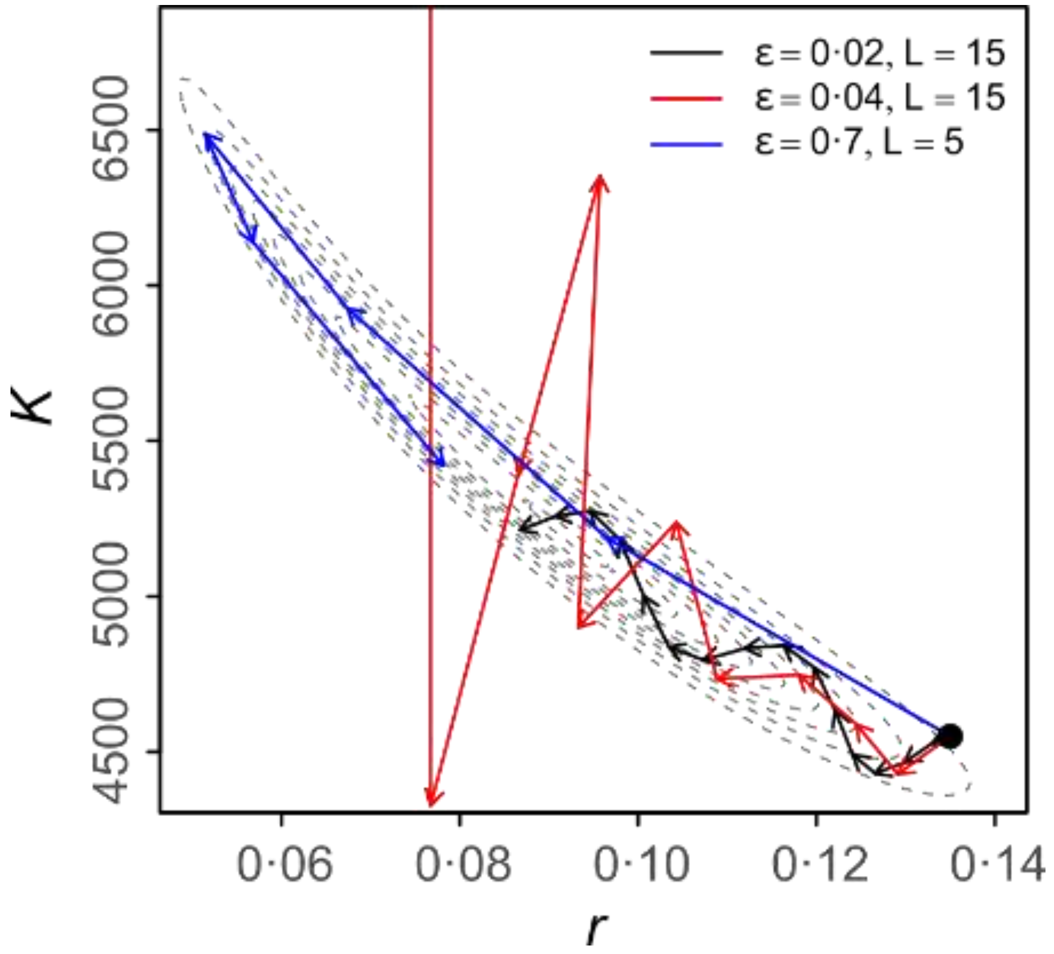
- Ignore compiler warnings (see example)
- Maximum treedepth exceeded
- Bayesian fraction of missing information was too low
  - BFMI low
  - Adaptation phase too short and chains did not explore posterior
  - Set higher `iter` or `warmup`
  - Note: iterations >10,000 not common

Some Stan warnings

- Ignore compiler warnings (see example)
- Maximum treedepth exceeded
- Bayesian fraction of missing information was too low
  - BFMI low
  - Adaptation phase too short and chains did not explore posterior
  - Set higher `iter` or `warmup`
  - Note: iterations >10,000 not common
- There were *15* divergent transitions after warmup.

Divergent transitions

- The HMC sampler trajectory determined by the step size (ε) and number of steps (L).
- The same length can be attained two ways:
  - Fewer steps of larger size
  - More steps of smaller size
- Large step size relative to L
- Divergent transitions are a **validity concern**



Monnahan, C.C., J.T. Thorson, and T.A. Branch. 2017. Faster estimation of Bayesian models in ecology using Hamiltonian Monte Carlo. Methods in Ecology and Evolution 8:339-348.

Divergent transitions: What to do?

- Increase adapt_delta
  - Target average proposal acceptance probability
  - Default = 0.80
  - Try 0.85 then increase by 0.05

```
stan(…, control = list(adapt_delta = 0.85)
```

- Doesn’t always work with hierarchical models

Divergent transitions: What to do?

- Centered hierarchical priors
  - Model random effect directly

$$\tau \sim Normal(\mu, \sigma)$$

- Non-center hierarchical priors
  - Model random effect indirectly

$$\tau = \mu + \sigma * Z$$

$$Z \sim Normal(0,1)$$

Lunch?

Workshop Outline

Morning

- Bayesian basics
- Stan background, syntax, and coding best practices
- Simple examples
  - Normal distribution
  - Linear regression
- Common errors and techniques to address them

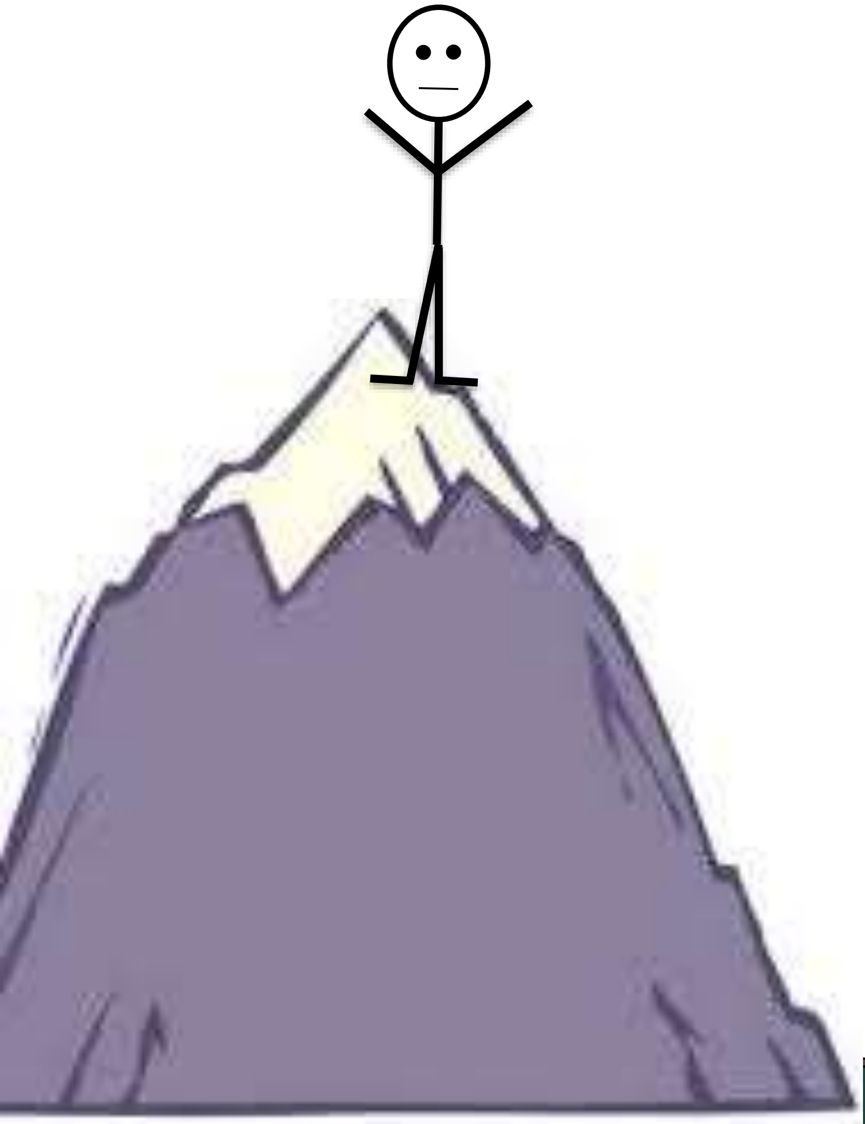
Afternoon

- Addressing common errors, cont.
- ShinyStan
- Mixed effects
- Model Selection

Divergent transitions

- HMC explores the target distribution by taking a step across parameter space of specified size
- Sometimes the step size needed is too small to explore the parameter space
- Thus, sampler misses features and returns biased estimates
- Luckily this mismatch of scales is flagged by the sampler

Step Size



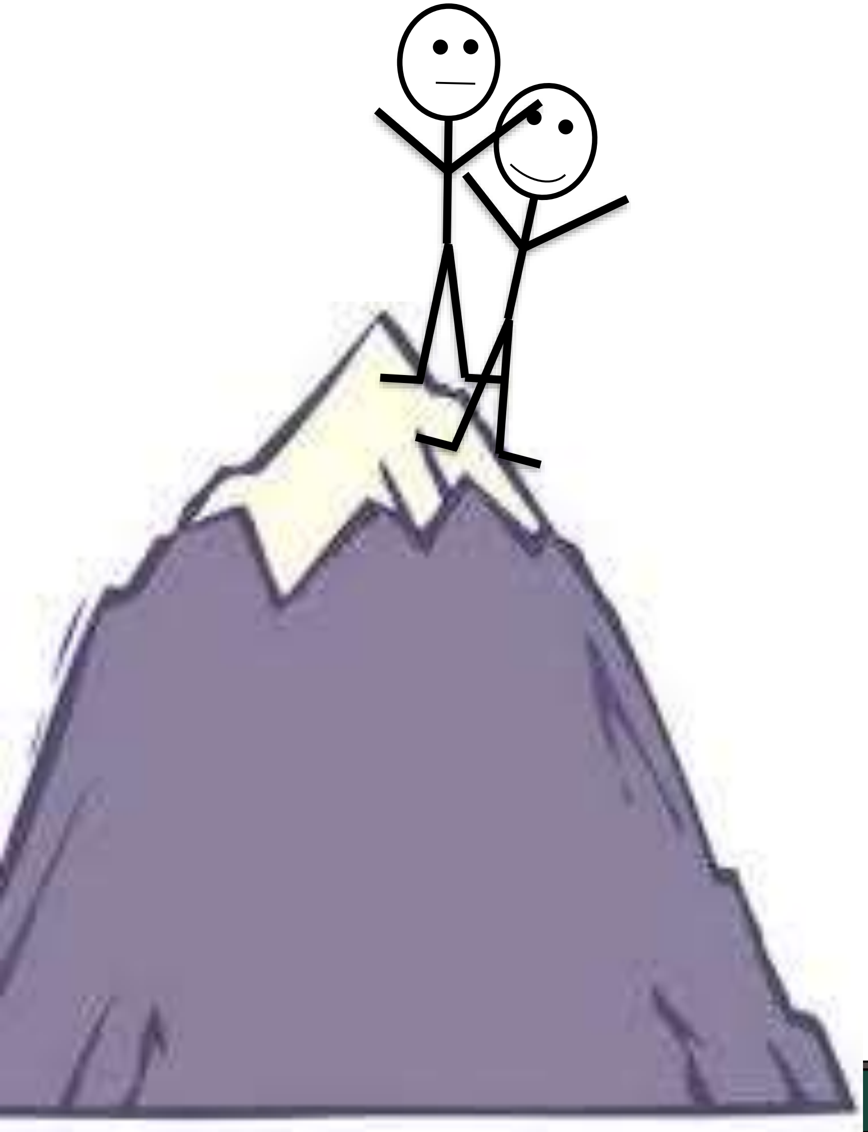
Step Size

Step size too LARGE

Quick but

Fall and miss mountain

Step Size



Step size small
Slow but get down
mountain safely

Exercise 3: Divergent Transitions

A. Describe data and model
B. Review R code (provided)
C. Review Stan code
D. You program the *.stan file following the screen or handout
E. Enjoy your results!

(A) Exercise 3: Divergent Transitions

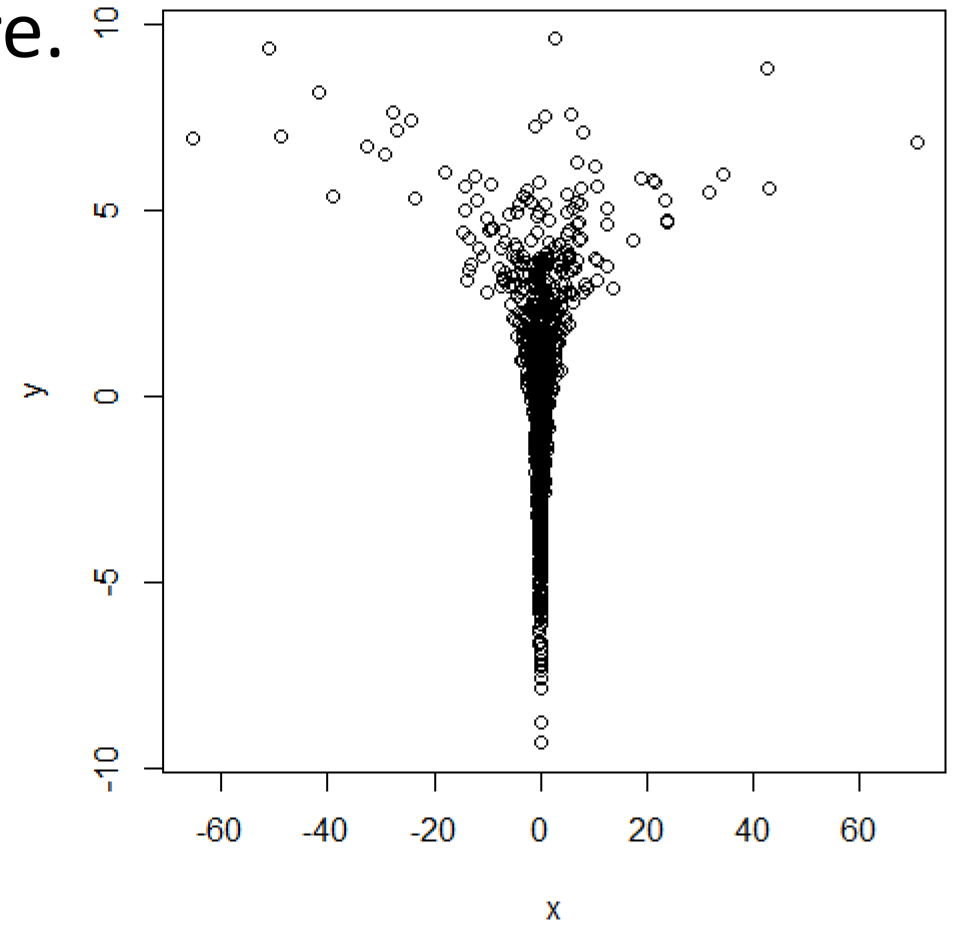
- We will use Stan to generate data based on fixed parameters of a normal distribution to generate divergent transitions
- We will create two Stan programs
  - 1. Centered parameterization (results in Divergent Transitions)
  - 2. Non-centered parametrization
- Inspect results and compare output

(A) Exercise 3: Divergent Transitions

- Two parameters with varying curvature.
- When Y is small, X range is small
- When Y is large, X range is large

$$y = \text{normal}(0, 3)$$

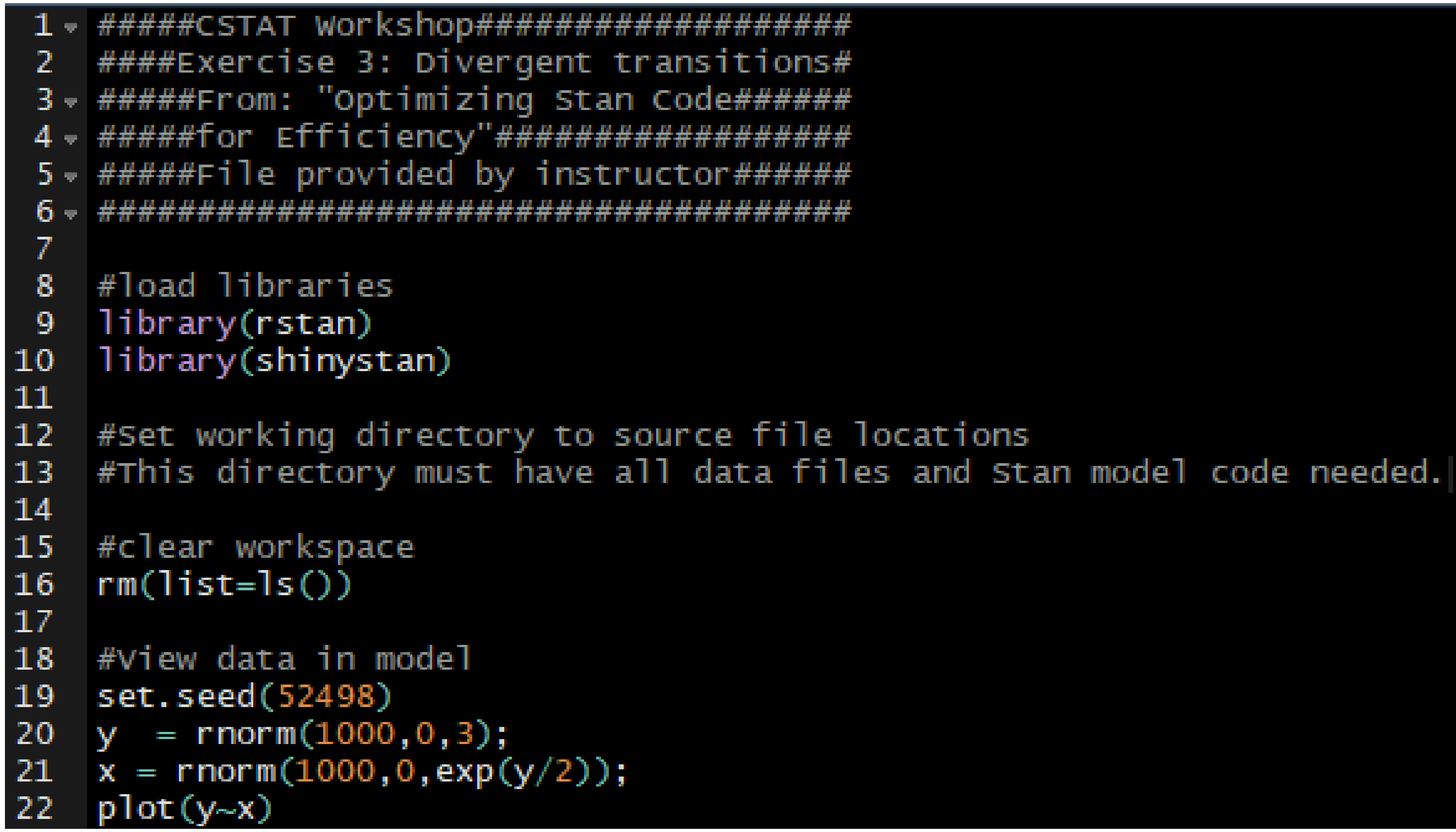
$$x = \text{normal}\left(0, \exp(y/2)\right)$$



Example from Stan manual

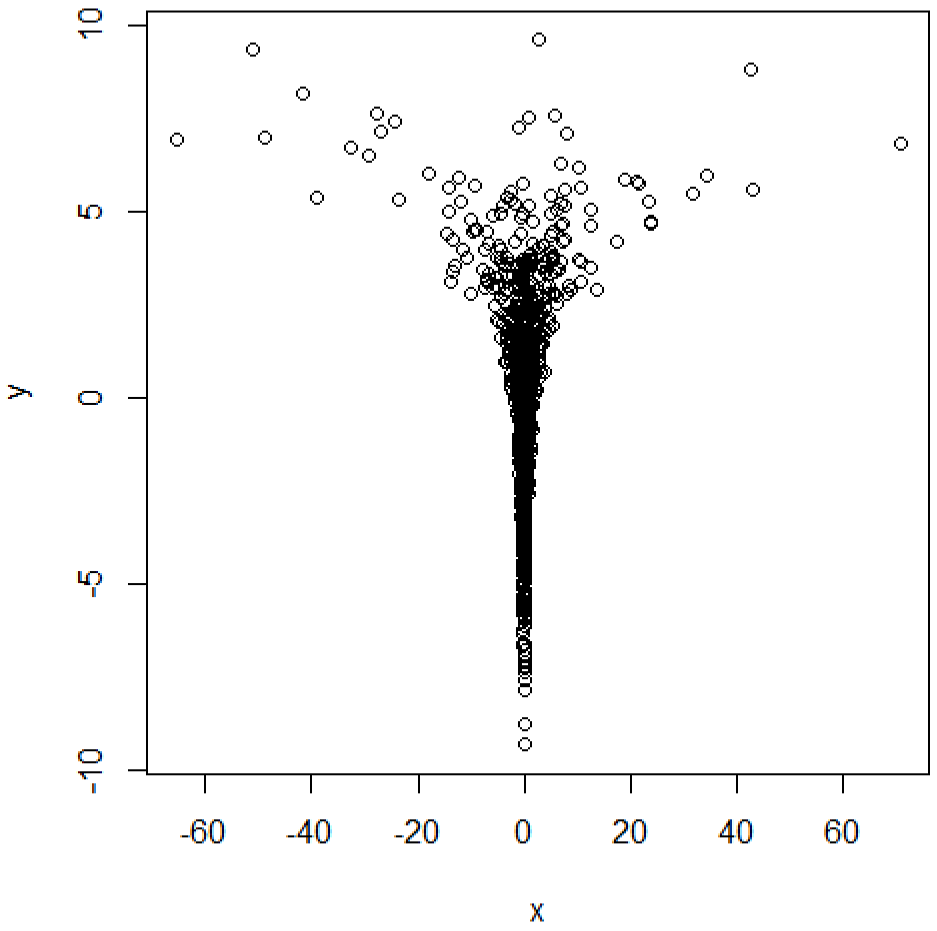
(B) Exercise 3: Divergent Transitions

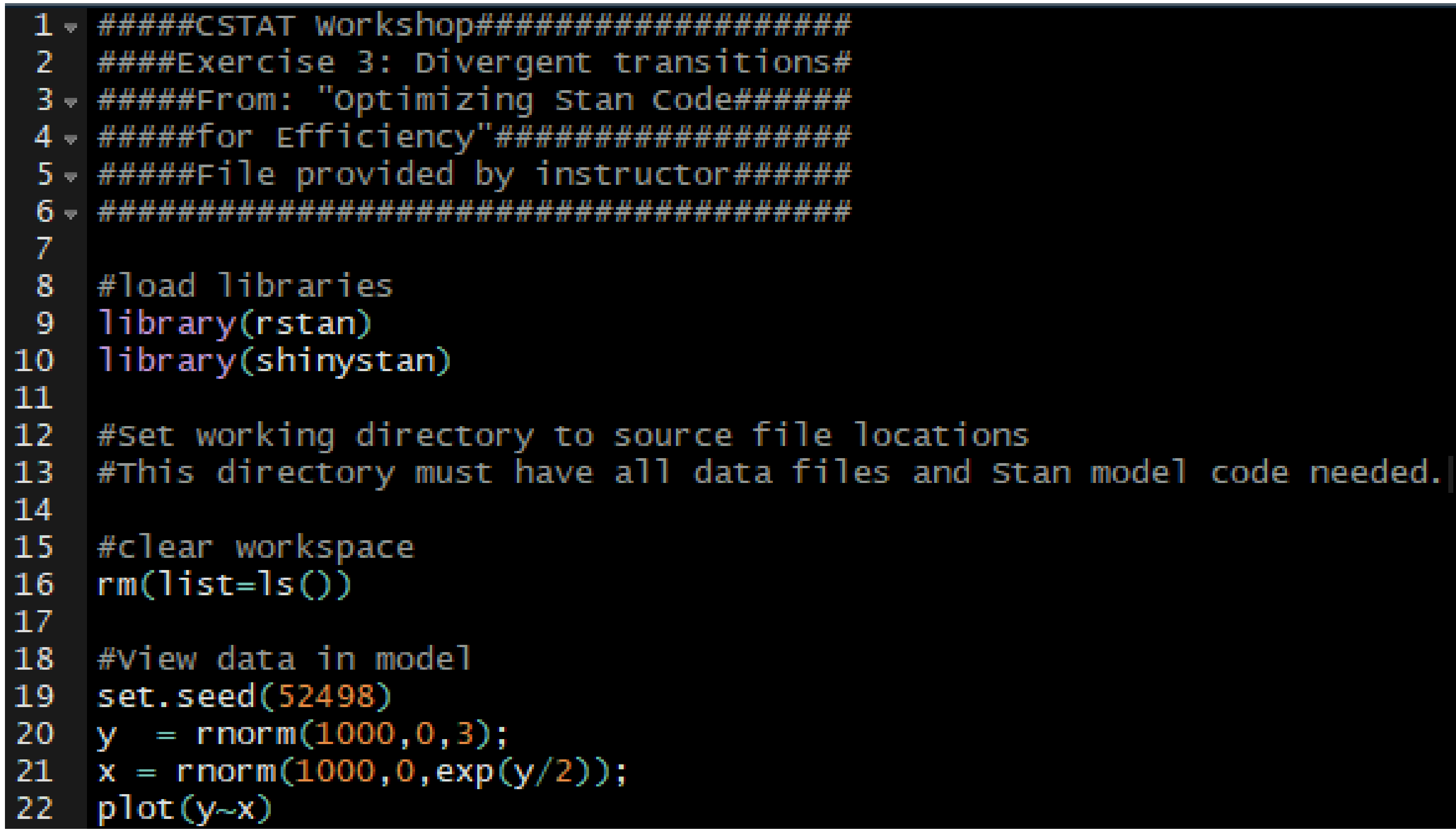
1. Open “Ex3_Div.R”
2. Load library
3. Set working directory
4. Clear workspace
5. View data



```
#####CSTAT Workshop###################
####Exercise 3: Divergent transitions#
#####From: "Optimizing Stan Code######
#####for Efficiency"##################
#####File provided by instructor######
######################################

#load libraries
library(rstan)
library(shinystan)

#Set working directory to source file locations
#This directory must have all data files and Stan model code needed.

#clear workspace
rm(list=ls())

#View data in model
set.seed(52498)
y  = rnorm(1000,0,3);
x = rnorm(1000,0,exp(y/2));
plot(y~x)
```

(B) Exercise 3: Divergent Transitions





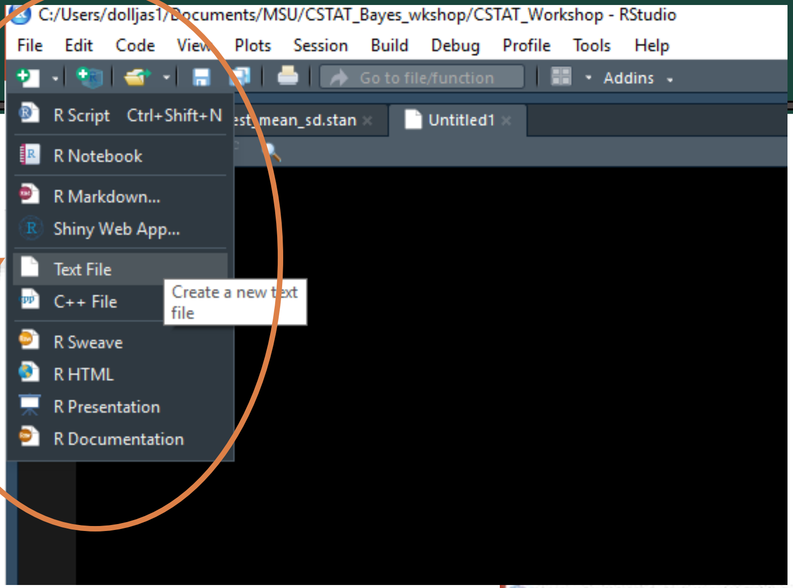
```
#####CSTAT Workshop###################
####Exercise 3: Divergent transitions#
#####From: "Optimizing Stan Code######
#####for Efficiency"##################
#####File provided by instructor######
######################################

#load libraries
library(rstan)
library(shinystan)

#Set working directory to source file locations
#This directory must have all data files and Stan model code needed.

#clear workspace
rm(list=ls())

#View data in model
set.seed(52498)
y  = rnorm(1000,0,3);
x = rnorm(1000,0,exp(y/2));
plot(y~x)
```

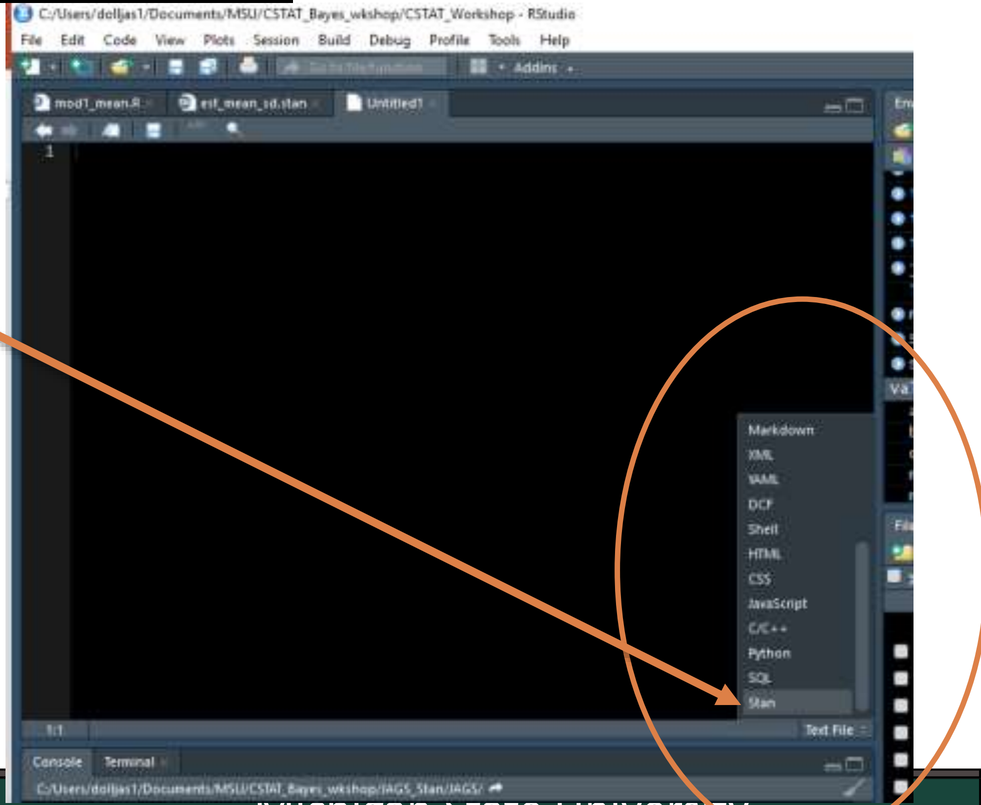
(C) Exercise 3: Centered

- Create *.stan file for centered parameterization

(C) Exercise 3

- Create new text file
- Change "Text File" to "Stan"




C:/Users/dolljas1/Documents/MSU/CSTAT_Bayes_wkshop/CSTAT_Workshop - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Addins
mod1_mean.R
est_mean_sd.stan
Untitled1
Markdown
XML
YAML
DCF
Shell
HTML
CSS
JavaScript
C/C++
Python
SQL
Stan
Text File
Console
Terminal

(C) Exercise 3: Centered

```
parameters{
  real y;
  vector[9] x;
}
```

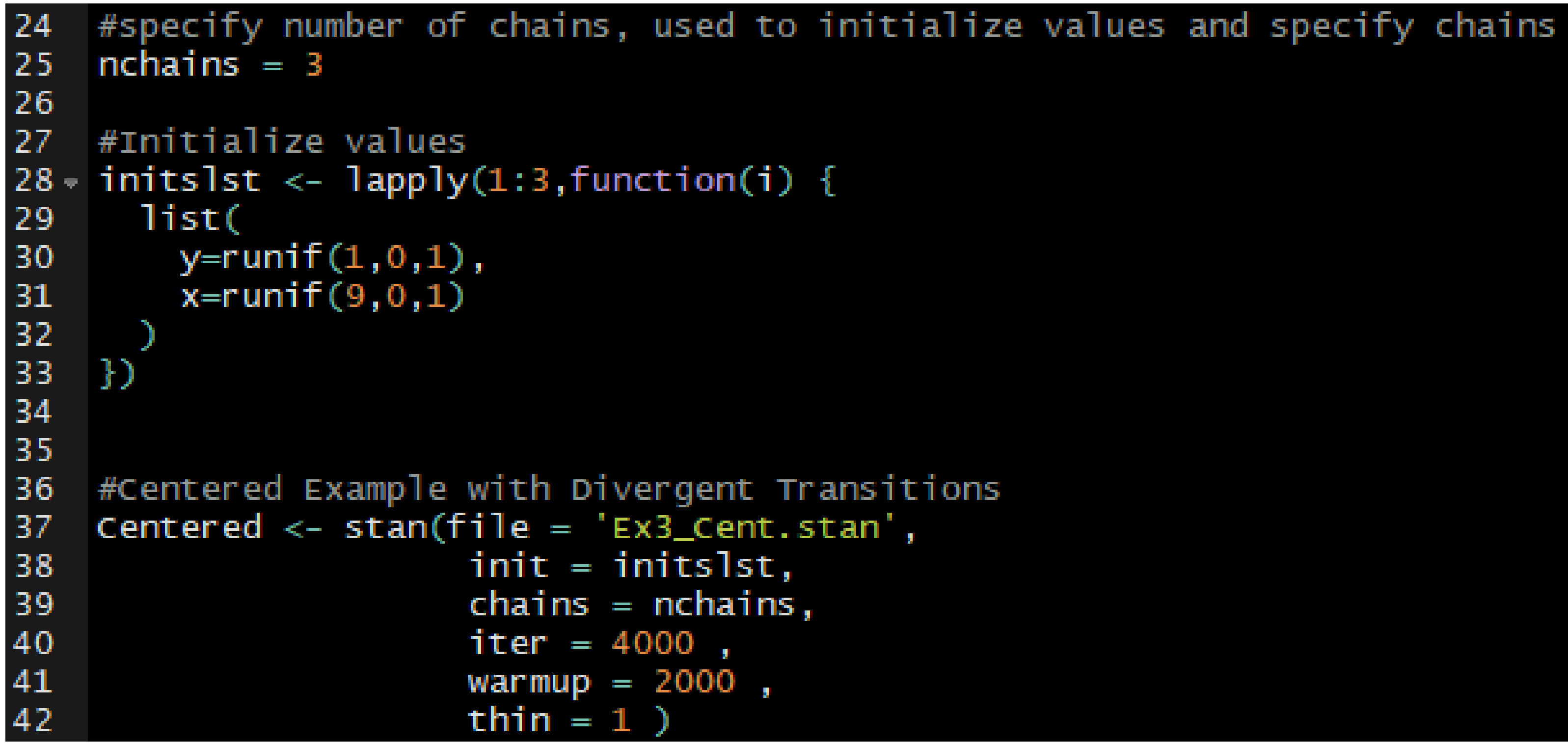
```
model {
  y~normal(0,3);
  x~normal(0,exp(y/2));
}
```

(D) Exercise 3: Centered

- Save Stan code file as “Ex3_Cent.stan”
- Return to “Ex3_Div.R”

(D) Exercise 3: Centered

6. Specify chains
7. Create initialization list
8. Send to Stan

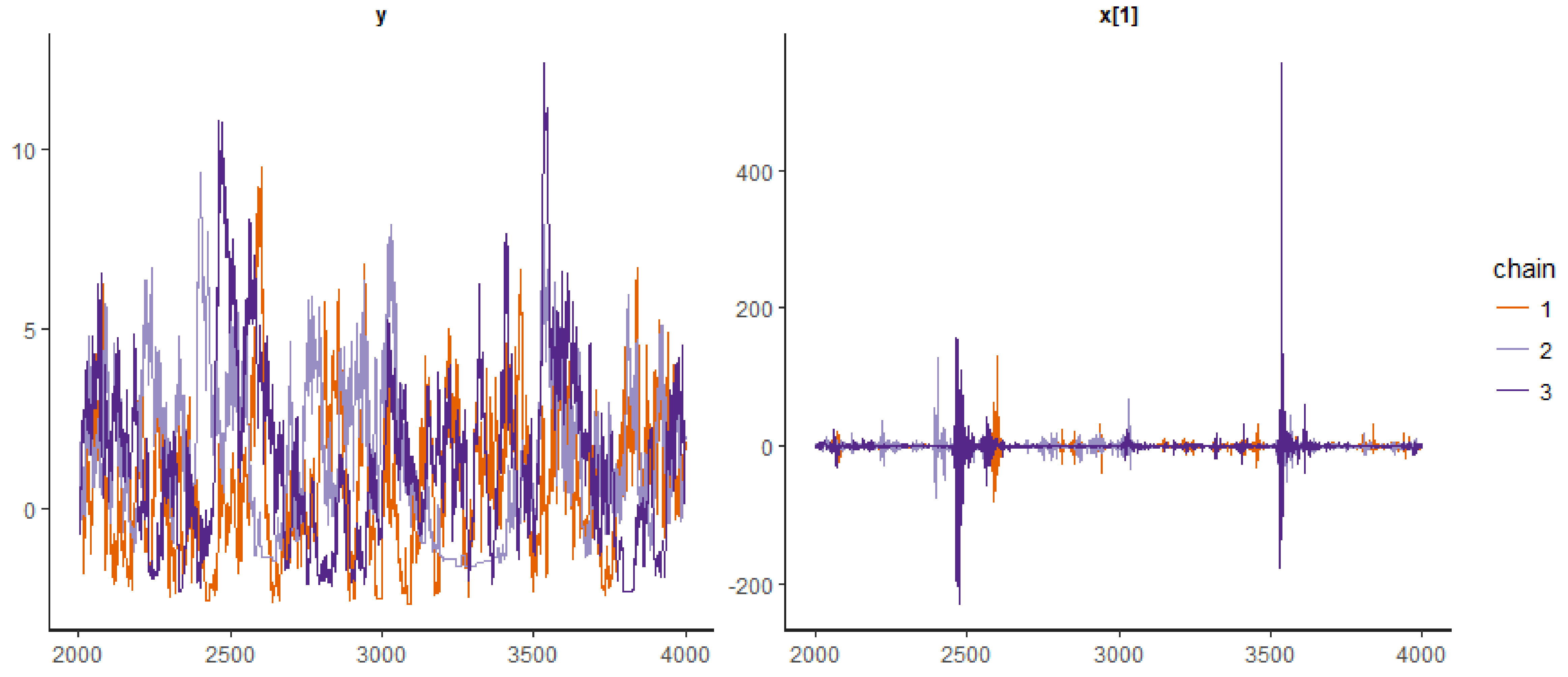


```r
#specify number of chains, used to initialize values and specify chains
nchains = 3

#Initialize values
initslst <- lapply(1:3,function(i) {
  list(
    y=runif(1,0,1),
    x=runif(9,0,1)
  )
})


#Centered Example with Divergent Transitions
Centered <- stan(file = 'Ex3_Cent.stan',
                 init = initslst,
                 chains = nchains,
                 iter = 4000 ,
                 warmup = 2000 ,
                 thin = 1 )
```

(E) Exercise 3: Centered

- Save Stan code file as “Ex3_Cent.stan”
- Return to “Ex3_Div.R”
- Run lines 1-42
- Stan compiles program (takes ~30-60 seconds)
- Review Results

(E) `traceplot(Centered,pars=c("y","x[1]"))`

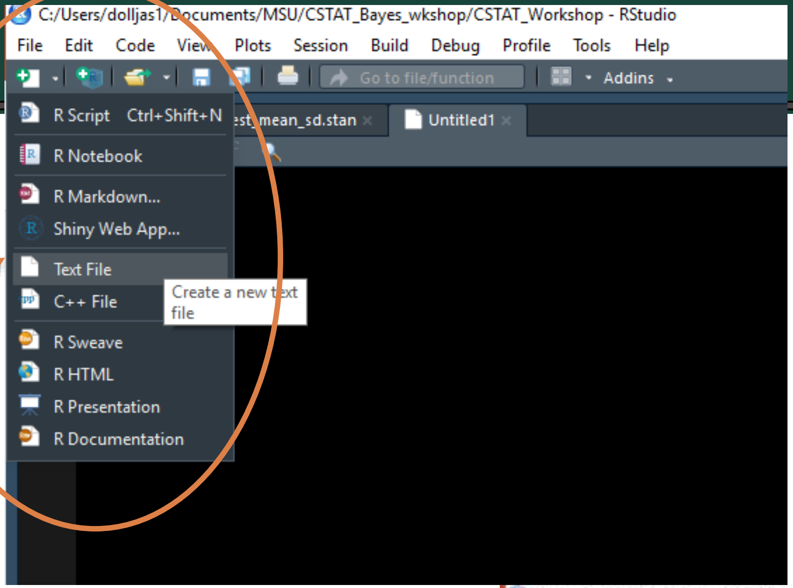


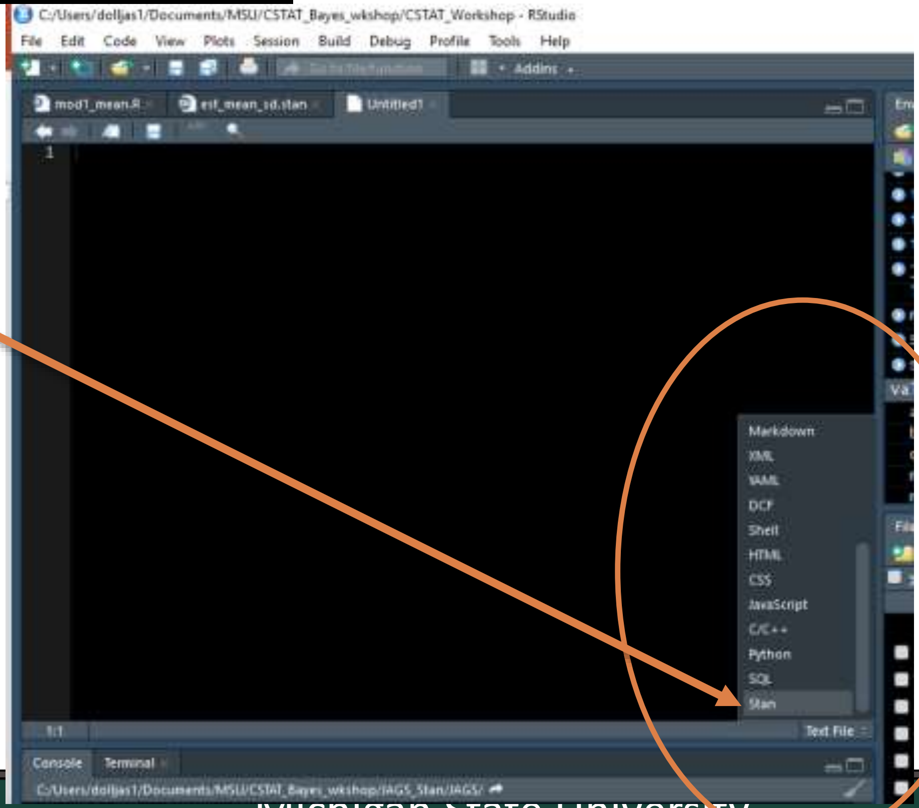
(C) Exercise 3: Non-Centered

- Create *.stan file for non-centered parameterization
- Try on your own following the handout
- Show next two slides to get started

(C) Exercise 3

- Create new text file
- Change “Text File” to “Stan”




C:/Users/dolljas1/Documents/MSU/CSTAT_Bayes_wkshop/CSTAT_Workshop - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Addins
mod1_mean.R
est_mean_sd.stan
Untitled1
Markdown
XML
YAML
DCF
Shell
HTML
CSS
JavaScript
C/C++
Python
SQL
Stan
Text File
Console
Terminal
C:/Users/dolljas1/Documents/MSU/CSTAT_Bayes_wkshop/JAGS_Stan/JAGS/

(C) Exercise 3: Non-Centered

1. Suppose original parameter is **beta**
2. Create new parameter and name it **beta_raw**
3. Give **beta_raw** a normal prior, mean = 0 and SD = 1
4. Move original beta parameter declaration to "*transformed parameter block*"
5. Equate original parameter in "*transformed parameter*" block to **beta = mean + sd * beta_raw**

Note: Above ONLY when original parameter is normally distributed. See Stan manual for other distributions

(C) Exercise 3: Non-Centered

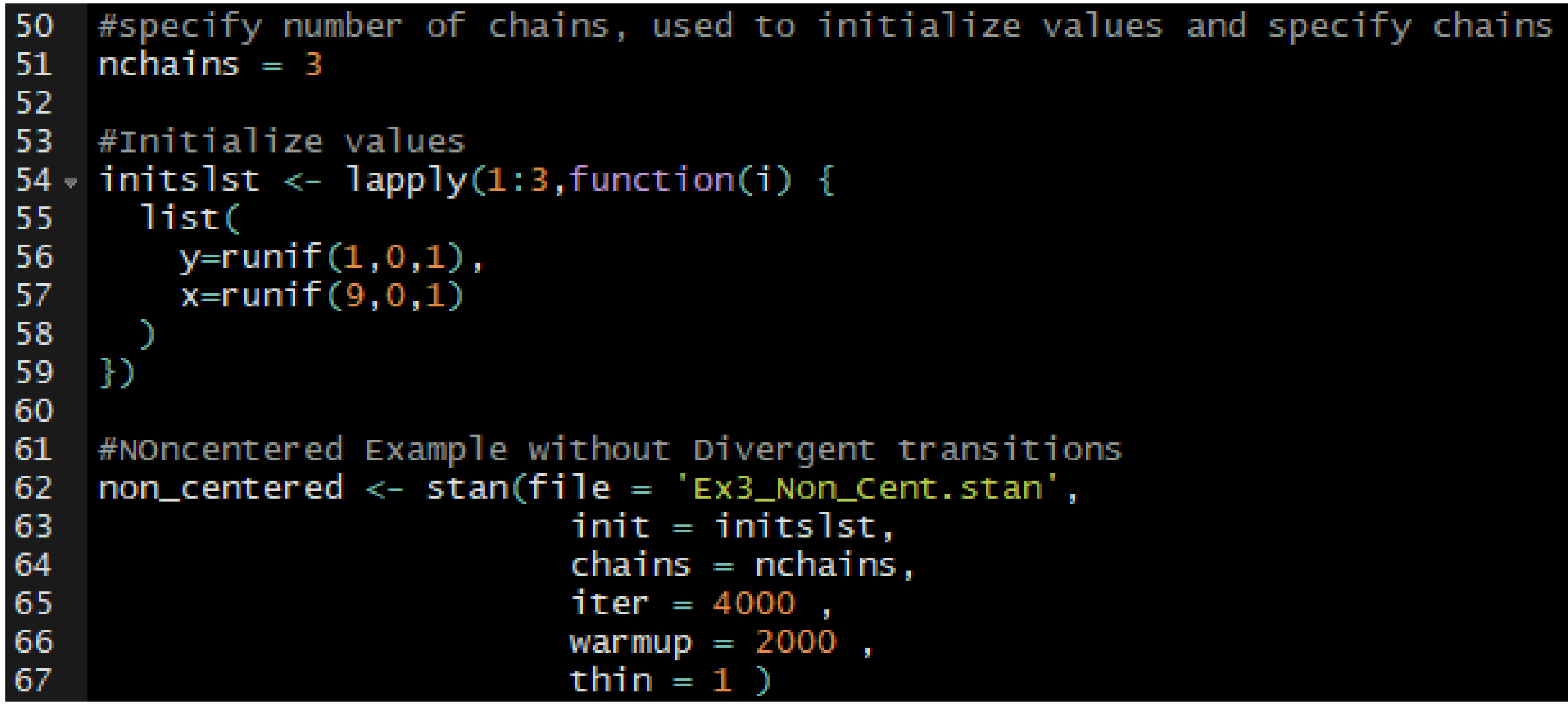
```
parameters {
  real y_raw;
  vector[9] x_raw;
}
transformed parameters {
  real y;
  vector[9] x;
  y = 3.0 * y_raw;
  x = exp(y/2) * x_raw;
}
model {
  y_raw ~ normal(0, 1); // implies y ~ normal(0, 3)
  x_raw ~ normal(0, 1); // implies x ~ normal(0, exp(y/2))
}
```

(D) Exercise 3: Non-Centered

- Save Stan code file as “Ex3_Non_Cent.stan”
- Return to “Ex3_Div.R”

(D) Exercise 3: Non-Centered

6. Specify chains
7. Create initialization list
8. Send to Stan

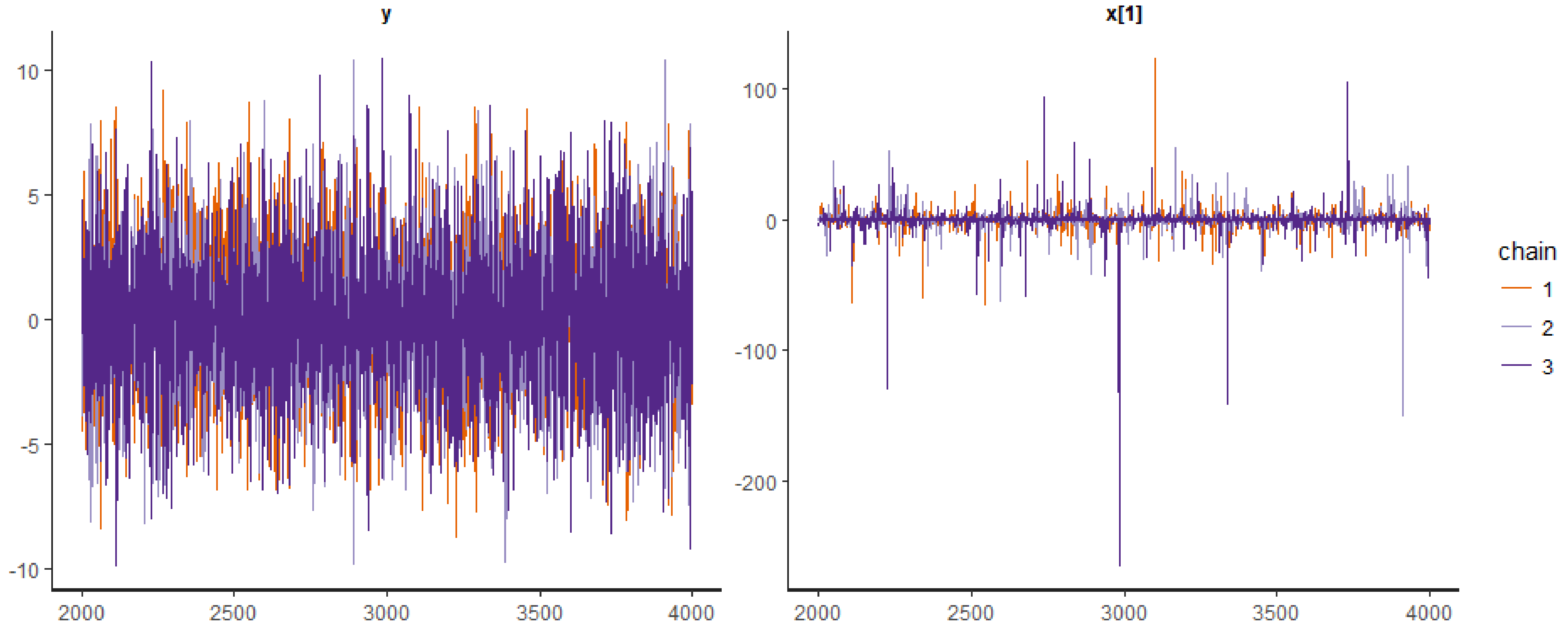


```r
#specify number of chains, used to initialize values and specify chains
nchains = 3

#Initialize values
initslst <- lapply(1:3,function(i) {
  list(
    y=runif(1,0,1),
    x=runif(9,0,1)
  )
})

#NOncentered Example without Divergent transitions
non_centered <- stan(file = 'Ex3_Non_Cent.stan',
                           init = initslst,
                           chains = nchains,
                           iter = 4000 ,
                           warmup = 2000 ,
                           thin = 1 )
```

(E) Exercise 3: Non-Centered

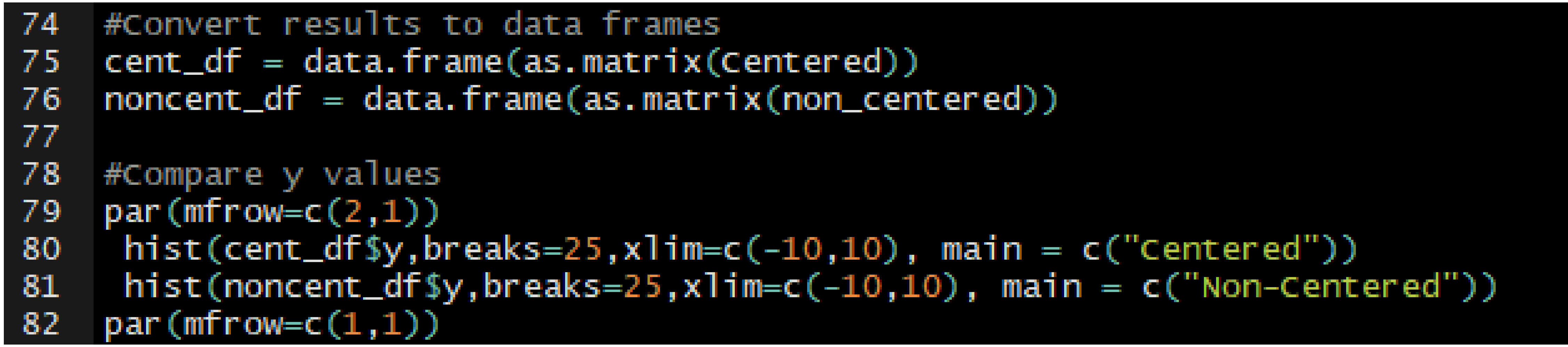
- Save Stan code file as “Ex3_Non_Cent.stan”
- Return to “Ex3_Div.R”
- Run lines 51-67
- Stan compiles program (takes ~30-60 seconds)
- Review Results

(E) traceplot(non_centered,pars=c("y","x[1]"))



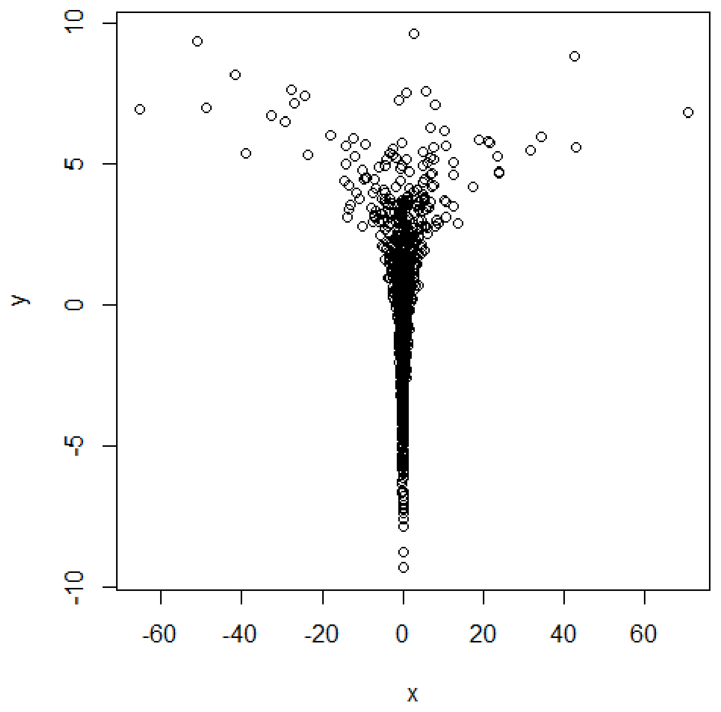
(D) Exercise 3: Non-Centered

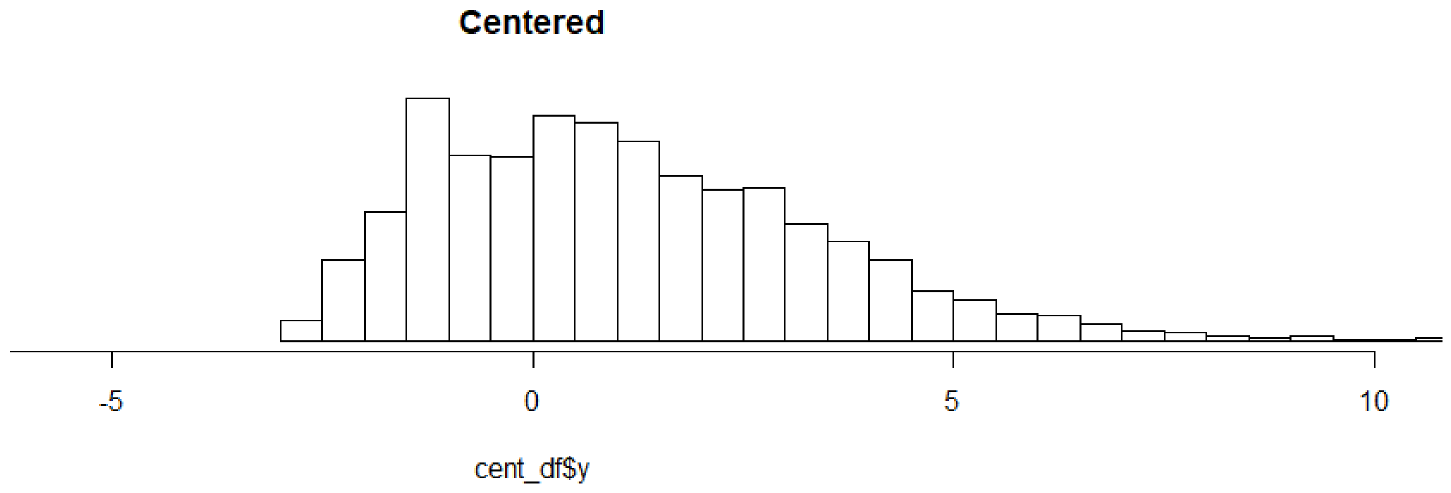
- Compare Y from both models
- Run lines 74-82

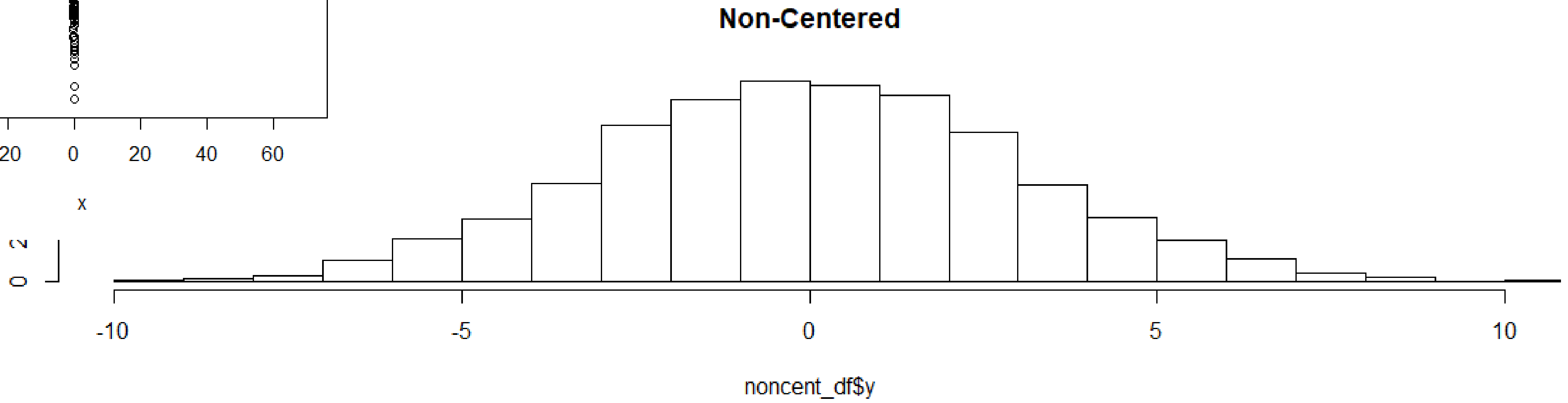


```
74  #Convert results to data frames
75  cent_df = data.frame(as.matrix(Centered))
76  noncent_df = data.frame(as.matrix(non_centered))
77
78  #Compare y values
79  par(mfrow=c(2,1))
80   hist(cent_df$y,breaks=25,xlim=c(-10,10), main = c("Centered"))
81   hist(noncent_df$y,breaks=25,xlim=c(-10,10), main = c("Non-Centered"))
82  par(mfrow=c(1,1))
```

(D) Exercise 3: Non-Centered







Questions?

Shinystan

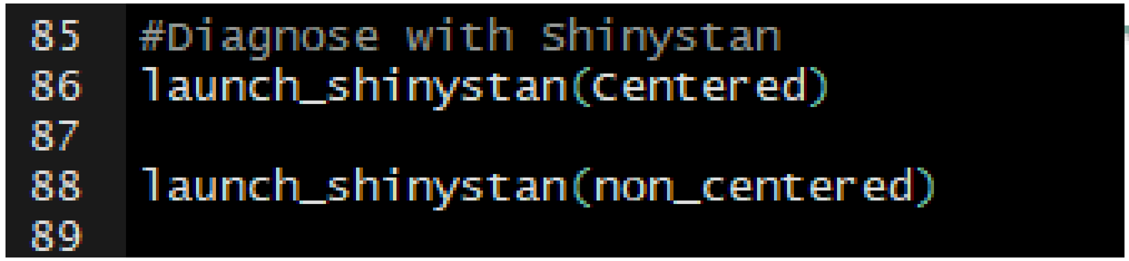
- Provides interactive visual and numerical summaries of model parameters
- Provides convergent diagnostics for MCMC simulations
- Can be used with output from many MCMC programs
  - e.g., Stan, JAGS, BUGS, MCMCPack, NIMBLE, emcee, and SAS
- Using the Shiny web application framework
- I consider this a must have package to compliment Stan

Shinystan

- Stan objects can be passed directly to Shinystan
- Output from other packages must be a mcmc.list object from the coda package.

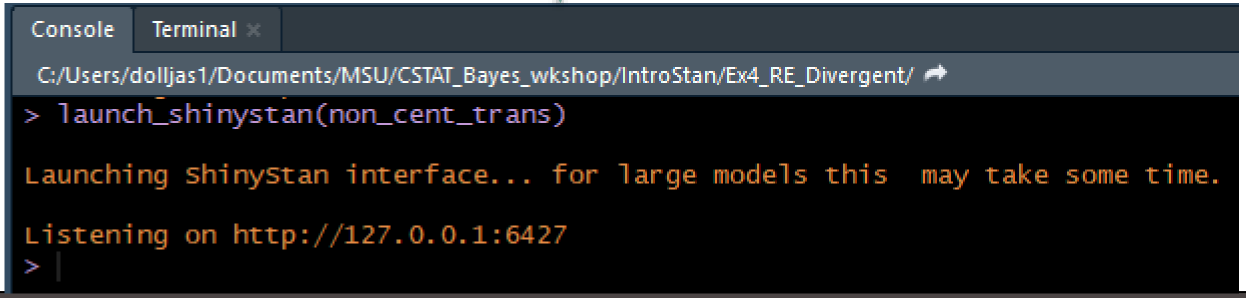
Shinystan

- The package should have already been loaded (see line 10 of Ex3_Div.R).
- Run line 86 to launch view the results of the “Centered” model using shinystan



```
85  #Diagnose with Shinystan
86  launch_shinystan(Centered)
87
88  launch_shinystan(non_centered)
89
```

Returns in Console



```
C:/Users/dolljas1/Documents/MSU/CSTAT_Bayes_wkshop/IntroStan/Ex4_RE_Divergent/
> launch_shinystan(non_cent_trans)

Launching ShinyStan interface... for large models this  may take some time.

Listening on http://127.0.0.1:6427
>
```

Save & Close

SHINYSTAN DIAGNOSE ESTIMATE EXPLORE MORE

ShinyStan

Model:

Ex3_Cent

DIAGNOSE ESTIMATE EXPLORE MORE

Shinystan

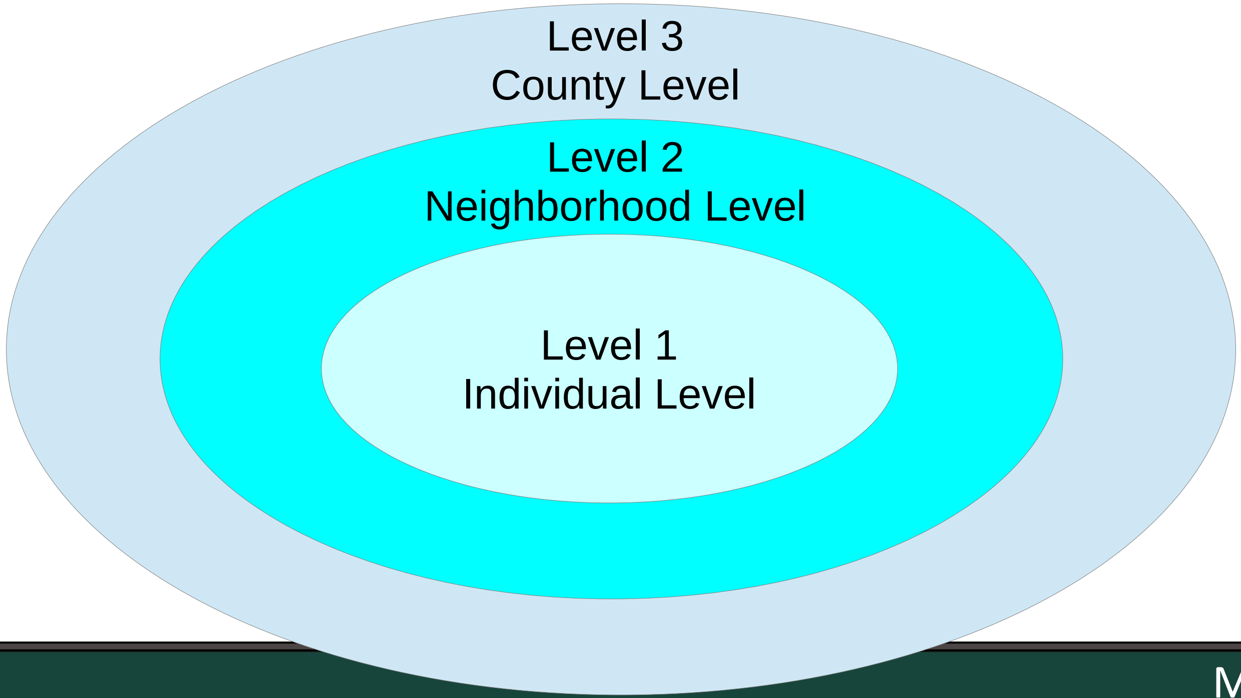
- The package should have already been loaded (see line 10 of Ex3_Div.R).
- Run line 86 to launch view the results of the “Centered” model using shinystan
- Explore output, close browser window when finished
- Run line 88 to examine results of “non_centered” model

Exercise 4: Mixed Effects Model

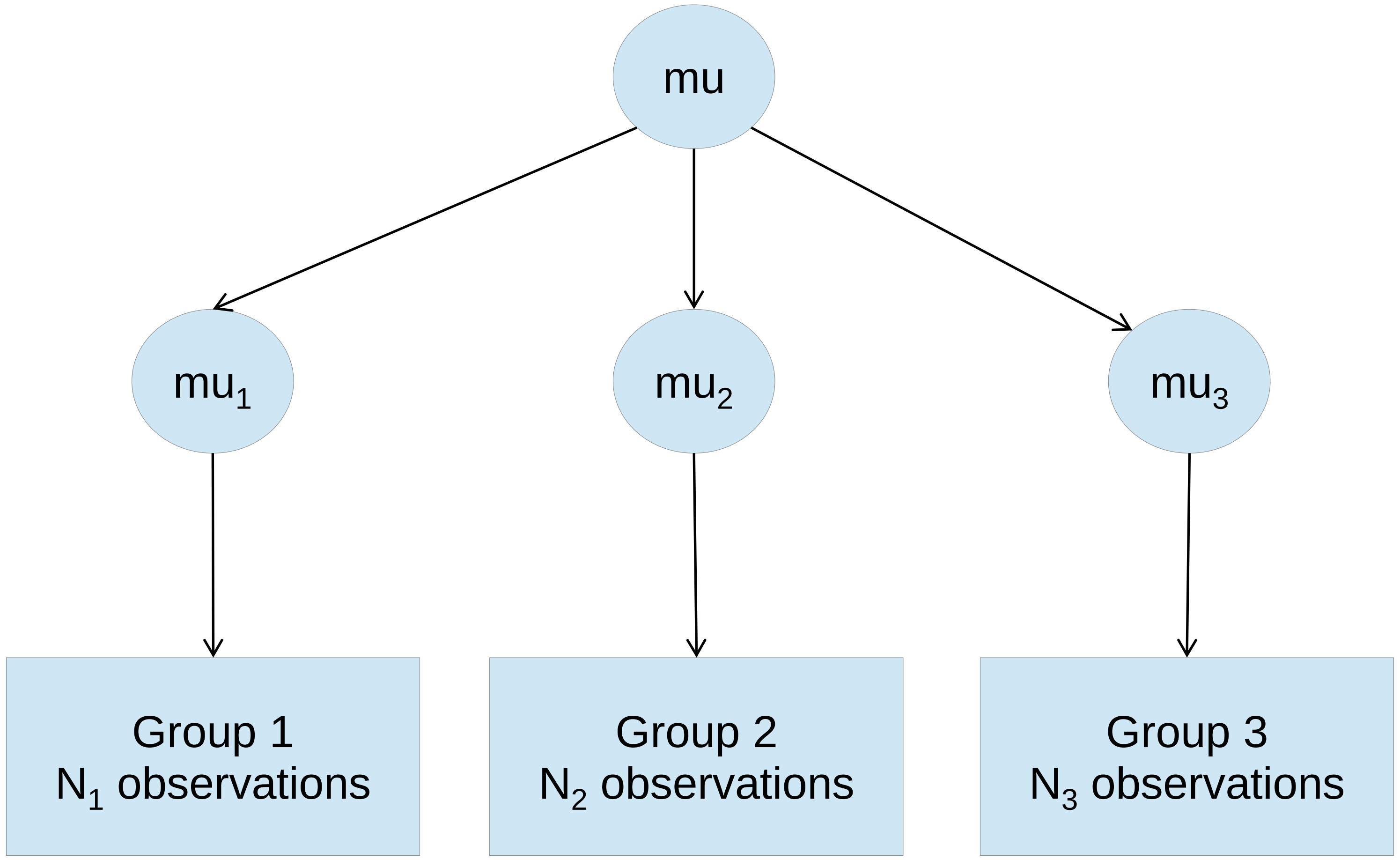
- Also called Hierarchical model

What are Mixed Effects Models?

- Also called – hierarchical model, random-effects, random coefficients, multilevel model, nested models
- A model which some coefficients vary by some level



Classic Mixed Effects Models



Hyperparameter

Per-group parameters

Multiple groups of observations

Exercise 4: Mixed Effects Model

- This exercise will demonstrate how to reparametrize a random effects model with non-centered prior
- First step, fit standard random effects model
- Second step, modify random effects term to eliminate divergent transitions

Exercise 4: Mixed Effects Model

A. Describe data and model
B. Review R code (provided)
C. Review Stan code
D. You program the *.stan file following the screen or handout
E. Enjoy your results!

(A) Exercise 4: Mixed Effects Model

- We will simulate data from 4 groups with 10 observations per group
- We will create two Stan programs
  - 1. Centered parameterization (results in Divergent Transitions)
  - 2. Non-centered parametrization
- Inspect results and compare output

(A) Exercise 4: Mixed Effects Model

Model: $\mathrm{y}_i = \alpha_{j[i]} + \beta \mathrm{x}_i + \varepsilon_i$

$$\varepsilon_i \sim \text{Normal}(0, \sigma)$$

Priors:

$\alpha_j \sim \text{Normal}(\mu, \tau)$

$\beta \sim \text{Normal}(0,100)$

$\sigma \sim \text{half} - \text{cauchy}(0,5)$

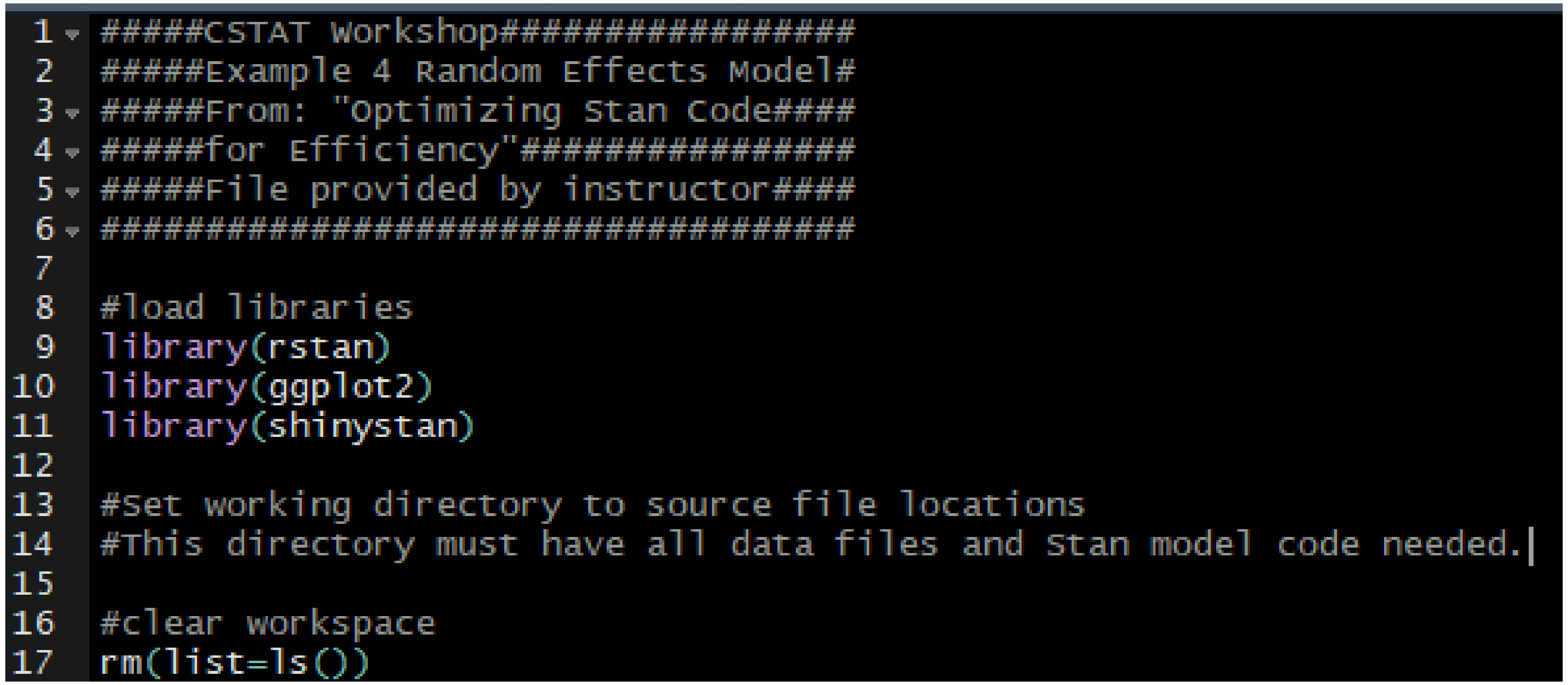
Hyperpriors:

$\mu \sim \text{Normal}(0,100)$

$\tau \sim \text{half} - \text{cauchy}(0,5)$

(B) Exercise 4: Mixed Effects Model

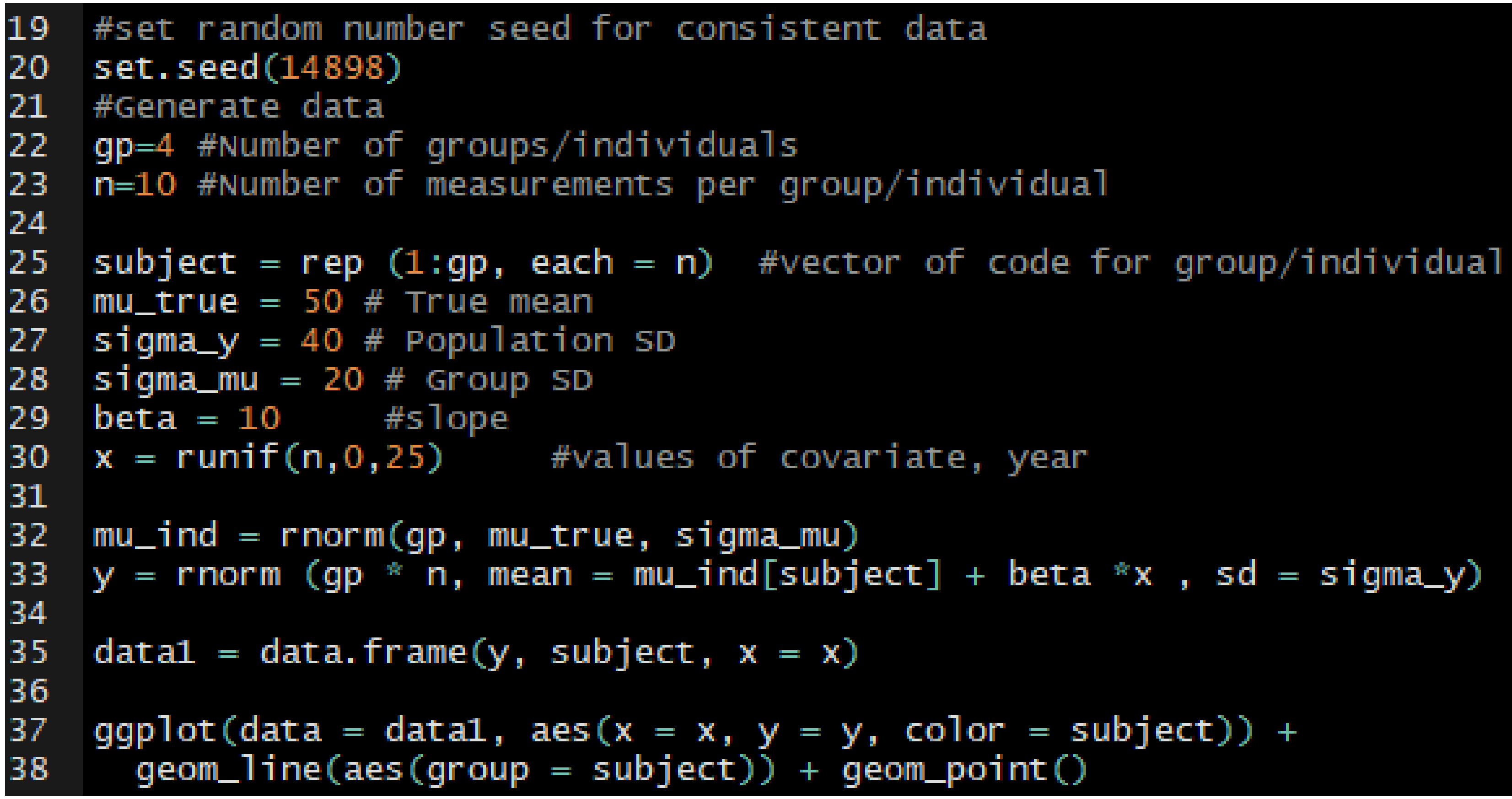
1. Open “Ex4_RE.R”
2. Load libraries
3. Set working directory
4. Clear workspace



```
#####CSTAT Workshop################
#####Example 4 Random Effects Model#
#####From: "Optimizing Stan Code####
#####for Efficiency"################
#####File provided by instructor####
###################################

#load libraries
library(rstan)
library(ggplot2)
library(shinystan)

#Set working directory to source file locations
#This directory must have all data files and Stan model code needed.

#clear workspace
rm(list=ls())
```

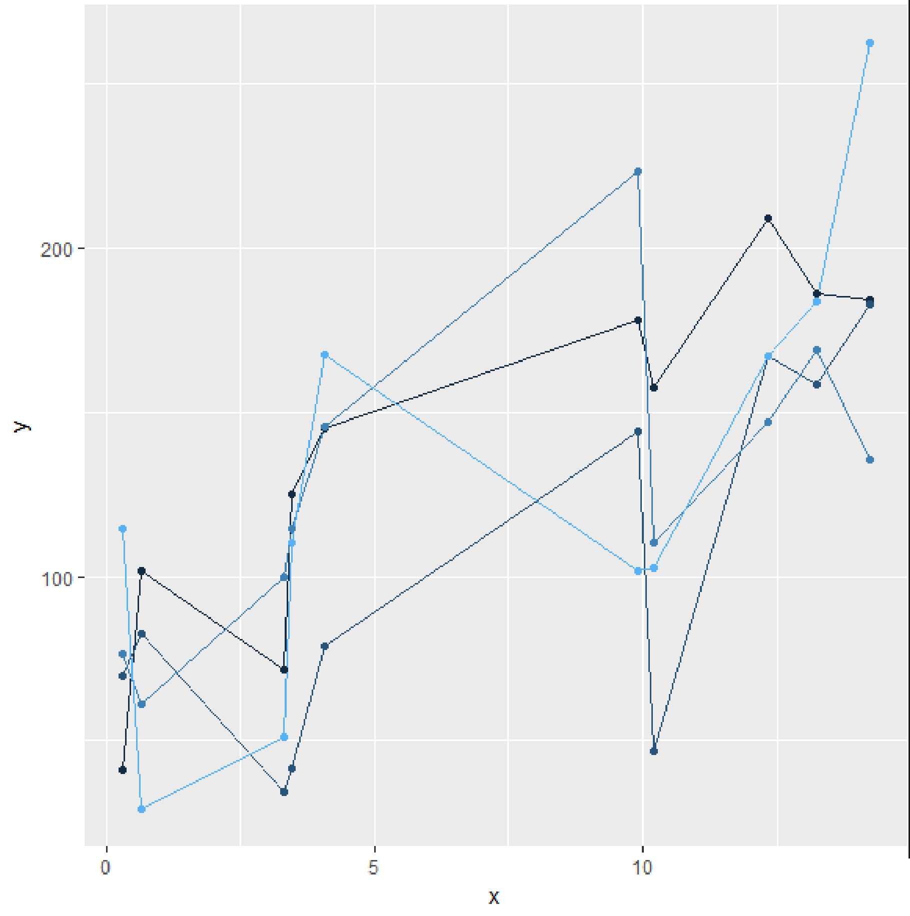
(B) Exercise 4: Mixed Effects Model

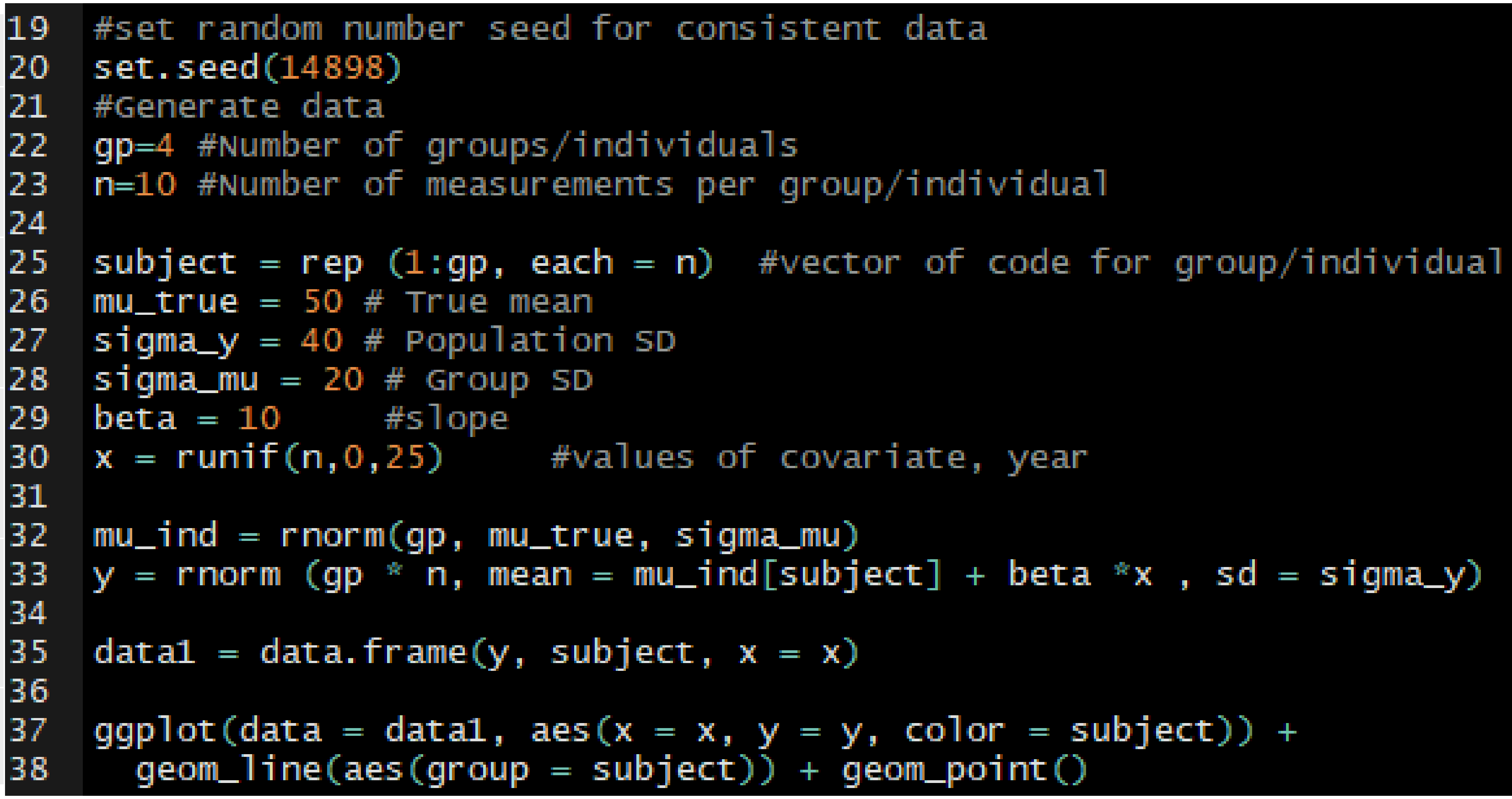
5. Generate data
6. View data



```r
#set random number seed for consistent data
set.seed(14898)
#Generate data
gp=4 #Number of groups/individuals
n=10 #Number of measurements per group/individual

subject = rep (1:gp, each = n)  #vector of code for group/individual
mu_true = 50 # True mean
sigma_y = 40 # Population SD
sigma_mu = 20 # Group SD
beta = 10      #slope
x = runif(n,0,25)     #values of covariate, year

mu_ind = rnorm(gp, mu_true, sigma_mu)
y = rnorm (gp * n, mean = mu_ind[subject] + beta *x , sd = sigma_y)

data1 = data.frame(y, subject, x = x)

ggplot(data = data1, aes(x = x, y = y, color = subject)) +
  geom_line(aes(group = subject)) + geom_point()
```

(B) Exercise 4: Mixed Effects Model





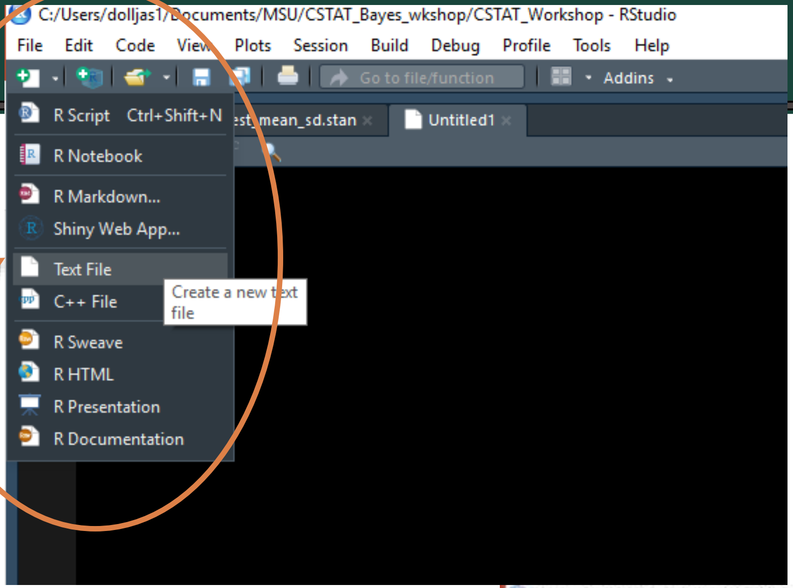
```r
#set random number seed for consistent data
set.seed(14898)
#Generate data
gp=4 #Number of groups/individuals
n=10 #Number of measurements per group/individual

subject = rep (1:gp, each = n)  #vector of code for group/individual
mu_true = 50 # True mean
sigma_y = 40 # Population SD
sigma_mu = 20 # Group SD
beta = 10     #slope
x = runif(n,0,25)     #values of covariate, year

mu_ind = rnorm(gp, mu_true, sigma_mu)
y = rnorm (gp * n, mean = mu_ind[subject] + beta *x , sd = sigma_y)

data1 = data.frame(y, subject, x = x)

ggplot(data = data1, aes(x = x, y = y, color = subject)) +
  geom_line(aes(group = subject)) + geom_point()
```

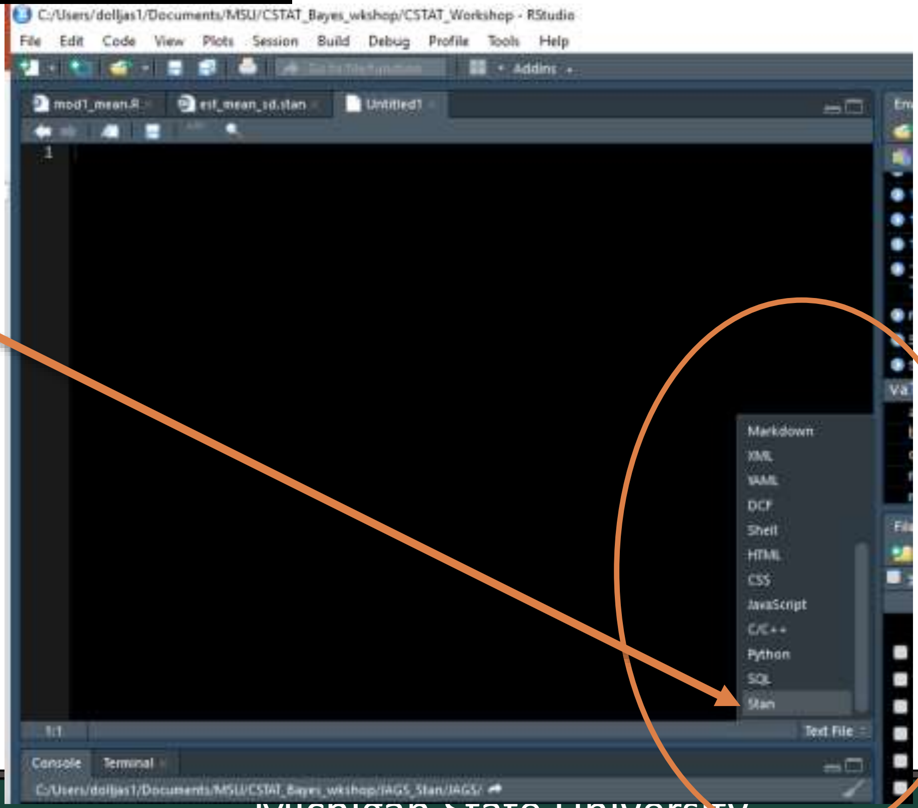
(C) Exercise 4: Centered

- Create *.stan file for centered parameterization
- Try on your own following the handout

(C) Exercise 4

- Create new text file
- Change "Text File" to "Stan"




C:/Users/dolljas1/Documents/MSU/CSTAT_Bayes_wkshop/CSTAT_Workshop - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Addins
mod1_mean.R
est_mean_sd.stan
Untitled1
Markdown
XML
YAML
DCF
Shell
HTML
CSS
JavaScript
C/C++
Python
SQL
Stan
Text File
Console
Terminal
C:/Users/dolljas1/Documents/MSU/CSTAT_Bayes_wkshop/JAGS_Stan/JAGS/

(C) Exercise 4: RE Centered

```
data{
  int<lower=0> n;  //number of observations
  vector[n] y;     //observations as a vector
  vector[n] x;     //observed x values, predictors
  int gp[n];       //subject indicator
  int ngps;        //number of groups
}
```

```
parameters{
  real<lower=0> sigma_y;   //individual standard deviation to be estimated
  real<lower=0> sigma_mu;  //group standard deviation to be estimated
  real gpmu[ngps];         //Group mean
  real mu;                 //Global mean across all groups
  real beta;               //slope
}
```

```
model {
  //reference priors
  sigma_y ~ cauchy(0,5);
  sigma_mu ~ cauchy(0,5);
  beta ~ normal(0,100);
  mu ~ normal(0,100); //overall mean

  for(k in 1:ngps){
    gpmu[k] ~ normal(mu,sigma_mu); //individual group means
  }

  //likelihood, loop through number of observations
  for(i in 1:n){
    y[i] ~ normal(gpmu[gp[i]] + beta * x[i], sigma_y);
  }
}
```

(C) Exercise 4: RE Centered

```
data{
  int<lower=0> n; //number of observations
  vector[n] y;    //response as a vector
  vector[n] x;    //observed x values, predictors
  int gp[n];      //subject indicator
  int ngps;       //number of groups
}
```

(C) Exercise 4: RE Centered

```
parameters{
 real<lower=0> sigma_y;   //individual standard deviation to be estimated
  real<lower=0> sigma_mu;  //group standard deviation to be estimated
  real gpmu[ngps];         //Group mean
  real mu;                 //Global mean across all groups
  real beta;               //slope
}
```

(C) Exercise 4: RE Centered

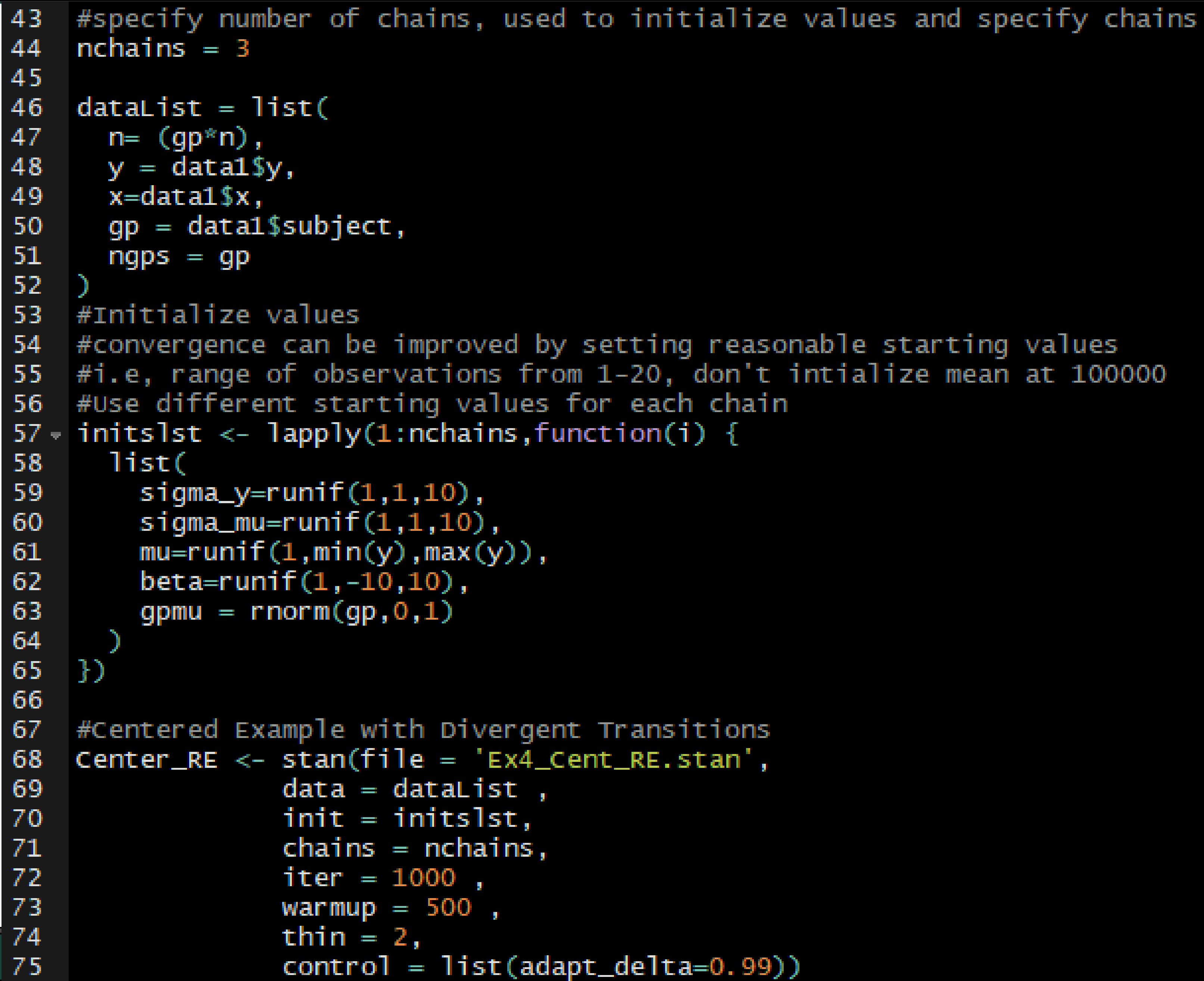
```
model {
  //reference priors
  sigma_y ~ cauchy(0,5);
  sigma_mu ~ cauchy(0,5);
  beta ~ normal(0,100);
  mu ~ normal(0,100); //overall mean

  for(k in 1:ngps){
    gpmu[k] ~ normal(mu,sigma_mu); //individual group means
  }

  //likelihood, loop through number of observations
  for(i in 1:n){
    y[i] ~ normal(gpmu[gp[i]] + beta * x[i], sigma_y);
  }
}
```

(D) Exercise 4: RE Centered

- Save Stan code file as “Ex4_Cent_RE.stan”
- Return to “Ex4_RE.R”

(D) Exercise 4: RE Centered

6. Specify chains
7. Create initialization list
8. Send to Stan

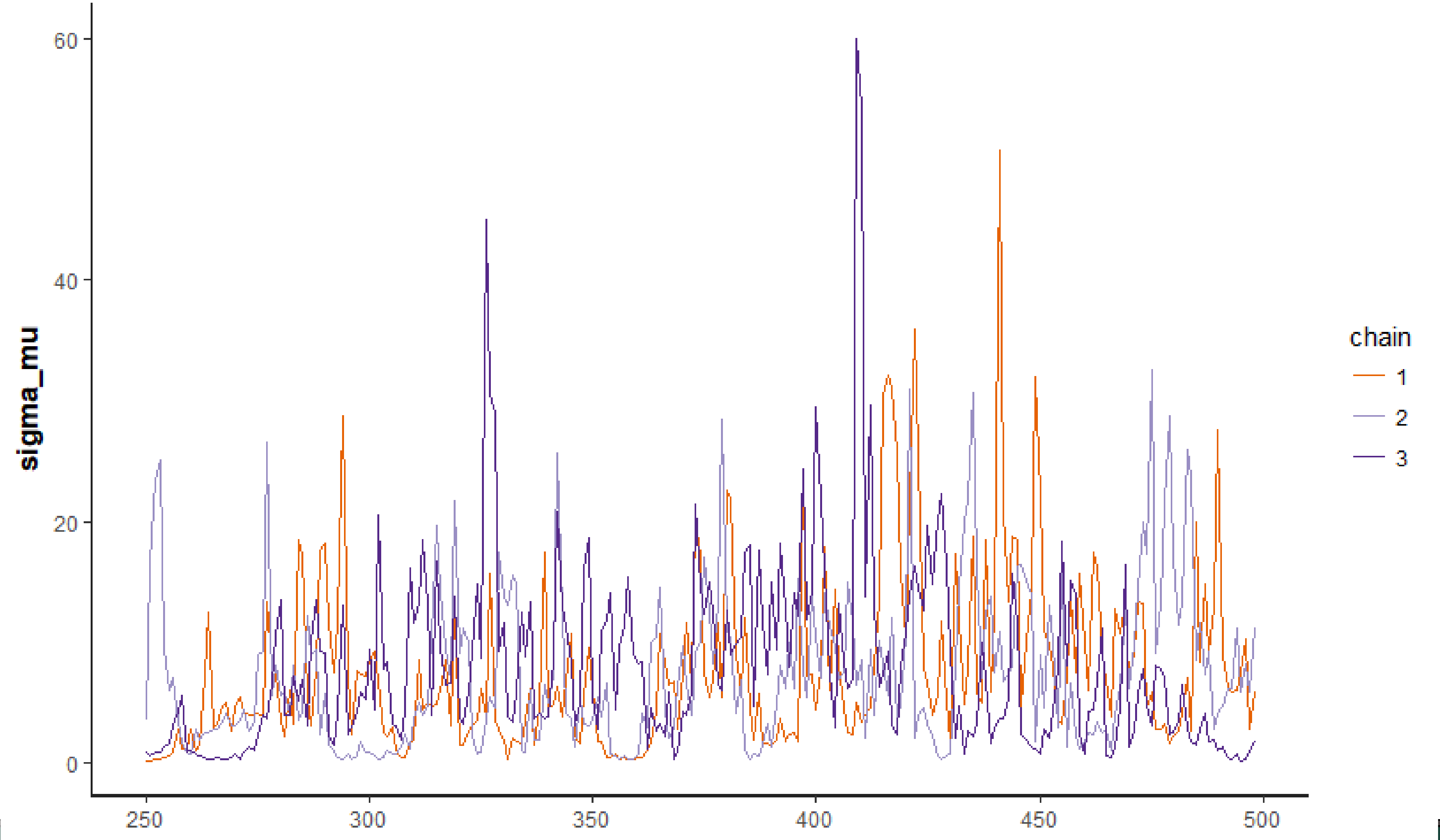


```r
#specify number of chains, used to initialize values and specify chains
nchains = 3

dataList = list(
  n= (gp*n),
  y = data1$y,
  x=data1$x,
  gp = data1$subject,
  ngps = gp
)
#Initialize values
#convergence can be improved by setting reasonable starting values
#i.e, range of observations from 1-20, don't intialize mean at 100000
#Use different starting values for each chain
initslst <- lapply(1:nchains,function(i) {
  list(
    sigma_y=runif(1,1,10),
    sigma_mu=runif(1,1,10),
    mu=runif(1,min(y),max(y)),
    beta=runif(1,-10,10),
    gpmu = rnorm(gp,0,1)
  )
})

#Centered Example with Divergent Transitions
Center_RE <- stan(file = 'Ex4_Cent_RE.stan',
             data = dataList ,
             init = initslst,
             chains = nchains,
             iter = 1000 ,
             warmup = 500 ,
             thin = 2,
             control = list(adapt_delta=0.99))
```

(E) Exercise 4: RE Centered

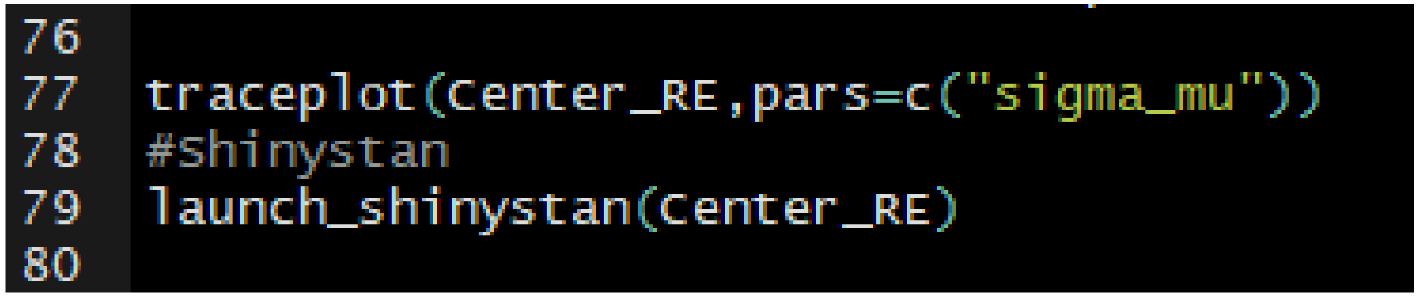
- Save Stan code file as “Ex4_Cent_RE.stan”
- Return to “Ex4_RE.R”
- Run lines 1-78
- Stan compiles program (takes ~30-60 seconds)
- Review Results

(E) traceplot(Centered_RE,pars=c(“sigma_mu"))



(E) Exercise 4: RE Centered

- View results in shinystan



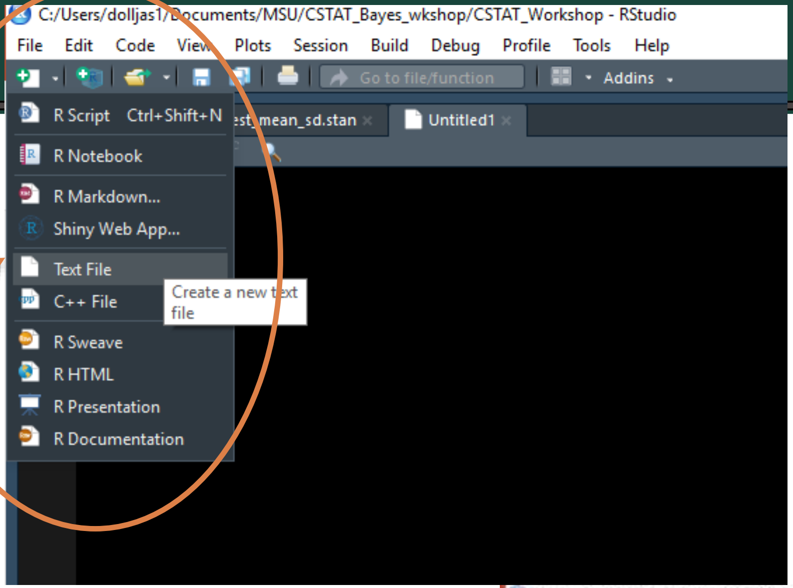
```
traceplot(Center_RE,pars=c("sigma_mu"))
#Shinystan
launch_shinystan(Center_RE)
```

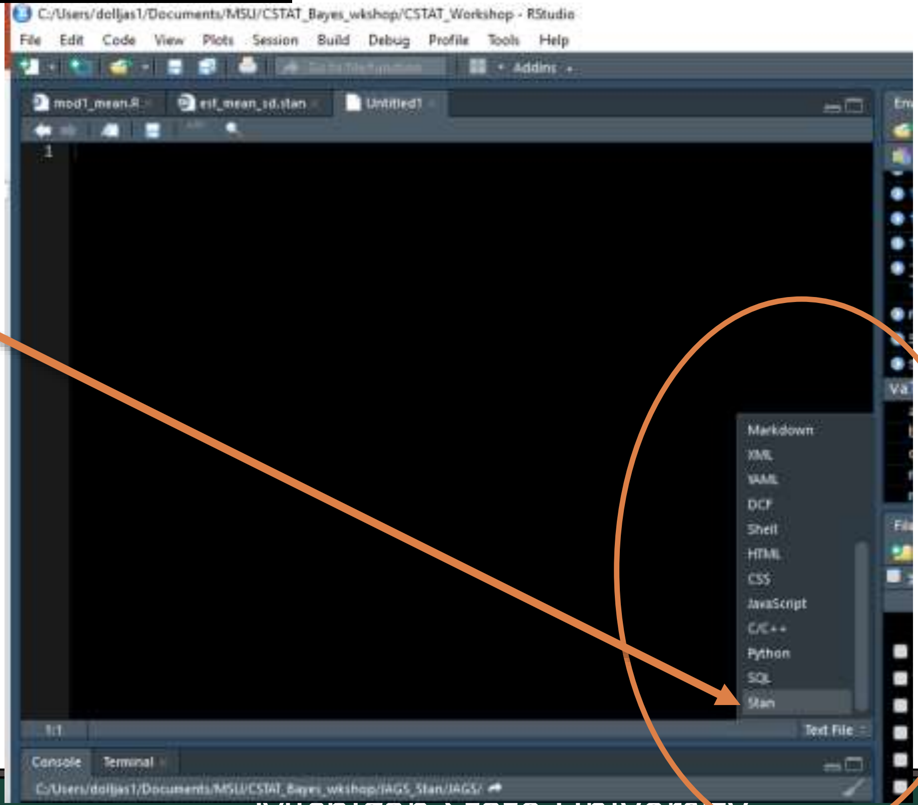
(C) Exercise 4: RE Non-Centered

- Create *.stan file for non-centered parameterization
- Try on your own following the handout

(C) Exercise 4

- Create new text file
- Change "Text File" to "Stan"




C:/Users/dolljas1/Documents/MSU/CSTAT_Bayes_wkshop/CSTAT_Workshop - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Addins
mod1_mean.R
est_mean_sd.stan
Untitled1
Markdown
XML
YAML
DCF
Shell
HTML
CSS
JavaScript
C/C++
Python
SQL
Stan
Text File
Console
Terminal

(C) Exercise 4: RE Non-Centered Reminder

1. Suppose original parameter is **beta**
2. Create new parameter and name it **beta_raw**
3. Give **beta_raw** a normal prior, mean = 0 and SD = 1
4. Move original beta parameter declaration to "*transformed parameter block*"
5. Equate original parameter in "*transformed parameter*" block to **beta = mean + sd * beta_raw**

Note: Above ONLY when original parameter is normally distributed. See Stan manual for other distributions

(C) Exercise 4: RE Non-Centered

```
data{
  int<lower=0> n; //number of obserations
  vector[n] y;    //observations as a vector
  vector[n] x;    //observed x values, predictors
  int gp[n];      //subject indicator
  int ngps;       //number of groups
}
```

(C) Exercise 4: RE Non-Centered

```
parameters{
 real<lower=0> sigma_y;    //individual standard deviation to be estimated
  real<lower=0> sigma_mu;  //group standard deviation to be estimated
  real gpmu[ngps];         //Group mean
  real mu;                 //Global mean across all groups
  real beta;               //slope
}
```

```
parameters{
  real<lower=0> sigma_y;   //individual standard deviation to be estimated
  real<lower=0> sigma_mu;  //group standard deviation to be estimated
  real gpmu_raw[ngps];     //Group mean_raw
  real mu;                 //Global mean across all groups
  real beta;               //slope
}
```

(C) Exercise 4: RE Non-Centered

NEW BLOCK

```
transformed parameters{
  real gpmu[ngps];                    //Group mean

  for(k in 1:ngps){
    gpmu[k] = mu + gpmu_raw[k] * sigma_mu;
  }
}
```

(C) Exercise 4: RE Non-Centered

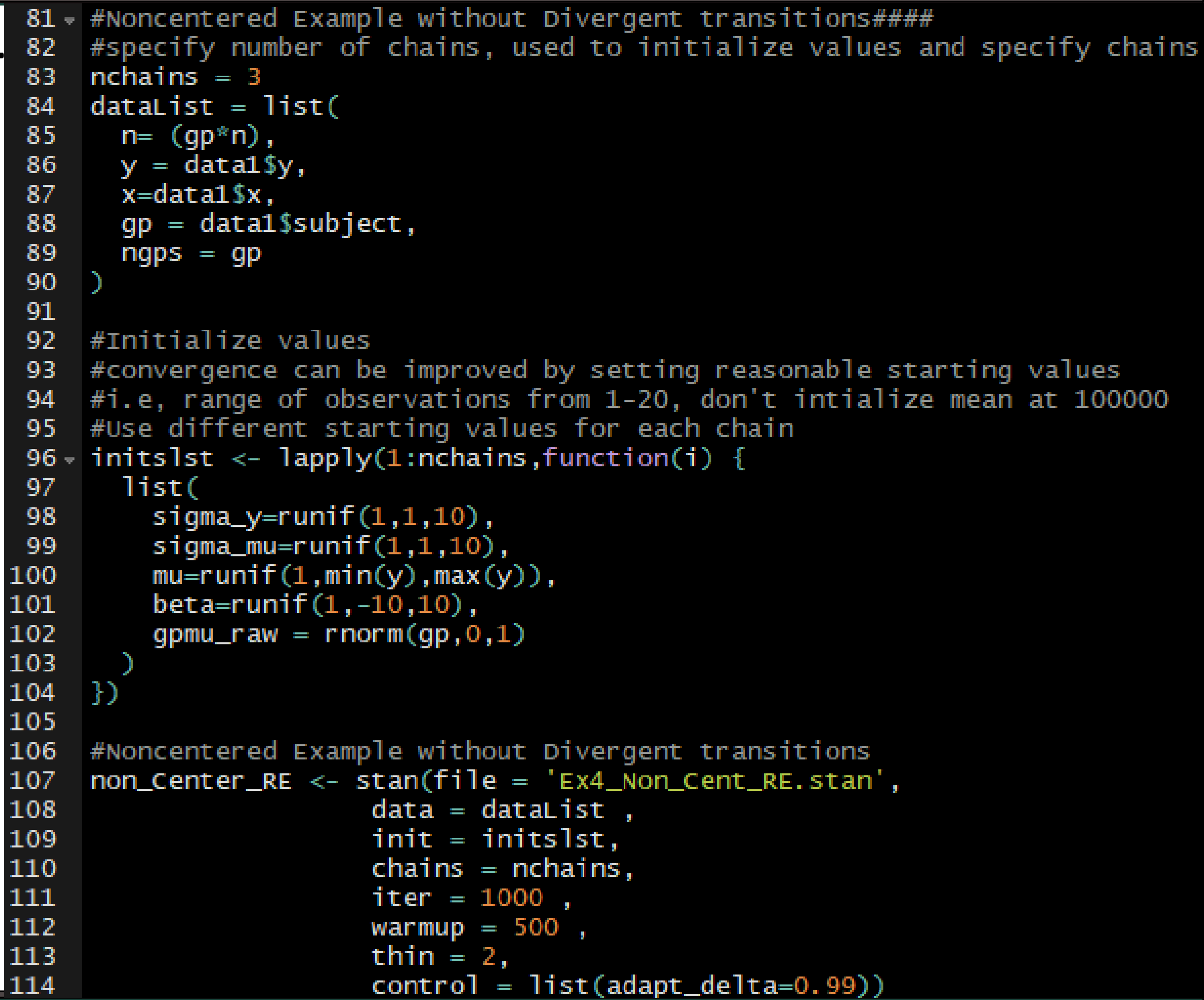
```
model {
  //reference priors
  sigma_y ~ cauchy(0,5);
  sigma_mu ~ cauchy(0,5);
  beta ~ normal(0,100);
  mu ~ normal(0,100);

  for(k in 1:ngps){
    gpmu_raw[k] ~ normal(0,1);//implies gpmu[k] ~ normal(mu,sigma_mu);
  }
  //likelihood, loop through number of observations
  for(i in 1:n){
    y[i] ~ normal(gpmu[gp[i]] + beta * x[i], sigma_y);
  }
}
```

(D) Exercise 4: RE Non-Centered

- Save Stan code file as “Ex4_Non_Cent_RE.stan”
- Return to “Ex4_RE.R”

(D) Exercise 4: RE Non-Centered

6. Specify chains
7. Create initialization list
   a) Note: gpmu_raw
8. Send to Stan

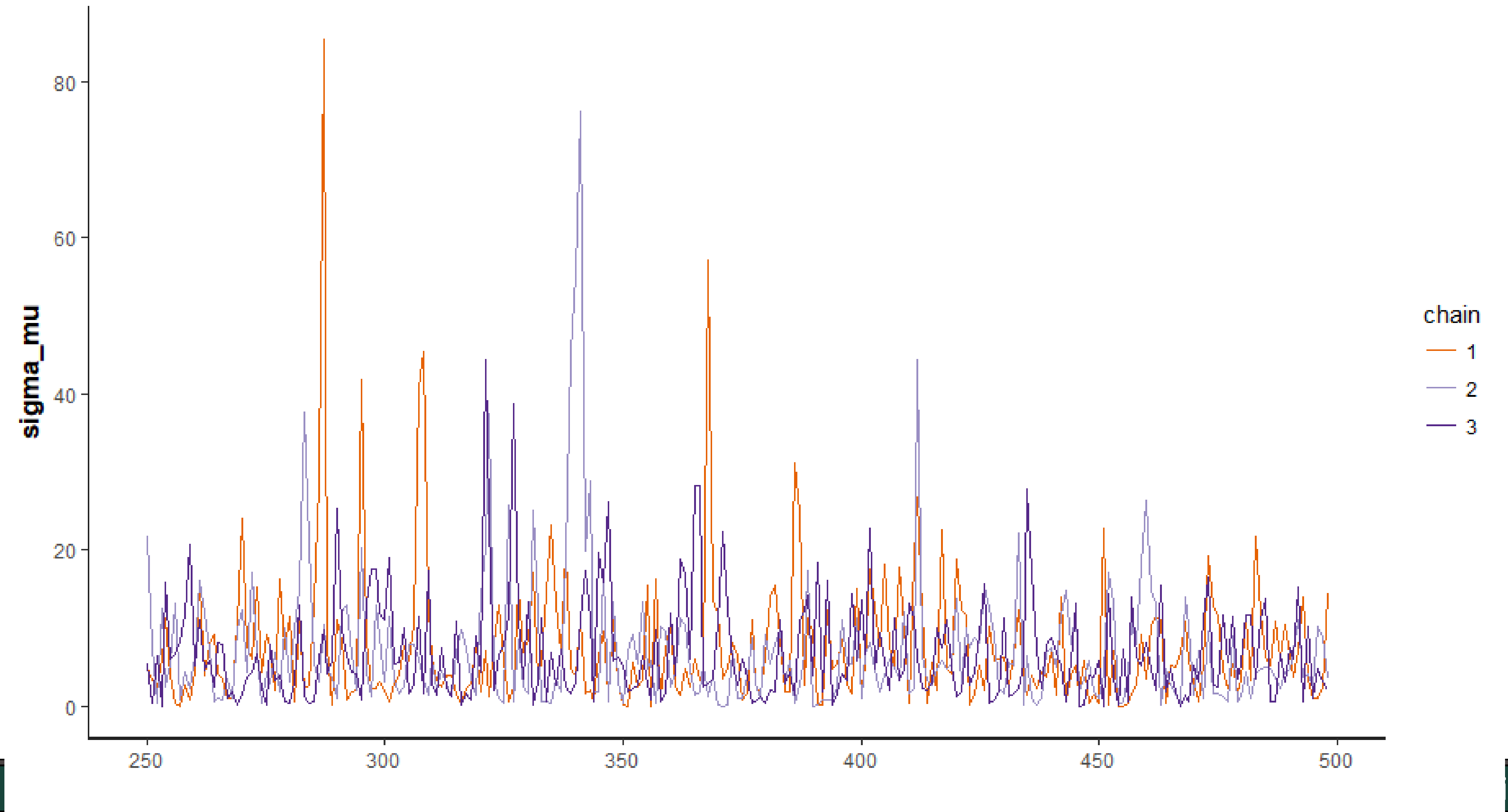


```
#Noncentered Example without Divergent transitions####
#specify number of chains, used to initialize values and specify chains
nchains = 3
dataList = list(
  n= (gp*n),
  y = data1$y,
  x=data1$x,
  gp = data1$subject,
  ngps = gp
)

#Initialize values
#convergence can be improved by setting reasonable starting values
#i.e, range of observations from 1-20, don't intialize mean at 100000
#Use different starting values for each chain
initslst <- lapply(1:nchains,function(i) {
  list(
    sigma_y=runif(1,1,10),
    sigma_mu=runif(1,1,10),
    mu=runif(1,min(y),max(y)),
    beta=runif(1,-10,10),
    gpmu_raw = rnorm(gp,0,1)
  )
})

#Noncentered Example without Divergent transitions
non_Center_RE <- stan(file = 'Ex4_Non_Cent_RE.stan',
                  data = dataList ,
                  init = initslst,
                  chains = nchains,
                  iter = 1000 ,
                  warmup = 500 ,
                  thin = 2,
                  control = list(adapt_delta=0.99))
```

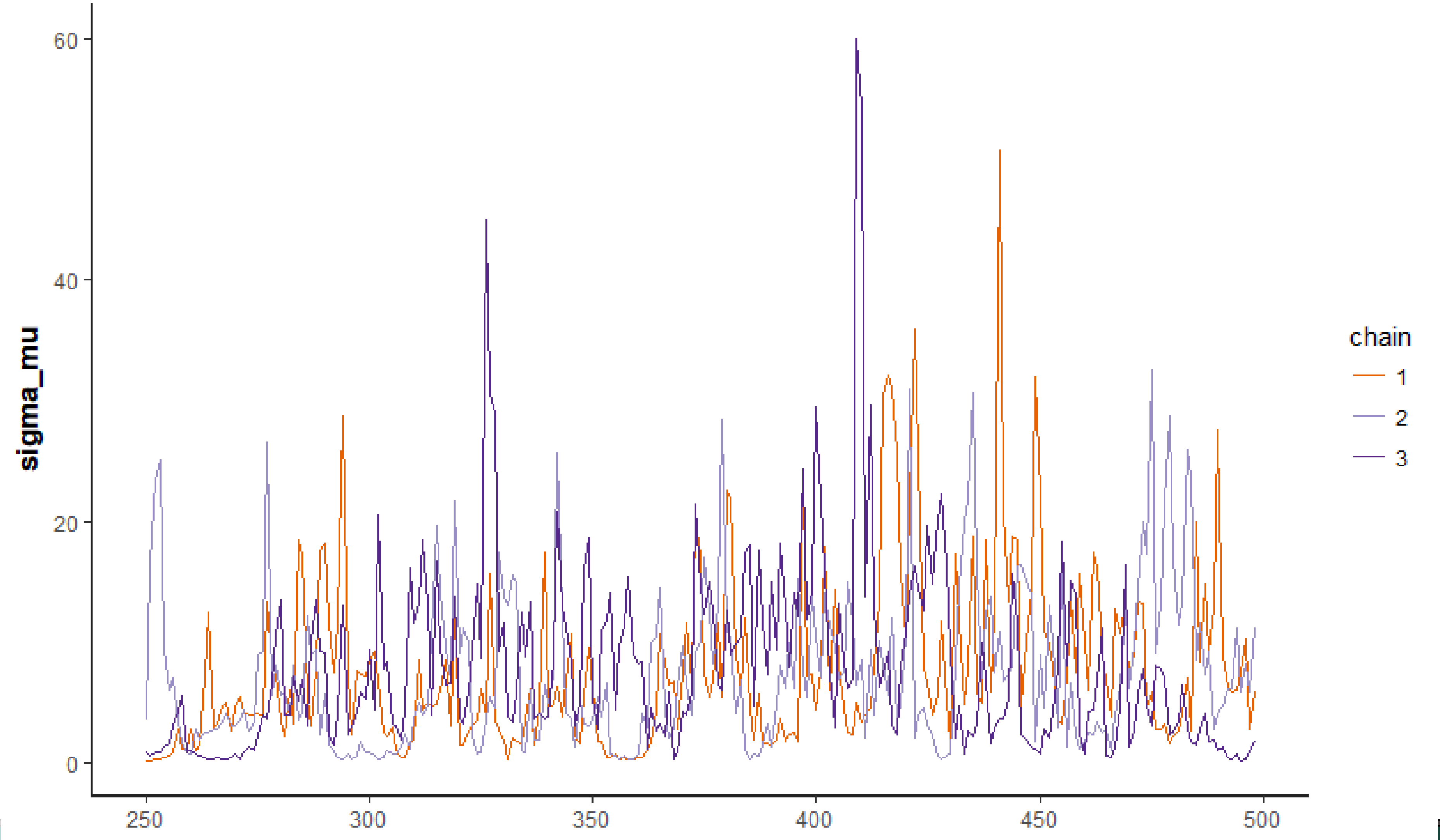
(E) Exercise 4: RE Non-Centered

- Save Stan code file as “Ex3_Non_Cent.stan”
- Return to “Ex3_Div.R”
- Run lines 83-117
- Stan compiles program (takes ~30-60 second)
- Review Results

(E) traceplot(non_Center_RE,pars=c(“sigma_mu"))

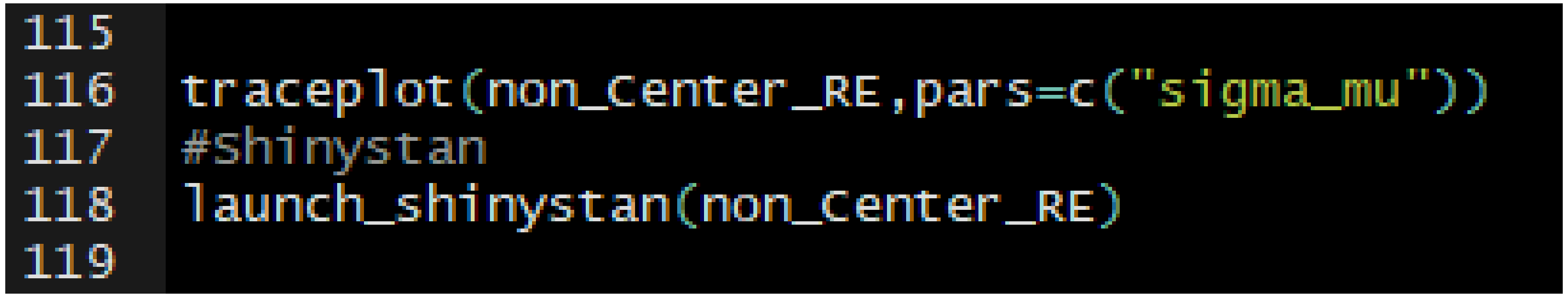


(E) traceplot(Centered_RE,pars=c(“sigma_mu"))



(E) Exercise 4: RE Non-Centered

- View results in shinystan



```

traceplot(non_Center_RE,pars=c("sigma_mu"))
#Shinystan
launch_shinystan(non_Center_RE)

```

Questions?

Model Selection

- Does the mixed effects model describe the data better than a fixed effect only model?
- Hierarchical vs Complete Pooling

Model Selection

- WAIC (Watanabe 2010)
- Fully Bayesian criterion for estimating out-of-sample expectation
- Uses log pointwise posterior predictive density with correction for effective number of parameters
- Lower the WAIC the better

$$\mathrm{WAIC} = -2\left(\widehat{\mathrm{lpd}} - \hat{\mathrm{p}}\right)$$

log pointwise predictive density

Measure of model fit

Penalty for increased model complexity

WAIC in Stan

- Requires loo package
- Requires “*generated quantities*” block to store log likelihood
- loo calculates lpd and p-hat

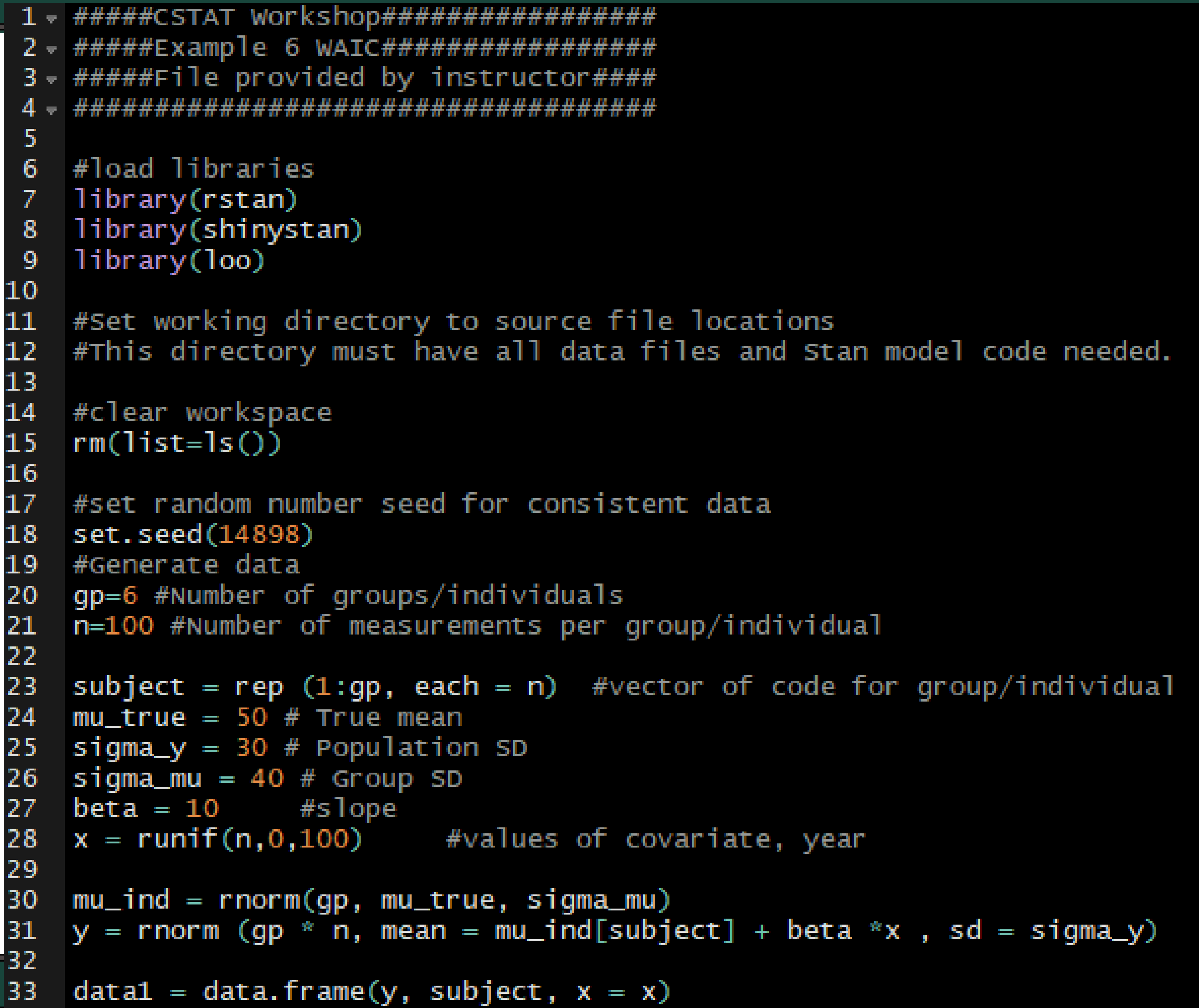
WAIC in Stan

```
generated quantities{
   vector[n] log_lik;      //vector to hold log-likelihood

   for ( i in 1:n ) {
      log_lik[i] = normal_lpdf( obs[i] | location , scale);
   }
}
```

Quantitative Fisheries Center

Ex.5: WAIC in Stan

- Open Ex6_WAIC.R
- Load packages
- Set wd
- Clear workspace
- Simulate data



```
#####CSTAT Workshop################
#####Example 6 WAIC################
#####File provided by instructor####
###################################

#load libraries
library(rstan)
library(shinystan)
library(loo)

#Set working directory to source file locations
#This directory must have all data files and Stan model code needed.

#clear workspace
rm(list=ls())

#set random number seed for consistent data
set.seed(14898)
#Generate data
gp=6 #Number of groups/individuals
n=100 #Number of measurements per group/individual

subject = rep (1:gp, each = n)  #vector of code for group/individual
mu_true = 50 # True mean
sigma_y = 30 # Population SD
sigma_mu = 40 # Group SD
beta = 10     #slope
x = runif(n,0,100)     #values of covariate, year

mu_ind = rnorm(gp, mu_true, sigma_mu)
y = rnorm (gp * n, mean = mu_ind[subject] + beta *x , sd = sigma_y)

data1 = data.frame(y, subject, x = x)
```

Ex.5: WAIC in Stan

- Open Ex5_WAIC_LC.stan and Ex5_WAIC_ME.stan
- Add a “*generated quantities*” block
- Add code to save log-likelihood in a new vector following template on handout.

WAIC in Stan – Ex6_WAIC_LC.stan

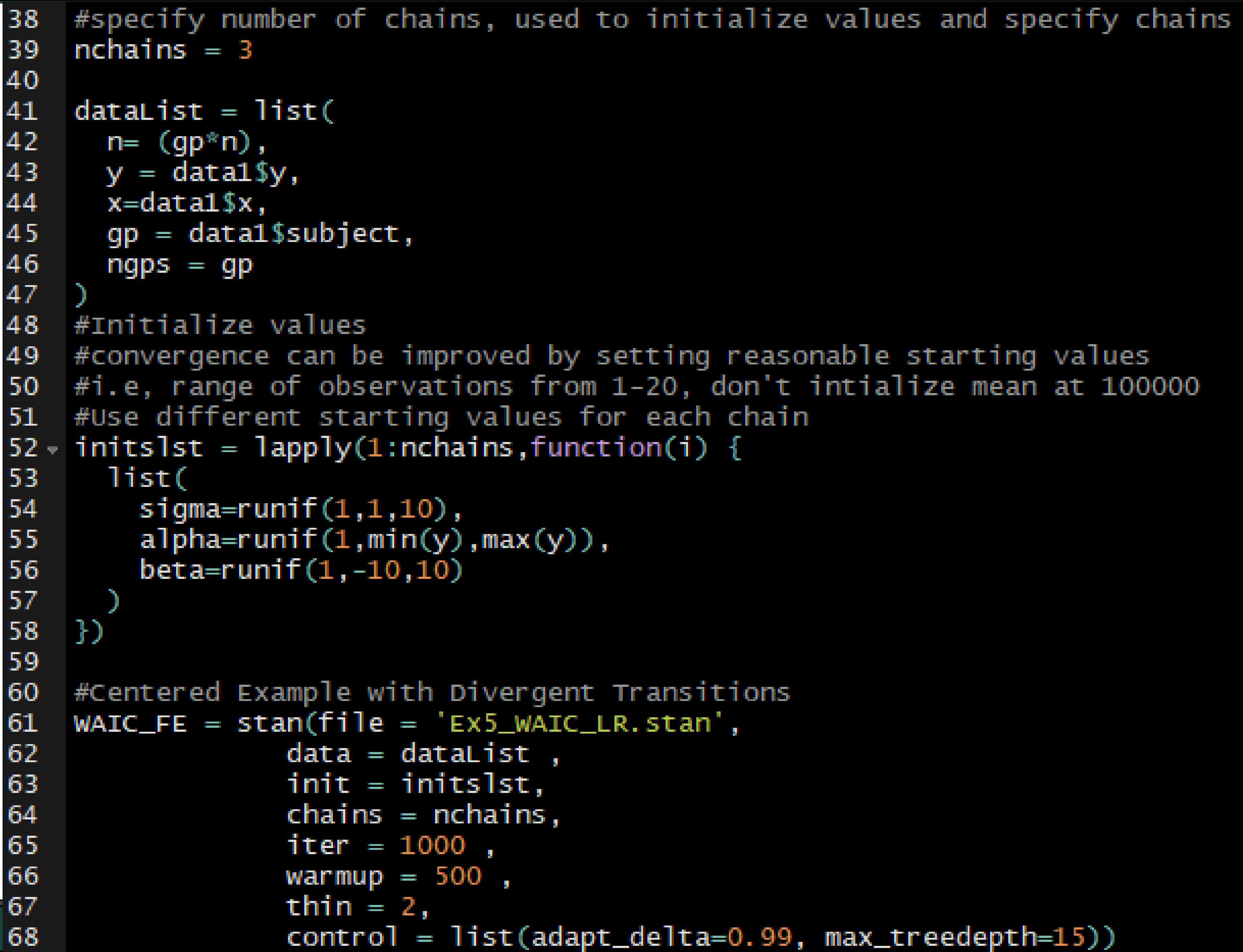
```
generated quantities{
  vector[n] log_lik;      //vector to hold log-likelihood

  for ( i in 1:n ) {
    log_lik[i] = normal_lpdf( y[i] | alpha + beta * x[i], sigma);
  }
}
```

WAIC in Stan – Ex6_WAIC_ME.stan

```
generated quantities{
   vector[n] log_lik;      //vector to hold log-likelihood

   for ( i in 1:n ) {
      log_lik[i] = normal_lpdf( y[i] | gpmu[gp[i]] + beta * x[i], sigma_y);
    }
}
```

Ex.5: WAIC in Stan

- Code block to fit fixed effect model

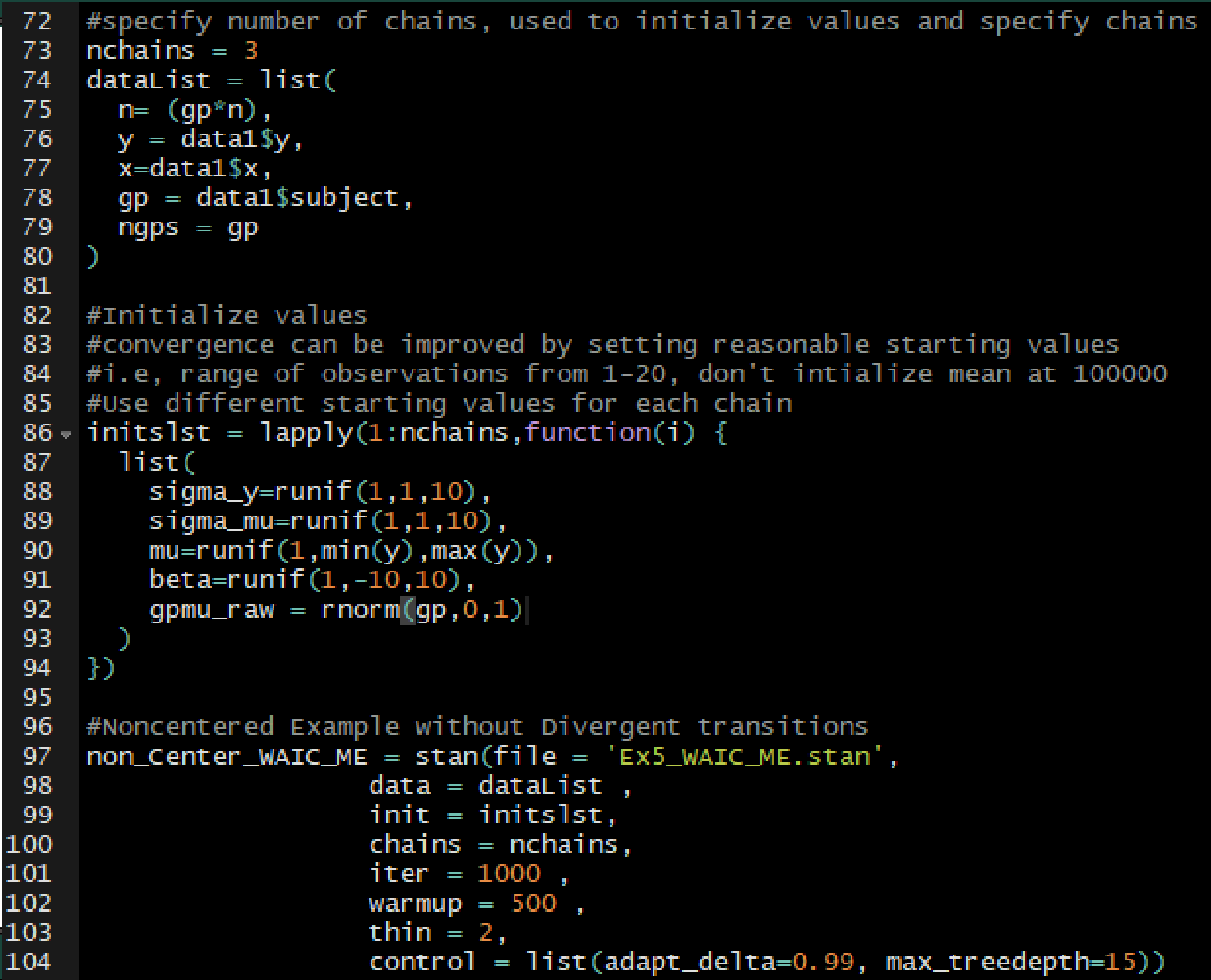


```r
#specify number of chains, used to initialize values and specify chains
nchains = 3

dataList = list(
  n= (gp*n),
  y = data1$y,
  x=data1$x,
  gp = data1$subject,
  ngps = gp
)
#Initialize values
#convergence can be improved by setting reasonable starting values
#i.e, range of observations from 1-20, don't intialize mean at 100000
#Use different starting values for each chain
initslst = lapply(1:nchains,function(i) {
  list(
    sigma=runif(1,1,10),
    alpha=runif(1,min(y),max(y)),
    beta=runif(1,-10,10)
  )
})

#Centered Example with Divergent Transitions
WAIC_FE = stan(file = 'Ex5_WAIC_LR.stan',
             data = dataList ,
             init = initslst,
             chains = nchains,
             iter = 1000 ,
             warmup = 500 ,
             thin = 2,
             control = list(adapt_delta=0.99, max_treedepth=15))
```

QFC Quantitative Fisheries Center

Ex.5: WAIC in Stan

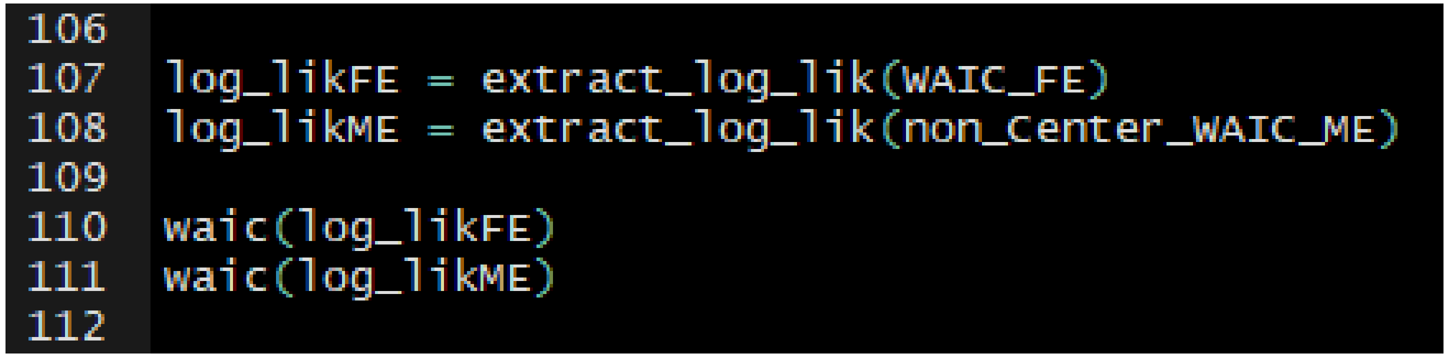
- Code block to fit mixed effect model



```
#specify number of chains, used to initialize values and specify chains
nchains = 3
dataList = list(
  n= (gp*n),
  y = data1$y,
  x=data1$x,
  gp = data1$subject,
  ngps = gp
)

#Initialize values
#convergence can be improved by setting reasonable starting values
#i.e, range of observations from 1-20, don't intialize mean at 100000
#Use different starting values for each chain
initslst = lapply(1:nchains,function(i) {
  list(
    sigma_y=runif(1,1,10),
    sigma_mu=runif(1,1,10),
    mu=runif(1,min(y),max(y)),
    beta=runif(1,-10,10),
    gpmu_raw = rnorm(gp,0,1)
  )
})

#Noncentered Example without Divergent transitions
non_Center_WAIC_ME = stan(file = 'Ex5_WAIC_ME.stan',
                  data = dataList ,
                  init = initslst,
                  chains = nchains,
                  iter = 1000 ,
                  warmup = 500 ,
                  thin = 2,
                  control = list(adapt_delta=0.99, max_treedepth=15))
```

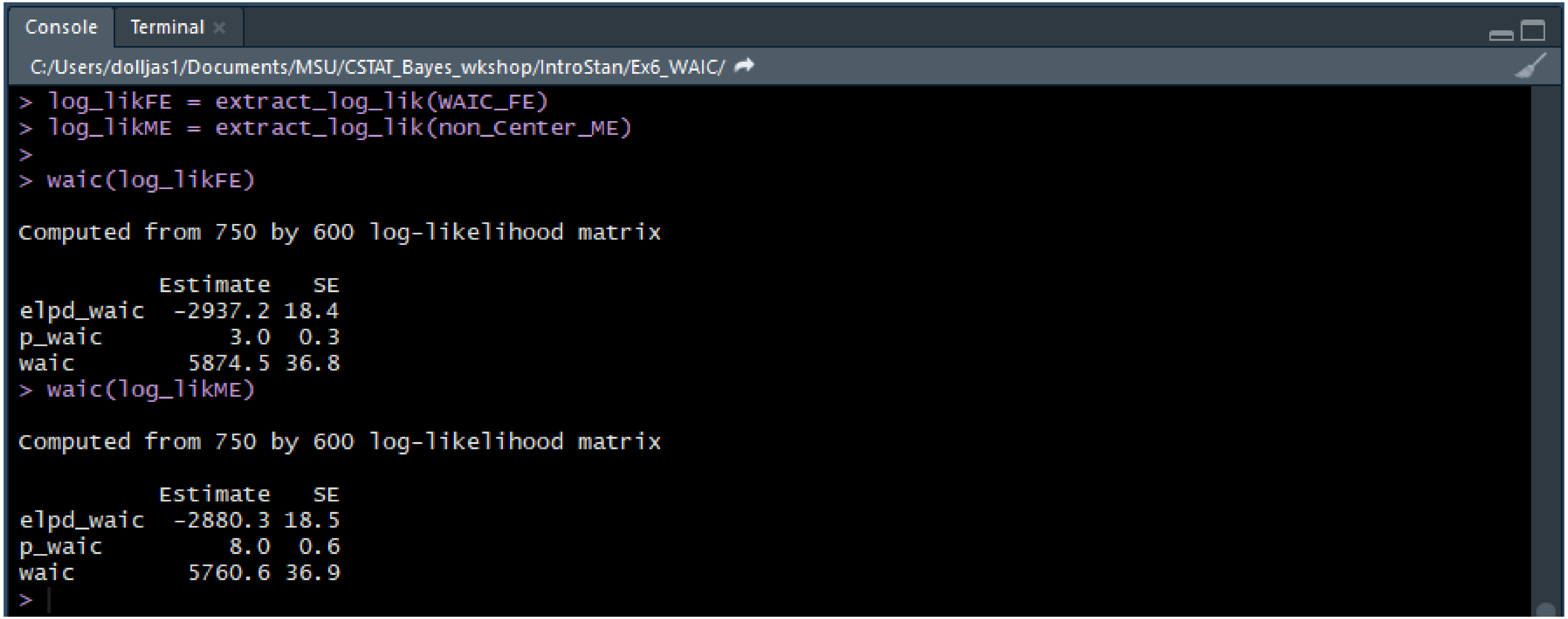
Ex.5: WAIC in Stan

- Extract log-likelihood
- Calculate WAIC



```

log_likFE = extract_log_lik(WAIC_FE)
log_likME = extract_log_lik(non_Center_WAIC_ME)

waic(log_likFE)
waic(log_likME)

```

Ex.5: WAIC in Stan


Console
Terminal
C:/Users/dolljas1/Documents/MSU/CSTAT_Bayes_wkshop/IntroStan/Ex6_WAIC/

```
> log_likFE = extract_log_lik(WAIC_FE)
> log_likME = extract_log_lik(non_Center_ME)
>
> waic(log_likFE)

Computed from 750 by 600 log-likelihood matrix

          Estimate   SE
elpd_waic  -2937.2 18.4
p_waic         3.0  0.3
waic        5874.5 36.8
> waic(log_likME)

Computed from 750 by 600 log-likelihood matrix

          Estimate   SE
elpd_waic  -2880.3 18.5
p_waic         8.0  0.6
waic        5760.6 36.9
> 
```

Questions?

- Can I create an executable file (.exe) and use later?
  - Not at this time, but they are working on it.
- Can I use Stan on the HPCC?
  - Yes, very easy. See me for details.
- Am I limited to the distributions supplied with Stan?
  - No, you can submit a request to Stan developers or program your own in C++
- Can Stan support sampling of discrete parameters? (e.g., mark-recapture)
  - No and Yes, discrete parameters must be marginalized out.

Thank you!

Exercise On Your Own

- Using the Iris dataset in base R
- Create the necessary Stan program to estimate parameters of a linear regression model to predict sepal width from sepal length

Exercise 6: Iris data analysis

Model: $Y_i = \alpha + \beta X_i + \varepsilon_i$

Y = sepal width

X = sepal length

$\varepsilon_i \sim \text{Normal}(0, \sigma)$

Priors: $\alpha \sim \text{Normal}(0,100)$

$\beta \sim \text{Normal}(0,100)$

$\sigma \sim \text{half} - \text{cauchy}(0,5)$

Exercise 6: Iris data analysis

- Open Ex5_est_lr.R
- Create a new text document, change to Stan, then save as Ex5_est_lr.stan

What blocks do you need?

- data
- parameter
- model