Using Hamiltonian Monte Carlo to Estimate Crop Response Functions with Stochastic Plateaus

John N. Ng'ombe and B. Wade Brorsen

John N. Ng'ombe (ngombe@okstate.edu) is a visiting assistant professor at Auburn University. B. Wade Brorsen (wade.brorsen@okstate.edu) is regents professor and A.J. and Susan Jacques Chair in the Department of Agricultural Economics at Oklahoma State University.

Selected presentation at the 2020 Southern Agricultural Economics Association Annual Meeting, February 1-4, 2020, Louisville, Kentucky

This research received funding from the Oklahoma Agricultural Experiment Station; the National Institute for Agriculture (NIFA) [Hatch project OKL02939]; and the A.J. and Susan Jacques Chair. The authors acknowledge helpful suggestions from Dayton Lambert.

Abstract
This paper seeks to make the latest in Bayesian methods easier for others to use with a focus on a stochastic plateau production function. A few studies have used Bayesian techniques to model a stochastic plateau function. No published work provides estimation of a stochastic plateau function by taking advantage of computationally efficient Hamiltonian Monte Carlo (HMC). HMC converges to high-dimensional target distributions much faster than the Gibbs Sampler and Metropolis algorithms. HMC is the default choice in Stan software, but the learning curve to master the software is high. This study aims to provide HMC estimation instructions of a univariate stochastic plateau function using **brms** and **RStan**. The programs are relatively compact and we include them within the paper for easy access. For our empirical example, we rely on experimental data from the Oklahoma Agricultural Experiment Station. Simulation results from HMC are consistent across different priors. As a robustness check, we compare empirical results from HMC and those obtained from maximum likelihood estimation. Empirical results from HMC are close to maximum likelihood parameter estimates obtained using SAS software. This paper serves as a tutorial for using HMC to estimate plateau-type production functions as well as other functions.

Introduction

Bayesian methods have received wide applications in the past decade thanks to the "Computational Revolution" (Basturk et al. 2013) since the 1990s. According to Lee and Wagenmakers (2014), and Jiang and Carter (2018), the number of publications with terms "Bayes" and/or "Bayesian" found on Google Scholar has from the early 2000s increased fivefold. When the search is extended to agricultural economics journals (e.g., *the American Journal of Agricultural Economics*, *Agricultural Economics*, *Journal of Agricultural Economics*, *Journal of Agricultural and Applied Economics*), dozens of papers that have used Bayesian technique. Adding to this evidence, the 2019 Agricultural and Applied Economics Association Annual (AAEA) annual meeting in Atlanta, GA, had a session titled "Bayesian Econometrics in Agricultural Economics." Taken together, this confirms proliferation of applied Bayesian econometrics in agricultural and resource economics (Bessler et al. 2010).

The learning curve required to learn Bayesian techniques limits their use. One example of where Bayesian methods have been used is in estimating the stochastic plateau crop-input response

function by Tembo et al. (2008). McFadden, Brorsen and Raun (2018), Ouédraogo and Brorsen (2018), and Ng'ombe (2019) use Bayesian techniques to model the stochastic plateau function to answer their research questions. McFadden, Brorsen, and Raun (2018) use Bayesian methods to combine prior information with plant sensing information assuming a stochastic plateau production function. Ouédraogo and Brorsen (2018) for winter wheat and Brorsen (2013) for cotton estimate stochastic plateaus with Bayesian methods. Ouédraogo and Brorsen consider nonnormal distributions to estimate the stochastic production function between wheat and N fertilizer. Ng'ombe (2019) uses a stochastic plateau production function with Monte Carlo data to determine optimal ways of conducting large-scale, on-farm field experiments assuming a stochastic plateau response function between corn and nitrogen (N).

Furthermore, studies by Ouédraogo and Brorsen (2018), and Ng'ombe (2019) do not take advantage of modern computationally efficient Markov chain simulation – Hamiltonian Monte Carlo (HMC). All these studies primarily use Metropolis-Hastings updates (Metropolis et al. 1953; Hastings 1970) and Gibbs sampling (Geman and Geman 1984; Gelfand and Smith 1990) in either SAS software (SAS Institute 2016) or JAGS (Plummer 2013). HMC and its extension, the no-U-turn sampler are the default choice in Stan software (Hoffman and Gelman 2014; Stan Development Team 2018). HMC employs "momentum" variables that fasten each iteration in a parameter space to allow faster mixing and convergence (Neal 2011; Gelman et al. 2013; Hoffman and Gelman 2014). Girolami and Caderhead (2011) show that both Gibbs sampling and Metropolis-Hastings (M-H) algorithms are generally less efficient than HMC. In addition, Gibbs sampling works more efficiently with conjugate priors (Gelman et al. 2014), which reduces the liberty of researchers in choosing priors that reflect their beliefs (Bürkner 2017a). In contrast, HMC and no-U-turn sampler algorithms converge much more quickly regardless of whether priors

are conjugate or not (Hoffman and Gelman 2014; Bürkner 2017a). Hoffman and Gelman (2014) suggest HMC much more quickly converges to high-dimensional target distributions than the Gibbs Sampler and M-H algorithm, which could speed up estimation of a stochastic plateau production function.

The learning curve to master Stan software, which does HMC, is rather high. So, the main goal of this study is to provide a tutorial about using HMC to estimate a stochastic plateau production function. We provide relatively compact programs showing how the stochastic plateau production function can be easily estimated via Stan. Stan software is an open source programming language for Bayesian inference and optimization. The Stan that we implement is done within the R programming language. The Stan software writes a program in C and calls a C++ compiler and thus avoids the operational overhead that slows many R programs.

The stochastic plateau response function

In agricultural economics, the stochastic plateau response function (Tembo et al. 2008) is a common response function to characterize crop-input relationships (see for example: Biermacher et al. 2009; Tumusiime et al. 2011; Boyer et al. 2012; Brorsen and Richter 2012; Boyer et al. 2013; Dhaka et al. 2019). The stochastic plateau response function is mathematically defined as

$$y_{it} = \min(\beta_0 + \beta_1 x_{it}, P + s_t) + u_t + \varepsilon_{it} \quad (1)$$

where y_{it} is the response variable (i.e., yield) from the *i*th plot in year *t*, β_0 and β_1 are model parameters to be estimated, x_{it} is the amount of the limiting input from the *i*th plot in year *t*, P is the average plateau yield, $s_t \sim N(0, \sigma_u^2)$ is the plateau year random effect, $u_t \sim N(0, \sigma_u^2)$ is the intercept year random effect, and $\varepsilon_{it} \sim N(0, \sigma_\varepsilon^2)$ is the random error term. The three random terms

in the model (s_t, u_t, and ε_{it}) are assumed to be independent of each other. Extending equation (1) to Bayesian inference implies that the model and plateau parameters are stochastic and therefore have prior distributions. By Bayes' rule, we can obtain model parameter's posterior distributions. The model in equation (1) can be specified in a Bayesian framework as

$$2) \qquad y_{it} = \min(\beta_0 + \beta_1 x_{it}, P_t + s_t) + u_t + \varepsilon_{it}$$

$$\beta_0 \sim N(\beta_{00}, \sigma_{\beta_0}^2), \quad \beta_1 \sim N(\beta_{11}, \sigma_{\beta_1}^2), \quad P_t \sim N(P_0, \sigma_{P_t}^2)$$

$$\sigma_u^2 \sim N(\gamma_u, \delta_u^2), \qquad \sigma_s^2 \sim N(\gamma_s, \delta_s^2), \qquad \sigma_\varepsilon^2 \sim N(\gamma_\varepsilon, \delta_\varepsilon^2)$$

where $\beta_{00}, \beta_{00}, P_0, \gamma_u, \gamma_s$, and γ_ε are prior means for respective parameters while $\sigma_{\beta_0}^2, \sigma_{\beta_0}^2, \sigma_{P_t}^2, \sigma_{\beta_u}^2$, $\sigma_{\beta_s}^2$, and $\sigma_{\beta_\varepsilon}^2$ are the variances of the priors.

Hamiltonian Monte Carlo and no-U-turn sampler

HMC algorithms substitute probability distributions assumed in a Markov chain with Hamiltonian dynamics to facilitate reaching the target distributions more efficiently. Stated differently, HMC extends the M-H procedure by providing proposal values that are more precise by using Hamiltonian dynamics (Jiang and Carter 2018). HMC shortens the time it takes to reach the posterior density space by ensuring that for every iteration, parameter values "leapfrog" to states closer to their posterior densities. This reduces the autocorrelation that plagues MCMC. After new values have been proposed, HMC uses M-H techniques to accept or reject them making HMC a more efficient type of Monte Carlo sampling (Neal 2011; Jiang and Carter 2018).

Following Jiang and Carter (2018), HMC algorithm originally has a trajectory length and step size. The step size indicates the size between possible solution points on a single leap trajectory. On the other hand, the trajectory length represents the length of a leap trajectory. But Hoffman and Gelman (2014) have shown that the performance of HMC largely depends on two parameters. For example, even if longer step sizes are likely to speed up computations, they may increase the number of rejections. In a similar manner, when the trajectory is short, HMC's random walks may be inefficient. Moreover, computation powers would be wasted when a trajectory is unnecessarily long. This is where the no-U-turn sampler comes in. According to Hoffman and Gelman (2014), the no-U-turn sampler reduces HMC's dependence on the two parameters by pre-establishing the sampling space and tuning the step size to a target acceptance rate rather than using a fixed step-size. In this manner, Bentacourt, Byrne and Girolami (2014) suggest a target acceptance rate of 0.8 to be optimal. The no-U-turn sampler uses a recursive tree-building technique to double the leap until when the U-turn when it comes across a computationally worthless exploration (Jiang and Carter 2018). Thus a trajectory length adapts only to a certain optimal range. More technical details about HMC can be found in Neal (2011), Hoffman and Gelman (2104) and Bentacourt, Byrne, and Girolami (2014).

Data

In our illustrations, we use both simulated and real data. For simulated data, we assume an agronomist who collects data on corn response to N from agronomic experiments conducted over a 10-year period. In each year, there are 100 plots resulting in 1000 data points for both corn and N. Equation (2) is the underlying data generating process.

For empirical analysis, we use cross-sectional time series data on wheat yield response to N. Data were obtained from the Oklahoma Agricultural Experiment Station in Oklahoma, U.S.A. The experiment (E502) was established in 1970 to determine effects of levels of N on winter wheat yield (in bushels) under conventional tillage. While phosphorus and potassium were also varied those plots are not used here. Regarding soil type, the soils are Grant silt loamy while wheat is seeded at 25.4 cm row spacing. In the regression, a single input response production function is used with N fertilizer being the only variable input. The experimental design used in the experiment is a randomized complete block design containing four replications and six N levels: 0, 22, 45, 67, 90, and 112 lbs. of N per acre. More information about the experiment can be found in Raun et al. (2010).

Stan Codes and Results

We demonstrate the use of HMC in Stan by using R and RStudio (RStudio Team 2015; R Core Team, 2018). R is used to call Stan and the latest versions of both are assumed to be installed on the computer. R can be installed from https://www.r-project.org/ while RStudio can be obtained from https://www.rstudio.com/. RStudio is not essential but we recommend it because of its ease at writing, editing, and running R code. Its particular feature is the capability to highlight Stan syntax (Lambert 2018). Another useful tool required to run the code is **Toolchain**. Stan depends on the computer's tools that handle C++ files because Stan code is translated into C++ code and then compiled. For ease of doing so, it is recommended that the **RTools** package is installed.

We assume that the reader uses Windows. The Stan website (https://mc-stan.org) has useful resources and outlines the procedure for installing these packages. We recommend similar resources to Linux users. Another required package used in this paper is **RStan**. **RStan** can be

installed within R using the `install.packages` function. But researchers may still waver to use Stan directly because every model written may have to be debugged or if possible optimized which could be time-consuming and an error-prone process despite their familiarity with Bayesian inference (Bürkner 2017a). Thus, we mostly use Bürkner's (2017a) package **brms** to make writing the Stan code easier. The **brms** package is user friendly and can estimate complex nonlinear models.

The following procedure is used to demonstrate our estimations: first we simulate some data that will be used to estimate equation (2). The R code to generate the data is in Listing 1.

{Listing 1}

We then use the **brms** package to recover the true parameters of the model. We do so by using two different prior distributions: normal and truncated Student *t*-distribution. The code to estimate the model using normal priors is presented in Listing 2.

{Listing 2}

The code in Listing 2 for Line 1 defines the stochastic plateau production function using the `bf` function, which is activated useful upon successful installation and loading of **brms**. The function `fmin` is equivalent to the min function in equation (2) while `b0` and `b1` represent β_0 and β_1 in equation (2). The variables *plate* and *intermain* in line 1 are respectively the plateau indexed by P_t and plateau year random effect (i.e., s_t in equation (2)). The formula *b1+b2~1* implies that both b1 and b2 are stochastic. In the case of *plate ~ 1+(1|TIME),* it means that the plateau intercept is stochastic and varies with TIME through the plateau year random effect s_t. The intercept year random effect is denoted as u_t in equation (2) and is shown in line 2 of Listing 2 as *intermain ~ 0+(1|TIME).* This is basically to tell Stan that the intercept year random effect is group-level and

varies with TIME. The option `nl = TRUE` tells Stan that the function being defined should be treated as non-linear.

Next, we set the priors for all stochastic parameters in the model. Following Bürkner (2017a) and Bürkner (2017b), **brms** requires the user to specify priors explicitly. For the code in Listing 2, all the priors are in line 4 through 8 defined by the `set_prior` function. We place a normal distribution with respective parameters on population-level effects (*b1, b2,* and *plate*), the group-level effects (*intermain*), and family-specific parameter (sigma). We assume a default prior for the plateau year random effect– a strictly positive truncated Student *t*-distribution with 3 degrees of freedom and scale parameter of 44 – out of preference. Using the fake data generated in Listing 1, the next step is to run the model using the `brm` function. The estimation code is presented in line 10 through 12 in Listing 2. The object containing priors, the model function previously defined and the data are specified in line 10 of Listing 2, while we specify control variables `adapt_delta` and `max_treedepth` to aid convergence. More details of these can be found in Burkner (2017a). Estimation results are saved in the object `fit`. The option `warmup` represents the burn-in phase while `iter` denotes the total MCMC samples. We use 4 cores and 4 chains to ensure that the chains are run in parallel cores. The results from our model are obtained using the `summary` function in R as shown in line 14 of Listing 2. Summary results of the fitted model are shown in table 1.

{table 1}

Results in table 1 indicate that we are able to recover the true parameter values pretty well. For example, the true values for population-level parameters are (40, 0.86, and 151.11) while the recovered parameters are respectively (41.33, 0.81, and 153.51). In terms of convergence, the Gelman-Rubin statistic (Gelman and Rubin 1992) (the $\widehat{R}$) is equal to 1 for all parameters indicating

successful convergence of the HMC chains. This result is consistent with trace plots of respective parameters shown in both panels of Figure 1.

{Figure 1}

All the trace plots indicate good mixing. While we imposed normal priors for model parameters of preceding results presented in table 1, we try to use a different prior for the next results as mentioned before. They are mainly weakly-informative priors. This is to check if the results would be robust to any prior change. We use truncated Student *t*-distribution priors with 4 degrees of freedom with a scale parameter of 10 on all the parameters. The **brms** code associated with these priors is presented in Listing 3.

{Listing 3}

The code in Listing 3 is syntactically similar to the one in Listing 2 except the section containing priors and thus we do not spend time detailing it. However, the priors shown are expected to be weakly informative and with heavier tails than the normal distribution. Results from the code in Listing 3 are presented in table 2. As can be observed in table 2, they are on average close to findings from table 2.

{table 2}

On average, we are able to recover true parameter values as in the first case implying that a change of priors in our example did not affect the novelty of our results. The trace plots from results in table 2 are shown in Figure 2. Both panels in Figure 2 indicate good mixing of the chains which suggests successful convergence of the chains to their target posterior distributions – a result that collaborates the Gelman-Rubin statistic which is equal to one for all parameter estimates.

{Figure 2}

As mentioned previously, we also estimate the stochastic plateau production function using empirical data. The **brms** code to estimate the stochastic plateau production function is presented in Listing 4. Like before, we impose weakly informative priors – the truncated Student *t*-distribution for all parameters. The rest remains the same as in Listing 3 except the data used.

{Listing 4}

Results from the code in Listing 4 are shown in table 3. The MCMC chains successfully converged based on the $\hat{R}$ values which are equal to one for model parameters. We find that an increase in N by an extra lb. per acre would result in an increase in wheat yield by 0.49 bushels per acre, when all other factors are held constant. The average plateau of wheat yield is about 44.47 bushels per acre.

{table 3}

We further try to estimate the same model in Stan using another useful package mentioned previously: **RStan.** The code that relies on **RStan** is presented in Listing 5. The code in Listing 5 differs from previous **brms** codes in that Listing 5 explicitly contains all Stan blocks. A block is simply a set of statements preceded by the block name and surrounded by statements (Sorensen, Hohenstein, and Vasishth 2016). Basic Stan blocks include: (1) **data** block, 2) **parameter** block, and (3) **model** block. Other blocks which are not mandatory are **functions**, **transformed data**, **transformed parameters**, and **generated quantities**. Based on Listing 5, the code is saved in an object named `model_2` (line 2) but can also be saved in a text file with an extension .stan for it to be executed. The **data** block as the name suggests is where data used to pass to Stan are declared. The number of observations, groups (i.e., years) and a variable that maps observations to groups are defined here. The word `int` refer to variables considered in integer form while `vector` implies the variable is a vector (it could be row or column vector).

{Listing 5}

The next block is **parameters** where all parameters to be used are defined. For example, `real plateau_error[M]` means that a plateau_error is a continuous variable that contains up to M elements. The variables named sigma and/or with some modifications are defined to range between 0 and 100 or 0 and 1000 are respective standard deviations. They range between the mentioned values to indicate that standard deviations are strictly positive. The next block is the **model** code block. This is where both the model's likelihood and priors for the parameters are specified. As shown in Listing 5, it is this section where parameters in the parameter block are assigned their prior distributions. For example, `plateau ~ normal(50, 100)` means that the wheat plateau follows a normal distribution with the mean of 50 bushels per acre and standard deviation of 100 bushels per acre. The response variable is defined from line 34 through 37 which is the stochastic plateau. The model's likelihood is defined in lines 38-41 of Listing 5, which completes the full code. The next codes in line 43 through 53 are R codes to define the dimensions of the variables for use in a data list. RStan recognizes data saved as a list as shown in line 54 of Listing 5. Next we execute the model using the `stan` function as shown in line 55. The arguments are almost similar to those used in **brms**. We then print the results of the parameters of interest by running the code in line 59. Results from this code are shown in table 4.

{table 4}

The results in table 4 are close to those in table 3 which suggests that we were able to demonstrate the estimation of the same empirical stochastic plateau production function for wheat in Stan using both **brms** and **RStan**. As a diagnostic check, we further estimate the model using maximum likelihood in SAS. We use the PROC NLMIXED procedure to do so and results are presented in table 5.

{table 5}

As shown in table 5, our maximum likelihood estimates are close to the posterior means obtained using **brms** and **RStan**. This suggests that our Bayesian computations illustrating estimation of the stochastic plateau in using HMC in Stan is working. The only difference is that SAS produces variances for random parameters when using maximum likelihood estimation (plateau year random effect, intercept year random effect and random error). In contrast, Stan reports standard deviations for the same parameters.

Conclusion

We showed how to estimate a stochastic plateau function using the Hamiltonian Monte Carlo (HMC) utilizing the no-U-turn sampler. We do so by using **brms** and **RStan** packages in R and RStudio. The intention of this study is to serve as a tutorial for estimation of the stochastic plateau production function (Tembo et al. 2008) for it has gained wide use in agricultural economics research.

Simulation results were consistent across different sets of priors suggesting that HMC yields robust results of the stochastic plateau model regardless of the type of priors used. As put forward by (Bürkner 2017a), HMC in Stan gives researchers the freedom to choose priors that may reflect their beliefs. Simulation results provided evidence that **brms** can both appropriately and accurately estimate the stochastic plateau production function with ease.

The findings from empirical data further show that the stochastic plateau model can be easily estimated in Stan by taking advantage of **brms** and **RStan**. Results from using both packages were consistent despite specifying different sets of priors in respective codes. A comparison of empirical results from Stan with those obtained by maximum likelihood estimation (obtained in SAS software) showed that they are very similar results. This result is an indication that the tutorial

provided in this paper is novel and would assist agricultural and applied economists interested in modeling production functions.

Being open source, Stan offers promising frontiers and we recommend its use for its ability to estimate complex nonlinear models. Even though we demonstrate it in R, Stan can be called from Matlab, Stata, Python, and Julia (Gelman, Lee, and Guo 2015). While we do not perform model selection tests and other avenues such as extending the univariate case to the multivariate case, we hope that this tutorial will give the reader a taste of fitting a Bayesian mixed effects model in a form of stochastic plateau production function and other functions.

Acknowledgments

The research was funded by the A. J. & Susan Jacques Chair and the Oklahoma Agricultural Experiment Station and USDA National Institute of Food and Agriculture, Hatch Project number OKL02939.

References

Basturk, N., C. Çakmaklı., S.P. Ceyhan., and H.K.V. Dijk. 2014. "On the Rise of Bayesian Econometrics after Cowles Foundation Monographs 10, 14." *Œconomia. History, Methodology, Philosophy* (4-3): 381-447.

Bessler, D. A., J.H. Dorfman., M.T. Holt., and J.T. LaFrance. 2010. "Econometric Developments in Agricultural and Resource Economics: The First 100 Years." *American Journal of Agricultural Economics* 92(2), 571-589.

Biermacher, J. T., B.W. Brorsen., F. M. Epplin, J.B. Solie., and W.R. Raun. 2009. "The Economic Potential of Precision Nitrogen Application with Wheat Based on Plant Sensing." *Agricultural Economics*, 40(4):397-407.

Boyer, C. N., Tyler, D. D., Roberts, R. K., English, B. C., and Larson, J. A. 2012. "Switchgrass Yield Response Functions and Profit-Maximizing Nitrogen Rates on Four Landscapes in Tennessee." *Agronomy Journal*, 104(6):1579-1588.

Boyer, C.N., J.A. Larson, R.K. Roberts, A.T. McClure, D.D. Tyler., and V. Zhou. 2013. "Stochastic Corn Yield Response Functions to Nitrogen for Corn after Corn, Corn after Cotton, and Corn after Soybeans." *Journal of Agricultural and Applied Economics* 45(1379-2016-113848):669.

Brorsen, B.W. 2013. "Using Bayesian Estimation and Decision Theory to Determine the Optimal Level of Nitrogen in Cotton." Selected paper, Southern Agricultural Economics Association annual meeting, Orlando, Florida, Feb. 3-5, 2013.

Brorsen, B.W., and F.G.C. Richter. 2012. "Experimental Designs for Estimating Plateau-Type Production Functions and Economically Optimal Input Levels." *Journal of Productivity Analysis* 38(1):45-52.

Bürkner, P. C. 2017a. "brms: An R Package for Bayesian Multilevel Models using Stan." *Journal of Statistical Software* 80(1), 1-28.

Bürkner, P. C. 2017b. "Advanced Bayesian Multilevel Modeling with the R package brms." arXiv preprint arXiv:1705.11123.

Dhakal, C., K. Lange., M.N. Parajulee., and E. Segarra. 2019. "Dynamic Optimization of Nitrogen in Plateau Cotton Yield Functions with Nitrogen Carryover Considerations." *Journal of Agricultural and Applied Economics* 1-17.

Gamerman, D., and H.F. Lopes. 2006. *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*. Chapman and Hall/CRC.

Girolami, M., and B. Calderhead. 2011. "Riemann Manifold Langevin and Hamiltonian Monte Carlo Methods." *Journal of the Royal Statistical Society: Series B* 73*:* 123–214.

Gelfand, A.E., and A.F. Smith. 1990. "Sampling-Based Approaches to Calculating Marginal Densities." *Journal of the American Statistical Association* 85(410):398-409.

Gelman, A., D. Lee., and J. Guo. 2015. "Stan: A Probabilistic Programming Language for Bayesian Inference and Optimization. *Journal of Educational and Behavioral Statistics* 40(5):530-543.

Gelman, A., and D.B. Rubin. 1992. "Inference from Iterative Simulation using Multiple Sequences." *Statistical Science,* 7(4), 457-472.

Gelman, A., H.S. Stern., J.B. Carlin., D.B. Dunson., A. Vehtari, D.B. Rubin. 2013. *Bayesian Data Analysis*, Third Ed. Chapman and Hall/CRC.

Geman, S., and D. Geman. 1984. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images." *IEEE Transactions on Pattern Analysis and Machine Intelligence* (6):721-741.

Hastings, W.K. 1970. “Monte Carlo Sampling Methods Using Markov Chains and their Applications." Biometrika 57(1):97-109

Hoffman, M. D., and A. Gelman. 2014. “The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo.” *Journal of Machine Learning Research* 15: 1593–1623.

Jiang, Z., and R. Carter. 2019. “Using Hamiltonian Monte Carlo to Estimate the Log-Linear Cognitive Diagnosis Model via Stan.” *Behavior Research Methods 51*(2), 651-662.

Lee, M. D., and E.J. Wagenmakers. 2014. *Bayesian Cognitive Modeling: A Practical Course*. New York, NY: Cambridge University Press.

Lunn, D. J., A. Thomas., N. Best., and D. Spiegelhalter. 2000. “WinBUGS—A Bayesian Modelling Framework: Concepts, Structure, and Extensibility.” *Statistics and Computing* 10: 325–337.

McFadden, B.R., B.W. Brorsen., and W.R, Raun. 2018. “Nitrogen Fertilizer Recommendations Based on Plant Sensing and Bayesian Updating.” *Precision Agriculture* 19(1):79-92.

Metropolis, N., A. W. Rosenbluth., M.N. Rosenbluth., A.H. Teller., and E. Teller. 1953. “Equation of State Calculations by Fast Computing Machines.” *The Journal of Chemical Physics* 21(6):1087-1092.

Neal, R. M. 2011. “MCMC using Hamiltonian Dynamics.” In S. Brooks (Ed.), Handbook of Markov Chain Monte Carlo (pp. 113–162). Boca Raton, FL: CRC Press/Taylor & Francis

Ng’ombe, J. N. 2019. “Economics of the Greenseeder Hand Planter, Discrete Choice modeling, and On-Farm Field Experimentation.” Ph.D. Dissertation. Oklahoma State University

Ouédraogo, F. and B.W. Brorsen. 2018. “Hierarchical Bayesian Estimation of a Stochastic Plateau Response Function: Determining Optimal Levels of Nitrogen Fertilization.” *Canadian Journal of Agricultural Economics* 66(1):87-102.

Plummer, M. 2003. “JAGS: A Program for Analysis of Bayesian Graphical Models using Gibbs Sampling.” In Proceedings of the 3rd International Workshop on Distributed Statistical Computing (Vol. 124, No. 125, p. 10).

R Core Team. 2018. "R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing." Vienna, Austria.

Raun, R. W., J. B. Solie, and M. L. Stone. 2010. “Independence of Yield Potential and Crop Nitrogen Response. *Precision Agriculture* 12: 508–18.

RStudio Team (2015). “RStudio: Integrated Development for R. RStudio, Inc., Boston, MA URL http://www.rstudio.com/.

SAS Institute Inc. 2016. SAS® 9.4 System Options: Reference, Fifth Edition. Cary, NC: SAS Institute Inc.

Stan Development Team. 2018. *Stan Modeling Language Users Guide and Reference Manual*, Version 2.18.0. http://mc-stan.org

Tembo, G., B.W. Brorsen, F.M. Epplin., and E. Tostão. 2008. “Crop Input Response Functions With Stochastic Plateaus.” *American Journal of Agricultural Economics* 90(2):424-434.

Tumusiime, E., B.W. Brorsen, J. Mosali, J. Johnson, J. Locke., and J.T, Biermacher. 2011. “Determining Optimal Levels of Nitrogen Fertilizer Using Random Parameter Models.” *Journal of Agricultural and Applied Economics* 43(1379-2016-113680):541.

Listing 1: Generation of Fake Data for the Stochastic Plateau Model

```
1         rm(list=ls())                          #Clear the console
2         set.seed(1)                            #set the seed for reproducibility
3         n=1000                                 #number of observations
4         nt<-100                                #number of plots
5         plterror <-  rnorm(nt, 0, 25)          #plateau year random effect
6         plate <- rep(150+plterror, nt)         #plateau
7         interror <-  rnorm(nt, 0, 10)          #error for intercept year random effect
8         intermain<-rep(0+interror, nt)         #intercept year random effect
9         corn     <-  numeric(n)                #empty object to store corn values
10        beta0 <-40;  beta1 <-0.86;             #true values for beta 0 and beta 1
11        NITROGEN<-runif(n, 0, 156)             #Nitrogen values drawn from uniform dist
12      for (k in 1:n){                          #Simulate corn values
13            rand_error<- rnorm(1, 0, 17 )#random error
14            corn0[k]<-min(beta0+beta1*NITROGEN[k], plate[k])+intermain[k]
              +rand_error
14                           }
15           summary(corn)                       #summary statistics for corn
16       #create a dataframe to store the variable TIME for each 100 observations
16    data1<-data.frame(y=corn0, x=NITROGEN, TIME=rep(1:10, each=100))
```

Listing 2: Stan Code for the Stochastic Plateau Production Function Using brms

```
1     function_1<- bf(y~fmin(b0+b1*x, plate)+intermain, b1+b2~1, plate~ 1+(1|TIME),
2        intermain~0+(1|TIME), nl = TRUE)              //function defining equation (2)
3        #priors for parameters are saved in the object priors
4   priors <-c(set_prior("normal(39,2.5)", nlpar="b1", class = "b"), //prior for beta
5            set_prior("normal(0.8, 2.5)", nlpar="b2", class = "b"),//prior for beta 2
6     set_prior("normal(150,10)", nlpar="plate",class = "b"),      //prior for plateau
7   set_prior("normal(0,5)", nlpar =  "intermain", class = "sd"), //plateau rand effect
8   set_prior("normal(0,17)", class = "sigma"))                 //prior for sigma
9   # run the model using brm function, model results will be saved in object fit
10     fit<-brm(function_1, prior = priors, data = data1,
11              control = list(adapt_delta=0.99, max_treedepth=15),
12                warmup = 2500, iter = 5000, cores = 4, chains = 4)
13     #summarize the results
14     summary(fit)
15          #plot of the model parameters
16          plot(fit, pars = c("sd", "b", "sigma"))
```

Table 1: Summaries of the Fitted Model Using `brms` and Normal Priors

Parameter	True Value	Posterior Mean	Standard Deviation	2.5%	97.5%	$\hat{R}$
$\hat{\beta}_0$	40.00	41.33	1.33	38.67	43.88	1.00
$\hat{\beta}_0$	0.86	0.81	0.02	0.78	0.85	1.00
$\hat{P}_t$	151.11	153.51	5.63	143.31	165.84	1.00
$\hat{\sigma}_s$	23.53	17.69	8.66	6.98	39.79	1.00
$\hat{\sigma}_u$	9.38	2.06	1.17	0.15	4.67	1.00
$\hat{\sigma}_\varepsilon$	17.68	20.90	0.47	19.99	21.84	1.00

Listing 3: Stan Code Using `brms` with Truncated Student *t*-Distributions

```
1     function_1<- bf(y~fmin(b0+b1*x, plate)+intermain, b1+b2~1, plate~ 1+(1|TIME),
2        intermain~0+(1|TIME), nl = TRUE)              //function defining equation (2)
3        #priors for parameters are saved in the object priors
4  priors <-c(set_prior("student_t(4,0,10)", nlpar="b1", class = "b"), //beta 0 prior
5        set_prior("student_t (4,0,10)", nlpar="b2", class = "b"), //beta 1 prior
6     set_prior("student_t(4,0,10)", nlpar="plate", class = "b"), //plateau prior
7 set_prior("student_t(4,0,10)",nlpar ="intermain",class = "sd"),//int rand effect prior
8 set_prior("student_t(4,0,10)",nlpar="plate", class = "sd"),//plate year rand eff prior
9 set_prior("student_t(4,0,10)", class = "sigma"))   //prior for sigma
9   # run the model using brm function, model results will be saved in object fit
10     fit<-brm(function_1, prior = priors, data = data1,
11              control = list(adapt_delta=0.99, max_treedepth=15),
12                warmup = 2500, iter = 5000, cores = 4, chains = 4)
13     #summarize the results
14     summary(fit)
15          #plot of the model parameters (not shown to conserve space)
16          plot(fit, pars = c("sd", "b", "sigma"))
```

Table 2: Summary Results Using Truncated Student *t*-Distribution Priors.

Parameter	True Value	Posterior Mean	Standard Deviation	2.5%	97.5%	$\hat{R}$
$\hat{\beta}_0$	40.00	41.95	1.60	38.77	45.07	1.00
$\hat{\beta}_0$	0.86	0.81	0.02	0.77	0.84	1.00
$\hat{P}_t$	151.11	155.43	9.40	140.69	178.07	1.00
$\hat{\sigma}_s$	23.53	19.91	10.92	7.42	48.16	1.00
$\hat{\sigma}_u$	9.38	2.16	1.24	0.18	4.99	1.00
$\hat{\sigma}_\varepsilon$	17.68	20.91	0.47	20.02	21.87	1.00

Listing 4: Stan Code using brms with Truncated Student *t*-Distribution for Empirical Data

```
1           #Empirical Data
2           setwd("C:/Users/njohn/Documents/brorsen")

3          lahoma<-read.csv(file = “lahoma.csv”)
4          #check names and missing values
5     model<-bf(BUAC~fmin(b1+b2*N, plate)+intermain, b1+b2~1,   //define equation (2)
6      plate~ 1+(1|YR),intermain~0+(1|YR),  nl=TRUE )
7 prior_model<-c(set_prior("student_t(4,0,10)",nlpar="b1",class = "b"),//beta 0 prior
8    set_prior("student_t(4,0,10)", nlpar="b2", class = "b"),   //beta 1 prior
9    set_prior("student_t(4,0,10)", nlpar="plate", class = "b"), //plateau prior
10 set_prior("student_t(4,0,10)",nlpar = "intermain",class= "sd"),//int ran eff prior
11 set_prior("student_t(4,0,10)", nlpar = "plate",class = "sd"),//plat rand eff prior
12   set_prior("student_t(4,0,10)", class = "sigma")) //sigma prior
13   fit<-brm(model, prior = prior_model,data = lahoma,  //fit the model in Stan
14      control = list(adapt_delta=0.99, max_treedepth=15), //to help convergence
15       warmup = 2500, iter = 5000, //burn-in phase and sampling
16      cores = 4, chains = 4, seed=1)   //cores and chains for parallel computation
17           #summarize the results
18    summary(fit)
19    plot(fit, pars = c("sd", "b", "sigma"))
```

Table 3: Summary Results for Wheat Response (bu/Acre) to Nitrogen (lb/acre) Using brms.

Parameter	Posterior Mean	Standard Deviation	2.5%	97.5%	$\hat{R}$
$\hat{\beta}_0$	25.11	1.46	22.20	27.94	1.00
$\hat{\beta}_0$	0.49	0.01	0.46	0.52	1.00
$\hat{P}_t$	44.47	2.63	39.34	49.73	1.00
$\hat{\sigma}_s$	14.80	1.71	11.94	18.60	1.00
$\hat{\sigma}_u$	9.57	1.08	7.74	11.96	1.00
$\hat{\sigma}_\varepsilon$	5.80	0.12	5.59	6.03	1.00

Listing 5: Stan Code to Estimate a Stochastic Plateau Model Using RStan for Empirical Data

```
1           #Empirical Data using the Rstan
2  model_2 <- '
3         data {
4            int N;                                 // number of obs
5            int M;                                 // number of groups (years)
6            int K;                                 // number of predictors
7            vector[N] y;                           // the response variable
8            row_vector[K] x[N];                    // predictors
9            int g[N];                              // map obs to groups
10             }
11          parameters {                            //define parameters
12              real alpha;                         //define beat 0 parameter
13              real a[M];                          //define mapping parameter
14              real plateau_error[M];              //define plateau random effect parameter
15              vector[K] beta;                     //define beta 1
16              real<lower=0,upper=100> plateau; //define standard deviations
17              real<lower=0,upper=1000> sigma;
18              real<lower=0,upper=1000> sigmae;
19              real<lower=0,upper=1000> sigma_plateau; //define plateau error
20              }
21          model {                                  //define the model
22            vector[N] response;                    //intermediate result variable
23            sigmae ~ gamma(2,.2);                  //weakly informative priors,
24                                                  //see section 6.9 in STAN user guide
25      sigma_plateau ~ gamma(2,.2);                //expected value is the product of the 1st
26                                                  //parameter times the inverse of the 2nd
27          sigma~gamma(2,.2);                      //RStan uses inverse scale
28          plateau ~ normal(50, 100);              //impose priors on define parameters
29          alpha ~ normal(0,100);
30          plateau_error~normal(0,1);
31          a ~ normal(0,1);
32          beta ~ normal(0,1);
33
34      for (j in 1:N) {
35          response[j]=fmin((alpha + x[j]*beta),
36                    plateau+sigma_plateau*plateau_error[g[j]]) +
36         sigma*a[g[j]];                           //define the stochastic plateau function
37                    }
38      for(n in 1:N) {
39        y[n] ~ normal(response[n],sigmae);   //RStan uses standard deviation
40       }
41      }
42      '
43      library(rstan)                             #load Rstan package
46      setwd("C:/Users/njohn/Documents/brorsen")
47      lahoma<-read.csv(file = “lahoma.csv”)
48      nobs <- nrow(lahoma)
49      years <- length(unique(lahoma[,"YR"]))
50      xx <- lahoma$N
51      gg <- group_indices(lahoma,YR)              #creates an index for year
52      y <- lahoma$BUAC
53      xx <- cbind(xx)                            #intercepts are added in the code
54      yield_data <- list(N=NROW(y),M=years, K=1,y=y,x=xx,g=gg)
55      fit <- stan(model_code=model_2, model_name="lrsp",
56             data=yield_data, iter=5000, warmup=2500, chains=4,
57            cores=4, seed = 1, control = list(adapt_delta = 0.80, max_treedepth = 15))
58      #print results for parameters of interest at 95% credible interval
59      print(fit, pars=c("alpha","beta[1]","sigma","sigma_plateau",
60        "sigmae","plateau"), probs=c(0.025, 0.975))
```

Table 4: Summary Results for Empirical Data Using RStan

Parameter	Posterior Mean	Standard Deviation	2.5%	97.5%	$\hat{R}$
$\hat{\beta}_0$	25.52	1.45	22.71	28.40	1.00
$\hat{\beta}_0$	0.49	0.01	0.46	0.52	1.00
$\hat{P}_t$	45.59	2.64	40.49	50.77	1.00
$\hat{\sigma}_s$	14.83	1.72	11.93	18.62	1.00
$\hat{\sigma}_u$	9.55	1.07	7.70	11.93	1.00
$\hat{\sigma}_\varepsilon$	5.80	0.12	5.58	6.05	1.00

Table 5: Maximum Likelihood Parameter Estimates for the Stochastic Plateau Function from SAS

Parameter	Estimate	Standard Error	t value
Dep Variable: Wheat (bu/acre)			
Intercept	25.14	0.74	33.69
Slope	0.49	0.01	34.99
Plateau	45.96	0.63	73.19
Plateau year random effect	201.51	31.24	6.45
Intercept year random effect	86.57	5.96	14.53
Random Error	33.43	1.36	24.60

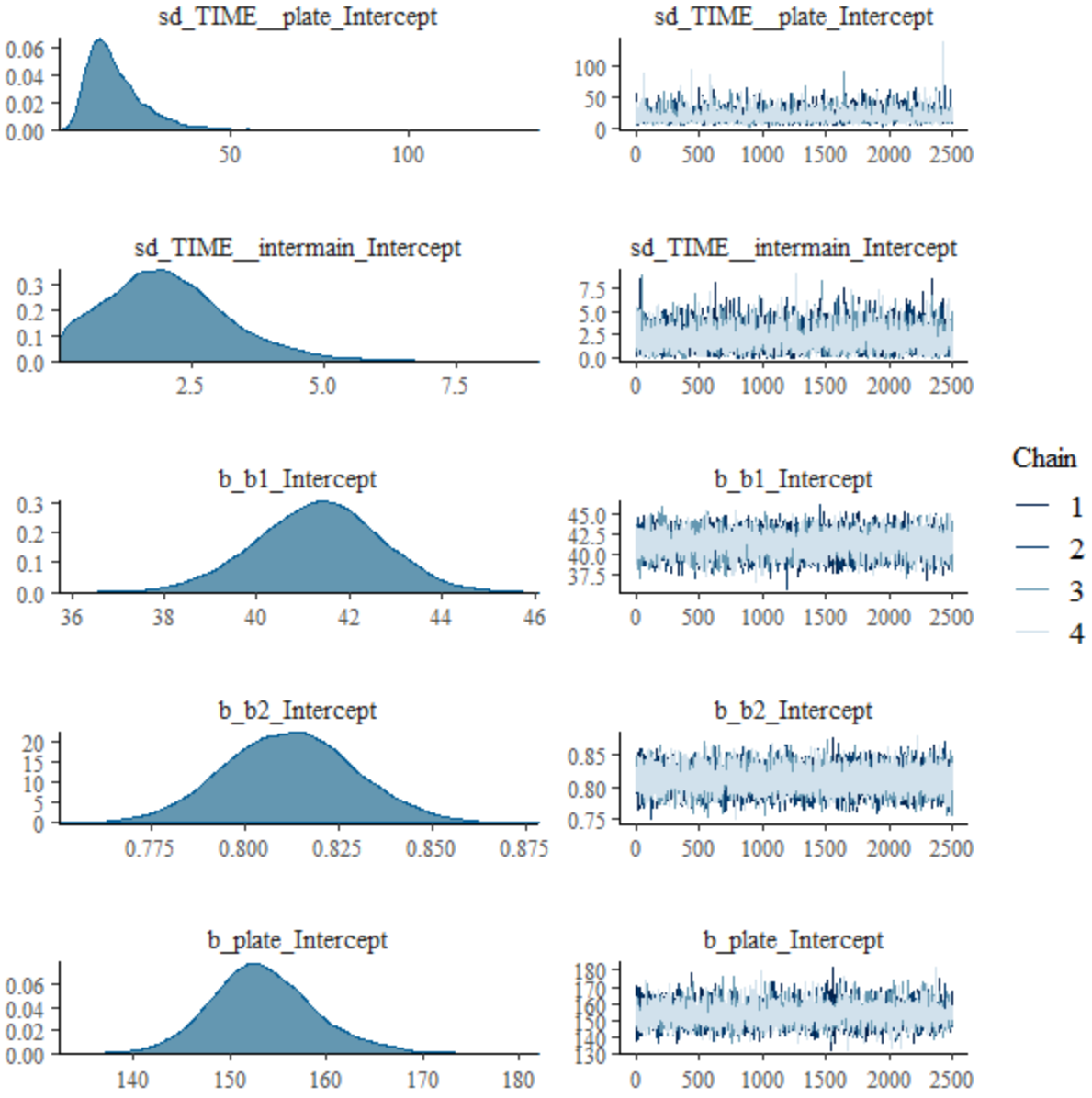

Figure 1: panel 1: Marginal densities and trace plots for stochastic plateau production function with normal priors

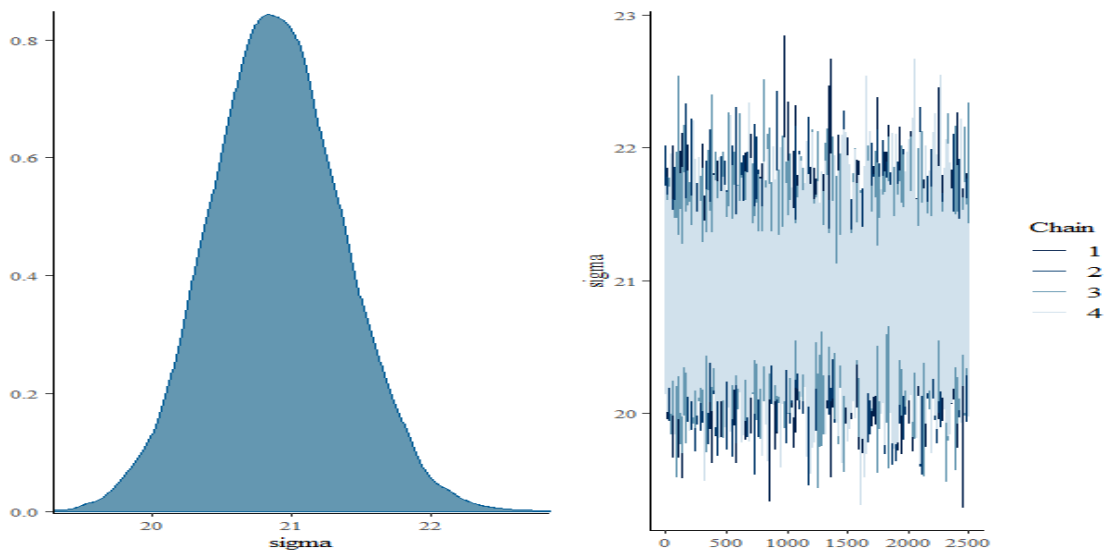

Figure 1: panel 2: Marginal density and trace plot for standard deviation of the stochastic plateau production function with a normal prior

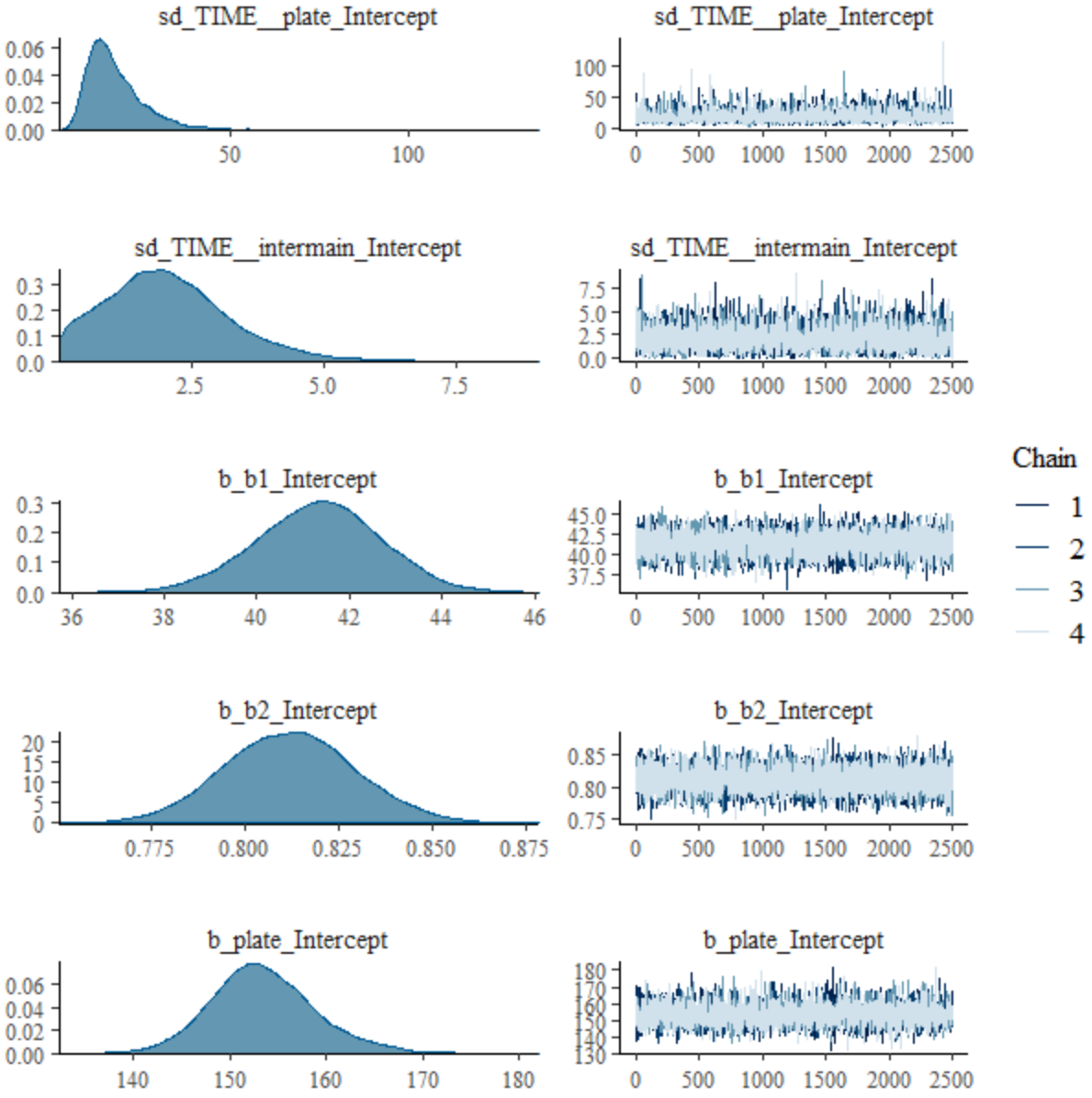

Figure 2: panel 1: Marginal densities and trace plots for stochastic plateau production function with truncated student t-distribution priors

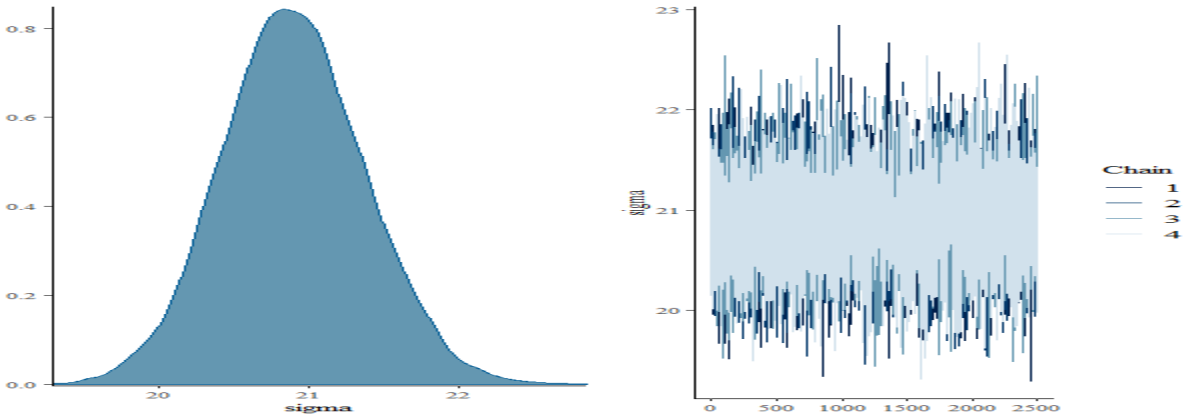

Figure 2: panel 2: Marginal density and trace plot for standard deviation of the stochastic plateau production function with truncated student t-distribution priors