A study of predictors for debugging quality among preservice, early childhood teachers

Brian R. Belland, ChanMin Kim, Eunseo Lee, Afaf Baabdullah, & Anna Y. Zhang

The Pennsylvania State University

Objectives

In this study, we provided debugging scaffolding to preservice, early childhood teachers as they engaged in debugging, and explored predictors of debugging quality from among the following data: motivation data, process data, and the number of words in and positive/negative valence of what learners typed in response to scaffolding prompts.

Theoretical Framework

Recently, there has been a push for the inclusion of computer science throughout the K-12 curriculum (Grover & Pea, 2013; K-12 Computer Science Framework Steering Committee, 2016). While there has been progress, there is a need to promote more equitable participation in computer science education and to improve teacher preparation (Blikstein, 2018). A learner population that is often overlooked when it comes to computer science education is that of early childhood learners, including infants, toddlers, preschoolers, kindergartners, and early elementary learners. While much work has addressed how to prepare secondary and primary teachers to teach computer science (Yadav, Mayfield, Zhou, Hambrusch, & Korb, 2014), there is a dearth of literature addressing how to prepare early childhood teachers to teach computer science. Within early childhood education, computer science is often taught by inviting children to control robots using block-based programming (Bers, 2010; Umaschi & Ettinger, 2012). Block-based programming is more intuitive and simpler to implement than text-based programming (e.g., python, C#), as it involves dragging and dropping blocks of code that are often represented by pictures. Still, as with text-based programming, bugs can and do occur, and resolving such can be challenging. Thus, a key challenge in preparing early childhood teachers to teach computer science is helping them learn to debug.

Debugging is a form of ill-structured problem solving (Fitzgerald et al., 2008; Jonassen, 2011; Vessey, 1985). Ill-structured problem solving ability can be enhanced by providing learners with scaffolding as they engage in problem solving (Jonassen, 2011; Reiser, 2004; Wood, Bruner, & Ross, 1976). Scaffolding can be defined as cognitive and motivational support that enlists student interest in problem solving, models expert processes, asks pertinent questions, provides feedback, highlights important problem features, and helps student control frustration (van de Pol, Volman, & Beishuizen, 2010; Wood et al., 1976). One way it can do so is by simplifying the problem elements that are not essential to learning the problem solving skill, but also problematizing those elements that are (Reiser, 2004). Scaffolding often takes the form of question prompts to which student need to type an answer, and then build on their answers when responding to further prompts.

Recent meta-analyses have shown that ill-structured problem-solving ability is enhanced more when learners receive scaffolding than when they receive lecture (Authors, 2017a, 2017b). Still, there can be variations in scaffolding's effectiveness based on such factors as the level with which students engage with scaffolding (Kim & Hannafin, 2011; Saye & Brush, 2002), achievement goal orientation (Duffy & Azevedo, 2015; Giesbers, Rienties, Tempelaar, & Gijselaers, 2013), and student motivation to engage with the subject and the problem being addressed (Giesbers et al., 2013). One can measure the level with which students engage with scaffolding by counting the number of words students type in response to scaffold prompts. Achievement goal orientation refers to what an individual wishes to accomplish within a learning task: mastery of the content and skills (mastery goal orientation), demonstration of competence, especially in comparison with others (performance-approach), and avoidance of challenging tasks to avoid appearing incompetent (performance-avoid) (Covington, 2000; Lehtinen, Vauras, Salonen, Olkinuora, & Kinnunen, 1995; Linnenbrink-Garcia et al., 2012; Pintrich, 2000). Motivation to engage with a subject and problem being addressed can also stem from one's domain identification, or sense of belonging and strong self-efficacy in a particular domain (e.g., English or science) (Thoman, Smith, Brown, Chase, & Lee, 2013).

Research Question

Do goal orientation, perceptions of STEM+CS, domain identification, debugging process quality, and the number of words in and positive/negative valence of what learners typed in response to scaffold prompts predict debugging quality?

Method

Note: Full elaboration of method details is not possible due to space constraints.

Setting and Participants

The study took place during three sessions of 2.5 hours each of a class on play in the early childhood curriculum in a large university in the eastern USA. Nineteen students (all female) consented to participate. Eighteen were early childhood education majors (4 seniors, 10 juniors, and 4 sophomores) who had all completed at least some field experience at the infant/toddler, preschool, kindergarten, or early elementary levels. The other was a non-major who has not completed any field experience. Two students were African American, while the rest were Caucasian.

The class covered play-based activities as educative processes in the early childhood classroom. The foundational assumption was that play is ideally child-directed rather than teacher-directed. Accordingly, students were encouraged to think of ways to integrate robots within early childhood instruction, but to leave room for children to direct their own play with robots.

Materials

Ozobot Evos were used, and they were controlled using Ozoblockly, a block-based coding program.

Scaffolding design was informed by clustering an expanded coding dataset from a meta-analysis (reference removed for blind review) in which we identified scaffolding features of clusters in which scaffolding was used at the university or graduate level and in computer science or engineering field, and for which effect size was medium or high. We used the results of this analysis, in addition to a literature review on learning debugging, to protype the scaffolding. The scaffolding invited students to identify the blocks used within the buggy code, the order in which the blocks were used, and to create and test hypotheses about the nature of the bug and how to can be fixed.

Data Collection

Presurvey. The presurvey was designed to assess participants':

- Perceptions of
    - science, technology, engineering, mathematics, computer science, and English
    - STEM and computer science careers
- Mathematics and English self-efficacy
- Goal orientation

Debugging process quality. The debugging process rubric assessed the quality with which each team (a) identified the bug, (b) located the bug, and (c) fixed the bug (see Table 1). Two raters applied the rubric independently to what each group articulated in response to scaffold prompts while completing two debugging challenges. Interrater reliability before coming to consensus was 0.954, based on the intraclass correlation coefficient metric. The raters met to come to consensus on scores, and consensus scores were used in the analysis.

Debugging quality. The debugging quality rubric assessed the extent to which participants found the location of the bug and fixed it (see Table 2). Two raters applied the rubric independently to the final code participants created at the end of the two debugging challenges. Interrater reliability before coming to consensus was 0.878, as measured by the intraclass correlation coefficient. The raters met to come to consensus on scores, and consensus scores were used in the analysis.

Sentiment analysis. The SentimentAnalysis package for r (Feuerriegel & Proellochs, 2019) was used to determine the degree to which what participants wrote in response to debugging scaffolding and when reflecting on the debugging process was positive or negative. Possible scores ranged from -1 to 1.

Analysis

We used a Bayesian multiple linear regression approach to predict debugging quality because, with small samples, results from frequentist statistical methods can often lack precision due to limitations in applying the central limit theorem (McNeish, 2017). In the Bayesian approach to statistics, one sets up a probability model through a prior distribution and then updates it using observed data to obtain the posterior distribution; the end result is a credible interval that represents a distribution of the parameter of interest (Little, 2006; Smith & Gelfand, 1992). This is because in the Bayesian approach, data is assumed to be fixed and the parameters are assumed to be random; to the contrary, in the frequentist approach (the most commonly used statistical approach in educational research), data is assumed to be random and parameters are assumed to be fixed (McNeish, 2017).

First, we used Bayesian automatic stepwise model selection to determine the combination of predictor variables that led to the ideal model fit. We started with 12 predictors. Next, two predictors were removed due to collinearity problems. Next, we established a prior distribution of inverse gamma for sigma^2 (variance) and uniform prior for Beta values. We then ran Markov Chain Monte Carlo Simulations (10,000 iterations) to determine the posterior distribution. Analysis was conducted using the Bayes.lm function in r.

Results

Convergence Diagnosis

As can be seen in Figure 1, there was convergence and normality of the MCMC simulated samples. Furthermore, it indicates that there were no issues of autocorrelation.

Do goal orientation, perceptions of STEM+CS, domain identification, debugging process quality, and the number of words in and positive/negative valence of what learners typed in response to scaffold prompts predict debugging quality?

The final Bayesian linear regression model (See Table 3) was:

Debugging quality = 0.554 * debugging process total score + 0.395 * English domain identification score + 0.588 * Performance approach goal orientation – 0.554 * sentiment analysis total score – 61.441.

Bayesian regression indicates that the variables included in this model predict debugging quality. The Beta for debugging process total score was 0.554. This means that for each increase of one point in debugging process total score, one can expect an increase in debugging quality score of 0.554 points. The Beta for English domain identification was 0.395. This means for each increase of one point in English domain identification, one can expect an increase of 0.394 points in debugging quality score (though cation is warranted as the credible interval for this Beta includes zero). The Beta for performance approach goals was 0.588. This means that for each one point move in the direction of performance approach goals, one can expect an increase in debugging quality score of 0.588 points. The Beta for sentiment analysis was -0.554. This means that for each increase in sentiment analysis of 1 point, one can expect a decrease in debugging quality of 0.554 points.

Scientific Significance

Due to space limitations, discussion of results is abbreviated, but will be expanded in the full paper.

One of the most intriguing results in this study was that sentiment analysis was associated with a sizable negative Beta. This in essence means that, as what people write in response to scaffold prompts demonstrates a more negative valence, debugging quality improves. It is important to note that sentiment analysis is performed by a computer with no human intervention, and that when the computer deems writing to be more negatively valenced, that does not necessarily mean that the author is writing negatively worded posts such as “I hate this scaffolding.” Rather, a response to scaffold prompt of “I did X, but I should have done Y,” would likely be interpreted as negatively valenced. However, that is precisely the type of response that would lead to better debugging, because it evidences critical reflection on action, and thoughts on how to improve.

The second most intriguing result was that English domain identification was a positive predictor of debugging quality. The literature shows that often students identify with a particular domain to the exclusion of other domains, and this can lead to poor performance in the latter domains (Thoman et al., 2013).

That performance approach goals are positive and sizable predictors of debugging quality is not surprising given the literature that shows that such goals can be quite adaptive (Elliot & Harackiewicz, 1996). And the finding that debugging process quality predicts debugging quality is to be expected. Still, it is helpful to determine the Betas for the predictors, as this can lead to knowledge of how to optimally support preservice teachers learning to debug.

Limitations

As is often the case in educational research, sample size was small. This can lead to erroneous conclusions. Using a Bayesian approach can partially assuage such concerns, but non-informative priors can become informative with small sample size, which can be problematic (McNeish, 2017). Still, using a Bayesian approach allows one to make probabilistic statements about population parameters, which can enhance generative potential.

Acknowledgments

This research was supported by grants 1906059 and 1927595 from the National Science Foundation. Any opinions, findings, and/or conclusions are those of the authors and do not necessarily represent official positions of NSF.

References

Bers, M. U. (2010). The tangibleK robotics program: Applied computational thinking for young children. *Early Childhood Research & Practice*, *12*(2).

Blikstein, P. (2018). *Pre-college computer science education: A survey of the field*. Mountain View, CA: Google LLC. Retrieved from https://goo.gl/gmS1Vm.

Covington, M. V. (2000). Goal theory, motivation, and school achievement: An integrative review. *Annual Review of Psychology*, *51*(1), 171–200. doi:10.1146/annurev.psych.51.1.171

Duffy, M. C., & Azevedo, R. (2015). Motivation matters: Interactions between achievement goals and agent scaffolding for self-regulated learning within an intelligent tutoring system. *Computers in Human Behavior*, *52*, 338–348. doi:10.1016/j.chb.2015.05.041

Elliot, A. J., & Harackiewicz, J. M. (1996). Approach and avoidance achievement goals and intrinsic motivation: A mediational analysis. *Journal of Personality and Social Psychology*, *70*(3), 461–475. doi:http://dx.doi.org/10.1037/0022-3514.70.3.461

Feuerriegel, S., & Proellochs, N. (2019). Retrieved from https://cran.r-project.org/web/packages/SentimentAnalysis/vignettes/SentimentAnalysis.html

Fitzgerald, S., Lewandowski, G., McCauley, R., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: Finding, fixing and flailing, a multi-institutional study of novice debuggers. *Computer Science Education*, *18*(2), 93–116. doi:10.1080/08993400802114508

Giesbers, B., Rienties, B., Tempelaar, D., & Gijselaers, W. (2013). Investigating the relations between motivation, tool use, participation, and performance in an e-learning course using web-videoconferencing. *Computers in Human Behavior*, *29*, 285–292. doi:10.1016/j.chb.2012.09.005

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher*, *42*(1), 38–43. doi:10.3102/0013189X12463051

Jonassen, D. H. (2011). *Learning to solve problems: A handbook for designing problem-solving learning environments*. New York, NY, USA: Routledge.

K-12 Computer Science Framework Steering Committee. (2016). *K–12 computer science framework*. Retrieved from http://www.k12cs.org

Kim, M., & Hannafin, M. (2011). Scaffolding 6th graders' problem solving in technology-enhanced science classrooms: A qualitative case study. *Instructional Science*, *39*, 255–282. doi:10.1007/s11251-010-9127-4

Lehtinen, E., Vauras, M., Salonen, P., Olkinuora, E., & Kinnunen, R. (1995). Long-term development of learning activity: Motivational, cognitive, and social interaction. *Educational Psychologist*, *30*, 21–35. doi:10.1207/s15326985ep3001_3

Linnenbrink-Garcia, L., Middleton, M., Ciani, K., Easter, M., O'Keefe, P., & Zusho, A. (2012). The strength of the relation between performance-approach and performance-avoidance goal orientations: Theoretical, methodological, and instructional implications. *Educational Psychologist*, *47*, 281–301. doi:10.1080/00461520.2012.722515

Little, R. J. (2006). Calibrated Bayes: A Bayes/frequentist roadmap. *American Statistician*, *60*, 213–223. doi:10.1198/000313006X117837

McNeish, D. M. (2017). Challenging conventional wisdom for multivariate statistical models with small samples. *Review of Educational Research*, 003465431772727. doi:10.3102/0034654317727727

Pintrich, P. R. (2000). Multiple goals, multiple pathways: The role of goal orientation in learning and achievement. *Journal of Educational Psychology*, *92*, 544–555. doi:10.1037/0022-0663.92.3.544

Reiser, B. (2004). Scaffolding complex learning: The mechanisms of structuring and problematizing student work. *Journal of the Learning Sciences*, *13*, 273–304. doi:10.1207/s15327809jls1303_2

Saye, J., & Brush, T. (2002). Scaffolding critical reasoning about history and social issues in multimedia-supported learning environments. *Educational Technology Research and Development*, *50*(3), 77–96. doi:10.1007/BF02505026

Smith, A. F. M., & Gelfand, A. E. (1992). Bayesian statistics without tears: A sampling-resampling perspective. *The American Statistician*, *46*(2), 84–88. doi:10.2307/2684170

Thoman, D. B., Smith, J. L., Brown, E. R., Chase, J., & Lee, J. Y. K. (2013). Beyond performance: A motivational experiences model of stereotype threat. *Educational Psychology Review*, *25*(2), 211–243. doi:10.1007/s10648-013-9219-1

Umaschi, M., & Ettinger, A. B. (2012). Programming robots in kindergarten to express identity: An ethnographic analysis. In B. S. Barker, G. Nugent, N. Grandgenett, & V. I. Adamchuk, *Robots in K-12 education a new technology for learning* (pp. 168–184). Information Science Reference.

van de Pol, J., Volman, M., & Beishuizen, J. (2010). Scaffolding in teacher–student interaction: A decade of research. *Educational Psychology Review*, *22*, 271–296. doi:10.1007/s10648-010-9127-6

Vessey, I. (1985). Expertise in debugging computer systems: A process analysis. *International Journal of Man-Machine Studies*, *23*(5), 459–494. doi:10.1016/S0020-7373(85)80054-7

Wood, D., Bruner, J., & Ross, G. (1976). The role of tutoring in problem solving. *Journal of Child Psychology and Psychiatry*, *17*, 89–100. doi:10.1111/j.1469-7610.1976.tb00381.x

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education*, *14*(1).

Table 1. Debugging process quality rubric.

Debugging processes	Debugging subprocesses	1	2	3	4
Identifying the bug	Reviewing the program (code)	Did not review the code (no listing of the blocks)		Reviewed the code but the list of blocks is incomplete	Reviewed the code and the list of blocks is complete
	Running the program to evaluate the output	Did not run the program			Ran the program
	Describing the discrepancy between program goal and program output	Did not describe the discrepancy between program goal and program output	Described the discrepancy between program goal and program output but incorrectly	Described the discrepancy between program goal and program output correctly, but in a vague manner	Described the discrepancy between program goal and program output correctly and clearly
	Generating a hypothesis for a bug causing the discrepancy	Did not generate a hypothesis	Generated a hypothesis that was unrelated to the discrepancy	Generated a hypothesis that was partially related to the discrepancy	Generated a hypothesis that was fully related to the discrepancy
Locating the bug	Reviewing the structure of the program to investigate the probable location of the bug	No attempt to locate the bug	Line-by-line review of the program was mentioned without talking about the structure of the program	Chunk-by-chunk review of the program was mentioned without paying attention to the structure of the program	The structure of the program was mentioned when reporting the review of the program
	Using cues while examining the program	No cue was mentioned	Cues were mentioned but they were unrelated to the bug	Cues were mentioned but they were partially related to the bug	Cues were mentioned and they were related to the bug
	Locating the bug	No attempt to locate the bug	Attempted but failed to locate the bug	The bug was partially located; (e.g., succeeded in locating the block where the bug was but failed to locate the exact code in the block causing the error)	The bug was correctly located
Fixing the bug	Examining program goal to determine how to fix the bug	Did not mention the program goal	NA	NA	Mentioned program goal

	Revising the program	No attempt to revise the program	Revised the program incorrectly	NA	Revised the program correctly
	Reevaluating the program after revision	No attempt to reevaluate	Failed numerous trial-and-error attempts	Success after numerous trial-and-error attempts	Success without trial-and-error attempt
	Concluding	The bug was not fixed	The bug was fixed but not in a structurally correct manner but the desired output was produced	The bug was fixed in a structurally correct manner but without the desired output	The bug was fixed in a structurally correct manner with the desired output

Table 2. Debugging quality.

	0	3	5	7	10	Score deduction
Bug A	Failed to find the buggy block	(a) Found the buggy block but not the exact location requiring change(s) in value and/or sequence and (b) failed to fix	(a) Found the buggy block and the exact location requiring change(s) in value and/or sequence but (b) failed to fix	(a) Found the buggy block and the exact location requiring change(s) in value and/or sequence but (b) fixed the bug only partially	Fixed the bug completely	Score deduction two points deduction (-2) for each unnecessary code change*
Bug B	Failed to find the bug block	(a) Found the buggy block but not the exact location requiring change(s) in value and/or sequence and (b) failed to fix	(a) Found the buggy block and the exact location requiring change(s) in value and/or sequence but (b) failed to fix	(a) Found the buggy block and the exact location requiring change(s) in value and/or sequence but (b) fixed the bug only partially	Fixed the bug completely	

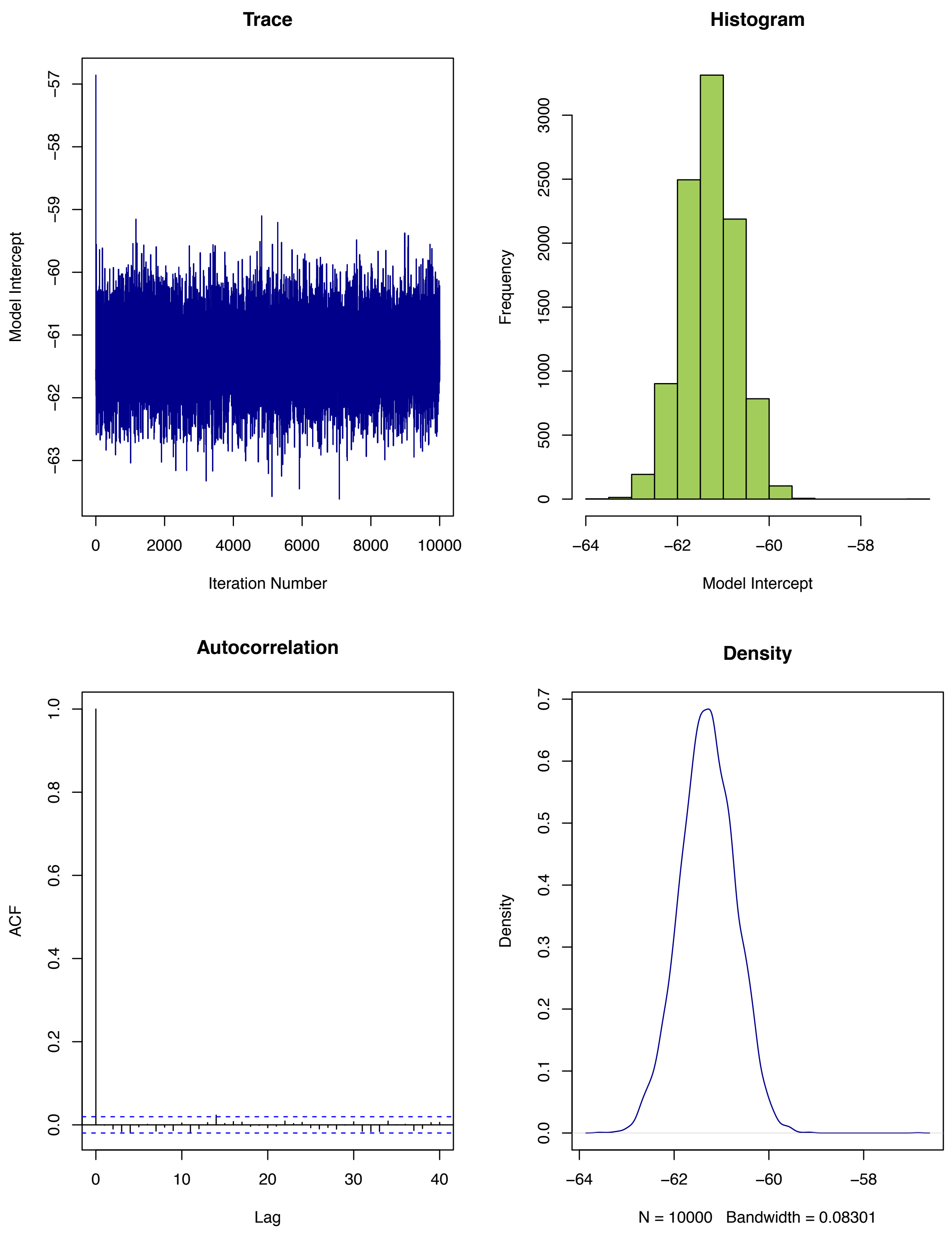
Table 3. Bayesian multiple regression model

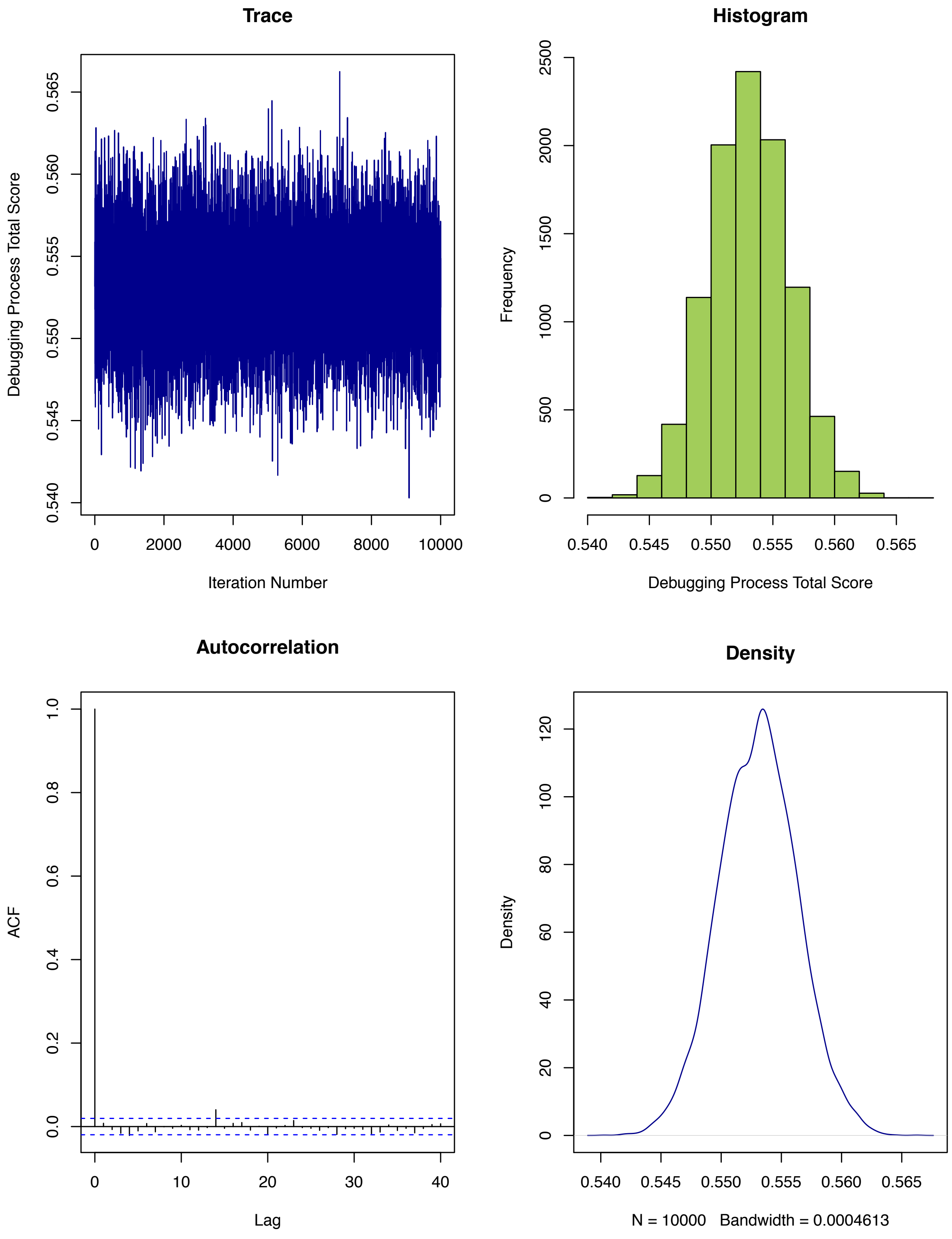
Coefficients	Estimate	Lower limit 95% credible interval	Upper limit 95% credible interval	Naïve standard error
(Intercept)	-61.445	-94.293	-27.68	0.167

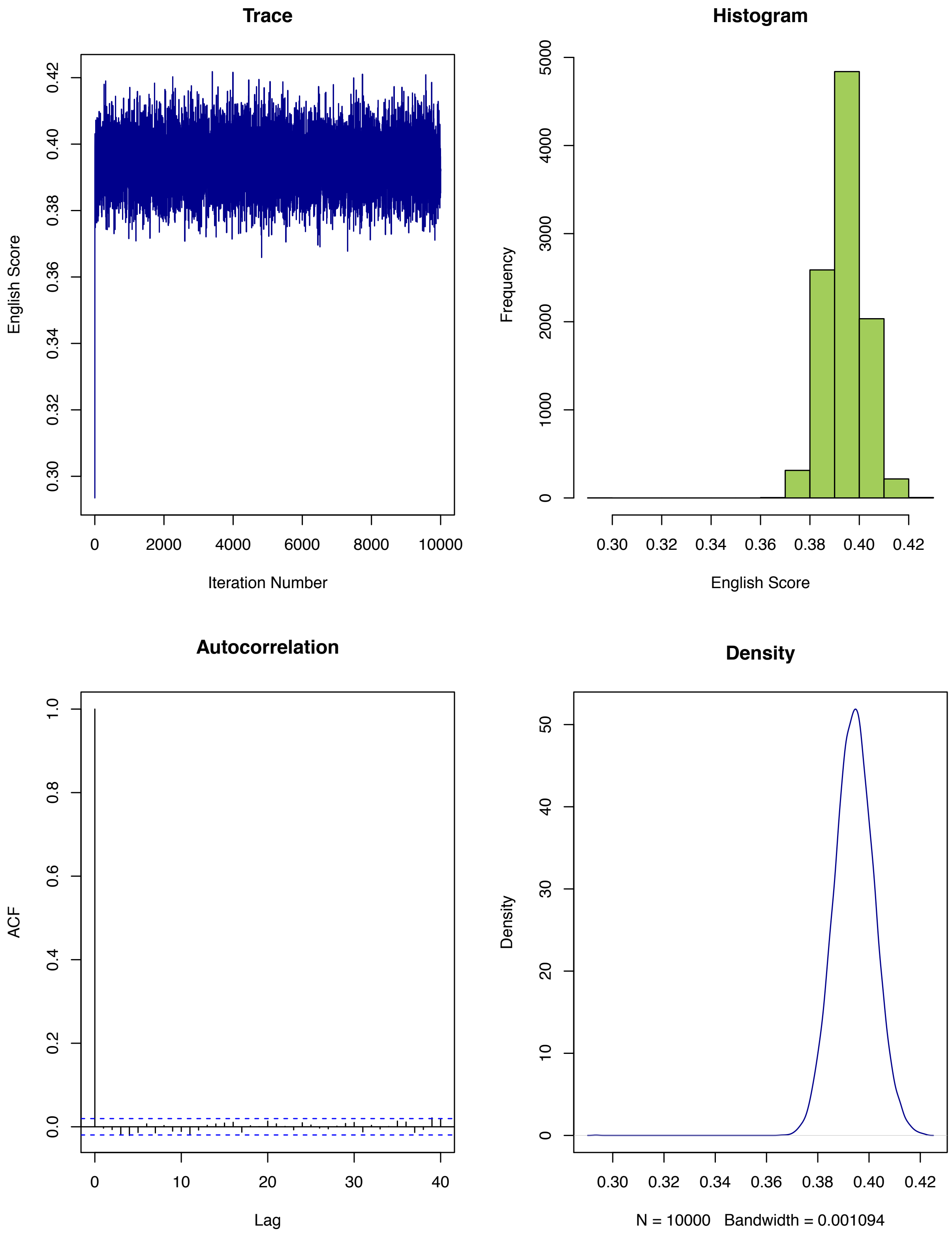
Debugging process total score	0.554	0.369	0.738	0.001
English domain identification	0.395	-0.05	0.828	0.002
Performance approach goals	0.588	0.172	1.009	0.002
Sentiment analysis	-0.554	-1.023	-0.077	0.002

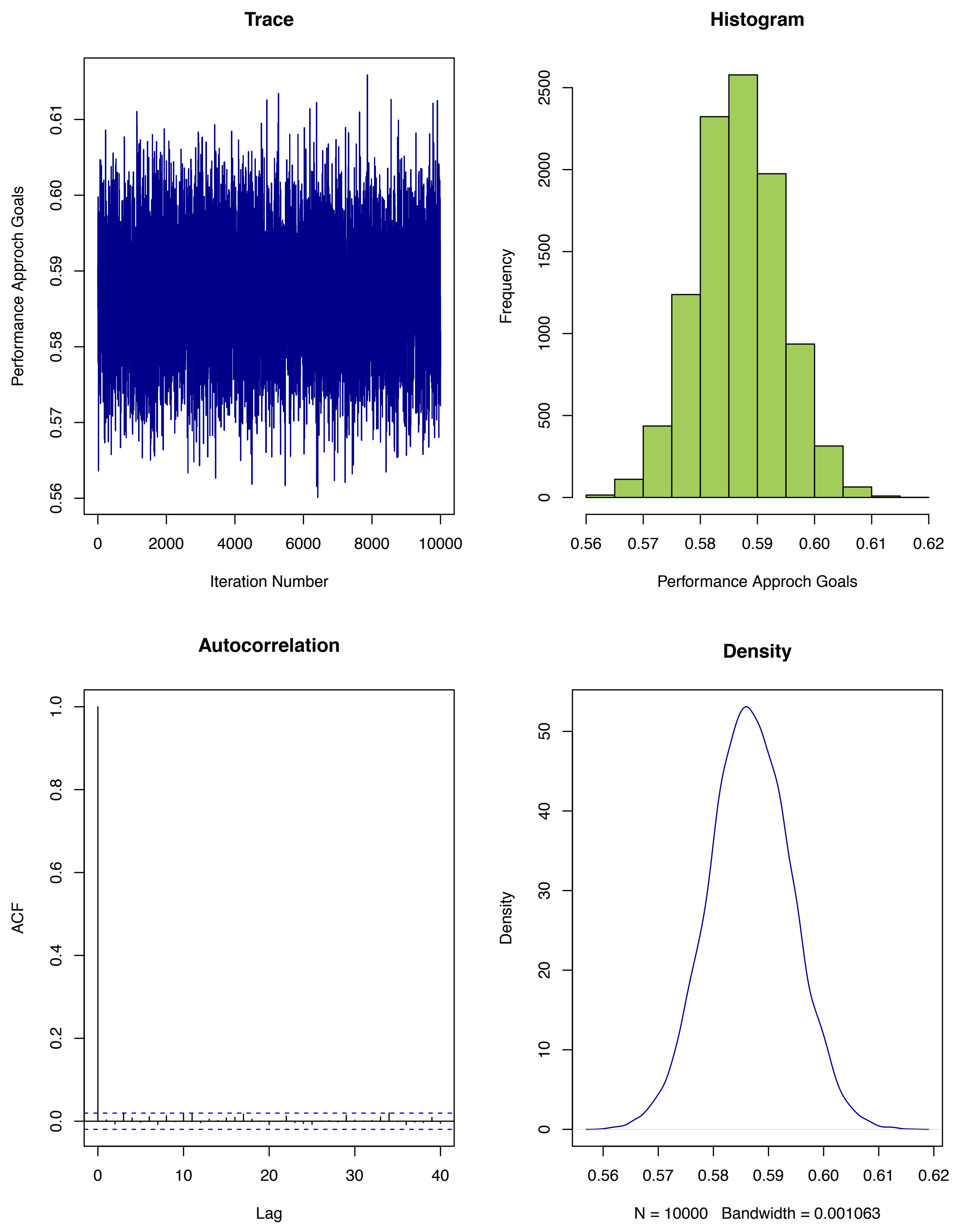
Note: MCMC iterations = 10,000

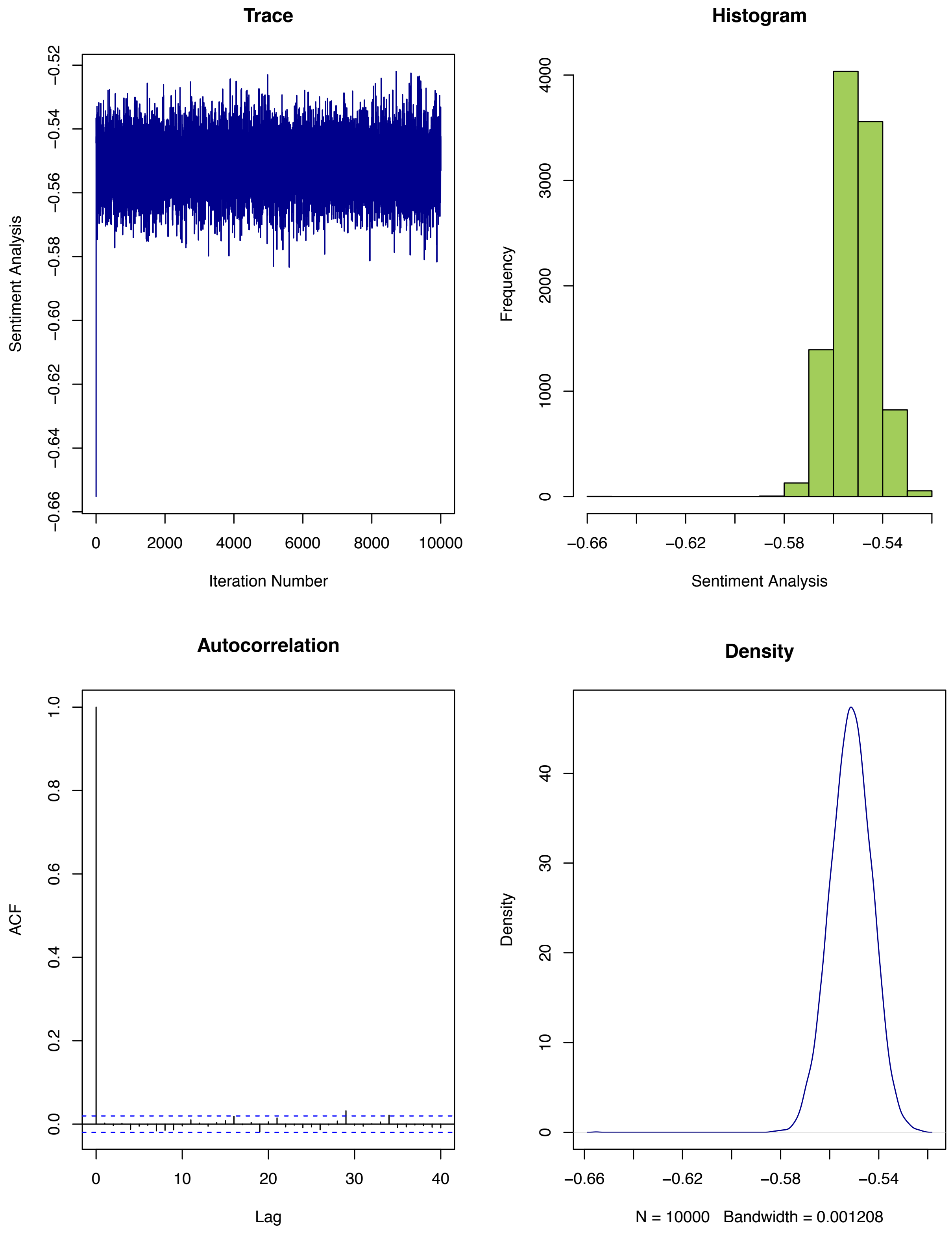
Figure 1. Trace plots.


Trace
Model Intercept
Iteration Number
0
2000
4000
6000
8000
10000
−57
−58
−59
−60
−61
−62
−63
Histogram
Frequency
Model Intercept
0
500
1000
1500
2000
2500
3000
−64
−62
−60
−58
Autocorrelation
ACF
Lag
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
Density
Density
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
−64
−62
−60
−58
N = 10000 Bandwidth = 0.08301


Trace
Debugging Process Total Score
0.540
0.545
0.550
0.555
0.560
0.565
0
2000
4000
6000
8000
10000
Iteration Number
Histogram
Frequency
0
500
1000
1500
2000
2500
0.540
0.545
0.550
0.555
0.560
0.565
Debugging Process Total Score
Autocorrelation
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
Lag
Density
Density
0
20
40
60
80
100
120
0.540
0.545
0.550
0.555
0.560
0.565
N = 10000   Bandwidth = 0.0004613


Trace
English Score
Iteration Number
Histogram
Frequency
English Score
Autocorrelation
ACF
Lag
Density
Density
N = 10000 Bandwidth = 0.001094


Trace
Performance Approch Goals
0.56
0.57
0.58
0.59
0.60
0.61
0
2000
4000
6000
8000
10000
Iteration Number
Histogram
Frequency
0
500
1000
1500
2000
2500
0.56
0.57
0.58
0.59
0.60
0.61
0.62
Performance Approch Goals
Autocorrelation
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
Lag
Density
Density
0
10
20
30
40
50
0.56
0.57
0.58
0.59
0.60
0.61
0.62
N = 10000 Bandwidth = 0.001063


Trace
Sentiment Analysis
Iteration Number
0
2000
4000
6000
8000
10000
−0.52
−0.54
−0.56
−0.58
−0.60
−0.62
−0.64
−0.66
Histogram
Frequency
Sentiment Analysis
0
1000
2000
3000
4000
−0.66
−0.62
−0.58
−0.54
Autocorrelation
ACF
Lag
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
Density
Density
0
10
20
30
40
−0.66
−0.62
−0.58
−0.54
N = 10000 Bandwidth = 0.001208