STAT 431 Final Project: The Distribution of Average Playtimes of Steam PC Games by Genre and Subgenre

Zihao Chen (zihaoc6) Afnan Dzaharudin (afnanfd2) Dominick Paolercio (djp6)
Hao Quan (haoquan2)

Introduction

Steam is one of the largest and well-known video-game platforms developed by Valve Corporation, primarily focusing on gaming on personal computers (PCs). Our interest is in analyzing the average playtime (minutes) - and the possible differences thereof - of certain categories of video games on Steam by using Bayesian hierarchical modelling and inference through Monte Carlo Markov Chains (MCMC).

Data

Source

- Galyonkin, S. (2015). *All the Data and Stats About Steam Games*. SteamSpy. Retrieved November 27, 2022, from https://steamspy.com/

We use the **steamspy.com** API to obtain data about individual Steam and combine them into a dataset. SteamSpy derives its data from the Steam API provided by Valve Corporation. Because the Steam API only allows users to access the data of Steam users who have manually set their profile information to be public, certain numerical information about games may not be 100% accurate, but we believe it is sufficient to make an interesting analysis.

The API provides data in JSON format, so we use the `rjson` library to parse the data and `tidyverse` tools to do some cleaning. Our derived dataset is viewable in **this repository**.

Dataset Contents

Below are *some* of the columns associated with a row, or a game. In bold are the columns of data we are using for Bayesian hierarchical modelling.

Column Name	Type	Description
appid	Numerical	Unique ID of the game on the Steam platform.
name	String	Title name of the game.
owners	String	The predicted *range* of the number of owners of the game.
avg_forever	**Numerical**	**The average total playtime, in minutes, of players of the game, measured since March 2009 or later time of release.**
ccu	Numerical	The peak of concurrent users of the game on the day the data was collected.
genre	**String**	**A comma-separated list of genres that describe the game**
tags	**String**	**A high-variety comma-separated list of tags that further and more specifically describe the game**

Processing for Modelling

We have a pre-established set of 'main genres' that tend to have many deviations in style and game design, and 'sub-genres' of those main genres that narrows them more. We then create a subset of our main dataset that filters only for games of our main genres and subgenres. At the subgenre level, games are mutually exclusive; for example, we do exclude Shooters that are both Co-Op *and* a Battle Royale. As for duplicate games that fit more than one main genre, we also remove those from the data entirely.

Below is our table of pre-established main genres, their subgenres, and the final count of filtered games that fit each of these categories.

Main Genre	Subgenre	Count
1: Shooter	1: Co-Op	21
1: Shooter	2: Singleplayer	164
1: Shooter	3: Battle Royale	5
2: RPG	1: JRPG	9
2: RPG	2: Massively Multiplayer	52
2: RPG	3: Hack and Slash	34
3: Strategy	1: RTS	53
3: Strategy	2: Trading Card	7
3: Strategy	3: Tower Defense	9
4: Simulation	1: Agriculture	4
4: Simulation	2: Automobile	7
4: Simulation	3: Colony Sim	4
5: Sports	1: Racing	6
5: Sports	2: Soccer	8
5: Sports	3: Basketball	4
6: Action	1: Roguelike	17
6: Action	2: Superhero	5
6: Action	3: Fighting	23

We take these to a 3-dimensional array that store each game's `average_forever` value for its respective `i` and `j` indexing.

Modelling

Model Definition

We assume the average playtime of a game to be normally distributed. The mean of this distribution is dependent on the genre of that game, and is further influenced by an additional subgenre the game is part of.

Our hierarchical model is as follows:

$$
\begin{aligned}
Y_{ijk} \mid \alpha_{1i}, \alpha_{2ij}, \tau^2 &\sim \text{Normal}\left(\alpha_{1i} + \alpha_{2ij}, \frac{1}{\tau^2}\right) \\
\\
\rightarrow \alpha_{1i} \mid \mu_{\alpha_1}, \sigma_{\alpha_1} &\sim \text{Normal}\left(\mu_{\alpha_1}, \sigma^2_{\alpha_1}\right) \\
\mu_{\alpha_1} &\sim \text{Normal}(0, 0.001) \\
\sigma_{\alpha_1} &\sim \text{Exponential}(0.001) \\
\\
\rightarrow \alpha_{2ij} \mid \mu_{\alpha_2}, \sigma_{\alpha_2} &\sim \text{Normal}\left(\mu_{\alpha_2}, \sigma^2_{\alpha_2}\right) \\
\mu_{\alpha_2} &\sim \text{Normal}(0, 0.001) \\
\sigma_{\alpha_2} &\sim \text{Exponential}(0.001) \\
\\
\rightarrow \tau^2 &\sim \text{Gamma}(0.001, 0.001)
\end{aligned}
$$

α_{1i} represents the influence of being part of a broader genre on the mean average playtime of a game. α_{2ij} represents the influence of being part of a subgenre of that broad genre on the mean average playtime of a game. We allow both of these αs to vary based on prior parameters. τ^2 is the precision (or inverse of variance) of the average playtime of video games in general. We use uninformative priors for all the next levels.

We *have* tried some other models. Modelling on `ccu` failed, whether or not it was log transformed, as the modelled variance was too high and means tended to be close to 0. We also tried modelling on `log(average_forever)`, but the model tended to fail to converge when using overly-large or overly-small initial values for the chains. This model has given us satisfactory results.

Chains

We use five chains for this model, using prior values that we think are appropriate to represent being (1) extremely low, (2) somewhat low, (3) middling, (4) somewhat high, and (5) extremely high.

The initial values for each chains are as follows:

Chain Number	τ^2	μ_{α_1}	μ_{α_2}	σ_{α_1}	σ_{α_1}
1	1000	$0.01 \times \min(Y)$	$0.01 \times \min(Y)$	0.01	0.01
2	100	$\min(Y)$	$\min(Y)$	0.1	0.1
3	1	$\text{mean}(Y)$	$\text{mean}(Y)$	1	1
4	0.01	$\max(Y)$	$\max(Y)$	100	100
5	0.001	$100 \times \max(Y)$	$100 \times \max(Y)$	1000	1000

Convergence Diagnostics

We found that convergence does occur, but rather slowly. For this report we perform burn-in for 10,000 iterations. Internally, we monitored the α values for convergence. On this paper, to limit the number of plots done, we will present the convergence diagnostics for the sum of those αs, which is $\mu_{Y,ij} = \alpha_{1i} + \alpha_{2ij}$. We will also present the convergence of $\sigma_y = \frac{1}{\sqrt{\tau^2}}$.

Below are our methods for checking convergence, and the related plots are available in the appendix.

Trace Plots

A "fuzzy worm" shape is visible for all the μ_Y values as well as the σ_y values. This implies convergence for both sets of variables.

Autocorrelation Plots

The autocorrelation for all monitored variables converge to 0, further implying convergence.

Gelman Diagnostic Plots

The shrink factor for all monitored variables, for all chains, converge to 1, again implying convergence.

Conclusion

We can safely conclude convergence for our hierachical model and proceed to take further samples from it.

Monte Carlo Error

We take 150,000 samples for this report. We use these samples to take the variance of each parameter, and calculate the Monte Carlo error using the following formula:

$$\text{Monte Carlo error of } \theta = \sqrt{\frac{\text{Sample variance of } \theta}{N_{\text{simulations}}}}$$

The Monte Carlo errors for each monitored parameter in the sample are as such:

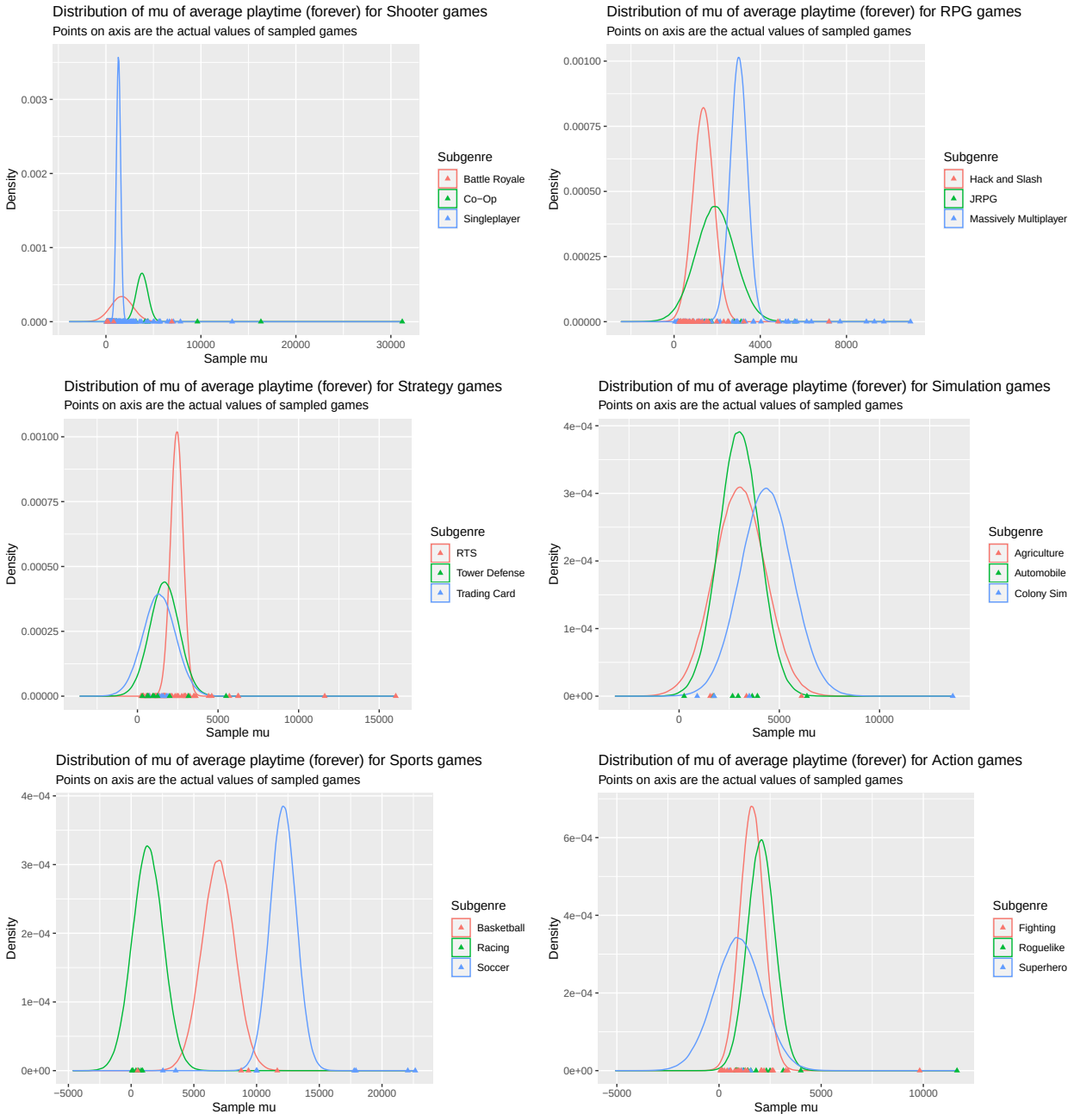
	Monte Carlo Error
mu_y[1,1]	1.5846
mu_y[2,1]	2.3378
mu_y[3,1]	1.0086
mu_y[4,1]	3.3443
mu_y[5,1]	3.1565
mu_y[6,1]	1.7442
mu_y[1,2]	0.5764
mu_y[2,2]	1.0174
mu_y[3,2]	2.6173
mu_y[4,2]	2.6300
mu_y[5,2]	2.6744
mu_y[6,2]	3.0283
mu_y[1,3]	3.0288
mu_y[2,3]	1.2533
mu_y[3,3]	2.3397
mu_y[4,3]	3.3502
mu_y[5,3]	3.3709
mu_y[6,3]	1.5117
sigma_y	0.2585

We find these values to be considerably low (especially σ_y), which we interpret to mean our estimations for these parameters are likely to be very accurate and close to the true values.

Findings

Final Distributions

μ_ys for each subgenres of a genre The following are the plot of the distributions of samples of μ_y for each level and sublevel, as well as the location of the actual Y data used in our model as reference for fit.



Our interpretations are as follows (numbers may be slightly off from the appendix summary table due to randomness):

For shooter games, Singleplayer games have the tightest spread of all measured games (possibly due to having the most data points associated), while Co-Op games tend to have the highest μ of average playtime in its genre.

Subgenre	Approx. μ of average playtime	Approx. SD of μ
Co-Op	3796 minutes	613 minutes
Singleplayer	1323 minutes	223 minutes
Battle Royale	1658 minutes	1175 minutes

For role-playing games (RPGs), Massively Multiplayer RPGs have a significantly higher μ of average playtime than the other two subgenres, while Japanese RPGs have a larger spread than the other two.

Subgenre	Approx. μ of average playtime	Approx. SD of μ
Japanese RPG (JRPG)	1910 minutes	906 minutes
Massively Multiplayer	3005 minutes	395 minutes
Hack & Slash	1360 minutes	485 minutes

For strategy games, Real-Time Strategies have the largest μ of average playtime and also varies the most; Trading Card and Tower Defense strategy games have a similar spread and close mean of μ.

Subgenre	Approx. μ of average playtime	Approx. SD of μ
Real Time Strategy (RTS)	2448 minutes	391 minutes
Trading Card	1351 minutes	1016 minutes
Tower Defense	1675 minutes	905 minutes

For simulation games, all three subgenres have relatively similar amounts spread. The μ of average playtime for Agriculture and Automobile simulation games tend similarly around about 3000 minutes, while Colony Simulations have a higher μ centered on 4363 minutes.

Subgenre	Approx. μ of average playtime	Approx. SD of μ
Agriculture	3018 minutes	1295 minutes
Automobile	2999 minutes	1019 minutes
Colony Simulation	4363 minutes	1298 minutes

For sports games, uniquely, all three subgenres have wildly different μs of average playtime. However, their spreads are all quite similar.

Subgenre	Approx. μ of average playtime	Approx. SD of μ
Racing	1313 minutes	1223 minutes
Soccer	12135 minutes	1037 minutes
Basketball	6958 minutes	1310 minutes

For action games, superhero games have the least μ of average playtime and varies the most as well. Roguelikes and Fighting games have similar spread, with Roguelikes having a relatively larger mean.

Subgenre	Approx. μ of average playtime	Approx. SD of μ
Roguelike	2060 minutes	676 minutes
Superhero	937 minutes	1173 minutes
Fighting	1611 minutes	585 minutes

Conclusion

Overall, within a genre, the average playtime of subgenres tend to have μ that are usually quite close to each other (with the distinct exception of Sports games), and that μ usually does not vary beyond 1200 minutes. Since μ is a function of the genre and subgenre of a genre (represented as α_{1i} and α_{2ij} in the hierarchical model), we can conclude that `(1)` the average playtime of a game is indeed influenced by the genre its in, and `(2)` which subgenre its in further influences that average playtime, both in where it's centered and how much it is spread.

Were to we have the time to further this study, we could have considered more genres and subgenres in Steam games for this model to observe more patterns in games. We also could have improved our filtering methods so that less games would have been entirely excluded due to genre overlap. Having repeated samples by using multiple datasets acquired over multiple weekends (when extracted data is richest) would also increase the number of data at our disposal. Were we able to perform a sensitivity analysis in time, it would help us be more confident in our choice of priors. So would comparing this model with other means of modelling the data, whether it be frequentist or Bayesian.

Contributions

- Afnan Dzaharudin (afnanfd2): Acquiring and processing the dataset, making visualizations on the results, and writing the report.
- Zihao Chen (zihaoc6): Early discussion on the ideas for the project, final selection of the form of the hierarchical model, and the selection of subgenres and genres to use for the model.
- Dominick Paolercio (djp6): Report proofreading & editing, discussion of the form of the hierarchical model, and the selection of subgenres and genres to use for the model.
- Hao Quan (haoquan2): Researching additional alternative methods of performing this study as a means of comparison with the Bayesian Hierarchical Model method, as well as sensitivity analysis.

Appendix

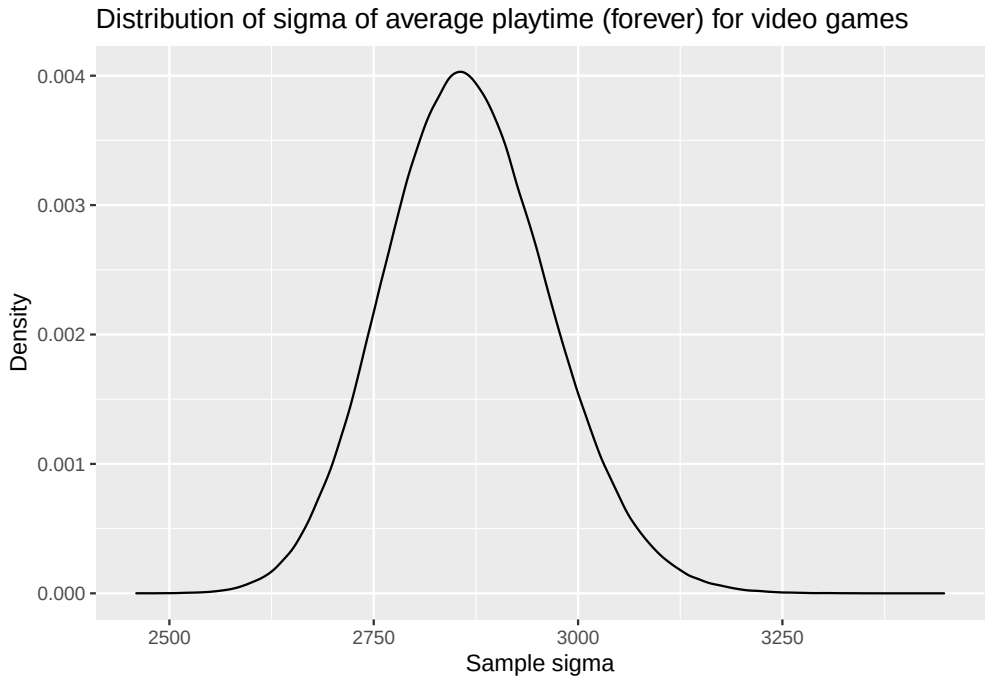
BUGS Model for the Project

```
  data {
    n_I = 6
    n_J = 3
    # n_observations is a 2d array
    # which stores the counts of each subgenre (j) of a genre (i)
  }

  model {
  for (i in 1:n_I) {
    for (j in 1:n_J) {
      for (k in 1:n_observations[i, j]) {
        Y[i, j, k] ~ dnorm(mu_y[i, j], tausq_y)
      }

      mu_y[i, j] = alpha_genre[i] + alpha_subgenre[i, j]

      alpha_subgenre[i, j] ~ dnorm(mu_subgenre, 1 / sigma_subgenre^2)
    }

    alpha_genre[i] ~ dnorm(mu_genre, 1 / sigma_genre^2)
  }

  mu_subgenre ~ dnorm(0, 0.001)
  sigma_subgenre ~ dexp(0.001)

  mu_genre ~ dnorm(0, 0.001)
  sigma_genre ~ dexp(0.001)

  tausq_y ~ dgamma(0.001, 0.001)

  sigma_y = sqrt(1 / tausq_y)
}
```

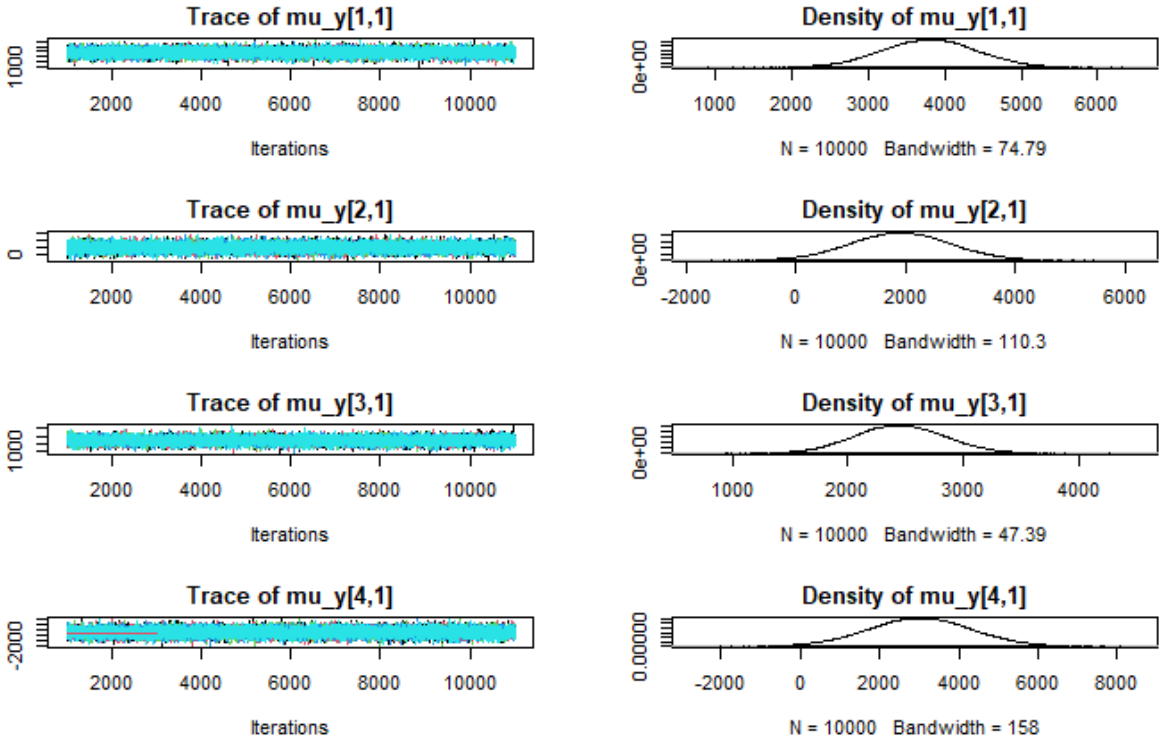
Summary Table for the μ_y and σ_y Parameters of the Model

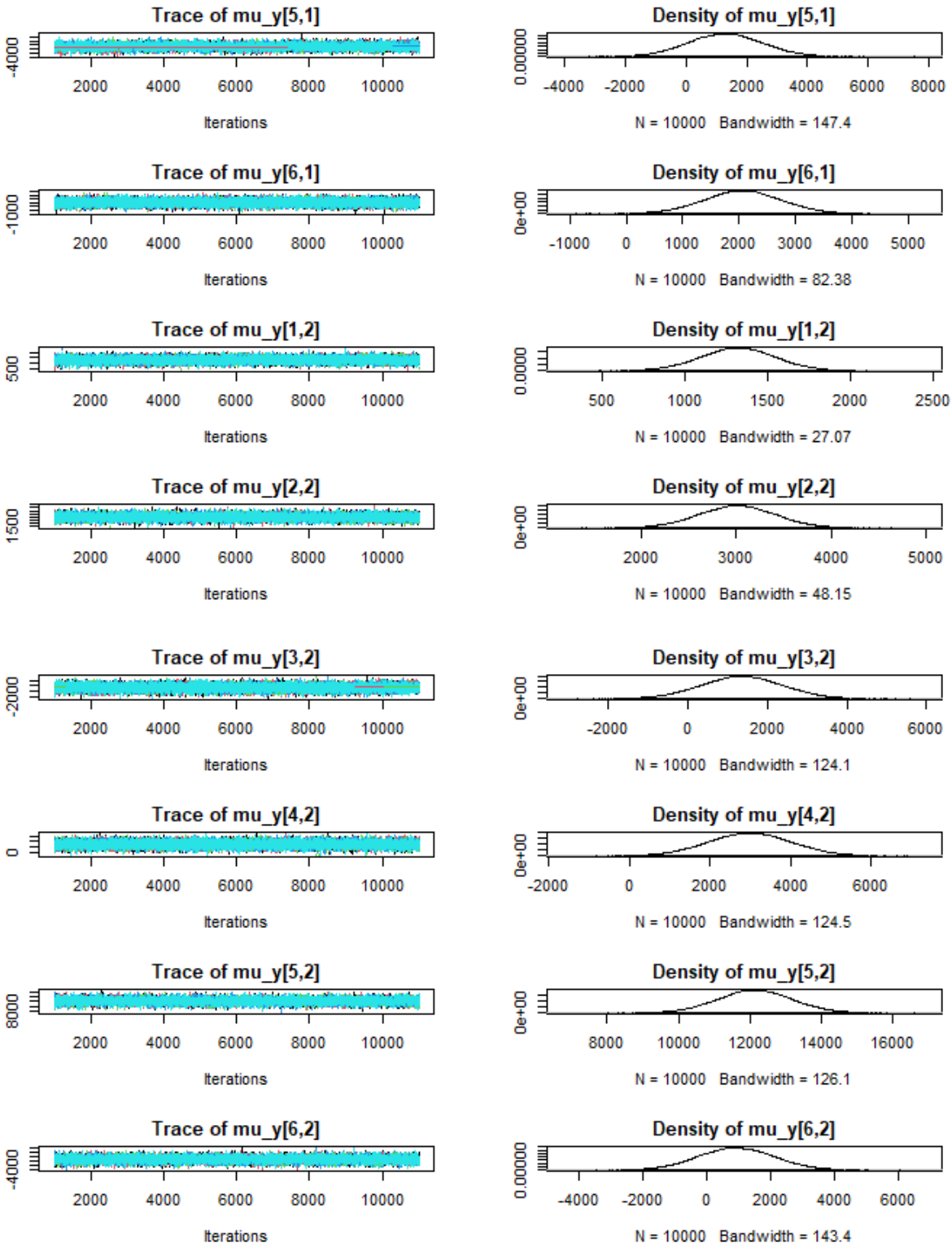
```
##
## Iterations = 11001:161000
## Thinning interval = 1
## Number of chains = 5
## Sample size per chain = 150000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##              Mean     SD Naive SE Time-series SE
## mu_y[1,1]  3792.9  613.7   0.7086         0.8928
## mu_y[2,1]  1913.3  905.4   1.0455         1.3465
## mu_y[3,1]  2447.8  390.6   0.4511         0.3939
## mu_y[4,1]  3021.0 1295.2   1.4956         1.7418
## mu_y[5,1]  1333.3 1222.5   1.4116         7.8141
## mu_y[6,1]  2058.2  675.5   0.7800         0.7423
## mu_y[1,2]  1322.9  223.3   0.2578         0.2341
## mu_y[2,2]  3004.5  394.0   0.4550         0.4232
## mu_y[3,2]  1348.7 1013.7   1.1705         1.5094
## mu_y[4,2]  2996.5 1018.6   1.1762         1.1831
## mu_y[5,2] 12116.8 1035.8   1.1960         4.2333
## mu_y[6,2]   940.6 1172.8   1.3543         1.7503
## mu_y[1,3]  1656.0 1173.1   1.3545         2.8653
## mu_y[2,3]  1361.3  485.4   0.5605         0.5563
## mu_y[3,3]  1675.0  906.2   1.0463         1.2123
## mu_y[4,3]  4357.7 1297.5   1.4983         1.6549
## mu_y[5,3]  6946.5 1305.5   1.5075         2.1158
## mu_y[6,3]  1610.8  585.5   0.6760         0.6168
## sigma_y    2865.8  100.1   0.1156         0.1216
##
## 2. Quantiles for each variable:
##
##              2.5%     25%     50%   75% 97.5%
## mu_y[1,1]  2589.7  3379.2  3792.6  4207  4995
## mu_y[2,1]   136.3  1304.4  1913.4  2524  3687
## mu_y[3,1]  1682.1  2184.9  2447.8  2711  3213
## mu_y[4,1]   480.3  2151.4  3023.1  3893  5551
## mu_y[5,1] -1050.7   506.5  1329.7  2158  3737
## mu_y[6,1]   734.5  1603.8  2058.5  2512  3385
## mu_y[1,2]   884.8  1172.4  1322.8  1473  1761
## mu_y[2,2]  2229.8  2739.7  3004.8  3270  3777
## mu_y[3,2]  -640.4   667.4  1348.2  2032  3332
## mu_y[4,2]   991.1  2312.3  2997.4  3683  4990
## mu_y[5,2] 10077.1 11421.6 12119.6 12817 14138
## mu_y[6,2] -1363.0   152.0   941.3  1730  3240
## mu_y[1,3]  -653.4   869.2  1658.2  2446  3956
## mu_y[2,3]   407.4  1034.9  1361.5  1688  2314
## mu_y[3,3]  -104.6  1064.1  1676.2  2286  3449
## mu_y[4,3]  1824.3  3481.8  4354.6  5228  6911
## mu_y[5,3]  4386.8  6068.6  6947.4  7826  9506
## mu_y[6,3]   463.2  1215.7  1611.3  2005  2761
## sigma_y    2678.5  2796.9  2862.7  2932  3071
```

Distribution of simulated σ_y values

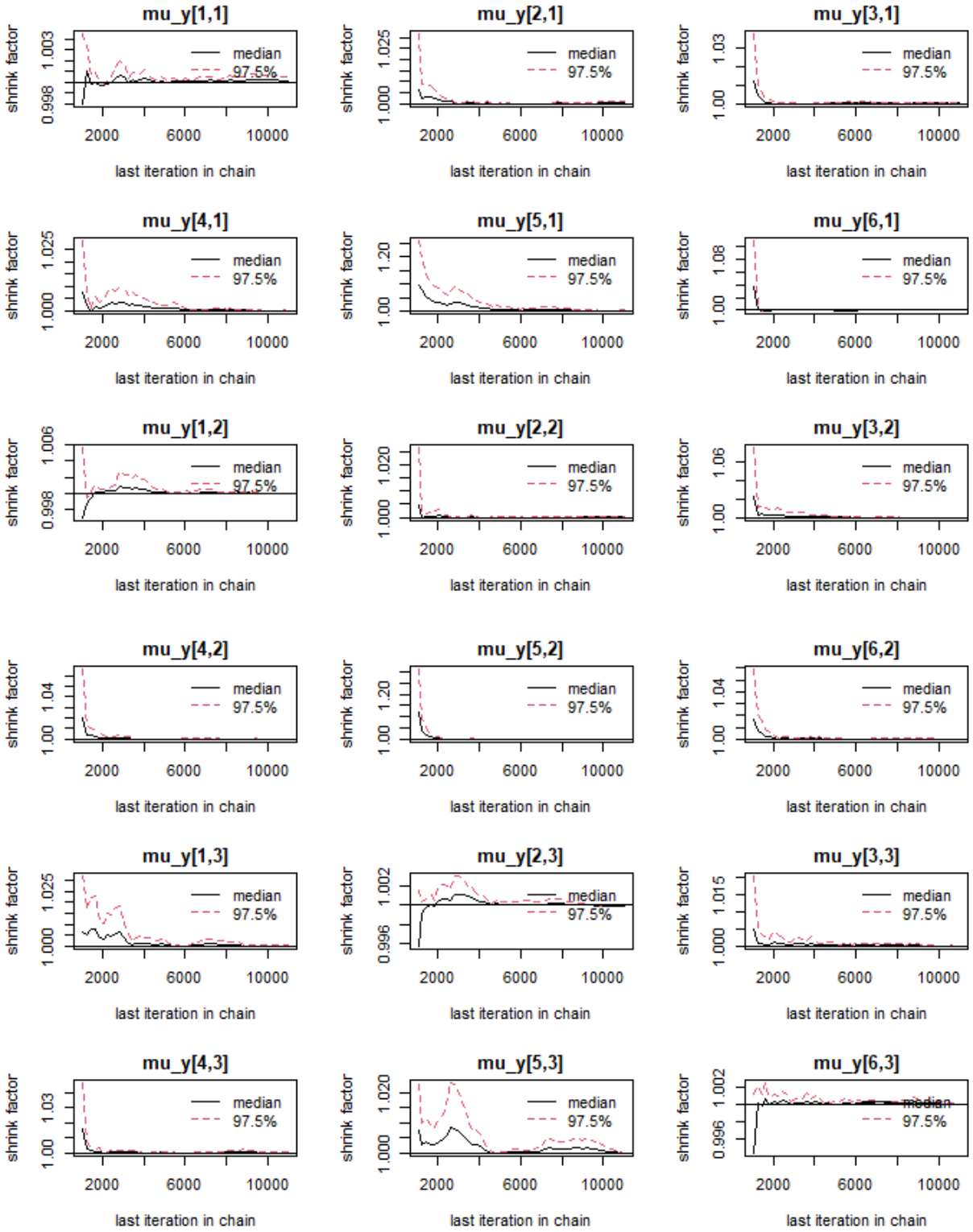


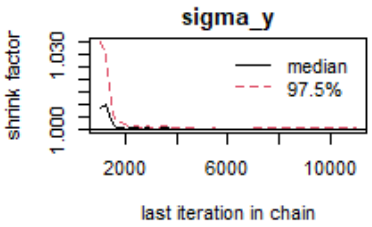
Trace & Density Plots on the Preliminary Sample



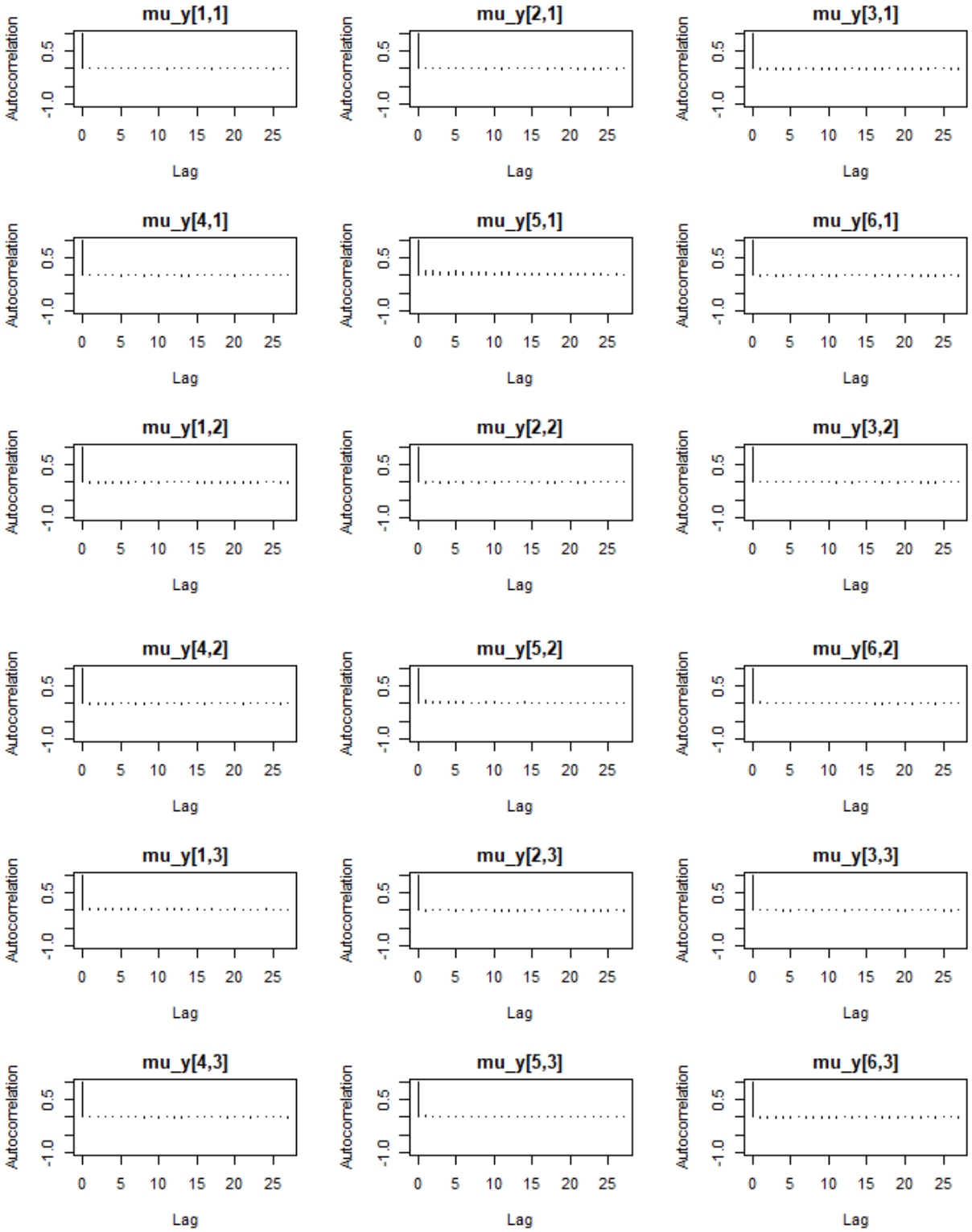

Trace of mu_y[5,1]
Iterations
Density of mu_y[5,1]
N = 10000 Bandwidth = 147.4
Trace of mu_y[6,1]
Iterations
Density of mu_y[6,1]
N = 10000 Bandwidth = 82.38
Trace of mu_y[1,2]
Iterations
Density of mu_y[1,2]
N = 10000 Bandwidth = 27.07
Trace of mu_y[2,2]
Iterations
Density of mu_y[2,2]
N = 10000 Bandwidth = 48.15
Trace of mu_y[3,2]
Iterations
Density of mu_y[3,2]
N = 10000 Bandwidth = 124.1
Trace of mu_y[4,2]
Iterations
Density of mu_y[4,2]
N = 10000 Bandwidth = 124.5
Trace of mu_y[5,2]
Iterations
Density of mu_y[5,2]
N = 10000 Bandwidth = 126.1
Trace of mu_y[6,2]
Iterations
Density of mu_y[6,2]
N = 10000 Bandwidth = 143.4

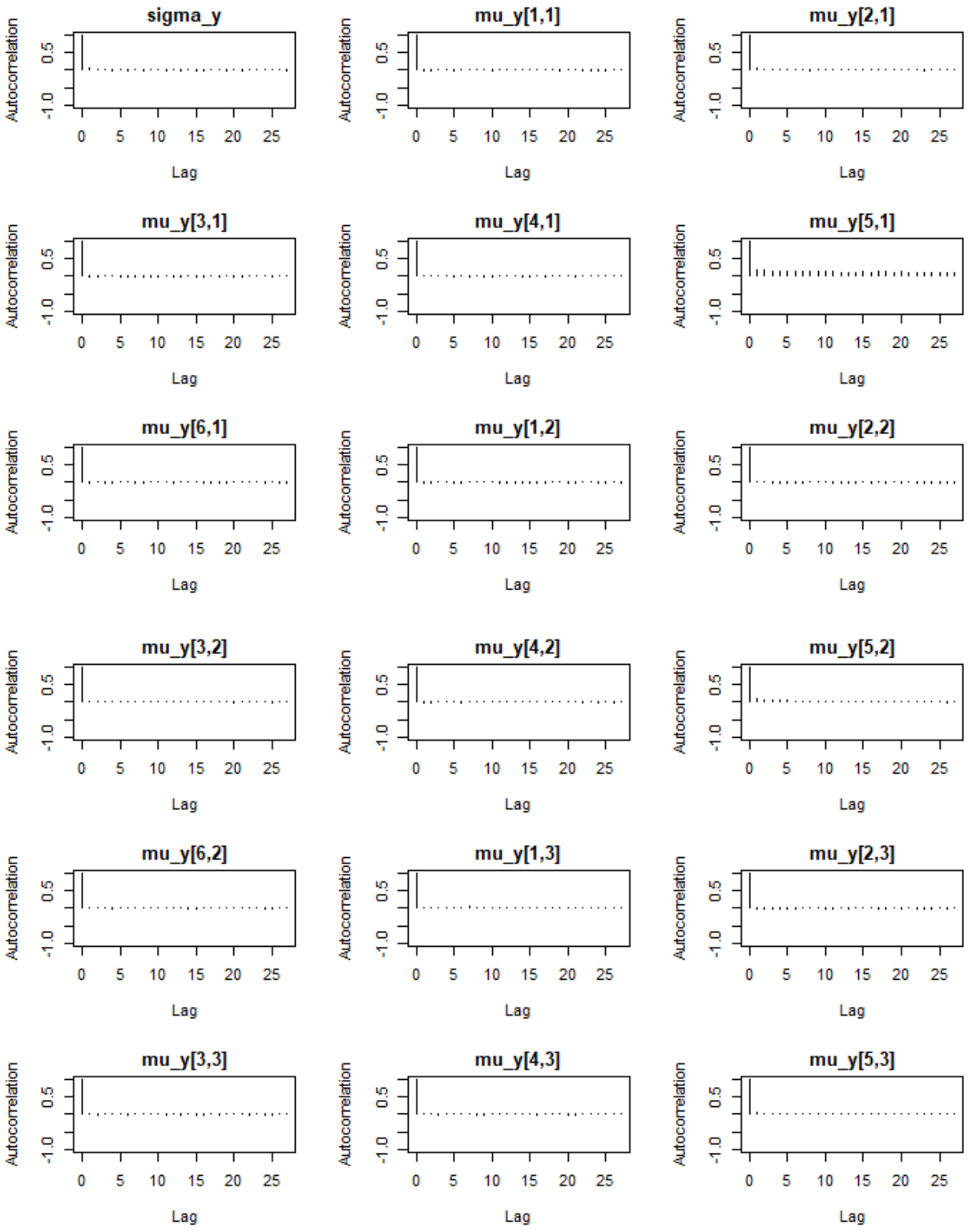
Gelman Diagnostic Plots on the Preliminary Sample

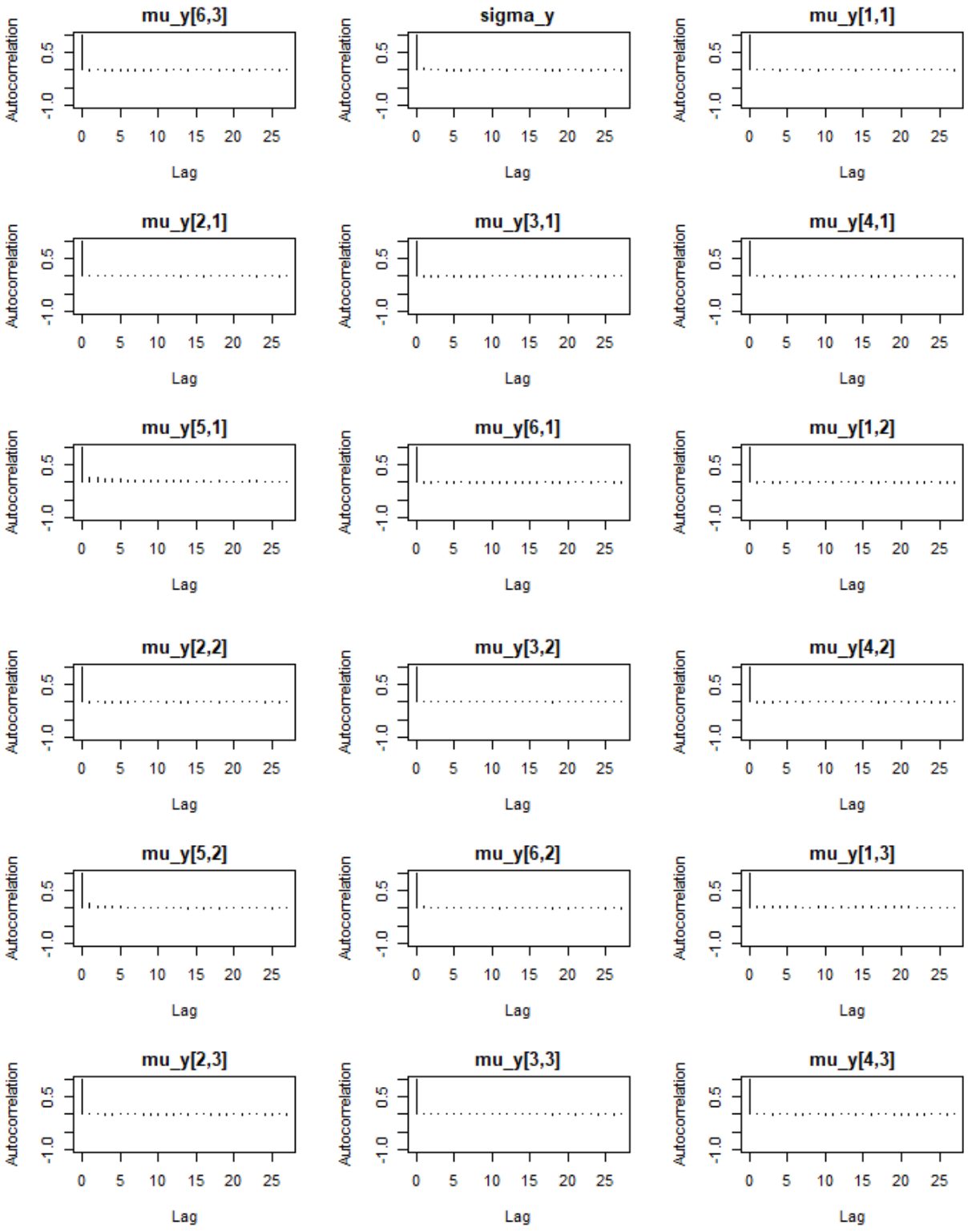


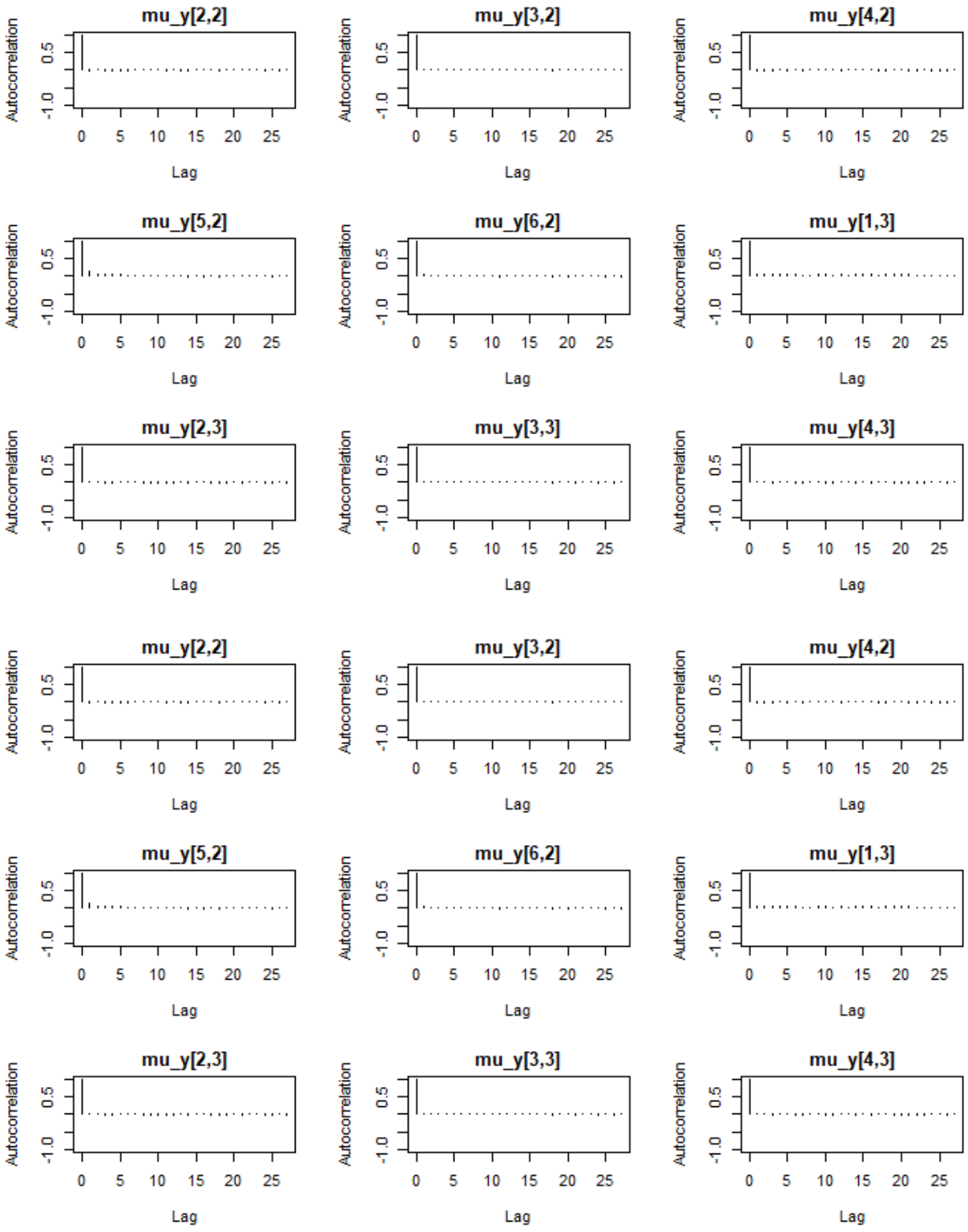

sigma_y
shrink factor
1.000
1.030
median
97.5%
2000
6000
10000
last iteration in chain

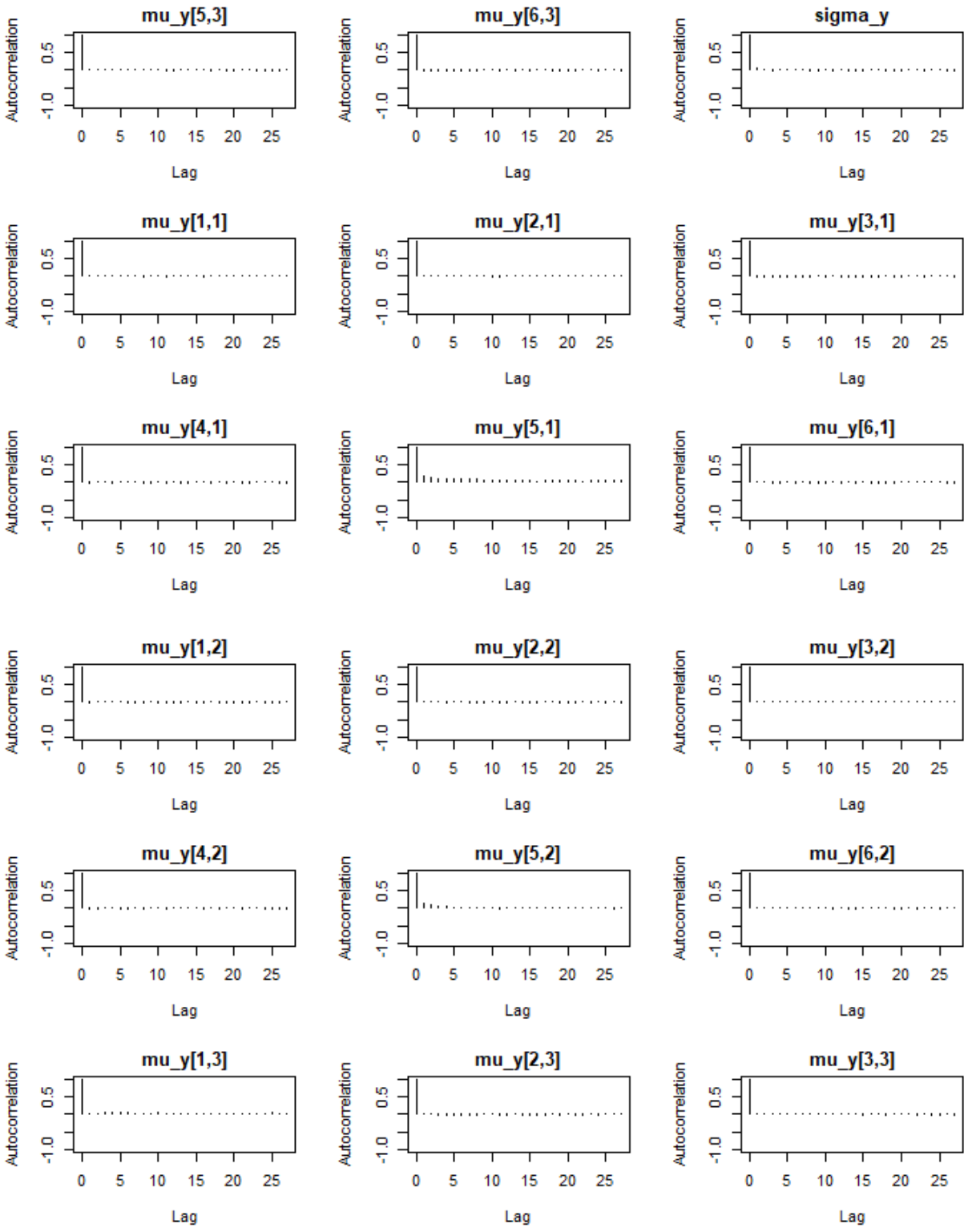
Autocorrelation Plots on the Preliminary Sample for all 5 Chains

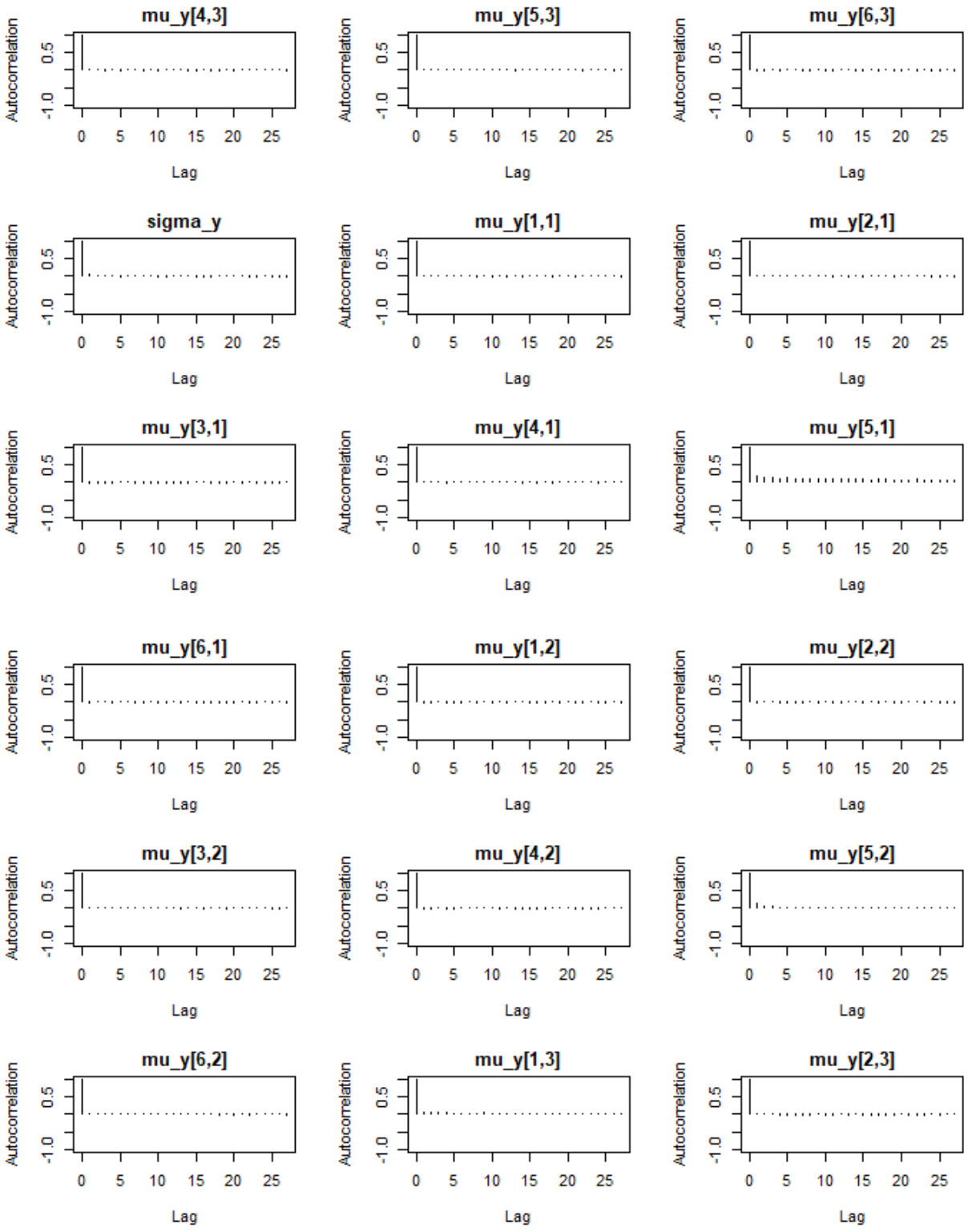


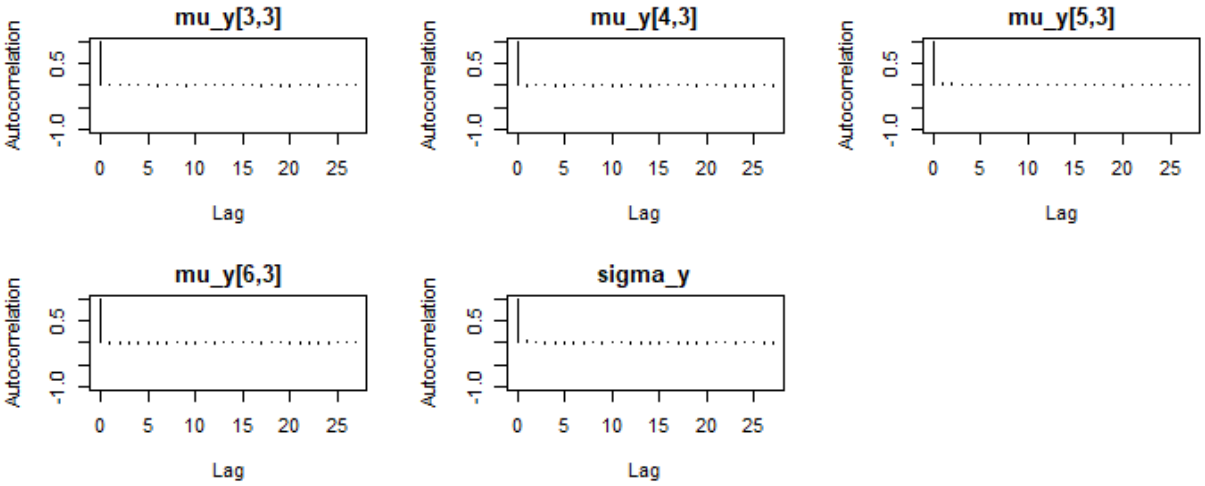

sigma_y
mu_y[1,1]
mu_y[2,1]
mu_y[3,1]
mu_y[4,1]
mu_y[5,1]
mu_y[6,1]
mu_y[1,2]
mu_y[2,2]
mu_y[3,2]
mu_y[4,2]
mu_y[5,2]
mu_y[6,2]
mu_y[1,3]
mu_y[2,3]
mu_y[3,3]
mu_y[4,3]
mu_y[5,3]
Autocorrelation
Lag
-1.0
0.5
0
5
10
15
20
25


mu_y[6,3]
sigma_y
mu_y[1,1]
mu_y[2,1]
mu_y[3,1]
mu_y[4,1]
mu_y[5,1]
mu_y[6,1]
mu_y[1,2]
mu_y[2,2]
mu_y[3,2]
mu_y[4,2]
mu_y[5,2]
mu_y[6,2]
mu_y[1,3]
mu_y[2,3]
mu_y[3,3]
mu_y[4,3]
Autocorrelation
Lag
-1.0
0.5
0
5
10
15
20
25


mu_y[2,2]
mu_y[3,2]
mu_y[4,2]
mu_y[5,2]
mu_y[6,2]
mu_y[1,3]
mu_y[2,3]
mu_y[3,3]
mu_y[4,3]
mu_y[2,2]
mu_y[3,2]
mu_y[4,2]
mu_y[5,2]
mu_y[6,2]
mu_y[1,3]
mu_y[2,3]
mu_y[3,3]
mu_y[4,3]
Autocorrelation
-1.0
0.5
0
5
10
15
20
25
Lag


mu_y[5,3]
mu_y[6,3]
sigma_y
mu_y[1,1]
mu_y[2,1]
mu_y[3,1]
mu_y[4,1]
mu_y[5,1]
mu_y[6,1]
mu_y[1,2]
mu_y[2,2]
mu_y[3,2]
mu_y[4,2]
mu_y[5,2]
mu_y[6,2]
mu_y[1,3]
mu_y[2,3]
mu_y[3,3]
Autocorrelation
Lag
-1.0
0.5
0
5
10
15
20
25


mu_y[4,3]
mu_y[5,3]
mu_y[6,3]
sigma_y
mu_y[1,1]
mu_y[2,1]
mu_y[3,1]
mu_y[4,1]
mu_y[5,1]
mu_y[6,1]
mu_y[1,2]
mu_y[2,2]
mu_y[3,2]
mu_y[4,2]
mu_y[5,2]
mu_y[6,2]
mu_y[1,3]
mu_y[2,3]
Autocorrelation
-1.0
0.5
0
5
10
15
20
25
Lag


mu_y[3,3]
mu_y[4,3]
mu_y[5,3]
mu_y[6,3]
sigma_y
Autocorrelation
1.0
0.5
-1.0
0
5
10
15
20
25
Lag