BDA Project

November 2021

Contents

1 Introduction

Decisions on the planning of store deliveries and the opening of new stores are of great importance for retail companies. To support such decisions it is of interest to understand sales dynamics and to predict future sales. This work focuses on modeling and predicting retail store sales with data provided by Walmart. In order to do precise predictions, sales time series by site location as well as by product category are given. By using Bayesian modelling approaches this work shows how to model and predict store and department specific sales.

This work limits itself to modeling sales for one department for multiple stores. Figure 1 visualizes the data structure for 4 stores of department 4.

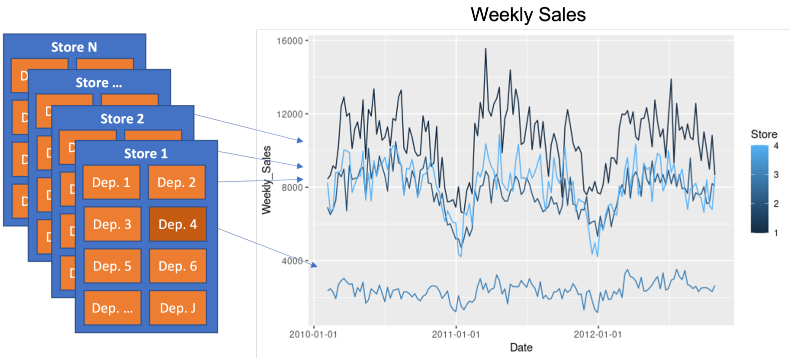

Figure 1: Data Structure

The basic modeling idea is that sales time series can be broken down into two main underlying processes: trend and seasonality. The sum of trend and seasonality gives the actual sales value. The trend is modeled by a simple linear trend whereas seasonality is enabled by Fourier series.

Two approaches are conducted: (1) separate model for store and department and (2) hierarchical model for store and department.

The separate model estimates all model parameters for each store independently. In contrast, the hierarchical model allows to use information about sales characteristics across different stores. Section 3 further elaborates the model formulation.

The reminder of the work is structured as follows: Section 2 introduces the data and analysis problem. In Section 3 the models are described. Section 4 shows the implementation of the models. Convergence and Diagnostics are discussed in Section 5. Models are compared in Section 8. Section 10 discusses and Section 11 concludes the project.

2 Data and analysis problem

We are using a dataset provided by Walmart, a popular US retail corporation. This dataset contains historical sales data from Walmart stores located in North America.

The dataset covers from 2010/02/05 to 2012/11/01, a little less than three years. Sales data on 45 different store locations are included. Each store contains a number of departments such as consumer products, grocery, or home improvement.

The task consists in forecasting the future weekly sales for each department in every store. For our analysis, we will only use past sales data, without adding additional explanatory variables such as the customer price index (CPI), the cost of fuel, or the presence of holidays.

This dataset was made public for the Walmart Store Sales Forecasting competition, which has been hosted on Kaggle in 2013. Most of the submissions in the Kaggle competitions use a nonlinear model such as random forests or gradient boosted trees. These models are usually trained on a sliding window of the past values plus additional engineered features.

Our approach differs from the Kaggle submissions as we implement a model heavily inspired by Facebook's Prophet, based on curve-fitting the time-series with a non-linear trend and multiple seasonality components. Finally, it differs from Prophet as we treat multiple time-series from different stores hierarchically.

In our analysis, for computational limitations, we will only use data on 4 stores and 1 department. In particular we will consider the first 4 stores and department 12.

Further we split the data into a train and test set, the first containing the first 100 weeks and the latter the remaining 43 weeks.

3 Model Description

3.1 Separate model

The separate model is a decomposable time-series model with two components: trend and seasonality. As the name suggest, we fit the parameters of each series separately, without sharing any information between the series.

Let N be the index of the number of observations for each time-series. Let J be the number of stores associated to one department.

3.1.1 Observation error

Let $n \in 1, ..., N$, let $j \in 1, ..., J$. Then y_{nj} is the standardized (zero mean and unit variance) weekly sale amount for the j-th store at time n. We model the prediction error using independent normal residuals:

$$y_{nj} \sim N(\hat{y}_{nj}, \sigma_{\text{obs}})$$

The residuals have zero mean and variance σ_{obs} shared among all time-steps and stores.

3.1.2 Decomposable time-series model

The actual model is specified in $\hat{y}_{nj}$. Let $\hat{y}_j$ be a n-dimensional vector defined as:

$$\hat{y}_j = \text{trend}_j + \text{offset}_j + X\beta$$

where trend_j and offset_j are, respectively, the piecewise linear trend component, and the offsets necessary to make this trend continuous. Finally $X\beta$ is the seasonality component of the j-th time-series.

3.1.3 Trend and offset

Let L be the number of changepoints in the piecewise linear trend. Let s_l for $l \in 1, ..., L$ be the times at which the changes occur. For our models, we set the parameter $L = 3$ representing roughly half-year trend changes.

The trend component is defined as:

$$\text{trend}_j = k_j t + A\delta_j \circ t$$

where k_j is the initial growth component associated to every store, A is a matrix defined as $(A)_{nl} = t_n \geq s_l$, δ_j is a vector containing the changes in growth at every changepoint time.

To make this linear trend continuous, we need to create an offset vector defined as follows:

$$\text{offset}_j = m_j + A(-s \circ \delta_k)$$

where m_j is the initial offset component for every store.

3.1.4 Seasonality

Following Prophet's approach, we model seasonality using Fourier series. Let K be the number of Fourier components, let $k \in 1, ..., K$. We define a matrix X and a vector $\beta \in \mathbb{R}^K$. For our models, we set the parameter $K = 10$.

$$X_n = \left[\cos\left(\frac{2\pi 1t}{365.25/7}\right), ..., \sin\left(\frac{2\pi Kt}{365.25/7}\right)\right]$$

Multiplying X and β gives us the seasonality element.

3.1.5 Priors

$$\begin{aligned}
\sigma_{\text{obs}} &\sim N(0, 0.5) \\
k_j &\sim N(0, 5) \\
m_j &\sim N(0, 5) \\
\delta_{lj} &\sim N(0, 5) \\
\beta_{kj} &\sim N(0, 5/20)
\end{aligned}$$

3.1.6 Model summary

$$\begin{aligned}
y_{nj} &\sim N(\hat{y}_{nj}, \sigma_{\text{obs}}) \\
\hat{y}_j &= \text{trend}_j + \text{offset}_j + X\beta \\
\text{trend}_j &= k_j t + A\delta_j \circ t \\
\text{offset}_j &= m_j + A(-s \circ \delta_k) \\
(A)_{nl} &= t_n \geq s_l \\
X_n &= \left[\cos\left(\frac{2\pi 1n}{365.25/7}\right), ..., \sin\left(\frac{2\pi Kn}{365.25/7}\right)\right]
\end{aligned}$$

3.2 Hierarchical model

From our exploratory data analysis, we noticed that the sales of one department were highly similar across various stores. Because of this commonality, we decided to use a hierarchical model, with parameters shared across different stores.

3.2.1 Priors

$$\begin{aligned}
\sigma_{\text{obs}} &\sim N(0, 0.5) \\
k_j &\sim N(\mu_k, \sigma_k) \\
m_j &\sim N(\mu_m, \sigma_m) \\
\delta_{lj} &\sim N(\mu_\delta, \sigma_\delta) \\
\beta_{kj} &\sim N(\mu_\beta, \sigma_\beta)
\end{aligned}$$

3.2.2 Hyperpriors

$$\mu_k \sim N(0,5)$$
$$\mu_m \sim N(0,5)$$
$$\mu_\delta \sim N(0,5)$$
$$\mu_\beta \sim N(0,5)$$
$$\sigma_k \sim \text{Inv-}\chi^2(1)$$
$$\sigma_m \sim \text{Inv-}\chi^2(1)$$
$$\sigma_\delta \sim \text{Inv-}\chi^2(1)$$
$$\sigma_\beta \sim \text{Inv-}\chi^2(1)$$

3.3 Priors and their Justification

For our separate model the priors are set as rather weakly informative as shown above. As we do have significant amount of data it is reasonable to allow for flexibility. Further, the specifications are aligned with the priors introduced in Facebook's Prophet. For our parameter k we set the prior centered around zero as we observe no clear overall trend in any sales time series. The same argument holds for the priors for δ. We also set the priors of β centered around zero with a variance of 5 to allow to infer seasonality from the data. The parameter σ_{obs} has a prior somewhat close to 0 since we want the model to predict y with a limited variance.

A sensitivity analysis is conducted where we set the priors more narrow as shown below:

$$\sigma_{\text{obs}} \sim N(0,0.5)$$
$$k_j \sim N(0,1)$$
$$m_j \sim N(0,1)$$
$$\delta_{lj} \sim N(0,1)$$
$$\beta_{kj} \sim N(0,1)$$

The results of the mean and standard deviation of the parameters of both separate models are shown in Table 3. Compared to the priors presented in sub-section 3.2 we see little differences. Thus, our model seems to be robust to more narrow priors. Different tests with unreasonable means reveal bad results. Further, no major differences between k-values and effective sample size are observed.

For our hierarchical model we do a similar kind of analysis. The hyper-priors are essentially derived from the separate models with the same reasoning as discussed above. We test an approach with even broader hyper-priors for our variances as shown below.

$$\mu_k \sim N(0,5)$$
$$\mu_m \sim N(0,5)$$
$$\mu_\delta \sim N(0,5)$$
$$\mu_\beta \sim N(0,5)$$
$$\sigma_k \sim \text{Inv-}\chi^2(2)$$
$$\sigma_m \sim \text{Inv-}\chi^2(2)$$
$$\sigma_\delta \sim \text{Inv-}\chi^2(2)$$
$$\sigma_\beta \sim \text{Inv-}\chi^2(2)$$

The result change little as reported in the figures (4 and 5) below and in Table 4. In our further approach we use the specifications presented in 3.2.

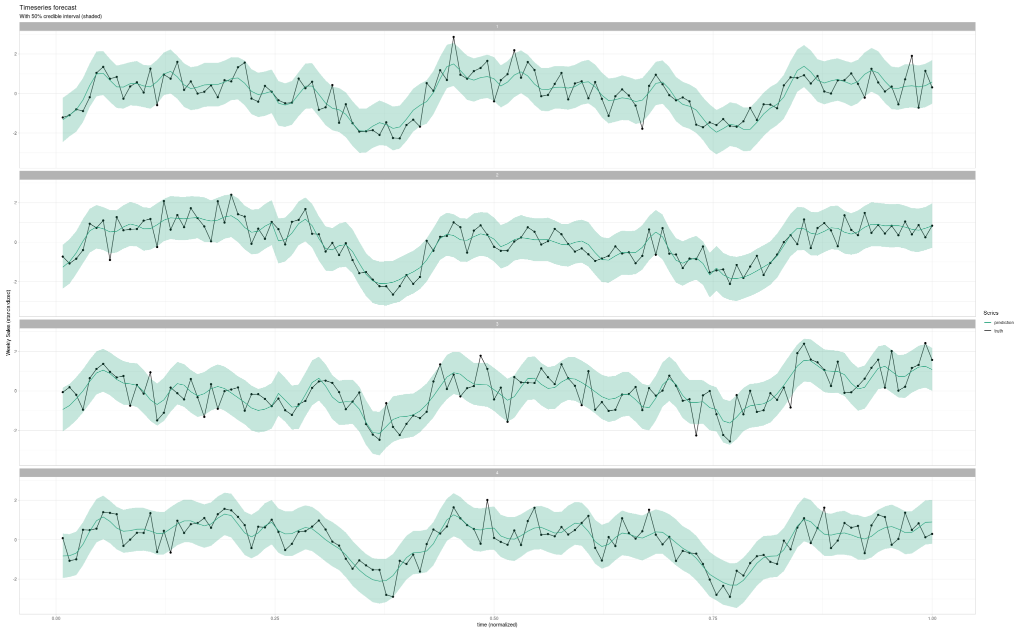

Figure 2: Predictive posterior results with wide priors - separate model

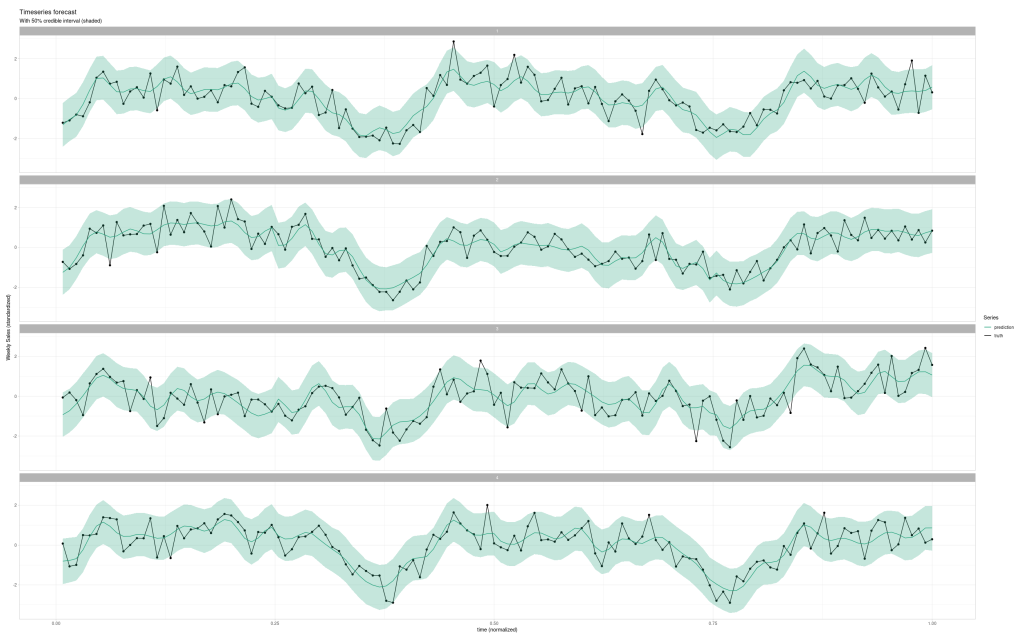

Figure 3: Predictive posterior results with narrow priors - separate model

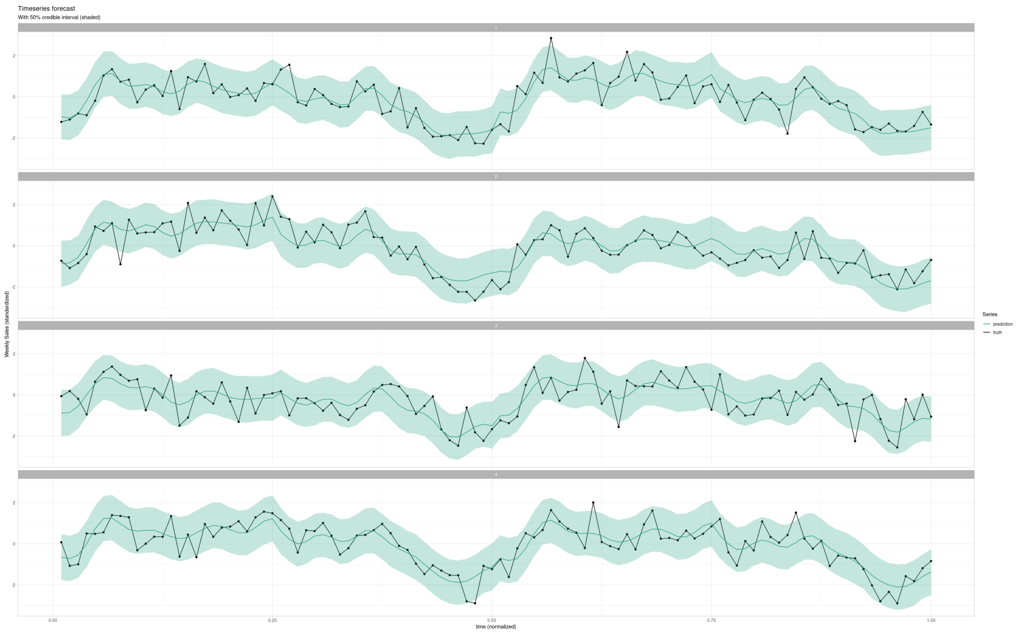

Figure 4: Predictive posterior results with narrow priors - hierarchical model

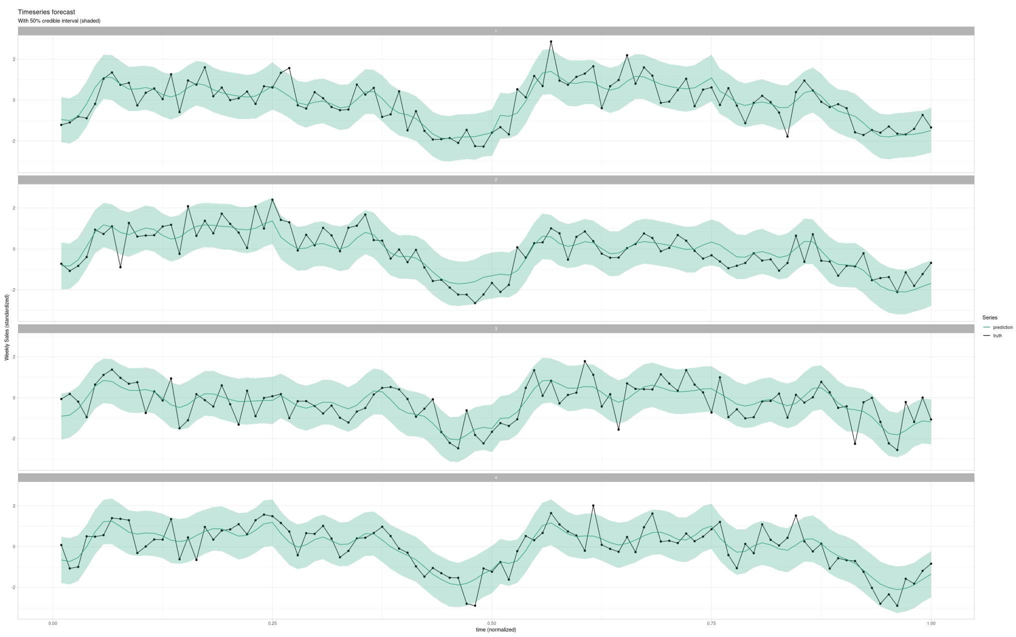

Figure 5: Predictive posterior results with wide priors - hierarchical model

4 Stan Implementation

The code listings below show the implmentation of the separate and hierarchical model.

```
data {
    int<lower=1> N; # Length of Time Series
    int<lower=1> J; # Number of Time Series
    int<lower=1> K; # Degree of Seasonality
    int<lower=1> L; # Number of Trend Changepoints
    vector[N] t; # week number
    matrix[N,J] y; # sales
    matrix[N,K] X; # Seasonality Matrix
    matrix[N, L] A; # Trend Matrix
    vector[L] s; # Change Points

    real sigma_k;
    real sigma_m;
    real sigma_beta;
    real sigma_delta;
}
parameters {
    vector [J] k;
    vector [J] m;
    matrix[K,J] beta;
    matrix[L,J] delta;
    real<lower=0> sigma_obs;
}
transformed parameters {
    matrix[N,J] yhat;
    matrix[N,J] trend;
    matrix[N,J] offset;
    //trend = A*delta;

    for (j in 1:J) {
        trend[,j] = k[j] * t + A*delta[,j] .* t;
        offset[,j] = m[j] + A * -(s .* delta[,j]);
        yhat[,j] = trend[,j] + X * beta[,j] + offset[,j];
    }
}


model {
    for (j in 1:J){
        // how does the model change when priors are adjusted (wider/narrow)
        k[j] ~ normal(0, sigma_k);
        m[j] ~ normal(0, sigma_m);
        beta[,j] ~ normal(0, sigma_beta);
        delta[,j] ~ double_exponential(0, sigma_delta);
    }
    sigma_obs ~ normal(0, 0.5);

    for  (j in 1:J) {
        y[,j] ~ normal(yhat[,j], sigma_obs);
    }
}
generated quantities{
    matrix[N,J] log_lik;
    matrix[N,J] y_pred;
    for (i in 1:N) {
        for(j in 1:J){
        log_lik[i,j] = normal_lpdf(y[i,j]|yhat[i,j], sigma_obs);
        y_pred[i,j] = normal_rng (yhat[i,j], sigma_obs);
        }
    }
}
```

Listing 1: Implementation of the Separate Model

```
data {
    int<lower=1> N; # Length of Time Series
    int<lower=1> J; # Number of Time Series
    int<lower=1> K; # Degree of Seasonality
    int<lower=1> L; # Number of Trend Changepoints
    vector[N] t; # week number
    matrix[N,J] y; # sales
    matrix[N,K] X; # Seasonality Matrix
    matrix[N, L] A; # Trend Matrix
    vector[L] s; # Change Points
}
parameters {
    vector [J] k;
    real k_shared;
    real <lower=0> sigma_k;
    vector [J] m;
    real m_shared;
    real <lower=0> sigma_m;
    matrix[K,J] beta;
    vector [K] beta_shared;
    real <lower=0> sigma_betas;
    matrix[L,J] delta;
    vector[L] delta_shared;
    real <lower=0> sigma_deltas;
    real<lower=0> sigma_obs;
}
transformed parameters {
    matrix[N,J] yhat;
    matrix[N,J] trend;
    matrix[N,J] offset;

    for (j in 1:J) {
        trend[,j] = k[j] * t + A*delta[,j] .* t;
        offset[,j] = m[j] + A * -(s .* delta[,j]);
        yhat[,j] = trend[,j] + X * beta[,j] + offset[,j];
    }
}
model {
    k_shared ~  normal(0,5);
    sigma_k ~ inv_chi_square(1);
    m_shared ~ normal(0,5);
    sigma_m ~ inv_chi_square(1);

    beta_shared ~  normal(0, 5);
    delta_shared ~ normal(0, 5);

    sigma_betas ~ inv_chi_square(1);
    sigma_deltas ~ inv_chi_square(1);

    sigma_obs ~ normal(0,0.5);

    for (j in 1:J){
        k[j] ~ normal(k_shared, sigma_k);
        m[j] ~ normal(m_shared, sigma_m);
        beta[,j] ~ normal(beta_shared, sigma_betas);
        delta[,j] ~ normal(delta_shared, sigma_deltas);
    }

    for  (j in 1:J) {
        y[,j] ~ normal(yhat[,j], sigma_obs);
    }
}
generated quantities{
    matrix[N,J] log_lik;
    matrix[N,J] y_pred;
    for (i in 1:N) {
        for(j in 1:J){
```

```
        log_lik[i,j] = normal_lpdf(y[i,j]|yhat[i,j], sigma_obs);
        y_pred[i,j] = normal_rng (yhat[i,j], sigma_obs);
        }
    }
}
```

Listing 2: Implementation of the Hierarchical Model

4.1 Stan Specification

In both the separate and hierarchical model we set the default specifications when running Stan, i.e. number of chains is set to 4, number of iterations is set to 2000 (first 1000 for warmup) and the maximum tree depth is 10.

The code below shows how the model is run.

```
sep_model_stores_department <- stan(file = 'Project_hier_v2.stan',
                                    data = data,
                                    refresh = 100,
                                    chains = 4,
                                    iter = 2000,
                                    warmup = 1000)
```

5 Convergence and Diagnostics

In this section the convergence of the presented models is analyzed. The $\hat{R}$ values of the separate and hierarchical model are presented below in Figure 6 and 7. Values close to one are observed.[1]. This means that the distribution of our chains are approaching each other, i.e. they are converging in both the separate and hierarchical model.

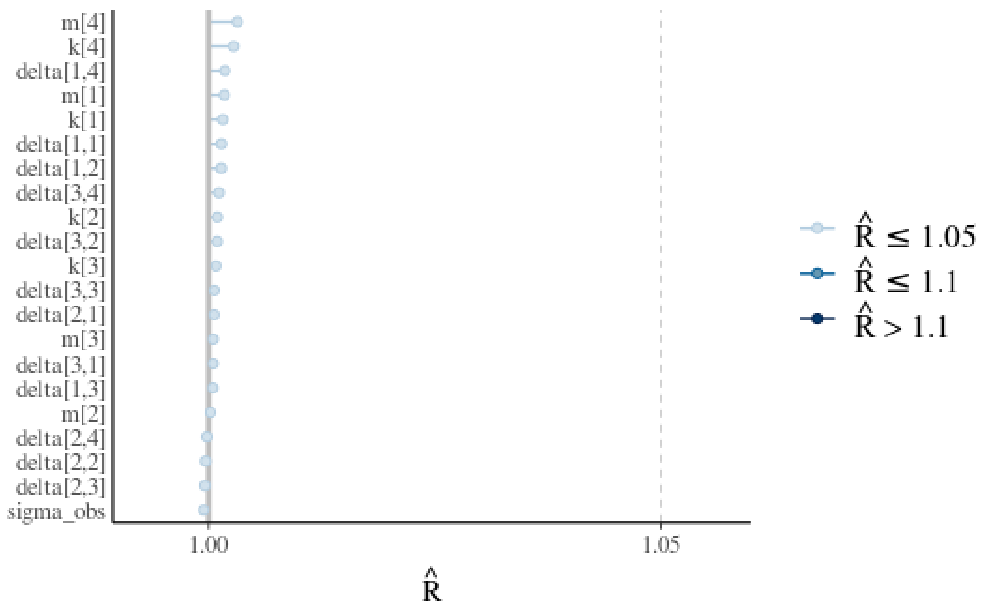

Figure 6: $\hat{R}$ values - separate model

[1]We rely on the Rhat function from the Rstan library to calculate $\hat{R}$ values.

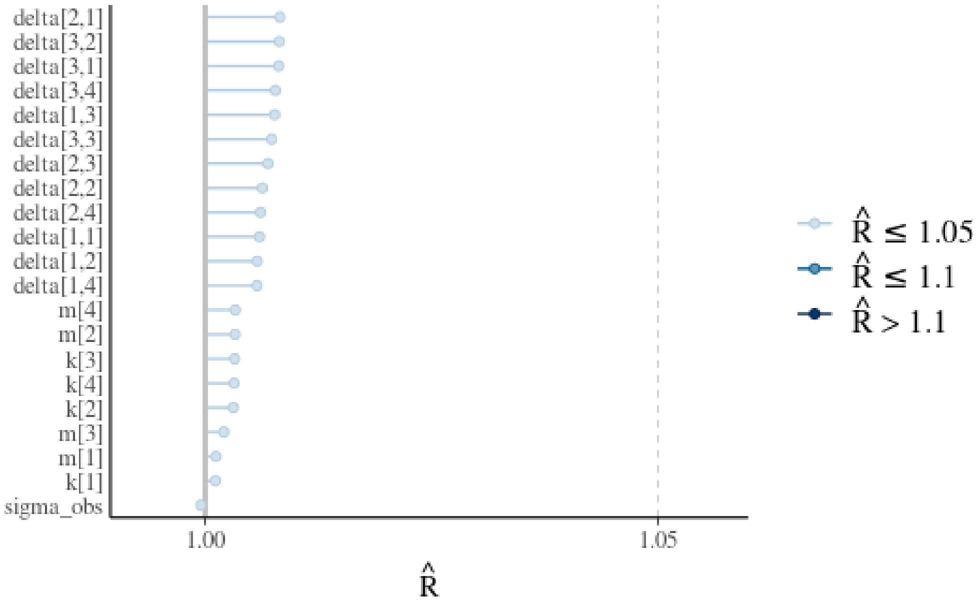

Figure 7: $\hat{R}$ values - hierarchical model, selected parameters

During the simulation no warnings about convergence problems and maximum tree depth are reported for the separate and hierarchical model. We use Stan standard specification when running the models, i.e. tree depth = 10. The code below shows the output of the HMC diagnostic check.

```
> check_hmc_diagnostics(fit)

Divergences:
0 of 4000 iterations ended with a divergence.

Tree depth:
0 of 4000 iterations saturated the maximum tree depth of 10.

Energy:
E-BFMI indicated no pathological behavior.
```

Figure 8 and 9 shows the ratio of the effective sample size for the parameters of the separate and hierarchical model respectively. For some parameters we observe values smaller than 0.5 which indicates that the those parameter estimation are less stable and show a higher standard deviation. For the hierarchical model we observe even values lower than 0.1 which resulted also in a warning during running the model. Further improvements are necessary to increase the effective sample size. One possibility is to increase the iteration size per chain. Also changes in the parameterization of the model could lead to better results.

Based on the analysed diagnostics it can be assumed that our sampler is working correctly and is able to adequately approximate the posterior distributions for both the separate and hierarchical model. For the hierarchical we may have higher uncertainty about some parameters since the effective sample size is low.

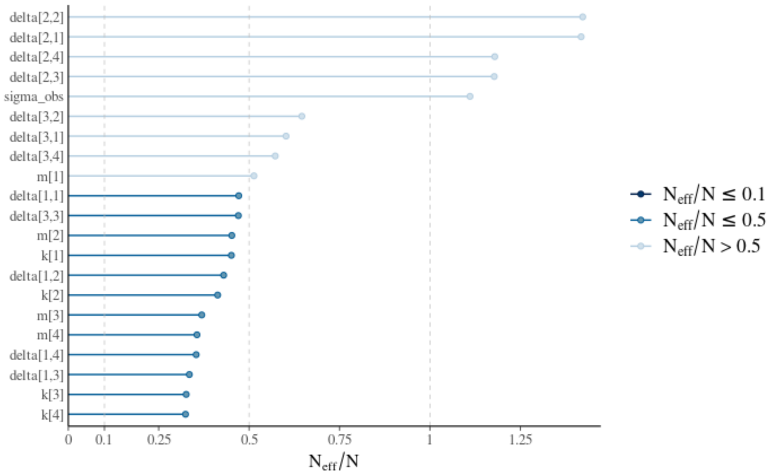

Figure 8: $\frac{n_{eff}}{n}$ - separate model

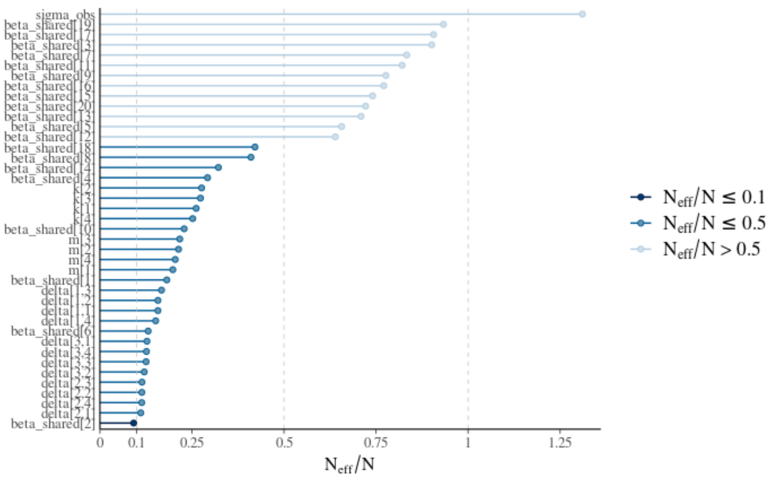

Figure 9: $\frac{n_{eff}}{n}$ - hierarchical model

6 Posterior predictive check

In figures 10 and 11 we plot the 1-week-ahead out-of-sample forecasts of, respectively the separate and the hierarchical model.

From just these plots we can observe how the seasonality has been modeled well, leading to a good fit. However, especially for the separate model, the trend element does not seem to capture the offset between seasonality and ground truth.

From these checks we can conclude that we need better trend modeling, perhaps adding additional explanatory variables.

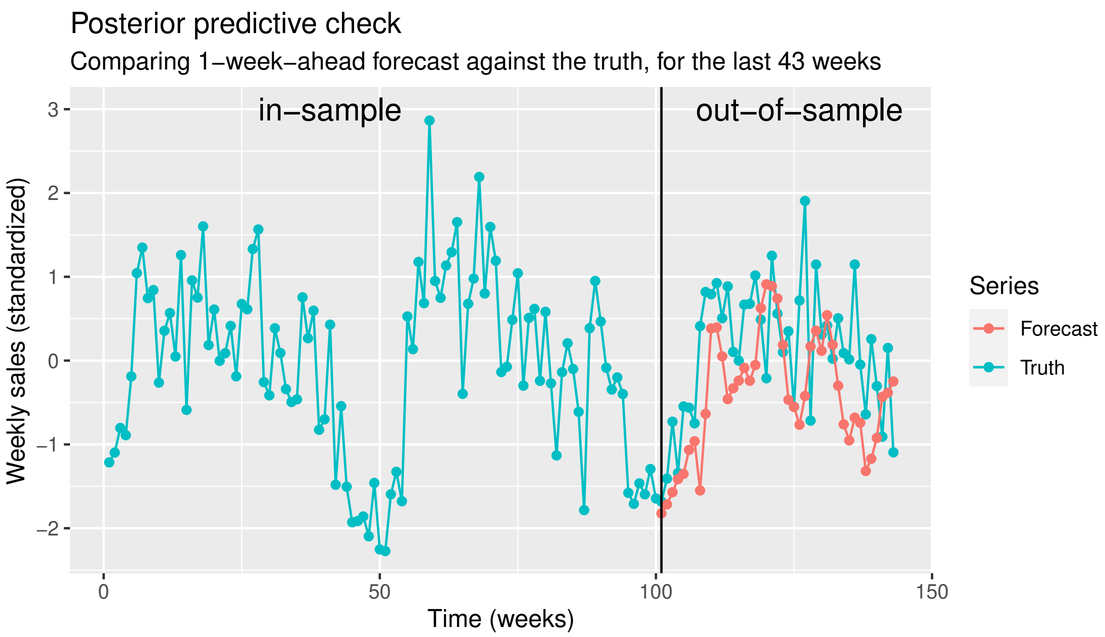

Figure 10: Posterior predictive check for store 1, separate model

7 Model comparison

To evaluate the performance of our models we use the Expected Log Predictive Density (ELPD). Because we are in a timeseries setting, with a focus on forecasting, we want to estimate the future predictive performance as precisely as possible.

Unfortunately, as discussed in Approximate leave-future-out cross-validation for Bayesian time series models, leave-one-out cross-validation is not optimal for evaluating predictive performance, as it allows information from the future i.e. $y_{t+1}, y_{t+2}, \ldots$ to influence predictions of the past e.g., y_t.

Because of this "information leakage" we turn to 1-step-ahead leave-future-out cross-validation, where a model trained on $y_1, \ldots, y_t$ is used to predict y_{t+1}. This step is repeated for $N - L$ times, where N is the total number of time steps and L is the minimum amount of time steps we will use to train our model.

7.1 Results

Shown in table 7.1.

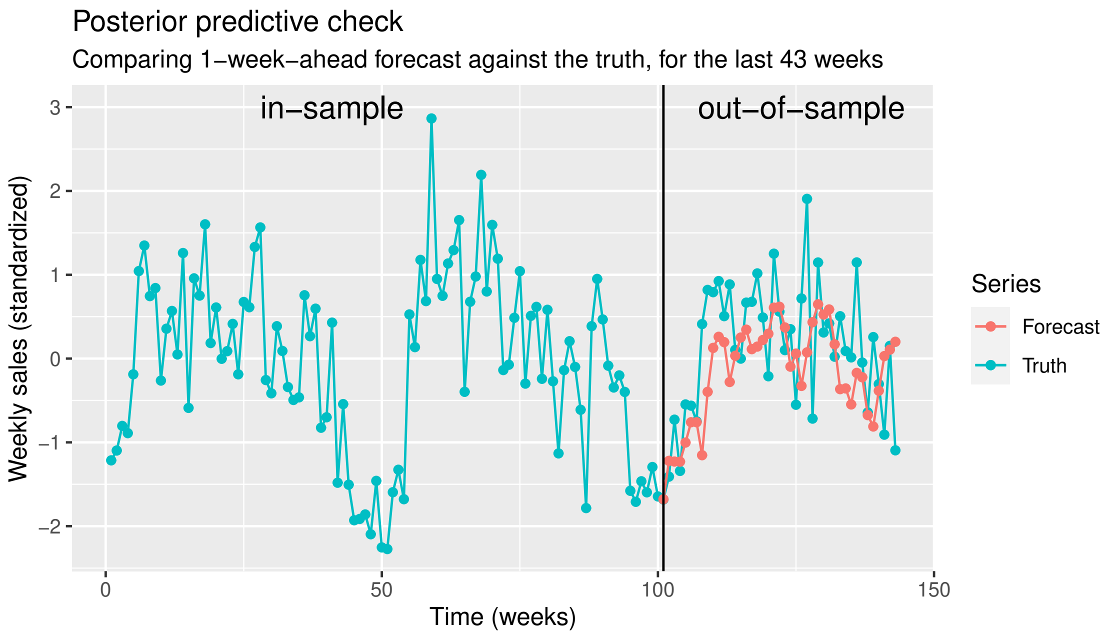

Figure 11: Posterior predictive check for store 1, hierarchical model

		Store			
		1	2	3	4
ELPD	Separate	-57	-45	-67	-50
	Hierarchical	**-48**	**-42**	**-53**	**-43**

Table 1: Expected Log Posterior Densities for both models, evaluated on every store.

7.2 Discussion

Comparing ELPDs, we see that the hierarchical model fits the data better in all stores. The difference in ELPDs is large enough (i.e. bigger the the Monte Carlo standard error) for us to conclude that the hierarchical model has achieved a better fit.

8 Predictive Performance Assessment

We evaluate the practical usefulness of the accuracy using a similar metric to the one used in the Kaggle competition. We will be using mean absolute error (MAE):

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

8.1 Results

Shown in table 8.1.

8.2 Discussion

Consistent with the ELPDs, we see in table 8.1 that the hierarchical model has a lower mean average error in all stores. Considering that the Kaggle competition winners achieved a (weighted)

		Store			
		1	2	3	4
MAE	Separate	0.7	0.5	0.9	0.6
	Hierarchical	**0.5**	**0.5**	**0.7**	**0.5**

Table 2: Mean Average Errors for both models, evaluated on every store.

MAE of about 0.14 (after standardization), we are satisfied with the performance of our model, especially considering the small dataset we have used for training.

9 Sensitivity Analysis

We find that narrowing the priors in our separate model does not significantly change our posteriors. Though, in the hierarchical model we see problems of low n_{eff} with both narrow and wider hyper-prior distributions.

Since no significant changes are observed, it can be concluded that there is enough data, i.e. the results are less sensitive to the specified priors.

For more detailed sensitivity analysis please refer to Section 3.3.

10 Discussion

10.1 Issues

The model building workflow is by nature an iterative process. This requires refitting the model several times. In our case, Stan's total time for the separate model is about 4 minutes, and for the hierarchical model it is about 6 minutes. Even though this is still manageable, it is still far from the near real-time performance we would get from just using maximum likelihood estimation (MLE) paired with an optimizer such as L-BFGS.

When plotting the predictions and the true values in our posterior predictive checks, we noticed that the seasonality was not perfect. That is, peaks in weekly sales could occasionally arrive later than expected. It is not clear why this happens (perhaps a late winter affected the demand for warm clothes), but the model in its current state cannot model this properly.

When trying to evaluate the predictive performance, we struggled to find a robust method for extending the estimated piecewise linear trend into the future. Prophet's paper proposes to estimate a new parameter λ from the data, and use that to simulate further realization of the trends. In our case we assumed that the trend would be similar to the initial growth, and used that to extend our predictions.

10.2 Potential improvements

To reduce the fitting time, a more optimized implementation of the Stan model could be helpful. Taking care to vectorize as much as possible and perhaps reparametrizing the model. Additionally it could be interesting to explore non-MCMC approaches, such as variational inference based methods.

In the full dataset from the Kaggle competition, a number of explanatory variables were provided. Adding them to the model could potentially lead to big improvements in the predictive performance.

Finally, testing a wider range of interpolating functions could lead to better model fits, at the cost of an higher number of parameters and risk of overfitting. In particular, we would like to test multiplicative effects (thus escaping the realm of Generalized Additive Models or GAMs) and other trends such as logistic and exponential.

11 Conclusion

This work introduced a Bayesian approach to model weekly sales of Walmart retail stores. Inspired by Facebook's Prophet a separate and hierarchical model is implemented in Stan.

We observe that both models are flexible enough to capture the sales dynamics. Also, we do not observe over-fitting nor under-fitting which we can see from our prior sensitivity analysis.

When it comes to predictive performance assessment we observe that the hierarchical model performs slightly better than the separate. A reason for this may be that dynamics are better generalized as it shares information across multiple stores.

In general, our run time for fitting the model is long which leaves room for further improvements of the model specifications. This will also facilitate running larger models that take into account more stores for a single department.

11.1 What was learned?

During our work we learned how to apply Bayesian modelling techniques to time series data. We were inspired by the modeling approach of Facebook's Prophet.

Further, we learned how to apply a Bayesian model building workflow including prior specification and sensitivity testing as well as convergence and posterior predictive checks.

Lastly, we discovered and tested a new cross validation technique - leave future out CV - which is designed for comparing different time series models.

	Separate Model with wide priors						Separate Model with narrow priors					
parameter	mean	sd	2.50%	97.50%	n_eff	Rhat	mean	sd	2.50%	97.50%	n_eff	Rhat
sigma_obs	0.6139	0.0212	0.5731	0.6564	4443.9570	0.9995	0.6138	0.0208	0.5754	0.6580	4817.2150	0.9997
m[1]	-0.6048	0.2396	-1.0750	-0.1396	2053.8380	1.0018	-0.5629	0.2348	-1.0247	-0.0883	1912.1770	1.0004
m[2]	0.0152	0.2428	-0.4654	0.4932	1808.9690	1.0002	0.0067	0.2355	-0.4574	0.4722	1871.8970	1.0005
m[3]	0.0030	0.2424	-0.4703	0.4855	1476.2580	1.0005	-0.0028	0.2301	-0.4507	0.4374	2031.5690	0.9998
m[4]	-0.1462	0.2425	-0.6245	0.3193	1423.5190	1.0032	-0.1272	0.2314	-0.5810	0.3513	2052.1430	1.0013
k[1]	0.1340	0.0703	0.0018	0.2747	1804.2120	1.0016	0.1240	0.0698	-0.0144	0.2636	1639.9320	1.0006
k[2]	0.1211	0.0713	-0.0171	0.2623	1652.4530	1.0010	0.1228	0.0702	-0.0149	0.2597	1773.1660	0.9998
k[3]	-0.1315	0.0712	-0.2711	0.0105	1304.3440	1.0008	-0.1291	0.0688	-0.2597	0.0039	1822.9220	1.0007
k[4]	0.0755	0.0720	-0.0648	0.2177	1297.0270	1.0028	0.0715	0.0684	-0.0638	0.2043	1855.0820	1.0010
beta[1,1]	-0.5714	0.1057	-0.7772	-0.3665	2566.1010	1.0007	-0.5719	0.1060	-0.7782	-0.3590	2381.9140	0.9998
beta[1,2]	-0.7873	0.1070	-0.9955	-0.5802	2434.8280	1.0009	-0.7805	0.1072	-0.9897	-0.5690	2478.9870	1.0016
beta[1,3]	-0.4829	0.1071	-0.6902	-0.2700	1988.0770	0.9997	-0.4790	0.1076	-0.6859	-0.2688	2704.5850	1.0000
beta[1,4]	-0.7988	0.1090	-1.0142	-0.5927	2071.3490	1.0007	-0.7973	0.1043	-1.0036	-0.5965	2813.2130	1.0001
beta[2,1]	0.7857	0.0862	0.6121	0.9514	5227.8240	0.9996	0.7757	0.0834	0.6109	0.9363	5104.5170	0.9995
beta[2,2]	0.3662	0.0876	0.1980	0.5375	5935.3340	0.9995	0.3654	0.0886	0.1884	0.5384	5288.5290	1.0007
beta[2,3]	0.4927	0.0878	0.3200	0.6597	5071.7150	0.9999	0.4903	0.0856	0.3225	0.6582	5182.2190	0.9996
beta[2,4]	0.5245	0.0870	0.3494	0.6971	5290.2360	0.9994	0.5165	0.0861	0.3532	0.6857	5508.7380	0.9997
beta[3,1]	-0.3860	0.0786	-0.5400	-0.2281	5290.9620	0.9996	-0.3831	0.0812	-0.5409	-0.2213	5257.6250	1.0000
beta[3,2]	-0.5571	0.0823	-0.7158	-0.3948	6010.7610	0.9997	-0.5543	0.0813	-0.7152	-0.3925	7016.5330	0.9997
beta[3,3]	-0.2942	0.0810	-0.4536	-0.1346	6248.2330	0.9999	-0.2946	0.0798	-0.4516	-0.1387	5405.4380	0.9997
beta[3,4]	-0.3323	0.0814	-0.4886	-0.1718	5049.9520	1.0009	-0.3305	0.0814	-0.4929	-0.1715	6658.1900	0.9993
beta[4,1]	0.4255	0.0812	0.2682	0.5835	7746.6380	1.0000	0.4192	0.0781	0.2659	0.5738	5869.2490	0.9994
beta[4,2]	0.2949	0.0791	0.1433	0.4511	6043.3130	0.9994	0.2920	0.0754	0.1453	0.4395	7771.3910	0.9997
beta[4,3]	0.3005	0.0783	0.1450	0.4529	7613.1050	0.9996	0.3013	0.0777	0.1498	0.4513	7654.2220	0.9993
beta[4,4]	0.5160	0.0772	0.3663	0.6677	7432.2490	0.9994	0.5131	0.0789	0.3647	0.6662	7322.3580	0.9996
beta[5,1]	-0.1139	0.0806	-0.2700	0.0404	5381.9490	0.9995	-0.1173	0.0776	-0.2646	0.0404	5753.0420	0.9992
beta[5,2]	-0.1099	0.0794	-0.2644	0.0431	5979.0560	0.9996	-0.1101	0.0821	-0.2713	0.0506	6110.4510	0.9997
beta[5,3]	-0.3420	0.0796	-0.5000	-0.1889	6398.8260	0.9995	-0.3415	0.0787	-0.4904	-0.1838	5693.9540	0.9993
beta[5,4]	-0.0981	0.0795	-0.2517	0.0593	6062.3670	0.9995	-0.0989	0.0797	-0.2559	0.0588	5569.1150	1.0002
beta[6,1]	0.3728	0.0810	0.2126	0.5347	5559.0460	0.9998	0.3679	0.0794	0.2136	0.5194	6571.1340	0.9993
beta[6,2]	0.4360	0.0784	0.2756	0.5851	5234.8140	0.9999	0.4362	0.0797	0.2796	0.5936	5835.0100	1.0001
beta[6,3]	0.4863	0.0805	0.3238	0.6461	5921.4790	0.9993	0.4864	0.0793	0.3328	0.6404	6124.5160	0.9998
beta[6,4]	0.4840	0.0803	0.3292	0.6438	5815.5720	1.0004	0.4820	0.0803	0.3258	0.6384	5593.8900	0.9996
beta[7,1]	0.0419	0.0772	-0.1055	0.1962	7597.5690	0.9999	0.0412	0.0773	-0.1138	0.1900	6230.7870	0.9997
beta[7,2]	-0.0489	0.0773	-0.1994	0.1050	6087.8920	0.9998	-0.0475	0.0807	-0.2075	0.1133	8191.5730	1.0002
beta[7,3]	-0.0109	0.0798	-0.1687	0.1486	6252.1640	1.0001	-0.0124	0.0786	-0.1653	0.1380	7066.2810	1.0004
beta[7,4]	0.0027	0.0797	-0.1476	0.1553	7676.6090	1.0000	0.0044	0.0788	-0.1499	0.1543	8314.2050	0.9995
beta[8,1]	0.0472	0.0797	-0.1091	0.2052	5150.2720	0.9993	0.0479	0.0810	-0.1112	0.2076	5550.7630	0.9991
beta[8,2]	0.0001	0.0815	-0.1627	0.1592	5150.5700	1.0008	0.0011	0.0810	-0.1606	0.1614	4476.2610	1.0007
beta[8,3]	0.0326	0.0817	-0.1250	0.1947	4443.1150	0.9992	0.0320	0.0814	-0.1278	0.1959	6496.9190	0.9996
beta[8,4]	-0.0360	0.0811	-0.1975	0.1218	5185.1820	0.9996	-0.0358	0.0826	-0.1933	0.1271	6300.7920	0.9995
beta[9,1]	-0.2551	0.0778	-0.4080	-0.1030	6448.1470	0.9996	-0.2533	0.0797	-0.4076	-0.0998	6506.1320	0.9993
beta[9,2]	-0.1431	0.0805	-0.3006	0.0181	6964.2860	0.9995	-0.1426	0.0792	-0.2967	0.0151	7046.0500	0.9998
beta[9,3]	0.0579	0.0805	-0.0975	0.2111	6708.3850	0.9992	0.0588	0.0815	-0.0989	0.2225	6806.5670	0.9996
beta[9,4]	-0.0095	0.0780	-0.1570	0.1427	6198.5590	1.0001	-0.0088	0.0792	-0.1662	0.1479	6097.6380	0.9997
beta[10,1]	-0.1769	0.0778	-0.3352	-0.0288	6973.1160	0.9991	-0.1761	0.0769	-0.3269	-0.0251	6352.0520	1.0000
beta[10,2]	0.0484	0.0779	-0.1026	0.2012	6120.8480	0.9995	0.0479	0.0769	-0.0998	0.1979	6115.9400	0.9999
beta[10,3]	-0.1461	0.0770	-0.2963	0.0023	6257.0190	0.9998	-0.1454	0.0790	-0.2978	0.0110	5968.0770	0.9991
beta[10,4]	-0.0343	0.0795	-0.1918	0.1205	6668.5170	1.0004	-0.0346	0.0775	-0.1857	0.1174	5356.9120	1.0005
beta[11,1]	0.0163	0.0786	-0.1397	0.1743	8019.0290	0.9999	0.0151	0.0764	-0.1384	0.1636	6162.8360	0.9999
beta[11,2]	0.1085	0.0770	-0.0427	0.2594	7518.8800	0.9996	0.1083	0.0774	-0.0411	0.2599	6637.5700	0.9996
beta[11,3]	0.0530	0.0771	-0.1005	0.2040	7808.0060	0.9992	0.0544	0.0767	-0.0942	0.2014	6649.5340	0.9991
beta[11,4]	0.1027	0.0764	-0.0487	0.2496	6758.7840	0.9995	0.1028	0.0783	-0.0491	0.2552	7262.6690	0.9997
beta[12,1]	-0.0599	0.0762	-0.2056	0.0881	6949.0480	0.9993	-0.0582	0.0767	-0.2105	0.0947	5685.8840	0.9993
beta[12,2]	0.0403	0.0778	-0.1089	0.1931	8529.0760	0.9995	0.0369	0.0796	-0.1191	0.1950	7154.8000	0.9998
beta[12,3]	-0.0760	0.0775	-0.2278	0.0734	8875.8220	0.9995	-0.0754	0.0774	-0.2289	0.0810	6292.9460	0.9995
beta[12,4]	-0.1046	0.0792	-0.2607	0.0481	8113.7680	0.9994	-0.1027	0.0767	-0.2589	0.0512	6947.4900	0.9997
beta[13,1]	0.0108	0.0770	-0.1381	0.1583	7394.0050	0.9994	0.0146	0.0777	-0.1362	0.1692	7337.0950	1.0004
beta[13,2]	0.0288	0.0763	-0.1202	0.1774	6465.4980	1.0001	0.0277	0.0769	-0.1230	0.1795	7683.7000	0.9996
beta[13,3]	-0.0167	0.0781	-0.1672	0.1398	6565.8760	0.9993	-0.0153	0.0791	-0.1694	0.1397	6398.8790	0.9996
beta[13,4]	0.0207	0.0803	-0.1360	0.1801	5839.4440	1.0000	0.0211	0.0792	-0.1347	0.1711	6029.3480	0.9999
beta[14,1]	0.0062	0.0782	-0.1488	0.1599	7321.1910	0.9995	0.0075	0.0793	-0.1507	0.1625	6480.6390	0.9998
beta[14,2]	-0.0977	0.0763	-0.2492	0.0522	7477.9610	1.0004	-0.0970	0.0766	-0.2451	0.0546	7833.3430	0.9994
beta[14,3]	0.0572	0.0762	-0.0911	0.2044	5816.2640	0.9997	0.0567	0.0763	-0.0898	0.2130	6217.7550	0.9993
beta[14,4]	-0.0050	0.0763	-0.1559	0.1425	6977.7380	0.9994	-0.0093	0.0761	-0.1517	0.1396	7568.0010	0.9993
beta[15,1]	0.0180	0.0746	-0.1281	0.1635	7400.4260	0.9996	0.0182	0.0765	-0.1291	0.1669	6234.6780	0.9993
beta[15,2]	-0.0067	0.0792	-0.1664	0.1509	7462.3400	0.9992	-0.0094	0.0773	-0.1631	0.1405	7329.8360	0.9994
beta[15,3]	0.1848	0.0791	0.0249	0.3392	6924.6970	1.0006	0.1843	0.0780	0.0330	0.3368	6653.4260	0.9996
beta[15,4]	0.1931	0.0779	0.0436	0.3492	6138.2680	0.9997	0.1902	0.0775	0.0400	0.3433	7690.4380	0.9996
beta[16,1]	-0.0516	0.0783	-0.2036	0.1033	5710.9160	0.9996	-0.0549	0.0775	-0.2087	0.0935	5839.1160	0.9999
beta[16,2]	-0.1785	0.0805	-0.3414	-0.0202	8070.8120	0.9993	-0.1772	0.0818	-0.3359	-0.0122	6018.2570	0.9999
beta[16,3]	-0.2195	0.0801	-0.3739	-0.0588	7121.3580	0.9995	-0.2192	0.0784	-0.3763	-0.0662	6669.3880	0.9996
beta[16,4]	-0.0877	0.0782	-0.2392	0.0626	6065.3950	0.9993	-0.0877	0.0773	-0.2385	0.0657	7233.6950	0.9994
beta[17,1]	0.0082	0.0781	-0.1455	0.1618	7429.1640	0.9993	0.0084	0.0769	-0.1447	0.1579	7020.3060	0.9992
beta[17,2]	-0.0746	0.0780	-0.2255	0.0768	7111.1690	1.0000	-0.0734	0.0760	-0.2222	0.0727	5860.4100	0.9995
beta[17,3]	-0.0101	0.0771	-0.1640	0.1419	7010.7420	0.9993	-0.0107	0.0784	-0.1636	0.1460	7678.9740	0.9996
beta[17,4]	-0.0266	0.0776	-0.1831	0.1280	6644.8500	1.0002	-0.0263	0.0812	-0.1838	0.1290	6947.9700	0.9992
beta[18,1]	0.0443	0.0778	-0.1089	0.1941	7852.1080	0.9998	0.0433	0.0768	-0.1077	0.1923	7395.8350	0.9992
beta[18,2]	-0.0044	0.0778	-0.1561	0.1507	7108.5210	0.9998	-0.0062	0.0781	-0.1599	0.1455	8101.1360	0.9991
beta[18,3]	-0.0644	0.0787	-0.2178	0.0924	8285.0900	0.9992	-0.0660	0.0784	-0.2221	0.0842	7050.6930	0.9998
beta[18,4]	0.1016	0.0792	-0.0502	0.2541	6004.2690	1.0004	0.1005	0.0776	-0.0492	0.2492	7015.0580	1.0002
beta[19,1]	0.0736	0.0768	-0.0738	0.2182	8456.3390	0.9996	0.0735	0.0778	-0.0785	0.2236	7653.7830	0.9996
beta[19,2]	0.1128	0.0765	-0.0383	0.2604	7938.5750	0.9993	0.1106	0.0777	-0.0412	0.2660	7035.0610	0.9994
beta[19,3]	-0.1483	0.0770	-0.3022	0.0026	7177.8900	0.9995	-0.1458	0.0764	-0.2979	0.0024	8252.9350	0.9998
beta[19,4]	0.0566	0.0775	-0.0941	0.2066	6718.8890	0.9999	0.0553	0.0739	-0.0885	0.2012	7169.7330	0.9998
beta[20,1]	0.2079	0.0781	0.0564	0.3579	7727.6030	0.9998	0.2066	0.0768	0.0509	0.3489	7764.1850	0.9992
beta[20,2]	0.0928	0.0773	-0.0598	0.2434	8118.2380	0.9994	0.0907	0.0756	-0.0519	0.2393	7810.6430	1.0001
beta[20,3]	0.0716	0.0750	-0.0741	0.2154	8355.2890	0.9996	0.0723	0.0790	-0.0803	0.2287	6544.2800	0.9992
beta[20,4]	0.0393	0.0771	-0.1101	0.1929	6553.0940	1.0002	0.0385	0.0775	-0.1148	0.1880	7535.3990	0.9995
delta[1,1]	-0.0837	0.0564	-0.1973	0.0254	1885.9070	1.0014	-0.0775	0.0564	-0.1868	0.0328	1700.5340	1.0007
delta[1,2]	-0.1422	0.0571	-0.2541	-0.0312	1718.7970	1.0014	-0.1433	0.0569	-0.2534	-0.0320	1874.8870	0.9999
delta[1,3]	0.1079	0.0572	-0.0049	0.2161	1339.1710	1.0005	0.1062	0.0561	-0.0048	0.2112	1944.1410	1.0010
delta[1,4]	-0.0586	0.0581	-0.1712	0.0523	1414.7980	1.0018	-0.0565	0.0551	-0.1647	0.0522	1954.7710	1.0010
delta[2,1]	-0.0205	0.0165	-0.0527	0.0124	5673.6140	1.0006	-0.0192	0.0163	-0.0503	0.0124	4651.5390	0.9995
delta[2,2]	-0.0248	0.0163	-0.0568	0.0068	5691.0430	0.9997	-0.0247	0.0160	-0.0557	0.0073	5391.8790	1.0008
delta[2,3]	0.0120	0.0159	-0.0193	0.0429	4712.2350	0.9996	0.0117	0.0163	-0.0206	0.0441	4804.4470	0.9994
delta[2,4]	-0.0224	0.0162	-0.0539	0.0088	4718.0570	0.9998	-0.0217	0.0161	-0.0537	0.0090	5803.8690	0.9996
delta[3,1]	-0.0098	0.0118	-0.0322	0.0136	2408.9620	1.0005	-0.0090	0.0115	-0.0315	0.0139	2539.0320	0.9996
delta[3,2]	0.0494	0.0115	0.0266	0.0720	2583.8540	1.0010	0.0491	0.0117	0.0257	0.0719	2522.0580	1.0009
delta[3,3]	0.0292	0.0118	0.0060	0.0521	1881.5860	1.0007	0.0291	0.0115	0.0066	0.0520	2951.5410	1.0000
delta[3,4]	0.0036	0.0118	-0.0197	0.0265	2289.4780	1.0012	0.0041	0.0114	-0.0179	0.0268	2974.5740	1.0000

Table 3: Parameter estimates for the separate model with narrow and wide priors.

	Separate Model with wide priors						Separate Model with narrow priors					
parameter	mean	sd	2.50%	97.50%	n_eff	Rhat	mean	sd	2.50%	97.50%	n_eff	Rhat
sigma_obs	0.621421954	0.0237098	0.57775831	0.66922105	5242.2062	0.9995554	0.62163917	0.02370051	0.577542882	0.67002397	5284.2287	0.999397
m[1]	0.092754625	0.2698571	-0.436662974	0.622248992	792.5745	1.0012073	0.107205319	0.27142482	-0.433843927	0.625440393	700.5659	1.0037518
m[2]	0.218302637	0.27543758	-0.315153056	0.755934996	854.4363	1.0033249	0.232235598	0.27723076	-0.310133727	0.777550868	751.373	1.0049047
m[3]	0.15139931	0.27646443	-0.39324788	0.705503965	867.8862	1.0020873	0.142845608	0.27498814	-0.398954882	0.678232149	715.32	1.0041114
m[4]	0.231853677	0.27598529	-0.3310606	0.763081856	818.6011	1.0033453	0.231605625	0.26820588	-0.282964119	0.762367597	751.5045	1.0037825
k[1]	-0.018171191	0.08098387	-0.174237707	0.143344048	1046.1861	1.0011508	-0.022765303	0.08030521	-0.18211605	0.130198391	1005.7922	1.0022398
k[2]	0.091839042	0.08389041	-0.07031347	0.258660877	1105.3129	1.00312	0.08477041	0.08435476	-0.078530124	0.255634085	1099.6077	1.0009746
k[3]	-0.157598584	0.08560036	-0.324888993	0.011814103	1092.7428	1.0032763	-0.152440756	0.08411378	-0.325669289	0.006468993	1045.9837	1.0004234
k[4]	0.014608737	0.08297491	-0.142294175	0.181746206	1008.3754	1.003207	0.012070422	0.07963368	-0.143595886	0.164828653	1035.4219	1.0000872
beta[1,1]	-0.586885272	0.08138156	-0.747244919	-0.424397366	805.8527	1.0024274	-0.589939702	0.07958174	-0.744267832	-0.43158244	1007.1868	1.003171
beta[1,2]	-0.639118972	0.08253912	-0.792322394	-0.478200478	880.3683	1.0026005	-0.64131692	0.08243087	-0.80462953	-0.483279683	917.2115	1.0034366
beta[1,3]	-0.460791984	0.08904907	-0.628638232	-0.281094968	848.4983	1.0036397	-0.463285961	0.08326773	-0.626853079	-0.295327777	971.5372	1.002675
beta[1,4]	-0.657376101	0.08484214	-0.82698269	-0.491762457	885.9695	1.0034499	-0.658355198	0.08256124	-0.815684497	-0.494520539	1014.2737	1.0030149
beta[2,1]	0.35514959	0.20962577	-0.063975645	0.766442986	387.1277	1.0091096	0.357535608	0.20243471	-0.048816393	0.758458453	394.8199	1.0089599
beta[2,2]	0.296975391	0.20880779	-0.125228401	0.705995303	390.7977	1.009414	0.301314906	0.20185764	-0.099039925	0.696480928	412.5959	1.0091118
beta[2,3]	0.313417686	0.21011255	-0.101373969	0.726527584	383.1484	1.0096431	0.315909106	0.20292145	-0.087818549	0.719349992	398.3551	1.0086576
beta[2,4]	0.292909819	0.2085382	-0.13333458	0.694058944	397.1295	1.007734	0.297968706	0.20317489	-0.10817437	0.696533544	403.2158	1.0100857
beta[3,1]	-0.380269585	0.06608019	-0.512816703	-0.250473129	4256.7865	1.0003583	-0.378278937	0.06607518	-0.507932125	-0.250555588	4365.1353	0.999409
beta[3,2]	-0.37108535	0.06642001	-0.498470817	-0.241295908	5630.6883	0.9996828	-0.371325659	0.06580294	-0.505579819	-0.245937708	4313.5932	1.0002005
beta[3,3]	-0.329141957	0.06714009	-0.457553493	-0.194271579	5253.3254	0.9994407	-0.327202312	0.06498334	-0.456571483	-0.199644984	4903.6222	1.0000309
beta[3,4]	-0.306316083	0.06608525	-0.430096418	-0.173699456	4985.1126	1.0000555	-0.307040002	0.06643697	-0.435969185	-0.176783947	5160.9238	0.9998251
beta[4,1]	0.36168969	0.08253224	0.200530662	0.524873246	1545.6174	0.9998539	0.359276582	0.08226087	0.194117043	0.518079712	1268.021	1.0007391
beta[4,2]	0.327278317	0.08182871	0.167067787	0.485634079	1478.2401	1.0008394	0.324257801	0.08308162	0.167160998	0.486978101	1344.1898	1.0013204
beta[4,3]	0.265861091	0.08332767	0.103277056	0.427389665	1272.609	1.0009499	0.267439731	0.08388072	0.095373238	0.426680542	1198.5291	1.0014738
beta[4,4]	0.43710927	0.08480091	0.274556133	0.604776267	1369.9628	1.0013762	0.433834109	0.08369977	0.270563874	0.599938476	1183.4995	1.0005041
beta[5,1]	-0.151355453	0.06722883	-0.281453075	-0.020849259	3567.5641	1.0003555	-0.150191932	0.0666103	-0.27858181	-0.013868038	3254.3303	1.0005415
beta[5,2]	-0.138368712	0.06481226	-0.26281224	-0.010490234	3190.3529	0.999873	-0.13997284	0.0666623	-0.26986631	-0.006315755	3527.6925	0.9999991
beta[5,3]	-0.203336131	0.06587816	-0.332386961	-0.073521751	3742.9376	0.9999819	-0.203129702	0.06700881	-0.335093574	-0.072310666	3749.4836	0.999874
beta[5,4]	-0.156666919	0.06547076	-0.282360677	-0.02832219	3378.0039	0.9998625	-0.156220967	0.06748336	-0.288801729	-0.025820652	4019.6543	1.000712
beta[6,1]	0.26378816	0.09332816	0.075617127	0.448333016	593.9718	1.0074282	0.267195882	0.09117168	0.086570224	0.446011083	629.9743	1.0049485
beta[6,2]	0.313373885	0.09245801	0.127232406	0.487765637	587.1235	1.0060799	0.315096827	0.09083894	0.139473714	0.494580525	658.8949	1.0056331
beta[6,3]	0.371719638	0.0933287	0.189048743	0.550552876	599.3853	1.005917	0.370560251	0.09147614	0.189636298	0.54622242	658.2305	1.0064667
beta[6,4]	0.362538578	0.09441177	0.17512591	0.545260053	598.1552	1.0050535	0.362480444	0.09121822	0.181945366	0.539951703	634.7838	1.0054305
beta[7,1]	-0.007568598	0.06500482	-0.134544306	0.118412202	4428.9781	1.000554	-0.006359195	0.06648619	-0.137360357	0.121494713	5195.1265	0.9994803
beta[7,2]	-0.044450478	0.06651676	-0.171622801	0.088521293	5115.8139	0.9998694	-0.043078829	0.06330535	-0.1658226	0.08321621	3813.8309	0.9994916
beta[7,3]	-0.011564107	0.06713762	-0.147341053	0.117520506	5076.6503	0.9993832	-0.012185701	0.06557272	-0.142683377	0.119055437	5034.1874	0.9995537
beta[7,4]	-0.009029444	0.06760022	-0.140134216	0.125292883	5060.6785	0.9994753	-0.008040397	0.06659677	-0.14033694	0.118019196	4711.0968	0.9993125
beta[8,1]	0.000825089	0.07186256	-0.139784159	0.143256172	2830.0214	1.0009811	-0.000607798	0.06931342	-0.133076042	0.140961237	2226.4341	1.0012115
beta[8,2]	-0.02891065	0.07034112	-0.16765512	0.108890925	2411.7181	1.0011315	-0.030476513	0.07110055	-0.172040893	0.110674643	2431.0333	1.000444
beta[8,3]	-0.046423001	0.06936878	-0.181145769	0.093838796	2623.29	1.0008394	-0.045787224	0.07073395	-0.180595912	0.091477962	2303.9201	1.000022
beta[8,4]	-0.042250298	0.07145859	-0.181846259	0.095370992	2660.9846	1.0011802	-0.041337026	0.06974001	-0.176807326	0.095765504	2314.7563	1.0018346
beta[9,1]	-0.13849462	0.0685863	-0.274281961	-0.006971297	3640.5405	1.0006473	-0.139146052	0.06881328	-0.275876536	-0.009655961	3716.9775	1.0004861
beta[9,2]	-0.087643226	0.06660158	-0.217755729	0.045478427	4484.9511	1.0000079	-0.088527805	0.06721727	-0.222415465	0.043559918	4393.9505	0.9998115
beta[9,3]	0.00151209	0.06680701	-0.129997943	0.13350358	3173.1422	0.9998313	-0.000383579	0.06680756	-0.131161699	0.131521353	3901.8391	1.0004567
beta[9,4]	-0.036694587	0.06753398	-0.166922297	0.100772573	3773.4735	1.0003591	-0.037548799	0.06409341	-0.163391915	0.089656165	3907.672	1.0005174
beta[10,1]	-0.247591708	0.07938804	-0.40069119	-0.088790343	1024.9641	1.0042067	-0.246144499	0.07853636	-0.401005233	-0.092058251	1092.0496	1.0028917
beta[10,2]	-0.120269094	0.07591383	-0.269306985	0.02563025	1031.6071	1.0028752	-0.118496738	0.07746258	-0.270301788	0.036615099	1008.1974	1.0028574
beta[10,3]	-0.154453301	0.07650918	-0.306189373	-0.004465106	1040.2986	1.0034383	-0.154159457	0.07563457	-0.301301658	-0.005661951	989.6653	1.0036018
beta[10,4]	-0.135470715	0.07580064	-0.283250602	0.012310078	1162.6539	1.0021329	-0.133970409	0.0760693	-0.280064379	0.017061184	1091.3043	1.003474
beta[11,1]	0.059829504	0.06493214	-0.069127658	0.188002866	4988.2972	0.9996475	0.06017243	0.06672092	-0.075375028	0.192671262	6220.3577	0.9996861
beta[11,2]	0.062965156	0.06519925	-0.062239568	0.19378905	4919.8587	0.9998178	0.064292571	0.06646591	-0.063858851	0.194251133	5247.196	0.999471
beta[11,3]	0.022387711	0.06438958	-0.10771205	0.146156594	4527.9432	0.9997341	0.023462031	0.06484598	-0.102708561	0.151556029	4713.629	0.999681
beta[11,4]	0.075587853	0.0663665	-0.051473637	0.207998088	4920.6734	1.0000444	0.076450573	0.06670148	-0.053082235	0.208059196	4901.0784	0.9995969
beta[12,1]	-0.059770362	0.06748318	-0.196795219	0.07404382	3182.4138	0.9996253	-0.059177364	0.06764378	-0.191572054	0.074151751	3756.5777	0.9998114
beta[12,2]	-0.000312043	0.06981368	-0.136195843	0.141102431	3301.5144	1.0004962	0.000298009	0.06821229	-0.129107239	0.132273037	3045.146	1.0000383
beta[12,3]	-0.045389355	0.06830077	-0.180043646	0.08798033	3731.7993	0.9992184	-0.043500848	0.06957104	-0.181260126	0.089591236	3459.4003	0.9996429
beta[12,4]	-0.072244028	0.06898412	-0.211311853	0.064089452	3572.8112	0.9993029	-0.071279093	0.06848125	-0.206128265	0.060620164	3808.4983	1.0003483
beta[13,1]	0.009750362	0.06521193	-0.115933403	0.139205051	4154.401	0.9999475	0.009958875	0.06635767	-0.11898455	0.140170864	4437.5863	0.9994404
beta[13,2]	0.01157247	0.06726369	-0.118376529	0.142138196	4971.5816	0.9995163	0.009874835	0.06764313	-0.121727382	0.143878264	5007.4592	0.9995773
beta[13,3]	-0.026745414	0.06628416	-0.159671008	0.101651069	4718.376	1.000167	-0.02726525	0.06663495	-0.157892621	0.097989667	3887.5691	0.9996199
beta[13,4]	0.003667074	0.06656425	-0.127948268	0.133210986	4064.515	1.001213	0.003094286	0.06690287	-0.128535911	0.132976096	4563.0423	1.0000503
beta[14,1]	-0.051427105	0.07191728	-0.194441593	0.086997474	1501.1381	1.0022203	-0.050523212	0.07132541	-0.190506513	0.091192352	1729.695	1.0008629
beta[14,2]	-0.08302713	0.07263858	-0.225700482	0.063574834	1461.2881	1.0031976	-0.084068658	0.07300782	-0.223762006	0.0585302	1620.9067	1.0017077
beta[14,3]	-0.049315095	0.07085533	-0.187039881	0.09297921	1723.7526	1.0020709	-0.050095018	0.07104785	-0.186215241	0.08836873	1689.1923	1.0006564
beta[14,4]	-0.058069628	0.07019646	-0.195744291	0.0791001	1523.231	1.0018809	-0.057314058	0.07051209	-0.194053957	0.080828989	1591.3106	1.0019298
beta[15,1]	0.085075363	0.06748748	-0.046975659	0.215889576	4695.5346	0.9993411	0.083703363	0.06744446	-0.052651461	0.216347166	5257.8853	0.9997897
beta[15,2]	0.079475346	0.06728468	-0.052543646	0.210511417	4695.1548	0.9994409	0.079307758	0.06487974	-0.04800781	0.207560638	4581.3281	0.9994327
beta[15,3]	0.151320627	0.06756878	0.021754565	0.282076783	4387.8151	0.9996599	0.149576387	0.06875519	0.014574099	0.283146087	5163.5606	1.0000642
beta[15,4]	0.181912472	0.0658299	0.054719367	0.309823663	4289.5707	0.9997941	0.179483579	0.06631702	0.053097087	0.311062998	5331.5906	0.9997712
beta[16,1]	-0.132006253	0.0658728	-0.263245488	-0.004512927	3916.8617	1.0001073	-0.132394883	0.06525665	-0.262658058	-0.003115993	3985.1429	0.9995579
beta[16,2]	-0.187715119	0.06706313	-0.318491165	-0.061610607	4168.0834	0.9999244	-0.188246044	0.06650894	-0.318624767	-0.056724002	3320.6342	1.0006717
beta[16,3]	-0.188430114	0.06682238	-0.323338571	-0.05725556	4103.7681	0.9996873	-0.186805572	0.06647831	-0.319118691	-0.0561168	3705.332	0.9993612
beta[16,4]	-0.137537021	0.06561944	-0.266358851	-0.008650965	3843.9477	0.9995677	-0.137583981	0.06723346	-0.266809586	-0.001347271	4715.2292	1.0001495
beta[17,1]	0.00053701	0.06660227	-0.129192746	0.129539927	4027.5569	0.999759	0.000148156	0.06676332	-0.124747272	0.129208022	4462.9706	0.9997433
beta[17,2]	-0.010323222	0.06475696	-0.133369863	0.117406688	4834.9974	1.0001538	-0.011383164	0.06691108	-0.144175525	0.121484866	4820.315	0.9996149
beta[17,3]	-0.006337885	0.06532982	-0.135624764	0.119909972	4092.5404	1.0001564	-0.006487736	0.06654132	-0.134582597	0.125940424	4747.3928	1.0002879
beta[17,4]	-0.02827501	0.06599109	-0.156981255	0.102757515	4344.9844	0.9999691	-0.028126892	0.06712679	-0.160284808	0.106234036	5173.696	0.9993994
beta[18,1]	-0.00122008	0.06779102	-0.134297075	0.129035134	1976.3662	1.0004896	-0.002886911	0.07120186	-0.142628527	0.132976234	2242.1821	0.9999702
beta[18,2]	-0.003189109	0.0687135	-0.140740151	0.131239555	1973.7146	0.9998556	-0.002444479	0.06929933	-0.137279944	0.12955115	1831.2675	1.0015548
beta[18,3]	-0.06836084	0.07136906	-0.209249494	0.068770246	2007.2828	1.0005032	-0.067913914	0.07210882	-0.208741504	0.069544934	2214.128	1.0004135
beta[18,4]	0.050513416	0.07132768	-0.089073594	0.191533357	1852.0726	1.0003607	0.049972029	0.06989245	-0.088123992	0.188434881	2346.5811	1.001324
beta[19,1]	0.083739923	0.06513415	-0.042958133	0.213271405	4685.1447	0.9999669	0.084099933	0.06540905	-0.044438141	0.211466735	5067.7202	0.9997151
beta[19,2]	0.10358831	0.06699681	-0.025543069	0.235321	5283.9851	0.9999051	0.104549296	0.0657363	-0.023241006	0.237096462	4192.3342	0.9997705
beta[19,3]	-0.028605509	0.06902528	-0.163749376	0.10660103	4056.0125	0.9997005	-0.026833493	0.06737471	-0.162456841	0.101773713	3215.5768	0.9995254
beta[19,4]	0.074613074	0.06475142	-0.050308425	0.200581998	4787.1098	1.0002163	0.07561471	0.06438142	-0.048581346	0.201660895	5067.4057	0.9994663
beta[20,1]	0.13875391	0.06610737	0.007672712	0.267990083	3936.8033	0.9991949	0.137754328	0.06743354	0.007980949	0.273753819	3819.1695	1.0006044
beta[20,2]	0.097841968	0.06542402	-0.032697461	0.228333021	4208.908	0.9993043	0.098410496	0.06648945	-0.030919686	0.229733064	4122.6577	1.0003891
beta[20,3]	0.058823541	0.06468063	-0.071901326	0.185185199	4651.9434	0.9998176	0.060634862	0.06899895	-0.074616288	0.190668007	3918.7205	1.0009249
beta[20,4]	0.055811666	0.06540711	-0.077746067	0.181133449	4163.7654	1.0002393	0.056693617	0.06545337	-0.074036706	0.18513073	4356.0749	1.0006964
delta[1,1]	-0.047499147	0.07537547	-0.195976133	0.096171921	630.1801	1.0060227	-0.044159345	0.0732632	-0.186264102	0.100659596	707.3328	1.0047886
delta[1,2]	-0.141515825	0.07707448	-0.29715512	0.003104855	630.3901	1.0057587	-0.135314621	0.07544737	-0.283482347	0.009425573	734.4117	1.0030799
delta[1,3]	0.083067554	0.07892181	-0.072078365	0.23732247	669.414	1.0076954	0.078893741	0.07585935	-0.067374758	0.233551008	758.9641	1.0034545
delta[1,4]	-0.063286146	0.07599491	-0.213590267	0.08656308	606.2034	1.0057395	-0.060331982	0.07322	-0.204241202	0.082499349	691.801	1.0025855
delta[2,1]	0.089771689	0.04524057	0.000367203	0.180592168	443.9878	1.0082595	0.08974179	0.04440276	0.000953232	0.176568525	463.1862	1.008294
delta[2,2]	0.011869474	0.04583155	-0.077757365	0.101193112	455.7355	1.0063258	0.011396313	0.04420657	-0.074959428	0.099562202	458.0687	1.0089385
delta[2,3]	0.064767951	0.045469	-0.025116682	0.152901641	457.2966	1.00695	0.063830436	0.04475566	-0.026532388	0.150195577	468.3301	1.0087777
delta[2,4]	0.040420001	0.04574522	-0.048572162	0.131290067	454.4799	1.0061279	0.039546161	0.04452219	-0.045849749	0.130405521	459.0708	1.0083215
delta[3,1]	-0.050410656	0.02296776	-0.096299171	-0.005862867	511.3769	1.0081263	-0.04967805	0.02229921	-0.095475777	-0.004894876	519.4829	1.0054422
delta[3,2]	-0.005401156	0.02286505	-0.051922056	0.0378887	481.9078	1.0081899	-0.004501427	0.02220411	-0.046919621	0.040023572	542.3483	1.0041134
delta[3,3]	-0.011291191	0.02263569	-0.054380292	0.033475089	501.9387	1.0073502	-0.010928218	0.02223063	-0.056457548	0.031676818	503.9973	1.0071123
delta[3,4]	-0.02642184	0.02281688	-0.072552882	0.018125644	505.8903	1.0077649	-0.025687485	0.0222032	-0.069683616	0.017784283	527.3599	1.0087089

Table 4: Parameter estimates for the hierarchical model with narrow and wide priors.