Impact of Global Warming on Finland

Introduction

Today, one of the biggest threat to the society is global warming. Sea level rise is the immediate result of global warming as it melts the ice sheets and expands the water volume. Cities like Venice, New Orleans and Osaka are expected to experience major problems with the rise in sea level in near future.

Venice is already facing major problems.

Figure 1: Venice Floods

Goal of the project is to examine how much the sea level rise affects the coastal cities in Finland. For purpose of the study, four cities from coastal Finland were selected and the data is from January 1970 to November 2019 is used. Linear regression with Gaussian noise, $Error \sim N(0, \sigma^2)$, is fitted on the data to check the threat on Finnish cities.

The further sections discuss the data and its pre-processing followed by a discussion on priors, Stan models, convergence statistics and the results obtained. The report is concluded with the findings and possible future work.

Data

To find out about the sea level rise in Finland, hostorical data is needed about the same. In Finland, the Finnish Meteorologigal Institute provides open data about the weather. It is an easy portal to download the data needed for the study. The data from January 1970 to November 2019 is taken for the study. The cities are selected in a way that they are evenly distributed around the coast line so that the study covers the overall situation in Finland. The selected cities are Kemi, Turku, Helsinki and Oulu. The selected cities on map is shown in Figure 2.

The data is useful for weather analysis and is available openly so the results can be made public as the data is public.

Figure 2: Cities selected for the study

Format of Data

Data by Finnish Meteorologigal Institute is collected every hour for every day throughout the year, i.e. 24 * 30 * 365, readings are available for each city per year. A portion of this raw data is shown in Figure 3.

	A	B	C	D	E	F	G
1	Year	m	d	Time	Time zone	Water level (mm)	
2	1971	1	7	00:00	UTC	150	
3	1971	1	7	01:00	UTC	155	
4	1971	1	7	02:00	UTC	157	
5	1971	1	7	03:00	UTC	150	
6	1971	1	7	04:00	UTC	160	
7	1971	1	7	05:00	UTC	215	
8	1971	1	7	06:00	UTC	230	
9	1971	1	7	07:00	UTC	207	
10	1971	1	7	08:00	UTC	190	

Figure 3: Raw Data

Data pre-processing

Once the data is downloaded, it is essential to know the zero of the scale of measurement i.e. the height system in which the values is given. In Finland there are a number of systems available from which N2000 is used.

Thus, the data is normalized using N2000 data which is available on the Finnish Meteorologigal Institute website, N2000 data

Also, the data is too accurate so the mean of sea level for each year is calculated.

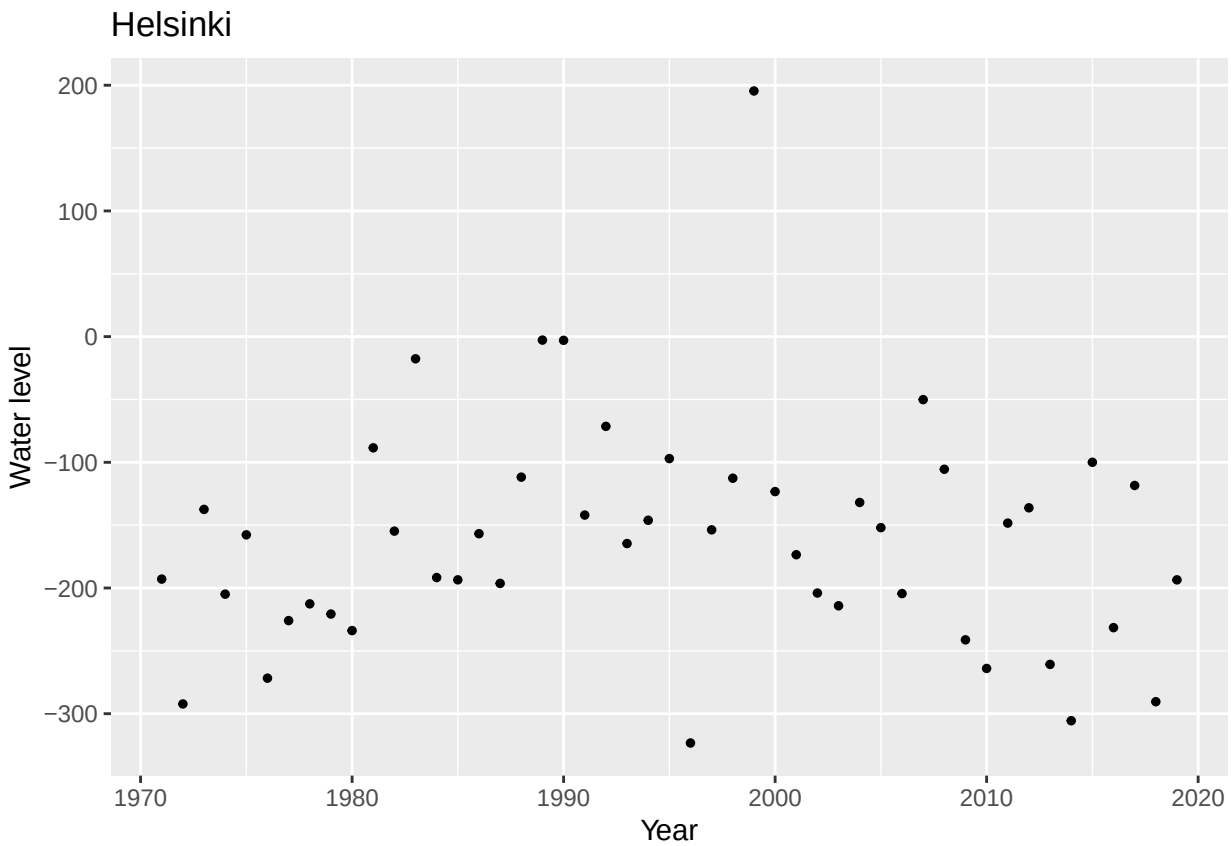
The whole data pre-processing script (written in Python) can be found in the Appendix section.

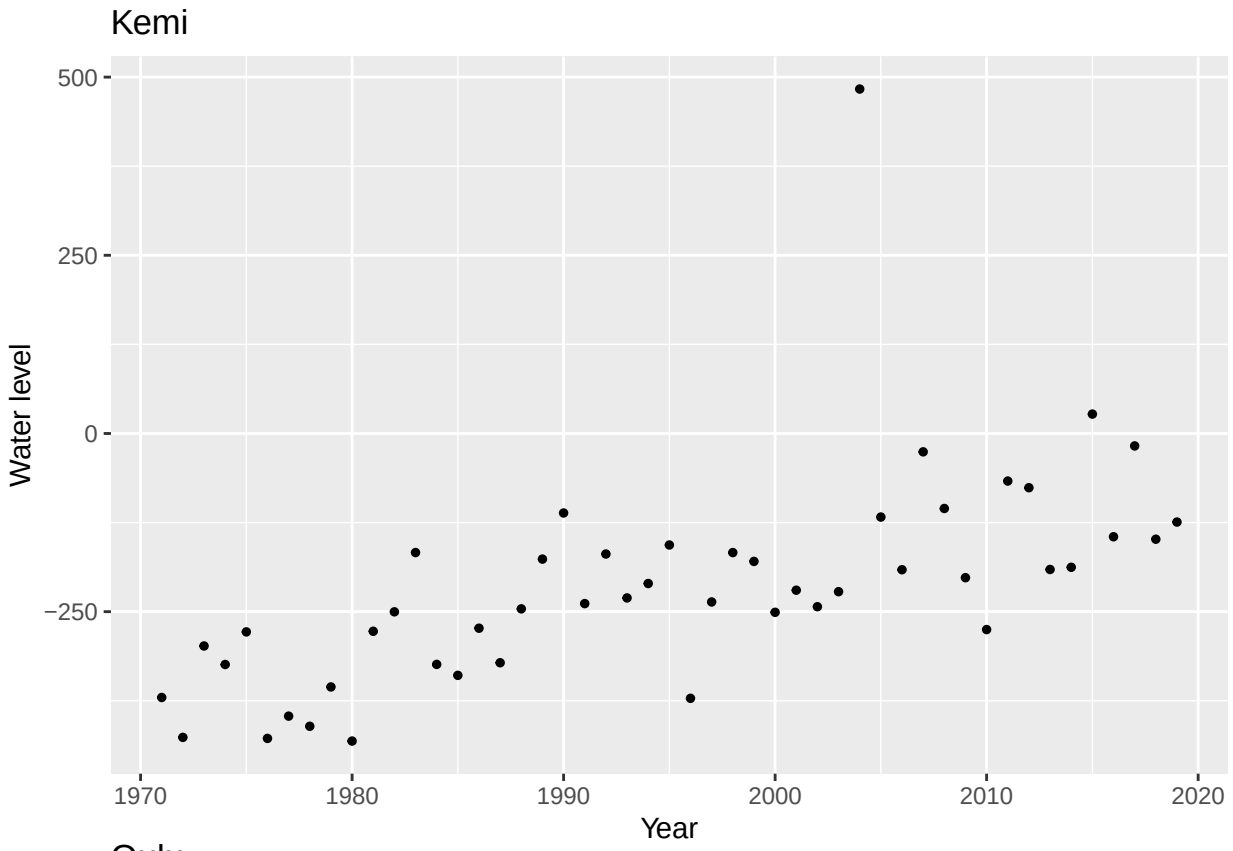
Processed Data

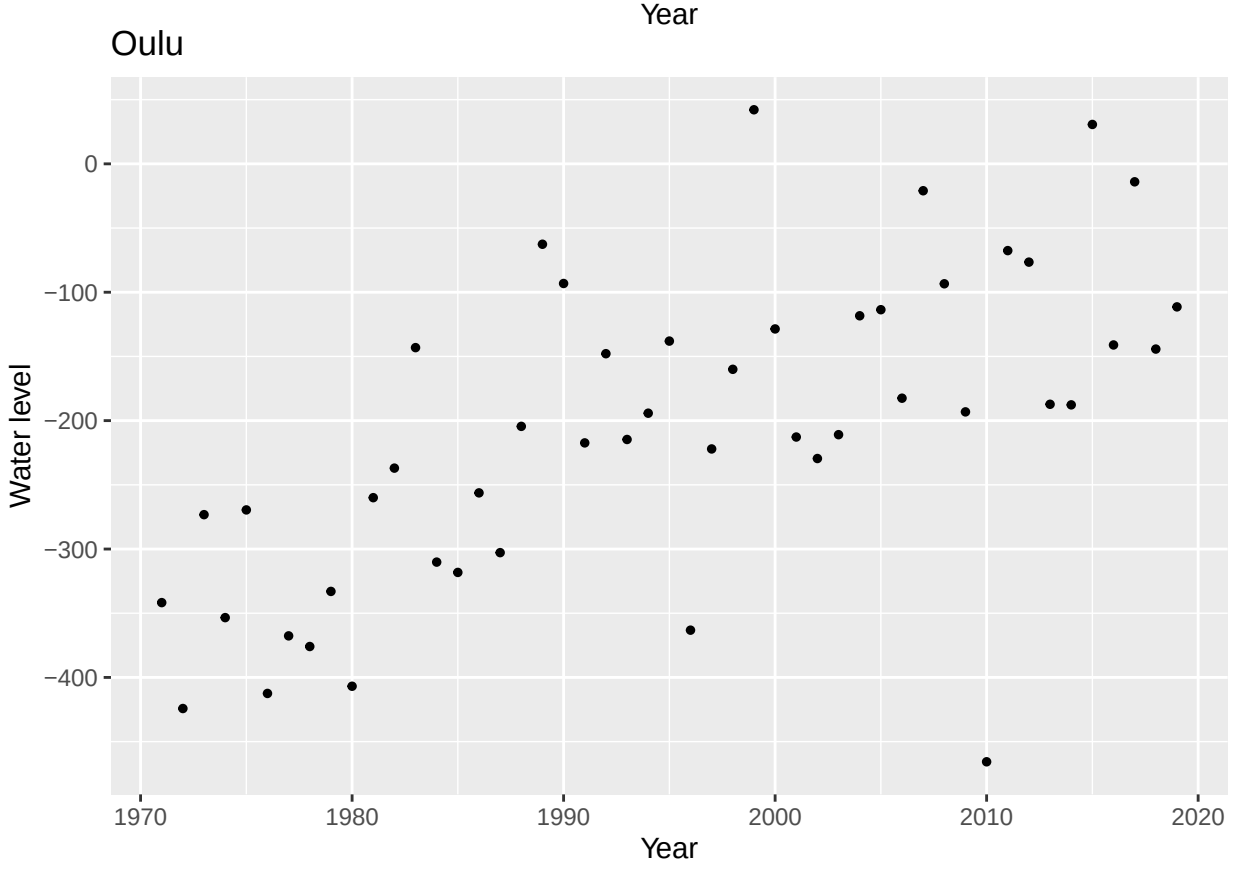
After pre-processing the data consists of one mean sea-level value per year. The example can be found in Figure 4. Visual representation of this processed data is plotted below for all the 4 cities.

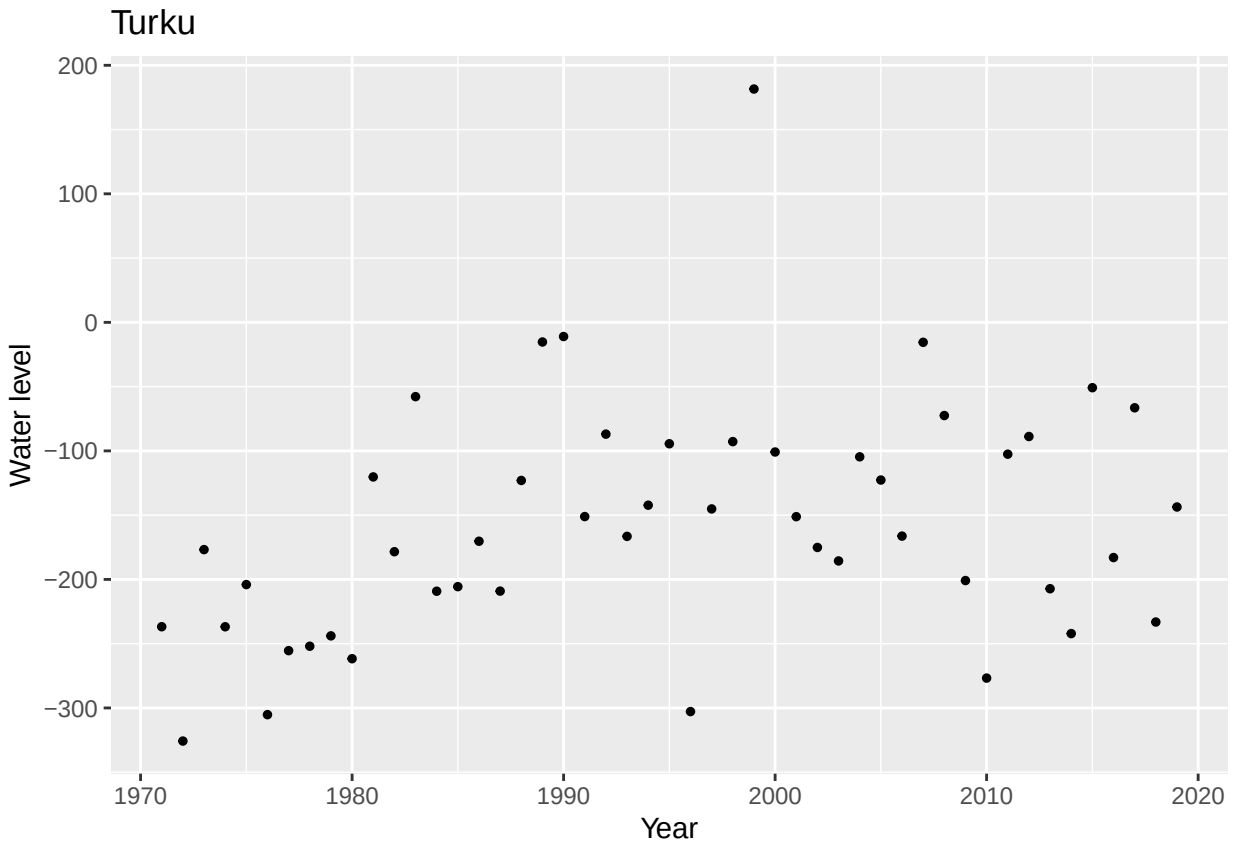
	A	B
1	1971	-341.740953063418
2	1972	-424.227342549923
3	1973	-273.2350853086
4	1974	-353.419487648673
5	1975	-269.536739380023
6	1976	-412.474499089253
7	1977	-367.63595890411
8	1978	-375.933789954338
9	1979	-333.019063926941
10	1980	-406.880009107468

Figure 4: Processed Data




Kemi
500
250
0
−250
Water level
1970
1980
1990
2000
2010
2020
Year


Oulu
0
−100
−200
−300
−400
Water level
1970
1980
1990
2000
2010
2020
Year



Project Workflow

In this section a general overview of what is done in the project is provided and it follows the Bayesian Work-flow. The steps are shown below:

- Define the Bayesian Models
- Perform sensitivity analysis of priors and choose appropriate prior(s)
- Define the models and fit the data
- Check Convergence Statistics
- Perform Posterior Predictive Analysis
- Visualize model through plots
- Perform PSIS LOO cross validation
- Perform Model evaluation

Models

A **gaussian linear model** to the time series data from 1970 to 2019 is fitted. Setting year 1970 as origin makes the slope parameter relative to year 1970 and the intercept modelled for year 1970. Since measurements are collected from 4 different geographical locations it is decided to try a **hierarchial model** in addition to a **separate model**.

The Guassian Linear Model is defined as : $y = \beta x + \alpha$ **where β is the slope, α is a constant and x is the year number.**

Different priors for both the models were tested which is described in the next section. Based on the results of these, suitable weakly informative priors are chosen for the models.

Sensitivity Analysis of Priors

Informative Priors

According to NASA, the global mean sea level rise is **3.3 mm** which can be used as a prior. However, it is a very informative prior so weakly informative priors were defined to fit the models.

Weakly Informative Priors

The weakly informative priors tested are:

1. $N(3.3, 100)$
2. $N(0, 100)$
3. $N(0, 10)$

However it was decided to use $N(0, 10)$ as the prior for both the models.

Model Fitting

Separate Model

A weakly informative prior which is $N \sim (0, 10)$ is being used, so 0 is the prior mean for the slope and 10 is the standard deviation. No prior is given to alpha because the slope should approximate the data as well as possible. Figure 5 shows the visual representation of the separate model.

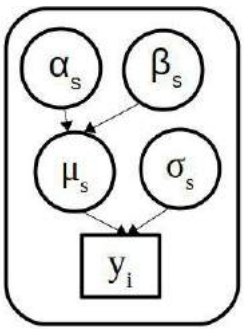


Figure 5: Separate Model

Priors

$p(\beta_s) \sim N(0, 10)$

Likelihood

$p(y|\theta) \sim N(\alpha_s + \beta_s(\mathbf{x} - 1970)|\sigma_s)$

Hierarchical Model

As data is available from 4 different locations it makes sense to try a hierarchial model. A prior distribution is used, based on the same global data as the seperate model $N \sim (0, 10)$. β_0 is used as a common mean for modelling the slope for each location. Then another hyperprior $p(\sigma_0) \sim N(0, 10)$ is used as a common variance for the slopes. Figure 6 shows the visual representation of the hierarchical model.

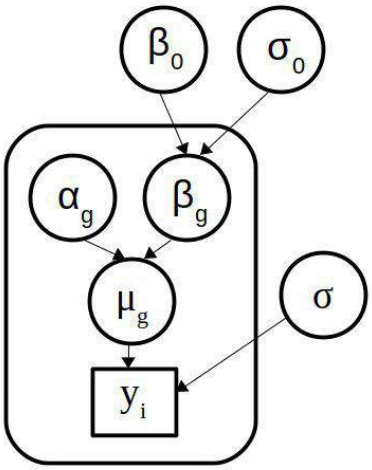


Figure 6: Hierarchical Model

Hyperpriors

$p(\beta_0) \sim N(0|10)$

$p(\sigma_0) \sim N(0|10)$

Priors

$p(\beta_g) \sim N(\beta_0|\sigma_0)$

Likelihood

$p(y|\theta) \sim N(\alpha_g + \beta_g(\mathbf{x} - 1970)|\sigma)$

Convergence Statistics in Stan

Variable	mean	se_mean	sd	2.50%	97.50%	n_eff	Rhat

Table 1: Sample table layout for convergence statistics

For each parameter (variable) of the model, the format as shown in Table 1 shows:

- The mean column shows the mean for each variable of all the posterior draws of that variable generated in Markov Chain Monte Carlo (MCMC) simulation.
- The se_mean column shows standard errors (se) for the posterior means. Standard error of the mean (SEM) depends both on the standard deviation (SD) and the sample size (n): SE = SD/$\sqrt{n}$.
- The sd column tells the standard deviation (SD). SD is a measure of variability or dispersion.
- The 2.5 % column reports the value corresponding the lower bound of 95% confidence interval.
- The 97.5 % column reports the value corresponding the upper bound of 95% confidence interval.
- The n_eff column tells the effective number of simulation draws, called the effective sample size, n_eff
- The Rhat column reports us the values of R$\hat{}$ that is our convergence statistic, R$\hat{}$. Here we will consider the threshold of 1.01 for the condition of R$\hat{}$ being 'near' 1 . If R$\hat{}$ > 1.01, we consider that the chains have not converged, while if R$\hat{}$ < 1.01 we consider that the chains have probably converged and estimates are reliable.

The Results section will show the details of the convergence statistics for both Separate and Hierarchical model.

The default tree depth in Stan is being used for HMC convergence diagnostic.

Leave One out cross validation

Leave-one-out cross-validation (LOO-CV or LOO) is a method to evaluate the predictive performance of fitted models for a data set. LOO-CV is an approach for estimating pointwise out-of-sample prediction accuracy from the fitted models using the log-likelihood assessed at the posterior simulations of the parameter values.

Here we use Pareto smoothed importance sampling (PSIS) LOO (PSIS-LOO) method for computing approximate LOO-CV given the posterior draws of the parameters. PSIS is a new approach that makes it possible to compute LOO using importance weights that would otherwise be unstable. PSIS fits a Pareto distribution to the upper tail of the distribution of the importance weights, and in this way provides relatively accurate and reliable estimate.

PSIS-LOO estimate is the sum of the LOO log predictive densities. The reliability of the PSIS-LOO estimates are assessed for a fitted model based on the k-values that are the estimated Pareto tail indices. The PSIS-LOO estimate can be considered reliable if all k-values are $k < 0.5$. Otherwise, we have a concern that the PSIS-LOO estimate may be biased i.e. it is possible that the estimate is too optimistic, overestimating the predictive accuracy of the fitted model.

When we are comparing different models which have the same target we should choose the model with the highest PSIS-LOO estimate. This is called elpd_loo which stands for "expected log predictive density" for the loo that is reliable i.e. all $k < 0.5$.

Results

Separate Model

Stan Code

```
data {
  int N;
  vector[N] y;
```

```
  vector[N] x;
  int k;
  int i[N];
}

parameters {
  vector[k] beta;
  vector[k] alpha;
  vector<lower=0>[k] sigma;
}

model {
  beta ~ normal(0,10);
  y ~ normal(alpha[i] + beta[i] .* (x-1970), sigma[i]);
}

generated quantities{
  vector[N] log_lik;
  vector[N] y_rep;

 for (j in 1:N) {
   log_lik[j] = normal_lpdf(y[j] | alpha[i[j]] + beta[i[j]] .* (x[j]-1970), sigma[i[j]]);
   y_rep[j] = normal_rng(alpha[i[j]] + beta[i[j]] .* (x[j]-1970), sigma[i[j]]);
 }
}
```

Convergence Statistics

Model convergences means that the sampling chains have merged and the samples start representing the sampled distribution. The R-hat values tells how well the model has convergenced with respect to a parameter. It generally should be 0.9 and 1.1. It can be seen from Figure 7, the R-hat values are all very close to 1 and therefore it can be said that the separate model has converged.

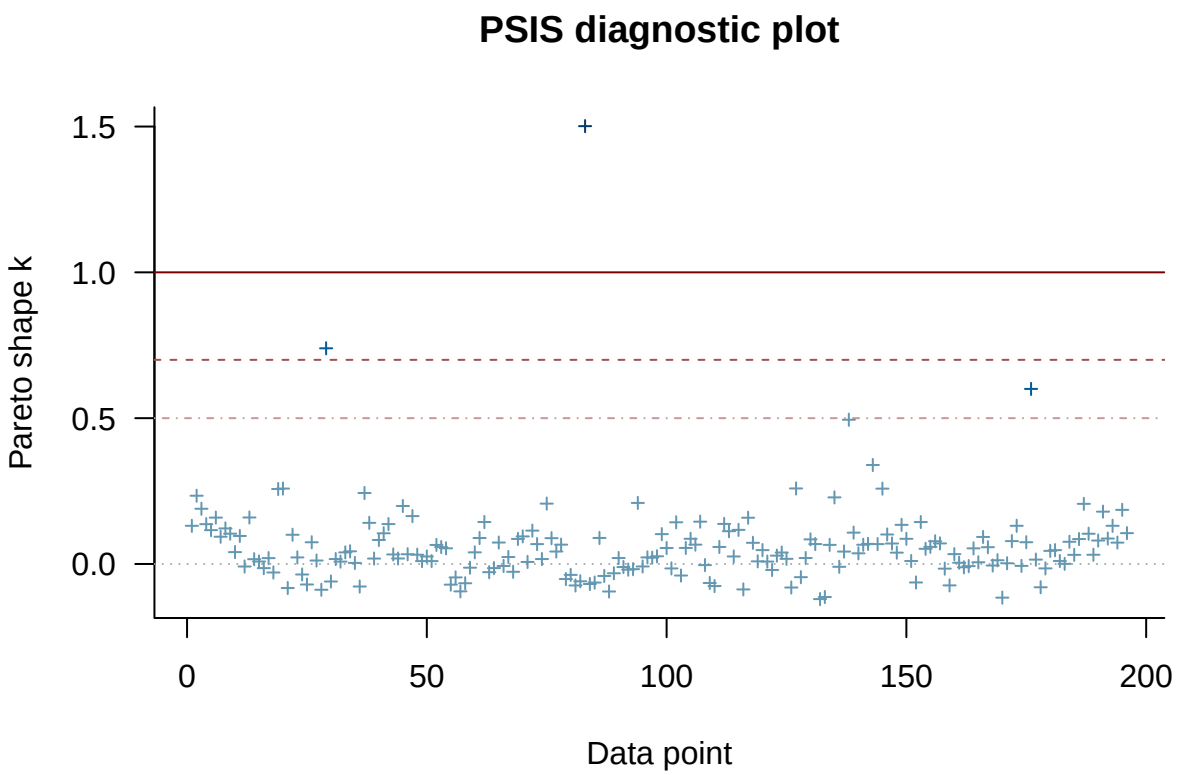
```
Inference for Stan model: separate.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

                mean se_mean     sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
beta[1]        -0.09    0.02   0.94   -1.92   -0.73   -0.08    0.53    1.78  3503    1
beta[2]         6.46    0.02   1.25    4.05    5.62    6.45    7.29    8.93  3813    1
beta[3]         5.36    0.02   0.98    3.43    4.70    5.37    6.03    7.25  3804    1
beta[4]         2.04    0.02   0.92    0.21    1.41    2.04    2.67    3.82  2877    1
alpha[1]     -159.75    0.46  27.23 -213.71 -178.53 -159.24 -142.01 -106.13  3551    1
alpha[2]     -375.36    0.58  36.15 -446.27 -399.99 -375.29 -350.53 -306.19  3929    1
alpha[3]     -342.75    0.46  28.25 -396.44 -361.91 -343.28 -323.92 -285.76  3788    1
alpha[4]     -207.86    0.46  26.56 -260.48 -225.88 -208.24 -190.02 -155.81  3319    1
sigma[1]       94.63    0.13   9.97   77.66   87.81   93.88  100.49  117.01  5584    1
sigma[2]      121.67    0.17  12.92   99.48  112.55  120.69  129.56  150.32  5491    1
sigma[3]       96.97    0.15  10.26   79.78   89.64   96.04  103.06  120.27  4454    1
sigma[4]       91.02    0.13   9.77   74.15   84.18   90.09   96.82  112.74  5361    1
```

Figure 7: Convergence Statistics for Separate Model

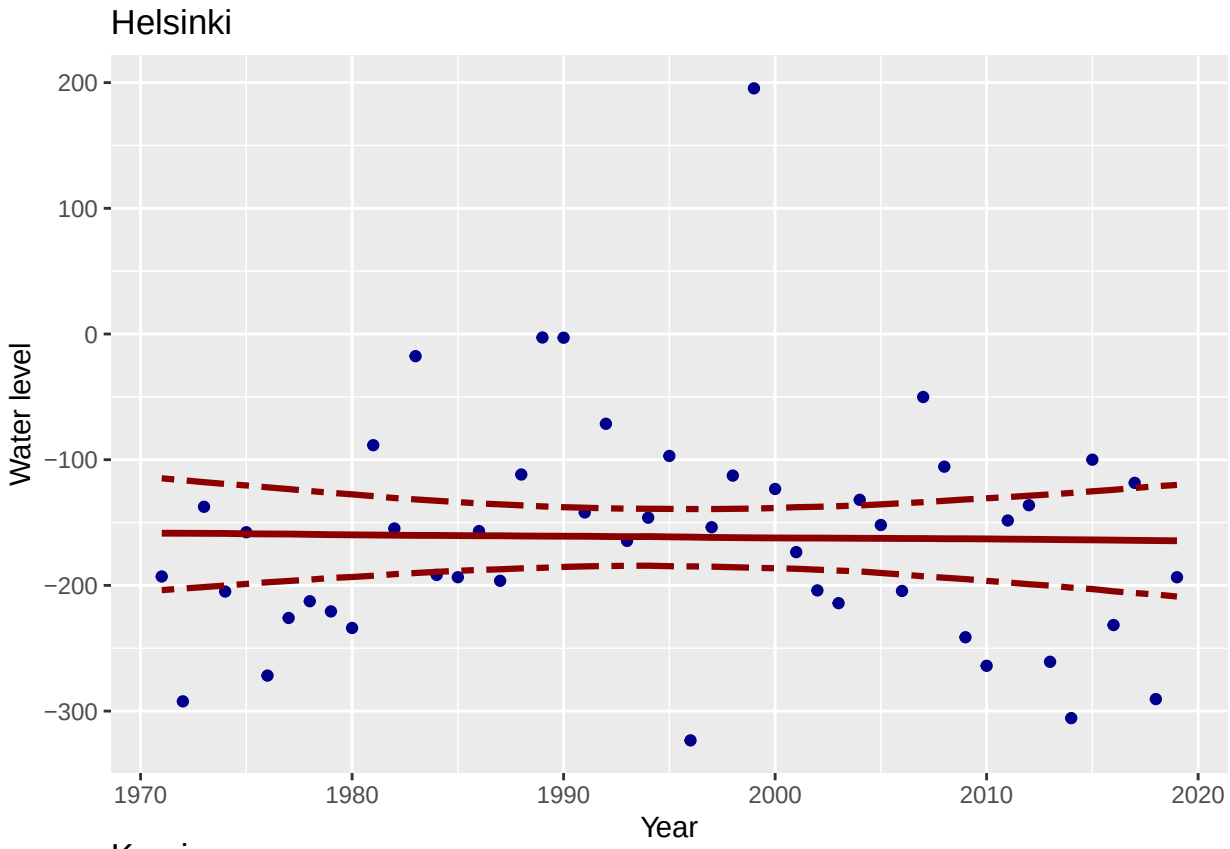
Loo

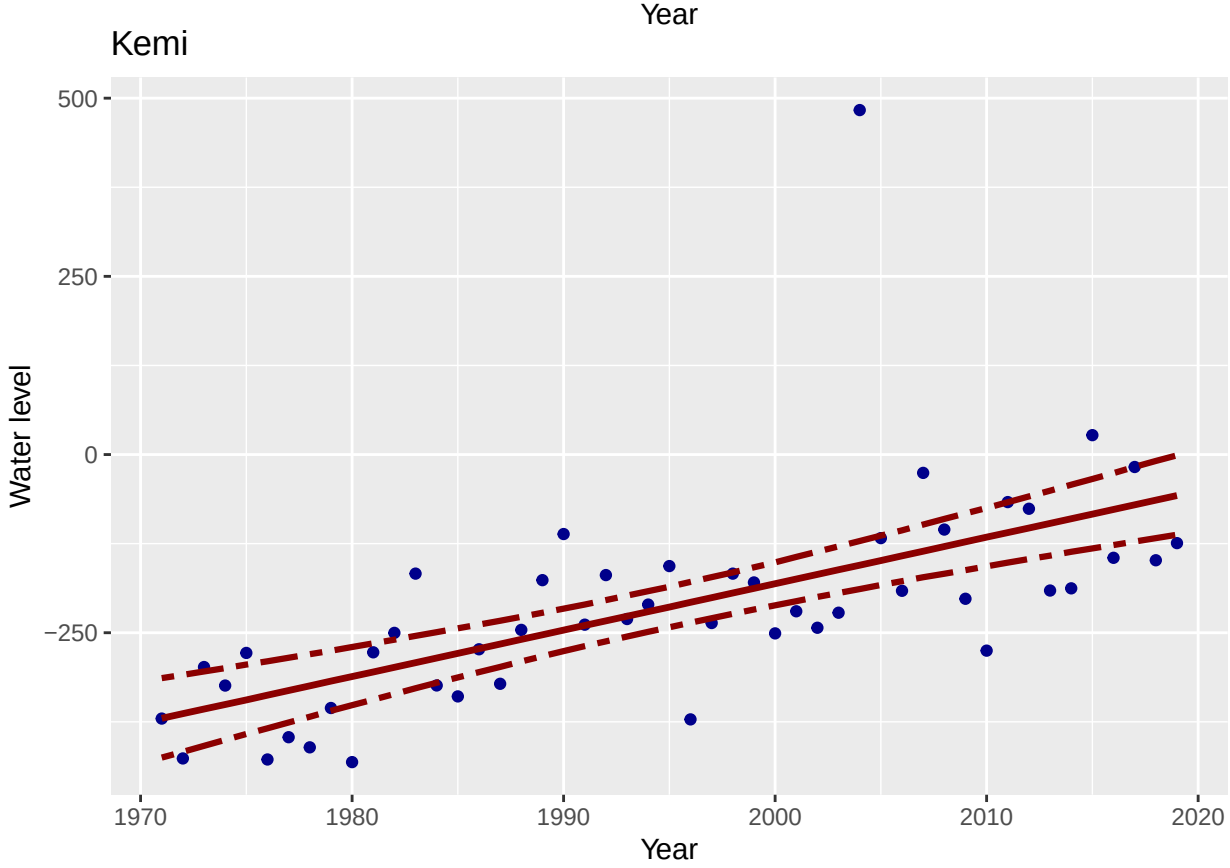
It can be seen from the values and plot below that almost all k values are less than 0.5 and a very few are above 0.5 which are biased. Although it is still a reliable model. Further hierarchical model is also calculated for comparison.

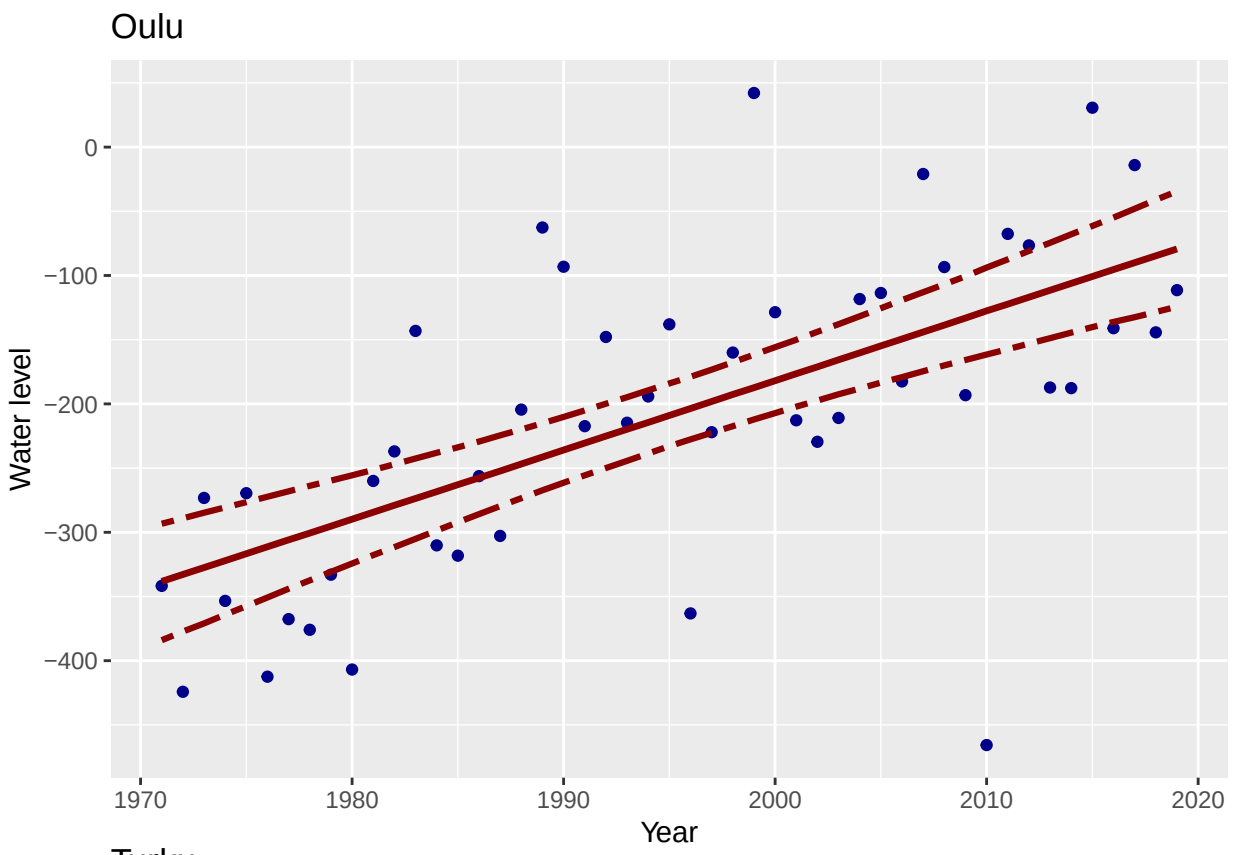
```
##
## Computed from 4000 by 196 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo  -1192.0 24.8
## p_loo        22.7 10.0
## looic      2384.0 49.7
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     193   98.5%   308
##  (0.5, 0.7]   (ok)         1    0.5%   244
##    (0.7, 1]   (bad)        1    0.5%   65
##    (1, Inf)   (very bad)   1    0.5%   5
## See help('pareto-k-diagnostic') for details.
```

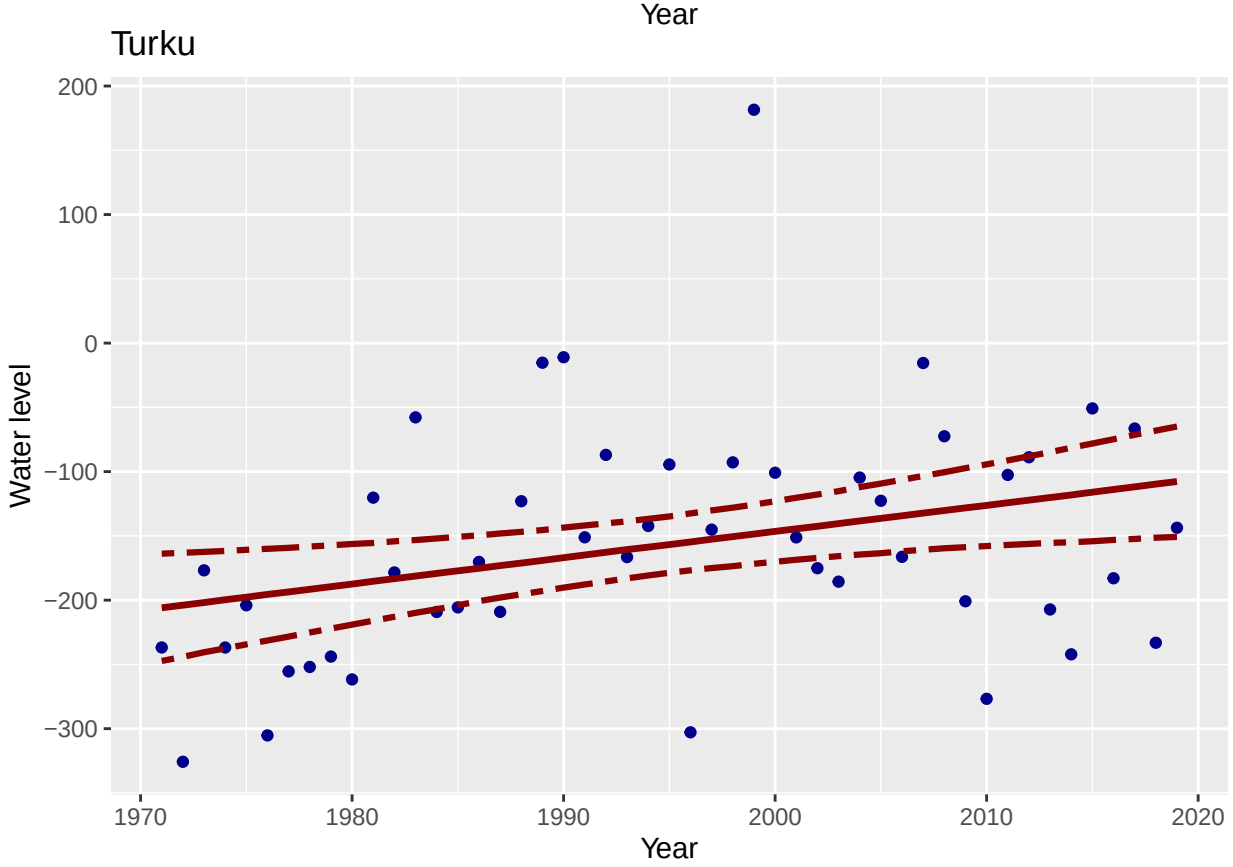


Plots after Stan fit






Oulu
0
−100
−200
−300
−400
Water level
1970
1980
1990
2000
2010
2020
Year


Turku
200
100
0
−100
−200
−300
Water level
1970
1980
1990
2000
2010
2020
Year

Hierarchical Model

Stan Code

```
data {
  int N;
  vector[N] y;
  vector[N] x;
  int k;
  int i[N];
}

parameters {
  vector[k] beta;
  vector[k] alpha;
  real <lower=0> sigma;
  real <lower=0> sigma0;
  real beta0;
}

model {
  beta0 ~ normal(0,10);
  sigma0 ~ normal(0, 10);
  beta ~ normal(beta0, sigma0);
  y ~ normal(alpha[i] + beta[i] .* (x-1970), sigma);
}

generated quantities{
  vector[N] log_lik;
  vector[N] y_rep;

 for (j in 1:N) {
   log_lik[j] = normal_lpdf(y[j] | alpha[i[j]] + beta[i[j]] .* (x[j]-1970), sigma);
   y_rep[j] = normal_rng(alpha[i[j]] + beta[i[j]] .* (x[j]-1970), sigma);
 }
}
```

Convergence Statistics

Model convergences means that the sampling chains have merged and the samples start representing the sampled distribution. The R-hat values tells how well the model has convergenced with respect to a parameter. It generally should be 0.9 and 1.1. It can be seen from Figure 8, the R-hat values are all very close to 1 and therefore it can be said that the hierarchical model has converged.

Loo

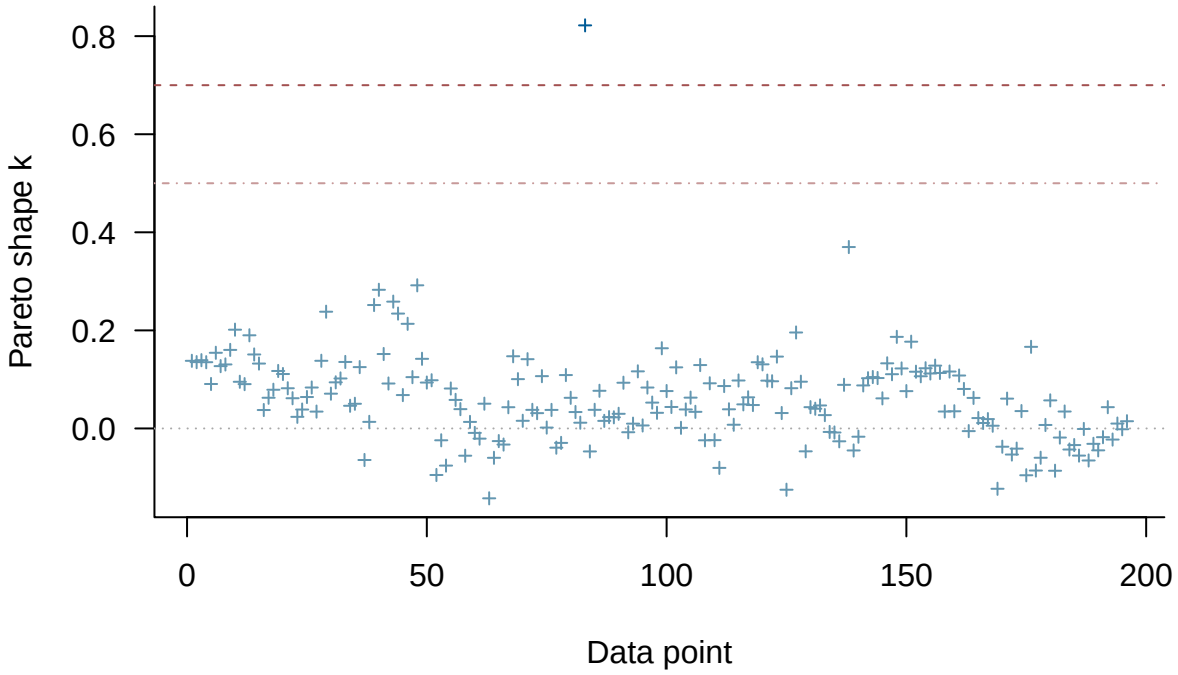
It can be seen from the values and plot below that almost all k values are less than 0.5 and almost no values are above 0.5. So it is a reliable model and performs marginally better than the Separate model computed above.

```
##
## Computed from 4000 by 196 log-likelihood matrix
##
```

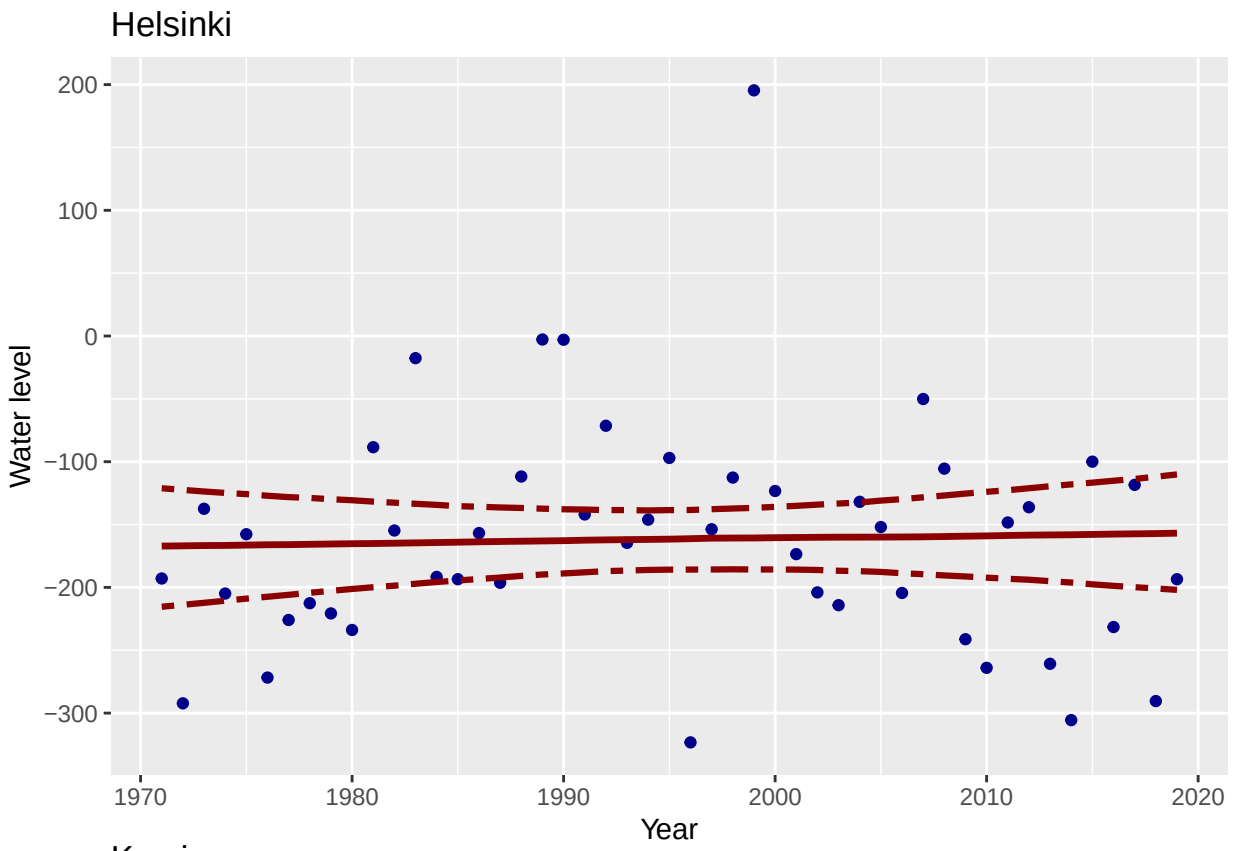
```
Inference for Stan model: hierarchical.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

                mean se_mean     sd     2.5%      25%      50%     75%   97.5% n_eff Rhat
beta[1]         0.19    0.02   1.03    -1.81    -0.48     0.21    0.87    2.20  3627    1
beta[2]         6.30    0.02   0.97     4.38     5.62     6.29    6.96    8.18  3808    1
beta[3]         5.26    0.02   0.98     3.38     4.56     5.25    5.92    7.13  3977    1
beta[4]         2.21    0.02   1.00     0.26     1.50     2.22    2.89    4.13  4058    1
alpha[1]     -166.57    0.52  29.23  -223.33  -186.35  -167.08 -146.86 -109.44  3182    1
alpha[2]     -370.86    0.44  27.90  -425.07  -390.11  -370.63 -351.86 -316.84  3959    1
alpha[3]     -339.81    0.47  28.16  -395.43  -357.67  -340.10 -320.57 -284.76  3592    1
alpha[4]     -211.60    0.45  29.23  -268.01  -231.56  -211.27 -190.96 -155.59  4241    1
sigma          99.77    0.07   5.13    90.40    96.26    99.55  103.09  110.21  5433    1
sigma0          4.47    0.05   2.53     1.53     2.72     3.79    5.49   10.98  2435    1
beta0           3.29    0.05   2.40    -2.12     2.05     3.40    4.61    8.08  2583    1
```

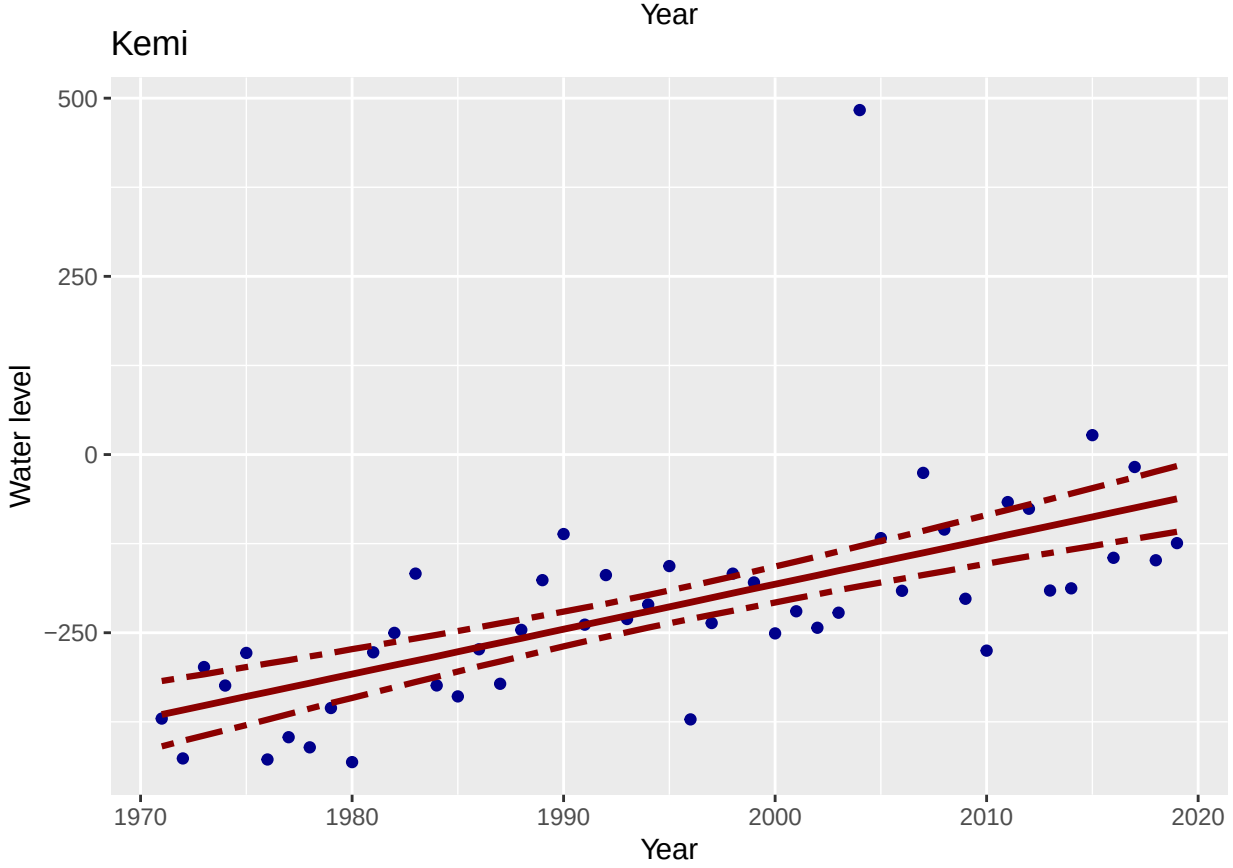
Figure 8: Convergence Statistics for Hierarchical model

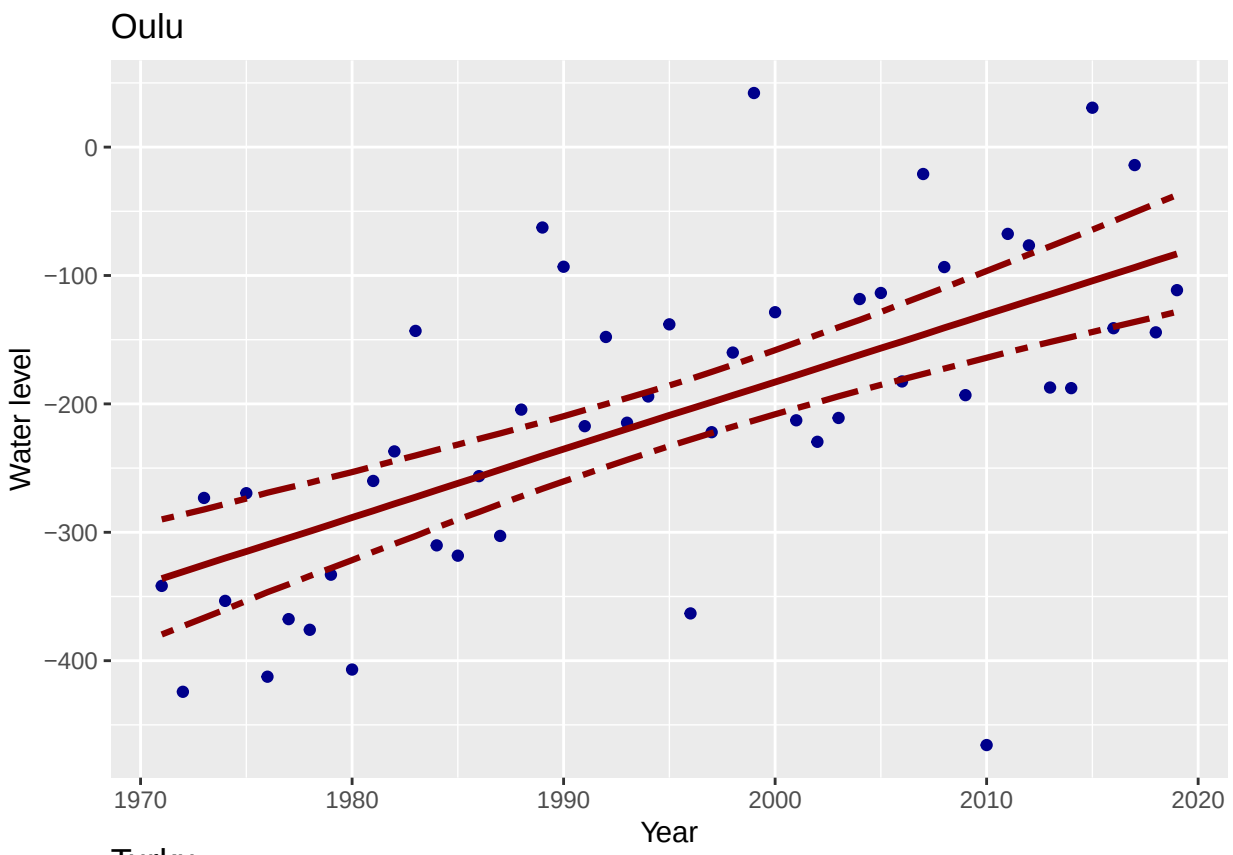
```
##          Estimate   SE
## elpd_loo  -1186.1 26.0
## p_loo        12.1  5.2
## looic      2372.2 52.1
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     195   99.5%   1252
##  (0.5, 0.7]   (ok)         0    0.0%   <NA>
##    (0.7, 1]   (bad)        1    0.5%   74
##    (1, Inf)   (very bad)   0    0.0%   <NA>
## See help('pareto-k-diagnostic') for details.
```

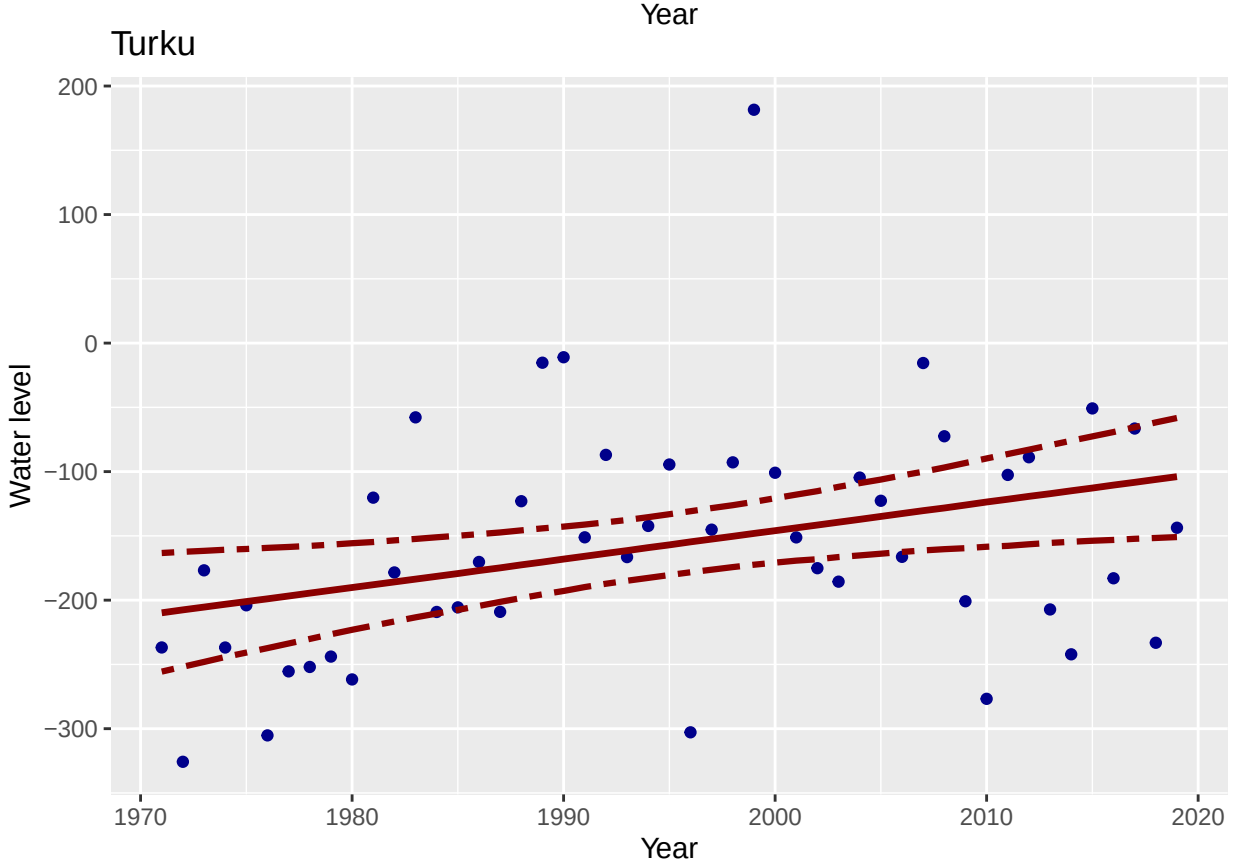

PSIS diagnostic plot
Pareto shape k
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
Data point

Plots after Stan fit






Oulu
0
−100
−200
−300
−400
Water level
1970
1980
1990
2000
2010
2020
Year


Turku
200
100
0
−100
−200
−300
Water level
1970
1980
1990
2000
2010
2020
Year

Effective Sample Size Diagnostic (ESS)

As it can be seen from the loo results table above, ESS, p-loo in table, for hierarchical model is around 12 whereas for separate it is around 23. Hence, hierarchical model performs better than separate model as it uses less number of parameters.

Model Comparison

Hierarchical model performs better than Separate model for the following reasons:

1. K-hat values in hierarchical model are very reliable
2. ELPD value is more for hierarchical model
3. Effective number of parameters p_loo is less for hierarchical model

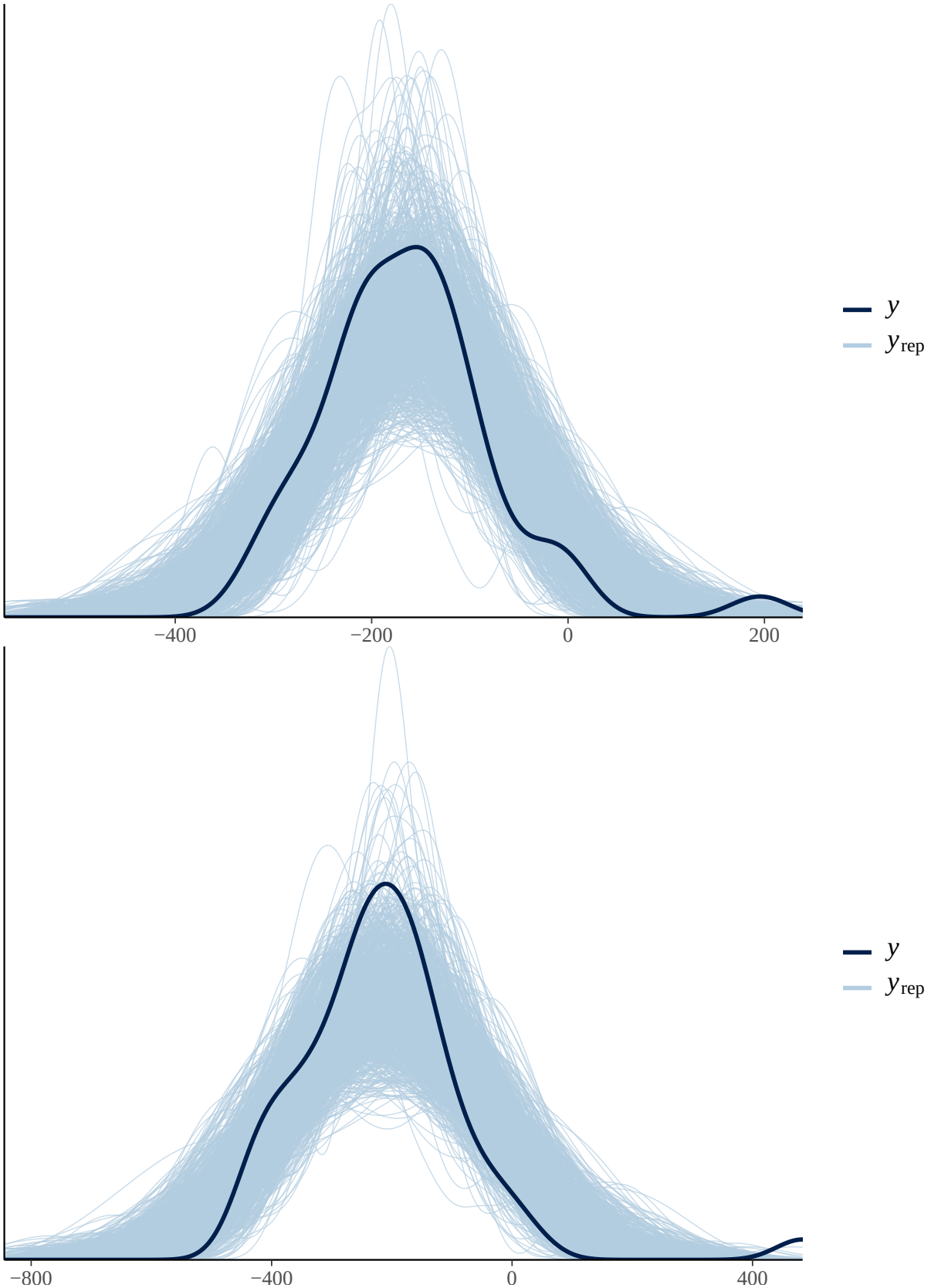
These results are separately printed above and below is the result obtained from the compare function in loo.

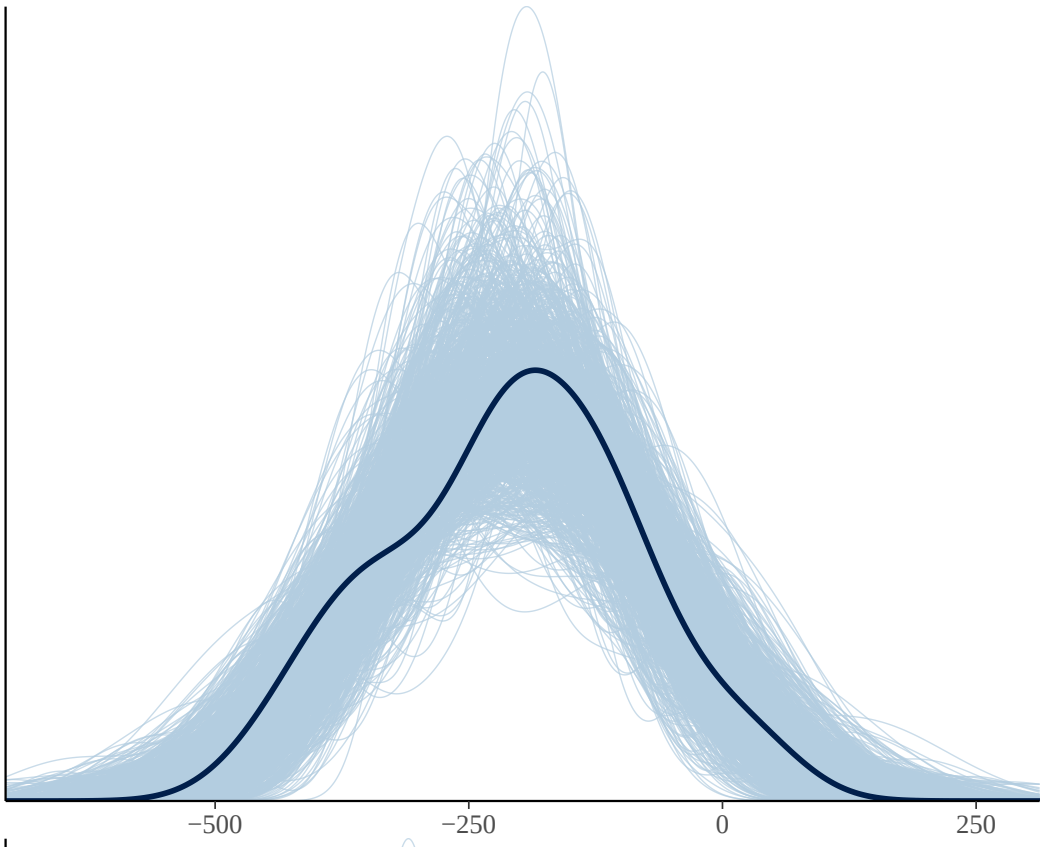
```
## elpd_diff        se
##      -5.9       4.7
```

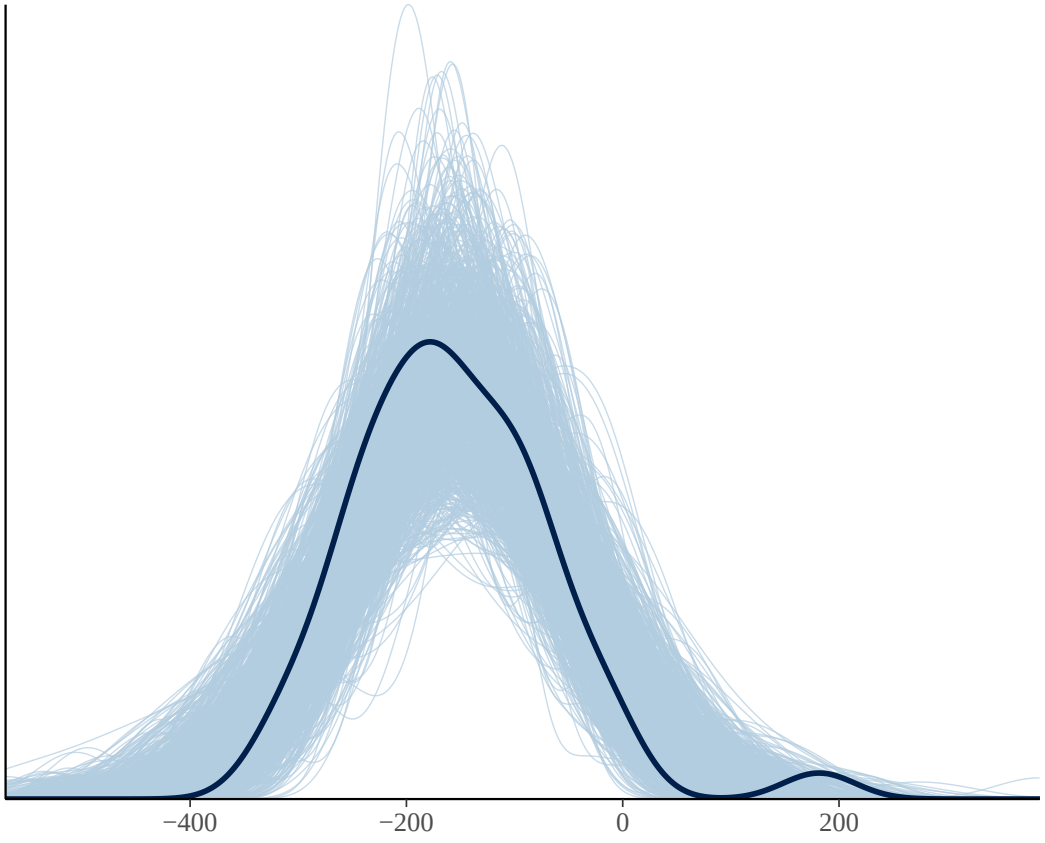
Posterior Predictive checking

Several replicated datasets are created and then these distributions are compared to the distribution of the data visually. It can be seen that the model fits the data quite well except around the positive values. Here the data is not normally distributed and there is a disproportionate amount of values around 0 compared to what a normal distribution would have. This could mean that the linear model is not optimal for this problem and that instead some form of exponential model should be used to capture this disproportionate amount of higher values. Based on this it might be possible that the sea level is increasing with an exponential trend in the Baltic Sea.

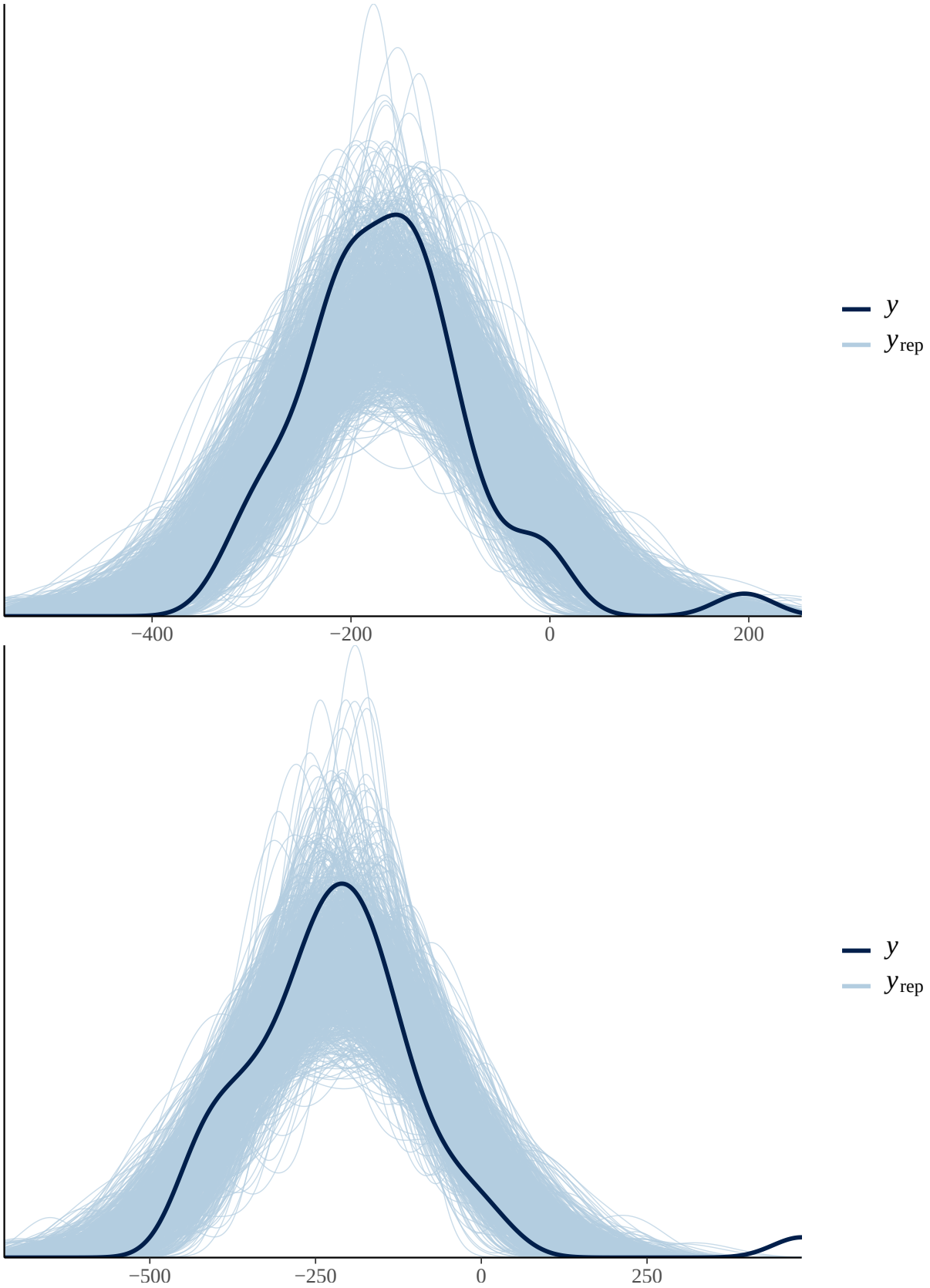
Poster Predictive Checking for Separate Mode for each city

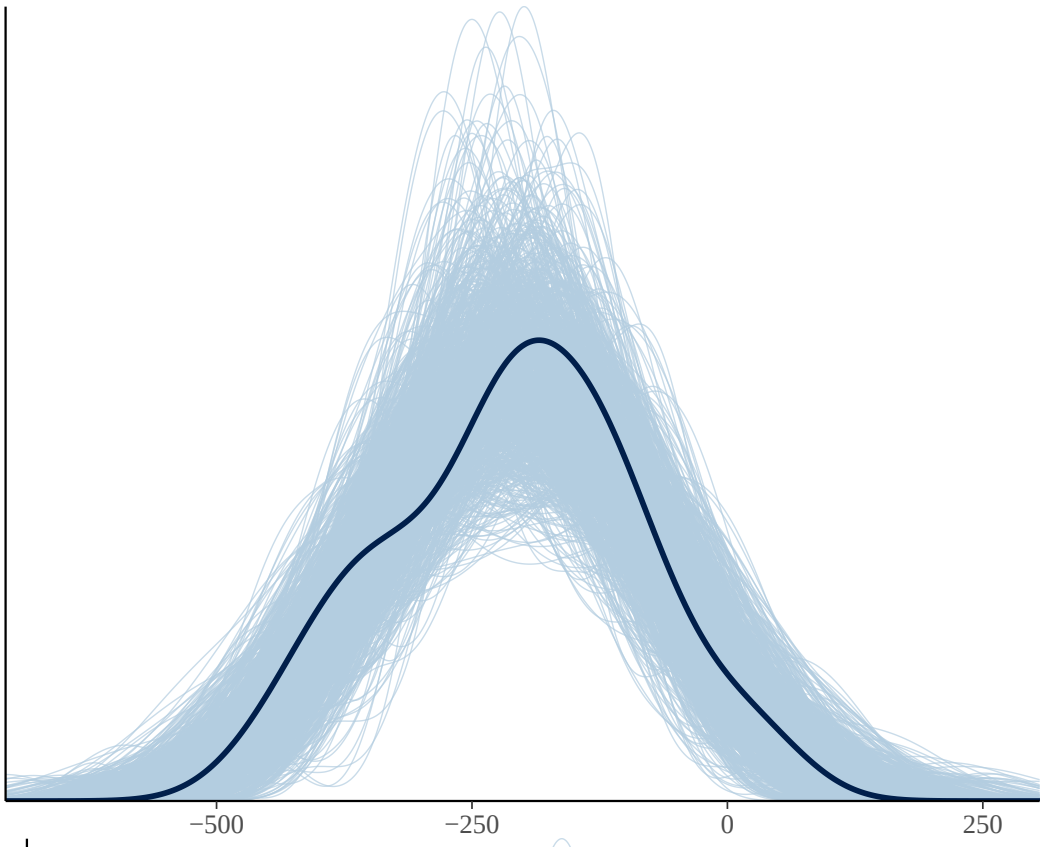


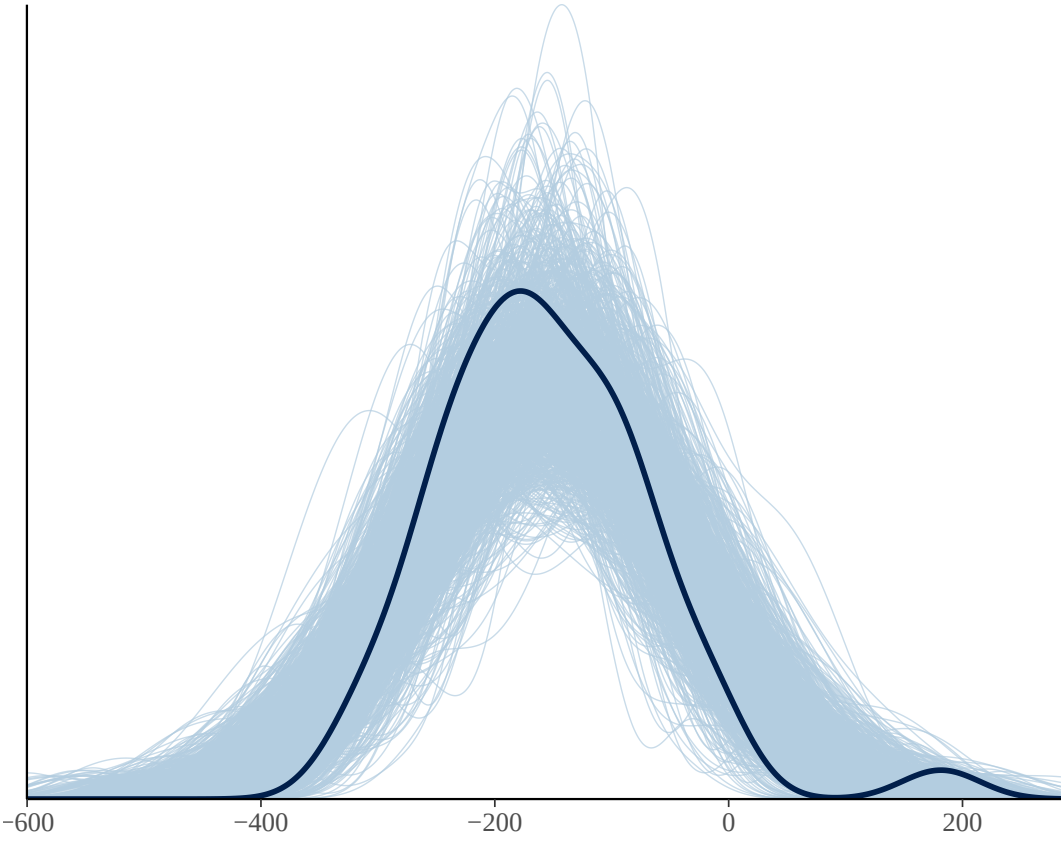

−500
−250
0
250
y
y rep


−400
−200
0
200
y
y rep

Poster Predictive Checking for Hierarchical Model for each city




−500
−250
0
250
y
y rep


−600
−400
−200
0
200
y
y rep

Conclusion

In Finland we are in lucky position that the changing conditions are not yet severe but the future changes are not yet known. The sea level rise in Finland is not a huge factor although in Gulf of Bothnia the sea level rises in some places 5 millimeters a year. However because of the Ice Age, the ground rises in that area almost 8 millimeters a year in that area. Also in southern Finland the Post-glacial rebound is about 4 to 5 millimeters a year and in the Gulf of Finland the sea levels are actually decreasing.

The fact that the sea level rise is not a huge factor in Finland does not, of course, mean that the climate change does not affect Finland and it can be ignored.

The summary of the results for various cities are shown in Figure 9.

CITY	MODELS		
		Hierarchical Mod	Separate Model
	Helsinki	0.19	-0.09
	Turku	2.21	2.04
	Oulu	5.26	5.36
	Kemi	6.3	6.46

Figure 9: Slope values for all cities from the two models

According to NASA, the average rise in water-level is 3.3 mm. Considering it, we can conclude Figure 10.

Helsinki	OK
Turku	OK
Oulu	Threat
Kemi	Threat

Figure 10: Analysis on the results obtained

Improvement to the Project

We settled into the linear regression quite early going into the project. We tried some polynomial fittings also but they started to over fit quite severely. To improve this project we could have maybe tried more to find better models to fit into our problem. However we are quite happy with the results we got out of the data.

Appendix

Data pre-processing script in Python

```
import os.path

import pandas as pd


def process_data(data_path, mean_sea_level_path, output_file_path, yearly_means_output_path):
    print(f"Pre-processing file: {data_path}")

    data = pd.read_csv(data_path)
    mean_sea_level_data = pd.read_csv(mean_sea_level_path)

    # if file exist then don't create a new one and read from it only
    if os.path.isfile(output_file_path):
        n_data = pd.read_csv(output_file_path)
        print(f"Read normalized data from {output_file_path}")
    else:
        n_data = data.copy()

        for i, row in n_data.iterrows():
            if i % 1000 == 0:
                print('\r ready: %.3f%%' % (i / len(n_data) * 100), end=" ")

            # main normalizing step, N2000
            # water_level - mean sea level data for that particular year
            n_data.iat[i, 5] = row['Water level (mm)'] -
                               mean_sea_level_data.loc[mean_sea_level_data['Year'] ==
                               row['Year']].values[0][1]

        n_data = n_data.rename(columns={'Water level (mm)': 'water_level'})

        n_data.to_csv(output_file_path)

        print(f"\nWrote output to {output_file_path}")

    # finding the mean value of each year
    yearly_means = n_data["water_level"].groupby(n_data["Year"]).mean().dropna()

    with open(yearly_means_output_path, "w") as f:
        f.writelines(pd.Series.to_csv(yearly_means))

    return yearly_means


if __name__ == '__main__':
    data_path = r"../data/Kemi/Kemi.csv"
    mean_sea_level_path = r"../data/Kemi/Kemi_mw_n2000.csv"
    normalized_output_path = r"../data/Kemi/Kemi_normalized.csv"
    yearly_means_output_path = r"../data/Kemi/Kemi_yearly_means.csv"
```

```
    yearly_mean = process_data(data_path,
                               mean_sea_level_path,
                               normalized_output_path,
                               yearly_means_output_path)

    print(yearly_mean)
```

R Code Below for the Analysis

Loading libraries

```
library(rstan)
library(ggplot2)
library(dplyr)
library(bayesplot)
library(loo)
library(matrixStats)
```

Setting all the paths

```
main_data_dir = "../data"
separate_stan_model = "../model/separate.stan"
hierarchical_stan_model = "../model/hierarchical.stan"
```

Loading data once it has been pre-processed

```
combineListsAsOne <-function(list1, list2){
  n <- c()
  for(x in list1){
    n<-c(n, x)
  }
  for(y in list2){
    n<-c(n, y)
  }
  return(n)
}
number_of_cities = 0
x = list()
y = list()
ii = list()
city_names = list()
for (data_file in list.files(main_data_dir,
                             pattern="*_yearly_means.csv",
                             full.names=TRUE,
                             recursive = TRUE)){

  preprocessed_data = read.csv(data_file, header=FALSE)
```

```
  c_name = strsplit(strsplit(data_file, "/")[[1]][4],"_")[[1]][1]
  city_names[number_of_cities + 1] = c_name

  x = combineListsAsOne(x, preprocessed_data[,1])
  y = combineListsAsOne(y, preprocessed_data[,2])
  number_of_cities = number_of_cities + 1

  for (j in seq(preprocessed_data[,1])){
    ii = combineListsAsOne(ii, number_of_cities)
  }
}
```

Plotting the data

```
data_plot = list(
    'N'= 49,
    'x'= x[1:49],
    'y'= y[1:49],
    'k' = number_of_cities
)
for (i in seq(1:number_of_cities)){
  x_temp = x
  y_temp = y

  a = ((i-1)*49) +1
  b = i*49
}
```

Preparing data for Stan model

```
data = list(
    'N'= length(x),
    'x'= x,
    'y'= y,
    'k' = number_of_cities,
    'i' = ii
)
```

Separate Stan Model

```
fit_separate = stan(separate_stan_model, data=data)
fit_separate_data = extract(fit_separate, permuted=T)
```

Plots for all cities with slope line and quantiles after Stan fit for Separate Model

```
mu = matrix(data = NA, nrow = 49, ncol = 4000)
```

```
for(i in seq(1:number_of_cities)){

  a = ((i-1)*49) +1
  b = i*49

  x_temp = x[a:b]
  y_temp = y[a:b]

  for(t in seq(from = 1970, to =2019, by =1)){
    mu[t-1970,] =  fit_separate_data$alpha[,i] + ((t-1970) *

                                                  fit_separate_data$beta[,i])
  }

  mu_quan = colQuantiles(t(mu), probs = c(0.05, 0.5, 0.95))

  df_post = data.frame(quan_05=mu_quan[,1],

                       quan_50=mu_quan[,2],

                       quan_95=mu_quan[,3],

                       x=x_temp)
}
```

LOO Fitting for Separate Model

```
loo_separate_data = loo(fit_separate)
```

Hierarchical Stan Model

```
fit_hierarchical = stan(hierarchical_stan_model, data=data)
fit_hierarchical_data = extract(fit_hierarchical, permuted=T)
```

Plots for all cities with slope line and quantiles after Stan fit for Hierarchical Model

```
mu = matrix(data = NA, nrow = 49, ncol = 4000)

for(i in seq(1:number_of_cities)){

  a = ((i-1)*49) +1
  b = i*49

  x_temp = x[a:b]
  y_temp = y[a:b]

  for(t in seq(from = 1970, to =2019, by =1)){
```

```
    mu[t-1970,] =  fit_hierarchical_data$alpha[,i] + ((t-1970) * fit_hierarchical_data$beta[,i])
  }

  mu_quan = colQuantiles(t(mu), probs = c(0.05, 0.5, 0.95))

  df_post = data.frame(quan_05=mu_quan[,1], quan_50=mu_quan[,2], quan_95=mu_quan[,3], x=x_temp)
}
```

Loo

```
loo_hierarchical_data = loo(fit_hierarchical)
```

Posterior Predictive Checking for Separate Model

```
y_rep <- as.matrix(fit_separate, pars = "y_rep")

for (i in seq(1:number_of_cities)){
  x_temp = x
  y_temp = y

  a = ((i-1)*49) +1
  b = i*49
}
```

Posterior Predictive Checking for Hierarchical Model

```
y_rep <- as.matrix(fit_hierarchical, pars = "y_rep")

for (i in seq(1:number_of_cities)){
  x_temp = x
  y_temp = y

  a = ((i-1)*49) +1
  b = i*49
}
```

Model Checking

```
compare(loo_hierarchical_data, loo_separate_data)
```