Bayesian Data Analysis - Final Project

Dmitry Donetskov

07 January 2018

Contents

1 Reseach Question (Problem Statement)

Water evaporation is a major concern in planning irrigation. Data are collected daily from June 6 through June 21 in a central Texas location on the following factors that may affect the amount of evaporation:

- DAY: the calendaric number of day
- For the air temperature:
  - MAXAT: Maximum daily air temperature;
  - MINAT: Minimum daily air temperature;
  - AVAT: The integrated area under the daily air temperature curve, a measure of average air temperature;
- For the soil temprature:
  - MAXST: Maximum daily soil temperature;
  - MINST: Minimum daily soil temperature;
  - AVST: The integrated area under the daily soil temperature curve, a measure of average soil temperature;
- For the daily humidity:
  - MAXH: Maximum daily humidity;
  - MINH: Minimum daily humidity;
  - AVH: The integrated area under the daily humidity curve, a measure of average humidity;
- For the wind:
  - WIND: Total wind, measured in miles per day.
- For the evaporation:
  - EVAP: Daily total evaporation from the soil.

2 Description of Files

File Name	Description
data/irrigation.csv	Data in the fixed width CSV format
script/evaporation.Rmd	RMarkdown document to generate the current report
pairs_plot_1.pdf	Pairs plot for the initial set of data (the A3 size)

3 Initial Configuration

```r
library(ggplot2)
library(psych)
library(rstan)
library(rstanarm)
library(reshape2)

# source("lm_util.r")

rstan_options(auto_write = T)
options(mc.cores = parallel::detectCores())
```

4 Data Load

```
evap_data <- read.table("../data/evaporation.csv", h = T)
```

```
nrow(evap_data)
```

```
## [1] 46
```

```
head(evap_data, 5)
```

```
##   DAY MAXST MINST AVST MAXAT MINAT AVAT MAXH MINH AVH WIND EVAP
## 1   6    84    65  147    85    59  151   95   40 398  273   30
## 2   7    84    65  149    86    61  159   94   28 345  140   34
## 3   8    79    66  142    83    64  152   94   41 388  318   33
## 4   9    81    67  147    83    65  158   94   50 406  282   26
## 5  10    84    68  167    88    69  180   93   46 379  311   41
```

5 Data Investigation

All variables are the range ones. The predicted factor is presented with the EVAP variable.

The DAY variable being *the month day number* day does not look like a good candidate to explain the evaporation for that reason it is reset (to 1) when one month ends and another starts. It might be a good predictor if there are cycles in the evaporation. Finding the association of DAY with the probable cycles is out of the scope of the current task.

In terms of data preparation for analysis, the data looks good. We don't have any missing values, the data is in the numeric format (as it is meant to be).

All the variables are on the range scale so we can use approprate probability distributions for the likelihood function e.g. the normal distribution or the Student's t one.

6 Variables to predict

We have one singe variable, EVAP.

7 Variables as predictors

Putting DAY aside for the time being, there are ten variables: MAXAT, MINAT, AVAT, MAXST, MINST, AVST, MAXH, MINH, AVH, WIND which span across four physical factors: * The air temperature, * The soil temprature, * The daily humidity, * The wind.

We may suspect that there is strong correlation between some variables e.g. between the variables for the air temperature with the ones for the soil temperature as these are two physical factors which tend to be strongly correlated in the nature with each other.

8 Exploratory data analysis

Let's check the pair plots to draw initial conclusions of our data.

The pairs plot is provided in the pairs_plot.pdf file.

```
## pdf
##   2
```

Observations from the pairs plot:

1. There are no variables with the normal distribution, the data is skewed.
2. The data ranges of variables are approximately of the same order.
3. DAY is not correlated with any other variables so let's exclude it.
4. EVAP is positevely correlated with MAXST, MAXAT and negatively with AVH which conforms to the laws of physics.

9 Regression Diagnostics (Classical Way)

```r
library(car)
```

```
##
## Attaching package: 'car'

## The following object is masked from 'package:psych':
##
##     logit
```

```r
evap_lm_1 <- lm(EVAP ~ ., data = evap_data)
vif(evap_lm_1)
```

```
##       DAY     MAXST     MINST      AVST     MAXAT     MINAT      AVAT
##  1.282430 39.391468 14.102516 53.510279  8.990055  9.415915 22.781715
##      MAXH      MINH       AVH      WIND
##  1.985018 25.706800 24.236330  2.155606
```

Most of coefficents are not statistically significant, the VIF factor is quite high (more than 10, which is large). That tells us we need to reduce our dimension to a more principal one.

The correlation matrix (obtained with PROC CORR) indicates there is strong (more than 0.7) correlation between various pairs like * MAXST and MINST * MAXST and AVST * MAXST and AVAT * MINST and AVST * MINSTR and AVAT * AVST and AVAT * MINH and AVH

And it's quite natural as the min/max temperature of air/soil/humidity can indeed explain the integrated area under the daily air/soil/humidity temperature curve. And the temperature ranges of different objects closely interacting with each other surely will influence each other. The physical model represented by these parameters can be looked quite complete but converting it blindly into the statistical one likely gives us results with little practical meanings as the prediction capability of the model will be weak due to large variance of coefficients.

So, we need to resolve this multicollinearity issue by either excluding those variables which are redundant to the model or by introducing new explanatory variables based on the original ones.

The attempt to group variables of the same meaning (min/max/avg) might make sense.

Let's check it.

```r
evap_data_2 <- data.frame(
  day  = evap_data$DAY,
  min  = evap_data$MINST + evap_data$MINAT + evap_data$MINH,
  max  = evap_data$MAXST + evap_data$MAXAT + evap_data$MAXH,
  avg  = evap_data$AVST  + evap_data$AVAT  + evap_data$AVH,
  wind = evap_data$WIND,
  evap = evap_data$EVAP)
```

```r
evap_lm_2 <- lm(evap ~ ., data = evap_data_2)
print(summary(evap_lm_2), digits = 5)
```

```
##
## Call:
## lm(formula = evap ~ ., data = evap_data_2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.5682  -2.7896   1.2746   3.9482  11.7113
##
## Coefficients:
##                Estimate  Std. Error t value  Pr(>|t|)
## (Intercept) -236.892297   44.267992 -5.3513 3.843e-06 ***
## day            0.288044    0.149545  1.9261  0.061214 .
## min            0.534725    0.360768  1.4822  0.146127
## max            1.834360    0.292853  6.2638 2.012e-07 ***
## avg           -0.449008    0.150177 -2.9899  0.004757 **
## wind           0.023923    0.008711  2.7463  0.008989 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.5552 on 40 degrees of freedom
## Multiple R-squared:  0.7632, Adjusted R-squared:  0.7336
## F-statistic: 25.784 on 5 and 40 DF,  p-value: 1.5434e-11
```

```r
vif(evap_lm_2)
```

```
##       day       min       max       avg      wind
##  1.030101 13.618437  7.805330 17.361394  1.329669
```

There are still VIF larger than 10 for MIN and AVG. The correlation matrix also shows that MIN and AVG are correlated.

Let's exclude MIN from the consideration. The evaporation is probably more explained by the MAX temperature (as the factor of more energy) and MIN is included into AVG so by excluding MIN the information of it will still be kept in AVG.

```r
evap_data_3 <- evap_data_2[, !(names(evap_data_2) %in% "min")]
evap_lm_3 <- lm(evap ~ ., data = evap_data_3)
print(summary(evap_lm_3), digits = 5)
```

```
##
## Call:
## lm(formula = evap ~ ., data = evap_data_3)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -21.5821  -3.1622   1.7352   4.1346  11.4006
##
## Coefficients:
##                Estimate  Std. Error t value  Pr(>|t|)
## (Intercept) -1.9173e+02  3.2577e+01 -5.8854 6.297e-07 ***
## day          2.8649e-01  1.5171e-01  1.8885  0.066054 .
## max          1.4456e+00  1.3213e-01 10.9403 9.834e-14 ***
## avg         -2.3712e-01  4.6673e-02 -5.0805 8.655e-06 ***
```

```
## wind         2.8507e-02  8.2616e-03  3.4506  0.001309 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.6647 on 41 degrees of freedom
## Multiple R-squared:  0.7502, Adjusted R-squared:  0.72582
## F-statistic: 30.782 on 4 and 41 DF,  p-value: 7.3301e-12
```

```
vif(evap_lm_3)
```

```
##      day      max      avg     wind
## 1.030051 1.543921 1.629312 1.162070
```

Looking further, one can notice the DAY variable does not probably make sense for the model. Indeed, how the number of day can reduce variance of the evaporation. Excluding it from of the model.

```
evap_data_4 <- evap_data_3[, !(names(evap_data_3) %in% "day")]
evap_lm_4 <- lm(evap ~ ., data = evap_data_4)
print(summary(evap_lm_4), digits = 5)
```

```
##
## Call:
## lm(formula = evap ~ ., data = evap_data_4)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -23.4874  -2.3504   1.3015   4.1395  14.2897
##
## Coefficients:
##               Estimate  Std. Error t value  Pr(>|t|)
## (Intercept) -1.8889e+02  3.3522e+01 -5.6348 1.329e-06 ***
## max          1.4121e+00  1.3488e-01 10.4694 2.802e-13 ***
## avg         -2.2280e-01  4.7439e-02 -4.6966 2.833e-05 ***
## wind         2.7207e-02  8.4806e-03  3.2082  0.002558 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.8954 on 42 degrees of freedom
## Multiple R-squared:  0.72847,    Adjusted R-squared:  0.70907
## F-statistic: 37.559 on 3 and 42 DF,  p-value: 5.8389e-12
```

```
vif(evap_lm_4)
```

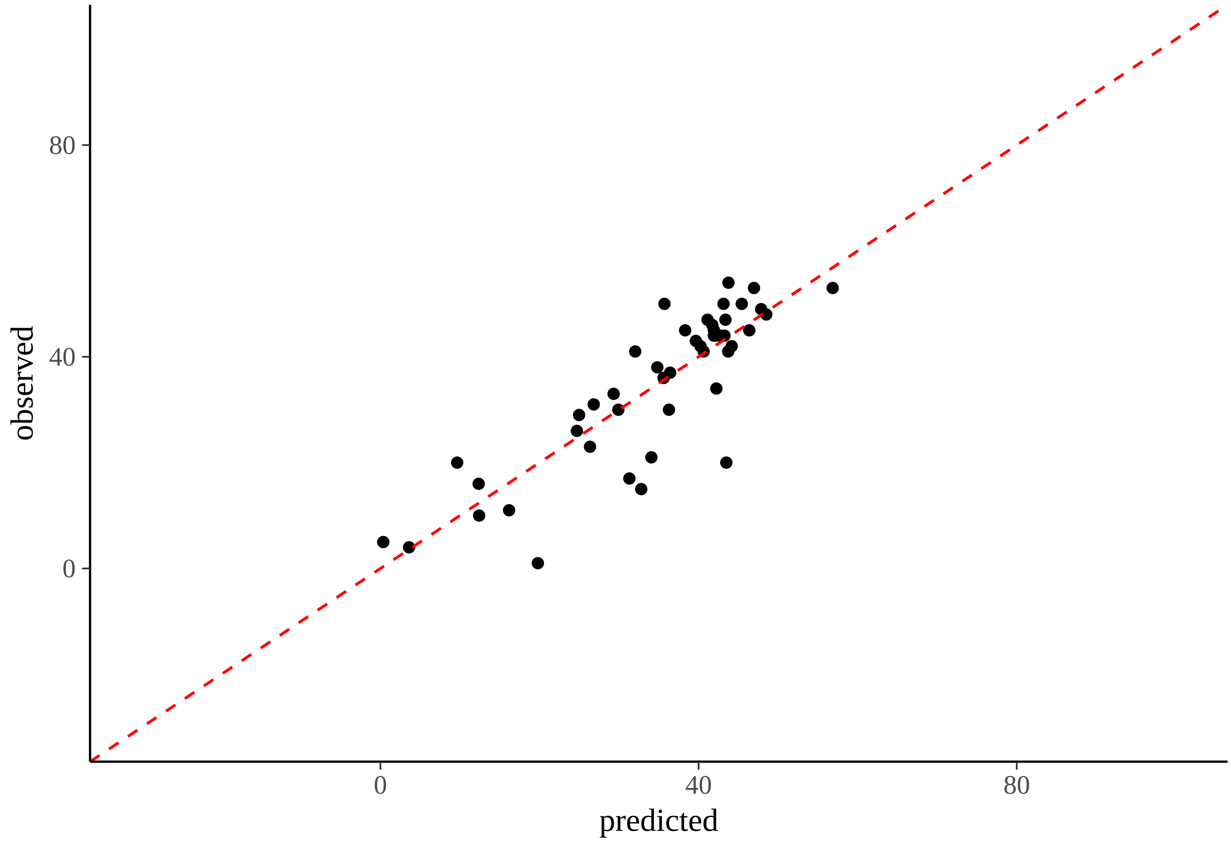
```
##      max      avg     wind
## 1.516173 1.586308 1.154002
```

VIF is not changed, R^2 slightly dropped to 0.7285, the model is statistically significant in overall (F-test) and its coefficients are also statistically significant. It looks like as an working one.

9.1 Prediction Plot

```
# prediction plot
g1 <- ggplot() + xlab("predicted") + ylab("observed") +
        geom_point(aes(x = predict(evap_lm_4), y = evap_data_4$evap)) +
        xlim(-30, 100) +
        ylim(-30, 100) +
```

```
        geom_abline(aes(slope = 1, intercept = 0), lty = "dashed", colour = "red")
plot(g1)
```



10 New Set of Explanatory Variables

Copying the reduced data set into the 'final' data set.

```
evap_data_f <- evap_data_4
```

11 Model Definition

Let's try to answer our research question with the multivariate linear model.

To define it in the Bayesian framework we ideally need to define probability distribution functions for

- the likelihood function,
- the prior probablity distribution,

and choose the link function.

For the current project, we use the default model built-in the 'stan_lm' function:

Building x and y for the model:

```
y <- evap_data_f[, "evap"]
x <- evap_data_f[, !(names(evap_data_f) %in% "evap")]
```

12 Considerations on Priors

Our Bayesian model's main parameters are the coefficients (B). We can presume little about them. Should we probably define their distributions as the uniform ones on some intervals? Information about those intervals we can get from the classical linear regression method.

Anyway, they are going to be rather wide intervals as we don't have any strong opinion about ourthe priors.

13 Bayesian Inference with MCMC

We use stan_lm from rstanarm to get the posteriors.

```
evap_fit_1 <- stan_lm(y ~ .,
                      chains = 1,
                      data   = x,
                      prior  = NULL,
                      adapt_delta = 0.99)
```

```
##
## SAMPLING FOR MODEL 'lm' NOW (CHAIN 1).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 8.56 seconds (Warm-up)
##                11.356 seconds (Sampling)
##                19.916 seconds (Total)
```

```
## Warning: There were 1 divergent transitions after warmup. Increasing adapt_delta above 0.99 may help
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

14 Diagnostics of MCMC

```r
#cat(get_stancode(evap_fit_1$stanfit))

summary(evap_fit_1, digits = 2)
```

```
##
## Model Info:
##
##  function:     stan_lm
##  family:       gaussian [identity]
##  formula:      y ~ .
##  algorithm:    sampling
##  priors:       see help('prior_summary')
##  sample:       1000 (posterior sample size)
##  observations: 46
##  predictors:   4
##
## Estimates:
##                 mean    sd      2.5%    25%     50%     75%     97.5%
## (Intercept)   -179.69   33.83 -241.53 -202.60 -179.48 -158.93 -108.98
## max              1.35    0.14    1.04    1.25    1.36    1.45    1.62
## avg             -0.21    0.05   -0.30   -0.25   -0.21   -0.18   -0.12
## wind             0.03    0.01    0.01    0.02    0.03    0.03    0.04
## sigma            8.19    0.90    6.66    7.51    8.11    8.76   10.15
## log-fit_ratio    0.00    0.07   -0.15   -0.05    0.00    0.05    0.14
## R2               0.68    0.07    0.53    0.64    0.69    0.74    0.79
## mean_PPD        34.72    1.69   31.38   33.54   34.76   35.88   37.82
## log-posterior -164.16    2.03 -168.86 -165.31 -163.85 -162.69 -161.13
##
## Diagnostics:
##               mcse Rhat n_eff
## (Intercept)   1.49 1.00 516
## max           0.01 1.00 435
## avg           0.00 1.00 668
## wind          0.00 1.00 695
## sigma         0.04 1.00 546
## log-fit_ratio 0.00 1.00 453
## R2            0.00 1.00 479
## mean_PPD      0.06 1.00 924
## log-posterior 0.13 1.00 235
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

14.1 Effective Sample Size

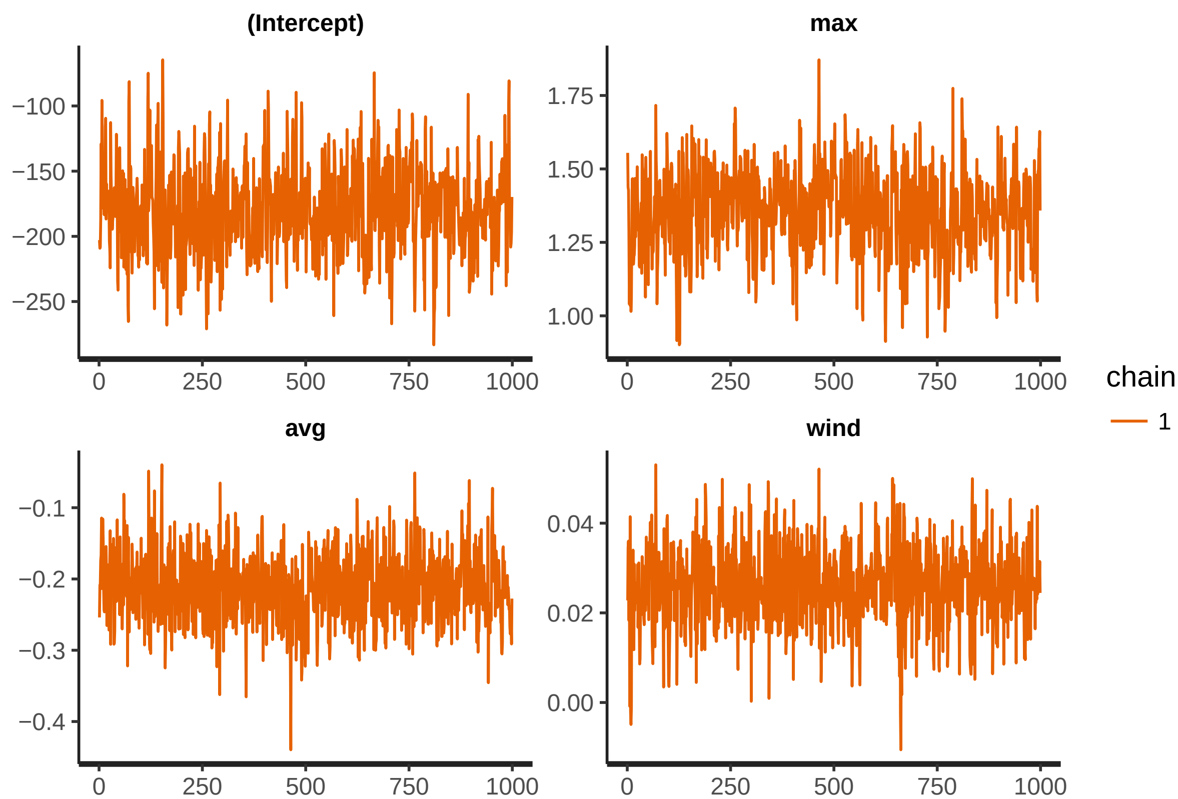
The effective sample size (ESS) for the coefficients of ‘max’ and ‘avg’ are of several hundrends which means the sampler has been successful to find different (effective) samples. Probably, we can increase their number by tuning the sampler’s parameters.

The ESS for ‘wind’ is equal to the total number of samples. That’s a bit unusual, might there be any problem here?

14.2 Trace Plot (Trajectory)

According the trace plot of trajectory below, the MCMC chain explores the space of potential values for the parameters quite well. It's not stuck in a same region for large number of iterations.

```
# g1 <- traceplot(evap_fit_1$stanfit, pars = c('(Intercept)', 'max', 'avg', 'wind'), ncol = 1)
g1 <- stan_trace(evap_fit_1)
plot(g1)
```



15 Reasoning About Coefficients

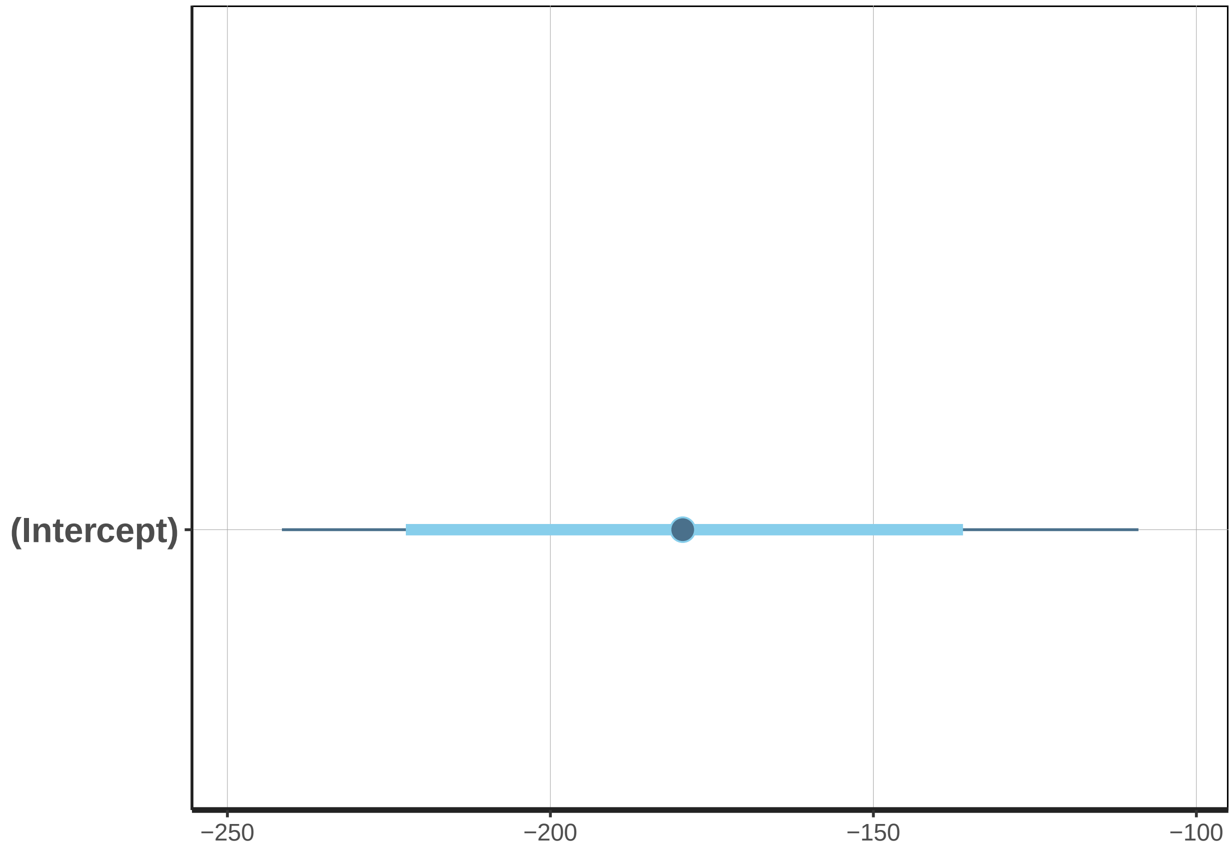
Infering about the coefficients based on the posterior sample (the MCMC draws from the posterior distribution).

```
# g1 <- plot(evap_fit_1) # plot posterior estimates and intervals

rstan_ggtheme_options(panel.background = element_rect(fill = "white"), legend.position = "top")
rstan_gg_options(fill = "skyblue", color = "skyblue4", pt_color = "red")

g1 <- stan_plot(evap_fit_1, pars = c('(Intercept)'))
```

```
## ci_level: 0.8 (80% intervals)
```

```
## outer_level: 0.95 (95% intervals)
```

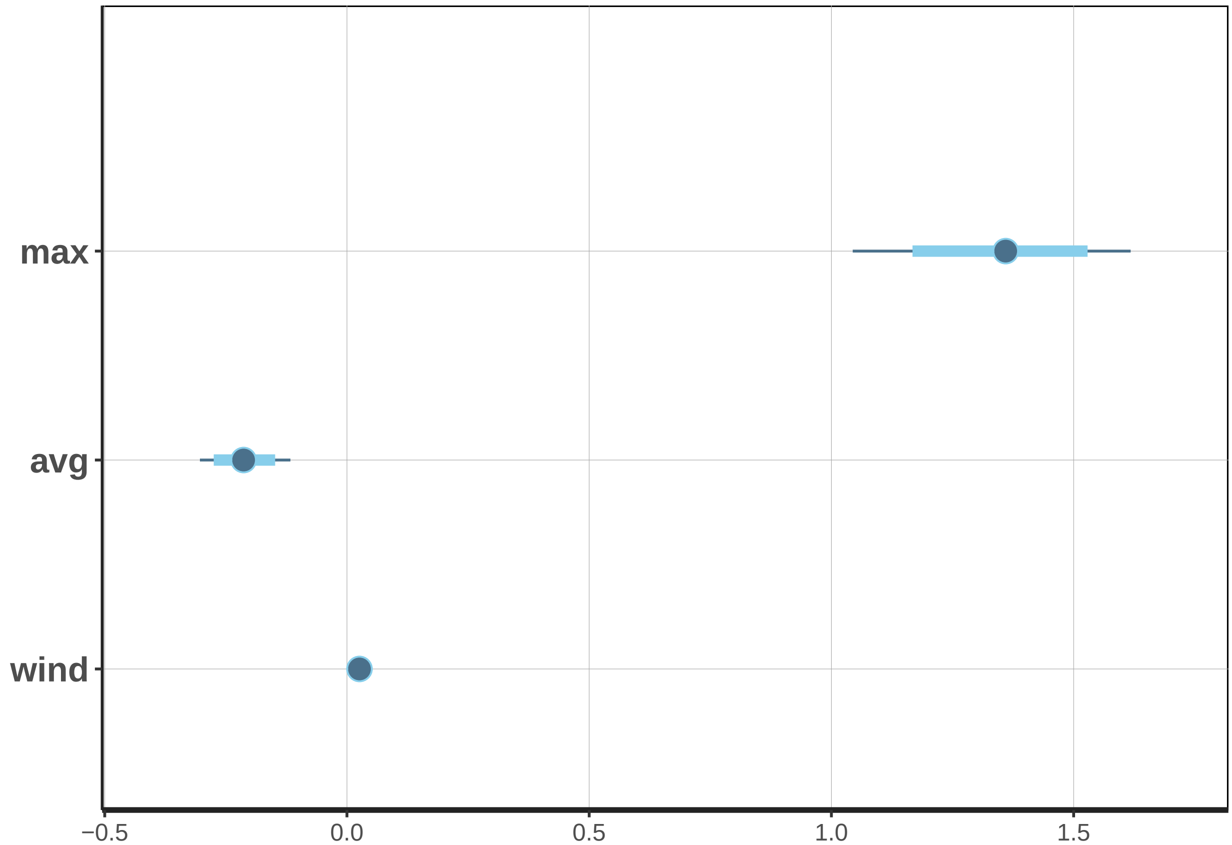
```
plot(g1)
```



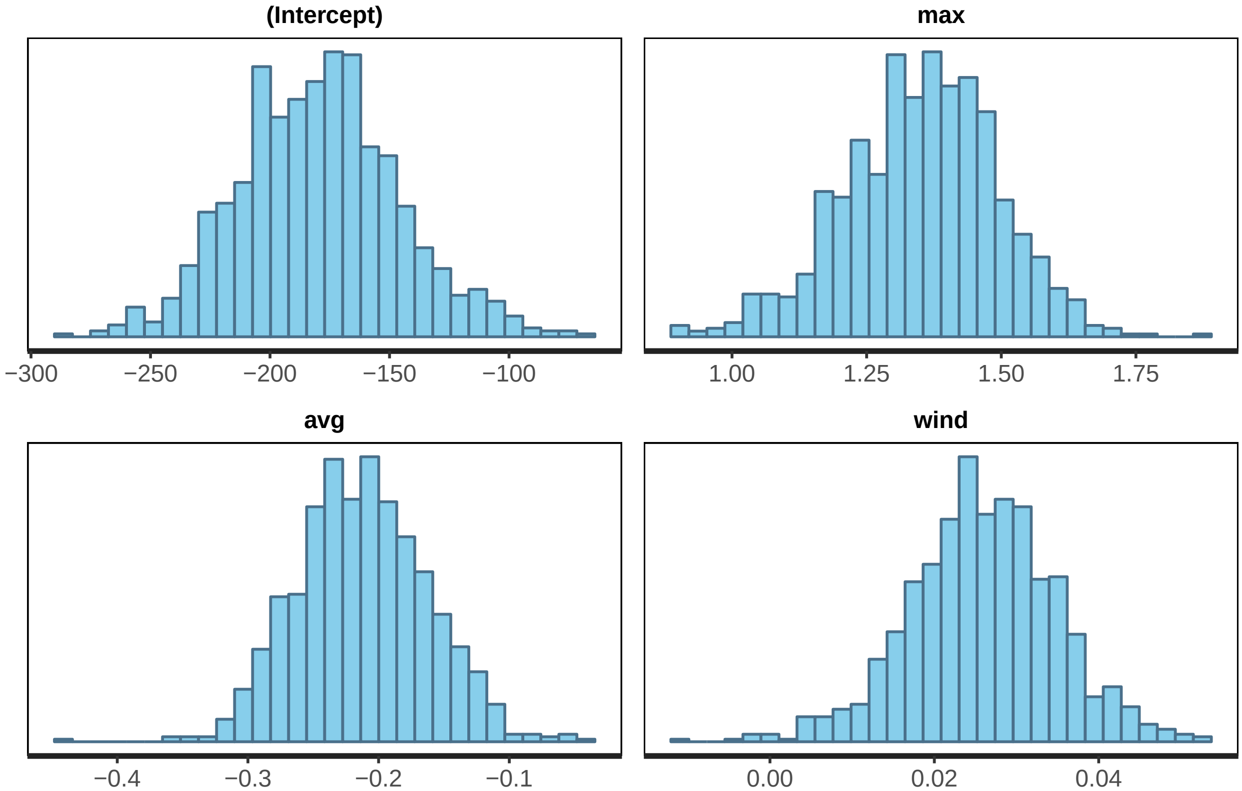

```r
g1 <- stan_plot(evap_fit_1, pars = c('max', 'avg', 'wind'))
```

```
## ci_level: 0.8 (80% intervals)
## outer_level: 0.95 (95% intervals)
```

```r
plot(g1)
```




```
g1 <- quietgg(stan_hist(evap_fit_1))
```

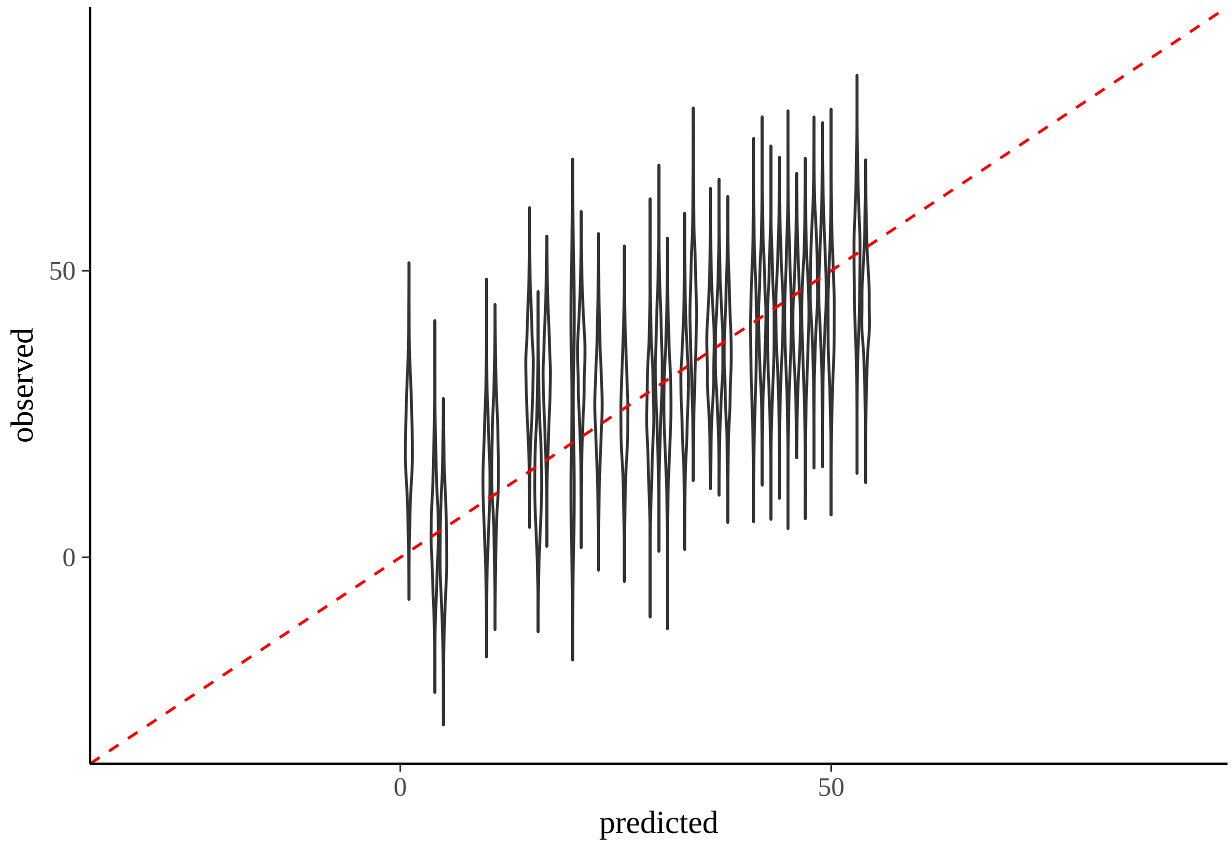



```
# plot(g1)

evap_coef <- extract(evap_fit_1$stanfit)$beta[,1,]
```

16 Prediction

Now, as there is the model built with Bayesian methods meaning the model can provide answers in terms of probabilities as beliefs, let's predict the evaporation ratio for a cold period and a hot period.

```
# samples from posterior predictive
evap_pred <- posterior_predict(evap_fit_1, newdata = evap_data_f)

x <- melt(data.frame(Actual = y, t(evap_pred)), id.vars = "Actual")

g1 <- ggplot(x, aes(x = Actual, y = value, group = Actual)) +
  xlab("predicted") + ylab("observed") +
  geom_violin() + xlim(-30, 90) + ylim(-30, 90) +
  geom_abline(aes(slope = 1, intercept = 0), lty = "dashed", colour = "red")
plot(g1)
```



17 Conclusion

We have found the linear model coefficients with two methods * the classical linear regression, * the MCMC sampling method from the Bayesian framework.

The obtained values are very simular, the MCMC method (in the STAN implementation) has been able to find means of the coefficients even with the default assumption of the priors. The samples generated provide us with additional information about the coefficients i.e. we may approximate their distribuiton and reason about their probable values in the concept of the probability as 'belief'.

18 Ideas for improvements

1. Draw the model in the "plate" notation or the hierarchical diagram.
2. Check if the MCMCM performance will be better if the data are standardized (check the example with bears).
3. Use k-fold cross-validation to get probably better model from the prediction point of view.
4. Try the pp_check function for graphical posterior predictive checks.
5. Try the loo function in the loo package for model comparison.
6. Try the launch_shinystan function in the shinystan package in order to visualize the posterior distribution using the ShinyStan graphical user interface.
7. Display the mean of predicted output on the "Actual vs. Predicted" plot.

19 References

1. The course lectures and examples.
2. Doing Bayesian Data Analysis : a Tutorial with R, JAGS, and Stan / John K. Kruschke.
3. rstanarm documentation: https://www.rdocumentation.org/packages/rstanarm
4. Accessing the contents of a stanfit object: https://cran.r-project.org/web/packages/rstan/vignettes/stanfit-objects.html

20 Appendix A Technical Details of Report

This version of the report was built with:

```
devtools::session_info()
```

```
## Session info -------------------------------------------------------------

##  setting  value
##  version  R version 3.4.3 (2017-11-30)
##  system   x86_64, mingw32
##  ui       RTerm
##  language en
##  collate  Russian_Russia.1251
##  tz       Europe/Moscow
##  date     2018-01-07

## Packages -----------------------------------------------------------------

##  package      * version date       source
##  assertthat     0.2.0   2017-04-11 CRAN (R 3.4.3)
##  backports      1.1.2   2017-12-13 CRAN (R 3.4.3)
##  base         * 3.4.3   2017-12-06 local
##  base64enc      0.1-3   2015-07-28 CRAN (R 3.4.1)
##  bayesplot      1.4.0   2017-09-12 CRAN (R 3.4.3)
##  bindr          0.1     2016-11-13 CRAN (R 3.4.3)
##  bindrcpp       0.2     2017-06-17 CRAN (R 3.4.3)
##  car          * 2.1-6   2017-11-19 CRAN (R 3.4.3)
##  codetools      0.2-15  2016-10-05 CRAN (R 3.4.3)
##  colorspace     1.3-2   2016-12-14 CRAN (R 3.4.3)
##  colourpicker   1.0     2017-09-27 CRAN (R 3.4.3)
##  compiler       3.4.3   2017-12-06 local
##  crosstalk      1.0.0   2016-12-21 CRAN (R 3.4.3)
##  datasets     * 3.4.3   2017-12-06 local
##  devtools       1.13.4  2017-11-09 CRAN (R 3.4.3)
##  digest         0.6.13  2017-12-14 CRAN (R 3.4.3)
##  dplyr          0.7.4   2017-09-28 CRAN (R 3.4.3)
##  DT             0.2     2016-08-09 CRAN (R 3.4.3)
##  dygraphs       1.1.1.4 2017-01-04 CRAN (R 3.4.3)
##  evaluate       0.10.1  2017-06-24 CRAN (R 3.4.3)
##  foreign        0.8-69  2017-06-22 CRAN (R 3.4.3)
##  ggplot2      * 2.2.1   2016-12-30 CRAN (R 3.4.3)
##  glue           1.2.0   2017-10-29 CRAN (R 3.4.3)
##  graphics     * 3.4.3   2017-12-06 local
##  grDevices    * 3.4.3   2017-12-06 local
##  grid           3.4.3   2017-12-06 local
```

```
##  gridExtra      2.3     2017-09-09 CRAN (R 3.4.3)
##  gtable         0.2.0   2016-02-26 CRAN (R 3.4.3)
##  gtools         3.5.0   2015-05-29 CRAN (R 3.4.1)
##  htmltools      0.3.6   2017-04-28 CRAN (R 3.4.3)
##  htmlwidgets    0.9     2017-07-10 CRAN (R 3.4.3)
##  httpuv         1.3.5   2017-07-04 CRAN (R 3.4.3)
##  igraph         1.1.2   2017-07-21 CRAN (R 3.4.3)
##  inline         0.3.14  2015-04-13 CRAN (R 3.4.3)
##  knitr          1.18    2017-12-27 CRAN (R 3.4.3)
##  labeling       0.3     2014-08-23 CRAN (R 3.4.1)
##  lattice        0.20-35 2017-03-25 CRAN (R 3.4.3)
##  lazyeval       0.2.1   2017-10-29 CRAN (R 3.4.3)
##  lme4           1.1-15  2017-12-21 CRAN (R 3.4.3)
##  loo            1.1.0   2017-03-27 CRAN (R 3.4.3)
##  magrittr       1.5     2014-11-22 CRAN (R 3.4.3)
##  markdown       0.8     2017-04-20 CRAN (R 3.4.3)
##  MASS           7.3-47  2017-02-26 CRAN (R 3.4.3)
##  Matrix         1.2-12  2017-11-20 CRAN (R 3.4.3)
##  MatrixModels   0.4-1   2015-08-22 CRAN (R 3.4.3)
##  matrixStats    0.52.2  2017-04-14 CRAN (R 3.4.3)
##  memoise        1.1.0   2017-04-21 CRAN (R 3.4.3)
##  methods      * 3.4.3   2017-12-06 local
##  mgcv           1.8-22  2017-09-24 CRAN (R 3.4.3)
##  mime           0.5     2016-07-07 CRAN (R 3.4.1)
##  miniUI         0.1.1   2016-01-15 CRAN (R 3.4.3)
##  minqa          1.2.4   2014-10-09 CRAN (R 3.4.3)
##  mnormt         1.5-5   2016-10-15 CRAN (R 3.4.1)
##  munsell        0.4.3   2016-02-13 CRAN (R 3.4.3)
##  nlme           3.1-131 2017-02-06 CRAN (R 3.4.3)
##  nloptr         1.0.4   2017-08-22 CRAN (R 3.4.3)
##  nnet           7.3-12  2016-02-02 CRAN (R 3.4.3)
##  parallel       3.4.3   2017-12-06 local
##  pbkrtest       0.4-7   2017-03-15 CRAN (R 3.4.3)
##  pillar         1.0.1   2017-11-27 CRAN (R 3.4.3)
##  pkgconfig      2.0.1   2017-03-21 CRAN (R 3.4.3)
##  plyr           1.8.4   2016-06-08 CRAN (R 3.4.3)
##  psych        * 1.7.8   2017-09-09 CRAN (R 3.4.3)
##  quantreg       5.34    2017-10-25 CRAN (R 3.4.3)
##  R6             2.2.2   2017-06-17 CRAN (R 3.4.3)
##  Rcpp         * 0.12.14 2017-11-23 CRAN (R 3.4.3)
##  reshape2     * 1.4.3   2017-12-11 CRAN (R 3.4.3)
##  rlang          0.1.6   2017-12-21 CRAN (R 3.4.3)
##  rmarkdown      1.8     2017-11-17 CRAN (R 3.4.3)
##  rprojroot      1.3-1   2017-12-18 CRAN (R 3.4.3)
##  rsconnect      0.8.5   2017-08-23 CRAN (R 3.4.3)
##  rstan        * 2.17.2  2017-12-21 CRAN (R 3.4.3)
##  rstanarm     * 2.17.2  2017-12-21 CRAN (R 3.4.3)
##  rstantools     1.4.0   2017-12-21 CRAN (R 3.4.3)
##  scales         0.5.0   2017-08-24 CRAN (R 3.4.3)
##  shiny          1.0.5   2017-08-23 CRAN (R 3.4.3)
##  shinyjs        0.9.1   2017-06-29 CRAN (R 3.4.3)
##  shinystan      2.4.0   2017-08-02 CRAN (R 3.4.3)
##  shinythemes    1.1.1   2016-10-12 CRAN (R 3.4.3)
##  SparseM        1.77    2017-04-23 CRAN (R 3.4.1)
```

```
##  splines        3.4.3   2017-12-06 local
##  StanHeaders  * 2.17.1  2017-12-20 CRAN (R 3.4.3)
##  stats        * 3.4.3   2017-12-06 local
##  stats4         3.4.3   2017-12-06 local
##  stringi        1.1.6   2017-11-17 CRAN (R 3.4.2)
##  stringr        1.2.0   2017-02-18 CRAN (R 3.4.3)
##  survival       2.41-3  2017-04-04 CRAN (R 3.4.3)
##  threejs        0.3.1   2017-08-13 CRAN (R 3.4.3)
##  tibble         1.4.1   2017-12-25 CRAN (R 3.4.3)
##  tools          3.4.3   2017-12-06 local
##  utils        * 3.4.3   2017-12-06 local
##  withr          2.1.1   2017-12-19 CRAN (R 3.4.3)
##  xtable         1.8-2   2016-02-05 CRAN (R 3.4.3)
##  xts            0.10-1  2017-12-20 CRAN (R 3.4.3)
##  yaml           2.1.16  2017-12-12 CRAN (R 3.4.3)
##  zoo            1.8-0   2017-04-12 CRAN (R 3.4.3)
```