Bayesian regression for predicting combined cycle power plant output

Khang Nguyen, khang.nguyen@aalto.fi Son Le, son.le@aalto.fi

Introduction

- Thermodynamic systems are hard to construct and solve
- Combined cycle power plants (CCPPs) are examples
- Predict electrical output of CCPPs to maximize profit
- Past attempts: mathematical nonlinear equations, machine learning models
- Answer this problem with Bayesian approach

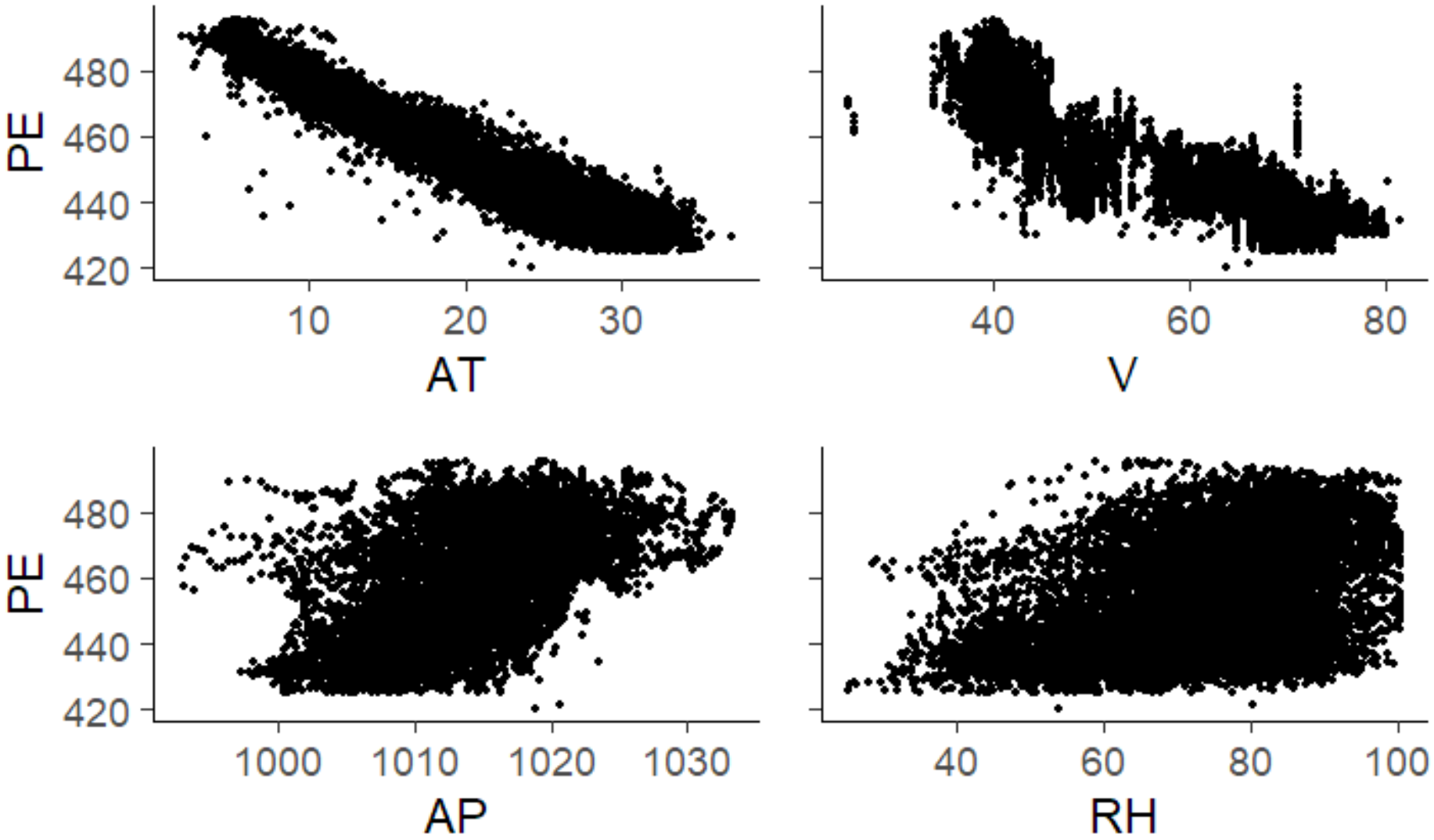
Data description

- Taken from the University of California Irvine Machine Learning Repository
- 9568 x 5 dataset, with AT (ambient temperature), V (exhaust vacuum), AP (atmospheric pressure), RH (relative humidity), and PE (full-load electrical output)

AT	V	AP	RH	PE
288.1	417.6	1024.1	731.7	463.3
298.3	629.6	1020.0	590.8	444.4
278.3	394.0	1012.2	921.4	488.6
294.0	573.2	1010.2	766.4	446.5
284.0	375.0	1009.2	966.2	473.9

Data description

Data description



Data analysis problem

- Fit regression models to predict the dependent variable by using independent variables
- Full-load electrical power output PE is predicted by the ambient temperature AT, the atmospheric pressure AP, the relative humidity RH, and the exhaust vacuum V
- Created two models to solve this problem: linear model and generalized additive model (GAM)

Model description

- Implemented as the baseline for this analysis
- Has the form

$$y_i = \mathbf{x_i}^T \mathbf{w} + \epsilon_i, \quad i = 1, \ldots, n$$
$$\epsilon_i \sim N(0, \sigma)$$

- Chosen priors

$$\mathbf{w} \sim N(0, 5)$$
$$\text{Intercept} \sim N(450, 50)$$
$$\sigma \sim \text{exponential}(0.05)$$

Convergence diagnostics

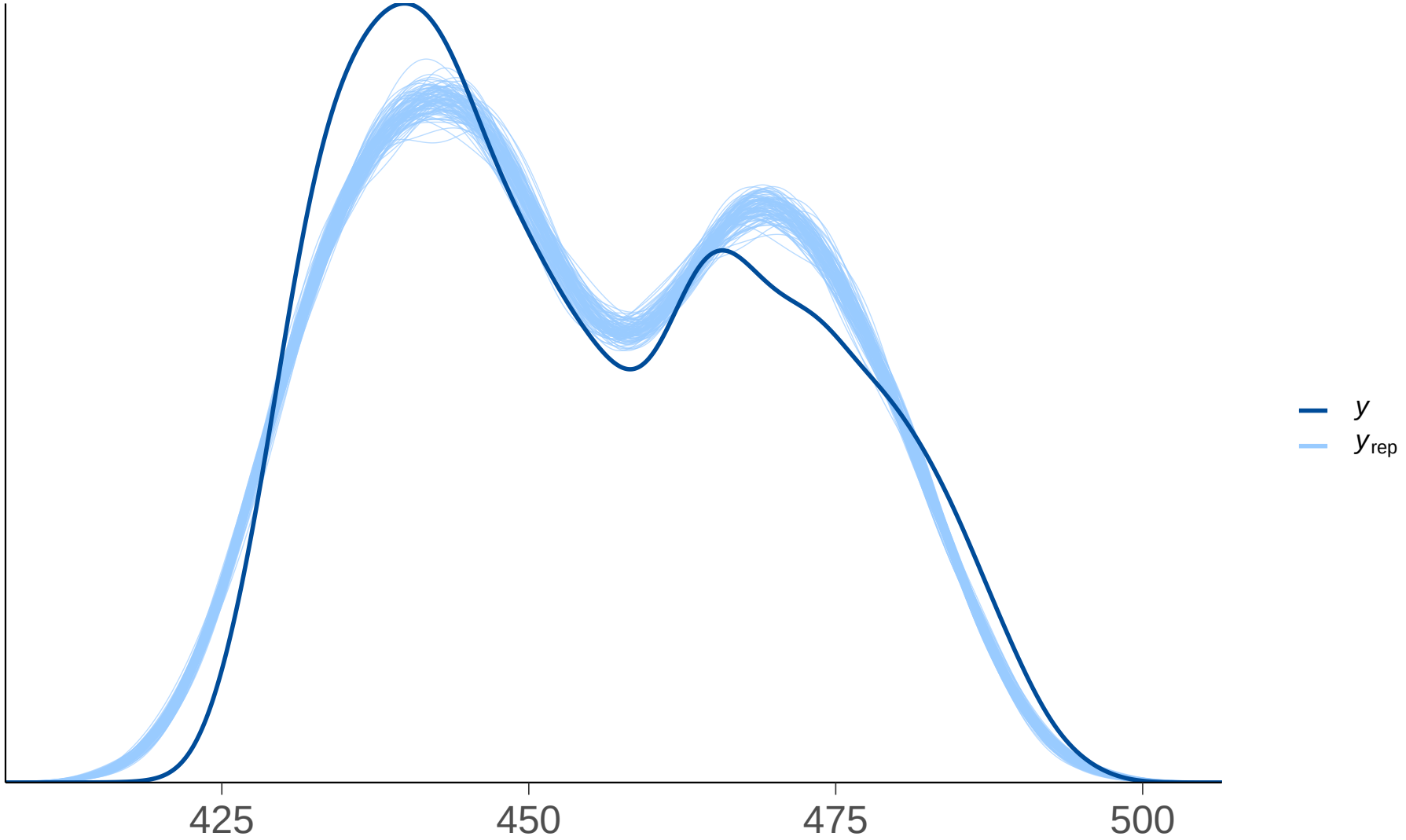
- Model built with `brms` and inference run with Stan
- No iteration exceededs maximum tree depth of 10
- Other HMC-specific convergence diagnostics were good
- $\hat{R}$-values were approximately 1
- n_{eff}/N ratios were good

Posterior predictive checks

- How well do the models describe the data?
- Simulate the data used for posterior predictive checks by sampling from the posterior predictive distribution:

$$p(\widetilde{y} \mid y) = \int p(\widetilde{y} \mid \theta)\, p(\theta \mid y)\, d\theta$$

Posterior predictive checks - kernel density estimates



Posterior predictive checks - error scatter plot

Sensitivity analysis

- 2 alternative set of priors were tested
- Uses only weakly informative priors, including the Intercept:

$$\text{Intercept} \sim \textit{student}T(3, 0, 1)$$

- Different set of priors with domain knowledge for Intercept:

$$\text{Intercept} \sim N(450, 50)$$

- Conclusion: priors built from domain knowledge perform better.

Model description

- Captures nonlinear relationships through linear combination of smooth functions of explanatory variables
  - A smooth function has a linear basis expansion in some chosen basis
  - Splines are piecewise polynomials
  - Knots connect the pieces
  - Thin plate spline basis for smooth functions - avoids knot selection
- Formula:

$$\mu = \text{Intercept} + s(x_1) + s(x_2) + s(x_3) + s(x_4)$$
$$y \sim N(\mu, \sigma)$$

- Each smooth term $s(x_i)$ has 2 parts: fixed effect and random effect

Prior

- Standard deviation of each smooth term:

$$\sigma_i \sim \text{exponential}(0.1), \quad i = 1, \ldots, 4$$

- Fixed effects:

$$\beta_i \sim N(0, 100), \quad i = 1, \ldots, 4$$

- Standard deviation of errors:

$$\sigma \sim \text{exponential}(0.05)$$

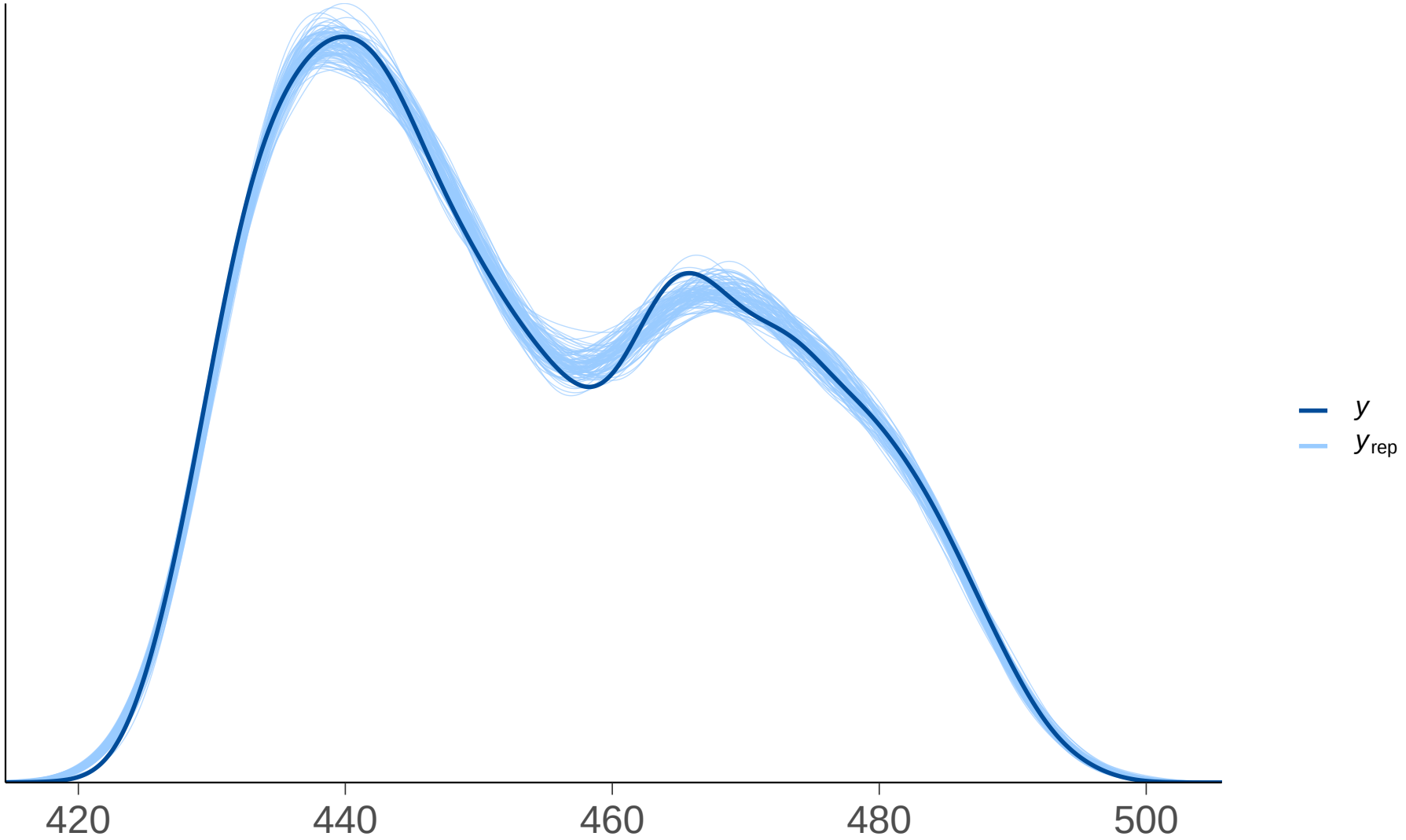
- Intercept:

$$\text{Intercept} \sim N(460, 75)$$

Convergence diagnostics

- Model built with `brms` and inference run with Stan
- 3 iterations exceeded maximum tree depth of 10
- Other HMC-specific convergence diagnostics were good
- $\hat{R}$-values were approximately 1
- n_{eff}/N ratios were good

Posterior predictive checks - kernel density estimates



Posterior predictive checks - error scatter plot

Sensitivity analysis

- Two extra set of priors were tested
- Wider prior for standard deviations σ_i of smooth terms:

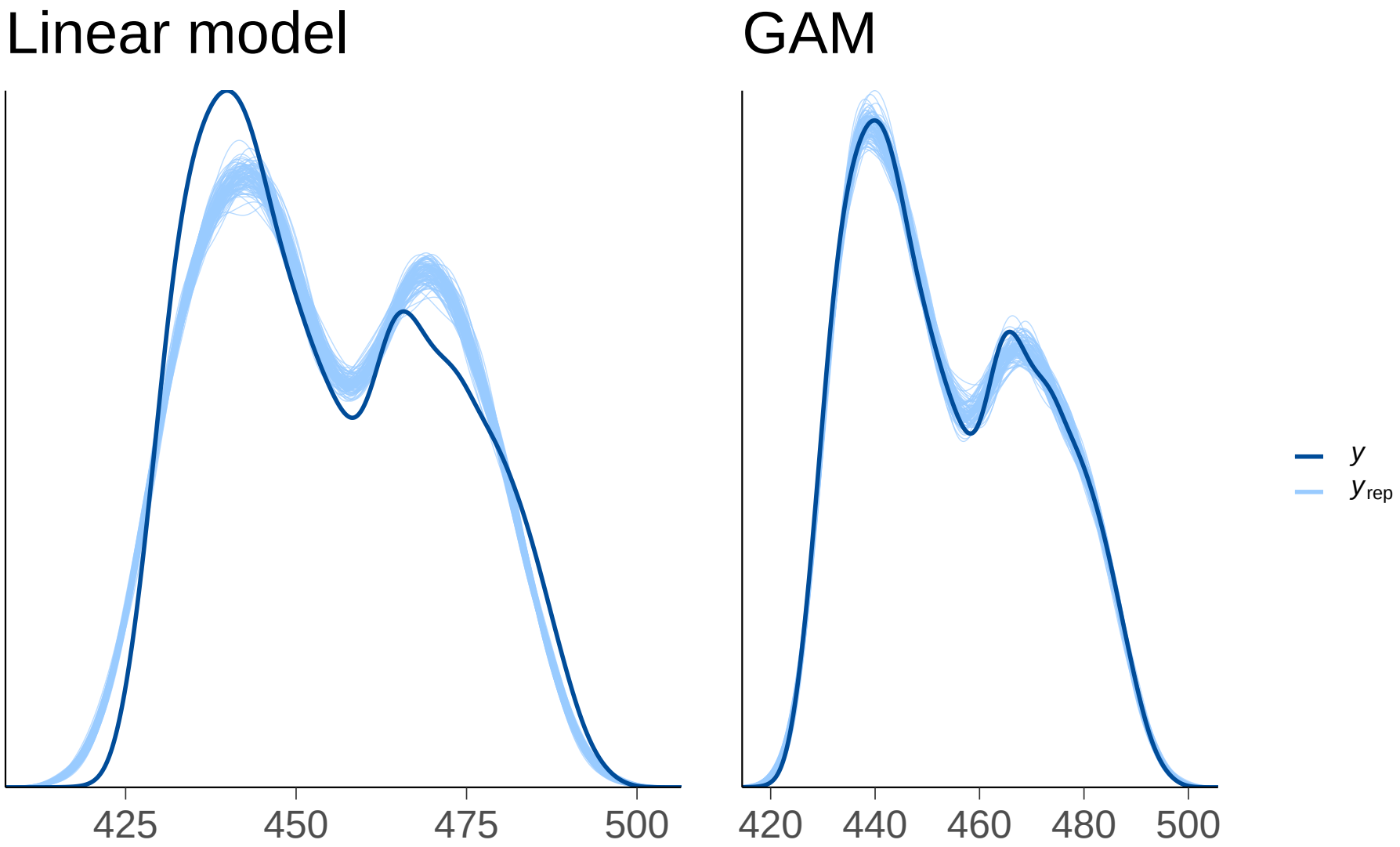
$$\sigma_i \sim \text{exponential}(0.001), \quad i = 1, \ldots, 4$$

- Narrower priors for σ_i and fixed effects:

$$\begin{aligned} \sigma_i &\sim \text{exponential}(1), & i &= 1, \ldots, 4 \\ \beta_i &\sim N(0, 1), & i &= 1, \ldots, 4 \end{aligned}$$

- Conclusion: priors for those parameters should not be too wide or too narrow.

Kernel density estimates



Leave-one-out cross-validation comparison

- The difference in ELPD estimates of linear model and GAM:

	elpd_diff	se_diff	elpd_loo	se_elpd_loo	p_loo
gam	0.0	0.0	-21873.9	105.0	34.4
linear	-733.9	42.2	-22607.7	88.5	7.1

$\Rightarrow$ GAM is better.

Predictive performance - RMSE

- Linear model:

	Estimate	Est.Error	Q2.5	Q97.5
RMSEs	4.738	0.004	4.731	4.747

- GAM:

	Estimate	Est.Error	Q2.5	Q97.5
RMSEs	4.405	0.008	4.391	4.421

Predictive performance - Bayesian R2

- Linear model:

	Estimate	Est.Error	Q2.5	Q97.5
R2	0.91737	0.00051	0.91636	0.91833

- GAM:

	Estimate	Est.Error	Q2.5	Q97.5
R2	0.92911	0.00044	0.92823	0.92995

Conclusion

- Predict the full-load electrical output of a combined cycle power plant with conditions of the plant
- 2 Bayesian models with our prior belief
- GAM perform significantly better than linear model (ELPD, RMSE, Bayesian R2, and posterior predictive density checking)
- Possible improvements: variable selection and hierarchical model with groups created by an external clustering method

Variable conversion

```
dat_modified <-
  dat %>% mutate(
    V = V*10, # from cmHg to mmHg
    RH = RH*10, # from percent to per-mille (‰)
    AT = AT %>% conv_unit("C", "K") # from Celsius to Kelvin
  )
```

Linear model - Stan code

```
// generated with brms 2.14.4
functions {
}
data {
  int<lower=1> N;  // total number of observations
  vector[N] Y;  // response variable
  int<lower=1> K;  // number of population-level effects
  matrix[N, K] X;  // population-level design matrix
  int prior_only;  // should the likelihood be ignored?
}
transformed data {
  int Kc = K - 1;
  matrix[N, Kc] Xc;  // centered version of X without an intercept
  vector[Kc] means_X;  // column means of X before centering
  for (i in 2:K) {
    means_X[i - 1] = mean(X[, i]);
    Xc[, i - 1] = X[, i] - means_X[i - 1];
  }
}
```

Linear model - Stan code

```
parameters {
  vector[Kc] b;  // population-level effects
  real Intercept;  // temporary intercept for centered predictors
  real<lower=0> sigma;  // residual SD
}
transformed parameters {
}
model {
  // likelihood including all constants
  if (!prior_only) {
    target += normal_id_glm_lpdf(Y | Xc, Intercept, b, sigma);
  }
  // priors including all constants
  target += normal_lpdf(b | 0, 5);
  target += normal_lpdf(Intercept | 450, 50);
  target += exponential_lpdf(sigma | 0.05)
}
```

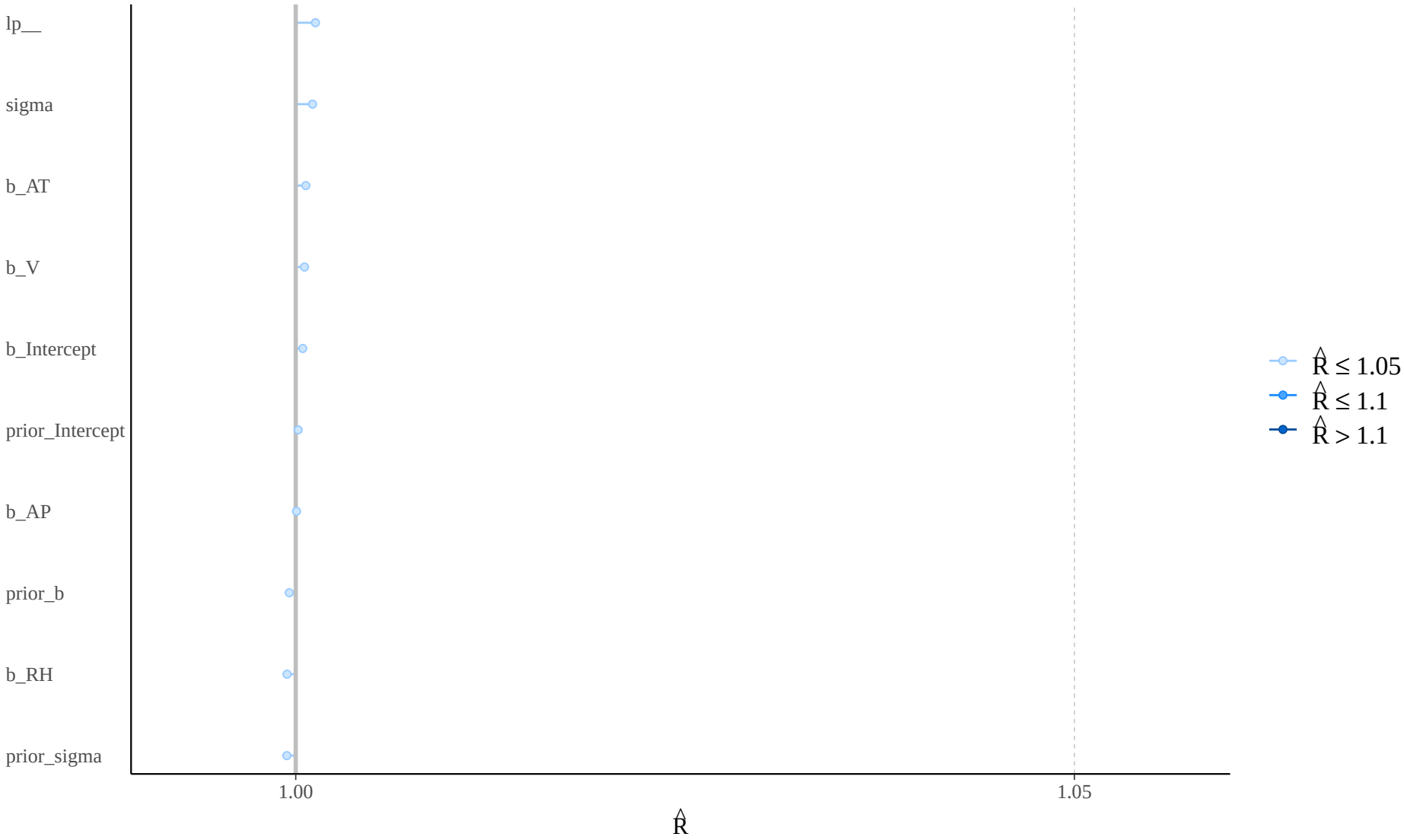
Linear model - Stan code

```
generated quantities {
  // actual population-level intercept
  real b_Intercept = Intercept - dot_product(means_X, b);
  // additionally draw samples from priors
  real prior_b = normal_rng(0,5);
  real prior_Intercept = normal_rng(450,50);
  real prior_sigma = exponential_rng(0.05);
  // use rejection sampling for truncated priors
  while (prior_sigma < 0) {
    prior_sigma = exponential_rng(0.05);
  }
```

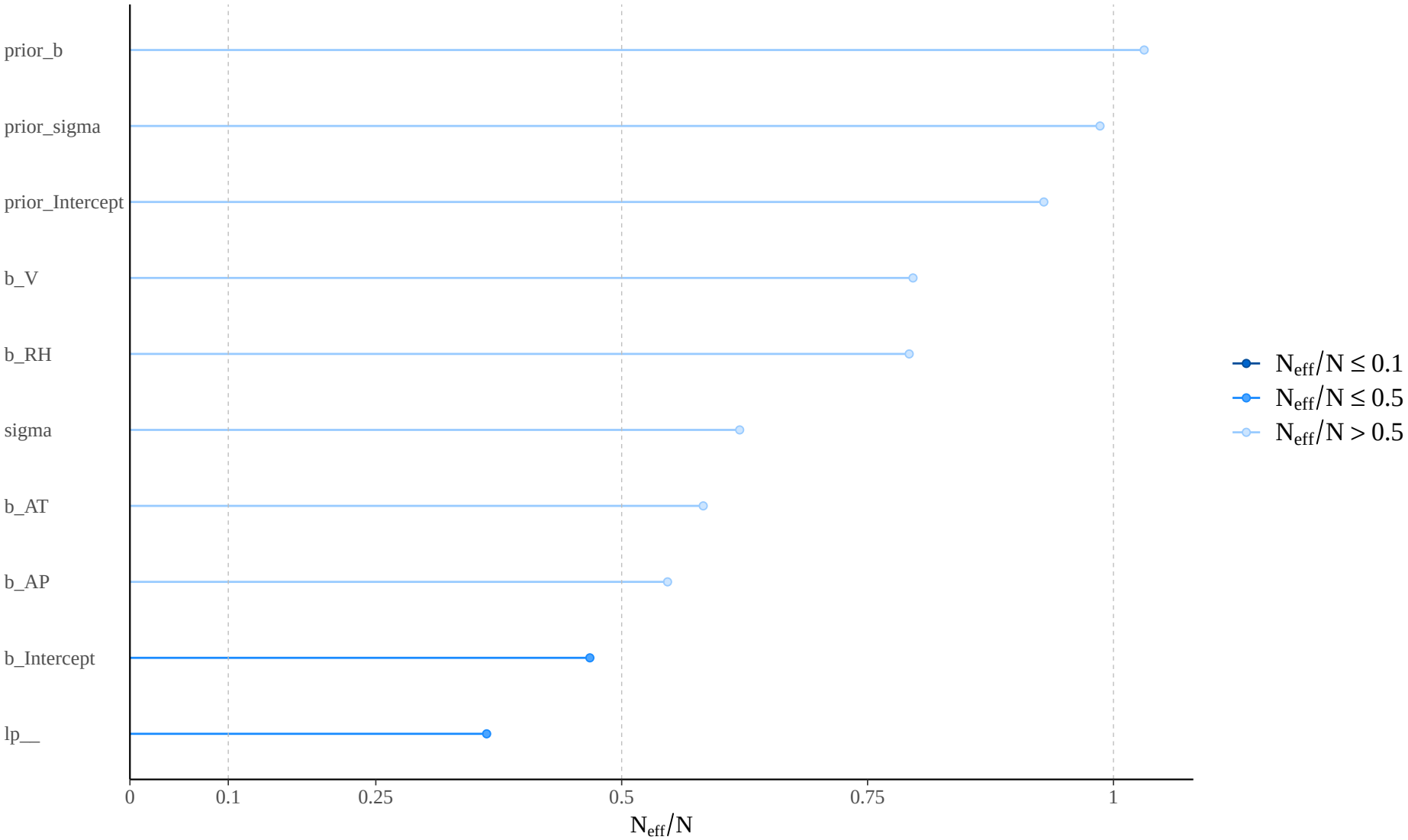
Linear model - Convergence diagnostics

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: PE ~ AT + V + AP + RH
##    Data: train (Number of observations: 7724)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept  1003.21     13.70   976.68  1029.96 1.00     1894     1969
## AT           -1.98      0.02    -2.02    -1.95 1.00     2355     2121
## V            -0.02      0.00    -0.02    -0.02 1.00     3209     2901
## AP            0.06      0.01     0.03     0.08 1.00     2189     1875
## RH           -0.02      0.00    -0.02    -0.01 1.00     3192     3220
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     4.52      0.04     4.45     4.59 1.00     2496     2292
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

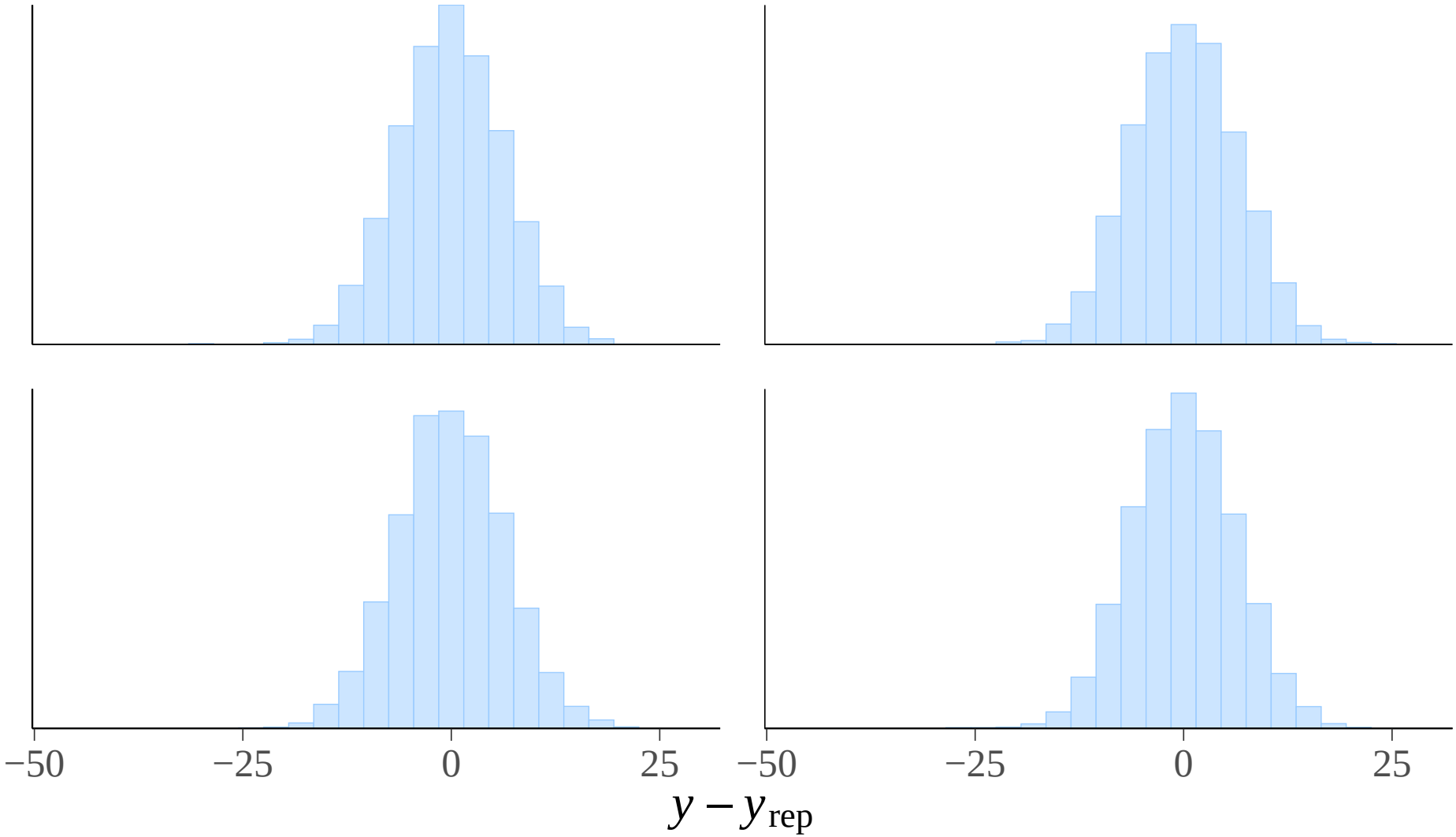
Linear model - Convergence diagnostics



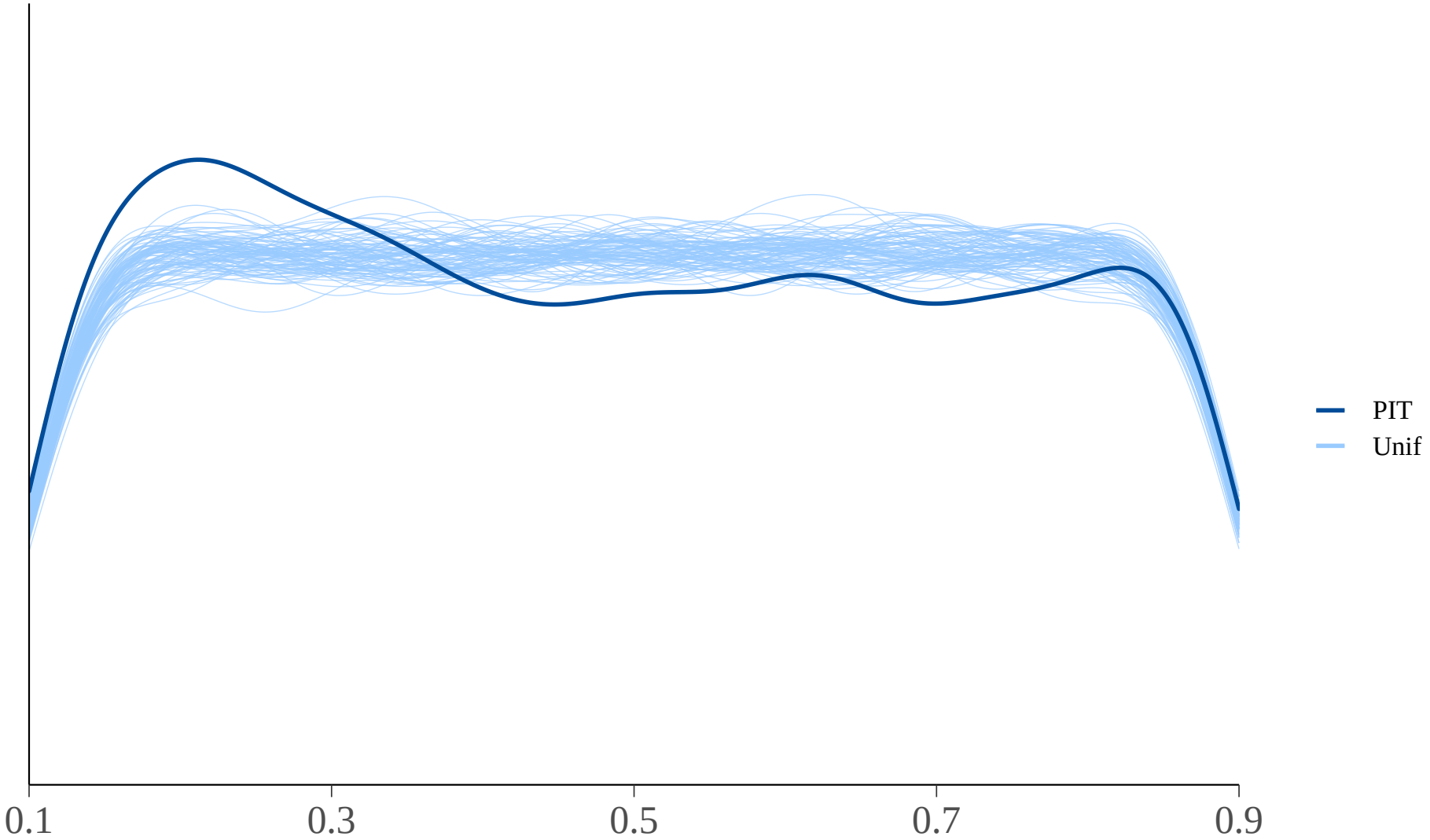
Linear model - Convergence diagnostics



Linear model - Posterior predictive checks, error histograms



Linear model - Posterior predictive checks, LOO-PIT



Linear model - LOO-CV

```
##
## Computed from 4000 by 7724 log-likelihood matrix
##
##          Estimate    SE
## elpd_loo -22607.7  88.5
## p_loo         7.1   0.7
## looic     45215.5 176.9
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

GAM - Stan code

```
// generated with brms 2.14.4
functions {
}
data {
  int<lower=1> N;  // total number of observations
  vector[N] Y;  // response variable
  // data for splines
  int Ks;  // number of linear effects
  matrix[N, Ks] Xs;  // design matrix for the linear effects
  // data for spline s(AT)
  int nb_1;  // number of bases
  int knots_1[nb_1];  // number of knots
  // basis function matrices
  matrix[N, knots_1[1]] Zs_1_1;
  // data for spline s(V)
  int nb_2;  // number of bases
  int knots_2[nb_2];  // number of knots
  // basis function matrices
  matrix[N, knots_2[1]] Zs_2_1;
```

GAM - Stan code

```
  // data for spline s(AP)
  int nb_3;  // number of bases
  int knots_3[nb_3];  // number of knots
  // basis function matrices
  matrix[N, knots_3[1]] Zs_3_1;
  // data for spline s(RH)
  int nb_4;  // number of bases
  int knots_4[nb_4];  // number of knots
  // basis function matrices
  matrix[N, knots_4[1]] Zs_4_1;
  int prior_only;  // should the likelihood be ignored?
}
transformed data {
}
```

GAM - Stan code

```
parameters {
  real Intercept;  // temporary intercept for centered predictors
  vector[Ks] bs;  // spline coefficients
  // parameters for spline s(AT)
  // standarized spline coefficients
  vector[knots_1[1]] zs_1_1;
  real<lower=0> sds_1_1;  // standard deviations of spline coefficients
  // parameters for spline s(V)
  // standarized spline coefficients
  vector[knots_2[1]] zs_2_1;
  real<lower=0> sds_2_1;  // standard deviations of spline coefficients
  // parameters for spline s(AP)
  // standarized spline coefficients
  vector[knots_3[1]] zs_3_1;
  real<lower=0> sds_3_1;  // standard deviations of spline coefficients
  // parameters for spline s(RH)
  // standarized spline coefficients
  vector[knots_4[1]] zs_4_1;
  real<lower=0> sds_4_1;  // standard deviations of spline coefficients
  real<lower=0> sigma;  // residual SD
}
```

GAM - Stan code

```
transformed parameters {
  // actual spline coefficients
  vector[knots_1[1]] s_1_1;
  // actual spline coefficients
  vector[knots_2[1]] s_2_1;
  // actual spline coefficients
  vector[knots_3[1]] s_3_1;
  // actual spline coefficients
  vector[knots_4[1]] s_4_1;
  // compute actual spline coefficients
  s_1_1 = sds_1_1 * zs_1_1;
  // compute actual spline coefficients
  s_2_1 = sds_2_1 * zs_2_1;
  // compute actual spline coefficients
  s_3_1 = sds_3_1 * zs_3_1;
  // compute actual spline coefficients
  s_4_1 = sds_4_1 * zs_4_1;
}
generated quantities {
  // actual population-level intercept
  real b_Intercept = Intercept;
}
```

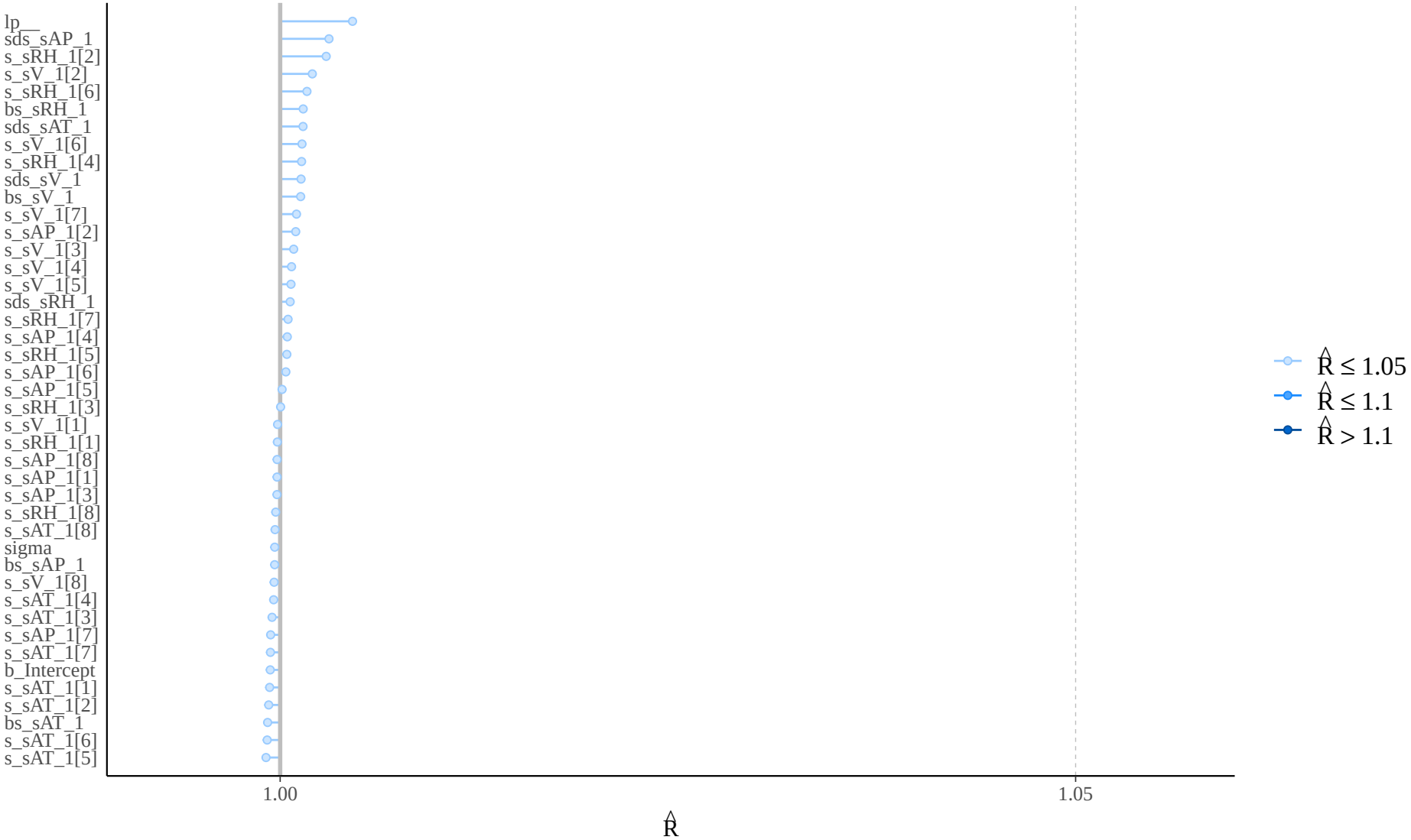
GAM - Stan code

```
model {
  // likelihood including all constants
  if (!prior_only) {
    // initialize linear predictor term
    vector[N] mu = Intercept + rep_vector(0.0, N) +
                   Xs * bs + Zs_1_1 * s_1_1 + Zs_2_1 * s_2_1 +
                   Zs_3_1 * s_3_1 + Zs_4_1 * s_4_1;
    target += normal_lpdf(Y | mu, sigma);
  }
  // priors including all constants
  target += normal_lpdf(Intercept | 460, 75);
  target += normal_lpdf(bs | 0, 100)
    - 4 * normal_lccdf(0 | 0, 100);
  target += exponential_lpdf(sds_1_1 | 0.1);
  target += std_normal_lpdf(zs_1_1);
  target += exponential_lpdf(sds_2_1 | 0.1);
  target += std_normal_lpdf(zs_2_1);
  target += exponential_lpdf(sds_3_1 | 0.1);
  target += std_normal_lpdf(zs_3_1);
  target += exponential_lpdf(sds_4_1 | 0.1);
  target += std_normal_lpdf(zs_4_1);
  target += exponential_lpdf(sigma | 0.05);
}
```

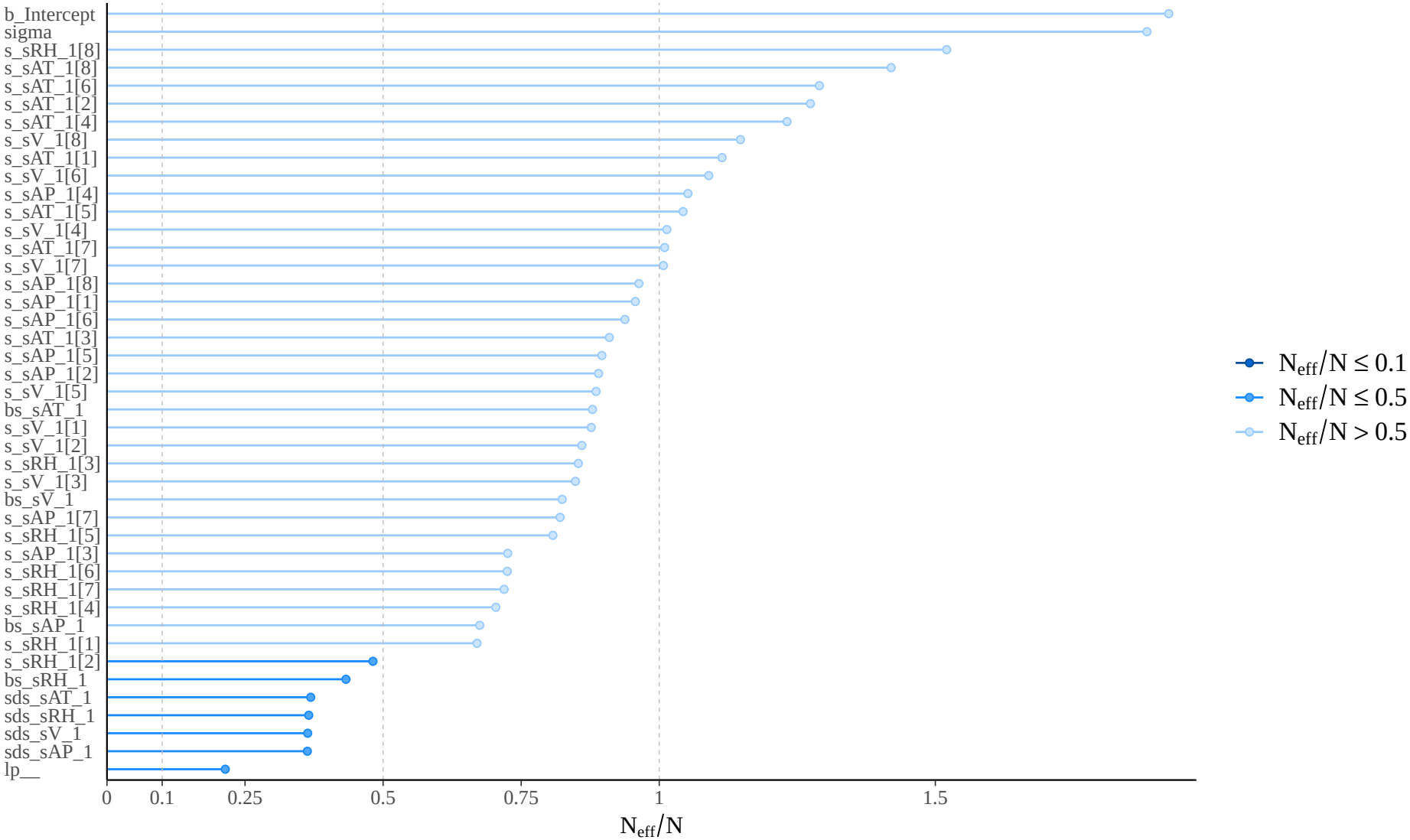
GAM - Convergence diagnostics

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: PE ~ s(AT) + s(V) + s(AP) + s(RH)
##    Data: train (Number of observations: 7724)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Smooth Terms:
##             Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sds(sAT_1)    14.36      4.14     8.38    24.45 1.00     1402     2044
## sds(sV_1)     19.79      5.21    11.93    32.27 1.00     1380     2046
## sds(sAP_1)     9.04      3.10     4.77    16.61 1.00     1386     2259
## sds(sRH_1)     6.19      3.08     1.92    13.51 1.00     1366     2126
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept   454.45      0.05   454.36   454.54 1.00     7461     2437
## sAT_1       -63.24     11.03   -83.87   -40.53 1.00     3462     2775
## sV_1         -4.68      9.96   -23.79    15.20 1.00     3319     2351
## sAP_1       -11.13      8.04   -26.79     4.56 1.00     2715     2393
## sRH_1        -1.22      8.26   -15.10    16.83 1.00     1682     2249
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     4.10      0.03     4.03     4.17 1.00     7378     2712
##
## Samples were drawn using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

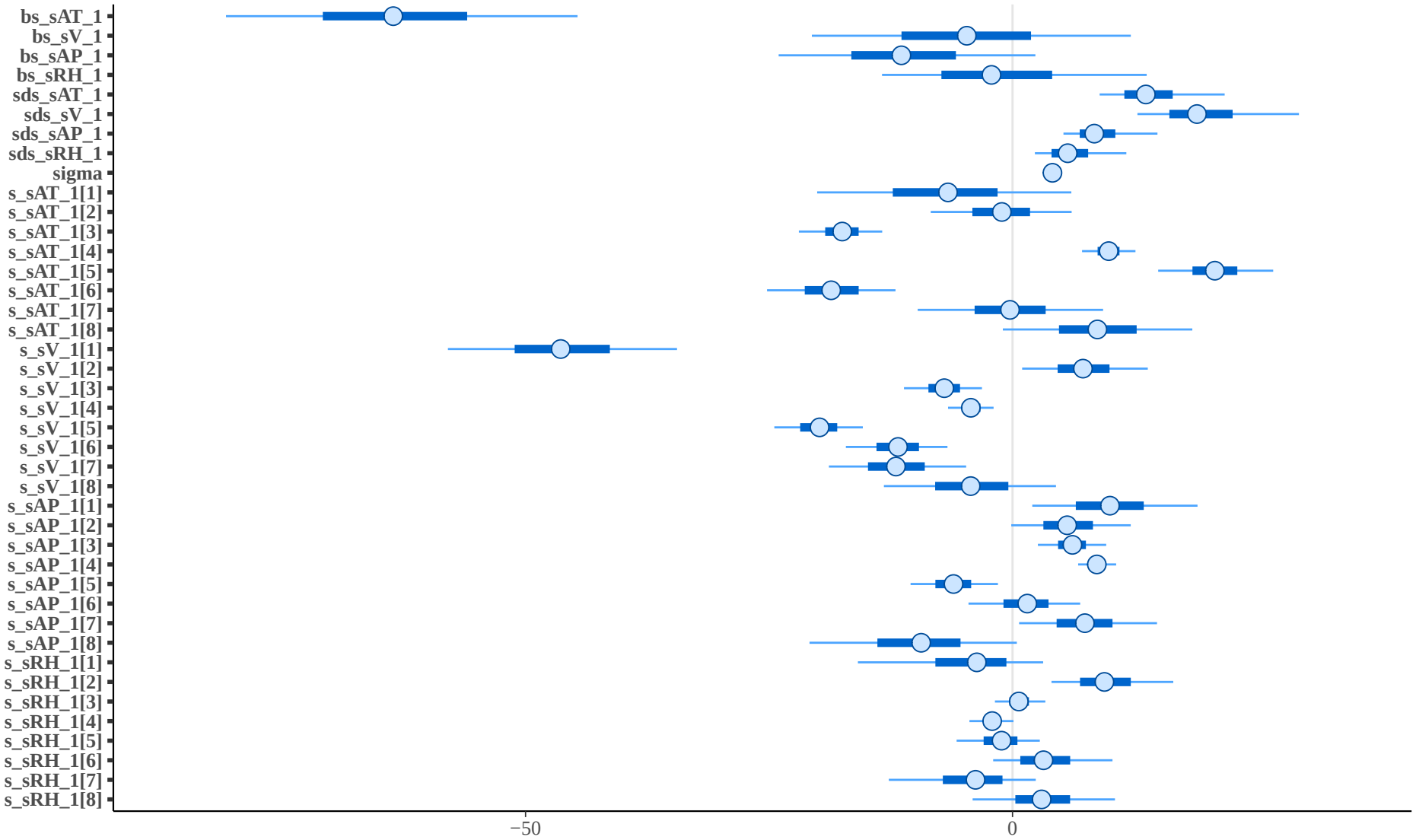
GAM - Convergence diagnostics



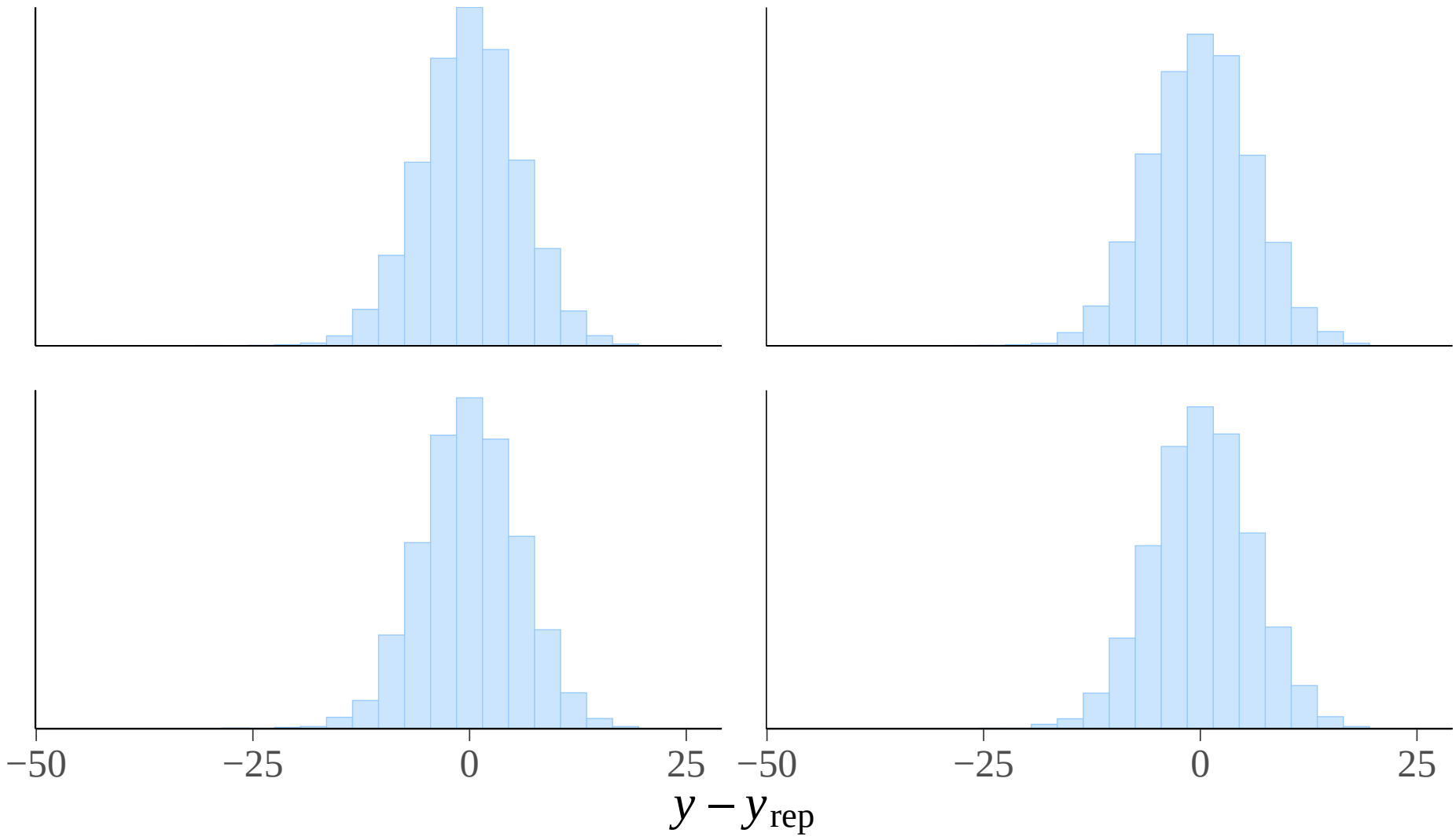
GAM - Convergence diagnostics



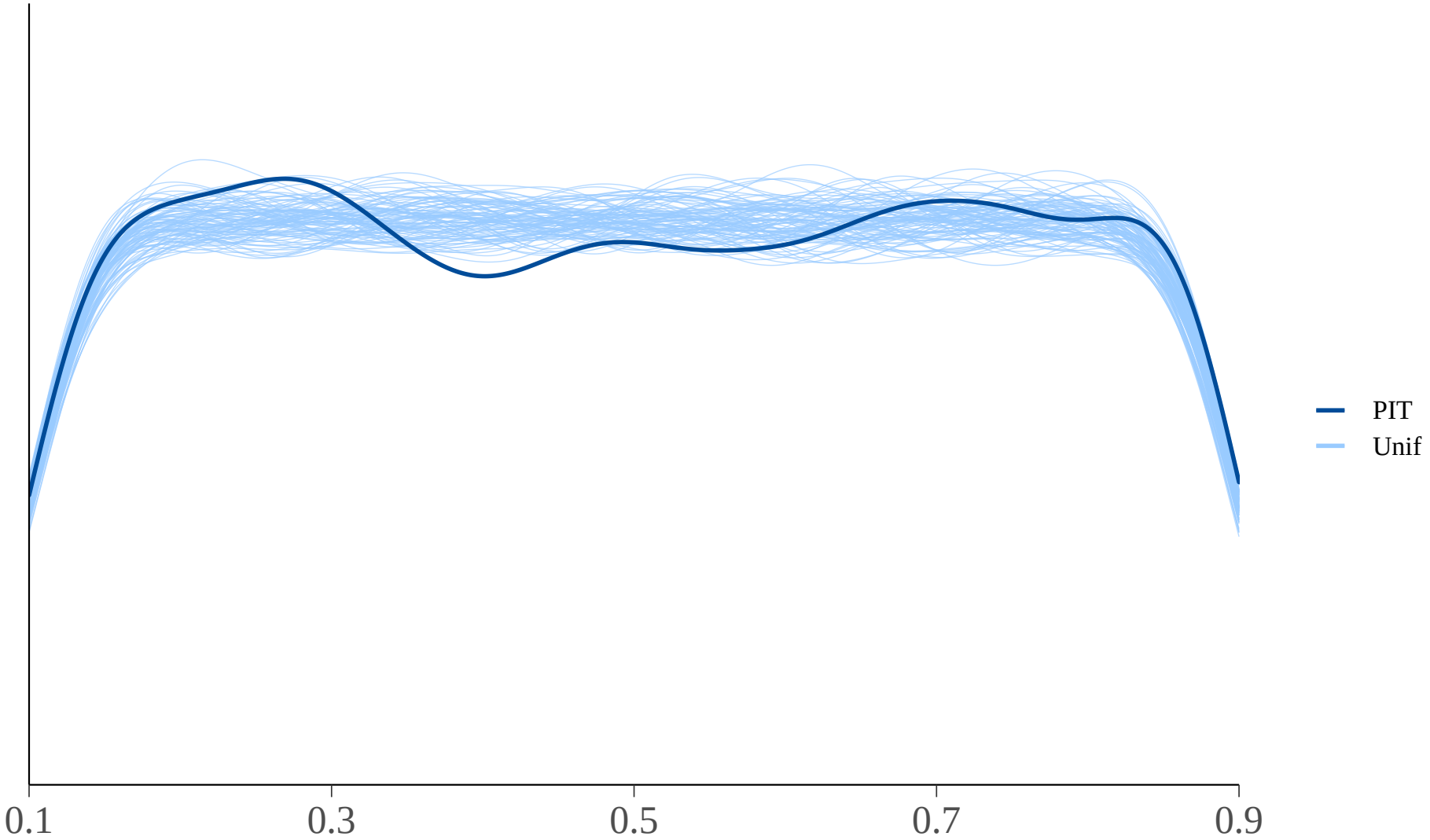
GAM - Convergence diagnostics



GAM - Posterior predictive checks, error histograms



GAM - Posterior predictive checks, LOO-PIT



GAM - LOO-CV

```
##
## Computed from 4000 by 7724 log-likelihood matrix
##
##          Estimate    SE
## elpd_loo -21873.9 105.0
## p_loo        34.4   2.1
## looic     43747.8 210.1
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```