Bayesian Data Analysis Project Report

Contents

Introduction

We use a dataset called Motor Trend Car Road Tests. The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models). Details of the dataset are discussed below. And here's a overview of the whole structure of this report.

- Overview of the dataset
- Data Exploration and Analysis Problem
- Model Formulation
- Model diagnostics and results
- Conclusion & Discussion

Overview of the dataset

The dataset consists of 32 cars, for which the following information has been recorded:

- Miles per gallon (a measure of fuel consumption)
- Number of cylinders
- Displacement (the total volume of the cylinders)
- Gross horsepower
- Rear axle ratio (related to towing capabilities)
- Weight
- Quarter mile time (how fast the car can traverse a quarter mile)
- Shape of the engine - straight vs V-shaped
- Transmission - automatic vs manual
- Number of forward gears
- Number of carburetors

We can see that it's pretty natural to study how car design can affect the 1/4 mile time, in other words, the performance of the car. From the information the dataset provided, it seems confusing because many of them are correlated, like displacement and horsepower. Because Engine displacement and horsepower of the engine (without any power boosters) are linearly proportional that a engine having swept volume of 14cc to 17cc will produce the power of 1hp. So how to choose the variables of our model? Let's first do the unit conversion and make some plots to explore the data further.

Data Exploration and Analysis Problem

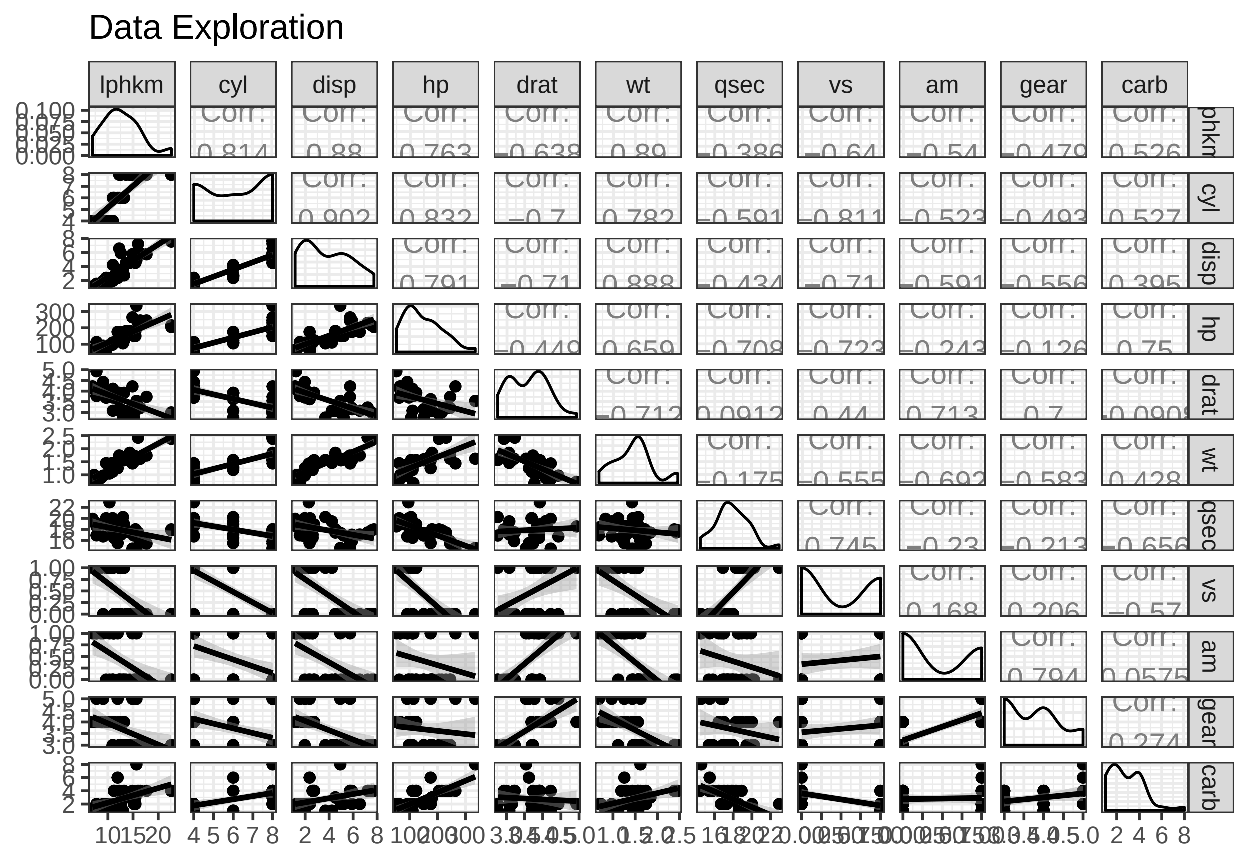
Data Preprocessing and Data Exploration

We would like to first convert data in imperial units to data in metric units for easier interpretation. - fuel consumption: From *Miles per US gallon* to *Litres per 100 km* - Weight: From *pound* to *ton* - Displacement: From *inch* to *litre*

```
car_properties <- data.frame(lphkm = 235.215/mtcars$mpg,
                             cyl = mtcars$cyl,
                             disp = mtcars$disp/61.024,
                             hp = mtcars$hp,
                             drat = mtcars$drat,
                             wt = mtcars$wt/2.205,
                             qsec = mtcars$qsec,
                             vs = mtcars$vs,
                             am = mtcars$am,
                             gear = mtcars$gear,
                             carb = mtcars$carb)
```

Then we can plot each pair of the variables in a matrix and see what we can get.

```r
# Make a copy for correct plotting
plotcars <- car_properties
# Change default plots styled for the lower triangular of the plot matrix
lower <- list(continuous = "smooth",
                combo = "facetdensity")
ggpairs(plotcars,
        lower = lower,
        showStrips = TRUE,
        progress = FALSE,
        title = "Data Exploration") +
  theme_bw()
```



First, we can notice that relationships of some pairs are already linear, which can be a desirable property. Then from the correlation measures, we can see that horsepower and displacement do have high correlation as we speculated above, and as for the 1/4 mile time, shape of the engine, horsepower, number of carburetors, number of cylinders and displacement are all highly correlated with it. But since they are all measures of engine, can we use only a few of them as some of the variables in our model?
However, counterintuitively, we also notice that car weight and transmission type (as automation transmission could optimize the performance) do not have relatively hugh impacts on car performance.
Consequently, we now have the variable that we're interested in based on our prior information about which parts can affect a car's performance: horsepower, number of carburetors, car weight, transmission type and shape of the engine.

```r
#Do a best subset selection for linear regression models trying out all possible combination
car.sum <- summary(regsubsets(qsec ~., data = plotcars))
car.sum$which[which.min(car.sum$bic), 2:11]
```

```
## lphkm   cyl  disp    hp  drat    wt    vs    am  gear  carb
## FALSE FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE  TRUE
```

But here's another question, the best subset selected by the *regsubsets* function is quite different from our model, which of these 2 models performs the best speaking of predicting 1/4 mile time? What if we include all variables in the Bayesian linear regression model?

Therefore, we've already had 3 models in mind for comparison. We will formally formulate our analysis problem in the next section.

Analysis Problem Defining

Let's summarize the problem and the chosen variables we have come up with so far.
- Analysis Problem: What are the best variables that can be used to predict car performance? - Variables: For multivariate model 1, horsepower, number of carburetors, car weight, transmission type and shape of the engine are the variables X. For multivariate model 2, displacement, car weight, shape of the engine, number of carburetors would be used as variables X. And for multivariate model 3, all variables will be used. ## Model Formulation Since there's no clear need for non-linearity as stated above, we can use bayesian linear regression, i.e.,

$$y \sim N(\alpha + \beta X, \sigma^2)$$

where y is 1/4 mile time, X is the matrix of predictor variables and α, β are the intercept and regression coefficients, respectively.

We will try the univariate linear regression and the multivariate linear regression to check the dependencies. Normal linear regression captures the dependencies between predictors and responses. We will then choose the best model and continue with the analysis.

Priors

We use weakly informative priors. The weakly informative priors are useful because they provide some information on the relative a priori plausibility of the possible parameter values. They also help to reduce posterior uncertainty and stabilize computations. The prior for α is a Cauchy distribution with center 0 and scale 10. As priors for the regression coefficients β we use a Student's t distribution with 3 degrees of freedom, location 0 and scale 2. For the standard deviation of the posterior distribution, σ we use a half-normal (0, 10) distribution.

The rule of thumb for weakly informative distributions is that the standard deviation of the posterior distribution should be less than 0.1 times that of the prior. The scales of the priors were chosen so that this rule is respected. To do this, we need to calculate the standard deviation of the prior. For a t-distribution with scale s and ν degrees of freedom, the standard deviation is:

$$\sqrt{s^2 \cdot \frac{\nu}{\nu - 2}} = \sqrt{2^2 * \frac{3}{3-2}} = 2 \cdot \sqrt{3} \approx 3.46$$

For a Cauchy distribution, the standard deviation is not defined, and for the normal prior of σ we use standard deviation of 10.

Then we would like to normalize all data into the range [0,1] except for engine type and transmission type as they are already in the range [0,1], which can benifit us a lot in the sense that if we are dealing with unit variables in the later section of model formulation, it's much easier for both interpretation and prior distribution selection of the regression coefficients.

```
# scale numeric variables
scaled_car_properties<-car_properties
for (i in seq(from=1,to=length(car_properties)))
  scaled_car_properties[i]<-scale(scaled_car_properties[i])
```

separate linear Modeling

```
stan_separate_model = '
data {
  int<lower=0> N;
  vector[N] x;
  vector[N] y;
}
parameters {
  real alpha;
  real beta;
  real<lower=0> sigma;
}
transformed parameters{
  vector[N] mu;
  mu = alpha + beta*x;
}
model {
  alpha ~ cauchy(0,10);
  beta ~ student_t(3,0,2);
  sigma ~ normal(0, 10);
  y ~ normal(mu, sigma);
}
// Log likelihoods genereated for LOO
generated quantities {
  vector[N] log_lik;
  for (i in 1:N)
    log_lik[i] = normal_lpdf(y[i] |alpha+x[i]*beta , sigma);
}
'

fit_lphkm = stan(data = list(N=length(scaled_car_properties$qsec),
                   x=c(scaled_car_properties$lphkm),
                   y=c(scaled_car_properties$qsec)),
                    model_code = stan_separate_model,refresh=0)
monitor(fit_lphkm, probs = c(0.1, 0.5, 0.9))
```

```
## Inference for the input samples (4 chains: each with iter = 2000; warmup = 0):
##
##                Q5   Q50   Q95  Mean  SD  Rhat Bulk_ESS Tail_ESS
## alpha        -0.3   0.0   0.3   0.0 0.2     1     2828     2137
## beta         -0.7  -0.4  -0.1  -0.4 0.2     1     3257     2555
## sigma         0.8   1.0   1.2   1.0 0.1     1     2639     2403
## mu[1]        -0.2   0.2   0.5   0.2 0.2     1     2945     2366
## mu[2]        -0.2   0.2   0.5   0.2 0.2     1     2945     2366
## mu[3]        -0.1   0.2   0.6   0.2 0.2     1     2962     2676
## mu[4]        -0.1   0.2   0.5   0.2 0.2     1     2948     2338
## mu[5]        -0.3   0.0   0.3   0.0 0.2     1     2836     2056
## mu[6]        -0.3   0.0   0.3   0.0 0.2     1     2815     2171
## mu[7]        -0.8  -0.4   0.0  -0.4 0.2     1     3042     2243
## mu[8]        -0.1   0.3   0.7   0.3 0.2     1     2975     2611
## mu[9]        -0.1   0.2   0.6   0.2 0.2     1     2962     2676
## mu[10]       -0.2   0.1   0.3   0.1 0.2     1     2853     2135
## mu[11]       -0.3   0.0   0.2   0.0 0.2     1     2803     2167
```

```
## mu[12]       -0.5  -0.2   0.2  -0.2 0.2     1     2856     1997
## mu[13]       -0.4  -0.1   0.2  -0.1 0.2     1     2811     2088
## mu[14]       -0.6  -0.3   0.1  -0.3 0.2     1     2936     2268
## mu[15]       -1.8  -1.0  -0.2  -1.0 0.5     1     3227     2605
## mu[16]       -1.8  -1.0  -0.2  -1.0 0.5     1     3227     2605
## mu[17]       -0.7  -0.3   0.1  -0.3 0.2     1     2994     2248
## mu[18]        0.0   0.5   1.0   0.5 0.3     1     3096     2645
## mu[19]        0.0   0.5   1.0   0.5 0.3     1     3085     2645
## mu[20]        0.0   0.6   1.1   0.6 0.3     1     3107     2646
## mu[21]       -0.1   0.2   0.5   0.2 0.2     1     2946     2460
## mu[22]       -0.6  -0.2   0.1  -0.2 0.2     1     2914     2192
## mu[23]       -0.6  -0.3   0.1  -0.3 0.2     1     2936     2268
## mu[24]       -1.0  -0.5   0.0  -0.5 0.3     1     3127     2342
## mu[25]       -0.2   0.1   0.3   0.1 0.2     1     2853     2135
## mu[26]        0.0   0.4   0.8   0.4 0.3     1     3052     2592
## mu[27]        0.0   0.4   0.8   0.4 0.3     1     3024     2647
## mu[28]        0.0   0.5   1.0   0.5 0.3     1     3085     2645
## mu[29]       -0.5  -0.2   0.1  -0.2 0.2     1     2897     1991
## mu[30]       -0.2   0.1   0.4   0.1 0.2     1     2874     2262
## mu[31]       -0.7  -0.3   0.1  -0.3 0.2     1     2952     2187
## mu[32]       -0.1   0.2   0.5   0.2 0.2     1     2948     2338
## log_lik[1]   -1.7  -1.4  -1.1  -1.4 0.2     1     3678     3213
## log_lik[2]   -1.4  -1.1  -0.9  -1.1 0.2     1     3238     2671
## log_lik[3]   -1.2  -0.9  -0.7  -0.9 0.1     1     2435     2284
## log_lik[4]   -1.5  -1.2  -0.9  -1.2 0.2     1     3221     2397
## log_lik[5]   -1.3  -1.0  -0.8  -1.0 0.1     1     2975     2016
## log_lik[6]   -2.4  -1.9  -1.5  -1.9 0.3     1     3364     2566
## log_lik[7]   -1.6  -1.2  -0.9  -1.2 0.2     1     3821     2692
## log_lik[8]   -1.8  -1.3  -1.0  -1.4 0.2     1     3609     2911
## log_lik[9]   -6.3  -4.4  -3.1  -4.5 1.0     1     3109     2291
## log_lik[10]  -1.2  -0.9  -0.7  -0.9 0.1     1     2424     2330
## log_lik[11]  -1.4  -1.1  -0.9  -1.1 0.1     1     2775     2088
## log_lik[12]  -1.2  -0.9  -0.7  -0.9 0.1     1     2396     2160
## log_lik[13]  -1.1  -0.9  -0.7  -0.9 0.1     1     2403     2314
## log_lik[14]  -1.2  -1.0  -0.8  -1.0 0.1     1     2405     1718
## log_lik[15]  -2.7  -1.5  -0.9  -1.6 0.6     1     3827     2957
## log_lik[16]  -2.5  -1.4  -0.9  -1.5 0.5     1     3823     2804
## log_lik[17]  -1.2  -0.9  -0.7  -0.9 0.1     1     2231     1837
## log_lik[18]  -1.3  -1.0  -0.7  -1.0 0.2     1     3250     2489
## log_lik[19]  -1.2  -0.9  -0.7  -0.9 0.2     1     2218     2228
## log_lik[20]  -1.6  -1.1  -0.8  -1.1 0.2     1     3598     2610
## log_lik[21]  -1.9  -1.5  -1.2  -1.5 0.2     1     3565     2913
## log_lik[22]  -1.2  -1.0  -0.7  -1.0 0.1     1     2615     2231
## log_lik[23]  -1.2  -0.9  -0.7  -0.9 0.1     1     2269     2116
## log_lik[24]  -1.9  -1.3  -1.0  -1.4 0.3     1     3876     3272
## log_lik[25]  -1.3  -1.0  -0.8  -1.0 0.1     1     2986     2081
## log_lik[26]  -1.2  -0.9  -0.7  -0.9 0.2     1     2441     2433
## log_lik[27]  -2.0  -1.5  -1.1  -1.5 0.3     1     3730     3082
## log_lik[28]  -2.1  -1.5  -1.1  -1.5 0.3     1     3798     3094
## log_lik[29]  -3.2  -2.4  -1.8  -2.4 0.4     1     2848     1868
## log_lik[30]  -2.6  -1.9  -1.5  -2.0 0.3     1     3108     2243
## log_lik[31]  -3.0  -2.2  -1.6  -2.2 0.4     1     3056     2017
## log_lik[32]  -1.2  -0.9  -0.7  -0.9 0.1     1     2484     2123
## lp__        -17.1 -14.4 -13.2 -14.7 1.3     1     1789     2123
```

```
##
## For each parameter, Bulk_ESS and Tail_ESS are crude measures of
## effective sample size for bulk and tail quantities respectively (an ESS > 100
## per chain is considered good), and Rhat is the potential scale reduction
## factor on rank normalized split chains (at convergence, Rhat <= 1.05).
```

```r
fit_cyl = stan(data = list(N=length(scaled_car_properties$qsec),
                  x=c(scaled_car_properties$cyl),
                  y=c(scaled_car_properties$qsec)),
                   model_code = stan_separate_model,refresh=0)
monitor(fit_cyl, probs = c(0.1, 0.5, 0.9))
```

```
## Inference for the input samples (4 chains: each with iter = 2000; warmup = 0):
##
##                Q5   Q50   Q95  Mean  SD  Rhat Bulk_ESS Tail_ESS
## alpha         -0.2   0.0   0.3   0.0 0.2     1     3487     2425
## beta          -0.8  -0.6 -0.3  -0.6 0.2     1     3239     2584
## sigma          0.7   0.8   1.1   0.9 0.1     1     3557     3062
## mu[1]         -0.2   0.1   0.3   0.1 0.2     1     3435     2374
## mu[2]         -0.2   0.1   0.3   0.1 0.2     1     3435     2374
## mu[3]          0.3   0.7   1.1   0.7 0.2     1     3172     2287
## mu[4]         -0.2   0.1   0.3   0.1 0.2     1     3435     2374
## mu[5]         -0.9  -0.6 -0.2  -0.6 0.2     1     3613     2718
## mu[6]         -0.2   0.1   0.3   0.1 0.2     1     3435     2374
## mu[7]         -0.9  -0.6 -0.2  -0.6 0.2     1     3613     2718
## mu[8]          0.3   0.7   1.1   0.7 0.2     1     3172     2287
## mu[9]          0.3   0.7   1.1   0.7 0.2     1     3172     2287
## mu[10]        -0.2   0.1   0.3   0.1 0.2     1     3435     2374
## mu[11]        -0.2   0.1   0.3   0.1 0.2     1     3435     2374
## mu[12]        -0.9  -0.6 -0.2  -0.6 0.2     1     3613     2718
## mu[13]        -0.9  -0.6 -0.2  -0.6 0.2     1     3613     2718
## mu[14]        -0.9  -0.6 -0.2  -0.6 0.2     1     3613     2718
## mu[15]        -0.9  -0.6 -0.2  -0.6 0.2     1     3613     2718
## mu[16]        -0.9  -0.6 -0.2  -0.6 0.2     1     3613     2718
## mu[17]        -0.9  -0.6 -0.2  -0.6 0.2     1     3613     2718
## mu[18]         0.3   0.7   1.1   0.7 0.2     1     3172     2287
## mu[19]         0.3   0.7   1.1   0.7 0.2     1     3172     2287
## mu[20]         0.3   0.7   1.1   0.7 0.2     1     3172     2287
## mu[21]         0.3   0.7   1.1   0.7 0.2     1     3172     2287
## mu[22]        -0.9  -0.6 -0.2  -0.6 0.2     1     3613     2718
## mu[23]        -0.9  -0.6 -0.2  -0.6 0.2     1     3613     2718
## mu[24]        -0.9  -0.6 -0.2  -0.6 0.2     1     3613     2718
## mu[25]        -0.9  -0.6 -0.2  -0.6 0.2     1     3613     2718
## mu[26]         0.3   0.7   1.1   0.7 0.2     1     3172     2287
## mu[27]         0.3   0.7   1.1   0.7 0.2     1     3172     2287
## mu[28]         0.3   0.7   1.1   0.7 0.2     1     3172     2287
## mu[29]        -0.9  -0.6 -0.2  -0.6 0.2     1     3613     2718
## mu[30]        -0.2   0.1   0.3   0.1 0.2     1     3435     2374
## mu[31]        -0.9  -0.6 -0.2  -0.6 0.2     1     3613     2718
## mu[32]         0.3   0.7   1.1   0.7 0.2     1     3172     2287
## log_lik[1]    -1.6  -1.3 -1.0  -1.3 0.2     1     3741     2972
## log_lik[2]    -1.2  -1.0 -0.8  -1.0 0.1     1     3436     2305
## log_lik[3]    -1.1  -0.8 -0.6  -0.9 0.2     1     3087     3062
## log_lik[4]    -1.6  -1.2 -1.0  -1.3 0.2     1     3710     2931
## log_lik[5]    -1.0  -0.8 -0.6  -0.8 0.1     1     2962     2617
```

```
## log_lik[6]    -2.5  -1.9 -1.5  -1.9 0.3     1     3621     2448
## log_lik[7]    -1.3  -1.0 -0.7  -1.0 0.2     1     3471     2719
## log_lik[8]    -1.3  -0.9 -0.7  -1.0 0.2     1     3324     2157
## log_lik[9]    -5.8  -3.9 -2.6  -4.0 1.0     1     3573     2749
## log_lik[10]   -1.0  -0.8 -0.6  -0.8 0.1     1     3217     2665
## log_lik[11]   -1.2  -1.0 -0.8  -1.0 0.1     1     3482     2316
## log_lik[12]   -1.1  -0.9 -0.6  -0.9 0.2     1     3323     2947
## log_lik[13]   -1.2  -0.9 -0.7  -0.9 0.2     1     3527     2707
## log_lik[14]   -1.5  -1.1 -0.8  -1.1 0.2     1     3901     2945
## log_lik[15]   -1.5  -1.1 -0.8  -1.1 0.2     1     3889     2868
## log_lik[16]   -1.4  -1.0 -0.7  -1.0 0.2     1     3779     2678
## log_lik[17]   -1.1  -0.9 -0.6  -0.9 0.2     1     3343     2900
## log_lik[18]   -1.1  -0.8 -0.6  -0.8 0.2     1     2815     2173
## log_lik[19]   -1.2  -0.9 -0.6  -0.9 0.2     1     3186     3193
## log_lik[20]   -1.3  -0.9 -0.7  -0.9 0.2     1     3235     2182
## log_lik[21]   -1.4  -0.9 -0.7  -1.0 0.2     1     3334     2207
## log_lik[22]   -1.0  -0.8 -0.6  -0.8 0.1     1     2907     2716
## log_lik[23]   -1.1  -0.8 -0.6  -0.8 0.2     1     3202     2870
## log_lik[24]   -1.7  -1.2 -0.9  -1.2 0.2     1     3863     2779
## log_lik[25]   -1.0  -0.8 -0.6  -0.8 0.1     1     2982     2652
## log_lik[26]   -1.1  -0.8 -0.6  -0.8 0.1     1     2800     2793
## log_lik[27]   -3.1  -2.1 -1.4  -2.1 0.5     1     3403     2273
## log_lik[28]   -2.7  -1.9 -1.3  -1.9 0.5     1     3430     2549
## log_lik[29]   -2.7  -1.9 -1.4  -2.0 0.4     1     3822     3030
## log_lik[30]   -2.8  -2.1 -1.6  -2.1 0.4     1     3543     2281
## log_lik[31]   -2.6  -1.8 -1.3  -1.9 0.4     1     3842     2893
## log_lik[32]   -1.1  -0.8 -0.6  -0.9 0.2     1     3096     3091
## lp__         -12.8 -10.1 -9.1 -10.4 1.2     1     1891     2414
##
## For each parameter, Bulk_ESS and Tail_ESS are crude measures of
## effective sample size for bulk and tail quantities respectively (an ESS > 100
## per chain is considered good), and Rhat is the potential scale reduction
## factor on rank normalized split chains (at convergence, Rhat <= 1.05).
```

```
fit_disp = stan(data = list(N=length(scaled_car_properties$qsec),
                   x=c(scaled_car_properties$disp),
                   y=c(scaled_car_properties$qsec)),
                    model_code = stan_separate_model,refresh=0)
monitor(fit_disp, probs = c(0.1, 0.5, 0.9))
```

```
## Inference for the input samples (4 chains: each with iter = 2000; warmup = 0):
##
##                 Q5   Q50   Q95  Mean  SD  Rhat Bulk_ESS Tail_ESS
## alpha         -0.3   0.0   0.3   0.0 0.2     1     2559     2254
## beta          -0.7  -0.4  -0.1  -0.4 0.2     1     2940     2579
## sigma          0.8   0.9   1.2   1.0 0.1     1     2752     2310
## mu[1]         -0.1   0.2   0.6   0.2 0.2     1     2939     2594
## mu[2]         -0.1   0.2   0.6   0.2 0.2     1     2939     2594
## mu[3]          0.0   0.4   0.8   0.4 0.2     1     3082     2512
## mu[4]         -0.4  -0.1   0.2  -0.1 0.2     1     2526     2210
## mu[5]         -0.9  -0.4   0.0  -0.4 0.2     1     2844     2563
## mu[6]         -0.3   0.0   0.3   0.0 0.2     1     2584     2184
## mu[7]         -0.9  -0.4   0.0  -0.4 0.2     1     2844     2563
## mu[8]         -0.1   0.3   0.6   0.3 0.2     1     2996     2582
## mu[9]          0.0   0.3   0.7   0.3 0.2     1     3016     2559
```

```
## mu[10]       -0.1  0.2  0.5  0.2 0.2   1    2901     2623
## mu[11]       -0.1  0.2  0.5  0.2 0.2   1    2901     2623
## mu[12]       -0.5 -0.2  0.1 -0.2 0.2   1    2558     2622
## mu[13]       -0.5 -0.2  0.1 -0.2 0.2   1    2558     2622
## mu[14]       -0.5 -0.2  0.1 -0.2 0.2   1    2558     2622
## mu[15]       -1.4 -0.8 -0.2 -0.8 0.4   1    2930     2625
## mu[16]       -1.4 -0.8 -0.2 -0.8 0.4   1    2932     2626
## mu[17]       -1.3 -0.7 -0.2 -0.7 0.3   1    2929     2654
## mu[18]        0.1  0.5  1.0  0.5 0.3   1    3101     2597
## mu[19]        0.1  0.5  1.0  0.5 0.3   1    3102     2580
## mu[20]        0.1  0.5  1.0  0.5 0.3   1    3104     2554
## mu[21]        0.0  0.4  0.8  0.4 0.2   1    3061     2455
## mu[22]       -0.6 -0.3  0.0 -0.3 0.2   1    2733     2519
## mu[23]       -0.6 -0.3  0.1 -0.3 0.2   1    2673     2555
## mu[24]       -0.8 -0.4  0.0 -0.4 0.2   1    2818     2598
## mu[25]       -1.1 -0.6 -0.1 -0.6 0.3   1    2903     2617
## mu[26]        0.1  0.5  1.0  0.5 0.3   1    3101     2579
## mu[27]        0.0  0.4  0.7  0.4 0.2   1    3061     2455
## mu[28]        0.0  0.5  0.9  0.5 0.3   1    3104     2534
## mu[29]       -0.8 -0.4  0.0 -0.4 0.2   1    2818     2598
## mu[30]       -0.1  0.3  0.6  0.3 0.2   1    3003     2492
## mu[31]       -0.6 -0.2  0.1 -0.2 0.2   1    2666     2644
## mu[32]        0.0  0.4  0.7  0.4 0.2   1    3061     2425
## log_lik[1]   -1.9 -1.5 -1.2 -1.5 0.2   1    3690     2911
## log_lik[2]   -1.5 -1.2 -0.9 -1.2 0.2   1    3518     2831
## log_lik[3]   -1.1 -0.9 -0.7 -0.9 0.1   1    2434     2271
## log_lik[4]   -1.8 -1.4 -1.2 -1.4 0.2   1    3064     3053
## log_lik[5]   -1.1 -0.9 -0.7 -0.9 0.1   1    2348     2143
## log_lik[6]   -2.4 -1.8 -1.5 -1.9 0.3   1    3255     2976
## log_lik[7]   -1.5 -1.1 -0.9 -1.2 0.2   1    3169     2672
## log_lik[8]   -1.7 -1.4 -1.1 -1.4 0.2   1    3650     2979
## log_lik[9]   -6.4 -4.4 -3.1 -4.6 1.0   1    2907     2703
## log_lik[10]  -1.1 -0.9 -0.7 -0.9 0.1   1    2536     2282
## log_lik[11]  -1.2 -1.0 -0.7 -1.0 0.1   1    2624     2161
## log_lik[12]  -1.1 -0.9 -0.7 -0.9 0.1   1    2519     2228
## log_lik[13]  -1.1 -0.9 -0.7 -0.9 0.1   1    2498     2230
## log_lik[14]  -1.2 -0.9 -0.7 -0.9 0.1   1    2525     2035
## log_lik[15]  -2.2 -1.3 -0.9 -1.4 0.4   1    3408     3197
## log_lik[16]  -1.9 -1.2 -0.9 -1.3 0.3   1    3342     2924
## log_lik[17]  -1.5 -1.0 -0.8 -1.1 0.2   1    2929     2752
## log_lik[18]  -1.3 -1.0 -0.7 -1.0 0.2   1    2774     2533
## log_lik[19]  -1.2 -0.9 -0.7 -0.9 0.2   1    2341     2512
## log_lik[20]  -1.5 -1.1 -0.8 -1.1 0.2   1    3177     2674
## log_lik[21]  -1.7 -1.3 -1.0 -1.3 0.2   1    3644     2964
## log_lik[22]  -1.2 -0.9 -0.7 -0.9 0.1   1    2568     2193
## log_lik[23]  -1.1 -0.9 -0.7 -0.9 0.1   1    2471     2203
## log_lik[24]  -1.9 -1.4 -1.0 -1.4 0.3   1    3318     3103
## log_lik[25]  -1.2 -0.9 -0.7 -0.9 0.2   1    2298     2195
## log_lik[26]  -1.2 -0.9 -0.7 -0.9 0.1   1    2419     2304
## log_lik[27]  -2.0 -1.5 -1.1 -1.5 0.3   1    3699     2779
## log_lik[28]  -2.0 -1.4 -1.1 -1.5 0.3   1    3699     2764
## log_lik[29]  -2.9 -2.1 -1.5 -2.1 0.4   1    3088     2649
## log_lik[30]  -3.2 -2.3 -1.7 -2.4 0.5   1    2935     2422
## log_lik[31]  -3.1 -2.3 -1.7 -2.3 0.4   1    2989     2293
```

```
## log_lik[32]  -1.1  -0.9  -0.7  -0.9 0.1     1     2469     2220
## lp__        -16.3 -13.6 -12.5 -13.9 1.3     1     1904     2146
##
## For each parameter, Bulk_ESS and Tail_ESS are crude measures of
## effective sample size for bulk and tail quantities respectively (an ESS > 100
## per chain is considered good), and Rhat is the potential scale reduction
## factor on rank normalized split chains (at convergence, Rhat <= 1.05).
```

```r
fit_hp = stan(data = list(N=length(scaled_car_properties$qsec),
                   x=c(scaled_car_properties$hp),
                   y=c(scaled_car_properties$qsec)),
                    model_code = stan_separate_model,refresh=0)
monitor(fit_hp, probs = c(0.1, 0.5, 0.9))
```

```
## Inference for the input samples (4 chains: each with iter = 2000; warmup = 0):
##
##                Q5  Q50  Q95 Mean  SD  Rhat Bulk_ESS Tail_ESS
## alpha        -0.2  0.0  0.2  0.0 0.1  1.00     3042     2281
## beta         -0.9 -0.7 -0.5 -0.7 0.1  1.00     3156     2346
## sigma         0.6  0.7  0.9  0.7 0.1  1.00     3012     2346
## mu[1]         0.1  0.4  0.6  0.4 0.2  1.00     2659     2125
## mu[2]         0.1  0.4  0.6  0.4 0.2  1.00     2659     2125
## mu[3]         0.3  0.6  0.8  0.6 0.2  1.00     2604     2195
## mu[4]         0.1  0.4  0.6  0.4 0.2  1.00     2659     2125
## mu[5]        -0.5 -0.3 -0.1 -0.3 0.1  1.00     3185     2488
## mu[6]         0.2  0.4  0.7  0.4 0.2  1.00     2632     2137
## mu[7]        -1.4 -1.0 -0.6 -1.0 0.2  1.00     3328     2426
## mu[8]         0.5  0.9  1.2  0.9 0.2  1.00     2604     2003
## mu[9]         0.2  0.5  0.8  0.5 0.2  1.00     2610     2183
## mu[10]        0.0  0.2  0.5  0.2 0.1  1.00     2774     2149
## mu[11]        0.0  0.2  0.5  0.2 0.1  1.00     2774     2149
## mu[12]       -0.6 -0.3 -0.1 -0.3 0.1  1.00     3211     2534
## mu[13]       -0.6 -0.3 -0.1 -0.3 0.1  1.00     3211     2534
## mu[14]       -0.6 -0.3 -0.1 -0.3 0.1  1.00     3211     2534
## mu[15]       -0.9 -0.6 -0.3 -0.6 0.2  1.00     3311     2543
## mu[16]       -1.0 -0.7 -0.4 -0.7 0.2  1.00     3322     2459
## mu[17]       -1.2 -0.9 -0.5 -0.9 0.2  1.00     3324     2394
## mu[18]        0.5  0.8  1.2  0.8 0.2  1.00     2600     2032
## mu[19]        0.6  1.0  1.4  1.0 0.2  1.00     2617     2049
## mu[20]        0.5  0.8  1.2  0.8 0.2  1.00     2601     1993
## mu[21]        0.2  0.5  0.8  0.5 0.2  1.00     2613     2158
## mu[22]       -0.3  0.0  0.2  0.0 0.1  1.00     3079     2347
## mu[23]       -0.3  0.0  0.2  0.0 0.1  1.00     3079     2347
## mu[24]       -1.4 -1.0 -0.6 -1.0 0.2  1.00     3328     2426
## mu[25]       -0.5 -0.3 -0.1 -0.3 0.1  1.00     3185     2488
## mu[26]        0.5  0.8  1.2  0.8 0.2  1.00     2600     2032
## mu[27]        0.3  0.6  0.9  0.6 0.2  1.00     2602     2220
## mu[28]        0.1  0.3  0.6  0.3 0.2  1.00     2684     2152
## mu[29]       -1.7 -1.2 -0.8 -1.2 0.3  1.00     3317     2458
## mu[30]       -0.5 -0.3 -0.1 -0.3 0.1  1.00     3185     2488
## mu[31]       -2.6 -1.9 -1.3 -1.9 0.4  1.00     3287     2448
## mu[32]        0.1  0.4  0.6  0.4 0.2  1.00     2651     2114
## log_lik[1]   -2.6 -1.8 -1.3 -1.9 0.4  1.00     2911     2145
## log_lik[2]   -1.7 -1.3 -1.0 -1.3 0.2  1.00     3380     2704
## log_lik[3]   -0.9 -0.7 -0.4 -0.7 0.1  1.00     2570     2176
```

```
## log_lik[4]  -1.2 -0.9 -0.6 -0.9 0.2  1.00     2930     2128
## log_lik[5]  -0.9 -0.7 -0.5 -0.7 0.1  1.00     2731     2128
## log_lik[6]  -1.9 -1.4 -1.0 -1.4 0.3  1.00     3094     2264
## log_lik[7]  -1.0 -0.7 -0.4 -0.7 0.2  1.00     2317     2243
## log_lik[8]  -1.1 -0.8 -0.5 -0.8 0.2  1.00     2619     2074
## log_lik[9]  -8.0 -5.5 -3.6 -5.6 1.4  1.00     3142     2084
## log_lik[10] -0.9 -0.6 -0.4 -0.6 0.1  1.00     2671     2285
## log_lik[11] -1.0 -0.7 -0.5 -0.8 0.1  1.00     2825     2018
## log_lik[12] -0.9 -0.6 -0.4 -0.6 0.1  1.00     2738     2165
## log_lik[13] -0.9 -0.7 -0.5 -0.7 0.1  1.00     2854     2320
## log_lik[14] -1.1 -0.8 -0.6 -0.8 0.1  1.00     3133     2457
## log_lik[15] -1.5 -1.1 -0.8 -1.1 0.2  1.00     3560     3072
## log_lik[16] -1.5 -1.1 -0.8 -1.1 0.2  1.00     3613     2928
## log_lik[17] -1.5 -1.0 -0.7 -1.0 0.2  1.00     3610     2931
## log_lik[18] -0.9 -0.7 -0.4 -0.7 0.1  1.00     2307     1879
## log_lik[19] -1.5 -1.0 -0.6 -1.0 0.3  1.00     3274     2303
## log_lik[20] -1.1 -0.7 -0.5 -0.8 0.2  1.00     2587     1999
## log_lik[21] -1.5 -1.1 -0.8 -1.1 0.2  1.00     3106     2690
## log_lik[22] -1.1 -0.9 -0.7 -0.9 0.1  1.00     3145     2231
## log_lik[23] -0.9 -0.7 -0.5 -0.7 0.1  1.00     2851     2372
## log_lik[24] -1.1 -0.8 -0.5 -0.8 0.2  1.00     2885     2473
## log_lik[25] -0.9 -0.7 -0.5 -0.7 0.1  1.00     2725     2293
## log_lik[26] -1.0 -0.7 -0.5 -0.7 0.2  1.00     2595     1883
## log_lik[27] -2.9 -2.0 -1.4 -2.0 0.5  1.00     2869     2168
## log_lik[28] -1.8 -1.3 -1.0 -1.4 0.3  1.00     3342     2725
## log_lik[29] -1.7 -1.0 -0.7 -1.1 0.3  1.00     3509     2605
## log_lik[30] -2.2 -1.6 -1.2 -1.6 0.3  1.00     3424     2851
## log_lik[31] -1.2 -0.7 -0.5 -0.8 0.2  1.00     1918     2011
## log_lik[32] -0.9 -0.6 -0.4 -0.6 0.1  1.00     2606     2160
## lp__        -9.0 -6.0 -5.0 -6.4 1.3  1.01     1553     1940
##
## For each parameter, Bulk_ESS and Tail_ESS are crude measures of
## effective sample size for bulk and tail quantities respectively (an ESS > 100
## per chain is considered good), and Rhat is the potential scale reduction
## factor on rank normalized split chains (at convergence, Rhat <= 1.05).
```

```r
fit_drat = stan(data = list(N=length(scaled_car_properties$qsec),
                   x=c(scaled_car_properties$drat),
                   y=c(scaled_car_properties$qsec)),
                    model_code = stan_separate_model,refresh=0)
monitor(fit_drat, probs = c(0.1, 0.5, 0.9))
```

```
## Inference for the input samples (4 chains: each with iter = 2000; warmup = 0):
##
##                Q5   Q50   Q95  Mean  SD  Rhat Bulk_ESS Tail_ESS
## alpha        -0.3   0.0   0.3   0.0 0.2  1.00     2852     2376
## beta         -0.2   0.1   0.4   0.1 0.2  1.00     2684     2585
## sigma         0.9   1.0   1.3   1.1 0.1  1.00     2974     1914
## mu[1]        -0.3   0.0   0.4   0.0 0.2  1.00     3061     2530
## mu[2]        -0.3   0.0   0.4   0.0 0.2  1.00     3061     2530
## mu[3]        -0.3   0.0   0.4   0.0 0.2  1.00     3044     2510
## mu[4]        -0.5  -0.1   0.4  -0.1 0.3  1.00     2521     2161
## mu[5]        -0.5  -0.1   0.3  -0.1 0.2  1.00     2530     2294
## mu[6]        -0.7  -0.1   0.4  -0.1 0.4  1.00     2468     2102
## mu[7]        -0.5  -0.1   0.3  -0.1 0.2  1.00     2550     2291
```

```
## mu[8]         -0.3   0.0   0.3   0.0 0.2  1.00     2991     2503
## mu[9]         -0.3   0.0   0.4   0.0 0.2  1.00     3069     2598
## mu[10]        -0.3   0.0   0.4   0.0 0.2  1.00     3069     2598
## mu[11]        -0.3   0.0   0.4   0.0 0.2  1.00     3069     2598
## mu[12]        -0.5  -0.1   0.4  -0.1 0.3  1.00     2518     2155
## mu[13]        -0.5  -0.1   0.4  -0.1 0.3  1.00     2518     2155
## mu[14]        -0.5  -0.1   0.4  -0.1 0.3  1.00     2518     2155
## mu[15]        -0.6  -0.1   0.4  -0.1 0.3  1.00     2498     2082
## mu[16]        -0.6  -0.1   0.4  -0.1 0.3  1.00     2516     2170
## mu[17]        -0.4  -0.1   0.3  -0.1 0.2  1.00     2558     2424
## mu[18]        -0.3   0.1   0.5   0.1 0.3  1.01     3099     2512
## mu[19]        -0.6   0.2   1.1   0.2 0.5  1.00     3035     2563
## mu[20]        -0.4   0.1   0.6   0.1 0.3  1.00     3106     2620
## mu[21]        -0.3   0.0   0.3   0.0 0.2  1.00     3001     2455
## mu[22]        -0.7  -0.1   0.4  -0.1 0.4  1.00     2468     2102
## mu[23]        -0.5  -0.1   0.3  -0.1 0.2  1.00     2530     2294
## mu[24]        -0.3   0.0   0.3   0.0 0.2  1.00     2979     2519
## mu[25]        -0.5  -0.1   0.4  -0.1 0.3  1.00     2521     2161
## mu[26]        -0.3   0.1   0.5   0.1 0.3  1.01     3099     2512
## mu[27]        -0.4   0.1   0.7   0.1 0.4  1.00     3100     2562
## mu[28]        -0.3   0.0   0.4   0.0 0.2  1.00     3002     2469
## mu[29]        -0.4   0.1   0.6   0.1 0.3  1.00     3106     2620
## mu[30]        -0.3   0.0   0.3   0.0 0.2  1.00     2876     2377
## mu[31]        -0.3   0.0   0.3   0.0 0.2  1.00     2792     2165
## mu[32]        -0.4   0.1   0.5   0.1 0.3  1.01     3102     2498
## log_lik[1]    -1.6  -1.3  -1.0  -1.3 0.2  1.00     3330     2599
## log_lik[2]    -1.4  -1.1  -0.9  -1.1 0.1  1.00     2977     1968
## log_lik[3]    -1.3  -1.0  -0.8  -1.1 0.1  1.00     2735     1861
## log_lik[4]    -1.9  -1.4  -1.1  -1.4 0.2  1.00     3154     3078
## log_lik[5]    -1.3  -1.1  -0.8  -1.1 0.1  1.00     2508     1840
## log_lik[6]    -2.9  -2.0  -1.3  -2.0 0.5  1.00     2714     2496
## log_lik[7]    -1.9  -1.5  -1.2  -1.5 0.2  1.00     2947     3331
## log_lik[8]    -2.0  -1.6  -1.3  -1.7 0.2  1.00     3392     2828
## log_lik[9]    -6.3  -4.5  -3.2  -4.6 1.0  1.00     3100     2032
## log_lik[10]   -1.3  -1.0  -0.8  -1.0 0.1  1.00     2558     1907
## log_lik[11]   -1.4  -1.1  -0.9  -1.1 0.2  1.00     3115     2196
## log_lik[12]   -1.3  -1.0  -0.8  -1.0 0.1  1.00     2499     1660
## log_lik[13]   -1.2  -1.0  -0.8  -1.0 0.1  1.00     2571     1915
## log_lik[14]   -1.3  -1.0  -0.8  -1.0 0.1  1.00     2642     2060
## log_lik[15]   -1.3  -1.0  -0.8  -1.0 0.2  1.00     2576     2042
## log_lik[16]   -1.3  -1.0  -0.8  -1.0 0.1  1.00     2555     1930
## log_lik[17]   -1.2  -1.0  -0.8  -1.0 0.1  1.00     2547     2023
## log_lik[18]   -1.7  -1.3  -1.0  -1.3 0.2  1.00     3447     2813
## log_lik[19]   -1.5  -1.1  -0.8  -1.1 0.2  1.00     2099     2341
## log_lik[20]   -2.0  -1.5  -1.1  -1.5 0.3  1.00     3486     3083
## log_lik[21]   -2.0  -1.6  -1.3  -1.7 0.2  1.00     3393     2833
## log_lik[22]   -1.4  -1.1  -0.8  -1.1 0.2  1.00     2472     1869
## log_lik[23]   -1.3  -1.0  -0.8  -1.0 0.1  1.00     2513     1776
## log_lik[24]   -2.4  -1.9  -1.5  -1.9 0.3  1.00     3337     2866
## log_lik[25]   -1.3  -1.0  -0.8  -1.1 0.2  1.00     2485     1820
## log_lik[26]   -1.4  -1.1  -0.9  -1.1 0.2  1.00     3035     2465
## log_lik[27]   -1.8  -1.3  -0.9  -1.3 0.3  1.01     3270     2657
## log_lik[28]   -1.4  -1.1  -0.9  -1.1 0.1  1.00     3006     2171
## log_lik[29]   -4.0  -2.7  -1.9  -2.8 0.6  1.00     3096     2417
```

```
## log_lik[30]  -2.2  -1.8  -1.4  -1.8 0.2  1.00     3323     2885
## log_lik[31]  -3.3  -2.5  -1.9  -2.5 0.4  1.00     3174     2016
## log_lik[32]  -1.3  -1.0  -0.8  -1.1 0.2  1.00     2641     2103
## lp__        -19.7 -16.6 -15.6 -17.0 1.3  1.00     1643     2140
##
## For each parameter, Bulk_ESS and Tail_ESS are crude measures of
## effective sample size for bulk and tail quantities respectively (an ESS > 100
## per chain is considered good), and Rhat is the potential scale reduction
## factor on rank normalized split chains (at convergence, Rhat <= 1.05).
```

```
fit_wt = stan(data = list(N=length(scaled_car_properties$qsec),
                    x=c(scaled_car_properties$wt),
                    y=c(scaled_car_properties$qsec)),
                     model_code = stan_separate_model,refresh=0)
monitor(fit_wt, probs = c(0.1, 0.5, 0.9))
```

```
## Inference for the input samples (4 chains: each with iter = 2000; warmup = 0):
##
##                Q5   Q50   Q95  Mean  SD  Rhat Bulk_ESS Tail_ESS
## alpha        -0.3   0.0   0.3   0.0 0.2     1     3938     2786
## beta         -0.5  -0.2   0.1  -0.2 0.2     1     3695     2433
## sigma         0.8   1.0   1.3   1.0 0.1     1     3569     2882
## mu[1]        -0.2   0.1   0.5   0.1 0.2     1     3758     2851
## mu[2]        -0.3   0.1   0.4   0.1 0.2     1     3879     2944
## mu[3]        -0.3   0.2   0.6   0.2 0.3     1     3676     2872
## mu[4]        -0.3   0.0   0.3   0.0 0.2     1     3938     2786
## mu[5]        -0.3   0.0   0.3   0.0 0.2     1     3909     2730
## mu[6]        -0.4   0.0   0.3   0.0 0.2     1     3908     2811
## mu[7]        -0.4  -0.1   0.3  -0.1 0.2     1     3896     2767
## mu[8]        -0.3   0.0   0.3   0.0 0.2     1     3935     2848
## mu[9]        -0.3   0.0   0.3   0.0 0.2     1     3935     2936
## mu[10]       -0.3   0.0   0.3   0.0 0.2     1     3909     2730
## mu[11]       -0.3   0.0   0.3   0.0 0.2     1     3909     2730
## mu[12]       -0.6  -0.2   0.3  -0.2 0.2     1     3816     2783
## mu[13]       -0.4  -0.1   0.2  -0.1 0.2     1     3870     2728
## mu[14]       -0.4  -0.1   0.2  -0.1 0.2     1     3862     2789
## mu[15]       -1.1  -0.4   0.4  -0.4 0.4     1     3741     2632
## mu[16]       -1.1  -0.4   0.4  -0.4 0.5     1     3732     2632
## mu[17]       -1.1  -0.4   0.4  -0.4 0.4     1     3737     2596
## mu[18]       -0.3   0.2   0.6   0.2 0.3     1     3659     2674
## mu[19]       -0.3   0.3   0.9   0.3 0.4     1     3618     2541
## mu[20]       -0.3   0.2   0.8   0.2 0.3     1     3625     2605
## mu[21]       -0.3   0.1   0.5   0.1 0.2     1     3706     2887
## mu[22]       -0.4  -0.1   0.3  -0.1 0.2     1     3903     2747
## mu[23]       -0.3   0.0   0.3   0.0 0.2     1     3910     2722
## mu[24]       -0.5  -0.1   0.2  -0.1 0.2     1     3847     2875
## mu[25]       -0.5  -0.1   0.2  -0.1 0.2     1     3846     2875
## mu[26]       -0.3   0.2   0.7   0.2 0.3     1     3632     2561
## mu[27]       -0.3   0.2   0.6   0.2 0.3     1     3649     2747
## mu[28]       -0.3   0.3   0.9   0.3 0.4     1     3620     2543
## mu[29]       -0.3   0.0   0.3   0.0 0.2     1     3934     2912
## mu[30]       -0.3   0.1   0.4   0.1 0.2     1     3828     2989
## mu[31]       -0.4  -0.1   0.3  -0.1 0.2     1     3896     2767
## mu[32]       -0.3   0.1   0.4   0.1 0.2     1     3831     2966
## log_lik[1]   -1.7  -1.3  -1.1  -1.3 0.2     1     4134     3103
```

```
## log_lik[2]   -1.3  -1.1  -0.9  -1.1 0.1     1     3595     2864
## log_lik[3]   -1.3  -1.0  -0.8  -1.0 0.1     1     3137     2708
## log_lik[4]   -1.6  -1.3  -1.1  -1.4 0.2     1     4150     2987
## log_lik[5]   -1.3  -1.0  -0.8  -1.1 0.1     1     3336     2630
## log_lik[6]   -2.4  -1.9  -1.5  -1.9 0.3     1     3965     2791
## log_lik[7]   -1.9  -1.5  -1.2  -1.5 0.2     1     4102     3163
## log_lik[8]   -2.1  -1.6  -1.4  -1.7 0.2     1     4103     2830
## log_lik[9]   -6.7  -4.7  -3.4  -4.8 1.0     1     3554     2712
## log_lik[10]  -1.2  -1.0  -0.8  -1.0 0.1     1     3466     2719
## log_lik[11]  -1.4  -1.2  -0.9  -1.2 0.1     1     3845     2812
## log_lik[12]  -1.2  -1.0  -0.8  -1.0 0.1     1     2968     2750
## log_lik[13]  -1.2  -1.0  -0.8  -1.0 0.1     1     3149     2900
## log_lik[14]  -1.2  -1.0  -0.8  -1.0 0.1     1     3341     2813
## log_lik[15]  -1.6  -1.1  -0.8  -1.1 0.2     1     3421     2922
## log_lik[16]  -1.6  -1.1  -0.8  -1.1 0.2     1     3202     2834
## log_lik[17]  -1.4  -1.0  -0.8  -1.0 0.2     1     2541     2683
## log_lik[18]  -1.6  -1.2  -1.0  -1.2 0.2     1     3853     2880
## log_lik[19]  -1.3  -1.0  -0.8  -1.0 0.2     1     2631     2606
## log_lik[20]  -1.9  -1.4  -1.0  -1.4 0.3     1     3962     2971
## log_lik[21]  -2.0  -1.5  -1.2  -1.5 0.2     1     4118     3185
## log_lik[22]  -1.3  -1.1  -0.9  -1.1 0.1     1     3401     2705
## log_lik[23]  -1.2  -1.0  -0.8  -1.0 0.1     1     3223     2787
## log_lik[24]  -2.2  -1.7  -1.3  -1.7 0.3     1     4009     3195
## log_lik[25]  -1.3  -1.0  -0.8  -1.0 0.1     1     3134     2748
## log_lik[26]  -1.3  -1.0  -0.8  -1.1 0.2     1     3138     2651
## log_lik[27]  -1.7  -1.3  -1.0  -1.3 0.2     1     4052     3088
## log_lik[28]  -1.9  -1.3  -0.9  -1.3 0.3     1     3987     3176
## log_lik[29]  -3.5  -2.6  -2.0  -2.7 0.5     1     3822     2847
## log_lik[30]  -2.4  -1.9  -1.5  -1.9 0.3     1     4044     3060
## log_lik[31]  -3.2  -2.4  -1.9  -2.5 0.4     1     3838     2879
## log_lik[32]  -1.3  -1.0  -0.8  -1.0 0.1     1     3424     2849
## lp__        -18.9 -16.3 -15.2 -16.6 1.2     1     1909     2750
##
## For each parameter, Bulk_ESS and Tail_ESS are crude measures of
## effective sample size for bulk and tail quantities respectively (an ESS > 100
## per chain is considered good), and Rhat is the potential scale reduction
## factor on rank normalized split chains (at convergence, Rhat <= 1.05).
```

```
fit_vs = stan(data = list(N=length(scaled_car_properties$qsec),
                   x=c(scaled_car_properties$vs),
                   y=c(scaled_car_properties$qsec)),
                    model_code = stan_separate_model,refresh=0)
monitor(fit_vs, probs = c(0.1, 0.5, 0.9))
```

```
## Inference for the input samples (4 chains: each with iter = 2000; warmup = 0):
##
##               Q5  Q50  Q95 Mean  SD  Rhat Bulk_ESS Tail_ESS
## alpha       -0.2  0.0  0.2  0.0 0.1  1.00     3495     2473
## beta         0.5  0.7  1.0  0.7 0.1  1.00     3288     2601
## sigma        0.6  0.7  0.9  0.7 0.1  1.00     3286     2809
## mu[1]       -0.9 -0.6 -0.4 -0.6 0.2  1.00     3622     2849
## mu[2]       -0.9 -0.6 -0.4 -0.6 0.2  1.00     3622     2849
## mu[3]        0.5  0.8  1.1  0.8 0.2  1.00     3092     2666
## mu[4]        0.5  0.8  1.1  0.8 0.2  1.00     3092     2666
## mu[5]       -0.9 -0.6 -0.4 -0.6 0.2  1.00     3622     2849
```

```
## mu[6]         0.5  0.8  1.1  0.8 0.2  1.00     3092     2666
## mu[7]        -0.9 -0.6 -0.4 -0.6 0.2  1.00     3622     2849
## mu[8]         0.5  0.8  1.1  0.8 0.2  1.00     3092     2666
## mu[9]         0.5  0.8  1.1  0.8 0.2  1.00     3092     2666
## mu[10]        0.5  0.8  1.1  0.8 0.2  1.00     3092     2666
## mu[11]        0.5  0.8  1.1  0.8 0.2  1.00     3092     2666
## mu[12]       -0.9 -0.6 -0.4 -0.6 0.2  1.00     3622     2849
## mu[13]       -0.9 -0.6 -0.4 -0.6 0.2  1.00     3622     2849
## mu[14]       -0.9 -0.6 -0.4 -0.6 0.2  1.00     3622     2849
## mu[15]       -0.9 -0.6 -0.4 -0.6 0.2  1.00     3622     2849
## mu[16]       -0.9 -0.6 -0.4 -0.6 0.2  1.00     3622     2849
## mu[17]       -0.9 -0.6 -0.4 -0.6 0.2  1.00     3622     2849
## mu[18]        0.5  0.8  1.1  0.8 0.2  1.00     3092     2666
## mu[19]        0.5  0.8  1.1  0.8 0.2  1.00     3092     2666
## mu[20]        0.5  0.8  1.1  0.8 0.2  1.00     3092     2666
## mu[21]        0.5  0.8  1.1  0.8 0.2  1.00     3092     2666
## mu[22]       -0.9 -0.6 -0.4 -0.6 0.2  1.00     3622     2849
## mu[23]       -0.9 -0.6 -0.4 -0.6 0.2  1.00     3622     2849
## mu[24]       -0.9 -0.6 -0.4 -0.6 0.2  1.00     3622     2849
## mu[25]       -0.9 -0.6 -0.4 -0.6 0.2  1.00     3622     2849
## mu[26]        0.5  0.8  1.1  0.8 0.2  1.00     3092     2666
## mu[27]       -0.9 -0.6 -0.4 -0.6 0.2  1.00     3622     2849
## mu[28]        0.5  0.8  1.1  0.8 0.2  1.00     3092     2666
## mu[29]       -0.9 -0.6 -0.4 -0.6 0.2  1.00     3622     2849
## mu[30]       -0.9 -0.6 -0.4 -0.6 0.2  1.00     3622     2849
## mu[31]       -0.9 -0.6 -0.4 -0.6 0.2  1.00     3622     2849
## mu[32]        0.5  0.8  1.1  0.8 0.2  1.00     3092     2666
## log_lik[1]   -0.8 -0.6 -0.4 -0.6 0.1  1.00     2855     2902
## log_lik[2]   -0.9 -0.6 -0.4 -0.6 0.1  1.00     2996     2314
## log_lik[3]   -1.1 -0.7 -0.5 -0.8 0.2  1.00     3130     2753
## log_lik[4]   -0.9 -0.6 -0.4 -0.6 0.1  1.00     2731     2475
## log_lik[5]   -0.9 -0.6 -0.4 -0.6 0.1  1.00     2996     2314
## log_lik[6]   -1.2 -0.8 -0.6 -0.9 0.2  1.00     3522     2845
## log_lik[7]   -1.2 -0.8 -0.6 -0.8 0.2  1.00     3653     3096
## log_lik[8]   -1.1 -0.7 -0.5 -0.8 0.2  1.00     3309     2708
## log_lik[9]   -7.1 -4.7 -3.1 -4.8 1.2  1.00     3141     2528
## log_lik[10] -1.4 -0.9 -0.6 -0.9 0.2  1.00     3453     2774
## log_lik[11] -0.9 -0.6 -0.4 -0.7 0.2  1.00     2816     2416
## log_lik[12] -1.0 -0.7 -0.5 -0.8 0.2  1.01     3549     2479
## log_lik[13] -1.2 -0.9 -0.6 -0.9 0.2  1.01     3856     2873
## log_lik[14] -1.6 -1.1 -0.8 -1.1 0.2  1.00     4030     3260
## log_lik[15] -1.6 -1.1 -0.8 -1.1 0.2  1.00     4034     3189
## log_lik[16] -1.4 -1.0 -0.7 -1.0 0.2  1.00     4010     2874
## log_lik[17] -1.1 -0.8 -0.5 -0.8 0.2  1.01     3583     2496
## log_lik[18] -0.9 -0.6 -0.4 -0.6 0.1  1.00     2745     2507
## log_lik[19] -1.2 -0.8 -0.5 -0.8 0.2  1.00     3255     2788
## log_lik[20] -1.0 -0.7 -0.5 -0.7 0.2  1.00     3190     2567
## log_lik[21] -1.1 -0.7 -0.5 -0.8 0.2  1.00     3321     2708
## log_lik[22] -0.8 -0.6 -0.4 -0.6 0.1  1.00     2874     2436
## log_lik[23] -1.0 -0.7 -0.5 -0.7 0.2  1.00     3381     2363
## log_lik[24] -1.6 -1.1 -0.8 -1.1 0.2  1.00     4028     3173
## log_lik[25] -0.9 -0.6 -0.4 -0.6 0.1  1.00     3026     2296
## log_lik[26] -0.9 -0.6 -0.4 -0.7 0.2  1.00     2816     2416
## log_lik[27] -0.8 -0.6 -0.4 -0.6 0.1  1.00     2812     2573
```

```
## log_lik[28] -3.7 -2.4 -1.6 -2.5 0.6  1.00     3331     2417
## log_lik[29] -3.1 -2.1 -1.5 -2.2 0.5  1.00     3657     2597
## log_lik[30] -1.5 -1.0 -0.7 -1.1 0.2  1.00     4003     3073
## log_lik[31] -2.9 -2.0 -1.4 -2.0 0.5  1.00     3701     2747
## log_lik[32] -1.1 -0.8 -0.5 -0.8 0.2  1.00     3142     2735
## lp__        -7.1 -4.3 -3.2 -4.6 1.3  1.00     1784     2477
##
## For each parameter, Bulk_ESS and Tail_ESS are crude measures of
## effective sample size for bulk and tail quantities respectively (an ESS > 100
## per chain is considered good), and Rhat is the potential scale reduction
## factor on rank normalized split chains (at convergence, Rhat <= 1.05).
```

```r
fit_am = stan(data = list(N=length(scaled_car_properties$qsec),
                   x=c(scaled_car_properties$am),
                   y=c(scaled_car_properties$qsec)),
                    model_code = stan_separate_model,refresh=0)
monitor(fit_am, probs = c(0.1, 0.5, 0.9))
```

```
## Inference for the input samples (4 chains: each with iter = 2000; warmup = 0):
##
##                 Q5   Q50   Q95  Mean  SD  Rhat Bulk_ESS Tail_ESS
## alpha         -0.3   0.0   0.3   0.0 0.2     1     3373     2606
## beta          -0.5  -0.2   0.1  -0.2 0.2     1     3175     2283
## sigma          0.8   1.0   1.3   1.0 0.1     1     3324     2962
## mu[1]         -0.7  -0.3   0.2  -0.3 0.3     1     3387     2878
## mu[2]         -0.7  -0.3   0.2  -0.3 0.3     1     3387     2878
## mu[3]         -0.7  -0.3   0.2  -0.3 0.3     1     3387     2878
## mu[4]         -0.2   0.2   0.6   0.2 0.2     1     3157     2282
## mu[5]         -0.2   0.2   0.6   0.2 0.2     1     3157     2282
## mu[6]         -0.2   0.2   0.6   0.2 0.2     1     3157     2282
## mu[7]         -0.2   0.2   0.6   0.2 0.2     1     3157     2282
## mu[8]         -0.2   0.2   0.6   0.2 0.2     1     3157     2282
## mu[9]         -0.2   0.2   0.6   0.2 0.2     1     3157     2282
## mu[10]        -0.2   0.2   0.6   0.2 0.2     1     3157     2282
## mu[11]        -0.2   0.2   0.6   0.2 0.2     1     3157     2282
## mu[12]        -0.2   0.2   0.6   0.2 0.2     1     3157     2282
## mu[13]        -0.2   0.2   0.6   0.2 0.2     1     3157     2282
## mu[14]        -0.2   0.2   0.6   0.2 0.2     1     3157     2282
## mu[15]        -0.2   0.2   0.6   0.2 0.2     1     3157     2282
## mu[16]        -0.2   0.2   0.6   0.2 0.2     1     3157     2282
## mu[17]        -0.2   0.2   0.6   0.2 0.2     1     3157     2282
## mu[18]        -0.7  -0.3   0.2  -0.3 0.3     1     3387     2878
## mu[19]        -0.7  -0.3   0.2  -0.3 0.3     1     3387     2878
## mu[20]        -0.7  -0.3   0.2  -0.3 0.3     1     3387     2878
## mu[21]        -0.2   0.2   0.6   0.2 0.2     1     3157     2282
## mu[22]        -0.2   0.2   0.6   0.2 0.2     1     3157     2282
## mu[23]        -0.2   0.2   0.6   0.2 0.2     1     3157     2282
## mu[24]        -0.2   0.2   0.6   0.2 0.2     1     3157     2282
## mu[25]        -0.2   0.2   0.6   0.2 0.2     1     3157     2282
## mu[26]        -0.7  -0.3   0.2  -0.3 0.3     1     3387     2878
## mu[27]        -0.7  -0.3   0.2  -0.3 0.3     1     3387     2878
## mu[28]        -0.7  -0.3   0.2  -0.3 0.3     1     3387     2878
## mu[29]        -0.7  -0.3   0.2  -0.3 0.3     1     3387     2878
## mu[30]        -0.7  -0.3   0.2  -0.3 0.3     1     3387     2878
## mu[31]        -0.7  -0.3   0.2  -0.3 0.3     1     3387     2878
```

```
## mu[32]        -0.7  -0.3   0.2  -0.3 0.3     1     3387     2878
## log_lik[1]    -1.4  -1.1  -0.9  -1.1 0.2     1     3173     2838
## log_lik[2]    -1.3  -1.0  -0.8  -1.0 0.1     1     2658     2473
## log_lik[3]    -1.6  -1.2  -0.9  -1.2 0.2     1     3490     2856
## log_lik[4]    -1.5  -1.2  -0.9  -1.2 0.2     1     3667     3143
## log_lik[5]    -1.5  -1.2  -0.9  -1.2 0.2     1     3486     2418
## log_lik[6]    -2.1  -1.6  -1.2  -1.6 0.3     1     3850     2768
## log_lik[7]    -2.4  -1.8  -1.4  -1.8 0.3     1     3501     3135
## log_lik[8]    -1.9  -1.5  -1.1  -1.5 0.2     1     3887     3170
## log_lik[9]    -6.2  -4.3  -3.0  -4.4 1.0     1     3422     2291
## log_lik[10]   -1.2  -1.0  -0.8  -1.0 0.1     1     2792     2787
## log_lik[11]   -1.3  -1.0  -0.8  -1.0 0.1     1     3090     2962
## log_lik[12]   -1.3  -1.1  -0.8  -1.1 0.2     1     3157     2233
## log_lik[13]   -1.3  -1.0  -0.8  -1.0 0.1     1     3008     2260
## log_lik[14]   -1.2  -1.0  -0.8  -1.0 0.1     1     2810     2789
## log_lik[15]   -1.2  -1.0  -0.8  -1.0 0.1     1     2816     2789
## log_lik[16]   -1.2  -1.0  -0.8  -1.0 0.1     1     2879     2520
## log_lik[17]   -1.3  -1.1  -0.8  -1.1 0.2     1     3141     2246
## log_lik[18]   -2.3  -1.6  -1.2  -1.7 0.3     1     3635     2969
## log_lik[19]   -1.6  -1.2  -0.9  -1.2 0.2     1     3432     2773
## log_lik[20]   -2.7  -1.9  -1.4  -2.0 0.4     1     3578     2912
## log_lik[21]   -1.9  -1.5  -1.1  -1.5 0.2     1     3886     3170
## log_lik[22]   -1.6  -1.2  -1.0  -1.2 0.2     1     3558     2409
## log_lik[23]   -1.4  -1.1  -0.9  -1.1 0.2     1     3242     2203
## log_lik[24]   -2.9  -2.1  -1.6  -2.2 0.4     1     3348     2874
## log_lik[25]   -1.5  -1.2  -0.9  -1.2 0.2     1     3465     2369
## log_lik[26]   -1.8  -1.3  -1.0  -1.3 0.2     1     3615     2968
## log_lik[27]   -1.3  -1.0  -0.8  -1.0 0.2     1     2909     2713
## log_lik[28]   -1.3  -1.0  -0.8  -1.0 0.2     1     2730     2418
## log_lik[29]   -3.2  -2.2  -1.6  -2.3 0.5     1     3543     2855
## log_lik[30]   -2.1  -1.5  -1.1  -1.5 0.3     1     3715     3222
## log_lik[31]   -3.0  -2.1  -1.5  -2.2 0.5     1     3559     2856
## log_lik[32]   -1.6  -1.2  -0.9  -1.2 0.2     1     3486     2814
## lp__         -18.7 -15.9 -14.9 -16.3 1.3     1     1561     2167
##
## For each parameter, Bulk_ESS and Tail_ESS are crude measures of
## effective sample size for bulk and tail quantities respectively (an ESS > 100
## per chain is considered good), and Rhat is the potential scale reduction
## factor on rank normalized split chains (at convergence, Rhat <= 1.05).
```

```r
fit_gear = stan(data = list(N=length(scaled_car_properties$qsec),
                 x=c(scaled_car_properties$gear),
                 y=c(scaled_car_properties$qsec)),
                  model_code = stan_separate_model,refresh=0)
monitor(fit_gear, probs = c(0.1, 0.5, 0.9))
```

```
## Inference for the input samples (4 chains: each with iter = 2000; warmup = 0):
##
##                 Q5   Q50   Q95  Mean  SD  Rhat Bulk_ESS Tail_ESS
## alpha         -0.3   0.0   0.3   0.0 0.2     1     3066     2640
## beta          -0.5  -0.2   0.1  -0.2 0.2     1     3438     2479
## sigma          0.8   1.0   1.3   1.0 0.1     1     3040     2454
## mu[1]         -0.4  -0.1   0.2  -0.1 0.2     1     3027     2657
## mu[2]         -0.4  -0.1   0.2  -0.1 0.2     1     3027     2657
## mu[3]         -0.4  -0.1   0.2  -0.1 0.2     1     3027     2657
```

```
## mu[4]         -0.2   0.2   0.6   0.2 0.3     1     3391      2443
## mu[5]         -0.2   0.2   0.6   0.2 0.3     1     3391      2443
## mu[6]         -0.2   0.2   0.6   0.2 0.3     1     3391      2443
## mu[7]         -0.2   0.2   0.6   0.2 0.3     1     3391      2443
## mu[8]         -0.4  -0.1   0.2  -0.1 0.2     1     3027      2657
## mu[9]         -0.4  -0.1   0.2  -0.1 0.2     1     3027      2657
## mu[10]        -0.4  -0.1   0.2  -0.1 0.2     1     3027      2657
## mu[11]        -0.4  -0.1   0.2  -0.1 0.2     1     3027      2657
## mu[12]        -0.2   0.2   0.6   0.2 0.3     1     3391      2443
## mu[13]        -0.2   0.2   0.6   0.2 0.3     1     3391      2443
## mu[14]        -0.2   0.2   0.6   0.2 0.3     1     3391      2443
## mu[15]        -0.2   0.2   0.6   0.2 0.3     1     3391      2443
## mu[16]        -0.2   0.2   0.6   0.2 0.3     1     3391      2443
## mu[17]        -0.2   0.2   0.6   0.2 0.3     1     3391      2443
## mu[18]        -0.4  -0.1   0.2  -0.1 0.2     1     3027      2657
## mu[19]        -0.4  -0.1   0.2  -0.1 0.2     1     3027      2657
## mu[20]        -0.4  -0.1   0.2  -0.1 0.2     1     3027      2657
## mu[21]        -0.2   0.2   0.6   0.2 0.3     1     3391      2443
## mu[22]        -0.2   0.2   0.6   0.2 0.3     1     3391      2443
## mu[23]        -0.2   0.2   0.6   0.2 0.3     1     3391      2443
## mu[24]        -0.2   0.2   0.6   0.2 0.3     1     3391      2443
## mu[25]        -0.2   0.2   0.6   0.2 0.3     1     3391      2443
## mu[26]        -0.4  -0.1   0.2  -0.1 0.2     1     3027      2657
## mu[27]        -1.0  -0.4   0.3  -0.4 0.4     1     3219      2618
## mu[28]        -1.0  -0.4   0.3  -0.4 0.4     1     3219      2618
## mu[29]        -1.0  -0.4   0.3  -0.4 0.4     1     3219      2618
## mu[30]        -1.0  -0.4   0.3  -0.4 0.4     1     3219      2618
## mu[31]        -1.0  -0.4   0.3  -0.4 0.4     1     3219      2618
## mu[32]        -0.4  -0.1   0.2  -0.1 0.2     1     3027      2657
## log_lik[1]    -1.5  -1.2  -1.0  -1.2 0.2     1     3270      2482
## log_lik[2]    -1.3  -1.0  -0.8  -1.0 0.1     1     2934      2453
## log_lik[3]    -1.3  -1.1  -0.9  -1.1 0.1     1     3130      2438
## log_lik[4]    -1.6  -1.2  -0.9  -1.2 0.2     1     3539      2844
## log_lik[5]    -1.5  -1.2  -0.9  -1.2 0.2     1     3205      2307
## log_lik[6]    -2.1  -1.6  -1.2  -1.6 0.3     1     3784      3328
## log_lik[7]    -2.4  -1.8  -1.3  -1.8 0.3     1     3668      3048
## log_lik[8]    -2.3  -1.8  -1.4  -1.8 0.3     1     3451      2773
## log_lik[9]    -7.0  -5.0  -3.5  -5.1 1.1     1     3073      2371
## log_lik[10]   -1.3  -1.0  -0.8  -1.0 0.1     1     2980      2695
## log_lik[11]   -1.5  -1.2  -1.0  -1.2 0.2     1     3311      2661
## log_lik[12]   -1.4  -1.1  -0.8  -1.1 0.2     1     2659      2182
## log_lik[13]   -1.3  -1.0  -0.8  -1.0 0.2     1     2417      2177
## log_lik[14]   -1.2  -1.0  -0.8  -1.0 0.1     1     2281      2131
## log_lik[15]   -1.2  -1.0  -0.8  -1.0 0.1     1     2274      2106
## log_lik[16]   -1.2  -1.0  -0.8  -1.0 0.1     1     2259      2086
## log_lik[17]   -1.3  -1.1  -0.8  -1.1 0.2     1     2623      2233
## log_lik[18]   -1.8  -1.4  -1.2  -1.5 0.2     1     3467      2912
## log_lik[19]   -1.3  -1.1  -0.9  -1.1 0.1     1     3080      2490
## log_lik[20]   -2.2  -1.7  -1.4  -1.7 0.3     1     3480      2772
## log_lik[21]   -1.9  -1.5  -1.1  -1.5 0.2     1     3775      3342
## log_lik[22]   -1.6  -1.2  -1.0  -1.2 0.2     1     3532      2381
## log_lik[23]   -1.4  -1.1  -0.8  -1.1 0.2     1     2784      2195
## log_lik[24]   -2.9  -2.1  -1.6  -2.2 0.4     1     3574      2579
## log_lik[25]   -1.5  -1.2  -0.9  -1.2 0.2     1     3156      2289
```

```
## log_lik[26]  -1.5  -1.2  -1.0  -1.2 0.2     1     3311     2661
## log_lik[27]  -1.4  -1.0  -0.8  -1.1 0.2     1     2676     2611
## log_lik[28]  -1.3  -1.0  -0.8  -1.0 0.2     1     2477     2599
## log_lik[29]  -3.2  -2.0  -1.3  -2.1 0.6     1     3523     2651
## log_lik[30]  -2.1  -1.4  -1.0  -1.4 0.4     1     3624     3189
## log_lik[31]  -3.1  -1.9  -1.3  -2.0 0.6     1     3535     2749
## log_lik[32]  -1.3  -1.1  -0.9  -1.1 0.1     1     3124     2438
## lp__        -19.0 -16.1 -15.0 -16.4 1.3     1     1597     2257
##
## For each parameter, Bulk_ESS and Tail_ESS are crude measures of
## effective sample size for bulk and tail quantities respectively (an ESS > 100
## per chain is considered good), and Rhat is the potential scale reduction
## factor on rank normalized split chains (at convergence, Rhat <= 1.05).
```

```r
fit_carb = stan(data = list(N=length(scaled_car_properties$qsec),
                   x=c(scaled_car_properties$carb),
                   y=c(scaled_car_properties$qsec)),
                    model_code = stan_separate_model,refresh=0)
monitor(fit_carb, probs = c(0.1, 0.5, 0.9))
```

```
## Inference for the input samples (4 chains: each with iter = 2000; warmup = 0):
##
##                Q5  Q50  Q95 Mean  SD  Rhat Bulk_ESS Tail_ESS
## alpha        -0.2  0.0  0.2  0.0 0.1     1     3212     2443
## beta         -0.9 -0.7 -0.4 -0.7 0.1     1     3337     2604
## sigma         0.6  0.8  1.0  0.8 0.1     1     3077     2577
## mu[1]        -0.8 -0.5 -0.2 -0.5 0.2     1     3285     2792
## mu[2]        -0.8 -0.5 -0.2 -0.5 0.2     1     3285     2792
## mu[3]         0.4  0.7  1.1  0.7 0.2     1     3247     2576
## mu[4]         0.4  0.7  1.1  0.7 0.2     1     3247     2576
## mu[5]         0.1  0.3  0.6  0.3 0.2     1     3212     2648
## mu[6]         0.4  0.7  1.1  0.7 0.2     1     3247     2576
## mu[7]        -0.8 -0.5 -0.2 -0.5 0.2     1     3285     2792
## mu[8]         0.1  0.3  0.6  0.3 0.2     1     3212     2648
## mu[9]         0.1  0.3  0.6  0.3 0.2     1     3212     2648
## mu[10]       -0.8 -0.5 -0.2 -0.5 0.2     1     3285     2792
## mu[11]       -0.8 -0.5 -0.2 -0.5 0.2     1     3285     2792
## mu[12]       -0.3 -0.1  0.2 -0.1 0.2     1     3222     2556
## mu[13]       -0.3 -0.1  0.2 -0.1 0.2     1     3222     2556
## mu[14]       -0.3 -0.1  0.2 -0.1 0.2     1     3222     2556
## mu[15]       -0.8 -0.5 -0.2 -0.5 0.2     1     3285     2792
## mu[16]       -0.8 -0.5 -0.2 -0.5 0.2     1     3285     2792
## mu[17]       -0.8 -0.5 -0.2 -0.5 0.2     1     3285     2792
## mu[18]        0.4  0.7  1.1  0.7 0.2     1     3247     2576
## mu[19]        0.1  0.3  0.6  0.3 0.2     1     3212     2648
## mu[20]        0.4  0.7  1.1  0.7 0.2     1     3247     2576
## mu[21]        0.4  0.7  1.1  0.7 0.2     1     3247     2576
## mu[22]        0.1  0.3  0.6  0.3 0.2     1     3212     2648
## mu[23]        0.1  0.3  0.6  0.3 0.2     1     3212     2648
## mu[24]       -0.8 -0.5 -0.2 -0.5 0.2     1     3285     2792
## mu[25]        0.1  0.3  0.6  0.3 0.2     1     3212     2648
## mu[26]        0.4  0.7  1.1  0.7 0.2     1     3247     2576
## mu[27]        0.1  0.3  0.6  0.3 0.2     1     3212     2648
## mu[28]        0.1  0.3  0.6  0.3 0.2     1     3212     2648
## mu[29]       -0.8 -0.5 -0.2 -0.5 0.2     1     3285     2792
```

```
## mu[30]       -1.8 -1.3 -0.8 -1.3 0.3     1     3341     2840
## mu[31]       -2.9 -2.1 -1.3 -2.1 0.5     1     3354     2781
## mu[32]        0.1  0.3  0.6  0.3 0.2     1     3212     2648
## log_lik[1]   -1.0 -0.8 -0.6 -0.8 0.2     1     2855     2436
## log_lik[2]   -1.0 -0.7 -0.5 -0.7 0.1     1     2670     2700
## log_lik[3]   -1.1 -0.8 -0.6 -0.8 0.2     1     2927     2680
## log_lik[4]   -1.0 -0.7 -0.5 -0.7 0.2     1     2805     2255
## log_lik[5]   -1.6 -1.2 -0.9 -1.2 0.2     1     3689     3229
## log_lik[6]   -1.4 -1.0 -0.7 -1.0 0.2     1     3757     2594
## log_lik[7]   -1.4 -1.0 -0.8 -1.1 0.2     1     3519     3130
## log_lik[8]   -1.8 -1.3 -1.0 -1.3 0.2     1     3630     3390
## log_lik[9]   -8.3 -5.7 -3.8 -5.8 1.4     1     3023     2187
## log_lik[10]  -1.5 -1.1 -0.8 -1.2 0.2     1     3651     3101
## log_lik[11]  -2.3 -1.6 -1.2 -1.7 0.3     1     3550     2978
## log_lik[12]  -1.0 -0.7 -0.5 -0.7 0.1     1     2777     2222
## log_lik[13]  -0.9 -0.7 -0.5 -0.7 0.1     1     2768     2358
## log_lik[14]  -1.0 -0.7 -0.5 -0.7 0.1     1     2896     2426
## log_lik[15]  -1.3 -1.0 -0.7 -1.0 0.2     1     3471     3102
## log_lik[16]  -1.2 -0.9 -0.6 -0.9 0.2     1     3340     2788
## log_lik[17]  -1.0 -0.8 -0.5 -0.8 0.1     1     2939     2928
## log_lik[18]  -1.0 -0.7 -0.5 -0.7 0.2     1     2843     2353
## log_lik[19]  -0.9 -0.7 -0.5 -0.7 0.1     1     2795     2255
## log_lik[20]  -1.2 -0.8 -0.6 -0.9 0.2     1     3428     2428
## log_lik[21]  -1.3 -0.9 -0.6 -0.9 0.2     1     3577     2437
## log_lik[22]  -1.8 -1.3 -1.0 -1.3 0.2     1     3709     3204
## log_lik[23]  -1.3 -1.0 -0.8 -1.0 0.2     1     3487     2936
## log_lik[24]  -1.8 -1.3 -1.0 -1.4 0.3     1     3679     3417
## log_lik[25]  -1.6 -1.2 -0.9 -1.2 0.2     1     3670     3285
## log_lik[26]  -1.0 -0.7 -0.5 -0.7 0.1     1     2650     2200
## log_lik[27]  -2.0 -1.5 -1.1 -1.5 0.3     1     3695     3089
## log_lik[28]  -1.7 -1.3 -1.0 -1.3 0.2     1     3707     3125
## log_lik[29]  -3.2 -2.2 -1.6 -2.3 0.5     1     3481     2648
## log_lik[30]  -1.1 -0.8 -0.5 -0.8 0.2     1     2322     2585
## log_lik[31]  -1.6 -0.8 -0.6 -0.9 0.4     1     2845     2998
## log_lik[32]  -1.0 -0.7 -0.5 -0.7 0.1     1     2835     2339
## lp__        -11.0 -8.1 -7.0 -8.4 1.3     1     1947     2398
##
## For each parameter, Bulk_ESS and Tail_ESS are crude measures of
## effective sample size for bulk and tail quantities respectively (an ESS > 100
## per chain is considered good), and Rhat is the potential scale reduction
## factor on rank normalized split chains (at convergence, Rhat <= 1.05).
```

Multivariate linear modeling

```
stan_nlin_model1 = '
data {
  int<lower=0> n;
  vector[n] hp;
  vector[n] wt;
  vector[n] vs;
  vector[n] am;
  vector[n] carb;
  vector[n] qsec;
}
```

```
parameters {
  real alpha;
  real beta_hp;
  real beta_wt;
  real beta_vs;
  real beta_am;
  real beta_carb;
  real<lower=0> sigma;
}
transformed parameters{
  vector[n] mu;
  mu = alpha + beta_hp*hp + beta_wt*wt + beta_vs*vs +
       beta_am*am + beta_carb*carb;
}
model {
  alpha ~ cauchy(0,10);
  beta_wt ~ student_t(3,0,2);
  beta_hp ~ student_t(3,0,2);
  beta_am ~ student_t(3,0,2);
  beta_vs ~ student_t(3,0,2);
  beta_carb ~ student_t(3,0,2);
  sigma ~ normal(0, 10);
  qsec ~ normal(mu, sigma);
}
// Log likelihoods genereated for LOO
generated quantities {
  vector[n] log_lik;
  for (i in 1:n)
    log_lik[i] = normal_lpdf(qsec[i] |mu[i] , sigma);
}
'

fit_nlin1 = stan(data = list(n=length(scaled_car_properties$hp),
                 hp=c(scaled_car_properties$hp),
                 wt=c(scaled_car_properties$wt),
                 vs=c(scaled_car_properties$vs),
                 am=c(scaled_car_properties$am),
                 carb=c(scaled_car_properties$carb),
                 qsec=c(scaled_car_properties$qsec)),
                  model_code = stan_nlin_model1,refresh=0)

monitor(fit_nlin1, probs = c(0.1, 0.5, 0.9))
```

```
## Inference for the input samples (4 chains: each with iter = 2000; warmup = 0):
##
##                Q5  Q50  Q95 Mean  SD  Rhat Bulk_ESS Tail_ESS
## alpha        -0.1  0.0  0.1  0.0 0.1     1     4583     2866
## beta_hp      -0.8 -0.6 -0.3 -0.6 0.2     1     3246     2251
## beta_wt       0.2  0.4  0.7  0.4 0.2     1     2826     2473
## beta_vs       0.3  0.6  0.8  0.5 0.1     1     4016     2700
## beta_am      -0.4 -0.2  0.1 -0.2 0.1     1     2883     2459
## beta_carb    -0.3 -0.1  0.1 -0.1 0.1     1     2926     2577
## sigma         0.4  0.5  0.6  0.5 0.1     1     2660     2195
## mu[1]        -1.1 -0.7 -0.3 -0.7 0.2     1     3676     2653
```

```
## mu[2]        -1.0 -0.6 -0.2 -0.6 0.2      1     3635     2685
## mu[3]         0.3  0.6  0.9  0.6 0.2      1     3253     2975
## mu[4]         0.9  1.1  1.4  1.1 0.2      1     4259     3388
## mu[5]        -0.7 -0.4 -0.2 -0.4 0.2      1     3719     2804
## mu[6]         1.0  1.3  1.6  1.3 0.2      1     4058     3408
## mu[7]        -1.4 -1.1 -0.7 -1.1 0.2      1     3435     2680
## mu[8]         1.1  1.5  1.8  1.5 0.2      1     4060     3265
## mu[9]         0.9  1.2  1.4  1.2 0.2      1     4456     3287
## mu[10]        0.6  1.0  1.3  1.0 0.2      1     3810     2690
## mu[11]        0.6  1.0  1.3  1.0 0.2      1     3810     2690
## mu[12]       -0.5 -0.3  0.0 -0.3 0.1      1     4309     3393
## mu[13]       -0.6 -0.4 -0.2 -0.4 0.1      1     3924     3502
## mu[14]       -0.6 -0.4 -0.2 -0.4 0.1      1     4021     3506
## mu[15]       -0.4  0.0  0.4  0.0 0.2      1     3500     3267
## mu[16]       -0.4  0.0  0.4  0.0 0.3      1     3389     2828
## mu[17]       -0.6 -0.2  0.2 -0.2 0.2      1     3415     2787
## mu[18]        0.5  0.7  1.1  0.8 0.2      1     3530     2823
## mu[19]        0.2  0.5  0.8  0.5 0.2      1     4593     3013
## mu[20]        0.3  0.6  0.9  0.6 0.2      1     4430     3027
## mu[21]        0.6  0.9  1.3  0.9 0.2      1     3969     3111
## mu[22]       -0.5 -0.2  0.1 -0.2 0.2      1     4084     3076
## mu[23]       -0.5 -0.2  0.0 -0.2 0.2      1     3970     2983
## mu[24]       -1.3 -1.0 -0.7 -1.0 0.2      1     3673     3104
## mu[25]       -0.5 -0.3  0.0 -0.3 0.2      1     3945     3041
## mu[26]        0.3  0.6  0.9  0.6 0.2      1     4246     2852
## mu[27]       -1.0 -0.6 -0.2 -0.6 0.2      1     4076     2974
## mu[28]       -0.3  0.0  0.3  0.0 0.2      1     4386     3036
## mu[29]       -2.1 -1.7 -1.3 -1.7 0.3      1     3720     2979
## mu[30]       -1.7 -1.3 -0.9 -1.3 0.2      1     3698     3112
## mu[31]       -2.9 -2.4 -1.8 -2.4 0.3      1     4602     3660
## mu[32]        0.3  0.6  0.9  0.6 0.2      1     3199     2543
## log_lik[1]   -0.7 -0.2  0.0 -0.3 0.2      1     2186     2585
## log_lik[2]   -0.8 -0.3  0.0 -0.3 0.3      1     2298     2808
## log_lik[3]   -0.7 -0.3  0.0 -0.3 0.2      1     2539     2554
## log_lik[4]   -0.8 -0.3  0.0 -0.4 0.2      1     3582     2985
## log_lik[5]   -0.5 -0.2  0.0 -0.2 0.2      1     2169     2381
## log_lik[6]   -0.5 -0.2  0.0 -0.2 0.2      1     2206     2460
## log_lik[7]   -0.6 -0.2  0.0 -0.3 0.2      1     2170     2508
## log_lik[8]   -0.9 -0.3  0.0 -0.4 0.3      1     3567     3516
## log_lik[9]  -10.6 -6.6 -3.7 -6.8 2.1      1     3371     2797
## log_lik[10]  -3.0 -1.3 -0.4 -1.5 0.8      1     3829     2970
## log_lik[11]  -1.4 -0.5 -0.1 -0.6 0.4      1     3707     3105
## log_lik[12]  -0.5 -0.2  0.0 -0.2 0.2      1     2474     2539
## log_lik[13]  -0.7 -0.3 -0.1 -0.4 0.2      1     3437     3445
## log_lik[14]  -1.3 -0.7 -0.3 -0.7 0.3      1     4050     3152
## log_lik[15]  -0.7 -0.2  0.0 -0.3 0.2      1     2329     2793
## log_lik[16]  -0.8 -0.3  0.0 -0.3 0.3      1     2175     2553
## log_lik[17]  -0.7 -0.3  0.0 -0.3 0.3      1     2240     2491
## log_lik[18]  -0.7 -0.3  0.0 -0.3 0.2      1     2416     3080
## log_lik[19]  -0.7 -0.3  0.0 -0.3 0.2      1     3022     3294
## log_lik[20]  -1.8 -0.9 -0.3 -1.0 0.5      1     4377     3484
## log_lik[21]  -1.0 -0.4 -0.1 -0.5 0.3      1     3383     3292
## log_lik[22]  -1.0 -0.5 -0.1 -0.5 0.3      1     3885     2968
## log_lik[23]  -0.5 -0.2  0.0 -0.2 0.2      1     2340     2714
```

```
## log_lik[24]  -1.3 -0.6 -0.2 -0.6 0.4     1     3794     3492
## log_lik[25]  -0.7 -0.3  0.0 -0.3 0.2     1     2423     2703
## log_lik[26]  -0.5 -0.2  0.0 -0.2 0.2     1     2173     2628
## log_lik[27]  -0.7 -0.2  0.0 -0.3 0.2     1     2070     2867
## log_lik[28]  -1.9 -0.8 -0.3 -0.9 0.5     1     4420     3514
## log_lik[29]  -1.0 -0.3  0.0 -0.4 0.3     1     2414     3020
## log_lik[30]  -0.7 -0.3  0.0 -0.3 0.2     1     2185     2766
## log_lik[31]  -3.0 -0.9 -0.2 -1.1 0.9     1     4485     3504
## log_lik[32]  -0.8 -0.3  0.0 -0.3 0.2     1     2487     2711
## lp__          3.9  8.3 10.7  8.0 2.1     1     1363     1897
##
## For each parameter, Bulk_ESS and Tail_ESS are crude measures of
## effective sample size for bulk and tail quantities respectively (an ESS > 100
## per chain is considered good), and Rhat is the potential scale reduction
## factor on rank normalized split chains (at convergence, Rhat <= 1.05).
```

```
stan_nlin_model2 = '
data {
  int<lower=0> n;
  vector[n] disp;
  vector[n] wt;
  vector[n] vs;
  vector[n] carb;
  vector[n] qsec;
}
parameters {
  real alpha;
  real beta_disp;
  real beta_wt;
  real beta_vs;
  real beta_carb;
  real<lower=0> sigma;
}
transformed parameters{
  vector[n] mu;
  mu = alpha + beta_disp*disp + beta_wt*wt + beta_vs*vs + beta_carb*carb;
}
model {
  alpha ~ cauchy(0,10);
  beta_disp ~ student_t(3,0,2);
  beta_wt ~ student_t(3,0,2);
  beta_vs ~ student_t(3,0,2);
  beta_carb ~ student_t(3,0,2);
  sigma ~ normal(0, 10);
  qsec ~ normal(mu, sigma);
}
// Log likelihoods genereated for LOO
generated quantities {
  vector[n] log_lik;
  for (i in 1:n)
    log_lik[i] = normal_lpdf(qsec[i] |mu[i] , sigma);
}
'
```

```r
fit_nlin2 = stan(data = list(n=length(scaled_car_properties$hp),
                 disp=c(scaled_car_properties$disp),
                 wt=c(scaled_car_properties$wt),
                 vs=c(scaled_car_properties$vs),
                 carb=c(scaled_car_properties$carb),
                 qsec=c(scaled_car_properties$qsec)),
                  model_code = stan_nlin_model2,refresh=0)

monitor(fit_nlin2, probs = c(0.1, 0.5, 0.9))
```

```
## Inference for the input samples (4 chains: each with iter = 2000; warmup = 0):
##
##                 Q5  Q50  Q95 Mean  SD  Rhat Bulk_ESS Tail_ESS
## alpha         -0.1  0.0  0.1  0.0 0.1     1     3679     2359
## beta_disp     -1.2 -0.8 -0.5 -0.8 0.2     1     1965     2093
## beta_wt        0.7  1.0  1.3  1.0 0.2     1     2132     2447
## beta_vs        0.2  0.4  0.6  0.4 0.1     1     2218     2488
## beta_carb     -0.7 -0.5 -0.3 -0.5 0.1     1     2648     2338
## sigma          0.4  0.4  0.6  0.4 0.1     1     2928     2543
## mu[1]         -1.2 -0.9 -0.6 -0.9 0.2     1     3752     2472
## mu[2]         -0.9 -0.6 -0.3 -0.6 0.2     1     2668     2086
## mu[3]          0.7  0.9  1.2  0.9 0.1     1     4569     3326
## mu[4]          0.6  0.9  1.2  0.9 0.2     1     3524     2678
## mu[5]         -1.0 -0.7 -0.4 -0.7 0.2     1     3660     2791
## mu[6]          1.1  1.3  1.6  1.3 0.2     1     5546     3013
## mu[7]         -1.5 -1.3 -1.0 -1.2 0.2     1     2661     2441
## mu[8]          1.0  1.3  1.5  1.3 0.1     1     4041     2861
## mu[9]          1.0  1.3  1.5  1.3 0.1     1     3945     2796
## mu[10]         0.4  0.7  1.1  0.7 0.2     1     3750     2548
## mu[11]         0.4  0.7  1.1  0.7 0.2     1     3750     2548
## mu[12]        -0.1  0.2  0.5  0.2 0.2     1     1941     2221
## mu[13]        -0.4 -0.2  0.0 -0.2 0.1     1     2182     2494
## mu[14]        -0.4 -0.1  0.1 -0.1 0.1     1     2114     2401
## mu[15]        -0.6 -0.3  0.1 -0.3 0.2     1     4624     3270
## mu[16]        -0.3  0.0  0.3  0.0 0.2     1     4335     3198
## mu[17]        -0.3  0.1  0.4  0.1 0.2     1     3947     3357
## mu[18]         0.8  1.0  1.2  1.0 0.1     1     4137     3119
## mu[19]        -0.2  0.1  0.4  0.1 0.2     1     2862     2910
## mu[20]         0.4  0.7  0.9  0.7 0.1     1     4379     3284
## mu[21]         0.8  1.0  1.2  1.0 0.1     1     4637     3385
## mu[22]        -0.6 -0.4 -0.1 -0.4 0.1     1     3529     2815
## mu[23]        -0.6 -0.4 -0.1 -0.4 0.1     1     3335     2621
## mu[24]        -1.1 -0.9 -0.7 -0.9 0.1     1     3554     2668
## mu[25]        -0.9 -0.6 -0.3 -0.6 0.2     1     3807     2973
## mu[26]         0.5  0.7  1.0  0.7 0.1     1     4419     3346
## mu[27]        -0.9 -0.5 -0.1 -0.5 0.2     1     2754     2607
## mu[28]        -0.5 -0.1  0.2 -0.1 0.2     1     2418     2633
## mu[29]        -1.9 -1.6 -1.2 -1.6 0.2     1     2392     2166
## mu[30]        -1.7 -1.3 -0.9 -1.3 0.2     1     4982     2292
## mu[31]        -2.7 -2.2 -1.6 -2.2 0.3     1     2927     2743
## mu[32]         0.8  1.0  1.2  1.0 0.1     1     4519     3177
## log_lik[1]    -0.6 -0.2  0.1 -0.2 0.2     1     2584     2786
## log_lik[2]    -0.7 -0.2  0.0 -0.3 0.2     1     2829     2904
## log_lik[3]    -1.5 -0.8 -0.4 -0.9 0.4     1     4614     3618
```

```
## log_lik[4]    -0.5 -0.2  0.1 -0.2 0.2     1     2336     2575
## log_lik[5]    -0.9 -0.3  0.0 -0.4 0.3     1     3370     2998
## log_lik[6]    -0.4 -0.1  0.1 -0.2 0.2     1     2257     2500
## log_lik[7]    -0.6 -0.2  0.1 -0.2 0.2     1     2537     2749
## log_lik[8]    -0.5 -0.2  0.1 -0.2 0.2     1     2370     2403
## log_lik[9]   -10.4 -6.5 -3.8 -6.7 2.0     1     2920     2367
## log_lik[10]   -1.7 -0.7 -0.2 -0.8 0.5     1     4253     2889
## log_lik[11]   -0.7 -0.2  0.1 -0.2 0.2     1     2662     2272
## log_lik[12]   -1.4 -0.6 -0.1 -0.6 0.4     1     2130     2894
## log_lik[13]   -0.4 -0.1  0.1 -0.2 0.2     1     2480     2176
## log_lik[14]   -0.7 -0.3  0.0 -0.3 0.2     1     2645     2486
## log_lik[15]   -1.2 -0.4  0.0 -0.5 0.4     1     4400     3876
## log_lik[16]   -0.6 -0.2  0.1 -0.2 0.2     1     2118     2679
## log_lik[17]   -1.1 -0.4  0.0 -0.4 0.3     1     3053     3284
## log_lik[18]   -0.5 -0.2  0.1 -0.2 0.2     1     2721     2883
## log_lik[19]   -0.9 -0.3  0.0 -0.4 0.3     1     3125     3255
## log_lik[20]   -1.4 -0.7 -0.2 -0.7 0.4     1     4443     3627
## log_lik[21]   -0.6 -0.2  0.0 -0.2 0.2     1     3344     3195
## log_lik[22]   -0.6 -0.2  0.0 -0.2 0.2     1     2764     2803
## log_lik[23]   -0.4 -0.1  0.1 -0.2 0.2     1     2404     2585
## log_lik[24]   -1.3 -0.7 -0.3 -0.7 0.3     1     3861     3146
## log_lik[25]   -0.6 -0.2  0.1 -0.2 0.2     1     2650     2961
## log_lik[26]   -0.5 -0.2  0.1 -0.2 0.2     1     3350     3162
## log_lik[27]   -0.8 -0.2  0.0 -0.3 0.3     1     2752     2765
## log_lik[28]   -1.5 -0.5 -0.1 -0.6 0.4     1     2967     3547
## log_lik[29]   -1.1 -0.3  0.0 -0.4 0.3     1     2717     2588
## log_lik[30]   -0.7 -0.2  0.1 -0.2 0.2     1     1949     2203
## log_lik[31]   -1.9 -0.4  0.0 -0.6 0.6     1     3238     3475
## log_lik[32]   -1.9 -1.0 -0.5 -1.1 0.4     1     4522     3324
## lp__           5.8  9.9 11.9  9.5 2.0     1     1646     2105
##
## For each parameter, Bulk_ESS and Tail_ESS are crude measures of
## effective sample size for bulk and tail quantities respectively (an ESS > 100
## per chain is considered good), and Rhat is the potential scale reduction
## factor on rank normalized split chains (at convergence, Rhat <= 1.05).
```

```
stan_nlin_model3 = '
data {
  int<lower=0> n;
  vector[n] lphkm;
  vector[n] cyl;
  vector[n] disp;
  vector[n] hp;
  vector[n] drat;
  vector[n] wt;
  vector[n] vs;
  vector[n] am;
  vector[n] gear;
  vector[n] carb;
  vector[n] qsec;
}
parameters {
  real alpha;
  real beta_lphkm;
```

```
  real beta_cyl;
  real beta_disp;
  real beta_hp;
  real beta_drat;
  real beta_wt;
  real beta_vs;
  real beta_am;
  real beta_gear;
  real beta_carb;
  real<lower=0> sigma;
}
transformed parameters{
  vector[n] mu;
  mu = alpha + beta_lphkm*lphkm + beta_cyl* cyl + beta_disp * disp +
  beta_hp*hp + beta_drat* drat + beta_wt*wt + beta_vs*vs +
  beta_am*am + beta_gear*gear + beta_carb*carb;
}
model {
  alpha ~ cauchy(0,10);
  beta_lphkm ~ student_t(3,0,2);
  beta_cyl ~ student_t(3,0,2);
  beta_hp ~ student_t(3,0,2);
  beta_drat ~ student_t(3,0,2);
  beta_wt ~ student_t(3,0,2);
  beta_am ~ student_t(3,0,2);
  beta_vs ~ student_t(3,0,2);
  beta_gear ~ student_t(3,0,2);
  beta_carb ~ student_t(3,0,2);
  sigma ~ normal(0, 10);
  qsec ~ normal(mu, sigma);
}
// Log likelihoods genereated for LOO
generated quantities {
  vector[n] log_lik;
  for (i in 1:n)
    log_lik[i] = normal_lpdf(qsec[i] |mu[i] , sigma);
}
'

fit_nlin3 = stan(data = list(n=length(scaled_car_properties$hp),
                             lphkm=c(scaled_car_properties$lphkm),
                             cyl=c(scaled_car_properties$cyl),
                             disp=c(scaled_car_properties$disp),
                             hp=c(scaled_car_properties$hp),
                             drat=c(scaled_car_properties$drat),
                             wt=c(scaled_car_properties$wt),
                             vs=c(scaled_car_properties$vs),
                             am=c(scaled_car_properties$am),
                             gear=c(scaled_car_properties$gear),
                             carb=c(scaled_car_properties$carb),
                             qsec=c(scaled_car_properties$qsec)),
                 model_code = stan_nlin_model3,refresh=0)
```

```r
monitor(fit_nlin3, probs = c(0.1, 0.5, 0.9))
```

```
## Inference for the input samples (4 chains: each with iter = 2000; warmup = 0):
##
##                Q5  Q50  Q95 Mean  SD  Rhat Bulk_ESS Tail_ESS
## alpha        -0.1  0.0  0.1  0.0 0.1     1     3874     2296
## beta_lphkm   -0.5 -0.1  0.3 -0.1 0.2     1     3555     2854
## beta_cyl     -0.9 -0.4  0.1 -0.4 0.3     1     2361     2409
## beta_disp    -1.0 -0.4  0.2 -0.4 0.4     1     1834     2337
## beta_hp      -0.5 -0.1  0.3 -0.1 0.3     1     2493     2802
## beta_drat    -0.3  0.0  0.2  0.0 0.2     1     3602     2008
## beta_wt       0.3  0.7  1.2  0.7 0.3     1     2063     2167
## beta_vs       0.0  0.3  0.6  0.3 0.2     1     2733     2521
## beta_am      -0.5 -0.2  0.1 -0.2 0.2     1     3093     2472
## beta_gear    -0.4 -0.1  0.2 -0.1 0.2     1     2946     2985
## beta_carb    -0.6 -0.2  0.1 -0.2 0.2     1     1849     2553
## sigma         0.4  0.5  0.6  0.5 0.1     1     2442     2440
## mu[1]        -1.2 -0.8 -0.4 -0.8 0.3     1     3563     2791
## mu[2]        -1.0 -0.6 -0.2 -0.6 0.3     1     4639     2993
## mu[3]         0.3  0.7  1.1  0.7 0.2     1     3529     2953
## mu[4]         0.5  1.0  1.3  0.9 0.2     1     3209     2638
## mu[5]        -1.0 -0.6 -0.2 -0.6 0.2     1     4383     3324
## mu[6]         0.8  1.2  1.6  1.2 0.3     1     4228     3158
## mu[7]        -1.5 -1.0 -0.6 -1.0 0.3     1     3124     2938
## mu[8]         1.1  1.6  2.0  1.6 0.3     1     4163     2841
## mu[9]         1.1  1.5  1.9  1.5 0.3     1     3733     3623
## mu[10]        0.3  0.8  1.2  0.8 0.3     1     3640     3157
## mu[11]        0.2  0.7  1.2  0.7 0.3     1     3477     3104
## mu[12]       -0.4  0.0  0.4  0.0 0.3     1     2321     2714
## mu[13]       -0.6 -0.3  0.1 -0.3 0.2     1     3433     2879
## mu[14]       -0.6 -0.3  0.1 -0.3 0.2     1     2955     2766
## mu[15]       -0.8 -0.2  0.3 -0.2 0.3     1     3715     3114
## mu[16]       -0.5 -0.1  0.4 -0.1 0.3     1     4737     2959
## mu[17]       -0.5  0.1  0.7  0.1 0.3     1     3500     3183
## mu[18]        0.5  0.8  1.2  0.8 0.2     1     2996     2986
## mu[19]       -0.3  0.2  0.8  0.2 0.3     1     3037     2515
## mu[20]        0.3  0.6  0.9  0.6 0.2     1     3493     3429
## mu[21]        0.8  1.3  1.8  1.3 0.3     1     3461     2877
## mu[22]       -0.8 -0.4  0.0 -0.4 0.2     1     4069     3402
## mu[23]       -0.8 -0.4 -0.1 -0.4 0.2     1     3531     3303
## mu[24]       -1.3 -0.8 -0.4 -0.8 0.3     1     4183     2738
## mu[25]       -0.9 -0.4  0.1 -0.4 0.3     1     4071     3110
## mu[26]        0.3  0.6  0.9  0.6 0.2     1     3456     3282
## mu[27]       -0.9 -0.3  0.2 -0.3 0.4     1     3272     2990
## mu[28]       -0.6 -0.1  0.4 -0.1 0.3     1     3324     2848
## mu[29]       -2.7 -2.0 -1.4 -2.0 0.4     1     3494     3115
## mu[30]       -1.7 -1.2 -0.7 -1.2 0.3     1     4497     2959
## mu[31]       -2.8 -2.2 -1.6 -2.2 0.4     1     4353     3263
## mu[32]        0.4  0.8  1.2  0.8 0.2     1     3350     3360
## log_lik[1]   -0.8 -0.2  0.0 -0.3 0.3     1     2032     2667
## log_lik[2]   -0.9 -0.3  0.0 -0.3 0.3     1     2442     3004
## log_lik[3]   -1.1 -0.4  0.0 -0.4 0.4     1     3313     3090
## log_lik[4]   -0.7 -0.2  0.0 -0.3 0.3     1     2025     2738
## log_lik[5]   -0.8 -0.3  0.0 -0.3 0.3     1     2499     3179
```

```
## log_lik[6]  -0.8 -0.3  0.0 -0.3 0.3     1     2409     3046
## log_lik[7]  -0.9 -0.3  0.0 -0.3 0.3     1     2448     2905
## log_lik[8]  -1.7 -0.5 -0.1 -0.7 0.5     1     4126     3524
## log_lik[9]  -8.5 -4.5 -1.9 -4.8 2.1     1     3365     2896
## log_lik[10] -2.3 -0.8 -0.1 -1.0 0.7     1     3754     3253
## log_lik[11] -1.0 -0.3  0.0 -0.4 0.4     1     2329     3035
## log_lik[12] -1.1 -0.3  0.0 -0.4 0.4     1     2620     3187
## log_lik[13] -0.7 -0.2  0.0 -0.3 0.2     1     2376     2880
## log_lik[14] -1.3 -0.5 -0.1 -0.6 0.4     1     2804     2927
## log_lik[15] -1.7 -0.4  0.0 -0.6 0.6     1     3519     3574
## log_lik[16] -1.0 -0.3  0.0 -0.3 0.3     1     2272     2882
## log_lik[17] -2.0 -0.5  0.0 -0.7 0.7     1     3665     3475
## log_lik[18] -0.6 -0.2  0.1 -0.3 0.2     1     2232     2922
## log_lik[19] -1.3 -0.3  0.0 -0.5 0.5     1     2643     3102
## log_lik[20] -2.0 -0.9 -0.3 -1.0 0.5     1     3637     3431
## log_lik[21] -1.1 -0.3  0.0 -0.4 0.4     1     2303     3035
## log_lik[22] -0.8 -0.3  0.0 -0.3 0.3     1     2701     3386
## log_lik[23] -0.8 -0.3  0.0 -0.3 0.3     1     2581     3147
## log_lik[24] -2.4 -0.8 -0.2 -1.0 0.7     1     4417     2768
## log_lik[25] -0.9 -0.3  0.0 -0.3 0.3     1     2323     2774
## log_lik[26] -0.6 -0.2  0.1 -0.2 0.2     1     2256     2411
## log_lik[27] -2.0 -0.4  0.0 -0.7 0.7     1     3283     2801
## log_lik[28] -2.1 -0.6 -0.1 -0.8 0.6     1     3307     3195
## log_lik[29] -1.6 -0.4  0.0 -0.5 0.5     1     2638     3318
## log_lik[30] -1.1 -0.3  0.0 -0.4 0.4     1     2448     3080
## log_lik[31] -2.6 -0.6 -0.1 -0.9 0.8     1     3922     3344
## log_lik[32] -1.5 -0.5 -0.1 -0.6 0.5     1     3463     3323
## lp__         2.6  8.8 12.6  8.4 3.1     1     1287     1888
##
## For each parameter, Bulk_ESS and Tail_ESS are crude measures of
## effective sample size for bulk and tail quantities respectively (an ESS > 100
## per chain is considered good), and Rhat is the potential scale reduction
## factor on rank normalized split chains (at convergence, Rhat <= 1.05).
```

Convergence diagnostics

$\hat{R}$ values

ALL obtained $\hat{R}$ values are rather close to 1. $\hat{R}$ is the potential scale reduction factor on split chains (at convergence, $\hat{R}$=1). At convergence, the chains will have mixed, so that the distribution of the simulations between and within chains will be identical, and the ratio $\hat{R}$ should equal 1. In practice the error less than 0.01 is acceptable so that we can say chains probably converged and estimates reliable.

ESS diagnostics

Also we make a discussion about effective sample size(ESS) values. The effective sample size (ESS) of a quantity of interest captures how many independent draws contain the same amount of information as the dependent sample obtained by the MCMC algorithm. From above monitor summary, the obtained Bulk_ESS and Tail_ESS are crude measures of effective sample size for bulk and tail quantities respectively (an $ESS > 100$ per chain is considered good). We can easily give a conclusion that the MCMC algorithm operated properly since the ESS values are much larger than 100.

HMC diagnostics

```r
print("separate-hp")
```

```
## [1] "separate-hp"
```

```r
check_hmc_diagnostics(fit_hp)
```

```
##
## 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.
```

```r
print("separate-wt")
```

```
## [1] "separate-wt"
```

```r
check_hmc_diagnostics(fit_wt)
```

```
##
## 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.
```

```r
print("separate-vs")
```

```
## [1] "separate-vs"
```

```r
check_hmc_diagnostics(fit_vs)
```

```
##
## 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.
```

```r
print("separate-am")
```

```
## [1] "separate-am"
```

```r
check_hmc_diagnostics(fit_am)
```

```
##
## 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.
```

```r
print("separate-crab")
```

```
## [1] "separate-crab"
```

```r
check_hmc_diagnostics(fit_carb)
```

```
##
## 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.
```

```r
print("separate-lphkm")
```

```
## [1] "separate-lphkm"
```

```r
check_hmc_diagnostics(fit_lphkm)
```

```
##
## 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.
```

```r
print("separate-cyl")
```

```
## [1] "separate-cyl"
```

```r
check_hmc_diagnostics(fit_cyl)
```

```
##
## 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.
```

```r
print("separate-disp")
```

```
## [1] "separate-disp"
```

```r
check_hmc_diagnostics(fit_disp)
```

```
##
## 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.
```

```r
print("separate-gear")
```

```
## [1] "separate-gear"
```

```r
check_hmc_diagnostics(fit_gear)
```

```
##
## 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.
```

```r
print("separate-drat")
```

```
## [1] "separate-drat"
```

```r
check_hmc_diagnostics(fit_drat)
```

```
##
## 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.
```

```r
print("multivariate1")
```

```
## [1] "multivariate1"
```

```r
check_hmc_diagnostics(fit_nlin1)
```

```
##
## 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.
```

```r
print("multivariate2")
```

```
## [1] "multivariate2"
```

```r
check_hmc_diagnostics(fit_nlin2)
```

```
##
## 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.
```

```r
print("multivariate3")
```

```
## [1] "multivariate3"
```

```r
check_hmc_diagnostics(fit_nlin3)
```

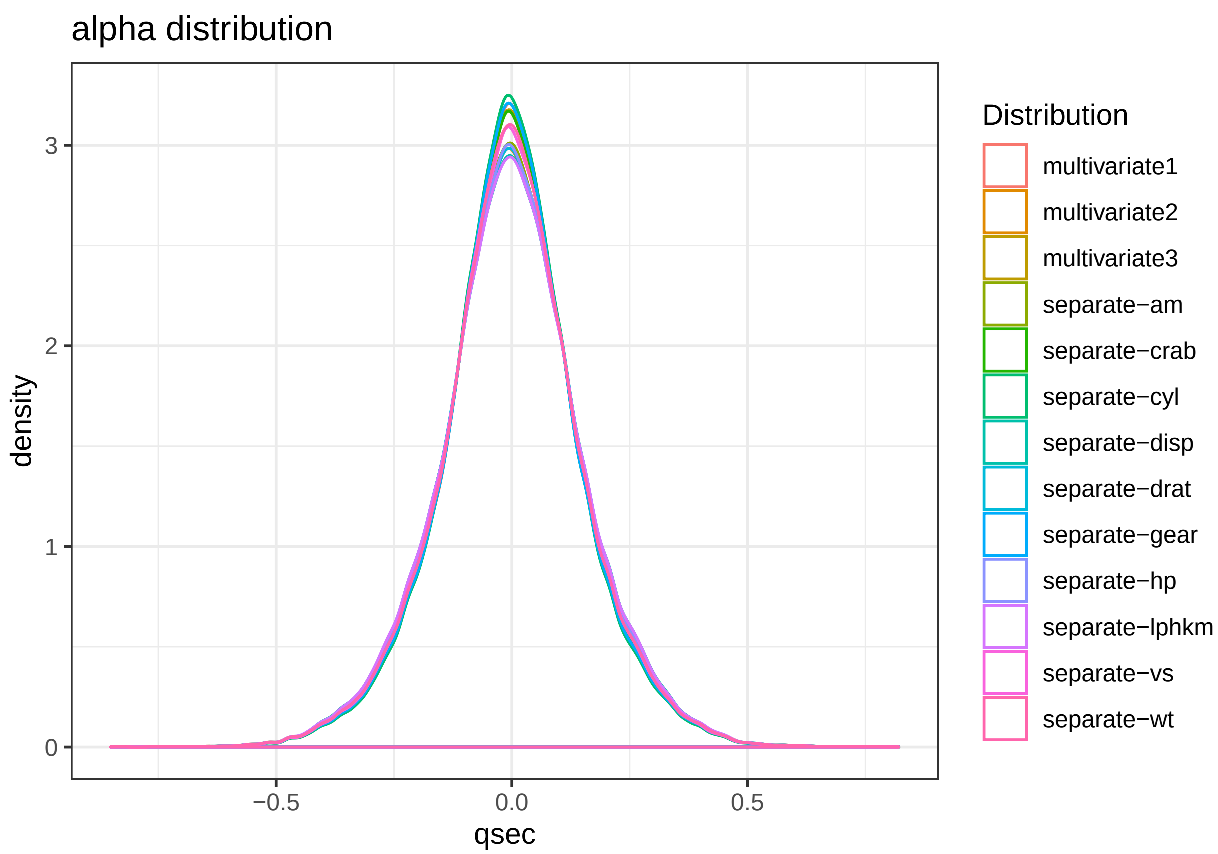
```
##
## Divergences:
```

```
## 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.
```

From the above results we can know that the adaptation phase of the Markov Chains did turn out well and those chains likely did explore the posterior distribution efficiently.

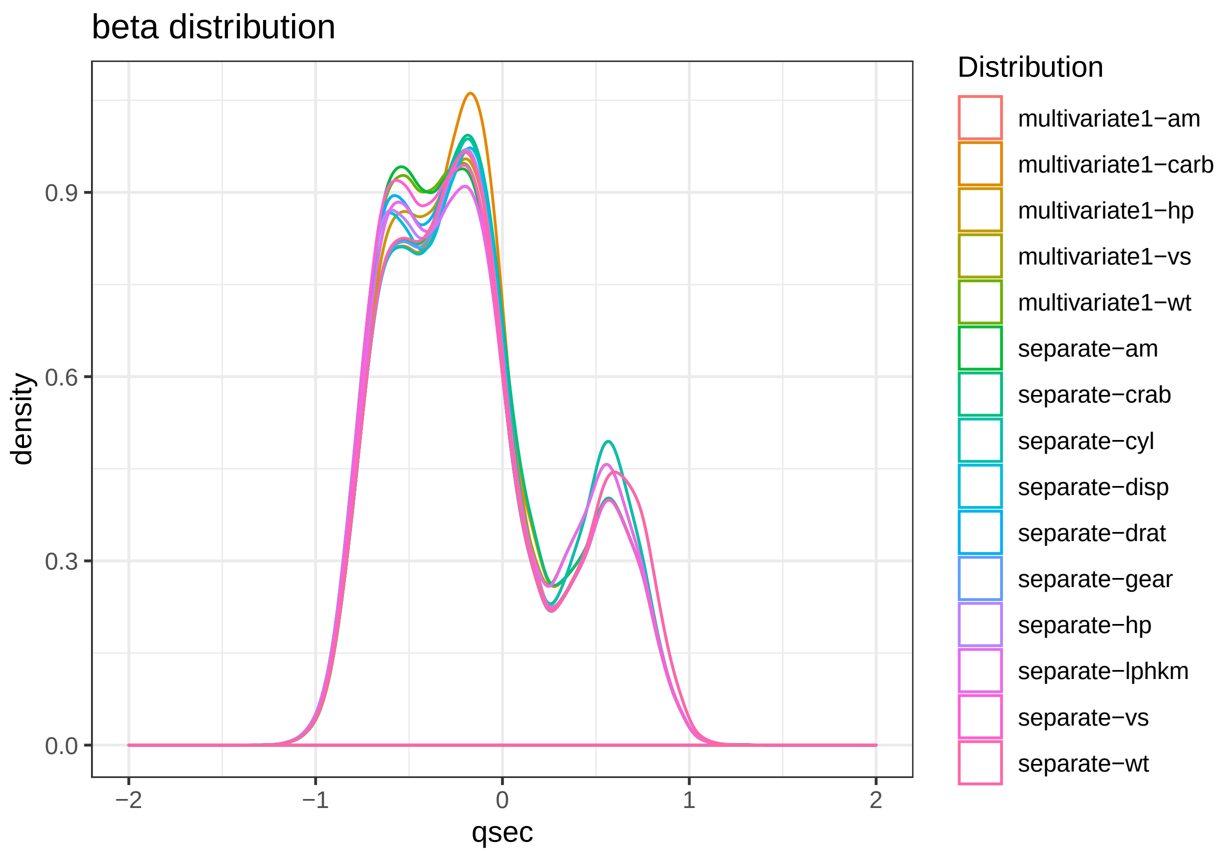
Posterior checking

```
plot_dfa <- data.frame(qsec = c(c(extract(fit_hp, pars = 'alpha', permuted = TRUE)$alpha),
                              c(extract(fit_wt, pars = 'alpha', permuted = TRUE)$alpha),
                              c(extract(fit_vs, pars = 'alpha', permuted = TRUE)$alpha),
                              c(extract(fit_am, pars = 'alpha', permuted = TRUE)$alpha),
                              c(extract(fit_carb, pars = 'alpha', permuted = TRUE)$alpha),
                              c(extract(fit_lphkm, pars = 'alpha', permuted = TRUE)$alpha),
                              c(extract(fit_cyl, pars = 'alpha', permuted = TRUE)$alpha),
                              c(extract(fit_disp, pars = 'alpha', permuted = TRUE)$alpha),
                              c(extract(fit_drat, pars = 'alpha', permuted = TRUE)$alpha),
                              c(extract(fit_gear, pars = 'alpha', permuted = TRUE)$alpha),
                              c(extract(fit_nlin1, pars = 'alpha', permuted = TRUE)$alpha),
                              c(extract(fit_nlin2, pars = 'alpha', permuted = TRUE)$alpha),
                              c(extract(fit_nlin3, pars = 'alpha', permuted = TRUE)$alpha)),
                      Distribution =rep(c("separate-hp","separate-wt","separate-vs",
                                          "separate-am","separate-crab",
                                          "separate-lphkm","separate-cyl",
                                          "separate-disp","separate-gear",
                                          "separate-drat","multivariate1",
                                          "multivariate2","multivariate3"),
                                         times=c(128000,128000,128000,128000,128000,128000,
                                                 128000,128000,128000,128000,128000,128000,128000)))
ggplot(plot_dfa, aes(qsec, color = Distribution)) +
  geom_density() +
  #scale_color_brewer(palette = "Set1")  +
  ggtitle("alpha distribution") +
  theme_bw()
```



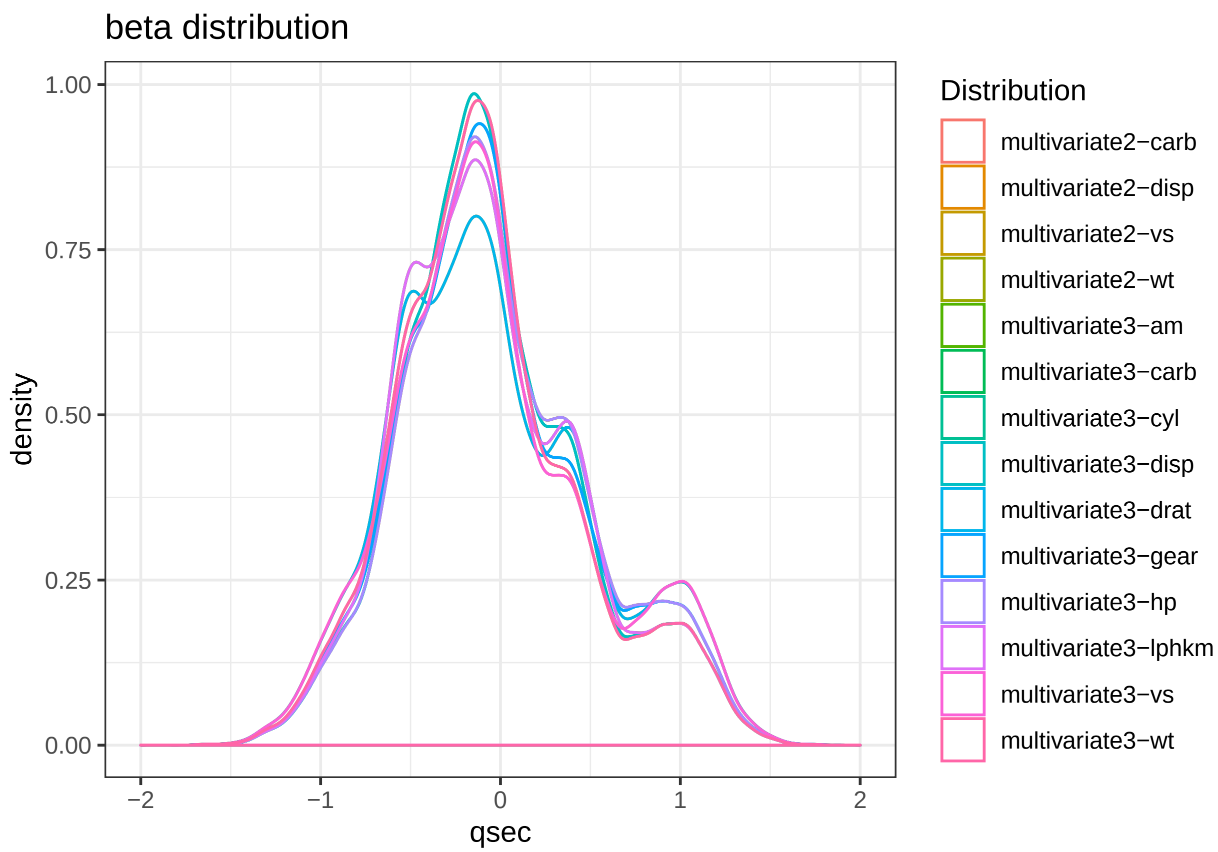

```
plot_dfb1 <- data.frame(qsec = c(c(extract(fit_hp, pars = 'beta', permuted = TRUE)$beta),
                               c(extract(fit_wt, pars = 'beta', permuted = TRUE)$beta),
                               c(extract(fit_vs, pars = 'beta', permuted = TRUE)$beta),
                               c(extract(fit_am, pars = 'beta', permuted = TRUE)$beta),
                               c(extract(fit_carb, pars = 'beta', permuted = TRUE)$beta),
                               c(extract(fit_lphkm, pars = 'beta', permuted = TRUE)$beta),
                               c(extract(fit_cyl, pars = 'beta', permuted = TRUE)$beta),
                               c(extract(fit_disp, pars = 'beta', permuted = TRUE)$beta),
                               c(extract(fit_drat, pars = 'beta', permuted = TRUE)$beta),
                               c(extract(fit_gear, pars = 'beta', permuted = TRUE)$beta),
                               c(extract(fit_nlin1, pars = 'beta_wt', permuted = TRUE)$beta_wt),
                               c(extract(fit_nlin1, pars = 'beta_hp', permuted = TRUE)$beta_hp),
                               c(extract(fit_nlin1, pars = 'beta_vs', permuted = TRUE)$beta_vs),
                               c(extract(fit_nlin1, pars = 'beta_am', permuted = TRUE)$beta_am),
                               c(extract(fit_nlin1, pars = 'beta_carb', permuted = TRUE)$beta_carb)),
                      Distribution = rep(c("separate-hp","separate-wt",
                                           "separate-vs","separate-am",
                                           "separate-crab","separate-lphkm",
                                           "separate-cyl","separate-disp",
                                           "separate-gear","separate-drat",
                                          "multivariate1-wt","multivariate1-hp",
                                           "multivariate1-vs","multivariate1-am",
                                          "multivariate1-carb"),
                                         times=c(128000,128000,128000,128000,128000,
                                                 128000,128000,128000,128000,128000,
                                                 128000,128000,128000,128000,128000)))
```

```r
ggplot(plot_dfb1, aes(qsec, color = Distribution)) +
  geom_density() +
  ggtitle("beta distribution") +
  theme_bw()+
  xlim(-2, 2)
```



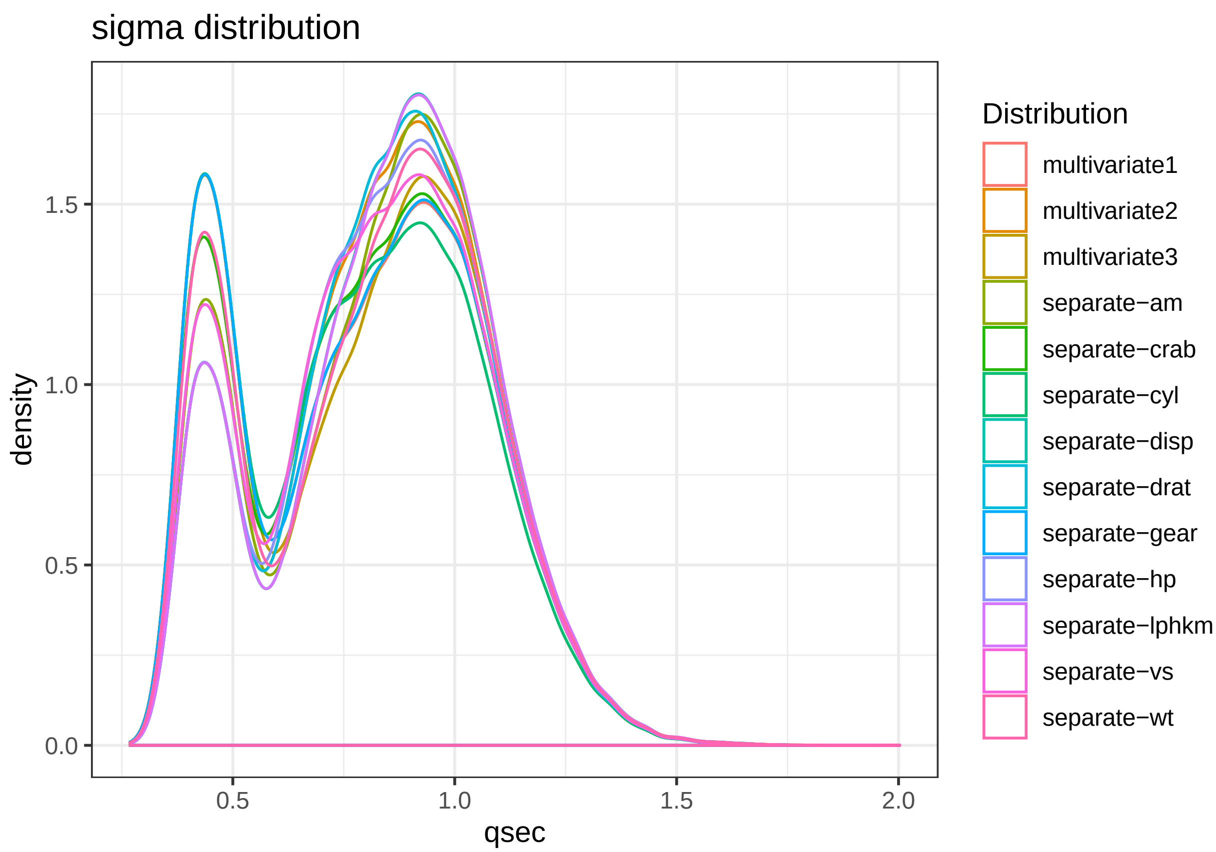

```r
plot_dfb2 <- data.frame(qsec = c(c(extract(fit_nlin2, pars = 'beta_disp', permuted = TRUE)$beta_disp),
                              c(extract(fit_nlin2, pars = 'beta_wt', permuted = TRUE)$beta_wt),
                              c(extract(fit_nlin2, pars = 'beta_vs', permuted = TRUE)$beta_vs),
                              c(extract(fit_nlin2, pars = 'beta_carb', permuted = TRUE)$beta_carb),
                              c(extract(fit_nlin3, pars = 'beta_lphkm', permuted = TRUE)$beta_lphkm),
                              c(extract(fit_nlin3, pars = 'beta_cyl', permuted = TRUE)$beta_cyl),
                              c(extract(fit_nlin3, pars = 'beta_disp', permuted = TRUE)$beta_disp),
                              c(extract(fit_nlin3, pars = 'beta_drat', permuted = TRUE)$beta_drat),
                              c(extract(fit_nlin3, pars = 'beta_wt', permuted = TRUE)$beta_wt),
                              c(extract(fit_nlin3, pars = 'beta_hp', permuted = TRUE)$beta_hp),
                              c(extract(fit_nlin3, pars = 'beta_vs', permuted = TRUE)$beta_vs),
                              c(extract(fit_nlin3, pars = 'beta_am', permuted = TRUE)$beta_am),
                              c(extract(fit_nlin3, pars = 'beta_gear', permuted = TRUE)$beta_gear),
                              c(extract(fit_nlin3, pars = 'beta_carb', permuted = TRUE)$beta_carb)),
                      Distribution = rep(c("multivariate2-disp",
                                           "multivariate2-wt","multivariate2-vs",
                                "multivariate2-carb","multivariate3-lphkm",
                                "multivariate3-cyl","multivariate3-disp",
                              "multivariate3-drat","multivariate3-wt",
                              "multivariate3-hp","multivariate3-vs",
```

```
                            "multivariate3-am","multivariate3-gear",
                            "multivariate3-carb"),
                                      times=c(128000,128000,128000,128000,128000,128000,128000,
                                              128000,128000,128000,128000,128000,128000,128000)))
ggplot(plot_dfb2, aes(qsec, color = Distribution)) +
  geom_density() +
  ggtitle("beta distribution") +
  theme_bw() +
  xlim(-2, 2)
```




```
plot_dfs <- data.frame(qsec = c(c(extract(fit_hp, pars = 'sigma', permuted = TRUE)$sigma),
                               c(extract(fit_wt, pars = 'sigma', permuted = TRUE)$sigma),
                               c(extract(fit_vs, pars = 'sigma', permuted = TRUE)$sigma),
                               c(extract(fit_am, pars = 'sigma', permuted = TRUE)$sigma),
                               c(extract(fit_carb, pars = 'sigma', permuted = TRUE)$sigma),
                               c(extract(fit_lphkm, pars = 'sigma', permuted = TRUE)$sigma),
                               c(extract(fit_cyl, pars = 'sigma', permuted = TRUE)$sigma),
                               c(extract(fit_disp, pars = 'sigma', permuted = TRUE)$sigma),
                               c(extract(fit_drat, pars = 'sigma', permuted = TRUE)$sigma),
                               c(extract(fit_gear, pars = 'sigma', permuted = TRUE)$sigma),
                               c(extract(fit_nlin1, pars = 'sigma', permuted = TRUE)$sigma),
                               c(extract(fit_nlin2, pars = 'sigma', permuted = TRUE)$sigma),
                               c(extract(fit_nlin3, pars = 'sigma', permuted = TRUE)$sigma)),
                      Distribution = rep(c("separate-hp","separate-wt",
                                           "separate-vs",
                                          "separate-am","separate-crab",
```

```r
                                                   "separate-lphkm","separate-cyl",
                                                   "separate-disp","separate-gear",
                                                   "separate-drat","multivariate1",
                                                   "multivariate2","multivariate3"),
                                                 times=c(128000,128000,128000,128000,128000,
                                                         128000,128000,128000,128000,128000,
                                                         128000,128000,128000)))
ggplot(plot_dfs, aes(qsec, color = Distribution)) +
  geom_density() +
  #scale_color_brewer(palette = "Set1") +
  ggtitle("sigma distribution") +
  theme_bw()
```



We sample and obtain posterior distributions for the parameters in the models. The posterior predictive distributions for Quarter mile time (how fast the car can traverse a quarter mile) the by various models are shown above.

α is the intercept term of the linear function and it ensures that the model will be unbiased–i.e., the mean of the residuals will be exactly zero. α also represents the Y-intercept of the regression line. We can notice that all models give the quiet similar distribution and present that the mean is close to zero.

σ is the variance (squared scale) of the estimation and it will affect the shape of the final normal distribution. The σ predictive distributions above follow the same trend and have the close peak position. Consequently, the shape of qsec predictive distributions won't affect by the variance parameter.

The interpretation of β is the expected change in qsec for a one-unit change in corresponding parameter when the other covariates are held fixed. We can find that the parameters from multivariate model 1 share the similar change performance with the univariate model. In addition, the shape of β predictive distributions from multivariate model 2 and multivariate model follow the same trend. It is possible that the best subset selected by the *regsubsets* function is significant.

Performance analysis

```
log_lik_hp <- extract_log_lik(fit_hp, merge_chains = FALSE)
loo_hp<-loo(log_lik_hp, r_eff = relative_eff(exp(log_lik_hp)))
```

```
## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.
```

```
log_lik_wt <- extract_log_lik(fit_wt, merge_chains = FALSE)
loo_wt<-loo(log_lik_wt, r_eff = relative_eff(exp(log_lik_wt)))
```

```
## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for deta
```

```
log_lik_vs <- extract_log_lik(fit_vs, merge_chains = FALSE)
loo_vs<-loo(log_lik_vs, r_eff = relative_eff(exp(log_lik_vs)))
```

```
## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for deta
```

```
log_lik_am <- extract_log_lik(fit_am, merge_chains = FALSE)
loo_am<-loo(log_lik_am, r_eff = relative_eff(exp(log_lik_am)))
```

```
## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for deta
```

```
log_lik_carb <- extract_log_lik(fit_carb, merge_chains = FALSE)
loo_carb<-loo(log_lik_carb, r_eff = relative_eff(exp(log_lik_carb)))
```

```
## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for deta
```

```
log_lik_lphkm <- extract_log_lik(fit_lphkm, merge_chains = FALSE)
loo_lphkm<-loo(log_lik_lphkm, r_eff = relative_eff(exp(log_lik_lphkm)))
log_lik_cyl <- extract_log_lik(fit_cyl, merge_chains = FALSE)
loo_cyl<-loo(log_lik_cyl, r_eff = relative_eff(exp(log_lik_cyl)))
log_lik_disp <- extract_log_lik(fit_disp, merge_chains = FALSE)
loo_disp<-loo(log_lik_disp, r_eff = relative_eff(exp(log_lik_disp)))
```

```
## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for deta
```

```
log_lik_gear <- extract_log_lik(fit_gear, merge_chains = FALSE)
loo_gear<-loo(log_lik_gear, r_eff = relative_eff(exp(log_lik_gear)))
```

```
## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for deta
```

```
log_lik_drat <- extract_log_lik(fit_drat, merge_chains = FALSE)
loo_drat<-loo(log_lik_drat, r_eff = relative_eff(exp(log_lik_drat)))
log_lik_nlin1 <- extract_log_lik(fit_nlin1, merge_chains = FALSE)
loo_nlin1<-loo(log_lik_nlin1, r_eff = relative_eff(exp(log_lik_nlin1)))
```

```
## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.
```

```
## Warning in log(z): NaNs produced
```

```
log_lik_nlin2 <- extract_log_lik(fit_nlin2, merge_chains = FALSE)
loo_nlin2<-loo(log_lik_nlin2, r_eff = relative_eff(exp(log_lik_nlin2)))
```

```
## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.
```

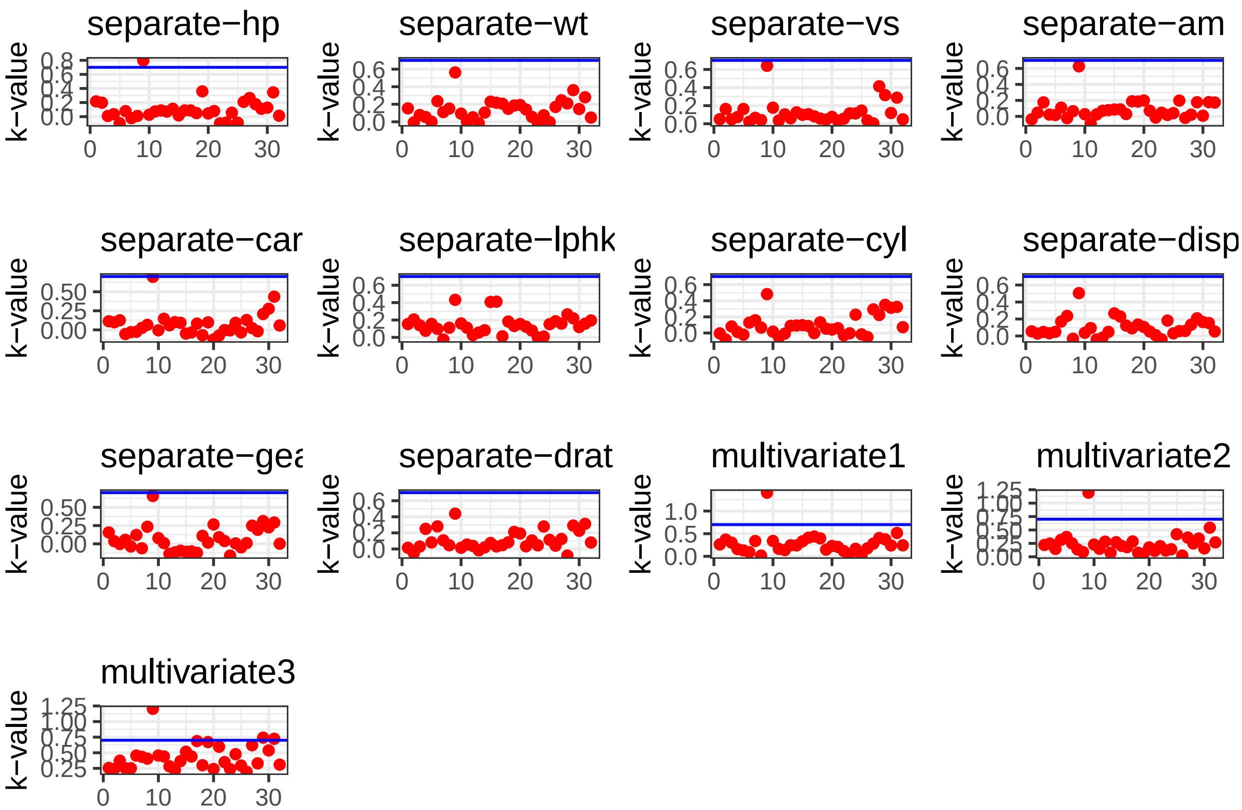
```
log_lik_nlin3 <- extract_log_lik(fit_nlin3, merge_chains = FALSE)
loo_nlin3<-loo(log_lik_nlin3, r_eff = relative_eff(exp(log_lik_nlin3)))
```

```
## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.

## Warning: NaNs produced
```

```r
kv <- data.frame(k = loo_hp$diagnostics$pareto_k,
                      x = 1:length(loo_hp$diagnostics$pareto_k))
p1<-ggplot(data=kv, aes(x=x, y=k)) +
    geom_point(color = 'red') +
    geom_hline(yintercept=0.7, color = 'blue') +
    ggtitle("separate-hp") +
    xlab("") +
    ylab("k-value") +
    theme_bw()

kv <- data.frame(k = loo_wt$diagnostics$pareto_k,
                      x = 1:length(loo_wt$diagnostics$pareto_k))
p2<-ggplot(data=kv, aes(x=x, y=k)) +
    geom_point(color = 'red') +
    geom_hline(yintercept=0.7, color = 'blue') +
    ggtitle("separate-wt") +
    xlab("") +
    ylab("k-value") +
    theme_bw()


kv <- data.frame(k = loo_vs$diagnostics$pareto_k,
                      x = 1:length(loo_vs$diagnostics$pareto_k))
p3<-ggplot(data=kv, aes(x=x, y=k)) +
    geom_point(color = 'red') +
    geom_hline(yintercept=0.7, color = 'blue') +
    ggtitle("separate-vs") +
    xlab("") +
    ylab("k-value") +
    theme_bw()

kv <- data.frame(k = loo_am$diagnostics$pareto_k,
                      x = 1:length(loo_am$diagnostics$pareto_k))
p4<-ggplot(data=kv, aes(x=x, y=k)) +
    geom_point(color = 'red') +
    geom_hline(yintercept=0.7, color = 'blue') +
    ggtitle("separate-am") +
    xlab("") +
    ylab("k-value") +
    theme_bw()

kv <- data.frame(k = loo_carb$diagnostics$pareto_k,
                      x = 1:length(loo_carb$diagnostics$pareto_k))
p5<-ggplot(data=kv, aes(x=x, y=k)) +
    geom_point(color = 'red') +
    geom_hline(yintercept=0.7, color = 'blue') +
    ggtitle("separate-carb") +
    xlab("") +
    ylab("k-value") +
    theme_bw()
```

```
kv <- data.frame(k = loo_lphkm$diagnostics$pareto_k,
                 x = 1:length(loo_lphkm$diagnostics$pareto_k))
p6<-ggplot(data=kv, aes(x=x, y=k)) +
    geom_point(color = 'red') +
    geom_hline(yintercept=0.7, color = 'blue') +
    ggtitle("separate-lphkm") +
    xlab("") +
    ylab("k-value") +
    theme_bw()

kv <- data.frame(k = loo_cyl$diagnostics$pareto_k,
                 x = 1:length(loo_cyl$diagnostics$pareto_k))
p7<-ggplot(data=kv, aes(x=x, y=k)) +
    geom_point(color = 'red') +
    geom_hline(yintercept=0.7, color = 'blue') +
    ggtitle("separate-cyl") +
    xlab("") +
    ylab("k-value") +
    theme_bw()


kv <- data.frame(k = loo_disp$diagnostics$pareto_k,
                 x = 1:length(loo_disp$diagnostics$pareto_k))
p8<-ggplot(data=kv, aes(x=x, y=k)) +
    geom_point(color = 'red') +
    geom_hline(yintercept=0.7, color = 'blue') +
    ggtitle("separate-disp") +
    xlab("") +
    ylab("k-value") +
    theme_bw()

kv <- data.frame(k = loo_gear$diagnostics$pareto_k,
                 x = 1:length(loo_gear$diagnostics$pareto_k))
p9<-ggplot(data=kv, aes(x=x, y=k)) +
    geom_point(color = 'red') +
    geom_hline(yintercept=0.7, color = 'blue') +
    ggtitle("separate-gear") +
    xlab("") +
    ylab("k-value") +
    theme_bw()

kv <- data.frame(k = loo_drat$diagnostics$pareto_k,
                 x = 1:length(loo_drat$diagnostics$pareto_k))
p10<-ggplot(data=kv, aes(x=x, y=k)) +
    geom_point(color = 'red') +
    geom_hline(yintercept=0.7, color = 'blue') +
    ggtitle("separate-drat") +
    xlab("") +
    ylab("k-value") +
    theme_bw()

kv <- data.frame(k = loo_nlin1$diagnostics$pareto_k,
                 x = 1:length(loo_nlin1$diagnostics$pareto_k))
```

```r
p11<-ggplot(data=kv, aes(x=x, y=k)) +
    geom_point(color = 'red') +
    geom_hline(yintercept=0.7, color = 'blue') +
    ggtitle("multivariate1") +
    xlab("") +
    ylab("k-value") +
    theme_bw()

kv <- data.frame(k = loo_nlin2$diagnostics$pareto_k,
                      x = 1:length(loo_nlin2$diagnostics$pareto_k))
p12<-ggplot(data=kv, aes(x=x, y=k)) +
    geom_point(color = 'red') +
    geom_hline(yintercept=0.7, color = 'blue') +
    ggtitle("multivariate2") +
    xlab("") +
    ylab("k-value") +
    theme_bw()

kv <- data.frame(k = loo_nlin3$diagnostics$pareto_k,
                      x = 1:length(loo_nlin3$diagnostics$pareto_k))
p13<-ggplot(data=kv, aes(x=x, y=k)) +
    geom_point(color = 'red') +
    geom_hline(yintercept=0.7, color = 'blue') +
    ggtitle("multivariate3") +
    xlab("") +
    ylab("k-value") +
    theme_bw()

grid.arrange(p1, p2, p3, p4,p5,p6,p7,p8,p9,p10,p11,p12,p13)
```




```r
(df<-data.frame("model"=c("separate-hp","separate-wt",
                          "separate-vs",
                          "separate-am","separate-crab",
                          "separate-lphkm",
                          "separate-cyl","separate-disp",
                          "separate-gear",
                          "separate-drat","multivariate1",
                          "multivariate2", "multivariate3"),
                "PSIS_LOO"=c(loo_hp$estimates[1],loo_wt$estimates[1],loo_vs$estimates[1],
                  loo_am$estimates[1],loo_carb$estimates[1],loo_lphkm$estimates[1],
                  loo_cyl$estimates[1],loo_disp$estimates[1],loo_gear$estimates[1],
                  loo_drat$estimates[1],loo_nlin1$estimates[1],loo_nlin2$estimates[1],loo_nlin3$estimate
                "p_eff"=c(loo_hp$estimates[2],loo_wt$estimates[2],loo_vs$estimates[2],
                  loo_am$estimates[2],loo_carb$estimates[2],loo_lphkm$estimates[2],
                  loo_cyl$estimates[2],loo_disp$estimates[2],loo_gear$estimates[2],
                  loo_drat$estimates[2],loo_nlin1$estimates[2],loo_nlin2$estimates[2],loo_nlin3$estimate
```

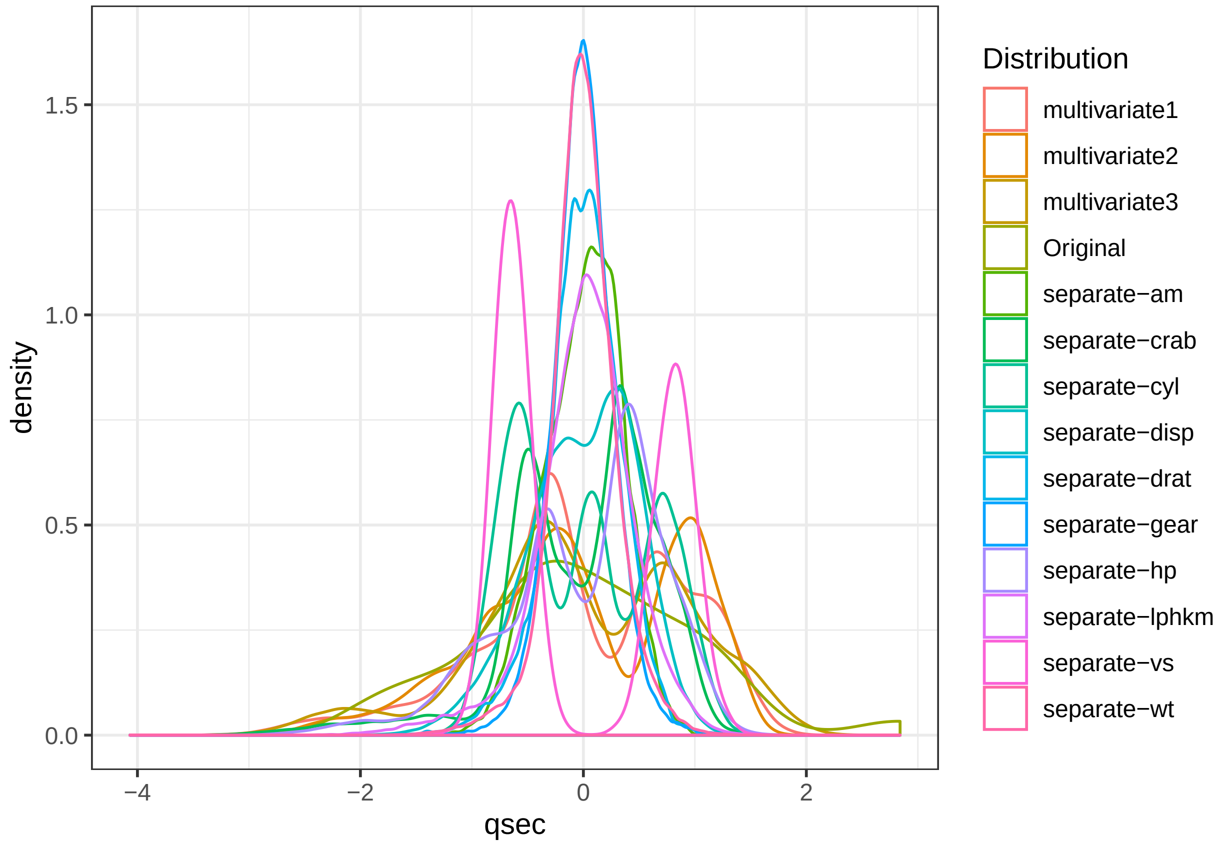
```
##             model  PSIS_LOO    p_eff
## 1     separate-hp -37.53700 3.580252
## 2     separate-wt -47.47434 2.613361
## 3     separate-vs -35.59620 3.470713
## 4     separate-am -47.39368 3.027007
## 5   separate-crab -39.52200 3.396820
## 6  separate-lphkm -45.79303 3.097921
## 7    separate-cyl -41.22803 2.858254
## 8   separate-disp -44.88257 2.940425
```

```
## 9   separate-gear -47.61016  3.179571
## 10  separate-drat -47.97634  2.807068
## 11  multivariate1 -27.27809  9.861408
## 12  multivariate2 -23.72125  7.715821
## 13  multivariate3 -28.73662 12.566629
```

We use leave-one-out cross validation (LOO-cv) to compare the performance of the models. With a few exceptions, most of the k-values seem to be lower than 0.7 so that we can trust the results. Next, let's look at the PSIS-LOO values. The higher the value, the better the performance of the model. The multivariate model 2 has highest PSIS-LOO value and shows better performance than multivariate model 1, multivariate model 3 or any other single-variable model. It seems that we need to trust the computer rather than our prior world information.

Model performance assessment

```r
plot_df <- data.frame(qsec = c(c(extract(fit_hp, pars = 'mu', permuted = TRUE)$mu),
                               c(extract(fit_wt, pars = 'mu', permuted = TRUE)$mu),
                               c(extract(fit_vs, pars = 'mu', permuted = TRUE)$mu),
                               c(extract(fit_am, pars = 'mu', permuted = TRUE)$mu),
                               c(extract(fit_carb, pars = 'mu', permuted = TRUE)$mu),
                               c(extract(fit_lphkm, pars = 'mu', permuted = TRUE)$mu),
                               c(extract(fit_cyl, pars = 'mu', permuted = TRUE)$mu),
                               c(extract(fit_disp, pars = 'mu', permuted = TRUE)$mu),
                               c(extract(fit_drat, pars = 'mu', permuted = TRUE)$mu),
                               c(extract(fit_gear, pars = 'mu', permuted = TRUE)$mu),
                               c(extract(fit_nlin1, pars = 'mu', permuted = TRUE)$mu),
                               c(extract(fit_nlin2, pars = 'mu', permuted = TRUE)$mu),
                               c(extract(fit_nlin3, pars = 'mu', permuted = TRUE)$mu),
                                  scaled_car_properties$qsec),
                      Distribution = rep(c("separate-hp","separate-wt",
                                           "separate-vs","separate-am",
                                           "separate-crab","separate-lphkm",
                                           "separate-cyl","separate-disp",
                                           "separate-gear","separate-drat",
                                           "multivariate1",
                                           "multivariate2","multivariate3" ,
                                           "Original"),
                                         times=c(128000,128000,128000,128000,128000,
                                                 128000,128000,128000,128000,128000,
                                                 128000, 128000,128000,nrow(scaled_car_properties))))
ggplot(plot_df, aes(qsec, color = Distribution)) +
  geom_density() +
  #scale_color_brewer(palette = "Set1") +
  theme_bw()
```



The predictive distributions are plotted above. The multivariate models's distribution is very close to the original one while the other single variable models change the peak of the density from the original one.

```
hp_lin<-c(mean(extract(fit_hp, pars = 'alpha', permuted = TRUE)$alpha),
          mean(extract(fit_hp, pars = 'beta', permuted = TRUE)$beta))
vs_lin<-c(mean(extract(fit_vs, pars = 'alpha', permuted = TRUE)$alpha),
          mean(extract(fit_vs, pars = 'beta', permuted = TRUE)$beta))
wt_lin<-c(mean(extract(fit_wt, pars = 'alpha', permuted = TRUE)$alpha),
          mean(extract(fit_wt, pars = 'beta', permuted = TRUE)$beta))
am_lin<-c(mean(extract(fit_am, pars = 'alpha', permuted = TRUE)$alpha),
          mean(extract(fit_am, pars = 'beta', permuted = TRUE)$beta))
carb_lin<-c(mean(extract(fit_carb, pars = 'alpha', permuted = TRUE)$alpha),
          mean(extract(fit_carb, pars = 'beta', permuted = TRUE)$beta))
lphkm_lin<-c(mean(extract(fit_lphkm, pars = 'alpha', permuted = TRUE)$alpha),
          mean(extract(fit_lphkm, pars = 'beta', permuted = TRUE)$beta))
cyl_lin<-c(mean(extract(fit_cyl, pars = 'alpha', permuted = TRUE)$alpha),
          mean(extract(fit_cyl, pars = 'beta', permuted = TRUE)$beta))
disp_lin<-c(mean(extract(fit_disp, pars = 'alpha', permuted = TRUE)$alpha),
          mean(extract(fit_disp, pars = 'beta', permuted = TRUE)$beta))
gear_lin<-c(mean(extract(fit_gear, pars = 'alpha', permuted = TRUE)$alpha),
          mean(extract(fit_gear, pars = 'beta', permuted = TRUE)$beta))
drat_lin<-c(mean(extract(fit_drat, pars = 'alpha', permuted = TRUE)$alpha),
          mean(extract(fit_drat, pars = 'beta', permuted = TRUE)$beta))
n_lin1<-c(mean(extract(fit_nlin1, pars = 'alpha', permuted = TRUE)$alpha),
         mean(extract(fit_nlin1, pars = 'beta_hp', permuted = TRUE)$beta_hp),
         mean(extract(fit_nlin1, pars = 'beta_wt', permuted = TRUE)$beta_wt),
         mean(extract(fit_nlin1, pars = 'beta_vs', permuted = TRUE)$beta_vs),
```

```
         mean(extract(fit_nlin1, pars = 'beta_am', permuted = TRUE)$beta_am),
          mean(extract(fit_nlin1, pars = 'beta_carb', permuted = TRUE)$beta_carb))
n_lin2<-c(mean(extract(fit_nlin2, pars = 'alpha', permuted = TRUE)$alpha),
         mean(extract(fit_nlin2, pars = 'beta_disp ', permuted = TRUE)$beta_disp),
         mean(extract(fit_nlin2, pars = 'beta_wt', permuted = TRUE)$beta_wt),
         mean(extract(fit_nlin2, pars = 'beta_vs', permuted = TRUE)$beta_vs),
         mean(extract(fit_nlin2, pars = 'beta_carb ', permuted = TRUE)$beta_carb))
n_lin3<-c(mean(extract(fit_nlin3, pars = 'alpha', permuted = TRUE)$alpha),
         mean(extract(fit_nlin3, pars = 'beta_lphkm', permuted = TRUE)$beta_lphkm),
         mean(extract(fit_nlin3, pars = 'beta_cyl', permuted = TRUE)$beta_cyl),
         mean(extract(fit_nlin3, pars = 'beta_disp', permuted = TRUE)$beta_disp),
         mean(extract(fit_nlin3, pars = 'beta_hp', permuted = TRUE)$beta_hp),
         mean(extract(fit_nlin3, pars = 'beta_drat', permuted = TRUE)$beta_drat),
         mean(extract(fit_nlin3, pars = 'beta_wt', permuted = TRUE)$beta_wt),
         mean(extract(fit_nlin3, pars = 'beta_vs', permuted = TRUE)$beta_vs),
         mean(extract(fit_nlin3, pars = 'beta_am', permuted = TRUE)$beta_am),
         mean(extract(fit_nlin3, pars = 'beta_gear', permuted = TRUE)$beta_gear),
          mean(extract(fit_nlin3, pars = 'beta_carb', permuted = TRUE)$beta_carb))

(res_df<-data.frame("separate-hp"=c(scaled_car_properties$qsec-
                                      hp_lin[1]+hp_lin[2]*scaled_car_properties$hp),
               "separate-wt"=c(scaled_car_properties$qsec-
                                 wt_lin[1]+wt_lin[2]*scaled_car_properties$wt),
               "separate-vs"=c(scaled_car_properties$qsec-
                                 vs_lin[1]+vs_lin[2]*scaled_car_properties$vs),
               "separate-am"=c(scaled_car_properties$qsec-
                                 am_lin[1]+am_lin[2]*scaled_car_properties$am),
               "separate-carb"=c(scaled_car_properties$qsec-
                                   carb_lin[1]+carb_lin[2]*scaled_car_properties$carb),
               "separate-lphkm"=c(scaled_car_properties$qsec-
                                    lphkm_lin[1]+lphkm_lin[2]*scaled_car_properties$lphkm),
               "separate-cyl"=c(scaled_car_properties$qsec-
                                  cyl_lin[1]+cyl_lin[2]*scaled_car_properties$cyl),
               "separate-disp"=c(scaled_car_properties$qsec-
                                   disp_lin[1]+disp_lin[2]*scaled_car_properties$disp),
               "separate-gear"=c(scaled_car_properties$qsec-
                                   gear_lin[1]+gear_lin[2]*scaled_car_properties$gear),
               "separate-drat"=c(scaled_car_properties$qsec-
                                   drat_lin[1]+drat_lin[2]*scaled_car_properties$drat),
               "multivariate1"=c(scaled_car_properties$qsec-
                                   (n_lin1[1]+n_lin1[2]*scaled_car_properties$hp+
                 n_lin1[3]*scaled_car_properties$wt+n_lin1[4]*scaled_car_properties$vs+
                 n_lin1[5]*scaled_car_properties$am+n_lin1[6]*scaled_car_properties$carb)),
               "multivariate2"=c(scaled_car_properties$qsec-
                                   (n_lin2[1]+n_lin2[2]*scaled_car_properties$disp+
                 n_lin2[3]*scaled_car_properties$wt+n_lin2[4]*scaled_car_properties$vs+
                 n_lin2[5]*scaled_car_properties$carb)),
               "multivariate3"=c(scaled_car_properties$qsec-
                                   (n_lin3[1]+n_lin3[2]*scaled_car_properties$lphkm+
                 n_lin3[3]*scaled_car_properties$cyl+n_lin3[4]*scaled_car_properties$disp+
                 n_lin3[5]*scaled_car_properties$hp+n_lin3[6]*scaled_car_properties$drat+
                 n_lin3[7]*scaled_car_properties$wt+n_lin3[8]*scaled_car_properties$vs+
                   n_lin3[9]*scaled_car_properties$am+n_lin3[10]*scaled_car_properties$gear+
```

```
                  n_lin3[11]*scaled_car_properties$carb))))
```

```
##    separate.hp separate.wt separate.vs separate.am separate.carb
## 1  -0.39862169 -0.66958484  -1.4154879 -1.05556140   -1.25702881
## 2  -0.08523737 -0.40066047  -1.1021035 -0.74217707   -0.94364449
## 3   0.97932396  0.58589295   1.2538641  0.14761056    1.16185907
## 4   1.26903061  0.89432756   1.7183444  1.07536146    1.62633941
## 5  -0.75348973 -0.49916979  -1.1021035 -0.27890651   -0.13316721
## 6   1.75693423  1.28811073   2.1548440  1.51186106    2.06283900
## 7  -2.13349166 -1.18218119  -1.7624491 -0.93925206   -1.60399003
## 8   2.07589360  1.21207071   2.0317287  1.38874579    1.53448509
## 9   3.35951013  2.84192793   3.6546119  3.01162890    3.15736821
## 10  0.49741919  0.21713724   1.0803835  0.43740051   -0.22733746
## 11  0.83318811  0.55290616   1.4161524  0.77316943    0.10843146
## 12 -0.59224011 -0.39635838  -0.8894499 -0.06625287   -0.32575220
## 13 -0.48031713 -0.22515547  -0.7775269  0.04567011   -0.21382923
## 14 -0.25647119 -0.01002716  -0.5536810  0.26951605    0.01001672
## 15 -0.52468362 -0.27751802  -0.5648733  0.25832376   -0.40641422
## 16 -0.71703006 -0.39739378  -0.6544116  0.16878538   -0.49595260
## 17 -1.09508809 -0.60746586  -0.8782576 -0.05506057   -0.71979854
## 18  1.73817450  1.08808407   1.7351329  0.62887935    1.64312785
## 19  1.35047165  0.65844631   1.2034987  0.09724523    0.70625509
## 20  1.98908970  1.39235722   1.9757673  0.86951374    1.88376225
## 21  1.72166156  1.34407261   2.0373249  1.39434194    1.94531988
## 22 -0.58041182 -0.59706024  -1.1860458 -0.36284874   -0.21710944
## 23 -0.33977743 -0.34160586  -0.9454114 -0.12221435    0.02352495
## 24 -2.37412605 -1.46989083  -2.0030835 -1.17988645   -1.84462442
## 25 -0.73670128 -0.55299421  -1.0853151 -0.26211807   -0.11637876
## 26  1.41919403  0.81530708   1.4161524  0.30989888    1.32414738
## 27 -0.06897882 -0.45158795  -1.2811803 -0.92125383   -0.31224397
## 28 -0.18323357 -0.23034579   0.2969227 -0.80933086   -0.20032099
## 29 -3.07871088 -1.86232400  -2.5123330 -2.15240654   -2.35387395
## 30 -1.60410432 -1.23296803  -1.9527181 -1.59279167   -2.60473637
## 31 -3.75268659 -1.87610362  -2.4563715 -2.09644505   -3.91886703
## 32  0.80923493  0.50009453   1.2482679  0.14201442    0.75102428
##    separate.lphkm separate.cyl separate.disp separate.gear separate.drat
## 1     -0.62735317  -0.71816124   -0.53236270   -0.86428506  -0.724438751
## 2     -0.31396884  -0.40477692   -0.21897837   -0.55090073  -0.411054426
## 3      0.66260935   1.14218305    0.84951709    0.33888690   0.470742597
## 4      1.06084768   0.94949106    0.79849409    1.09052905   0.812167471
## 5     -0.44918266  -1.06194925   -0.90631616   -0.26373892  -0.530913645
## 6      1.30066024   1.38599065    1.34840442    1.52702865   1.197527133
## 7     -1.48939310  -1.72229479   -1.56666170   -0.92408446  -1.181670450
## 8      1.50687117   1.92004771    1.49438189    1.11675157   1.223037294
## 9      3.06335713   3.54293082    3.13754147    2.73963468   2.882677233
## 10     0.29927439   0.31153011    0.47120982    0.16540630   0.308448848
## 11     0.54047167   0.64729903    0.80697874    0.50117522   0.644217768
## 12    -0.40966865  -0.84929560   -0.40429330   -0.05108527  -0.331044979
## 13    -0.22451270  -0.73737263   -0.29237033    0.06083770  -0.219122006
## 14    -0.18503399  -0.51352668   -0.06852439    0.28468365   0.004723941
## 15    -0.89722697  -0.52471898   -0.75399505    0.27349135  -0.028842078
## 16    -0.98676535  -0.61425735   -0.80229316    0.18395297  -0.107193596
## 17    -0.56127166  -0.83810330   -0.95740533   -0.03989298  -0.294282715
## 18     1.44389489   1.62345183    1.43148086    0.82015569   0.988768209
```

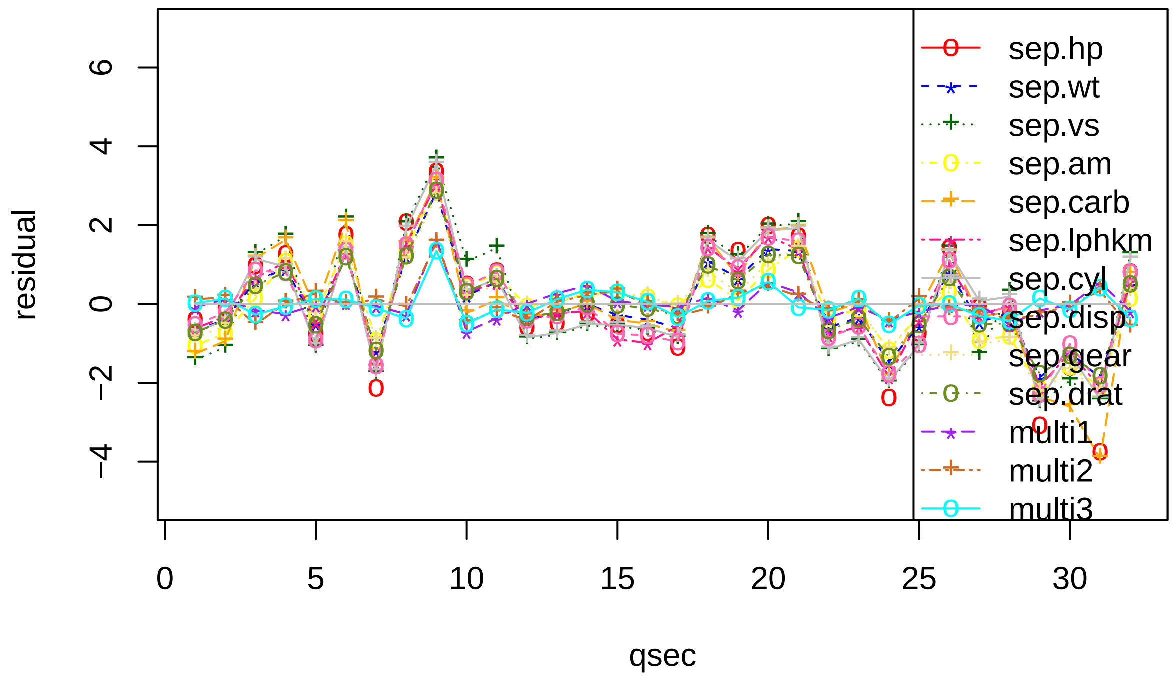
```
## 19     0.86538315   1.09181771    0.91015680    0.28852157   0.592974534
## 20     1.71605759   1.86408622    1.69823409    1.06079008   1.251776322
## 21     1.38484582   1.92564386    1.59139396    1.40950953   1.230231565
## 22    -0.78800198  -1.14589148   -0.84591746   -0.34768115  -0.677182669
## 23    -0.57676439  -0.90525709   -0.55716942   -0.10704676  -0.374221482
## 24    -1.85141289  -1.96292918   -1.77292921   -1.16471886  -1.339202450
## 25    -0.40024419  -1.04516080   -1.02699527   -0.24695048  -0.525312059
## 26     0.99180259   1.30447136    1.11146938    0.50117522   0.669787735
## 27    -0.28163270   0.07331865   -0.26161858   -1.01713930  -0.505430669
## 28    -0.04119293   0.18524162   -0.06309105   -0.90521633  -0.498983808
## 29    -2.08600875  -2.47217871   -2.28561543   -2.24829201  -1.770143956
## 30    -1.23712919  -1.25539152   -1.01804264   -1.68867714  -1.306416465
## 31    -2.10797566  -2.41621722   -2.05781949   -2.19233052  -1.822854827
## 32     0.59077119   1.13658690    0.79924398    0.33329076   0.506697644
##    multivariate1 multivariate2 multivariate3
## 1   -0.080944711    0.11919445  0.0201283457
## 2    0.118667768    0.16770156  0.1398213678
## 3   -0.156865593   -0.51876537 -0.2714849266
## 4   -0.254508932    0.02039198 -0.0567078860
## 5   -0.027404028    0.27106310  0.1256197749
## 6    0.031736681   -0.01855401  0.1134396853
## 7   -0.048295480    0.12456124 -0.1019295356
## 8   -0.260909501   -0.06103911 -0.3757156945
## 9    1.650049107    1.56388917  1.3478361130
## 10  -0.700039175   -0.48325019 -0.5010312618
## 11  -0.364270256   -0.14748127 -0.1372160054
## 12   0.007231283   -0.41001806 -0.2254575015
## 13   0.270850051    0.05507454  0.1230024391
## 14   0.472387792    0.22698377  0.3635459766
## 15   0.072174421    0.32697387  0.2951555004
## 16  -0.013108956   -0.02365178  0.0539847592
## 17  -0.078876606   -0.29935028 -0.3317042459
## 18   0.156846485   -0.10903037  0.0767602184
## 19  -0.166302947    0.27134770  0.1602138601
## 20   0.552142020    0.45985524  0.5651590084
## 21   0.292640797    0.19509608 -0.0857364892
## 22  -0.351758274   -0.17719422 -0.1518834372
## 23  -0.073199933    0.05799379  0.1353272937
## 24  -0.409394180   -0.46348766 -0.5177117970
## 25  -0.191312043    0.13498929 -0.0421609244
## 26  -0.043900501   -0.15073759 -0.0004345865
## 27  -0.012307266   -0.16260110 -0.3026309115
## 28  -0.527856205   -0.39938233 -0.4276161150
## 29  -0.162111755   -0.27008965  0.1397349073
## 30  -0.028587952   -0.02540123 -0.1390616211
## 31   0.545270806    0.33335504  0.4172485833
## 32  -0.174555872   -0.59069653 -0.3742524178
```

```r
(res_mean<-data.frame("model"=c("separate-hp","separate-wt","separate-vs","separate-am",
                              "separate-crab","separate-lphkm","separate-cyl","separate-disp",
                              "separate-gear","separate-drat","multivariate1",
                              "multivariate2","multivariate3"),
                  "mean.res"=c(mean(t(res_df["separate.hp"])),mean(t(res_df["separate.wt"])),
                              mean(t(res_df["separate.vs"])),mean(t(res_df["separate.am"])),
```

```r
                              mean(t(res_df["separate.carb"])),mean(t(res_df["separate.lphkm"])),
                              mean(t(res_df["separate.cyl"])),mean(t(res_df["separate.disp"])),
                              mean(t(res_df["separate.gear"])),mean(t(res_df["separate.drat"])),
                              mean(t(res_df["multivariate1"])),mean(t(res_df["multivariate2"])),
                              mean(t(res_df["multivariate3"]))))
```

```
##             model      mean.res
## 1     separate-hp  0.0013664005
## 2     separate-wt  0.0034481127
## 3     separate-vs  0.0031310298
## 4     separate-am -0.0033293583
## 5   separate-crab  0.0013572172
## 6  separate-lphkm -0.0027466534
## 7    separate-cyl -0.0026060053
## 8   separate-disp  0.0017533601
## 9   separate-gear  0.0026181530
## 10  separate-drat  0.0042333519
## 11  multivariate1  0.0013589701
## 12  multivariate2  0.0005543766
## 13  multivariate3  0.0010700774
```

```r
plot(1:32,t(res_df["separate.hp"]),type="p",pch="o",col="red",
     lty=1,ylab="residual",xlab="qsec",ylim=c(-5,7))
lines(1:32,t(res_df["separate.wt"]), type ="o",lty=2,col="blue",pch="*")
lines(1:32,t(res_df["separate.vs"]), type ="o",lty=3,col="dark green",pch="+")
lines(1:32,t(res_df["separate.am"]), type ="o",lty=4,col="yellow",pch="o")
lines(1:32,t(res_df["separate.carb"]), type ="o",lty=5,col="orange",pch="+")
lines(1:32,t(res_df["separate.lphkm"]), type ="o",lty=6,col="deeppink",pch="*")
lines(1:32,t(res_df["separate.cyl"]), type ="o",lty=7,col="gray",pch="+")
lines(1:32,t(res_df["separate.disp"]), type ="o",lty=8,col="hotpink",pch="o")
lines(1:32,t(res_df["separate.gear"]), type ="o",lty=9,col="lightgoldenrod",pch="+")
lines(1:32,t(res_df["separate.drat"]), type ="o",lty=10,col="olivedrab",pch="o")
lines(1:32,t(res_df["multivariate1"]), type ="o",lty=11,col="purple",pch="*")
lines(1:32,t(res_df["multivariate2"]), type ="o",lty=12,col="chocolate",pch="+")
lines(1:32,t(res_df["multivariate3"]), type ="o",lty=13,col="cyan",pch="o")
lines(1:32,rep(0,32), type ="l",pch="-",col="grey")
legend("topright",legend=c("sep.hp", "sep.wt", "sep.vs",
                           "sep.am","sep.carb","sep.lphkm", "sep.cyl", "sep.disp","sep.gear","sep.drat"
       col=c("red", "blue", "dark green","yellow","orange","deeppink","gray","hotpink",
             "lightgoldenrod","olivedrab","purple","chocolate","cyan"),
       pch=c("o","*","+","o","+","*","+","o","+","o","*","+","o"),
       lty=c(1,2,3,4,5,6,7,8,9,10,11,12,13))
```



As we can see, the mean residuals of different models are rather close to 0. If we look at each residual value, we can find that some of them are quiet large and can approximately reach 4. Different models share the same residual pattern.

In addition, single-variate models has mean residuals of the order of 10^{-3}, while multivariate models achieve better results than it. One thing worth noting is that multivariate model 1 this time is better than multivariate model 2. The possible explanation is that a linear relationship may not be entirely suitable for describing our target parameters. Some correlations can help improve model performance.

Statistical inference

Use multivariate model to check human readable real world statistics dependency.

```
sds <- sapply(car_properties[c("wt", "hp","vs","am","carb", "qsec")], sd)

beta_smry <- summary(fit_nlin1, pars = c("beta_wt", "beta_hp",
                                         "beta_vs","beta_am","beta_carb"))$summary
orig_smry <- beta_smry[, c(1, 4:8)]
orig_smry[1,] <- orig_smry[1,]/sds["wt"] * sds["qsec"]
orig_smry[2,] <- orig_smry[2,]/sds["hp"] * sds["qsec"]
orig_smry[3,] <- orig_smry[3,]/sds["vs"] * sds["qsec"]
orig_smry[4,] <- orig_smry[4,]/sds["am"] * sds["qsec"]
orig_smry[5,] <- orig_smry[5,]/sds["carb"] * sds["qsec"]
knitr::kable(orig_smry, digits = 3)
```

	mean	2.5%	25%	50%	75%	97.5%
beta_wt	1.758	0.360	1.298	1.764	2.225	3.100

	mean	2.5%	25%	50%	75%	97.5%
beta_hp	-0.015	-0.023	-0.018	-0.015	-0.012	-0.006
beta_vs	1.943	1.053	1.650	1.954	2.238	2.815
beta_am	-0.540	-1.558	-0.870	-0.548	-0.206	0.490
beta_carb	-0.111	-0.413	-0.215	-0.108	-0.009	0.197

```
sds <- sapply(car_properties[c("disp", "wt","vs","carb", "qsec")], sd)

beta_smry <- summary(fit_nlin2, pars = c("beta_disp",
                                         "beta_wt","beta_vs","beta_carb"))$summary
orig_smry <- beta_smry[, c(1, 4:8)]
orig_smry[1,] <- orig_smry[1,]/sds["disp"] * sds["qsec"]
orig_smry[2,] <- orig_smry[2,]/sds["wt"] * sds["qsec"]
orig_smry[3,] <- orig_smry[3,]/sds["vs"] * sds["qsec"]
orig_smry[4,] <- orig_smry[4,]/sds["carb"] * sds["qsec"]
knitr::kable(orig_smry, digits = 3)
```

	mean	2.5%	25%	50%	75%	97.5%
beta_disp	-0.730	-1.133	-0.860	-0.728	-0.603	-0.332
beta_wt	4.093	2.542	3.580	4.098	4.603	5.663
beta_vs	1.485	0.530	1.164	1.480	1.800	2.476
beta_carb	-0.580	-0.806	-0.657	-0.581	-0.504	-0.341

```
sds <- sapply(car_properties[c("lphkm", "cyl","disp","hp","drat",
                               "wt", "vs", "am", "gear", "carb","qsec")], sd)

beta_smry <- summary(fit_nlin3, pars = c("beta_lphkm", "beta_cyl","beta_disp",
                                         "beta_hp","beta_drat","beta_wt",
                                         "beta_vs","beta_am","beta_gear","beta_carb"))$summary
orig_smry <- beta_smry[, c(1, 4:8)]
orig_smry[1,] <- orig_smry[1,]/sds["lphkm"] * sds["qsec"]
orig_smry[2,] <- orig_smry[2,]/sds["cyl"] * sds["qsec"]
orig_smry[3,] <- orig_smry[3,]/sds["disp"] * sds["qsec"]
orig_smry[4,] <- orig_smry[4,]/sds["hp"] * sds["qsec"]
orig_smry[5,] <- orig_smry[5,]/sds["drat"] * sds["qsec"]
orig_smry[6,] <- orig_smry[6,]/sds["wt"] * sds["qsec"]
orig_smry[7,] <- orig_smry[7,]/sds["vs"] * sds["qsec"]
orig_smry[8,] <- orig_smry[8,]/sds["am"] * sds["qsec"]
orig_smry[9,] <- orig_smry[9,]/sds["gear"] * sds["qsec"]
orig_smry[10,] <- orig_smry[10,]/sds["carb"] * sds["qsec"]
knitr::kable(orig_smry, digits = 3)
```

	mean	2.5%	25%	50%	75%	97.5%
beta_lphkm	-0.052	-0.251	-0.122	-0.055	0.015	0.157
beta_cyl	-0.408	-0.991	-0.611	-0.410	-0.210	0.183
beta_disp	-0.372	-1.009	-0.585	-0.378	-0.157	0.277
beta_hp	-0.003	-0.016	-0.008	-0.003	0.001	0.010
beta_drat	-0.096	-1.114	-0.434	-0.097	0.230	0.946
beta_wt	2.993	0.594	2.233	2.999	3.767	5.265
beta_vs	1.059	-0.180	0.658	1.056	1.457	2.253

	mean	2.5%	25%	50%	75%	97.5%
beta_am	-0.744	-1.984	-1.171	-0.742	-0.340	0.530
beta_gear	-0.232	-1.155	-0.542	-0.224	0.077	0.707
beta_carb	-0.261	-0.756	-0.423	-0.261	-0.095	0.237

For the regression coefficients, some are positive with a rather large impact but still some are negative with smaller impact.Based on these three results, we would say that shape of the engine(straight vs V-shaped) and the weight parameter give much more support for cars traversing a quarter mile.

Sensitive analysis

Let's check how much the results would change if we used a different set of priors.The examine is done with default noninformative priors of Stan.

```
stan_separate_model_de = '
data {
  int<lower=0> N;
  vector[N] x;
  vector[N] y;
}
parameters {
  real alpha;
  real beta;
  real<lower=0> sigma;
}
transformed parameters{
  vector[N] mu;
  mu = alpha + beta*x;
}
model {
  y ~ normal(mu, sigma);
}
// Log likelihoods genereated for LOO
generated quantities {
  vector[N] log_lik;
  for (i in 1:N)
    log_lik[i] = normal_lpdf(y[i] |alpha+x[i]*beta , sigma);
}
'

fit_hpd = stan(data = list(N=length(scaled_car_properties$qsec),
                   x=c(scaled_car_properties$hp),
                   y=c(scaled_car_properties$qsec)),
                    model_code = stan_separate_model_de,refresh=0)
```

```
## Warning: There were 1 transitions after warmup that exceeded the maximum treedepth. Increase max_tre
## http://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded
```

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

```
fit_wtd = stan(data = list(N=length(scaled_car_properties$qsec),
                   x=c(scaled_car_properties$wt),
                   y=c(scaled_car_properties$qsec)),
                    model_code = stan_separate_model_de,refresh=0)
```

```
fit_vsd = stan(data = list(N=length(scaled_car_properties$qsec),
                   x=c(scaled_car_properties$vs),
                   y=c(scaled_car_properties$qsec)),
                    model_code = stan_separate_model_de,refresh=0)

fit_amd = stan(data = list(N=length(scaled_car_properties$qsec),
                   x=c(scaled_car_properties$am),
                   y=c(scaled_car_properties$qsec)),
                    model_code = stan_separate_model_de,refresh=0)

fit_carbd = stan(data = list(N=length(scaled_car_properties$qsec),
                   x=c(scaled_car_properties$carb),
                   y=c(scaled_car_properties$qsec)),
                    model_code = stan_separate_model_de,refresh=0)

fit_lphkmd = stan(data = list(N=length(scaled_car_properties$qsec),
                   x=c(scaled_car_properties$lphkm),
                   y=c(scaled_car_properties$qsec)),
                    model_code = stan_separate_model_de,refresh=0)

fit_cyld = stan(data = list(N=length(scaled_car_properties$qsec),
                   x=c(scaled_car_properties$cyl),
                   y=c(scaled_car_properties$qsec)),
                    model_code = stan_separate_model_de,refresh=0)

fit_dispd = stan(data = list(N=length(scaled_car_properties$qsec),
                   x=c(scaled_car_properties$disp),
                   y=c(scaled_car_properties$qsec)),
                    model_code = stan_separate_model_de,refresh=0)

fit_geard = stan(data = list(N=length(scaled_car_properties$qsec),
                   x=c(scaled_car_properties$gear),
                   y=c(scaled_car_properties$qsec)),
                    model_code = stan_separate_model_de,refresh=0)

fit_dratd = stan(data = list(N=length(scaled_car_properties$qsec),
                   x=c(scaled_car_properties$drat),
                   y=c(scaled_car_properties$qsec)),
                    model_code = stan_separate_model_de,refresh=0)

stan_nlind_model1 = '
data {
  int<lower=0> n;
  vector[n] hp;
  vector[n] wt;
  vector[n] vs;
  vector[n] am;
  vector[n] carb;
  vector[n] qsec;
}
parameters {
  real alpha;
  real beta_hp;
```

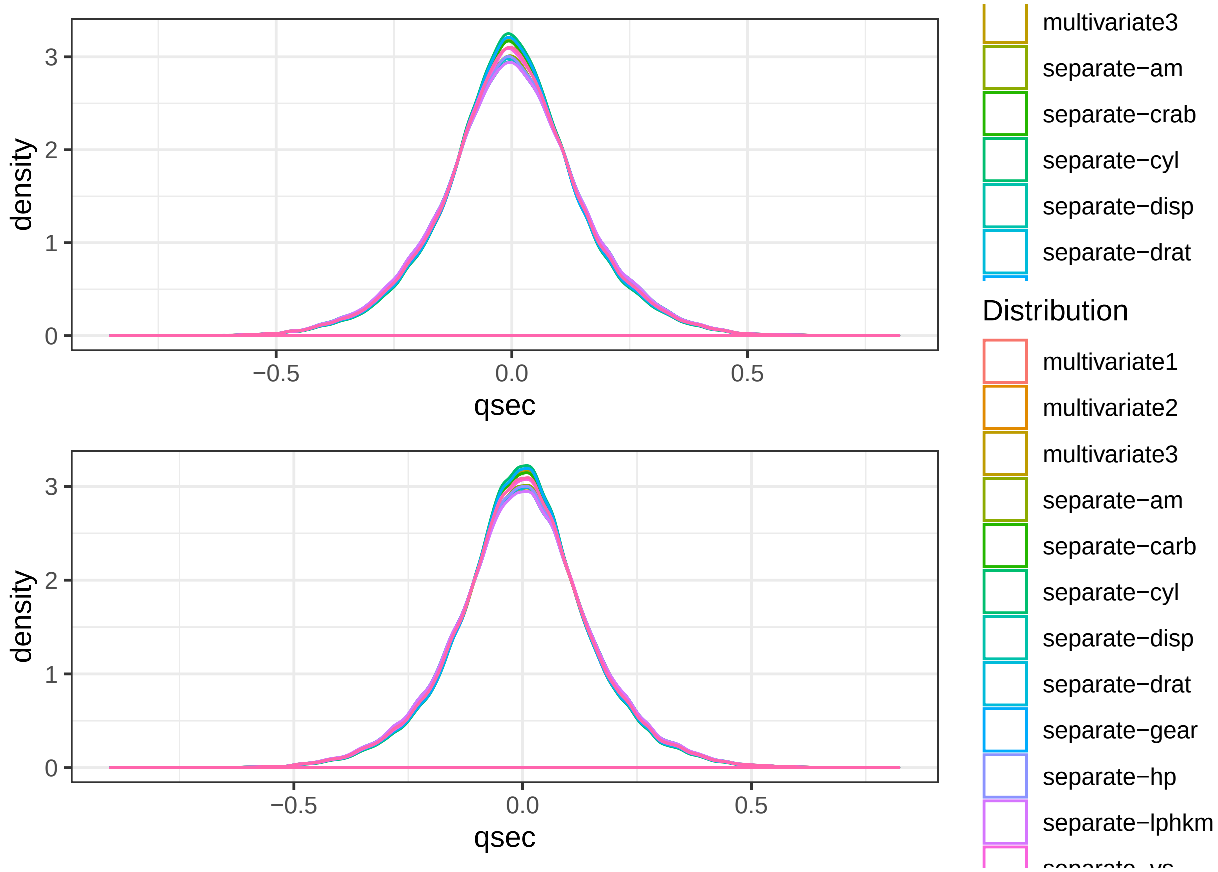
```
  real beta_wt;
  real beta_vs;
  real beta_am;
  real beta_carb;
  real<lower=0> sigma;
}
transformed parameters{
  vector[n] mu;
  mu = alpha + beta_hp*hp + beta_wt*wt + beta_vs*vs +
       beta_am*am + beta_carb*carb;
}
model {
  qsec ~ normal(mu, sigma);
}
// Log likelihoods genereated for LOO
generated quantities {
  vector[n] log_lik;
  for (i in 1:n)
    log_lik[i] = normal_lpdf(qsec[i] |mu[i] , sigma);
}
'

fit_nlind1 = stan(data = list(n=length(scaled_car_properties$hp),
                 hp=c(scaled_car_properties$hp),
                 wt=c(scaled_car_properties$wt),
                 vs=c(scaled_car_properties$vs),
                 am=c(scaled_car_properties$am),
                 carb=c(scaled_car_properties$carb),
                 qsec=c(scaled_car_properties$qsec)),
                  model_code = stan_nlind_model1,refresh=0)

stan_nlind_model2 = '
data {
  int<lower=0> n;
  vector[n] disp;
  vector[n] wt;
  vector[n] vs;
  vector[n] carb;
  vector[n] qsec;
}
parameters {
  real alpha;
  real beta_disp;
  real beta_wt;
  real beta_vs;
  real beta_carb;
  real<lower=0> sigma;
}
transformed parameters{
  vector[n] mu;
  mu = alpha + beta_disp*disp + beta_wt*wt + beta_vs*vs + beta_carb*carb;
}
model {
```

```
  qsec ~ normal(mu, sigma);
}
// Log likelihoods genereated for LOO
generated quantities {
  vector[n] log_lik;
  for (i in 1:n)
    log_lik[i] = normal_lpdf(qsec[i] |mu[i] , sigma);
}
'

fit_nlind2 = stan(data = list(n=length(scaled_car_properties$hp),
                  disp=c(scaled_car_properties$disp),
                  wt=c(scaled_car_properties$wt),
                  vs=c(scaled_car_properties$vs),
                  carb=c(scaled_car_properties$carb),
                  qsec=c(scaled_car_properties$qsec)),
                   model_code = stan_nlind_model2,refresh=0)

stan_nlind_model3 = '
data {
  int<lower=0> n;
  vector[n] lphkm;
  vector[n] cyl;
  vector[n] disp;
  vector[n] hp;
  vector[n] drat;
  vector[n] wt;
  vector[n] vs;
  vector[n] am;
  vector[n] gear;
  vector[n] carb;
  vector[n] qsec;
}
parameters {
  real alpha;
  real beta_lphkm;
  real beta_cyl;
  real beta_disp;
  real beta_hp;
  real beta_drat;
  real beta_wt;
  real beta_vs;
  real beta_am;
  real beta_gear;
  real beta_carb;
  real<lower=0> sigma;
}
transformed parameters{
  vector[n] mu;
  mu = alpha + beta_lphkm*lphkm + beta_cyl* cyl +
  beta_disp * disp + beta_hp*hp + beta_drat* drat + beta_wt*wt +
  beta_vs*vs + beta_am*am + beta_gear*gear + beta_carb*carb;
}
```

```
model {
  qsec ~ normal(mu, sigma);
}
// Log likelihoods genereated for LOO
generated quantities {
  vector[n] log_lik;
  for (i in 1:n)
    log_lik[i] = normal_lpdf(qsec[i] |mu[i] , sigma);
}
'

fit_nlind3 = stan(data = list(n=length(scaled_car_properties$hp),
                              lphkm=c(scaled_car_properties$lphkm),
                              cyl=c(scaled_car_properties$cyl),
                              disp=c(scaled_car_properties$disp),
                              hp=c(scaled_car_properties$hp),
                              drat=c(scaled_car_properties$drat),
                              wt=c(scaled_car_properties$wt),
                              vs=c(scaled_car_properties$vs),
                              am=c(scaled_car_properties$am),
                              gear=c(scaled_car_properties$gear),
                              carb=c(scaled_car_properties$carb),
                              qsec=c(scaled_car_properties$qsec)),
                 model_code = stan_nlind_model3,refresh=0)
```

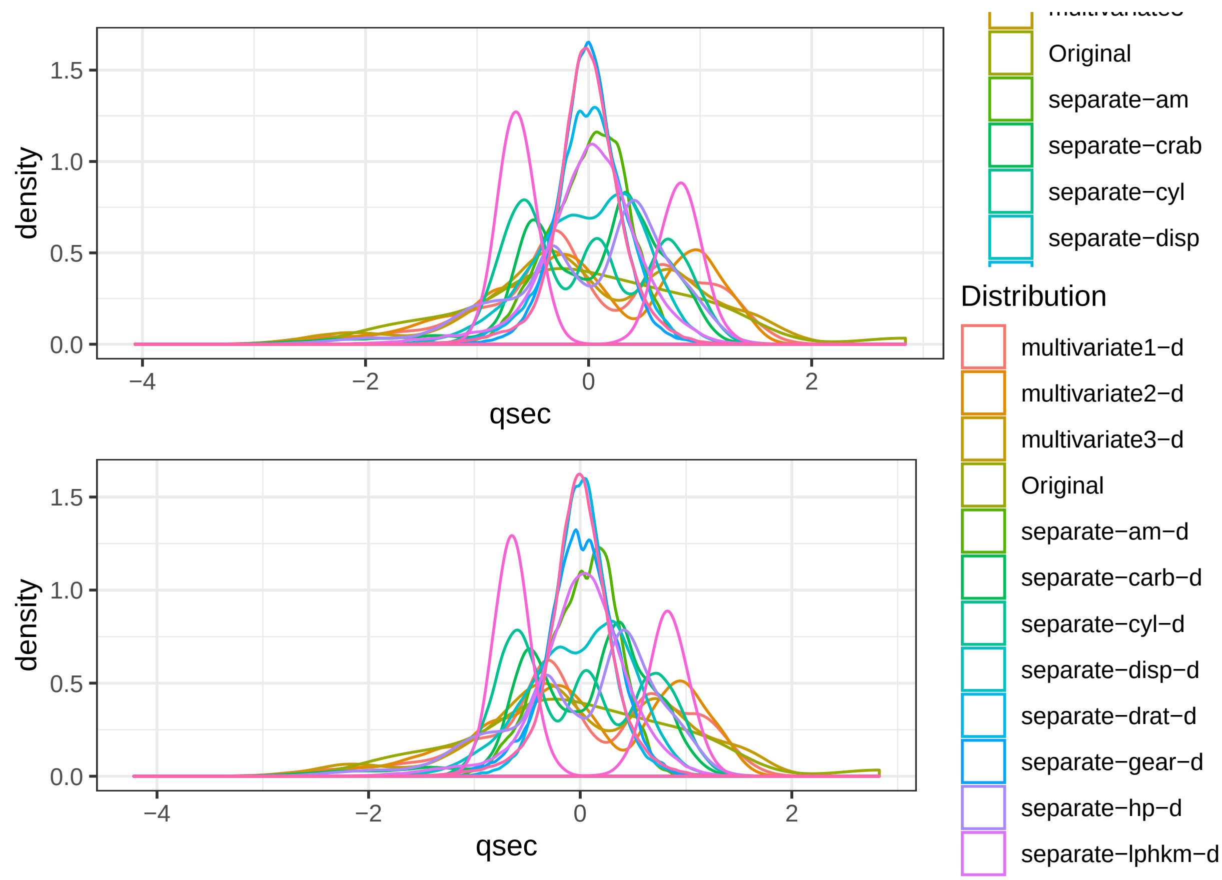
```
p1<-ggplot(plot_dfa, aes(qsec, color = Distribution)) +
  geom_density() +
  #scale_color_brewer(palette = "Set1") +
  theme_bw()
plot_dfad <- data.frame(qsec = c(c(extract(fit_hpd, pars = 'alpha', permuted = TRUE)$alpha),
                              c(extract(fit_wtd, pars = 'alpha', permuted = TRUE)$alpha),
                              c(extract(fit_vsd, pars = 'alpha', permuted = TRUE)$alpha),
                              c(extract(fit_amd, pars = 'alpha', permuted = TRUE)$alpha),
                              c(extract(fit_carbd, pars = 'alpha', permuted = TRUE)$alpha),
                              c(extract(fit_lphkmd, pars = 'alpha', permuted = TRUE)$alpha),
                              c(extract(fit_cyld, pars = 'alpha', permuted = TRUE)$alpha),
                              c(extract(fit_dispd, pars = 'alpha', permuted = TRUE)$alpha),
                              c(extract(fit_geard, pars = 'alpha', permuted = TRUE)$alpha),
                              c(extract(fit_dratd, pars = 'alpha', permuted = TRUE)$alpha),
                              c(extract(fit_nlind1, pars = 'alpha', permuted = TRUE)$alpha),
                              c(extract(fit_nlind2, pars = 'alpha', permuted = TRUE)$alpha),
                              c(extract(fit_nlind3, pars = 'alpha', permuted = TRUE)$alpha)),
                      Distribution = rep(c("separate-hp","separate-wt",
                                           "separate-vs",
                                          "separate-am","separate-carb",
                                          "separate-lphkm",
                                          "separate-cyl","separate-disp",
                                          "separate-gear","separate-drat"
                                           ,"multivariate1","multivariate2",
                                          "multivariate3"),
                                         times=c(128000,128000,128000,128000,128000,
                                                 128000,128000,128000,128000,128000,
                                                 128000,128000,128000)))
p2<-ggplot(plot_dfad, aes(qsec, color = Distribution)) +
```

```r
  geom_density() +
  #scale_color_brewer(palette = "Set1") +
  theme_bw()
grid.arrange(p1, p2)
```




```r
plot_dfd <- data.frame(qsec = c(c(extract(fit_hpd, pars = 'mu', permuted = TRUE)$mu),
                              c(extract(fit_wtd, pars = 'mu', permuted = TRUE)$mu),
                              c(extract(fit_vsd, pars = 'mu', permuted = TRUE)$mu),
                              c(extract(fit_amd, pars = 'mu', permuted = TRUE)$mu),
                              c(extract(fit_carbd, pars = 'mu', permuted = TRUE)$mu),
                              c(extract(fit_lphkmd, pars = 'mu', permuted = TRUE)$mu),
                              c(extract(fit_cyld, pars = 'mu', permuted = TRUE)$mu),
                              c(extract(fit_dispd, pars = 'mu', permuted = TRUE)$mu),
                              c(extract(fit_geard, pars = 'mu', permuted = TRUE)$mu),
                              c(extract(fit_dratd, pars = 'mu', permuted = TRUE)$mu),
                              c(extract(fit_nlind1, pars = 'mu', permuted = TRUE)$mu),
                              c(extract(fit_nlind2, pars = 'mu', permuted = TRUE)$mu),
                              c(extract(fit_nlind3, pars = 'mu', permuted = TRUE)$mu),
                                 scaled_car_properties$qsec),
                     Distribution = rep(c("separate-hp-d","separate-wt-d",
                                          "separate-vs-d",
                                          "separate-am-d","separate-carb-d",
                                          "separate-lphkm-d",
                                          "separate-cyl-d","separate-disp-d",
                                          "separate-gear-d",
                                          "separate-drat-d","multivariate1-d",
```

```
                                                  "multivariate2-d",
                                                  "multivariate3-d", "Original"),
                                                times=c(128000,128000,128000,128000,128000,
                                                        128000,128000,128000,128000,128000,
                                                        128000,128000,128000, nrow(scaled_car_properties))))
p1<-ggplot(plot_df, aes(qsec, color = Distribution)) +
  geom_density() +
  #scale_color_brewer(palette = "Set1") +
  theme_bw()
p2<-ggplot(plot_dfd, aes(qsec, color = Distribution)) +
  geom_density() +
  #scale_color_brewer(palette = "Set1") +
  theme_bw()
grid.arrange(p1, p2)
```



Even though the priors have changed a lot, the distributions look very similar. We conclude that the posterior distribution is mainly determined by the data, and the effect of the priors is not obvious.

Conclusion

We made a experiment based a dataset called Motor Trend Car Road Testson and studied how car design can affect the 1/4 mile time, in other words, the performance of the car. We proposed three different mutltivariate models to check the affect of common design parameter. Our model produces a credible posterior predictive distribution and is deemed usable as such. We cannot give a general conclusion that how to make better varivale selection but we can give a conclusion that the shape of the engine(straight vs V-shaped) and the weight parameter give much more support for cars traversing a quarter mile.

However, regarding the residuals, we found kind of pattern. For the future work, it is better to consider the non-linear relationshp.

Acknowledge

At first our main inspiration to explore linear regression model using *mtcars* dataset stemmed from [1], but our following work are all done by ourselves.

Reference

[1] Anton Mattsson, bdacars, (2018), GitHub repository, https://github.com/antonvsdata/bdacars