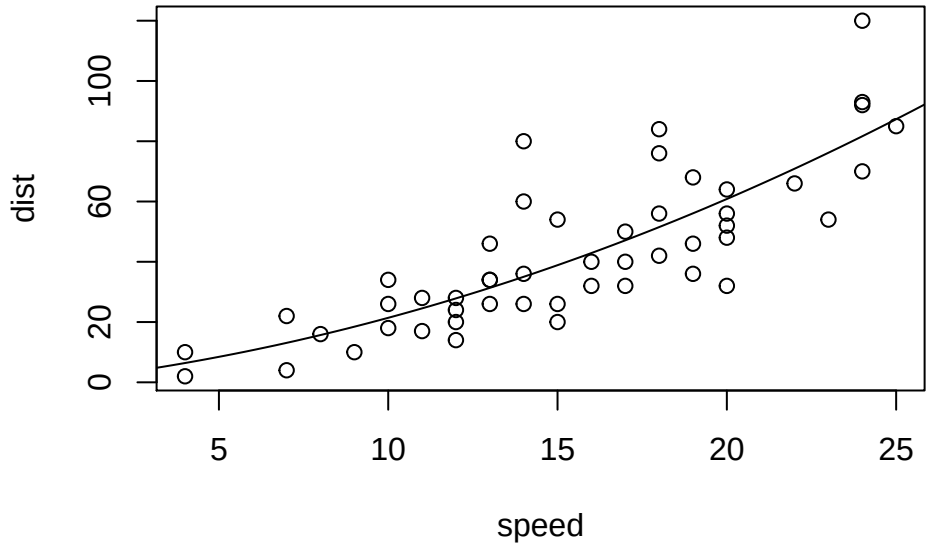
Project 2: Regression Models in JAGS

Dongrui Shen

Background and models

The `cars` data are from an experiment on the stopping distances of cars from different initial speeds. A linear model $E(\texttt{dist}_i) = \mu_i = \beta_1 \texttt{speed}_i + \beta_2 \texttt{speed}_i^2$ is a plausible fit to the data, as illustrated here.



However, notice that the variance in the data may be increasing with the mean. To address this, 3 alternative models for the data will be compared using Bayesian methods. The models are

1. $\texttt{dist}_i \sim N(\mu_i, \sigma^2)$, priors $\beta_1 \sim U(0, 2.5)$, $\beta_2 \sim N(0, 100)$ and $\log(1/\sigma^2) \sim N(0, 100)$.
2. $\texttt{dist}_i \sim N(\mu_i, (\beta_3 + \beta_4 \texttt{speed}_i)^2)$, priors as above for β_1 and β_2. $\beta_3 \sim U(0, 20)$ and $\beta_4 \sim U(0, 2)$.
3. $\texttt{dist}_i \sim \text{Gamma}(\mu_i/\phi, \phi)$ where μ_i/ϕ is the shape parameter and ϕ the scale parameter. Priors $\phi \sim U(0, 10)$, $\beta_1 \sim U(0, 2.5)$, $\log \beta_2 \sim N(-2, 100)$.

Gibbs sampling was used for inference as implemented in JAGS, which was here used via package `rjags`. Note that in JAGS normal (and log-normal) densities are parameterized in terms of mean and precision, $\tau = 1/\sigma^2$, while gamma densities are parameterized in terms of shape and rate $= 1/\phi$.

JAGS model specifications.

The following codes are used to implement the three models (See the corresponding text files in "/scripts/").

```
# Model 1: a constant variance normal model
model {
  for (i in 1:N) {  ## loop over observations
    # statistical model
    mu[i] <- beta1*x[i] + beta2*(x[i]^2)
    y[i] ~ dnorm(mu[i],tau) ## yi ~ N(mui,1/tau), where tau is the precision
  }
```

```
  # priors
  beta1 ~ dunif(0,2.5)
  beta2 ~ dnorm(0,.01) ## beta2 ~ N(0,100)
  tau ~ dlnorm(0,.01)  ## log(tau) ~ N(0,100)
}
```

```
# Model 2: a normal model in which the standard deviation increases with speed
model {
  for (i in 1:N) {  ## loop over observations
    # statistical model
    mu[i] <- beta1*x[i] + beta2*(x[i]^2)
    y[i] ~ dnorm(mu[i],1/(beta3 + beta4*x[i])^2) ## yi ~ N(mui,(beta3+beta4*xi)^2)
  }
  # priors
  beta1 ~ dunif(0,2.5)
  beta2 ~ dnorm(0,.01) ## beta2 ~ N(0,100)
  beta3 ~ dunif(0,20)
  beta4 ~ dunif(0,2)
}
```

```
# Model 3: a gamma model
model {
  for (i in 1:N) {  ## loop over observations
    # statistical model
    mu[i] <- beta1*x[i] + beta2*(x[i]^2)
    y[i] ~ dgamma(mu[i]/phi,1/phi) ## yi ~ gamma(mui/phi,phi), where phi is the scale
  }
  # priors
  phi ~ dunif(0,10)
  beta1 ~ dunif(0,2.5)
  beta2 ~ dlnorm(-2,.01)  ## log(beta2) ~ N(-2,100)
}
```

The JAGS files are compiled into a Gibbs sampler via a call to the function `jags.model`. Note that we take two parallel chains for each parameter since it is useful for judging convergences and computing DIC to compare models.

```
library(rjags)  ## include the package

# create JAGS models
# take 2 parallel chains for each parameter by setting n.chains=2
mod1 <- jags.model("./scripts/model1.jags",
                   data=list(x=cars$speed,y=cars$dist,N=nrow(cars)), n.chains=2, quiet=TRUE)
mod2 <- jags.model("./scripts/model2.jags",
                   data=list(x=cars$speed,y=cars$dist,N=nrow(cars)), n.chains=2, quiet=TRUE)
mod3 <- jags.model("./scripts/model3.jags",
                   data=list(x=cars$speed,y=cars$dist,N=nrow(cars)), n.chains=2, quiet=TRUE)
```

Simulation and checking

First, use `coda.samples` to run simulation. It is a wrapper function for `jags.samples`. Unlike `jags.samples`, its output is one single `mcmc.list` object.

```
sam1.coda <- coda.samples(mod1, c("beta1","beta2","tau"), n.iter=50000, thin=10)
sam2.coda <- coda.samples(mod2, c("beta1","beta2","beta3","beta4"), n.iter=40000, thin=5)
sam3.coda <- coda.samples(mod3, c("phi","beta1","beta2"), n.iter=50000, thin=10)
```

Then, we use other functions in the `coda` package to check the convergence and mixing of the chains, including:

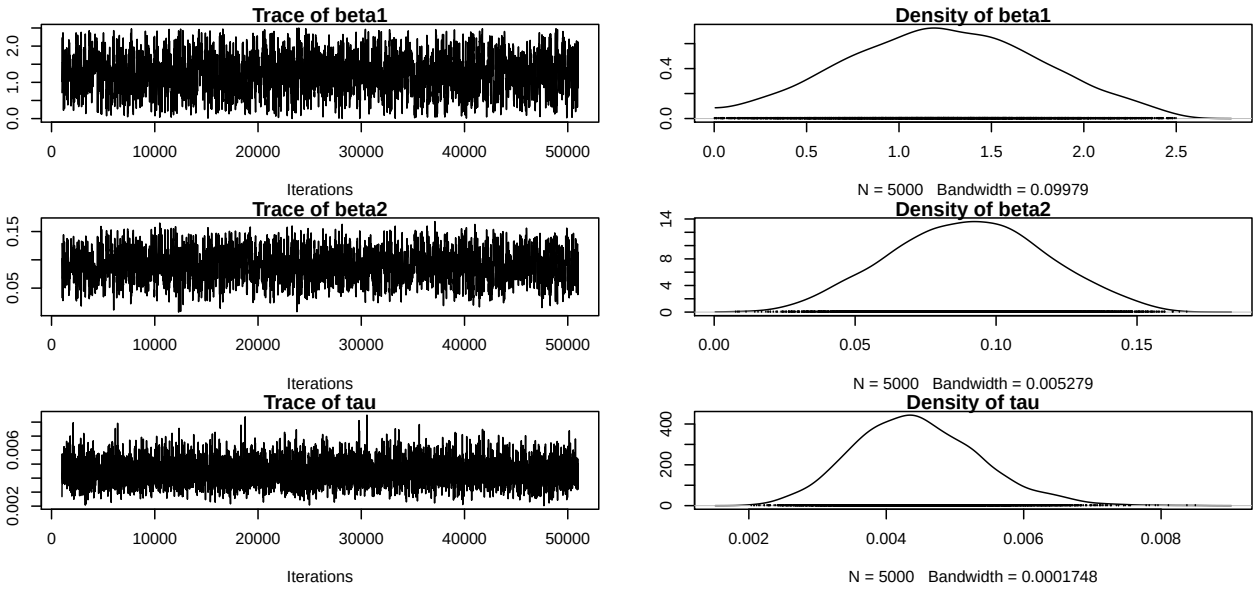
- `plot.mcmc(sample, trace = TRUE, density = TRUE, ...)`: plot the chains of the sampled output and the kernel density estimates (see details in `?density`) for the posterior distributions.
- `effectiveSize(sample)`: to get the effective sample sizes of the sampled output. Here, we consider effective sample sizes of over 1000 for each parameter to be sensible.

Besides, we use `acfplot`, `crosscorr` to check the auto-correlations and cross-correlations. We also compare the highest posterior density intervals with the 95% credible intervals with `HPDinterval` and `quantile` (see details in `/scripts/simulate.r`).

```
# sample code: check convergence and mixing
par(mfrow=c(3,2), mar=c(4,4,1,1))
plot(sam1.coda[[1]])  ## plot chains and density estimates of chain 1
plot(sam1.coda[[2]])  ## plot chains and density estimates of chain 2
effectiveSize(sam1.coda[[1]]) ## print effective sample size of chain 1
effectiveSize(sam1.coda[[2]]) ## print effective sample size of chain 2

# the two chains should behave similar so we only display checking results for chain 1
```

Checking results for Model 1



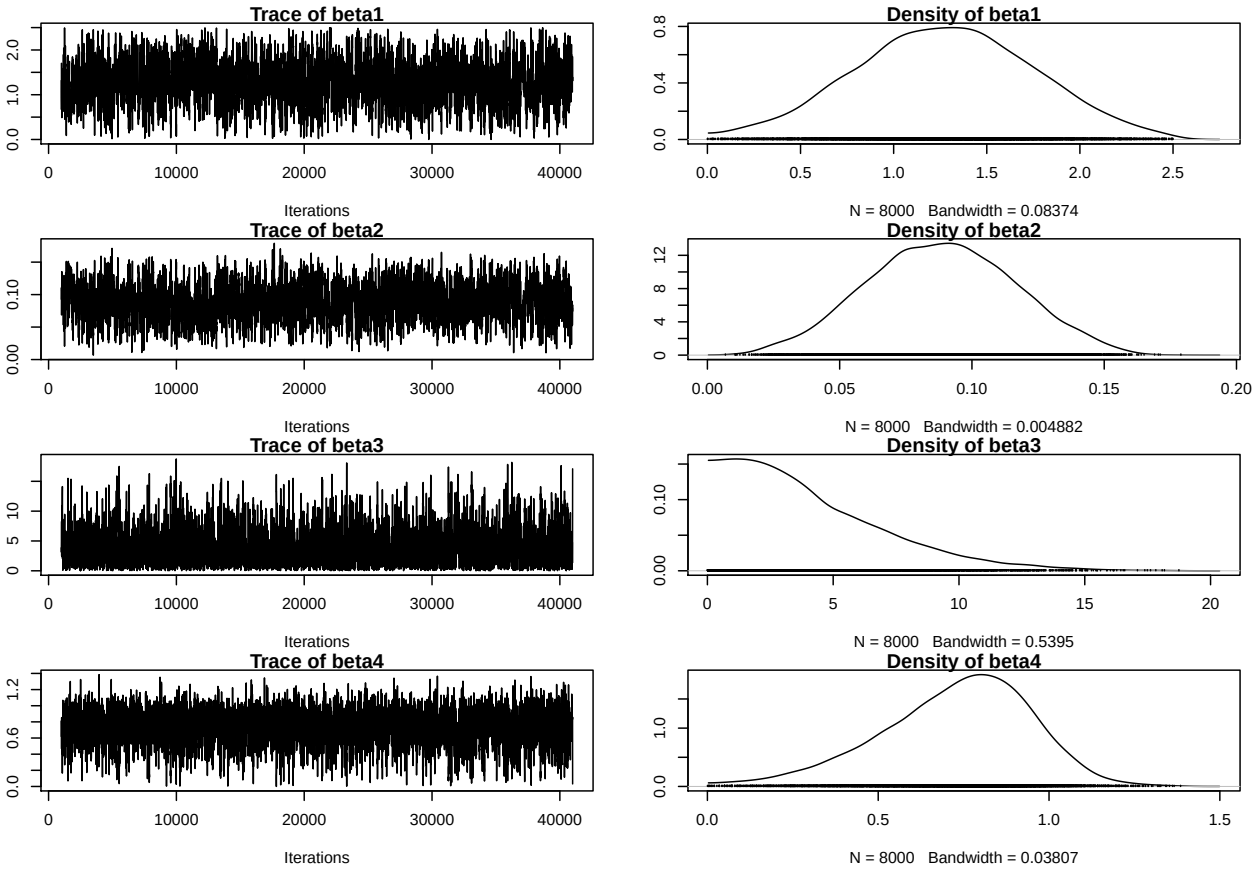

```
##    beta1    beta2      tau
## 1360.051 1393.169 4715.683
```

Here we generate two parallel chains with 50000 samples each and save every 10th sample. In the chain convergence plots for each parameters, it shows that the chains have a very short burn-in period (take 1000 iterations for adaptation). Since we take 2 parallel chains for each parameter, after comparing their trace plots and posterior density estimate plots, we can observe they both have a high amount of fluctuation (good mixing) and no real patterns in the graphs indicative of low auto-correlation. The effective sample sizes for each parameter are over 1000. Especially, the effective size for τ is close to 5000, indicating high independence between samples.

In the posterior density estimate plots, the prior and posterior densities of $\beta_1, beta_2, \tau$ are quite different from each other. To be exact, $\beta_1 \sim U(0, 2.5)$, $\beta_2 \sim U(0, 100)$ and $\log(\tau) \sim U(0, 100)$. However, their posterior

densities show a similar pattern to a normal distribution with a small standard deviation respectively. Thus, the priors for Model 1 are sensible since observing the data updated our beliefs about the parameters.

Checking results for Model 2

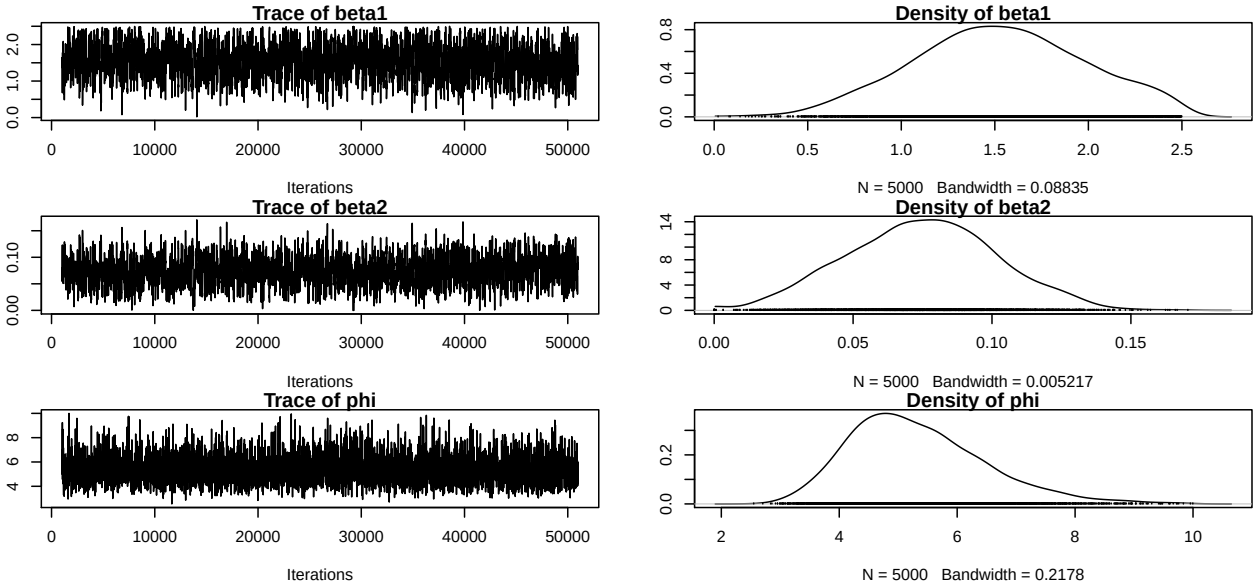



```
##    beta1    beta2    beta3    beta4
## 1314.147 1352.420 2085.280 2249.049
```

The chain plots for Model 2 show a good mixing and convergence as Model 1. Through the auto-correlation checking, we find the acf plots converges to 0 around lag < 10. Thereafter, we only need 40000 samples each and save every 5th sample to achieve effective sample sizes of over 1000.

In the posterior density estimate plots, for β_i, where $i = 1, 2, 4$, we can observe obvious differences from their prior densities. For β_3, its posterior shows a monotonically decreasing pattern, which is somewhat not that different from its prior $\beta_3 \sim U(0, 20)$, indicating the choice of the prior for β_3 might be improper. The highest posterior density intervals are a little different from the 95% credible intervals except for β_3, which is resulted from its skewed posterior distribution.

Checking results for Model 3




```
##    beta1    beta2      phi
## 1304.109 1322.647 4731.292
```

We take 50000 iterations and save every 10th sample to get effective sizes of over 1000. The chain plots for Model 3 also show a good mixing and convergence. However, the simulation process is much slower when compared with the other two models. We compare the posterior means of each model to the linear regression coefficients and found the posterior means of Model 3 is more different from the linear regression coefficients.

For β_1, β_2 and ϕ, their posteriors are very different from priors, which is what we expect to see.

Other checks We checked the two chains both and found their summary plots are very similar, indicating good convergence and mixing. We also checked acf plots, basically, chains for parameters excluding β_1 and β_2 would have lower auto-correlation. In the cross-correlation checking for all three models, we found β_1 and β_2 are highly correlated.

Model comparison

Deviance information criterion Define the deviance as $D(\theta) = -2\log(p(y|\theta)) + \text{const}$, where θ are unknown parameters of the model and $p(y|\theta)$ is the likelihood. Then the deviance information criterion is calculated as

$$\text{DIC} = p_D + E(D(\theta)),$$

where p_D is the effective number of parameters. The larger the effective number of parameters is, the easier it is for the model to fit the data, and so the deviance needs to be penalized.

The `dic.samples` generates penalized deviance statistics to compare models. Basically, a low deviance means a good fit. The p_D term compensates for this effect by favoring models with a smaller number of parameters. As suggested in the following outputs, Model 3 has the lowest DIC, so it is possible that Model 3 is the best choice.

```
dic.samples(mod1, n.iter=50000, thin=10)
```

```
## Mean deviance:  413.7
## penalty 2.955
## Penalized deviance: 416.7
```

```
dic.samples(mod2, n.iter=40000, thin=5)
```

```
## Mean deviance:  406.9
## penalty 3.988
## Penalized deviance: 410.9
```

```
dic.samples(mod3, n.iter=50000, thin=10)
```

```
## Mean deviance:  397.2
## penalty 3.14
## Penalized deviance: 400.3
```

Model fitting As a check that the choice is sensible, we take evenly spaced parameters from each model's posteriors to produce 1000 approximately independent curves (500 for each chain) of $E(\text{dist})$ as a function of speed.

The first figure below show that all the three models for sampling result in parameter estimates that perform well in fitting the `cars` data, while it is true that the variation in our estimates increases as speed increases. Model 2 and Model 3 show a higher variability compared with Model 1. However, the fitting performances for these two models are very close to each other. In some simulations, Model 3 could overlay a little bit more data points, but sometimes it does not.Nevertheless, considering that the Model 3 is less affected by the prior guesstimates when compared with Model 2, in the case of lacking prior empirical information, we would prefer Model 3.

In addition, we use the the 2.5, 50 and 97.5 percentiles of the coefficients of each model to construct the predictions and then plot them against the observed points. Model 3 seems to have a narrower credible interval, implying a good fit (see details in "/scripts/simulate.r").