CODE

Introduction to Bayesian Regression Modeling in R using rstanarm

Nathan T. James

2019-09-27

Introduction

The Bayesian paradigm has become increasingly popular, but is still not as widespread as classical statistical methods (e.g.Â maximum likelihood estimation, null hypothesis significance testing, etc.). One reason for this disparity is the somewhat steep learning curve for Bayesian statistical software. The `rstanarm` package aims to address this gap by allowing `R` users to fit common Bayesian regression models using an interface very similar to standard functions R functions such as `lm()` and `glm()`. In this seminar we will provide an introduction to Bayesian inference and demonstrate how to fit several basic models using `rstanarm`.

Bayesian Inference

Bayesian inference provides a principled method of combining prior information concerning parameters (such as regression coefficients) with observed outcome and covariate data based on Bayes Theorem:

$$\underbrace{p(\theta|X,y)}_{\text{posterior distribution}} = \frac{\overbrace{p(\theta)}^{\text{prior}}\overbrace{p(y|X,\theta)}^{\text{likelihood}}}{\underbrace{\int p(\theta)p(y|X,\theta)\,d\theta}_{\text{marginal likelihood}}}$$

The likelihood (aka sampling distribution) $p(y|X,\theta)$ represents the contribution from the observed data and $p(\theta)$ represents prior information (or absence of information) about the parameters. The combination of these two components yields the posterior probability $p(\theta|y,X)$ which represents an updated distribution for the parameters conditional on the observed data. Note that if y is fixed the marginal likelihood is constant with respect to θ, so we can write the equation above in terms of the unnormalized posterior distribution: $p(\theta|X,y) \propto p(\theta)p(y|X,\theta)$. Performing inference for regression models in a Bayesian framework has several advantages:

- Can formally incorporate information from multiple sources including prior information if available
- Inference is conditional on *actual* observed data, rather than data that *could have* been observed
- Can answer inferential questions using interpretable posterior probability statements rather than hypothesis tests and p-values which are dominant in the frequentist paradigm (e.g., â€œGiven our model, prior knowledge, and the observed data, there is a 92% probability that the drug is effectiveâ€)
- All inference proceeds *directly* from the posterior distribution

Stan, rstan, and rstanarm

Stan (https://mc-stan.org/) is a general purpose probabilistic programming language for Bayesian statistical inference. It has interfaces for many popular data analysis languages including `Python`, `MATLAB`, `Julia`, and `Stata`. The `R` interface for `Stan` is called `rstan` and `rstanarm` is a front-end to `rstan` that allows regression models to be fit using a standard `R` regression model interface.

CODE

Steps for Bayesian inference

1. Specify the probability model. As discussed above the model has two parts, a likelihood for the observed data and a prior distribution. In practice, the form of the likelihood is usually based on the outcome type and is often the same as a frequentist analysis. Specifying the prior distribution can be more involved, but `rstanarm` includes default priors that work well in many cases.

2. Draw samples from the posterior distribution. Once the model is specified, we need to get an updated distribution of the parameters conditional on the observed data. Conceptually, this step is simple, but in practice it can be quite complicated. By default, `Stan` uses a method called Markov Chain Monte Carlo (MCMC) to get these samples.

3. Evaluate the model. There are several important checks that are necessary to ensure that there are no problems with the MCMC procedure used to get samples or the posterior distribution.

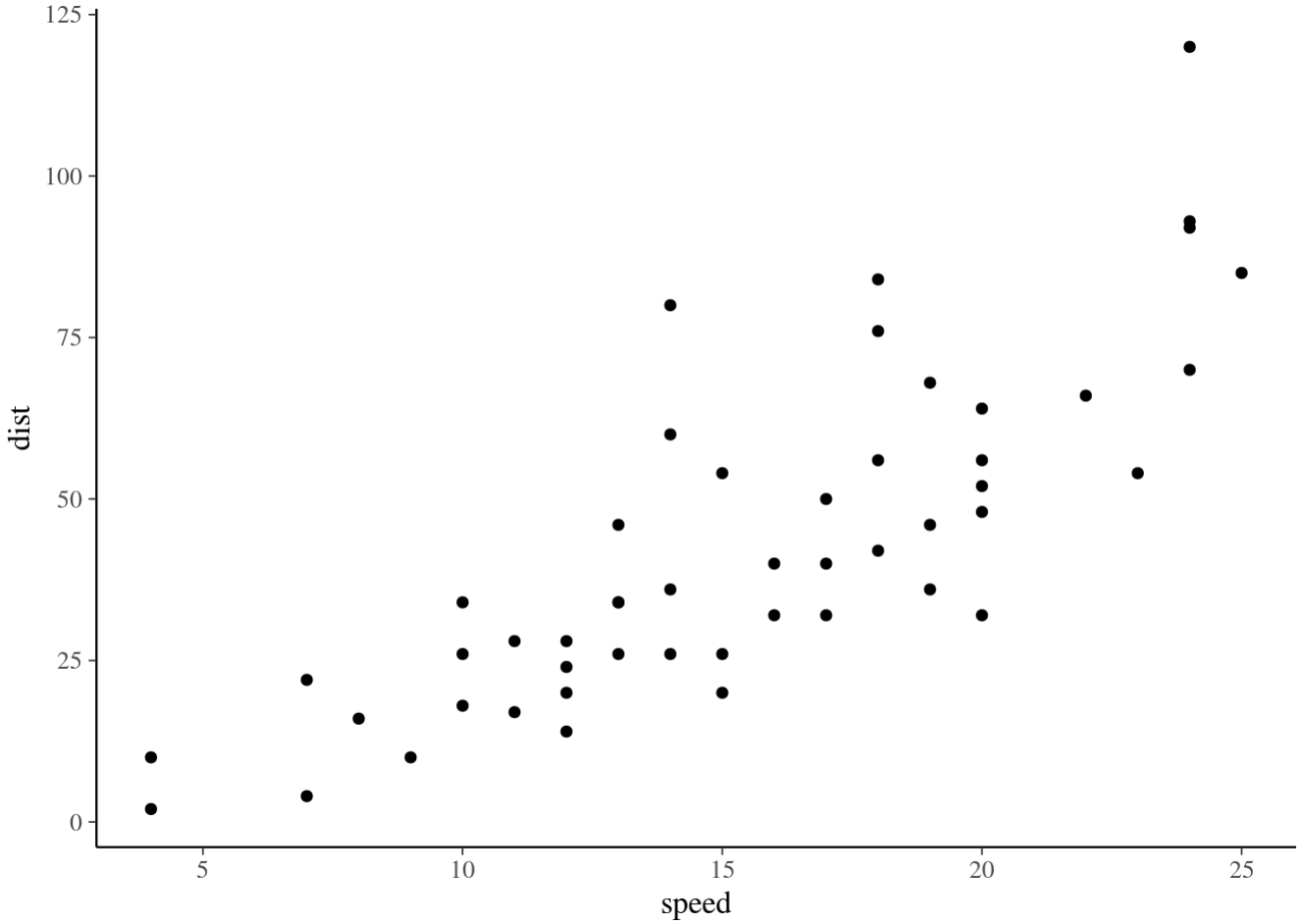
4. Perform inference. Once we have verified that everything looks good with our model and MCMC sampling, we can use the samples to perform inference on any quantity involving the posterior parameter distribution.

Linear Regression model with continuous outcome

Letâ€™s see how to apply these steps for a linear regression model. Weâ€™ll use the simple `cars` dataset which gives the speed of 50 cars along with the corresponding stopping distances.

HIDE

```
qplot(speed, dist, data=cars)
```



1. Specify the probability model. For this example, assume the usual linear regression model where each outcome observation (`dist`) is conditionally normal given the covariate (`speed`) with independent errors, $\text{dist} = \beta_0 + \beta_1 \cdot \text{speed} + \varepsilon,\ \varepsilon \sim N(0, \sigma^2)$. This implies the posterior will have 3 parameters, β_0, β_1 and σ^2. We will let `rstanarm` use the default priors for now to complete the probability model specification.

2. Draw samples from the posterior distribution. We can use the `rstanarm` function `stan_glm()` to draw samples from the posterior using the model above. Comparing the documentation for the `stan_glm()` function and the `glm()` function in base `R` , we can see the main arguments are identical.

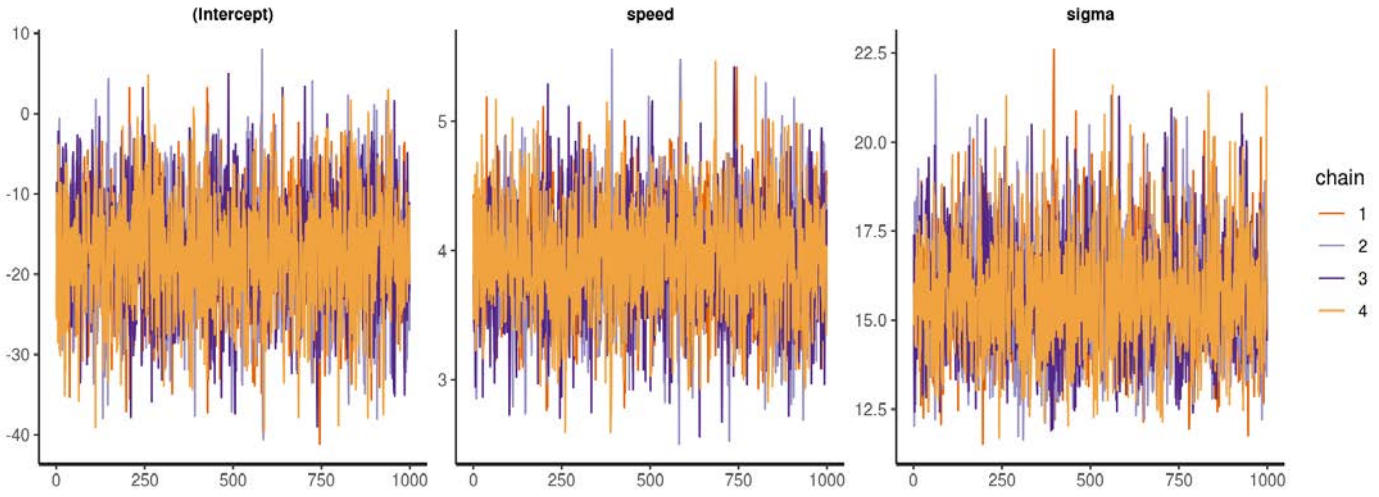
```
stan_glm(formula, family = gaussian(), data, weights, subset,
  na.action = NULL, offset = NULL, model = TRUE, x = FALSE,
  y = TRUE, contrasts = NULL, ..., prior = normal(),
  prior_intercept = normal(), prior_aux = exponential(),
  prior_PD = FALSE, algorithm = c("sampling", "optimizing",
  "meanfield", "fullrank"), mean_PPD = algorithm != "optimizing",
  adapt_delta = NULL, QR = FALSE, sparse = FALSE)

glm(formula, family = gaussian, data, weights, subset,
    na.action, start = NULL, etastart, mustart, offset,
    control = list(...), model = TRUE, method = "glm.fit",
    x = FALSE, y = TRUE, singular.ok = TRUE, contrasts = NULL, ...)
```

The actual model can be fit with a single line of code.

CODE

3. Evaluate the model. Two common checks for the MCMC sampler are trace plots and $\hat{R}$. We use the function `stan_trace()` to draw the trace plots which show sequential draws from the posterior distribution. Ideally we want the chains in each trace plot to be stable (centered around one value) and well-mixed (all chains are overlapping around the same value).

CODE



$\hat{R}$ uses estimates of variance within and between the chains to monitor convergence; at convergence $\hat{R} = 1$. Summarizing the model fit with `summary()` we see that $\hat{R}$ values are, in fact, 1 for all parameters. The output also gives us information about the model used and estimates of the posterior parameter distributions.

CODE

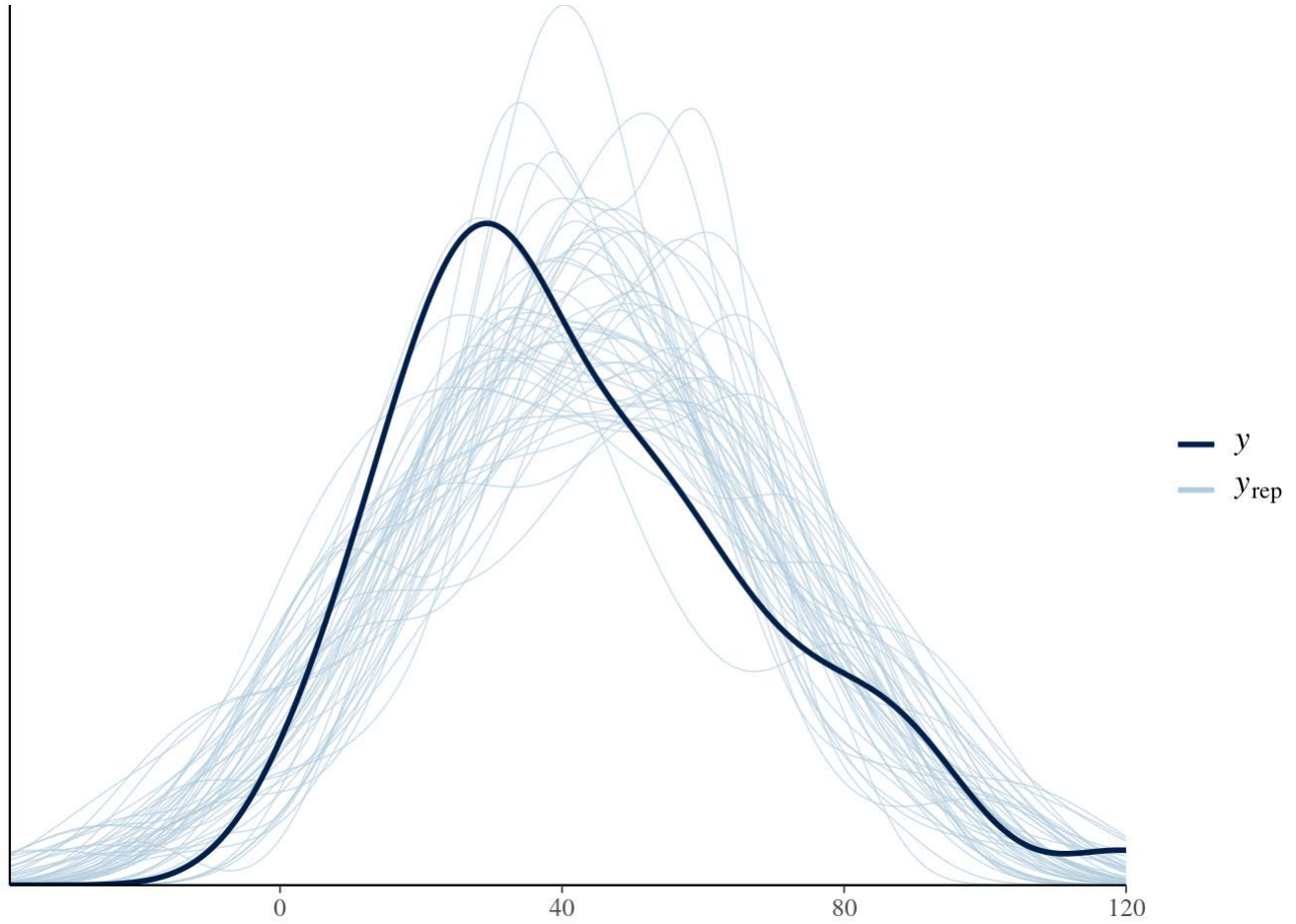
```
##
## Model Info:
##
##  function:     stan_glm
##  family:       gaussian [identity]
##  formula:      dist ~ speed
##  algorithm:    sampling
##  priors:       see help('prior_summary')
##  sample:       4000 (posterior sample size)
##  observations: 50
##  predictors:   2
##
## Estimates:
##                mean   sd     2.5%   25%    50%    75%    97.5%
## (Intercept)    -17.5    7.1  -31.6  -22.1  -17.6  -12.8   -3.3
## speed            3.9    0.4    3.1    3.7    3.9    4.2    4.8
## sigma           15.7    1.6   12.9   14.6   15.6   16.7   19.2
## mean_PPD        43.1    3.2   36.8   41.0   43.0   45.2   49.4
## log-posterior -216.7    1.3 -219.9 -217.2 -216.4 -215.8 -215.3
##
## Diagnostics:
##               mcse Rhat n_eff
## (Intercept)   0.1  1.0  3512
## speed         0.0  1.0  3637
## sigma         0.0  1.0  3022
## mean_PPD      0.1  1.0  3629
## log-posterior 0.0  1.0  1813
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sampl
e size, and Rhat is the potential scale reduction factor on split chains (at convergence Rhat=1).
```

In addition to checks on the MCMC sampler, we can look at posterior predictive checks.

> The idea behind posterior predictive checking is simple: If our model is a good fit then we should be able to use it to generate data that looks a lot like the data we observed.

The dark blue line shows the observed data while the light blue lines are simulations from the posterior predictive distribution.

CODE

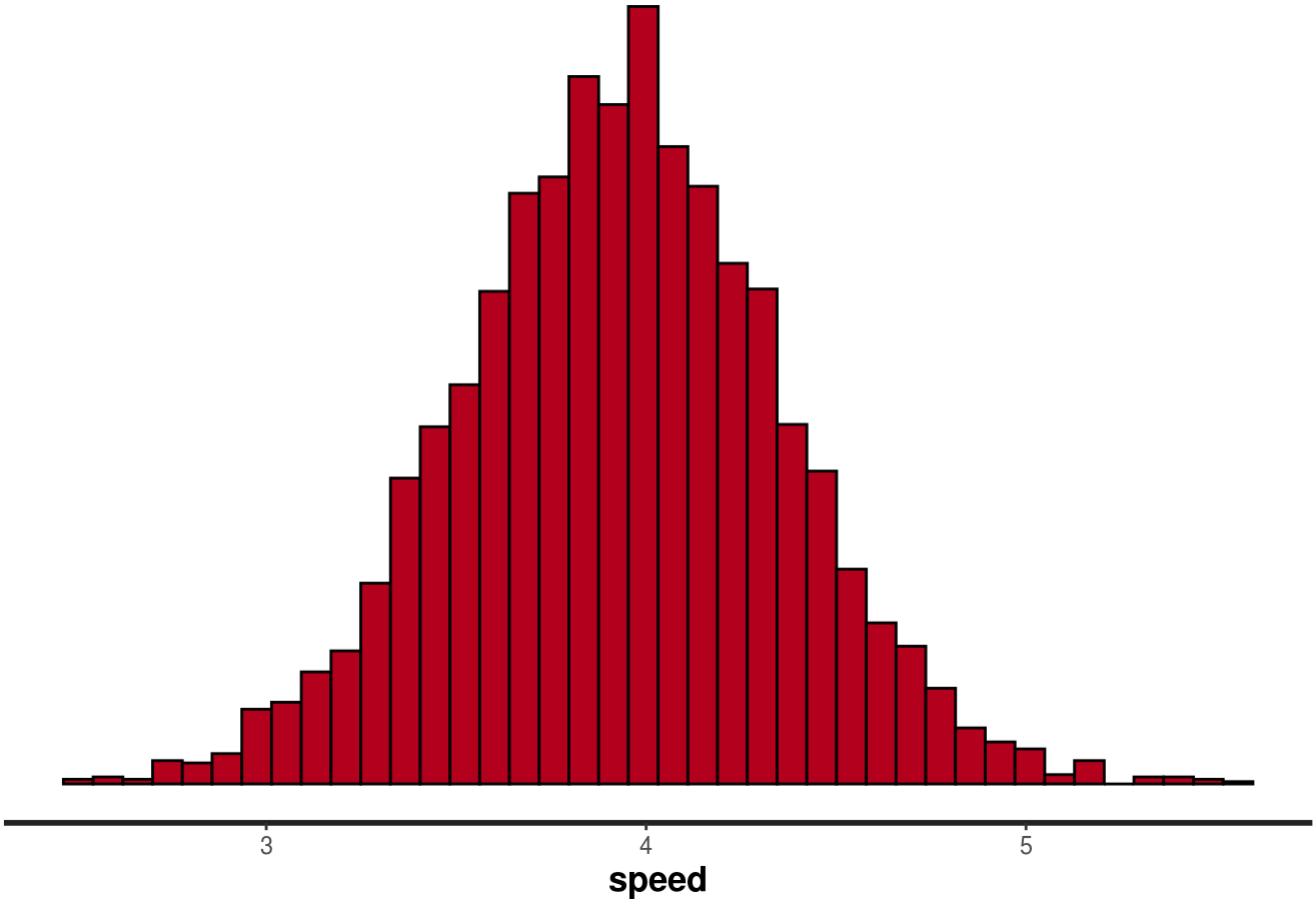


CODE

There are many other types of checks that can be explored interactively by running the `launch_shinystan()` function on the fit object.

4. Perform inference. We can use the posterior samples to perform inference. Focusing on the `speed` parameter, we first plot a histogram of the posterior samples. The posterior for this coefficient is centered at around 4 and is symmetric.

CODE



We can also extract these posterior samples to get an estimate of the mean and a 95% credible interval.

CODE

The speed parameter is the slope of the regression line from the model defined above. The average change in stopping distance associated with a one unit change in speed is 3.93. There is a 90% probability that the change in stopping distance associated with a one unit change in speed is between 3.23 and 4.62. We can also use the posterior to calculate other values of interest, for example the probability that the change in stopping distance for a unit change in speed is greater than 3.5 is 0.85 (calculated using `mean(post_samps_speed > 3.5)`).

Now, letâ€™s compare to a frequentist analysis. We can find maximum likelihood estimates for the linear regression model using `glm(..., family=gaussian)`.

CODE

```
##
## Call:
## glm(formula = dist ~ speed, family = gaussian, data = cars)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -29.069   -9.525   -2.272    9.215   43.201
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -17.5791     6.7584  -2.601   0.0123 *
## speed         3.9324     0.4155   9.464 1.49e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 236.5317)
##
##     Null deviance: 32539  on 49  degrees of freedom
## Residual deviance: 11354  on 48  degrees of freedom
## AIC: 419.16
##
## Number of Fisher Scoring iterations: 2
```

The maximum likelihood estimates for the intercept and slope are -17.58 and 3.93 which are nearly identical to the Bayesian posterior median values of -17.57 and 3.93.

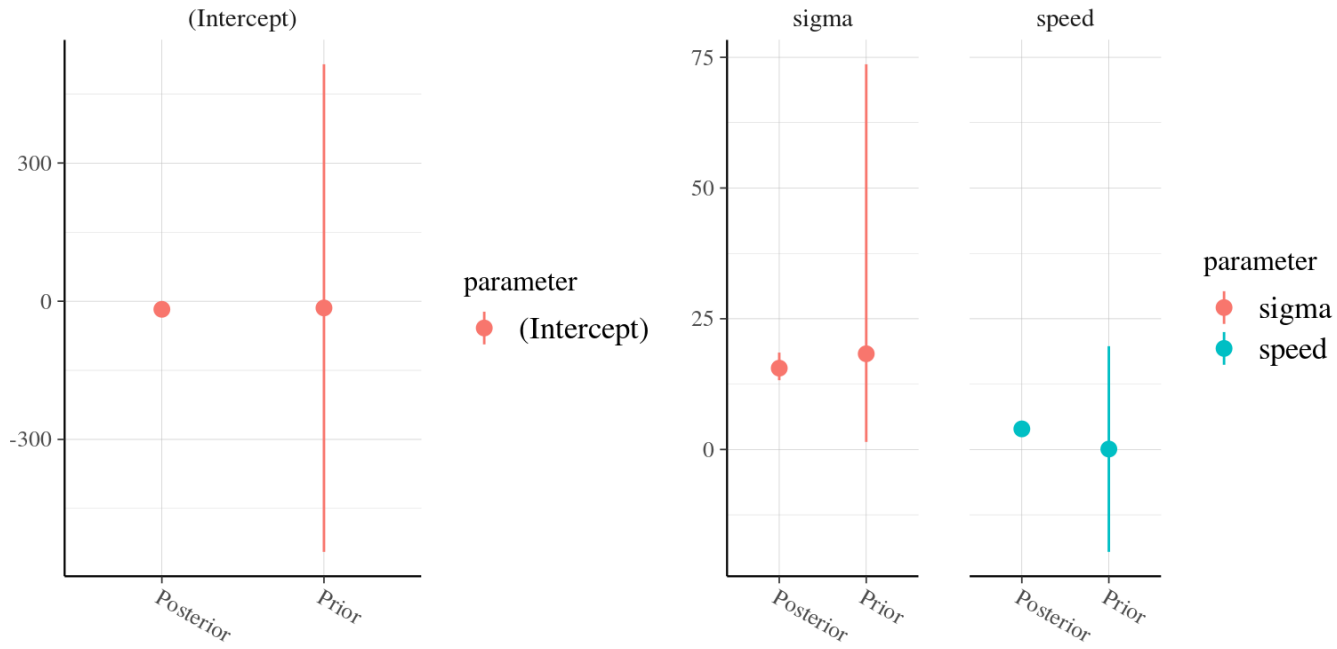
We can take a look at default priors used by `rstanarm` with the `prior_summary()` function. The intercept and coefficients both use a normal distribution centered around 0 as a prior, but the distribution for the intercept has higher variance (scale). The error term uses an exponential distribution as a prior.

CODE

```
## Priors for model 'glm_post1'
## ------
## Intercept (after predictors centered)
##  ~ normal(location = 0, scale = 10)
##      **adjusted scale = 257.69
##
## Coefficients
##  ~ normal(location = 0, scale = 2.5)
##      **adjusted scale = 12.18
##
## Auxiliary (sigma)
##  ~ exponential(rate = 1)
##      **adjusted scale = 25.77 (adjusted rate = 1/adjusted scale)
## ------
## See help('prior_summary.stanreg') for more details
```

It can also be helpful to juxtapose intervals from the prior distribution and the posterior distribution to see how the observed data has changed the parameter estimates.

CODE

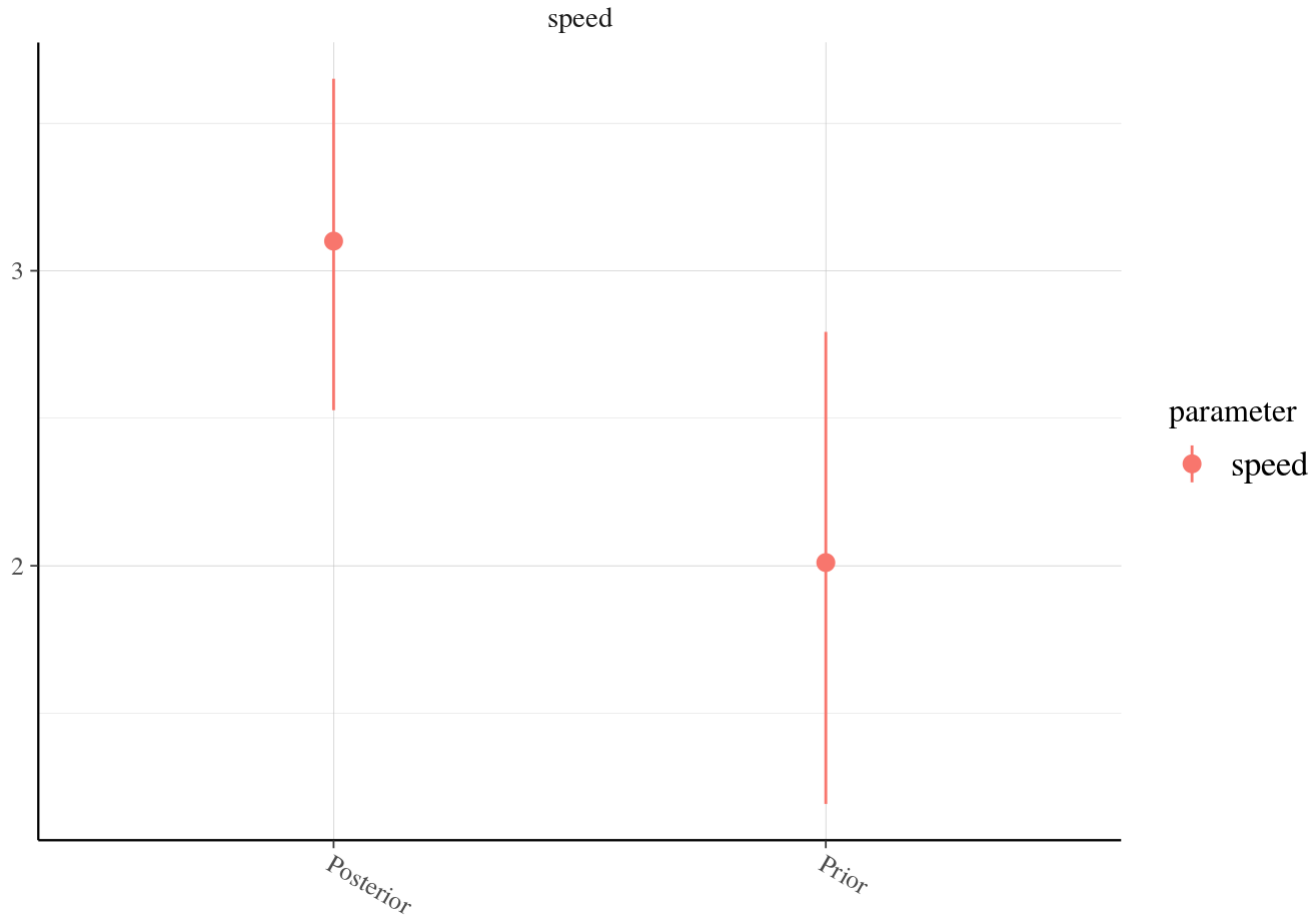


We can use a different prior for the slope by changing the `prior` argument in the call to `stan_glm()`. Letâ€™s use a much more informative normal prior for the slope centered around 2 with variance 0.5.

CODE

The posterior estimate for the slope is adjusted toward the prior value.

CODE



CODE

```
##
## Model Info:
##
##  function:     stan_glm
##  family:       gaussian [identity]
##  formula:      dist ~ speed
##  algorithm:    sampling
##  priors:       see help('prior_summary')
##  sample:       4000 (posterior sample size)
##  observations: 50
##  predictors:   2
##
## Estimates:
##                mean   sd     2.5%   25%    50%    75%    97.5%
## (Intercept)    -4.7    5.8  -16.1   -8.6   -4.7   -1.1    6.7
## speed           3.1    0.3    2.4    2.9    3.1    3.3    3.8
## sigma          16.2    1.7   13.2   15.0   16.1   17.3   20.0
## mean_PPD       42.9    3.2   36.5   40.8   43.0   45.1   49.1
## log-posterior -221.0    1.2 -224.2 -221.6 -220.7 -220.1 -219.6
##
## Diagnostics:
##               mcse Rhat n_eff
## (Intercept)   0.1  1.0  3275
## speed         0.0  1.0  3216
## sigma         0.0  1.0  3125
## mean_PPD      0.1  1.0  3477
## log-posterior 0.0  1.0  1782
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sampl
e size, and Rhat is the potential scale reduction factor on split chains (at convergence Rhat=1).
```

Generalized Linear Regression with binary outcome

We illustrate a logistic regression model using the example from a `rstanarm` vignette (https://cran.r-project.org/web/packages/rstanarm/vignettes/binomial.html (https://cran.r-project.org/web/packages/rstanarm/vignettes/binomial.html))

Gelman and Hill describe a survey of 3200 residents in a small area of Bangladesh suffering from arsenic contamination of groundwater. Respondents with elevated arsenic levels in their wells had been encouraged to switch their water source to a safe public or private well in the nearby area and the survey was conducted several years later to learn which of the affected residents had switched wells. The goal of the analysis presented by Gelman and Hill is to learn about the factors associated with switching wells.

CODE

```
##      switch          arsenic           dist             assoc
##  Min.   :0.0000   Min.   :0.510   Min.   :  0.387   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.820   1st Qu.: 21.117   1st Qu.:0.0000
##  Median :1.0000   Median :1.300   Median : 36.761   Median :0.0000
##  Mean   :0.5752   Mean   :1.657   Mean   : 48.332   Mean   :0.4228
##  3rd Qu.:1.0000   3rd Qu.:2.200   3rd Qu.: 64.041   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :9.650   Max.   :339.531   Max.   :1.0000
##       educ
##  Min.   : 0.000
##  1st Qu.: 0.000
##  Median : 5.000
##  Mean   : 4.828
##  3rd Qu.: 8.000
##  Max.   :17.000
```

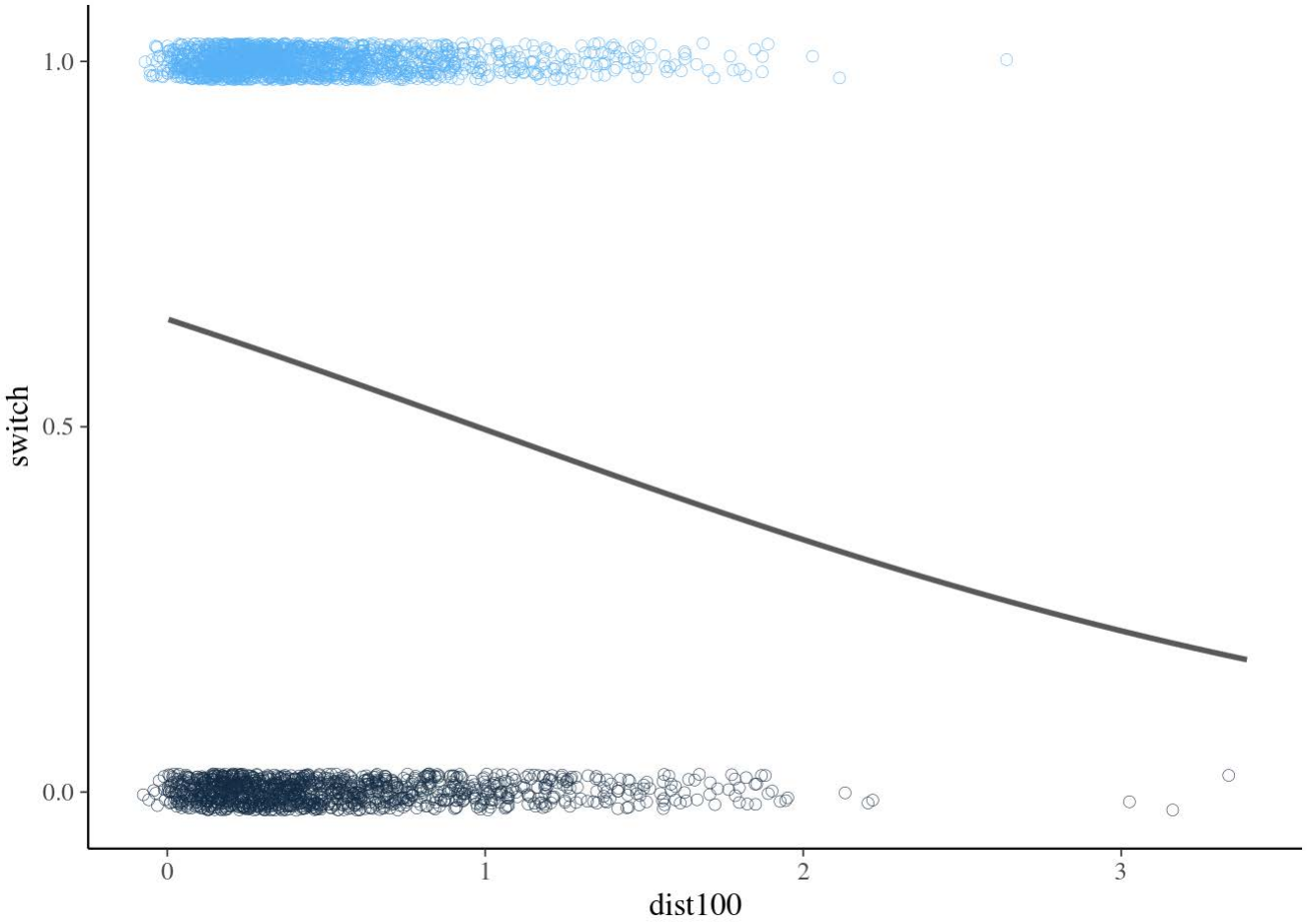
CODE

Weâ€™ll use a logistic regression model for the likelihood with Student t distributions centered around 0 for the intercept and covariate priors. For the first model, we only consider a single covariate, `dist100`, which is the distance from the well (in units of 100 meters).

CODE

Plotting the median predicted posterior probability of switching as a function of distance, we see that increased distance does reduce the probability of switching, as would be expected.

CODE

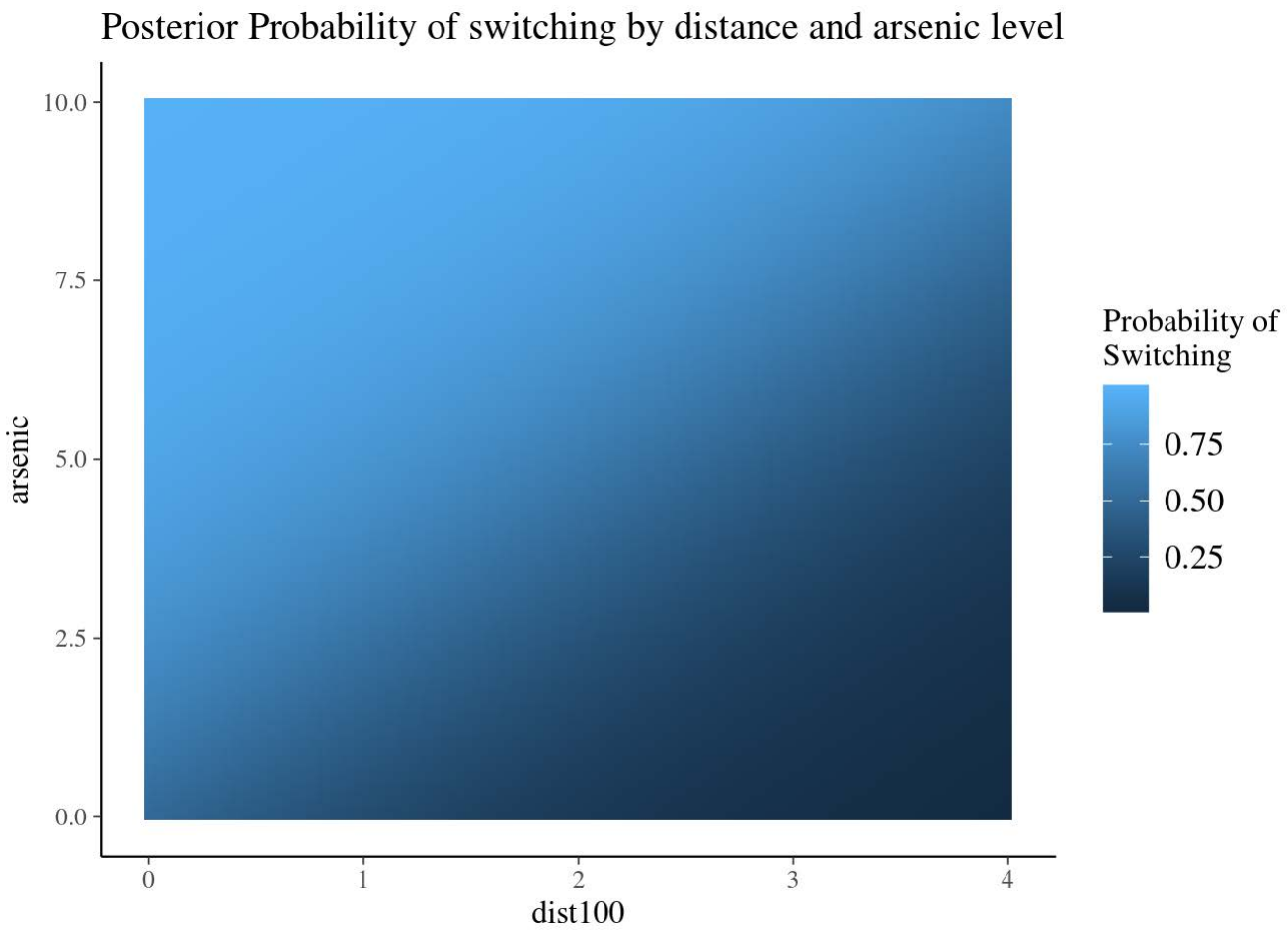


For the next model, we also add the covariate `arsenic` which is a measure of the arsenic level in the residentsâ€™ well (before switching). We can use the `update()` function to change the formula statement and rerun the model.

CODE

Now we plot the median predicted posterior probability of switching as a function of both distance and arsenic level.

CODE

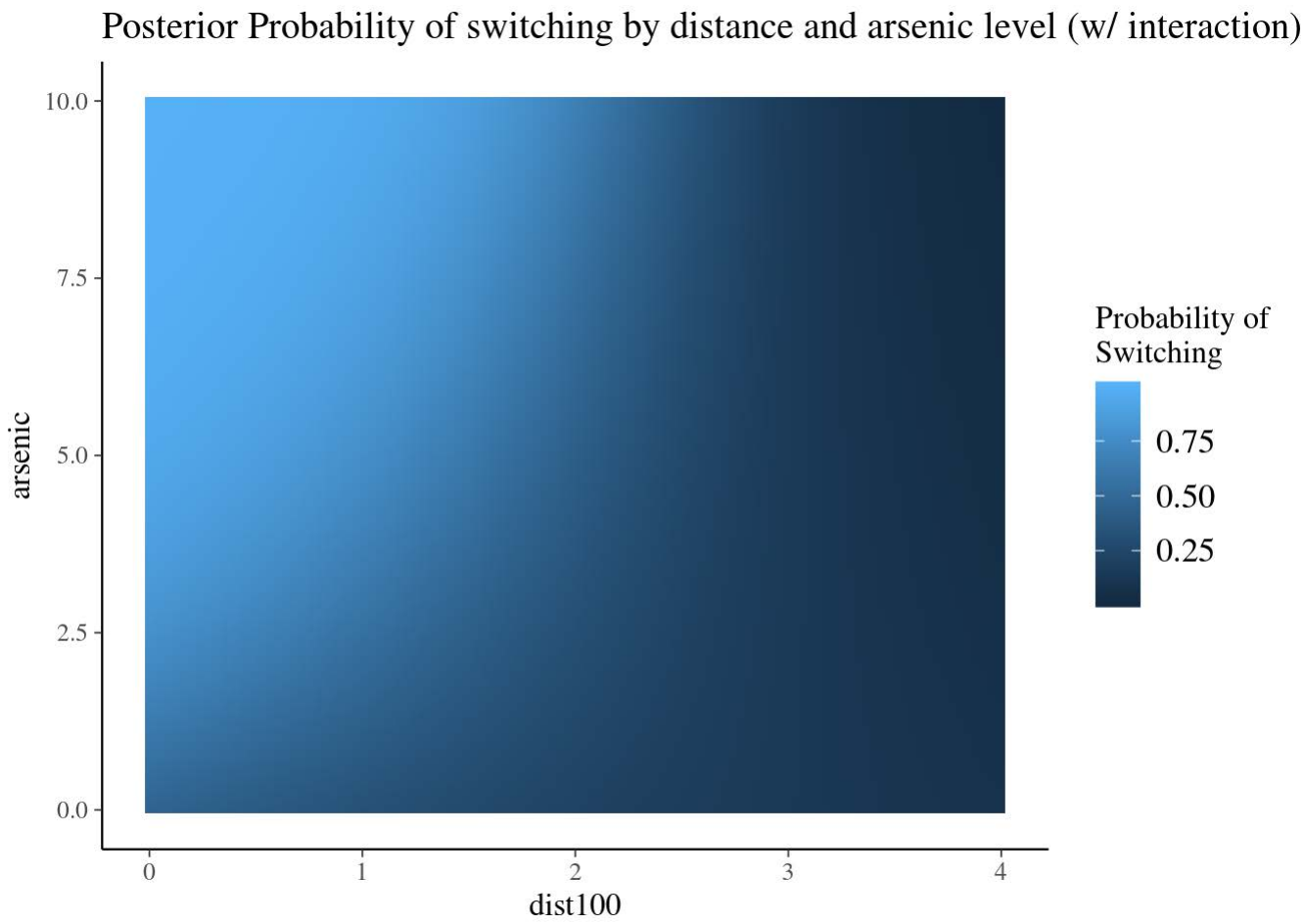


Finally, we add in an interaction between distance and arsenic level so the change in the odds of switching associated with higher arsenic or farther distance depends on the value of the other covariate.

CODE

Recreating the previous plot with the new model, it appears that those with very elevated arsenic levels only have a high probability of switching if they are also within about 200m of the safe public or private well.

CODE



We can compare our three models using an approximation to Leave-One-Out (LOO) cross-validation, which is a method for estimating out of sample predictive performance and is implemented by the `loo` function in the `loo` package:

CODE

```
##
## Model comparison:
## (ordered by highest ELPD)
##
##           elpd_diff se_diff
## glm_post5   0.0       0.0
## glm_post4  -0.4       1.9
## glm_post3 -72.0      12.4
```

Including `arsenic` gives much better estimated predictive performance than `dist100` alone, but the interaction model is similar to the model with separate linear terms only.

Other rstanarm models

stan_betareg() - beta regression models

stan_biglm() - regularized linear but big models (data too large to fit in memory)

stan_clogit() - conditional logistic regression models

stan_gamm4() - generalized linear additive models with optional group-specific terms
stan_glmer() - generalized linear models with group-specific terms
stan_lm() - regularized linear models
stan_mvmer() - multivariate generalized linear models with correlated group-specific terms stan_nlmer() - nonlinear models with group-specific terms
stan_polr() - ordinal regression models
stan_jm() - joint longitudinal and time-to-event models

Troubleshooting

If your get an error message or run into problems with your model checks, you can get help at the rstanarm troubleshooting (http://mc-stan.org/rstanarm/articles/rstanarm.html#troubleshooting) page.

Going beyond rstanarm

Stan without rstanarm

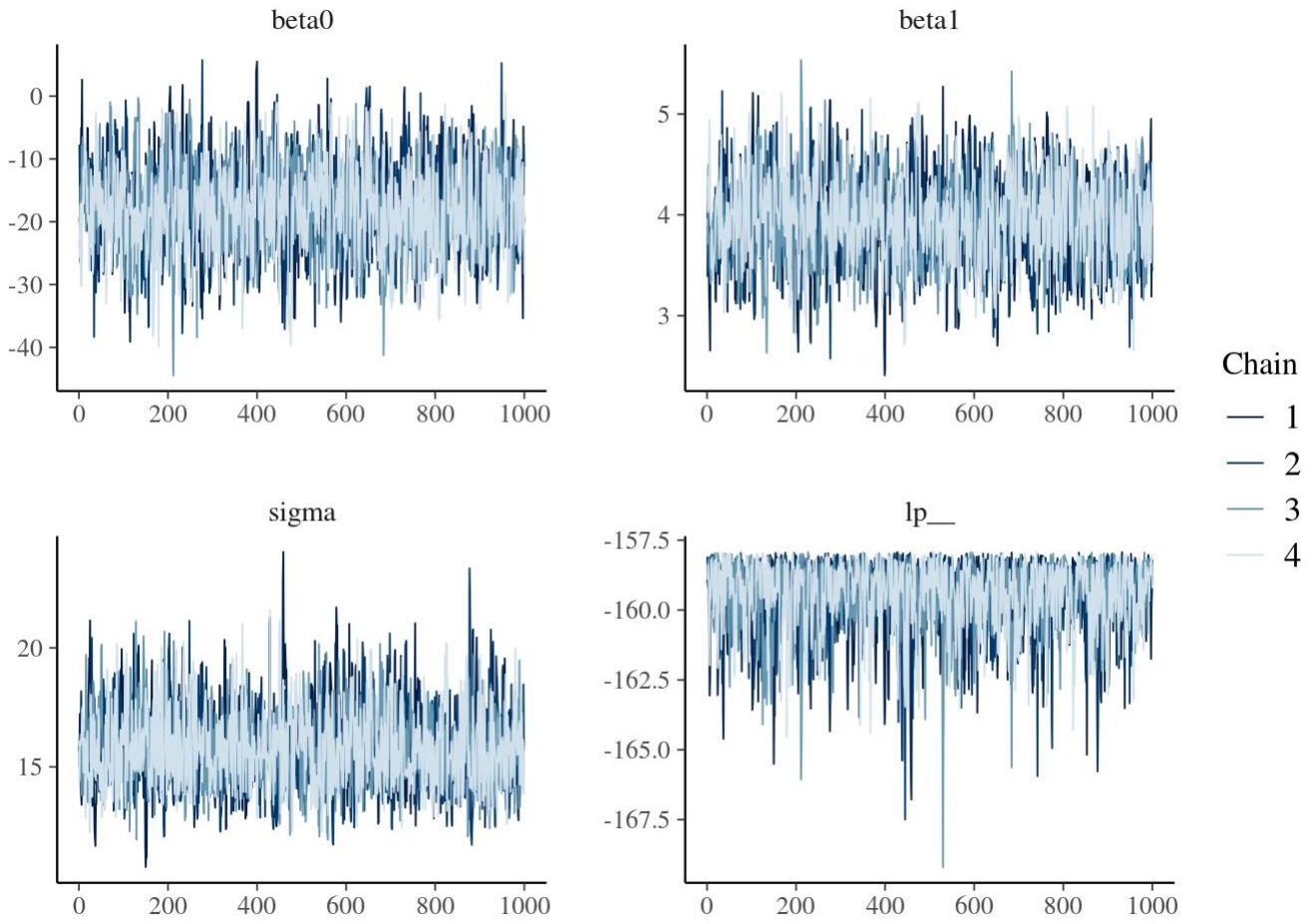
A good place to start is the first chapter of the Stan Userâ€™s guide (https://mc-stan.org/users/documentation/ (https://mc-stan.org/users/documentation/)).

The `rstan` code to fit the first linear regression model is below:

CODE

We can perform the same model checks and posterior inference using the `rstan` model fit.

CODE



CODE

```
##              mean    se_mean        sd        2.5%        25%         50%
## beta0  -17.724217 0.17411048 6.9579361  -31.505316  -22.30772  -17.790955
## beta1    3.942747 0.01081580 0.4275461    3.126162    3.64796    3.939267
## sigma   15.772270 0.03607351 1.6513163   12.961283   14.59771   15.634328
## lp__  -159.468207 0.03184993 1.2524041 -162.693288 -160.03626 -159.152486
##               75%       97.5%    n_eff      Rhat
## beta0  -12.967300   -4.055019 1597.023 1.0005815
## beta1    4.231986    4.781553 1562.604 1.0004745
## sigma   16.786673   19.476697 2095.481 0.9999995
## lp__  -158.551635 -158.033690 1546.223 1.0005155
```

You can also get a sense for what is going on under the hood by looking at the stan model underlying any `rstanarm` model with the `get_stanmodel()` function, but be warned that this code is designed for efficiency rather than readability.

CODE

Survival analysis

The current production version of `rstanarm` (v 2.18.2) does not have a function for survival analysis, but there are ongoing discussions (https://discourse.mc-stan.org/t/survival-models-in-rstanarm/3998) about implementing survival models in `rstanarm`. You can also check details about the experimental function (http://rstudio-pubs-static.s3.amazonaws.com/438966_3b8a25efb9b84454b8d69b7a15e3ebc5.html) `stan_surv()`, currently on development branch of `rstanarm`.

A good alternative is `brms`, another front-end to `rstan` that also uses formula syntax and can fit many of the same models.

CODE

References

Code for this Rmd file is at: https://github.com/ntjames/rstanarm_seminar (https://github.com/ntjames/rstanarm_seminar)

The `rstanarm` page on the Stan site: http://mc-stan.org/rstanarm/index.html (http://mc-stan.org/rstanarm/index.html)

A lot of this material is borrowed from the official `rstanarm` vignettes by Jonah Gabry and Ben Goodrich:
https://cran.r-project.org/web/packages/rstanarm/vignettes/rstanarm.html (https://cran.r-project.org/web/packages/rstanarm/vignettes/rstanarm.html)
https://cran.r-project.org/web/packages/rstanarm/vignettes/continuous.html (https://cran.r-project.org/web/packages/rstanarm/vignettes/continuous.html)
https://cran.rstudio.com/web/packages/rstanarm/vignettes/binomial.html (https://cran.rstudio.com/web/packages/rstanarm/vignettes/binomial.html)

A few good reference for Bayesian regression modeling are:

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). *Bayesian data analysis*. Chapman and Hall/CRC.

Carlin, B. P., & Louis, T. A. (2008). *Bayesian methods for data analysis*. Chapman and Hall/CRC.

McElreath, R. (2018). *Statistical rethinking: A Bayesian course with examples in R and Stan*. Chapman and Hall/CRC.