Module 10: rstan

To learn about rstan, consider this vignette:

https://cran.r-project.org/web/packages/rstan/vignettes/rstan.html

See here for prior choice recommendation:

https://github.com/stan-dev/stan/wiki/Prior-Choice-Recommendations

In this notebook, we start simpler, estimating just a mean and variance, followed by the fitting of a hierarchical model. We compare the model used to that used (implicitly) in brm.

Radon data

Read in the radon data and process (copied from earlier module)

```
# house level data
d <- read.table(url("http://www.stat.columbia.edu/~gelman/arm/examples/radon/srrs2.dat"),
                header=T, sep=",")

# deal with zeros, select what we want, make a fips (county) variable to match on
d <- d %>%
  mutate(activity = ifelse(activity==0, 0.1, activity)) %>%
  mutate(fips = stfips * 1000 + cntyfips) %>%
  dplyr::select(fips, state, county, floor, activity)

# county level data
cty <- read.table(url("http://www.stat.columbia.edu/~gelman/arm/examples/radon/cty.dat"),
                  header = T, sep = ",")
cty <-
  cty %>%
  mutate(fips = 1000 * stfips + ctfips) %>%
  dplyr::select(fips, Uppm) %>%
  rename(ura_county = (Uppm))

dmn <- d %>%
  filter(state=="MN") %>% # Minnesota data only
  dplyr::select(fips, county, floor, activity) %>%
  left_join(cty)

dat <-
  dmn%>%
  mutate(y = log(activity), log_ur = log(ura_county))
head(dat)
```

```
##    fips                county floor activity ura_county         y     log_ur
## 1 27001 AITKIN                    1      2.2   0.502054 0.7884574 -0.6890476
```

```
## 2 27001 AITKIN                    0       2.2   0.502054 0.7884574 -0.6890476
## 3 27001 AITKIN                    0       2.9   0.502054 1.0647107 -0.6890476
## 4 27001 AITKIN                    0       1.0   0.502054 0.0000000 -0.6890476
## 5 27003 ANOKA                     0       3.1   0.428565 1.1314021 -0.8473129
## 6 27003 ANOKA                     0       2.5   0.428565 0.9162907 -0.8473129
```

Using rstan to estimate mean and variance

Start simple, let $y_i = \log(\text{radon})$, and assume

$$y_i|\mu, \sigma \sim N(\mu, \sigma^2),$$

to estimate μ and σ.

We need to define data inputs and a stan model file.

For the stan model file template, go to "File", "new file", "stan file". Note that the default is to estimate μ and σ as in our seting! The default template has flat priors, we can add priors. Save it, I called it "module10_mean_addpriors.stan".

Create data inputs:

```
stan_dat <- list(y = dat$y, N = length(dat$y))
```

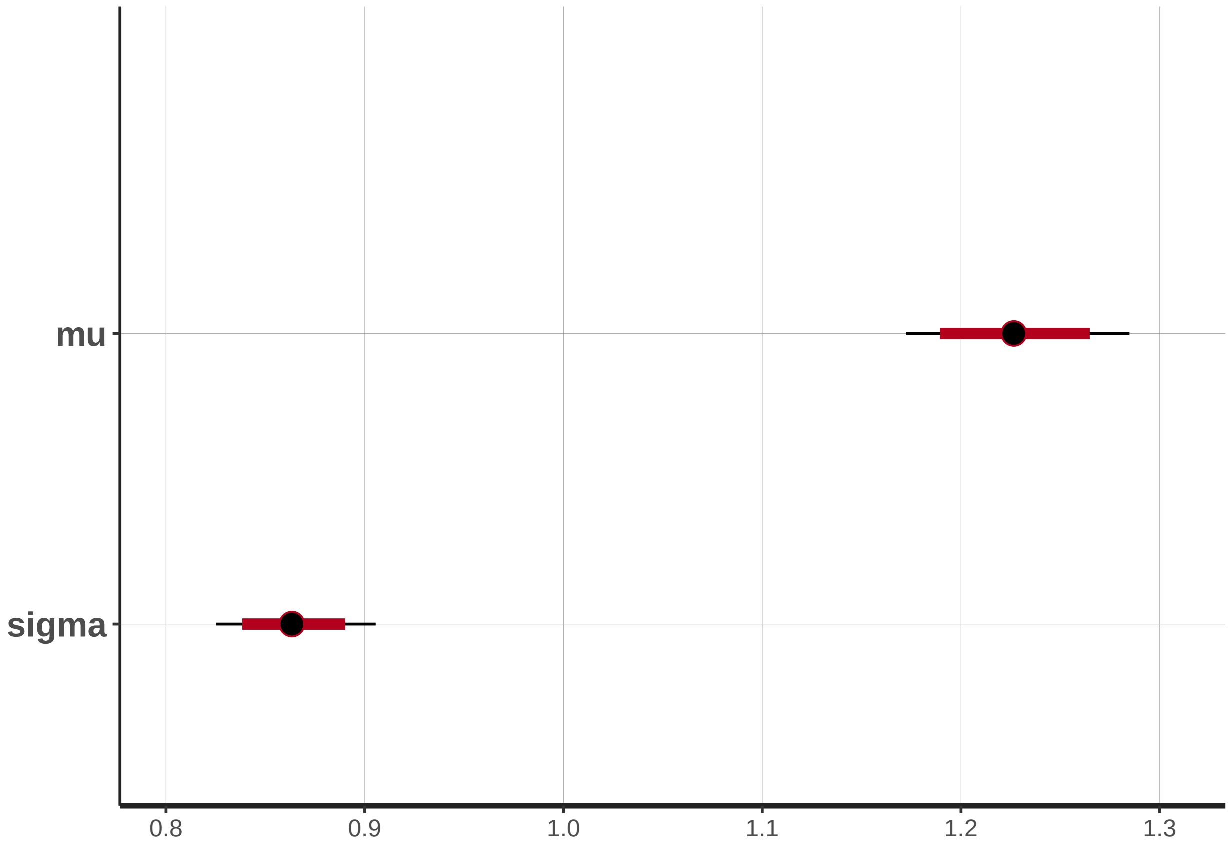
```
fit <- stan(file = 'module10_stan_mean_addpriors.stan', data = stan_dat)
```

Showing some default oupputs, see also the Rstan vignette

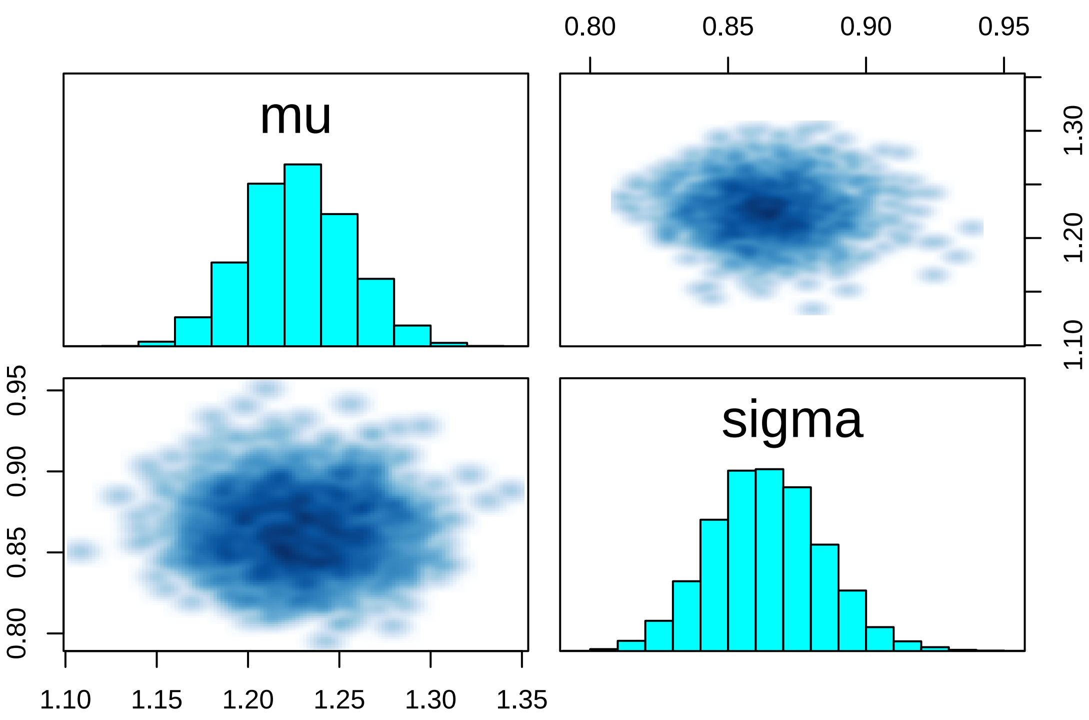
```
print(fit)
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
## mu       1.23    0.00 0.03    1.17    1.21    1.23    1.25    1.28  2984    1
## sigma    0.86    0.00 0.02    0.83    0.85    0.86    0.88    0.91  3255    1
## lp__  -329.22    0.03 1.02 -331.81 -329.62 -328.91 -328.49 -328.21  1569    1
##
## Samples were drawn using NUTS(diag_e) at Tue Oct 18 14:09:56 2022.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

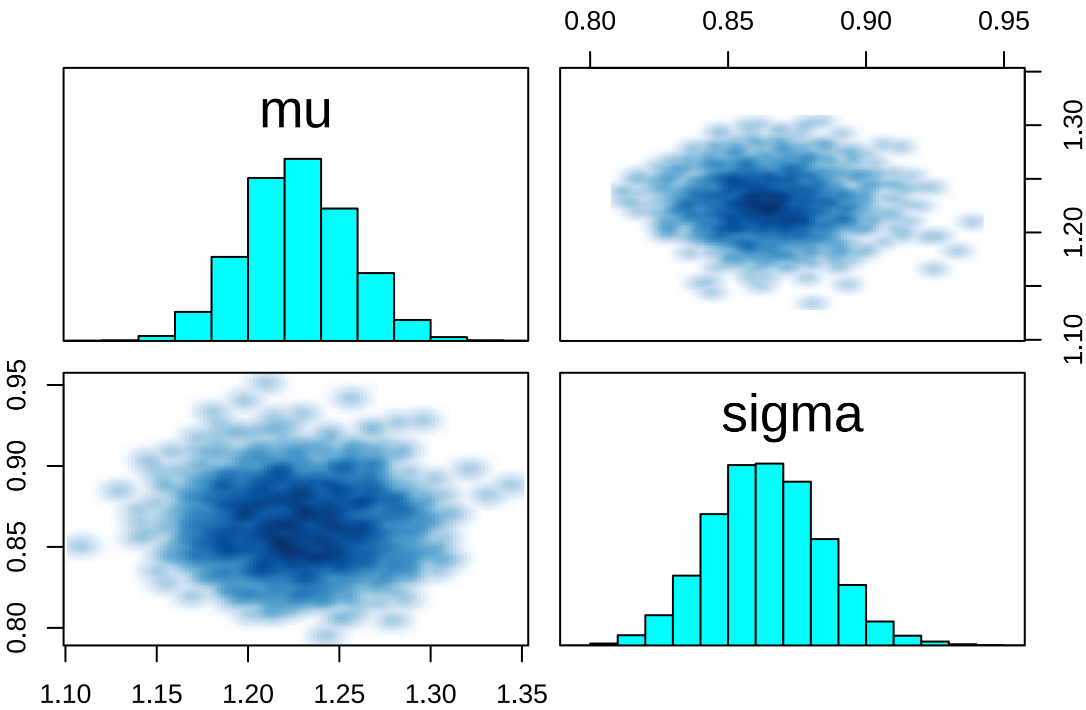
```
plot(fit)
```



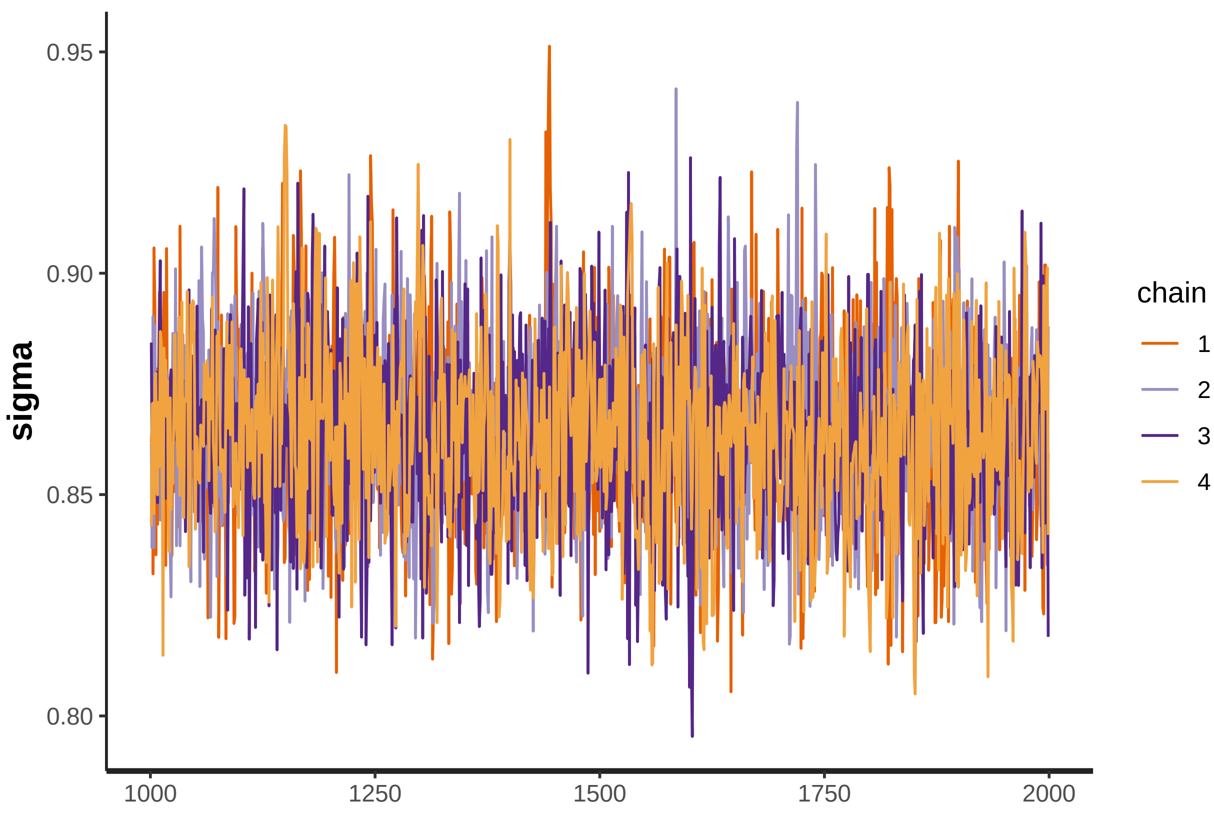

```
pairs(fit, pars = c("mu", "sigma"))
```



```r
pairs(fit, pars = c("mu", "sigma"))
```




```r
traceplot(fit, pars = c("sigma"))
```



Compare to brm-based model

```r
fit_brm <- brm(y ~ 1, data = dat,
        chains = 4, iter = 1000, warmup = 500, cores = getOption("mc.cores", 4))
```

```r
summary(fit_brm)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: y ~ 1
##    Data: dat (Number of observations: 927)
##   Draws: 4 chains, each with iter = 1000; warmup = 500; thin = 1;
##          total post-warmup draws = 2000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     1.23      0.03     1.17     1.29 1.00     1794     1275
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.86      0.02     0.83     0.91 1.00     1636     1276
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```r
print(fit)
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
## mu       1.23    0.00 0.03    1.17    1.21    1.23    1.25    1.28  2984    1
## sigma    0.86    0.00 0.02    0.83    0.85    0.86    0.88    0.91  3255    1
## lp__  -329.22    0.03 1.02 -331.81 -329.62 -328.91 -328.49 -328.21  1569    1
##
## Samples were drawn using NUTS(diag_e) at Tue Oct 18 14:09:56 2022.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```r
stan_data_from_brm <- standata(fit_brm)
names(stan_data_from_brm)
```

```
## [1] "N"          "Y"          "K"          "X"          "prior_only"
```

```r
# N and y (called Y here) are same as in stan_dat
stan_dat$N
```

```
## [1] 927
```

```
stan_data_from_brm$N
```

```
## [1] 927
```

```
summary(stan_data_from_brm$Y)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -2.3026  0.6419  1.2809  1.2275  1.8245  3.8754
```

```
summary(dat$y)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -2.3026  0.6419  1.2809  1.2275  1.8245  3.8754
```

```
stancode(fit_brm)
```

```
## // generated with brms 2.17.0
## functions {
## }
## data {
##   int<lower=1> N;  // total number of observations
##   vector[N] Y;  // response variable
##   int prior_only;  // should the likelihood be ignored?
## }
## transformed data {
## }
## parameters {
##   real Intercept;  // temporary intercept for centered predictors
##   real<lower=0> sigma;  // dispersion parameter
## }
## transformed parameters {
##   real lprior = 0;  // prior contributions to the log posterior
##   lprior += student_t_lpdf(Intercept | 3, 1.3, 2.5);
##   lprior += student_t_lpdf(sigma | 3, 0, 2.5)
##     - 1 * student_t_lccdf(0 | 3, 0, 2.5);
## }
## model {
##   // likelihood including constants
##   if (!prior_only) {
##     // initialize linear predictor term
##     vector[N] mu = Intercept + rep_vector(0.0, N);
##     target += normal_lpdf(Y | mu, sigma);
##   }
##   // priors including constants
##   target += lprior;
## }
## generated quantities {
##   // actual population-level intercept
##   real b_Intercept = Intercept;
## }
```

See slides re what's going on in this model!

For just the priors

```
get_prior(y~1, data = dat)
```

```
##                  prior     class coef group resp dpar nlpar lb ub  source
##  student_t(3, 1.3, 2.5) Intercept                                  default
##    student_t(3, 0, 2.5)     sigma                            0    default
```

Fit a multilevel regression model

Consider the following model: $y_i|\mu_i, \sigma \sim N(\mu_i, \sigma^2)$, where

$$\mu_i = \mu_\alpha + \eta_{j[i]} + \beta x_i,$$

where $j[i]$ refers to county and x_i to floor indicator.

The stan model for this is given in module10_hier_regression.stan.

```
dat <- dat %>%
  mutate(county_id = as.numeric(as_factor(county)))
stan_dat2 <- list(y = dat$y, N = length(dat$y), x = dat$floor, county_id = dat$county_id,
                  J = max(dat$county_id))
```

```
fit2 <- stan(file = 'module10_hier_regression.stan', data = stan_dat2)
```

```
print(fit)
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
## mu       1.23    0.00 0.03    1.17    1.21    1.23    1.25    1.28  2984    1
## sigma    0.86    0.00 0.02    0.83    0.85    0.86    0.88    0.91  3255    1
## lp__  -329.22    0.03 1.02 -331.81 -329.62 -328.91 -328.49 -328.21  1569    1
##
## Samples were drawn using NUTS(diag_e) at Tue Oct 18 14:09:56 2022.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```