4.6 regression models for prediction

Chris Parrish

July 3, 2016

Contents

4.6 regression models for prediction

reference:
- ARM chapter 04, github

```
library(rstan)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
library(ggplot2)
```

mesquite

data

```
### Data
source("mesquite.data.R", echo = TRUE)
```

```
##
## > N <- 46
##
```

```
## > weight <- c(401.3, 513.7, 1179.2, 308, 855.2, 268.7,
## +     155.5, 1253.2, 328, 614.6, 60.2, 269.6, 448.4, 120.4, 378.7,
## +     266.4, 138.9, 1020.8 .... [TRUNCATED]
##
## > diam1 <- c(1.8, 1.7, 2.8, 1.3, 3.3, 1.4, 1.5, 3.9,
## +     1.8, 2.1, 0.8, 1.3, 1.2, 1.5, 2.8, 1.4, 1.5, 2.4, 1.9, 2.3,
## +     2.1, 2.4, 1, 1.3, 1.1,  .... [TRUNCATED]
##
## > diam2 <- c(1.15, 1.35, 2.55, 0.85, 1.9, 1.4, 0.5,
## +     2.3, 1.35, 1.6, 0.63, 0.95, 0.9, 0.7, 1.7, 0.85, 0.6, 2.4,
## +     1.55, 1.6, 1.7, 1.3, 0.4, .... [TRUNCATED]
##
## > canopy_height <- c(1, 1.33, 0.6, 1.2, 1.05, 1, 0.9,
## +     1.3, 0.6, 0.8, 0.6, 0.95, 1.2, 0.7, 1.2, 1.1, 0.64, 1.2,
## +     1.2, 1.3, 1, 0.9, 1, 0.5, .... [TRUNCATED]
##
## > total_height <- c(1.3, 1.35, 2.16, 1.8, 1.55, 1.2,
## +     1, 1.7, 0.8, 1.2, 0.9, 1.35, 1.4, 1, 1.7, 1.5, 0.65, 1.5,
## +     1.7, 1.7, 1.5, 1.5, 1.2,  .... [TRUNCATED]
##
## > density <- c(1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1,
## +     1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 5, 9, 1, 1, 1, 3,
## +     1, 3, 7, 1, 2, 2, 2, 3, 1 .... [TRUNCATED]
##
## > group <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
## +     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
## +     1, 1, 1, 1, 1, 1, 1, 1, 1,  .... [TRUNCATED]
```

model

mesquite.stan

```
data {
  int<lower=0> N;
  vector[N] weight;
  vector[N] diam1;
  vector[N] diam2;
  vector[N] canopy_height;
  vector[N] total_height;
  vector[N] density;
  vector[N] group;
}
parameters {
  vector[7] beta;
  real<lower=0> sigma;
}
model {
  weight ~ normal(beta[1] +
                  beta[2] * diam1 +
                  beta[3] * diam2 +
                  beta[4] * canopy_height +
                  beta[5] * total_height +
                  beta[6] * density +
                  beta[7] * group,
```

```
                  sigma);
}
```

fit

```
### First model: weight ~ diam1 + diam2 + canopy.height + total.height + density + group
data.list <- c("N", "weight", "diam1", "diam2", "canopy_height", "total_height",
               "density", "group")
mesquite.sf <- stan(file='mesquite.stan', data=data.list,
                    iter=1000, chains=4)
print(mesquite.sf)
```

```
## Inference for Stan model: mesquite.
## 4 chains, each with iter=1000; warmup=500; thin=1;
## post-warmup draws per chain=500, total post-warmup draws=2000.
##
##            mean se_mean     sd     2.5%     25%     50%     75%   97.5%
## beta[1] -730.56    4.32 148.51 -1018.89 -827.82 -732.49 -631.82 -433.06
## beta[2]  186.37    4.10 120.70   -59.30  107.12  186.05  270.88  414.39
## beta[3]  375.98    4.21 131.60   130.60  286.54  371.89  464.42  646.15
## beta[4]  361.85    6.86 216.92   -76.96  217.49  365.06  507.16  786.54
## beta[5] -103.86    5.83 191.54  -474.83 -229.62 -102.72   23.80  264.42
## beta[6]  130.77    0.96  35.55    58.78  106.77  129.95  154.06  197.10
## beta[7] -366.63    2.81 101.19  -574.64 -431.13 -365.12 -304.05 -163.63
## sigma    279.00    0.88  31.42   225.41  256.13  276.17  299.27  345.82
## lp__    -275.41    0.08   2.18  -280.67 -276.70 -275.08 -273.73 -272.27
##         n_eff Rhat
## beta[1]  1180    1
## beta[2]   866    1
## beta[3]   975    1
## beta[4]   999    1
## beta[5]  1079    1
## beta[6]  1373    1
## beta[7]  1300    1
## sigma    1264    1
## lp__      791    1
##
## Samples were drawn using NUTS(diag_e) at Wed Jul  6 00:46:55 2016.
## 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).
##  The estimated Bayesian Fraction of Missing Information is a measure of
##  the efficiency of the sampler with values close to 1 being ideal.
##  For each chain, these estimates are
##  0.9 1 0.9 0.9
```

log model

model

mesquite_log.stan

```
data {
  int<lower=0> N;
  vector[N] weight;
  vector[N] diam1;
  vector[N] diam2;
  vector[N] canopy_height;
  vector[N] total_height;
  vector[N] density;
  vector[N] group;
}
transformed data {           // log transformations
  vector[N] log_weight;
  vector[N] log_diam1;
  vector[N] log_diam2;
  vector[N] log_canopy_height;
  vector[N] log_total_height;
  vector[N] log_density;
  log_weight        = log(weight);
  log_diam1         = log(diam1);
  log_diam2         = log(diam2);
  log_canopy_height = log(canopy_height);
  log_total_height  = log(total_height);
  log_density       = log(density);
}
parameters {
  vector[7] beta;
  real<lower=0> sigma;
}
model {
  log_weight ~ normal(beta[1] +
                      beta[2] * log_diam1 +
                      beta[3] * log_diam2 +
                      beta[4] * log_canopy_height +
                      beta[5] * log_total_height +
                      beta[6] * log_density +
                      beta[7] * group,
                      sigma);
}
```

fit

```
### Log model: log(weight) ~ log(diam1) + log(diam2) + log(canopy.height)
###                          + log(total.height) + log(density) + group

mesquite_log.sf <- stan(file='mesquite_log.stan', data=data.list,
                        iter=1000, chains=4)
print(mesquite_log.sf)
```

```
## Inference for Stan model: mesquite_log.
## 4 chains, each with iter=1000; warmup=500; thin=1;
## post-warmup draws per chain=500, total post-warmup draws=2000.
##
##          mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
```

```
## beta[1]  5.35    0.01 0.17  5.02  5.24  5.34  5.46  5.68   866 1.00
## beta[2]  0.40    0.01 0.29 -0.18  0.21  0.40  0.58  0.99  1043 1.00
## beta[3]  1.15    0.01 0.22  0.72  1.01  1.16  1.29  1.57  1072 1.01
## beta[4]  0.37    0.01 0.28 -0.18  0.18  0.37  0.55  0.94  1189 1.00
## beta[5]  0.39    0.01 0.31 -0.25  0.19  0.38  0.60  1.01  1178 1.00
## beta[6]  0.11    0.00 0.13 -0.15  0.03  0.11  0.19  0.36  1222 1.00
## beta[7] -0.58    0.00 0.13 -0.84 -0.68 -0.58 -0.49 -0.33  1196 1.00
## sigma    0.34    0.00 0.04  0.27  0.31  0.34  0.36  0.43  1713 1.00
## lp__    26.33    0.08 2.14 21.22 25.19 26.64 27.87 29.51   748 1.00
##
## Samples were drawn using NUTS(diag_e) at Wed Jul  6 00:46:59 2016.
## 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).
##  The estimated Bayesian Fraction of Missing Information is a measure of
##  the efficiency of the sampler with values close to 1 being ideal.
##  For each chain, these estimates are
##  0.9 0.9 0.8 1
```

volume model

model

mesquite_volume.stan

```
data {
  int<lower=0> N;
  vector[N] weight;
  vector[N] diam1;
  vector[N] diam2;
  vector[N] canopy_height;
}
transformed data {
  vector[N] log_weight;
  vector[N] log_canopy_volume;
  log_weight        = log(weight);
  log_canopy_volume = log(diam1 .* diam2 .* canopy_height);
}
parameters {
  vector[2] beta;
  real<lower=0> sigma;
}
model {
  log_weight ~ normal(beta[1] +
                      beta[2] * log_canopy_volume,
                      sigma);
}
```

fit

```
### Volume model: log(weight) ~ log(canopy_volume)
# canopy_volume <- diam1 * diam2 * canopy_height
mesquite_volume.sf <- stan(file='mesquite_volume.stan', data=data.list,
                           iter=1000, chains=4)
print(mesquite_volume.sf)
```

```
## Inference for Stan model: mesquite_volume.
## 4 chains, each with iter=1000; warmup=500; thin=1;
## post-warmup draws per chain=500, total post-warmup draws=2000.
##
##          mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## beta[1]  5.17    0.00 0.08  5.00  5.11  5.16  5.22  5.33   952    1
## beta[2]  0.73    0.00 0.06  0.62  0.69  0.73  0.76  0.83   973    1
## sigma    0.43    0.00 0.05  0.35  0.40  0.42  0.46  0.53  1319    1
## lp__    16.11    0.04 1.28 12.74 15.54 16.47 17.00 17.55  1159    1
##
## Samples were drawn using NUTS(diag_e) at Wed Jul  6 00:47:02 2016.
## 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).
##  The estimated Bayesian Fraction of Missing Information is a measure of
##  the efficiency of the sampler with values close to 1 being ideal.
##  For each chain, these estimates are
##  1.1 0.9 1.2 0.9
```

volume, area and shape model

model

mesquite_vas.stan

```
data {
  int<lower=0> N;
  vector[N] weight;
  vector[N] diam1;
  vector[N] diam2;
  vector[N] canopy_height;
  vector[N] total_height;
  vector[N] density;
  vector[N] group;
}
transformed data {
  vector[N] log_weight;
  vector[N] log_canopy_volume;
  vector[N] log_canopy_area;
  vector[N] log_canopy_shape;
  vector[N] log_total_height;
  vector[N] log_density;
  log_weight        = log(weight);
  log_canopy_volume = log(diam1 .* diam2 .* canopy_height);
  log_canopy_area   = log(diam1 .* diam2);
  log_canopy_shape  = log(diam1 ./ diam2);
```

```
  log_total_height  = log(total_height);
  log_density       = log(density);
}
parameters {
  vector[7] beta;
  real<lower=0> sigma;
}
model {
  log_weight ~ normal(beta[1] +
                      beta[2] * log_canopy_volume +
                      beta[3] * log_canopy_area +
                      beta[4] * log_canopy_shape +
                      beta[5] * log_total_height +
                      beta[6] * log_density +
                      beta[7] * group,
                      sigma);
}
```

fit

```
### Volume, area & shape model:
# log(weight) ~ log(canopy.volume) + log(canopy.area) + log(canopy.shape)
#               + log(total.height) + log(density) + group
# canopy_volume <- diam1 * diam2 * canopy_height
# canopy_area   <- diam1 * diam2
# canopy_shape  <- diam1 / diam2

mesquite_vas.sf <- stan(file='mesquite_vas.stan', data=data.list,
                        iter=1000, chains=4)
print(mesquite_vas.sf)
```

```
## Inference for Stan model: mesquite_vas.
## 4 chains, each with iter=1000; warmup=500; thin=1;
## post-warmup draws per chain=500, total post-warmup draws=2000.
##
##          mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## beta[1]  5.36    0.01 0.18  5.01  5.25  5.36  5.47  5.73   731 1.00
## beta[2]  0.37    0.01 0.29 -0.19  0.18  0.37  0.57  0.93   771 1.00
## beta[3]  0.40    0.01 0.30 -0.20  0.20  0.40  0.60  0.99   808 1.00
## beta[4] -0.39    0.01 0.24 -0.89 -0.54 -0.38 -0.23  0.09   985 1.00
## beta[5]  0.39    0.01 0.32 -0.26  0.18  0.40  0.60  1.02   751 1.00
## beta[6]  0.12    0.00 0.13 -0.13  0.03  0.12  0.20  0.37   899 1.01
## beta[7] -0.59    0.00 0.13 -0.85 -0.67 -0.58 -0.50 -0.34   936 1.00
## sigma    0.34    0.00 0.04  0.28  0.31  0.34  0.36  0.43  1266 1.00
## lp__    26.38    0.10 2.16 21.04 25.20 26.74 27.99 29.46   463 1.00
##
## Samples were drawn using NUTS(diag_e) at Wed Jul  6 00:47:06 2016.
## 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).
##  The estimated Bayesian Fraction of Missing Information is a measure of
##  the efficiency of the sampler with values close to 1 being ideal.
##  For each chain, these estimates are
```

```
##  0.8 1 1 1
```

va model

model

mesquite_va.stan

```
data {
  int<lower=0> N;
  vector[N] weight;
  vector[N] diam1;
  vector[N] diam2;
  vector[N] canopy_height;
  vector[N] group;
}
transformed data {
  vector[N] log_weight;
  vector[N] log_canopy_volume;
  vector[N] log_canopy_area;
  log_weight        = log(weight);
  log_canopy_volume = log(diam1 .* diam2 .* canopy_height);
  log_canopy_area   = log(diam1 .* diam2);
}
parameters {
  vector[4] beta;
  real<lower=0> sigma;
}
model {
  log_weight ~ normal(beta[1] +
                      beta[2] * log_canopy_volume +
                      beta[3] * log_canopy_area +
                      beta[4] * group,
                      sigma);
}
```

fit

```
### Last two models
# log(weight) ~ log(canopy_volume) + log(canopy_area) + group
mesquite_va.sf <- stan(file='mesquite_va.stan', data=data.list,
                       iter=1000, chains=4)
print(mesquite_va.sf)
```

```
## Inference for Stan model: mesquite_va.
## 4 chains, each with iter=1000; warmup=500; thin=1;
## post-warmup draws per chain=500, total post-warmup draws=2000.
##
##          mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## beta[1]  5.22    0.00 0.09  5.05  5.16  5.22  5.28  5.39  1224    1
## beta[2]  0.62    0.01 0.19  0.23  0.50  0.62  0.75  0.97   760    1
## beta[3]  0.29    0.01 0.24 -0.16  0.13  0.28  0.44  0.76   774    1
```

```
## beta[4] -0.53    0.00 0.12 -0.75 -0.61 -0.53 -0.45 -0.29  1324    1
## sigma    0.35    0.00 0.04  0.28  0.32  0.34  0.37  0.43  1314    1
## lp__    25.39    0.06 1.66 21.19 24.61 25.75 26.59 27.55   776    1
##
## Samples were drawn using NUTS(diag_e) at Wed Jul  6 00:47:10 2016.
## 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).
##  The estimated Bayesian Fraction of Missing Information is a measure of
##  the efficiency of the sampler with values close to 1 being ideal.
##  For each chain, these estimates are
##  1 0.9 1.1 0.9
```

vash model

model

mesquite_vash.stan

```
data {
  int<lower=0> N;
  vector[N] weight;
  vector[N] diam1;
  vector[N] diam2;
  vector[N] canopy_height;
  vector[N] total_height;
  vector[N] group;
}
transformed data {
  vector[N] log_weight;
  vector[N] log_canopy_volume;
  vector[N] log_canopy_area;
  vector[N] log_canopy_shape;
  vector[N] log_total_height;
  log_weight        = log(weight);
  log_canopy_volume = log(diam1 .* diam2 .* canopy_height);
  log_canopy_area   = log(diam1 .* diam2);
  log_canopy_shape  = log(diam1 ./ diam2);
  log_total_height  = log(total_height);
}
parameters {
  vector[6] beta;
  real<lower=0> sigma;
}
model {
  log_weight ~ normal(beta[1] +
                      beta[2] * log_canopy_volume +
                      beta[3] * log_canopy_area +
                      beta[4] * log_canopy_shape +
                      beta[5] * log_total_height +
                      beta[6] * group,
                      sigma);
}
```

fit

```
# log(weight) ~ log(canopy_volume) + log(canopy_area) + log(canopy_shape)
#               + log(total_height) + group
mesquite_vash.sf <- stan(file='mesquite_vash.stan', data=data.list,
                         iter=1000, chains=4)
print(mesquite_vash.sf)
```

```
## Inference for Stan model: mesquite_vash.
## 4 chains, each with iter=1000; warmup=500; thin=1;
## post-warmup draws per chain=500, total post-warmup draws=2000.
##
##          mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
## beta[1]  5.32    0.01 0.17  4.98  5.21  5.32  5.43  5.65   805 1.00
## beta[2]  0.39    0.01 0.30 -0.19  0.20  0.39  0.60  1.00   674 1.00
## beta[3]  0.40    0.01 0.32 -0.25  0.20  0.40  0.61  1.00   756 1.00
## beta[4] -0.33    0.01 0.22 -0.75 -0.47 -0.33 -0.18  0.10  1048 1.00
## beta[5]  0.42    0.01 0.34 -0.27  0.20  0.43  0.66  1.06  1201 1.00
## beta[6] -0.55    0.00 0.12 -0.78 -0.63 -0.55 -0.47 -0.29  1201 1.00
## sigma    0.34    0.00 0.04  0.27  0.31  0.34  0.37  0.44  1244 1.01
## lp__    26.40    0.08 2.06 21.19 25.33 26.80 27.87 29.33   693 1.00
##
## Samples were drawn using NUTS(diag_e) at Wed Jul  6 00:47:15 2016.
## 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).
##  The estimated Bayesian Fraction of Missing Information is a measure of
##  the efficiency of the sampler with values close to 1 being ideal.
##  For each chain, these estimates are
##  0.7 0.8 1 0.9
```