Logistic regression in Stan

Read in the same hiring data

```
library(readr)
#hiredata <- read.csv("~/Dropbox/My Documents/Teaching/772 Statistics/772 Spring 2020/Classes/Week 9-Lo
hiredata <- read_csv("~/Google Drive crowston@syr.edu/Courses/IST 772 Crowston/Week 9/Week9hiringData.c
```

```
## Parsed with column specification:
## cols(
##   row = col_double(),
##   hired = col_double(),
##   rater = col_character(),
##   recommend = col_double(),
##   vision = col_double(),
##   issues = col_double(),
##   trends = col_double(),
##   consult = col_double(),
##   lead = col_double(),
##   collab = col_double()
## )
```

```
hiredata$recInv <- 4 - hiredata$recommend
```

Bayesian logistic regression with the MCMC package

```
#install.packages("MCMCpack")    # Download MCMCpack package
library(MCMCpack) # Load the package
```

```
## Loading required package: coda
```

```
## Loading required package: MASS
```

```
## ##
## ## Markov Chain Monte Carlo Package (MCMCpack)
```

```
## ## Copyright (C) 2003-2020 Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park
```

```
## ##
## ## Support provided by the U.S. National Science Foundation
```

```
## ## (Grants SES-0350646 and SES-0350613)
## ##
```

Logistic regression using MCMClogit.

```
bayesLogitOut <- MCMClogit(formula = hired ~ recInv + vision, data = hiredata)
```

Examine the results.

```
summary(bayesLogitOut)
```

```
##
## Iterations = 1001:11000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                Mean     SD Naive SE Time-series SE
## (Intercept) -2.7353 0.8843 0.008843       0.030465
## recInv       1.1701 0.2618 0.002618       0.008992
## vision      -0.6224 0.2857 0.002857       0.009739
##
## 2. Quantiles for each variable:
##
##                2.5%     25%     50%     75%    97.5%
## (Intercept) -4.5287 -3.3336 -2.7372 -2.1157 -1.01825
## recInv       0.6595  0.9914  1.1691  1.3473  1.69961
## vision      -1.1954 -0.8149 -0.6164 -0.4229 -0.06194
```

Logistic regression in Stan.

Logistic regression works the same way as regular regression but with a different link function, which means a different distribution for y in the model. We'll again use the QR decomposition of the x data for efficiency.

```
data {
  int<lower=0> N;   // number of data items
  int<lower=0> K;   // number of predictors
  matrix[N, K] x;   // predictor matrix
  int<lower=0,upper=1> y[N];      // outcome vector
}

transformed data {
  matrix[N, K] Q_ast;
  matrix[K, K] R_ast;
  matrix[K, K] R_ast_inverse;

  // thin and scale the QR decomposition
  Q_ast = qr_Q(x)[, 1:K] * sqrt(N - 1);
  R_ast = qr_R(x)[1:K, ] / sqrt(N - 1);
  R_ast_inverse = inverse(R_ast);
}

parameters {
  real mu; // intercept
  vector[K] theta; // coefficients on Q_ast
}

model {
  // likelihood
  y ~ bernoulli_logit(Q_ast * theta + mu);
}
```

```
generated quantities {
  vector[K] beta;

  beta = R_ast_inverse * theta; // coefficients on x
}
```

Put data in a list for Stan.

```r
fit_data<-list(N=length(hiredata$hired), y=hiredata$hired,
               K=2, x=subset(hiredata, select=c(recInv, vision)))
```

```r
#install.packages("rstan")
library(rstan)
```

```
## Loading required package: StanHeaders

## Loading required package: ggplot2

## rstan (Version 2.19.3, GitRev: 2e1f913d3ca3)

## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)

##
## Attaching package: 'rstan'

## The following object is masked from 'package:coda':
##
##     traceplot
```

```r
options(mc.cores = parallel::detectCores())

logistic_qr_fit <- sampling(logistic_qr_model, data=fit_data)
```

Examine the results.

```r
summary(logistic_qr_fit)
```

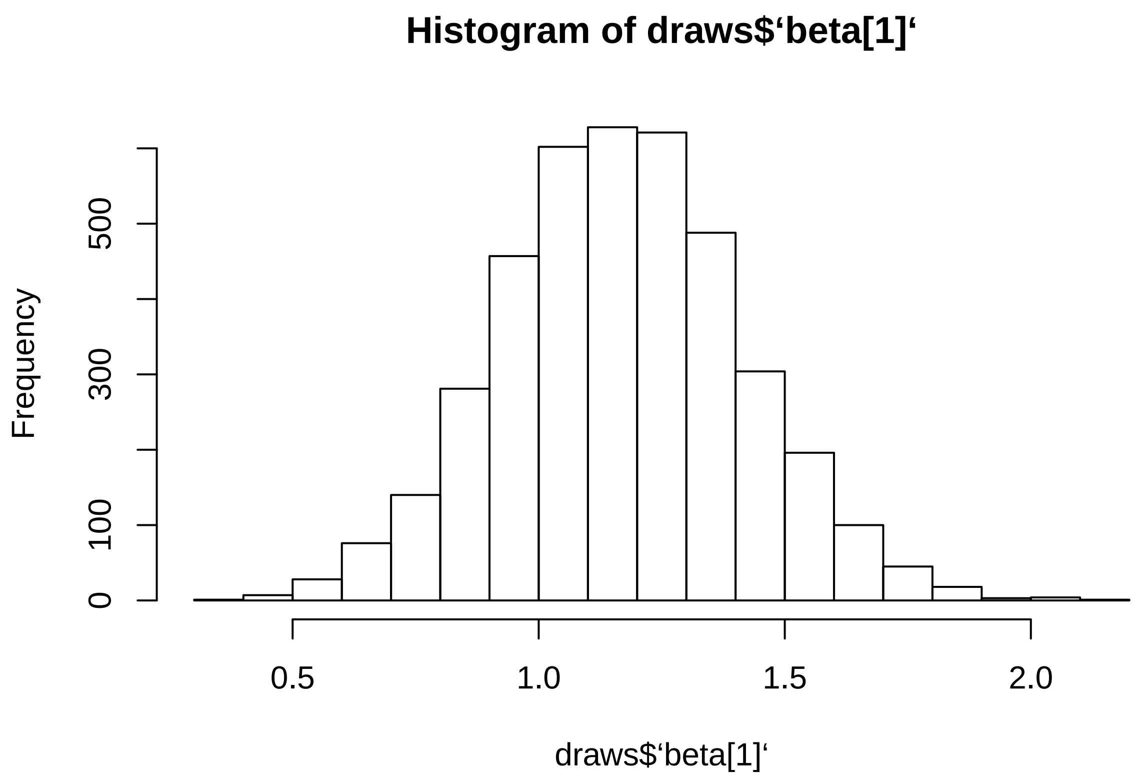
```
## $summary
##                   mean     se_mean        sd          2.5%          25%
## mu          -2.7456639 0.026918368 0.8238033   -4.33277485   -3.2978281
## theta[1]     1.6304963 0.025430441 0.7741085    0.06099423    1.1252016
## theta[2]    -0.6835672 0.009380055 0.3101270   -1.31732584   -0.8878538
## beta[1]      1.1682562 0.007164916 0.2454563    0.69363172    1.0020117
## beta[2]     -0.6058368 0.008313421 0.2748616   -1.16752880   -0.7868933
## lp__      -142.0759729 0.035279704 1.1752513 -145.15266490 -142.6290881
##                    50%          75%        97.5%     n_eff      Rhat
## mu          -2.7457903   -2.1958508   -1.12213844  936.5902 1.004556
## theta[1]     1.6353606    2.1582850    3.10501653  926.6076 1.004316
## theta[2]    -0.6825035   -0.4638465   -0.09988943 1093.1217 1.003262
## beta[1]      1.1672482    1.3290030    1.66575050 1173.6160 1.003738
## beta[2]     -0.6048940   -0.4111011   -0.08853071 1093.1217 1.003262
## lp__      -141.7824268 -141.1967673 -140.71218716 1109.7155 1.005158
##
## $c_summary
## , , chains = chain:1
```

```
##
##           stats
## parameter          mean        sd          2.5%         25%          50%
##   mu         -2.8314546 0.8459039   -4.33277485   -3.4688578   -2.8513576
##   theta[1]    1.7107592 0.7946229    0.07124628    1.1694672    1.7227897
##   theta[2]   -0.6696969 0.3235188   -1.39971578   -0.8781362   -0.6580601
##   beta[1]     1.1969824 0.2493057    0.72438559    1.0226597    1.1863670
##   beta[2]    -0.5935436 0.2867305   -1.24054994   -0.7782807   -0.5832301
##   lp__     -142.1692197 1.2383248 -145.49590324 -142.7762421 -141.8394151
##           stats
## parameter           75%         97.5%
##   mu         -2.2487579   -1.09008405
##   theta[1]    2.3169374    3.08563001
##   theta[2]   -0.4441776   -0.09797500
##   beta[1]     1.3682715    1.66959239
##   beta[2]    -0.3936689   -0.08683397
##   lp__     -141.2400193 -140.74076321
##
## , , chains = chain:2
##
##           stats
## parameter          mean        sd          2.5%         25%          50%
##   mu         -2.8000155 0.8630518   -4.47706078   -3.3480099   -2.8046535
##   theta[1]    1.6789206 0.8113582    0.01816738    1.1657969    1.6721837
##   theta[2]   -0.6551390 0.3229010   -1.28434415   -0.8750231   -0.6412752
##   beta[1]     1.1734371 0.2527242    0.68555163    1.0123224    1.1757882
##   beta[2]    -0.5806412 0.2861830   -1.13829756   -0.7755216   -0.5683539
##   lp__     -142.1344120 1.2230035 -145.34580893 -142.6918537 -141.8631489
##           stats
## parameter           75%         97.5%
##   mu         -2.2571628   -1.08383542
##   theta[1]    2.1864398    3.35901495
##   theta[2]   -0.4408885   -0.05026263
##   beta[1]     1.3234979    1.71610815
##   beta[2]    -0.3907538   -0.04454712
##   lp__     -141.2016047 -140.69886567
##
## , , chains = chain:3
##
##           stats
## parameter          mean        sd         2.5%         25%          50%
##   mu         -2.6854396 0.7487547   -4.1787982   -3.1714545   -2.6901813
##   theta[1]    1.5746066 0.7001598    0.2348902    1.1298486    1.5928393
##   theta[2]   -0.7125820 0.2805337   -1.2889135   -0.8913934   -0.7078888
##   beta[1]     1.1599771 0.2350662    0.7052734    1.0023844    1.1641970
##   beta[2]    -0.6315522 0.2486334   -1.1423473   -0.7900304   -0.6273927
##   lp__     -141.9542082 1.1289180 -144.8275925 -142.4520482 -141.6480668
##           stats
## parameter           75%        97.5%
##   mu         -2.1950198   -1.2588518
##   theta[1]    2.0493108    2.9610278
##   theta[2]   -0.5307490   -0.1709346
##   beta[1]     1.3101047    1.6241189
##   beta[2]    -0.4703959   -0.1514971
```

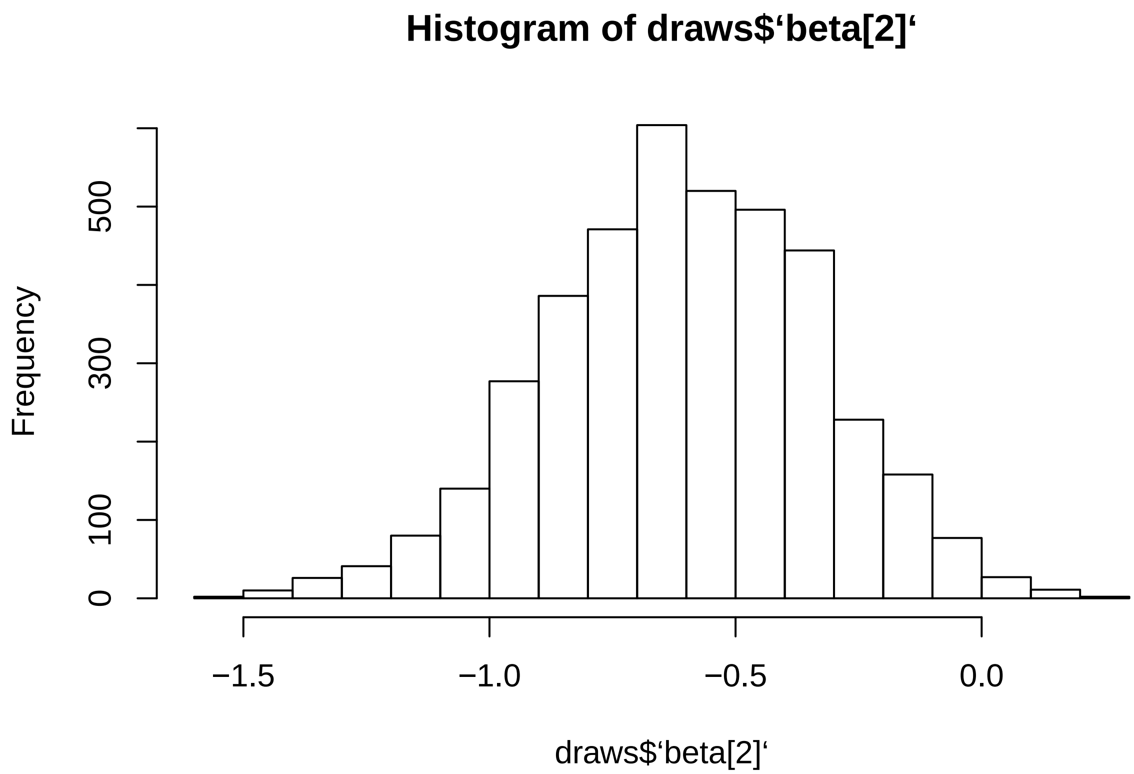
```
##   lp__      -141.1331802 -140.6862179
##
## , , chains = chain:4
##
##            stats
## parameter          mean        sd          2.5%         25%         50%
##   mu         -2.6657459 0.8217371   -4.20805824   -3.2010988   -2.6871928
##   theta[1]    1.5576988 0.7756264    0.03119257    1.0117644    1.5778866
##   theta[2]   -0.6968511 0.3087951   -1.32526697   -0.9164627   -0.6994418
##   beta[1]     1.1426280 0.2414914    0.68392410    0.9772661    1.1390312
##   beta[2]    -0.6176101 0.2736811   -1.17456692   -0.8122490   -0.6199062
##   lp__      -142.0460518 1.0943074 -144.84044931 -142.6165795 -141.7943868
##            stats
## parameter           75%        97.5%
##   mu         -2.1063378   -1.0370654
##   theta[1]    2.0720042    3.0533636
##   theta[2]   -0.4613992   -0.1222515
##   beta[1]     1.3092530    1.6077933
##   beta[2]    -0.4089321   -0.1083499
##   lp__      -141.2249363 -140.7403300
```

Extract and plot the coefficients.

```r
draws<-as.data.frame(logistic_qr_fit)
hist(draws$`beta[1]`)
```




```r
hist(draws$`beta[2]`)
```



Logistic regression using brms

The brms library can also perform logistic regression by specifying the link function, just as with glm. However, with brm and binary outcomes, the recommended link is Bernoulli rather than binomial (as in the Stan model).

```
library(brms)
```

```
## Loading required package: Rcpp

## Loading 'brms' package (version 2.11.1). Useful instructions
## can be found by typing help('brms'). A more detailed introduction
## to the package is available through vignette('brms_overview').

##
## Attaching package: 'brms'

## The following object is masked from 'package:rstan':
##
##     loo

## The following objects are masked from 'package:MCMCpack':
##
##     ddirichlet, rdirichlet
```

```
brm_fit<-brm(hired ~ recInv + vision, data = hiredata, family=bernoulli(link="logit"), file="hire_brm")
```

Examine the results.

```
summary(brm_fit)
```

```
##  Family: bernoulli
```

```
##   Links: mu = logit
## Formula: hired ~ recInv + vision
##    Data: hiredata (Number of observations: 295)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept    -2.75      0.85    -4.46    -1.12 1.00     2285     2009
## recInv        1.17      0.26     0.68     1.69 1.00     2530     2228
## vision       -0.60      0.27    -1.15    -0.09 1.00     2908     2551
##
## Samples were drawn 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).
```