0.1 `logit.bayes`: Bayesian Logistic Regression

Logistic regression specifies a dichotomous dependent variable as a function of a set of explanatory variables using a random walk Metropolis algorithm. For a maximum likelihood implementation, see Section **??**.

Syntax

```
> z.out <- zelig(Y ~ X1 + X2, model = "logit.bayes", data = mydata)
> x.out <- setx(z.out)
> s.out <- sim(z.out, x = x.out)
```

Additional Inputs

Use the following arguments to monitor the Markov chain:

- `burnin`: number of the initial MCMC iterations to be discarded (defaults to 1,000).
- `mcmc`: number of the MCMC iterations after burnin (defaults to 10,000).
- `thin`: thinning interval for the Markov chain. Only every `thin`-th draw from the Markov chain is kept. The value of `mcmc` must be divisible by this value. The default value is 1.
- `tune`: Metropolis tuning parameter, either a positive scalar or a vector of length k, where k is the number of coefficients. The tuning parameter should be set such that the acceptance rate of the Metropolis algorithm is satisfactory (typically between 0.20 and 0.5) before using the posterior density for inference. The default value is 1.1.
- `verbose`: defaults to `FALSE`. If `TRUE`, the progress of the sampler (every 10%) is printed to the screen.
- `seed`: seed for the random number generator. The default is `NA` which corresponds to a random seed of 12345.
- `beta.start`: starting values for the Markov chain, either a scalar or vector with length equal to the number of estimated coefficients. The default is `NA`, such that the maximum likelihood estimates are used as the starting values.

Use the following parameters to specify the model's priors:

- `b0`: prior mean for the coefficients, either a numeric vector or a scalar. If a scalar value, that value will be the prior mean for all the coefficients. The default is 0.
- `B0`: prior precision parameter for the coefficients, either a square matrix (with the dimensions equal to the number of coefficients) or a scalar. If a scalar value, that value times an identity matrix will be the prior precision parameter. The default is 0, which leads to an improper prior.

Zelig users may wish to refer to `help(logit.bayes)` for more information.

Convergence

Users should verify that the Markov Chain converges to its stationary distribution. After running the `zelig()` function but before performing `setx()`, users may conduct the following convergence diagnostics tests:

- `geweke.diag(z.out$coefficients)`: The Geweke diagnostic tests the null hypothesis that the Markov chain is in the stationary distribution and produces z-statistics for each estimated parameter.
- `heidel.diag(z.out$coefficients)`: The Heidelberger-Welch diagnostic first tests the null hypothesis that the Markov Chain is in the stationary distribution and produces p-values for each estimated parameter. Calling `heidel.diag()` also produces output that indicates whether the mean of a marginal posterior distribution can be estimated with sufficient precision, assuming that the Markov Chain is in the stationary distribution.
- `raftery.diag(z.out$coefficients)`: The Raftery diagnostic indicates how long the Markov Chain should run before considering draws from the marginal posterior distributions sufficiently representative of the stationary distribution.

If there is evidence of non-convergence, adjust the values for `burnin` and `mcmc` and rerun `zelig()`.

Advanced users may wish to refer to `help(geweke.diag)`, `help(heidel.diag)`, and `help(raftery.diag)` for more information about these diagnostics.

Examples

1. Basic Example
   Attaching the sample dataset:

   ```
   > data(turnout)
   ```

   Estimating the logistic regression using `logit.bayes`:

   ```
   > z.out <- zelig(vote ~ race + educate, model = "logit.bayes",
   +     data = turnout, verbose = TRUE)
   ```

   Convergence diagnostics before summarizing the estimates:

   ```
   > geweke.diag(z.out$coefficients)
   ```

   ```
   > heidel.diag(z.out$coefficients)
   ```

```
> raftery.diag(z.out$coefficients)

> summary(z.out)
```

Setting values for the explanatory variables to their sample averages:

```
> x.out <- setx(z.out)
```

Simulating quantities of interest from the posterior distribution given `x.out`.

```
> s.out1 <- sim(z.out, x = x.out)

> summary(s.out1)
```

2. Simulating First Differences
   Estimating the first difference (and risk ratio) in individual's probability of voting when education is set to be low (25th percentile) versus high (75th percentile) while all the other variables held at their default values.

```
> x.high <- setx(z.out, educate = quantile(turnout$educate, prob = 0.75))

> x.low <- setx(z.out, educate = quantile(turnout$educate, prob = 0.25))

> s.out2 <- sim(z.out, x = x.high, x1 = x.low)

> summary(s.out2)
```

Model

Let Y_i be the binary dependent variable for observation i which takes the value of either 0 or 1.

- The *stochastic component* is given by

$$\begin{aligned} Y_i &\sim \text{Bernoulli}(\pi_\text{i}) \\ &= \pi_i^{Y_i}(1-\pi_i)^{1-Y_i}, \end{aligned}$$

  where $\pi_i = \Pr(Y_i = 1)$.

- The *systematic component* is given by

$$\pi_i = \frac{1}{1+\exp(-x_i\beta)},$$

  where x_i is the vector of k explanatory variables for observation i and β is the vector of coefficients.

- The *prior* for β is given by

$$\beta \sim \text{Normal}_k\left(b_0, B_0^{-1}\right)$$

  where b_0 is the vector of means for the k explanatory variables and B_0 is the $k \times k$ precision matrix (the inverse of a variance-covariance matrix).

Quantities of Interest

- The expected values (`qi$ev`) for the logit model are simulations of the predicted probability of a success:

$$E(Y) = \pi_i = \frac{1}{1 + \exp(-x_i\beta)},$$

  given the posterior draws of β from the MCMC iterations.

- The predicted values (`qi$pr`) are draws from the Bernoulli distribution with mean equal to the simulated expected value π_i.

- The first difference (`qi$fd`) for the logit model is defined as

$$\text{FD} = \Pr(Y = 1 \mid X_1) - \Pr(Y = 1 \mid X).$$

- The risk ratio (`qi$rr`)is defined as

$$\text{RR} = \Pr(Y = 1 \mid X_1) \ / \ \Pr(Y = 1 \mid X).$$

- In conditional prediction models, the average expected treatment effect (`qi$att.ev`) for the treatment group is

$$\frac{1}{\sum t_i} \sum_{i:t_i=1} [Y_i(t_i = 1) - E[Y_i(t_i = 0)]],$$

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups.

- In conditional prediction models, the average predicted treatment effect (`qi$att.pr`) for the treatment group is

$$\frac{1}{\sum t_i} \sum_{i:t_i=1} [Y_i(t_i = 1) - \widehat{Y_i(t_i = 0)}],$$

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups.

Output Values

The output of each Zelig command contains useful information which you may view. For example, if you run

```
z.out <- zelig(y ~ x, model = "logit.bayes", data)
```

then you may examine the available information in `z.out` by using `names(z.out)`, see the draws from the posterior distribution of the `coefficients` by using `z.out$coefficients`, and a default summary of information through `summary(z.out)`. Other elements available through the `$` operator are listed below.

- From the `zelig()` output object `z.out`, you may extract:
  - `coefficients`: draws from the posterior distributions of the estimated parameters.
  - `zelig.data`: the input data frame if `save.data = TRUE`.
  - `seed`: the random seed used in the model.
- From the `sim()` output object `s.out`:
  - `qi$ev`: the simulated expected values(probabilities) for the specified values of `x`.
  - `qi$pr`: the simulated predicted values for the specified values of `x`.
  - `qi$fd`: the simulated first difference in the expected values for the values specified in `x` and `x1`.
  - `qi$rr`: the simulated risk ratio for the expected values simulated from `x` and `x1`.
  - `qi$att.ev`: the simulated average expected treatment effect for the treated from conditional prediction models.
  - `qi$att.pr`: the simulated average predicted treatment effect for the treated from conditional prediction models.

Contributors

Bayesian logistic regression function is part of the MCMCpack library by Andrew D. Martin and Kevin M. Quinn. If you use this model, please cite:

The convergence diagnostics are part of the CODA library by Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. These diagnostics should be cited as:

Ben Goodrich and Ying Lu enabled `logit.bayes` to work with Zelig.