Zelig Documentation

Release 5.0-1

The Zelig Team

October 30, 2014

Zelig is a framework with an easy-to-use program that can estimate, help interpret, and present the results of a large range of statistical methods. It literally is "everyone's statistical software" because Zelig uses (R) code from many researchers. We also hope it will become "everyone's statistical software" for applications, and we have designed it so that anyone can use it or add their methods to it. Zelig comes with detailed, self-contained documentation that minimizes startup costs for Zelig and R (with all methods described in exactly the same notation, syntax, and style), automates graphics and summaries for all models, and, with only three simple required commands, makes the power of R accessible for all users. Zelig also works well for teaching, and is designed so that scholars can use the same program with students that they use for their research. Zelig is built on a wide ranging ontology of statistical methods.

Zelig adds considerable infrastructure to improve the use of existing methods. It interfaces with a wide range of statistical models through a simple and common intelligible call structure. It generalizes the program Clarify (for Stata), which translates hard-to-interpret coefficients into quantities of interest; combines multiply imputed data sets (such as output from Amelia) to deal with missing data; automates bootstrapping for all models; uses sophisticated nonparametric matching commands which improve parametric procedures (via MatchIt); allows one-line commands to run analyses in all designated strata; automates the creation of replication data files so that you (or, if you wish, anyone else) can replicate the results of your analyses (hence satisfying the replication standard); makes it easy to evaluate counterfactuals (via WhatIf); and allows conditional population and superpopulation inferences. Zelig includes many specific methods, based on likelihood, frequentist, Bayesian, robust Bayesian, and nonparametric theories of inference. Developers make their R packages usable from Zelig by writing a few simple bridge functions.

For users, see the *Installation and Quickstart* guide and then a full PDF of the documentation here. Please also join our Zelig Google Group, where you can ask questions, report bugs, and help others.

For developers, to view the code-base, visit the source repository at https://github.com/IQSS/Zelig and for regular updates and release information be sure to follow us on twitter at @IQSS.

You can also find the

CHAPTER

ONE

INSTALLATION AND QUICKSTART

This guide is designed to get you up and running with the current *beta* release of Zelig (5.0-1).

1.1 Installing R and Zelig

Before using Zelig, you will need to download and install both the R statistical program and the Zelig package:

Installing R

To install R, go to http://www.r-project.org/ Select the `CRAN` option from the left-hand menu (CRAN is the Comprehensive R Archive Network where all files related to R can be found). Pick a CRAN mirror closest to your current geographic location (there are multiple mirrors of this database in various locations, selecting the one closest to you will be sure to maximize your the speed of your download). Follow the instructions for downloading R for Linux, Mac OS X, or Windows.

Installing Zelig

Zelig 5 is not available on `CRAN` yet.

Beta Release

Beta releases are updated with the latest fixes and newest experimental features, and generally reflect a copy currently being tested before submission to CRAN. To download this release, enter the following into an R console:

```
install.packages("Zelig", type = "source", repos = "http://r.iq.harvard.edu/")
```

Development Release

Development versions contain the latest code in-development. This means that the development version contains the latest code which may not be fully tested. To download this release:

```
# This installs devtools package, if not already installed
install.packages("devtools")
# This loads devtools
library(devtools)
# This downloads Zelig 5.0-1 from the IQSS Github repo
install_github('IQSS/Zelig')
```

If you have successfully installed the program, you will see a the following message: *"DONE (Zelig)"*.

1.2 Quickstart Guide

Now that we have successfully downloaded and installed Zelig, we will load the package and walk through am example. The scenario is a simple one: imagine you want to estimate the distance a car needs to stop given its speed and you have a dataset of speed and stopping distances of cars. Throughout the rest of this guide, we will walk you through building a statistical model from this data using Zelig.

Loading Zelig

First, we have to load Zelig into R. After installing both R and Zelig, open R and type:

```
library(Zelig)
```

Building Models

Now, lets build a statistical model that captures the relationship a cars stopping distance and speed, where distance is the outcome (dependent) variable and speed is the only explanatory (independent) variable. The first decision we must make is what statistical model to test for a relationship between a cars speed and distance required for it to come to a full stop. To do this, we plot the two variables in our dataset to visually inspect any potential relationship:

```
# Scatterplot of car speed and distance required for full stop
plot(cars$speed, cars$dist, main = "Scatterplot of car speed and distance required for full stop", yl
# Fit regression line to data
abline(lm(cars$dist ~ cars$speed), col = "firebrick")
```

Also included in the scatter plot is a "best-fit" regression line that indicates a positive and linear relationship between our two variables. This basic test coupled with the fact that our outcome variable (distance) is continuous suggests that an appropriate model to use is least squares regression.

To fit this model to our data, we must first create Zelig least squares object, then specify our model, and finally regress distance on speed to estimate the relationship between speed and distance:

```
# load dataset (when you install R, example datasets are also installed)
data(cars)
# initialize Zelig5 least squares object
z5 <- zls$new()
# estimate ls model
z5$zelig(dist ~ speed, data = cars)
# you can now get model summary estimates
summary(z5)

## Model:
## $by
## [1] 1
##
##
## Call:
## stats::lm(formula = dist ~ speed, data = .)
##
## Coefficients:
## (Intercept)        speed
##     -17.579        3.932
##
## Next step: Use 'setx' method
```

So what do our model estimates tell us? First off, we can see that the positive 3.93 estimate for speed suggests a positive relationship between speed and distance a car needs to stop. That is, the faster a car is going, the longer the distance it needs to come to a full stop. In particular, we would interpret this coefficient as a one unit increase in speed

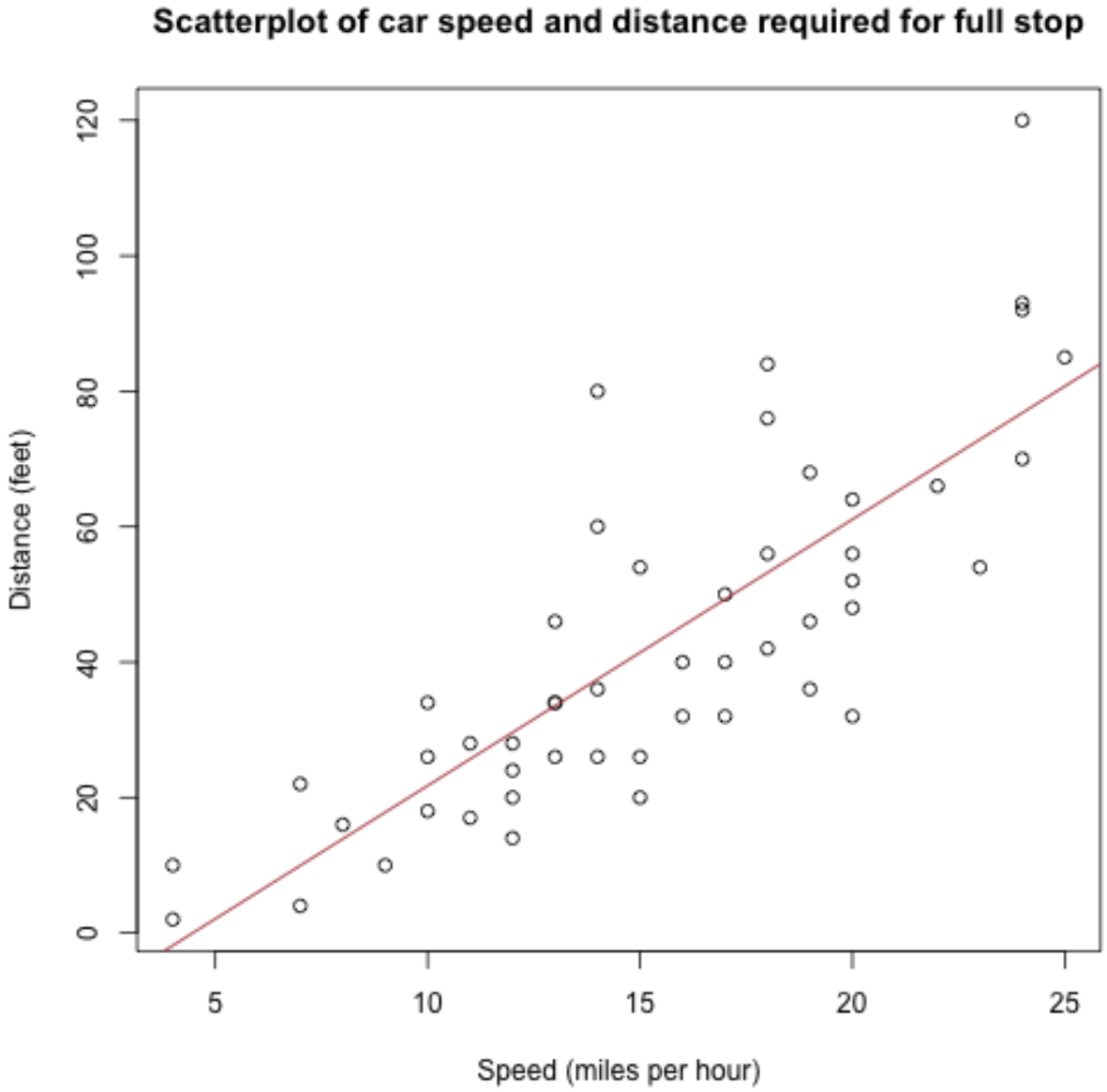


Figure 1.1: Scatterplot

(e.g., mph) leads to a 3 unit increase in distance (e.g., miles) needed for a car to stop. This interpretation is not very intuitive, however, and we might be interested in answering a particular question such as how much more distance does a car need to stop if it traveling 30 versus 50 miles per hour.

Zelig makes this simple, by automating the translation of model estimates in interpretable quantities of interest (more on this below) using Monte Carlo simulations. To get this process started we need to set explanatory variables in our model (i.e., speed) using the `$setx()` method:

```
# set speed to 30
z5$setx(speed = 30)

# set speed to 50
z5$setx1(speed = 50)
```

Now that we've set our variables, all we have to do is run our simulations:

```
# run simulations and estimate quantities of interest
z5$sim()
z5

##
##  sim x :
##  -----
## ev
##       mean      sd      50%     2.5%    97.5%
## 1 100.5217 6.26558 100.5612 88.24683 112.2449
## pv
##       mean      sd      50%     2.5%    97.5%
## 1 100.5217 6.26558 100.5612 88.24683 112.2449
##
##  sim x1 :
##  -----
## ev
##       mean       sd      50%     2.5%    97.5%
## 1 179.4012 14.10867 179.6203 152.0762 205.6916
## pv
##       mean       sd      50%     2.5%    97.5%
## 1 179.4012 14.10867 179.6203 152.0762 205.6916
## fd
##       mean       sd      50%     2.5%    97.5%
## 1 78.87952 8.056696 79.07112 63.53169 93.79235
```

Now we've estimated a model and calculated interpretable estimates at two speeds (30 versus 50 mph). What can we do with them? Zelig gives you access to estimated quantities of interest and makes plotting and presenting them particularly easy.

Quantities of Interest

As mentioned earlier, a major feature of Zelig is the translation of model estimates into easy to interpret quantities of interest (QIs). These QIs (e.g., expected and predicted values) can be accessed via the `$sim.out` field:

```
z5$sim.out

## $x
## Source: local data frame [1 x 2]
## Groups: <by row>
##
##              ev            pv
```

```
## 1 <dbl[1000,1]> <dbl[1000,1]>
##
## $x1
## Source: local data frame [1 x 3]
## Groups: <by row>
##
##              ev            pv            fd
## 1 <dbl[1000,1]> <dbl[1000,1]> <dbl[1000,1]>
```

Plots

A second major Zelig feature is how easy it is to plot QIs for presentation in slides or an article. Using the `plot()` function on the `z5$s.out` will produce ready-to-use plots with labels and confidence intervals.

Plots of QI's:

```
z5$graph()
```

Help

Finally, model documentation can be accessed using the `z5$help()` method after a model object has been initialized:

```
# documentation for least squares model
z5 <- zls$new()
z5$help()

# documentation for logistic regression
z5 <- zlogit$new()
z5$help()
```

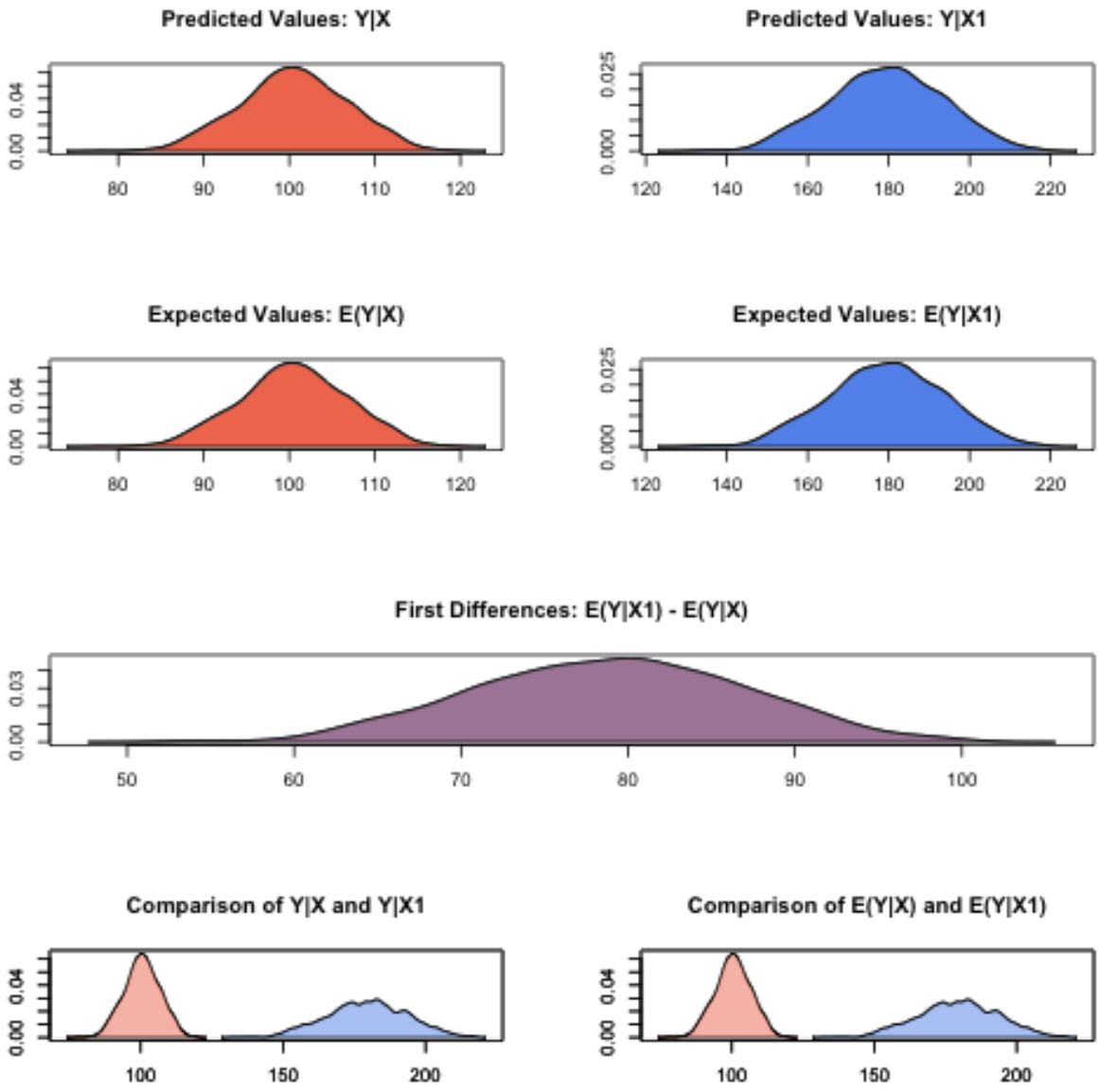


Figure 1.2: QIs

CHAPTER

TWO

MODEL REFERENCE AND VIGNETTES

This section includes technical information on the models currently implemented in Zelig (5.0-1). This includes a reference with a list of supported models as well as individual model vignettes with detailed information on the model, quantities of interest and syntax.

2.1 Reference

The following models are currently supported in Zelig 5.0-1:

- *Exponential Regression*: `zexp$new()`
- *Gamma Regression*: `zgamma()`
- *Logistic Regression*: `zlogit$new()`
- *Log Normal Regression*: `zlognorm$new()`
- *Least Squares Regression*: `zls$new()`
- *Negative Binomial Regression*: `zbinom$new()`
- *Normal Regression*: `znormal$new()`
- *Poisson Regression*: `zpoisson$new()`
- *Probit Regression*: `zprobit$new()`
- *Rare Events Logistic Regression*: `zrelogit$new()`
- *Tobit Regression*: `ztobit$new()`
- *Bayesian Factor Analysis*: `zfactorbayes$new()`
- *Bayesian Multinomial Logistic Regression*: `zmlogitbayes$new()`
- *Bayesian Ordered Probit Regression*: `zoprobitbayes$new()`
- *Bayesian Poisson Regression*: `zpoissonbayes$new()`
- *Bayesian Probit Regression*: `zprobitbayes$new()`
- *Bayesian Tobit Regression*: `ztobitbayes$new()`
- *Generalized Estimating Equation for Gamma Regression*: `zgammagee$new()`

2.2 zelig-exp

Exponential Regression for Duration Dependent Variables

Use the exponential duration regression model if you have a dependent variable representing a duration (time until an event). The model assumes a constant hazard rate for all events. The dependent variable may be censored (for observations have not yet been completed when data were collected).

2.2.1 Syntax

With reference classes:

```
z5 <- zexp$new()
z5$zelig(Surv(Y, C) ~ X, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

```
z.out <- zelig(Surv(Y, C) ~ X, model = "exp", data = mydata)
x.out <- setx(z.out)
s.out <- sim(z.out, x = x.out)
```

Exponential models require that the dependent variable be in the form Surv(Y, C), where Y and C are vectors of length n. For each observation i in 1, ..., n, the value y_i is the duration (lifetime, for example), and the associated c_i is a binary variable such that $c_i = 1$ if the duration is not censored (*e.g.*, the subject dies during the study) or $c_i = 0$ if the duration is censored (*e.g.*, the subject is still alive at the end of the study and is know to live at least as long as y_i). If c_i is omitted, all Y are assumed to be completed; that is, time defaults to 1 for all observations.

2.2.2 Input Values

In addition to the standard inputs, zelig() takes the following additional options for exponential regression:

- robust: defaults to FALSE. If TRUE, zelig() computes robust standard errors based on sandwich estimators (see and) and the options selected in cluster.
- cluster: if robust = TRUE, you may select a variable to define groups of correlated observations. Let x3 be a variable that consists of either discrete numeric values, character strings, or factors that define strata. Then

```
z.out <- zelig(y ~ x1 + x2, robust = TRUE, cluster = "x3",
                        model = "exp", data = mydata)
```

means that the observations can be correlated within the strata defined by the variable x3, and that robust standard errors should be calculated according to those clusters. If robust = TRUE but cluster is not specified, zelig() assumes that each observation falls into its own cluster.

2.2.3 Example

Attach the sample data:

```
data(coalition)
```

Estimate the model:

```
z.out <- zelig(Surv(duration, ciep12) ~ fract + numst2, model = "exp", data = coalition)

## How to cite this model in Zelig:
##   Olivia Lau, Kosuke Imai, Gary King. 2011.
##   exp: Exponential Regression for Duration Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

View the regression output:

```
summary(z.out)

## Model:
## $by
## [1] 1
##
## Call:
## survival::survreg(formula = Surv(duration, ciep12) ~ fract +
##     numst2, data = ., dist = "exponential", model = FALSE)
##
## Coefficients:
##  (Intercept)        fract       numst2
##  5.535872596 -0.003908965  0.461179302
##
## Scale fixed at 1
##
## Loglik(model)= -1077.4   Loglik(intercept only)= -1100.7
##  Chisq= 46.66 on 2 degrees of freedom, p= 7.4e-11
## n= 314
## Next step: Use 'setx' method
```

Set the baseline values (with the ruling coalition in the minority) and the alternative values (with the ruling coalition in the majority) for X:

```
x.low <- setx(z.out, numst2 = 0)

## Error in terms(lm(formula, data), "predvars"): error in evaluating the argument 'x' in selecting a
## Calls: lm -> lm.fit -> Ops.Surv

x.high <- setx(z.out, numst2 = 1)

## Error in terms(lm(formula, data), "predvars"): error in evaluating the argument 'x' in selecting a
## Calls: lm -> lm.fit -> Ops.Surv
```

Simulate expected values and first differences:

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

Summarize quantities of interest and produce some plots:

```
summary(s.out)

##
##  sim x :
##  -----
## ev
##       mean        sd      50%     2.5%    97.5%
## 1 4.599351 0.1804811 4.601767 4.246964 4.961869
## pv
```

```
##       mean        sd      50%     2.5%    97.5%
## 1 4.599351 0.1804811 4.601767 4.246964 4.961869
##
##  sim x1 :
##  -----
## ev
##       mean       sd      50%     2.5%    97.5%
## 1 5.427579 0.191717 5.434867 5.057529 5.816774
## pv
##       mean       sd      50%     2.5%    97.5%
## 1 5.427579 0.191717 5.434867 5.057529 5.816774
## fd
##        mean        sd       50%      2.5%    97.5%
## 1 0.8282286 0.2208927 0.8332688 0.4155774 1.268978
```

```
plot(s.out)
```

2.2.4 Model

Let Y_i^* be the survival time for observation i. This variable might be censored for some observations at a fixed time y_c such that the fully observed dependent variable, Y_i, is defined as

$$Y_i = \begin{cases} Y_i^* & \text{if } Y_i^* \leq y_c \\ y_c & \text{if } Y_i^* > y_c \end{cases}$$

- The *stochastic component* is described by the distribution of the partially observed variable Y^*. We assume Y_i^* follows the exponential distribution whose density function is given by

  $$f(y_i^* \mid \lambda_i) = \frac{1}{\lambda_i} \exp\left(-\frac{y_i^*}{\lambda_i}\right)$$

  for $y_i^* \geq 0$ and $\lambda_i > 0$. The mean of this distribution is λ_i.

  In addition, survival models like the exponential have three additional properties. The hazard function $h(t)$ measures the probability of not surviving past time t given survival up to t. In general, the hazard function is equal to $f(t)/S(t)$ where the survival function $S(t) = 1 - \int_0^t f(s)ds$ represents the fraction still surviving at time t. The cumulative hazard function $H(t)$ describes the probability of dying before time t. In general, $H(t) = \int_0^t h(s)ds = -\log S(t)$. In the case of the exponential model,

  $$\begin{aligned} h(t) &= \frac{1}{\lambda_i} \\ S(t) &= \exp\left(-\frac{t}{\lambda_i}\right) \\ H(t) &= \frac{t}{\lambda_i} \end{aligned}$$

  For the exponential model, the hazard function $h(t)$ is constant over time. The Weibull model and lognormal models allow the hazard function to vary as a function of elapsed time (see and respectively).

- The *systematic component* λ_i is modeled as

  $$\lambda_i = \exp(x_i\beta),$$

  where x_i is the vector of explanatory variables, and β is the vector of coefficients.

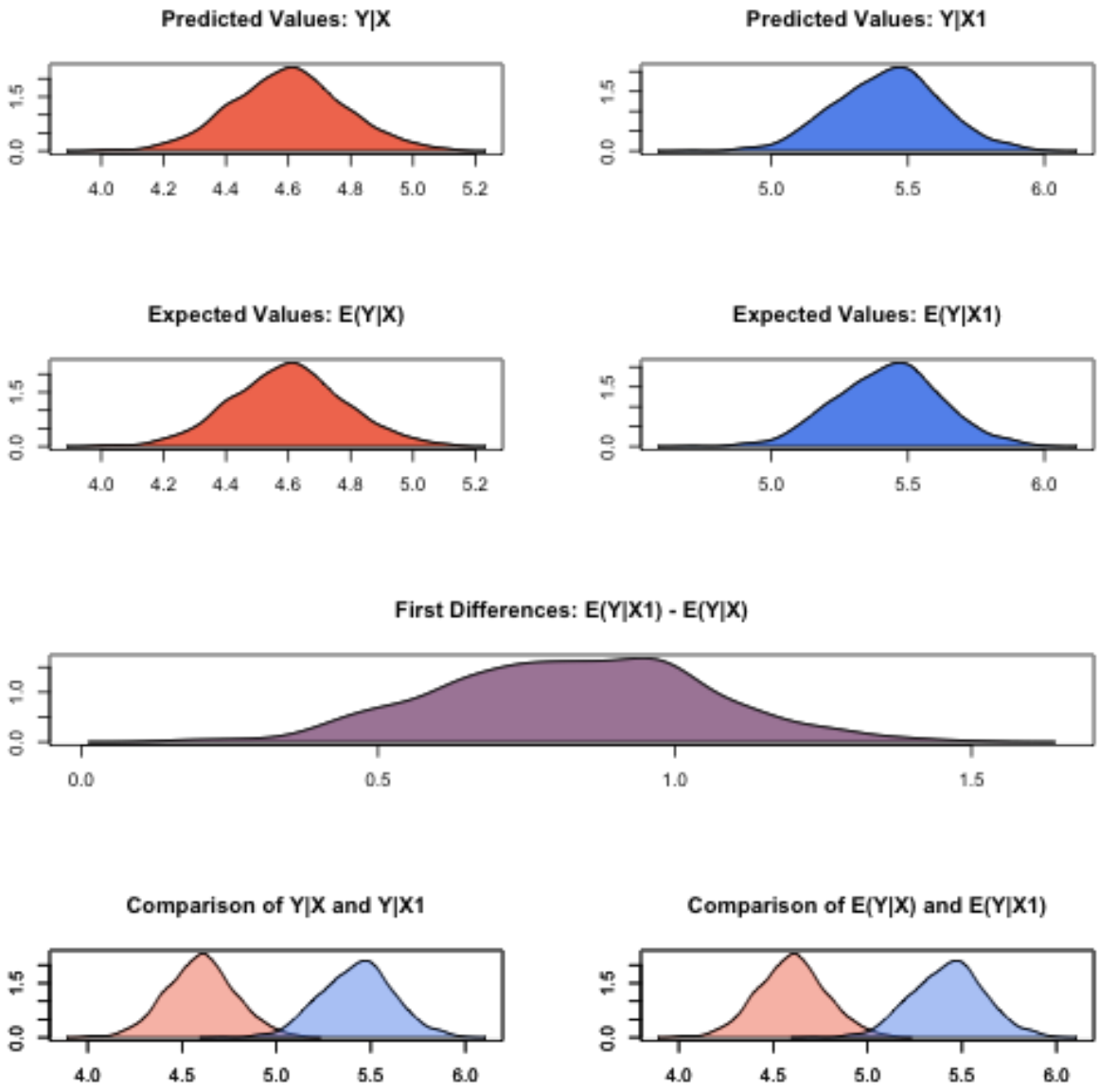


Figure 2.1: Zelig-exp

2.2.5 Quantities of Interest

- The expected values (qi$ev) for the exponential model are simulations of the expected duration given x_i and draws of β from its posterior,

$$E(Y) = \lambda_i = \exp(x_i\beta).$$

- The predicted values (qi$pr) are draws from the exponential distribution with rate equal to the expected value.
- The first difference (or difference in expected values, qi$ev.diff), is

$$\text{FD} \;=\; E(Y \mid x_1) - E(Y \mid x),$$

  where x and x_1 are different vectors of values for the explanatory variables.
- In conditional prediction models, the average expected treatment effect (att.ev) for the treatment group is

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

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. When $Y_i(t_i = 1)$ is censored rather than observed, we replace it with a simulation from the model given available knowledge of the censoring process. Variation in the simulations is due to two factors: uncertainty in the imputation process for censored y_i^* and uncertainty in simulating $E[Y_i(t_i = 0)]$, the counterfactual expected value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.
- In conditional prediction models, the average predicted treatment effect (att.pr) for the treatment group is

$$\frac{1}{\sum_{i=1}^{n} t_i} \sum_{i:t_i=1}^{n} \left\{ Y_i(t_i = 1) - Y_i(\widehat{t_i = 0}) \right\},$$

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. When $Y_i(t_i = 1)$ is censored rather than observed, we replace it with a simulation from the model given available knowledge of the censoring process. Variation in the simulations is due to two factors: uncertainty in the imputation process for censored y_i^* and uncertainty in simulating $Y_i(\widehat{t_i = 0})$, the counterfactual predicted value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

2.2.6 Output Values

The output of each Zelig command contains useful information which you may view. For example, if you run `z.out <- zelig(Surv(Y, C) ~ X, model = exp, data)`, then you may examine the available information in `z.out` by using `names(z.out)`, see the coefficients by using z.out$coefficients, and a default summary of information through `summary(z.out)`.

2.2.7 See also

The exponential function is part of the survival library by Terry Therneau, ported to R by Thomas Lumley. Advanced users may wish to refer to `help(survfit)` in the survival library.

2.3 zelig-gamma

Gamma Regression for Continuous, Positive Dependent Variables

Use the gamma regression model if you have a positive-valued dependent variable such as the number of years a parliamentary cabinet endures, or the seconds you can stay airborne while jumping. The gamma distribution assumes that all waiting times are complete by the end of the study (censoring is not allowed).

2.3.1 Syntax

With reference classes:

```
z5 <- zgamma$new()
z5$zelig(Y ~ X1 + X ~ X, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

```
z.out <- zelig(Y ~ X1 + X2, model = "gamma", data = mydata)
x.out <- setx(z.out)
s.out <- sim(z.out, x = x.out, x1 = NULL)
```

2.3.2 Example

Attach the sample data:

```
data(coalition)
```

Estimate the model:

```
z.out <- zelig(duration ~ fract + numst2, model = "gamma", data = coalition)
```

```
## How to cite this model in Zelig:
##   Kosuke Imai, Gary King, Olivia Lau. 2007.
##   gamma: Gamma Regression for Continuous, Positive Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

View the regression output:

```
summary(z.out)
```

```
## Model:
## $by
## [1] 1
##
##
## Call:  stats::glm(formula = duration ~ fract + numst2, family = Gamma("inverse"),
##     data = .)
##
## Coefficients:
## (Intercept)        fract       numst2
##  -0.0129597    0.0001149   -0.0173875
##
## Degrees of Freedom: 313 Total (i.e. Null);  311 Residual
```

```
## Null Deviance:       300.7
## Residual Deviance: 272.2         AIC: 2428
## Next step: Use 'setx' method
```

Set the baseline values (with the ruling coalition in the minority) and the alternative values (with the ruling coalition in the majority) for X:

```
x.low <- setx(z.out, numst2 = 0)
x.high <- setx(z.out, numst2 = 1)
```

Simulate expected values (qi$ev) and first differences (qi$fd):

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

summary(s.out)

##
##  sim x :
##  -----
## ev
##          mean       sd      50%     2.5%    97.5%
## [1,] 14.43674 1.093498 14.33691 12.57224 17.01294
## pv
##         mean       sd      50%      2.5%    97.5%
## [1,] 14.6374 12.91882 11.19025 0.6425469 48.15589
##
##  sim x1 :
##  -----
## ev
##          mean      sd     50%     2.5%    97.5%
## [1,] 19.20518 1.14026 19.1172 17.25049 21.60574
## pv
##          mean       sd     50%     2.5%    97.5%
## [1,] 18.62654 16.58418 13.7207 1.164515 64.19589
## fd
##         mean       sd     50%     2.5%    97.5%
## [1,] 4.76844 1.583298 4.72891 1.474829 7.913904

plot(s.out)
```

2.3.3 Model

- The Gamma distribution with scale parameter α has a *stochastic component*:

$$\begin{aligned} Y \sim & \quad \text{Gamma}(y_i \mid \lambda_i, \alpha) \\ f(y) = & \quad \frac{1}{\alpha^{\lambda_i} \Gamma \lambda_i} y_i^{\lambda_i - 1} \exp - \left\{ \frac{y_i}{\alpha} \right\} \end{aligned}$$

  for $\alpha, \lambda_i, y_i > 0$.

- The *systematic component* is given by

$$\lambda_i = \frac{1}{x_i \beta}$$

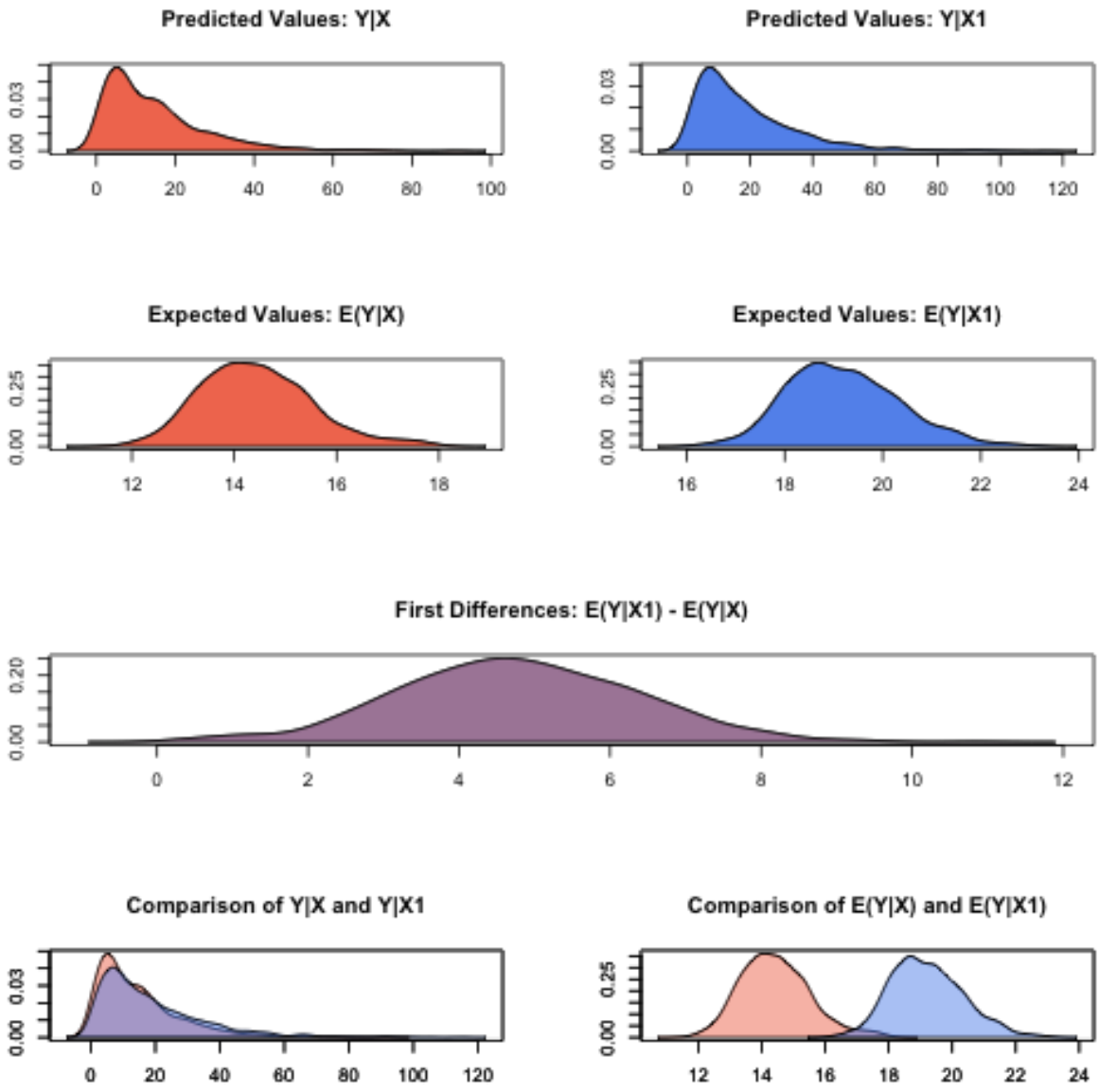


Figure 2.2: Zelig-gamma

2.3.4 Quantities of Interest

- The expected values (qi$ev) are simulations of the mean of the stochastic component given draws of α and β from their posteriors:

$$E(Y) = \alpha\lambda_i.$$

- The predicted values (qi$pr) are draws from the gamma distribution for each given set of parameters (α, λ_i).
- If x1 is specified, sim() also returns the differences in the expected values (qi$fd),

$$E(Y \mid x_1) - E(Y \mid x)$$

.

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

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

where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $E[Y_i(t_i = 0)]$, the counterfactual expected value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

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

$$\frac{1}{\sum_{i=1}^{n} t_i} \sum_{i:t_i=1}^{n} \left\{ Y_i(t_i = 1) - Y_i(\widehat{t_i = 0}) \right\},$$

where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $Y_i(\widehat{t_i = 0})$, the counterfactual predicted value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

2.3.5 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 = gamma, data)`, then you may examine the available information in `z.out` by using `names(z.out)`, see the coefficients by using z.out$coefficients, and a default summary of information through `summary(z.out)`.

2.3.6 See also

The gamma model is part of the stats package. Advanced users may wish to refer to `help(glm)` and `help(family)`.

2.4 zelig-logit

Logistic Regression for Dichotomous Dependent Variables

Logistic regression specifies a dichotomous dependent variable as a function of a set of explanatory variables.

2.4.1 Syntax

With reference classes:

```
z5 <- zlogit$new()
z5$zelig(Y ~ X1 + X ~ X, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

```
z.out <- zelig(Y ~ X1 + X2, model = "logit", data = mydata)
x.out <- setx(z.out)
s.out <- sim(z.out, x = x.out, x1 = NULL)
```

2.4.2 Examples

Basic Example

Attaching the sample turnout dataset:

```
data(turnout)
```

Estimating parameter values for the logistic regression:

```
z.out1 <- zelig(vote ~ age + race, model = "logit", data = turnout)

## How to cite this model in Zelig:
##   Kosuke Imai, Gary King, Olivia Lau. 2007.
##   logit: Logistic Regression for Dichotomous Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

Setting values for the explanatory variables:

```
x.out1 <- setx(z.out1, age = 36, race = "white")
```

Simulating quantities of interest from the posterior distribution.

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

summary(s.out1)

##
##  sim x :
##  -----
## ev
##           mean         sd       50%      2.5%     97.5%
## [1,] 0.7480614 0.01138812 0.7484417 0.7258056 0.7699426
## pv
##          0     1
## [1,] 0.275 0.725

plot(s.out1)
```

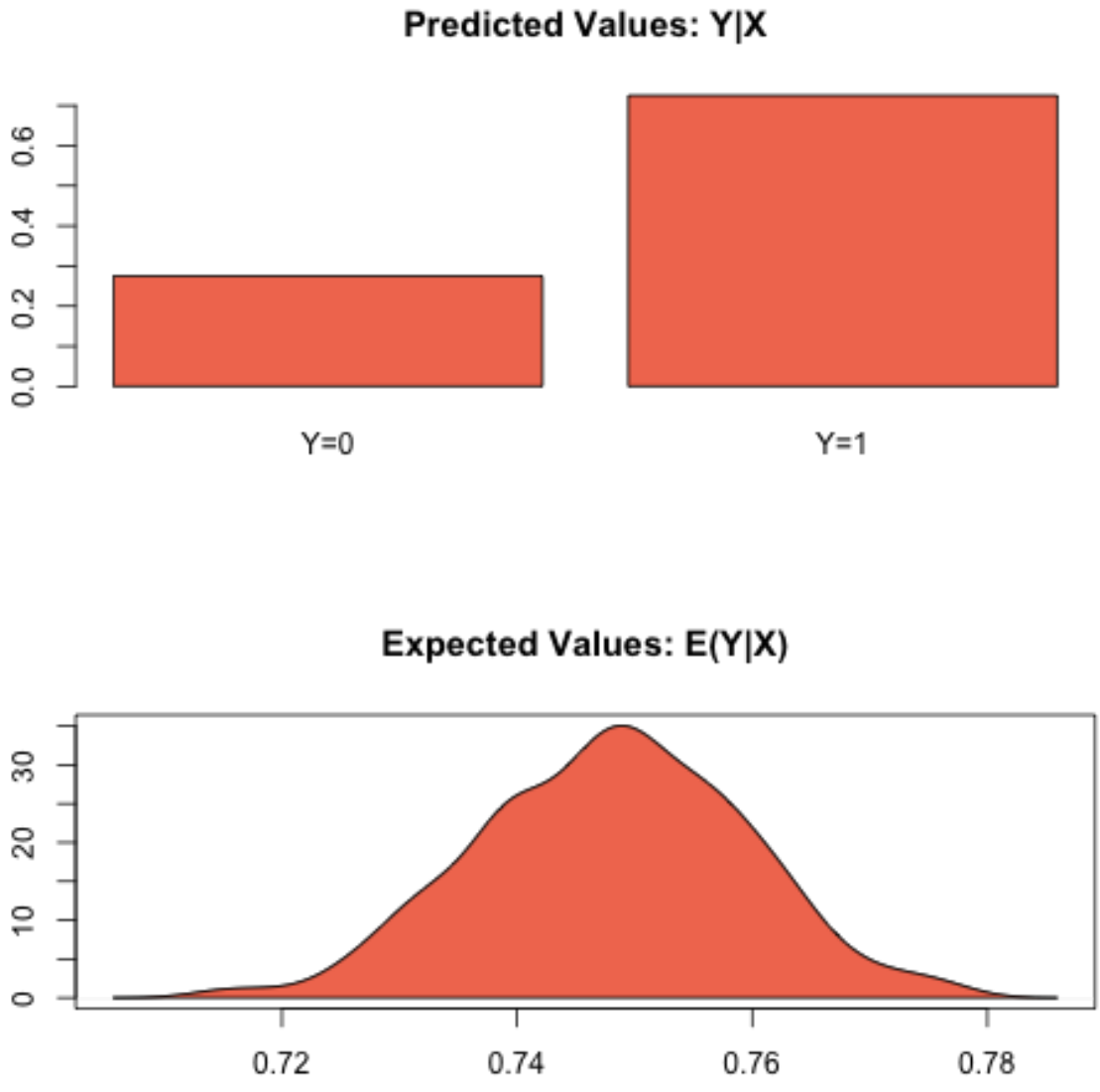


Figure 2.3: Zelig-logit-1

Simulating First Differences

Estimating the risk difference (and risk ratio) between low education (25th percentile) and high education (75th percentile) while all the other variables held at their default values.

```
z.out2 <- zelig(vote ~ race + educate, model = "logit", data = turnout)

## How to cite this model in Zelig:
##   Kosuke Imai, Gary King, Olivia Lau. 2007.
##   logit: Logistic Regression for Dichotomous Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/

x.high <- setx(z.out2, educate = quantile(turnout$educate, prob = 0.75))
x.low <- setx(z.out2, educate = quantile(turnout$educate, prob = 0.25))
s.out2 <- sim(z.out2, x = x.high, x1 = x.low)
summary(s.out2)

##
##  sim x :
##  -----
## ev
##           mean         sd       50%      2.5%    97.5%
## [1,] 0.8230526 0.01064902 0.8234506 0.8006938 0.843638
## pv
##          0     1
## [1,] 0.177 0.823
##
##  sim x1 :
##  -----
## ev
##           mean         sd       50%      2.5%     97.5%
## [1,] 0.7095084 0.01275007 0.7103366 0.6835694 0.7324867
## pv
##         0    1
## [1,] 0.31 0.69
## fd
##            mean         sd        50%       2.5%       97.5%
## [1,] -0.1135442 0.01117081 -0.1134733 -0.1356103 -0.09268328

plot(s.out2)
```

2.4.3 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

$$\begin{aligned} Y_i \sim & \text{ Bernoulli}(y_i \mid \pi_i) \\ = & \ \pi_i^{y_i}(1-\pi_i)^{1-y_i} \end{aligned}$$

  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.

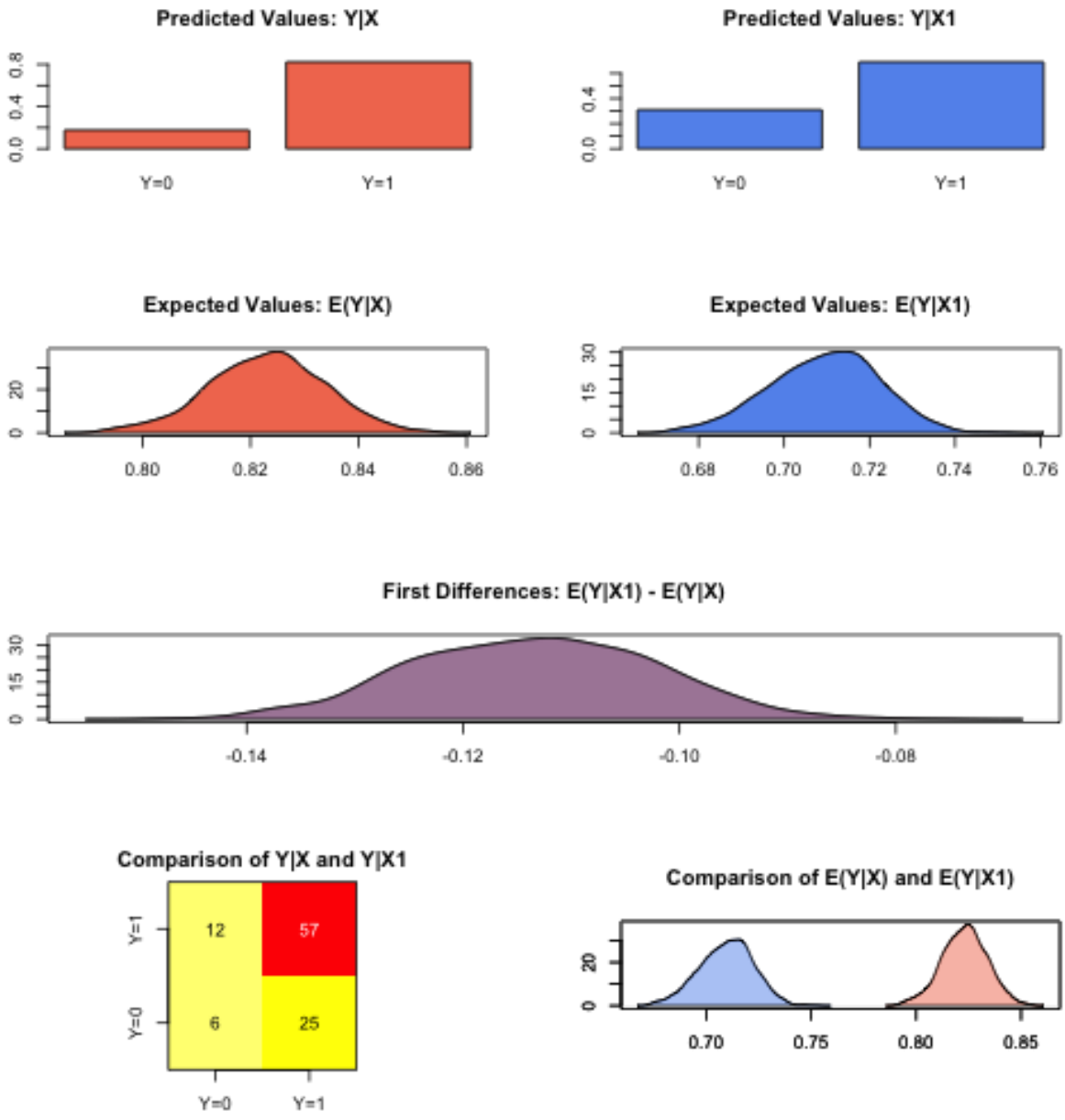


Figure 2.4: Zelig-logit-2

2.4.4 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 draws of β from its sampling distribution.

- The predicted values (qi$pr) are draws from the Binomial 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 (att.ev) for the treatment group is

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

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $E[Y_i(t_i = 0)]$, the counterfactual expected value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

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

$$\frac{1}{\sum_{i=1}^{n} t_i} \sum_{i:t_i=1}^{n} \left\{ Y_i(t_i = 1) - Y_i(\widehat{t_i = 0}) \right\},$$

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $Y_i(\widehat{t_i = 0})$, the counterfactual predicted value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

2.4.5 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, data)`, then you may examine the available information in `z.out` by using `names(z.out)`, see the coefficients by using z.out$coefficients, and a default summary of information through `summary(z.out)`.

2.4.6 See also

The logit model is part of the stats package. Advanced users may wish to refer to `help(glm)` and `help(family)`.

2.5 zelig-lognorm

Log-Normal Regression for Duration Dependent Variables

The log-normal model describes an event's duration, the dependent variable, as a function of a set of explanatory variables. The log-normal model may take time censored dependent variables, and allows the hazard rate to increase and decrease.

2.5.1 Syntax

With reference classes:

```
z5 <- zlognorm$new()
z5$zelig(Surv(Y, C) ~ X, data = mydata)
z5$setx()
z5$sim()
```

With reference classes:

```
z5 <- zlognorm$new()
z5$zelig(Surv(Y, C) ~ X, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

```
z.out <- zelig(Surv(Y, C) ~ X, model = "lognorm", data = mydata)
x.out <- setx(z.out)
s.out <- sim(z.out, x = x.out)
```

Log-normal models require that the dependent variable be in the form Surv(Y, C), where Y and C are vectors of length n. For each observation i in 1, ..., n, the value y_i is the duration (lifetime, for example) of each subject, and the associated c_i is a binary variable such that $c_i = 1$ if the duration is not censored (*e.g.*, the subject dies during the study) or $c_i = 0$ if the duration is censored (*e.g.*, the subject is still alive at the end of the study). If c_i is omitted, all Y are assumed to be completed; that is, time defaults to 1 for all observations.

2.5.2 Input Values

In addition to the standard inputs, zelig() takes the following additional options for lognormal regression:

- robust: defaults to FALSE. If TRUE, zelig() computes robust standard errors based on sandwich estimators (see and) based on the options in cluster.
- cluster: if robust = TRUE, you may select a variable to define groups of correlated observations. Let x3 be a variable that consists of either discrete numeric values, character strings, or factors that define strata. Then

```
z.out <- zelig(y ~ x1 + x2, robust = TRUE, cluster = "x3", model = "exp", data = mydata)
```

means that the observations can be correlated within the strata defined by the variable x3, and that robust standard errors should be calculated according to those clusters. If robust = TRUE but cluster is not specified, zelig() assumes that each observation falls into its own cluster.

2.5.3 Example

Attach the sample data:

```
data(coalition)
```

Estimate the model:

```
z.out <- zelig(Surv(duration, ciep12) ~ fract + numst2, model ="lognorm",  data = coalition)

## How to cite this model in Zelig:
##   Matthew Owen, Olivia Lau, Kosuke Imai, Gary King. 2007.
##   lognorm: Log-Normal Regression for Duration Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

View the regression output:

```
summary(z.out)

## Model:
## $by
## [1] 1
##
## Call:
## survival::survreg(formula = Surv(duration, ciep12) ~ fract +
##     numst2, data = ., dist = "lognormal", model = FALSE)
##
## Coefficients:
## (Intercept)       fract      numst2
##  5.36666977 -0.00443755  0.55983251
##
## Scale= 1.20008
##
## Loglik(model)= -1077.9   Loglik(intercept only)= -1101.2
##  Chisq= 46.58 on 2 degrees of freedom, p= 7.7e-11
## n= 314
## Next step: Use 'setx' method
```

Set the baseline values (with the ruling coalition in the minority) and the alternative values (with the ruling coalition in the majority) for X:

```
x.low <- setx(z.out, numst2 = 0)

## Error in terms(lm(formula, data), "predvars"): error in evaluating the argument 'x' in selecting a
## Calls: lm -> lm.fit -> Ops.Surv

x.high <- setx(z.out, numst2= 1)

## Error in terms(lm(formula, data), "predvars"): error in evaluating the argument 'x' in selecting a
## Calls: lm -> lm.fit -> Ops.Surv
```

Simulate expected values (qi$ev) and first differences (qi$fd):

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

summary(s.out)

##
##  sim x :
##  -----
## ev
##           mean         sd       50%      2.5%     97.5%
```

```
## [1,] 0.7100528 0.01309083 0.7102303 0.6833987 0.7348151
## pv
##          0     1
## [1,] 0.289 0.711
##
##  sim x1 :
##  -----
## ev
##           mean         sd       50%      2.5%    97.5%
## [1,] 0.8231874 0.01030848 0.8233464 0.8015839 0.841741
## pv
##          0     1
## [1,] 0.161 0.839
## fd
##           mean         sd       50%       2.5%    97.5%
## [1,] 0.1131346 0.01174466 0.1133255 0.08971273 0.135815
```

```
plot(s.out)
```

2.5.4 Model

Let Y_i^* be the survival time for observation i with the density function $f(y)$ and the corresponding distribution function $F(t) = \int_0^t f(y)dy$. This variable might be censored for some observations at a fixed time y_c such that the fully observed dependent variable, Y_i, is defined as

$$Y_i = \begin{cases} Y_i^* & \text{if } Y_i^* \leq y_c \\ y_c & \text{if } Y_i^* > y_c \end{cases}$$

- The *stochastic component* is described by the distribution of the partially observed variable, Y^*. For the lognormal model, there are two equivalent representations:

  $$Y_i^* \sim \text{LogNormal}(\mu_i, \sigma^2) \text{ or } \quad \log(Y_i^*) \sim \text{Normal}(\mu_i, \sigma^2)$$

  where the parameters μ_i and σ^2 are the mean and variance of the Normal distribution. (Note that the output from zelig() parameterizes scale:math:‘ = sigma‘.)

  In addition, survival models like the lognormal have three additional properties. The hazard function $h(t)$ measures the probability of not surviving past time t given survival up to t. In general, the hazard function is equal to $f(t)/S(t)$ where the survival function $S(t) = 1 - \int_0^t f(s)ds$ represents the fraction still surviving at time t. The cumulative hazard function $H(t)$ describes the probability of dying before time t. In general, $H(t) = \int_0^t h(s)ds = -\log S(t)$. In the case of the lognormal model,

  $$\begin{aligned} h(t) &= \frac{1}{\sqrt{2\pi}\,\sigma t\, S(t)} \exp\left\{-\frac{1}{2\sigma^2}(\log \lambda t)^2\right\} \\ S(t) &= 1 - \Phi\left(\frac{1}{\sigma}\log \lambda t\right) \\ H(t) &= -\log\left\{1 - \Phi\left(\frac{1}{\sigma}\log \lambda t\right)\right\} \end{aligned}$$

  where $\Phi(\cdot)$ is the cumulative density function for the Normal distribution.

- The *systematic component* is described as:

  $$\mu_i = x_i\beta.$$

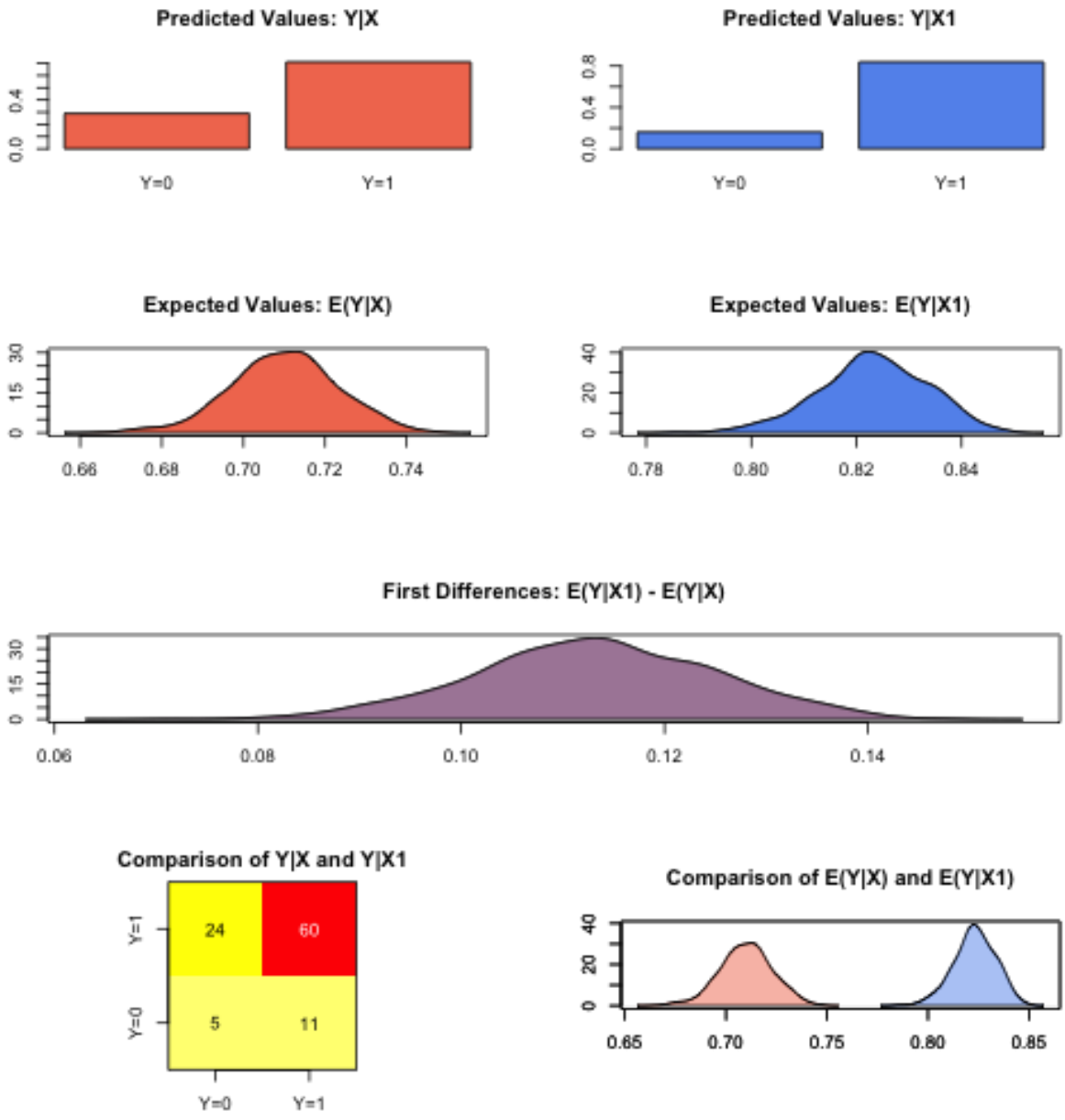


Figure 2.5: Zelig-lognorm

2.5.5 Quantities of Interest

- The expected values (qi$ev) for the lognormal model are simulations of the expected duration:

$$E(Y) = \exp\left(\mu_i + \frac{1}{2}\sigma^2\right),$$

  given draws of β and σ from their sampling distributions.

- The predicted value is a draw from the log-normal distribution given simulations of the parameters (λ_i, σ).

- The first difference (qi$fd) is

$$\text{FD} = E(Y \mid x_1) - E(Y \mid x).$$

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

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

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. When $Y_i(t_i = 1)$ is censored rather than observed, we replace it with a simulation from the model given available knowledge of the censoring process. Variation in the simulations is due to two factors: uncertainty in the imputation process for censored y_i^* and uncertainty in simulating $E[Y_i(t_i = 0)]$, the counterfactual expected value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

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

$$\frac{1}{\sum_{i=1}^{n} t_i} \sum_{i:t_i=1}^{n} \{Y_i(t_i = 1) - Y_i(\widehat{t_i = 0})\},$$

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. When $Y_i(t_i = 1)$ is censored rather than observed, we replace it with a simulation from the model given available knowledge of the censoring process. Variation in the simulations are due to two factors: uncertainty in the imputation process for censored y_i^* and uncertainty in simulating $Y_i(\widehat{t_i = 0})$, the counterfactual predicted value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

2.5.6 Output Values

The output of each Zelig command contains useful information which you may view. For example, if you run `z.out <- zelig(Surv(Y, C) ~ X, model = lognorm, data)`, then you may examine the available information in `z.out` by using `names(z.out)`, see the coefficients by using z.out$coefficients, and a default summary of information through `summary(z.out)`.

2.5.7 See also

The exponential function is part of the survival library by by Terry Therneau, ported to R by Thomas Lumley. Advanced users may wish to refer to `help(survfit)` in the survival library.

2.6 zelig-ls

Least Squares Regression for Continuous Dependent Variables

Use least squares regression analysis to estimate the best linear predictor for the specified dependent variables.

2.6.1 Syntax

With reference classes:

```
z5 <- zls$new()
z5$zelig(Y ~ X1 + X ~ X, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

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

2.6.2 Examples

Basic Example with First Differences

Attach sample data:

```
data(macro)
```

Estimate model:

```
z.out1 <- zelig(unem ~ gdp + capmob + trade, model = "ls", data = macro)
```

```
## How to cite this model in Zelig:
##   Kosuke Imai, Gary King, and Olivia Lau. 2007.
##   ls: Least Squares Regression for Continuous Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

Summarize regression coefficients:

```
summary(z.out1)
```

```
## Model:
## $by
## [1] 1
##
##
## Call:
## stats::lm(formula = unem ~ gdp + capmob + trade, data = .)
##
## Coefficients:
## (Intercept)          gdp       capmob        trade
##     6.18129     -0.32360      1.42194      0.01985
##
## Next step: Use 'setx' method
```

Set explanatory variables to their default (mean/mode) values, with high (80th percentile) and low (20th percentile) values for the trade variable:

```
x.high <- setx(z.out1, trade = quantile(macro$trade, 0.8))
x.low <- setx(z.out1, trade = quantile(macro$trade, 0.2))
```

Generate first differences for the effect of high versus low trade on GDP:

```
s.out1 <- sim(z.out1, x = x.high, x1 = x.low)

summary(s.out1)
```

```
##
##  sim x :
##  -----
## ev
##       mean        sd      50%     2.5%    97.5%
## 1 5.425577 0.1913408 5.426885 5.071941 5.790102
## pv
##       mean        sd      50%     2.5%    97.5%
## 1 5.425577 0.1913408 5.426885 5.071941 5.790102
##
##  sim x1 :
##  -----
## ev
##       mean        sd      50%    2.5%    97.5%
## 1 4.607931 0.1868002 4.607313 4.23521 4.976644
## pv
##       mean        sd      50%    2.5%    97.5%
## 1 4.607931 0.1868002 4.607313 4.23521 4.976644
## fd
##         mean        sd        50%      2.5%      97.5%
## 1 -0.8176458 0.2351794 -0.8129858 -1.264778 -0.3534317
```

```
plot(s.out1)
```

Using Dummy Variables

Estimate a model with fixed effects for each country (see for help with dummy variables). Note that you do not need to create dummy variables, as the program will automatically parse the unique values in the selected variable into discrete levels.

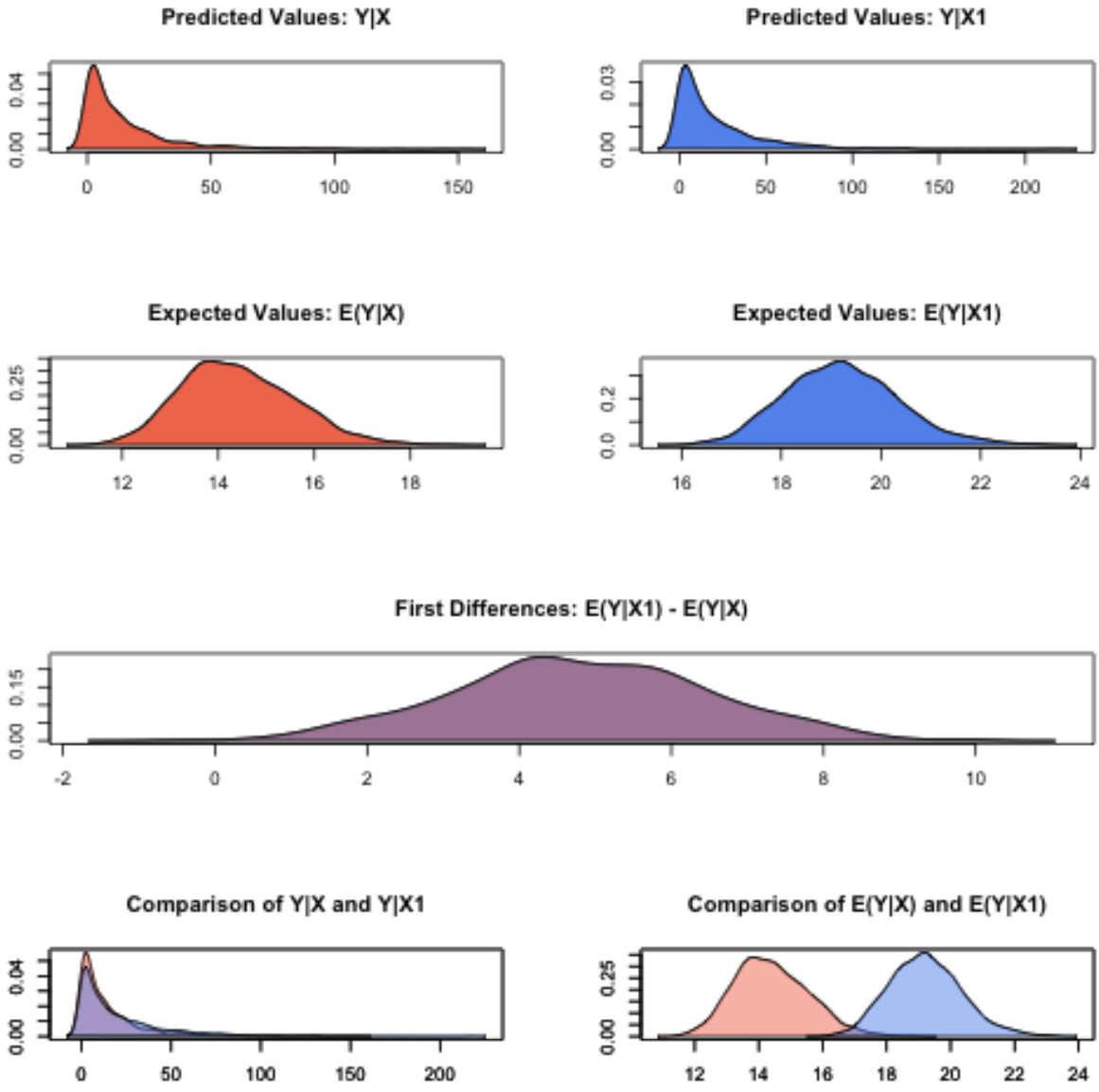
```
z.out2 <- zelig(unem ~ gdp + trade + capmob + as.factor(country), model = "ls", data = macro)
```

```
## How to cite this model in Zelig:
##   Kosuke Imai, Gary King, and Olivia Lau. 2007.
##   ls: Least Squares Regression for Continuous Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

Set values for the explanatory variables, using the default mean/mode values, with country set to the United States and Japan, respectively:

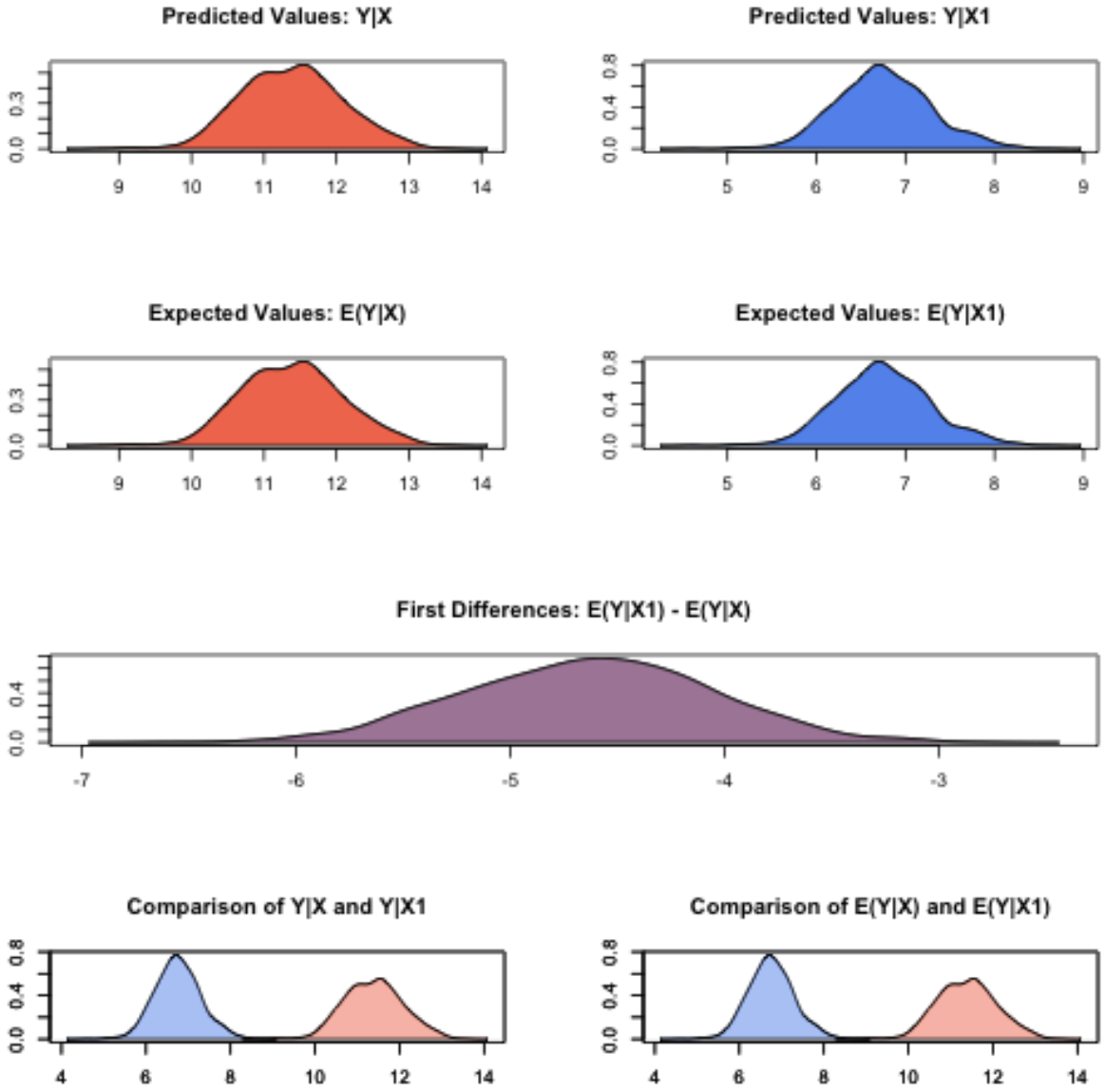
```
x.US <- setx(z.out2, country = "United States")
x.Japan <- setx(z.out2, country = "Japan")
```

Simulate quantities of interest:


Predicted Values: Y|X
0.00
0.04
0
50
100
150
Predicted Values: Y|X1
0.00
0.03
0
50
100
150
200
Expected Values: E(Y|X)
0.00
0.25
12
14
16
18
Expected Values: E(Y|X1)
0.0
0.2
16
18
20
22
24
First Differences: E(Y|X1) - E(Y|X)
0.00
0.15
-2
0
2
4
6
8
10
Comparison of Y|X and Y|X1
0.00
0.04
0
50
100
150
200
Comparison of E(Y|X) and E(Y|X1)
0.00
0.25
12
14
16
18
20
22
24

```
s.out2 <- sim(z.out2, x = x.US, x1 = x.Japan)

plot(s.out2)
```



2.6.3 Model

- The *stochastic component* is described by a density with mean μ_i and the common variance σ^2

$$Y_i \sim f(y_i \mid \mu_i, \sigma^2).$$

- The *systematic component* models the conditional mean as

$$\mu_i = x_i \beta$$

where x_i is the vector of covariates, and β is the vector of coefficients.

The least squares estimator is the best linear predictor of a dependent variable given x_i, and minimizes the sum of squared residuals, $\sum_{i=1}^{n}(Y_i - x_i\beta)^2$.

2.6.4 Quantities of Interest

- The expected value (qi$ev) is the mean of simulations from the stochastic component,

$$E(Y) = x_i\beta,$$

given a draw of β from its sampling distribution.

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

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

where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $E[Y_i(t_i = 0)]$, the counterfactual expected value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

2.6.5 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 = ls, data)`, then you may examine the available information in `z.out` by using `names(z.out)`, see 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: parameter estimates for the explanatory variables.
  - residuals: the working residuals in the final iteration of the IWLS fit.
  - fitted.values: fitted values.
  - df.residual: the residual degrees of freedom.
  - zelig.data: the input data frame if save.data = TRUE.
- From summary(z.out), you may extract:
  - coefficients: the parameter estimates with their associated standard errors, p-values, and t-statistics.

$$\hat{\beta} = \left(\sum_{i=1}^{n} x_i' x_i \right)^{-1} \sum x_i y_i$$

  - sigma: the square root of the estimate variance of the random error e:

$$\hat{\sigma} = \frac{\sum (Y_i - x_i\hat{\beta})^2}{n - k}$$

  - r.squared: the fraction of the variance explained by the model.

$$R^2 = 1 - \frac{\sum (Y_i - x_i\hat{\beta})^2}{\sum (y_i - \bar{y})^2}$$

- adj.r.squared: the above R^2 statistic, penalizing for an increased number of explanatory variables.
- cov.unscaled: a $k \times k$ matrix of unscaled covariances.

2.6.6 See also

The least squares regression is part of the stats package by William N. Venables and Brian D. Ripley .In addition, advanced users may wish to refer to `help(lm)` and `help(lm.fit)`.

2.7 zelig-negbin

Negative Binomial Regression for Event Count Dependent Variables

Use the negative binomial regression if you have a count of events for each observation of your dependent variable. The negative binomial model is frequently used to estimate over-dispersed event count models.

2.7.1 Syntax

With reference classes:

```
z5 <- znegbin$new()
z5$zelig(Y ~ X1 + X ~ X, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

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

2.7.2 Example

Load sample data:

```
data(sanction)
```

Estimate the model:

```
z.out <- zelig(num ~ target + coop, model = "negbin", data = sanction)

## How to cite this model in Zelig:
##   Kosuke Imai, Gary King, Olivia Lau. 2008.
##   negbinom: Negative Binomial Regression for Event Count Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/

summary(z.out)
```

```
## Model:
## $by
## [1] 1
##
##
## Call:  MASS::glm.nb(formula = num ~ target + coop, data = ., init.theta = 1.841603403,
##     link = log)
##
## Coefficients:
## (Intercept)       target         coop
##      -1.564        0.151        1.286
##
## Degrees of Freedom: 77 Total (i.e. Null);  75 Residual
## Null Deviance:       237.1
## Residual Deviance: 56.55         AIC: 360.2
## Next step: Use 'setx' method
```

Set values for the explanatory variables to their default mean values:

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

Simulate fitted values:

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

summary(s.out)
```

```
##
##  sim x :
##  -----
## ev
##          mean       sd      50%     2.5%    97.5%
## [1,] 2.979505 0.354668 2.957069 2.386268 3.744801
## pv
## qi
##     0     1     2     3     4     5     6     7     8     9    10    11
## 0.212 0.207 0.201 0.123 0.098 0.051 0.041 0.023 0.017 0.008 0.004 0.004
##    12    13    14    19
## 0.004 0.004 0.002 0.001
```

```
plot(s.out)
```

2.7.3 Model

Let Y_i be the number of independent events that occur during a fixed time period. This variable can take any non-negative integer value.

- The negative binomial distribution is derived by letting the mean of the Poisson distribution vary according to a fixed parameter ζ given by the Gamma distribution. The *stochastic component* is given by

$$\begin{aligned} Y_i \mid \zeta_i \sim & \quad \text{Poisson}(\zeta_i \mu_i), \\ \zeta_i \sim & \quad \frac{1}{\theta}\text{Gamma}(\theta). \end{aligned}$$

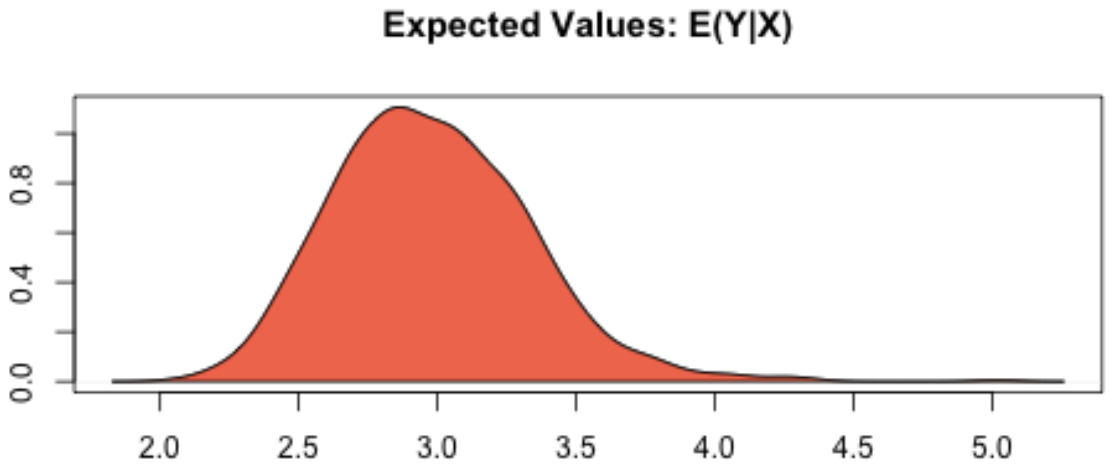


Figure 2.6: Zelig-negbin

The marginal distribution of Y_i is then the negative binomial with mean μ_i and variance $\mu_i + \mu_i^2/\theta$:

$$\begin{aligned} Y_i \sim & \quad \text{NegBin}(\mu_i, \theta), \\ = & \quad \frac{\Gamma(\theta + y_i)}{y!\,\Gamma(\theta)} \frac{\mu_i^{y_i}\,\theta^{\theta}}{(\mu_i + \theta)^{\theta + y_i}}, \end{aligned}$$

where θ is the systematic parameter of the Gamma distribution modeling ζ_i.

- The *systematic component* is given by

$$\mu_i = \exp(x_i \beta)$$

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

2.7.4 Quantities of Interest

- The expected values (qi$ev) are simulations of the mean of the stochastic component. Thus,

$$E(Y) = \mu_i = \exp(x_i \beta),$$

given simulations of β.

- The predicted value (qi$pr) drawn from the distribution defined by the set of parameters (μ_i, θ).
- The first difference (qi$fd) is

$$\text{FD} \;=\; E(Y|x_1) - E(Y \mid x)$$

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

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

where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $E[Y_i(t_i = 0)]$, the counterfactual expected value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

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

$$\frac{1}{\sum_{i=1}^{n} t_i} \sum_{i:t_i=1}^{n} \left\{ Y_i(t_i = 1) - Y_i(\widehat{t_i = 0}) \right\},$$

where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $Y_i(\widehat{t_i = 0})$, the counterfactual predicted value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

2.7.5 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 = negbin, data)`, then you may examine the available information in `z.out` by using `names(z.out)`, see the coefficients by using z.out$coefficients, and a default summary of information through `summary(z.out)`.

2.7.6 See also

The negative binomial model is part of the MASS package by William N. Venable and Brian D. Ripley . Advanced users may wish to refer to ‘‘help(glm.nb)‘.

2.8 zelig-normal

Normal Regression for Continuous Dependent Variables

The Normal regression model is a close variant of the more standard least squares regression model (see). Both models specify a continuous dependent variable as a linear function of a set of explanatory variables. The Normal model reports maximum likelihood (rather than least squares) estimates. The two models differ only in their estimate for the stochastic parameter σ.

2.8.1 Syntax

With reference classes:

```
z5 <- znormal$new()
z5$zelig(Y ~ X1 + X ~ X, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

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

2.8.2 Examples

Basic Example with First Differences

Attach sample data:

```
data(macro)
```

Estimate model:

```
z.out1 <- zelig(unem ~ gdp + capmob + trade, model = "normal", data = macro)

## How to cite this model in Zelig:
##   Kosuke Imai, Gary King, Olivia Lau. 2008.
##   normal: Normal Regression for Continuous Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

Summarize of regression coefficients:

```
summary(z.out1)
```

```
## Model:
## $by
## [1] 1
##
##
## Call:  stats::glm(formula = unem ~ gdp + capmob + trade, family = gaussian("identity"),
##     data = .)
##
## Coefficients:
## (Intercept)          gdp       capmob        trade
##     6.18129     -0.32360      1.42194      0.01985
##
## Degrees of Freedom: 349 Total (i.e. Null);  346 Residual
## Null Deviance:       3665
## Residual Deviance: 2610  AIC: 1706
## Next step: Use 'setx' method
```

Set explanatory variables to their default (mean/mode) values, with high (80th percentile) and low (20th percentile) values for trade:

```
x.high <- setx(z.out1, trade = quantile(macro$trade, 0.8))
x.low <- setx(z.out1, trade = quantile(macro$trade, 0.2))
```

Generate first differences for the effect of high versus low trade on GDP:

```
s.out1 <- sim(z.out1, x = x.high, x1 = x.low)

summary(s.out1)
```

```
##
##  sim x :
##  -----
## ev
##          mean        sd      50%     2.5%    97.5%
## [1,] 5.422257 0.1989366 5.424447 5.028838 5.809112
## pv
##          mean       sd      50%      2.5%    97.5%
## [1,] 5.333821 2.744388 5.396203 -0.368982 10.55112
##
##  sim x1 :
##  -----
## ev
##          mean        sd      50%     2.5%    97.5%
## [1,] 4.596018 0.1876396 4.598034 4.238753 4.954279
## pv
##          mean       sd      50%     2.5%    97.5%
## [1,] 4.466941 2.863781 4.397681 -1.18165 10.10794
## fd
##            mean        sd        50%      2.5%      97.5%
## [1,] -0.8262389 0.2342038 -0.8192718 -1.279973 -0.3696419
```

A visual summary of quantities of interest:

```
plot(s.out1)
```

2.8.3 Model

Let Y_i be the continuous dependent variable for observation i.

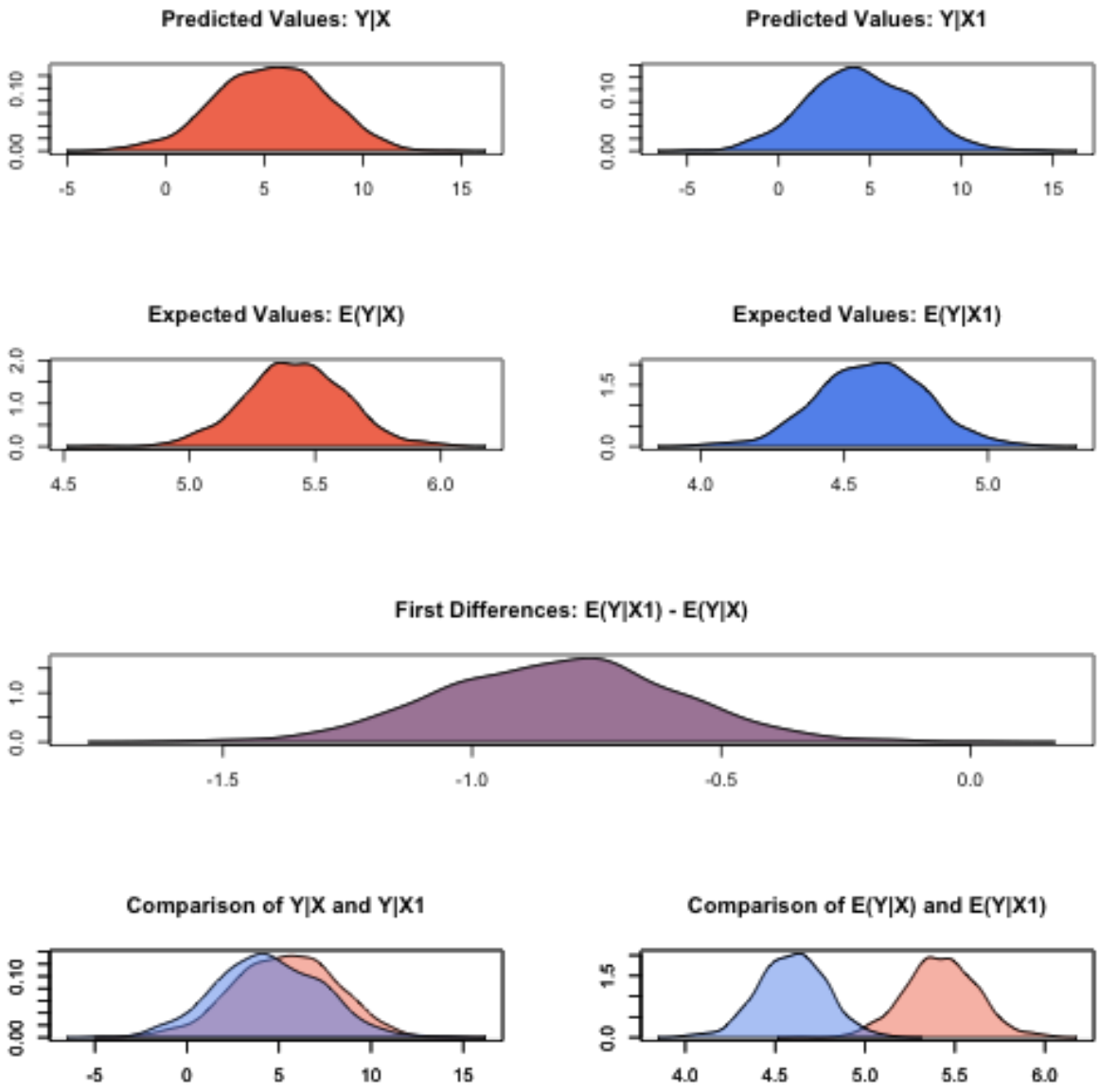


Figure 2.7: Zelig-normal

- The *stochastic component* is described by a univariate normal model with a vector of means μ_i and scalar variance σ^2:

$$Y_i \sim \text{Normal}(\mu_i, \sigma^2).$$

- The *systematic component* is

$$\mu_i = x_i\beta,$$

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

2.8.4 Quantities of Interest

- The expected value (qi$ev) is the mean of simulations from the the stochastic component,

$$E(Y) = \mu_i = x_i\beta,$$

given a draw of β from its posterior.

- The predicted value (qi$pr) is drawn from the distribution defined by the set of parameters (μ_i, σ).
- The first difference (qi$fd) is:

$$\text{FD} = E(Y \mid x_1) - E(Y \mid x)$$

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

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

where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $E[Y_i(t_i = 0)]$, the counterfactual expected value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

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

$$\frac{1}{\sum_{i=1}^{n} t_i} \sum_{i:t_i=1}^{n} \left\{ Y_i(t_i = 1) - Y_i(\widehat{t_i = 0}) \right\},$$

where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $Y_i(\widehat{t_i = 0})$, the counterfactual predicted value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

2.8.5 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 = normal, data)`, then you may examine the available information in `z.out` by using `names(z.out)`, see the coefficients by using z.out$coefficients, and a default summary of information through `summary(z.out)`.

2.8.6 See also

The normal model is part of the stats package by . Advanced users may wish to refer to `help(glm)` and `help(family)`.

2.9 zelig-poisson

Poisson Regression for Event Count Dependent Variables

Use the Poisson regression model if the observations of your dependent variable represents the number of independent events that occur during a fixed period of time (see the negative binomial model, , for over-dispersed event counts.) For a Bayesian implementation of this model, see .

2.9.1 Syntax

With reference classes:

```
z5 <- zpoisson$new()
z5$zelig(Y ~ X1 + X ~ X, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

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

2.9.2 Example

Load sample data:

```
data(sanction)
```

Estimate Poisson model:

```
z.out <- zelig(num ~ target + coop, model = "poisson", data = sanction)

## How to cite this model in Zelig:
##   Kosuke Imai, Gary King, Olivia Lau. 2007.
##   poisson: Poisson Regression for Event Count Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/

summary(z.out)

## Model:
## $by
## [1] 1
##
##
## Call:  stats::glm(formula = num ~ target + coop, family = poisson("log"),
```

```
##     data = .)
##
## Coefficients:
## (Intercept)       target         coop
##    -0.96772     -0.02102      1.21082
##
## Degrees of Freedom: 77 Total (i.e. Null);  75 Residual
## Null Deviance:       1584
## Residual Deviance: 720.8         AIC: 944.3
## Next step: Use 'setx' method
```

Set values for the explanatory variables to their default mean values:

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

Simulate fitted values:

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

```
##
##  sim x :
##  -----
## ev
##          mean        sd      50%     2.5%    97.5%
## [1,] 3.247022 0.2367508 3.240276 2.803836 3.728323
## pv
##      mean       sd 50% 2.5% 97.5%
## [1,] 3.14 1.743973   3    0     7
```

```
plot(s.out)
```

2.9.3 Model

Let Y_i be the number of independent events that occur during a fixed time period. This variable can take any non-negative integer.

- The Poisson distribution has *stochastic component*

$$Y_i \sim \text{Poisson}(\lambda_i),$$

  where λ_i is the mean and variance parameter.

- The *systematic component* is

$$\lambda_i = \exp(x_i\beta),$$

  where x_i is the vector of explanatory variables, and β is the vector of coefficients.

2.9.4 Quantities of Interest

- The expected value (qi$ev) is the mean of simulations from the stochastic component,

$$E(Y) = \lambda_i = \exp(x_i\beta),$$

  given draws of β from its sampling distribution.

- The predicted value (qi$pr) is a random draw from the poisson distribution defined by mean λ_i.

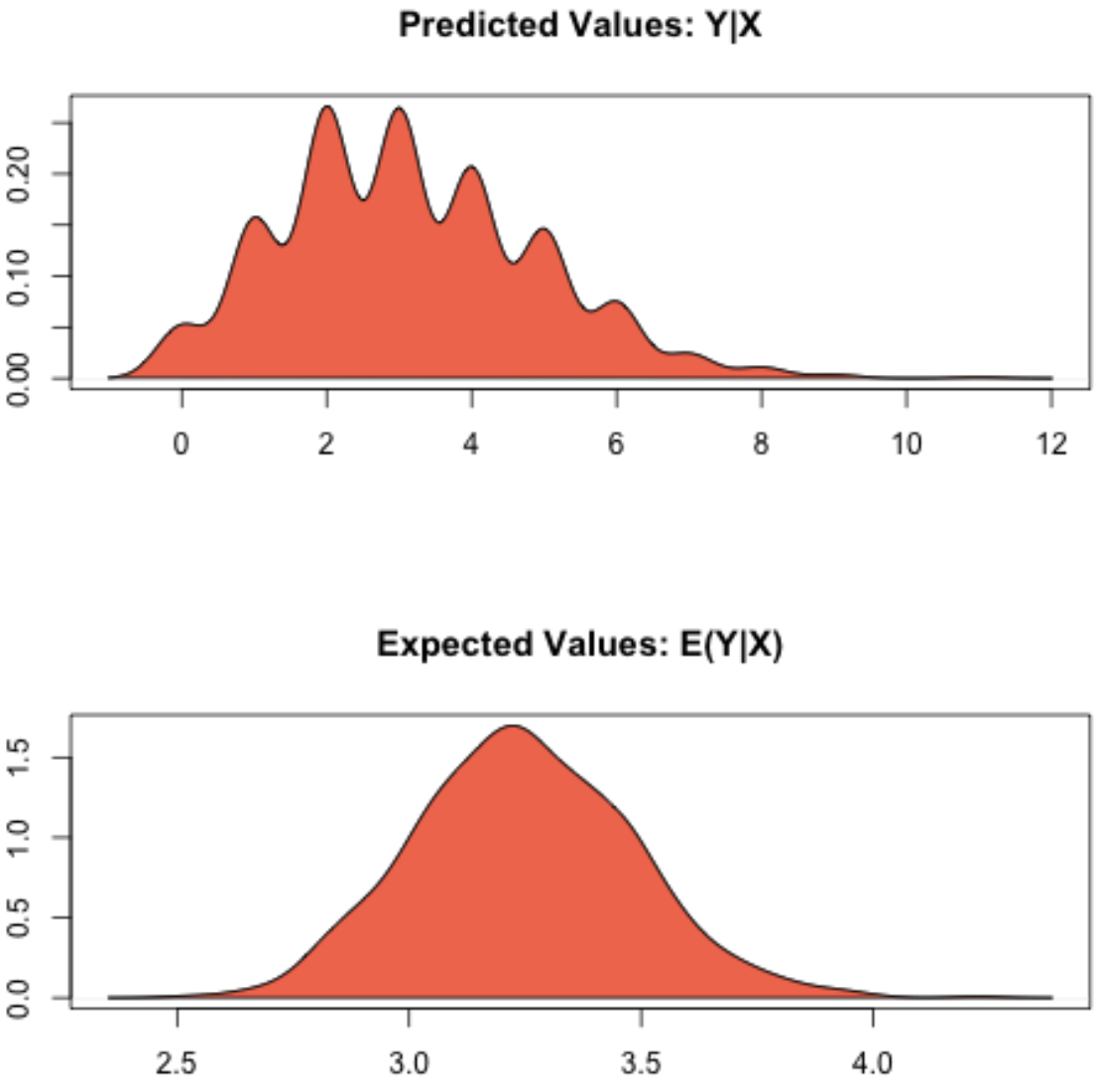


Figure 2.8: Zelig-poisson

- The first difference in the expected values (qi$fd) is given by:

$$\text{FD} = E(Y|x_1) - E(Y \mid x)$$

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

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

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $E[Y_i(t_i = 0)]$, the counterfactual expected value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

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

$$\frac{1}{\sum_{i=1}^{n} t_i} \sum_{i:t_i=1}^{n} \left\{ Y_i(t_i = 1) - Y_i(\widehat{t_i = 0}) \right\},$$

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $Y_i(\widehat{t_i = 0})$, the counterfactual predicted value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

2.9.5 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 = poisson, data)`, then you may examine the available information in `z.out` by using `names(z.out)`, see the coefficients by using z.out$coefficients, and a default summary of information through `summary(z.out)`.

2.9.6 See also

The poisson model is part of the stats package by . Advanced users may wish to refer to `help(glm)` and `help(family)`.

2.10 zelig-probit

Probit Regression for Dichotomous Dependent Variables

Use probit regression to model binary dependent variables specified as a function of a set of explanatory variables.

2.10.1 Syntax

With reference classes:

```
z5 <- zprobit$new()
z5$zelig(Y ~ X1 + X ~ X, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

```
z.out <- zelig(Y ~ X1 + X2, model = "probit", data = mydata)
x.out <- setx(z.out)
s.out <- sim(z.out, x = x.out, x1 = NULL)
```

2.10.2 Example

Attach the sample turnout dataset:

```
data(turnout)
```

Estimate parameter values for the probit regression:

```
z.out <- zelig(vote ~ race + educate, model = "probit", data = turnout)
```

```
## How to cite this model in Zelig:
##   Kosuke Imai, Gary King, Olivia Lau. 2007.
##   probit: Probit Regression for Dichotomous Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

```
summary(z.out)
```

```
## Model:
## $by
## [1] 1
##
##
## Call:  stats::glm(formula = vote ~ race + educate, family = binomial("probit"),
##     data = .)
##
## Coefficients:
## (Intercept)    racewhite      educate
##    -0.72595      0.29908      0.09712
##
## Degrees of Freedom: 1999 Total (i.e. Null);  1997 Residual
## Null Deviance:       2267
## Residual Deviance: 2136  AIC: 2142
## Next step: Use 'setx' method
```

Set values for the explanatory variables to their default values.

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

Simulate quantities of interest from the posterior distribution.

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

```
summary(s.out)
```

```
plot(s.out1)
```

2.10.3 Model

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

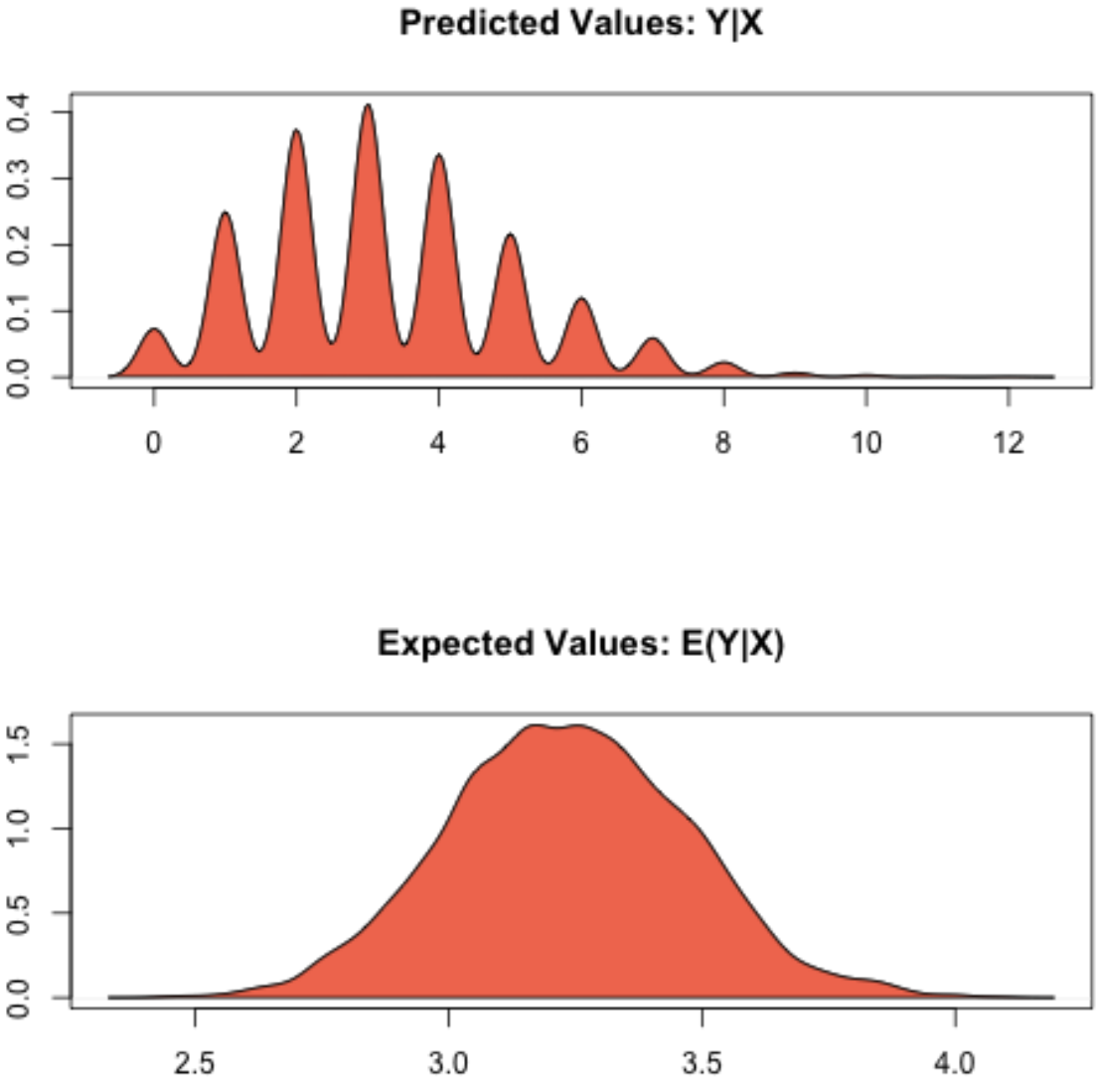


Figure 2.9: Zelig-probit

- The *stochastic component* is given by

$$Y_i \sim \text{Bernoulli}(\pi_i),$$

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

- The *systematic component* is

$$\pi_i = \Phi(x_i\beta)$$

where $\Phi(\mu)$ is the cumulative distribution function of the Normal distribution with mean 0 and unit variance.

2.10.4 Quantities of Interest

- The expected value (qi$ev) is a simulation of predicted probability of success

$$E(Y) = \pi_i = \Phi(x_i\beta),$$

given a draw of β from its sampling distribution.

- The predicted value (qi$pr) is a draw from a Bernoulli distribution with mean π_i.
- The first difference (qi$fd) in expected values 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 (att.ev) for the treatment group is

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

where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $E[Y_i(t_i = 0)]$, the counterfactual expected value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

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

$$\frac{1}{\sum_{i=1}^{n} t_i} \sum_{i:t_i=1}^{n} \left\{ Y_i(t_i = 1) - Y_i(\widehat{t_i = 0}) \right\},$$

where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $Y_i(\widehat{t_i = 0})$, the counterfactual predicted value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

2.10.5 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 = probit, data)`, then you may examine the available information in `z.out` by using `names(z.out)`, see the coefficients by using z.out$coefficients, and a default summary of information through `summary(z.out)`.

2.10.6 See also

The probit model is part of the stats package by . Advanced users may wish to refer to `help(glm)` and `help(family)`.

2.11 zelig-relogit

Rare Events Logistic Regression for Dichotomous Dependent Variables

The relogit procedure estimates the same model as standard logistic regression (appropriate when you have a dichotomous dependent variable and a set of explanatory variables; see), but the estimates are corrected for the bias that occurs when the sample is small or the observed events are rare (i.e., if the dependent variable has many more 1s than 0s or the reverse). The relogit procedure also optionally uses prior correction for case-control sampling designs.

2.11.1 Syntax

With reference classes:

```
z5 <- zrelogit$new()
z5$zelig(Y ~ X1 + X2, tau = NULL,
                      case.control = c("prior", "weighting"),
                      bias.correct = TRUE, robust = FALSE,
                      data = mydata, ...)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

```
z.out <- zelig(Y ~ X1 + X2, model = "relogit", tau = NULL,
                      case.control = c("prior", "weighting"),
                      bias.correct = TRUE, robust = FALSE,
                      data = mydata, ...)
x.out <- setx(z.out)
s.out <- sim(z.out, x = x.out)
```

2.11.2 Arguments

The relogit procedure supports four optional arguments in addition to the standard arguments for zelig(). You may additionally use:

- tau: a vector containing either one or two values for τ, the true population fraction of ones. Use, for example, tau = c(0.05, 0.1) to specify that the lower bound on tau is 0.05 and the upper bound is 0.1. If left unspecified, only finite-sample bias correction is performed, not case-control correction.
- case.control: if tau is specified, choose a method to correct for case-control sampling design: "prior" (default) or "weighting".
- bias.correct: a logical value of TRUE (default) or FALSE indicating whether the intercept should be corrected for finite sample (rare events) bias.

Note that if tau = NULL, bias.correct = FALSE, the relogit procedure performs a standard logistic regression without any correction.

2.11.3 Example 1: One Tau with Prior Correction and Bias Correction

Due to memory and space considerations, the data used here are a sample drawn from the full data set used in King and Zeng, 2001, The proportion of militarized interstate conflicts to the absence of disputes is $\tau = 1,042/303,772 \approx 0.00343$. To estimate the model,

```
data(mid)

z.out1 <- zelig(conflict ~ major + contig + power + maxdem + mindem + years, data = mid, model = "rel

## How to cite this model in Zelig:
##   Kosuke Imai, Gary King, and Olivia Lau. 2014.
##   relogit: Rare Events Logistic Regression for Dichotomous Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

Summarize the model output:

```
summary(z.out1)

## Model:
## $by
## [1] 1
##
##
## Call:  relogit(formula = cbind(conflict, 1 - conflict) ~ major + contig +
##     power + maxdem + mindem + years, data = ., tau = 0.00343020423212146,
##     bias.correct = TRUE, case.control = "prior")
##
## Coefficients:
## (Intercept)        major       contig        power       maxdem
##    -7.50836      2.43196      4.10797      1.05358      0.04804
##      mindem        years
##    -0.06413     -0.06293
##
## Degrees of Freedom: 3125 Total (i.e. Null);  3119 Residual
## Null Deviance:       3979
## Residual Deviance: 1869  AIC: 1883
## Next step: Use 'setx' method
```

Set the explanatory variables to their means:

```
x.out1 <- setx(z.out1)
```

Simulate quantities of interest:

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

##
##  sim x :
##  -----
## ev
##             mean           sd         50%        2.5%      97.5%
## [1,] 0.002405825 0.0001548869 0.002401648 0.002119784 0.00270734
## pv
##          0     1
## [1,] 0.999 0.001
```

```
plot(s.out1)
```

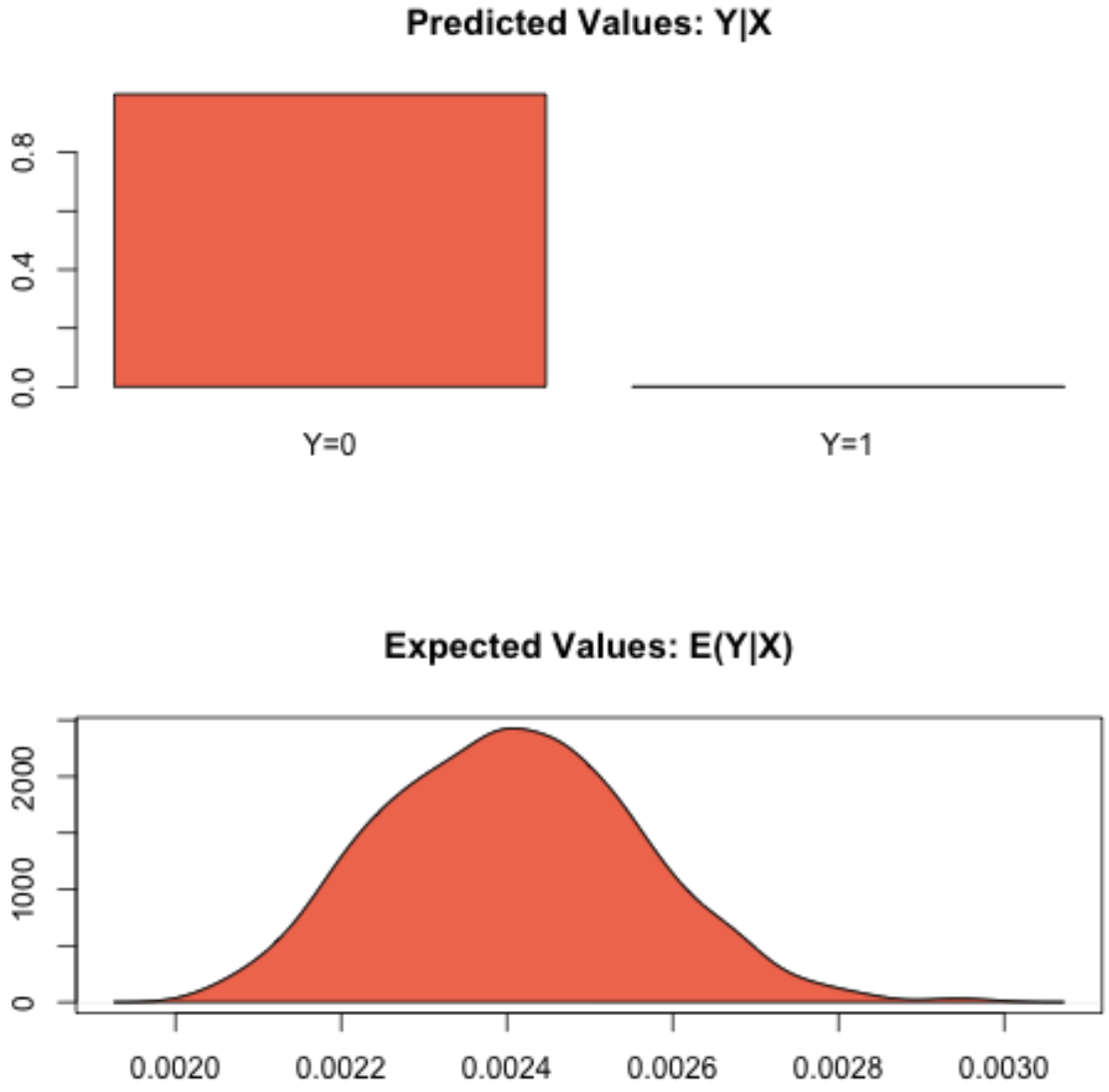


Figure 2.10: Zelig-relogit

2.11.4 Example 2: One Tau with Weighting, Robust Standard Errors, and Bias Correction

Suppose that we wish to perform case control correction using weighting (rather than the default prior correction). To estimate the model:

```
z.out2 <- zelig(conflict ~ major + contig + power + maxdem + mindem + years, data = mid, model = "rel
```

```
## Error in glm.control(robust = TRUE): unused argument (robust = TRUE)
```

Summarize the model output:

```
summary(z.out2)
```

```
## Model:
## $by
## [1] 1
##
##
## Call:  stats::glm(formula = vote ~ race + educate, family = binomial("logit"),
##     data = .)
##
## Coefficients:
## (Intercept)    racewhite      educate
##     -1.2189       0.5022       0.1610
##
## Degrees of Freedom: 1999 Total (i.e. Null);  1997 Residual
## Null Deviance:       2267
## Residual Deviance: 2138  AIC: 2144
## Next step: Use 'setx' method
```

Set the explanatory variables to their means:

```
x.out2 <- setx(z.out2)
```

Simulate quantities of interest:

```
s.out2 <- sim(z.out2, x = x.out2)
summary(s.out2)
```

```
##
##  sim x :
##  -----
## ev
##           mean         sd       50%      2.5%     97.5%
## [1,] 0.7728179 0.01032537 0.7730781 0.7496465 0.7921772
## pv
##          0     1
## [1,] 0.225 0.775
```

2.11.5 Example 3: Two Taus with Bias Correction and Prior Correction

Suppose that we did not know that $\tau \approx 0.00343$, but only that it was somewhere between $(0.002, 0.005)$. To estimate a model with a range of feasible estimates for τ (using the default prior correction method for case control correction):

```
z.out2 <- zelig(conflict ~ major + contig + power + maxdem + mindem + years, data = mid, model = "rel
```

```
## How to cite this model in Zelig:
##   Kosuke Imai, Gary King, and Olivia Lau. 2014.
##   relogit: Rare Events Logistic Regression for Dichotomous Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

Summarize the model output:

```
z.out2
```

```
## Model:
## $by
## [1] 1
##
## $lower.estimate
##
## Call:  (function (formula, data = sys.parent(), tau = NULL, bias.correct = TRUE,
##     case.control = "prior", ...)
## {
##     mf <- match.call()
##     mf$tau <- mf$bias.correct <- mf$case.control <- NULL
##     if (!is.null(tau)) {
##         tau <- unique(tau)
##         if (length(case.control) > 1)
##             stop("You can only choose one option for case control correction.")
##         ck1 <- grep("p", case.control)
##         ck2 <- grep("w", case.control)
##         if (length(ck1) == 0 & length(ck2) == 0)
##             stop("choose either case.control = \"prior\" ", "or case.control = \"weighting\"")
##         if (length(ck2) == 0)
##             weighting <- FALSE
##         else weighting <- TRUE
##     }
##     else weighting <- FALSE
##     if (length(tau) > 2)
##         stop("tau must be a vector of length less than or equal to 2")
##     else if (length(tau) == 2) {
##         mf[[1]] <- relogit
##         res <- list()
##         mf$tau <- min(tau)
##         res$lower.estimate <- eval(as.call(mf), parent.frame())
##         mf$tau <- max(tau)
##         res$upper.estimate <- eval(as.call(mf), parent.frame())
##         res$formula <- formula
##         class(res) <- c("Relogit2", "Relogit")
##         return(res)
##     }
##     else {
##         mf[[1]] <- glm
##         mf$family <- binomial(link = "logit")
##         y2 <- model.response(model.frame(mf$formula, data))
##         if (is.matrix(y2))
##             y <- y2[, 1]
##         else y <- y2
##         ybar <- mean(y)
##         if (weighting) {
##             w1 <- tau/ybar
##             w0 <- (1 - tau)/(1 - ybar)
##             wi <- w1 * y + w0 * (1 - y)
##             mf$weights <- wi
##         }
##         res <- eval(as.call(mf), parent.frame())
##         res$call <- match.call(expand.dots = TRUE)
##         res$tau <- tau
##         X <- model.matrix(res)
##         if (bias.correct) {
```

```
##             pihat <- fitted(res)
##             if (is.null(tau))
##                 wi <- rep(1, length(y))
##             else if (weighting)
##                 res$weighting <- TRUE
##             else {
##                 w1 <- tau/ybar
##                 w0 <- (1 - tau)/(1 - ybar)
##                 wi <- w1 * y + w0 * (1 - y)
##                 res$weighting <- FALSE
##             }
##             W <- pihat * (1 - pihat) * wi
##             Qdiag <- lm.influence(lm(y ~ X - 1, weights = W))$hat/W
##             if (is.null(tau))
##                 xi <- 0.5 * Qdiag * (2 * pihat - 1)
##             else xi <- 0.5 * Qdiag * ((1 + w0) * pihat - w0)
##             res$coefficients <- res$coefficients - lm(xi ~ X -
##                 1, weights = W)$coefficients
##             res$bias.correct <- TRUE
##         }
##         else res$bias.correct <- FALSE
##         if (!is.null(tau) & !weighting) {
##             if (tau <= 0 || tau >= 1)
##                 stop("\ntau needs to be between 0 and 1.\n")
##             res$coefficients["(Intercept)"] <- res$coefficients["(Intercept)"] -
##                 log(((1 - tau)/tau) * (ybar/(1 - ybar)))
##             res$prior.correct <- TRUE
##             res$weighting <- FALSE
##         }
##         else res$prior.correct <- FALSE
##         if (is.null(res$weighting))
##             res$weighting <- FALSE
##         res$linear.predictors <- t(res$coefficients) %*% t(X)
##         res$fitted.values <- 1/(1 + exp(-res$linear.predictors))
##         res$zelig <- "Relogit"
##         class(res) <- c("Relogit", "glm")
##         return(res)
##     }
## })(formula = cbind(conflict, 1 - conflict) ~ major + contig +
##     power + maxdem + mindem + years, data = ., tau = 0.002)
##
## Coefficients:
## (Intercept)        major       contig        power       maxdem
##    -8.04923      2.43196      4.10791      1.05357      0.04804
##     mindem        years
##    -0.06412     -0.06293
##
## Degrees of Freedom: 3125 Total (i.e. Null);  3119 Residual
## Null Deviance:       3979
## Residual Deviance: 1869  AIC: 1883
##
## $upper.estimate
##
## Call:  (function (formula, data = sys.parent(), tau = NULL, bias.correct = TRUE,
##     case.control = "prior", ...)
## {
##     mf <- match.call()
##     mf$tau <- mf$bias.correct <- mf$case.control <- NULL
```

```
##     if (!is.null(tau)) {
##         tau <- unique(tau)
##         if (length(case.control) > 1)
##             stop("You can only choose one option for case control correction.")
##         ck1 <- grep("p", case.control)
##         ck2 <- grep("w", case.control)
##         if (length(ck1) == 0 & length(ck2) == 0)
##             stop("choose either case.control = \"prior\" ", "or case.control = \"weighting\"")
##         if (length(ck2) == 0)
##             weighting <- FALSE
##         else weighting <- TRUE
##     }
##     else weighting <- FALSE
##     if (length(tau) > 2)
##         stop("tau must be a vector of length less than or equal to 2")
##     else if (length(tau) == 2) {
##         mf[[1]] <- relogit
##         res <- list()
##         mf$tau <- min(tau)
##         res$lower.estimate <- eval(as.call(mf), parent.frame())
##         mf$tau <- max(tau)
##         res$upper.estimate <- eval(as.call(mf), parent.frame())
##         res$formula <- formula
##         class(res) <- c("Relogit2", "Relogit")
##         return(res)
##     }
##     else {
##         mf[[1]] <- glm
##         mf$family <- binomial(link = "logit")
##         y2 <- model.response(model.frame(mf$formula, data))
##         if (is.matrix(y2))
##             y <- y2[, 1]
##         else y <- y2
##         ybar <- mean(y)
##         if (weighting) {
##             w1 <- tau/ybar
##             w0 <- (1 - tau)/(1 - ybar)
##             wi <- w1 * y + w0 * (1 - y)
##             mf$weights <- wi
##         }
##         res <- eval(as.call(mf), parent.frame())
##         res$call <- match.call(expand.dots = TRUE)
##         res$tau <- tau
##         X <- model.matrix(res)
##         if (bias.correct) {
##             pihat <- fitted(res)
##             if (is.null(tau))
##                 wi <- rep(1, length(y))
##             else if (weighting)
##                 res$weighting <- TRUE
##             else {
##                 w1 <- tau/ybar
##                 w0 <- (1 - tau)/(1 - ybar)
##                 wi <- w1 * y + w0 * (1 - y)
##                 res$weighting <- FALSE
##             }
##             W <- pihat * (1 - pihat) * wi
##             Qdiag <- lm.influence(lm(y ~ X - 1, weights = W))$hat/W
```

```
##             if (is.null(tau))
##                 xi <- 0.5 * Qdiag * (2 * pihat - 1)
##             else xi <- 0.5 * Qdiag * ((1 + w0) * pihat - w0)
##             res$coefficients <- res$coefficients - lm(xi ~ X -
##                 1, weights = W)$coefficients
##             res$bias.correct <- TRUE
##         }
##         else res$bias.correct <- FALSE
##         if (!is.null(tau) & !weighting) {
##             if (tau <= 0 || tau >= 1)
##                 stop("\ntau needs to be between 0 and 1.\n")
##             res$coefficients["(Intercept)"] <- res$coefficients["(Intercept)"] -
##                 log(((1 - tau)/tau) * (ybar/(1 - ybar)))
##             res$prior.correct <- TRUE
##             res$weighting <- FALSE
##         }
##         else res$prior.correct <- FALSE
##         if (is.null(res$weighting))
##             res$weighting <- FALSE
##         res$linear.predictors <- t(res$coefficients) %*% t(X)
##         res$fitted.values <- 1/(1 + exp(-res$linear.predictors))
##         res$zelig <- "Relogit"
##         class(res) <- c("Relogit", "glm")
##         return(res)
##     }
## })(formula = cbind(conflict, 1 - conflict) ~ major + contig +
##     power + maxdem + mindem + years, data = ., tau = 0.005)
##
## Coefficients:
## (Intercept)        major       contig        power       maxdem
##    -7.13001      2.43197      4.10805      1.05358      0.04804
##     mindem        years
##    -0.06413     -0.06294
##
## Degrees of Freedom: 3125 Total (i.e. Null);  3119 Residual
## Null Deviance:       3979
## Residual Deviance: 1869  AIC: 1883
##
## $formula
## cbind(conflict, 1 - conflict) ~ major + contig + power + maxdem +
##     mindem + years
## <environment: 0x7fc26fdce8e0>
##
## attr(,"class")
## [1] "Relogit2" "Relogit"
## Next step: Use 'setx' method
```

Set the explanatory variables to their means:

```
x.out2 <- setx(z.out2)
```

Simulate quantities of interest:

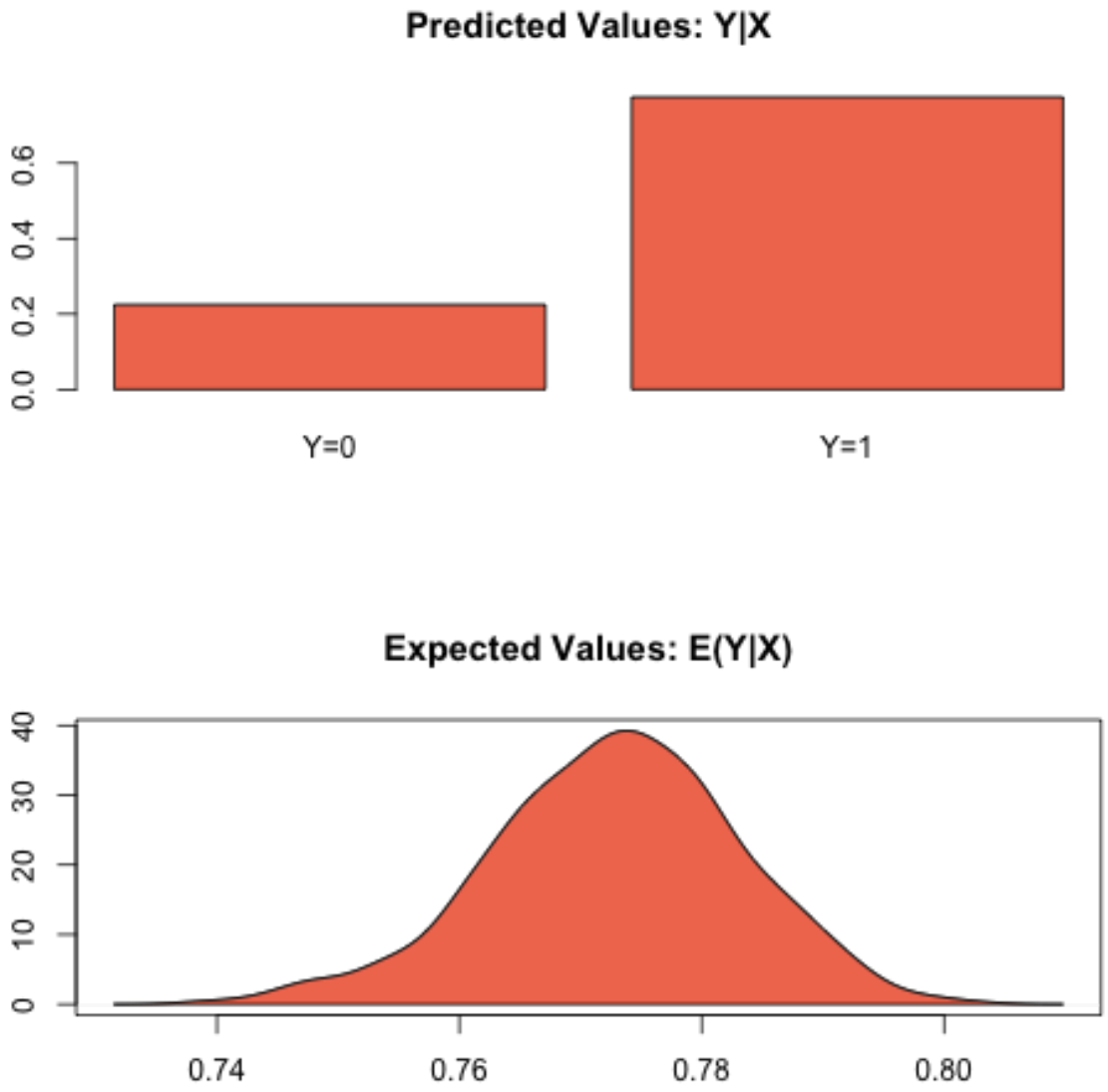
```
s.out <- sim(z.out2, x = x.out2)
```

```
## Error in UseMethod("vcov"): no applicable method for 'vcov' applied to an object of class "c('Rel
```

```
summary(s.out2)
```

```
##
##  sim x :
##  -----
## ev
##           mean         sd       50%      2.5%     97.5%
## [1,] 0.7728179 0.01032537 0.7730781 0.7496465 0.7921772
## pv
##          0     1
## [1,] 0.225 0.775
```

```
plot(s.out2)
```



The cost of giving a range of values for τ is that point estimates are not available for quantities of interest. Instead, quantities are presented as confidence intervals with significance less than or equal to a specified level (e.g., at least

95% of the simulations are contained in the nominal 95% confidence interval).

2.11.6 Model

- Like the standard logistic regression, the *stochastic component* for the rare events logistic regression is:

$$Y_i \sim \text{Bernoulli}(\pi_i),$$

where Y_i is the binary dependent variable, and takes a value of either 0 or 1.

- The *systematic component* is:

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

- If the sample is generated via a case-control (or choice-based) design, such as when drawing all events (or "cases") and a sample from the non-events (or "controls") and going backwards to collect the explanatory variables, you must correct for selecting on the dependent variable. While the slope coefficients are approximately unbiased, the constant term may be significantly biased. Zelig has two methods for case control correction:

  1. The "prior correction" method adjusts the intercept term. Let τ be the true population fraction of events, $\bar{y}$ the fraction of events in the sample, and $\hat{\beta}_0$ the uncorrected intercept term. The corrected intercept β_0 is:

$$\beta = \hat{\beta}_0 - \ln\left[\left(\frac{1-\tau}{\tau}\right)\left(\frac{\bar{y}}{1-\bar{y}}\right)\right].$$

  2. The "weighting" method performs a weighted logistic regression to correct for a case-control sampling design. Let the 1 subscript denote observations for which the dependent variable is observed as a 1, and the 0 subscript denote observations for which the dependent variable is observed as a 0. Then the vector of weights w_i

$$\begin{aligned} w_1 &= \frac{\tau}{\bar{y}} \\ w_0 &= \frac{(1-\tau)}{(1-\bar{y})} \\ w_i &= w_1 Y_i + w_0(1-Y_i) \end{aligned}$$

  If τ is unknown, you may alternatively specify an upper and lower bound for the possible range of τ. In this case, the relogit procedure uses "robust Bayesian" methods to generate a confidence interval (rather than a point estimate) for each quantity of interest. The nominal coverage of the confidence interval is at least as great as the actual coverage.

- By default, estimates of the the coefficients β are bias-corrected to account for finite sample or rare events bias. In addition, quantities of interest, such as predicted probabilities, are also corrected of rare-events bias. If $\widehat{\beta}$ are the uncorrected logit coefficients and bias($\widehat{\beta}$) is the bias term, the corrected coefficients $\tilde{\beta}$ are

$$\widehat{\beta} - \text{bias}(\widehat{\beta}) = \tilde{\beta}$$

  The bias term is

$$\text{bias}(\widehat{\beta}) = (X'WX)^{-1}X'W\xi$$

  where

$$\begin{aligned} \xi_i &= 0.5Q_{ii}\Big((1+w-1)\widehat{\pi}_i - w_1\Big) \\ Q &= X(X'WX)^{-1}X' \\ W &= \text{diag}\{\widehat{\pi}_i(1-\widehat{\pi}_i)w_i\} \end{aligned}$$

  where w_i and w_1 are given in the "weighting" section above.

2.11.7 Quantities of Interest

- For either one or no τ:
  - The expected values (qi$ev) for the rare events logit are simulations of the predicted probability

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

    given draws of β from its posterior.
  - The predicted value (qi$pr) is a draw from a binomial distribution with mean equal to the simulated π_i.
  - The first difference (qi$fd) is defined as

$$\text{FD} = \Pr(Y = 1 \mid x_1, \tau) - \Pr(Y = 1 \mid x, \tau).$$

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

$$\text{RR} = \Pr(Y = 1 \mid x_1, \tau) \;/\; \Pr(Y = 1 \mid x, \tau).$$

- For a range of τ defined by $[\tau_1, \tau_2]$, each of the quantities of interest are $n \times 2$ matrices, which report the lower and upper bounds, respectively, for a confidence interval with nominal coverage at least as great as the actual coverage. At worst, these bounds are conservative estimates for the likely range for each quantity of interest. Please refer to for the specific method of calculating bounded quantities of interest.
- In conditional prediction models, the average expected treatment effect (att.ev) for the treatment group is

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

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $E[Y_i(t_i = 0)]$, the counterfactual expected value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.
- In conditional prediction models, the average predicted treatment effect (att.pr) for the treatment group is

$$\frac{1}{\sum_{i=1}^{n} t_i} \sum_{i:t_i=1}^{n} \left\{ Y_i(t_i = 1) - Y_i(\widehat{t_i = 0}) \right\},$$

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $Y_i(\widehat{t_i = 0})$, the counterfactual predicted value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

2.11.8 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 = relogit, data)`, then you may examine the available information in `z.out` by using `names(z.out)`, see the coefficients by using z.out$coefficients, and a default summary of information through `summary(z.out)`.

2.11.9 Differences with Stata Version

The Stata version of ReLogit and the R implementation differ slightly in their coefficient estimates due to differences in the matrix inversion routines implemented in R and Stata. Zelig uses orthogonal-triangular decomposition (through lm.influence()) to compute the bias term, which is more numerically stable than standard matrix calculations.

2.11.10 See also

2.12 zelig-tobit

Linear Regression for a Left-Censored Dependent Variable

Tobit regression estimates a linear regression model for a left-censored dependent variable, where the dependent variable is censored from below. While the classical tobit model has values censored at 0, you may select another censoring point. For other linear regression models with fully observed dependent variables, see Bayesian regression (), maximum likelihood normal regression (), or least squares ().

2.12.1 Syntax

```
z5 <- ztobit$new()
z5$zelig(Y ~ X1 + X2, below = 0, above = Inf, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

```
z.out <- zelig(Y ~ X1 + X2, below = 0, above = Inf, model = "tobit", data = mydata)
x.out <- setx(z.out)
s.out <- sim(z.out, x = x.out)
```

2.12.2 Inputs

zelig() accepts the following arguments to specify how the dependent variable is censored.

- `below`: (defaults to 0) The point at which the dependent variable is censored from below. If any values in the dependent variable are observed to be less than the censoring point, it is assumed that that particular observation is censored from below at the observed value. (See for a Bayesian implementation that supports both left and right censoring.)
- robust: defaults to FALSE. If TRUE, zelig() computes robust standard errors based on sandwich estimators (see and) and the options selected in cluster.
- cluster: if robust = TRUE, you may select a variable to define groups of correlated observations. Let x3 be a variable that consists of either discrete numeric values, character strings, or factors that define strata. Then

  ```
  > z.out <- zelig(y ~ x1 + x2, robust = TRUE, cluster = "x3",
                   model = "tobit", data = mydata)
  ```

  means that the observations can be correlated within the strata defined by the variable x3, and that robust standard errors should be calculated according to those clusters. If robust = TRUE but cluster is not specified, zelig() assumes that each observation falls into its own cluster.

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

2.12.3 Examples

Basic Example

Attaching the sample dataset:

```
data(tobin)
```

Estimating linear regression using `tobit`:

```
z.out <- zelig(durable ~ age + quant, model = "tobit", data = tobin)

## How to cite this model in Zelig:
##   Kosuke Imai, Gary King, Olivia Lau. 2011.
##   tobit: Linear regression for Left-Censored Dependent Variable
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

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)

##
##  sim x :
##  -----
## ev
##       mean        sd      50%      2.5%    97.5%
## 1 1.585198 0.6748437 1.514415 0.5011679 3.044189
## pv
##          mean       sd      50% 2.5%    97.5%
## [1,] 3.252505 4.272525 1.582813    0 14.40365
```

Simulating First Differences

Set explanatory variables to their default(mean/mode) values, with high (80th percentile) and low (20th percentile) liquidity ratio (`quant`):

```
x.high <- setx(z.out, quant = quantile(tobin$quant, prob = 0.8))
x.low <- setx(z.out, quant = quantile(tobin$quant, prob = 0.2))
```

Estimating the first difference for the effect of high versus low liquidity ratio on duration(`durable`):

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

summary(s.out2)

##
##  sim x :
##  -----
## ev
##       mean        sd      50%      2.5%    97.5%
## 1 1.162711 0.7248203 1.040084 0.1338574 2.950335
## pv
##          mean       sd     50% 2.5%    97.5%
## [1,] 2.763072 3.929297 0.61721    0 12.85066
##
##  sim x1 :
##  -----
```

```
## ev
##       mean        sd      50%      2.5%   97.5%
## 1 2.056761 0.9245484 1.892733 0.6058304 4.08382
## pv
##          mean       sd      50% 2.5%    97.5%
## [1,] 3.496784 4.334128 1.932983    0 14.11736
## fd
##        mean       sd       50%      2.5%    97.5%
## 1 0.8940502 1.123632 0.8431692 -1.251169 3.200797
```

```
plot(s.out1)
```

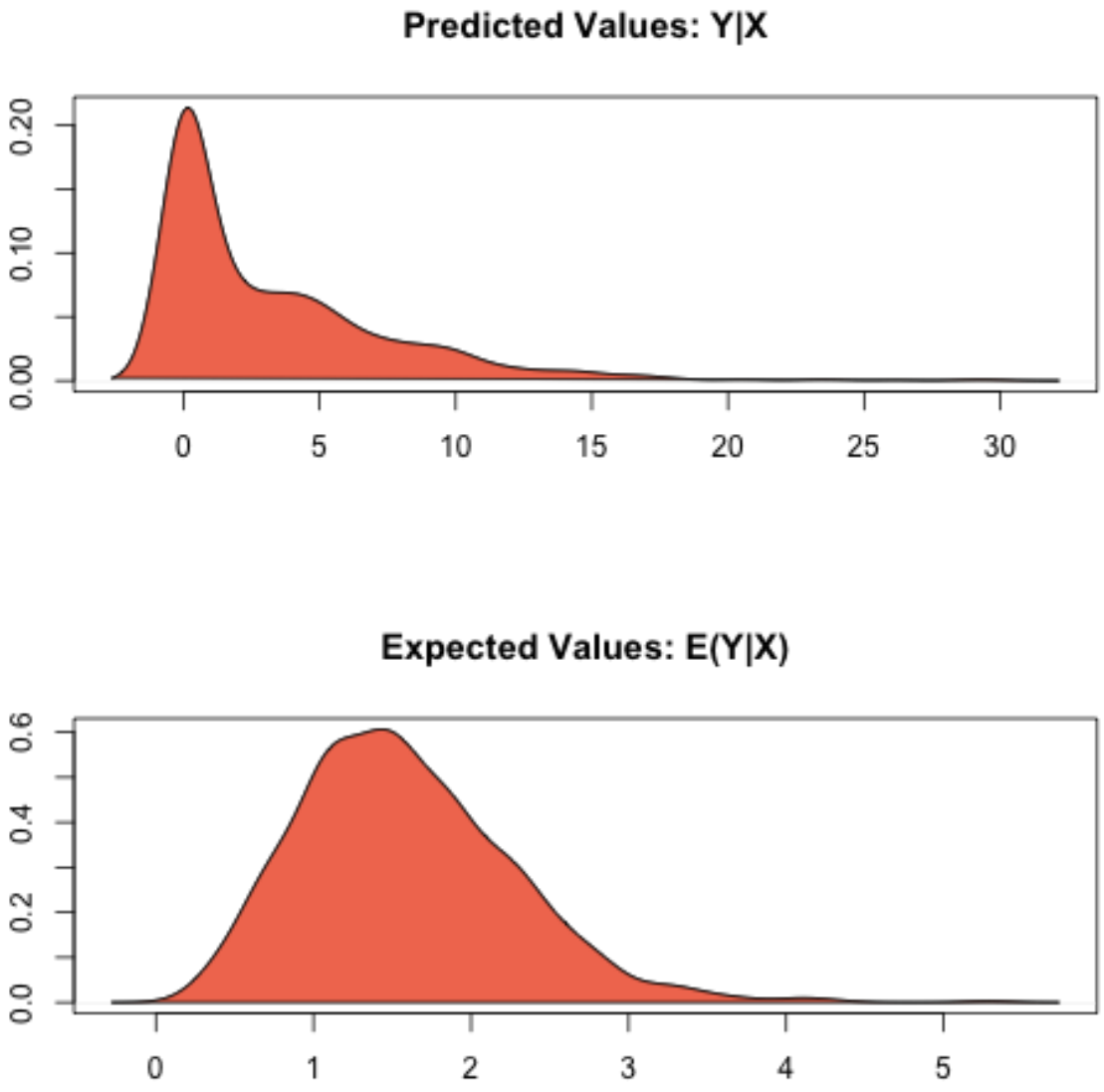


Figure 2.11: Zelig-tobit

2.12.4 Model

- Let Y_i^* be a latent dependent variable which is distributed with *stochastic* component

$$Y_i^* \sim \text{Normal}(\mu_i, \sigma^2)$$

where μ_i is a vector means and σ^2 is a scalar variance parameter. Y_i^* is not directly observed, however. Rather we observed Y_i which is defined as:

$$Y_i = \begin{cases} Y_i^* & \text{if} \quad c < Y_i^* \\ c & \text{if} \quad c \geq Y_i^* \end{cases}$$

where c is the lower bound below which Y_i^* is censored.

- The *systematic component* is given by

$$\mu_i = x_i\beta,$$

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

2.12.5 Quantities of Interest

- The expected values (`qi$ev`) for the tobit regression model are the same as the expected value of $Y*$:

$$E(Y^*|X) = \mu_i = x_i\beta$$

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

$$\text{FD} = E(Y^* \mid x_1) - E(Y^* \mid x).$$

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

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

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

2.12.6 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 = "tobit", data)
```

then you may examine the available information in ‘‘z.out‘.

2.12.7 See also

The tobit function is part of the survival library by Terry Therneau, ported to R by Thomas Lumley. Advanced users may wish to refer to `help(survfit)` in the survival library.

2.13 zelig-factorbayes

Given some unobserved explanatory variables and observed dependent variables, the Normal theory factor analysis model estimates the latent factors. The model is implemented using a Markov Chain Monte Carlo algorithm (Gibbs sampling with data augmentation). For factor analysis with ordinal dependent variables, see ordered factor analysis (), and for a mix of types of dependent variables, see the mixed factor analysis model ().

2.13.1 Syntax

With reference classes:

```
z5 <- zfactorbayes$new()
z5$zelig(cbind(Y1 ,Y2, Y3) ~ NULL, factors = 2,
         model = "factor.bayes", data = mydata)
```

With the Zelig 4 compatibility wrappers:

```
z.out <- zelig(cbind(Y1 ,Y2, Y3) ~ NULL, factors = 2,
               model = "factor.bayes", data = mydata)
```

2.13.2 Inputs

zelig() takes the following functions for factor.bayes:

- `Y1`, `Y2`, and `Y3`: variables of interest in factor analysis (manifest variables), assumed to be normally distributed. The model requires a minimum of three manifest variables.
- `factors`: number of the factors to be fitted (defaults to 2).

2.13.3 Additional Inputs

In addition, zelig() accepts the following additional arguments for model specification:

- `lambda.constraints`: list containing the equality or inequality constraints on the factor loadings. Choose from one of the following forms:
  - varname = list(): by default, no constraints are imposed.
  - `varname = list(d, c)`: constrains the dth loading for the variable named `varname` to be equal to `c`.
  - `varname = list(d, +)`: constrains the dth loading for the variable named `varname` to be positive;
  - `varname = list(d, -)`: constrains the dth loading for the variable named `varname` to be negative.
- `std.var`: defaults to FALSE (manifest variables are rescaled to zero mean, but retain observed variance). If `TRUE`, the manifest variables are rescaled to be mean zero and unit variance.

In addition, zelig() accepts the following additional inputs for bayes.factor:

- `burnin`: number of the initial MCMC iterations to be discarded (defaults to 1,000).
- `mcmc`: number of the MCMC iterations after burnin (defaults to 20,000).
- `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.
- `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 12345.
- `Lambda.start`: starting values of the factor loading matrix Λ, either a scalar (all unconstrained loadings are set to that value), or a matrix with compatible dimensions. The default is `NA`, where the start value are set to be 0 for unconstrained factor loadings, and 0.5 or -0.5 for constrained factor loadings (depending on the nature of the constraints).
- `Psi.start`: starting values for the uniquenesses, either a scalar (the starting values for all diagonal elements of Ψ are set to be this value), or a vector with length equal to the number of manifest variables. In the latter case, the starting values of the diagonal elements of Ψ take the values of `Psi.start`. The default value is `NA` where the starting values of the all the uniquenesses are set to be 0.5.
- `store.lambda`: defaults to TRUE, which stores the posterior draws of the factor loadings.
- `store.scores`: defaults to FALSE. If TRUE, stores the posterior draws of the factor scores. (Storing factor scores may take large amount of memory for a large number of draws or observations.)

The model also accepts the following additional arguments to specify prior parameters:

- `l0`: mean of the Normal prior for the factor loadings, either a scalar or a matrix with the same dimensions as Λ. If a scalar value, that value will be the prior mean for all the factor loadings. Defaults to 0.
- `L0`: precision parameter of the Normal prior for the factor loadings, either a scalar or a matrix with the same dimensions as Λ. If `L0` takes a scalar value, then the precision matrix will be a diagonal matrix with the diagonal elements set to that value. The default value is 0, which leads to an improper prior.
- `a0`: the shape parameter of the Inverse Gamma prior for the uniquenesses is `a0/2`. It can take a scalar value or a vector. The default value is 0.001.
- `b0`: the shape parameter of the Inverse Gamma prior for the uniquenesses is `b0/2`. It can take a scalar value or a vector. The default value is 0.001.

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

2.13.4 Example

Attaching the sample dataset:

```
data(swiss)
names(swiss) <- c("Fert", "Agr", "Exam", "Educ", "Cath", "InfMort")
```

Factor analysis:

```
z.out <- zelig(cbind(Agr, Exam, Educ, Cath, InfMort) ~ NULL,
               model = "factor.bayes", data = swiss, factors = 2, verbose = TRUE,
               a0 = 1, b0 = 0.15, burnin = 5000, mcmc = 50000)
```

Checking for convergence before summarizing the estimates:

```
algor <- try(geweke.diag(z.out$coefficients), silent=T)
if (class(algor) == "try-error")
    print(algor)
```

Since the algorithm did not converge, we now add some constraints on Λ to optimize the algorithm:

```
z.out <- zelig(cbind(Agr, Exam, Educ, Cath, InfMort) ~ NULL,
               model = "factor.bayes", data = swiss, factors = 2,
               lambda.constraints = list(Exam = list(1,"+"),
                   Exam = list(2,"-"), Educ = c(2, 0),
                   InfMort = c(1, 0)),
               verbose = TRUE, a0 = 1, b0 = 0.15,
```

```
                burnin = 5000, mcmc = 50000)
geweke.diag(z.out$coefficients)
heidel.diag(z.out$coefficients)
raftery.diag(z.out$coefficients)
summary(z.out)
```

2.13.5 Model

Suppose for observation i we observe K variables and hypothesize that there are d underlying factors such that:

$$Y_i = \Lambda\phi_i + \epsilon_i$$

where Y_i is the vector of K manifest variables for observation i. Λ is the $K \times d$ factor loading matrix and ϕ_i is the d-vector of latent factor scores. Both Λ and ϕ need to be estimated.

- The *stochastic component* is given by:

  $$\epsilon_i \sim \text{Normal}(0, \Psi).$$

  where Ψ is a diagonal, positive definite matrix. The diagonal elements of Ψ are referred to as uniquenesses.

- The *systematic component* is given by

  $$\mu_i = E(Y_i) = \Lambda\phi_i$$

- The independent conjugate *prior* for each Λ_{ij} is given by

  $$\Lambda_{ij} \sim \text{Normal}(l_{0_{ij}}, L_{0_{ij}}^{-1}) \text{ for } i = 1, \ldots, k; \quad j = 1, \ldots, d.$$

- The independent conjugate *prior* for each Ψ_{ii} is given by

  $$\Psi_{ii} \sim \text{InverseGamma}(\frac{a_0}{2}, \frac{b_0}{2}), \text{ for } i = 1, \ldots, k.$$

- The *prior* for ϕ_i is

  $$\phi_i \sim \quad \text{Normal}(0, I_d), \text{ for } i = 1, \ldots, n.$$

  where I_d is a :math:‘ dtimes d ‘ identity matrix.

2.13.6 Output Values

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

```
z.out <- zelig(cbind(Y1, Y2, Y3), model = "factor.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$coefficients`, and view a default summary of information through `summary(z.out)`. Other elements available through the `$` operator are listed below.

- From the `zelig()` output object `z.out`, you may extract:
  - `coefficients`: draws from the posterior distributions of the estimated factor loadings and the uniquenesses. If `store.scores = TRUE`, the estimated factors scores are also contained in `coefficients`.
  - `data`: the name of the input data frame.

  - `seed`: the random seed used in the model.
- Since there are no explanatory variables, the `sim()` procedure is not applicable for factor analysis models.

2.14 zelig-mlogitbayes

Use Bayesian multinomial logistic regression to model unordered categorical variables. The dependent variable may be in the format of either character strings or integer values. The model is estimated via a random walk Metropolis algorithm or a slice sampler. See for the maximum-likelihood estimation of this model.

2.14.1 Syntax

With reference classes:

```
z5 <- zmlogitbayes$new()
z5$zelig(Y ~ X1 + X2, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

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

2.14.2 Additional Inputs

zelig() accepts the following arguments for mlogit.bayes:

- `baseline`: either a character string or numeric value (equal to one of the observed values in the dependent variable) specifying a baseline category. The default value is `NA` which sets the baseline to the first alphabetical or numerical unique value of the dependent variable.

The model accepts the following additional arguments to monitor the Markov chains:

- `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.
- `mcmc.method`: either "MH" or "slice", specifying whether to use Metropolis Algorithm or slice sampler. The default value is `MH`.
- `tune`: tuning parameter for the Metropolis-Hasting step, either a scalar or a numeric vector (for k coefficients, enter a k vector). The tuning parameter should be set such that the acceptance rate is satisfactory (between 0.2 and 0.5). 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 a vector (for k coefficients, enter a k vector). The default is `NA` where the maximum likelihood estimates are used as the starting values.

Use the following arguments to specify the priors for the model:

- `b0`: prior mean for the coefficients, either a scalar or vector. If a scalar, 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, 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(MCMCmnl)` for more information.

2.14.3 Examples

Basic Example

Attaching the sample dataset:

```
data(mexico)
```

Estimating multinomial logistics regression using `mlogit.bayes`:

```
z.out <- zelig(vote88 ~ pristr + othcok + othsocok,
               model = "mlogit.bayes", data = mexico,
               verbose = FALSE)

## Calculating MLEs and large sample var-cov matrix.
## This may take a moment...
## Inverting Hessian to get large sample var-cov matrix.

## Warning in if (mcmc.method == "RWM") {: the condition has length > 1 and
## only the first element will be used

## Warning in if (mcmc.method == "IndMH") {: the condition has length > 1 and
## only the first element will be used

## How to cite this model in Zelig:
##   Ben Goodrich, Ying Lu. 2013.
##   mlogitbayes: Bayesian Multinomial Logistic Regression for Dependent Variables with Unordered Ca
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

Checking for convergence before summarizing the estimates:

```
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)

##
##  sim x :
##  -----
```

```
## ev
##             mean         sd       50%      2.5%     97.5%
## P(Y=1) 0.5613368 0.01592425 0.5615034 0.5306640 0.5914963
## P(Y=2) 0.2099124 0.01273148 0.2098424 0.1854312 0.2350891
## P(Y=3) 0.2287508 0.01360126 0.2285987 0.2033590 0.2558153
## pv
## qi
##      1      2      3
## 0.5586 0.2083 0.2331
```

Simulating First Differences

Estimating the first difference (and risk ratio) in the probabilities of voting different candidates when `pristr` (the strength of the PRI) is set to be weak (equal to 1) versus strong (equal to 3) while all the other variables held at their default values.

```
x.weak <- setx(z.out, pristr = 1)
x.strong <- setx(z.out, pristr = 3)
s.out2 <- sim(z.out, x = x.strong, x1 = x.weak)
summary(s.out2)
```

```
##
##  sim x :
##  -----
## ev
##             mean         sd       50%      2.5%     97.5%
## P(Y=1) 0.7156880 0.02127842 0.7158103 0.6725681 0.7561260
## P(Y=2) 0.1270237 0.01458077 0.1265905 0.1000858 0.1562571
## P(Y=3) 0.1572883 0.01646202 0.1568142 0.1260809 0.1909916
## pv
## qi
##      1      2      3
## 0.7158 0.1258 0.1584
##
##  sim x1 :
##  -----
## ev
##             mean         sd       50%      2.5%     97.5%
## P(Y=1) 0.4028126 0.02357831 0.4028038 0.3563194 0.4483880
## P(Y=2) 0.3037026 0.02130587 0.3029289 0.2638074 0.3470994
## P(Y=3) 0.2934848 0.02189140 0.2931780 0.2517546 0.3372056
## pv
## qi
##      1      2      3
## 0.4055 0.2996 0.2949
## fd
##              mean         sd        50%        2.5%      97.5%
## P(Y=1) -0.3128754 0.03459857 -0.3128662 -0.38111485 -0.2442630
## P(Y=2)  0.1766789 0.02735176  0.1764581  0.12360341  0.2313796
## P(Y=3)  0.1361965 0.02881430  0.1363242  0.07966018  0.1935930
```

2.14.4 Model

Let Y_i be the (unordered) categorical dependent variable for observation i which takes an integer values $j = 1, \ldots, J$.

- The *stochastic component* is given by:

$$Y_i \sim \quad \text{Multinomial}(Y_i \mid \pi_{ij}).$$

where $\pi_{ij} = \Pr(Y_i = j)$ for $j = 1, \ldots, J$.

- The *systematic component* is given by

$$\pi_{ij} = \frac{\exp(x_i \beta_j)}{\sum_{k=1}^{J} \exp(x_i \beta_k)}, \text{ for } j = 1, \ldots, J-1,$$

where x_i is the vector of k explanatory variables for observation i and β_j is the vector of coefficient for category j. Category J is assumed to be the baseline category.

- The *prior* for β is given by

$$\beta_j \sim \text{Normal}_k \left(b_0, B_0^{-1}\right) \text{ for } j = 1, \ldots, J-1,$$

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).

2.14.5 Quantities of Interest

- The expected values (`qi$ev`) for the multinomial logistics regression model are the predicted probability of belonging to each category:

$$\Pr(Y_i = j) = \pi_{ij} = \frac{\exp(x_i \beta_j)}{\sum_{k=1}^{J} \exp(x_J \beta_k)}, \quad \text{for } j = 1, \ldots, J-1,$$

and

$$\Pr(Y_i = J) = 1 - \sum_{j=1}^{J-1} \Pr(Y_i = j)$$

given the posterior draws of β_j for all categories from the MCMC iterations.

- The predicted values (`qi$pr`) are the draws of Y_i from a multinomial distribution whose parameters are the expected values(`qi$ev`) computed based on the posterior draws of β from the MCMC iterations.

- The first difference (`qi$fd`) in category j for the multinomial logistic model is defined as

$$\text{FD}_j = \Pr(Y_i = j \mid X_1) - \Pr(Y_i = j \mid X).$$

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

$$\text{RR}_j = \Pr(Y_i = j \mid X_1) \; / \; \Pr(Y_i = j \mid X).$$

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

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

where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups, and n_j is the number of treated observations in category j.

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

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

where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups, and n_j is the number of treated observations in category j.

2.14.6 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 = "mlogit.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$coefficients`, and view a default summary of information through `summary(z.out)`. Other elements available through the `$` operator are listed below.

2.14.7 See also

Bayesian logistic regression is part of the MCMCpack library by Andrew D. Martin and Kevin M. Quinn . The convergence diagnostics are part of the CODA library by Martyn Plummer, Nicky Best, Kate Cowles, Karen Vines, Deepayan Sarkar, Russell Almond.

2.15 zelig-oprobitbayes

Use the ordinal probit regression model if your dependent variables are ordered and categorical. They may take either integer values or character strings. The model is estimated using a Gibbs sampler with data augmentation. For a maximum-likelihood implementation of this models, see *probit*.

2.15.1 Syntax

With reference classes:

```
z5 <- zoprobitbayes$new()
z5$zelig(Y ~ X1 + X2, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

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

2.15.2 Additional Inputs

zelig() accepts 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 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`: tuning parameter for the Metropolis-Hasting step. The default value is `NA` which corresponds to 0.05 divided by the number of categories in the response variable.
- `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 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`, which uses the maximum likelihood estimates 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 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(MCMCoprobit)` for more information.

2.15.3 Examples

Basic Example

Attaching the sample dataset:

```
data(sanction)
```

Estimating ordered probit regression using `oprobit.bayes`:

```
z.out <- zelig(ncost ~ mil + coop, model = "oprobit.bayes",
               data = sanction, verbose = FALSE)

## Warning in model.response(mf, "numeric"): using type = "numeric" with a
## factor response will be ignored

## How to cite this model in Zelig:
##   Ben Goodrich, Ying Lu. 2013.
##   oprobitbayes: Bayesian Probit Regression for Dichotomous Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

Creating an ordered dependent variable:

```
sanction$ncost <- factor(sanction ~ ncost, ordered = TRUE,
                         levels = c("net gain", "little effect", "modest loss",
                                    "major loss"))
```

```
## Error in as.vector(x, mode): invalid 'mode' argument
```

Checking for convergence before summarizing the estimates:

```
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)

##
##  sim x :
##  -----
## ev
##                      mean         sd        50%       2.5%      97.5%
## little effect 0.44981581 0.05601155 0.44883053 0.34151148 0.56103192
## major loss    0.04473004 0.02101827 0.04176495 0.01271310 0.09412959
## modest loss   0.12341501 0.03950140 0.11984735 0.06005967 0.22834556
## net gain      0.38203914 0.05548348 0.38097907 0.27602445 0.49205957
## pv
## qi
## little effect    major loss   modest loss      net gain
##        0.1868        0.2731        0.5213        0.0188
```

Simulating First Differences

Estimating the first difference (and risk ratio) in the probabilities of incurring different level of cost when there is no military action versus military action while all the other variables held at their default values.

```
x.high <- setx(z.out, mil = 0)
x.low <- setx(z.out, mil = 1)

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

##
##  sim x :
##  -----
## ev
##                      mean         sd        50%       2.5%      97.5%
## little effect 0.43844669 0.05843957 0.43767439 0.32758262 0.55410854
## major loss    0.04458012 0.02095061 0.04165022 0.01271633 0.09387773
## modest loss   0.12377654 0.03963082 0.12024928 0.05993637 0.22927426
## net gain      0.39319665 0.05795514 0.39196529 0.28336346 0.50825527
## pv
## qi
## little effect    major loss   modest loss      net gain
##        0.1491        0.2382        0.5780        0.0347
##
##  sim x1 :
```

```
##  -----
## ev
##                     mean         sd        50%       2.5%      97.5%
## little effect 0.5464229 0.16109327 0.54796938 0.23474072 0.84451223
## major loss    0.0407613 0.01998891 0.03763614 0.01085385 0.08828575
## modest loss   0.1075956 0.03975880 0.10421634 0.04208043 0.20132712
## net gain      0.3052203 0.14485132 0.29018143 0.07362204 0.62315150
## pv
## qi
## little effect    major loss   modest loss      net gain
##        0.6116        0.0963        0.1862        0.1059
## fd
##                        mean         sd          50%        2.5%       97.5%
## little effect  0.107976200 0.17020693  0.111082214 -0.22740084 0.426282862
## major loss    -0.003818825 0.00665593 -0.001418880 -0.02253975 0.002180789
## modest loss   -0.016180976 0.02172754 -0.008327805 -0.07580491 0.005015329
## net gain      -0.087976398 0.15275950 -0.102164237 -0.34480882 0.241227653
```

2.15.4 Model

Let Y_i be the ordered categorical dependent variable for observation i which takes an integer value $j = 1, \ldots, J$.

- The *stochastic component* is described by an unobserved continuous variable, Y_i^*,

$$Y_i^* \sim \text{Normal}(\mu_i, 1).$$

  Instead of Y_i^*, we observe categorical variable Y_i,

$$Y_i = j \quad \text{if } \tau_{j-1} \leq Y_i^* \leq \tau_j \text{ for } j = 1, \ldots, J.$$

  where τ_j for $j = 0, \ldots, J$ are the threshold parameters with the following constraints, $\tau_l < \tau_m$ for $l < m$, and $\tau_0 = -\infty, \tau_J = \infty$.

  The probability of observing Y_i equal to category j is,

$$\Pr(Y_i = j) = \quad \Phi(\tau_j \mid \mu_i) - \Phi(\tau_{j-1} \mid \mu_i) \text{ for } j = 1, \ldots, J$$

  where $\Phi(\cdot \mid \mu_i)$ is the cumulative distribution function of the Normal distribution with mean μ_i and variance 1.

- The *systematic component* is given by

$$\mu_i = x_i \beta,$$

  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).

2.15.5 Quantities of Interest

- The expected values (`qi$ev`) for the ordered probit model are the predicted probability of belonging to each category:

$$\Pr(Y_i = j) = \Phi(\tau_j \mid x_i \beta) - \Phi(\tau_{j-1} \mid x_i \beta),$$

  given the posterior draws of β and threshold parameters τ from the MCMC iterations.

- The predicted values (`qi$pr`) are the observed values of Y_i given the observation scheme and the posterior draws of β and cut points τ from the MCMC iterations.
- The first difference (`qi$fd`) in category j for the ordered probit model is defined as

$$\text{FD}_j = \Pr(Y_i = j \mid X_1) - \Pr(Y_i = j \mid X).$$

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

$$\text{RR}_j = \Pr(Y_i = j \mid X_1) \; / \; \Pr(Y_i = j \mid X).$$

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

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

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups, and n_j is the number of observations in the treatment group that belong to category j.
- In conditional prediction models, the average predicted treatment effect (`qi$att.pr`) for the treatment group in category j is

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

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups, and n_j is the number of observations in the treatment group that belong to category j.

2.15.6 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 = "oprobit.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$coefficients`, and view a default summary of information through `summary(z.out)`. Other elements available through the `$` operator are listed below.

- From the `zelig()` output object `z.out`, you may extract:
  - `coefficients`: draws from the posterior distributions of the estimated coefficients β and threshold parameters τ. Note, element τ_1 is normalized to 0 and is not returned in the `coefficients` object.
  - 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$ev`: the simulated expected values (probabilities) of each of the J categories for the specified values of `x`.
  - `qi$pr`: the simulated predicted values (observed values) for the specified values of `x`.
  - `qi$fd`: the simulated first difference in the expected values of each of the J categories for the values specified in `x` and `x1`.
  - `qi$rr`: the simulated risk ratio for the expected values of each of the J categories 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.

2.15.7 See also

Bayesian ordinal probit regression is part of the MCMCpack library by Andrew D. Martin and Kevin M. Quinn . The convergence diagnostics are part of the CODA library by Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines.

2.16 zelig-poissonbayes

Use the Poisson regression model if the observations of your dependent variable represents the number of independent events that occur during a fixed period of time. The model is fit using a random walk Metropolis algorithm. For a maximum-likelihood estimation of this model see *poisson*.

2.16.1 Syntax

With reference classes:

```
z5 <- zpoissonbayes$new()
z5$zelig(Y ~ X1 + X2, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

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

2.16.2 Additional Inputs

Use the following argument 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). The default value is 1.1.
- `verbose`: default 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, 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 the coefficients) or a scalar. If a scalar, 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(MCMCpoisson)` for more information.

2.16.3 Examples

Basic Example

Attaching the sample dataset:

```
data(sanction)
```

Estimating the Poisson regression using `poisson.bayes`:

```
z.out <- zelig(num ~ target + coop, model = "poisson.bayes",
               data = sanction, verbose = FALSE)

## How to cite this model in Zelig:
##   Ben Goodrich, Ying Lu. 2013.
##   poissonbayes: Bayesian Poisson Regression
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

Checking 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)
```

Simulating First Differences

Estimating the first difference in the number of countries imposing sanctions when the number of targets is set to be its maximum versus its minimum :

```
x.max <- setx(z.out, target = max(sanction$target))
x.min <- setx(z.out, target = min(sanction$target))

s.out2 <- sim(z.out, x = x.max, x1 = x.min)
summary(s.out2)

##
##  sim x :
##  -----
## ev
##          mean        sd      50%     2.5%    97.5%
## [1,] 3.191614 0.2936585 3.183013 2.642371 3.803733
## pv
##        mean       sd 50% 2.5% 97.5%
## [1,] 3.2091 1.805409   3    0     7
##
##  sim x1 :
##  -----
## ev
##          mean        sd      50%     2.5%    97.5%
## [1,] 3.306252 0.3059862 3.300095 2.729466 3.944022
## pv
##        mean       sd 50% 2.5% 97.5%
## [1,] 3.3227 1.851353   3    0     7
## fd
##           mean        sd       50%       2.5%     97.5%
## [1,] 0.1146376 0.3671544 0.1265036 -0.6072035 0.8342282
```

2.16.4 Model

Let Y_i be the number of independent events that occur during a fixed time period.

- The *stochastic component* is given by

$$Y_i \sim \quad \text{Poisson}(\lambda_i)$$

  where λ_i is the mean and variance parameter.

- The *systematic component* is given by

$$\lambda_i = \exp(x_i \beta)$$

  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).

2.16.5 Quantities of Interest

- The expected values (`qi$ev`) for the Poisson model are calculated as following:

$$E(Y \mid X) = \lambda_i = \exp(x_i \beta),$$

  given the posterior draws of β based on the MCMC iterations.

- The predicted values (`qi$pr`) are draws from the Poisson distribution with parameter λ_i.
- The first difference (`qi$fd`) for the Poisson model is defined as

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

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

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

  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

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

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

2.16.6 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 = "poisson.bayes", data)
```

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$coefficients`, and view a default summary of information through `summary(z.out)`. Other elements available through the `$` 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$ev`: the simulated expected values for the specified values of `x`.
  - `qi$pr`: the simulated predicted values for the specified values of `x`.
  - `qi$fd`: the simulated first difference in the expected values for the values specified in `x` and `x1`.
  - `qi$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.

2.16.7 See also

Bayesian poisson regression is part of the MCMCpack library by Andrew D. Martin and Kevin M. Quinn . The convergence diagnostics are part of the CODA library by Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines.

2.17 zelig-probitbayes

Use the probit regression model for model binary dependent variables specified as a function of a set of explanatory variables. The model is estimated using a Gibbs sampler. For other models suitable for binary response variables, see Bayesian logistic regression, maximum likelihood logit regression, and maximum likelihood probit regression.

2.17.1 Syntax

With reference classes:

```
z5 <- zprobitbayes$new()
z5$zelig(Y ~ X1 + X2, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

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

2.17.2 Additional Inputs

Using the following arguments to monitor the Markov chains:

- `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.
- `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 the 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.

Use the following arguments to specify optional output for the model:

- `bayes.resid`: defaults to FALSE. If TRUE, the latent Bayesian residuals for all observations are returned. Alternatively, users can specify a vector of observations for which the latent residuals should be returned.

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

2.17.3 Examples

Basic Example

Attaching the sample dataset:

```
data(turnout)
```

Estimating the probit regression using `probit.bayes`:

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

## How to cite this model in Zelig:
##   Ben Goodrich, Ying Lu. 2013.
##   probitbayes: Bayesian Probit Regression for Dichotomous Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

Checking for convergence 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)
```

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 are 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)
```

2.17.4 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

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

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

- The *systematic component* is given by

$$\pi_i = \Phi(x_i\beta),$$

  where $\Phi(\cdot)$ is the cumulative density function of the standard Normal distribution with mean 0 and variance 1, 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).

2.17.5 Quantities of Interest

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

$$E(Y \mid X) = \pi_i = \Phi(x_i\beta),$$

  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 probit 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

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

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

2.17.6 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 = "probit.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$coefficients`, and view a default summary of information through `summary(z.out)`. Other elements available through the `$` 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.
  - `bayes.residuals`: When `bayes.residual` is `TRUE` or a set of observation numbers is given, this object contains the posterior draws of the latent Bayesian residuals of all the observations or the observations specified by the user.
  - `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.

2.17.7 See also

Bayesian probit regression is part of the MCMCpack library by Andrew D. Martin and Kevin M. Quinn . The convergence diagnostics are part of the CODA library by Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines.

2.18 zelig-tobitbayes

Bayesian tobit regression estimates a linear regression model with a censored dependent variable using a Gibbs sampler. The dependent variable may be censored from below and/or from above. For other linear regression models with fully observed dependent variables, see Bayesian regression, maximum likelihood normal regression, or least squares.

2.18.1 Syntax

With reference classes:

```
z5 <- zprobitbayes$new()
z5$zelig((Y ~ X1 + X2, below = 0, above = Inf, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

```
z.out <- zelig(Y ~ X1 + X2, below = 0, above = Inf,
               model = "tobit.bayes", data = mydata)
x.out <- setx(z.out)
s.out <- sim(z.out, x = x.out)
```

2.18.2 Inputs

zelig() accepts the following arguments to specify how the dependent variable is censored.

- `below`: point at which the dependent variable is censored from below. If the dependent variable is only censored from above, set `below = -Inf`. The default value is 0.
- `above`: point at which the dependent variable is censored from above. If the dependent variable is only censored from below, set `above = Inf`. The default value is `Inf`.

2.18.3 Additional Inputs

Use the following arguments to monitor the convergence of 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.
- `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 least squares 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, that value will be the prior mean for all coefficients. The default is 0.
- `B0`: prior precision parameter for the coefficients, either a square matrix (with the dimensions equal to the number of the coefficients) or a scalar. If a scalar, that value times an identity matrix will be the prior precision parameter. The default is 0, which leads to an improper prior.
- `c0`: `c0/2` is the shape parameter for the Inverse Gamma prior on the variance of the disturbance terms.
- `d0`: `d0/2` is the scale parameter for the Inverse Gamma prior on the variance of the disturbance terms.

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

2.18.4 Examples

Basic Example

Attaching the sample dataset:

```
data(tobin)
```

Estimating linear regression using `tobit.bayes`:

```
z.out <- zelig(durable ~ age + quant, model = "tobit.bayes",
               data = tobin, verbose = FALSE)

## Error in zelig(durable ~ age + quant, model = "tobit.bayes", data = tobin, : Model 'tobit.bayes' r
```

Checking for convergence 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)
```

Simulating First Differences

Set explanatory variables to their default(mean/mode) values, with high (80th percentile) and low (20th percentile) liquidity ratio (`quant`):

```
x.high <- setx(z.out, quant = quantile(tobin$quant, prob = 0.8))
x.low <- setx(z.out, quant = quantile(tobin$quant, prob = 0.2))
```

Estimating the first difference for the effect of high versus low liquidity ratio on duration(`durable`):

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

summary(s.out2)
```

2.18.5 Model

Let Y_i^* be the dependent variable which is not directly observed. Instead, we observe Y_i which is defined as following:

$$Y_i = \begin{cases} Y_i^* & \text{if} \quad c_1 < Y_i^* < c_2 \\ c_1 & \text{if} \quad c_1 \geq Y_i^* \\ c_2 & \text{if} \quad c_2 \leq Y_i^* \end{cases}$$

where c_1 is the lower bound below which Y_i^* is censored, and c_2 is the upper bound above which Y_i^* is censored.

- The *stochastic component* is given by

$$\epsilon_i \sim \quad \text{Normal}(0, \sigma^2)$$

where $\epsilon_i = Y_i^* - \mu_i$.

- The *systematic component* is given by

$$\mu_i = x_i\beta,$$

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

- The *semi-conjugate priors* for β and σ^2 are given by

$$\begin{aligned} \beta \sim & \quad \text{Normal}_k\left(b_0, B_0^{-1}\right) \\ \sigma^2 \sim & \quad \text{InverseGamma}\left(\frac{c_0}{2}, \frac{d_0}{2}\right) \end{aligned}$$

where b_0 is the vector of means for the k explanatory variables, B_0 is the $k \times k$ precision matrix (the inverse of a variance-covariance matrix), and $c_0/2$ and $d_0/2$ are the shape and scale parameters for σ^2. Note that β and σ^2 are assumed *a priori* independent.

2.18.6 Quantities of Interest

- The expected values (`qi$ev`) for the tobit regression model is calculated as following. Let

$$\begin{aligned} \Phi_1 = & \quad \Phi\left(\frac{(c_1 - x\beta)}{\sigma}\right) \\ \Phi_2 = & \quad \Phi\left(\frac{(c_2 - x\beta)}{\sigma}\right) \\ \phi_1 = & \quad \phi\left(\frac{(c_1 - x\beta)}{\sigma}\right) \\ \phi_2 = & \quad \phi\left(\frac{(c_2 - x\beta)}{\sigma}\right) \end{aligned}$$

where $\Phi(\cdot)$ is the (cumulative) Normal density function and $\phi(\cdot)$ is the Normal probability density function of the standard normal distribution. Then the expected values are

$$\begin{aligned} E(Y|x) = & \quad P(Y^* \leq c_1|x)c_1 + P(c_1 < Y^* < c_2|x)E(Y^* \mid c_1 < Y^* < c_2, x) + P(Y^* \geq c_2)c_2 \\ = & \quad \Phi_1 c_1 + x\beta(\Phi_2 - \Phi_1) + \sigma(\phi_1 - \phi_2) + (1 - \Phi_2)c_2, \end{aligned}$$

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

$$\text{FD} = E(Y \mid x_1) - E(Y \mid x).$$

- 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.

2.18.7 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 = "tobit.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$coefficients`, and view a default summary of information through `summary(z.out)`. Other elements available through the `$` operator are listed below.

- From the `zelig()` output object `z.out`, you may extract:
  - `coefficients`: draws from the posterior distributions of the estimated parameters. The first k columns contain the posterior draws of the coefficients β, and the last column contains the posterior draws of the variance σ^2.
  - 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$ev`: the simulated expected value for the specified values of `x`.
  - `qi$fd`: the simulated first difference in the expected values given the values specified in `x` and `x1`.
  - `qi$att.ev`: the simulated average expected treatment effect for the treated from conditional prediction models.

2.18.8 See also

Bayesian tobit regression is part of the MCMCpack library by Andrew D. Martin and Kevin M. Quinn . The convergence diagnostics are part of the CODA library by Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines.

2.19 zelig-gammagee

The GEE gamma is similar to standard gamma regression (appropriate when you have an uncensored, positive-valued, continuous dependent variable such as the time until a parliamentary cabinet falls). Unlike in gamma regression, GEE gamma allows for dependence within clusters, such as in longitudinal data, although its use is not limited to just panel data. GEE models make no distributional assumptions but require three specifications: a mean function, a variance function, and a "working" correlation matrix for the clusters, which models the dependence of each observation with other observations in the same cluster. The "working" correlation matrix is a $T \times T$ matrix of correlations, where T is the size of the largest cluster and the elements of the matrix are correlations between within-cluster observations. The appeal of GEE models is that it gives consistent estimates of the parameters and consistent estimates of the standard errors can be obtained using a robust "sandwich" estimator even if the "working" correlation matrix is incorrectly specified. If the "working" correlation matrix is correctly specified, GEE models will give more efficient estimates of the parameters. GEE models measure population-averaged effects as opposed to cluster-specific effects.

2.19.1 Syntax

With reference classes:

```
z5 <- zgammagee$new()
z5$zelig(Y ~ X1 + X2, model = "gamma.gee",
         id = "X3", data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

```
z.out <- zelig(Y ~ X1 + X2, model = "gamma.gee",
               id = "X3", data = mydata)
x.out <- setx(z.out)
s.out <- sim(z.out, x = x.out)
```

where `id` is a variable which identifies the clusters. The data should be sorted by `id` and should be ordered within each cluster when appropriate.

2.19.2 Additional Inputs

Use the following arguments to specify the structure of the "working" correlations within clusters:

- `corstr`: character string specifying the correlation structure: "independence", "exchangeable", "ar1", "unstructured" and "userdefined"
- See `geeglm` in package `geepack` for other function arguments.

2.19.3 Examples

Example with Exchangeable Dependence

Attaching the sample turnout dataset:

```
data(coalition)
```

Sorted variable identifying clusters

```
coalition$cluster <- c(rep(c(1:62), 5),rep(c(63), 4))
sorted.coalition <- coalition[order(coalition$cluster), ]
```

Estimating model and presenting summary:

```
z.out <- zelig(duration ~ fract + numst2, model = "gamma.gee",
               id = "cluster", data = sorted.coalition,
               corstr = "exchangeable")

## How to cite this model in Zelig:
##   Patrick Lam. 2011.
##   zgammagee: General Estimating Equation for Gamma Regression
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/

summary(z.out)

## Model:
## $by
## [1] 1
##
##
## Call:
## geepack::geeglm(formula = duration ~ fract + numst2, family = Gamma("inverse"),
##     data = ., id = c(1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3,
##     3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 7, 7, 7,
##     7, 7, 8, 8, 8, 8, 8, 9, 9, 9, 9, 9, 10, 10, 10, 10, 10, 11,
##     11, 11, 11, 11, 12, 12, 12, 12, 12, 13, 13, 13, 13, 13, 14,
```

```
##     14, 14, 14, 14, 15, 15, 15, 15, 15, 16, 16, 16, 16, 16, 17,
##     17, 17, 17, 17, 18, 18, 18, 18, 18, 19, 19, 19, 19, 19, 20,
##     20, 20, 20, 20, 21, 21, 21, 21, 21, 22, 22, 22, 22, 22, 23,
##     23, 23, 23, 23, 24, 24, 24, 24, 24, 25, 25, 25, 25, 25, 26,
##     26, 26, 26, 26, 27, 27, 27, 27, 27, 28, 28, 28, 28, 28, 29,
##     29, 29, 29, 29, 30, 30, 30, 30, 30, 31, 31, 31, 31, 31, 32,
##     32, 32, 32, 32, 33, 33, 33, 33, 33, 34, 34, 34, 34, 34, 35,
##     35, 35, 35, 35, 36, 36, 36, 36, 36, 37, 37, 37, 37, 37, 38,
##     38, 38, 38, 38, 39, 39, 39, 39, 39, 40, 40, 40, 40, 40, 41,
##     41, 41, 41, 41, 42, 42, 42, 42, 42, 43, 43, 43, 43, 43, 44,
##     44, 44, 44, 44, 45, 45, 45, 45, 45, 46, 46, 46, 46, 46, 47,
##     47, 47, 47, 47, 48, 48, 48, 48, 48, 49, 49, 49, 49, 49, 50,
##     50, 50, 50, 50, 51, 51, 51, 51, 51, 52, 52, 52, 52, 52, 53,
##     53, 53, 53, 53, 54, 54, 54, 54, 54, 55, 55, 55, 55, 55, 56,
##     56, 56, 56, 56, 57, 57, 57, 57, 57, 58, 58, 58, 58, 58, 59,
##     59, 59, 59, 59, 60, 60, 60, 60, 60, 61, 61, 61, 61, 61, 62,
##     62, 62, 62, 62, 63, 63, 63, 63), corstr = "exchangeable")
##
## Coefficients:
##   (Intercept)         fract        numst2
## -0.0129634262  0.0001149139 -0.0174009664
##
## Degrees of Freedom: 314 Total (i.e. Null);  311 Residual
##
## Scale Link:                   identity
## Estimated Scale Parameters:  [1] 0.6231419
##
## Correlation:  Structure = exchangeable    Link = identity
## Estimated Correlation Parameters:
##        alpha
## -0.008086333
##
## Number of clusters:   63   Maximum cluster size: 5
##
## Next step: Use 'setx' method
```

Setting the explanatory variables at their default values (mode for factor variables and mean for non-factor variables), with numst2 set to the vector 0 = no crisis, 1 = crisis.

```
x.low <- setx(z.out, numst2 = 0)
x.high <- setx(z.out, numst2 = 1)
```

Simulate quantities of interest

```
s.out <- sim(z.out, x = x.low, x1 = x.high)
summary(s.out)
```

```
##
##  sim x :
##  -----
## ev
##          mean       sd     50%     2.5%    97.5%
## [1,] 14.40728 1.150051 14.3262 12.36709 16.89536
## pv
##          mean       sd      50%       2.5%    97.5%
## [1,] 14.07835 18.14224 7.603815 0.06235246 65.07778
##
##  sim x1 :
##  -----
```

```
## ev
##          mean       sd      50%    2.5%    97.5%
## [1,] 19.21808 1.117628 19.18863 17.2676 21.63706
## pv
##          mean       sd      50%       2.5%    97.5%
## [1,] 19.61789 23.70813 11.10875 0.04327947 82.02938
## fd
##          mean       sd      50%     2.5%   97.5%
## [1,] 4.810795 1.677545 4.759365 1.566173 8.00721
```

Generate a plot of quantities of interest:

```
plot(s.out)
```

2.19.4 The Model

Suppose we have a panel dataset, with Y_{it} denoting the positive-valued, continuous dependent variable for unit i at time t. Y_i is a vector or cluster of correlated data where y_{it} is correlated with $y_{it'}$ for some or all t, t'. Note that the model assumes correlations within i but independence across i.

- The *stochastic component* is given by the joint and marginal distributions

$$\begin{aligned} Y_i \sim & \quad f(y_i \mid \lambda_i) \\ Y_{it} \sim & \quad g(y_{it} \mid \lambda_{it}) \end{aligned}$$

  where f and g are unspecified distributions with means λ_i and λ_{it}. GEE models make no distributional assumptions and only require three specifications: a mean function, a variance function, and a correlation structure.

- The *systematic component* is the *mean function*, given by:

$$\lambda_{it} = \frac{1}{x_{it}\beta}$$

  where x_{it} is the vector of k explanatory variables for unit i at time t and β is the vector of coefficients.

- The *variance function* is given by:

$$V_{it} = \lambda_{it}^2 = \frac{1}{(x_{it}\beta)^2}$$

- The *correlation structure* is defined by a $T \times T$ "working" correlation matrix, where T is the size of the largest cluster. Users must specify the structure of the "working" correlation matrix *a priori*. The "working" correlation matrix then enters the variance term for each i, given by:

$$V_i = \phi\, A_i^{\frac{1}{2}} R_i(\alpha) A_i^{\frac{1}{2}}$$

  where A_i is a $T \times T$ diagonal matrix with the variance function $V_{it} = \lambda_{it}^2$ as the tth diagonal element, $R_i(\alpha)$ is the "working" correlation matrix, and ϕ is a scale parameter. The parameters are then estimated via a quasi-likelihood approach.

- In GEE models, if the mean is correctly specified, but the variance and correlation structure are incorrectly specified, then GEE models provide consistent estimates of the parameters and thus the mean function as well, while consistent estimates of the standard errors can be obtained via a robust "sandwich" estimator. Similarly, if the mean and variance are correctly specified but the correlation structure is incorrectly specified, the parameters can be estimated consistently and the standard errors can be estimated consistently with the sandwich estimator. If all three are specified correctly, then the estimates of the parameters are more efficient.

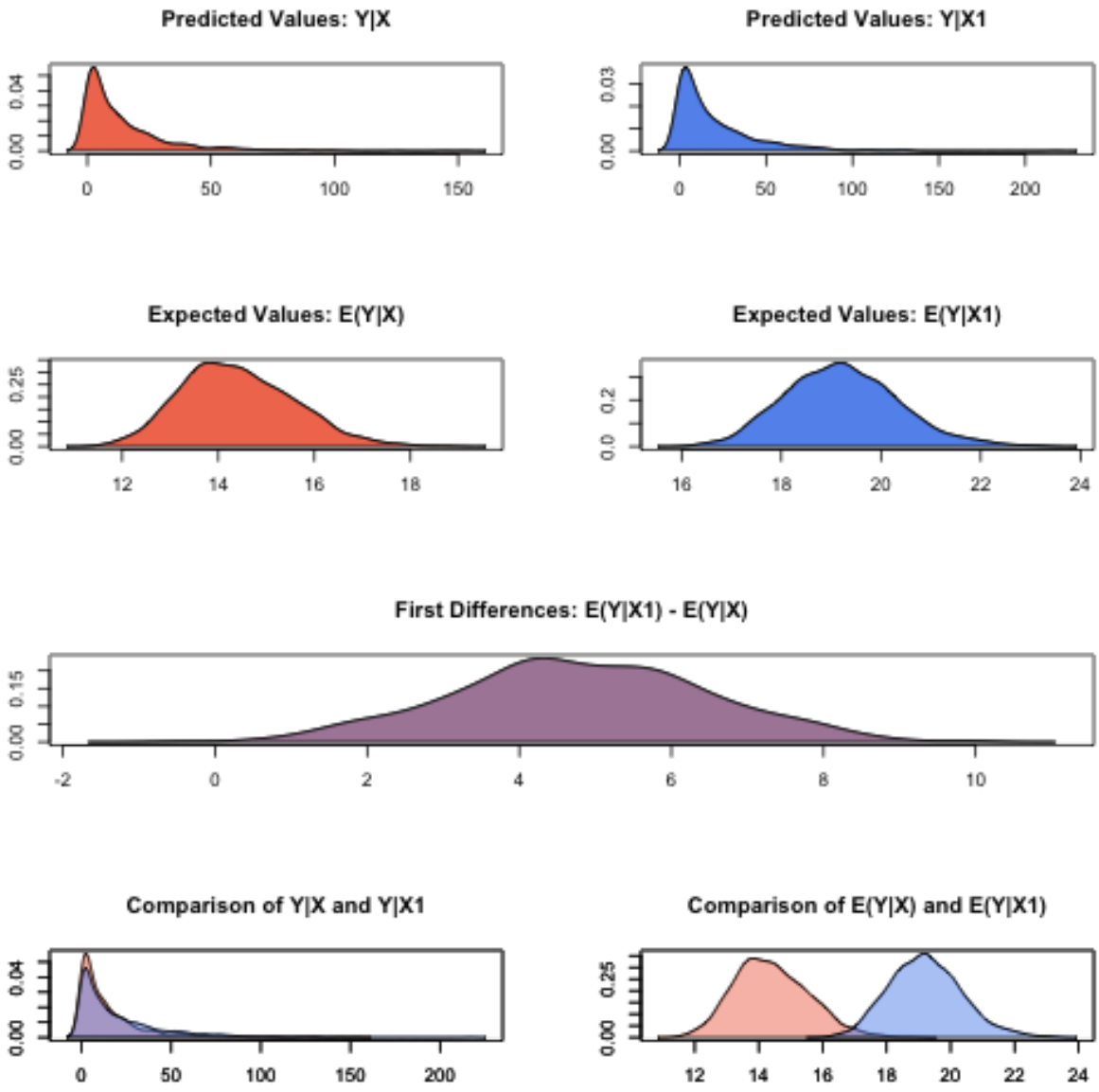


Figure 2.12: plot of chunk unnamed-chunk-10

2.19.5 Quantities of Interest

- All quantities of interest are for marginal means rather than joint means.
- The method of bootstrapping generally should not be used in GEE models. If you must bootstrap, bootstrapping should be done within clusters, which is not currently supported in Zelig. For conditional prediction models, data should be matched within clusters.
- The expected values (qi$ev) for the GEE gamma model is the mean:

$$E(Y) = \lambda_c = \frac{1}{x_c \beta},$$

  given draws of β from its sampling distribution, where x_c is a vector of values, one for each independent variable, chosen by the user.
- The first difference (qi$fd) for the GEE gamma model is defined as

$$\text{FD} = \Pr(Y = 1 \mid x_1) - \Pr(Y = 1 \mid x).$$

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

$$\frac{1}{\sum_{i=1}^{n} \sum_{t=1}^{T} tr_{it}} \sum_{i:tr_{it}=1}^{n} \sum_{t:tr_{it}=1}^{T} \left\{ Y_{it}(tr_{it} = 1) - E[Y_{it}(tr_{it} = 0)] \right\},$$

  where tr_{it} is a binary explanatory variable defining the treatment ($tr_{it} = 1$) and control ($tr_{it} = 0$) groups. Variation in the simulations are due to uncertainty in simulating $E[Y_{it}(tr_{it} = 0)]$, the counterfactual expected value of Y_{it} for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $tr_{it} = 0$.

2.19.6 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 = "gamma.gee", id, data)
```

then you may see 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: parameter estimates for the explanatory variables.
  - residuals: the working residuals in the final iteration of the fit.
  - fitted.values: the vector of fitted values for the systemic component.
  - linear.predictors: the vector of $x_{it}\beta$
  - max.id: the size of the largest cluster.
- From summary(z.out), you may extract:
  - coefficients: the parameter estimates with their associated standard errors, p-values, and z-statistics.
  - working.correlation: the "working" correlation matrix
- From the sim() output object s.out, you may extract quantities of interest arranged as matrices indexed by simulation $\times$ x-observation (for more than one x-observation). Available quantities are:
  - qi$ev: the simulated expected values for the specified values of x.

- qi$fd: the simulated first difference in the expected probabilities for the values specified in x and x1.
- qi$att.ev: the simulated average expected treatment effect for the treated from conditional prediction models.

2.19.7 See also

The gee function is part of the geepack package by Søren Højsgaard, Ulrich Halekoh and Jun Yan. Advanced users may wish to refer to `help(geepack)` and `help(family)`.

---

zelig-exp

Exponential Regression for Duration Dependent Variables

Use the exponential duration regression model if you have a dependent variable representing a duration (time until an event). The model assumes a constant hazard rate for all events. The dependent variable may be censored (for observations have not yet been completed when data were collected).

Syntax

With reference classes:

```
z5 <- zexp$new()
z5$zelig(Surv(Y, C) ~ X, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

```
z.out <- zelig(Surv(Y, C) ~ X, model = "exp", data = mydata)
x.out <- setx(z.out)
s.out <- sim(z.out, x = x.out)
```

Exponential models require that the dependent variable be in the form Surv(Y, C), where Y and C are vectors of length n. For each observation i in 1, ..., n, the value y_i is the duration (lifetime, for example), and the associated c_i is a binary variable such that $c_i = 1$ if the duration is not censored (*e.g.*, the subject dies during the study) or $c_i = 0$ if the duration is censored (*e.g.*, the subject is still alive at the end of the study and is know to live at least as long as y_i). If c_i is omitted, all Y are assumed to be completed; that is, time defaults to 1 for all observations.

Input Values

In addition to the standard inputs, zelig() takes the following additional options for exponential regression:

- robust: defaults to FALSE. If TRUE, zelig() computes robust standard errors based on sandwich estimators (see and) and the options selected in cluster.
- cluster: if robust = TRUE, you may select a variable to define groups of correlated observations. Let x3 be a variable that consists of either discrete numeric values, character strings, or factors that define strata. Then

```
z.out <- zelig(y ~ x1 + x2, robust = TRUE, cluster = "x3",
                        model = "exp", data = mydata)
```

means that the observations can be correlated within the strata defined by the variable x3, and that robust standard errors should be calculated according to those clusters. If robust = TRUE but cluster is not specified, zelig() assumes that each observation falls into its own cluster.

Example

Attach the sample data:

```
data(coalition)
```

Estimate the model:

```
z.out <- zelig(Surv(duration, ciep12) ~ fract + numst2, model = "exp", data = coalition)
```

```
## How to cite this model in Zelig:
##   Olivia Lau, Kosuke Imai, Gary King. 2011.
##   exp: Exponential Regression for Duration Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

View the regression output:

```
summary(z.out)
```

```
## Model:
## $by
## [1] 1
##
## Call:
## survival::survreg(formula = Surv(duration, ciep12) ~ fract +
##     numst2, data = ., dist = "exponential", model = FALSE)
##
## Coefficients:
##  (Intercept)        fract       numst2
##  5.535872596 -0.003908965  0.461179302
##
## Scale fixed at 1
##
## Loglik(model)= -1077.4   Loglik(intercept only)= -1100.7
##  Chisq= 46.66 on 2 degrees of freedom, p= 7.4e-11
## n= 314
## Next step: Use 'setx' method
```

Set the baseline values (with the ruling coalition in the minority) and the alternative values (with the ruling coalition in the majority) for X:

```
x.low <- setx(z.out, numst2 = 0)
```

```
## Error in terms(lm(formula, data), "predvars"): error in evaluating the argument 'x' in selecting a
## Calls: lm -> lm.fit -> Ops.Surv
```

```
x.high <- setx(z.out, numst2 = 1)
```

```
## Error in terms(lm(formula, data), "predvars"): error in evaluating the argument 'x' in selecting a
## Calls: lm -> lm.fit -> Ops.Surv
```

Simulate expected values and first differences:

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

Summarize quantities of interest and produce some plots:

```
summary(s.out)
```

```
##
##  sim x :
##  -----
## ev
##       mean        sd      50%     2.5%    97.5%
## 1 4.599351 0.1804811 4.601767 4.246964 4.961869
## pv
##       mean        sd      50%     2.5%    97.5%
## 1 4.599351 0.1804811 4.601767 4.246964 4.961869
##
##  sim x1 :
##  -----
## ev
##       mean       sd      50%     2.5%    97.5%
## 1 5.427579 0.191717 5.434867 5.057529 5.816774
## pv
##       mean       sd      50%     2.5%    97.5%
## 1 5.427579 0.191717 5.434867 5.057529 5.816774
## fd
##        mean        sd       50%      2.5%    97.5%
## 1 0.8282286 0.2208927 0.8332688 0.4155774 1.268978
```

```
plot(s.out)
```

Model

Let Y_i^* be the survival time for observation i. This variable might be censored for some observations at a fixed time y_c such that the fully observed dependent variable, Y_i, is defined as

$$Y_i = \begin{cases} Y_i^* & \text{if } Y_i^* \leq y_c \\ y_c & \text{if } Y_i^* > y_c \end{cases}$$

- The *stochastic component* is described by the distribution of the partially observed variable Y^*. We assume Y_i^* follows the exponential distribution whose density function is given by

  $$f(y_i^* \mid \lambda_i) = \frac{1}{\lambda_i} \exp\left(-\frac{y_i^*}{\lambda_i}\right)$$

  for $y_i^* \geq 0$ and $\lambda_i > 0$. The mean of this distribution is λ_i.

  In addition, survival models like the exponential have three additional properties. The hazard function $h(t)$ measures the probability of not surviving past time t given survival up to t. In general, the hazard function is equal to $f(t)/S(t)$ where the survival function $S(t) = 1 - \int_0^t f(s)ds$ represents the fraction still surviving at time t. The cumulative hazard function $H(t)$ describes the probability of dying before time t. In general, $H(t) = \int_0^t h(s)ds = -\log S(t)$. In the case of the exponential model,

  $$\begin{aligned} h(t) &= \frac{1}{\lambda_i} \\ S(t) &= \exp\left(-\frac{t}{\lambda_i}\right) \\ H(t) &= \frac{t}{\lambda_i} \end{aligned}$$

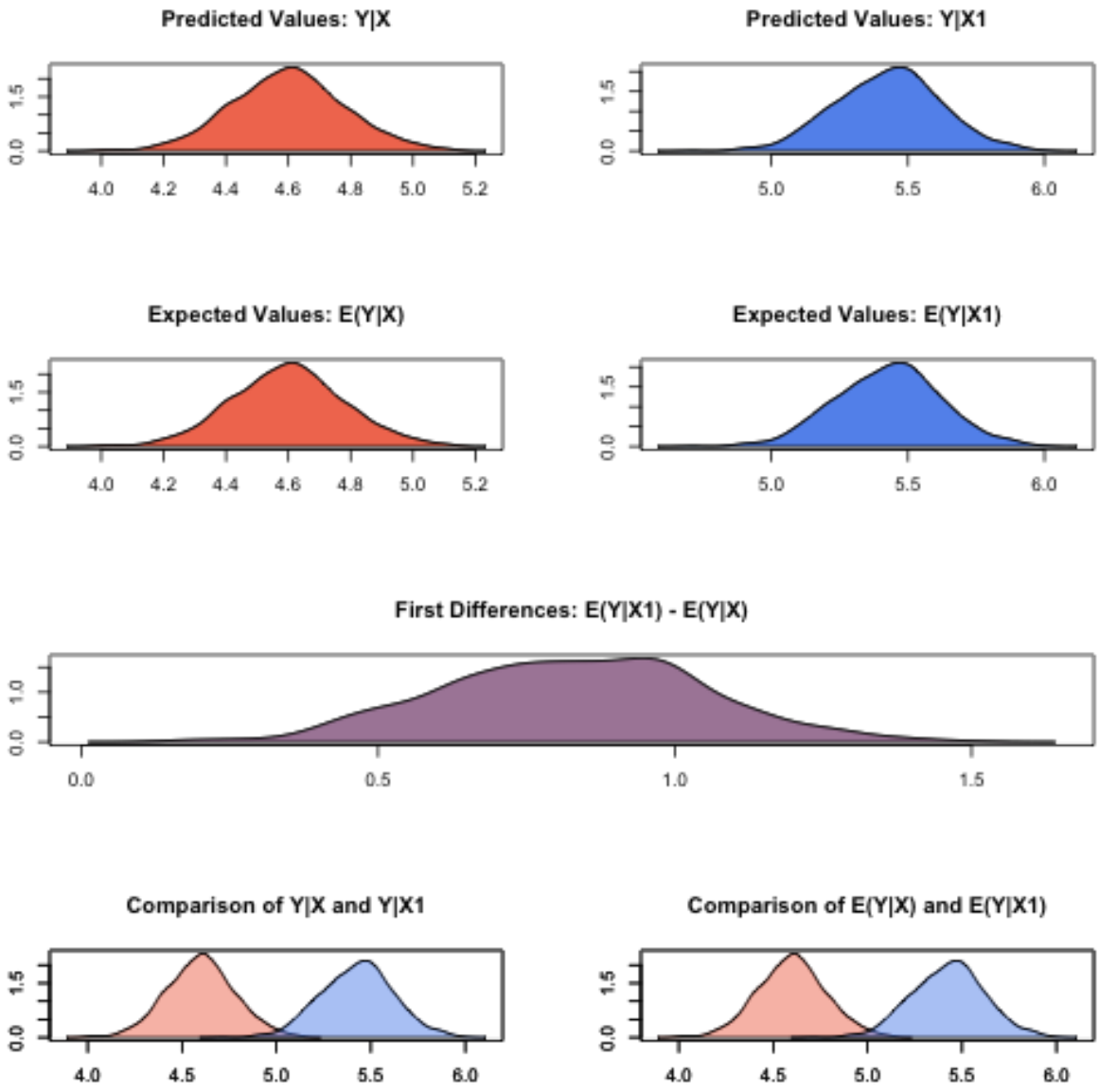


Figure 2.13: Zelig-exp

For the exponential model, the hazard function $h(t)$ is constant over time. The Weibull model and lognormal models allow the hazard function to vary as a function of elapsed time (see and respectively).

- The *systematic component* λ_i is modeled as

$$\lambda_i = \exp(x_i\beta),$$

where x_i is the vector of explanatory variables, and β is the vector of coefficients.

Quantities of Interest

- The expected values (qi$ev) for the exponential model are simulations of the expected duration given x_i and draws of β from its posterior,

$$E(Y) = \lambda_i = \exp(x_i\beta).$$

- The predicted values (qi$pr) are draws from the exponential distribution with rate equal to the expected value.
- The first difference (or difference in expected values, qi$ev.diff), is

$$\text{FD} \;=\; E(Y \mid x_1) - E(Y \mid x),$$

where x and x_1 are different vectors of values for the explanatory variables.

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

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

where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. When $Y_i(t_i = 1)$ is censored rather than observed, we replace it with a simulation from the model given available knowledge of the censoring process. Variation in the simulations is due to two factors: uncertainty in the imputation process for censored y_i^* and uncertainty in simulating $E[Y_i(t_i = 0)]$, the counterfactual expected value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

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

$$\frac{1}{\sum_{i=1}^{n} t_i} \sum_{i:t_i=1}^{n} \left\{ Y_i(t_i = 1) - Y_i(\widehat{t_i = 0}) \right\},$$

where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. When $Y_i(t_i = 1)$ is censored rather than observed, we replace it with a simulation from the model given available knowledge of the censoring process. Variation in the simulations is due to two factors: uncertainty in the imputation process for censored y_i^* and uncertainty in simulating $Y_i(\widehat{t_i = 0})$, the counterfactual predicted value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

Output Values

The output of each Zelig command contains useful information which you may view. For example, if you run `z.out <- zelig(Surv(Y, C) ~ X, model = exp, data)`, then you may examine the available information in `z.out` by using `names(z.out)`, see the coefficients by using z.out$coefficients, and a default summary of information through `summary(z.out)`.

See also

The exponential function is part of the survival library by Terry Therneau, ported to R by Thomas Lumley. Advanced users may wish to refer to `help(survfit)` in the survival library.

zelig-gamma

Gamma Regression for Continuous, Positive Dependent Variables

Use the gamma regression model if you have a positive-valued dependent variable such as the number of years a parliamentary cabinet endures, or the seconds you can stay airborne while jumping. The gamma distribution assumes that all waiting times are complete by the end of the study (censoring is not allowed).

Syntax

With reference classes:

```
z5 <- zgamma$new()
z5$zelig(Y ~ X1 + X ~ X, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

```
z.out <- zelig(Y ~ X1 + X2, model = "gamma", data = mydata)
x.out <- setx(z.out)
s.out <- sim(z.out, x = x.out, x1 = NULL)
```

Example

Attach the sample data:

```
data(coalition)
```

Estimate the model:

```
z.out <- zelig(duration ~ fract + numst2, model = "gamma", data = coalition)
```

```
## How to cite this model in Zelig:
##   Kosuke Imai, Gary King, Olivia Lau. 2007.
##   gamma: Gamma Regression for Continuous, Positive Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

View the regression output:

```
summary(z.out)
```

```
## Model:
## $by
## [1] 1
##
##
## Call:  stats::glm(formula = duration ~ fract + numst2, family = Gamma("inverse"),
##     data = .)
```

```
##
## Coefficients:
## (Intercept)        fract       numst2
##  -0.0129597    0.0001149   -0.0173875
##
## Degrees of Freedom: 313 Total (i.e. Null);  311 Residual
## Null Deviance:       300.7
## Residual Deviance: 272.2         AIC: 2428
## Next step: Use 'setx' method
```

Set the baseline values (with the ruling coalition in the minority) and the alternative values (with the ruling coalition in the majority) for X:

```
x.low <- setx(z.out, numst2 = 0)
x.high <- setx(z.out, numst2 = 1)
```

Simulate expected values (qi$ev) and first differences (qi$fd):

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

summary(s.out)
```

```
##
##  sim x :
##  -----
## ev
##          mean       sd      50%     2.5%    97.5%
## [1,] 14.43674 1.093498 14.33691 12.57224 17.01294
## pv
##         mean       sd      50%      2.5%    97.5%
## [1,] 14.6374 12.91882 11.19025 0.6425469 48.15589
##
##  sim x1 :
##  -----
## ev
##          mean      sd     50%     2.5%    97.5%
## [1,] 19.20518 1.14026 19.1172 17.25049 21.60574
## pv
##          mean       sd     50%     2.5%    97.5%
## [1,] 18.62654 16.58418 13.7207 1.164515 64.19589
## fd
##         mean       sd     50%     2.5%    97.5%
## [1,] 4.76844 1.583298 4.72891 1.474829 7.913904
```

```
plot(s.out)
```

Model

- The Gamma distribution with scale parameter α has a *stochastic component*:

$$
\begin{aligned}
Y \sim & \quad \text{Gamma}(y_i \mid \lambda_i, \alpha) \\
f(y) = & \quad \frac{1}{\alpha^{\lambda_i}\,\Gamma\lambda_i}\, y_i^{\lambda_i - 1} \exp - \left\{ \frac{y_i}{\alpha} \right\}
\end{aligned}
$$

  for $\alpha, \lambda_i, y_i > 0$.

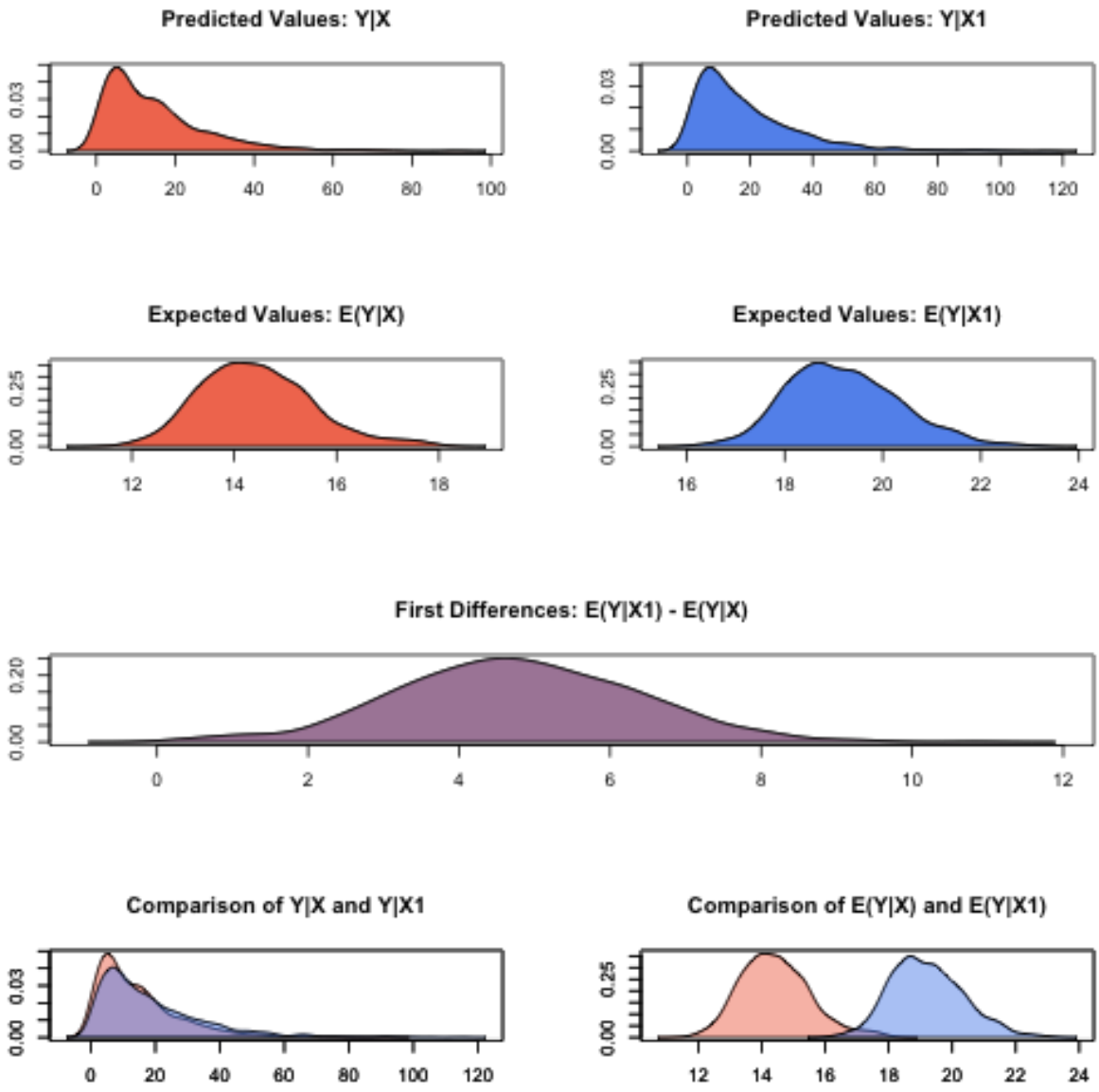


Figure 2.14: Zelig-gamma

- The *systematic component* is given by

$$\lambda_i = \frac{1}{x_i \beta}$$

Quantities of Interest

- The expected values (qi$ev) are simulations of the mean of the stochastic component given draws of α and β from their posteriors:

$$E(Y) = \alpha \lambda_i.$$

- The predicted values (qi$pr) are draws from the gamma distribution for each given set of parameters (α, λ_i).
- If x1 is specified, sim() also returns the differences in the expected values (qi$fd),

$$E(Y \mid x_1) - E(Y \mid x)$$

.

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

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

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $E[Y_i(t_i = 0)]$, the counterfactual expected value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

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

$$\frac{1}{\sum_{i=1}^{n} t_i} \sum_{i:t_i=1}^{n} \left\{ Y_i(t_i = 1) - Y_i(\widehat{t_i = 0}) \right\},$$

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $Y_i(\widehat{t_i = 0})$, the counterfactual predicted value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

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 = gamma, data)`, then you may examine the available information in `z.out` by using `names(z.out)`, see the coefficients by using z.out$coefficients, and a default summary of information through `summary(z.out)`.

See also

The gamma model is part of the stats package. Advanced users may wish to refer to `help(glm)` and `help(family)`.

zelig-logit

Logistic Regression for Dichotomous Dependent Variables

Logistic regression specifies a dichotomous dependent variable as a function of a set of explanatory variables.

Syntax

With reference classes:

```
z5 <- zlogit$new()
z5$zelig(Y ~ X1 + X ~ X, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

```
z.out <- zelig(Y ~ X1 + X2, model = "logit", data = mydata)
x.out <- setx(z.out)
s.out <- sim(z.out, x = x.out, x1 = NULL)
```

Examples

Basic Example Attaching the sample turnout dataset:

```
data(turnout)
```

Estimating parameter values for the logistic regression:

```
z.out1 <- zelig(vote ~ age + race, model = "logit", data = turnout)
```

```
## How to cite this model in Zelig:
##   Kosuke Imai, Gary King, Olivia Lau. 2007.
##   logit: Logistic Regression for Dichotomous Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

Setting values for the explanatory variables:

```
x.out1 <- setx(z.out1, age = 36, race = "white")
```

Simulating quantities of interest from the posterior distribution.

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

```
summary(s.out1)
```

```
##
##  sim x :
##  -----
## ev
##           mean         sd       50%      2.5%     97.5%
## [1,] 0.7480614 0.01138812 0.7484417 0.7258056 0.7699426
## pv
##          0     1
## [1,] 0.275 0.725
```

```
plot(s.out1)
```

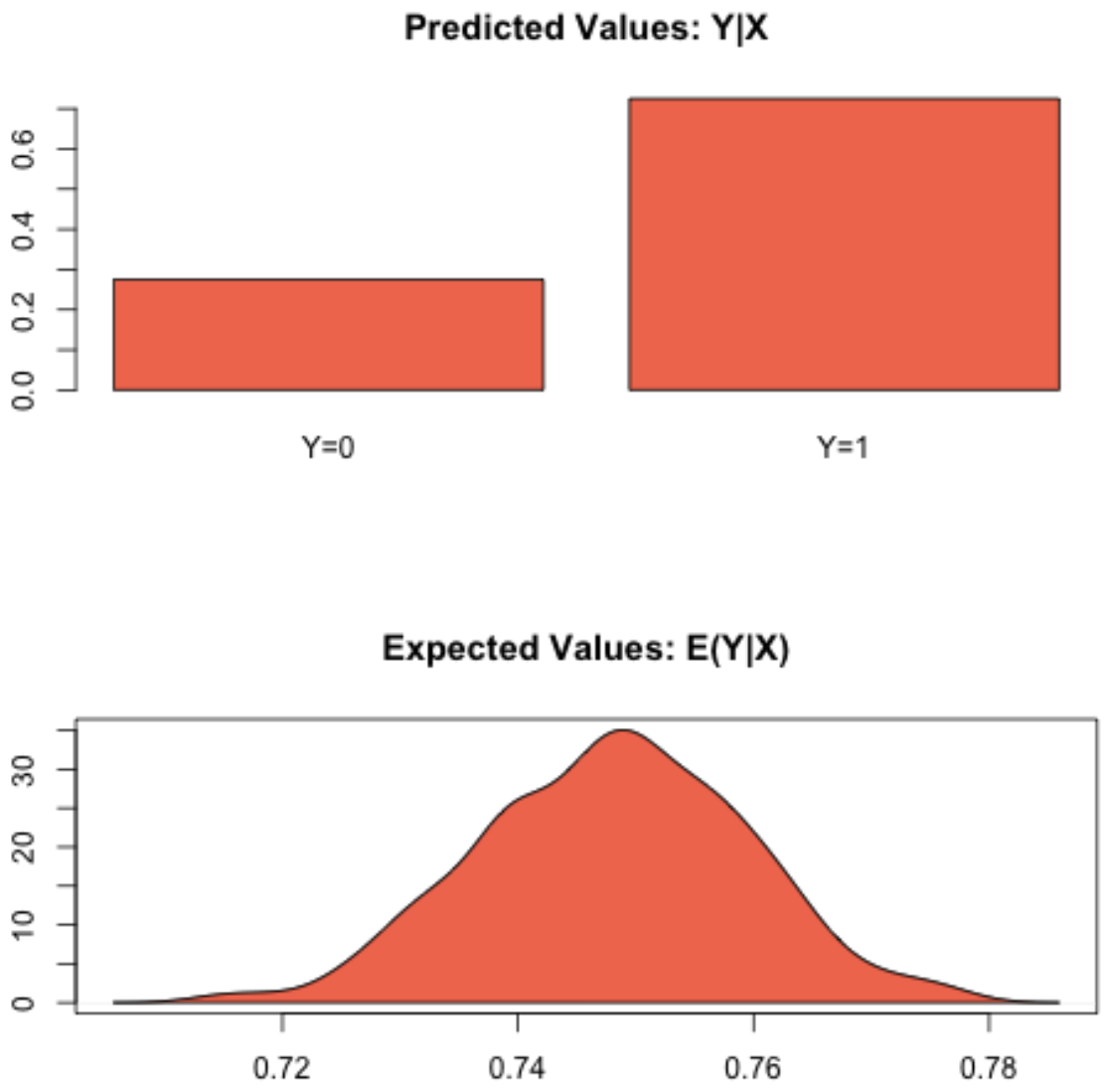


Figure 2.15: Zelig-logit-1

Simulating First Differences Estimating the risk difference (and risk ratio) between low education (25th percentile) and high education (75th percentile) while all the other variables held at their default values.

```
z.out2 <- zelig(vote ~ race + educate, model = "logit", data = turnout)

## How to cite this model in Zelig:
##   Kosuke Imai, Gary King, Olivia Lau. 2007.
##   logit: Logistic Regression for Dichotomous Dependent Variables
```

```
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/

x.high <- setx(z.out2, educate = quantile(turnout$educate, prob = 0.75))
x.low <- setx(z.out2, educate = quantile(turnout$educate, prob = 0.25))
s.out2 <- sim(z.out2, x = x.high, x1 = x.low)
summary(s.out2)

##
##  sim x :
##  -----
## ev
##           mean         sd       50%      2.5%    97.5%
## [1,] 0.8230526 0.01064902 0.8234506 0.8006938 0.843638
## pv
##          0     1
## [1,] 0.177 0.823
##
##  sim x1 :
##  -----
## ev
##           mean         sd       50%      2.5%     97.5%
## [1,] 0.7095084 0.01275007 0.7103366 0.6835694 0.7324867
## pv
##         0    1
## [1,] 0.31 0.69
## fd
##            mean         sd        50%       2.5%       97.5%
## [1,] -0.1135442 0.01117081 -0.1134733 -0.1356103 -0.09268328

plot(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

$$\begin{aligned} Y_i \sim & \text{ Bernoulli}(y_i \mid \pi_i) \\ = & \ \pi_i^{y_i}(1-\pi_i)^{1-y_i} \end{aligned}$$

  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.

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 draws of β from its sampling distribution.

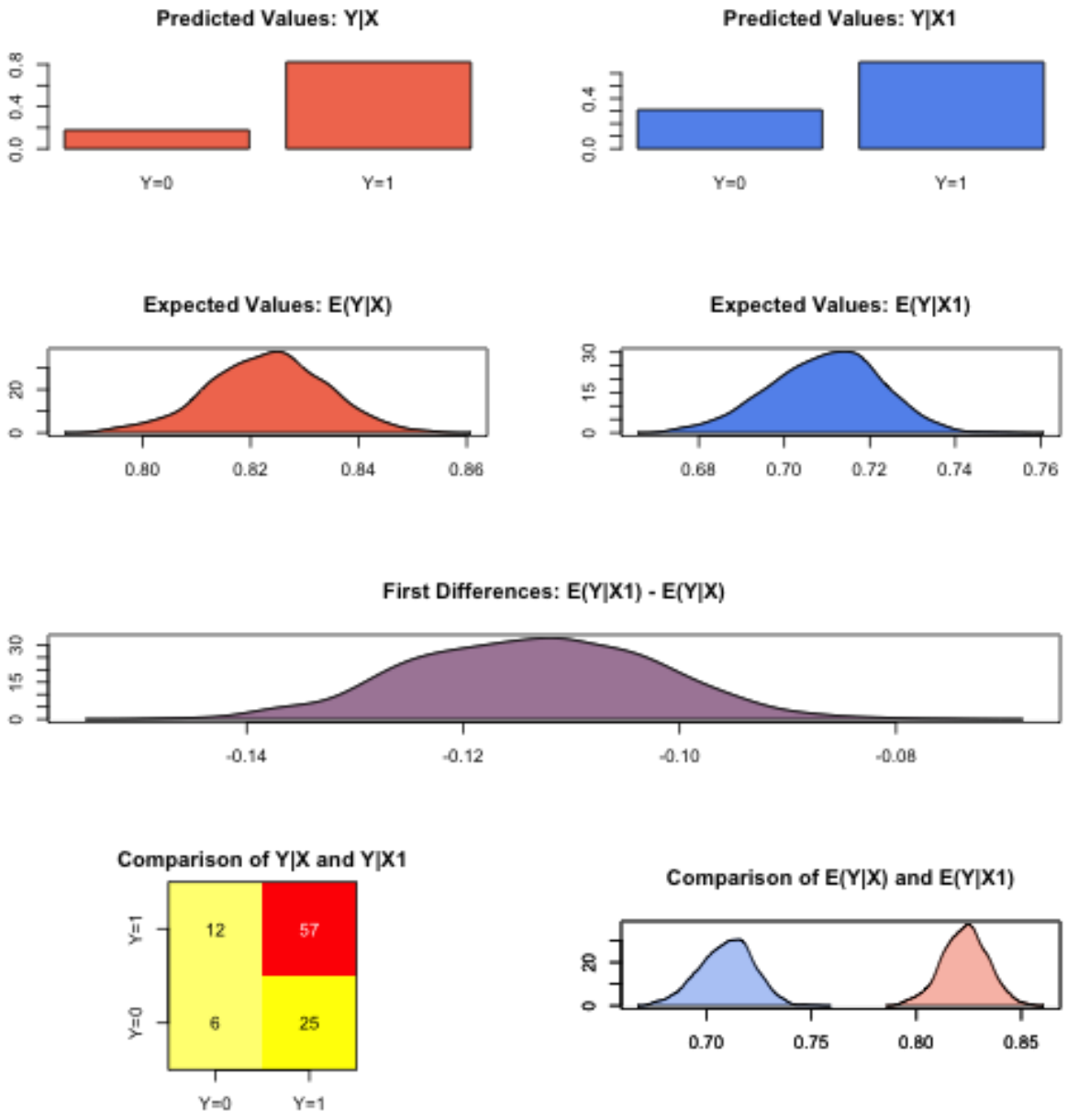


Figure 2.16: Zelig-logit-2

- The predicted values (qi$pr) are draws from the Binomial distribution with mean equal to the simulated expected value π_i.
- The first difference (qi$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 (att.ev) for the treatment group is

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

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $E[Y_i(t_i = 0)]$, the counterfactual expected value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.
- In conditional prediction models, the average predicted treatment effect (att.pr) for the treatment group is

$$\frac{1}{\sum_{i=1}^{n} t_i} \sum_{i:t_i=1}^{n} \left\{ Y_i(t_i = 1) - Y_i(\widehat{t_i = 0}) \right\},$$

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $Y_i(\widehat{t_i = 0})$, the counterfactual predicted value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

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, data)`, then you may examine the available information in `z.out` by using `names(z.out)`, see the coefficients by using z.out$coefficients, and a default summary of information through `summary(z.out)`.

See also

The logit model is part of the stats package. Advanced users may wish to refer to `help(glm)` and `help(family)`.

zelig-lognorm

Log-Normal Regression for Duration Dependent Variables

The log-normal model describes an event's duration, the dependent variable, as a function of a set of explanatory variables. The log-normal model may take time censored dependent variables, and allows the hazard rate to increase and decrease.

Syntax

With reference classes:

```
z5 <- zlognorm$new()
z5$zelig(Surv(Y, C) ~ X, data = mydata)
z5$setx()
z5$sim()
```

With reference classes:

```
z5 <- zlognorm$new()
z5$zelig(Surv(Y, C) ~ X, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

```
z.out <- zelig(Surv(Y, C) ~ X, model = "lognorm", data = mydata)
x.out <- setx(z.out)
s.out <- sim(z.out, x = x.out)
```

Log-normal models require that the dependent variable be in the form Surv(Y, C), where Y and C are vectors of length n. For each observation i in 1, ..., n, the value y_i is the duration (lifetime, for example) of each subject, and the associated c_i is a binary variable such that $c_i = 1$ if the duration is not censored (*e.g.*, the subject dies during the study) or $c_i = 0$ if the duration is censored (*e.g.*, the subject is still alive at the end of the study). If c_i is omitted, all Y are assumed to be completed; that is, time defaults to 1 for all observations.

Input Values

In addition to the standard inputs, zelig() takes the following additional options for lognormal regression:

- robust: defaults to FALSE. If TRUE, zelig() computes robust standard errors based on sandwich estimators (see and) based on the options in cluster.
- cluster: if robust = TRUE, you may select a variable to define groups of correlated observations. Let x3 be a variable that consists of either discrete numeric values, character strings, or factors that define strata. Then

```
z.out <- zelig(y ~ x1 + x2, robust = TRUE, cluster = "x3", model = "exp", data = mydata)
```

means that the observations can be correlated within the strata defined by the variable x3, and that robust standard errors should be calculated according to those clusters. If robust = TRUE but cluster is not specified, zelig() assumes that each observation falls into its own cluster.

Example

Attach the sample data:

```
data(coalition)
```

Estimate the model:

```
z.out <- zelig(Surv(duration, ciep12) ~ fract + numst2, model ="lognorm",  data = coalition)

## How to cite this model in Zelig:
##   Matthew Owen, Olivia Lau, Kosuke Imai, Gary King. 2007.
##   lognorm: Log-Normal Regression for Duration Dependent Variables
```

```
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

View the regression output:

```
summary(z.out)
```

```
## Model:
## $by
## [1] 1
##
## Call:
## survival::survreg(formula = Surv(duration, ciep12) ~ fract +
##     numst2, data = ., dist = "lognormal", model = FALSE)
##
## Coefficients:
## (Intercept)       fract      numst2
##  5.36666977 -0.00443755  0.55983251
##
## Scale= 1.20008
##
## Loglik(model)= -1077.9   Loglik(intercept only)= -1101.2
##  Chisq= 46.58 on 2 degrees of freedom, p= 7.7e-11
## n= 314
## Next step: Use 'setx' method
```

Set the baseline values (with the ruling coalition in the minority) and the alternative values (with the ruling coalition in the majority) for X:

```
x.low <- setx(z.out, numst2 = 0)
```

```
## Error in terms(lm(formula, data), "predvars"): error in evaluating the argument 'x' in selecting a
## Calls: lm -> lm.fit -> Ops.Surv
```

```
x.high <- setx(z.out, numst2= 1)
```

```
## Error in terms(lm(formula, data), "predvars"): error in evaluating the argument 'x' in selecting a
## Calls: lm -> lm.fit -> Ops.Surv
```

Simulate expected values (qi$ev) and first differences (qi$fd):

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

```
summary(s.out)
```

```
##
##  sim x :
##  -----
## ev
##           mean         sd       50%      2.5%     97.5%
## [1,] 0.7100528 0.01309083 0.7102303 0.6833987 0.7348151
## pv
##          0     1
## [1,] 0.289 0.711
##
##  sim x1 :
##  -----
## ev
##           mean         sd       50%      2.5%    97.5%
```

```
## [1,] 0.8231874 0.01030848 0.8233464 0.8015839 0.841741
## pv
##           0     1
## [1,] 0.161 0.839
## fd
##           mean         sd       50%       2.5%    97.5%
## [1,] 0.1131346 0.01174466 0.1133255 0.08971273 0.135815
```

```
plot(s.out)
```

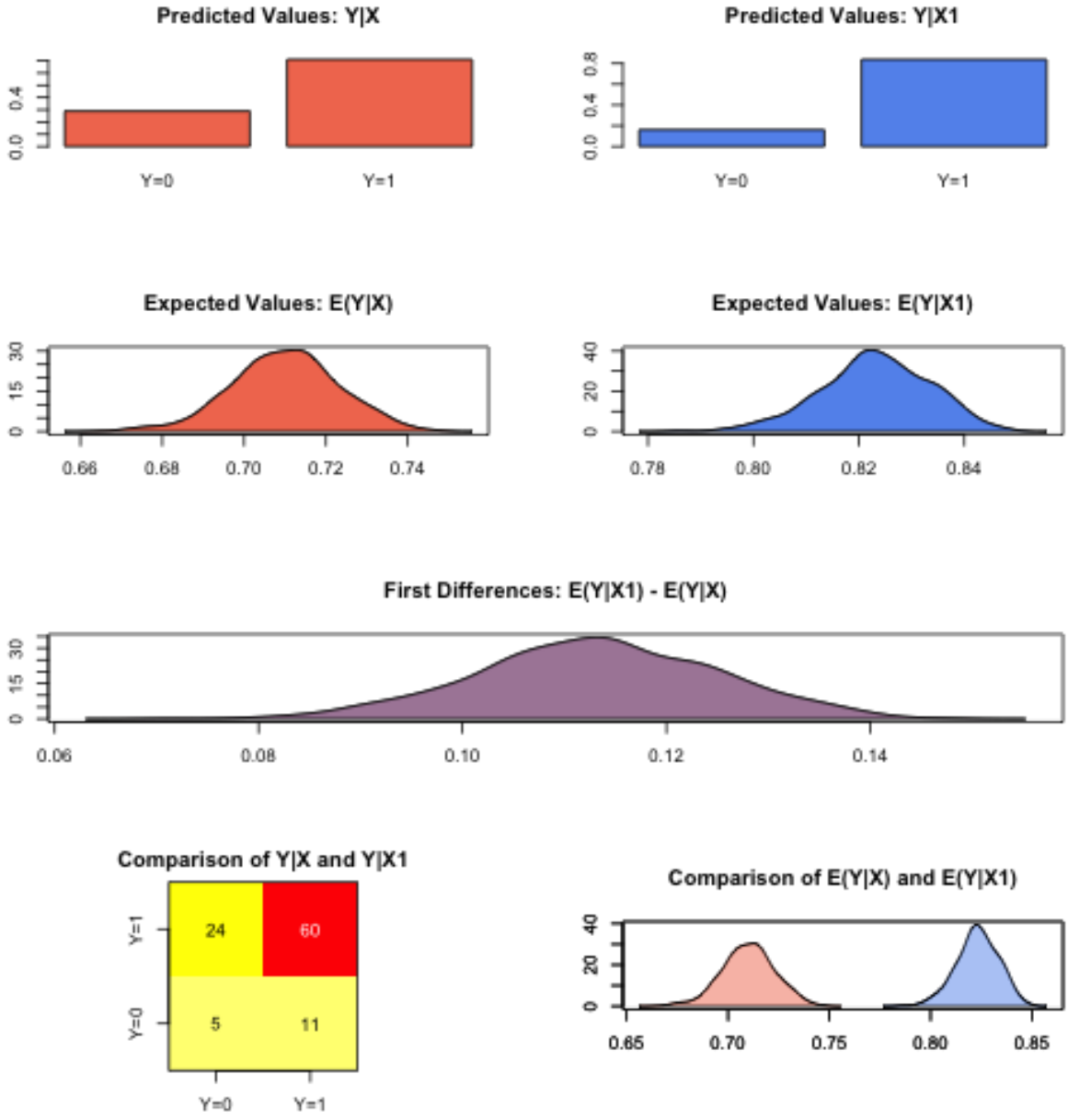


Figure 2.17: Zelig-lognorm

Model

Let Y_i^* be the survival time for observation i with the density function $f(y)$ and the corresponding distribution function $F(t) = \int_0^t f(y)dy$. This variable might be censored for some observations at a fixed time y_c such that the fully observed dependent variable, Y_i, is defined as

$$Y_i = \begin{cases} Y_i^* & \text{if } Y_i^* \leq y_c \\ y_c & \text{if } Y_i^* > y_c \end{cases}$$

- The *stochastic component* is described by the distribution of the partially observed variable, Y^*. For the lognormal model, there are two equivalent representations:

  $$Y_i^* \sim \text{LogNormal}(\mu_i, \sigma^2) \text{ or } \quad \log(Y_i^*) \sim \text{Normal}(\mu_i, \sigma^2)$$

  where the parameters μ_i and σ^2 are the mean and variance of the Normal distribution. (Note that the output from zelig() parameterizes scale:math:‘ = sigma‘.)

  In addition, survival models like the lognormal have three additional properties. The hazard function $h(t)$ measures the probability of not surviving past time t given survival up to t. In general, the hazard function is equal to $f(t)/S(t)$ where the survival function $S(t) = 1 - \int_0^t f(s)ds$ represents the fraction still surviving at time t. The cumulative hazard function $H(t)$ describes the probability of dying before time t. In general, $H(t) = \int_0^t h(s)ds = -\log S(t)$. In the case of the lognormal model,

  $$\begin{aligned} h(t) &= \frac{1}{\sqrt{2\pi}\,\sigma t\, S(t)} \exp\left\{-\frac{1}{2\sigma^2}(\log \lambda t)^2\right\} \\ S(t) &= 1 - \Phi\left(\frac{1}{\sigma}\log \lambda t\right) \\ H(t) &= -\log\left\{1 - \Phi\left(\frac{1}{\sigma}\log \lambda t\right)\right\} \end{aligned}$$

  where $\Phi(\cdot)$ is the cumulative density function for the Normal distribution.

- The *systematic component* is described as:

  $$\mu_i = x_i\beta.$$

Quantities of Interest

- The expected values (qi$ev) for the lognormal model are simulations of the expected duration:

  $$E(Y) = \exp\left(\mu_i + \frac{1}{2}\sigma^2\right),$$

  given draws of β and σ from their sampling distributions.

- The predicted value is a draw from the log-normal distribution given simulations of the parameters (λ_i, σ).

- The first difference (qi$fd) is

  $$\text{FD} = E(Y \mid x_1) - E(Y \mid x).$$

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

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

where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. When $Y_i(t_i = 1)$ is censored rather than observed, we replace it with a simulation from the model given available knowledge of the censoring process. Variation in the simulations is due to two factors: uncertainty in the imputation process for censored y_i^* and uncertainty in simulating $E[Y_i(t_i = 0)]$, the counterfactual expected value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

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

$$\frac{1}{\sum_{i=1}^{n} t_i} \sum_{i:t_i=1}^{n} \{Y_i(t_i = 1) - Y_i(\widehat{t_i = 0})\},$$

where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. When $Y_i(t_i = 1)$ is censored rather than observed, we replace it with a simulation from the model given available knowledge of the censoring process. Variation in the simulations are due to two factors: uncertainty in the imputation process for censored y_i^* and uncertainty in simulating $Y_i(\widehat{t_i = 0})$, the counterfactual predicted value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

Output Values

The output of each Zelig command contains useful information which you may view. For example, if you run `z.out <- zelig(Surv(Y, C) ~ X, model = lognorm, data)`, then you may examine the available information in `z.out` by using `names(z.out)`, see the coefficients by using z.out$coefficients, and a default summary of information through `summary(z.out)`.

See also

The exponential function is part of the survival library by by Terry Therneau, ported to R by Thomas Lumley. Advanced users may wish to refer to `help(survfit)` in the survival library.

zelig-ls

Least Squares Regression for Continuous Dependent Variables

Use least squares regression analysis to estimate the best linear predictor for the specified dependent variables.

Syntax

With reference classes:

```
z5 <- zls$new()
z5$zelig(Y ~ X1 + X ~ X, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

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

Examples

Basic Example with First Differences Attach sample data:

```
data(macro)
```

Estimate model:

```
z.out1 <- zelig(unem ~ gdp + capmob + trade, model = "ls", data = macro)
```

```
## How to cite this model in Zelig:
##   Kosuke Imai, Gary King, and Olivia Lau. 2007.
##   ls: Least Squares Regression for Continuous Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

Summarize regression coefficients:

```
summary(z.out1)
```

```
## Model:
## $by
## [1] 1
##
##
## Call:
## stats::lm(formula = unem ~ gdp + capmob + trade, data = .)
##
## Coefficients:
## (Intercept)          gdp       capmob        trade
##     6.18129     -0.32360      1.42194      0.01985
##
## Next step: Use 'setx' method
```

Set explanatory variables to their default (mean/mode) values, with high (80th percentile) and low (20th percentile) values for the trade variable:

```
x.high <- setx(z.out1, trade = quantile(macro$trade, 0.8))
x.low <- setx(z.out1, trade = quantile(macro$trade, 0.2))
```

Generate first differences for the effect of high versus low trade on GDP:

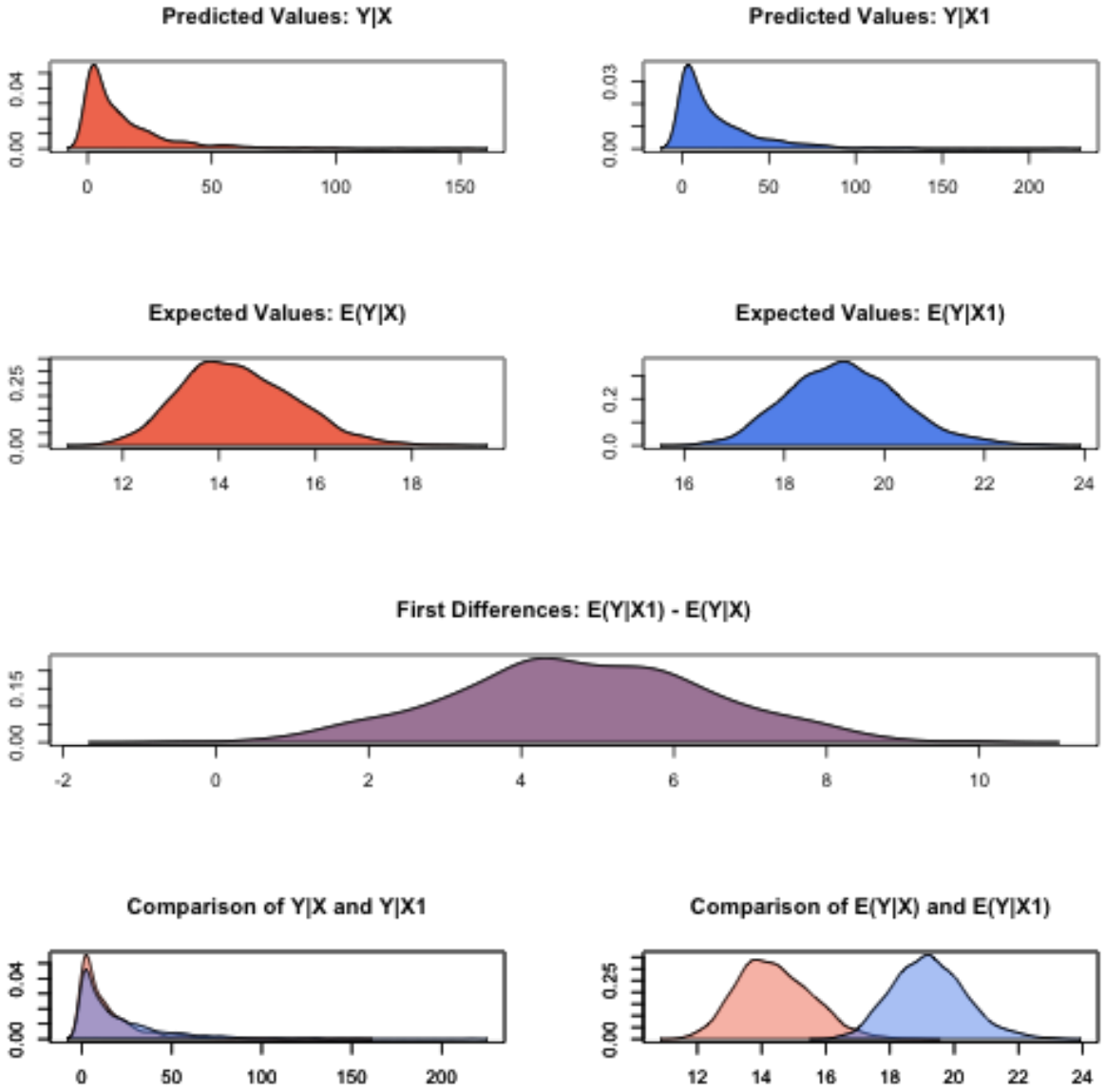
```
s.out1 <- sim(z.out1, x = x.high, x1 = x.low)
```

```
summary(s.out1)
```

```
##
##  sim x :
##  -----
## ev
##       mean        sd      50%     2.5%    97.5%
## 1 5.425577 0.1913408 5.426885 5.071941 5.790102
## pv
##       mean        sd      50%     2.5%    97.5%
## 1 5.425577 0.1913408 5.426885 5.071941 5.790102
##
##  sim x1 :
##  -----
## ev
```

```
##       mean        sd      50%    2.5%    97.5%
## 1 4.607931 0.1868002 4.607313 4.23521 4.976644
## pv
##       mean        sd      50%    2.5%    97.5%
## 1 4.607931 0.1868002 4.607313 4.23521 4.976644
## fd
##         mean        sd        50%      2.5%      97.5%
## 1 -0.8176458 0.2351794 -0.8129858 -1.264778 -0.3534317
```

```
plot(s.out1)
```



Using Dummy Variables Estimate a model with fixed effects for each country (see for help with dummy variables). Note that you do not need to create dummy variables, as the program will automatically parse the unique values in the

selected variable into discrete levels.

```
z.out2 <- zelig(unem ~ gdp + trade + capmob + as.factor(country), model = "ls", data = macro)

## How to cite this model in Zelig:
##   Kosuke Imai, Gary King, and Olivia Lau. 2007.
##   ls: Least Squares Regression for Continuous Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

Set values for the explanatory variables, using the default mean/mode values, with country set to the United States and Japan, respectively:

```
x.US <- setx(z.out2, country = "United States")
x.Japan <- setx(z.out2, country = "Japan")
```

Simulate quantities of interest:

```
s.out2 <- sim(z.out2, x = x.US, x1 = x.Japan)

plot(s.out2)
```

Model

- The *stochastic component* is described by a density with mean μ_i and the common variance σ^2

$$Y_i \sim f(y_i \mid \mu_i, \sigma^2).$$

- The *systematic component* models the conditional mean as

$$\mu_i = x_i \beta$$

where x_i is the vector of covariates, and β is the vector of coefficients.

The least squares estimator is the best linear predictor of a dependent variable given x_i, and minimizes the sum of squared residuals, $\sum_{i=1}^{n}(Y_i - x_i\beta)^2$.

Quantities of Interest

- The expected value (qi$ev) is the mean of simulations from the stochastic component,

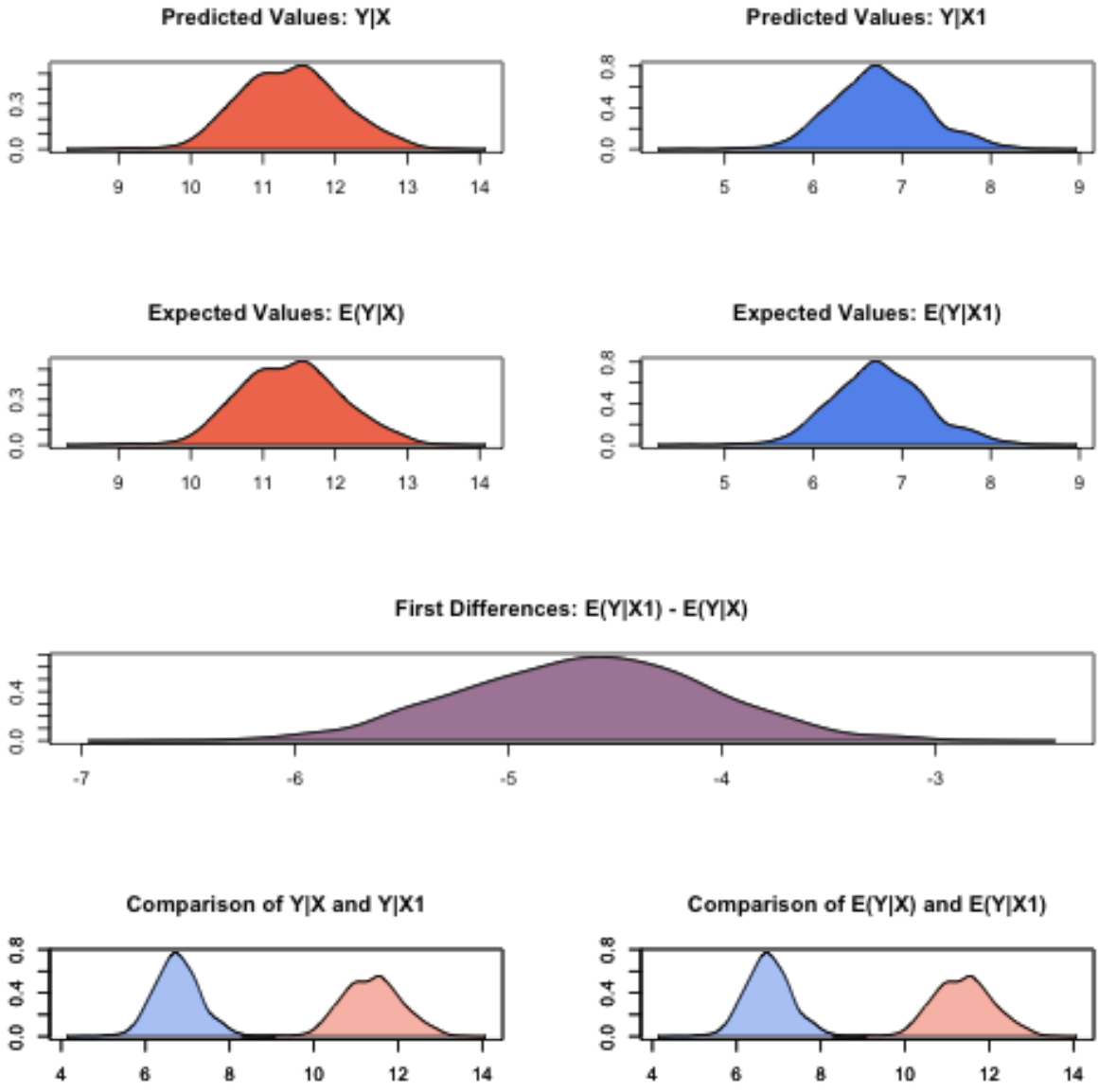
$$E(Y) = x_i \beta,$$

given a draw of β from its sampling distribution.

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

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

where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $E[Y_i(t_i = 0)]$, the counterfactual expected value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.


Predicted Values: Y|X
0.0 0.3
9 10 11 12 13 14
Predicted Values: Y|X1
0.0 0.4 0.8
5 6 7 8 9
Expected Values: E(Y|X)
0.0 0.3
9 10 11 12 13 14
Expected Values: E(Y|X1)
0.0 0.4 0.8
5 6 7 8 9
First Differences: E(Y|X1) - E(Y|X)
0.0 0.4
-7 -6 -5 -4 -3
Comparison of Y|X and Y|X1
0.0 0.4 0.8
4 6 8 10 12 14
Comparison of E(Y|X) and E(Y|X1)
0.0 0.4 0.8
4 6 8 10 12 14

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 = ls, data)`, then you may examine the available information in `z.out` by using `names(z.out)`, see 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: parameter estimates for the explanatory variables.
  - residuals: the working residuals in the final iteration of the IWLS fit.
  - fitted.values: fitted values.
  - df.residual: the residual degrees of freedom.
  - zelig.data: the input data frame if save.data = TRUE.
- From summary(z.out), you may extract:
  - coefficients: the parameter estimates with their associated standard errors, p-values, and t-statistics.

$$\hat{\beta} = \left(\sum_{i=1}^{n} x_i' x_i\right)^{-1} \sum x_i y_i$$

  - sigma: the square root of the estimate variance of the random error e:

$$\hat{\sigma} = \frac{\sum (Y_i - x_i\hat{\beta})^2}{n-k}$$

  - r.squared: the fraction of the variance explained by the model.

$$R^2 = 1 - \frac{\sum (Y_i - x_i\hat{\beta})^2}{\sum (y_i - \bar{y})^2}$$

  - adj.r.squared: the above R^2 statistic, penalizing for an increased number of explanatory variables.
  - cov.unscaled: a $k \times k$ matrix of unscaled covariances.

See also

The least squares regression is part of the stats package by William N. Venables and Brian D. Ripley .In addition, advanced users may wish to refer to `help(lm)` and `help(lm.fit)`.

zelig-negbin

Negative Binomial Regression for Event Count Dependent Variables

Use the negative binomial regression if you have a count of events for each observation of your dependent variable. The negative binomial model is frequently used to estimate over-dispersed event count models.

Syntax

With reference classes:

```
z5 <- znegbin$new()
z5$zelig(Y ~ X1 + X ~ X, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

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

Example

Load sample data:

```
data(sanction)
```

Estimate the model:

```
z.out <- zelig(num ~ target + coop, model = "negbin", data = sanction)

## How to cite this model in Zelig:
##   Kosuke Imai, Gary King, Olivia Lau. 2008.
##   negbinom: Negative Binomial Regression for Event Count Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/

summary(z.out)

## Model:
## $by
## [1] 1
##
##
## Call:  MASS::glm.nb(formula = num ~ target + coop, data = ., init.theta = 1.841603403,
##     link = log)
##
## Coefficients:
## (Intercept)       target         coop
##      -1.564        0.151        1.286
##
## Degrees of Freedom: 77 Total (i.e. Null);  75 Residual
## Null Deviance:       237.1
## Residual Deviance: 56.55         AIC: 360.2
## Next step: Use 'setx' method
```

Set values for the explanatory variables to their default mean values:

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

Simulate fitted values:

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

summary(s.out)
```

```
##
##  sim x :
##  -----
## ev
##          mean       sd      50%     2.5%    97.5%
## [1,] 2.979505 0.354668 2.957069 2.386268 3.744801
## pv
## qi
##     0     1     2     3     4     5     6     7     8     9    10    11
## 0.212 0.207 0.201 0.123 0.098 0.051 0.041 0.023 0.017 0.008 0.004 0.004
##    12    13    14    19
## 0.004 0.004 0.002 0.001
```

```
plot(s.out)
```

Model

Let Y_i be the number of independent events that occur during a fixed time period. This variable can take any non-negative integer value.

- The negative binomial distribution is derived by letting the mean of the Poisson distribution vary according to a fixed parameter ζ given by the Gamma distribution. The *stochastic component* is given by

$$\begin{aligned} Y_i \mid \zeta_i \sim & \quad \text{Poisson}(\zeta_i \mu_i), \\ \zeta_i \sim & \quad \frac{1}{\theta}\text{Gamma}(\theta). \end{aligned}$$

  The marginal distribution of Y_i is then the negative binomial with mean μ_i and variance $\mu_i + \mu_i^2/\theta$:

$$\begin{aligned} Y_i \sim & \quad \text{NegBin}(\mu_i, \theta), \\ = & \quad \frac{\Gamma(\theta + y_i)}{y!\,\Gamma(\theta)} \frac{\mu_i^{y_i}\,\theta^{\theta}}{(\mu_i + \theta)^{\theta + y_i}}, \end{aligned}$$

  where θ is the systematic parameter of the Gamma distribution modeling ζ_i.

- The *systematic component* is given by

$$\mu_i = \exp(x_i \beta)$$

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

Quantities of Interest

- The expected values (qi$ev) are simulations of the mean of the stochastic component. Thus,

$$E(Y) = \mu_i = \exp(x_i \beta),$$

  given simulations of β.

- The predicted value (qi$pr) drawn from the distribution defined by the set of parameters (μ_i, θ).
- The first difference (qi$fd) is

$$\text{FD} \;=\; E(Y|x_1) - E(Y \mid x)$$

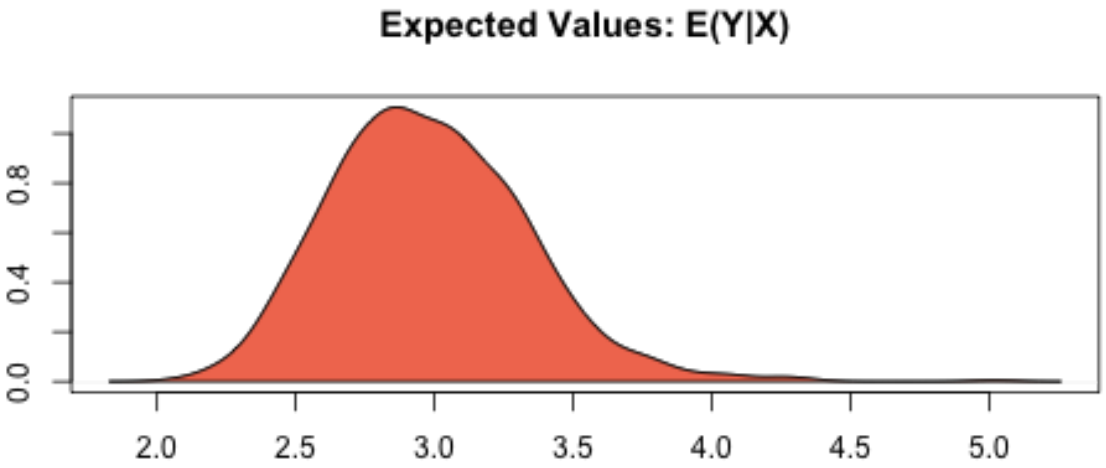


Figure 2.18: Zelig-negbin

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

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

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $E[Y_i(t_i = 0)]$, the counterfactual expected value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

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

$$\frac{1}{\sum_{i=1}^{n} t_i} \sum_{i:t_i=1}^{n} \left\{Y_i(t_i = 1) - Y_i(\widehat{t_i = 0})\right\},$$

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $Y_i(\widehat{t_i = 0})$, the counterfactual predicted value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

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 = negbin, data)`, then you may examine the available information in `z.out` by using `names(z.out)`, see the coefficients by using z.out$coefficients, and a default summary of information through `summary(z.out)`.

See also

The negative binomial model is part of the MASS package by William N. Venable and Brian D. Ripley . Advanced users may wish to refer to ‘‘help(glm.nb)‘.

zelig-normal

Normal Regression for Continuous Dependent Variables

The Normal regression model is a close variant of the more standard least squares regression model (see). Both models specify a continuous dependent variable as a linear function of a set of explanatory variables. The Normal model reports maximum likelihood (rather than least squares) estimates. The two models differ only in their estimate for the stochastic parameter σ.

Syntax

With reference classes:

```
z5 <- znormal$new()
z5$zelig(Y ~ X1 + X ~ X, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

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

Examples

Basic Example with First Differences Attach sample data:

```
data(macro)
```

Estimate model:

```
z.out1 <- zelig(unem ~ gdp + capmob + trade, model = "normal", data = macro)

## How to cite this model in Zelig:
##   Kosuke Imai, Gary King, Olivia Lau. 2008.
##   normal: Normal Regression for Continuous Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

Summarize of regression coefficients:

```
summary(z.out1)

## Model:
## $by
## [1] 1
##
##
## Call:  stats::glm(formula = unem ~ gdp + capmob + trade, family = gaussian("identity"),
##     data = .)
##
## Coefficients:
## (Intercept)          gdp       capmob        trade
##     6.18129     -0.32360      1.42194      0.01985
##
## Degrees of Freedom: 349 Total (i.e. Null);  346 Residual
## Null Deviance:       3665
## Residual Deviance: 2610  AIC: 1706
## Next step: Use 'setx' method
```

Set explanatory variables to their default (mean/mode) values, with high (80th percentile) and low (20th percentile) values for trade:

```
x.high <- setx(z.out1, trade = quantile(macro$trade, 0.8))
x.low <- setx(z.out1, trade = quantile(macro$trade, 0.2))
```

Generate first differences for the effect of high versus low trade on GDP:

```
s.out1 <- sim(z.out1, x = x.high, x1 = x.low)

summary(s.out1)

##
##  sim x :
##  -----
## ev
##          mean        sd      50%     2.5%    97.5%
```

```
## [1,] 5.422257 0.1989366 5.424447 5.028838 5.809112
## pv
##          mean       sd      50%      2.5%    97.5%
## [1,] 5.333821 2.744388 5.396203 -0.368982 10.55112
##
##  sim x1 :
##  -----
## ev
##          mean        sd      50%     2.5%    97.5%
## [1,] 4.596018 0.1876396 4.598034 4.238753 4.954279
## pv
##          mean       sd      50%     2.5%    97.5%
## [1,] 4.466941 2.863781 4.397681 -1.18165 10.10794
## fd
##            mean        sd        50%      2.5%      97.5%
## [1,] -0.8262389 0.2342038 -0.8192718 -1.279973 -0.3696419
```

A visual summary of quantities of interest:

```
plot(s.out1)
```

Model

Let Y_i be the continuous dependent variable for observation i.

- The *stochastic component* is described by a univariate normal model with a vector of means μ_i and scalar variance σ^2:

$$Y_i \sim \text{Normal}(\mu_i, \sigma^2).$$

- The *systematic component* is

$$\mu_i = x_i\beta,$$

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

Quantities of Interest

- The expected value (qi$ev) is the mean of simulations from the the stochastic component,

$$E(Y) = \mu_i = x_i\beta,$$

given a draw of β from its posterior.

- The predicted value (qi$pr) is drawn from the distribution defined by the set of parameters (μ_i, σ).
- The first difference (qi$fd) is:

$$\text{FD} = E(Y \mid x_1) - E(Y \mid x)$$

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

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

where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $E[Y_i(t_i = 0)]$, the counterfactual expected value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

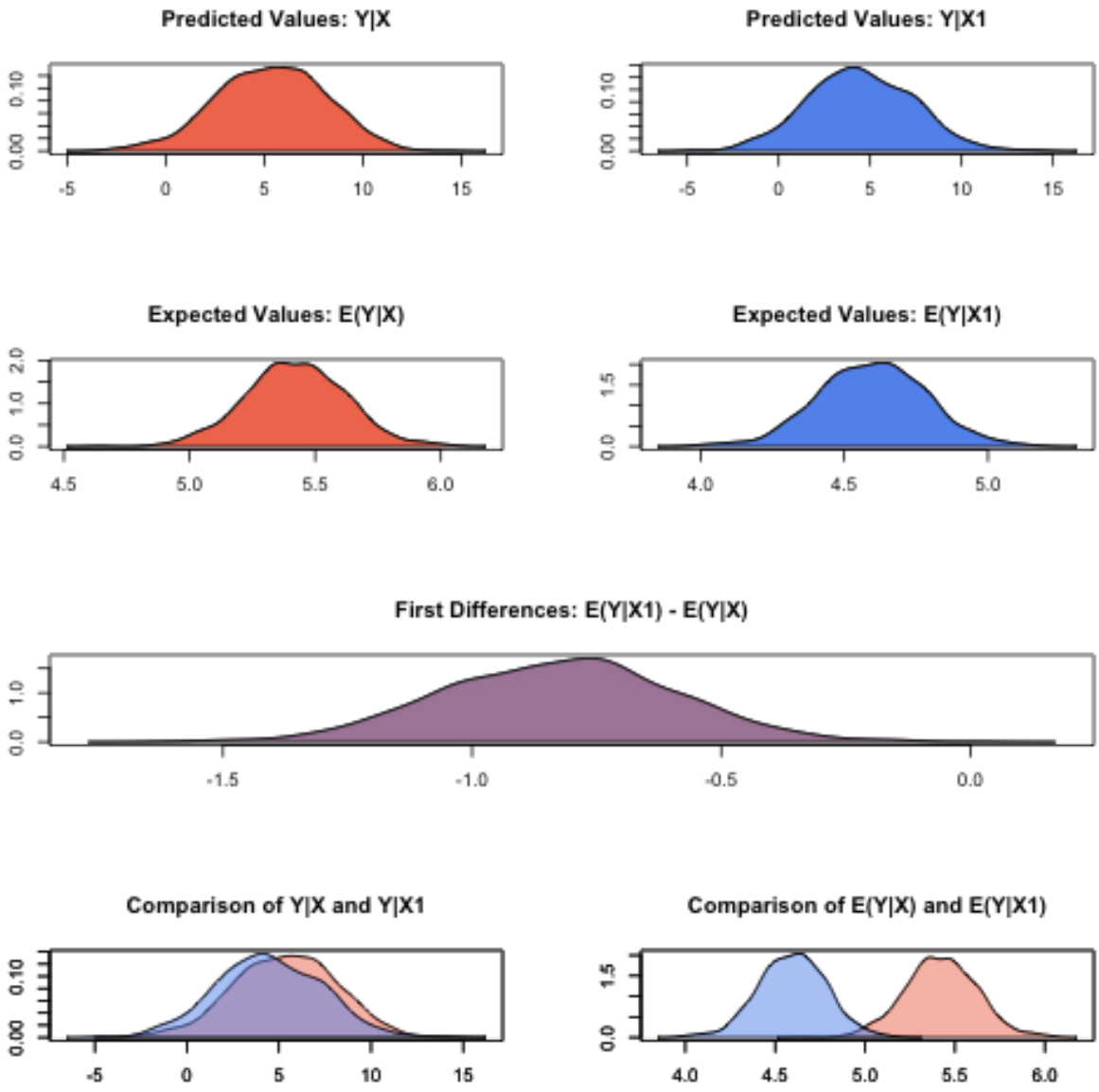


Figure 2.19: Zelig-normal

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

$$\frac{1}{\sum_{i=1}^{n} t_i} \sum_{i:t_i=1}^{n} \left\{ Y_i(t_i = 1) - Y_i(\widehat{t_i = 0}) \right\},$$

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $Y_i(\widehat{t_i = 0})$, the counterfactual predicted value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

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 = normal, data)`, then you may examine the available information in `z.out` by using `names(z.out)`, see the coefficients by using z.out$coefficients, and a default summary of information through `summary(z.out)`.

See also

The normal model is part of the stats package by . Advanced users may wish to refer to `help(glm)` and `help(family)`.

zelig-poisson

Poisson Regression for Event Count Dependent Variables

Use the Poisson regression model if the observations of your dependent variable represents the number of independent events that occur during a fixed period of time (see the negative binomial model, , for over-dispersed event counts.) For a Bayesian implementation of this model, see .

Syntax

With reference classes:

```
z5 <- zpoisson$new()
z5$zelig(Y ~ X1 + X ~ X, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

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

Example

Load sample data:

```
data(sanction)
```

Estimate Poisson model:

```
z.out <- zelig(num ~ target + coop, model = "poisson", data = sanction)

## How to cite this model in Zelig:
##   Kosuke Imai, Gary King, Olivia Lau. 2007.
##   poisson: Poisson Regression for Event Count Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/

summary(z.out)

## Model:
## $by
## [1] 1
##
##
## Call:  stats::glm(formula = num ~ target + coop, family = poisson("log"),
##     data = .)
##
## Coefficients:
## (Intercept)       target         coop
##    -0.96772     -0.02102      1.21082
##
## Degrees of Freedom: 77 Total (i.e. Null);  75 Residual
## Null Deviance:       1584
## Residual Deviance: 720.8         AIC: 944.3
## Next step: Use 'setx' method
```

Set values for the explanatory variables to their default mean values:

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

Simulate fitted values:

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

##
##  sim x :
##  -----
## ev
##          mean        sd      50%     2.5%    97.5%
## [1,] 3.247022 0.2367508 3.240276 2.803836 3.728323
## pv
##      mean       sd 50% 2.5% 97.5%
## [1,] 3.14 1.743973   3    0     7

plot(s.out)
```

Model

Let Y_i be the number of independent events that occur during a fixed time period. This variable can take any non-negative integer.

- The Poisson distribution has *stochastic component*

$$Y_i \sim \text{Poisson}(\lambda_i),$$

where λ_i is the mean and variance parameter.

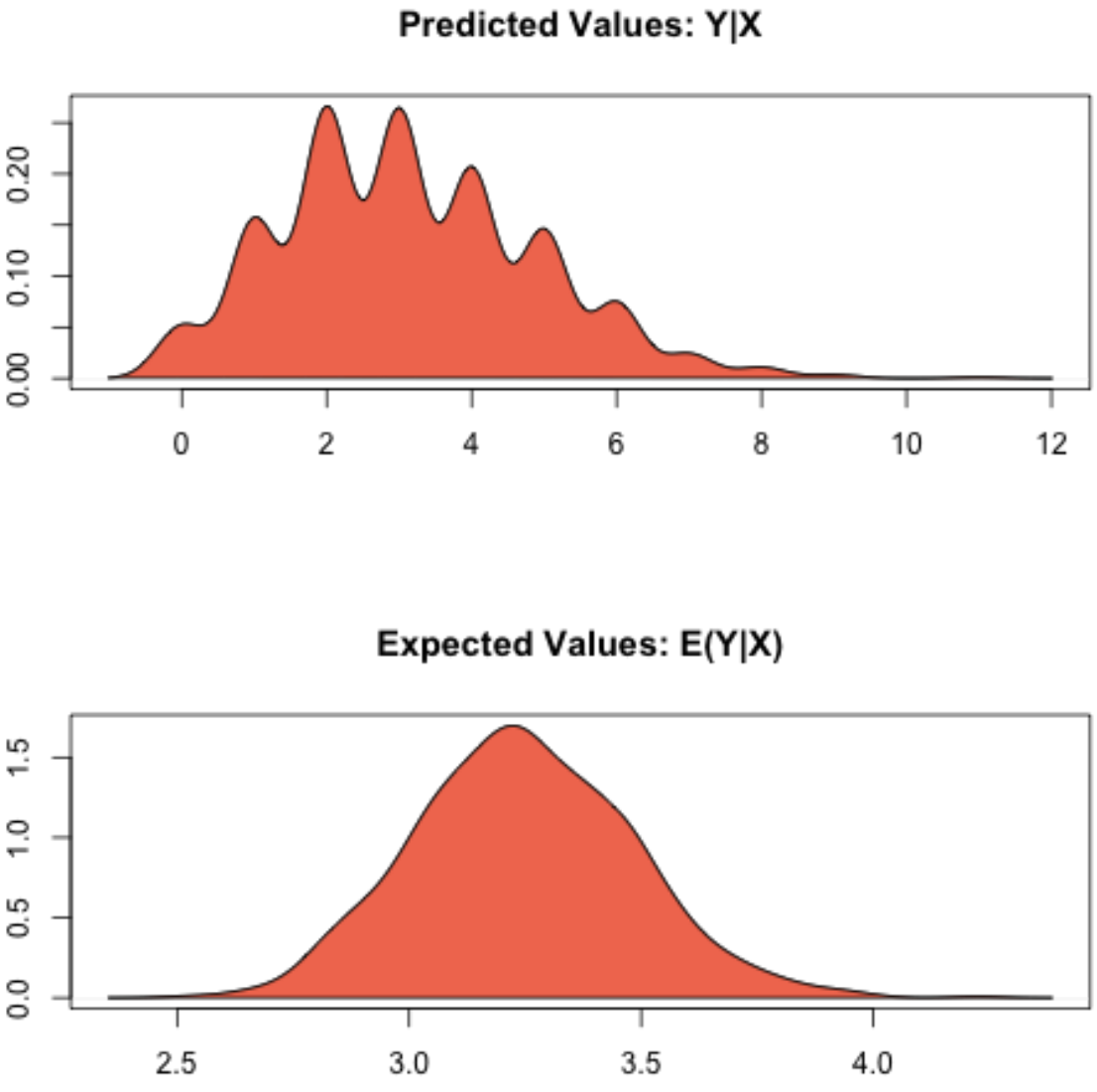


Figure 2.20: Zelig-poisson

- The *systematic component* is

$$\lambda_i \ = \ \exp(x_i\beta),$$

where x_i is the vector of explanatory variables, and β is the vector of coefficients.

Quantities of Interest

- The expected value (qi$ev) is the mean of simulations from the stochastic component,

$$E(Y) = \lambda_i = \exp(x_i\beta),$$

given draws of β from its sampling distribution.

- The predicted value (qi$pr) is a random draw from the poisson distribution defined by mean λ_i.
- The first difference in the expected values (qi$fd) is given by:

$$\text{FD} \ = \ E(Y|x_1) - E(Y \mid x)$$

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

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

where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $E[Y_i(t_i = 0)]$, the counterfactual expected value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

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

$$\frac{1}{\sum_{i=1}^{n} t_i} \sum_{i:t_i=1}^{n} \left\{ Y_i(t_i = 1) - Y_i(\widehat{t_i = 0}) \right\},$$

where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $Y_i(\widehat{t_i = 0})$, the counterfactual predicted value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

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 = poisson, data)`, then you may examine the available information in `z.out` by using `names(z.out)`, see the coefficients by using z.out$coefficients, and a default summary of information through `summary(z.out)`.

See also

The poisson model is part of the stats package by . Advanced users may wish to refer to `help(glm)` and `help(family)`.

zelig-probit

Probit Regression for Dichotomous Dependent Variables

Use probit regression to model binary dependent variables specified as a function of a set of explanatory variables.

Syntax

With reference classes:

```
z5 <- zprobit$new()
z5$zelig(Y ~ X1 + X ~ X, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

```
z.out <- zelig(Y ~ X1 + X2, model = "probit", data = mydata)
x.out <- setx(z.out)
s.out <- sim(z.out, x = x.out, x1 = NULL)
```

Example

Attach the sample turnout dataset:

```
data(turnout)
```

Estimate parameter values for the probit regression:

```
z.out <- zelig(vote ~ race + educate, model = "probit", data = turnout)
```

```
## How to cite this model in Zelig:
##   Kosuke Imai, Gary King, Olivia Lau. 2007.
##   probit: Probit Regression for Dichotomous Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

```
summary(z.out)
```

```
## Model:
## $by
## [1] 1
##
##
## Call:  stats::glm(formula = vote ~ race + educate, family = binomial("probit"),
##     data = .)
##
## Coefficients:
## (Intercept)    racewhite      educate
##    -0.72595      0.29908      0.09712
##
## Degrees of Freedom: 1999 Total (i.e. Null);  1997 Residual
## Null Deviance:       2267
## Residual Deviance: 2136  AIC: 2142
## Next step: Use 'setx' method
```

Set values for the explanatory variables to their default values.

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

Simulate quantities of interest from the posterior distribution.

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

summary(s.out)

plot(s.out1)
```

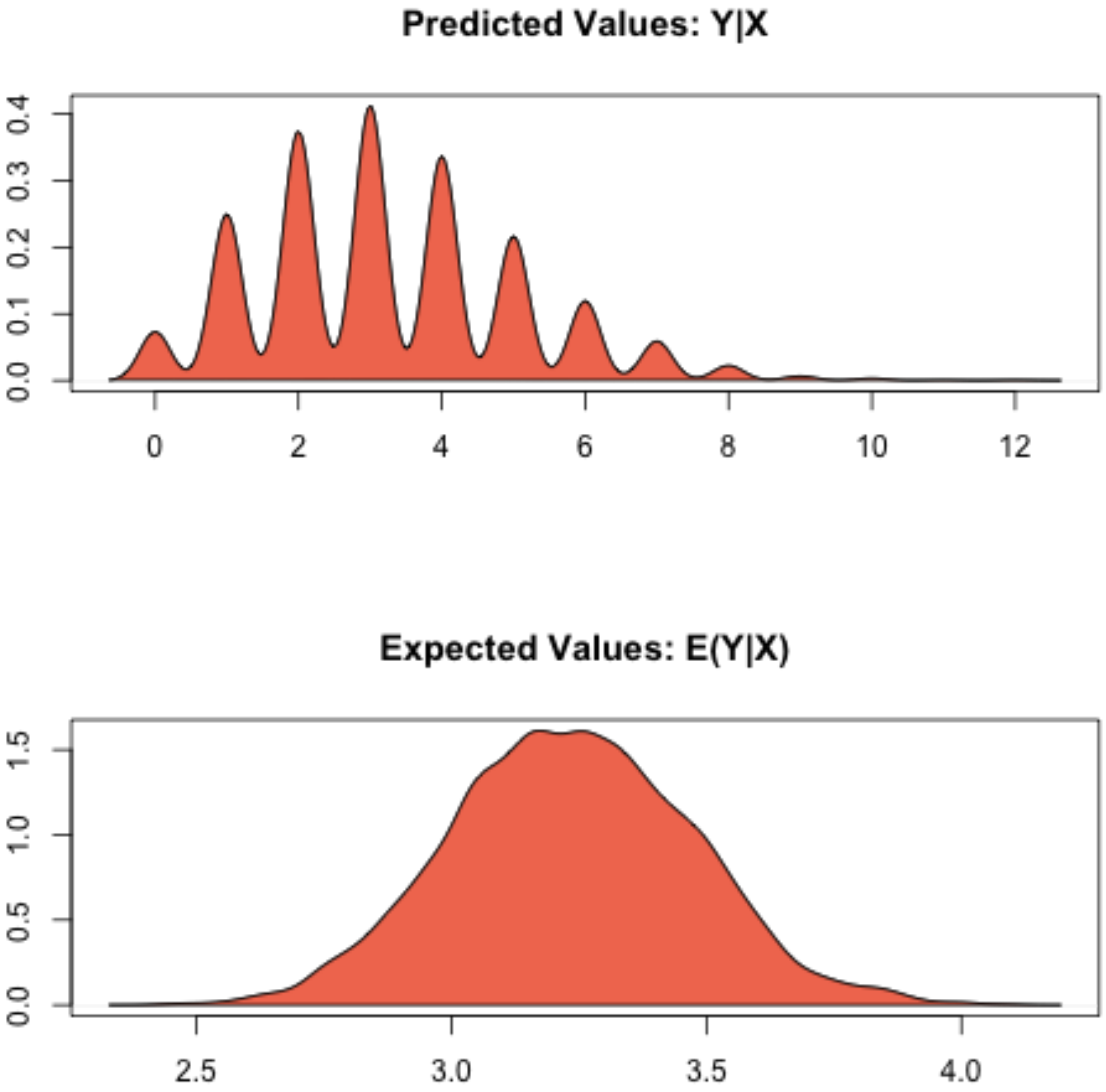


Figure 2.21: Zelig-probit

Model

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

- The *stochastic component* is given by

$$Y_i \sim \text{Bernoulli}(\pi_i),$$

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

- The *systematic component* is

$$\pi_i = \Phi(x_i\beta)$$

  where $\Phi(\mu)$ is the cumulative distribution function of the Normal distribution with mean 0 and unit variance.

Quantities of Interest

- The expected value (qi$ev) is a simulation of predicted probability of success

$$E(Y) = \pi_i = \Phi(x_i\beta),$$

  given a draw of β from its sampling distribution.

- The predicted value (qi$pr) is a draw from a Bernoulli distribution with mean π_i.

- The first difference (qi$fd) in expected values 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 (att.ev) for the treatment group is

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

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $E[Y_i(t_i = 0)]$, the counterfactual expected value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

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

$$\frac{1}{\sum_{i=1}^{n} t_i} \sum_{i:t_i=1}^{n} \left\{ Y_i(t_i = 1) - Y_i(\widehat{t_i = 0}) \right\},$$

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $Y_i(\widehat{t_i = 0})$, the counterfactual predicted value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

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 = probit, data)`, then you may examine the available information in `z.out` by using `names(z.out)`, see the coefficients by using z.out$coefficients, and a default summary of information through `summary(z.out)`.

See also

The probit model is part of the stats package by . Advanced users may wish to refer to `help(glm)` and `help(family)`.

zelig-relogit

Rare Events Logistic Regression for Dichotomous Dependent Variables

The relogit procedure estimates the same model as standard logistic regression (appropriate when you have a dichotomous dependent variable and a set of explanatory variables; see), but the estimates are corrected for the bias that occurs when the sample is small or the observed events are rare (i.e., if the dependent variable has many more 1s than 0s or the reverse). The relogit procedure also optionally uses prior correction for case-control sampling designs.

Syntax

With reference classes:

```
z5 <- zrelogit$new()
z5$zelig(Y ~ X1 + X2, tau = NULL,
                        case.control = c("prior", "weighting"),
                        bias.correct = TRUE, robust = FALSE,
                        data = mydata, ...)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

```
z.out <- zelig(Y ~ X1 + X2, model = "relogit", tau = NULL,
                        case.control = c("prior", "weighting"),
                        bias.correct = TRUE, robust = FALSE,
                        data = mydata, ...)
x.out <- setx(z.out)
s.out <- sim(z.out, x = x.out)
```

Arguments

The relogit procedure supports four optional arguments in addition to the standard arguments for zelig(). You may additionally use:

- tau: a vector containing either one or two values for τ, the true population fraction of ones. Use, for example, tau = c(0.05, 0.1) to specify that the lower bound on tau is 0.05 and the upper bound is 0.1. If left unspecified, only finite-sample bias correction is performed, not case-control correction.
- case.control: if tau is specified, choose a method to correct for case-control sampling design: "prior" (default) or "weighting".

- bias.correct: a logical value of TRUE (default) or FALSE indicating whether the intercept should be corrected for finite sample (rare events) bias.

Note that if tau = NULL, bias.correct = FALSE, the relogit procedure performs a standard logistic regression without any correction.

Example 1: One Tau with Prior Correction and Bias Correction

Due to memory and space considerations, the data used here are a sample drawn from the full data set used in King and Zeng, 2001, The proportion of militarized interstate conflicts to the absence of disputes is $\tau = 1,042/303,772 \approx 0.00343$. To estimate the model,

```
data(mid)

z.out1 <- zelig(conflict ~ major + contig + power + maxdem + mindem + years, data = mid, model = "rel

## How to cite this model in Zelig:
##   Kosuke Imai, Gary King, and Olivia Lau. 2014.
##   relogit: Rare Events Logistic Regression for Dichotomous Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

Summarize the model output:

```
summary(z.out1)

## Model:
## $by
## [1] 1
##
##
## Call:  relogit(formula = cbind(conflict, 1 - conflict) ~ major + contig +
##     power + maxdem + mindem + years, data = ., tau = 0.00343020423212146,
##     bias.correct = TRUE, case.control = "prior")
##
## Coefficients:
## (Intercept)        major       contig        power       maxdem
##    -7.50836      2.43196      4.10797      1.05358      0.04804
##     mindem        years
##    -0.06413     -0.06293
##
## Degrees of Freedom: 3125 Total (i.e. Null);  3119 Residual
## Null Deviance:       3979
## Residual Deviance: 1869  AIC: 1883
## Next step: Use 'setx' method
```

Set the explanatory variables to their means:

```
x.out1 <- setx(z.out1)
```

Simulate quantities of interest:

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

##
##  sim x :
##  -----
```

```
## ev
##             mean           sd         50%        2.5%      97.5%
## [1,] 0.002405825 0.0001548869 0.002401648 0.002119784 0.00270734
## pv
##          0     1
## [1,] 0.999 0.001
```

```
plot(s.out1)
```

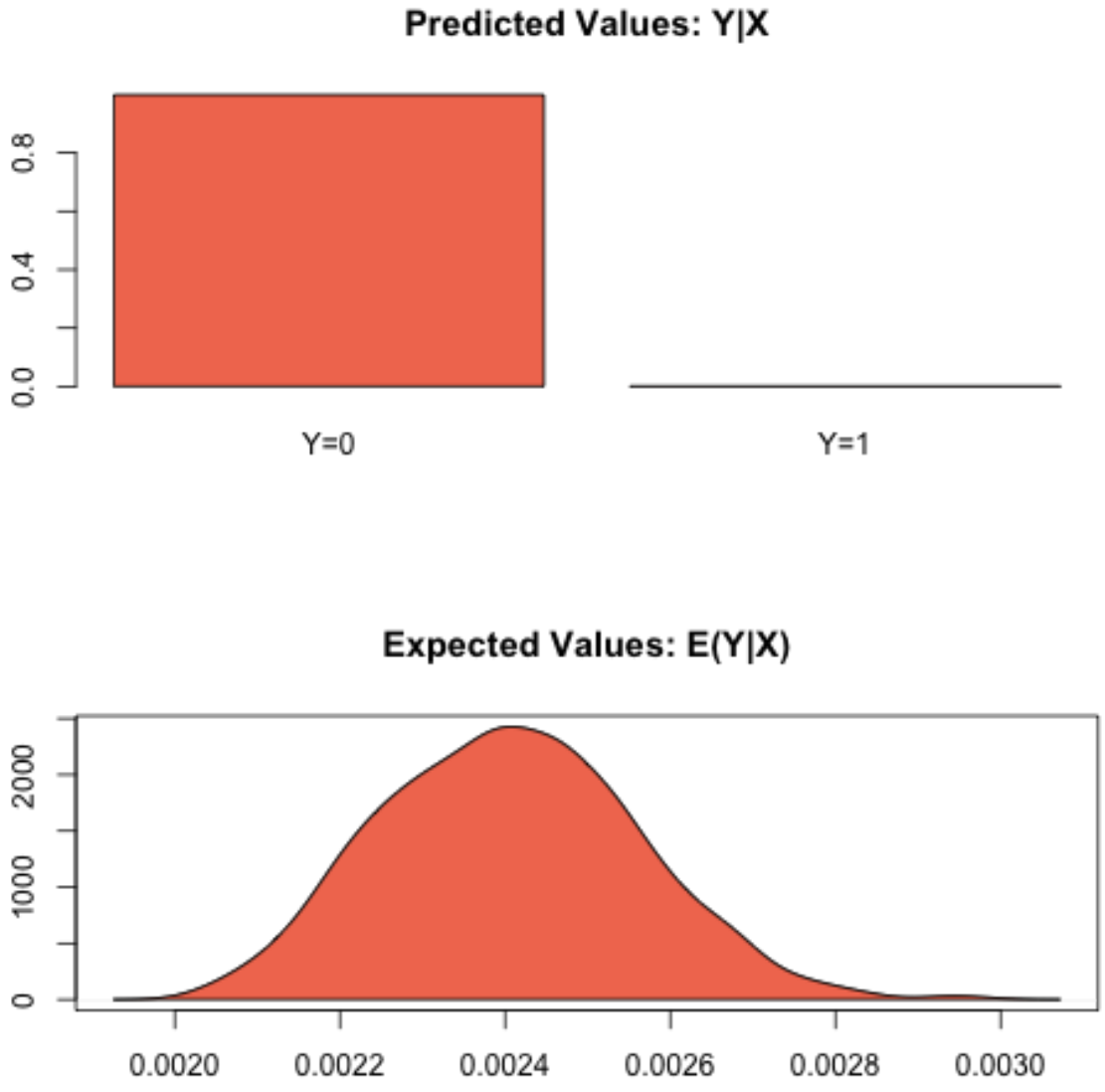


Figure 2.22: Zelig-relogit

Example 2: One Tau with Weighting, Robust Standard Errors, and Bias Correction

Suppose that we wish to perform case control correction using weighting (rather than the default prior correction). To estimate the model:

```
z.out2 <- zelig(conflict ~ major + contig + power + maxdem + mindem + years, data = mid, model = "rel

## Error in glm.control(robust = TRUE): unused argument (robust = TRUE)
```

Summarize the model output:

```
summary(z.out2)

## Model:
## $by
## [1] 1
##
##
## Call:  stats::glm(formula = vote ~ race + educate, family = binomial("logit"),
##     data = .)
##
## Coefficients:
## (Intercept)    racewhite      educate
##     -1.2189       0.5022       0.1610
##
## Degrees of Freedom: 1999 Total (i.e. Null);  1997 Residual
## Null Deviance:       2267
## Residual Deviance: 2138  AIC: 2144
## Next step: Use 'setx' method
```

Set the explanatory variables to their means:

```
x.out2 <- setx(z.out2)
```

Simulate quantities of interest:

```
s.out2 <- sim(z.out2, x = x.out2)
summary(s.out2)

##
##  sim x :
##  -----
## ev
##           mean         sd       50%      2.5%     97.5%
## [1,] 0.7728179 0.01032537 0.7730781 0.7496465 0.7921772
## pv
##          0     1
## [1,] 0.225 0.775
```

Example 3: Two Taus with Bias Correction and Prior Correction

Suppose that we did not know that $\tau \approx 0.00343$, but only that it was somewhere between $(0.002, 0.005)$. To estimate a model with a range of feasible estimates for τ (using the default prior correction method for case control correction):

```
z.out2 <- zelig(conflict ~ major + contig + power + maxdem + mindem + years, data = mid, model = "rel
```

```
## How to cite this model in Zelig:
##   Kosuke Imai, Gary King, and Olivia Lau. 2014.
##   relogit: Rare Events Logistic Regression for Dichotomous Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

Summarize the model output:

```
z.out2
```

```
## Model:
## $by
## [1] 1
##
## $lower.estimate
##
## Call:  (function (formula, data = sys.parent(), tau = NULL, bias.correct = TRUE,
##     case.control = "prior", ...)
## {
##     mf <- match.call()
##     mf$tau <- mf$bias.correct <- mf$case.control <- NULL
##     if (!is.null(tau)) {
##         tau <- unique(tau)
##         if (length(case.control) > 1)
##             stop("You can only choose one option for case control correction.")
##         ck1 <- grep("p", case.control)
##         ck2 <- grep("w", case.control)
##         if (length(ck1) == 0 & length(ck2) == 0)
##             stop("choose either case.control = \"prior\" ", "or case.control = \"weighting\"")
##         if (length(ck2) == 0)
##             weighting <- FALSE
##         else weighting <- TRUE
##     }
##     else weighting <- FALSE
##     if (length(tau) > 2)
##         stop("tau must be a vector of length less than or equal to 2")
##     else if (length(tau) == 2) {
##         mf[[1]] <- relogit
##         res <- list()
##         mf$tau <- min(tau)
##         res$lower.estimate <- eval(as.call(mf), parent.frame())
##         mf$tau <- max(tau)
##         res$upper.estimate <- eval(as.call(mf), parent.frame())
##         res$formula <- formula
##         class(res) <- c("Relogit2", "Relogit")
##         return(res)
##     }
##     else {
##         mf[[1]] <- glm
##         mf$family <- binomial(link = "logit")
##         y2 <- model.response(model.frame(mf$formula, data))
##         if (is.matrix(y2))
##             y <- y2[, 1]
##         else y <- y2
##         ybar <- mean(y)
##         if (weighting) {
##             w1 <- tau/ybar
##             w0 <- (1 - tau)/(1 - ybar)
##             wi <- w1 * y + w0 * (1 - y)
```

```
##             mf$weights <- wi
##         }
##         res <- eval(as.call(mf), parent.frame())
##         res$call <- match.call(expand.dots = TRUE)
##         res$tau <- tau
##         X <- model.matrix(res)
##         if (bias.correct) {
##             pihat <- fitted(res)
##             if (is.null(tau))
##                 wi <- rep(1, length(y))
##             else if (weighting)
##                 res$weighting <- TRUE
##             else {
##                 w1 <- tau/ybar
##                 w0 <- (1 - tau)/(1 - ybar)
##                 wi <- w1 * y + w0 * (1 - y)
##                 res$weighting <- FALSE
##             }
##             W <- pihat * (1 - pihat) * wi
##             Qdiag <- lm.influence(lm(y ~ X - 1, weights = W))$hat/W
##             if (is.null(tau))
##                 xi <- 0.5 * Qdiag * (2 * pihat - 1)
##             else xi <- 0.5 * Qdiag * ((1 + w0) * pihat - w0)
##             res$coefficients <- res$coefficients - lm(xi ~ X -
##                 1, weights = W)$coefficients
##             res$bias.correct <- TRUE
##         }
##         else res$bias.correct <- FALSE
##         if (!is.null(tau) & !weighting) {
##             if (tau <= 0 || tau >= 1)
##                 stop("\ntau needs to be between 0 and 1.\n")
##             res$coefficients["(Intercept)"] <- res$coefficients["(Intercept)"] -
##                 log(((1 - tau)/tau) * (ybar/(1 - ybar)))
##             res$prior.correct <- TRUE
##             res$weighting <- FALSE
##         }
##         else res$prior.correct <- FALSE
##         if (is.null(res$weighting))
##             res$weighting <- FALSE
##         res$linear.predictors <- t(res$coefficients) %*% t(X)
##         res$fitted.values <- 1/(1 + exp(-res$linear.predictors))
##         res$zelig <- "Relogit"
##         class(res) <- c("Relogit", "glm")
##         return(res)
##     }
## })(formula = cbind(conflict, 1 - conflict) ~ major + contig +
##     power + maxdem + mindem + years, data = ., tau = 0.002)
##
## Coefficients:
## (Intercept)        major       contig        power       maxdem
##    -8.04923      2.43196      4.10791      1.05357      0.04804
##     mindem        years
##    -0.06412     -0.06293
##
## Degrees of Freedom: 3125 Total (i.e. Null);  3119 Residual
## Null Deviance:       3979
## Residual Deviance: 1869  AIC: 1883
##
```

```
## $upper.estimate
##
## Call:  (function (formula, data = sys.parent(), tau = NULL, bias.correct = TRUE,
##     case.control = "prior", ...)
## {
##     mf <- match.call()
##     mf$tau <- mf$bias.correct <- mf$case.control <- NULL
##     if (!is.null(tau)) {
##         tau <- unique(tau)
##         if (length(case.control) > 1)
##             stop("You can only choose one option for case control correction.")
##         ck1 <- grep("p", case.control)
##         ck2 <- grep("w", case.control)
##         if (length(ck1) == 0 & length(ck2) == 0)
##             stop("choose either case.control = \"prior\" ", "or case.control = \"weighting\"")
##         if (length(ck2) == 0)
##             weighting <- FALSE
##         else weighting <- TRUE
##     }
##     else weighting <- FALSE
##     if (length(tau) > 2)
##         stop("tau must be a vector of length less than or equal to 2")
##     else if (length(tau) == 2) {
##         mf[[1]] <- relogit
##         res <- list()
##         mf$tau <- min(tau)
##         res$lower.estimate <- eval(as.call(mf), parent.frame())
##         mf$tau <- max(tau)
##         res$upper.estimate <- eval(as.call(mf), parent.frame())
##         res$formula <- formula
##         class(res) <- c("Relogit2", "Relogit")
##         return(res)
##     }
##     else {
##         mf[[1]] <- glm
##         mf$family <- binomial(link = "logit")
##         y2 <- model.response(model.frame(mf$formula, data))
##         if (is.matrix(y2))
##             y <- y2[, 1]
##         else y <- y2
##         ybar <- mean(y)
##         if (weighting) {
##             w1 <- tau/ybar
##             w0 <- (1 - tau)/(1 - ybar)
##             wi <- w1 * y + w0 * (1 - y)
##             mf$weights <- wi
##         }
##         res <- eval(as.call(mf), parent.frame())
##         res$call <- match.call(expand.dots = TRUE)
##         res$tau <- tau
##         X <- model.matrix(res)
##         if (bias.correct) {
##             pihat <- fitted(res)
##             if (is.null(tau))
##                 wi <- rep(1, length(y))
##             else if (weighting)
##                 res$weighting <- TRUE
##             else {
```

```
##                 w1 <- tau/ybar
##                 w0 <- (1 - tau)/(1 - ybar)
##                 wi <- w1 * y + w0 * (1 - y)
##                 res$weighting <- FALSE
##             }
##             W <- pihat * (1 - pihat) * wi
##             Qdiag <- lm.influence(lm(y ~ X - 1, weights = W))$hat/W
##             if (is.null(tau))
##                 xi <- 0.5 * Qdiag * (2 * pihat - 1)
##             else xi <- 0.5 * Qdiag * ((1 + w0) * pihat - w0)
##             res$coefficients <- res$coefficients - lm(xi ~ X -
##                 1, weights = W)$coefficients
##             res$bias.correct <- TRUE
##         }
##         else res$bias.correct <- FALSE
##         if (!is.null(tau) & !weighting) {
##             if (tau <= 0 || tau >= 1)
##                 stop("\ntau needs to be between 0 and 1.\n")
##             res$coefficients["(Intercept)"] <- res$coefficients["(Intercept)"] -
##                 log(((1 - tau)/tau) * (ybar/(1 - ybar)))
##             res$prior.correct <- TRUE
##             res$weighting <- FALSE
##         }
##         else res$prior.correct <- FALSE
##         if (is.null(res$weighting))
##             res$weighting <- FALSE
##         res$linear.predictors <- t(res$coefficients) %*% t(X)
##         res$fitted.values <- 1/(1 + exp(-res$linear.predictors))
##         res$zelig <- "Relogit"
##         class(res) <- c("Relogit", "glm")
##         return(res)
##     }
## })(formula = cbind(conflict, 1 - conflict) ~ major + contig +
##     power + maxdem + mindem + years, data = ., tau = 0.005)
##
## Coefficients:
## (Intercept)        major       contig        power       maxdem
##    -7.13001      2.43197      4.10805      1.05358      0.04804
##     mindem        years
##    -0.06413     -0.06294
##
## Degrees of Freedom: 3125 Total (i.e. Null);  3119 Residual
## Null Deviance:       3979
## Residual Deviance: 1869  AIC: 1883
##
## $formula
## cbind(conflict, 1 - conflict) ~ major + contig + power + maxdem +
##     mindem + years
## <environment: 0x7fc26fdce8e0>
##
## attr(,"class")
## [1] "Relogit2" "Relogit"
## Next step: Use 'setx' method
```

Set the explanatory variables to their means:

```
x.out2 <- setx(z.out2)
```

Simulate quantities of interest:

```
s.out <- sim(z.out2, x = x.out2)

## Error in UseMethod("vcov"): no applicable method for 'vcov' applied to an object of class "c('Rel

summary(s.out2)

##
##  sim x :
##  -----
## ev
##           mean         sd       50%      2.5%     97.5%
## [1,] 0.7728179 0.01032537 0.7730781 0.7496465 0.7921772
## pv
##          0     1
## [1,] 0.225 0.775

plot(s.out2)
```

The cost of giving a range of values for τ is that point estimates are not available for quantities of interest. Instead, quantities are presented as confidence intervals with significance less than or equal to a specified level (e.g., at least 95% of the simulations are contained in the nominal 95% confidence interval).

Model

- Like the standard logistic regression, the *stochastic component* for the rare events logistic regression is:

$$Y_i \sim \text{Bernoulli}(\pi_i),$$

  where Y_i is the binary dependent variable, and takes a value of either 0 or 1.

- The *systematic component* is:

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

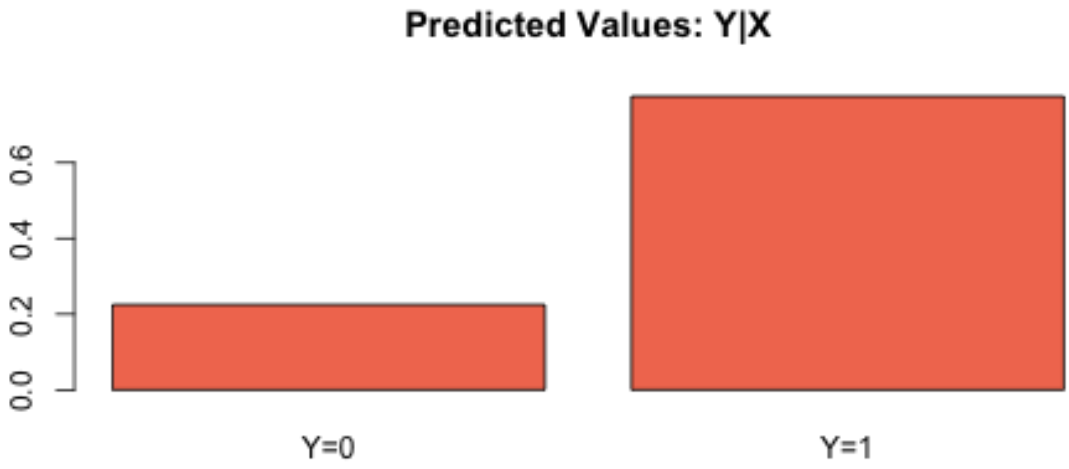
- If the sample is generated via a case-control (or choice-based) design, such as when drawing all events (or "cases") and a sample from the non-events (or "controls") and going backwards to collect the explanatory variables, you must correct for selecting on the dependent variable. While the slope coefficients are approximately unbiased, the constant term may be significantly biased. Zelig has two methods for case control correction:

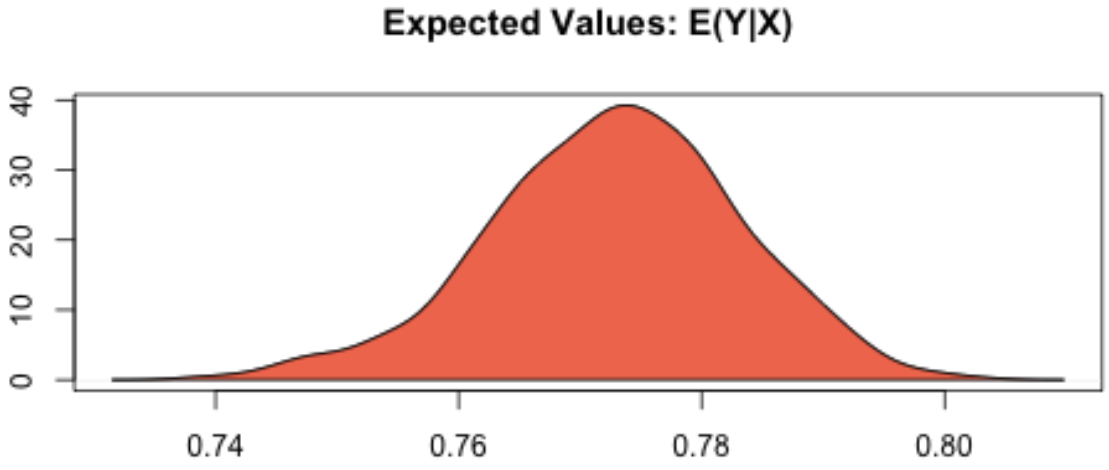
  1. The "prior correction" method adjusts the intercept term. Let τ be the true population fraction of events, $\bar{y}$ the fraction of events in the sample, and $\hat{\beta}_0$ the uncorrected intercept term. The corrected intercept β_0 is:

$$\beta = \hat{\beta}_0 - \ln\left[\left(\frac{1-\tau}{\tau}\right)\left(\frac{\bar{y}}{1-\bar{y}}\right)\right].$$

  2. The "weighting" method performs a weighted logistic regression to correct for a case-control sampling design. Let the 1 subscript denote observations for which the dependent variable is observed as a 1, and the 0 subscript denote observations for which the dependent variable is observed as a 0. Then the vector of weights w_i

$$\begin{aligned} w_1 &= \frac{\tau}{\bar{y}} \\ w_0 &= \frac{(1-\tau)}{(1-\bar{y})} \\ w_i &= w_1 Y_i + w_0(1-Y_i) \end{aligned}$$


Predicted Values: Y|X
0.0
0.2
0.4
0.6
Y=0
Y=1


Expected Values: E(Y|X)
0
10
20
30
40
0.74
0.76
0.78
0.80

If τ is unknown, you may alternatively specify an upper and lower bound for the possible range of τ. In this case, the relogit procedure uses "robust Bayesian" methods to generate a confidence interval (rather than a point estimate) for each quantity of interest. The nominal coverage of the confidence interval is at least as great as the actual coverage.

- By default, estimates of the the coefficients β are bias-corrected to account for finite sample or rare events bias. In addition, quantities of interest, such as predicted probabilities, are also corrected of rare-events bias. If $\widehat{\beta}$ are the uncorrected logit coefficients and bias($\widehat{\beta}$) is the bias term, the corrected coefficients $\tilde{\beta}$ are

$$\widehat{\beta} - \text{bias}(\widehat{\beta}) = \tilde{\beta}$$

The bias term is

$$\text{bias}(\widehat{\beta}) = (X'WX)^{-1}X'W\xi$$

where

$$\begin{aligned} \xi_i &= 0.5Q_{ii}\Big((1+w-1)\widehat{\pi}_i - w_1\Big) \\ Q &= X(X'WX)^{-1}X' \\ W &= \text{diag}\{\widehat{\pi}_i(1-\widehat{\pi}_i)w_i\} \end{aligned}$$

where w_i and w_1 are given in the "weighting" section above.

Quantities of Interest

- For either one or no τ:
  - The expected values (qi$ev) for the rare events logit are simulations of the predicted probability

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

    given draws of β from its posterior.
  - The predicted value (qi$pr) is a draw from a binomial distribution with mean equal to the simulated π_i.
  - The first difference (qi$fd) is defined as

$$\text{FD} = \Pr(Y = 1 \mid x_1, \tau) - \Pr(Y = 1 \mid x, \tau).$$

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

$$\text{RR} = \Pr(Y = 1 \mid x_1, \tau) \;/\; \Pr(Y = 1 \mid x, \tau).$$

- For a range of τ defined by $[\tau_1, \tau_2]$, each of the quantities of interest are $n \times 2$ matrices, which report the lower and upper bounds, respectively, for a confidence interval with nominal coverage at least as great as the actual coverage. At worst, these bounds are conservative estimates for the likely range for each quantity of interest. Please refer to for the specific method of calculating bounded quantities of interest.
- In conditional prediction models, the average expected treatment effect (att.ev) for the treatment group is

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

where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $E[Y_i(t_i = 0)]$, the counterfactual expected value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

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

$$\frac{1}{\sum_{i=1}^{n} t_i} \sum_{i:t_i=1}^{n} \left\{ Y_i(t_i = 1) - Y_i(\widehat{t_i = 0}) \right\},$$

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups. Variation in the simulations are due to uncertainty in simulating $Y_i(\widehat{t_i = 0})$, the counterfactual predicted value of Y_i for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $t_i = 0$.

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 = relogit, data)`, then you may examine the available information in `z.out` by using `names(z.out)`, see the coefficients by using z.out$coefficients, and a default summary of information through `summary(z.out)`.

Differences with Stata Version

The Stata version of ReLogit and the R implementation differ slightly in their coefficient estimates due to differences in the matrix inversion routines implemented in R and Stata. Zelig uses orthogonal-triangular decomposition (through lm.influence()) to compute the bias term, which is more numerically stable than standard matrix calculations.

See also

zelig-tobit

Linear Regression for a Left-Censored Dependent Variable

Tobit regression estimates a linear regression model for a left-censored dependent variable, where the dependent variable is censored from below. While the classical tobit model has values censored at 0, you may select another censoring point. For other linear regression models with fully observed dependent variables, see Bayesian regression (), maximum likelihood normal regression (), or least squares ().

Syntax

```
z5 <- ztobit$new()
z5$zelig(Y ~ X1 + X2, below = 0, above = Inf, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

```
z.out <- zelig(Y ~ X1 + X2, below = 0, above = Inf, model = "tobit", data = mydata)
x.out <- setx(z.out)
s.out <- sim(z.out, x = x.out)
```

Inputs

zelig() accepts the following arguments to specify how the dependent variable is censored.

- `below`: (defaults to 0) The point at which the dependent variable is censored from below. If any values in the dependent variable are observed to be less than the censoring point, it is assumed that that particular observation is censored from below at the observed value. (See for a Bayesian implementation that supports both left and right censoring.)
- robust: defaults to FALSE. If TRUE, zelig() computes robust standard errors based on sandwich estimators (see and) and the options selected in cluster.
- cluster: if robust = TRUE, you may select a variable to define groups of correlated observations. Let x3 be a variable that consists of either discrete numeric values, character strings, or factors that define strata. Then

  ```
  > z.out <- zelig(y ~ x1 + x2, robust = TRUE, cluster = "x3",
                   model = "tobit", data = mydata)
  ```

  means that the observations can be correlated within the strata defined by the variable x3, and that robust standard errors should be calculated according to those clusters. If robust = TRUE but cluster is not specified, zelig() assumes that each observation falls into its own cluster.

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

Examples

Basic Example Attaching the sample dataset:

```
data(tobin)
```

Estimating linear regression using `tobit`:

```
z.out <- zelig(durable ~ age + quant, model = "tobit", data = tobin)

## How to cite this model in Zelig:
##   Kosuke Imai, Gary King, Olivia Lau. 2011.
##   tobit: Linear regression for Left-Censored Dependent Variable
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

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)

##
##  sim x :
##  -----
## ev
##       mean        sd      50%      2.5%    97.5%
## 1 1.585198 0.6748437 1.514415 0.5011679 3.044189
## pv
##          mean       sd      50% 2.5%    97.5%
## [1,] 3.252505 4.272525 1.582813    0 14.40365
```

Simulating First Differences Set explanatory variables to their default(mean/mode) values, with high (80th percentile) and low (20th percentile) liquidity ratio (`quant`):

```
x.high <- setx(z.out, quant = quantile(tobin$quant, prob = 0.8))
x.low <- setx(z.out, quant = quantile(tobin$quant, prob = 0.2))
```

Estimating the first difference for the effect of high versus low liquidity ratio on duration(`durable`):

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

summary(s.out2)
```

```
##
##  sim x :
##  -----
## ev
##       mean        sd      50%      2.5%    97.5%
## 1 1.162711 0.7248203 1.040084 0.1338574 2.950335
## pv
##          mean       sd     50% 2.5%    97.5%
## [1,] 2.763072 3.929297 0.61721    0 12.85066
##
##  sim x1 :
##  -----
## ev
##       mean        sd      50%      2.5%   97.5%
## 1 2.056761 0.9245484 1.892733 0.6058304 4.08382
## pv
##          mean       sd      50% 2.5%    97.5%
## [1,] 3.496784 4.334128 1.932983    0 14.11736
## fd
##        mean       sd       50%      2.5%    97.5%
## 1 0.8940502 1.123632 0.8431692 -1.251169 3.200797
```

```
plot(s.out1)
```

Model

- Let Y_i^* be a latent dependent variable which is distributed with *stochastic* component

$$Y_i^* \sim \text{Normal}(\mu_i, \sigma^2)$$

where μ_i is a vector means and σ^2 is a scalar variance parameter. Y_i^* is not directly observed, however. Rather we observed Y_i which is defined as:

$$Y_i = \begin{cases} Y_i^* & \text{if} \quad c < Y_i^* \\ c & \text{if} \quad c \geq Y_i^* \end{cases}$$

where c is the lower bound below which Y_i^* is censored.

- The *systematic component* is given by

$$\mu_i = x_i\beta,$$

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

Quantities of Interest

- The expected values (`qi$ev`) for the tobit regression model are the same as the expected value of $Y*$:

$$E(Y^*|X) = \mu_i = x_i\beta$$

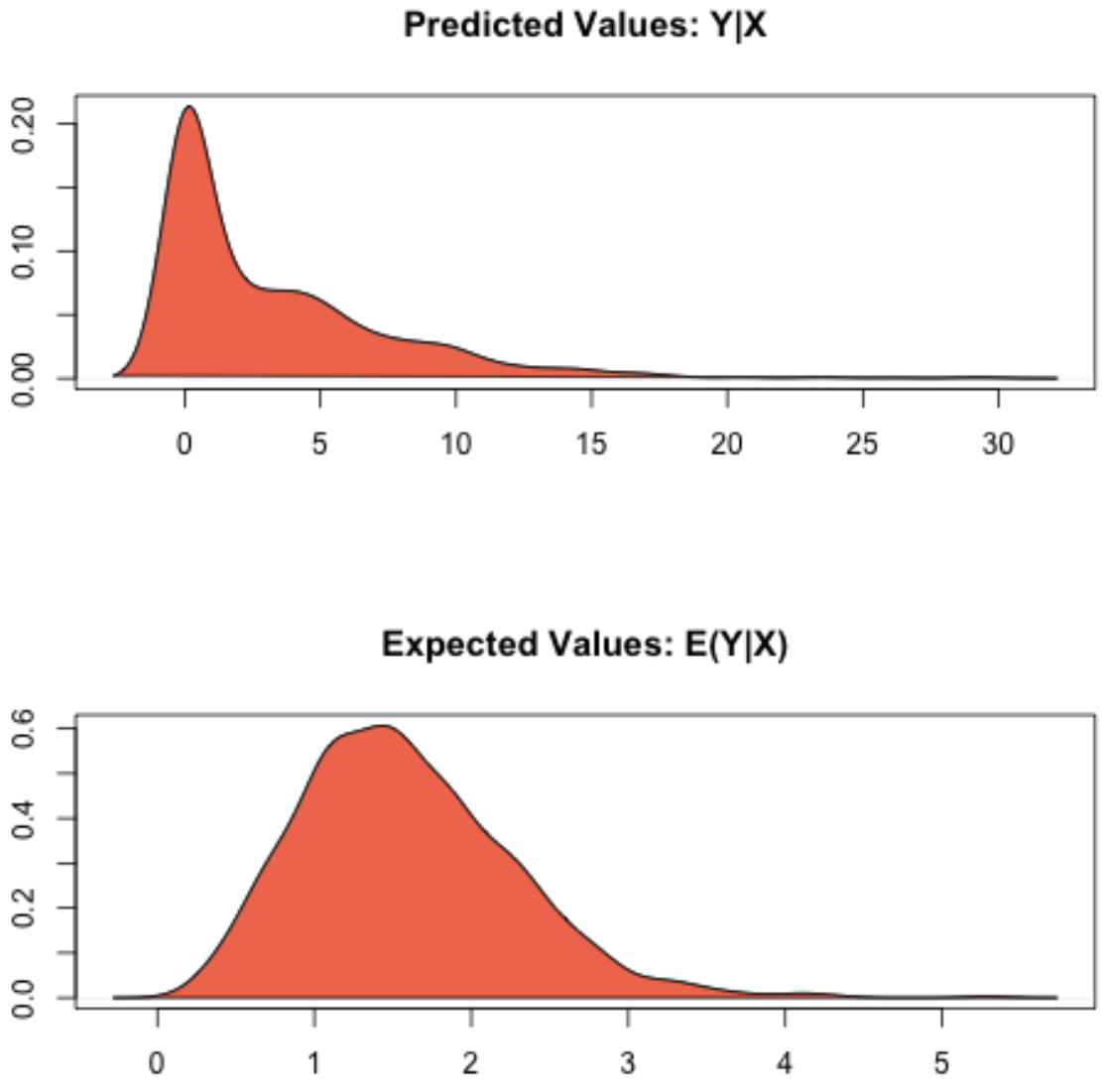


Figure 2.23: Zelig-tobit

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

$$\text{FD} = E(Y^* \mid x_1) - E(Y^* \mid x).$$

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

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

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

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 = "tobit", data)
```

then you may examine the available information in ‘‘z.out‘.

See also

The tobit function is part of the survival library by Terry Therneau, ported to R by Thomas Lumley. Advanced users may wish to refer to `help(survfit)` in the survival library.

zelig-factorbayes

Given some unobserved explanatory variables and observed dependent variables, the Normal theory factor analysis model estimates the latent factors. The model is implemented using a Markov Chain Monte Carlo algorithm (Gibbs sampling with data augmentation). For factor analysis with ordinal dependent variables, see ordered factor analysis (), and for a mix of types of dependent variables, see the mixed factor analysis model ().

Syntax

With reference classes:

```
z5 <- zfactorbayes$new()
z5$zelig(cbind(Y1 ,Y2, Y3) ~ NULL, factors = 2,
         model = "factor.bayes", data = mydata)
```

With the Zelig 4 compatibility wrappers:

```
z.out <- zelig(cbind(Y1 ,Y2, Y3) ~ NULL, factors = 2,
               model = "factor.bayes", data = mydata)
```

Inputs

zelig() takes the following functions for factor.bayes:

- `Y1`, `Y2`, and `Y3`: variables of interest in factor analysis (manifest variables), assumed to be normally distributed. The model requires a minimum of three manifest variables.
- `factors`: number of the factors to be fitted (defaults to 2).

Additional Inputs

In addition, zelig() accepts the following additional arguments for model specification:

- `lambda.constraints`: list containing the equality or inequality constraints on the factor loadings. Choose from one of the following forms:
  - varname = list(): by default, no constraints are imposed.
  - `varname = list(d, c)`: constrains the dth loading for the variable named `varname` to be equal to `c`.
  - `varname = list(d, +)`: constrains the dth loading for the variable named `varname` to be positive;
  - `varname = list(d, -)`: constrains the dth loading for the variable named `varname` to be negative.
- `std.var`: defaults to FALSE (manifest variables are rescaled to zero mean, but retain observed variance). If `TRUE`, the manifest variables are rescaled to be mean zero and unit variance.

In addition, zelig() accepts the following additional inputs for bayes.factor:

- `burnin`: number of the initial MCMC iterations to be discarded (defaults to 1,000).
- `mcmc`: number of the MCMC iterations after burnin (defaults to 20,000).
- `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.
- `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 12345.
- `Lambda.start`: starting values of the factor loading matrix Λ, either a scalar (all unconstrained loadings are set to that value), or a matrix with compatible dimensions. The default is `NA`, where the start value are set to be 0 for unconstrained factor loadings, and 0.5 or -0.5 for constrained factor loadings (depending on the nature of the constraints).
- `Psi.start`: starting values for the uniquenesses, either a scalar (the starting values for all diagonal elements of Ψ are set to be this value), or a vector with length equal to the number of manifest variables. In the latter case, the starting values of the diagonal elements of Ψ take the values of `Psi.start`. The default value is `NA` where the starting values of the all the uniquenesses are set to be 0.5.
- `store.lambda`: defaults to TRUE, which stores the posterior draws of the factor loadings.
- `store.scores`: defaults to FALSE. If TRUE, stores the posterior draws of the factor scores. (Storing factor scores may take large amount of memory for a large number of draws or observations.)

The model also accepts the following additional arguments to specify prior parameters:

- `l0`: mean of the Normal prior for the factor loadings, either a scalar or a matrix with the same dimensions as Λ. If a scalar value, that value will be the prior mean for all the factor loadings. Defaults to 0.
- `L0`: precision parameter of the Normal prior for the factor loadings, either a scalar or a matrix with the same dimensions as Λ. If `L0` takes a scalar value, then the precision matrix will be a diagonal matrix with the diagonal elements set to that value. The default value is 0, which leads to an improper prior.
- `a0`: the shape parameter of the Inverse Gamma prior for the uniquenesses is `a0/2`. It can take a scalar value or a vector. The default value is 0.001.
- `b0`: the shape parameter of the Inverse Gamma prior for the uniquenesses is `b0/2`. It can take a scalar value or a vector. The default value is 0.001.

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

Example

Attaching the sample dataset:

```
data(swiss)
names(swiss) <- c("Fert", "Agr", "Exam", "Educ", "Cath", "InfMort")
```

Factor analysis:

```
z.out <- zelig(cbind(Agr, Exam, Educ, Cath, InfMort) ~ NULL,
               model = "factor.bayes", data = swiss, factors = 2, verbose = TRUE,
               a0 = 1, b0 = 0.15, burnin = 5000, mcmc = 50000)
```

Checking for convergence before summarizing the estimates:

```
algor <- try(geweke.diag(z.out$coefficients), silent=T)
if (class(algor) == "try-error")
    print(algor)
```

Since the algorithm did not converge, we now add some constraints on Λ to optimize the algorithm:

```
z.out <- zelig(cbind(Agr, Exam, Educ, Cath, InfMort) ~ NULL,
               model = "factor.bayes", data = swiss, factors = 2,
               lambda.constraints = list(Exam = list(1,"+"),
                   Exam = list(2,"-"), Educ = c(2, 0),
                   InfMort = c(1, 0)),
               verbose = TRUE, a0 = 1, b0 = 0.15,
               burnin = 5000, mcmc = 50000)
geweke.diag(z.out$coefficients)
heidel.diag(z.out$coefficients)
raftery.diag(z.out$coefficients)
summary(z.out)
```

Model

Suppose for observation i we observe K variables and hypothesize that there are d underlying factors such that:

$$Y_i = \Lambda\phi_i + \epsilon_i$$

where Y_i is the vector of K manifest variables for observation i. Λ is the $K \times d$ factor loading matrix and ϕ_i is the d-vector of latent factor scores. Both Λ and ϕ need to be estimated.

- The *stochastic component* is given by:

  $$\epsilon_i \sim \text{Normal}(0, \Psi).$$

  where Ψ is a diagonal, positive definite matrix. The diagonal elements of Ψ are referred to as uniquenesses.

- The *systematic component* is given by

  $$\mu_i = E(Y_i) = \Lambda\phi_i$$

- The independent conjugate *prior* for each Λ_{ij} is given by

  $$\Lambda_{ij} \sim \text{Normal}(l_{0_{ij}}, L_{0_{ij}}^{-1}) \text{ for } i = 1, \ldots, k; \quad j = 1, \ldots, d.$$

- The independent conjugate *prior* for each Ψ_{ii} is given by

  $$\Psi_{ii} \sim \text{InverseGamma}(\frac{a_0}{2}, \frac{b_0}{2}), \text{ for } i = 1, \ldots, k.$$

- The *prior* for ϕ_i is

$$\phi_i \sim \quad \text{Normal}(0, I_d), \text{ for } i = 1, \ldots, n.$$

  where I_d is a :math:‘ dtimes d ‘ identity matrix.

Output Values

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

```
z.out <- zelig(cbind(Y1, Y2, Y3), model = "factor.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$coefficients`, and view a default summary of information through `summary(z.out)`. Other elements available through the `$` operator are listed below.

- From the `zelig()` output object `z.out`, you may extract:
  - `coefficients`: draws from the posterior distributions of the estimated factor loadings and the uniquenesses. If `store.scores = TRUE`, the estimated factors scores are also contained in `coefficients`.
  - `data`: the name of the input data frame.
  - `seed`: the random seed used in the model.
- Since there are no explanatory variables, the `sim()` procedure is not applicable for factor analysis models.

zelig-mlogitbayes

Use Bayesian multinomial logistic regression to model unordered categorical variables. The dependent variable may be in the format of either character strings or integer values. The model is estimated via a random walk Metropolis algorithm or a slice sampler. See for the maximum-likelihood estimation of this model.

Syntax

With reference classes:

```
z5 <- zmlogitbayes$new()
z5$zelig(Y ~ X1 + X2, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

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

Additional Inputs

zelig() accepts the following arguments for mlogit.bayes:

- `baseline`: either a character string or numeric value (equal to one of the observed values in the dependent variable) specifying a baseline category. The default value is `NA` which sets the baseline to the first alphabetical or numerical unique value of the dependent variable.

The model accepts the following additional arguments to monitor the Markov chains:

- `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.
- `mcmc.method`: either "MH" or "slice", specifying whether to use Metropolis Algorithm or slice sampler. The default value is `MH`.
- `tune`: tuning parameter for the Metropolis-Hasting step, either a scalar or a numeric vector (for k coefficients, enter a k vector). The tuning parameter should be set such that the acceptance rate is satisfactory (between 0.2 and 0.5). 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 a vector (for k coefficients, enter a k vector). The default is `NA` where the maximum likelihood estimates are used as the starting values.

Use the following arguments to specify the priors for the model:

- `b0`: prior mean for the coefficients, either a scalar or vector. If a scalar, 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, 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(MCMCmnl)` for more information.

Examples

Basic Example Attaching the sample dataset:

```
data(mexico)
```

Estimating multinomial logistics regression using `mlogit.bayes`:

```
z.out <- zelig(vote88 ~ pristr + othcok + othsocok,
               model = "mlogit.bayes", data = mexico,
               verbose = FALSE)

## Calculating MLEs and large sample var-cov matrix.
## This may take a moment...
## Inverting Hessian to get large sample var-cov matrix.

## Warning in if (mcmc.method == "RWM") {: the condition has length > 1 and
## only the first element will be used

## Warning in if (mcmc.method == "IndMH") {: the condition has length > 1 and
## only the first element will be used

## How to cite this model in Zelig:
##   Ben Goodrich, Ying Lu. 2013.
##   mlogitbayes: Bayesian Multinomial Logistic Regression for Dependent Variables with Unordered Ca
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

Checking for convergence before summarizing the estimates:

```
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)
```

```
##
##  sim x :
##  -----
## ev
##             mean         sd       50%      2.5%     97.5%
## P(Y=1) 0.5613368 0.01592425 0.5615034 0.5306640 0.5914963
## P(Y=2) 0.2099124 0.01273148 0.2098424 0.1854312 0.2350891
## P(Y=3) 0.2287508 0.01360126 0.2285987 0.2033590 0.2558153
## pv
## qi
##      1      2      3
## 0.5586 0.2083 0.2331
```

Simulating First Differences Estimating the first difference (and risk ratio) in the probabilities of voting different candidates when `pristr` (the strength of the PRI) is set to be weak (equal to 1) versus strong (equal to 3) while all the other variables held at their default values.

```
x.weak <- setx(z.out, pristr = 1)
x.strong <- setx(z.out, pristr = 3)
s.out2 <- sim(z.out, x = x.strong, x1 = x.weak)
summary(s.out2)
```

```
##
##  sim x :
##  -----
## ev
##             mean         sd       50%      2.5%     97.5%
## P(Y=1) 0.7156880 0.02127842 0.7158103 0.6725681 0.7561260
## P(Y=2) 0.1270237 0.01458077 0.1265905 0.1000858 0.1562571
## P(Y=3) 0.1572883 0.01646202 0.1568142 0.1260809 0.1909916
## pv
## qi
##      1      2      3
## 0.7158 0.1258 0.1584
##
##  sim x1 :
##  -----
## ev
##             mean         sd       50%      2.5%     97.5%
## P(Y=1) 0.4028126 0.02357831 0.4028038 0.3563194 0.4483880
## P(Y=2) 0.3037026 0.02130587 0.3029289 0.2638074 0.3470994
## P(Y=3) 0.2934848 0.02189140 0.2931780 0.2517546 0.3372056
## pv
```

```
## qi
##      1      2      3
## 0.4055 0.2996 0.2949
## fd
##               mean         sd        50%        2.5%      97.5%
## P(Y=1) -0.3128754 0.03459857 -0.3128662 -0.38111485 -0.2442630
## P(Y=2)  0.1766789 0.02735176  0.1764581  0.12360341  0.2313796
## P(Y=3)  0.1361965 0.02881430  0.1363242  0.07966018  0.1935930
```

Model

Let Y_i be the (unordered) categorical dependent variable for observation i which takes an integer values $j = 1, \ldots, J$.

- The *stochastic component* is given by:

$$Y_i \sim \quad \text{Multinomial}(Y_i \mid \pi_{ij}).$$

  where $\pi_{ij} = \Pr(Y_i = j)$ for $j = 1, \ldots, J$.

- The *systematic component* is given by

$$\pi_{ij} = \frac{\exp(x_i \beta_j)}{\sum_{k=1}^{J} \exp(x_i \beta_k)}, \text{ for } j = 1, \ldots, J-1,$$

  where x_i is the vector of k explanatory variables for observation i and β_j is the vector of coefficient for category j. Category J is assumed to be the baseline category.

- The *prior* for β is given by

$$\beta_j \sim \text{Normal}_k \left(b_0, B_0^{-1}\right) \text{ for } j = 1, \ldots, J-1,$$

  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 multinomial logistics regression model are the predicted probability of belonging to each category:

$$\Pr(Y_i = j) = \pi_{ij} = \frac{\exp(x_i \beta_j)}{\sum_{k=1}^{J} \exp(x_J \beta_k)}, \quad \text{for } j = 1, \ldots, J-1,$$

  and

$$\Pr(Y_i = J) = 1 - \sum_{j=1}^{J-1} \Pr(Y_i = j)$$

  given the posterior draws of β_j for all categories from the MCMC iterations.

- The predicted values (`qi$pr`) are the draws of Y_i from a multinomial distribution whose parameters are the expected values(`qi$ev`) computed based on the posterior draws of β from the MCMC iterations.

- The first difference (`qi$fd`) in category j for the multinomial logistic model is defined as

$$\text{FD}_j = \Pr(Y_i = j \mid X_1) - \Pr(Y_i = j \mid X).$$

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

$$\text{RR}_j = \Pr(Y_i = j \mid X_1) \;/\; \Pr(Y_i = j \mid X).$$

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

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

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups, and n_j is the number of treated observations in category j.

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

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

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups, and n_j is the number of treated observations in category j.

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 = "mlogit.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$coefficients`, and view a default summary of information through `summary(z.out)`. Other elements available through the `$` operator are listed below.

See also

Bayesian logistic regression is part of the MCMCpack library by Andrew D. Martin and Kevin M. Quinn . The convergence diagnostics are part of the CODA library by Martyn Plummer, Nicky Best, Kate Cowles, Karen Vines, Deepayan Sarkar, Russell Almond.

zelig-oprobitbayes

Use the ordinal probit regression model if your dependent variables are ordered and categorical. They may take either integer values or character strings. The model is estimated using a Gibbs sampler with data augmentation. For a maximum-likelihood implementation of this models, see *probit*.

Syntax

With reference classes:

```
z5 <- zoprobitbayes$new()
z5$zelig(Y ~ X1 + X2, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

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

Additional Inputs

zelig() accepts 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 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`: tuning parameter for the Metropolis-Hasting step. The default value is `NA` which corresponds to 0.05 divided by the number of categories in the response variable.
- `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 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`, which uses the maximum likelihood estimates 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 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(MCMCoprobit)` for more information.

Examples

Basic Example Attaching the sample dataset:

```
data(sanction)
```

Estimating ordered probit regression using `oprobit.bayes`:

```
z.out <- zelig(ncost ~ mil + coop, model = "oprobit.bayes",
               data = sanction, verbose = FALSE)

## Warning in model.response(mf, "numeric"): using type = "numeric" with a
## factor response will be ignored

## How to cite this model in Zelig:
##   Ben Goodrich, Ying Lu. 2013.
##   oprobitbayes: Bayesian Probit Regression for Dichotomous Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

Creating an ordered dependent variable:

```
sanction$ncost <- factor(sanction ~ ncost, ordered = TRUE,
                         levels = c("net gain", "little effect", "modest loss",
                                    "major loss"))

## Error in as.vector(x, mode): invalid 'mode' argument
```

Checking for convergence before summarizing the estimates:

```
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)

##
##  sim x :
##  -----
## ev
##                      mean         sd        50%       2.5%      97.5%
## little effect 0.44981581 0.05601155 0.44883053 0.34151148 0.56103192
## major loss    0.04473004 0.02101827 0.04176495 0.01271310 0.09412959
## modest loss   0.12341501 0.03950140 0.11984735 0.06005967 0.22834556
## net gain      0.38203914 0.05548348 0.38097907 0.27602445 0.49205957
## pv
## qi
## little effect    major loss   modest loss      net gain
##        0.1868        0.2731        0.5213        0.0188
```

Simulating First Differences Estimating the first difference (and risk ratio) in the probabilities of incurring different level of cost when there is no military action versus military action while all the other variables held at their default values.

```
x.high <- setx(z.out, mil = 0)
x.low <- setx(z.out, mil = 1)

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

##
##  sim x :
##  -----
## ev
##                      mean         sd        50%       2.5%      97.5%
## little effect 0.43844669 0.05843957 0.43767439 0.32758262 0.55410854
## major loss    0.04458012 0.02095061 0.04165022 0.01271633 0.09387773
## modest loss   0.12377654 0.03963082 0.12024928 0.05993637 0.22927426
## net gain      0.39319665 0.05795514 0.39196529 0.28336346 0.50825527
## pv
## qi
## little effect    major loss   modest loss      net gain
```

```
##        0.1491        0.2382        0.5780        0.0347
##
##  sim x1 :
##  -----
## ev
##                     mean         sd        50%       2.5%      97.5%
## little effect 0.5464229 0.16109327 0.54796938 0.23474072 0.84451223
## major loss    0.0407613 0.01998891 0.03763614 0.01085385 0.08828575
## modest loss   0.1075956 0.03975880 0.10421634 0.04208043 0.20132712
## net gain      0.3052203 0.14485132 0.29018143 0.07362204 0.62315150
## pv
## qi
## little effect    major loss   modest loss      net gain
##        0.6116        0.0963        0.1862        0.1059
## fd
##                       mean         sd          50%        2.5%       97.5%
## little effect  0.107976200 0.17020693  0.111082214 -0.22740084 0.426282862
## major loss    -0.003818825 0.00665593 -0.001418880 -0.02253975 0.002180789
## modest loss   -0.016180976 0.02172754 -0.008327805 -0.07580491 0.005015329
## net gain      -0.087976398 0.15275950 -0.102164237 -0.34480882 0.241227653
```

Model

Let Y_i be the ordered categorical dependent variable for observation i which takes an integer value $j = 1, \ldots, J$.

- The *stochastic component* is described by an unobserved continuous variable, Y_i^*,

$$Y_i^* \sim \text{Normal}(\mu_i, 1).$$

  Instead of Y_i^*, we observe categorical variable Y_i,

$$Y_i = j \quad \text{if } \tau_{j-1} \leq Y_i^* \leq \tau_j \text{ for } j = 1, \ldots, J.$$

  where τ_j for $j = 0, \ldots, J$ are the threshold parameters with the following constraints, $\tau_l < \tau_m$ for $l < m$, and $\tau_0 = -\infty, \tau_J = \infty$.

  The probability of observing Y_i equal to category j is,

$$\Pr(Y_i = j) = \quad \Phi(\tau_j \mid \mu_i) - \Phi(\tau_{j-1} \mid \mu_i) \text{ for } j = 1, \ldots, J$$

  where $\Phi(\cdot \mid \mu_i)$ is the cumulative distribution function of the Normal distribution with mean μ_i and variance 1.

- The *systematic component* is given by

$$\mu_i = x_i \beta,$$

  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 ordered probit model are the predicted probability of belonging to each category:

$$\Pr(Y_i = j) = \Phi(\tau_j \mid x_i\beta) - \Phi(\tau_{j-1} \mid x_i\beta),$$

  given the posterior draws of β and threshold parameters τ from the MCMC iterations.

- The predicted values (`qi$pr`) are the observed values of Y_i given the observation scheme and the posterior draws of β and cut points τ from the MCMC iterations.

- The first difference (`qi$fd`) in category j for the ordered probit model is defined as

$$\text{FD}_j = \Pr(Y_i = j \mid X_1) - \Pr(Y_i = j \mid X).$$

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

$$\text{RR}_j = \Pr(Y_i = j \mid X_1) \;/\; \Pr(Y_i = j \mid X).$$

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

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

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups, and n_j is the number of observations in the treatment group that belong to category j.

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

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

  where t_i is a binary explanatory variable defining the treatment ($t_i = 1$) and control ($t_i = 0$) groups, and n_j is the number of observations in the treatment group that belong to category j.

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 = "oprobit.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$coefficients`, and view a default summary of information through `summary(z.out)`. Other elements available through the `$` operator are listed below.

- From the `zelig()` output object `z.out`, you may extract:
  - `coefficients`: draws from the posterior distributions of the estimated coefficients β and threshold parameters τ. Note, element τ_1 is normalized to 0 and is not returned in the `coefficients` object.
  - 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$ev`: the simulated expected values (probabilities) of each of the J categories for the specified values of `x`.
- `qi$pr`: the simulated predicted values (observed values) for the specified values of `x`.
- `qi$fd`: the simulated first difference in the expected values of each of the J categories for the values specified in `x` and `x1`.
- `qi$rr`: the simulated risk ratio for the expected values of each of the J categories 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.

See also

Bayesian ordinal probit regression is part of the MCMCpack library by Andrew D. Martin and Kevin M. Quinn . The convergence diagnostics are part of the CODA library by Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines.

zelig-poissonbayes

Use the Poisson regression model if the observations of your dependent variable represents the number of independent events that occur during a fixed period of time. The model is fit using a random walk Metropolis algorithm. For a maximum-likelihood estimation of this model see *poisson*.

Syntax

With reference classes:

```
z5 <- zpoissonbayes$new()
z5$zelig(Y ~ X1 + X2, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

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

Additional Inputs

Use the following argument 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). The default value is 1.1.
- `verbose`: default 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, 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 the coefficients) or a scalar. If a scalar, 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(MCMCpoisson)` for more information.

Examples

Basic Example Attaching the sample dataset:

```
data(sanction)
```

Estimating the Poisson regression using `poisson.bayes`:

```
z.out <- zelig(num ~ target + coop, model = "poisson.bayes",
               data = sanction, verbose = FALSE)

## How to cite this model in Zelig:
##   Ben Goodrich, Ying Lu. 2013.
##   poissonbayes: Bayesian Poisson Regression
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

Checking 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)
```

Simulating First Differences Estimating the first difference in the number of countries imposing sanctions when the number of targets is set to be its maximum versus its minimum :

```
x.max <- setx(z.out, target = max(sanction$target))
x.min <- setx(z.out, target = min(sanction$target))

s.out2 <- sim(z.out, x = x.max, x1 = x.min)
summary(s.out2)
```

```
##
##  sim x :
##  -----
## ev
##          mean        sd      50%     2.5%    97.5%
## [1,] 3.191614 0.2936585 3.183013 2.642371 3.803733
## pv
##        mean       sd 50% 2.5% 97.5%
## [1,] 3.2091 1.805409   3    0     7
##
##  sim x1 :
##  -----
## ev
##          mean        sd      50%     2.5%    97.5%
## [1,] 3.306252 0.3059862 3.300095 2.729466 3.944022
## pv
##        mean       sd 50% 2.5% 97.5%
## [1,] 3.3227 1.851353   3    0     7
## fd
##           mean        sd       50%       2.5%     97.5%
## [1,] 0.1146376 0.3671544 0.1265036 -0.6072035 0.8342282
```

Model

Let Y_i be the number of independent events that occur during a fixed time period.

- The *stochastic component* is given by

$$Y_i \sim \text{Poisson}(\lambda_i)$$

  where λ_i is the mean and variance parameter.

- The *systematic component* is given by

$$\lambda_i = \exp(x_i \beta)$$

  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 Poisson model are calculated as following:

$$E(Y \mid X) = \lambda_i = \exp(x_i \beta),$$

given the posterior draws of β based on the MCMC iterations.

- The predicted values (`qi$pr`) are draws from the Poisson distribution with parameter λ_i.
- The first difference (`qi$fd`) for the Poisson model is defined as

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

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

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

  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

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

  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 = "poisson.bayes", data)
```

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$coefficients`, and view a default summary of information through `summary(z.out)`. Other elements available through the `$` 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$ev`: the simulated expected values for the specified values of `x`.
  - `qi$pr`: the simulated predicted values for the specified values of `x`.
  - `qi$fd`: the simulated first difference in the expected values for the values specified in `x` and `x1`.
  - `qi$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.

See also

Bayesian poisson regression is part of the MCMCpack library by Andrew D. Martin and Kevin M. Quinn . The convergence diagnostics are part of the CODA library by Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines.

zelig-probitbayes

Use the probit regression model for model binary dependent variables specified as a function of a set of explanatory variables. The model is estimated using a Gibbs sampler. For other models suitable for binary response variables, see Bayesian logistic regression, maximum likelihood logit regression, and maximum likelihood probit regression.

Syntax

With reference classes:

```
z5 <- zprobitbayes$new()
z5$zelig(Y ~ X1 + X2, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

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

Additional Inputs

Using the following arguments to monitor the Markov chains:

- `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.
- `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 the 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.

Use the following arguments to specify optional output for the model:

- `bayes.resid`: defaults to FALSE. If TRUE, the latent Bayesian residuals for all observations are returned. Alternatively, users can specify a vector of observations for which the latent residuals should be returned.

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

Examples

Basic Example Attaching the sample dataset:

```
data(turnout)
```

Estimating the probit regression using `probit.bayes`:

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

## How to cite this model in Zelig:
##   Ben Goodrich, Ying Lu. 2013.
##   probitbayes: Bayesian Probit Regression for Dichotomous Dependent Variables
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/
```

Checking for convergence 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)
```

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 are 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

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

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

- The *systematic component* is given by

$$\pi_i = \Phi(x_i\beta),$$

where $\Phi(\cdot)$ is the cumulative density function of the standard Normal distribution with mean 0 and variance 1, 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 probit model are the predicted probability of a success:

$$E(Y \mid X) = \pi_i = \Phi(x_i\beta),$$

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 probit 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

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

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 = "probit.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$coefficients`, and view a default summary of information through `summary(z.out)`. Other elements available through the `$` 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.
  - `bayes.residuals`: When `bayes.residual` is `TRUE` or a set of observation numbers is given, this object contains the posterior draws of the latent Bayesian residuals of all the observations or the observations specified by the user.
  - `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.

See also

Bayesian probit regression is part of the MCMCpack library by Andrew D. Martin and Kevin M. Quinn . The convergence diagnostics are part of the CODA library by Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines.

zelig-tobitbayes

Bayesian tobit regression estimates a linear regression model with a censored dependent variable using a Gibbs sampler. The dependent variable may be censored from below and/or from above. For other linear regression models with fully observed dependent variables, see Bayesian regression, maximum likelihood normal regression, or least squares.

Syntax

With reference classes:

```
z5 <- zprobitbayes$new()
z5$zelig((Y ~ X1 + X2, below = 0, above = Inf, data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

```
z.out <- zelig(Y ~ X1 + X2, below = 0, above = Inf,
               model = "tobit.bayes", data = mydata)
x.out <- setx(z.out)
s.out <- sim(z.out, x = x.out)
```

Inputs

zelig() accepts the following arguments to specify how the dependent variable is censored.

- `below`: point at which the dependent variable is censored from below. If the dependent variable is only censored from above, set `below = -Inf`. The default value is 0.
- `above`: point at which the dependent variable is censored from above. If the dependent variable is only censored from below, set `above = Inf`. The default value is `Inf`.

Additional Inputs

Use the following arguments to monitor the convergence of 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.
- `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 least squares 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, that value will be the prior mean for all coefficients. The default is 0.
- `B0`: prior precision parameter for the coefficients, either a square matrix (with the dimensions equal to the number of the coefficients) or a scalar. If a scalar, that value times an identity matrix will be the prior precision parameter. The default is 0, which leads to an improper prior.
- `c0`: `c0/2` is the shape parameter for the Inverse Gamma prior on the variance of the disturbance terms.
- `d0`: `d0/2` is the scale parameter for the Inverse Gamma prior on the variance of the disturbance terms.

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

Examples

Basic Example Attaching the sample dataset:

```
data(tobin)
```

Estimating linear regression using `tobit.bayes`:

```
z.out <- zelig(durable ~ age + quant, model = "tobit.bayes",
               data = tobin, verbose = FALSE)
```

```
## Error in zelig(durable ~ age + quant, model = "tobit.bayes", data = tobin, : Model 'tobit.bayes' r
```

Checking for convergence 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)
```

Simulating First Differences Set explanatory variables to their default(mean/mode) values, with high (80th percentile) and low (20th percentile) liquidity ratio (`quant`):

```
x.high <- setx(z.out, quant = quantile(tobin$quant, prob = 0.8))
x.low <- setx(z.out, quant = quantile(tobin$quant, prob = 0.2))
```

Estimating the first difference for the effect of high versus low liquidity ratio on duration(`durable`):

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

summary(s.out2)
```

Model

Let Y_i^* be the dependent variable which is not directly observed. Instead, we observe Y_i which is defined as following:

$$Y_i = \begin{cases} Y_i^* & \text{if} \quad c_1 < Y_i^* < c_2 \\ c_1 & \text{if} \quad c_1 \geq Y_i^* \\ c_2 & \text{if} \quad c_2 \leq Y_i^* \end{cases}$$

where c_1 is the lower bound below which Y_i^* is censored, and c_2 is the upper bound above which Y_i^* is censored.

- The *stochastic component* is given by

$$\epsilon_i \sim \quad \text{Normal}(0, \sigma^2)$$

  where $\epsilon_i = Y_i^* - \mu_i$.

- The *systematic component* is given by

$$\mu_i = x_i \beta,$$

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

- The *semi-conjugate priors* for β and σ^2 are given by

$$\begin{aligned} \beta \sim & \qquad \text{Normal}_k \left(b_0, B_0^{-1}\right) \\ \sigma^2 \sim & \quad \text{InverseGamma} \left(\frac{c_0}{2}, \frac{d_0}{2}\right) \end{aligned}$$

  where b_0 is the vector of means for the k explanatory variables, B_0 is the $k \times k$ precision matrix (the inverse of a variance-covariance matrix), and $c_0/2$ and $d_0/2$ are the shape and scale parameters for σ^2. Note that β and σ^2 are assumed *a priori* independent.

Quantities of Interest

- The expected values (`qi$ev`) for the tobit regression model is calculated as following. Let

$$\begin{aligned}\Phi_1 &= \Phi\left(\frac{(c_1 - x\beta)}{\sigma}\right)\\ \Phi_2 &= \Phi\left(\frac{(c_2 - x\beta)}{\sigma}\right)\\ \phi_1 &= \phi\left(\frac{(c_1 - x\beta)}{\sigma}\right)\\ \phi_2 &= \phi\left(\frac{(c_2 - x\beta)}{\sigma}\right)\end{aligned}$$

  where $\Phi(\cdot)$ is the (cumulative) Normal density function and $\phi(\cdot)$ is the Normal probability density function of the standard normal distribution. Then the expected values are

$$\begin{aligned}E(Y|x) &= P(Y^* \leq c_1|x)c_1 + P(c_1 < Y^* < c_2|x)E(Y^* \mid c_1 < Y^* < c_2, x) + P(Y^* \geq c_2)c_2\\ &= \Phi_1 c_1 + x\beta(\Phi_2 - \Phi_1) + \sigma(\phi_1 - \phi_2) + (1 - \Phi_2)c_2,\end{aligned}$$

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

$$\text{FD} = E(Y \mid x_1) - E(Y \mid x).$$

- 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.

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 = "tobit.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$coefficients`, and view a default summary of information through `summary(z.out)`. Other elements available through the `$` operator are listed below.

- From the `zelig()` output object `z.out`, you may extract:
  - `coefficients`: draws from the posterior distributions of the estimated parameters. The first k columns contain the posterior draws of the coefficients β, and the last column contains the posterior draws of the variance σ^2.
  - 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$ev`: the simulated expected value for the specified values of `x`.
  - `qi$fd`: the simulated first difference in the expected values given the values specified in `x` and `x1`.
  - `qi$att.ev`: the simulated average expected treatment effect for the treated from conditional prediction models.

See also

Bayesian tobit regression is part of the MCMCpack library by Andrew D. Martin and Kevin M. Quinn . The convergence diagnostics are part of the CODA library by Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines.

zelig-gammagee

The GEE gamma is similar to standard gamma regression (appropriate when you have an uncensored, positive-valued, continuous dependent variable such as the time until a parliamentary cabinet falls). Unlike in gamma regression, GEE gamma allows for dependence within clusters, such as in longitudinal data, although its use is not limited to just panel data. GEE models make no distributional assumptions but require three specifications: a mean function, a variance function, and a "working" correlation matrix for the clusters, which models the dependence of each observation with other observations in the same cluster. The "working" correlation matrix is a $T \times T$ matrix of correlations, where T is the size of the largest cluster and the elements of the matrix are correlations between within-cluster observations. The appeal of GEE models is that it gives consistent estimates of the parameters and consistent estimates of the standard errors can be obtained using a robust "sandwich" estimator even if the "working" correlation matrix is incorrectly specified. If the "working" correlation matrix is correctly specified, GEE models will give more efficient estimates of the parameters. GEE models measure population-averaged effects as opposed to cluster-specific effects.

Syntax

With reference classes:

```
z5 <- zgammagee$new()
z5$zelig(Y ~ X1 + X2, model = "gamma.gee",
         id = "X3", data = mydata)
z5$setx()
z5$sim()
```

With the Zelig 4 compatibility wrappers:

```
z.out <- zelig(Y ~ X1 + X2, model = "gamma.gee",
               id = "X3", data = mydata)
x.out <- setx(z.out)
s.out <- sim(z.out, x = x.out)
```

where `id` is a variable which identifies the clusters. The data should be sorted by `id` and should be ordered within each cluster when appropriate.

Additional Inputs

Use the following arguments to specify the structure of the "working" correlations within clusters:

- `corstr`: character string specifying the correlation structure: "independence", "exchangeable", "ar1", "unstructured" and "userdefined"
- See `geeglm` in package `geepack` for other function arguments.

Examples

Example with Exchangeable Dependence Attaching the sample turnout dataset:

```
data(coalition)
```

Sorted variable identifying clusters

```
coalition$cluster <- c(rep(c(1:62), 5),rep(c(63), 4))
sorted.coalition <- coalition[order(coalition$cluster), ]
```

Estimating model and presenting summary:

```
z.out <- zelig(duration ~ fract + numst2, model = "gamma.gee",
               id = "cluster", data = sorted.coalition,
               corstr = "exchangeable")

## How to cite this model in Zelig:
##   Patrick Lam. 2011.
##   zgammagee: General Estimating Equation for Gamma Regression
##   in Kosuke Imai, Gary King, and Olivia Lau, "Zelig: Everyone's Statistical Software,"
##   http://zeligproject.org/

summary(z.out)

## Model:
## $by
## [1] 1
##
##
## Call:
## geepack::geeglm(formula = duration ~ fract + numst2, family = Gamma("inverse"),
##     data = ., id = c(1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3,
##     3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 7, 7, 7,
##     7, 7, 8, 8, 8, 8, 8, 9, 9, 9, 9, 9, 10, 10, 10, 10, 10, 11,
##     11, 11, 11, 11, 12, 12, 12, 12, 12, 13, 13, 13, 13, 13, 14,
##     14, 14, 14, 14, 15, 15, 15, 15, 15, 16, 16, 16, 16, 16, 17,
##     17, 17, 17, 17, 18, 18, 18, 18, 18, 19, 19, 19, 19, 19, 20,
##     20, 20, 20, 20, 21, 21, 21, 21, 21, 22, 22, 22, 22, 22, 23,
##     23, 23, 23, 23, 24, 24, 24, 24, 24, 25, 25, 25, 25, 25, 26,
##     26, 26, 26, 26, 27, 27, 27, 27, 27, 28, 28, 28, 28, 28, 29,
##     29, 29, 29, 29, 30, 30, 30, 30, 30, 31, 31, 31, 31, 31, 32,
##     32, 32, 32, 32, 33, 33, 33, 33, 33, 34, 34, 34, 34, 34, 35,
##     35, 35, 35, 35, 36, 36, 36, 36, 36, 37, 37, 37, 37, 37, 38,
##     38, 38, 38, 38, 39, 39, 39, 39, 39, 40, 40, 40, 40, 40, 41,
##     41, 41, 41, 41, 42, 42, 42, 42, 42, 43, 43, 43, 43, 43, 44,
##     44, 44, 44, 44, 45, 45, 45, 45, 45, 46, 46, 46, 46, 46, 47,
##     47, 47, 47, 47, 48, 48, 48, 48, 48, 49, 49, 49, 49, 49, 50,
##     50, 50, 50, 50, 51, 51, 51, 51, 51, 52, 52, 52, 52, 52, 53,
##     53, 53, 53, 53, 54, 54, 54, 54, 54, 55, 55, 55, 55, 55, 56,
##     56, 56, 56, 56, 57, 57, 57, 57, 57, 58, 58, 58, 58, 58, 59,
##     59, 59, 59, 59, 60, 60, 60, 60, 60, 61, 61, 61, 61, 61, 62,
##     62, 62, 62, 62, 63, 63, 63, 63), corstr = "exchangeable")
##
## Coefficients:
##   (Intercept)         fract        numst2
## -0.0129634262  0.0001149139 -0.0174009664
##
## Degrees of Freedom: 314 Total (i.e. Null);  311 Residual
##
## Scale Link:                   identity
## Estimated Scale Parameters:  [1] 0.6231419
##
```

```
## Correlation:  Structure = exchangeable    Link = identity
## Estimated Correlation Parameters:
##        alpha
## -0.008086333
##
## Number of clusters:   63   Maximum cluster size: 5
##
## Next step: Use 'setx' method
```

Setting the explanatory variables at their default values (mode for factor variables and mean for non-factor variables), with numst2 set to the vector 0 = no crisis, 1 = crisis.

```
x.low <- setx(z.out, numst2 = 0)
x.high <- setx(z.out, numst2 = 1)
```

Simulate quantities of interest

```
s.out <- sim(z.out, x = x.low, x1 = x.high)
summary(s.out)
```

```
##
##  sim x :
##  -----
## ev
##          mean       sd     50%     2.5%    97.5%
## [1,] 14.40728 1.150051 14.3262 12.36709 16.89536
## pv
##          mean       sd      50%       2.5%    97.5%
## [1,] 14.07835 18.14224 7.603815 0.06235246 65.07778
##
##  sim x1 :
##  -----
## ev
##          mean       sd      50%    2.5%    97.5%
## [1,] 19.21808 1.117628 19.18863 17.2676 21.63706
## pv
##          mean       sd      50%       2.5%    97.5%
## [1,] 19.61789 23.70813 11.10875 0.04327947 82.02938
## fd
##          mean       sd      50%     2.5%   97.5%
## [1,] 4.810795 1.677545 4.759365 1.566173 8.00721
```

Generate a plot of quantities of interest:

```
plot(s.out)
```

The Model

Suppose we have a panel dataset, with Y_{it} denoting the positive-valued, continuous dependent variable for unit i at time t. Y_i is a vector or cluster of correlated data where y_{it} is correlated with $y_{it'}$ for some or all t, t'. Note that the model assumes correlations within i but independence across i.

- The *stochastic component* is given by the joint and marginal distributions

$$\begin{aligned} Y_i \sim & \quad f(y_i \mid \lambda_i) \\ Y_{it} \sim & \quad g(y_{it} \mid \lambda_{it}) \end{aligned}$$

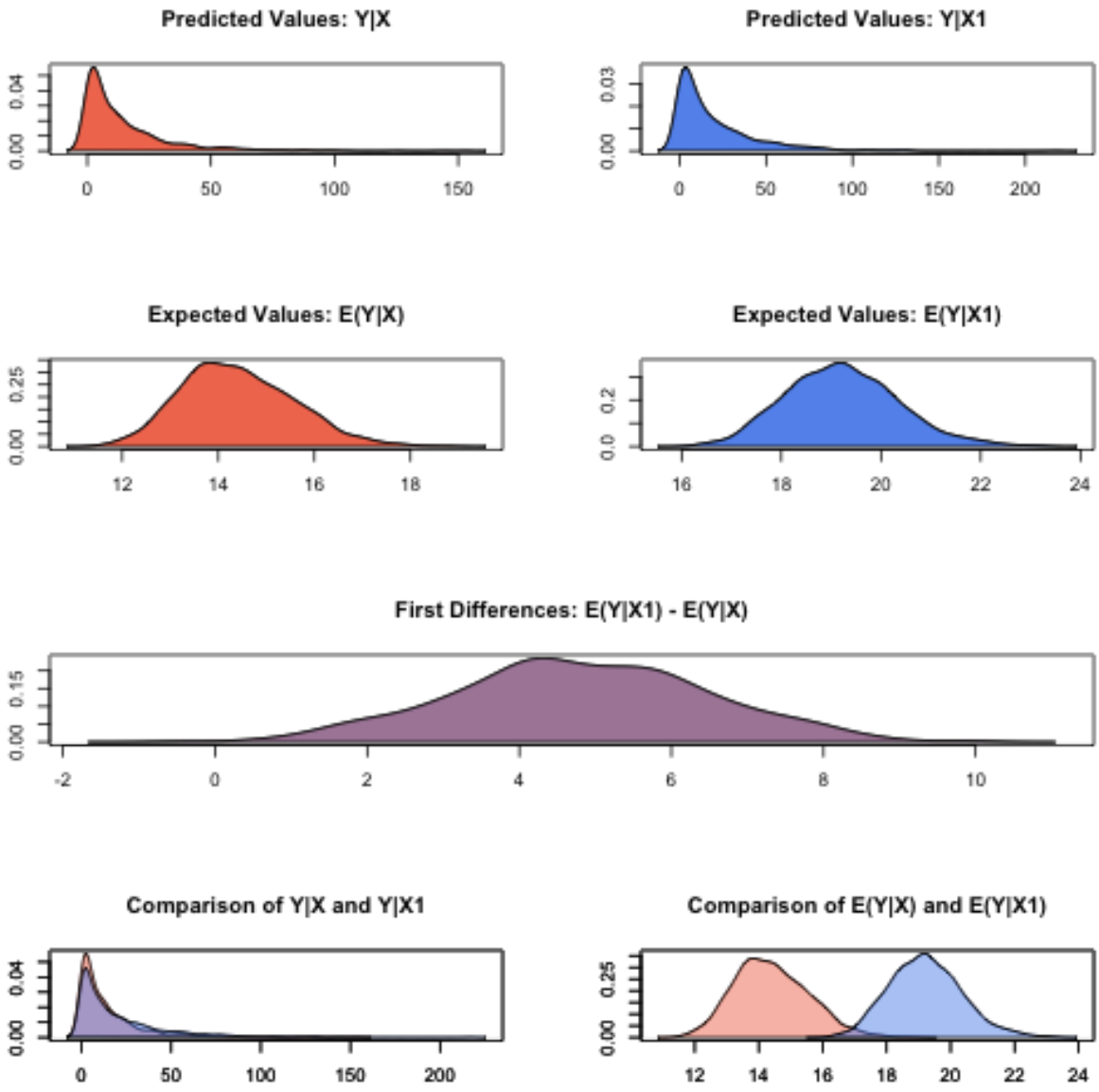


Figure 2.24: plot of chunk unnamed-chunk-10

where f and g are unspecified distributions with means λ_i and λ_{it}. GEE models make no distributional assumptions and only require three specifications: a mean function, a variance function, and a correlation structure.

- The *systematic component* is the *mean function*, given by:

$$\lambda_{it} = \frac{1}{x_{it}\beta}$$

where x_{it} is the vector of k explanatory variables for unit i at time t and β is the vector of coefficients.

- The *variance function* is given by:

$$V_{it} = \lambda_{it}^2 = \frac{1}{(x_{it}\beta)^2}$$

- The *correlation structure* is defined by a $T \times T$ "working" correlation matrix, where T is the size of the largest cluster. Users must specify the structure of the "working" correlation matrix *a priori*. The "working" correlation matrix then enters the variance term for each i, given by:

$$V_i = \phi\, A_i^{\frac{1}{2}} R_i(\alpha) A_i^{\frac{1}{2}}$$

where A_i is a $T \times T$ diagonal matrix with the variance function $V_{it} = \lambda_{it}^2$ as the tth diagonal element, $R_i(\alpha)$ is the "working" correlation matrix, and ϕ is a scale parameter. The parameters are then estimated via a quasi-likelihood approach.

- In GEE models, if the mean is correctly specified, but the variance and correlation structure are incorrectly specified, then GEE models provide consistent estimates of the parameters and thus the mean function as well, while consistent estimates of the standard errors can be obtained via a robust "sandwich" estimator. Similarly, if the mean and variance are correctly specified but the correlation structure is incorrectly specified, the parameters can be estimated consistently and the standard errors can be estimated consistently with the sandwich estimator. If all three are specified correctly, then the estimates of the parameters are more efficient.

Quantities of Interest

- All quantities of interest are for marginal means rather than joint means.
- The method of bootstrapping generally should not be used in GEE models. If you must bootstrap, bootstrapping should be done within clusters, which is not currently supported in Zelig. For conditional prediction models, data should be matched within clusters.
- The expected values (qi$ev) for the GEE gamma model is the mean:

$$E(Y) = \lambda_c = \frac{1}{x_c\beta},$$

given draws of β from its sampling distribution, where x_c is a vector of values, one for each independent variable, chosen by the user.

- The first difference (qi$fd) for the GEE gamma model is defined as

$$\text{FD} = \Pr(Y = 1 \mid x_1) - \Pr(Y = 1 \mid x).$$

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

$$\frac{1}{\sum_{i=1}^{n}\sum_{t=1}^{T} tr_{it}} \sum_{i:tr_{it}=1}^{n} \sum_{t:tr_{it}=1}^{T} \left\{ Y_{it}(tr_{it}=1) - E[Y_{it}(tr_{it}=0)] \right\},$$

where tr_{it} is a binary explanatory variable defining the treatment ($tr_{it} = 1$) and control ($tr_{it} = 0$) groups. Variation in the simulations are due to uncertainty in simulating $E[Y_{it}(tr_{it} = 0)]$, the counterfactual expected value of Y_{it} for observations in the treatment group, under the assumption that everything stays the same except that the treatment indicator is switched to $tr_{it} = 0$.

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 = "gamma.gee", id, data)
```

then you may see 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: parameter estimates for the explanatory variables.
  - residuals: the working residuals in the final iteration of the fit.
  - fitted.values: the vector of fitted values for the systemic component.
  - linear.predictors: the vector of $x_{it}\beta$
  - max.id: the size of the largest cluster.
- From summary(z.out), you may extract:
  - coefficients: the parameter estimates with their associated standard errors, p-values, and z-statistics.
  - working.correlation: the "working" correlation matrix
- From the sim() output object s.out, you may extract quantities of interest arranged as matrices indexed by simulation $\times$ x-observation (for more than one x-observation). Available quantities are:
  - qi$ev: the simulated expected values for the specified values of x.
  - qi$fd: the simulated first difference in the expected probabilities for the values specified in x and x1.
  - qi$att.ev: the simulated average expected treatment effect for the treated from conditional prediction models.

See also

The gee function is part of the geepack package by Søren Højsgaard, Ulrich Halekoh and Jun Yan. Advanced users may wish to refer to `help(geepack)` and `help(family)`.

CHAPTER

THREE

FREQUENTLY ASKED QUESTIONS

If you find a bug, or cannot figure something out after reading through the FAQs below, please send your question to the Zelig listserv at: https://groups.google.com/forum/#!forum/zelig-statistical-software. Please explain exactly what you did and include the full error message, including the traceback(). You should get an answer from the developers or another user in short order.

3.1 Why can't I install Zelig?

We recommend that you first check your internet connection, as you must be connected to install packages. In addition, there are a few platform-specific reasons why you may be having installation problems:

- **On Windows**: If you are using the very latest version of R, you may not be able to install Zelig until we update Zelig to work with this latest release. Currently Zelig 5.0-1 is compatible with R (>= 3.0.2). If you wish to install Zelig in the interim, install the appropriate version of R and try to reinstall Zelig.
- **On Mac or Linux systems**: If you get the following warning message at the end of your installation:

  ```
  > Installation of package VGAM had non-zero exit status in ...
  ```

  this means that you were not able to install VGAM properly. Make sure that you have the g77 Fortran compiler. For Intel Macs, download the Apple developer tools. After installation, try to install Zelig again.

If neither solution works, feel free email the Zelig mailing list directly at: https://groups.google.com/forum/#!forum/zelig-statistical-software.

3.2 Why can't I install R?

If you have problems installing R, you should search the internet for the R help mailing list, check out technical Q & A forums (e.g., StackOverflow), or email the Zelig mailing list directly at: https://groups.google.com/forum/#!forum/zelig-statistical-software.

3.3 Why can't I load data?

It is likely that the reason you are unable to load data because you have not specified the correct working directory (e.g., the location of the data you are trying to load). You should specify you working directory use the `setwd()` function in which you will include the the file path to your working director. For example, if I wanted to load a file that is my *Documents* folder, I must first:

```
> setwd("path/to/Documents")
```

File paths can be found by right clicking the working directory folder in any file browser and clicking "Get Info" (on Mac) or "Properties" (on Windows). Black-slashes (\) in file paths copied from the "Properties" link on Windows machines must be replace with forward-slashes (/). For example, the Windows path: `C:\Program Files\R`, would be typed as `C:/Program Files/R`.

3.4 R is neat. How can I find out more?

R is a collective project with contributors from all over the world. Their website (http://www.r-project.org) has more information on the R project, R packages, conferences, and other learning material.

CHAPTER

FOUR

ABOUT ZELIG

Zelig is an open-source project developed and maintained by the Data Science group at Harvard's Institute for Quantitative Social Science. It was conceived and created by Kosuke Imai, Gary King, and Olivia Lau in 2007. It is named for Leonard Zelig, a fictional character in a Woody Allen movie who takes on the personality of anyone around him, and thus fits into any situation. Likewise, Zelig software easily adapts to any statistical model and similarly fits into any situation.

Zelig leverages (R) code from many researchers and is designed to allow anyone to contribute their methods to it. Hence, we often refer to Zelig as "everyone's statistical software" and our aim is to make it, as well as the models it wraps, as accessible as possible. As such, it comes with self-contained documentation that minimizes startup costs, automates model summaries and graphics, and bridges existing R implementations through an intelligible call structure.

Contact: For questions, please join the Zelig mailing list.

Principal Investigator: Gary King

Project Team Leads: James Honaker, Christine Choirat, Muhammed Y. Idris

Original Authors: Kosuke Imai, Gary King, Olivia Lau

Contributors: Christine Choirat, Matt Owen, Justin Grimmer, Jason Wittenberg, Badri Narayan Bhaskar, Skyler J. Cranmer, Ben Goodrich, Ying Lu, Patrick Lam, Nicholas Carnes, Alexander D'Amour, Delia Bailey, Ferdinand Alimadhi, Elena Villalon

To Cite Zelig, please reference these two sources:

> Kosuke Imai, Gary King, and Olivia Lau. 2007. "Zelig: Everyone's Statistical Software," http://zeligproject.org.
>
> Imai, Kosuke, Gary King, and Olivia Lau. 2008. "Toward A Common Framework for Statistical Analysis and Development." Journal of Computational and Graphical Statistics, Vol. 17, No. 4 (December), pp. 892-913, http://j.mp/msE15c.

CHAPTER

FIVE

LICENSE: GPL-2 | GPL-3 [EXPANDED FROM: GPL (>= 2)]

5.1 Technical Vision

Zelig is a framework for interfacing a wide range of statistical models and analytic methods in a common and simple way. Above and beyond estimation, Zelig adds considerable infrastructure to existing heterogeneous R implementations by translating hard-to-interpret coefficients into quantities of interest (e.g., expected and predicted values) through a simple call structure. This includes many specific methods, based on likelihood, frequentist, Bayesian, robust Bayesian and nonparametric theories of inference. Developers are encouraged to add their R packages to the Zelig toolkit by writing a few simple bridge functions.

Additional features include:

- Dealing with missing data by combining multiply imputed datasets
- Automating statistical bootstrapping
- Improving parametric procedures by leveraging nonparametric matching methods
- Evaluating counterfactuals
- Allowing conditional population and super population inferences
- Automating the creation of replication data files

5.2 Release Notes

v 5.0-1

This release provides a set of core models, while simplifying the model wrapping process, and solving architectural problems by completely rewriting into R's Reference Classes for a fully object-oriented architecture.

Inheritance Tree