IPEGLIM User's Guide
- Health Quality Council -

Chel Hee Lee
The Collaborative Biostatistics PhD Program
University of Saskatchewan

Mikelis Bickis
Department of Mathematics and Statistics
University of Saskatchewan

September 20, 2013

Abstract

This vignettes and related materials are designed for the staffs in the Saskatchewan Health Quality Council who consider the problem of estimating a number of missed cases on some identification process in a closed population with a certain condition. This user's manual will guide the staffs how to use the package `ipeglim` based on our proposed methodology called an `imprecise inferential framework`. Implementation of several existing statistical models is also included. All essentials on the use of this package are presented for data analysis and simulation studies in the forms of input arguments of given functions and their outputs.

Note: This vignettes includes auto-generated outputs produced from the `R` code chunks inserted in this document; thus, numerical results and its graphs will be changed whenever the new is provided.

Contents

1 Brief Description of Research Objective

Background: Administrative health databases are considered as a source of an epidemiological surveillance system for knowing the burden of a disease in a population of interest. However, this system is not free from the problem of underreporting (i.e., a number of reported cases is smaller than a number of actual cases) because of various reasons. Since the size of true cases cannot be known, questions of "how useful our surveillance system is?" or "Does the count include all records that should be included?" cannot be answered.

Quality Measurement – Completeness: Consider a hypothetical population with a certain condition of size N and assume that there is some identification process (or reporting system) for health researchers to know who is with that condition in the form of Table 1. If this reporting system has a maximal performance (or capacity) that captures all cases occurred in this population, the expectation exists that all cases will be reported without any missed cases. In this sense, a measurement of `completeness` is defined as a proportion of the number n of reported cases to the expected number $\hat{N}$ of all cases so that a degree of usefulness of this reporting system can be measured. Hence, the objective of this research is to estimate N by estimating the size n_0 of missed cases.

Table 1: Data Structure (Y may be a binary indicator or a frequency of some conditions)

PID	Source 1	Source 2	Observed
P00001	$Y_{1,1}$	$Y_{2,1}$	Yes
P00002	$Y_{1,2}$	$Y_{2,2}$	Yes
P00003	$Y_{1,3}$	$Y_{2,3}$	Yes
⋮	⋮	⋮	⋮
P20332	$Y_{1,n}$	$Y_{2,n}$	Yes
Pxxxxx	$Y_{1,n+1}$	$Y_{2,n+1}$	No
Pxxxxx	$Y_{1,n+2}$	$Y_{2,n+2}$	No
⋮	⋮	⋮	No
Pxxxxx	$Y_{1,N}$	$Y_{2,N}$	No

Existing Statistical Methodology: Capture-Recapture (CR) method seems to be the best existing efforts for achieving this research objective. Details of model specification, its assumptions, advantages and disadvantages, simulation results under various conditions, and suggestions are written in the sections of Introduction and Data Analysis of Lee (2013).

Imprecise Inferential Framework One approach to model an uncertainty of some event of our interest in a mathematical sense is to employ an `imprecise prior` (i.e., a set of probability distributions that represent a prior knowledge) for leading a reasonable course

of decision making. `Imprecise inferential framework` is presented for such a statistical reasoning in Lee (2013).

NOTE: More contents will be added by feedback or request.

2 Installation and Configuration

Installation of the package `ipeglim` can be done by typing the following command:

```
R> install.packages("ipeglim", dependencies=TRUE)
```

For the actual use, the package `ipeglim` should be attached to the library on your system:

```
R> library(ipeglim)
```

When the installation is successfully done, basic information about the package is retrieved by:

```
R> packageDescription("ipeglim")
```

```
Package: ipeglim
Version: 0.2.4
Date: 2013-09-17
Title: Imprecise Inferential Framework on Statistical
      Reasoning
Authors@R: c(person(given="Chel Hee", family="Lee",
      role=c("aut","cre", "cph", "trl"),
      email="gnustats@gmail.com"),
      person(given="Mikelis", family="Bickis",
      role=c("aut", "ths"),
      email="bickis@snoopy.usask.ca"))
Author: Chel Hee Lee and Mikelis Bickis
Maintainer: Chel Hee Lee <gnustats@gmail.com>
Depends: R (>= 2.7.1),
Suggests: mvtnorm, bindata, MASS, mgcv, lattice, sn,
      grid
Description: Imprecise inferential framework on
      statistical reasoning with count data based on
      the Walley's imprecise probability theory.
Type: Package
License: GPL (>= 2)
LazyData: true
URL: http://r-forge.r-project.org/projects/ipeglim/
BugReports:
      http://r-forge.r-project.org/projects/ipeglim/
Collate: 'a90logit.R' 'addOnBoundaries.R' .....
Packaged: 2013-09-19 19:58:54 UTC; gnustats
Built: R 3.0.1; ; 2013-09-19 19:58:56 UTC; unix

-- File: /home/gnustats/R/x86_64-pc-linux-gnu-library/3.0/ipeglim/Meta/package.rds
```

You are also to see the same list of functions as below:

```
R> ls("package:ipeglim")
```

```
 [1] "a90logit"                "addOnBoundaries"
 [3] "bayesPoisLogGammaIS"     "bayesPoisLogGammaLA"
 [5] "bayesPoisLogGammaMH"     "bayesPoisLogGammaNA"
 [7] "bayesPoisNormalIS"       "bayesPoisNormalLA"
 [9] "bayesPoisNormalMH"       "bayesPoisNormalNA"
[11] "bayesPoisXregIS"         "bayesPoisXregLA"
[13] "bayesPoisXregMH"         "cv"
[15] "dztpois"                 "IINEE"
[17] "impose"                  "model"
[19] "print.summary.a90logit" "print.summary.ztpr"
[21] "pztpois"                 "setGrid"
[23] "simulateYX"              "simulateYX2"
[25] "skewness"                "summary.a90logit"
[27] "summary.ztpr"            "ztpr"
```

In order to cite this package,

```
R> citation(package="ipeglim")


To cite package 'ipeglim' in publications use:

  Chel Hee Lee and Mikelis Bickis (2013). ipeglim:
  Imprecise Inferential Framework on Statistical
  Reasoning. R package version 0.2.4.
  http://r-forge.r-project.org/projects/ipeglim/

A BibTeX entry for LaTeX users is

  @Manual{,
    title = {ipeglim: Imprecise Inferential Framework on Statistical Reasoning},
    author = {Chel Hee Lee and Mikelis Bickis},
    year = {2013},
    note = {R package version 0.2.4},
    url = {http://r-forge.r-project.org/projects/ipeglim/},
  }
```

The imprecise inferential framework is performed by the use of three functions: `model`, `impose`, and `update`. Input arguments, output values, purposes, detailed methodological description, limitations, special notes on the use, references, TODO list, and FIXME items about these functions are provided in the HTML format on a default web-browser installed on your operating system as shown in the Figure 1. For example, the help file about `model` is invoked by the following command:

```
R> ?model
```

Please try to type `impose` and `update`.

update {ipeglim} R Documentation

Applying Bayes Rule to Imprecise Prior

Description

`update` is used to update an imprecise prior `M0` by the Bayes' rule with a given `formula` when `data` is newly taken.

Usage

```
update.imprecise(obj, method = c("MH", "IS", "LA", "NA"),
  priorType = c("lgamma", "normal"), ...)
```

Arguments

`obj`	an optional object, vector, matrix, or list containing a set of hyper-parameters that characterize an imprecise prior. The details of hyperparameter spacification are given under 'Details'.
`method`	the method to be used for a numerical approximation in fitting model. See 'Details'.
`priorType`	the choice of prior distribution for a model without covariates.
`...`	other arguments

Details

Note that arguments after ... must be matched exactly.

Method `"MH"` is an implementation of Metropolis-Hastings algorithm described in the study of Author (XXXX). Lengths of Markov chain `nchains` and burn-in period are set as `2e3` and `5e2` by default. A `p`-dimensional multivariate normal distribution with mean `jDistMean` and variance-covariance matrix `jDistVcov` is used for generating a candidate density. Initial values of `jDistMean` and `jDistVcov` are set as the estimated parameters and its variance-covariance fitted by `glm` with `log`-link function.

Method `"LA"` is an implementation of Laplace Approximation described in the study of Tierney (1986). Hessian and mode of a log-posterior are found by `optim` with `BFGS` which is a quasi-Newton method. `iniParams` are initial values for the parameters of log-posterior to be optimized over.

Method `"IS"` is an implementation of Importance Sampling described in the book of Author (XXXX). A sample of size `1e3` is used.

Method `"NA"` is an analytic solution (if it exists).

By default, `update.m0` uses the numerical method `"LA"`. Methods are applied to each model.

Value

An object of class `mlclass` with components including

Figure 1: HTML-based HELP page

model package:ipeglim R Documentation

Sampling Model Specification for Imprecise Inferential Framework

Description:

‘model’ is used to describe a sampling model by giving a symbolic description of the linear predictor in ‘formula’ to be fitted to the ‘data’ and a description of the sampling distribution in ‘dist’.

Usage:

```
model(formula, data, dist, ztrunc = FALSE, ...)
```

Arguments:

formula:	*an object of class "formula": a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.*
data:	*an optional data frame, list or environment containing variables to be used in the model. If not found in ‘data’, variables are taken from ‘environment{formula}’.*
dist:	
ztrunc:	*logical, Is a sampling model truncated at zero?*

```
     ...: other arguments

Details:

     Arguments after ... must be matched exactly to those in the
     environment.

        • 'y~0' models without predictors

        • 'y~1' models with an intercept but no preditors

        • 'y~.' models with all predictors in the 'data'

Value:

     An object of class 'imprecise' with components including

    Xreg: the logical value determined by 'formula' and 'data', Is the
          model a regression?

  ztrunc: the logical value supplied

       y: the model reponse used

       X: the model matrix used

 formula: the formula supplied

    init: the list of regression coefficients and its
          variance-covariance matrix which are fitted by 'glm'

TODO:

        • It seems to be reasonable to get 'init' by the use of the
          zero-truncated Poisson regression model when 'ztrunc=TRUE'
          rather than the simple use of 'glm'.

FIXME:

        • No bugs are found yet.

Note:

     All details are identical to those of 'formula' in 'glm'.

Author(s):

     Chel Hee Lee <<email: gnustats@gmail.com>>

References:

     Walley (1991)

See Also:

     'glm', 'formula', 'bayesPoisLogGamma'

Examples:

     # Do not run
```

In order to make sure input argumnets of a function which are intended to use, the command `args` is helpful:

```
R> args(model)


function (formula, data, dist, ztrunc = FALSE, ...)
NULL
```

3 Learn Basic Use (Generating Random Variates)

Throughout this section, basic use of this package will be exercised by several examples on the generation of random variates which of characteristics are close enough to those of the actual data compiled from the administrative health databases in Saskatchewan. Following three questions are mainly examined during a simulation study before applying this package to the actual data analysis:

- Are all results reproducible under a certain controlled condition?
- Are values of estimated parameters sufficiently close enough to those which were originally used on generating random variates under various simulation conditions?
- Is a computing time considerably fast enough in practice?

Example 3.1. Consider a hypothetical situation where a population is closed and its size is N. We also assume that only n subjects are captured by some identification process which of system is $\log(\mu_i) = \beta_0 + \beta_1 x_1$, where $x_1 \sim N(0, 1)$ and $\beta = (-1.5, 0.5)^T$. When random variates are generated, it will be structured as shown in the Table 1. **Note:** The values of regression parameters gives a similar characteristics of the actual data.

```
R> param <- c(-1.5, 0.5)   # (i.e., beta)
R> N <- 1e3                # (i.e., a population size)
R> TRUNCATED <- TRUE       # it could be FALSE.
```

A predictor matrix $X = [x_0, x_1]$ is structured by the use of correlation matrix `XR`, vectors of means of x_0 and x_1, and its standard deviations:

```
R> XR <- diag(2)           # Correlation between x0 and x1
R> Xsd <- c(0,1)           # Standard deviations of x0 and x1
R> Xmean <- c(1,0)         # Means of x0 and x1
```

To see the resulted structure of `X`,

```
R> Xmat <- diag(Xsd) %*% XR %*% t(diag(Xsd))
R> Xmat
```

```
     [,1] [,2]
[1,]    0    0
[2,]    0    1
```

The function `simulateYX()` is used to generating random variates with this structured `X`.

```
R> args(simulateYX)
```

```
function (N, param, Xstr = list(type = c("mvnorm", "mvbion",
    "mixed", "multinomial"), mean, R, sd, Xmat), shape, ztrunc = FALSE,
    Xreg = FALSE, seed = NULL, ...)
NULL
```

```
R> mydata <- simulateYX(N=N, Xreg=TRUE, param=param,
+                 Xstr=list(type="mvnorm", mean=Xmean, R=XR, sd=Xsd),
+                 link="log", ztrunc=TRUE, seed=NULL)$yX
R> head(mydata)
```

```
   y          x1
1  1  0.6649553
10 1  0.3987912
18 1 -0.5896040
29 1 -0.4618881
35 1 -0.2466177
36 1  0.9310450
```

```
R> table(mydata$y)


  1   2   3   5
196  25   3   2
```

Example 3.2. Alternative approach is to use the option `Xmat` without explicit description of correlation, means, and standard deviations of `X`.

```
R> Xmat <- cbind(x0=rnorm(N,1,0), x1=rnorm(N,0,1))
R> tmp <- simulateYX(N=N, Xreg=TRUE, param=param,
+                    Xstr=list(type="mixed", Xmat=Xmat), link="log",
+                    ztrunc=TRUNCATED, seed=NULL)
```

Note that `simulateYX` returns several object with this generation process:

```
R> names(tmp)


 [1] "N"      "yX"     "y"      "X"      "Xstr"   "ztrunc"
 [7] "Xreg"   "seed"   "n"      "mu"
```

Details about the returned values are noted on the help page(`?simulateYX`). With this simulated data, the rate of missing cases is computed by

```
R> n.missed <- tmp$N - tmp$n
R> n.missed


[1] 818
```

Example 3.3. Now, consider another system of $\log(\mu_i) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$, where $x_1 \sim N(0,1)$, $x_2 \sim N(0,1)$ and $\beta = (-1.5, 0.5, -0.5)^T$.

```
R> Xmat <- cbind(x0=rnorm(N,1,0), x1=rnorm(N,0,1), x1=rnorm(N,0,1))
R> tmp <- simulateYX(N=N, Xreg=TRUE, param=c(-1.5, 0.5, -0.5),
+                    Xstr=list(type="mixed", Xmat=Xmat), link="log",
+                    ztrunc=TRUNCATED, seed=NULL)
R> mydata3 <- tmp$yX
R> head(mydata3)
```

```
   y           x1          x2
2  1 -1.186532848  0.01020061
6  1 -1.726442233 -1.34034384
9  2  0.382808892  0.20843416
16 2  0.478243748 -1.08655302
21 1 -0.002173377  0.35396581
30 1 -0.412249334 -0.54454641
```

```
R> table(mydata3$y)
```

```
  1   2   3   4
187  27   4   2
```

Example 3.4. One may consider a system of $\log(\mu_i) = b_1 x_1 + b_2 x_2$, where $x_1 \sim Bernoulli(0.5)$, $x_2 \sim Bernoulli(0.3)$, and $\beta = (-1.5, 1.0)^T$.

```
R> Xmat <- cbind(x1=rbinom(N, size=1, prob=0.5), x2=rbinom(N, size=1, prob=0.3))
R> tmp <- simulateYX(N=N, Xreg=TRUE, param=c(-1.5, 1.0),
+                    Xstr=list(type="mixed", Xmat=Xmat), link="log",
+                    ztrunc=TRUNCATED, seed=NULL)
R> mydata4 <- tmp$yX
R> head(mydata4)
```

```
   y x1
3  3  0
7  1  0
9  1  0
11 1  0
13 5  1
14 5  1
```

```
R> table(mydata4$y)
```

```
  1   2   3   4   5   6   7   8
250 115  74  28  12   7   3   1
```

Example 3.5. You may consider a case where types of predictors are mixed such as $\log(\mu_i) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$, where $x_1 \sim N(0, 1)$, $x_2 \sim Bernoulli(0.4)$, and $x_3 \sim Multinomial(0.3, 0.2, 0.5)$ with coefficients $beta = (-1.5, 1.0, \beta_3)$. β_3 can be set as $\beta_3 = (0.5, 0.2, -0.5)$ when a dummy coding is performed with x_3.

```
R> Xmat <- cbind(1, rnorm(N,0,1), rbinom(N,1,0.4),
+                t(rmultinom(N, size=1, prob=c(0.3, 0.2, 0.5))))
R> tmp <- simulateYX(N=N, Xreg=TRUE, param=c(-1.5, 1.0, -0.5, 0.5, 0.2, -0.5),
+                    Xstr=list(type="mixed", Xmat=Xmat), log="log",
+ ztrunc=TRUNCATED)
R> mydata5 <- tmp$yX
R> head(mydata5)
```

```
   y        x1 x2 x3 x4 x5
7  1 0.7821571  0  1  0  0
8  1 0.1633791  0  1  0  0
13 2 1.3544918  1  0  1  0
16 1 0.6832420  0  1  0  0
17 1 0.4496003  0  0  1  0
19 1 1.6474607  1  0  0  1
```

```
R> table(mydata5$y)
```

```
  1   2   3   4   5   6   7
173  44  14   1   1   1   1
```

Example 3.6. Generation of random variates from the model without predictors is also possible. In this case, the `Xstr` agrument is not neccessary on the use of `simulateYX`.

```
R> mydata0 <- simulateYX(N=N, Xreg=FALSE, param=1, ztrunc=TRUNCATED, link="log")$y
R> head(mydata0)
```

```
[1] 3 1 1 3 1 2
```

```
R> table(mydata0)
```

```
mydata0
  1   2   3   4   5   6
370 174  57   7   2   2
```

4 Zero-Truncated Poisson and Negative Binomial Regression Models

Requirements:

- Y_i in the Table 1 is a frequency data of interest at the individual level which is compiled from a single data source;
- Zero-truncated Negative Binomial regression model is an alternative of zero-truncated Poisson regression model when an over-dispersion is highly suspected;
- Useful references are van˜der Heijden et˜al. (2003) and Cruyff and van˜der Heijden (2008).

Example 4.1. Consider a mechanism $\log(\mu_i) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$, where x_1 and x_2 are continuous and discrete types of variables, respectively, with $\beta = (-1.0, 0.5, -0.5)$.

Zero-truncated Poisson regression model is demonstrated first.

```
R> rm(list=ls())
R> N <- 1e3
R> b <- c(-1.0, 1.0, -0.5)
R> Xmat <- cbind(rnorm(N,1,0), rnorm(N,0,1), rbinom(N, size=1, p=0.5))
R> Dt.poisson <- simulateYX(N=N, Xreg=TRUE, Xstr=list(type="mixed", Xmat=Xmat),
+                           ztrunc=TRUE, param=b)$yX
R> table(Dt.poisson$y)


  1   2   3   4   5   6   9
224  66  22   5   3   2   1


R> head(Dt.poisson)


  y         x1 x2
1 1  0.4569557  1
2 1  0.2707489  1
3 2  1.3581471  0
4 1 -1.5605277  1
9 1  1.0620594  1
12 1 -0.6312651  1


R> # Check a number of missed data
R> (N - nrow(Dt.poisson))/N


[1] 0.677


R> # Fit a model to the data
R> fit <- ztpr(formula=y˜x1+x2, data=Dt.poisson, dist="poisson", ztrunc=TRUE)
R> summary(fit)
```

```
The model is  successfully converged
Optimization method BFGS is used
Number of iterations in optimization is  40

Coefficients for zero-truncated  Poisson  model with log link

             Estimate        SE z-score  Pr(>|z|)
(Intercept) -1.110520  0.167764 -6.6195 3.604e-11 ***
x1           0.977437  0.085944 11.3729 < 2.2e-16 ***
x2          -0.320121  0.165339 -1.9361   0.05285 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log-likelihood =  -240.7756  on  3
AIC =  487.5512
```

Now, zero-truncated Negative Binomial regression model is illustrated.

```
R> Dt.nbinom <- simulateYX(N=N, Xreg=TRUE, Xstr=list(type="mixed", Xmat=Xmat),
+                          ztrunc=TRUE, param=b, shape=0.5)$yX
R> table(Dt.nbinom$y)

  1   2   3   4   5   6   7   8  10  11  12  13  18
143  41  15  15  10   2   2   2   1   2   1   1   1

R> head(Dt.nbinom)

   y        x1 x2
20 1 0.8267912  1
26 1 0.5680219  0
27 3 1.1780081  1
29 1 1.0188100  0
30 3 1.9354110  1
32 1 1.8924609  1
```

Random variates with a negative binomial distribution has a longer tail rather than the one with a Poisson distribution.

```
R> # Check a number of missed data
R> (N - nrow(Dt.nbinom))/N

[1] 0.764

R> # fit a model with the data
R> fit <- ztpr(formula=y~x1+x2, data=Dt.nbinom, dist="nbinom", ztrunc=TRUE)
R> summary(fit)

The model is  successfully converged
Optimization method BFGS is used
Number of iterations in optimization is  39

Coefficients for zero-truncated  Negative Binomial  model with log link

            Estimate       SE z-score  Pr(>|z|)
(Intercept) -1.32965  0.61642 -2.1570    0.0310 *
x1           0.95819  0.14024  6.8327 8.335e-12 ***
x2          -0.27311  0.24087 -1.1339    0.2569
log(shape)  -1.14261  0.84146 -1.3579    0.1745
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log-likelihood =  -301.6745  on  4
AIC =  611.349
```

5 Imprecise Zero-Truncated Poisson Regression Model

The imprecise inferential framework is demonstrated in this section.

Example 5.1. Consider a system of $\log(\mu_i) = -1.0 + 1.5x_1$.

```
R> rm(list=ls())
R> N <- 1e3
R> Xmat <- cbind(x0=rnorm(N,1,0), x1=rnorm(N,0,1))
R> Dt <- simulateYX(N=N, Xreg=TRUE, param=c(-1.0, 1.5),
+                   Xstr=list(type="mixed", Xmat=Xmat), link="log",
+                   ztrunc=TRUE, seed=NULL)$yX
R> table(Dt$y)


  1   2   3   4   5   6   7   8   9  10  12  13  14  15  16
201  80  32  22  13   7   8   4   6   2   2   1   4   2   1
 17  18  23  36  51  59
  1   1   1   1   1   1


R> xi <- as.vector(as.matrix(Dt)[5,])
R> xi    # Values of fifth observation


[1] 1.0000000 0.7720286
```

Imprecise inferential framework is performed in a sequence of

- `model` for representing a mathematical relation between zero-truncated `y` and its corresponding predictors `X` organized in the data frame `Dt`:

```
R> mfit <- model(formula=y~x1, data=Dt, ztrunc=TRUE)
```

- `impose` for modelling uncertainty about the model parameters with a set of linear inequality constraints:

```
R> b <- ztpr(formula=y~x1, data=Dt, ztrunc=TRUE, dist="poisson")$cfs
R> cmfit <- impose(obj=mfit, circle=list(x0=b[1], y0=b[2], r=1,len=15))
```

  where `b` is a vector of estimated coefficients with the existing zero-truncated Poisson regression model since these estimates are the best information in our hand at this moment. Ambiguity of model parameters is expressed by a circle of radius `r` from the central point `b`.

- `update` for incorporting the data with prior strength `B` and predicting `y` given values of `xi`,

```
R> B <- diag(c(0.001, 0.001))
R> results <- update(obj=cmfit, method="LA", B=B, xi=xi)
```

- `summary` for creating human-readable results.

```
R> output <- summary(results)
R> output

Zero-truncated Poisson regression model is fitted
with an imprecise  prior
Laplace approximation is used for numerical approximation

Estimates of expected regression parameters
       (Intercept)     x1
xtm.1      -1.81761 1.7525
xtm.2      -1.63412 1.5909
xtm.3      -1.31868 1.3981
xtm.4      -0.94832 1.2249
xtm.5      -0.59138 1.1162
xtm.6      -0.33864 1.0810
xtm.7      -0.22881 1.1251
xtm.8      -0.27313 1.2303
xtm.9      -0.45969 1.3751
xtm.10     -0.75451 1.5351
xtm.11     -1.10406 1.6864
xtm.12     -1.44485 1.7991
xtm.13     -1.71271 1.8632
xtm.14     -1.84838 1.8468
xtm.15     -1.81761 1.7525

First five estimates are printed
Please type following commands for further statistics:
'obj$est' -> all estimates
'obj$imprecision' -> degree of imprecision
```

Please see details of these functions by typing `?model`, `?impose`, `?update`, and `?summary`.

Visualized `output` is shown in the Figure 2.

```
R> plot(output)
```

The circle with blue colour and the ellipse with yellow colour represent prior ignorance and its corresponding posterior imprecision, respectively. Detailed interpretation about this figure is written in the Lee (2013).

If you are in the wrong sequence of imprecise inferential framework, error messages will guide you. For example,

```
R> summary(cmfit)
```

You will see the message as

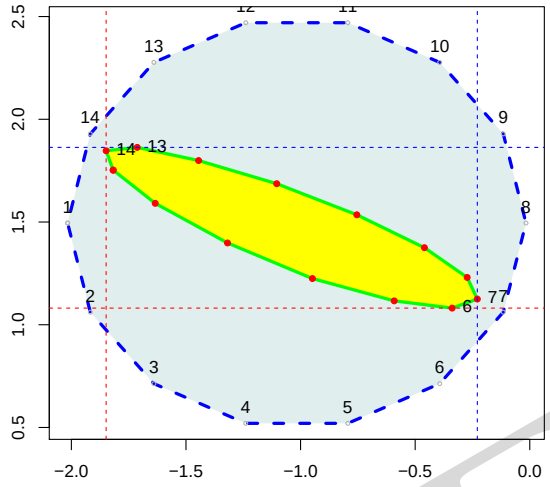

Figure 2: Prior Shrinkage by Taking Data

```
Error in summary.imprecise(cmfit) :
  Not correct order of imprecise inferential framework.
'summary' should be followed by 'update'
```

Analogously,

```
R> update(mfit)
```

```
Error in update.imprecise(mfit) :
  Not correct order of imprecise inferential framework.
'update' should be followed by 'impose'
```

Example 5.2. One may consider a situation where a log-gamma imprecise prior is imposed on the model parameter for the model without predictors.

```
R> rm(list=ls())
R> y <- simulateYX(N=10, param=1, ztrunc=TRUE, Xreg=FALSE)$y
R> table(y)
```

```
y
1 2
5 1
```

```
R> mfit <- model(formula=y~0, ztrunc=TRUE)
R> lc <- list(lhs=rbind(c(1,0), c(-1,0), c(0,1), c(0,-1)),
+             rhs=c(0.05, -10, 0.05, -10))
R> cmfit <- impose(obj=mfit, eqns=lc)
R> fitall <- update(obj=cmfit, method="LA", priorType="lgamma")
R> output <- summary(fitall)
R> output
```

```
Zero-truncated Poisson  model is fitted
with an imprecise  lgamma prior
Laplace approximation is used for numerical approximation

Estimates of expected canonical parameter

            theta exp(theta)
xtm.1 -1.6115103    0.19959
xtm.2 -2.9933370    0.05012
xtm.3  0.9100074    2.48434
xtm.4 -0.2443516    0.78321

Please type following commands for further statistics:
'obj$est' -> all estimates
'obj$imprecision' -> degree of imprecision
```

The function `impose` provides a number of options how to impose a set of linear inequality constraints for modelling a prior ignorance. For example, a box-constrained type is shown in the left panel of the Figure 3.

```
R> plot(cmfit)
```

```
R> cmfit1 <- setGrid(cmfit, len=10)
R> plot(cmfit1)
```

Two kinds of prior shrinkage with an imprecise prior (box- and polygon-constrained) are shown in the Figure 4.

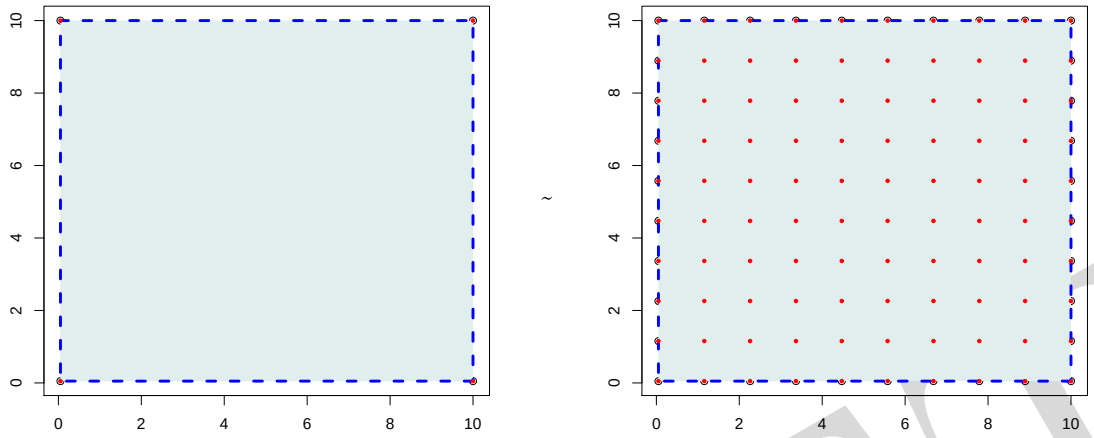

Figure 3: Box-constrained Imprecise Prior

```
R> plot(output, clevels=10)
```

```
R> fitall1 <- update(obj=cmfit1, method="LA", priorType="lgamma")
R> output1 <- summary(fitall1)
R> z <- range(output1$est)
R> plot(output1, xlim=c(-2,12), ylim=c(-2, 12),
+      zlim=c(z[1]-0.5, z[2]+0.5), clevels=10)
```

```
R> bounds <- impose(circle=list(x0=5,y0=5,r=5,len=7))
R> plot(output1, xlim=c(-2,12), ylim=c(-2,12), bnd=bounds,
+      zlim=c(z[1]-0.5, z[2]+0.5), clevels=10)
```

Example 5.3. Consider another model without predictors where a normal imprecise prior is imposed on the model parameter.

```
R> rm(list=ls())
R> y <- simulateYX(N=10, param=1, ztrunc=TRUE, Xreg=FALSE)$y
R> table(y)
```

```
y
1 2 3
3 1 1
```

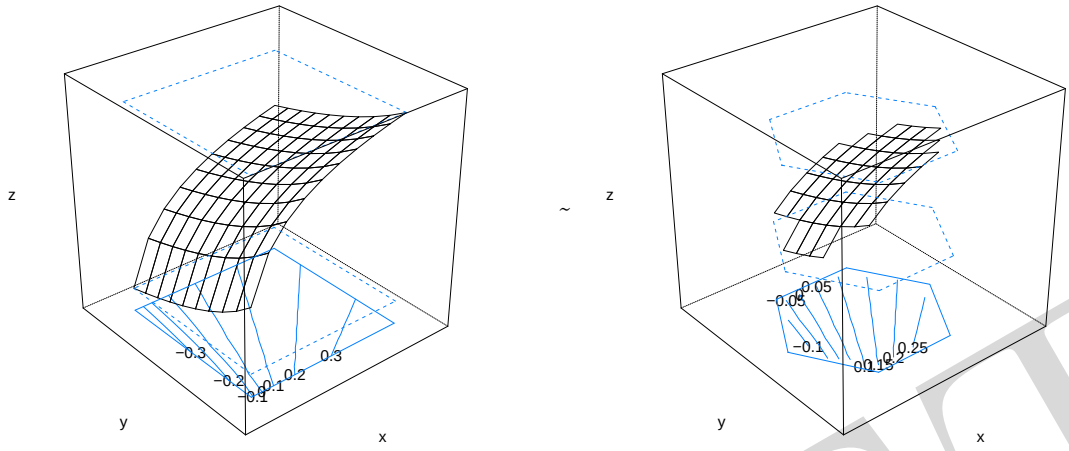

Figure 4: Surface Plot of Imprecise Posterior with Given Constraints

```
R> mfit <- model(formula=y~0, ztrunc=TRUE)
R> lc <- list(lhs=rbind(c(1,0), c(-1,0), c(0,1), c(0,-1)), rhs=c(-5,-5,0.5,-2))
R> cmfit <- impose(obj=mfit, eqns=lc)
R> cmfit <- setGrid(obj=cmfit, len=10)
R> fitall <- update(obj=cmfit, method="LA", priorType="normal")
R> output2 <- summary(fitall)
R> output2


Zero-truncated Poisson  model is fitted
with an imprecise  normal prior
Laplace approximation is used for numerical approximation

Estimates of expected canonical parameter

           theta exp(theta)
xtm.1 -3.5452157    0.02886
xtm.2 -2.5158804    0.08079
xtm.3 -1.5978118    0.20234
xtm.4 -0.8496082    0.42758
xtm.5 -0.2812892    0.75481

Please type following commands for further statistics:
'obj$est' -> all estimates
'obj$imprecision' -> degree of imprecision



R> z <- range(output2$est)
R> plot(output2, xlim=c(-6,6), ylim=c(0, 3),
+      zlim=c(z[1]-0.5, z[2]+0.5), drape=TRUE, clevels=10)
```

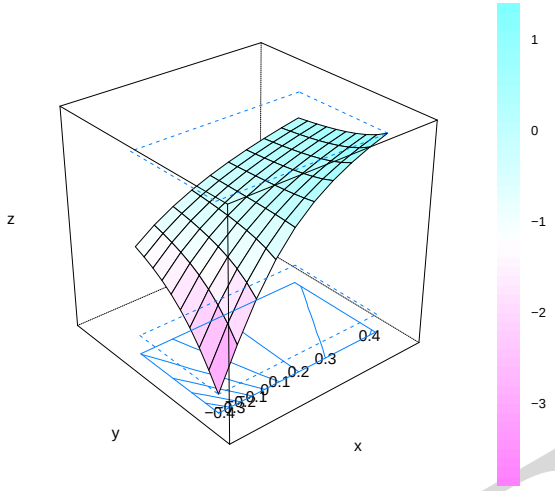

Figure 5: Surface Plot of Imprecise Posterior with Normal Imprecise Prior

6 Application I. Cholera Epidemic in India

This example is taken from the study of Böhning et˜al. (2005). Since there is no available predictors, the data is fitted to the model with an intercept.

```
R> rm(list=ls())
R> tab <- c(32, 16, 6, 1)
R> Kendrick <- as.data.frame(rep(1:length(tab), times=tab))
R> names(Kendrick) <- c("y")
R> fit <- ztpr(formula= y~1, data=Kendrick, dist="poisson")
R> summary(fit)


The model is  successfully converged
Optimization method BFGS is used
Number of iterations in optimization is  14

Coefficients for zero-truncated  Poisson  model with log link

             Estimate        SE z-score Pr(>|z|)
(Intercept) -0.028218  0.168707 -0.1673   0.8672

Log-likelihood =  -54.77768  on  1
AIC =  111.5554
```

7 Application 2. Illigal Immigrants in the Netherlands

This example is taken from the study of van˜der Heijden et˜al. (2003).

```
R> rm(list=ls())
R> data(IINEE)
R> fit <- ztpr(capture ~ ., data=IINEE, dist="poisson")
R> print(summary(fit))

The model is  successfully converged
Optimization method BFGS is used
Number of iterations in optimization is  45

Coefficients for zero-truncated  Poisson  model with log link

             Estimate        SE z-score  Pr(>|z|)
(Intercept) -2.317185  0.449371 -5.1565 2.516e-07 ***
gender       0.397373  0.163047  2.4372 0.0148029 *
age          0.974439  0.408204  2.3871 0.0169801 *
nation1     -1.674381  0.602882 -2.7773 0.0054814 **
nation2      0.190023  0.194003  0.9795 0.3273385
nation3     -0.911244  0.300968 -3.0277 0.0024641 **
nation4     -2.337257  1.013891 -2.3052 0.0211534 *
nation5     -1.092308  0.301634 -3.6213 0.0002931 ***
reason       0.010969  0.161527  0.0679 0.9458606
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Log-likelihood =  -848.4481  on  9
AIC =  1714.896
```

This result is very close enough to the result in the Table 4. of the study of van˜der Heijden et˜al. (2003).

```
R> fit0 <- summary(ztpr(capture ~ 1, data=IINEE, dist="poisson"))
R> fit1 <- summary(ztpr(capture ~ gender, data=IINEE, dist="poisson"))
R> fit2 <- summary(ztpr(capture ~ gender + age, data=IINEE, dist="poisson"))
R> fit3 <- summary(ztpr(capture ~ gender + age + nation1 + nation2 + nation3
+                       + nation4 + nation5, data=IINEE, dist="poisson"))
R> fit4 <- summary(ztpr(capture ~ ., data=IINEE, dist="poisson"))
R> fit <- list(fit0, fit1, fit2, fit3, fit4)
R> tab <- data.frame(AIC=unlist(lapply(fit, "[[", "aic")),
+                     DF=unlist(lapply(fit, "[[", "df")))
R> tab$G2 <- c(NA, diff(-(tab$AIC-2*tab$DF), lag=1))
R> tab$G2DF <- c(NA, diff(tab$DF, lag=1))
R> tab$P <- pchisq(q=tab$G2, df=tab$G2DF, lower.tail=FALSE)
R> tab$N <- unlist(lapply(fit, "[[", "N"))
R> tab$cil <- unlist(lapply(fit, "[[", "cil"))
R> tab$ciu <- unlist(lapply(fit, "[[", "ciu"))
R> rownames(tab) <- c("NULL", "G", "G+A", "G+A+N", "G+A+N+R")
R> print(tab)

             AIC DF           G2 G2DF           P
NULL    1805.904  1           NA   NA          NA
G       1798.278  2  9.626259681    1 1.918148e-03
G+A     1789.043  3 11.234558538    1 8.028820e-04
G+A+N   1712.901  8 86.142191041    5 4.336493e-17
G+A+N+R 1714.896  9  0.004567914    1 9.461149e-01
```

Again, this result is very close enough to the result in the Table 5. in the study of van˜der Heijden et˜al. (2003).

8 Log-Linear Model

Requirements:

- Y_i in the Table 1 is a binary indication of presence or absence of some condition of interest;
- At least two data sources should be linked without mismatches at the individual level.

NOTE: Imprecise inferential framework is not supported with this model.

With Two Data Sources

```
R> rm(list=ls())
R> library(ipeglim)
R> tmp <- simulateYX2(N=1e3, Ystr=list(mean=c(0.5, 0.5), sigma=diag(2)),
+ Xreg=FALSE, ztrunc=TRUE)
R> y <- tmp$y
R> head(y)


  y1 y2
1  1  1
2  1  1
4  1  0
5  1  0
6  1  1
7  1  0


R> ytab <- tmp$ytab
R> ytab


  y1 y2 freq
1  0  0   NA
2  1  0  261
3  0  1  228
4  1  1  259


R> n <- sum(ytab, na.rm=TRUE)
R> fit <- glm(formula=freq~y1+y2, data=ytab, family=poisson(link="log"))
R> summary(fit)


Call:
glm(formula = freq ~ y1 + y2, family = poisson(link = "log"),
    data = ytab)

Deviance Residuals:
[1]  0  0  0

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  5.437038   0.109902  49.472   <2e-16 ***
y1           0.127482   0.090813   1.404     0.16
y2          -0.007692   0.087706  -0.088     0.93
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance:  2.7873e+00  on 2  degrees of freedom
Residual deviance: -5.1070e-15  on 0  degrees of freedom
  (1 observation deleted due to missingness)
AIC: 28.066

Number of Fisher Scoring iterations: 2


R> n00 <- exp(fit$coef[1])
R> N <- n + n00
R> N


(Intercept)
   981.7606
```

With More Than Three Data Sources Contents for this paragraph will be added by request.

NOTES:

- The number of data sources supported by this package is three.
- Imprecise inferential framework is not supported with this model.

9 Logistic Regression Model

Requirements:

- Y_i in the Table 1 is a binary indication of presence or absence of some condition of interest;
- A number of data sources required for this model is only two.

NOTE: Imprecise inferential framework is not supported with this model.

Example 9.1. Consider the case where

$$\log\left(\frac{p_{1i}}{1-p_{1i}}\right) = 0.5 + 0.8x_i, \quad (1)$$

$$\log\left(\frac{p_{2i}}{1-p_{2i}}\right) = 1.5 + 0.4x_i \quad (2)$$

(Alho, 1990, p.˜630).

```
R> rm(list=ls()) ## 100, 300, 1000
R> set.seed(1)
R> N <- 1e3
R> beta1 <- c(0.5, 0.8)
R> beta2 <- c(1.5, 0.4)
R> Xmat <- cbind(rnorm(N,1,0), rnorm(N,0,1))
R> Dt <- simulateYX2(N=N, param1=beta1, param2=beta2, Xstr=list(type="mixed",
+ Xmat=Xmat), Xreg=TRUE, ztrunc=TRUE)$yX
R> head(Dt)
```

```
  y1 y2         x1
1  1  0 -0.6264538
2  0  1  0.1836433
3  1  1 -0.8356286
4  1  1  1.5952808
5  1  1  0.3295078
6  1  1  0.4874291
```

```
R> fit <- a90logit(formula=cbind(y1, y2)~ x1 , data=Dt, nlists=2)
R> summary(fit)
```

```
The model is  successfully converged
Newton-Raphson method is used
Number of iterations in optimization is  6

Coefficients for Logistic Regression Model with logit link

            Estimate       SE z-score  Pr(>|z|)
(Intercept) 0.550587 0.079010  6.9686 3.202e-12 ***
x1          0.804251 0.086654  9.2812 < 2.2e-16 ***
            1.493914 0.108665 13.7478 < 2.2e-16 ***
```

```
            0.360309 0.114337  3.1513  0.001626 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log-likelihood =  -794.0573  on  4
AIC =  1596.115

N = 976.8863  (with SE= 15.23253 )
95 % CI of N = [ 947.0311 , 1006.741 ]
```

Example 9.2.

$$\log\left(\frac{p_{1i}}{1-p_{1i}}\right) = -0.5 + 0.8x_i, \tag{3}$$

$$\log\left(\frac{p_{2i}}{1-p_{2i}}\right) = -1.0 + 0.4x_i \tag{4}$$

(Alho, 1990, p˜630).

```
R> rm(list=ls()) ## 100, 300, 1000
R> N <- 1e3
R> beta1 <- c(-0.5, 0.8)
R> beta2 <- c(-1.0, 0.4)
R> Xmat <- cbind(rnorm(N,1,0), rnorm(N,0,1))
R> Dt <- simulateYX2(N=N, param1=beta1, param2=beta2, Xstr=list(type="mixed",
+ Xmat=Xmat), Xreg=TRUE, ztrunc=TRUE)$yX
R> head(Dt)

  y1 y2          x1
1  1  1  1.61970074
2  1  0 -0.05584993
3  1  1  0.69641761
4  0  1 -1.31028350
5  1  0 -0.20807859
6  1  0 -0.31278658



R> fit <- a90logit(formula=cbind(y1, y2) ~ x1 + 1 , data=Dt, nlists=2)
R> summary(fit)


The model is  successfully converged
Newton-Raphson method is used
Number of iterations in optimization is  6

Coefficients for Logistic Regression Model with logit link

            Estimate       SE z-score  Pr(>|z|)
(Intercept) -0.38565  0.13857 -2.7831 0.0053841 **
x1           0.91496  0.14283  6.4061 1.493e-10 ***
            -1.01624  0.12651 -8.0327 9.533e-16 ***
             0.38924  0.11812  3.2953 0.0009831 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log-likelihood =  -547.9902  on  4
AIC =  1103.98

N = 1004.769  (with SE= 84.75674 )
95 % CI of N = [ 838.6488 , 1170.889 ]
```

Example 9.3. Consider a situation where there are three predictors are in hand.

$$\log\left(\frac{p_{1i}}{1-p_{1i}}\right) = 0.5 + 0.8x_1 - 0.2x_2 \quad (5)$$

$$\log\left(\frac{p_{2i}}{1-p_{2i}}\right) = 1.5 + 0.4x_1 + 0.5x_2 \quad (6)$$

```
R> rm(list=ls()) ## 100, 300, 1000
R> N <- 1e3
R> beta1 <- c(0.5, 0.8, -0.2)
R> beta2 <- c(1.5, 0.4, 0.5)
R> Xmat <- cbind(rnorm(N,1,0), rnorm(N,0,1), rnorm(N,0,1))
R> Dt <- simulateYX2(N=N, param1=beta1, param2=beta2, Xstr=list(type="mixed",
+ Xmat=Xmat), Xreg=TRUE, ztrunc=TRUE)$yX
R> head(Dt)

  y1 y2         x1         x2
1  1  0  0.7645571  0.6291412
2  1  1  0.5707101 -1.6781940
3  1  1 -1.3516939  1.1797811
4  0  1 -2.0298855  1.1176545
5  1  0  0.5904787 -1.2377359
6  0  1 -1.4130700 -1.2301645


R> fit <- a90logit(formula=cbind(y1, y2)~x1+x2 , data=Dt, nlists=2)
R> summary(fit)

The model is  successfully converged
Newton-Raphson method is used
Number of iterations in optimization is  7

Coefficients for Logistic Regression Model with logit link

             Estimate        SE z-score  Pr(>|z|)
(Intercept)  0.500954  0.077142  6.4939 8.363e-11 ***
x1           0.716037  0.085002  8.4238 < 2.2e-16 ***
x2          -0.186490  0.079177 -2.3554 0.0185052 *
             1.477575  0.111434 13.2597 < 2.2e-16 ***
             0.382725  0.113043  3.3856 0.0007101 ***
             0.486217  0.108583  4.4779 7.540e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Log-likelihood =  -836.5418  on  6
AIC =  1685.084

N = 1014.328  (with SE= 15.73687 )
95 % CI of N = [ 983.4842 , 1045.172 ]
```

10 Contact Information for Maintenance

The project website is running at the http://ipeglim.r-forge.r-project.org. Bugs reports or requests for additional features of this package should be sent to gnustats@gmail.com or chl948@mail.usask.ca.

References

Alho, J.˜M. (1990). Logistic Regression in Capture-Recapture Models. Biometrics, 46(3):pp. 623–635.

Böhning, D., Dietz, E., Kuhnert, R., and Schön, D. (2005). Mixture models for capture-recapture count data. Statistical Methods & Applications, 14:29–43. 10.1007/BF02511573.

Cruyff, M. J. L.˜F. and van˜der Heijden, P. G.˜M. (2008). Point and Interval Estimation of the Population Size Using a Zero-Truncated Negative Binomial Regression Model. Biometrical Journal, 50(6):1035–1050.

Lee, C.˜H. (2013). Imprecise inferential framework on statistical reasoning with a Poisson data. PhD thesis, University of Saskatchewan.

van˜der Heijden, P.˜G., Bustami, R., Cruyff, M.˜J., Engbersen, G., and van Houwelingen, H.˜C. (2003). Point and interval estimation of the population size using the truncated Poisson regression model. Statistical Modelling, 3(4):305–322.

Walley, P. (1991). Statistical reasoning with imprecise probabilities. Monographs on statistics and applied probability. Chapman and Hall.