pba: Probabilistic Bias Analysis

Jeremy Thoms Hetzel

April 10, 2011

1 Introduction

Error in the estimation of measures of effect consists of two components: random error and systematic error. The scientific analysis of data has historically been dominated by the quantification of random error, most often with hypothesis testing, p values, and confidence intervals of the frequentist statistical tradition. Systematic error is largely ignored, or at most addressed qualitatively. While methods to quantitate systematic error have been published, implementation of these methods generally require tailoring custom programming to individual analyses. Consequently, systematic error quantification has been largely inaccessible to the scientific community.

Recently, methods to automate systematic error quantification have been developed. In a 2005 paper, Fox and colleagues describe an approach termed "probabilistic bias analysis" that quantifies systematic error from misclassification bias, selection bias, and bias from unmeasured confounding. (Int J Epidemiol. 2005 Dec;34(6):1370-6) The authors implement the methods using Excel spreadsheets and SAS code. Orsini adapted their code in the STATA package EPISENS. (http://nicolaorsini.altervista.org/stata/tutorial/e/episens.htm) Gustafson has provided R code for quantitative bias analysis in various of his publications (http://www.stat.ubc.ca/ gustaf/pubs.html), but a unified package does not exist.

Briefly, probabilistic bias analysis allows for the specification of probability distributions that approximate an expected bias. An estimate of the bias is randomly sampled from the distribution and used to calculate an adjusted effect estimate. The process is repeated in a Monte Carlo fashion, and the bias-adjusted effect estimates are summarized with the median effect estimate and a simulation interval (similar to the confidence interval). For example, in a hypothetical study, investigators estimate the association between smoking and colorectal cancer. Smoking was self-reported by the study subjects, and previous literature suggests that subjects misreport their smoking status with a sensitivity and specificity between 85probabilistic bias analysis, sensitivity and specificity of smoking misclassification can be approximated with a triangular probability distribution, with a minimum of 0.85, a mode of 0.90, and a maximum of 0.95.

The purpose of the Probabilistic Bias Analysis package is to implement and extend the probabilistic bias analysis methodologies of Fox and colleagues in a convenient R package to facilitate the quantification of systematic error by scientists. The package will initially be limited to the quantification of error

due to misclassification bias, selection bias, and unmeasured confounding by binary variables. Later, the package will be extended to allow for adjustment of categorical and continuous variables. The package aims to allow users to conveniently perform a probability bias analysis on any lm or glm object.

The project will use the resources of R-Forge to host the package's code while under development. R-Forge will facilitate development of the code and allow exposure of the package to other users interested in the development or use of systematic bias quantification.

2 Example

The following example is taken from Lash and colleagues.

```
> require(pba)
> data(LungCancerResins)
> str(LungCancerResins)

'data.frame':        1341 obs. of  3 variables:
 $ i   : int  1 2 3 4 5 6 7 8 9 10 ...
 $ case: int  1 1 1 1 1 1 1 1 1 1 ...
 $ exp : int  1 1 1 1 1 1 1 1 1 1 ...

> head(LungCancerResins)

  i case exp
1 1    1   1
2 2    1   1
3 3    1   1
4 4    1   1
5 5    1   1
6 6    1   1
```

The estimate of the effect of resin exposure on occurence of lung cancer can be modeled by logistic regression:

```
> glm1 <- glm(case ~ exp, data = LungCancerResins,
+     family = binomial())
> summary(glm1)

Call:
glm(formula = case ~ exp, family = binomial(), data = LungCancerResins)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.5681  -0.4355  -0.4355  -0.4355   2.1921

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -2.3079     0.1082 -21.340  < 2e-16 ***
exp           0.5655     0.1944   2.908  0.00364 **
---
```

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

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 893.21  on 1340  degrees of freedom
Residual deviance: 885.21  on 1339  degrees of freedom
AIC: 889.21

Number of Fisher Scoring iterations: 5
```

Accounting for random error only, the regression estimates that a true effect of resin on lung cancer. However, the model ignores any systematic error, such as misclassification of the exposure.

3 Defining bias parameters

The pba package supports specification of three forms of bias: misclassification, selection, and unmeasured confounding. Each type of bias is defined by a list of parameters. For example, misclassification bias is defined by sensitivity among cases, specificity among cases, sensitivity among non-cases, and specificity among non-cases. Parameters for the other biases are listed in ?pbaVariable. In a simple bias analysis, each parameter would take a single numeric value, which would then be used to reverse calculate the true effect estimate from the observed effect estimate. In a probabilistic bias analysis, each parameter instead takes a distribution of values. In the pba package, these distributions are defined by objects of class *pba.distr* using the `pbaDitr` function.

For example, suppose that the literature suggests that resin exposure is systematicaly misclassified with a sensitivity and specificity between 85% and 100%, regardless of exposure or outcome status. The probable values of sensitivity and specificity can be represented as a uniform distribution with a minimum value of 0.85 and a maximum value of 1.00. In pba, this distribution is defined as follows:

```
> uniform <- pbaDistr(distr = "runif", args = list(min = 0.85,
+     max = 1))
> uniform

$distr
[1] "runif"

$args
$args$min
[1] 0.85

$args$max
[1] 1
```

The `distr` argument defines the name of the function to use to generate the distribution (in this case the runif function), and the `args` argument is a list containing arguments to pass to the `distr` function. Note that the `n` argument

does not need to be specified, as it will automatically be added by the `pba` function later.

The `pbaVariable` function is used to define biases effecting a variable. Using the uniform *pba.distr* object, an object defining bias effecting the exp variable is defined as follows:

```
> exp.bias <- pbaVariable(variable = "exp", misclassification = list(se.a.distr = uniform,
+     sp.a.distr = uniform, se.b.distr = uniform,
+     sp.b.distr = uniform, se.cor = 1, sp.cor = 1))
```

The `exp.bias` object defines misclassification of the `exp` variable with sensitivity and specificity among cases and non-cases floowing a uniform distribution between 0.85 and 1.00. The `se.cor` and `sp.cor` arguments define the correlation between case and control sensitivities and specificities, respectively. Values equal to 1 describe non-differential misclassification bias, where the sensitivities (and specificities) among cases and controls will always be equal. Values less than 1 allow for increasing independence of sensitivities (and specificities) amon cases and controls, allowing for differential misclassification bias.

While the `uniform` *pba.distr* object limits the misclassification parameters to between 0.85 and 1.00, it might be unrealistic to assume an equal probability of all values between 0.85 and 1.00. Instead, a trapezoidal distribution with a minimum of 0.85, first mode of 0.90, second mode of 0.95, and maximum of 1.00 may be more realistic.

```
> trapezoid <- pbaDistr(distr = "rtrapezoid", args = list(min = 0.85,
+     mode1 = 0.9, mode2 = 0.95, max = 1))
> exp.bias <- pbaVariable(variable = "exp", misclassification = list(se.a.distr = trapezoi
+     sp.a.distr = trapezoid, se.b.distr = trapezoid,
+     sp.b.distr = trapezoid, se.cor = 1, sp.cor = 1))
```

Any random number generation function can be used to describe a *pba.distr* probability distribution object. More common examples include `rnorm`, `rbinom`, or `rpois`, and a more complete list of available distributions is available at `http://cran.r-project.org/web/views/Distributions.html`.

Having defined a *pba.variable* object `exp.bias`, the next step is to perform the probabilistic bias analysis on a `lm` or `glm` object.

```
> pba1 <- pba(model = glm1, pba.variables = exp.bias,
+     iter = 100, progress = 10)
```

In this example, the number of iteratinos performed (`iter`) is kept small in the interest of time. For a publication quality analysis, `iter` should be set much higher. The `progress` argument prints the progress of the analysis after every specified iterations.

The results of the *pba* analysis are sumarized with the `summary` method:

```
> summary(pba1)

$star
              estimate       2.5%      97.5% precision
(Intercept) -2.3078901 -2.5262508 -2.1018233 0.4244275
```

```
exp          0.5654766  0.1772723  0.9411851 0.7639128
                Pr(>|z|)
(Intercept) 4.866172e-101
exp          3.636068e-03

$hat
              estimate       2.5%     97.5% precision p
(Intercept) -2.3330408 -2.3765419 -2.257440 0.1191020 0
exp          0.7805996  0.4556199  1.172772 0.7171525 0

$hat.random
              estimate       2.5%     97.5% precision p
(Intercept) -2.3363905 -2.7856587 -2.071263 0.7143958 0
exp          0.7665723  0.3010034  1.255295 0.9542917 0

attr(,"class")
[1] "summary.pba"
```

The `coefficients.star` data frame summarizes the original coefficients of the `glm1` model, prior to adjustment for bias. The `coefficients.hat` data frame summarizes the coefficients after adjusting for bias, but before including random error. The `coefficients.hat.random` data frame summarizes the coefficients after adjusting for both bias and including random error. The `precision` column summarizes the width of the confidence or simulation limits, which in this case is the upper limit minus the lower limit.

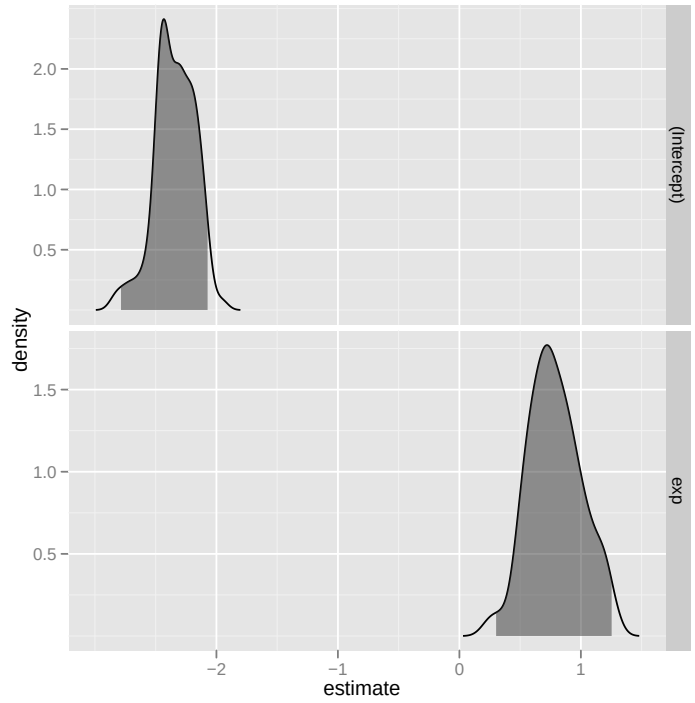
In this example, since `glm1` is a logistic regression model, it is often more intuitive to report the effect estimates as odds ratios by exponentiating the coefficients. For convenience, this can be done with the following:

```
> summary(pba1, transformation = "exp", scale = "multiplicative")

$star
              estimate       2.5%     97.5% precision
(Intercept) 0.0994709 0.07995824 0.1222334  1.528715
exp         1.7602864 1.19395620 2.5630171  2.146659
                Pr(>|z|)
(Intercept) 4.866172e-101
exp          3.636068e-03

$hat
              estimate       2.5%     97.5% precision p
(Intercept) 0.09700034 0.09287118 0.1046180  1.126485 0
exp         2.18278063 1.57715080 3.2309377  2.048591 0

$hat.random
              estimate       2.5%     97.5% precision p
(Intercept) 0.09667596 0.06168844 0.1260265  2.042952 0
exp         2.15237593 1.35121396 3.5088739  2.596831 0

attr(,"class")
[1] "summary.pba"
```

Here, the transformation specifies that coefficients should be transformed by the exponential function `exp`, however any other appropriate transformative funciton may be specified. Since the exponential function transforms the coefficients from the additive scale to the multiplicative scale, specifying `scale` as "multicplicative" assures that `precision` is correctly calculated as the upper limit divided by the lower limit, instead of the upper limit minus the lower limit.

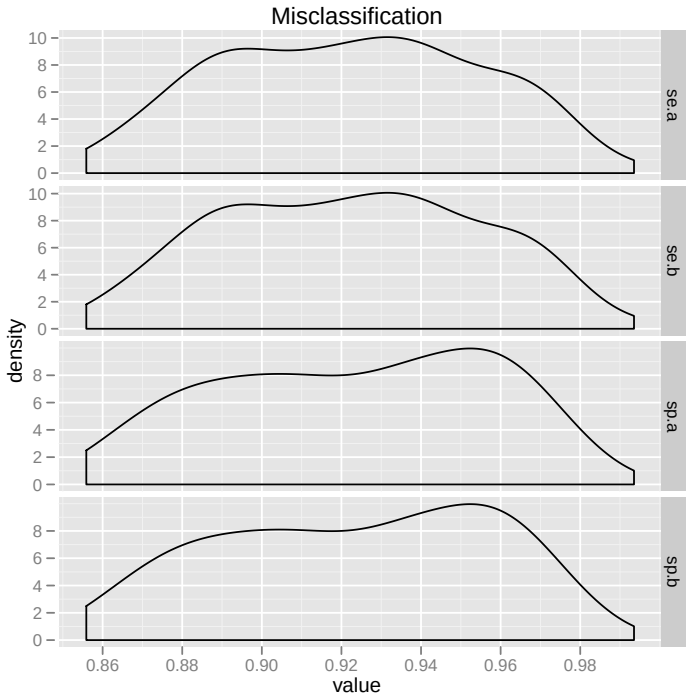
The distribution of the effect estimates may be plotted as follows:

```
> plotEstimates(pba1)
```



The distribution of the bias parameters can also be plotted:

```
> plotBias(pba1)
```


Misclassification
density
value
se.a
se.b
sp.a
sp.b
0
2
4
6
8
10
0.86
0.88
0.90
0.92
0.94
0.96
0.98