BIVpack: Bayesian methods for IV regression

Purushottam Laud, Rodney Sparapani,
Jessica Pruszynski and Robert McCulloch

November 26, 2013

1 BIVpack: Bayesian methods for IV regression

BIVpack is an R package implementing Bayesian methods for IV regression. BIVpack supports both parametric and nonparametric calculations; the parametric calculations are purely of academic interest only. Paraphrasing Imbens and Angrist (1994): Causal inference of an observational study requires the nonparametric identification of treatment effects without relying on functional form restrictions or distributional assumptions.

On the MCW Division of Biostatistics web page, there is a link to BIVpack at: [`http://www.mcw.edu/biostatistics/statisticalresources/CollaborativeSoftware.htm`]. This link will take you to R-Forge: an open platform for the development of R packages, R-related software and further projects. It is based on SVN (Collins-Sussman et al., 2011) and web technology to provide R packages, mailing lists, bug tracking, message boards/forums, site hosting, permanent file archival, full backups, and total web-based administration. The permanent BIVpack R-forge location is [`http://r-forge.r-project.org/projects/bivpack`].

2 Background

Statisticians have a long history of using specialized, interactive programming environments for data processing and statistical analysis (since modern general purpose interactive languages like Perl and Python do not readily provide the mathematical and statistical building blocks statisticians require). According to the TIOBE popularity rankings of programming languages [`http://www.tiobe.com/index.php/content/paperinfo/tpci`], R (R, 2013) is the second most popular statistical programming language.

R is an interpreted, object-oriented language and environment for statistical computing and graphics; it is a free software project falling under the GNU Public License (GPL). R is based on other GPL technologies like the GNU Compiler Collection (GCC) of C, C++ and Fortran compilers: [`http://gcc.gnu.org`]. R provides the

basis upon which over 5000 R packages have been created to perform ever more specialized purposes: [http://lib.stat.cmu.edu/R/CRAN].

BIVpack was created in this nutrient rich gene pool. BIVpack relies heavily on two R packages: Rcpp, [http://lib.stat.cmu.edu/R/CRAN/web/packages/Rcpp], and RcppEigen, [http://lib.stat.cmu.edu/R/CRAN/web/packages/RcppEigen]. Rcpp provides an interface between the relatively slow, interactive performance of object-oriented R and fast, efficient, object-oriented, C++ compiled code. RcppEigen uses Rcpp to integrate R with Eigen, [http://eigen.tuxfamily.org]: a C++ template library for linear algebra, e.g. matrices, vectors, numerical solvers, and related algorithms.

3 Parametric Models

BIVpack provides 3 functions for estimating parametric models: `nniv` for a numeric treatment and outcome; `bniv` for a binary treatment and a numeric outcome; and `bbiv` for a binary treatment and outcome. Each of these functions takes 3 arguments; `info`, `data` and `mcmc`; and returns the posterior samples of the parameters as a matrix.

`info` is a list of parameters with their initial values and prior parameter settings. `data` is a list containing an `X` matrix for the confounders (if no confounders are present, then provide a matrix with a column of zeros), a `Z` matrix for the instruments, a treatment vector `t` and an outcome vector `y`: for a binary treatment the vector should be called `tbin` and for a binary outcome `ybin`. `mcmc` is a list of the Markov chain Monte Carlo parameters: `M` for the length of the chain, `burnin` for the amount to discard from the beginning and `thin` for reducing auto-correlation by only keeping a fraction of the chain.

In BIVpack, there is an example provided in `man/BIVpackage.Rd` ; we will demonstrate these functions via excerpts from this file.

3.1 nniv

```
require(BIVpack)

N <- 10
p <- 0
q <- 1
p1 <- max(1, p)
r <- p1+q
s <- p1+r

gamma <- 0
delta <- 4
eta  <- 0
beta <- 0.5
```

```
mu   <- 0
rho  <- 0.6

mcmc <- list(M=1, burnin=0, thin=1)

info <- list(theta=list(init=c(rep(gamma, p1), rep(delta, q), rep(eta, p1)),
               prior=list(mean=rep(0., s),
                 prec=diag(0.001, s))),
             beta=list(init=beta, prior=list(mean=0., prec=0.001)),
             Tprec=list(init=solve(matrix(c(1, rho, rho, 1), 2, 2)),
               prior=list(nu=4, Psi=diag(1, 2, 2))),
             mu=list(init=c(mu, mu),
               prior=list(mean=c(0., 0.), prec=diag(0.001, 2))))

data <-  list(X=matrix(0, nrow=N, ncol=p1),
             Z=matrix(c(-0.24146164, -0.29673723, -0.27538621,
               0.41463628, 0.39023100, -0.22045922, -0.07062149,
               -0.22595298, 0.01247487, -0.14472589), nrow=N, ncol=q),
             t=c(-2.01322819, -2.04167660, -0.56128516,
               0.20783192, 0.31477076, -1.41477107, -0.38701899,
               -0.59955150, 0.01197733, -0.79804809),
             y=c(-1.9924944, -1.9345279, -1.3781082, -0.7646928,
               -0.2881649, 0.1545577, -0.6114224, -0.3703420,
               0.2320320, 0.7451867))

set.seed(42)
nniv(info, data, mcmc)
##should produce approx...
##        gamma1   delta1     eta1      beta        mu1         mu2      T11
## [1,] -17.85732 4.008851 20.01287 0.8217393 -0.3821324 -0.05601638 3.108135 -1.3
```

3.2 bniv

```
info <- list(beta=list(init=0.,
               prior=list(mean=0., prec=0.001)),
             rho=list(init=0.6),
             mu=list(init=c(0.,0.),
               prior=list(mean=c(0.,0.), prec=diag(0.001,2))),
             tau=list(init=1., prior=list(alpha0=0.1, lambda0=0.1)),
             theta=list(init=c(rep(0., p1), rep(0., q), rep(0., p1)),
               prior=list(mean=rep(0.,s), prec=diag(0.001, s))))

data <-  list(X=matrix(0, nrow=N, ncol=p1),
```

```
            Z=matrix(c(-0.24146164, -0.29673723, -0.27538621,
              0.41463628, 0.39023100, -0.22045922, -0.07062149,
              -0.22595298, 0.01247487, -0.14472589), nrow=N, ncol=q),
            tbin=as.integer(c(0, 0, 0, 1, 1, 0, 0, 0, 1, 0)),
            y=c(-1.9924944, -1.9345279, -1.3781082, -0.7646928,
              -0.2881649, 0.1545577, -0.6114224, -0.3703420,
              0.2320320, 0.7451867))

set.seed(42)
(par.post <- bniv(info, data, mcmc))
##should produce approx...
##       gamma1   delta1     eta1       beta       mu1        mu2       rho     s2
## [1,] 47.79852 1.231516 63.82816 -0.1581546 0.1076077 -0.2405642 0.4071552 0.626
```

3.3 bbiv

```
info <- list(theta=list(init=c(rep(gamma, p1), rep(delta, q), rep(eta, p1)),
               prior=list(mean=rep(0., s),
                 prec=diag(0.001, s))),
             beta=list(init=beta, prior=list(mean=0., prec=0.001)),
             rho=list(init=rho),
             mu=list(init=c(mu, mu),
               prior=list(mean=c(0., 0.), prec=diag(0.001, 2))))

data <-  list(X=matrix(0, nrow=N, ncol=p1),
             Z=matrix(c(-0.24146164, -0.29673723, -0.27538621,
               0.41463628, 0.39023100, -0.22045922, -0.07062149,
               -0.22595298, 0.01247487, -0.14472589), nrow=N, ncol=q),
             tbin=as.integer(c(0, 0, 0, 1, 1, 0, 0, 0, 1, 0)),
             ybin=as.integer(c(0, 0, 0, 0, 0, 1, 0, 0, 1, 1)))

set.seed(42)
(par.post <- bbiv(info, data, mcmc))
##should produce approx...
##      gamma1   delta1      eta1      beta        mu1        mu2       rho
## [1,] 3.41756 3.697175 -15.94475 0.1152464 -0.7233647 -0.8709668 0.3693283
```

2SLS estimates the IVE according to Imbens and Angrist (1994). We provide a function to compute the IVE from the binary treatment and outcome model.

```
bbivE(par.post[1, ], data$Z, data$X)
##should produce approx...
## 0.0995336
```

4 Nonparametric Models

Technically, the models we present are semiparametrics models, but we stick with the nonparametric nomenclature in this document for convenience. We do not provide a function for nonparametric model with a numeric treatment and outcome; for that see the bayesm package [`http://lib.stat.cmu.edu/R/CRAN/web/packages/bayesm`].

BIVpack provides 2 functions for estimating nonparametric models: `bnivDPM` for a binary treatment and a numeric outcome; and `bbivDPM` for a binary treatment and outcome. Each of these functions takes 3 arguments; `info`, `data` and `mcmc`; and returns the posterior samples of the parameters as a list. Besides being nonparametric, these functions provide smarter handling of the `info` and `data` parameters.

We follow the advice of Gelman et al. (2008). There are two parts relevant to our models: weakly informative prior parameters and data standardization. If you pass a `NULL` list for the `info` parameter, then a default prior parameterization is constructed for you. And, if you pass the optional parameter `stdize=TRUE`, then data standardization is performed and a back-transformation is employed.

4.1 bnivDPM

```
data <-  list(X=matrix(0, nrow=N, ncol=p1),
              Z=matrix(c(-0.24146164, -0.29673723, -0.27538621,
                0.41463628, 0.39023100, -0.22045922, -0.07062149,
                -0.22595298, 0.01247487, -0.14472589), nrow=N, ncol=q),
              tbin=as.integer(c(0, 0, 0, 1, 1, 0, 0, 0, 1, 0)),
              y=c(-1.9924944, -1.9345279, -1.3781082, -0.7646928,
                -0.2881649, 0.1545577, -0.6114224, -0.3703420,
                0.2320320, 0.7451867))

info <- list(beta=list(init=0, prior=list(mean=0, prec=0.001)),
              rho=list(init=0),
              mu=list(init=c(0, 0)),
              tau=list(init=1),
              theta=list(init=rep(0, s),
                prior=list(mean=rep(0, s),
                    prec=diag(c(rep(0.04, r), rep(0.001, p1)), s, s))),
              dpm=list(m=as.integer(3),
                  alpha=list(fixed=as.integer(0), init=1, prior=list(a=3, b=4)),
                  C=as.integer(0*(1:N)), states=as.integer(N),
                  prior=list(mu0=c(0, 0), T0=diag(0.001, 2), S0=diag(1000, 2),
                     alpha0=0.5, lambda0=0.4)))

set.seed(42)
(non.post <- bnivDPM(info, data, mcmc))
##should produce approx...
```

```
## [[1]]
## [[1]]$beta
## [1] 0.1070952

## [[1]]$theta
## [1]  7.557610  2.473789 63.828162

## [[1]]$C
##  [1] 0 0 0 0 0 0 0 0 0 0

## [[1]]$phi
##             [,1]      [,2]       [,3]    [,4]
## [1,] -0.07192311 -1.002996 -0.3468734 1.80566

## [[1]]$states
## [1] 10

## [[1]]$alpha
## [1] 0.1844611

set.seed(42)
(non.post <- bnivDPM(NULL, data, mcmc))
##should produce approx...
## same as above
```

4.2 bbivDPM

```
data <-  list(X=matrix(0, nrow=N, ncol=p1),
              Z=matrix(c(-0.24146164, -0.29673723, -0.27538621,
                0.41463628, 0.39023100, -0.22045922, -0.07062149,
                -0.22595298, 0.01247487, -0.14472589), nrow=N, ncol=q),
              tbin=as.integer(c(0, 0, 0, 1, 1, 0, 0, 0, 1, 0)),
              ybin=as.integer(c(0, 0, 0, 0, 0, 1, 0, 0, 1, 1)))

info <- list(beta=list(init=0, prior=list(mean=0, prec=1)),
              rho=list(init=0),
              mu=list(init=c(0, 0)),
              tau=list(init=1),
              theta=list(init=rep(0, s),
                prior=list(mean=rep(0, s), prec=diag(0.04, s))),
              dpm=list(m=as.integer(3),
                  alpha=list(fixed=as.integer(0), init=1, prior=list(a=3, b=4)),
                  C=as.integer(0*(1:N)), states=as.integer(N),
```

```
              prior=list(mu0=c(0, 0), T0=diag(1, 2), S0=diag(1, 2),
                   alpha0=0.5, lambda0=0.4)))

set.seed(42)
(non.post <- bbivDPM(info, data, mcmc))
##should produce approx...
## [[1]]
## [[1]]$beta
## [1] -0.2440909

## [[1]]$theta
## [1]  0.5403636  1.4322290 -2.5210857

## [[1]]$C
##  [1] 0 0 0 0 1 0 1 0 0 0

## [[1]]$phi
##           [,1]       [,2]  [,3]
## [1,] 0.5215824 -0.9161644 -0.39
## [2,] 0.1711343 -0.9186419 -0.43

## [[1]]$states
## [1] 8 2

## [[1]]$alpha
## [1] 1.029461

set.seed(42)
(non.post <- bbivDPM(NULL, data, mcmc))
##should produce approx...
## same as above

set.seed(42)
bbivDPM(info, data, mcmc, stdize=TRUE)
##should produce approx...
## [[1]]
## [[1]]$beta
## [1] -0.2440909

## [[1]]$theta
## [1]  0.2701818  1.3766962 -1.2605428

## [[1]]$C
```

```
##  [1] 0 0 0 0 1 0 1 0 0 0

## [[1]]$phi
##           [,1]       [,2]  [,3]
## [1,] 0.5256154 -0.9161644 -0.39
## [2,] 0.2071443 -0.9162833 -0.42

## [[1]]$states
## [1] 8 2

## [[1]]$alpha
## [1] 1.029461

bbivE(non.post[[1]], data$Z, data$X)
##should produce approx...
##  -0.1687742
```

References

(2013). *R: A Language and Environment for Statistical Computing*. [http://www.R-project.org].

Collins-Sussman, B., B. Fitzpatrick, and C. Pilato (2011). *Version control with subversion*. [http://svnbook.red-bean.com/en/1.7/svn-book.pdf].

Gelman, A., A. Jakulin, M. Grazia Pittau, and Y. Su (2008). A weakly informative default prior distribution for logistic and other regression models. *Annals of Applied Statistics 2*, 1360–83.

Imbens, G. and J. Angrist (1994). Identification and estimation of local average treatment effects. *Econometrica 62*, 467–75.