IPD meta-analysis

Michael Seo

2022-06-03

Fitting one-stage IPD meta-analysis model

We demonstrate how to run one-stage IPD meta-analysis using this package. First, let's generate sample IPD for illustration.

```
# devtools::install_github('MikeJSeo/bipd')
library(bipd)

## load in data
ds <- generate_ipdma_example(type = "continuous")
ds2 <- generate_ipdma_example(type = "binary")
head(ds)
#>   studyid treat          z1         z2  y
#> 1       1     1 -0.76655048 -0.2846117  9
#> 2       1     0 -0.75104148 -0.6713279 11
#> 3       1     0  0.11553318  0.5018156 11
#> 4       1     1 -0.22103264 -0.8402231  8
#> 5       1     0 -1.00228171  0.8083868 11
#> 6       1     1 -0.01249973 -0.5759004  8
```

The main function to set up the function for one-stage IPD meta-analysis is ipdma.model.onestage function. Refer to help(ipdma.model.onestage) for more details. Briefly to describe, "y" is the outcome of the study; "study" is a vector indicating which study the patient belongs to in a numerical sequence (i.e. 1, 2, 3, etc); "treat" is a vector indicating which treatment the patient was assigned to (i.e. 1 for treatment, 0 for placebo); "x" is a matrix of covariates for each patients; "response" is the outcome type, either "normal" or "binomial".

Another important parameter is the "shrinkage" parameter. To specify IPD meta-analysis without shrinkage, we set shrinkage = "none".

```
# continuous outcome
ipd <- with(ds, ipdma.model.onestage(y = y, study = studyid,
    treat = treat, X = cbind(z1, z2), response = "normal", shrinkage = "none"))
```

To view the JAGS code that was used to run the model, we can run the following command. Note that "alpha" is the study intercept, "beta" is the coefficient for main effects of the covariates, "gamma" is the coefficient for effect modifier, and "delta" is the average treatment effect.

```
cat(ipd$code)
#> model {
#>
#> ########## IPD-MA model
#> for (i in 1:Np) {
#>  y[i] ~ dnorm(mu[i], sigma)
#>  mu[i] <- alpha[studyid[i]] + inprod(beta[], X[i,]) +
```

```
#>      (1 - equals(treat[i],1)) * inprod(gamma[], X[i,]) + d[studyid[i],treat[i]]
#> }
#> sigma ~ dgamma(0.001, 0.001)
#>
#> #####treatment effect
#> for(j in 1:Nstudies){
#>  d[j,1] <- 0
#>  d[j,2] ~ dnorm(delta[2], tau)
#> }
#> sd ~ dnorm(0, 1)T(0,)
#> tau <- pow(sd, -2)
#>
#> ## prior distribution for the average treatment effect
#> delta[1] <- 0
#> delta[2] ~ dnorm(0, 0.001)
#>
#> ## prior distribution for the study intercept
#> for (j in 1:Nstudies){
#>  alpha[j] ~ dnorm(0, 0.001)
#> }
#>
#> ## prior distribution for the main effect of the covariates
#> for(k in 1:Ncovariate){
#>  beta[k] ~ dnorm(0, 0.001)
#> }
#> ## prior distribution for the effect modifiers under no shrinkage
#> for(k in 1:Ncovariate){
#>  gamma[k] ~ dnorm(0, 0.001)
#> }
#> }
```

Once the model is set up using ipdma.model.onestage function, we use ipd.run function to run the model. help(ipd.run) describes possible parameters to specify.

```
samples <- ipd.run(ipd, n.chains = 3, n.burnin = 500, n.iter = 5000)
#> Compiling model graph
#>    Resolving undeclared variables
#>    Allocating nodes
#> Graph information:
#>    Observed stochastic nodes: 600
#>    Unobserved stochastic nodes: 19
#>    Total graph size: 6034
#>
#> Initializing model

samples <- samples[, -3]  #remove delta[1] which is 0
summary(samples)
#>
#> Iterations = 1501:6500
#> Thinning interval = 1
#> Number of chains = 3
#> Sample size per chain = 5000
#>
#> 1. Empirical mean and standard deviation for each variable,
```

```
#>    plus standard error of the mean:
#>
#>              Mean      SD  Naive SE Time-series SE
#> alpha[1] 11.0927 0.05330 0.0004352      0.0007817
#> alpha[2]  8.0512 0.05668 0.0004628      0.0009121
#> alpha[4]  9.6084 0.05062 0.0004133      0.0007013
#> alpha[5] 12.9145 0.05356 0.0004373      0.0007973
#> alpha[6] 15.8189 0.05189 0.0004237      0.0007985
#> beta[1]   0.1666 0.02250 0.0001837      0.0003485
#> beta[2]   0.2731 0.02067 0.0001688      0.0002867
#> delta[1]  0.0000 0.00000 0.0000000      0.0000000
#> delta[2] -2.3472 0.76583 0.0062530      0.0061900
#> gamma[1] -0.5167 0.03041 0.0002483      0.0004696
#> gamma[2]  0.5539 0.03057 0.0002496      0.0004304
#>
#> 2. Quantiles for each variable:
#>
#>              2.5%     25%     50%     75%   97.5%
#> alpha[1] 10.9874 11.0562 11.0932 11.1289 11.1960
#> alpha[2]  7.9385  8.0136  8.0512  8.0893  8.1618
#> alpha[4]  9.5103  9.5744  9.6081  9.6421  9.7096
#> alpha[5] 12.8102 12.8782 12.9142 12.9508 13.0181
#> alpha[6] 15.7177 15.7840 15.8187 15.8534 15.9213
#> beta[1]   0.1222  0.1516  0.1665  0.1817  0.2105
#> beta[2]   0.2330  0.2593  0.2734  0.2868  0.3139
#> delta[1]  0.0000  0.0000  0.0000  0.0000  0.0000
#> delta[2] -3.8495 -2.8403 -2.3518 -1.8558 -0.8020
#> gamma[1] -0.5761 -0.5373 -0.5166 -0.4965 -0.4568
#> gamma[2]  0.4944  0.5334  0.5541  0.5744  0.6139
# plot(samples) #traceplot and posterior of parameters
# coda::gelman.plot(samples) #gelman diagnostic plot
```

We can find patient-specific treatment effect using the treatment.effect function. To do this we need to specify the covariate values for the patient that we want to predict patient-specific treatment effect.

```
treatment.effect(ipd, samples, newpatient = c(1, 0.5))
#>     0.025       0.5     0.975
#> -4.118233 -2.624881 -1.061838
```

Incorporating shrinkage and variable selection

For the second example, let's use the same data, but include shrinkage (i.e. Bayesian LASSO) in the effect modifiers (i.e. treatment-covariate interactions). We can specify Bayesian LASSO by setting shrinkage = "laplace". Lambda is the shrinkage parameter and we can set the prior for lambda using lambda.prior parameter. The default lambda prior for Bayesian LASSO is $\lambda^{-1} \sim dunif(0, 5)$.

```
ipd <- with(ds, ipdma.model.onestage(y = y, study = studyid,
    treat = treat, X = cbind(z1, z2), response = "normal", shrinkage = "laplace"))
samples <- ipd.run(ipd, pars.save = c("beta", "gamma", "delta",
    "lambda", "tt"), n.chains = 3, n.burnin = 500, n.iter = 5000)
#> Compiling model graph
#>    Resolving undeclared variables
```

```
#>    Allocating nodes
#> Graph information:
#>    Observed stochastic nodes: 600
#>    Unobserved stochastic nodes: 20
#>    Total graph size: 6039
#>
#> Initializing model
summary(samples)
#>
#> Iterations = 1501:6500
#> Thinning interval = 1
#> Number of chains = 3
#> Sample size per chain = 5000
#>
#> 1. Empirical mean and standard deviation for each variable,
#>    plus standard error of the mean:
#>
#>             Mean      SD  Naive SE Time-series SE
#> beta[1]   0.1653 0.02212 0.0001806      0.0003722
#> beta[2]   0.2748 0.02057 0.0001680      0.0003302
#> delta[1]  0.0000 0.00000 0.0000000      0.0000000
#> delta[2] -2.3569 0.76231 0.0062242      0.0062243
#> gamma[1] -0.5144 0.02997 0.0002447      0.0005373
#> gamma[2]  0.5509 0.03060 0.0002499      0.0005249
#> lambda    0.3240 0.12605 0.0010292      0.0014112
#> tt        2.4696 0.95869 0.0078276      0.0105926
#>
#> 2. Quantiles for each variable:
#>
#>             2.5%     25%     50%     75%   97.5%
#> beta[1]   0.1212  0.1505  0.1654  0.1799  0.2085
#> beta[2]   0.2343  0.2609  0.2749  0.2886  0.3153
#> delta[1]  0.0000  0.0000  0.0000  0.0000  0.0000
#> delta[2] -3.8825 -2.8461 -2.3501 -1.8715 -0.8193
#> gamma[1] -0.5726 -0.5346 -0.5146 -0.4946 -0.4548
#> gamma[2]  0.4922  0.5301  0.5509  0.5714  0.6115
#> lambda    0.2033  0.2356  0.2854  0.3701  0.6596
#> tt        1.5055  1.8002  2.1787  2.8313  5.0371
```

We can also use SSVS (stochastic search variable selection) by setting shrinkage = “SSVS”. This time let’s use the binomial dataset. “Ind” is the indicator for assigning a slab prior (instead of a spike prior) i.e. indicator for including a covariate. “eta” is the standard deviation of the slab prior.

```
ipd <- with(ds2, ipdma.model.onestage(y = y, study = studyid,
    treat = treat, X = cbind(w1, w2), response = "binomial",
    shrinkage = "SSVS"))
samples <- ipd.run(ipd, pars.save = c("beta", "gamma", "delta",
    "Ind", "eta"), n.chains = 3, n.burnin = 500, n.iter = 5000)
#> Compiling model graph
#>    Resolving undeclared variables
#>    Allocating nodes
#> Graph information:
#>    Observed stochastic nodes: 600
#>    Unobserved stochastic nodes: 21
```

```
#>    Total graph size: 6649
#>
#> Initializing model
summary(samples)
#>
#> Iterations = 1501:6500
#> Thinning interval = 1
#> Number of chains = 3
#> Sample size per chain = 5000
#>
#> 1. Empirical mean and standard deviation for each variable,
#>    plus standard error of the mean:
#>
#>              Mean      SD  Naive SE Time-series SE
#> Ind[1]    0.15927 0.36594 0.0029879       0.009244
#> Ind[2]    0.19767 0.39825 0.0032517       0.010316
#> beta[1]   0.19194 0.09674 0.0007899       0.001255
#> beta[2]   0.03942 0.10113 0.0008257       0.001594
#> delta[1]  0.00000 0.00000 0.0000000       0.000000
#> delta[2] -0.13557 0.36049 0.0029434       0.005014
#> eta       1.96340 1.49457 0.0122031       0.045752
#> gamma[1] -0.00518 0.08612 0.0007032       0.001394
#> gamma[2]  0.04635 0.10390 0.0008484       0.002133
#>
#> 2. Quantiles for each variable:
#>
#>              2.5%       25%       50%     75%  97.5%
#> Ind[1]    0.00000  0.000000  0.000000 0.00000 1.0000
#> Ind[2]    0.00000  0.000000  0.000000 0.00000 1.0000
#> beta[1]   0.00388  0.127129  0.191287 0.25561 0.3831
#> beta[2]  -0.16401 -0.026796  0.042097 0.10653 0.2336
#> delta[1]  0.00000  0.000000  0.000000 0.00000 0.0000
#> delta[2] -0.86009 -0.345124 -0.131825 0.07379 0.5812
#> eta       0.03396  0.595621  1.714467 3.19693 4.8127
#> gamma[1] -0.20357 -0.038593 -0.001112 0.03054 0.1801
#> gamma[2] -0.11410 -0.007833  0.017468 0.08630 0.3161
treatment.effect(ipd, samples, newpatient = c(1, 0.5))  # binary outcome reports odds ratio
#>     0.025       0.5     0.975
#> 0.4206599 0.8940541 1.8442153
```

Fitting one-stage IPD network meta-analysis

We now demonstrate how to run IPD network meta-analysis using this package.

```
## load in data
ds <- generate_ipdnma_example(type = "continuous")
ds2 <- generate_ipdnma_example(type = "binary")
head(ds)
#>   studyid treat          z1           z2  y
#> 1       1     2  0.06051735  0.517791695  9
#> 2       1     1  0.99963768 -1.983681235 11
#> 3       1     1 -1.17001002 -0.713301704 11
#> 4       1     2 -1.29930850 -0.045076496  9
```

```
#> 5       1     2  0.84904840  0.946693526  8
#> 6       1     2  0.19602346 -0.001556904  8
```

The main function to set up the function for one-stage IPD network meta-analysis is ipdnma.model.onestage function. The function is very similar to ipdma.model.onestage except that now we have number of treatments greater than 2. Consequently, “treat” parameter is defined differently i.e. 1 assigns baseline treatment and other treatments should be be assigned 2, 3, 4, etc.

```
# continuous outcome
ipd <- with(ds, ipdnma.model.onestage(y = y, study = studyid,
    treat = treat, X = cbind(z1, z2), response = "normal", shrinkage = "none"))
cat(ipd$code)
#> model {
#>
#> ########## IPD-NMA model
#> for (i in 1:Np) {
#>  y[i] ~ dnorm(mu[i], sigma)
#>  mu[i] <- alpha[studyid[i]] + inprod(beta[], X[i,]) +
#>      inprod(gamma[treat[i],], X[i,]) + d[studyid[i],treatment.arm[i]]
#> }
#> sigma ~ dgamma(0.001, 0.001)
#>
#> #####treatment effect
#> for(i in 1:Nstudies){
#>  w[i,1] <- 0
#>  d[i,1] <- 0
#>  for(k in 2:na[i]){
#>      d[i,k] ~ dnorm(mdelta[i,k], taudelta[i,k])
#>      mdelta[i,k] <-  delta[t[i,k]] - delta[t[i,1]] + sw[i,k]
#>      taudelta[i,k] <- tau * 2 * (k-1)/k
#>      w[i,k] <- d[i,k] - delta[t[i,k]] + delta[t[i,1]]
#>      sw[i,k] <- sum(w[i, 1:(k-1)]) / (k-1)
#>  }
#> }
#> sd ~ dnorm(0, 1)T(0,)
#> tau <- pow(sd, -2)
#>
#> ## prior distribution for the average treatment effect
#> delta[1] <- 0
#> for(k in 2:Ntreat){
#>  delta[k] ~ dnorm(0, 0.001)
#> }
#>
#> ## prior distribution for the study intercept
#> for (j in 1:Nstudies){
#>  alpha[j] ~ dnorm(0, 0.001)
#> }
#>
#> ## prior distribution for the main effect of the covariates
#> for(k in 1:Ncovariate){
#>  beta[k] ~ dnorm(0, 0.001)
#> }
#> ## prior distribution for the effect modifiers under no shrinkage
#> for(k in 1:Ncovariate){
```

```
#>  gamma[1,k] <- 0
#>  for(m in 2:Ntreat){
#>      gamma[m,k] ~ dnorm(0, 0.001)
#>  }
#> }
#> }
samples <- ipd.run(ipd, pars.save = c("beta", "gamma", "delta"),
    n.chains = 3, n.burnin = 500, n.iter = 5000)
#> Compiling model graph
#>    Resolving undeclared variables
#>    Allocating nodes
#> Graph information:
#>    Observed stochastic nodes: 1000
#>    Unobserved stochastic nodes: 33
#>    Total graph size: 10133
#>
#> Initializing model
summary(samples)
#>
#> Iterations = 1501:6500
#> Thinning interval = 1
#> Number of chains = 3
#> Sample size per chain = 5000
#>
#> 1. Empirical mean and standard deviation for each variable,
#>    plus standard error of the mean:
#>
#>               Mean      SD  Naive SE Time-series SE
#> beta[1]     0.2035 0.01883 0.0001537      0.0003311
#> beta[2]     0.3042 0.02078 0.0001697      0.0003855
#> delta[1]    0.0000 0.00000 0.0000000      0.0000000
#> delta[2]   -2.9863 0.05921 0.0004835      0.0008742
#> delta[3]   -1.1801 0.06271 0.0005120      0.0008914
#> gamma[1,1]  0.0000 0.00000 0.0000000      0.0000000
#> gamma[2,1] -0.5814 0.02813 0.0002297      0.0004301
#> gamma[3,1] -0.3287 0.02852 0.0002329      0.0004306
#> gamma[1,2]  0.0000 0.00000 0.0000000      0.0000000
#> gamma[2,2]  0.5354 0.02896 0.0002364      0.0004787
#> gamma[3,2]  0.3893 0.02882 0.0002353      0.0004780
#>
#> 2. Quantiles for each variable:
#>
#>               2.5%     25%     50%     75%   97.5%
#> beta[1]     0.1663  0.1910  0.2036  0.2159  0.2408
#> beta[2]     0.2635  0.2902  0.3042  0.3184  0.3447
#> delta[1]    0.0000  0.0000  0.0000  0.0000  0.0000
#> delta[2]   -3.1089 -3.0225 -2.9853 -2.9491 -2.8693
#> delta[3]   -1.3076 -1.2189 -1.1792 -1.1398 -1.0593
#> gamma[1,1]  0.0000  0.0000  0.0000  0.0000  0.0000
#> gamma[2,1] -0.6370 -0.6004 -0.5811 -0.5626 -0.5265
#> gamma[3,1] -0.3852 -0.3477 -0.3288 -0.3098 -0.2722
#> gamma[1,2]  0.0000  0.0000  0.0000  0.0000  0.0000
#> gamma[2,2]  0.4790  0.5157  0.5354  0.5550  0.5917
#> gamma[3,2]  0.3327  0.3699  0.3891  0.4090  0.4458
treatment.effect(ipd, samples, newpatient = c(1, 0.5))
```

```
#> $`treatment 2`
#>     0.025       0.5     0.975
#> -3.446742 -3.314446 -3.186788
#>
#> $`treatment 3`
#>     0.025       0.5     0.975
#> -1.470633 -1.327761 -1.192262
```

We can apply shrinkage on the effect modifiers (treatment-covariate interactions) as before.

```
# SSVS
ipd <- with(ds, ipdnma.model.onestage(y = y, study = studyid,
    treat = treat, X = cbind(z1, z2), response = "normal", shrinkage = "SSVS"))
samples <- ipd.run(ipd, pars.save = c("beta", "gamma", "delta",
    "Ind", "eta"), n.chains = 3, n.burnin = 500, n.iter = 5000)
#> Compiling model graph
#>    Resolving undeclared variables
#>    Allocating nodes
#> Graph information:
#>    Observed stochastic nodes: 1000
#>    Unobserved stochastic nodes: 38
#>    Total graph size: 10155
#>
#> Initializing model
summary(samples)
#>
#> Iterations = 1501:6500
#> Thinning interval = 1
#> Number of chains = 3
#> Sample size per chain = 5000
#>
#> 1. Empirical mean and standard deviation for each variable,
#>    plus standard error of the mean:
#>
#>               Mean      SD  Naive SE Time-series SE
#> Ind[2,1]    0.9995 0.02309 0.0001885      0.0003235
#> Ind[3,1]    0.9785 0.14494 0.0011834      0.0060205
#> Ind[2,2]    0.9986 0.03739 0.0003053      0.0005795
#> Ind[3,2]    0.9876 0.11067 0.0009036      0.0037112
#> beta[1]     0.2024 0.01928 0.0001574      0.0003250
#> beta[2]     0.3059 0.02109 0.0001722      0.0004329
#> delta[1]    0.0000 0.00000 0.0000000      0.0000000
#> delta[2]   -2.9866 0.05915 0.0004830      0.0008783
#> delta[3]   -1.1812 0.06214 0.0005073      0.0009171
#> eta         0.8369 0.74731 0.0061018      0.0353363
#> gamma[1,1]  0.0000 0.00000 0.0000000      0.0000000
#> gamma[2,1] -0.5798 0.02839 0.0002318      0.0004445
#> gamma[3,1] -0.3269 0.02896 0.0002365      0.0004303
#> gamma[1,2]  0.0000 0.00000 0.0000000      0.0000000
#> gamma[2,2]  0.5328 0.02921 0.0002385      0.0005036
#> gamma[3,2]  0.3873 0.02921 0.0002385      0.0005037
#>
#> 2. Quantiles for each variable:
#>
```

```
#>               2.5%     25%     50%     75%   97.5%
#> Ind[2,1]    1.0000  1.0000  1.0000  1.0000  1.0000
#> Ind[3,1]    1.0000  1.0000  1.0000  1.0000  1.0000
#> Ind[2,2]    1.0000  1.0000  1.0000  1.0000  1.0000
#> Ind[3,2]    1.0000  1.0000  1.0000  1.0000  1.0000
#> beta[1]     0.1647  0.1894  0.2024  0.2156  0.2402
#> beta[2]     0.2646  0.2917  0.3058  0.3202  0.3476
#> delta[1]    0.0000  0.0000  0.0000  0.0000  0.0000
#> delta[2]   -3.1069 -3.0230 -2.9857 -2.9495 -2.8714
#> delta[3]   -1.3052 -1.2195 -1.1813 -1.1421 -1.0590
#> eta         0.3027  0.4653  0.6177  0.8853  3.7834
#> gamma[1,1]  0.0000  0.0000  0.0000  0.0000  0.0000
#> gamma[2,1] -0.6352 -0.5987 -0.5798 -0.5607 -0.5247
#> gamma[3,1] -0.3835 -0.3468 -0.3272 -0.3075 -0.2692
#> gamma[1,2]  0.0000  0.0000  0.0000  0.0000  0.0000
#> gamma[2,2]  0.4752  0.5134  0.5328  0.5525  0.5897
#> gamma[3,2]  0.3301  0.3676  0.3872  0.4069  0.4445
treatment.effect(ipd, samples, newpatient = c(1, 0.5))
#> $`treatment 2`
#>     0.025       0.5     0.975
#> -3.446273 -3.314349 -3.188055
#>
#> $`treatment 3`
#>     0.025       0.5     0.975
#> -1.465770 -1.328106 -1.191418
# Bayesian LASSO ipd <- with(ds, ipdnma.model.onestage(y =
# y, study = studyid, treat = treat, X = cbind(z1, z2),
# response = 'normal', shrinkage = 'laplace', lambda.prior
# = list('dgamma',2,0.1))) samples <- ipd.run(ipd,
# pars.save = c('beta', 'gamma', 'delta', 'lambda', 'tt'),
# n.chains = 3, n.burnin = 500, n.iter = 5000)
```

p.s. Note that in the network meta-analysis literature, “d” usually refers to average treatment effect and “delta” refers to study-specific treatment effect. In this R package, we have flipped around the two notations.