Implementing methods for multipollutant mixtures analyses: mmpack tutorial

Lauren Hoskovec

August 23, 2019

Introduction

The R package mmpack implements five Bayesian methods developed for estimating the association between health outcomes and exposures to mixtures of environmental pollutants. These methods include two versions of nonparametric Bayes shrinkage, two versions of Bayesian profile regression, and Bayesian kernel machine regression. The package includes functions to fit each method and post-process the output. In two cases there is existing software to implement the desired methods and mmpack provides a wrapper function to call those existing packages and post-process the output. We will demonstrate each method separately.

Package Details

First, load the package:

```
library(devtools)
install_github("lvhoskovec/mmpack")
# devtools::install_github("lvhoskovec/mmpack", build_vignettes = TRUE)
```

```
library(mmpack)
```

The following functions are available in the package mmpack:

function	brief description of usage
bkmr_wrapper	wrapper function to fit Bayesian kernel machine regression using R package bkmr and post-process output
npb	function to fit nonparametric Bayes shrinkage models
premium_wrapper	wrapper function to fit supervised profile regression using R package PReMiuM and post-process output
profileReg	function to fit supervised and unsupervised profile regression models
simexpodat	simulate up to 1000 observations of air pollution and pesticide exposure data, covariate data, and continuous response data
Xdat	documentation and dataset for air pollution and pesticide exposure data included in package
simLinearResponse	simulate response data as a linear function of predictors

function	brief description of usage
simNonlinearResponse	simulate response data as a nonlinear function of predictors
simProfilesResponse	simulate response data as a fixed profiles (piece-wise constant) function of predictors
fitModels	fit and evaluate 5 Bayesian methods plus linear models in 3 different scenarios
summarizeSimulation	summarize results from a simulation using `fitModels` function

Simulate Data

`Xdat` is a data frame included in the package mmpack that has 1000 observations with exposure concentrations for four air pollutants (NO_2, O_3, PM_{10}, and $PM_{2.5}$) and three pesticides (C, OP, and MeBr) generated from random locations in the Fresno, CA area. The air pollution data come from the EPA Air Quality System Data Mart and the pesiticide data come from the California Pesticide Use Report.

```
head(Xdat)
```

```
##      C MeBr       NO2         O3    OP      PM10  PM2.5
## [1,] 0    0  9.636079 0.05361471 0e+00 40.000000  8.100
## [2,] 0    0  7.552415 0.04533614 0e+00 36.666667 13.460
## [3,] 0    0  8.569427 0.04324357 1e-05 33.000000  5.780
## [4,] 0    0  6.046497 0.03153514 0e+00  5.333333  2.350
## [5,] 0    0  8.624817 0.04758800 0e+00 62.333333  9.825
## [6,] 0    0 11.753044 0.02549586 0e+00 38.000000  6.460
```

Next we will simulate response data. The function `simexpodat` takes in two parameters: `n` is a sample of size of up to 1000 and `Xdat` is a matrix of exposure data. The function randomly selects n unique subjects from `Xdat` to be used as exposure data, which is then scaled to have mean 0 and variance 1. The exposure-response function, $h(\mathbf{x})$ is:

$$h(\mathbf{x}) = 3NO_2 - 2O_3 + 2.5OP - 4PM_{2.5} + 0.3NO_2 * O_3 - 0.6OP * PM_{2.5}$$

The function also simulates 10 iid N(0,1) covariates for each observation. Finally, the health response is calculated as the exposure-response function plus a linear combination of covariates and iid N(0,1) error.

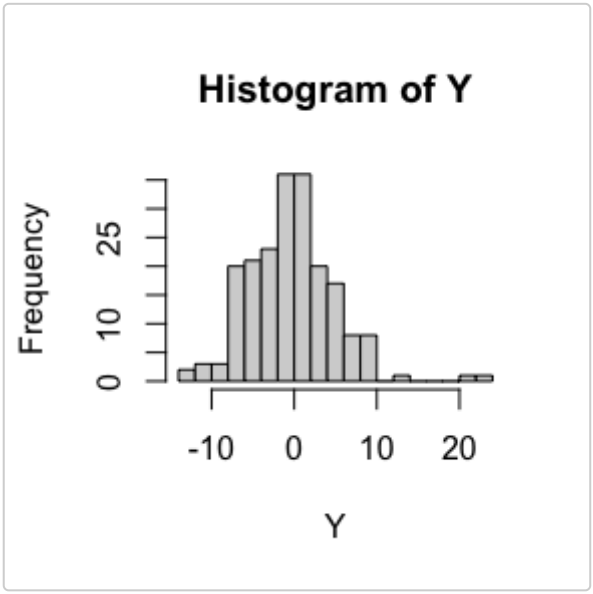
```
set.seed(12345)
dat <- simexpodat(n = 200, Xdat = Xdat)
X <- dat$X # exposure data
W <- dat$W # covariate data
Y <- dat$Y # response data
```

Also included in `dat` is the true exposure-response function `h` for each individual, the active main effect mixture components `active` and the active interactions `active.ints`.

```
h <- dat$h
active <- dat$active
active.ints <- dat$active.ints
```

We can look at the distribution of the observed response.

```
hist(Y, breaks = 20)
```



It is helpful to look at the variance of the exposure data for a given simulated data set because some exposures have mostly zero values and any given sample of the data may not have sufficient variation to use (e.g. MeBr below).

```
var(X)
```

```
##                C MeBr         NO2          O3          OP        PM10
## C      1.00000000   NA -0.10223407  0.01998937  0.17895996 -0.05803235
## MeBr           NA   NA          NA          NA          NA          NA
## NO2   -0.10223407   NA  1.00000000 -0.29779859  0.09869839  0.38701167
## O3     0.01998937   NA -0.29779859  1.00000000 -0.06758498  0.46212385
## OP     0.17895996   NA  0.09869839 -0.06758498  1.00000000 -0.05581503
## PM10  -0.05803235   NA  0.38701167  0.46212385 -0.05581503  1.00000000
## PM2.5 -0.02407493   NA  0.66923229 -0.31071018  0.06302807  0.40319922
##             PM2.5
## C     -0.02407493
## MeBr           NA
## NO2    0.66923229
## O3    -0.31071018
## OP     0.06302807
## PM10   0.40319922
## PM2.5  1.00000000
```

Since there is no variation in MeBr, will we remove it from the exposure data matrix.

```
X <- X[,-2]
```

Now we will walk through implementing each method. All models are fit using MCMC methods and in each example we run a total of 200 iterations of the sampler, including 100 burn-in iterations. This should be increased for any analysis as 200 iterations is not sufficient for adequate mixing and convergence of the

model. Running 20,000 iterations is often sufficient, but convergence should always be checked via visual inspection of traceplots or comparison with multiple chains to determine an adequate chain length.

Fit Models

Nonparametric Bayes Shrinkage

Nonparametric Bayes shrinkage (NPB) is a Bayesian linear model that places a Dirichlet Process (DP) prior on the regression coefficients to set some coefficients exactly to 0, effectively excluding them from the model, and clusters correlated exposures to reduce variance of the estimator.

The function `npb` fits NPB using Markov chain Monte Carlo (MCMC) methods. The function takes in the following arguments: `niter` is the number of total iterations including burn-in, `nburn` is the number of burn-in iterations, `X` is a matrix of predictor data, `Y` is a vector of continuous response data, `W` is a matrix of covariate data. The argument `scaleY` is logical and indicates if the user wants the response to be scaled before the model is fit. The argument `priors` is an optional list of prior hyperparameters (see details below). The function `npb` allows the user to fit NPB with main effects only by choosing `interact = FALSE` or to fit NPB with main effects and all pairwise multiplicative interactions by choosing `interact = TRUE`. If `interact = TRUE` you can also choose `XWinteract = TRUE` to allow the interactions between X and W to be simultaneously estimated with the interactions among X. Finally, `intercept` is a logical parameter that indicates if an overall intercept should be estimated with the covariates.

```
npb(niter, nburn, X, Y, W, scaleY = FALSE, priors, interact = FALSE, intercept = TRUE)
```

If priors is missing, then priors will be set to NULL and the default priors will be used (see below).

The response is modeled as:

$$y_i|\boldsymbol{\beta},\boldsymbol{\gamma},\boldsymbol{\zeta},\sigma^2 \sim \mathrm{N}(\gamma_0 + \mathbf{x}_i^T\boldsymbol{\beta} + \mathbf{z}_i^T\boldsymbol{\zeta} + \mathbf{w}_i^T\boldsymbol{\gamma}, \sigma^2),$$

where $\mathbf{x_i}$ is a vector of exposures, $\mathbf{z_i}$ is a vector of pairwise multiplicative interactions between the exposures, and $\mathbf{w_i}$ is a vector of covariates.

The regression coefficients $\boldsymbol{\beta}$ for main effects and $\boldsymbol{\zeta}$ for the interactions are modeled with a DP prior and we place semi-conjugate priors on the covariate coefficients $\boldsymbol{\gamma}$ and the error variance σ^2.

$$
\begin{aligned}
\beta_j|D_1 &\overset{iid}{\sim} D_1, j = 1, \dots p \\
D_1|\alpha_1, D_{01} &\sim DP(\alpha_1 D_{01}) \\
D_{01}|\pi_{01} &= \pi_{01}\delta_0 + (1-\pi_{01})G_1 \\
G_1|\mu_1, \phi_1^2 &\equiv \mathrm{N}(\mu_1, \phi_1^2) \\
\mu_1 &\sim \mathrm{N}(0, \sigma_{\mu 1}^2) \\
\phi_1^{-2} &\sim \mathrm{Gamma}(\alpha_{\phi 1}, \beta_{\phi 1}) \\
\pi_{01} &\sim \mathrm{Beta}(\alpha_\pi, \beta_\pi) \\
\alpha_1 &\sim \mathrm{Gamma}(\alpha_{\alpha 1}, \beta_{\alpha 1})
\end{aligned}
$$

$$
\begin{aligned}
\zeta_{jk}|D_2 &\overset{iid}{\sim} D_2, j = 1, \dots p-1, k = j+1, \dots, p \\
D_2|\alpha_{02}, D_{02} &\sim DP(\alpha_2, D_{02}) \\
D_{02}|\pi_{02}, G_2 &= \pi_{02}\delta_0 + (1-\pi_{02})G_2 \\
G_2|\mu_2, \phi_2^2 &\equiv \mathrm{N}(\mu_2, \phi_2^2) \\
\mu_2 &\sim \mathrm{N}(0, \sigma_{\mu 2}^2) \\
\phi_2^{-2} &\sim \mathrm{Gamma}(\alpha_{\phi 2}, \beta_{\phi 2}) \\
\pi_{02} &\sim \mathrm{Beta}(\alpha_{\pi 2}, \beta_{\pi 2}) \\
\alpha_2 &\sim \mathrm{Gamma}(\alpha_{\alpha 2}, \beta_{\alpha 2})
\end{aligned}
$$

$$
\begin{aligned}
\gamma_0 &\sim N(\mu_0, \kappa_0^2) \\
\boldsymbol{\gamma} &\sim \mathrm{N}(\boldsymbol{\mu}_\gamma, \boldsymbol{\kappa}^2 \mathbf{I}) \\
\sigma^{-2} &\sim \mathrm{Gamma}(\alpha_\sigma, \beta_\sigma)
\end{aligned}
$$

Prior hyperparameters that the user may specify are listed as follows. The following table gives the control parameter (the name of the parameter in the list of priors), the model parameter, the default value, and a description for each prior hyperparameter.

control parameter	model parameter	default value	description
a.sig	α_σ	1	shape parameter for gamma prior on σ^{-2}
b.sig	β_σ	1	rate parameter for gamma prior on σ^{-2}
a.phi1	$\alpha_{\phi 1}$	1	shape parameter for gamma prior on ϕ_1^{-2}
b.phi1	$\beta_{\phi 1}$	1	rate parameter for gamma prior on ϕ_1^{-2}
a.phi2	$\alpha_{\phi 2}$	1	shape parameter for gamma prior on ϕ_2^{-2}
b.phi2	$\beta_{\phi 2}$	1	rate parameter for gamma prior on ϕ_2^{-2}
alpha.a	$\alpha_{\alpha 1}$	2	shape parameter for gamma prior on α_1

control parameter	model parameter	default value	description
alpha.b	$\beta_{\alpha 1}$	1	rate parameter for gamma prior on α_1
alpha.2.a	$\alpha_{\alpha 2}$	2	shape parameter for gamma prior on α_2
alpha.2.b	$\beta_{\alpha 2}$	1	rate parameter for gamma prior on α_2
mu.0	μ_0	0	mean parameter for normal prior on γ_0, intercept
kappa2inv.0	κ_0^{-2}	1	precision paramter for normal prior on γ_0, intercept
mu.gamma	$\boldsymbol{\mu}_\gamma$	$\mathbf{0}$	q-length vector of the mean parameters for independent normal priors on covariates
kap2inv	$\boldsymbol{\kappa}^{-2}$	$\mathbf{1}$	q-length vector of the precision parameters for independent normal priors on covariates
sig2inv.mu1	$\sigma_{\mu 1}^{-2}$	1	precision parameter for normal prior on mean of D_1
sig2inv.mu2	$\sigma_{\mu 2}^{-2}$	1	precision parameter for normal prior on mean of D_2
alpha.pi	α_π	1	shape1 parameter for beta prior on π_{01}
beta.pi	β_π	1	shape2 parameter for beta prior on π_{01}
alpha.pi2	$\alpha_{\pi 2}$	9	shape1 parameter for beta prior on π_{02}
beta.pi2	$\beta_{\pi 2}$	1	shape2 parameter for beta prior on π_{02}

We can change any of the prior hyperparameters by specifying new values in a list.

```
priors.npb <- list(alpha.pi = 2, beta.pi = 2, alpha.pi2 = 2, beta.pi2 = 2)
```

Now, we can fit the NPB model with our user-specified priors and use the other default priors. Let's fit NPB with interactions (`interact = TRUE`). We must specify the number of iterations, number of burn-in interations, and data (X = predictors, Y = response, W = covariates).

```
fit.npb <- npb(niter = 200, nburn = 100, X = X, Y = Y, W = W, scaleY = TRUE,
               priors = priors.npb, interact = TRUE)
```

Now that the model has been fit, we want to look at the output. The `summary` function provides a summary of the posterior distribution of main effect and interaction estimates. See the help file for `summary.npb` to see a detailed description of the summary output. The following shows a summary of the posterior distribution of the main effect regression coefficient estimates.

```
npb.sum <- summary(fit.npb)
npb.sum$main.effects
```

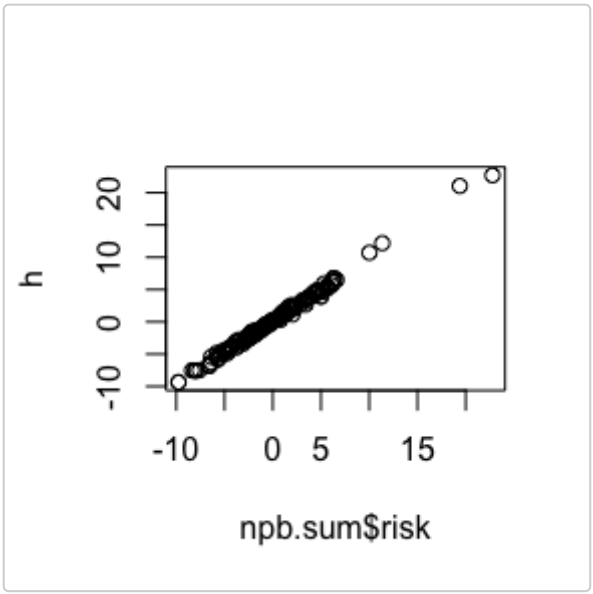
```
##      Posterior Mean          SD 95% CI Lower 95% CI Upper  PIP
## [1,]  -0.0008060289 0.007925666     0.000000   0.00000000 0.04
## [2,]   2.9914244797 0.111902651     2.802920   3.22413644 1.00
## [3,]  -2.1402729896 0.078913899    -2.303637  -1.98575614 1.00
## [4,]   2.3554528266 0.077556429     2.226992   2.52860131 1.00
## [5,]   0.0063926219 0.033548772     0.000000   0.09220273 0.05
## [6,]  -4.1317712959 0.107240173    -4.365872  -3.93938663 1.00
```

We can also look at a summary of the posterior distribution of the estimated exposure-response function, called risk, for each individual. Then we plot the estimated risk against the true exposure-response function.

```
head(apply(npb.sum$risk.summary, 2, FUN = function(x) round(x, 2)))
```

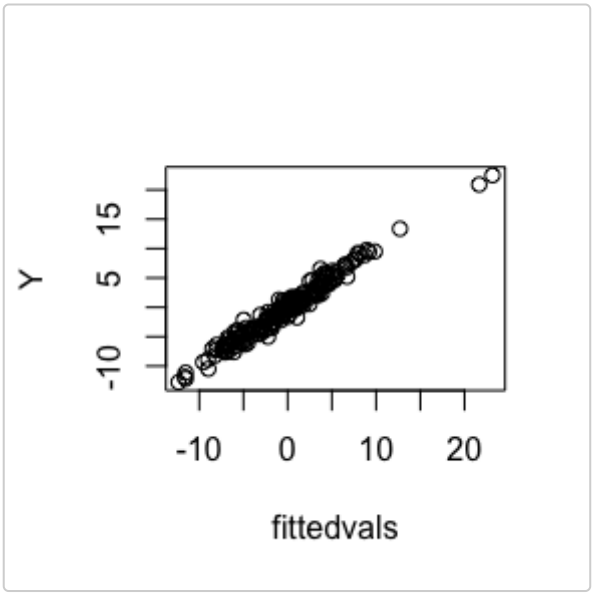
```
##      Posterior Mean   SD 95% CI Lower 95% CI Upper
## [1,]          -0.83 0.11        -1.04        -0.63
## [2,]          -6.56 0.28        -7.05        -6.02
## [3,]           2.38 0.11         2.17         2.60
## [4,]          -2.16 0.18        -2.50        -1.79
## [5,]          -2.45 0.12        -2.66        -2.20
## [6,]           1.43 0.10         1.24         1.65
```

```
plot(npb.sum$risk, h)
```



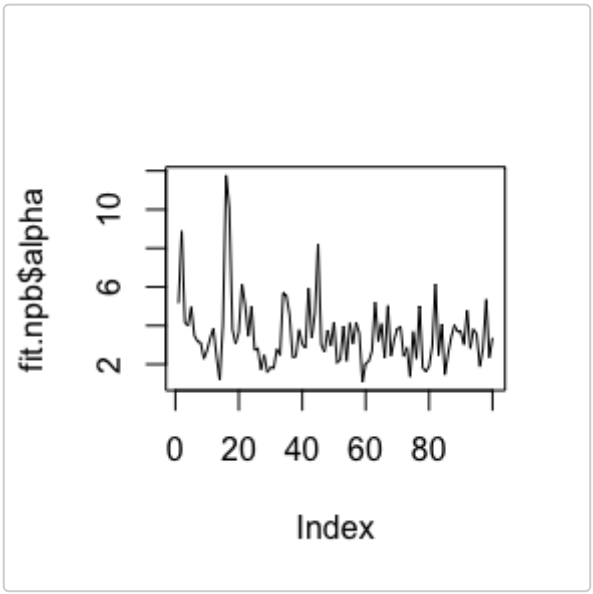
The `predict` function gives the posterior mean fitted values and the posterior distribution of fitted values for each individual (again, see the help file for `predict.npb`). We plot the predicted versus observed values of the response.

```
pred.npb <- predict(fit.npb)
fittedvals <- pred.npb$fitted.vals
plot(fittedvals, Y)
```

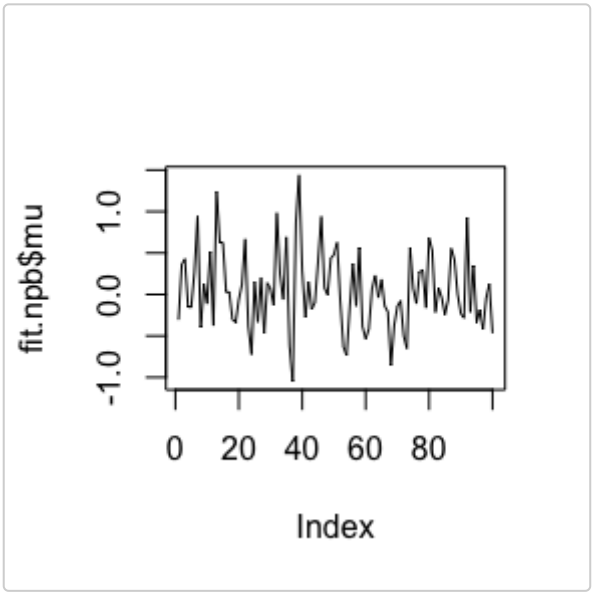


To assess model convergence, we can look at trace plots of the model estimates from `fit.npb`.

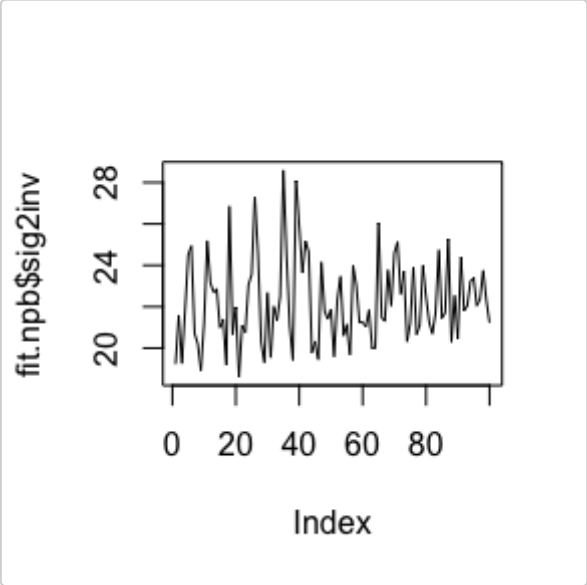
```
plot(fit.npb$alpha, type = "l")
```




```
plot(fit.npb$mu, type = "l")
```




```
plot(fit.npb$sig2inv, type = "l")
```



Bayesian Profile Regression

Bayesian profile regression (BPR) classifies individual profiles, $\mathbf{x}_i$, into a parsimonious set of clusters using a Dirichlet process (DP) mixture model. We implement BPR using a trunctated DP approach to approximate an infinite mixture model with a finite one. Profile regression involves two parts: a profile assignment model that assigns exposure profiles to clusters and a response model that regresses the outcome on cluster indicators to estimate cluster-specific intercepts, or health risks associated with cluster membership.

We offer two variations of BPR: unsupervised (UPR) and supervised (SPR). In UPR, exposure profiles are assigned to clusters without regard to the outcome, while in SPR the outcome informs cluster assignment. Both UPR and SPR can be fit directly using the package mmpack. SPR can also be fit using a wrapper function that utilizes methods in the package PReMiuM, most importantly the `profRegr` function to fit the model. First, we show how to fit BPR with mmpack.

The function `profileReg` fits BPR using MCMC methods and takes in the following arguments: `niter` is the number of total iterations including burn-in; `nburn` is the number of burn-in iterations; `X` is a matrix of predictor data; `Y` is a vector of continuous response data; `W` is a matrix of covariate data; `C` is the maximum number of clusters allowed; `scaleY` indicates if the user wants the response to be scaled prior to model fit; `DPgamma` indicates if the DP parameter α should have a gamma prior, otherwise it will have a Uniform(0.3, 10) prior; `varsel` indicates if variable selection should be implemented; `priors` is an optional list of prior hyperparameters (see below); and `sup` indicates if SPR should be fit, otherwise UPR will be fit.

```
profileReg(niter, nburn, X, Y, W, C = 20, scaleY = FALSE,
           DPgamma = TRUE, varsel = FALSE, priors,
           sup = TRUE)
```

Profile regression includes a profile assignment model and a response model. The profile assignment model involves a mixture of normally distributed clusters and the probability of assignment to each cluster. Thus the likelihood of an exposure profile is a possibly infinite mixture, but here we truncate the mixture with a maximum allowable number of clusters. A truncated stick-breaking prior is placed on the cluster weights. The response is modeled simultaneously.

$$
\begin{aligned}
f(\mathbf{x}_i|\boldsymbol{\psi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) &= \sum_{c=1}^{C} \psi_c f(\mathbf{x}_i|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \\
\mathbf{x}_i|z_i = c, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c &\sim \mathrm{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \\
\boldsymbol{\mu}_c &\sim \mathrm{N}(\boldsymbol{\nu}, \boldsymbol{\Lambda}) \\
\boldsymbol{\Sigma}_c^{-1} &\sim \mathrm{Wish_p}(\mathbf{R}, r) \\
V_1, \ldots, V_{C-1}|\alpha &\sim \mathrm{Beta}(1, \alpha),\ V_C = 1 \\
\alpha &\sim \mathrm{Gamma}(\alpha_\alpha, \beta_\alpha) \text{ or } \mathrm{Unif}(0.3, 10) \\
Pr(z_i = c) = \psi_c &= V_c \prod_{h=1}^{c-1}(1 - V_h) \\
z_i &\sim \mathrm{Categorical}(\boldsymbol{\psi}) \\
\sum_{c=1}^{C} \psi_c &= 1 \text{ a.s.}
\end{aligned}
$$

$$
\begin{aligned}
y_i|z_i = c, \theta_c, \boldsymbol{\gamma}, \sigma^2 &\sim \mathrm{N}(\theta_c + \mathbf{w}_i^T \boldsymbol{\gamma}, \sigma^2) \\
\theta_c|\kappa_c^{-2} &\sim \mathrm{N}(0, \kappa_c^2) \\
\kappa_c^{-2} &\sim \mathrm{Gamma}(\alpha_\kappa, \beta_\kappa) \\
\gamma_j|\phi_j^{-2} &\sim \mathrm{N}(0, \phi_j^2) \\
\phi_j^{-2} &\sim \mathrm{Gamma}(\alpha_\phi, \beta_\phi) \\
\sigma^{-2} &\sim \mathrm{Gamma}(\alpha_\sigma, \beta_\sigma)
\end{aligned}
$$

If variable selection is implemented by choosing `varsel = TRUE`, then we have the following additional model for variable selection:

$$
\begin{aligned}
\mu_{c,j}^* &= \pi_{c,j}\mu_{c,j} + (1 - \pi_{c,j})\bar{x}_j \\
\pi_{c,j}|\rho_j, n_c = 0 &= 0 \\
\pi_{c,j}|\rho_j, n_c > 0 &\sim \mathrm{Ber}(\rho_j) \\
\rho_j|\omega_j &\sim I(\omega_j = 0)\delta_0 + I(\omega_j = 1)\mathrm{Beta}(\alpha_\rho, \beta_\rho) \\
\omega_j &\sim \mathrm{Ber}(0.5),
\end{aligned}
$$

where n_c is the number of individuals assigned to cluster c and $\pi_{c,j}$ is a binary random variable that equals 1 if exposure j is important in assigning individuals to cluster c and 0 otherise. Then, $\boldsymbol{\mu}_c^*$ replaces $\boldsymbol{\mu}$ in the likelihood equation above.

Prior hyperparameters that the user may specify are listed as follows. Default values are given as well as the notation used above (model parameter) and the appropriate name in the list (control parameter). Here, p is the number of columns in $\mathbf{x}$, equivalently the number of predictors.

control parameter	model parameter	default value	description
nu	$\boldsymbol{\nu}$	vector of empirical exposure means	$p \times 1$ vector, mean parameter for normal prior on μ_c
Lambda	$\boldsymbol{\Lambda}$	diag(vector of squared empirical exposure ranges)	$p \times p$ covariance matrix parameter for normal prior of μ_c

control parameter	model parameter	default value	description
R	$\mathbf{R}$	var$(X)^{-1}$/p	scale matrix parameter for Wishart prior on Σ_c^{-1}
r	r	number of exposures	degrees of freedom parameter for Wishart prior on Σ_c^{-1}
alpha.alpha	α_α	2	shape parameter for gamma prior on α
beta.alpha	β_α	1	rate parameter for gamma prior on α
alpha.phi	α_ϕ	3.5	shape parameter for gamma prior on ϕ_j^{-2}
beta.phi	β_ϕ	21.875	rate parameter for gamma prior on ϕ_j^{-2}
alpha.kap	α_κ	3.5	shape parameter for gamma prior on κ_c^{-2}
beta.kap	β_κ	21.875	scale parameter for gamma prior on κ_c^{-2}
alpha.sig	α_σ	2.5	shape parameter for gamma prior on σ^{-2}
beta.sig	β_σ	2.5	scale parameter for gamma prior on σ^{-2}
alpha.rho	α_ρ	0.5	shape1 parameter on beta dist for ρ
beta.rho	β_ρ	0.5	shape2 parameter on beta dist for ρ

We will use the same data as before to demonstrate profile regression. We can change priors by making a list with whichever hyperparameters we wish to specify:

```
priors.bpr <- list(alpha.sig = 1, beta.sig = 1)
```

Now, let's run unsupervised profile regression, letting `sup = FALSE`, with variable selection.

```
fit.upr <- profileReg(niter = 200, nburn = 100, X = X, Y = Y, W = W,
                      scaleY = TRUE, DPgamma = TRUE,
                      varsel = TRUE, priors = priors.bpr, sup = FALSE)
```

We can look at the output using the `summary` function (see the help file for `summary.bpr`). The `summary` function computes model averaged estimated mean health risk for each clustering identified in the best clustering of the data and this information in `exposure.response`. It also computes the mean and standard deviation of the exposures for the individuals assigned to each cluster in `cluster.summary`.

```r
upr.sum <- summary(fit.upr)
upr.sum$exposure.response
```

```
##            mean risk       SD 95% CI Lower 95% CI Upper   n    mean(Y)    SD(Y)
## cluster 1  1.5856604 2.184954    -3.525721    5.7922361   2  6.4642464 3.934673
## cluster 2  5.2304241 1.133977     2.977376    7.3021030  14  4.9453103 7.976587
## cluster 3 -0.7198157 0.300517    -1.225380   -0.1352555 181 -0.8341534 4.723351
## cluster 4 -0.3740698 3.057939    -6.822086    4.2728168   2  1.1238921 7.069886
## cluster 5  5.1443040 4.150285    -4.839474   12.0866315   1 -1.4343919       NA
```

```r
round(upr.sum$cluster.summary,4)
```

```
## , , cluster 1
##
##                mean     SD
## exposure 1 -0.1339 0.0369
## exposure 2 -0.9304 0.8860
## exposure 3 -0.0808 0.4836
## exposure 4  0.2890 0.7369
## exposure 5  1.8337 0.2792
## exposure 6 -0.5805 0.3453
##
## , , cluster 2
##
##                mean     SD
## exposure 1  0.5381 0.8001
## exposure 2 -0.0863 0.8773
## exposure 3 -0.2807 1.2374
## exposure 4  2.3227 2.4781
## exposure 5 -0.5786 0.5987
## exposure 6 -0.2028 0.5647
##
## , , cluster 3
##
##                mean     SD
## exposure 1 -0.1527 0.0549
## exposure 2  0.0169 0.9946
## exposure 3  0.0229 0.9836
## exposure 4 -0.2205 0.0599
## exposure 5  0.0227 1.0047
## exposure 6  0.0117 1.0065
##
## , , cluster 4
##
##                mean     SD
## exposure 1  9.1169 3.3193
## exposure 2 -1.1683 0.1763
## exposure 3  0.6794 0.7728
## exposure 4  0.6092 1.1896
## exposure 5 -0.2953 0.2064
## exposure 6 -0.6363 0.6204
```

```
##
## , , cluster 5
##
##                mean SD
## exposure 1  2.1379 NA
## exposure 2  2.3445 NA
## exposure 3 -1.4143 NA
## exposure 4  5.5967 NA
## exposure 5  0.9151 NA
## exposure 6  3.1486 NA
```

We can also look at the posterior inclusion probabilities for each parameter:

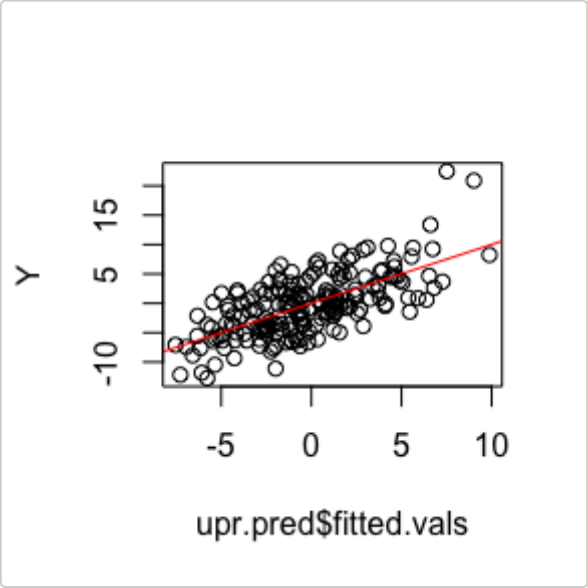
```
round(upr.sum$rho,2)
```

```
## [1] 0.43 0.02 0.03 0.10 0.05 0.04
```

Other attributes include subject-specific risks and summaries of risk, the best clustering of the data, and which individuals are in each group in the best clustering.

We can use the `predict` function (see help file for `predict.bpr`) to see the posterior mean fitted values for each subject and plot them against the true response.

```
upr.pred <- predict(fit.upr)
plot(upr.pred$fitted.vals, Y)
abline(0, 1, col = "red")
```



The package mmpack also includes a wrapper function, named `premium_wrapper`, to fit SPR using the PReMiuM software. We use the same arguments in `premium_wrapper` as we do in `profileReg` with some minor differences. First, the argument `varSelectType` must be specificed as either `None` for no variable selection or `BinaryCluster` to implement binary cluster variable selection. The user can also specify `simnum` if they want to fit `premium_wrapper` more than once and not overwrite the results. The text file output will then be stored with `simnum` in the file name. The user can also set a seed using the argument `seed`. Last, priors is again an optional list, but the specification is a bit different than in `profileReg`.

```
premium_wrapper(niter, nburn, Y, X, W, scaleY = FALSE,
  varSelectType = "None", simnum = NULL, priors, seed = NULL)
```

See the function documentation for `setHyperparams` in the R package PReMiuM for details on prior specification. The priors used in `premium_wrapper` include the following, with their mmpack equivalent notation (if applicable), default values, and description from the PReMiuM package documentation.

control parameter	mmpack equivalent model parameter	default value	description
shapeAlpha	α_α	2	shape parameter for Gamma prior on alpha
rateAlpha	α_β	1	inverse-scale (rate) parameter for the Gamma prior on alpha
muTheta	NA	0	location parameter for the t-Distribution for θ_c
sigmaTheta	NA	2.5	scale parameter for the t-Distribution for θ_c
dofTheta	NA	7	degrees of freedom parameter for the t-Distribution for θ_c
muBeta	NA	0	location parameter for the t-Distribution for beta
sigmaBeta	NA	2.5	scale parameter for the t-Distribution for beta
dofBeta	NA	7	dof parameter for the t-Distribution for beta
shapeSigmaSqY	α_σ	2.5	shape parameter of inverse-gamma prior for σ_Y^2
scaleSigmaSqY	β_σ	2.5	scale parameter of inverse-gamma prior for σ_Y^2
aRho	α_ρ	0.5	parameter for beta distribution for prior on rho in variable selection
bRho	β_ρ	0.5	parameter for beta distribution for prior on rho in variable selection
mu0	$\boldsymbol{\nu}$	vector of empirical covariate means	mean vector for mu_c in the Normal covariate case

control parameter	mmpack equivalent model parameter	default value	description
Tau0	$\mathbf{\Lambda}^{-1}$	inverse of diag(vector of squared empirical covariate ranges)	precision matrix for mu_c in the Normal covariate case
R0	$\mathbf{R}$	$\text{var}(X)^{-1}/p$	scale parameter for the Wishart distribution for Tau_c if useHyperpriorR1=FALSE in the function profRegr
kappa0	r	p	degrees of freedom for the Wishart distribution for Tau_c if useHyperpriorR1=FALSE in the function profRegr

Let's set some priors and fit the SPR model using `premium_wrapper`. The `premium_wrapper` function requires that we create a folder call "Premium_output" for storing the .txt file output. First, create this folder in your working directory and then run the following lines.

```
priors.prem <- list(shapeSigmaSqY = 1, scaleSigmaSqY = 1)
fit.prem <- premium_wrapper(niter = 200, nburn = 100, Y = Y, X = X, W = W,
              scaleY = FALSE, varSelectType = "BinaryCluster", simnum = 1,
              priors = priors.prem, seed = 1234)
```

The `premium_wrapper` function returns output similar to that of `profileReg` plus some other output. For example, we can calculate posterior inclusion probabiltiies as the mean posterior probability of inclusion (rho) for each exposure. We can also look at a summary of the exposure-response function (risk) for each cluster.

```
round(apply(fit.prem$rho,2, mean),2)
```

```
fit.prem$exposure.response
```

You can also retrieve the original model fit, an object of type `runInfoObj` and utilize functions in the package PReMiuM as desired. Or for even more options, you can fit SPR directly through PReMiuM. To do so, see Liverani et al (2015).

```
runInfoObj <- fit.prem$fit
```

Bayesian Kernel Machine Regression

Finally, mmpack includes a wrapper function to fit Bayesian Kernel Machine Regression (BKMR). The function `bkmr_wrapper` takes in the following arguments: `niter` is the total number of iterations, `nburn` is the number of burn-in iterations, `Y` is a vector of response data, `X` is a matrix of exposure data, `W` is a matrix of covariate data, `varsel` indicates if variable selection should be implemented, and `groups` is an optional vector of group membership for exposures if the user wants to implement the hierarchical variable selection option.

```
bkmr_wrapper <- function(niter, nburn, Y, X, W, varsel = FALSE, groups = NULL)
```

For example, suppose we want to perform hierarchical variable selection, grouping the exposures by type: pesticides (C, OP) and air pollutants (NO_2, O_3, PM_{10}, $PM_{2.5}$).

```
colnames(X)
```

```
## [1] "C"     "NO2"   "O3"    "OP"    "PM10"  "PM2.5"
```

```
groups <- c(1,2,2,1,2,2)
```

Then we can fit BKMR with hierarchical variable selection.

```
fit.bkmr <- bkmr_wrapper(niter = 200, nburn = 100, Y = Y, X = X, W = W, varsel = TRUE,
                         groups = groups)
```

```
## Warning in if (class(Znew) == "data.frame") Znew <- data.matrix(Znew): the
## condition has length > 1 and only the first element will be used
```

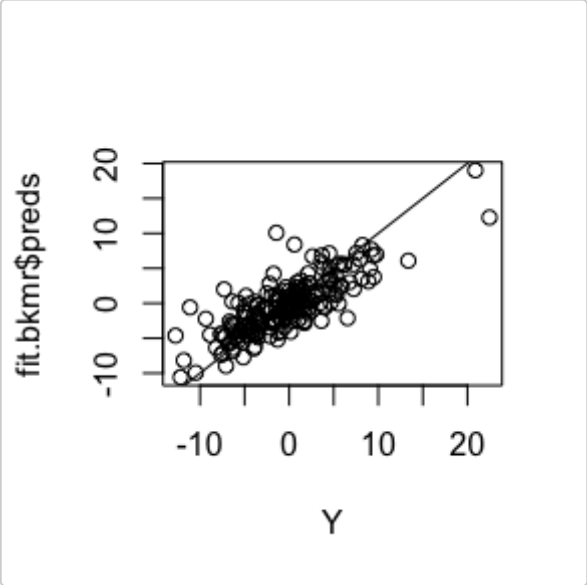
```
## Warning in if (class(Xnew) != "matrix") Xnew <- matrix(Xnew, nrow = 1): the
## condition has length > 1 and only the first element will be used
```

```
## Warning in if (class(Znew) == "data.frame") Znew <- data.matrix(Znew): the
## condition has length > 1 and only the first element will be used
```

```
## Warning in if (class(Xnew) != "matrix") Xnew <- matrix(Xnew, nrow = 1): the
## condition has length > 1 and only the first element will be used
```

The function `bkmr_wrapper` includes some post-processing methods. We can get the predicted values and plot them against the observed response:

```
plot(Y, fit.bkmr$preds)
abline(0,1)
```



We can also see the group and conditional posterior inclusion probabilities (PIP) for exposures. Recall that group 1 is for pesticides and group 2 is for air pollutants. Component-wise PIPs can be calculated by multiplying the group PIP by the conditional PIP for each exposure.

```
round(fit.bkmr$group.pips,2)
```

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

```
round(fit.bkmr$pips,2)
```

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

The wrapper function also returns the original model fit. For more options, you can use the R package bkmr directly. Reference https://jenfb.github.io/bkmr/overview.html for guided examples on fitting the model and/or more detailed post-processing.

```
original.fit <- fit.bkmr$fit
```

Reproduce Simulation

Finally, we provide an example of how to use the package mmpack to reproduce simulation results. We use iid N(0,1) data for exposures (X) and covariates (W) in this example. We use the functions `simLinearResponse`, `simNonlinearResponse`, and `simProfilesResponse` to simulate response data as linear, nonlinear, or fixed profiles (piece-wise constant) functions of the predictor data, respectively. In each simulation we generate new response data and the exposure-response function includes a new random subset of the exposures. We use the function `fitModels` to fit each model, post-process the output, and summarize the evaluation criteria. For demonstration we show 5 simulations and each method is run for 20 iterations with a burn-in of 10 iterations. The number of iterations and simulations will need to be increased for any practical application of this example. Last we use the function `summarizeSimulation` to summarize the method performance across all simulated data sets and show how to present the results in a nicely formatted LaTeX table.

```r
set.seed(12345)
n <- 100
p <- 7
q <- 10
X <- matrix(rnorm(n*p), n, p)
W <- matrix(rnorm(n*q), n, q)
nsims <- 5 # number of simulations
niter <- 20 # number of iterations
nburn <- 10 # burn-in
df <- NULL
for(i in 1:nsims){
  # set new seed each time
  simnum <- i
  seed <- 5*simnum
  # simulate responses from each scenario
  lin <- simLinearResponse(X, W)
  nonlin <- simNonlinearResponse(X, W)
  prof <- simProfilesResponse(X, W)
  # fit and evaluate models
  df.lin <- fitModels(names = "lin", simnum = simnum, niter = niter, nburn = nburn,
                      X = X, W = W, data = lin, seed = seed)
  df.nonlin <- fitModels(names = "nonlin", simnum = simnum, niter = niter, nburn = nburn,
                         X = X, W = W, data = nonlin, seed = seed)
  df.prof <- fitModels(names = "prof", simnum = simnum, niter = niter, nburn = nburn,
                       X = X, W = W, data = prof, seed = seed)

  df.new <- rbind(df.lin, df.nonlin, df.prof)
  df <- rbind(df, df.new)
}
```

The following code produces a data frame of the simulation results for each method in each scenario. The columns include method, root mean squared error (RMSE), coverage (Cvg), true selection rate for main effects (tsr), false selection rate for main effects (fsr), true selection rate for interactions (tsr.int), and false selection rate for interactions (fsr.int).

```r
require(xtable)
sumSim <- summarizeSimulation(df = df)
print(xtable(sumSim, caption = "Simulation results", digits = 2, align =
        c("l","l",rep("r",7))),
        comment = FALSE, include.rownames = FALSE, caption.placement = "top")
```