bayespca Package

Davide Vidotto d.vidotto@uvt.nl

2019-06-03

bayespca: A package for Variational Bayes PCA

Theoretical background

Principal Components Analysis (PCA) allows performing dimensionality reduction via matrix factorization. While there are several ways to express a PCA model, in what follows will we consider the formulation

$$X = XWP^T + E,$$

where X is a $I \times J$ data matrix (I is the number of units; J the number of continuous variables); W is a $J \times D$ weight matrix ($D \leq J$ is the rank of the reduced matrix); P is the orthogonal loading matrix, such that $P^T P = I_{D \times D}$; and E is an $I \times J$ error matrix. The D principal components can be retrieved with $Z = XW$. In this context, the focus of the inference is typically on W. In particular, when J is large and the main inferential goal is components' interpretation, it is important for the analyst to obtain simple and interpretable components.

The `bayespca` package allows performing the following operations:

1. estimation of the PCA model, with a Variational Bayes algorithm;

2. regularization of the elements of W by means of its prior variances;
3. variable selection, via a Stochastic Search Variable Selection method (a form of "spike-and-slab" prior).

The Variational Bayes algorithm sees the columns of W as latent variables, and P as a fixed parameter. Furthermore, the residuals E are assumed to be distributed according to a Normal distribution with mean 0 and variance σ^2. The following prior is assumed for the d-th column of W:

$$w_d \sim MVN(0, T_d)$$

where $MVN()$ denotes the density of the Multivariate Normal Matrix, and T_d denotes the prior (diagonal) covariance matrix of the d-th component. The j-th element of the diagonal of T_d will be denoted τ_{dj}. where $MVN()$ denotes the density of the Multivariate Normal Matrix, and T_d denotes the prior (diagonal) covariance matrix of the d-th component. The j-th element of the diagonal of T_d will be denoted τ_j.

The `bayespca` package

Variational Bayes PCA is implemented through the `vbpca` function, which takes the following arguments as inputs:

- `X` the input matrix;
- `D` the number of components to be estimated;
- `maxIter` the maximum number of iterations for the Variational Bayes algorithm;
- `tolerance` convergence criterion of the algorithm (relative difference between ELBO values);
- `verbose` logical parameter which prints estimation information on screen when `TRUE`;
- `tau` value of the prior variances; starting value when `updatetau=TRUE` or `priorvar!='fixed'`
- `updatetau` logical parameter denoting whether the prior variances should be updated when `priorvar='fixed'`;

- `priorvar` character argument denoting whether the prior variances should be `'fixed'`, or random with `'jeffrey'` or `'invgamma'` priors;

- `SVS` logical argument which activates Stochastic Variable Selection when set to `TRUE`;
- `priorInclusion` prior inclusion probabilities for the elements of W in the model;
- `global.var` logical parameter which activates component-specific prior variances when set to `TRUE`;
- `control` other control parameters, such as Inverse Gamma hyperparameters (see `?vbpca_control` for more information).

`vbpca` returns a vbpca object, which is a list containing various aspect of the model results. See `?vbpca` for further information. Internally, `vbpca` calls a C++ function (written with Rcpp) to estimate the model.

In what follows, the various estimation modalities allowed by `vbpca` will be introduced. For presentation purposes, a synthetic data matrix with $I = 100$ rows and $J = 20$ columns genereted from three components will be used:

```
set.seed(141)
I <- 100
J <- 20
V1 <- rnorm(I, 0, 50)
V2 <- rnorm(I, 0, 30)
V3 <- rnorm(I, 0, 10)
X <- matrix(c(rep(V1, 7), rep(V2, 7), rep(V3, 6)), I, J)
X <- X + matrix(rnorm(I * J, 0, 1), I, J)
```

I will now proceed with the estimation of the PCA model.

Levels of regularization on the W matrix

Fixed `tau`

With fixed tau, it is possible to specify the model as follows:

```
# Install and load package
# devtools::install_github("davidevdt/bayespca")
library(bayespca)


# De-activate data center and scaling;
ctrl <- vbpca_control(center = FALSE, scalecorrection = -1,
                      plot.lowerbound = FALSE)


# Estimate vbpca with fixed prior variances (equal to 1)
# for the elements of W
mod1 <- vbpca(X, D = 3, maxIter = 1e+03, priorvar = 'fixed',
              control = ctrl, verbose = FALSE )


# Test the class of mod1:
is.vbpca(mod1)
```

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

The estimate posterior means of the W matrix can be viewed with:

```r
mod1$muW
```

```
##        Component 1 Component 2   Component 3
##  [1,] -0.376589698 -0.04416511  0.0003399127
##  [2,] -0.373939776 -0.04582346 -0.0111489577
##  [3,] -0.375148656 -0.04305857 -0.0078831833
##  [4,] -0.374770076 -0.04473100 -0.0031124936
##  [5,] -0.376808025 -0.04285791 -0.0100250665
##  [6,] -0.375114064 -0.04446329 -0.0015012122
##  [7,] -0.375069345 -0.04364081 -0.0007181137
##  [8,]  0.043916073 -0.37610684 -0.0194987814
##  [9,]  0.044338996 -0.37382689 -0.0224165501
## [10,]  0.043216238 -0.37319455 -0.0161965551
## [11,]  0.043432789 -0.37311089 -0.0246530479
## [12,]  0.045420158 -0.37574266 -0.0200072027
## [13,]  0.045158091 -0.37616395 -0.0206149535
## [14,]  0.044605650 -0.37571347 -0.0144837510
## [15,]  0.002905219  0.02229238 -0.4057459196
## [16,]  0.003409761  0.02199152 -0.4068881251
## [17,]  0.003232844  0.02063894 -0.4106993259
## [18,]  0.002919709  0.02319335 -0.4056784549
## [19,]  0.002019259  0.02192116 -0.4088023613
## [20,]  0.001874207  0.02043128 -0.4078307380
```

and the P matrix:

```r
mod1$P
```

```
##        Component 1 Component 2   Component 3
##  [1,] -0.376589904 -0.04416517  0.0003399179
##  [2,] -0.373939981 -0.04582353 -0.0111491289
##  [3,] -0.375148862 -0.04305863 -0.0078833043
##  [4,] -0.374770282 -0.04473106 -0.0031125414
##  [5,] -0.376808232 -0.04285797 -0.0100252205
##  [6,] -0.375114270 -0.04446335 -0.0015012352
##  [7,] -0.375069551 -0.04364087 -0.0007181247
##  [8,]  0.043916097 -0.37610735 -0.0194990808
##  [9,]  0.044339020 -0.37382740 -0.0224168943
## [10,]  0.043216262 -0.37319506 -0.0161968038
## [11,]  0.043432813 -0.37311139 -0.0246534264
## [12,]  0.045420183 -0.37574317 -0.0200075099
## [13,]  0.045158115 -0.37616446 -0.0206152700
## [14,]  0.044605675 -0.37571398 -0.0144839734
## [15,]  0.002905220  0.02229241 -0.4057521497
## [16,]  0.003409762  0.02199155 -0.4068943728
## [17,]  0.003232846  0.02063897 -0.4107056321
## [18,]  0.002919710  0.02319338 -0.4056846840
## [19,]  0.002019260  0.02192119 -0.4088086384
## [20,]  0.001874208  0.02043131 -0.4078370001
```

Among other things, the function returns also the model evidence lower bound (ELBO) and the estimation time:

```r
mod1$elbo
```

```
## [1] -2834.329
```

```r
mod1$time
```

```
##    user  system elapsed 
##       0       0       0
```

Fixed, updatable tau

The prior variances τ_{dj} can also be updated via Type-II Maximum Likelihood (empirical Bayes updates):

```r
mod2 <- vbpca(X, D = 3, maxIter = 1e+03, priorvar = 'fixed',
              updatetau = TRUE, control = ctrl, verbose = FALSE )

mod2$muW
```

```
##         Component 1  Component 2  Component 3
##  [1,] -3.770417e-01 -0.051573499 -0.001788633
##  [2,] -3.747997e-01 -0.025120541 -0.002279085
##  [3,] -3.749896e-01 -0.039169668 -0.002284711
##  [4,] -3.695986e-01 -0.062446420 -0.002188222
##  [5,] -3.798267e-01 -0.031100065 -0.002172447
##  [6,] -3.809607e-01 -0.058646547 -0.001863930
##  [7,] -3.707615e-01 -0.037076450 -0.001858450
##  [8,]  4.610643e-02 -0.388243166 -0.019761887
##  [9,]  4.122547e-02 -0.376699340 -0.023207406
## [10,]  2.542203e-02 -0.376987920 -0.006935656
## [11,]  5.233439e-02 -0.374340987 -0.022206305
## [12,]  4.536997e-02 -0.374395905 -0.013879307
## [13,]  6.378896e-02 -0.370090650 -0.031591876
## [14,]  3.106493e-02 -0.364598986 -0.002519184
## [15,]  5.847012e-06  0.006198051 -0.406003865
## [16,]  6.041405e-06  0.018464891 -0.407039992
## [17,]  5.235991e-06  0.009013294 -0.411151002
## [18,]  3.518425e-06  0.036121023 -0.398782724
## [19,]  6.288835e-06  0.005824170 -0.410563013
## [20,]  6.214579e-06  0.034945874 -0.412808540
```

The matrix of the inverse prior variances can be called with

```r
mod2$invTau
```

```
##        Component 1 Component 2  Component 3
##  [1,] 6.723601e+00  184.301093 31906.370474
##  [2,] 6.797876e+00  437.595911 25321.510591
##  [3,] 6.720477e+00  245.899700 25234.633828
##  [4,] 6.909797e+00  136.385519 26302.854238
##  [5,] 6.614756e+00  336.071862 26671.584534
##  [6,] 6.538464e+00  150.961175 30414.866197
##  [7,] 6.920149e+00  278.098093 30606.846239
##  [8,] 1.973803e+02    6.278101   670.761526
##  [9,] 2.288287e+02    6.657350   569.759379
## [10,] 4.149292e+02    6.642324  1989.950540
## [11,] 1.699204e+02    6.739745   597.851184
## [12,] 2.026152e+02    6.706750   970.392618
## [13,] 1.292935e+02    6.822902   390.944658
## [14,] 3.536290e+02    7.210202  5658.371089
```

```
## [15,] 3.158381e+05 2396.464633     5.835080
## [16,] 3.162767e+05  757.091273     5.769939
## [17,] 3.264009e+05 1565.810039     5.674352
## [18,] 3.196566e+05  348.615143     5.995065
## [19,] 3.198439e+05 2526.927990     5.707424
## [20,] 3.182544e+05  363.292431     5.588104
```

Random `tau`: Jeffrey's prior

By assuming Jeffrey's hyperpriors on $\tau_{d,j}$ we set:

$$p(\tau_{d,j}) \propto \frac{1}{\tau_{d,j}}.$$

The following code runs the algorithm with Jeffrey's priors on `tau`:

```r
mod3 <- vbpca(X, D = 3, maxIter = 1e+03,
              priorvar = 'jeffrey', control = ctrl, verbose = FALSE )


mod3$muW
```

```
##         Component 1  Component 2  Component 3
##  [1,] -3.757646e-01 -0.051773159 -0.001684598
##  [2,] -3.756863e-01 -0.024773150 -0.002122048
##  [3,] -3.758101e-01 -0.039042545 -0.002125161
##  [4,] -3.692192e-01 -0.062594644 -0.002042117
##  [5,] -3.808786e-01 -0.030957878 -0.002028616
##  [6,] -3.802877e-01 -0.058870639 -0.001748039
##  [7,] -3.703552e-01 -0.036931472 -0.001746713
##  [8,]  4.721080e-02 -0.388387203 -0.019681028
##  [9,]  4.143813e-02 -0.376808913 -0.023157487
## [10,]  2.659771e-02 -0.377040717 -0.006727279
## [11,]  5.011560e-02 -0.374286630 -0.022163915
## [12,]  4.536128e-02 -0.374432238 -0.013520621
## [13,]  6.304093e-02 -0.370068016 -0.031431258
## [14,]  3.134111e-02 -0.364405737 -0.002414999
## [15,] -1.171811e-05  0.005936644 -0.406010842
## [16,] -1.146225e-05  0.018071121 -0.406913644
## [17,] -1.192092e-05  0.008654023 -0.411149650
## [18,] -1.359439e-05  0.036363148 -0.398751160
## [19,] -1.127322e-05  0.005569321 -0.410649936
## [20,] -1.140226e-05  0.034939861 -0.412915798
```

```r
mod3$invTau
```

```
##        Component 1 Component 2  Component 3
##  [1,] 6.767590e+00  183.413238 35792.389912
##  [2,] 6.767276e+00  444.807011 28717.737791
##  [3,] 6.692741e+00  246.914789 28653.864296
##  [4,] 6.923187e+00  135.973350 29770.336711
##  [5,] 6.579887e+00  337.917428 30166.978271
##  [6,] 6.560514e+00  150.229799 34271.225136
##  [7,] 6.934665e+00  279.497090 34407.243772
##  [8,] 1.915903e+02    6.273751   676.164203
```

```
##  [9,] 2.275212e+02    6.653730   573.008331
## [10,] 3.946490e+02    6.640583  2059.789146
## [11,] 1.794492e+02    6.741653   601.092050
## [12,] 2.026344e+02    6.705600  1000.451289
## [13,] 1.312116e+02    6.823768   394.628449
## [14,] 3.499053e+02    7.217607  5917.649066
## [15,] 3.510386e+05 2512.207936     5.834906
## [16,] 3.515804e+05  778.039952     5.773433
## [17,] 3.627243e+05 1637.770077     5.674439
## [18,] 3.549667e+05  347.219129     5.996028
## [19,] 3.555498e+05 2652.878686     5.705122
## [20,] 3.539059e+05  364.874639     5.585392
```

Random `tau`: Inverse Gamma prior

It is possible to specify an inverse gamma prior on $\tau_{d,j}$:

$$\tau_{d,j} \sim IG(\alpha, \beta)$$

with α shape parameter and β scale parameter. The following code implements an IG(2, .5) prior on the variances:

```r
# Set hyperparameter values
ctrl2 <- vbpca_control(center = FALSE, scalecorrection = -1,
                       plot.lowerbound = FALSE,
                       alphatau = 2, betatau = .5)



# Estimate the model
mod4 <- vbpca(X, D = 3, maxIter = 1e+03, priorvar = 'invgamma',
              control = ctrl2, verbose = FALSE )


mod4$muW
```

```
##        Component 1 Component 2   Component 3
##  [1,] -0.376590647 -0.04416831  0.0002942783
##  [2,] -0.373916145 -0.04580796 -0.0111136115
##  [3,] -0.375152370 -0.04306335 -0.0078482119
##  [4,] -0.374771589 -0.04473679 -0.0031289018
##  [5,] -0.376819230 -0.04285286 -0.0099901143
##  [6,] -0.375128398 -0.04445794 -0.0015202569
##  [7,] -0.375056419 -0.04365149 -0.0007404096
##  [8,]  0.043918311 -0.37612413 -0.0195192789
##  [9,]  0.044336971 -0.37380675 -0.0224249935
## [10,]  0.043215376 -0.37316095 -0.0162247651
## [11,]  0.043435543 -0.37309948 -0.0245954650
## [12,]  0.045416531 -0.37575287 -0.0200104086
## [13,]  0.045161134 -0.37621159 -0.0206043931
## [14,]  0.044602975 -0.37569136 -0.0144839228
## [15,]  0.002901088  0.02228831 -0.4056882953
## [16,]  0.003405222  0.02198276 -0.4068836318
## [17,]  0.003228802  0.02064986 -0.4107113728
```

```
## [18,]  0.002920826  0.02319232 -0.4056318015
## [19,]  0.002018519  0.02191736 -0.4087805942
## [20,]  0.001885975  0.02043594 -0.4078297610
```

```r
mod4$invTau
```

```
##       Component 1 Component 2 Component 3
##  [1,]    4.349513    4.952224    4.961845
##  [2,]    4.357094    4.951479    4.961185
##  [3,]    4.348866    4.946652    4.955442
##  [4,]    4.346970    4.942133    4.951918
##  [5,]    4.346201    4.949399    4.957941
##  [6,]    4.348419    4.945415    4.955085
##  [7,]    4.353566    4.952076    4.961437
##  [8,]    4.940075    4.341259    4.947637
##  [9,]    4.943619    4.350731    4.950784
## [10,]    4.946272    4.354243    4.954138
## [11,]    4.941374    4.350682    4.947635
## [12,]    4.939956    4.342697    4.948068
## [13,]    4.934268    4.336865    4.942105
## [14,]    4.953804    4.353473    4.962576
## [15,]    4.963713    4.961313    4.266473
## [16,]    4.954953    4.952599    4.256391
## [17,]    4.956272    4.954215    4.246050
## [18,]    4.953989    4.951371    4.259347
## [19,]    4.962076    4.959717    4.256072
## [20,]    4.950147    4.948089    4.249971
```

`alphatau` and `betatau` can also be specified as D-dimensional array, in which case the Inverse Gamma will have component-specific hyperparameters:

$$\tau_{d,j} \sim IG(\alpha_d, \beta_d)$$

.

```r
# Set hyperparameter values
ctrl3 <- vbpca_control(center = FALSE, scalecorrection = -1,
                       plot.lowerbound = FALSE,
                       alphatau = c(.5, 50, 3), betatau = c(.5, .01, 10),
                       hypertype = 'component')


# Estimate the model
mod5 <- vbpca(X, D = 3, maxIter = 1e+03, priorvar = 'invgamma',
              control = ctrl3, verbose = FALSE )


mod5$muW
```

```
##        Component 1  Component 2   Component 3
##  [1,] -0.378543478 -0.022403852  0.0025386295
##  [2,] -0.376019722 -0.021196983 -0.0088612115
##  [3,] -0.377066494 -0.022557126 -0.0057270031
##  [4,] -0.376774699 -0.022650241 -0.0008814042
##  [5,] -0.378720012 -0.022357681 -0.0078768015
##  [6,] -0.377102619 -0.019528818  0.0007032582
##  [7,] -0.376989874 -0.026248142  0.0014710887
```

```
##  [8,]  0.021470521 -0.045359009 -0.0012343390
##  [9,]  0.022018693 -0.049787441 -0.0042453561
## [10,]  0.020952041 -0.037610876  0.0018996414
## [11,]  0.021150966 -0.048838642 -0.0065159768
## [12,]  0.022990213 -0.048959831 -0.0017490218
## [13,]  0.022694280 -2.350959262 -0.0024775177
## [14,]  0.022197501 -0.048663411  0.0037657268
## [15,]  0.002988814  0.002048412 -0.4063509956
## [16,]  0.003472519 -0.001954198 -0.4074615231
## [17,]  0.003201167  0.008962963 -0.4112475488
## [18,]  0.003059452  0.006509927 -0.4063451138
## [19,]  0.002074293  0.003760222 -0.4093932228
## [20,]  0.001853458 -0.003935007 -0.4083201342
```

```
mod5$invTau
```

```
##       Component 1 Component 2 Component 3
##  [1,]    1.734909  4885.51101   0.3498302
##  [2,]    1.737796  4897.91867   0.3498294
##  [3,]    1.734283  4883.57903   0.3498025
##  [4,]    1.733024  4882.65984   0.3497829
##  [5,]    1.733356  4885.97064   0.3498127
##  [6,]    1.733921  4914.16337   0.3497986
##  [7,]    1.736550  4841.25146   0.3498290
##  [8,]    1.974069  4535.17321   0.3497716
##  [9,]    1.975981  4450.34973   0.3497902
## [10,]    1.977182  4671.07826   0.3498009
## [11,]    1.974679  4468.90214   0.3497765
## [12,]    1.974249  4466.51594   0.3497748
## [13,]    1.971273    18.20417   0.3497460
## [14,]    1.981100  4472.66011   0.3498376
## [15,]    1.982219  5007.34354   0.3469767
## [16,]    1.977733  5007.36672   0.3469196
## [17,]    1.978365  4988.53185   0.3468718
## [18,]    1.977266  4997.70264   0.3469310
## [19,]    1.981324  5004.97797   0.3469253
## [20,]    1.975283  5004.39805   0.3468847
```

Notice the different level of regularization obtained across the different components. In order to activate these 'component-specific' hyperpriors, `hypertype = 'component'` was specified.

Random tau, random betatau

It is also possible to specify a Gamma hyperprior on β (while α remains fixed):

$$\beta \sim Ga(\gamma, \delta).$$

This is achievable by setting `gammatau` (and `deltatau`) larger than 0 in the control parameters:

```
# Specify component-specific Gamma(.01, 10) hyperpriors on betatau
ctrl4 <- vbpca_control(center = FALSE, scalecorrection = -1,
                       plot.lowerbound = FALSE,
                       alphatau = 1, betatau = 1,
                       gammatau = .01, deltatau = 10,
                       hypertype = 'component')
```

```r
# Estimate the model
mod6 <- vbpca(X, D = 3, maxIter = 1e+03, priorvar = 'invgamma',
              control = ctrl4, verbose = FALSE )


mod6$muW
```

```
##         Component 1 Component 2  Component 3
##  [1,] -0.376611437 -0.04414830 -0.001252671
##  [2,] -0.373522305 -0.04527018 -0.009487805
##  [3,] -0.375303126 -0.04330237 -0.006687095
##  [4,] -0.374545174 -0.04476030 -0.003779739
##  [5,] -0.377135323 -0.04290004 -0.008530456
##  [6,] -0.375471472 -0.04449079 -0.002519439
##  [7,] -0.374841518 -0.04378470 -0.001756540
##  [8,]  0.044091734 -0.37646021 -0.020075478
##  [9,]  0.044279171 -0.37340282 -0.022181983
## [10,]  0.043230863 -0.37275698 -0.017353060
## [11,]  0.043633682 -0.37285414 -0.022657108
## [12,]  0.045225592 -0.37597481 -0.020153766
## [13,]  0.045217385 -0.37690958 -0.020321214
## [14,]  0.044311525 -0.37546975 -0.014960798
## [15,]  0.002829728  0.02204140 -0.405116611
## [16,]  0.003208895  0.02187057 -0.407344521
## [17,]  0.003155312  0.02095411 -0.410909109
## [18,]  0.002824576  0.02288662 -0.405229626
## [19,]  0.002136107  0.02174443 -0.408747297
## [20,]  0.002177548  0.02077097 -0.407952211
```

```r
mod6$invTau
```

```
##       Component 1 Component 2 Component 3
##  [1,]    14.89847    49.95597    64.79114
##  [2,]    15.06380    49.85157    64.48104
##  [3,]    14.87935    49.45351    63.73992
##  [4,]    14.89953    49.19672    63.74737
##  [5,]    14.83146    49.84022    64.32954
##  [6,]    14.87624    49.45017    63.91513
##  [7,]    14.97986    49.94842    64.56200
##  [8,]    48.95314    14.77489    62.76748
##  [9,]    49.16960    14.97374    62.96668
## [10,]    49.41724    15.02556    63.45826
## [11,]    49.04970    14.98706    62.69018
## [12,]    48.92027    14.79264    62.77075
## [13,]    48.53974    14.68576    62.16584
## [14,]    50.07132    15.00763    64.71429
## [15,]    51.73889    51.41043    14.14011
## [16,]    51.12230    50.69433    13.94199
## [17,]    51.27494    50.94174    13.77126
## [18,]    51.01890    50.58728    14.03940
## [19,]    51.71281    51.34808    13.93690
## [20,]    50.73849    50.38297    13.85877
```

The posterior means of β can be accessed via

```r
mod6$priorBeta
```

```
##             [,1]       [,2]       [,3]
## [1,] 0.02643281 0.02638618 0.02076472
## attr(,"names")
## [1] "beta 1" "beta 2" "beta 3"
```

`hypertype` specify the type of hyperprior for `beta`:

- `'common'` implies $\beta \sim Ga(\alpha, \beta)$;
- `'component'` implies $\beta_d \sim Ga(\alpha_d, \beta_d)$;
- `'local'` implies $\beta_{dj} \sim Ga(\alpha_{dj}, \beta_{dj})$.

Similar to `alphatau` and `betatau`, `gammatau` and `deltatau` can also be D-dimensional arrays for component-specific hyperpriors on β.

Global prior variances

So far, the parameter `global.var` has always ben set to `FALSE`, implying

$$w_{j,d} \sim N(0, \tau_{j,d}).$$

Setting `global.var = TRUE` will modify this formulation, which will switch to

$$w_{j,d} \sim N(0, \tau_d)$$

that is, component-specific variances (called 'global variances' in `vbpca`) will be estimated instead:

```r
# Fixed prior global variances, updated via Type-II maximum likelihood:
mod7 <- vbpca(X, D = 3, maxIter = 1e+03, priorvar = 'fixed',
              updatetau = TRUE, control = ctrl, verbose = FALSE,
              global.var = TRUE)
```

```r
mod7$muW
```

```
##        Component 1 Component 2   Component 3
##  [1,] -0.376586344 -0.04416414  0.0003398280
##  [2,] -0.373936445 -0.04582246 -0.0111461792
##  [3,] -0.375145315 -0.04305763 -0.0078812187
##  [4,] -0.374766738 -0.04473002 -0.0031117179
##  [5,] -0.376804669 -0.04285697 -0.0100225682
##  [6,] -0.375110723 -0.04446231 -0.0015008380
##  [7,] -0.375066004 -0.04363985 -0.0007179347
##  [8,]  0.043915682 -0.37609858 -0.0194939220
##  [9,]  0.044338601 -0.37381868 -0.0224109636
## [10,]  0.043215853 -0.37318635 -0.0161925187
## [11,]  0.043432402 -0.37310269 -0.0246469040
## [12,]  0.045419753 -0.37573441 -0.0200022167
## [13,]  0.045157688 -0.37615568 -0.0206098160
## [14,]  0.044605253 -0.37570521 -0.0144801414
## [15,]  0.002905193  0.02229189 -0.4056448021
## [16,]  0.003409730  0.02199104 -0.4067867230
## [17,]  0.003232816  0.02063848 -0.4105969740
## [18,]  0.002919683  0.02319284 -0.4055773542
## [19,]  0.002019241  0.02192068 -0.4087004821
## [20,]  0.001874190  0.02043083 -0.4077291009
```

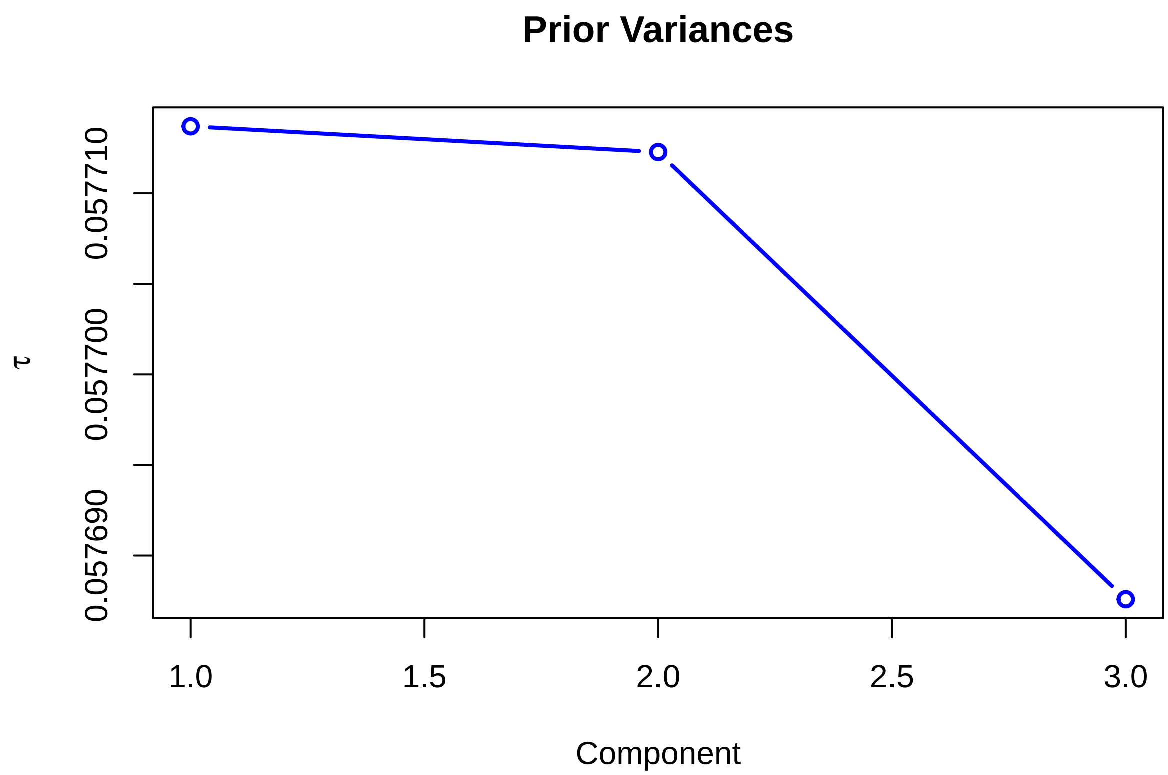

Figure 1: Prior variances for the first 3 components.

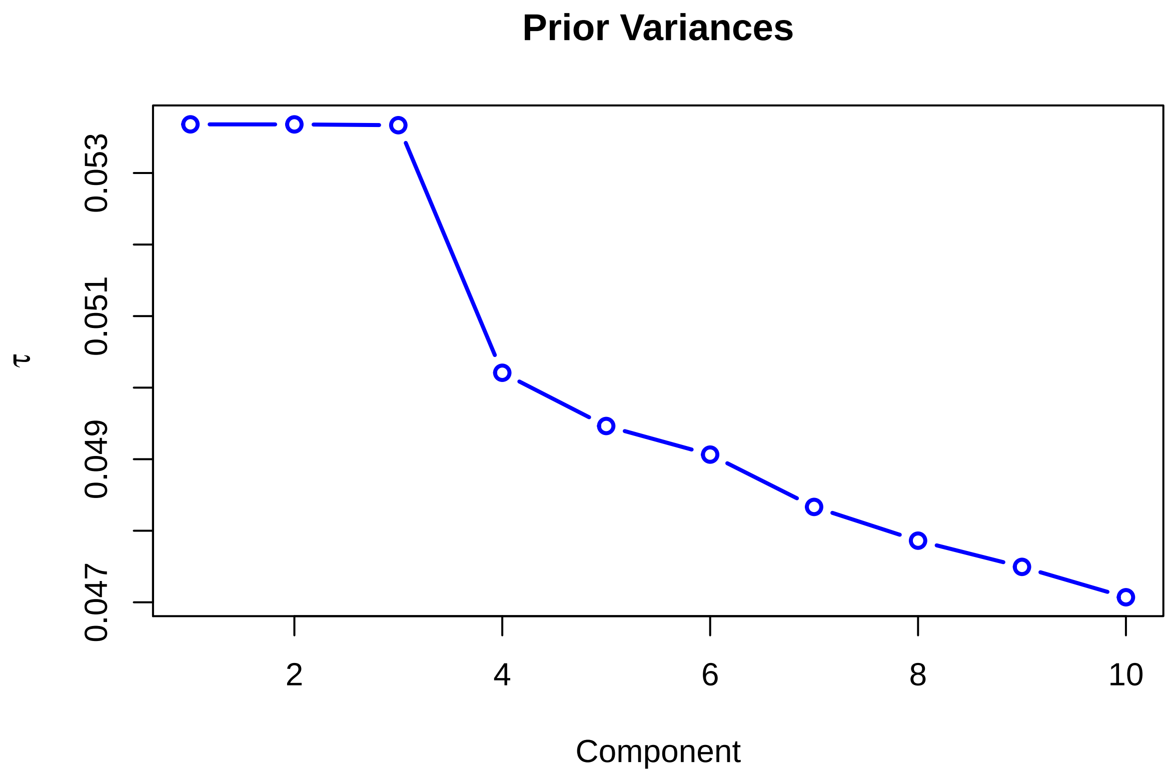

Figure 2: Prior variances for 10 components.

```
mod7$invTau
```

```
## [1] 17.32691 17.32734 17.33475
```

Notice the plot of the prior variances (inverse precisions) that appears in this case. This is useful when the number of components supported by the data is uncertain (elbow method):

```
mod8 <- vbpca(X, D = 10, maxIter = 1e+03, priorvar = 'fixed',
              updatetau = TRUE, global.var = TRUE,
              control = ctrl, verbose = FALSE )
```

Stochastic Search Variable Selection

By requiring `SVS = TRUE`, the model activates stochastic-search-variable-selection, a method described by George ad McCulloch (1993) for the Gibbs Sampler. The method has been adapted in *bayespca* for the Variational Bayes algorithm. The assumed 'spike-and-slab' prior for the (j, d)-th element of W becomes:

$$w_{j,d} \sim N(0, \pi\tau + (1-\pi)\tau v_0)$$

where v_0 is a scalar which rescales the spike variance to a value close to 0. For this reason, v_0 should be a number included in $(0, 1)$, as close as possible to 0. π represents the prior probability of inclusion of the j-th variable in the d-th component of the model. `vbpca` estimates the posterior probabilities of inclusion, conditional on X and the values in W.

While v_0 should be a small value close to 0, too small values of such parameter will shrink the variances τ too much, and no variable will eventually be included in the model. On the other hand, using a too large value for v_0 will not shrink the variances enough, and all posterior inclusion probabilities will be close to 1. v_0 should then be set with a grain of salt. Preliminary results from partial simulation studies have shown that values between 0.0001 and 0.005 lead to acceptable results, but adequate values of v_0 can be dataset-specific. Preliminary simulation studies have also shown that the method works better when Inverse Gamma priors are specified for τ, with `betatau` equal to 1 and larger values of `alphatau`. However, the technique has just been devised and further studies must be carried out to further test the functioning of Stochastic Variable Selection in this context.

In `vbpca`, the parameter v_0 is called `v0` in the control parameters of `vbpca_control`, while the prior inclusion probability is called `priorInclusion`. `priorInclusion` can be fixed, updated via Type-II maximum likelihood, or assigned to a Beta hyperprior:

- among the control parameters of `vbpca_control`, set `beta1pi` smaller than 0 for fixed π;
- set `beta1pi = 0` instead for Type-II ML updates;
- last, set `beta1pi` larger than 0 for Beta specifications.

When `beta1pi` is larger than 0, a Beta prior is assumed for π:

$$\pi \sim Beta(\beta_1, \beta_2).$$

In `vbpca`, $\beta 1$ can be controlled with the `beta1pi` argument and $\beta 2$ with the `beta2pi` argument in `vbpca_control`.

```
# SVS, fixed priorInclusion and InverseGamma(5, 1) for tau, v0 = .005
ctrl5 <- vbpca_control(center = FALSE, scalecorrection = -1,
                       plot.lowerbound = FALSE,
                       alphatau = 5, betatau = 1,
                       beta1pi = -1, v0 = 5e-03)

# Estimate the model with priorInclusion = 0.5
mod9 <- vbpca(X, D = 3, maxIter = 1e+03, priorvar = 'invgamma',
              SVS = TRUE, priorInclusion = 0.5, control = ctrl5,
              verbose = FALSE )

mod9$muW
```

```
##        Component 1 Component 2  Component 3
##   [1,] -0.376750692 -0.04448495 -0.004123683
##   [2,] -0.372459887 -0.04361185 -0.005752993
##   [3,] -0.375671975 -0.04356723 -0.004989270
##   [4,] -0.373836916 -0.04448097 -0.004703139
##   [5,] -0.377543656 -0.04319604 -0.005454948
##   [6,] -0.375846023 -0.04472475 -0.004528453
##   [7,] -0.375420482 -0.04399690 -0.004254785
##   [8,]  0.044408056 -0.37704714 -0.019871878
##   [9,]  0.043996739 -0.37297274 -0.020136604
##  [10,]  0.043266209 -0.37568704 -0.019144528
##  [11,]  0.043855992 -0.37127741 -0.019779142
##  [12,]  0.044744851 -0.37518584 -0.019753061
##  [13,]  0.045169754 -0.37455442 -0.019722877
##  [14,]  0.043743494 -0.37734332 -0.018263649
##  [15,]  0.002622365  0.02125415 -0.405655204
##  [16,]  0.002687542  0.02139614 -0.407985339
##  [17,]  0.002847711  0.02135195 -0.409082736
```

```
## [18,]  0.002536920  0.02132424 -0.406350545
## [19,]  0.002543555  0.02126640 -0.408688107
## [20,]  0.002678762  0.02114151 -0.408155324
```

```r
# SVS, priorInclusion update via Type-II ML and InverseGamma(5, 1) for tau, v0 = .005
ctrl6 <- vbpca_control(center = FALSE, scalecorrection = -1,
                       plot.lowerbound = FALSE,
                       alphatau = 5, betatau = 1,
                       beta1pi = 0, v0 = 5e-03)


# Estimate the model
mod10 <- vbpca(X, D = 3, maxIter = 1e+03, priorvar = 'invgamma',
               SVS = TRUE, control = ctrl6, verbose = FALSE )

mod10$muW
```

```
##        Component 1 Component 2  Component 3
##  [1,] -0.376822858 -0.04443304 -0.004110117
##  [2,] -0.372325895 -0.04358556 -0.005696672
##  [3,] -0.375719676 -0.04343733 -0.004947834
##  [4,] -0.373891994 -0.04417952 -0.004672995
##  [5,] -0.377499434 -0.04330105 -0.005404333
##  [6,] -0.375807989 -0.04447934 -0.004504553
##  [7,] -0.375545922 -0.04396995 -0.004237517
##  [8,]  0.044270327 -0.37708845 -0.019689948
##  [9,]  0.043950486 -0.37294924 -0.019941341
## [10,]  0.043388634 -0.37576275 -0.018993761
## [11,]  0.043790469 -0.37125593 -0.019593898
## [12,]  0.044483556 -0.37516369 -0.019574536
## [13,]  0.044624502 -0.37452245 -0.019541209
## [14,]  0.043971197 -0.37746934 -0.018159259
## [15,]  0.002607960  0.02107537 -0.405654575
## [16,]  0.002664143  0.02120185 -0.407997814
## [17,]  0.002842471  0.02116014 -0.409084282
## [18,]  0.002533579  0.02111624 -0.406361382
## [19,]  0.002532490  0.02108300 -0.408694793
## [20,]  0.002668127  0.02094020 -0.408183022
```

```r
# SVS, priorInclusion with Beta(1,1) priors and InverseGamma(5, 1) for tau, v0 = .005
ctrl7 <- vbpca_control(center = FALSE, scalecorrection = -1,
                       plot.lowerbound = FALSE, alphatau = 5,
                       betatau = 1, beta1pi = 1, beta2pi = 1,
                       v0 = 1e-03)


# Estimate the model
mod11 <- vbpca(X, D = 3, maxIter = 1e+03, priorvar = 'invgamma',
              SVS = TRUE, priorInclusion = 0.5, control = ctrl7,
              verbose = FALSE )


mod11$muW
```

```
##        Component 1  Component 2  Component 3
##  [1,] -0.377585969 -0.047939333 -0.002001648
```

```
##  [2,] -0.373162862 -0.042487944 -0.002374892
##  [3,] -0.374424310 -0.040392824 -0.002183428
##  [4,] -0.375075559 -0.046537326 -0.002133489
##  [5,] -0.376001850 -0.039418960 -0.002301476
##  [6,] -0.375864889 -0.047130795 -0.002098348
##  [7,] -0.375409214 -0.045695502 -0.002031570
##  [8,]  0.043401229 -0.378488759 -0.006154046
##  [9,]  0.044375834 -0.373875543 -0.006198091
## [10,]  0.042239525 -0.377115242 -0.005992542
## [11,]  0.044761079 -0.368634823 -0.006106240
## [12,]  0.044958241 -0.376871710 -0.006118384
## [13,]  0.045641060 -0.375075551 -0.006110655
## [14,]  0.044348933 -0.377244104 -0.005812578
## [15,]  0.001435196  0.006309373 -0.405990765
## [16,]  0.001444623  0.006308304 -0.408536058
## [17,]  0.001463239  0.006364100 -0.409391471
## [18,]  0.001438362  0.006326944 -0.406792705
## [19,]  0.001422580  0.006352733 -0.409223547
## [20,]  0.001425179  0.006319707 -0.409095996
```

The estimated posterior inclusion probabilities for the three models:

```
mod9$inclusionProbabilities
```

```
##             [,1]      [,2]       [,3]
##  [1,] 1.00000000 0.2120296 0.09726907
##  [2,] 1.00000000 0.2068326 0.09851144
##  [3,] 1.00000000 0.2084065 0.09897047
##  [4,] 1.00000000 0.2139049 0.09843787
##  [5,] 1.00000000 0.2044355 0.09823061
##  [6,] 1.00000000 0.2146309 0.09848977
##  [7,] 1.00000000 0.2091004 0.09793821
##  [8,] 0.21428441 1.0000000 0.11723947
##  [9,] 0.21111001 1.0000000 0.11752795
## [10,] 0.20612101 1.0000000 0.11542999
## [11,] 0.21070545 1.0000000 0.11707616
## [12,] 0.21594418 1.0000000 0.11702863
## [13,] 0.21982827 1.0000000 0.11758358
## [14,] 0.20661409 1.0000000 0.11227938
## [15,] 0.09660274 0.1179197 1.00000000
## [16,] 0.09702372 0.1193704 1.00000000
## [17,] 0.09668529 0.1185038 1.00000000
## [18,] 0.09705420 0.1192275 1.00000000
## [19,] 0.09635180 0.1178253 1.00000000
## [20,] 0.09741905 0.1191173 1.00000000
```

```
mod10$inclusionProbabilities
```

```
##             [,1]       [,2]       [,3]
##  [1,] 1.00000000 0.14604123 0.06595144
##  [2,] 1.00000000 0.14241876 0.06677927
##  [3,] 1.00000000 0.14288168 0.06707145
##  [4,] 1.00000000 0.14595656 0.06672092
##  [5,] 1.00000000 0.14130427 0.06659022
##  [6,] 1.00000000 0.14681895 0.06675491
##  [7,] 1.00000000 0.14405489 0.06639296
```

```
##  [8,] 0.14694535 1.00000000 0.07949594
##  [9,] 0.14519685 1.00000000 0.07968820
## [10,] 0.14248443 1.00000000 0.07830934
## [11,] 0.14478837 1.00000000 0.07938071
## [12,] 0.14753285 1.00000000 0.07935458
## [13,] 0.14873386 1.00000000 0.07970621
## [14,] 0.14356166 1.00000000 0.07625060
## [15,] 0.06548873 0.08003940 1.00000000
## [16,] 0.06576005 0.08097049 1.00000000
## [17,] 0.06553993 0.08040118 1.00000000
## [18,] 0.06578191 0.08085128 1.00000000
## [19,] 0.06532184 0.07996842 1.00000000
## [20,] 0.06601984 0.08076926 1.00000000
```

```
mod11$inclusionProbabilities
```

```
##             [,1]       [,2]       [,3]
##  [1,] 1.00000000 1.00000000 0.06864236
##  [2,] 1.00000000 1.00000000 0.06905001
##  [3,] 1.00000000 1.00000000 0.06903261
##  [4,] 1.00000000 1.00000000 0.06892463
##  [5,] 1.00000000 1.00000000 0.06891797
##  [6,] 1.00000000 1.00000000 0.06892288
##  [7,] 1.00000000 1.00000000 0.06880774
##  [8,] 1.00000000 1.00000000 0.07471343
##  [9,] 1.00000000 1.00000000 0.07480727
## [10,] 1.00000000 1.00000000 0.07439398
## [11,] 1.00000000 1.00000000 0.07467890
## [12,] 1.00000000 1.00000000 0.07465084
## [13,] 1.00000000 1.00000000 0.07471088
## [14,] 1.00000000 1.00000000 0.07374679
## [15,] 0.06814353 0.07456737 1.00000000
## [16,] 0.06827512 0.07470610 1.00000000
## [17,] 0.06806801 0.07459243 1.00000000
## [18,] 0.06826865 0.07474289 1.00000000
## [19,] 0.06804514 0.07456130 1.00000000
## [20,] 0.06823635 0.07470003 1.00000000
```

It is also possible to compare the (known) variable inclusion matrix vs. the estimated ones graphically. Let's plot a heatmap of such probabilities for model `mod9`:

```
trueInclusions <- matrix(0, J, 3)
trueInclusions[1:7, 1] <- 1
trueInclusions[8:14, 2] <- 1
trueInclusions[15:20, 3] <- 1

par(mfrow=c(1,2))
image(1:ncol(trueInclusions), 1:nrow(trueInclusions),
      t(trueInclusions[J:1, ]), ylab = "", axes = FALSE,
      main = "True Inclusions", xlab = "",
      col =  RColorBrewer::brewer.pal(9, "Blues"))
axis(side = 1, at = 1:3, labels = paste("Component ", 1:3 ))
axis(side = 2, at = 1:20, labels = paste("Var ", J:1 ))

fields::image.plot(1:ncol(trueInclusions), 1:nrow(trueInclusions),
      t(mod9$inclusionProbabilities[J:1, ]), ylab = "", axes = FALSE,
```

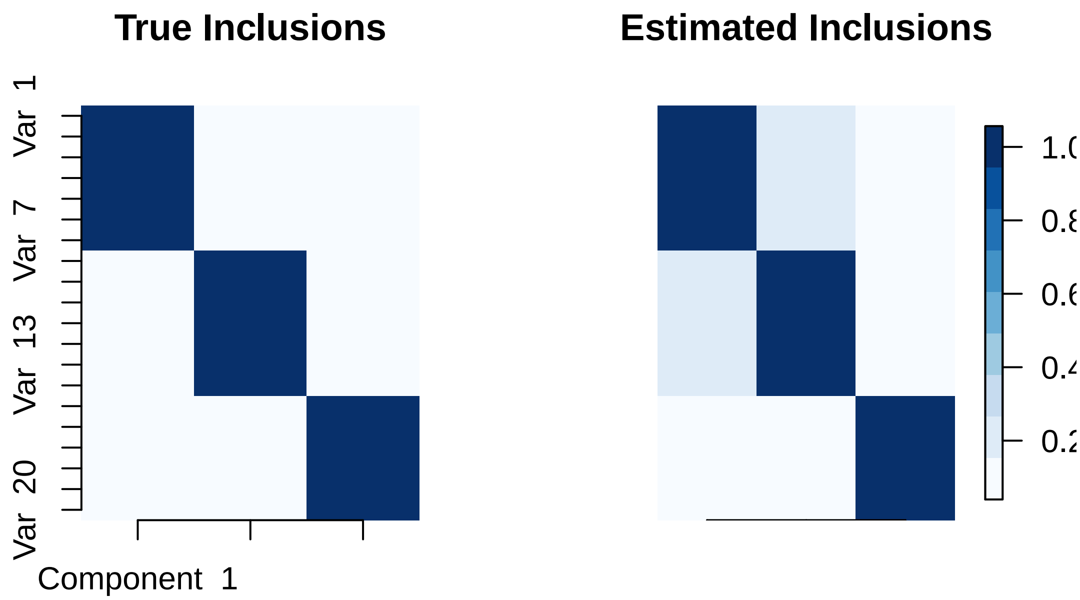

Figure 3: True and Estimated inclusion probabilities.

```
      main = "Estimated Inclusions", xlab = "",
      col = RColorBrewer::brewer.pal(9, "Blues"))
axis(side = 1, at = 1:3, labels = paste("Component ", 1:3 ))
```

For `mod10` and `mod11` we can observe the estimated prior inclusion probabilities:

```
mod10$priorInclusion
```

```
##           [,1]
## [1,] 0.3987346
## [2,] 0.3987346
## [3,] 0.3987346
```

```
mod11$priorInclusion
```

```
##           [,1]
## [1,] 0.5945308
## [2,] 0.5945308
## [3,] 0.5945308
```

Similar to the hyperparameters of the Inverse Gamma priors on τ, `priorInclusion`, `beta1pi` and `beta2pi`can also be specified as D-dimensional arrays. This will allow estimating the inclusion probabilities with different degrees of 'sparsity' for each component. For Type-II maximum likelihood updates of `priorInclusion`, all elements of `beta1pi` must be equal to 0. For Beta priors, all elements of `beta1pi` must be larger than 0. Let us look at two examples:

```
# Type-II maximum likelihood (component-specific) updates
ctrl8 <- vbpca_control(center = FALSE, scalecorrection = -1,
```

```
                        plot.lowerbound = FALSE,
                        alphatau = 5, betatau = 1,
                        beta1pi = rep(0, 3), v0 = 5e-03)

# Estimate the model
mod12 <- vbpca(X, D = 3, maxIter = 1e+03, priorvar = 'invgamma',
                SVS = TRUE, priorInclusion = rep(0.5, 3),
                control = ctrl8, verbose = FALSE )


mod12$muW
```

```
##        Component 1 Component 2  Component 3
##  [1,] -0.376817775 -0.04443026 -0.004110336
##  [2,] -0.372360508 -0.04355404 -0.005678225
##  [3,] -0.375711830 -0.04340495 -0.004935813
##  [4,] -0.373900042 -0.04418527 -0.004666015
##  [5,] -0.377497008 -0.04324630 -0.005388196
##  [6,] -0.375809312 -0.04449066 -0.004500207
##  [7,] -0.375532827 -0.04394650 -0.004236029
##  [8,]  0.044261824 -0.37708496 -0.019640556
##  [9,]  0.043929313 -0.37297274 -0.019886136
## [10,]  0.043342443 -0.37573263 -0.018956990
## [11,]  0.043766769 -0.37127627 -0.019542974
## [12,]  0.044481053 -0.37518476 -0.019526532
## [13,]  0.044636397 -0.37455303 -0.019492259
## [14,]  0.043931728 -0.37743803 -0.018141983
## [15,]  0.002607056  0.02102946 -0.405651922
## [16,]  0.002663824  0.02115916 -0.408001417
## [17,]  0.002840070  0.02111391 -0.409084633
## [18,]  0.002532502  0.02107713 -0.406361999
## [19,]  0.002530700  0.02103708 -0.408695945
## [20,]  0.002665014  0.02089807 -0.408194788
```

```
mod12$priorInclusion
```

```
##           [,1]
## [1,] 0.4294635
## [2,] 0.4359775
## [3,] 0.3399416
```

```
mod12$inclusionProbabilities
```

```
##             [,1]       [,2]       [,3]
##  [1,] 1.00000000 0.16810342 0.05171407
##  [2,] 1.00000000 0.16382116 0.05235869
##  [3,] 1.00000000 0.16439404 0.05258104
##  [4,] 1.00000000 0.16809442 0.05230977
##  [5,] 1.00000000 0.16244473 0.05221079
##  [6,] 1.00000000 0.16908934 0.05233593
##  [7,] 1.00000000 0.16573007 0.05205611
##  [8,] 0.16509191 1.00000000 0.06237005
##  [9,] 0.16306252 1.00000000 0.06251995
## [10,] 0.15990420 1.00000000 0.06145088
## [11,] 0.16260305 1.00000000 0.06227781
```

```
## [12,] 0.16576567 1.00000000 0.06225943
## [13,] 0.16719978 1.00000000 0.06252745
## [14,] 0.16109063 1.00000000 0.05986055
## [15,] 0.07390531 0.09243547 1.00000000
## [16,] 0.07421527 0.09352661 1.00000000
## [17,] 0.07396456 0.09285742 1.00000000
## [18,] 0.07424059 0.09339617 1.00000000
## [19,] 0.07371601 0.09235335 1.00000000
## [20,] 0.07451101 0.09330382 1.00000000
```

```r
# Beta priors with different degrees of sparsity for each component
ctrl9 <- vbpca_control(center = FALSE, scalecorrection = -1,
                       plot.lowerbound = FALSE,
                       alphatau = 5, betatau = 1,
                       beta1pi = c(0.01, 1, 10), beta2pi = 1,
                       v0 = 5e-03)



# Estimate the model
mod13 <- vbpca(X, D = 3, maxIter = 1e+03, priorvar = 'invgamma', SVS = TRUE,
              priorInclusion = rep(0.5, 3), control = ctrl9, verbose = FALSE )

mod13$muW
```

```
##        Component 1 Component 2  Component 3
##  [1,] -0.376823553 -0.04445292 -0.004069605
##  [2,] -0.372321780 -0.04356864 -0.005750480
##  [3,] -0.375720588 -0.04342350 -0.004970829
##  [4,] -0.373891258 -0.04421414 -0.004669279
##  [5,] -0.377499571 -0.04325604 -0.005447522
##  [6,] -0.375807748 -0.04451895 -0.004487080
##  [7,] -0.375547731 -0.04396570 -0.004205898
##  [8,]  0.044271252 -0.37708007 -0.019742308
##  [9,]  0.043952908 -0.37296691 -0.020024420
## [10,]  0.043394060 -0.37573006 -0.018982397
## [11,]  0.043792974 -0.37126432 -0.019656898
## [12,]  0.044483312 -0.37518104 -0.019620517
## [13,]  0.044621267 -0.37454675 -0.019593263
## [14,]  0.043976056 -0.37743206 -0.018045949
## [15,]  0.002608530  0.02110045 -0.405681172
## [16,]  0.002664620  0.02123110 -0.407991414
## [17,]  0.002843244  0.02118633 -0.409091995
## [18,]  0.002534198  0.02114931 -0.406372349
## [19,]  0.002533170  0.02110874 -0.408700786
## [20,]  0.002668932  0.02097038 -0.408132057
```

```r
mod13$priorInclusion
```

```
##           [,1]
## [1,] 0.3999559
## [2,] 0.4432304
## [3,] 0.5825461
```

```r
mod13$inclusionProbabilities
```

```
##              [,1]       [,2]      [,3]
```

```
##  [1,] 1.00000000 0.17112197 0.1337330
##  [2,] 1.00000000 0.16672704 0.1354631
##  [3,] 1.00000000 0.16733858 0.1361515
##  [4,] 1.00000000 0.17115717 0.1353939
##  [5,] 1.00000000 0.16530310 0.1350775
##  [6,] 1.00000000 0.17216159 0.1354701
##  [7,] 1.00000000 0.16869081 0.1346833
##  [8,] 0.14487471 1.00000000 0.1602905
##  [9,] 0.14315770 1.00000000 0.1606833
## [10,] 0.14049548 1.00000000 0.1577618
## [11,] 0.14275404 1.00000000 0.1600890
## [12,] 0.14545003 1.00000000 0.1600062
## [13,] 0.14661853 1.00000000 0.1608249
## [14,] 0.14156091 1.00000000 0.1533005
## [15,] 0.06453304 0.09420471 1.0000000
## [16,] 0.06480003 0.09532060 1.0000000
## [17,] 0.06458335 0.09463845 1.0000000
## [18,] 0.06482151 0.09518785 1.0000000
## [19,] 0.06436869 0.09412224 1.0000000
## [20,] 0.06505576 0.09509410 1.0000000
```

High posterior density intervals

It is also possible to require the computation of high probability density intervals for the elements of W, which can then be plotted with the `plothpdi` function, which internally calls `ggplot2` functionalities.

```
# Set hyperparameter values and require 50% probability density intervals
ctrl10 <- vbpca_control(center = FALSE, scalecorrection = -1,
                        plot.lowerbound = FALSE,
                        alphatau = 2, betatau = .5,
                        hpdi = TRUE, probHPDI = 0.5)

# Estimate the model
mod14 <- vbpca(X, D = 3, maxIter = 1e+03, priorvar = 'invgamma',
               control = ctrl10, verbose = FALSE )

# Plot HPD intervals for variables 1:10 component 1
plothpdi(mod14, d = 1, X, vars = 1:10)
```

Retrieve Principal Components

To obtain the component scores, simpy call:

```
# Example with model mod1
PCs <- X %*% mod1$muW
head(PCs, 15)
```

```
##       Component 1 Component 2 Component 3
##  [1,]   -59.19132  -78.592706  31.3401006
##  [2,]    28.97173 -118.789001 -29.0200757
##  [3,]   -11.00518   14.227039  -4.8429359
##  [4,]    92.16140  -33.606390 -28.1184464
##  [5,]   -41.61482 -212.440556  13.4800625
```

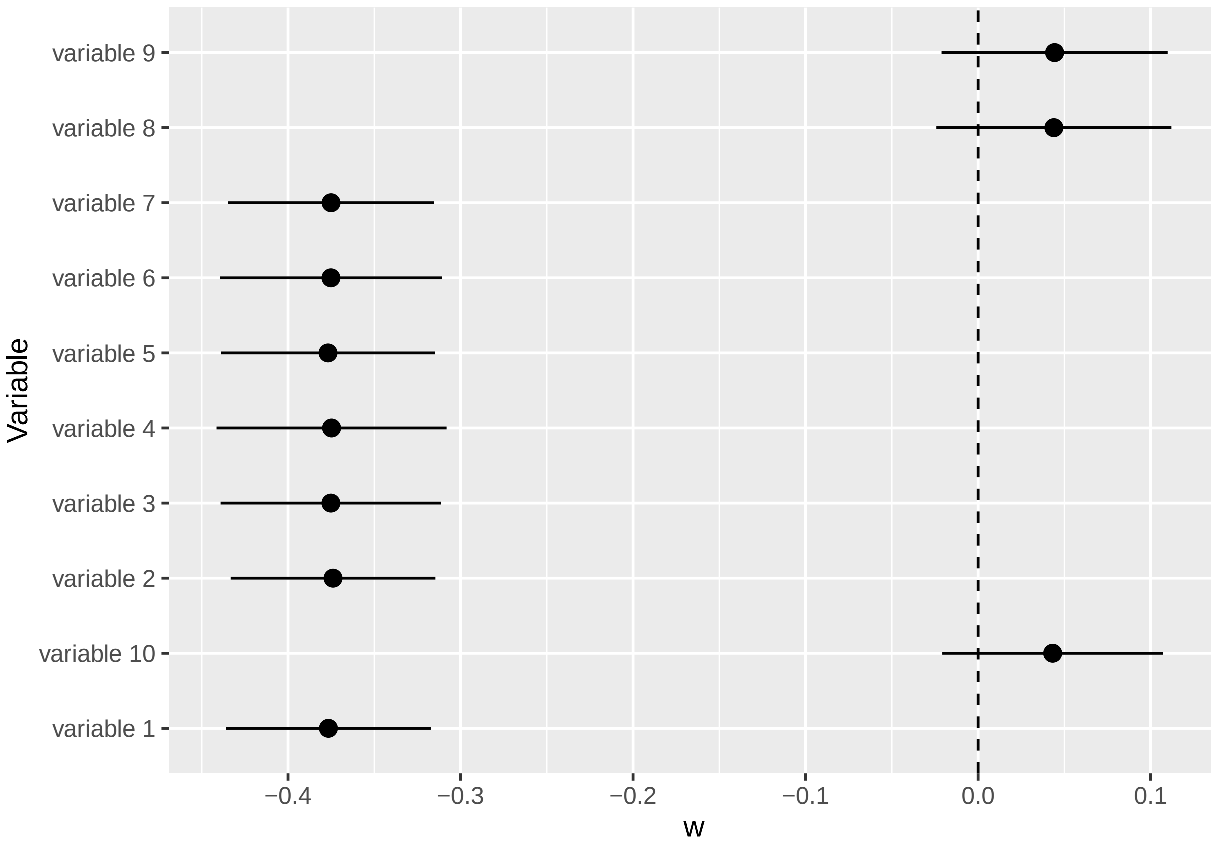

Figure 4: High posterior density intervals.

```
##  [6,]   113.51610  -20.107248   5.6778539
##  [7,]    98.45308  -73.892682  17.2711799
##  [8,]    42.05467 -142.922656 -68.0937444
##  [9,]   -57.38540  -66.586046  17.5396890
## [10,]    42.94090   51.286634  -0.2553017
## [11,]    36.39523  -11.871548  13.9383073
## [12,]   109.60474   -6.656482  25.3900540
## [13,]  -196.01791  110.020823  -9.5996904
## [14,]  -267.42318   71.336728  14.1676674
## [15,]    38.49334   22.034659 -32.6994037
```

References

1. C. M. Bishop. 'Variational PCA'. In Proc. Ninth Int. Conf. on Artificial Neural Networks. ICANN, 1999.
2. E. I. George, R. E. McCulloch (1993). 'Variable Selection via Gibbs Sampling'. Journal of the American Statistical Association (88), 881-889.