bayespca Package

Davide Vidotto d.vidotto@uvt.nl

2020-09-07

Contents

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}.

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 precisions; 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 )
```

```
## Warning: unscaled data - ELBO values might be positive.
```

```
# 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
## variable 1  -0.376589700 -0.04416511  0.0003399127
## variable 2  -0.373939778 -0.04582346 -0.0111489595
## variable 3  -0.375148658 -0.04305857 -0.0078831845
## variable 4  -0.374770078 -0.04473100 -0.0031124941
## variable 5  -0.376808027 -0.04285792 -0.0100250681
## variable 6  -0.375114066 -0.04446329 -0.0015012124
## variable 7  -0.375069347 -0.04364081 -0.0007181138
## variable 8   0.043916074 -0.37610685 -0.0194987844
## variable 9   0.044338996 -0.37382690 -0.0224165536
## variable 10  0.043216238 -0.37319456 -0.0161965576
## variable 11  0.043432789 -0.37311089 -0.0246530518
## variable 12  0.045420158 -0.37574267 -0.0200072059
## variable 13  0.045158091 -0.37616395 -0.0206149568
## variable 14  0.044605651 -0.37571347 -0.0144837533
## variable 15  0.002905219  0.02229238 -0.4057459837
## variable 16  0.003409761  0.02199152 -0.4068881894
## variable 17  0.003232845  0.02063894 -0.4106993908
## variable 18  0.002919709  0.02319335 -0.4056785190
## variable 19  0.002019259  0.02192116 -0.4088024260
## variable 20  0.001874207  0.02043128 -0.4078308025
```

and the P matrix:

```r
mod1$P
```

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

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

```
mod1$elbo
```

```
## [1] -2834.277
```

```
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):

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

```
## Warning: unscaled data - ELBO values might be positive.
```

```
mod2$muW
```

```
##                 Component 1  Component 2  Component 3
## variable 1  -3.774720e-01 -0.051470102 -0.001822156
## variable 2  -3.744848e-01 -0.025307896 -0.002337156
## variable 3  -3.747070e-01 -0.039179989 -0.002345241
## variable 4  -3.697882e-01 -0.062258504 -0.002242303
## variable 5  -3.794353e-01 -0.031311390 -0.002225193
## variable 6  -3.811333e-01 -0.058594825 -0.001901847
## variable 7  -3.709497e-01 -0.037084346 -0.001895318
## variable 8   4.572526e-02 -0.388171353 -0.019796029
## variable 9   4.119102e-02 -0.376636645 -0.023267758
## variable 10  2.518537e-02 -0.376940208 -0.006967316
## variable 11  5.297341e-02 -0.374414696 -0.022267903
## variable 12  4.534880e-02 -0.374381287 -0.013986121
## variable 13  6.385742e-02 -0.370055099 -0.031673499
## variable 14  3.111306e-02 -0.364729611 -0.002535536
## variable 15  1.288107e-05  0.006270320 -0.406011655
## variable 16  1.305259e-05  0.018689756 -0.407077598
## variable 17  1.211560e-05  0.009078483 -0.411159927
## variable 18  1.039548e-05  0.036022360 -0.398788351
## variable 19  1.331297e-05  0.005889322 -0.410538009
## variable 20  1.326763e-05  0.035028974 -0.412758321
```

The matrix of the inverse prior variances can be called with

```
mod2$Tau
```

```
##               Component 1 Component 2  Component 3
## variable 1  6.711653e+00  185.292815 30792.083009
## variable 2  6.811715e+00  435.456531 24282.641765
## variable 3  6.733579e+00  246.605678 24174.268760
## variable 4  6.906649e+00  137.269136 25240.818694
## variable 5  6.630610e+00  334.561356 25606.215195
## variable 6  6.535962e+00  151.540810 29303.191065
## variable 7  6.916642e+00  278.901798 29505.931801
## variable 8  2.000395e+02    6.283433   670.346568
## variable 9  2.297542e+02    6.662857   568.834746
## variable 10 4.206301e+02    6.647409  1984.202810
## variable 11 1.678142e+02    6.740676   596.809053
```

```
## variable 12 2.033657e+02    6.710938   964.165004
## variable 13 1.294742e+02    6.828244   390.109550
## variable 14 3.541861e+02    7.207997  5635.426286
## variable 15 3.077602e+05 2373.138268     5.837005
## variable 16 3.081676e+05  748.154670     5.771306
## variable 17 3.180735e+05 1556.626020     5.676298
## variable 18 3.115912e+05  349.915331     5.997557
## variable 19 3.116412e+05 2503.322335     5.710167
## variable 20 3.100489e+05  362.448139     5.591906
```

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`:

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

```
## Warning: unscaled data - ELBO values might be positive.
```

```
## Warning: vbpca has not converged. Please re-run by increasing <maxIter> or
## the convergence criterion <tolerance>.
```

```
mod3$muW
```

```
##              Component 1   Component 2   Component 3
## variable 1  -3.747045e-01 -3.323508e-02 -3.469376e-09
## variable 2  -3.768041e-01 -3.546025e-02 -1.175874e-08
## variable 3  -3.767929e-01 -4.253830e-02 -8.255266e-09
## variable 4  -3.696327e-01 -4.764153e-02 -6.556791e-09
## variable 5  -3.830454e-01 -2.826201e-02 -1.077942e-08
## variable 6  -3.804134e-01 -3.528604e-02 -5.789307e-09
## variable 7  -3.716495e-01 -3.597893e-02 -4.568835e-09
## variable 8   4.222188e-02 -3.791051e-01 -6.419779e-09
## variable 9   3.590253e-02 -3.754983e-01 -1.234420e-08
## variable 10  2.604233e-02 -3.751967e-01  1.982054e-09
## variable 11  3.791750e-02 -3.760256e-01 -6.043133e-09
## variable 12  3.932376e-02 -3.760470e-01 -4.831003e-09
## variable 13  5.335474e-02 -3.776686e-01 -2.909598e-09
## variable 14  2.342717e-02 -3.735192e-01  1.798051e-08
## variable 15 -2.141733e-07 -3.653921e-08 -4.063669e-01
## variable 16 -1.912349e-07 -3.917626e-08 -4.088009e-01
## variable 17 -2.196324e-07 -4.735018e-09 -4.085224e-01
## variable 18 -2.948107e-07  2.194439e-08 -4.083168e-01
## variable 19 -2.101471e-07 -5.903770e-08 -4.093516e-01
## variable 20 -2.057003e-07 -4.254817e-08 -4.081807e-01
```

```
mod3$Tau
```

```
##             Component 1  Component 2  Component 3
## variable 1  6.811129e+00 3.275521e+02 2.096339e+08
## variable 2  6.734307e+00 3.035430e+02 2.068832e+08
## variable 3  6.664783e+00 2.307734e+02 2.079720e+08
```

```
## variable 4  6.916776e+00 2.006493e+02 2.076421e+08
## variable 5  6.515547e+00 3.916175e+02 2.098365e+08
## variable 6  6.561549e+00 2.948426e+02 2.079227e+08
## variable 7  6.893544e+00 2.984187e+02 2.080337e+08
## variable 8  2.261027e+02 6.577232e+00 5.967677e+07
## variable 9  2.791784e+02 6.706563e+00 5.919419e+07
## variable 10 4.116337e+02 6.712694e+00 5.861855e+07
## variable 11 2.627713e+02 6.701316e+00 5.873562e+07
## variable 12 2.492845e+02 6.660847e+00 5.967282e+07
## variable 13 1.680250e+02 6.580189e+00 5.982249e+07
## variable 14 5.034247e+02 6.894501e+00 5.933761e+07
## variable 15 8.531621e+06 6.326452e+06 5.829825e+00
## variable 16 8.589341e+06 6.295699e+06 5.731758e+00
## variable 17 8.817647e+06 6.474544e+06 5.755315e+00
## variable 18 8.561941e+06 6.243056e+06 5.741057e+00
## variable 19 8.642399e+06 6.433209e+06 5.745663e+00
## variable 20 8.643250e+06 6.361800e+06 5.723730e+00
```

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 )
```

```
## Warning: unscaled data - ELBO values might be positive.
```

```r
mod4$muW
```

```
##              Component 1 Component 2   Component 3
## variable 1  -0.376590725 -0.04416826  0.0002947089
## variable 2  -0.373916478 -0.04580818 -0.0111139002
## variable 3  -0.375152260 -0.04306330 -0.0078485203
## variable 4  -0.374771451 -0.04473662 -0.0031287802
## variable 5  -0.376819149 -0.04285298 -0.0099904658
## variable 6  -0.375128188 -0.04445803 -0.0015201459
## variable 7  -0.375056611 -0.04365135 -0.0007401483
## variable 8   0.043918292 -0.37612390 -0.0195191684
## variable 9   0.044336996 -0.37380699 -0.0224249101
## variable 10  0.043215418 -0.37316141 -0.0162245050
## variable 11  0.043435548 -0.37309952 -0.0245960805
## variable 12  0.045416539 -0.37575273 -0.0200103648
## variable 13  0.045161055 -0.37621087 -0.0206044028
```

```
## variable 14  0.044603008 -0.37569186 -0.0144838955
## variable 15  0.002901159  0.02228837 -0.4056891063
## variable 16  0.003405260  0.02198287 -0.4068836504
## variable 17  0.003228832  0.02064976 -0.4107113982
## variable 18  0.002920806  0.02319222 -0.4056322666
## variable 19  0.002018556  0.02191743 -0.4087810436
## variable 20  0.001885825  0.02043591 -0.4078295957
```

```
mod4$Tau
```

```
##             Component 1 Component 2 Component 3
## variable 1     4.349798    4.952591    4.962213
## variable 2     4.357381    4.951847    4.961554
## variable 3     4.349198    4.947077    4.955868
## variable 4     4.347326    4.942587    4.952373
## variable 5     4.346511    4.949795    4.958338
## variable 6     4.348754    4.945842    4.955514
## variable 7     4.353855    4.952447    4.961810
## variable 8     4.940550    4.341632    4.948114
## variable 9     4.944064    4.351079    4.951230
## variable 10    4.946698    4.354576    4.954566
## variable 11    4.941841    4.351048    4.948103
## variable 12    4.940429    4.343068    4.948542
## variable 13    4.934790    4.337277    4.942629
## variable 14    4.954153    4.353743    4.962926
## variable 15    4.964065    4.961664    4.266733
## variable 16    4.955381    4.953028    4.256713
## variable 17    4.956688    4.954630    4.246359
## variable 18    4.954427    4.951810    4.259675
## variable 19    4.962441    4.960082    4.256343
## variable 20    4.950618    4.948561    4.250325
```

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

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

.

```
# 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 )
```

```
## Warning: unscaled data - ELBO values might be positive.
```

```
mod5$muW
```

```
##              Component 1  Component 2   Component 3
## variable 1  -0.378535069 -0.022550177  0.0025320655
## variable 2  -0.376010407 -0.021331002 -0.0088680201
## variable 3  -0.377058121 -0.022705524 -0.0057334841
```

```
## variable 4  -0.376765736 -0.022799099 -0.0008880582
## variable 5  -0.378711737 -0.022502952 -0.0078832752
## variable 6  -0.377093814 -0.019646614  0.0006966940
## variable 7  -0.376981673 -0.026435232  0.0014645273
## variable 8   0.021618521 -0.045787586 -0.0012802351
## variable 9   0.022165897 -0.050285073 -0.0042910494
## variable 10  0.021098823 -0.037934476  0.0018542188
## variable 11  0.021297952 -0.049321093 -0.0065615922
## variable 12  0.023138118 -0.049443570 -0.0017949202
## variable 13  0.022842413 -2.348222240 -0.0025229191
## variable 14  0.022345189 -0.049142867  0.0037198670
## variable 15  0.002992116  0.002091087 -0.4063507109
## variable 16  0.003475947 -0.001948958 -0.4074613372
## variable 17  0.003205251  0.009070114 -0.4112473691
## variable 18  0.003062334  0.006593144 -0.4063446402
## variable 19  0.002077795  0.003818705 -0.4093929706
## variable 20  0.001857368 -0.003947850 -0.4083201743
```

```r
mod5$Tau
```

```
##             Component 1 Component 2 Component 3
## variable 1     1.735068  4883.95439   0.3498320
## variable 2     1.737956  4896.56336   0.3498312
## variable 3     1.734465  4881.98803   0.3498046
## variable 4     1.733223  4881.05886   0.3497852
## variable 5     1.733528  4884.42780   0.3498147
## variable 6     1.734107  4913.05908   0.3498008
## variable 7     1.736710  4838.95652   0.3498308
## variable 8     1.974312  4527.17251   0.3497740
## variable 9     1.976205  4440.53154   0.3497924
## variable 10    1.977395  4665.76589   0.3498030
## variable 11    1.974916  4459.48648   0.3497789
## variable 12    1.974487  4457.06148   0.3497771
## variable 13    1.971541    18.24647   0.3497487
## variable 14    1.981271  4463.32575   0.3498394
## variable 15    1.982402  5007.31481   0.3469784
## variable 16    1.977961  5007.38609   0.3469217
## variable 17    1.978586  4988.06776   0.3468739
## variable 18    1.977499  4997.44534   0.3469332
## variable 19    1.981517  5004.88288   0.3469270
## variable 20    1.975535  5004.38616   0.3468870
```

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:

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

```r
                        gammatau = .01, deltatau = 10,
                        hypertype = 'component')


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

```
## Warning: unscaled data - ELBO values might be positive.
```

```r
mod6$muW
```

```
##              Component 1 Component 2  Component 3
## variable 1  -0.376612836 -0.04414791 -0.001248665
## variable 2  -0.373527098 -0.04527330 -0.009492228
## variable 3  -0.375300115 -0.04330108 -0.006689935
## variable 4  -0.374545218 -0.04475906 -0.003778637
## variable 5  -0.377133411 -0.04290093 -0.008534217
## variable 6  -0.375467993 -0.04449136 -0.002517447
## variable 7  -0.374843813 -0.04378361 -0.001752866
## variable 8   0.044090921 -0.37645722 -0.020075378
## variable 9   0.044279637 -0.37340637 -0.022183245
## variable 10  0.043230956 -0.37276205 -0.017350634
## variable 11  0.043633248 -0.37285565 -0.022662370
## variable 12  0.045225801 -0.37597165 -0.020153747
## variable 13  0.045216689 -0.37689952 -0.020320873
## variable 14  0.044313381 -0.37547626 -0.014958361
## variable 15  0.002830475  0.02204270 -0.405124312
## variable 16  0.003209544  0.02187185 -0.407342357
## variable 17  0.003155435  0.02095304 -0.410906971
## variable 18  0.002824749  0.02288630 -0.405233175
## variable 19  0.002136077  0.02174539 -0.408749536
## variable 20  0.002176061  0.02077001 -0.407947441
```

```r
mod6$Tau
```

```
##             Component 1 Component 2 Component 3
## variable 1     14.91448    50.12038    65.06736
## variable 2     15.08009    50.01542    64.75625
## variable 3     14.89640    49.61854    64.01402
## variable 4     14.91646    49.35990    64.02027
## variable 5     14.84777    50.00467    64.60364
## variable 6     14.89316    49.61383    64.18919
## variable 7     14.99621    50.11334    64.83842
## variable 8     49.11895    14.79168    63.03510
## variable 9     49.33578    14.99044    63.23453
## variable 10    49.58424    15.04221    63.72938
## variable 11    49.21586    15.00398    62.95653
## variable 12    49.08557    14.80959    63.03858
## variable 13    48.70446    14.70323    62.43163
## variable 14    50.23753    15.02297    64.98818
## variable 15    51.91718    51.58330    14.15424
## variable 16    51.29932    50.86616    13.95682
## variable 17    51.45181    51.11369    13.78557
## variable 18    51.19581    50.75885    14.05424
```

```
## variable 19    51.89061    51.52052    13.95095
## variable 20    50.91504    50.55465    13.87395
```

The posterior means of β can be accessed via

```r
mod6$priorBeta
```

```
##             [,1]       [,2]       [,3]
## [1,] 0.02635365 0.02630866 0.02068517
## 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)
```

```
## Warning: unscaled data - ELBO values might be positive.
```

```r
mod7$muW
```

```
##               Component 1 Component 2   Component 3
## variable 1  -0.376586376 -0.04416415  0.0003398288
## variable 2  -0.373936478 -0.04582247 -0.0111462062
## variable 3  -0.375145347 -0.04305764 -0.0078812377
## variable 4  -0.374766771 -0.04473003 -0.0031117254
## variable 5  -0.376804702 -0.04285698 -0.0100225924
## variable 6  -0.375110756 -0.04446232 -0.0015008417
## variable 7  -0.375066036 -0.04363986 -0.0007179364
## variable 8   0.043915686 -0.37609866 -0.0194939691
## variable 9   0.044338605 -0.37381876 -0.0224110177
## variable 10  0.043215857 -0.37318643 -0.0161925578
## variable 11  0.043432406 -0.37310277 -0.0246469635
## variable 12  0.045419757 -0.37573449 -0.0200022650
## variable 13  0.045157692 -0.37615576 -0.0206098658
## variable 14  0.044605257 -0.37570529 -0.0144801764
## variable 15  0.002905193  0.02229190 -0.4056457820
## variable 16  0.003409731  0.02199105 -0.4067877056
## variable 17  0.003232816  0.02063849 -0.4105979658
```

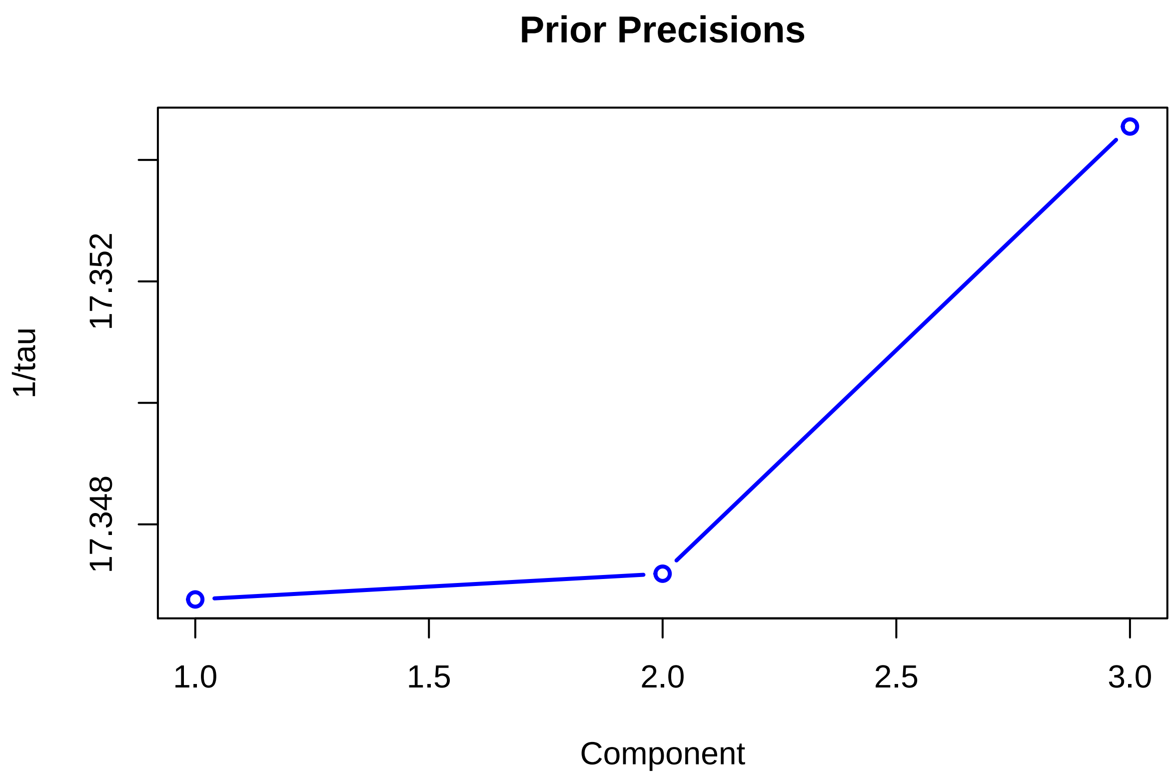

Figure 1: Prior variances for the first 3 components.

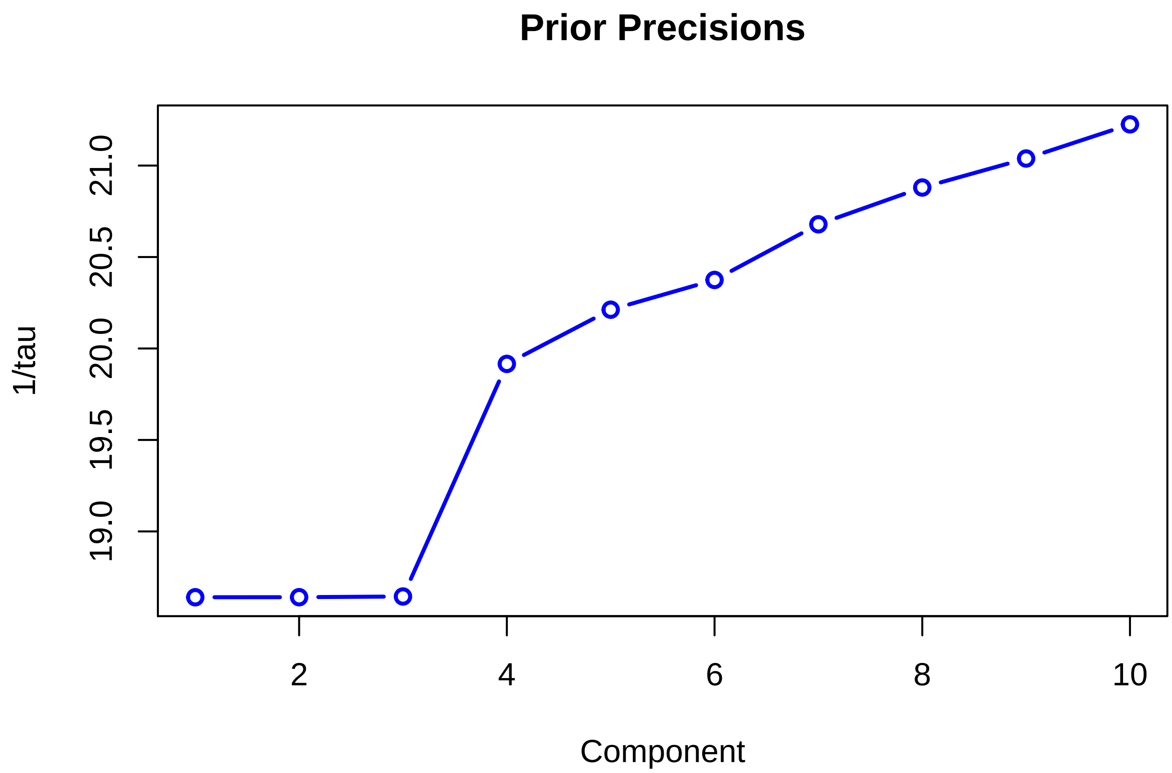

Figure 2: Elbow method for 10 components.

```
## variable 18  0.002919683  0.02319284 -0.4055783339
## variable 19  0.002019241  0.02192068 -0.4087014694
## variable 20  0.001874190  0.02043084 -0.4077300859
```

```
mod7$Tau
```

```
## [1] 17.34676 17.34719 17.35455
```

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 - see Figure 2):

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

```
## Warning: unscaled data - ELBO values might be positive.
```

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. Simulation studies have also shown that the method works better when Gamma priors are specified for τ.

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, or assigned to a Beta hyperprior:

- among the control parameters of `vbpca_control`, set `beta1pi` smaller than or equal to 0 for fixed π;
- 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`.

```r
# 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 )
```

```
## Warning: unscaled data - ELBO values might be positive.
```

```r
mod9$muW
```

```
##              Component 1 Component 2  Component 3
## variable 1  -0.376748781 -0.04448419 -0.004118464
## variable 2  -0.372475633 -0.04361763 -0.005760081
## variable 3  -0.375667438 -0.04356739 -0.004991567
## variable 4  -0.373841437 -0.04448232 -0.004702291
## variable 5  -0.377538204 -0.04319580 -0.005460548
## variable 6  -0.375841926 -0.04472555 -0.004526083
## variable 7  -0.375415421 -0.04399647 -0.004250611
## variable 8   0.044408685 -0.37703894 -0.019875476
## variable 9   0.043998937 -0.37297946 -0.020143278
## variable 10  0.043266722 -0.37566487 -0.019143767
## variable 11  0.043857102 -0.37129411 -0.019783811
## variable 12  0.044746209 -0.37519130 -0.019756230
## variable 13  0.045169995 -0.37456826 -0.019725427
## variable 14  0.043746355 -0.37732922 -0.018255778
## variable 15  0.002623419  0.02125881 -0.405658519
## variable 16  0.002688735  0.02140113 -0.407985305
```

```
## variable 17  0.002848870  0.02135504 -0.409081847
## variable 18  0.002537808  0.02133019 -0.406355278
## variable 19  0.002543663  0.02127051 -0.408688560
## variable 20  0.002678639  0.02114513 -0.408149455
```

```
# SVS, priorInclusion with Beta(1,1) priors and InverseGamma(5, 1) for tau, v0 = .005
ctrl6 <- vbpca_control(center = FALSE, scalecorrection = -1,
                       plot.lowerbound = FALSE, alphatau = 5,
                       betatau = 1, beta1pi = 1, beta2pi = 1,
                       v0 = 5e-03)


# Estimate the model
mod10 <- vbpca(X, D = 3, maxIter = 1e+03, priorvar = 'invgamma',
               SVS = TRUE, priorInclusion = 0.5, control = ctrl6,
               verbose = FALSE )
```

```
## Warning: unscaled data - ELBO values might be positive.
```

```
mod10$muW
```

```
##              Component 1 Component 2  Component 3
## variable 1  -0.376819427 -0.04443455 -0.004104646
## variable 2  -0.372347064 -0.04359019 -0.005704779
## variable 3  -0.375713290 -0.04343755 -0.004950714
## variable 4  -0.373896945 -0.04418555 -0.004672391
## variable 5  -0.377493453 -0.04329708 -0.005410811
## variable 6  -0.375803896 -0.04448535 -0.004502273
## variable 7  -0.375537932 -0.04396982 -0.004233235
## variable 8   0.044273658 -0.37707884 -0.019695715
## variable 9   0.043953150 -0.37295786 -0.019950557
## variable 10  0.043385804 -0.37573646 -0.018994338
## variable 11  0.043791774 -0.37127385 -0.019600790
## variable 12  0.044489314 -0.37517115 -0.019579772
## variable 13  0.044632773 -0.37453929 -0.019545883
## variable 14  0.043970055 -0.37745104 -0.018151342
## variable 15  0.002609329  0.02108215 -0.405658327
## variable 16  0.002665695  0.02120927 -0.407997632
## variable 17  0.002843748  0.02116543 -0.409083146
## variable 18  0.002534781  0.02112488 -0.406366622
## variable 19  0.002532807  0.02108929 -0.408695261
## variable 20  0.002668028  0.02094633 -0.408176063
```

The estimated posterior inclusion probabilities for the two models:

```
mod9$inclusionProbabilities
```

```
##             Component 1 Component 2 Component 3
## variable 1   1.00000000   0.2119197  0.09722059
## variable 2   1.00000000   0.2067622  0.09847080
## variable 3   1.00000000   0.2083132  0.09893331
## variable 4   1.00000000   0.2138161  0.09839703
## variable 5   1.00000000   0.2043346  0.09818925
## variable 6   1.00000000   0.2145334  0.09844897
## variable 7   1.00000000   0.2089924  0.09789326
## variable 8   0.21419852   1.0000000  0.11720313
## variable 9   0.21103005   1.0000000  0.11749524
```

```
## variable 10  0.20602971   1.0000000  0.11538121
## variable 11  0.21062206   1.0000000  0.11704114
## variable 12  0.21585734   1.0000000  0.11699144
## variable 13  0.21973753   1.0000000  0.11755002
## variable 14  0.20652527   1.0000000  0.11220954
## variable 15  0.09656056   0.1178761  1.00000000
## variable 16  0.09698534   0.1193360  1.00000000
## variable 17  0.09664596   0.1184619  1.00000000
## variable 18  0.09701643   0.1191947  1.00000000
## variable 19  0.09630871   0.1177812  1.00000000
## variable 20  0.09738379   0.1190830  1.00000000
```

```r
mod10$inclusionProbabilities
```

```
##             Component 1 Component 2 Component 3
## variable 1   1.00000000  0.14841902  0.06706106
## variable 2   1.00000000  0.14475267  0.06790906
## variable 3   1.00000000  0.14521657  0.06820936
## variable 4   1.00000000  0.14836490  0.06784986
## variable 5   1.00000000  0.14358706  0.06771620
## variable 6   1.00000000  0.14923530  0.06788454
## variable 7   1.00000000  0.14639379  0.06751311
## variable 8   0.14936431  1.00000000  0.08084268
## variable 9   0.14758001  1.00000000  0.08104081
## variable 10  0.14479747  1.00000000  0.07962609
## variable 11  0.14716229  1.00000000  0.08072651
## variable 12  0.14996793  1.00000000  0.08069827
## variable 13  0.15120628  1.00000000  0.08105917
## variable 14  0.14588418  1.00000000  0.07751491
## variable 15  0.06659518  0.08138748  1.00000000
## variable 16  0.06687420  0.08234224  1.00000000
## variable 17  0.06664933  0.08175705  1.00000000
## variable 18  0.06689692  0.08222256  1.00000000
## variable 19  0.06642465  0.08131492  1.00000000
## variable 20  0.06714090  0.08213839  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`:

```r
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,
      main = "Estimated Inclusions", xlab = "",
      col = RColorBrewer::brewer.pal(9, "Blues"))
```

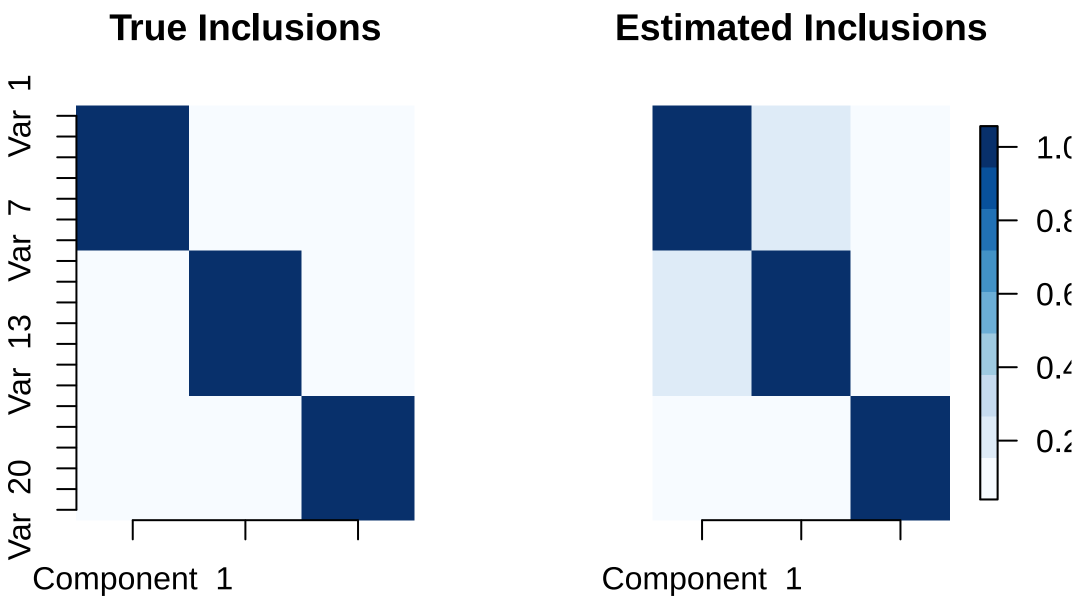

Figure 3: True and Estimated inclusion probabilities.

```
axis(side = 1, at = 1:3, labels = paste("Component ", 1:3 ))
```

We can observe the estimated prior inclusion probabilities for `mod10`:

```
mod10$priorInclusion
```

```
##           [,1]
## [1,] 0.4030537
## [2,] 0.4030537
## [3,] 0.4030537
```

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 Beta priors, all elements of `beta1pi` must be larger than 0. Let us look at one example:

```
# Beta priors with different degrees of sparsity for each component
ctrl7 <- 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
mod11 <- vbpca(X, D = 3, maxIter = 1e+03, priorvar = 'invgamma', SVS = TRUE,
               priorInclusion = rep(0.5, 3), control = ctrl7, verbose = FALSE )
```

```
## Warning: unscaled data - ELBO values might be positive.
```

```r
mod11$muW
```

```
##              Component 1 Component 2  Component 3
## variable 1  -0.376819438 -0.04445502 -0.004064610
## variable 2  -0.372345225 -0.04357450 -0.005755290
## variable 3  -0.375713439 -0.04342462 -0.004971688
## variable 4  -0.373896322 -0.04422102 -0.004667658
## variable 5  -0.377493429 -0.04325300 -0.005451157
## variable 6  -0.375803731 -0.04452566 -0.004484299
## variable 7  -0.375538283 -0.04396613 -0.004201703
## variable 8   0.044275871 -0.37707088 -0.019738172
## variable 9   0.043955770 -0.37297606 -0.020022566
## variable 10  0.043389314 -0.37570390 -0.018975810
## variable 11  0.043794531 -0.37128339 -0.019653649
## variable 12  0.044491305 -0.37518859 -0.019616159
## variable 13  0.044634385 -0.37456385 -0.019588044
## variable 14  0.043973315 -0.37741367 -0.018035232
## variable 15  0.002609665  0.02109798 -0.405684026
## variable 16  0.002666015  0.02122927 -0.407991886
## variable 17  0.002844155  0.02118225 -0.409091308
## variable 18  0.002535116  0.02114880 -0.406377061
## variable 19  0.002533189  0.02110572 -0.408701413
## variable 20  0.002668477  0.02096725 -0.408128069
```

```r
mod11$priorInclusion
```

```
##           [,1]
## [1,] 0.4016637
## [2,] 0.4443057
## [3,] 0.5816101
```

```r
mod11$inclusionProbabilities
```

```
##             Component 1 Component 2 Component 3
## variable 1   1.00000000  0.17340247   0.1318316
## variable 2   1.00000000  0.16897004   0.1335467
## variable 3   1.00000000  0.16958070   0.1342278
## variable 4   1.00000000  0.17347654   0.1334771
## variable 5   1.00000000  0.16748789   0.1331657
## variable 6   1.00000000  0.17448645   0.1335521
## variable 7   1.00000000  0.17093139   0.1327720
## variable 8   0.14858441  1.00000000   0.1580473
## variable 9   0.14681126  1.00000000   0.1584393
## variable 10  0.14404661  1.00000000   0.1555414
## variable 11  0.14639585  1.00000000   0.1578498
## variable 12  0.14918457  1.00000000   0.1577659
## variable 13  0.15041441  1.00000000   0.1585751
## variable 14  0.14512881  1.00000000   0.1511230
## variable 15  0.06623074  0.09549546   1.0000000
## variable 16  0.06650808  0.09663596   1.0000000
## variable 17  0.06628454  0.09593692   1.0000000
## variable 18  0.06653064  0.09650339   1.0000000
## variable 19  0.06606118  0.09541130   1.0000000
```

```
## variable 20  0.06677322  0.09640743   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. *Note*: when normalised weights are require from the corresponding `vbpca_control` argument, the posterior density interval will still be returned in the original weights scale (thus, no normalisation is performed on the HPDIs).

```
# Set hyperparameter values and require 50% probability density intervals
ctrl8 <- vbpca_control(center = FALSE, scalecorrection = -1,
                       plot.lowerbound = FALSE,
                       alphatau = 2, betatau = .5,
                       hpdi = TRUE, probHPDI = 0.5)

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

```
## Local prior variances : Inverse-Gamma, fixed hyperparameters.

## Warning: unscaled data - ELBO values might be positive.

## Iteration: 1 - ELBO: -2803.47
## Start # 1 has converged in 2 iterations; lower bound = -2802.92
```

```
# Plot HPD intervals for variables 1:10, component 1
plothpdi(mod12, d = 1, vars = 1:10)
```

Retrieve Principal Components

To compute the estimated components, simpy call:

```
PCs <- X %*% mod1$muW
head(PCs, 15)
```

```
##       Component 1 Component 2 Component 3
##  [1,]   -59.19132  -78.592707  31.3401056
##  [2,]    28.97173 -118.789002 -29.0200803
##  [3,]   -11.00518   14.227039  -4.8429367
##  [4,]    92.16140  -33.606390 -28.1184509
##  [5,]   -41.61482 -212.440559  13.4800647
##  [6,]   113.51610  -20.107248   5.6778548
##  [7,]    98.45308  -73.892683  17.2711826
##  [8,]    42.05467 -142.922658 -68.0937551
##  [9,]   -57.38540  -66.586047  17.5396918
## [10,]    42.94090   51.286634  -0.2553017
## [11,]    36.39523  -11.871548  13.9383095
## [12,]   109.60474   -6.656482  25.3900580
## [13,]  -196.01791  110.020825  -9.5996919
## [14,]  -267.42318   71.336729  14.1676697
## [15,]    38.49334   22.034659 -32.6994089
```

References

1. C. M. Bishop. 'Variational PCA'. In Proc. Ninth Int. Conf. on Artificial Neural Networks. ICANN, 1999.

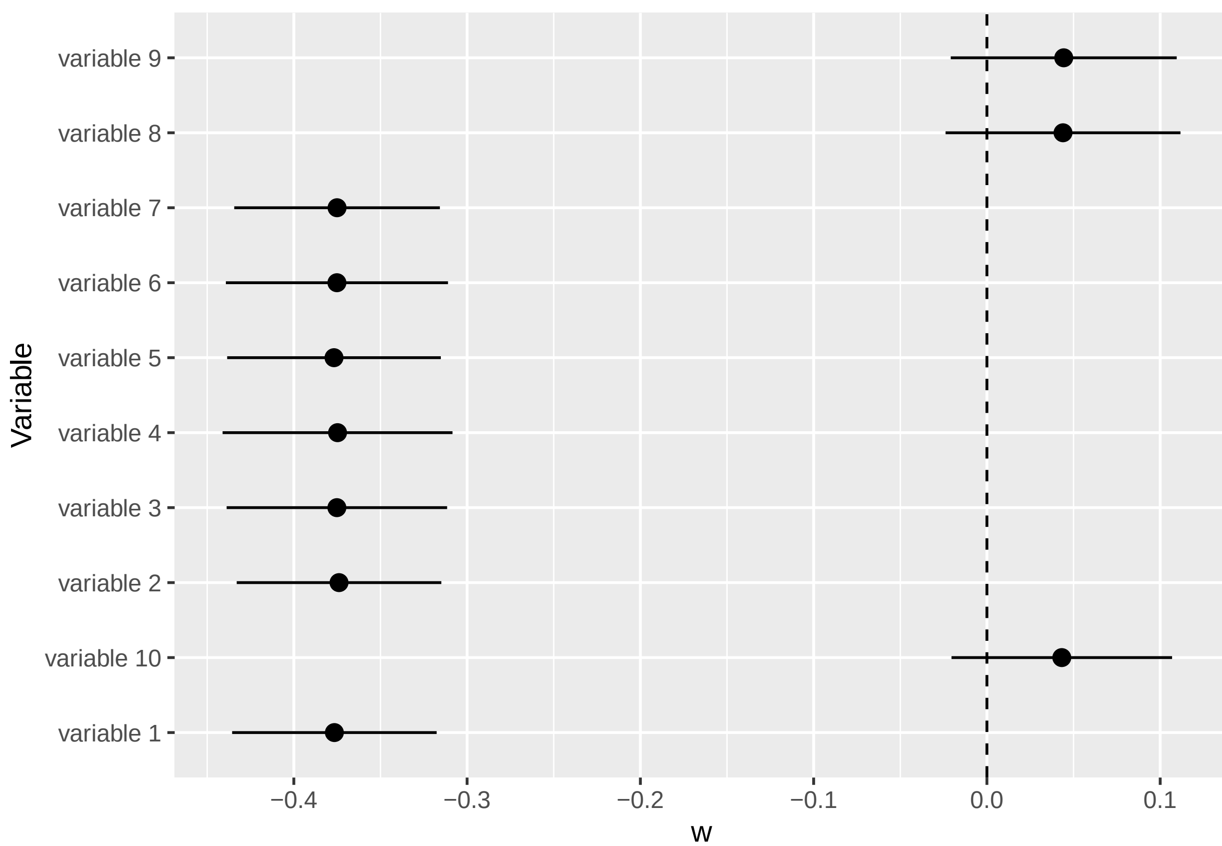

Figure 4: High posterior density intervals.

2. E. I. George, R. E. McCulloch (1993). 'Variable Selection via Gibbs Sampling'. Journal of the American Statistical Association (88), 881-889.