bayespca Package

Davide Vidotto d.vidotto@uvt.nl

2020-06-02

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 automatic relevance determination (ARD).

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:

$$w_d \sim MVN(0, T_d^{-1})$$

where $MVN()$ denotes the density of the Multivariate Normal Matrix, and T_d denotes the prior (diagonal) precision 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;
- `nstart` number of times a different run of the algorithm is performed (with varying starting values);
- `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`;
- `center` boolean indicating whether to center the variables in X;
- `scalecorrection`, a float which is >= than 0 if the variables are scaled (by a factor `scalecorrection`), and <0 otherwise;
- `svdStart`, a boolean denoting whether to use the values of SVD decomposition for the starting values (opposed to random starts);
- `tau` value of the prior precisions; starting value when `updatetau=TRUE` or `alphatau > 0` (for Gamma priors)
- `updatetau` logical parameter denoting whether the prior precisions should be updated when `priorvar='fixed'`;
- `alphatau` values of the shape parameter of the Gamma priors for the precisions; prior precisions are fixed when set to 0
- `betatau` values of the scale parameter of the Gamma priors for the precisions;
- `plot.lowerbound` boolean indicating whether to plot the history of the ELBO values calculated during the variational iterations;

- `hpdi` logical indicating whether to calcualte the HPD intervals of the weights;
- `probHPDI` the probability density covered by the HPDI's;

- `global.var` logical parameter which activates component-specific prior variances when set to `TRUE`;

`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. When `nstart`>1, the algorithm will autmatically pick (and output) the best run in terms of final ELBO value.

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)

# Estimate vbpca with fixed prior precisions (equal to 1)
# for the elements of W
mod1 <- vbpca(X, D = 3, maxIter = 1e+03, alphatau=0,
              center = FALSE, scalecorrection = -1,
              plot.lowerbound = FALSE,
              verbose = FALSE )

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

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

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

```
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:

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

```
mod2 <- vbpca(X, D = 3, maxIter = 1e+03, alphatau=0,
             updatetau = TRUE, center = FALSE,
             scalecorrection = -1,
             plot.lowerbound = FALSE,
             verbose = FALSE )
```

```
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 prior precisions 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: Gamma prior

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

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

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

```
# Estimate the model
mod3 <- vbpca(X, D = 3, maxIter = 1e+03,
              alphatau = 2, betatau = .5,
              center = FALSE, scalecorrection = -1,
              plot.lowerbound = FALSE,
              verbose = FALSE )
mod3$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
```

```
mod3$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 Gamma will have component-specific hyperparameters:

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

.

```
# Estimate the model
mod4 <- vbpca(X, D = 3, maxIter = 1e+03,
              center = FALSE, scalecorrection = -1,
              alphatau = c(.5, 50, 3), betatau = c(.5, .01, 10),
              plot.lowerbound = FALSE,
              verbose = FALSE )
```

```
mod4$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
```

```
mod4$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
```

Global prior variances

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

$$w_{j,d} \sim N(0, \tau_{j,d}^{-1}).$$

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

$$w_{j,d} \sim N(0, \tau_d{-}1)$$

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

```
# Fixed prior global variances, updated via Type-II maximum likelihood:
mod5 <- vbpca(X, D = 3, maxIter = 1e+03, alphatau=0,
              updatetau = TRUE,
              center = FALSE, scalecorrection = -1,
              plot.lowerbound = FALSE,
              verbose = FALSE, global.var = TRUE)
```

```
mod5$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
```

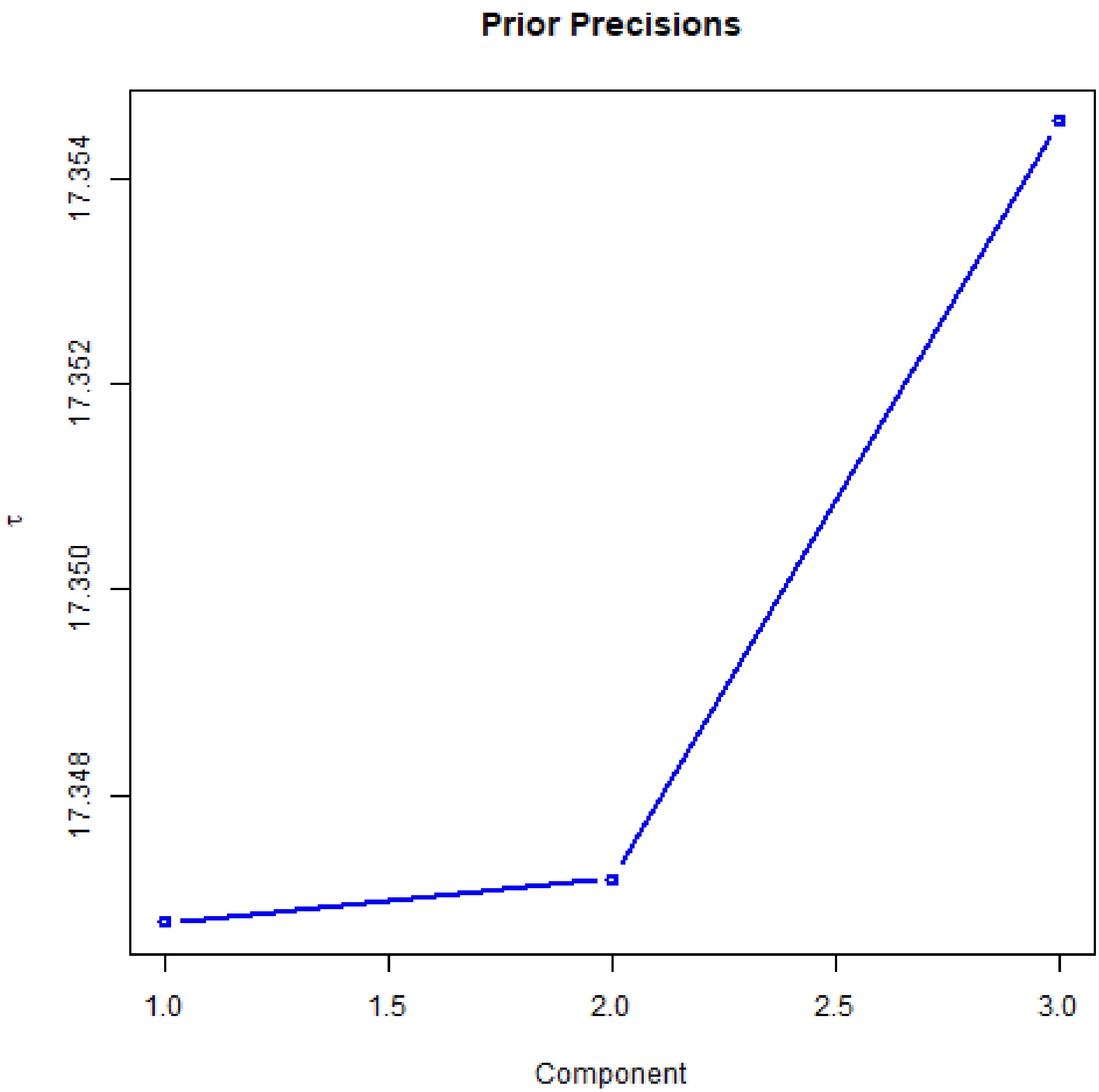

Figure 1: Prior precisions for the first 3 components.

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

```
mod5$Tau
```

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

Notice the plot of the precisions that appears in this case. This is useful when the number of components supported by the data is uncertain (scree-plot - see Figure 2):

```
mod6 <- vbpca(X, D = 10, maxIter = 1e+03, alphatau=0,
              updatetau = TRUE,
              center = FALSE, scalecorrection = -1,
              plot.lowerbound = FALSE,
              verbose = FALSE, global.var = TRUE)
```

```
mod6$Tau
```

```
##  [1] 18.64014 18.64036 18.64422 19.91587 20.21201 20.37469 20.67901
##  [8] 20.87993 21.03862 21.22528
```

Automatic Relevance Determination

When the prior precisions are updated, they can help to perform component-specific variable selection through Automatic Relevance Determination (ARD). In particular, values in the Tau matrix that are extremely large determine the values of the weights that can be set to 0. This is because their inverse (prior variances) are very close to 0, and thus making the elements of W also close to 0 with high probability. We're going to show an example here, with fixed (updated through Type-II maximum likelihood) precisions (in case of Gamma prior, it is recommended to use hyperparamter values close to 0).

```
mod7 <- vbpca(X, D = 3, maxIter = 1e+03, alphatau=0,
              updatetau = TRUE, center = FALSE,
              scalecorrection = -1,
              plot.lowerbound = FALSE,
              verbose = FALSE)
```

```
mod7$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
```

```
mod7$Tau
```

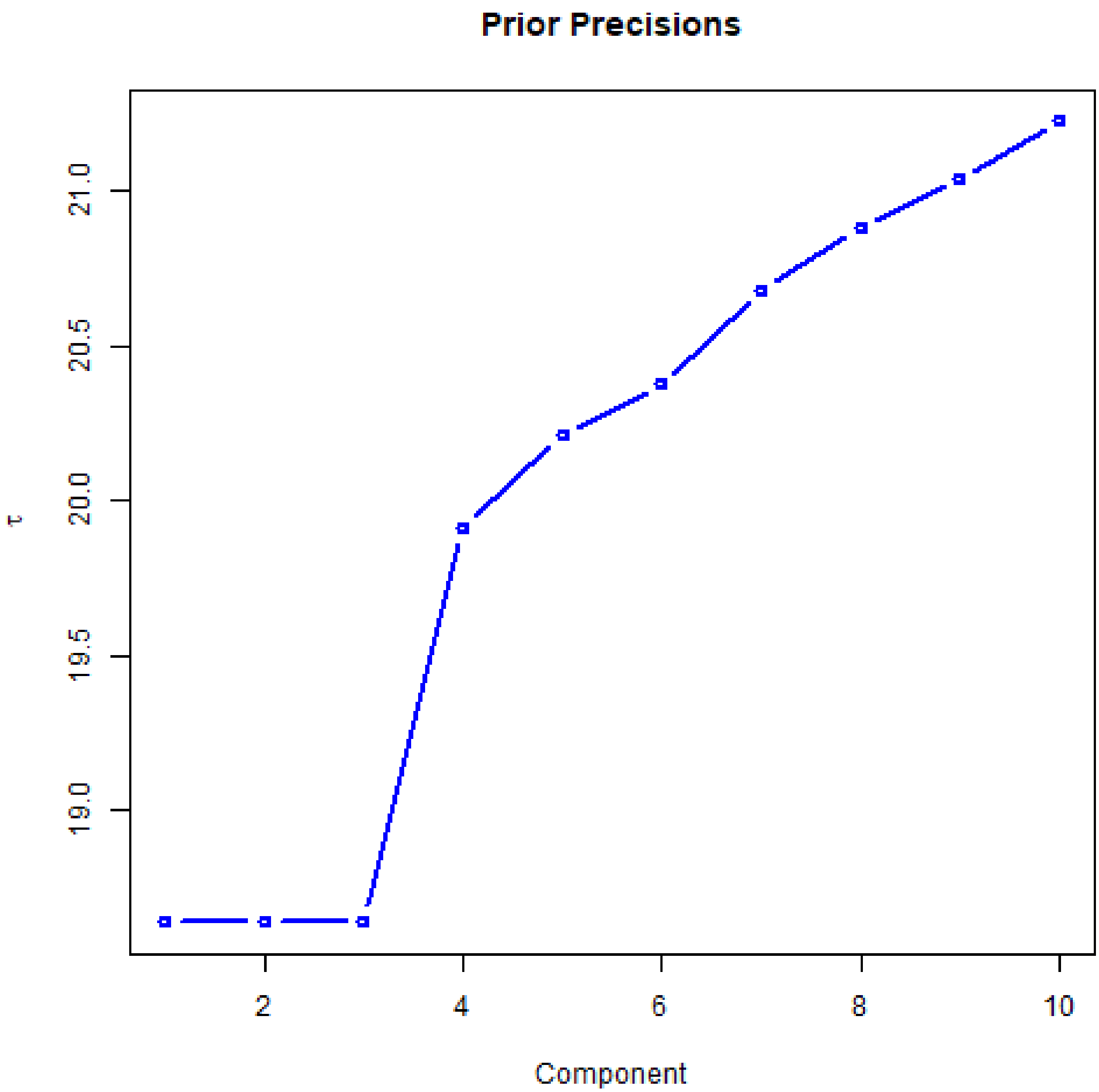

Figure 2: Scree-plot for 10 components.

```
##               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
```

We can also plot an heatmap of the resulting precision matrix; we set a threshold parameter (to establish when the precision elements are too large) equal to 50:

```
mat_mod_7 <- plotheatmap(mod7, matrix_type="Tau", bound_tau=50)
```

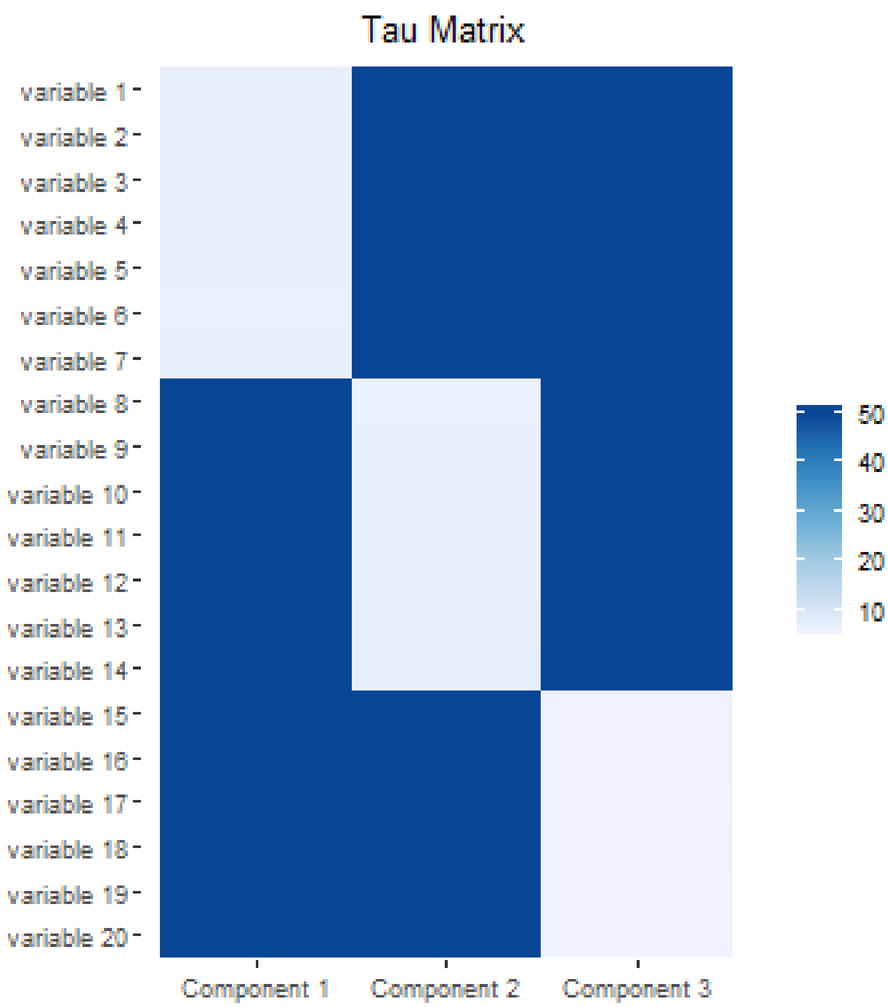

Figure 3: Heatmap of Tau.

```
mat_mod_7$W
```

```
##              Component 1 Component 2 Component 3
## variable 1   -0.3774720   0.0000000   0.0000000
## variable 2   -0.3744848   0.0000000   0.0000000
## variable 3   -0.3747070   0.0000000   0.0000000
## variable 4   -0.3697882   0.0000000   0.0000000
## variable 5   -0.3794353   0.0000000   0.0000000
## variable 6   -0.3811333   0.0000000   0.0000000
## variable 7   -0.3709497   0.0000000   0.0000000
## variable 8    0.0000000  -0.3881714   0.0000000
## variable 9    0.0000000  -0.3766366   0.0000000
## variable 10   0.0000000  -0.3769402   0.0000000
## variable 11   0.0000000  -0.3744147   0.0000000
## variable 12   0.0000000  -0.3743813   0.0000000
## variable 13   0.0000000  -0.3700551   0.0000000
## variable 14   0.0000000  -0.3647296   0.0000000
## variable 15   0.0000000   0.0000000  -0.4060117
## variable 16   0.0000000   0.0000000  -0.4070776
## variable 17   0.0000000   0.0000000  -0.4111599
## variable 18   0.0000000   0.0000000  -0.3987884
## variable 19   0.0000000   0.0000000  -0.4105380
## variable 20   0.0000000   0.0000000  -0.4127583
```

```
mat_mod_7$Tau
```

```
##              Component 1 Component 2 Component 3
## variable 1     6.711653   50.000000   50.000000
## variable 2     6.811715   50.000000   50.000000
## variable 3     6.733579   50.000000   50.000000
## variable 4     6.906649   50.000000   50.000000
## variable 5     6.630610   50.000000   50.000000
## variable 6     6.535962   50.000000   50.000000
## variable 7     6.916642   50.000000   50.000000
## variable 8    50.000000    6.283433   50.000000
## variable 9    50.000000    6.662857   50.000000
## variable 10   50.000000    6.647409   50.000000
## variable 11   50.000000    6.740676   50.000000
## variable 12   50.000000    6.710938   50.000000
## variable 13   50.000000    6.828244   50.000000
## variable 14   50.000000    7.207997   50.000000
## variable 15   50.000000   50.000000    5.837005
## variable 16   50.000000   50.000000    5.771306
## variable 17   50.000000   50.000000    5.676298
## variable 18   50.000000   50.000000    5.997557
## variable 19   50.000000   50.000000    5.710167
## variable 20   50.000000   50.000000    5.591906
```

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 the weights are required in normalised form, 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 90% probability density intervals
# Estimate the model
mod8 <- vbpca(X, D = 3, maxIter = 1e+03,
                alphatau = .001, betatau = .001,
                center = FALSE, scalecorrection = -1,
                hpdi = TRUE, probHPDI = 0.9,
                plot.lowerbound = FALSE,
                verbose = TRUE )
```

```
## Local prior variances : Gamma.
## Iteration: 1 - ELBO: -3088.1
## Start # 1 has converged in 4 iterations; lower bound = -3077.58
```

```r
# Plot HPD intervals for variables 1:10, component 1
plothpdi(mod8, d = 1, vars = 1:20)
```

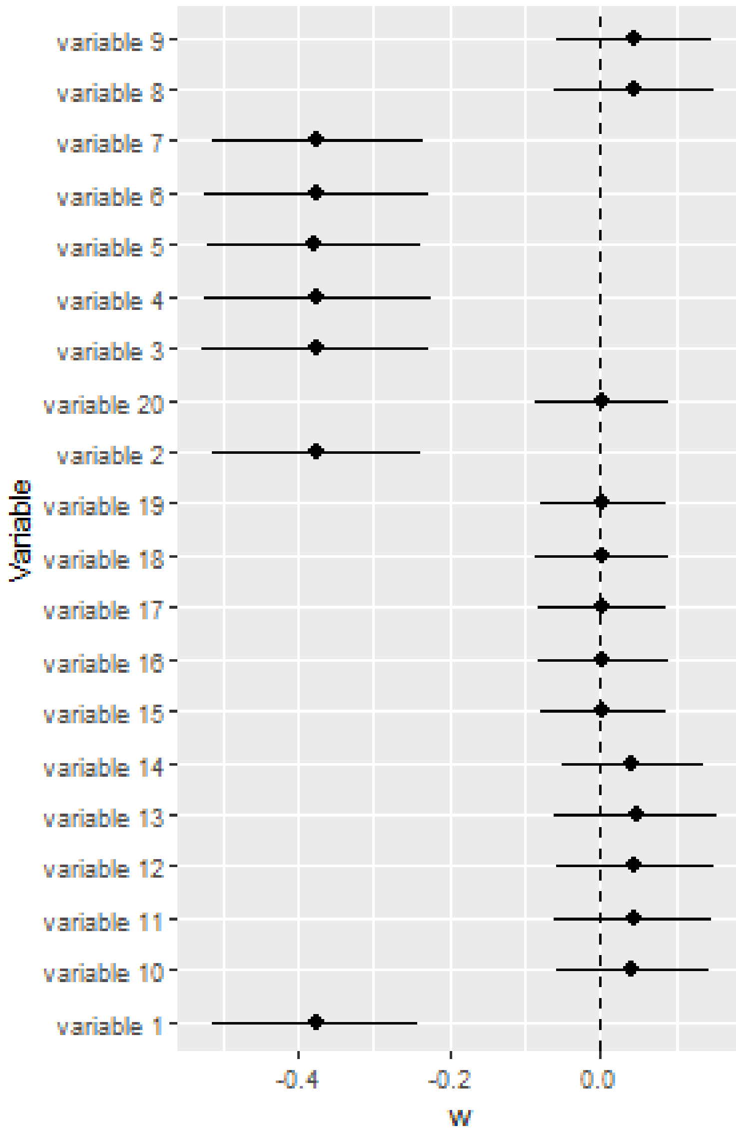

Figure 4: High posterior density intervals.

Retrieve Principal Components

To compute the estimated components, simpy call:

```r
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.