Probabilistic Principal Component Analysis for the analysis of the performance of combined events athletes

Blai Ras, David Farré, Irene Bonafonte

June 2021

Contents

1 Introduction

1.1 Principal Component Analysis

Principal Component Analysis (PCA) is one of the most popular dimensionality reduction techniques, which is used in many fields for data compression, feature extraction and data visualization. Its purpose is to find, from a data matrix $X \in \mathbb{R}^{d \times n}$ of samples n and features d, the directions that capture the maximal amount of variation in a dataset —or, in other terms, those that minimize the distance between the data points and their linear projections—. This is particularly useful for highly dimensional datasets, were we would like to be able to represent our data using less variables. What PCA does is to project the data into a principal subspace, in which all the components are orthogonal and capture, consecutively, the maximal amount of variance possible.

PCA can be defined in different terms, but we will use the "maximum variance formulation" (as referred in the Bishop's book) to briefly explain the principles behind PCA.

1.1.1 Traditional Approach

The idea behind PCA is to find the linear combinations $v_1x, v_2x, ..., v_,x$ —principal components— that successively have maximum variance for the data, while being uncorrelated to the previous components [Jolliffe, 2005]. These m components, in fact correspond to the dominant eigenvectors of the covariance matrix of the data (those associated with the m largest eigenvalues). Below, we explain in more detail using Singular Value Decomposition (SVD) and what we have learned in the Numerical Linear Algebra course, the idea behind PCA and why, when using the eigenvectors from the covariance matrix associated to the largest eigenvalues, we get an orthogonal projection that maximizes the variance of the dataset.

The SVD is a very useful factorization for rectangular matrices ($d \geq n$) in which:

$$X = U\Sigma V^T$$

with $X \in \mathbb{R}^{d \times n}$. $U \in \mathbb{R}^{d \times d}$ and $V \in \mathbb{R}^{n \times n}$ are two orthogonal basis and $\Sigma \in \mathbb{R}^{d \times n}$ is divided in a diagonal block $\Sigma[1:n, 1:n]$ with the singular values σ_i in the diagonal and a zero block $\Sigma[d-n:d, d-n:d]$. The singular values are ordered in such way that $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$

In the reduced form of the SVD, with $rank(X) = r$, $X \in \mathbb{R}^{d \times n}$, $U \in \mathbb{R}^{d \times r}$, $V \in \mathbb{R}^{r \times n}$ and $\Sigma \in \mathbb{R}^{r \times r}$. This decomposition can be defined based on the diagonalization of X^TX and XX^T. As U and V are orthogonal and therefore $V^T = V^{-1}$ and $V^{-1}V = \mathbb{I}$, we have:

$$X^TX = (V\Sigma U^T)(U\Sigma V^T) = V\Sigma V^T$$

$$XX^T = (U\Sigma V^T)(V\Sigma U^T) = U\Sigma U^T$$

As U and V are orthogonal we can see that this corresponds in fact to the $Q\Lambda Q^T$ diagonalization of X^TX and XX^T respectively. Therefore, V corresponds to the eigenvectors of X^TX, which are the columns of Q, U corresponds to the eigenvectors of XX^T, and the singular values $\sigma_1, ..., \sigma_r$ correspond to square root of the ordered non-zero eigenvalues of X^TX and XX^T (which are the same).

To perform PCA, we can use SVD to project our matrix X in the orthogonal directions V_k^T. With $k = rank(X^T)$:

$$B_k = V_k^T X = \Sigma_k U_k^T$$

where B_k are the k Principal Components of X^T and V, Σ and U come from the reduced SVD of X^T:

$$X^T = U_k \Sigma_k V_k^T \Rightarrow X = (U_k \Sigma_k V_k^T)^T = V_k \Sigma_k U_k^T \Rightarrow V_k^T X = \Sigma_k U_k^T$$

The variance of a variable and the covariance between two variables are defined as:

$$\text{Var}(Y) = \frac{1}{n-1} \sum_i^n (y_i - \bar{y})$$

$$\text{Cov}(YZ) = \frac{1}{n-1} \sum_i^n (y_i - \bar{y})(z_i - \bar{z})$$

We can easily see that if we centre to the mean a matrix A by removing from each variable row the mean of that variable, we can get the covariance between all pair of variables by computing the covariance matrix:

$$C_A = \frac{1}{n-1} AA^T$$

The diagonal entries represent the variance of each variable and the other entries the covariance among the corresponding variables. For the case of B_k:

$$C_B = \frac{1}{n-1} B_k B_k^T = \frac{1}{n-1} (\Sigma_k U_k^T)(\Sigma_k U_k^T)^T = (\Sigma_k U_k^T)(U_k \Sigma_k) = \Sigma_k^2$$

As Σ_k is a diagonal matrix with the diagonal elements representing the ordered eigenvalues of B_k, we can see that the new directions (the principal components) are uncorrelated among them, and that each component captures more variation than the next. In conclusion, applying SVD to the covariance matrix $\frac{1}{n-1} XX^T$ we obtain a projection of the matrix X into a principal subspace

$$B_k = V_k^T X$$

in which all the directions are orthogonal and capture the maximal amount of variation. If we take the first m components of this subspace, we will maximize the variance in the projected data using m dimensions. As a final comment, we could do the same but based on the correlation matrix, by not only centring but also scaling our data (dividing by the standard deviation of the variable).

1.1.2 Probabilistic PCA (PPCA)

In [Tipping and Bishop, 1999], the authors formulate a generative latent variable model, the maximum likelihood solution of which corresponds to the principal sub-space of the dataset.

A latent variable model relates a set of d-dimensional data vectors x_n to a set of m latent variables z_n, with $m < d$, so that the latent variables represent the dataset in a more simple manner:

$$\mathrm{X} = \mathrm{W}\mathrm{z} + \mu + \epsilon$$

with the latent variables coming from an isotropic Gaussian distribution $z \sim N(0, \sigma I)$ and the error being also Gaussian $\epsilon \sim N(0, \psi)$, with ψ diagonal. W defines the *factor loadings*, which are used to project the latent representation of the samples to the data space. μ is a constant that will represent the mean of the data. We can see that, under this formulation, X is also normal $X \sim N(\mu, C) = N(\mu, \phi + \mathrm{W}\mathrm{W}^T)$: the variance of X is given by the covariance matrix of W. The model assumes that the observed variables X are conditionally independent given the latent variables z. Therefore, z models the dependencies between the observed variables and ϵ represents the independent noise.

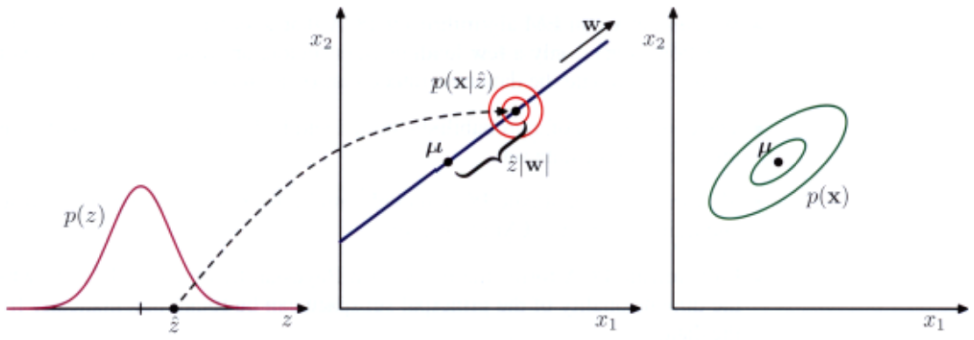


Figure 1: Illustration of the generative process in PPCA (from [Bishop, 2006]).

In [Tipping and Bishop, 1999], the authors show that, if we consider the noise to also follow an isotropic Gaussian, $\psi = \sigma^2 I$, the maximum-likelihood estimator W_{ML} corresponds in fact to the principal components of the data matrix. Additionally, they derive an computationally efficient Expectation Maximization (EM) algorithm which computes the maximum likelihood estimate of W, W_{ML}, and the rest of the parameters of the model. This allows to express PCA from a probabilistic point of view (see Figure 1), as a probability density model of the observed data. The reformulation of PCA in probabilistic terms, has several advantages over traditional PCA:

- It allows to measure the likelihood of the model, allowing to compare it to other models and facilitating statistical testing. This is, we can now test the uncertainty of the resulting model, or how well do the principal components represent each of our samples.
- The model can be extended to a mixture of PPCA models, by combining a collection of probabilistic PPCA models, which can also be trained using the EM algorithm.

- When estimating the model parameters using EM, we can also deal in an efficient manner with missing data.
- We can use PPCA to model class-conditional densities in a classification problem. This allows to use the posterior densities for classification, by evaluating the posterior probability of belonging to a class.
- We can use the trained PPCA model in a generative manner, to generate more samples from the distribution.

1.1.3 Bayesian PCA (BPCA)

An additional advantage of PPCA is that we can treat it using the Bayesian inferential framework. This opens up to many possibilities, through the ability to add priors to the model to include any previous information that we have. In [Bishop, 1999], the authors use the previous reformulation of PCA into PPCA, as the basis for a Bayesian treatment PCA (BPCA), by adding a prior to the model parameters $p(\mu, \mathrm{W}, \sigma^2)$. In particular, they focus on choosing a prior which helps addressing the issue of controlling the dimensionality of the latent space.

Up to this point, we have talked of a latent space of m dimensions, with $m < d$ (with d the dimension of the data space). However, we have not specified how should we choose the proper value of m. In traditional PCA, one can use certain rules, such as the elbow rule: the scree plot is used to detect the dimension at which the eigenvalues level off, and only the principal components associated to the previous eigenvalues are kept. However, we can clearly see that this test is very subjective and gives no measure on how optimal is the final choice of m. Another option, in the probabilistic setting, is to use cross-validation to compare the different possible values of m. However, this becomes untractable if we are using mixtures of PPCA models and need to choose a different m for each model.

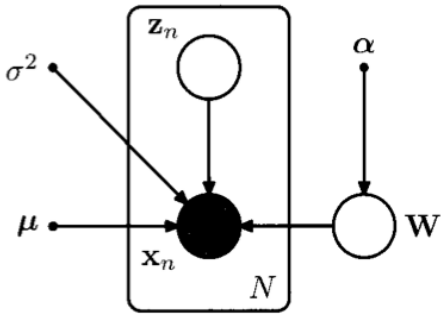


Figure 2: Illustration of the generative process in Bayesian PCA (from [Bishop, 2006]).

In [Bishop, 1999], the authors, inspired by the *automatic relevance determination* (ARD) framework, propose to use an ARD prior, consisting on continuous hyper-parameters that automatically determine the effective dimensionality of the latent space as part of the Bayesian inference process. This is achieved by introducing a hierarchical prior $\alpha = \alpha_1, ..., \alpha_m$ on the W matrix. Each

hyper-parameter α_i controls one of the columns of W through a conditional Gaussian distribution. The probabilistic structure of the model is shown in Figure 2.

For the model to be used, the posterior distribution of W needs to be marginalized. However, this is analytically intractable. In the publication, the authors propose an alternative way for computing the posterior by approximating it using a type-II maximum likelihood using a local Gaussian approximation. Another option would be to use a sampling strategy, using probabilistic programming languages such as Stan for inference using Markov Chain Monte Carlo algorithms, or to use variational inference.

1.2 Combined events athletes performance

In this work we intend to apply bayesian PCA to the analysis of the athletes performance in the combined events. The combined events are an athletic event in which the athletes compete in ten (in the case of men —decathlon—) or seven (in the case of women —heptathlon—) different track and field events in the course of two days.

This type of data has already been used in the past to evaluate important biological questions [Van Damme et al., 2002]. In particular, the cited paper was interested in the hypothesis that physical performance by vertebrates is constrained by a trade-off between antagonistic pairs of ecologically relevant traits and between conflicting specialist and generalist phenotypes. Using the performance of world-class decathletes and heptathletes —athletes who attempt to improve at the same time on different events, requiring each of these events the improvement of different abilities or traits—, is a rationale way to study this hypothesis. To put it in a simple manner, it is clear that an athlete requires certain abilities to perform at the 1,500m (endurance, speed endurance, running economy —which is coupled with a low weight—), another set of abilities to perform well in the shot put (power and strength, which are coupled with a high weight) and yet another set of abilities to perform well in pole vault (flexibility and motor skills, among other). Intuition says that some of these abilities are antagonic, meaning that improving in one leads to a detriment in the other, but this intuition needs to be formally tested.

When we start considering this problem, we can see that we start thinking of concepts such as interconnected "abilities", instead of independent track events. Our intuition says that there are a set of abilities which are required to perform in each event, and that some of these abilities are common between the events, some are not. We could think on the final performance in each event as the combination of a certain set of abilities, or latent factors, with each ability having a different weight in each event. This suggests that our data can be studied using principal component analysis, to derive the latent variables that are behind athletic performance, and how these contribute to the performance of each event, as done in [Fanshawe, 2012]. Studying this data using PCA will, in fact, also help us explore the previous hypothesis: each latent factor will capture one of these abilities. If there are antagonic abilities, we will see them captured in the same latent factor, with some of the events having a positive weight in that factor, and some of the other events having a negative weight. This approach may be more suitable than that use in [Van Damme et al., 2002]. In that study, they observed that their results were confounded by a global variable representing overall fitness. However, if we perform the analysis using PCA, if such a latent variable representing overall fitness exists, it will be captured in a particular component,

and the other components will reflect other characteristics. Using PCA will also be very useful because it will allow us to explore how many independent axis, or traits, are required to describe physical performance. For this purpose, it will particularly useful to use bayesian PCA, because we will be able to use an ARD prior so that the effective dimensionality of the latent space is revealed.

2 Methods

2.1 Datasets

To perform the study, we started by using data from two sources: packages from CRAN which contain various datasets of decathlons or heptathlons and datasets from papers recommended by the teacher. All these datasets refer to a particular competition, describing the results obtained by each athlete in each event of the competition. For these datasets, performance is described using a IAAF scoring table that measures how good a performance is.

Additionally, we have queried the Power of ten website, which has historical data of the performance of British athletes along the years, to obtain the performance of the top athletes in the all-time British ranking. For each athlete, we have obtained their best performance in each event over the course of years (not necessarily done in the same competition). This dataset will allow us to more accurately study the abilities of the athletes in the different events, and to study the progression of their latent variables over the course of time. In this case, we do not use the general scoring system but directly the performance values. This should be taken into account, because in distance metrics, the highest values are the best, whereas in time metrics, the lowest values are the best. The scripts `query_power_of_ten.py` and `preprocess_power_of_ten.r` download and preprocess this dataset. The main characteristics of the compiled datasets are summarized in Table 1.

Table 1: Description of the datasets used for the project.

Name	Source	Rows	Columns	Performance metric
decathlon2	Package factoextra	27	13	IAAF Score
2008 Woman Hepathlon	Paper	33	10	IAAF Score
Olympic Heptathlon Seoul '88	Package HSAUR	25	8	IAAF Score
Decathlon Austria 1982	Paper	20	9	IAAF Score
UK Decathlon performances (all time)	Power of ten	317	12	Time and distance
UK Heptathlon performances (all time)	Power of ten	565	9	Time and distance

2.2 Probabilistic and Bayesian PCA implementation

We have followed two different approaches to try to perform this analysis. First, in order to gain more insight on how the model actually works, we have tried to implement Bayesian PCA using

`Stan` and `tensorflow`. The details, problems and outcomes of these implementations are explained in section 3.1. Second, we have used some `R` packages in which probabilistic and bayesian PCA are already implemented. For comparison, we have also performed a traditional PCA analysis using the `prcomp` function from the package and the `factoextra` package for visualization.

2.2.1 Implementations from packages

After a brief search we find out that there are mainly two packages in R to perform PPCA.

- **pcaMethods**: R package for performing principal component analysis PCA with applications to missing value imputation. Provides a single interface to performing PCA using SVD, PPCA and others like NLPCA: Non-linear PCA. Our function call is the following: `ppca(Matrix, nPcs = 2, seed = NA, threshold = 1e-05, maxIterations = 1000)`. Same arguments but `bpca` in order to call the Bayesian PCA method. In this case, the Bayesian PCA refers to a method for dealing with data with missing values: EM is coupled with a Bayesian PCA model with an ARD prior to calculate the likelihood for a reconstructed value [Oba et al., 2003]. In our case, we do not have missing data, but the implementation should also work for fitting a bayesian PCA model using EM.
- **Rdimtools**: R package for dimension reduction (DR) - including feature selection and manifold learning - and intrinsic dimension estimation (IDE) methods. Our function call is the following: `do.ppca(X, ndim = 2)` and `do.bpca(X, ndim = 2)`. In this case, Bayesian PCA with an ARD prior as described in [Bishop, 1999] is implemented.

Both packages work pretty similar. Nevertheless, `pcaMethods` offers more personalization with the possibility of adding a threshold for convergence and a maximum number of iterations.

We created two separated notebooks in order to test each one. We tested both probabilistic PCA methods and Bayesian. Even though both packages return different parameters, we looked for the final distribution matrix, the loadings matrix and drew some recommended plots.

We then created the notebook `Package Comparison` where we put face to face both packages. We focused on comparing the probabilistic and bayesian method mainly through plots. We also tried some well-known metrics.

Finally, in order to visualize the Hinton diagrams, we saved the loadings matrix of each method of each package in `.csv` format. When looking at Hinton diagrams plot implementations, we found one done in Python inside the Matplotlib package[1]. Check all the diagrams in section 3.1.6 or inside the notebook `Hinton Diagrams`.

We also tried testing the package "bayesian-model-zoo", which has the function `ppaard` but it had a lot of dependencies and it has not been updated since 2018.

[1] Matplotlib is one of the most famous and reliable plotting packages avalaible in Python.

3 Results

3.1 Exploring the different implementations

Before doing the final analysis of the dataset, we have explored how the different methods work, how reliable are the obtained results and how easy it is to use them.

3.1.1 Tensorflow

We have used the TensorFlow package for python to solve the Probabilistic PCA of our data.

We define a model with variables W, Z, and $X = WZ$. The variables are initialized randomly according to a Normal distribution.

Then we use the Tensorflow minimization package to learn the W, Z that best fit the training data. This optimization is performed using the Adam Algorithm. We tried playing with the optimization parameters, but the results obtained were very similar, and always stabilized with some error. It is important to note that the model performed very poorly with unnormalized data.

When using the Bayesian Approach, we model the variables W, Z as Normal distributions, and initialize them with a random mean (as before), and standard deviation 1.

The optimization process is similar as before, but we learn the parameters of mean and standard deviation of W, Z, rather than just the mean. With this approach, we go from obtaining the MAP, to obtaining a full distribution of the posterior. This process is somewhat simplified by modeling the resulting Variables as Normal distributions as well, but still gives a broader picture than just computing the MAP.

3.1.2 Stan

We have tried to implement using Stan a sampling solution to the bayesian PCA problem. This should be quite stright forward, as one only needs to define the model and its priors as specified in [Bishop, 1999], and sample from the defined distribution to perform Bayesian inference on the desired parameters. To do these implementations, we have followed the theory and also some implementations that we have found online (see PPCA and BPCA). For these implementations, we have used the UK decathlon dataset.

First, as baseline comparison, we have implemented probabilistic PCA without adding ARD priors on W. The stan code for this implementation is shown below:

```
data{
    int<lower=1> m; //Number of dimensions of the latent space
    int<lower=1> d; //Number of dimensions of the data space
    int<lower=1> n; //Number of samples
    matrix[d,n] X; // data matrix
}

transformed data {
```

```
    //Zero vector for the mean of the Z prior
    vector<lower=0>[m] Mu_z = rep_vector(0, m);

    // Unit covariance matrix for the z prior
    matrix[m,m] Sigma_mu = diag_matrix(rep_vector(1, m));
}

parameters{
    // X distribution mu and covariance
    vector[d] Mu_x;
    real<lower=0> Sigma_x;

    // PCA matrices
    matrix[m,n] Z; //Latent matrix (one column per sample)
    matrix[d,m] W; //Predictors matrix
}

model{
    for (i in 1:n){
        Z[:,i] ~ multi_normal(Mu_z, Sigma_mu);
        X[:,i] ~ multi_normal(W * Z[:,i] + Mu_x, diag_matrix(Sigma_x));
    }
}
```

Next, we have modified the code to add an ARD prior to the W parameters:

```
data{
    int<lower=1> M; //Number of dimensions of the latent space
    int<lower=1> D; //Number of dimensions of the data space
    int<lower=1> N; //Number of samples
    matrix[D,N] X; // data matrix
}

transformed data {

    //Zero vector for the mean of the Z prior
    vector<lower=0>[M] Mu_z = rep_vector(0, M);

    // Unit covariance matrix for the z prior
    matrix[M,M] Sigma_mu = diag_matrix(rep_vector(1, M));

}

parameters{
    // X distribution mu and covariance
    real<lower=0> Sigma_x;

    // ARD prior for W
    vector<lower=0>[M] alpha;

    // PCA matrices
    matrix[D,M] W; //Predictors matrix
    matrix[M,N] Z; //Latent matrix (one column per sample)
}

transformed parameters{
    vector<lower=0>[M] t_alpha;
    real<lower=0> t_Sigma_x;
```

```
        t_alpha = inv(sqrt(alpha));
        t_Sigma_x = inv(sqrt(Sigma_x));
}

    model{
        Sigma_x ~ gamma(1,1);
        alpha ~ gamma(1e-3, 1e-3);

        for(m in 1:M) W[:,m] ~ normal(0, t_alpha[m]);

        for (n in 1:N){
            Z[:,n] ~ multi_normal(Mu_z, Sigma_mu); //0 mean, unit variance
            X[:,n] ~ multi_normal(W * Z[:,n], diag_matrix(rep_vector(t_Sigma_x, D)));
        }
    }
```

However, both of these models had severe problems. First, running them was extremely slow. Second, no matter how much we increased the number of iterations, we kept having problems due to a large R-hat values (not very large, but greater than > 1.01 for many samples) and low effective sample sizes. See for instance the histogram in Figure 3 of the effective sample size for each parameter after running the model for 7,500 iterations. For many parameters, we have < 75 effective samples, which is way below the expected 10% of effective samples. This could be due to the model not being well implemented, but we have also tried to execute the model from external sources (BPCA) and the problem persisted —although at least the execution of this model was faster—.

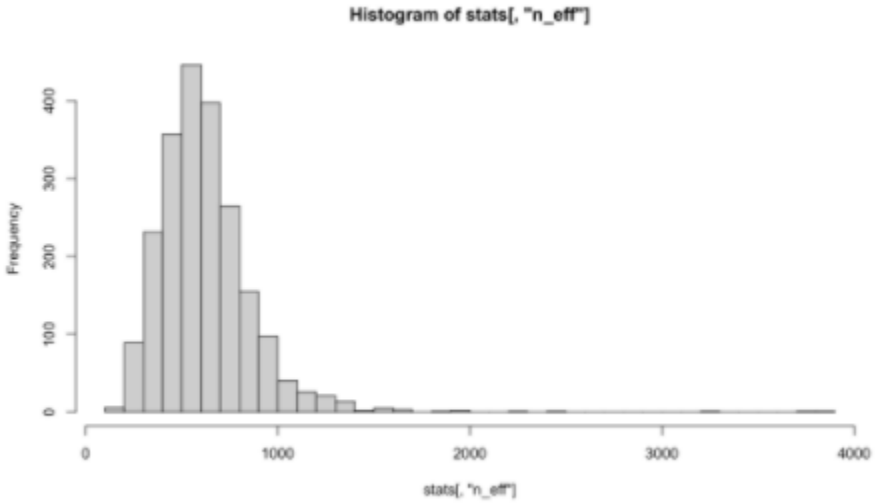


Figure 3: Effective sample size histogram for each parameter of the ARD-BPCA model after running it for 7,500 iterations.

To approximate the results, even with them being less reliable, we have also tried to run the Stan meanfield variational algorithm to approximate the posterior. However, also in this case our model (and the model found online) had convergence problems, with the pareto k diagnostic value being 3.56 on the best case. However, after greatly increasing the number of iterations (100,000)

and the tolerance for convergence (0.0001) the results were relatively good. As observed in 4, two components were selected as the effective dimensionality of the latent space. In this case, we observe a first component that seems to measure overall performance (take into account that in jumps and throws performance is measured in distance —the largest the better— and in races performance is measured in time —the lowest the better—). This component is quite general, but contributes a little bit less to the 1,500m performance and the throwing events (javelin throw and discus). This matches with our idea that these events are quite different from the rest. A second component seems to capture "bad" performance in throwing events, and is inversely correlated with performance in jumping and running events, again matching with our expectations.

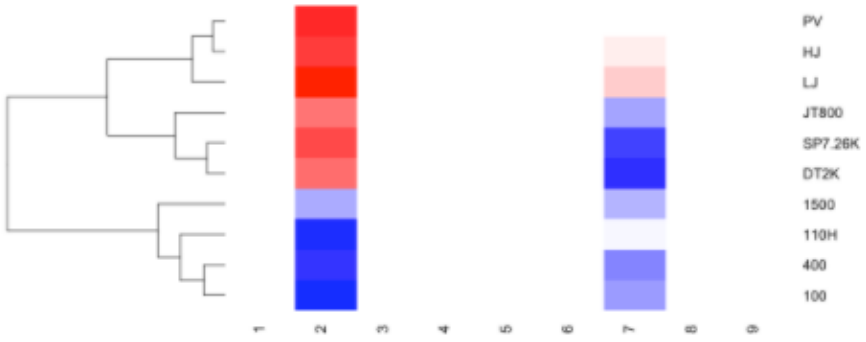


Figure 4: Loadings matrix (W) heatmap. Heatmap representing the loadings matrix W obtained by training the ARD BPCA model using variational inference. Red represents positive weights, blue represents negative weights.

Finally, it may be important to note that in our first attempts, we forgot to scale the dataset, and this lead to a really long execution times and no-convergence.

3.1.3 Rdimtools

Rdimtools proved to be fast and easy to use. We can't personalize as much as in pcaMethods package though. We tried comparing its conventional PCA method, its probabilistic approach and finally its bayesian PCA approach using the `2008WomanHeptathlon` dataset. We can see the result in Figure 5, where the PCA and probabilistic approaches are pretty similar but the bayesian one is totally weird.

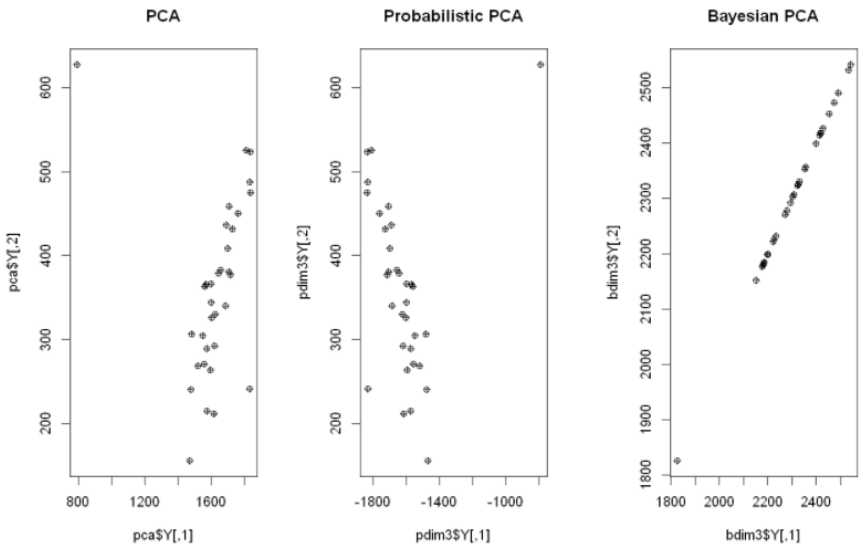


Figure 5: Rdimtools PCA methods in dimension 2

3.1.4 pcaMethods

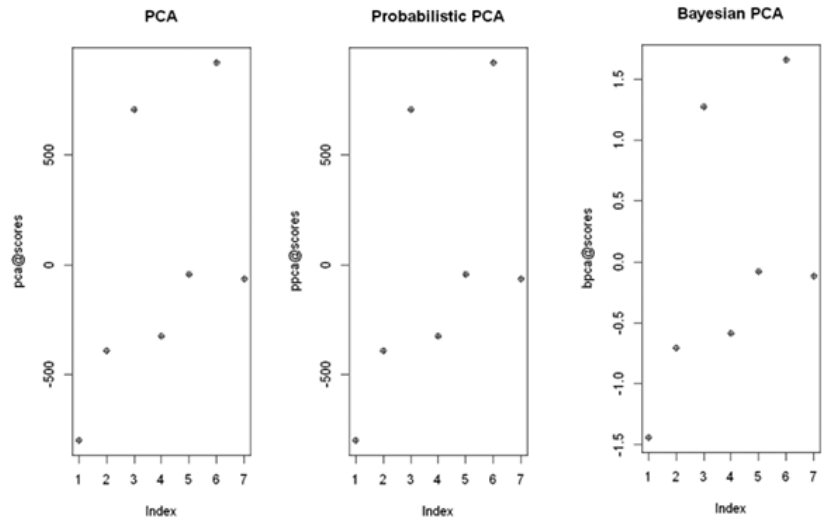


Figure 6: pcaMethods PCA results in dimension 1

pcaMethods is also fast and easy to use, but it has a better documentation and it is very complete. We also compared all of its 3 methods using the recommended plots that you can find in the

documentation. In Figure 7, for example, we can look at each distribution of scores for every dimension.

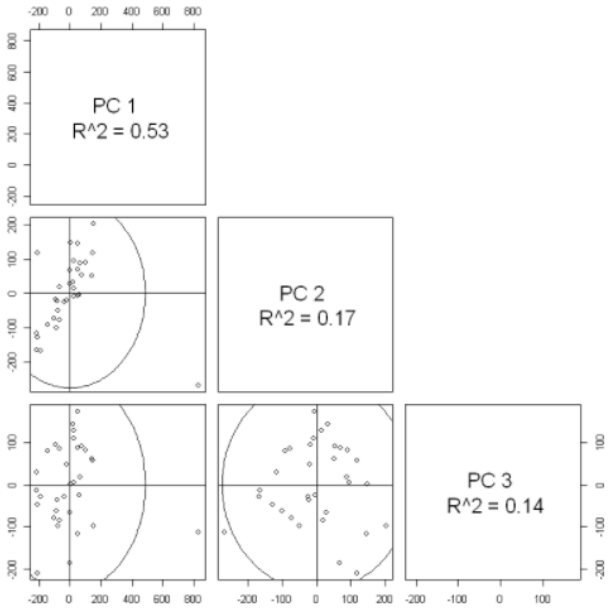


Figure 7: Distribution plot of each dimension with their R^2 Score

The following in Figure 8 plot is more interesting. This `slplot` tells us the percentage of explained variance and plot our matrix scores followed by the distribution of our loadings matrix. We can see that we get more or less the same explained variance with each method.

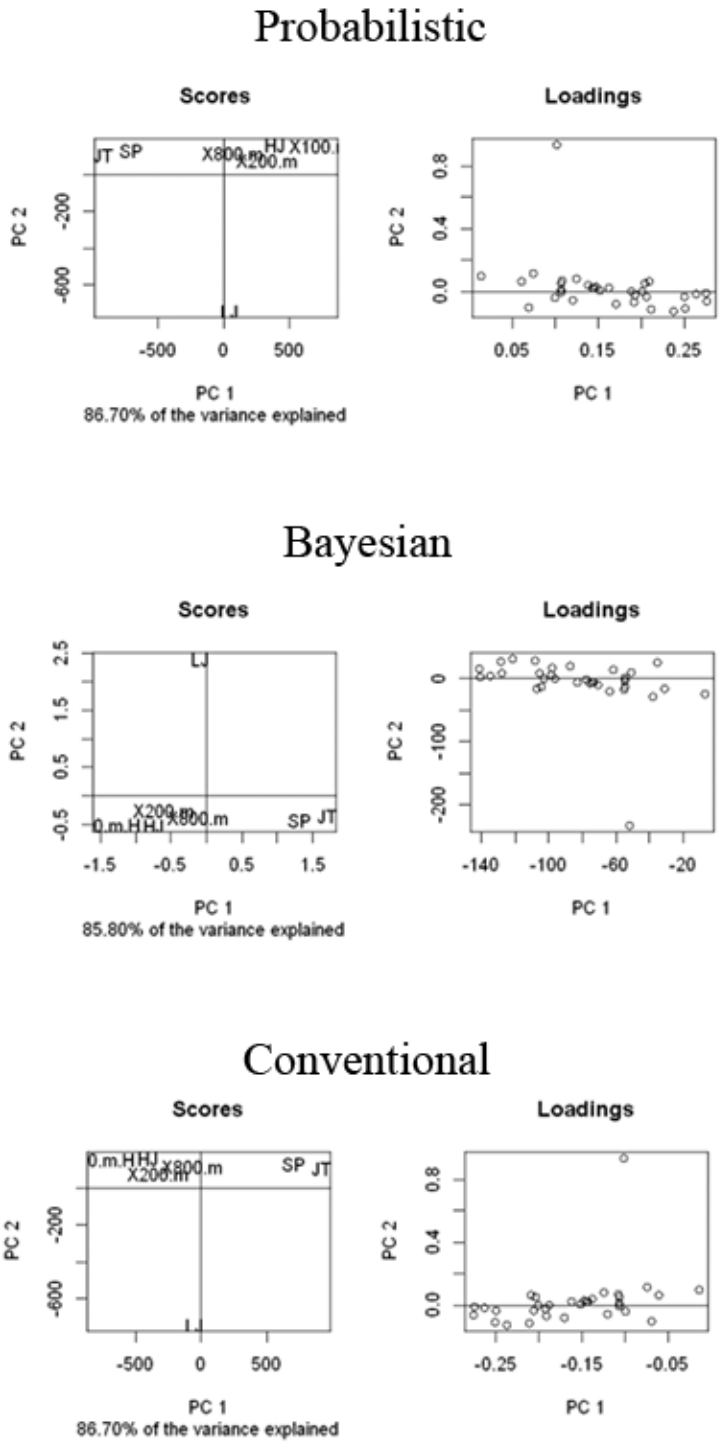


Figure 8: Explained Variance of each method with the package pcaMethods

3.1.5 Package comparison

When comparing PCA and PPCA results, plots are our most useful tools. Nevertheless, we though of using famous error metrics such as RMSE (Root Mean Square Error) and MedianAPE (Median Absolute Percentage Error loss). We also use the R^2 (Coefficient of Determination) Regression Score, in order to measure how close the data is fitted to the regression line.

Even though this evaluation metrics are often used in order to check for accuracy and predictions error in Machine Learning contexts, we believed that they can be informative because we are "putting a number" to this dissimilarities between different implementations. Just remark that we used this metrics that usually take as parameter the "ground truth" and the "predictions" and in our context there is no "y_pred" or "y_true".

We defined these mentioned metrics using the following implementations:

$$\textbf{RMSE:} \sqrt{\frac{\sum_{i=1}^{N} (x_i - \hat{x_i})^2}{N}}$$

$$\textbf{MedianAPE:} \ \text{median}\Big(\sum_{t=1}^{n} | \frac{x_i - \hat{x_i}}{x_i} | \Big)$$

$$\textbf{MeanAPE:} \ \frac{1}{n} \sum_{t=1}^{n} | \frac{x_i - \hat{x_i}}{x_i} |$$

$$\textbf{R-Squared Regression Score:} \ 1 - \frac{\sum_{t=1}^{n} (x_i - \hat{x_i})^2}{\sum_{t=1}^{n} (x_i - (\frac{1}{n} x_i))^2}$$

Probabilistic approach

Here are our overall results on the dataset `Decathlon_Austria82` when performing the probabilistic PCA:

Number of components to estimate	**RMSE**	**MedianAPE**	**MAPE**	R^2 **Regression Score**
1	52,47	0,02	0,02	0,84
3	230,8	0,12	0,91	0,96
5	272	2,03	1,37	0,91

Looking at the RMSE we can see bigger numbers of what we are used to. This is totally normal, due to the fact that (i) PPCA has a random factor in between and (ii) they are two different implementations (even though they follow the same theory principle).

Therefore, in our opinion MedianAPE is a more suitable metric when comparing different implementations. The median is a statistical metric that is not affected by outliers. Therefore, MedianAPE is not affected by each and every single difference between the values of both methods. We can see that we get more reasonable results.

In contrast, MAPE is affected by outliers because we are doing an average of our differences. With only one component we good a pretty good result, but we start getting high values when we increase the dimension.

The R^2 coefficient is the proportion of the variance in one variable that is predictable from the other one. Obviously, the more similar are two distributions, the higher R^2 will be. We can see that we get very high values in almost all dimensions, concluding that both packages are very similar and they approach the problem equally.

We can also prove it when plotting our results. In figure 9 we can see that both resulting distribution with only one component to estimate are almost equal.

Nevertheless, in figure 10 we can clearly see more differences when increasing the dimension to 3. Weirdly, it's like both plots are mirrored. The x-axis is reversed, but if we flip upside down one of the plots we could appreciate better the similarity.

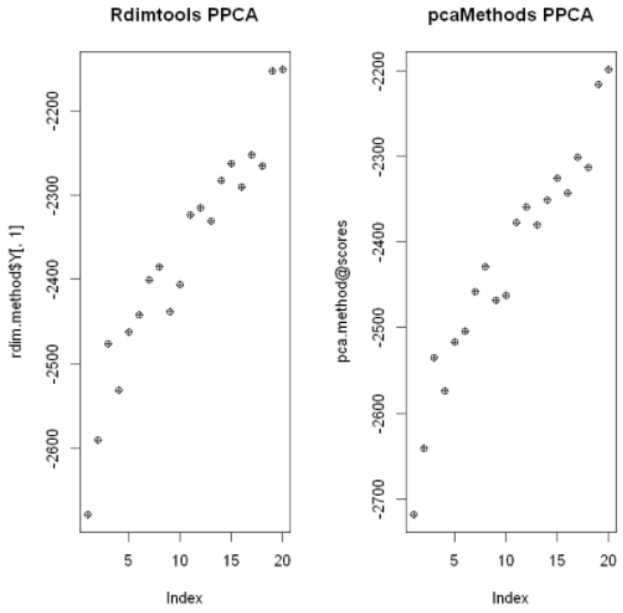


Figure 9: Resulting plot of PPCA for both methods in dimension 1

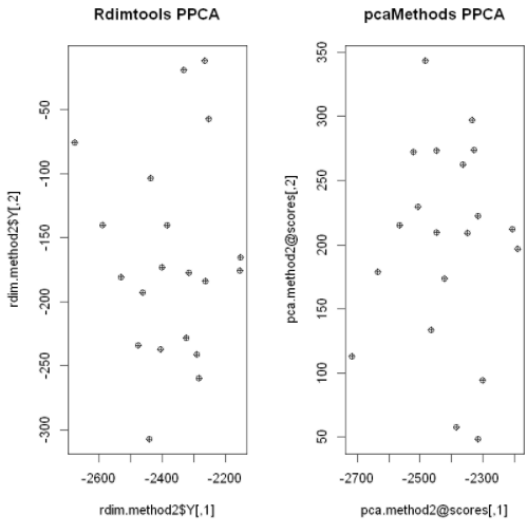


Figure 10: Resulting plot of PPCA for both methods in dimension 3

Bayesian approach

Here are our overall results on the dataset `Decathlon_Austria82` when performing the bayesian PCA:

Number of components to estimate	**RMSE**	**MedianAPE**	**MAPE**	R^2 **Regression Score**
1	2427,02	1	1	-345,23
3	230,83	0,91	0,12	0,96
5	272,01	2,03	1,37	0,91

When performing the Bayesian PCA we start noticing more "crazy" numbers. Is it clear that both packages work differently in this function because we cannot see any similarity between both scores. Only in dimension 3 we can see a little bit of alikeness, but we get a crazy MedianAPE.

What is even more crazy is the R^2 score when we are in dimension 1. We don't know if is it because the Rdimtools package cannot perform the bayesian PCA in dimension 1 without "cheating" (we compute it in dimension 2 and then took only the first part of the matrix). In Figure 11 we can visually confirm this dissimilarity.

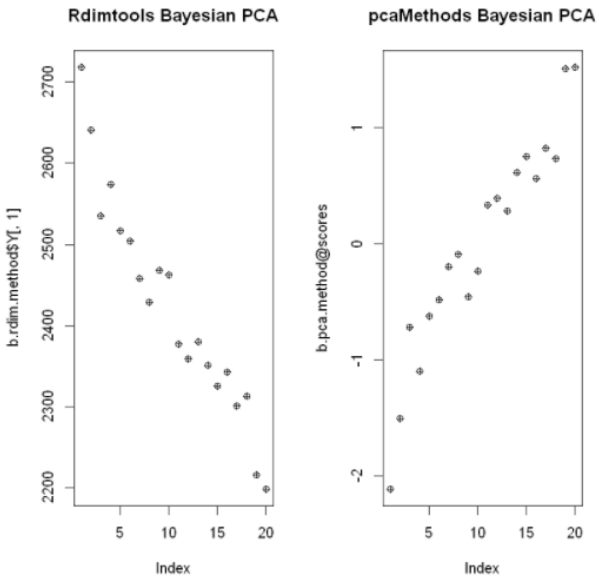


Figure 11: Resulting plot of Bayesian PCA for both methods in dimension 1

3.1.6 Hinton Diagrams

Hinton Diagrams are useful for representing PCA results. A Hinton diagram allows us to visualize the values of our loading matrix with positive (white) and negative (black) squares. The bigger the square, the more contribution to the PCA. Obviously, the less number of squares the more sparser is our "model".

In figure 12 we visualize the virtues of the Bayesian PCA. As expected, this approach gives the sparsest Hinton diagram when comparing with the regular PCA implementation and the probabilistic approach of both packages. All the available Hinton diagrams can be found in the `Hinton Diagrams` notebook.

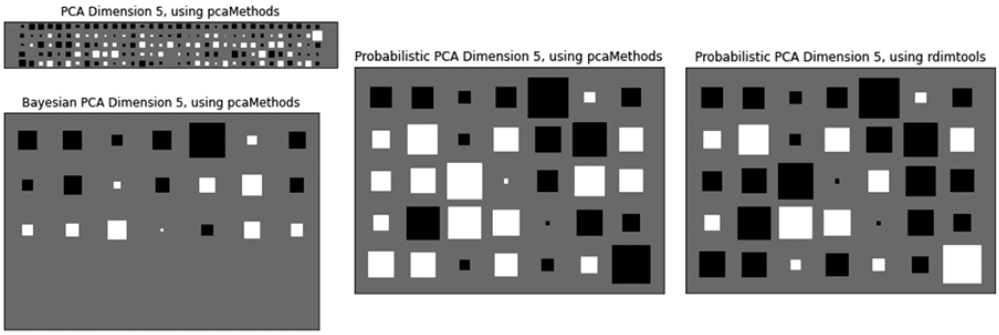


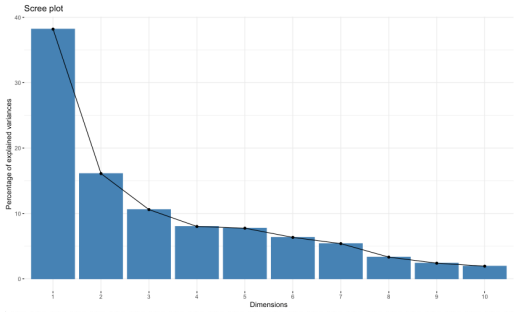
Figure 12: Hinton Diagrams in dimension 5 comparison

3.2 BPCA of decathlon performance of all-time top UK athletes

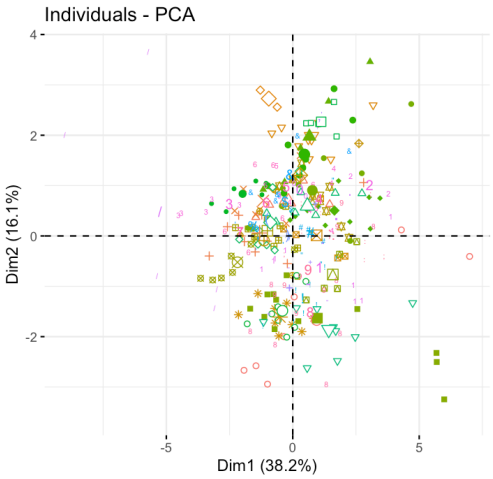
Using the implementations that we have found to be more suitable, we will explore the decathlon performance of all-time UK athltes. We will use this dataset because its the one with more data, it has temporal data on athletes, and it has more events than the heptathlon dataset, and so, we think that the latent variables may be more insightful.

First, let's see what we obtain when evaluating the dataset using conventional PCA (Figure 13. First, looking at the scree plot, we can observe that there is a first component that captures most of the variance, a second one with also quite a lot of variance and the following ones capture a moderate amount of variance, but it is not clear how many components should be used. We can observe that the first component captures general performance in the running and jumping events. The performance in the throwing events, is captured by the second components. This suggests that performance in running and jumping is closely related (being the performance in the 1,500m the one that correlates the less) and that performance in the throwing events is controlled by a different set of abilities which are not contradictory with the previous —one can improve both set of abilities at the same time—. If we go to less variant components, we start to see some set of abilities that are indeed antagonic: just as examples, we see that increasing pole vault performance decreases 400m performance and increasing jumping performance decreases throwing performance.

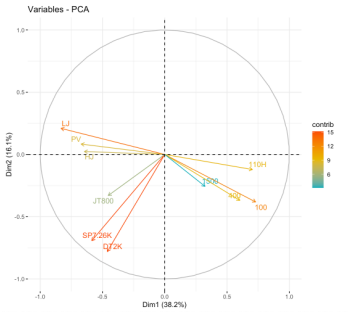
Using these latent variables, we can study athletes progression over the course of years (Figure 14). We observe in the first case that the athlete greatly increased its overall jumping and running performance in 2018 but that this was coupled with a decrease in the throwing performance. Athlete two, conversely, decreased its performance after 2007 in all the different abilities, and started to pick up its performance by 2018, again, for all the abilities at the same time.



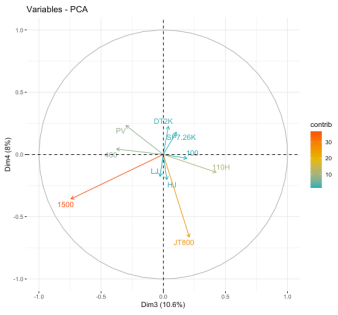
(a) Scree plot.



(b) Individual representation using PC1 and PC2.

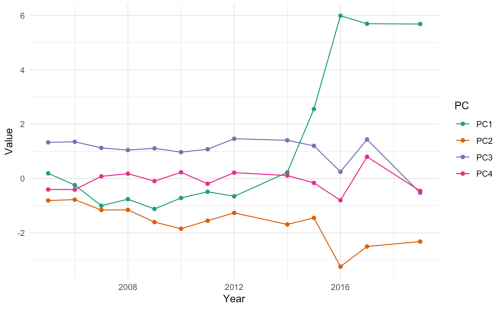


(c) PC1 and PC2 variable importance.

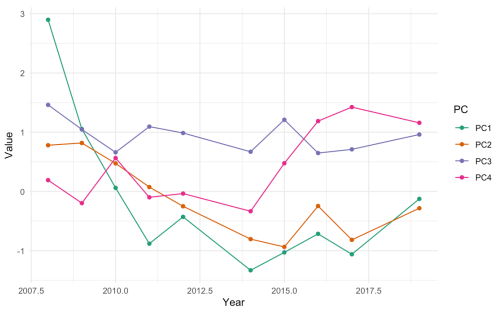


(d) PC3 and PC4 variable importance.

Figure 13: Conventional PCA. The scree plot, the individuals representations using the two first components, and the variable loadings for the first four components are shown.



(a) Athlete 1.



(b) Athelte 2.

Figure 14: Progression study using PCA. The evolution of the PCs of two individual athletes over the course of their career is shown. We have chosen those athletes with the longest careers.

These findings are similar to the ones shown in Figure 4, referring to the stan implementation of Bayesian PCA, in which we also saw to separate components: one for running and jumping, a

second one for throwing. However, that analysis revealed that, in fact, the first two components are enough to represent the dataset. Therefore, using the Bayesian analysis, we can dismiss our previous observation, in which we said that there were some less important components representing some abilities that are antagonic and make it difficult to improve in all the events at the same time. Using Bayesian methods, we see that those components were in fact not important. These shows the usefulness of applying Bayesian methods to study this kind of problems.

Because we were not absolutely confident in our Stan implementation of Bayesian PCA, we also performed this analysis using the `Rdimtools` package, which implements bayesian PCA and its solution as described in the original publication [Bishop, 1999]. However, as shown in the previous section, this implementation leads to really confusing results. We observe (Figure 15) again that two components are selected as the effective dimensionality of the latent space. However, these two components are perfectly correlated, because BPCA does not ensure that the components are orthogonal. It seems weird that, when trying to find the effective dimensionality of the dataset, two components representing the exact same information are kept. Therefore, we can not draw conclusions from this analysis.

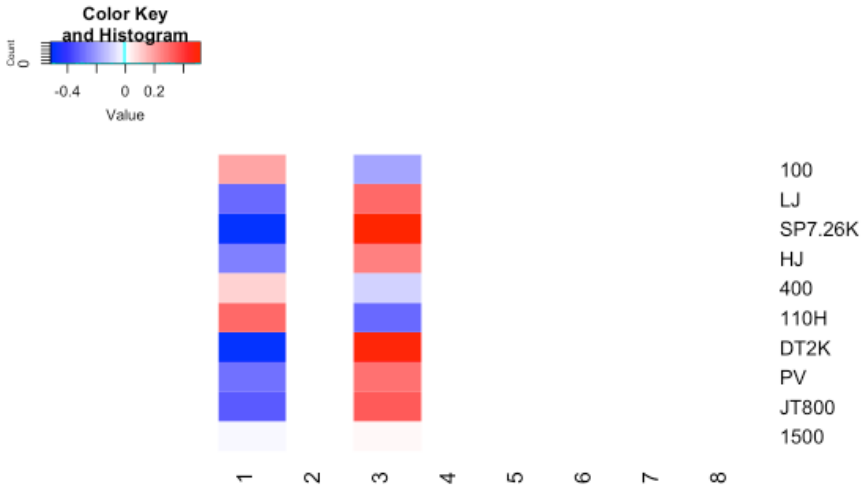


Figure 15: Loadings matrix (W) heatmap. Heatmap representing the loadings matrix W obtained by training the ARD BPCA model using Rdimtools. Red represents positive weights, blue represents negative weights.

4 Conclusions

All in all, we have seen that PCA is a revealing way to study combined events performance and extract conclusions on the latent factors that drive performance in each event, and how these are interconnected between events. We tried to approach this BPCA methodology using all the possible tools available for us: through CRAN packages and using our own implementation, in Python and Stan. We always used graphic resources in order to explain our work.

BPCA is specially useful for this analysis because it allows to see how many latent variables are really driving this performance. However, at least in our experience, we have observed that BPCA

is quite hard to implement, may take a long time to run if fitted using sampling, and can lead to different and confusing results.

Therefore, we are not yet convinced of the suitability of this method for our problem. Perhaps, if we had also missing data and wanted to generate samples or assess the certainty of our predictions, this method would be more useful.

References

[Bishop, 1999] Bishop, C. M. (1999). Bayesian pca. *Advances in neural information processing systems*, pages 382–388.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern recognition and Machine Learning*. Springer Science, New York.

[Fanshawe, 2012] Fanshawe, T. (2012). Seven into two: Principal components analysis and the olympic heptathlon. *Significance*, 9(2):40–42.

[Jolliffe, 2005] Jolliffe, I. (2005). Principal component analysis. *Encyclopedia of statistics in behavioral science*.

[Oba et al., 2003] Oba, S., Sato, M.-a., Takemasa, I., Monden, M., Matsubara, K.-i., and Ishii, S. (2003). A bayesian missing value estimation method for gene expression profile data. *Bioinformatics*, 19(16):2088–2096.

[Tipping and Bishop, 1999] Tipping, M. E. and Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622.

[Van Damme et al., 2002] Van Damme, R., Wilson, R. S., Vanhooydonck, B., and Aerts, P. (2002). Performance constraints in decathletes. *Nature*, 415(6873):755–756.