

Finding the Number of Principal Components

Min Wang¹, Steven M. Kornblau², and Kevin R. Coombes³

¹Mathematical Biosciences Institute, The Ohio State University

²Dept. of Leukemia, The University of Texas MD Anderson Cancer Center

³Dept. of Biomedical Informatics, The Ohio State University

September 25, 2026

Contents

1	Getting Started	2
2	Testing on Unstructured Data	2
2.1	Bartlett's Test	2
2.2	Randomization-Based Methods	4
2.3	Broken-Stick	4
2.4	Auer-Gervini	5
3	Testing on Data with Structure	8
3.1	Bartlett's Test	8
3.2	Randomization-Based Methods	9
3.3	Broken-Stick	10
3.4	Auer-Gervini	10
4	References	14

1 Getting Started

In this section, we describe how to select significant number of PCs using the **PCDimension** R package. The latest version of the package is always available from the R-Forge webpage (http://r-forge.r-project.org/R/?group_id=1900); the latest stable version can be found on CRAN. We illustrate the methods by exploring a small simulated data set.

First, we load all of the R library packages that we need for this analysis. Note that **PCDimension** implements the Broken-Stick model, the randomization-based procedure of ter Braak [1,2], and the Auer-Gervini model [3], while **nFactors** (developed by Raiche and Magis) is used to run Bartlett's test and its variants.

```
> library(PCDimension)
> library(nFactors) # implements Bartlett's test
> library(MASS) # for mvrnorm to simulate data
```

2 Testing on Unstructured Data

Next, we simulate an unstructured data set with random noise. That is, the variation of the data is isotropic and the number of significant PCs is 0. The data is generated via the command:

```
> set.seed(12345)
> NC <- 15
> NS <- 200
> ranData <- matrix(rnorm(NS*NC, 6), ncol=NC)
```

2.1 Bartlett's Test

Now, we apply Bartlett's test to the simulated data. The required input includes the raw data and the number of subjects (rows).

```
> nBartlett(data.frame(ranData), nrow(ranData))
```

\$detail

	v	values	bartlett	bartlett.chi	bartlett.df	bartlett.p	anderson.chi
1	1	1.4646022	0.5811268	104.8482325	105	0.4858195	108.5572726
2	2	1.3301697	0.6364281	86.9876297	91	0.5995868	90.3767582
3	3	1.2587707	0.6773467	74.7328956	78	0.5838392	77.9144003

4	4	1.2303421	0.7138211	64.4466584	66	0.5311236	67.4245772
5	5	1.1537568	0.7543483	53.7021487	55	0.5243156	56.3802086
6	6	1.1204785	0.7881762	45.1867156	45	0.4641529	47.6067241
7	7	1.0644224	0.8245035	36.5042393	36	0.4452273	38.5947904
8	8	1.0357668	0.8581814	28.8291453	28	0.4212319	30.5879526
9	9	0.9636934	0.8968358	20.4517639	21	0.4928295	21.7765010
10	10	0.8806406	0.9291664	13.7506548	15	0.5445180	14.6934869
11	11	0.8216507	0.9510221	9.3656543	10	0.4977885	10.0435971
12	12	0.7727787	0.9679198	6.0592821	6	0.4165822	6.5212004
13	13	0.7216466	0.9826413	3.2424724	3	0.3557300	3.5022204
14	14	0.6272634	0.9961553	0.7107126	1	0.3992073	0.7704202
15	15	0.5540174	1.0000000	0.0000000	0	1.0000000	0.0000000

	anderson.df	anderson.p	lawley.chi	lawley.df	lawley.p
1	119	0.7435170	104.848232	105	0.4858195
2	104	0.8270435	87.059078	91	0.5974780
3	90	0.8144247	74.799593	78	0.5816965
4	77	0.7739155	64.506178	66	0.5290321
5	65	0.7682480	53.751380	55	0.5224186
6	54	0.7178408	45.229911	45	0.4623519
7	44	0.7018560	36.539032	36	0.4436253
8	35	0.6810483	28.857191	28	0.4197981
9	27	0.7486461	20.471358	21	0.4916057
10	20	0.7936695	13.764296	15	0.5434754
11	14	0.7589891	9.375495	10	0.4968766
12	9	0.6868314	6.065849	6	0.4158543
13	5	0.6230517	3.246031	3	0.3552250
14	2	0.6803077	0.711493	1	0.3989486
15	0	1.0000000	0.000000	0	1.0000000

\$nFactors

bartlett	anderson	lawley
15	15	0

attr(,"class")

[1] "nFactors" "list"

The original version of Bartlett's test, and the Anderson variant, fail to return the correct number of components. The Lawley variant does yield the correct value, 0.

2.2 Randomization-Based Methods

The **PCDimension** package implements both of the randomization-based statistics that were identified as successful in a previous study [4]. The number of permutations (default, $B = 1000$) and the significance level (default, $\alpha = 0.05$) are optional input arguments in addition to the required data set. The estimated number of PCs is the last point at which the p-value of the statistic of interest greater than the observed one is at least the threshold significance level.

```
> rndLambdaF(ranData) # input argument is data
```

```
rndLambda    rndF
      0         0
```

The randomization-based procedure successfully recovers the true number of PCs.

2.3 Broken-Stick

The **PCDimension** package also implements the Broken-Stick model. Both this model and the Auer-Gervini model require the eigenvalues from the singular value decomposition of the data matrix used to compute the principal components. We compute the decomposition using the **SamplePCA** function from the **ClassDiscovery** package, and then extract the variances.

```
> spca <- SamplePCA(t(ranData))
> lambda <- spca@variances[1:(NC-1)]
> bsDimension(lambda)
```

```
[1] 0
```

In the Broken-Stick model, the individual percentages of variance of the components are compared with the values expected from the “broken stick” distribution. The two distributions are compared element-by-element, and first value $d + 1$ where the expected value is larger than the observed value determines the dimension. The Broken-Stick model also correctly finds that there are zero significant PCs.

Note: In our implementation of the **bsDimension** function, we add an extra parameter (**FUZZ**, with default value 0.005) for this comparison to deal with numerical errors in the estimates of the eigenvalues.

2.4 Auer-Gervini

We now use the `SamplePCA` object to construct an Auer-Gervini object.

```
> ag.obj <- AuerGervini(spca)
> agDimension(ag.obj)
```

```
[1] 0
```

The `agDimension` function takes an optional argument, `agfun` that specifies the method used to automate the computation of the number of PCs. The default value uses the **TwiceMean** method, which correctly concludes that there are zero significant PCs. We can also compare the results of multiple algorithms to automate the procedure.

```
> f <- makeAgCpmFun("Exponential")
> agfuns <- list(twice=agDimTwiceMean, specc=agDimSpectral,
+               km=agDimKmeans, km3=agDimKmeans3,
+               tt=agDimTtest, tt2=agDimTtest2,
+               cpt=agDimCPT, cpm=f)
> compareAgDimMethods(ag.obj, agfuns) # compare the list of all criteria
```

twice	specc	km	km3	tt	tt2	cpt	cpm
0	0	0	0	1	0	0	0

Overall, the Auer-Gervini model does an excellent job in selecting the actual number of components since 7 criteria out of 8 return 0 while only the “Ttest” procedure yields 1. If the majority rule is applied, that is, the estimated number of PCs is the one selected in more than 4 criteria in Auer-Gervini model, then we will definitely have 0 as the estimated number of components.

To get a more comprehensive understanding of the Broken-Stick method and the Auer-Gervini model, we use the command to generate the plots of these two models in Figures 1 and 2:

```
> bs <- brokenStick(1:NC, NC)
> bs0 <- brokenStick(1:(NC-1), (NC-1))
```

Figure 1 shows the broken stick distributions and the relative proportions of the variation that are explained by all the components in the simulated data set. The blue dotted line represents the broken stick distributions under the condition that p equals the number

```

> pts <- screeplot(sPCA, ylim=c(0, 0.2))
> lines(pts, bs, type='b', col='blue', lwd=2, pch=16)
> lines(pts[-NC], bs0, type='b', col='red', lwd=2, pch=16)

```

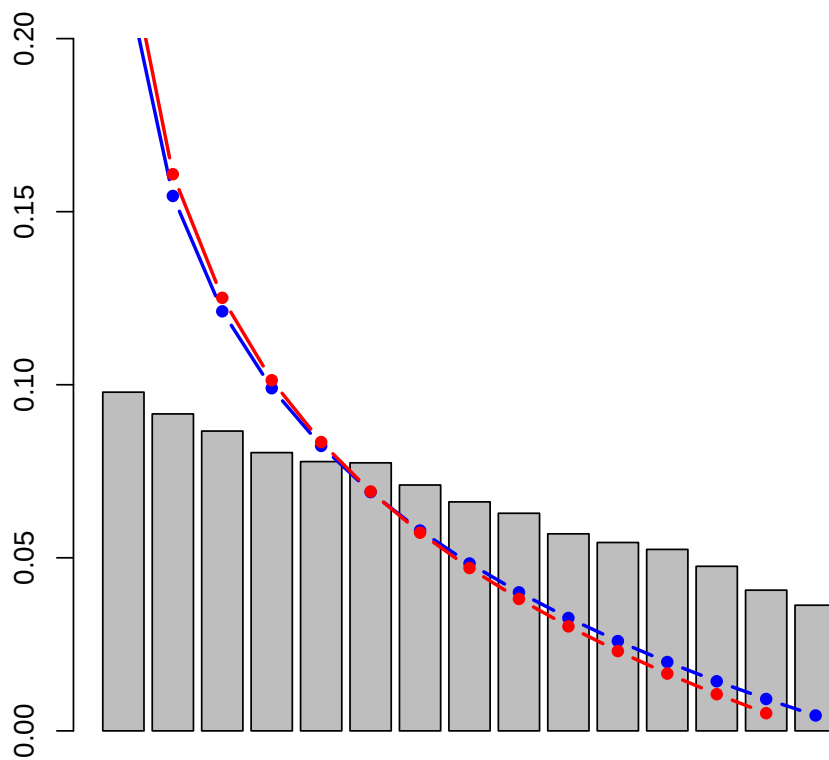


Figure 1: Screeplot of the data and the Broken-Stick model.

```
> plot(ag.obj, agfuns)
```

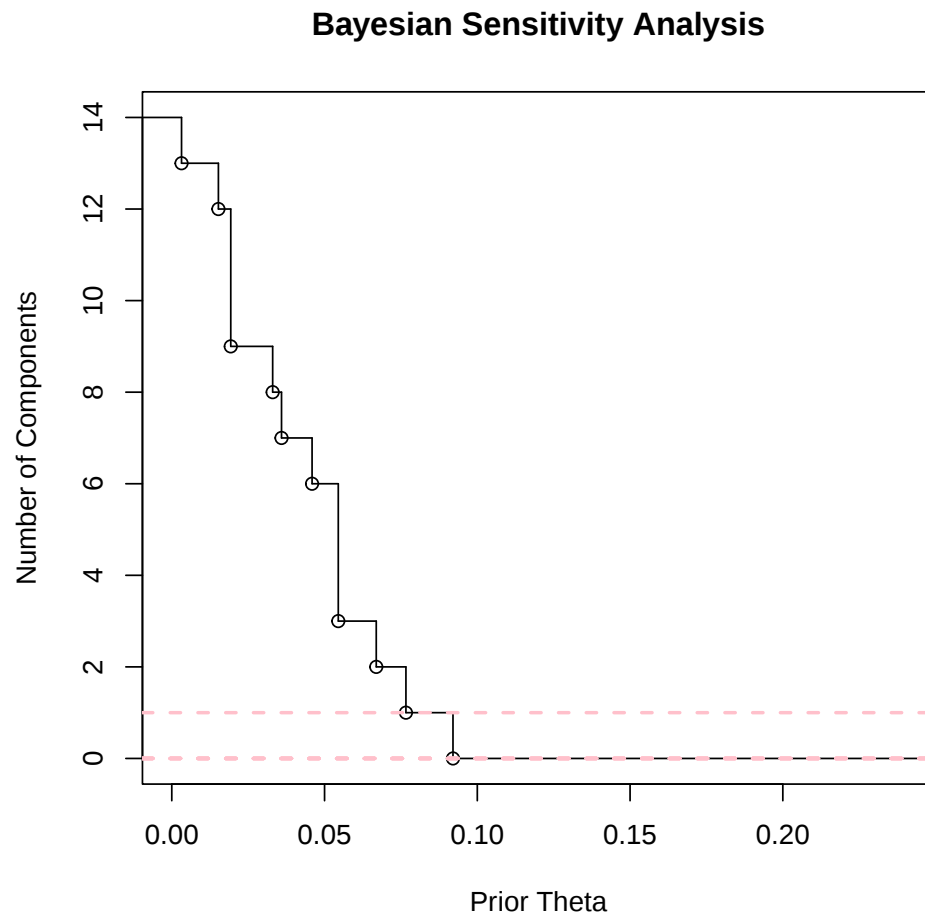


Figure 2: Auer-Gervini step function relating the prior hyperparameter θ to the maximum posterior estimate of the number K of significant principal components.

of features, while the red one means the broken stick distributions after removing the 0 eigenvalue with p equating with the number of features minus one. And there is almost no difference after removing the effect of eigenvalue 0. The grey rectangles are the relative proportions of the variation explained by all PCs. This figure provides a clear illustration on how the relative proportions are compared with the broken stick distributions and how the estimated number of PCs is chosen from the Broken-Stick model. Figure 2 illustrates how the Auer-Gervini model works. For the simulated data set, there are $NC = 15$ features, so the possible models $\mathcal{M}_d$ range from $\mathcal{M}_0$ to $\mathcal{M}_{14}$. The values $d = 0, 3, 4, 6, 7, 8, 11, 12, 13$ and 14 should be retained since the step functions are flat on those vertical coordinates. From the plot, we can see that the highest dimension d for which the step is significantly large is at $d = 0$. So $\mathcal{M}_0$ is a reasonable model.

3 Testing on Data with Structure

Now we simulate another dataset, with two groups of correlated samples.

```
> mu <- rep(0, 15)
> sigma <- matrix(0, 15, 15)
> sigma[1:8, 1:8] <- 0.7
> sigma[9:15, 9:15] <- 0.3
> diag(sigma) <- 1
> struct <- mvrnorm(200, mu, sigma)
```

3.1 Bartlett's Test

As before, we start by applying Bartlett's test to the simulated data.

```
> nBartlett(data.frame(struct), nrow(struct))
```

```
$detail
      v  values      bartlett bartlett.chi bartlett.df      bartlett.p
1    1 6.0810120 0.0001832254 1662.1592934        105 2.561146e-278
2    2 2.6210352 0.0166105559  788.8105014        104 9.426647e-106
3    3 1.0094725 0.1419738398  374.4802412         90 2.285714e-36
4    4 0.7866754 0.2121517991  296.3949752         77 1.751407e-27
5    5 0.7411964 0.2683570789  250.5907107         65 1.343425e-23
6    6 0.7038556 0.3452489339  201.8857709         54 6.149201e-19
7    7 0.6553115 0.4613915368  146.3219822         44 6.484416e-13
```

8	8	0.5446527	0.6422826664	83.4539984	35	7.780985e-06
9	9	0.3880309	0.8414173931	32.4327001	27	2.164379e-01
10	10	0.2990730	0.9311136612	13.3588195	20	8.614634e-01
11	11	0.2860686	0.9561475369	8.3632289	14	8.695576e-01
12	12	0.2473571	0.9834091489	3.1089955	9	9.598163e-01
13	13	0.2319970	0.9923990427	1.4128201	5	9.229058e-01
14	14	0.2091488	0.9987945770	0.2225347	2	8.946995e-01
15	15	0.1951131	1.0000000000	0.0000000	0	1.000000e+00
		anderson.chi	anderson.df	anderson.p	lawley.chi	lawley.df
1		1720.958716	119	5.698231e-282	1662.1592934	105
2		819.543378	104	1.396252e-111	788.8473505	91
3		390.422493	90	4.881417e-39	374.5277480	78
4		310.090645	77	9.971799e-30	296.5381658	66
5		263.087360	65	1.184840e-25	250.7447552	55
6		212.697915	54	1.053760e-20	202.0062587	45
7		154.701655	44	3.098939e-14	146.4039267	36
8		88.545356	35	1.568219e-06	83.4976948	28
9		34.533487	27	1.510814e-01	32.4512213	21
10		14.274785	20	8.162948e-01	13.3694995	15
11		8.968610	14	8.330561e-01	8.3718671	10
12		3.346004	9	9.489850e-01	3.1121719	6
13		1.525998	5	9.100503e-01	1.4144460	3
14		0.241230	2	8.863751e-01	0.2227962	1
15		0.000000	0	1.000000e+00	0.0000000	0
					lawley.p	
					2.561146e-278	
					1.062088e-111	
					2.540012e-40	
					5.828867e-31	
					8.639871e-27	
					8.985241e-22	
					2.931901e-15	
					1.988980e-07	
					5.266764e-02	
					5.737813e-01	
					5.925628e-01	
					7.946422e-01	
					7.021520e-01	
					6.369176e-01	
					1.000000e+00	

```
$nFactors
```

```
bartlett anderson lawley
      8      8      8
```

```
attr("class")
```

```
[1] "nFactors" "list"
```

All three variants grossly overestimate the dimension.

3.2 Randomization-Based Methods

Next, we apply ter Braak's randomization procedures. Here we again use the default values of the parameters, but include them explicitly to illustrate the function.

```
> rndLambdaF(struct, B = 1000, alpha = 0.05) # input argument is data
```

<code>rndLambda</code>	<code>rndF</code>
2	9

The randomization-based procedure `rndLambda` successfully recovers the true number of PCs, but the `rndF` procedure overestimates it. These results are consistent with a larger set of simulations that we have performed..

3.3 Broken-Stick

Next, we apply the broken-stick model. As before, we first compute the singular value decomposition using the `SamplePCA` function from the **ClassDiscovery** package.

```
> spca <- SamplePCA(t(struct))
> lambda <- spca@variances[1:(NC-1)]
> bsDimension(lambda)
```

```
[1] 2
```

The scree-plot (Figure 3) explains how the broken-stick model gets the correct answer.

```
> bs <- brokenStick(1:NC, NC)
> bs0 <- brokenStick(1:(NC-1), (NC-1))
```

3.4 Auer-Gervini

We now use the `SamplePCA` object to construct an Auer-Gervini object.

```
> ag.obj <- AuerGervini(spca)
> agDimension(ag.obj)
```

```
[1] 2
```

The `agDimension` function takes an optional argument, `agfun` that specifies the method used to automate the computation of the number of PCs. The default value uses the **TwiceMean** method, which correctly concludes that there are zero significant PCs. We can also compare the results of multiple algorithms to automate the procedure.

```

> pts <- screeplot(sPCA)
> lines(pts, bs, type='b', col='blue', lwd=2, pch=16)
> lines(pts[-NC], bs0, type='b', col='red', lwd=2, pch=16)

```

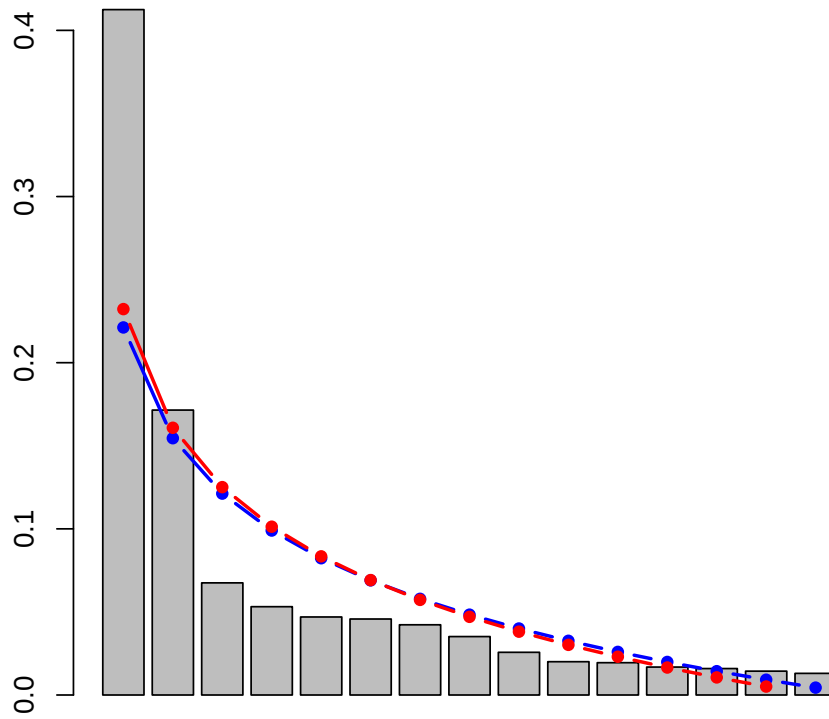


Figure 3: Screeplot of the structured data and the Broken-Stick model.

```

> f <- makeAgCpmFun("Exponential")
> agfuncs <- list(twice=agDimTwiceMean, specc=agDimSpectral,
+               km=agDimKmeans, km3=agDimKmeans3,
+               tt=agDimTtest, tt2=agDimTtest2,
+               cpt=agDimCPT, cpm=f)
> compareAgDimMethods(ag.obj, agfuncs) # compare the list of all criteria

```

twice	specc	km	km3	tt	tt2	cpt	cpm
2	2	2	2	3	2	1	2

Again, most of the criteria used to automate the Auer-Gervni method get te right answer. Two of them (Ttest and CPM), however, grossly overestimate it. Figure 4 shows the Auer-Gervini plot.

```
> plot(ag.obj, agfuns)
```

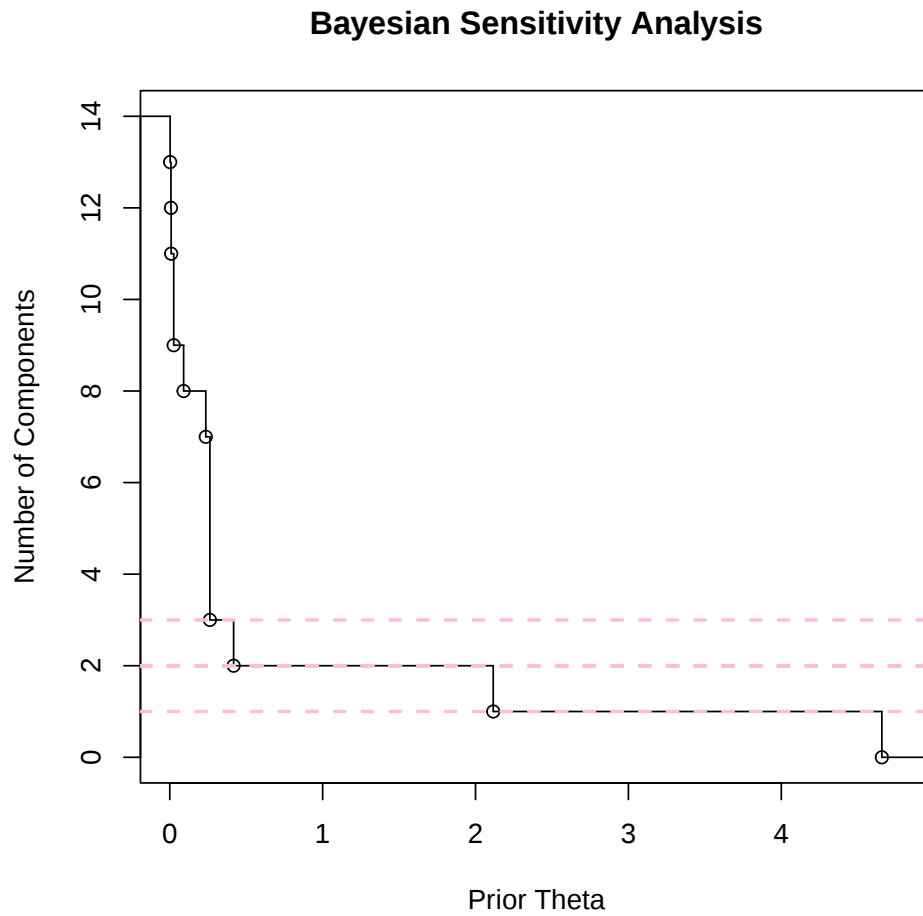


Figure 4: Auer-Gervini step function relating the prior hyperparameter θ to the maximum posterior estimate of the number K of significant principal components in the example with structured data.

4 References

- [1] ter Braak CFJ. *CANOCO – a Fortran program for canonical community ordination by [partial] [detrended] [canonical] correspondence analysis, principal component analysis and redundancy analysis (version 2.1)*. Agricultural Mathematics Group, Report LWA-88-02, Wageningen, 1988.
- [2] ter Braak CFJ. *Update notes: CANOCO (version 3.1)*. Agricultural Mathematics Group, Wageningen, 1990.
- [3] Auer P and Gervini D. Choosing principal components: A new graphical method based on Bayesian model selection. *Communications in Statistics - Simulation and Computation* 2008; 37: 962–977.
- [4] Peres-Neto PR, Jackson DA and Somers KM. How many principal components? Stopping rules for determining the number of non-trivial axes revisited. *Computational Statistics and Data Analysis* 2005; 49: 974–997.