**Computational Statistics manuscript No.**
(will be inserted by the editor)

# The Spectral Condition Number Plot for Regularization Parameter Evaluation: Supplementary Material

**Carel F.W. Peeters** ·
**Mark A. van de Wiel** ·
**Wessel N. van Wieringen**

Received: date / Accepted: date

This supplement contains figures and R code in support of the main text. In addition, this supplement contains a second illustration of (the use of) the spectral condition number plot.

## 1. Behavior largest and smallest eigenvalues

This section contain a visual impression (Figure S1) of the behavior of the largest and smallest eigenvalues along the domain of the penalty parameter. The setting is given in Example 1 of the main text.

This research was supported by grant FP7-269553 (EpiRadBio) through the European Community's Seventh Framework Programme (FP7, 2007-2013).

C.F.W. Peeters (Corresponding author)
Dept. of Epidemiology & Biostatistics, Amsterdam Public Health research institute, Amsterdam University Medical Centers, Location VUmc, Amsterdam, The Netherlands
E-mail: cf.peeters@vumc.nl

M.A. van de Wiel
Dept. of Epidemiology & Biostatistics, Amsterdam Public Health research institute, Amsterdam University Medical Centers, Location VUmc, Amsterdam, The Netherlands; and MRC Biostatistics Unit, University of Cambridge, Cambridge, United Kingdom

W.N. van Wieringen
Dept. of Epidemiology & Biostatistics, Amsterdam Public Health research institute, Amsterdam University Medical Centers, Location VUmc, Amsterdam, The Netherlands; and Dept. of Mathematics, VU University Amsterdam, Amsterdam, The Netherlands

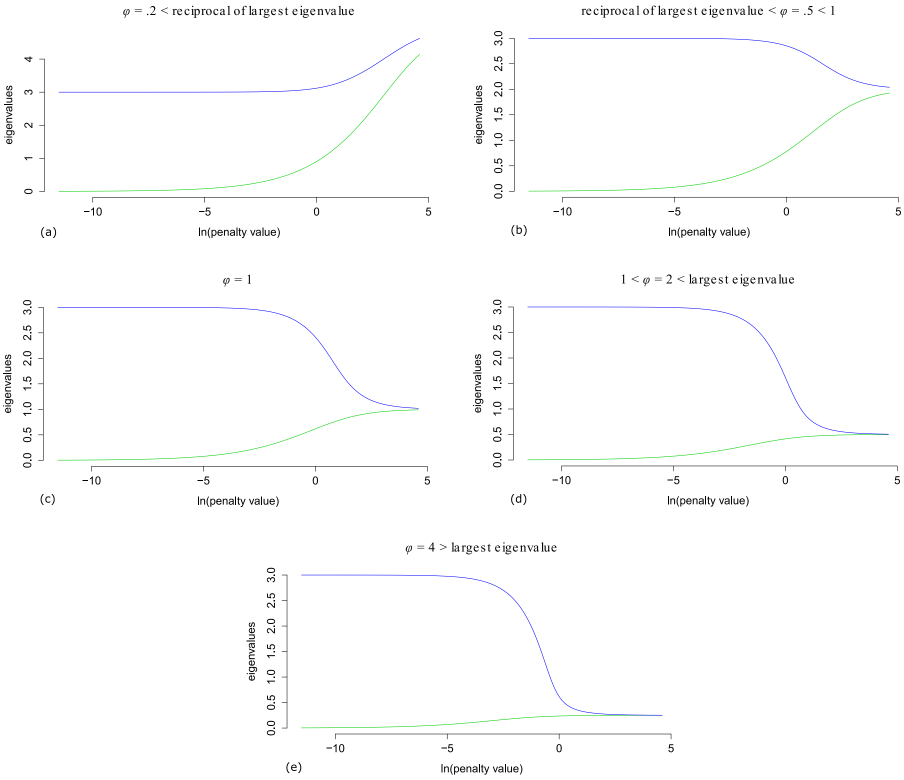


**Fig. S1** Visualization of the behavior of the largest and smallest eigenvalues along the domain of the penalty parameter. The green line represents the smallest eigenvalue while the blue line represents the largest eigenvalue. The panels represent different choices for the scalar $\varphi$: (a) $\varphi < 1/d(\hat{\mathbf{\Sigma}})_1$; (b) $1/d(\hat{\mathbf{\Sigma}})_1 < \varphi < 1$; (c) $\varphi = 1$; (d) $1 < \varphi < d(\hat{\mathbf{\Sigma}})_1$; and (e) $\varphi > d(\hat{\mathbf{\Sigma}})_1$.

## 2. A second Illustration

### 2.1. Context and data

Prostate cancer refers to an adenocarcinoma in the prostate gland. It is the most common solid tumor diagnosed in western men [19]. Its prognosis is largely determined by metastasis with low survival rates for metastatic forms in comparison to organ-confined forms [1].

Vascular endothelial growth factor (VEGF) is a signal protein that supports vascularization [7]. Vascular endothelial cells are cells that line the interior of blood vessels. The formation of new blood vessels is pivotal to the growth and metastasis of solid tumors. VEGF is actually at the helm of a cascade of signaling pathways that form the VEGF-signaling pathway. In specific, the activation of VEGF leads to the activation of the mitogen-activated protein kinase (MAPK) and phosphatidylinositol 3'-kinase (PI3K)-AKT signaling pathways [13]. The VEGF-signaling pathway thus largely consists of two subpathways. These subpathways mediate “the proliferation and migration of endothelial cells and” promote “their survival and vascular permeability” [13]. Hence, VEGF overexpression supports tumor growth and metastasis.

VEGF-signaling is likely active in metastatic prostate cancer. This contention is supported by recent evidence that changes in the PI3K-pathway are present in metastatic tumor samples [20]. Cancer may be viewed, from a pathway perspective, as consisting of a loss of normal biochemical connections and a gain of abnormal biochemical connections. Severe deregulation would then imply the loss of normal subpathways and/or the gain of irregular subpathways. Our goal here is to explore if the VEGF-signaling pathway in metastatic prostate cancer can still be characterized as consisting of the MAPK and PI3K-AKT subpathways. The exploration will make use of a pathway-based factor analysis in which the retained latent factors are taken to represent biochemical subpwathways [cf. 3]. This exercise hinges upon a well-conditioned covariance matrix.

We attained data on prostate cancer from the Memorial Sloan-Kettering Cancer Center [20] as queried through the Cancer Genomics Data Server [4, 8] using the `cgdsr` R-package [9]. All metastatic samples were retrieved for which messenger ribonucleic acid (mRNA) data is available, giving a total of $n = 19$ samples. The data stem from the Affymetrix Human Exon 1.0 ST array platform and consist of $\log_2$ whole-transcript mRNA expression values. All Human Genome Organization (HUGO) Gene Nomenclature Committee (HGNC) curated features were retained that map to the VEGF-signaling pathway according to the Kyoto Encyclopedia of Genes and Genomes (KEGG) [12], giving a total of $p = 75$ gene features. Regularization of the desired covariance matrix is needed as $p > n$. Regularization is performed (as in the illustration in the main text) on the standardized scale.

### 2.2. Model

We work with standardized data. Hence, let $\mathbf{z}_i \in \mathbb{R}^p$ denote a centered and scaled $p$-dimensional observation vector available for $i = 1, \ldots, n$ persons. The factor model states that

$$\mathbf{z}_i = \boldsymbol{\Gamma}\boldsymbol{\xi}_i + \boldsymbol{\epsilon}_i,$$

where $\boldsymbol{\xi}_i \in \mathbb{R}^m$ denotes an $m$-dimensional vector of latent variables typically called ‘factors’, and where $\boldsymbol{\Gamma} \in \mathbb{R}^{p \times m}$ is a matrix whose entries $\gamma_{jk}$ denote the loading of the $j$th variable on the $k$th factor, $j = 1, \ldots, p$, $k = 1, \ldots, m$. Finally, the $\boldsymbol{\epsilon}_i \in \mathbb{R}^p$ denote error measurements. Hence, the model can be conceived of as a multivariate regression with latent predictors. An important assumption in this model is that $m < p$: the dimension of the latent vector is smaller than the dimension of the observation vector. The following additional assumptions are made: (i) The observation vectors are independent; (ii) $\text{rank}(\boldsymbol{\Gamma}) = m$; (iii) $\boldsymbol{\xi}_i \sim \mathcal{N}_m(\mathbf{0}, \mathbf{I}_m)$; (iv) $\boldsymbol{\epsilon}_i \sim \mathcal{N}_p(\mathbf{0}, \boldsymbol{\Psi})$, with $\boldsymbol{\Psi}$ a $(p \times p)$-dimensional diagonal matrix with strictly positive diagonal entries $\psi_j$; and (v) $\boldsymbol{\xi}_i$ and $\boldsymbol{\epsilon}_{i'}$ are independent for all $i$ and $i'$. The preceding assumptions establish a distributional assumption on the covariance structure of the observed data-vector: (vi) $\mathbf{z}_i \sim \mathcal{N}_p(\mathbf{0}, \mathbf{\Sigma} = \mathbf{\Gamma}\mathbf{\Gamma}^{\mathrm{T}} + \mathbf{\Psi})$.

Hence, to characterize the model, we need to estimate $\mathbf{\Gamma}$ and $\mathbf{\Psi}$ on the basis of the sample correlation matrix. As $p > n$ for the data at hand, the sample correlation matrix is singular and standard maximum likelihood estimation (MLE) is not available. Recent efforts deal with such situations by (Bayesian) sparsity modeling of the factor model [e.g., 3], i.e., by imposing sparsity constraints in the loadings matrix. Our approach differs. We first ensure that we have a well-conditioned correlation matrix by using a regularized (essentially a Bayesian) estimator. Afterwards, standard MLE techniques are employed to estimate $\mathbf{\Gamma}$ and $\mathbf{\Psi}$ on the basis of this regularized correlation matrix. Hence, we perform MLE on the basis of $\hat{\mathbf{\Sigma}}(\lambda^*)$ where $\lambda^*$ is (in some sense) deemed optimal.

### 2.3. Penalty parameter selection

The estimator of choice is again (3) of the main text where we replace $\hat{\mathbf{\Sigma}}$ with the sample correlation matrix $\mathbf{R}$. The target matrix is chosen as $\mathbf{T} = \mathbf{I}_p$, such that a regularized correlation matrix ensues. Again, the aLOOCV procedure was tried first in finding an optimal value for $\lambda_a$ under the given target and data settings. We searched for the optimal value $\lambda_a^*$ in the domain $\lambda_a \in [1 \times 10^{-5}, 20]$ with $10,000$ log-equidistant steps along this domain. Again, the procedure pointed to $1 \times 10^{-5}$ as being the optimal value for the penalty (in the chosen domain), which seems low given the $p/n$ ratio of the data. The condition number plot (covering the same penalty-domain considered by the aLOOCV procedure) indeed indicates that the precision estimate at $\lambda_a = 1 \times 10^{-5}$ is not well-conditioned in the sense of the Heuristic Definition, exhibiting a condition number of approximately $9,456$ (Figure S2). A reasonable minimal penalty-value (in accordance with the Heuristic Definition) can be found at approximately $\exp(-6.5)$, at which $\mathcal{C}_2[\hat{\mathbf{\Sigma}}^a(\exp(-6.5))] \approx 785.01$. This reasonable minimal value is subsequently used to constrain the search-domain to the region of well-conditionedness. A root-finding (by the Brent algorithm [2]) LOOCV procedure is then told to search for the optimal value $\lambda_a^*$ in the domain $\lambda_a \in [\exp(-6.5), 20]$. The optimal penalty-value is found at .422. At this value, indicated by the red vertical line in Figure S2, $\mathcal{C}_2[\hat{\mathbf{\Sigma}}^a(.422)] \approx 62.39$.

The Kaiser-Meyer-Olkin index [11] of .98 indicates that a reasonable proportion of variance among the variables might be common variance and, hence, that $\hat{\mathbf{\Sigma}}^a(.422)$ is suitable for a factor analysis. The regularized estimate $\hat{\mathbf{\Sigma}}^a(.422)$ is used in further analysis.

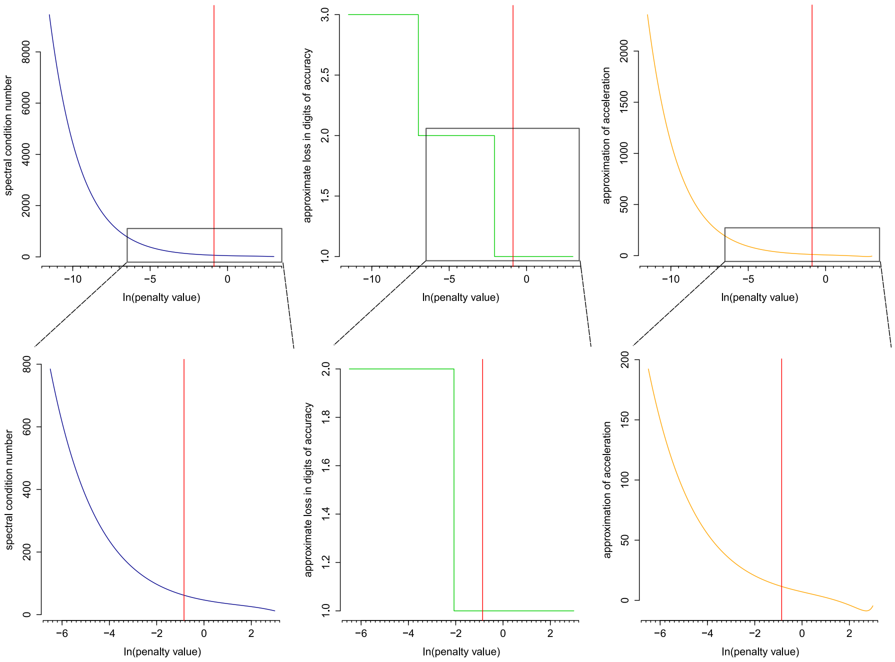


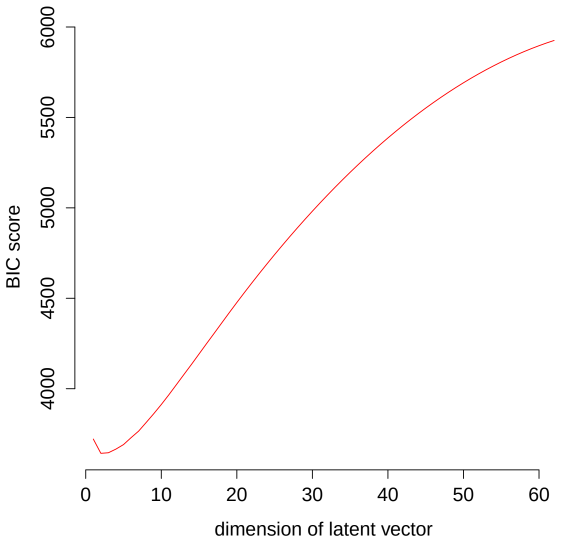
**Fig. S2** The left-hand panels give the basic spectral condition number plot. The middle and right-hand panels exemplify the interpretational aids to the basic plot: the approximate loss in digits of accuracy (middle panel) and the approximation of the acceleration along the curve in the basic plot (right-hand panel). The top panels give the basic condition number plot and its interpretational aids for the domain $\lambda_a \in [1 \times 10^{-5}, 20]$. The bottom panels zoom in on the boxed areas. The boxed areas cover a domain of well-conditionedness according to the heuristically chosen minimal penalty-value: $\lambda_a \in [\exp(-6.5), 20]$. The red vertical line indicates the value of the penalty that was chosen as optimal by the root-finding LOOCV procedure (.422).

## 2.4. Further analysis

The dimension of the factor solution is unknown. Hence, the optimal factor dimension needs to be determined in conjunction with the estimation of the model. Now, let $\hat{\mathbf{\Sigma}}_m = \hat{\mathbf{\Lambda}}_m \hat{\mathbf{\Lambda}}_m^{\mathrm{T}} + \hat{\mathbf{\Psi}}$ denote the MLE solution under $m$ factors. Then, we determine the optimal dimension (contingent upon the MLE solutions) using the Bayesian Information Criterion (BIC; [18]), which, for the problem at hand, amounts to:

$$n \left\{ p \ln(2\pi) + \ln |\hat{\mathbf{\Sigma}}_m| + \mathrm{tr}\left( \hat{\mathbf{\Sigma}}_m^{-1} \hat{\mathbf{\Sigma}}^a(.422) \right) \right\} + \ln(n)\eta,$$

where $\eta = p(m+1) - m(m-1)/2$ indicates the free parameters in the model. Figure S3 gives the BIC scores for each dimension allowed by the existence condition $p(p+1)/2 - \eta \geq 0$. The solution with the lowest BIC score is deemed optimal, indicating that $m = 2$ is the preferred solution.



| $m$ | BIC |
|---|---|
| 1 | $3,721.762$ |
| 2 | $3,642.678$ |
| 3 | $3,645.902$ |
| 4 | $3,666.222$ |
| 5 | $3,691.069$ |

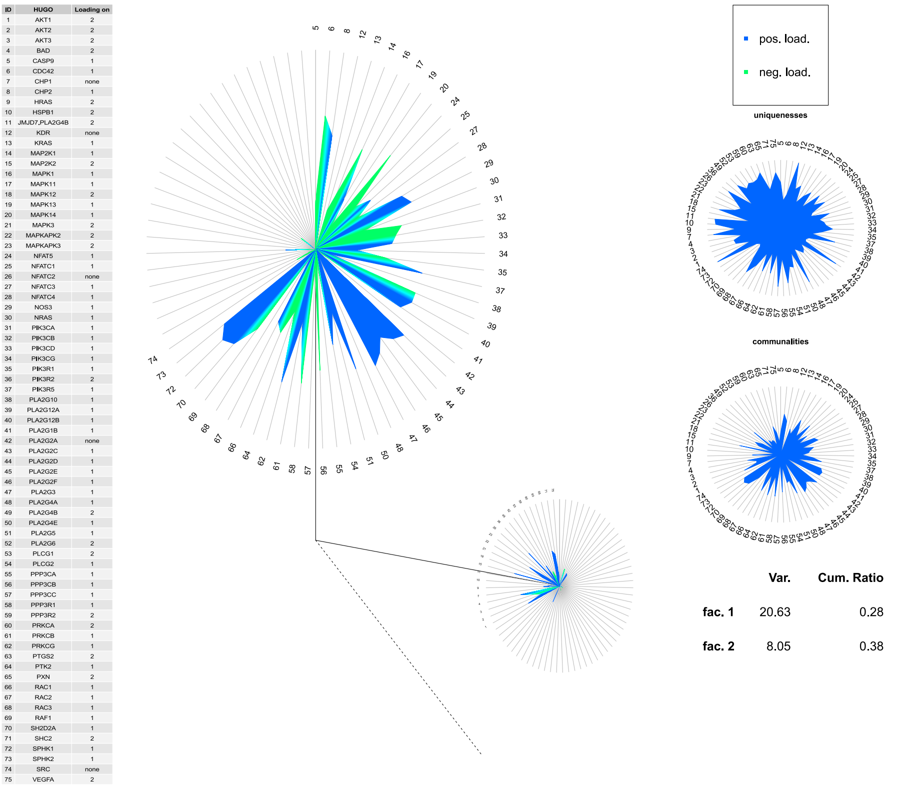
**Fig. S3** BIC scores for various dimensions of the latent vector. The left-hand panel gives the trace of the BIC scores for all dimensions of the latent vector allowed by the bound $p(p+1)/2 - \eta \geq 0$. The right-hand panel gives a table with BIC scores for $m = 1, \ldots, 5$. The $m = 2$ solution is to be preferred according to the BIC.

The specified FA model is inherently underidentified. Assume $\mathbf{H} \in \mathbb{R}^{m \times m}$ is an arbitrary orthogonal matrix. Considering the implied covariance structure of the observed data we may write:

$$\mathbf{\Gamma}\mathbf{\Gamma}^{\mathrm{T}} + \mathbf{\Psi} = (\mathbf{\Gamma}\mathbf{H})(\mathbf{\Gamma}\mathbf{H})^{\mathrm{T}} + \mathbf{\Psi}.$$

This equality implies that, given $\mathbf{\Psi}$, there is an infinite number of alternative loading matrices that generate the same covariance structure as $\mathbf{\Gamma}$. Thus, in any solution, $\mathbf{\Gamma}$ can be made to satisfy $m(m-1)/2$ additional conditions, and, hence, the structure of the existence condition given above. Naturally, any estimation method then requires a minimum of $m(m-1)/2$ restrictions on $\mathbf{\Gamma}$ to attain uniqueness (up to possibly polarity reversals in the columns of $\mathbf{\Gamma}$). In MLE this is achieved by requiring that $\mathbf{\Gamma}^{\mathrm{T}}\mathbf{\Psi}^{-1}\mathbf{\Gamma}$ be diagonal along with an order-condition on its diagonal elements. As this is a convenience solution that has no direct interpretational meaning, usually a post-hoc rotation is applied, whence estimation is settled, to enhance interpretation. Here, the Varimax [10] rotation to an orthogonal simple structure was ultimately used as oblique rotation (that allows for factor correlations) indicated a near-zero correlation between the two retained latent factors (note that any orthogonal representation has equivalent oblique representations).

The final factor solution is represented in Figure S4. This Dandelion plot [15] visualizes the magnitudes of the elements of $\hat{\mathbf{\Lambda}}$ and $\hat{\mathbf{\Psi}}$ and their relation



| ID | HUGO | Loading on |
|---|---|---|
| 1 | AKT1 | 2 |
| 2 | AKT2 | 2 |
| 3 | AKT3 | 2 |
| 4 | BAD | 2 |
| 5 | CASP9 | 1 |
| 6 | CDC42 | 1 |
| 7 | CHP1 | none |
| 8 | CHP2 | 1 |
| 9 | HRAS | 2 |
| 10 | HSPB1 | 2 |
| 11 | JMJD7,PLA2G4B | 2 |
| 12 | KDR | none |
| 13 | KRAS | 1 |
| 14 | MAP2K1 | 1 |
| 15 | MAP2K2 | 2 |
| 16 | MAPK1 | 1 |
| 17 | MAPK11 | 1 |
| 18 | MAPK12 | 2 |
| 19 | MAPK13 | 1 |
| 20 | MAPK14 | 1 |
| 21 | MAPK3 | 2 |
| 22 | MAPKAPK2 | 2 |
| 23 | MAPKAPK3 | 2 |
| 24 | NFAT5 | 1 |
| 25 | NFATC1 | 1 |
| 26 | NFATC2 | none |
| 27 | NFATC3 | 1 |
| 28 | NFATC4 | 1 |
| 29 | NOS3 | 1 |
| 30 | NRAS | 1 |
| 31 | PIK3CA | 1 |
| 32 | PIK3CB | 1 |
| 33 | PIK3CD | 1 |
| 34 | PIK3CG | 1 |
| 35 | PIK3R1 | 1 |
| 36 | PIK3R2 | 2 |
| 37 | PIK3R5 | 1 |
| 38 | PLA2G10 | 1 |
| 39 | PLA2G12A | 1 |
| 40 | PLA2G12B | 1 |
| 41 | PLA2G1B | 1 |
| 42 | PLA2G2A | none |
| 43 | PLA2G2C | 1 |
| 44 | PLA2G2D | 1 |
| 45 | PLA2G2E | 1 |
| 46 | PLA2G2F | 1 |
| 47 | PLA2G3 | 1 |
| 48 | PLA2G4A | 1 |
| 49 | PLA2G4B | 2 |
| 50 | PLA2G4E | 1 |
| 51 | PLA2G5 | 1 |
| 52 | PLA2G6 | 2 |
| 53 | PLCG1 | 2 |
| 54 | PLCG2 | 1 |
| 55 | PPP3CA | 1 |
| 56 | PPP3CB | 1 |
| 57 | PPP3CC | 1 |
| 58 | PPP3R1 | 1 |
| 59 | PPP3R2 | 2 |
| 60 | PRKCA | 2 |
| 61 | PRKCB | 1 |
| 62 | PRKCG | 1 |
| 63 | PTGS2 | 2 |
| 64 | PTK2 | 1 |
| 65 | PXN | 2 |
| 66 | RAC1 | 1 |
| 67 | RAC2 | 1 |
| 68 | RAC3 | 1 |
| 69 | RAF1 | 1 |
| 70 | SH2D2A | 1 |
| 71 | SHC2 | 2 |
| 72 | SPHK1 | 1 |
| 73 | SPHK2 | 1 |
| 74 | SRC | none |
| 75 | VEGFA | 2 |

| | Var. | Cum. Ratio |
|---|---|---|
| fac. 1 | 20.63 | 0.28 |
| fac. 2 | 8.05 | 0.38 |

**Fig. S4** Dandelion plot of the factor solution. Each central solid line represents a factor and is connected to a star plot (also known as a Kiviat diagram). Each spoke in each of the star plots then represents a variable. The length of the spokes corresponds to the maximum magnitude of each loading (which is unity). The extension of the polygon along each spoke then indicates the magnitude of a factor loading in the obtained solution, with the sign of the corresponding loading represented by color coding: green for a negative loading and blue for a positive loading. The representation suppresses absolute loadings lower than .3 for enhanced visualization. The angle between the solid lines corresponds to the amount of variance explained by the first factor (approximately 28%). The dashed line represents the cumulative percentage of variance explained by all retained factors (approximately 38% in this case). The star plots at the far-right then represent the magnitudes of the communalities $\kappa_j = \sum_j \gamma_{jk}^2$ and the uniquenesses $\psi_j = 1 - \kappa_j$. Variables are represented by index numbers. The table at the far-left gives the corresponding HUGO-curated gene names. See [15] for more information on the Dandelion plot.

to the two retained factors. The latent factors are taken to represent biochemical subpwathways. Table S1 characterizes the factors according to their constituting genes that achieve the highest absolute loading. These genes are furthermore associated with VEGF-signaling subpathways as indicated by the WikiPathways database [14]. Factor 1 consists mostly of MAPK-related genes. In addition, genes related to the glycerol phospholipid biosynthesis pathway load on factor 1. The MAPK pathway has been associated with lipid home-

**Table S1** Top genes per factor according to absolute factor loadings. 'HUGO' refers to Human Genome Organization Gene Nomenclature Committee (HGNC) curated gene names. 'Pathway' refers to the VEGF-subpathway a certain gene belongs to according to the WikiPathways database [14].

| Factor | HUGO Symbol | Pathway |
|---|---|---|
| **1** | CDC42 | MAPK-signaling |
| | MAPK14 | MAPK-signaling |
| | PIK3R5 | - |
| | PLA2G2C | Glycerol phospholipid biosynthesis |
| | PLA2G3 | Glycerol phospholipid biosynthesis |
| | PLA2G4E | Glycerol phospholipid biosynthesis |
| | PPP3R1 | MAPK-signaling |
| | PRKCG | MAPK-signaling |
| | SPHK1 | VEGF-signaling |
| | SPHK2 | - |
| **2** | AKT1 | PI3K/AKT-signaling in cancer |
| | AKT2 | AKT-signaling / VEGF-signaling |
| | AKT3 | AKT-signaling / VEGF-signaling |
| | BAD | AKT-signaling / Prostate cancer |
| | MAP2K2 | MAPK-signaling / Prostate cancer |
| | MAPKAPK2 | MAPK-signaling / Prostate cancer |
| | PIK3R2 | AKT-signaling / VEGF-signaling |
| | PLA2G6 | Glycerol phospholipid biosynthesis |
| | PXN | VEGF-signaling / Prostate cancer |
| | SHC2 | VEGF-signaling |

ostase in yeast [17] and "phospholipid composition and synthesis are similar in yeast and mammalian cells" [5]. Genes with pathway association indicated by '-' in Table S1 (PIK3R5 and SPHK2) could not be associated to VEGF-subpathways according to the WikiPathways database. PIK3R5 is, however, related to the MAPK-signaling pathway according to the Ingenuity Target Explorer [6]. Factor 2 consists mostly of genes related to the PI3K/AKT-signaling and prostate cancer pathways. Moreover, this factor also includes general VEGF-signaling pathway genes that are conducive in activating the MAPK and PI3K/AKT cascade. These results suggest that the compositional subpathway structure of the VEGF-signaling pathway in metastatic prostate cancer can still be characterized as consisting of the MAPK and PI3K-AKT cascades. Hence, VEGF-pathway degerulation in metastatic prostate cancer is more likely characterized by intricate changes in its known subpathways than by the loss of these (normal) subpathways or the gain of (abnormal) subpathways.

## 3. Benchmark

A benchmarking exercise was conducted to get an indication of the execution time of the `CNplot` function. The sample size $n$ is not a factor in the execution time as the function operates on the covariance (or precision) matrix. Hence, time complexity hinges on the dimension $p$ and the number of steps $S$ taken along the (specified) domain of the penalty-parameter. Execution time was thus evaluated for various combinations of $p$ and $S$.

Timing evaluations were done for all ridge estimators mentioned in Section 2.1 of the main text and all combinations of the elements in $p \in \{125, 250, 500, 1000\}$ and $S \in \{125, 250, 500, 1000\}$. The basic condition number plot was produced 50 times (on simulated covariance matrices) for each combination of estimator, $p$, and $S$. In addition, both a rotation equivariant situation (using a scalar target) and a rotation non-equivariant situation (using a non-scalar target) were considered for estimators (1) and (3) of the main text (note that estimator (2) is always rotation equivariant). All execution times of the `CNplot` call in R were evaluated with the `microbenchmark` package [16]. All timings were carried out on a Intel$^{\circledR}$ Core$^{\text{TM}}$ i7-4702MQ 2.2 GHz processor. The results, stated as median runtimes in seconds, are given in Tables S2 to S4. The code used in this benchmarking exercise can be found in Listing 6 in Section 4 of this Supplement.

Tables S2 to S4 indicate that, in general, the condition number plot can be produced quite swiftly. As stated in the main text: in rotation equivariant situations only a single run of the implicitly restarted Lanczos method (IRLM) is required to obtain the complete solution path of the condition number. Hence, the worst-case time complexity of the `CNplot` call in these situations is approximately $\mathcal{O}(p^3)$ (corresponding to the worst-case time complexity of a spectral decomposition) as, after the required IRLM run, the solution path can be obtained in linear time only. Increasing the number of steps $S$ along the penalty-domain thus comes at little additional computational cost. For the rotation non-equivariant setting the worst-case time complexity is approximately $\mathcal{O}(Sp^3)$ as a spectral decomposition or IRLM run is required for all $s = 1, \ldots, S$. The runtime of the condition number plot function under Estimator (3) of the main text is then somewhat longer than the corresponding runtime under Estimator (1). This is because the spectral decomposition in the former situation is also used for computing the (relatively expensive) matrix square root. As $S$ acts as a scaling factor in rotation non-equivariant settings the computation time of the plot can be reduced by coarsening the search-grid along the penalty-domain.

To compare the runtimes for the `CNplot` call we also conducted timing evaluations for the approximate leave-one-out cross-validation (aLOOCV) and root-finding LOOCV procedures as used in the main text (implemented in `rags2ridges`). Time complexity for the root-finding LOOCV procedure hinges on the dimension $p$ and the sample size $n$. In contrast, time complexity for for the aLOOCV procedure is dependent on $p$, $n$, and $S$. Timings were done for all combinations of the elements in $p \in \{125, 250\}$, $n \in \{100, 200\}$ and (when

**Table S2** Benchmarking results for various values of $p$ and $S$ for estimator (3) of the main text. The results represent median runtimes in seconds.

| | $p = 125$ | $p = 250$ | $p = 500$ | $p = 1000$ |
|---|---|---|---|---|
| | Rotation equivariant setting | | | |
| $S = 125$ | .027 | .039 | .055 | .345 |
| $S = 250$ | .028 | .038 | .055 | .349 |
| $S = 500$ | .028 | .039 | .058 | .344 |
| $S = 1000$ | .031 | .044 | .059 | .345 |
| | Rotation non-equivariant setting | | | |
| $S = 125$ | .689 | 3.267 | 18.801 | 139.982 |
| $S = 250$ | 1.315 | 6.661 | 39.022 | 278.454 |
| $S = 500$ | 2.703 | 13.418 | 76.489 | 553.812 |
| $S = 1000$ | 4.995 | 26.112 | 152.427 | 1,106.002 |

**Table S3** Benchmarking results for various values of $p$ and $S$ for estimator (1) of the main text. The results represent median runtimes in seconds.

| | $p = 125$ | $p = 250$ | $p = 500$ | $p = 1000$ |
|---|---|---|---|---|
| | Rotation equivariant setting | | | |
| $S = 125$ | .028 | .039 | .161 | .371 |
| $S = 250$ | .027 | .038 | .164 | .371 |
| $S = 500$ | .029 | .039 | .169 | .373 |
| $S = 1000$ | .031 | .042 | .168 | .378 |
| | Rotation non-equivariant setting | | | |
| $S = 125$ | .339 | 1.448 | 9.701 | 65.988 |
| $S = 250$ | .590 | 2.849 | 18.128 | 129.262 |
| $S = 500$ | 1.090 | 5.385 | 35.188 | 281.583 |
| $S = 1000$ | 2.366 | 12.199 | 75.222 | 522.910 |

**Table S4** Benchmarking results for various values of $p$ and $S$ for estimator (2) of the main text. The results represent median runtimes in seconds.

| | $p = 125$ | $p = 250$ | $p = 500$ | $p = 1000$ |
|---|---|---|---|---|
| | Rotation equivariant setting | | | |
| $S = 125$ | .027 | .041 | .101 | .345 |
| $S = 250$ | .028 | .042 | .107 | .343 |
| $S = 500$ | .028 | .042 | .106 | .361 |
| $S = 1000$ | .030 | .043 | .140 | .376 |

relevant) $S \in \{125, 250\}$. The LOOCV procedures were run 50 times (on simulated data of dimension $n \times p$) for each combination of $p$, $n$ and (possibly) $S$. For the root-finding LOOCV procedure both a rotation equivariant situation and a rotation non-equivariant situation were considered. Ridge estimator (3) of the main text was used for all timing evaluations. The results, stated as median runtimes in seconds, are given in Tables S5 and S6. The code used in this exercise can also be found in Listing 6 in Section 4 of this Supplement. Tables S5 and S6 indicate that, even under these relatively light settings, the runtimes for the aLOOCV and root-finding LOOCV procedures far exceed the runtime of (corresponding) calls to `CNplot`. The runtime for the root-finding LOOCV

**Table S5** Benchmarking results for various values of $p$ and $n$ for the root-finding LOOCV procedure. The ridge estimator considered is given in equation (3) of the main text. The results represent median runtimes in seconds.

| | $p = 125$ | $p = 250$ |
|---|---|---|
| | Rotation equivariant setting | |
| $n = 100$ | 36.843 | 186.662 |
| $n = 200$ | 83.932 | 438.080 |
| | Rotation non-equivariant setting | |
| $n = 100$ | 37.151 | 186.780 |
| $n = 200$ | 84.335 | 439.607 |

**Table S6** Benchmarking results for various values of $p$, $n$, and $S$ for the approximate LOOCV procedure. The ridge estimator considered is given in equation (3) of the main text. The results represent median runtimes in seconds.

| | $p = 125$ | $p = 250$ |
|---|---|---|
| | $n = 100$, Rotation equivariant setting | |
| $S = 125$ | 53.059 | 393.309 |
| $S = 250$ | 100.733 | 802.741 |
| | $n = 200$, Rotation equivariant setting | |
| $S = 125$ | 105.372 | 779.881 |
| $S = 250$ | 212.129 | 1561.064 |

procedure is (given $p$) exacerbated for larger sample sizes. The runtime for the aLOOCV procedure is (given $p$) exacerbated for larger samples sizes, finer search-grids, and (not shown) situations of rotation non-equivariance.

The LOOCV is often preferred for its predictive accuracy. This comes at the price of relatively heavy computational loads. One can choose $k < n$, giving a general $k$-fold CV ($k$CV), for more efficient computation in the cross-validation setting. We have performed an additional comparative exercise in this respect. Again, the $k$CV procedure is combined with a root-finding method. The time complexity for the root-finding $k$CV procedure hinges on the dimension $p$ and the choice of $k$. We choose $k = 5$, which is a small popular choice in practice. Timings were assessed for all combinations of the elements in $p \in \{125, 250, 1000\}$ and $n \in \{100, 200\}$. The root-finding $k$CV procedure was run 50 times (on simulated data of dimension $n \times p$) for each combination of $p$ and $n$. Both a rotation equivariant situation and a rotation non-equivariant situation were considered. Ridge estimator (3) of the main text was again used for all timing evaluations. The results, stated as median runtimes in seconds, are given in Table S7. The code used in this exercise is also found in Listing 6 in Section 4 of this Supplement.

The results show that the procedure does not depend on $n$ or equivariance in a manner that is relevant for the computational complexity. In addition, they show that the root-finding 5CV procedure is fast compared to the preceding root-finding and approximate LOOCV procedures. In rotation non-equivariant situations the root-finding 5CV procedure may be faster than (corresponding) calls to `CNplot` when the latter would take many steps along the penalty do-

**Table S7** Benchmarking results for various values of $p$ and $n$ for the root-finding $k$CV procedure. The ridge estimator considered is given in equation (3) of the main text. The results represent median runtimes in seconds.

| | $p = 125$ | $p = 250$ | $p = 1000$ |
|---|---|---|---|
| | Rotation equivariant setting | | |
| $n = 100$ | 1.55 | 8.34 | 403.20 |
| $n = 200$ | 1.80 | 8.64 | 398.37 |
| | Rotation non-equivariant setting | | |
| $n = 100$ | 1.40 | 7.28 | 374.61 |
| $n = 200$ | 1.77 | 8.57 | 470.47 |

main. It seems, however, that one could always coarsen the grid along the penalty-domain to outperform root-finding 5CV in this situation. For rotation equivariant situations (the most common situation) the root-finding 5CV procedure cannot match the speed of `CNplot`. For example, in a situation with $p = 1000$ where we take 1000 steps along the penalty-domain the median runtime of a call to `CNplot` is more than 1000-fold faster than the median runtime of a corresponding root-finding 5CV procedure.

## 4. R Codes

The R libraries needed to run the code snippets below can be found in Listing 1:

**Listing 1** Packages and dependencies

```
#####################################################
##---------------------------------------------------
## Packages and dependencies
##---------------------------------------------------
#####################################################

## R libraries
library(biomaRt)
library(cgdsr)
library(KEGG.db)
library(microbenchmark)
library(psych)
library(DandEFA)
library(gridExtra)
library(plyr)
library(rags2ridges)
```

The code in Listing 2 will produce Figure 1 of the main text:

**Listing 2** Code for Figure 1

```
#####################################################
##---------------------------------------------------
## First example of copy number Plot
##---------------------------------------------------
```

```
####################################################

## Obtain covariance matrix on p > n data
p = 100
n = 25
set.seed(333)
X = matrix(rnorm(n*p), nrow = n, ncol = p)
Cx <- covML(X)

## Obtain basic spectral condition number plot
CNplot(Cx, lambdaMin = .001, lambdaMax = 10, step = 1000,
       type = "ArchII")

## Condition number at exp(-3)
EVs <- eigen(ridgeP(Cx, lambda = exp(-3),
                    type = "ArchII"))$values
Cn  <- EVs[1]/EVs[ncol(Cx)]; Cn
```

The code in Listing 3 will produce (the components of) Figure 2 of the main text:

**Listing 3** Code for Figure 2

```
####################################################
##--------------------------------------------------
## Example copy number Plot with interpretational aids
##--------------------------------------------------
####################################################

## Spectral condition number plot with interpretational aids
CNplot(Cx, lambdaMin = .00001, lambdaMax = 1000, step = 2000,
       Iaids = TRUE, type = "Alt",
       target = default.target(Cx, type = "DEPV"))

## Zoom
CNplot(Cx, lambdaMin = .01, lambdaMax = 1000, step = 2000,
       Iaids = TRUE, type = "Alt",
       target = default.target(Cx, type = "DEPV"))
```

The code in Listing 4 was used for Section 4 of the main text:

**Listing 4** Code for Illustration 1

```
####################################################
##--------------------------------------------------
## Illustration 1: Kidney cancer data
##--------------------------------------------------
####################################################

####################################################
## Probe MSKCC Cancer Genomics Data Server for data
####################################################

## Get list all human genes
ensembl = useMart("ENSEMBL_MART_ENSEMBL",
                  dataset = "hsapiens_gene_ensembl",
                  host="www.ensembl.org")
```

```
geneList <- getBM(attributes = c("hgnc_symbol", "entrezgene"),
                  mart = ensembl)
geneList <- geneList[!is.na(geneList[,2]),]

## Obtain entrez IDs of genes that map to Hedgehog signaling
    pathway
kegg2entrez <- as.list(KEGGPATHID2EXTID)
entrezIDs   <- as.numeric(kegg2entrez[which(names(kegg2entrez) %in
    % "hsa04340")][[1]])
entrez2name <- match(entrezIDs, geneList[,2])
geneList    <- geneList[entrez2name[!is.na(entrez2name)],]

## Specify data set details
mskccDB     <- CGDS("http://www.cbioportal.org/")
studies     <- getCancerStudies(mskccDB)
cancerStudy <- "kirc_tcga_pub"
caseList    <- getCaseLists(mskccDB, cancerStudy)
caseList    <- getCaseLists(mskccDB, cancerStudy)[3,1]
mygeneticprofile = getGeneticProfiles(mskccDB,cancerStudy)
mrnaProf    <- "kirc_tcga_pub_rna_seq_v2_mrna_median_Zscores"

## Extract data
Y <- getProfileData(mskccDB, geneList[,1], mrnaProf, caseList)
Y <- as.matrix(Y)

## Filter no-data samples and genes and scale data
sRemove <- which(rowSums(is.na(Y)) > ncol(Y)/2); sRemove
gRemove <- which(colSums(is.na(Y)) > 0); gRemove
Y       <- scale(Y, center = TRUE, scale = TRUE)


####################################################
## Regularize the precision matrix
####################################################

## Approximate LOOCV
## Chooses very small penalty
aLOOCVres <- optPenalty.aLOOCV(Y, 1e-05, 20, 10000,
                               type = "Alt", cor = TRUE,
                               target = default.target(cor(Y),
                                                       type = "
                                                           DUPV"))

## Condition number plot
## aLOOCV penalty indeed too small
## Can give idea good heuristic value for penalty
CNplot(cor(Y), 1e-05, 20, 5000, type = "Alt",
       target = default.target(cor(Y), type = "DUPV"))

## Perform LOOCV (with Brent) and restrict search space on basis
    CnPlot
LOOCVres  <- optPenalty.LOOCVauto(Y, exp(-6), 20,
                                  type = "Alt", cor = TRUE,
                                  target = default.target(cor(Y),
                                                       type = "
                                                           DUPV"))
LOOCVres$optLambda
```

```
## Condition number plot with optimal LOOCV-determined penalty
    indicated
## 'Optimal' approx. LOOCV-determined penalty is also indicated
CNplot(cor(Y), 1e-05, 20, 5000, type = "Alt",
       target = default.target(cor(Y), type = "DUPV"),
       vertical = TRUE, value = LOOCVres$optLambda)
abline(v = log(aLOOCVres$optLambda), col = "green", lty = 2, lwd =
     2)


#####################################################
## Assessment condition numbers
#####################################################

## Condition number at optimal value indicated by aLOOCV
EVs <- eigen(ridgeP(cor(Y), lambda = 1e-05, type = "Alt",
                    target = default.target(cor(Y),
                                            type = "DUPV")))$
                                                values
Cn  <- EVs[1]/EVs[ncol(Y)]; Cn

## Condition number at heuristic value
EVs <- eigen(ridgeP(cor(Y), lambda = exp(-6), type = "Alt",
                    target = default.target(cor(Y),
                                            type = "DUPV")))$
                                                values
Cn  <- EVs[1]/EVs[ncol(Y)]; Cn

## Condition number at optimal value indicated by LOOCV
EVs <- eigen(ridgeP(cor(Y), lambda = LOOCVres$optLambda, type = "
    Alt",
                    target = default.target(cor(Y),
                                            type = "DUPV")))$
                                                values
Cn  <- EVs[1]/EVs[ncol(Y)]; Cn


#####################################################
## Downstream graphical modeling
#####################################################

Pp0 <- sparsify(LOOCVres$optPrec, "localFDR", FDRcut = .8)
edgeHeat(Pp0$sparseParCor)
Ugraph(Pp0$sparseParCor, type = "fancy",
       lay = "layout_with_fr",
       Vcolor = "white", VBcolor = "black",
       Vcex = .5, cut = .Machine$double.xmin,
       prune = T)
```

The code in Listing 5 was used for the second illustration contained in Section 2 of this supplement:

**Listing 5** Code for Illustration 2

```
#####################################################
##---------------------------------------------------
## Illustration 2: Prostate cancer data
```

```
##----------------------------------------------------
####################################################

####################################################
## Probe MSKCC Cancer Genomics Data Server for data
####################################################

## Get list all human genes
ensembl = useMart("ENSEMBL_MART_ENSEMBL",
                  dataset = "hsapiens_gene_ensembl",
                  host="www.ensembl.org")
geneList <- getBM(attributes = c("hgnc_symbol", "entrezgene"),
                  mart = ensembl)
geneList <- geneList[!is.na(geneList[,2]),]

## Obtain entrez IDs of genes that map to VEGF signaling pathway
kegg2entrez <- as.list(KEGGPATHID2EXTID)
entrezIDs   <- as.numeric(kegg2entrez[which(names(kegg2entrez) %in
    % "hsa04370")][[1]])
entrez2name <- match(entrezIDs, geneList[,2])
geneList    <- geneList[entrez2name[!is.na(entrez2name)],]

## Specify data set details
mskccDB     <- CGDS("http://www.cbioportal.org/")
studies     <- getCancerStudies(mskccDB)
cancerStudy <- "prad_mskcc"
caseList    <- getCaseLists(mskccDB, cancerStudy)
caseList    <- getCaseLists(mskccDB, cancerStudy)[15,1]
mygeneticprofile = getGeneticProfiles(mskccDB,cancerStudy)
mrnaProf    <- "prad_mskcc_mrna_median_Zscores"

## Extract data
Y2 <- getProfileData(mskccDB, geneList[,1], mrnaProf, caseList)
Y2 <- as.matrix(Y2)

## Filter no-data samples and genes and scale data
sRemove <- which(rowSums(is.na(Y2)) > ncol(Y2)/2); sRemove
gRemove <- which(colSums(is.na(Y2)) > 0); gRemove
Y2      <- scale(Y2, center = TRUE, scale = TRUE)


####################################################
## Regularize the precision matrix
####################################################

## Approximate LOOCV
## Chooses very small penalty
aLOOCVres <- optPenalty.aLOOCV(Y2, 1e-05, 20, 10000,
                               type = "Alt", cor = TRUE,
                               target = default.target(cor(Y2),
                                                       type = "
                                                           DUPV"))
aLOOCVres$optLambda

## Condition number plot
## aLOOCV penalty indeed too small
## Can give idea good heuristic value for penalty
CNplot(cor(Y2), 1e-05, 20, 5000, type = "Alt",
       target = default.target(cor(Y2), type = "DUPV"))

## Perform LOOCV (with Brent) and restrict search space on basis
    CnPlot
LOOCVres  <- optPenalty.LOOCVauto(Y2, exp(-6.5), 20,
                                  type = "Alt", cor = TRUE,
                                  target = default.target(cor(Y2),
                                                       type = "
                                                           DUPV"))
LOOCVres$optLambda

## Condition number plot with optimal LOOCV-determined penalty
    indicated
CNplot(cor(Y2), 1e-05, 20, 5000, type = "Alt",
       target = default.target(cor(Y2), type = "DUPV"),
       vertical = TRUE, Iaids = TRUE, value = LOOCVres$optLambda)
CNplot(cor(Y2), exp(-6.5), 20, 5000, type = "Alt",
       target = default.target(cor(Y2), type = "DUPV"),
       vertical = TRUE, Iaids = TRUE, value = LOOCVres$optLambda)


####################################################
## Assessment condition numbers
####################################################

## Condition number at optimal value indicated by aLOOCV
EVs <- eigen(ridgeP(cor(Y2), lambda = aLOOCVres$optLambda,
                    type = "Alt",
                    target = default.target(cor(Y2),
                                            type = "DUPV")))$
                                                values
Cn  <- EVs[1]/EVs[ncol(Y2)]; Cn

## Condition number at heuristic value
EVs <- eigen(ridgeP(cor(Y2), lambda = exp(-6.5), type = "Alt",
                    target = default.target(cor(Y2),
                                            type = "DUPV")))$
                                                values
Cn  <- EVs[1]/EVs[ncol(Y2)]; Cn

## Condition number at optimal value indicated by LOOCV
EVs <- eigen(ridgeP(cor(Y2), lambda = LOOCVres$optLambda,
                    type = "Alt",
                    target = default.target(cor(Y2),
                                            type = "DUPV")))$
                                                values
Cn  <- EVs[1]/EVs[ncol(Y2)]; Cn


####################################################
## Downstream factor analytic modeling
####################################################

##--------------------------------------
## Obtain regularized correlation matrix
## Assess factorability
##--------------------------------------
R = cov2cor(solve(LOOCVres$optPrec))
```

```
KMO(R)

##--------------------------------------
## Determine dimension latent vector
##--------------------------------------

## Function
BICfacM <- function(S, n, m){
  ######################################
  ## S > (regularized) covariance or
  ##      correlation matrix
  ## n > sample size
  ## m > desired number of factors
  ######################################

  ## Preliminaries
  p     <- ncol(S)
  fit   <- factanal(factors = m, covmat = S, rotation = "none")
  loadings <- fit$loadings[1:p,]
  Uniqueness <- diag(fit$uniquenesses)
  Sfit <- loadings %*% t(loadings) + Uniqueness

  ## Calculate BIC
  fit <- n * (p*log(2*pi) + log(det(Sfit)) +
         sum(diag(solve(Sfit) %*% S)))
  penalty <- p*(m+1) - (m*(m-1))/2
  BIC <- fit + penalty

  ## Return
  return(BIC)
}

## Ledermann bound
p     <- ncol(R)
mmax <- floor((2*p+1 - sqrt(8*p+1))/2)

## Determine optimal dimension
BIC <- numeric()
for(m in 1:(mmax - 1)){
  BIC[m] <- BICfacM(R, n = 19, m = m)
}; BIC

## Plot
dims <- seq(1, (mmax - 1), 1)
plot(dims, BIC, axes = FALSE, type = "l",
     col = "red", xlab = "dimension␣of␣latent␣vector",
     ylab = "BIC␣score")
axis(2, ylim = c(min(BIC),max(BIC)), col = "black", lwd = 1)
axis(1, xlim = c(0,(mmax - 1)), col = "black", lwd = 1, tick =
    TRUE)

##--------------------------------------
## Fit under optimal dimension
##--------------------------------------
fit <- factanal(factors = 2, covmat = R, rotation = "promax")
print(fit, digits = 2, cutoff = .3, sort = FALSE)
fit <- factanal(factors = 2, covmat = R, rotation = "varimax")
print(fit, digits = 2, cutoff = .3, sort = TRUE)
```

```
##--------------------------------------
## Visualizing solution
##--------------------------------------

## Dandelion plot
Loading <- character()
for (i in 1:nrow(fit$loadings)){
  if (abs(fit$loadings[i,1]) >= .3 & abs(fit$loadings[i,1]) > abs(
      fit$loadings[i,2])){
    Loading[i] <- "1"
  }
  if (abs(fit$loadings[i,2]) >= .3 & abs(fit$loadings[i,2]) > abs(
      fit$loadings[i,1])){
    Loading[i] <- "2"
  }
  if (abs(fit$loadings[i,1]) == abs(fit$loadings[i,2])){
    Loading[i] <- "both"
  }
  if (abs(fit$loadings[i,1]) < .3 & abs(fit$loadings[i,2]) < .3){
    Loading[i] <- "none"
  }
}
Names <- rownames(fit$loadings)
rownames(fit$loadings) <- c(1:ncol(R))
Ident <- as.data.frame(cbind(c(1:ncol(R)), Names, Loading))
colnames(Ident) <- c("ID","HUGO","Loading␣on")

pdf("Identifiers.pdf", height = 23)
grid.table(Ident)
dev.off()

dandpal <- rev(rainbow(100, start = 0.4, end = 0.6))
dandelion(fit$loadings, bound = .3, mcex = c(1,1), palet = dandpal
    )
```

The code in Listing 6 was used in the benchmark exercise contained in Section 3 of this supplement:

**Listing 6** Code used in benchmarking

```
####################################################
##--------------------------------------------------
## Benchmark Cn-plot and other methods
##--------------------------------------------------
####################################################

S <- c(125,250,500,1000)
p <- c(125,250,500,1000)

####################################################
## Alternative ridge estimator (equation 3)
####################################################

## Rotation equivariant setting
seed <- 1234
for (i in 1:length(S)){
  for (j in 1:length(p)){
    ## Generate data
    su = S[i]
    pu = p[j]
    nu = 200
    set.seed(seed)
    Y = matrix(rnorm(nu*pu), nrow = nu, ncol = pu)
    Target <- default.target(cor(Y), type = "DUPV")
    Sy <- cor(Y)

    ## Benchmark
    tm <- microbenchmark(CNplot(Sy, .00001, 20,
                                step = su, type = "Alt",
                                target = Target,
                                verbose = FALSE),
                         times = 50L)

    ## Save
    tm$expr <- mapvalues(tm$expr,
                         from = c(levels(tm$expr)[1]),
                         to = c("Condition␣number␣plot"))
    save(tm, file = paste("Alt.BM.RE.S",su,"p",pu,".Rdata", sep =
        ""))

    ## Plot
    boxplot(tm, log = FALSE)
  }
}

## Rotation non-equivariant setting
seed <- 5678
for (i in 1:length(S)){
  for (j in 1:length(p)){
    ## Generate data
    su = S[i]
    pu = p[j]
    nu = 200
    set.seed(seed)
    Y = matrix(rnorm(nu*pu), nrow = nu, ncol = pu)
    Target <- default.target(cor(Y), type = "DUPV")
    Target[1,1] <- 2
    Sy <- cor(Y)

    ## Benchmark
    tm <- microbenchmark(CNplot(Sy, .00001, 20,
                                step = su, type = "Alt",
                                target = Target,
                                verbose = FALSE),
                         times = 50L)

    ## Save
    tm$expr <- mapvalues(tm$expr,
                         from = c(levels(tm$expr)[1]),
                         to = c("Condition␣number␣plot"))
    save(tm, file = paste("Alt.BM.RNE.S",su,"p",pu,".Rdata", sep =
         ""))

    ## Plot
    boxplot(tm, log = FALSE)
  }
}


####################################################
## Archetypal Type I ridge estimator (equation 1)
####################################################

## Rotation equivariant setting
seed <- 9101112
for (i in 1:length(S)){
  for (j in 1:length(p)){
    ## Generate data
    su = S[i]
    pu = p[j]
    nu = 200
    set.seed(seed)
    Y = matrix(rnorm(nu*pu), nrow = nu, ncol = pu)
    Target <- default.target(cor(Y), type = "DUPV")
    Sy <- cor(Y)

    ## Benchmark
    tm <- microbenchmark(CNplot(Sy, .00001, 1,
                                step = su, type = "ArchI",
                                target = Target,
                                verbose = FALSE),
                         times = 50L)

    ## Save
    tm$expr <- mapvalues(tm$expr,
                         from = c(levels(tm$expr)[1]),
                         to = c("Condition␣number␣plot"))
    save(tm, file = paste("ArchI.BM.RE.S",su,"p",pu,".Rdata",
         sep = ""))

    ## Plot
    boxplot(tm, log = FALSE)
  }
}

## Rotation non-equivariant setting
seed <- 13141516
for (i in 1:length(S)){
  for (j in 1:length(p)){
    ## Generate data
    su = S[i]
    pu = p[j]
    nu = 200
    set.seed(seed)
    Y = matrix(rnorm(nu*pu), nrow = nu, ncol = pu)
    Target <- default.target(cor(Y), type = "DUPV")
    Target[1,1] <- 2
    Sy <- cor(Y)


    ## Benchmark
    tm <- microbenchmark(CNplot(Sy, .00001, 1,
                                step = su, type = "ArchI",
                                target = Target,
                                verbose = FALSE),
                         times = 50L)

    ## Save
    tm$expr <- mapvalues(tm$expr,
                         from = c(levels(tm$expr)[1]),
                         to = c("Condition␣number␣plot"))
    save(tm, file = paste("ArchI.BM.RNE.S",su,"p",pu,".Rdata",
         sep = ""))

    ## Plot
    boxplot(tm, log = FALSE)
  }
}


####################################################
## Archetypal Type II ridge estimator (equation 2)
####################################################

seed <- 17181920
for (i in 1:length(S)){
  for (j in 1:length(p)){
    ## Generate data
    su = S[i]
    pu = p[j]
    nu = 200
    set.seed(seed)
    Y = matrix(rnorm(nu*pu), nrow = nu, ncol = pu)
    Sy <- cor(Y)

    ## Benchmark
    tm <- microbenchmark(CNplot(Sy, .00001, 20,
                                step = su, type = "ArchII",
                                verbose = FALSE),
                         times = 50L)

    ## Save
    tm$expr <- mapvalues(tm$expr,
                         from = c(levels(tm$expr)[1]),
                         to = c("Condition␣number␣plot"))
    save(tm, file = paste("ArchII.BM.RE.S",su,"p",pu,".Rdata",
         sep = ""))

    ## Plot
    boxplot(tm, log = FALSE)
  }
}


####################################################
## Root-finding LOOCV and approximate LOOCV
####################################################

n <- c(100,200)
p <- c(125,250)
```

```
S <- c(125,250)

## Root-finding LOOCV, equivariant
seed <- 90210
for (i in 1:length(n)){
  for (j in 1:length(p)){
    ## Generate data
    pu = p[j]
    nu = n[i]
    set.seed(seed)
    Y = matrix(rnorm(nu*pu), nrow = nu, ncol = pu)
    Target <- default.target(cor(Y), type = "DUPV")

    ## Benchmark
    tm <- microbenchmark(optPenalty.LOOCVauto(Y, .00001, 20,
                                              type = "Alt",
                                              target = Target),
                         times = 50L)

    ## Save
    tm$expr <- mapvalues(tm$expr,
                         from = c(levels(tm$expr)[1]),
                         to = c("Root-finding␣LOOCV"))
    save(tm, file = paste("rfLOOCV.BM.RE.n",nu,"p",pu,".Rdata",
         sep = ""))

    ## Plot
    boxplot(tm, log = FALSE)
  }
}

## Root-finding LOOCV, non-equivariant
seed <- 902102
for (i in 1:length(n)){
  for (j in 1:length(p)){
    ## Generate data
    pu = p[j]
    nu = n[i]
    set.seed(seed)
    Y = matrix(rnorm(nu*pu), nrow = nu, ncol = pu)
    Target <- default.target(cor(Y), type = "DUPV")
    Target[1,1] <- 2

    ## Benchmark
    tm <- microbenchmark(optPenalty.LOOCVauto(Y, .00001, 20,
                                              type = "Alt",
                                              target = Target),
                         times = 50L)

    ## Save
    tm$expr <- mapvalues(tm$expr,
                         from = c(levels(tm$expr)[1]),
                         to = c("Root-finding␣LOOCV"))
    save(tm, file = paste("rfLOOCV.BM.RNE.n",nu,"p",pu,".Rdata",
         sep = ""))

    ## Plot
    boxplot(tm, log = FALSE)
  }
}

## Approximate LOOCV, equivariant
seed <- 902103
for (i in 1:length(n)){
  for (j in 1:length(p)){
    for(k in 1:length(S)){
      ## Generate data
      pu = p[j]
      nu = n[i]
      su = S[k]
      set.seed(seed)
      Y = matrix(rnorm(nu*pu), nrow = nu, ncol = pu)
      Target <- default.target(cor(Y), type = "DUPV")

      ## Benchmark
      tm <- microbenchmark(optPenalty.aLOOCV(Y, .00001, 20, step =
           su,
                                             type = "Alt",
                                             target = Target,
                                             verbose = FALSE),
                          times = 50L)

      ## Save
      tm$expr <- mapvalues(tm$expr,
                           from = c(levels(tm$expr)[1]),
                           to = c("Approximate␣LOOCV"))
      save(tm, file = paste("aLOOCV.BM.RE.n",nu,"p",pu,"S",su,
           ".Rdata", sep = ""))

      ## Plot
      boxplot(tm, log = FALSE)
    }
  }
}


####################################################
## Root-finding k-CV
####################################################

n <- c(100,200)
p <- c(125,250,1000)

## Root-finding k-CV, equivariant
## k = 5
seed <- 902104
for (i in 1:length(n)){
  for (j in 1:length(p)){
    ## Generate data
    pu = p[j]
    nu = n[i]
    set.seed(seed)
    Y = matrix(rnorm(nu*pu), nrow = nu, ncol = pu)
    Target <- default.target(cor(Y), type = "DUPV")

    ## Benchmark
    tm <- microbenchmark(optPenalty.kCVauto(Y, .00001, 20,
                                            type = "Alt",
                                            target = Target,
                                            fold = 5),
                         times = 50L)

    ## Save
    tm$expr <- mapvalues(tm$expr,
                         from = c(levels(tm$expr)[1]),
                         to = c("Root-finding␣k-CV"))
    save(tm, file = paste("rfKCV.BM.RE.n",nu,"p",pu,".Rdata", sep
        = ""))

    ## Plot
    boxplot(tm, log = FALSE)
  }
}

## Root-finding k-CV, non-equivariant
## k = 5
seed <- 902105
for (i in 1:length(n)){
  for (j in 1:length(p)){
    ## Generate data
    pu = p[j]
    nu = n[i]
    set.seed(seed)
    Y = matrix(rnorm(nu*pu), nrow = nu, ncol = pu)
    Target <- default.target(cor(Y), type = "DUPV")
    Target[1,1] <- 2

    ## Benchmark
    tm <- microbenchmark(optPenalty.kCVauto(Y, .00001, 20,
                                            type = "Alt",
                                            target = Target,
                                            fold = 5),
                         times = 50L)

    ## Save
    tm$expr <- mapvalues(tm$expr,
                         from = c(levels(tm$expr)[1]),
                         to = c("Root-finding␣k-CV"))
    save(tm, file = paste("rfKCV.BM.RNE.n",nu,"p",pu,".Rdata", sep
        = ""))

    ## Plot
    boxplot(tm, log = FALSE)
  }
}
```

## References

1. American Cancer Society. URL http://www.cancer.org/cancer/prostatecancer/detailedguide/prostate-cancer-survival-rates.

2. R. P. Brent. An algorithm with guaranteed convergence for finding a zero of a function. *The Computer Journal*, 14:422–425, 1971.
3. C. M. Carvalho, J. Chang, J. E. Lucas, J. R. Nevins, Q. Wang, and M. West. High-dimensional sparse factor modeling: Applications in gene expression genomics. *Journal of the American Statistical Association*, 103: 1438–1456, 2008.
4. E. Cerami, J. Gao, U. Dogrusoz, B. E. Gross, S. O. Sumer, B. A. Aksoy, A. Jacobsen, C. J. Byrne, M. L. Heuer, E. Larsson, Y. Antipin, B. Reva, A. P. Goldberg, C. Sander, and N. Schultz. The cBio cancer genomics portal: An open platform for exploring multidimensional cancer genomics data. *Cancer Discovery*, 2:401–404, 2012.
5. A. I. P. M. de Kroon, P. J. Rijken, and C. H. de Smet. Checks and balances in membrane phospholipid class and acyl chain homeostasis, the yeast perspective. *Progress in Lipid Research*, 52:374–394, 2013.
6. QIAGEN Ingenuity Target Explorer. URL `https://targetexplorer.ingenuity.com`.
7. N. Ferrara, H. P. Gerber, and J. LeCouter. The biology of VEGF and its receptors. *Nature Medicine*, 9:669–676, 2003.
8. J. Gao, B. A. Aksoy, U. Dogrusoz, G. Dresdner, B. Gross, S. O. Sumer, Y. Sun, A. Jacobsen, R. Sinha, E. Larsson, E. Cerami, C. Sander, and N. Schultz. Integrative analysis of complex cancer genomics and clinical profiles using the cBioPortal. *Science Signaling*, 6:pl1, 2013.
9. A. Jacobsen. *cgdsr: R-Based API for Accessing the MSKCC Cancer Genomics Data Server (CGDS)*, 2015. URL `http://CRAN.R-project.org/package=cgdsr`. R package version 1.2.5.
10. H. F. Kaiser. The varimax criterion for analytic rotation in factor analysis. *Psychometrika*, 23:187–200, 1958.
11. H. F. Kaiser. A second generation little jiffy. *Psychometrika*, 35:401–415, 1970.
12. M. Kanehisa and S. Goto. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Research*, 28(1):27–30, 2000.
13. KEGG. URL `http://www.genome.jp/kegg-bin/show_pathway?hsa04370`.
14. M. Kutmon, A. Riutta, N. Nunes, K. Hanspers, E. L. Willighagen, A. Bohler, J. Mélius, A. Waagmeester, S. R. Sinha, R. Miller, S. L. Coort, E. Cirillo, B. Smeets, C. T. Evelo, and A. R. Pico. WikiPathways: capturing the full diversity of pathway knowledge. *Nucleic Acids Research*, 44(D1):D488–D494, 2016.
15. A. Manukyan, E. Çene, A. Sedef, and I. Demir. Dandelion plot: a method for the visualization of R-mode exploratory factor analyses. *Computational Statistics*, 29:1769–1791, 2014.
16. Olaf Mersmann. *microbenchmark: Accurate Timing Functions*, 2014. URL `http://CRAN.R-project.org/package=microbenchmark`. R package version 1.4-2.
17. L. R. Nunez, S. A. Jesch, M. L. Gaspar, C. Almaguer, M. Villa-Garcia, M. Ruiz-Noriega, J. Patton-Vogt, and S. A. Henry. Cell Wall Integrity

MAPK Pathway is essential for lipid homeostasis. *Journal of Biological Chemistry*, 283:34204 –34217, 2008.

18. G. E. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.
19. R. L. Siegel, K. D. Miller, and A. Jemal. Cancer Statistics, 2016. *CA: A Cancer Journal for Clinicians*, 66:7–30, 2016.
20. B. S. Taylor, N. Schultz, H. Hieronymus, A. Gopalan, Y. Xiao, B. S. Carver, V. K. Arora, P. Kaushik, E. Cerami, B. Reva, Y. Antipin, N. Mitsiades, T. Landers, I. Dolgalev, J. E. Major, M. Wilson, N. D. Socci, A. E. Lash, A. Heguy, J. A. Eastham, H. I. Scher, V. E. Reuter, P. T. Scardino, C. Sander, C. L. Sawyers, and W. L. Gerald. Integrative genomic profiling of human prostate cancer. *Cancer Cell*, 18:11–22, 2010.