singR: An R Package for Simultaneous Non-Gaussian Component Analysis for Data Integration

by Liangkang Wang, Irina Gaynanova, and Benjamin Risk

Abstract This paper introduces an R package singR that implements Simultaneous non-Gaussian Component Analysis (SING) for data integration. SING uses a non-Gaussian measure of information to extract feature loadings and scores (latent variables) that are shared across multiple datasets. We describe the functions implemented in singR and showcase their use on two examples. The first example is a toy example working with images. The second example is a simulated study integrating functional connectivity estimates from a resting-state functional magnetic resonance imaging dataset and task activation maps from a working memory functional magnetic resonance imaging dataset. The SING model can produce joint components that accurately reflect information shared by multiple datasets, particularly for datasets with non-Gaussian features such as neuroimaging.

1 Introduction

Combining information across different datasets collected on the same individuals is an important task, especially in biology and medicine. For example, in neuroimaging research, combining information across different modalities, or types of imaging data, can lead to a more comprehensive picture of human brain activity. Commonly used modalities to investigate brain function include functional magnetic resonance imaging (fMRI), resting-state fMRI (rs-fMRI), diffusion MRI, structural images, electroencephalography, and positron emission tomography. Combining data from multiple modalities can result in a better understanding of the underlying biology than analyzing each modality separately (Calhoun and Sui 2016). In previous studies, researchers use data fusion to define a set of joint components that are composed of subject scores (a vector in $\mathbb{R}^n$, where n is the number of subjects) and loadings (a vector in $\mathbb{R}^{p_k}$, where p_k is the number of variables in the kth dataset). For a particular component, the subject scores are equal or strongly correlated across datasets. The loadings represent the relative importance of each variable to each component. A higher subject score implies the vector of loadings is more important in that individual.

Data integration approaches for neuroimaging should accommodate the distinct statistical properties of imaging data. Imaging features characterizing brain activation and intrinsic functional connectivity are substantially more non-Gaussian compared to noise. Methods employing principal component analysis (PCA) for dimension reduction and independent component analysis (ICA) have been widely used in neuroscience (Calhoun and Sui 2016; J. Sui et al. 2012; Zhou et al. 2016). ICA maximizes the non-Gaussianity of components, which is useful for extracting interesting features from imaging data. ICA is commonly used to estimate resting-state networks (Beckmann et al. 2005), estimate task-activated components (Jing Sui et al. 2010), and has been used to derive network structure in resting-state correlation matrices (Amico et al. 2017). Risk and Gaynanova (2021) proposed simultaneous non-Gaussian component analysis (SING) for analyzing two datasets, which uses an objective function that maximizes the skewness and kurtosis of latent components with a penalty to enhance the similarity between subject scores. Unlike previous methods, SING does not use PCA for dimension reduction, but rather uses non-Gaussianity measured by skewness and kurtosis, which can improve feature extraction.

Some useful software have been developed in this area. In multimodal analysis and multitask fMRI data fusion, **FIT** is a Matlab toolbox that implements jointICA, parallel ICA (Vergara et al. 2014), and multimodal/multiset CCA (J. Sui et al. 2011). **GIFT** provides functions for conducting group ICA on fMRI data from multiple subjects from a single modality (Calhoun, Liu, and Adali 2009; Calhoun, Adali, Giuliani, et al. 2006; Calhoun, Adali, Kiehl, et al. 2006). On the Comprehensive R Archive Network (CRAN), there are several R packages for ICA functions, including **steadyICA** (Risk, James, and Matteson 2015), **ica** (Helwig 2018), **fastICA** (Marchini, Heaton, and Ripley 2021), **JADE** (Miettinen, Nordhausen, and Taskinen 2017), and **templateICAr** (Mejia et al. 2020). These R packages use different algorithms to extract non-Gaussian features but are designed for decomposing a single modality. For data integration, **r.jive** (O'Connell and Lock 2020) and **ajive** (Carmichael 2022; Feng et al. 2018) capture the joint variation, or variance shared between datasets, and individual variation, or variance unique to a dataset. JIVE methods use singular value decompositions, which are related to maximizing variance instead of non-Gaussianity. In this way, there exists a need for freely available software for extracting joint structure from multiple datasets using non-Gaussian measures of information.

This paper introduces **singR**, an R package to implement Risk and Gaynanova (2021). This paper is structured as follows. In Section 2, we review the Simultaneous non-Gaussian component analysis (SING) model. In Sections 3 and 4, we present the main functions in the **singR** package and show how to utilize it for joint components estimation in two example datasets. Finally, Section 5 summarizes our conclusions.

2 Methods

2.1 Linear non-Gaussian Component Analysis

Matrix decomposition for one dataset. Based on linear non-Gaussian component analysis (LNGCA), we first summarize a matrix decomposition for a single dataset $X \in \mathbb{R}^{n\times p_x}$ (n subjects and p_x features) into a non-Gaussian subspace and a Gaussian subspace. Each row of X is a vector of features from the ith subject. Let X_c denote the double-centered data matrix such that $1^T X_c = 0^T$ and $X_c 1 = 0$ where 1 denotes the vector of ones of appropriate dimension, which has rank $n-1$ when $p_x > n$. Let I_{r_x} denote the $r_x \times r_x$ identity matrix. Then define the matrix decomposition

$$X_c = M_x S_x + M_{N_x} N_x. \tag{1}$$

Here, $M_x \in \mathbb{R}^{n\times r_x}$, and the columns of M_x are called subject scores. $M_{N_x} \in \mathbb{R}^{n\times(n-r_x-1)}$, and the matrix $[M_x, M_{N_x}]$ is called the mixing matrix and has rank $n-1$. $S_x \in \mathbb{R}^{r\times p_x}$ and $N_x \in \mathbb{R}^{(n-r_x-1)\times p_x}$. $S_x S_x^T = p_x I_{r_x}$, $N_x S_x^T = 0_{(n-r_x-1)\times r_x}$. The rows of S_x are the non-Gaussian components, and elements of S_x are called variable loadings because $\frac{1}{p_x} X_c S_x^T = M_x$. The rows of N_x are the Gaussian components. The rows of S_x have the largest non-Gaussianity, as described below.

This decomposition may be meaningful in neuroimaging studies because: 1) vectorized components like brain activation maps and resting-state networks have highly non-Gaussian distributions, and 2) it is often the situation that $p_x \gg n$, i.e., the number subjects is smaller than the number of voxels or edges.

To achieve the matrix decomposition in (1), we need to find a separating matrix A_x that maximizes non-Gaussianity and satisfies the constraint $A_x X_c X_c^T A_x^T = S_x S_x^T = p_x I_{r_x}$. We utilize a prewhitening matrix and then reparameterize the model with a semiorthogonal separating matrix to enforce this constraint. Let $\widehat{\Sigma}_x = X_c X_c^\top / p_x$. Then define the eigenvalue decomposition $\widehat{\Sigma}_x = V_x \Lambda_x V_x^\top$ and the prewhitening matrix $\widehat{L}_x = V_x \Lambda_x^{-1/2} V_x^\top$ (for double-centered X_c, this is understood to be the square root of the generalized inverse from the non-zero eigenvalues). Note $A_x = U_x \widehat{L}_x$, and it follows that $\widehat{M}_x = \widehat{L}_x^- U_x^\top$, with $\widehat{L}_x^-$ denoting the generalized inverse. Let $f()$ be a measure of non-Gaussianity. Then (1) is estimated using

$$\begin{aligned} \underset{U_x}{\text{minimize}} \quad & -\sum_{l=1}^{r_x} f\left(u_{xl}^\top \widehat{L}_x X_c\right), \\ \text{subject to} \quad & U_x U_x^\top = I_{r_x}, \end{aligned} \tag{2}$$

where $u_{xl}^\top$ is the lth row of the $r_x \times n$ matrix U_x.

We measure non-Gaussianity with the **Jarque-Bera (JB) statistic**, a combination of squared skewness and kurtosis. For a vector $s \in \mathbb{R}^p$, the JB statistic is

$$f(s) = 0.8\left(\frac{1}{p}\sum_j s_j^3\right)^2 + 0.2\left(\frac{1}{p}\sum_j s_j^4 - 3\right)^2. \tag{3}$$

2.2 Simultaneous non-Gaussian component analysis model

Matrix decomposition for two datasets. We now decompose $X \in \mathbb{R}^{n\times p_x}$ and $Y \in \mathbb{R}^{n\times p_y}$ into a joint non-Gaussian subspace defined by shared subject score directions, individual non-Gaussian subspaces, and Gaussian subspaces. Let r_j denote the rank of the joint non-Gaussian subspace and r_x, r_y denote the rank of the non-Gaussian subspaces for X and Y, respectively. Define the double-centered X_c and Y_c, i.e., $1^\top X_c = 0$ and $X_c 1 = 0$. In data applications, we also recommend standardizing each feature to have unit variance, as is common in PCA. The double centering with standardization requires an iterative algorithm that standardizes each feature (mean 0 variance 1 across subjects), then centers the

features for a given subject (the mean of the features for a subject is 0), and repeats (typically < 10 iterations on real data suffices). The function `standard` is described in Section [3].

The SING matrix decomposition is

$$\begin{aligned} X_c &= M_J D_x S_{Jx} + M_{Ix} S_{Ix} + M_{Nx} N_x, \\ Y_c &= M_J D_y S_{Jy} + M_{Iy} S_{Iy} + M_{Ny} N_y. \end{aligned} \tag{4}$$

Here, $M_J \in \mathbb{R}^{n \times r_J}$, $M_{I_x} \in \mathbb{R}^{n \times (r_x - r_J)}$, $M_{I_y} \in \mathbb{R}^{n \times (r_y - r_J)}$, $M_{N_x} \in \mathbb{R}^{n \times (n - r_x - 1)}$, and $M_{N_y} \in \mathbb{R}^{n \times (n - r_y - 1)}$. D_x and D_y are diagonal and allow the scaling of M_J to vary between datasets. S_{Jx} are the joint non-Gaussian components, S_{Ix} are the individual non-Gaussian components, and N_x are the individual Gaussian components, respectively, for X, with constraints $S_{Jx} S_{Jx}^T = p_x I_{r_J}$, $S_{Ix} S_{Ix}^T = p_x I_{r_x - r_J}$, $S_{Jx} S_{Ix}^T = 0_{r_J \times (r_x - r_J)}$, $N_x S_{Jx}^T = 0_{(n - r_x - 1) \times r_J}$, $N_x S_{Ix}^T = 0_{(n - r_x - 1) \times (r_x - r_J)}$, and similarly define the components of Y.

Simultaneous non-Gaussian Component Analysis fitting and algorithm. Recall the whitening matrix for X_c is $\widehat{L}_x$, and define its generalized inverse $\widehat{L}_x^- = (X_c X_c^T / p_x)^{1/2} = V_x \Lambda_x^{1/2} V_x^\top$. We will estimate a semiorthogonal unmixing matrix $\widehat{U}_x$ such that $\widehat{M}_x = \widehat{L}_x^- \widehat{U}_x^T$. Similarly define the whitening matrix $\widehat{L}_y$ and $\widehat{M}_y$ for Y_c. Let f be the JB statistic, as defined in (3). We consider

$$\begin{aligned} \underset{U_x, U_y}{\text{minimize}} \quad & -\sum_{l=1}^{r_x} f(u_{xl}^\top \widehat{L}_x X_c) - \sum_{l=1}^{r_y} f(u_{yl}^\top \widehat{L}_y Y_c) + \rho \sum_{l=1}^{r_J} d(\widehat{L}_x^- u_{xl}, \widehat{L}_y^- u_{yl}) \\ \text{subject to} \quad & U_x U_x^\top = I_{r_x},\ U_y U_y^\top = I_{r_y}, \end{aligned} \tag{5}$$

where $d(x, y)$ is the chosen distance metric between vectors x and y, calculated using the chordal distance: $d(x, y) = \left\| \frac{xx^\top}{\|x\|_2^2} - \frac{yy^\top}{\|y\|_2^2} \right\|_F^2$. When columns are mean zero, the chordal distance between joint scores in the SING objective function is equal to zero when their correlation is equal to one. Larger values of the tuning parameter ρ result in common M_J, but smaller values result in highly correlated joint structure and could also be considered. In our examples, we match components from the separate LNGCA, determine candidate joint components from a permutation test, and then set ρ equal to the sum of the JB statistics of all candidate joint loadings divided by 10, which results in $\widehat{L}_x^- u_{xl} \approx \widehat{L}_y^- u_{yl}$, i.e., a shared M_J.

Let $\widehat{U}_x$ and $\widehat{U}_y$ be the estimated value of U_x and U_y in (5). The corresponding estimated non-Gaussian components are defined as $\widehat{S}_x = \widehat{U}_x X_w$ and $\widehat{S}_y = \widehat{U}_x Y_w$, where $X_w = \widehat{L}_x X_c$, and $Y_w = \widehat{L}_y Y_c$, respectively. Then the first r_J columns of $\widehat{M}_x = \widehat{L}_x^- \widehat{U}_x^\top$, scaled to unit norm, define $\widehat{M}_{Jx}$. We can similarly define $\widehat{M}_{Jy}$. For sufficiently large ρ, $\widehat{M}_{Jx} = \widehat{M}_{Jy} = \widehat{M}_J$, and more generally, $\widehat{M}_J$ is defined from their average. Additionally, the first r_J rows of $\widehat{S}_x$ correspond to $\widehat{S}_{Jx}$.

3 Overview of functions

The R package **singR** implements simultaneous non-Gaussian component analysis for data integration in neuroimaging. Highlighted below are key functions:

lngca: This function estimates non-Gaussian components for a single dataset. Non-Gaussian components can be estimated using the Jarque-Bera test statistic, which is the non-Gaussian measure used in SING, or using likelihood component analysis, which achieves dimension reduction while estimating the densities of non-Gaussian components (Risk, Matteson, and Ruppert 2019). It returns $\widehat{U}_x$ and $\widehat{S}_x$ from decomposing X_c through (2). It also returns the non-Gaussianity of each estimated component.

standard: This function is an iterative algorithm that standardizes each feature (mean 0 variance 1 across subjects), then centers the features for a given subject (the mean of the features for a subject is 0) for the original datasets ($\mathbb{R}^{n \times p}$), and repeats until the variance is approximately equal to 1 (typically < 10 iterations on real data suffices).

est.M.ols: This function returns $\widehat{M}_x$ with input $\widehat{S}_x$ and X_c.

greedymatch: This function reorders the columns in $\widehat{U}_x$ to match the columns (subject scores) in $\widehat{U}_y$ based on the chordal distances between corresponding $\widehat{M}_x$ and $\widehat{M}_y$.

permTestJointRank: This function tests whether the correlation between matched columns (subject scores) is significant and returns the family wise error rate corrected p-values.

%^%: Calculates the matrix exponential. For example, `A%^%0.5` returns a matrix square root. Used during prewhitening.

calculateJB: This function calculates the sum of the JB statistics across components and is useful for determining the size of the penalty parameter ρ (sufficiently large ρ results in the chordal distance between M_{Jx} and M_{Jy} equal to 0). Assumes the variance of each row of the input S is equal to 1 and mean of each row is equal to 0.

curvilinear: This function gives the final estimates of $\widehat{U}_x$ and $\widehat{U}_y$ using the curvilinear algorithm derived from (5). This is a pure R implementation but is slow.

curvilinear_c: This implements the curvilinear algorithm in C++, which is faster.

NG_number: This is a wrapper function for `FOBIasymp` from **ICtest** (Nordhausen et al. 2022) that can be used to estimate the number of non-Gaussian components in a single dataset.

signchange: This function makes the skewness of each row of $\widehat{S}_x$ positive, which is useful for visualizing non-Gaussian component loadings.

singR: This function integrates all the functions above. We can use this function to estimate joint scores and loadings from two datasets X and Y and optionally return the individual scores and loadings.

4 Examples

To illustrate the use of **singR**, we provide two examples.

4.1 Example 1. The toy datasets decomposition

The tutorial dataset `exampledata` are included in the **singR** package. We generate the SING model in (4) as follows. We generate joint subject scores $M_J = [m_{J1}, m_{J2}] \in \mathbb{R}^{n\times 2}$ with $m_{J1} \sim N(\mu_1, I_n)$, $m_{J2} \sim N(\mu_2, I_n)$, $\mu_1 = (1_{24}^\top, -1_{24}^\top)^\top$ and $\mu_2 = (-1_{24}^\top, 1_{24}^\top)^\top$. We set $D_x = I$ and $D_y = diag(-5, 2)$ to have differences in both sign and scale between the two datasets. We generate M_{Ix} and M_{Iy} similar to M_J using iid unit variance Gaussian entries with means equal to $\mu_{3y} = (-1_6^\top, 1_6^\top, -1_6^\top, 1_6^\top, -1_6^\top, 1_6^\top - 1_6^\top, -1_6^\top)^\top$, $\mu_{4y} = (1_{24}^\top, -1_{24}^\top)^\top$, $\mu_{3x} = (-1_{12}^\top, 1_{12}^\top, -1_{12}^\top, 1_{12}^\top)^\top$ and $\mu_{4x} = (1_{12}^\top, -1_{12}^\top, 1_{12}^\top, -1_{12}^\top)^\top$. These means result in various degrees of correlation between the columns of the mixing matrices. For the Gaussian noise, we generate M_{Nx}, M_{Ny}, N_x and N_y using iid standard Gaussian mean zero entries.

Each row of S_{Jx} and S_{Ix} is a vectorized image. We can reshape the loadings back to their image dimensions for visualization. The loadings S_{Jx} are inspired by activation patterns found in functional MRI, and similar simulations were considered in (Risk, Matteson, and Ruppert 2019). The rows of S_{Jy} and S_{Iy} are formed from the lower diagonal of a symmetric matrix, which are inspired by ICA of correlation matrices (Amico et al. 2017), and we can visualize the loadings by reshaping the vectors back to the symmetric matrix. The true loadings of latent non-Gaussian components are plotted in figure 1.

```
library(singR)
data(exampledata)
data <- exampledata

lgrid = 33
par(mfrow = c(2, 4))
# Components for X
image(matrix(data$sjX[1, ], lgrid, lgrid), col = heat.colors(12),
    xaxt = "n", yaxt = "n", main = expression("True S"["Jx"] * ",1"))
image(matrix(data$sjX[2, ], lgrid, lgrid), col = heat.colors(12),
    xaxt = "n", yaxt = "n", main = expression("True S"["Jx"] * ", 2"))
image(matrix(data$siX[1, ], lgrid, lgrid), col = heat.colors(12),
    xaxt = "n", yaxt = "n", main = expression("True S"["Ix"] * ", 1"))
image(matrix(data$siX[2, ], lgrid, lgrid), col = heat.colors(12),
    xaxt = "n", yaxt = "n", main = expression("True S"["Ix"] * ", 2"))

# Components for Y
image(vec2net(data$sjY[1, ]), col = heat.colors(12), xaxt = "n", yaxt = "n",
    main = expression("True S"["Jy"] * ", 1"))
image(vec2net(data$sjY[2, ]), col = heat.colors(12), xaxt = "n", yaxt = "n",
```

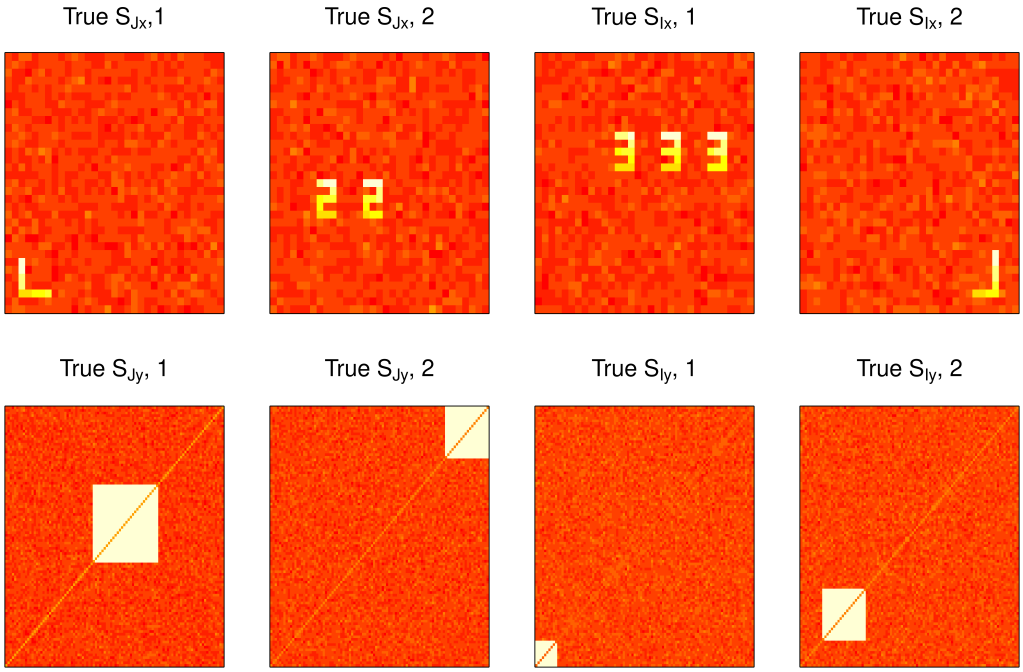

Figure 1: True joint and individual loadings in example 1.

```
    main = expression("True S"["Jy"] * ", 2"))
image(vec2net(data$siY[1, ]), col = heat.colors(12), xaxt = "n", yaxt = "n",
    main = expression("True S"["Iy"] * ", 1"))
image(vec2net(data$siY[2, ]), col = heat.colors(12), xaxt = "n", yaxt = "n",
    main = expression("True S"["Iy"] * ", 2"))
```

Function singR performs all steps in the SING pipeline as a single function

We first illustrate the use of the wrapper function `singR` using the default settings. We will describe optional arguments in more detail in example 2.

```
example1 = singR(dX = data$dX, dY = data$dY, individual = T)
```

Details of the SING pipeline

We next explain each of the steps involved in SING estimation. Using these individual functions in place of the high-level `singR` function allows additional fine-tuning and can be helpful for large datasets.

Estimate the number of non-Gaussian components in datasets dX and dY using `FOBIasymp` from ICtest:

```
n.comp.X = NG_number(data$dX)
n.comp.Y = NG_number(data$dY)
```

Apply `lngca` separately to each dataset using the JB statistic as the measure of non-Gaussianity:

```
# JB on X
estX_JB = lngca(xData = data$dX, n.comp = n.comp.X, whiten = "sqrtprec",
    restarts.pbyd = 20, distribution = "JB")
Uxfull <- estX_JB$U
Mx_JB = est.M.ols(sData = estX_JB$S, xData = data$dX)

# JB on Y
estY_JB = lngca(xData = data$dY, n.comp = n.comp.Y, whiten = "sqrtprec",
    restarts.pbyd = 20, distribution = "JB")
Uyfull <- estY_JB$U
My_JB = est.M.ols(sData = estY_JB$S, xData = data$dY)
```

Use `greedymatch` to reorder $\widehat{U}_x$ and $\widehat{U}_y$ by descending matched correlations and use `permTestJointRank` to estimate the number of joint components:

```
matchMxMy = greedymatch(scale(Mx_JB, scale = F), scale(My_JB, scale = F),
    Ux = Uxfull, Uy = Uyfull)
permJoint <- permTestJointRank(matchMxMy$Mx, matchMxMy$My)
joint_rank = permJoint$rj
```

For preparing input to `curvilinear_c`, manually prewhiten dX and dY to get $\widehat{L}_x^{-1}$ and $\widehat{L}_y^{-1}$:

```
# Center X and Y
dX = data$dX
dY = data$dY
n = nrow(dX)
pX = ncol(dX)
pY = ncol(dY)
dXcentered <- dX - matrix(rowMeans(dX), n, pX, byrow = F)
dYcentered <- dY - matrix(rowMeans(dY), n, pY, byrow = F)

# For X Scale rowwise
est.sigmaXA = tcrossprod(dXcentered)/(pX - 1)
whitenerXA = est.sigmaXA %^% (-0.5)
xDataA = whitenerXA %*% dXcentered
invLx = est.sigmaXA %^% (0.5)

# For Y Scale rowwise
est.sigmaYA = tcrossprod(dYcentered)/(pY - 1)
whitenerYA = est.sigmaYA %^% (-0.5)
yDataA = whitenerYA %*% dYcentered
invLy = est.sigmaYA %^% (0.5)
```

Obtain a reasonable value for the penalty ρ by calculating the JB statistics for all the joint components:

```
# Calculate the Sx and Sy.
Sx = matchMxMy$Ux[1:joint_rank, ] %*% xDataA
Sy = matchMxMy$Uy[1:joint_rank, ] %*% yDataA

# Calculate total JB
JBall = calculateJB(Sx) + calculateJB(Sy)

# Penalty used in curvilinear algorithm:
rho = JBall/10
```

Estimate $\widehat{U}_x$ and $\widehat{U}_y$ with `curvilinear_c`:

```
# alpha=0.8 corresponds to JB weighting of skewness and kurtosis
# (can customize to use different weighting):
alpha = 0.8
# tolerance:
tol = 1e-10

out <- curvilinear_c(invLx = invLx, invLy = invLy, xData = xDataA,
    yData = yDataA, Ux = matchMxMy$Ux, Uy = matchMxMy$Uy, rho = rho,
    tol = tol, alpha = alpha, maxiter = 1500, rj = joint_rank)
```

Obtain the final result:

```
# Estimate Sx and Sy and true S matrix using rotation matrices
# of Ux and Uy
Sjx = out$Ux[1:joint_rank, ] %*% xDataA
Six = out$Ux[(joint_rank + 1):n.comp.X, ] %*% xDataA
Sjy = out$Uy[1:joint_rank, ] %*% yDataA
```

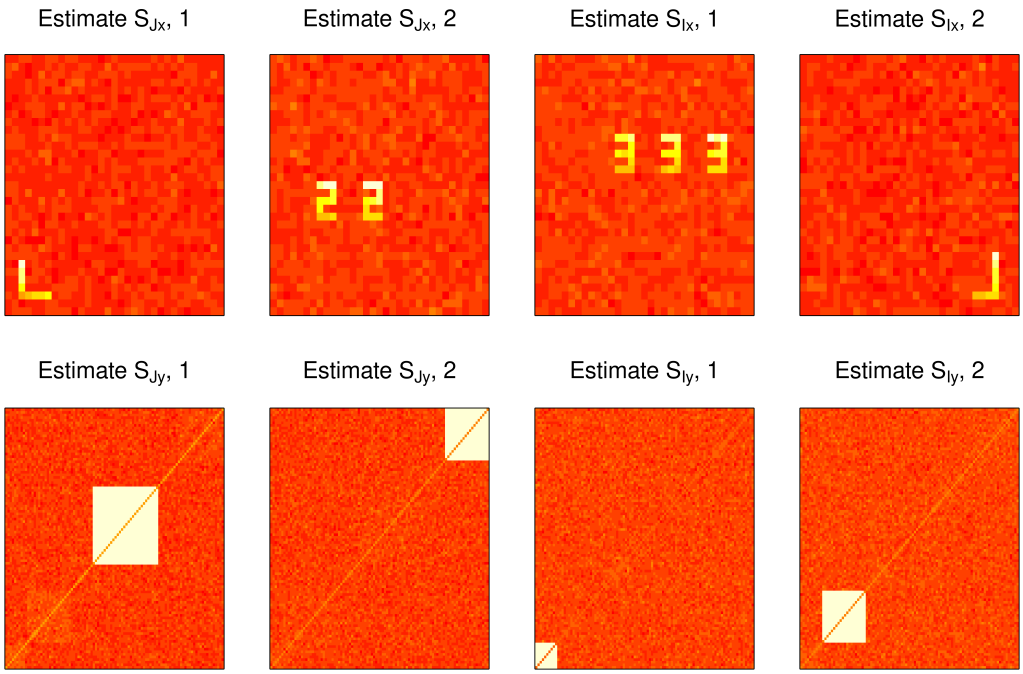

Figure 2: Estimated joint and individual loadings in example 1.

```
Siy = out$Uy[(joint_rank + 1):n.comp.Y, ] %*% yDataA

# Estimate Mj
Mxjoint = tcrossprod(invLx, out$Ux[1:joint_rank, ])
Mxindiv = tcrossprod(invLx, out$Ux[(joint_rank + 1):n.comp.X, ])
Myjoint = tcrossprod(invLy, out$Uy[1:joint_rank, ])
Myindiv = tcrossprod(invLy, out$Uy[(joint_rank + 1):n.comp.Y, ])

# signchange to make the skewness of the rows of S positive
Sjx_sign = signchange(Sjx, Mxjoint)
Sjy_sign = signchange(Sjy, Myjoint)
Six_sign = signchange(Six, Mxindiv)
Siy_sign = signchange(Siy, Myindiv)

Sjx = Sjx_sign$S
Sjy = Sjy_sign$S
Six = Six_sign$S
Siy = Siy_sign$S

Mxjoint = Sjx_sign$M
Myjoint = Sjy_sign$M
Mxindiv = Six_sign$M
Myindiv = Siy_sign$M

est.Mj = aveM(Mxjoint, Myjoint)

trueMj <- data.frame(mj1 = data$mj[, 1], mj2 = data$mj[, 2], number = 1:48)
SINGMj <- data.frame(mj1 = est.Mj[, 1], mj2 = est.Mj[, 2], number = 1:48)
```

Plot $\widehat{S}_{Jx}$, $\widehat{S}_{Jy}$, $\widehat{S}_{Ix}$, and $\widehat{S}_{Iy}$ in figure 2.

Plot $\widehat{M}_J$ in figure 3.

```
library(tidyverse)
library(ggpubr)

# true Mj
```

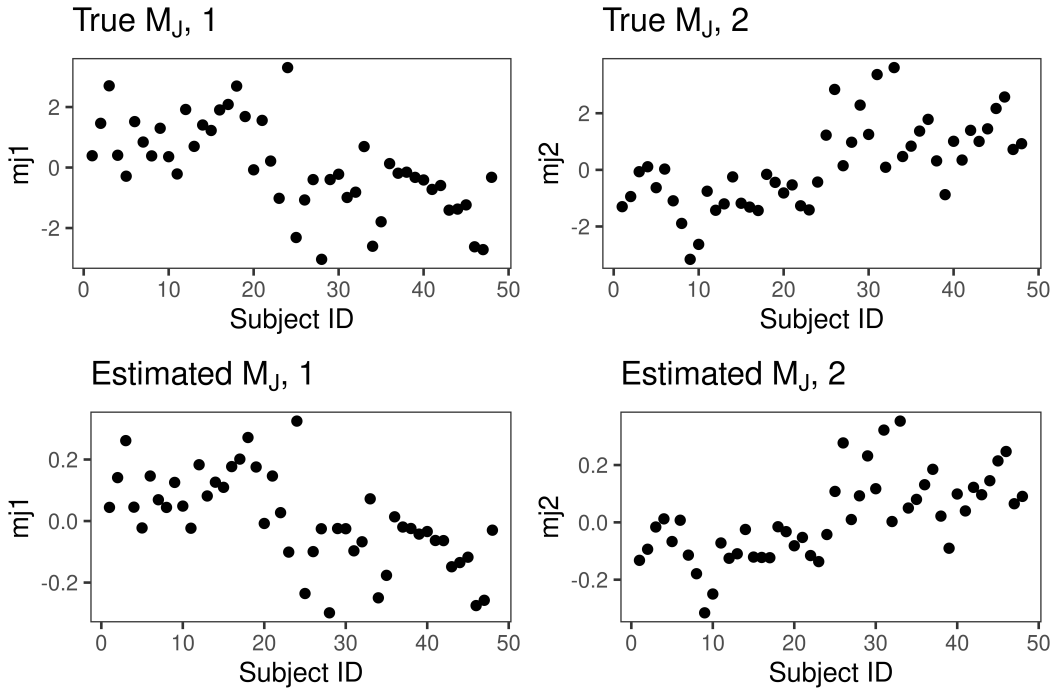

Figure 3: Estimated joint subject scores in example 1.

```
t1 <- ggplot(data = trueMj) + xlab("Subject ID") + geom_point(mapping = aes(y = mj1,
    x = number)) + ggtitle(expression("True M"["J"] * ", 1")) + theme_bw() +
    theme(panel.grid = element_blank())

t2 <- ggplot(data = trueMj) + xlab("Subject ID") + geom_point(mapping = aes(y = mj2,
    x = number)) + ggtitle(expression("True M"["J"] * ", 2")) + theme_bw() +
    theme(panel.grid = element_blank())

# SING estimated Mj
S1 <- ggplot(data = SINGMj) + xlab("Subject ID") + geom_point(mapping = aes(y = mj1,
    x = number)) + ggtitle(expression("Estimated M"["J"] * ", 1")) +
    theme_bw() + theme(panel.grid = element_blank())

S2 <- ggplot(data = SINGMj) + xlab("Subject ID") + geom_point(mapping = aes(y = mj2,
    x = number)) + ggtitle(expression("Estimated M"["J"] * ", 2")) +
    theme_bw() + theme(panel.grid = element_blank())

ggarrange(t1, t2, S1, S2, ncol = 2, nrow = 2)
```

4.2 Example 2. MRI data simulation

This example is a simulation inspired by the real data analysis of the Human Connectome Project from Risk and Gaynanova (2021). X are generated from $\widehat{S}_X$ from working memory task maps and Y are generated from $\widehat{S}_Y$ from resting-state correlations from a previous SING analysis of the Human Connectome Project. The working memory loadings are defined on the cortical surface, which is the highly folded ribbon of gray matter forming the outer layer of the brain containing billions of neural bodies and dendrites. Large working memory loadings indicate locations in the brain that tend to work together during memory tasks. The resting-state correlation loadings are defined using a brain parcellation from (Glasser et al. 2016) and (Akiki and Abdallah 2019). Large resting-state loadings are related to large correlations between brain regions occurring when a participant is lying in a scanner performing no task. Additional details are in (Risk and Gaynanova 2021). For the purposes of this example, we reduce computation time by lowering the resolution of the working memory loadings in dataset X from 60,000 to 2,000. For the resting-state correlation loadings, we subset from the 360 x 360 loadings matrices formed from the 360 regions in the multimodal parcellation (MMP) to 100 x

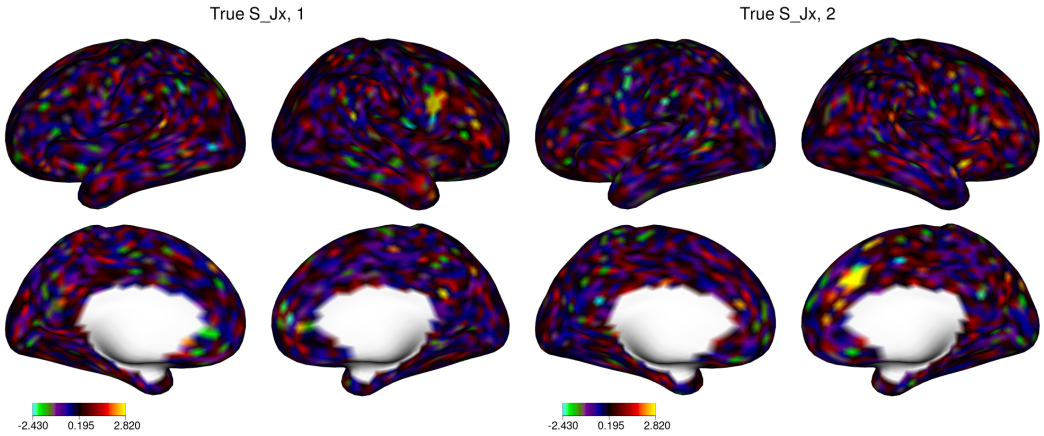

Figure 4: True joint loadings in dataset X in example 2.

100. To run this example, download the files in the folder `extdata` from the github repository (Wang, Gaynanova, and Risk 2022).

```
# Load the package
library(singR)

# Read and visualize data
load("extdata/simdata.rda")

# sign change makes the skewness positive, which makes the
# region of 'activation' yellow in the plots that follow
Sxtrue = signchange(simdata$sjx)$S
Sytrue = signchange(simdata$sjy)$S
```

The `simdata.rda` have already been resampled from 32k to 2k resolution to reduce computation time. Next, we resample the background surface (i.e., template found on (Wang, Gaynanova, and Risk 2022)) to the same resolution, which will allow us to plot the loadings on the cortical surface. This step uses **ciftiTools** (Pham, Muschelli, and Mejia 2021) and connectome workbench (Marcus et al. 2011). To run this code, one needs to install connectome workbench, as described in (https://github.com/mandymejia/ciftiTools).

```
library(ciftiTools)
ciftiTools.setOption("wb_path", "C:/Software/workbench")

## the template cifti file is on
## https://github.com/thebrisklab/singR/tree/main/extdata.
## here, resample to 2k resolution.
xii_template <- read_cifti("extdata/template.dtseries.nii", brainstructures = c("left",
    "right"), resamp_res = 2000)

xii_new <- newdata_xifti(xii_template, t(Sxtrue))
view_xifti_surface(select_xifti(xii_new, 1), zlim = c(-2.43, 2.82))  ## true S_JX1
view_xifti_surface(select_xifti(xii_new, 2), zlim = c(-2.43, 2.82))  ## true S_JX2
```

In figure 4, the yellow regions indicate locations with large loadings. Similar plots can be created for the two individual components (not shown). When applied to fMRI activation maps, SING tends to identify small patches of cortex, similar to this figure.

Next, we convert the rows of S_{Jy} to symmetric matrices and create plots. The nodes are organized into communities (i.e., modules) to aid visualization. SING tends to identify a single node and the connections with this node, which result in a cross-like pattern in the matrix representation. The joint loadings are plotted in figure 5 with `plotNetwork_change`, which is defined below. Similar plots can be created for the loadings from the two individual components.

```
# define plotNetwork_change
plotNetwork_change = function(component, title = "", qmin = 0.005,
```

```
    qmax = 0.995, path = "mmpplus.csv", make.diag = NA) {
    # component: vectorized network of length choose(n,2)
    require(ggplot2)
    require(grid)
    require(scales)

    # load communities for plotting:
    mmp_modules = read.csv(path, header = TRUE)
    mmp_order = order(mmp_modules$Community_Vector)

    zmin = quantile(component, qmin)
    zmax = quantile(component, qmax)

    netmat = vec2net(component, make.diag)

    meltsub = create.graph.long(netmat, mmp_order)

    g2 = ggplot(meltsub, aes(X1, X2, fill = value)) + geom_tile() +
        scale_fill_gradient2(low = "blue", high = "red", limits = c(zmin,
            zmax), oob = squish) + labs(title = title, x = "Node 1",
        y = "Node 2") + coord_cartesian(clip = "off", xlim = c(-0,
        100))

    loadingsummary = apply(abs(netmat), 1, sum, na.rm = TRUE)
    loadingsum2 = loadingsummary[mmp_order]

    Community = factor(mmp_modules$Community_Label)[mmp_order]

    g3 = qplot(c(1:100), loadingsum2, col = Community, size = I(3)) +
        xlab("MMP Index") + ylab("L1 Norm of the Rows")

    return(list(netmatfig = g2, loadingsfig = g3, netmat = netmat,
        loadingsummary = loadingsummary))
}

library(cowplot)
# plot for the true component of Y
path = "extdata/new_mmp.csv"
out_true1 = plotNetwork_change(Sytrue[1, ], title = expression("True S"["Jy"] *
    ", 1"), qmin = 0.005, qmax = 0.995, path = path)
out_true2 = plotNetwork_change(Sytrue[2, ], title = expression("True S"["Jy"] *
    ", 2"), qmin = 0.005, qmax = 0.995, path = path)

p1 = out_true1$netmatfig
p2 = out_true1$loadingsfig
p3 = out_true2$netmatfig
p4 = out_true2$loadingsfig

plot_grid(p1, p2, p3, p4, nrow = 2)
```

In figure 5, the left plots depict the loadings in the network space, where each element represents the strength of the connection between two nodes (i.e., regions). Then if the subject score corresponding to the component is large, the loadings make a large contribution to the subject's functional connectivity. In our previous work, we found that the loadings for a component tend to be structured such that a single node is prominent, resulting in a cross-like pattern. To easily identify which node or nodes are prominent, the right plots depict the L1-norms of the rows of the loadings matrices. In this example, a single node stands out for each of the components. For additional interpretation, see (Risk and Gaynanova 2021). In (Risk and Gaynanova 2021), it was discovered that for a given joint component, the patch of cortex with largest loadings in the working memory task tended to be located in the same area as the node whose loadings have the largest L1-norm in the resting-state dataset.

Function singR performs all steps in the SING pipeline as a single function

In example 1, we introduced the pipeline of the SING method. We will use example 2 to explain the `singR` function in detail. The default output of `singR` is a list of $\widehat{S}_{Jx}$, $\widehat{S}_{Jy}$, $\widehat{M}_J$, $\widehat{M}_{Jx}$, and $\widehat{M}_{Jy}$. By default, it will center the data such that the mean of each row is equal to zero. In our simulated dataset, all

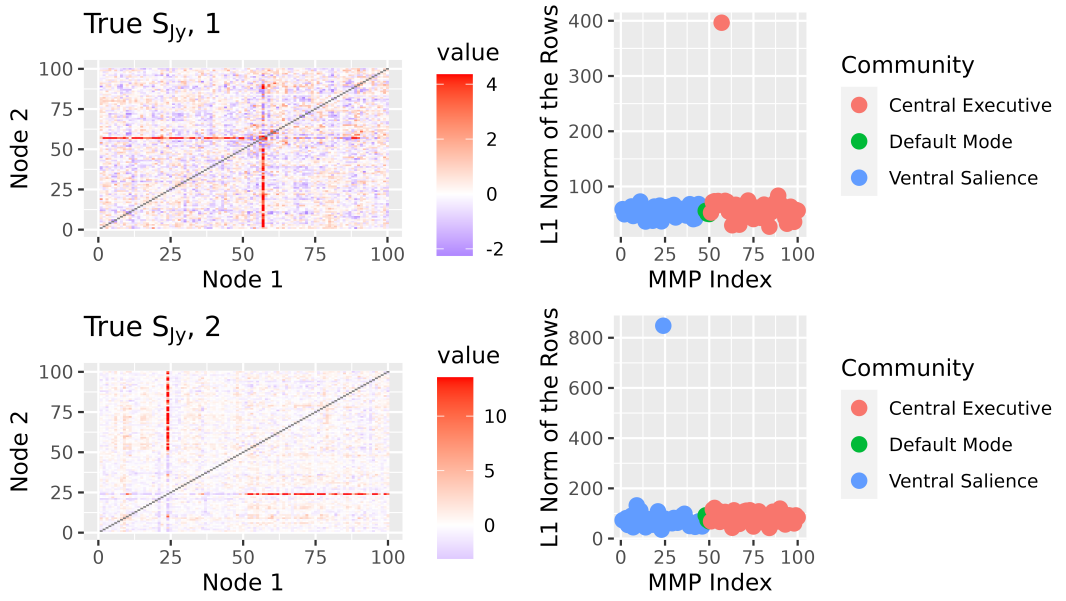

Figure 5: True joint loadings in dataset Y in example 2.

variables are on the same scale, and consequently we do not perform standardization (`stand=FALSE`). When `stand=TRUE`, the data are additionally standardized to have the mean of each column equal to zero and variance of each column equal to one, which is the standardization commonly used in PCA. If `n.comp.X` and `n.comp.Y` are not specified, `singR` will use `FOBIasymp` from **ICtest** to estimate the number of non-Gaussian components in each dataset, which requires additional computational expense. Other tests of the number of non-Gaussian components accounting for spatial smoothing/autocorrelation can be found in (Zhao et al. 2022). These may be more effective for spatially correlated data but are generally slower.

When `individual = TRUE`, the `singR` will additionally output $\widehat{M}_{Ix}$, $\widehat{M}_{Iy}$, $\widehat{S}_{Ix}$ and $\widehat{S}_{Iy}$. When `distribution = "tiltedgaussian"`, non-Gaussian components will be estimated through `lngca` using likelihood component analysis, which is slower but can be more accurate. By default `distribution = "JB"`, and `lngca` will use the Jarque-Bera test statistic as the measure of non-Gaussianity of each component.

The `Cplus` argument determines whether to use `curvilinear_c` or `curvilinear` in `singR`. `curvilinear` is implemented with pure R but is slow while `curvilinear_c` uses C++. The parameter `rho_extent` can be one of `c("small", "medium", "large")` or a number. This determines the penalty ρ in `curvilinear` or `curvilinear_c` that results in equal or highly correlated $\widehat{M}_{Jx}$ and $\widehat{M}_{Jy}$. Additionally, we can use `pmse()` to evaluate the distance between two subject score matrices. With larger ρ, the $pmse\left(\widehat{M}_{Jx}, \widehat{M}_{Jy}\right)$ value will be smaller. Usually, "small" ρ is sufficient for approximately equal $\widehat{M}_{Jx}$ and $\widehat{M}_{Jy}$. We have observed that very large ρ can adversely impact the accuracy of the loadings. Our recommendation is to use "small" ρ and check if it results in equal scores, and if not, then try other settings. The code below took approximately 20 seconds to run on a 2.8 GHz processor.

```
example2 = singR(dX = simdata$dX, dY = simdata$dY, rho_extent = "small",
    Cplus = TRUE, stand = FALSE, individual = TRUE, distribution = "JB")
```

The joint loadings $\widehat{S}_{JX}$ are depicted in figure 6.

```
xii_new <- newdata_xifti(xii_template, t(example2$Sjx))

view_xifti_surface(select_xifti(xii_new, 1), zlim = c(-2.43, 2.82)) ## component1 small rho
view_xifti_surface(select_xifti(xii_new, 2), zlim = c(-2.43, 2.82)) ## component2 small rho
```

The joint loadings $\widehat{S}_{Jy}$ are depicted in figure 7.

```
library(cowplot)
path = "extdata/new_mmp.csv"
out_rhoSmall1 = plotNetwork_change(example2$Sjy[1, ], title = expression("Estimate S"["Jy"] *
```

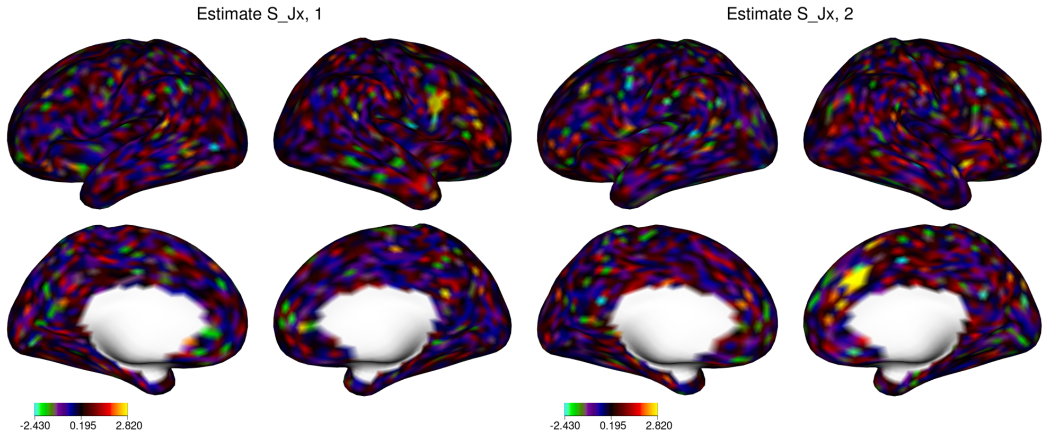

Figure 6: Estimated joint loadings in dataset X in example 2.

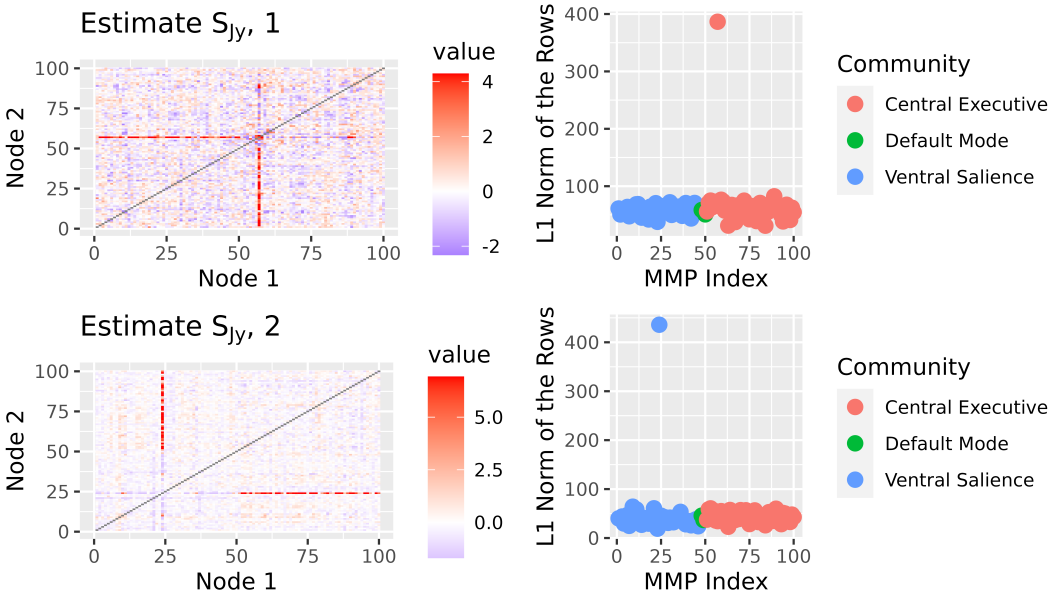

Figure 7: Estimated joint loadings in dataset Y in example 2.

```
    ", 1"), qmin = 0.005, qmax = 0.995, path = path)
out_rhoSmall2 = plotNetwork_change(example2$Sjy[2, ], title = expression("Estimate S"["Jy"] *
    ", 2"), qmin = 0.005, qmax = 0.995, path = path)

p5 = out_rhoSmall1$netmatfig
p6 = out_rhoSmall1$loadingsfig
p7 = out_rhoSmall2$netmatfig
p8 = out_rhoSmall2$loadingsfig

plot_grid(p5, p6, p7, p8, nrow = 2)
```

5 Summary

This paper introduces the **singR** package and demonstrates how simultaneous non-Gaussian component analysis can be used to extract shared features from two datasets using R. The main contribution of the R package **singR** is to provide easy code for data integration in neuroscience. We introduce the function `singR`, which combines the SING pipeline into one function that performs data standardization, estimates the number of non-Gaussian components and estimates the number of joint components. Previous analyses indicate the joint structure estimated by SING can improve upon other neuroimaging data integration methods. SING can reveal new insights by using non-Gaussianity for

both dimension reduction and latent variable extraction, whereas ICA methods involve an initial PCA step that tends to aggregate features and can remove information.

6 Acknowledgments

Research reported in this publication was supported by the National Institute of Mental Health of the National Institutes of Health under award number R01MH129855 to BBR. The research was also supported by the Division of Mathematical Sciences of the National Science Foundation under award number DMS-2044823 to IG. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health and National Science Foundation.

Simulated data were based on a previous analysis of data from the Human Connectome Project. These data were provided [in part] by the Human Connectome Project, WU-Minn Consortium (Principal Investigators: David Van Essen and Kamil Ugurbil; 1U54MH091657) funded by the 16 NIH Institutes and Centers that support the NIH Blueprint for Neuroscience Research; and by the McDonnell Center for Systems Neuroscience at Washington University.

References

Akiki, Teddy J, and Chadi G Abdallah. 2019. "Determining the Hierarchical Architecture of the Human Brain Using Subject-Level Clustering of Functional Networks." *Scientific Reports* 9 (1): 1–15. https://doi.org/10.1038/s41598-019-55738-y.

Amico, Enrico, Daniele Marinazzo, Carol Di Perri, Lizette Heine, Jitka Annen, Charlotte Martial, Mario Dzemidzic, et al. 2017. "Mapping the Functional Connectome Traits of Levels of Consciousness." *NeuroImage* 148: 201–11. https://doi.org/https://doi.org/10.1016/j.neuroimage.2017.01.020.

Beckmann, Christian F, Marilena DeLuca, Joseph T Devlin, and Stephen M Smith. 2005. "Investigations into Resting-State Connectivity Using Independent Component Analysis." *Philosophical Transactions of the Royal Society B: Biological Sciences* 360 (1457): 1001–13. https://doi.org/10.1098/rstb.2005.1634.

Calhoun, V. D., T. Adali, N. R. Giuliani, J. J. Pekar, K. A. Kiehl, and G. D. Pearlson. 2006. "Method for Multimodal Analysis of Independent Source Differences in Schizophrenia: Combining Gray Matter Structural and Auditory Oddball Functional Data." Journal Article. *Hum Brain Mapp* 27 (1): 47–62. https://doi.org/10.1002/hbm.20166.

Calhoun, V. D., T. Adali, K. A. Kiehl, R. Astur, J. J. Pekar, and G. D. Pearlson. 2006. "A Method for Multitask fMRI Data Fusion Applied to Schizophrenia." Journal Article. *Hum Brain Mapp* 27 (7): 598–610. https://doi.org/10.1002/hbm.20204.

Calhoun, V. D., J. Liu, and T. Adali. 2009. "A Review of Group ICA for fMRI Data and ICA for Joint Inference of Imaging, Genetic, and ERP Data." Journal Article. *Neuroimage* 45 (1 Suppl): S163–72. https://doi.org/10.1016/j.neuroimage.2008.10.057.

Calhoun, V. D., and J. Sui. 2016. "Multimodal Fusion of Brain Imaging Data: A Key to Finding the Missing Link(s) in Complex Mental Illness." Journal Article. *Biol Psychiatry Cogn Neurosci Neuroimaging* 1 (3): 230–44. https://doi.org/10.1016/j.bpsc.2015.12.005.

Carmichael, Iain. 2022. *Ajive: Angle Based Joint and Individual Variation Explained*. https://cran.r-project.org/web/packages/RaJIVE/index.html.

Feng, Qing, Meilei Jiang, Jan Hannig, and JS Marron. 2018. "Angle-Based Joint and Individual Variation Explained." *Journal of Multivariate Analysis* 166: 241–65. https://doi.org/10.1016/j.jmva.2018.03.008.

Glasser, Matthew F, Timothy S Coalson, Emma C Robinson, Carl D Hacker, John Harwell, Essa Yacoub, Kamil Ugurbil, et al. 2016. "A Multi-Modal Parcellation of Human Cerebral Cortex." *Nature* 536 (7615): 171–78. https://doi.org/10.1038/nature18933.

Helwig, Nathaniel E. 2018. *Ica: Independent Component Analysis*. https://CRAN.R-project.org/package=ica.

Marchini, J L, C Heaton, and B D Ripley. 2021. *fastICA: FastICA Algorithms to Perform ICA and Projection Pursuit*. https://CRAN.R-project.org/package=fastICA.

Marcus, Daniel S, John Harwell, Timothy Olsen, Michael Hodge, Matthew F Glasser, Fred Prior, Mark Jenkinson, Timothy Laumann, Sandra W Curtiss, and David C Van Essen. 2011. "Informatics and Data Mining Tools and Strategies for the Human Connectome Project." *Frontiers in Neuroinformatics* 5: 4. https://doi.org/10.3389/fninf.2011.00004/full.

Mejia, Amanda F, Mary Beth Nebel, Tikai Wang, Brian S Caffo, and Ying Guo. 2020. "templateICAr." https://doi.org/10.1080/01621459.2019.1679638.

Miettinen, Jari, Klaus Nordhausen, and Sara Taskinen. 2017. "Blind Source Separation Based on Joint Diagonalization in R: The Packages JADE and BSSasymp." *Journal of Statistical Software* 76 (2): 1–31. https://doi.org/10.18637/jss.v076.i02.

Nordhausen, Klaus, Hannu Oja, David E. Tyler, and Joni Virta. 2022. *ICtest: Estimating and Testing the Number of Interesting Components in Linear Dimension Reduction*. https://CRAN.R-project.org/package=ICtest.

O'Connell, Michael J., and Eric F. Lock. 2020. *R.jive: Perform JIVE Decomposition for Multi-Source Data*. https://CRAN.R-project.org/package=r.jive.

Pham, Damon D, John Muschelli, and Amanda F Mejia. 2021. "ciftiTools: A Package for Reading, Writing, Visualizing and Manipulating CIFTI Files in r." https://arxiv.org/abs/2106.11338.

Risk, Benjamin B., and Irina Gaynanova. 2021. "Simultaneous Non-Gaussian Component Analysis (SING) for Data Integration in Neuroimaging." Journal Article. *The Annals of Applied Statistics* 15 (3): 1431–54, 24. https://doi.org/10.1214/21-AOAS1466.

Risk, Benjamin B., Nicholas A. James, and David S. Matteson. 2015. *steadyICA: ICA and Tests of Independence via Multivariate Distance Covariance*. https://CRAN.R-project.org/package=steadyICA.

Risk, Benjamin B., David S. Matteson, and David Ruppert. 2019. "Linear Non-Gaussian Component Analysis via Maximum Likelihood." *Journal of the American Statistical Association* 114 (525): 332–43. https://doi.org/10.1080/01621459.2017.1407772.

Sui, J., T. Adali, Q. Yu, J. Chen, and V. D. Calhoun. 2012. "A Review of Multivariate Methods for Multimodal Fusion of Brain Imaging Data." Journal Article. *J Neurosci Methods* 204 (1): 68–81. https://doi.org/10.1016/j.jneumeth.2011.10.031.

Sui, Jing, Tülay Adali, Godfrey Pearlson, Honghui Yang, Scott R. Sponheim, Tonya White, and Vince D. Calhoun. 2010. "A CCA+ICA Based Model for Multi-Task Brain Imaging Data Fusion and Its Application to Schizophrenia." *NeuroImage* 51 (1): 123–34. https://doi.org/10.1016/j.neuroimage.2010.01.069.

Sui, J., G. Pearlson, A. Caprihan, T. Adali, K. A. Kiehl, J. Liu, J. Yamamoto, and V. D. Calhoun. 2011. "Discriminating Schizophrenia and Bipolar Disorder by Fusing fMRI and DTI in a Multimodal CCA+ Joint ICA Model." Journal Article. *Neuroimage* 57 (3): 839–55. https://doi.org/10.1016/j.neuroimage.2011.05.055.

Vergara, V. M., A. Ulloa, V. D. Calhoun, D. Boutte, J. Chen, and J. Liu. 2014. "A Three-Way Parallel ICA Approach to Analyze Links Among Genetics, Brain Structure and Brain Function." Journal Article. *Neuroimage* 98: 386–94. https://doi.org/10.1016/j.neuroimage.2014.04.060.

Wang, Liangkang, Irina Gaynanova, and Benjamin B. Risk. 2022. "singR." https://github.com/thebrisklab/singR.

Zhao, Yuxuan, David S Matteson, Stewart H Mostofsky, Mary Beth Nebel, and Benjamin B Risk. 2022. "Group Linear Non-Gaussian Component Analysis with Applications to Neuroimaging." *Computational Statistics & Data Analysis* 171: 107454. https://doi.org/10.1016/j.csda.2022.107454.

Zhou, Guoxu, Qibin Zhao, Yu Zhang, Tülay Adalı, Shengli Xie, and Andrzej Cichocki. 2016. "Linked Component Analysis from Matrices to High-Order Tensors: Applications to Biomedical Data." *Proceedings of the IEEE* 104 (2): 310–31. https://doi.org/10.1109/JPROC.2015.2474704.

Liangkang Wang
Brown University
Department of Biostatistics
Providence, Rhode Island, US
ORCiD: 0000-0003-3393-243X
liangkang_wang@brown.edu

Irina Gaynanova
University of Michigan
Department of Biostatistics
Ann Arbor, MI, US
https://irinagain.github.io/
ORCiD: 0000-0002-4116-0268
irinagn@umich.edu

Benjamin Risk
Emory University
Department of Biostatistics and Bioinformatics
Atlanta, Georgia, US
https://github.com/thebrisklab/
ORCiD: 0000-0003-1090-0777

benjamin.risk@emory.edu