Alpha-centrality in the context of stream DAGs

Ken Aho

May 30, 2026

Contents

1 Introduction

This document tries to provide a background concerning the workings of the Katz centrality (Katz, 1953) and α-centrality (Bonacich & Lloyd, 2001) which are essentially identical, and provide the same rank order of centrality measures. Particular emphasis is given to directed acyclic graph representations of stream networks (stream DAGs). The content is intended for a general audience and so background concerning matrix terminology (Section 1.1), matrix operations (Section 1.2) and graph theory terminology (Section 1.3) is provided. The geometric infinite series, vital for understanding the translation of alpha-centrality to a linear algebra format, is summarized in Section 1.4.

1.1 Matrix terminology

Scalar:

An entity that can be represented by a single number. Scalars are generally denoted with lower-case italicized letters. For instance, $x = 3$.

Matrix:

A rectangular array whose elements are arranged into a table with $i = 1, 2, 3, \ldots, m$ rows and $j = 1, 2, 3 \ldots, n$ columns. Matrices are generally denoted with capitalized bold letters. The matrix $\boldsymbol{A}$ below has $m = 3$ rows and $n = 3$ columns. That is, $\boldsymbol{A}$ has **dimension** (3×3).

$$\underset{(3\times3)}{\boldsymbol{A}} = \begin{bmatrix} 4 & 3 & 2 \\ 1 & -3 & 3 \\ 7 & -8 & 2 \end{bmatrix}$$

The term a_{ij} represents the element at the intersection of the ith row and the jth column. For example, $a_{1,2} = -3$ in $\boldsymbol{A}$.

Column vector:

A matrix made up a single column. Vectors (representing a column or row) are generally denoted with a lower-case bold letter (or number). For example, $\boldsymbol{a}$ is a column vector, and $\mathbf{1}$ is a column vector of ones.

$$\underset{(3\times1)}{\boldsymbol{a}} = \begin{bmatrix} 3 \\ 7 \\ 6 \end{bmatrix}, \qquad \underset{(2\times1)}{\mathbf{1}} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Row vector:

A matrix made up a single row. For example, $\boldsymbol{b}$ is a row vector.

$$\underset{(1\times3)}{\boldsymbol{b}} = \begin{bmatrix} 3 & 7 & 6 \end{bmatrix}$$

Square matrix:

A matrix with the same number of rows and columns. Thus, the matrix $\boldsymbol{A}$ above is a square matrix.

Major diagonal:

Often simply called the **diagonal**, the entity comprises the a_{ii} elements of a square matrix. For instance, the diagonal of $\boldsymbol{A}$ above is $\{4, -3, 2\}$.

Triangular matrix:

The tabular triangle of data *below* the diagonal is the **lower triangle**. The tabular triangle *above* the diagonal is the **upper triangle**. The matrices $\boldsymbol{L}$ and $\boldsymbol{U}$ below are are the lower and upper triangle matrices of $\boldsymbol{A}$, respectively.

$$\underset{(3\times3)}{\boldsymbol{L}} = \begin{bmatrix} 4 & 0 & 0 \\ 1 & -3 & 0 \\ 7 & -8 & 2 \end{bmatrix}, \qquad \underset{(3\times3)}{\boldsymbol{U}} = \begin{bmatrix} 4 & 1 & 7 \\ 0 & -3 & -8 \\ 0 & 0 & 2 \end{bmatrix}$$

Note that the diagonal is included in both triangular matrices.

Diagonal matrix:

A square matrix with only zeroes on off-diagonal elements. The matrix $\boldsymbol{B}$ is a diagonal matrix.

$$\underset{(3\times3)}{\boldsymbol{B}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

Identity matrix:

A diagonal matrix with ones on the diagonal. The identity matrix is generally denoted $\boldsymbol{I}$:

$$\underset{(3\times3)}{\boldsymbol{I}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The identity matrix is often denoted $\boldsymbol{I}_n$, indicating it has dimensions $n \times n$. Thus, $\boldsymbol{I}_2$ is the 2×2 identity matrix.

- If $\boldsymbol{A}$ is an $n \times n$ matrix, $\boldsymbol{I}_m \cdot \boldsymbol{A} = \boldsymbol{A} \cdot \boldsymbol{I}_n = \boldsymbol{A}$. Thus, $\boldsymbol{I}$ serves as the matrix equivalent of a scalar one.
- $I_n \cdot \boldsymbol{I}_n = \boldsymbol{I}_n$.
- All rows and columns in $\boldsymbol{I}_n$ are linearly independent.

1.2 Matrix operations

Addition and subtraction:

The sum or difference of two matrices with the same dimensions is calculated element-wise. That is, if

$$\underset{(3\times3)}{\boldsymbol{A}} = \begin{bmatrix} 4 & 3 & 2 \\ 1 & -3 & 3 \\ 7 & -8 & 2 \end{bmatrix} \quad \text{and} \quad \underset{(3\times3)}{\boldsymbol{B}} = \begin{bmatrix} 0 & 1 & 2 \\ 2 & 4 & 6 \\ -1 & 2 & -2 \end{bmatrix}$$

$$\underset{(3\times3)}{\boldsymbol{A} - \boldsymbol{B}} = \begin{bmatrix} 4 & 2 & 0 \\ -1 & -7 & -3 \\ -8 & -10 & 4 \end{bmatrix}$$

To add or subtract a scalar to or from a matrix, one simply performs the arithmetic with every element in the matrix. For instance, if $x = 2$, then:

$$\underset{(3\times3)}{x - \boldsymbol{B}} = \begin{bmatrix} 2 & 1 & 0 \\ 0 & -2 & -4 \\ 3 & 0 & 4 \end{bmatrix}$$

Transpose:

The transpose of an $m \times n$ matrix $\boldsymbol{A}$ is the $n \times m$ matrix $\boldsymbol{A^T}$ which is results form converting rows into columns, and vice versa. That is, if

$$\underset{(3\times3)}{\boldsymbol{A}} = \begin{bmatrix} 4 & 3 & 2 \\ 1 & -3 & 3 \\ 7 & -8 & 2 \end{bmatrix} \quad \text{then} \quad \underset{(3\times3)}{\boldsymbol{A^T}} = \begin{bmatrix} 4 & 1 & 7 \\ 3 & -3 & -8 \\ 3 & 3 & 2 \end{bmatrix}$$

The transpose function in **R** is `t()`:

```
A <- matrix(nrow = 3, ncol = 3, data = c(4,3,2,1,-3,3,7,-8,2),
            byrow = T)
A

     [,1] [,2] [,3]
[1,]    4    3    2
[2,]    1   -3    3
[3,]    7   -8    2

t(A)

     [,1] [,2] [,3]
[1,]    4    1    7
[2,]    3   -3   -8
[3,]    2    3    2
```

Symmetric matrix:

A square matrix in which $\boldsymbol{A} = \boldsymbol{A^T}$, because its upper and lower triangles are "reflections" of each other. The matrix $\boldsymbol{A}$ below is symmetric.

$$\underset{(3\times3)}{\boldsymbol{A}} = \begin{bmatrix} 4 & 1 & 7 \\ 1 & -3 & -8 \\ 7 & -8 & 2 \end{bmatrix}$$

If $\boldsymbol{A} = -\boldsymbol{A^T}$, then $\boldsymbol{A}$ is a **skew-symmetric** matrix.

Multiplication:

- Multiplication of two matrices is defined if the number of columns of the multiplicand (left matrix) is equal to the number of rows in multiplier (right matrix). Thus, while $\underset{(2\times3)}{\boldsymbol{A}} \cdot \underset{(3\times1)}{\boldsymbol{B}}$ exists, $\underset{(3\times1)}{\boldsymbol{B}} \cdot \underset{(2\times3)}{\boldsymbol{A}}$ does not exist. That is (unlike scalar multiplication), matrix multiplication is not commutative.

- If $\boldsymbol{A}$ is an $(m \times n)$ matrix and $\boldsymbol{B}$ is an $(n \times p)$ matrix, then $\boldsymbol{A} \cdot \boldsymbol{B}$ is the $(m \times p)$ matrix[1] whose entries are given by the **dot product** (sum of element-wise vector products) of the corresponding row of $\boldsymbol{A}$ and the corresponding column of $\boldsymbol{B}$.

  For instance, if

$$\underset{(2\times2)}{\boldsymbol{A}} = \begin{bmatrix} a & c \\ b & d \end{bmatrix}, \quad \text{and} \quad \underset{(2\times2)}{\boldsymbol{B}} = \begin{bmatrix} e & g \\ f & h \end{bmatrix}, \quad \text{then} \quad \underset{(2\times2)}{\boldsymbol{A}\cdot\boldsymbol{B}} = \begin{bmatrix} (ae+cf) & (ag+ch) \\ (be+df) & (bg+dh) \end{bmatrix}.$$

[1] That is, the number of columns in $\boldsymbol{A} \cdot \boldsymbol{B}$ will equal the number of columns in $\boldsymbol{B}$, and the number of rows in $\boldsymbol{A} \cdot \boldsymbol{B}$ will be the number of rows in $\boldsymbol{A}$.

As a numerical example, if

$$\underset{(2\times 2)}{\boldsymbol{A}} = \begin{bmatrix} 3 & 1 \\ 2 & 3 \end{bmatrix}, \quad \text{and} \quad \underset{(2\times 1)}{\mathbf{1}} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \text{then} \quad \underset{(2\times 1)}{\boldsymbol{A}\cdot\mathbf{1}} = \begin{bmatrix} (3\cdot 1 + 1\cdot 1) \\ (2\cdot 1 + 3\cdot 1) \end{bmatrix} = \begin{bmatrix} 4 \\ 5 \end{bmatrix}.$$

Thus, multiplying an $m \times n$ matrix by an $n \times 1$ column vector of ones, results in an $n \times 1$ column vector of row sums from the first matrix.

To multiply a scalar by a matrix, or multiply a matrix by a scalar, one simply performs the arithmetic to every element in the matrix. That is, *this* operation is commutative. For instance, if $x = 2$ then:

$$\underset{(2\times 2)}{x\cdot\boldsymbol{A}} = \boldsymbol{A}\cdot x = \begin{bmatrix} 3\cdot 2 & 1\cdot 2 \\ 2\cdot 2 & 3\cdot 2 \end{bmatrix} = \begin{bmatrix} 6 & 2 \\ 4 & 6 \end{bmatrix}.$$

The matrix multiply operator in **R** is `%*%`.

```
A <- matrix(2, 3, data = c(1, 2, 3, 4, 5, 6))
A

     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

B <- matrix(3, 2, data = c(2, 1, -2, 0, 1, 9))
B

     [,1] [,2]
[1,]    2    0
[2,]    1    1
[3,]   -2    9

AB <- A%*%B
AB

     [,1] [,2]
[1,]   -5   48
[2,]   -4   58
```

Trace:

The trace is simply the sum of the diagonal of a square matrix. For example, if

$$\underset{(2\times 2)}{\boldsymbol{A}} = \begin{bmatrix} 3 & 1 \\ 2 & 3 \end{bmatrix}, \quad \text{then} \quad tr(\boldsymbol{A}) = 6$$

One can obtain the trace in **R** using `sum(diag(A))` where `A` is a square matrix.

```r
A <- matrix(nrow = 3, ncol = 3, data = c(4,3,2,1,-3,3,7,-8,2),
            byrow = T)
A
```

```
     [,1] [,2] [,3]
[1,]    4    3    2
[2,]    1   -3    3
[3,]    7   -8    2
```

```r
trace <- sum(diag(A))
trace
```

```
[1] 3
```

Determinant:

A scalar-valued summary function of a square matrix that allows calculation of the inverse of a matrix (see below) and determines if a matrix is invertible (see below).

The determinant of a matrix $\boldsymbol{A}$ is often denoted $\det(\boldsymbol{A})$ or $|\boldsymbol{A}|$. The determinant of a matrix larger than 3×3 is difficult to compute. Methods like the [Laplace expansion] allow fairly intuitive procedures for calculating the determinant of larger square matrices. The determinant of a 2×2 matrix $\boldsymbol{A}$ is

$$\det(\boldsymbol{A}) = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

In the expression above, if $(0, 0)$, (a, c), (b, d), $(a + b, c + d)$ represent the vertices of a parallelogram, then the absolute value of the determinant gives the area of the parallelogram. Thus, the (absolute value) of a determinant can be interpreted as an area/volume summary of a matrix.

Additional insights are possible if we consider a matrix as the result of a linear transformation. For example,

$$\underset{(2\times2)}{\boldsymbol{A}} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = \; 2 \cdot \boldsymbol{I}$$

Multiplying the unit square represented by $\boldsymbol{I}$, by 2, to obtain $\boldsymbol{A}$, increases its sides two-fold and increases its area four-fold. We also note that $\det(\boldsymbol{A}) = 2 \cdot 2 - 0 = 4$. Thus, $\det(\boldsymbol{A})$ represents the scale factor by which areas are transformed by the "linear map" represented by $\boldsymbol{A}$.

Importantly,

- $\det(\boldsymbol{I}) = 1$.
- Multiplying any row in a matrix by a number multiplies the determinant by that number.
- $\det(\boldsymbol{A} \cdot \boldsymbol{B}) = \det(\boldsymbol{A}) \cdot \det(\boldsymbol{B})$.

- $\det(\boldsymbol{A}) = \det(\boldsymbol{A}^T)$.
- In an upper or lower triangular matrix of any size $n \times n$, the determinant is the product of the diagonal.
- For any matrix $\boldsymbol{A}$) with two equal rows or columns $\det(\boldsymbol{A}) = 0$, indicating the vectors of $\boldsymbol{A}$ are linearly dependent, and that the matrix is not invertible (see below).

The determinant of a matrix can be calculated in **R** using the function `det()`.

```
A <- matrix(2,2, data = c(2,0,0,2), byrow = T)
A

     [,1] [,2]
[1,]    2    0
[2,]    0    2

det(A)

[1] 4
```

Adjugate matrix:

A manipulation of a square matrix that allows calculation of the inverse of matrix (see below). To obtain the adjugate of a square matrix $\boldsymbol{A}$, $adj(\boldsymbol{A})$, we: 1) obtain the **matrix minors** by finding the determinants of 2×2 sub-matrices of $\boldsymbol{A}$, and 2) find the matrix of **cofactors** by changing the sign of adjacent cells in the matrix of minors, and 3) transposing the matrix of cofactors. For a 2×2 matrix:

$$\boldsymbol{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$adj(\boldsymbol{A}) = \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

The adjugate of a matrix (larger than 2×2) can be calculated in **R** using the function `RConics::adjoint()`.

```
library(RConics)
A <- matrix(3,3, data = c(5,7,6,1,5,4,1,3,2), byrow = T)
A
adjoint(A)
```

Inverse:

An $n \times n$ square matrix $\boldsymbol{A}$ is **invertible** if a $n \times n$ matrix $\boldsymbol{B}$ exists that allows: $\boldsymbol{A} \cdot \boldsymbol{B} = \boldsymbol{B} \cdot \boldsymbol{A} = \boldsymbol{I}_n$, where $\boldsymbol{I}_n$ is the $n \times n$ identity matrix. The inverse of a matrix $\boldsymbol{A}$ can be defined by its adjugate matrix and determinant. Specifically, if $\boldsymbol{A}$ is invertable:

$$\boldsymbol{A}^{-1} = \frac{1}{\det(\boldsymbol{A})} adj(\boldsymbol{A}) \tag{1}$$

Additionally, If $\boldsymbol{A}$ is invertible:

- $(x \cdot \boldsymbol{A})^{-1} = x^{-1} \cdot \boldsymbol{A}^{-1}$ where x is a non-zero scalar.
- $\det(\boldsymbol{A}) \neq 0$.
- the number 0 is not an eigenvalue of $\boldsymbol{A}$ (see below).

In addition:

- $\boldsymbol{I}^{-1} = \boldsymbol{I}$

There are a number of ways of computing $\boldsymbol{A}^{-1}$. However, because, of the reliance of the inverse on the determinant and the adjugate matrix, none of them are straightforward for matrices larger than 2×2.

Consider the simple case where

$$\underset{(2\times2)}{\boldsymbol{A}} = \begin{bmatrix} 3 & 2 \\ -1 & 0 \end{bmatrix}$$

$$\begin{aligned} \underset{(2\times2)}{\boldsymbol{A}^{-1}} &= \frac{1}{\det(\boldsymbol{A})} adj(\boldsymbol{A}) \\ &= \frac{1}{\det(0-(-2)} \cdot \begin{bmatrix} 0 & -2 \\ 1 & 3 \end{bmatrix} \\ &= \begin{bmatrix} 0 & -2 \cdot 0.5 \\ 1 \cdot 0.5 & 3 \cdot 0.5 \end{bmatrix} \\ &= \begin{bmatrix} 0 & -1 \\ 0.5 & 1.5 \end{bmatrix} \end{aligned}$$

The inverse of a matrix can be calculated in **R** using the function `solve()`.

```
A <- matrix(2,2,data = c(3,-1,2,0))
solve(A)
```

```
     [,1] [,2]
[1,]  0.0 -1.0
[2,]  0.5  1.5
```

Eigenanalysis:

For an $n \times n$ square matrix $\boldsymbol{A}$, with eigenvector, $\boldsymbol{v}$, and eigenvalue, λ, we have:

$$\boldsymbol{A} \cdot \boldsymbol{v} = \lambda \cdot \boldsymbol{v}, \tag{2}$$

There will be n corresponding eigenvalues $(\lambda_1, \lambda_2, \ldots, \lambda_n)$ and eigenvectors $(\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n)$ in the matrix decomposition, and the length (number of rows) of each eigenvector will also equal n. Spectral decomposition of a matrix through eigenanalysis is often used for the purpose of dimension reduction in approaches like principal components analysis.

Eigenvector generally refers to a **right eigenvector**. That is, the eigenvector column vector $\boldsymbol{v}$ is placed to the right of $\boldsymbol{A}$ in the multiplication, $\boldsymbol{A} \cdot \boldsymbol{v}$, as defined in Eq.2. The **left eigenvector** would result from $\boldsymbol{u} \cdot \boldsymbol{A}$, where $\boldsymbol{u}$ is a $1 \times n$ row vector.

For a square matrix $\boldsymbol{A}$,

- $\sum_{i=1}^{n} \lambda_i = tr(\boldsymbol{A})$
- $\prod_{i=1}^{n} \lambda_i = \det(\boldsymbol{A})$
- The left and right eigenvectors will only be equal for a symmetric matrix.

Example: As a numerical example, let

$$\underset{(2\times 2)}{\boldsymbol{A}} = \begin{bmatrix} 1 & -2 \\ -2 & 1 \end{bmatrix}$$

Rearranging Eq. 2 we have:

$$\det(\boldsymbol{A} - \lambda \cdot \boldsymbol{I}_n) = 0$$

To solve for λ we have:

$$\begin{aligned} \begin{bmatrix} 1 & -2 \\ -2 & 1 \end{bmatrix} - \lambda \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} &= 0 \\ \begin{vmatrix} 1-\lambda & -2 \\ -2 & 1-\lambda \end{vmatrix} &= 0 \\ (1-\lambda)(1-\lambda) - 4 &= 0 \\ (\lambda^2 - 2\lambda + 1) - 4 &= 0 \end{aligned}$$

We have the quadratic equation: $\lambda^2 - 2\lambda - 3 = 0$.

Solving this equation for λ (using the quadratic formula), we have the solutions $\lambda_1 = 3$, $\lambda_2 = -1$. These are first two (and in this case, only) eigenvalues.

Inserting λ_1 into Eq.2 we obtain the first eigenvector

$$\begin{aligned}
\boldsymbol{A} \cdot \boldsymbol{v_1} &= \lambda_1 \cdot \boldsymbol{v_1} \\
\begin{bmatrix} 1 & -2 \\ -2 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} &= 3 \cdot \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \\
\begin{bmatrix} 1 \cdot x_1 + (-2 \cdot y_1) \\ -2 \cdot x_1 + (1 \cdot y_1) \end{bmatrix} &= \begin{bmatrix} 3 \cdot x_1 \\ 3 \cdot y_1 \end{bmatrix}
\end{aligned}$$

We have two equations:

$$\begin{aligned}
x_1 - 2y_1 &= 3x_1 \\
-2x_1 - y_1 &= 3y_1
\end{aligned}$$

Rearranging, we have:

$$\begin{aligned}
-2x_1 - 2y_1 &= 0 \\
-2x_1 - 2y_1 &= 0
\end{aligned}$$

Both equations indicate that $x_1 = -y_1$. Arbitrarily letting $x_1 = -1$, we have $y_1 = 1$. The resulting *unstandardized* first eigenvector is $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$. Eigenanalysis algorithms generally rescale eigenvectors so that their sum of squares = 1 (so they have unit length). The scaling coefficient for the ith eigenvector is found using:

$$k_i = \frac{1}{\sqrt{\sum_{i=1}^{n} \boldsymbol{v}_i^2}}$$

For our example,

$$k_1 = \frac{1}{\sqrt{-1^2 + 1^2}} = 0.7071$$

To get our rescaled eigenvector, we multiply $\boldsymbol{v}_1$ by the scalar, k_1. Thus our first scaled eigenvector, $\boldsymbol{e}_1$ is: $\begin{bmatrix} -0.7071 \\ 0.7071 \end{bmatrix}$. To get the second eigenvector, we repeat this process using the second eigenvalue. We find that $\boldsymbol{e}_2 = \begin{bmatrix} -0.7071 \\ -0.7071 \end{bmatrix}$:

As with other matrix operations, obtaining eigenvalues and eigenvectors for matrices larger than 3×3 is exceedingly difficult to do "by hand".

Eigenvalues and eigenvectors of a matrix can be obtained in **R** using the function `eigen()`. Here are the right-hand eigenvectors

```
A <- matrix(2,2,data = c(1,-2,-2,1))
eigen(A)

eigen() decomposition
$values
[1]  3 -1
```

```
$vectors
           [,1]       [,2]
[1,] -0.7071068 -0.7071068
[2,]  0.7071068 -0.7071068
```

Here are the left-hand eigenvectors

```
LH.eigen <- function(A){
  eigen(t(A))
}

LH.eigen(A)

eigen() decomposition
$values
[1]  3 -1

$vectors
           [,1]       [,2]
[1,] -0.7071068 -0.7071068
[2,]  0.7071068 -0.7071068
```

Because (in this case) $\boldsymbol{A}$ is symmetric, the left and right-hand eigenvectors are the same.

1.3 Graph theory terminology

The graph theory definitions and descriptions given here generally follow Aho *et al.* (2023).

Graph:

An ostensibly graphical method of representing systems of potentially connected **nodes** (also called vertices) (Fig 1).

Figure 1: A simple graph with two connected nodes.

Undirected versus Directed Graph:

With an **undirected graph** we can assume that communication can occur bidirectionally along connecting lines (generally called **edges**) between nodes. Conversely, with a **directed graph** (or

digraph), connections between node (generally called **arcs**) are assumed to be unidirectional (Fig 2).

A digraph can formally defined as an ordered pair, $D = (N, A)$ where N is a set of nodes and A is a set of arcs that link the nodes. If z is an arc from node u to node v, we denote this as $z = \overrightarrow{uv}$.

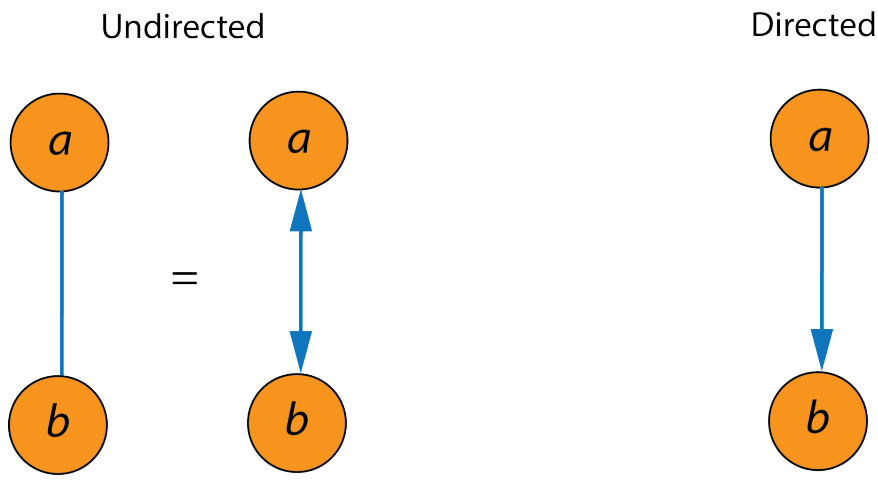

Figure 2: An undirected graph versus a directed graph.

Walk and Path:

If W_k is a **walk** of length k, from node n_0 to node n_k, then we have a finite sequence $W_k = (N, A)$ of the form:

$$N = \{n_0, n_1, \ldots, n_k\}$$
$$A = \{\overrightarrow{n_0, n_1}, \overrightarrow{n_1, n_2}, \ldots, \overrightarrow{n_{k-1}, n_k}\}$$

where n_i and n_{i+1} are adjacent.

A **path** is a walk in which no nodes appear more than once, except in the case of a cycle (see below). In this case, n_k can equal to n_0.

Cyclic versus Acyclic:

A graph **cycle** occurs when a path among nodes starts and ends at the same node. An acyclic graph will have no cycles (Fig 3). The term **DAG** is often used to indicate a **directed acyclic graph**.

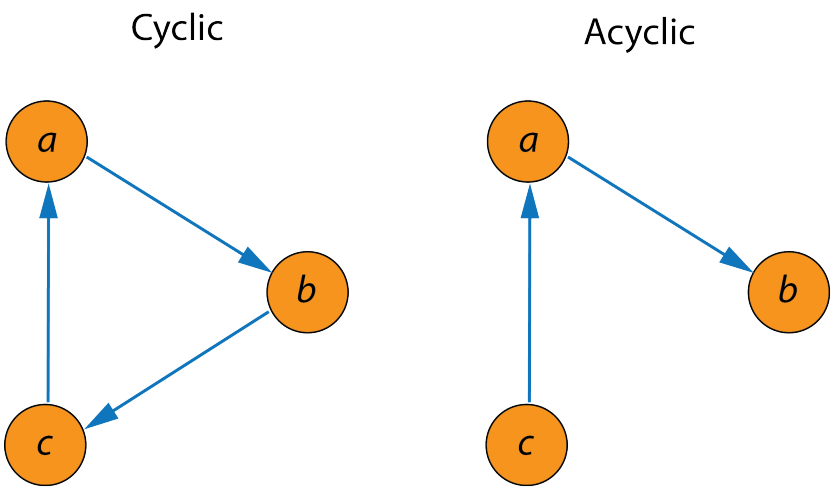

Figure 3: A cyclic graph versus an acyclic graph.

Strongly Connected, Weakly Connected, and Disconnected:

A digraph will be **strongly connected** if every node is reachable from every other node, **weakly connected** if every node is reachable after replacing all oriented arcs with bidirectional links between adjacent nodes, and **disconnected** if at least two nodes cannot be connected, even after applying bidirectional links (Fig 4).

Local versus Global Graph Perspectives:

Graph-theoretic approaches for describing graphs can be separated into **local** measures that describe the characteristics of individual nodes or arcs, and **global** measures that summarize the characteristics of an entire graph.

Degree:

The local importance of nodes to the functioning of a network can be assessed with a large number of approaches. The simplest of these is the nodal **degree**. That is, the number of arcs connected to a node. In a digraph we can distinguish the **indegree** and **outdegree** of a node as the number of arcs with that node as head and the number of arcs with that node as tail. Thus, the degree of a node will be the sum of its indegree and outdegree. As an example, in left-hand graph in Fig 4, node *b* has degree 4, because indegree + outdegree = 2 + 2 = 4. In the center graph, however, node *b* has degree 2, because indegree + outdegree = 1 + 1 = 2.

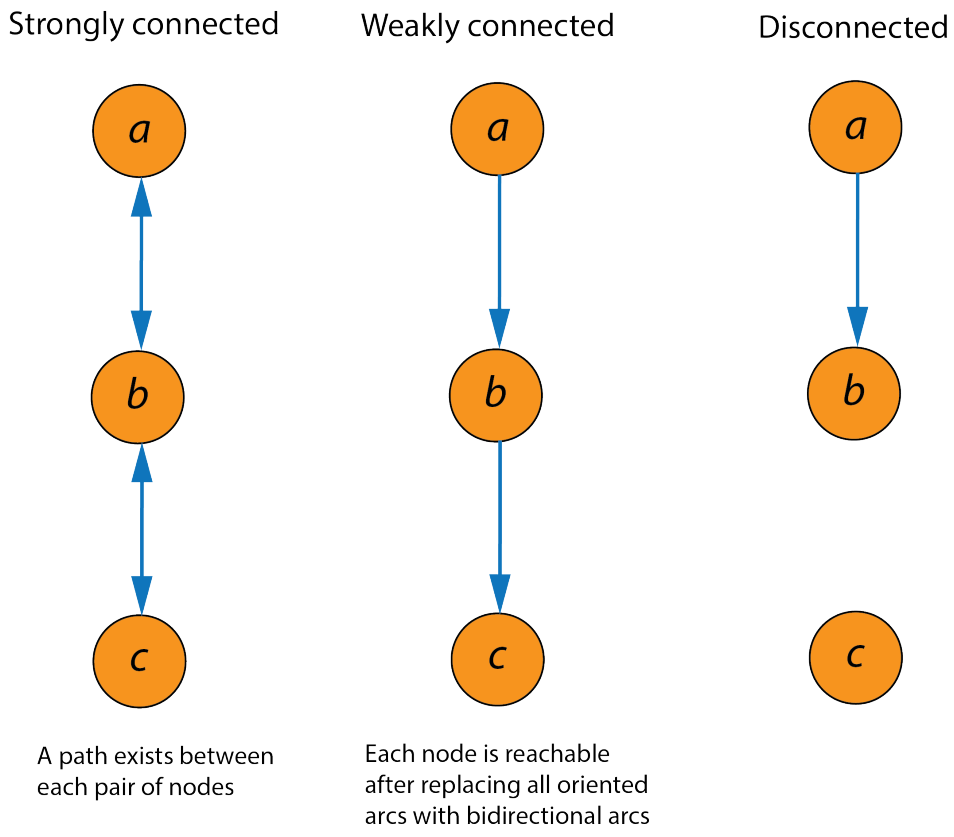

Figure 4: Strongly connected, weakly connected, and disconnected graphs.

Adjacency Matrix:

Graphs can be represented with an $n \times n$ adjacency matrix, $\boldsymbol{A}$, whose entries, A_{ij} indicate that an arc exists from node i to j with the designation: $A_{ij} = 1$, or that there is no arc from i to j, with the designation: $A_{ij} = 0$.

Consider the central (weakly connected) graph in Fig 4. Its adjacency matrix is:

$$\underset{(3\times3)}{\boldsymbol{A}} = \begin{array}{c} \begin{matrix} \text{a} & \text{b} & \text{c} \end{matrix} \\ \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} \text{a} \\ \text{b} \\ \text{c} \end{matrix}$$

In the matrix above, the numbers characterize whether arcs connect nodes or not. Rows indicate the starting location (tail) of an arc, and columns indicate the ending location (head) of an arc. The entry 1 at cell $A_{1,2}$ indicates the presence of the arc $\overrightarrow{ab}$. Note, however, because the adjacency matrix represents a weakly connected DAG, there is no connecting arc from *b* to *a*. That is, while

$\overrightarrow{ab}$ exists, $\overleftarrow{ab}$ does not exist.

Graphs, and their adjacency matrices can be obtained using functions from the **R** package *igraph*. Here we codify the central graph in Fig 4 and obtain its adjacency matrix.

```
library(igraph)

G <- graph_from_literal(a --+ b --+ c)
A <- as_adjacency_matrix(G, sparse = F)
A

  a b c
a 0 1 0
b 0 0 1
c 0 0 0
```

The adjacency matrix can be used to describe many useful network characteristics. For example, the i, j entry in $\boldsymbol{A}^k$ will give the number of paths in the graph from node i to node j, of length k.

```
A %*% A

  a b c
a 0 0 1
b 0 0 0
c 0 0 0
```

There is one path of length 2 in the central graph in Fig 4. It goes from node a to node c.

Stream DAGs:

From Aho *et al.* (2023)

> "Streams networks can be represented using graphs, with streams segments as arcs bounded by nodes occurring at hydrologically meaningful locations such as sensor sites, network confluences or splits, sources, sinks. Because they are strongly driven by hydrological potentials resulting from fixed elevational gradients, graphs that are most appropriate for describing passive stream network characteristics such as transport and discharge, will be both directed (with an orientation from sources to sink) and acyclic."

Graph Weights:

From Aho *et al.* (2023)

> "Nuance and realism can be enhanced in graphs by adding information to nodes and/or arcs in the form of weights. Weighting information particularly relevant to non-perennial stream DAGs includes flow rates, instream lengths, probabilities of aquatic

organism dispersal, water quality components including nutrients or sediment, upstream drainage area, and/or probabilities of surface and subsurface water presence. Weights can be assessed alongside the strictly topological relationships of nodes and arcs when describing DAGs."

Weights can be added to *igraph* graph objects in several ways. First one can add a `weight` attribute to arcs using the function `igraph::E()` (for edge). For illustration, consider the central (weakly connected) graph in Fig 4, which has two arcs.

```
G1 <- G
E(G1)$weight <- c(0.1, 0.2)
```

The weights should be assigned in the order that they are listed in the `E()` output. Thus, the first weight, 0.1, will go with $\overrightarrow{ab}$ and the second weight, 0.2, will go with $\overrightarrow{bc}$.

```
E(G1)

+ 2/2 edges from 3d2de23 (vertex names):
[1] a->b b->c
```

Second, one can use the function `set_edge_attr()`.

```
weights <- c(0.1, 0.2)
G2 <- set_edge_attr(G, "weight", value = as.numeric(weights))
attr <- "weight"
```

Third, many *igraph* functions have an edge weight argument that allows user imputation of weights.

We see that the inclusion of weights has altered the adjacency matrix. Specifically, weights are now listed at connecting arc locations (instead of ones).

```
as_adjacency_matrix(G2, attr = "weight", sparse = F)

  a   b   c
a 0 0.1 0.0
b 0 0.0 0.2
c 0 0.0 0.0
```

1.4 Infinite Series

Geometric series:

The following is a **geometric series**:

$$\sum_{n=1}^{\infty} ar^{n-1} = a + ar + ar^2 + \cdots$$

If $|r| < 1$, then the infinite series converges to:

$$\sum_{n=1}^{\infty} ar^{n-1} = \frac{a}{1-r}$$

If $|r| \geq 1$, then the series is divergent.

2 Alpha centrality

2.1 Motivation and Derivation

In a graph theoretic representation (see Section 1.3), **degree centrality** is simply the degree of a node. **Eigenvector centrality** refers to the corresponding entry of a node in the principal eigenvector of the graph adjacency matrix. This method extends degree centrality by accounting for a node's connection to nodes that are themselves important to the network (Newman, 2018). Eigenvector centrality, however poses a number of problems for stream DAGs (Aho *et al.*, 2023).

1. First, the adjacency matrix of a DAG will be asymmetric, and will thus possess distinct left- and right-hand eigenvectors.

2. Second, "source nodes", which must have indegree zero, will drive all downstream nodes to also have an eigenvector centrality of zero (Newman, 2018).

Alpha centrality and Katz centrality address several of these issues. Alpha centrality can be defined as the infinite series:

$$\boldsymbol{x} = \sum_{k=0}^{\infty} \left(\alpha^k (\boldsymbol{A^T})^k \right) \cdot \mathbf{1} = \sum_{k=0}^{\infty} (\alpha \boldsymbol{A^T})^k \cdot \mathbf{1} \tag{3}$$

where $\boldsymbol{x}$ is a vector of resulting alpha-centralities for each graph node, $\boldsymbol{A^T}$ is the transposed $n \times n$ graph adjacency matrix, $\mathbf{1}$ is a column vector of ones with n entries, and α is a user-defined scalar.

Expanding Eq 3 we have:

$$\boldsymbol{x} = \sum_{k=0}^{\infty} (\alpha \boldsymbol{A^T})^k \cdot \mathbf{1} = (\boldsymbol{I} + \alpha \boldsymbol{A^T} + (\alpha \boldsymbol{A^T})^2 + \cdots) \cdot \mathbf{1} \tag{4}$$

where $\boldsymbol{I}$ is the $n \times n$ identity matrix. If this sum converges, then, under the geometric series (Section 1.4), we can use the matrix notation[2]:

$$\boldsymbol{x} = \left(\boldsymbol{I} - \alpha \boldsymbol{A^T} \right)^{-1} \cdot \mathbf{1} \tag{5}$$

[2]Newman (2018) and other sources (including Aho *et al.* (2023), who probably based their definition on Newman (2018)), leave out the transpose of $\boldsymbol{A}$. This transpose (for the form of $\boldsymbol{A}$ and $\mathbf{1}$ used here) is required, however, to calculate alpha-centrality correctly.

Alpha-centrality provides a "free centrality" (often termed β) to all nodes, regardless of their topology. Explicit inclusion of this β term in Eq. 5 results in: $\left(\boldsymbol{I} - \alpha \boldsymbol{A}^{\boldsymbol{T}}\right)^{-1} \cdot \beta \mathbf{1}$. For convenience, β is generally taken to be 1 (Newman, 2018), resulting in the simpler form of alpha-centrality shown in Eq. 5. The α parameter specifies the balance between eigenvector centrality and the "free centrality". As α increases, the "free centrality" is de-emphasized compared to eigenvector centrality.

2.2 α

Calculation of alpha-centrality requires specification of the α scalar in Eq. 5. Katz (1953) recommended that α be in (0, 1]. This would allow one to essentially *de-emphasize longer paths* in the network when computing the centrality of nodes (see below). The α term often defaults to 1 in α-centrality computational algorithms, including those in *igraph* and *streamDAG*, which wraps the function `igraph::alpha_centrality`.

As $\alpha \rightarrow 0$, $\alpha \boldsymbol{A}^{\boldsymbol{T}}$ drops from Eq. 5, and all nodes will have an alpha-centrality of one, because $\boldsymbol{I}^{-1} = \boldsymbol{I}$. When α equals the reciprocal of the of the largest eigenvalue of $\boldsymbol{A}$, non-finite alpha-centrality outcomes will occur (Newman, 2018). As general guidance, one can set α to be slightly less than λ_{max}^{-1} to obtain outcomes that are numerically similar to those from conventional eigenvector centrality. Unfortunately, this guidance is not useful for DAGs for the very reason that alpha-centrality was developed in the first place: a useful eigendecomposition will not exist for $\boldsymbol{A}$.

Because alpha-centrality equals degree centrality in the limit $\alpha \rightarrow 0$, and equals eigenvector centrality in the limit $\alpha \rightarrow \lambda_{max}^{-1}$, these measures can both be viewed as special cases of alpha-centrality (Newman, 2018).

2.3 Mechanics of Alpha-centrality

The mechanics of alpha-centrality become evident upon deconstructing Eq. 5.

- If the underlying graph is unweighted (Section 1.3), the elements of its adjacency matrix $\boldsymbol{A}$ will be ones if they represent a connecting arc and zero otherwise.
- The operation $\alpha \boldsymbol{A}^{\boldsymbol{T}}$ results in values of α being set for connecting arcs, if we let columns represent the start of the arc and rows represent the end of the arc. The opposite of the conventional interpretation.
- The difference $\boldsymbol{I} - \alpha \boldsymbol{A}^{\boldsymbol{T}}$ results in an $n \times n$ matrix with ones on the diagonal (if there are no self-looping nodes in the graph) and $-\alpha$ at connecting arcs. For a stream DAG, these negative values will now generally occur in the lower triangle.
- Let:

$$\boldsymbol{B} = \boldsymbol{I} - \alpha \boldsymbol{A}^{\boldsymbol{T}}$$

The inverse $\boldsymbol{B}^{-1}$ will also be an $n \times n$ matrix. The rows of $\boldsymbol{B}^{-1}$ represent *paths* associated with a particular node, along with an additional 1 (representing "free centrality") on the diagonal. The appearance of ones at existing paths follows from the form of Alpha-centrality given in Eq 3. Specifically, because it tracks the infinite sum of matrix powers, $\boldsymbol{B}^{-1}$, counts the number of walks of any length starting from node represented by jth column of $\boldsymbol{B}^{-1}$, and ending at the node represented by the ith row, adjusted by the corresponding power of the decay parameter, α.

- The complete operation $\boldsymbol{B}^{-1} \cdot \mathbf{1}$ results in a $n \times 1$ column vector comprised of the row sums of $\boldsymbol{B}^{-1}$.

2.4 Introductory Example

Assume that we wish to calculate alpha-centrality for the DAG in Fig 5

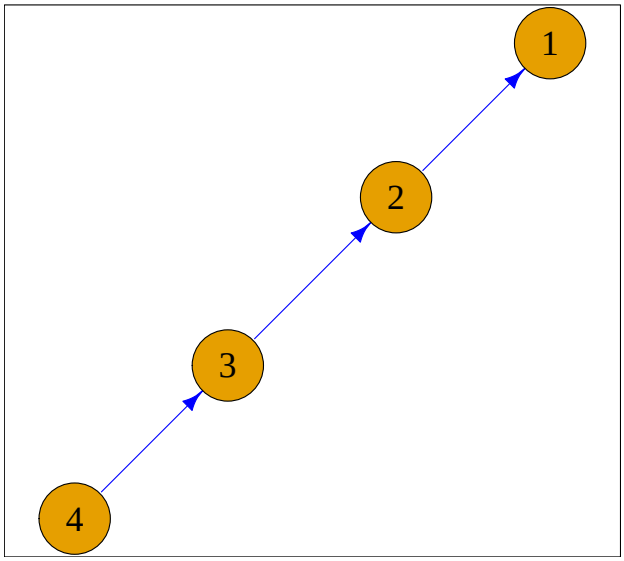

Figure 5: Simple unbranched DAG.

- We let $\alpha = 1$.

```
alpha <- 1
```

- Here we codify the graph, and view its adjacency matrix.

```
G <- graph_from_literal(4 --+ 3 --+ 2 --+ 1)
A <- as_adjacency_matrix(G, sparse = F)
A
```

```
  4 3 2 1
4 0 1 0 0
3 0 0 1 0
2 0 0 0 1
1 0 0 0 0
```

- Here is $\alpha(=1)$ times the transpose of the adjacency matrix.

```
alpha * t(A)
```

```
  4 3 2 1
4 0 0 0 0
3 1 0 0 0
2 0 1 0 0
1 0 0 1 0
```

- And here is $\boldsymbol{I} - \alpha \boldsymbol{A^T}$

```
I <- matrix(0,4,4); diag(I) <- 1
I - (alpha * t(A))
```

```
   4  3  2 1
4  1  0  0 0
3 -1  1  0 0
2  0 -1  1 0
1  0  0 -1 1
```

- Let $\boldsymbol{B} = \boldsymbol{I} - \alpha \boldsymbol{A^T}$, then $\boldsymbol{B}^{-1}$ will also be an $n \times n$ matrix, whose rows now represent paths associated with a particular node (along with an additional one on the diagonal, representing the "free centrality" assigned to each node).

```
B <- I - (alpha * t(A))
solve(B)
```

```
  4 3 2 1
4 1 0 0 0
3 1 1 0 0
2 1 1 1 0
1 1 1 1 1
```

- $\boldsymbol{B}^{-1} \cdot \mathbf{1}$ results in a $n \times 1$ column vector comprised of the row sums of $\boldsymbol{B}^{-1}$. These are the alpha centralities for each node.

```
one <- matrix(4,1, data = rep(1,4))
solve(B) %*% one
```

```
  [,1]
4    1
3    2
2    3
1    4
```

2.5 The Effect of α

Assume that we wish to obtain alpha-centralities for the simple DAG in Fig 5, but wish to use $\alpha = 0.5$.

- We have:

```
A <- as_adjacency_matrix(G, sparse = F)
alpha <- 0.5
alpha * t(A)
```

```
    4   3   2 1
4 0.0 0.0 0.0 0
3 0.5 0.0 0.0 0
2 0.0 0.5 0.0 0
1 0.0 0.0 0.5 0
```

- Subtracting this product from $\boldsymbol{I}$ results in:

```
I <- matrix(0,4,4); diag(I) <- 1
B <- I - alpha * t(A)
B
```

```
     4    3    2 1
4  1.0  0.0  0.0 0
3 -0.5  1.0  0.0 0
2  0.0 -0.5  1.0 0
1  0.0  0.0 -0.5 1
```

- And the inverse of $\boldsymbol{B}$ is:

```
solve(B)
```

```
      4    3   2 1
4 1.000 0.00 0.0 0
3 0.500 1.00 0.0 0
2 0.250 0.50 1.0 0
1 0.125 0.25 0.5 1
```

- As before, the row sums of $\boldsymbol{B}^{-1}$ give the alpha-centralities.

```
solve(B) %*% rep(1,4)
```

```
   [,1]
4 1.000
3 1.500
2 1.750
1 1.875
```

It is clear that for a DAG with order n, and no branching (no joins or splits), we will have the following matrix framework for $\boldsymbol{B}^{-1}$

$$\underset{(n\times n)}{\boldsymbol{B}^{-1}} = (\boldsymbol{I} - \alpha \boldsymbol{A}^{\boldsymbol{T}})^{-1} = \begin{bmatrix} 1.0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \alpha & 1.0 & 0 & 0 & 0 & \cdots & 0 \\ \alpha^2 & \alpha & 1.0 & 0 & 0 & \cdots & 0 \\ \alpha^3 & \alpha^2 & \alpha & 1.0 & 0 & \cdots & 0 \\ \alpha^4 & \alpha^3 & \alpha^2 & \alpha & 1.0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha^{n-1} & \alpha^{n-2} & \alpha^{n-3} & \alpha^{n-4} & \alpha^{n-5} & \cdots & 1.0 \end{bmatrix} \tag{6}$$

Thus, paths given in the off-diagonal elements of $\boldsymbol{B}^{-1}$ are weighted by α raised to the power of the path length.

- For $\alpha \in (0, 1]$, longer paths will have less influence when calculating alpha-centrality for a node. A path with maximum length $(n-1)$ will have minimum weight, α^{n-1}, whereas a path with minimum length, 1, will have maximum weight α. For this reason α has been called a decay parameter.

- Specifying $\alpha = 1$ will cause α to drop out of Eq. 5, and all paths will have the same α (non-)weighting, regardless of length.

2.6 The Effect of Branching

Lets now consider the effect of network branching on alpha-centrality using a graph with a join (Fig 6A), and a graph with a split and join (an island) (Fig 6B).

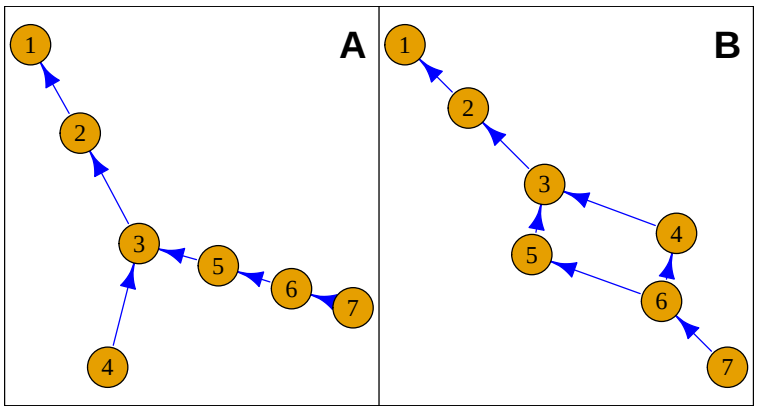

Figure 6: Simple branched DAGs.

- We have:

```
graph_A <- graph_from_literal(7 --+ 6 --+ 5 --+ 3,
                              4 --+ 3, 3--+ 2, 2--+ 1)
graph_B <- graph_from_literal(7 --+ 6 --+ 5, 6 --+ 4,
                              5 --+ 3, 4 --+ 3, 3--+ 2, 2--+ 1)
```

- The adjacency matrices are:

```
A_A <- as_adjacency_matrix(graph_A, sparse = F)
A_A

  7 6 5 3 4 2 1
7 0 1 0 0 0 0 0
6 0 0 1 0 0 0 0
5 0 0 0 1 0 0 0
3 0 0 0 0 0 1 0
4 0 0 0 1 0 0 0
2 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0

A_B <- as_adjacency_matrix(graph_B, sparse = F)
A_B

  7 6 5 4 3 2 1
7 0 1 0 0 0 0 0
6 0 0 1 1 0 0 0
5 0 0 0 0 1 0 0
4 0 0 0 0 1 0 0
3 0 0 0 0 0 1 0
2 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0
```

Note that joins show up as multiple ones in adjacency matrix columns (see column for node 3 in A_A) and splits appear as multiple ones in rows (see row for node 6 in A_B).

- Letting $\alpha = 1$, we find $\boldsymbol{I} - \alpha \boldsymbol{A}^{\boldsymbol{T}}$

```
alpha <- 1
I <- matrix(0,7,7); diag(I) <- 1
I - (alpha * t(A_A))
```

```
   7  6  5  3  4  2 1
7  1  0  0  0  0  0 0
6 -1  1  0  0  0  0 0
5  0 -1  1  0  0  0 0
3  0  0 -1  1 -1  0 0
4  0  0  0  0  1  0 0
2  0  0  0 -1  0  1 0
1  0  0  0  0  0 -1 1
```

```
I - (alpha * t(A_B))
```

```
   7  6  5  4  3  2 1
7  1  0  0  0  0  0 0
6 -1  1  0  0  0  0 0
5  0 -1  1  0  0  0 0
4  0 -1  0  1  0  0 0
3  0  0 -1 -1  1  0 0
2  0  0  0  0 -1  1 0
1  0  0  0  0  0 -1 1
```

And $\boldsymbol{B}^{-1}$ where $\boldsymbol{B} = \boldsymbol{I} - \alpha \boldsymbol{A}^{\boldsymbol{T}}$.

```
B_A <- I - alpha * t(A_A)
solve(B_A)
```

```
  7 6 5 3 4 2 1
7 1 0 0 0 0 0 0
6 1 1 0 0 0 0 0
5 1 1 1 0 0 0 0
3 1 1 1 1 1 0 0
4 0 0 0 0 1 0 0
2 1 1 1 1 1 1 0
1 1 1 1 1 1 1 1
```

```
B_B <- I - alpha * t(A_B)
solve(B_B)
```

```
  7 6 5 4 3 2 1
7 1 0 0 0 0 0 0
6 1 1 0 0 0 0 0
5 1 1 1 0 0 0 0
4 1 1 0 1 0 0 0
3 2 2 1 1 1 0 0
2 2 2 1 1 1 1 0
1 2 2 1 1 1 1 1
```

Note that there are *no* paths to node 4 from nodes 7, 6, 5, and 3 in graph A, and that there are two paths to nodes 1, 2, and 3 from nodes 7, 6, 5 in graph B.

- As before, row sums of $\boldsymbol{B}^{-1}$ provide alpha-centralities.

```
solve(B_A) %*% rep(1, 7)

  [,1]
7    1
6    2
5    3
3    5
4    1
2    6
1    7

solve(B_B) %*% rep(1, 7)

  [,1]
7    1
6    2
5    3
4    3
3    7
2    8
1    9
```

2.7 The Combined Effect of α *and* Branching

We now consider the interplay of α and branching:

- In particular, we let $\alpha = 0.5$ and calculate alpha-centralities for the graphs in Fig 6

```
alpha <- 0.5
I <- matrix(0,7,7); diag(I) <- 1
B_A <- I - (alpha * t(A_A))
B_B <- I - (alpha * t(A_B))

solve(B_A)
```

```
        7      6     5    3     4   2 1
7 1.00000 0.0000 0.000 0.00 0.000 0.0 0
6 0.50000 1.0000 0.000 0.00 0.000 0.0 0
5 0.25000 0.5000 1.000 0.00 0.000 0.0 0
3 0.12500 0.2500 0.500 1.00 0.500 0.0 0
4 0.00000 0.0000 0.000 0.00 1.000 0.0 0
2 0.06250 0.1250 0.250 0.50 0.250 1.0 0
1 0.03125 0.0625 0.125 0.25 0.125 0.5 1
```

```
solve(B_B)
```

```
       7     6     5     4    3   2 1
7 1.0000 0.000 0.000 0.000 0.00 0.0 0
6 0.5000 1.000 0.000 0.000 0.00 0.0 0
5 0.2500 0.500 1.000 0.000 0.00 0.0 0
4 0.2500 0.500 0.000 1.000 0.00 0.0 0
3 0.2500 0.500 0.500 0.500 1.00 0.0 0
2 0.1250 0.250 0.250 0.250 0.50 1.0 0
1 0.0625 0.125 0.125 0.125 0.25 0.5 1
```

- A resemblance to the basic unbranched framework for $\boldsymbol{B}^{-1}$ shown in Eq. 6 remains, although it is now more difficult to discern. For instance, in graph A the path from node 7 to node 2 has length 4 (Fig 6A), so it is given weight $\alpha^4 = 0.5^4 = 0.0625$ in the $\boldsymbol{B}^{-1}$ matrix for graph A (intersection of row 6 and column 1). There is no path between nodes 7 and 4 in graph A, so this path is given weight 0 in the $\boldsymbol{B}^{-1}$ matrix for graph A (intersection of row 5 and column 1). In graph B there are two ways to get from node 7 to node 2, and both of these have length 4 (Fig 6B). Thus, the resulting weight for this connection is $2 \cdot 0.5^4 = 0.125$ in the $\boldsymbol{B}^{-1}$ matrix for graph B (intersection of row 6 and column 1). Likewise, there are two paths from node 7 to node 1 and both of these have length 5, so we have the weight $2 \cdot 0.5^5 = 0.0625$ for this path (intersection of row 7 and column 1).

2.8 Interpreting Alpha-centrality in Stream DAGs

2.8.1 Unweighted DAGs

As additional unweighted graph examples, consider the stream DAGs in Fig 7. The relationship between the alpha-centrality (for three different values of α), the number of paths, and upstream

network order for the outlet node (i.e., node 1) for each of the DAGs shown in Fig 7, is given in Table 1.

Note that in the special (but frequently observed) case that a graph is unweighted, and $\alpha = 1$, alpha centrality will be the number of paths to a node, plus 1 (Table 1). Thus a reasonable modification to Eq. 5 would be:

$$\begin{aligned} \boldsymbol{x_1} &= \left(\boldsymbol{I} - \alpha \boldsymbol{A^T}\right)^{-1} \cdot \mathbf{1} - 1, \quad \text{or} \\ \boldsymbol{x_1} &= \left\{\left(\boldsymbol{I} - \alpha \boldsymbol{A^T}\right)^{-1} - \boldsymbol{I}\right\} \cdot \mathbf{1} \end{aligned} \tag{7}$$

As shown in Section 2.6 and 2.7, network splits followed by joins (resulting from islands) increase the number of paths to the outlet. For a stream DAG in the absence of conjoined splits and joins, the alpha-centrality of a stream DAG node will also be the order (the number of nodes) in the intact network upstream, plus one.

Also apparent in Table 1 is the effect of varying the α parameter in Eq. 5. Note that as α goes toward zero, the alpha-centrality of the outlet node decreases (it approaches the free centrality of 1 given to *all* nodes) because, even though the outlets will have more paths than any other node, longer paths associated with the outlet are down-weighted.

Table 1: Summary of outlet nodes (node 1) for DAGs in Fig 7. Alpha centralities calculated using $\alpha = 1$, $\alpha = 0.75$, and $\alpha = 0.5$.

Graph	Alpha-centrality			Paths	Order
	$\alpha = 0.5$	$\alpha = 0.75$	$\alpha = 1.0$		
A	1.50	1.75	2.00	1.00	1.00
B	1.75	2.31	3.00	2.00	2.00
C	2.00	3.16	5.00	4.00	4.00
D	2.44	4.46	8.00	7.00	7.00
E	2.50	4.77	9.00	8.00	8.00
F	3.00	5.52	10.00	9.00	9.00
G	2.25	4.21	8.00	7.00	7.00
H	2.31	4.69	10.00	9.00	8.00
I	2.34	5.04	12.00	11.00	9.00

2.8.2 Weighted DAGs

In addition to modifying α which (if $\alpha \in (0, 1]$) will diminish the importance of a path as a function of its length, numerical information can be added to stream DAG arcs, reflecting specific arc characteristics (e.g., flow rates, instream lengths). The result is a **weighted graph**.

From the perspective of alpha-centrality, weighting modifies the adjacency matrix $\boldsymbol{A}$ in Eq. 5, so that weights occur at cells representing connecting arcs, instead of ones.

As a relatively complex weighted example, consider the intermittent stream DAG in Fig 8, with 19 nodes and 18 arcs. Here, arcs are weighted by the probability of surface water at that arc (numbers above arcs).

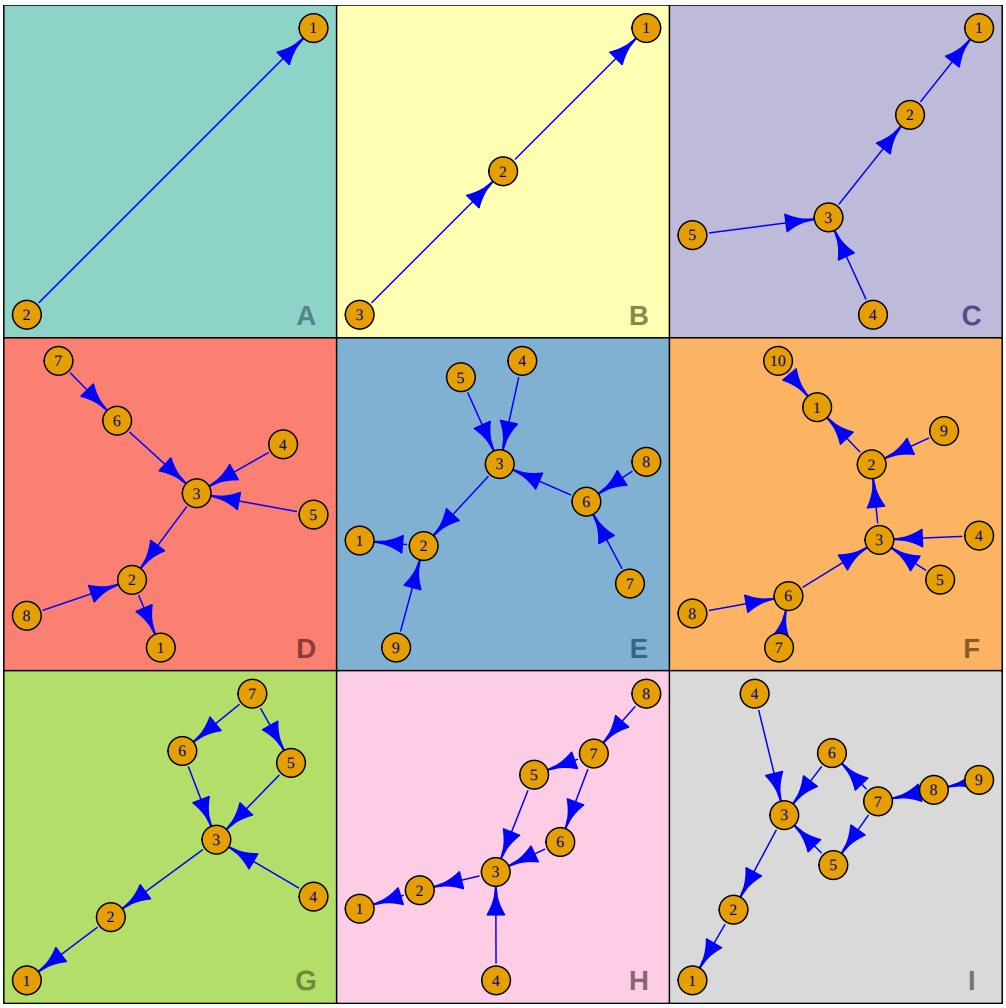

Figure 7: Example stream DAGs. Note that the outlet node has the label 1 in each DAG.

- As before, we define define the graph using *igraph* scripts.

```
G <- graph_from_literal(a --+ c, c --+ e, e --+ f, f --+ p,
                        p --+ q, q --+ r,
                        b --+ d, d --+ e,
                        g --+ i, i --+ j, i --+ k, k --+ m,
                        j --+ m, m --+ n,  n--+ o, o --+ p,
                        h --+ l, l --+ n)

weight.matrix <- data.frame(matrix(ncol = 2, nrow = 18, data = c(
  "a->c", 0.2, "c->e", 0.2, "e->f", 0.2,  "f->p", 0.5,  "p->q", 0.7,
  "q->r", 0.9, "b->d", 0.1, "d->e", 0.3,  "g->i", 0.2,  "i->j", 0.1,
  "i->k", 0.6, "j->m", 0.5, "k->m", 0.5,  "m->n", 0.4,  "n->o", 0.3,
  "o->p", 0.2,  "h->l", 0.1, "l->n", 0.3), byrow = T))

names(weight.matrix) <- c("Arc", "Weight")
weight.matrix$"Weight" <- as.numeric(weight.matrix$"Weight")
```

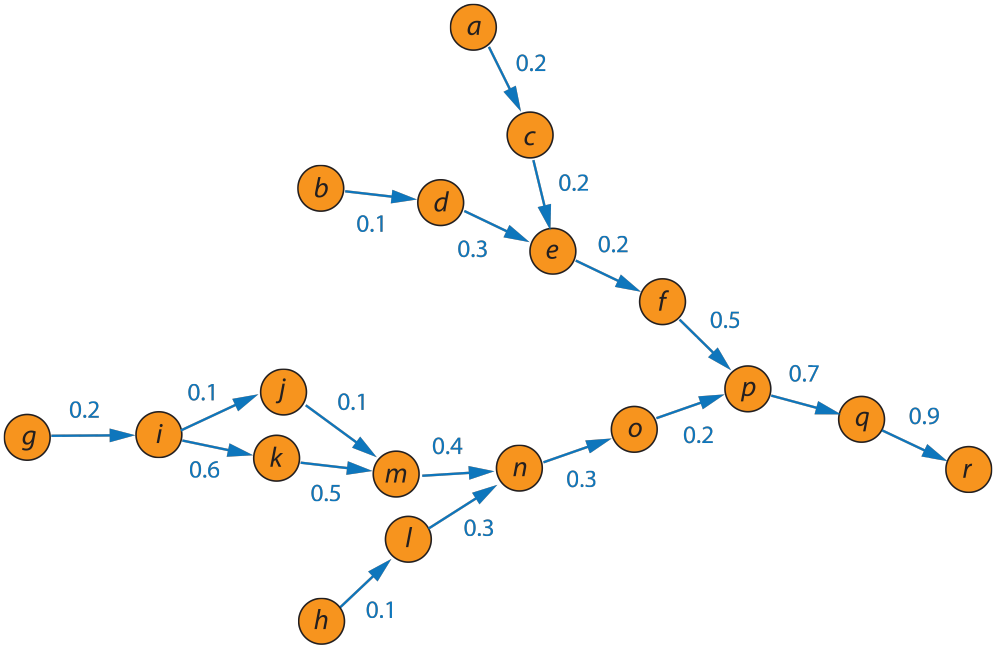

Figure 8: Example stream DAG. Numbers above arcs are probabilities of segment presence.

- We will let $\alpha = 1$.

```
alpha <- 1
```

- Here is the adjacency matrix.

```
A <- as_adjacency_matrix(G, sparse = F)
A
```

```
  a c e f p q r b d g i j k m n o h l
a 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
c 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
e 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
f 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
p 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
q 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
r 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
b 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
d 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
g 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
i 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0
j 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
k 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
```

```
m 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
n 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
o 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
h 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
l 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
```

- To set graph weights I use:

```
W <- G
weights <- weight.matrix[,2]

W <- set_edge_attr(W, "weight", value = as.numeric(weights))
attr <- "weight"
```

- Note that weighting graph arcs has radically changed the form of the adjacency matrix, now termed $\boldsymbol{D}$. Specifically, weights are now listed at connecting arc locations, instead of ones.

```
D <- as_adjacency_matrix(W, attr = attr, sparse = FALSE)
D
```

```
  a   c   e   f   p   q   r b   d g   i   j   k   m   n   o h   l
a 0 0.2 0.0 0.0 0.0 0.0 0.0 0 0.0 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0.0
c 0 0.0 0.2 0.0 0.0 0.0 0.0 0 0.0 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0.0
e 0 0.0 0.0 0.2 0.0 0.0 0.0 0 0.0 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0.0
f 0 0.0 0.0 0.0 0.5 0.0 0.0 0 0.0 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0.0
p 0 0.0 0.0 0.0 0.0 0.7 0.0 0 0.0 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0.0
q 0 0.0 0.0 0.0 0.0 0.0 0.9 0 0.0 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0.0
r 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0.0 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0.0
b 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0.1 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0.0
d 0 0.0 0.3 0.0 0.0 0.0 0.0 0 0.0 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0.0
g 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0.0 0 0.2 0.0 0.0 0.0 0.0 0.0 0 0.0
i 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0.0 0 0.0 0.1 0.6 0.0 0.0 0.0 0 0.0
j 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0.0 0 0.0 0.0 0.0 0.5 0.0 0.0 0 0.0
k 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0.0 0 0.0 0.0 0.0 0.5 0.0 0.0 0 0.0
m 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0.0 0 0.0 0.0 0.0 0.0 0.4 0.0 0 0.0
n 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0.0 0 0.0 0.0 0.0 0.0 0.0 0.3 0 0.0
o 0 0.0 0.0 0.0 0.2 0.0 0.0 0 0.0 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0.0
h 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0.0 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0.1
l 0 0.0 0.0 0.0 0.0 0.0 0.0 0 0.0 0 0.0 0.0 0.0 0.0 0.3 0.0 0 0.0
```

- Here I calculate $\boldsymbol{B} = \boldsymbol{I} - \alpha \boldsymbol{D}^{\boldsymbol{T}}$

```
I <- matrix(0, nrow = nrow(D), ncol = ncol(D)); diag(I) <- 1
B <- I - alpha * t(D)
```

- And here is $\boldsymbol{B}^{-1}$

```
solve(B)
```

```
        a      c     e     f    p   q r       b      d         g        i
a 1.00000 0.0000 0.000 0.000 0.00 0.0 0 0.00000 0.0000 0.0000000 0.000000
c 0.20000 1.0000 0.000 0.000 0.00 0.0 0 0.00000 0.0000 0.0000000 0.000000
e 0.04000 0.2000 1.000 0.000 0.00 0.0 0 0.03000 0.3000 0.0000000 0.000000
f 0.00800 0.0400 0.200 1.000 0.00 0.0 0 0.00600 0.0600 0.0000000 0.000000
p 0.00400 0.0200 0.100 0.500 1.00 0.0 0 0.00300 0.0300 0.0016800 0.008400
q 0.00280 0.0140 0.070 0.350 0.70 1.0 0 0.00210 0.0210 0.0011760 0.005880
r 0.00252 0.0126 0.063 0.315 0.63 0.9 1 0.00189 0.0189 0.0010584 0.005292
b 0.00000 0.0000 0.000 0.000 0.00 0.0 0 1.00000 0.0000 0.0000000 0.000000
d 0.00000 0.0000 0.000 0.000 0.00 0.0 0 0.10000 1.0000 0.0000000 0.000000
g 0.00000 0.0000 0.000 0.000 0.00 0.0 0 0.00000 0.0000 1.0000000 0.000000
i 0.00000 0.0000 0.000 0.000 0.00 0.0 0 0.00000 0.0000 0.2000000 1.000000
j 0.00000 0.0000 0.000 0.000 0.00 0.0 0 0.00000 0.0000 0.0200000 0.100000
k 0.00000 0.0000 0.000 0.000 0.00 0.0 0 0.00000 0.0000 0.1200000 0.600000
m 0.00000 0.0000 0.000 0.000 0.00 0.0 0 0.00000 0.0000 0.0700000 0.350000
n 0.00000 0.0000 0.000 0.000 0.00 0.0 0 0.00000 0.0000 0.0280000 0.140000
o 0.00000 0.0000 0.000 0.000 0.00 0.0 0 0.00000 0.0000 0.0084000 0.042000
h 0.00000 0.0000 0.000 0.000 0.00 0.0 0 0.00000 0.0000 0.0000000 0.000000
l 0.00000 0.0000 0.000 0.000 0.00 0.0 0 0.00000 0.0000 0.0000000 0.000000
        j       k       m      n     o        h       l
a 0.00000 0.00000 0.00000 0.0000 0.000 0.000000 0.00000
c 0.00000 0.00000 0.00000 0.0000 0.000 0.000000 0.00000
e 0.00000 0.00000 0.00000 0.0000 0.000 0.000000 0.00000
f 0.00000 0.00000 0.00000 0.0000 0.000 0.000000 0.00000
p 0.01200 0.01200 0.02400 0.0600 0.200 0.001800 0.01800
q 0.00840 0.00840 0.01680 0.0420 0.140 0.001260 0.01260
r 0.00756 0.00756 0.01512 0.0378 0.126 0.001134 0.01134
b 0.00000 0.00000 0.00000 0.0000 0.000 0.000000 0.00000
d 0.00000 0.00000 0.00000 0.0000 0.000 0.000000 0.00000
g 0.00000 0.00000 0.00000 0.0000 0.000 0.000000 0.00000
i 0.00000 0.00000 0.00000 0.0000 0.000 0.000000 0.00000
j 1.00000 0.00000 0.00000 0.0000 0.000 0.000000 0.00000
k 0.00000 1.00000 0.00000 0.0000 0.000 0.000000 0.00000
m 0.50000 0.50000 1.00000 0.0000 0.000 0.000000 0.00000
n 0.20000 0.20000 0.40000 1.0000 0.000 0.030000 0.30000
o 0.06000 0.06000 0.12000 0.3000 1.000 0.009000 0.09000
h 0.00000 0.00000 0.00000 0.0000 0.000 1.000000 0.00000
l 0.00000 0.00000 0.00000 0.0000 0.000 0.100000 1.00000
```

- As before, the row sums of $\boldsymbol{B}^{-1}$ give the alpha-centralities.

```
solve(B) %*% rep(1,nrow(D))
```

```
      [,1]
a 1.000000
c 1.200000
e 1.570000
```

```
f 1.314000
p 1.994880
q 2.396416
r 3.156774
b 1.000000
d 1.100000
g 1.000000
i 1.200000
j 1.120000
k 1.720000
m 2.420000
n 2.298000
o 1.689400
h 1.000000
l 1.100000
```

It is possible to trace entries in $\boldsymbol{B}^{-1}$ (in the previous **R** output) to the graphical representation in Fig 8. For example, the arc $\overrightarrow{ac}$ has path length 1, and so the weight (probability of stream presence) for this arc, 0.2 (Fig 8), is given directly in $\boldsymbol{B}^{-1}$ (intersection of row *c* and column *a*). The path $\overrightarrow{ace}$ is comprised of two vectors, $\overrightarrow{ac}$ and $\overrightarrow{ae}$, and each of these has weight 0.2 (Fig 8). Thus, the weight given to the path $\overrightarrow{ace}$ is $0.2 \times 0.2 = 0.04$ (intersection of row *e* and column *a*). As a more complex example, there are two paths from node g to node n: g, i, j, m, n and $\{g, i, k, m, n\}$ (Fig 8). The accumulated weights for these paths are: $0.2 \cdot 0.1 \cdot 0.1 \cdot 0.4 = 0.0008$ and $0.2 \cdot 0.6 \cdot 0.5 \cdot 0.4 = 0.024$ (Fig 8). Thus, the weight reported in $\boldsymbol{B}^{-1}$ for the connection between g and n, is $0.0008 + 0.0248 = 0.0248$ (intersection of row *n* and column *g*).

Clearly, even though the α parameter is being dropped here by letting it equal one, the effect of path length is still be being expressed in the path weights given in $\boldsymbol{B}^{-1}$. Specifically, if weights are in $(0, 1]$ paths of greater length will be increasingly down-weighted in the calculation of alpha-centrality, and will be further decreased if one specifies $\alpha \in (0, 1]$, because the initial weights will be multiplied by a proportion. Conversely, if weights are greater than one, longer path lengths will be exponentially up-weighted/emphasized when calculating alpha-centrality!!

References

Aho, K., Kriloff, C., Godsey, S.E., Ramos, R., Wheeler, C., You, Y., Warix, S., Derryberry, D., Zipper, S., Hale, R.L. *et al.* (2023) Non-perennial stream networks as directed acyclic graphs: The r-package streamdag. *Environmental Modelling & Software*, **167**, 105775.

Bonacich, P. & Lloyd, P. (2001) Eigenvector-like measures of centrality for asymmetric relations. *Social networks*, **23**, 191–201.

Katz, L. (1953) A new status index derived from sociometric analysis. *Psychometrika*, **18**, 39–43.

Newman, M. (2018) *Networks*. Oxford university press.