

MultilayerGraphs.jl: Multilayer Network Science in Julia

Claudio Moroni ^{1,2*} and Pietro Monticone ^{1,2*}

¹ University of Turin, Italy ² Interdisciplinary Physics Team, Italy * These authors contributed equally.

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Open Journals](#) 

Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

MultilayerGraphs.jl is a Julia package for the creation, manipulation and analysis of the structure, dynamics and functions of multilayer graphs.

A multilayer graph is a graph consisting of multiple standard subgraphs called *layers* which can be interconnected through [bipartite graphs](#) called *interlayers* composed of the vertex sets of two different layers and the edges between them. The vertices in each layer represent a single set of nodes, although not all nodes have to be represented in every layer.

Formally, a multilayer graph can be defined as a triple $G = (V, E, L)$, where:

- V is the set of vertices;
- E is the set of edges, pairs of nodes (u, v) representing a connection, relationship or interaction between the nodes u and v ;
- L is a set of layers, which are subsets of V and E encoding the nodes and edges within each layer.

Each layer ℓ in L is a tuple (V_ℓ, E_ℓ) , where V_ℓ is a subset of V that represents the vertices within that layer, and E_ℓ is a subset of E that represents the edges within that layer.

MultilayerGraphs.jl is an integral part of the [JuliaGraphs](#) ecosystem extending Graphs.jl ([Fairbanks et al., 2021](#)) so all the methods and metrics exported by Graphs.jl work for multilayer graphs, but due to the special nature of multilayer graphs the package features a peculiar implementation that maps a standard integer-labelled vertex representation to a more user-friendly framework exporting all the objects an experienced practitioner would expect such as nodes (`Node`), vertices (`MultilayerVertex`), layers (`Layer`), interlayers (`Interlayer`), etc.

MultilayerGraphs.jl features multilayer-specific methods and metrics including the global clustering coefficient, the overlay clustering coefficient, the multilayer eigenvector centrality, the multilayer modularity and the Von Neumann entropy.

Finally, MultilayerGraphs.jl has been integrated within the [JuliaDynamics](#) ecosystem so that any `Multilayer(Di)Graph` can be utilised as an argument to the `GraphSpace` constructor in `Agents.jl` ([Datseris et al., 2022](#)).

Statement of Need

Several theoretical frameworks have been proposed to formally subsume all instances of multilayer graphs ([Aleta & Moreno, 2019](#); [Artime et al., 2022](#); [Bianconi, 2018](#); [Boccaletti et al., 2014](#); [Cozzo et al., 2018](#); [M. D. Domenico et al., 2013](#); [M. D. Domenico, 2022](#); [Kivela et al., 2014](#); [Lee et al., 2015](#)).

Multilayer graphs have been adopted to model the structure and dynamics of a wide spectrum of high-dimensional, non-linear, multi-scale, time-dependent complex systems including physi-

cal, chemical, biological, neuronal, socio-technical, epidemiological, ecological and economic networks (Aleta et al., 2022, 2020; Amato et al., 2017; Arruda et al., 2017; Azimi-Tafreshi, 2016; Baggio et al., 2016; Buldú & Porter, 2018; Cozzo et al., 2013; M. D. Domenico, 2017; M. D. Domenico et al., 2016; Estrada & Gómez-Gardeñes, 2014; Gosak et al., 2018; Granell et al., 2013; Lim et al., 2019; Mangioni et al., 2020; Massaro & Bagnoli, 2014; Pilosof et al., 2017; Soriano-Paños et al., 2018; Timóteo et al., 2018).

We have chosen the [Julia language](#) for this software package because it is a modern, open-source, high-level, high-performance dynamic language for technical computing (Bezanson et al., 2017). At the best of our knowledge there are currently no software packages dedicated to the creation, manipulation and analysis of multilayer graphs implemented in the Julia language apart from MultilayerGraphs.jl itself (Moroni & Monticone, 2022).

Main Features

The two main data structures are `MultilayerGraph` and `MultilayerDiGraph`: collections of layers connected through interlayers.

The **vertices** of a multilayer graph are representations of one set of distinct objects called **Nodes**. Each layer may represent all the node set or just a subset of it. The vertices of `Multilayer(Di)Graph` are implemented via the `MultilayerVertex` custom type. Each `MultilayerVertex` encodes information about the node it represents, the layer it belongs to and its metadata.

Both the **intra-layer** and **inter-layer edges** are embedded in the `MultilayerEdge` struct, whose arguments are the two connected multilayer vertices, the edge weight and its metadata. It's important to highlight that `Multilayer(Di)Graphs` are weighted and able to store metadata by default (i.e. they have been assigned the `IsWeighted` and `IsMeta` traits from [SimpleTraits.jl](#)).

The **layers** are implemented via the `Layer` struct composed of an underlying graph and a mapping from its integer-labelled vertices to the collection of `MultilayerVertex`s the layer represents. **Interlayers** are similarly implemented via the `Interlayer` mutable struct, and they are generally constructed by providing the two connected layers, the (multilayer) edge list between them and a graph. This usage of underlying graphs allows for an easier debugging procedure during construction and a more intuitive analysis afterwards allowing the package to leverage all the features of the JuliaGraphs ecosystem so that it can be effectively considered as a real proving ground of its internal consistency.

The `Multilayer(Di)Graph` structs are weighted and endowed with the functionality to store both vertex-level and edge-level metadata by default so that at any moment the user may add or remove a `Layer` or specify an `Interlayer` and since different layers and interlayers could be better represented by graphs that are weighted or unweighted and with or without metadata, it was crucial for us to provide the most general and adaptable structure. A `Multilayer(Di)Graph` is instantiated by providing the ordered list of layers and the list of interlayers to the constructor. The latter are automatically specified, so there is no need to instantiate all of them.

Alternatively, it is possible to construct a `Multilayer(Di)Graph` making use of a graph generator-like signature allowing the user to set the degree distribution or the degree sequence and employs graph realisation methods such as the Havel-Hakimi algorithm for undirected graphs (Hakimi, 1962) and the Kleitman-Wang algorithm for directed ones (Kleitman & Wang, 1973).

`Multilayer(Di)Graphs` structure may be represented via dedicated `WeightTensor`, `MetadataTensor` and `SupraWeightMatrix` structs, all of which support indexing with `MultilayerVertex`s. Once a `Multilayer(Di)Graph` has been instantiated, its layers and interlayers can be accessed as their properties.

87 Installation and Usage

88 To install MultilayerGraphs.jl it is sufficient to activate the pkg mode by pressing] in the Julia
89 REPL and then run the following command:

```
pkg> add MultilayerGraphs
```

90 In the following code chunks we synthetically illustrate how to define, handle and analyse a
91 MultilayerGraph in order to showcase some of the main features outlined in the previous
92 section.

```
using Revise
using Graphs, SimpleValueGraphs
using Distributions
using MultilayerGraphs

# The objects that `MultilayerVertex`s represent are `Node`s
const n_nodes = 100
const nodes = [Node("node_$(i)") for i in 1:n_nodes]

#= # Every layer and Interlayer that is involved in a `Multilayer(Di)Graph` must have the
const vertex_type = Int64
const weight_type = Float64 =#

# We will be instantiating layers and interlayers with randomly-chosen edges and vertices
## Here, we define a layer with an underlying simple directed graph, using a configuration
_nv = rand(1:100)
_layer_simpdigraph = layer_simpdigraph(
    :layer_simpdigraph, # The name of the layer_metadata
    sample(nodes, _nv, replace=false), # The nodes that the layer will use
    Truncated(Normal(5,5), 0, 20), # The distribution from which the weights are sampled
    Truncated(Normal(5,5), 0, 20) # The distribution from which the edge weights are sampled
)

## Next, we define a layer with an underlying simple weighted directed graph. This time
_nv = rand(1:n_nodes)
_ne = rand(_nv:(_nv*(_nv-1) - 1))
_layer_simpleweighteddigraph = layer_simpleweighteddigraph(
    :layer_simpleweighteddigraph,
    sample(nodes, _nv, replace=false),
    _ne; # The number of edges
    default_edge_weight = (src,dst) -> rand() # The function to generate edge weights
)

## Similar constructors, more flexible at the cost of ease of use, allows for finer tuning
### NB: This constructor should be necessary only in rare circumstances, where e.g. the
### the correct return types of the `default_vertex/edge_metadata`s, or to use underlying
_nv = rand(1:n_nodes)
_ne = rand(_nv:(_nv*(_nv-1) - 1))
default_vertex_metadata = v -> ("vertex_$(v)_metadata",)
default_edge_metadata = (s,d) -> (rand(),)
_layer_valdigraph = Layer(
    :layer_simplevaldigraph,
    sample(nodes, _nv, replace=false),
    _ne,
```

```

ValDiGraph(
    SimpleDiGraph{Int64}();
    vertexval_types = (String,),
    vertexval_init = default_vertex_metadata,
    edgeval_types = (Float64,),
    edgeval_init = default_edge_metadata
),
Float64;
default_vertex_metadata = default_vertex_metadata,
default_edge_metadata = default_edge_metadata
)

layers = [_layer_simpdigraph, _layer_simpleweighteddigraph, _layer_valdigraph]

# There are many more constructors that the user is encouraged to explore in the package

# We may now move to Interlayers. Note that, in order to define a `Multilayer(Di)Graph`,
# to be explicitly constructed by the user, since they are automatically specified by the
# Anyway, more complex interlayers need to be manually instantiated. The interface is ve

## Interlayer with an underlying simple directed graph and `_ne` edges
nv_1 = nv(_layer_simpdigraph)
nv_2 = nv(_layer_simpleweighteddigraph)
_ne = rand(1:(nv_1*nv_2 - 1)) # The interlayer is a bipartite graph between t
_interlayer_simpdigraph = interlayer_simpdigraph(
    _layer_simpdigraph, # One of the two layers
    _layer_simpleweighteddigraph, # One of the two layers
    _ne # The number of edges
)

## The interlayer exports a more flexible constructor too.
nv_1 = nv(_layer_simpdigraph)
nv_2 = nv(_layer_simpleweighteddigraph)
_ne = rand(1:(nv_1*nv_2 - 1))
_interlayer_metadigraph = interlayer_metadigraph(
    _layer_simpleweighteddigraph,
    _layer_valdigraph,
    _ne;
    default_edge_metadata = (src,dst) -> (edge_metadata = "metad",
    transfer_vertex_metadata = true # Whether to have the vertic
)

interlayers = [_interlayer_simpdigraph, _interlayer_metadigraph]

# Layers and Interlayers are not immutable, and mostly behave like normal graphs. The re
# A MultilayerDiGraph (i.e. a directed multilayer graph, following the naming convention

multilayerdigraph = MultilayerDiGraph(
    layers, # The (ordered) collection of layers that constitute t
    interlayers, # The manually-specified interlayers. The interlayers
    # (in this example, the interlayer between `_layer_sim
    default_interlayers_structure = "multiplex" # The automatically-sp
)

```

93 For a more comprehensive exploration of the package features and functionalities we strongly

94 recommend consulting the [tutorial](#) included in the package documentation.

95 Related Packages

96 R

97 Here is a list of software packages for the creation, manipulation, analysis and visualisation of
98 multilayer graphs implemented in the [R language](#):

- 99 ■ [muxViz](#) implements functions to perform multilayer correlation analysis, multilayer central-
100 ity analysis, multilayer community structure detection, multilayer structural reducibility,
101 multilayer motifs analysis and utilities to statically and dynamically visualise multilayer
102 graphs ([D. Domenico et al., 2014](#));
- 103 ■ [multinet](#) implements functions to import, export, create and manipulate multilayer
104 graphs, several state-of-the-art multiplex graph analysis algorithms for centrality measures,
105 layer comparison, community detection and visualization ([Magnani et al., 2021](#));
- 106 ■ [mully](#) implements functions to import, export, create, manipulate and merge multilayer
107 graphs and utilities to visualise multilayer graphs in 2D and 3D ([Hammoud & Kramer,
108 2018](#));
- 109 ■ [multinets](#) implements functions to import, export, create, manipulate multilayer graphs
110 and utilities to visualise multilayer graphs ([Lazega et al., 2008](#)).

111 Python

112 Here is a list of software packages for the creation, manipulation, analysis and visualisation of
113 multilayer graphs implemented in the [Python language](#):

- 114 ■ [MultiNetX](#) implements methods to create undirected networks with weighted or un-
115 weighted links, to analyse the spectral properties of adjacency or Laplacian matrices and
116 to visualise multilayer graphs and dynamical processes by coloring the nodes and links
117 accordingly;
- 118 ■ [PyMNet](#) implements data structures for multilayer graphs and multiplex graphs, methods
119 to import, export, create, manipulate multilayer graphs and for the rule-based generation
120 and lazy-evaluation of coupling edges and utilities to visualise multilayer graphs ([Kivela
121 et al., 2014](#)).

122 Acknowledgements

123 This open-source research software project received no financial support.

124 References

- 125 Aleta, A., Martín-Corral, D., Bakker, M. A., Piontti, A. P. y, Ajelli, M., Litvinova, M., Chinazzi,
126 M., Dean, N. E., Halloran, M. E., Longini, I. M., Pentland, A., Vespignani, A., Moreno, Y.,
127 & Moro, E. (2022). Quantifying the importance and location of SARS-CoV-2 transmission
128 events in large metropolitan areas. *Proceedings of the National Academy of Sciences*,
129 119(26). <https://doi.org/10.1073/pnas.2112182119>
- 130 Aleta, A., Martín-Corral, D., Piontti, A. P. y, Ajelli, M., Litvinova, M., Chinazzi, M., Dean,
131 N. E., Halloran, M. E., Jr, I. M. L., Merler, S., Pentland, A., Vespignani, A., Moro, E.,
132 & Moreno, Y. (2020). Modelling the impact of testing, contact tracing and household
133 quarantine on second waves of COVID-19. *Nature Human Behaviour*, 4(9), 964–971.
134 <https://doi.org/10.1038/s41562-020-0931-9>

- 135 Aleta, A., & Moreno, Y. (2019). Multilayer networks in a nutshell. *Annual Re-*
 136 *view of Condensed Matter Physics*, 10(1), 45–62. [https://doi.org/10.1146/](https://doi.org/10.1146/annurev-conmatphys-031218-013259)
 137 [annurev-conmatphys-031218-013259](https://doi.org/10.1146/annurev-conmatphys-031218-013259)
- 138 Amato, R., Díaz-Guilera, A., & Kleineberg, K.-K. (2017). Interplay between social influence
 139 and competitive strategical games in multiplex networks. *Scientific Reports*, 7(1). <https://doi.org/10.1038/s41598-017-06933-2>
 140 <https://doi.org/10.1038/s41598-017-06933-2>
- 141 Arruda, G. F. de, Cozzo, E., Peixoto, T. P., Rodrigues, F. A., & Moreno, Y. (2017). Disease
 142 localization in multilayer networks. *Physical Review X*, 7(1). [https://doi.org/10.1103/](https://doi.org/10.1103/physrevx.7.011014)
 143 [physrevx.7.011014](https://doi.org/10.1103/physrevx.7.011014)
- 144 Artime, O., Benigni, B., Bertagnolli, G., dAndrea, V., Gallotti, R., Ghavasieh, A., Raimondo,
 145 S., & Domenico, M. D. (2022). *Multilayer network science*. Cambridge University Press.
 146 <https://doi.org/10.1017/9781009085809>
- 147 Azimi-Tafreshi, N. (2016). Cooperative epidemics on multiplex networks. *Physical Review E*,
 148 93(4). <https://doi.org/10.1103/physreve.93.042303>
- 149 Baggio, J. A., BurnSilver, S. B., Arenas, A., Magdanz, J. S., Kofinas, G. P., & Domenico, M.
 150 D. (2016). Multiplex social ecological network analysis reveals how social changes affect
 151 community robustness more than resource depletion. *Proceedings of the National Academy*
 152 *of Sciences*, 113(48), 13708–13713. <https://doi.org/10.1073/pnas.1604401113>
- 153 Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to
 154 numerical computing. *SIAM Review*, 59(1), 65–98. <https://doi.org/10.1137/141000671>
- 155 Bianconi, G. (2018). *Multilayer networks*. Oxford University Press. [https://doi.org/10.1093/](https://doi.org/10.1093/oso/9780198753919.001.0001)
 156 [oso/9780198753919.001.0001](https://doi.org/10.1093/oso/9780198753919.001.0001)
- 157 Boccaletti, S., Bianconi, G., Criado, R., Genio, C. I. del, Gómez-Gardeñes, J., Romance, M.,
 158 Sendiña-Nadal, I., Wang, Z., & Zanin, M. (2014). The structure and dynamics of multilayer
 159 networks. *Physics Reports*, 544(1), 1–122. <https://doi.org/10.1016/j.physrep.2014.07.001>
- 160 Buldú, J. M., & Porter, M. A. (2018). Frequency-based brain networks: From a multiplex
 161 framework to a full multilayer description. *Network Neuroscience*, 2(4), 418–441. https://doi.org/10.1162/netn_a_00033
 162 https://doi.org/10.1162/netn_a_00033
- 163 Cozzo, E., Arruda, G. F. de, Rodrigues, F. A., & Moreno, Y. (2018). *Multiplex networks*.
 164 Springer International Publishing. <https://doi.org/10.1007/978-3-319-92255-3>
- 165 Cozzo, E., Baños, R. A., Meloni, S., & Moreno, Y. (2013). Contact-based social contagion in
 166 multiplex networks. *Physical Review E*, 88(5). <https://doi.org/10.1103/physreve.88.050801>
- 167 Datseris, G., Vahdati, A. R., & DuBois, T. C. (2022). Agents.jl: A performant and
 168 feature-full agent-based modeling software of minimal code complexity. *SIMULATION*,
 169 003754972110688. <https://doi.org/10.1177/00375497211068820>
- 170 Domenico, D., Porter, & Arenas. (2014). MuxViz: A tool for multilayer analysis and
 171 visualization of networks. *Journal of Complex Networks*, 3(2), 159–176. [https://doi.org/10.1093/](https://doi.org/10.1093/comnet/cnu038)
 172 [comnet/cnu038](https://doi.org/10.1093/comnet/cnu038)
- 173 Domenico, M. D. (2017). Multilayer modeling and analysis of human brain networks. *Giga-*
 174 *Science*, 6(5). <https://doi.org/10.1093/gigascience/gix004>
- 175 Domenico, M. D. (2022). *Multilayer networks: Analysis and visualization*. Springer International
 176 Publishing. <https://doi.org/10.1007/978-3-030-75718-2>
- 177 Domenico, M. D., Granell, C., Porter, M. A., & Arenas, A. (2016). The physics of spreading
 178 processes in multilayer networks. *Nature Physics*, 12(10), 901–906. [https://doi.org/10.](https://doi.org/10.1038/nphys3865)
 179 [1038/nphys3865](https://doi.org/10.1038/nphys3865)

- 180 Domenico, M. D., Solé-Ribalta, A., Cozzo, E., Kivelä, M., Moreno, Y., Porter, M. A., Gómez,
181 S., & Arenas, A. (2013). Mathematical formulation of multilayer networks. *Physical Review*
182 *X*, 3(4). <https://doi.org/10.1103/physrevx.3.041022>
- 183 Estrada, E., & Gómez-Gardeñes, J. (2014). Communicability reveals a transition to coordinated
184 behavior in multiplex networks. *Physical Review E*, 89(4). [https://doi.org/10.1103/](https://doi.org/10.1103/physreve.89.042819)
185 [physreve.89.042819](https://doi.org/10.1103/physreve.89.042819)
- 186 Fairbanks, J., Besançon, M., Simon, S., Hoffiman, J., Eubank, N., & Karpinski, S. (2021).
187 *JuliaGraphs/graphs.jl: An optimized graphs package for the julia programming language.*
188 <https://github.com/JuliaGraphs/Graphs.jl/>
- 189 Gosak, M., Markovič, R., Dolenšek, J., Rupnik, M. S., Marhl, M., Stožer, A., & Perc, M.
190 (2018). Network science of biological systems at different scales: A review. *Physics of Life*
191 *Reviews*, 24, 118–135. <https://doi.org/10.1016/j.pprev.2017.11.003>
- 192 Granell, C., Gómez, S., & Arenas, A. (2013). Dynamical interplay between awareness and
193 epidemic spreading in multiplex networks. *Physical Review Letters*, 111(12). <https://doi.org/10.1103/physrevlett.111.128701>
- 194 <https://doi.org/10.1103/physrevlett.111.128701>
- 195 Hakimi, S. L. (1962). On realizability of a set of integers as degrees of the vertices of a linear
196 graph. i. *Journal of the Society for Industrial and Applied Mathematics*, 10(3), 496–506.
197 <https://doi.org/10.1137/0110037>
- 198 Hammoud, Z., & Kramer, F. (2018). Mully: An r package to create, modify and visualize
199 multilayered graphs. *Genes*, 9(11), 519. <https://doi.org/10.3390/genes9110519>
- 200 Kivela, M., Arenas, A., Barthélemy, M., Gleeson, J. P., Moreno, Y., & Porter, M. A. (2014).
201 Multilayer networks. *Journal of Complex Networks*, 2(3), 203–271. [https://doi.org/10.](https://doi.org/10.1093/comnet/cnu016)
202 [1093/comnet/cnu016](https://doi.org/10.1093/comnet/cnu016)
- 203 Kleitman, D. J., & Wang, D. L. (1973). Algorithms for constructing graphs and digraphs with
204 given valences and factors. *Discrete Mathematics*, 6(1), 79–88. [https://doi.org/10.1016/](https://doi.org/10.1016/0012-365x(73)90037-x)
205 [0012-365x\(73\)90037-x](https://doi.org/10.1016/0012-365x(73)90037-x)
- 206 Lazega, E., Jourda, M.-T., Mounier, L., & Stofer, R. (2008). Catching up with big fish in
207 the big pond? Multi-level network analysis through linked design. *Social Networks*, 30(2),
208 159–176. <https://doi.org/10.1016/j.socnet.2008.02.001>
- 209 Lee, K.-M., Min, B., & Goh, K.-I. (2015). Towards real-world complexity: An introduction to
210 multiplex networks. *The European Physical Journal B*, 88(2). [https://doi.org/10.1140/](https://doi.org/10.1140/epjb/e2015-50742-1)
211 [epjb/e2015-50742-1](https://doi.org/10.1140/epjb/e2015-50742-1)
- 212 Lim, S., Radicchi, F., Heuvel, M. P. van den, & Sporns, O. (2019). Discordant attributes of
213 structural and functional brain connectivity in a two-layer multiplex network. *Scientific*
214 *Reports*, 9(1). <https://doi.org/10.1038/s41598-019-39243-w>
- 215 Magnani, M., Rossi, L., & Vega, D. (2021). Analysis of multiplex social networks with r.
216 *Journal of Statistical Software*, 98(8). <https://doi.org/10.18637/jss.v098.i08>
- 217 Mangioni, G., Jurman, G., & Domenico, M. D. (2020). Multilayer flows in molecular networks
218 identify biological modules in the human proteome. *IEEE Transactions on Network Science*
219 *and Engineering*, 7(1), 411–420. <https://doi.org/10.1109/tNSE.2018.2871726>
- 220 Massaro, E., & Bagnoli, F. (2014). Epidemic spreading and risk perception in multiplex
221 networks: A self-organized percolation method. *Physical Review E*, 90(5). [https://doi.](https://doi.org/10.1103/physreve.90.052817)
222 [org/10.1103/physreve.90.052817](https://doi.org/10.1103/physreve.90.052817)
- 223 Moroni, C., & Monticone, P. (2022). *MultilayerGraphs.jl: A julia package for the creation,*
224 *manipulation and analysis of the structure, dynamics and functions of multilayer graphs.*
225 University of Turin (UniTO); Interdisciplinary Physics Team (InPhyT). [https://doi.org/10.](https://doi.org/10.5281/zenodo.7009172)
226 [5281/zenodo.7009172](https://doi.org/10.5281/zenodo.7009172)

- 227 Pilosof, S., Porter, M. A., Pascual, M., & Kéfi, S. (2017). The multilayer nature of eco-
228 logical networks. *Nature Ecology & Evolution*, 1(4). [https://doi.org/10.1038/](https://doi.org/10.1038/s41559-017-0101)
229 [s41559-017-0101](https://doi.org/10.1038/s41559-017-0101)
- 230 Soriano-Paños, D., Lotero, L., Arenas, A., & Gómez-Gardeñes, J. (2018). Spreading processes
231 in multiplex metapopulations containing different mobility networks. *Physical Review X*,
232 8(3). <https://doi.org/10.1103/physrevx.8.031039>
- 233 Timóteo, S., Correia, M., Rodríguez-Echeverría, S., Freitas, H., & Heleno, R. (2018). Multilayer
234 networks reveal the spatial structure of seed-dispersal interactions across the great rift
235 landscapes. *Nature Communications*, 9(1). <https://doi.org/10.1038/s41467-017-02658-y>

DRAFT