Director

Katherine Icay

December 18, 2015

Abstract

Director is an R package designed to streamline the visualization of multiple levels of interacting RNA-seq data. It utilizes a modified Sankey plugin of the JavaScript library D3 to provide a fast and easy, web-based solution to discovering potentially interesting downstream effects of regulatory and/or co-expressed molecules. The diagrams are dynamic, interactive, and packaged as HTML files – making them highly portable and eliminating the need for third-party software. This enables a straightforward approach for scientists to interpret the data produced, and bioinformatics developers an alternative means to present relevant data.

Contents

1 Sankey diagram

1.1 What is it?

Sankey diagrams are types of flow diagrams for path analytics, where paths represent a distinct sequence of events. They are particularly useful for ientifying the contributions of these events in a directional cascade. This package applies Sankey diagrams to represent the effect of specific molecules in a regulatory cascade.

Dynamic visualization of data is achieved with the Sankey plugin of Mike Bostock's JavaScript library, D3, modified with user-defined parameters to incorporate *two* (versus the traditional one)

set of quantitative values. *Director* thus makes it possible to assign specific colours to paths and to nodes, and to customize the look of the diagrams produced.

1.2 Limitations

Sankey diagrams are strictly directional flow diagrams and are not suitable for depicting feedback loops (i.e. a path from one node that leads back to itself).

1.3 Diagram features

The advantage of an HTML output is that it encourages user interaction with the data. Quantitative and qualitative descriptions about each node and path are easily referenced with mouseover. Nodes are dynamic and can be moved around the space for optimal presentation. Furthermore, entire pathways can be highlighted in the overall diagram by clicking on the node (or nodes) of interest.

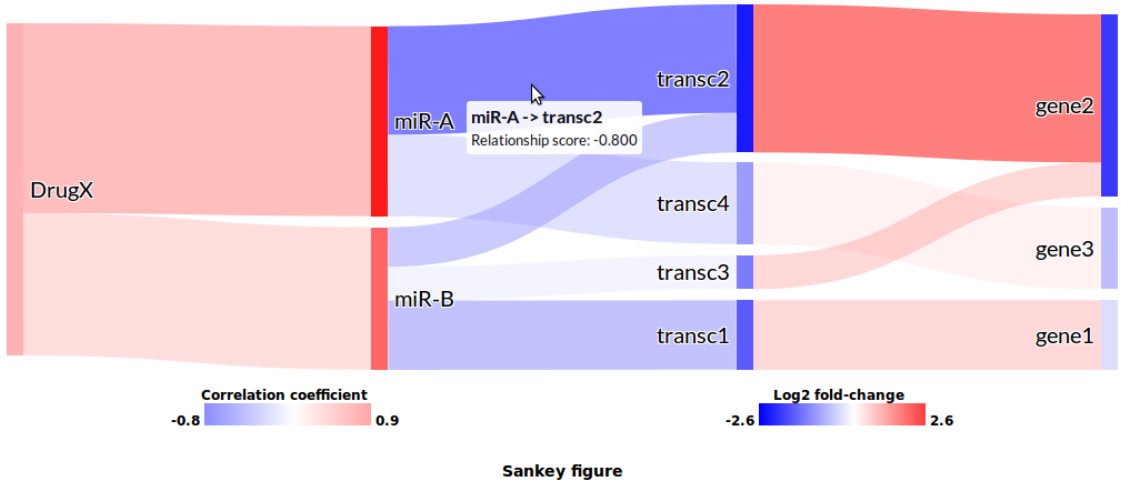

Figure 1: Sankey plot with mouseover information and connected paths to *transc2* highlighted.

2 Standard workflow

2.1 Getting started

First, a working directory must be specified and initialized with *initSankey*. This writes the necessary scripts enabling dynamic HTML and specifies the basic look of the figures. Specifically, the D3 JavaScript library (currently, version 3.5.6) and Mike Bostock's D3 plugin for Sankey. The function requires no inputs but optional parameters can be used to define an absolute path (default is the working directory obtained from *getwd*()), alternate download links to necessary scripts and download methods, path opacity, font, and font sizes.

Running *initSankey* in the working directory produces the following files and folders:

```
/
├── css
│   └── sankey.css
└── js
    ├── d3.min.js
    └── d3.v3.js
```

```
    ├── LICENSE
    ├── sankey.js
├── layouts
    ├── charts.html
├── config.yml
```

These files are necessary for functions *makeSankey* and *drawSankey* to work so it must always be executed first. The parameters defined in *initSankey* will apply to all HTML files written to the specified directory. Rerunning this function with different configurations will, therefore, affect the look of all HTML files within the directory (see section 4.2.1 and Figure 4).

2.2 Input data

Director requires two types of quantitive information as input: values for *nodes* and values for *paths*. The package assumes nodes are molecules of interest and that the quantitative value is the feature which makes them interesting, e.g. expression fold-change, significance value, or methylation score. Similarly, paths represent a predictive or quantitative measure of the interaction between molecules (e.g. correlation, target prediction score).

Node and path values can be input in two ways.

1. As a single table with each row defining a source node, target node, relationship value, node values, and (optional) additional information.

2. As two separate tables: one with each row defining a source node, target node, relationship value, and (optional) additional values; and another with two columns defining all nodes and corresponding node values.

Suppose we have the following information:

source	target	impact score	source fold-change	target fold-change
DrugX	miR-B	0.6	1	1.5
DrugX	miR-A	0.8	1	2.6
miR-B	transc1	-0.7	1.5	-1.7
miR-B	transc2	-0.4	1.5	-2.6
miR-B	transc3	-0.34	1.5	-1.5
miR-A	transc2	-0.8	2.6	-2.6
miR-A	transc4	-0.6	2.6	-1.2
trasc1	gene1	0.8	-1.7	-0.6
transc2	gene2	0.9	-2.6	-2.2
transc3	gene2	0.88	-1.5	-2.2
transc4	gene3	0.6	-1.2	-0.7

We define information for a Sankey diagram with the function *createList* which takes as input either a single table:

```
> library(Director)
> table1 <- data.frame(source=c("DrugX", "DrugX", "miR-B", "miR-B", "miR-B",
+          "miR-A", "miR-A", "transc1", "transc2", "transc3", "transc4"),
+     target=c("miR-B", "miR-A", "transc1", "transc2", "transc3", "transc2",
+          "transc4", "gene1", "gene2", "gene2", "gene3"),
+     value=c(0.6, 0.8, -0.7, -0.4, -0.34, -0.8, -0.6, 0.8, 0.9, 0.88, 0.6),
+     sourcefc=c(1,1,1.5,1.5,1.5, 2.6,2.6,-1.7,-2.6,-1.5,-1.2),
+     targetfc=c(1.5,2.6,-1.7,-2.6,-1.5,-2.6,-1.2,-0.6,-2.2,-2.2,-0.7),
+     stringsAsFactors=FALSE)

> tempList <- createList(table1)
```

```
    source  target description value sourcefc targetfc
1    DrugX   miR-B              0.60      1.0      1.5
2    DrugX   miR-A              0.80      1.0      2.6
3    miR-B transc1             -0.70      1.5     -1.7
4    miR-B transc2             -0.40      1.5     -2.6
5    miR-B transc3             -0.34      1.5     -1.5
6    miR-A transc2             -0.80      2.6     -2.6
7    miR-A transc4             -0.60      2.6     -1.2
8  transc1   gene1              0.80     -1.7     -0.6
9  transc2   gene2              0.90     -2.6     -2.2
10 transc3   gene2              0.88     -1.5     -2.2
11 transc4   gene3              0.60     -1.2     -0.7
```

Or two separate tables:

```
> List <- data.frame(source=c("DrugX", "DrugX",
+         "miR-B", "miR-B", "miR-B", "miR-A", "miR-A", "transc1",
+         "transc2", "transc3", "transc4"),
+     target=c("miR-B", "miR-A", "transc1", "transc2", "transc3",
+         "transc2", "transc4", "gene1", "gene2", "gene2", "gene3"),
+     value=c(0.6, 0.8, -0.7, -0.4, -0.34, -0.8, -0.6, 0.8, 0.9, 0.88, 0.6),
+                 stringsAsFactors=FALSE)
> ListFC <- data.frame(genes=c("DrugX", "miR-B", "miR-A", "transc1",
+     "transc2", "transc3", "transc4", "gene1", "gene2", "gene3"),
+     foldChange=c(1, 1.5, 2.6, -1.7, -2.6, -1.5, -1.2, -0.6, -2.2, -0.7),
+         stringsAsFactors=FALSE)

> tempList2 <- createList(List,ListFC)

    source  target description value sourcefc targetfc
1    DrugX   miR-B              0.60      1.0      1.5
2    DrugX   miR-A              0.80      1.0      2.6
3    miR-B transc1             -0.70      1.5     -1.7
4    miR-B transc2             -0.40      1.5     -2.6
5    miR-B transc3             -0.34      1.5     -1.5
6    miR-A transc2             -0.80      2.6     -2.6
7    miR-A transc4             -0.60      2.6     -1.2
8  transc1   gene1              0.80     -1.7     -0.6
9  transc2   gene2              0.90     -2.6     -2.2
10 transc3   gene2              0.88     -1.5     -2.2
11 transc4   gene3              0.60     -1.2     -0.7
```

Note that *description* is a required column to make and draw a Sankey diagram, but column content is optional and will be empty if not specified. It can be used, for example, to provide the gene name of a target transcript or additional information about the source and target interaction. Descriptions are accessed via mouseover of the respective paths.

3 Data filtering

Director has several functions that attempt to streamline the visualization process. For useful and efficient visualization, data should first be filtered for the most valuable, user-defined set of information.

3.1 What's being visualized?

Hundreds and thousands of molecules can be detected in a single living cell at any given time. Molecules that have mRNA content sequenced and quantified are considered 'expressed', but are not necessarily interesting. In fact, depending on the questions a user wishes to answer, most of the expressed molecules are noise and the potentially interesting molecules difficult to distinguish.

Basic functions are provided with *Director* to filter the list of source and target nodes for:

- $filterNumeric$ Minimum, maximum, or absolute numerical values.
- $filterSubset$ User-defined qualitative values grep'd from up to two qualitative columns (e.g. source and target).
- $filterRelation$ Positively correlated or inversely related source-target pair node or path values.

Multiple lists can be created representing different levels of source-target relationships and combined with $append2List$. The package then visually organizes downstream effects to several biologically functional levels. See the package manual for more details and examples for each function.

4 Visual parameters

After filtering, the ready list is input to $makeSankey$. This function calculates and assigns drawing values for nodes, assigns colours to nodes and colours to paths. $drawSankey$ then takes these values to create the HTML containing the interactive Sankey diagram. These HTML files can then be saved to the working directory with $writeSankey$, where the supporting files generated with $initSankey$ are already saved.

4.1 Nodes and paths

The primary function of the package is to make a custom Sankey diagram that takes in additional source and target values (*sourcefc* and *targetfc*, respectively) on top of the traditional *source, target, value* to label nodes and determine path widths. These additional source and target values enable colouring of the nodes to quantitative measures.

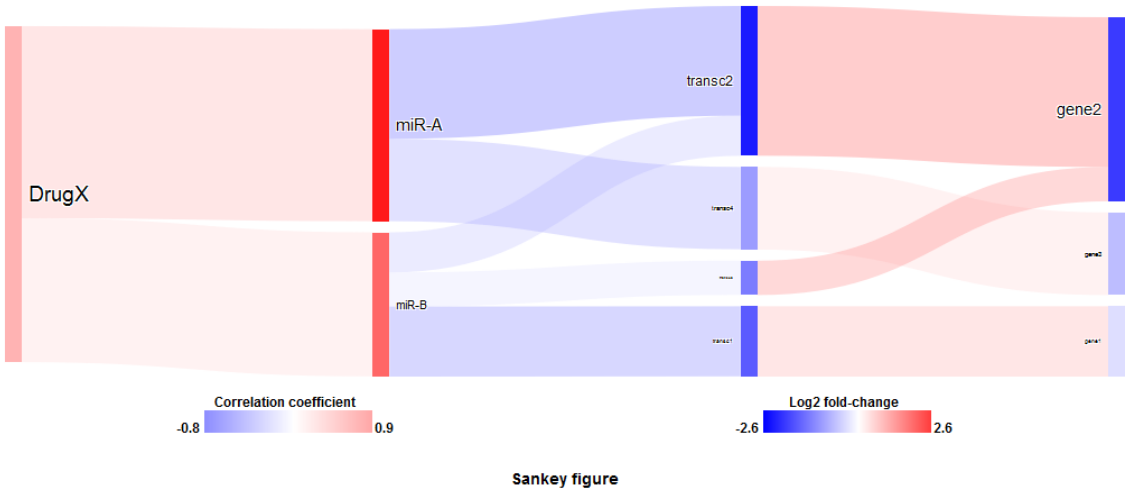

Figure 2: Sankey plot

Before drawing the diagram, colour scales must first be assigned to the node and path values.

```
> initSankey()
> sankey <- makeSankey(tempList2)
> drawSankey(sankey)
```

By default, the nodes and paths are coloured using a blue-red palette. However, users can individually define the node and the path colour scales in the function, as well as the 'zero' or *nought* value of the colour scales. This is useful when highlighting smaller versus larger values (e.g. p-values), and customizing the colours used in the figure to match a theme. Furthermore, both names and hex codes can be input as colour values.

4.1.1 *averagePath* parameter

What if there is no unique path values after the first level of interacting nodes? What if interest is only in the downstream effect of the first level of interactions (e.g. drugs and their molecular targets)? *makeSankey* has a Boolean parameter called *averagePath* that, when set to *TRUE*, will replace the *List$value* of a source-target pair of molecules (say, miRNA-1 and target Gene A) with an average value of the incoming paths to the source molecule (say, Drug X and Drug Y targeting miRNA-1). Assuming a uniform downstream effect, this colours downstream paths in the figure based on the average effect of the initial level of source-target values. Consequently, this gives more emphasis to the quantitative node values.

```
> initSankey()
> sankey2 <- makeSankey(tempList2, averagePath=TRUE)
> drawSankey(sankey2)
```

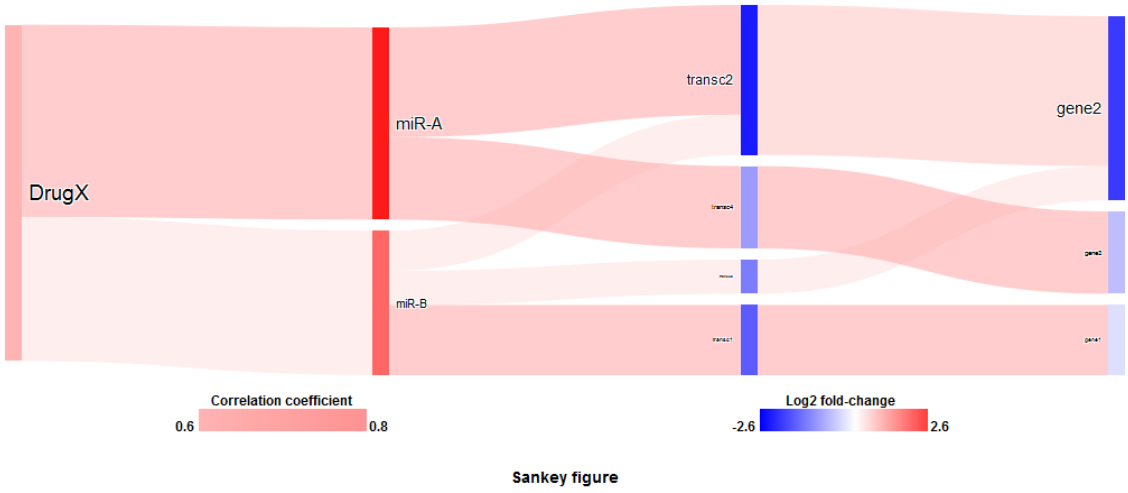

Figure 3: Sankey plot with averagePath enabled.

Note the value differences with and without averagePath enabled.

```
> truevalue <-  sankey$reference[,"truevalue"]

> cbind(sankey2$reference[,c("source", "target","averagePath_value")],truevalue)

    source  target averagePath_value truevalue
1    DrugX   miR-B               0.6      0.60
2    DrugX   miR-A               0.8      0.80
```

```
3    miR-B transc1              0.6     -0.70
4    miR-B transc2              0.6     -0.40
5    miR-B transc3              0.6     -0.34
6    miR-A transc2              0.8     -0.80
7    miR-A transc4              0.8     -0.60
8  transc1   gene1              0.6      0.80
9  transc2   gene2              0.7      0.90
10 transc3   gene2              0.6      0.88
11 transc4   gene3              0.8      0.60
```

4.2 Figure dimensions and properties

Each figure includes a legend containing the colour scales for node values and for path values. *drawSankey* parameters allow customization of the legend font, font size, legend text, and figure caption.

By default, figures are drawn with a 1000px width and height adjusted to the number of nodes/unique molecules up to 1800px (or approximately 300 nodes). If the figure fails to appear in the HTML file with *drawSankey* defaults, the options are either to

1. Define *drawSankey* parameters *height* > 1800px and *width* > 1000px.
2. Use *Director* filters to reduce the number of nodes to draw.

The goal of this package is to highlight key molecules and regulatory interactions in a select pathway. Having more molecules than necessary makes the figure 'noisy'.

Fortunately, users can view quantitative and qualitative information, including optional descriptions, directly from nodes and paths in the diagram via mouseover. This is useful to determine, for example, what quantitative thresholds to filter for to remove uninteresting paths from the figure.

4.2.1 Figure modifications

When any part of the List information is modified, *makeSankey* needs to be rerun to update colour assignments (quantitative values), nodes, and path connections. Changes to the general appearance of the figure can be made with parameters in *initSankey* and *drawSankey*. Details are available in the manual. To update the figure with the changes, simply rerun *drawSankey*. Figure 4 is drawn with the exact same List information as Figure 3 but using non-default parameters.

```
> initSankey(pathOpacity = 0.3, pathHover = 0.6,
+     fontsizeProportion = FALSE, fontsize=20)
> # Makes the path colours darker by 10%  and uses a single, uniform font size
> # for the figure rather than font sizes proportional to node size.
>
> sankey2 <- makeSankey(tempList2, averagePath=TRUE,
+     nodeMin = "orange", nodeMax = "purple", pathMax = "#333333")
> # Changes the default colours used.
>

> drawSankey(sankey2, caption = "Alternate figure",
+     nodeValue = "Node colour scale", pathValue = "Path colour scale",
+     legendsize = 14, height = 300)
> # Changes to legend values and overall figure size
```

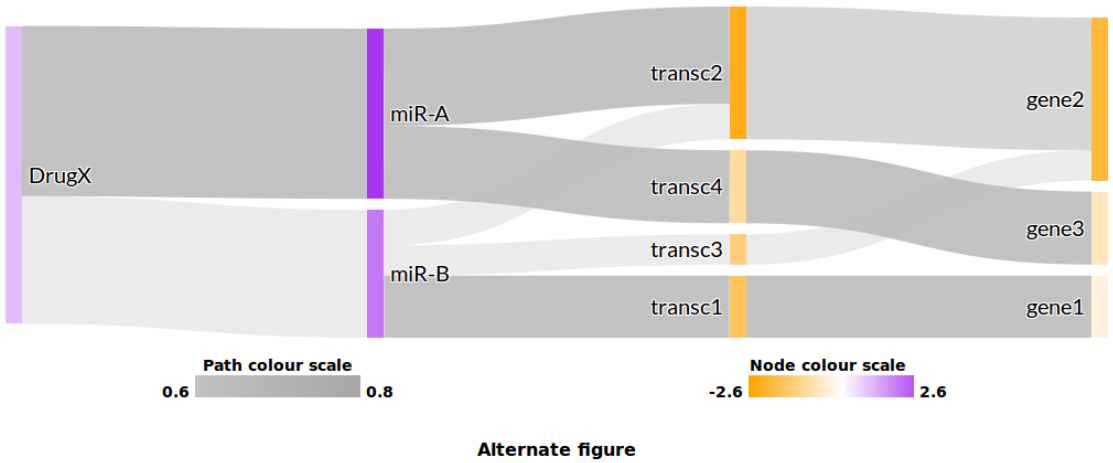

Figure 4: Sankey plot with averagePath enabled and non-default parameters.

5 Saving data

The relevant outputs generated are:

1. The List created with *createList* used as input for the main Sankey functions.
2. The Sankey diagram produced with *drawSankey*.

The former is a standard R data frame and can be saved with normal methods, e.g. *write.table*(). This simplifies later usage as it is already in the appropriate List format and can immediately be used with filtering and Sankey functions.

The latter can be saved to file using the function *writeSankey*. HTML has the dual benefit of preserving information dynamics of the diagram and vector graphics quality. As a result, adjusting the flow of the diagram and saving as a high-quality PDF (or .png) in any dimension is simple.