Genome Visualization with Circos

Session 4 — Self Study – Yeast Genome Comparison

Martin Krzywinski
Genome Sciences Centre
100-570 West 7th Ave
Vancouver BC V5Z 4S6 Canada
1-604-877-6000 x 673262
martink@bcgsc.ca
http://mkweb.bcgsc.ca

Circos
http://circos.ca

Course Materials
http://circos.ca/documentation/course

Genome Sciences Center
http://bcgsc.ca

Version History
v0.15 7 Jun 2017
v0.14 3 May 2016
v0.13 12 May 2014
v0.12 6 May 2014
v0.11 29 Jun 2011
v0.10 28 Jun 2011

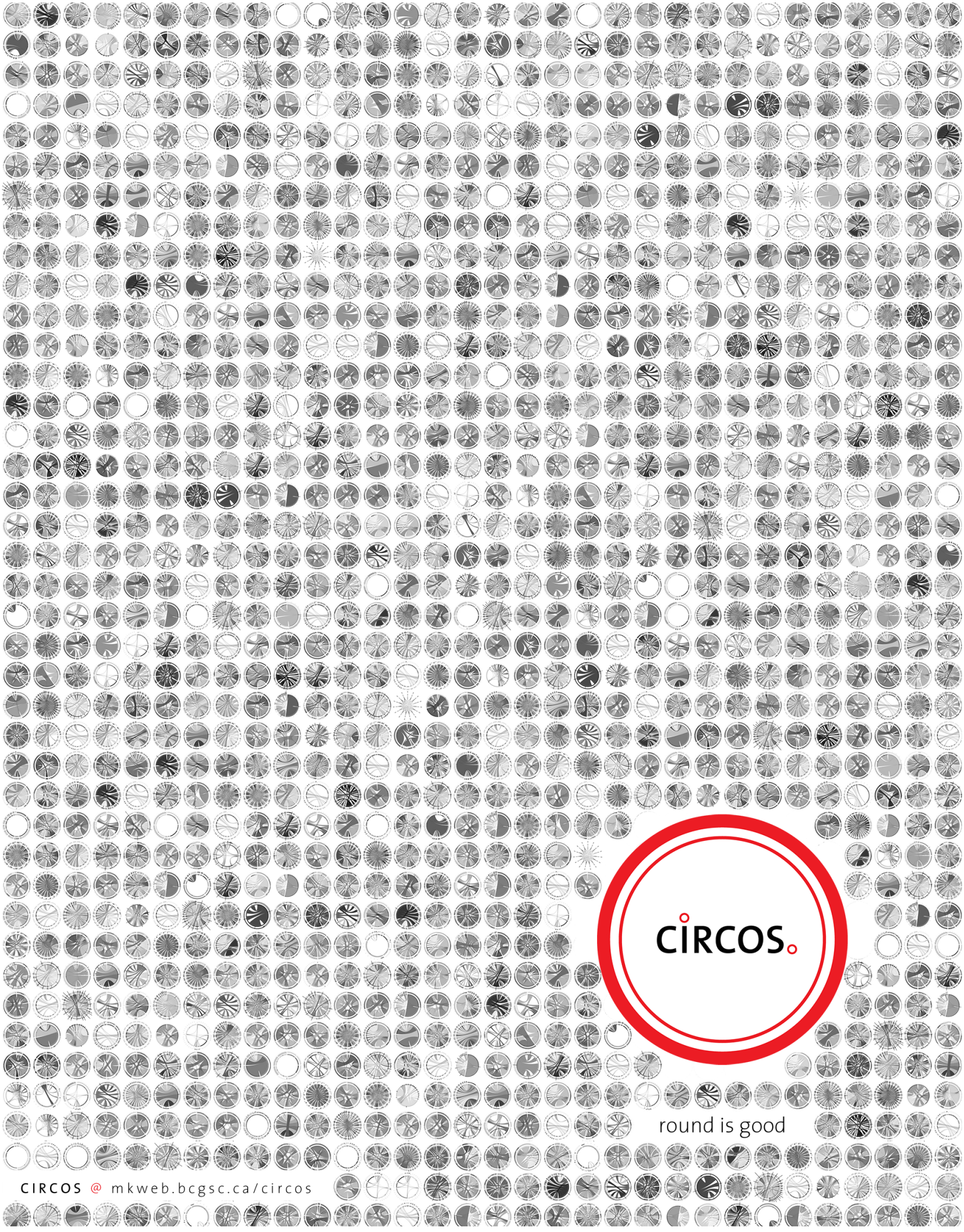
CIRCOS.
round is good
CIRCOS @ mkweb.bcgsc.ca/circos

Table of Contents

Self Study Example

In this session you will work on your own to create several images based on the comparison of yeast species (*Candida glabrata* (CAGL), *Zygosaccharomyces rouxii* (ZYRO), *Saccharomyces cerevisiae* (SACE)) you performed in a previous day of the course. Over the course of this example, you will explore the contents of Circos files, generate configuration files for ideogram layout and plot links.

Each of the four lessons in this session focuses on different aspects of displaying links and applying rules to how they are drawn.

Each lesson starts with template files. Your goal will be to extend the template to draw each of the images shown in Figure 1.

It's more important that you understand everything that you do than that you do everything.

INPUT FILES

You were previously asked to generate the data files for this example. If you were successful – good work! I have included versions of these files in `4/data` for you to use.

SOLUTIONS

The solutions—your own answers may vary—are provided in the directories `4/*.solutions`.

USING ONLINE TUTORIALS

To help you with configuration syntax, make use of the online Circos tutorials

http://circos.ca/documentation/tutorials

and be sure to make use of the section about errors in and debugging in the Session 1 Preamble handout.

IN CASE OF PANIC

Stay calm.

These lessons aren't meant to be trivial. But they're also not meant to be impossible. If you can't figure something out (but try first!), just ask.

I encourage you to work with your neighbor.

And if everything is very easy for you and you're bored then look for bugs in the source code.

Lesson 1 – Ideogram and Tick Layout

You'll find the `circos.conf` template for this lesson in `4/1/etc`. Parts to create a complete image are already defined for you. The template lists the steps you need to complete to create images shown in Figure 3, Figure 4 and Figure 5.

```
karyotype = ../data/CIRCOS/Karyotype.tab

chromosomes_display_default = yes
chromosomes_units = 1000

# 1.
# define colors for ideograms
# sace - green
# cagl - orange
# zyro - blue

# 2.
# show only cagl ideograms

# 3.
# show only cagl-l and cagl-m, each occupying 50% of image

<<include ideogram.conf>>

<<include ../../etc/ticks.conf>>
<<include ../../etc/image.conf>>
<<include etc/colors_fonts_patterns.conf>>
<<include etc/housekeeping.conf>>
```

You'll recognize all the elements in the template from previous sessions.

CAUTION Don't forget that Circos is case sensitive—all parameters are lowercase.

YEAST KARYOTYPES

You created the karyotype file that defines the names and lengths of the chromosomes of the 3 species of yeast.

```
chr - sace-a  saceA  0 230218 black
chr - sace-b  saceB  0 813184 black
...
chr - sace-p  saceP  0 948066 black
chr - cagl-a  caglA  0 491328 black
chr - cagl-b  caglB  0 502101 black
...
chr - cagl-m  caglM  0 1402899 black
chr - zyro-a  zyroA  0 1114666 black
chr - zyro-b  zyroB  0 1388208 black
...
chr - zyro-g  zyroG  0 1865392 black
```

Another option would be to store each of the species karyotypes in a separate file (e.g. `karyotype.sace.txt`, `karyotype.cagl.txt`, `karyotype.zyro.txt`) and use them as a list

```
karyotype = karyotype.sace.txt, karyotype.cagl.txt, karyotype.zyro.txt
```

What would be the benefits of doing this?

DEFINING COLORS—FIGURE 3

To change the colors of the ideograms, you can use the `chromosomes_color` parameter. Another way is to change the color value in the file used for the karyotype.

Try defining a new color, named the same as the genome (e.g. `sace`)

```
<colors>
sace = …
cagl = …
zyro = …
</colors>
```

This block will be merged with the default color definitions imported from `etc/colors_fonts_patterns.conf`. You can use either existing color names in the definitions (e.g. `blue`) or pick you own RGB value. I strongly recommend using existing color definitions and Brewer palettes. The qualitative Brewer palettes are useful for this

By having a color name the same as a genome name, you'll be able to refer to it more naturally in other parts of the configuration file because it can be derived from the ideogram name (e.g. `sace-a`).

FILTERING IDEOGRAMS—FIGURE 4

Use the chromosomes parameter to limit the ideograms drawn to those from CAGL. There are two ways of doing this. You can explicitly exclude all other ideograms or set `chromosomes_display_default=no` and explicitly include CAGL ideograms.

The names of the ideograms for CAGL are `cagl-a`, `cagl-b`, … and so on. Make sure you are familiar with the karyotype file so that you know the names of the ideograms.

Remember to use a regular expression when including (or excluding) ideograms in the configuration file—do not list all the ideograms.

RESIZING IDEOGRAMS—FIGURE 5

Change the image to show only two CAGL ideograms (`cagl-l` and cagl-m). Here's where it's better to specify the ideograms you want to draw, not those you don't.

For resizing, use `chromosomes_scale`. To make both ideograms occupy ½ of the figure, you have two options. You can set one of them to occupy ½ of the figure using relative scale—the other one will be resized to fill available space. Alternatively, you can use the `rn` suffix to divide the ideograms evenly into a fraction of the image.

Lesson 2 – Links

Let's draw some links. You'll find that the template for this lesson already includes a blank `<link>` block, which includes a blank `<rules>` block. When editing the file be careful not to accidentally delete closing block tags, like `</rule>` or `</link>`.

```
karyotype = ../data/CIRCOS/Karyotype.tab

chromosomes_units           = 1000

# 1.
# use your solution from previous session to draw
# only the zyro genome with blue ideograms

<links>

<link>

# 2.
# Load links from ../data/CIRCOS/DUPLICATION/link_zyro_zyro
# Set radius to 0.95r, bezier_radius to 0r
# Use black with transparency.

<rules>
use = no
<rule>
# Hide links smaller than 4kb
</rule>
<rule>
# Color link by size using spectral-11-div palette.
# Map var(size1) from domain (4000,6000) onto color index range (1,11)
# Set z parameter to be the size of the link
</rule>
</rules>

</link>

</links>

<<include ideogram.conf>>

<<include ../../etc/ticks.conf>>
<<include ../../etc/image.conf>>
<<include etc/colors_fonts_patterns.conf>>
<<include etc/housekeeping.conf>>
```

LOADING LINKS FROM FILE—FIGURE 6

Links are defined in `<link>` files, which themselves are defined within a `<links>` block.

First, limit the ideograms to be displayed only to those from ZYRO. Load links from the `CIRCOS/DUPLICATION/link_zyro_zyro` file. Be careful when setting paths. If you specify a relative path (I suggest you avoid absolute paths to make the configurations portable), Circos will try different parent directories to find the file.

Set the link color to `black`, with transparency (e.g. `black_a5`). This image is a bit of a mess. There are about 15,000 links shown and it's hard to tell what's going on.

Remember, Circos will only draw data. It does not perform analysis. However, there are ways in which you can toggle the display of data and change how it is formatted to focus on what is important.

HIDING LINKS WITH RULES—FIGURE 7

Create a rule that hides links whose first coordinate span is less than 4kb.

You can access data element properties using `var()`. For example, `var(size1)` refers to the size of the first coordinate (start of link) and `var(size2)` refers to the second (end of link). From other sessions, recall that you can reference `var(start1)`, `var(end2)`, `var(chr)` and so on.

To hide a data point, set `show=no`. This is analogous to the use of `use=no` in `<plot>`, `<link>` and `<rule>` blocks, which can be used to turn them off.

COLORING LINKS WITH RULES—FIGURE 8

Add a second rule that changes the color of the link, as a function of the `var(size1)`.

We've done this before in other sessions. The approach is to set the color of the link with code that maps a value (e.g. size of link) to a different color index. First, use `eval()` to indicate that the value is meant to be interpreted as code (it will be passed to Perl's `eval` operator).

```
<rule>
...
color = eval(...)
...
</rule>
```

The use of `sprintf` is natural here, since we want to generate a string (color name) from a value.

```
sprint("spectral-11-div-%d_a2",...)
```

What's the role of `_a2` here?

You have a handout of all the Brewer palettes. The `spectral-11-div` palette is the diverging 11-color palette that looks like a rainbow. It's reasonably perceptually uniform, but if you are very keen on avoiding issues in interpretation caused by hue boundaries, I suggest the sequential palettes. Now might be a good time to think about palettes that are colorblind-safe—browse and explore these palettes at *www.colorbrewer.org*.

Map the link size, `var(size1)`, onto the color index range `[1,11]` using `remap_int`, which is a function defined in Circos to help you remap values. The syntax is

```
remap_int(x,min,max,range_min,range_max)
```

to map a number x from domain `[min,max]` onto range `[range_min,range_max]`. At the extremes, `x<min` → `range_min` and `x<min` → `range_min` .

Experiment with `[min,max]=[4000,6000]`. How would you pick these values? How would you explore the range and distribution of values in the input file at the command line?

Lesson 3 – Link Geometry

In this lesson we'll play with link geometry to distinguish between *intra*chromosomal (between the same chromosome) and *inter*chromosomal (between different chromosomes) links.

```
karyotype = ../data/CIRCOS/DUPLICATION/Karyotype.tab

chromosomes_units           = 1000

# change ideogram radius to 0.5r and label radius to 1.9r in ideogram.conf
# remember to use * suffix to indicate you want the parameter to be overridden

<<include ideogram.conf>>

# 1.
# draw cagl-k and cagl-m ideograms, color them grey
# make them each occupy 1/2 of image
# reverse orientation of cagl-m

<links>
<link>

# 2.
# load from ../data/CIRCOS/DUPLICATION/link_conf(genome)_conf(genome)
# set radius to 1r, bezier_radius to 0r, bezier_radius_purity to 0.5, crest to 0.5

<rules>
use = no

<rule>
# 3.
# for intrachromosomal links, map the absolute difference in their start positions
# in the range 0 to 1e6 onto bezier_radius range 1.25-6. Use remap()
</rule>

<rule>
# 4.
# assign color based on start position. use remap() to map start position 0-1e6
# onto color index 1-11 of spectral-11-div palette
#
# 5.
# change thickness based on start coordinate size
# (e.g. remap 1-5kb onto thickness 1-3)
#
# 6.
# change z parameter proportional to link start
#
</rule>
</rules>
</link>
</links>
```

FILTERING IDEOGRAMS—FIGURE 9

Draw `cagl-k` and `cagl-m` ideograms.

Reverse the orientation of `cagl-m` using the `chromosomes_reverse` parameter. Why is this helpful?

In `ideogram.conf`, change the radius of the ideograms (`radius`) to `0.5r` and their labels (`label_radius`) to `1.9r`. These parameters are already being included from the default file for this session, so you'll need to override them. Use the `*` suffix to do this (e.g. `radius*`). What happens if you forget to use `*`? Try it. Does the error make sense?

DRAWING LINKS—FIGURE 10

Draw links from the `link_cagl_cagl` file. Set the link geometry parameters as indicated in the configuration template.

Try different values of crest (e.g. 0, 0.25, 0.5, 1). Can you figure out what this parameter does?

USING RULES TO CHANGE LINK GEOMETRY—FIGURE 10

Previously, we've used rules to change link color. Now, let's look at how these can be used to change their geometry. We'll change the `bezier_radius` of intrachromosomal links to make them appear outside of the ideograms.

Write a rule that selects intrachromosomal links. For these links, `var(intrachr)` is true, so you can use this as a condition. Alternatively, you can also try `var(chr1) eq var(chr2)`. Why `eq` here instead of `==`?

Change the `bezier_radius` of the link. Use `remap()` to map the distance between the start and end of the link (assume this ranges from 0 to 1Mb—why?) onto the `bezier_radius` range of `[1.25,6]`. The goal will be to make links that connect more distant regions to reach further from the ideogram.

Finally, we'll be creating a second rule that we'll want to apply, so set the `flow` parameter to `continue` so that the rule testing doesn't short-circuit.

USING RULES TO CHANGE COLOR AND THICKNESS OF LINKS—FIGURE 11

We'll change the color of the links like before, but now we'll use the start position do determine the color of the link. The goal will be to have links starting at 0 to map onto the first color (index 1) and those starting at the end of the chromosome (~1Mb) to map onto the last color (index 11).

Use `remap_int()` to set the thickness (1-3) based on size (1000-5000). Use `var(size1)`.

Finally, set the `z` parameter to be proportional to link start position. What is the purpose of the z parameter? Why would you want to use it? Can you think of other situations where it would be helpful?

CHANGING GENOMES—FIGURE 12 (ADVANCED)

Instead of CAGL, create the image for SACE.

Since many variables use the string "`cagl`", you can change them all to "`sace`" by search-and-replace in a text editor. This isn't ideal however. Define a parameter, genome, in the root of `circos.conf`

```
genome=cagl
```

You can refer to this parameter anywhere in the configuration file using `conf(genome)`. Replace all instances of "`cagl`" with `conf(genome)`. For example,

```
file = ../data/CIRCOS/DUPLICATION/link_cagl_cagl
```

becomes

```
file = ../data/CIRCOS/DUPLICATION/link_conf(genome)_conf(genome)
```

You can change the value of a parameter on the command line using `-param`

```
> circos -param genome=cagl
> circos -param genome=sace
```

This allows you to draw images with data from different sources without editing the configuration file.

Now set `genome=zyro`. You'll see an error message. Does it make sense?

Lesson 4 – Focusing on Two Genomes

We'll continue with looking how rules work by drawing conservation links between the three genomes.

```
karyotype = ../data/CIRCOS/DUPLICATION/Karyotype.tab

chromosomes_units           = 1000

# 1.
#
# draw cagl-m, zyro-g and sace-f ideograms
# make them each occupy 1/3 of the image

<links>

<link>

# 2.
#
# collect 250 largest links links from each link_* file in CIRCOS/CONSERVATION
# into links.top250.txt
#
# draw them as black lines

<rules>
use           = no
flow          = continue
<rule>
# make links from cagl orange
</rule>
<rule>
# make links from sace green
</rule>
<rule>
# make links from zyro blue
</rule>
<rule>
# add _a4 to color name
</rule>
<rule>
# color red links that have start and end coordinate sizes larger than 5kb
</rule>
</rules>

</link>

</links>
```

FILTERING IDEOGRAMS—FIGURE 13

Draw `sace-f`, `cagl-m` and `zygo-g` ideograms, scaled equally to fill the full image.

DRAWING LINKS—FIGURE 14

There are 6 different lists of conservation alignments in `data/CIRCOS/CONSERVATION`, corresponding to each of the 6 possible pairs of the 3 genomes. In total there are over 100,000 links in these files, so we'll narrow this down to speed things up.

Write a script that extracts the 250 largest links (as measured by the size of the first coordinate span) from each of these files and store it in `links.top250.txt` in the same directory as the link files.

This is doable on the command line using `grep`, `awk`, `sort`, `head` and `cut` command line tools. Don't spend too much time on this step, though. If it's taking you more than 5-10 minutes, look at the `topN` script, which provides the answer. Make sure you understand how it works.

Draw the links from the `links.top250.txt` file.

COLORING LINKS BY SOURCE—FIGURE 15 AND FIGURE 16

Color the links from `cagl-m` to be the same color as the `cagl` ideograms. To check that a link originates from an ideogram use `from(RX)` where `RX` is a regular expression to select the ideograms. This is shown in Figure 15

Do the same for links from `sace-f` and `zyro-g`, coloring them by the respective colors of their ideograms. This is shown in Figure 16.

USING RULES TO ADD TRANSPARENCY AND SELECTIVELY CHANGE COLOR—FIGURE 17

After changing the colors of the links based on their source, write another rule that makes the color of the link transparent. You can do this by suffixing `_aN` (e.g. `_a4`) to the end of the color name.

Like before, use make use of `sprintf()` and `eval()` in the rule. You can access the link's color using `var(color)`. The goal of the format string in `sprintf` will be to suffix the color string with `_a4`.

```
color = eval(sprintf("...",var(color)))
```

Finally, write a rule that changes the color to red of links whose start and end coordinate sizes are both >5kb. Recall that you can combine conditions with `&&` (also `and`) or set up two conditions, which will automatically be combined in this way.

You are done!

◦

Figures

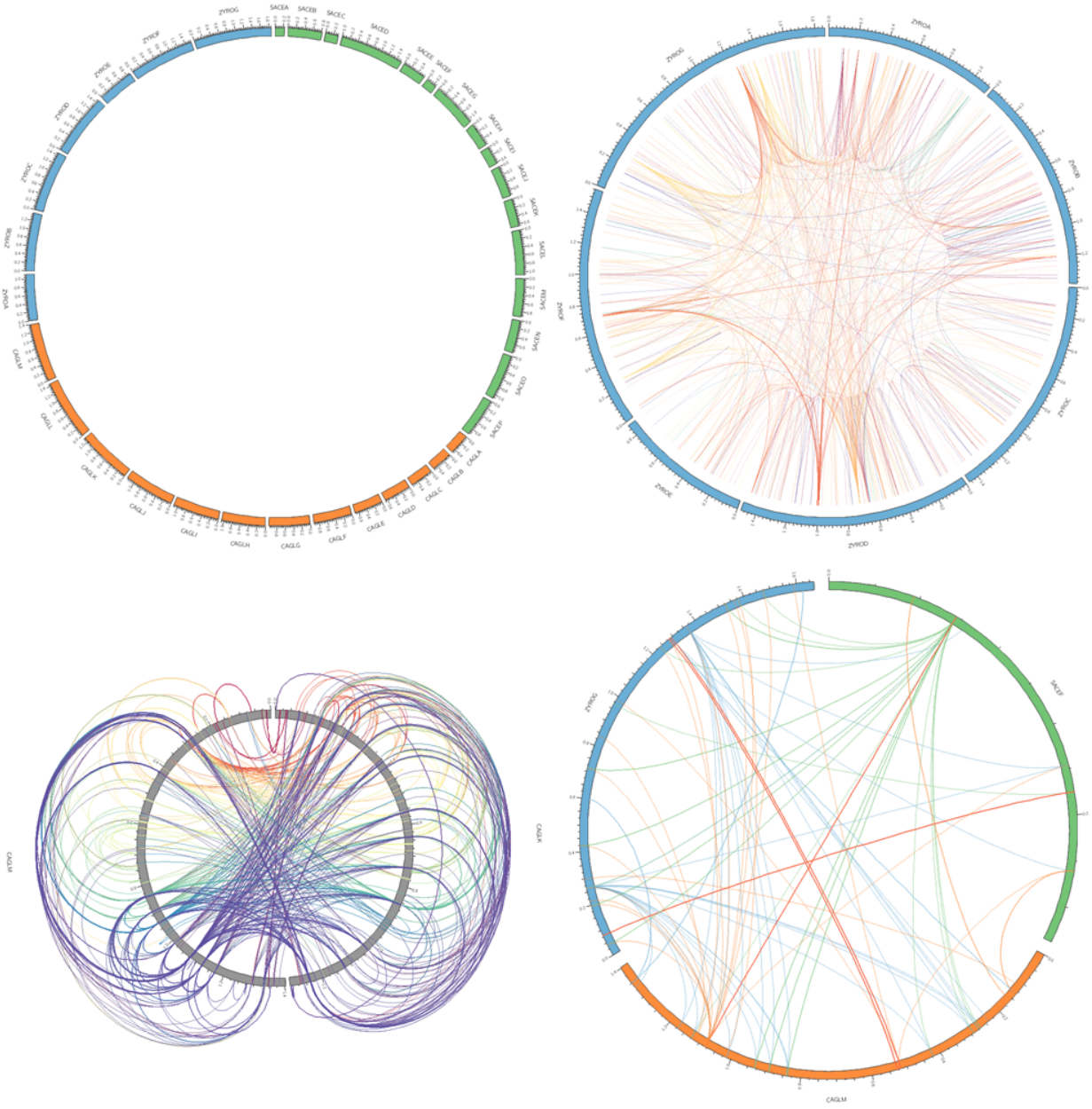

FIGURE 1

Four images you will create in this session using Yeast data from Fredj's Genome Comparison lectures..

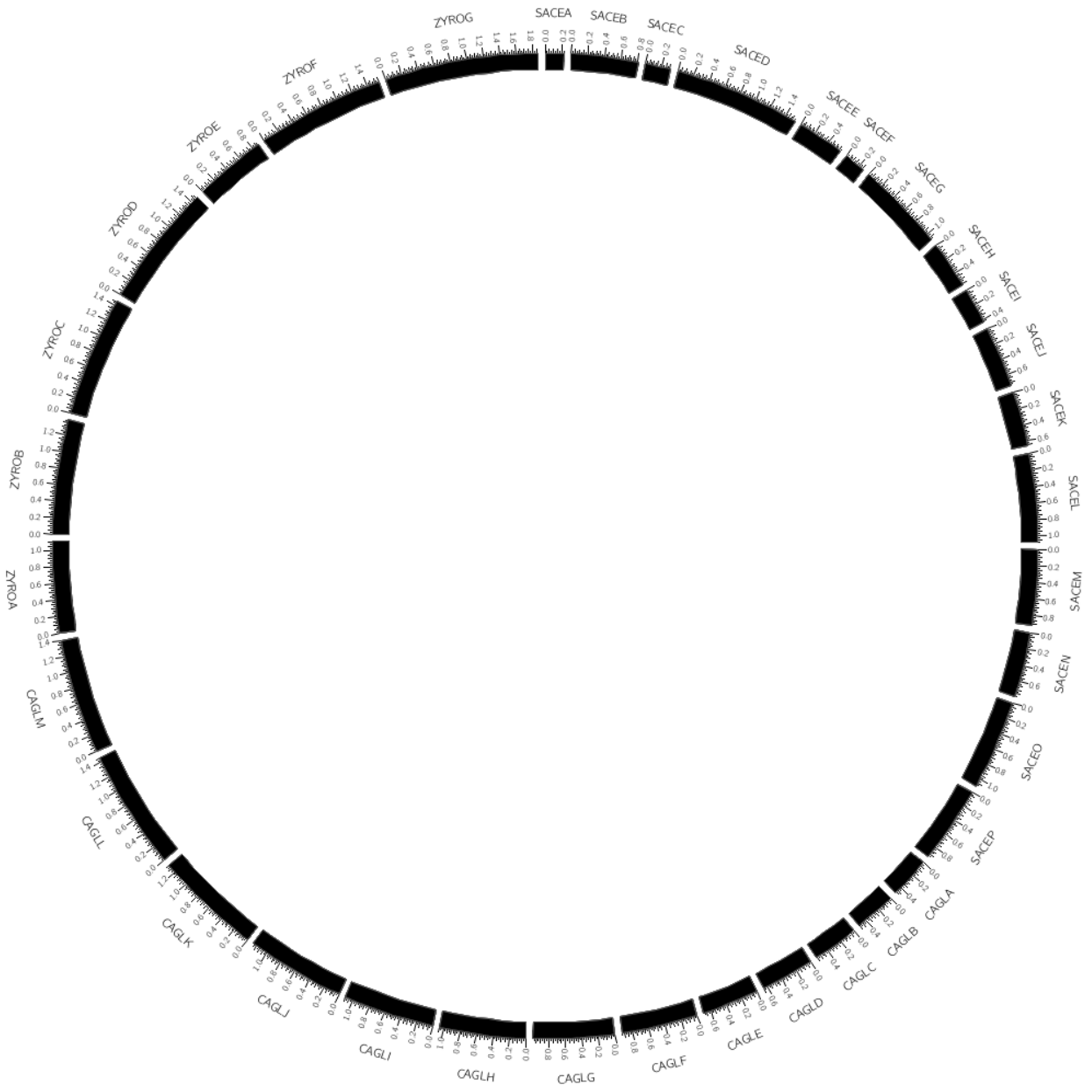

FIGURE 2

You will begin with this image, which shows all ideograms from SACE, CAGL and ZYRO genomes.

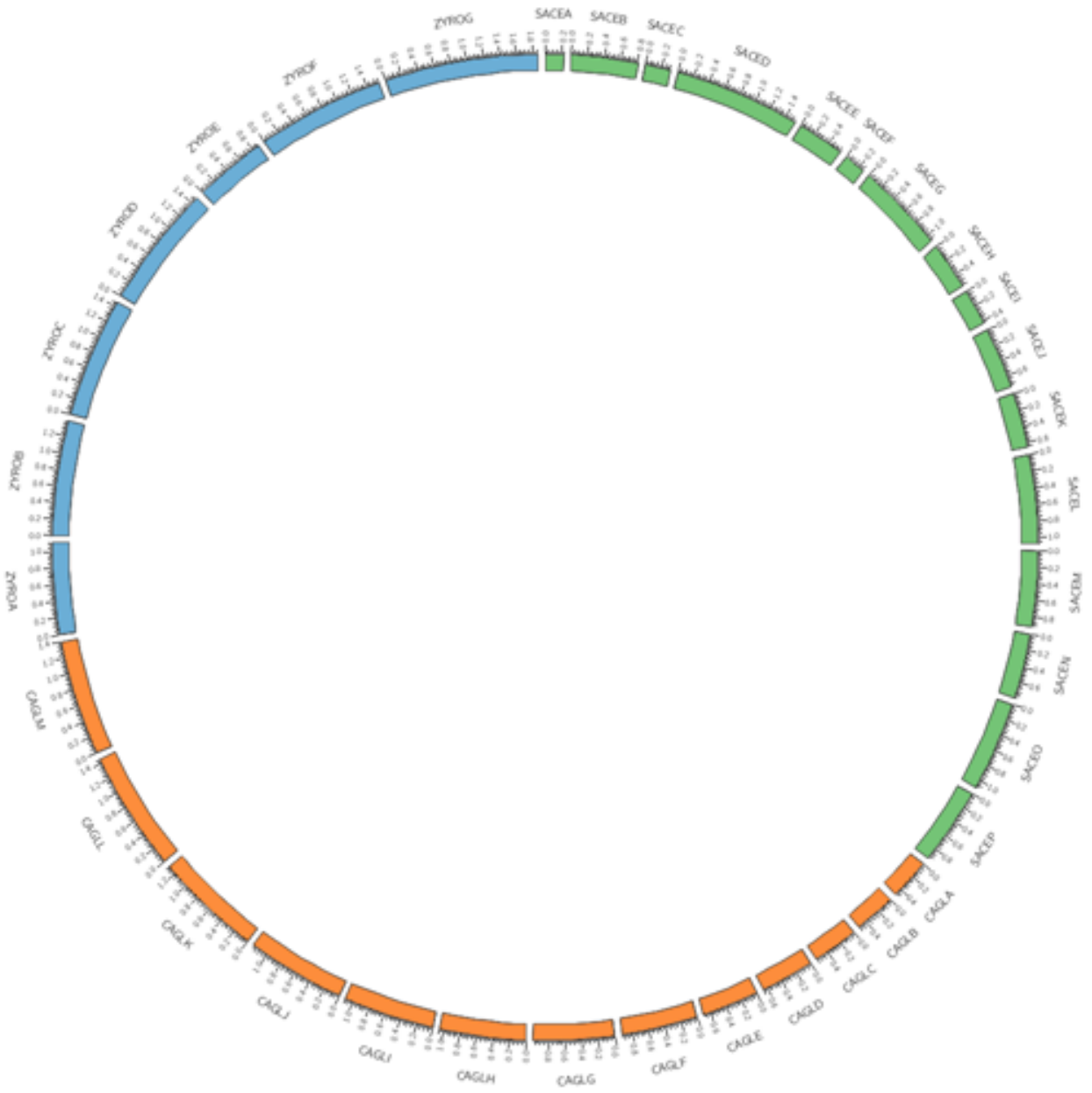

FIGURE 3

Color is applied to ideograms using `chromosomes_color`.

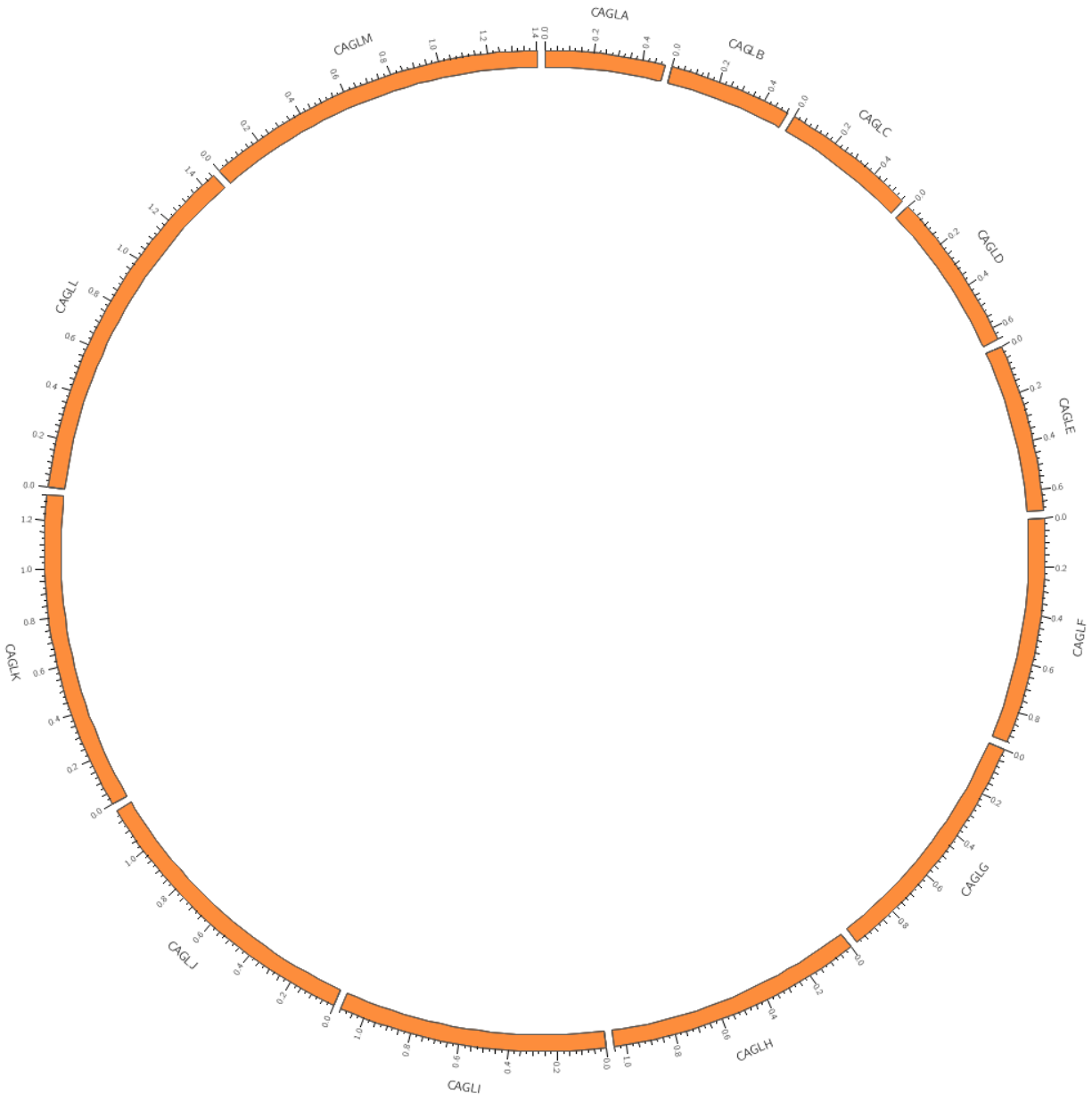

FIGURE 4

Ideograms filtered to show only those from CAGL. Ideograms can be excluded from the image by providing a list, each with a "-" prefix, using the `chromosomes` parameter.

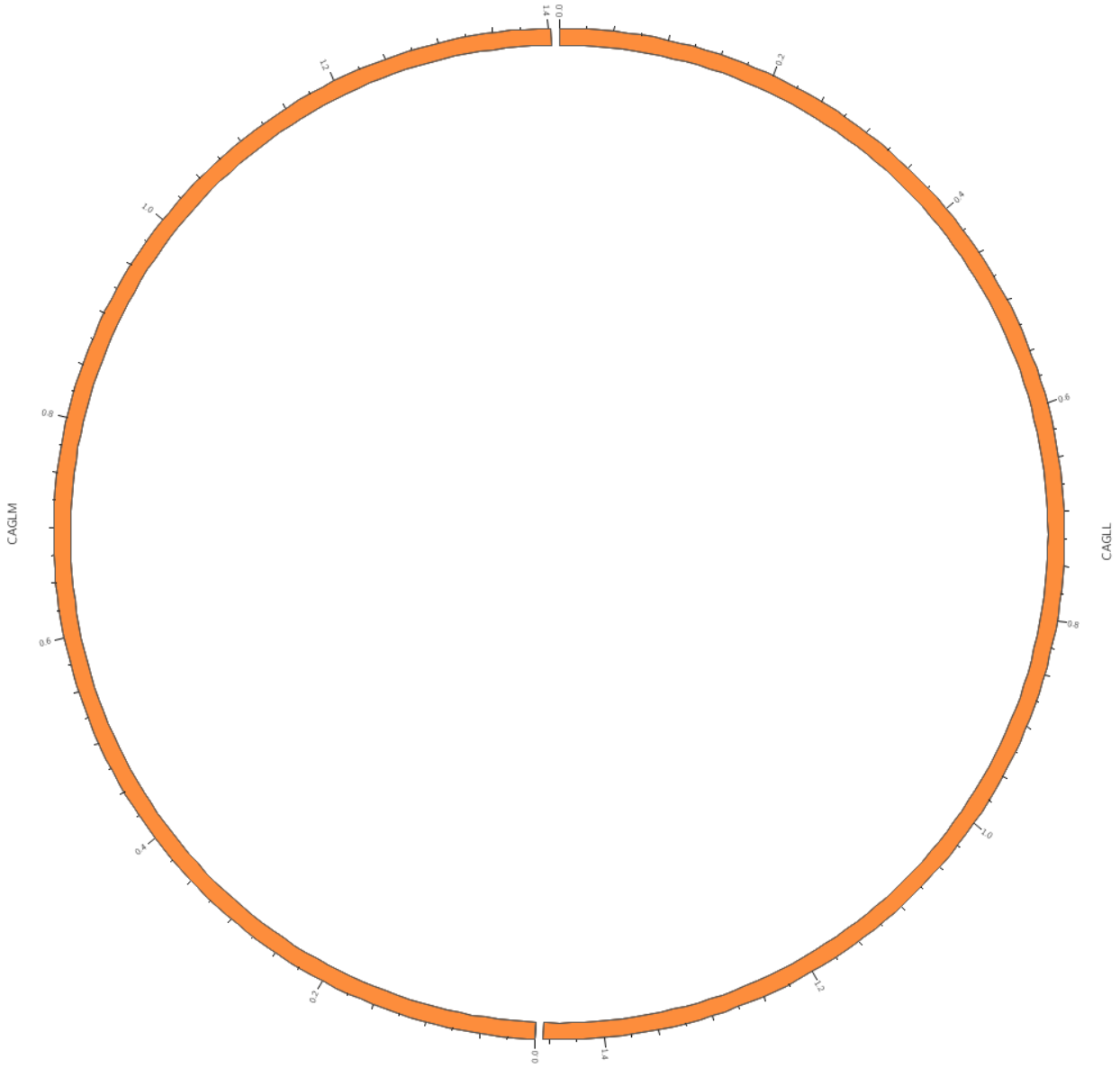

FIGURE 5

Two ideograms scaled to evenly fill the image.

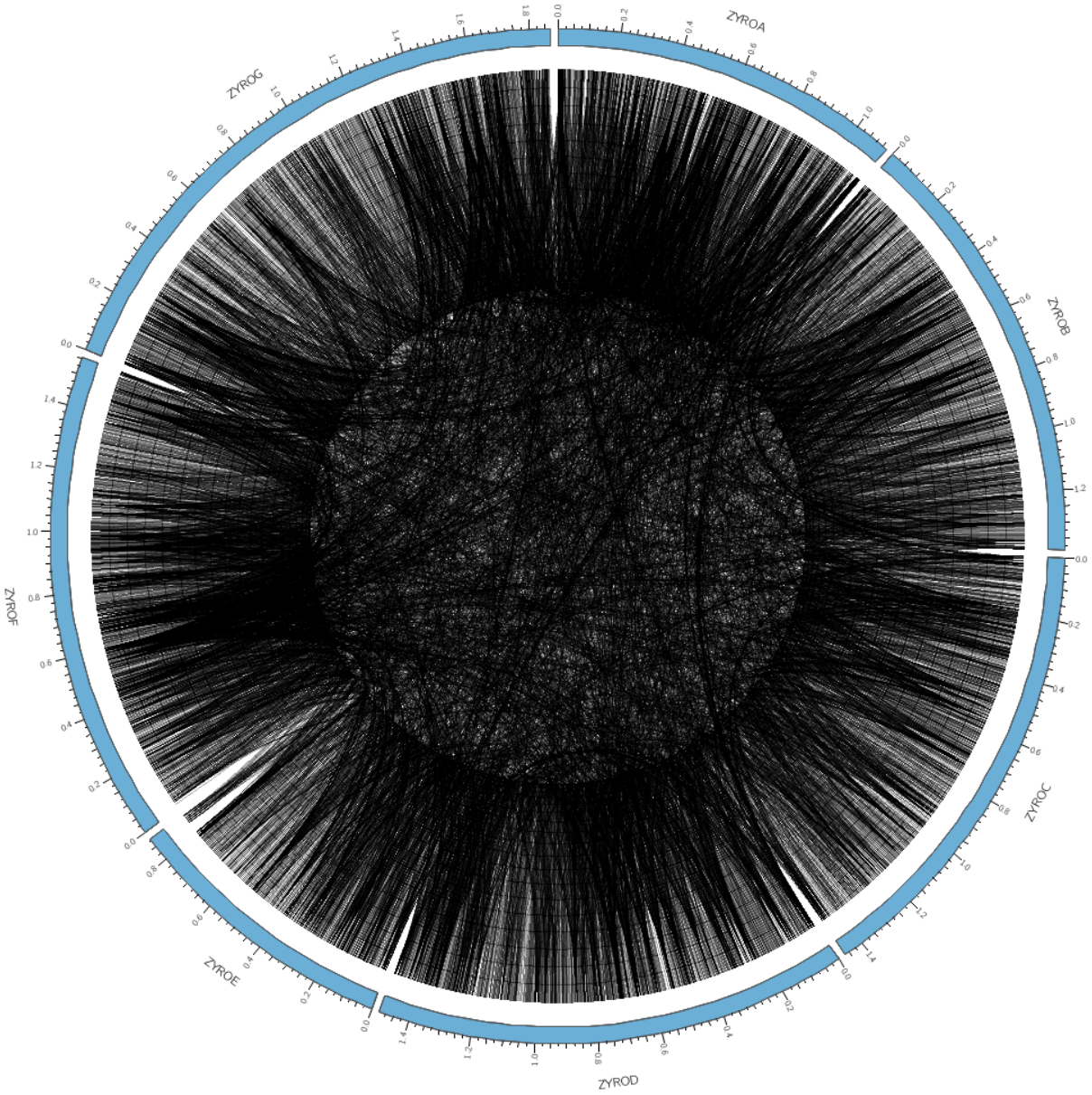

FIGURE 6

When drawing a lot of links, use transparency. In data set with a very large number of links (e.g. >1,000) it's a good idea to filter those that are less important, or otherwise reduce their visibility.

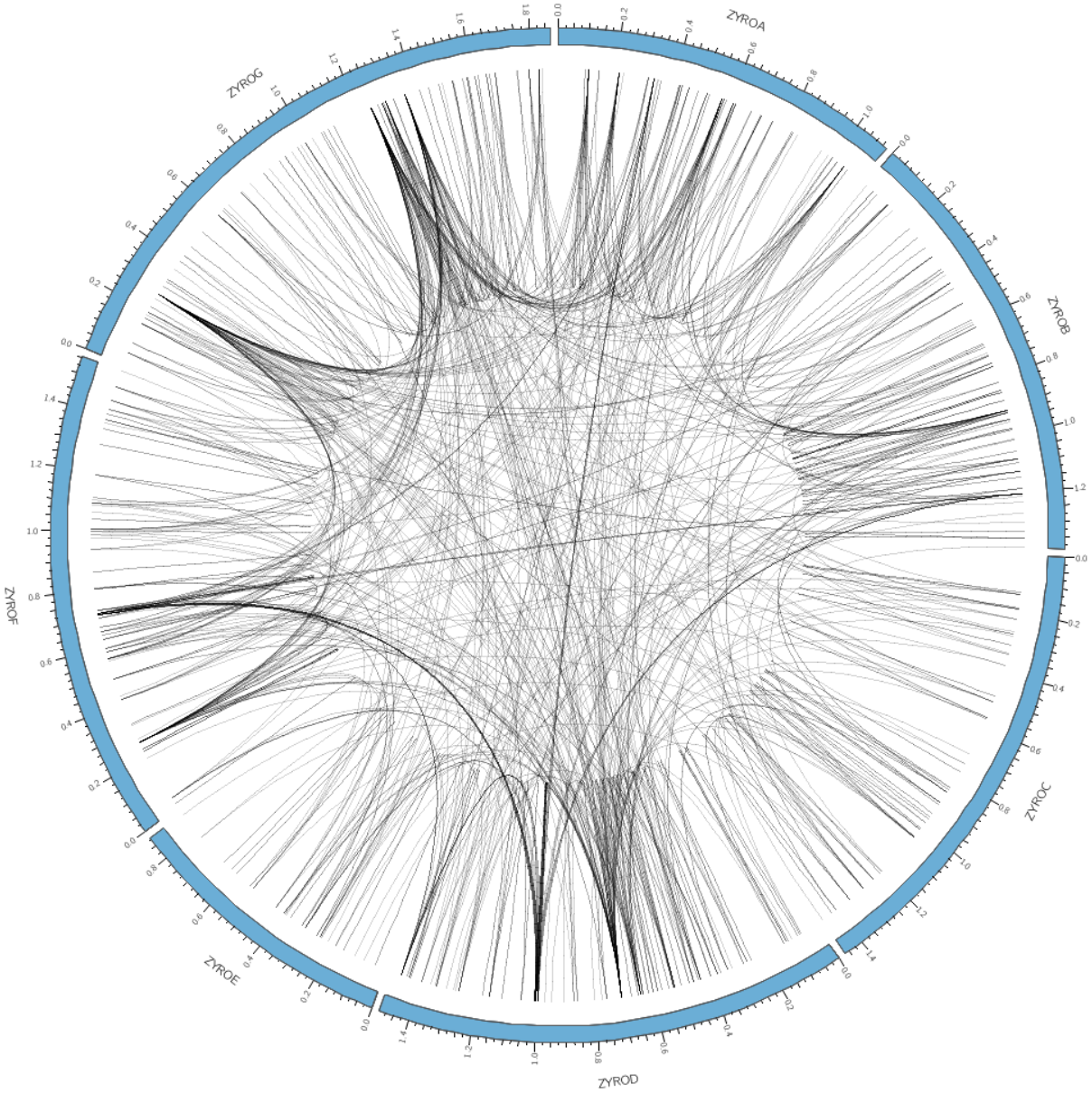

FIGURE 7

Rules are useful to limit what data is displayed. Here, small links are hidden.

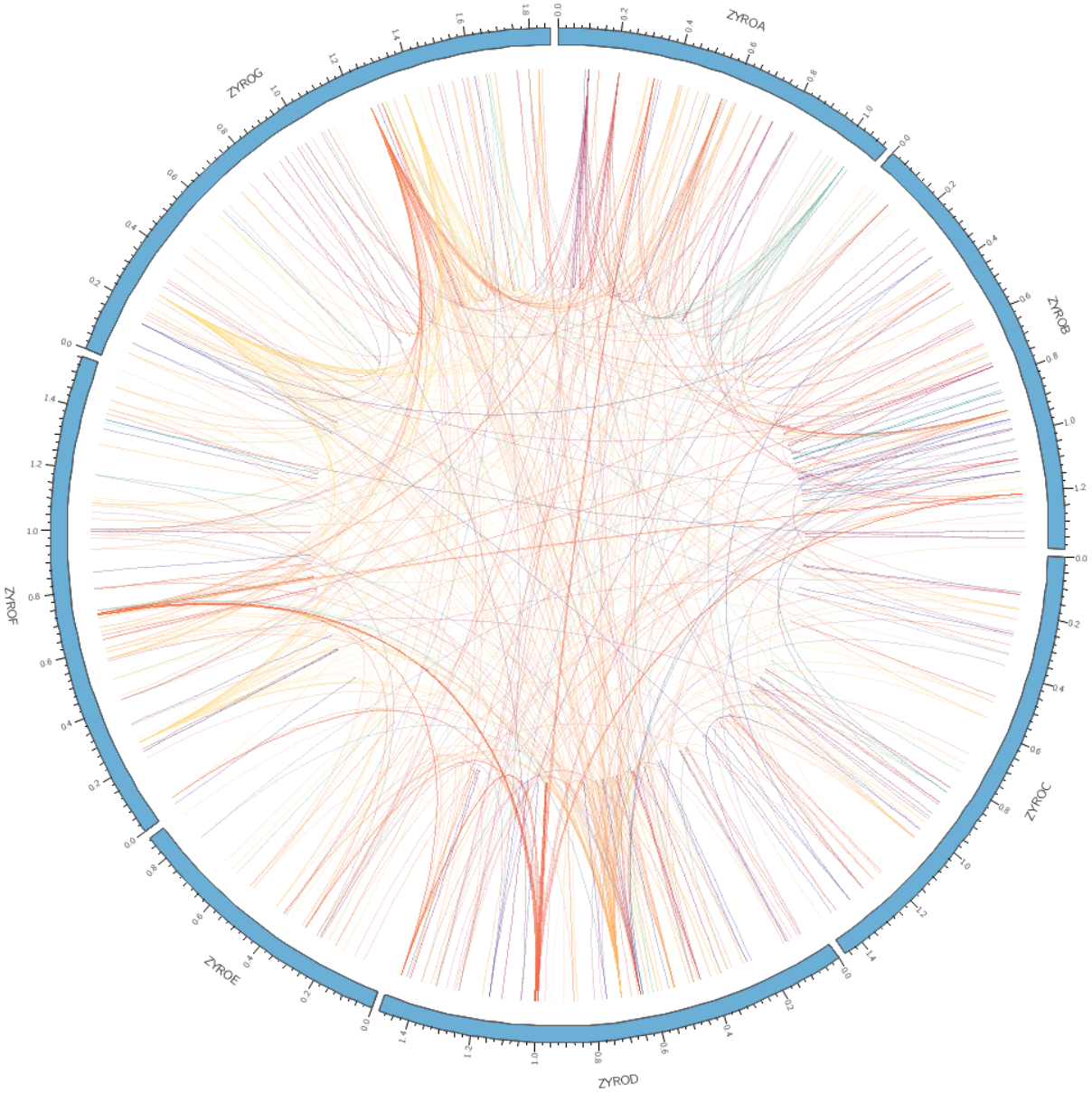

FIGURE 8

Rules are useful to color data. Here, links are colored by size.

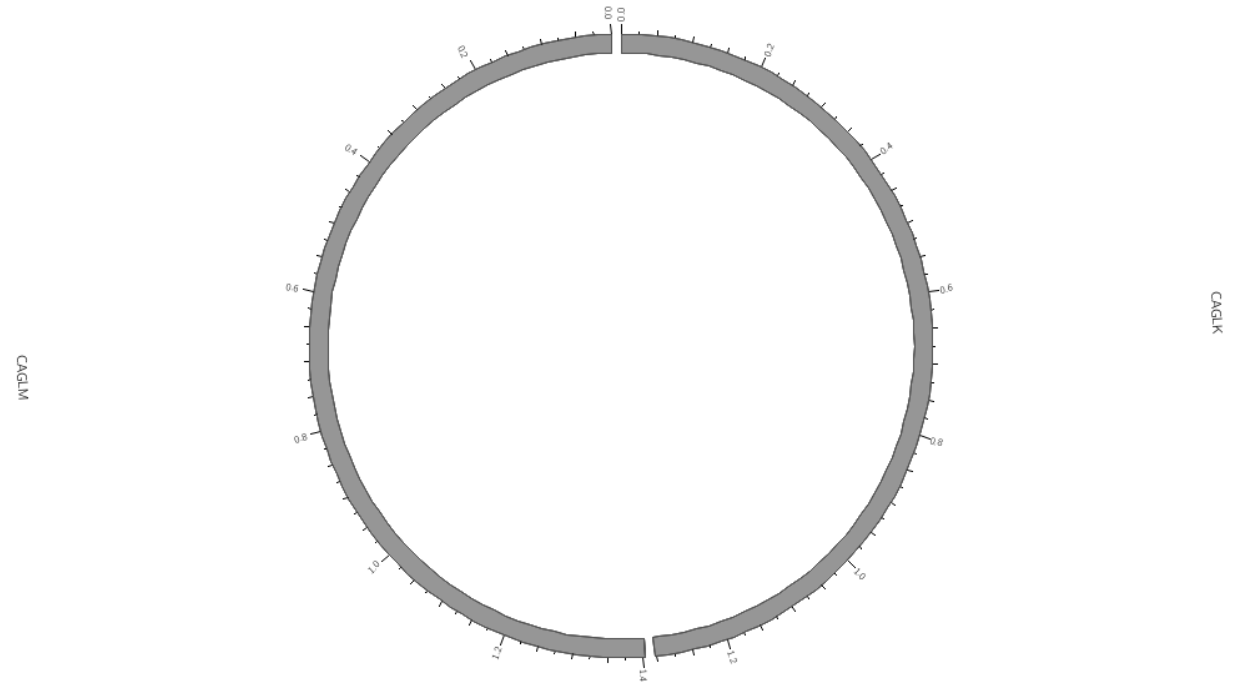

FIGURE 9

Two CAGL ideograms. The ideogram radius was reduced to accommodate links outside the circle.

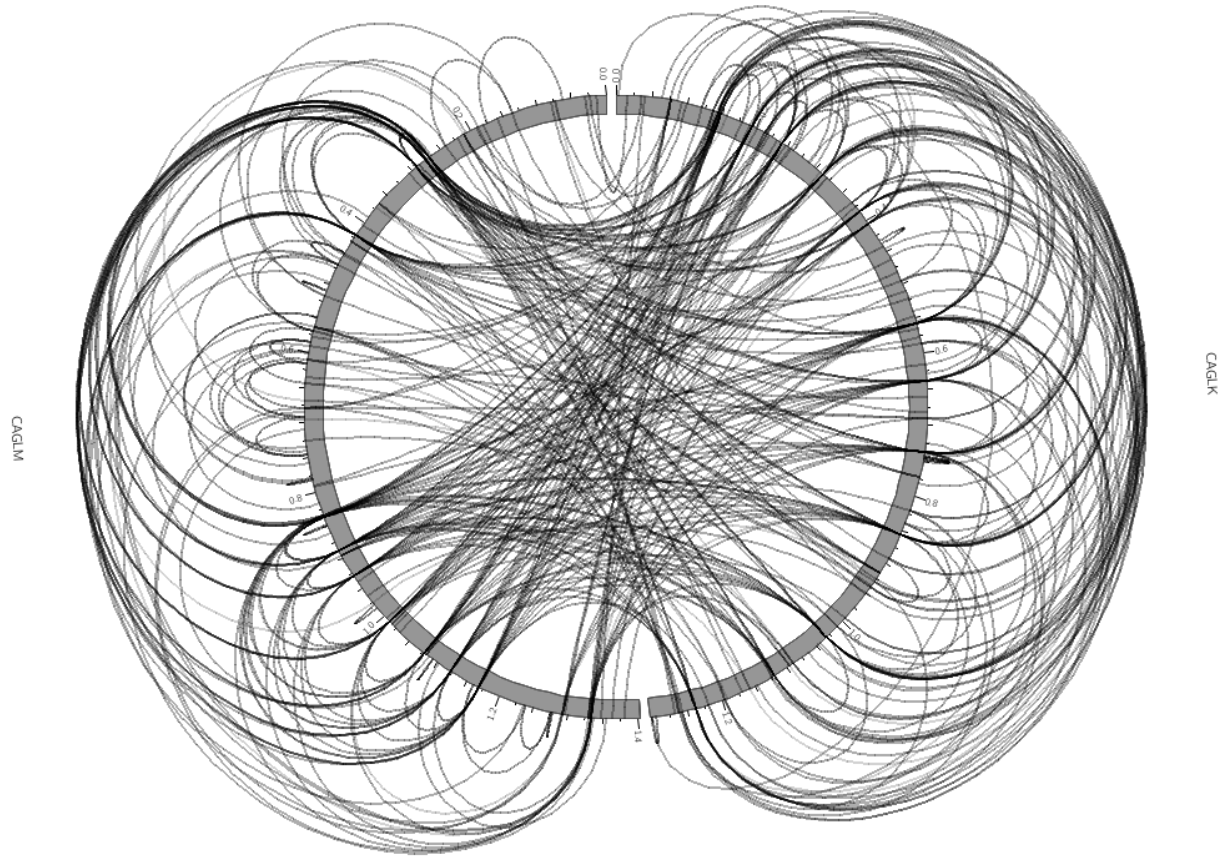

FIGURE 10

Links can be made to curve outwards, away from the circle, by changing the position of the control point (`bezier_radius`) to be larger than the radius of the ideogram.

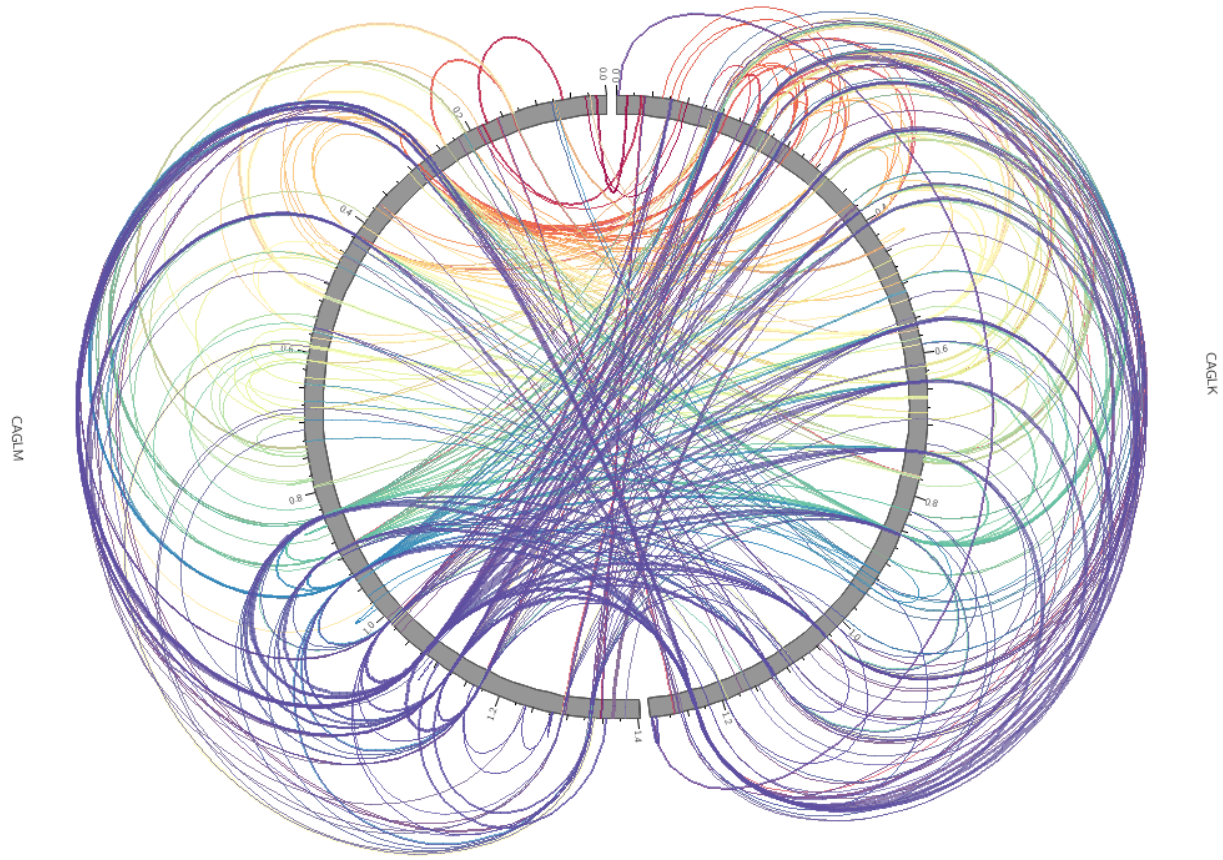

FIGURE 11

Rules can be used to color links by their position.

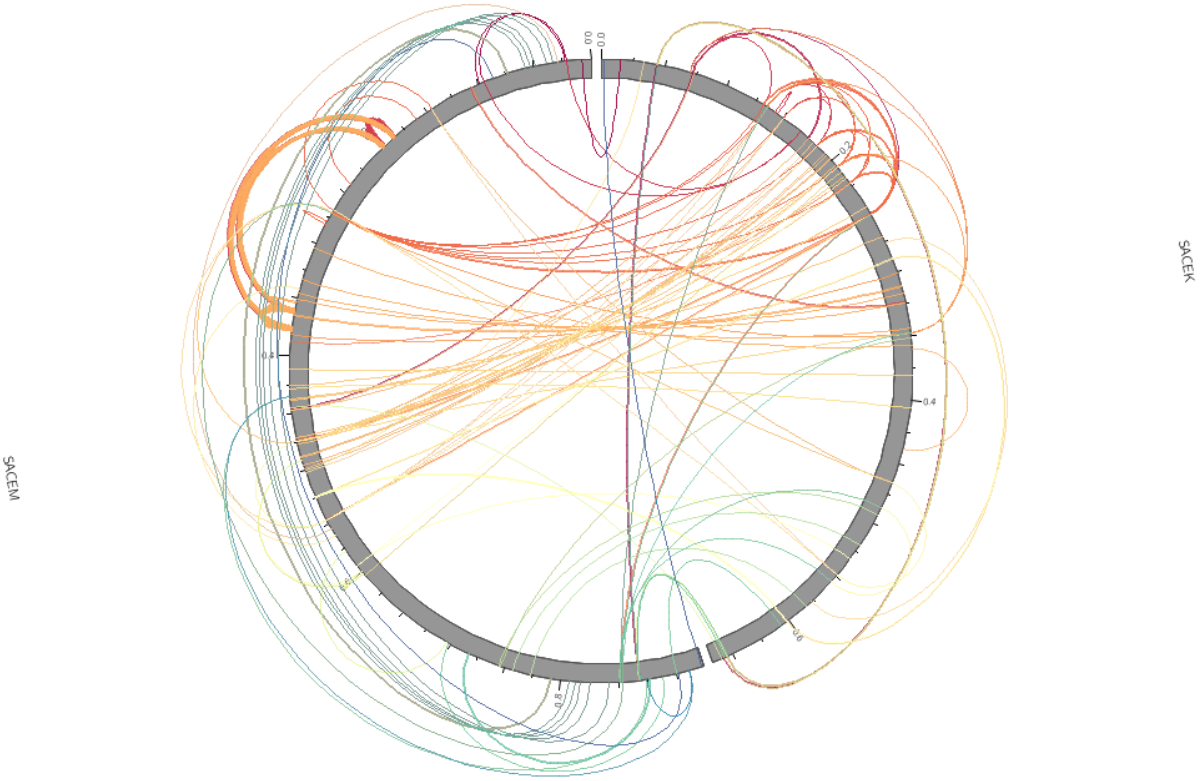

FIGURE 12

By defining a configuration parameter `genomes` (e.g. `genomes=sace`) you can use the sample configuration file to load data from different species. Wherever you would normally use the "`sace`" string, use `conf(genomes)` to recall the value of the parameter.

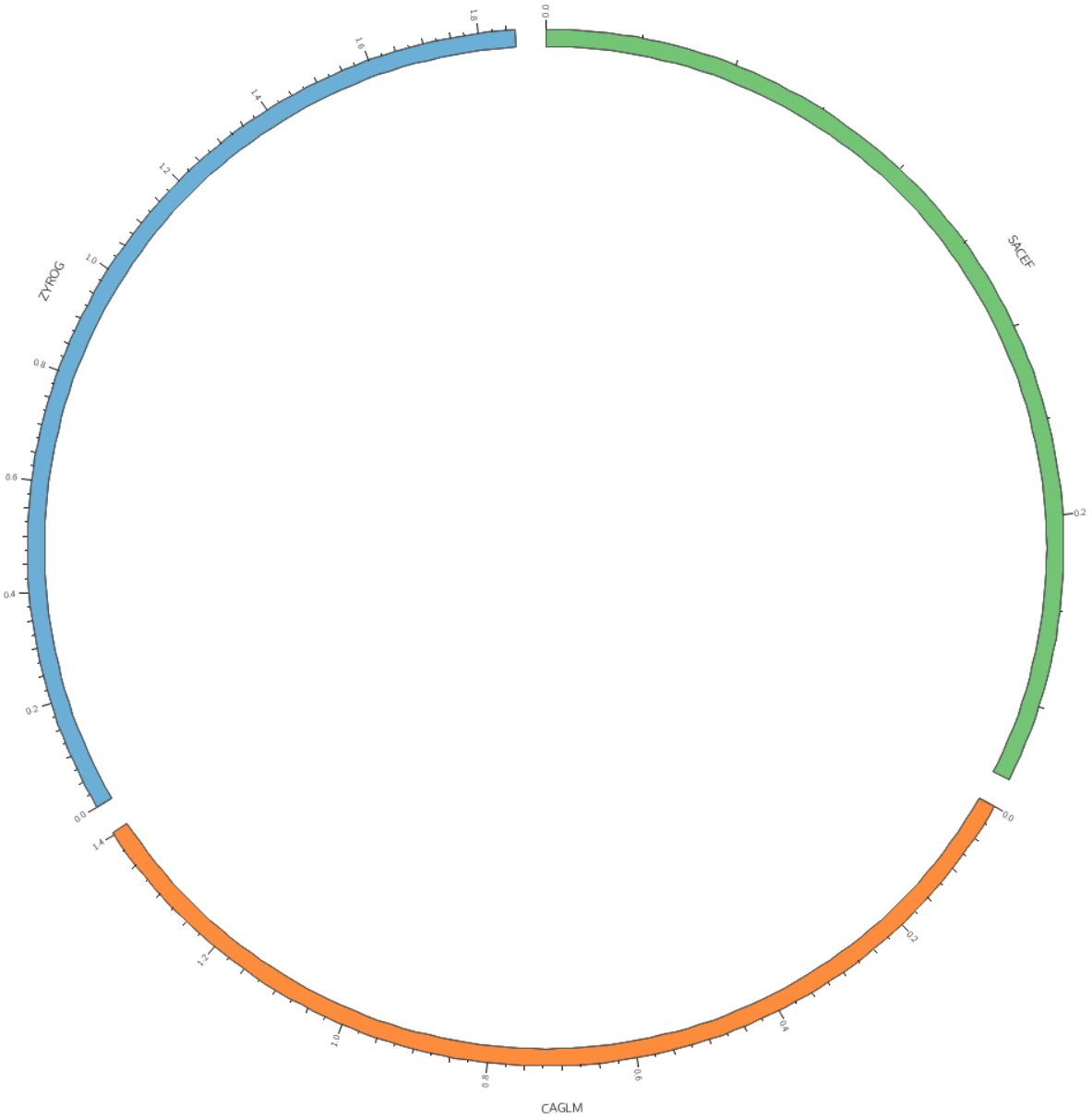

FIGURE 13

Ideograms can be uniformly distributed within an image.

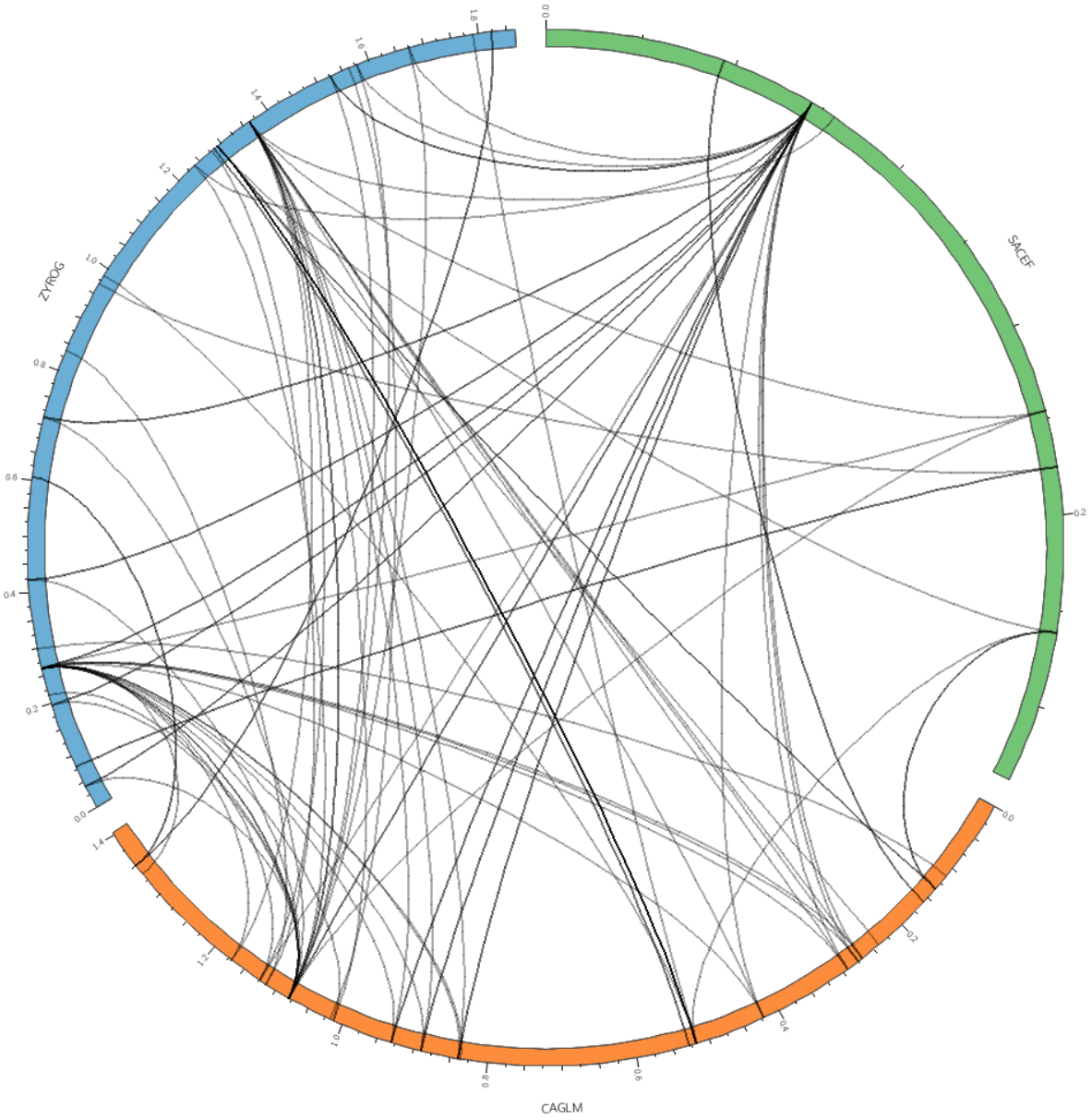

FIGURE 14

Links have been added.

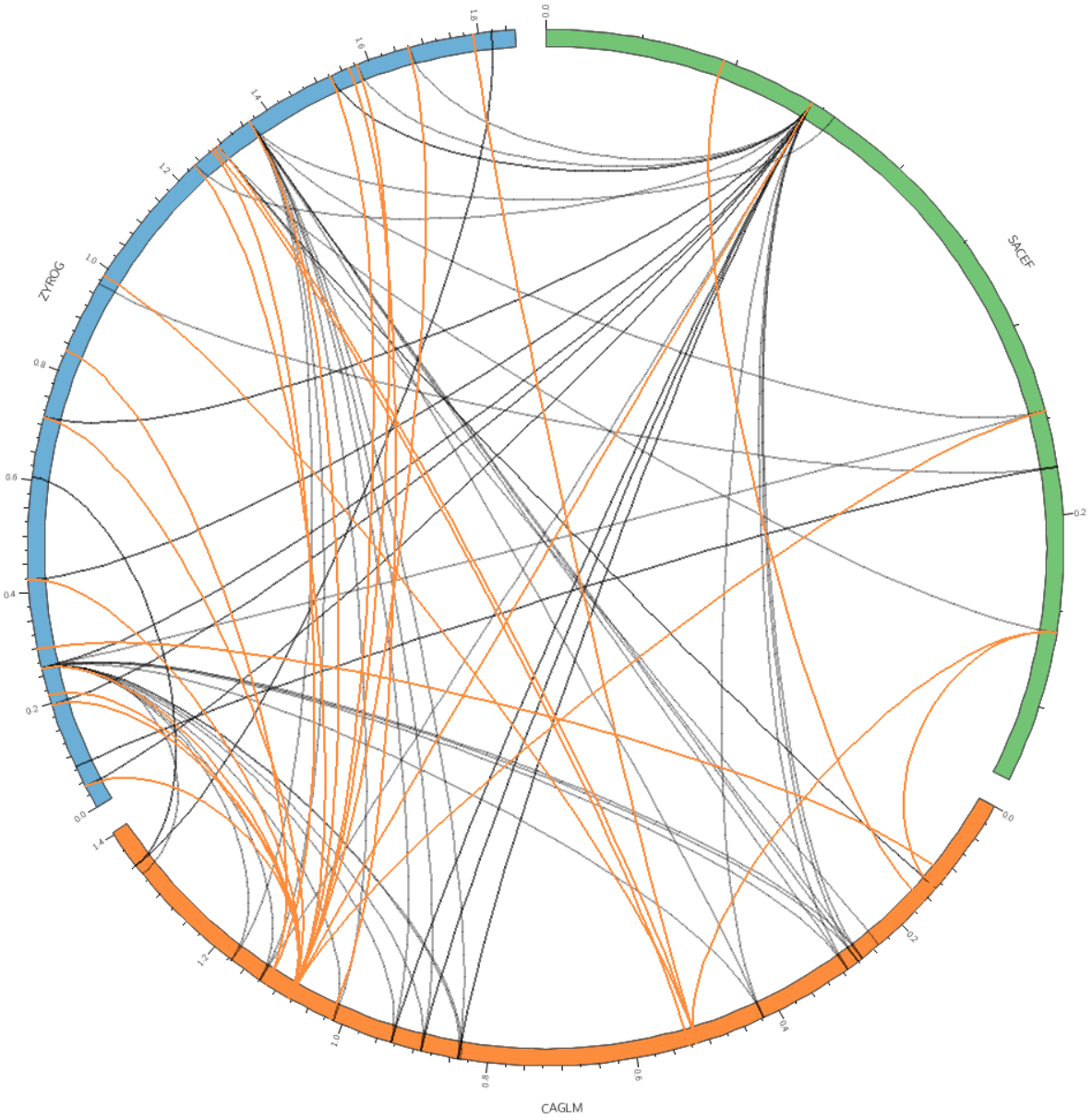

FIGURE 15

Rules can be used to color links by their source ideogram. Here, this is applied only to links from the ideogram from CAGL.

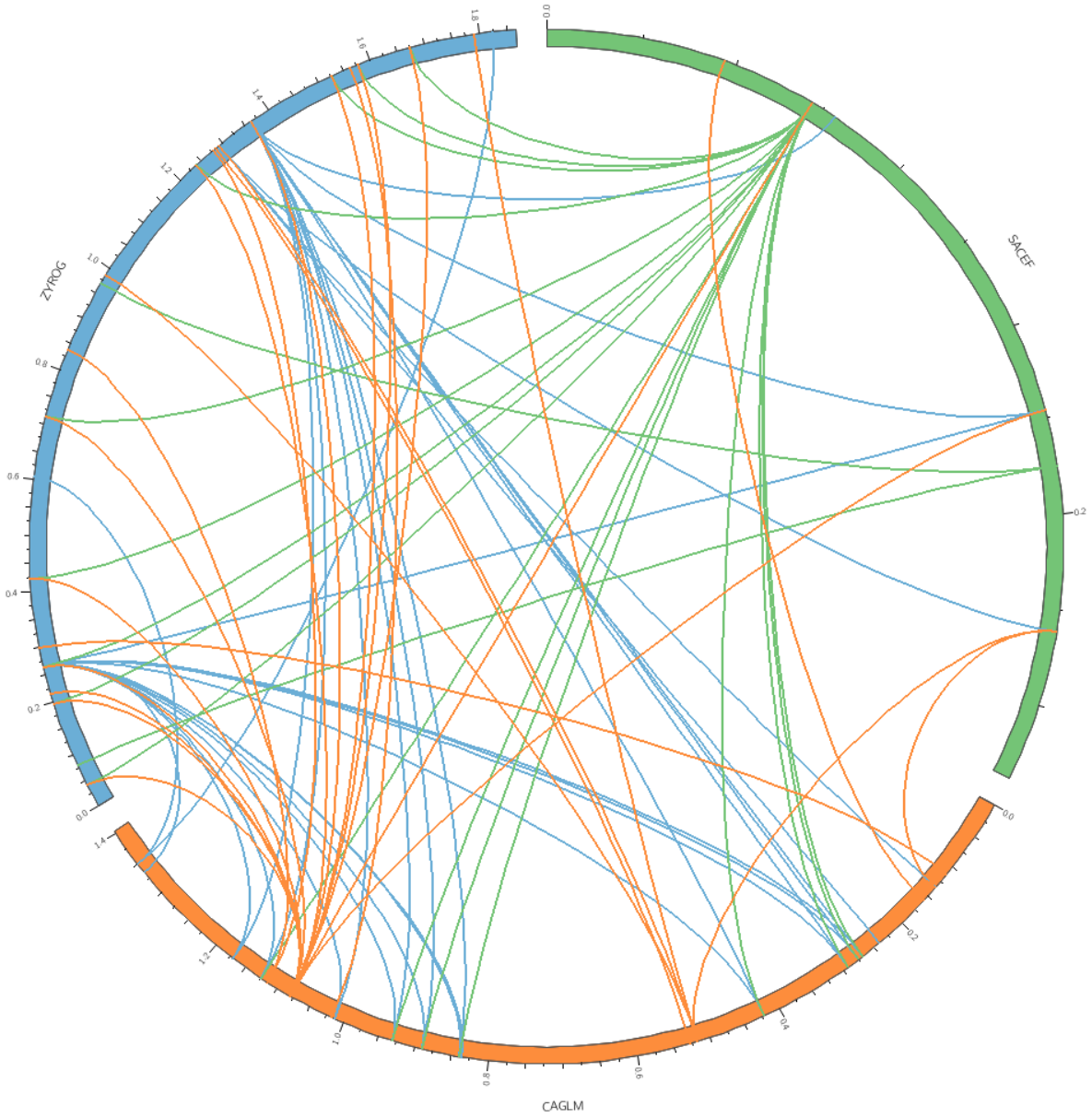

FIGURE 16

All links are colored by their source ideogram using rules.

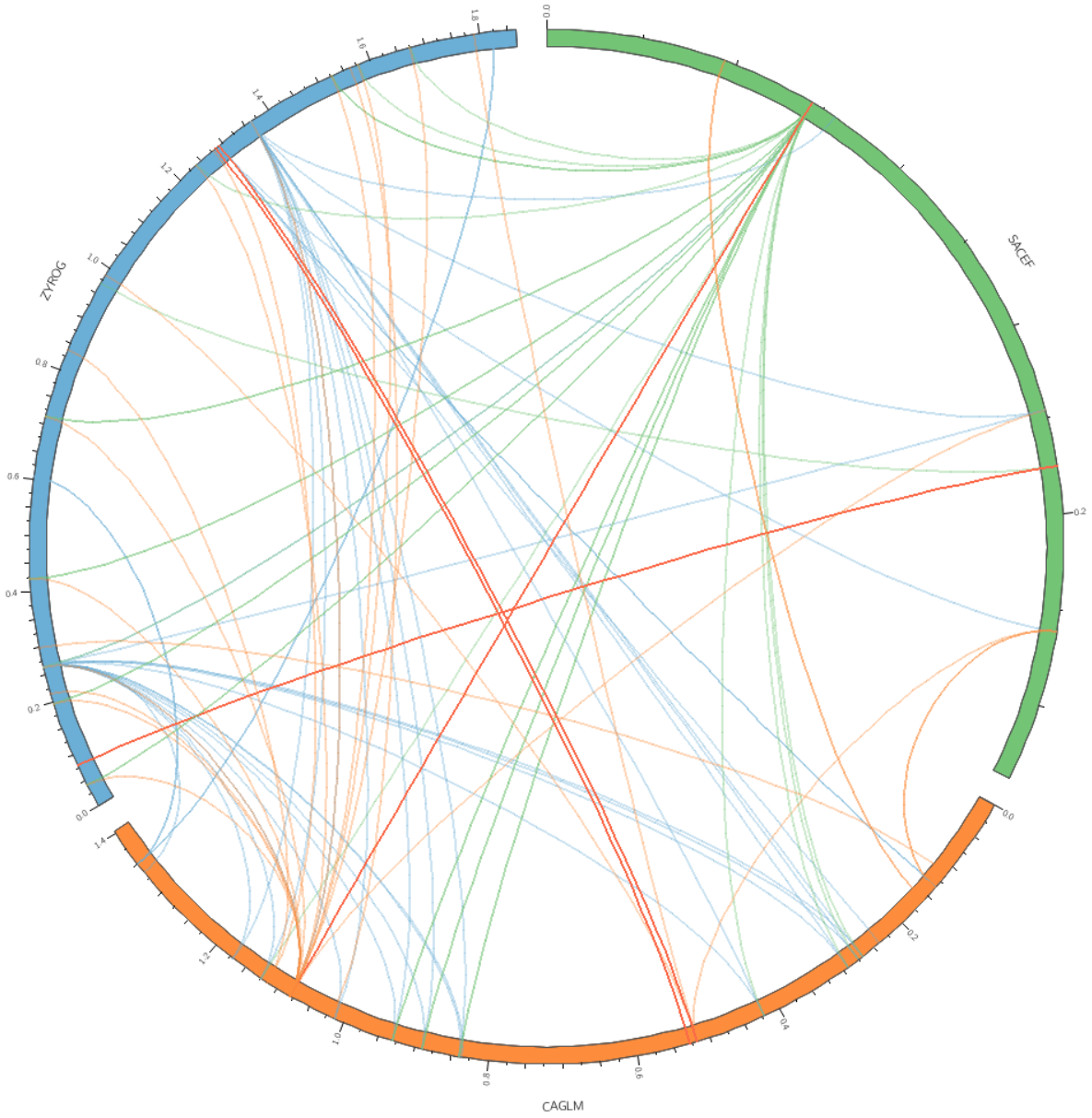

FIGURE 17

Rules have been applied to add transparency to all links and emphasize large ones with a distinct red color.