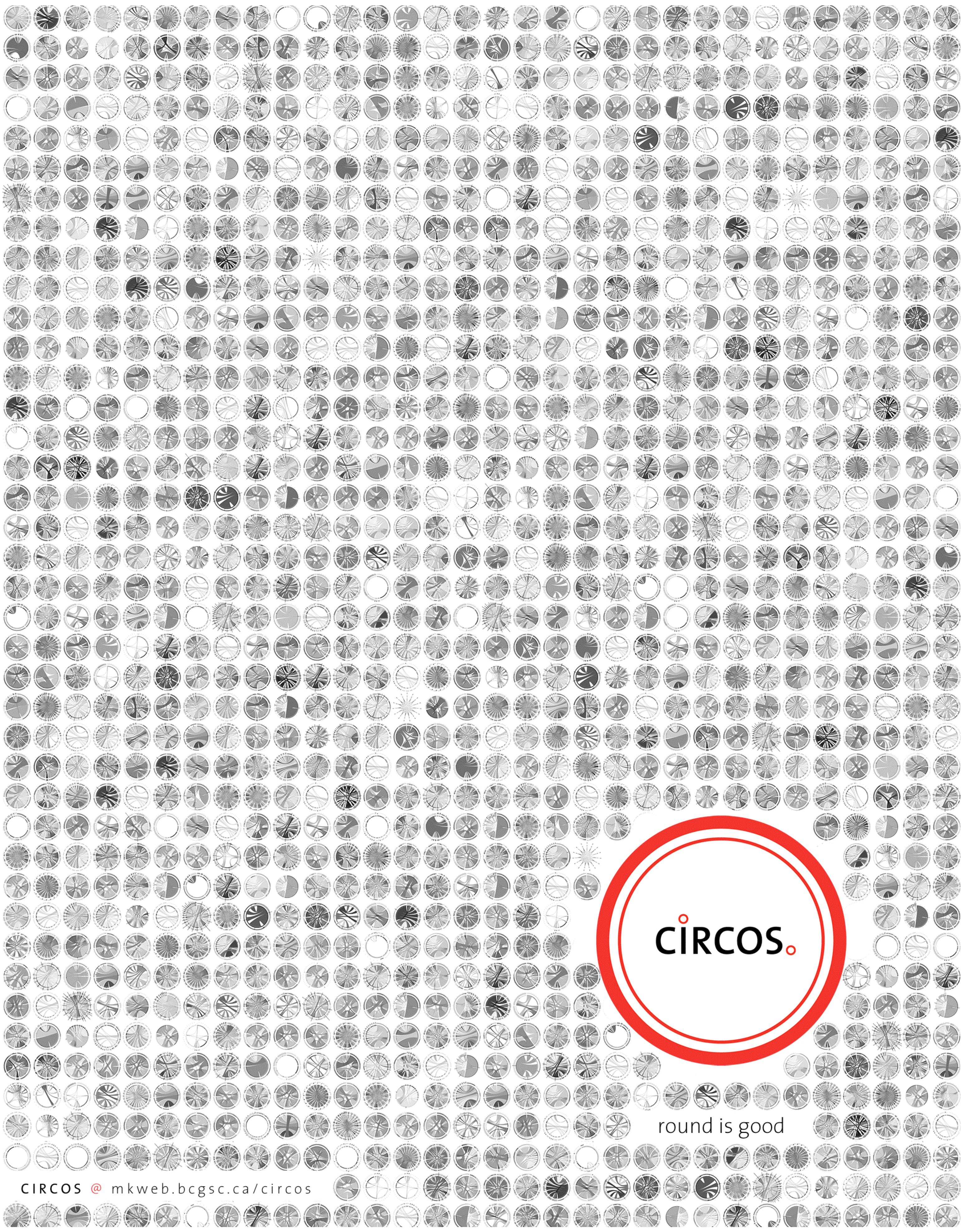
Version History

v0.15	12 Jul 2010
v0.14	30 Jun 2010
v0.13	30 Jun 2010
v0.12	29 Jun 2010
v0.11	17 Jun 2010
v0.10	16 Jun 2010



CIRCOS @ mkweb.bcgsc.ca/circos

Table of Contents

Introduction to Data Tracks

Circos is designed to allow you flexible placement of a variety of data tracks in the figure. Tracks can be positioned anywhere within the figure (Figure 1). It is common to overlap tracks (Figure 1D) to layer data sets together (e.g. drawing multiple histograms within the same region, as we'll see in Lesson 2).

Circos supports a large number of various track types (Figure 2). Many of them use the same input data format, allowing you to switch the track type without altering the input files.

In this session you will learn how to place and format histograms, heat maps, links and highlights. Other data tracks such as scatter plots, line plots, connectors and text tracks work similarly and I encourage you to refer to the online tutorials to learn more about these tracks.

http://mkweb.bcgsc.ca/circos/tutorials/

For this session we will be using genome conservation data downloaded from the UCSC Genome Browser table viewer and comparing chromosomes 1,2 of human with chromosomes 1,2 of mouse. All of the data files for this session are found in `session/3/data`. The original downloaded data files are in `session/3/data/ucsc`. You can regenerate the track data by using the `create.tracks` script.

```
> cd session/3/data/ucsc
> ./create.tracks
Creating data tracks from UCSC data files. This might take 2-3 minutes.
Creating histogram tracks for Lesson 2
 statistic avg
 statistic min
 statistic max
Creating heat map tracks for Lesson 3
 species rhe
 species rn
 species danrer
 species fr
Creating links for Lesson 4
Creating density tracks for Lesson 5
Creating random tiles for Lesson 6
Creating top and bottom 20 conserved regions
```

The `create.tracks` script uses several scripts (`bincons`, `makelinks`, `maketiles`) also found in `session/3/data/ucsc` to bin and format the UCSC input into a format that Circos understands.

Lesson 1 – Ideogram, Tick, Grid and Label Layout

This lesson repeats content from Lesson 12 of the previous session (Session 2). During lessons of the previous session, ideogram layout was introduced and the session ended with the creation of figure framework which we are now going to populate with data tracks.

Let's review the configuration file for this lesson. We will be adding to this file over the course of this session.

```
<<include ideogram.conf>>
<<include ticks.conf>>

<image>
file = s3-1.png
<<include ../../etc/image.conf>>
</image>

karyotype = ../data/karyotype.human_mouse_labels.txt

chromosomes_units           = 1000000
chromosomes_display_default = no

chromosomes         = hs1;hs2;mm1;mm2
chromosomes_order   = hs1;hs2;mm2;mm1
chromosomes_reverse = mm2;mm1
chromosomes_scale   = hs2:1.018;mm1:1.254;mm2:1.360

<highlights>
<highlight>
show = yes
file = ../data/highlight.txt
r0   = 1r+40p
r1   = 1r+45p
</highlight>
# additional <highlight> blocks will go here
</highlights>

# data tracks will go here

<<include ../../etc/housekeeping.conf>>
```

The figure uses the combined karyotype definition of human and mouse (`karyotype.human_mouse_labels.txt`), with altered labels for each chromosome to include the species prefix (`h` for human and `m` for mouse).

Using the `chromosomes` parameter, the ideograms to draw are specified. To adjust the order in which they are displayed (by default they would be drawn in the order of appearance in the karyotype file), `chromsomes_order` is used.

The scale of the mouse ideograms is reversed using `chromosomes_reverse`, and the ideograms are scaled using `chromosomes_scale` so that each occupies ¼ of the figure.

The result is a symmetric and tidy figure layout (Figure 3). It is a good idea to make all the ideograms the same size – it creates a more symmetric and pleasing layout. Since the chromosomes in the figure are all within a relatively narrow range of sizes (181 – 247 Mb), the scale adjustment does not introduce significant stretching that might interfere with interpreting the data tracks.

Notice that the thickness of the ideograms is kept small (15 pixels, which is 1.5% of the figure width). The ideograms play to role of both axes and scale, and are used for navigation. The ideograms should be visible, but also be subtle, as not to draw attention away from the data.

The color key uses a 4-color Brewer palette (Figure 4). Human chromosomes are assigned brown and mouse chromosomes are assigned blue-green. Chromosome 1 from both species is assigned a darker color and chromosome 2 a lighter color.

More Brewer palettes are defined in the color file `sessions/etc/brewer.conf`. This file is imported into the `<color>` block, which in turn is imported into the main configuration file by `sessions/etc/housekeeping.conf`. This color block is shown below.

```
<colors>

# standard colors, from color.conf

...

vlred   = 255,193,200
lred    = 255,122,137
red     = 247,42,66
dred    = 205,51,69

vlgreen = 204,255,218
lgreen  = 128,255,164
green   = 51,204,94
dgreen  = 38,153,71

...

# now brewer palettes, from brewer.conf

...

#################################################################
# diverging

# 4 color, red-yellow-green

bd1-1 = 215,25,28
bd1-2 = 253,174,97
bd1-3 = 166,217,106
bd1-4 = 26,150,65

# 4 color, brown-blue-green

bd2-1 = 166,97,26
bd2-2 = 223,194,125
bd2-3 = 128,205,193
bd2-4 = 1,133,113

...

</colors>
```

Introduction to Brewer Palettes

This is a supplemental section to Lesson 1. You can skip it on first reading.

Brewer palettes were designed by Cynthia Brewer. She proposed her palettes as standard choices for color indexing in maps and charts. The design of the Brewer palettes was based on the science of color perception. The palettes therefore embody desirable perceptual characteristics such as perceived order, equal perceived distance and equal importance.

There are three groups of palettes: sequential, diverging and qualitative (Figure 5). For each group, palettes from 3-12 colors are available, across a range of hues and saturation.

Using Brewer palettes is a *very good idea*. These color choices have been validated by experts and designers and have an excellent track record of usability. If you are interested in generating these palettes algorithmically, see Wijffelaars *et al.* (2008).

References

http://www.colorbrewer.org

http://www.personal.psu.edu/cab38/Brewer_pubs.html

C.A. Brewer, **1999**, **Color Use Guidelines for Data Representation**, *Proceedings of the Section on Statistical Graphics*, American Statistical Association, Baltimore, pp. 55-60.

M. Wijffelaars, R. Vliegen, J.J. van Wijk, E.-J. van der Linden, 2008, **Generating color palettes using intuitive parameters.** *Computer graphics forum.* vol.27, no. 4, p. 743-750.

Lesson 2 – Histograms

In this lesson, you will see how to create your first data track. We will draw a histogram.

There are two kinds of data tracks: plots and links. We'll cover links later.

Plots are defined individually in `<plot>` blocks, within an encompassing `<plots>` block.

```
<plots>

<plot>
# data track definition
...
</plot>

<plot>
# another data track
...
</plot>

</plots>
```

To start, we'll create an image with one histogram. This histogram is defined in the first `<plot>` block in this lesson's `etc/circos.conf` file. For now, don't worry about the lines that are commented out. If you wish to disable a track, define `show = no` within the `<plot>` block.

```
<plot>
type        = histogram
file        = ../data/both.cons.2e6.max.txt
min         = 0
max         = 1
r0          = 0.85r
r1          = 0.975r
color       = black
thickness   = 2p
#fill_under  = yes
#fill_color  = bs8-1
</plot>
```

The syntax of the plot block is straightforward. The `type` defines what kind of track this is (histogram, scatter, line, heatmap, etc) and `file` specifies the input data file which contains the track data. The histogram is placed between inner radius `r0 = 0.85r` and outer radius `r1 = 0.975r`. These values are relative to the ideogram circle.

The histogram bins are outlined with `color = black` and the outline has a `thickness = 2p` (pixels). The result is shown in Figure 6.

The format of the histogram data file is

```
# chr start end value
mm1 10000000 11999999 0.439079022807017
mm1 12000000 13999999 0.44630188700565
mm1 14000000 15999999 0.452856295597484
```

Most data track files have this format, which assigns a value to a chromosome and region. Notice that the histogram value is not defined necessarily at a single point, but over a region. The region controls the width of each histogram bin. Histograms can have variable size bins and bins do not need to abut (but they should not overlap). For data tracks like the scatter plot, the data point is drawn in the middle of the region.

By adding `fill_under` and `fill_color` parameters, the histogram bins are filled. Here, color bs8-1 is used (first color of a 3-color sequential Brewer palette: ● `bs8-1`, ● `bs8-2`, ● `bs8-3`).

```
fill_under  = yes
fill_color  = bs8-1
```

The result is shown in Figure 7.

Let's add another histogram on top of the one we've already drawn. This histogram will occupy the same region in the figure, but will be drawn on top of the previous histogram. The order in which tracks are drawn is controlled by the `z` (think of it as depth) parameter. By default, each track has `z = 0`. Tracks are drawn in order of their `z` value; thus, tracks with higher `z` are drawn on top of tracks with a lower `z`. Individual data points can have their own `z` value as well.

The second histogram will use another data file and we'll fill it with color ● `bs8-2`. The result is shown in Figure 8.

```
<plot>
show         = yes
type         = histogram
file         = ../data/both.cons.2e6.avg.txt
min          = 0
max          = 1
r0           = 0.85r
r1           = 0.975r
color        = black
#color        = white
thickness    = 2p
fill_under = yes
fill_color = bs8-2
z          = 5
</plot>
```

Finally, we'll add a third histogram also in the same region of the figure, filled with ● `bs8-3` (Figure 9).

```
 <plot>
show         = yes
type         = histogram
file         = ../data/both.cons.2e6.min.txt
min          = 0
max          = 1
r0           = 0.845r
r1           = 0.975r
color        = black
thickness    = 2p
fill_under   = yes
fill_color   = bs8-3
z            = 10
</plot>
```

Let's now reformat the three histograms in Figure 9 to create an interesting visual effect.

```
<plots>

<plot>
...
# comment out color and thickness to
# remove the outline on this histogram
#color       = black
#thickness   = 2p
...
</plot>

<plot>
...
# make the outline of this histogram white
color       = white
...
</plot>

<plot>
...
# comment out color and thickness to remove outline on this histogram
#color       = black
#thickness   = 2p
# make the fill color white
fill_under  = yes
fill_color  = white
...
</plot>

</plots>
```

The result is shown in Figure 10. By setting the fill of the bottom histogram to white, we've made the figure look as if the second histogram drops down. By setting the outline of the second histogram to white, we've effectively separated the first and second histograms.

What is Shown in the Figure?

The three histograms in the figure show the maximum, average, and minimum conservation scores between mouse and human in 2Mb bins. The histogram values on the mouse chromosomes correspond to conservation with the human genome, and on the human chromomosomes to conservation with the mouse genome.

The original data can be downloaded from the UCSC Genome Browser table viewer (*group* Comparative Genomics *track* 28-way Cons *table* multiz28waySummary). The downloaded data files are available in `sessions/3/data/ucsc` as `multiz.*.txt`. To learn more about the data in this track (*multiz* alignments scored by *phastCons*), click "describe table schema" for the *multiz28* table in the UCSC Genome Browser table viewer.

The `bincons` script in `sessions/3/data/ucsc` scans the contents from the files, bins the scores for alignments between mouse and human and calculates the minimum, maximum and average score values for each bin. For example,

```
cat multiz.mm9.*  | ./bincons -stat min -binsize 2e6 -chrprefix mm -species
hg18
cat multiz.hg18.* | ./bincons -stat min -binsize 2e6 -chrprefix hs -species
mm8
```

and the output of these commands is used as the input for the minimum score histogram. This and other commands to generate all the data tracks for this session can be found in `sessions/3/data/ucsc/create.tracks`.

Lesson 3 – Heat Maps

Heat maps are tracks which map a range of colors onto a value range and display data as colored blocks.

A heat map is defined similarly to a histogram – it uses a `<plot>` block and has the same `type,` `min`, `max`, `r0` and `r1` parameters. The `color` parameter, however, is a comma-separated list of the colors that are mapped onto the `min-max` range. Values beyond the range are assigned to the nearest color.

```
<plot>
type = heatmap
min  = 0.1
max  = 0.9
r0   = 0.73r
r1   = 0.75r
file  = ../data/both.cons.2e6.rhe.avg.txt
color = bd3-1,bd3-2,bd3-3,bd3-4,bd3-5,bd3-6,bd3-7,bd3-8,bd3-9,bd3-10,bd3-11
</plot>
```

Figure 11 shows the heat map between radii `0.73r` and `0.75r`.

Let's add three more heat maps, showing conservation with rat, zebra fish and fugu. These will be spaced by `0.01r` and have a width of `0.02r`.

```
<plot>
type = heatmap
min  = 0.1
max  = 0.9
r0   = 0.70r
r1   = 0.72r
file  = ../data/both.cons.2e6.rn.avg.txt
color = bd3-1,bd3-2,bd3-3,bd3-4,bd3-5,bd3-6,bd3-7,
        bd3-8,bd3-9,bd3-10,bd3-11
</plot>

<plot>
type = heatmap
min  = 0.1
max  = 0.9
r0   = 0.67r
r1   = 0.69r
file  = ../data/both.cons.2e6.danrer.avg.txt
color = bd3-1,bd3-2,bd3-3,bd3-4,bd3-5,bd3-6,bd3-7,
        bd3-8,bd3-9,bd3-10,bd3-11
</plot>

<plot>
type = heatmap
min  = 0.1
max  = 0.9
r0   = 0.64r
r1   = 0.66r
file  = ../data/both.cons.2e6.fr.avg.txt
color = bd3-1,bd3-2,bd3-3,bd3-4,bd3-5,bd3-6,bd3-7,
        bd3-8,bd3-9,bd3-10,bd3-11
</plot>
```

Figure 12 shows all four heat maps. The data for these heat maps is binned in 2Mb windows, which is just large enough to make out in the figure. In general, you want to keep your bins large enough to be easily visible.

By default, color mapping is done linearly within the min-max range of the heat map. In other words, the range is divided by the number of colors, and each color is assigned the same interval. When your input values data are highly skewed, a linear mapping results in most of the values assigned to a small number of colors. Here, a logarithmic mapping is helpful, because it distributes values among colors more uniformly.

To apply logarithmic mapping, use the `scale_log_base` parameter, and provide a value which is used as follows

$$i = n\left(\frac{x-\min}{\max-\min}\right)^{\frac{1}{\text{scale\_log\_base}}}$$

where i is the color index, n is the number of colors, and x is the value of the data point. As `scale_log_base > 1` grows, the color index increases for a given value, skewing the mapping towards colors at the end of the list. This is useful to expand the dynamic range of the data when you have many values near the minimum of the set. Conversely, when `scale_log_base < 1`, the skew is towards colors at the start of the list, useful when you have many values near the maximum.

Logarithmic mapping is shown in Figure 13. As `scale_log_base` increases to 1.25, 1.5 and 1.75 more yellow and green colors appear. When `scale_log_base` is made smaller than 1 (0.75 and 0.5), the mapping produces more reds.

What is Shown in the Figure?

Data files for the heatmaps are generated in the same manner as done for the histograms in the previous lesson. Again, the `bincons` script in `sessions/3/data/ucsc` is used to scan the `multiz.*.txt` files to generate average conservation score between mouse and human and the genomes of rhesus, rat, zebra fish and fugu.

The format of data files for the heatmaps are identical to those for histograms. You can easily change a heatmap to a histogram (and vice versa) by simply changing the `type` parameter in the `<plot>` block. Don't forget that a heatmap requires a list of colors, whereas a histogram only a single color.

Lesson 4 – Links

Links are the raison d'être of Circos. This track type allows you to connect two positions of any ideograms with Bezier curves. Several parameters allow you to change the curve geometry, turn curves into ribbons and control the ribbon twists.

Links are defined in named `<link>` blocks, within an outer `<links>` block.

```
<links>
<link chain>
file          = ../data/links.txt
bezier_radius = 0r
radius        = 0.5r
thickness     = 1p
color         = black
#color        = black_a5
</link>
</links>
```

Note that the `<link>` block has a name (e.g. `<link chain>`). The name is arbitrary, but must be unique. For now, the name is not directly used. It is likely that future versions of Circos will require that `<plot>` blocks have names as well. For now, only `<link>` blocks need to be named.

Links are drawn as quadratic Bezier curves (see *http://www.ibiblio.org/e-notes/Splines/Bezier.htm*). This kind of curve is specified by two ends and one control point. The control point determines the direction of the curve at its ends. The *radius* parameter controls the radial position of the ends of the curve. The *bezier_radius* parameter controls the radial position of the control point, which is placed midway between the ends. Figure 14 shows the construction of a link.

There are several parameters that help you further control the curve geometry. These are described in the *Links geometry* tutorial.

http://mkweb.bcgsc.ca/circos/tutorials/lessons/links/geometry

For example, the `crest` parameter makes the curve perpendicular to its end radius and `bezier_radius_purity` is used to dynamically adjust be `bezier_radius` parameter based on the length of the curve.

Figure 15 shows the links as defined in the above block.

The format of the input file for links is as follows. Each link is specified on two lines, one line for the start of the link and one for its end. The two lines are associated together with a unique identifier, which is the first field. The identifier is arbitrary, as long as it is unique.

```
...
# ID chr start end
link7572 hs1 120788758 120834144
link7572 mm1 63903740 63957877
link54010 hs2 190247704 190320005
link54010 mm1 124448312 124506291
...
```

By adding transparency to the color of the link lines (Figure 16) dense links can be more easily interpretable (compare Figure 16 with Figure 15).

```
#color        = black
color        = black_a5
```

Recall that suffixing a color name with `_a` + *digit* creates a transparent color. The degree of transparency depends on `auto_alpha_steps` (in these lessons this parameter is set to 5, which provides 5 levels of transparency). For example, the extent of transparency (smaller values mean that the color is *more* opaque) for variants of red is

opaque					*very transparent*
`red` 0%	`red_a1` 16%	`red_a2` 33%	`red_a3` 50%	`red_a4` 66%	`red_a5` 83%

What is Shown in the Figure?

The data for the links were obtained from the human hg18 assembly in the UCSC Genome Browser table viewer (*group* Comparative Genomics *track* Mouse Chain/Net *table* Mouse chain). These files are available in `sessions/3/data/ucsc` in `hg18.mm.chr*.txt`. The Circos link data files are generated with the `makelinks` script, which parses the UCSC gapped alignments into a format that Circos understands. The top 1000 alignments up to 100kb are selected.

Lesson 5 – Density and Stacked Histograms

In this lesson we will add another histogram immediately outside of the links to indicate the number of link ends within a region. In effect, the histogram will act to represent link density.

One of the utility scripts that comes with Circos creates this kind of histogram from a link file. This is the `binlinks` utility (`/usr/shared/circos-0.52/tools/binlinks`).

The syntax that defines the density histograms is nearly identical to what you've seen in Lesson 2. The first histogram (`link.density.txt`) is placed between `r0 = 0.5r` and `r1 = 0.55r` and maps onto a range `0-20`. This histogram counts the number of link ends within each 1 Mb window.

The second histogram is a special kind of histogram – a stacked histogram. In this kind of histogram, each bin is characterized by multiple values. The bin values are set by a comma-delimited list in the input file

```
hs2 5000000 9999999 1.0000,0.0000,0.0000,1.0000
hs2 10000000 14999999 1.0000,0.0000,0.0000,2.0000
```

The colors associated with each value index are given by a comma-delimited list assigned to `fill_color`.

```
<plot>
type       = histogram
min        = 0
max        = 20
r0         = 0.5r
r1         = 0.55r
file       = ../data/link.density.txt
fill_under = yes
fill_color = black
</plot>

<plot>
type       = histogram
min        = 0
max        = 40
r0         = 0.55r
r1         = 0.65r
file       = ../data/link.density.stacked.txt
fill_under = yes
fill_color = bd2-4,bd2-1,bd2-2,bd2-3
</plot>
```

For a given value index a separate histogram is drawn, with its bins sitting on top of those of the histogram for the previous value index. The result is show in Figure 17.

Density histograms are excellent for providing a summary of the link data set. When the stacked histogram option is used, it is easy to see the relative number of links between a given position and all other ideograms.

Figure 18 shows a zoomed version of Figure 17, with only the links and density histograms shown. It is easy to see that in region (A) the majority of links depart to terminate on h2. Region (B), on the other hand, is more similar to h1 as seen by the fact that most links that depart from this region terminate on h1. Similarly regions (C) and (D) on h2 are more similar to m1 and m2, respectively.

The stacked density histograms help interpret the extent of synteny between a given position and other chromosomes of the other species. Without the links, however, it is not possible to determine from the histograms the exact positional syntenic relationship. The link and histogram tracks complement each other, presenting the data at two different levels (histograms are the summary, links provide the detail).

Lesson 6 - Tiles

Tiles (or cover elements) are an important data type in genomics. They are useful to visualize data that is interpreted as sampling or coverage process (reads, clones, alignments, genes, etc).

The tile track is one of the tracks in which position of elements is determined algorithmically. For tiles, elements are stacked in layers to avoid overlap. You do not have direct control in which layer within the track the element will appear. Several parameters control element placement, including element padding and margin and how elements that cannot fit within the track are to be handled.

The tile track shown in Figure 19 is defined below. A `<plot>` block is used and again `type`, `file`, `r0` and `r1` are common parameters.

```
<plot>
type    = tile
file    = ../data/tiles.txt
r0      = 1r+3p
r1      = 1r+40p

layers  = 5
layers_overflow       = hide
layers_overflow_color = red

margin    = 1u
thickness = 5
padding   = 2

orientation = out

color            = white
stroke_thickness = 1
stroke_color     = dgrey

background = yes
background_color = grey_a2

</plot>
```

Within the track, tile elements are stacked in layers. The number of layers is given by the `layers` parameter. Elements in the same layer cannot be closer than `margin` distance. In this example, `margin = 1u`, which means that there is at least 1Mb of free space between two tiles in the same layer. The radial size of each tile is given by `thickness` and the distance between layers by `padding`. See Figure 20 for a visual explanation of tile placement.

Tiles can stack inward (`orientation = in`) or outward (`orientation = out`). They can also stack in both directions, with the baseline layer placed in the middle of the track. If you are placing tiles outside (inside) of the ideograms, it is sensible to use an outward (inward) orientation.

Each track can be given a background color and in this case I've set the tile track background to a transparent grey (`grey_a2`). Transparency allows the grid (see next lesson) underneath the track to show through.

For more information about tiles, see the tutorial at

http://mkweb.bcgsc.ca/circos/tutorials/lessons/2d_tracks/tiles/lesson

What is Shown in the Figure?

The tiles are randomly generated using `sessions/3/data/ucsc/maketiles`. For each chromosome 100 tiles of average length 5Mb are created. For each tile, length can vary between 0-10Mb.

Lesson 7 - Highlights

We've already used the highlight track to generate colored strips at the foot of ideogram ticks to provide a color index for each ideogram. Now, we'll come back and add two more highlights to draw attention to regions that had very low and very high conservation levels. One highlight track will show the top 20 conserved regions (`scatter.max.top20.txt`) and another the bottom 20 (`scatter.min.top20.txt`).

The data for these tracks were generated by sorting the histogram files by value (either descending or ascending) and taking the first 20 lines.

```
# sort bins by maximum conservation score
# and take 20 bins with largest scores
cat ../both.cons.2e6.max.txt | sort -nr +3 | head -20 |
   cut -d " " -f 1-3 > ../highlight.max.top20.txt
# sort bins by minimum conservation score
# and take 20 bins with smallest scores
cat ../both.cons.2e6.min.txt | sort -n +3  | head -20 |
    cut -d " " -f 1-3 > ../highlight.min.top20.txt
```

I've added outlines to these highlight elements because the yellow color does not contrast well with white for small elements.

```
<highlight>
file = ../data/scatter.max.top20.txt
r0   = 1r+45p
r1   = 1r+57p
fill_color = bs8-1
stroke_thickness = 1p
stroke_color = black
</highlight>

<highlight>
file = ../data/scatter.min.top20.txt
r0   = 1r+45p
r1   = 1r+57p
fill_color = bs8-2
stroke_thickness = 1p
stroke_color = black
</highlight>
```

For this lesson, the grid has also been turned on. The grid was initially shown in Figure 3, but was later turned off to keep the examples simpler. Now, in `etc/ticks.conf` the `show_grid` parameters is set to `yes`.

```
show_grid       = yes
```

You probably noticed that Circos has parameters like `show` and `show_*` variants. These parameters allow you to toggle the track (or other elements) without having to comment out (or remove) their definition.

Figure 21 shows the addition of the two highlight tracks and the grid.

Lesson 8 – Introduction to Rules

Rules are conditional blocks which change the format of individual data points in a track. Rules can contain simple statements which assign a new value to a parameter, such as

```
thickness = 3p
```

or Perl code which is evaluated, possibly using features of the data point to generate a new value

```
thickness = eval( min(5,int(_SIZE1_/10e3) ) )
```

where `_SIZE1_` is the size of the start span of a link.

To introduce rules, we're going to dynamically format the links in the image from the last lesson.

```
<links>

<link chain>

...

<rules>
<rule>
# each link is tested with a rule
# if the condition passes...
condition   = _CHR1_ eq "hs1"
# the color is changed to bd2-1
color       = bd2-1
</rule>
</rules>

</link>

</links>
```

The rule shown above will change the color all links that start on human chromosome 1 (`hs1`). The assigned color will be `bd2-1`, which is the color used for the chromosomes color index. The result is shown in Figure 22.

Multiple rules can be used. All links are tested with rules in order of the `importance` parameter. When the `condition` of a rule evaluates to true, the rule is applied to the link and the next link is tested. In other words, the first rule that matches a link short-circuits further testing. This can be adjusted using `flow = continue` to allow multiple rules to match a data point or `flow = restart` to start testing from the start of the rule list.

For this example, we'll add another rule that hides all the links that do not pass the first rule.

```
<rule>
importance = 100
condition  = _CHR1_ eq "hs1"
color      = bd2-1
</rule>

# any link that passed the above rule is not tested further

<rule>
importance = 90
# this rule always matches, since its condition is always true
# any link that filed the previous rule is matched by this one
condition  = 1
# the link will be hidden
show       = no
</rule>
```

The result of the two rules is shown in Figure 23. To further refine which links are shown, we can select links based on position. By changing the condition of the first rule to

```
condition = _CHR1_ eq "hs1"
            &&
            _CHR2_ eq "mm1"
            &&
            (_START2_ < 40e6 ||_START2_ > 160e6)
```

Only links that are between hs1 and mm1 and which start before 40Mb or after 160Mb on mm1 are shown (Figure 24).

You can make the value of parameters set by a rule a function of the properties of the link. Certain keywords are parsed during rule evaluation and the corresponding property of the link is substituted. These are `_CHR1_` and `_CHR2_` for the start and end chromosome, `_START1_`, `_END1_` and `_START2_`, `_END2_` for the coordinates of the link start and end, as well as `_{PARAMETER}_` (e.g. `_COLOR_`, `_THICKNESS_`, etc).

Below we'll write a rule that changes the thickness, color and order of a link. To use Perl code in a rule, you must delimit the code using `eval()`. For example,

```
thickness = eval(2+2 . “p”)
```

is the same as

```
thickness = 4p
```

Let's make the thickness a function of the size of the start of the link. We'll want to map the thickness to a range between 1 and 5. This will do the job

```
thickness = eval( min( 5,max( 1, int( _SIZE1_/5e3 ) ) )  . “p”)
```

In this rule, first the size of the link is divided by `5000` and the result is truncated. For example, if `_SIZE1_ = 2600` then `int(_SIZE1_/5e3) = 0`. If `_SIZE1_ = 11000` then the value of the division is 2. We now limit the value between 1 and 5 by first taking `max(1,2) = 2` and then `min(5,2) = 2`.

The rules for color and `z` work similarly. We map the size onto colors `bd3-1` to `bd3-9`. The `z` parameter, which controls the order in which links are drawn, is simply made to be the result of the truncated division `_SIZE1_/5000` (largest links are drawn last).

The full rule changes `thickness`, `color` and `z` is

```
<rule>
importance  = 100
condition   = _CHR1_ eq "hs1" && _CHR2_ eq "mm1"
                               && (_START2_ < 40e6 ||_START2_  > 160e6)
thickness  = eval(min(5,max(1,int(_SIZE1_/5e3))) . "p")
color      = eval("bd3-"  . max(min(11,int(_SIZE1_/5e3)),1))
z          = eval(int(_SIZE1_/5e3))
</rule>
```

and the resulting image is shown in Figure 25.

You can change the color of the link based on the position of its ends. For example, to assign a color from the Brewer 11-color diverging palette (`bd3-1` ... `bd3-11`), this expression for color achieves the mapping

```
condition = _CHR1_ eq "hs1"
z         = eval(int(_SIZE1_/5e3))
color     = eval("bd3-" . max( 1, int(_START1_/220e5)))
```

The `_START1_` keyword is replaced by the start position of the beginning of the link. The truncated division maps it to a value in the range `0-11` (size of hs1 is ~247Mb and we know that the link starts on this chromosome because of the condition of the rule). We're again using the condition that selects all links on hs1. The result of this color mapping is a rainbow effect shown in Figure 26.

Figures

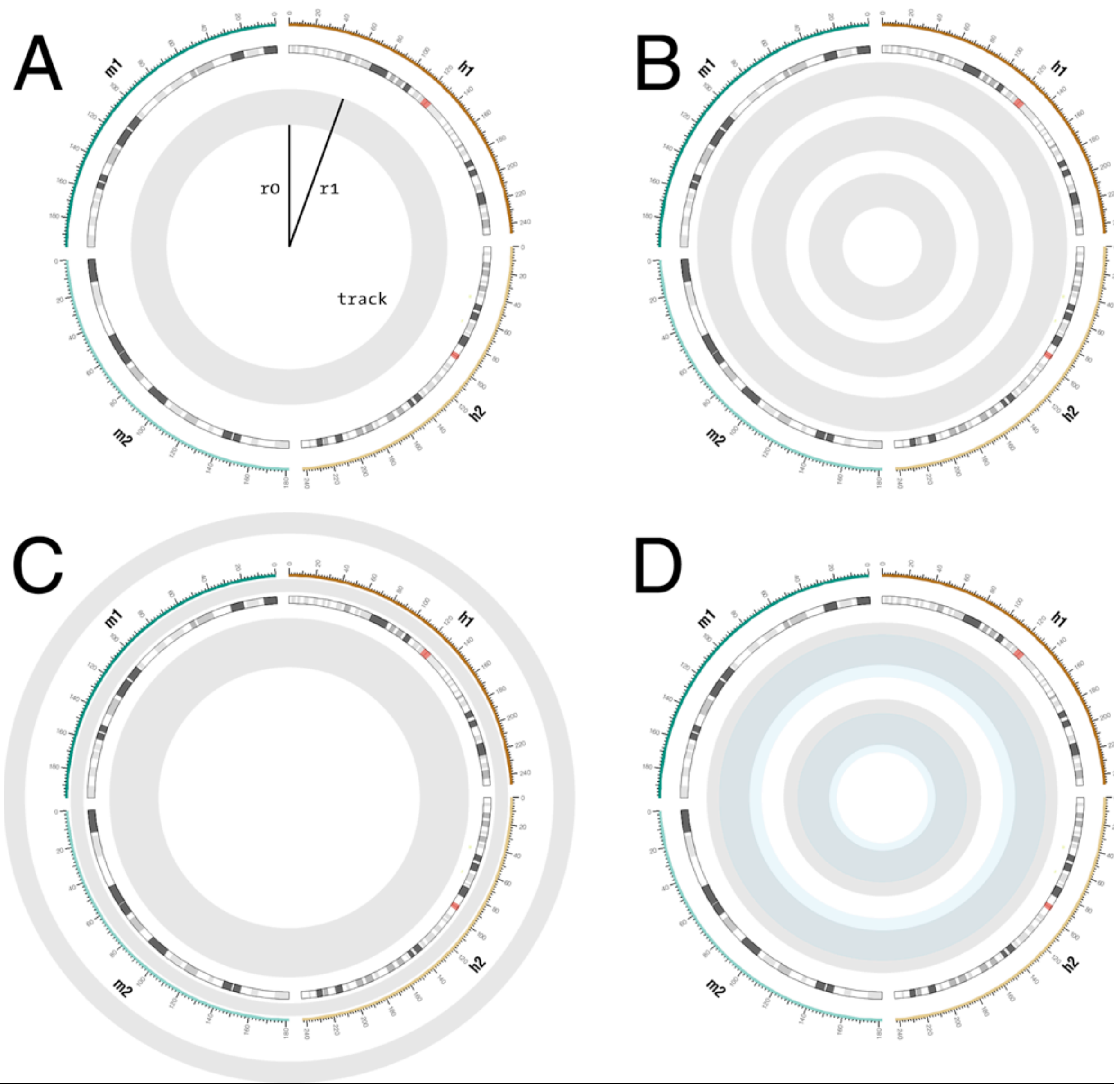


FIGURE 1

(A) Each data track confined to an annulus bounded by radii `r0` and `r1`. (B) Any number of tracks can be placed on the figure, and (C) at any radial position, including inside/outside ideogram circle and inside/outside ticks. (D) Tracks can be made to overlap and the order in which they are drawn is controlled by the `z` parameter.

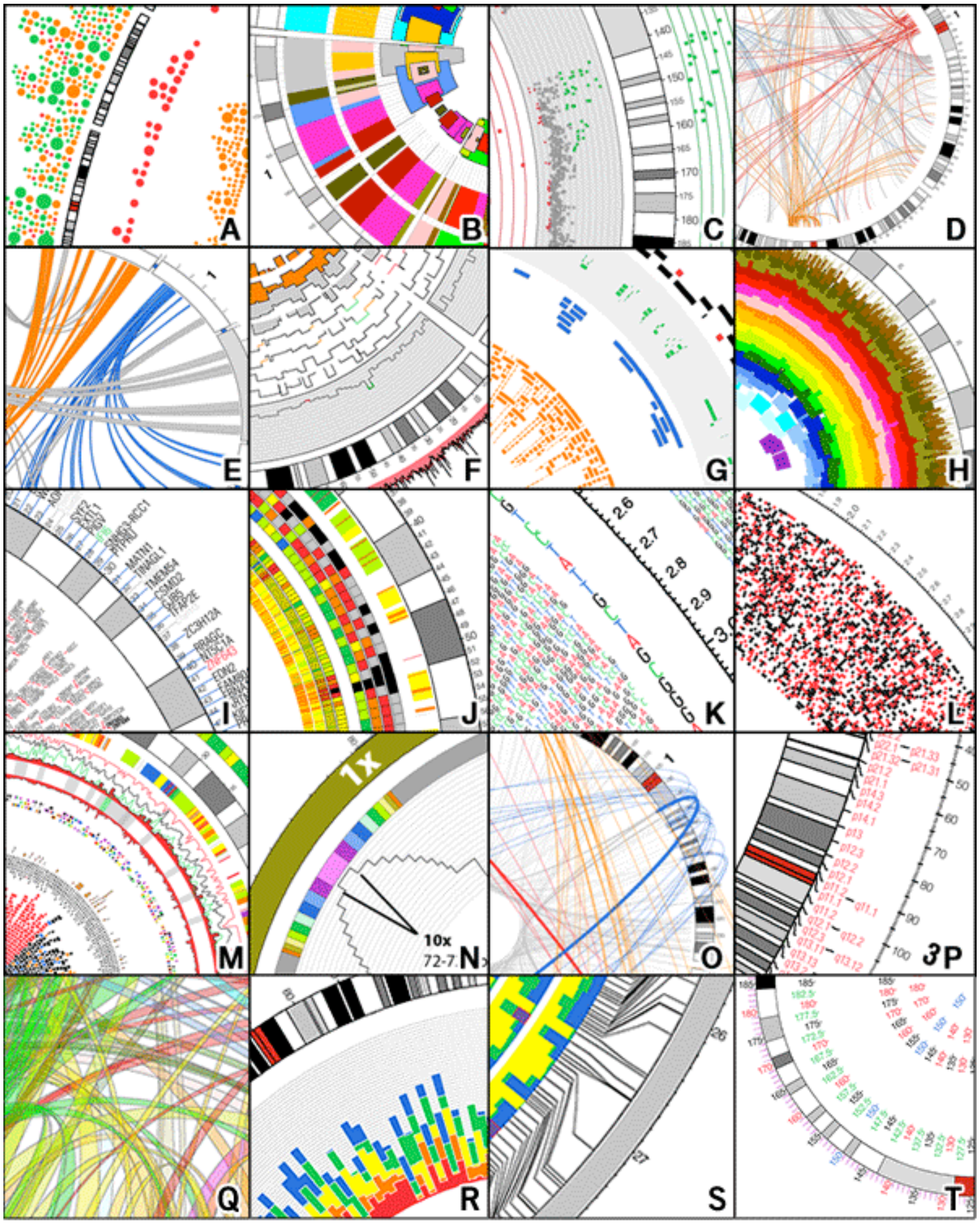


FIGURE 2

Some examples of Circos plots. (A) glyph (B) highlight with depth control (C) scatter (D) paired-location (E) ribbon (F) histogram (G) tile (H) highlight with auto depth (I) text with auto arrange (J) heat map (K) high-density text (L) high-density glyph (M) multi-type composite (N) variable scale control (O) fine geometry control (P) flexible text and element placement (Q) transparent ribbons (R) stacked histogram (S) connectors (T) tick rings.

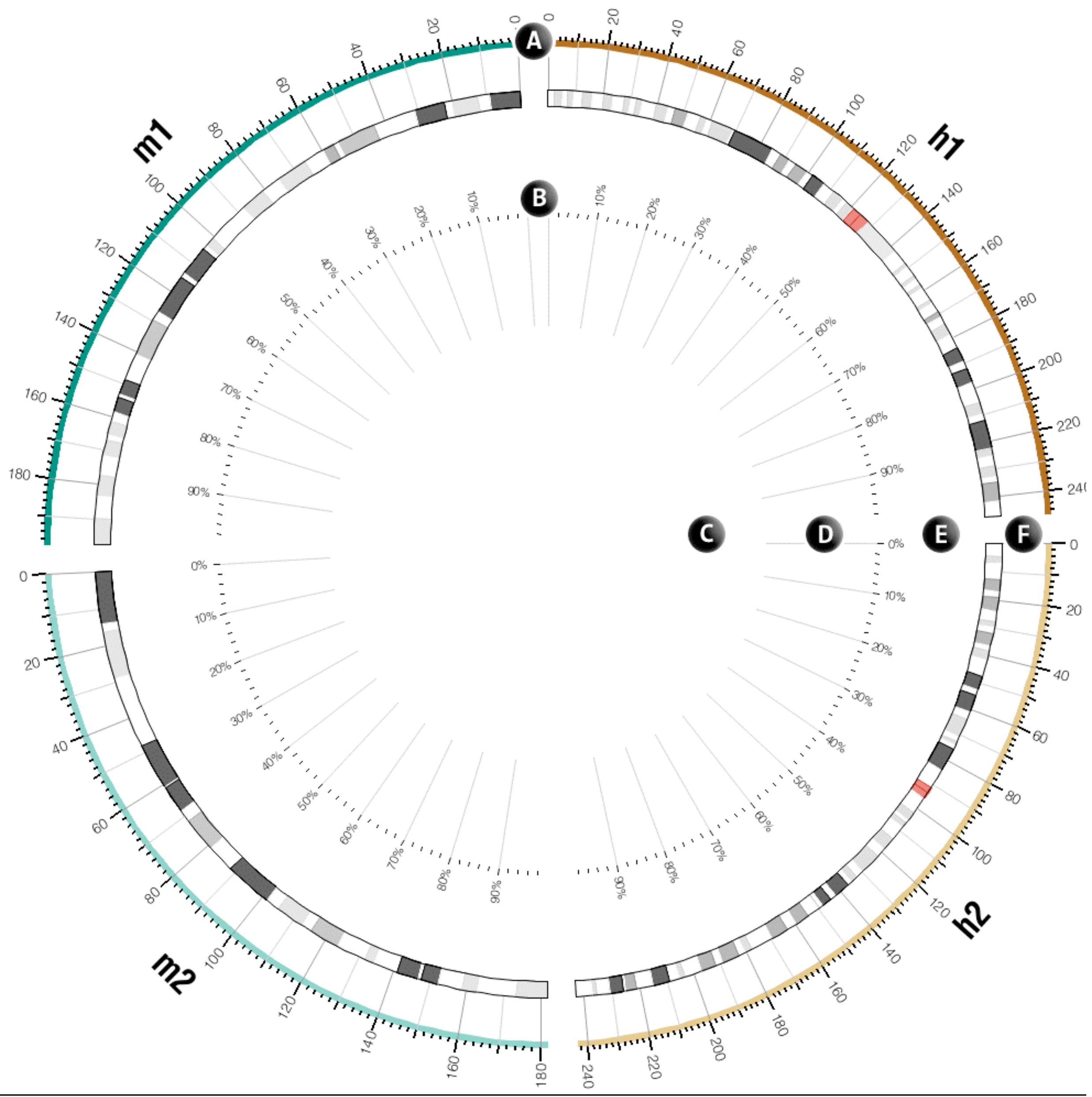


FIGURE 3

This figure shows the ideogram, grid and tick layout that will be used for data tracks in this series of lessons. In the figure are human chromosomes 1 and 2 (labeled h1 and h2) and mouse chromosomes 1 and 2 (labeled m1 and m2). Note the orientation of the scale – clockwise for human and counter-clockwise for mouse ideograms.

There are three groups of absolute ticks (A), spaced at 20Mb, 10Mb and 2Mb. Ticks at 20Mb and 10Mb have a grid.

There are two groups of relative ticks (B), spaced at 10% and 2%. Ticks spaced at 10% have a grid.

Data tracks will be added to regions C, D, E and F.

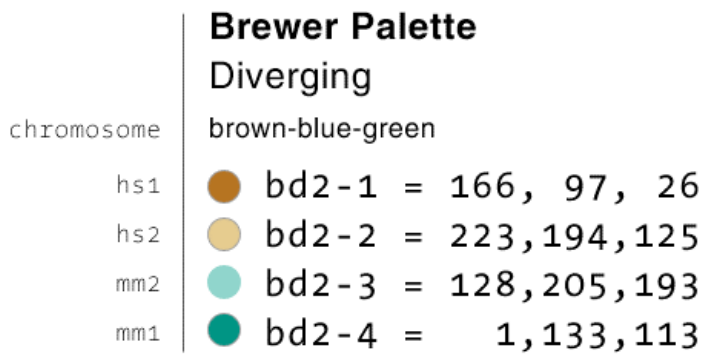


FIGURE 4

The ideograms are color coded using the 4-color diverging brown-blue-green Brewer palette, shown here.

```
chromosome   color name
hs1          bd2-1
hs2          bd2-2
mm2          bd2-3
mm1          bd2-4
```

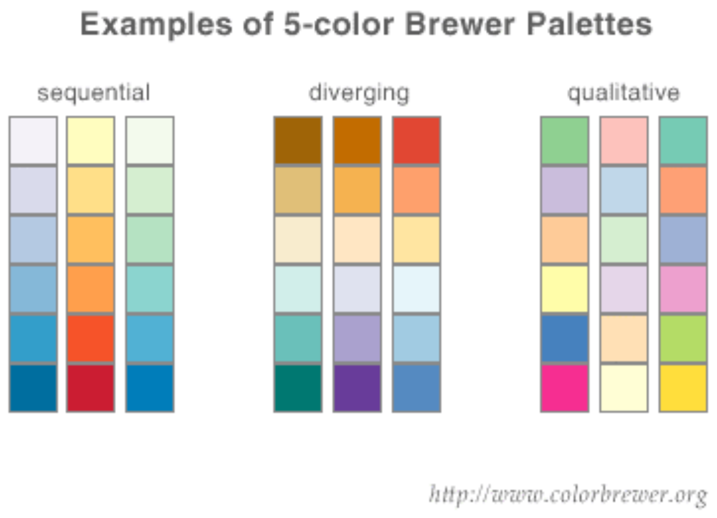

Examples of 5-color Brewer Palettes
sequential
diverging
qualitative
http://www.colorbrewer.org

FIGURE 5

Examples of 5-color sequential, diverging and qualitative palettes.

For palette type, several color combinations are available, ranging from 3-12 colors.

For more information about Brewer palettes, see *www.colorbrewer.org*.

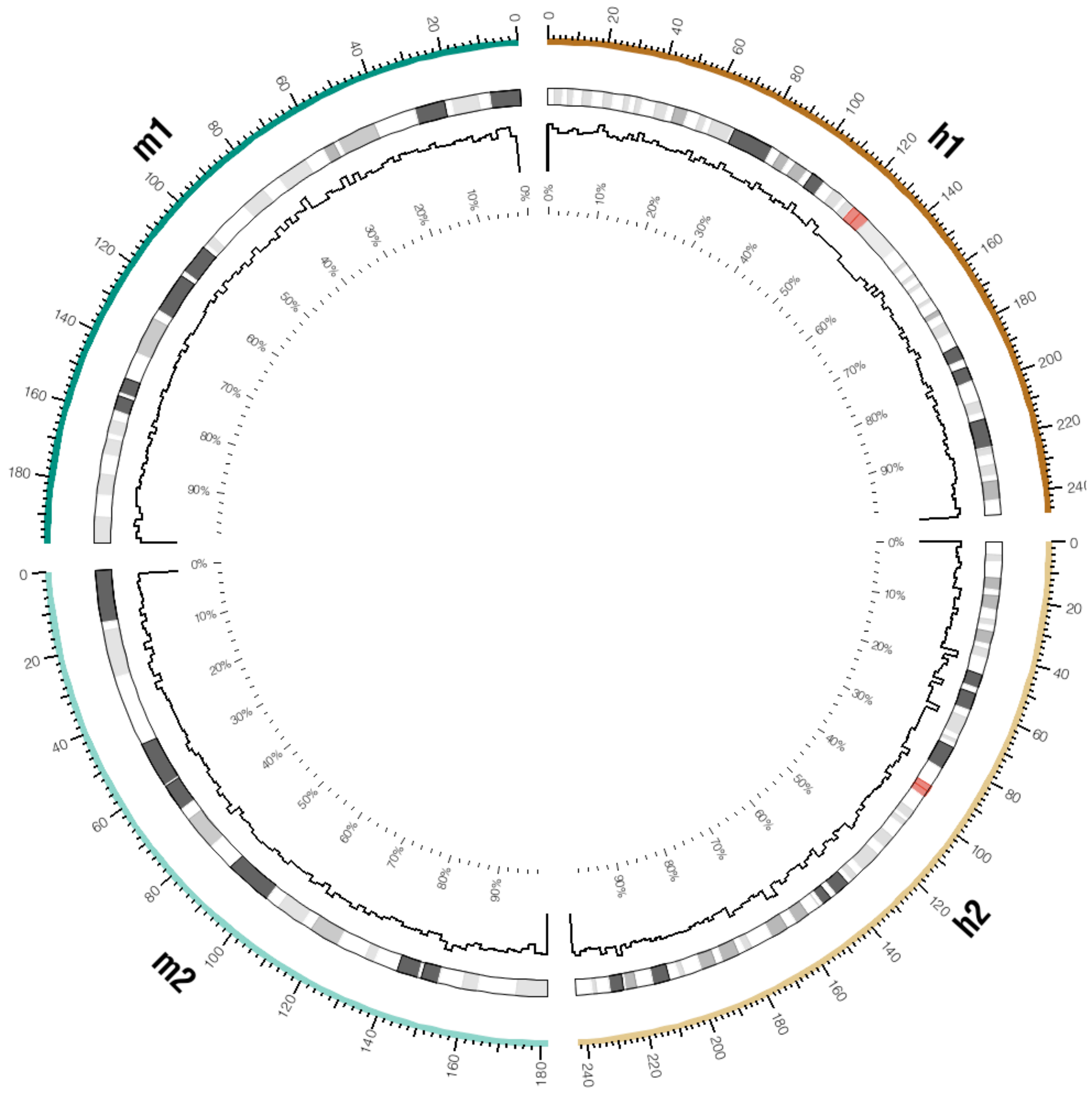


FIGURE 6

A histogram data track located within the annulus bounded by `r0 = 0.85r` and `r1 = 0.975r`.

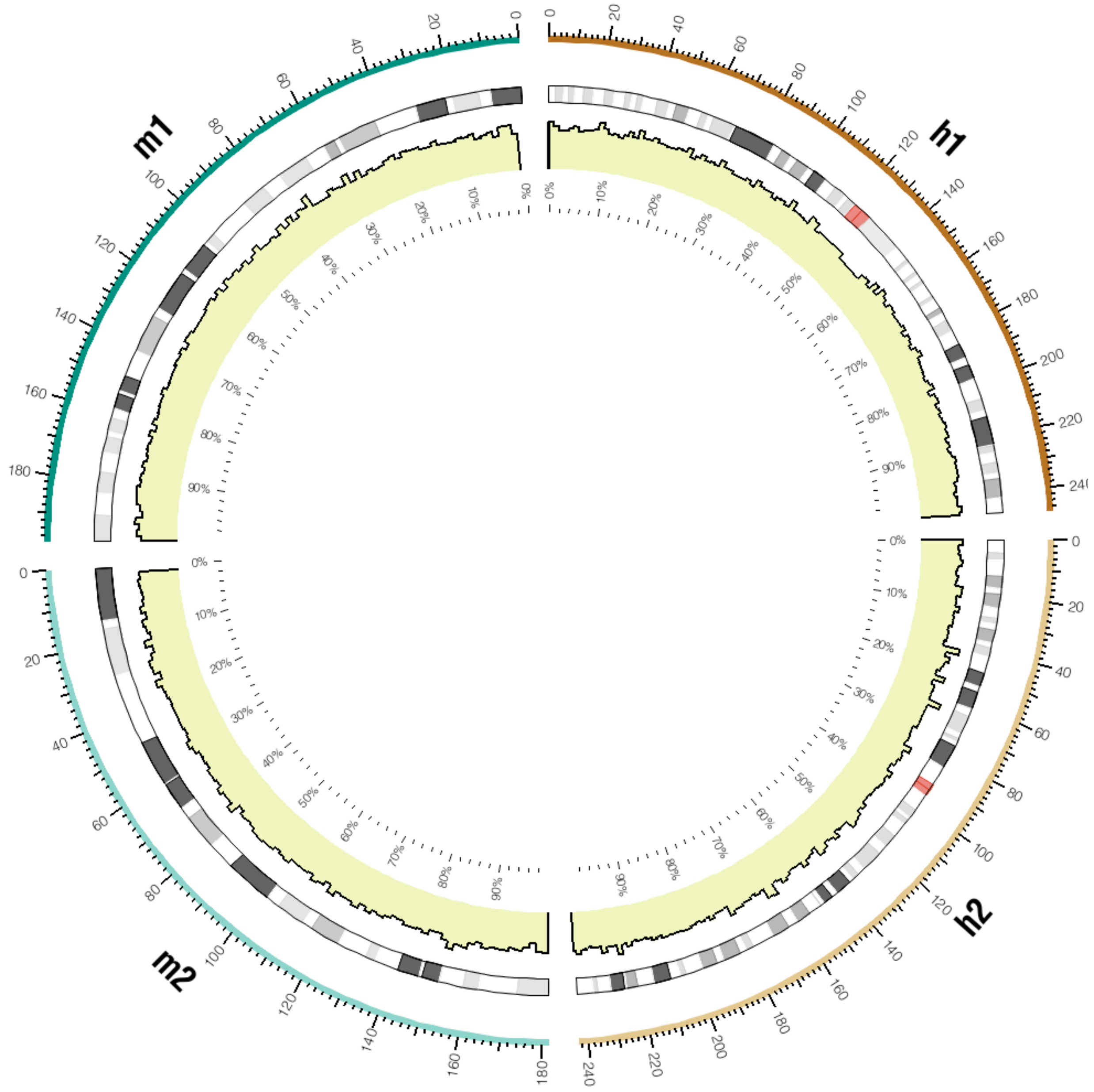


FIGURE 7

The histogram has been adjusted to include a fill color, using `fill_under` and `fill_color` parameters.

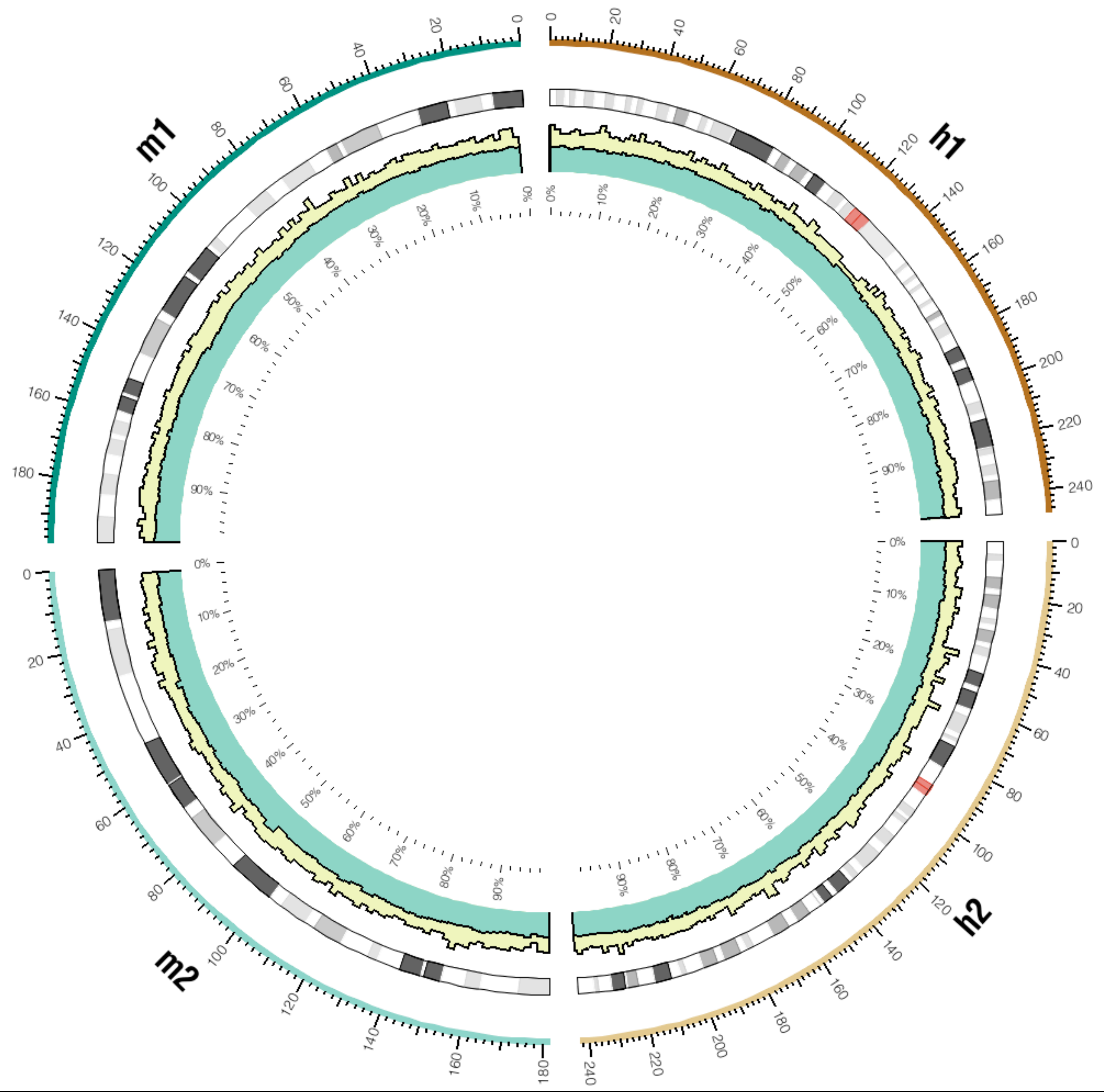


FIGURE 8

A second histogram has been added, drawn on top of the first histogram.

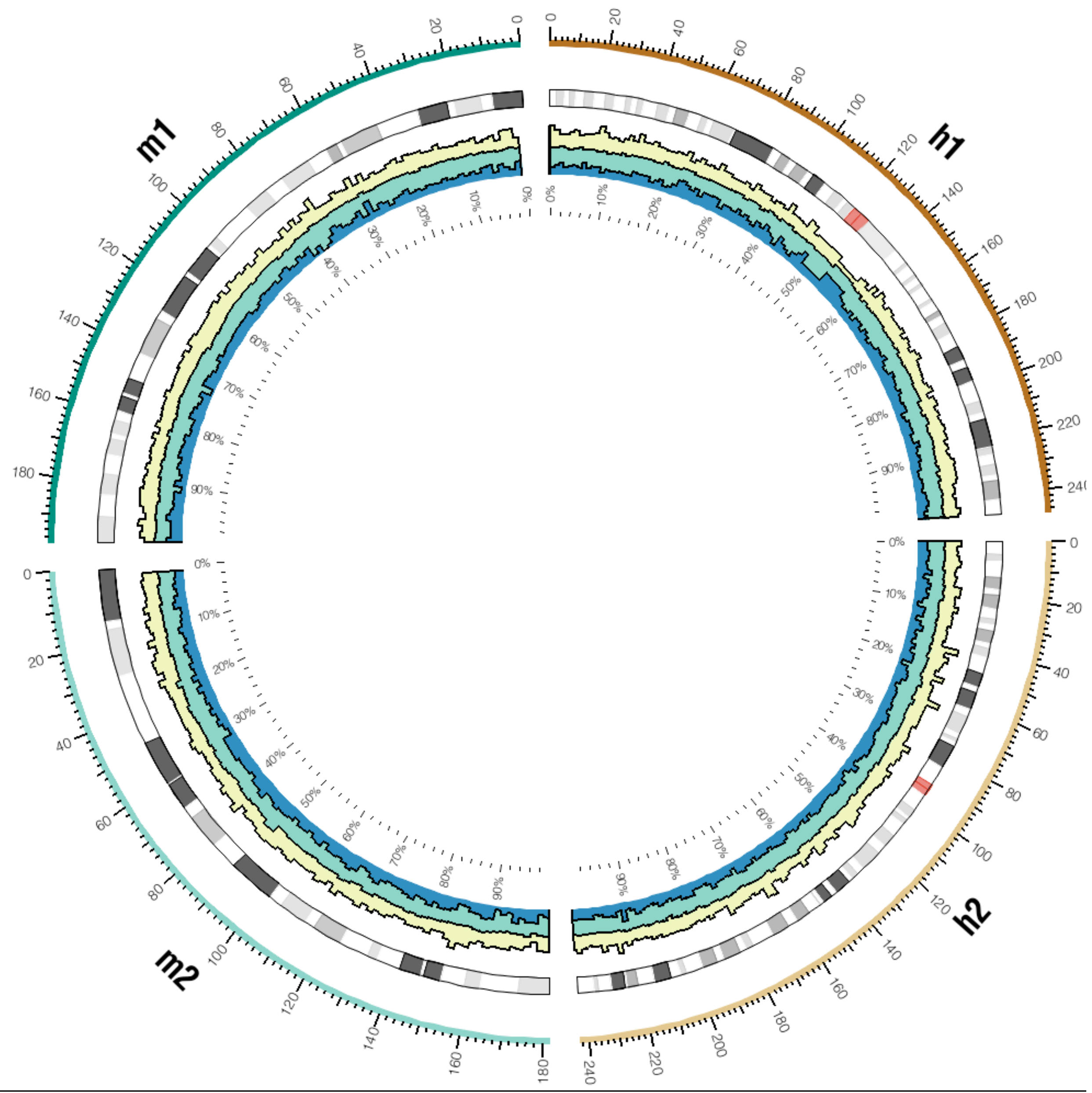


FIGURE 9

A third histogram has been added, drawn on top of the first two histograms.

The region between the relative ticks and the ideogram now shows three histograms, each with a different fill color.

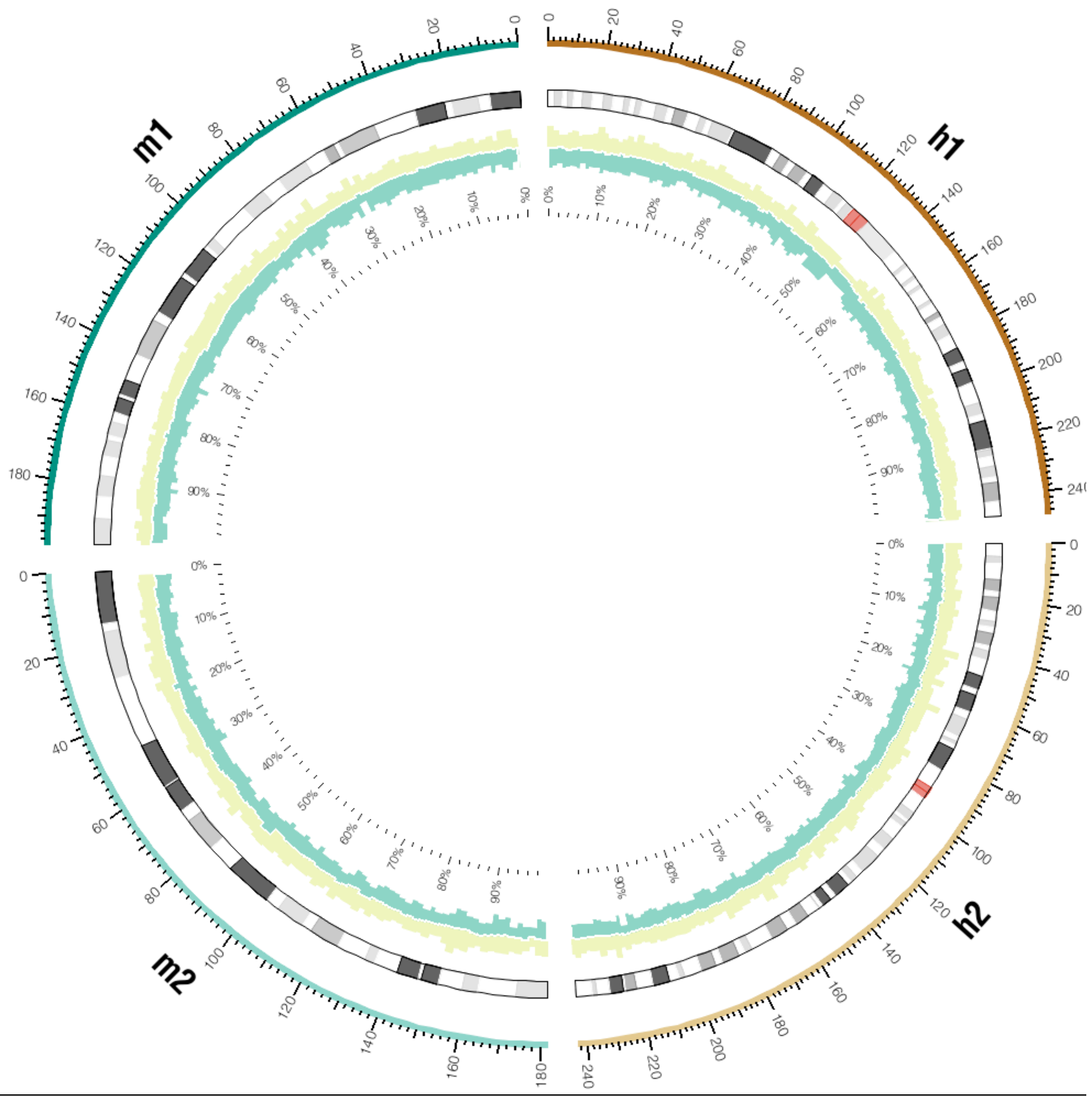
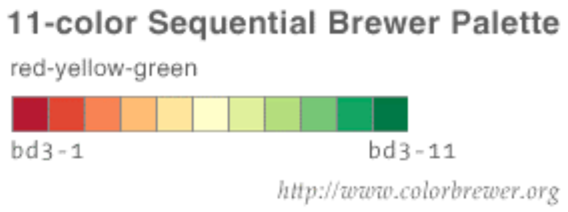


FIGURE 10

By removing the outline on all histograms and setting the fill of the bottom histogram to white, we can create a compound data tracks which effectively shows low/mid/high range.


11-color Sequential Brewer Palette
red-yellow-green
bd3-1
bd3-11
http://www.colorbrewer.org

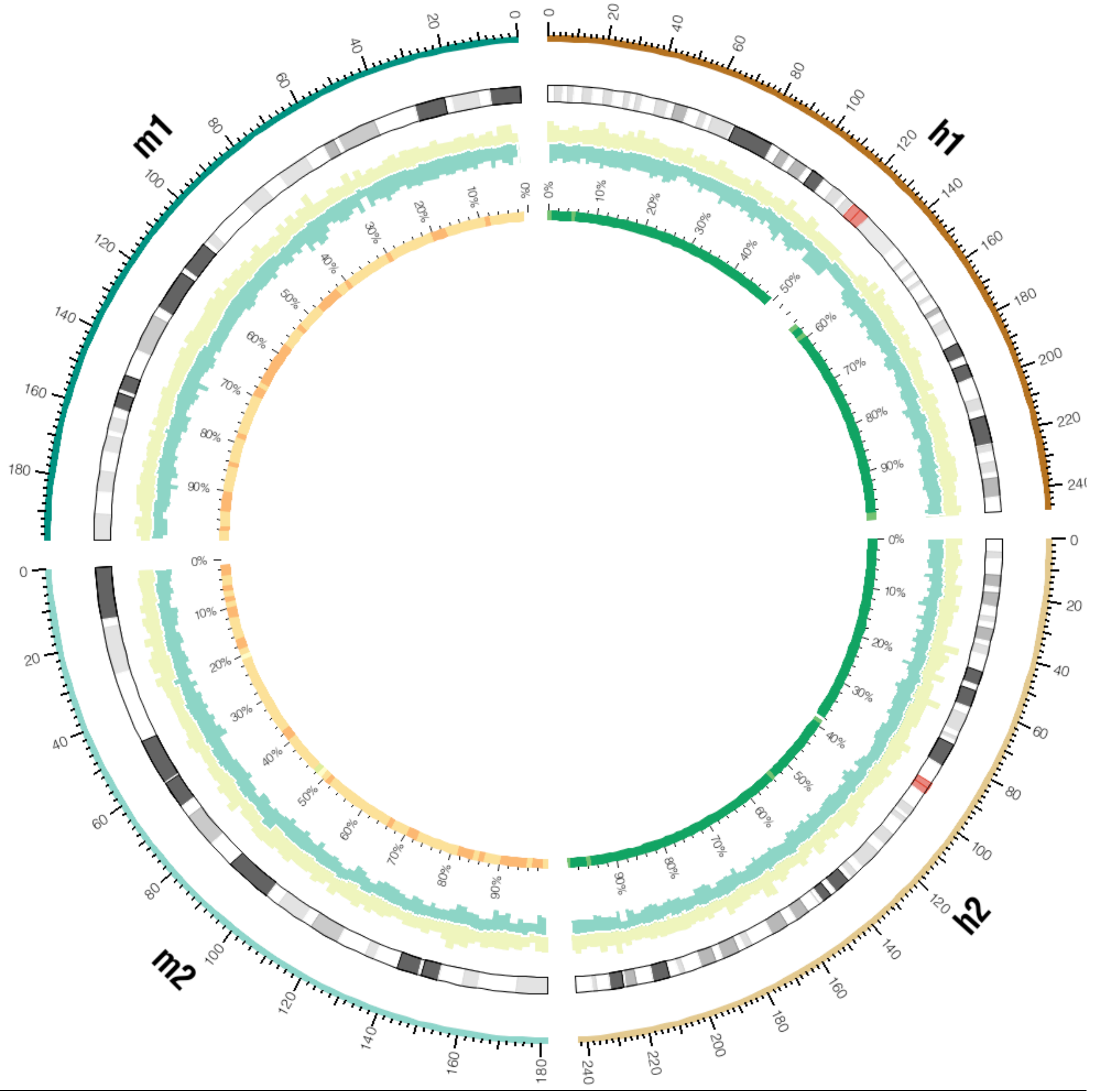


FIGURE 11

A heat map shows the average conservation score in 2Mb bins relative to the rhesus genome. Note that the human genome has a high degree of conservation with rhesus, since both are primates. Mouse, on the other hand, has a significantly lower conservation.

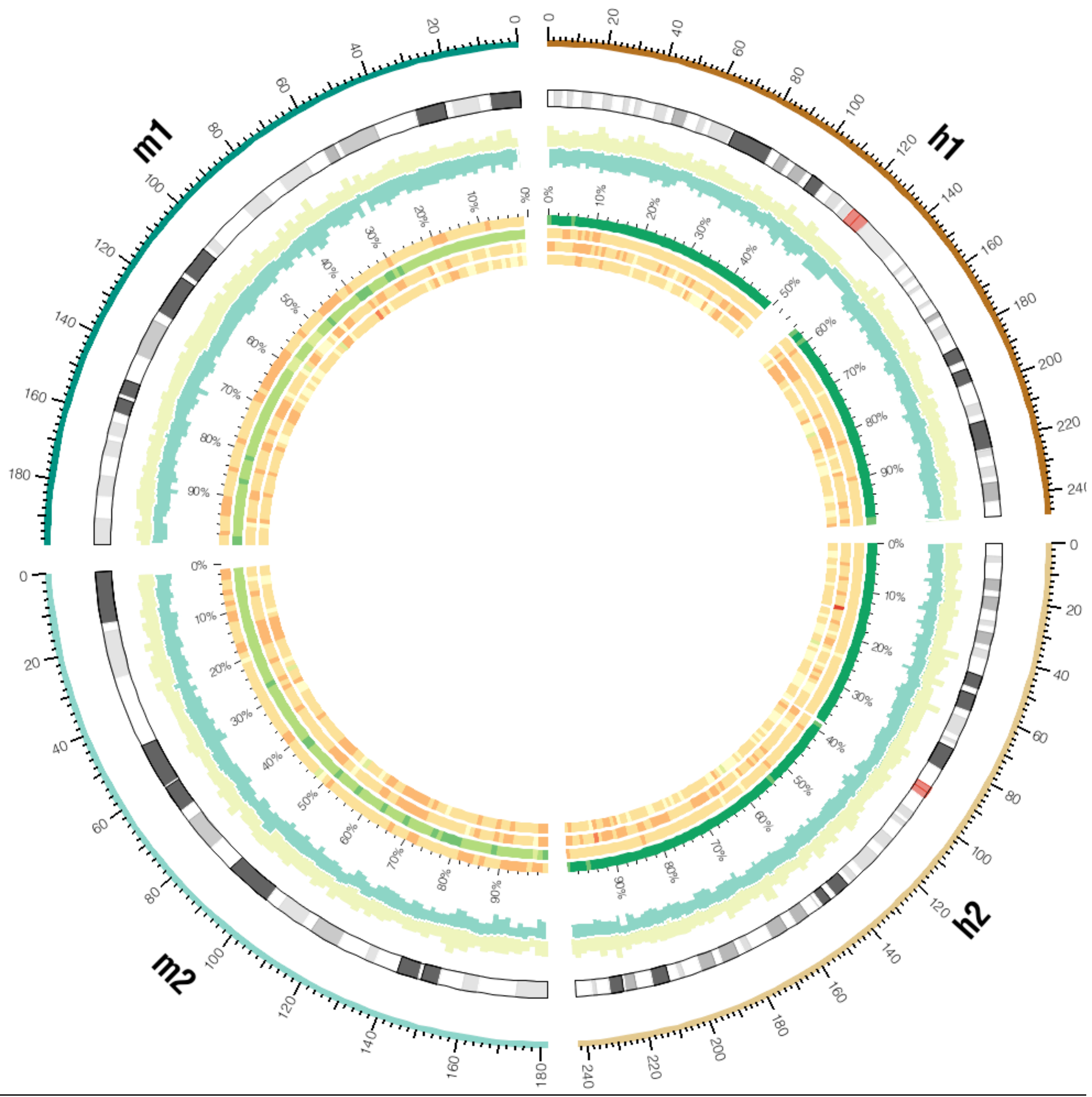


FIGURE 12

Heat maps show the average conservation score in 2Mb bins relative to the rhesus (outer heatmap), rat, zebra fish and fugu (inner heatmap) genomes.

Note that the human genome has a high degree of conservation with rhesus, since both are primates. Mouse, on the other hand, has a significantly lower conservation with rhesus but high conservation with rat (second heatmap).

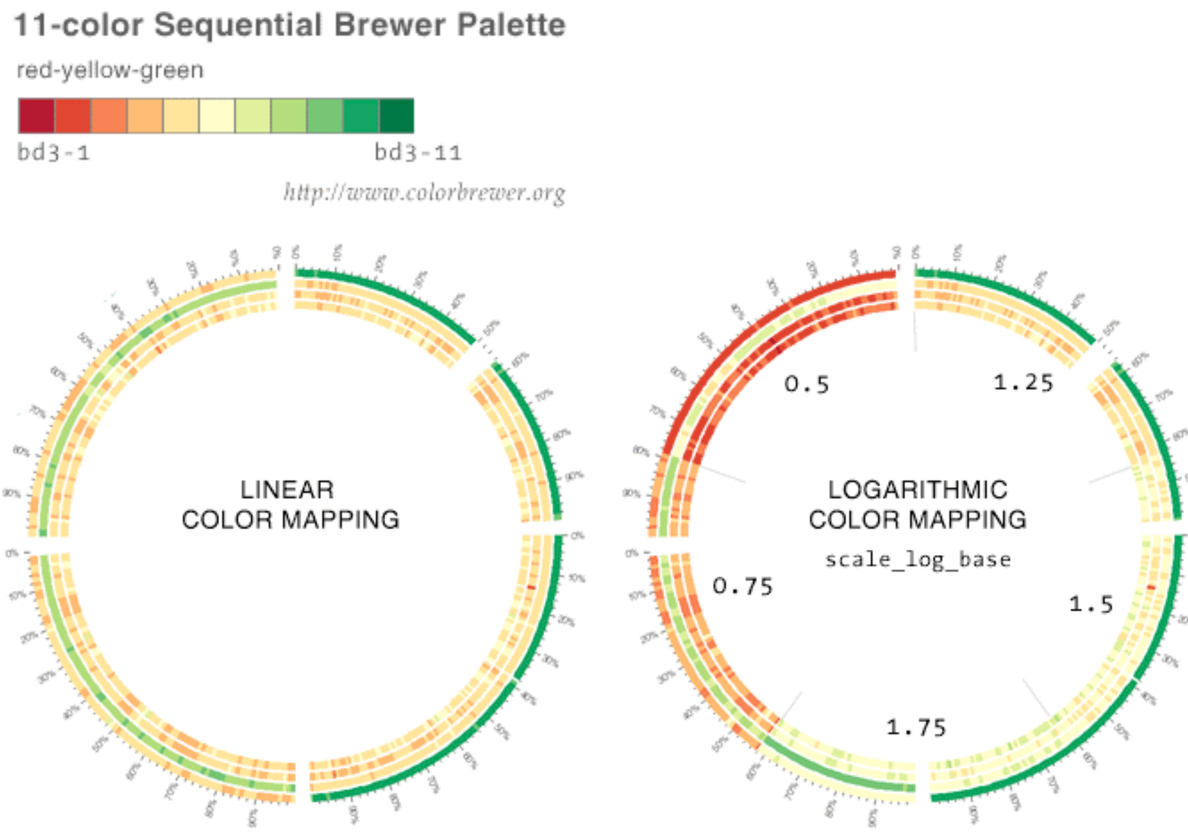


FIGURE 13

Color mapping in a heat map can be altered from its default linear mapping (left) to logarithmic (right) using the `scale_log_base` parameter.

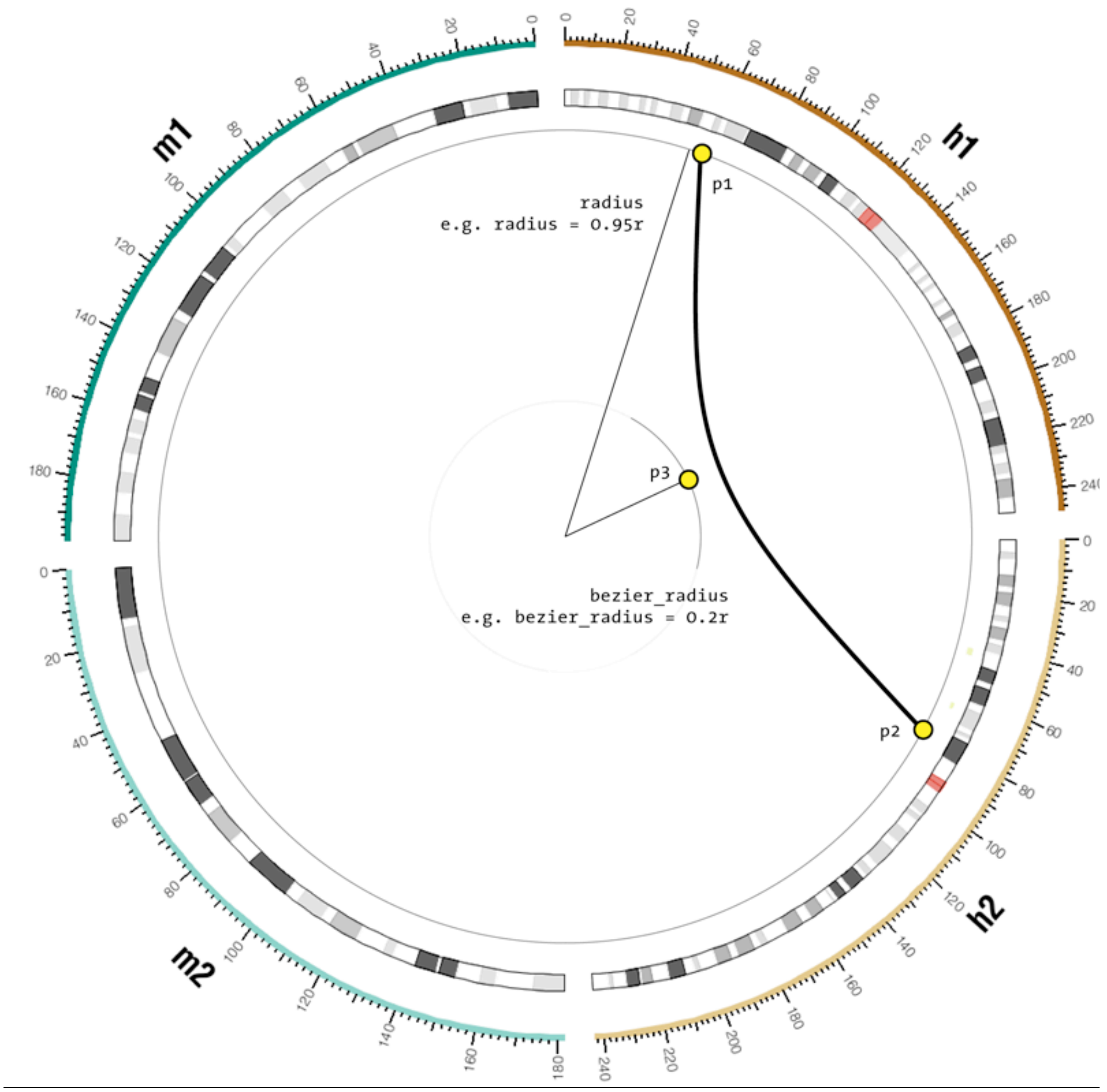


FIGURE 14

Links in Circos are drawn as quadratic Bezier curves which are specified by two ends (p1, p2) and a control point (p3).

The control point determines the direction of the curve at its ends and is placed midway (in angle) between the ends.

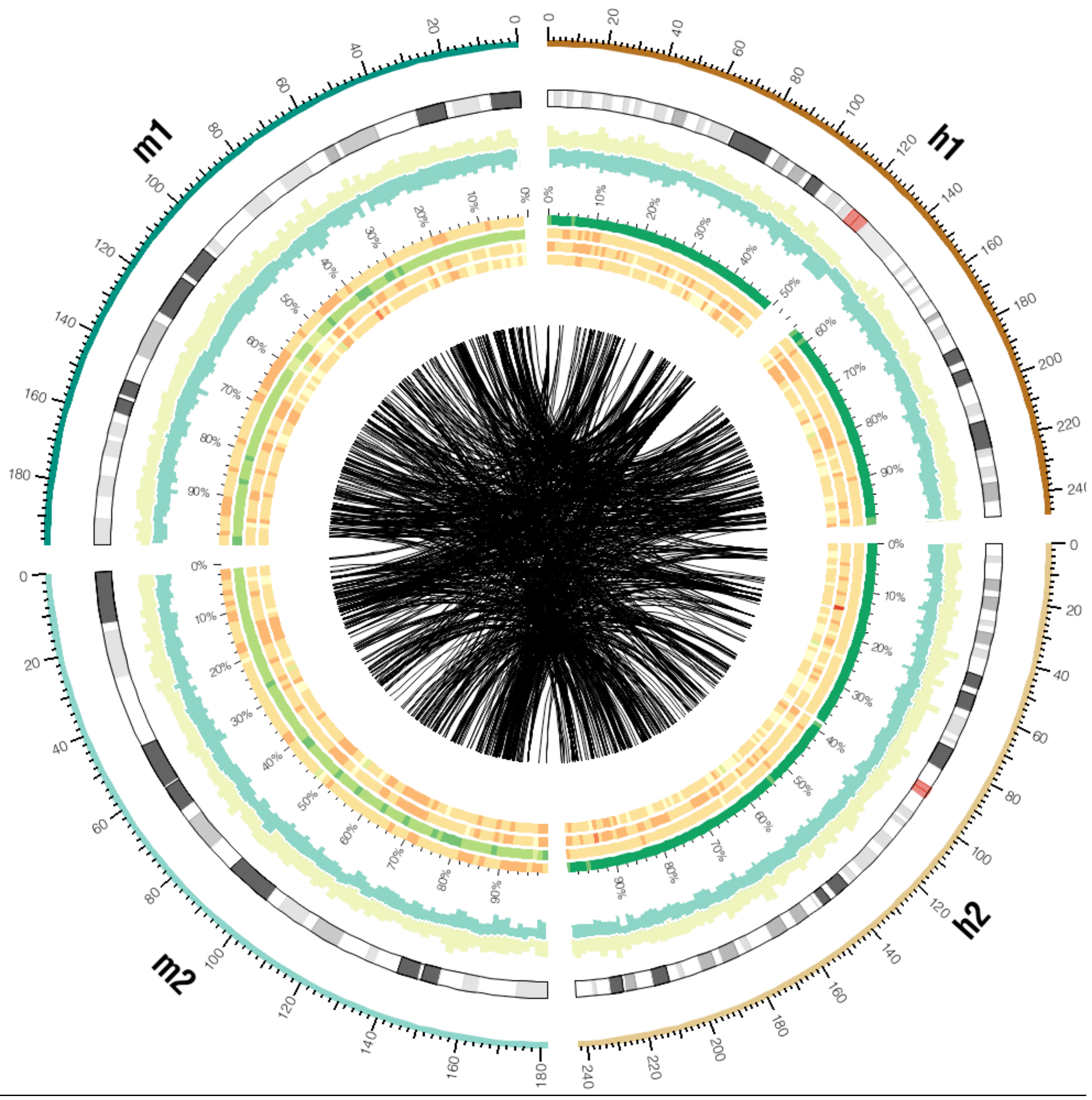


FIGURE 15

Links connecting regions with high sequence similarity between human and mouse chromosomes.

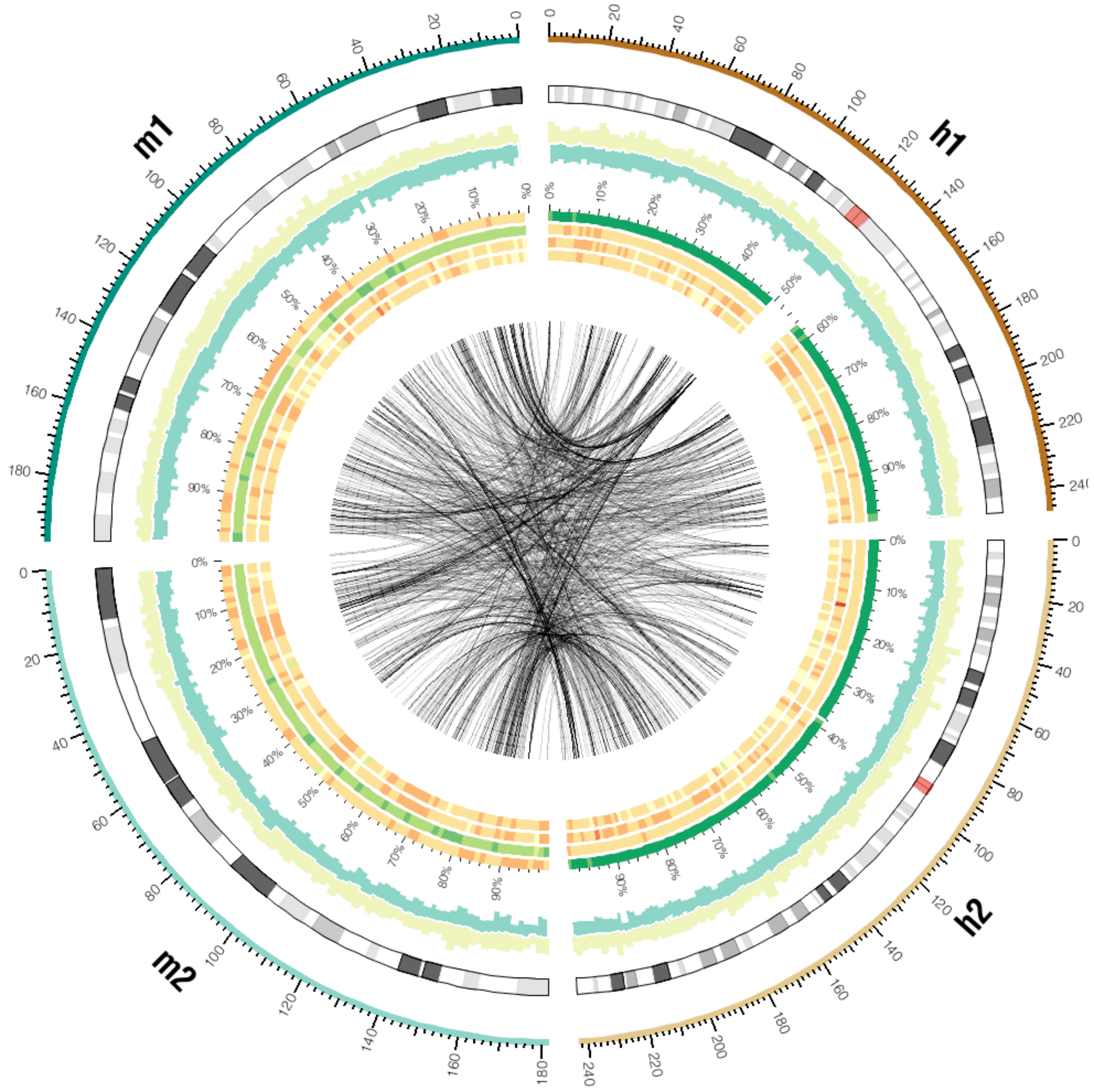


FIGURE 16

By making the link curves transparent, patterns in the data can be more easily discerned when the links are dense.

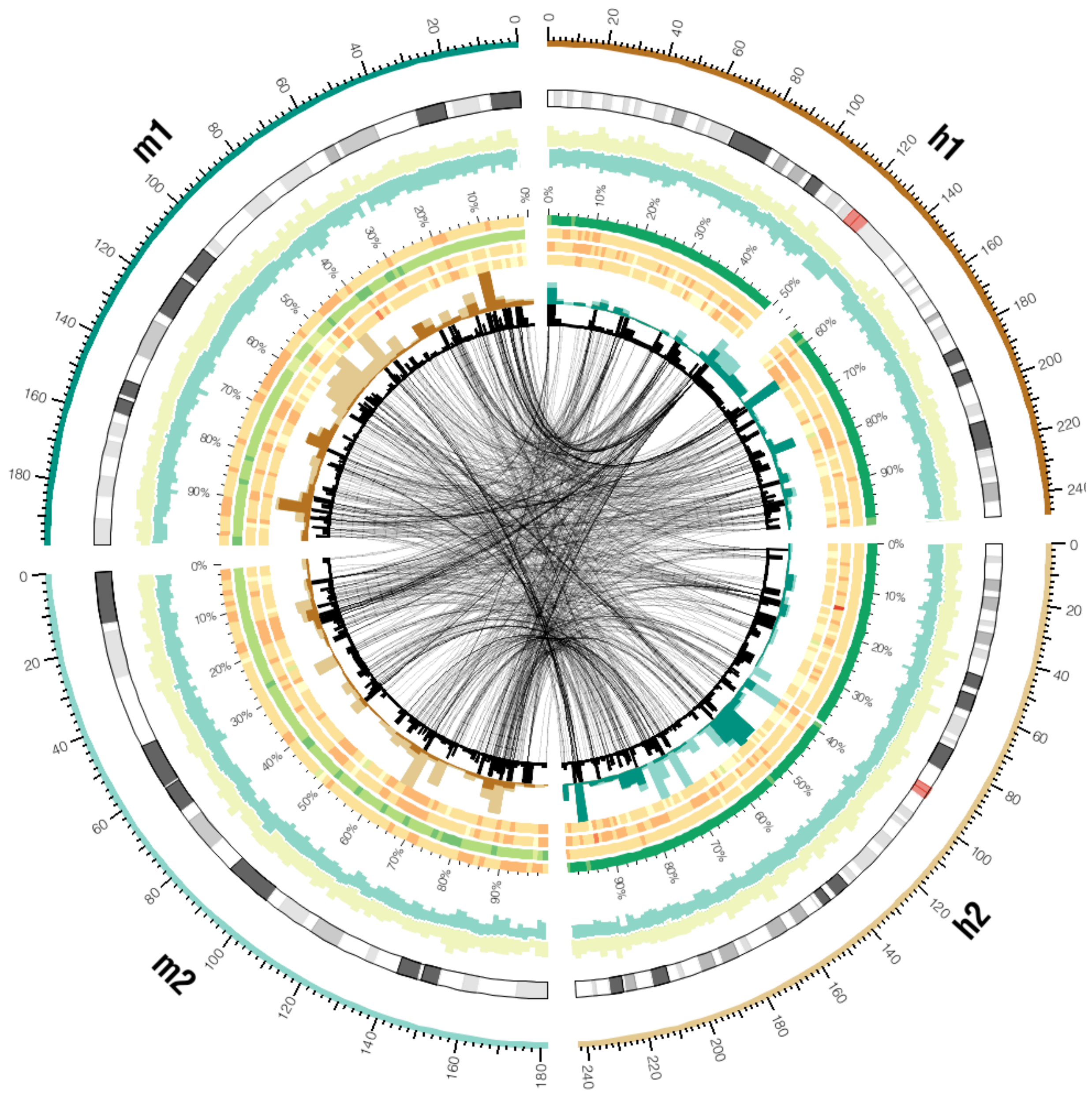


FIGURE 17

The two histograms between the heatmaps and links represent link density and were created by the `binlinks` tool.

The inner histogram (black) simply shows the number of link ends within a 1Mb bin. The outer stacked histogram (multi-colored, 5Mb bin) shows the total size of links outgoing from a bin for each target chromosome.

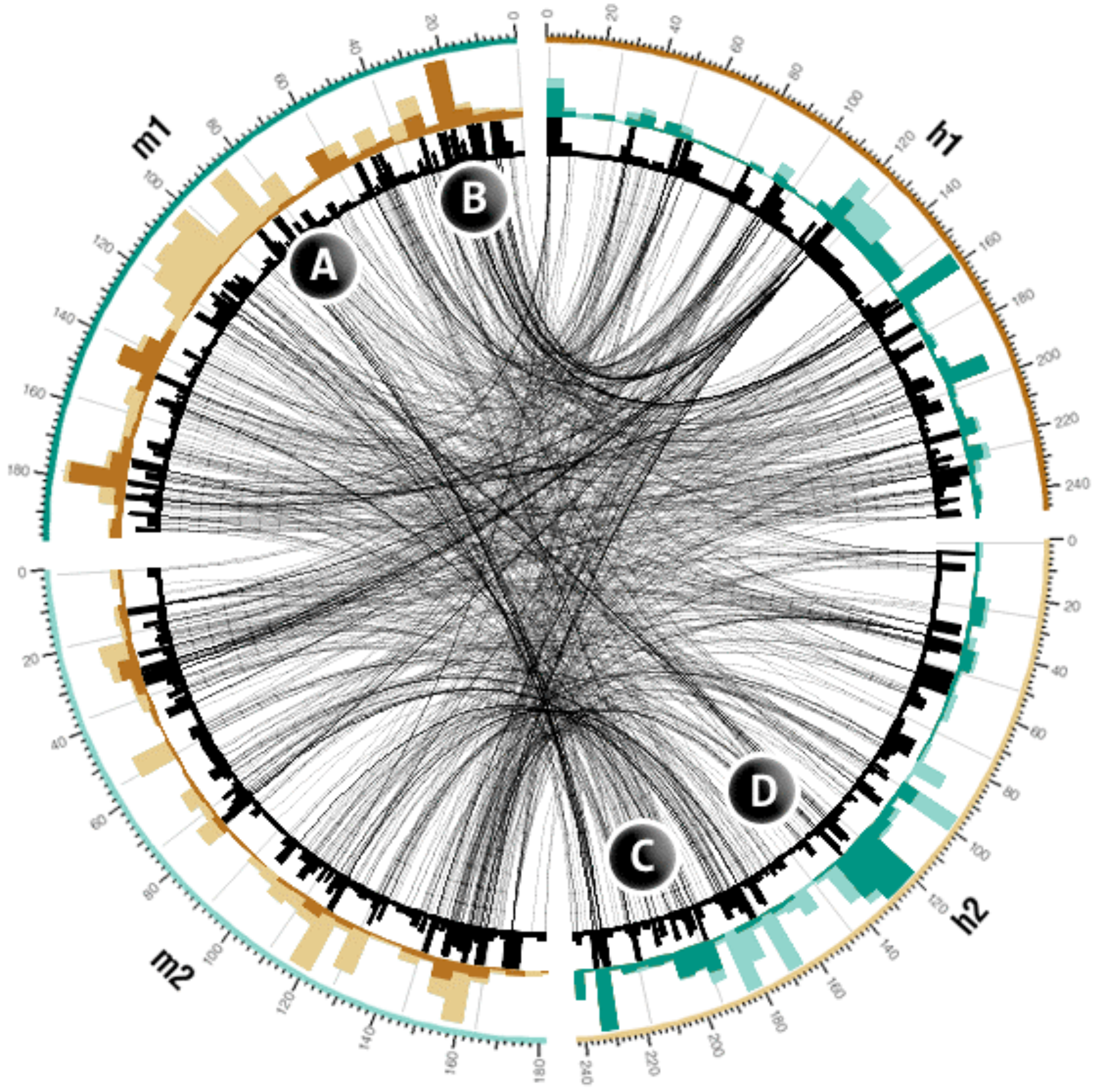


FIGURE 18

The two histograms between the heatmaps and links represent link density and were created by the `binlinks` tool.

The inner histogram (black) simply shows the number of link ends within a 1Mb bin. The outer stacked histogram (multi-colored) shows the number of links outgoing from a bin for each target chromosome.

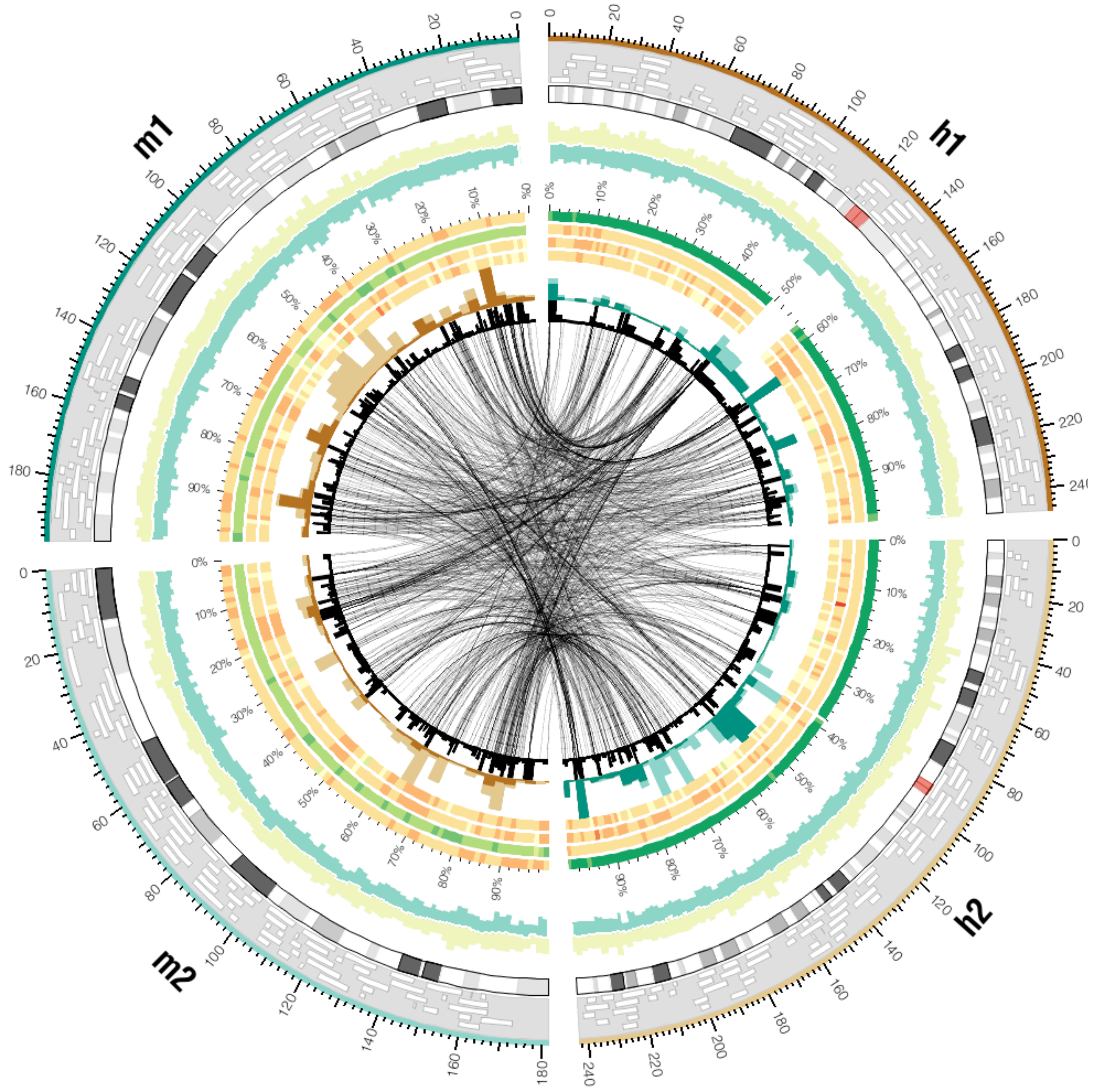
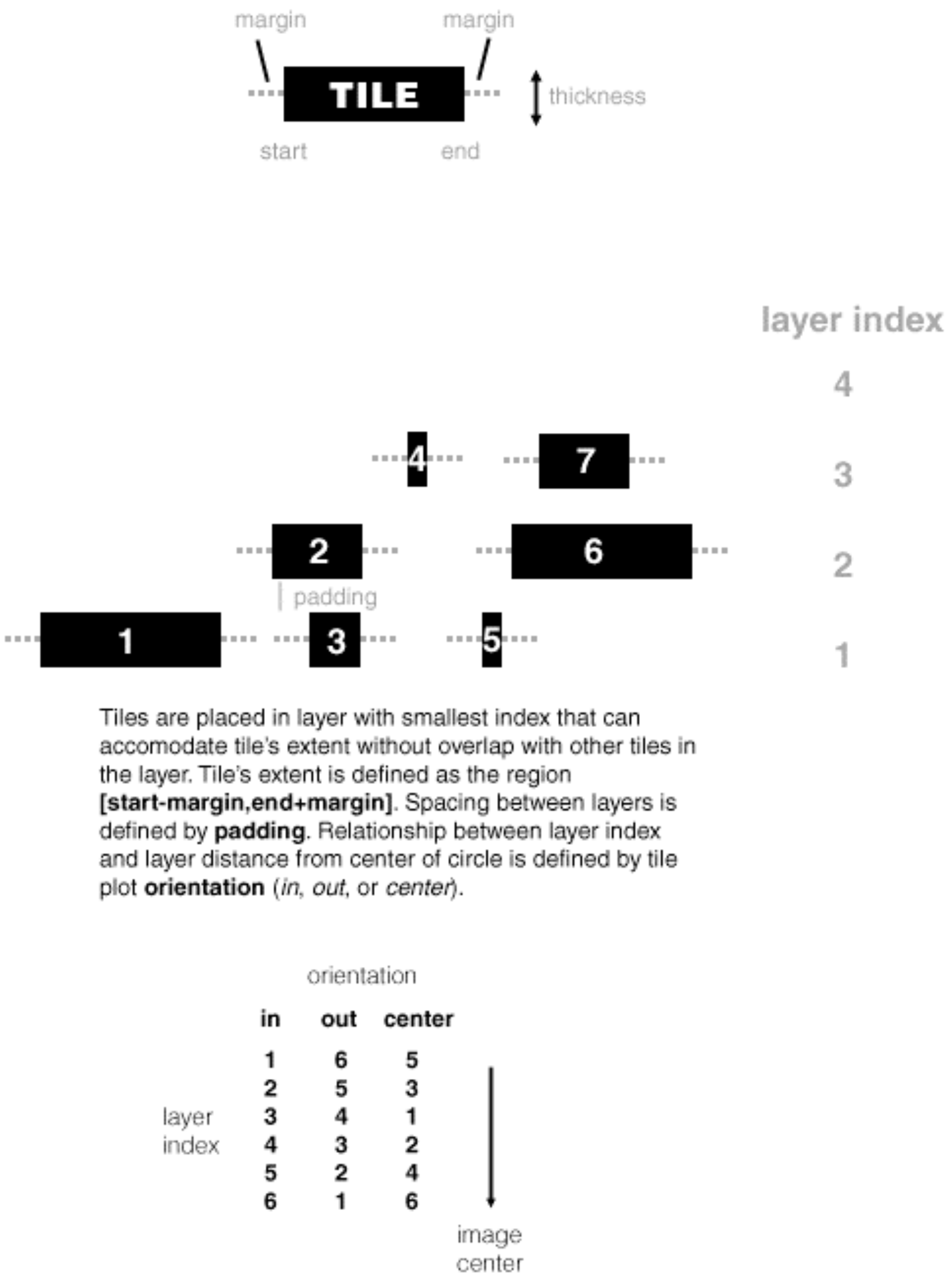


FIGURE 19

A tile track is shown outside the ideogram circle.

Using `background_*` parameters, the track is given a grey background.



Tiles are placed in layer with smallest index that can accomodate tile's extent without overlap with other tiles in the layer. Tile's extent is defined as the region **[start-margin,end+margin]**. Spacing between layers is defined by **padding**. Relationship between layer index and layer distance from center of circle is defined by tile plot **orientation** (*in*, *out*, or *center*).

	orientation		
	in	**out**	**center**
layer index	1	6	5
	2	5	3
	3	4	1
	4	3	2
	5	2	4
	6	1	6

FIGURE 20

Illustration of how tiles are placed in layers. Parameters such as `thickness`, `margin`, `padding` and `layers` control placement.

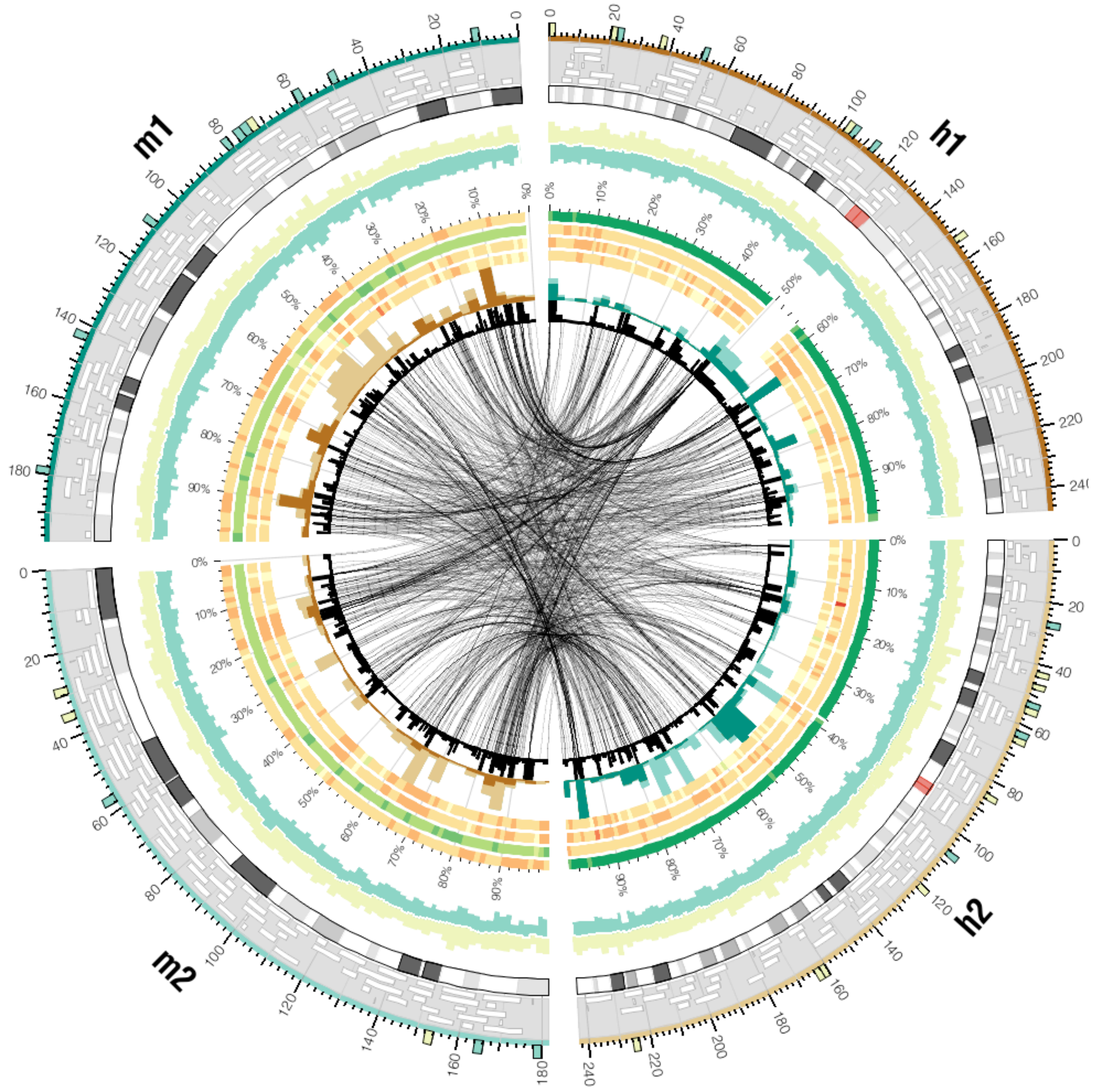


FIGURE 21

Two highlight tracks have been added to indicate the position of regions with very low and very high conservation.

The grid has been also toggled on. Notice how the transparent background of the tile track allows the grid to show through.

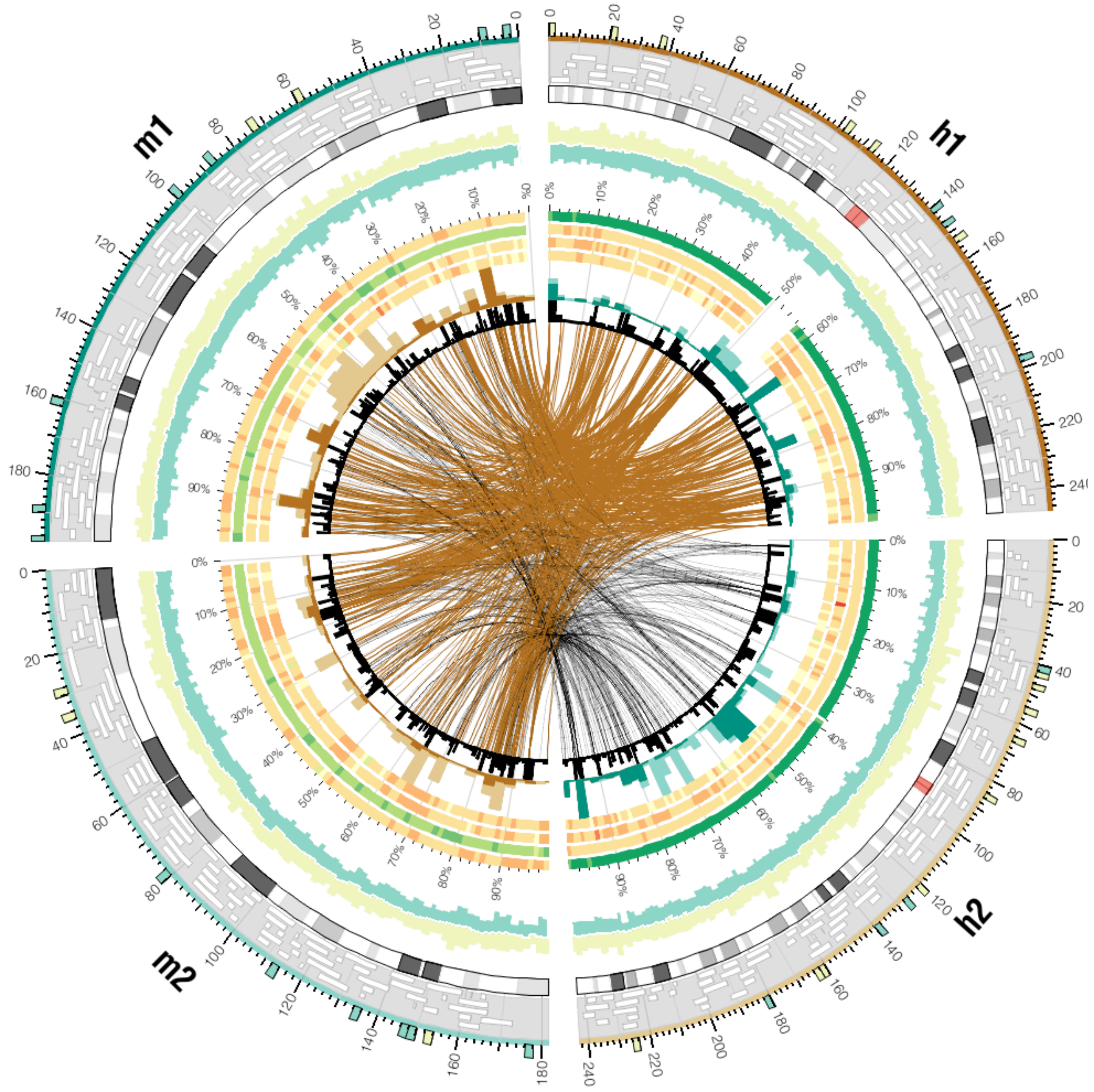


FIGURE 22

Using a rule, all links that start on hs1 are colored brown.

```
<rule>
condition = _CHR1_ eq "hs1"
color     = bd2-1
</rule>
```

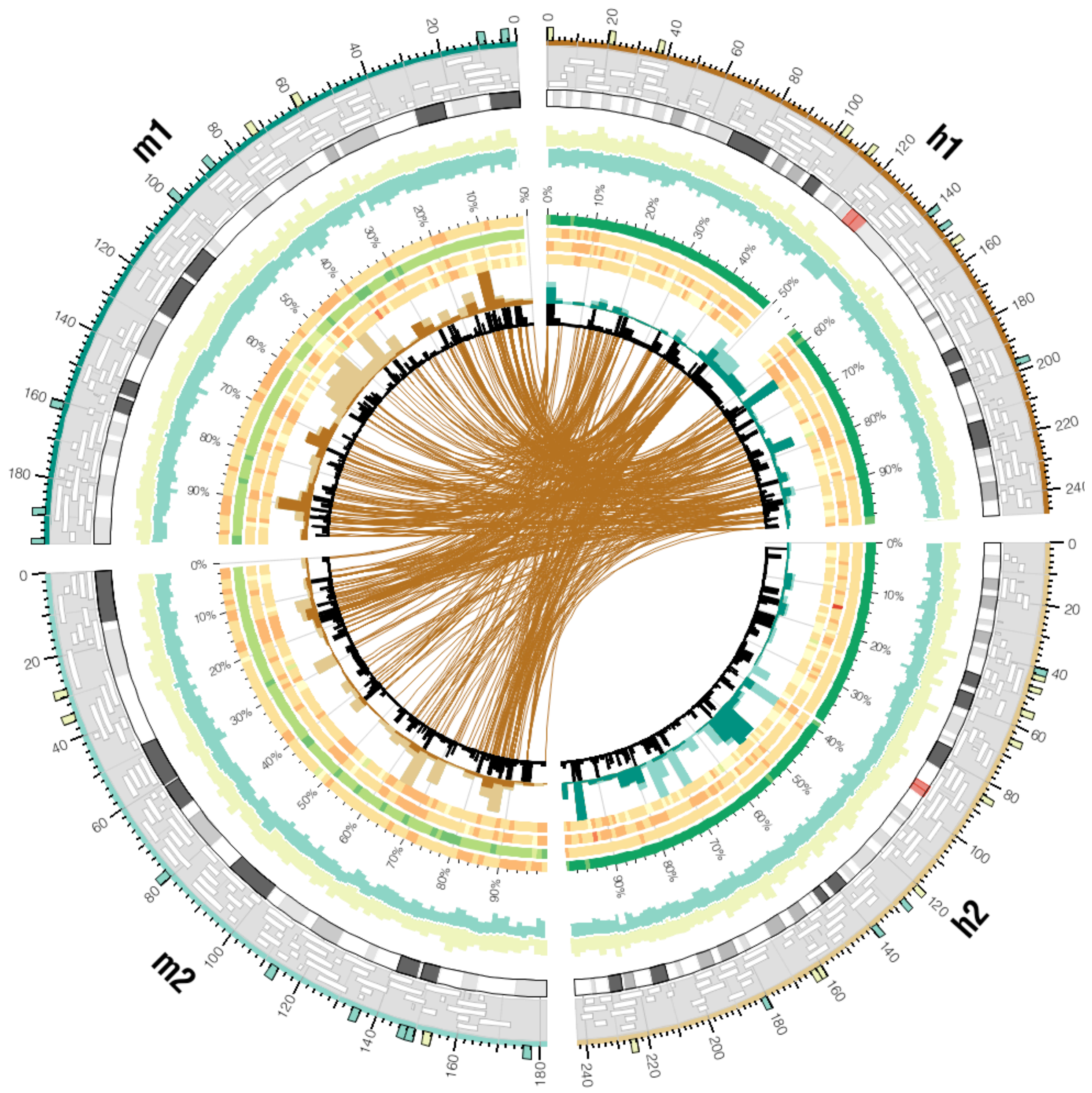


FIGURE 23

By adding a second rule that hides links, links that fail to match the first rule are not shown.

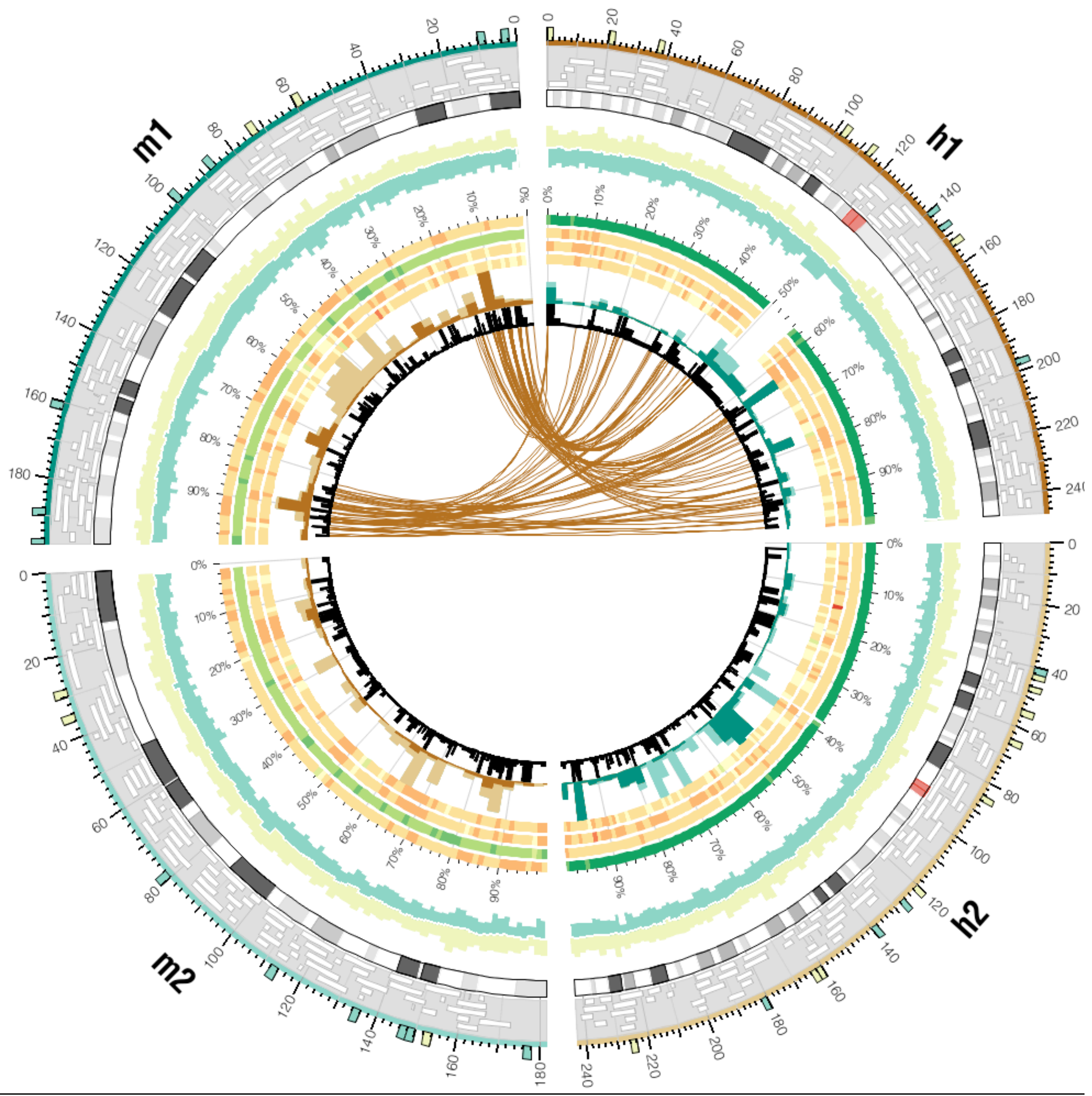


FIGURE 24

Rules can be used to select links based on position. Here, rules are used to show only links between hs1 and mm1, which start before 40Mb or after 160Mb on mm1.

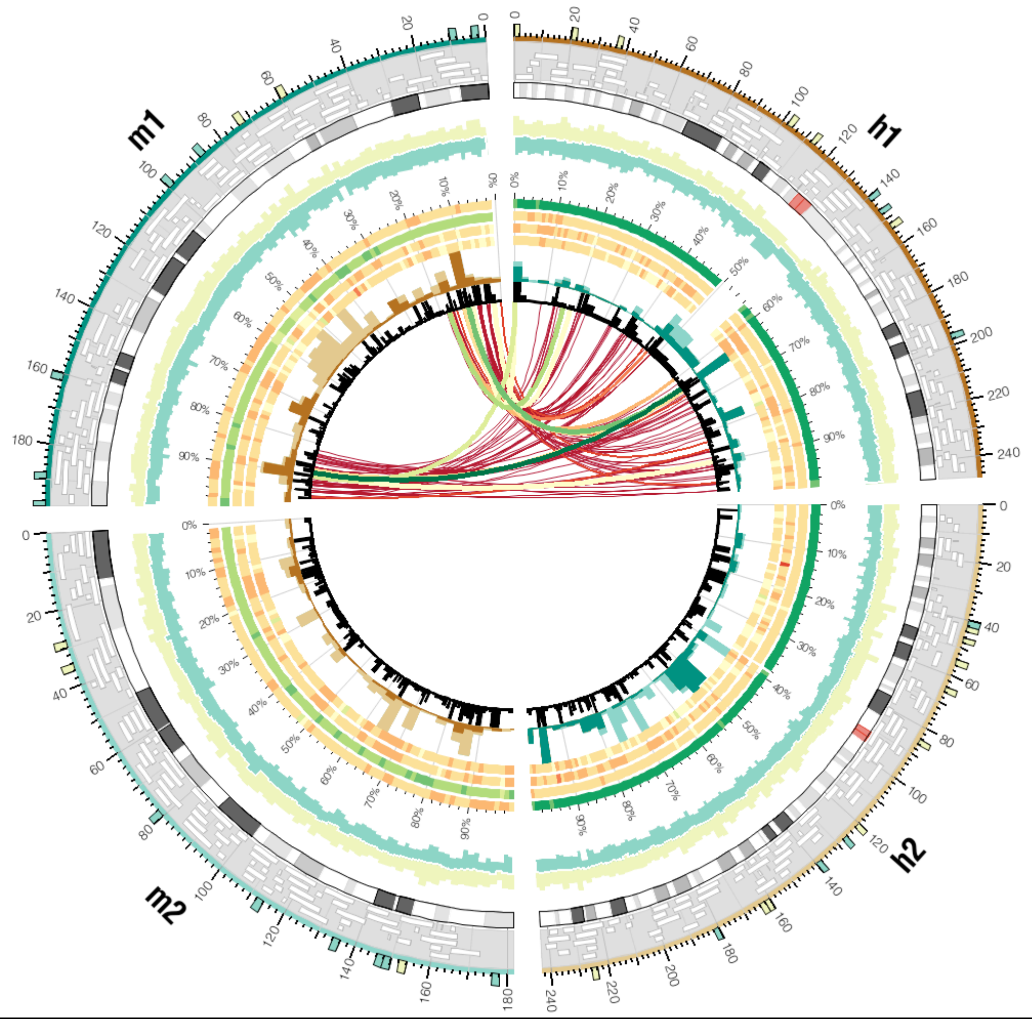


FIGURE 25

Using Perl code in the rule parameters, we can change the format of a link based on its properties.

Here, the `color`, `thickness` and `z` parameters are changed based on the size of the link.

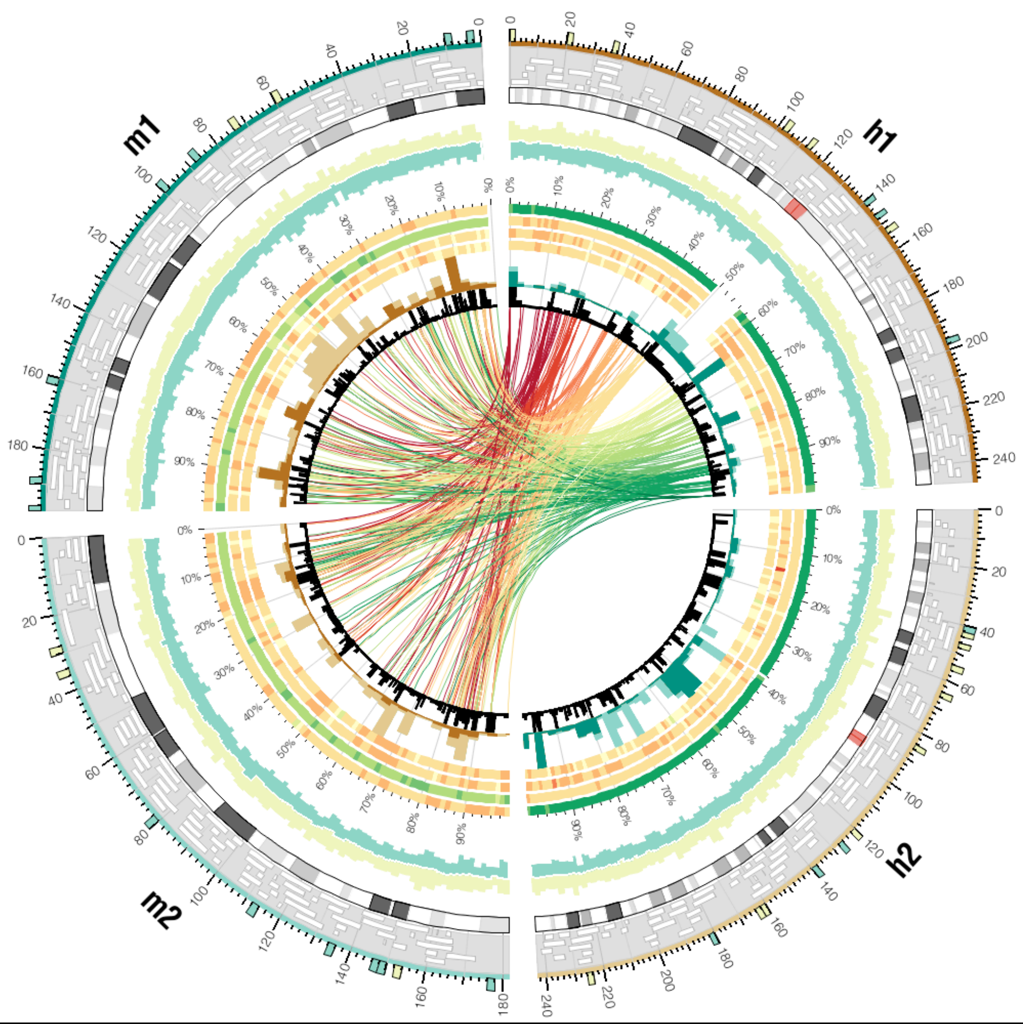


FIGURE 26

Using Perl code in the rule parameters, we can change the format of a link based on its properties.

Here, the `color` of the link is changed based on the position of its start.