Genome Visualization with Circos

Session 2 — Data Tracks

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
v0.22 29 June 2017
v0.21 3 May 2016
v0.20 12 May 2014
v0.19 6 May 2014
v0.18 7 May 2012
v0.17 6 Jul 2011
v0.16 23 Jul 2010
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

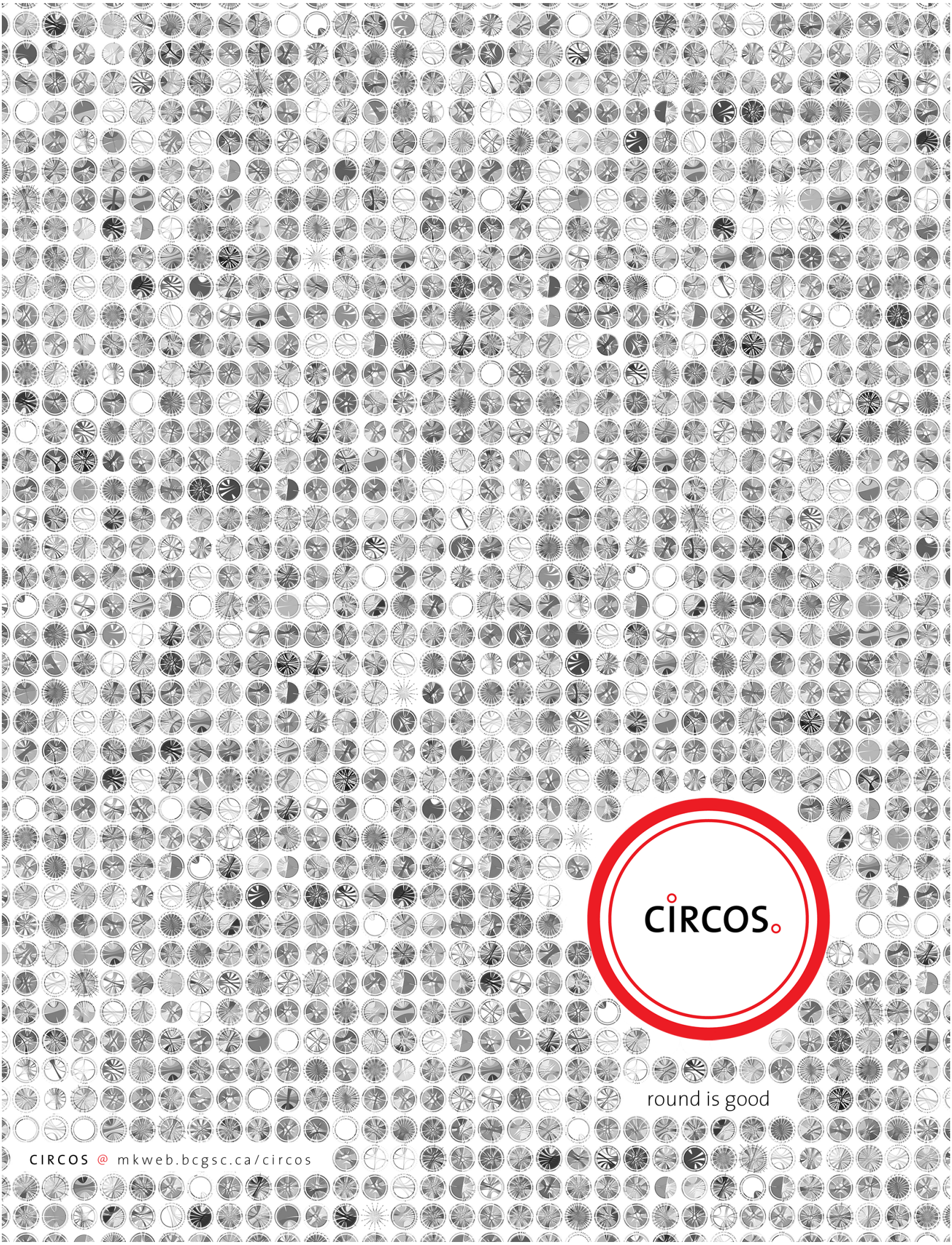

CIRCOS.
round is good
CIRCOS @ mkweb.bcgsc.ca/circos

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://www.circos.ca/documentation/tutorials

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.

```
# shared by all lessons in this session
<<include ../etc/karyotype.and.layout.conf>>
<<include ../etc/ideogram.conf>>
<<include ../etc/ticks.conf>>

# shared by all sessions
<<include ../../etc/image.conf>>

# from Circos distribution
<<include etc/colors_fonts_patterns.conf>>
<<include etc/housekeeping.conf>>
```

`../etc/karyotype.and.layout.conf` is a file that is shared by all the lessons in this session. It will be included in each lesson's `circos.conf`.

```
# ../etc/karyotype.and.layout.conf
karyotype = data/karyotype/karyotype.human.txt,
            data/karyotype/karyotype.mouse.txt

chromosomes_units           = 1000000
chromosomes_display_default = no

chromosomes       = hs1;hs2;mm1;mm2
chromosomes_order = hs1,hs2,mm2,mm1
chromosomes_color = hs1=rdylbu-11-div-2,hs2=rdylbu-11-div-3,
                    mm1=rdylbu-11-div-10,mm2=rdylbu-11-div-9

chromosomes_reverse = mm1,mm2
chromosomes_scale   = /./=1rn

<highlights>
<highlight>
file = ../data/highlight.txt
r0   = 1r+40p
r1   = 1r+45p
</highlight>
</highlights>
```

The figure uses two karyotype files (human and mouse) and shows chromosomes 1 and 2 from each genome. The the `chromosomes` parameter is used to limit which ideograms are drawn. Their order and color is defined by `chromosomes_order` and `chromosomes_color`, respectively.

The scale of the mouse ideograms is reversed using `chromosomes_reverse`, and the ideograms are each scaled using `chromosomes_scale` to `1rn` to make them occupy equally sized regions in 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 ideogram color key is taken from an 11-color Brewer palette (Figure 4). Human chromosomes are assigned red/orange and mouse chromosomes are assigned blue/light blue.

All the Brewer palettes are defined in `etc/brewer.conf` in the Circos distribution directory. This file is imported into the `<color>` block, which in turn is imported into the main configuration file by `etc/colors_fonts_patterns.conf`.

The relevant excerpt from `brewer.conf` is

```
# 11 color, red-yellow-blue
rdylbu-11-div-1 = 165,0,38
rdylbu-11-div-2 = 215,48,39
rdylbu-11-div-3 = 244,109,67
rdylbu-11-div-4 = 253,174,97
rdylbu-11-div-5 = 254,224,144
rdylbu-11-div-6 = 255,255,191
rdylbu-11-div-7 = 224,243,248
rdylbu-11-div-8 = 171,217,233
rdylbu-11-div-9 = 116,173,209
rdylbu-11-div-10 = 69,117,180
rdylbu-11-div-11 = 49,54,149
```

INTRODUCTION TO BREWER PALETTES

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

If you didn't get a handout of Brewer palettes (Figure 6), download it from

http://mkweb.bcgsc.ca/brewer/swatches/brewer-palettes-swatches.pdf

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 n=3-12 colors are available, across a range of hues and saturation (Figure 6). In addition, for diverging and sequential palettes the integrated 13- or 15-color palette from which the n-color variants are selected are shown in Figure 6 on the first line of each palette.

You can explore these palettes, see them on a map (they were originally designed for cartography) and filter them based on restrictions imposed by color printers, photocopiers (black and white output), laptop screens (can wash out weak colors) and color blind people (red/green).

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://mkweb.bcgsc.ca/brewer

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 data track. We will draw a histogram, which is a *plot*-type track. There are two kinds of data tracks: *plots* and *links*. We'll be covering links later.

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

```
<plots>
# global settings for all plots
<plot>
# plot track definition and overriding settings
</plot>
<plot>
# another plot track and overriding settings
</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, use `show=no` within the `<plot>` block.

```
<plots>
# global settings for each <plot> block
type        = histogram
thickness   = 1p
color       = black
#color      = white
min         = 0
max         = 1
r0          = 0.85r
r1          = 0.975r

<plot>
file        = ../data/both.cons.2e6.max.txt
fill_color  = spectral-5-div-3 # yellow
</plot>

<plot>
show        = no
file        = ../data/both.cons.2e6.avg.txt
fill_color  = spectral-5-div-4 # green
#thickness   = 2p
z           = 5
</plot>

<plot>
show        = no
file        = ../data/both.cons.2e6.min.txt
fill_color  = spectral-5-div-5 # blue
#fill_color  = white
z           = 10
</plot>
</plots>
```

We will draw three histograms, which share a large number of settings (e.g. location, min/max range, etc). These settings are defined globally inside the `<plots>` block, but outside the individual `<plot>` blocks. The global parameters define each plot to be a histogram using the type parameter and to have a data min/max range of [0,1]. The histograms are all placed between inner radius `r0 = 0.85r` and outer radius `r1 = 0.975r` (relative to the ideogram circle).

The histogram bins are outlined with `color = black` and the outline has a `thickness = 1p` (pixels). The first histogram, without any fill, is shown in Figure 7,

The `fill_color` parameter determines whether and how the histograms will be filled.

```
fill_color = spectral-5-div-3
```

The result is shown in Figure 8.

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 a different color. The result is shown in Figure 9.

```
<plot>
show        = yes
file        = ../data/both.cons.2e6.avg.txt
fill_color  = spectral-5-div-4
#color       = white
#thickness   = 2p
z           = 5
</plot>
```

Finally, we'll add a third histogram also in the same region of the figure (Figure 10).

```
<plot>
show        = yes
file        = ../data/both.cons.2e6.min.txt
fill_color  = spectral-5-div-5
#fill_color  = white
z           = 10
</plot>
```

Let's now reformat the three histograms in Figure 10 to create an interesting visual effect. The values below are the ones which will change.

```
<plots>

# each histogram will have a white outline
color = white
```

```
<plot>
# first plot will be unchanged
...
</plot>

<plot>
show        = yes
...
# thicker outline to separate the two histograms
thickness   = 2p
</plot>

<plot>
show        = yes
# white fill to remove the histogram from view, but use it
# to mask out the bottom of the second histogram
fill_color  = white
</plot>

</plots>
```

The result is shown in Figure 11. 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.

HISTOGRAM DATA FORMAT

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.

DATA OPTIONS

All input data files can have an optional field that associates parameters with a data point. These are usually formatting parameter, but they don't have to be.

```
# chr start end value
mm1 10000000 11999999 0.439079022807017 color=blue
mm1 12000000 13999999 0.44630188700565 color=blue,thickness=3p
mm1 14000000 15999999 0.452856295597484 color=green,id=abc
```

Format parameters like `color` and `thickness` will override any track settings. Parameters like `id` can be used in rules, which we'll see later, to change how data is displayed.

TRACK DEFAULTS

All track types have sufficient parameters set to default values to be drawn by only specifying the file name and track type. This gets you started quickly.

For example, this block creates a histogram

```
<plot>
type = histogram
file = ../data/both.cons.2e6.avg.txt
</plot>
```

whose remaining necessary parameters are set by the `etc/track/histogram.conf` file in the Circos distribution directory.

```
# etc/track/histogram.conf
color            = black
thickness        = 1
r1               = 0.89r
r0               = 0.8r
orientation      = out
```

These defaults are applied to all the tracks of a specific type. Thus, if you draw multiple histograms using their default values, they will be drawn on top of each other because their `r0` and `r1` values will be the same.

You can undefined a parameter, such as one set by default, using `undef`

```
thickness = undef
```

You have the ability to turn off the use of default settings in `etc/housekeeping.conf`. Either comment out

```
track_defaults = etc/tracks
```

or override it in your main configuration file with

```
track_default* = undef
```

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 chromosomes 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 `2/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 `2/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 `2/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 either a comma-separated list of the colors that are mapped onto the `min-max` range, or the name of a color list. Values beyond the range are assigned to the nearest color.

```
<plot>
type  = heatmap
file  = ../data/both.cons.2e6.rhe.avg.txt
min   = 0.1
max   = 0.9
r0    = 0.73r
r1    = 0.75r
color = spectral-11-div
#scale_log_base = 0.5
</plot>
```

Figure 12 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 = spectral-11-div
#scale_log_base = 0.5
</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 = spectral-11-div
#scale_log_base = 0.5
</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 = spectral-11-div
#scale_log_base = 0.5
</plot>
```

Figure 13 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.

COLOR LISTS

The colors of the heatmap can be defined by using an explicit list of colors (tedious),

```
colors = spectral-11-div-1,spectral-11-div-2,spectral-11-div-3,spectral-11-div-4,
         spectral-11-div-5,spectral-11-div-6,spectral-11-div-7,spectral-11-div-8,
         spectral-11-div-9,spectral-11-div-10,spectral-11-div-11
```

or by using a *color list* (convenient)

```
colors = spectral-11-div
```

Currently, color lists are defined for every Brewer palette (Figure 6). The format of the color name is `PALETTE-COLORS-TYPE-INDEX`. The name of the list is the same as the color, but without the. For example, `rdylbu-5-div` is the 5-color diverging red-yellow-blue palette list, `spectral-7-div` is the 7-color spectral palette list, and so on. Lists are defined in `etc/brewer.lists.conf` in the Circos distribution. For example, the spectral palette lists are

```
spectral-3-div      = spectral-3-div-(\d+)
spectral-3-div-rev  = rev(spectral-3-div-(\d+))
spectral-4-div      = spectral-4-div-(\d+)
spectral-4-div-rev  = rev(spectral-4-div-(\d+))
spectral-5-div      = spectral-5-div-(\d+)
spectral-5-div-rev  = rev(spectral-5-div-(\d+))
spectral-6-div      = spectral-6-div-(\d+)
spectral-6-div-rev  = rev(spectral-6-div-(\d+))
spectral-7-div      = spectral-7-div-(\d+)
spectral-7-div-rev  = rev(spectral-7-div-(\d+))
spectral-8-div      = spectral-8-div-(\d+)
spectral-8-div-rev  = rev(spectral-8-div-(\d+))
spectral-9-div      = spectral-9-div-(\d+)
spectral-9-div-rev  = rev(spectral-9-div-(\d+))
spectral-10-div     = spectral-10-div-(\d+)
spectral-10-div-rev = rev(spectral-10-div-(\d+))
spectral-11-div     = spectral-11-div-(\d+)
spectral-11-div-rev = rev(spectral-11-div-(\d+))
```

The expressions that define the list are a regular expression which is used to match the colors that belong to the list. Colors are ordered by increasing value of the numbers captured by the regular expression. Optionally, if the list definition is composited with `rev()`, the list will have the colors reversed.

Reversing the colors in a heatmap is very easy, if a list was used. Just append `-rev` to the color name.

```
color = spectral-11-div-rev
```

You can combine color lists

```
color = reds-9-seq-rev,blues-9-seq
```

to create longer lists. In this case, an 18-color red-blue color palette was generated. You could use the already defined 11-color `rdbu-11-div` list, but by combining the red and blue lists you get a palette with many colors. Don't over do it on the different number of colors.

You can interchangeably mix colors and lists together. Below, the spectral color list will be flanked by black. Smallest and largest values will be black.

```
color = black,spectral-11-div,black
```

COLOR ENCODING

There are several ways in which you can map colors onto values. The default `color_mapping=0` will likely suit you just fine.

However, if you want to emphasize values at the extreme ends of heatmap values, you can switch to `color_mapping=2`, in which the first and last color in the list will be used to color values `<min` and `>max`. For more details, see

http://www.circos.ca/documentation/tutorials/2d_tracks/heat_maps

LINEAR AND LOGARITHMIC MAPPING

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 (Figure 15).

To apply logarithmic mapping, use the `scale_log_base` parameter. For `scale_log_base < 1` the color index grows quickly at the beginning (emphasizing differences in small values) and slowly at the end (attenuating differences in large values). The opposite happens for `scale_log_base > 1`.

HEATMAP DATA FORMAT

The data format for heatmaps is the same as for histograms.

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

Thus, you can change the type between histogram and heatmap without needing to adjust the data file.

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 `2/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 `<link>` blocks, within an outer `<links>` block.

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

Links are drawn as quadratic Bezier curves, described at

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 16 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://www.circos.ca/documentation/tutorials/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 17 shows the links as defined in the above block. By adding transparency to the color of the link lines (Figure 18) dense links can be more easily interpretable (compare Figure 18 with Figure 17).

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

COLOR TRANSPARENCY

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      slightly transparent ---------->------------- very transparent
```

```
red_a0 0%  red_a1 16%  red_a2 33%  red_a3 50%  red_a4 66%  red_a5 83%
red    0%
```

Notice that `red` and `red_a0` are both the same color—a fully opaque red.

LINK DATA FORMAT

Each link is specified as a pair of coordinates on a single line.

```
# chr1 start1 end1 chr2 start2 end2
hs1 120788758 120834144 mm1 63903740 63957877
hs2 190247704 190320005 mm1 124448312 124506291
...
```

The last field is optional and stores format properties and any other variables

```
hs1 120788758 120834144 mm1 63903740 63957877 id=abc,color=blue
```

WHAT IS SHOWN IN THE FIGURE?

The data for the links were obtained from the human assembly in the UCSC Genome Browser table viewer (*group* Comparative Genomics *track* Mouse Chain/Net *table* Mouse chain). These files are available in `sessions/2/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 – Histograms

In this lesson we will add two histograms immediately outside of the links to indicate the number of link ends within a region. These 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 (`tools/binlinks` in the Circos distribution).

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

```
<plot>
type       = histogram
file       = ../data/links.density.txt
min        = 0
max        = 10
r0         = 0.5r
r1         = 0.55r
thickness  = 0
fill_color = black
</plot>

<plot>
type       = histogram
file       = ../data/links.density.stacked.txt
min        = 0
max        = 300000
r0         = 0.55r
r1         = 0.65r
thickness  = 0
fill_color = rdylbu-11-div-10,rdylbu-11-div-2,rdylbu-11-div-3,rdylbu-11-div-9
</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 19.

Histograms are excellent for providing a summary of the link data set by showing the number of links incident on a region (link density). 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 20 shows a version of Figure 19, with only the links and density histograms shown. In region (A) the majority of links depart to terminate on `hs2`. Region (B), on the other hand, is more similar to `hs1` as seen by the fact that most links that depart from this region terminate on hs1. Similarly regions (C) and (D) on hs2 are more similar to `mm1` and `mm2`, respectively.

AXIS GRIDS

Tracks that show data over a range can have axis grids at flexible spacing intervals.

```
<plot>
...
<axes>
# relative spacing
<axis>
color     = grey_a4
spacing   = 0.25r
thickness = 1
</axis>
<axis>
color     = grey_a1
spacing   = 0.5r
thickness = 1
</axis>
# specific position
<axis>
color     = red
position  = 0.75r
thickness = 2
</axis>
</axes>
</plot>
```

These axes grids are shown in Figure 21 and Figure 22.

STACKED HISTOGRAM DATA FORMAT

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
```

WHAT IS SHOWN IN THE FIGURE?

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 23 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+2p
r1      = 1r+40p

layers                = 7
layers_overflow       = hide
layers_overflow_color = red

margin    = 1u
thickness = 3
padding   = 2

orientation = out

color            = black
stroke_thickness = 1
stroke_color     = vdgrey
</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`. Figure 24 explains how tiles are placed.

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.

For more information about tiles, see the tutorial at

http://www.circos.ca/documentation/tutorials/2d_tracks/tiles

TRACK BACKGROUND

Each track can have its background colored flexibly. You can define the start and end of any number of colored regions.

```
<plot>
<backgrounds>
<background>
y0    = 0.75r
color = grey_a1
</background>
<background>
y0    = 0.25r
y1    = 0.75r
color = grey_a3
</background>
<background>
y1    = 0.25r
color = grey_a5
</background>
</backgrounds>
</plots>
```

I've set the tile track background to have transparency so that the grid, drawn underneath, can show through.

TILES DATA FORMAT

The format of tiles is

```
# chr start end
mm1 10000000 11999999
mm1 12000000 13999999
mm1 14000000 15999999
```

Notice that a tile does not have an associated value. Format for tiles is shared by highlights, allowing you to interchange these track types.

WHAT IS SHOWN IN THE FIGURE?

The tiles are randomly generated using `2/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. We used the `<highlight>` block which produces highlights underneath data and other image elements, such as grids.

Now, we'll come back and add two more highlights to draw attention to regions that had very low and very high conservation levels. In this case, we'll use a plot block (*type* highlight) to demonstrate another way in which highlights can be drawn. If you wish to draw a highlight on top of elements, such as data, you must use a highlight *plot*.

Figure 25 illustrates the concept of a highlight. One highlight track will show the top 20 conserved regions (`scatter.max.top20.txt`) and another the bottom 20 (`scatter.min.top20.txt`). I've added outlines to these highlight elements because the yellow color does not contrast well with white for small elements.

```
<plot>
type = highlight
file = ../data/highlight.max.top20.txt
r0   = 0.975r
r1   = 0.995r
#r0  = dims(ideogram,radius_inner)+5p
#r1  = dims(ideogram,radius_outer)-5p
fill_color       = spectral-5-div-3
stroke_thickness = 1p
stroke_color     = red
z = 15
</plot>

<plot>
type = highlight
file = ../data/highlight.min.top20.txt
r0   = 0.835r
r1   = 0.855r
#r0  = dims(ideogram,radius_inner)+5p
#r1  = dims(ideogram,radius_outer)-5p
fill_color       = spectral-5-div-4
stroke_thickness = 1p
stroke_color     = red
z = 15
</plot>
```

Figure 26 shows the addition of the two highlight tracks. The highlights are colored using the same color as the conservation histograms (`spectral-5-div-3` for the upper one, and `spectral-5-div-4` for the lower one).

Highlights are a very flexible data track which can be used to include both data and annotations on your figure. Individual highlights can have their `r0` and `r1` values adjusted, giving opportunity for creative uses of this track type. For example, by removing the fill from the highlights and stretching them across the height of the histogram, they can be used to outline regions of the figure, as shown in Figure 27.

```
<plot>
```

```
...
r0         = 0.9r
r1         = 0.975r
#fill_color = red
</plot>

<plot>
...
r0   = 0.835r
r1   = 0.9r
#fill_color = red
</plot>
```

Highlights can be placed anywhere—including on top of ideograms. By using the `dims()` function, you can reference the inner and outer radius of the ideograms to draw the highlights on top of ideograms, as shown in Figure 28.

```
<plot>
...
r0  = dims(ideogram,radius_inner)+5p
r1  = dims(ideogram,radius_outer)-5p
fill_color = red
</plot>

<plot>
...
r0  = dims(ideogram,radius_inner)+5p
r1  = dims(ideogram,radius_outer)-5p
fill_color = red
</plot>
```

HIGHLIGHTS DATA FORMAT

The format of highlights is

```
# chr start end
mm1 10000000 11999999
mm1 12000000 13999999
mm1 14000000 15999999
```

Notice that a highlight does not have an associated value (like, for example, histograms). Format for highlights is shared the same as for tiles, allowing you to interchange these track types.

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
```

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(var(size1)/10e3) ) )
```

where `var(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>
...

<rules>
<rule>
# each link is tested with a rule
# if the condition passes...
condition   = on(hs1)
# the color will be changed
color       = rdylbu-11-div-2_a3
# as well as the z value
z           = 10
</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 `rdylbu-11-div-2`, which is the color used for the chromosomes color index. The result is shown in Figure 29.

RULE CHAIN

Multiple rules can be used. All links are tested with rules in order of appearance (or an optional `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>
```

```
condition   = on(hs1)
color       = rdylbu-11-div-2_a3
z           = 10
</rule>

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

<rule>
# 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 30.

To further refine which links are shown, we can add another condition that further selects links based on position. When multiple conditions are defined, they are combined with `AND`.

```
condition = between(hs1,mm1)
condition = var(start2) < 40mb || var(start2) > 160mb
```

Only links that are between `hs1` and `mm1` and which start before 40Mb or after 160Mb on `mm1` are shown (Figure 31).

Suffixes like "`kb`", "`mb`" and "`gb`" are understood and correspond to multipliers of 10^3, 10^6 and 10^9.

DYNAMIC VALUES

You can write rules that use the properties of the data point to set parameter values. These are dynamic rules—their behavior changes depending on the data.

Certain keywords are parsed during rule evaluation and the corresponding property of the link is substituted. These are `startN`, `endN`, `chrN (N=1,2)` as well as parameters such as color, thickness and others. These values are accessed using `var()` (e.g. `var(size1)`).

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. The following will do the job, and although it seems complicated at first I will walk you through the expression.

```
thickness = eval( sprintf("%dp",remap_round(var(size1),0,50000,1,5) )
```

The `sprintf()` function takes a format string and a list of values which are interpolated into the format string. Special tokens (e.g. `%d`) in the format string correspond to different ways in which numbers (and strings in general) can be formatted. For example `%d` means integer, created by truncating any fractional component, so that `sprintf("%d",2.2)` will return `2`. Other useful tokens are `%s` (string) and `%f` (float).

In the above rule, the format string is `%dp` which produces an integer followed by a "`p`", to express the value in units of pixels. Note that the "`p`" is not part of the special format token. The value to be formatted is provided by the `remap_round` function, available to you in all rules. This function takes a `value` (first argument), and linearly remaps it from (`min`,`max`) (second and third arguments) onto the range (`remap_min`,`remap_max`) (fourth and fifth arguments)

```
remap_round(value,min,max,remap_min,remap_max)
```

Thus, the size of the start of the link `size1` will be remapped from 0-50kb to 1-5. Any values of `size1` > 50kb will be remapped to 5.

The rules for color and `z` work similarly. We will remap size onto `z` and position onto colors `rdylbu-11-div-{1...11}_a3`.

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

```
<rule>
condition   = fromto(hs1,mm1)
condition   = var(start2) < 40mb || var(start2) > 160mb
thickness   = eval(sprintf("%dp",remap_round(var(size1),0,50000,1,5)))
z           = eval(sprintf("%dp",remap_round(var(size1),0,50000,1,5)))
color       = eval(sprintf("rdylbu-11-div-%d_a3",
                           remap_round(var(size1),0,250e6,1,11)))

</rule>
```

and the resulting image is shown in Figure 32. You can compare how hs1/mm1 and hs2/mm2 are related by applying the same rule to links between hs1 and mm2.

```
<rule>
condition   = fromto(hs1,mm1)
condition   = var(start2) < 40mb || var(start2) > 160mb
thickness   = eval(sprintf("%dp",remap_round(var(size1),0,50000,1,5)))
z           = eval(sprintf("%dp",remap_round(var(size1),0,50000,1,5)))
color       = eval(sprintf("rdylbu-11-div-%d_a3",
                           remap_round(var(size1),0,250e6,1,11)))

</rule>

<rule>
# you can toggle whether a rule is considered using 'use'
condition   = fromto(hs2,mm2)
```

```
condition   = var(start2) < 40mb || var(start2) > 160mb
thickness   = eval(sprintf("%dp",remap_round(var(size1),0,50000,1,5)))
z           = eval(sprintf("%dp",remap_round(var(size1),0,50000,1,5)))
color       = eval(sprintf("rdylbu-11-div-%d_a3",
                           remap_round(var(size1),0,250e6,1,11)))

</rule>

</rule>
```

There is quite a lot of duplication in these two rules, which vary only by the "1" and "2" chromosome number in the condition. The condition can be generalized to

```
<rule>
# you can toggle whether a rule is considered using 'use'
condition   = fromto(hs1,mm1) || fromto(hs2,mm2)
condition   = var(start2) < 40mb || var(start2) > 160mb
thickness   = eval(sprintf("%dp",remap_round(var(size1),0,50000,1,5)))
z           = eval(sprintf("%dp",remap_round(var(size1),0,50000,1,5)))
color       = eval(sprintf("rdylbu-11-div-%d_a3",
                           remap_round(var(size1),0,250e6,1,11)))
</rule>
```

Learning how to use rules can save you a lot of time, because you can use them to change the look of the figure without changing your input data file.

There are several detailed tutorials about rules at *http://www.circos.ca/documentation/tutorials/links/*.

Figures

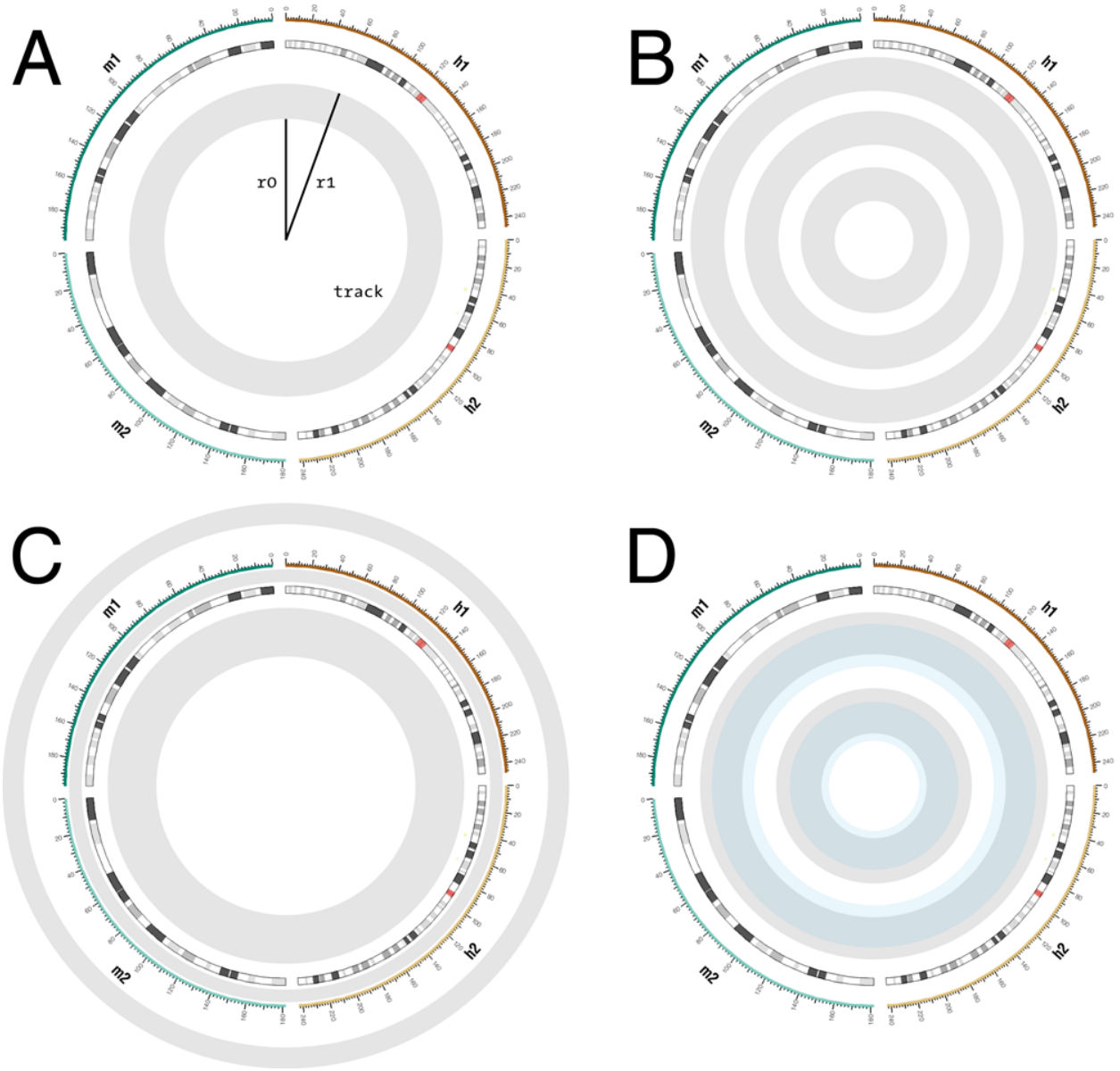


FIGURE 1

(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.

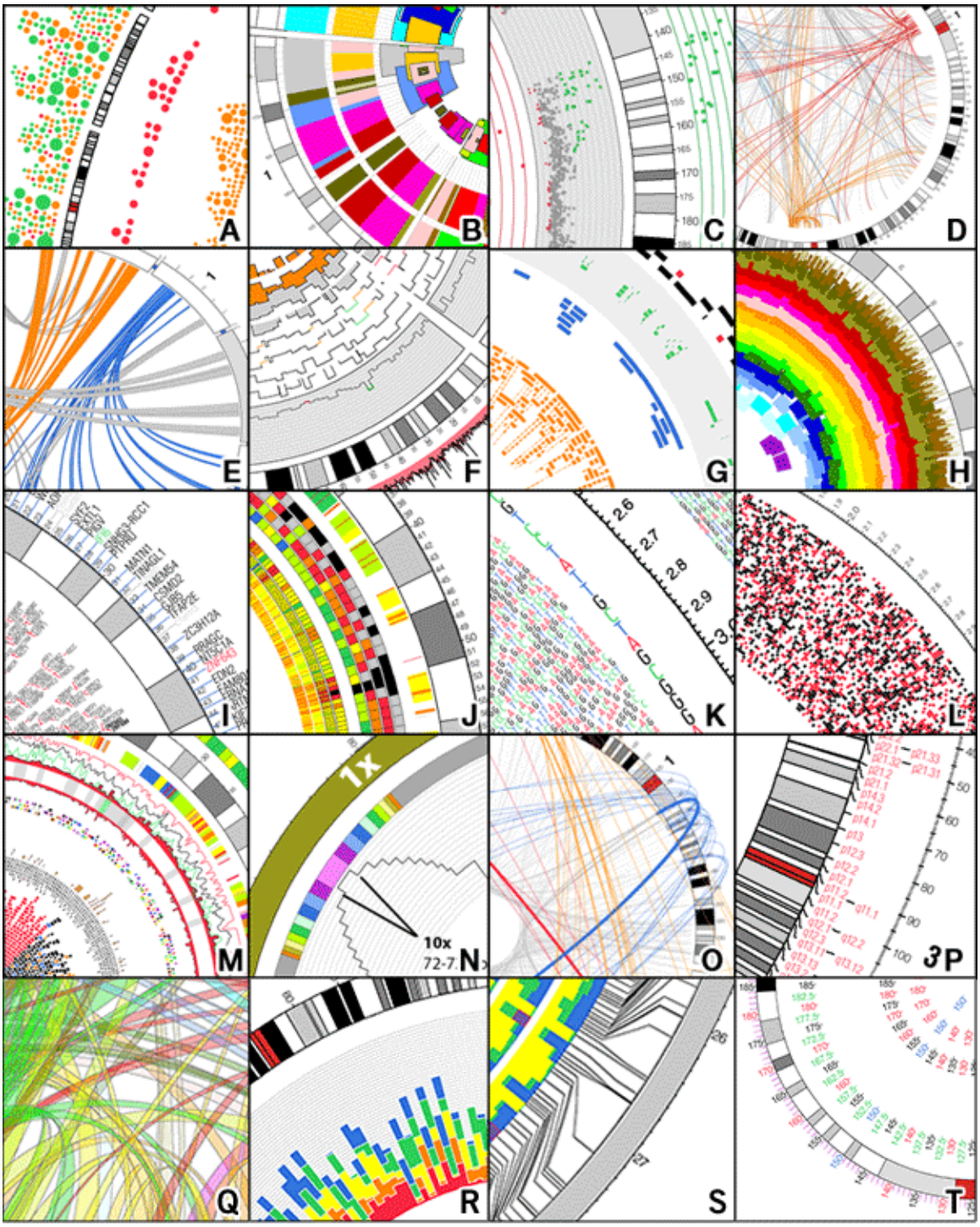


FIGURE 2

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.

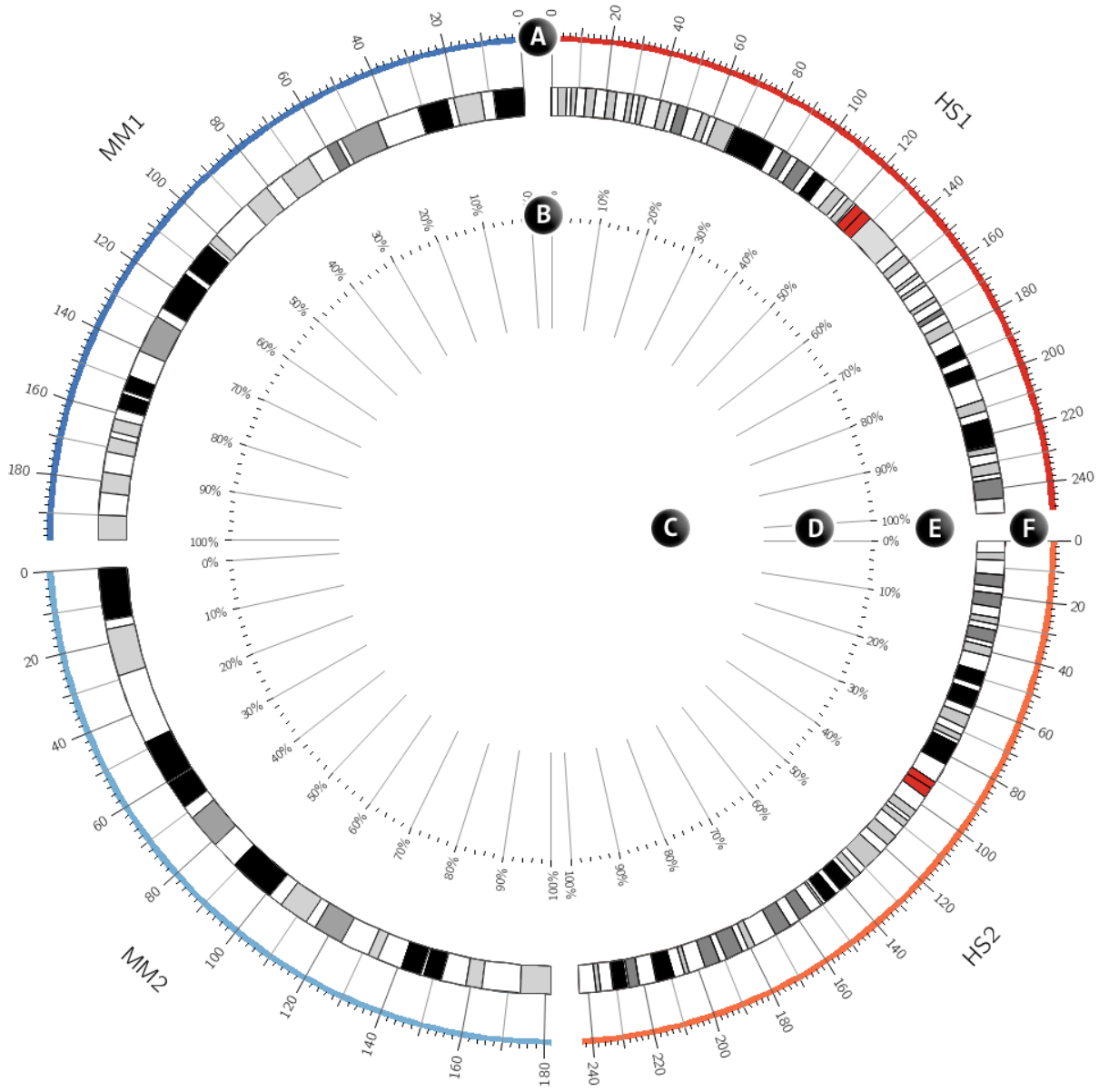


FIGURE 3

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 hs1 and hs2) and mouse chromosomes 1 and 2 (labeled mm1 and mm2). 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.

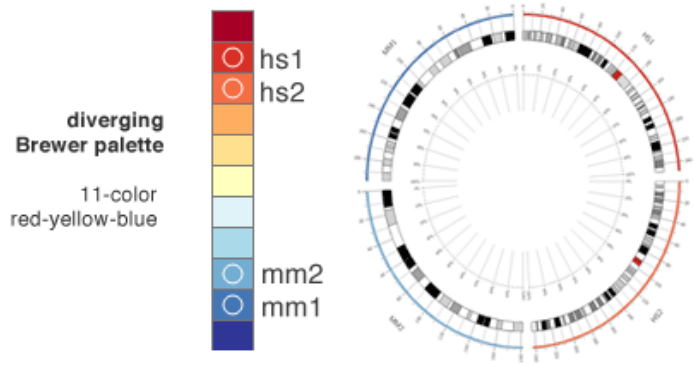


FIGURE 4

The ideograms are color coded using the 1-color diverging red-yellow-blue Brewer palette, shown here.

```
chromosome    color name
hs1           rdylbu-11-div-2
hs2           rdylbu-11-div-3
mm1           rdylbu-11-div-10
mm2           rdylbu-11-div-9
```

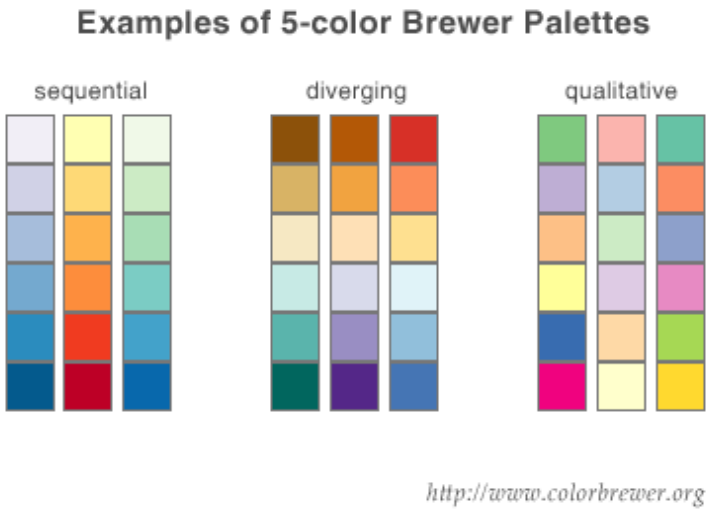


FIGURE 5

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.

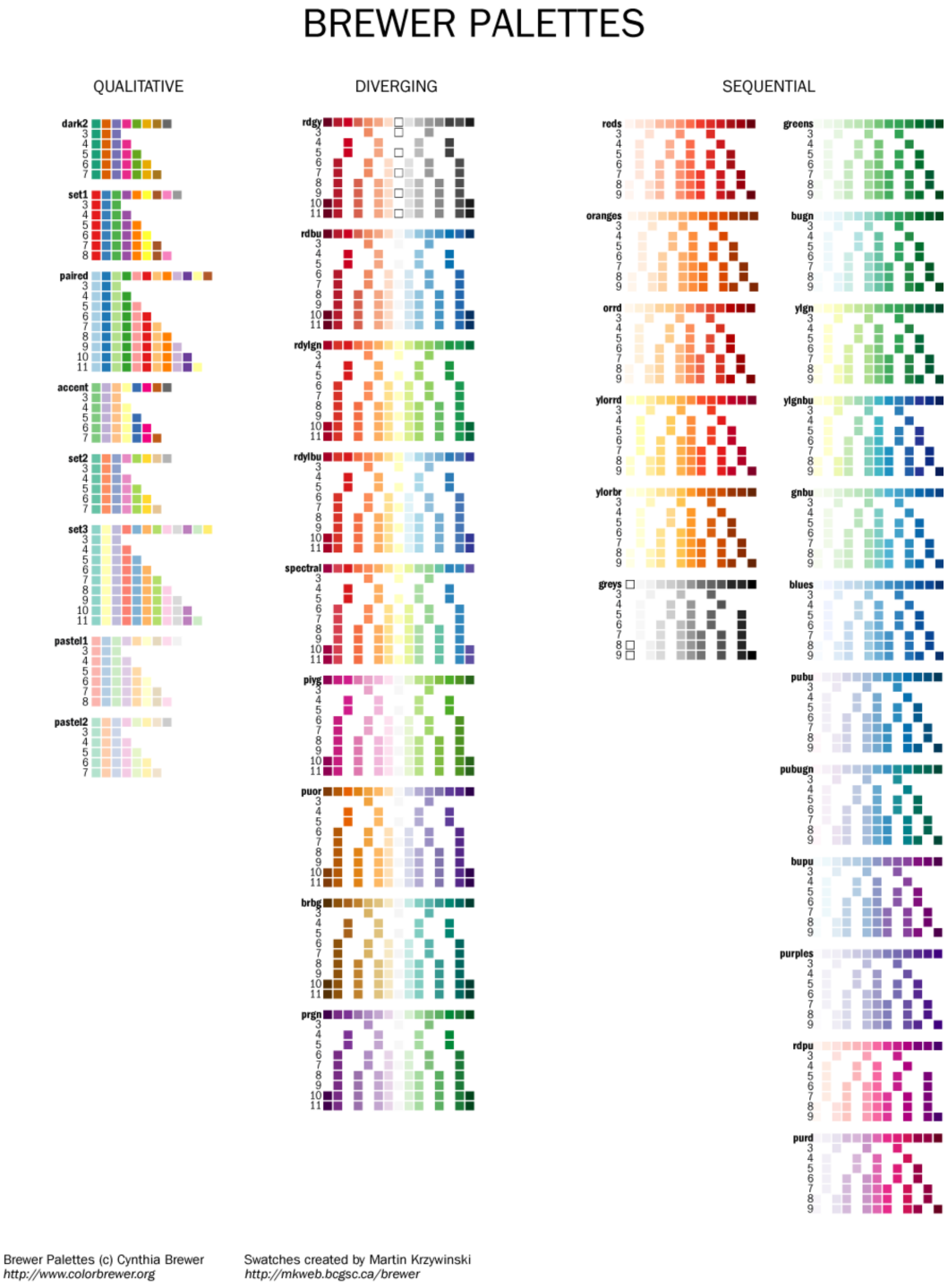


FIGURE 6

All Brewer palettes. This image is available at

http://mkweb.bcgsc.ca/brewer/swatches/brewer-palettes-swatches.pdf

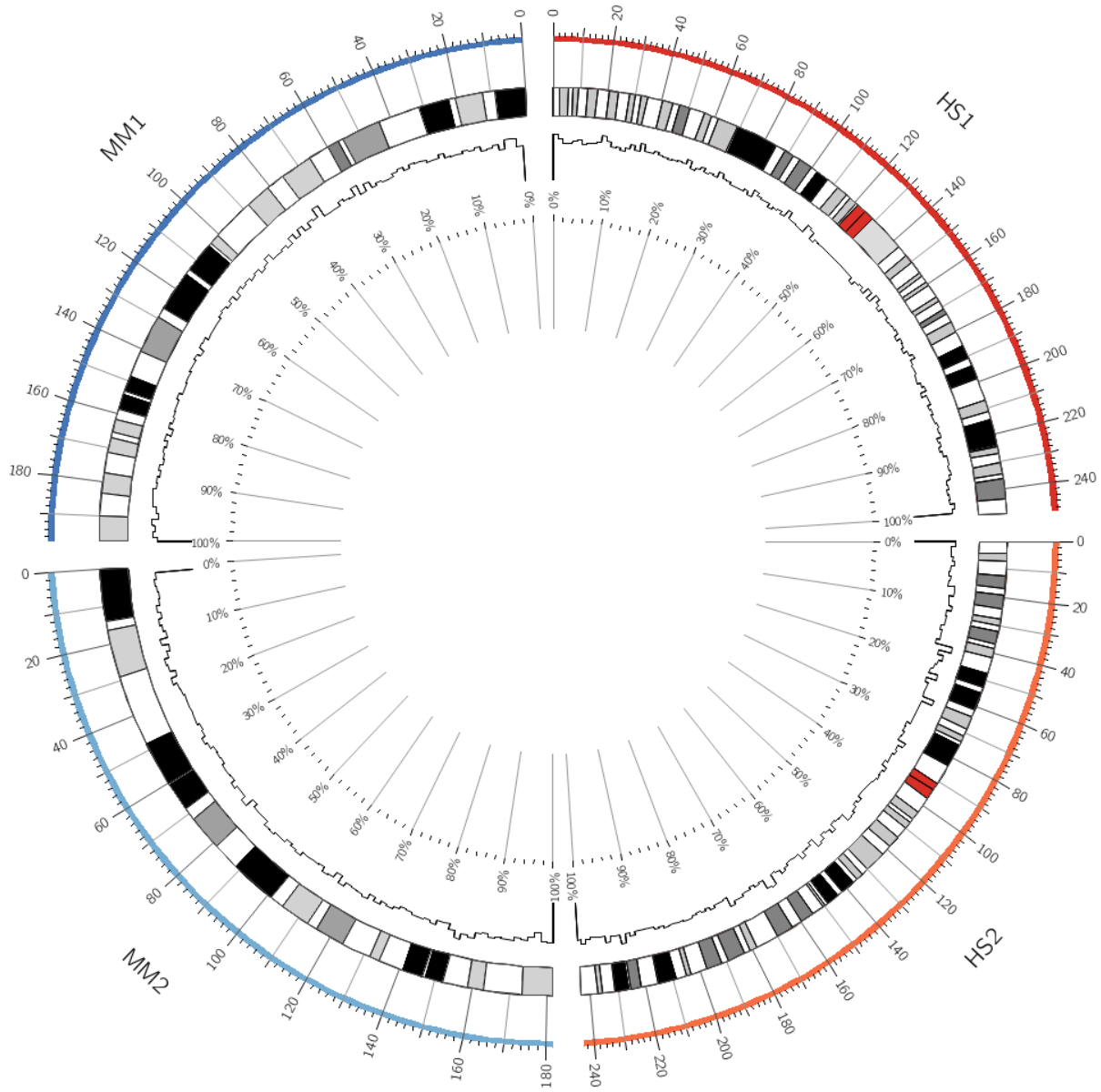


FIGURE 7

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

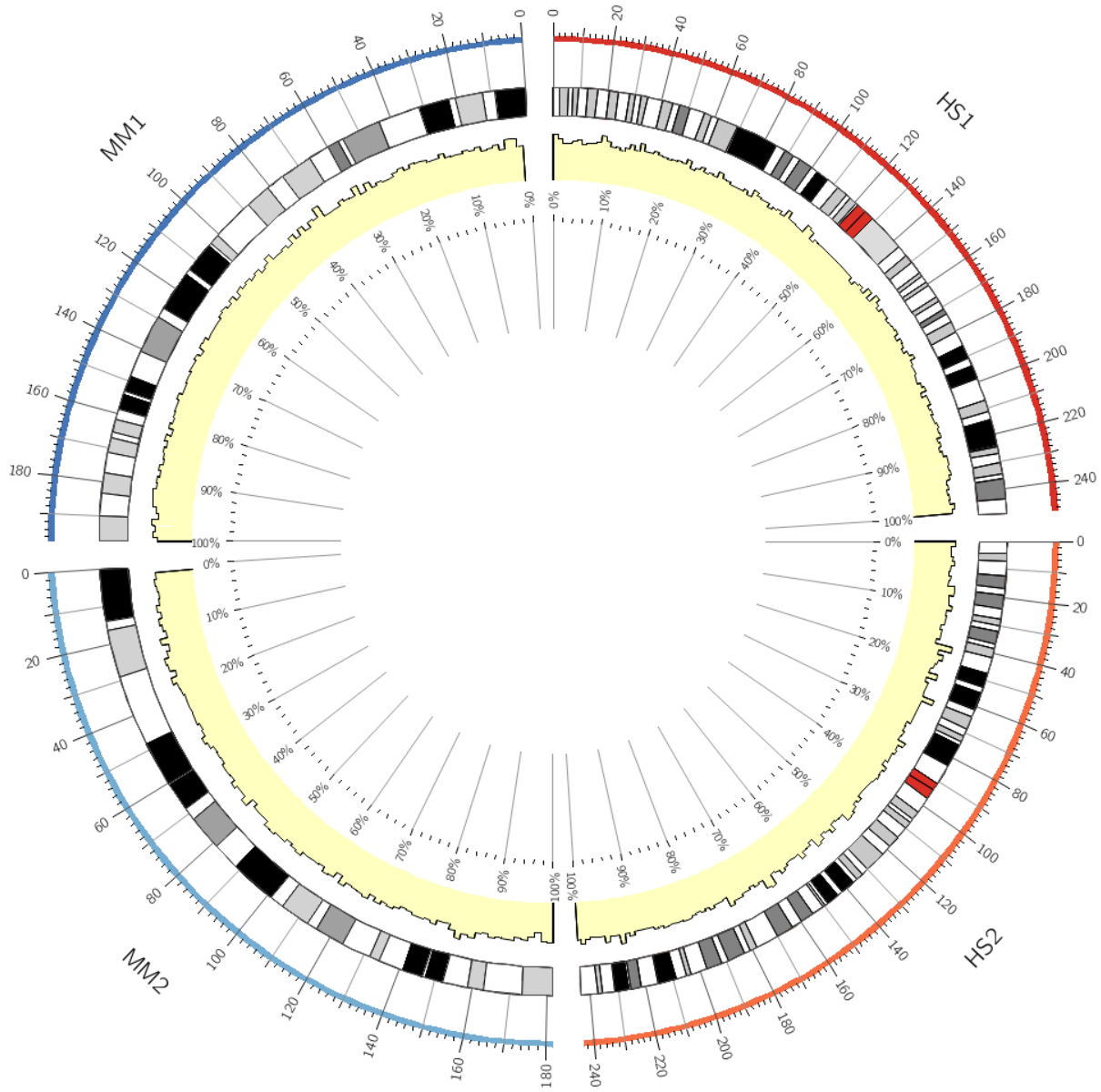


FIGURE 8

The histogram has been adjusted to include a fill color, using the `fill_color` parameter.

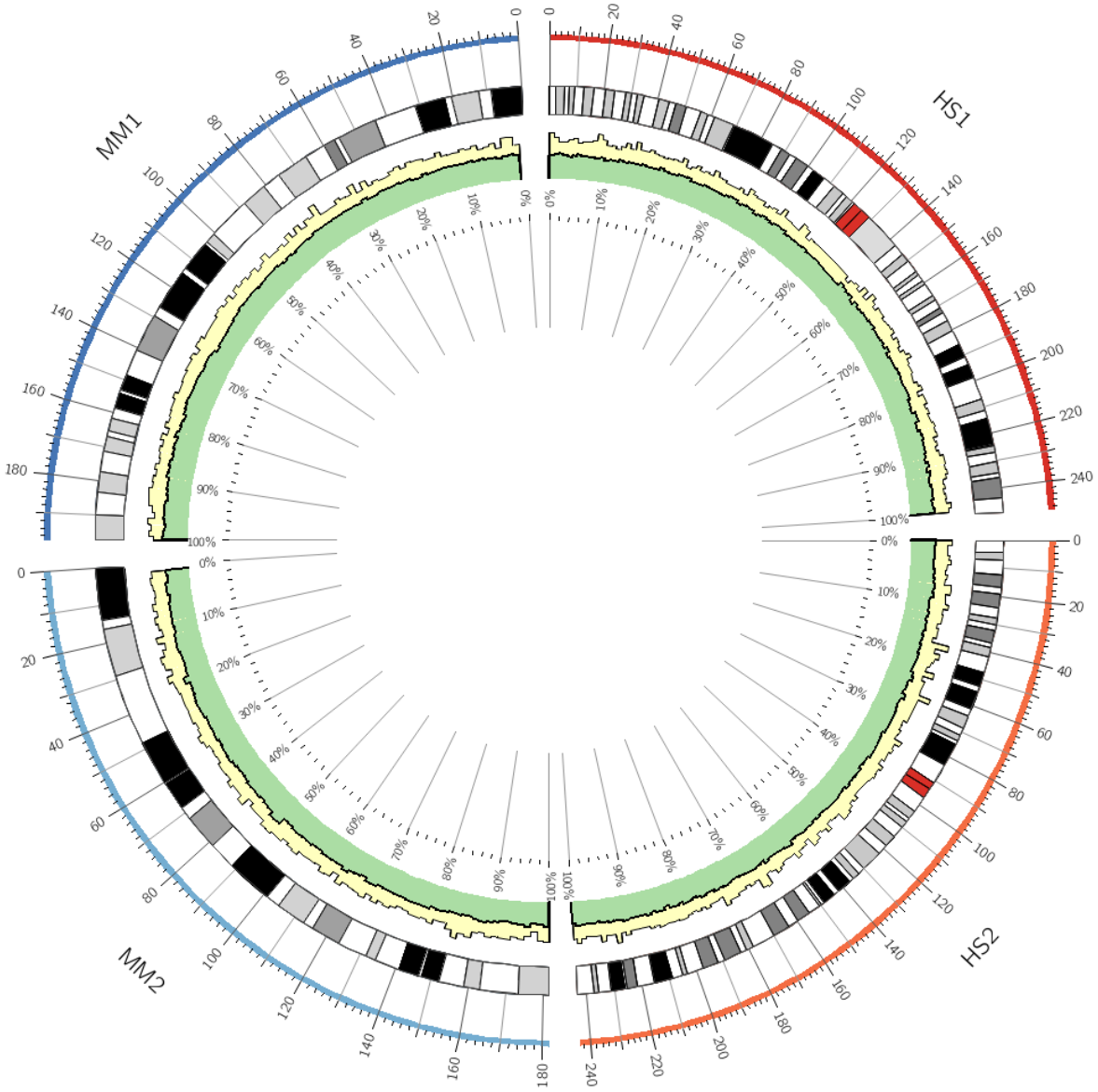


FIGURE 9

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

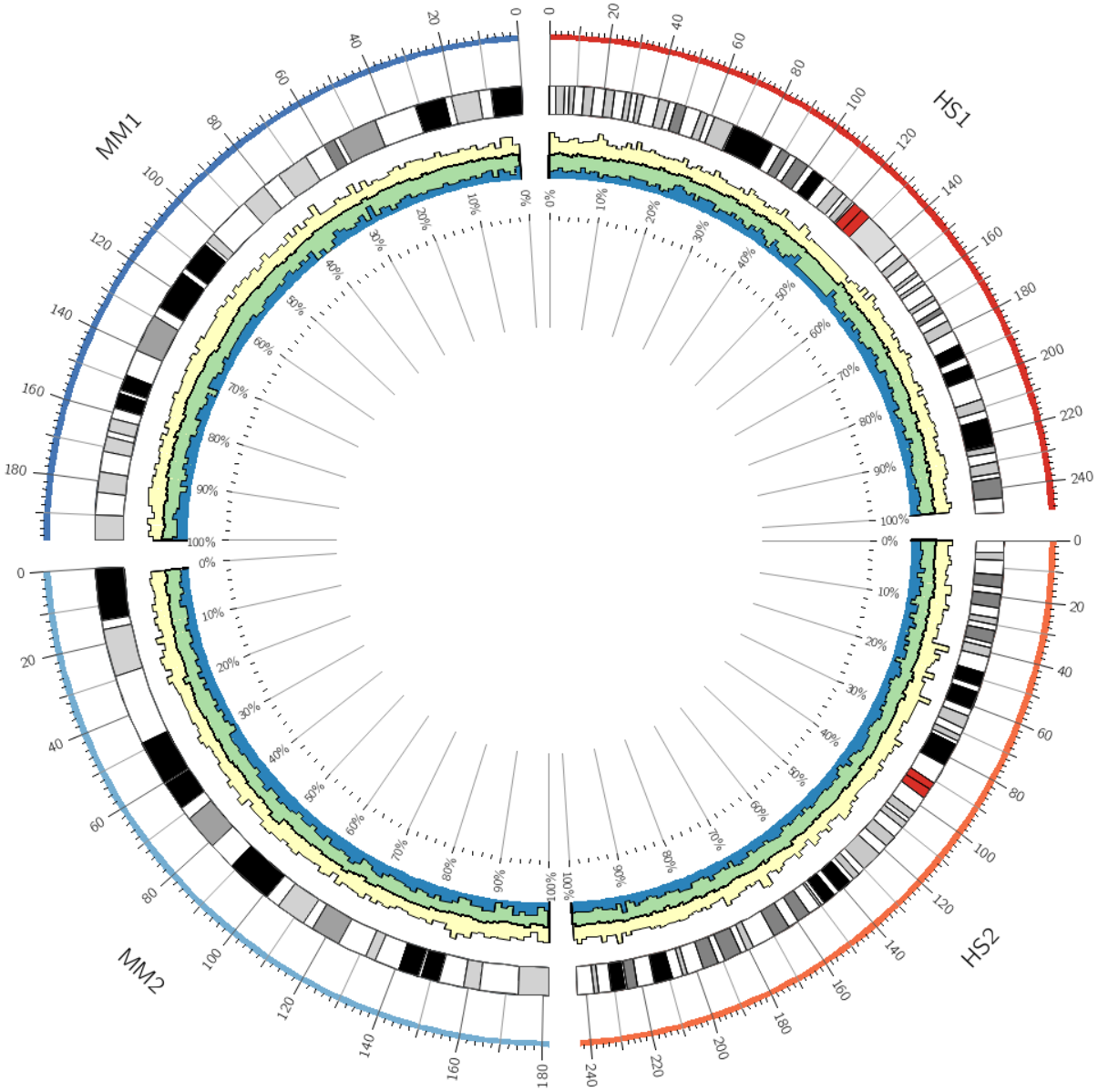


FIGURE 10

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

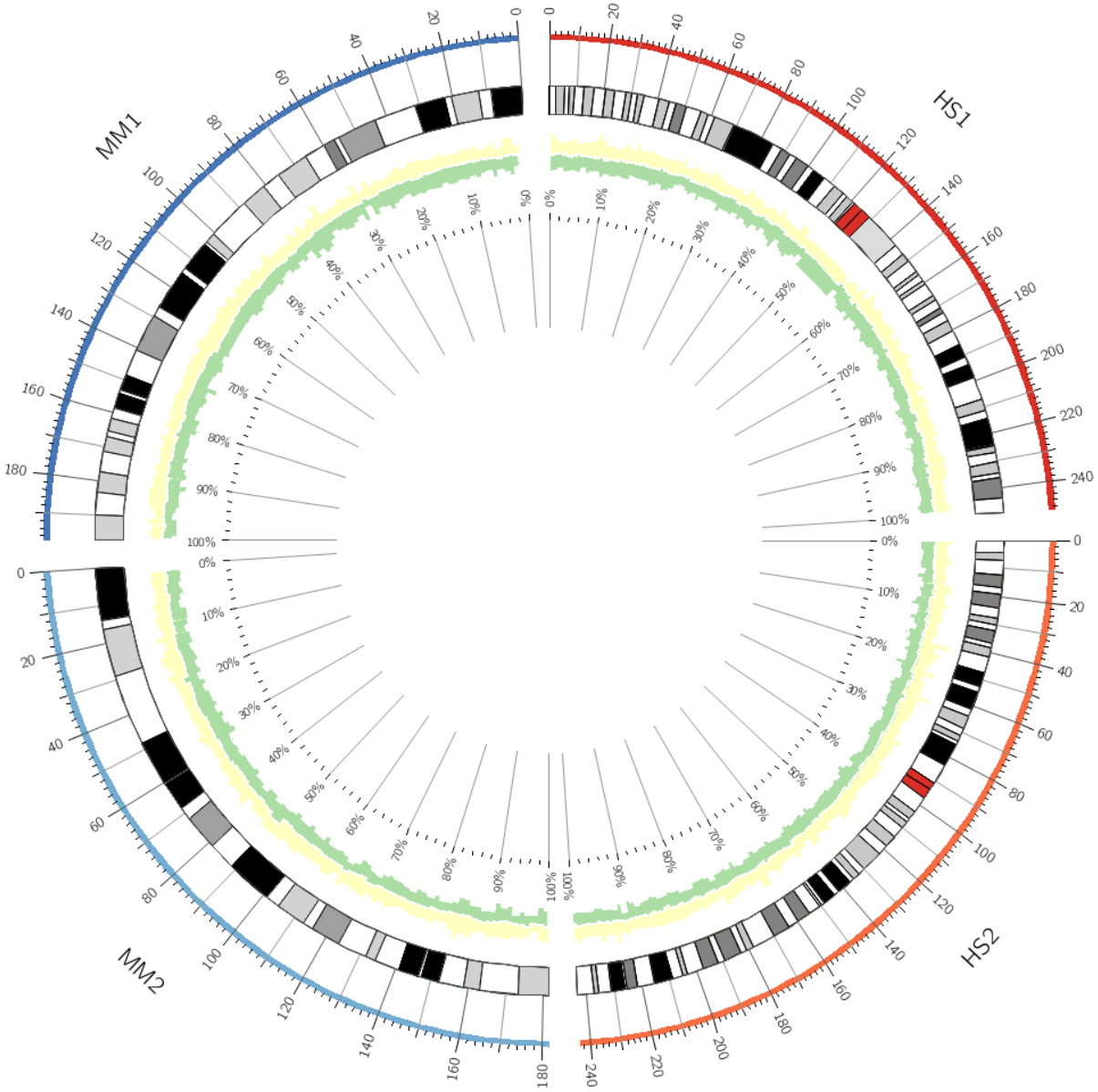
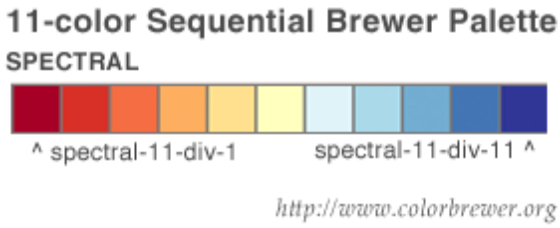


FIGURE 11

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.



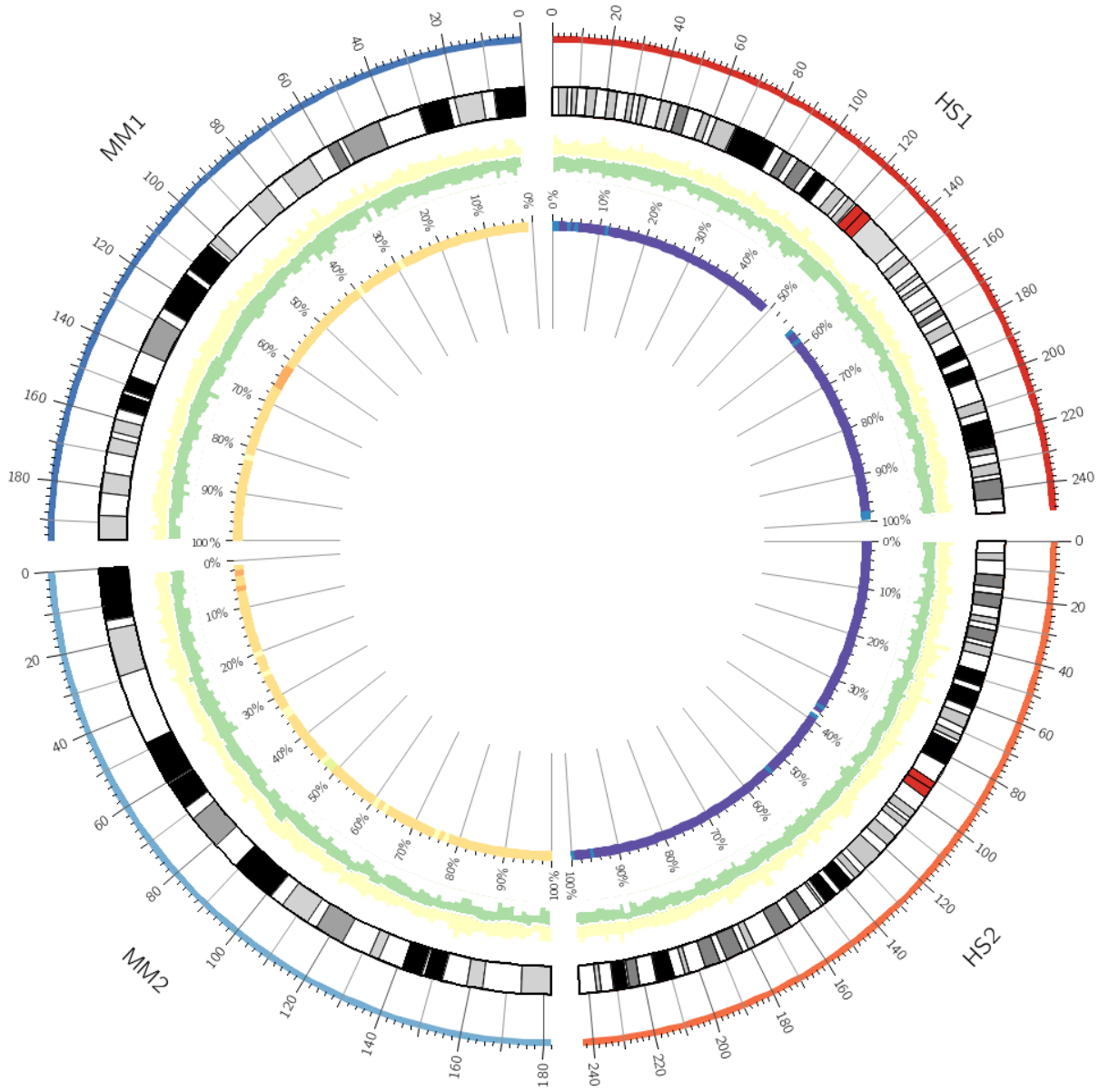


FIGURE 12

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.

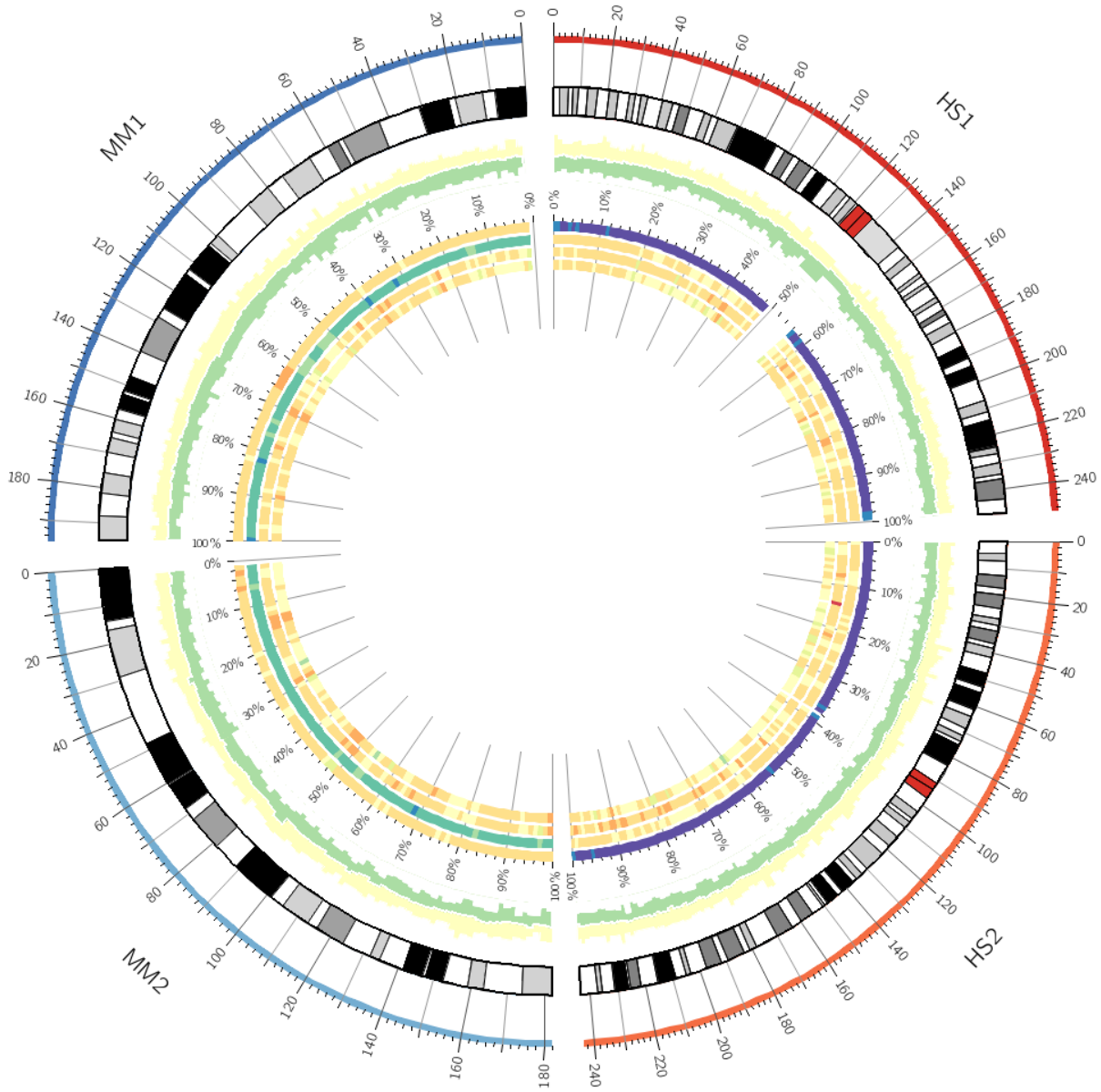


FIGURE 13

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).

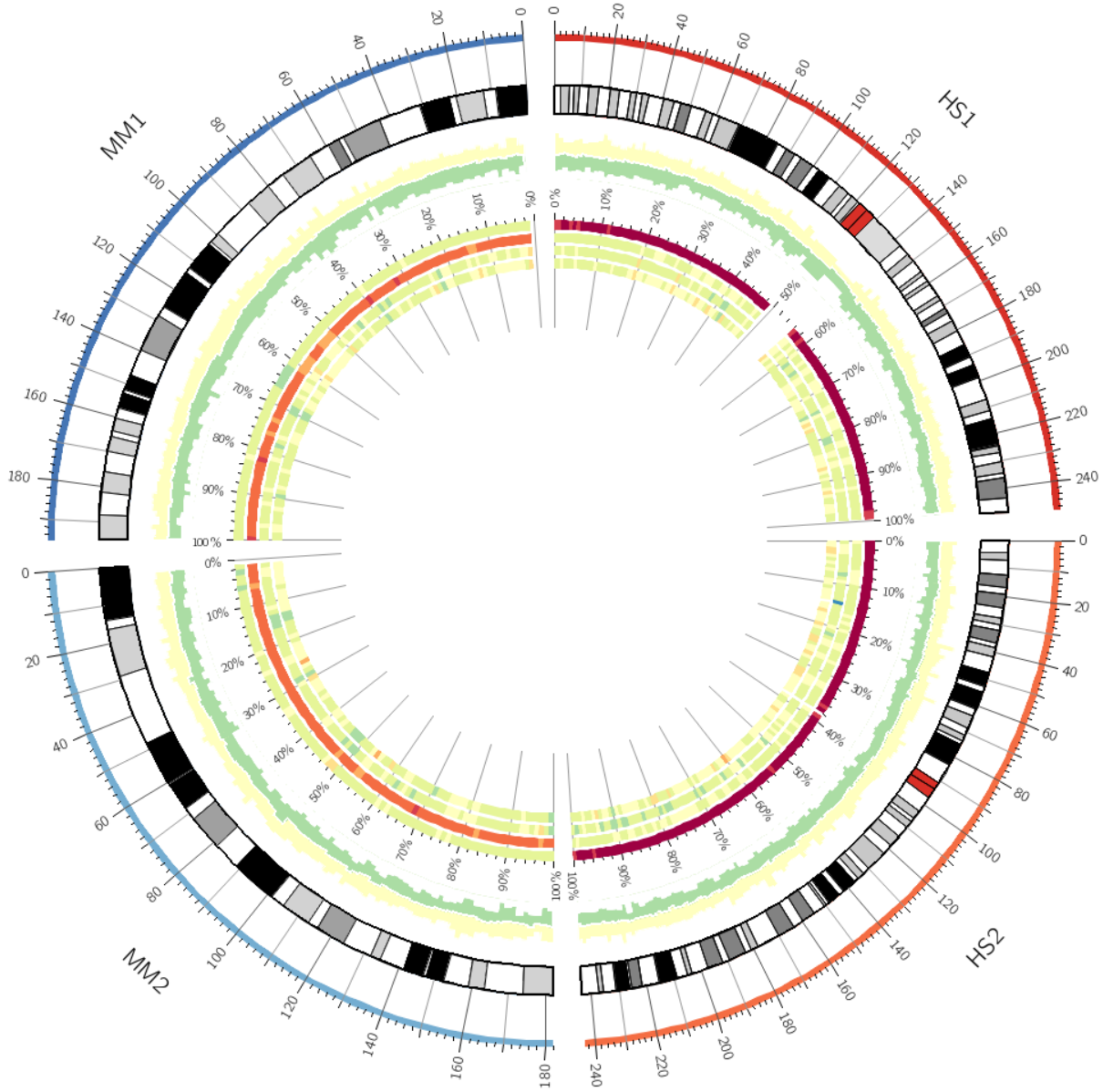


FIGURE 14

Heatmaps using a reverse spectral color scheme (`spectral-11-div-rev`).

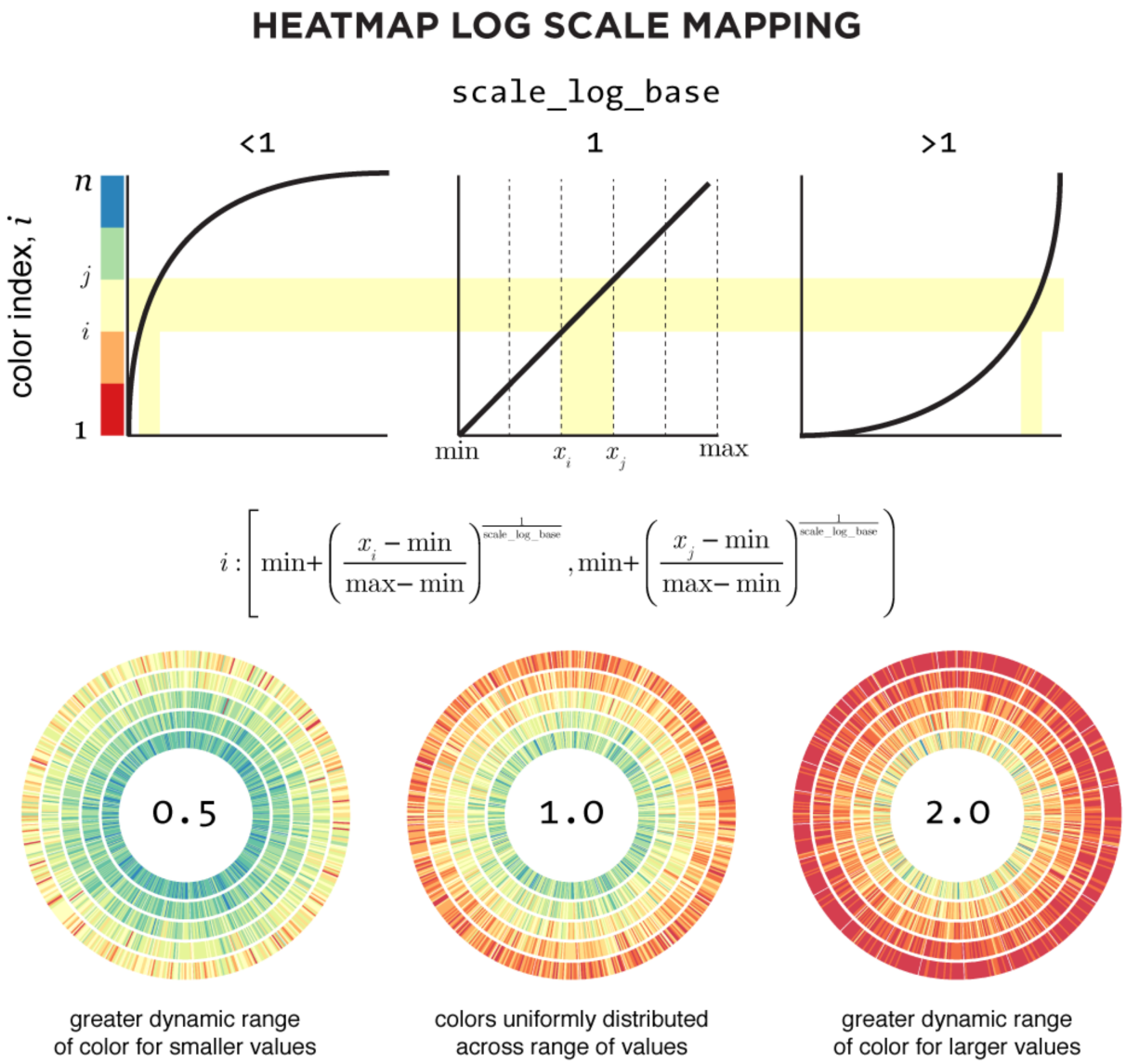


FIGURE 15

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

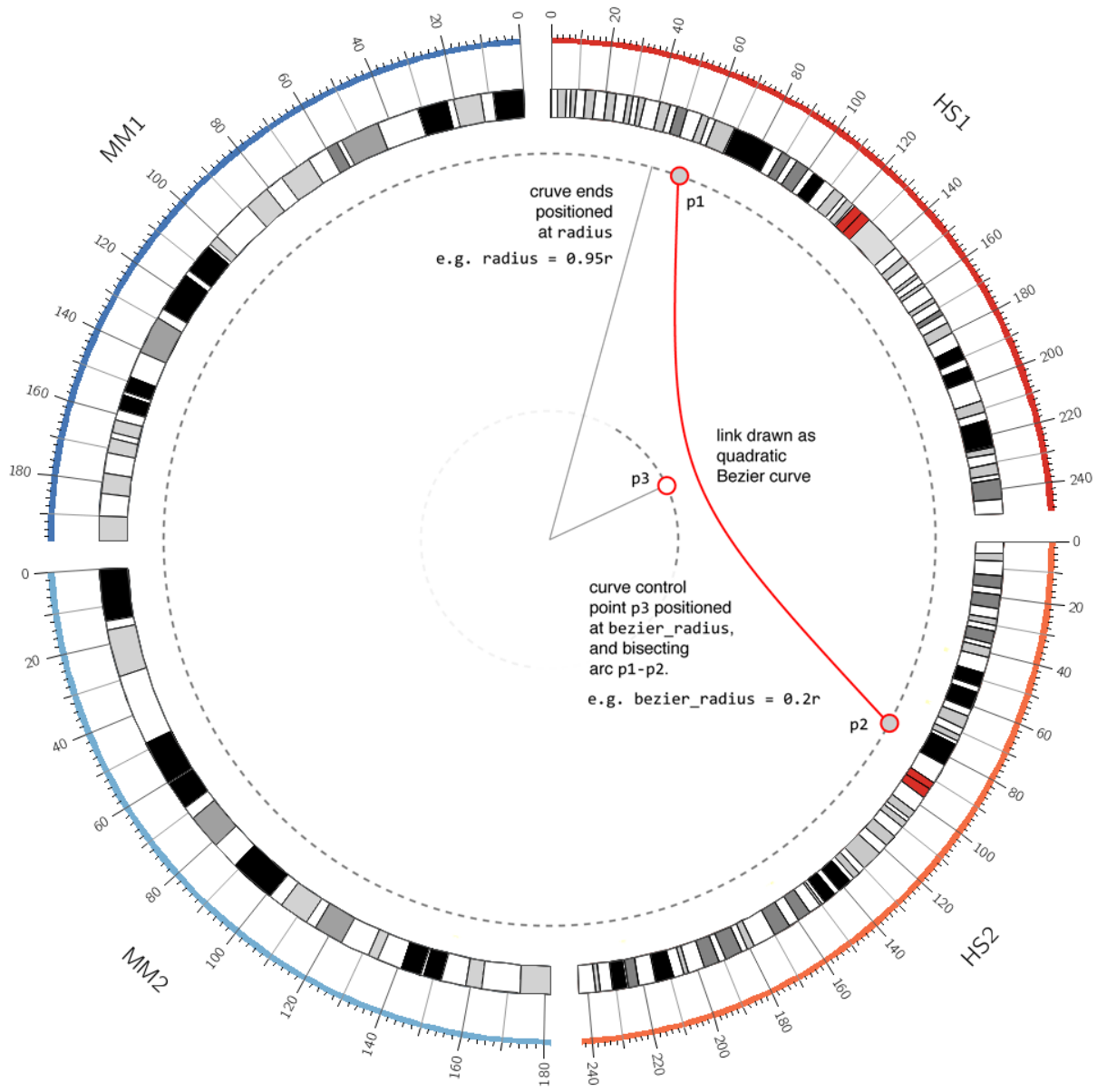


FIGURE 16

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.

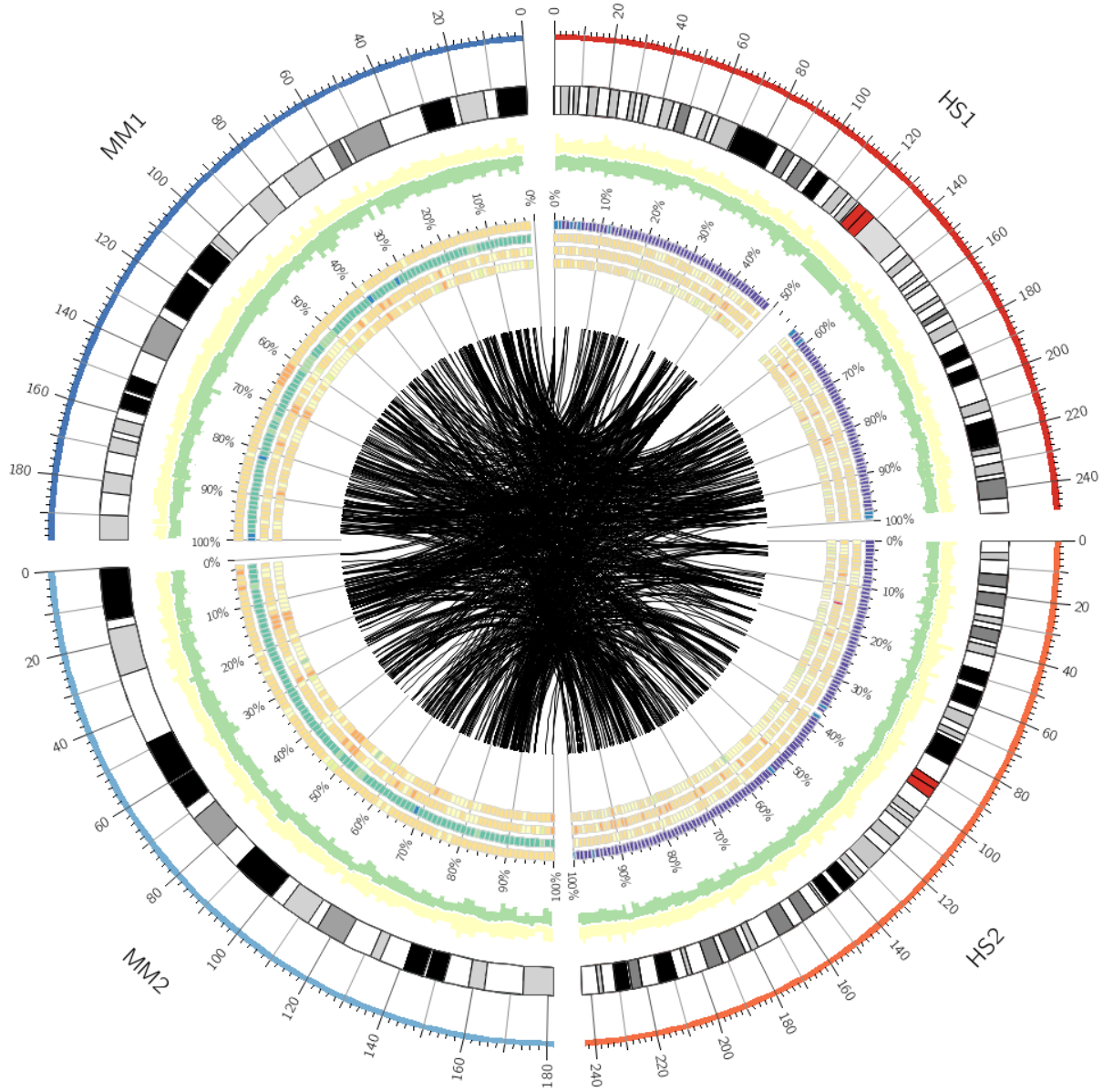


FIGURE 17

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

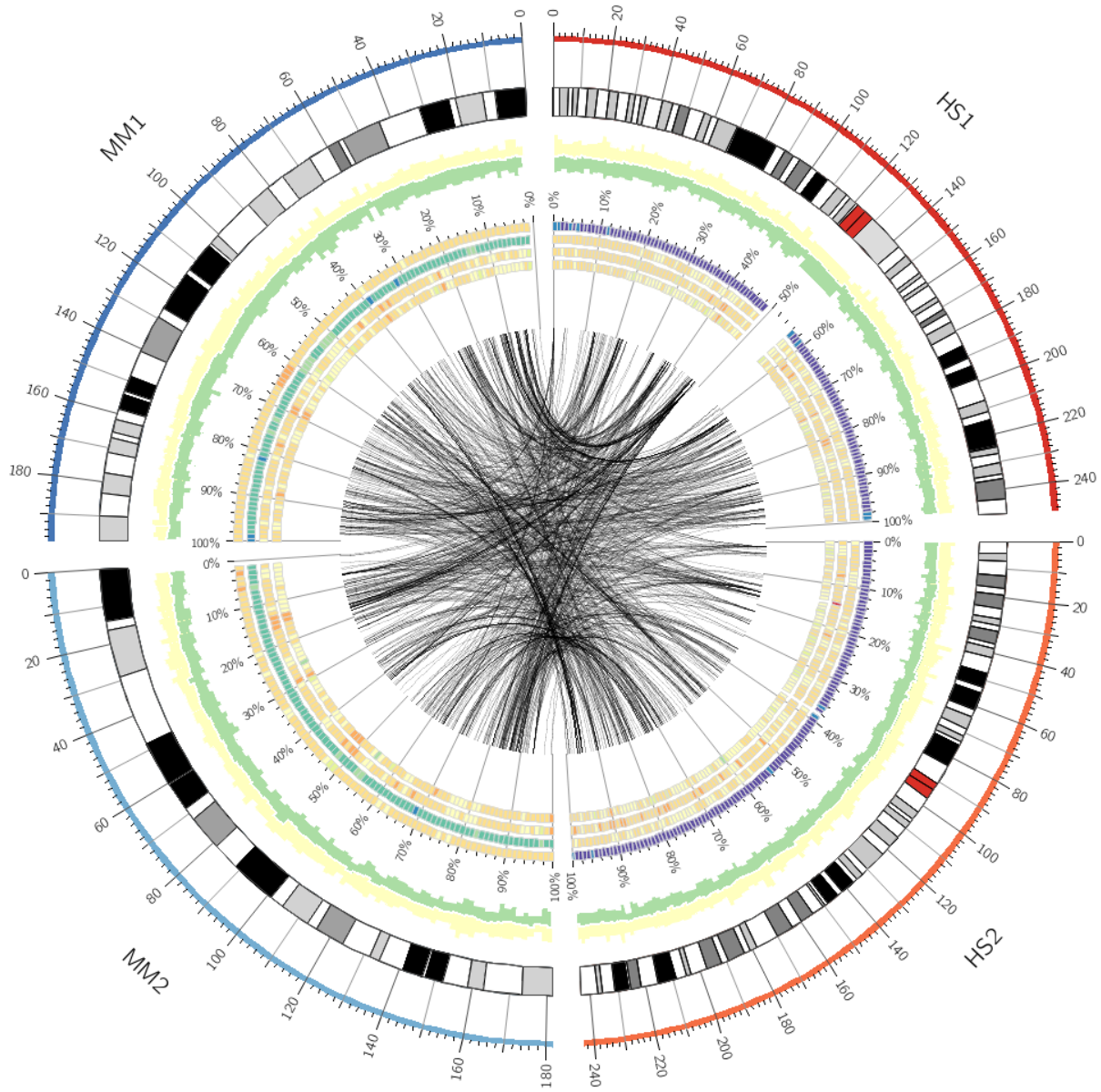


FIGURE 18

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

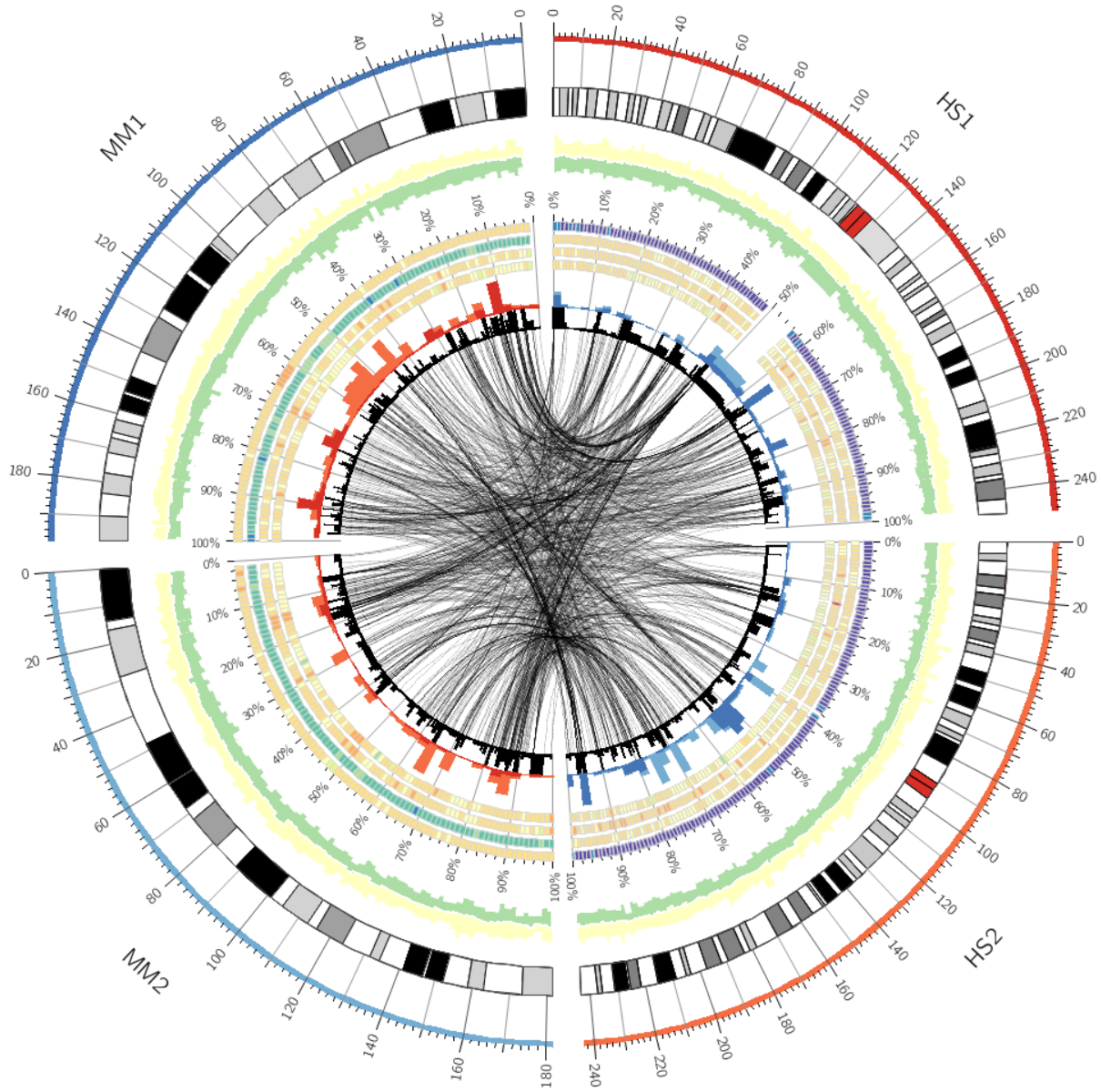


FIGURE 19

The two histograms between the heatmaps and links represent link density and were created by the `binlinks` tool. The inner histogram (black) 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.

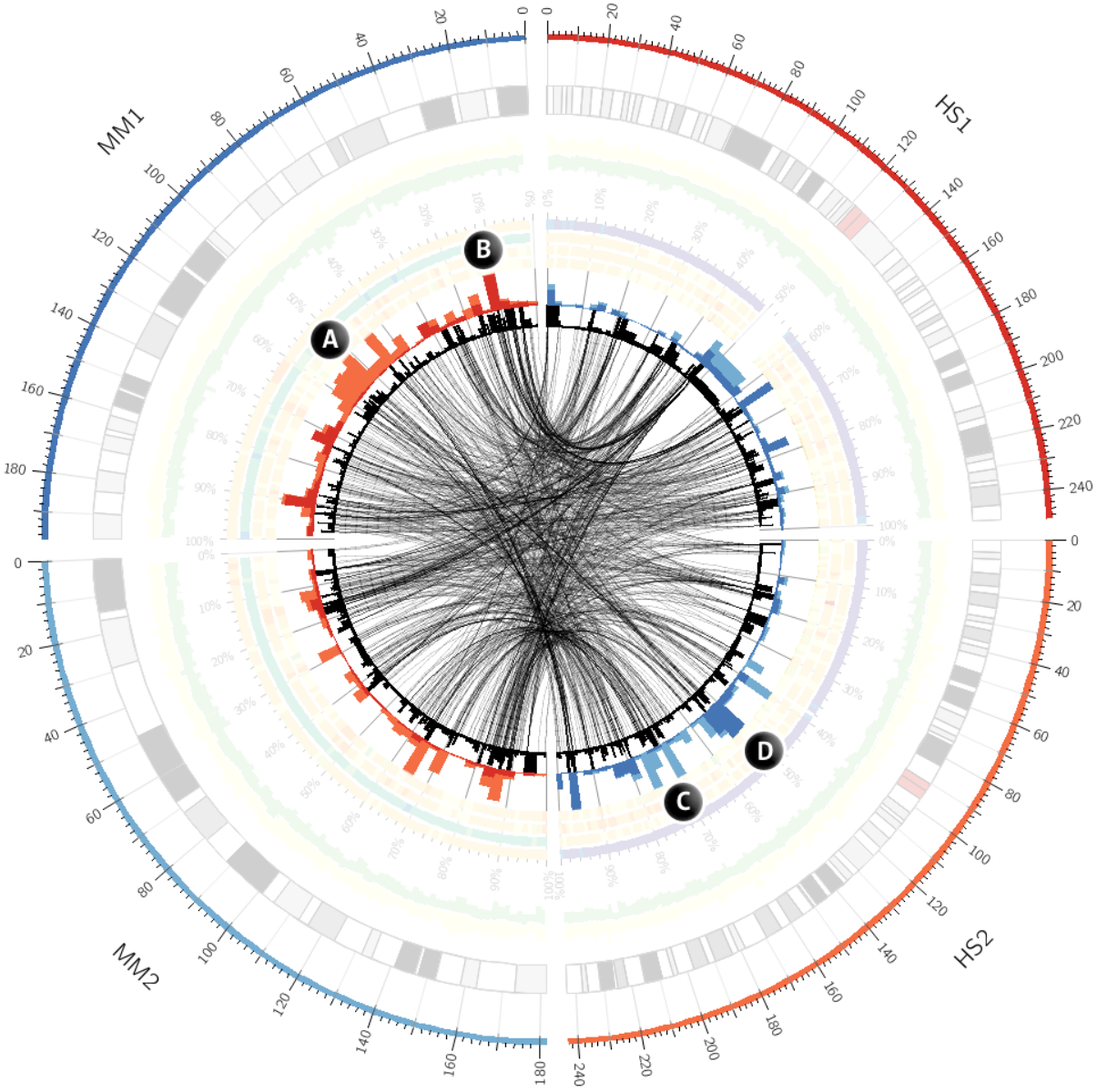


FIGURE 20

(A) Majority of links from `hs2`. (B) Majority of links from `hs1`. (C) Majority of links from `mm2`. (D) Majority of links from `mm1`.

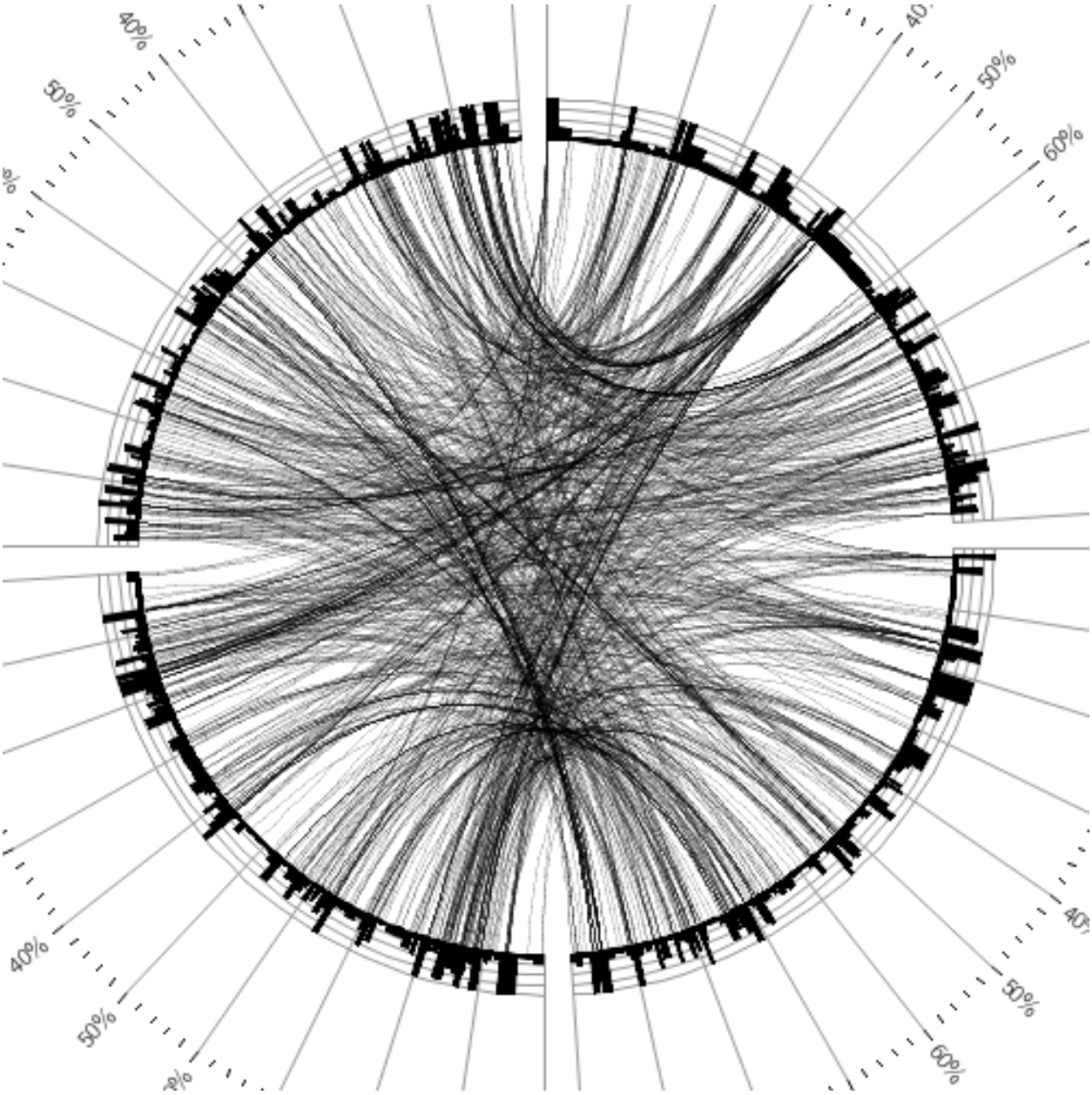


FIGURE 21

Axis grids defined with relative spacing, every `0.25r`.

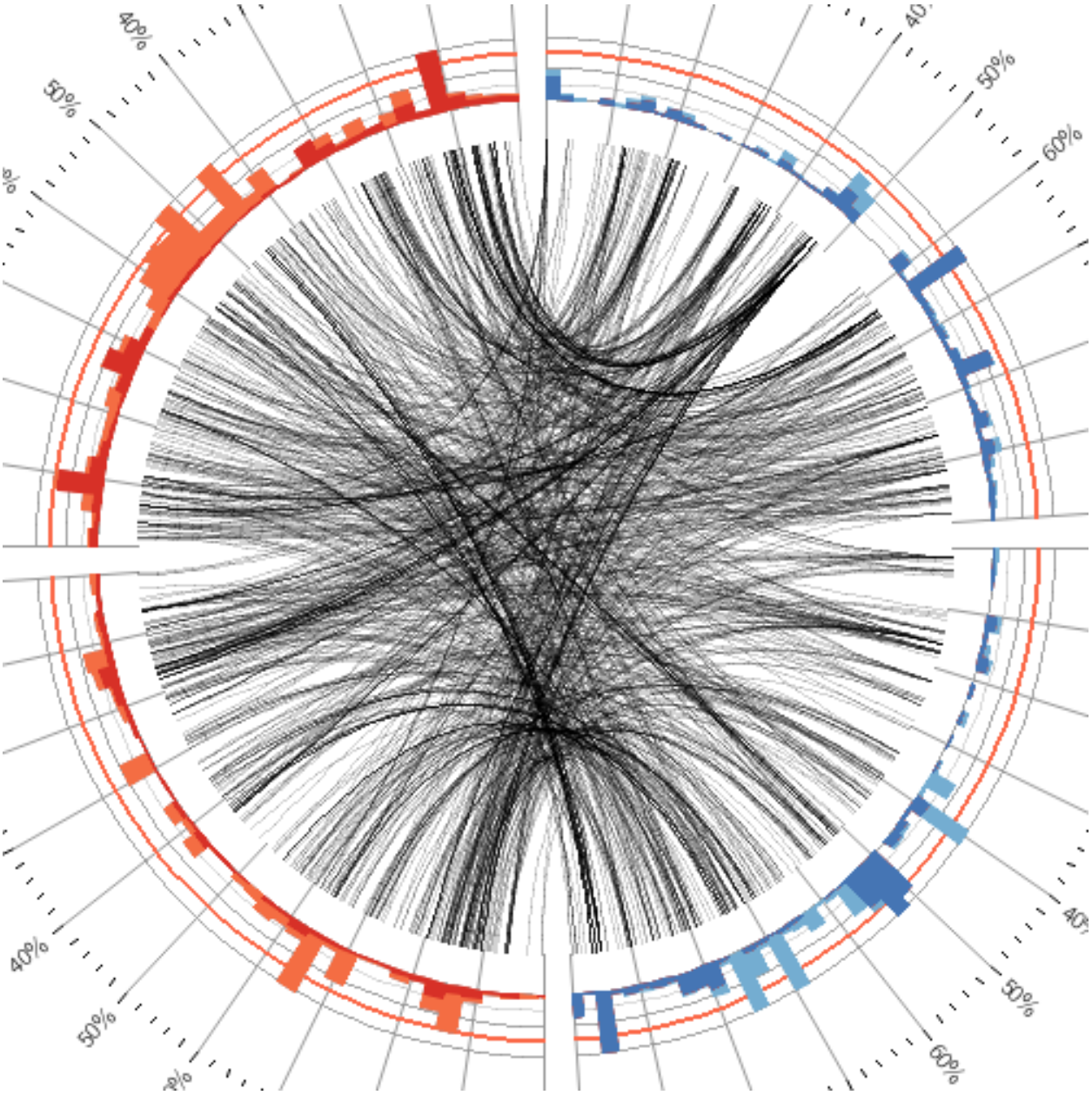


FIGURE 22

Axis grids can be defined at specific positions. The red grid is drawn at `0.75r`.

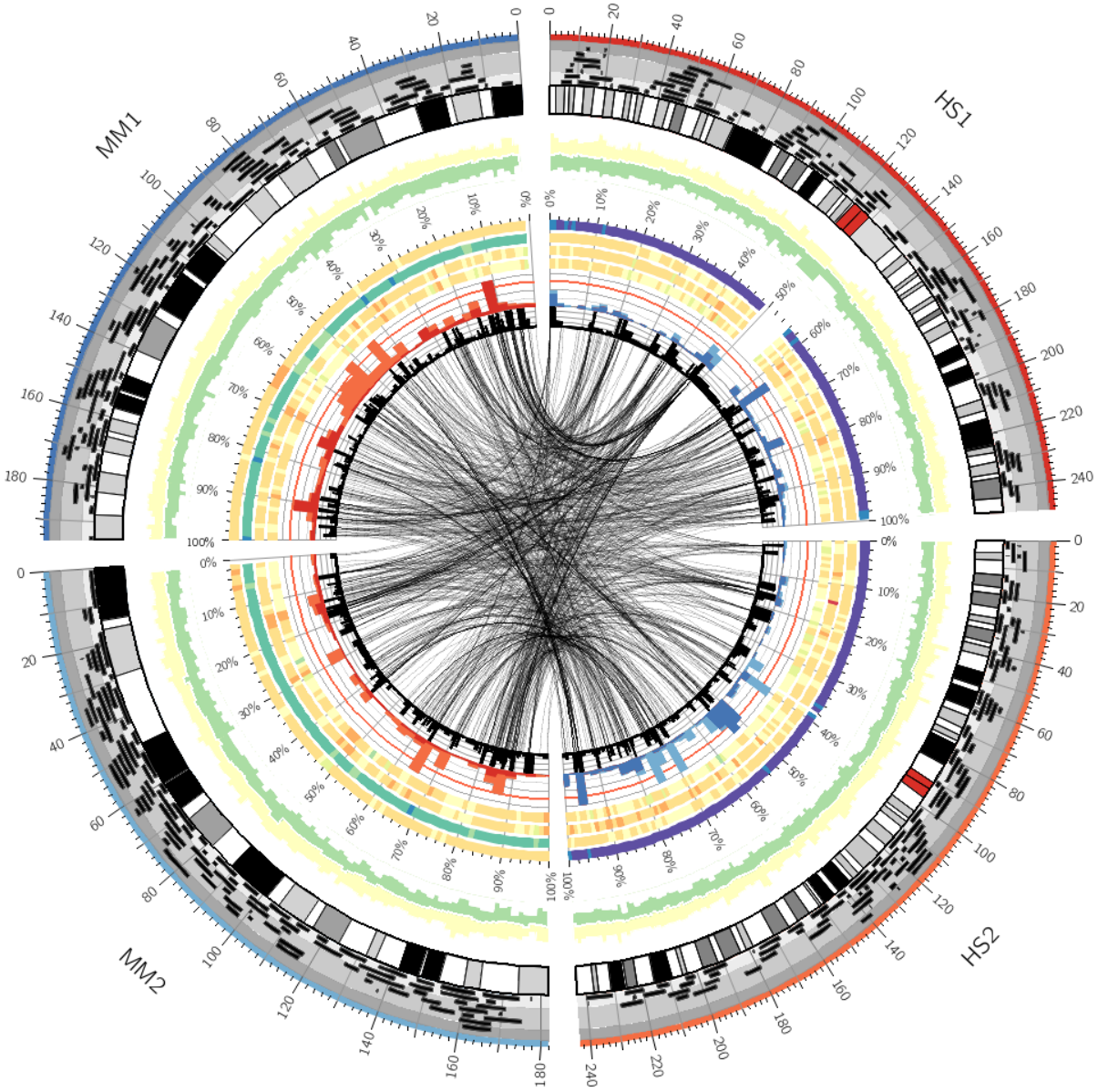
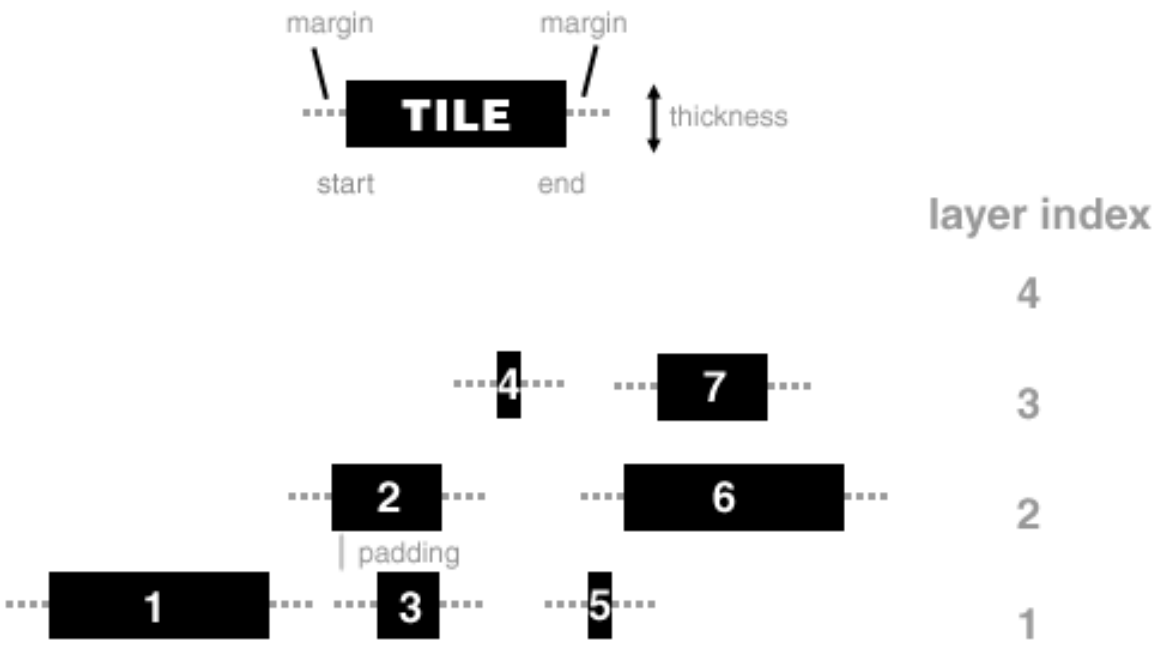


FIGURE 23

A tile track is shown outside the ideogram circle. A track can be drawn on a background of have any number of colored regions. Here the tile track has three grey regions.



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

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

image center

FIGURE 24

Tiles placement. Parameters such as `thickness`, `margin`, `padding` and `layers` control placement.

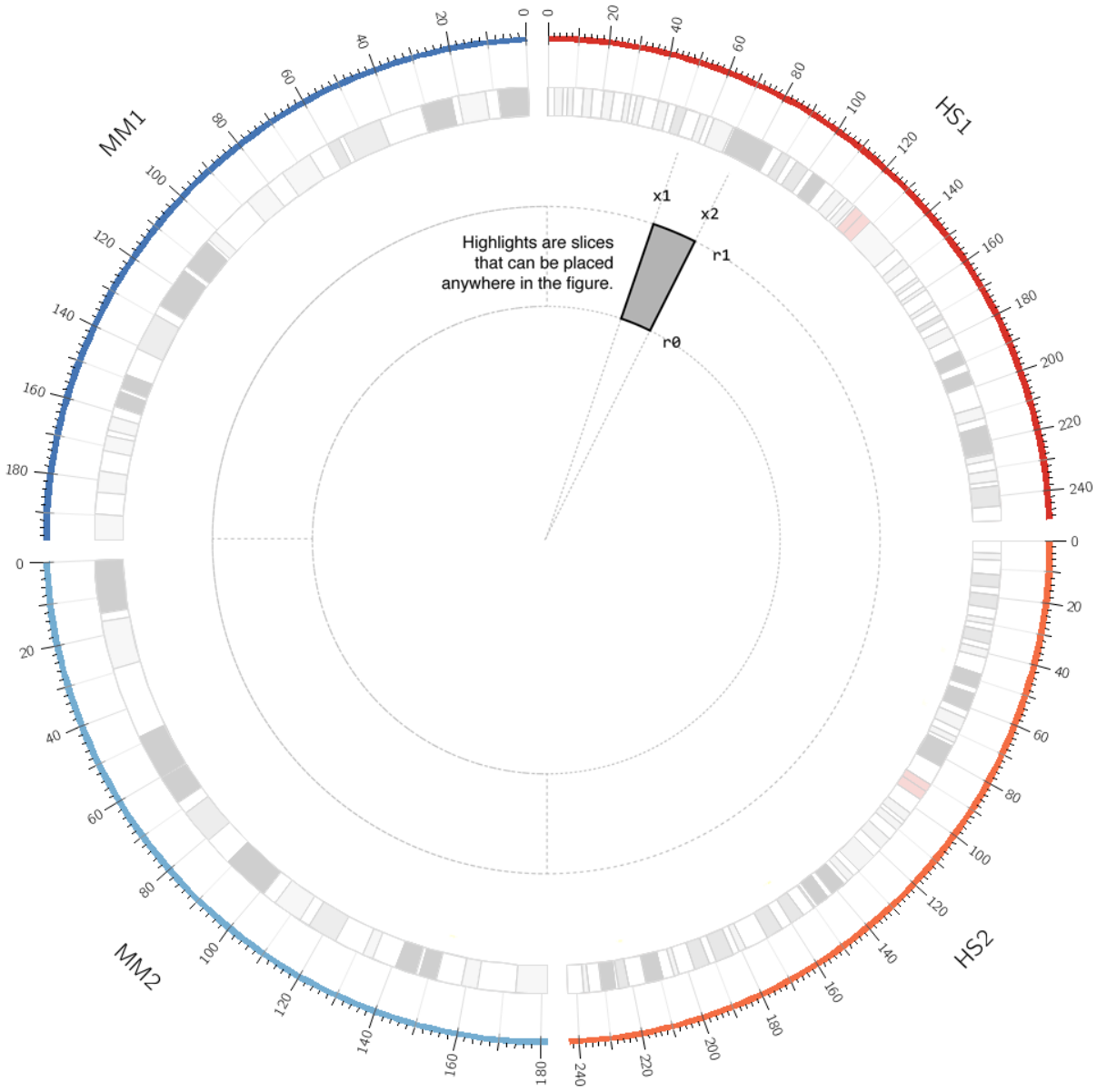


FIGURE 25

A highlight is a slice bounded by inner and outer radii `r0` and `r1` and start and end positions `x1` and `x2` on the same ideogram.

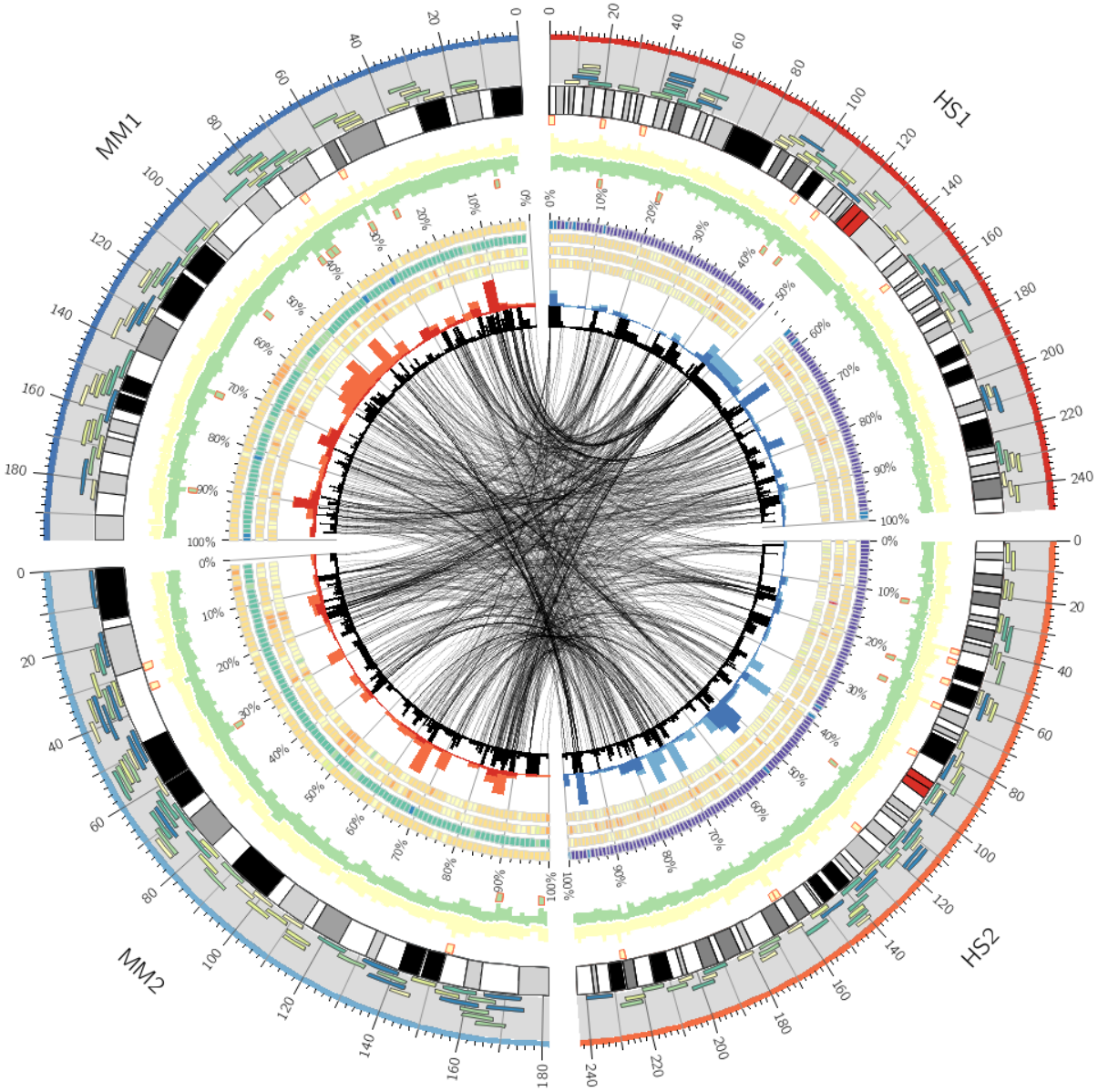


FIGURE 26

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.

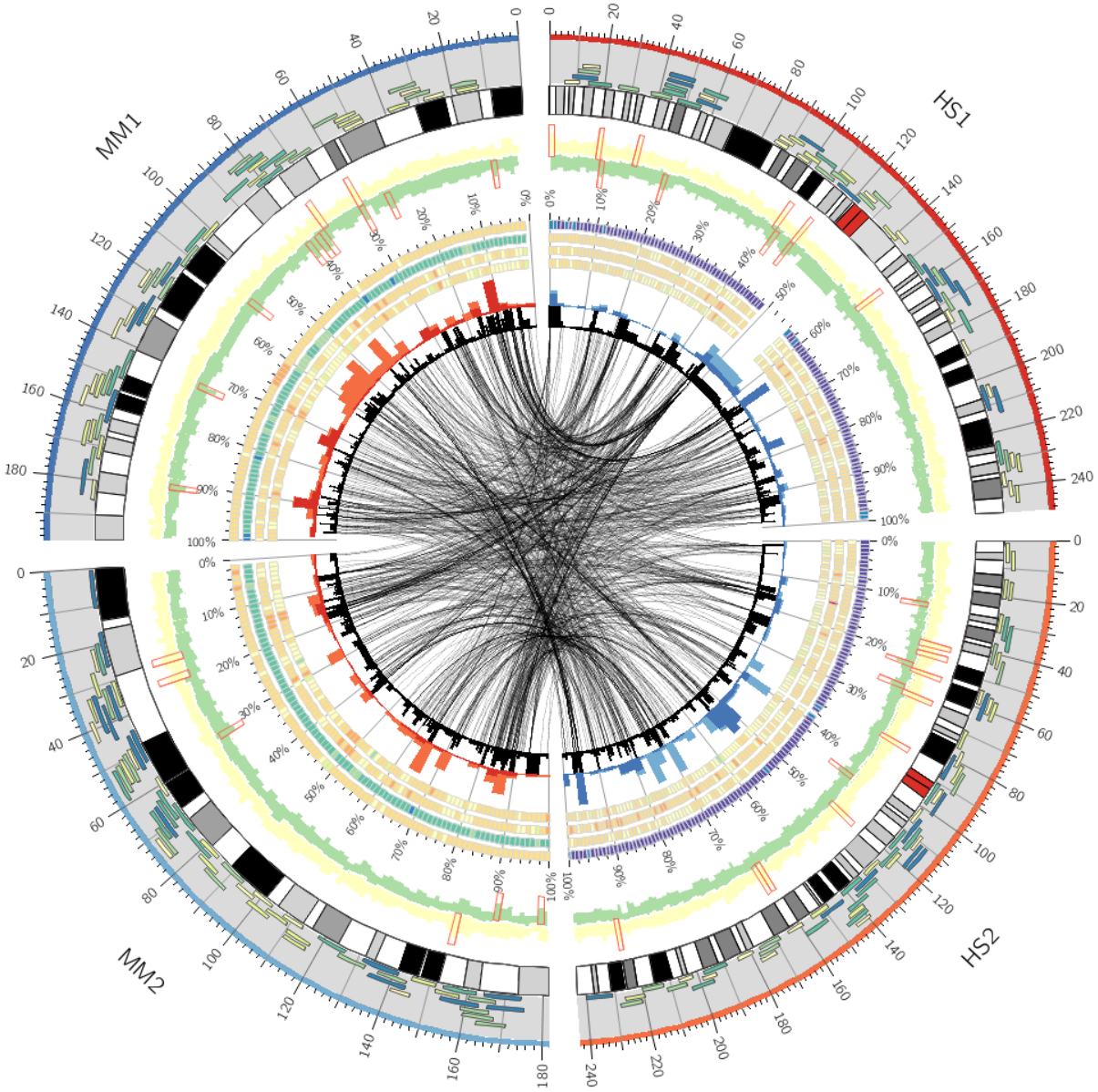


FIGURE 27

Highlights can be used to outline portions of the figure by drawing them as outlines.

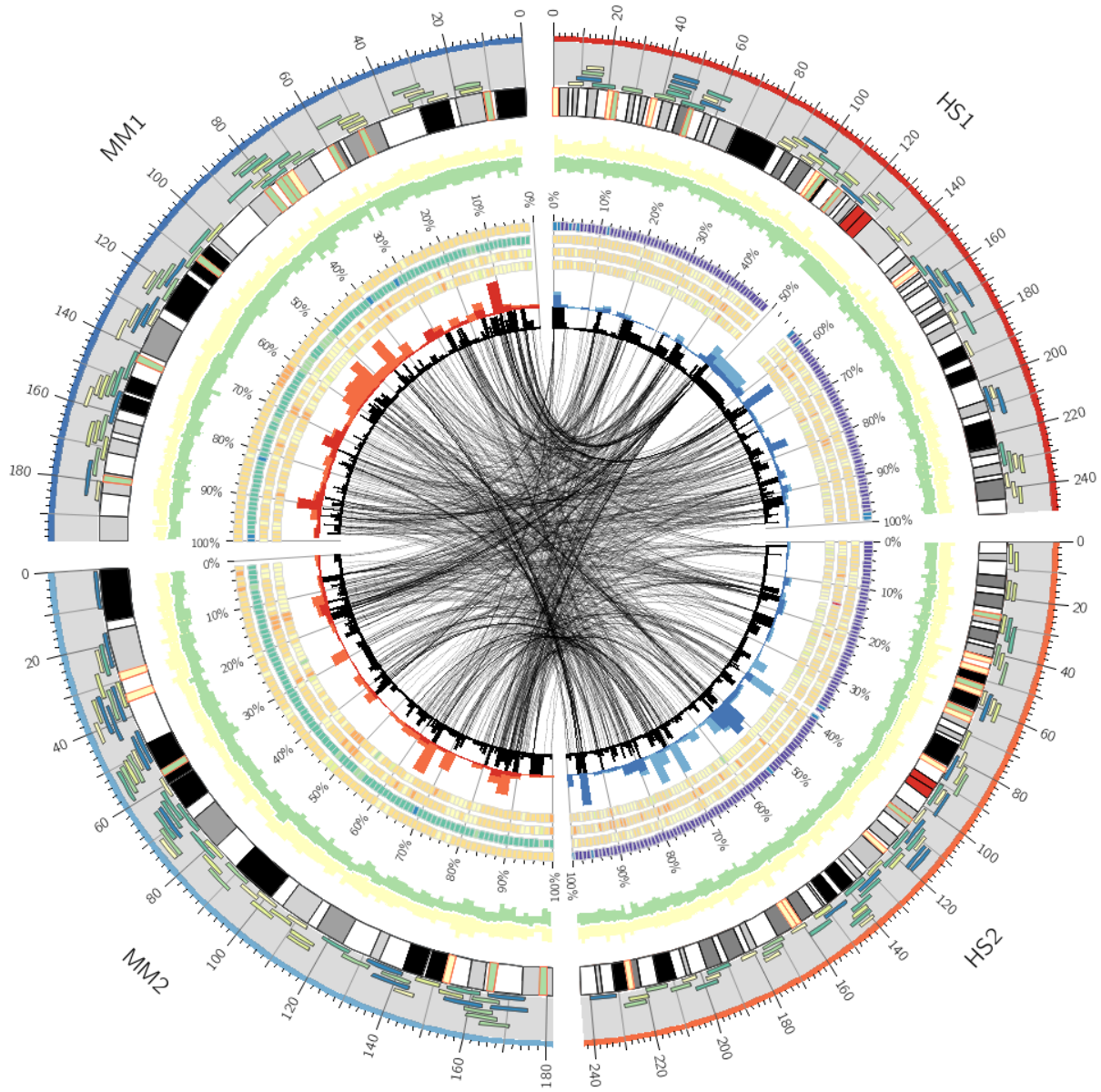


FIGURE 28

Highlights shown on top of ideograms by using the `dims()` macro which provides shortcut to ideograms' inner and outer radius.

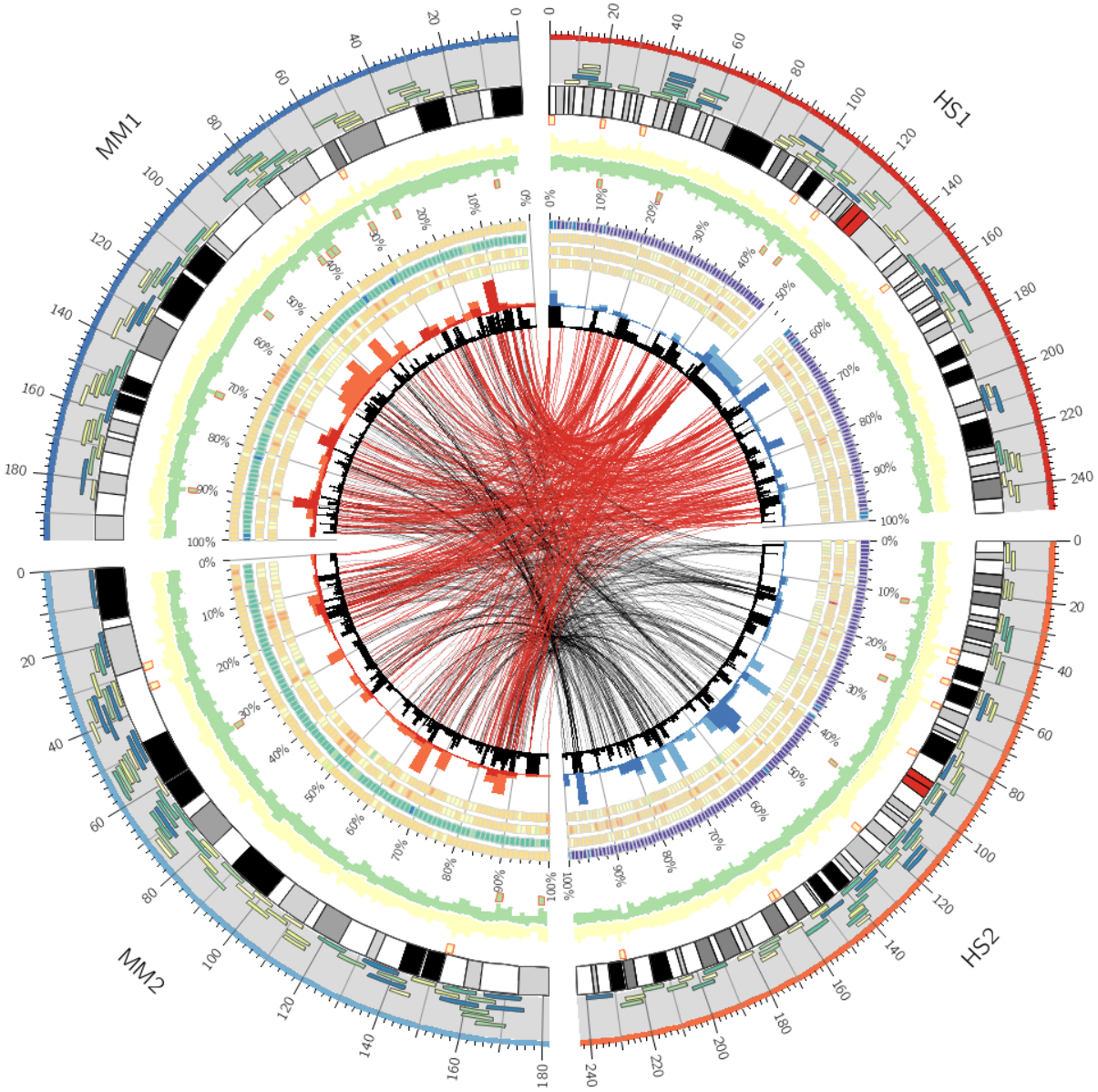


FIGURE 29

Using a rule, all links that start on `hs1` are colored by the ideogram's color.

```
<rule>
condition = on(hs1)
color     = rdylbu-11-div-2_a3
z         = 10
</rule>
```

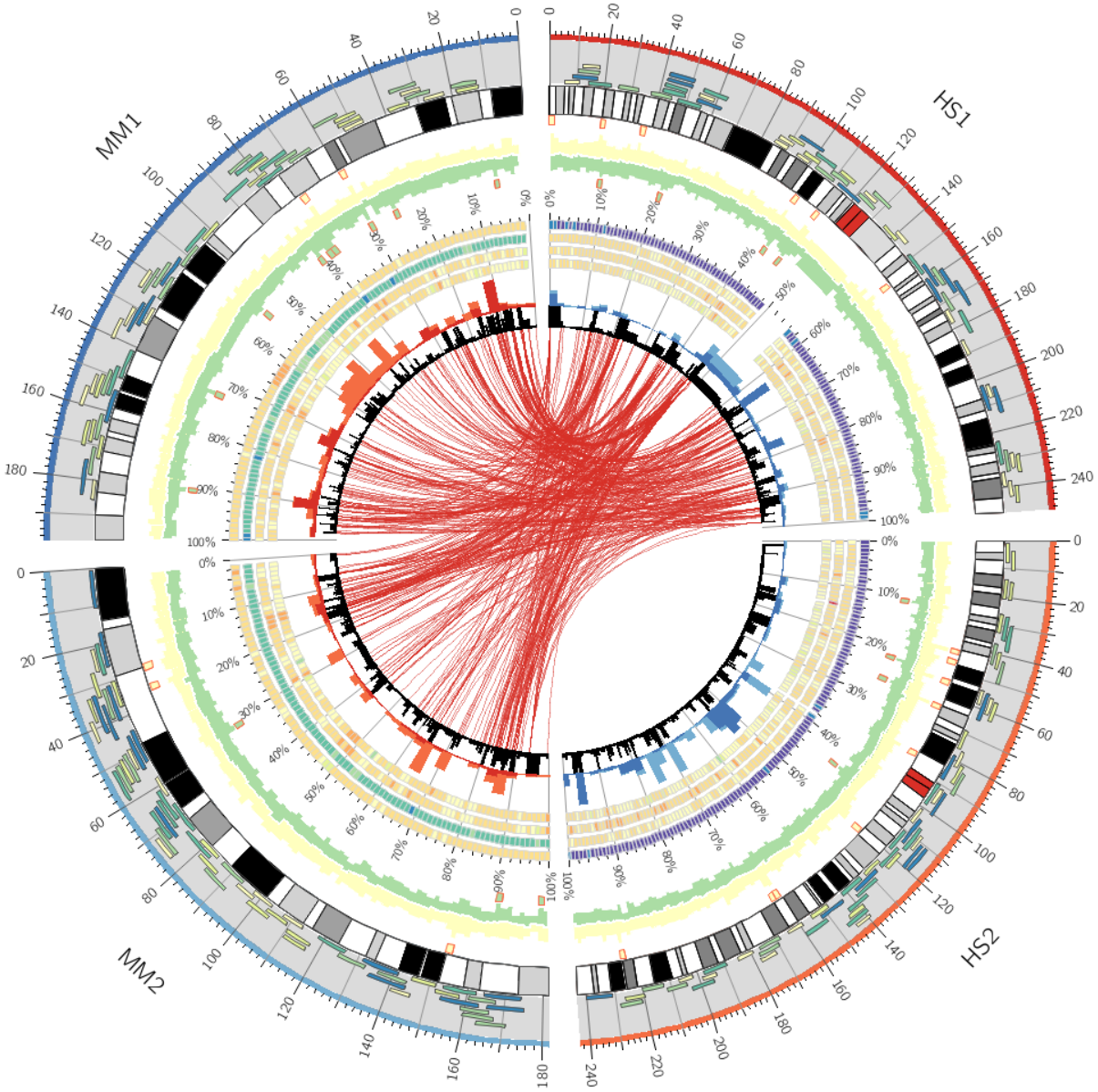


FIGURE 30

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

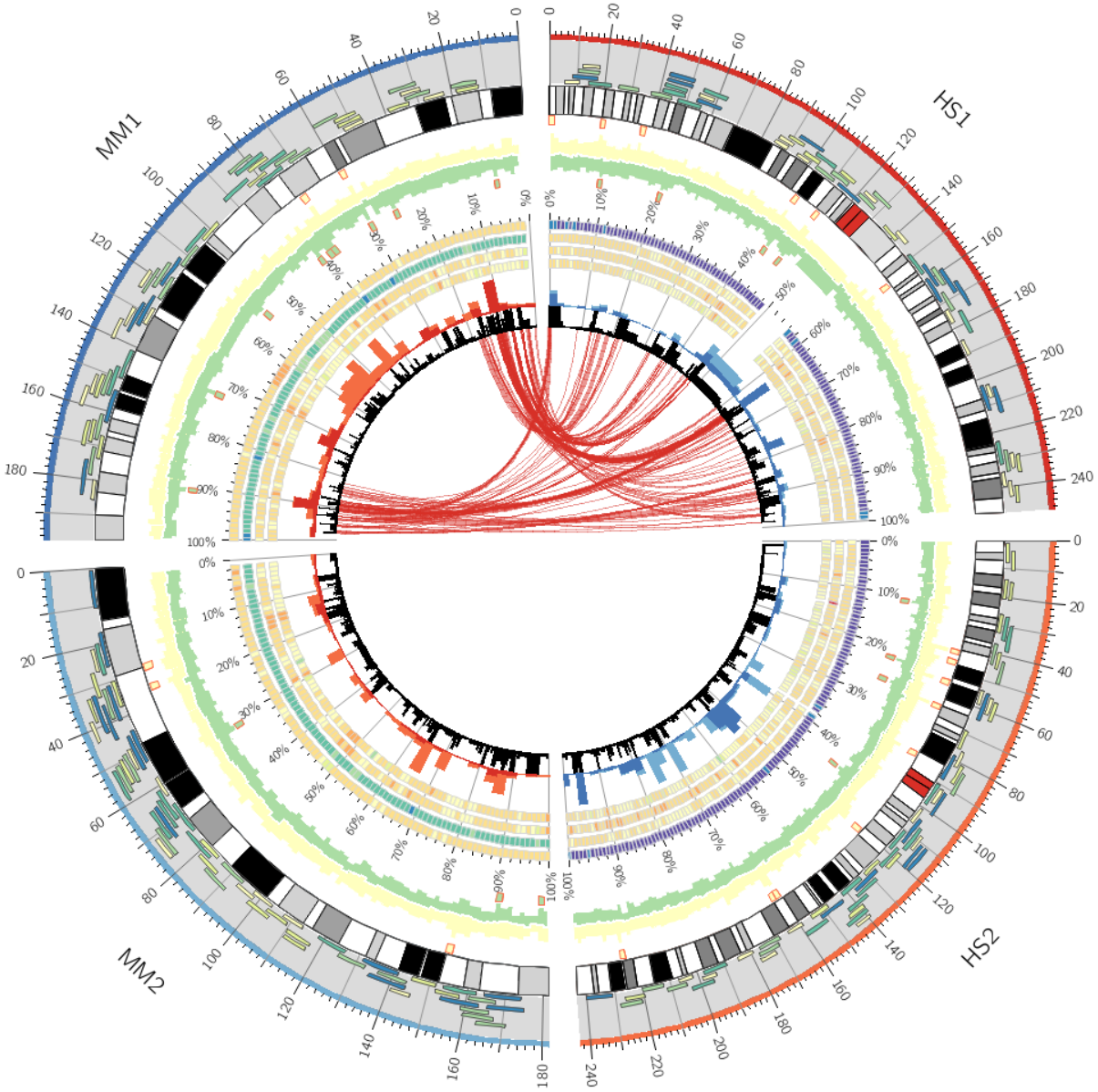


FIGURE 31

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.

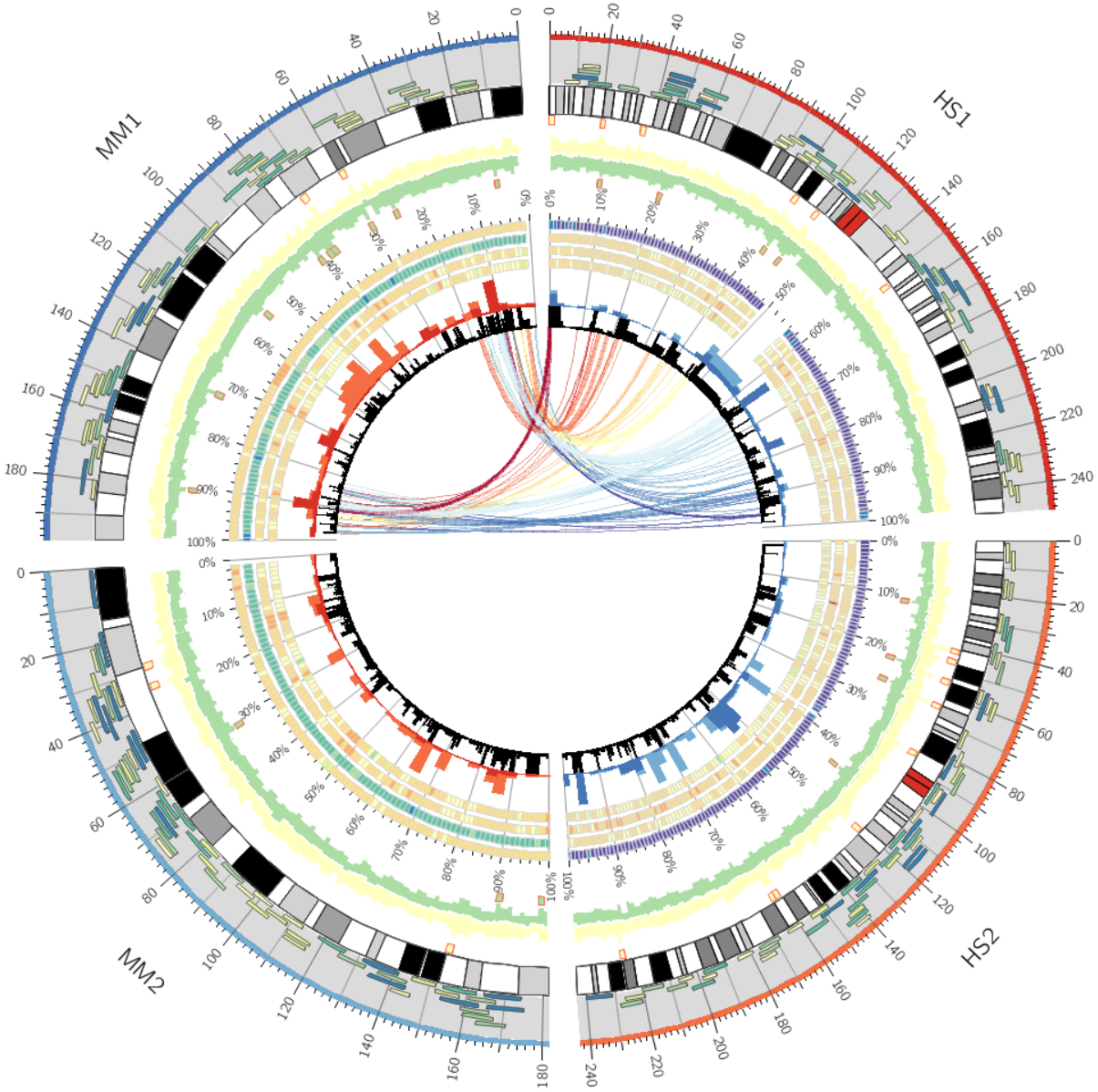


FIGURE 32

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.

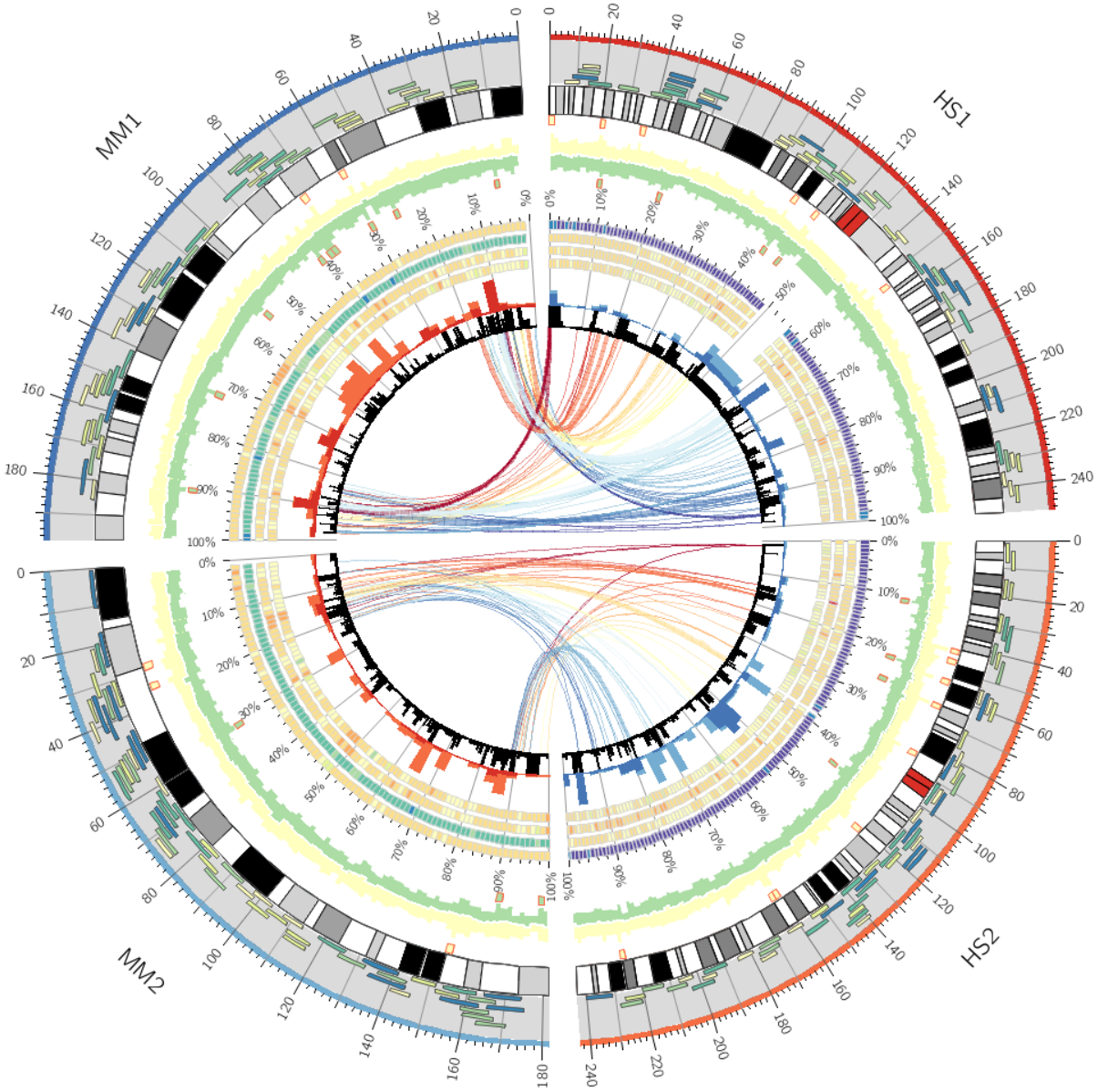


FIGURE 33

The rule from Figure 32 is duplicated to similarly format links between hs2 and mm2.