Version History
v0.15 12 Jul 2010
v0.14 30 Jun 2010
v0.13 30 Jun 2010
v0.12 29 Jun 2010
v0.11 29 Jun 2010
v0.10 21 Jun 2010

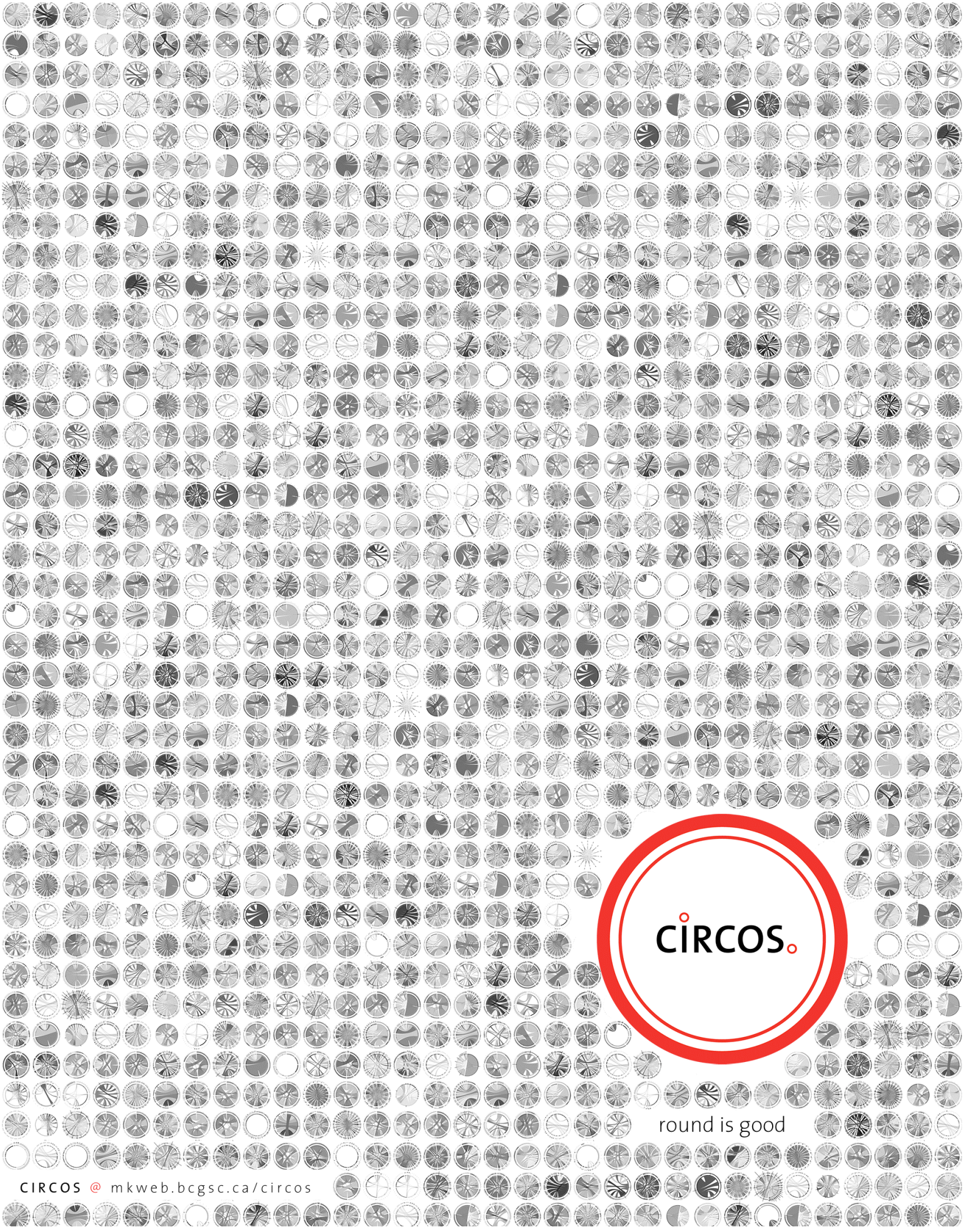
CIRCOS.
round is good
CIRCOS @ mkweb.bcgsc.ca/circos

Table of Contents

Introduction to Links and Rules

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

Rules are blocks which modify the format of data points (e.g. scatter plot points, histogram bins, etc) or links. Using rules you can adjust how a data point (or link) is display by altering its color, shape, geometry, transparency or visibility.

Figure 1 shows examples of how links and rules can be combined. Figure 1A shows a large number of links between three ideograms. In Figure 1B, the links are given a default grey color and rules have been used to turn links that start near specific regions red and black. In Figure 1C, rules were used to change the geometry and color of links that connect nearby intra-chromosomal region to make these links blue and point outwards. Rules in Figure 1D were used to adjust the thickness and color of the links.

Rules are extremely useful in creating texture in the links, which can be very dense and difficult to otherwise interpret. By changing link color, geometry and transparency, based on link position and/or size, rules allow you to quickly draw focus to specific parts of the data set without having to adjust the input data files.

Links that connect large regions are best drawn as ribbons, as shown in Figure 2. Thus, instead of using rules to adjust the thickness of the link line, as shown in Figure 1D, turning the links into ribbons automates this process. Furthermore, whereas line links always have constant thickness, the thickness of a ribbon link varies across the length of the link.

In this lesson we will create a figure that compares human chromosome 1 to the entire mouse genome, showing synteny between these regions.

All the data tracks for this session can be created using `sessions/4/data/create.tracks`. The same raw UCSC Genome Browser download data is used for this session as for the previous one.

Lesson 1 – Ideogram, Tick, Grid and Label Layout

In this lesson, we'll setup the ideogram, tick and grid layout for the rest of the images in this session. Using the combined human and mouse karyotype file, we select only the mouse chromosomes mm1-mm19 and human chromosome 1 (hs1) to be displayed. This is done using the `chromosomes` parameter.

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

<image>
file = s4-1.png
<<include ../../etc/image.conf>>

</image>

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

chromosomes_units           = 1000000
chromosomes_display_default = no

chromosomes         = mm1;mm2;mm3;mm4;mm5;mm6;mm7;mm8;mm9;mm10;
                      mm11;mm12;mm13;mm14;mm15;mm16;mm17;mm18;mm19;hs1
chromosomes_reverse = mm1;mm2;mm3;mm4;mm5;mm6;mm7;mm8;mm9;mm10;
                      mm11;mm12;mm13;mm14;mm15;mm16;mm17;mm18;mm19

chromosomes_breaks  = -hs1:120-140

chromosomes_scale   = hs1:11.8

<highlights>
<highlight>
show = yes
file = ../data/highlight.txt
r0   = 0.99r
r1   = 0.999r
</highlight>
</highlights>

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

We'll reverse the scale of all the mouse chromosomes using `chromosomes_reverse` so that both genomes start at the top of the figure. Since there is no data for the region around the centromere of human chromsome 1, we'll remove the region 120-140Mb using a break with the `chromosomes_breaks` parameter.

The highlights block provides the color index for the mouse chromosomes. We're using the UCSC Genome Browser color convention, but applying it to mouse chromosomes. The result is shown in .

To depict the details of information on human chromosome 1 at higher resolution, we'll change the scale of the chromosome so that it occupies ½ of the figure. To do this, we require that

$$\frac{s \cdot L}{G + S} = 1$$

where `s` is the scale, L is the amount of chromosome 1 that is shown (remember, we've removed 20Mb), `G` is the size of the mouse genome shown (chrs 1-19) and `S` is the total size of spaces in the figure. Solving for s yields approximately 11.8.

When the scale of human chromosome 1 is adjusted to 11.8, the result is Figure 4.

As before, 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 tick marks have been designed to accommodate the increased scale of hs1. Mouse chromosomes have 50Mb ticks and the human chromosome has additional 5Mb and 1Mb ticks.

Lesson 2 – Drawing Links

In this lesson, you will see how to create links. Links are represented by a pair of genomic regions. The link is used to connect these regions with a line, or a ribbon. Each of the regions can be of any size, but a region is confined to a single chromosome.

Links are defined in named `<link>` blocks which belong to an outer `<links>` block. The `radius` parameter specifies the radial position of the ends of the link and the `bezier_radius` parameter gives the radial position of the control point for the Bezier curve. The smaller the `bezier_radius`, the closer the link comes to the center of the circle. Note that the Bezier curve does not generally pass through its control point – the point is used to define the tangent of the curve at its ends.

```
<links>
<link chain>
file          = ../data/links.txt
bezier_radius = 0r
radius        = 0.85r
thickness     = 1p
color         = black_a5
</link>
</links>
```

Drawing the links as a 1 pixel thick line and using black with transparency (`black_a5`), the result is shown in Figure 5. Using transparent lines for links is a very good idea when the link data set is dense. Compare Figure 5, where transparency is used, with Figure 6, where each link is an opaque black line. The latter is very difficult to interpret.

What is Shown in the Figure?

For each 2 Mb window on hs1, the figure shows the top 20 alignments between hs1 and mouse chromosmoes 1-19. The link data is created using the `makelinks` script which is found in `sessions/3/data/ucsc`.

```
> cd sessions/4/data
> cat ../../3/data/ucsc/hg18.mm.chr1.txt | egrep -v -i "(random|x|y|chrm)" |
      ../../3/data/ucsc/makelinks -maxsize 1e6 -binsize 2e6 -numall 20 >
links.txt
```

The link data file stores links as two lines, one line for the start of the link and one for the end.

```
link129 hs1 46417 61698
link129 mm2 111313326 111326293
link2246 hs1 395264 426112
link2246 mm1 18566118 18592077
link2416 hs1 538220 574932
link2416 mm1 178603355 178629314
link1383 hs1 96349 115321
...
```

The two link lines are associated together using a unique identifier, which is the first field. The identifier can be any string, as long as it is unique for each link.

Lesson 3 – Rules – Coloring Links by Position, Part 1

Rules are used to change the format of data in the figure based on position, value and other characteristics of the data. Because links are typically dense and span many regions (some regions may be interesting, and some not), rules greatly aid in identifying meaningful patterns in links and communicating the results to the reader.

In this lesson, we'll see how to color links based on their start and end positions. When comparing syteny across a large number of chromosomes, it is useful to identify the relationship between a given chromosome and all others of the other species. This can be done by coloring all links that start (or end) at a given chromosome.

The rule below will color all links that end on mm11 using color chr11 (with transparency, `_a3`), change the `thickness` of the link to 2 pixels and set their `z` value to be higher than all other links, thus making these links drawn on top.

```
<rule>
condition   = _CHR2_ eq "mm11"
color       = chr11_a3
z           = 10
thickness   = 2p
</rule>
```

The result is shown in Figure 7. Notice that `_CHR2_` was used as a parsable field in the rule condition. This field is replaced by the chromosome of the end of the link. In the input data file, all links had hs1 as their start chromosome and a mouse chromosome as the end chromosome.

```
link2246 hs1 395264 426112
link2246 mm1 18566118 18592077
```

Thus, to test whether a link connects to mouse chromosome, we use `_CHR2_`. When comparing multiple species, or two individual chromosomes, it is helpful to always assign the same end of each link to the same species (or chromosome). This will make the rule condition simpler. For example, if all mouse chromosomes are always at the end of a link, the condition is

```
_CHR2_ eq “mm1”
```

rather than

```
_CHR1_ eq “mm11” || _CHR2_ eq “mm11”
```

Rules can be combined into rule chains. Multiple rules can be designed and applied in sequence to each data set. By default, rules are applied in order of their `importance` parameter and the first rule that matches terminates the rule chain. If you wish for a data point to continue to be tested with additional rules after being matched, use `flow = continue`.

Below is an extension of the rule above. The condition of the second rule requires that a data point be colored chr11_a3 (i.e. as assigned by the previous rule, thus this condition tests whether the previous rule matched) as well as requires that the start and end position be within 20-50Mb on hs1 (all links have their start assigned to hs1). Thus, the second rule applies to all links that start on 20-50Mb on hs1 and end on mm11. The rule colors these links red (with transparency, `_a3`), sets their `thickness` to 3 pixels and their `z` value to a high value, to make these links drawn on top (Figure 8).

```
<rule>
importance  = 100
condition   = _CHR2_ eq "mm11"
color       = chr11_a3
z           = 10
thickness   = 2p
# links that pass are tested by remaining rules
flow        = continue
</rule>

<rule>
importance  = 90
condition   = _COLOR_ eq "chr11_a3" && _START1_ > 20e6 && _END1_ < 50e6
color       = red_a3
z           = 20
thickness   = 3p
</rule>
```

We could have tested the second rule using

```
condition   = _CHR2_ eq "mm11" && _START1_ > 20e6 && _END1_ < 50e6
```

and obtained the same result. But for the purpose of the lesson, the condition `_COLOR_ eq "chr11_a3"` explicitly demonstrates that a rule’s condition can depend on a value set by a previous rule.

Lesson 4 – Rules – Coloring Links by Position, Part 2

In the previous lesson we've colored links to a single chromosome by the color index of that chromosome. In this lesson, we'll expand this to color any link by its destination chromosome.

Remember that in the data set for this session, every link has its start chromosome as hs1 and its end chromosome as one of the mouse chromosomes. Therefore, coloring a link by its start chromosome would not help distinguish links, because all links have the same start chromosome.

Let's start by coloring all links between hs1 10-20Mb by the end chromosome. What we'll need to do is develop an expression which turns the end chromosome (e.g. mm11) into a color (e.g. `chr11_a4`). To do this, we'll extract the "11" from the chromosome name like this

```
substr(_CHR2_,rindex(_CHR2_,"m")+1)
```

The `rindex()` function gives the position last instance of a character in a string (instance of "m" in `_CHR2_`) and `substr()` extracts a portion of a string from an offset (string in `_CHR2_` after last "m"). Once we have this, we'll add prefix "`chr`" and add suffix "`_a4`".

```
"chr" . substr(_CHR2_,rindex(_CHR2_,"m")+1) . "_a4"
```

Now that we have the condition, the rule is

```
<rules>
<rule>
condition   = _START1_ > 10e6 && _END1_ < 20e6
color       = eval( "chr" . substr(_CHR2_,rindex(_CHR2_,"m")+1)
                          . "_a4" )
z           = 10
thickness   = 2p
</rule>
</rules>
```

As before, since we're formatting only a small number of links with this rule, they are made thicker (`thickness = 2p`) and drawn on top of other links (`z = 10`). The result is shown in Figure 9.

All links can be formatted with this color scheme by setting the condition to evaluate to true for all links.

```
<rule>
condition   = 1
color       = eval( "chr" . substr(_CHR2_,rindex(_CHR2_,"m")+1)
                          . "_a4" )
#z           = 10
#thickness   = 2p
</rule>
```

Since all links are being colored, there is no need to make any links thicker or change their drawing order. Thus, `thickness` and `z` are commented out. The result is shown in Figure 10.

Lesson 5 – Bundling Links

As you can see from Figure 10, when a large number of links is drawn, the distribution of links can be difficult to discern. The links appear to cover the figure uniformly, making it hard to identify regions that are linked by multiple links. Moreover, using lines to depict links hides the size of the ends of the link.

Using the `bundlelinks` tool (see `tools/bundlelinks` in the Circos distribution) you can process a link file to generate a new file in which links have been bundled together, to form larger links, based on the distance between and size of the individual links.

http://mkweb.bcgsc.ca/circos/tutorials/lessons/utilities/bundling_links

This process is illustrated in Figure 11. Links that connect the same chromosomes (e.g. chrA and chrB, or chrA and chrC) are eligible to be grouped into a bundle (Figure 11A). A bundle is seeded from a single link, with additional links added to the bundle if the ends of the link are within `max_gap` to the bundle (Figure 11B). A different cutoff can be imposed on the start and ends of the link (useful if you are comparing different genomes and wish to control density of links in a bundle at only one end of the bundle). Bundles can be filtered by imposing a limit on the minimum number of links in a bundle.

The script `sessions/4/data/create.tracks` includes calls to `bundlelinks` to generate the bundle tracks for this session. For example, to bundle all links that are within 3Mb on hs1 into bundles with at least 3 links, the `max_gap_1` parameter is used to limit the distance between link starts.

```
> bundlelinks -links links.txt -max_gap_1 3e6
              -min_bundle_membership 3  > bundles.txt
```

The bundled links are shown in Figure 12 and below is the link and rules block for this track.

```
<links>
<link chain>
ribbon        = yes
file          = ../data/bundles.txt
bezier_radius = 0r
radius        = 0.85r
thickness     = 0p
color         = black_a10

<rules>
<rule>
condition   = 1
color       = eval( "chr" . substr(_CHR2_,rindex(_CHR2_,"m")+1) . "_a2")
radius2     = 0.99r
z           = eval(_SIZE1_)
</rule>
</rules>

</link>
```

First, notice that `ribbon = yes` is set. This turns the links into ribbons – curved regions whose ends reflect the size of the ends of the link in the data file. Because bundles have ends that are typically much larger than the ends of the links from which they are formed (see Figure 11A), using ribbons to draw bundles is a good idea.

As before, the rule block changes the color of the bundle to reflect the color index of the associated mouse chromosome.

From this lesson onwards, we'll draw the link ends closer to the mouse chromosomes. This can be done by adjusting the `radius2` parameter, which controls the radial position of a link's end (`radius1` controls position of the link's start).

Finally, the `z` value is made proportional to the size of the link on hs1. This has the result of larger bundles drawn on top.

If you would like to change the order of the links to draw the small links on top, change the `z` value to be inversely proportional to the link size. It's enough to make the `z` value to be the negative of the link size. Since links are drawn in order of ascending z, this scheme will place small links last.

```
z = eval(-_SIZE1_)
```

This change in `z` is shown in Figure 13

What is Shown in the Figure?

By bundling the links we've created a visualization of syntenic blocks between human chromosome 1 and the mouse genome. Evaluating these blocks is made difficult when a large number of links is present in the data set. Drawing all the links (Figure 10) results in a busy figure which is difficult to interpret.

By requiring that links be turned into bundles, which are formed from at least 3 links that are no further than 3Mb apart (pairwise) on hs1, it is possible to show patterns of similarity which involve multiple links. Recall that the links shown in Figure 10 correspond to individual alignments from the Chain/Net UCSC track. By bundling these alignments together, we are effectively creating a large gapped alignment.

The `sessions/4/data/create.tracks` a script creates six different bundle sets. Each set requires that bundled links be no further than 3Mb apart, but varies the minimum number of links per bundle from 2 to 20. These six bundle sets are shown in Figure 14.

```
# sessions/4/data/create.tracks
...
bundlelinks -links links.txt -max_gap_1 3e6
            -min_bundle_membership 20 > bundles.0.txt
bundlelinks -links links.txt -max_gap_1 3e6
            -min_bundle_membership 10 > bundles.1.txt
bundlelinks -links links.txt -max_gap_1 3e6
            -min_bundle_membership 5  > bundles.2.txt
bundlelinks -links links.txt -max_gap_1 3e6
            -min_bundle_membership 4  > bundles.3.txt
bundlelinks -links links.txt -max_gap_1 3e6
            -min_bundle_membership 3  > bundles.4.txt
bundlelinks -links links.txt -max_gap_1 3e6
            -min_bundle_membership 2  > bundles.5.txt
...
```

Lesson 6 – Density Histograms

We've seen density histograms in the previous session. These are histograms whose content is generated from a link file using binlinks. Their data summarizes the number (or size) of links that start or end within a certain position window.

In Figure 15 three density histograms are shown, defined by the blocks below.

The first block defines the histogram from `../data/histogram.mm.txt`, which is the histogram found outside the mouse ideograms. This histogram shows the total size of links fall within 5Mb windows on the mouse chromosomes.

The second block, from `../data/histogram.hs.txt`, counts the total size of links per 1Mb on the human chromosome, and colors them according to the mouse chromosome that is associated with the largest number of links for each window. This format is useful to show both (a) extent of similarity between one position (e.g. a 1Mb window) and all other locations and (b) identity of chromosome that contributes the largest fraction of links to this total.

Finally, the third block, from `../data/histogram.hs.stacked.txt`, expands the detail shown in the second histogram by showing the contribution from each mouse chromosome to every 1Mb window of hs1. You can tell that the histogram is formatted as a stacked histogram because the `fill_color` is a list and because in the data file the value for each bin is also a list.

```
...
hs1 81000000 81999999 1.0000,0.0000,0.0000,1.0000,0.0000,0.0000,
                      1.0000,0.0000,1.0000,0.0000,0.0000,0.0000,
                      0.0000,0.0000,1.0000,2.0000,0.0000,0.0000,0.0000
...
```

```
<plot>
show       = yes
type       = histogram
min        = 0
max        = 50e3
#r0        = 0.9r
#r1        = 0.98r
r0         = 1r
r1         = 1r+43p
file       = ../data/histogram.mm.txt
fill_under = yes
fill_color = black
</plot>

<plot>
show       = yes
type       = histogram
min        = 0
max        = 50e3
r0         = 0.9r
r1         = 0.98r
file       = ../data/histogram.hs.txt
fill_under = yes
fill_color = black
</plot>

<plot>
show       = yes
type       = histogram
min        = 0
max        = 25
r0         = 1r
r1         = 1r+43p
file       = ../data/histogram.hs.stacked.txt
# when using the normalized histogram, set max=1
file       = ../data/histogram.hs.stacked.norm.txt
fill_under = yes
fill_color = chr1,chr2,chr3,chr4,chr5,chr6,chr7,chr8,chr9,chr10,chr11,
             chr12,chr13,chr14,chr15,chr16,chr17,chr18,chr19
</plot>
```

The `binlinks` tool can create normalized stacked histograms, in which the total of each bin is normalized to 1. The maximum for the stacked normalized histogram is set to 1. This version of the stacked histogram is shown in Figure 16.

```
<plot>
show       = yes
type       = histogram
min        = 0
max        = 1
r0         = 1r
r1         = 1r+43p
file       = ../data/histogram.hs.stacked.norm.txt
fill_under = yes
fill_color = chr1,chr2,chr3,chr4,chr5,chr6,chr7,chr8,chr9,chr10,
             chr11,chr12,chr13,chr14,chr15,chr16,chr17,chr18,chr19
</plot>
```

Finally, `binlinks` can be asked to sort the bins based on value, placing larger bins at the bottom of the stacked histogram. This is shown in Figure 17. This approach can create a busy track, since the order for each bin varies based on individual values.

What is Shown in the Figure

Density histograms reflect the total number and size of links within a genomic window. These tracks were created from the link data file (`links.txt`) using the `binlinks` tool.

```
binlinks -links links.txt -link_end 1 -output_style 0
   -bin 5e6                   > histogram.mm.txt
binlinks -links links.txt -link_end 0 -output_style 1
   -bin 1e6 | sed 's/mm/chr/' > histogram.hs.txt
binlinks -links links.txt -link_end 0 -output_style 3
   -bin 1e6 -num              > histogram.hs.stacked.txt
binlinks -links links.txt -link_end 0 -output_style 3
   -bin 1e6 -num -normalize   > histogram.hs.stacked.norm.txt
```

The `-link_end` controls which end of the link is binned (`0` start, `1` end, `2` both) and the `-output_style` controls the detail in the histogram file (apply color, create stacked histogram, etc). When `-num` is used, the number (rather than size) of links is binned and `-normalize` applies a normalization factor to make the total bin size of stacked histograms unity.

For more information about `binlinks`, see

http://mkweb.bcgsc.ca/circos/tutorials/lessons/utilities/density_tracks/

Lesson 7 – Scatter Plots

Scatter plots use the same input format as histograms, but draw the data using glyphs. Glyph size and color can be controlled, as can glyph shape. You can probably guess that we'll be using rules to adjust glyphs!

In Figure 18, the scatter plot shows the average conservation score between human and mouse for each 1Mb window on hs1.

The default appearance of each glyph is a small grey square, as defined by this block.

```
<plot>
type       = scatter
file       = ../data/scatter.cons.txt
min        = 0.39
max        = 0.55
r0         = 0.80r
r1         = 0.90r
glyph      = square
glyph_size = 3
fill_color = grey
...
```

You'll notice what may appear to be a strange first block in the rule chain for this plot.

```
<rule>
importance = 110
condition  = 1
</rule>
```

This rules applies to all points (condition is always true), but appears to do nothing. Exactly right! It doesn't change the format of the point but it successfully short-circuits further testing with subsequent rules (remember that unless `flow = continue` is set, a rule that passes terminates the chain).

Rules adjust the color of the glyph based on the value. Independently, I calculated the statistics for the scatter plot, which are

$n = 229$

mean = 0.455

10% percentile = 0.416

90% percentile = 0.495

The rules below successively test each data point and color large values green, small values red and values very close to the average a dark grey.

```
<rule>
# 90% percentile
importance = 100
condition  = _VALUE_ >= 0.489
fill_color = green
flow       = continue
</rule>

<rule>
# 10% percentile
importance = 90
condition  = _VALUE_ <= 0.416
fill_color = red
flow       = continue
</rule>

<rule>
# within 1 std of mean
importance = 80
condition  = abs(_VALUE_ - 0.455) < 0.01
fill_color = dgrey
flow       = continue
</rule>
```

The size of the glyph can be adjusted – and be made a function of the data value. By adding this rule

```
<rule>
importance = 70
condition  = 1
glyph_size = eval( abs(_VALUE_ - 0.455)/0.005)
fill_color = eval(_fill_color_ . "_a3")
flow = continue
</rule>
```

the glyph size is proportional to the deviation of the value from the average. Because large glyphs overlap, I've added transparency to the color. The plot with adjusted glyph size is shown in Figure 19.

The scatter plot can be subverted into a type of glyph track, by mapping the value onto the size of the glyph (as was done above and shown in Figure 19), and then forcing the value of the plot to 0, using a rule like

```
<rule>
importance = 60
condition  = 1
value      = 0.47
</rule>
```

Note that we had to change the glyph size and value in two separate rules, because the rule that changes value might adversely affect the glyph size. By setting the value of each data point to 0, all the glyphs are at the baseline as shown in Figure 20.

There are other ways of achieving this, such as setting `r0` and `r1` to be the same value (thus collapsing a track onto a single radial position), or setting a very small min and very large max (e.g. `min = -100`, `max = 100`).

Lesson 8 –Rules – Coloring Links by Size

In the final lesson of this session, we will apply a different coloring scheme to the bundles.

First, let's map all the links onto a grey scale, with the shade of grey proportional to the size of the link. This can be achieved by mapping the size onto transparent versions of black (`black_a1` ... `black_a5`) using this rule.

```
<rule>
importance = 100
condition  = 1
color      = eval( "black_a" . int(max(1,6-_SIZE1_/5e6)) )
#flow       = continue
</rule>
<rule>
```

The result is shown in Figure 21.

Now let's add two more rules which focus on links on mm8, mm11 and mm5.

The first rule will color mm8 and mm11 links based on the color of the mouse chromosome, with light transparency.

```
<rule>
importance = 90
condition  = _CHR2_ eq "mm8" || _CHR2_ eq "mm11"
color      = eval( "chr" . substr(_CHR2_,rindex(_CHR2_,"m")+1) . "_a1")
z          = eval(_SIZE1_)
</rule>
```

This kind of selective coloring is useful if you are interested in knowing which regions on hs1 are related to mm8 and mm11.

With the second rule, we'll highlight links on mm5, but to a lesser extent than with the rule above. We'll use greater transparency (_a3 instead of _a1) and set a `z` value which places the mm5 bundles above those not affected by rules (by default, z = 0), but below the bundles matched by the above rule (whose z value was set to the bundle size, which is much larger than 10).

```
<rule>
importance = 80
condition  = _CHR2_ eq "mm5"
color      = eval( "chr" . substr(_CHR2_,rindex(_CHR2_,"m")+1) . "_a3")
z          = 10
</rule>
```

The effect of these two rules is shown in Figure 22. The red links (mm5) sit on top of the black links, but below the orange (mm8) and green (mm11) links. The red tint is subtle (transparency `_a3`). On top of the red links, we have the orange and green links, which overlap based on their size (larger links drawn on top).

□

Figures

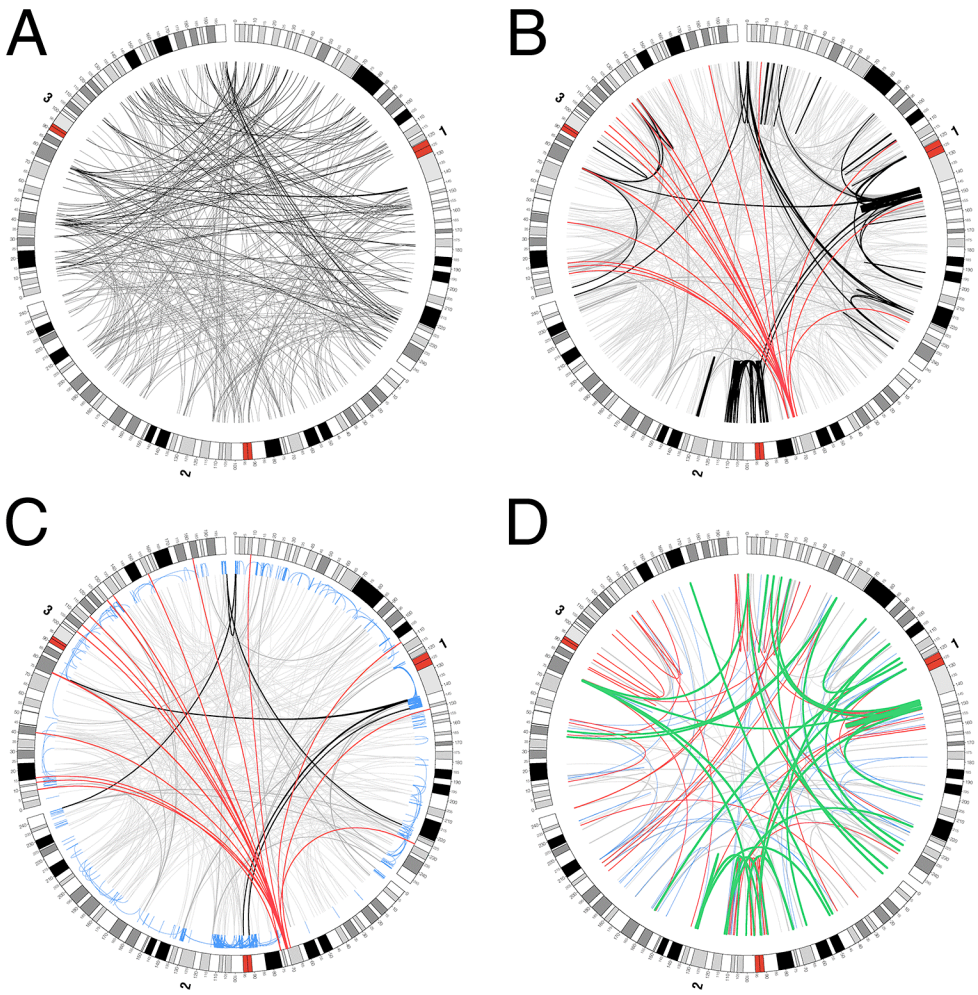

FIGURE 1

Examples of how rules and links are combined. (A) Original data set. (B) Color of certain links is modified using rules. (C) Geometry of nearby intra-chromosomal links has been adjusted to point the link outwards. (D) Rules were used to change the thickness of links.

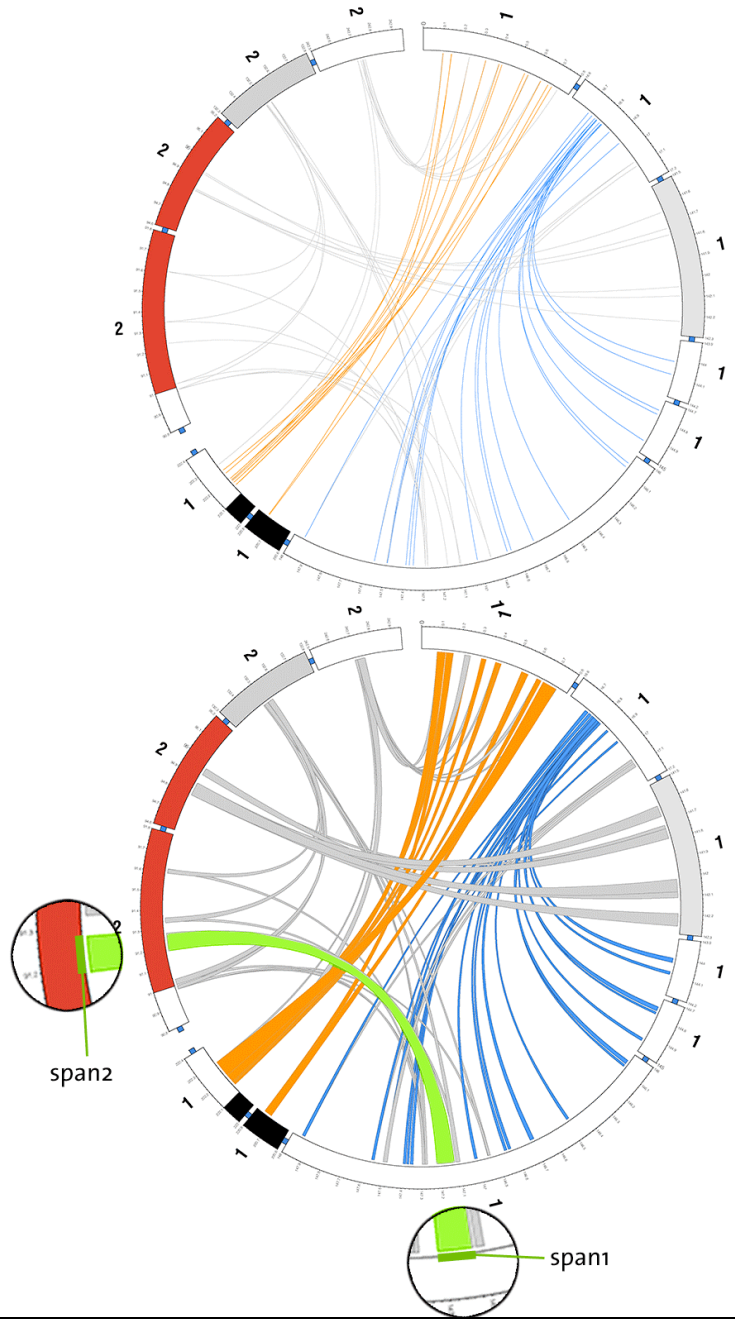

FIGURE 2

Links are defined by two regions, which can be of any size. *top* By default, links are drawn as lines (with adjustable, but constant, thickness). The lines start and end in the middle of the regions that define the link. *bottom* When the regions that define the link are large, it is helpful to use the thickness of the link to reflect the region size. To do this, links can be drawn as ribbons whose ends take on the thickness of the regions that define the link. When links are drawn as ribbons, thickness is not necessarily constant across the link. Depending on the orientation of the start and end regions, and the relative orientation of the scales of the ideograms that the link connects, ribbons can twist. This twisting can be explicitly controlled (e.g. all ribbons can be made flat, regardless of orientation of scale and link regions).

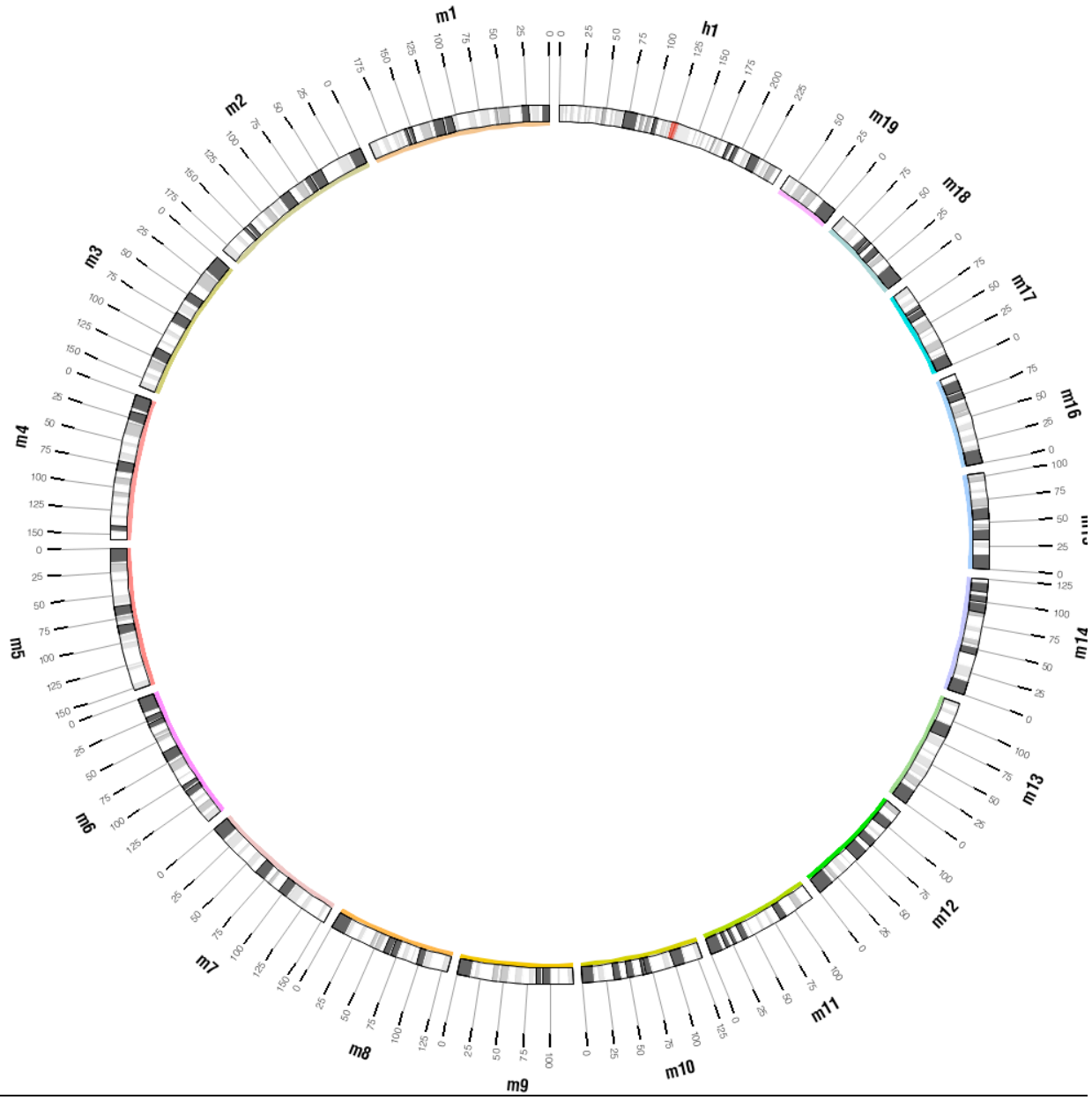

FIGURE 3

The figure template showing mouse chromosomes 1-19 and human chromosome 1.

We will be rescaling the human chromosome so that it occupies ½ of the figure in order to reveal detail.

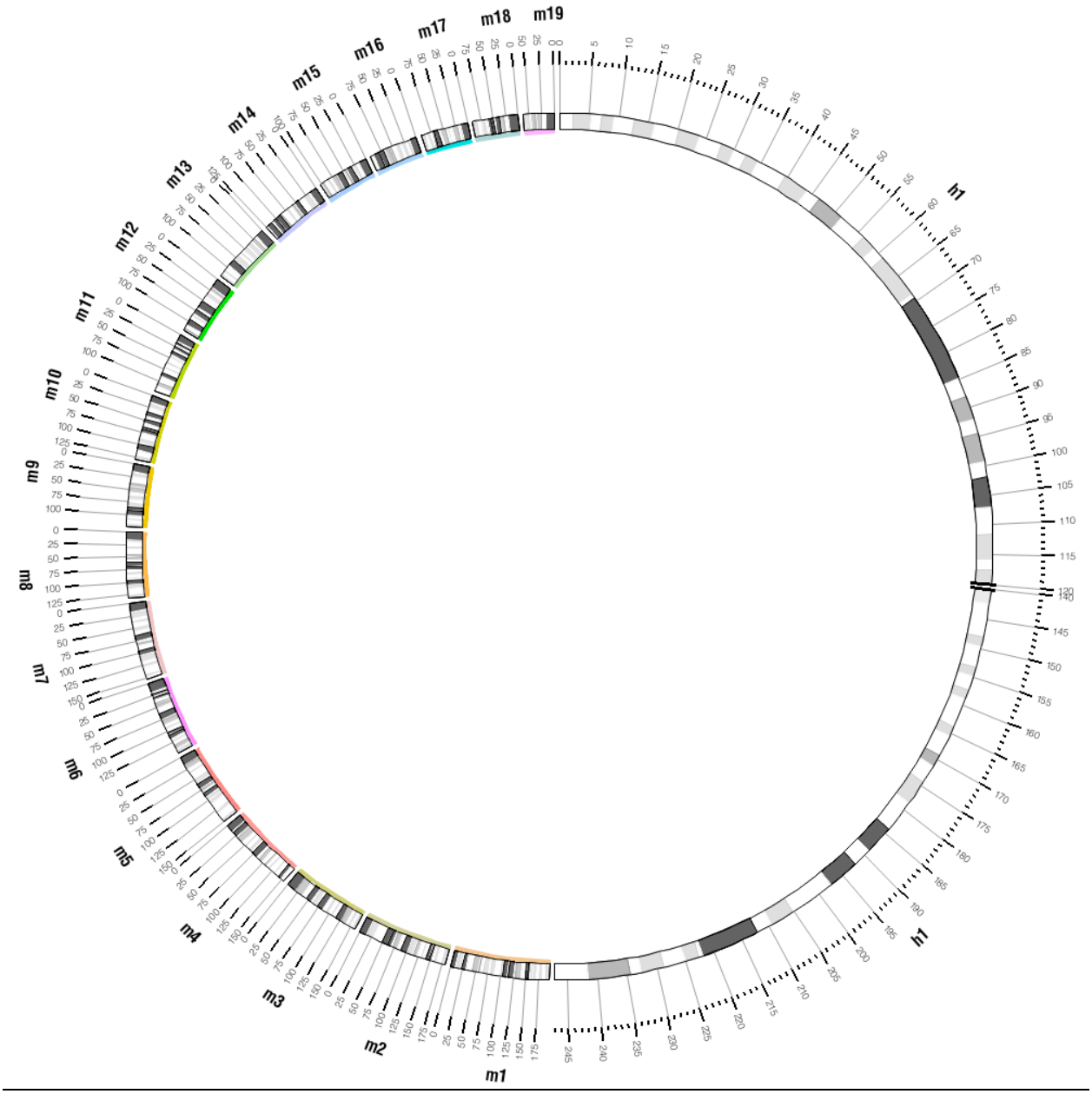

FIGURE 4

Mouse chromosomes 1-19 occupy ½ of the figure and human chromosome 1 is shown in the other ½.

The human chromosome has an axis break at 120-140Mb to remove the centromere from the display (there is no data for this region).

Notice that the scale of mouse chromosomes runs counter-clockwise.

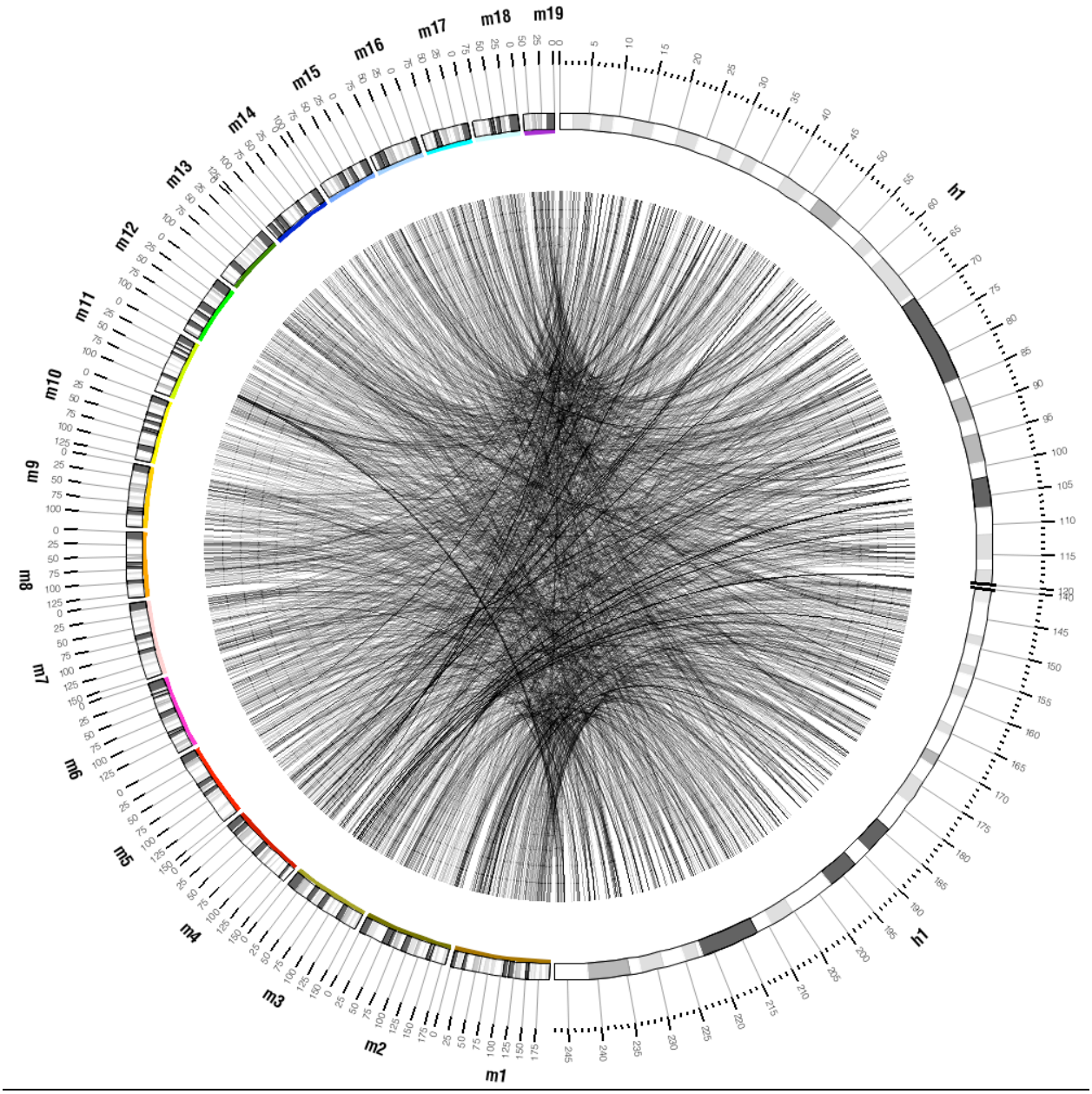

FIGURE 5

The links show 2,300 top alignments between human chromosome 1 and mouse chromosomes 1-19.

When transparency is used for link lines, it is possible to discern regions where the links are denser. The color for each link line here is `black_a5`.

Compare this figure with Figure 6, where transparency was not used.

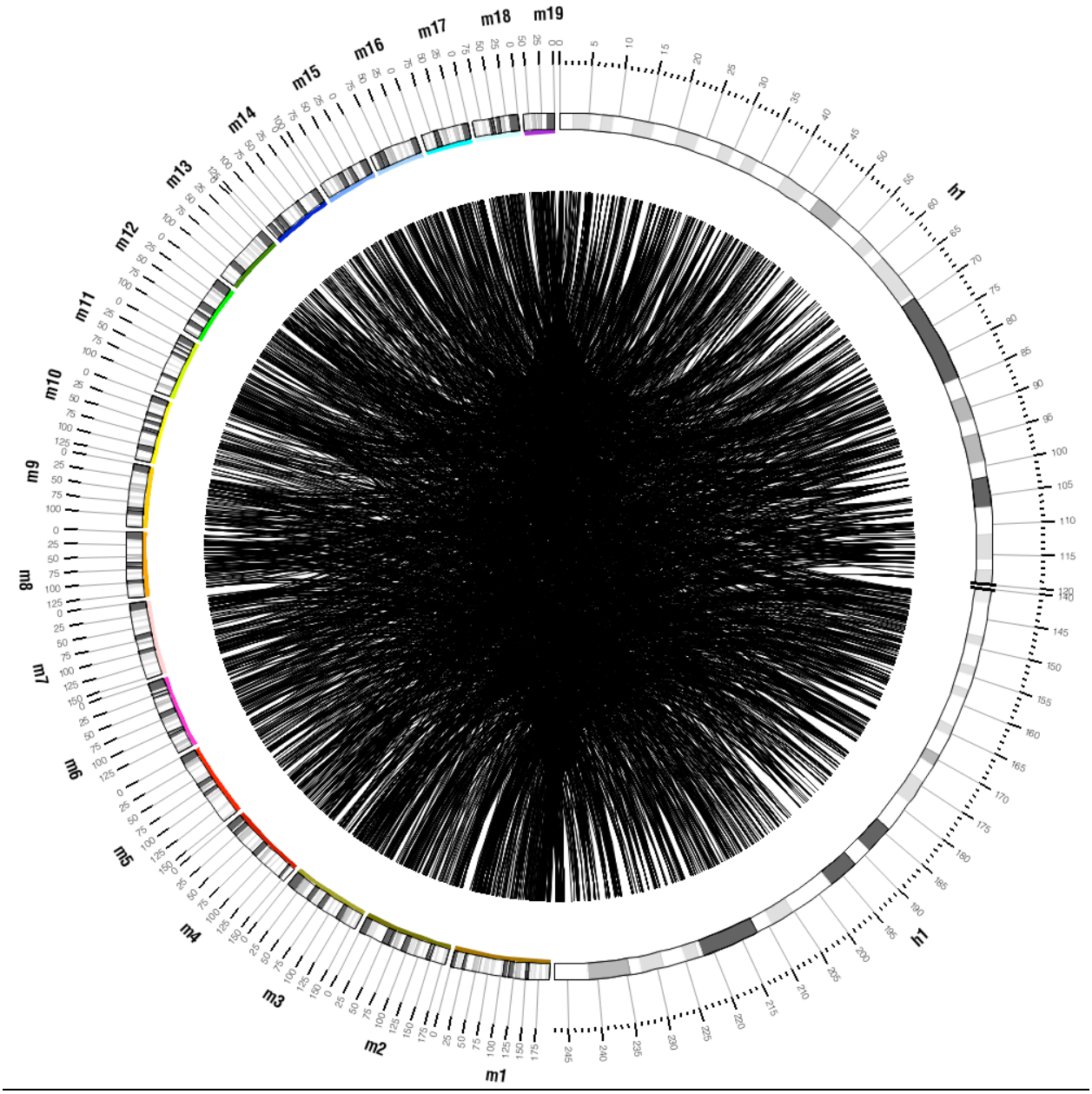

FIGURE 6

The links show 2,300 top alignments between human chromosome 1 and mouse chromosomes 1-19.

When transparency is not used for link lines, dense links form a solid shape making it impossible to discern regions where the links are denser. The color for each link line here is `black` (note, no `_aN` suffix).

Compare this figure with Figure 5, where transparency was used.

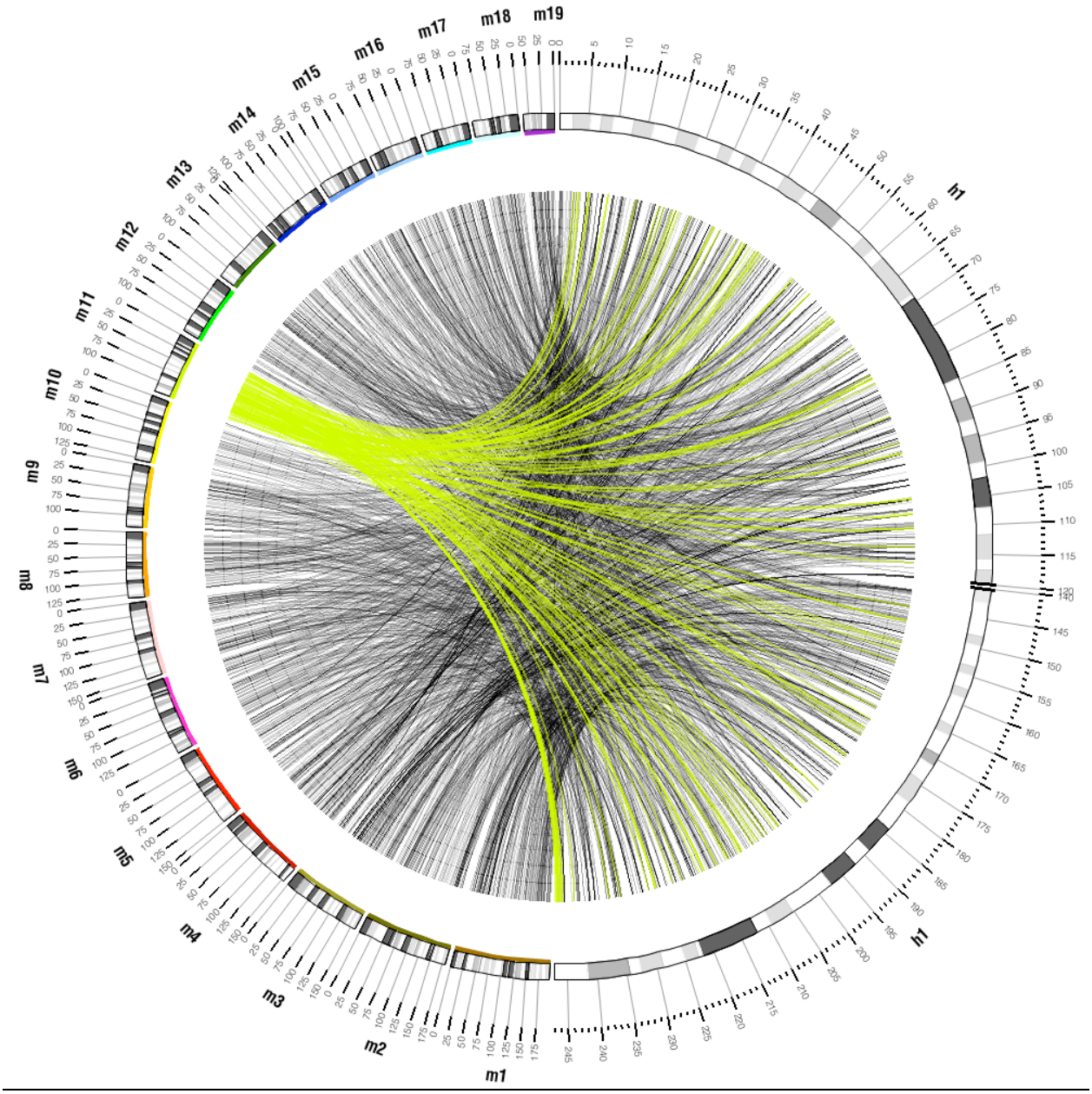

FIGURE 7

Rules are used to color all links that impinge on mouse chromosome 11.

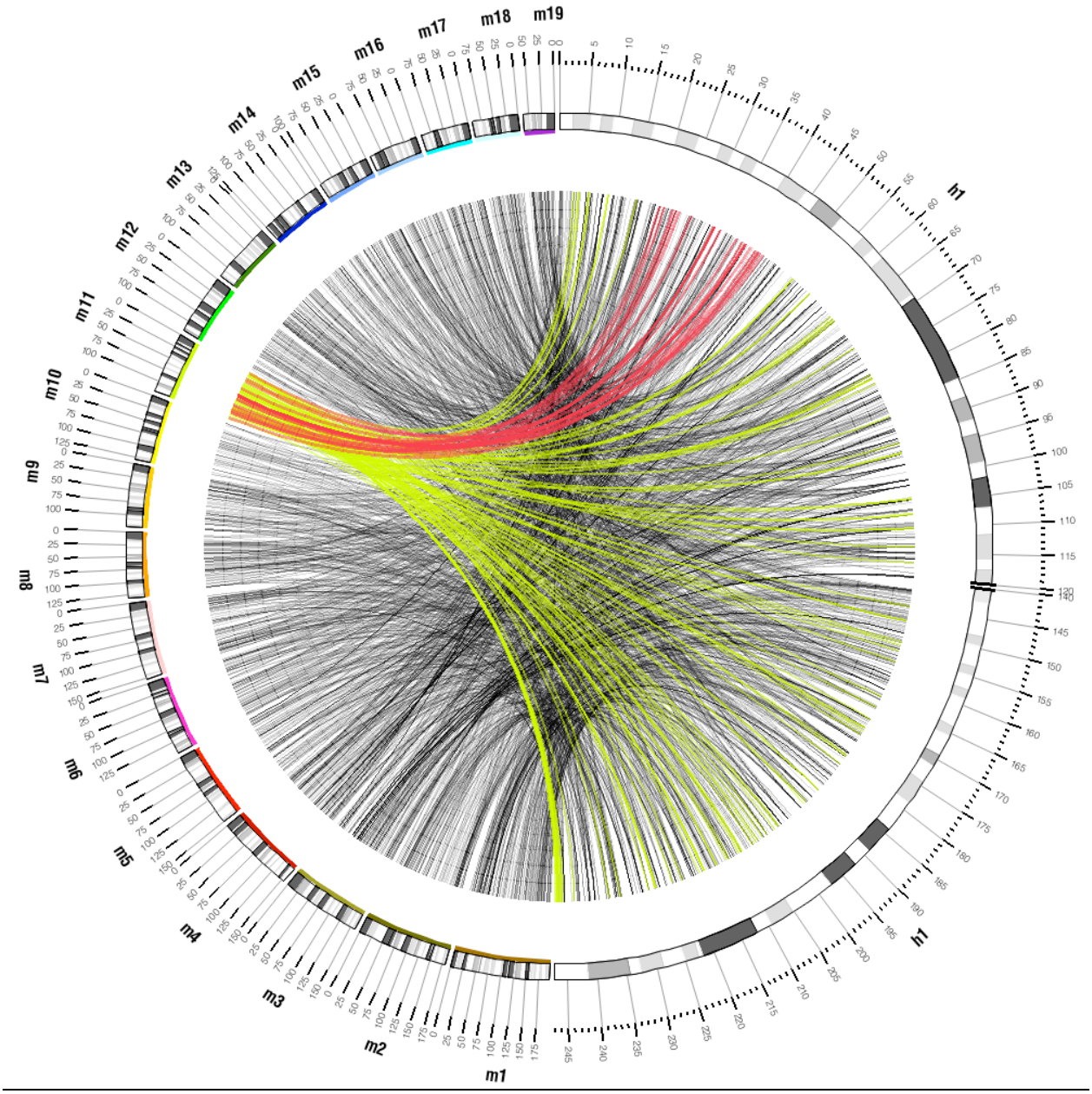

FIGURE 8

A second rule is added to uniquely color all mm11 links that start at 20-50Mb of hs1.

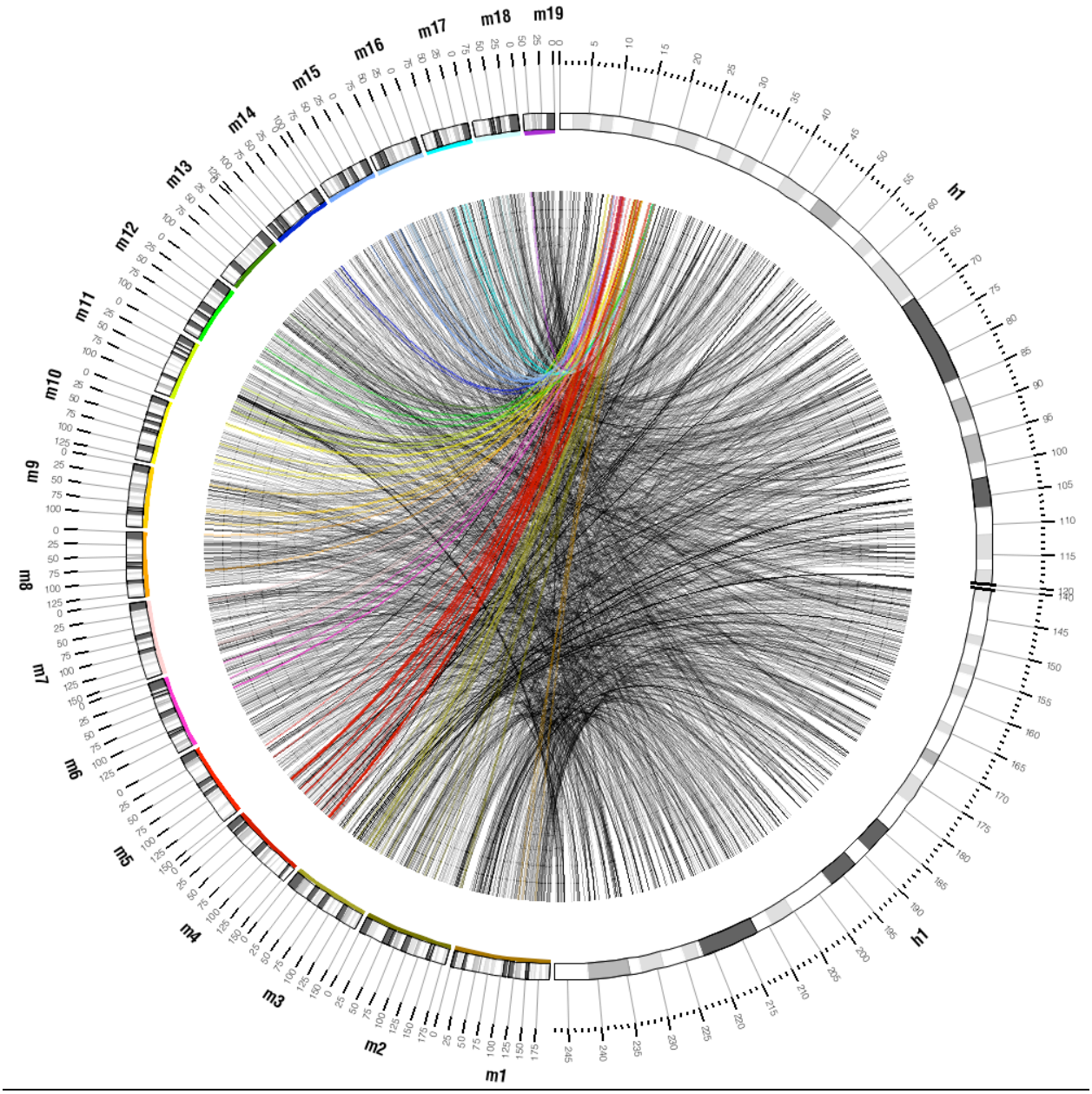

FIGURE 9

Links within 10-20Mb on hs1 are colored by their destination chromosome.

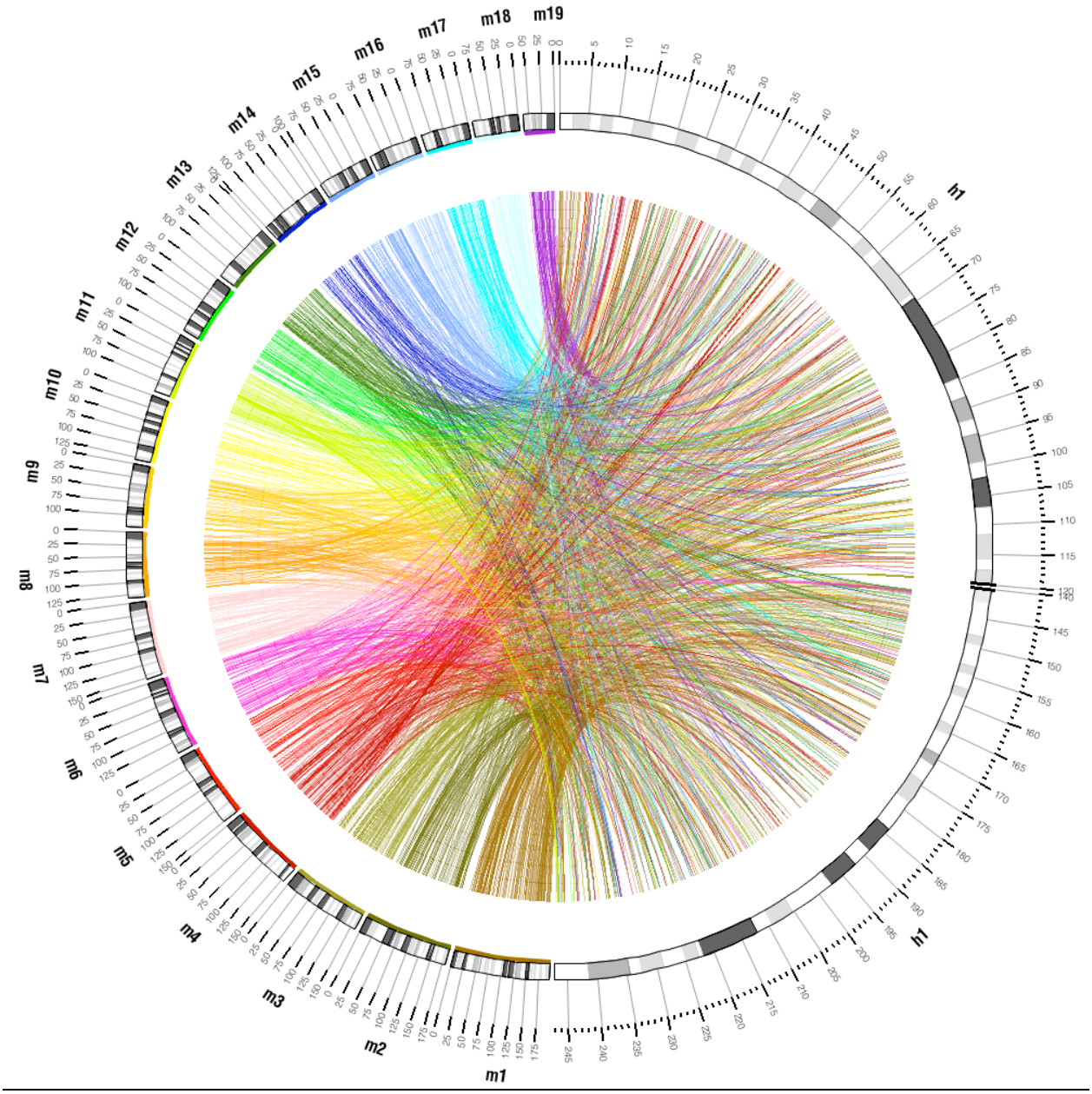

FIGURE 10

Using a rule, all links are colored by the chromosome associated with their ends.

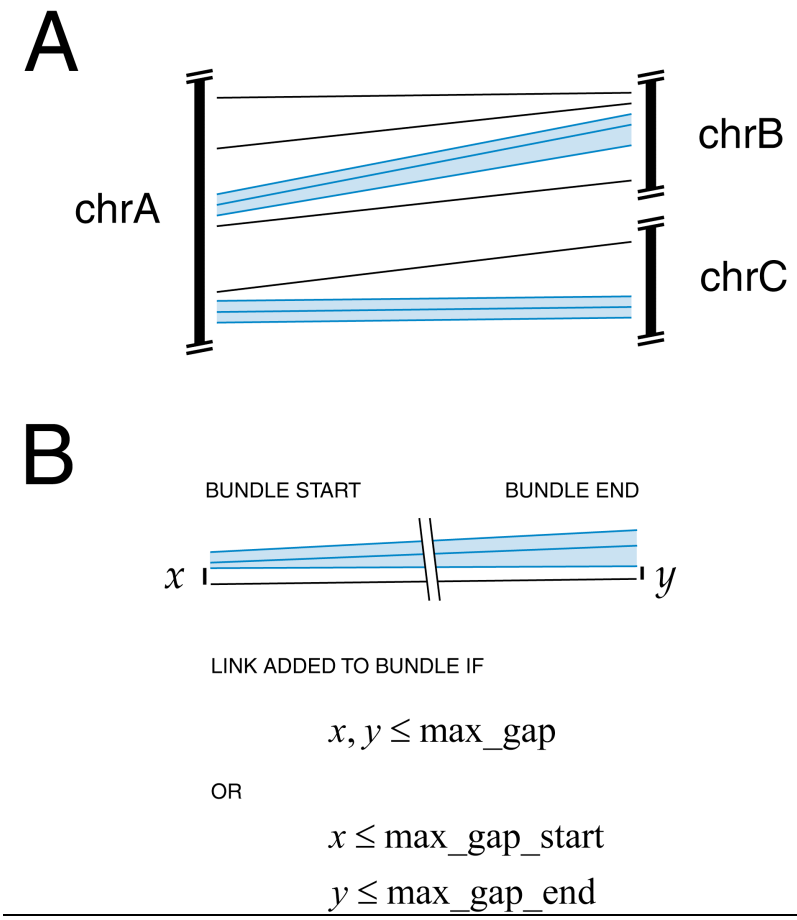

FIGURE 11

The `bundlelinks` tools is used to logically group adjacent links together, forming larger links. Links are bundled based on their size and distance to each other.

Bundles are ideally drawn as ribbons, rather than lines, because bundle ends typically span a significant section of an ideogram.

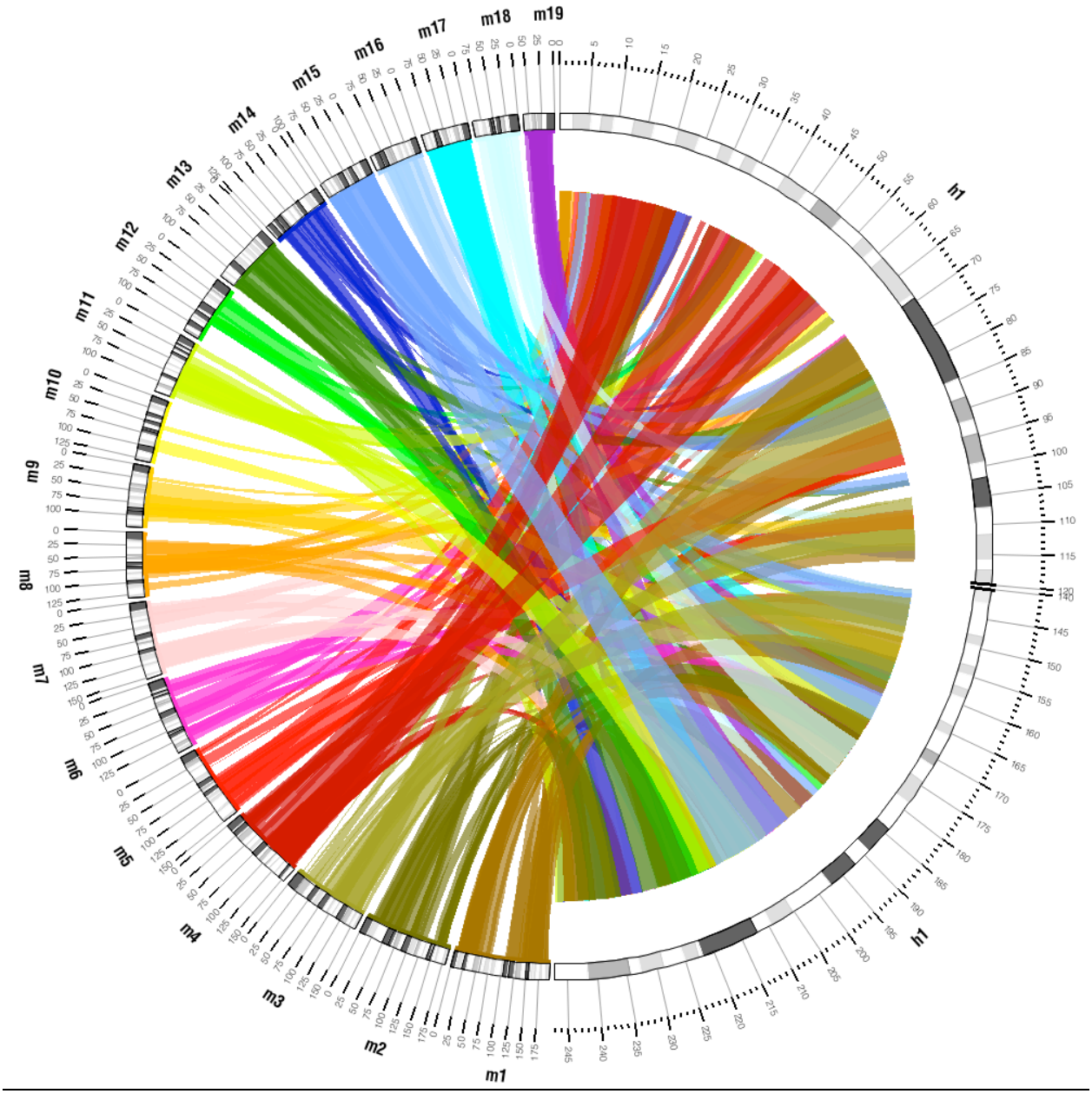

FIGURE 12

The result of bundling links shown in Figure 10.

Using `radius2`, the ends of the links are drawn closer to the mouse chromosomes.

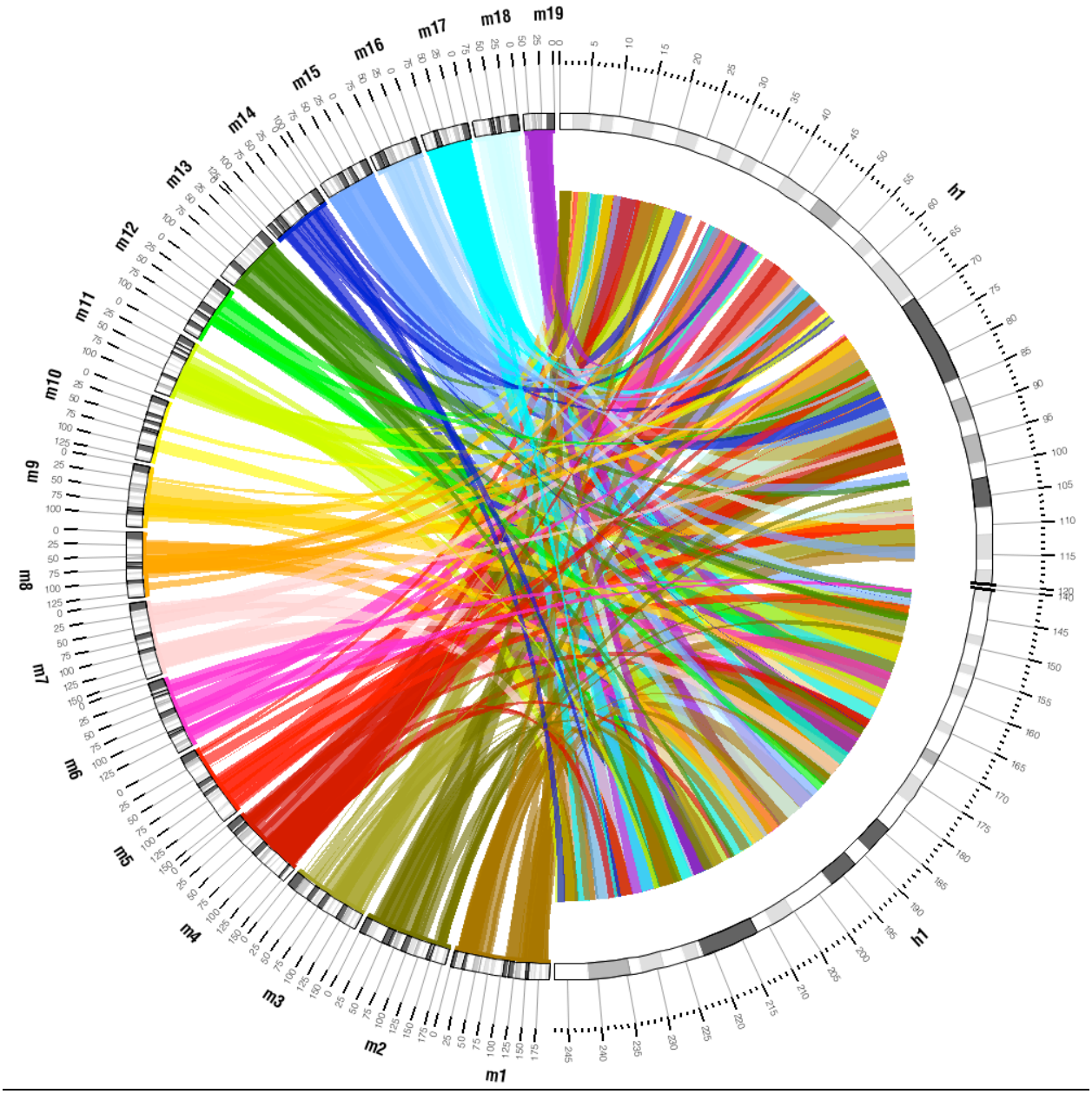

FIGURE 13

By setting the z value to be inversely proportional to link size, small links are drawn on top.

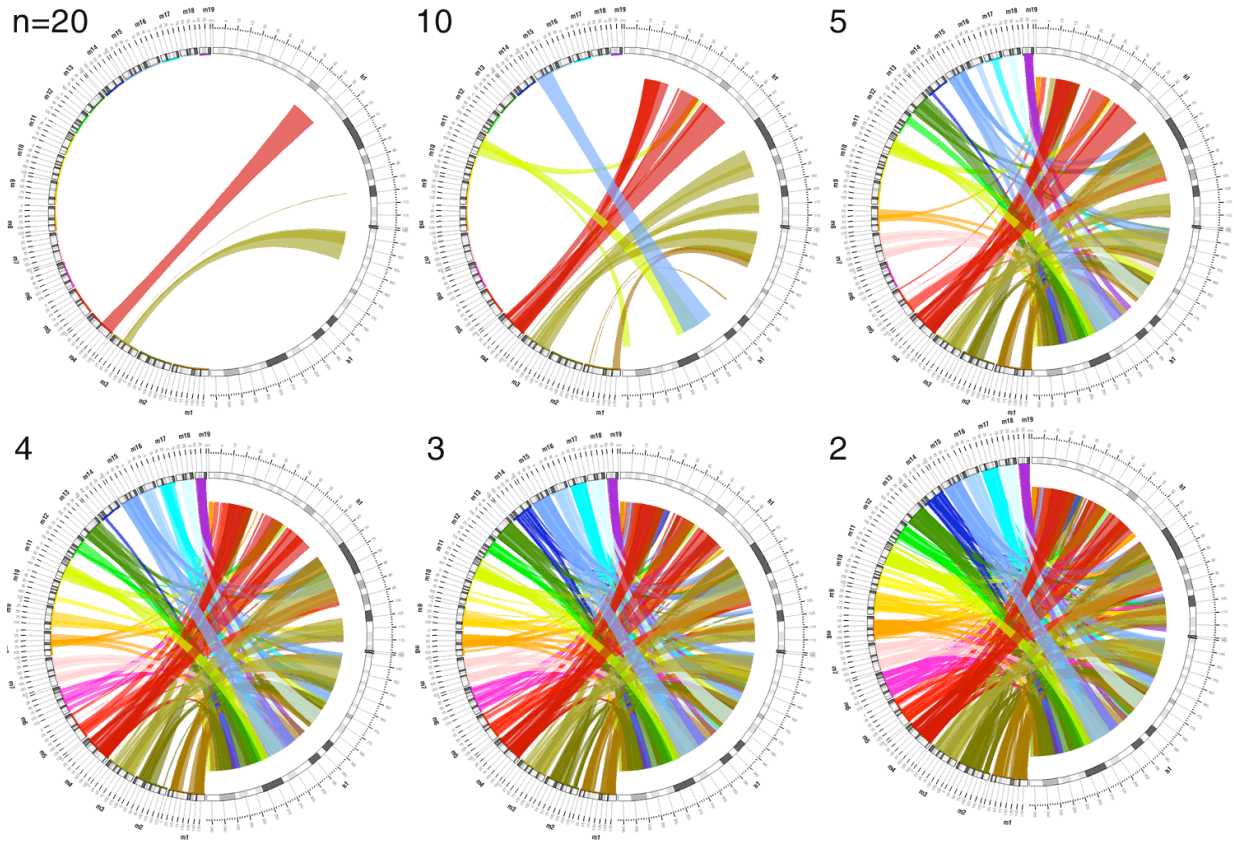

FIGURE 14

Varying the minimum number of links per bundle changes the sensitivity of bundling.

When a large number of links is required (e.g. n=20), only those regions that are connected by a large number of links are turned into bundles. When this number is decreased (e.g. `n = 10, n = 5, ...`), the number of bundles increases.

If the cutoff is small (e.g. `n = 2, 3`), it is possible to create a large number of bundles, because fewer links are required to form a bundle.

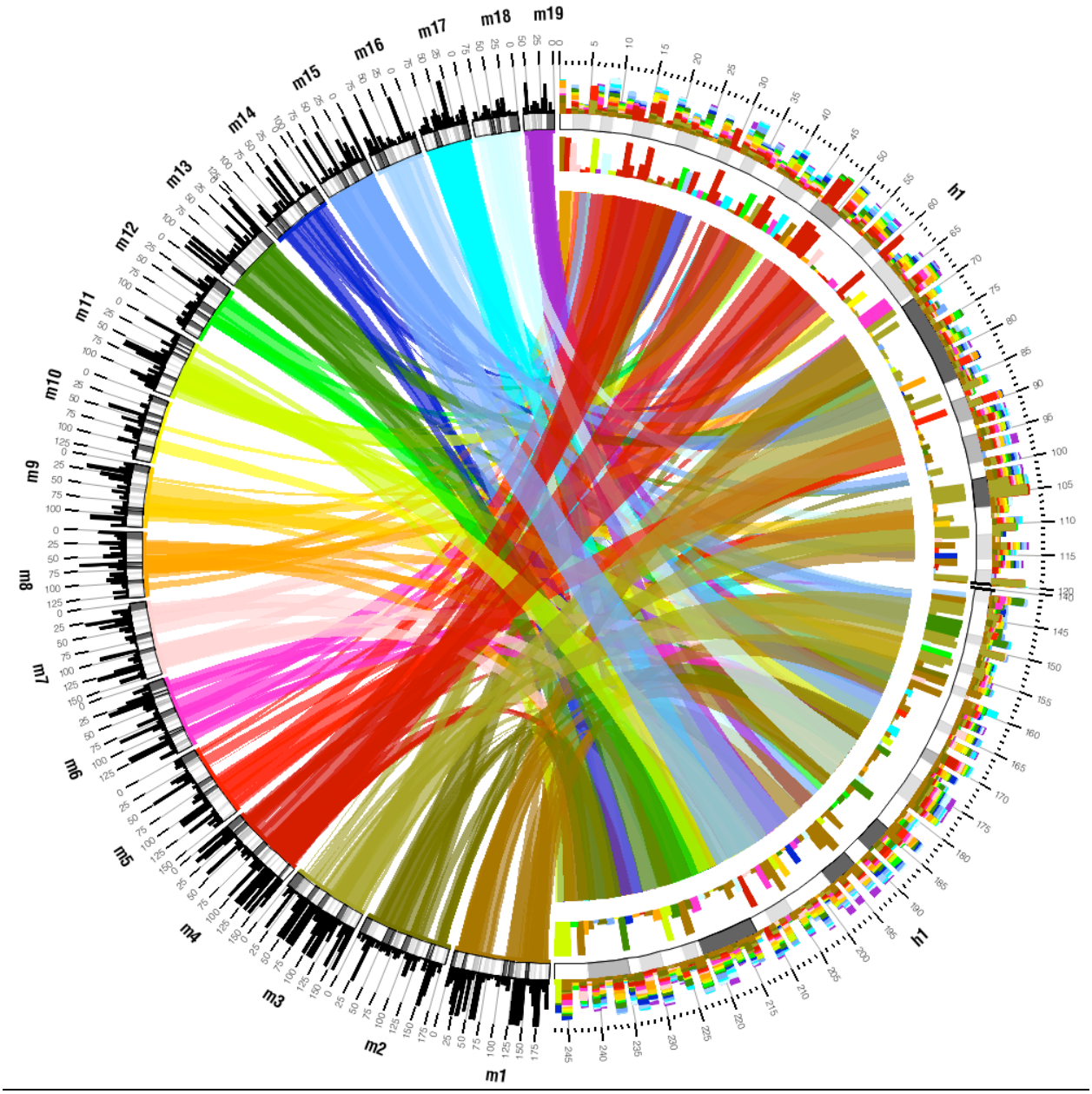

FIGURE 15

Three density histograms summarize information about the synteny between human chromosome 1 and the mouse genome.

The links in this figure are drawn as bundles, but the density histograms are calculated based on the individual links.

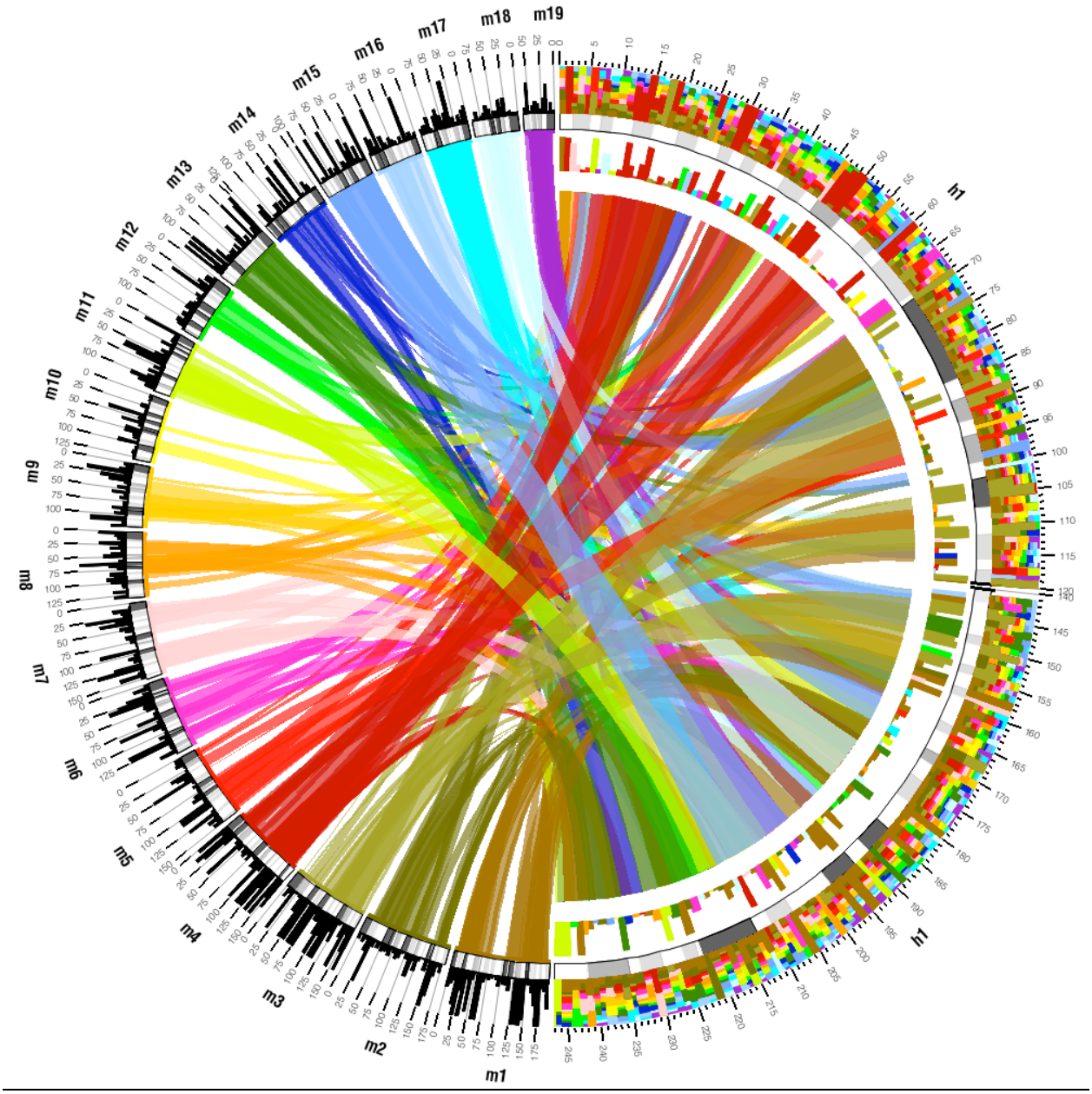

FIGURE 16

The outer stacked histogram is normalized – the total of each bin is 1.

Together with the inner density histogram placed on hs1, we see the absolute size of links between a window on hs1 and the mouse chromosome with greatest similarity, as well as the relative fraction of links between the window and each mouse chromosome.

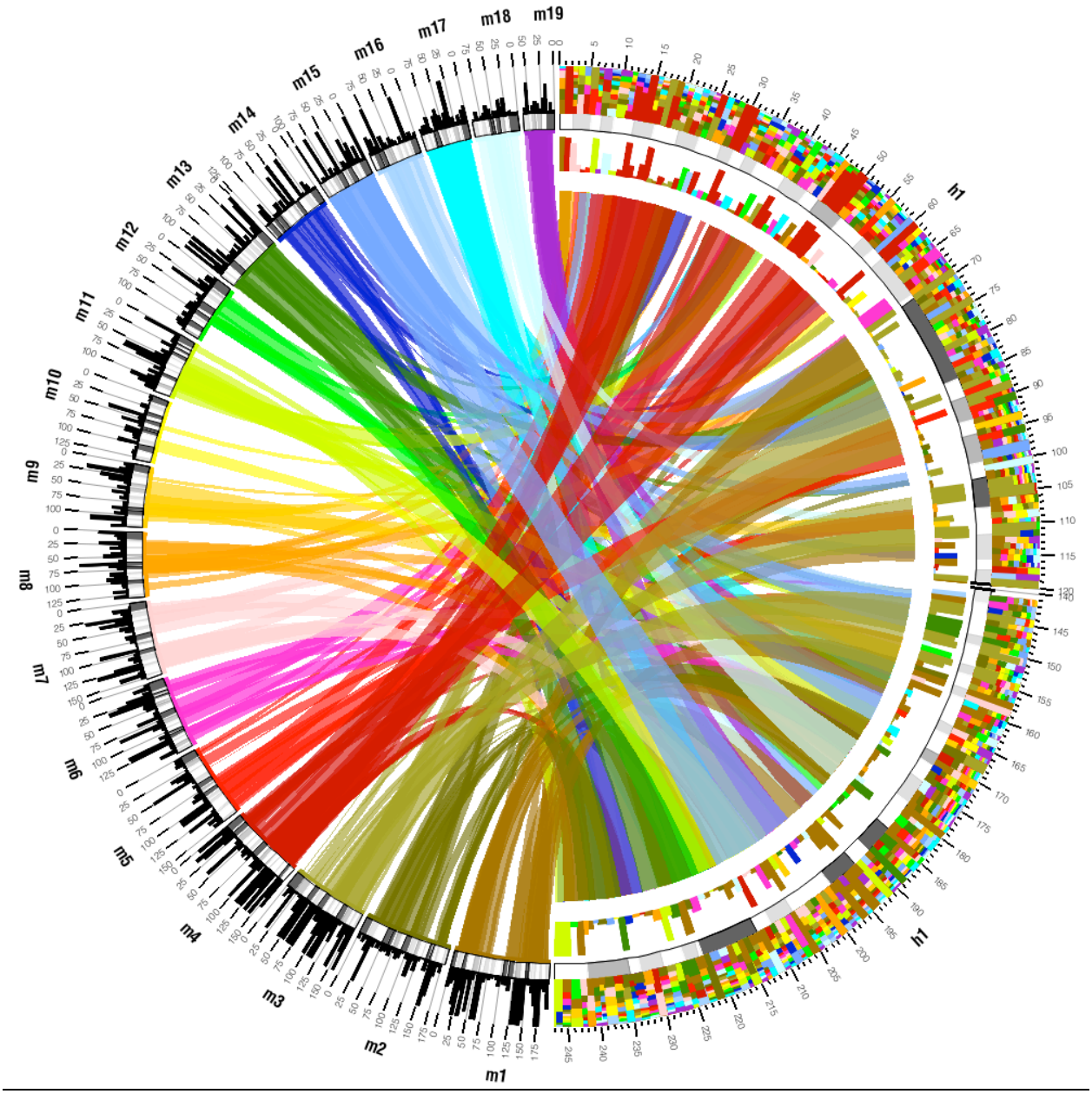

FIGURE 17

The normalized histogram is further modified by sorting the contributions to each bin by value.

The first contribution (bottom-most) is the largest and recapitulates the inner density histogram, which shows the same information but in absolute terms (and based on size of links, rather than number). The second contribution (second bottom-most) represents the next most similar chromosome, and so on.

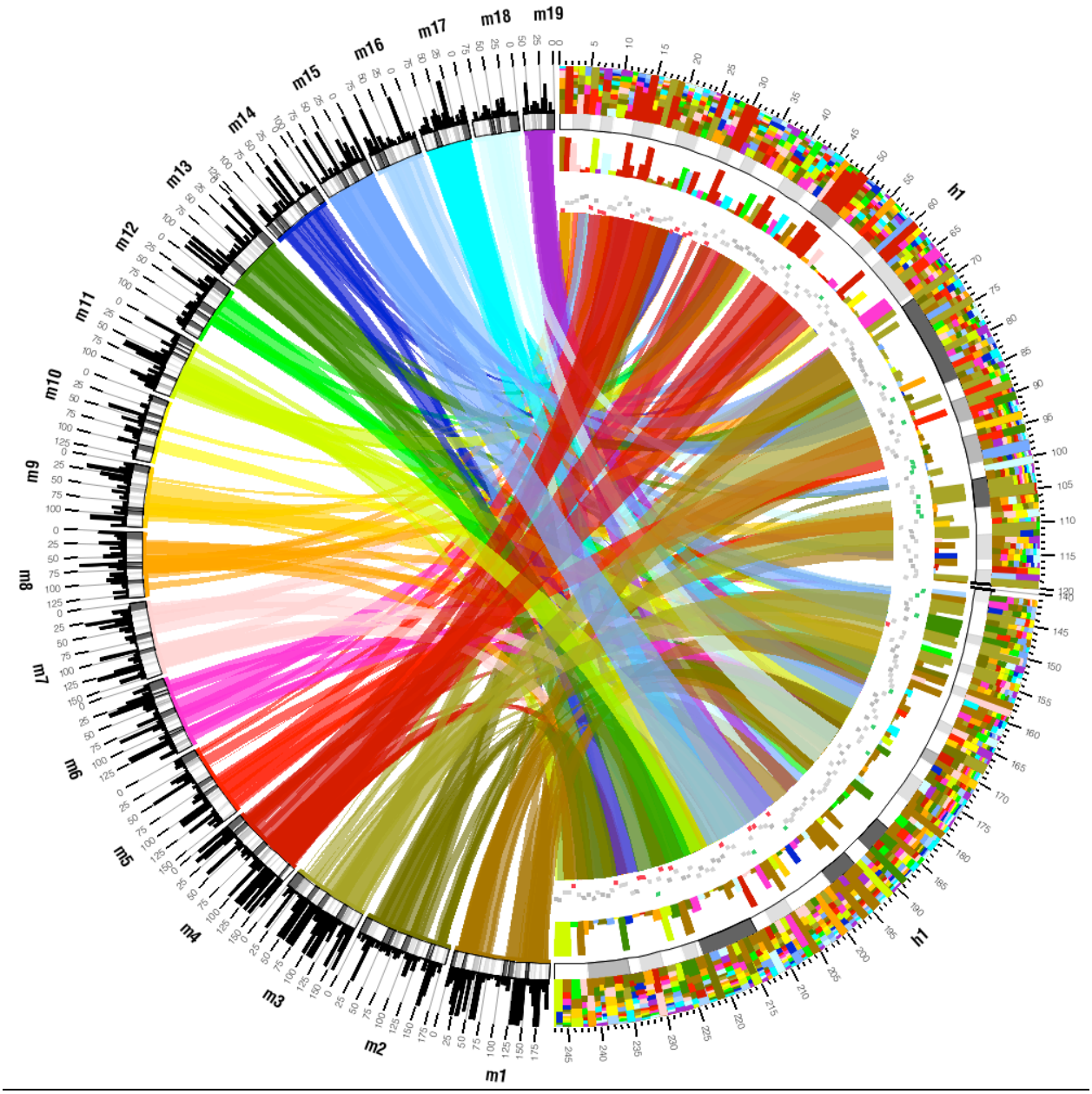

FIGURE 18

A scatter plot is added to the figure to show average conservation within 1Mb bins on hs1.

Rules are applied to the plot to color glyphs based on value.

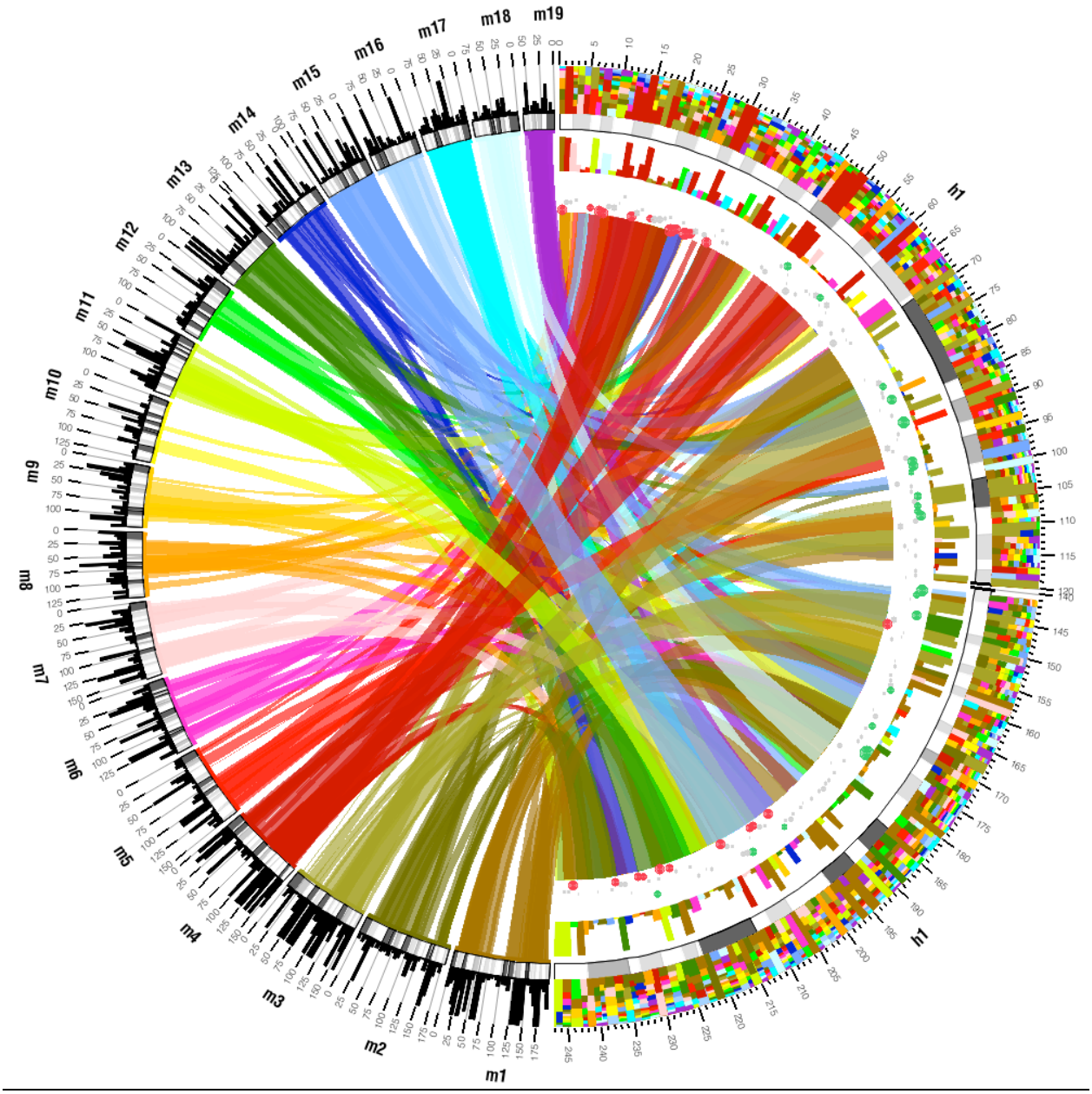

FIGURE 19

Glyph size is made proportional to the deviation of the data point (distance to average).

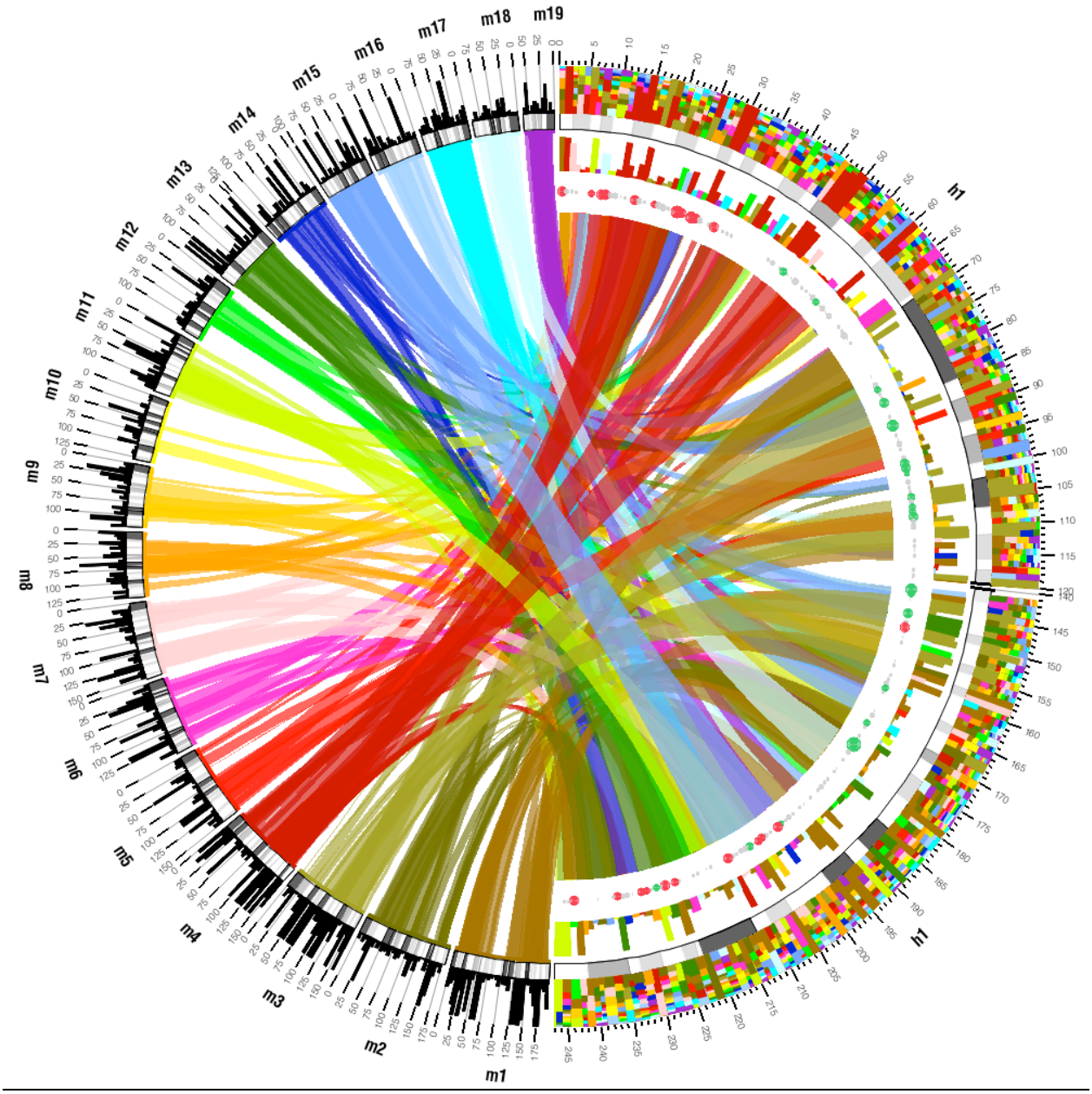

FIGURE 20

By mapping value onto glyph size and then placing all the glyphs at the same radial position (by changing data values), a glyph track is created.

Stacking such glyph tracks can create very interesting (and attractive) visualizations.

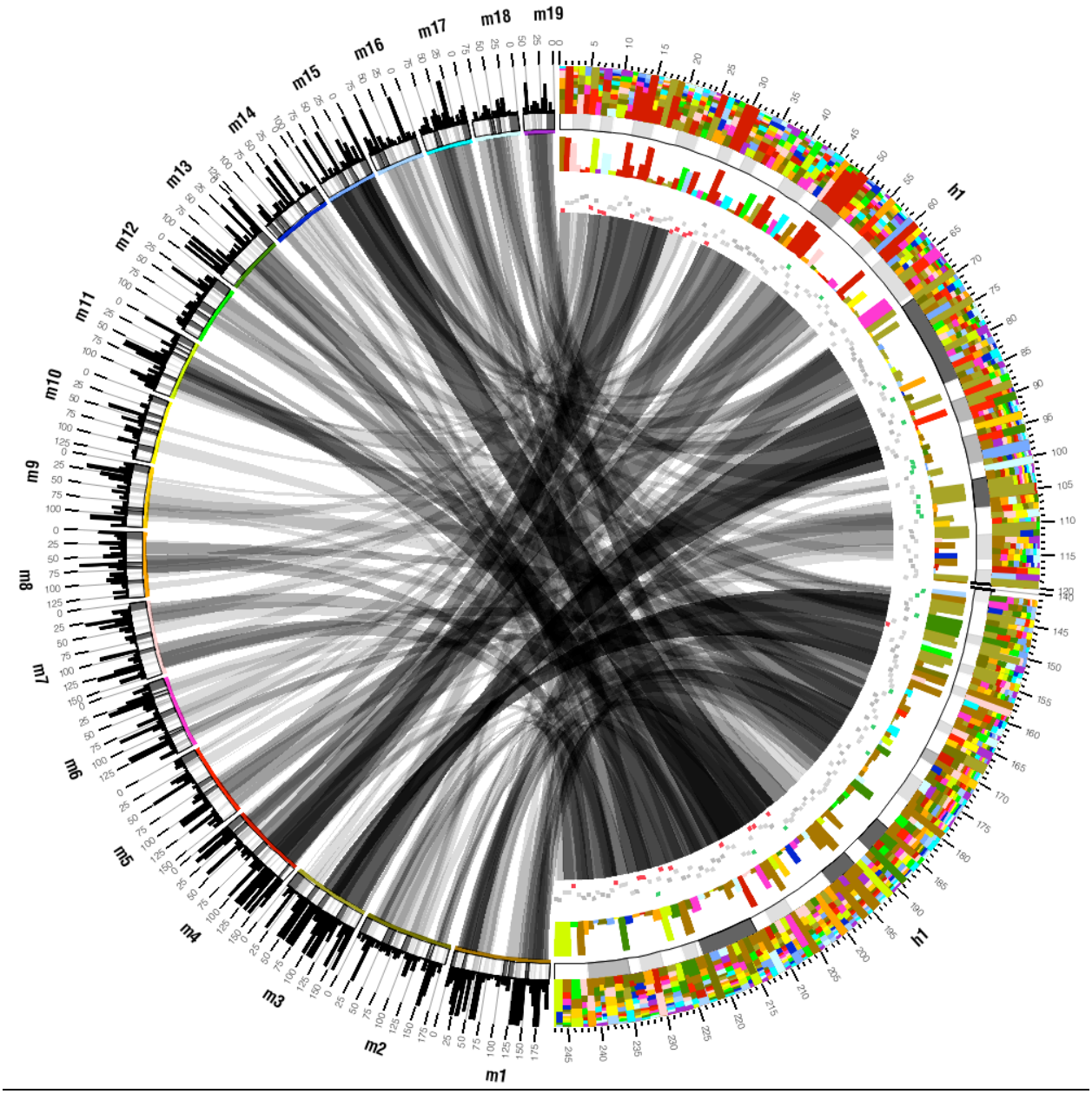

FIGURE 21

Bundles are shaded in proportion to their size on hs1.

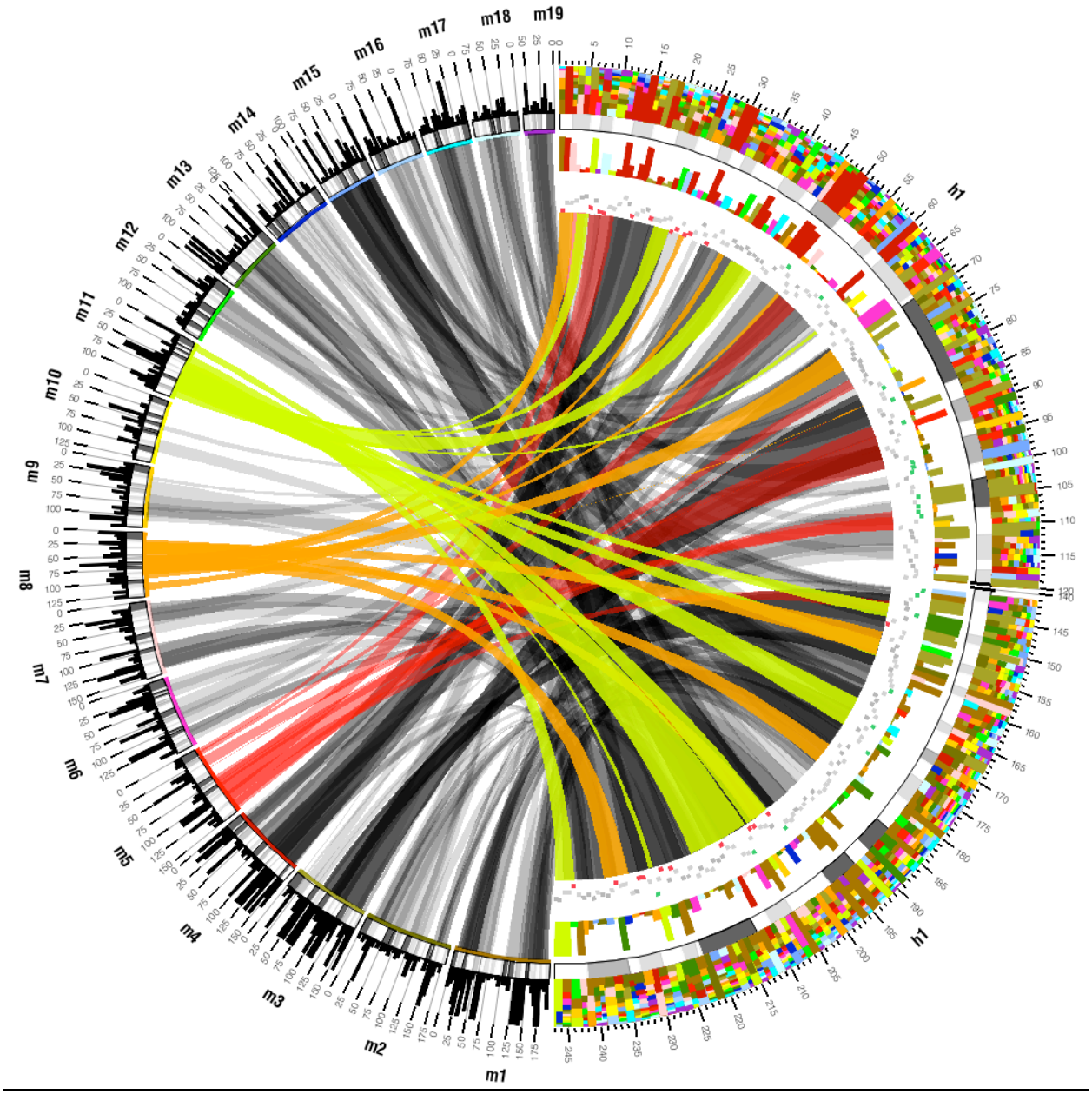

FIGURE 22

Rules help create three groups of links.

Links on mm8 and mm11 are drawn on top, in order of link size, and colored by mouse chromosome color. Links on mm5 are drawn next, with a subtler red tint. All other links are drawn below and shaded in proportion to their size.