Genome Visualization with Circos

Session 6 — Expression Data from a Single Sample

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
v0.24 7 Jun 2017
v0.23 3 May 2016
v0.22 17 Sep 2014
v0.21 12 May 2014
v0.20 6 May 2014
v0.19 21 Apr 2014
v0.18 7 May 2012
v0.17 5 Jun 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 29 Jun 2010
v0.10 15 Jun 2010

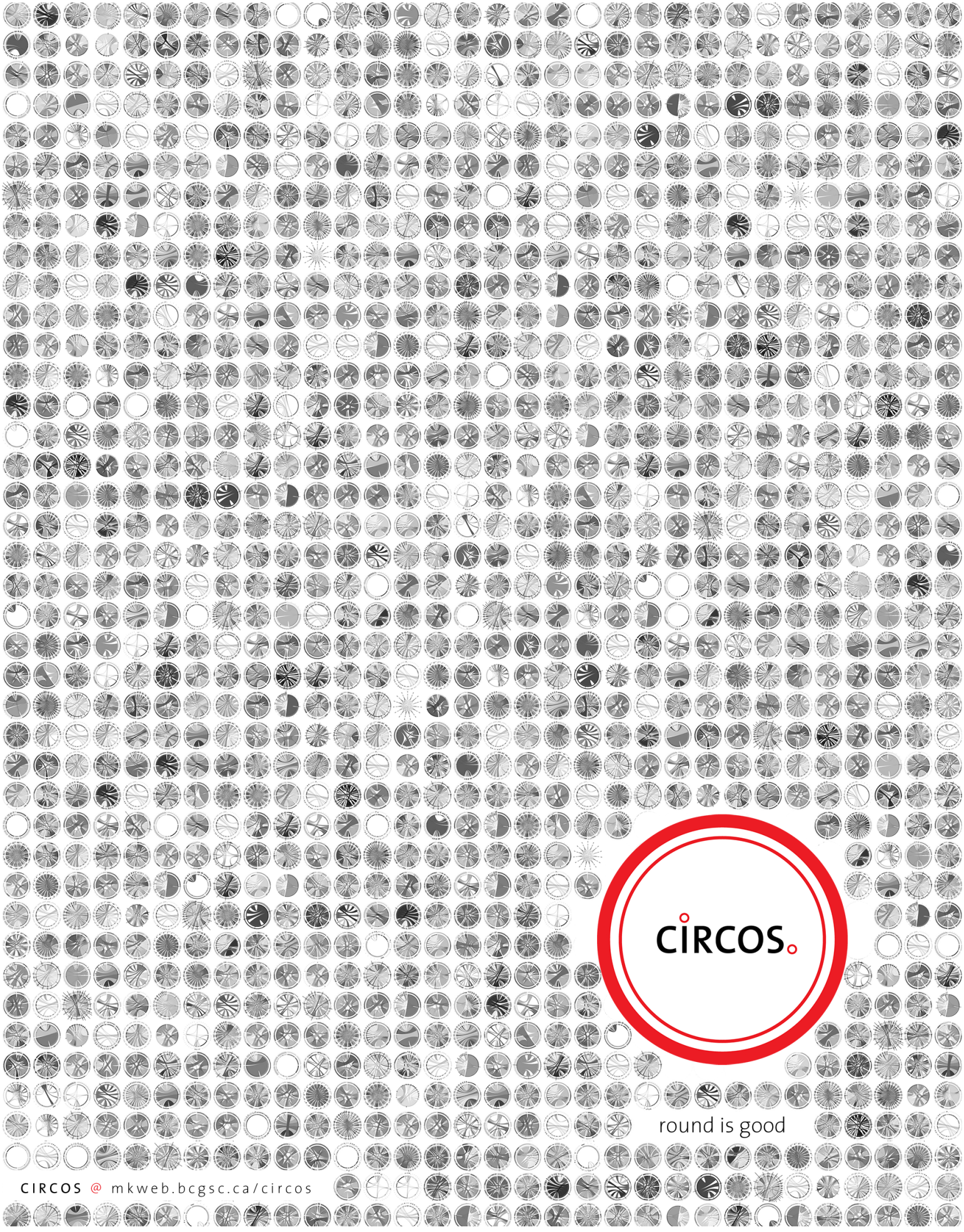

CIRCOS.
round is good
CIRCOS @ mkweb.bcgsc.ca/circos

Table of Contents

Introduction to Ideograms and Data Tracks

Central in a Circos figure are the *ideograms* and their order, scale and orientation. An ideogram is the graphical depiction of a chromosome. A chromosome may be shown as a single ideogram, in whole or cropped, or as multiple ideograms, if you divide it into several regions. It is important to create an ideogram layout that helps the reader parse and understand the data.

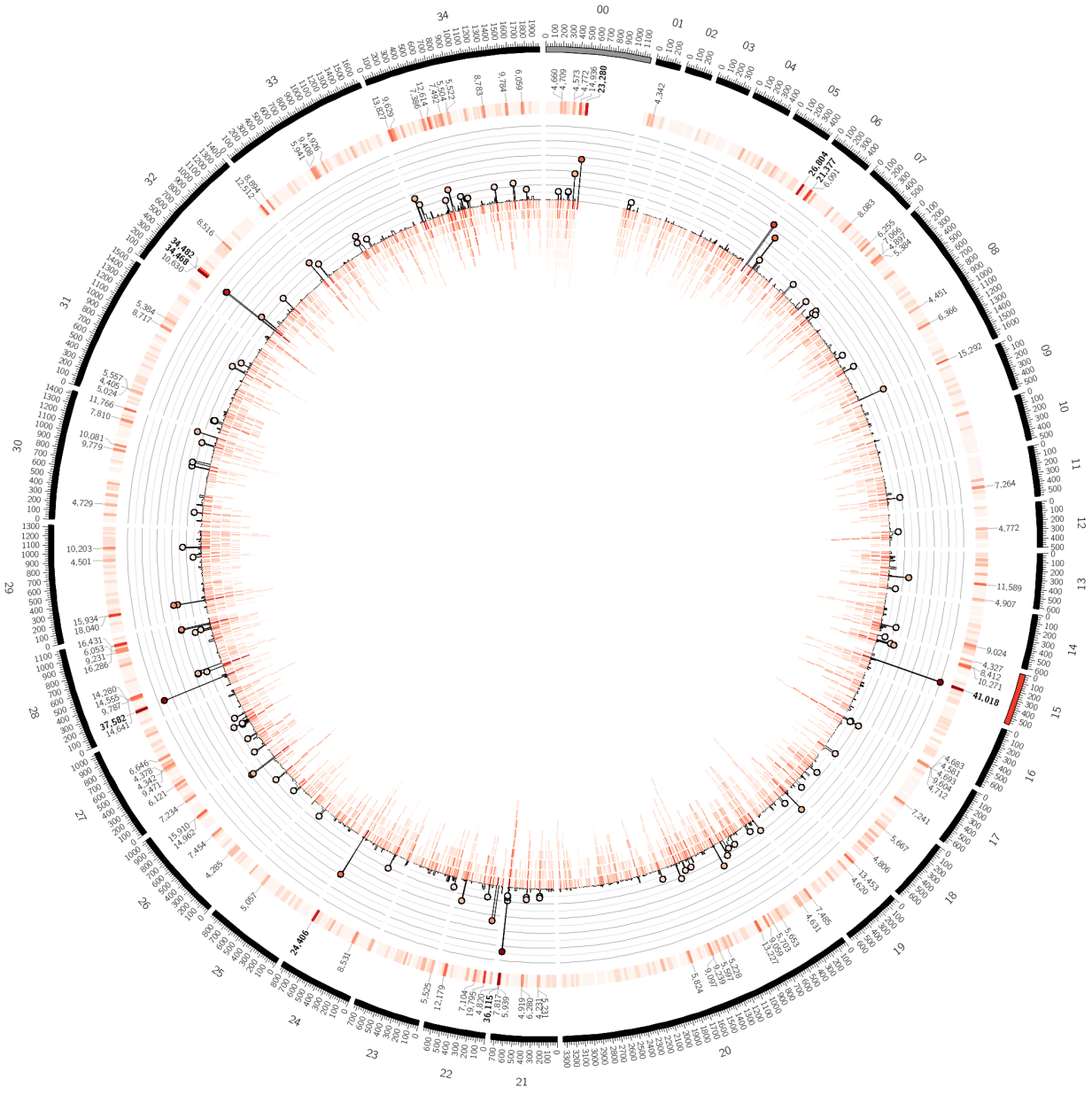


FIGURE 1
This is the image that we'll be working towards. It shows expression data from the ah063 sample for *Leishmania mexicana*.

For example, if you are comparing a single chromosome of one genome (e.g. human chr1) to an entire mammalian genome (e.g. mouse), it is helpful to magnify the human chromosome (e.g. so that it takes 50% of the figure) to present its data at a higher resolution.

On the other hand, if you are comparing two genomes, or roughly equal size subsets of two genomes, it is helpful to scale the ideograms to have the same size to create a pleasing symmetrical layout. You can see an example of this in **Session 1** and **7**.

Switch to Session 6 Lesson 1 directory and follow along.

```
# or wherever the lesson directory can be found
> cd ~/circos-course/session/
```

Make the requested changes to the configuration file. Each time, create an image by running Circos.

```
> circos
> eog circos.png
# now edit files as required and recreate the image
> eog circos.png
# when done with the lesson, go to the next one
> cd ../2
```

IN CASE OF PANIC

Stay calm.

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

I encourage you to work with your neighbor.

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

Lesson 1 – Drawing Ideograms and Debugging

Let's take a look at the configuration file for this lesson. Subsequent lessons will build on this file, therefore it is important to understand all of its components.

```
# The order, name, label and size of each chromosome is defined in the
# karyotype file. Look at this karyotype file to understand the
# format.
#
#  chr - NAME LABEL START END COLOR
karyotype = ../data/lm.karyotype.txt

# Any parameter values with suffix "u" will be a multiple of this
# number (e.g. 1000 = 1kb). Useful in definitions of ticks and ranges.
chromosomes_units = 1000

# Position and size of ideograms
<<include ideogram.conf>>

# Ticks and tick labels
<<include ticks.conf>>

# Image size, location
<<include ../etc/image.conf>>

# General parameters like colors, fonts, patterns and system settings.
<<include etc/colors_fonts_patterns.conf>>
<<include etc/housekeeping.conf>>
```

DEFINING A KARYOTYPE

The karyotype files defines the name, size and color of chromosomes. Any subset of these chromosomes (or regions thereof) can be drawn. Once this file is defined for a specific genome, you don't need to edit it again (unless the chromosome size and number changes with a new assembly). Karyotypes for common organisms (human, mouse, rat, chimp, fly, etc.) are in `data/karyotype` in the Circos distribution.

```
# lm.karyotype.txt
chr - LmxM.00 00 0 1171052 black
chr - LmxM.01 01 0 273291 black
chr - LmxM.02 02 0 298030 black
...
chr - LmxM.32 32 0 1407111 black
chr - LmxM.33 33 0 1649823 black
chr - LmxM.34 34 0 1968239 black
```

I'm using the same chromosome names as in other lectures in the workshop (`LmxM.NN`). You can define the names of chromosomes to be whatever you want. For simplicity Circos uses a species acronym for its chromosomes. Human chromosomes are `hsNN` (homo sapiens), mouse mmNN (mus musculus), rat `rnNN` (rattus norvegicus).

By this convention the *L. mexicana* chromosomes would be named `lmNN`, which is shorter. However, *L. major* would conflict with this, so we'd need to use something like `lmexNN` and

`1majNN`. In general, it's good where possible to avoid capitalization and punctuation in chromosome names.

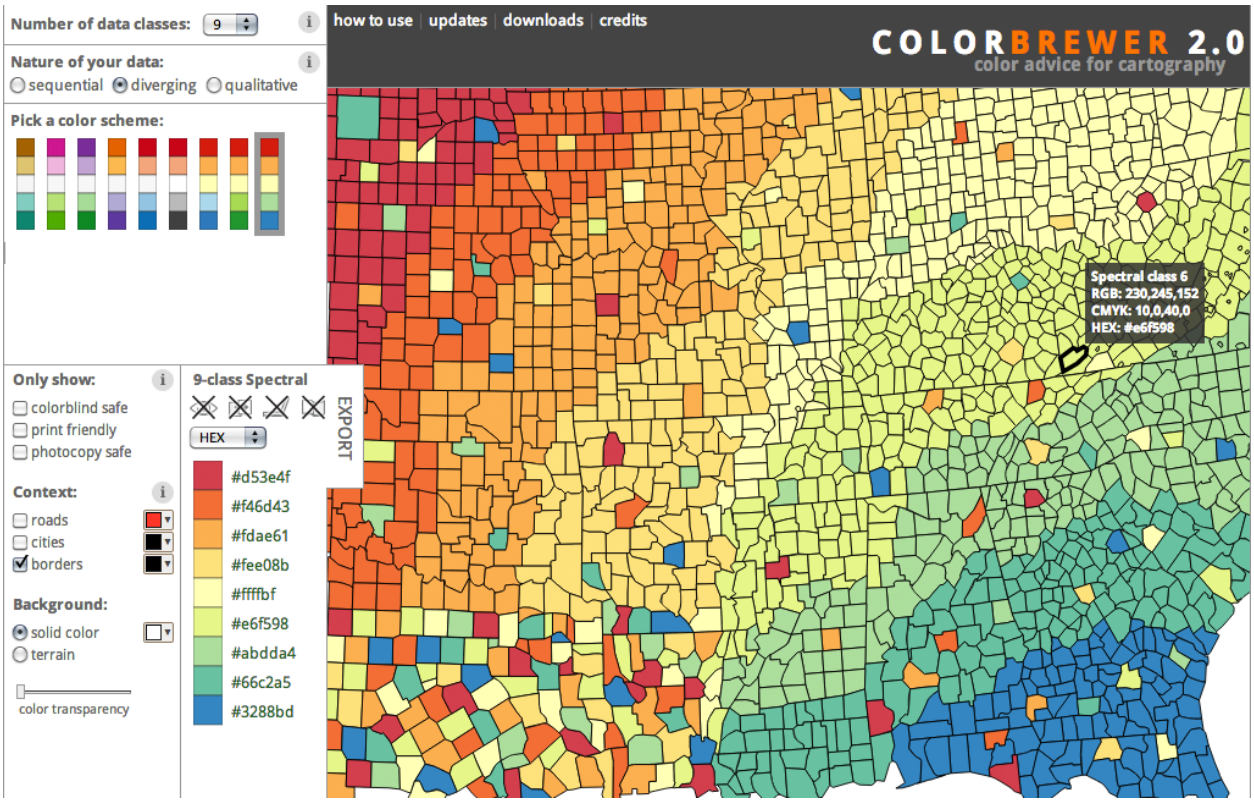


FIGURE 2
A 9-color spectral Brewer palette, as shown with Colorbrewer 2.0 (retrieved 6 May 2014). For more palettes, see *www.colorbrewer.org*.

Any colors in the karyotype file must be defined in the `<colors>` block in the configuration file (this block is defined in the `etc/colors_fonts_patterns.conf` file, which is imported). Colors are predefined for each hue (e.g. `vdred`, `dred`, `red`, `lred`, `vlred`, `v=very`, `d=dark`, `l=light`) as well as Brewer palettes (*http://www.colorbrewer.org*). The named red colors actually point to Brewer colors

```
vvlred  = reds-7-seq-1 # very very light red
vlred   = reds-7-seq-2 #      very light red
lred    = reds-7-seq-3 #           light red
red     = reds-7-seq-4 #                 red
dred    = reds-7-seq-5 #            dark red
vdred   = reds-7-seq-6 #       very dark red
vvdred  = reds-7-seq-7 #  very very dark red
```

For the *L. mexicana* chromosomes in this lesson we will use `black` for the color. We'll see how to change the color later.

GENERATING THE IMAGE

To generate the image, run Circos using this lessons configuration file.

```
> circos
debuggroup summary 0.16s welcome to circos v0.67-pre7 17 Sep 2014
debuggroup summary 0.16s guessing configuration file
debuggroup summary 0.16s found conf file
debuggroup summary 0.31s debug will appear for these features: summary
debuggroup summary 0.32s bitmap output image ./circos.png
debuggroup summary 0.32s parsing karyotype and organizing ideograms
debuggroup summary 0.34s applying global and local scaling
debuggroup summary 0.36s allocating image, colors and brushes
debuggroup summary 1.15s drawing highlights and ideograms
debuggroup summary,output 2.24s generating output
debuggroup summary,output 2.59s created PNG image ./circos.png (331 kb)
```

Circos will generate some output that describes what it is doing and the location of the output image. The image is shown in Figure 3.

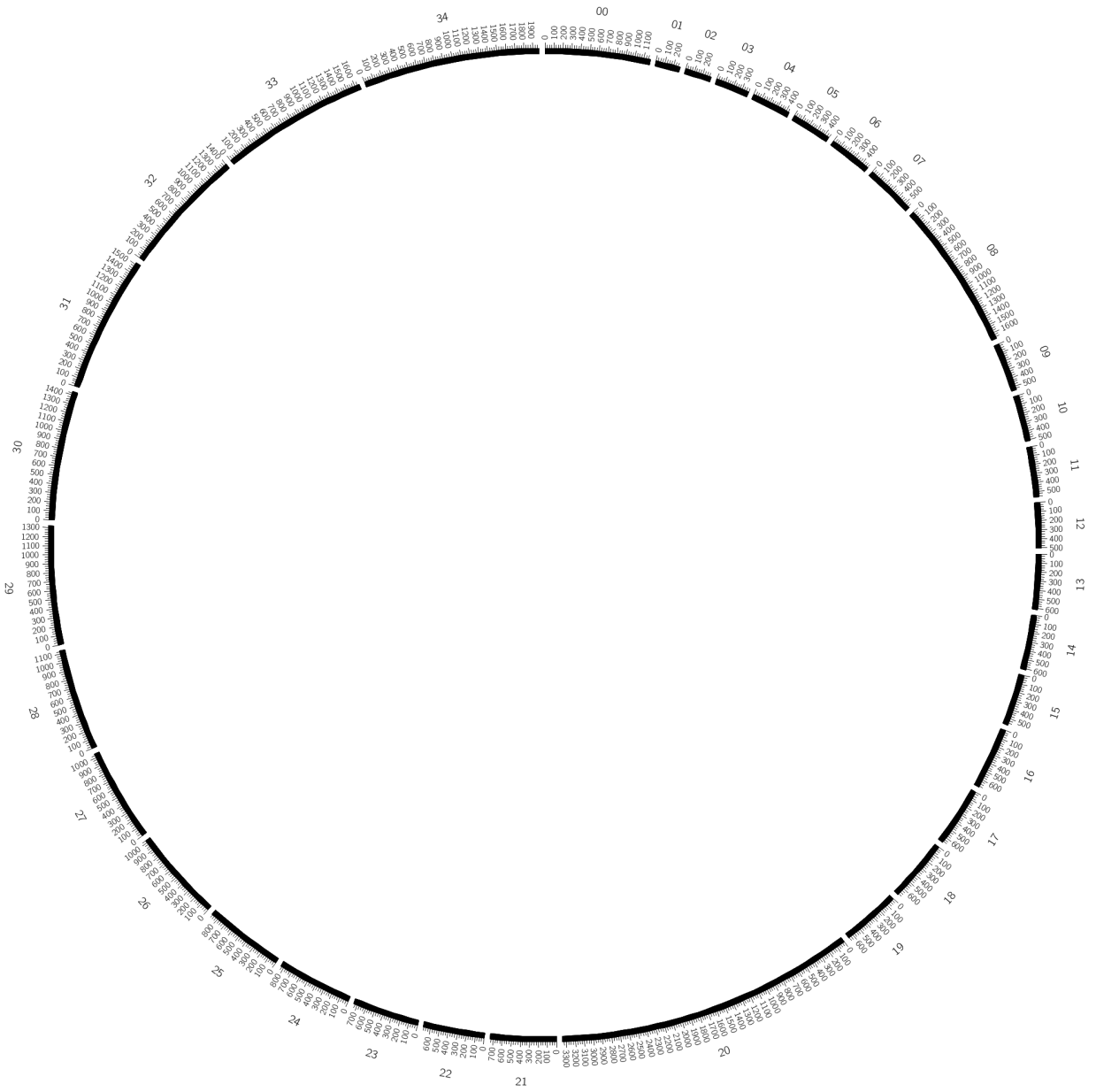


FIGURE 3
L. mexicana karyotype.

SPACING BETWEEN IDEOGRAMS

Spacing between ideograms is controlled in the `<ideogram>` block, by variables in the `<spacing>` block. The `<ideogram>` block is added to the `circos.conf` configuration file through the `<<include ideogram.conf>>` directive. The relevant parts of the `ideogram.conf` file for this lesson are

```
# 6/1/etc/ideogram.conf
<ideogram>
<spacing>
```

```
default = 0.002r

#default = 200u

#<pairwise LmxM.00 LmxM.34>
#spacing = 5r
#</pairwise>

#<pairwise LmxM.00>
#spacing = 5r
#</pairwise>

</spacing>
</ideogram>
```

In this example, the spacing between ideograms is `default = 0.002r`. The r suffix indicates that the units of this value are given relative to circumference of the ideograms. The u suffix is relative to the value defined by `chromosomes_units`. Since `chromosomes_units = 1000`, the spacing is `200×1000` (2 Mb).

Change the default spacing to 200u and observe the difference.

```
<spacing>
#default = 0.002u
default = 200u
```

SPACING BETWEEN SPECIFIC PAIRS OF IDEOGRAMS

Spacing between individual ideogram pairs can be adjusted using `<pairwise>` blocks within the `<spacing>` block. For each pair, a different `<pairwise>` block is defined, with the name of the block given by "`name1 name2`" where `name1` and `name2` are the names of the adjacent chromosomes. You can also use `name1,name2` as alternative syntax. If you specify only one ideogram in the `<pairwise>` block, spacing on either side of it will be changed.

```
# 6/1/etc/ideogram.conf
<pairwise LmxM.00 LmxM.34>
spacing = 5r
</pairwise>

#<pairwise LmxM.00>
#spacing = 5r
#</pairwise>
```

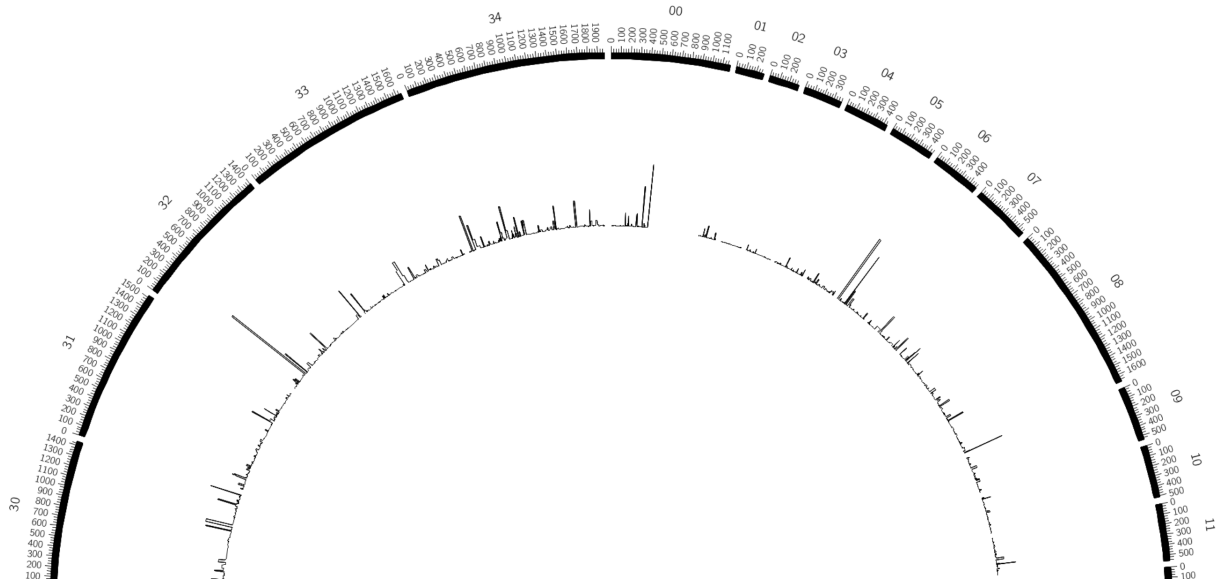
By uncommenting the first `<pairwise>` block, the space betweeen chromosomes 34 and 00 is changed to 5× the default spacing. If instead you use the second block, spacing on either side of chromosome 00 will be increased.

Lesson 2 – Histograms, Axis grids, Macros and Track Defaults

Let's start adding data to the figure.

First, we'll draw the expression values stored in `lm.exp.ah063.txt`

```
<plots>

<plot>
type      = histogram
r1        = 0.9r
r0        = 0.7r
file      = conf(datadir)/lm.exp.ah063.txt

</plot>

</plots>
```



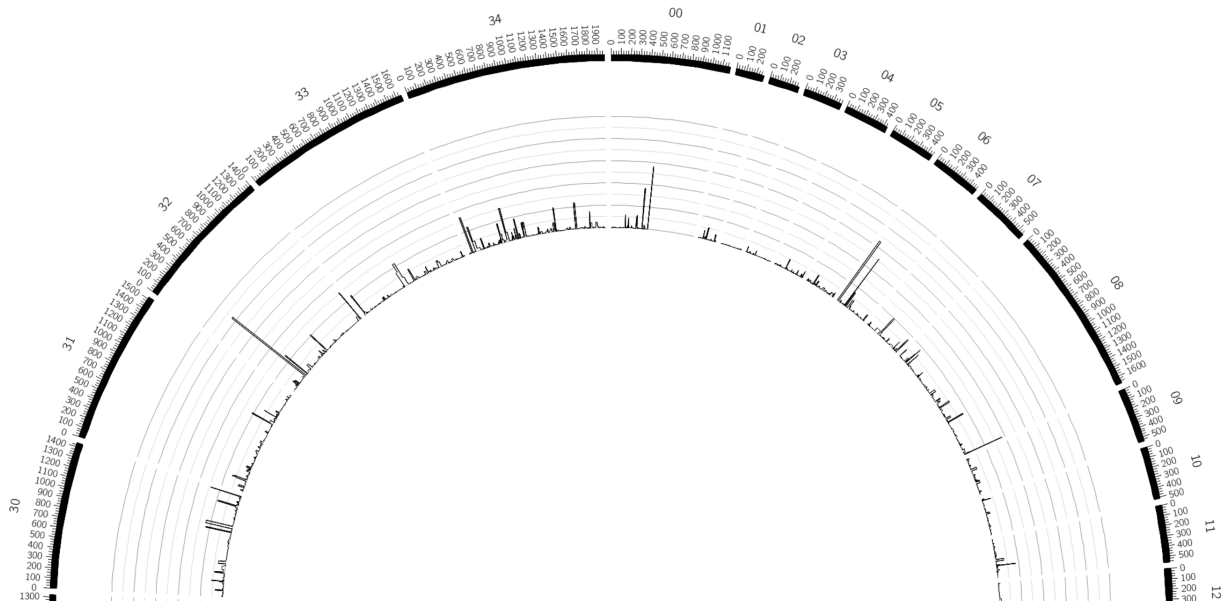
Look in the data file for the histogram.

```
> cat ../data/lm.exp.ah063.txt
LmxM.01 9205 11223 76 gene=LmxM.01.0030
LmxM.01 14846 16843 64 gene=LmxM.01.0050
LmxM.01 27727 28314 72 gene=LmxM.01.0110
...
```

What are the first four fields?

Every data point can have any number of parameters defined as `variable=value` pairs. Why might we want to define the `gene` parameter?

Turn on the axes by setting `show=yes` in the `<axis>` block.



You'll notice that the parameter named `datadir` is referenced throughout the configuration file using `conf(datadir)`.

This is something that you can do anywhere—define a parameter once and then refer to it. For example, `fill_color = blue` can be referenced using

```
stroke_color = conf(fill_color)
```

Let's see how this can be done for Leishmania. You'll see that the files reuse the `lm` string. First, define

```
species = lm
```

in the root of the configuration file. Then replace "lm" with conf(species). For example,

```
karyotype = conf(datadir)/conf(species).karyotype.txt
```

If you're ever uncertain about what the values in the configuration blocks are, use run Circos with the –cdump flag.

```
circos –cdump
```

to show the full configuration (long) or to show only a specific block (e.g. <plots>)

```
circos –cdump plots
```

As you plot other data tracks, you'll notice that many have default settings. For example, histograms are shown as black lines with a stroke thickness of 1 pixel.

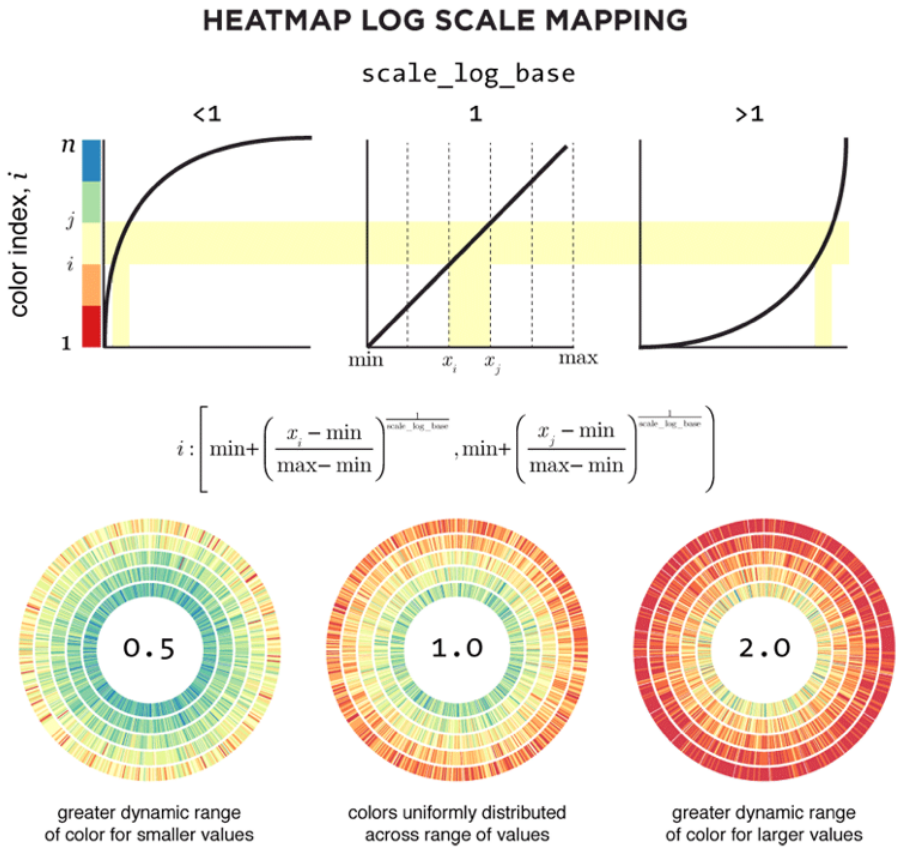
These default settings are part of the Circos distribution. You can find them in `$CIRCOS/etc/tracks`, where `$CIRCOS` is the directory where you installed Circos (e.g. `~/circos-0.69-5`).

Lesson 3 – Heatmaps, Dynamic evaluation, Rules and Overriding Parameters

Let's add a heatmap to the image. It will use the same data file as the histogram, `lm.exp.ah063.txt`, to show you how the same data looks with different encodings. Generally, you don't want to do this kind of thing—show the same data twice in two different ways—but for this exercise it's instructive.

```
<plot>
type        = heatmap
r1          = 0.975r
r0          = 0.92r
file        = conf(datadir)/lm.exp.ah063.txt
color       = reds-8-seq
#scale_log_base = 0.5
#minsize    = 25u
</plot>
```

The heatmap will map the colors from the Brewer `reds-8-seq` palette linearly onto the range of values in the input data file. As you can see from the histogram, there is a large number of small values and relatively few large values. You can change how the color mapping is done by adjusting `scale_log_base`.



Fiddle with the value from 0.5 to 0.1 and maybe even smaller. Is it helpful?

You'll also notice that some of the heatmap bins are very small—these are all genes and they can be smaller than a single pixel on the image. The `minsize` parameter enlarges the size of each data point (or interval) in a track to a minimum value. Set it to

```
minsize = 25u
```

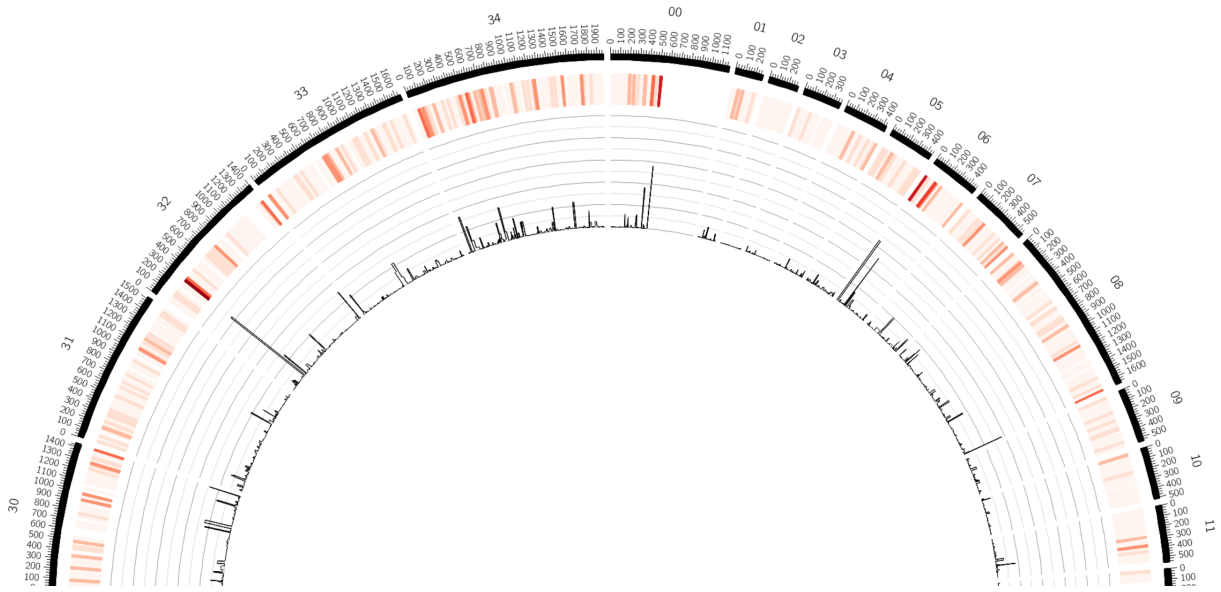
You'll see that now all the heatmap bins are larger but many are overlapping because the distance between adjacent bins is smaller than the bin width.

What we want to do is make sure we don't miss any large expression bins. This can be done by drawing them on top of bins with lower expression.

Add a rule block to the heatmap `<plot>` block that sets the `z` parameter, which controls the order of how the bins are drawn, based on the value of the track, which can be referenced using `var(value)`.

```
<rules>
use       = no
<rule>
condition = 1
z         = eval(var(value))
</rule>
</rules>
```

What is the purpose of `eval()` here?



What would happen if we instead used

```
z = eval(-var(value))
```

There are many parameters that can be accessed using var() and it's hard to remember them all. If you want to see what you can use for any given data point, ask for help like this

```
z = eval(var(?))
```

By using the ? here Circos will report (and quit) the list of parameters when it is running.

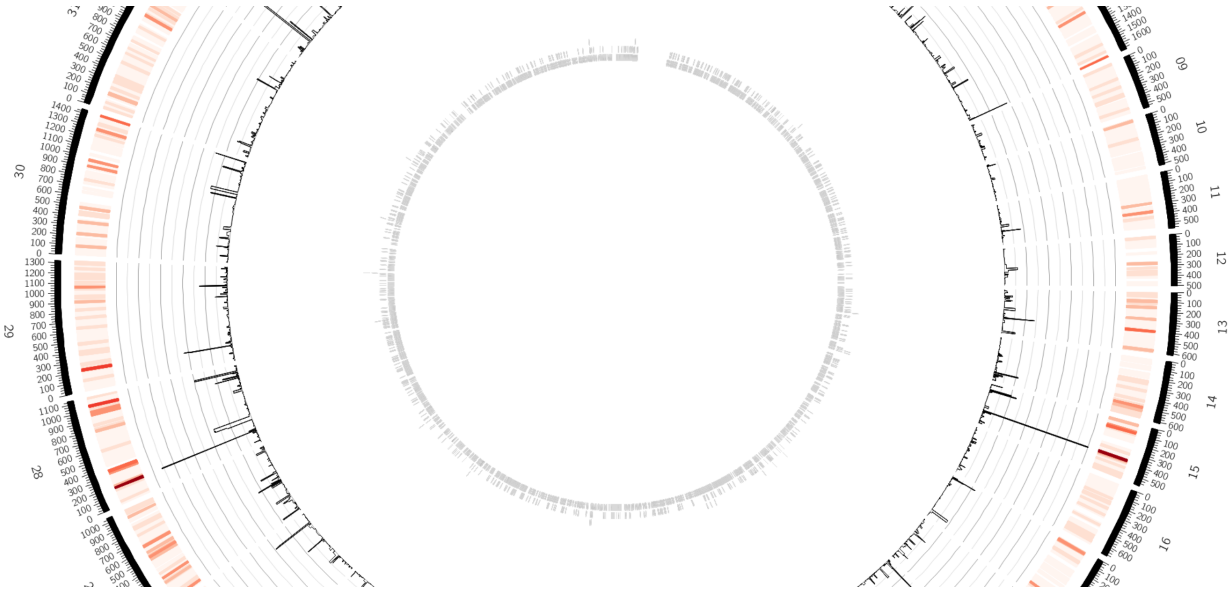
```
You asked for help in the expression [eval(var(?))].
In this expression the arguments marked with * are available for the var()
function.

                chr * LmxM.01
                end * 24999
               gene * LmxM.01.0030
                  i * 0
         next_value * 64
       plot_average * 708.355828220859
           plot_avg * 708.355828220859
           plot_max * 41018
          plot_mean * 708.355828220859
           plot_min * 0
             plot_n * 2934
            plot_sd * 2434.10318623493
        plot_stddev * 2434.10318623493
           plot_var * 5924858.32123903
                rev * 0
              start * 0
              value * 76
```

Lesson 4 – Tiles

If you want to actually draw the genome intervals (genes, exons, clones, etc) then the tile track is ideal. The input data file defines the start and end of the tile and you define how thick they should be drawn and in how many layers and Circos automatically places them within the track.

We'll draw the genes. Notice that we can use the same data file as the histogram and heatmap—at the moment we're just using the start and end positions.

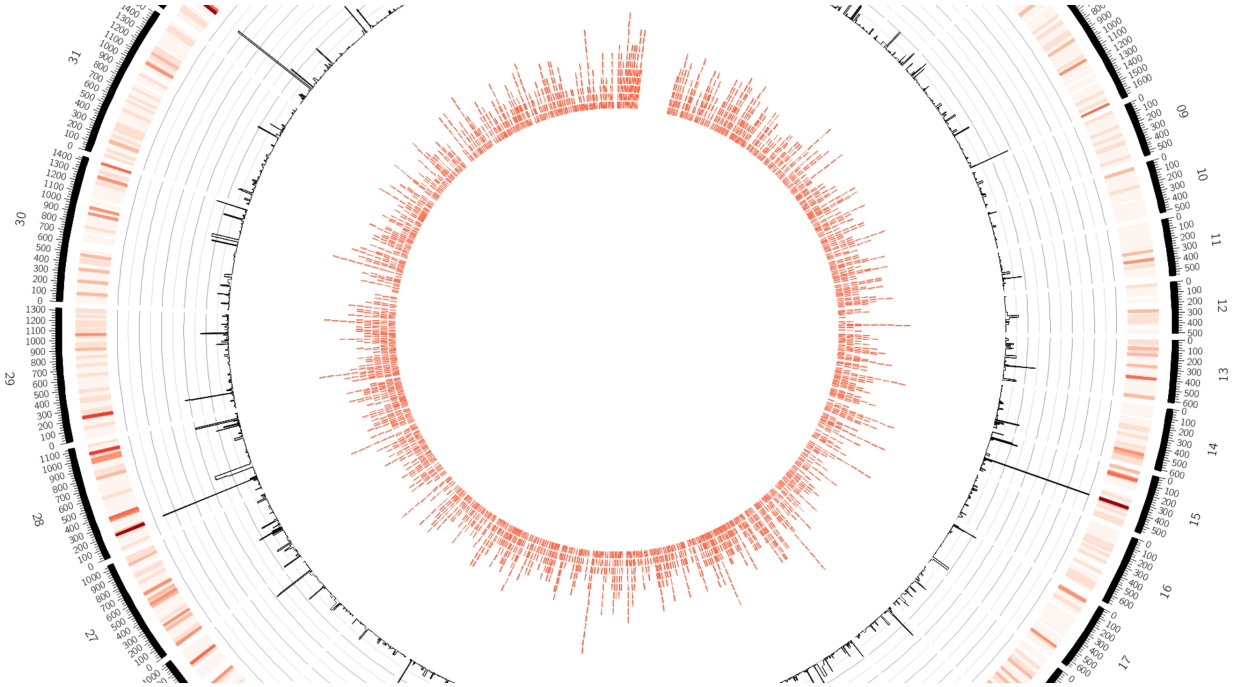
```
<plot>
type      = tile
r1        = 0.675r
r0        = 0.4r
file      = conf(datadir)/lm.exp.ah063.txt
</plot>
```



I hope you'll agree that this looks a little disappointing. But why? Think about the challenges of showing small elements in an image when they map onto a size that's not much bigger (and often smaller!) than a pixel.

There are things we can do to make these elements more visible. First, let's change the color and remove the outline. Also, we'll make each element at least `10u` in size (what does the `u` suffix do?).

```
color     = red
stroke_thickness = undef
minsize   = 10u
thickness = 15
margin    = 5u
```

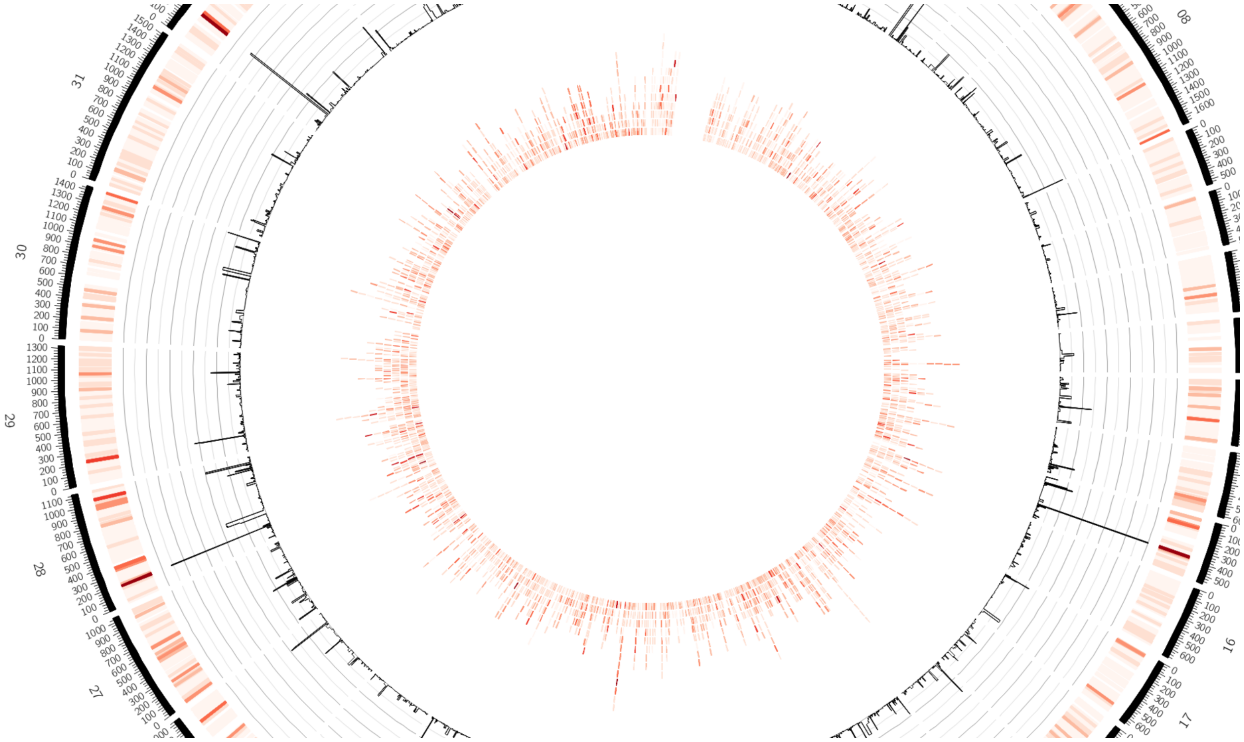


Ok, that looks better! Now recall that our data file had an expression value for each gene. We're drawing the genes as tiles, but we can easily map the expression value onto the color of the tile. All we do is instead of a specific color (e.g. red) we use the name of a color list.

```
color = reds-8-seq
```

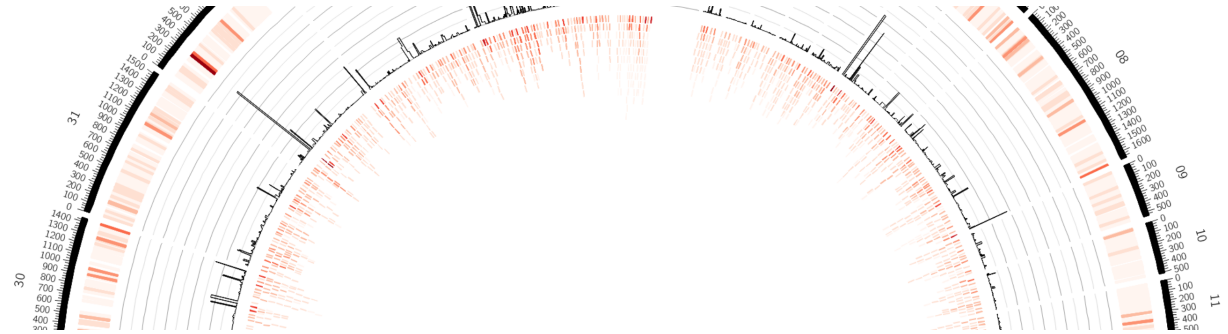
We'll run into the same color mapping problem as in the heatmap track—many expression values are small and they'll be drawn with a faint red. Alter the remapping by using

```
scale_log_base = 0.25
```



You can also play with whether the tiles are oriented inward or outward and how they are ordered within the layers.

```
orientation      = in
sort             = value
sort_direction   = desc
```



Now try using the `spectral-11-div` palette and use a `minsize` of 50u.

Lesson 5 – Scatter Plots

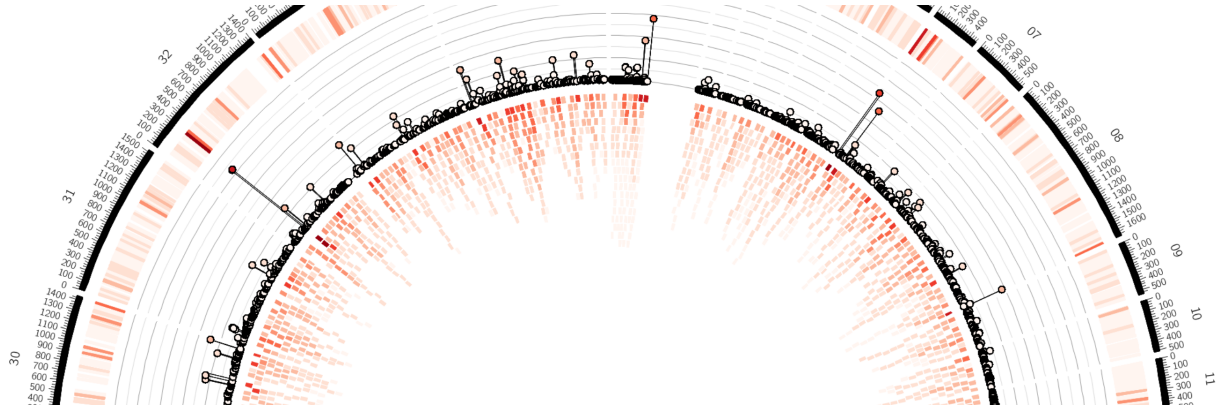
As before, to learn about this track, we'll plot the expression data again.

```
<plot>
type       = scatter
r1         = 0.9r
r0         = 0.7r
file       = conf(datadir)/lm.exp.ah063.txt
</plot>
```

Except the trick here will be to draw the scatter plot in the same position as the histogram track we've already drawn. Because the scatter track `<plot>` block comes after the histogram, it will be drawn second (i.e. on top) to give us a result that looks like a lollipop diagram.

As with tiles you can automatically map the value onto the color of the scatter symbol by setting the color to be a list.

```
color            = reds-8-seq
stroke_color     = black
stroke_thickness = 2
glyph_size       = 12
```

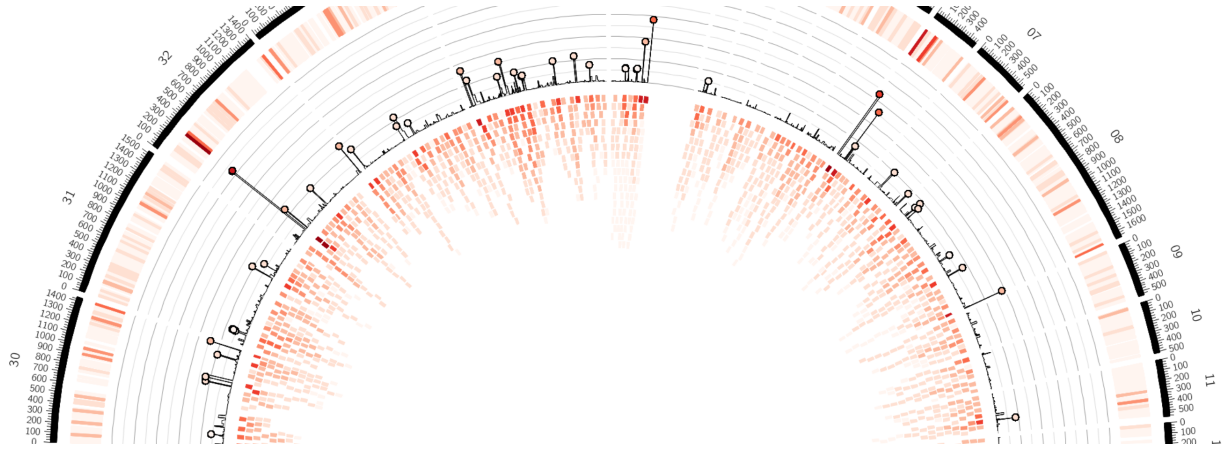


You can adjust the symbol shape using `glyph_shape`. For example, try `square`.

To emphasize large expression values, we'll hide scatter symbols for any values less than 10% of the maximum.

```
<rules>
use        = yes
<rule>
condition  = var(value) < 0.1*var(plot_max)
glyph_size = 0
</rule>
</rules>
```

The `var(plot_max)` gives us access to the maximum value of the track, which is one of the aggregate statistics computed for each track. Others are `plot_min` and `plot_avg`.



Lesson 6 – Text, Parameter Inheritance

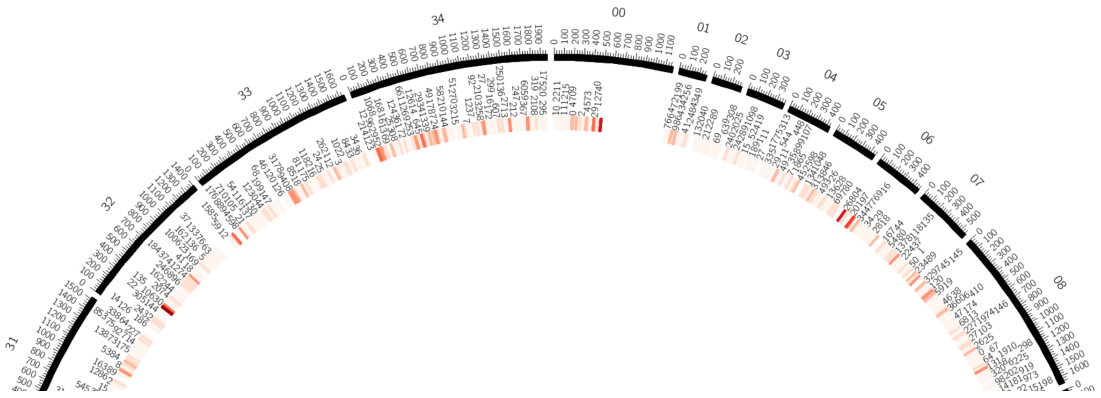
Let's draw some text labels.

When experimenting with new tracks it's useful to turn the display of all other tracks off and only display the track you are debugging.

To achieve this, set `show = no` globally for all `<plot>` blocks by defining this parameter immediately in the `<plots>` block. Then, for the block you wish to draw (i.e. the text track), set `show = yes`.

```
<plots>

show = no

<plot>
show = yes
type = text
r1   = dims(ideogram,radius_inner)
r0   = 0.9r
file = conf(datadir)/lm.exp.ah063.txt

</plot>

</plots>
```

If a block has `show` defined, this value will be used. If not, then Circos will look in the parent block.

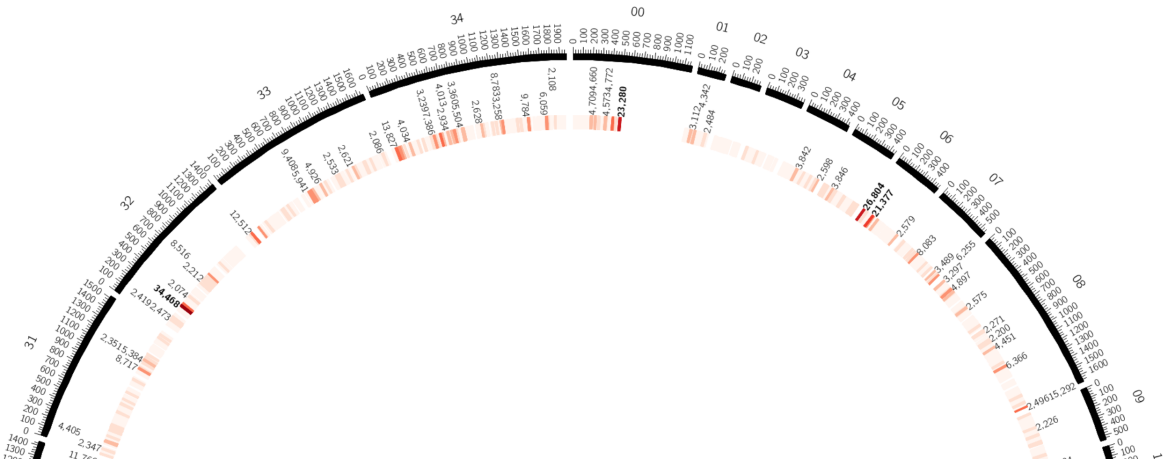


What's being shown in the text track is the expression value. Let's manipulate this by using rules.

```
<rules>
use = yes
<rule>
condition  = var(value) < 0.05*var(plot_max)
show       = no
</rule>
<rule>
condition  = var(value) > 0.5*var(plot_max)
```

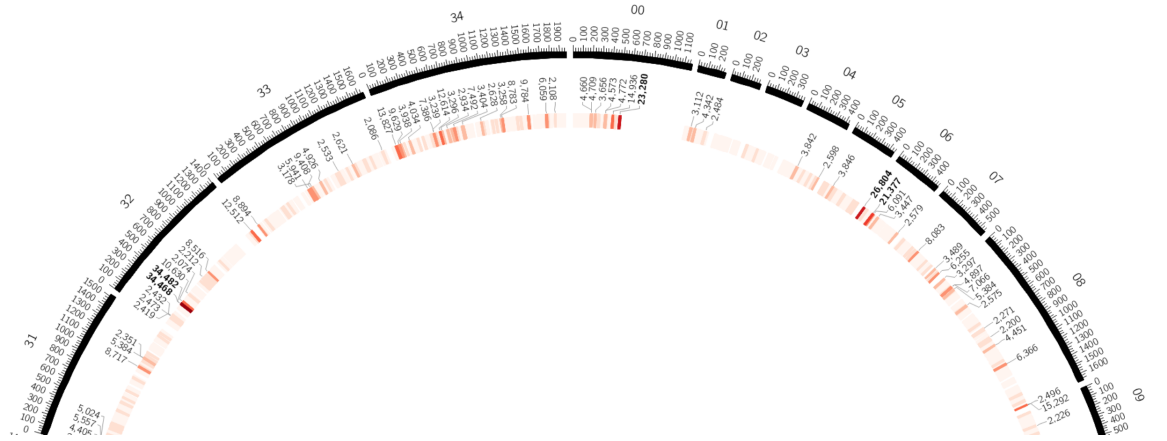
```
label_font = bold
flow       = continue
</rule>
<rule>
condition  = 1
value      = eval(replace(var(value),qr/\B(?=(\d{3})+(?!\d))/,","))
</rule>
</rules>
```

You should be able to figure out what the first two rules do. The last one is a complex regular expression that adds thousands separators to a number (e.g. 10235 becomes 10,235).



But things are still messy. You can ask Circos to slightly adjust the position of the labels to avoid overlap and unnecessary stacking.

```
label_snuggle = yes
show_links    = yes
link_color    = grey
max_snuggle_distance = 3r
```



Now turn on the display of all the other tracks by setting the outer `show = yes`.

Lesson 7 – Modular Configuration

By now our configuration file for this example has grown. When you need to scroll across many screenfulls, it can be hard to keep track of what's what.

You've already seen the `<<include>>` directive in the configuration file—it was used to define default parameters such as system values, image size, and so on. We can also use it to factor out functional (and/or commonly reused) parts of the file to other files.

For example, instead of

```
<plot>
...
<axes>
...
</axes>
</plot>
```

we can move the `<axes>` block to `axes.conf` and

```
<plot>
...
<<include axes.conf>>
</plot>
```

This is very useful if we are reusing the same axes definitions in other blocks.

Factor out all the `<rules>` blocks from the configuration file as instructed in the configuration file. Make sure not to forget any closing blocks!

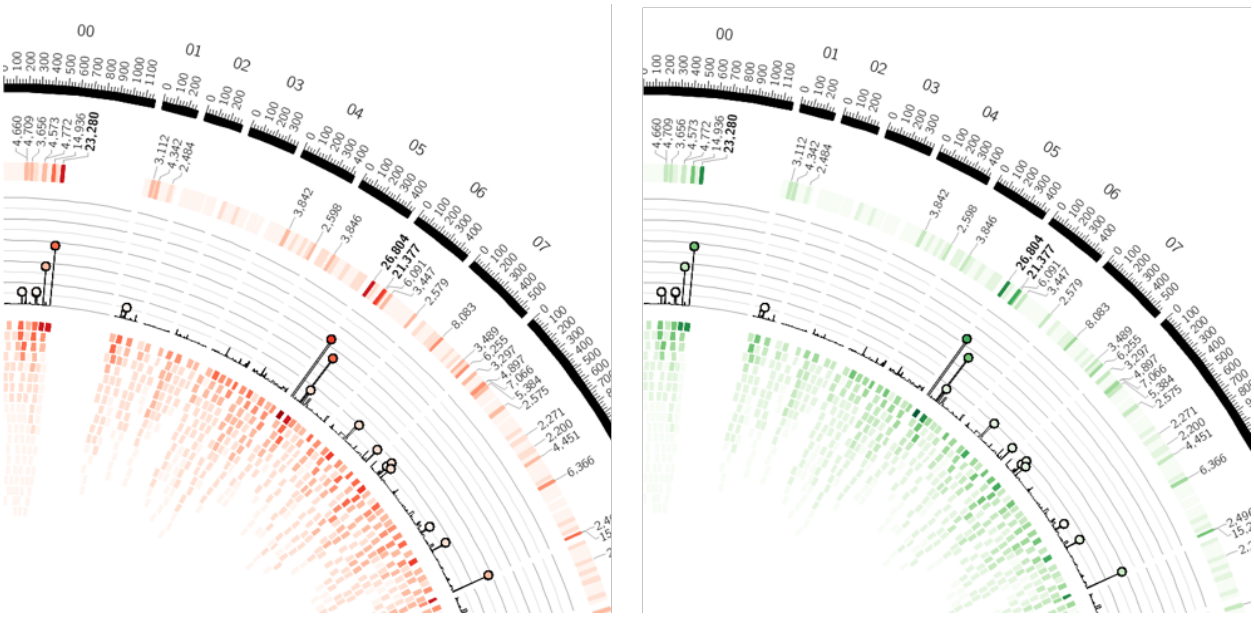
Lesson 8 – Parameter Inheritance and Ideogram Filtering

You've already seen how the show parameter can be inherited by child blocks.

Let's now put this to good use. Many of the `<plot>` blocks define color. Let's define this globally.

```
<plots>
color = reds-8-seq
...
</plots>
```

The benefit of doing this is that you can change the file in one place and the colors of all the tracks change.



For the image on the right, I've set

```
<plots>
colors = greens-8-seq
</plots>
```

Now define

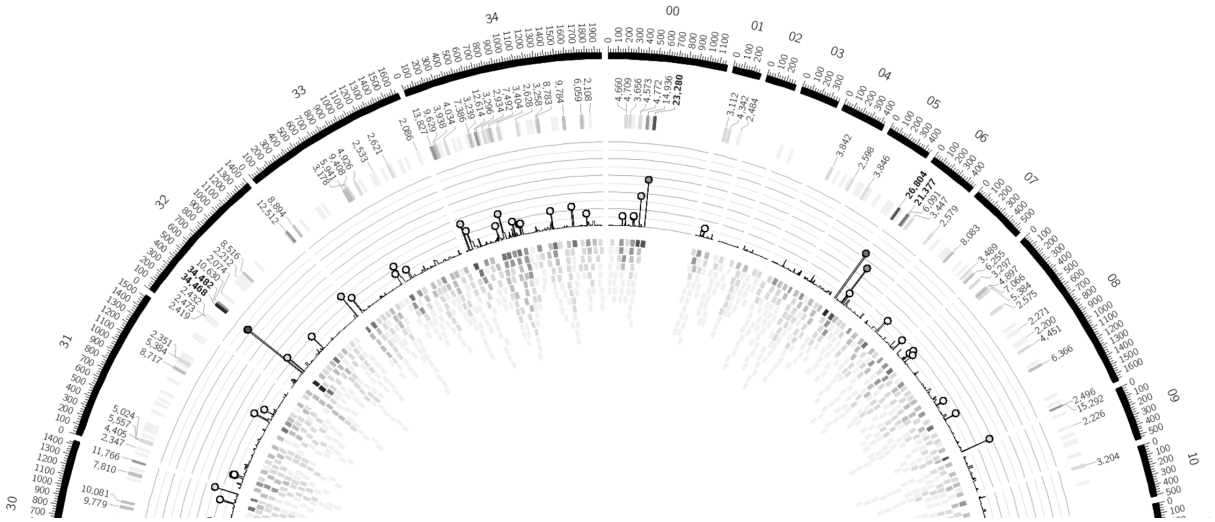
```
scheme = reds
...
<plots>
colors = conf(scheme)-8-seq
...
</plots>
```

Great! Now you only have to specify the colors (e.g. reds, blues, greens) and not bother with the number of colors in the palette (8, which is hard-coded).

Now you're free to adjust the colors on the command line.

```
circos -param scheme=reds
circos -param scheme=blues
circos -param scheme=greens
```

Any scheme that has an associated Brewer palette defined will work. Don't forget the power of black-and-white using `scheme=greys`.



For a challenge, how would you similarly change the number of colors in the palette, independently of the color?

Finally, let's change which ideograms are being drawn.

```
chromosomes_display_default = no

chromosomes = LmxM.00:0-600;LmxM.06;LmxM.15;LmxM.21;LmxM.24;LmxM.28;LmxM.29;LmxM.32
```

or more conveniently using regular expressions

```
chromosomes = /00/:0-600;/06/;/15/;/2[1489]/;/32/
chromosomes_color = /00/=grey;/15/=dgreen
```

And to end things off, try

```
circos -random black,white
```

this will randomize all colors except black and white.