Genome Visualization with Circos

Session 7 — Comparing Genomes with Orthology

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
v0.23 7 Jun 2017
v0.22 27 Apr 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

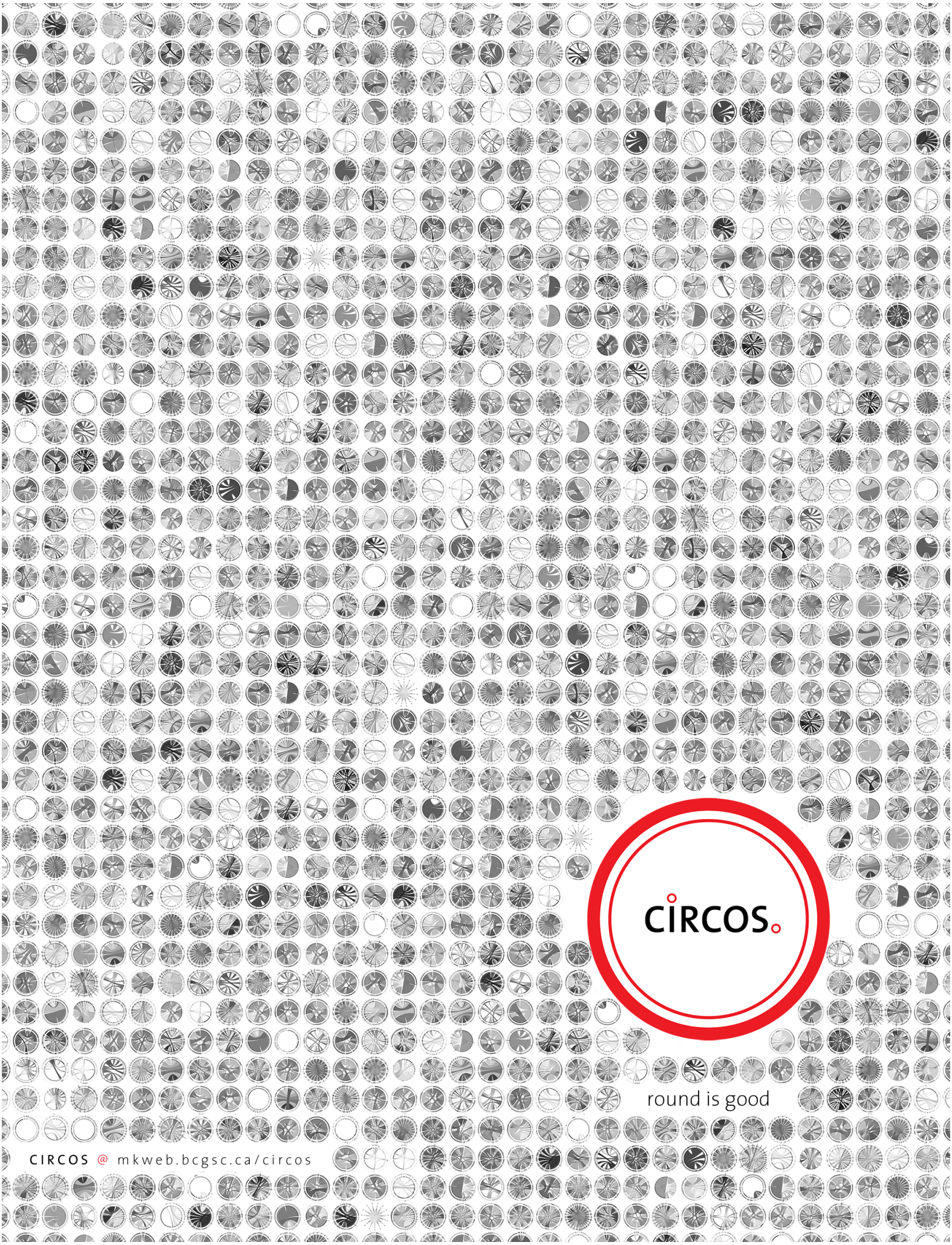

CIRCOS.
round is good
CIRCOS @ mkweb.bcgsc.ca/circos

Table of Contents

Drawing Data from Multiple Samples

Refer to `handout/session-1-preamble.pdf` for a refresher about file organization, commands and common errors.

There is nothing inherently different between drawing chromosomes from different genomes than chromosomes from the same genome. Circos has no concept of "genome"—only "chromosomes".

Conventionally in Circos, chromosomes are named using the species acronym. Thus, `hsNN` for *Homo sapiens*, `mmNN` for *Mus musculus*, and so on. The karyotype files that come with Circos (check `$CIRCOS/data/karoytypes`, if you're curious) reflect this convention. The goal of this approach is to make drawing multiple genomes possible (all chromosomes must have unique names) and make the chromosomes easily recognizable. For example, if you see a data file with the line

```
rn5 1100000 1250000 0.7
```

you know right away that this is for a rat genome (*Rattus Norvegicus*, `rn`), as long as the file respects Circos' convention for naming.

Switch to Session 7 Lesson 1 directory and follow along.

```
# or wherever the lesson directory can be found
> cd ~/circos-course/session/7/1
```

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 – Multiple Karyotypes and Links

We'll be drawing both *Leishmania* genomes. So, we need to karyotype files. These are

```
# ../data/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.03 03 0 375930 black
chr - LmxM.04 04 0 438817 black
...
# ../data/lf.karyotype.txt
chr - LmjF.36 36 0 2682151 black
chr - LmjF.35 35 0 2090474 black
chr - LmjF.34 34 0 1866748 black
chr - LmjF.33 33 0 1583653 black
chr - LmjF.32 32 0 1604637 black
...
```

These were extracted from the GFF files with the `data/make.data.files` script.

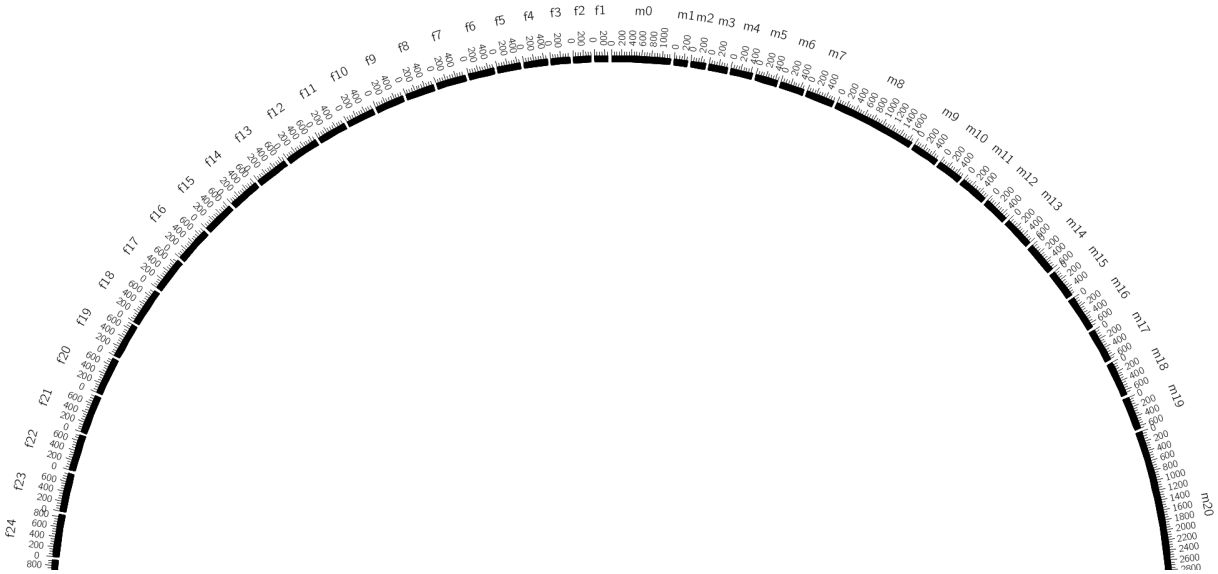
If I were to use the Circos naming convention, the *L. mexicana* chromosomes would be named `lmexNN` and *L. Major* lmajNN. However, I've kept the names that you've been using in other lectures (`LmxM.NN`, `LmjF.NN`) for consistency.

In general, it's a good idea to avoid capitalization and unnecessary punctuation, where possible.

In circos.conf you'll find that the karyotype parameter is now a list

```
karyotype = ../data/lm.karyotype.txt,../data/lf.karyotype.txt
```

The ideograms are drawn in the order of appearance in the karyotype files.



When comparing two genomes, it's useful to reverse the orientation of the chromosomes from one genome. This way, when links are used to show alignments, identical neighbouring alignments that are not inverted do not cross in the image.

```
chromosomes_reverse = /jF/
chromosomes_color   = /./=var(chr)
```

This can be achieved with the `chromosomes_reverse` parameter, which takes a list of chromosome names or a regular expression. Which chromosomes match `/jF/`?

The `chromosomes_color` parameter assigns each chromosome a color named after the chromosome.

These colors are actually quite special. They are not selected from the HSV space, as you might imagine, since they appear to be a spectrum.

LCH COLORS

Look in the configuration file for the extra color definition.

```
luminance = 70
chroma    = 100

<colors>
lmjf.00=lch(conf(luminance),conf(chroma),0)
lmjf.01=lch(conf(luminance),conf(chroma),9)
lmjf.02=lch(conf(luminance),conf(chroma),19)
lmjf.03=lch(conf(luminance),conf(chroma),29)
lmjf.04=lch(conf(luminance),conf(chroma),38)
...
```

The `lch(L,C,H)` function defines a color using the LCH color space. This color space is like a perceptually uniform HSB. L is the perceived brightness of the color, C is the chroma, or richness, and H is the hue.

The benefit of LCH is that if you fix L and change H, the perceived brightness of a color doesn't change, unlike in HSB, where a pure blue and pure yellow have a very different apparent brightness (one is very dark and the other very bright). Some colors are impossible. For example, there is no such thing as a low L and high C yellow—yellow is either bright and pure (L,C both high) or dark and muddy (L,C both low).

Generate the image with L=70, C=100 and again with L=100, C=70. Which one do you prefer?

You can change the configuration parameters at runtime using the `-param` flag. For example

```
> circos -param luminance=70 -param chroma=100
> circos -param luminance=100 -param chroma=70
```

Lesson 2 – Links

Links are defined using a pair of coordinates—the start and the end of the link.

```
# data/ortho.links.txt
LmxM.01 9205 11223 LmjF.01 9061 11067 gene1=LmjF.01.0030,gene2=LmxM.01.0030
LmxM.13 567145 569319 LmjF.01 9061 11067 gene1=LmjF.01.0030,gene2=LmxM.13.1610
LmxM.01 14846 16843 LmjF.01 15025 17022 gene1=LmjF.01.0050,gene2=LmxM.01.0050
LmxM.30 1416292 1418355 LmjF.01 15025 17022 gene1=LmjF.01.0050,gene2=LmxM.30.3130
LmxM.01 27727 28314 LmjF.01 28521 29108 gene1=LmjF.01.0110,gene2=LmxM.01.0110
LmxM.01 42656 44968 LmjF.01 44148 46466 gene1=LmjF.01.0180,gene2=LmxM.01.0180
```

Just like other data files, each data point (now a link) can be combined with a list of parameters. Here I store the name of the gene at the start (`gene1`) and end (`gene2`) of the link.

These links represent pairs of orthologous genes.

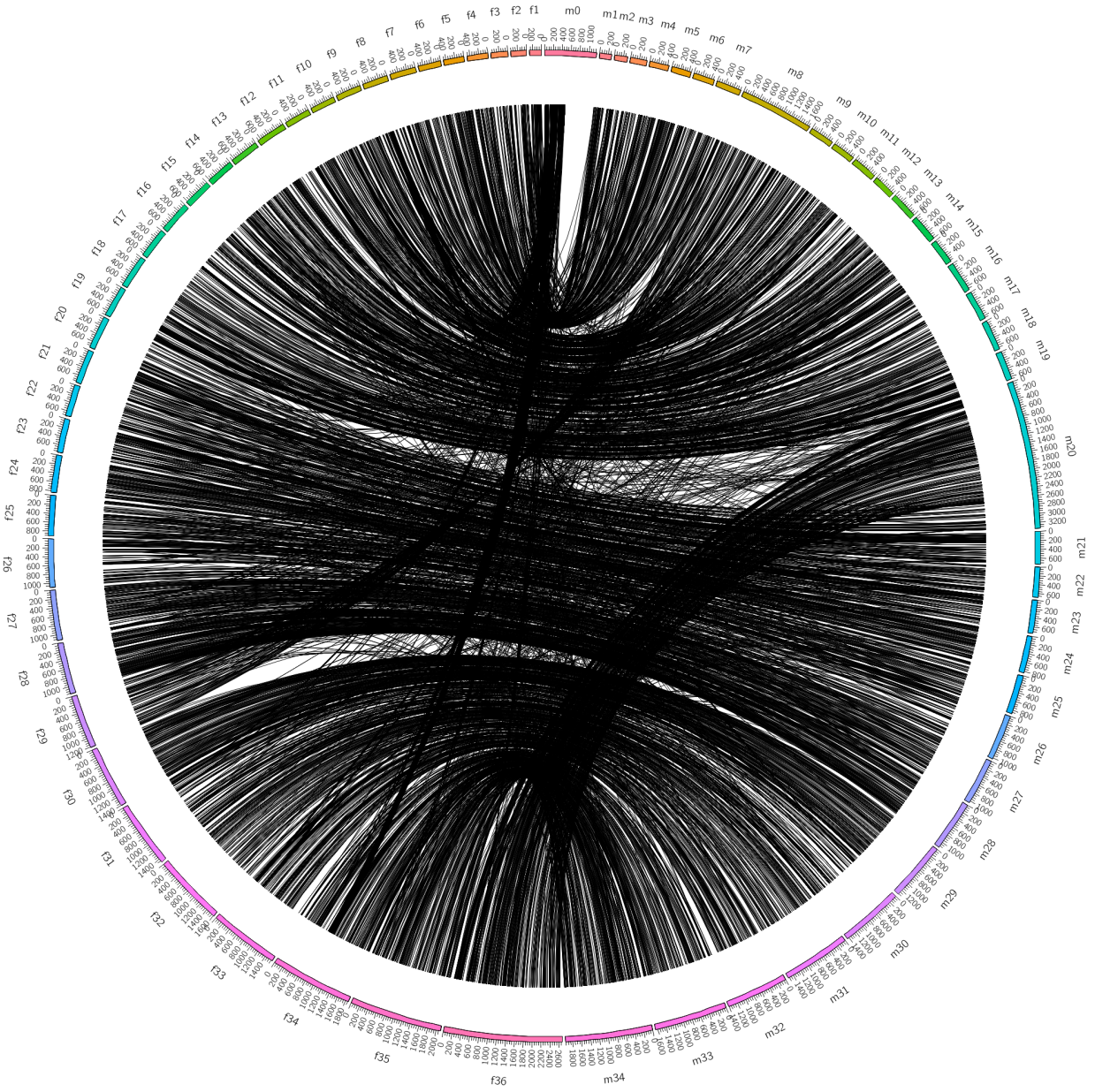
Links are defined in a `<link>` blocks, which work just like `<plot>` blocks. Eventually, I will remove the need for this distinction.

```
<links>
<link>
file   = conf(datadir)/ortho.links.txt
radius = 0.90r

<rules>
use = no
<rule>
condition = substr(var(chr1),-2,2) eq substr(var(chr2),-2,2)
show      = no
</rule>
<rule>
condition = 1
color     = eval(sprintf("%s_a4",lc var(chr1)))
</rule>
</rules>

</link>
</links>
```

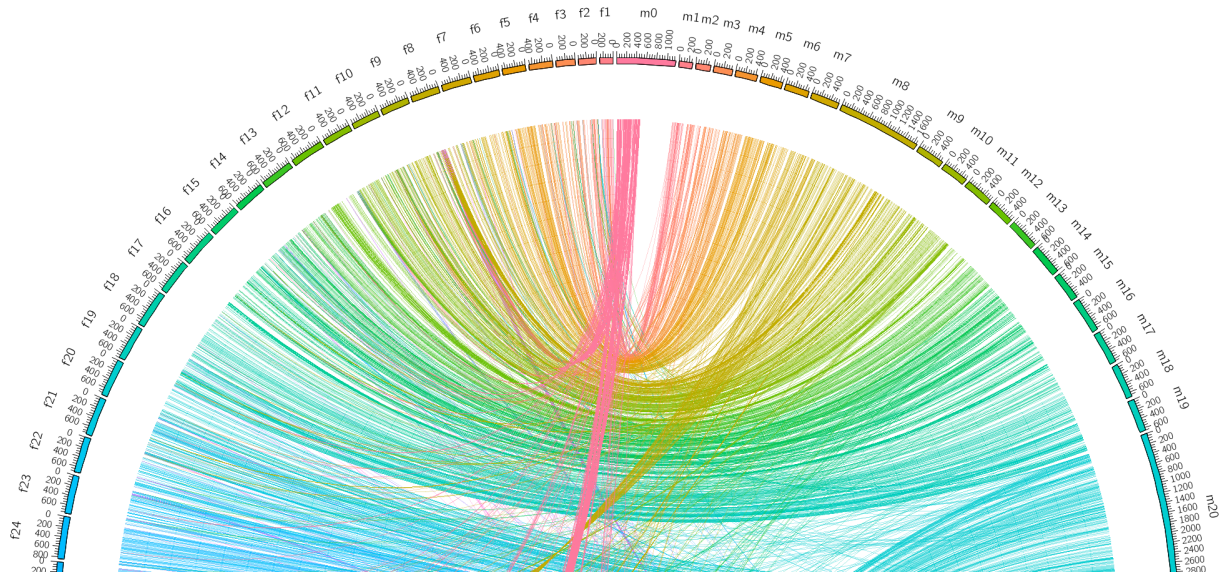
Just like other tracks, links accept default values. List the default values for the link track by looking in `$CIRCOS/etc/tracks/link.conf`.

The default values help you draw tracks quickly. You can adjust parameters in their block later. Below is the image you should get when you run Circos for this lesson.

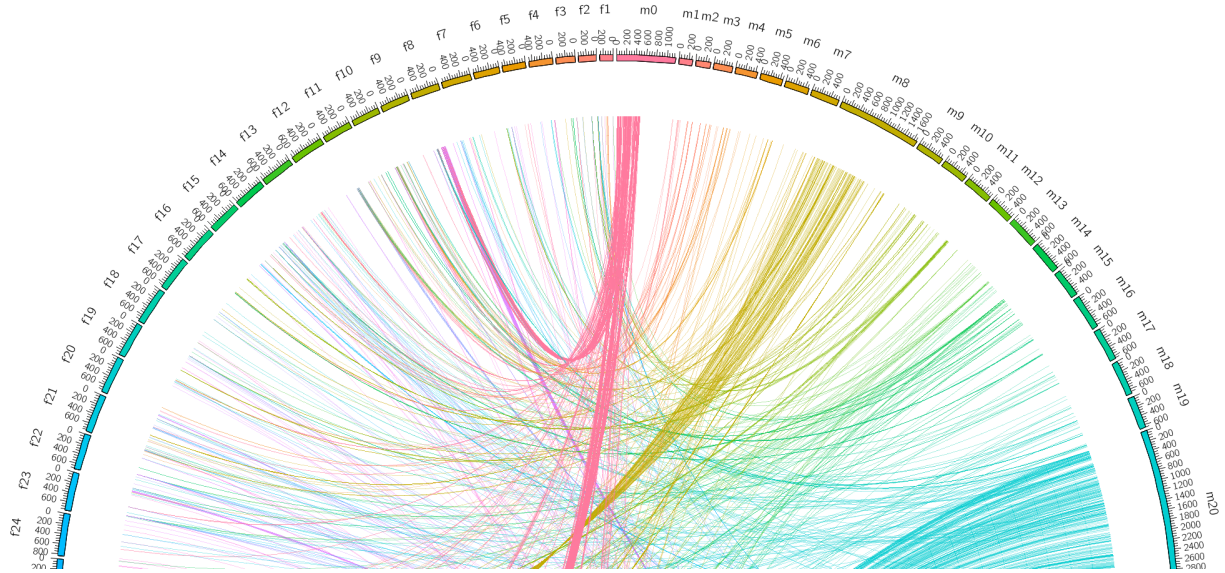
How many links are there in the input file? Hint: use the `wc` command.



There are a couple of rules. Turn on the second rule only (not the first). How would you do that? What is the `_a4` suffix?



Now turn on all rules.



What do you think the first rule is doing? Hint: `var(chr1)` and `var(chr2)` are the chromosome names of the start and end of the link, respectively. The `substr(X,OFFSET,LEN)` function returns a part of a string starting at `OFFSET` of length `LEN`. If LEN is omitted, it is assumed to be to the end of the string. If `OFFSET` is negative, then it counts from the end of the string. Thus, `substr(STR,-2)` returns the last two characters of `STR`.

Lesson 3 – Counting Links and Consensus Chrs

When drawing relationships between genomes (e.g. orthology, or any other kind of alignments), you may ask which chromosomes of one species connect to a given chromosome of the other. For example, are all the genes on `LmxM.01` orthologous to genes on `LmjF.01`, or are other `LmjF` chromosomes involved?

BINNING LINKS (OPTIONAL)

If you installed the Circos tools, do this section. If you want to install them, you can get the tools at

http://www.circos.ca/software/download/utilities/

A tool that helps you do this is `binlinks`.

```
> cd 6/data
# assuming you installed the tools in $CIRCOS
> cat ortho.links.txt |
    $CIRCOS/tools/binlinks/bin/binlinks -bin 50000 -link_end 2 -num -color_by_chr
```

Look at the output. The number of link ends in each 50kb bin is listed and the color is the color of the chromosome with the largest number of incident links to this bin. By convention, all Circos color names are lowercase.

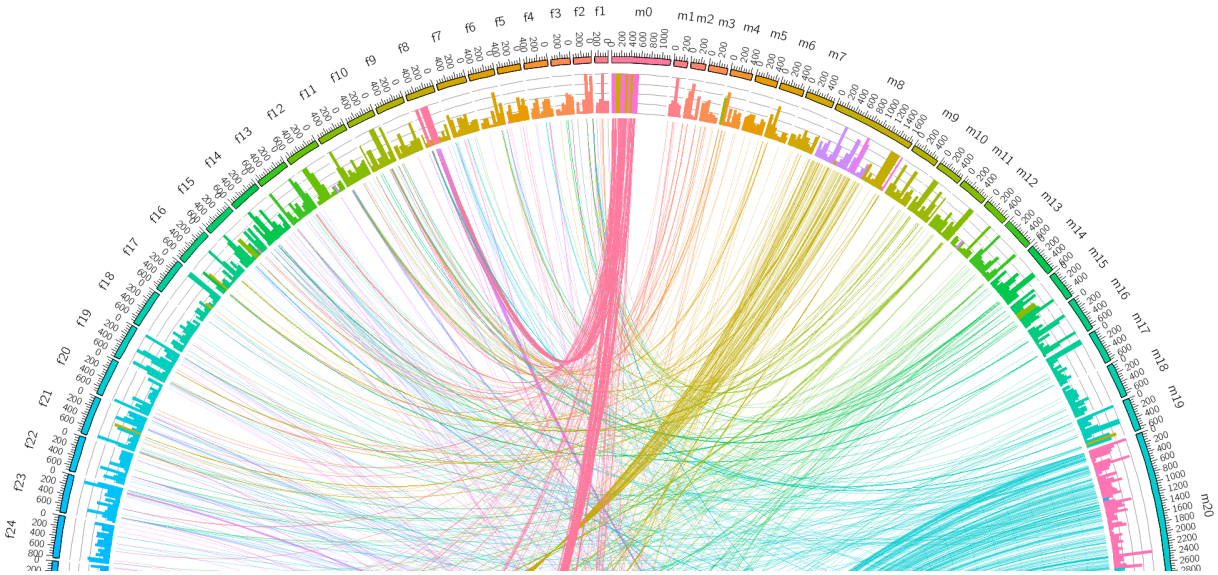
```
LmjF.01 0 49999 6.0000 fill_color=lmxm.01
LmjF.01 50000 99999 6.0000 fill_color=lmxm.01
LmjF.01 100000 149999 25.0000 fill_color=lmxm.01
LmjF.01 150000 199999 7.0000 fill_color=lmxm.01
LmjF.01 200000 249999 5.0000 fill_color=lmxm.01
LmjF.01 250000 299999 4.0000 fill_color=lmxm.01
...
```

DRAWING HISTOGRAM OF BINNED LINKS

This data file can be used by a histogram!

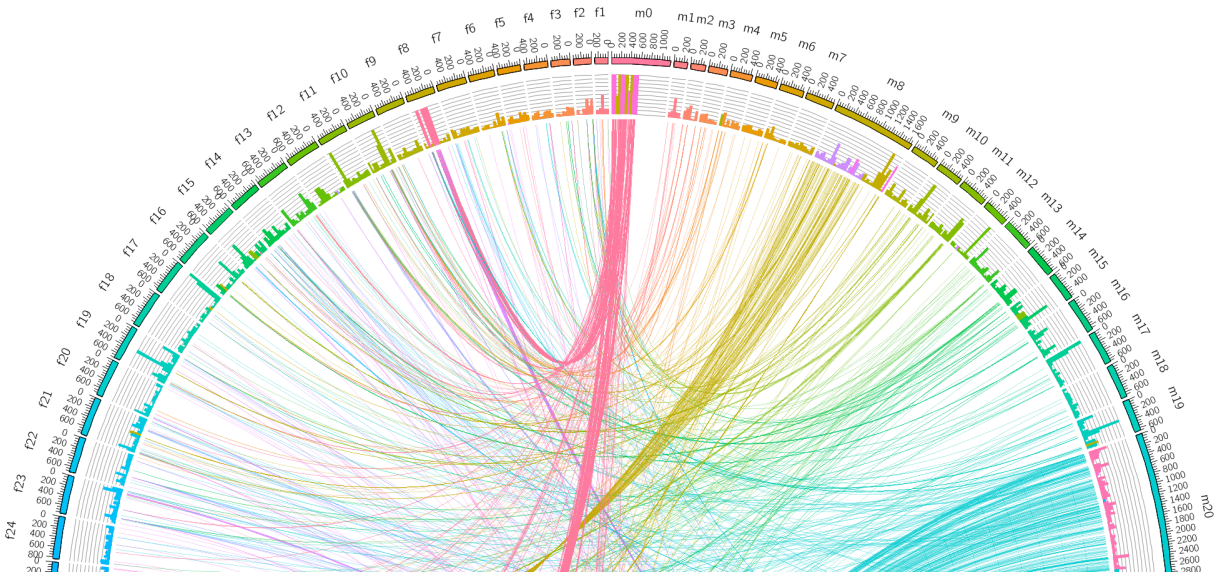
```
<plot>
type  = histogram
file  = conf(datadir)/hist.links.txt
r1    = 0.98r
r0    = 0.91r
color = undef
min   = 0
max   = 20

<axes>
<axis>
spacing = 5
</axis>
</axes>

</plot>
```

I've already created the input file `hist.links.txt` for you.



Notice that some of the histogram bins are capped—they exceed the maximum value of the histogram plot. If `min` and/or `max` are not defined, they are set to the data min and max values. Sometimes it's useful to explicitly define these bounds, for example, if you have outliers.

Change the `max` to 50.



Notice that there is many small counts and a few large counts. What would be a way to reduce the dynamic range of these values? What function would you use? Write a rule for the track that applies `sqrt()` to the values. Hint: don't forget `eval()`.

What is the largest bin value in the data file? Hint: at the command line, use `sort` with numerical sort flag.

Lesson 4 – Highlights from Binned Links

As you've already seen, you can use the same input data file to many different tracks. The bin count file we created in the previous lesson was used to draw a histogram.

Now, we'll use it to draw a highlight that emphasizes areas of `LmJF` chromosomes that are associated with a `LmxM` chromosome with a different index.

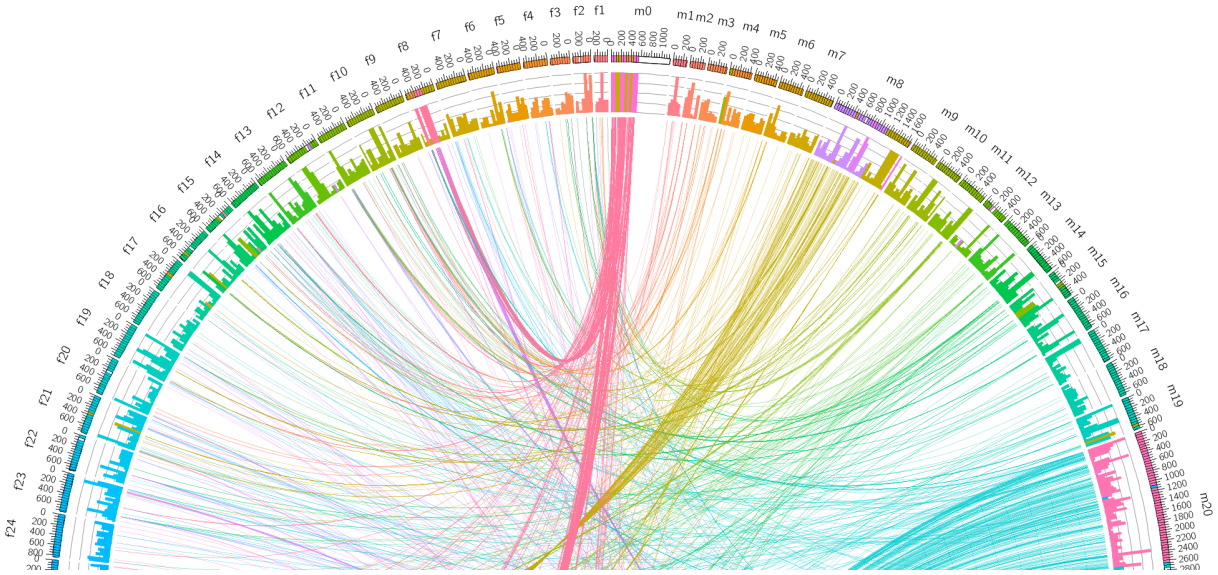
Let's look at the `hist.links.txt` file again.

```
LmjF.16 550000 599999 12.0000 fill_color=lmxm.08
LmjF.16 600000 649999 15.0000 fill_color=lmxm.14
LmjF.16 650000 699999 5.0000 fill_color=lmxm.16
LmjF.17 0 49999 57.0000 fill_color=lmxm.17
LmjF.17 50000 99999 36.0000 fill_color=lmxm.17
LmjF.17 100000 149999 5.0000 fill_color=lmxm.17
LmjF.17 150000 199999 2.0000 fill_color=lmxm.17
LmjF.17 200000 249999 1.0000 fill_color=lmxm.17
LmjF.17 250000 299999 3.0000 fill_color=lmxm.17
LmjF.17 300000 349999 4.0000 fill_color=lmxm.07
LmjF.17 350000 399999 7.0000 fill_color=lmxm.17
LmjF.17 400000 449999 3.0000 fill_color=lmxm.17
...
```

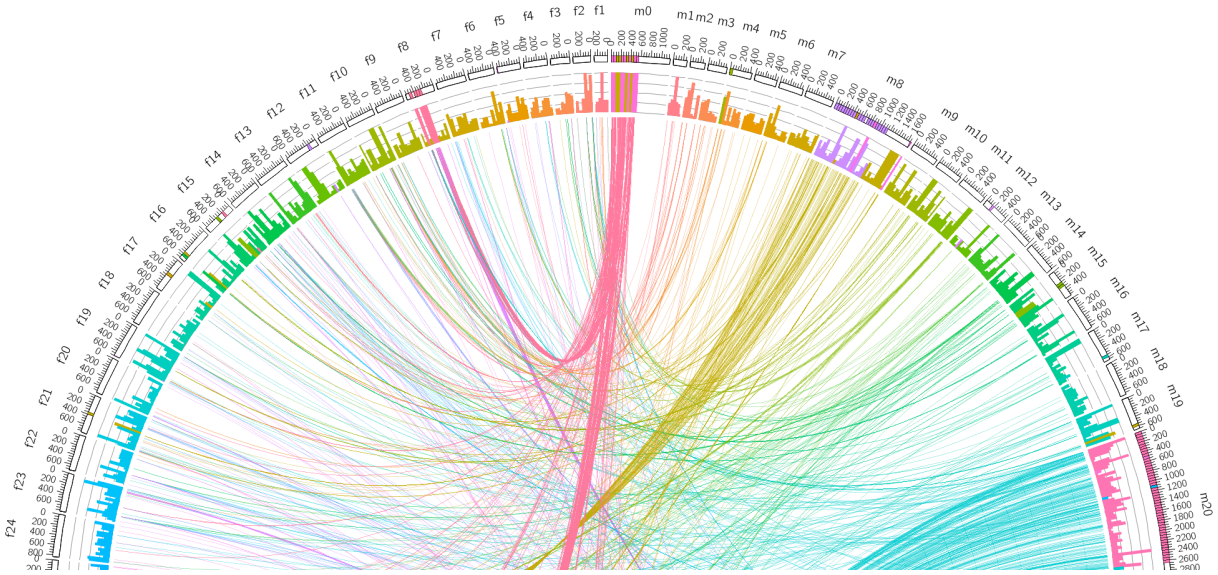
I've highlighted the chromosome indeces of links in which they are different.

```
chromosomes_color = /./=white

<plot>
type  = highlight
file  = conf(datadir)/hist.links.txt
r1    = dims(ideogram,radius_outer)
r0    = dims(ideogram,radius_inner)
stroke_color = black_a3

<rules>
use = no
<rule>
condition = var(chr) =~ /xm/i
   || substr(var(fill_color),-2) eq substr(var(chr),-2)
show      = no
</rule>
</rules>

</plot>
```

Look at the ideogram ring of the image carefully. Why are some parts of the ideogram white? Where are areas where adjacent highlights have different color? What do these represent?



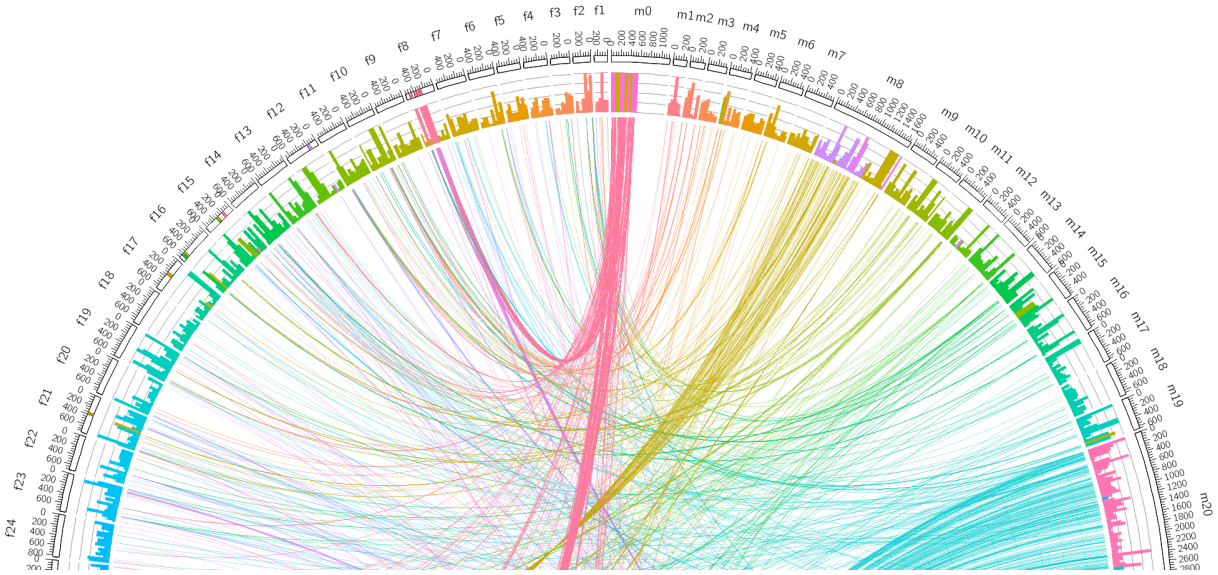
Now turn on the rules. What's the difference between setting `use=yes` and removing this parameter altogether? Can you think of use cases for each?



Notice that the rule has two conditions, with the first one commented out.

```
#condition = var(chr) =~ /xm/i
condition = substr(var(fill_color),-2) eq substr(var(chr),-2)
show      = no
```

Combine both into a single condition using the `||` operator. You can also use `or`.



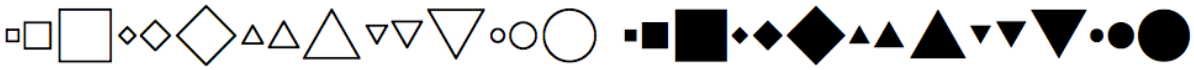
What has happened? Why?

Lesson 5 – Bubble Tracks

A bubble track is a text track in which the text has been replaced by a symbol. By using the symbol font, you can gain access to many common shapes.

Lower case letters are hollow shapes and capital letters are filled shapes. Note that you cannot fill the hollow shapes with a color.

```
abcdefghijklmno   ABCDEFGHIJKLMNO
```



We'll start with an input file that lists the chromosome of a gene's orthology partner.

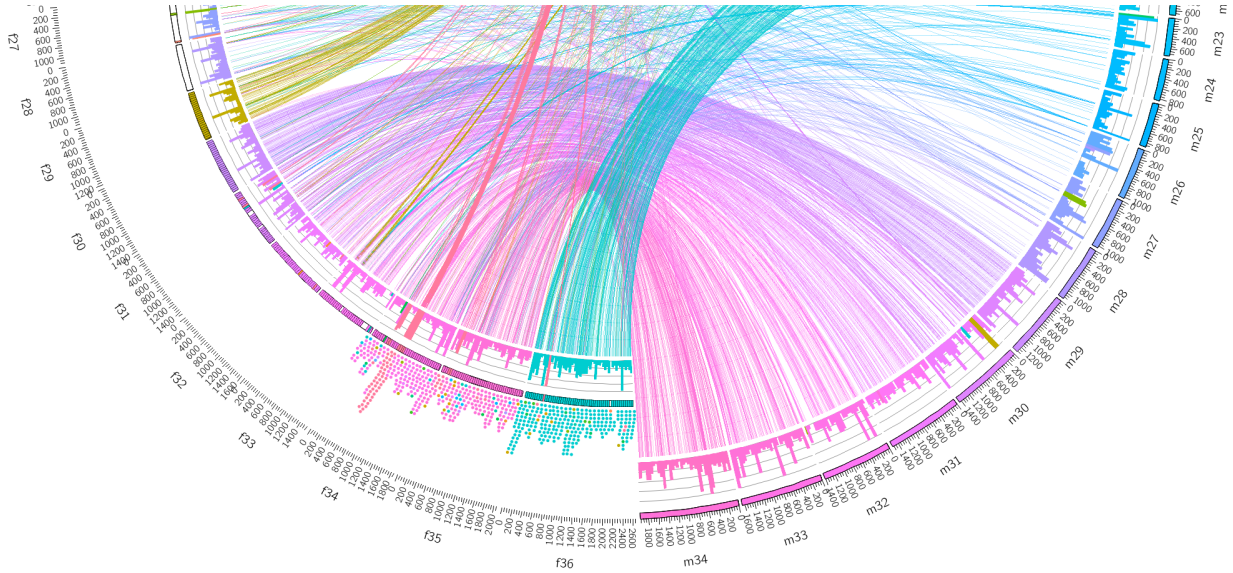
```
LmjF.01 100528 101130 LmxM.01 gene1=LmjF.01.0410,gene2=LmxM.01.0410
LmjF.01 102444 103046 LmxM.01 gene1=LmjF.01.0420,gene2=LmxM.01.0410
LmjF.01 112139 112795 LmxM.01 gene1=LmjF.01.0450,gene2=LmxM.01.0450
LmjF.01 121759 123849 LmxM.01 gene1=LmjF.01.0470,gene2=LmxM.01.0470
...
```

The first thing to notice is the `chromosomes_radius` parameter. It's been changed to `0.8r` for some of the chromosomes. Which ones?

The text track applies the `glyph` font and turns on label snuggling.

```
<plot>
type  = text
file  = conf(datadir)/ortho.labels.txt
r0    = 1r+5p
r1    = dims(image,radius)-100p
label_font = glyph
label_snuggle = yes

<rules>
<rule>
condition = var(chr) !~ /\.3[456]/
#condition = abs(substr(var(chr),-2) - substr(var(value),-2)) == 0
show      = no
</rule>
<rule>
# Sets the value of the label to N, which is a filled circle.
condition = 1
color     = eval(lc var(value))
value     = N
</rule>
</rules>

</plot>
```

To speed up creation of the track, glyphs for only some chromosomes are being shown. Which rule achieves this? Which chromosomes match the regular expression?



Now change the conditions so that the second one is being used.

```
<rules>
<rule>
#condition = var(chr) !~ /\.3[456]/
condition = abs(substr(var(chr),-2) - substr(var(value),-2)) == 0
show      = no
</rule>
<rule>
# Sets the value of the label to N, which is a filled circle.
condition = 1
color     = eval(lc var(value))
value     = N
</rule>
</rules>
```

What is the condition in the first rule actually testing? Can you think of the benefit of doing this over

```
condition = substr(var(chr),-2) eq substr(var(value),-2)
```


m0 m1 m2 m3 m4 m5 m6 m7 m8 m9 m10 m11 m12 m13 m14 m15 m16 m17 m18 m19 m20 m21 m22 m23 m24 m25 m26 m27 m28 m29 m30 m31 m32 m33 m34
f1 f2 f3 f4 f5 f6 f7 f8 f9 f10 f11 f12 f13 f14 f15 f16 f17 f18 f19 f20 f21 f22 f23 f24 f25 f26 f27 f28 f29 f30 f31 f32 f33 f34 f35 f36

Lesson 6 – Bundling Links

When you are showing a very large number of links in an image (e.g. > 10,000) it can be difficult to discern patterns—even using colors with transparency can result in overplotting (a general term to describe many overlapping data points and, in this case, lines).

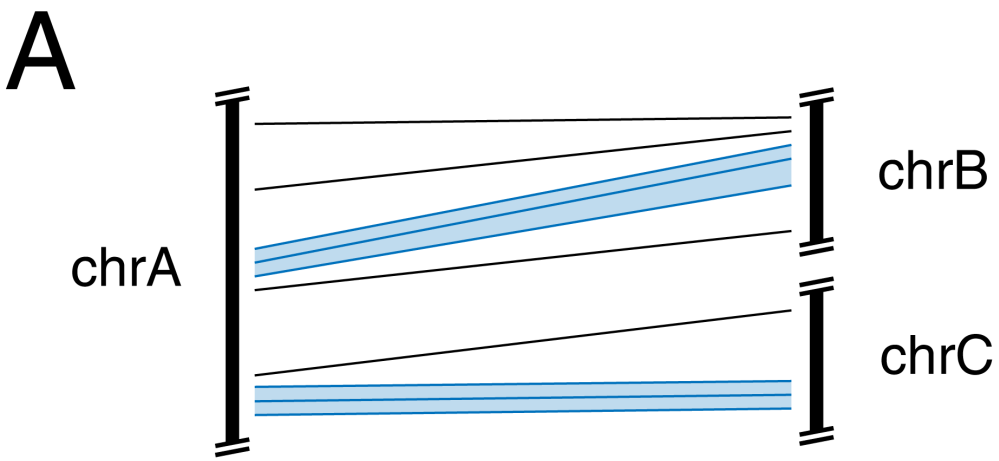
Data sets with many links (e.g. alignments) likely contain spurious entries which aren't biologically meaningful. For example, a single small alignment may not be as interesting as a chain (several with neighbouring start and end positions).

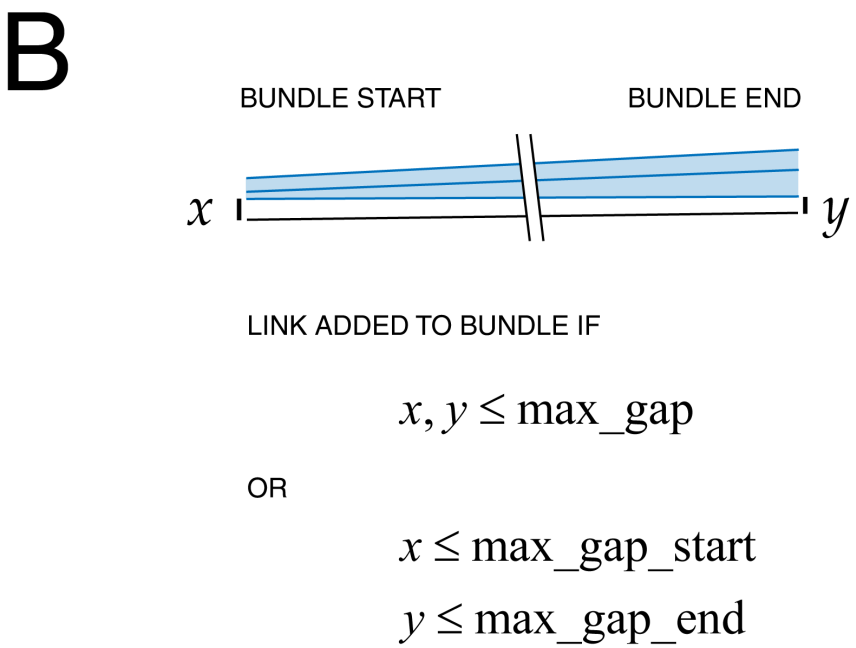
BUNDLING LINKS (OPTIONAL)

If you installed the Circos tools, do this section. If you want to install them, you can get the tools at

http://www.circos.ca/software/download/utilities/

The `bundlelinks` tool (`tools/bundlelinks`) allows you to combine the coordinates of adjacent links, creating links with larger start and end coordinates. You can control the maximum distance between links to bundle, the minimum number of links required to make a bundle, and other parameters. This principle is shown in the figure below.





More information about `bundlelinks` is available at

http://www.circos.ca/documentation/tutorials/utilities/bundling_links

Run the tool on the link file that contains the orthology relationships.

```
> cd ~/circos/data
> cat ortho.links.txt | $CIRCOS/tools/bundlelinks/bin/bundlelinks
    -min_bundle_membership 3
```

How many links are there in the original file? How many bundles did the script create? Try running the script with `-min_bundle_membership 5`, which filters for bundles with at least 5 links. How many bundles were created now?

I've already created the bundle file for you using `-min_bundle_membership 3`.

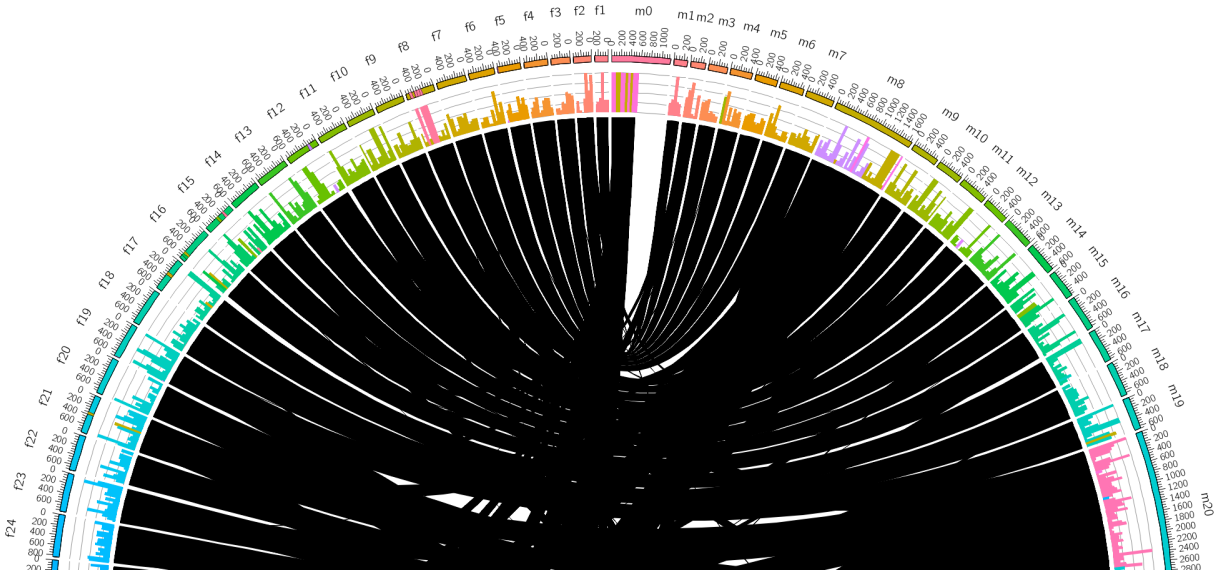
DRAWING LINKS WITH RIBBONS

To draw links as ribbons, instead of lines, you simply put

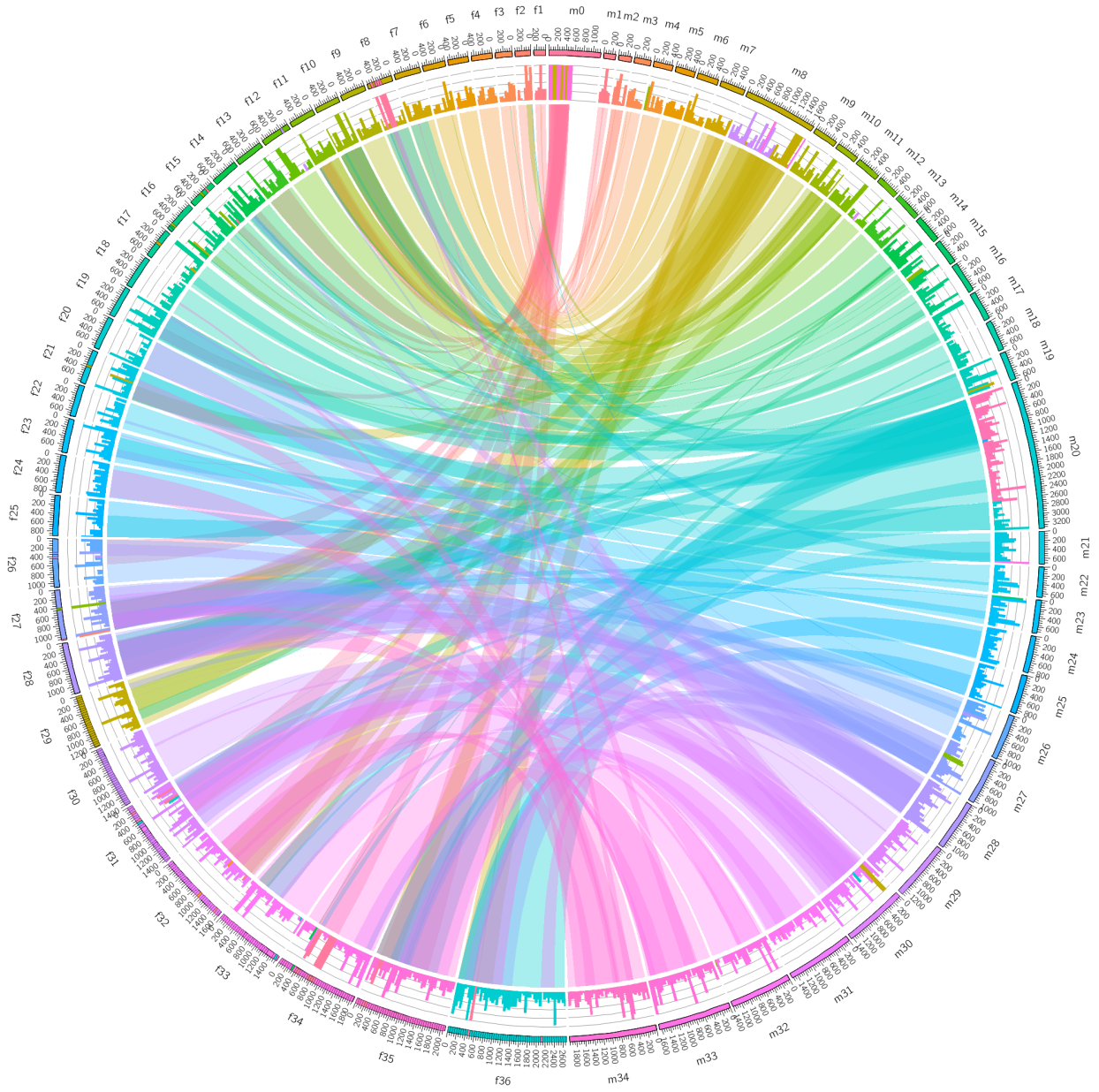
```
ribbon = yes
```

in the `<link>` block. This is only useful if the start and/or end coordinates of the link are large enough to have discernable size in the image. The start and end of the bundle will be proportional to the size of the coordinate, unlike when a link is drawn with a line, whose thickness is uniform.

Create the image for this lesson.



It's hard to see anything. Using a color without transparency makes it very hard to see the thickness of the bundles. Turn the rules on for the `<link>` block and create the image again.



That looks much better, doesn't it? What does the rule do? What is the purpose of the `_a4` suffix.

Let's remove the links that connect chromosomes with the same index. This is the purpose of the second rule. However, because the first rule triggers for all data points, the second rule never gets a chance to run. Add

```
flow = continue
```

to the first rule and create the image again.

Now you're only seeing bundles between different chromosomes (i.e. different index).

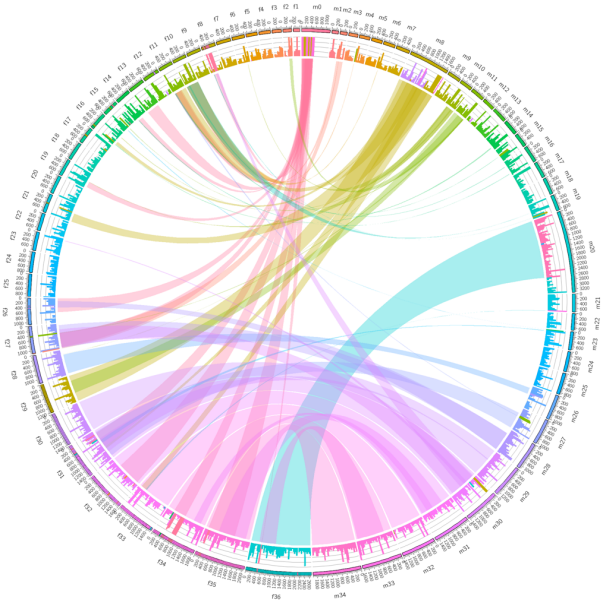
How could the rule blocks be changed so that `flow=continue` is not required? Hint: what happens if you change the order of the blocks?

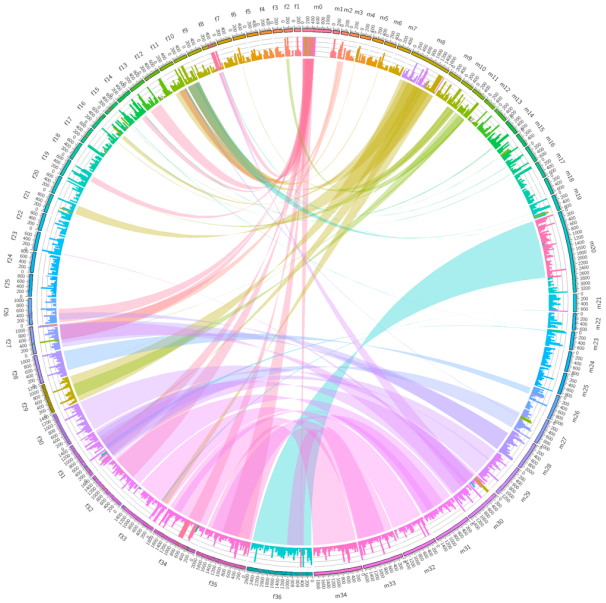
Compare the difference between the images when links are drawn with lines and ribbons.

The quality of the image is completely different. Bundling links results in some links being discarded and the fine detail between others is lost—use bundling when it serves a purpose. Often it's not straightforward to select the parameters for the bundling. Try different combinations and consider the quality and similarity of the genomes you are comparing.

For example, if you increase the minimum number of links in a bundle to 5, you get a much sparser image, as expected.

Notice that the ribbons do not twist. This is why we reversed the orientation of the *L. major* chromosomes. Comment out the line that performs this reversal and create the image again.





When would twists be important to show? Hint: how would an inverted alignment be distinguished? You can force ribbons to always be untwisted by setting

```
flat = yes
```

in the `<link>` block.

Now think about why the image draws the *L. major* chromosomes in reverse order. What would the image look like if the order of chromosomes was 1–36? Make a sketch.

How would you draw thinner links on top? Hint: `z` parameter controls order of drawing while `var(size1)` and `var(size2)` references the size of the start and end coordinate. Add a rule that does this.