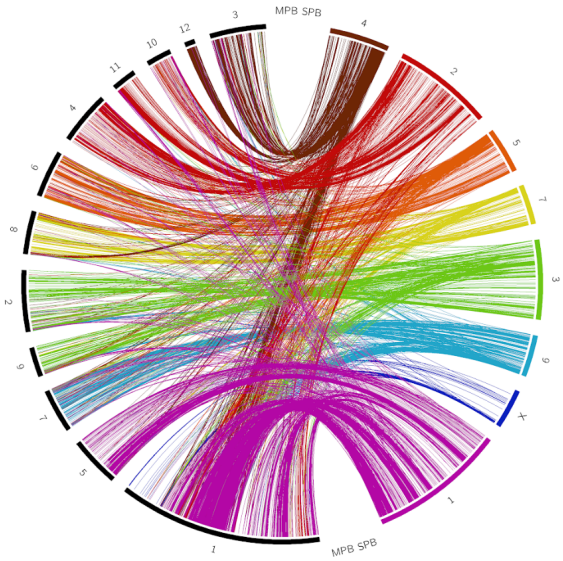
Authors: Megan Copeland - https://github.com/Megan-Copeland
Zachary Hoover - https://github.com/zachhoov
Date: Jun 2, 2023
Purpose: Generating Circos Plots

COMPREHENSIVE GUIDE TO CREATING SYNTENY PLOTS IN CIRCOS

Introduction



Welcome to this concise guide on creating a synteny plot in Circos using alignment data. Synteny plots offer a powerful visual representation of genomic regions, enabling us to explore the conservation and rearrangement of genetic information. We can effectively analyze and communicate complex genomic relationships by employing Circos, a widely-used software package for circular visualization.

In this document, I will walk you through the step-by-step process of generating a synteny plot using alignment data. From installation and data preparation to plot customization, I will cover essential concepts and provide practical instructions. Whether you are a bioinformatician, geneticist, or researcher interested in comparative genomics, this guide will equip you with the necessary skills to utilize Circos effectively for your synteny analysis.

Necessary Tools/Scripts

- Circos
- Circos-tools-0.23
- Minimap2
- RepeatMasker or EDTA
- Seqkit
- Homebrew
- Karyotypify.sh
- Linkify.sh
- Circos.conf
- Colors.conf
- Ideogram.conf
- Link.rules.conf
- Karyotype.and.layout.conf

Files Needed:

- Masked FASTA file for your species of interest (denoted as Species1 in this guide)
- Masked FASTA file for your comparison species (denoted as Species2 in this guide)

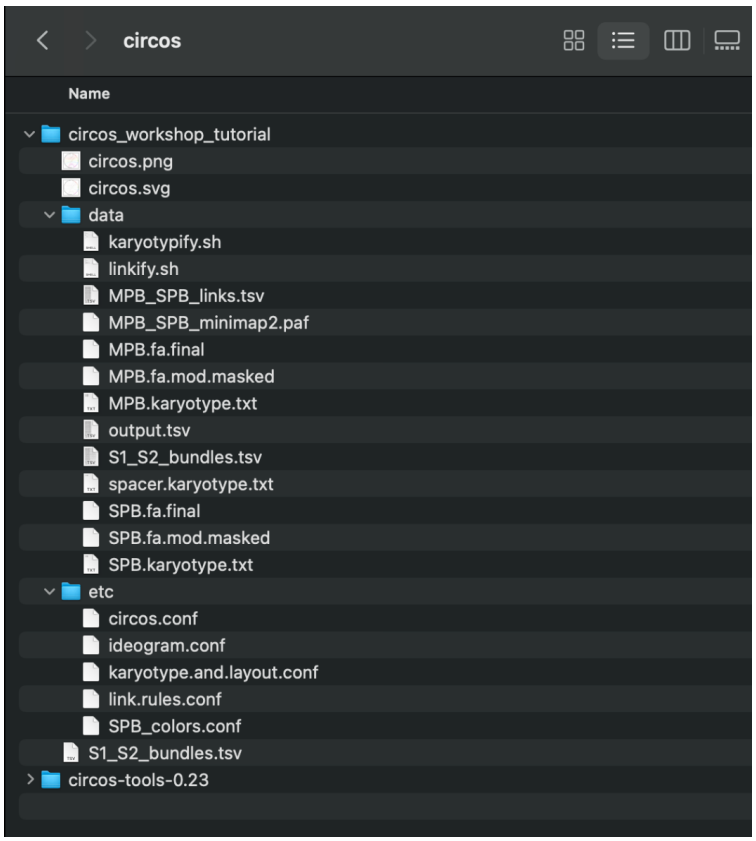
** The sequence IDs should be short names without any spaces. (ex. Ctg1 or scaf1) **

File Structure

Outlined below is the general file structure recommended when plotting with circos. Before you begin this tutorial, set up this structure and place the necessary files in their respective directories, excluding the final FASTA files, Karyotype files, and links/bundles files that are generated during this guide.

- circos
    - Circos tools file
    - Project_Name
        - data
            - Masked FASTA files
            - Final FASTA files
            - Karyotype files
            - Links/bundles files
            - Spacer karyotype
            - Karyotypify.sh script
            - Linkify.sh script
        - etc
            - Circos.conf
            - Ideogram.conf
            - Karyotype.and.layout.conf
            - Link.rules.conf
            - Colors.conf

Image of file structure. The project name is denoted as 'circos_workshop_tutorial':



Before We Begin- Masking Repeats

This guide assumes you are starting with masked FASTA files for your two species. Repetitive elements can cause ambiguity and complexity in sequence alignments, so I recommend using masked files to proceed with your circos plots. Masking these repeats can also reduce visual clutter when plotting. It is essential to know that RepeatMasker requires a Unix system with Perl 5.8.0 or higher installed.

- RepeatMasker: https://www.repeatmasker.org/RepeatMasker/
- EDTA: https://github.com/oushujun/EDTA

Installing Homebrew

Homebrew is an open-source software package management system that allows for easy installation of software packages on MacOS and Linux systems. This guide utilizes homebrew to install circos, seqkit, and minimap2. You can use the link to install Homebrew on your system- https://brew.sh/.

Installing Circos

For ease of maintaining the file structure shown above, I recommend installing circos before any data processing begins. There are many ways to install circos. You can use Homebrew or install directly by downloading the zipped package from the website. I recommend using Homebrew. Let's get started by installing circos using Homebrew. Type and execute the following commands in your terminal:

- brew tap brewsci/bio
- brew install circos

Circos-tools [Optional]

Circos-tools is optional for this guide. It is used to bundle links, which can improve visual clarity depending on how homologous the genomes are of your two species. You can download circos-tools and extract that into your circos directory (http://circos.ca/software/download/tools). You can extract the files inside your circos directory but not in your project folder (see example structure above).To use circos tools later, you will need Perl and quite a few dependencies.

First, check if you have Perl:

- perl -v

If you do not already have Perl, install it using the following command:

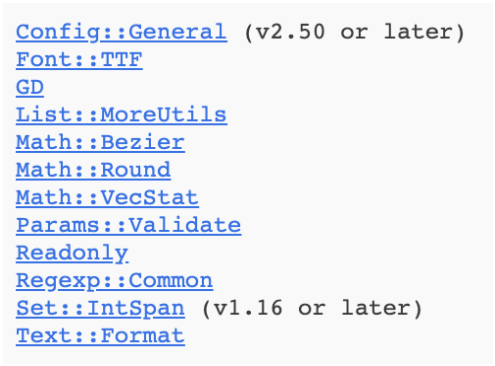
- Brew install perl

Now that you have Perl, we will use cpan to install the necessary dependencies. If you are not using the homebrew installation of circos and instead download the package from the website, you will also need these dependencies. First, open cpan by just typing it into the terminal. Your screen will look like this:

```
(circos) mcc146@client-10-228-173-238 data % cpan
Loading internal logger. Log::Log4perl recommended for better logging
Terminal does not support AddHistory.

To fix enter>  install Term::ReadLine::Perl


cpan shell -- CPAN exploration and modules installation (v2.22)
Enter 'h' for help.

cpan[1]>
```

From here, we can begin installing the dependencies. The image below shows the dependencies required according to the circos website. You can install each one using the following command and just replacing the package name:

- Install Package::Name



To exit the cpan environment, type 'exit' in the terminal and hit enter.

Filtering Contigs/Scaffolds

If you have yet to apply any filtering beforehand, your masked FASTA files will likely contain all of the contigs/scaffolds, including many that may not be necessary for your intended plot. To address this, you can employ the `seqkit` and `awk` commands to filter the sequences based on their sizes. To determine the appropriate size threshold for filtering, you can execute the following command. It will display each contig/scaffold and its corresponding size, arranged in ascending order. Remember to apply this command to both of your masked FASTA files.

Install seqkit with the following command:

- brew install seqkit

Use the following command to move into your data directory:

- cd path/to/data/directory

Use the following command to sort your FASTA file and print the corresponding sequence sizes to determine your cutoff threshold:

- seqkit sort -l Species1.fa.mod.masked | awk '/^>/ {if (seqlen){print seqlen}; print ;seqlen=0;next; } { seqlen += length($0)}END{print seqlen}'
    - **Change files names as needed/desired for this command and all proceeding commands**

You can now create a new file containing only the contigs/scaffolds within your length preference. Use the command below to remove smaller contigs/scaffolds and keep only those bigger than 2Mb (or change to your desired size):

- seqkit seq -m 2000000 Species1.fa.mod.masked > Species1.fa.final
- seqkit seq -m 2000000 Species2.fa.mod.masked > Species2.fa.final

Generating Karyotype Files

The next step involves generating karyotype files for your species. To do this, we need to know the length of each chromosome. Use the commands below on each of your final FASTA finals to print the lengths of each:

- ./karyotypify.sh -g Species1.fa.final -k Species1.karyotype.txt
- ./karyotypify.sh -g Species2.fa.final -k Species2.karyotype.txt

**If this does not work, you may not have the permission to execute the karyotypify.sh file. If this does happen, it will likely happen with the linkify.sh script as well To fix this use the following commands:

- chmod +x karyotypify.sh
- chmod +x linkify.sh

The karyotype files are formatted into seven columns, as seen below, and are not ordered by sequence size. The first two columns will never change. Column 3 is the unique identifier for a given sequence– what it is labeled as in the FASTA files. If you want the ID/label to appear differently on your plot, you can change the value in column 4. Columns 5 and 6 are the start and stop positions of the sequences, respectively, and the last column is the color assigned to each contig/scaffold block in the ideogram.

```
(circos) mcc146@client-10-228-173-238 data % cat SPB.karyotype.txt
chr - ctg699 ctg699 0 12376499 black
chr - ctg700 ctg700 0 13063269 black
chr - ctg701 ctg701 0 12518577 black
chr - ctg702 ctg702 0 14339049 black
chr - ctg703 ctg703 0 18689033 black
chr - ctg704 ctg704 0 24829404 black
chr - ctg705 ctg705 0 31302036 black
chr - ctg706 ctg706 0 42498342 black
(circos) mcc146@client-10-228-173-238 data %
```

You can open the file in a text editor or use nano to change the labels. To use nano, see the command below:

- nano Species1.karyotype.txt

Since we have colors.conf file we are going to call on that designates a color to each contig/scaffold, I change the seventh column to the corresponding contig/scaffold name for that line using an awk command. Use the command below for this step:

- awk '{ if ($7 == "black") { $7 = $3 } print }' Species1.karyotype.txt > temp.txt && mv temp.txt Species1.karyotype.txt

Generating Alignment Data

As mentioned in the introduction, this circos plot generated by this tutorial utilizes alignment data. It is important to note that the best alignment tool for your project may vary depending on your data type (i.e., long-read vs. short-read sequencing data). Minimap2 works well with short and long-read data, so this tutorial will use this tool.

Install minimap2 with the following command:

- brew install minimap2

Align your species genomes:

- minimap2 Species2.fa.final Species1.fa.final > S1_S2_minimap2.paf

Generating & Bundling Links

The lines or bands showing the syntenic regions on a circos plot are often called links. Now that we have our alignment data, we need to generate the links from that to direct circos where each line/band needs to be. We will generate the links file using the linkify.sh script.

Generate your links file with the following command:

- ./linkify.sh -a S1_S2_minimap2.paf -l S1_S2_links.tsv

The links file will follow a seven-column format. The first three columns describe the link's location in Species 1, with column 1 being the contig number and columns 2 and 3 being the start and stop positions, respectively, for that link. Columns 4-6 describe the same thing but for the link's location in Species 2. The last column provides the color value for each connection.

Your links file should look like this:

```
ctg699 2597 3677309 scaf1 59596038 63381769 color=black
ctg699 8465546 11248262 scaf1 51808169 54365555 color=black
ctg699 6172900 8464757 scaf1 54389351 57205288 color=black
ctg699 5321822 6170463 scaf1 58721617 59593737 color=black
ctg699 3691743 4437460 scaf1 57218829 58047941 color=black
ctg699 4442654 5078282 scaf1 58068430 58719823 color=black
ctg699 5079987 5206220 scaf1 63386919 63502296 color=black
ctg699 11888958 11979143 scaf1 51329799 51419726 color=black
ctg699 11290850 11703662 scaf1 51505527 51786758 color=black
ctg699 12019704 12223602 scaf1 51162965 51324843 color=black
ctg699 5261594 5319098 scaf1 63503171 63561402 color=black
```

Like when we colored the ideogram, we can change the color value in the seventh column to reflect the contig value. You color the links based on their location in your desired species. Once again, our colors.conf file, which will have colors assigned to each contig/scaffold it will call on when we run circos.

You can use the following command to change the color value in the seventh column to the name of the contig given in the first column:

- awk '{gsub(/^[[:space:]]+|[[:space:]]+$/, ""); $7 = "color=" $1; print}' S1_S2_links.tsv > final_links.tsv

Your file should now look like this:

```
ctg699 2597 3677309 scaf1 59596038 63381769 color=ctg699
ctg699 8465546 11248262 scaf1 51808169 54365555 color=ctg699
ctg699 6172900 8464757 scaf1 54389351 57205288 color=ctg699
ctg699 5321822 6170463 scaf1 58721617 59593737 color=ctg699
ctg699 3691743 4437460 scaf1 57218829 58047941 color=ctg699
ctg699 4442654 5078282 scaf1 58068430 58719823 color=ctg699
ctg699 5079987 5206220 scaf1 63386919 63502296 color=ctg699
ctg699 11888958 11979143 scaf1 51329799 51419726 color=ctg699
ctg699 11290850 11703662 scaf1 51505527 51786758 color=ctg699
ctg699 12019704 12223602 scaf1 51162965 51324843 color=ctg699
ctg699 5261594 5319098 scaf1 63503171 63561402 color=ctg699
```

[OPTIONAL]

The following step is optional and depends on your preference. You can bundle the links together based on their distance from each other using the circos tools. If you have many links between your species, it may be beneficial to bundle the links together to reduce visual clutter. This step has no set parameters because it will vary depending on your preference, so you may have to play around with the flags to find your desired look.

You will want to adjust the two parameters -max_gap and -min_bundle_membership. The -max_gap flag merges links together into a bundle based on proximity. Simply put, it specifies the maximum allowed gap between neighboring links for them to be considered as part of the same bundle. The second flag, -min_bundle_membership, filters out bundles based on the number of links contained within it. It sets a threshold for the minimum number of links that must be present within a certain distance (controlled by the max_gap parameter) in order to form a bundle.

To generate your bundles file, we will call on circos tools while still working out of our data directory, so you will need to define the path to the bundlelinks script (yellow). You can use the following command:

- ../../circos-tools-0.23/tools/bundlelinks/bin/bundlelinks -links output.tsv -max_gap 1000000 -min_bundle_membership 11 > S1_S2_bundles.tsv
    - Note the PATH to bundlelinks will vary depending on your file structure. If you adhere to the file structure outlined above, then it should be the same.

The file will generally look like this:

```
ctg699 2597 12362647 scaf1 51010179 63617481 nlinks=43,bsize1=11245567,bsize2=11513599,bidentity1=0.909832,bidentit
y2=0.913248,depth1=0,depth2=0,color=ctg699
ctg700 11342169 12692317 scaf1 43129305 45891880 nlinks=14,bsize1=6897,bsize2=6294,bidentity1=0.005108,bidentity2=0
.002278,depth1=1,depth2=3,color=ctg700
ctg700 11886270 13047763 scaf1 32453594 35908338 nlinks=19,bsize1=6141,bsize2=6558,bidentity1=0.005287,bidentity2=0
.001898,depth1=2,depth2=2,color=ctg700
ctg700 2826 5029127 scaf7 3007821 7537047 nlinks=165,bsize1=3484453,bsize2=3890769,bidentity1=0.693244,bidentity2=0
.859036,depth1=0,depth2=0,color=ctg700
ctg700 670946 2073162 scaf7 20043 1069811 nlinks=30,bsize1=947314,bsize2=968401,bidentity1=0.675583,bidentity2=0.92
2490,depth1=1,depth2=1,color=ctg700
```

Configuration & Design Customization

By this point, we have generated all the necessary data files and can proceed with editing our configuration files. There are five main configuration files that we will have to edit (see above section ‘Necessary tools/scripts’). You can use cd to switch from your data directory to your etc directory containing the config files.

- Cd ../etc

<u>Circos.conf</u>

Circos utilizes a central configuration file that acts as the core driver for the image generation process and imports additional configuration files that define global color schemes and font settings, among other parameters. In this section, I will break down the parts of the circos.conf file that will be used more commonly in customization. The best way to see each line's effects is to edit that line, save the file, and run circos to

```
show_links      = yes
show_highlights = yes
show_text       = no
show_heatmaps   = no
show_scatter    = no
show_histogram  = no

<<include karyotype.and.layout.conf>>

### links
<links>

show          = conf(show_links)
ribbon        = no
flat          = yes
radius        = 0.99r
bezier_radius = 0r
color         = black_a5
thickness     = 1.5

<link>
file = data/MPB_SPB_links.tsv
<<include link.rules.conf>>
</link>

</links>

<<include ideogram.conf>>

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

see what changes occurred. I will also include links from the circos website that review some of the parameters.

The first chunk of code in this file will never change, assuming you are creating a standard circos plot as seen at the beginning of this guide. The next line is calling on our karyotype and layout configuration file. If you renamed this file, you will need to change this line to reflect that.

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

<colors>
<<include SPB_colors.conf>>
</colors>

<<include etc/housekeeping.conf>>
data_out_of_range* = trim
```

The next chunk of code shows customization options for the links, see http://circos.ca/documentation/tutorials/links/basic_links/ for more details. The key part in this section is the ribbon option, which allows for you to have the links visualized in a ribbon type as opposed to individual lines. Click on the following link to see the difference between the two options: http://circos.ca/documentation/tutorials/links/ribbons/images. In this links section, we are going to call on our links file seen in the line “file = data/MPB_SPB_links.tsv.” If you generate a bundles file or have a different name, you will need to change the file name here.

After the links section, we call on multiple files, including the ideogram, colors/fonts, species-specific colors, and the housekeeping configuration files. Once again, if you changed the name of any of these files, you will need to change it here to reflect that.

Link.rules.conf

If you are creating a relatively simple circos plot, such as the one generated by this guide, you will not need to mess with the link rules configuration file. You can create a decision chain in this file by adding rule blocks, which can change your data points' format. The file we will be using is shown below:

```
(circos) mcc146@client-10-228-173-238 etc % cat link.rules.conf
<rules>
<rule>
condition  = var(size1) < 1
show       = no
</rule>
</rules>
```

We will not be messing with this file, but I recommend looking at the circos website for more specific details (http://circos.ca/tutorials/lessons/quick_start/links_and_rules/).

Kayotype.and.layout.conf & spacer.karyotype.txt

The karyotype and layout configuration file incorporates the species karyotype files, along with the spacer karyotype file, from our data directory and specifies the arrangement of

contigs/scaffolds on the ideogram. Before delving into this file's specifics, let's first discuss the spacer file in the data directory.

The spacer file serves the purpose of visually distinguishing the ideograms of our two species by introducing spaces between them. While this file is optional, if you prefer not to have gaps between your genomes, you can remove it from the karyotype and layout configuration file by deleting its reference.

The structure of the spacer file aligns with that of the species' karyotype files. However, there are specific modifications: the third column has been modified to indicate that these entries represent spacers, while the fourth column contains the text that will be displayed on the ideogram. In this case, the text denotes the genomes being represented. The seventh column is set to white to maintain the space between the two ideograms. Here's an example of how the spacer file can be structured:

```
(circos) mcc146@client-10-228-173-238 data % cat spacer.karyotype.tx
chr - spacer1 MPB 0 2500000 white
chr - spacer2 MPB 0 2500000 white
chr - spacer3 SPB 0 2500000 white
chr - spacer4 SPB 0 2500000 white
```

Below, we present an example of a karyotype layout file. In the first line, we reference the relevant karyotype files located in our data directory. The second line instructs Circos to arrange the contigs/scaffolds in the plot according to the karyotype files' order. Alternatively, you have the flexibility to manually order them using the "chromosome_order" setting, as demonstrated in the last line. To achieve the most optimal visualization, I recommend experimenting with the order until you achieve your desired plot. It may require some trial and error, but obtaining the perfect visualization will be worth the effort.

```
(circos) mcc146@client-10-228-173-238 etc % cat karyotype.and.layout.conf

karyotype = data/SPB.karyotype.txt,data/MPB.karyotype.txt,data/spacer.karyotype.txt

chromosomes_order_by_karyotype = yes
chromosomes_units              = 1000000
chromosomes_display_default    = yes
chromosomes_order = spacer2,spacer3,ctg703,ctg705,ctg702,ctg701,ctg704,ctg700,ctg699,ctg706,spacer4,spacer1,scaf1,scaf5,scaf7,scaf9,sca
f2,scaf8,scaf6,scaf4,scaf11,scaf10,scaf12,scaf3,
```

Ideogram.conf

The "ideogram.conf" file comprises configuration settings for the ideogram component in the Circos visualization. Here's an example of its contents. While I won't delve into the specifics of each line, this file governs various aspects of the ideogram, such as thickness, spacing between ideogram blocks, and label sizes. I suggest experimenting with these parameters to achieve an optimal visualization for your plot until you reach the desired look.

The image below shows the contents of the example ideogram configuration file:

```
(circos) mcc146@client-10-228-173-238 etc % cat ideogram.conf
<ideogram>

show = yes

<spacing>

default = 5u

# The " " are required because
# the trailing /> is interpreted as a block end.
# Otherwise, you can also use a space
# <pairwise /hs/ /hs/ >

<pairwise "/hs/ /hs/">
spacing = 0.5r
</pairwise>

</spacing>

thickness         = 25p

#stroke_thickness = 1
#stroke_color     = black

fill           = yes
fill_color     = black

radius         = 0.90r
show_label     = yes
label_font     = default
label_radius   = dims(ideogram,radius_outer) + 50p
label_size     = 40p
label_parallel = yes

show_bands            = yes
fill_bands            = yes
band_stroke_thickness = 0
band_stroke_color     = black
band_transparency     = 4

</ideogram>
```

Colors.conf

As previously mentioned, during the generation of the ideogram and links files, the "colors.conf" file assigns colors to each contig/scaffold of your target species that Circos will reference. The structure of this file is straightforward, with each line beginning with the contig/scaffold name, followed by an equals sign, and then the corresponding color represented by a HEX code. Here's an example to illustrate the format:

```
(circos) mcc146@client-10-228-173-238 etc % cat SPB_colors.conf
ctg703 = 6E2605
ctg705 = C20A0A
ctg702 = DF5B08
ctg701 = D8D21D
ctg704 = 6CC717
ctg700 = 22A5C9
ctg706 = B307A5
ctg699 = 0921BD
```

To easily visualize the flow of your color palette, you can leverage the Coolors website. This convenient platform enables you to create a color palette by utilizing up to 10 colors without requiring a paid subscription. By adjusting the color for each column, you can fine-tune your palette. Once satisfied with your selection, copy the corresponding HEX codes into your "colors.conf" file. Here's an example palette that you can modify to align with your personal preferences:
https://coolors.co/2d3047-419d78-e0a458-ffdbb5-c04abc-c65ac2-cb69c8-d077cd-d483d2-d88ed6

**Since this guide is generating a simple circos plot that displays synteny, these files will not contain a lot. However, the more complex of a plot you are generating, the same applies to the contents of these five files. You can refer to the example plot and scripts in the circos package (downloaded from the website) to see more.

Running Circos

To run circos and generate your plots, you must work out of the project directory. In the file structure given at the beginning of this tutorial, the project directory is labeled as "circos_workshop_tutorial." You can use the cd command to change your current directory location. For example, if you are in the etc directory, you can use the command below to go out one level:

- Cd ../

Once you are in the correct directory, you can run the circos command. You must define the path to your circos configuration file (highlighted in yellow). See the example below:

- circos

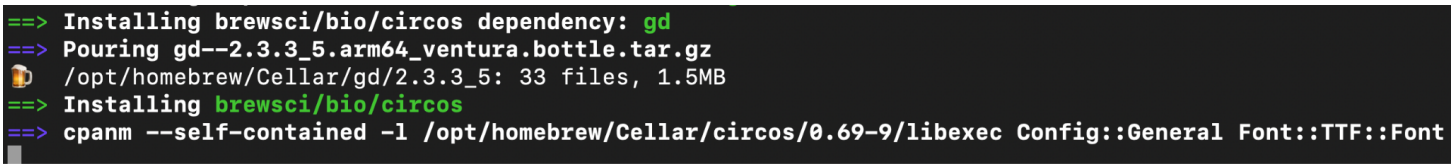
Your plots should be in your working directory if the command worked. If not, see the 'Troubleshooting' section below for some common errors.

Troubleshooting

Here are a couple options to try if you experienced issues with running the circos command:

1. Error: command not found
    a. This can be caused because either circos failed to install or homebrew was not added to your PATH. You can use 'echo $PATH to see if homebrew was successfully added. If so, re-run the two commands to install circos. Double-check that no errors occur.
2. Missing dependencies GD, Polyline…
    a. Sometimes when downloading with brew, it will not install all the dependencies initially. If this is the case, you can try using the following commands:
        i. brew update
        ii. brew install GD (It may say that GD is already installed. If so, follow the command to reinstall it.)
        iii. brew reinstall GD (Often this will be enough, and it will begin to reinstall GD along with the other necessary dependencies). As shown in the

image below, you should see GD being installed followed by the other dependencies, ex. Config::General, Font::TTF::Font, etc



b. Try running the circos command again.