MCScan (Python-version) Workflow on SCELSE Hadley-HPCC

by Mark Chan and Wee Soon Keong

5 October 2020

Using genomic islands from *Burkholderia thailandensis* AW34-19p (pink), AW34-19pp (purple) and E264 (ATCC) chromosome 1 as examples.

---

1. Generating `.gff` and `.faa` input files

1.1 Identifying genomic islands using cumulative GC skew plot in python

For each bacteria strain, I generated one `.fa` output containing concatenated sequences of all detected genomic islands. This can be input into a prokaryotic gene annotation software such as Prokka to generate `.gff` and `.faa` files:

> ***Important***:
>
> If you have multiple nucleotide sequences in separate `fa` files, make sure to concatenate all `fa` files before using Prokka annotation. Do not run Prokka on multiple `fa` files and concatenate the `.gff` output. This is because conversion of `.gff` files to `.bed` format later on would only retain the first table in the `.gff` file. (If you concatenated `fa` files at the start, all annotations of separate sequences will be in the first/only/same table)
>
> **tl;dr** if you have multiple sequences in separate `fasta` files, concatenate them first before running on Prokka

1.2 Obtaining files

Possible to annotate sequences using Prokka from Galaxy server. Easier to view output files and is relatively fast.

Download general feature format `.gff` and amino acid fasta `.faa` files. Only these two file formats would be needed for MCScan (Python).

2. Importing data

2.1 Connecting to SCELSE Hadley HPCC

Connect to hadley.scelse.ntu.edu.sg using SSH client (PuTTY):

> IP address: 172.21.32.46
>
> Port: 22

Enter login credentials when prompted.

2.2 Create a working directory

```
#go to directory
cd /gpfs1/scratch/keong/mark

#create directory and enter it
mkdir -p gi/bt/pinkpurple
cd gi/bt/pinkpurple
##The -p flag tells the mkdir command to create the main directory first if it
doesn't already exist
```

2.3 Upload files using FileZilla

Login to FileZilla and access our working directory using remote site

```
/gpfs1/scratch/keong/mark/gi/bt/pinkpurple
```

Drag and drop the following files into the `pinkpurple` directory:

> pinkchr1.gff
>
> pinkchr1.faa
>
> purplechr1.gff
>
> purplechr1.faa

Check for successful upload of files

```
ls
```

3. Converting Prokka output files to MCScan input format

You can do this on Galaxy too, they have these file conversion functions available. Also allows you to view both files nicely in a tabular format.

Alternatively, we can convert the files locally as well (only takes a few seconds)

3.1 Converting `gff` files to `bed` format

```
#loading jcvi module
module load jcvi/1.2.3

#converting .gff to .bed
python -m jcvi.formats.gff bed --type=CDS --key=ID pinkchr1.gff -o pinkchr1.bed
python -m jcvi.formats.gff bed --type=CDS --key=ID purplechr1.gff -o
purplechr1.bed

##Variables
##--type: Feature type (third column of gff file; in this case its CDS.
Alternatively you can input mRNA for transcriptomics analysis)
##--key: Prefix of column 9 'Group' (in this case its ID)
##pinkchr1.gff is my input file in .gff format
##-o is output
##pinkchr1.bed is my output file name in .bed format
```

> Example `gff` file in a tabular format. *Note --type=CDS and --key=ID*:

Seqname	Source	Feature	Start	End	Score	Strand	Frame	Group
##gff-version 3								
##sequence-region 1 1 3545013								
1	Prodigal:002006	CDS	118	552	.	+	0	ID=MBPJGMNK_00001;inference=ab initio predi

> During the format conversion you should see something like this:

```
15:24:07 [base] Load file `AWCtg1.gff`
15:24:07 [gff] Extracted 3086 features (type=CDS id=ID)
```

```
#check for successful .bed file format conversion
ls
```

3.2 Reformatting amino acid `fasta` files

```
#converting .faa to .cds
python -m jcvi.formats.fasta format pinkchr1.faa pinkchr1.cds
python -m jcvi.formats.fasta format purplechr1.faa purplechr1.cds

#then, converting .cds to .pep
cp pinkchr1.cds pinkchr1.pep
cp purplechr1.cds purplechr1.pep

#check for successful .pep file format conversion
ls
```

At this point, we would only need `.pep` and `.bed` files to proceed. We can remove all the other files.

```
#removing .gff .faa and .cds files
rm *.gff *.faa *.cds
```

4. Running pairwise synteny search on MCScan (Python)

4.1 Creating PBS script

```
#create job submission script using nano text editor
nano submit.sh
```

Edit the following script accordingly and copy it into the terminal. Save and exit:

```
#!/bin/bash

# Run this as "qsub submit.sh"

# Set the resource requirements; 1 CPU, 5 GB memory and 5 minutes wall time.
#PBS -l select=1:ncpus=1:mem=8GB
#PBS -q std
#PBS -l walltime=00:05:00

# Name your Job
#PBS -N jcvi-test

# Load the necessary application

export PATH=$PATH:/gpfs1/scratch/keong/softwares/last-1080/src
```

```
module load jcvi/1.2.3



# Run your program.
cd ${PBS_O_WORKDIR}

python -m jcvi.compara.catalog ortholog pinkchr1 purplechr1 --no_strip_names --
dbtype prot --cscore=0.99
```

4.2 Submitting job to Hadley

```
#submit PBS script as a job
qsub submit.sh
```

4.3 Checking MCScan output files

```
#check for files
ls

#check for files and file sizes
ll
```

You should see MCScan output files in the same folder. The `.pdf` file size is a good indicator of whether the analysis worked and LaTeX was able to successfully generate the plot. You can visualize the plot by downloading the `pdf` file onto your local computer.

The `.last` file is raw LAST output, `.last.filtered` is filtered LAST output, `.anchors` is the seed synteny blocks (high quality), `.lifted.anchors` recruits new additional anchors to form the final synteny blocks.

5. Macrosynteny visualization

First, we generate a `.simple` file, which is a more succinct form than the `.anchors` file:

```
#generating .simple file from .anchors file
python -m jcvi.compara.synteny screen --minspan=30 --simple
pinkchr1.purplechr1.anchors pinkchr1.purplechr1.anchors.new
```

Aside from `.bed` and synteny files we already have, we need to prepare 2 more additional files:

1. `seqids` file, which tells the plotter which genomic islands (or chromosomes/genes) to include, and
2. `layout` file, which tells the plotter where to draw what.

5.1 Generating `seqids` file

For this analysis, pinkchr1 contains 8 genomic islands (named GI1-GI8) whereas purplechr1 contains 9 (named GI1-GI9). I want to look at all the genomic islands, so I will be including all of them. Note the order of the names in the `pinkchr1.purplechr1.anchors` file.

```
#generating seqids file
nano seqids
```

According to the naming order of the `anchors` file, the first line should contain pinkchr1 genomic islands (8 of them) and second line purplechr1 genomic islands (9 of them). Copy the following into the terminal. Save and exit:

```
GI1,GI2,GI3,GI4,GI5,GI6,GI7,GI8
GI1,GI2,GI3,GI4,GI5,GI6,GI7,GI8,GI9
```

5.2 Generating `layout` file

```
#generating layout file
nano layout
```

It is useful to know that the whole canvas is 0-1 on both x and y-axes. The first three columns specify the position of the track. Then rotation (in degrees), colour, label, vertical alignment (`va`), and then the genome `.bed` file. Track 0 is pinkchr1 (python starts reading from 0 #justpythings), and track 1 is purplechr1. We can leave the colour blank for now (it will be default).

The next stanza specifies what edges to draw between the tracks. `e, 0, 1` asks to draw edges between track 0 and 1, using information from the `.simple` file.

Copy the following into the terminal. Save and exit:

```
# y, xstart, xend, rotation, color, label, va,  bed
 .7,     .1,    .8,       0,      , BtPINKchr1, top, pinkchr1.bed
 .5,     .1,    .8,       0,      , BtPURPLEchr1, top, purplechr1.bed
# edges
e, 0, 1, pinkchr1.purplechr1.anchors.simple
```

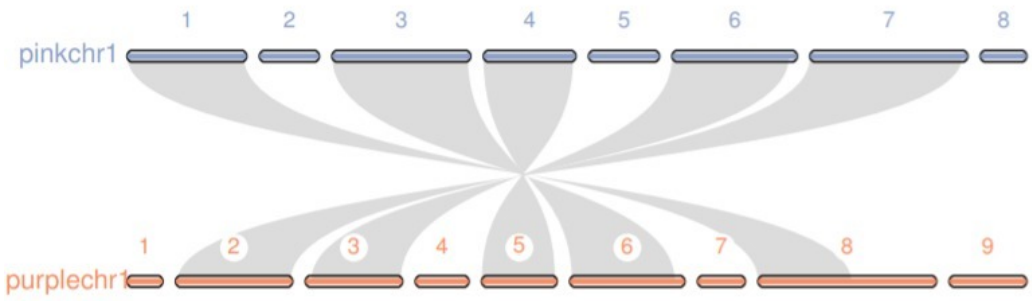
> ***Important***:
>
> Make sure there is no space ( ) at the start of each line below `# edges`.

5.3 Plotting figure

Now we have our input files ready, we can plot.

```
#plotting
python -m jcvi.graphics.karyotype seqids layout
```

This generates a `karyotype.pdf` that we can download and visualize:

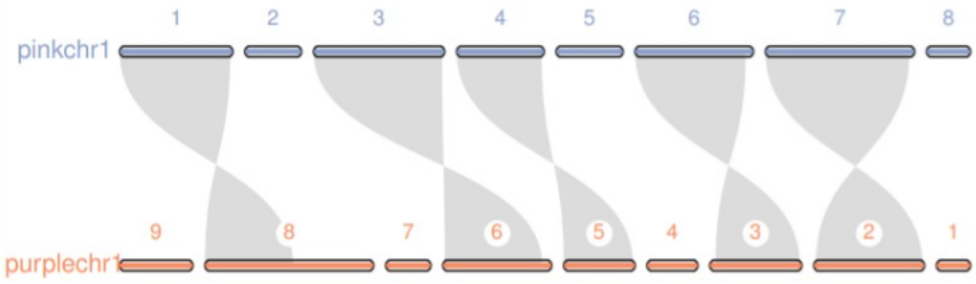


Further customization is possible. We can play around with the positions, colour, labels etc. in the `layout` file. Instead of a M&S candy design, we can invert the order of the chromosomes on purplechr1 by editing the `seqids` file, and it should generate a figure that is visually more appealing:

Big brains?

```
GI1,GI2,GI3,GI4,GI5,GI6,GI7,GI8
GI9,GI8,GI7,GI6,GI5,GI4,GI3,GI2,GI1
```

At least that was what i thought.



Seems like I still need to go back to my raw fasta files and invert the sequence. Sad.

6. Macrosynteny getting fancy

6.1 Highlighting a specific block

We can highlight a specific gene or even a particular set of genes by editing the `.simple` file. Download the `.simple` file and edit it using a text editor (e.g. Sublime Text).

Note that each line in the `.simple` file is a synteny block, with start and stop of pinkchr1, then start and stop of purplechr1, final two columns are score and orientation. Mine looks something like this:

```
1   GPAOEDFI_00001  GPAOEDFI_00044  OFBCPLGC_00281  OFBCPLGC_00328  45  -
2   GPAOEDFI_00045  GPAOEDFI_00096  OFBCPLGC_00195  OFBCPLGC_00248  51  -
3   GPAOEDFI_00097  GPAOEDFI_00131  OFBCPLGC_00155  OFBCPLGC_00193  35  -
4   GPAOEDFI_00158  GPAOEDFI_00202  OFBCPLGC_00081  OFBCPLGC_00127  44  -
5   GPAOEDFI_00205  GPAOEDFI_00261  OFBCPLGC_00020  OFBCPLGC_00077  57  -
```

We can highlight each synteny block individually with any colour abbreviation supported by matplotlib:

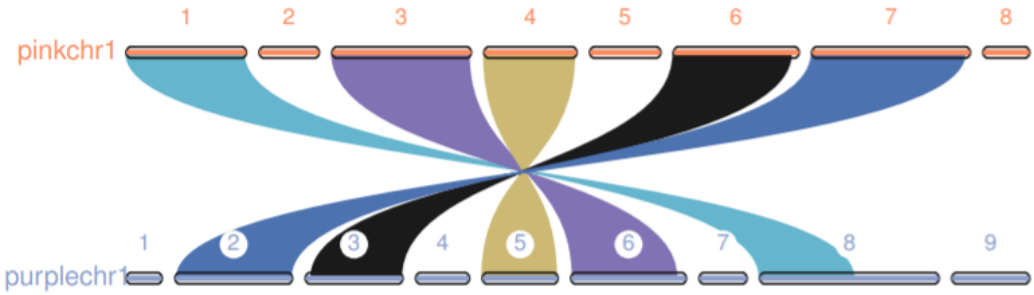
```
1   c*GPAOEDFI_00001    GPAOEDFI_00044  OFBCPLGC_00281  OFBCPLGC_00328  45  -
2   m*GPAOEDFI_00045    GPAOEDFI_00096  OFBCPLGC_00195  OFBCPLGC_00248  51  -
3   y*GPAOEDFI_00097    GPAOEDFI_00131  OFBCPLGC_00155  OFBCPLGC_00193  35  -
4   k*GPAOEDFI_00158    GPAOEDFI_00202  OFBCPLGC_00081  OFBCPLGC_00127  44  -
5   b*GPAOEDFI_00205    GPAOEDFI_00261  OFBCPLGC_00020  OFBCPLGC_00077  57  -
```

Character	Colour
b	blue
g	green
r	red
c	cyan
m	magenta
y	yellow
k	black
w	white

Remember to save the updated changes before uploading the `.simple` file into FileZilla.

Lets plot:

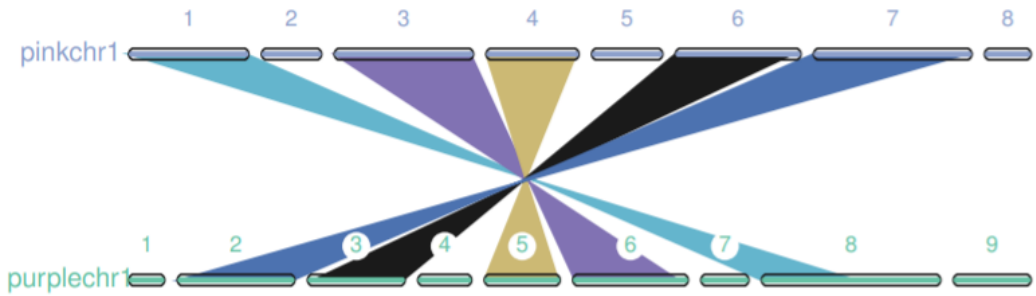
```
#plotting
python -m jcvi.graphics.karyotype seqids layout
```



6.2 Changing shade styles

Instead of Bezier curves, we can change the shade area into lines:

```
#changing shade style to line
python -m jcvi.graphics.karyotype seqids layout --shadestyle=line
```



6.3 Adding more genomes

We can also add in as many genomes as we want. However, do note that if we want to compare between any two genomes, we would have to generate a `.simple` file between those two genomes. For example, `GenomeA.GenomeB.anchors.simple` and `GenomeB.GenomeC.anchors.simple` would allow us to compare between Genomes A and B as

well as Genomes B and C, but not Genomes A and C.

For this example, I added in another sequence containing genomic islands from *Burkholderia thailandensis* E264 Chromosome 1. Similarly, I start with two input files: `e264chr1.faa` and `e264chr1.gff` and I generate the corresponding `e264chr1.bed` and `e264chr1.pep` files.

Lets compare pinkchr1 vs. purplechr1 vs. e264chr1:

```
#since we already have pinkchr1 vs. purplechr1, lets generate a pairwise synteny
between purplechr1 and e264chr1
#you can submit a pbs script for this, by editing the script from Section 4.1
python -m jcvi.compara.catalog ortholog purplechr1 e264chr1 --no_strip_names --
dbtype prot --cscore=0.99

#generating .simple file from .anchors file
python -m jcvi.compara.synteny screen --minspan=30 --simple
purplechr1.e264chr1.anchors purplechr1.e264chr1.anchors.new
```

6.3.1 Updating `seqids` file

Next, we would need to update our `seqids` file.

```
#editing seqids file
nano seqids
```

`e264chr1` has 6 genomic islands, named GI1-GI6. Save and exit:

```
GI1,GI2,GI3,GI4,GI5,GI6,GI7,GI8
GI1,GI2,GI3,GI4,GI5,GI6,GI7,GI8,GI9
GI1,GI2,GI3,GI4,GI5,GI6
```

6.3.2 Updating `layout` file

Let's update the `layout` file next:

```
#editing layout file
nano layout
```

We can change the plot coordinates and rotation in the file according to how we want our final figure to be. For now, let us proceed with this:

```
# y, xstart, xend, rotation, color, label, va,  bed
 .7,     .2,    .9,       0,      , pinkchr1, top, pinkchr1.bed
 .5,     .2,    .9,       0,      , purplechr1, top, purplechr1.bed
 .3,     .2,    .7,       0,      , e264chr1, bottom, e264chr1.bed
# edges
e, 0, 1, pinkchr1.purplechr1.anchors.simple
e, 1, 2, purplechr1.e264chr1.anchors.simple
```

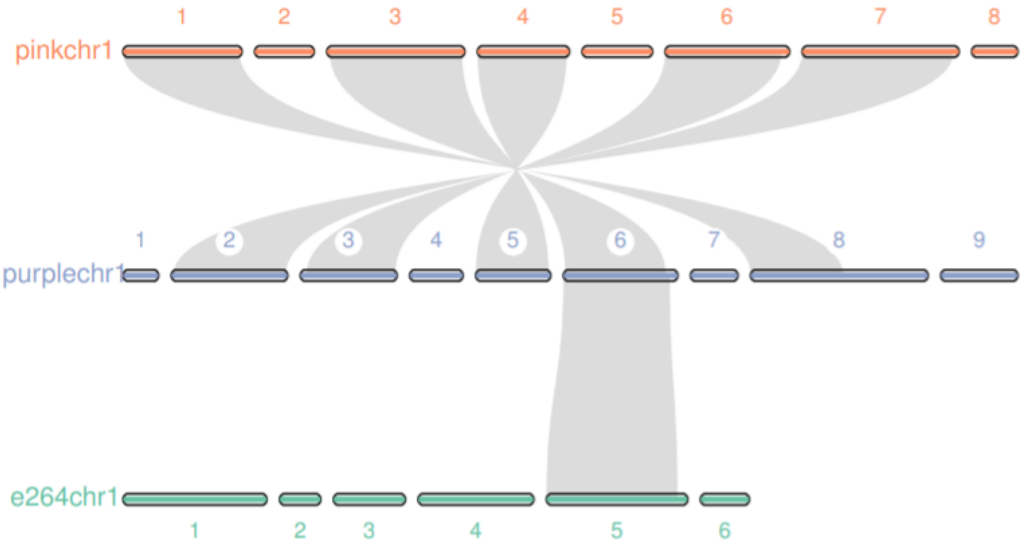
> ***Important***:
>
> Make sure there is no space ( ) at the start of each line below `# edges`.

6.3.3 Plotting figure

Now we have our input files ready, we can plot.

```
#plotting
python -m jcvi.graphics.karyotype seqids layout
```

This generates another `karyotype.pdf` that we can download and visualize:



6.3.4 Highlighting a specific block across multiple alignments

Looking at the plot, we notice that pinkGI3 ≈ purpleGI6' ≈ e264GI5'. Let us highlight this specific block. Remember that jcvi compares genomes in pairs, so we would have to download both `pinkchr1.purplechr1.anchors.simple` and `purplechr1.e264chr1.anchors.simple` and update the corresponding blocks with a specific colour that we want.

`pinkchr1.purplechr1.anchors.simple`:

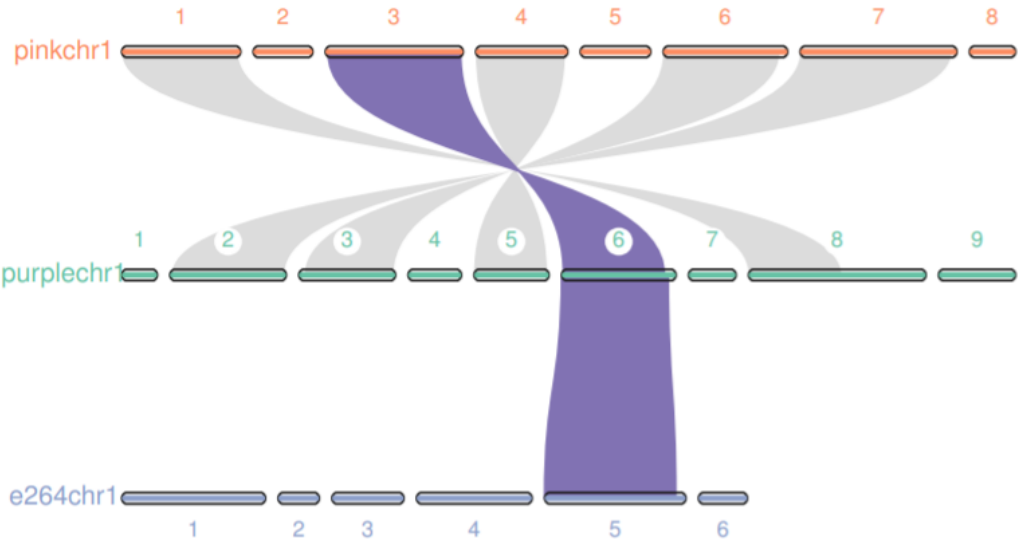
```
GPAOEDFI_00001  GPAOEDFI_00044  OFBCPLGC_00281  OFBCPLGC_00328  45  -
m*GPAOEDFI_00046    GPAOEDFI_00096  OFBCPLGC_00195  OFBCPLGC_00247  50  -
GPAOEDFI_00097  GPAOEDFI_00131  OFBCPLGC_00155  OFBCPLGC_00193  35  -
GPAOEDFI_00158  GPAOEDFI_00202  OFBCPLGC_00081  OFBCPLGC_00127  44  -
GPAOEDFI_00205  GPAOEDFI_00261  OFBCPLGC_00020  OFBCPLGC_00077  57  -
```

`purplechr1.e264chr1.anchors.simple`:

```
m*OFBCPLGC_00194    OFBCPLGC_00252  FAOFBIEC_00147  FAOFBIEC_00202  53  +
```

Save and import both `.simple` files back into FileZilla. Lets plot:

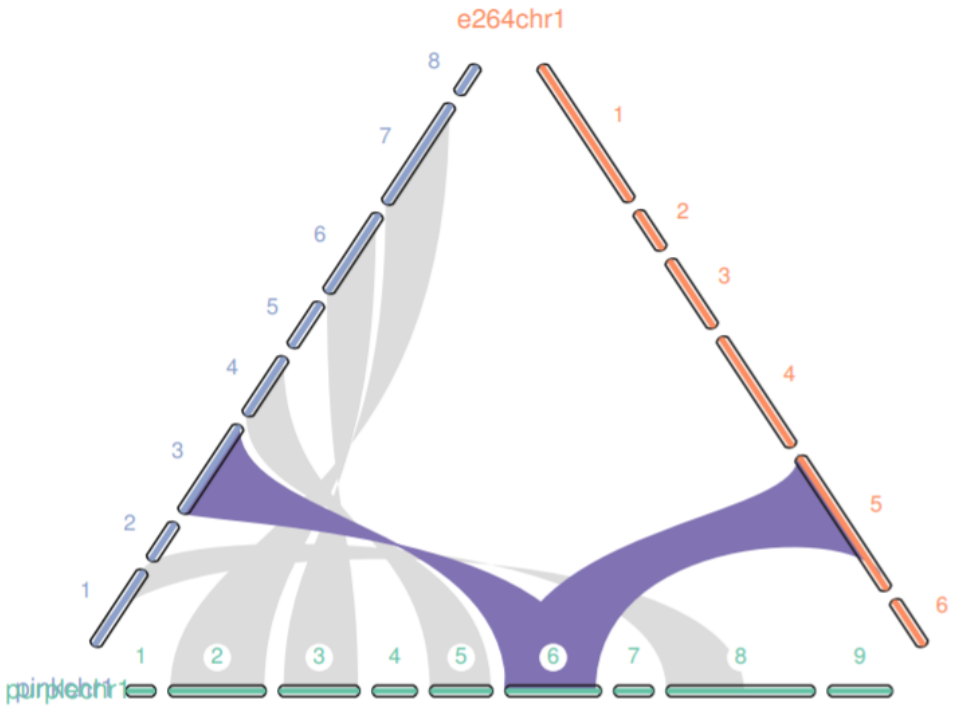
```
#plotting
python -m jcvi.graphics.karyotype seqids layout
```



6.3.5 Rotating alignments

We can explore this function too, which gives us some pretty fancy plots. All we need to do is edit the `layout` file and replot the figure:

```
# y, xstart, xend, rotation, color, label, va,  bed
 .5,   .025,  .625,      60,      , pinkchr1, top, pinkchr1.bed
 .2,    .2,    .8,       0,      , purplechr1, top, purplechr1.bed
 .5,   .375,  .975,     -60,      , e264chr1, top, e264chr1.bed
# edges
e, 0, 1, pinkchr1.purplechr1.anchors.simple
e, 1, 2, purplechr1.e264chr1.anchors.simple
```



However, this figure only shows synteny between pinkchr1 and purplechr1, purplechr1 and e264chr1. Let us add in pinkchr1 and e264chr1:

```
#running pairwise synteny between pinkchr1 and e264chr1
python -m jcvi.compara.catalog ortholog pinkchr1 e264chr1 --no_strip_names --
dbtype prot --cscore=0.99

#generating corresponding .simple file
python -m jcvi.compara.synteny screen --minspan=30 --simple
pinkchr1.e264chr1.anchors pinkchr1.e264chr1.anchors.new
```

Let us update the `.simple` file to highlight our block of interest. Remember to save changes before uploading the file into Filezilla:

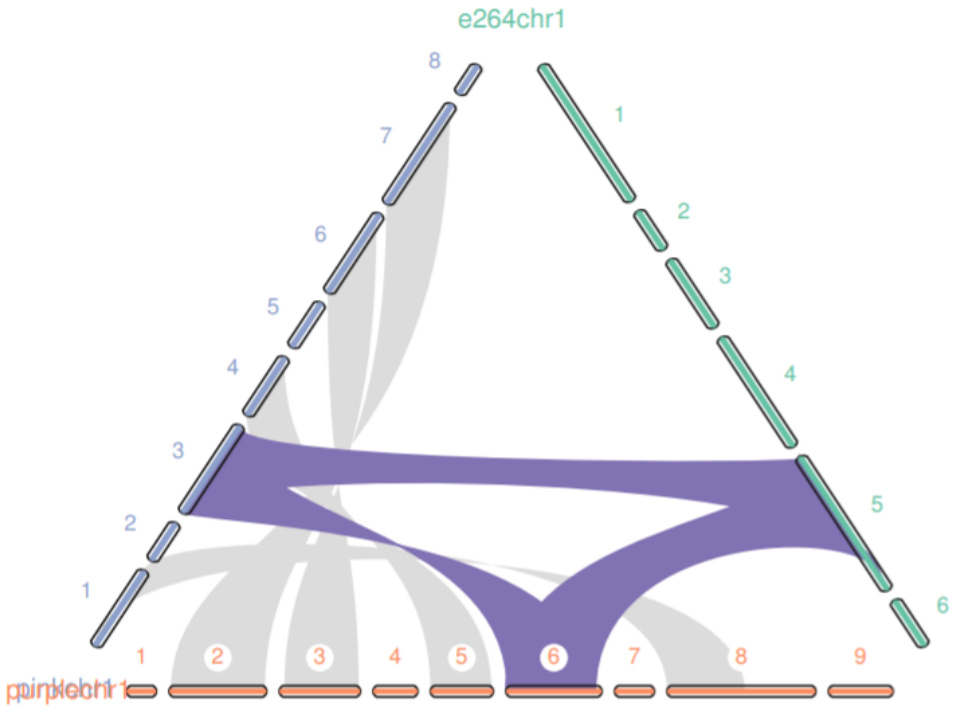
```
1    m*GPAOEDFI_00048    GPAOEDFI_00096  FAOFBIEC_00148  FAOFBIEC_00198  49  -
```

And update the `layout` file accordingly:

```
# y, xstart, xend, rotation, color, label, va,  bed
 .5,    .025,  .625,      60,       , pinkchr1, top, pinkchr1.bed
 .2,      .2,    .8,       0,       , purplechr1, top, purplechr1.bed
 .5,    .375,  .975,     -60,       , e264chr1, top, e264chr1.bed
# edges
e, 0, 1, pinkchr1.purplechr1.anchors.simple
e, 1, 2, purplechr1.e264chr1.anchors.simple
e, 0, 2, pinkchr1.e264chr1.anchors.simple
```

Finally, let's plot:

```
#plotting
python -m jcvi.graphics.karyotype seqids layout
```



> *'it's quite a feat to choose dataset until can draw this HAHAHA', SK 2020*

7. Microsynteny visualization

It is also possible for us to zoom in and visualize pairwise synteny at the gene-level. Microsynteny shows the matching regions along aligned gene models but first we need to compute the layout for gene-level matchings:

```
#computing layout
python -m jcvi.compara.synteny mcscan pinkchr1.bed
pinkchr1.purplechr1.lifted.anchors --iter=1 -o pinkchr1.purplechr1.il.blocks
```

> **Note**:
>
> `--iter=1` means we are extracting *one* best region matching each pinkchr1 region. If you look at the resulting `pinkchr1.purplechr1.il.blocks` file, it contains pinkchr1 as the first column and purplechr1 as the second column. If the option `--iter` is set to 2, there will be 2 purplechr1 regions, and so on. Specifically, this will be useful to plot regions resulted from genome duplications. Example of `il.blocks` file below:

```
1   GPAOEDFI_00001  OFBCPLGC_00328
2   GPAOEDFI_00002  OFBCPLGC_00327
3   GPAOEDFI_00003  OFBCPLGC_00326
4   GPAOEDFI_00004  OFBCPLGC_00325
5   GPAOEDFI_00005  OFBCPLGC_00324
6   GPAOEDFI_00006  OFBCPLGC_00323
7   GPAOEDFI_00007  OFBCPLGC_00322
8   GPAOEDFI_00008  OFBCPLGC_00321
9   GPAOEDFI_00009  OFBCPLGC_00320
10  GPAOEDFI_00010  OFBCPLGC_00297
11  GPAOEDFI_00011  OFBCPLGC_00296
12  GPAOEDFI_00012  OFBCPLGC_00295
```

7.1 Selecting plot regions

The `pinkchr1.purplechr1.il.blocks` contains lots of local regions. We can choose any arbitrary region to visualize:

```
#selecting region to include in plot
head -25 pinkchr1.purplechr1.il.blocks > blocks
```

> Still figuring out context of head, -25. Plot first 25 genes starting from the front?

7.2 Generating `layout` file

Similarly, we will need a `layout` file just like in macrosynteny plots. Create a `blocks.layout` file:

```
#creating blocks.layout file
nano blocks.layout
```

Copy the following into the terminal. (ha = horizontal alignment, va = vertical alignment). Save and exit:

```
# x,   y, rotation,   ha,     va,   color, ratio,           label
0.5, 0.6,        0, left, center,       m,     1,         pinkchr1
0.5, 0.4,        0, left, center, #fc8d62,     1,       purplechr1
# edges
e, 0, 1
```

> ***Important***:

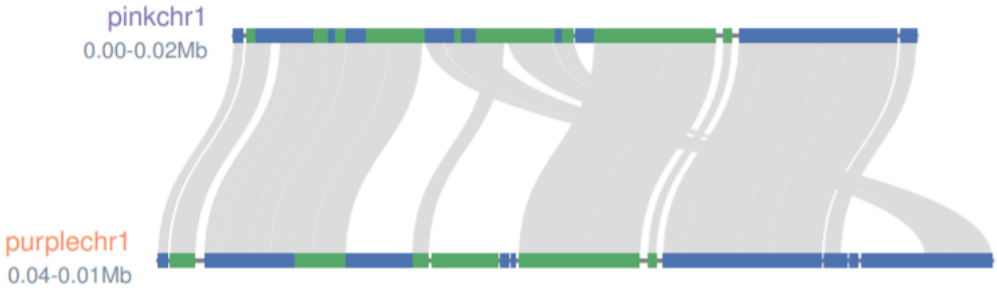
> Make sure there is no space ( ) at the start of each line below `# edges`.

7.3 Generating local synteny plot

```
#concatenating .bed files
cat pinkchr1.bed purplechr1.bed > pinkchr1_purplechr1.bed
```

Now, we are ready to plot:

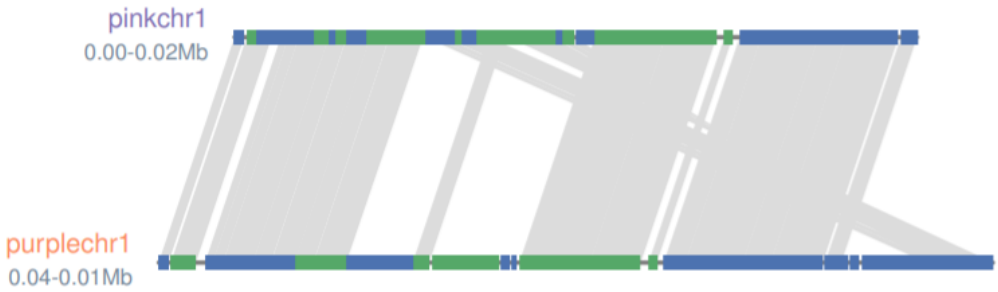
```
#plotting
python -m jcvi.graphics.synteny blocks pinkchr1_purplechr1.bed blocks.layout
```



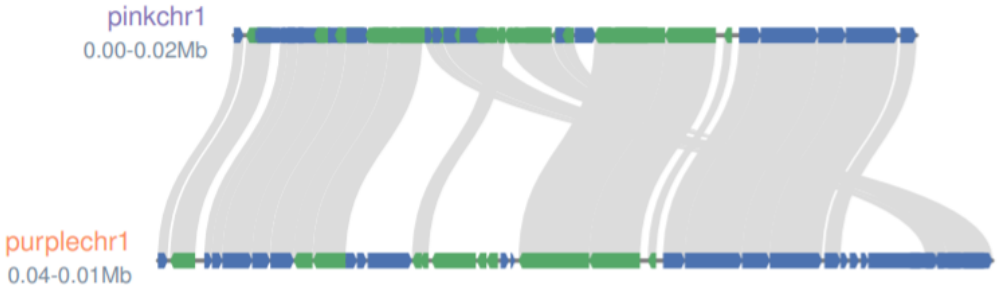
7.4 Changing plot styles

Similar to macrosynteny plots, we can have alternative shade styles:

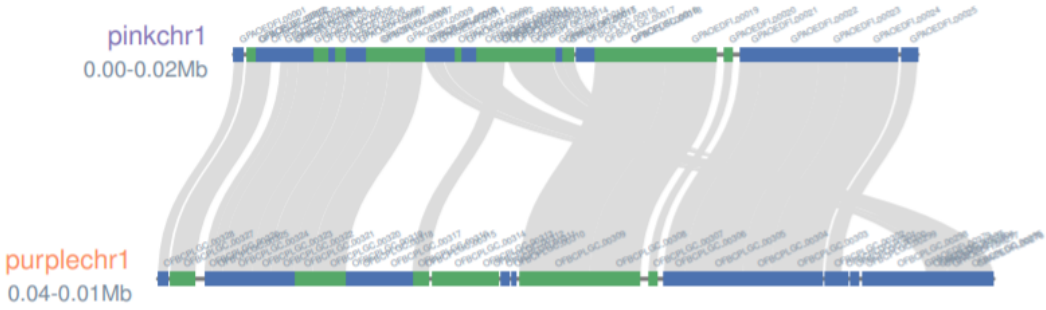
```
#changing shade style to line
python -m jcvi.graphics.synteny blocks pinkchr1_purplechr1.bed blocks.layout --
shadestyle=line
```



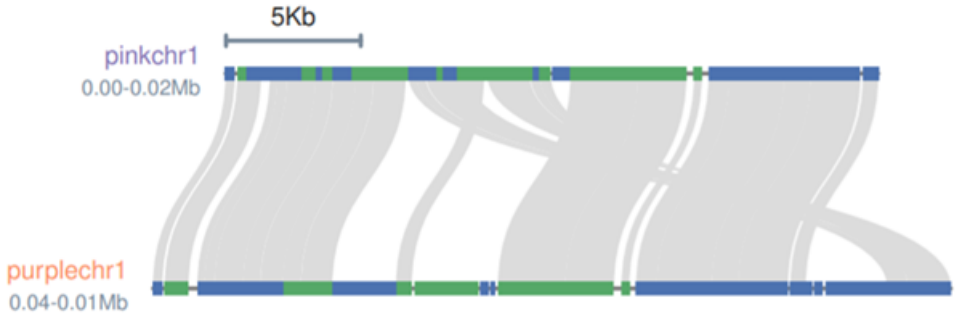

```
#alternative arrow glyph style
python -m jcvi.graphics.synteny blocks pinkchr1_purplechr1.bed blocks.layout --
glyphstyle=arrow
```




```
#adding gene labels
python -m jcvi.graphics.synteny blocks pinkchr1_purplechr1.bed blocks.layout --
genelabelsize 4
```




```
#adding scalebar
python -m jcvi.graphics.synteny blocks pinkchr1_purplechr1.bed blocks.layout --
scalebar
```



8. Microsynteny getting fancy

Similar to macrosynteny plots, we can also include more that two matching regions. Let's make a `blocks` file with the three genomes, using pinkchr1 as the reference, and aligning purplechr1 and e264chr1 separately to pinkchr1.

We already did pink vs. purple, so all that is left is to do the same for e264:

```
#computing layouts
python -m jcvi.compara.synteny mcscan pinkchr1.bed
pinkchr1.e264chr1.lifted.anchors --iter=1 -o pinkchr1.e264chr1.il.blocks
```

8.1 Combining `blocks` and selecting plot regions

Next, we can use the following command to combine our `blocks` files:

```
#combining pink vs purple and pink vs e264 blocks files
python -m jcvi.formats.base join pinkchr1.purplechr1.il.blocks
pinkchr1.e264chr1.il.blocks --noheader | cut -f1,2,4,6 > pinkchr1.blocks
```

We can download `pinkchr1.blocks` and visualize it, which looks something like this (notice 3 columns):

```
GPAOEDFI_00001  OFBCPLGC_00328  .
GPAOEDFI_00002  OFBCPLGC_00327  .
GPAOEDFI_00003  OFBCPLGC_00326  .
GPAOEDFI_00004  OFBCPLGC_00325  FAOFBIEC_00125
GPAOEDFI_00005  OFBCPLGC_00324  .
GPAOEDFI_00006  OFBCPLGC_00323  .
GPAOEDFI_00007  OFBCPLGC_00322  .
GPAOEDFI_00008  OFBCPLGC_00321  .
GPAOEDFI_00009  OFBCPLGC_00320  .
GPAOEDFI_00010  OFBCPLGC_00297  FAOFBIEC_00129
GPAOEDFI_00011  OFBCPLGC_00296  FAOFBIEC_00128
GPAOEDFI_00012  OFBCPLGC_00295  FAOFBIEC_00127
```

Next, let's select our region of interest to plot:

```
#selecting region to include in plot
head -25 pinkchr1.blocks > blocks2
```

8.2 Generating `layout` file

Let's create a new layout file called `blocks2.layout`.

```
#creating blocks2.layout file
nano blocks2.layout
```

Copy the following into the terminal. Save and exit:

```
# x,   y, rotation,     ha,     va,   color, ratio,           label
0.5, 0.6,        0, center,    top,        ,     1,         pinkchr1
0.3, 0.4,        0, center, bottom,        ,   0.5,       purplechr1
0.8, 0.4,        0, center, bottom,        ,   0.5,         e264chr1
# edges
e, 0, 1
e, 0, 2
```

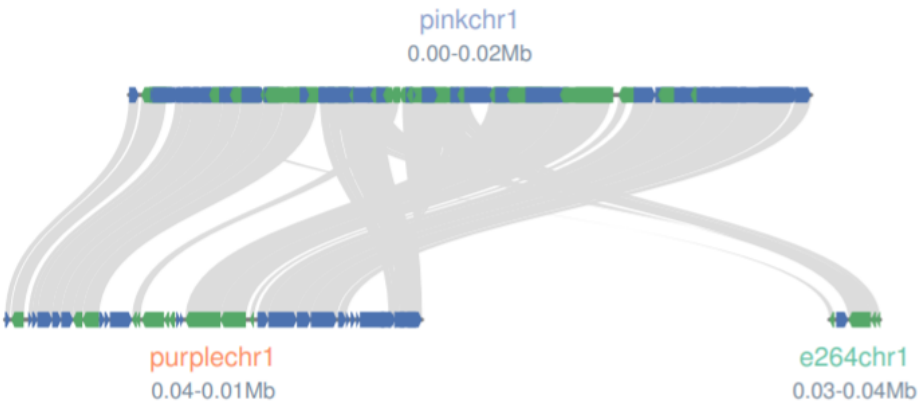
> ***Important***:
>
> Make sure there is no space ( ) at the start of each line below `# edges`.

8.3 Generating local synteny plot with multiple regions

```
#concatenating .bed files
cat pinkchr1.bed purplechr1.bed e264chr1.bed > pinkchr1_purplechr1_e264chr1.bed
```

Now, we are ready to plot:

```
#plotting
python -m jcvi.graphics.synteny blocks2 pinkchr1_purplechr1_e264chr1.bed
blocks2.layout --glyphstyle=arrow
```


pinkchr1
0.00-0.02Mb
purplechr1
0.04-0.01Mb
e264chr1
0.03-0.04Mb