*Dr. Ryan F. Seipke, University of Leeds, April 2020*

**On programmatically draw genes using R**

After spending a decade and a half drawing PowerPoint/Keynote schematics of biosynthetic gene clusters (BGCs) for manuscripts and presentations, I finally decided to invest time in learning how to programmatically draw genes using R.

Most of us are self-taught and if you are like me, then you have always found R a little bit challenging to work despite the impressive documentation available! So I thought I would share my workflow in case others wanted to embark on a similar journey of discovery.

Below I describe a workflow for drawing a BGC.

**Step 1: Generate a tab-delimited gene table.**

*There are of course a variety of ways to do this, below is just one strategy I came up with if you do not feel like manually entering gene coordinates for a large number of genes.*

**1:** Download the Genbank file for your BGC, for example this may be from the from the MIBiG database (https://mibig.secondarymetabolites.org).

**2:** Open the Genbank file in Artemis (https://www.sanger.ac.uk/science/tools/artemis).

**3:** Convert the Genbank file into a GFF file using 'Save Entry As GFF' (note, you will receive warnings that header information will be lost, but it is not important).

**4:** Open your Command Prompt and use the Unix tool `grep` to pull out lines with 'CDS' in them.

For instance at the command prompt:

```
$ grep 'CDS' file.gff > CDSs.txt
```

**5:** Open the `CDSs.txt` file in Excel and use manual manipulation in combination with the text-to-columns tool to generate a sheet that looks like the below. Copy and paste the tab-delimited gene table into a simple plain text editor (e.g. Notepad, TextWrangler or Brackets; I prefer Brackets) and save. In this example I use the file name `BGC.tab`.

Your gene table should be a tab-delimited file that looks like this:

```
molecule    gene  start end   strand     direction
BGC XYZ     gene1 1617  1     reverse    -
BGC XYZ     gene2 2520  1927  reverse    -
BGC XYZ     gene3 3270  2554  reverse    -
BGC XYZ     gene4 3380  3796  forward    +
```

*Note that if the gene is on the REVERSE strand, then the `start` and `end` coordinates need to reflect this. This has to do with how the R packages gggenes and ggplot2 interprets information.*

**Step 2: Setup up R so you can plot your BGC.**

*I am not an R expert and barely consider myself above the beginner level. However, the below assumes you have some working knowledge of R otherwise you will probably be a little bit lost.*

**1:** Install the latest version of R, I am currently using R 3.6.3 GUI on MacOSX Catalina

**2:** Open R

**3:** Install the `ggplot2` and `gggenes` packages from the CRAN binaries repository using the `Package Installer` located under the `Packages & Data` menu.

**4:** Load ggplot2 and gggenes using the Packager Manager located under the `Packages & Data menu` or by typing `library(ggplot2)` followed by the enter key and then `library(gggenes)` followed by the enter key.

**5:** Import your gene table using this command:

```
genes <- read.table("/Full path/to the location/of your
file/BGC.tab", header=TRUE, sep="\t")
```

**6:** Check to make sure your gene table has been imported by simply typing the word `genes` and hitting the enter key.

This should return the below:

```
> genes
  molecule  gene start  end  strand direction
1  BGC XYZ gene1  1617    1 reverse         -
2  BGC XYZ gene2  2520 1927 reverse         -
3  BGC XYZ gene3  3270 2554 reverse         -
4  BGC XYZ gene4  3380 3796 forward         +
>
```

**Step 3:** Drawing your BGC using `gggenes`.

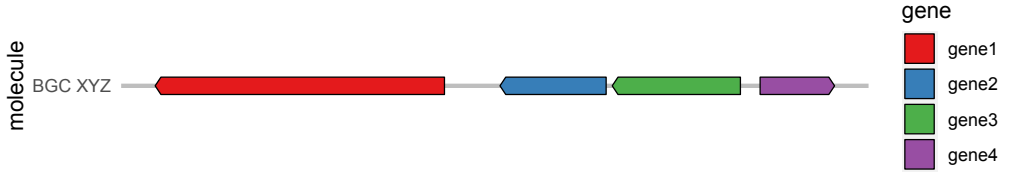
**1:** Read the brief tutorial about `gggenes` available here: https://cran.r-project.org/web/packages/gggenes/vignettes/introduction-to-gggenes.html

**2:** Become familiar with the functionality of `gggenes` by drawing either the four gene example I provided above or the set of genes provided within `gggenes` (note: the file `example_genes` used in documentation commands is loaded when you load the `gggenes` package).

To plot your genes execute the following command, note I have bolded where the parameter that specifies your gene table. Do not worry about the awkward spacing of this command, just copy/paste it in and you should be good to go. Also please note that the bulk of this code originated from the `gggenes` documentation.

```
ggplot(genes, aes(xmin = start, xmax = end, y = molecule, fill =
gene, label = gene)) +
geom_gene_arrow(arrowhead_height = unit(3, "mm"), arrowhead_width =
unit(1, "mm")) +
facet_wrap(~ molecule, scales = "free", ncol = 1) +
scale_fill_brewer(palette = "Set1") +
theme_genes()
```

*Note: A PDF should pop up that you can save. It should look like this the figure below.*



In the `gggenes` documentation, the author shows various aesthetics, but all utilise a pre-set colour palette using `scale_fill_brewer`. If you want **control** over the colour of individual genes (*i.e.* to colour-code them based on putative function, for instance), then you will need to specify this in the plotting command.

Although R accepts HTML hex codes, I find it easier to use colour names. The names of colours can be obtained by executing the command `colors()`. Although this provides a list of colour names, it is obviously fairly useless without knowing what they look like. At the end of this document I have provide colour cheat sheet that I generate using code produced by Dr. Michal Bojanowski from Kozminki University available on his website: http://bc.bojanorama.pl/2013/04/r-color-reference-sheet/.

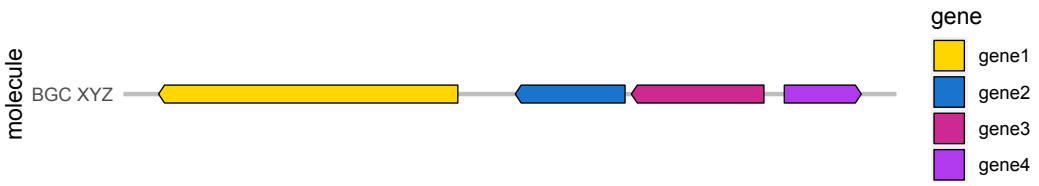
**3:** Plot your BGC specifying individual colours using the below command. I selected these four colours: `gold1`, `dodgerblue3`, `maroon3` and `darkorchid2`.

Before the BGC can be plotted with specified colours, you need to ensure that ggplot2 knows that the gene names in your gene table are unique factors otherwise depending how your genes are named there is a chance that the colours could be incorrectly assigned. To do this you need to execute this command (which I have borrowed from MajoroMask on the tidyverse GitHub space):

```
genes$gene <- factor(genes$gene, levels = unique(genes$gene))
```

Now you can plot using this command:

```
ggplot(genes, aes(xmin = start, xmax = end, y = molecule, fill =
gene, label = gene)) +
geom_gene_arrow(arrowhead_height = unit(3, "mm"), arrowhead_width =
unit(1, "mm")) +
facet_wrap(~ molecule, scales = "free", ncol = 1) +
scale_fill_manual(values=c("gold1", "dodgerblue3", "maroon3",
"darkorchid2")) +
theme_genes()
```




**Troubleshooting:** You may get the below error message the *second* time you try to plot the same imported gene table:

```
Error in UseMethod("depth") :
  no applicable method for 'depth' applied to an object of class "NULL"
```

I do not understand why this happens (because I am not an R expert!), but my workaround for this is to just re-import the gene table and re-specify that gene names are unique factors every time I re-plot it. I hope this tutorial was useful and happy plotting!