# The Grammar of Graphics and Its Use in Evolutionary Biology

*Post 01 - Stat 133, Fall 2017*

*Will Freyman*

*October 29, 2017*

## Introduction

In this post I'll first introduce the concept behind a powerful approach to describe statistical graphics called the *grammar of graphics*. I'll briefly describe the basics of the popular implementation of the grammar found in the `R` package `ggplot2`. Then I demonstrate how the grammar has been extended to handle particularly complex plots commonly used in evolutionary biology. In this last section I'll focus on the `R` packages `ggtree` and `revgadgets`.

Even if you are not interested specifically in evolutionary biology, a primary objective of this post is to demonstrate how the grammar of graphics can generally make data driven science more reproducible and open.

## The grammar of graphics

A common challenge for scientists is crafting graphics that succinctly communicate insight into their data and analysis. One possible solution to this problem was proposed in an influential book by Wilkinson (2005). Wilkinson developed what he called a *grammar of graphics* –a *grammar* is a structure or system of language– so here Wilkinson describes a system of language for defining statistical graphics. This system takes as input numerous layers of data and hierarchically arranges standardized components to construct a graphic.

## The grammar of graphics implemented in `R`

While Wilkinson's book was conceptually influential, it lacked a concrete implementation that would make it widely applicable. The `ggplot2` `R` package (Wickham 2010) was developed so that users of `R` could create plots using the grammar of graphics. The grammar Wickham defined in `ggplot2` has five core components that make up a graphic:

- a default dataset with default aesthetic mappings
- at least one layer of geometric objects and statistical transformation
- a scale for each aesthetic mapping
- a coordinate system
- a faceting assignment

The term *aesthetic mapping* refers to the way the variables in the data are mapped to visual elements of the plot. Let's look at an example from one of our earlier homework assignments.

First, we'll simply load the `ggplot2` package and read in a `csv` file as a `data.frame` object:

```
library(ggplot2)
d = read.csv("data/nba2017-teams.csv")
```

Now we will use the grammar of graphics to build up our plot piece by piece:

```
p = ggplot(d, aes(x=reorder(team, salary), y=salary))
```

Here we created a `ggplot` object and saved it as the variable `p`. We also defined the first core component of our graphic: the default dataset for the graphic was defined as the `d` data frame, and the default aesthetic mapping was constructed using the `aes` function.

```
p = p + ggtitle("NBA Teams ranked by Total Salary") +
        xlab("Teams") +
        ylab("Salary in Millions")
```

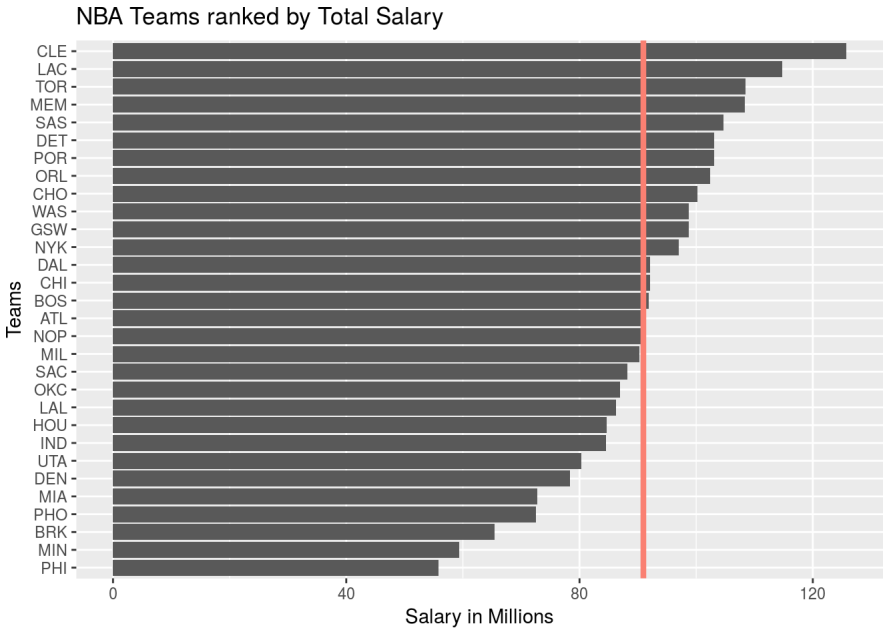
Now we have added various geometric objects to our `ggplot` object, specifically we added the title and x- and y-axis labels.

```
p = p + geom_bar(stat="identity") +
        geom_hline(yintercept=mean(d$salary), linetype=1, color="salmon", size=1.5) +
        coord_flip()
```

Next, we statistically transformed the data using `geom_bar(stat="identity")` and added a bar plot geometric object. In this case, our statistical tranformation is actually to leave the data as it is. If we had used `stat="count"` it would make the height of each bar proportional to the number of cases in each group. We additionally added a horizontal line object using `geom_hline` and transformed the default coordinate system using `coord_flip()`.

Finally, let's view our `ggplot` object:

```
print(p)
```



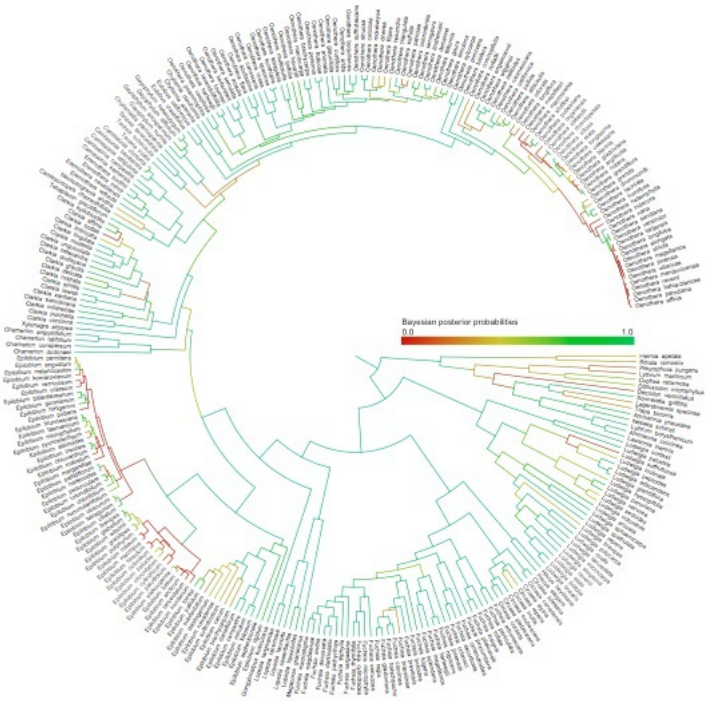

From this example I hope readers can see that the grammar of graphics makes programmatically creating complex plots simple and efficient. One can write a script that generates different plots depending on the output of a number of different steps. This is very important for reproducibility in science, and for making science more open. When computational scientists write code for their research, they can make the code open source and available via public code repositories (for example GitHub). Other scientists can then reproduce the research using the original scripts, and even generate the plots that summarize the results of the research and were published as part of the scientific paper.

My goal here is not to provide a comprehensive `ggplot2` tutorial. For readers interested in more details about using `ggplot2` please check out the official documentation (Wickham and Chang 2017) or any of the excellent tutorials on the web such as this one (IQSS Harvard 2017). Instead, I'd like to focus the rest of this post on how the grammar of graphics has made research in evolutionary biology more reproducible.

# The grammar of graphics in evolutionary biology

I'm a PhD student in evolutionary biology taking this class as a requirement for the Designated Emphasis in Computational Biology. In evolutionary biology we rely on plots of phylogenies –evolutionary trees– that are inferred using genome sequence from extant species. These plots can contain a lot of details such as:

1. the names of extant species at every tip of the tree
2. the divergence times of lineages in millions of years
3. the statistical support of the topology of the tree
4. reconstruction of ancestral character states




Above is an example of a phylogeny from my own research. At the tips of the tree are extant species that were sampled. In the center is the common ancestor of all the sampled species, which was estimated to exist approximately 95 million years ago. The branch lengths are proportianal to the amount of evolutionary time, and each branch is colored by its posterior probability.

These types of plots used to be quite difficult to generate in a reproducible way, however recently `ggplot2` was extended to handle evolutionary trees with the `R` package `ggtree` (Yu et al. 2017). Below is an example of how to use `ggtree`. Each geometry object and `aes()` works just like in `ggtree2`.

First we need to load the `ggtree` library and read in our data file. The data file is in a common bioinformatic format called `NEXUS`, which hold phylogenies and molecular sequence data.

```
library(ggtree)
```

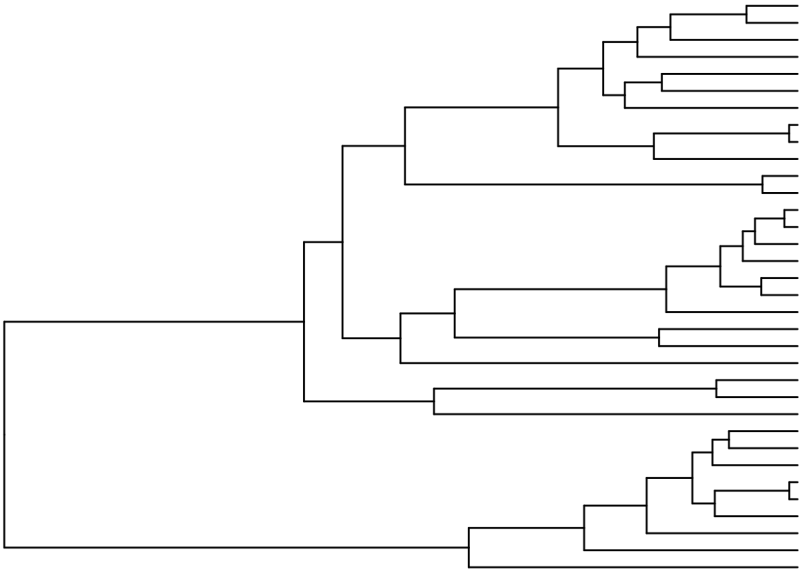
```
## Loading required package: treeio
```

```
## ggtree v1.9.1  For help: https://guangchuangyu.github.io/ggtree
##
## If you use ggtree in published research, please cite:
## Guangchuang Yu, David Smith, Huachen Zhu, Yi Guan, Tommy Tsan-Yuk Lam. ggtree: an R package for visualization a
nd annotation of phylogenetic trees with their covariates and other associated data. Methods in Ecology and Evolut
ion 2017, 8(1):28-36, doi:10.1111/2041-210X.12628
```

```
t = read.beast("data/aristolochia.tree")
```

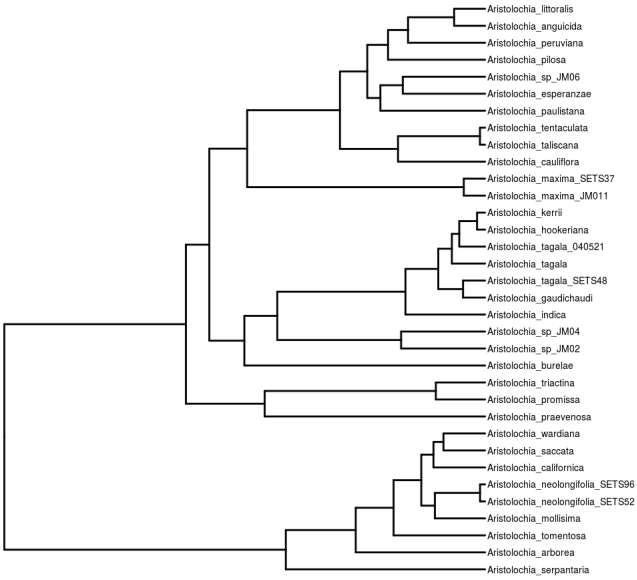
Now let's create our `ggtree` object and print it:

```
p = ggtree(t)
print(p)
```



This is a pretty basic tree. Let's add the species names to the tips. We do this by adding a geometry to the plot and ajusting the coordinate system using the same syntax as `ggplot2`:

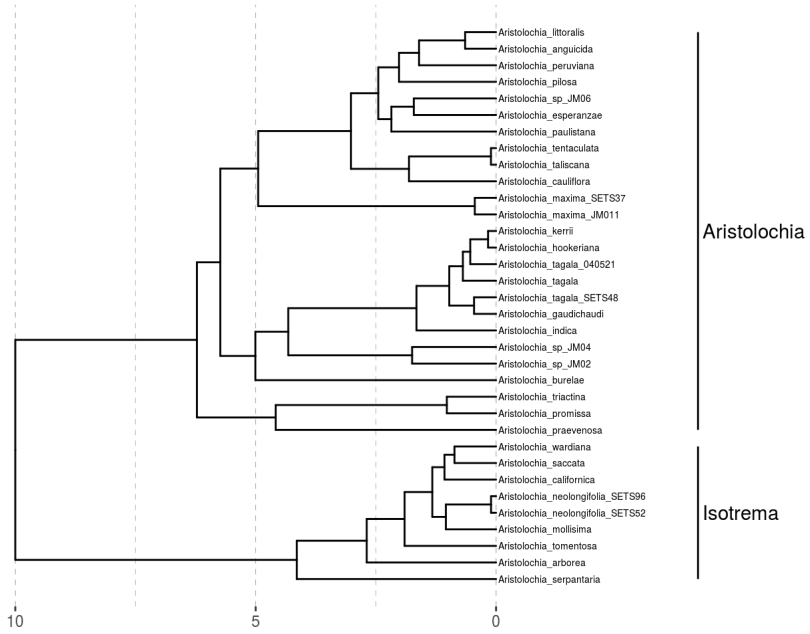
```
p = p + geom_tiplab(size=1.85) + coord_cartesian(xlim = c(0, 16.5))
print(p)
```




Using the grammar of graphics, we can add many more details to our `ggtree` object. For example, we can add the names of entire groups

(clades) and add a time scale that shows when each lineage diverged in millions of years:

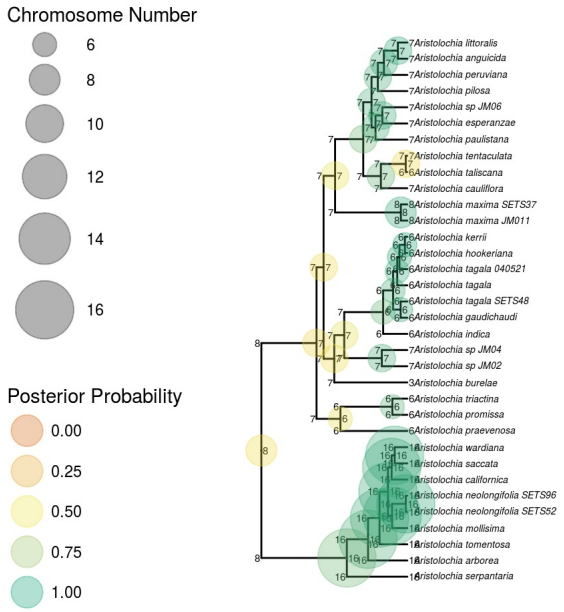
```
p = p + theme_tree2() +
        scale_x_continuous(breaks=c(10, 5, 0), minor_breaks=c(2.5, 7.5), labels=c("0", "5", "10")) +
        theme(panel.grid.major = element_line(color="darkgrey", size=.2, linetype="dashed"),
              panel.grid.minor = element_line(color="grey", size=.2, linetype="dashed"),
              panel.grid.major.y = element_blank(),
              panel.grid.minor.y = element_blank(),
              legend.position="left")
p = p + geom_cladelabel(node=60, label="Isotrema", align=TRUE, offset=4.0, size=2)
p = p + geom_cladelabel(node=36, label="Aristolochia", align=TRUE, offset=4.0, size=2)
print(p)
```




This is super useful to researchers like me who want scripts that make every aspect of my research reproducible. However, in my research I develop new software to analyze how certain traits evolve through time. For example, I have worked on a set of mathematical models that describe how chromosome number (which is a key feature of the higher-order organization of the genome) evolves through time (Freyman and Höhna 2017). These models are implemented in the software `RevBayes` (Höhna et al. 2016), and we needed to develop new data formats to output the results of our analyses. `ggtree` does not handle these new formats, so we have implemented a new `R` package called `RevGadgets` (Höhna et al. 2017) that extends `ggtree` and other `R` packages to process output from `RevBayes`.

First we load `RevGadgets` and use the function `plot_ancestral_states` to generate a `ggtree` object modifed to handle the estimates generated by `RevBayes`:

```
library(RevGadgets)
p = plot_ancestral_states("data/aristolochia.tree",
                          size=0.4,
                          include_start_states=TRUE,
                          summary_statistic="MAPChromosome",
                          tip_label_size=1.85,
                          tip_label_offset=0.4,
                          tip_label_italics=TRUE,
                          node_label_size=2,
                          node_label_nudge_x=0.1,
                          shoulder_label_size=2,
                          alpha=.3)
```



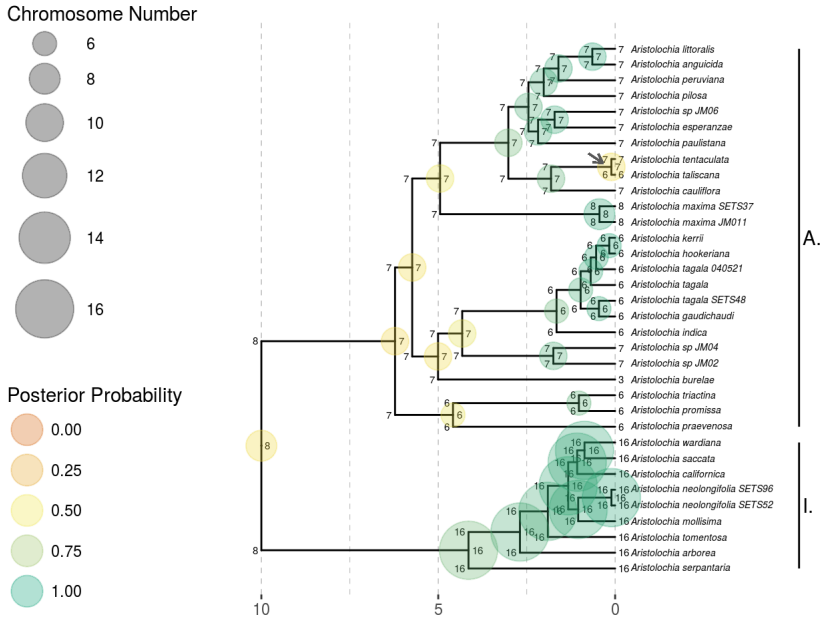

The plot created by default is ok, but it needs some tweaking. We can now use the grammar of graphics and `ggtree` to make more customizations:

```
root_age = 10.0
p = p + theme_tree2() +
    scale_x_continuous(breaks=c(10, 5, 0), minor_breaks=c(2.5, 7.5), labels=c("0", "5", "10")) +
    theme(panel.grid.major = element_line(color="darkgrey", size=.2, linetype="dashed"),
          panel.grid.minor = element_line(color="grey", size=.2, linetype="dashed"),
          panel.grid.major.y = element_blank(),
          panel.grid.minor.y = element_blank(),
          legend.position="left")

# annotate some of the clades
p = p + geom_cladelabel(node=60, label="I.", align=TRUE, offset=5.0, size=2)
p = p + geom_cladelabel(node=36, label="A.", align=TRUE, offset=5.0, size=2)

# add an arrow to show a possible dysploid speciation event
arrow_xend = 9.65
arrow_yend = 26.75
p = p + geom_segment(x=arrow_xend-0.4, y=arrow_yend+0.6, xend=arrow_xend, yend=arrow_yend,
                     size=0.5, color="grey35", arrow=arrow(length=unit(0.2, "cm"), type="open",
                                                           angle=30), inherit.aes=FALSE)

# set the final dimensions
p = p + coord_cartesian(xlim = c(0, 16.5))
print(p)
```




This plot shows the maximum a posteriori ancestral chromosome numbers at every node of the phylogeny, and colors the circles by their posterior probability as well as scaling the size of each circle to the number of ancestral chromosomes. Additionally we added a small grey arrow to highlight the speciation event that was estimated to be driven by the dysploid loss of a single chromosome.

# Conclusion

Even if you do not understand the biological results displayed above (or are simply not interested), hopefully I have succesfully demonstrated how useful the grammar of graphics can when applied to any data driven science. The grammar of graphics as implemented in `R` scripts can be crucial towards making science reproducible. In my own research, I have a single `git` repository for each project that contains all the scripts to process raw data, run analyses, and produce the plots that end up in the published manuscript. This code repository can then be shared with other scientists so they can evaluate, reproduce, and build upon my work. Hopefully this helps the entire field move forward and advance human knowledge – and the grammar of graphics plays a crucial role!

# References


Freyman, W.A. & S. Höhna. 2017. Cladogenetic and anagenetic models of chromosome number evolution: a Bayesian model averaging approach. *Systematic Biology* syx065.

Höhna, S. Freyman, W.A., & M.J. Landis. 2017. RevGadgets: Postprocessing gadgets for output generated by RevBayes. Retrieved from https://github.com/revbayes/RevGadgets

Höhna, S., Landis, M. J., Heath, T. A., Boussau, B., Lartillot, N., Moore, B. R., Huelsenbeck, J. P. & Ronquist, F. (2016). RevBayes: Bayesian phylogenetic inference using graphical models and an interactive model-specification language. *Systematic Biology*, 65(4), 726-736.

Institute For Quantitative Social Science (IQSS) at Harvard. 2017. Introduction to R graphics with ggplot2. Retrieved from http://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html

Wickham, H. 2010. A layered grammar of graphics. *Journal of Computational and Graphical Statistics*, 19(1), 3-28.

Wickham, H., Chang, W. 2017. ggplot2 official documentation. Retrieved from http://ggplot2.tidyverse.org/

Wilkinson, L. 2005. *The Grammar of Graphics* (2nd ed.). Statistics and Computing, New York: Springer

Yu, G., Smith, D. K., Zhu, H., Guan, Y., & Lam, T. T. Y. 2017. ggtree: an R package for visualization and annotation of phylogenetic trees with their covariates and other associated data. *Methods in Ecology and Evolution*, 8(1), 28-36.