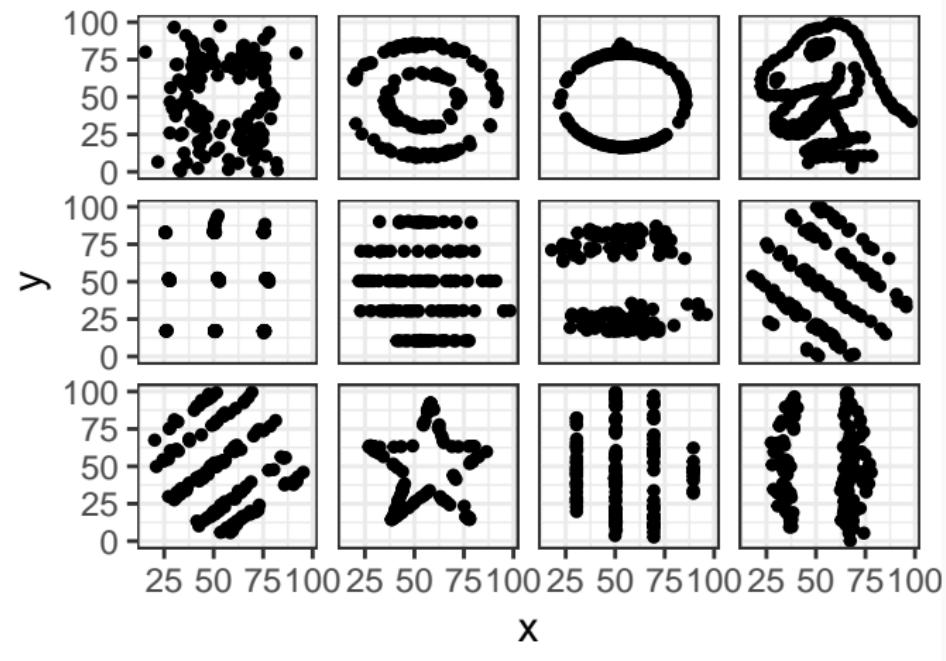


Data visualisation with ggplot2

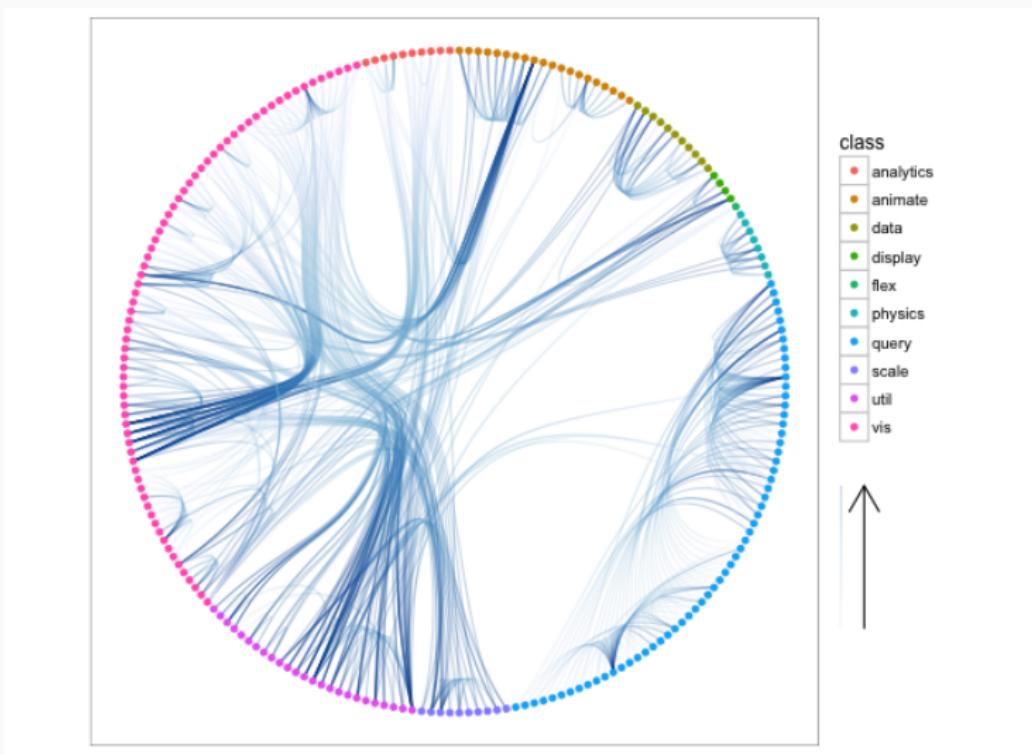
Francisco Rodriguez-Sanchez (@frod_san)

Always plot data!



<https://github.com/stephlocke/datasauRus>

Made with ggplot



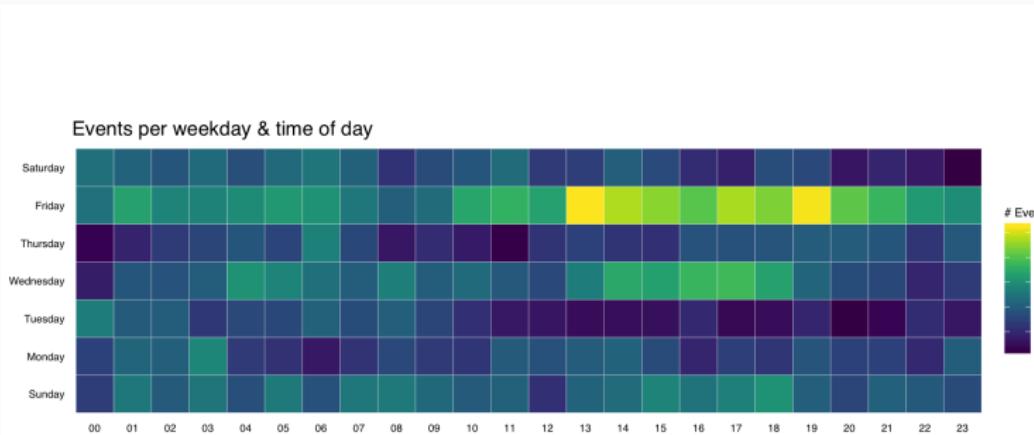
<https://github.com/thomasp85/ggraph>

Made with ggplot



<http://spatial.ly/2012/02/great-maps-ggplot2/>

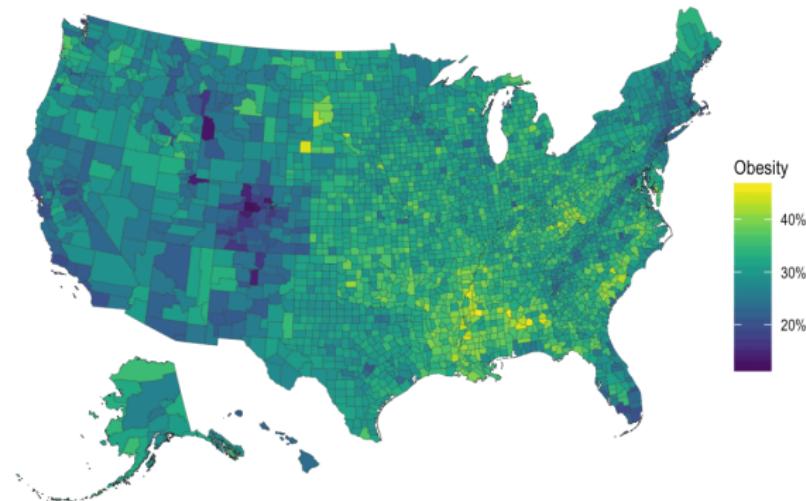
Made with ggplot



<https://rud.is/b/2016/02/14/making-faceted-heatmaps-with-ggplot2/>

U.S. Obesity Rate by County (2012)

Content source: Centers for Disease Control and Prevention

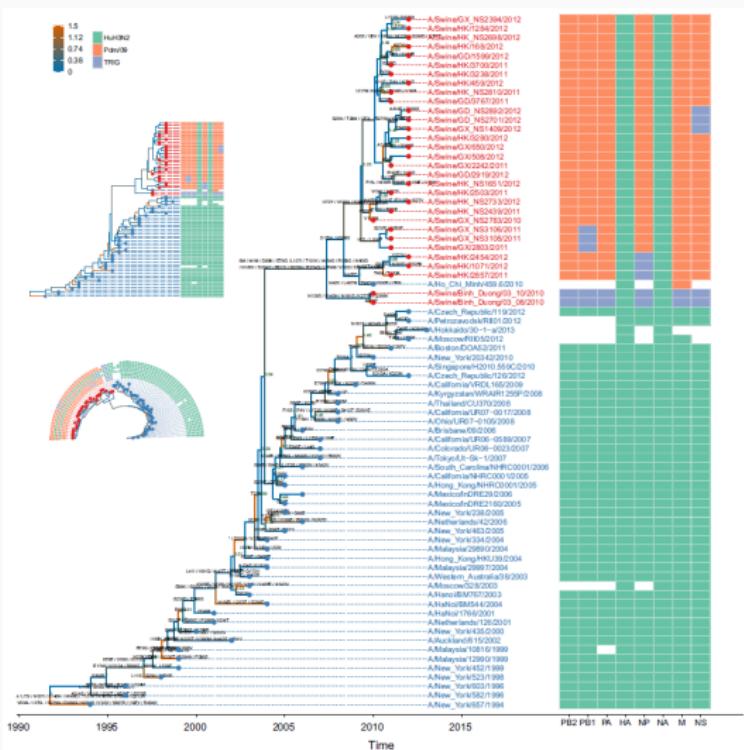


Data from http://www.cdc.gov/diabetes/atlas/countydata/County_ListofIndicators.html

[https:](https://rud.is/b/2016/03/29/easier-composite-u-s-choropleths-with-albersusa/)

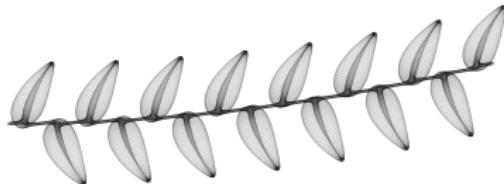
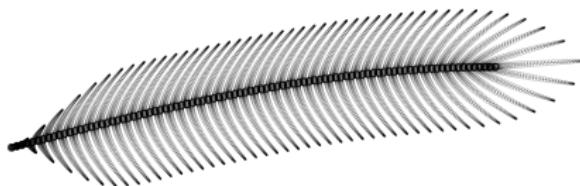
//rud.is/b/2016/03/29/easier-composite-u-s-choropleths-with-albersusa/

Made with ggplot



<https://guangchuangyu.github.io/ggtree/>

Made with ggplot



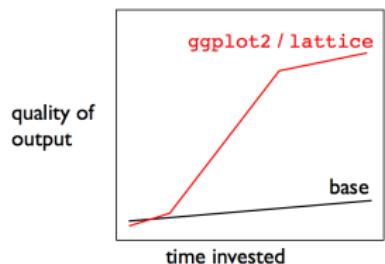
<https://github.com/marcusvolz/mathart>

Why ggplot

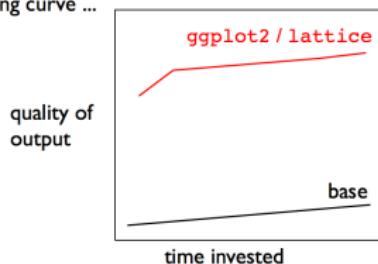
- Extremely powerful and flexible
- Consistent (grammar of graphics)
- Very powerful user base and active development

At the beginning it's hard, but then it pays off

week one



after you've climbed the steepest part of the learning curve ...



* figure is totally fabricated but, I claim, still true

* figure is totally fabricated but, I claim, still true

Source: <https://github.com/jennybc/ggplot2-tutorial>

Very good documentation and tutorials

- Official `ggplot2` documentation
- `ggplot2` book
- `R graphics cookbook` and `Cookbook for R`
- Beautiful plotting in R: A `ggplot2` cheatsheet
- Introduction to `ggplot2`
- Tutorial: `ggplot2`
- How to format plots for publication using `ggplot2`
- Visualising data with `ggplot2`
- Data Visualization with R and `ggplot2`
- `ggplot2` tutorial
- Data visualisation chapter in R for Data Science
- The complete `ggplot2` tutorial
- Data visualization: a practical introduction (K. Healy)
- Fundamentals of data visualization (C. Wilke)

Cheatsheet

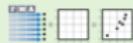
Data Visualization with ggplot2

Cheat Sheet



Basics

`ggplot2` is based on the **grammar of graphics**. The idea is that you can build every graph from the same components: a **data set**, a **coordinate system**, and **geoms**—visual marks that represent data points.



To display values, map variables in the data to visual properties of the geom's aesthetics (like size, color, and x/y locations).



Complete the template below to build a graph.

```
ggplot([data = mpg], aes(x = cyl, y = hwy))  
  + [ ]  
  + stat = [ ]  
  + position = [ ]  
  + [ ]  
  + [ ]  
  + [ ]  
  + [ ]  
  + [ ]
```

Required
Not required, sensible defaults supplied

Begin a plot that you finish by adding layers to. Add one geom function per layer.

```
ggplot([x = cyl, y = hwy, data = mpg], geom = "point")  
  + [ ]  
  + [ ]  
  + [ ]  
  + [ ]  
  + [ ]  
  + [ ]  
  + [ ]
```

Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

`last_plot()`

Returns the last plot.

```
ggsave("plot.png", width = 5, height = 5)  
  + [ ]
```

Saves last plot as 5" x 5" file named "plot.png" in working directory. Makes file type to file extension.

RStudio® is a trademark of RStudio, Inc. | www.rstudio.com | 844-448-2222 | rstudio.com

Geoms - Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

Graphical Primitives

- a `geom_abline`(*intercept*, *slope*)
- b `geom_bar`(*group*)
- c `geom_blank`()
- d `geom_curve`(*angle*=0, *x*=*mean*, *y*=*mean*, *xend*=*x*, *yend*=*y*, *arrow*, *color*, *curvature*, *length*, *size*)
- e `geom_path`(*method*="butt", *linejoin*=c("miter", "round", "bevel"))
- f `geom_polygon`(*group*)
- g `geom_rect`(*xmin*=*x*, *xmax*=*x*, *ymin*=*y*, *ymax*=*y*, *alpha*, *color*, *fill*, *linetype*, *size*)
- h `geom_ribbon`(*lower*=*mean*, *upper*=*mean*+900, *min_hue*=900, *max_hue*=900, *alpha*, *color*, *fill*, *group*, *linetype*, *size*)

Line Segments

- common aesthetics: *x*, *y*, *alpha*, *color*, *linetype*, *size*
- i `geom_abline`(*intercept*, *slope*)
 - j `geom_hline`(*intercept*)
 - k `geom_vline`(*intercept*)
 - l `geom_segment`(*x*=*x0*, *y*=*y0*, *xend*=*x1*, *yend*=*y1*)
 - m `geom_spline`(*control*=c(1.15, 0.1))

One Variable

- Continuous
- c `geom_area`(*hsewheight*)
 - d `geom_area`(*stat*="binwidth")
 - e `geom_area(stat = "bin")`
 - f `geom_density`(*kernel*="gaussian")
 - g `geom_densplot`
 - h `geom_freqpoly`
 - i `geom_histogram`(*binwidth*=5)
 - j `geom_qqplot`(*sample*=*hwj*)
 - k `geom_violin`(*scale*="area")

Discrete

- l `geom_bar`
- m `geom_count`
- n `geom_dotplot`(*binwidth*=2)
- o `geom_hex`
- p `geom_map`(*map*=*state*, *map*=*map* + *expand_limits*(*x*=*map\$lon*, *y*=*map\$lat*))
- q `geom_raster`(*res*=16, *xj*=*hj*, *yhj*=0.5, *interp*=FALSE)
- r `geom_tile`(*res*=16)

Two Variables

Continuous X, Continuous Y

- a `geom_abline`(*intercept*, *slope*)
- b `geom_jitter`(*height*=2, *width*=2)
- c `geom_point`
- d `geom_quasirandom`
- e `geom_rug`(*side*="l")
- f `geom_smooth`(*method*="loess")
- g `geom_text`(*label*=*label*, *method*=*xyjitter*, *x*=*x*, *y*=*y*, *alpha*, *color*, *group*, *linetype*, *size*)

Continuous Function

- h `geom_area`
- i `geom_line`
- j `geom_step`(*direction*="hv")

Continuous Bivariate Distribution

- k `geom_bivar2d`(*smooth*=0.025, *grid*)
- l `geom_hex`
- m `geom_hex`(*alpha*, *color*, *fill*, *size*)

Visualizing error

- n `geom_crossbar`(*stat*=2)
- o `geom_errorbar`
- p `geom_errorbarh`
- q `geom_linerange`
- r `geom_pointrange`

Maps

- s `geom_map`(*map*=*usArrests*, *map*=*state*, *map*=*usArrests*)
- t `geom_map`(*map*=*state*, *map*=*map* + *expand_limits*(*x*=*map\$lon*, *y*=*map\$lat*))

Three Variables

- u `geom_raster`(*res*=16, *xj*=*hj*, *yhj*=0.5, *interp*=FALSE)
- v `geom_hex`(*res*=16)
- w `geom_tile`(*res*=16)

Learn more at ggplot2.org and ggplot2-exts.org | ggplot2 2.1.0 • Updated 11/18

<https://www.rstudio.com/resources/cheatsheets/>

Repos of figures + code

- R graph catalog
- From Data to Viz
- The R graph gallery
- R graph gallery
- Cookbook for R: Graphs
- Graphical data analysis with R
- IEG figures

Find answers for all your questions in Stack Overflow



Search

ggplot2

36,854 results



The Practical Dev
@ThePracticalDev



The last programming book you'll ever need

Cutting corners to meet arbitrary management deadlines



Essential

Copying and Pasting
from Stack Overflow

Building a ggplot figure

Our example dataset: tree heights and DBH

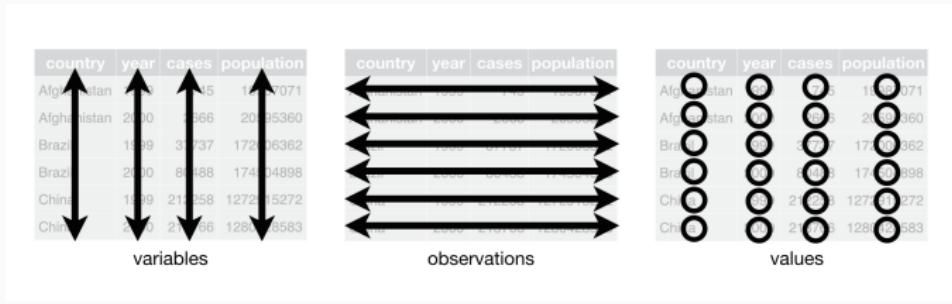
<http://tinyurl.com/treesdata>

- One species
- 10 plots
- 1000 trees
- Number of trees per plot ranging from 4 to 392

```
trees <- read.csv("data/trees.csv")
summary(trees[, 1:3])
```

	plot	dbh	height
Min.	: 1.0	Min. : 5.06	Min. :13.40
1st Qu.	: 1.0	1st Qu.:17.69	1st Qu.:29.68
Median	: 2.0	Median :28.62	Median :36.55
Mean	: 2.7	Mean :27.88	Mean :36.51
3rd Qu.	: 4.0	3rd Qu.:38.97	3rd Qu.:43.33
Max.	:10.0	Max. :49.92	Max. :59.30

Data must be a tidy data frame



```
tidy::gather(table4, key = "year", value = "cases", "1999", "2000")
```

The diagram illustrates the process of gathering data from a wide table into a tidy table:

The original wide table "table4" is shown at the bottom right:

country	year	cases
Afghanistan	1999	745
Afghanistan	2000	2666
Brazil	1999	37737
Brazil	2000	80488
China	1999	212258
China	2000	213766

Arrows point from specific rows in the wide table to the columns in the tidy table above it:

- Rows for Afghanistan point to the "1999" and "2000" columns.
- Rows for Brazil point to the "1999" and "2000" columns.
- Rows for China point to the "1999" and "2000" columns.

country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

<http://r4ds.had.co.nz/tidy-data.html>

Calling ggplot

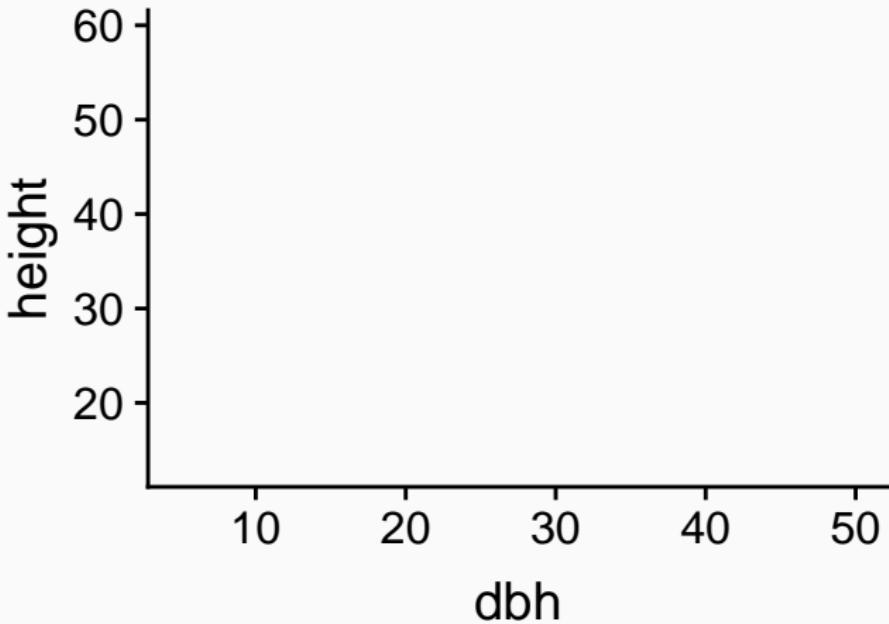
```
library(ggplot2)  
ggplot(trees)
```

```
ggplot(trees)
```

First argument is a tidy data frame

What variables as axes?

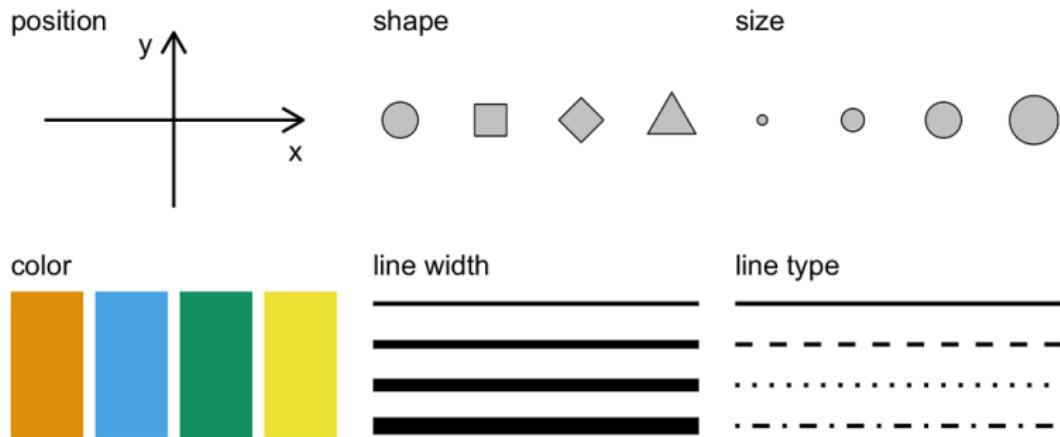
```
ggplot(trees) +  
  aes(x = dbh, y = height)
```



Note syntax: + followed by new line

```
ggplot(trees) +  
  aes(x = dbh, y = height)
```

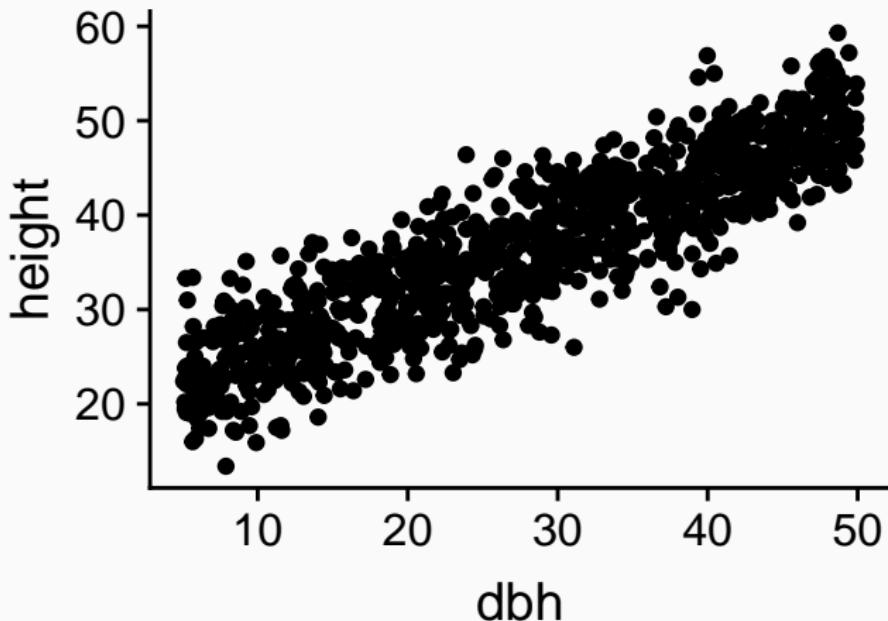
Aesthetics (`aes`) map data variables (`dbh, height`) to graphic elements (`axes`)



<http://serialmentor.com/dataviz/aesthetic-mapping.html>

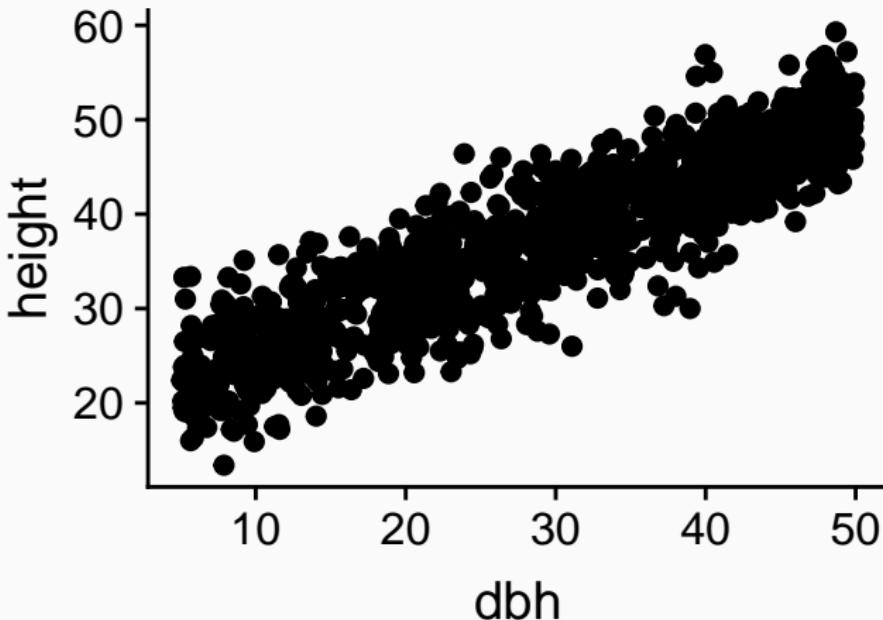
Adding layers (geoms)

```
ggplot(trees) +  
  aes(x = dbh, y = height) +  
  geom_point()
```



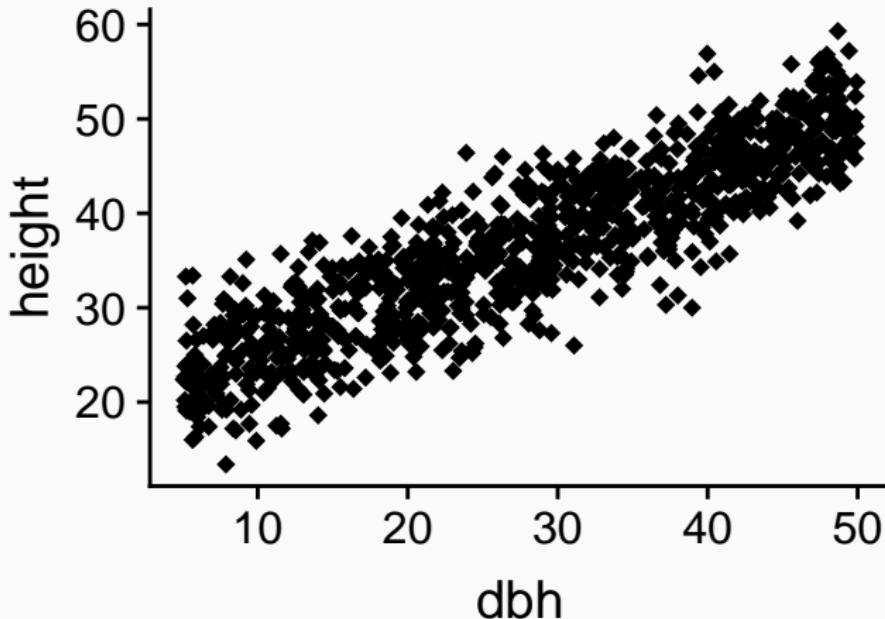
Changing point size

```
ggplot(trees) +  
  aes(x = dbh, y = height) +  
  geom_point(size = 2)
```



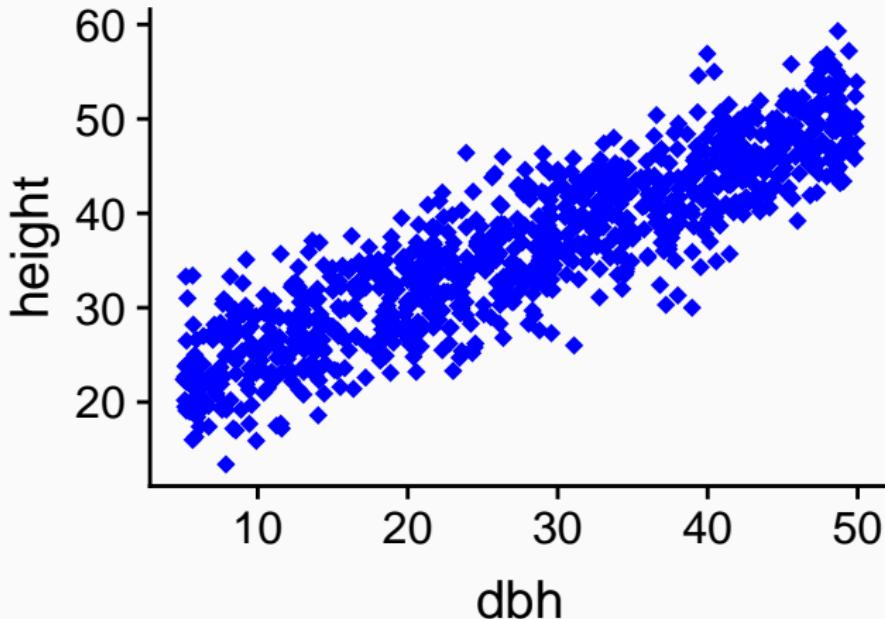
Changing point size and type

```
ggplot(trees) +  
  aes(x = dbh, y = height) +  
  geom_point(size = 2, shape = 18)
```



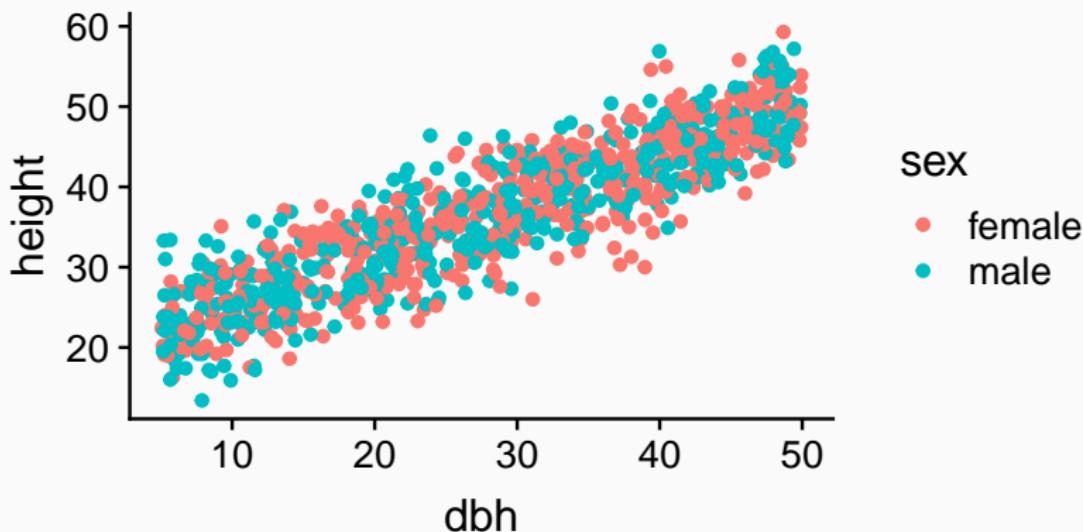
Changing point size and type

```
ggplot(trees) +  
  aes(x = dbh, y = height) +  
  geom_point(size = 2, shape = 18, colour = 'blue')
```



Map geom aesthetics (e.g. colour) to variable

```
ggplot(trees) +  
  aes(x = dbh, y = height) +  
  geom_point(aes(colour = sex))
```



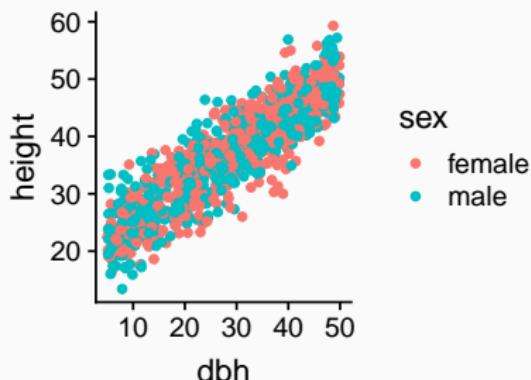
Note difference between

```
geom_point(colour = "blue")
# colour is given a concrete value ('blue')
```

```
geom_point(aes(colour = sex))
# colour maps a *variable* (using `aes`)
```

This works:

```
ggplot(trees) +  
  aes(x = dbh, y = height) +  
  geom_point(aes(colour = sex))
```



This doesn't work:

```
ggplot(trees) +  
  aes(x = dbh, y = height) +  
  geom_point(colour = sex)
```

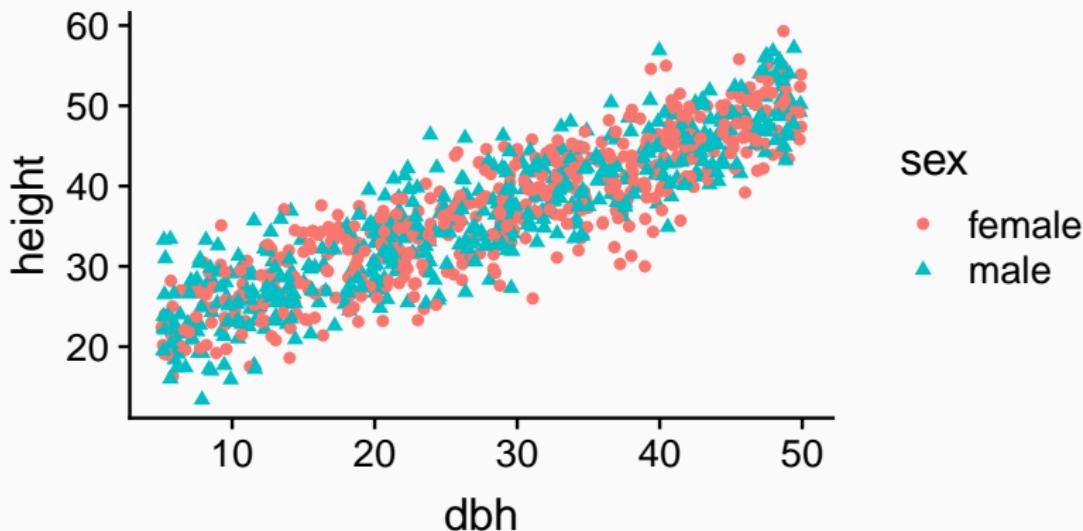
*Error in layer(data = data, mapping =
mapping, stat = stat, geom =
GeomPoint, : object 'sex' not found*

'sex' is a variable in dataframe

Must use aes

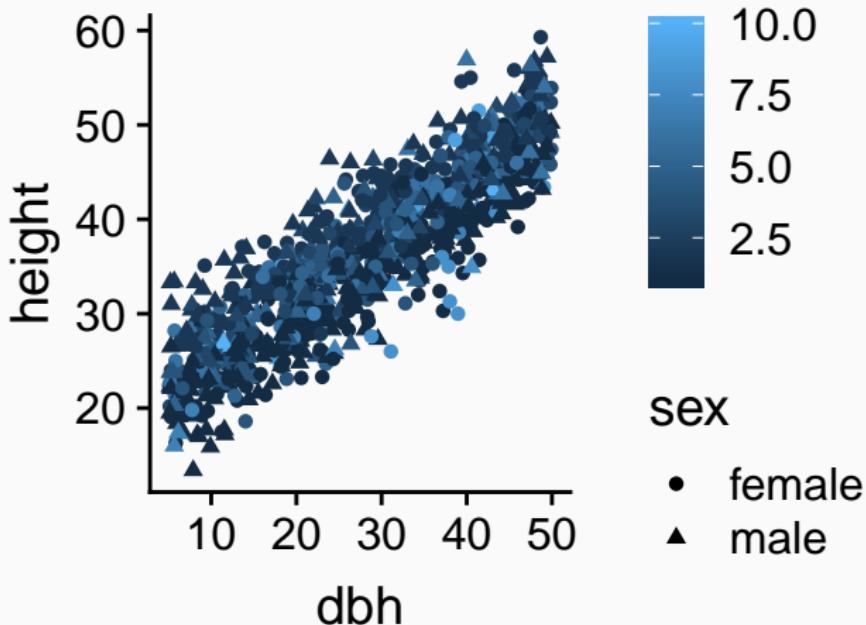
Map geom aesthetics (colour, shape) to variable

```
ggplot(trees) +  
  aes(x = dbh, y = height) +  
  geom_point(aes(colour = sex, shape = sex))
```



Map geom aesthetics (colour, shape) to variable

```
ggplot(trees) +  
  aes(x = dbh, y = height) +  
  geom_point(aes(colour = plot, shape = sex))
```

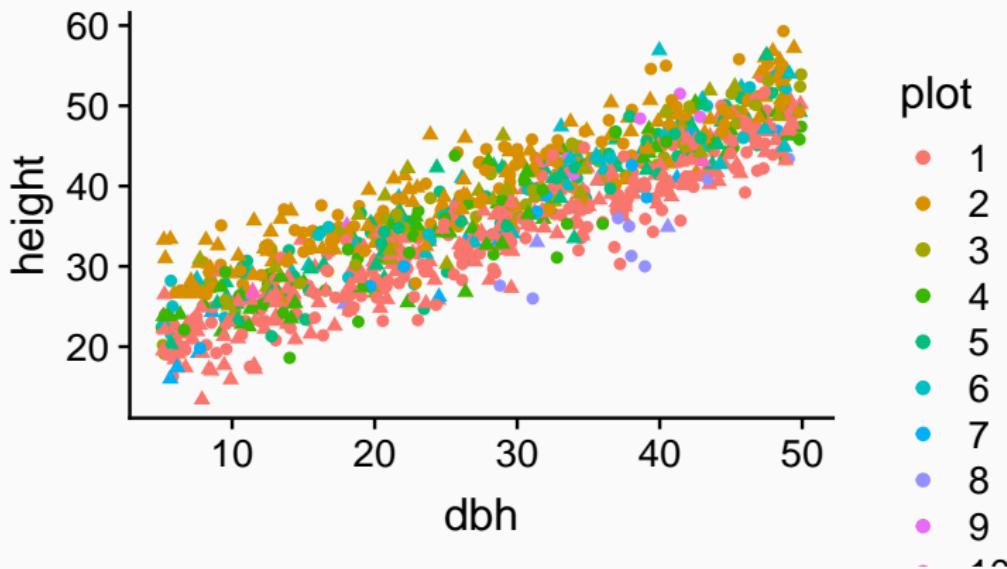


Ensuring 'plot' is a factor, not numeric

```
trees$plot <- as.factor(trees$plot)
```

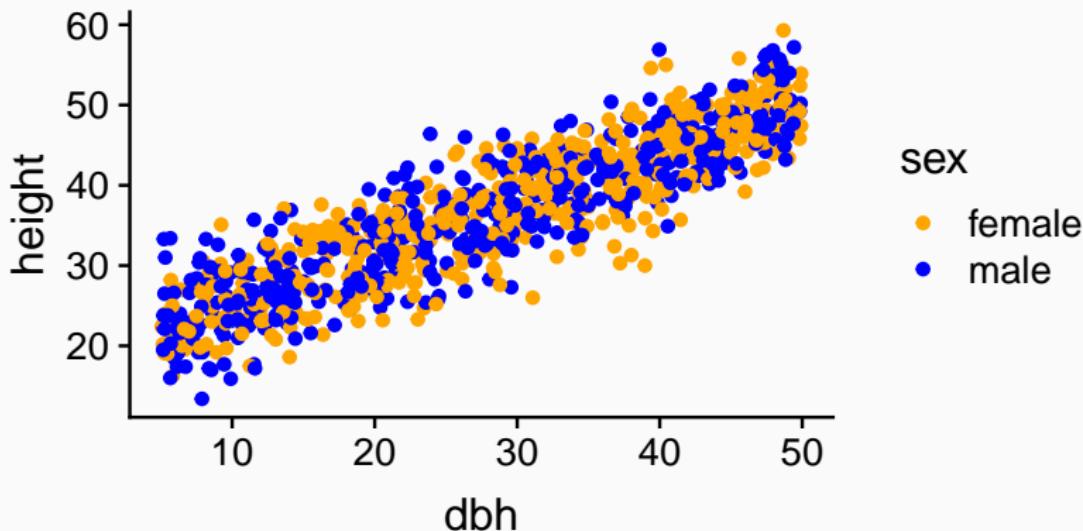
Map geom aesthetics (colour, shape) to variable

```
ggplot(trees) +  
  aes(x = dbh, y = height) +  
  geom_point(aes(colour = plot, shape = sex))
```



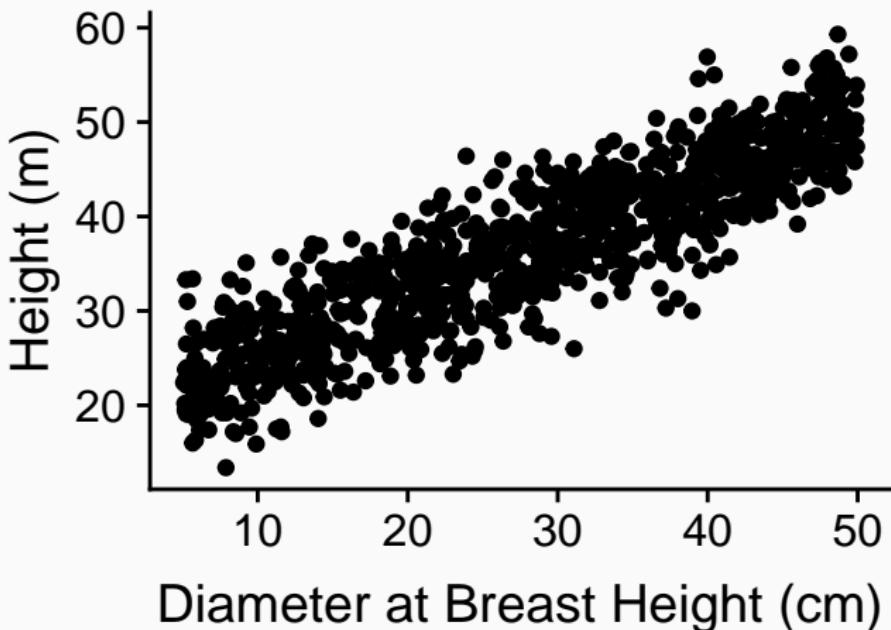
Change colour scale

```
ggplot(trees) +  
  aes(x = dbh, y = height) +  
  geom_point(aes(colour = sex)) +  
  scale_colour_manual(values = c("orange", "blue"))
```



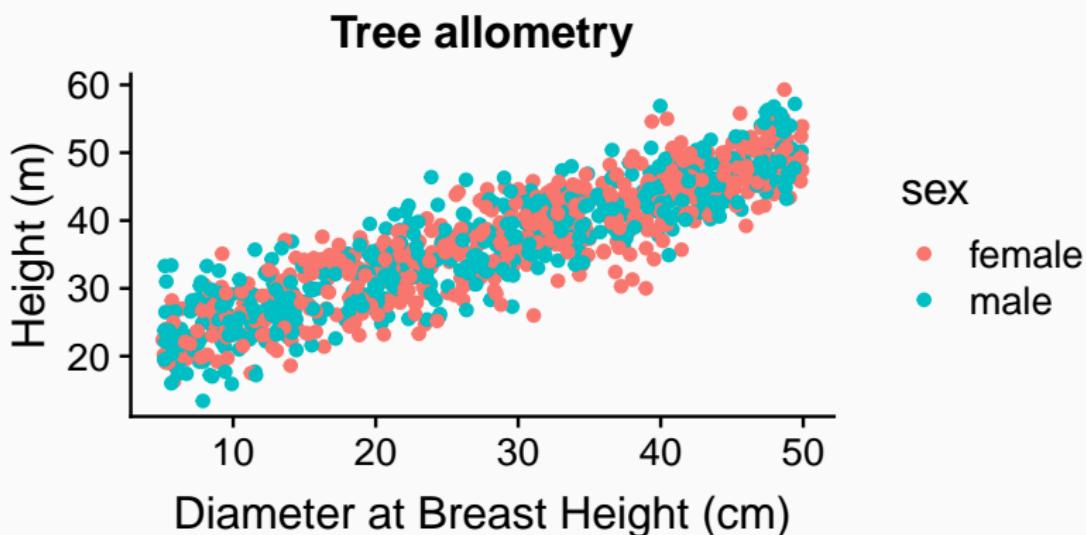
Change axis labels: `xlab` & `ylab`

```
ggplot(trees) +  
  aes(x = dbh, y = height) +  
  geom_point() +  
  labs(x = "Diameter at Breast Height (cm)", y = "Height (m)")
```



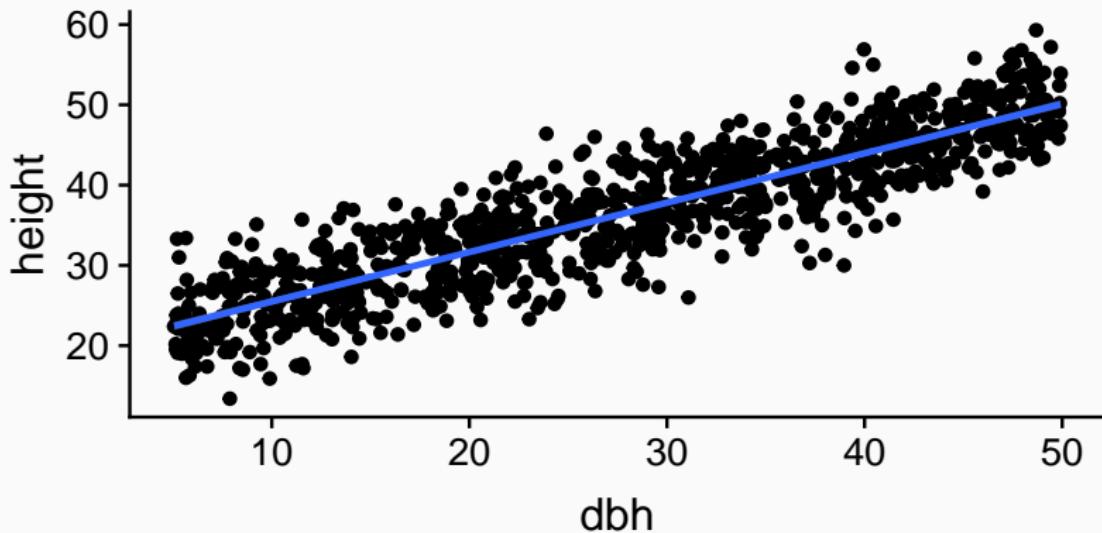
Set title

```
ggplot(trees) +  
  aes(x = dbh, y = height) +  
  geom_point(aes(colour = sex)) +  
  labs(x = "Diameter at Breast Height (cm)", y = "Height (m)") +  
  labs(title = "Tree allometry")
```



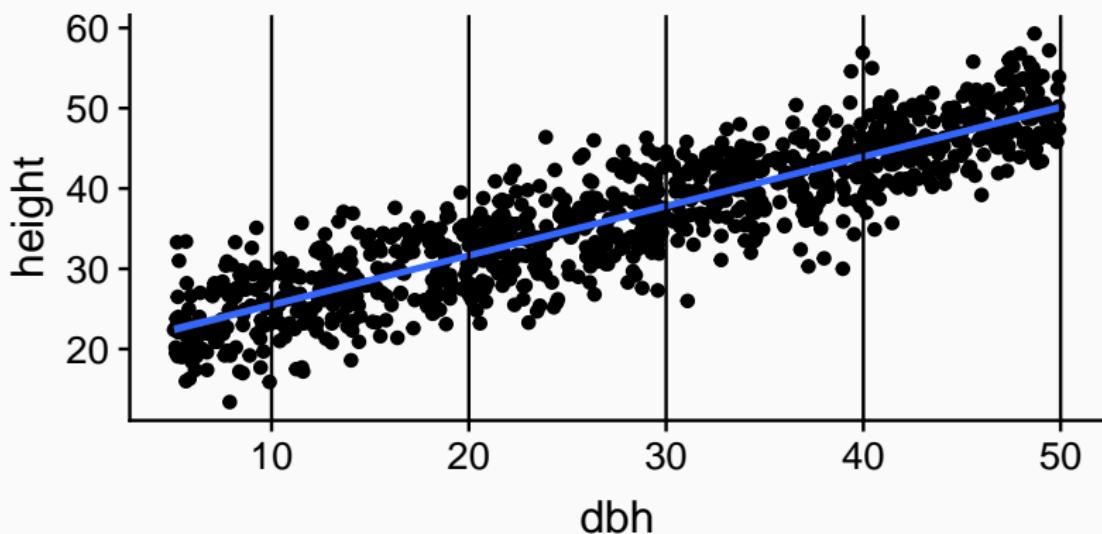
Adding more layers

```
ggplot(trees) +  
  aes(x = dbh, y = height) +  
  geom_point() +  
  geom_smooth(method = "lm")
```



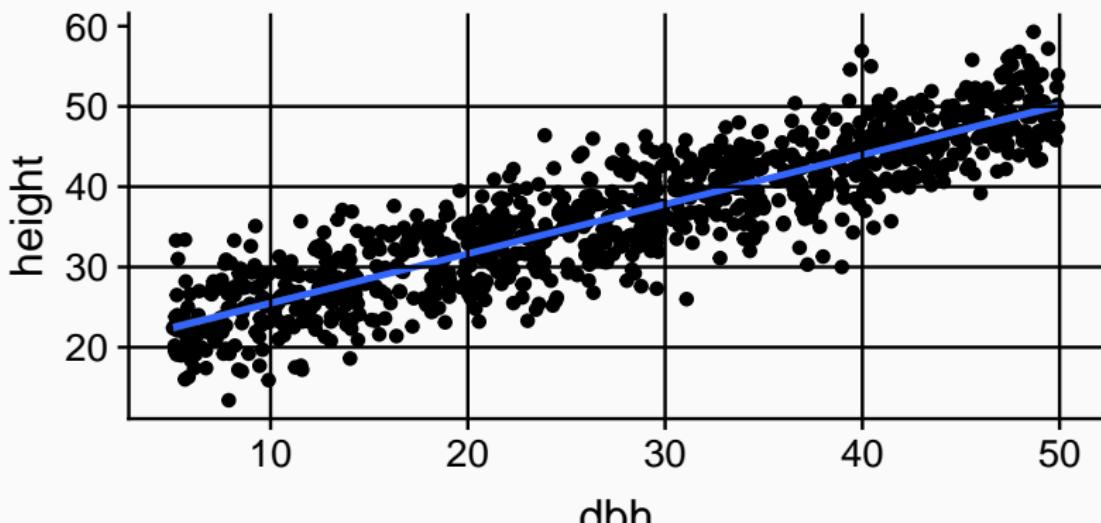
Adding more layers

```
ggplot(trees) +  
  aes(x = dbh, y = height) +  
  geom_point() +  
  geom_smooth(method = "lm") +  
  geom_vline(xintercept = c(10, 20, 30, 40, 50))
```



Adding more layers

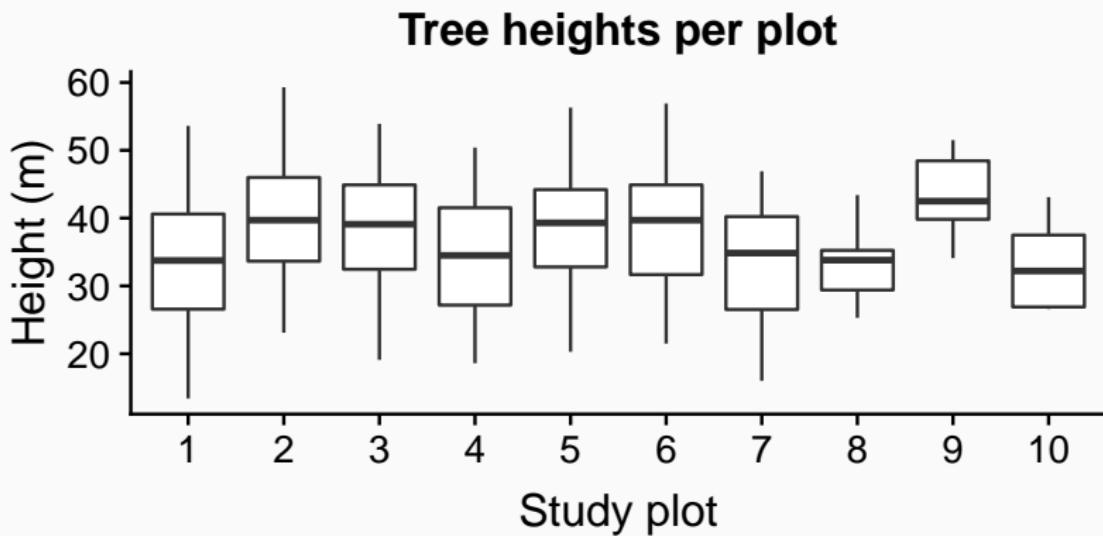
```
ggplot(trees) +  
  aes(x = dbh, y = height) +  
  geom_point() +  
  geom_smooth(method = "lm") +  
  geom_vline(xintercept = c(10, 20, 30, 40, 50)) +  
  geom_hline(yintercept = c(20, 30, 40, 50))
```



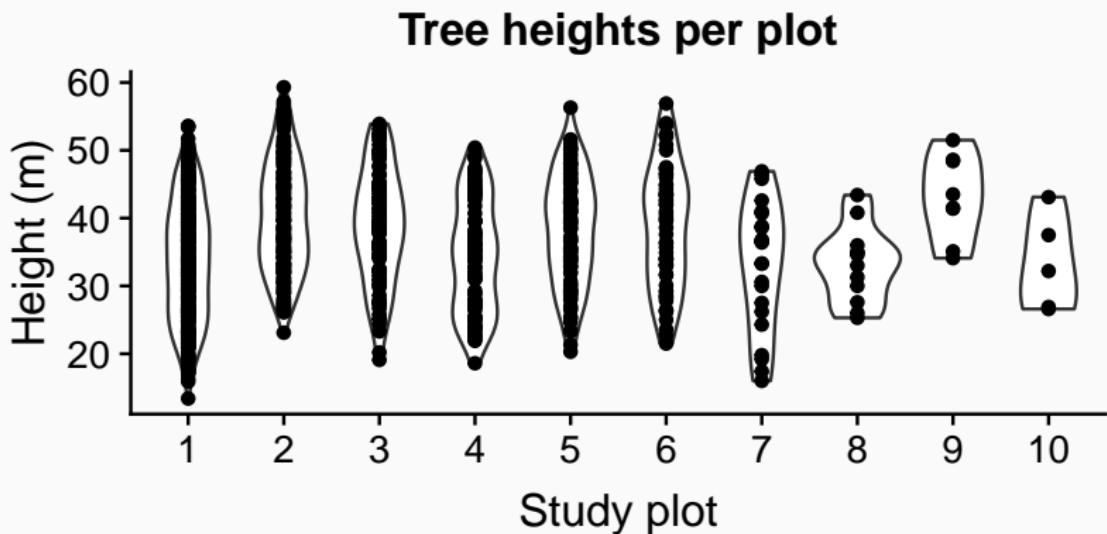
Summary

```
ggplot(trees) +          # Name of (tidy) data frame  
  aes(x = dbh, y = height) + # Aesthetics (variables to map in axes)  
  geom_point()             # Geoms: geometric objects
```

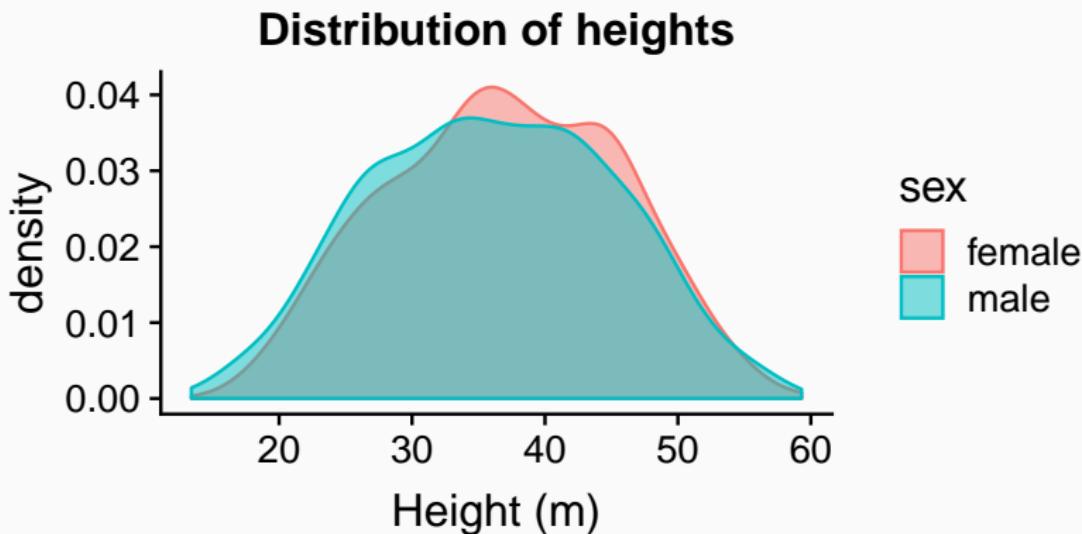
Exercise: Make a plot like this one



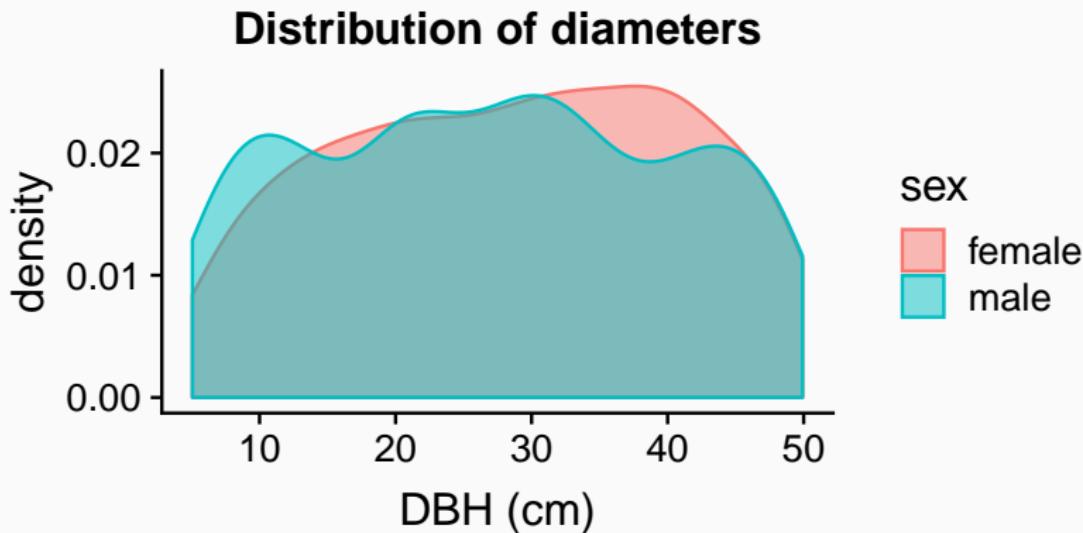
Exercise: Make a plot like this one



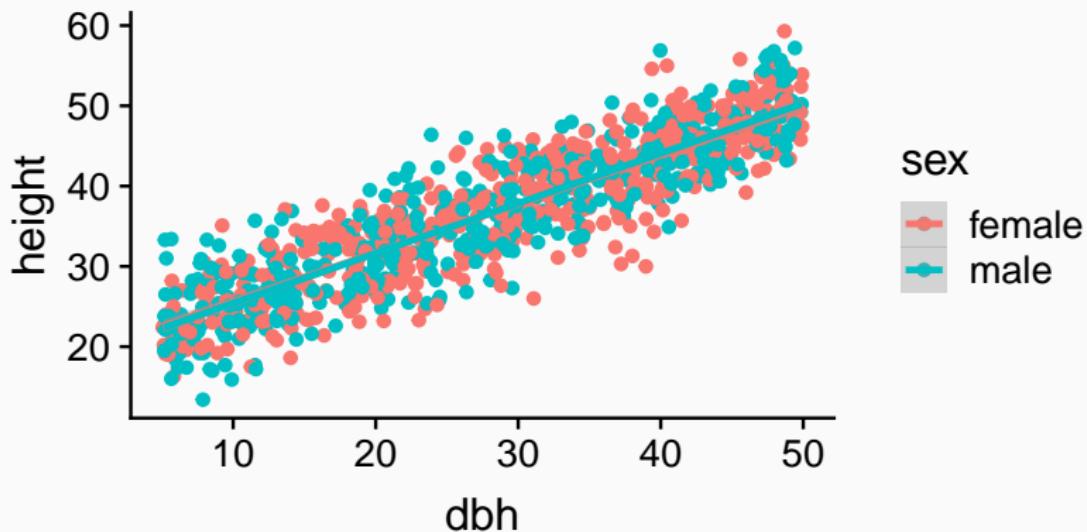
Exercise: Make a plot like this one



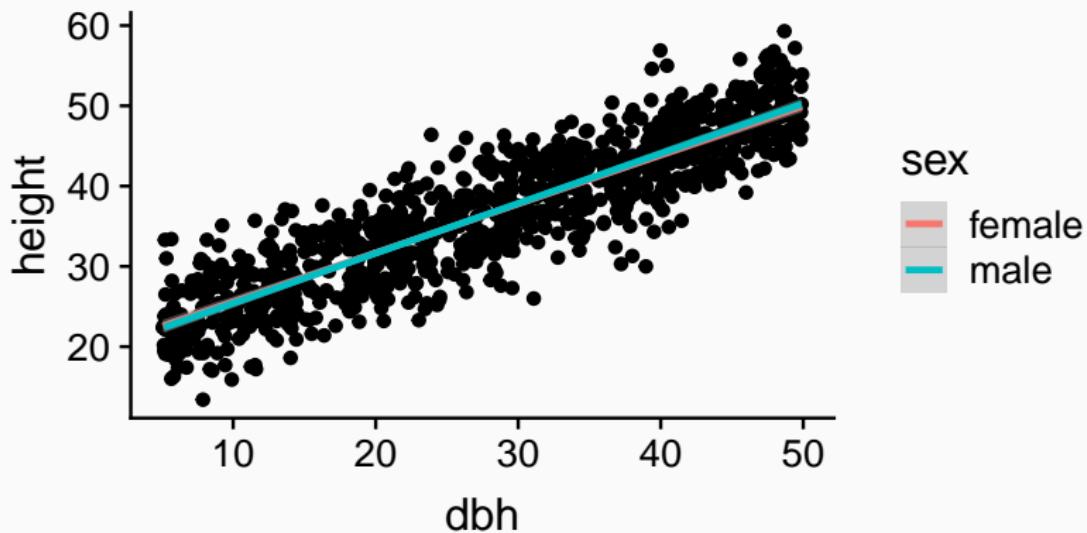
Exercise: Make a plot like this one



Exercise: Make a plot like this one



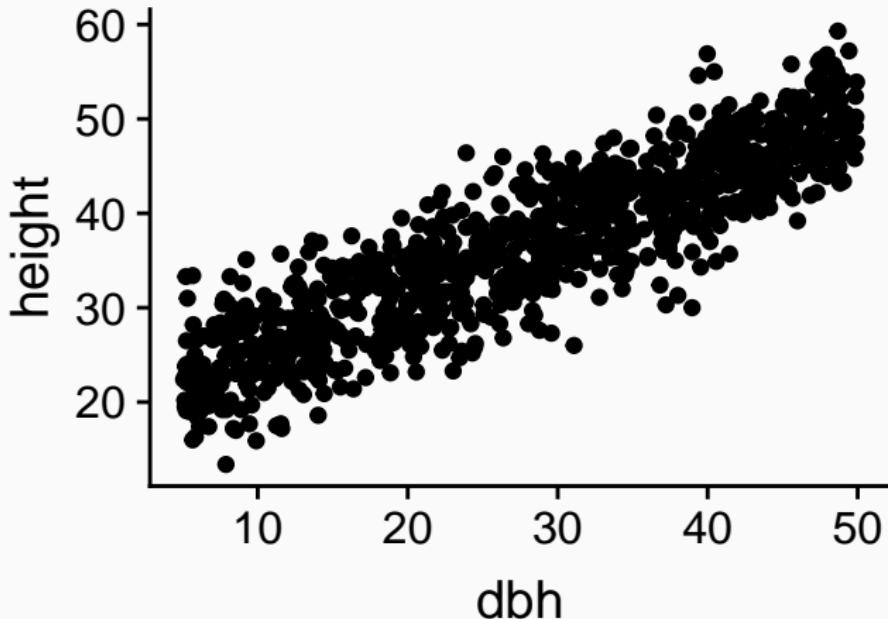
Exercise: Make a plot like this one



**ggplot2 figures can be assigned to R
objects**

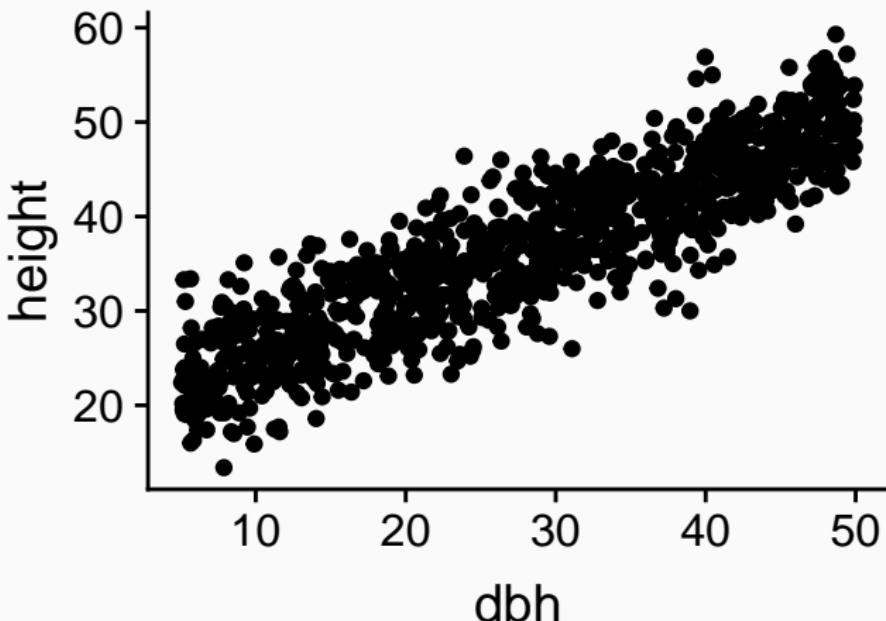
Assigning ggplot objects

```
myplot <- ggplot(trees) +  
  aes(x = dbh, y = height)  
myplot + geom_point()
```



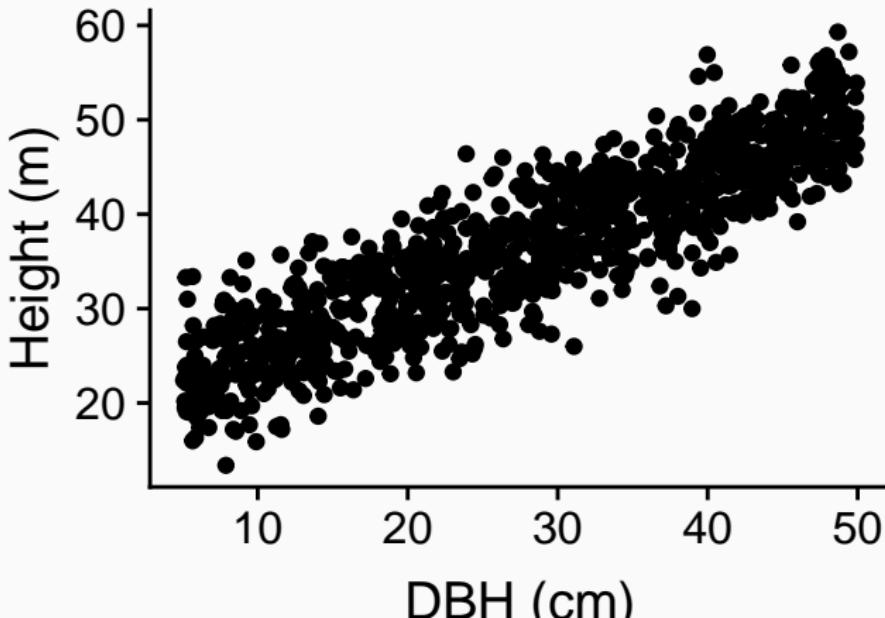
Assigning ggplot objects

```
myplot <- ggplot(trees) +  
  aes(x = dbh, y = height)  
myplot <- myplot + geom_point()  
myplot
```



Assigning ggplot objects

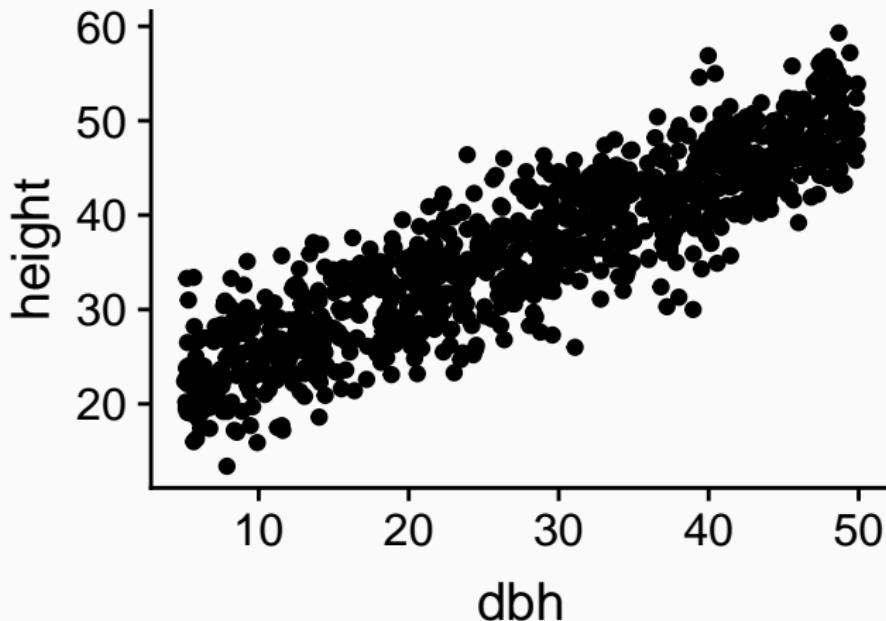
```
baseplot <- ggplot(trees) +  
  aes(x = dbh, y = height)  
scatterplot <- baseplot + geom_point()  
labelled <- scatterplot + labs(x = "DBH (cm)", y = "Height (m)")  
labelled
```



Themes: changing plot appearance

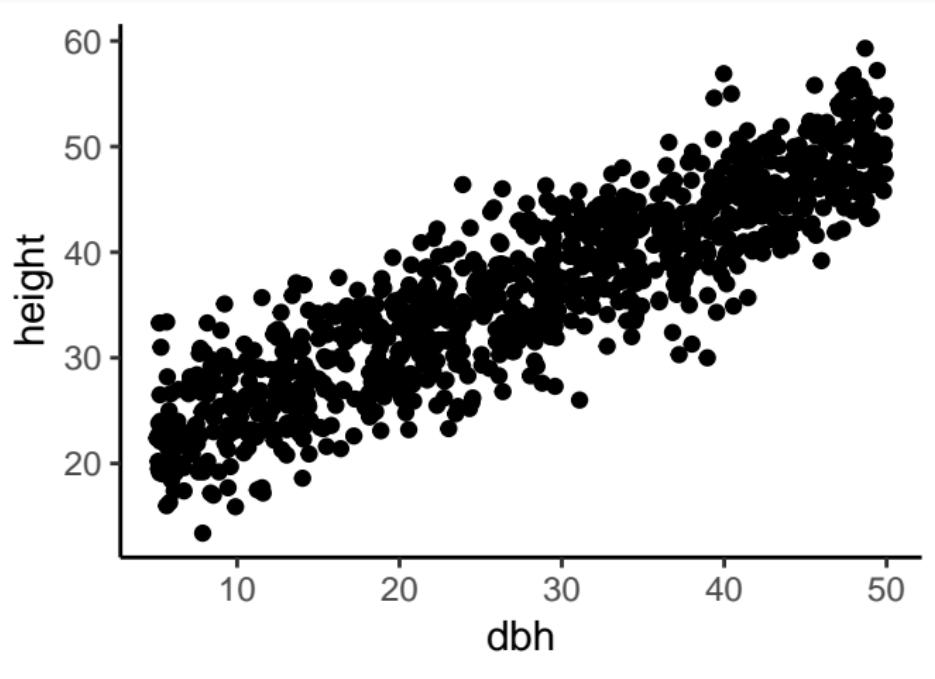
myplot

```
myplot <- ggplot(trees) +  
  aes(x = dbh, y = height) +  
  geom_point()
```



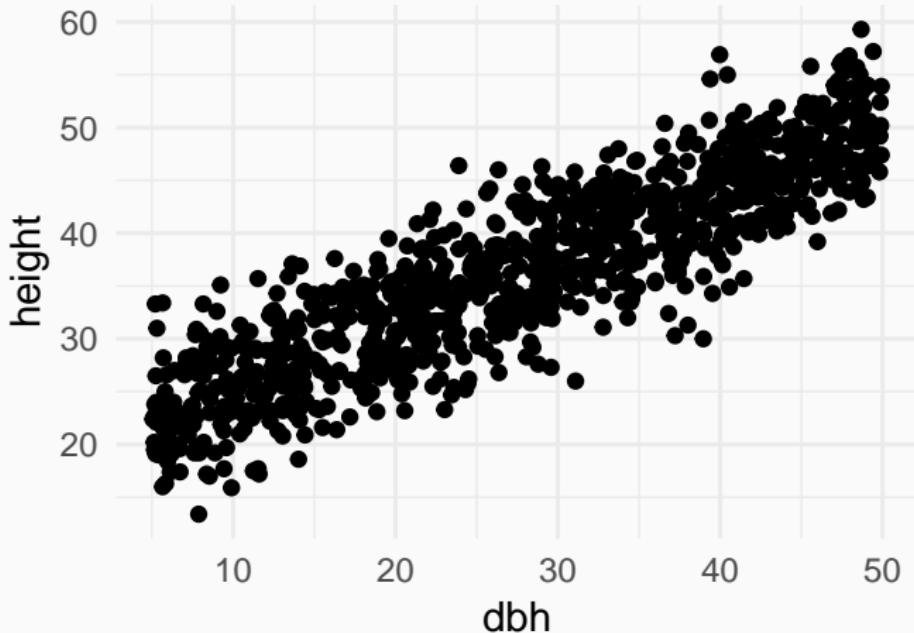
theme_classic

```
myplot + theme_classic()
```



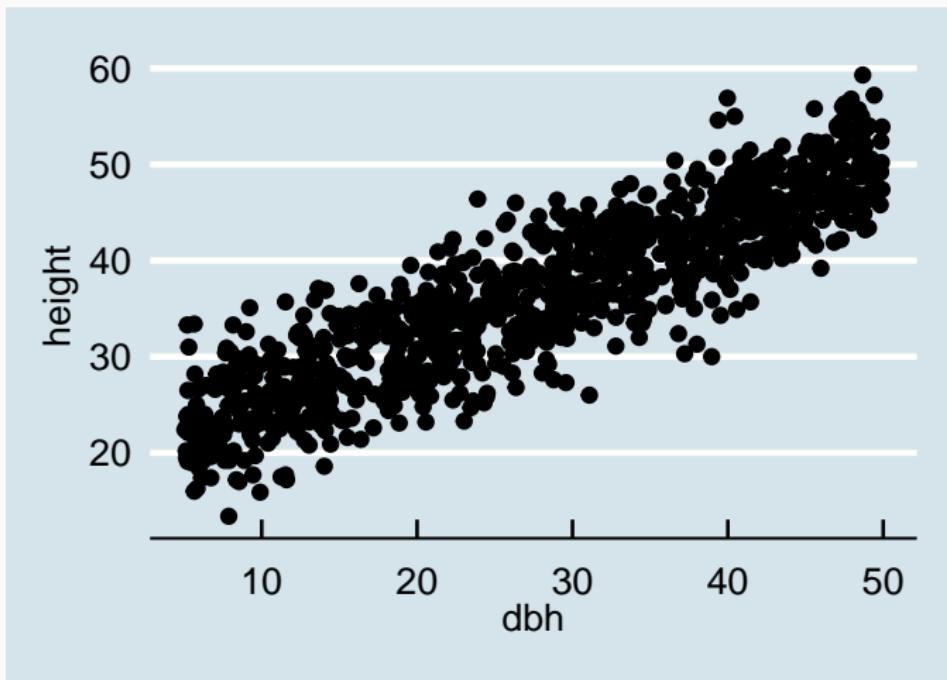
theme_minimal

```
myplot + theme_minimal()
```



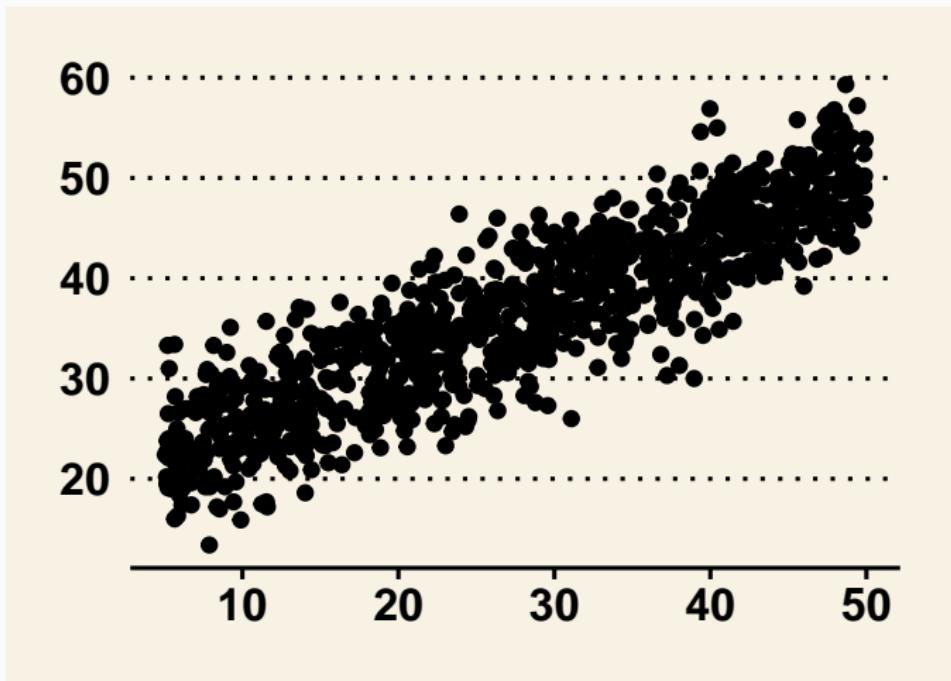
Lots of themes out there

```
library(ggthemes)  
myplot + theme_economist()
```



Lots of themes out there

```
myplot + theme_wsj()
```

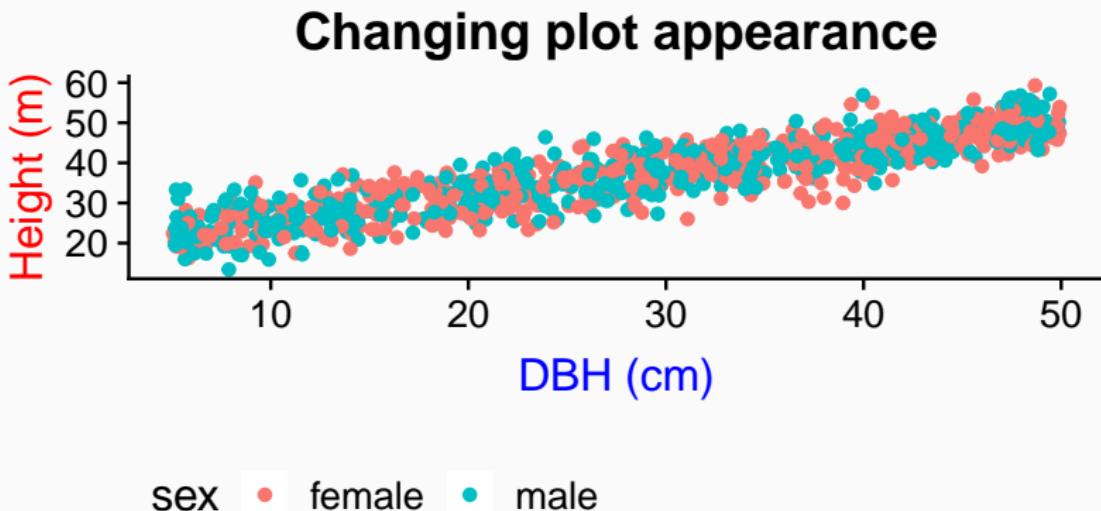


Editing themes

```
?theme
```

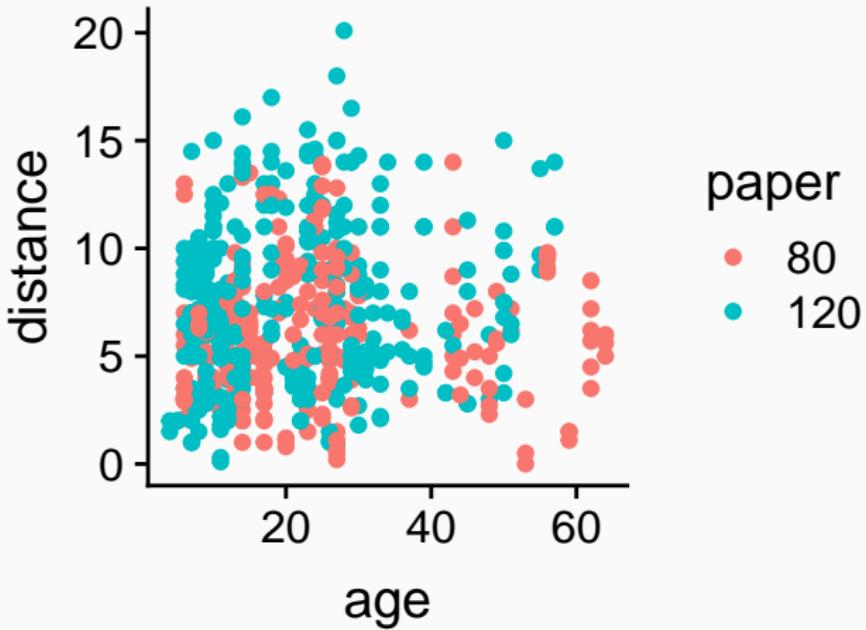
- element_blank
- element_text
- element_line
- element_rect (borders & backgrounds)

Exercise: make a plot like this one



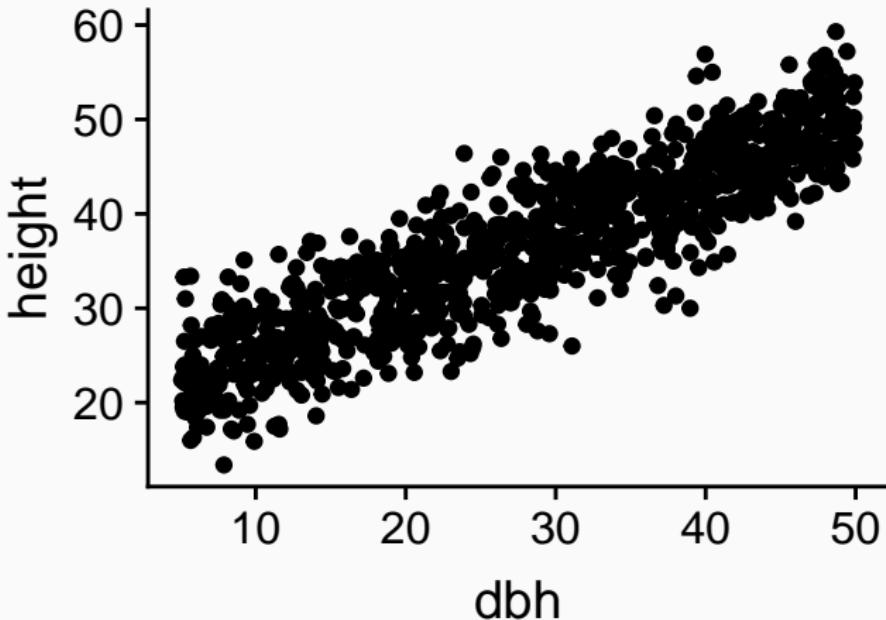
Easily changing appearance with ggthemearr (Rstudio addin)

<https://github.com/calligross/ggthemearr>



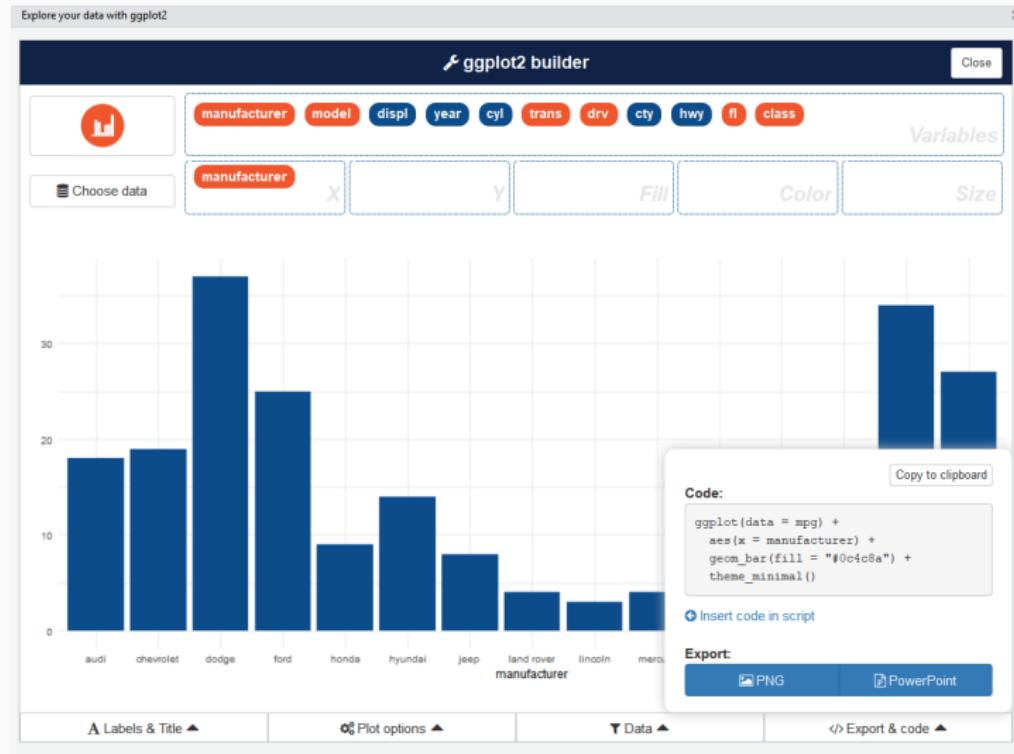
Easily changing appearance with ggedit

<https://github.com/metrumresearchgroup/ggedit>



esquisse: ggplot2 builder addin

<https://github.com/dreamRs/esquisse>





Trevor A. Branch
@TrevorABranch

Follow

My rule of thumb: every analysis you do on a dataset will have to be redone 10–15 times before publication. Plan accordingly. #Rstats

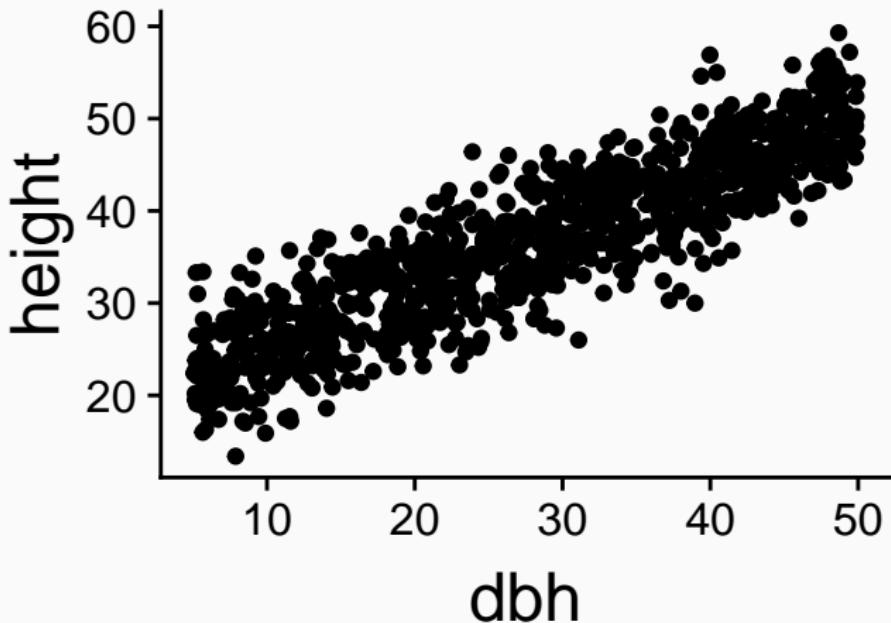
<http://mbjoseph.github.io/2015/02/26/plotting.html>

serialmentor.com/dataviz/choosing-the-right-visualization-software.html

Think twice before editing plots out of R

Referee #3: "Please increase font size in all figures"

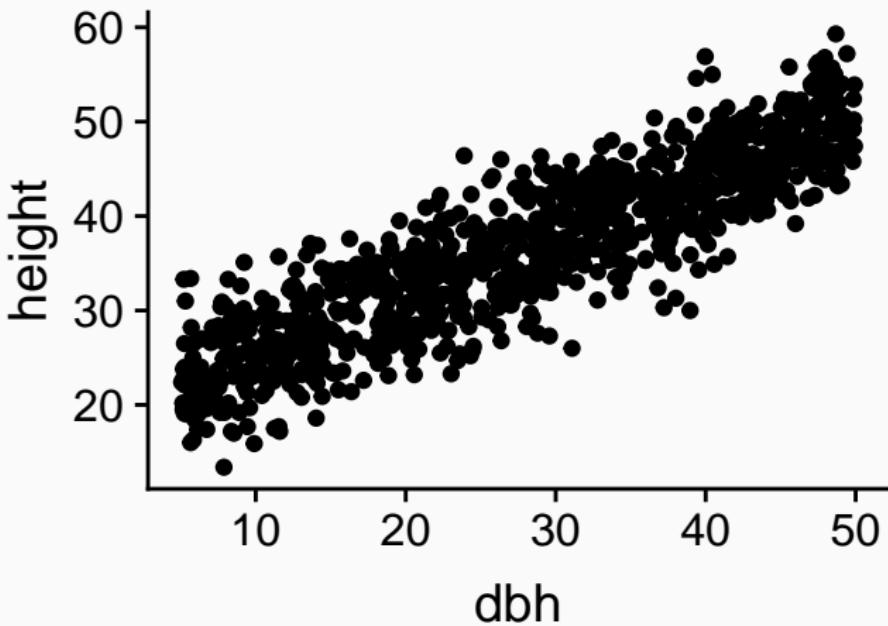
```
myplot +  
  theme(axis.title = element_text(size = 18))
```



Publication-quality plots

```
library(cowplot)
```

```
myplot
```



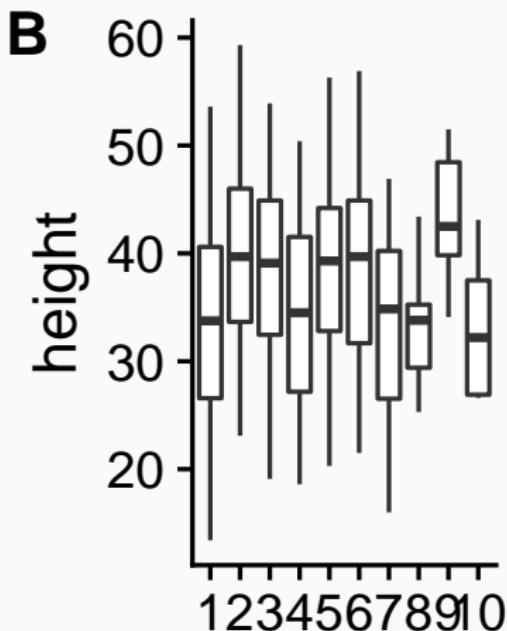
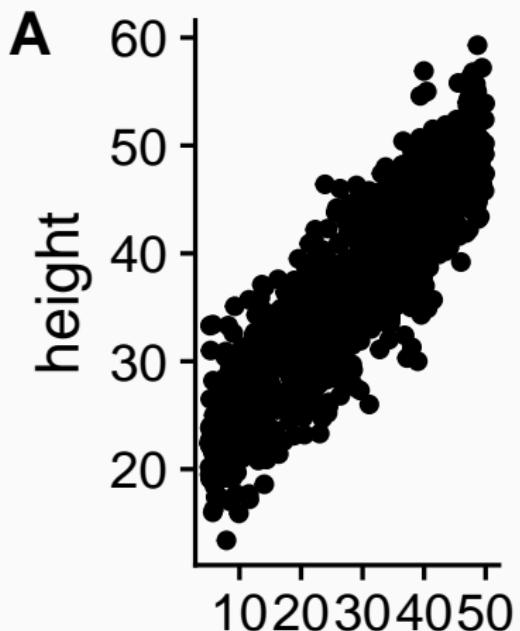
Some publication themes:

<https://gist.github.com/Pakillo/c2c7ea11c528cc2ee20f#themes>

Composite figures

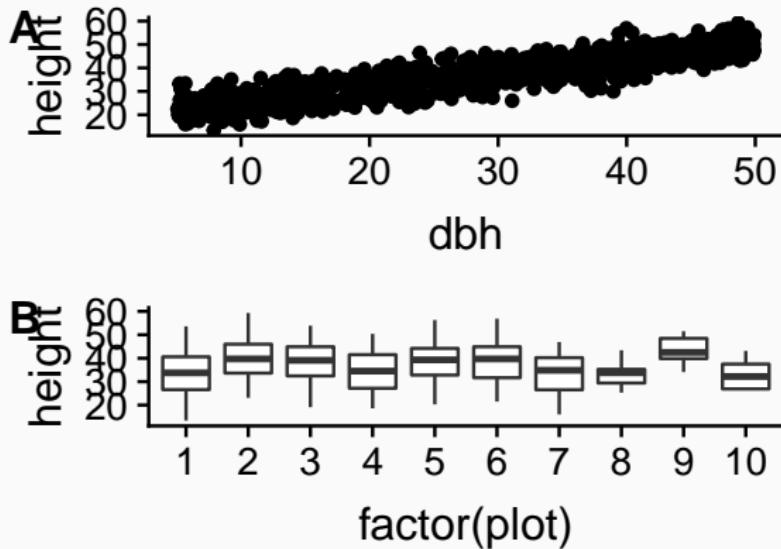
Composite figures: cowplot

```
library(cowplot)  
plot1 <- ggplot(trees) + aes(dbh, height) + geom_point()  
plot2 <- ggplot(trees) + aes(factor(plot), height) + geom_boxplot()  
plot_grid(plot1, plot2, labels = "AUTO")
```

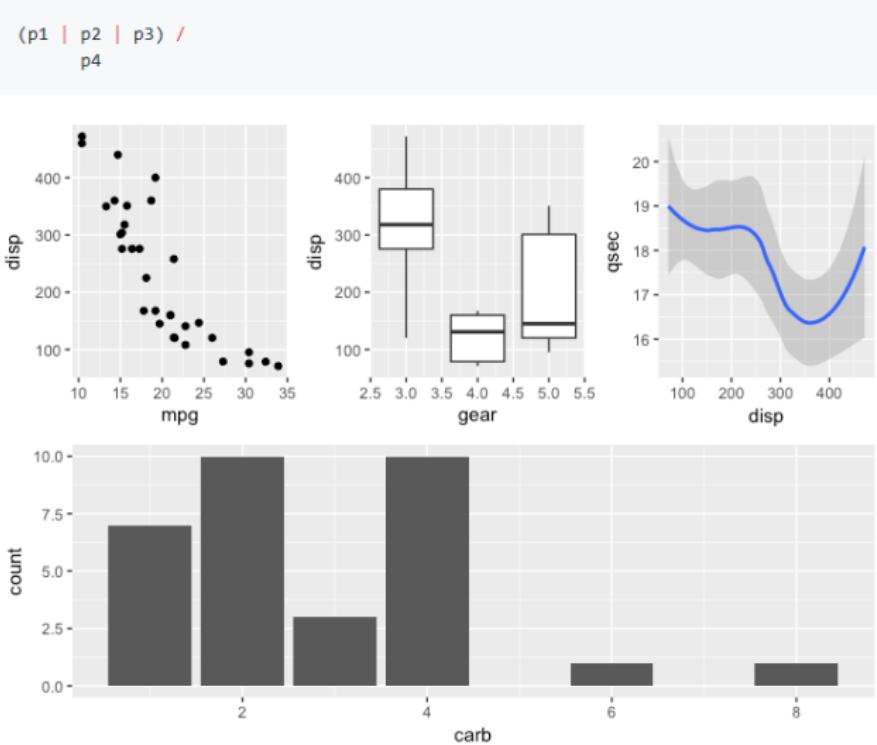


Composite figures

```
plot_grid(plot1, plot2, labels = "AUTO", ncol = 1)
```

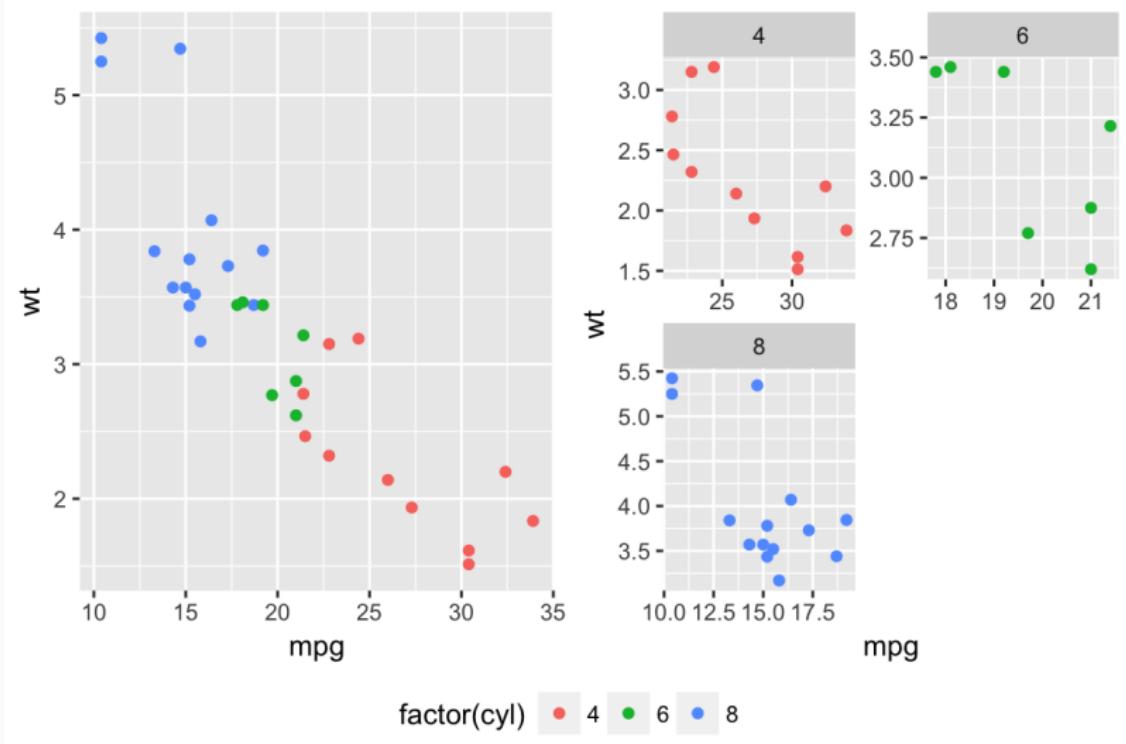


Composite figures: patchwork



<https://github.com/thomasp85/patchwork>

Composite figures: egg



<https://cran.r-project.org/web/packages/egg/index.html>

Saving plot

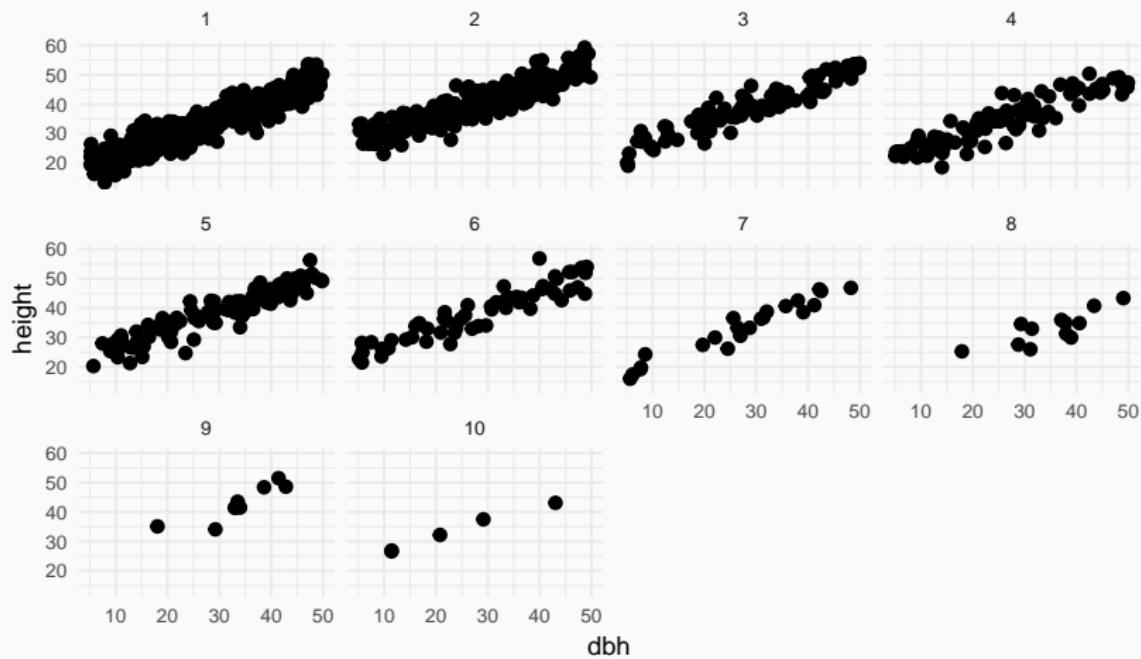
```
ggsave("myplot.pdf")
```

```
save_plot("myplot.pdf")
```

Facetting (small multiples)

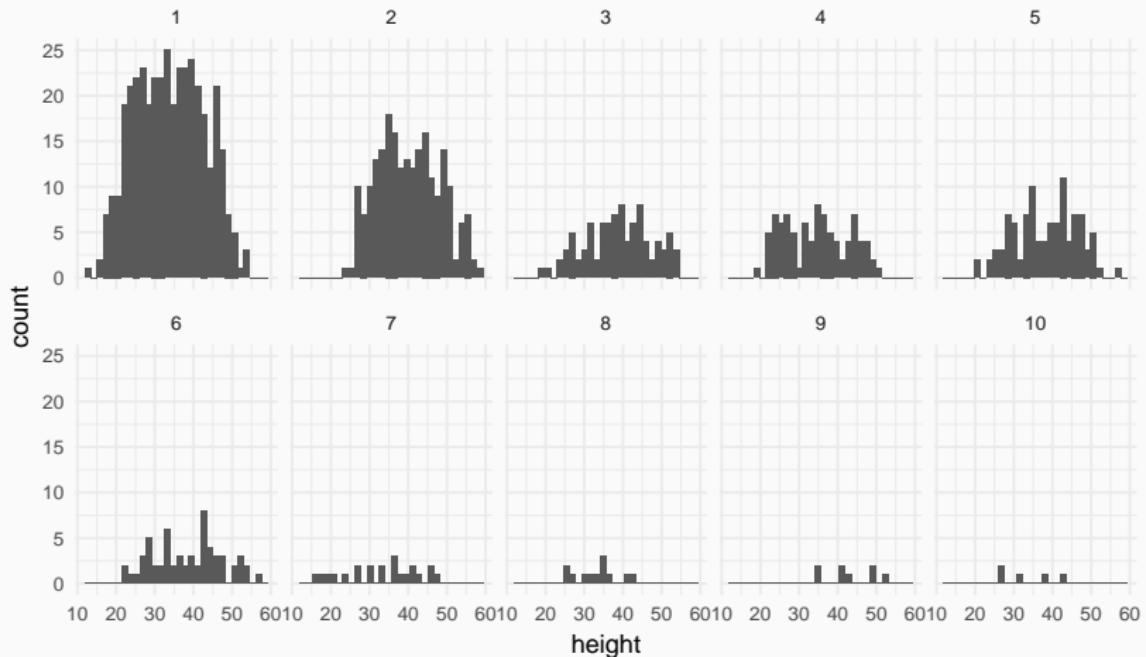
Facetting

```
ggplot(trees) + aes(dbh, height) +  
  geom_point() + theme_minimal(base_size = 8) +  
  facet_wrap(~plot)
```



Facetting

```
ggplot(trees) +  
  geom_histogram(aes(height)) + theme_minimal(base_size = 8) +  
  facet_wrap(~plot, nrow = 2)
```

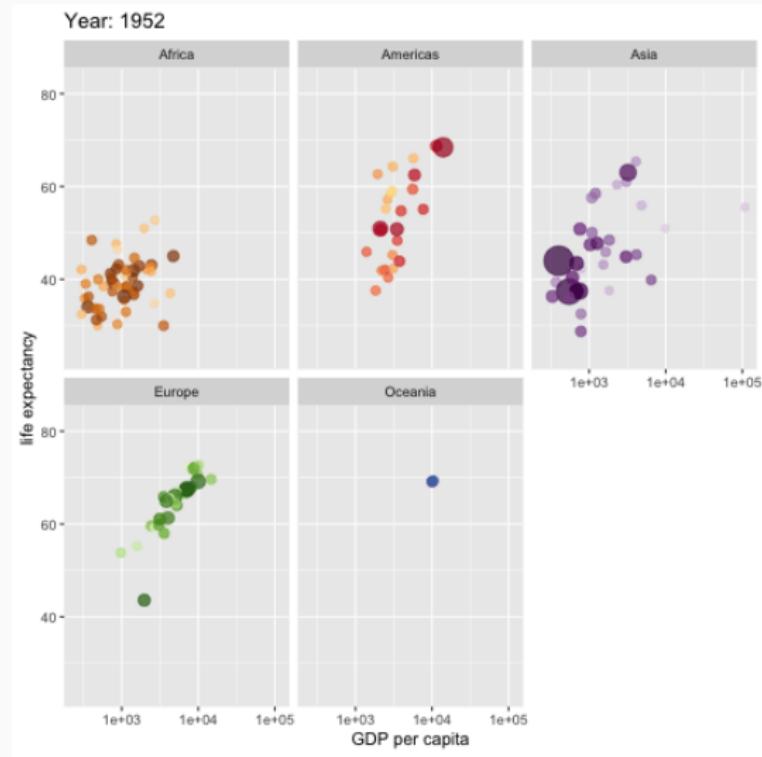


Interactivity: plotly

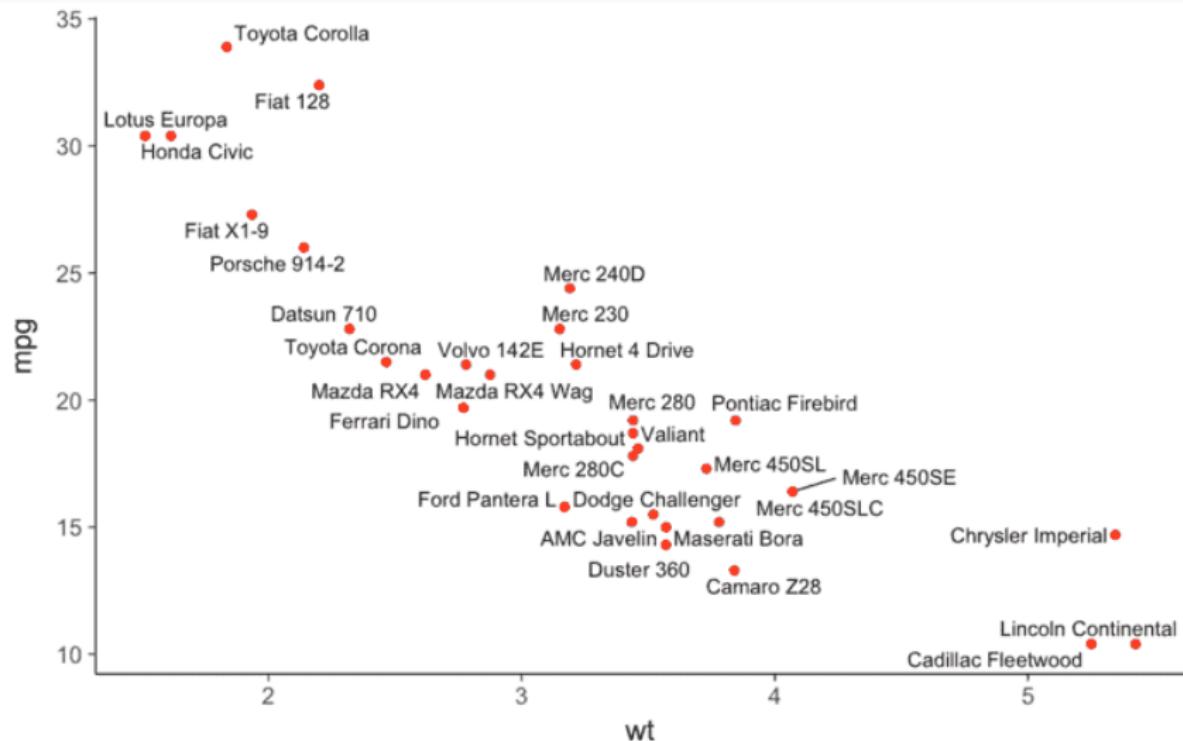
```
library(plotly)
myplot <- ggplot(trees) +
  aes(x = dbh, y = height) +
  geom_point()
ggplotly(myplot)
```

Animated graphs

<https://github.com/thomasp85/gganimate>



Automatic label placement



Many extensions!

<https://www.ggplot2-exts.org/>

40 registered extensions available to explore

Sort: Github stars Filter: search name, author, desc... Author Filter Tag Filter CRAN Only

Showing 40 of 40

gg3D (star)
3D perspective plots for ggplot2.
• **author:** Daniel Adler
• **tags:** 3D, Visualization
• **js libraries:**

ggQC (star)
Use ggQC to plot single, faceted and multi-layered quality control charts.
• **author:** Keith Gray
• **tags:** QC, XMR, XbarR, SixSigma, Visualization
• **js libraries:**

ggedit (star)
ggedit is aimed to interactively edit ggplot layers, scales and themes aesthetics
• **author:** jordic
• **tags:** visualization, interactive, shiny, general
• **js libraries:**

ggforce (star)
ggforce is aimed at providing missing functionality to ggplot2 through the extension system introduced with ggplot2 v2.0.0.
• **author:** thomasp85
• **tags:** visualization, general
• **js libraries:**

ggalt (star)
A compendium of 'geom's, 'coords' and 'stats' for 'ggplot2'.
• **author:** hrbrmstr
• **tags:** visualization, general
• **js libraries:**

ggraph (star)
A bridge between networkx and ggplot2.
• **author:** denigris
• **tags:** visualization, general
• **js libraries:**

ggstance (star)
ggstance implements horizontal versions of common ggplot2 geoms.
• **author:** forbesl
• **tags:** visualization, general
• **js libraries:**

ggrepel (star)
A tool for repelling overlapping text labels away from each other.
• **author:** slowflow
• **tags:** visualization, general
• **js libraries:**

ggraph (star)
ggraph is tailored at plotting graph-like data structures (graphs, networks, trees, hierarchies...).
• **author:** thomasp85
• **tags:** visualization, general
• **js libraries:**

ggpmisc (star)
Miscellaneous Extensions to 'ggplot2'.
• **author:**
• **tags:** visualization, general
• **js libraries:**

geomnet (star)
geomnet implements network visualizations in ggplot2 via geom.net.
• **author:** schmitt
• **tags:** visualization, general
• **js libraries:**

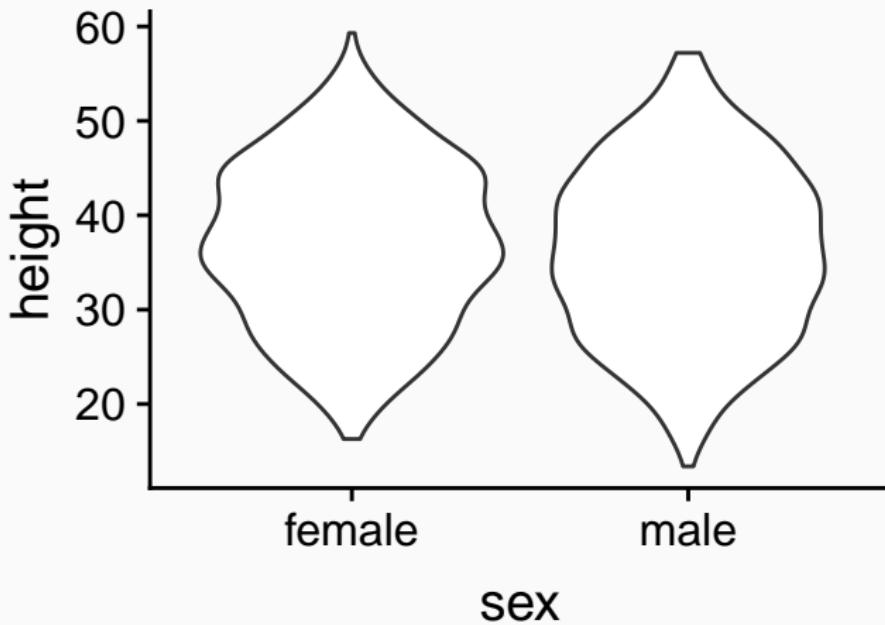
ggExtra (star)
ggExtra lets you add histograms to ggplot2.
• **author:** damian
• **tags:** visualization, general
• **js libraries:**

Summary

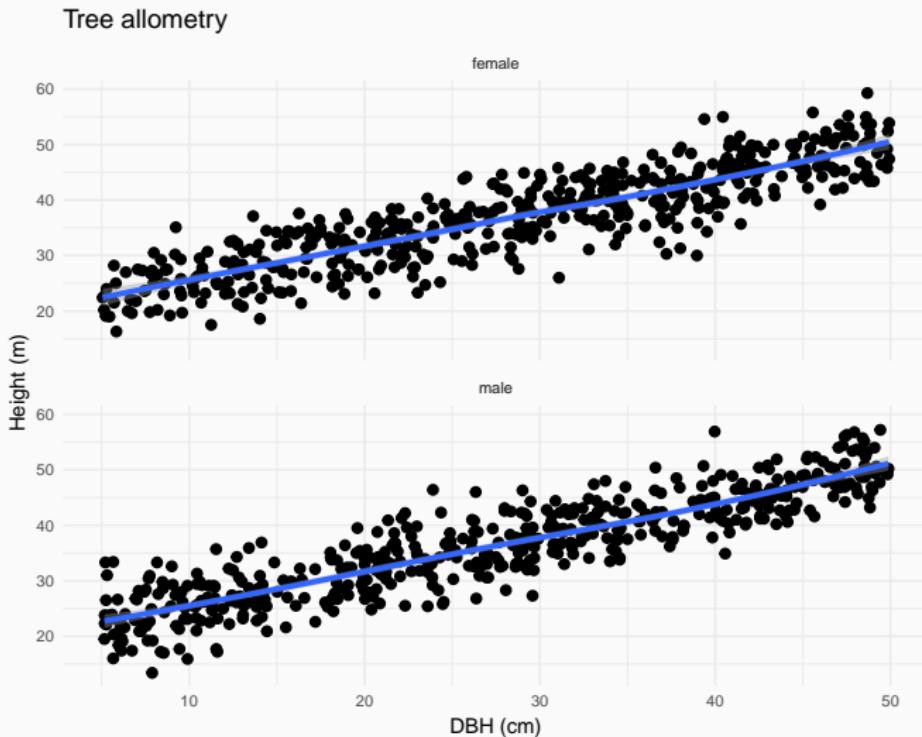
Grammar of graphics

- **Data** (tidy data frame)
- **Layers** (*geoms*: points, lines, polygons...)
- **Aesthetics** mappings (x, y, size, colour...)
- **Scales** (colour, size, shape...)
- **Facets** (small multiples)
- **Themes** (appearance)
- **Coordinate system** (Cartesian, polar, map projections...)

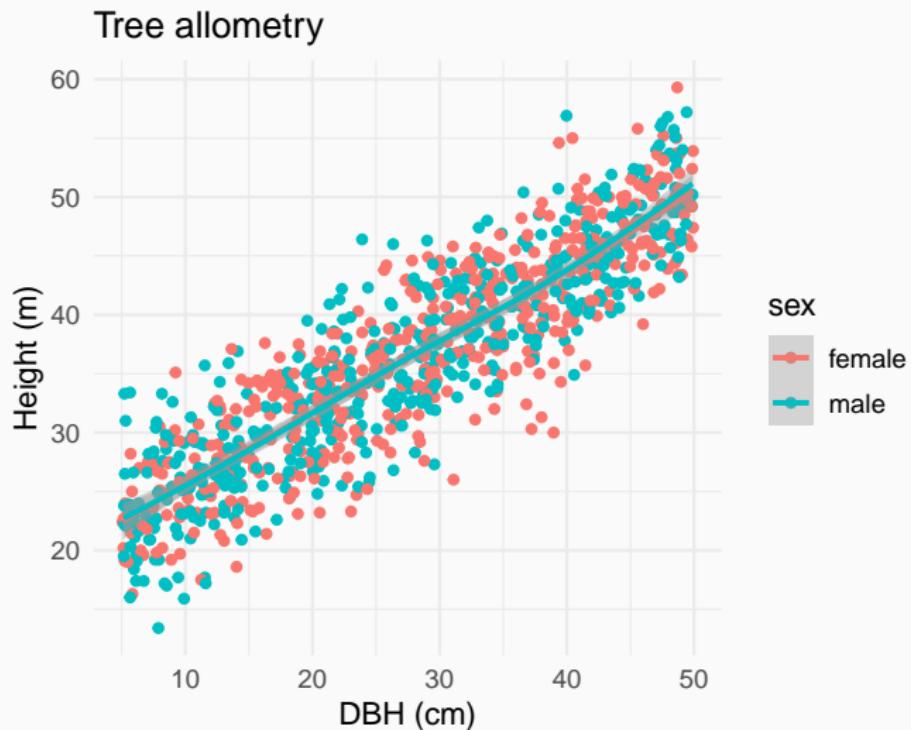
Exercise: make a plot like this one



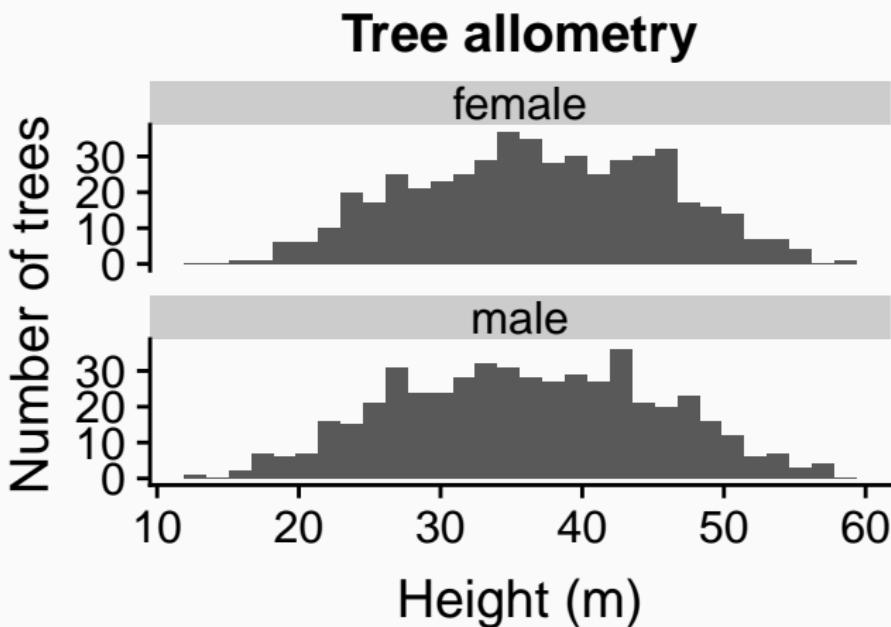
Exercise: make a plot like this one



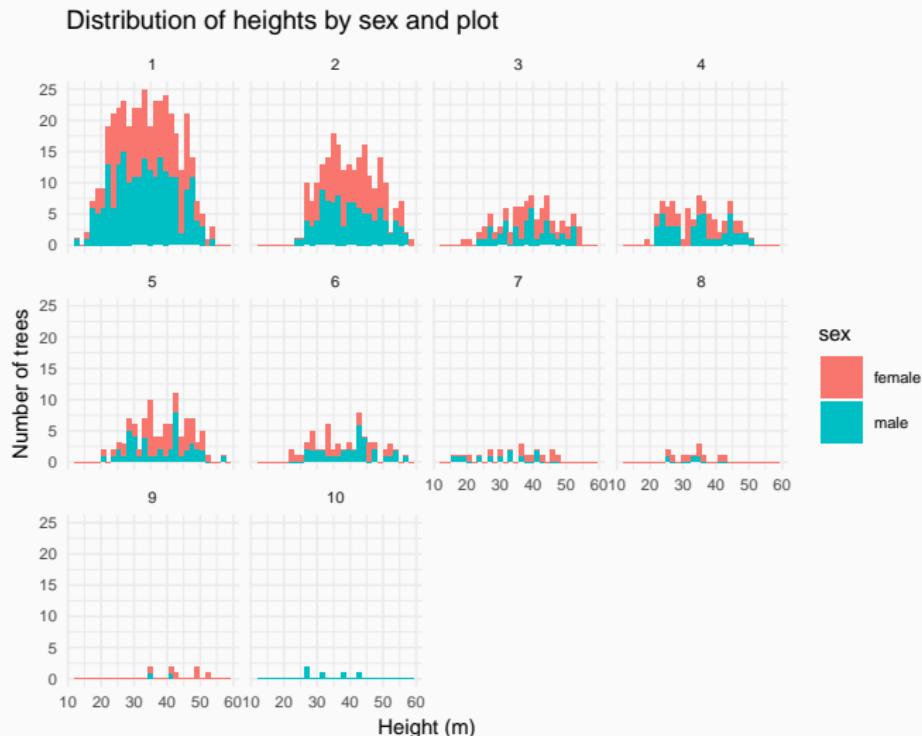
Exercise: make a plot like this one



Exercise: make a plot like this one

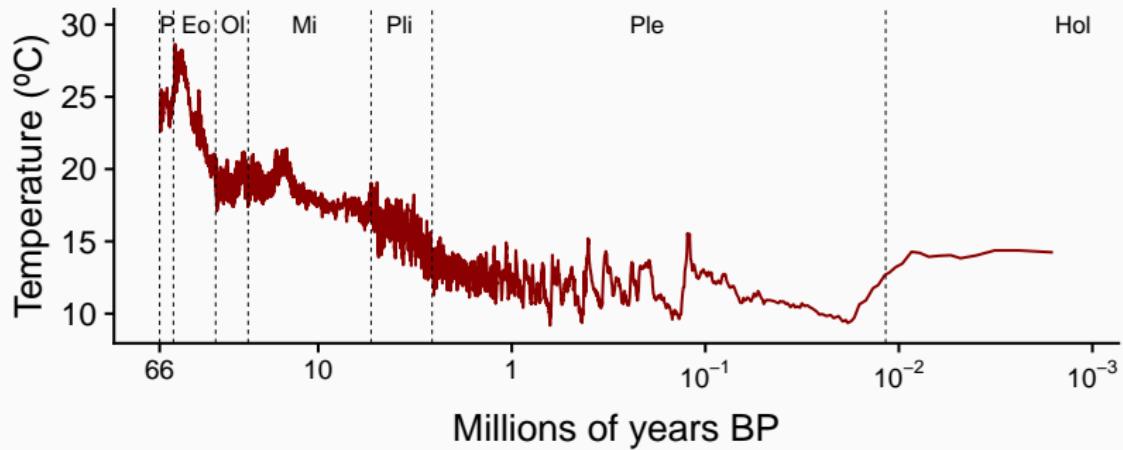


Exercise: make a plot like this one

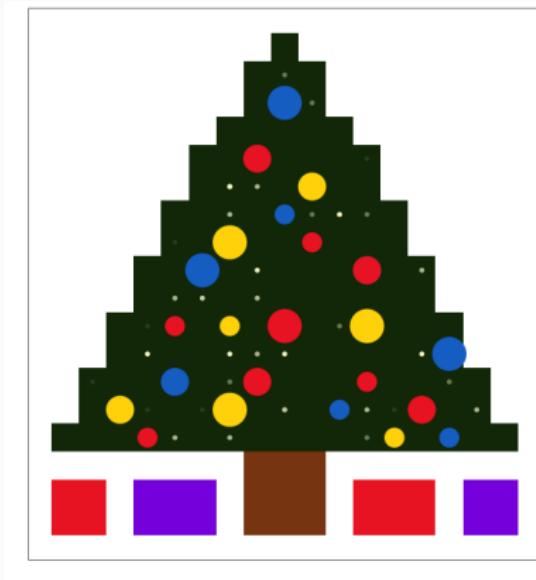


Exercise: make a plot like this one

Data from <http://www.columbia.edu/~mhs119/Sensitivity+SL+CO2/Table.txt>



Exercise: make a plot like this one



END



Slides and source code available at <https://github.com/Pakillo/ggplot-intro>