

Visualization

Week 15

PH 700A, Spring 2025

Rick Calvo

Table of contents

1	Data Visualization with R	2
1.1	Housekeeping	2
1.2	Packages	2
1.3	Communicating With Graphics!	2
1.4	Professional Appeal	3
1.5	Elegant Graphics for Data Analysis	3
1.6	Graphics Layering	4
1.7	<i>The Grammar of Graphics</i> , by Leland Wilkinson	4
1.8	ggplot2 Geometric Objects	5
1.9	Geom Layering	6
1.10	ggplot2 Coding Examples	6
1.11	“Mappings” vs. “Settings”	8
1.12	Additional Specifications	8
	1.12.1 Scaling	8
	1.12.2 Coordinate Systems	8
	1.12.3 Faceting	8
1.13	Default Code	9
1.14	Coordinate Systems	9
	1.14.1 Cartesian Coordinates	10
	1.14.2 Non-linear Coordinates	11
1.15	Faceting	11
1.16	Individual Figures	12
1.17	Faceted Figure	12
2	Diagrams with DiagrammeR	14
2.1	Purpose	14
2.2	GraphViz DOT Syntax	15
2.3	Diagram Rendering	15
2.4	DOT Code Explanation	17
2.5	Histogram Examples	18
2.6	geom_bar() vs. geom_histogram()	19
2.7	geom_bar() with specific Y-values	21

2.8	Scatterplots	22
3	Final Comments	23
3.1	R for Visualizations	23
3.2	Every Graphic is its Own Story	23
3.3	Data Analysis as an “Art Form”	24
3.4	Be Curious and Bold	24
4	Appendix	25
4.1	ggplot2 Links	25
4.2	DiagrammeR Links	25

1 Data Visualization with R

1.1 Housekeeping

- Today is the last meeting
- No final project
- There is no final exam!
- Student survey reminder
- Course materials on Canvas and GitHub
- Zoom Lecture Recordings

1.2 Packages

- `library(ggplot2)`
- `library(DiagrammeR)`

1.3 Communicating With Graphics!

- R is particularly known for its ability to render graphics.
- Graphics allow for rapid exploration of data
- For non-analysts, graphical representation of results will be preferred.
- Effective figures will improve journal and conference acceptance rates
- Many venues require “visual abstracts” to accompany the manuscripts/presentations

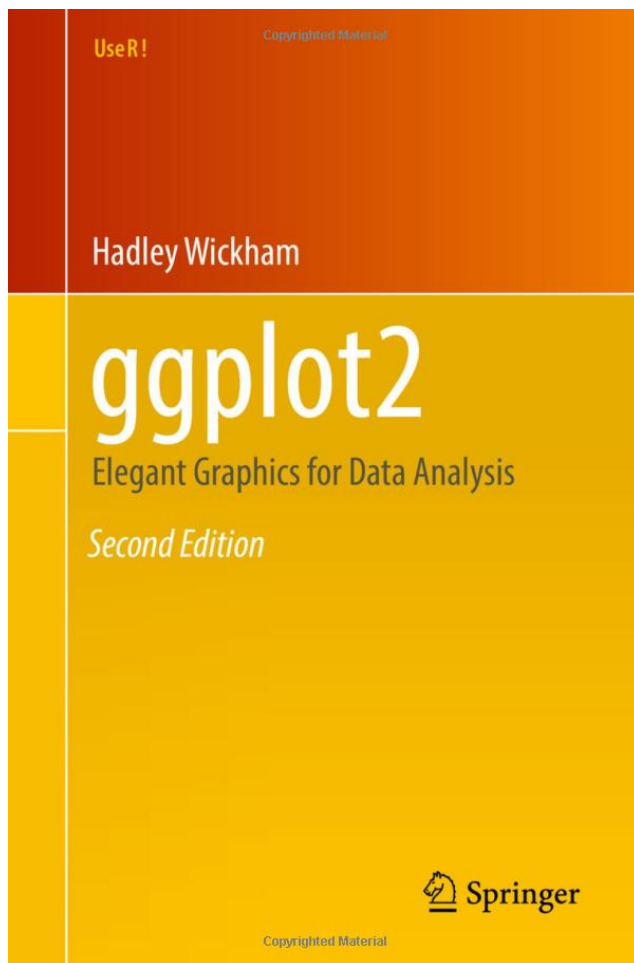
1.4 Professional Appeal

- The ability to convey information is a powerful skill that can only be improved through experience
- It extends beyond coding and outputting results
- Development of the ability to create images that can be easily interpreted and understood is recommended

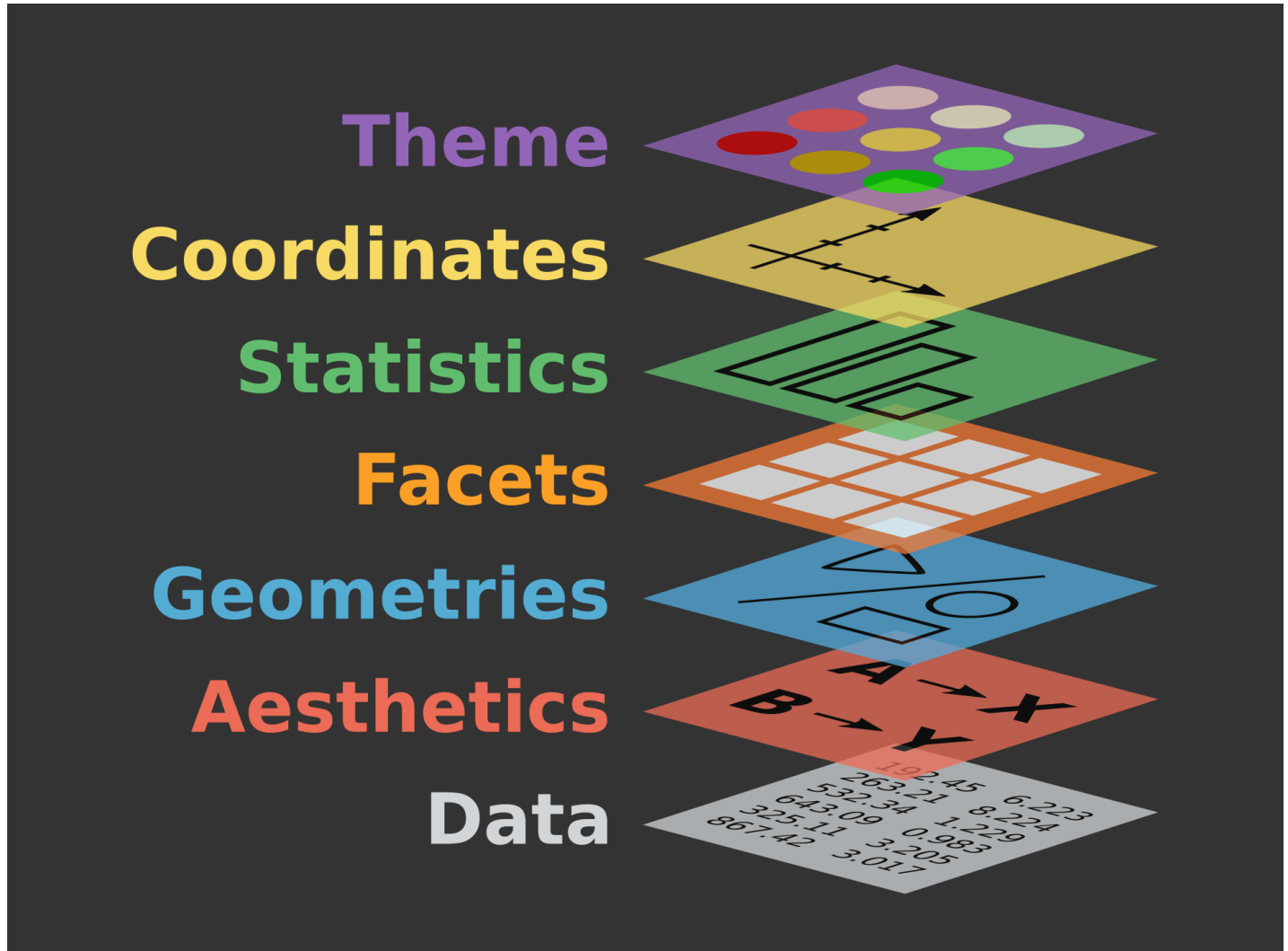
1.5 Elegant Graphics for Data Analysis

Online: 3rd Edition <https://ggplot2-book.org/>

In-print: 2nd Edition



1.6 Graphics Layering



Source: [Quebec Center for Biodiversity Science](#)

1.7 *The Grammar of Graphics*, by Leland Wilkinson

Has served as a philosophy for visualizing data and a framework for coding statistical graphics.

Primary elements:

1. The Data: Your data frame
2. Aesthetic Mapping: How the *columns* will be coordinated to create a visual
3. Geometry Settings: How *observations* will be displayed









```
ggplot(data = dataframe,
       mapping = aes(x = var1,
                     y = var2,
                     color = catvar1)) +
  geom_point()
```

R will look at the parameters provided and try to complete the task:

1. `ggplot()` is called first and the `data` defined
2. `var1`, `var2`, and `catvar1` columns are assigned to arguments and found to be *numeric*, *numeric*, and *factor*, respectively
3. `geom_point()` has no arguments so it uses the default options to place **observations** on an x,y scatterplot. Since the `+` sign is used, it will inherit previously-stated parameters for its task

1.8 ggplot2 Geometric Objects

Anything after the `geom_` part can be modified with the following to create a different type of graph. See <https://ggplot2.tidyverse.org/reference/> for all options. Examples include:

geom_		geom_	
point		bar	
line		histogram	
density		violin	
dotplot		boxplot	

Focus today: `geom_histogram()`, `geom_bar()`, and `geom_point()`.

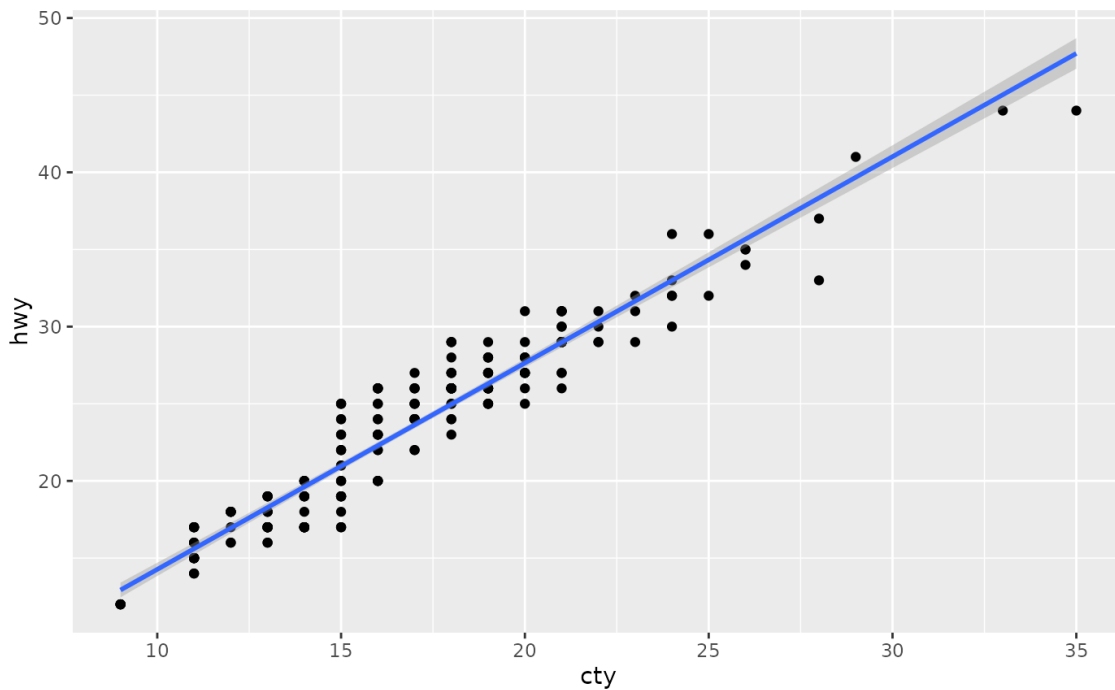
1.9 Geom Layering

Objects can be classified by how many dimensions they take:

- 0 dims: points, text
- 1 dim: lines, paths
- 2 dims: polygons, intervals

Layering of *mapped aesthetics* and *geoms* can address many dims.

```
ggplot(data = df, aes(x = cty, y = hwy)) +  
  geom_point() +  
  geom_smooth(formula = y ~ x, method = "lm")
```



1.10 ggplot2 Coding Examples

The R Graph Gallery <https://r-graph-gallery.com/> - A collection of charts made with the R programming language. Sample code is included with each figure.

The R Graph Gallery



Welcome to the R graph gallery, a collection of charts made with the [R programming language](#). Hundreds of charts are displayed in several sections, always with their reproducible code available. The gallery makes a focus on the tidyverse and [ggplot2](#). Feel free to suggest a chart or [report a bug](#): any feedback is highly welcome! Note that if you like this gallery, you'll love my [newsletter](#)!

Subscribe

One short, impactful dataviz tip delivered to your inbox each Saturday.

THE LARGEST COLLECTION OF R CHART EXAMPLES

The R Graph Gallery boasts the **most extensive compilation of R-generated graphs** on the web.

Featuring over **400 examples**, our collection is meticulously organized into nearly **50 chart types**, following the [data-to-viz](#) classification. Each example comes with reproducible code and a detailed explanation of its functionality.

We begin each chart type with a **foundational tutorial** that outlines its core structure and purpose. Once you've grasped the basics, our **step-by-step guides** offer insights into elementary customizations. This ensures that your charts not only

1.11 “Mappings” vs. “Settings”

Mapping refers to the assignment of an *argument* to a *column in a data frame*. The value of the column is *variable* which means it is allowed to range.

Settings are pre-specified and finite items that can only be specified within the *geometry*.

WRONG

```
ggplot(data = df,
       mapping = aes(x = x,
                     y = y,
                     color = "blue"
                     )
) +
  geom_point()
```

RIGHT

```
ggplot(data = df,
       mapping = aes(x = x,
                     y = y
                     )
) +
  geom_point(color = "blue")
```

1.12 Additional Specifications

These may have to be *situationally* incorporated based on the `geom_` type and `mapping` method.

1.12.1 Scaling

The sizing of items. A scale is applied to each aesthetic mapping. If nothing is specified, it takes a *default* value.

1.12.2 Coordinate Systems

This regards an *x,y coordinate* system and the arrangement of objects in relation to each other. Each layer can only capture up to two dimensions (x,y).

1.12.3 Faceting

Displays stratified figures to depict different subsets in the same image.

1.13 Default Code

Arguments that are not specified will be filled in by a default value that varies by `mapping` and `geom_` type.

Viewing R syntax shorthand is not beginner-friendly so read the help file [`help(ggplot2)`] for the defaults.

Shorthand

```
ggplot(df, aes(var1, var2)) +  
  geom_point() +  
  stat_smooth(method = lm) +  
  scale_x_log10() +  
  scale_y_log10()
```

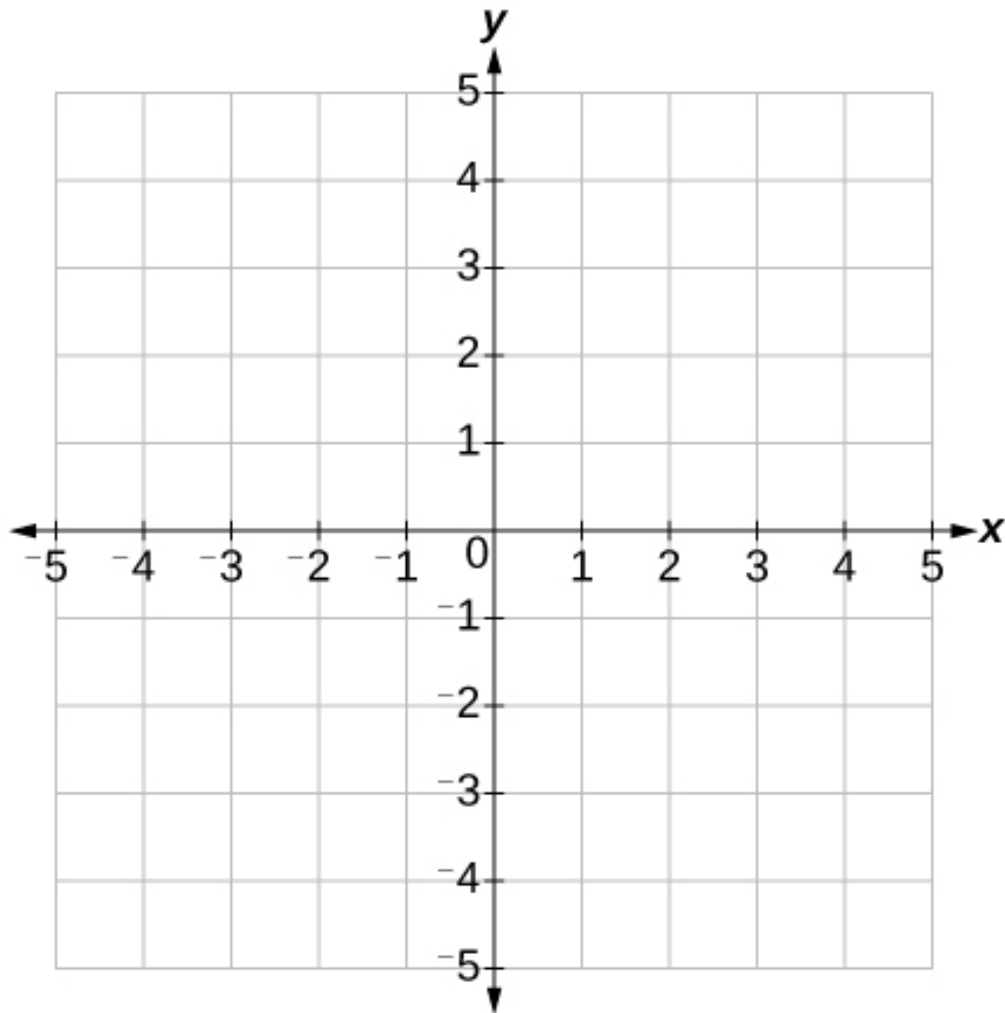
Verbose

```
ggplot() +  
  layer(  
    data = df,  
    mapping = aes(x = var1, y = var2),  
    geom = "point",  
    stat = "identity",  
    position = "identity"  
  ) +  
  layer(  
    data = df,  
    mapping = aes(x = var1, y = var2),  
    geom = "smooth",  
    position = "identity",  
    stat = "smooth",  
    method = lm  
  ) +  
  scale_y_log10() +  
  scale_x_log10() +  
  coord_cartesian()
```

1.14 Coordinate Systems

Plots can either be **Cartesian** or **Non-linear**.

1.14.1 Cartesian Coordinates



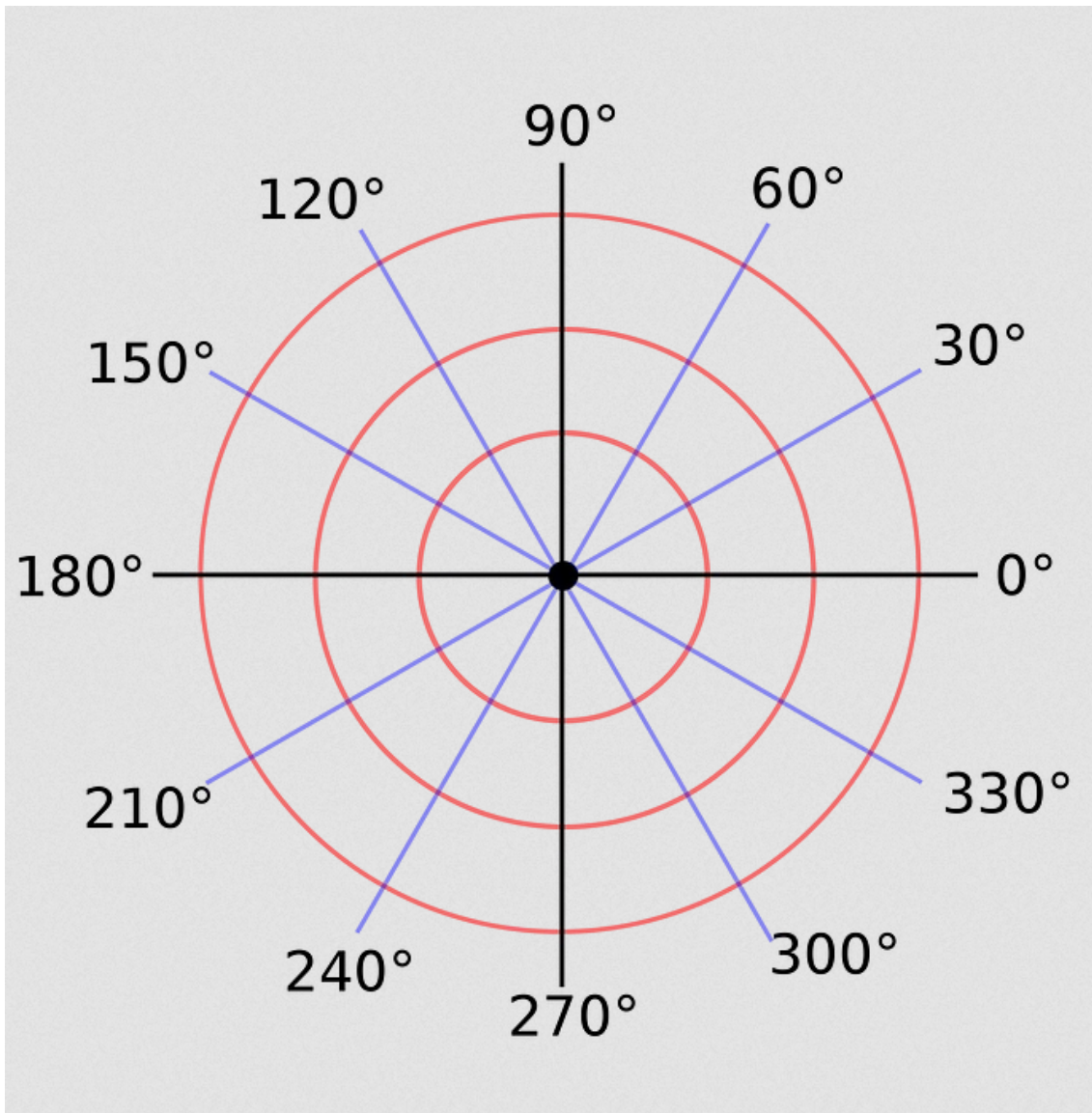
default

`coord_flip()`: x and y axes flipped

`coord_fixed()`: fixed aspect ratio

`coord_cartesian()`:

1.14.2 Non-linear Coordinates



`coord_quickmap():`

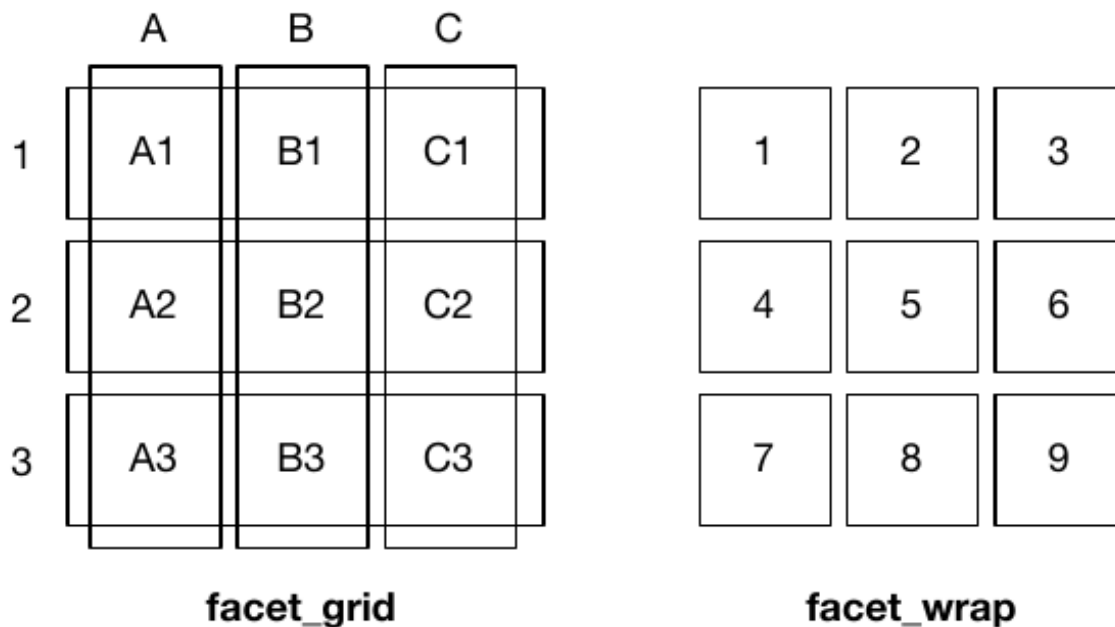
for map projections

`coord_polar():` circular projection

1.15 Faceting

Displays graphs as separate frames rather than overlaying in a single image.

- `facet_null()` is the default: shows a single plot with aggregated data or overlays by categorical variables
- `facet_wrap()` shows a multi-panel graph for valid combinations of categorical variables
- `facet_grid()` requires a formula; shows a matrix of graphs for every combination of categorical variables



1.16 Individual Figures

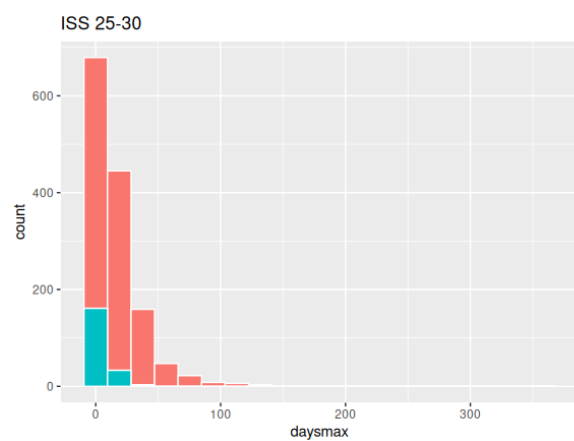
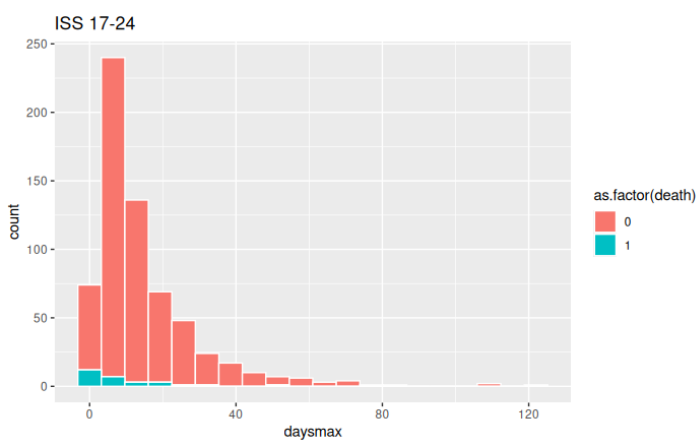
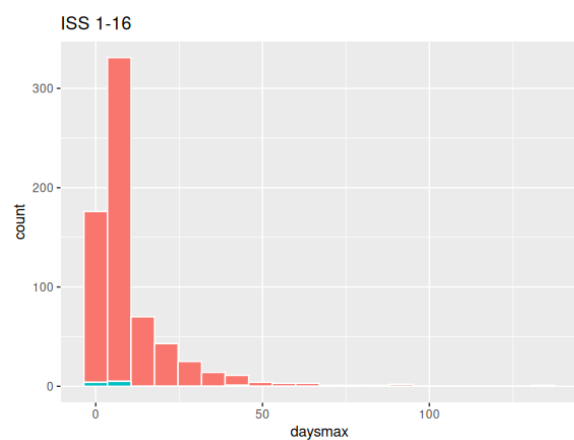
```
df %>% filter(iss5cat == 0) %>%
  ggplot(aes(x = daysmax, fill = as.factor(death))) +
  geom_histogram(colour = 'white', bins = 20) +
  ggtitle("ISS 1-16")

df %>% filter(iss5cat == 1) %>%
  ggplot(aes(x = daysmax, fill = as.factor(death))) +
  geom_histogram(colour = 'white', bins = 20) +
  ggtitle("ISS 17-24")

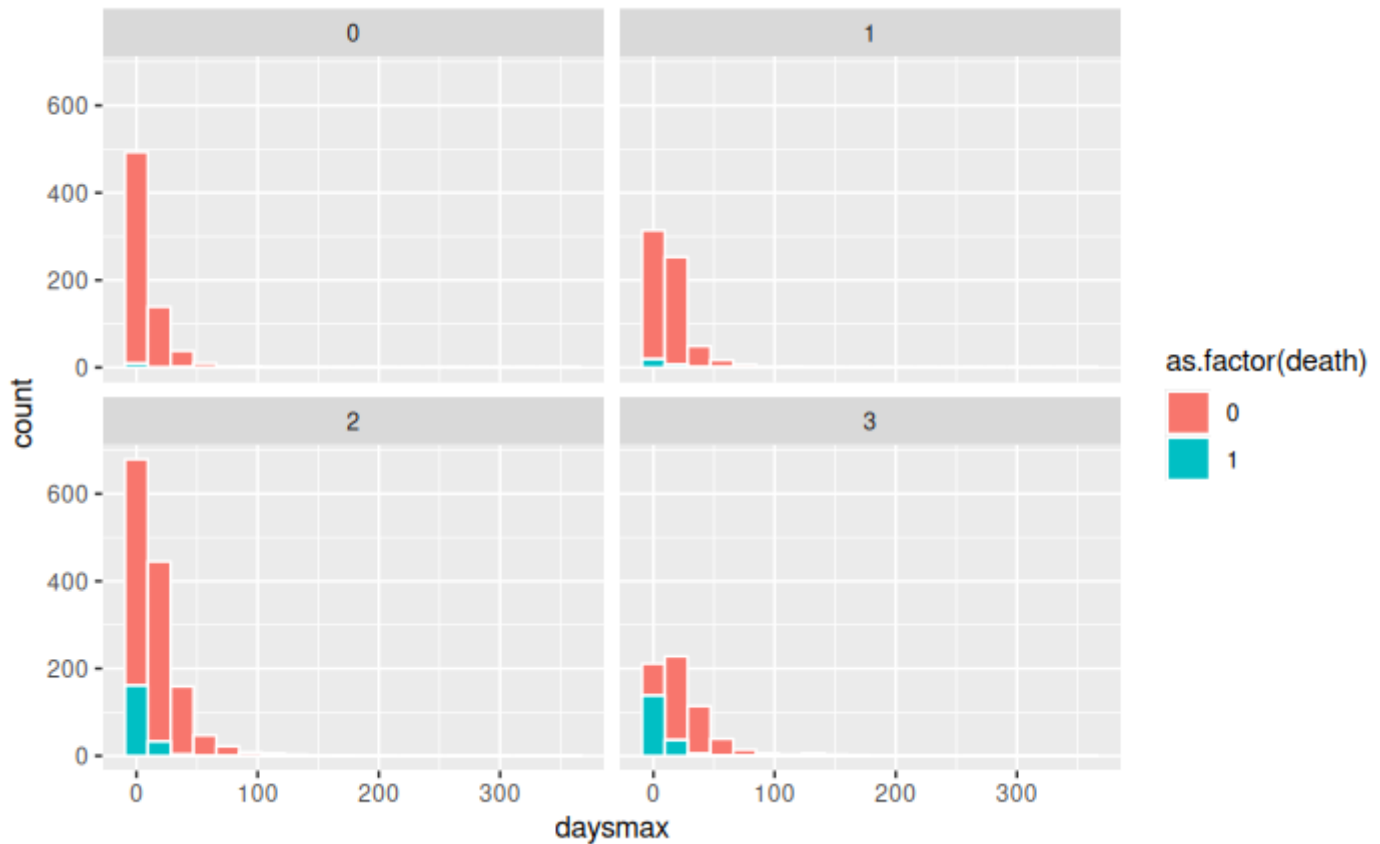
df %>% filter(iss5cat == 2) %>%
  ggplot(aes(x = daysmax, fill = as.factor(death))) +
  geom_histogram(colour = 'white', bins = 20) +
  ggtitle("ISS 25-30")

df %>% filter(iss5cat == 3) %>%
  ggplot(aes(x = daysmax, fill = as.factor(death))) +
  geom_histogram(colour = 'white', bins = 20) +
  ggtitle("ISS 31-74")
```

1.17 Faceted Figure



```
df %>%
  ggplot(aes(x = daysmax, fill = as.factor(death))) +
  geom_histogram(colour = 'white', bins = 20) +
  facet_wrap(~iss5cat)
```



2 Diagrams with DiagrammeR

2.1 Purpose

Sometimes you just want to convey a mechanism without having to use data.

`library(DiagrammeR)` allows for the in-syntax creation of diagrams.

Useful if:

- You want to stay in the R environment
- You appreciate the [Markdown](#) style of coding, annotating, and documenting
- You are compiling a single `rmarkdown` or `quarto` file and want to refrain from fragmenting your analysis/results

- You don't want to deal with the minutia of modifying font size, shapes, colors, etc. and just want to maintain a *plaintext* document

i Note

Aesthetics are generally not a priority with **DiagrammeR**. We are simply attempting to create quick and functional diagrams to assist in our communication. Development of high quality imagery should be reserved for more specialized programs.

2.2 GraphViz DOT Syntax

DiagrammeR has two markdown systems:

- ***Graphviz*** (Today's Focus)
- *MermaidJS*

Graphviz uses the *DOT Language*

- Object-oriented graph description language
- DOT stands for “_D_AG [directed acyclic graph] _O_f _T_omorrow”
- Items in the graph can be manipulated with freetext
- Code is rendered from top to bottom
- Each line pertains to a separate item

2.3 Diagram Rendering

```
grViz("digraph {  
  
graph[layout = dot, rankdir = BT]  
  
node [shape = rectangle, style = filled]  
  
node [fillcolor = White]  
  ep1 [label = 'Humberto']  
  ep2 [label = 'Hector']  
  hp1 [label = 'Ben']  
  hp2 [label = 'Jerel']  
  en1 [label = 'Elizabeth']  
  en2 [label = 'Goran']  
  en3 [label = 'Edward']  
  
node [fillcolor = Gray]  
  ad1 [label = 'Emily']
```

```

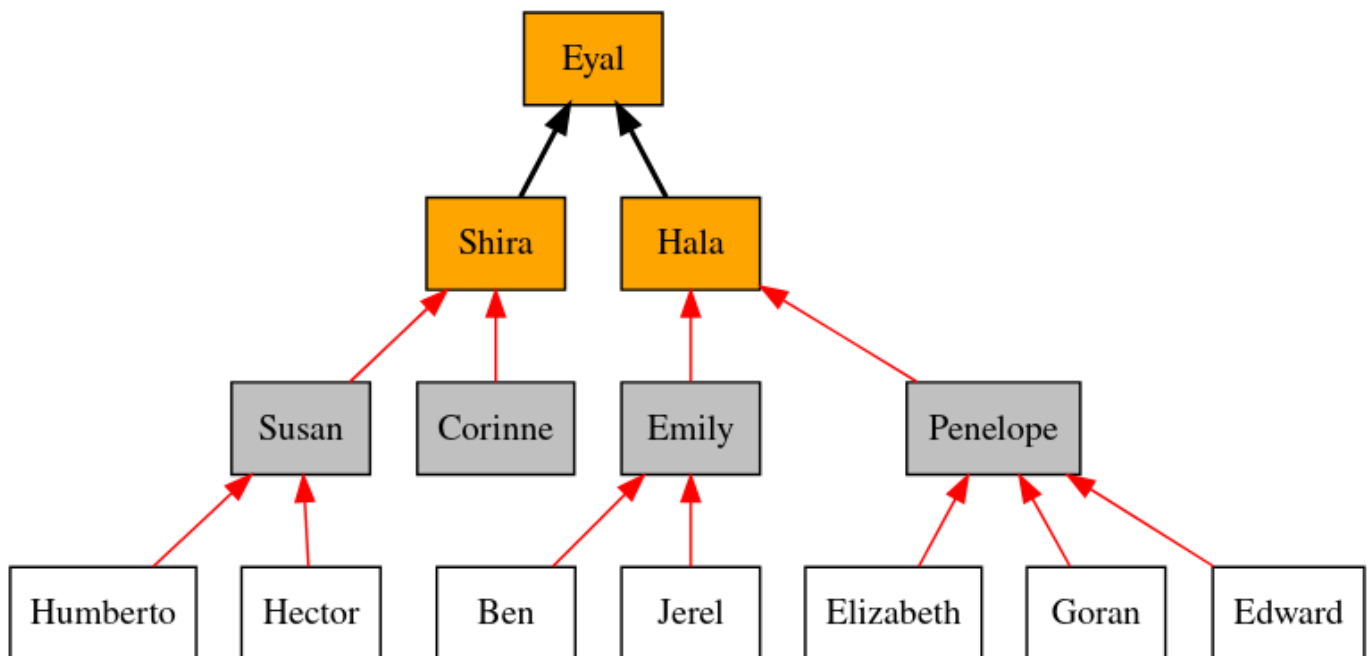
ad2 [label = 'Susan']
ad3 [label = 'Corinne']
ad4 [label = 'Penelope']

node [fillcolor = Orange]
  dir1 [label = 'Eyal']
  dir2 [label = 'Shira']
  dir3 [label = 'Hala']

edge [color=red, penwidth=1, constraint = TRUE]
  {ep1 ep2} -> ad2
  {hp1 hp2} -> ad1
  {en1 en2 en3} -> ad4
  {ad2 ad3} -> dir2
  {ad1 ad4} -> dir3

edge [color=black, penwidth=2, constraint = TRUE]
  {dir2 dir3} -> dir1
})

```



2.4 DOT Code Explanation

```
grViz("digraph {  
  
graph[layout = dot, rankdir = BT]  
  
node [shape = rectangle, style = filled]  
  
node [fillcolor = White]  
  ep1 [label = 'Humberto']  
  ep2 [label = 'Hector']  
  hp1 [label = 'Ben']  
  hp2 [label = 'Jerel']  
  en1 [label = 'Elizabeth']  
  en2 [label = 'Goran']  
  en3 [label = 'Edward']  
  
node [fillcolor = Gray]  
  ad1 [label = 'Emily']  
  ad2 [label = 'Susan']  
  ad3 [label = 'Corinne']  
  ad4 [label = 'Penelope']  
  
node [fillcolor = Orange]  
  dir1 [label = 'Eyal']  
  dir2 [label = 'Shira']  
  dir3 [label = 'Hala']  
  
edge [color=red, penwidth=1, constraint = TRUE]  
  {ep1 ep2} -> ad2  
  {hp1 hp2} -> ad1  
  {en1 en2 en3} -> ad4  
  {ad2 ad3} -> dir2  
  {ad1 ad4} -> dir3  
  
edge [color=black, penwidth=2, constraint = TRUE]  
  {dir2 dir3} -> dir1  
}").
```

`grViz()` is the R call for the function, but shifts to *DOT syntax* within `()`.

- `"digraph{}"` (directed graph) or `"graph{}"` must be specified to define what type of figure to create

All subsequent code defines the elements depicted in the figure.

- `graph` declares what [layout engine](#) you will use
- `node` defines the shape attributes

- `edge` defines the connecting line attributes

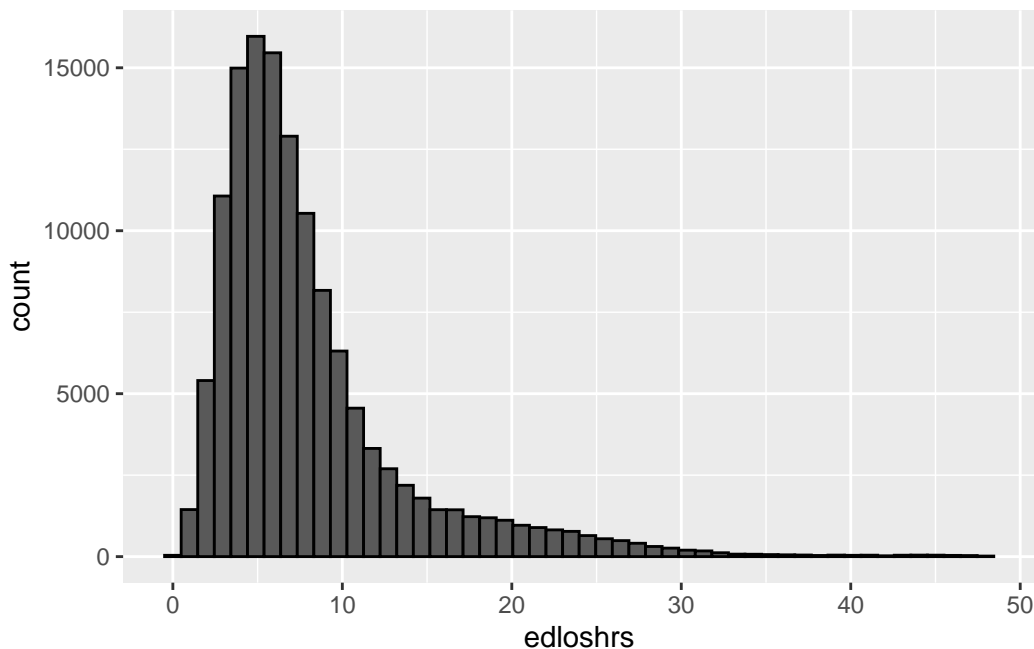
Previously-defined element attributes are applied to subsequent elements unless overwritten by a new declaration.

- *Objects* should be nested in nodes and provided a unique name
- *Edges* to connect **objects** are defined as a `->` and must go from left-to-right
- Optional elements include **subgraph** and **cluster** for complex diagrams

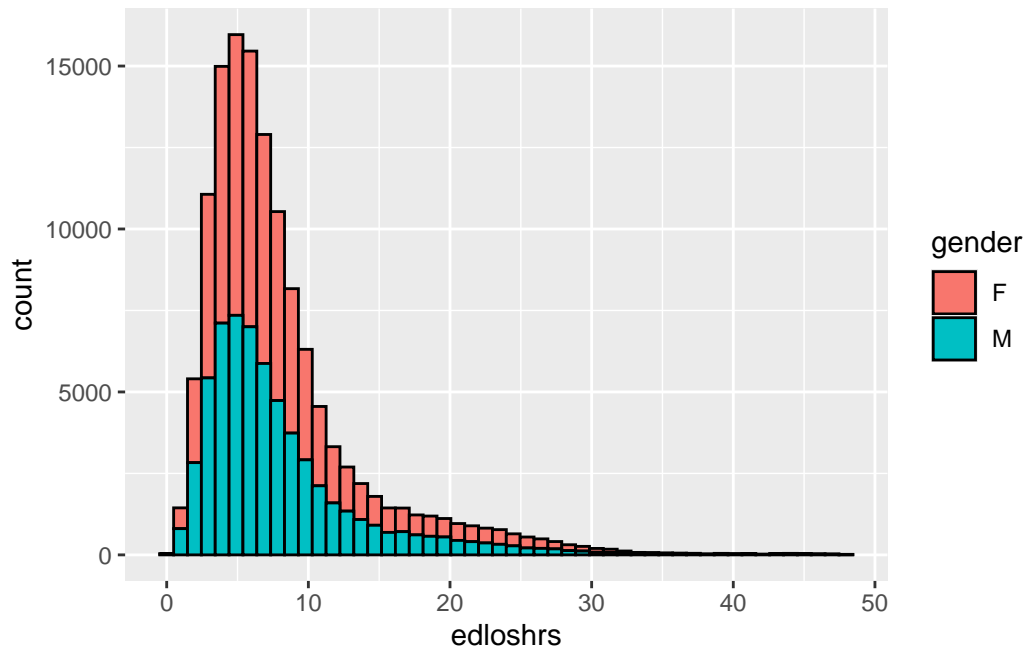
2.5 Histogram Examples

Additional parameters can always be specified to “enhance” a figure.

```
ggplot(data = df,
       mapping = aes(x = edloshrs)) +
  geom_histogram(bins = 50,
                colour = 'black')
```



```
ggplot(data = df,
       mapping = aes(x = edloshrs, fill = gender)) +
  geom_histogram(bins = 50,
                colour = 'black')
```

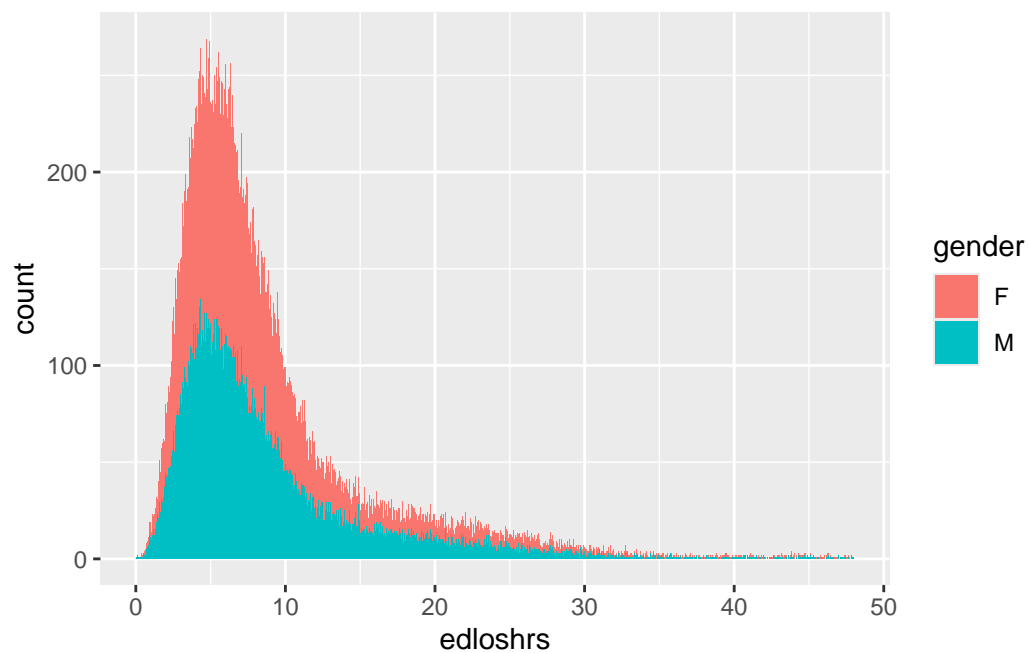


2.6 geom_bar() vs. geom_histogram()

`geom_histogram()` will *bin* frequencies over continuous X values

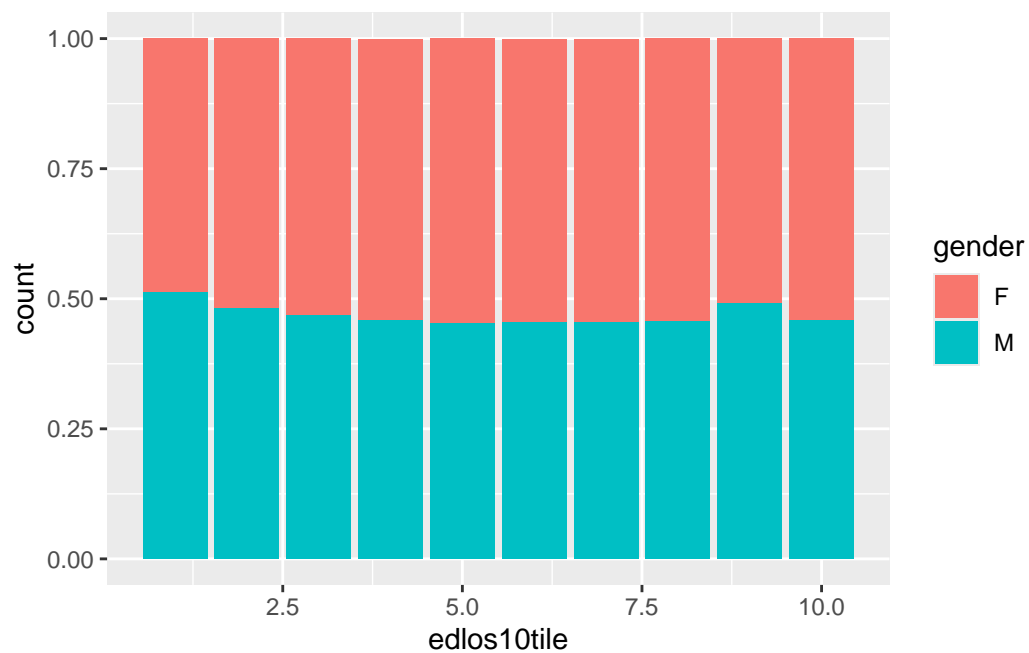
`geom_bar()` will use raw X values; Unless Y is specified, counts are used

```
ggplot(data = df,  
       mapping = aes(x = edloshrs, fill = gender)) +  
  geom_bar()
```

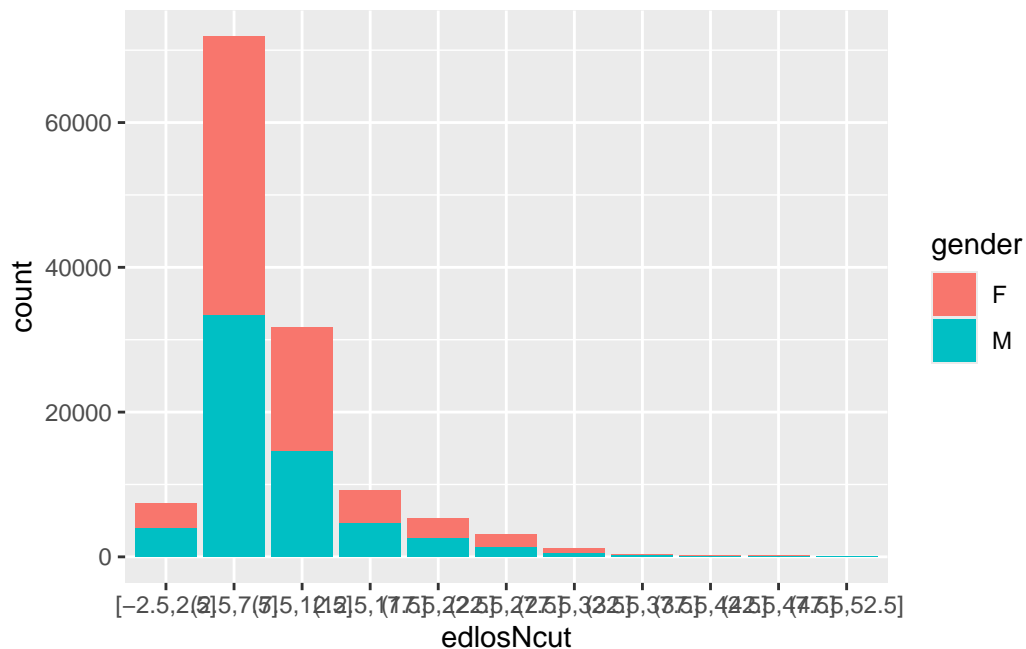


```
df <- df %>%
  mutate(edlos10tile = ntile(edloshrs, 10),
         edlosNcut = cut_width(edloshrs, 5)
  )
```

```
ggplot(data = df,
       mapping = aes(x = edlos10tile, fill = gender)) +
  geom_bar(position = "fill")
```



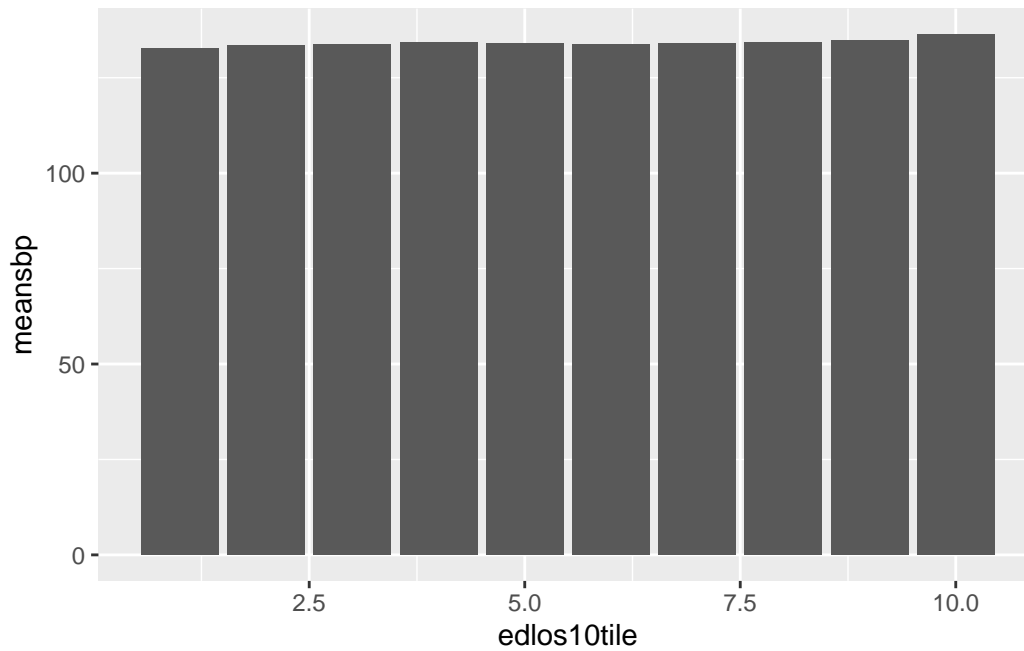
```
ggplot(data = df,
       mapping = aes(x = edlosNcut, fill = gender)) +
  geom_bar()
```



2.7 geom_bar() with specific Y-values

```
tempdf <- df %>%
  mutate(edlos10tile = ntile(edloshrs, 10)) %>%
  group_by(edlos10tile) %>%
  summarise(meansbp = mean(sbp)) %>%
  ungroup()

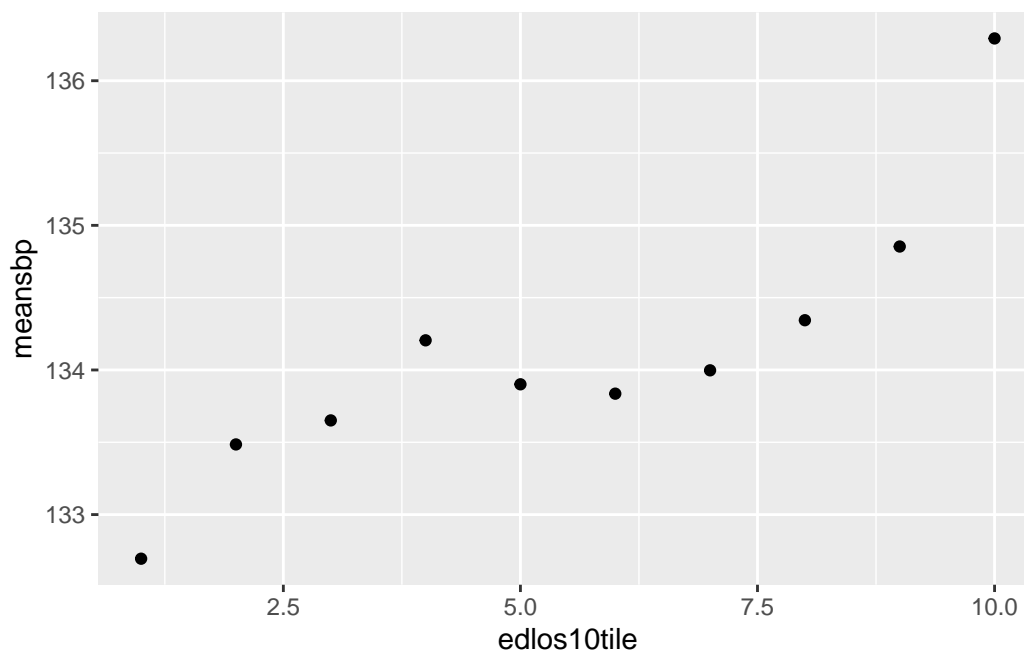
ggplot(data = tempdf,
       mapping = aes(x = edlos10tile, y = meansbp)) +
  geom_bar(stat = "identity") +
  scale_y_continuous(labels = abs)
```



2.8 Scatterplots

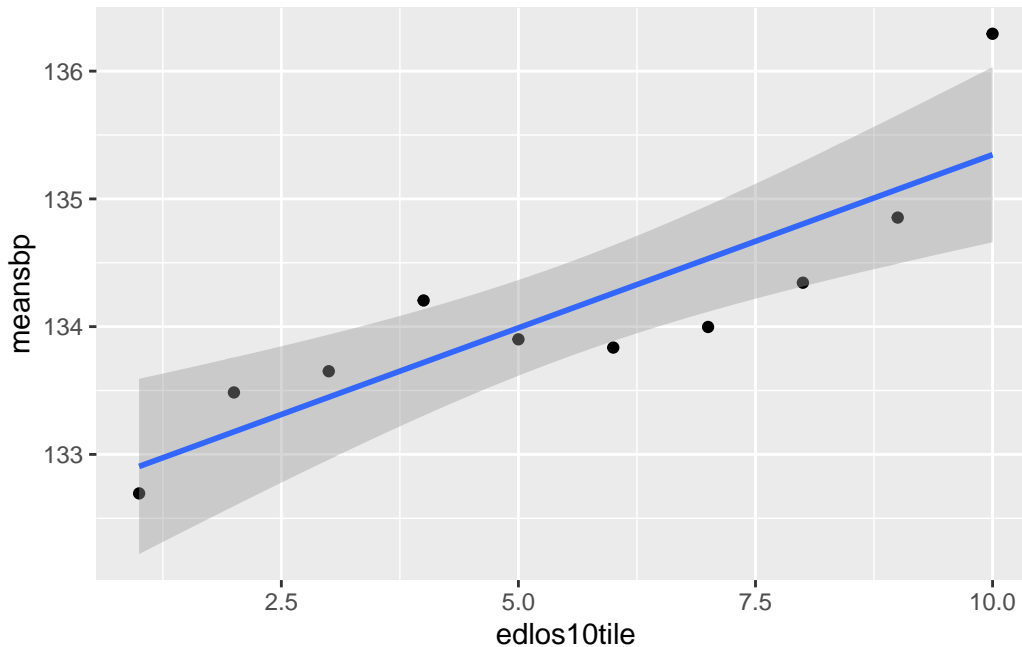
Axis are automatically scaled when using `geom_point()`

```
ggplot(data = tempdf,  
       mapping = aes(x = edlos10tile, y = meansbp)) +  
  geom_point()
```



Additional layers can be added, including a regression line.'

```
ggplot(data = tempdf,  
       mapping = aes(x = edlos10tile, y = meansbp)) +  
  geom_point() +  
  geom_smooth(method='lm', formula= y~x)
```



3 Final Comments

3.1 R for Visualizations

ggplot2 is the core of almost all visualization packages.

Other packages exist to facilitate creation of unique imagery.

Images should complement your results and be strong enough to stand alone.

3.2 Every Graphic is its Own Story

Intelligent Brevity - Images should reduce complexity, but the *essence* should remain - Removal of components that do not fulfill a *purpose* in communication should be removed - Retention of details should be based on conveying a critical aspect

Intrigue vs. Results - Sometimes it's not just about putting data in a picture - Should raise curiosity (but not frustration), then you can distill results

Subjectivity \neq **Arbitrary** - You can do what you want, but make sure you can justify it

Know Your Audience - Every discipline has an audience - Understand what is permissible in each discipline



Figure 1: Alberto Cairo, Professor at the University of Miami

3.3 Data Analysis as an “Art Form”

Like model development, everyone does it differently.

Every investigator has the freedom to evaluate and emphasize different things.

R is an embodiment of this versatility and flexibility.

3.4 Be Curious and Bold

There is no single correct path.

There is no wrong way to learn.

Science is incremental and so is learning.

As long as you’re curious, you will do well in this world.

I wish you all the best in the future and know that each and every one of you will do great things.

4 Appendix

4.1 ggplot2 Links

<https://r-charts.com/ggplot2>

<https://r-graph-gallery.com/>

<https://r-graphics.org/>

<http://vita.had.co.nz/papers/layered-grammar.pdf>

<https://www.stat20.org/2-summarizing-data/03-a-grammar-of-graphics/notes>

<https://r4ds.had.co.nz/data-visualisation.html>

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

4.2 DiagrammeR Links

<https://rich-iannone.github.io/DiagrammeR/>

<https://builtin.com/data-science/diagrammer>

<https://graphviz.org/documentation/>