Graphics with "ggplot2"

Tidyverse

Gaston Sanchez

STAT 33B, Fall 2025

About

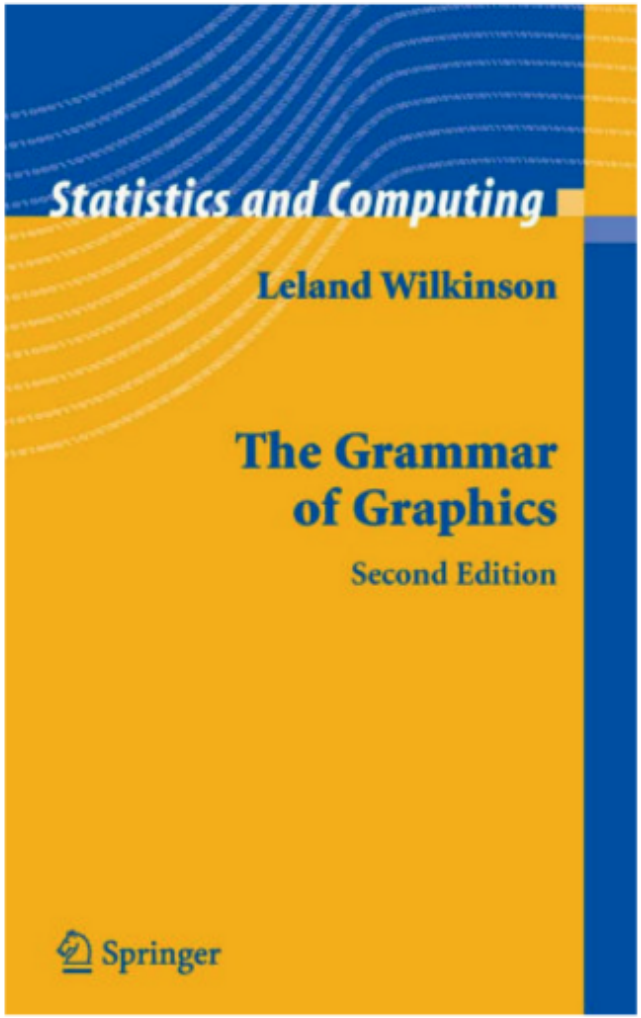
In this slides we provide a quick introduction to "ggplot2".

About "ggplot2"

- "ggplot2" (by Hadley Wickham) is an R package for producing statistical graphics.
- It is part of the ecosystem of data science packages known as "tidyverse".
- "ggplot2" is based on the **Grammar of Graphics**, a theoretical framework proposed by the late data visualization expert Leland Wilkinson.
- Underlying framework allows us to describe a wide range of graphics with a compact syntax and independent components.
- Compared to R's base graphics, "ggplot2" provides beautiful plots while taking care of fiddly details like legends, axes, colors, etc.

The Grammar of Graphcis

The Grammar of Graphics



Leland Wilkinson (1944-2021)

American Statistician

Scientific Visualization and Statistical Graphics expert

About the Grammar of Graphics (GG)

The Grammar of Graphics is Wilkinson's attempt to define a theoretical framework for graphics.

Grammar: Formal system of rules for generating graphics.

- Some rules are mathematic.
- Some rules are aesthetic (visual).
- Nearly every current software tool used to build plots has been informed by the GG.
- Its influence can be found in Tableau, Plotly, and the Python libraries bokeh, altair, seaborn, and plotnine.
- The most complete implementation of the grammar is found in the R package called `"ggplot2"` by Hadley Wickham.

Meaning of Aesthetic in GG

- A fundamental term—that is somewhat confusing for beginners—within the *GG* is that of **aesthetic**
- Aesthetic $\neq$ Beauty or Pretty.
- Meaning of *aesthetic* in the GG sense:
  - pertaining to sense perception
  - From greek *Aisthesthai* = perceive
- Aesthetics (GG): perception of visual properties that affect the way observations are displayed.

Visualization is simply
mapping data to geometry and color

Visualization is simply
mapping data to
geometry and color

Scatterplot Example

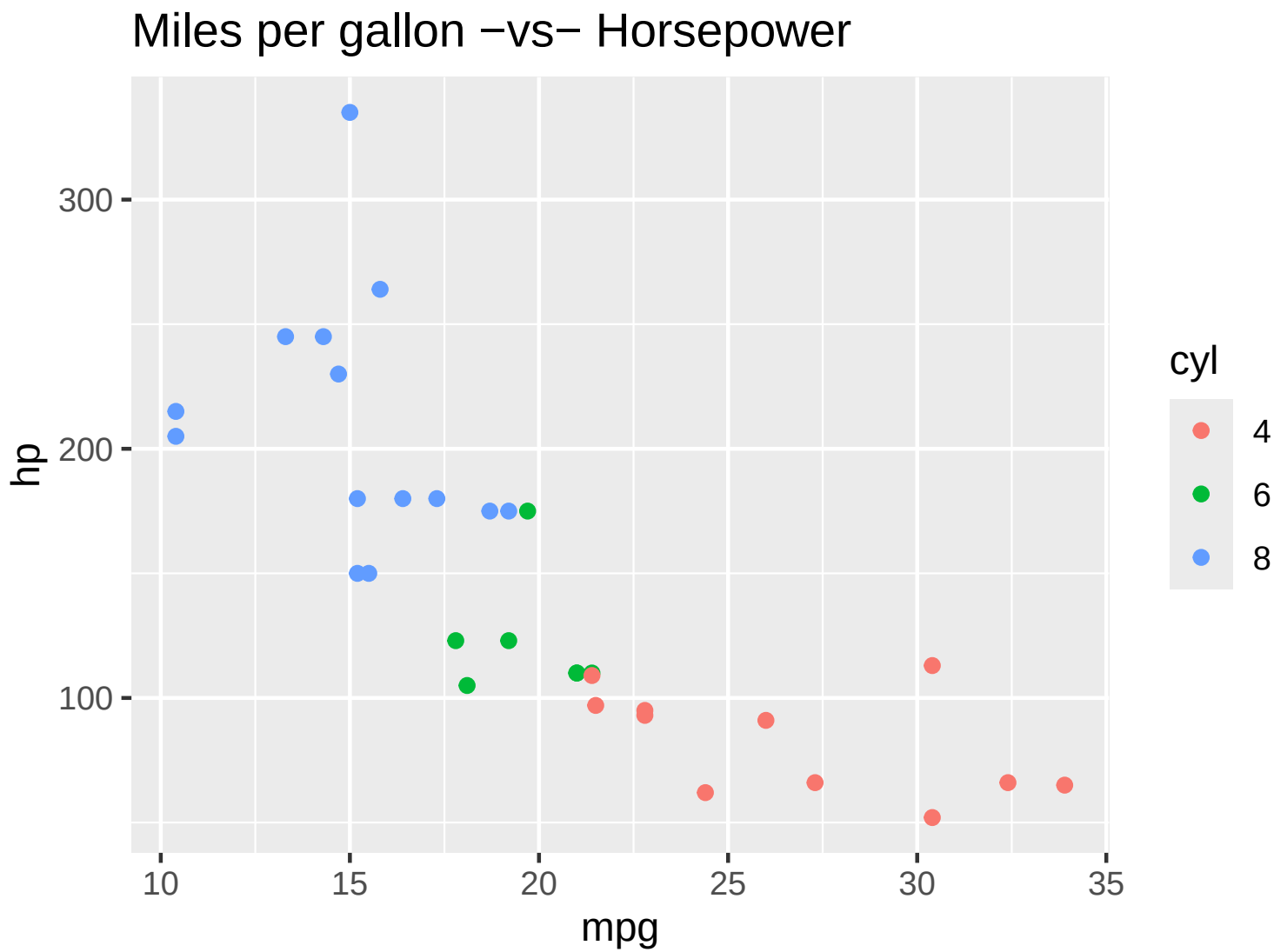
Data set `mtcars` (a few rows & cols shown)

```
mtcars[1:10, c('mpg', 'cyl', 'disp', 'hp')]
```

```
                   mpg cyl  disp  hp
Mazda RX4         21.0   6 160.0 110
Mazda RX4 Wag     21.0   6 160.0 110
Datsun 710        22.8   4 108.0  93
Hornet 4 Drive    21.4   6 258.0 110
Hornet Sportabout 18.7   8 360.0 175
Valiant           18.1   6 225.0 105
Duster 360        14.3   8 360.0 245
Merc 240D         24.4   4 146.7  62
Merc 230          22.8   4 140.8  95
Merc 280          19.2   6 167.6 123
```

Consider the graphic displayed in the next slide

Scatterplot Example



Important Terminology

The starting point is the **Data** that we want to visualize. The convention is to have data in a table object (e.g. `data.frame`, `tibble`) in which variables are stored as columns.

Then we have so-called **Geoms**, short for *geometric objects*; these are basically things such as bars, lines, points, polygons, and other kind of marks that are drawn to represent the data.

Geoms have **visual properties**, formally known as *aesthetic attributes*, and colloquially referred to as **aesthetics**; these are things such as x and y positions, line color, fill color, point shapes, etc.

Important Terminology

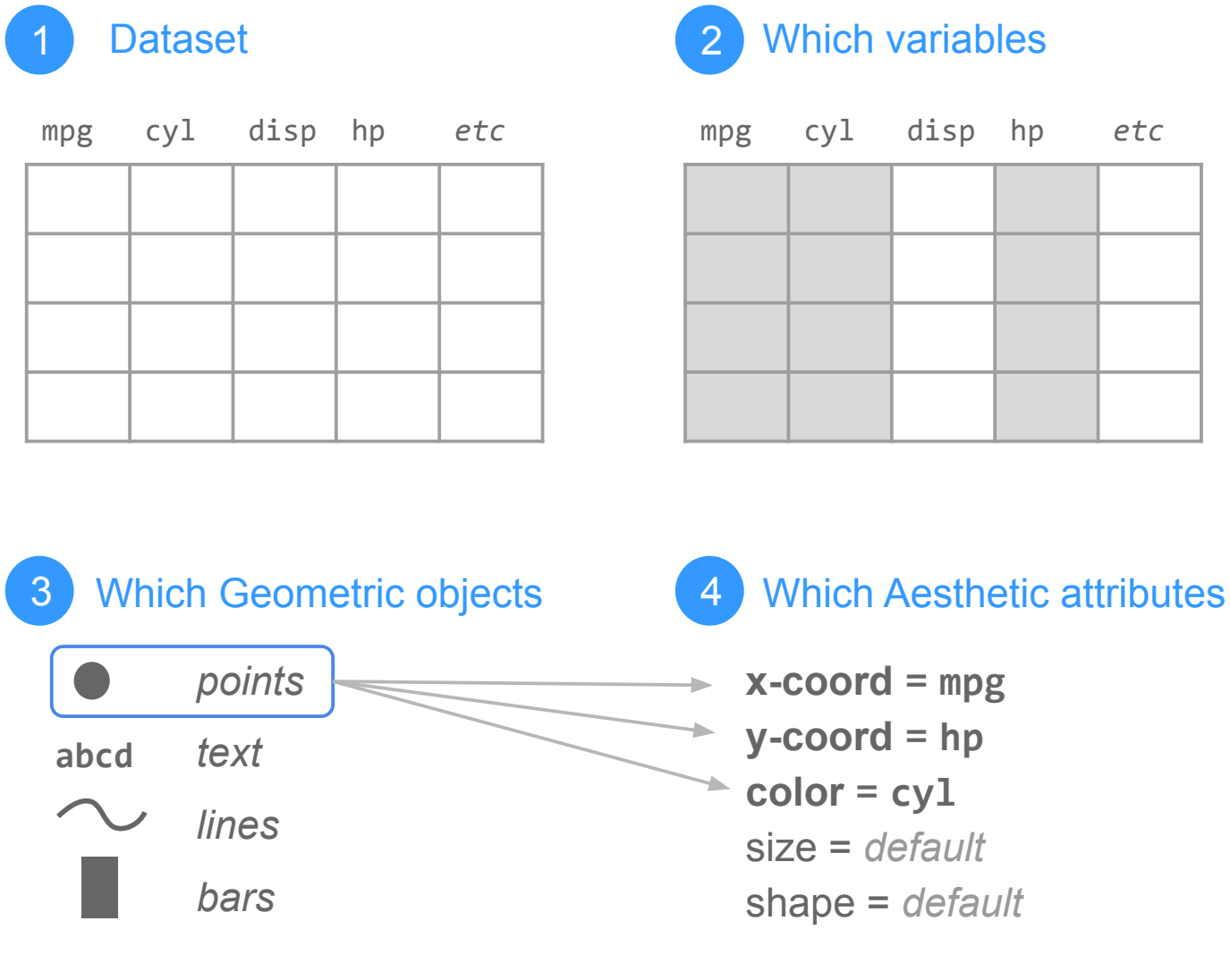
The use of a variable (from the data) to encode a visual property of a geom is called a **mapping.**

The use of a constant (or a value outside the data) to encode a visual property of a geom is called a **setting.**

Scales are used to handle the mapping from the values in the data space to values in the aesthetic space.

Guides are those auxiliary elements that allow the viewer to decode the mapping of the visual properties back to the data space. Perhaps the most typical guides are the tick marks, the labels on an axis, and legends (when applicable).

Scatterplot Example



Scatterplot (from the Grammar of Graphics)

Mapping data to geometric objects and their attributes

- Dataset: `mtcars`
- Variables: `mpg`, `hp`, `cyl`
- Geometry or *geom*: points
- Aesthetic mappings (perceptive attributes):
  - X-axis: `mpg`
  - Y-axis: `hp`
  - Color: `cyl`

R package "ggplot2"

About "ggplot2"

- "ggplot2" is the name of the package
- The gg in "ggplot2" stands for *Grammar of Graphics*
- Inspired in the **Grammar of Graphics** by Lee Wilkinson
- "ggplot" is the class of objects (plots)
- ggplot() is the main function in "ggplot2"

Toy Example

```r
# data.frame
sw_dat = data.frame(
  name = c('Leia', 'Luke', 'Han'),
  sex = c('female', 'male', 'male'),
  force = c(TRUE, TRUE, FALSE),
  height = c(150, 172, 180),
  weight = c(49, 77, 80)
)

# tibble
sw_tbl = tibble(
  name = c('Leia', 'Luke', 'Han'),
  sex = c('female', 'male', 'male'),
  force = c(TRUE, TRUE, FALSE),
  height = c(150, 172, 180),
  weight = c(49, 77, 80)
)
```

Toy Example

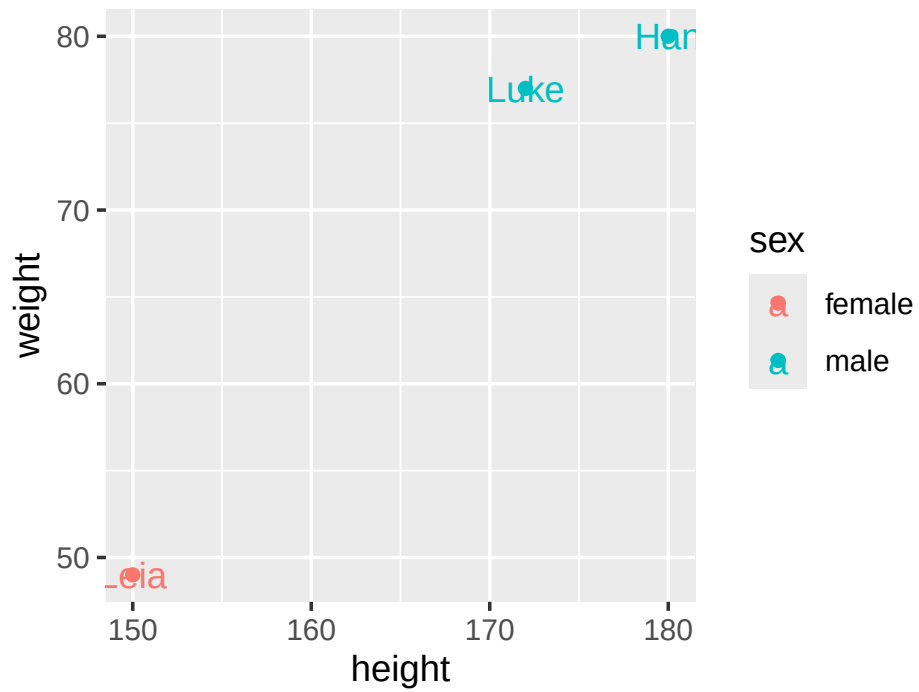
```
sw_dat
```

```
  name    sex force height weight
1 Leia female  TRUE    150     49
2 Luke   male  TRUE    172     77
3  Han   male FALSE    180     80
```

```
sw_tbl
```

```
# A tibble: 3 x 5
  name  sex    force height weight
  <chr> <chr>  <lgl>  <dbl>  <dbl>
1 Leia  female TRUE     150     49
2 Luke  male   TRUE     172     77
3 Han   male   FALSE    180     80
```

```r
# scatter plot example
ggplot(data = sw_dat,
       mapping = aes(x = height,
                     y = weight,
                     color = sex,
                     label = name)) +
  geom_point() +
  geom_text()
```



How does `ggplot2` work?

How does "ggplot2" work?

At its core, in "ggplot"'s adaptation of the grammar of graphics, a plot can be decomposed into three primary elements:

- the **data**,
- the **geometry** used to encode the observations on the plot, and
- the **aesthetic mapping** of the variables in the data to visual attributes of the geometries.

How does "ggplot2" work?

- Plots are created layer-by-layer (i.e. piece-by-piece).
- Plot components added with `+` (plus) operator.
- Always start with `ggplot()`
- Add at least one **geometry** layer, via `geom_...()` functions.
- Specify which variables in the data are **mapped** to visual properties; this is done through the `aes()` function at the `ggplot()` level and/or at the `geom_()` level.
- Optionally, you can also specify statistical transformations, computation of scales for visual properties, specific coordinate systems, and many other secondary graphical elements.

```r
# scatter plot example
ggplot(data = sw_dat,
       mapping = aes(x = height,
                     y = weight,
                     color = sex,
                     label = name)) +
  geom_point() +
  geom_text()
```

- `data = sw_dat` specifies the table containing the variables.
- `mapping = aes(...)` specifies the mapping of variables in the data to visual properties.
- `geom_point()` is the geometry layer to graph points (i.e. scatter plot).
- `geom_text()` is the geometry layer to graph text.

The data must be in a `data.frame` or `tibble`

For better or for worse, there are multiple ways to obtain the same graphic with slightly different commands.

Various way to get same scatter plot

```
# initial option: mappings at the "global" ggplot() level
ggplot(data = sw_dat,
       mapping = aes(x = height,
                     y = weight,
                     color = sex,
                     label = name)) +
  geom_point() +
  geom_text()
```

```
# another option (abbreviated)
ggplot(sw_dat,
       aes(x = height,
           y = weight,
           color = sex,
           label = name)) +
  geom_point() +
  geom_text()
```

Various way to get same scatter plot

```
# another option: mappings at various layers
ggplot(data = sw_dat,
       mapping = aes(x = height,
                     y = weight,
                     color = sex)) +
  geom_point() +
  geom_text(mapping = aes(label = name))
```

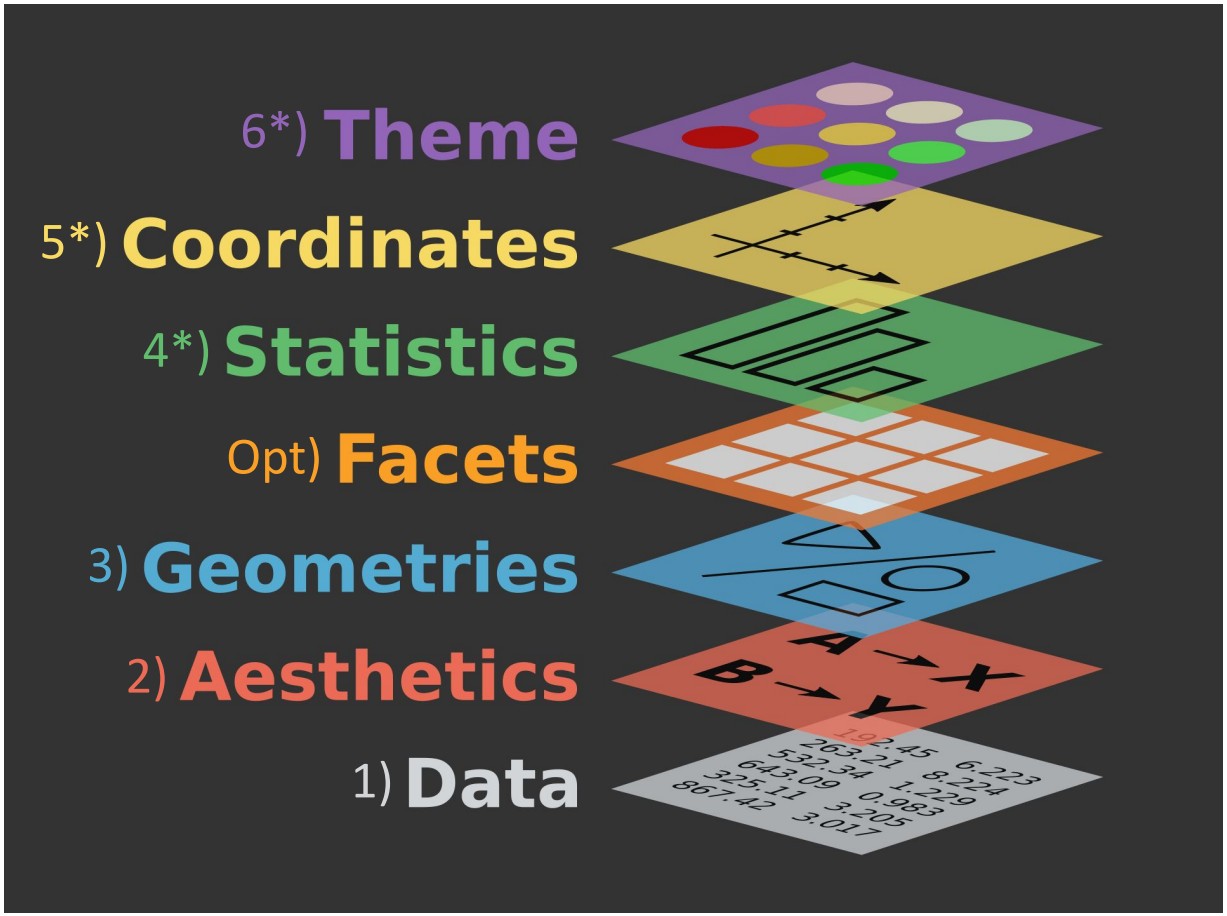
```
# another option: mappings in all layers
ggplot(data = sw_dat,
       mapping = aes(x = height, y = weight)) +
  geom_point(mapping = aes(color = sex)) +
  geom_text(mapping = aex(label = name))
```

Various way to get same scatter plot

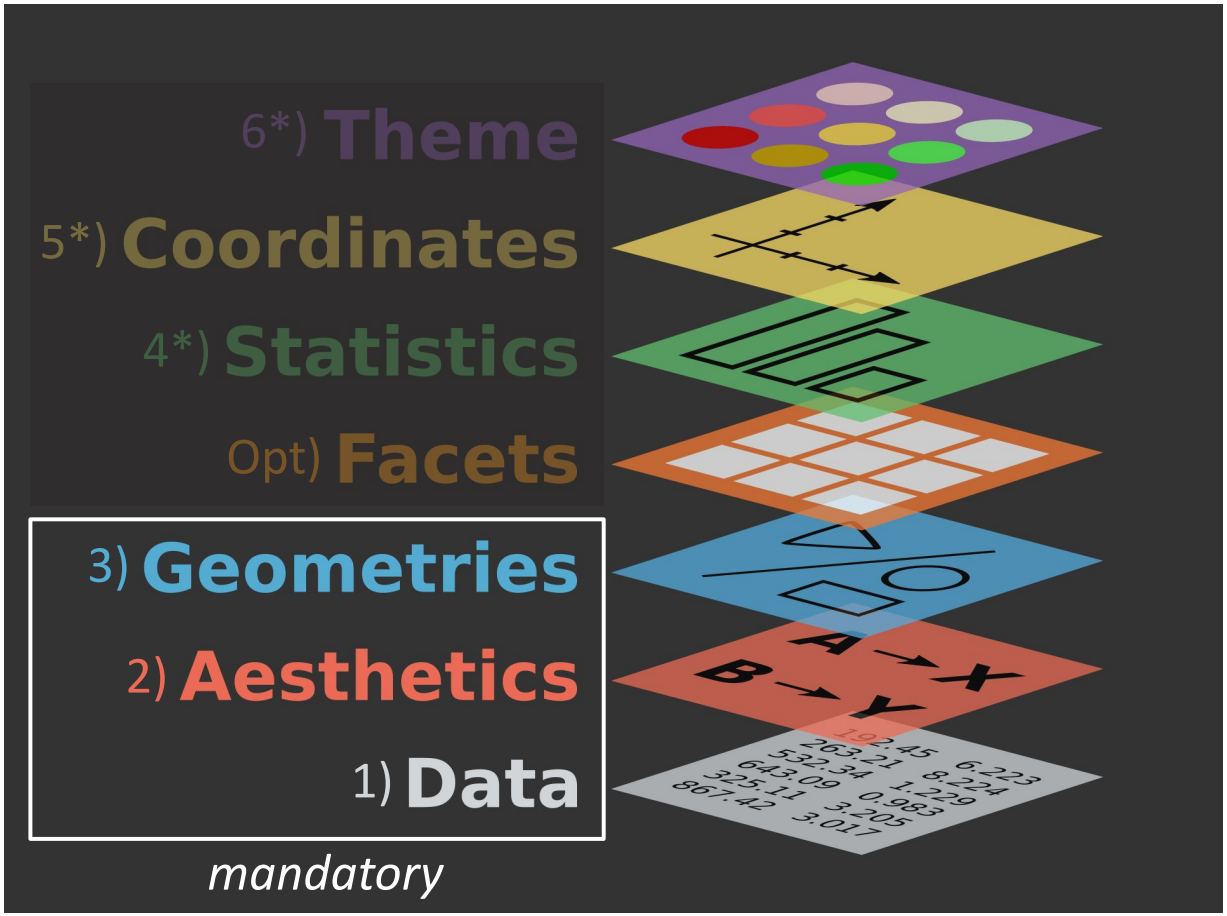
```
# another option: mappings at the "local" geom_() level
ggplot(data = sw_dat) +
  geom_point(aes(x = height, y = weight, color = sex)) +
  geom_text(aes(x = height, y = weight, label = name))
```

```
# another option: empty call to ggplot()
# data and mappings at geom levels (very repetitive)
ggplot() +
  geom_point(data = sw_dat,
             aes(x = height, y = weight, color = sex)) +
  geom_text(data = sw_dat,
            aes(x = height, y = weight, label = name))
```

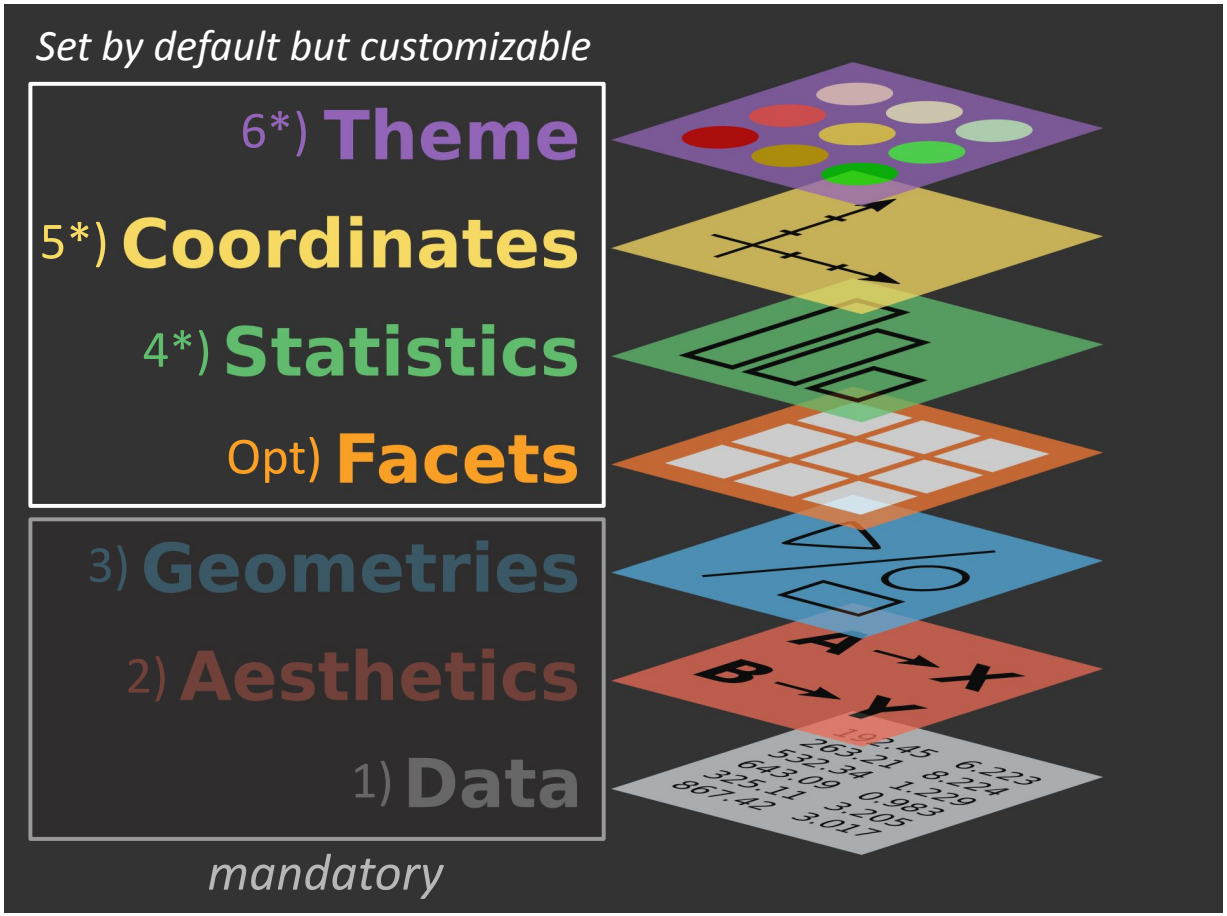
Grammar of Graphics in ggplot



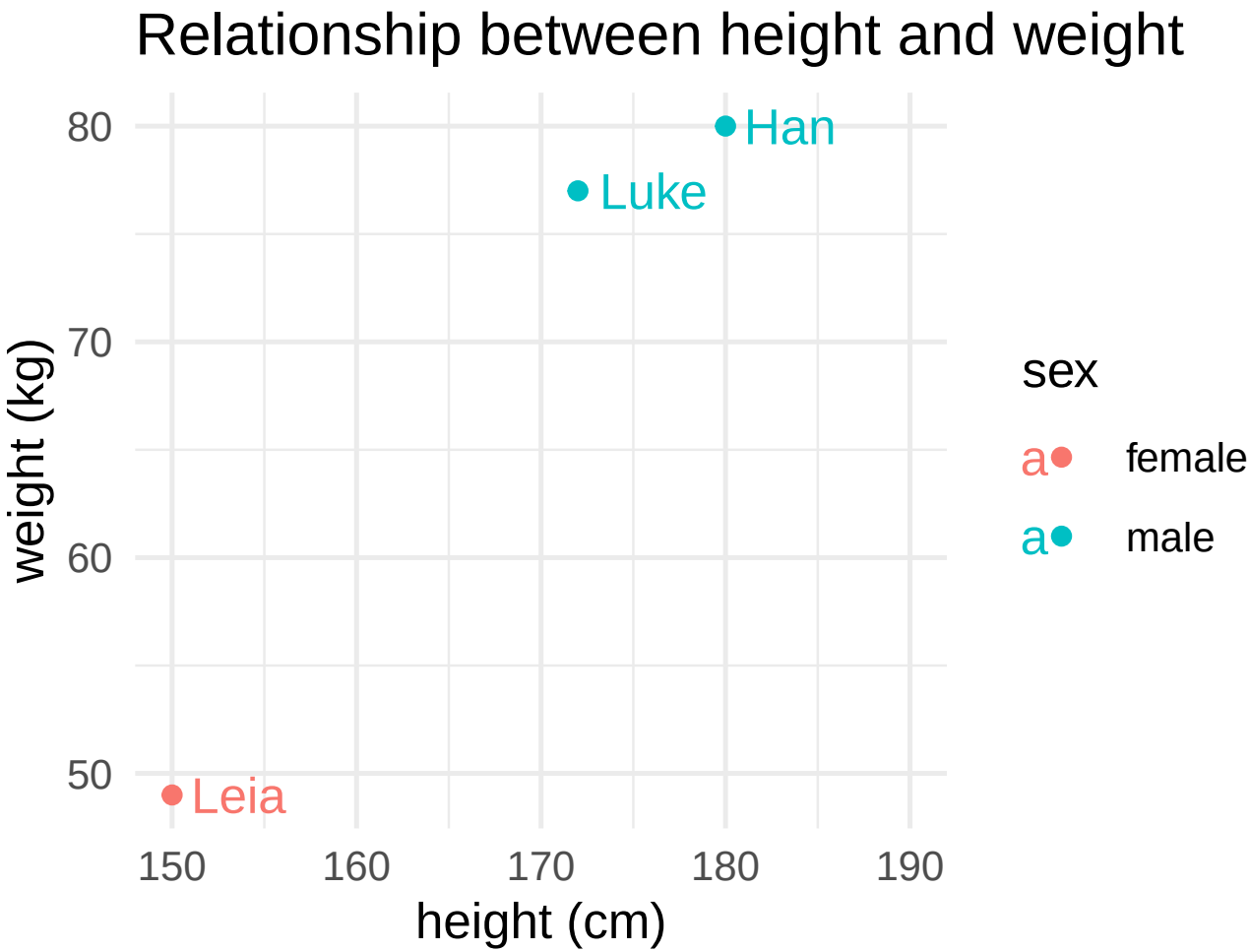
Grammar of Graphics in ggplot



Grammar of Graphics in ggplot



Extended Example 1



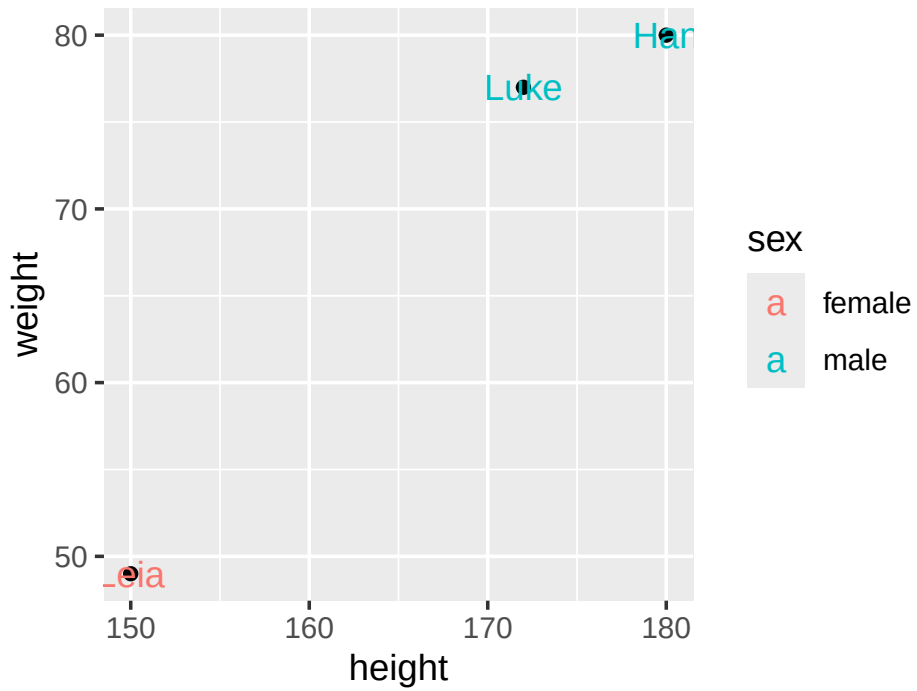
Extended Example 1

```
ggplot(data = sw_dat,
       mapping = aes(x = height,
                     y = weight,
                     color = sex,
                     label = name)) +
  geom_point() +
  geom_text(hjust = -0.2) +
  scale_x_continuous(limits = c(150, 190)) +
  labs(title = "Relationship between height and weight",
       x = "height (cm)",
       y = "weight (kg)") +
  theme_minimal()
```

Extended Example 2

Where we put the aesthetic mappings matters:

```
ggplot(data = sw_dat,
       aes(x = height, y = weight, label = name)) +
  geom_point() +
  geom_text(aes(color = sex))
```



Always ask:

- What is the data set of interest?
- What variables (columns) will be used to make the plot?
- What graphic shapes (geoms) will be used to display the data?
- What features of the shapes will be used to represent the data values?

Considerations

- How many variables?
  - One variable
  - Two variables
  - Three or more
- What type of variables?
  - Quantitative (e.g. continuous, integer)
  - Qualitative (e.g. factor, ordered factor, character)
  - Date (time)

"`ggplot2`" functions may behave differently depending on the data type of the mapped variables.

Customizing Plots

Customizing Plots in "ggplot2"

Layer	Description
data	A data frame to visualize
aesthetics	The map or "wires" between data and geometry
geometry	Geometry to represent the data visually
labels	Titles and axis labels
scales	How numbers in data are converted to numbers on screen
guides	Legend settings
annotations	Additional geoms that are not mapped to data
facets	Side-by-side panels
coordinates	Coordinate systems (Cartesian, logarithmic, polar)
statistics	An alternative to geometry

Customizing plots in "ggplot2"

How else can we make our plot look more like the Best in Show plot?

Add more geometries to add additional details to a plot:

```
ggplot(data = sw_dat,
       aes(x = height, y = weight, label = name)) +
  geom_point(aes(color = force)) +
  geom_text(size = 2, hjust = 1, vjust = 1, nudge_x = -0.05)
```

Customizing plots in "ggplot2"

We can also **set** parameters outside of the aesthetics.

Doing so sets a constant value instead of mapping to a feature in the data. For instance, here's how to set size to 5 for all points:

```
ggplot(data = sw_dat,
       aes(x = height, y = weight, label = name)) +
  geom_point(aes(color = force), size = 5) +
  geom_text(size = 2, hjust = 1, vjust = 1, nudge_x = -0.05)
```

Customizing plots in "ggplot2"

Note that if you want to **set** a constant color for all points, you need to do so outside of `aes()`:

```
# this works
ggplot(data = sw_dat,
       aes(x = height, y = weight)) +
  geom_point(color = "blue")
```

```
# this does NOT work (as you would expect)
ggplot(data = sw_dat,
       aes(x = height, y = weight)) +
  geom_point(aes(color = "blue"))
```

You can also use the scales layer to customize the color choices.

Saving Plots

Saving Plots

Recall the plot we made of height and weight:

```
ggplot(data = sw_dat) +
  geom_point(aes(x = height, y = weight))
```

In ggplot2, use `ggsave()` to save the most recent plot you created:

```
ggsave("scatterplot.png")
```

The file format is selected automatically based on the extension.

Common formats are PNG, JPEG, and PDF.

Saving Plots

If you are going to be saving several plots, its better to create objects for each of them

```
scatter1 = ggplot(data = sw_dat) +
  geom_point(aes(x = height, y = weight))

scatter2 = ggplot(data = sw_dat) +
  geom_point(aes(x = height, y = weight, color = sex))
```

```
ggsave(filename = "scatterplot1.png", plot = scatter1)
ggsave(filename = "scatterplot2.png", plot = scatter2)
```

Savings plots

You can also save a plot with one of R's "plot device" functions.

The steps are:

1. Call a plot device function: `png()`, `jpeg()`, `pdf()`, `bmp()`, `tiff()`, or `svg()`.
2. Run your code to make the plot.
3. Call `dev.off()` to indicate that you're done plotting.

Note: This will only work in the console!

Resources

Website: https://ggplot2.tidyverse.org/

Book: ggplot2: Elegant Graphics for Data Analysis by Hadley Wickham https://ggplot2-book.org/

Book: R Graphics Cookbook by Winston Chang https://r-graphics.org/

RStudio ggplot2 cheat sheet: https://github.com/rstudio/cheatsheets/raw/master/data-visualization-2.1.pdf