UCLA QCBio Collaboratory

Introduction to R and data visualization

Wenbin Guo
Bioinformatics IDP, UCLA
wbguo@ucla.edu

2024 Spring

UCLA Bioinformatics

Notation of the slides

- `Code` or `Pseudo-Code` chunk starts with " ➢ ", e.g.
  - ➢ `print("Hello world!")`

- Link is underlined

- Important terminology is in **bold** font

- Practice comes with

Agenda

- Day 1: R basics
  - Environment setup
  - Variable, Operators
  - Data structure: Vector, Matrix, List, Data frame
- Day 2: R advanced topics
  - Flow control, Loops
  - Function, Packages, File Input/Output
  - Data wrangling with tidyverse toolkit
- Day 3: Data visualization with ggplot2
  - ggplot2 syntax, grammar, and elements
  - Basic plot types and customization

Image from internet

UCLA QCBio Collaboratory

Day 3: R data visualization

Wenbin Guo
Bioinformatics IDP, UCLA
wbguo@ucla.edu
2024 Spring

Overview

Time

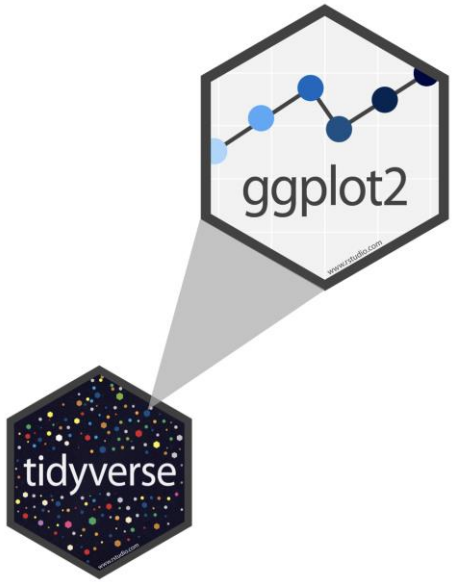
- 3-hour workshop (45min + 45min + 30min + practice/Q&A)

Topics

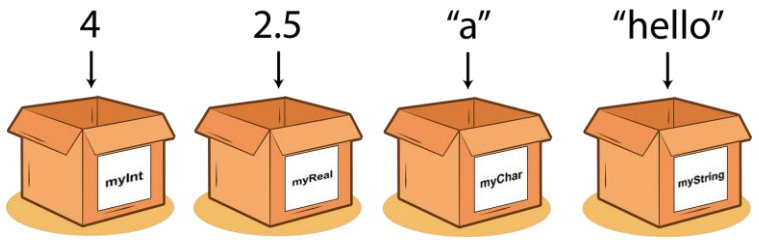
❑Getting started with ggplot2 visualization

❑Plot types and customization

❑Examples and practices



Summary – Day1&2

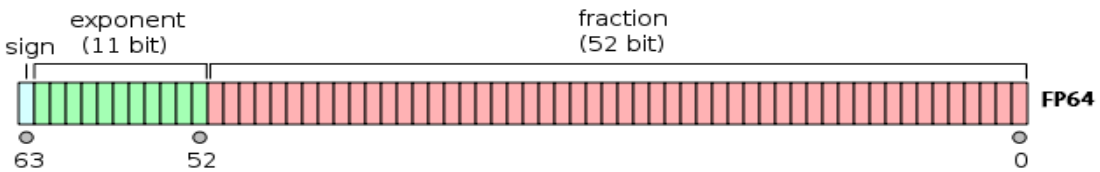


Variables: the container of data

- ❑ Naming rules
- ❑ Value assignment
- ❑ Variable classes
- ❑ Inspecting the variable
- ❑ Inspecting the workspace

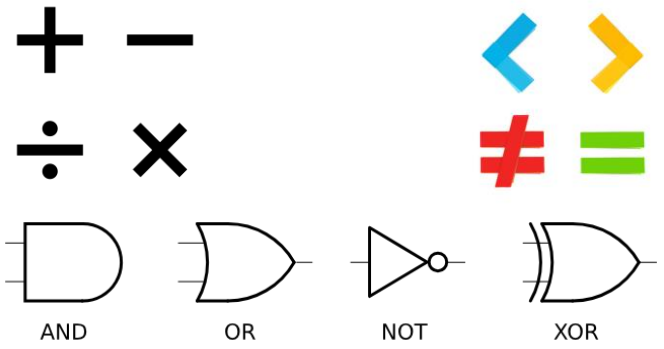
Numbers

- ❑ Number representation
- ❑ Special numbers



$$\pm d_0 . d_1 d_2 \cdots d_p \times b^e$$

Summary – Day1&2



Operators: the actions on variables

- ❑ Arithmetic
- ❑ Relational
- ❑ Logical
- ❑ Operator's precedence

Operator	Description
+	addition
-	subtraction
*	multiplication
/	division
^ or **	exponentiation
x %% y	modulus (x mod y) 5%%2 is 1
x %/% y	integer division 5%/%2 is 2

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
!x	Not x
x | y	x OR y
x & y	x AND y
isTRUE(x)	test if X is TRUE

Precedence	Operator	Description
18	:: :::	access variables in a namespace
17	$ @	component / slot extraction
16	[ ] [[ ]]	indexing
15	^	Exponentiation operator (Right to Left)
14	+a -a	Unary plus, Unary minus
13	:	Sequence operator
12	%% %*% %/% %in% %o% %x%	Special operators
11	* /	Multiplication, Division
10	+ -	Addition, Subtraction
9	< <= > >=	Less than, Less than or equal, Greater than, and Greater than or equal
	== !=	Equality and Inequality
8	!	Logical NOT
7	& &&	Logical AND
6	| ||	Logical OR
5	~	as in formulae
4	-> ->>	Right assignment operator, Global right assignment operator
3	<- <<-	Left assignment operator, Global left assignment operator (Right to Left)
2	=	Left assignment operator (Right to Left)
1	?	help (unary and binary)

Top to bottom in **descending precedence**

Summary – Day1&2

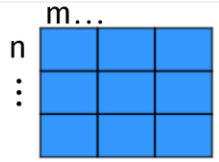
R objects: data container

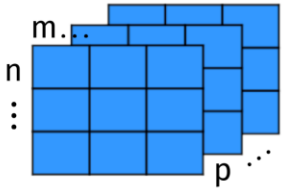
- ❑ Vectors
- ❑ String and Factor
- ❑ Matrix
- ❑ List
- ❑ Data frame

Operations on the R objects

- ❑ Create
- ❑ Indexing
- ❑ Update
- ❑ …

	Dimensions	Mode (data "type")	Example
Vector		Identical	`c(10,0.2,34,48,53)`

Matrix		Identical	`matrix(c(1,2,3,`
			`11,12,13),`
			`nrow = 2,`
			`ncol = 3)`

Data frame		Can be different	`data.frame(x = 1:3,` `y = 5:7)`
Array		Identical	`array(data = 1:3,` `dim = c(2,4,2))`
List	Vector Matrix Data frame Array	Can be different	`list(x = cars[,1],` `y = cars[,2])`

Summary – Day1&2

R programming basics

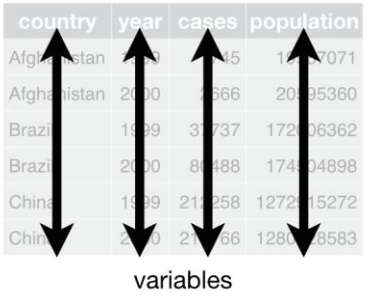
- ❑ Flows
  - ❑ if-else
  - ❑ switch
- ❑ Loops
  - ❑ for, while, repeat
  - ❑ apply family
- ❑ Functions
  - ❑ Built-in
  - ❑ Self-defined
- ❑ Packages
  - ❑ Install/update/remove
  - ❑ Load/unload

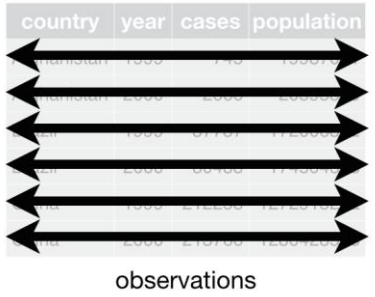
Building blocks for a complicated program

Image from internet

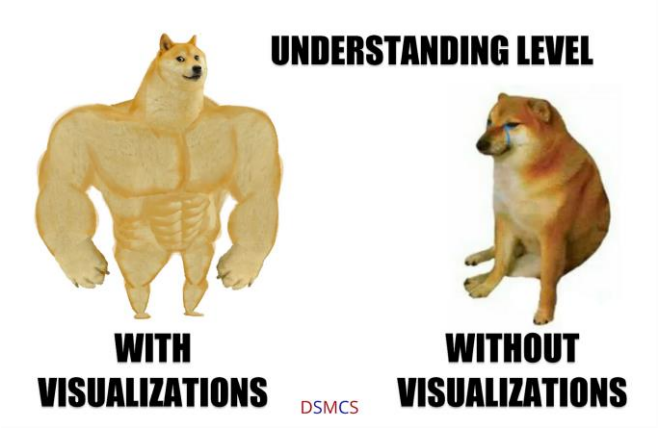
Summary – Day1&2

Data wrangling functions



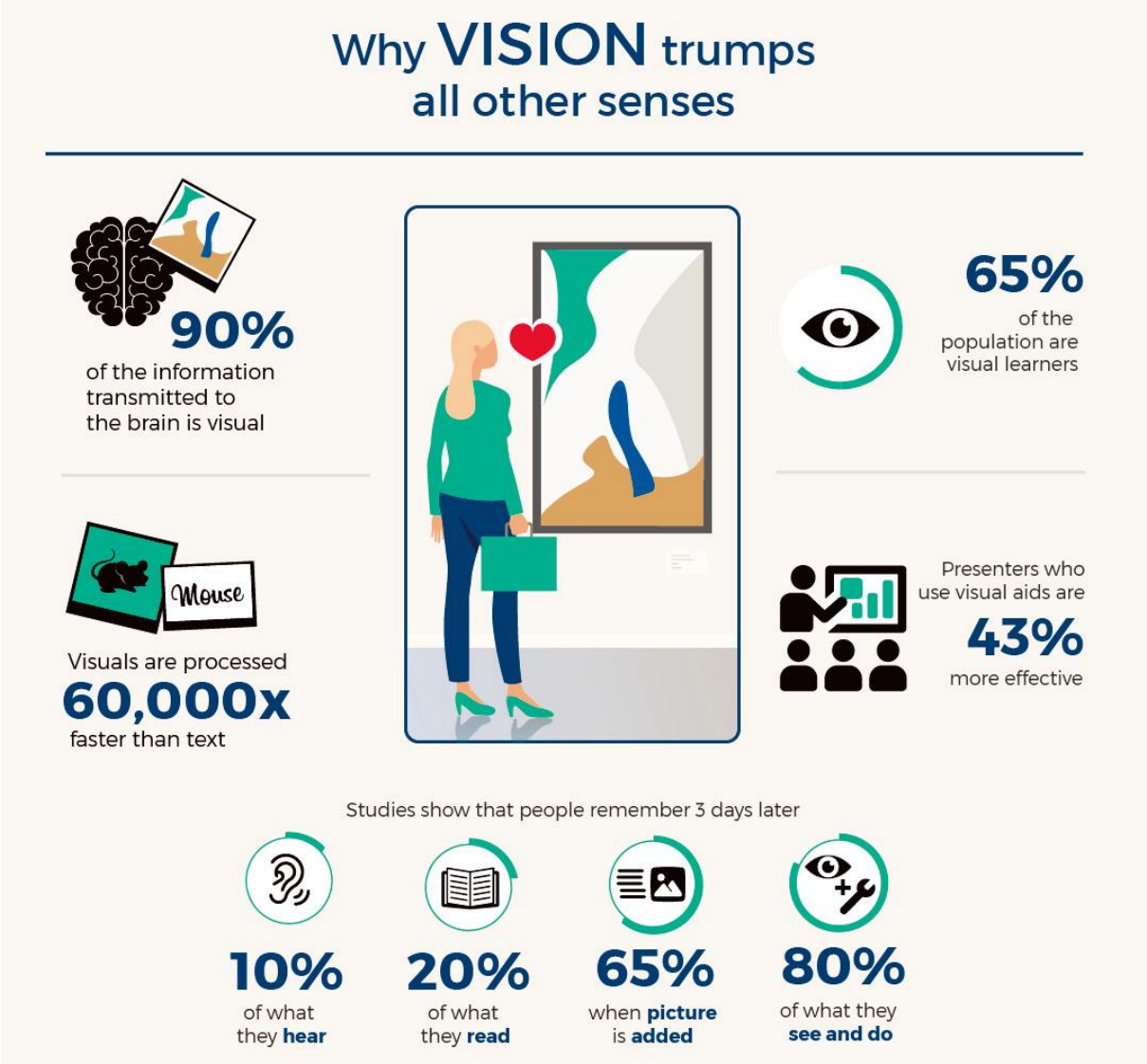


Category	Function	Usage
Manipulate observations	`filter()`	keep rows that meet criterions
	`arrange()`	orders observations according to variables
	`bind_rows()`	bind any number of data frames by row
Manipulate variables	`select()`	keep variables using their names or types
	`mutate()`	create new variables
	`*_join()`	merge data frames by columns
Reshape data	`pivot_longer()`	convert data frame from wide to long format
	`pivot_wider()`	convert data frame from long to wide format
Summarize data	`group_by()`	group data frame by variable
	`summarize()`	summarize the grouped data frame
pipe	`%>%`	chain operations together



Data Visualization

Facts on how our brain reacts to visuals



Martin, K. *Medical Writing* (2020)

"A picture may be worth a thousand words"

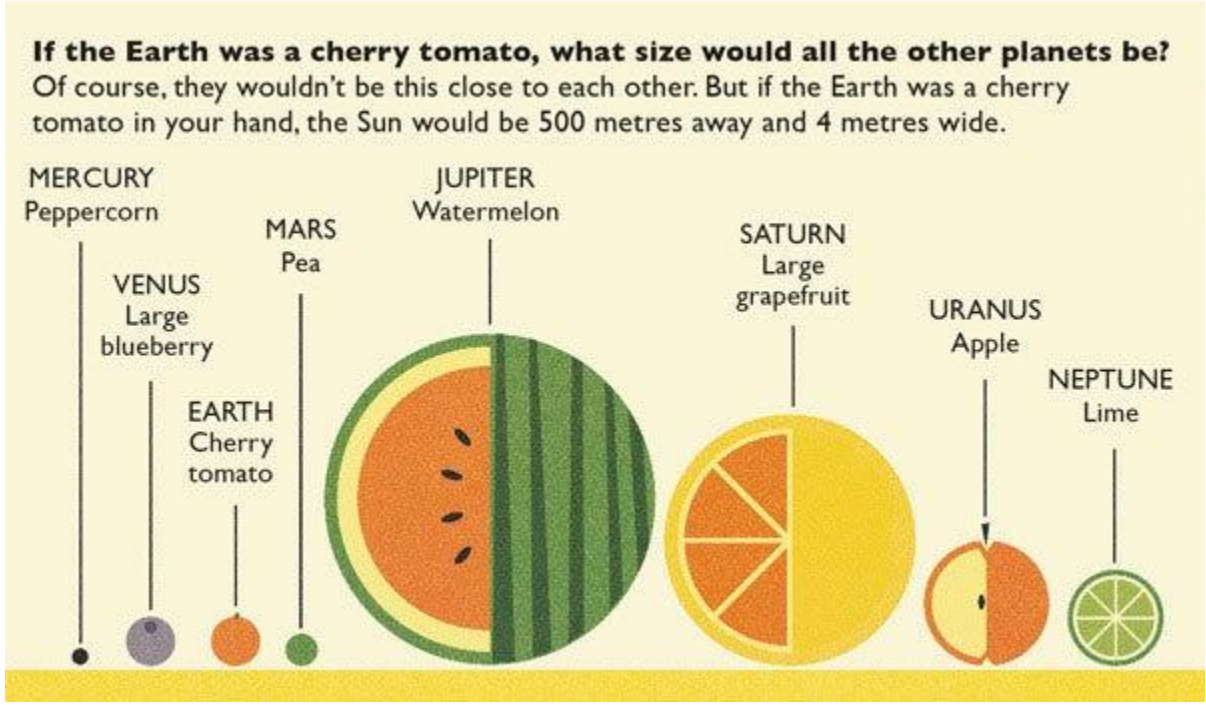
Example

You
list all the planet's diameters in the solar system

ChatGPT
Here are the diameters of the planets in our solar system:

1. **Mercury**: Approximately 4,880 kilometers (3,032 miles).
2. **Venus**: Approximately 12,104 kilometers (7,521 miles).
3. **Earth**: Approximately 12,742 kilometers (7,918 miles).
4. **Mars**: Approximately 6,779 kilometers (4,212 miles).
5. **Jupiter**: Approximately 139,820 kilometers (86,881 miles).
6. **Saturn**: Approximately 116,460 kilometers (72,366 miles).
7. **Uranus**: Approximately 50,724 kilometers (31,518 miles).
8. **Neptune**: Approximately 49,244 kilometers (30,598 miles).

These values represent the equatorial diameters of the planets, which are the most commonly used measurements for planetary diameters.

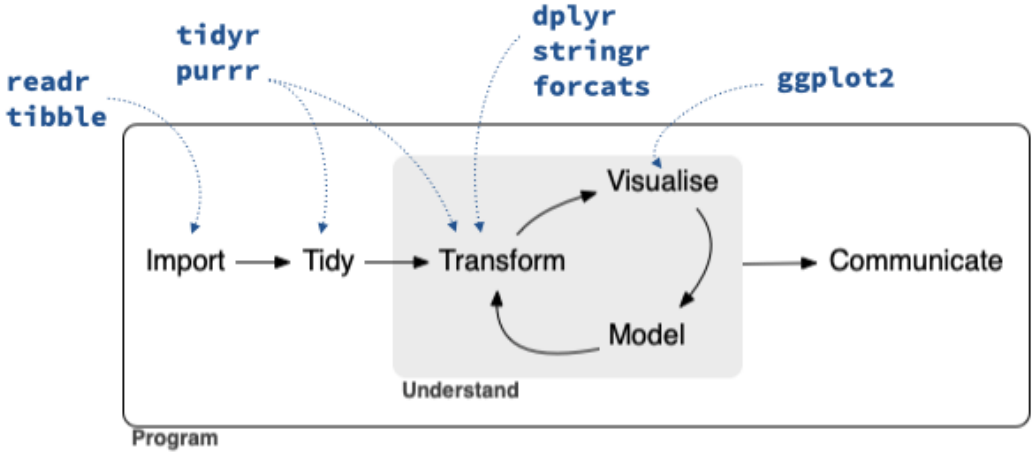


But the visualization can be imprecise (e.g. the distance between planets)

Image from Internet

The role and road of data visualization in science

As a scientist, we use data visualization techniques to explore and explain

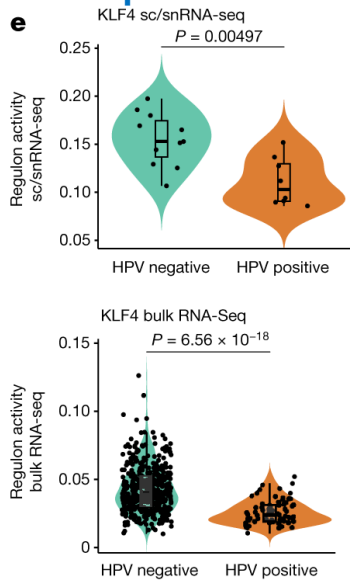


Visualization is a core step in a data analysis project

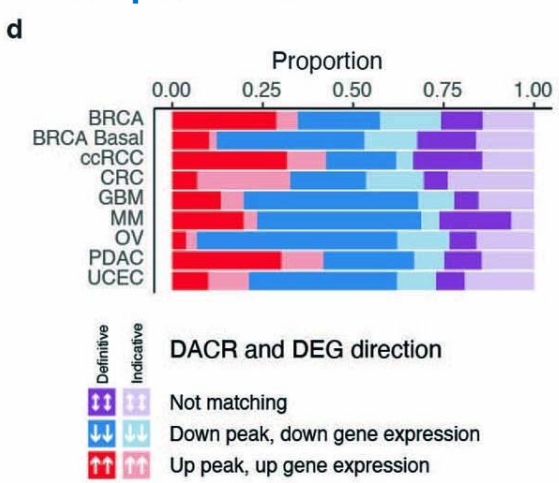
Image from Internet

data visualization in science (example)

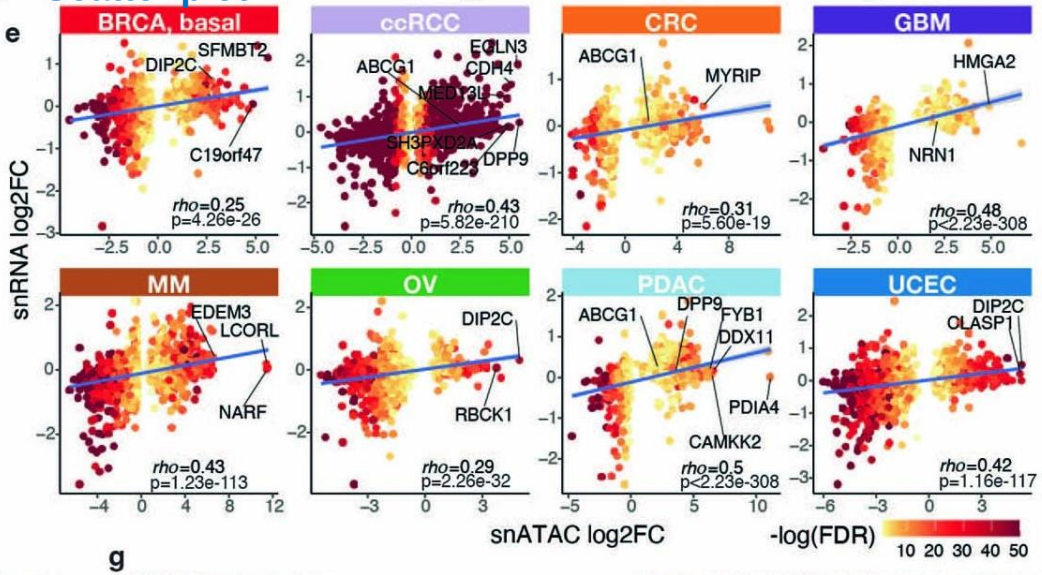
Boxplot/Violin plot



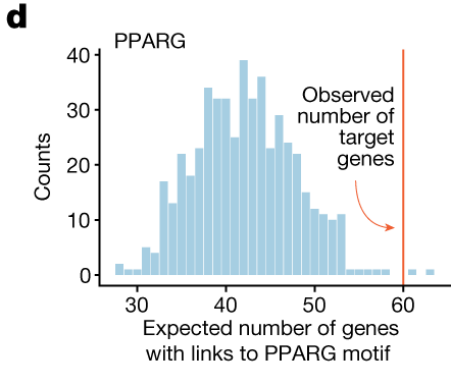
Bar plot



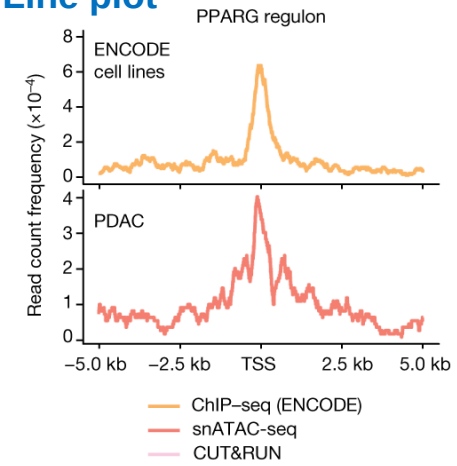
Scatter plot



Histogram



Line plot



Heatmap

And more …

Terekhanova, et al. *Nature* (2023)

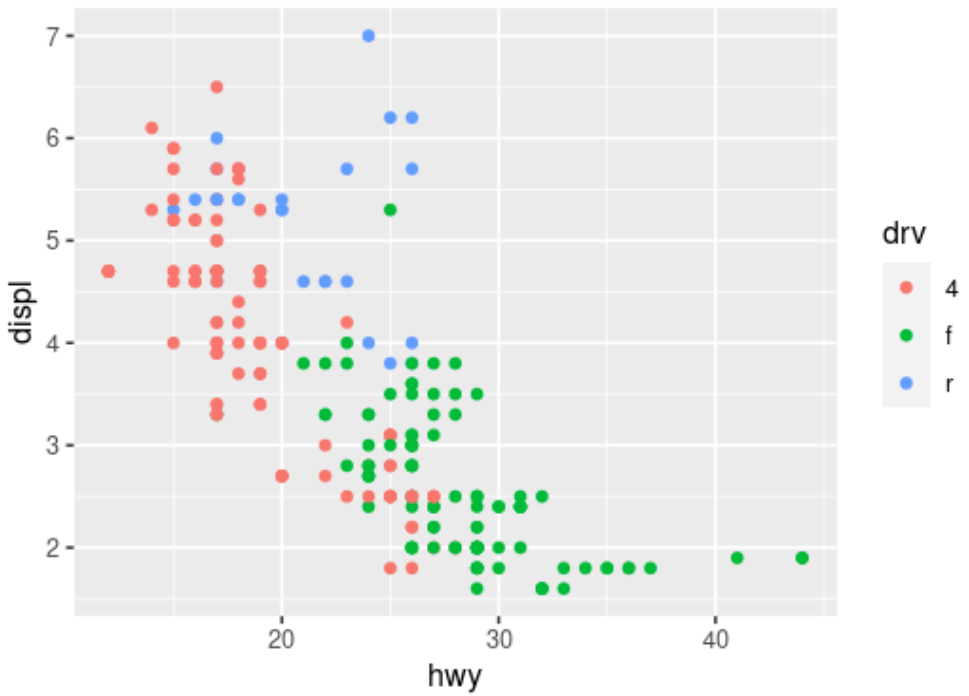
Idea behind ggplot2 visualization

Take scatter plot as an example

Filter

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
1	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
2	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
3	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
4	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
5	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
6	audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact
7	audi	a4	3.1	2008	6	auto(av)	f	18	27	p	compact
8	audi	a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p	compact
9	audi	a4 quattro	1.8	1999	4	auto(l5)	4	16	25	p	compact
10	audi	a4 quattro	2.0	2008	4	manual(m6)	4	20	28	p	compact
11	audi	a4 quattro	2.0	2008	4	auto(s6)	4	19	27	p	compact
12	audi	a4 quattro	2.8	1999	6	auto(l5)	4	15	25	p	compact
13	audi	a4 quattro	2.8	1999	6	manual(m5)	4	17	25	p	compact
14	audi	a4 quattro	3.1	2008	6	auto(s6)	4	17	25	p	compact
15	audi	a4 quattro	3.1	2008	6	manual(m6)	4	15	25	p	compact

Showing 1 to 16 of 234 entries, 11 total columns



Data source: mpg

Idea behind ggplot2 visualization

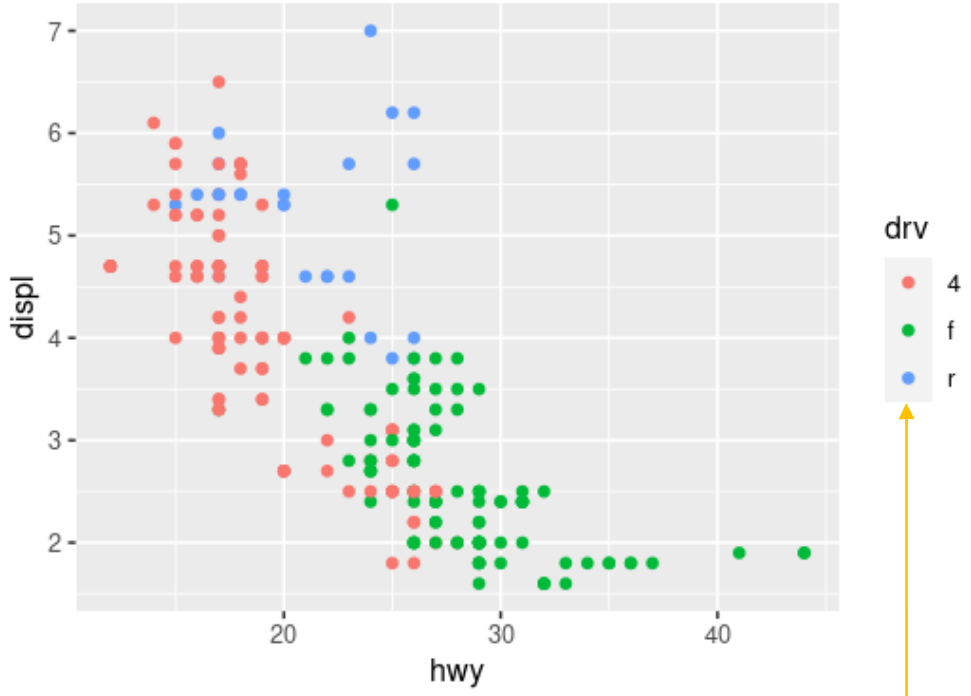
Take scatter plot as an example

hwy → x
displ → y
drv → colour

Filter

	manufacturer	model	displ	year	cyl	trans	drv	cty	hwy	fl	class
1	audi	a4	1.8	1999	4	auto(l5)	f	18	29	p	compact
2	audi	a4	1.8	1999	4	manual(m5)	f	21	29	p	compact
3	audi	a4	2.0	2008	4	manual(m6)	f	20	31	p	compact
4	audi	a4	2.0	2008	4	auto(av)	f	21	30	p	compact
5	audi	a4	2.8	1999	6	auto(l5)	f	16	26	p	compact
6	audi	a4	2.8	1999	6	manual(m5)	f	18	26	p	compact
7	audi	a4	3.1	2008	6	auto(av)	f	18	27	p	compact
8	audi	a4 quattro	1.8	1999	4	manual(m5)	4	18	26	p	compact
9	audi	a4 quattro	1.8	1999	4	auto(l5)	4	16	25	p	compact
10	audi	a4 quattro	2.0	2008	4	manual(m6)	4	20	28	p	compact
11	audi	a4 quattro	2.0	2008	4	auto(s6)	4	19	27	p	compact
12	audi	a4 quattro	2.8	1999	6	auto(l5)	4	15	25	p	compact
13	audi	a4 quattro	2.8	1999	6	manual(m5)	4	17	25	p	compact
14	audi	a4 quattro	3.1	2008	6	auto(s6)	4	17	25	p	compact
15	audi	a4 quattro	3.1	2008	6	manual(m6)	4	15	25	p	compact

Showing 1 to 16 of 234 entries, 11 total columns



Data source: mpg

Create a ggplot object

Create the initial plot object

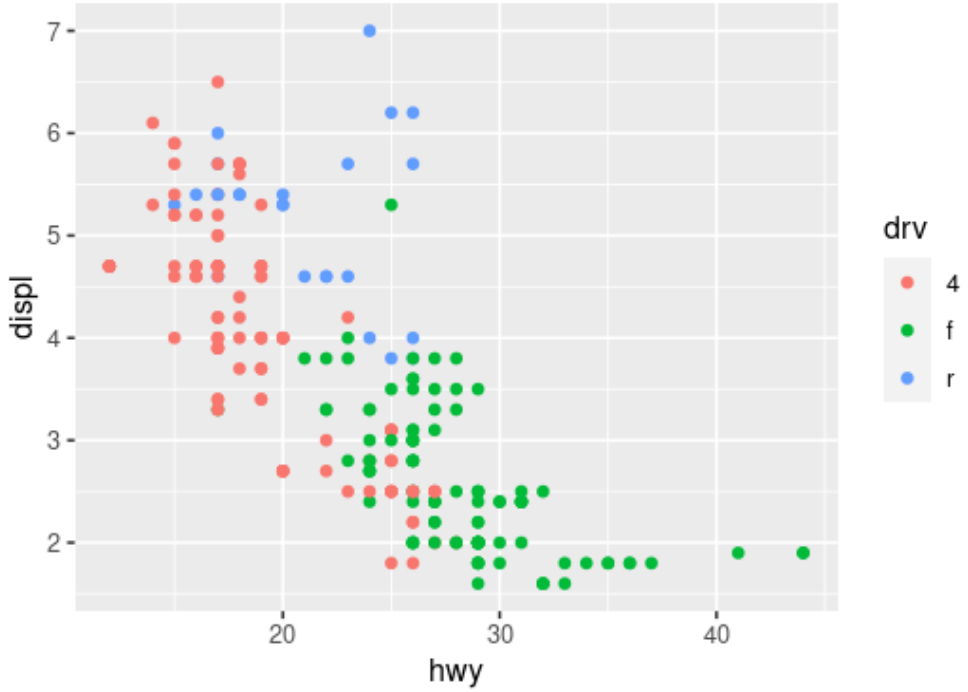
- `ggplot(data = NULL, mapping = aes(), ...)`
  - `data`: dataset to use for plot
  - `mapping`: list of aesthetic mappings to use for plot
  - `...` : other arguments

- `ggplot()` is usually followed by a plus sign (+) to add additional components/**layers** to the plot

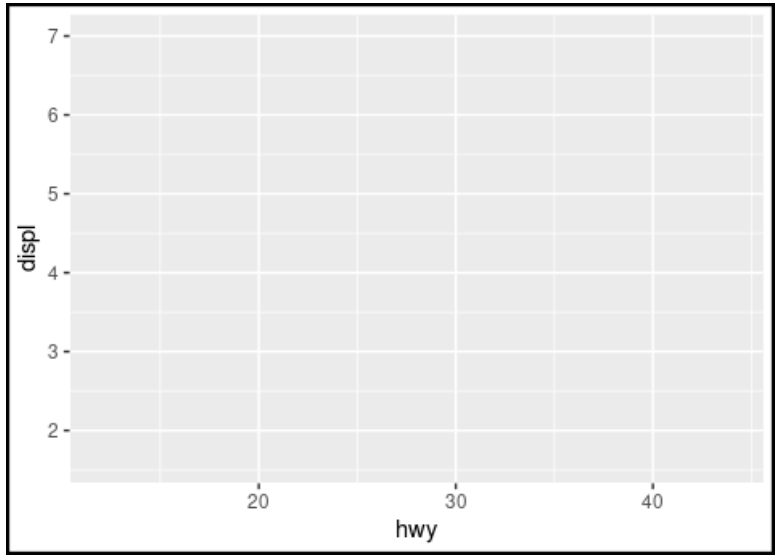
https://ggplot2.tidyverse.org/reference/ggplot.html

Key components of a ggplot object

- The **data**
- A set of **aesthetic mappings** between *variables* and *visual properties*
- At least one **layer** describing how to render the observations

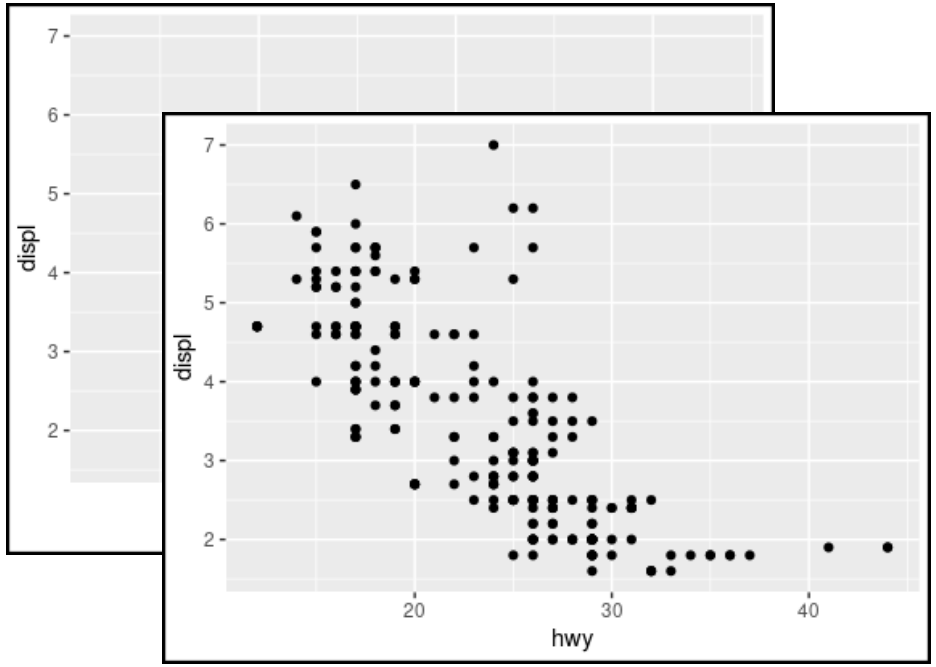


Building plots layer by layer (example)



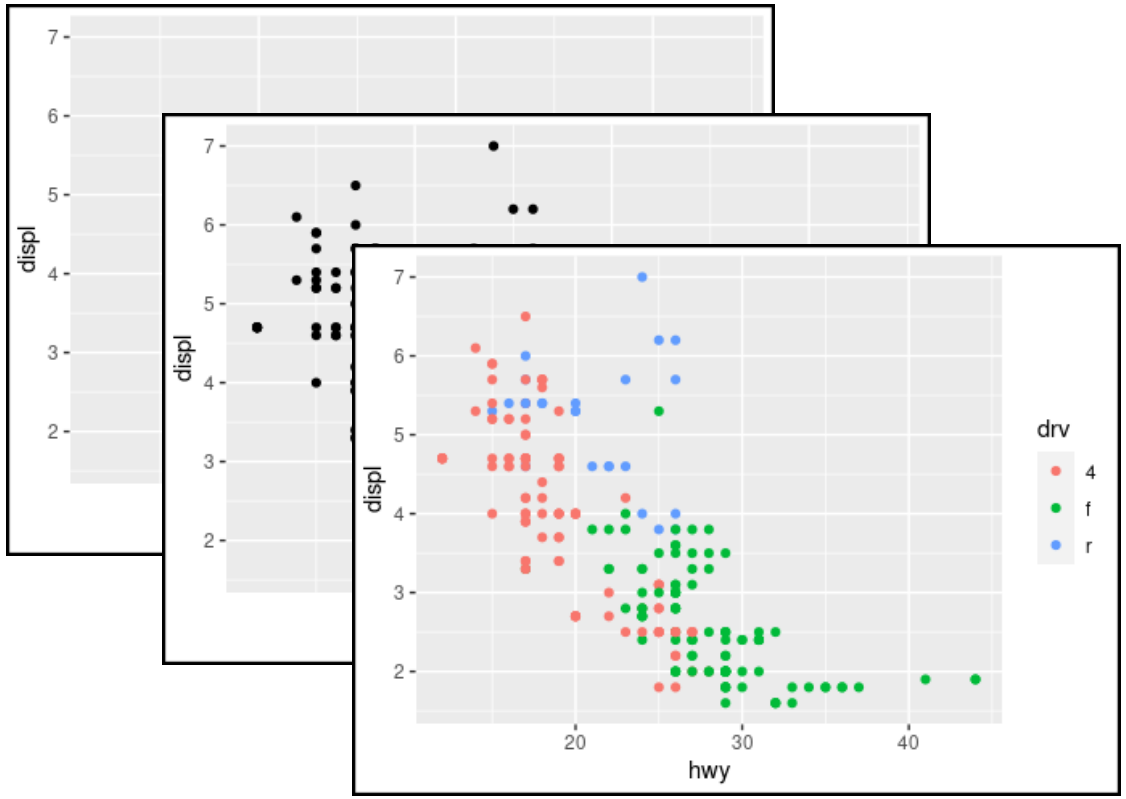

```
base = ggplot(data=mpg, aes(x=hwy, y=displ))
base
```

Building plots layer by layer (example)



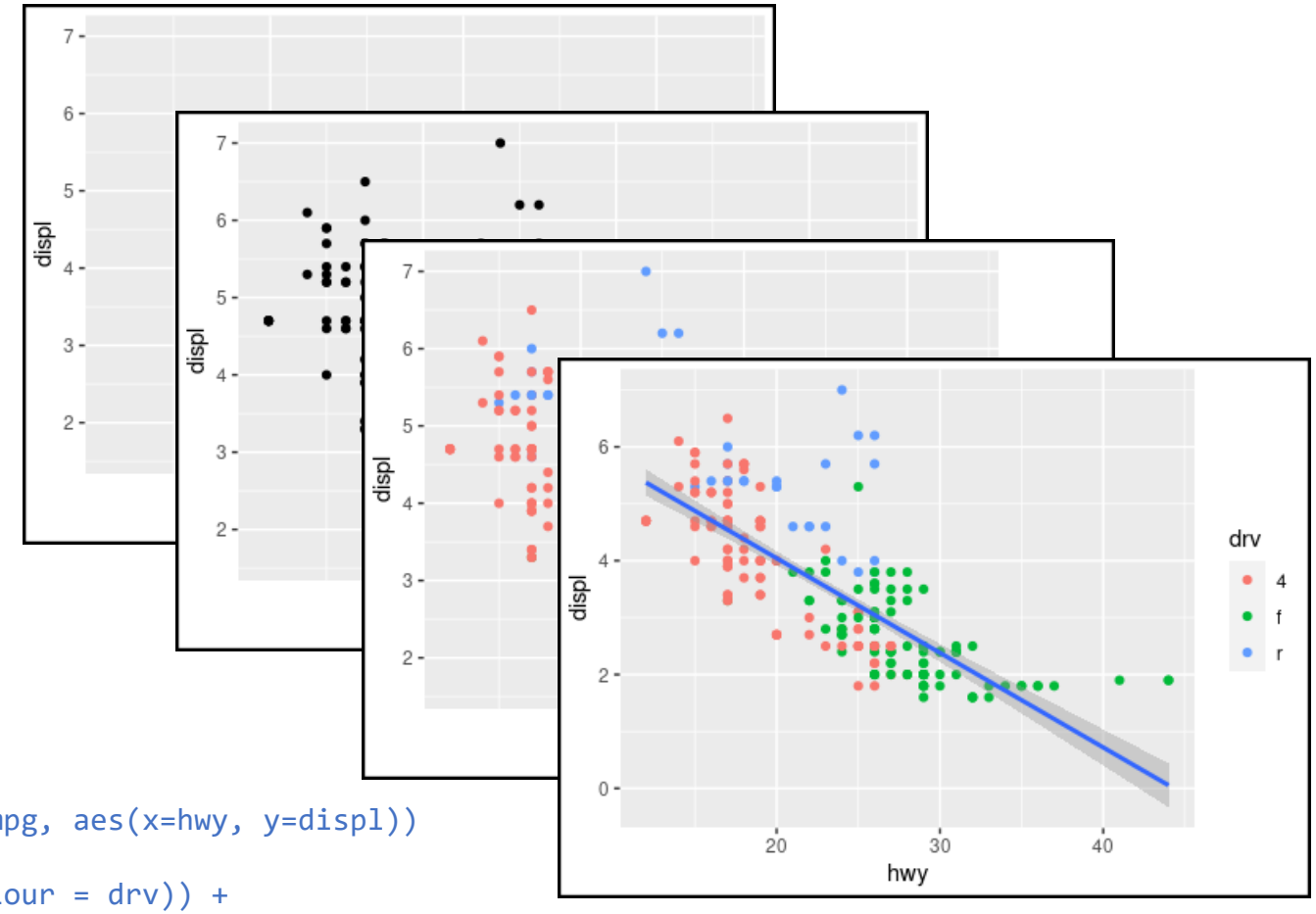

```
base = ggplot(data=mpg, aes(x=hwy, y=displ))
base +
  geom_point()
```

Building plots layer by layer (example)



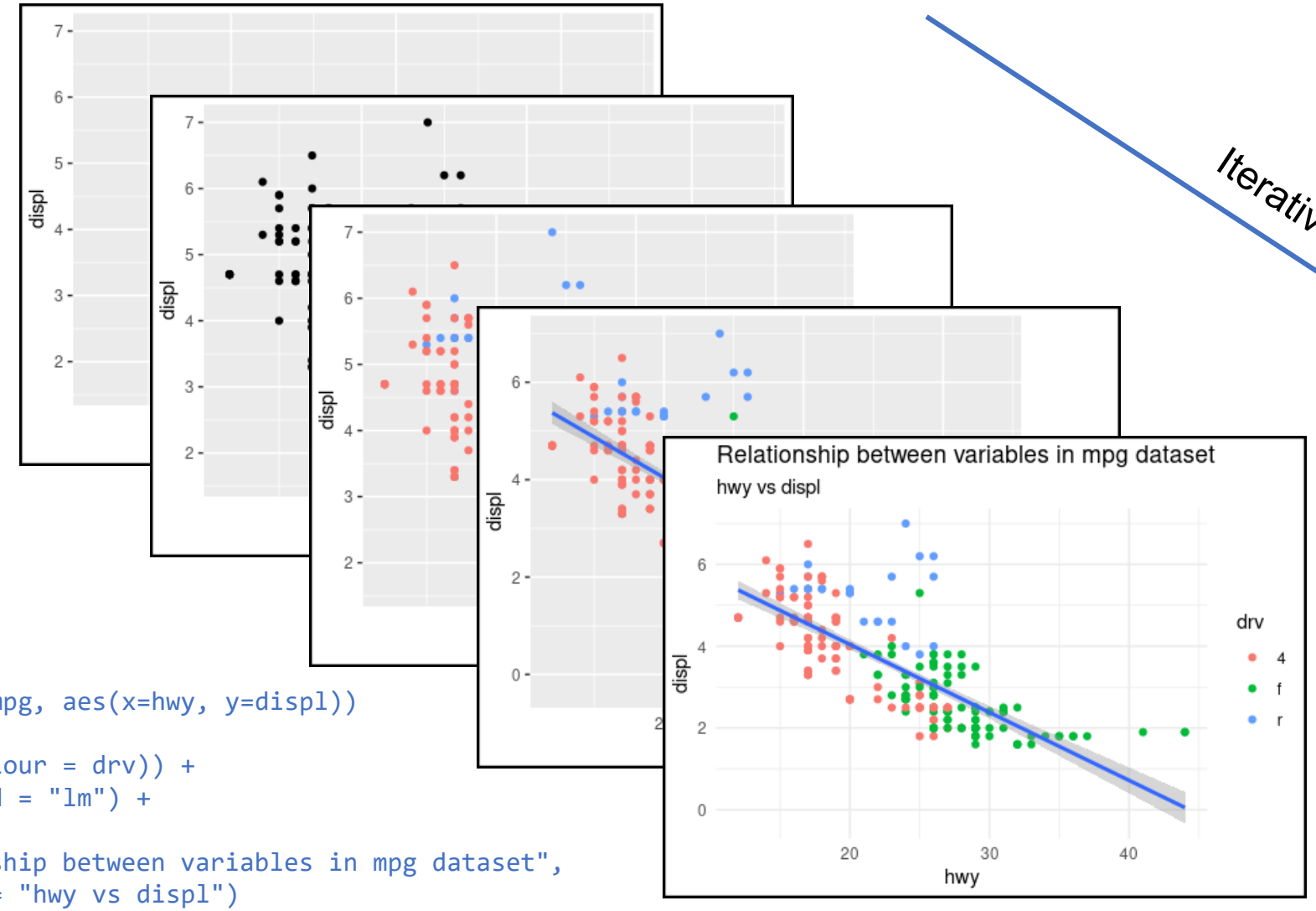

```
base = ggplot(data=mpg, aes(x=hwy, y=displ))
base +
  geom_point(aes(colour = drv))
```

Building plots layer by layer (example)




```
base = ggplot(data=mpg, aes(x=hwy, y=displ))
base +
  geom_point(aes(colour = drv)) +
  geom_smooth(method = "lm")
```

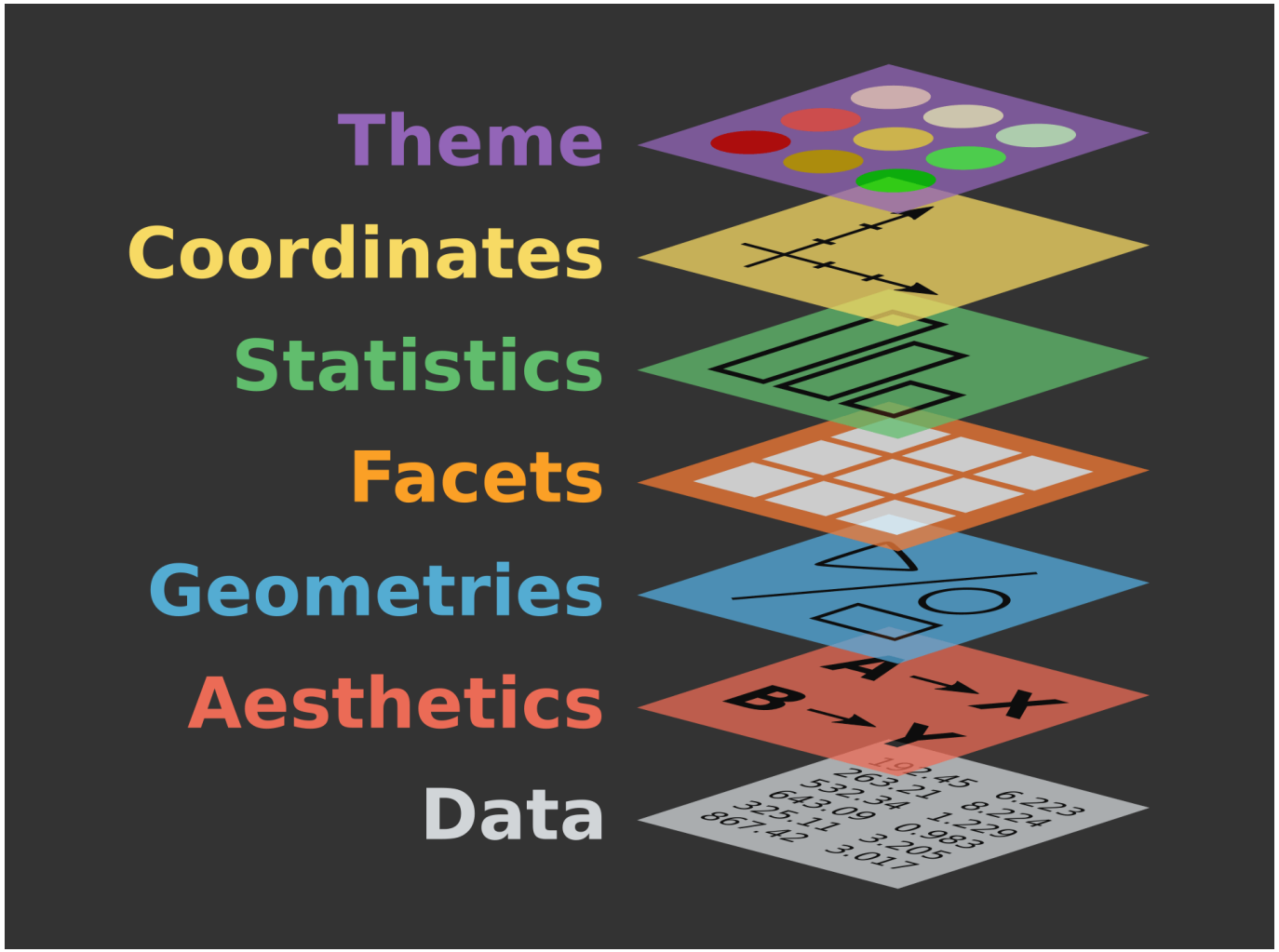
Building plots layer by layer (example)



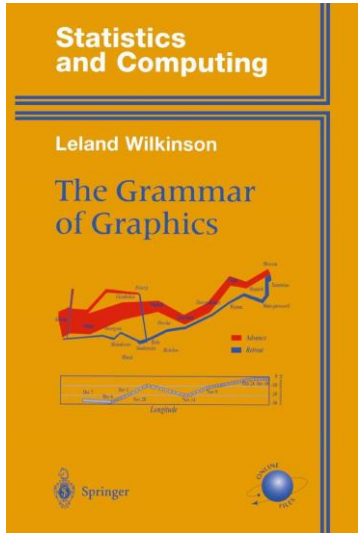
Iteratively

```
base = ggplot(data=mpg, aes(x=hwy, y=displ))
base +
  geom_point(aes(colour = drv)) +
  geom_smooth(method = "lm") +
  theme_minimal() +
  ggtitle("Relationship between variables in mpg dataset",
          subtitle = "hwy vs displ")
```

The layered grammar of graphics

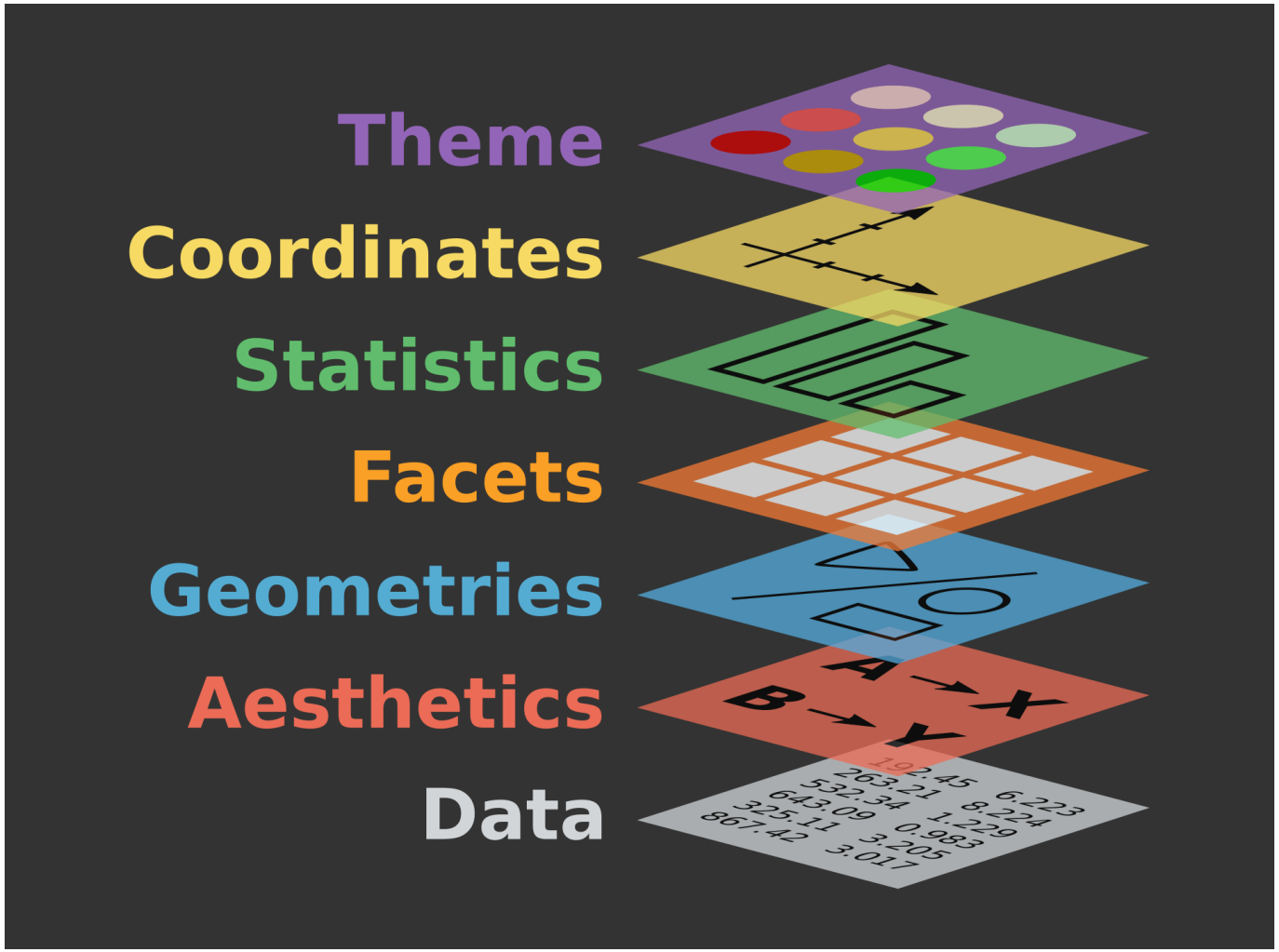


Leland Wilkinson
(1944-2021)



The Grammar of Graphics (1999)

The layered grammar of graphics



Describes all the non-data ink

Plotting space for the data

Statistical models and summaries

Rows and columns of sub-plots

Shapes used to represent the data

Scales onto which data is mapped

The actual variables to be plotted

Aesthetics

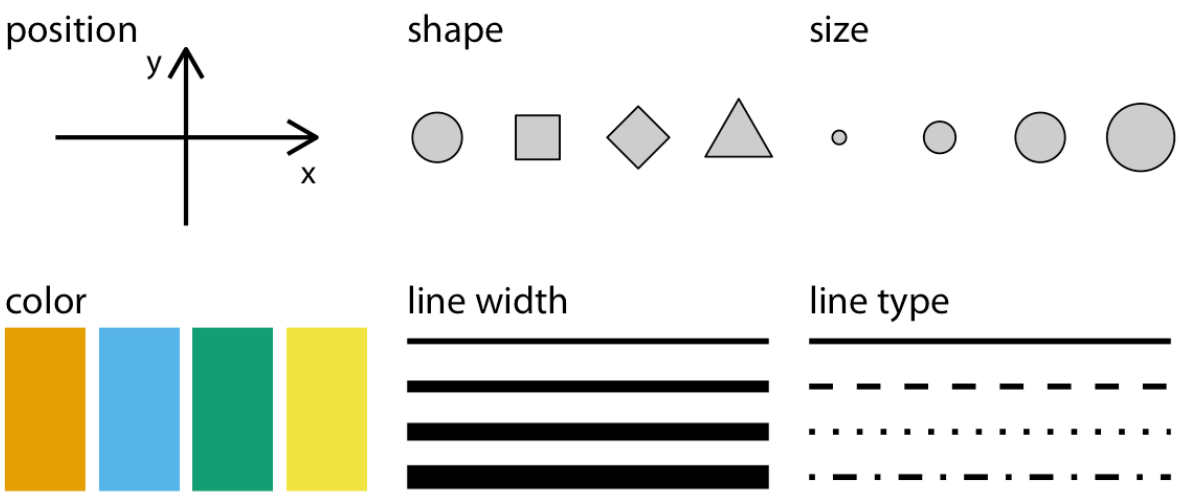


Image from book *Fundamentals of Data Visualization*

Aesthetic mappings

Aesthetic mappings `aes()` describe how variables are mapped to visual properties or aesthetics. It takes aesthetic-variable pairs

- `aes(x = displ, y = hwy, colour = class)`

Aesthetic	Description
x	x-axis position
y	y-axis position
colour	Color of points or outlines of other shapes
fill	Fill color
size	size of the point or thickness of line
alpha	Transparency of the shape
linetype	Line type such a solid, dashed, dotted
labels	Text on the plot
shape	Shape of the geometry

Image from internet

Aesthetic mappings

Aesthetic mappings `aes()` describe how variables are mapped to visual properties or aesthetics. It takes aesthetic-variable pairs

- `aes(x = displ, y = hwy, colour = class)`

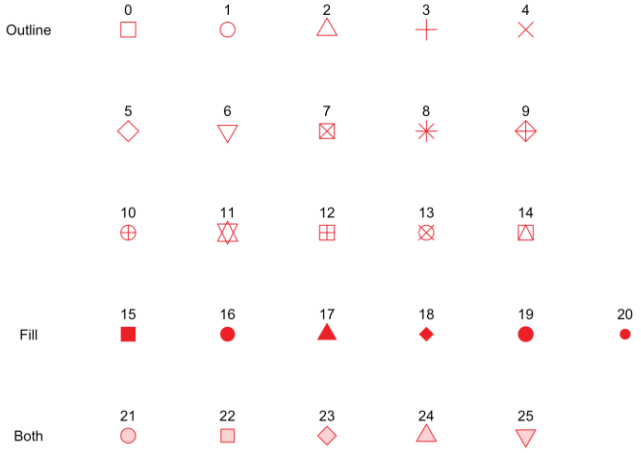
Aesthetic	Description
x	x-axis position
y	y-axis position
colour	Color of points or outlines of other shapes
fill	Fill color
size	size of the point or thickness of line
alpha	Transparency of the shape
linetype	Line type such a solid, dashed, dotted
labels	Text on the plot
shape	Shape of the geometry

Check available options

- `vignette("ggplot2-specs")`

linetype

shape



Aesthetic mappings and setting

Map an aesthetic to a variable when the appearance is governed by a variable

- `ggplot(mpg) + geom_point(aes(displ, hwy, colour = "blue"))`

Set the aesthetic atrribute to a single value in the layer parameters

- `ggplot(mpg) + geom_point(aes(displ, hwy), colour = "blue")`

Geometry

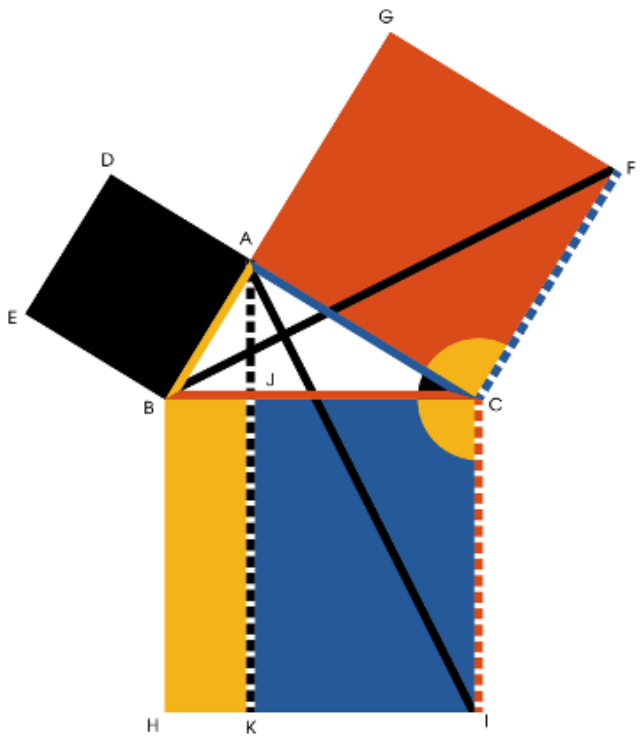
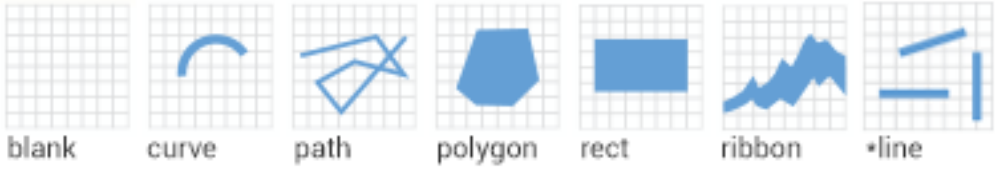


Image from book *The Elements of Euclid*

Geometric objects

Geometric objects **geom**s control plot type

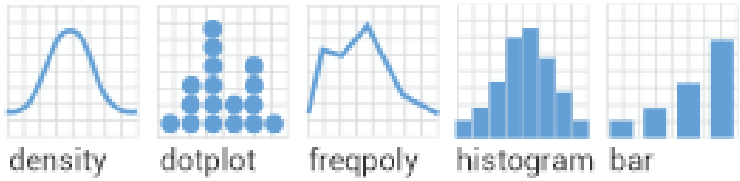
Function	Usgae
`geom_blank()`	display nothing.
`geom_segment()`	draw a line segment, specified by start and end position
`geom_abline()`	draw a straight line, specified by slope and intercept
`geom_path()`	connect observations in order of the data
`geom_line()`	connect observation in order of the variables on x axis
`geom_rect()`	draw rectangles
`geom_ribbon()`	draw ribbons, a path with vertical thickness



One variable

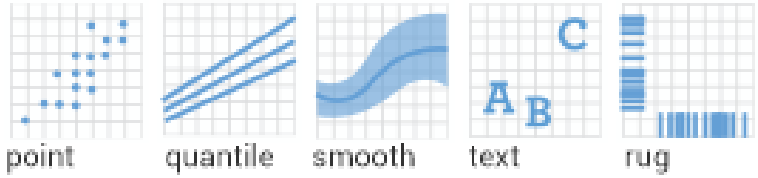
- Discrete:
  - `geom_bar()`: display count distribution of discrete variable

- Continuous:
  - `geom_histogram()`: bin and count continuous variable, display with bars
  - `geom_density()`: smoothed density estimate

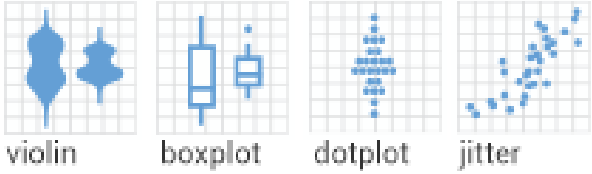


Two variables

- Both continuous:
  - `geom_point()`: scatterplot
  - `geom_smooth()`: smoothed line of best fit
  - `geom_rug()`: marginal rug plots



- One continuous, one discrete:
  - `geom_boxplot()`: boxplots
  - `geom_violin()`: show density of values in each group
  - `geom_jitter()`: randomly jitter overlapping points



Choose geometry based on visualization goal

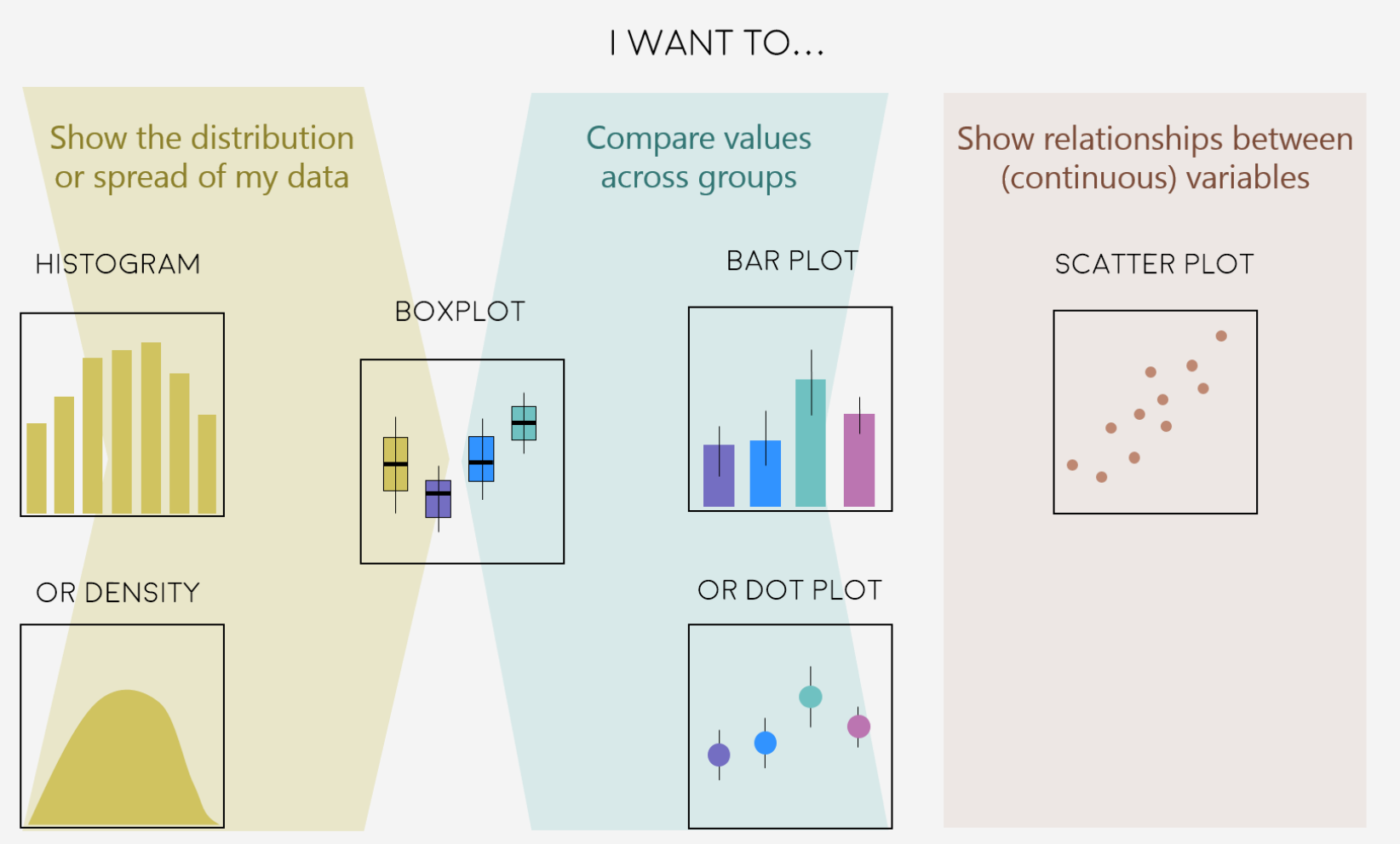
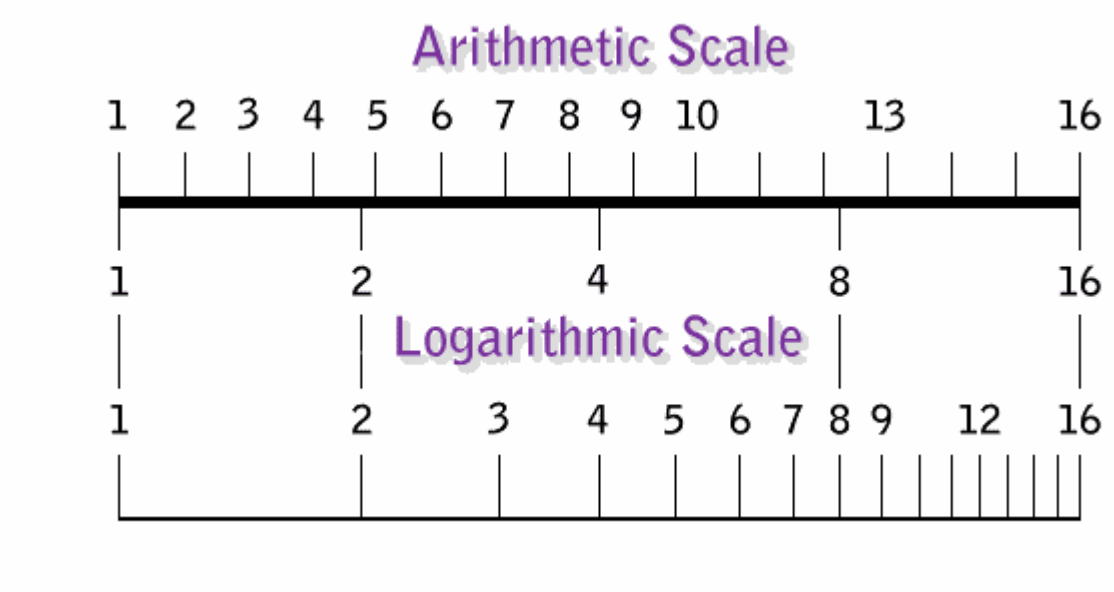


Image from Internet

Scales



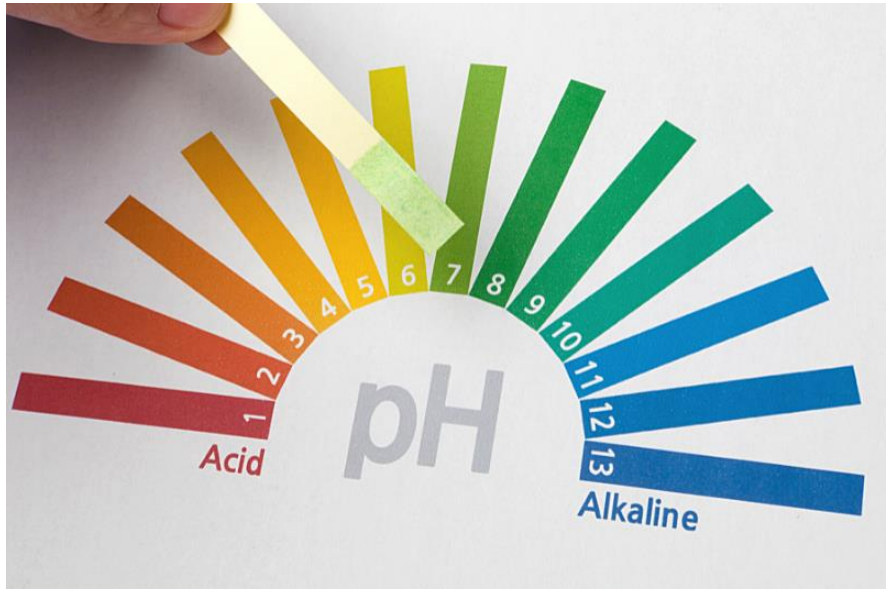


Image from Internet

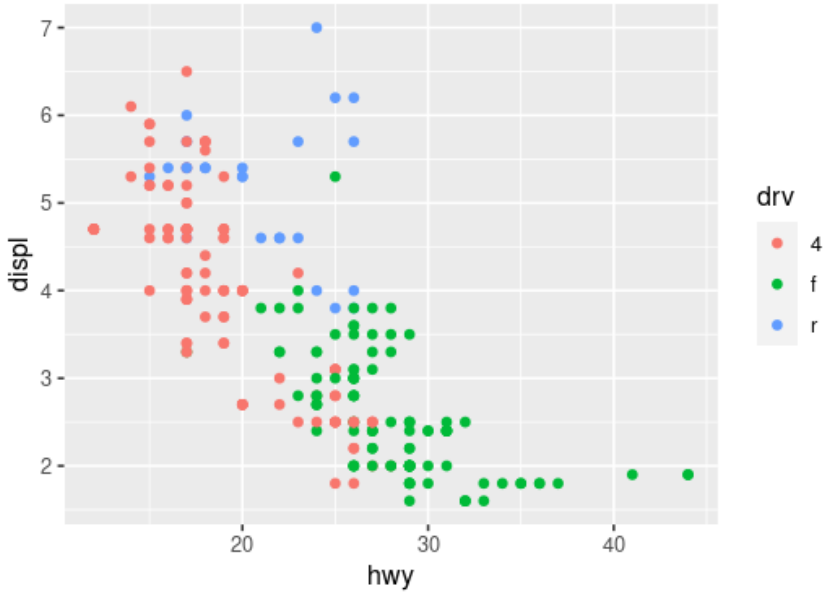
Scale settings

Syntax: `scale_<aesthetic>_<type>`

Aesthetics: `position`, `color`, `fill`, `size`, `shape`, `alpha`, `linetype`

Type: `continuous`, `discrete`

Example:



`aes()` maps the color attribute to a variable in data
`scale*` maps the color value to the variable value

Position scales

syntax: `scale_<axis>_<type>`

axis: x, y

type: `discrete`, `continuous`

arguments: `name`, `limits`, `breaks`, `labels`

Position scales

Name:

- `scale_x_continuous(names = "x")`

Limits:

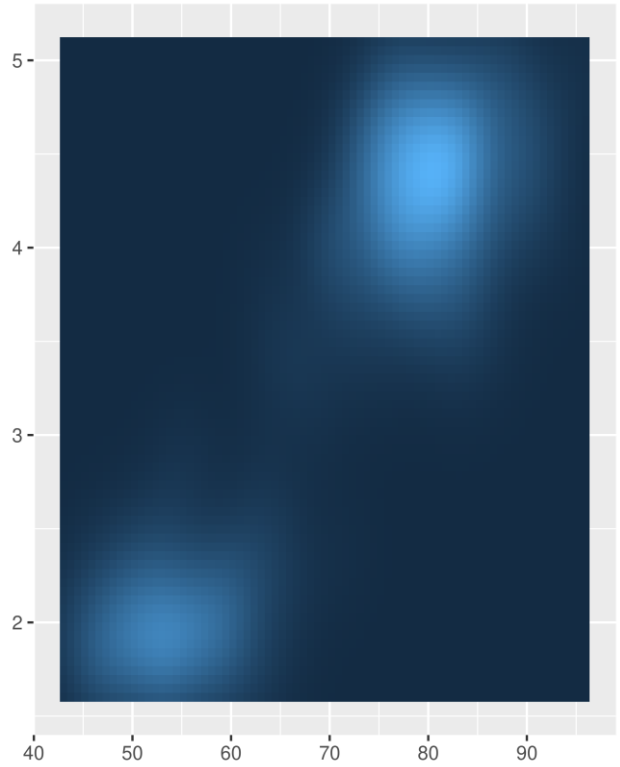
- `scale_x_continuous(limits=c(1,7))`

Breaks:

- `scale_x_continuous(breaks = c(50, 75, 100))`

Labels:

- `scale_x_continuous(labels = c(1/2, 3/4, 1))`



Color scales

syntax: `scale_<aes>_<type>`

aes: `fill`, `color`

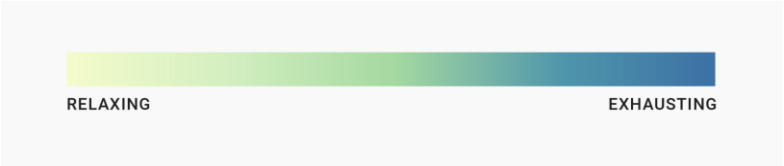
type: `discrete`, `continuous`, `gradient`, `manual`…

use `fill` as an example

Color scales (continuous)

Set color palettes

- `scale_fill_gradient()`: produces a 2 colour gradient
  - `scale_fill_gradient(low = "grey", high = "brown")`



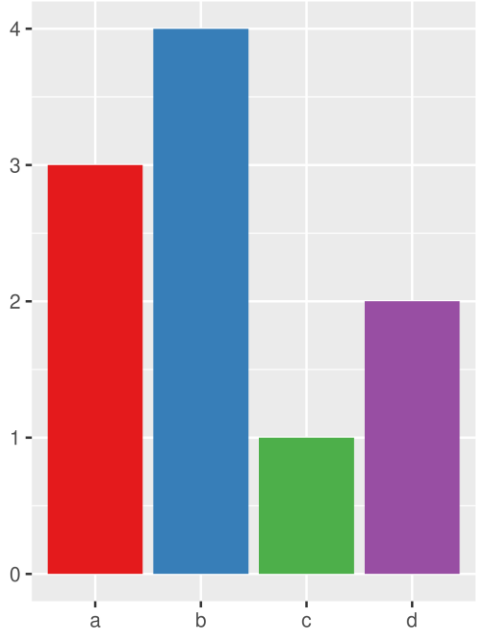
- `scale_fill_gradient2()`: produces a 3 colour gradient with midpoint
  - `scale_fill_gradient2(low = "grey", mid = "white", high = "brown", midpoint = .02 )`

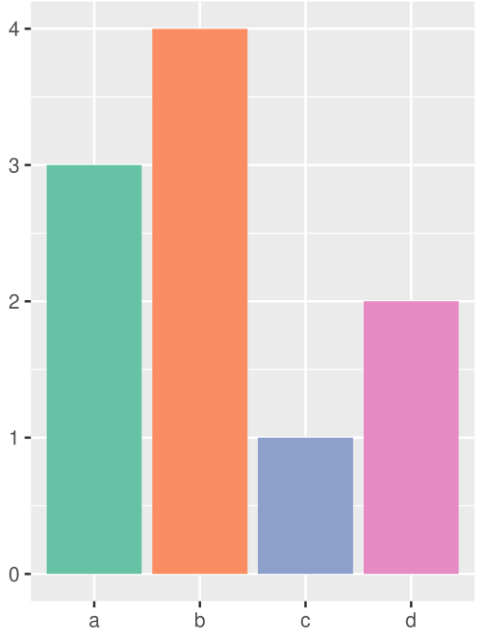
- `scale_fill_gradientn()`: produces an n-colour gradient

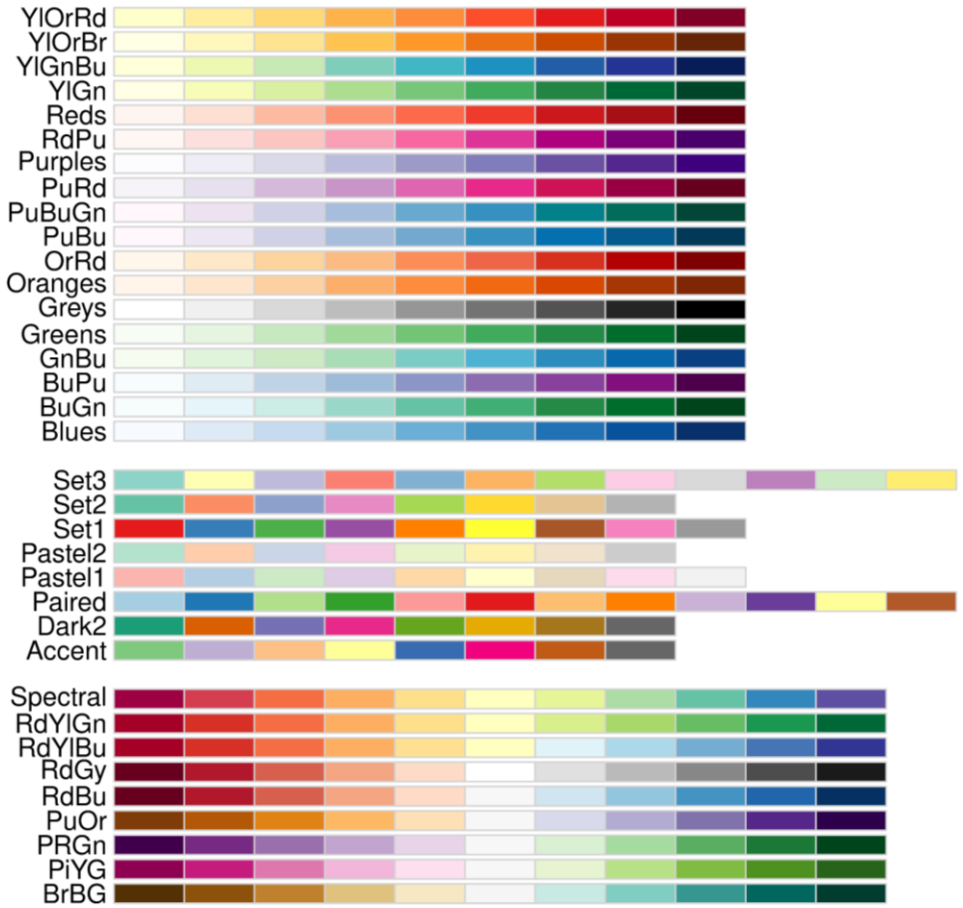
Color scales (discrete)

Brewer scales

- `scale_fill_brewer(palette="Set1")`
- `scale_fill_brewer(palette="Set2")`







Color scales (discrete)

Manually set color

Use a vector

- `scale_fill_manual(values = c("grey", "black", "grey", "grey"))`

Use a named vector

- `scale_fill_manual(values = c( "d" = "grey", "c" = "grey",`
  `"b" = "black", "a" = "grey" ))`

Facet

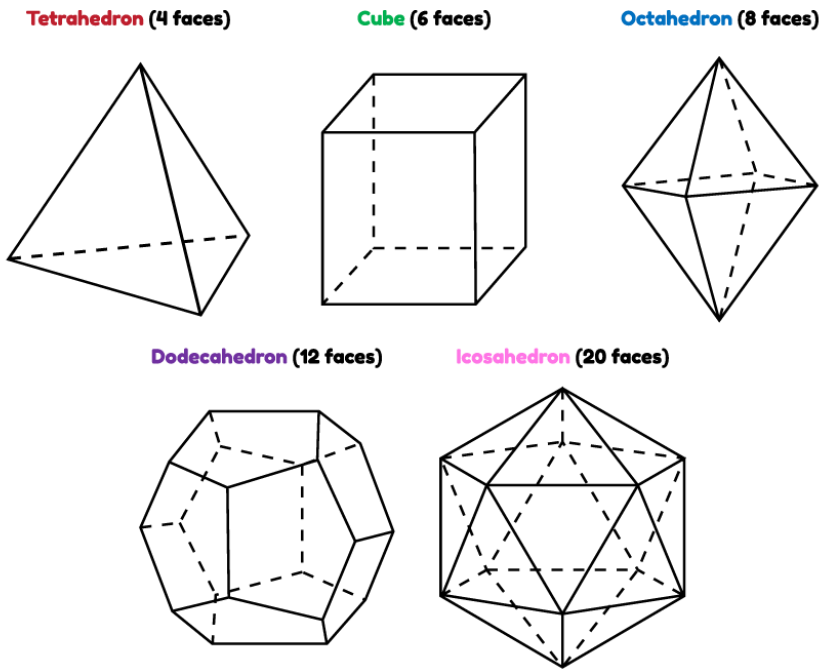
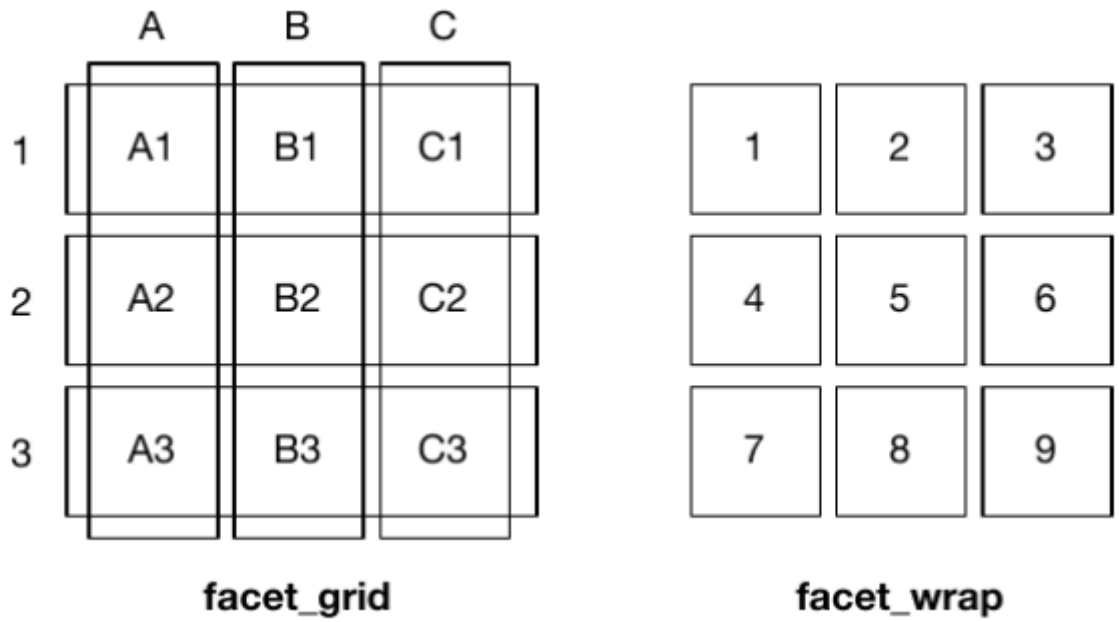


Image from Internet

3 types of facet

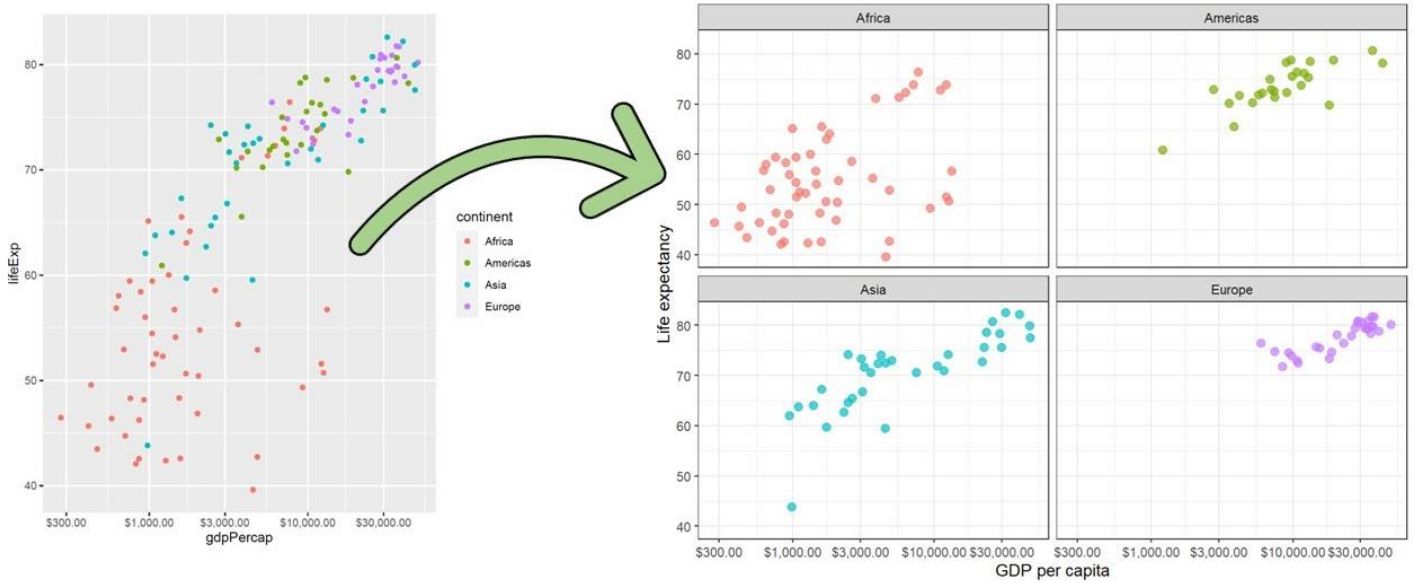
- `facet_null()`: a single plot, the default.
- `facet_wrap()`: “wraps” a 1d ribbon of panels into 2d.
- `facet_grid()`: produces a 2d grid of panels defined by variables which form the rows and columns.



Facet wrap

`facet_wrap()` makes a long ribbon of panels (generated by any number of variables) and wraps it into 2d

- `facet_wrap(~a, …)`

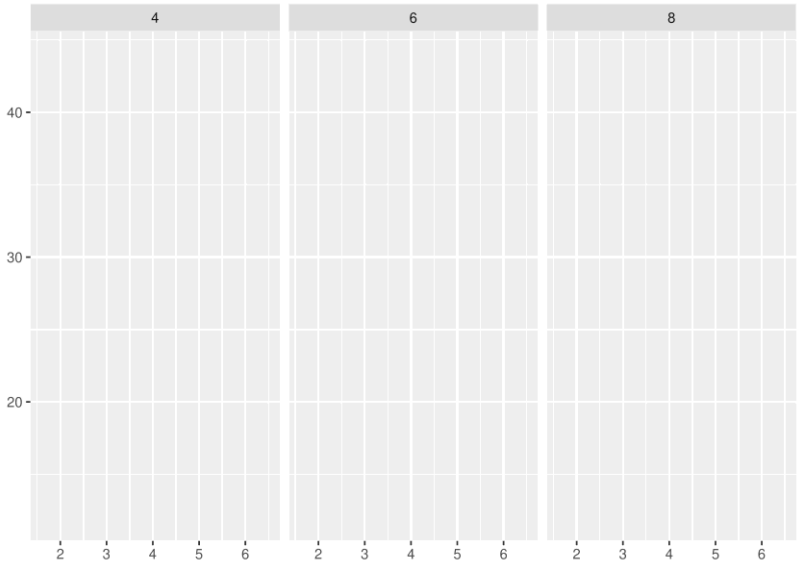


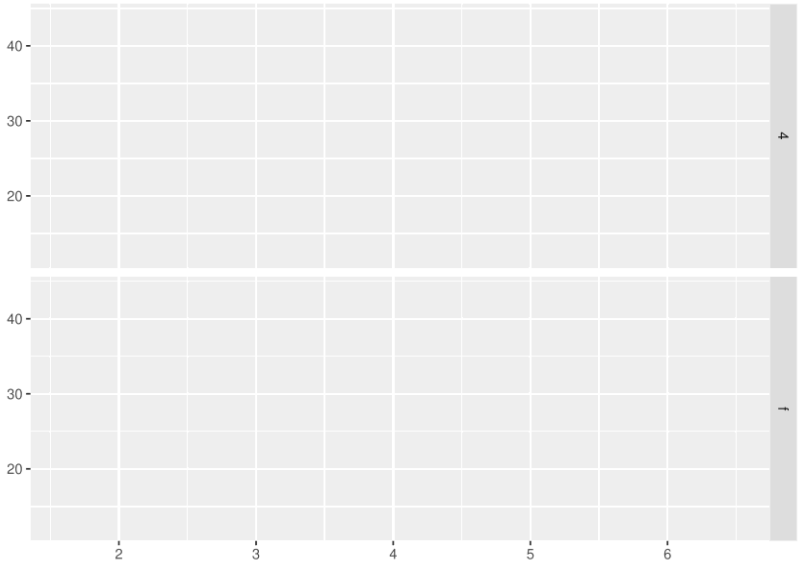
Parameters

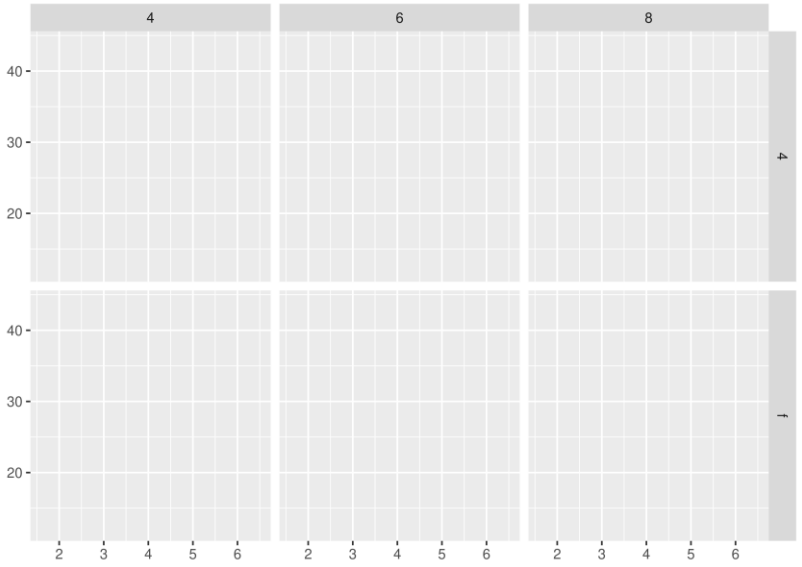
- `ncol`, `nrow`: controls how many rows/columns in the 2d panel
- `dir`: controls the direction of wrap: horizontal or vertical.

Facet grid

- `facet_grid()` lays out plots in a 2d grid
  - `facet_grid(. ~ a)`: spreads the values of a across the columns
  - `facet_grid(b ~ .)`: spreads the values of b down the rows
  - `facet_grid(b ~ a)`: b ~ a spreads a across columns and b down rows







Controlling panel scales

control whether the position scales are the same in all panels (fixed) or allowed to vary between panels (free) with the `scales` parameter:

- `scales = "fixed"`: x and y scales are fixed across all panels.
- `scales = "free_x"`: the x scale is free, and the y scale is fixed.
- `scales = "free_y"`: the y scale is free, and the x scale is fixed.
- `scales = "free"`: x and y scales vary across panels.

Plot customization

Image from book Internet

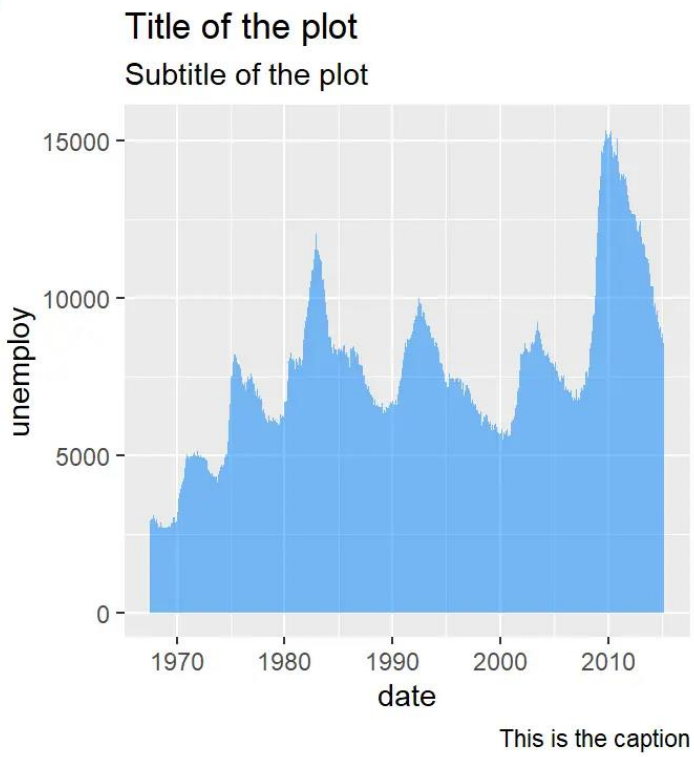
Plot title

Change the title:

- ```
  labs(title = "Title of the plot",
       subtitle = "Subtitle of the plot",
       caption = "This is the caption",
       tag = "Fig. 1")
  ```

- ```
  ggtitle("Title of the plot",
          subtitle = "Subtitle of the plot")
  ```

Fig. 1



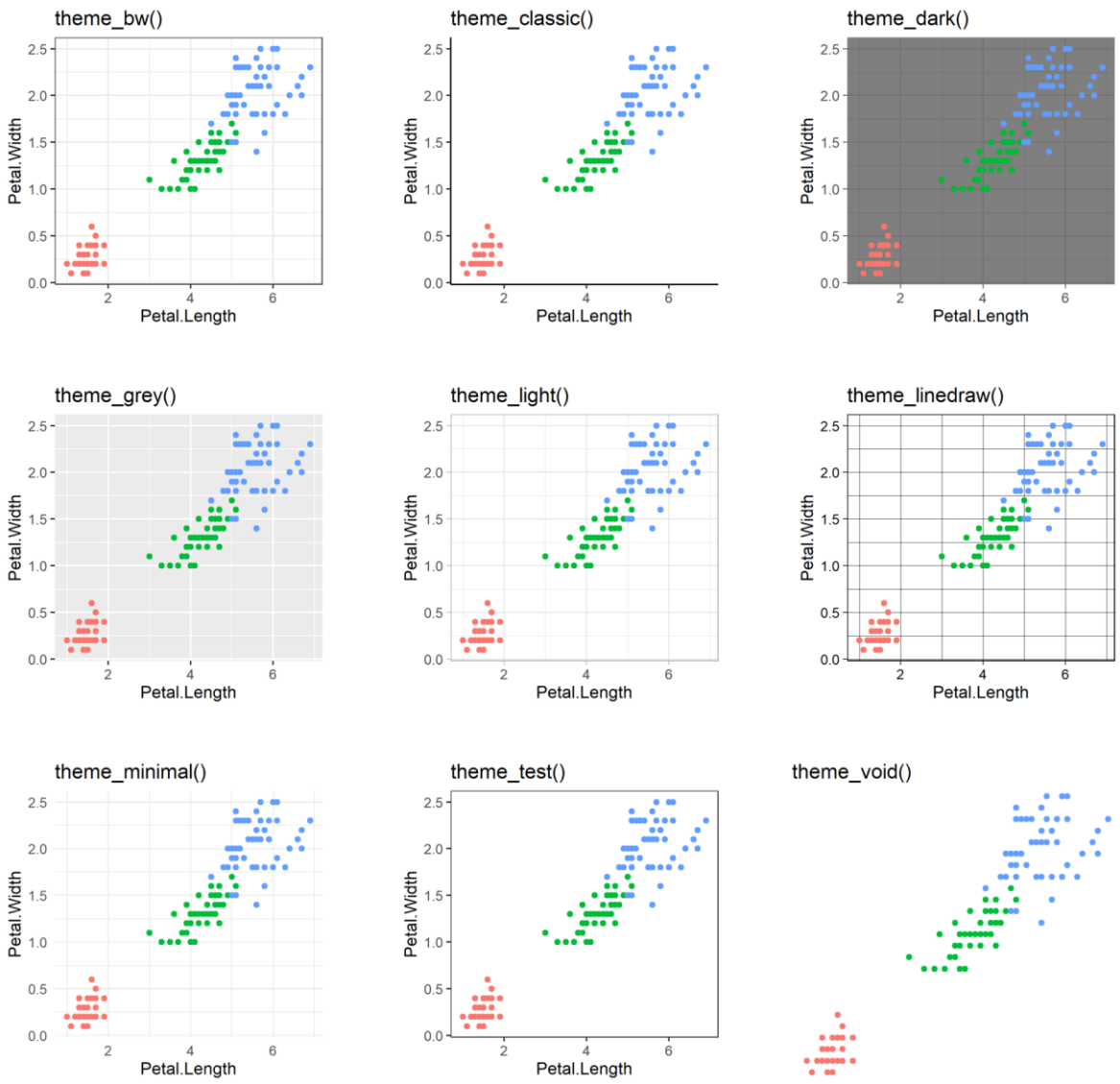
themes in ggplot2

Applying theme for a plot at a time

- `plot + theme_bw()`

Use `theme_set()` to change the default theme for all plots

- `theme_set(theme_bw())`



Save plots

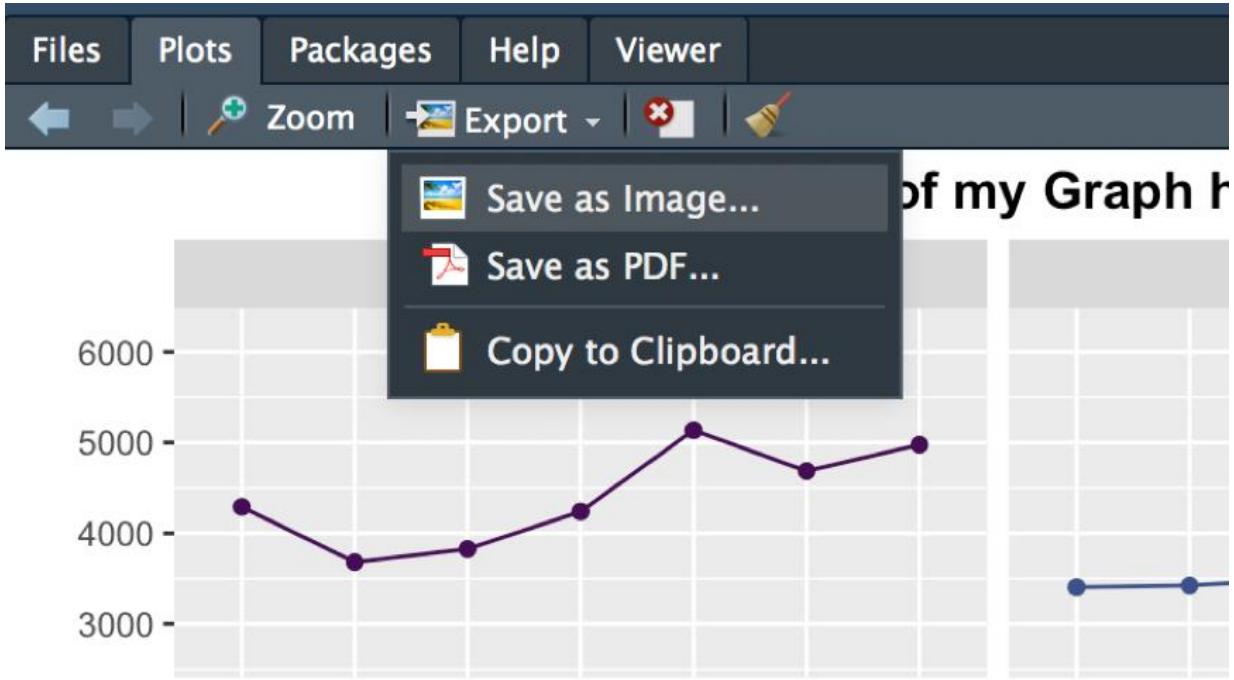

Files
Plots
Packages
Help
Viewer
Zoom
Export
Save as Image...
Save as PDF...
Copy to Clipboard...
of my Graph h
6000
5000
4000
3000

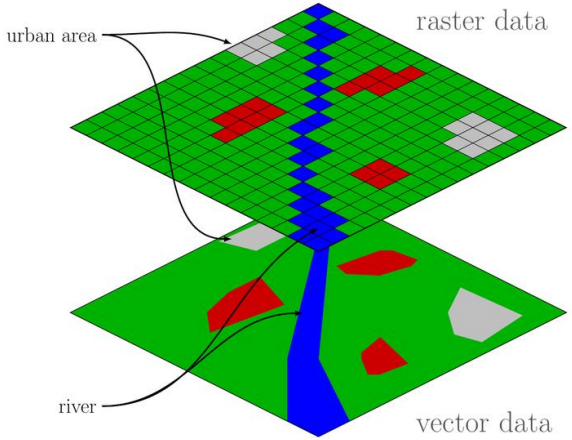
Figure file format

Vector graphics describe a plot as sequence of operations:

- draw a line from ($x1$, $y1$) to ($x2$, $y2$)
- draw a circle at ($x3$, $x4$) with radius r

This means that they are effectively ‘infinitely’ zoomable; there is no loss of detail. Widely-used vector graphic formats are `pdf` and `svg`

Raster graphics store plot as an array of pixel colours and have a fixed optimal viewing size. Widely-used raster graphic format are `png` and `jpg`



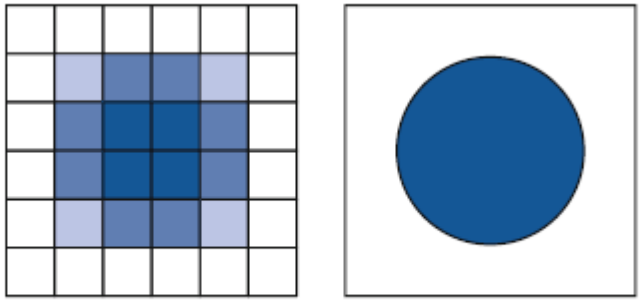


Fig. 8.1 The schematic difference between raster (*left*) and vector (*right*) graphics

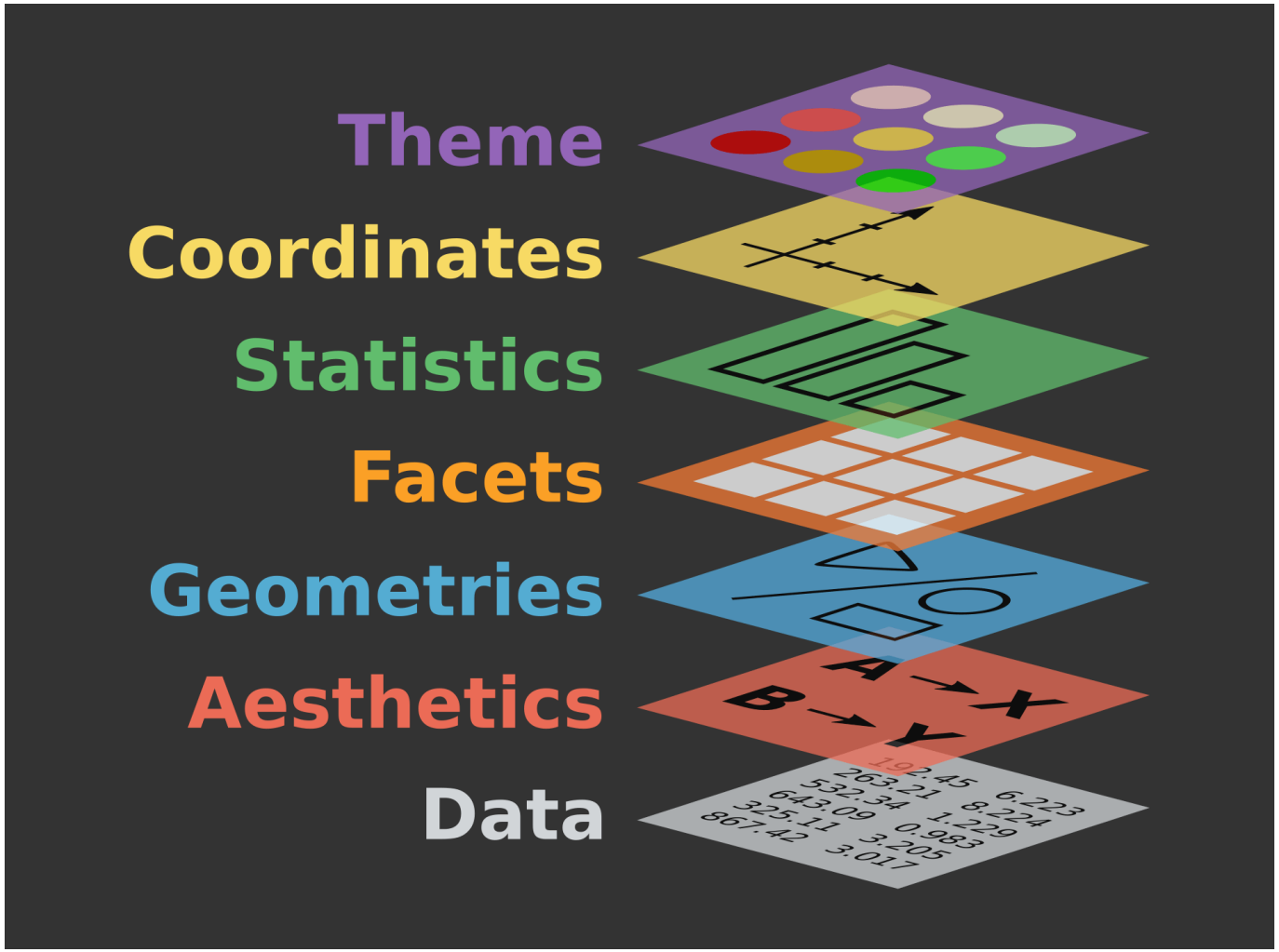
Save plots to file

- `pdf()/png()`: the standard R approach where you open a graphics device, generate the plot, then close the device
  - `pdf("output.pdf", width = 6, height = 6)`
  - `ggplot(mpg, aes(displ, cty)) + geom_point()`
  - `dev.off()`

- `ggsave()`: save the ggplot plot to a file, it can automatically select the graphics device based on the file extension
  - `plot = ggplot(mpg, aes(displ, cty)) + geom_point()`
  - `ggsave("output.pdf", plot)`

  Parameters
  - width and height: control the output size, specified in inches. If left blank, they'll use the size of the on-screen graphics device.
  - dpi: control the resolution of plots for raster graphics
  - plot: ggplot object. If omitted, it will save the last plot

ggplot2 visualization summary




```
ggplot (data = <DATA>) +
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
 stat = <STAT>, position = <POSITION>) +
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

required

Not required, sensible defaults supplied

Syntax of ggplot2

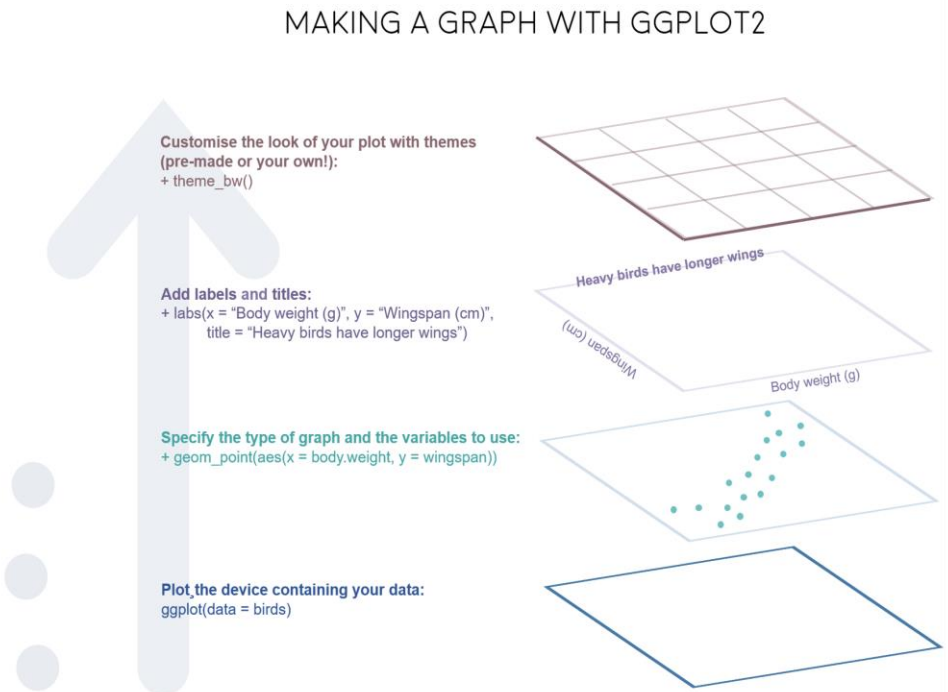
```
ggplot(df5, aes(date, cases, color = type)) +
  geom_point(size = 0.5) + geom_line(aes(y = cases7d)) +
  scale_x_date(date_breaks = "1 month", date_labels = "%d-%b") +
  scale_color_manual(values=c("darkorange2","firebrick","dodgerblue2")) +
  theme_classic(base_size = 24) +
  theme(axis.text.x = element_text(angle = 30, hjust = 1))
```

Layer	Function
Data	`ggplot(data)`
Aesthetics	`aes()`
Layers	`geom_*()` and `stat_*()`
Scales	`scale_*()`
Coordinate System	`coord_*()`
Facets	`facet_*()`
Visual Themes	`theme()` and `theme_*()`

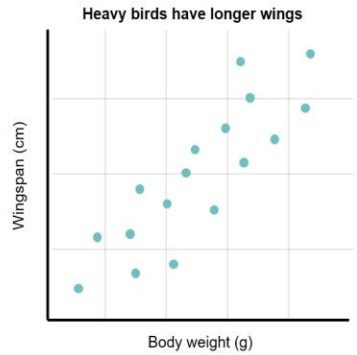
Image from Internet

ggplot2 visualization summary

- ❑ Data
- ❑ Aesthetic mappings
- ❑ Geometry
- ❑ Scale
- ❑ Facet
- ❑ Themes



For advanced usage of ggplot2, check the previous workshop:
https://github.com/wbvguo/qcbio-DataViz_w_ggplot2.git



https://madd.seas.gwu.edu/2021-Fall/class/3-rmd-plotting/index.html

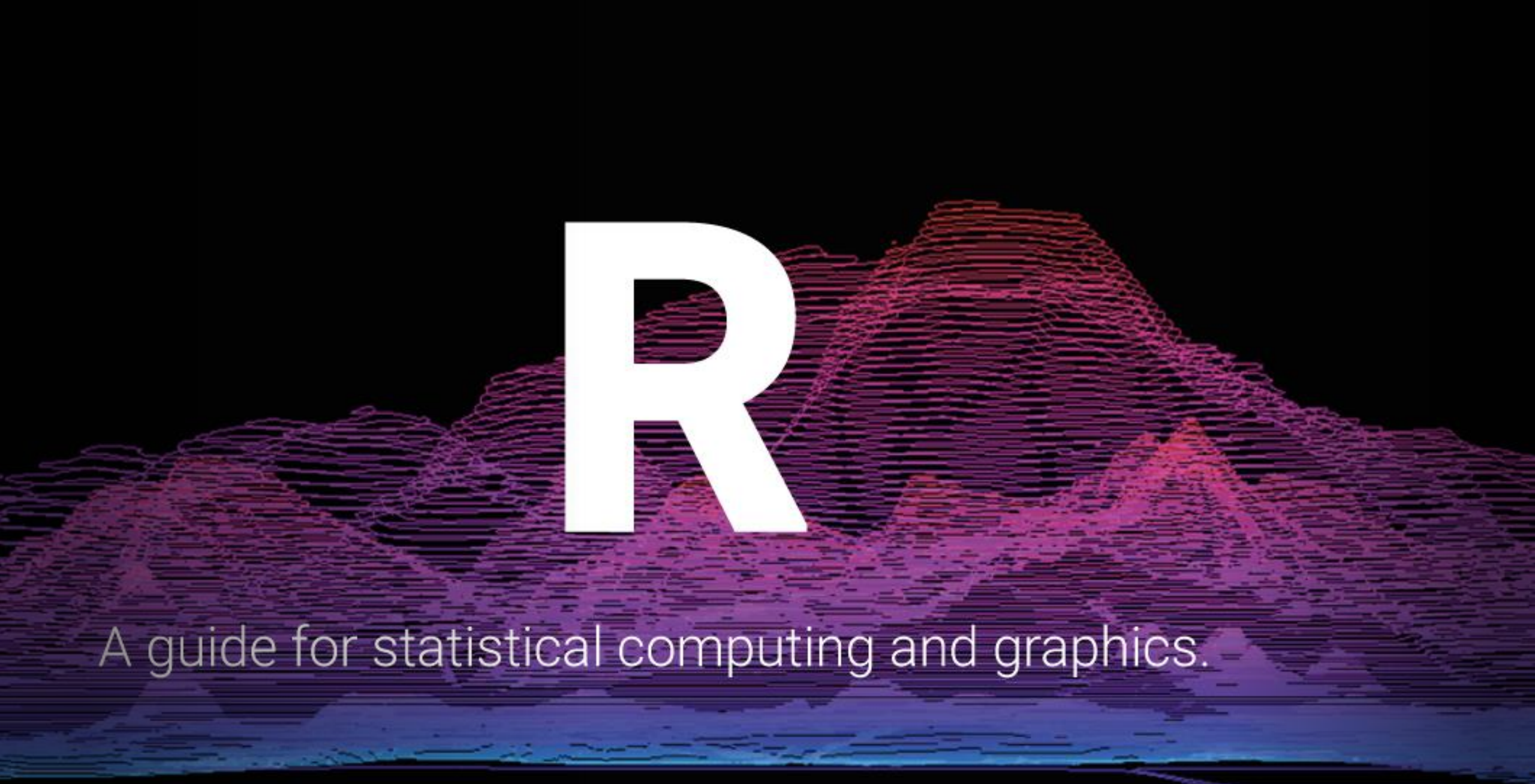
Additional quick resources

- R basics cheatsheet
- R data wrangling cheatsheet
- ggplot2 visualization cheatsheet
- R tutorials

Practice makes perfect

Where to get help?

- https://forum.posit.co/
- https://education.rstudio.com/
- https://www.google.com
- https://stackoverflow.com
- https://chat.openai.com/

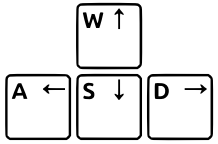


Thank you, good luck and have fun!

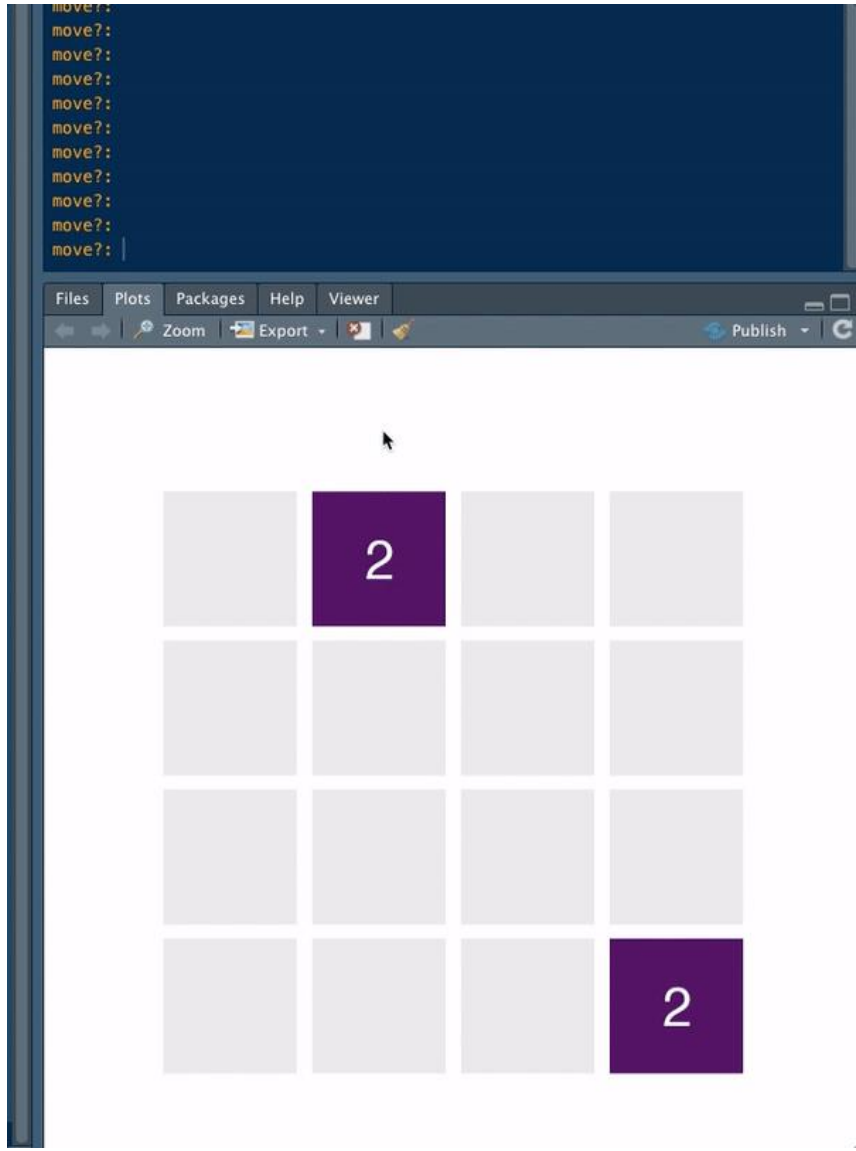
Bonus time: 2048 in R!

Code logic:

- ✓ Initialize a 4 x 4 matrix with 2 random cells with value 2
- ✓ Program waits for and records to user input
  - Direction signals
  - Exit signal ("q")
  - Other signals



- ✓ React to user input
  - Direction signals: Merge blocks according to the direction, and randomly pop up a value of 2 in an empty cell. Update the matrix
  - Exit signal: terminate the program
  - Other signals: ignore
- ✓ Visualize the results using ggplot2
- ✓ Check if the program should terminate
  - 2048 is obtained (success)
  - no empty cell is left and no action can be performed (failed)



Check the cool code here