Socio-Informatics 348

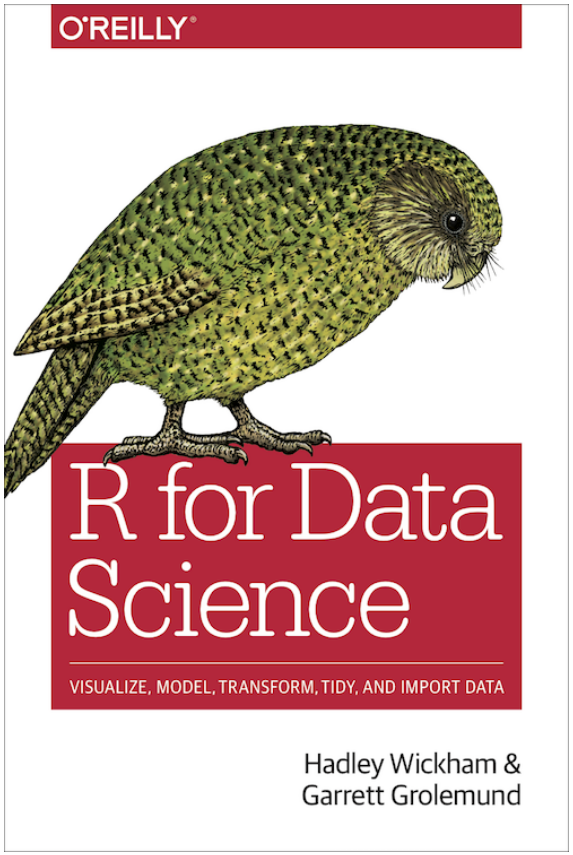
Data Visualisation

Layers

Dr Lisa Martin

Department of Information Science
Stellenbosch University

Today's Reading



R for Data Science, Chapter 9, Visualize

Chapter Overview

- Explore the **layered grammar of graphics**: aesthetics, geoms, stats, positions, coordinates, facets.
- Build more powerful plots with a flexible template.
- Understand how ggplot2 constructs and renders plots layer by layer.

Key Components of a Layered Plot

Basic template:

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

Key Components of a Layered Plot

Core elements of the grammar:

- Dataset
- Geom (geometric object)
- Aesthetic mappings
- Statistical transformation (stat)
- Position adjustment
- Coordinate system
- Faceting
- Theme – introduced later in Chapter 11

1. Data

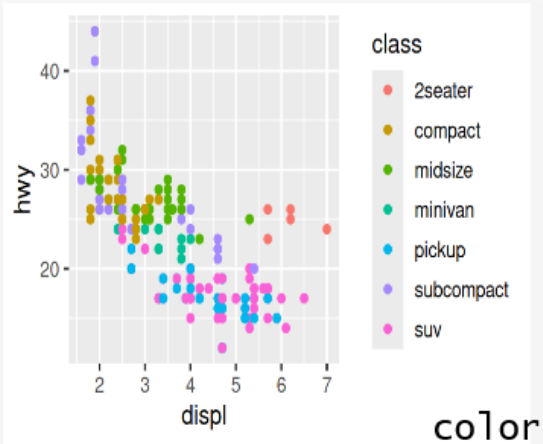
- The raw information to be visualised.
- Provided as a data frame or tibble.
- Each row = an observation, each column = a variable.
- Can supply data globally (to `ggplot()`) or locally (to a single geom).

2. Aesthetic Mappings

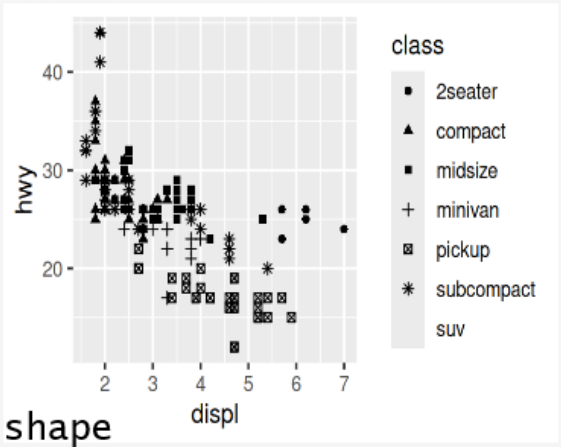
- Define how variables map to visual properties.
- Common aesthetics: `x`, `y`, `colour`, `size`, `shape`, `fill`.
- Specified inside `aes()`.
- Global mappings apply to all layers; local mappings apply only to a given geom.

2. Aesthetic Mappings

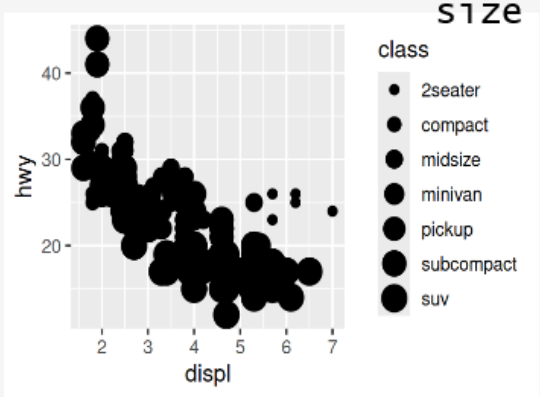
```
ggplot(mpg, aes(x = displ, y = hwy,        = class)) +
  geom_point()
```



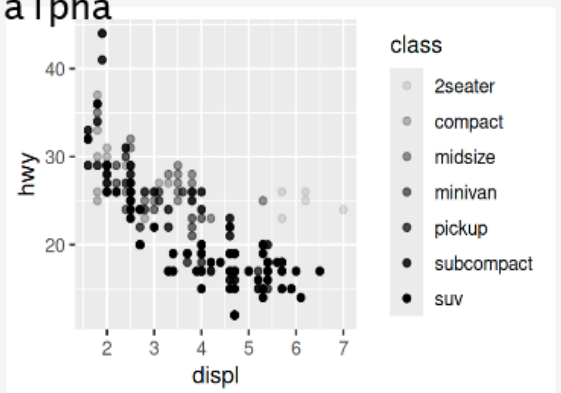
color



size



shape
alpha

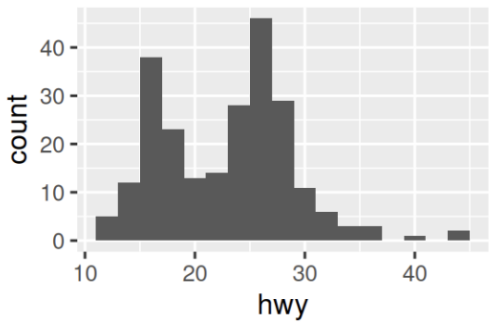


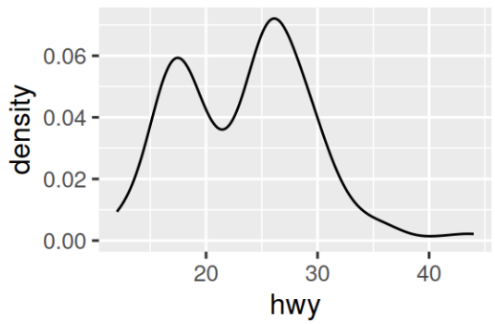
3. Geoms (Geometric Objects)

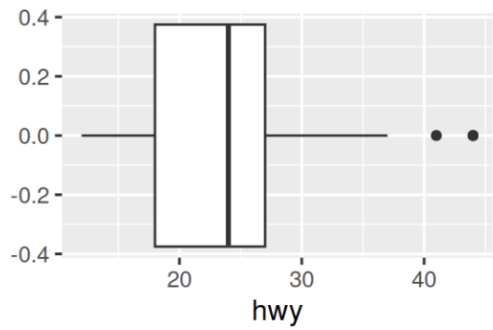
- Various ways to represent your data.
- Examples:
  - Points (`geom_point()`)
  - Lines (`geom_line()`)
  - Bars (`geom_bar()`)
  - Boxplots (`geom_boxplot()`)
- Choose the geom to suit the story you want to tell.

3. Geoms (Geometric Objects)

- Different geoms can reveal different patterns in your data.
- Histogram and density plots reveal a bimodal distribution
- Boxplot reveals outliers



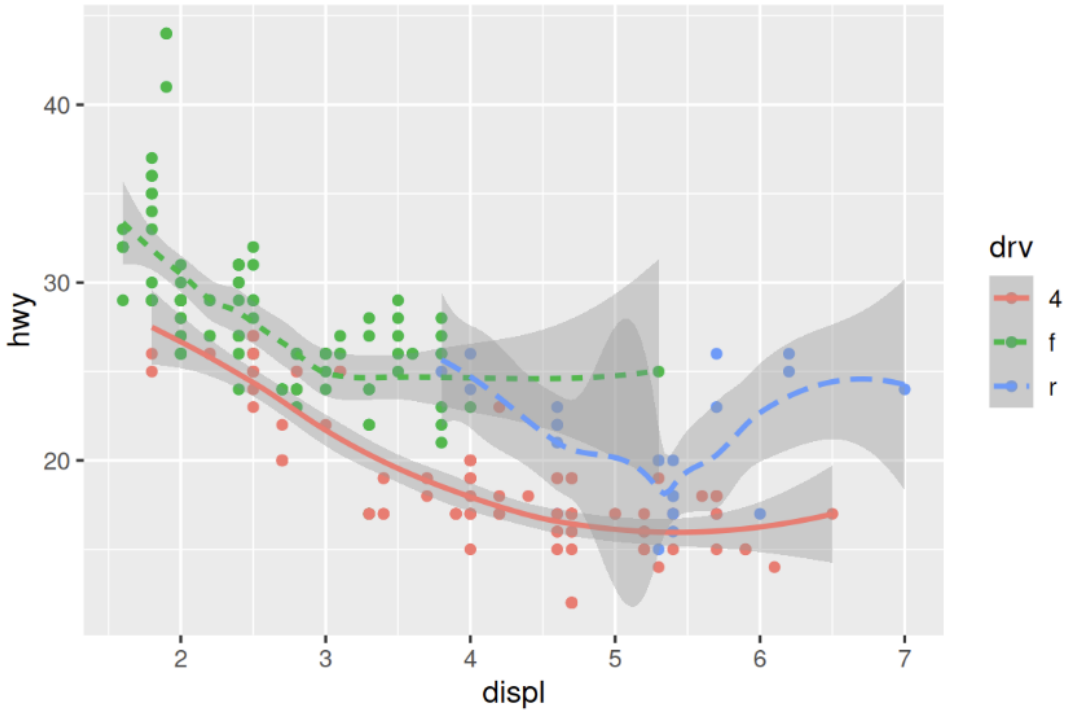




3. Geoms (Geometric Objects)

Layering geoms can help convey your story

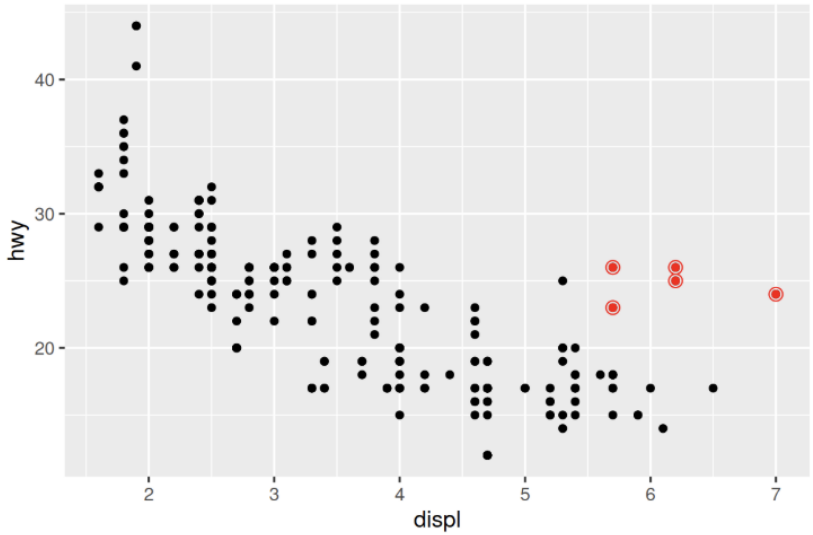
```
ggplot(mpg, aes(x = displ, y = hwy, color = drv)) +
  geom_point() +
  geom_smooth(aes(linetype = drv))
```



3. Geoms (Geometric Objects)

Layering geoms can help convey your story

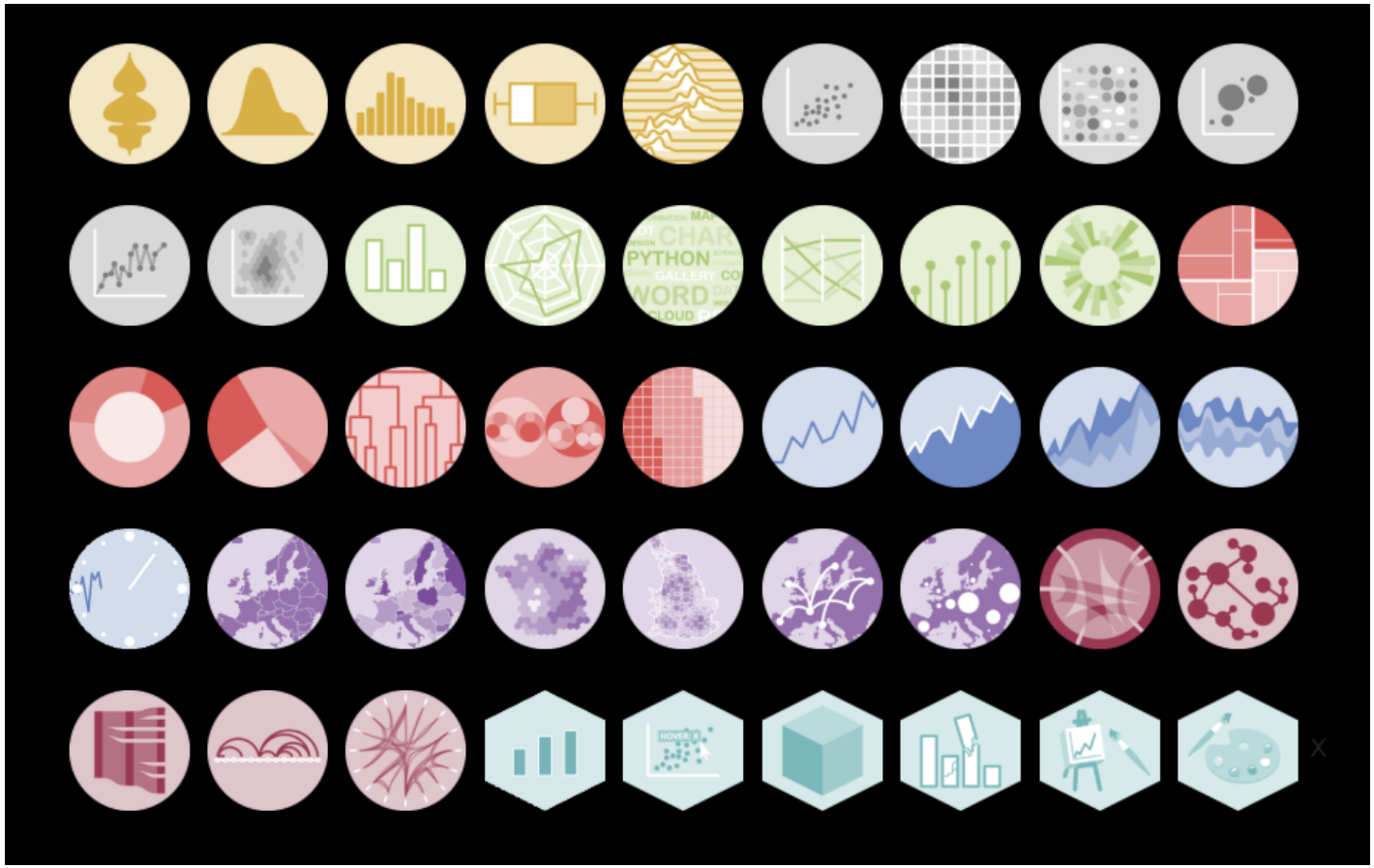
```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  geom_point(
    data = mpg |> filter(class == "2seater"),
    color = "red"
  ) +
  geom_point(
    data = mpg |> filter(class == "2seater"),
    shape = "circle open", size = 3, color = "red"
  )
```



3. Geoms (Geometric Objects)

Very useful website – R Graph Gallery

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



3. Geoms (Geometric Objects)

Very useful website – R Graph Gallery

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

3. Geoms (Geometric Objects)

Very useful website – R Graph Gallery

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

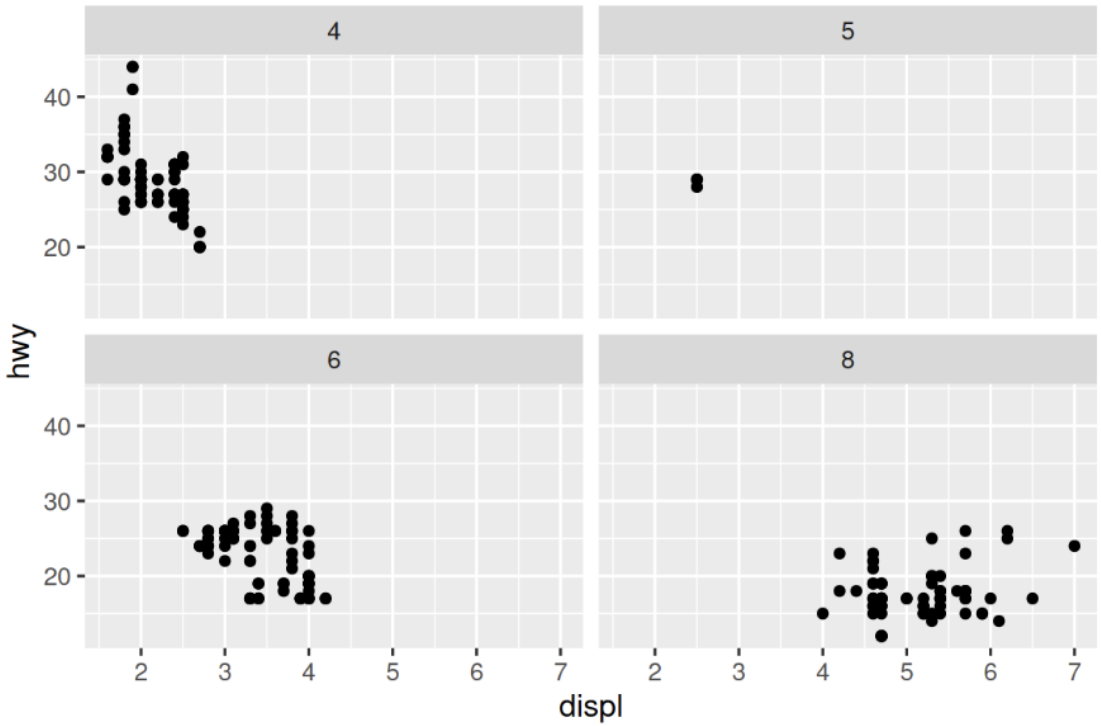
4. Faceting

- Split data into subplots for comparison.
- `facet_wrap()` — one variable, arranged in a grid.
- `facet_grid()` — two variables (rows $\times$ columns).
- Useful for exploring categorical comparisons.

4. Faceting

- `facet_wrap()` — one variable, arranged in a grid.

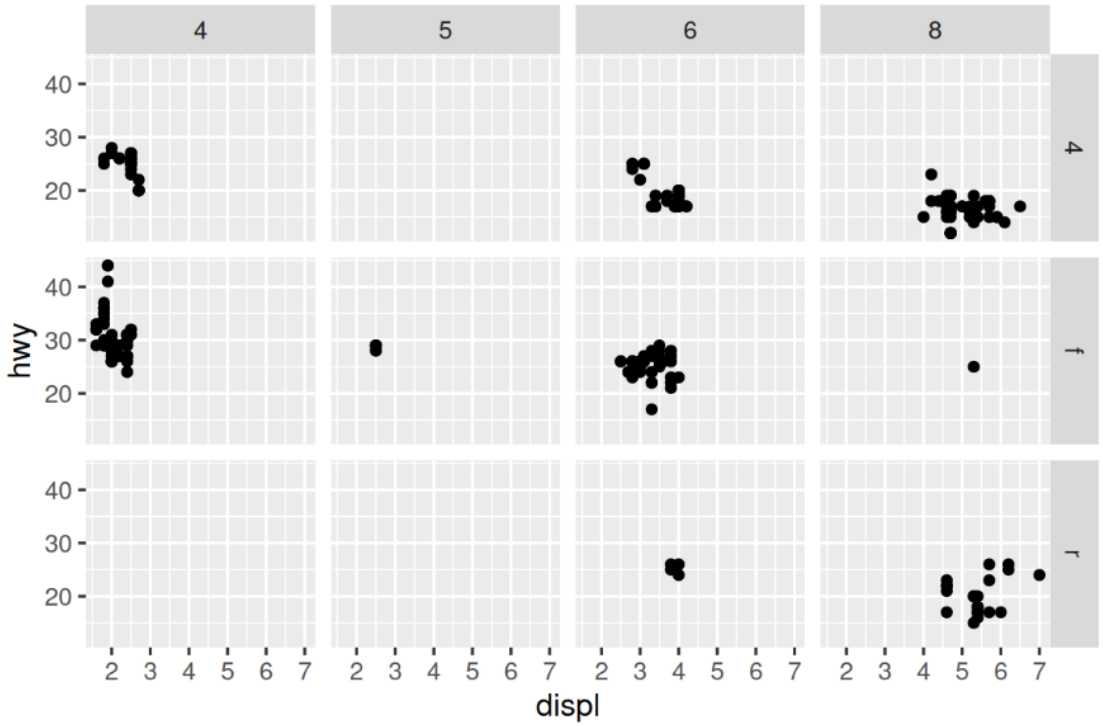
```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  facet_wrap(~cyl)
```



4. Faceting

- `facet_grid()` — two variables (rows × columns).

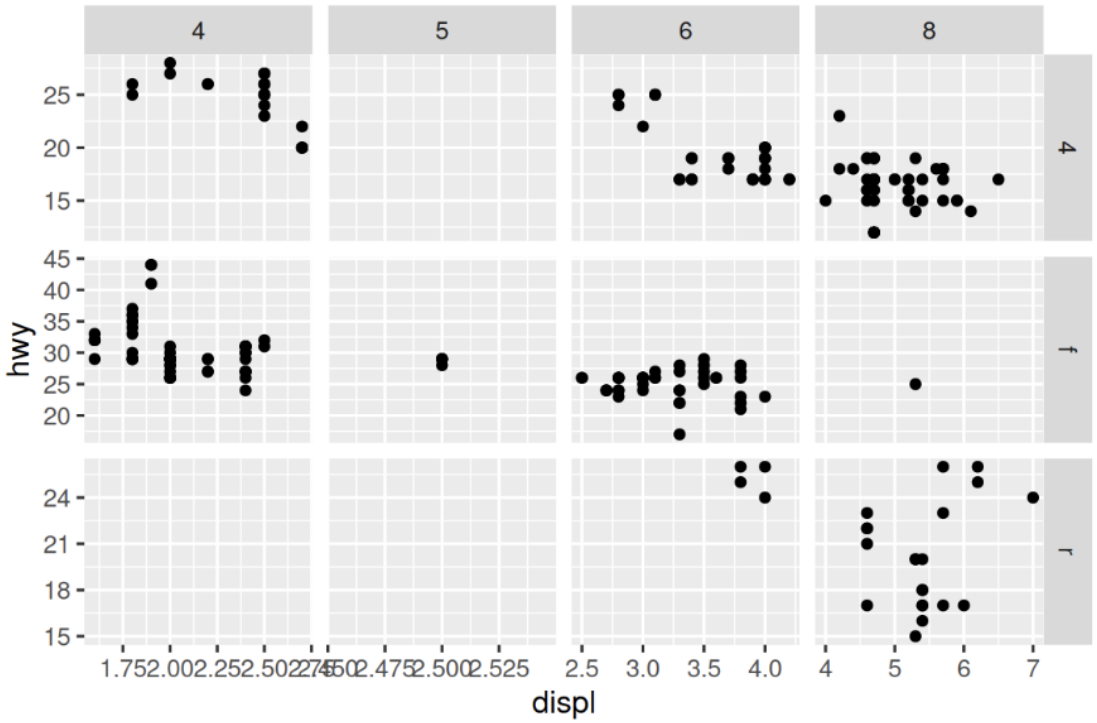
```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  facet_grid(drv ~ cyl)
```



4. Faceting

- `scales = "free" / "fixed" / "free_y" / "free_x"`

```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point() +
  facet_grid(drv ~ cyl, scales = "free")
```



5. Stats (Statistical Transformations)

- Transform raw data before plotting.
- Many geoms have a default stat.
- Examples:
  - Count cases for bar charts (`stat_count`).
  - Smooth a trend (`stat_smooth`).
  - Bin continuous variables (`stat_bin`).

6. Position Adjustments

- Resolve overlapping elements.
- Examples:
  - `position = "stack"` for stacked bars.
  - `position = "dodge"` for side-by-side bars.
  - `position = "jitter"` to add small random noise to points.
- Adjustments improve readability of plots = improved storytelling.

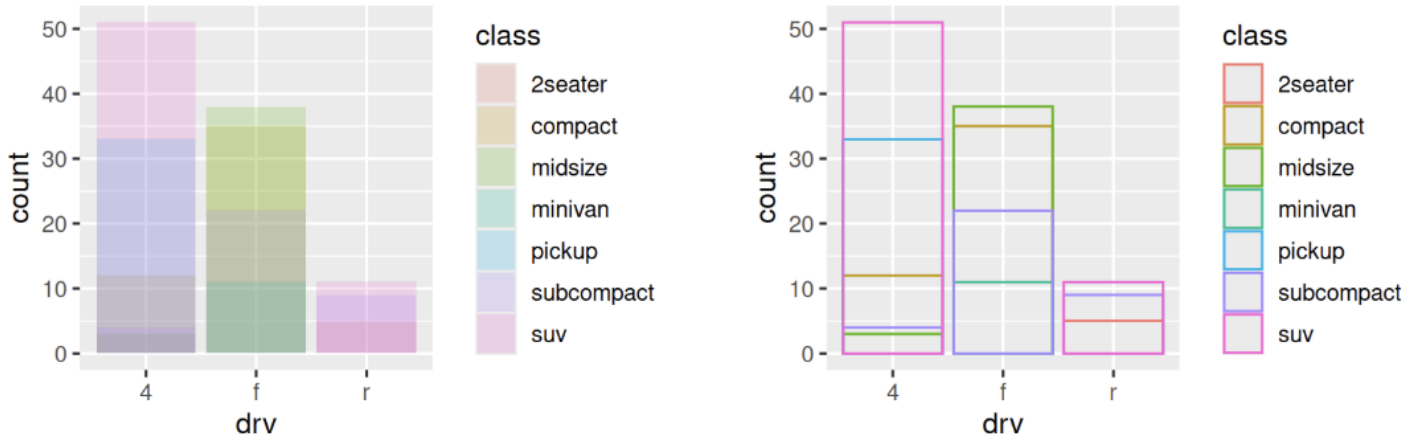
6. Position Adjustments

Bars: `position = "identity"`

- Places bars on top of each other.
- Not useful when we have multiple groups.

```
# Left
ggplot(mpg, aes(x = drv, fill = class)) +
  geom_bar(alpha = 1/5, position = "identity")

# Right
ggplot(mpg, aes(x = drv, color = class)) +
  geom_bar(fill = NA, position = "identity")
```



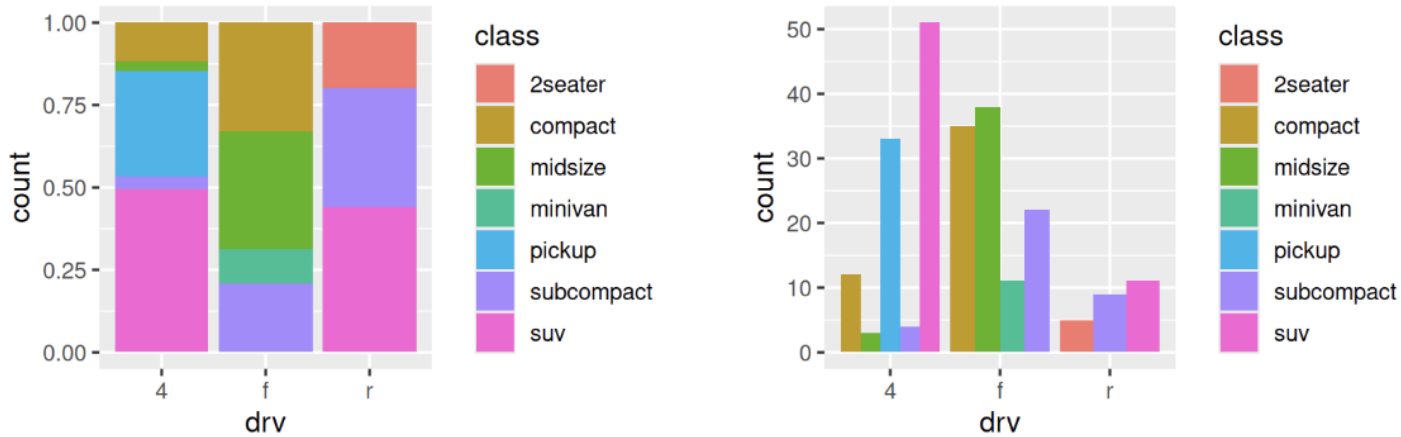
6. Position Adjustments

Bars: `position = "fill"` or `position = "dodge"`

- Alternative positions solve this problem.

```
# Left
ggplot(mpg, aes(x = drv, fill = class)) +
  geom_bar(position = "fill")

# Right
ggplot(mpg, aes(x = drv, fill = class)) +
  geom_bar(position = "dodge")
```

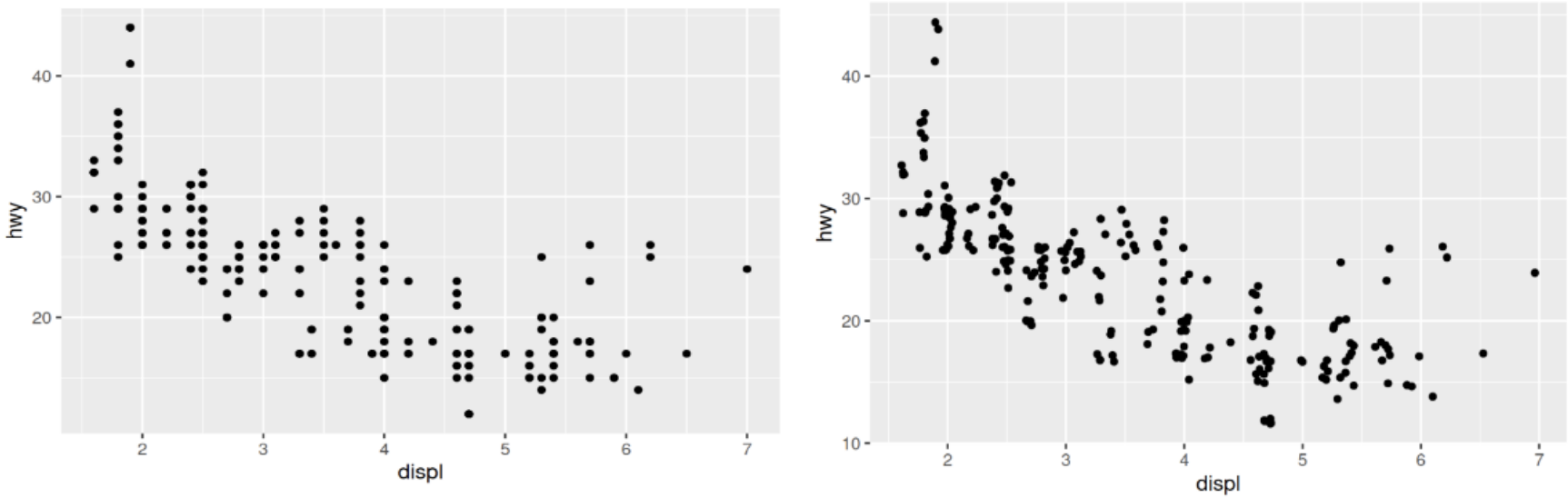


6. Position Adjustments

Scatterplot: `position = "jitter"`

- Actual distribution can be seen more clearly.

```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point(position = "jitter")
```



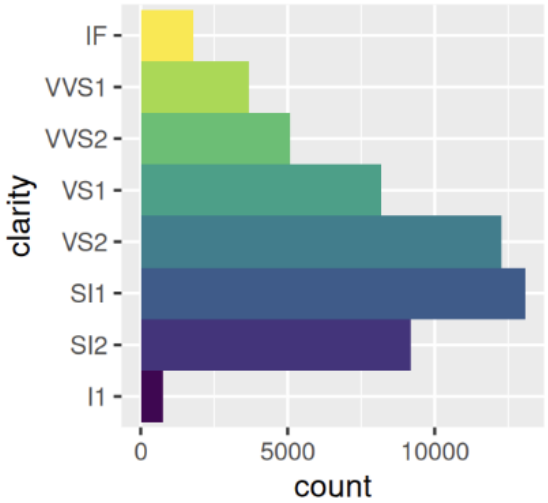
7. Coordinate Systems

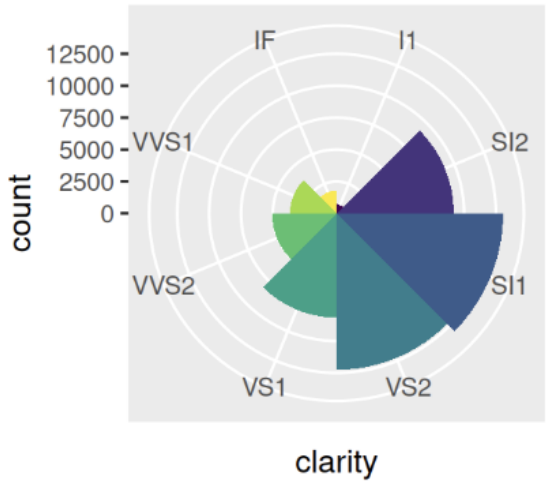
- Control how data coordinates are mapped to the plot plane.
- Common systems:
  - Cartesian (default) - x and y positions
  - Polar (`coord_polar()`) – circular plots
  - Flipped axes (`coord_flip()`) – swap x and y
- Note that plotting spatial data is outside the scope of this course.

7. Coordinate Systems

```
bar <- ggplot(data = diamonds) +
  geom_bar(
    mapping = aes(x = clarity, fill = clarity),
    show.legend = FALSE,
    width = 1
  ) +
  theme(aspect.ratio = 1)

bar + coord_flip()
bar + coord_polar()
```





Layers in Action

1. Start with **raw data**.
2. Add a **geom** to visualize transformed data.
3. Map data variables to visual properties via **aesthetics**.
4. Apply a **statistical transformation** (e.g., counts, medians).
5. Transform to plot space via a **coordinate system**.
6. Optionally, tweak **geom positions** to avoid overlap.
7. Optionally, apply **faceting** to split plot into subplots.
8. Stack additional layers to combine more visual elements.

Concluding Remarks

You should now understand:

- How ggplot2 decomposes plots into layered grammatical components.
- That this grammar enables building complex, customised, multi-layer plots.
- The role of each layer: mappings, geoms, stats, positions, coordinates, facets.
- That extending ggplot2 (e.g., with new geoms) is a common practice.