ST720 Data Science

Grammar of Graphics

Seung Jun Shin (sjshin@krea.ac.kr)

Department of Statistics, Korea University

Introduction

- Important questions in statistical graphics:
  - What is a graphic?
  - How can we succinctly describe a graphic?
  - And how can we create the graphic that we have described?
- One way to answer these questions is to develop a grammar.
  - A grammar provides a strong foundation for understanding a diverse range of graphics.
  - A grammar may also help guide us on what a well-formed or correct graphic looks like, but there will still be many grammatically correct but nonsensical graphics (like English).

Introduction

- `ggolot2` (Wickham, 2010) is the implementation of *Grammar of Graphics* (Wilkinson, 2005).
- Need to master the underlying grammar to unlock the full power of `ggplot2`.

A Basic Plot

- Let us draw a scatterplot of A versus C:

A	B	C	D
2	3	4	a
1	2	1	a
4	5	15	b
9	10	80	b

Q. What exactly is a scatterplot?

A. One way to describe it is that we are going to draw a point for each observation, and we will position the point horizontally according to the value of A, and vertically according to C. For this example, we will also map categorical variable D to the shape of the points.

Mapping To Aesthetics

- The first step is to create a new dataset that reflects the mapping of x-position to A, y-position to C, and shape to D.
- x-position, y-position, and shape are examples of aesthetics (things that we can perceive on the graphic)
- `ggplot2` removes all others that do not appear in the plot, and thus yields

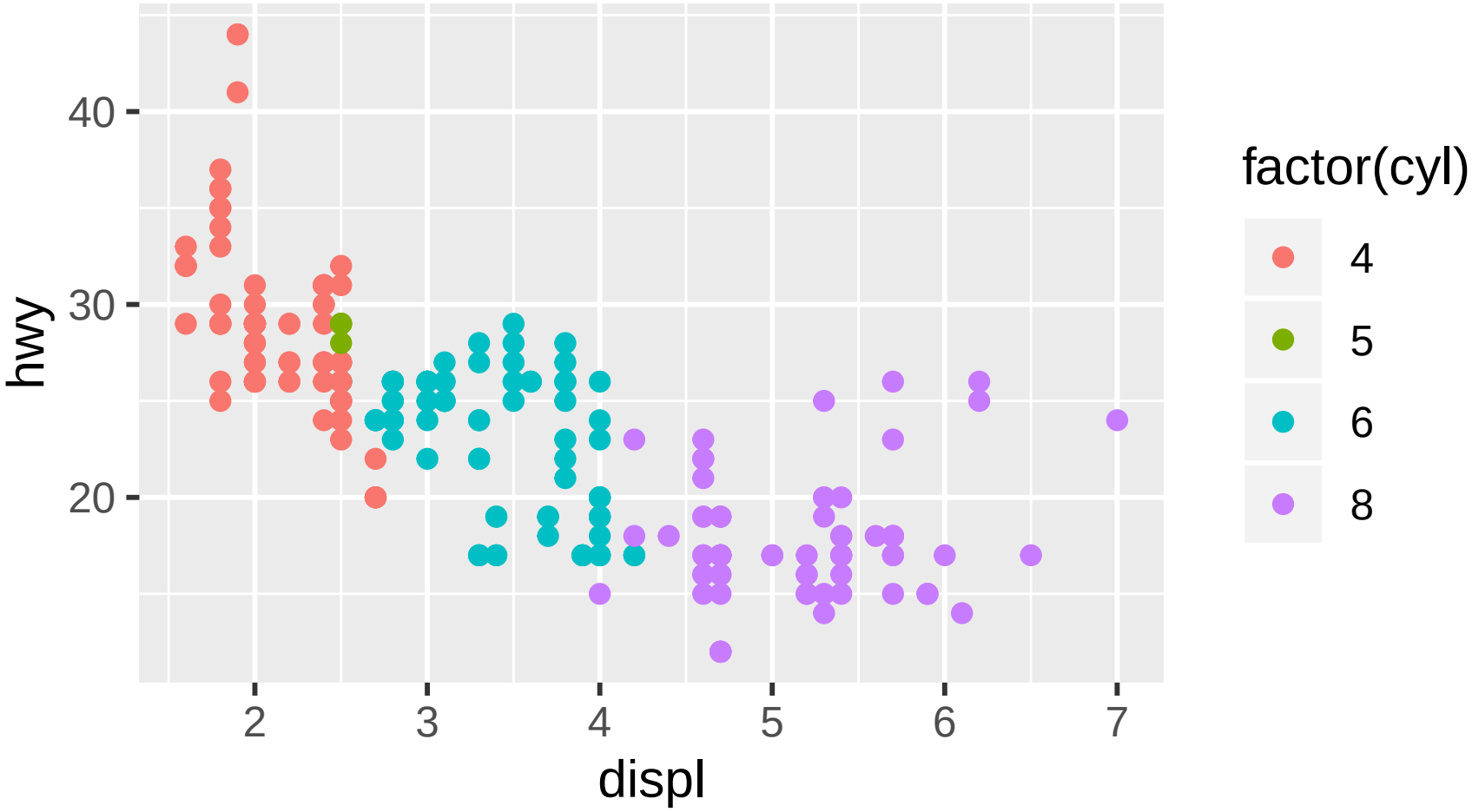
x	y	Shape
2	4	a
1	1	a
4	15	b
9	80	b

Geometric Objects

- We can create many different types of plots using this same basic specification.
- For example, if we were to draw lines instead of points, we would get a line plot.
- If we used bars, we would get a bar plot.
- Bars, lines, and points are all examples of geometric objects.

Mapping To Aesthetics: mpg data

```
ggplot(mpg, aes(displ, hwy, colour = factor(cyl))) +
  geom_point()
```



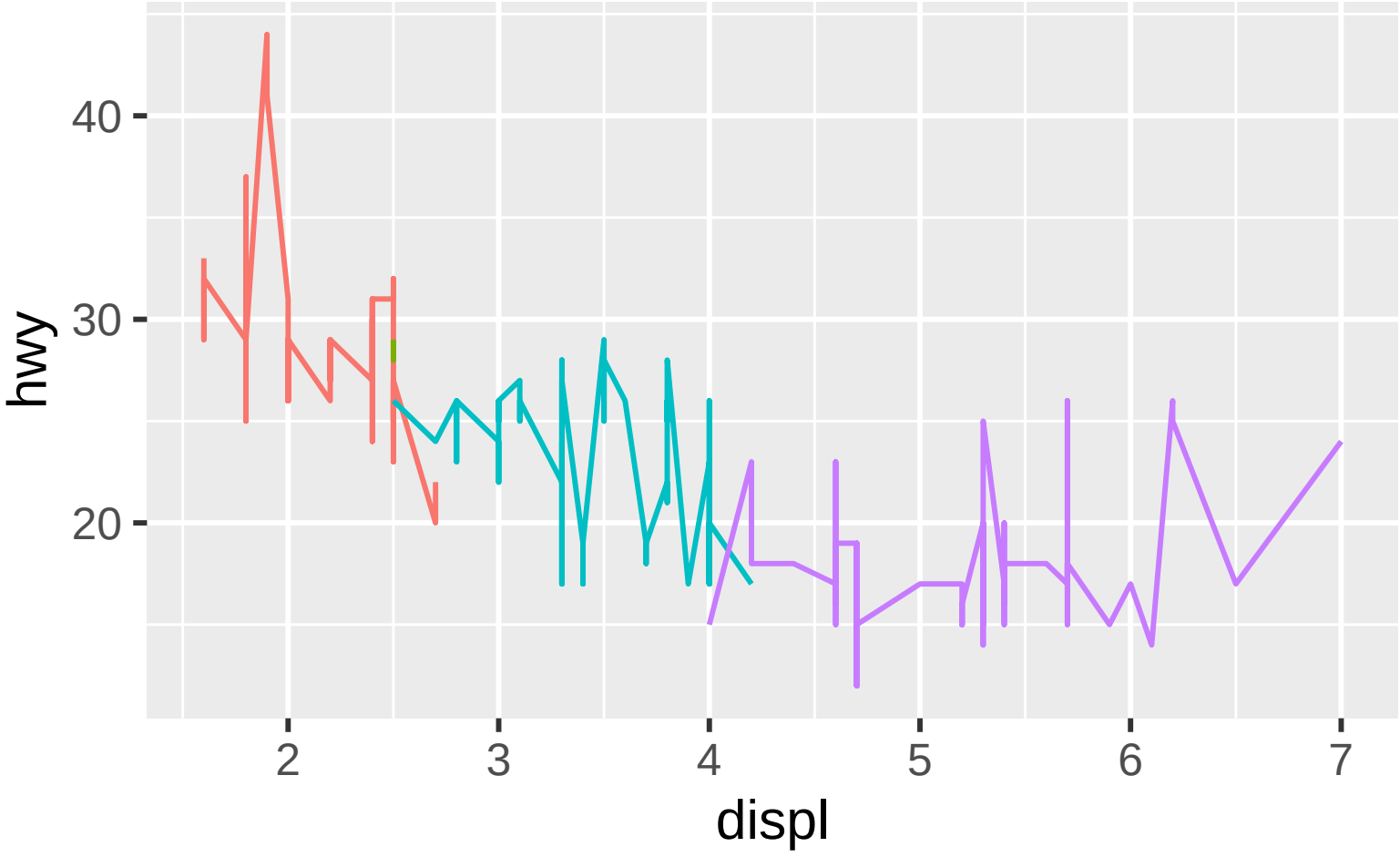
Mapping To Aesthetics: `mpg` data

x	y	color
1.8	29	4
1.8	29	4
2.0	31	4
2.0	30	4
2.8	26	6
2.8	26	6
3.1	27	6
1.8	26	4

- Aesthetic can be mapped to a variable, or set to a constant value.
- May not be sensible but grammatically valid.

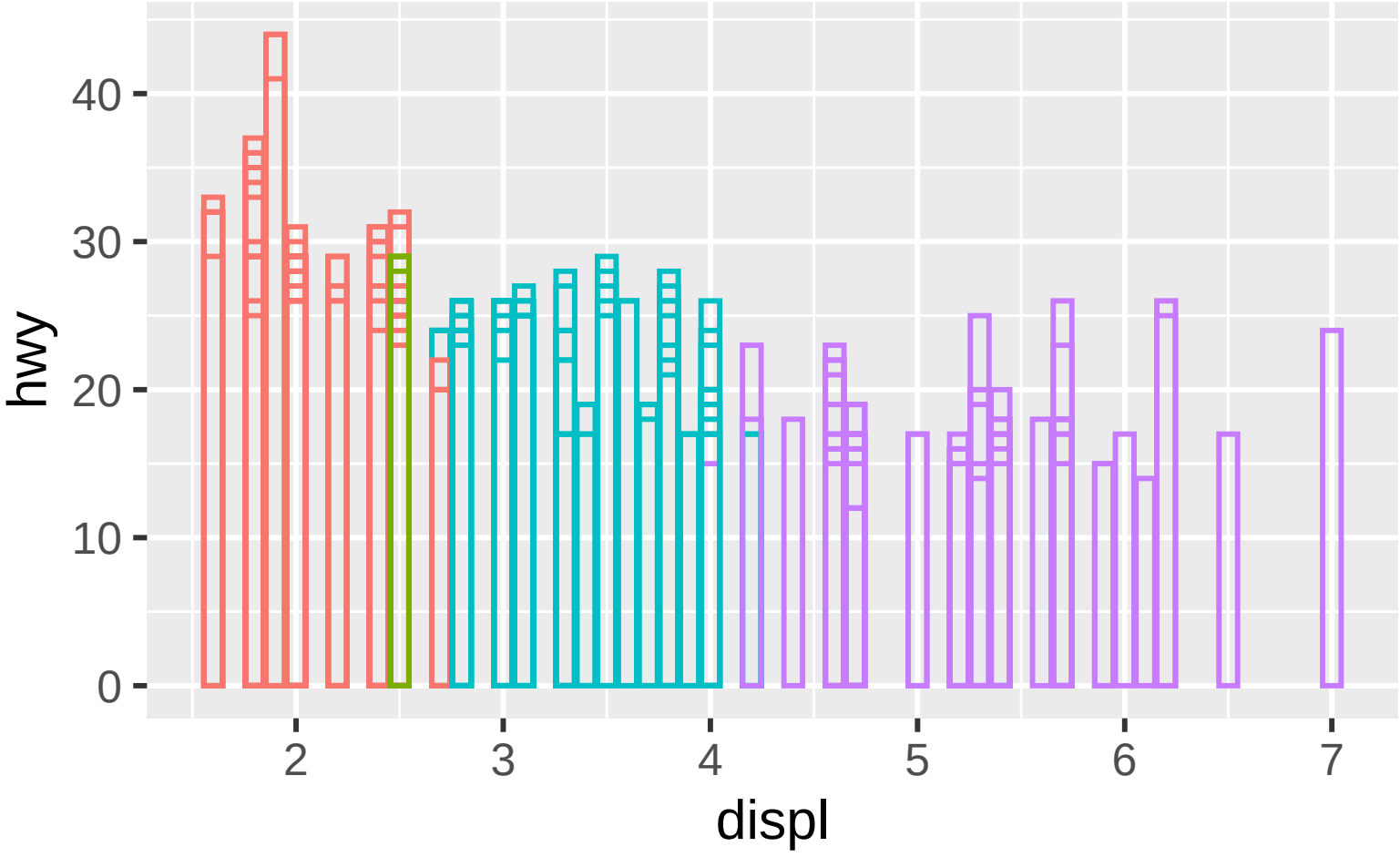
Sensible but grammatically valid (1)

```
ggplot(mpg, aes(displ, hwy, colour = factor(cyl))) +
geom_line() +
theme(legend.position = "none")
```



Sensible but grammatically valid (2)

```
ggplot(mpg, aes(displ, hwy, colour = factor(cyl))) +
geom_bar(stat = "identity", position = "identity", fill = NA) +
theme(legend.position = "none")
```



Scaling

- The values in the previous table have no meaning to the computer.
- We need to convert them from data units to graphical units (such as pixels and colours) that the computer can understand.
- This conversion process is called scaling.

Scaling: Simple Example

- Position: To convert from a continuous data value to a horizontal pixel coordinate, we need a function like:

$$\text{floor}\left(\frac{x - \min(x)}{\text{range}(x)} \times \text{screen width}\right)$$

- Color: map "a" to a circle, and "b" to a square
- This yields

x	y	Shape
25	11	circle
0	0	circle
75	53	square
200	300	square

Scaling: mpg data

- In mpg data, we have three aesthetics to be scaled
  - horizontal position (x) and vertical position (y)
  - : Only a linear mapping from the range of the data to $[0, 1]$.
  - color: more complicated

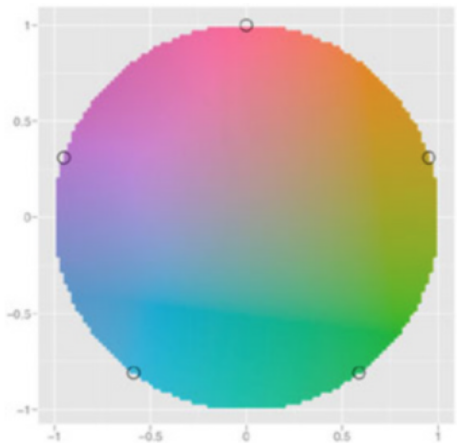


Figure 1: A color wheel illustrating the choice of five equally spaced colors. This is the default scale for discrete variables.

Scaling: `mpg` data

x	y	colour	size	shape
0.037	0.531	#F8766D	1	19
0.037	0.531	#F8766D	1	19
0.074	0.594	#F8766D	1	19
0.074	0.562	#F8766D	1	19
0.222	0.438	#00BFC4	1	19
0.222	0.438	#00BFC4	1	19
0.278	0.469	#00BFC4	1	19
0.037	0.438	#F8766D	1	19

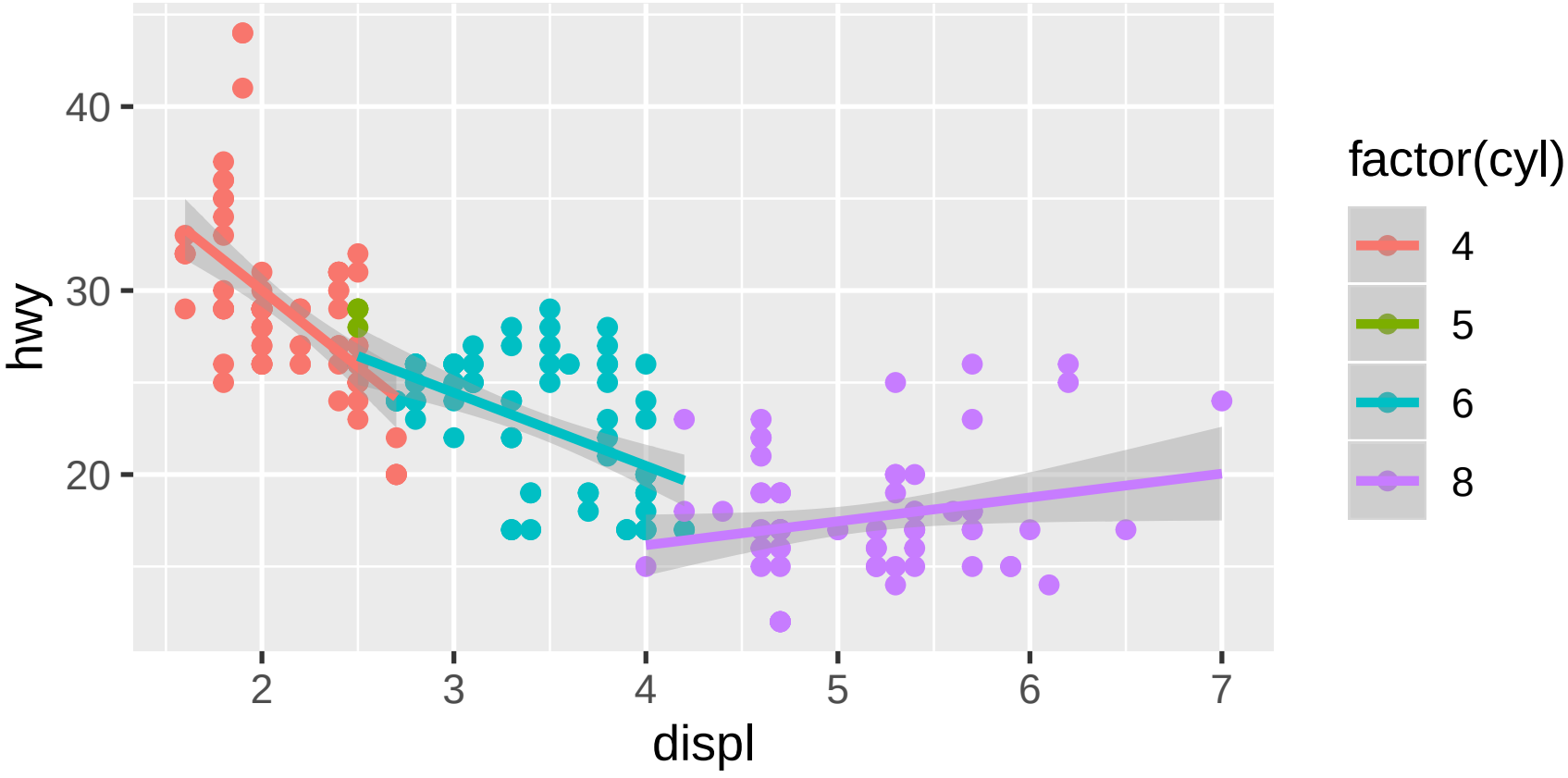
Table 4: Illustration of the data after Scale.

Statistical Transformation

- In general, there is another step that we have skipped in this simple example: statistical transformation or stats.
- Here we are using the identity transformation, but there are many others that are useful, such as binning or aggregating.

Statistical Transforatm: mpg data

```
ggplot(mpg, aes(displ, hwy, colour = factor(cyl))) +
geom_point() +
geom_smooth(method = "lm")
```



Rendering

- Render this data to create the graphical objects on the screen.
- Need to combine graphical objects from three sources:
  - data, represented by the point `geom`.
  - scales and coordinate system, which generate axes and legends.
  - plot annotations, such as the background and plot title.

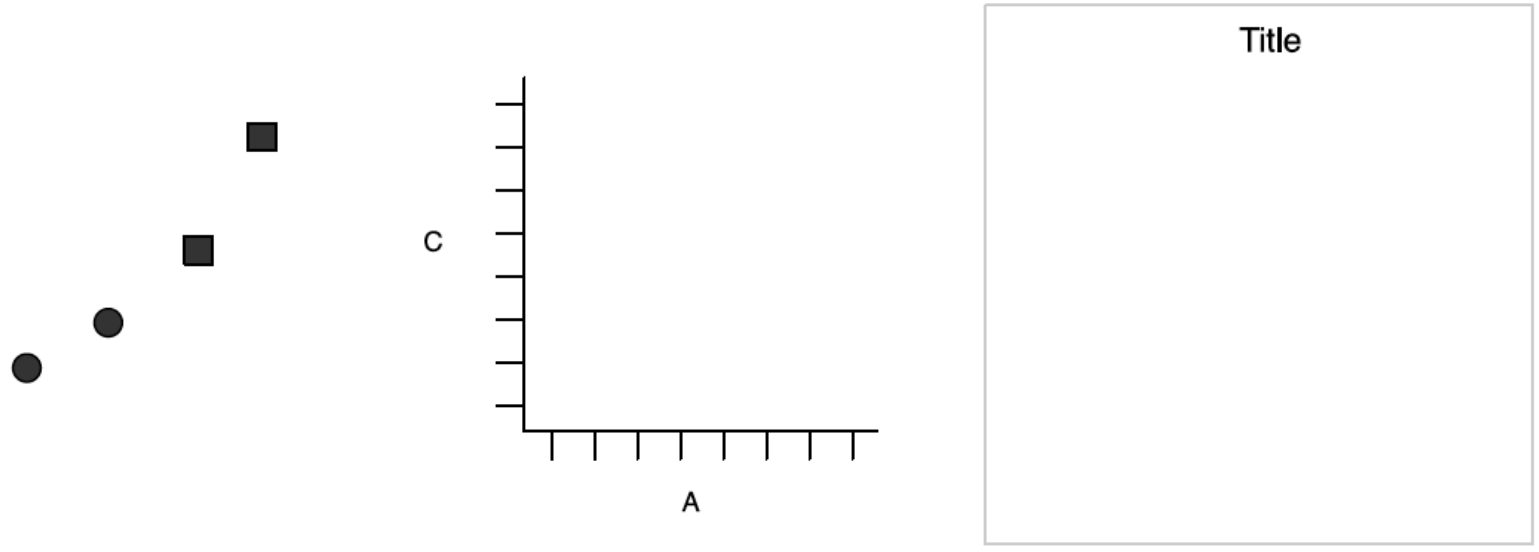


Figure 2: Graphics objects produced by geometric objects, scales and coordinate system, plot annotations.

Rendering

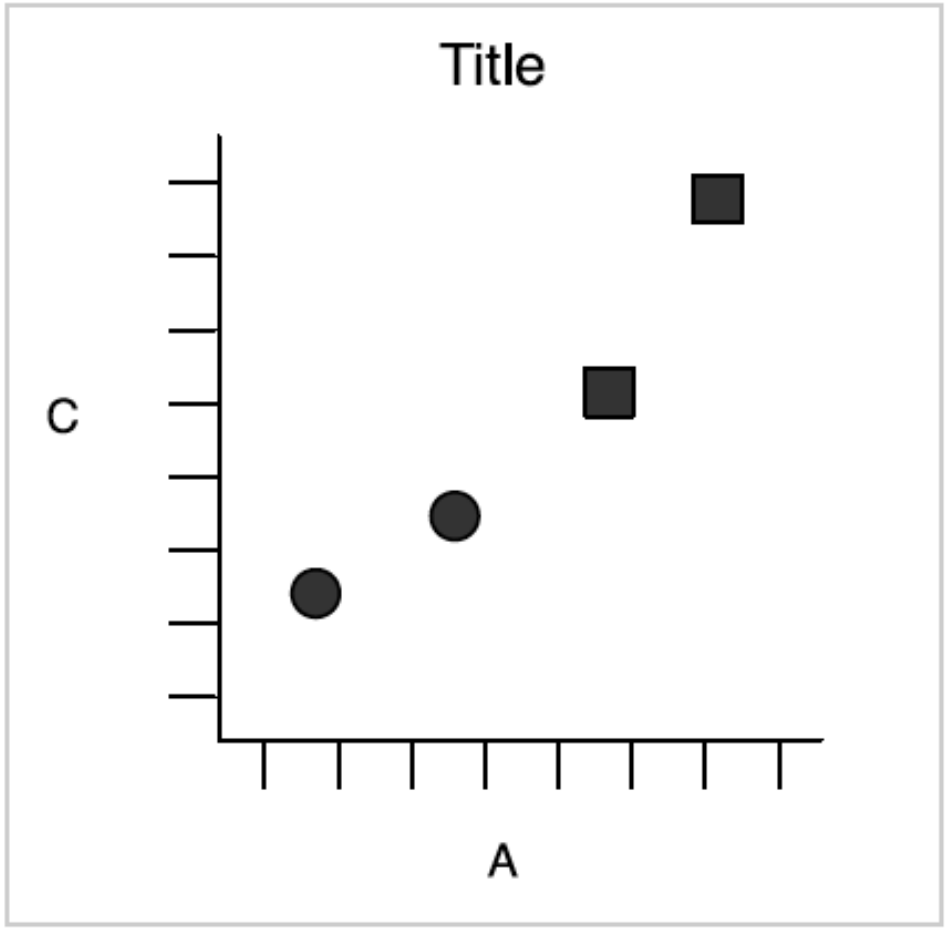


Figure 3: The final graphic, produced by combining the pieces in Figure 2.

Adding Complexity: Facet

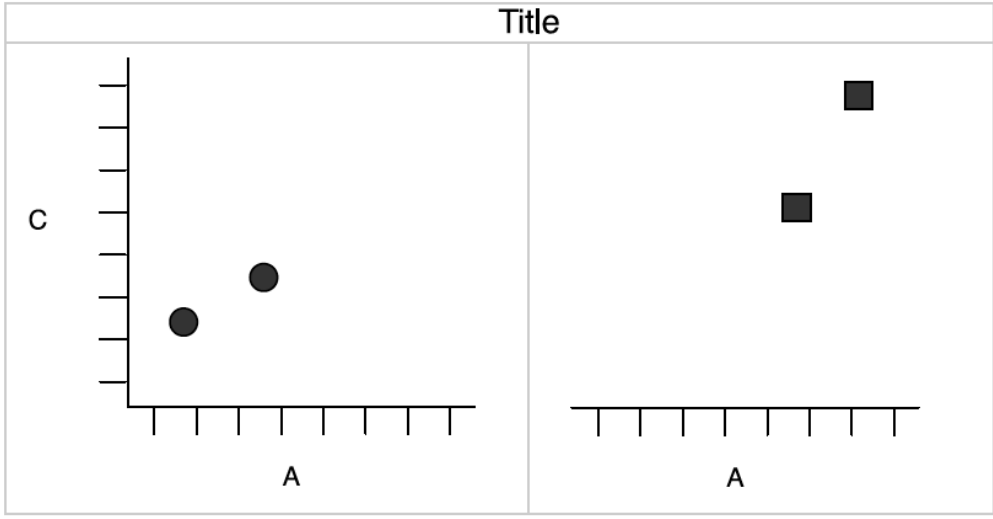
- Can create a more complicated plot using faceting.
- Faceting is a more general case of the techniques known as conditioning, trellising, and latticing, and produces small multiples showing different subsets of the data.

Adding Complexity: Facet

- If we facet the previous plot by D, we have

	x	y	Shape
a	25	11	circle
a	0	0	circle
b	75	53	square
b	200	300	square

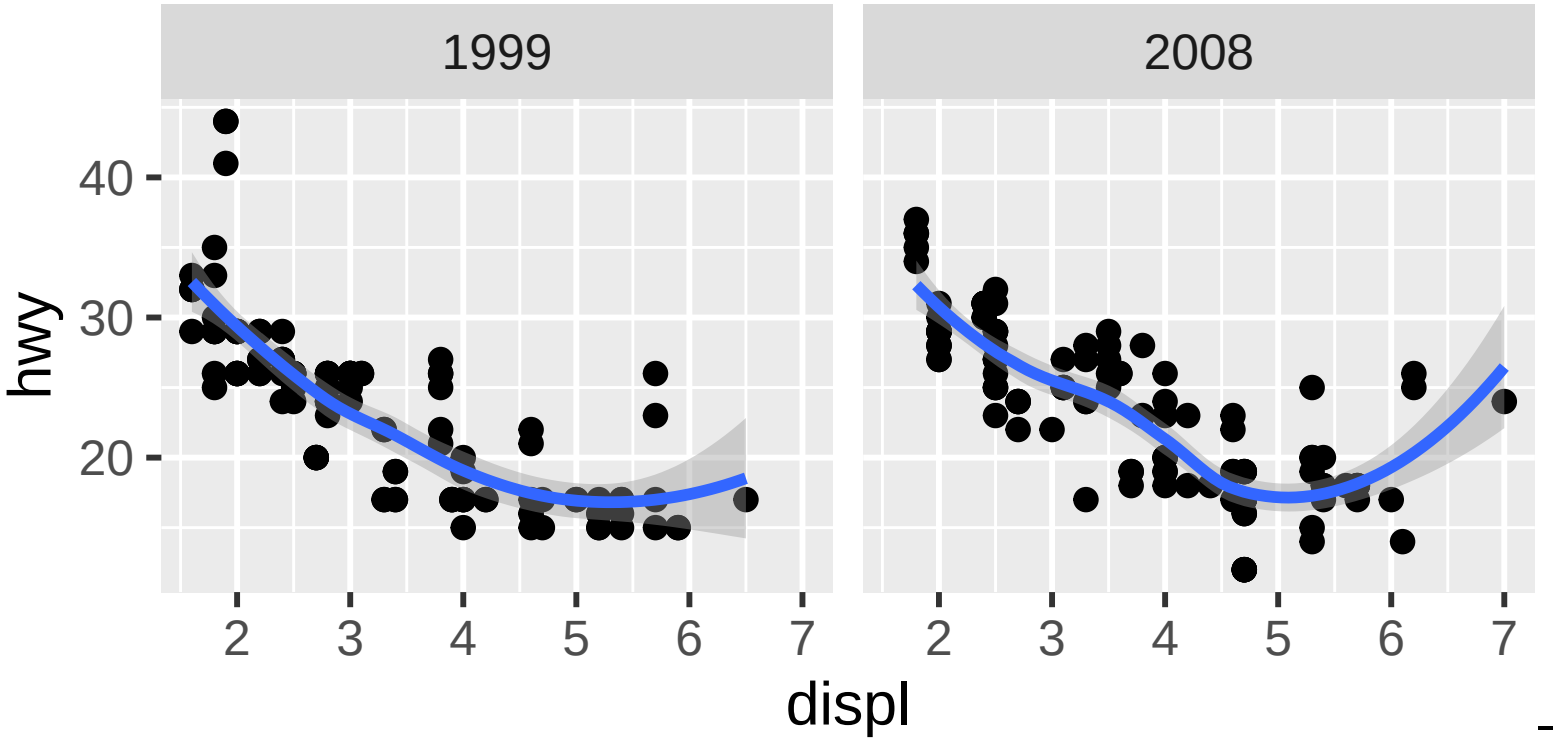
- This yields



Adding Complexity: mpg data

```
ggplot(mpg, aes(displ, hwy)) +
    geom_point() +
    geom_smooth() +
    facet_wrap(~year)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

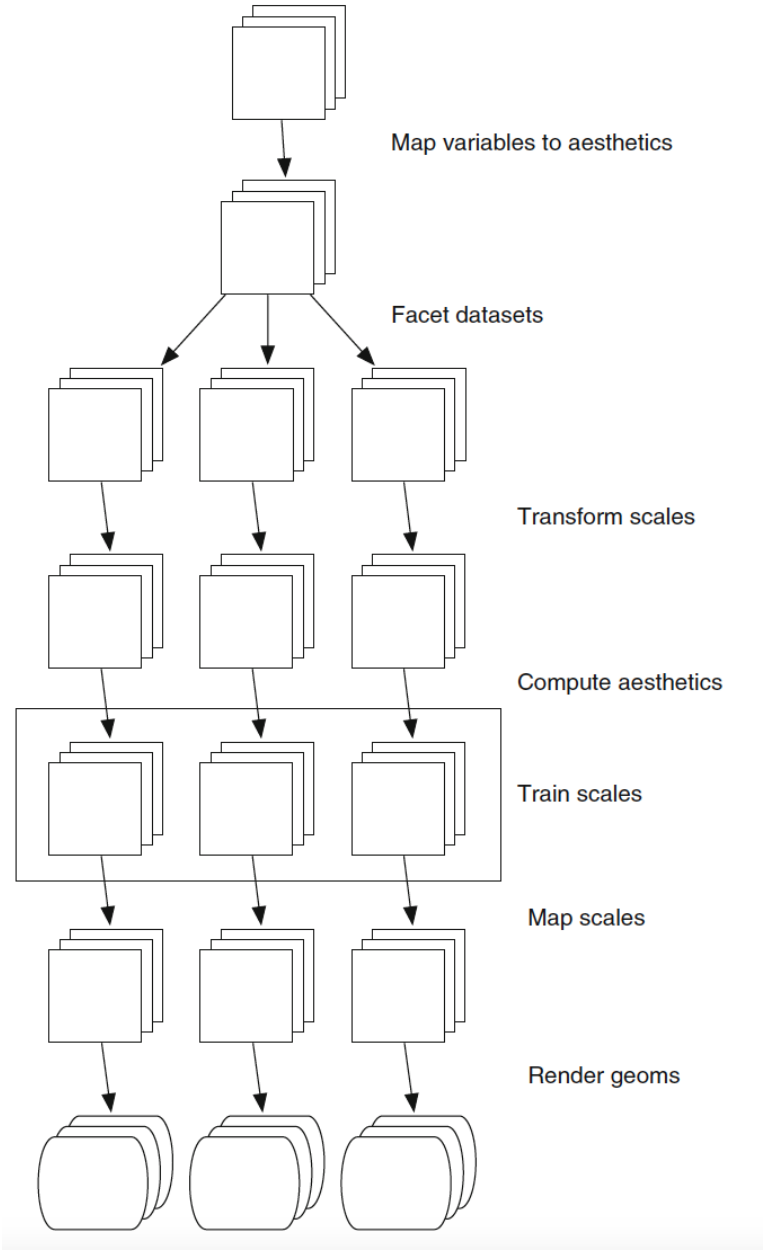


Added new components to the mix facets, multiple layers and statistics.

Scaling in More Complicated Plot

- need some extra steps when we get to the scales, because we have multiple datasets (for the different facets and layers).
- need to make the scales be the same across all of them.
- Scaling actually occurs in three parts:
  - transforming: scale transformation occurs before statistical transformation so that statistics are computed on the scale-transformed data.
  - training: after the statistics are computed, each scale is trained on every dataset from all the layers and facets. Local scale many not bemeaningful for the comparison.
  - mapping: the scales map the data values into aesthetic values

Schematic description of the plot generation process



Components of Layered Grammar

- Components that make up a plot:
  - data and aesthetic mappings,
  - geometric objects,
  - scales, and
  - facet specification.
- two other components:
  - statistical transformations, and
  - the coordinate system.
- Together, the data, mappings, statistical transformation, and geometric object form a layer.

Components of Layered Grammar

- A plot may have multiple layers.
- The layered grammar defines the components of a plot as:
  - A default dataset and set of mappings from variables to aesthetics
  - one or more layers, with each layer having one geometric object, one statistical transformation, one position adjustment, and optionally, one dataset and set of aesthetic mappings,
  - one scale for each aesthetic mapping used,
  - coordinate system,
  - facet specification.

Components of Layered Grammar

- The layer component determines the physical representation of the data, with the combination of `stat` and `geom` defining many familiar graphics such as the scatterplot, histogram, ...
- Many plots have (at least) three layers: the data, contextfor the data, and a statistical summary of the data.

Why Grammar Useful?

For Users

- easier to iteratively update a plot, changing a single feature at a time.
- suggests the high-level aspects of a plot that can be changed, giving us a framework to think about graphics, and hopefully shortening the distance from mind to paper.
- encourages the use of graphics customized to a particular problem rather than relying on generic named graphics.

For Developers

- easier to create new capabilities: only need to add the one component that you need, and can continue to use all the other existing components
- useful for discovering new types of graphics, as the grammar defines the parameter space of statistical graphics.

Layers

A layer is composed

- data and aesthetic mapping
- a statistical transformation (stat),
- a geometric object (geom), and
- a position adjustment

Data and Mapping

- Data are a critical part of the plot and independent from the other components.
- Along with the data, we need a specification of which variables are mapped to which aesthetics. eg. map weight to x position, height to y position, and age to size. (The details of the mapping are described by the scales)

Statistical Transformation

- A statistical transformation, or stat, transforms the data, typically by summarizing them in some manner.
- To make sense in a graphical context a stat must be location-scale invariant:

$$f(x + a) = f(x) + a \text{ and } f(b \cdot x) = b \cdot f(x).$$

Name	Description
`bin`	Divide continuous range into bins, and count number of points in each
`boxplot`	Compute statistics necessary for boxplot
`contour`	Calculate contour lines
`density`	Compute 1d density estimate
`identity`	Identity transformation, $f(x) = x$
`jitter`	Jitter values by adding small random value
`qq`	Calculate values for quantile-quantile plot
`quantile`	Quantile regression
`smooth`	Smoothed conditional mean of y given x
`summary`	Aggregate values of y for given x
`unique`	Remove duplicated observations

Statistical Transformation

- A stat takes a dataset as input and returns a dataset as output.
- A stat can add new variables to the original dataset.
- Possible to map aesthetics to these new variables.
- The statistical method used by a stat should be conditional on the coordinate system, so we typically use Cartesian coordinates for calculation.

Geometric Object (Graphical primitives)

- Geometric objects, or **geoms** for short, perform the actual rendering of the layer, controlling the type of plot that you create.
  - `geom_blank()`: display nothing.
  - `geom_point()`: points.
  - `geom_path()`: paths.
  - `geom_ribbon()`: ribbons, a path with vertical thickness.
  - `geom_segment()`: a line segment, specified by start and end position.
  - `geom_rect()`: rectangles.
  - `geom_polyon()`: filled polygons.
  - `geom_text()`: text.

Geometric Object (One variable)

- **Discrete**
  - `geom_bar()`: display distribution of discrete variable.
- **Continuous**
  - `geom_histogram()`: bin & count continuous variable, display with bars.
  - `geom_density()`: smoothed density estimate.
  - `geom_dotplot()`: stack individual points into a dot plot.
  - `geom_ freqpoly()`: bin & count continuous variable, display with lines.

Geometric Object (Two variables)

- **Both continuous**
  - `geom_point()` scatterplot.
  - `geom_quantile()`: smoothed quantile regression.
  - `geom_rug()`: marginal rug plots.
  - `geom_smooth()`: smoothed line of best fit.
  - `geom_text()`: text labels.
- **Show distribution**
  - `geom_bin2d()`: bin into rectangles and count.
  - `geom_density2d()`: smoothed 2d density estimate.
  - `geom_hex()`: bin into hexagons and count.
- **At least one discrete**
  - `geom_count()`: count number of point at distinct locations
  - `geom_jitter()`: randomly jitter overlapping points.

Geometric Object (Two variables)

- **One continuous, one discrete**
  - `geom_bar(stat = "identity")`: a bar chart of precomputed summaries.
  - `geom_boxplot()`: boxplots.
  - `geom_violin()`: show density of values in each group.
- One time, one continuous
  - `geom_area()`: area plot.
  - `geom_line()`: line plot.
  - `geom_step()`: step plot.
- **Display uncertainty**
  - `geom_crossbar()`: vertical bar with center.
  - `geom_errorbar()`: error bars.
  - `geom_linerange()`: vertical line.
  - `geom_pointrange()`: vertical line with center.

Geometric Object (Three variables)

- `geom_contour()`: contours.
- `geom_tile()`: tile the plane with rectangles.
- `geom_raster()`: fast version of `geom_tile()` for equal sized tiles.

Stats

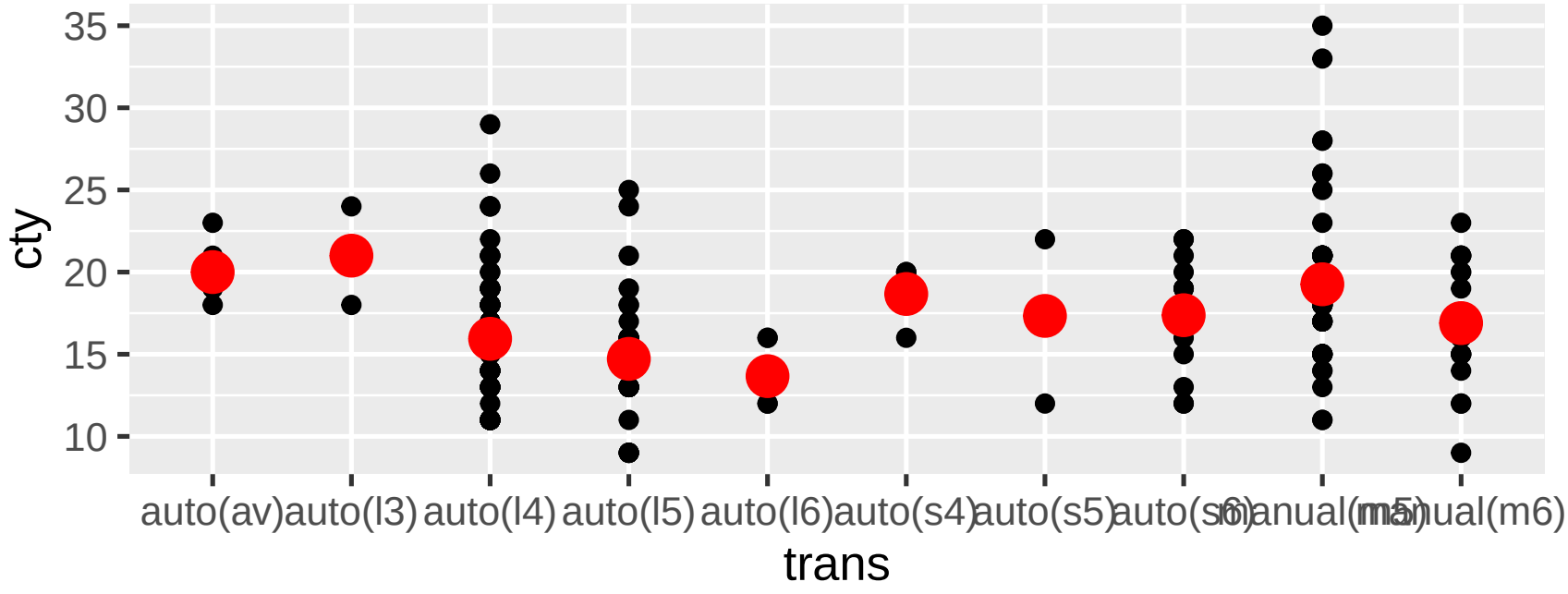
- A statistical transformation, or stat, transforms the data, typically by summarising it in some manner.
- Many of ggplot2's stats because they're used behind the scenes to generate many important geoms:
  - `stat_bin()`: `geom_bar()`, `geom_freqpoly()`, `geom_histogram()`
  - `stat_bin2d()`: `geom_bin2d()`
  - `stat_bindot()`: `geom_dotplot()`
  - `stat_binhex()`: `geom_hex()`
  - `stat_boxplot()`: `geom_boxplot()`
  - `stat_contour()`: `geom_contour()`
  - `stat_quantile()`: `geom_quantile()`
  - `stat_smooth()`: `geom_smooth()`
  - `stat_sum()`: `geom_count()`

Stats

- Other stats can't be created with a `geom_` function:
  - `stat_ecdf()`: compute a empirical cumulative distribution plot.
  - `stat_function()`: compute y values from a function of x values.
  - `stat_summary()`: summarise y values at distinct x values.
  - `stat_summary2d()`, `stat_summary_hex()`: summarise binned values.
  - `stat_qq()`: perform calculations for a quantile-quantile plot.
  - `stat_spoke()`: convert angle and radius to position.
  - `stat_unique()`: remove duplicated rows.

Stats: Illustration

```
ggplot(mpg, aes(trans, cty)) +
  geom_point() +
  stat_summary(geom = "point", fun.y = "mean",
               colour = "red", size = 4)
```



Position Adjustments

- To Bars
  - `position_stack()`: stack overlapping bars (or areas) on top of each other.
  - `position_fill()`: stack overlapping bars, scaling so the top is always at 1.
  - `position_dodge()`: place overlapping bars (or boxplots) side-by-side.
- To Points
  - `position_nudge()`: move points by a fixed offset.
  - `position_jitter()`: add a little random noise to every position.
  - `position_jitterdodge()`: dodge points within groups, then add a little random noise.

Scales

- A scale controls the mapping from data to aesthetic attributes, and we need a scale for every aesthetic used on a plot.
- Each scale operates across all the data in the plot, ensuring a consistent mapping from data to aesthetics. (See Figure 4.)

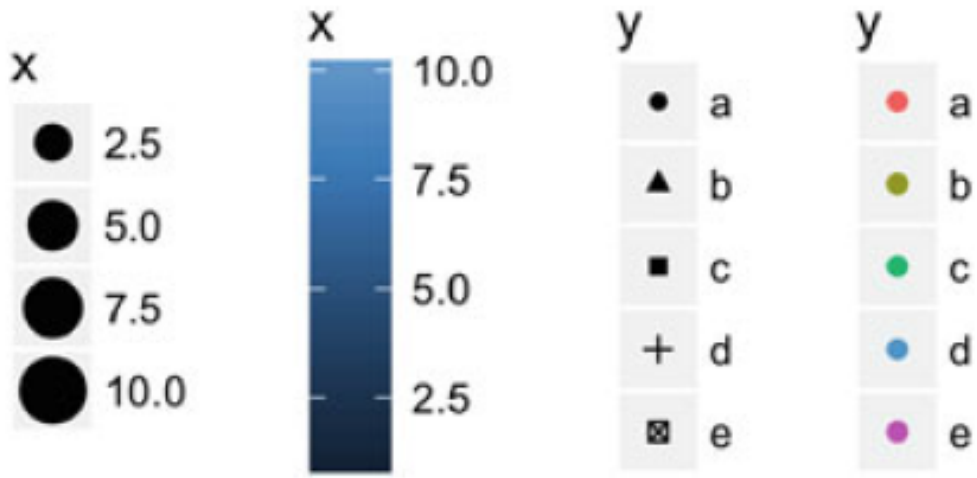


Figure 4: Examples of legends from four different scales.

- Scale is often an invertable function, and the inverse function is used to draw a guide so that you can read values from the graph. (ex. axis or legend)
-

Coordinate System

- A coordinate system, or **coord** for short, maps the position of objects onto the plane of the plot.
- Coordinate systems control how the axes and grid lines are drawn. (See Figure 5.)

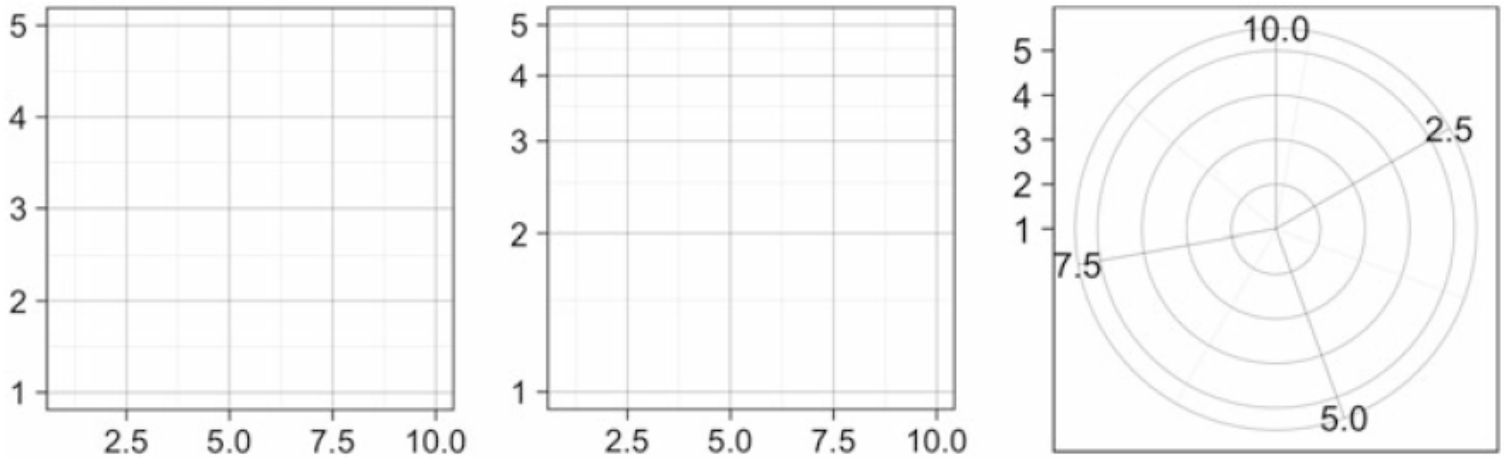


Figure 5: Examples of axes and grid lines for three coordinate systems: Cartesian, semilog and polar.

Facetting

- Facetting is a general case of conditioned or trellised plots and makes it easy to create small multiples, each showing a different subset of the whole dataset.
- Powerful when investigating whether patterns hold across all conditions.
- The facetting specification describes which variables should be used to split up the data, and whether position scales should be free or constrained.

Reference

- Wilkinson, L. (2005) The grammar of graphics. Statistics and computing, 2nd Edition. Springer, New York.
- Wickham, H. (2010) A layered gammar of graphics, JCGS, 19(1), 3-28.
- Wickham, H. (2016) ggplot2: elegant graphics for data analysis, 2nd Edition, Springer, New York. Part II.