Data Visualization in R

Matt Arthur

09 Dec 2020

Agenda

1. Introduction to `ggplot2`
2. Plotting Scenarios
   - One Continuous Variable
   - One Categorical Variable
   - Two Continuous Variables
   - One Categorical, and One Continuous Variable
3. Plot Options
   - faceting
4. Axes, Legends, and Themes
5. Miscellaneous

Why Visualize Data in R with ggplot?

Three main reasons:

1. Quality
   - ggplot is widely used and has a good reputation
2. Ease
   - Can make complex graphs relatively quickly
   - Can correct mistakes easily
   - Syntax is logical
3. Cohesion
   - Easy to keep data visualization consistent with data manipulation and analyses

About ggplot2

ggplot2 is an R package used for data visualization.

- Created by Hadley Wickham (and others)
- Part of tidyverse
- Based on the Grammar of Graphics (Wilkinson, 2005)

Installation

```
install.packages("ggplot2")
```

Preparing our Environment

```
library(ggplot2)
data(mpg)
```

`mpg` is a built-in R dataset which includes fuel economy data from 1999 and 2008 for 38 popular models of car.

The Grammar of Graphics

All plots are composed of:

- **Aesthetic Mappings** which describe how variables in the data are mapped to aesthetic attributes.
- **Layers** made up of
  - Geometric objects, or *geoms* for short: Points, Lines, etc.
  - Statistical transformations: For example, binning/counting observations to create a histogram, or summarizing 2D data with a linear model.
- **Scales** which
  - Map values in the data space to values in the aesthetic space (i.e. color)
  - Draw a legend or axes that make it possible to read original data values from the plot

The Grammar of Graphics

All plots are composed of:

- A coordinate system, of *coord* for short, which
  - Describes how data coordinates are mapped to the plane of the graphics (i.e. cartesian versus polar coordinates)
  - Provides axes and gridlines to make it easier to read the graph.
- A **theme** which controls the finer points of the display (i.e. background color, font size, etc)
- Optionally, a faceting specification, describing how to break the data into subsets and displays those subsets

Three Essential Elements of a Plot

You will always need to specify the following:

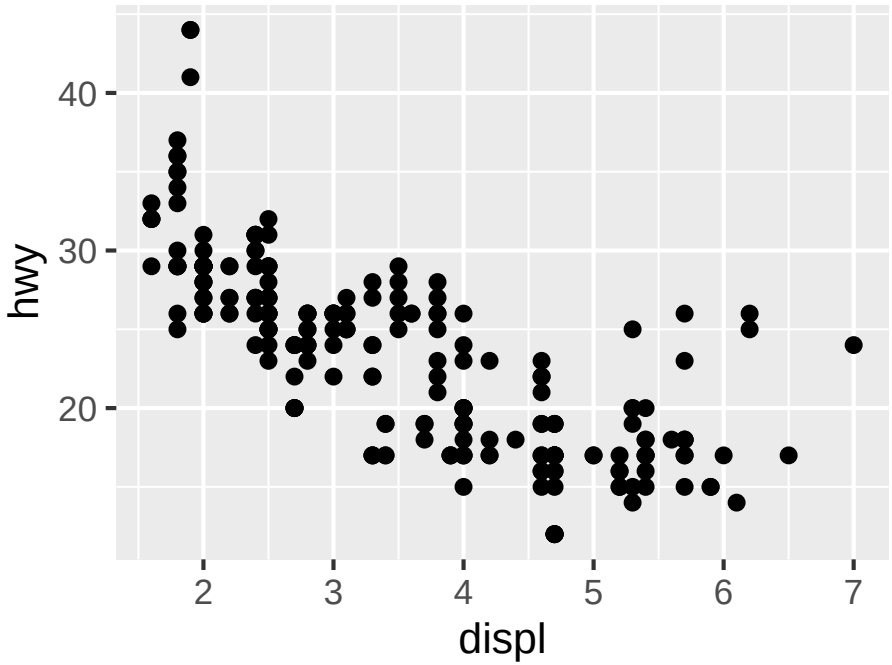
- Data to plot
- Aesthetic mappings
- At least one layer, with a `geom()` function

Data and aesthetic mappings are supplied using the `ggplot()` function.

Then, additional layers are added to the plot with +

Three Essential Elements of a Plot

```
ggplot(mpg, aes(displ, hwy)) + geom_point()
```



Almost every plot maps a variable to x and y, so the first two unnamed arguments of `aes()` will be mapped to the x and y axes, respectively.

Layers

Recall that **Layers** are made up of

- Geometric objects, or **geoms** for short (i.e. points or lines)
- Statistical transformations, or **stats** for short

Each geom has a set of aesthetic and stats that it understands.

Popular Geoms

- `geom_point()`
- `geom_line()`

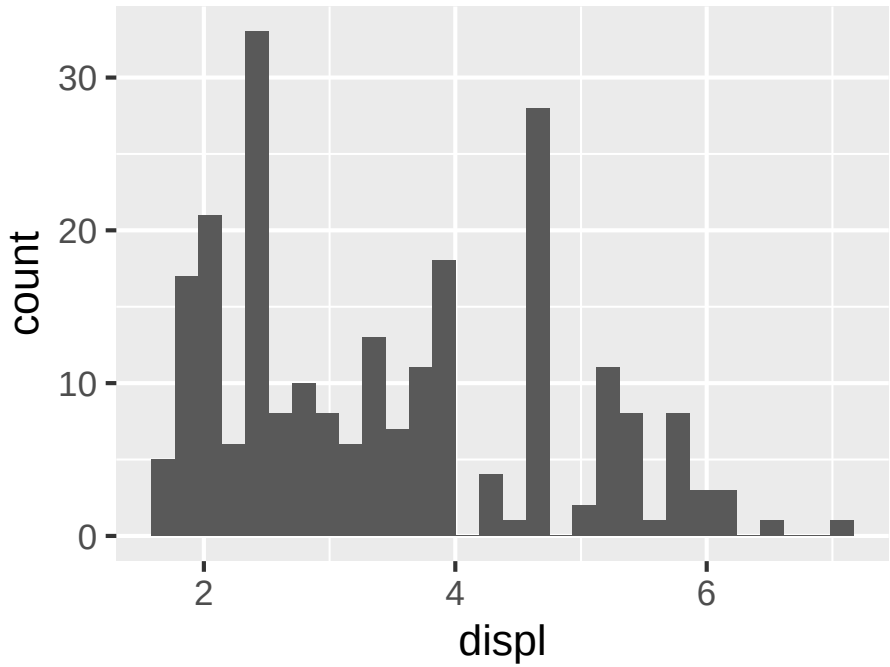
Plan

1. Introduction to ggplot2
2. Plotting Scenarios
   - One Continuous Variable
   - One Categorical Variable
   - Two Continuous Variables
   - One Categorical, and One Continuous Variable
3. Plot Options
   - faceting
4. Axes, Legends, and Themes
5. Miscellaneous

Histograms

```
ggplot(mpg, aes(displ)) + geom_histogram()
```

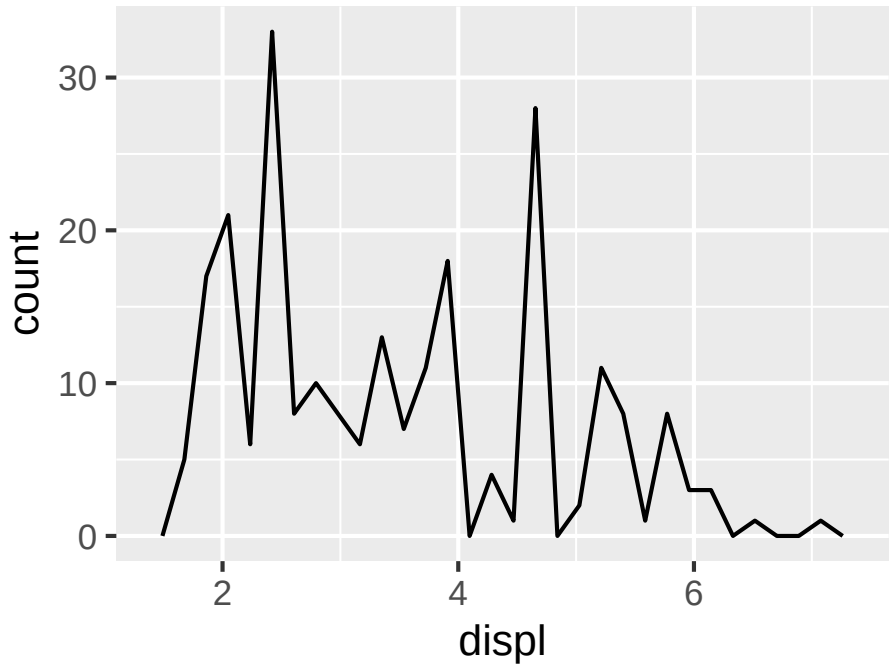
```
## `stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
```



Frequency Polygon

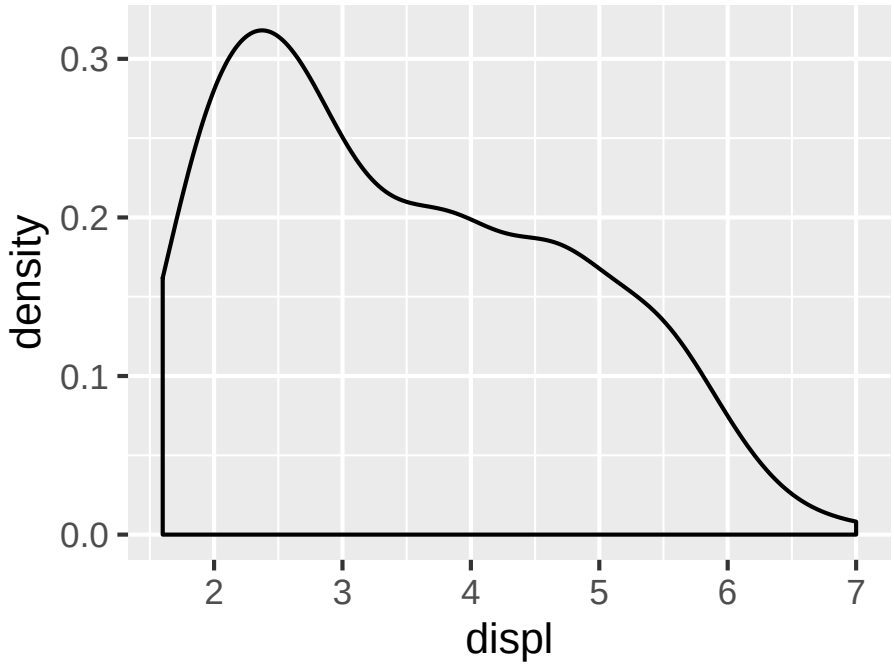
```
ggplot(mpg, aes(displ)) + geom_freqpoly()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with
`binwidth`.
```



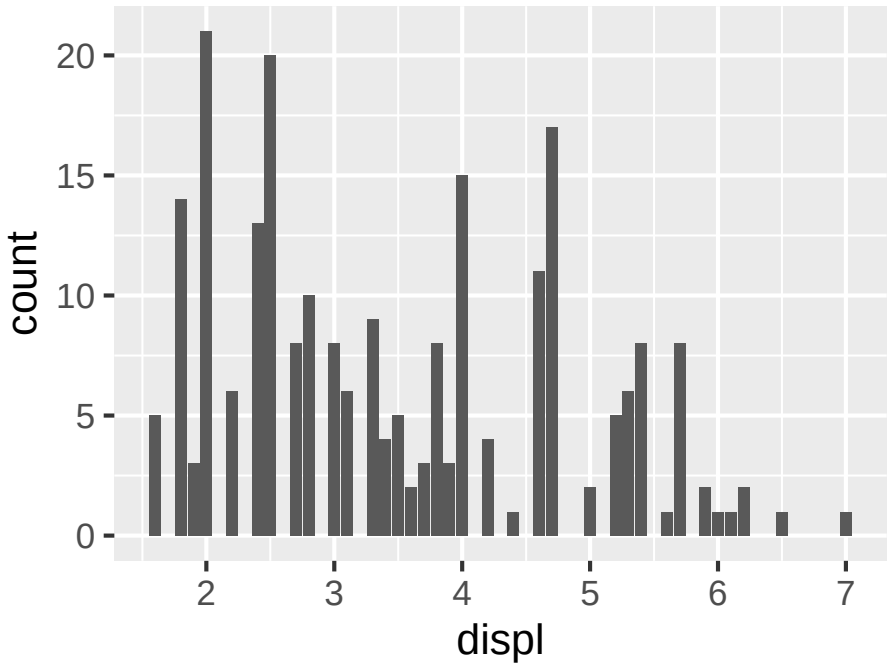
Smoothed Density Estimate

```
ggplot(mpg, aes(displ)) + geom_density()
```



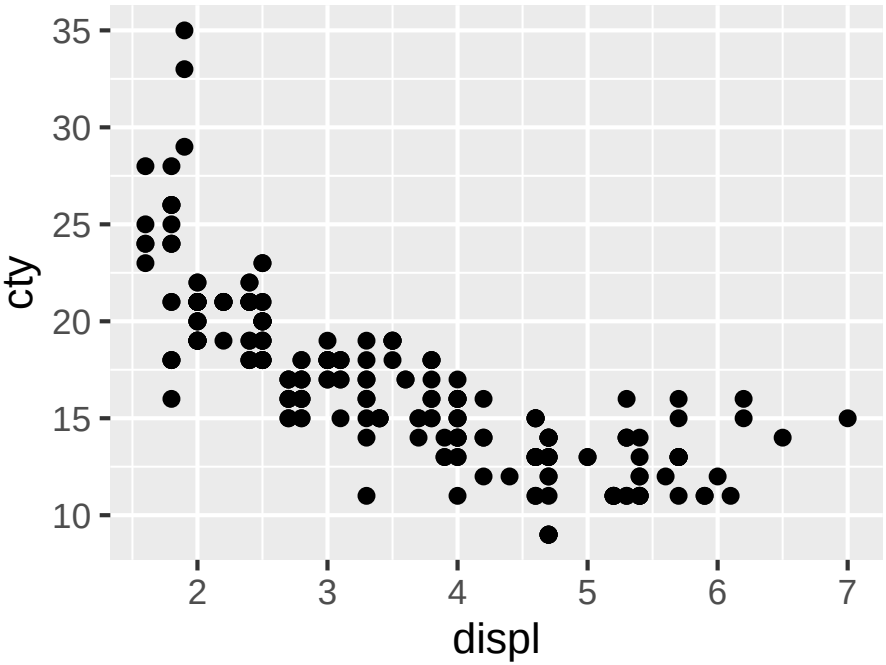
Bar Graph

```
ggplot(mpg, aes(displ)) + geom_bar()
```



Scatterplot

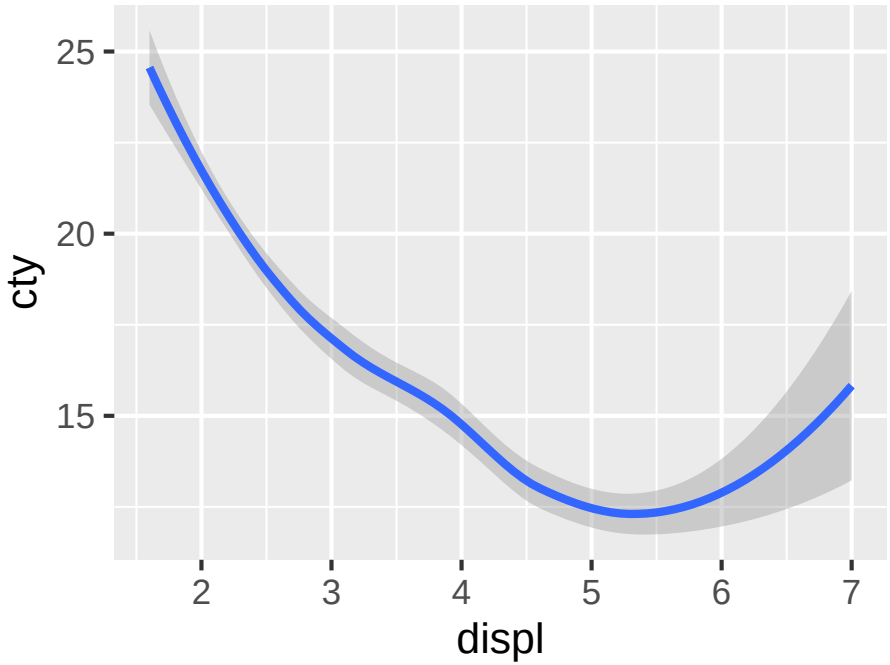
```
ggplot(mpg, aes(displ, cty)) + geom_point()
```



Scatterplot

```
ggplot(mpg, aes(displ, cty)) + geom_smooth()
```

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




```
# Uses geom_smooth(method='loess', formula=y~x) as default
```

Economics Data

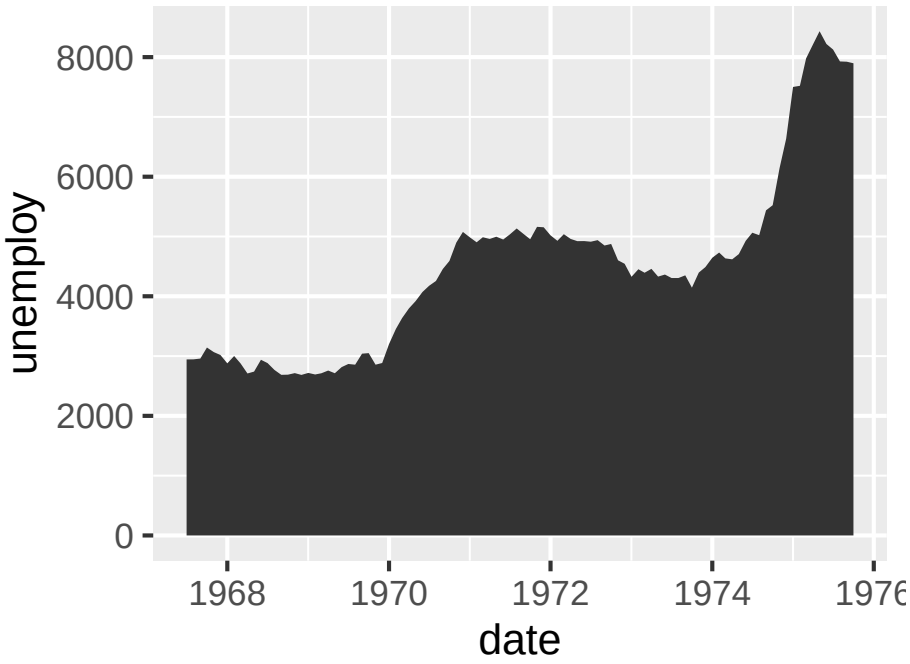
The `economics` dataset is included with the `ggplot2` library. It includes 478 observations on 6 variables:

```
  head(economics)
```

```
## # A tibble: 6 x 6
##   date         pce    pop psavert uempmed unemploy
##   <date>     <dbl>  <dbl>   <dbl>   <dbl>    <dbl>
## 1 1967-07-01  507. 198712    12.6     4.5     2944
## 2 1967-08-01  510. 198911    12.6     4.7     2945
## 3 1967-09-01  516. 199113    11.9     4.6     2958
## 4 1967-10-01  512. 199311    12.9     4.9     3143
## 5 1967-11-01  517. 199498    12.8     4.7     3066
## 6 1967-12-01  525. 199657    11.8     4.8     3018
```

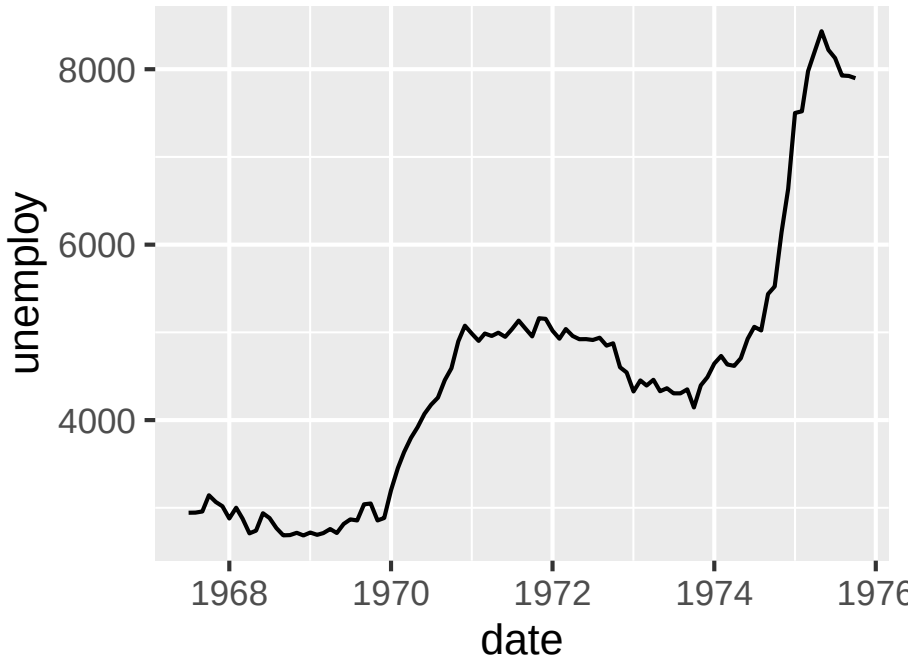
Scatterplot

```
ggplot(economics[1:100,], aes(date, unemploy)) +
  geom_area()
```



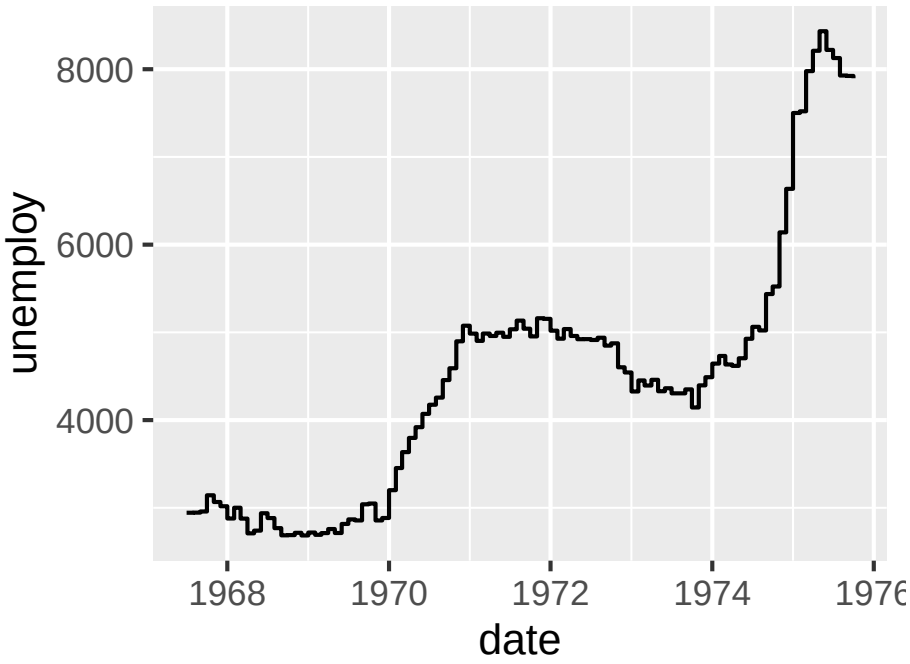
Line Plot

```
ggplot(economics[1:100,], aes(date, unemploy)) +
  geom_line()
```



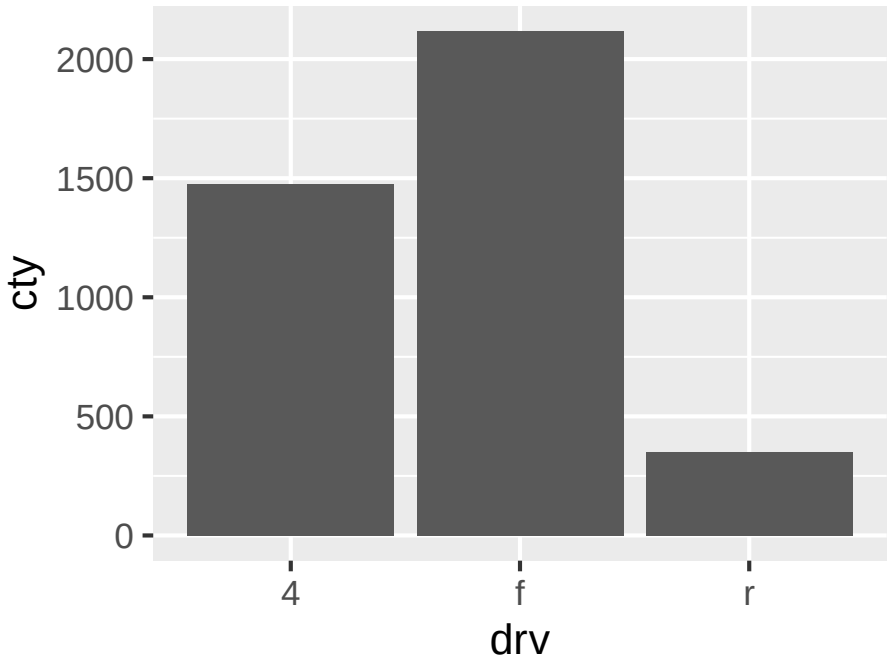
Step Plot

```
ggplot(economics[1:100,], aes(date, unemploy)) +
  geom_step()
```



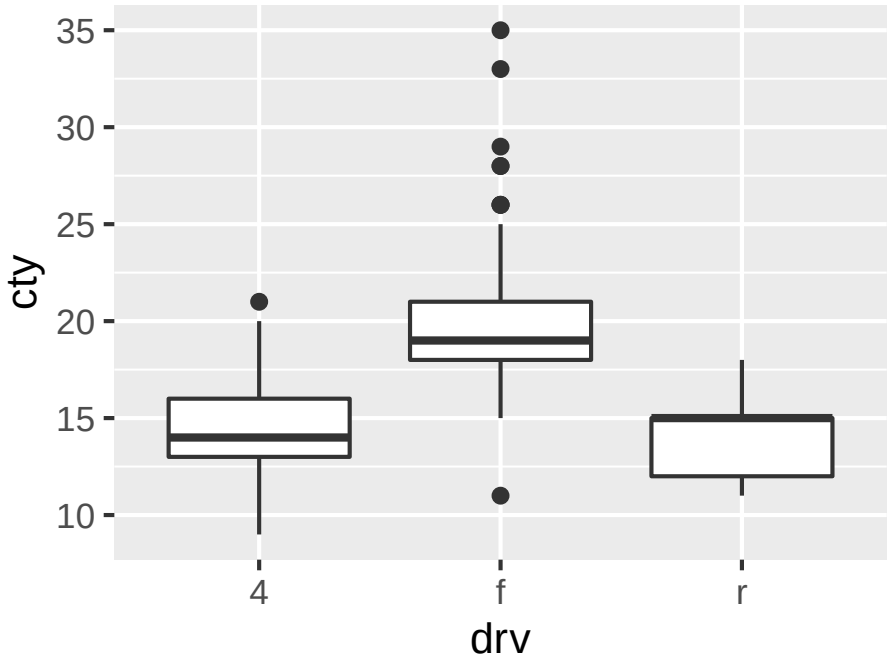
Bar Plot

```
ggplot(mpg, aes(drv, cty)) +
  geom_bar(stat="identity")
```



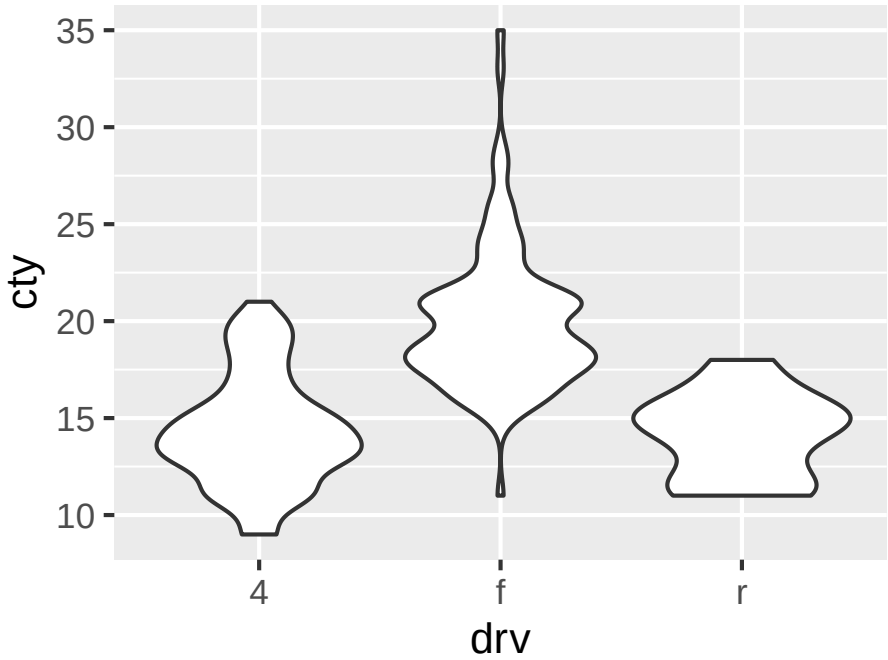
Box Plot

```
ggplot(mpg, aes(drv, cty)) +
  geom_boxplot()
```



Violin Plot

```
ggplot(mpg, aes(drv, cty)) +
  geom_violin()
```



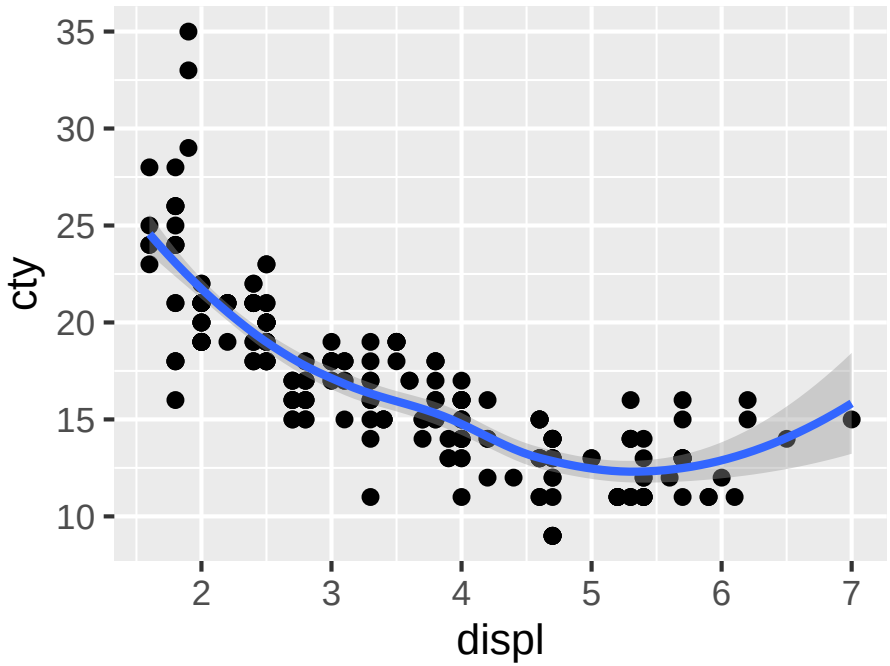
Plan

1. Introduction to `ggplot2`
2. Plotting Scenarios
   - One Continuous Variable
   - One Categorical Variable
   - Two Continuous Variables
   - One Categorical, and One Continuous Variable
3. Plot Options
   - faceting
4. Axes, Legends, and Themes
5. Miscellaneous

Adding Multiple Layers to a Plot

```
  ggplot(mpg, aes(displ, cty)) +
    geom_point() + geom_smooth()
```

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

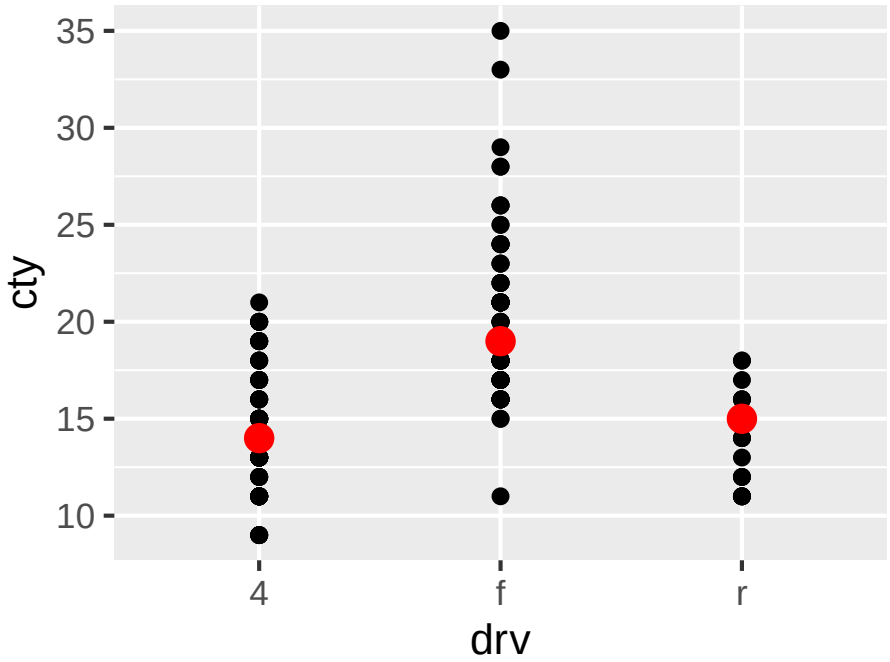


Stats

Stats can be used when you need to do a statistical transformation of the data that a `geom` can't already do.

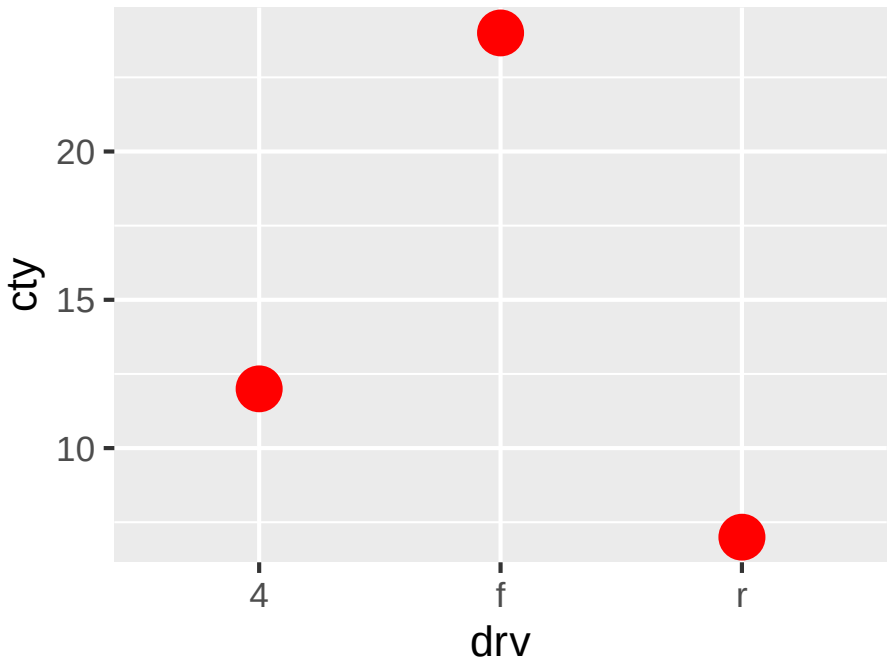
stats

```
ggplot(mpg, aes(drv, cty)) +
  geom_point() +
  stat_summary(fun.y="median",
               color="red",
               size=3, geom="point")
```



stats

```
rng <- function(x) { return( max(x) - min(x) ) }
ggplot(mpg, aes(drv, cty)) +
  stat_summary(fun.y="rng",
               color="red",
               size=5, geom="point")
```



Aesthetics

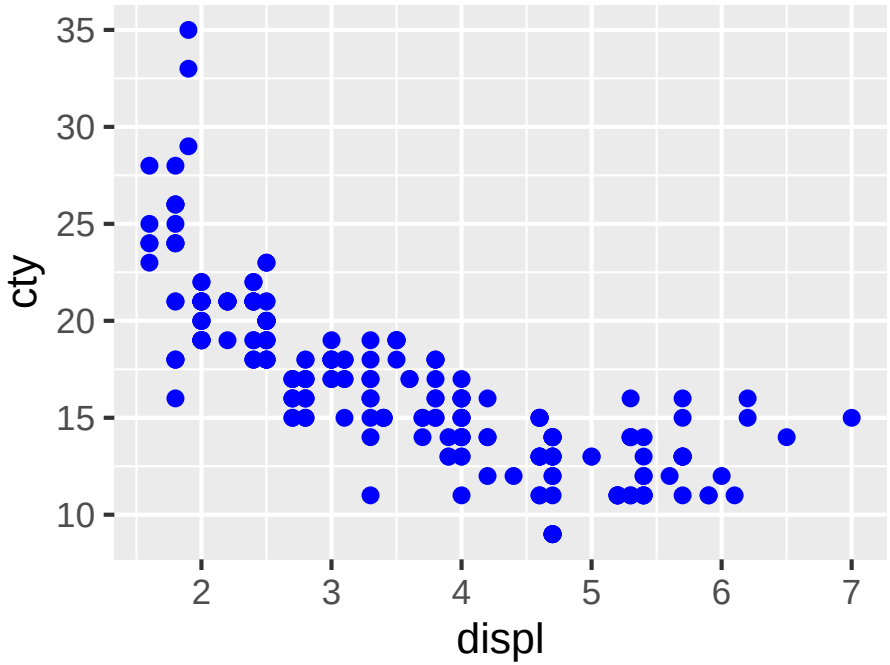
Recall: **Aesthetic mappings** describe how variables in the data are mapped to aesthetic attributes.

An aesthetic can be mapped to a variable or set to a constant.

Aesthetics

If you want appearance to be constant, put the value inside the `geom()` function:

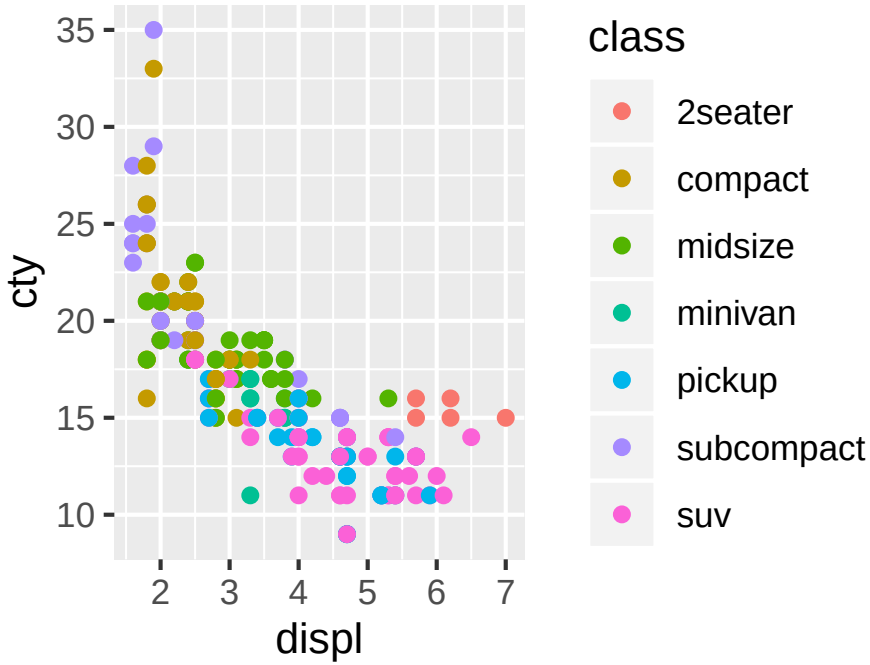
```
ggplot(mpg, aes(displ, cty)) +
  geom_point(color="blue")
```



Aesthetics

If you want appearance to be governed by some variable, put the specification inside `aes()` in the `ggplot()` function:

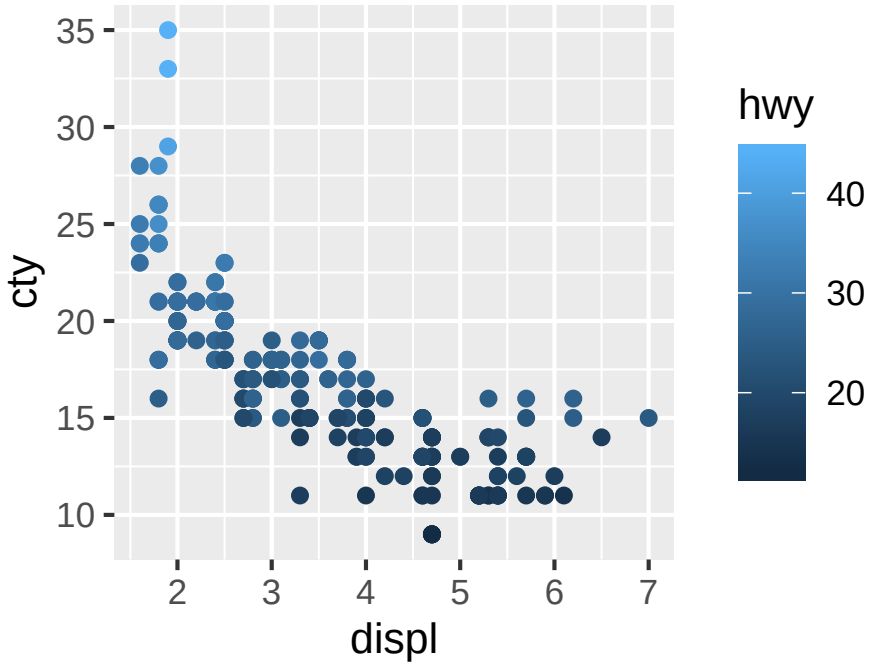
```
ggplot(mpg, aes(displ, cty, color = class)) +
  geom_point()
```



Common Aesthetic Attributes for Variables

Color and Fill:

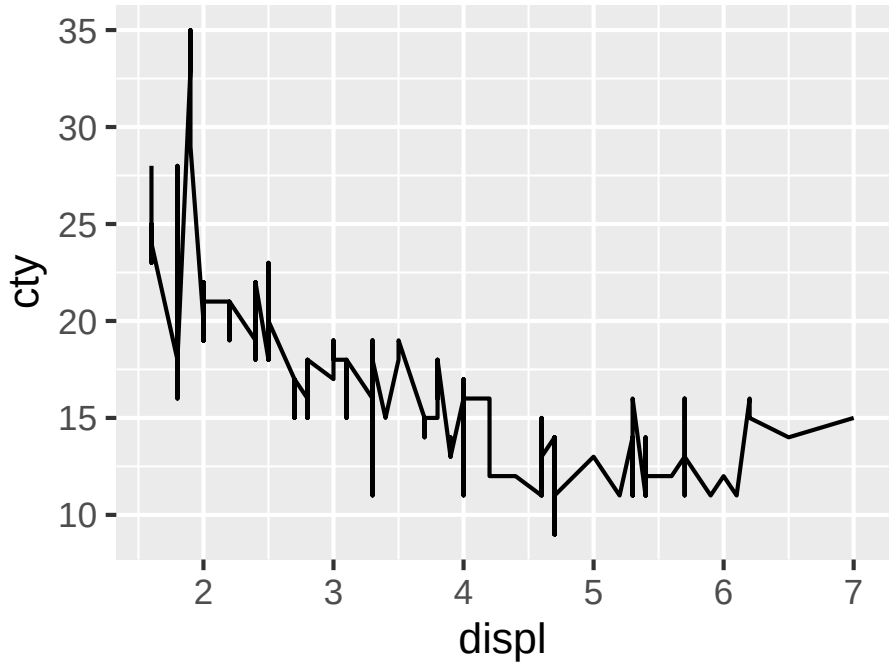
```
ggplot(mpg, aes(displ, cty, color=hwy)) +
    geom_point()
```



Common Aesthetic Attributes for Variables

Color and Fill: Good for Continous and Categorical Variables

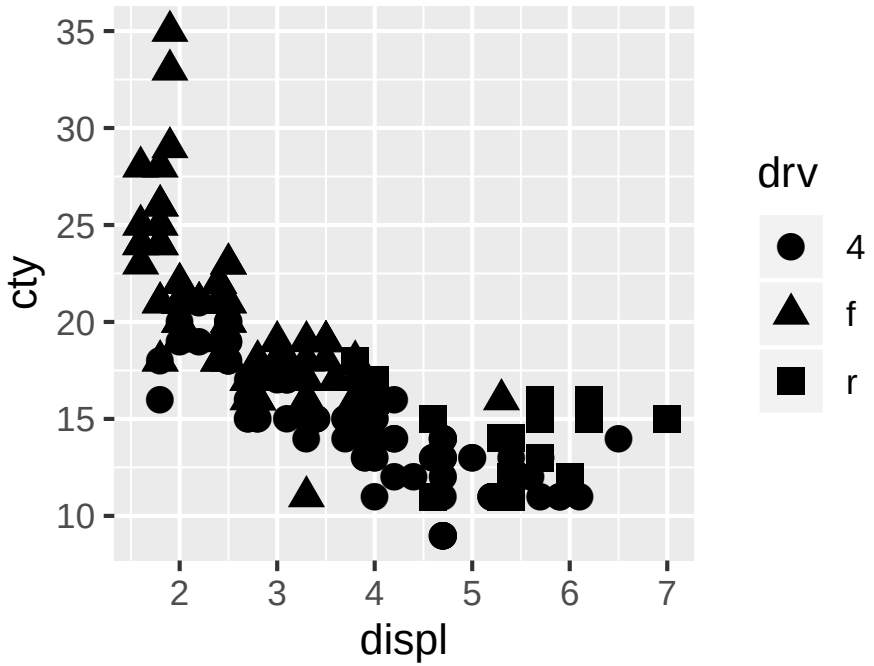
```
ggplot(mpg, aes(displ, cty, fill=hwy)) +
    geom_line()
```



Common Aesthetic Attributes for Variables

Shape: Good for Categorical Variables

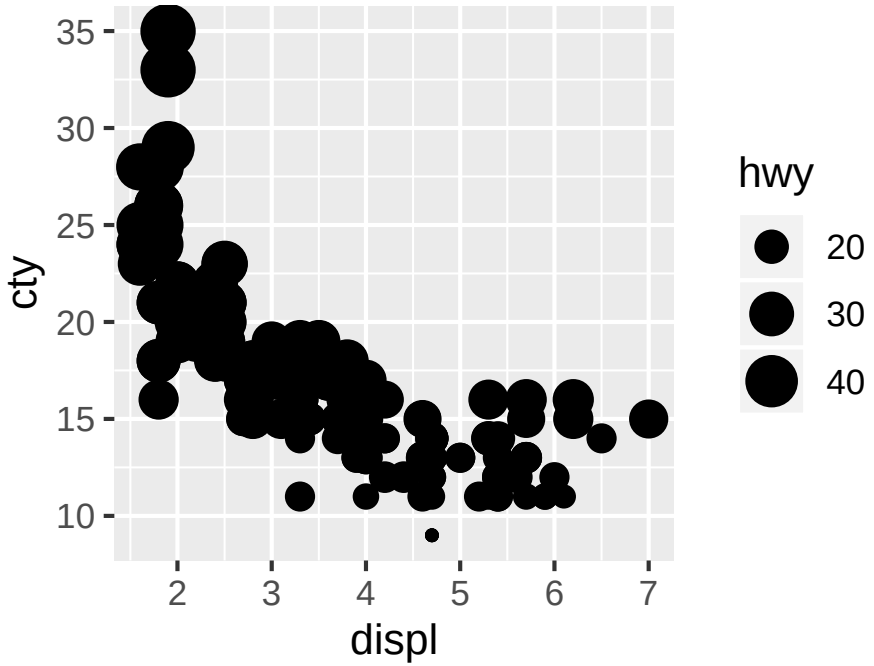
```
ggplot(mpg, aes(displ, cty, shape = drv)) +
    geom_point(size=3)
```



Common Aesthetic Attributes for Variables

Size: Good for Continuous Variables

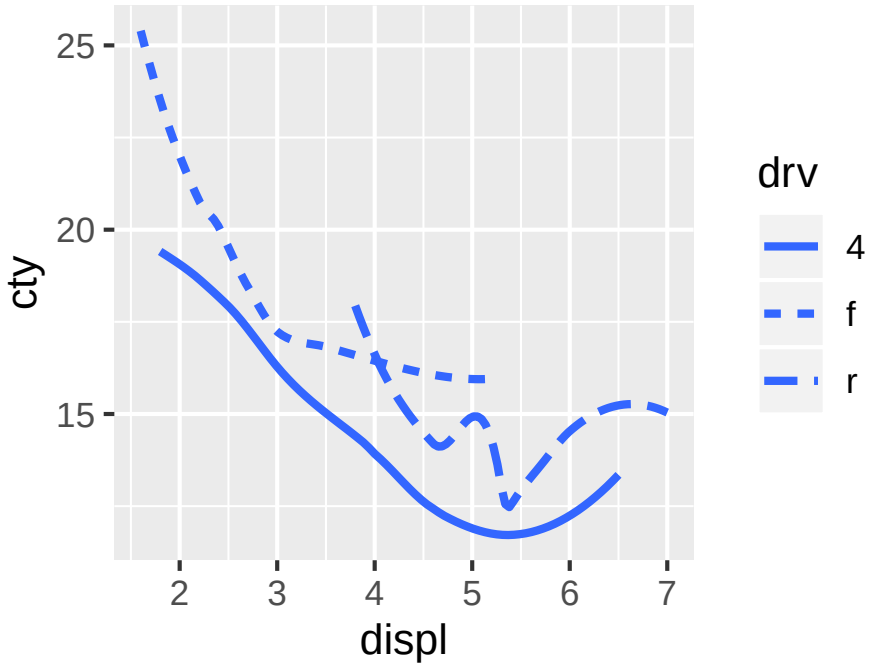
```
ggplot(mpg, aes(displ, cty, size = hwy)) +
    geom_point()
```



Common Aesthetic Attributes for Variables

Line Type: Good for Categorical Variables

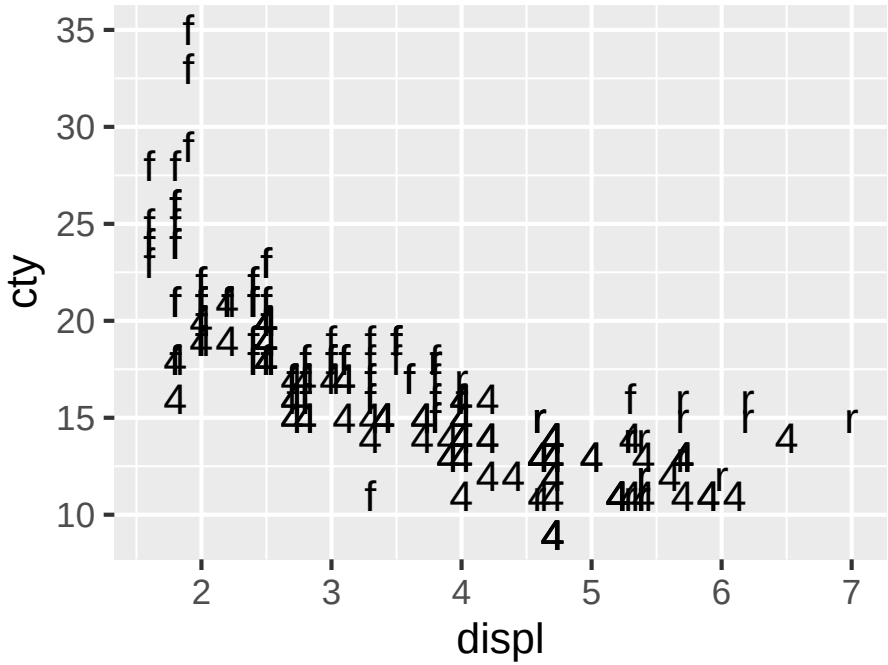
```
  ggplot(mpg, aes(displ, cty, linetype = drv)) +
    geom_smooth(se = FALSE)

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



Common Aesthetic Attributes for Variables

Label and Family: Good for Categorical Variables. Use with `geom_text()`

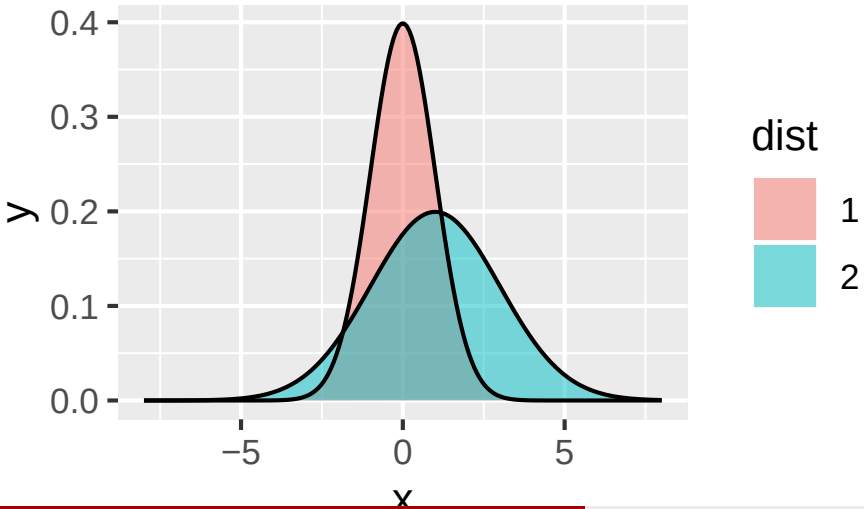
```
ggplot(mpg, aes(displ, cty, label = drv)) +
  geom_text()
```



Common Constant Aesthetic Attributes

Alpha: Good for overlapping data

```
xvals <- seq(-8, 8, length.out=200)
y1 <- dnorm(xvals, mean=0, sd=1)
y2 <- dnorm(xvals, mean=1, sd=2)
dx <- data.frame(x=rep(xvals, 2), y=c(y1,y2),
                 dist=c(rep("1",200), rep("2", 200)))
ggplot(dx, aes(x,y,  fill=dist)) + geom_polygon(alpha=0.5) +
  geom_line()
```



Facetting

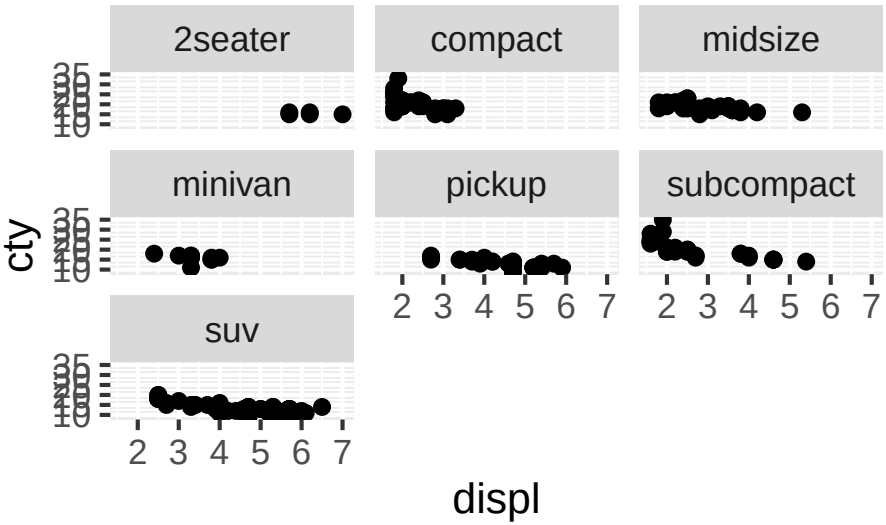
An alternative to using aesthetics to map properties of the data is to use **faceting**

Faceting describes how to break up the data into subsets and how to display those subsets.

- `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()

```
ggplot(mpg, aes(displ, cty)) +
  geom_point() +
  facet_wrap(~class, as.table=T)
```



facet_wrap()

You can control how the ribbon is wrapped into a grid with the following arguments:

- `nrow` and `ncol` control how many rows and columns are displayed in the grid (only need to set one)
- `as.table` controls how the facets are laid out
  - `TRUE`: the highest values at the bottom-right
  - `FALSE`: the highest values at the top-right
- `dir` controls the direction of the wrap: horizontal (`h`) or vertical (`v`)

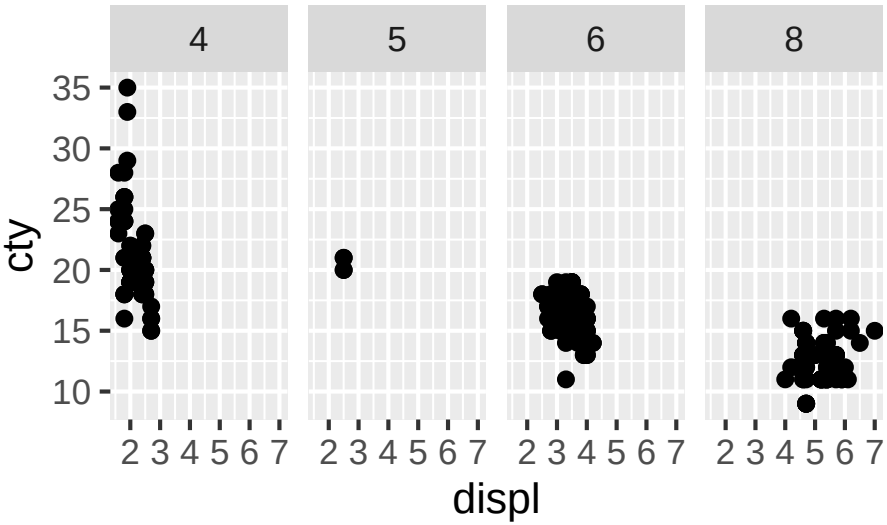
facet_grid()

facet_grid() lays out plots in a 2D grid, as defined by a formula:

- a spreads the values of a across the columns. This direction facilitates comparisons of the y-position, because vertical scales are aligned.
- b . spreads the values of b down the rows. This direction facilitates comparison of the x position, because the horizontal scales are aligned. This makes it particularly useful for comparing distributions.
- a b spreads a across columns and b down rows.

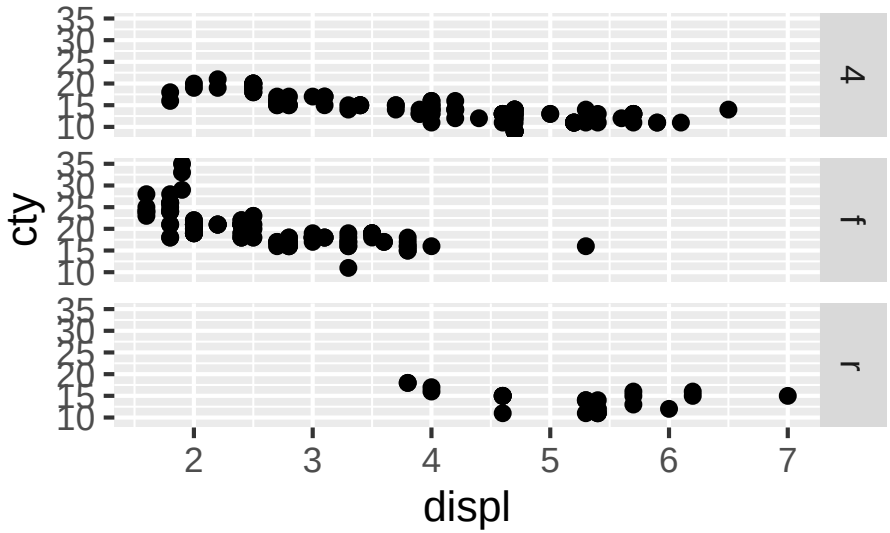
facet_grid()

```
ggplot(mpg, aes(displ, cty)) +
  geom_point() +
  facet_grid(. ~ cyl)
```



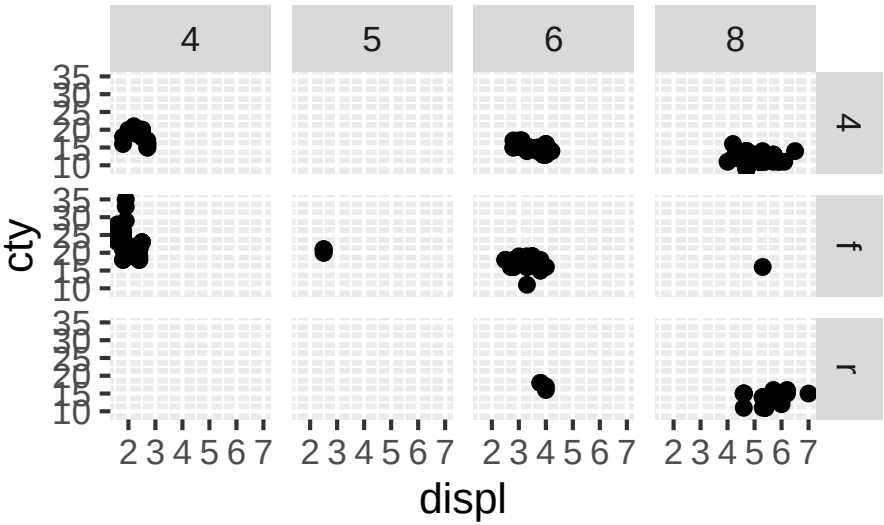
facet_grid()

```
ggplot(mpg, aes(displ, cty)) +
  geom_point() +
  facet_grid(drv ~ .)
```



facet_grid()

```
ggplot(mpg, aes(displ, cty)) +
  geom_point() +
  facet_grid(drv ~ cyl)
```



Plan

1. Introduction to ggplot2
2. Plotting Scenarios
   - One Continuous Variable
   - One Categorical Variable
   - Two Continuous Variables
   - One Categorical, and One Continuous Variable
3. Plot Options
   - faceting
4. Axes, Legends, and Themes
5. Miscellaneous

Scales and Axes

Scales:

- Map values in the data space to values in the aesthetic space (i.e., color)
- Draw a legend or axes that provide and inverse mapping to make it possible to read the original data from the plot

Scales and Axes

Use `scale_()` functions to adjust:

- Scale/axis names
- Breaks and labels

See:

- `scale_x_continuous()`
- `scale_x_discrete()`
- `scale_fill_gradient()`

Themes

There are around 40 unique elements that control the appearance of the plot. They can be roughly grouped into 5 categories:

1. Plot
2. Axis
3. Legend
4. Pane
5. Facet

Themes

Some elements that affect the plot as a whole:

- `plot.background` (set with `element_rect()`)
- `plot.title` (set with `element_text()`)
- `plot.margin` (set with `margin()`)

Axis Elements

- `axis.line` and `axis.ticks` are set with `element_line()`
- `axis.ticks.length` is set with `unit()`
- `axis.text`, `axis.text.x`, `axis.text.y`, `axis.title`, `axis.title.x`, `axis.title.y` are all set with `element_text()`

Legends

Legends elements control the appearance of all legends in the plot. Individual legends can be modified by modifying the same elements in `guide_legend()` and `guide_colourbar`

Legends

Legends elements control the appearance of all legends in the plot. Individual legends can be modified by modifying the same elements in `guide_legend()` and `guide_colourbar`

- `legend.text.align` and `legend.title.align`
- `legend.text` and `legend.title` are set with
- `legend.background` and `legend.key`
- `legend.text.size`, `legend.key.height`, `legend.key.width`, and `legend.margin`

Legends

Legends elements control the appearance of all legends in the plot. Individual legends can be modified by modifying the same elements in `guide_legend()` and `guide_colourbar`

- `legend.text.align` and `legend.title.align` are set with a number from 0 to 1.
- `legend.text` and `legend.title` are set with `element_text()`
- `legend.background` and `legend.key` are set with `element_rect()`
- `legend.text.size`, `legend.key.height`, `legend.key.width`, and `legend.margin` are set with `unit()`

There are four other properties that control how legends are laid out in the context of the plot: `legend.position`, `legend.direction`, `legend.justification`, `legend.box`

Plan

1. Introduction to ggplot2
2. Plotting Scenarios
   - One Continuous Variable
   - One Categorical Variable
   - Two Continuous Variables
   - One Categorical, and One Continuous Variable
3. Plot Options
   - faceting
4. Axes, Legends, and Themes
5. Miscellaneous

Panel Elements

- `aspect.ratio` is set with a numeric value.
- `panel.background` and `panel.border` are set with `element_rect()`
- `panel.grid.major`, `panel.grid.major.x`, `panel.grid.major.y`, `panel.grid.minor`, `panel.grid.minor.x`, and `panel.grid.minor.y` are set with `element_line()`

Panel Elements

- `aspect.ratio` is set with a numeric value.
- `panel.background` and `panel.border` are set with `element_rect()`
- `panel.grid.major`, `panel.grid.major.x`, `panel.grid.major.y`, `panel.grid.minor`, `panel.grid.minor.x`, and `panel.grid.minor.y` are set with `element_line()`

The main difference between `panel.background` and `panel.border` is that the background is drawn underneath the data, and the border is drawn on top of the data. Therefore, should always set `fill = NA` when overriding `panel.border`

Note: `aspect.ratio` control the aspect ratio of the panel, not the entire plot.

Faceting Elements

- `panel.margin`, `panel.margin.x`, and `panel.margin.y` are set with `unit()`
- `strip.background` is set with `element_rect()`
- `strip.text`, `strip.text.x`, and `strip.text.y` are set with `element_text()`

Faceting Elements

- `panel.margin`, `panel.margin.x`, and `panel.margin.y` are set with `unit()`
- `strip.background` is set with `element_rect()`
- `strip.text`, `strip.text.x`, and `strip.text.y` are set with `element_text()`

`strip.text.x` affects both `facet_wrap()` and `facet_grid()`.
`strip.text.y` affects only `facet_grid()`

What We Didn't Cover

- Coordinate Systems: see `coord_map()`, `coord_polar()`, and `coord_trans()`
- Position Adjustments. For example: bars on top of each other or side-by-side. These settings can be set in `geom_bar()`
- Many Smaller Details: see *ggplot2: Elegant Graphics for Data Analysis* by Hadley Wickham
- And/or: visit the Tidyverse website: `https://ggplot2.tidyverse.org`

Credit

This workshop is based on slides compiled by Stephanie L. DeMora (GreadQuant, 2020)

Questions?