Data Visualization and Maps I

HES 505 Fall 2024: Session 25

Carolyn Koehn

Objectives

By the end of today you should be able to:

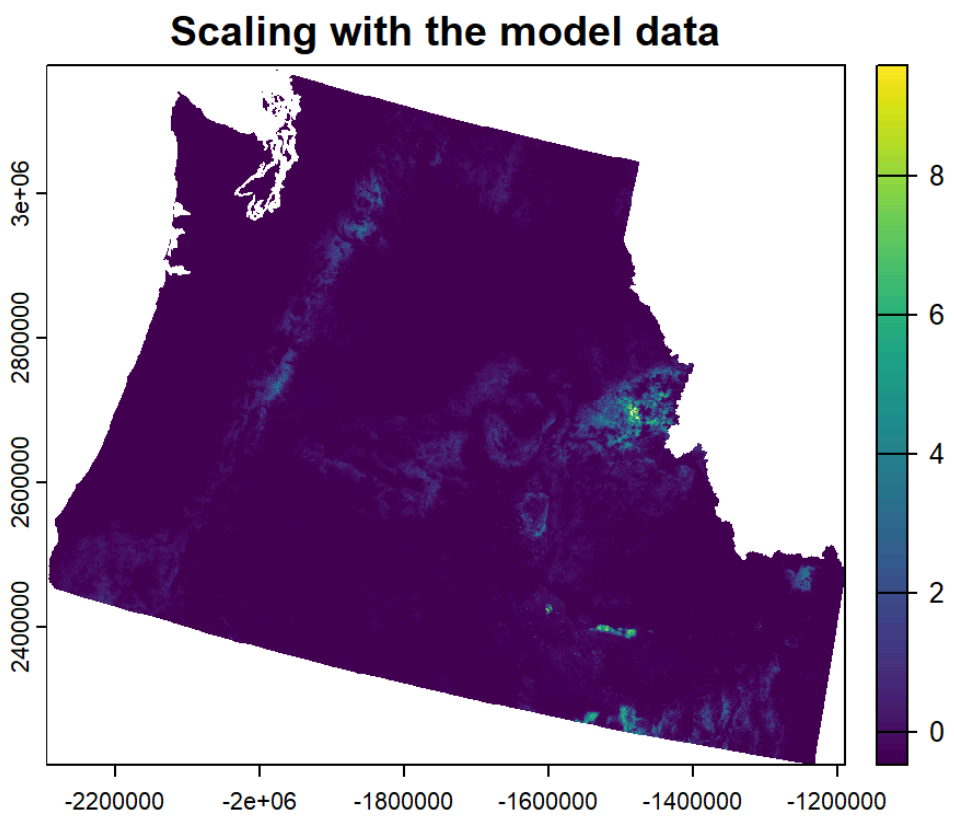
- Describe some basic principles of data visualization
- Extend principles of data visualization to the development of maps
- Distinguish between several common types of spatial data visualization
- Understand the relationship between the Grammar of Graphics and `ggplot` syntax
- Describe the various options for customizing `ggplots` and their syntactic conventions

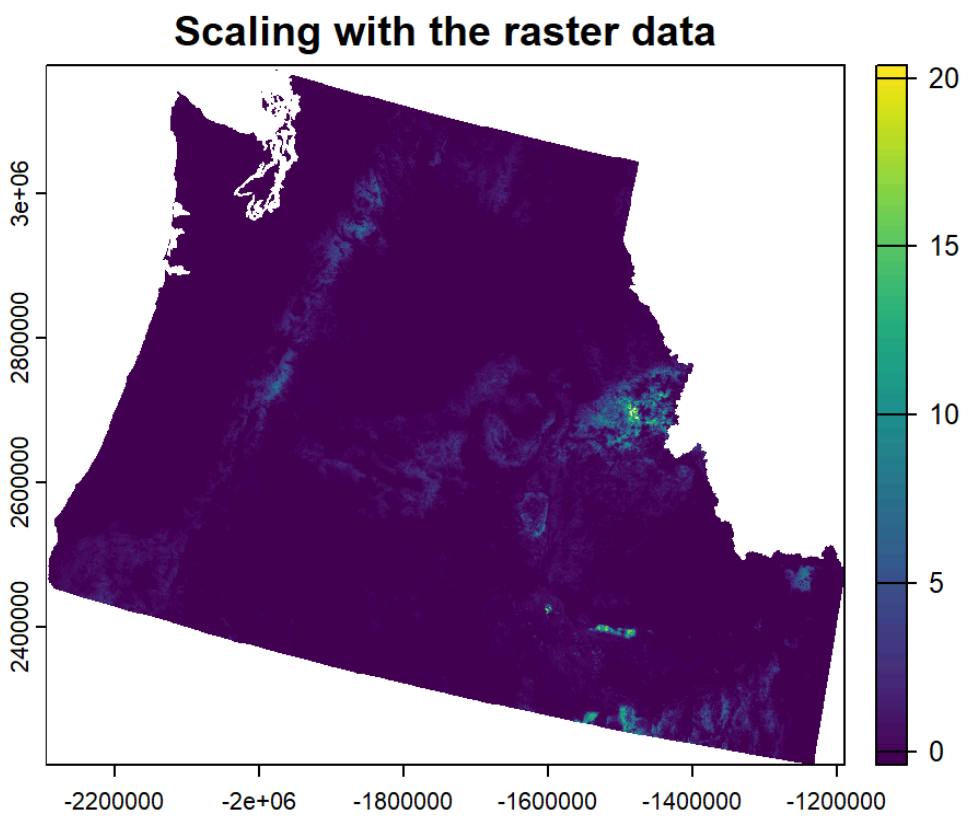
But first… Scaling

Assignment 9: Scaling the hazard data

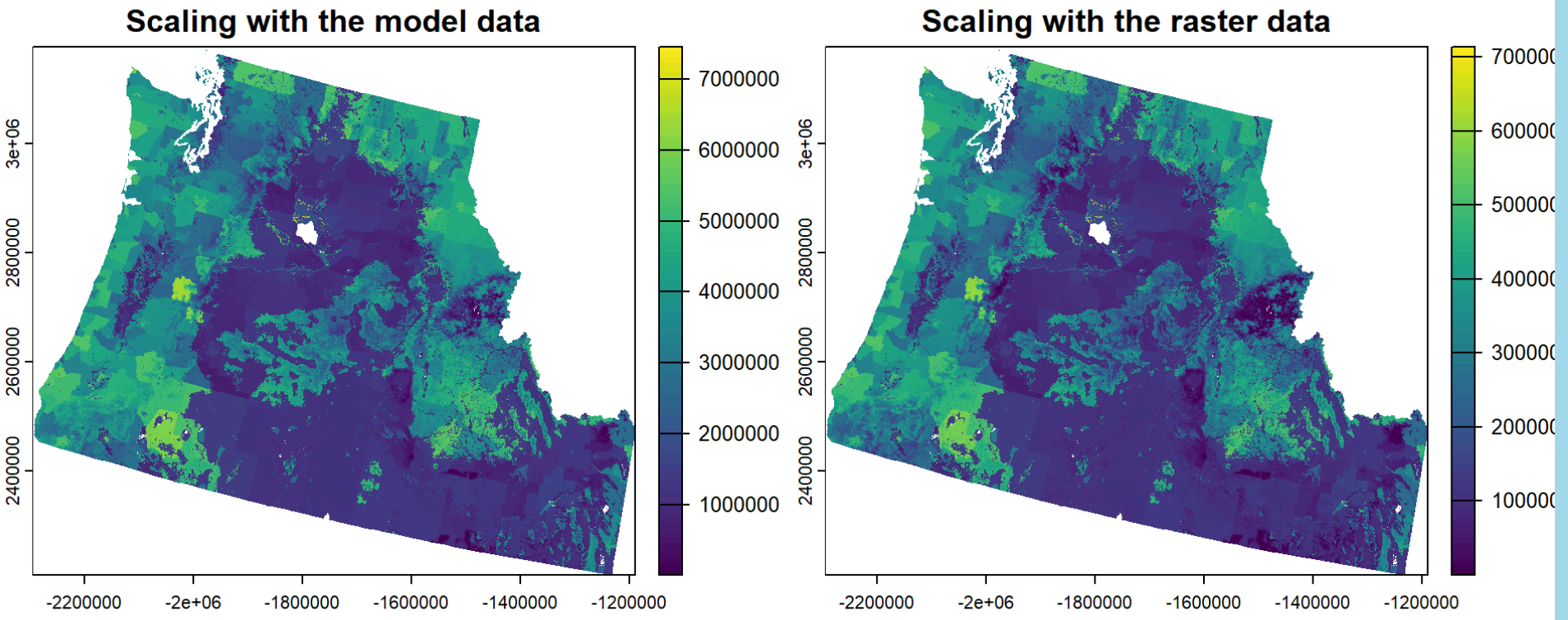
```
1 hazard.smooth.scl <- (hazard.smooth - mean(incident.cejst.prep$hazard))/sd(
2 #versus
3 hazard.smooth.scl.nogood <- scale(hazard.smooth)
```

Assignment 9: Scaling the hazard data





Assignment 9: Different predictions for different scaling



Introduction to Data Visualization

Principles vs. Rules

- Lots of examples of *good* and *bad* data visualization
- What makes a graphic good (or bad)?
- Who decides?

- **Rule:** externally compels you, through force, threat or punishment, to do the things someone else has deemed good or right.
- **Principle:** internally motivating because it is a *good practice*; a general statement describing a philosophy that good rules should satisfy
- Rules contribute to the design process, but do not guarantee a satisfactory outcome

> “Graphical excellence is the well-designed presentation of interesting data—a matter of substance, of statistics, and of design … [It] consists of complex ideas communicated with clarity, precision, and efficiency. … [It] is that which gives to the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space … [It] is nearly always multivariate … And graphical excellence requires telling the truth about the data.”
> — Edward Tufte

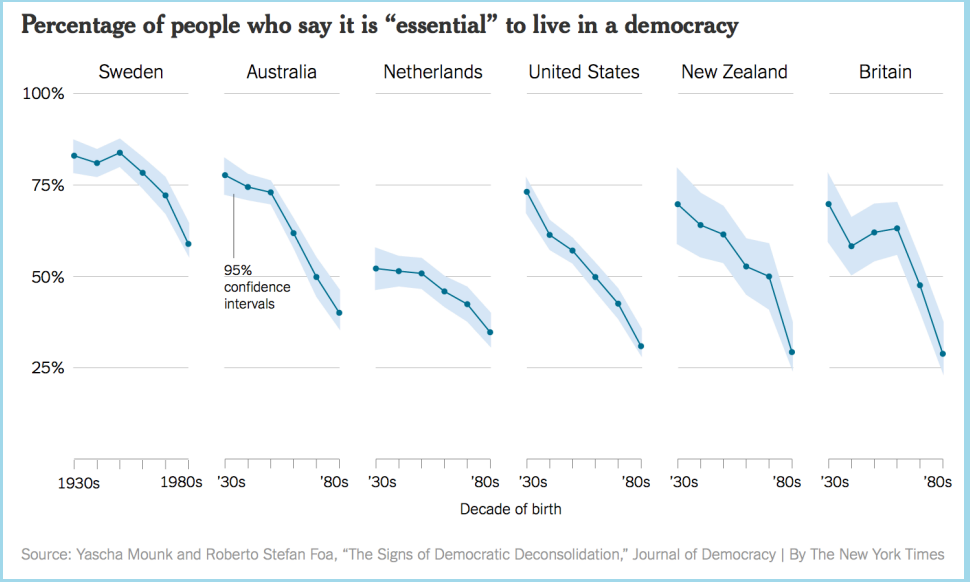
Ugly, Wrong, and Bad

- *Ugly*: graphic is clear and informative, but has aesthetic issues
- *Bad*: graphic is unclear, confusing, or decieving
- *Wrong*: the figure is objectively incorrect

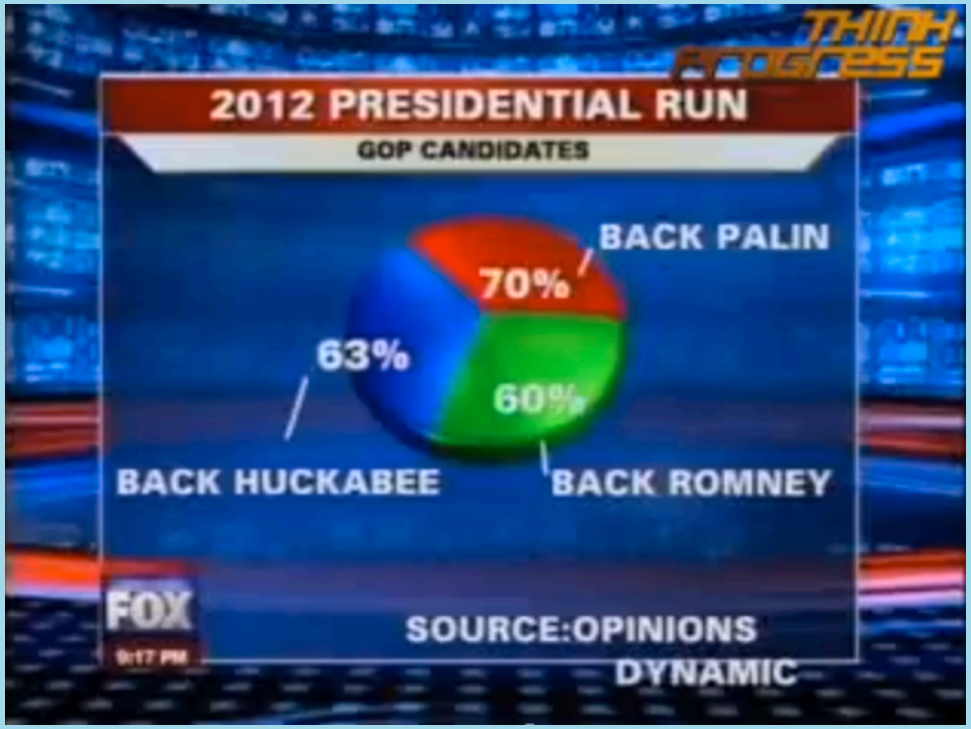
Monstrous Costs' by Nigel Holmes from Healy 2018

Bad and Wrong

- Presentation of the data is (intentionally?) decieving
- Presentation is just incorrect



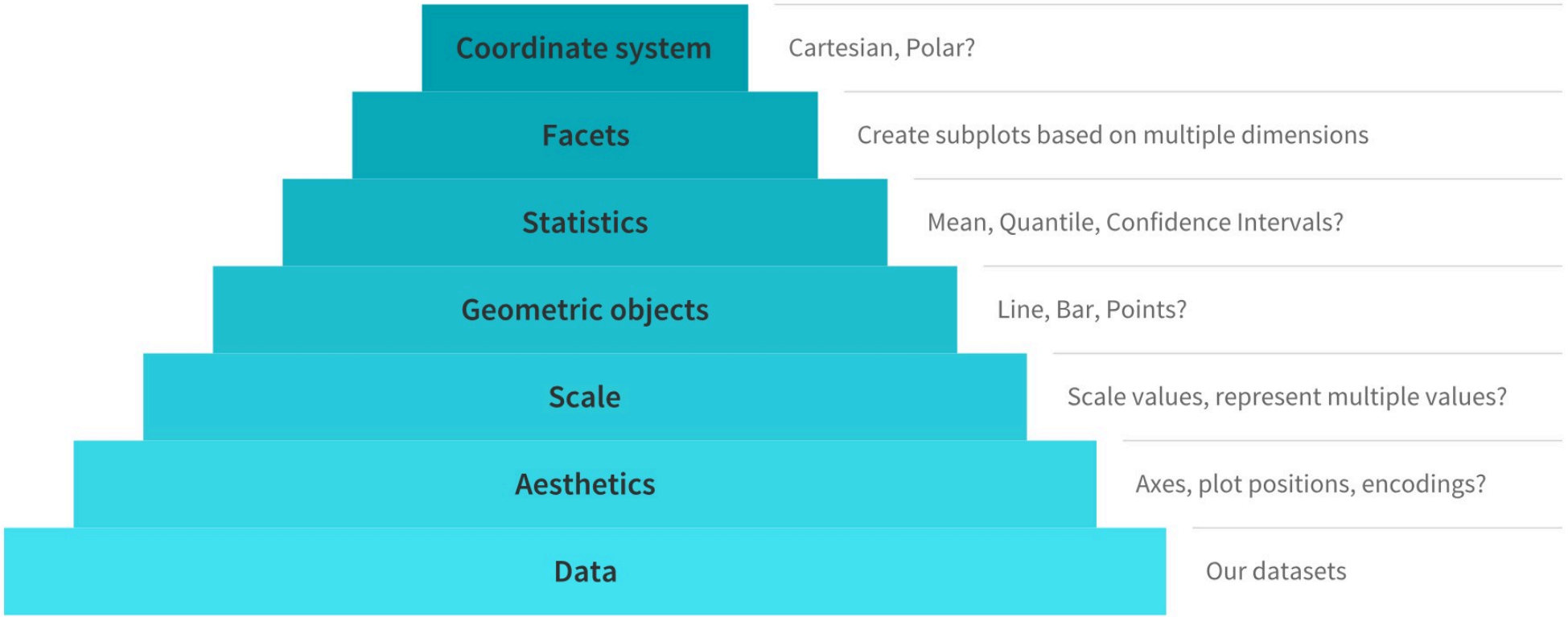
Tricky (from Healy 2018)



Wrong

Grammar of Graphics (Wilkinson 2005)

Major Components of the Grammar of Graphics



Aesthetics: Mapping Data to Visual Elements

- Define the systematic conversion of data into elements of the visualization
- Are either categorical or continuous (exclusively)
- Examples include `x`, `y`, `fill`, `color`, and `alpha`

Table 2.1: Types of variables encountered in typical data visualization scenarios.

Type of variable	Examples	Appropriate scale	Description
quantitative/numerical continuous	1.3, 5.7, 83, $1.5x10^{-2}$	continuous	Arbitrary numerical values. These can be integers, rational numbers, or real numbers.
quantitative/numerical discrete	1, 2, 3, 4	discrete	Numbers in discrete units. These are most commonly but not necessarily integers. For example, the numbers 0.5, 1.0, 1.5 could also be treated as discrete if intermediate values cannot exist in the given dataset.
qualitative/categorical unordered	dog, cat, fish	discrete	Categories without order. These are discrete and unique categories that have no inherent order. These variables are also called *factors*.
qualitative/categorical ordered	good, fair, poor	discrete	Categories with order. These are discrete and unique categories with an order. For example, "fair" always lies between "good" and "poor". These variables are also called *ordered factors*.
date or time	Jan. 5 2018, 8:03am	continuous or discrete	Specific days and/or times. Also generic dates, such as July 4 or Dec. 25 (without year).
text	The quick brown fox jumps over the lazy dog.	none, or discrete	Free-form text. Can be treated as categorical if needed.

From Wilke 2019

Scales

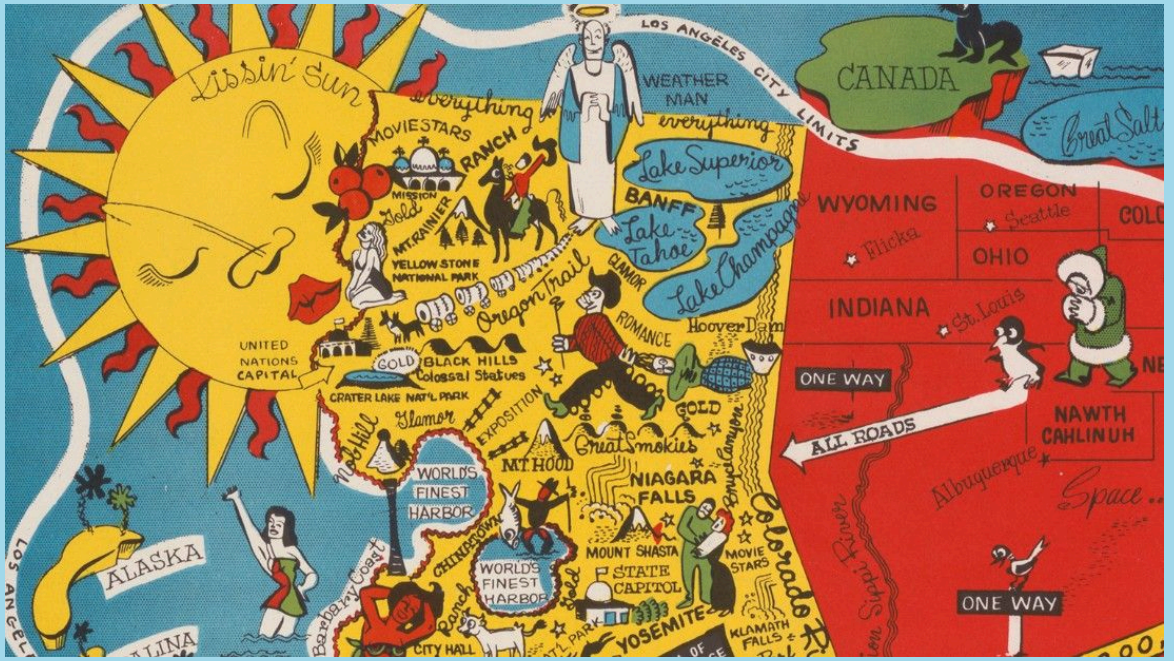
- Scales map data values to their aesthetics
- Must be a one-to-one relationship; each specific data value should map to only one aesthetic

Principles of Data Visualization

- Be Honest
- Principle of proportional ink
- Avoid unnecessary ‘chart junk’
- Use color judiciously
- Balance data and context

Extending Data Viz to Maps

Telling stories with maps



- Maps organize a lot of information in a coherent way
- They invite critique and inspection
- They are also aesthetic objects that can engage broader audiences

Key Issues

- Thinking about projections
- Scale of the map
- Errors of Omission

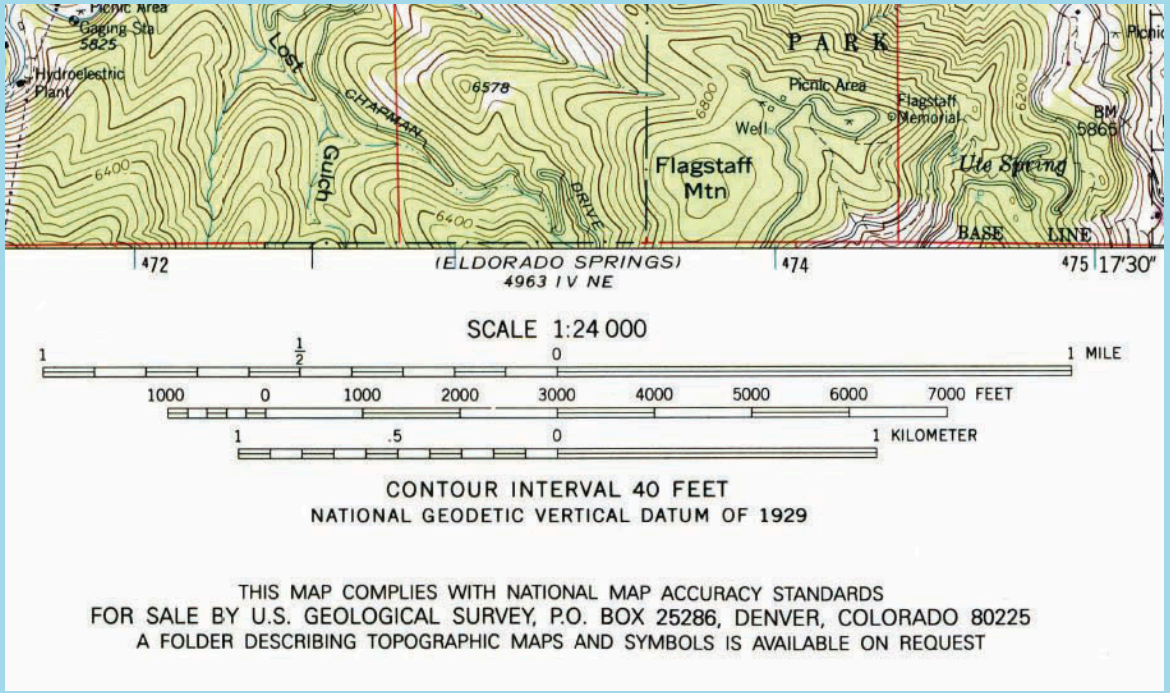
Cartographic Principles

1. Concept before compilation
2. Hierarchy with harmony (Important things should look important)
3. Simplicity from sacrifice
4. Maximum information at minimum cost
5. Engage emotion to enhance understanding

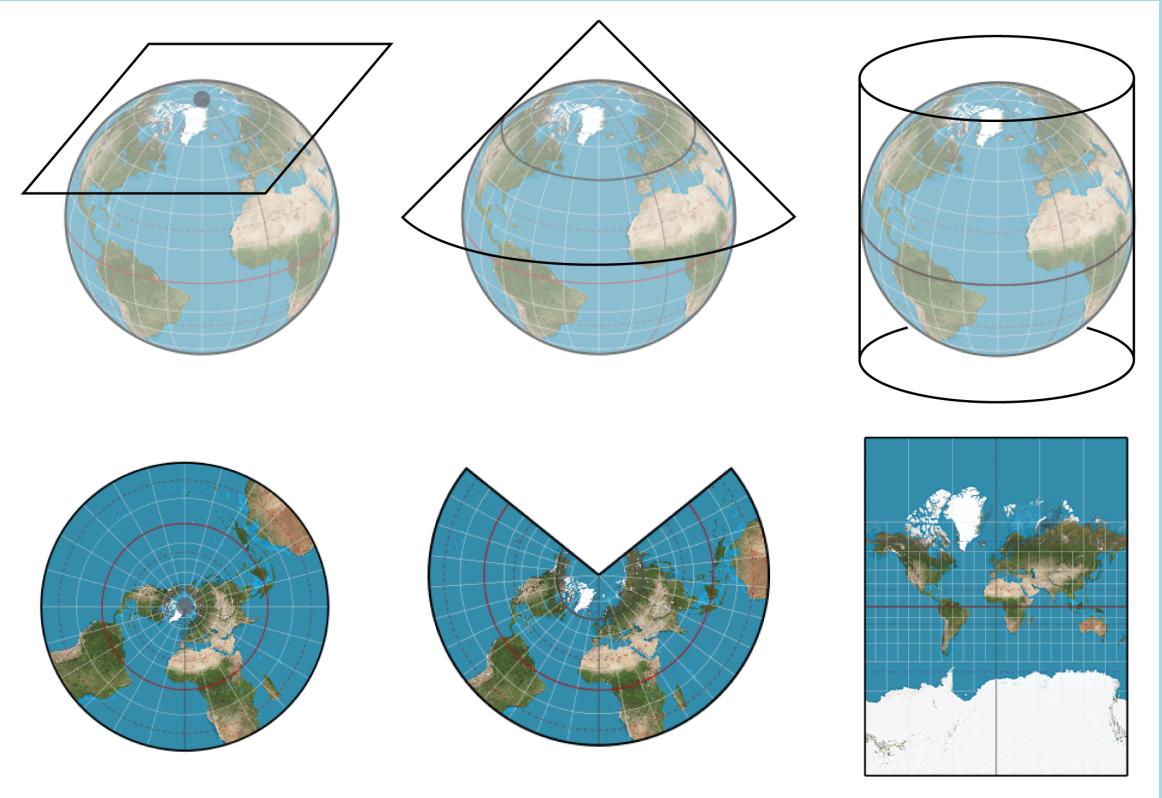
Map Elements

Scale

- Relates map distance to distance on the ground
- Ratio scales (1:24,000 or 1/24,000)
- Graphic scales
- Large vs. small-scale?



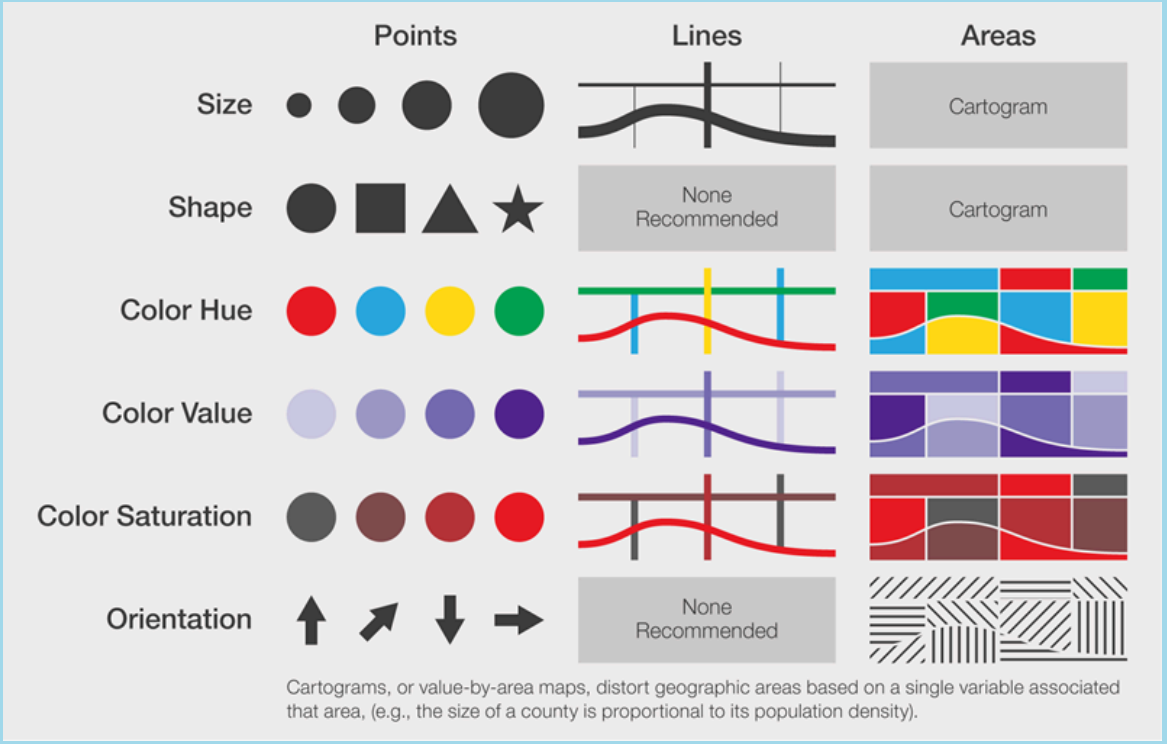
Projection



Developable Surfaces

- Distortion makes scale invalid across large areas
- Distortion increases with distance from standard line
- Five distortions: areas, angles, shapes, distances, and direction

Map Symbols

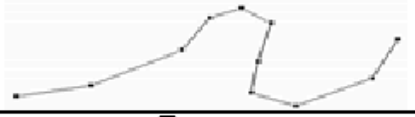
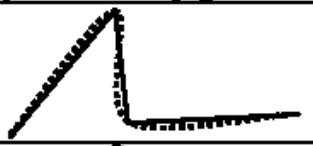


- Graphic code for retrieving information
- (De-)emphasize (un)important information
- Contrast and the role of colors

Generalization

> A good map tells a multitude of little white lies: it supresses truth to help the user see what needs to be seen…
> — Mark Monmonier

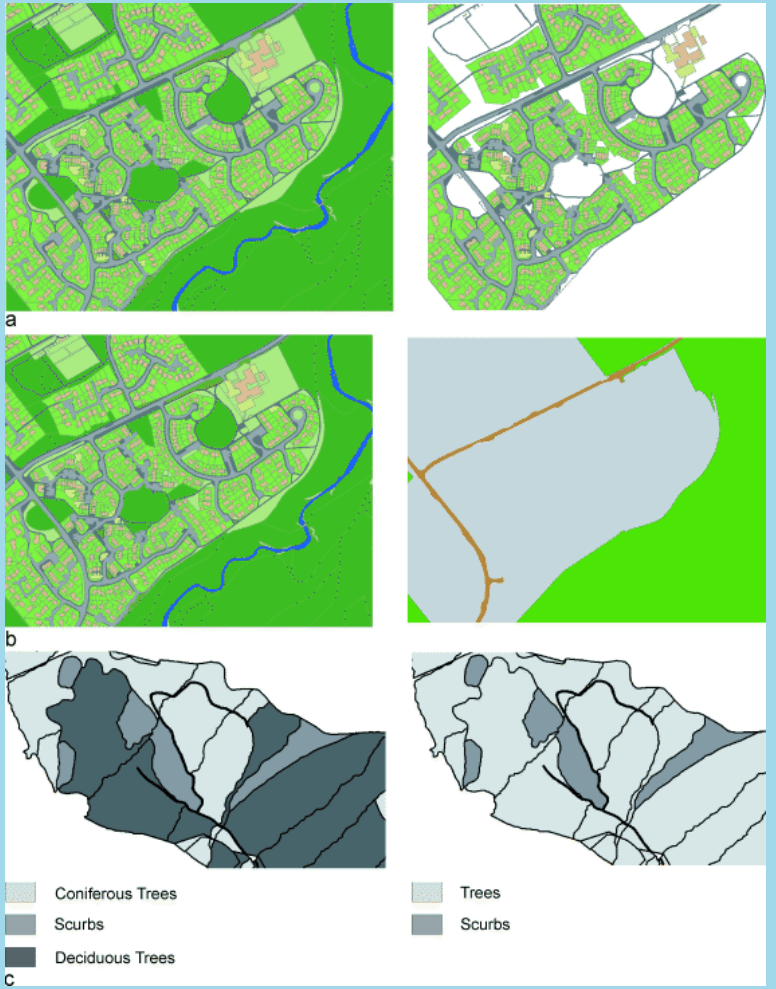
Geometry

Operations		Large-scale	Photo-reduced	Small-scale
Displacement				
Elimination				
(Scale-driven) generalisation				
Partial modification				
Point-reduction				
Smoothing	Curve-fitting			
	Filtering			
Typification				

Zhilin et al. 2008

Context

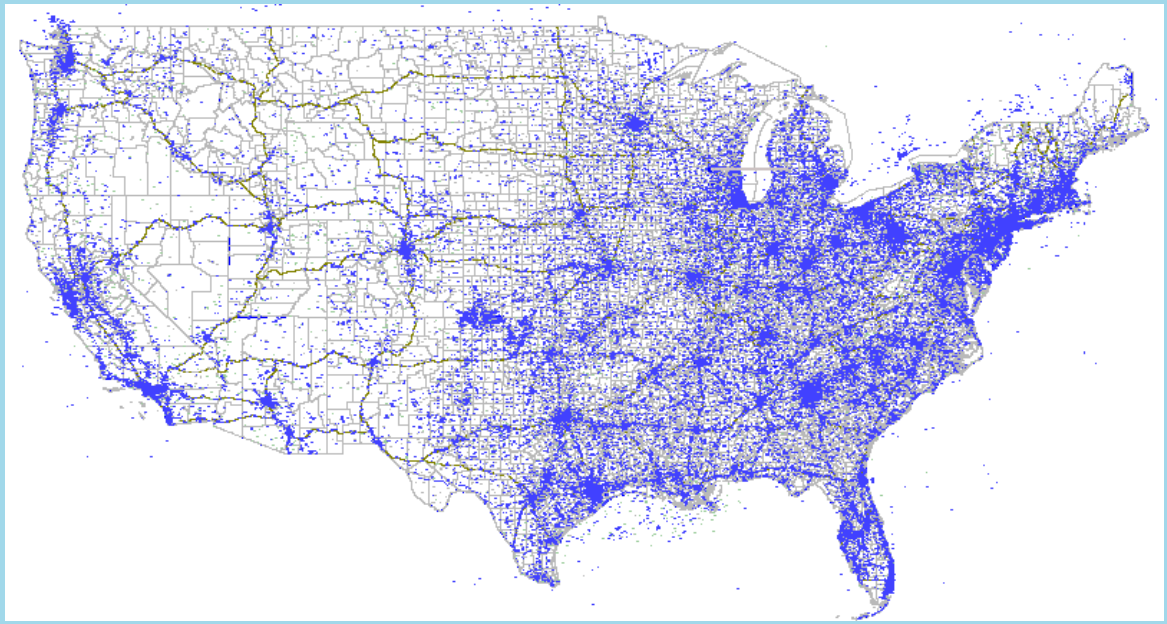
- Filter out irrelevant details
- Two elements: selection and classification
- Reflect interpretations of the relative importance of different features



Mackaness and Chaudry

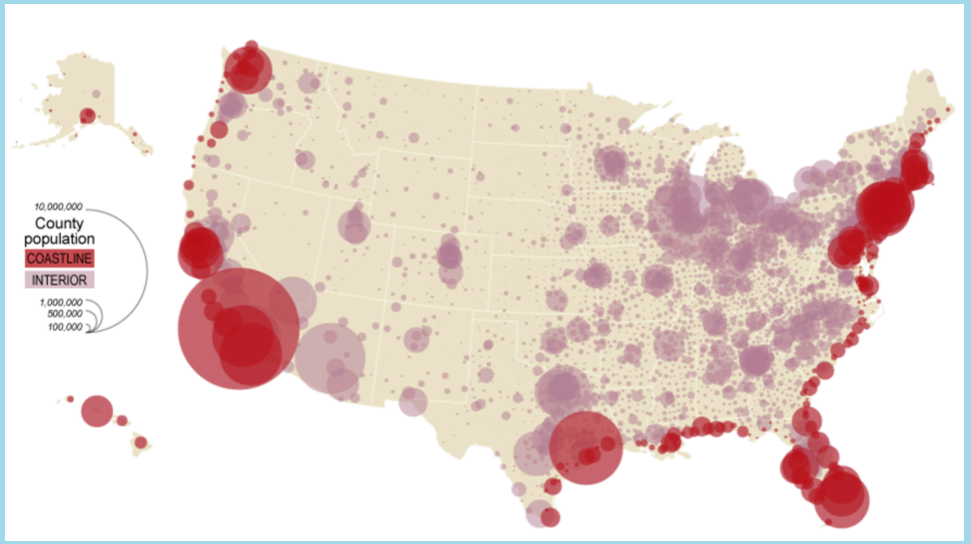
Data Maps

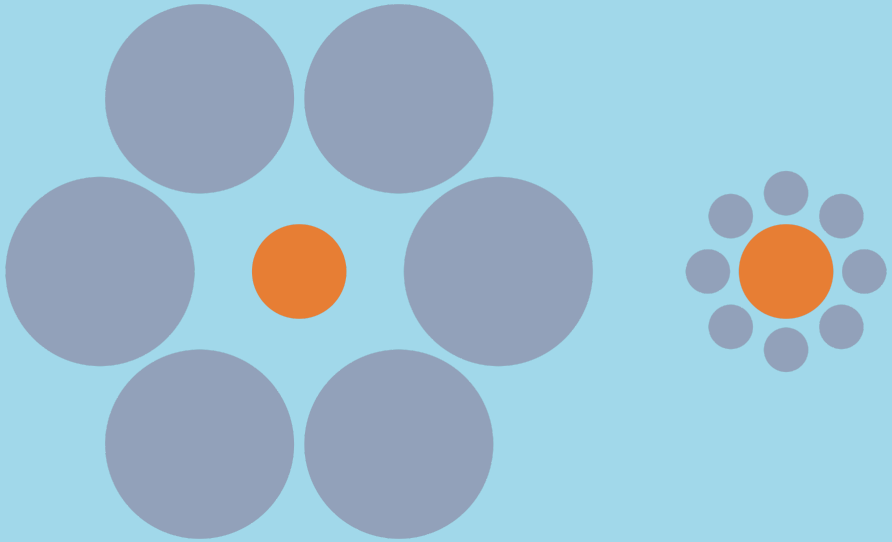
Point Maps



- Dot Maps: quantity represented by amount and concentration of dots
- Proportional Symbol Map: Geometric symbols scaled in proportion to a quantity

Ebbinghaus' illusion



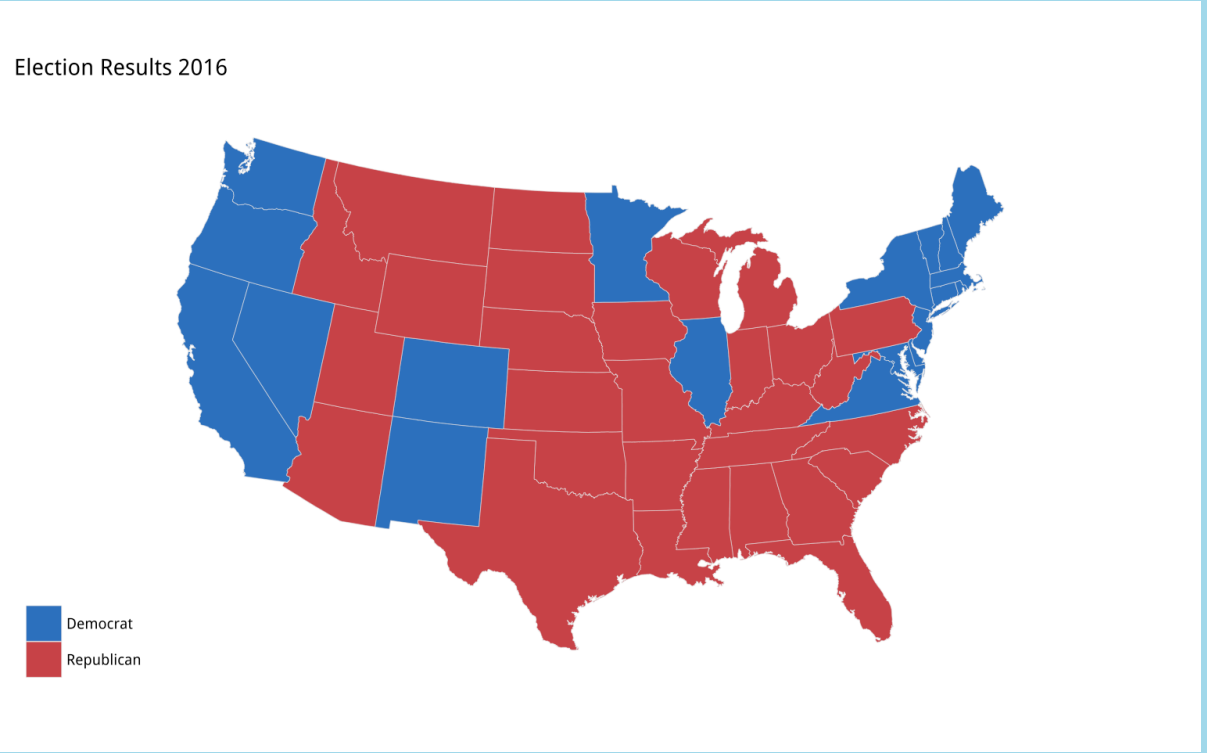


Line Maps

From High Country News

Choropleth

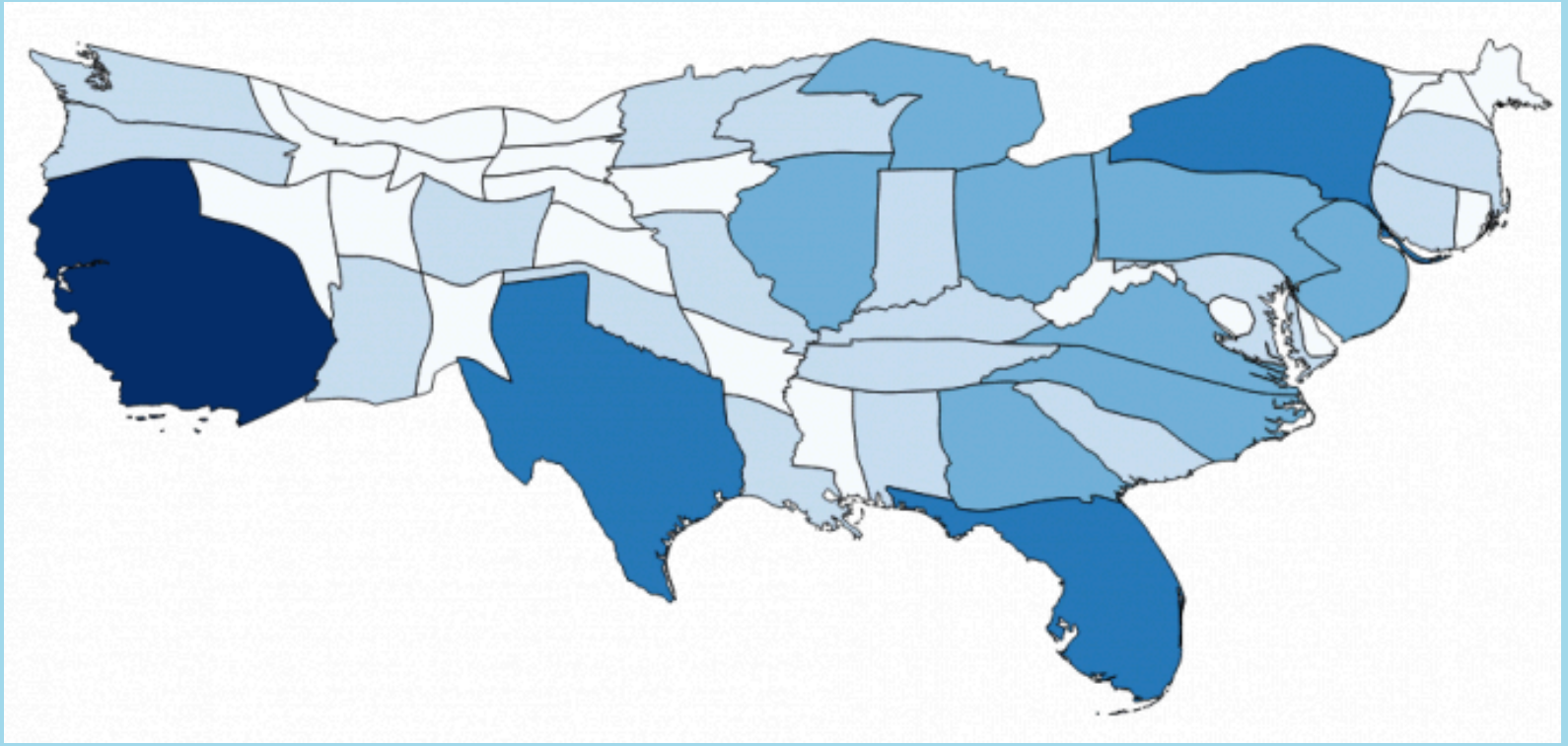
- Mapping color to geographies
- Common problems



From Healy 2019

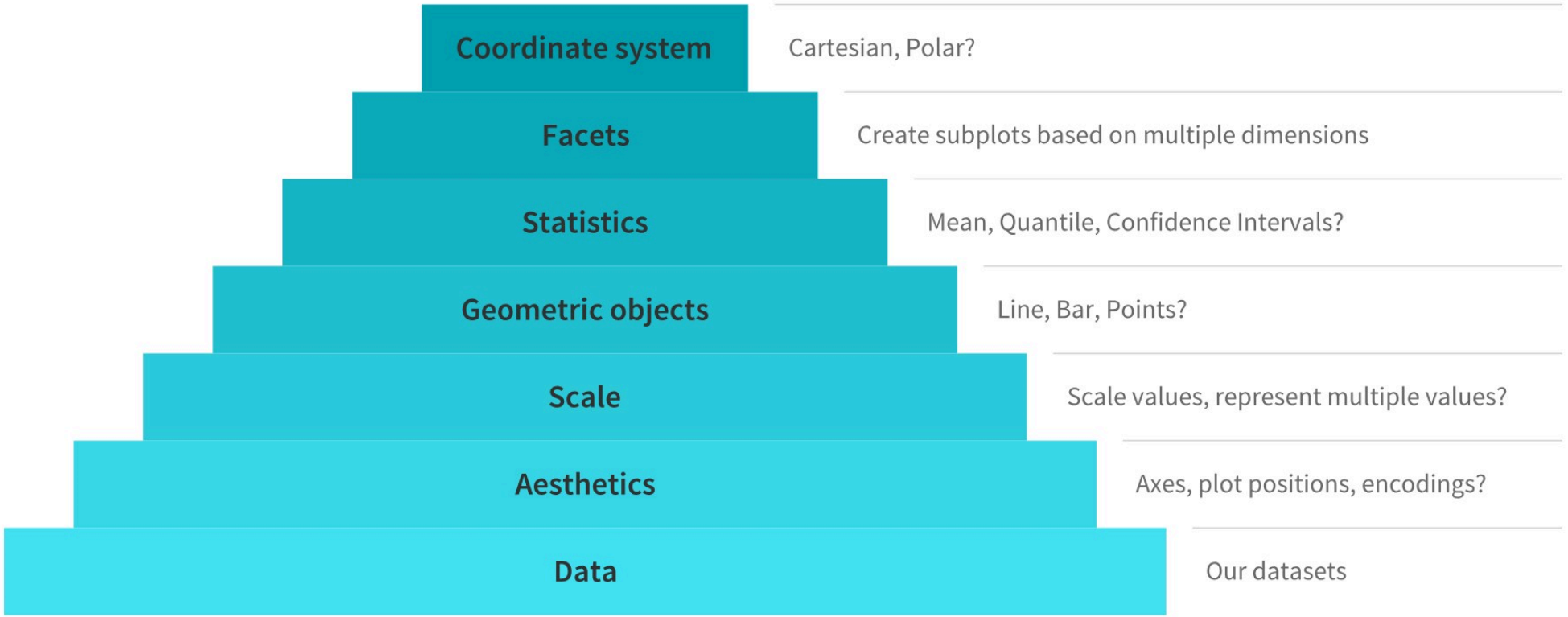
Cartogram

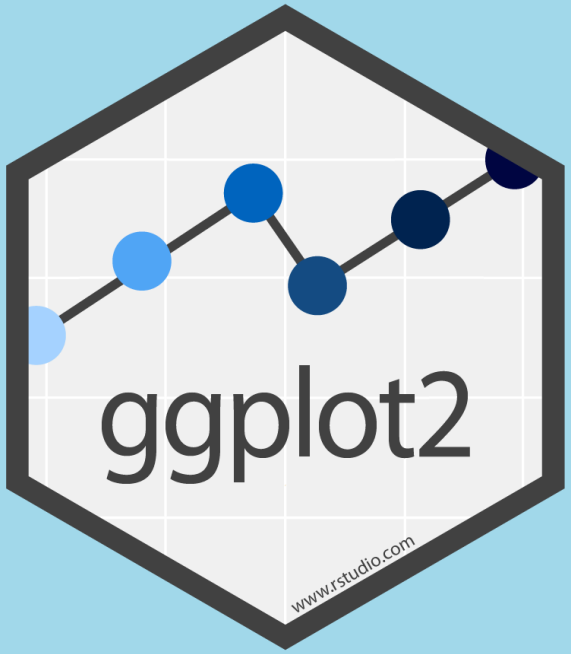
- Adjusts for differences in area, population, etc
- Common Problems



From Healy 2019

Major Components of the Grammar of Graphics





`{ggplot2}` is a system for declaratively creating graphics, based on "The Grammar of Graphics" (Wilkinson, 2005).

You provide the data, tell `ggplot2` how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.

Advantages of {ggplot2}

- consistent underlying “grammar of graphics” (Wilkinson 2005)
- very flexible, layered plot specification
- theme system for polishing plot appearance
- lots of additional functionality thanks to extensions
- active and helpful community

The Grammar of {ggplot2}

Component	Function	Explanation
Data	`ggplot(data)`	*The raw data that you want to visualise.*
Aesthetics	`aes()`	*Aesthetic mappings between variables and visual properties.*
Geometries	`geom_*()`	*The geometric shapes representing the data.*

The Grammar of {ggplot2}

Component	Function	Explanation
Data	`ggplot(data)`	*The raw data that you want to visualise.*
Aesthetics	`aes()`	*Aesthetic mappings between variables and visual properties.*
Geometries	`geom_*()`	*The geometric shapes representing the data.*
Statistics	`stat_*()`	*The statistical transformations applied to the data.*
Scales	`scale_*()`	*Maps between the data and the aesthetic dimensions.*
Coordinate System	`coord_*()`	*Maps data into the plane of the data rectangle.*
Facets	`facet_*()`	*The arrangement of the data into a grid of plots.*
Visual Themes	`theme()` and `theme_*()`	*The overall visual defaults of a plot.*

A Basic ggplot Example

The Data

Bike sharing counts in London, UK, powered by TfL Open Data

- covers the years 2015 and 2016
- incl. weather data acquired from freemeteo.com
- prepared by Hristo Mavrodiev for Kaggle
- further modification by myself

Variable	Description	Class
date	Date encoded as `YYYY-MM-DD`	date
day_night	`day` (6:00am–5:59pm) or `night` (6:00pm–5:59am)	character
year	`2015` or `2016`	factor
month	`1` (January) to `12` (December)	factor
season	`winter`, `spring`, `summer`, or `autumn`	factor
count	Sum of reported bikes rented	integer
is_workday	`TRUE` being Monday to Friday and no bank holiday	logical
is_weekend	`TRUE` being Saturday or Sunday	logical
is_holiday	`TRUE` being a bank holiday in the UK	logical
temp	Average air temperature (°C)	double
temp_feel	Average feels like temperature (°C)	double
humidity	Average air humidity (%)	double
wind_speed	Average wind speed (km/h)	double
weather_type	Most common weather type	character

ggplot2::ggplot()

ggplot: Create a new ggplot

Description

`ggplot()` initializes a ggplot object. It can be used to declare the input data frame for a graphic and to specify the set of plot aesthetics intended to be common throughout all subsequent layers unless specifically overridden.

Usage

```
ggplot(data = NULL, mapping = aes(), ..., environment = parent.frame())
```

Arguments

`data`	Default dataset to use for plot. If not already a data.frame, will be converted to one by `fortify()`. If not specified, must be supplied in each layer added to the plot.
`mapping`	Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot.
`...`	Other arguments passed on to methods. Not currently used.
`environment`	DEPRECATED. Used prior to tidy evaluation.

Details

`ggplot()` is used to construct the initial plot object, and is almost always followed by `+` to add component to the plot. There are three common ways to invoke `ggplot()`:

- `ggplot(df, aes(x, y, other aesthetics))`
- `ggplot(df)`
- `ggplot()`

Data

```
1  ggplot(data = bikes)
```

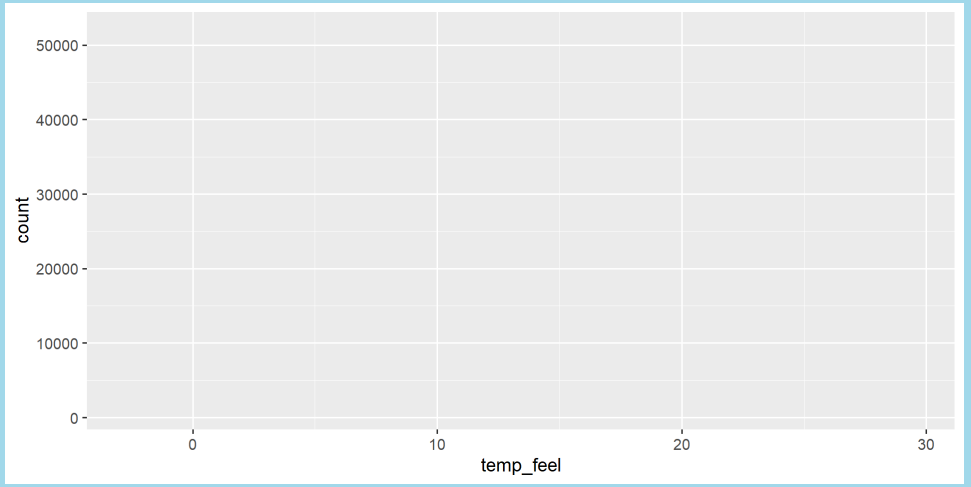
Aesthetic Mapping

= link variables to graphical properties

- positions (`x`, `y`)
- colors (`color`, `fill`)
- shapes (`shape`, `linetype`)
- size (`size`)
- transparency (`alpha`)
- groupings (`group`)

Aesthetic Mapping

```
1  ggplot(data = bikes) +
2    aes(x = temp_feel, y = count)
```



aesthetics

aes() outside as component

```
1 ggplot(data = bikes) +
2   aes(x = temp_feel, y = count)
```

aes() inside, explicit matching

```
1 ggplot(data = bikes, mapping = aes(x = temp_feel, y = count))
```

aes() inside, implicit matching

```
1 ggplot(bikes, aes(temp_feel, count))
```

aes() inside, mixed matching

```
1 ggplot(bikes, aes(x = temp_feel, y = count))
```

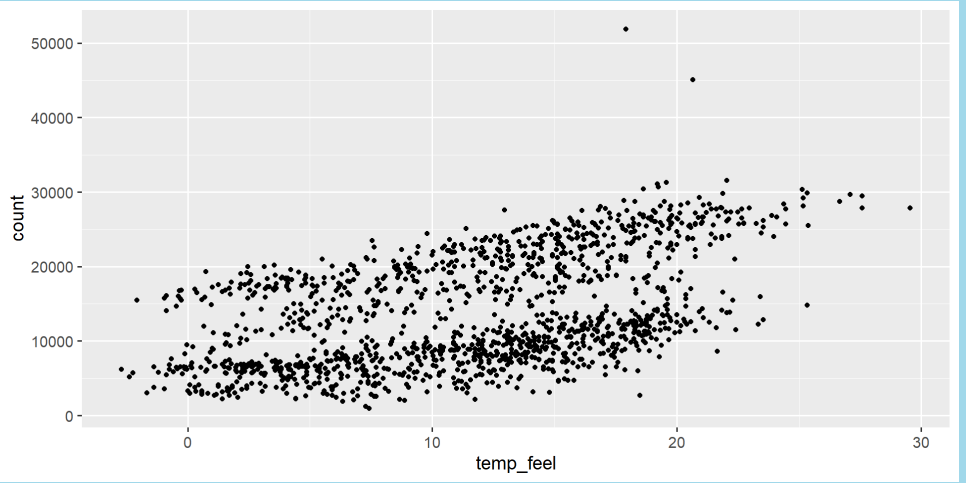
Geometries

= interpret aesthetics as graphical representations

- points
- lines
- polygons
- text labels
- …

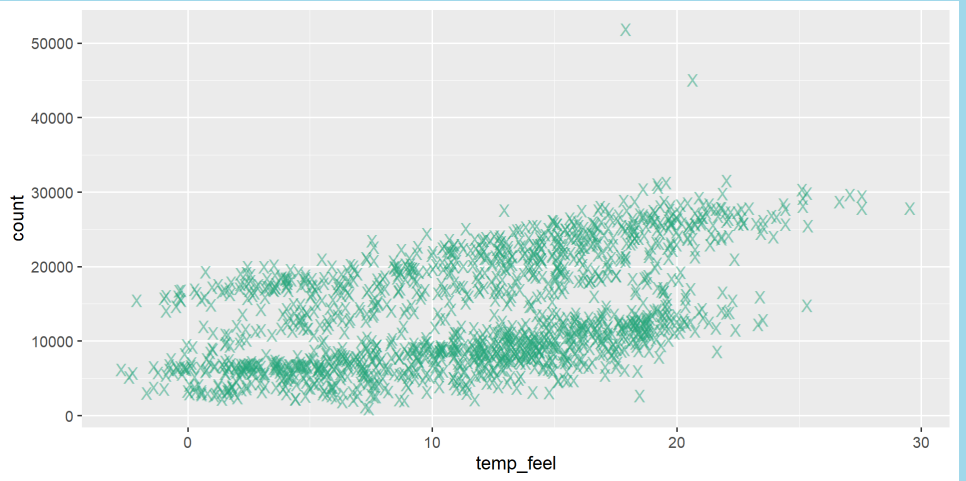
Geometries

```
ggplot(
    bikes,
    aes(x = temp_feel, y = count)
  ) +
  geom_point()
```



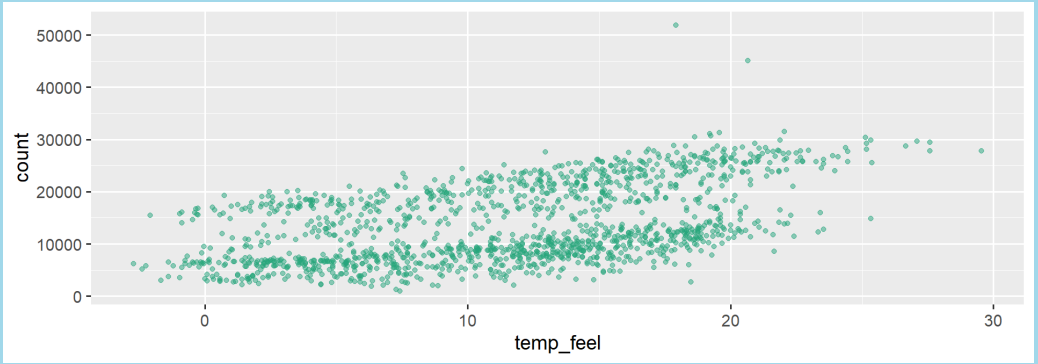
Visual Properties of Layers

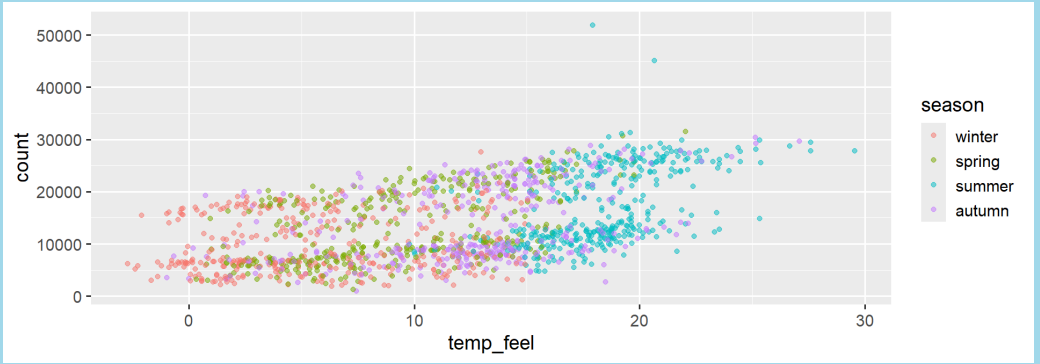
```
ggplot(
    bikes,
    aes(x = temp_feel, y = count)
  ) +
  geom_point(
    color = "#28a87d",
    alpha = .5,
    shape = "X",
    stroke = 1,
    size = 4
  )
```



Setting vs Mapping of Visual Properties

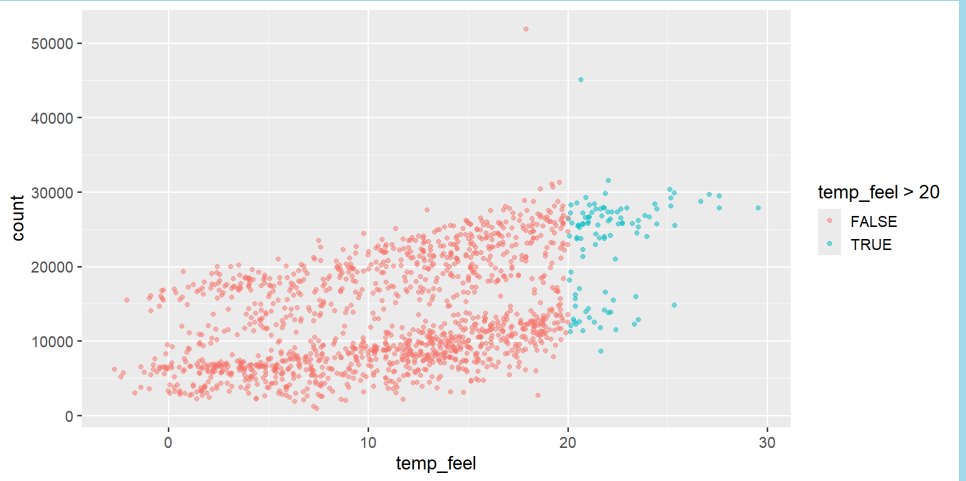
```
ggplot(
    bikes,
    aes(x = temp_feel, y = count)
  ) +
  geom_point(
    color = "#28a87d",
    alpha = .5
  )
ggplot(
    bikes,
    aes(x = temp_feel, y = count)
  ) +
  geom_point(
    aes(color = season),
    alpha = .5
  )
```


count
50000
40000
30000
20000
10000
0
0
10
20
30
temp_feel


count
50000
40000
30000
20000
10000
0
0
10
20
30
temp_feel
season
winter
spring
summer
autumn

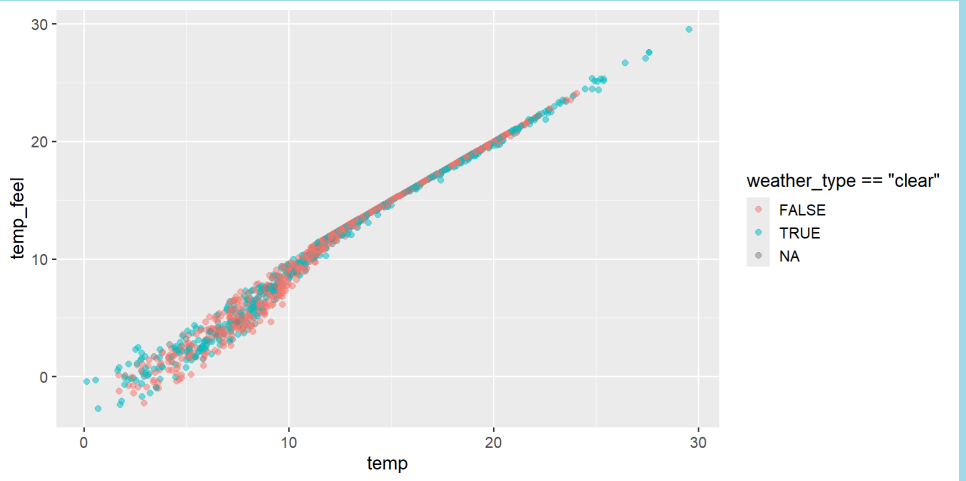
Mapping Expressions

```
1  ggplot(
2      bikes,
3      aes(x = temp_feel, y = count)
4    ) +
5    geom_point(
6      aes(color = temp_feel > 20),
7      alpha = .5
8    )
```



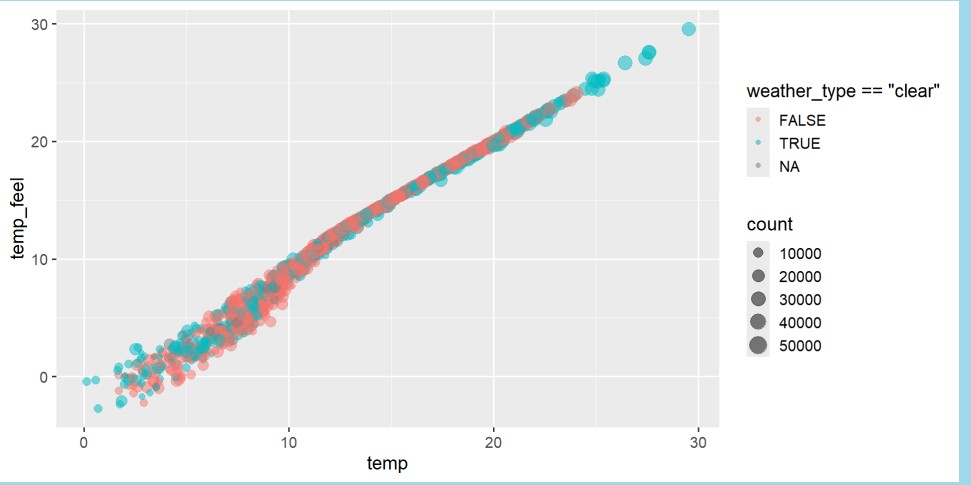
Mapping Expressions

```
ggplot(
    bikes,
    aes(x = temp, y = temp_feel)
  ) +
  geom_point(
    aes(color = weather_type == "c
    alpha = .5,
    size = 2
  )
```



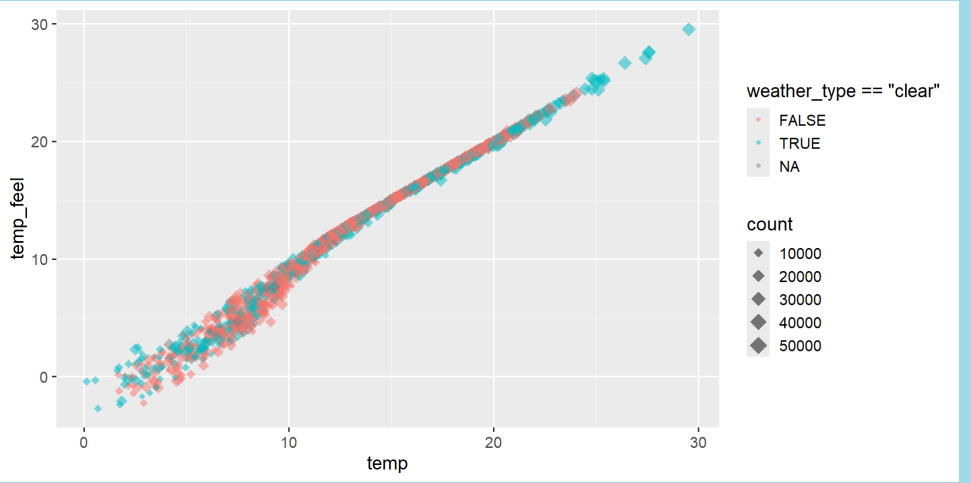
Mapping to Size

```
ggplot(
    bikes,
    aes(x = temp, y = temp_feel)
  ) +
  geom_point(
    aes(color = weather_type == "c
        size = count),
    alpha = .5
  )
```



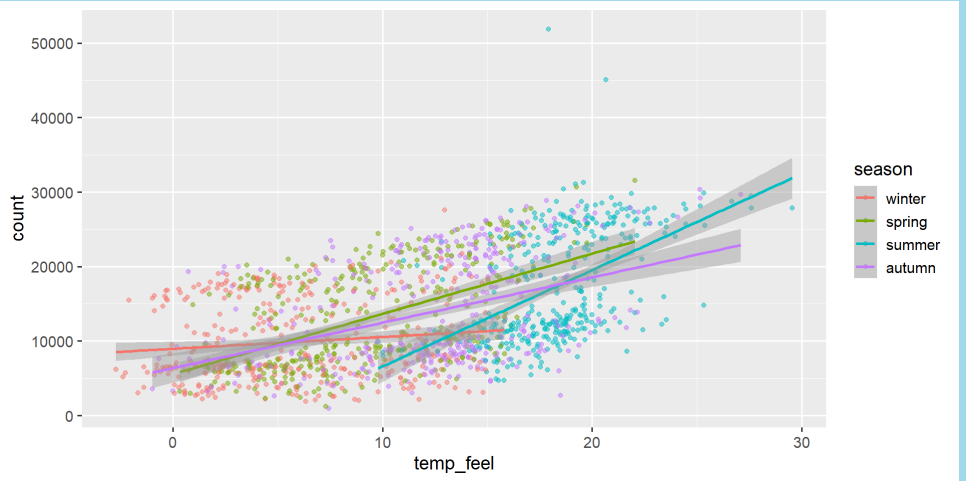
Setting a Constant Property

```
ggplot(
    bikes,
    aes(x = temp, y = temp_feel)
  ) +
  geom_point(
    aes(color = weather_type == "c
        size = count),
    shape = 18,
    alpha = .5
  )
```



Adding More Layers

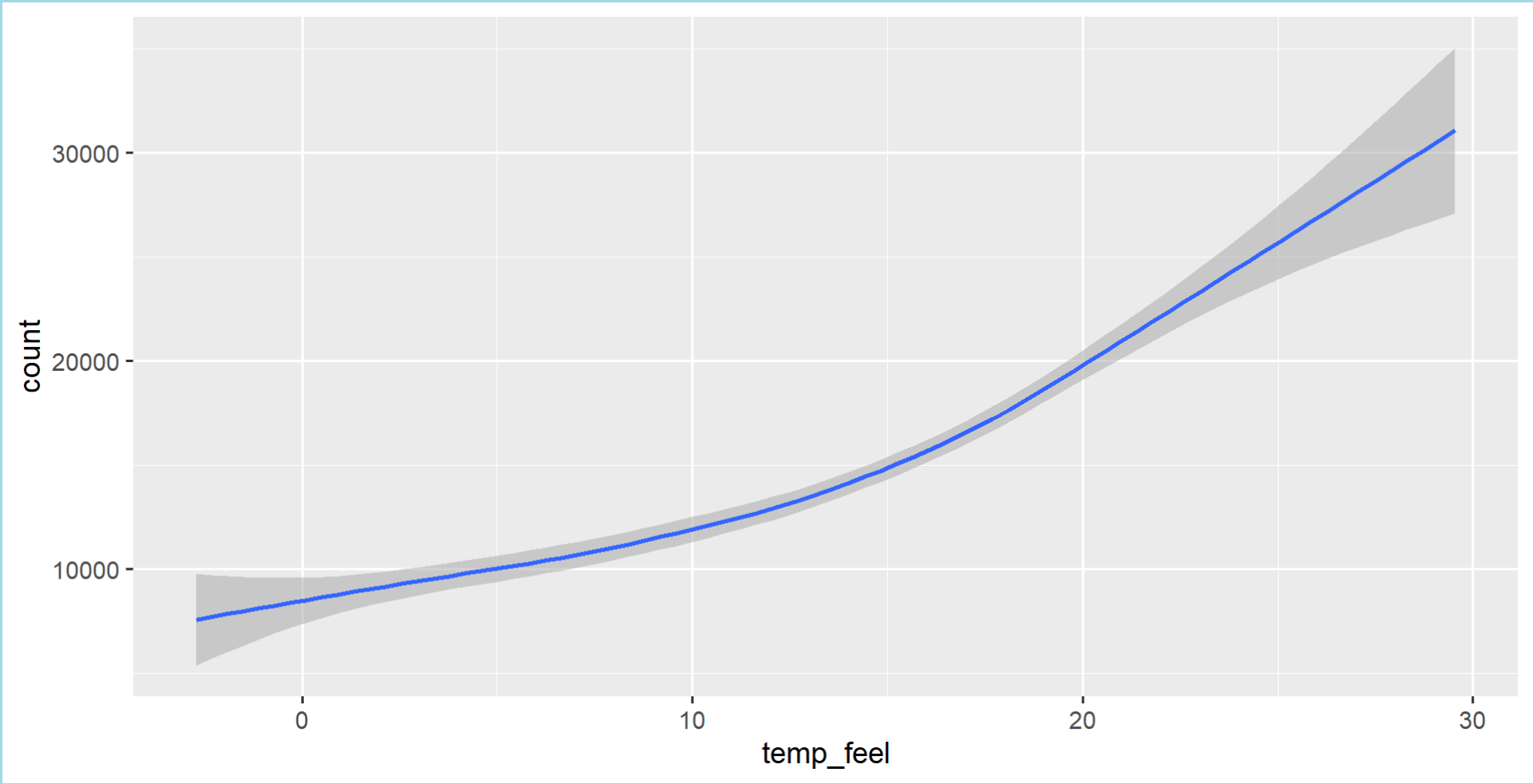
```
ggplot(
    bikes,
    aes(x = temp_feel, y = count,
        color = season)
  ) +
  geom_point(
    alpha = .5
  ) +
  geom_smooth(
    method = "lm"
  )
```



Statistical Layers

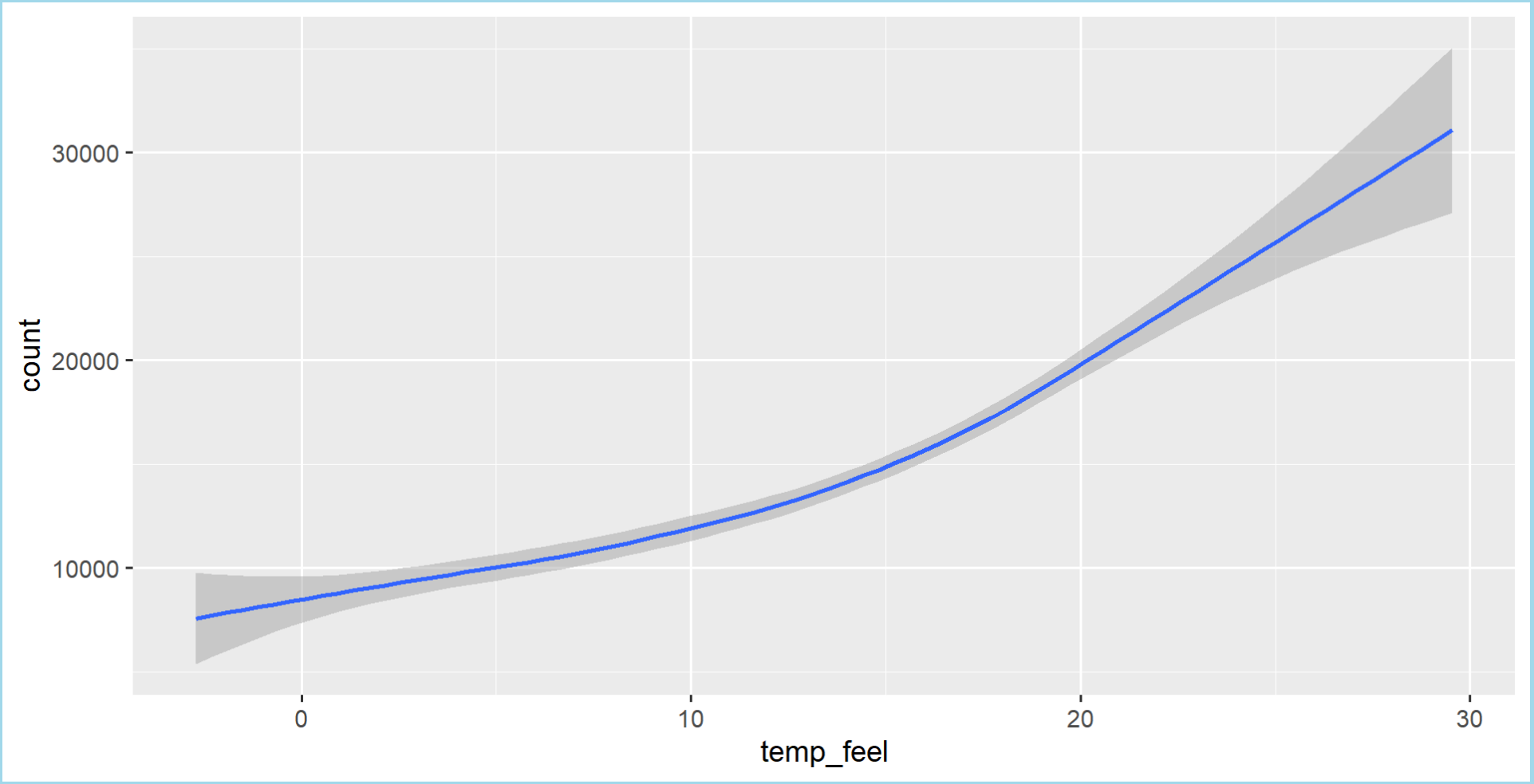
`stat_*()` and `geom_*()`

```
1  ggplot(bikes, aes(x = temp_feel, y = count)) +
2    stat_smooth(geom = "smooth")
```



`stat_*()` and `geom_*()`

```
1  ggplot(bikes, aes(x = temp_feel, y = count)) +
2    geom_smooth(stat = "smooth")
```

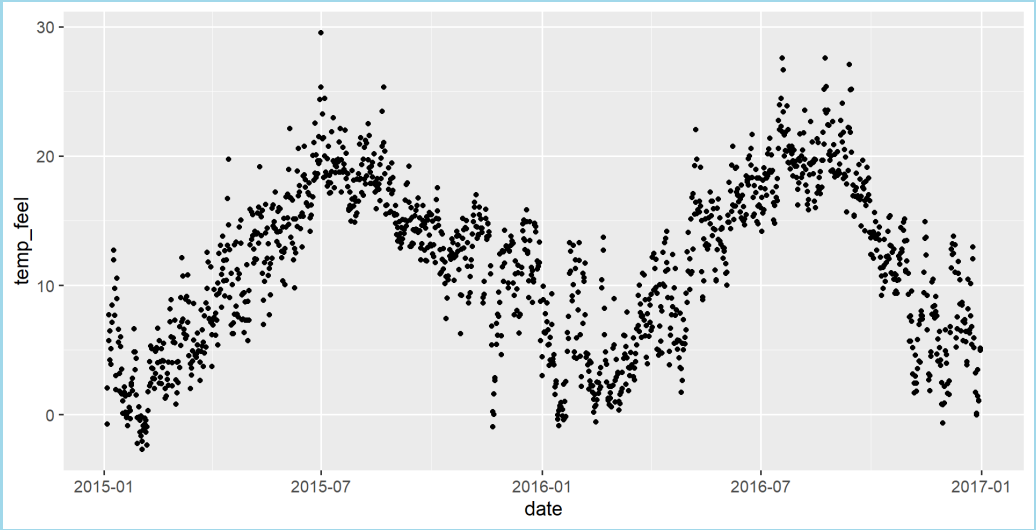


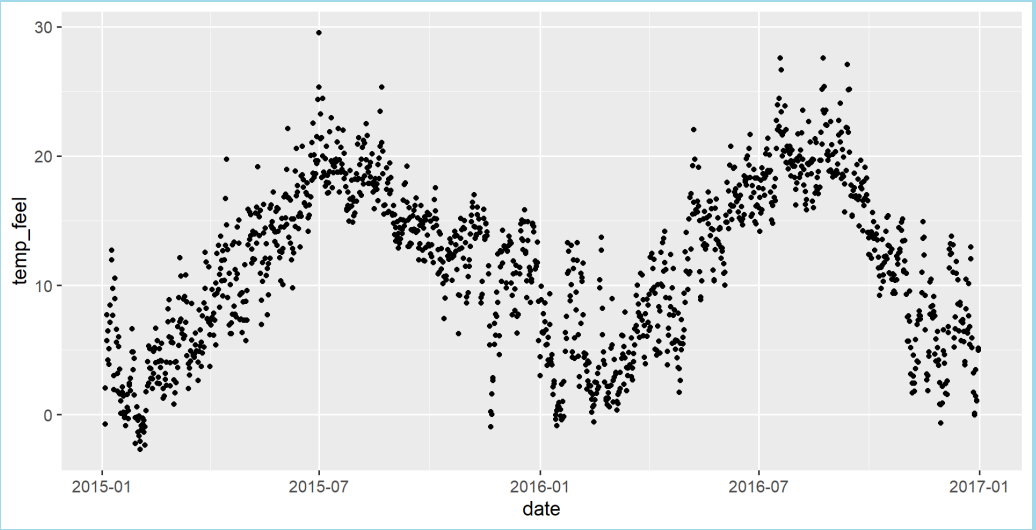
`stat_*()` and `geom_*()`

```
1 ggplot(bikes, aes(x = season)) +
2   stat_count(geom = "bar")
3 ggplot(bikes, aes(x = season)) +
4   geom_bar(stat = "count")
```

`stat_*()` and `geom_*()`

```
ggplot(bikes, aes(x = date, y = temp_feel)) +
  stat_identity(geom = "point")
ggplot(bikes, aes(x = date, y = temp_feel)) +
  geom_point(stat = "identity")
```





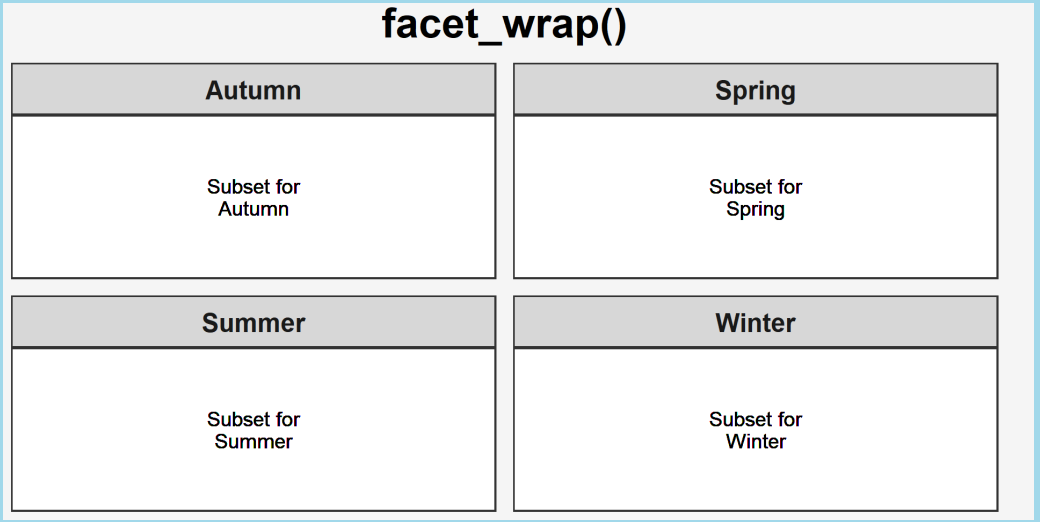
Facets

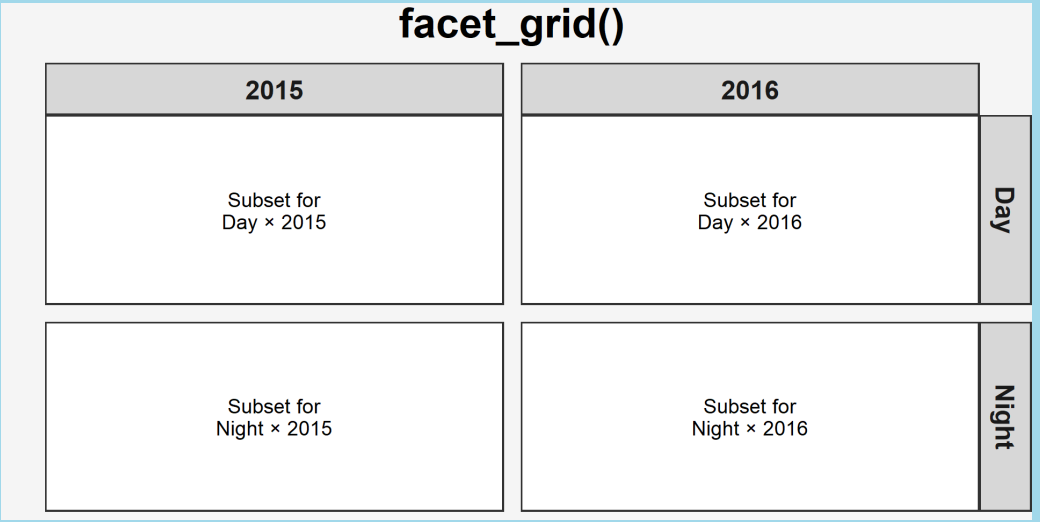
Facets

= split variables to multiple panels

Facets are also known as:

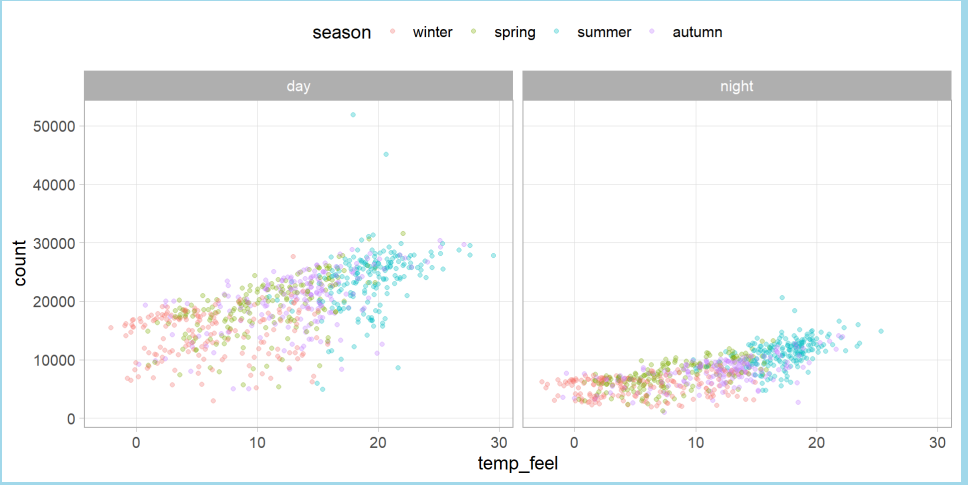
- small multiples
- trellis graphs
- lattice plots
- conditioning


facet_wrap()
Autumn
Subset for Autumn
Spring
Subset for Spring
Summer
Subset for Summer
Winter
Subset for Winter


facet_grid()
2015
2016
Day
Subset for Day × 2015
Subset for Day × 2016
Night
Subset for Night × 2015
Subset for Night × 2016

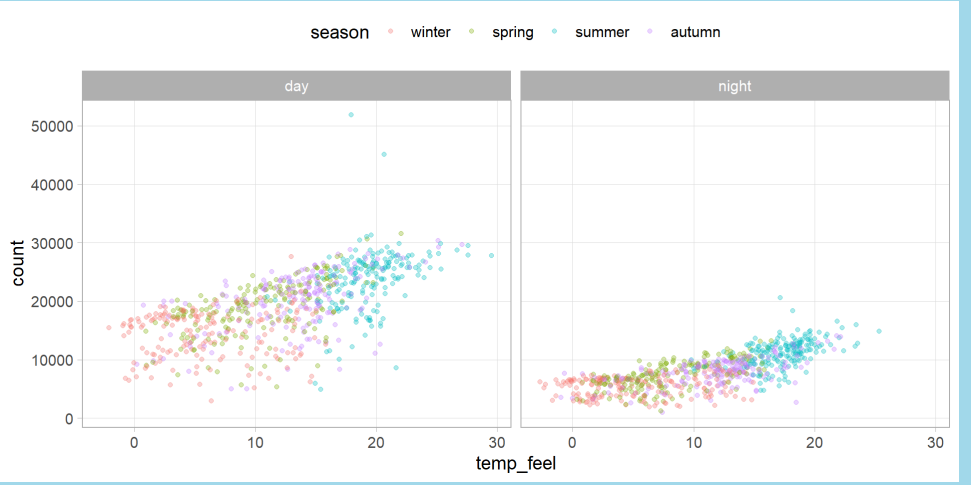
Wrapped Facets

```
g <-
  ggplot(
    bikes,
    aes(x = temp_feel, y = count,
        color = season)
  ) +
  geom_point(
    alpha = .3,
    guide = "none"
  )
g +
  facet_wrap(
    vars(day_night)
  )
```



Wrapped Facets

```
g +
  facet_wrap(
    ~ day_night
  )
```



Scales

Scales

= translate between variable ranges and property ranges

- feels-like temperature ⇄ x
- reported bike shares ⇄ y
- season ⇄ color
- year ⇄ shape
- …

Scales

The `scale_*()` components control the properties of all the **aesthetic dimensions mapped to the data**.

Consequently, there are `scale_*()` functions for all aesthetics such as:

Scales

The `scale_*()` components control the properties of all the **aesthetic dimensions mapped to the data**.

The extensions (*) can be filled by e.g.:

- `continuous()`, `discrete()`, `reverse()`, `log10()`, `sqrt()`, `date()` for positions
- `continuous()`, `discrete()`, `manual()`, `gradient()`, `gradient2()`, `brewer()` for colors
- `continuous()`, `discrete()`, `manual()`, `ordinal()`, `area()`, `date()` for sizes
- `continuous()`, `discrete()`, `manual()`, `ordinal()` for shapes
- `continuous()`, `discrete()`, `manual()`, `ordinal()`, `date()` for transparency

Continuous vs. Discrete in {ggplot2}

Continuous:
quantitative or numerical data

- height
- weight
- age
- counts

Discrete:
qualitative or categorical data

- species
- sex
- study sites
- age group

Continuous vs. Discrete in {ggplot2}

Continuous:
quantitative or numerical data

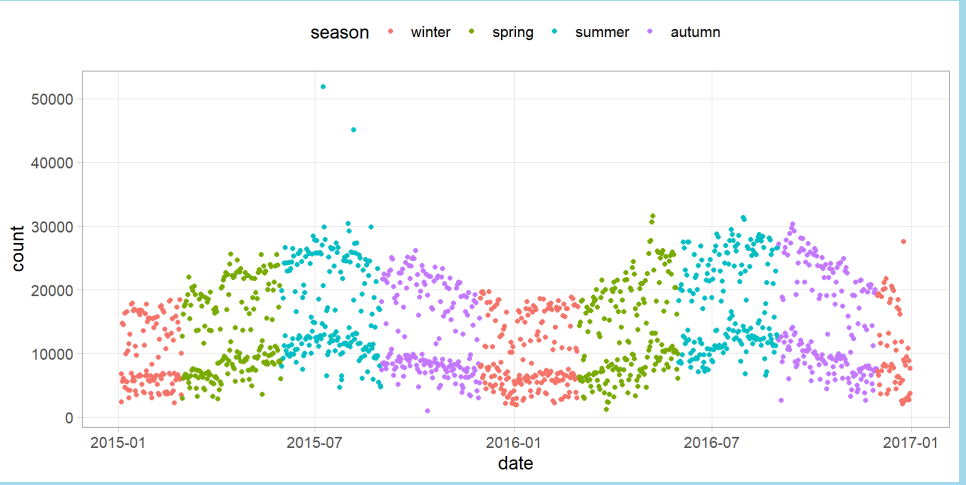
- height (continuous)
- weight (continuous)
- age (continuous or discrete)
- counts (discrete)

Discrete:
qualitative or categorical data

- species (nominal)
- sex (nominal)
- study site (nominal or ordinal)
- age group (ordinal)

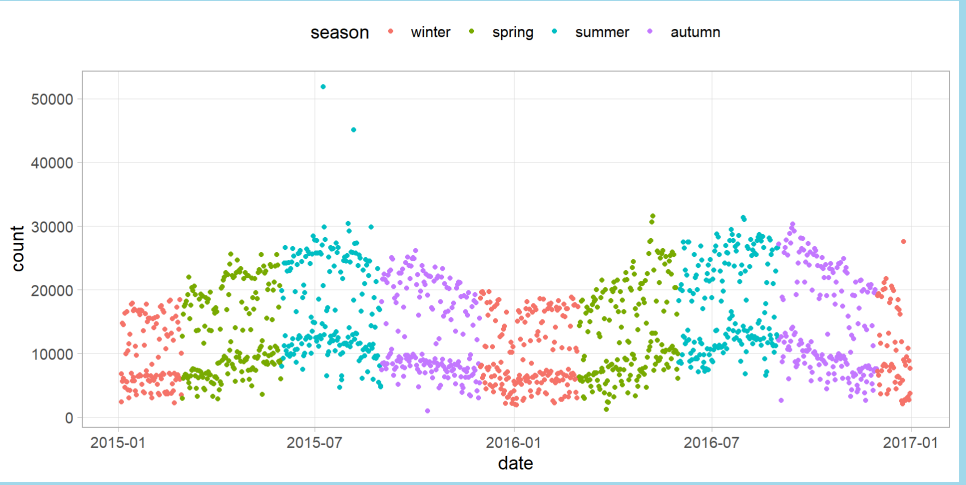
Aesthetics + Scales

```
ggplot(
    bikes,
    aes(x = date, y = count,
        color = season)
  ) +
  geom_point()
```



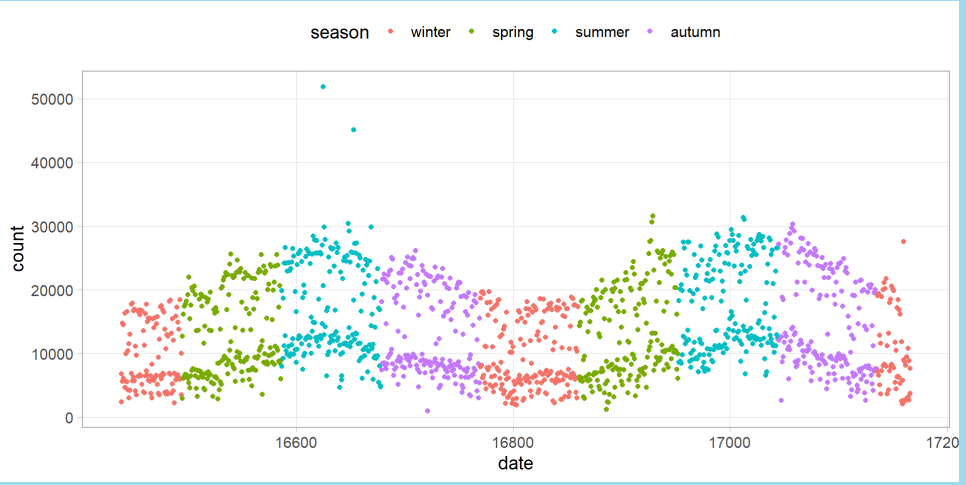
Aesthetics + Scales

```
ggplot(
    bikes,
    aes(x = date, y = count,
        color = season)
  ) +
  geom_point() +
  scale_x_date() +
  scale_y_continuous() +
  scale_color_discrete()
```



Scales

```
ggplot(
    bikes,
    aes(x = date, y = count,
        color = season)
  ) +
  geom_point() +
  scale_x_continuous() +
  scale_y_continuous() +
  scale_color_discrete()
```



Coordinate Systems

= interpret the position aesthetics

- **linear coordinate systems:** preserve the geometrical shapes
  - `coord_cartesian()`
  - `coord_fixed()`
  - `coord_flip()`
- **non-linear coordinate systems:** likely change the geometrical shapes
  - `coord_polar()`
  - `coord_map()` and `coord_sf()`
  - `coord_trans()`