Data Visualization and Maps II

HES 505 Fall 2023: Session 30

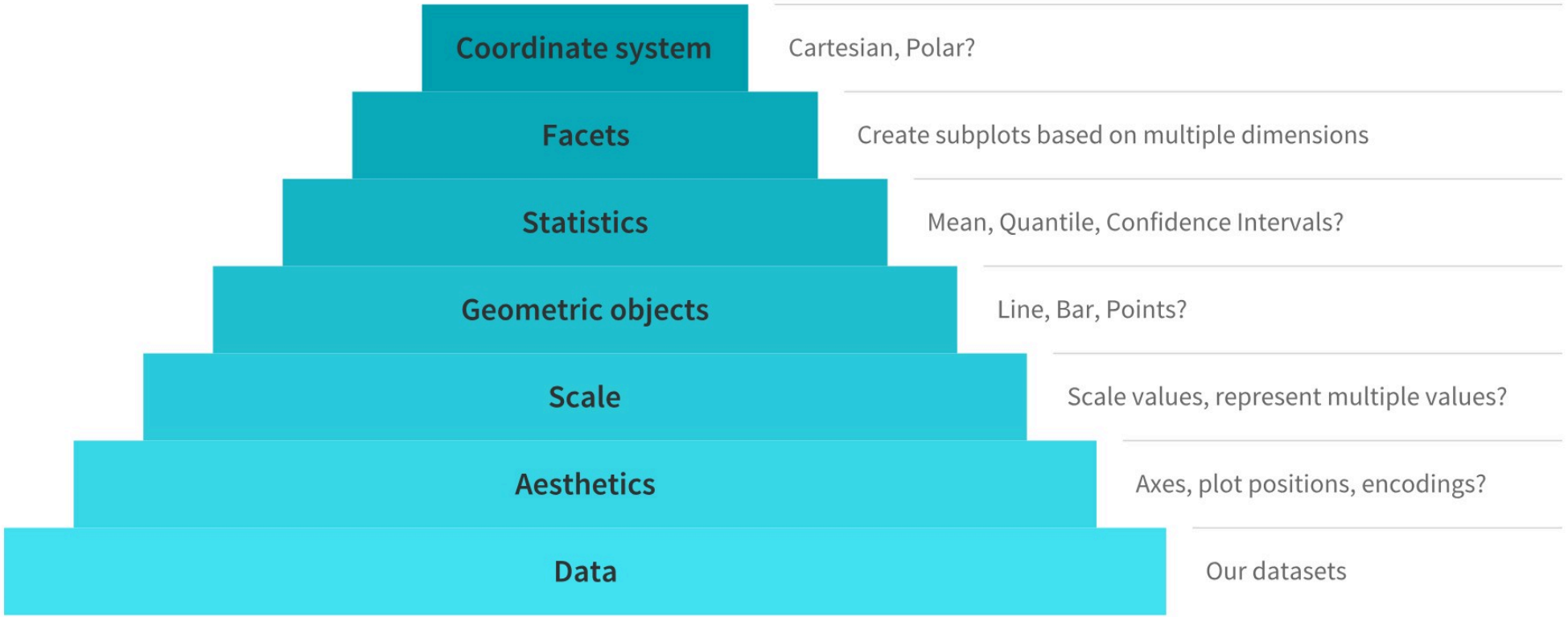
Matt Williamson

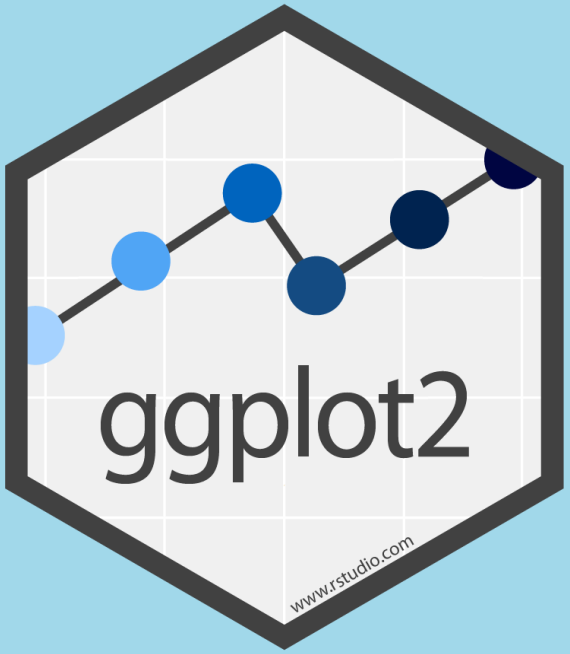
Objectives

By the end of today you should be able to: * Understand the relationship between the Grammar of Graphics and `ggplot` syntax

- Describe the various options for customizing `ggplots` and their syntactic conventions
- Generate complicated plot layouts without additional pre-processing
- Construct a map using `ggplot2` and `tmap`
- Combine vector and raster data in the same map

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)
```

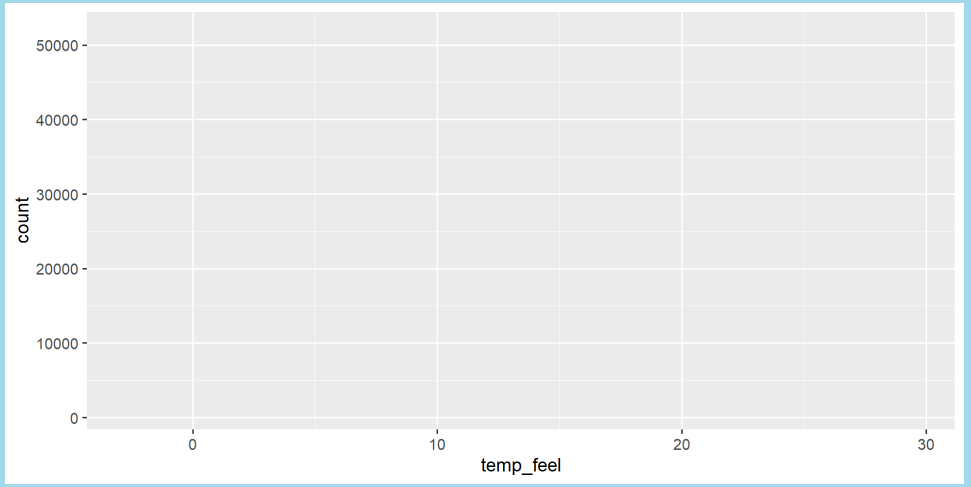
Aesthetic Mapping

= 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

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

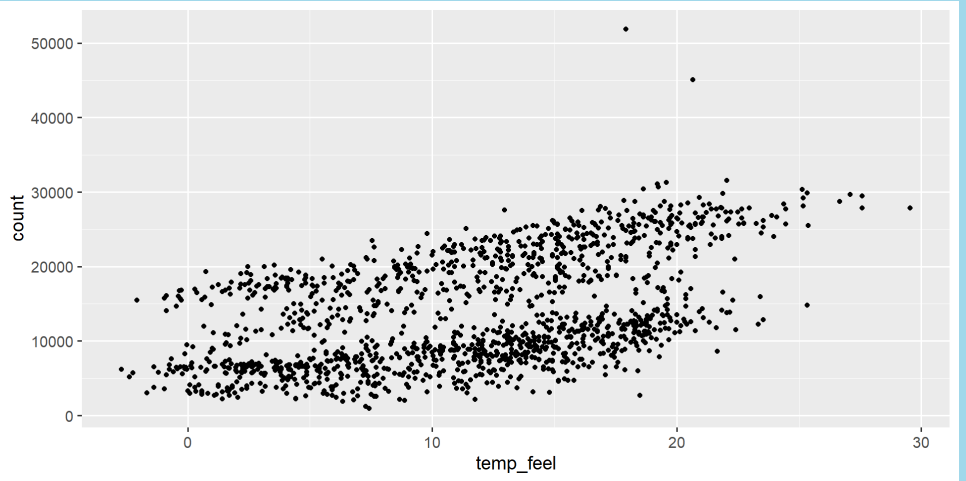
Geometries

= interpret aesthetics as graphical representations

- points
- lines
- polygons
- text labels
- …

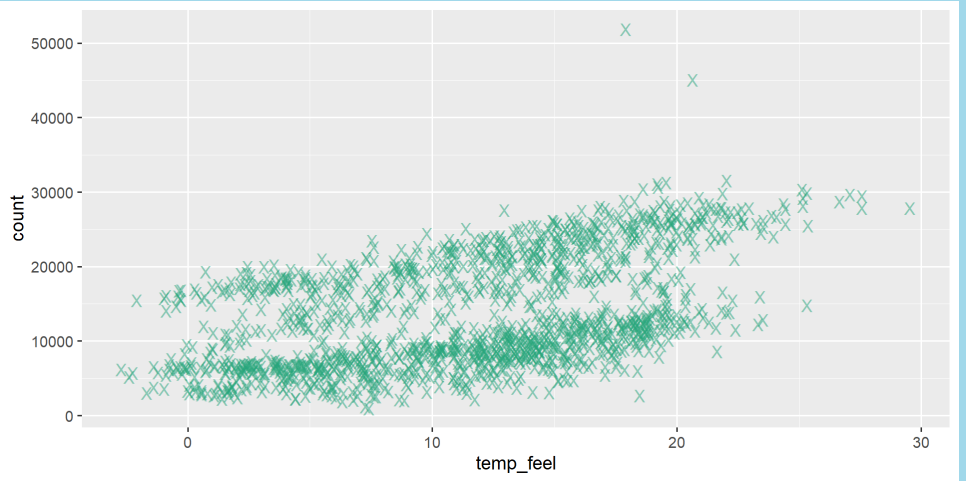
Geometries

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



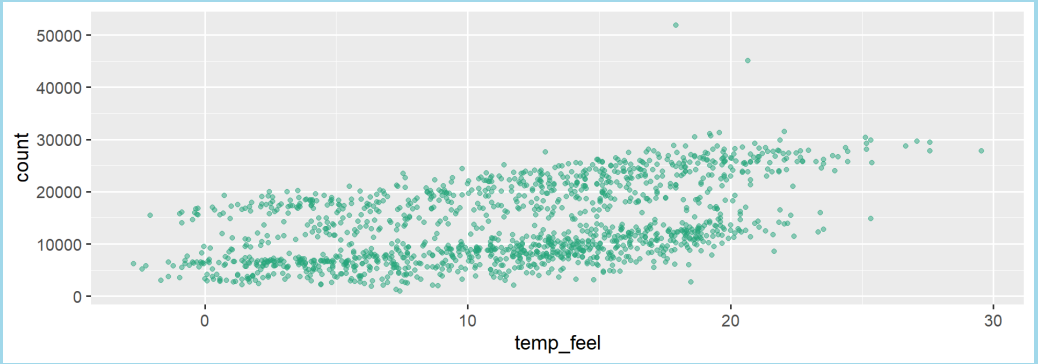
Visual Properties of Layers

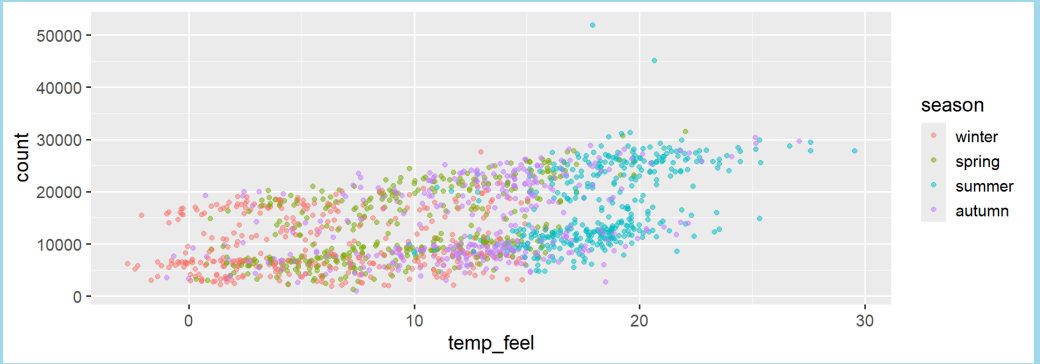
```
ggplot(
    bikes,
    aes(x = temp_feel, y = count)
  ) +
  geom_point(
    color = "#28a87d",
    alpha = .5,
    shape = "X",
    stroke = 1,
    size = 4
  )
```



Setting vs Mapping of Visual Properties

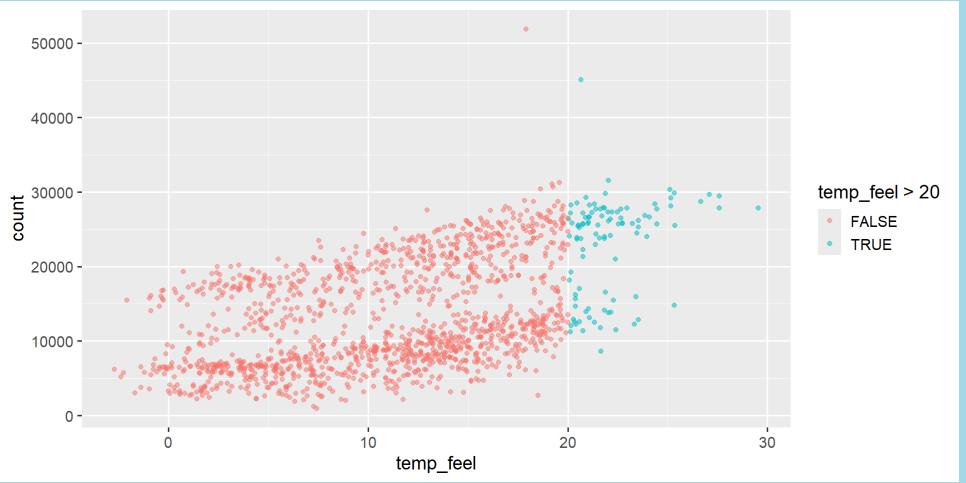
```
ggplot(
    bikes,
    aes(x = temp_feel, y = count)
  ) +
  geom_point(
    color = "#28a87d",
    alpha = .5
  )
ggplot(
    bikes,
    aes(x = temp_feel, y = count)
  ) +
  geom_point(
    aes(color = season),
    alpha = .5
  )
```


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

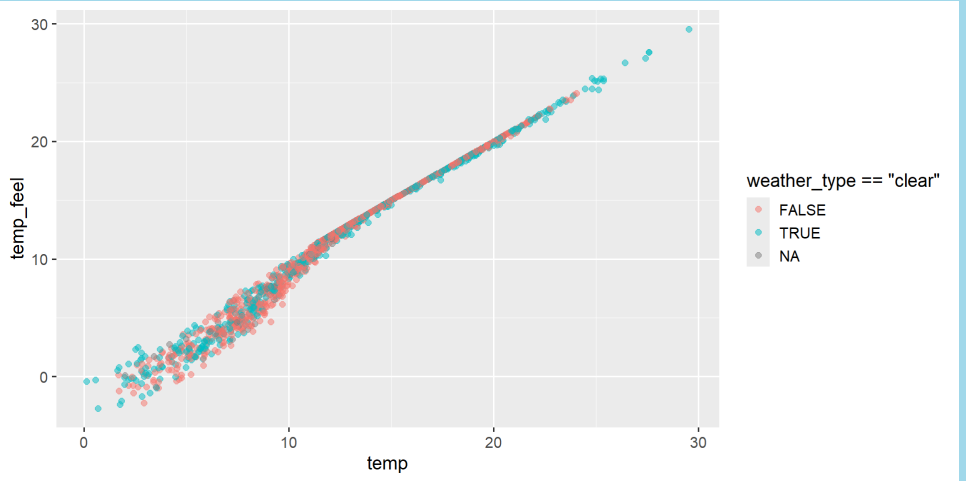
Mapping Expressions

```
ggplot(
    bikes,
    aes(x = temp_feel, y = count)
  ) +
  geom_point(
    aes(color = temp_feel > 20),
    alpha = .5
  )
```



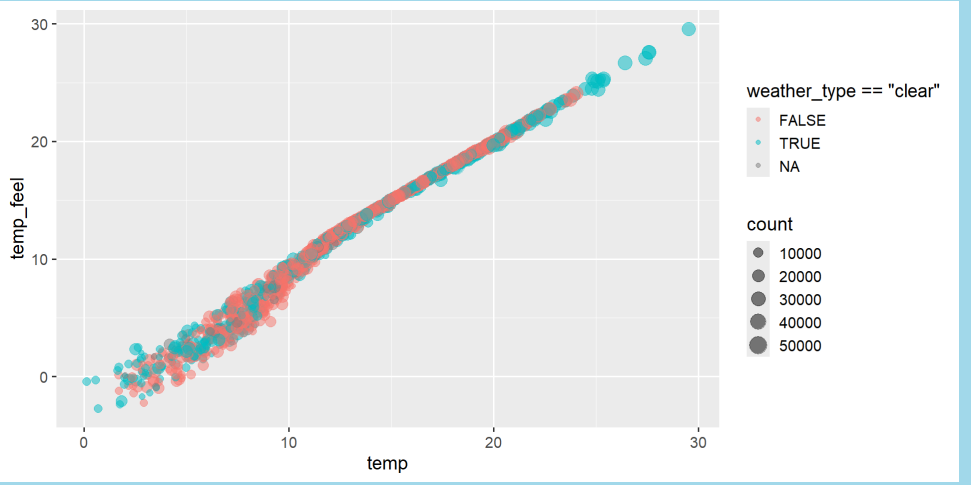
Mapping Expressions

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



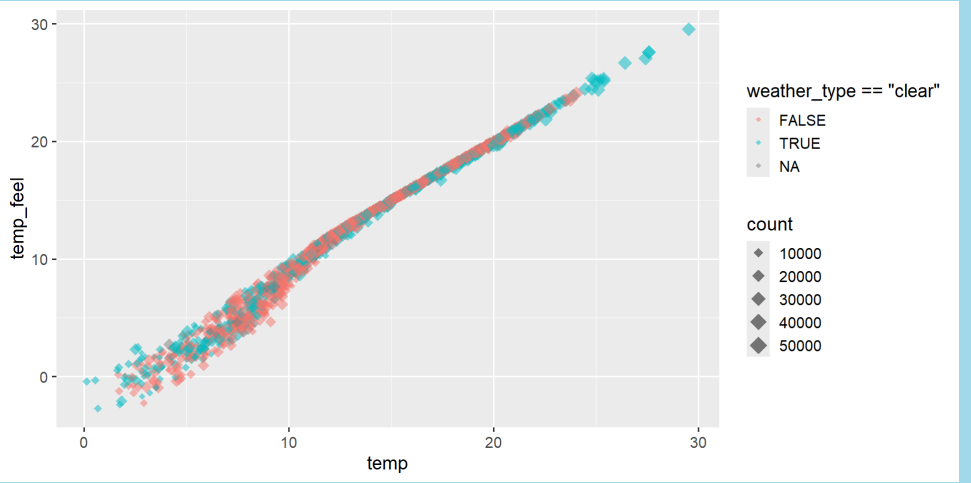
Mapping to Size

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



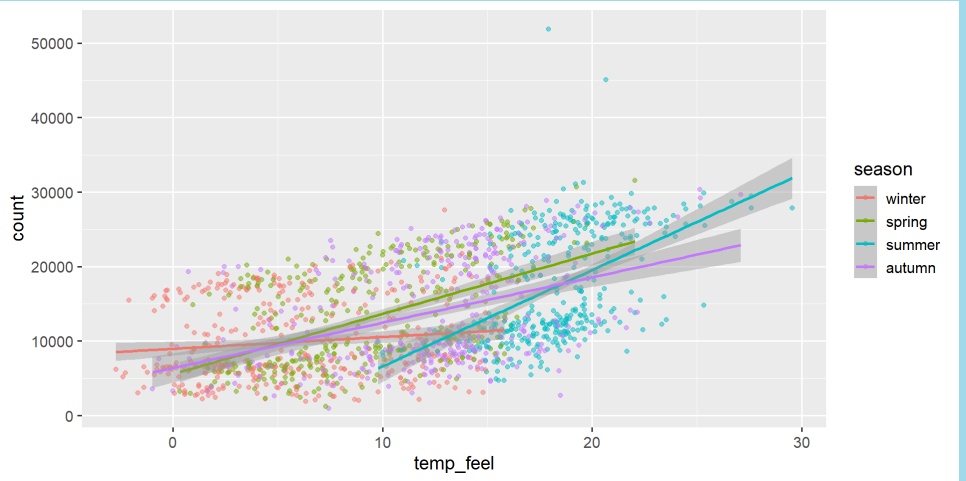
Setting a Constant Property

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



Adding More Layers

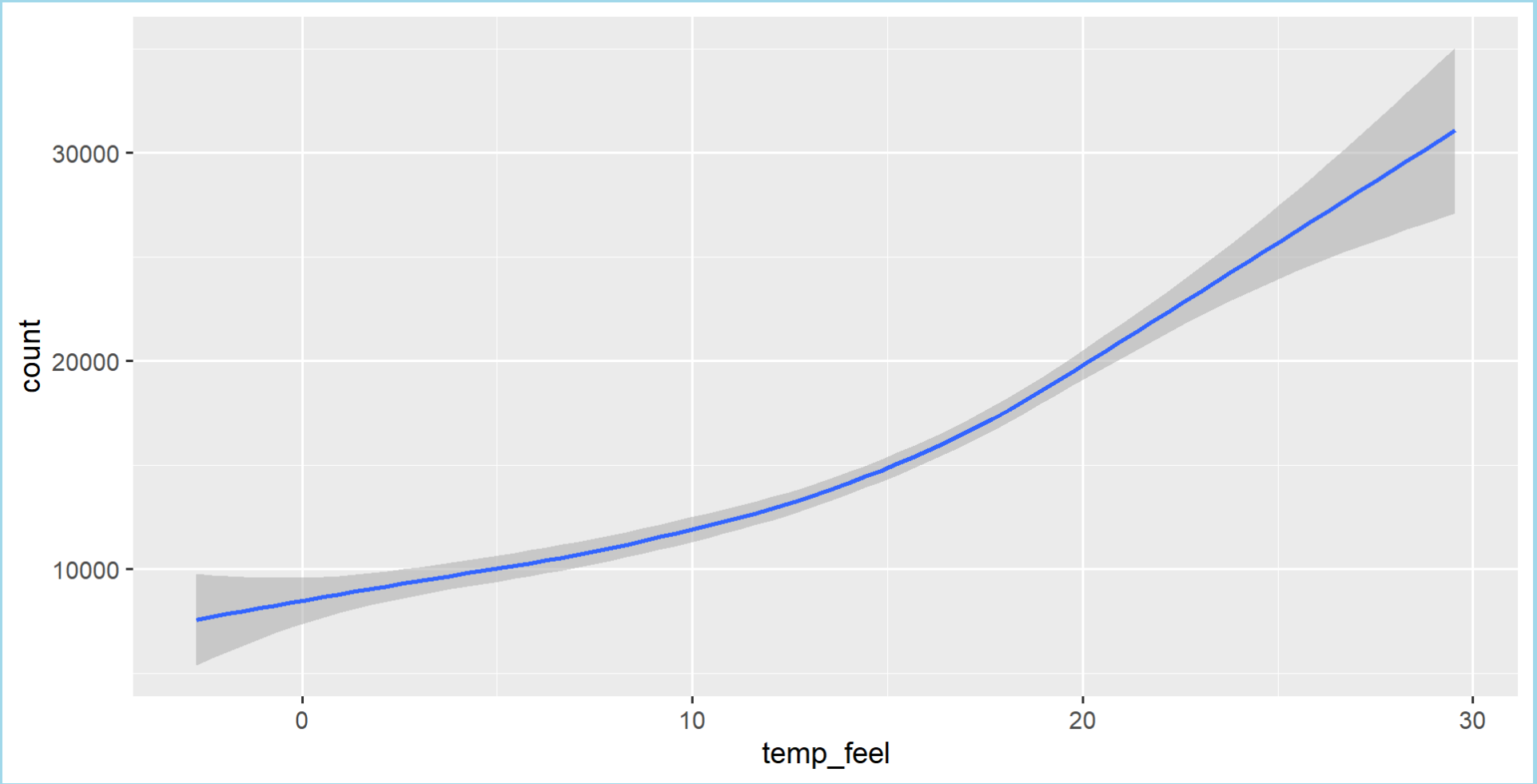
```
ggplot(
    bikes,
    aes(x = temp_feel, y = count,
        color = season)
  ) +
  geom_point(
    alpha = .5
  ) +
  geom_smooth(
    method = "lm"
  )
```



Statistical Layers

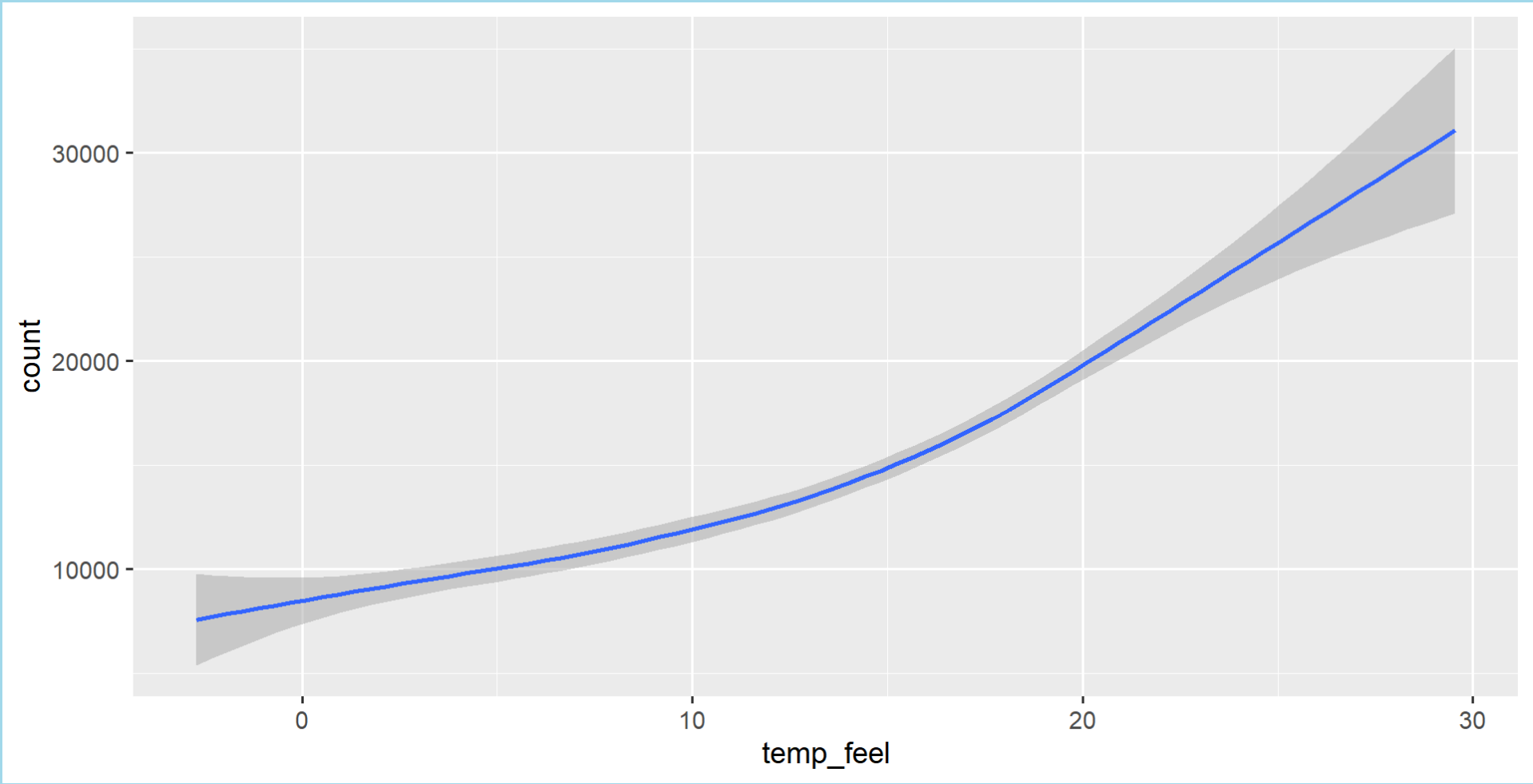
`stat_*()` and `geom_*()`

```
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")
```

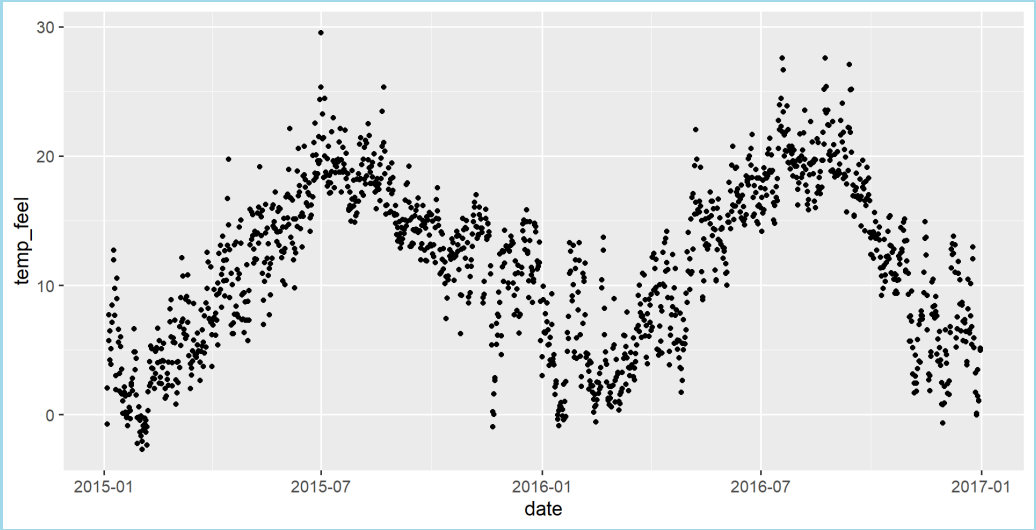


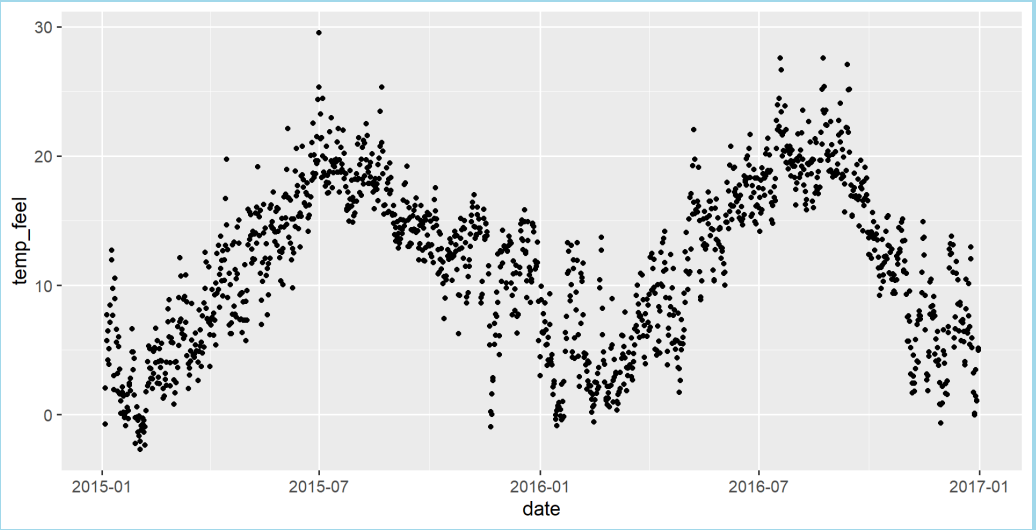
`stat_*()` and `geom_*()`

```
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_*()`

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





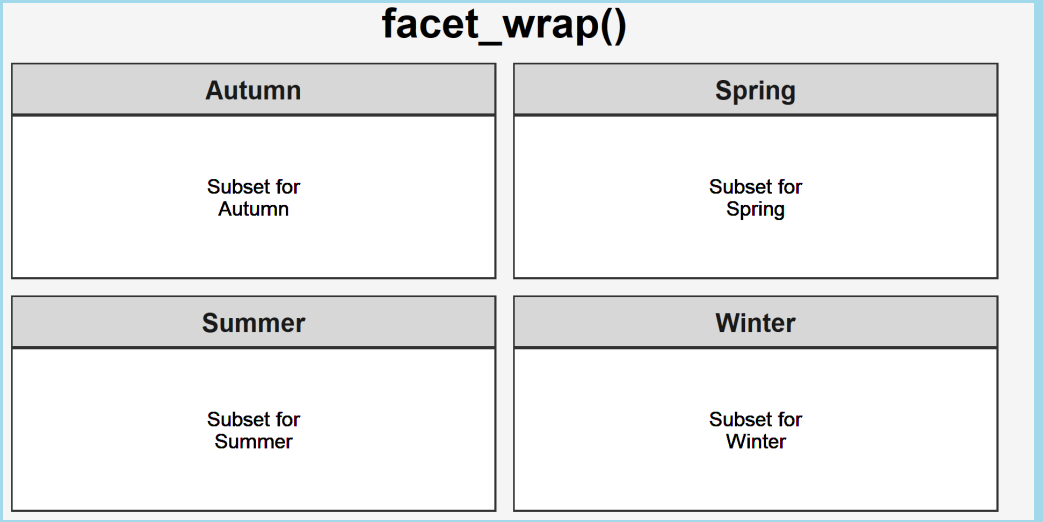
Facets

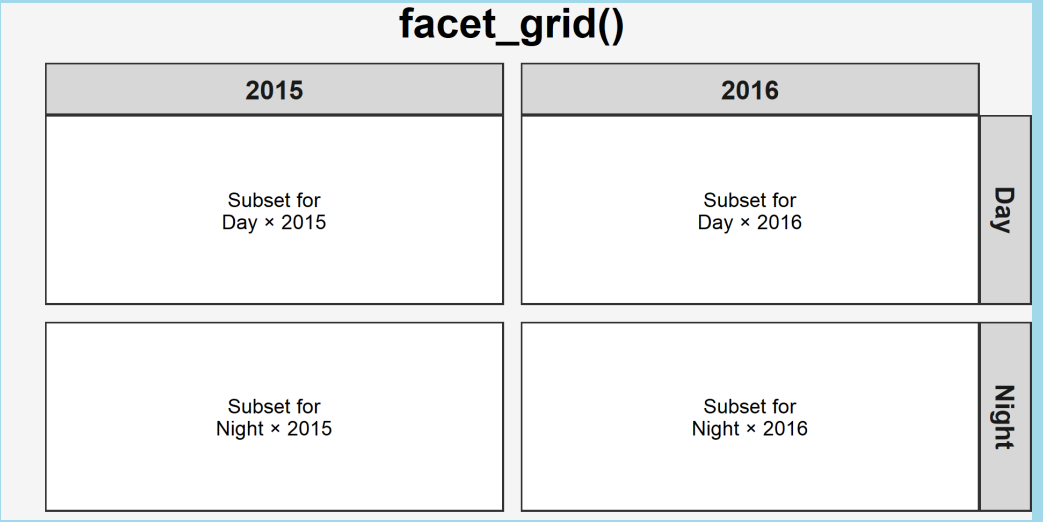
Facets

= split variables to multiple panels

Facets are also known as:

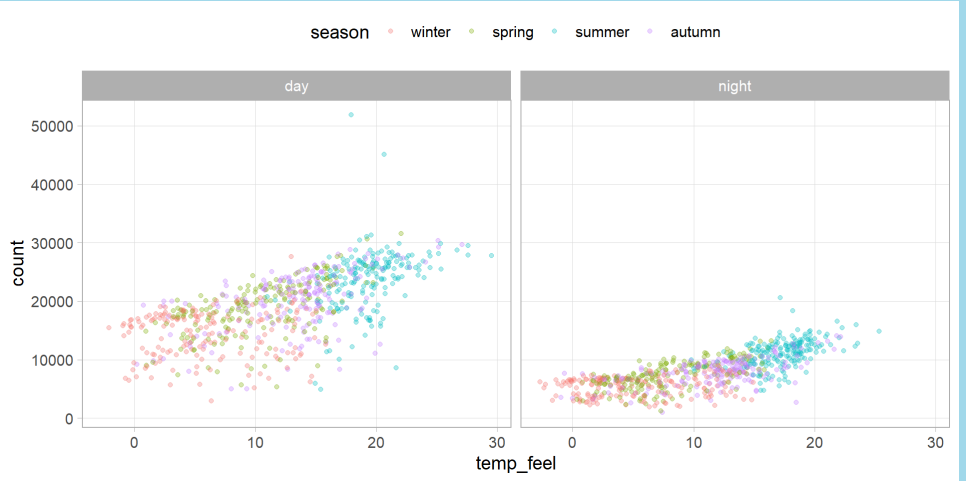
- small multiples
- trellis graphs
- lattice plots
- conditioning


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

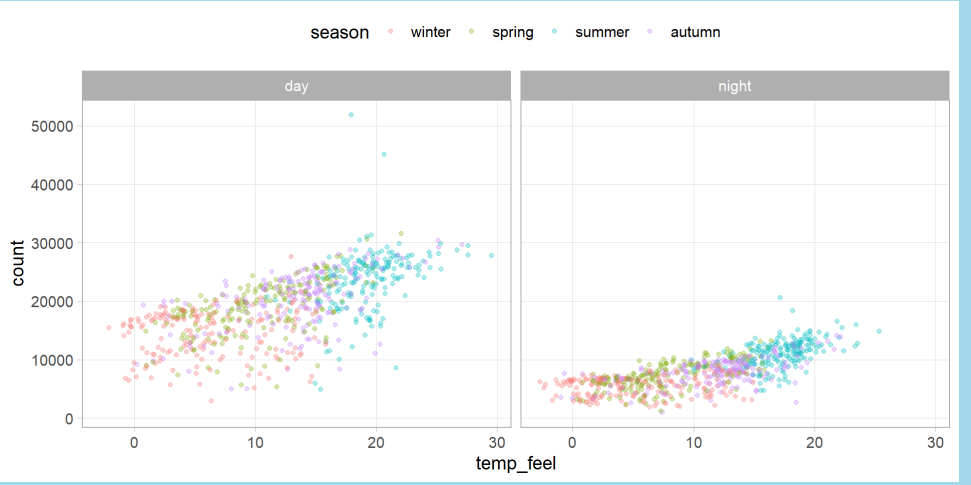
Wrapped Facets

```
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

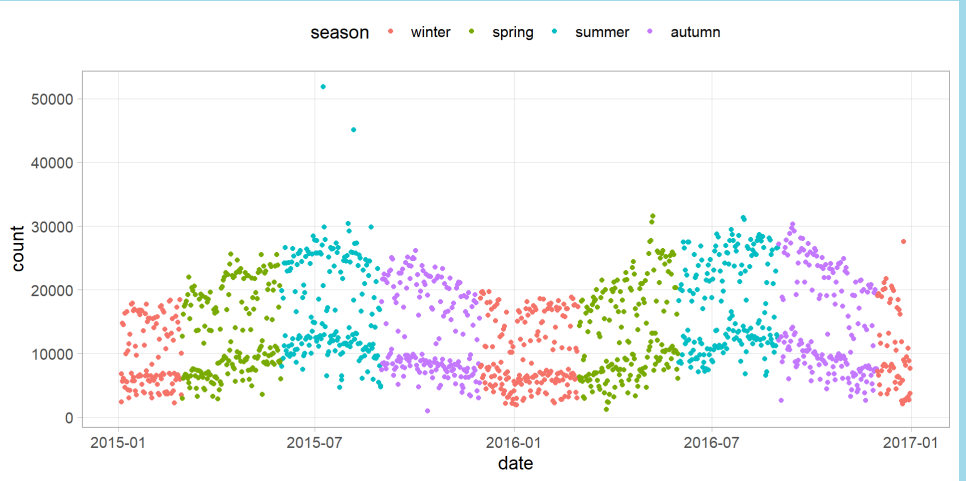
- height (continuous)
- weight (continuous)
- age (continuous or discrete)
- counts (discrete)

Discrete: qualitative or categorical data

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

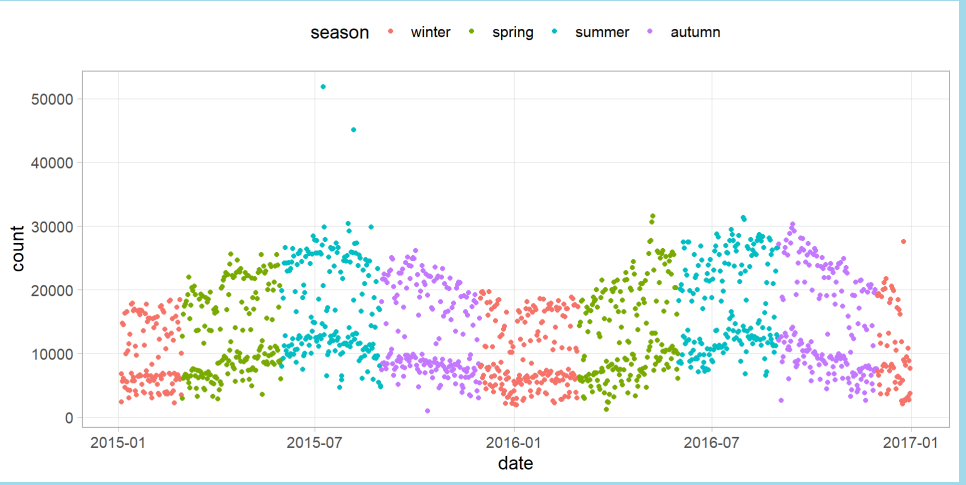
Aesthetics + Scales

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



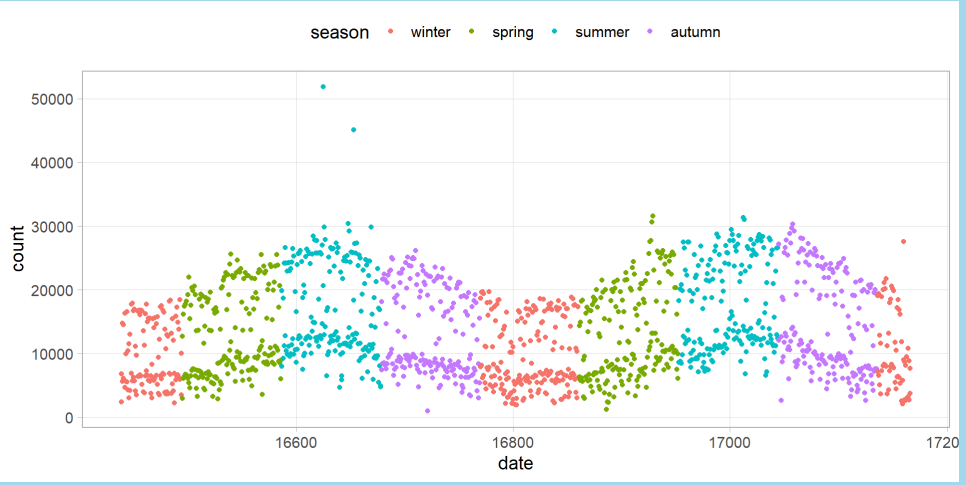
Aesthetics + Scales

```
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()`

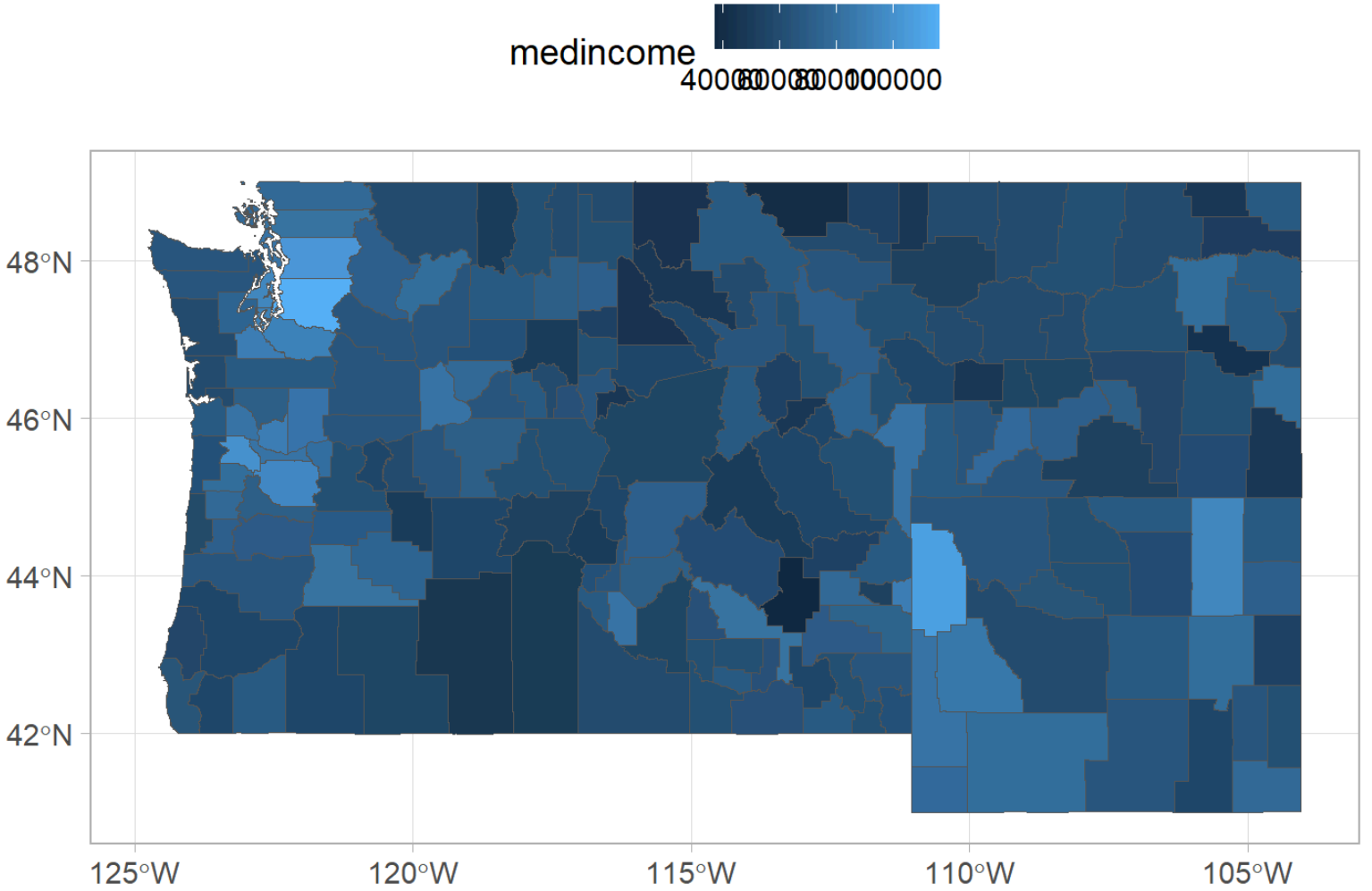
Building Choropleth Maps

Using ggplot2

```
cty.info <- get_acs(geography = "county",
                        variables = c(pop="B01003_001",
                                      medincome = "B19013_001"),
                        survey="acs5",
                        state = c("WA", "OR", "ID", "MT", "WY"),
                        geometry = TRUE, key = censkey, progress_bar=FALSE) %
  select(., -moe) %>%
  pivot_wider(
    names_from = "variable",
    values_from = "estimate"
  )

p <- ggplot(data=cty.info) +
  geom_sf(mapping=aes(fill=medincome))
```

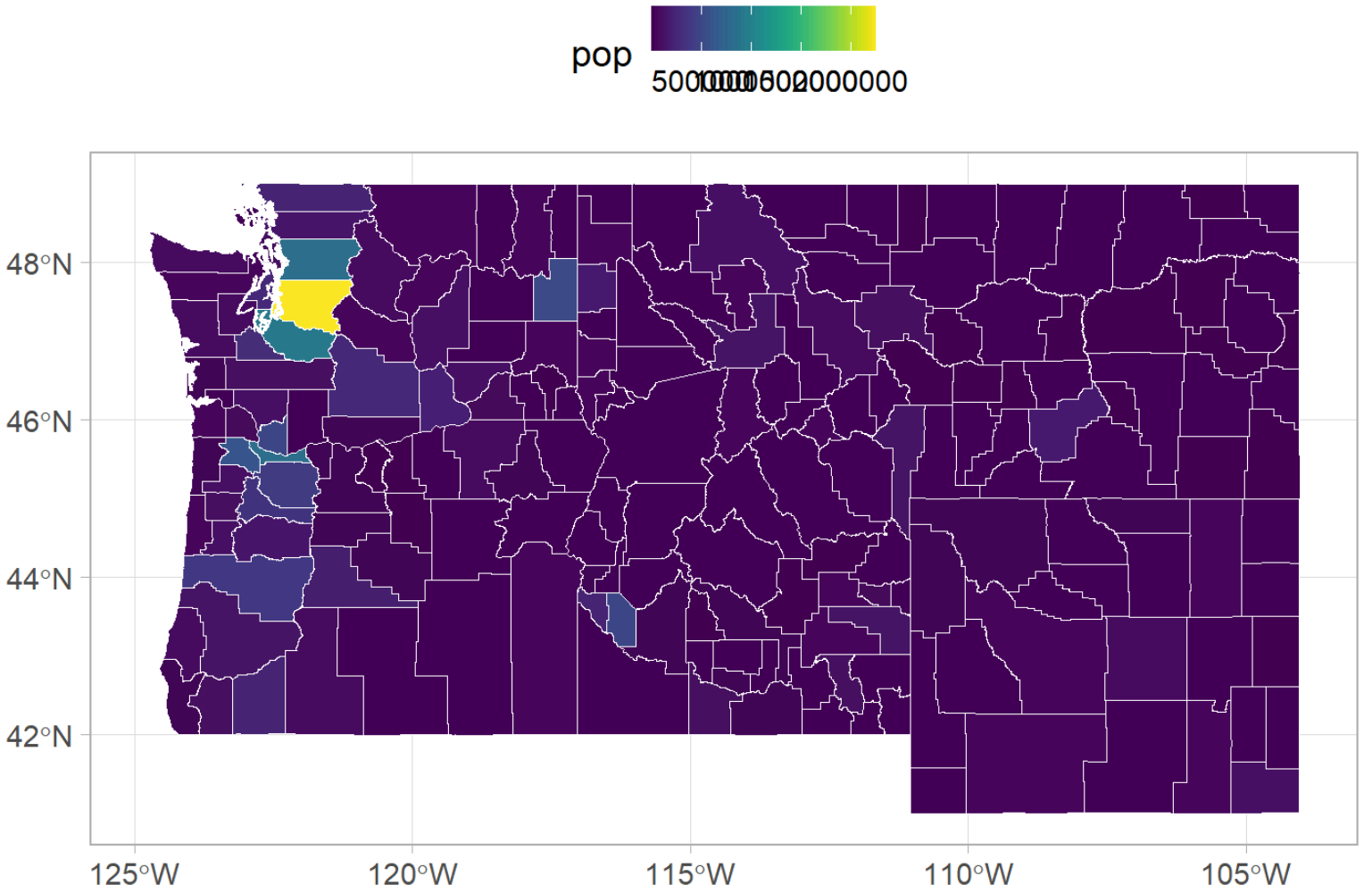
Static Maps with ggplot2



Changing aesthetics

```
1 p <- ggplot(data=cty.info) +
2   geom_sf(mapping=aes(fill=pop), color="white") +
3   scale_fill_viridis()
```

Changing aesthetics

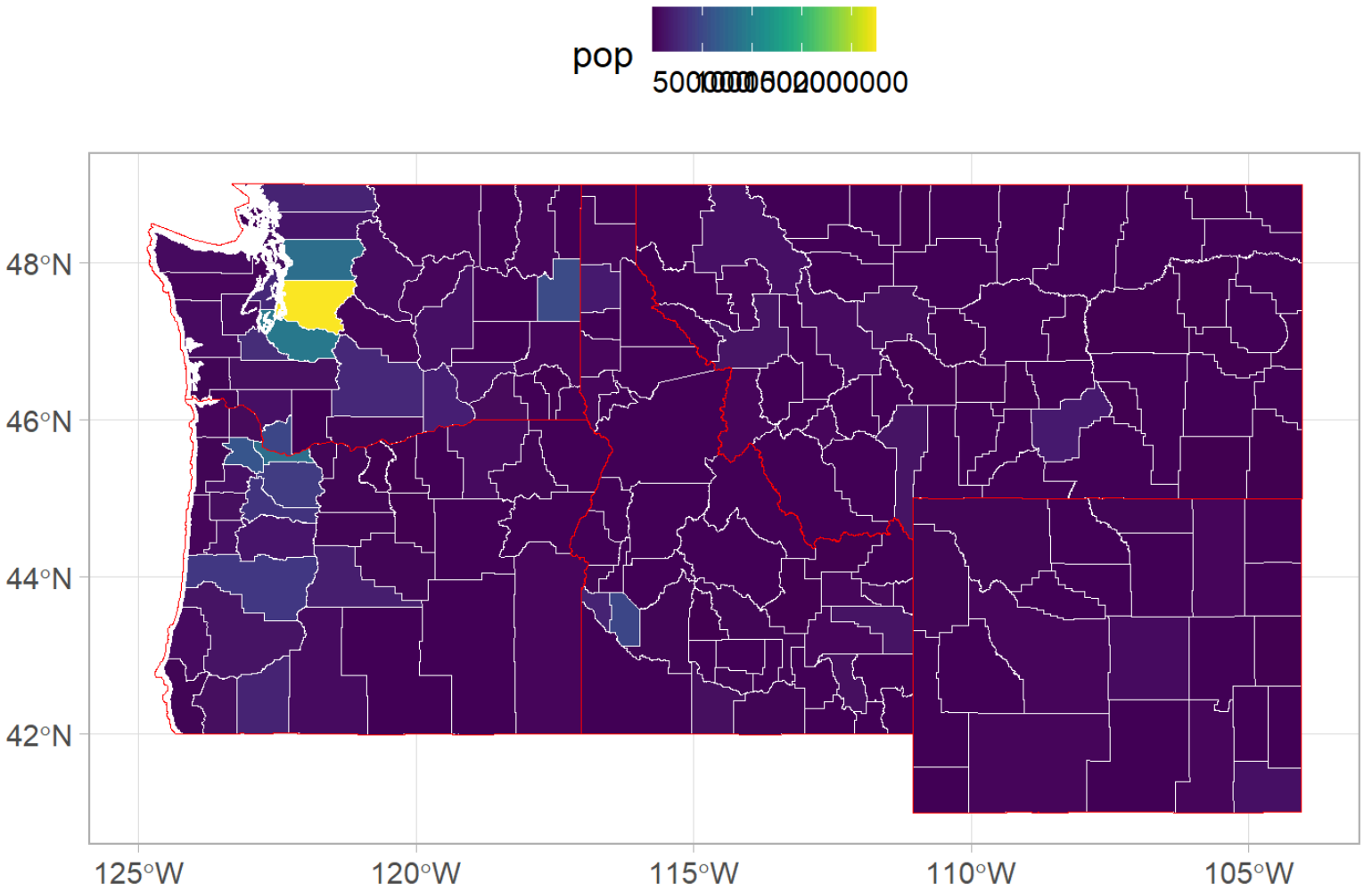


Adding layers

```
st <- tigris::states(progress_bar=FALSE) %>%
  filter(., STUSPS %in% c("WA", "OR", "ID", "MT", "WY"))

p <- ggplot(data=cty.info) +
  geom_sf(mapping=aes(fill=pop), color="white") +
  geom_sf(data=st, fill=NA, color="red") +
  scale_fill_viridis()
```

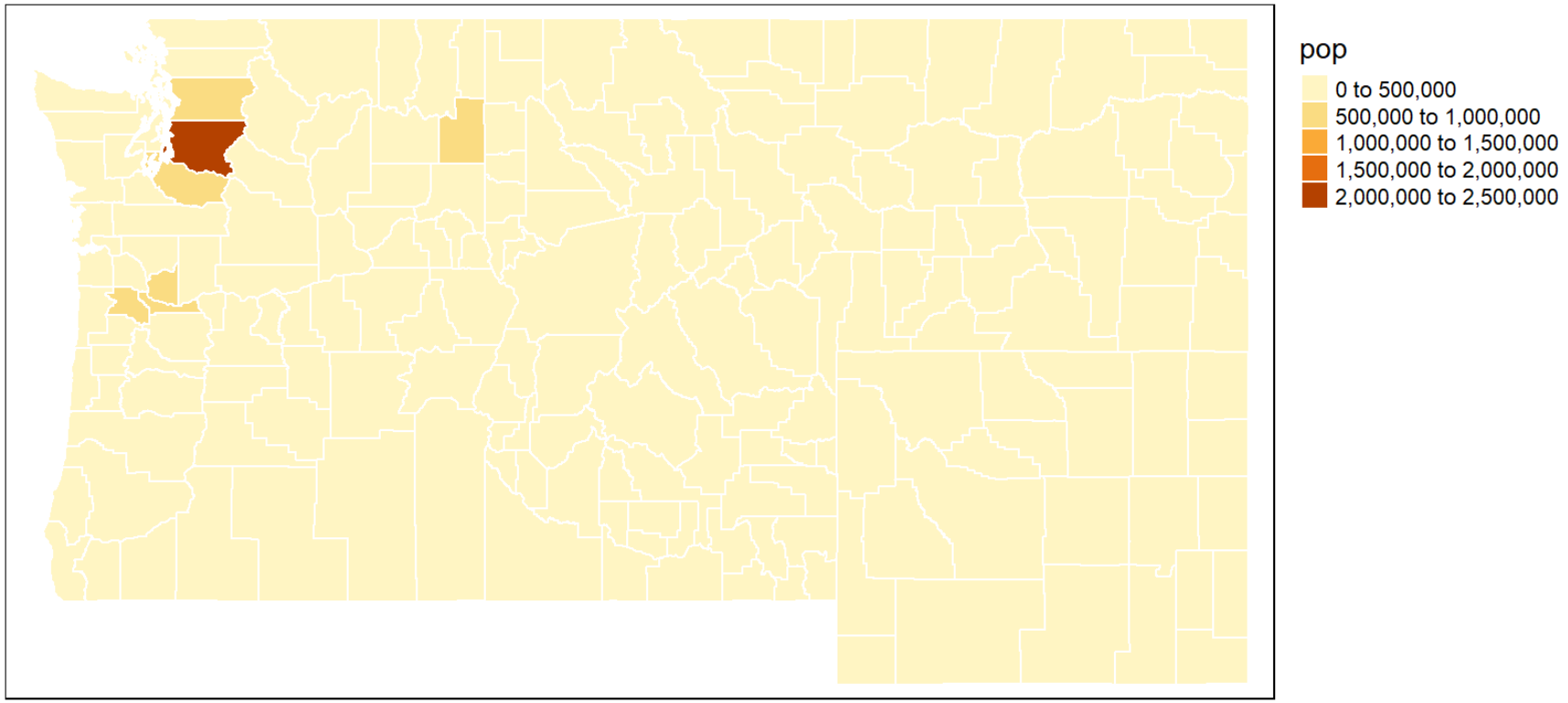
Adding layers



Using tmap

```
pt <- tm_shape(cty.info) +
  tm_polygons(col = "pop",
              border.col = "white") +
  tm_legend(outside = TRUE)
```

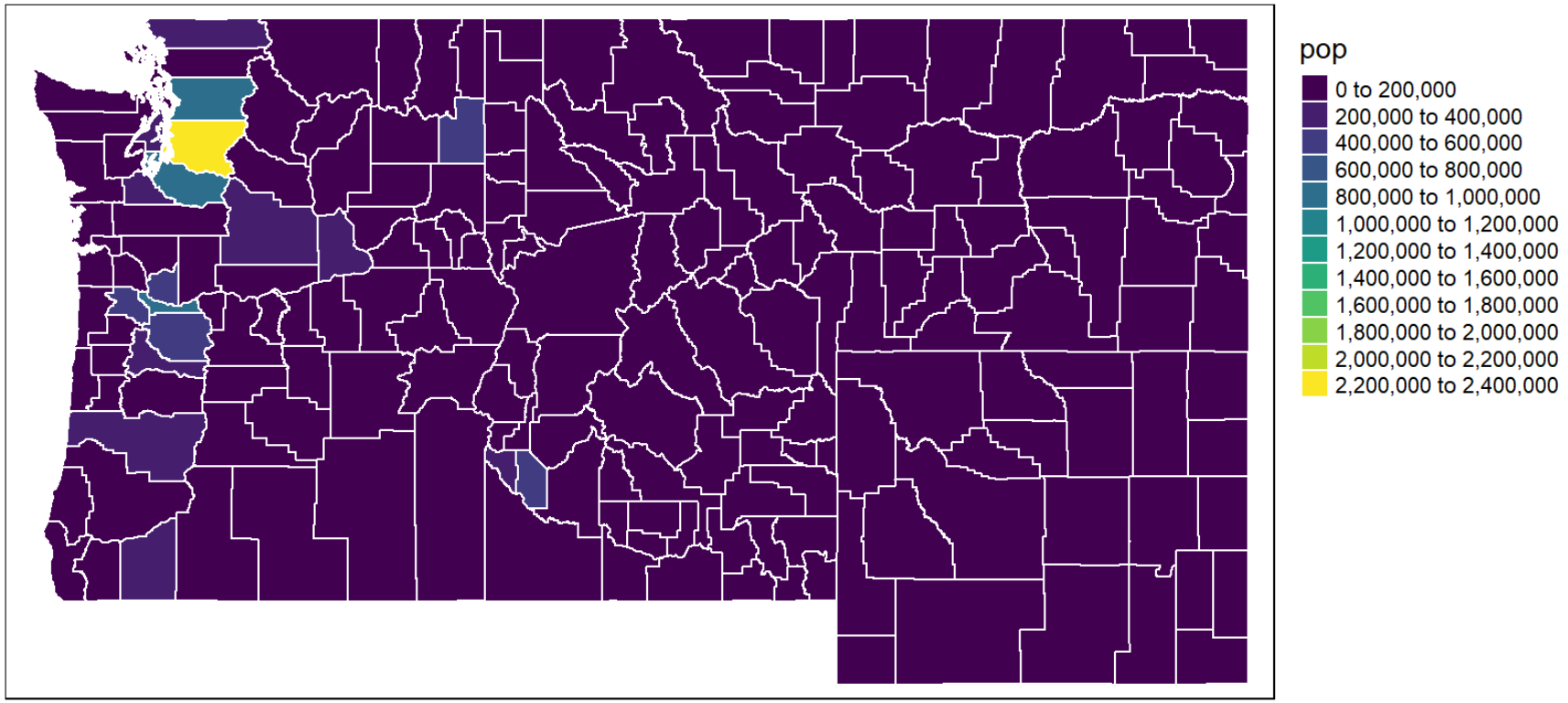
Using tmap


pop
0 to 500,000
500,000 to 1,000,000
1,000,000 to 1,500,000
1,500,000 to 2,000,000
2,000,000 to 2,500,000

Changing aesthetics

```
pt <- tm_shape(cty.info) +
  tm_polygons(col = "pop", n=10,palette=viridis(10),
              border.col = "white") +
  tm_legend(outside = TRUE)
```

Changing aesthetics



Adding layers

```
pt <- tm_shape(cty.info) +
  tm_polygons(col = "pop", n=10,palette=viridis(10),
              border.col = "white") +
  tm_shape(st) +
  tm_borders("red") +
  tm_legend(outside = TRUE)
```

Adding layers