

Data Visualization and Maps II

HES 505 Fall 2023: Session 30

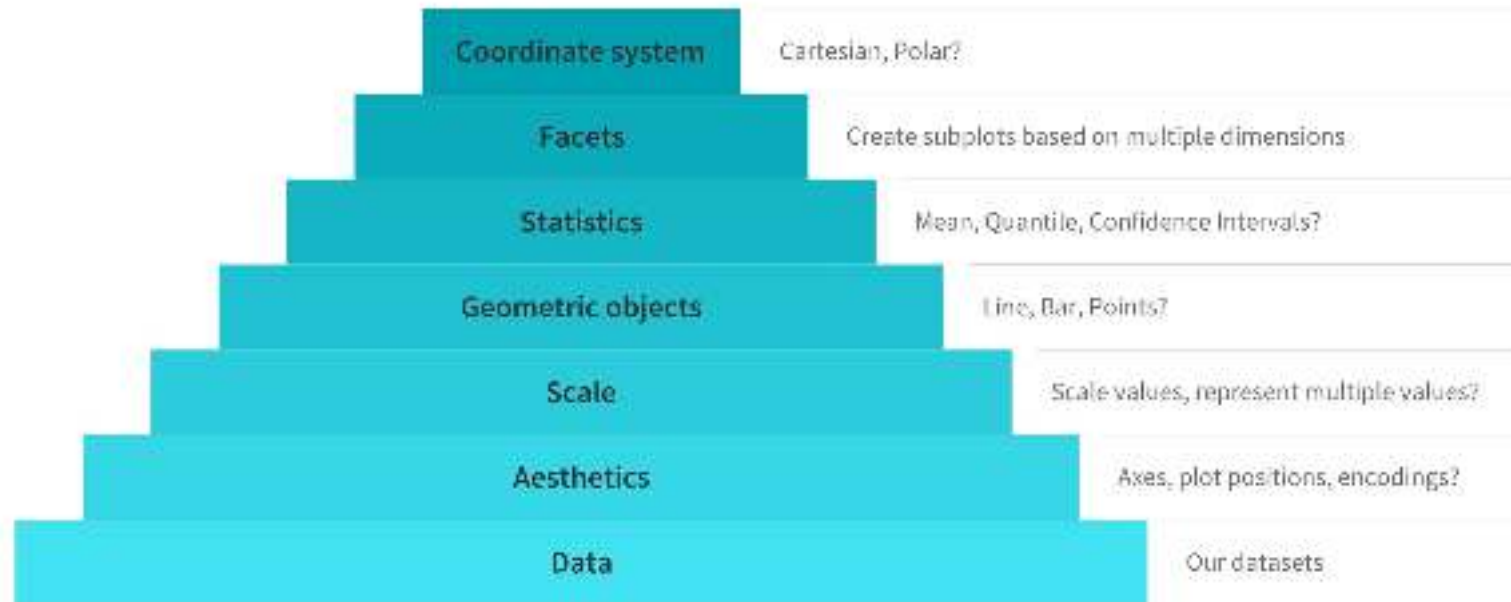
Matt Williamson

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	<code>ggplot(data)</code>	<i>The raw data that you want to visualise.</i>
Aesthetics	<code>aes()</code>	<i>Aesthetic mappings between variables and visual properties.</i>
Geometries	<code>geom_*()</code>	<i>The geometric shapes representing the data.</i>

The Grammar of {ggplot2}

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

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](https://www.kaggle.com)
- 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 graphics 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 <code>as.data.frame()</code> . 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 like <code>geom_point()</code> .
environment	DEPRECATED. Used prior to tidy evaluation.

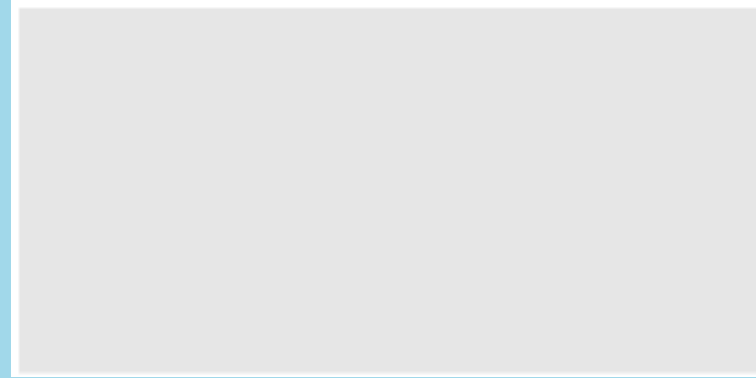
Details

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

- 1. `ggplot(data, mapping, aes, theme, ...) { ... }`
- 2. `ggplot(data)`
- 3. `ggplot()`

Data

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



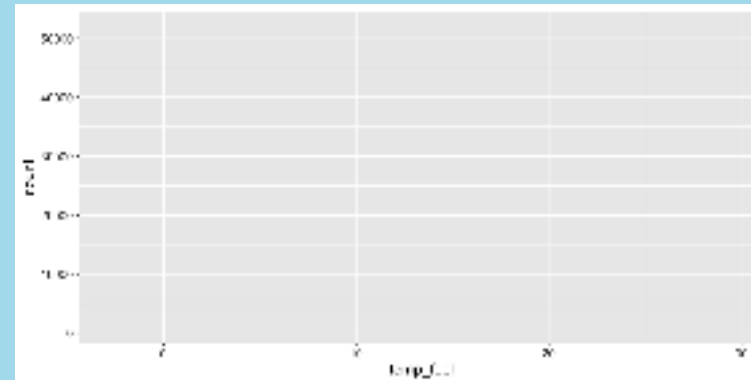
Aesthetic Mapping

= 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))
```

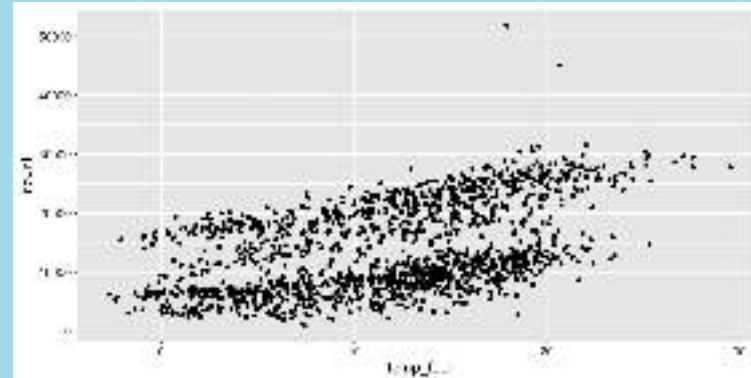

Geometries

= interpret aesthetics as graphical representations

- points
- lines
- polygons
- text labels
- ...

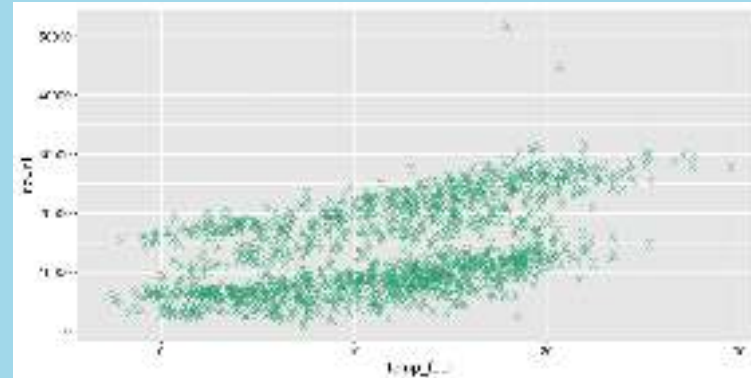
Geometries

```
1 ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = count)  
4 ) +  
5   geom_point()
```



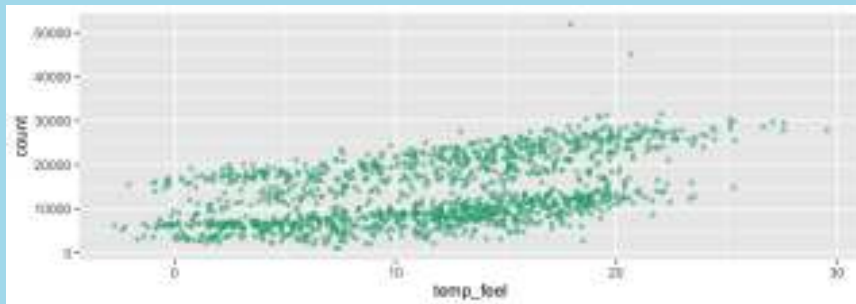
Visual Properties of Layers

```
1 ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = count)  
4 ) +  
5 geom_point(  
6   color = "#28a87d",  
7   alpha = .5,  
8   shape = "x",  
9   stroke = 1,  
10  size = 4  
11 )
```

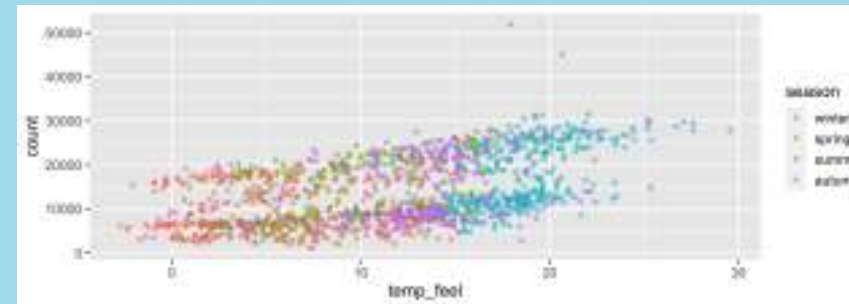


Setting vs Mapping of Visual Properties

```
1 ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = count)  
4 ) +  
5 geom_point(  
6   color = "#28a87d",  
7   alpha = .5  
8 )
```

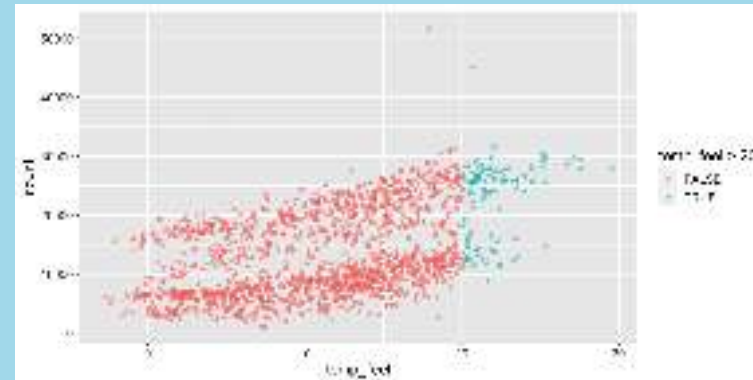


```
1 ggplot(  
2   bikes,  
3   aes(x = temp_feel, y = count)  
4 ) +  
5 geom_point(  
6   aes(color = season),  
7   alpha = .5  
8 )
```



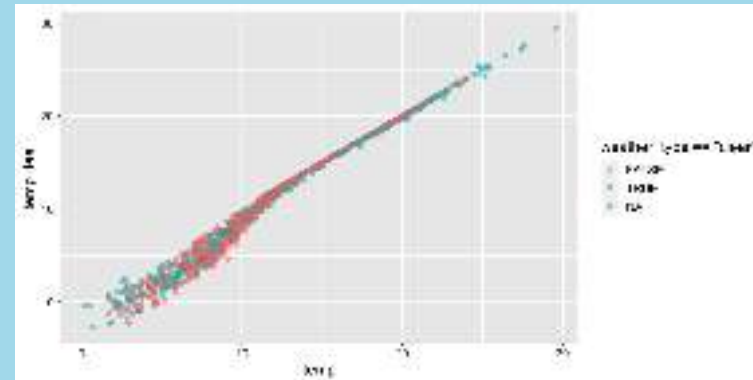
Mapping Expressions

```
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 )
```



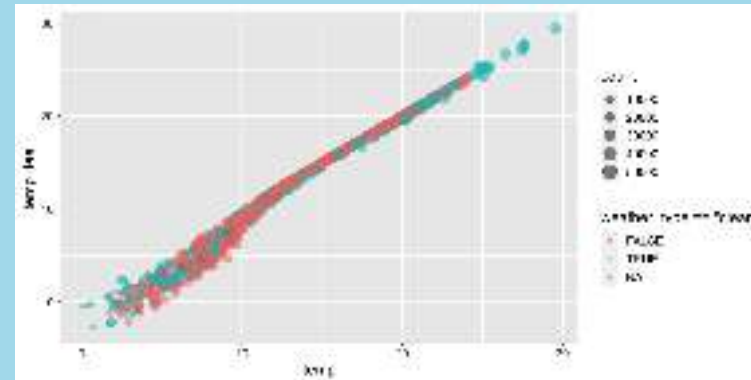
Mapping Expressions

```
1 ggplot(  
2   bikes,  
3   aes(x = temp, y = temp_feel)  
4 ) +  
5 geom_point(  
6   aes(color = weather_type == "cloudy"),  
7   alpha = .5,  
8   size = 2  
9 )
```



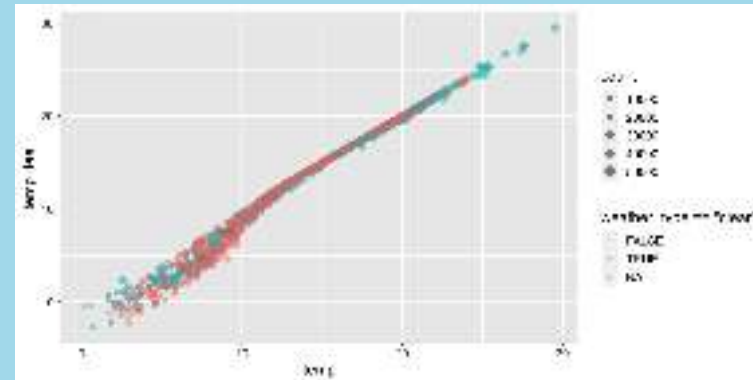
Mapping to Size

```
1 ggplot(  
2   bikes,  
3   aes(x = temp, y = temp_feel)  
4 ) +  
5 geom_point(  
6   aes(color = weather_type == "cloudy",  
7       size = count),  
8   alpha = .5  
9 )
```



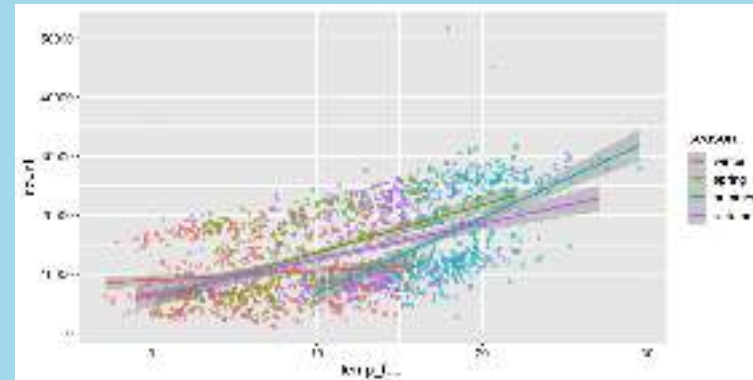
Setting a Constant Property

```
1 ggplot(  
2   bikes,  
3   aes(x = temp, y = temp_feel)  
4 ) +  
5   geom_point(  
6     aes(color = weather_type == "cloudy",  
7         size = count),  
8     shape = 18,  
9     alpha = .5  
10  )
```



Adding More Layers

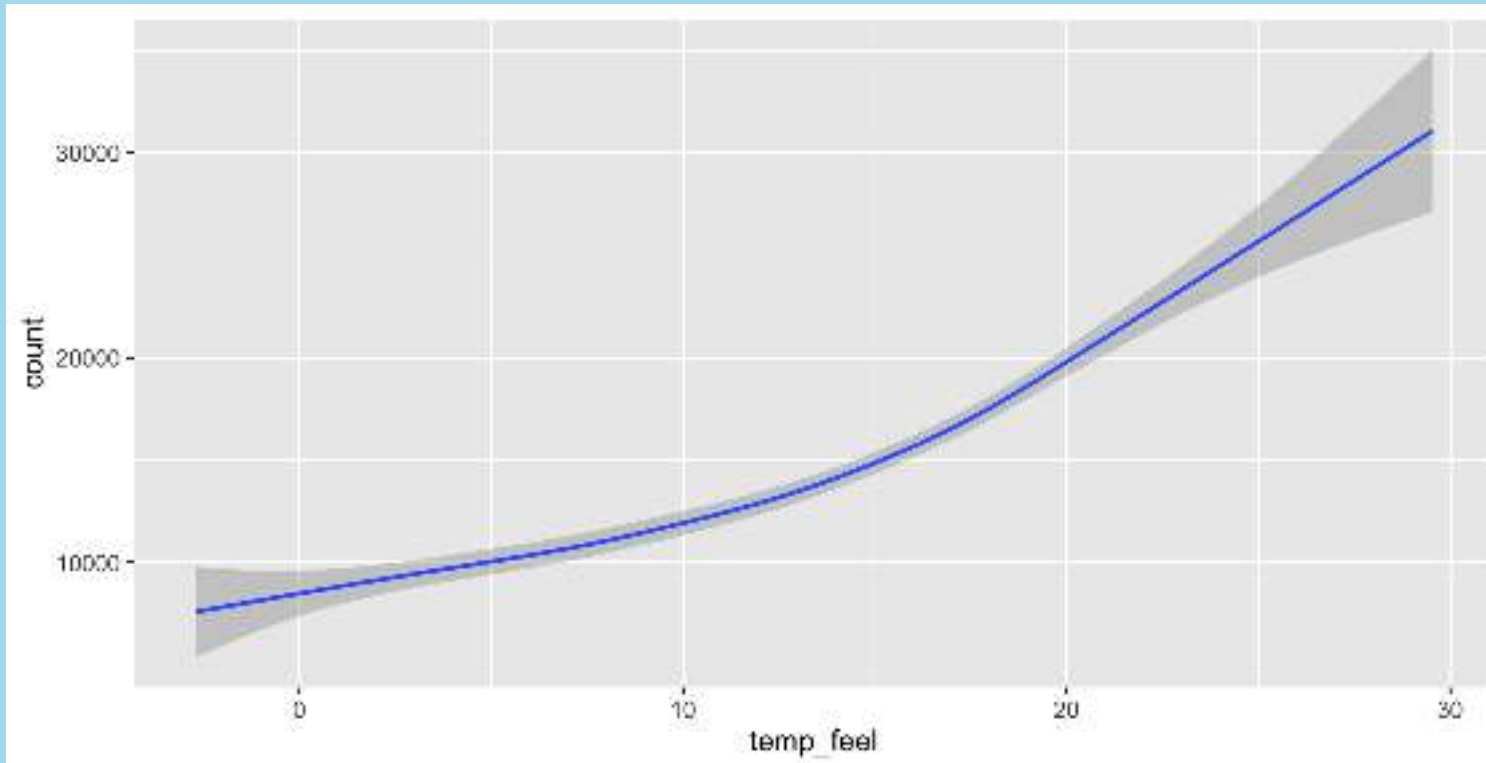
```
1  ggplot(  
2    bikes,  
3    aes(x = temp_feel, y = count,  
4        color = season)  
5  ) +  
6  geom_point(  
7    alpha = .5  
8  ) +  
9  geom_smooth(  
10    method = "lm"  
11  )
```



Statistical Layers

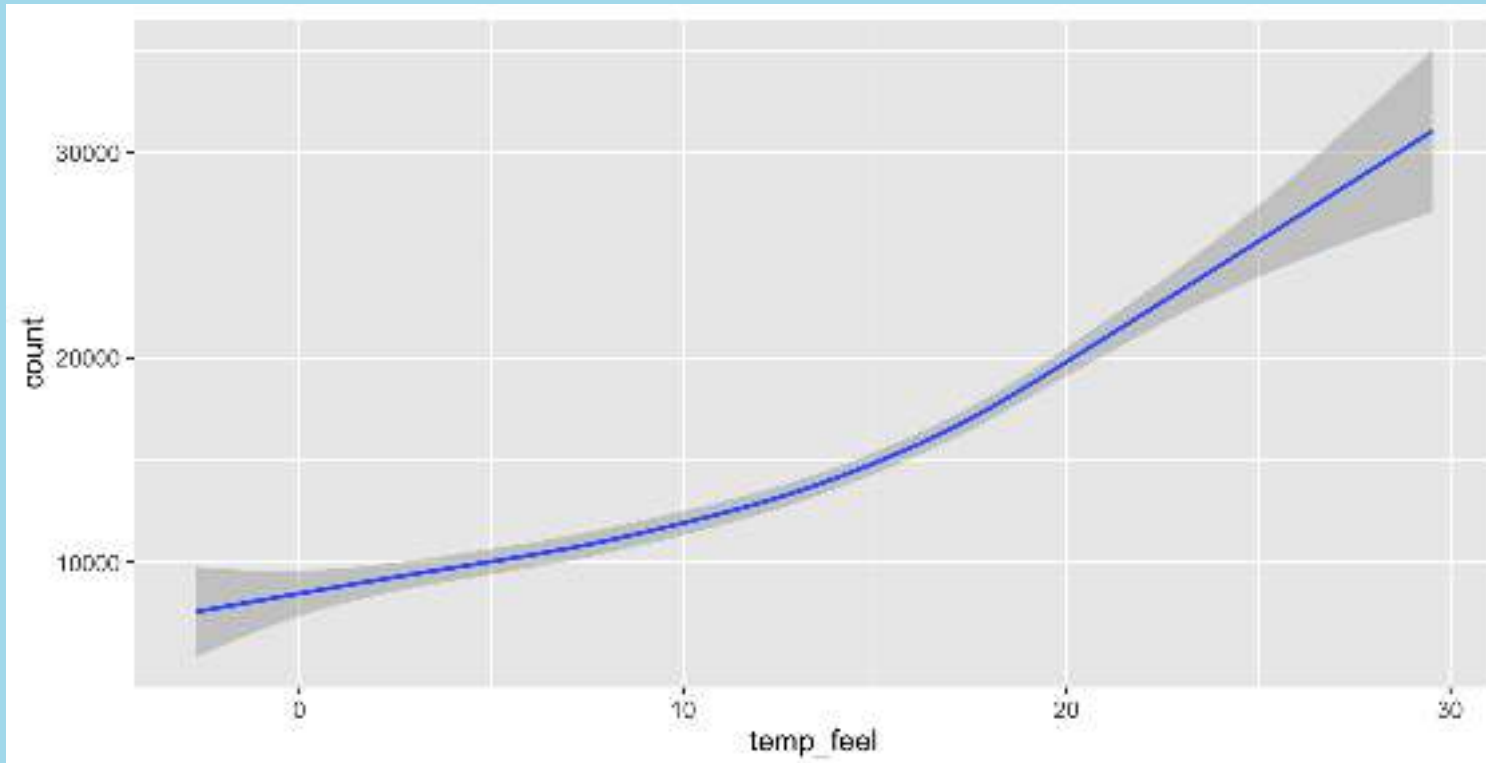
`stat_*()` and `geom_*()`

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



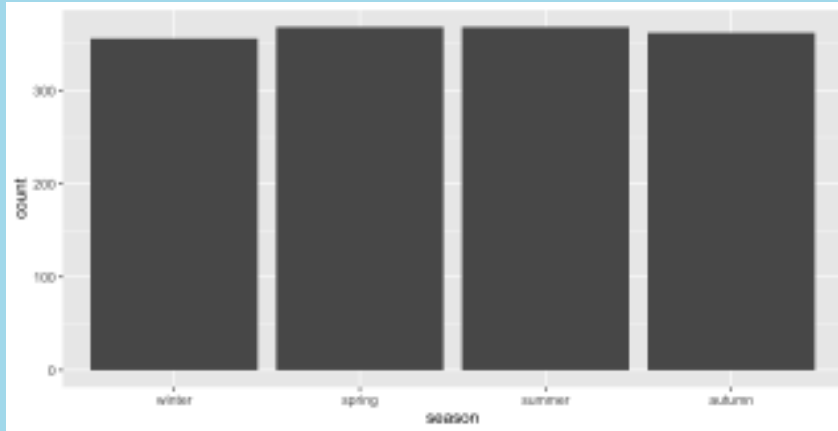
`stat_*()` and `geom_*()`

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

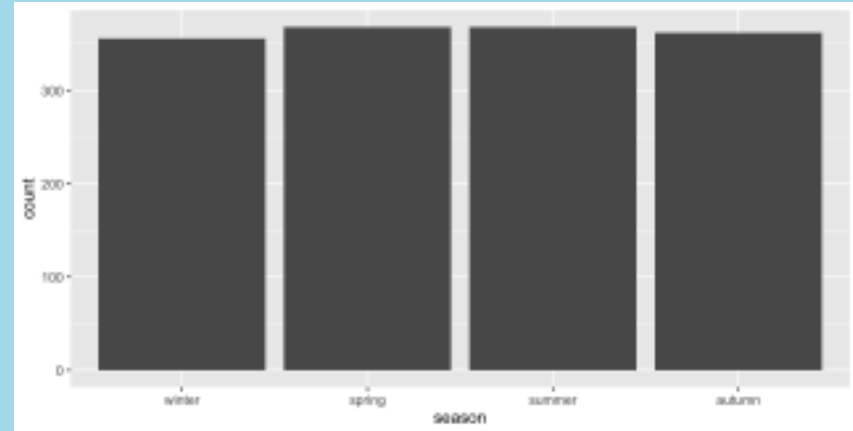


`stat_*()` and `geom_*()`

```
1 ggplot(bikes, aes(x = season)) +  
2   stat_count(geom = "bar")
```

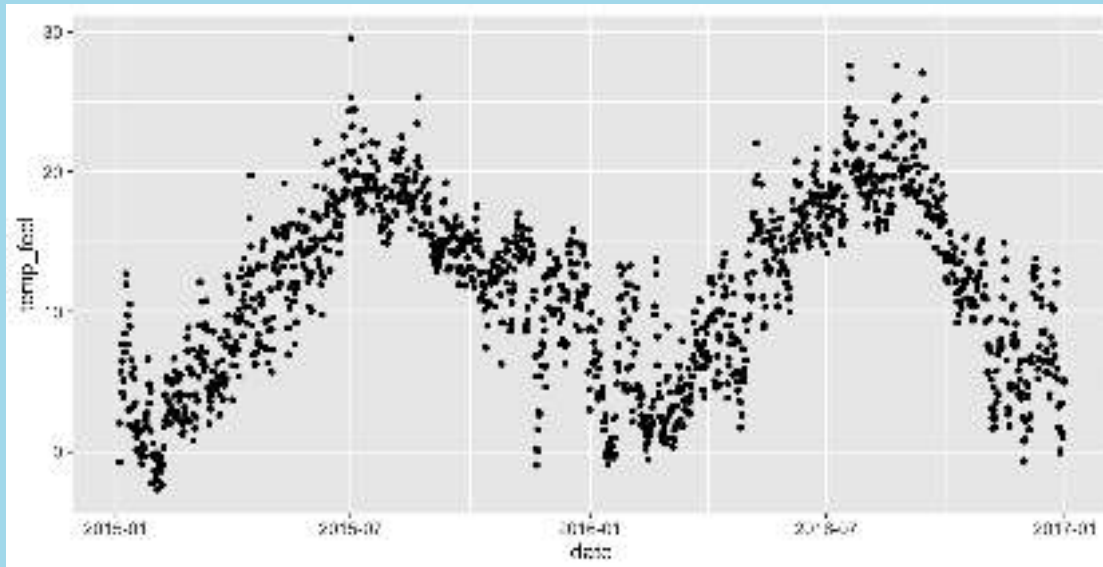


```
1 ggplot(bikes, aes(x = season)) +  
2   geom_bar(stat = "count")
```

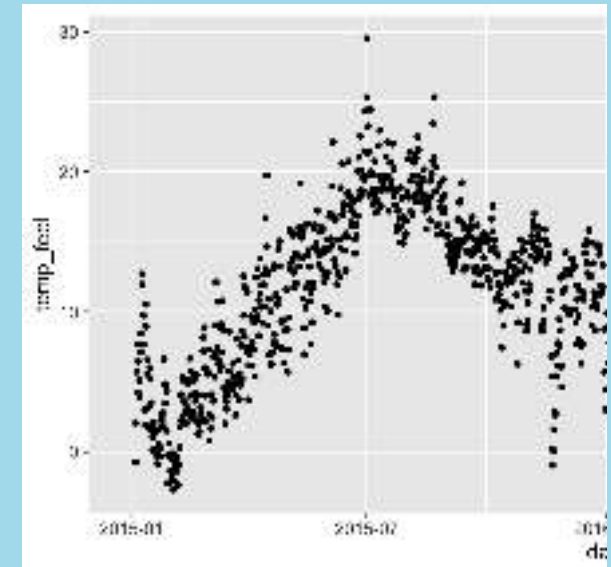


`stat_*()` and `geom_*()`

```
1 ggplot(bikes, aes(x = date, y = temp_feel)) +  
2   stat_identity(geom = "point")
```



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

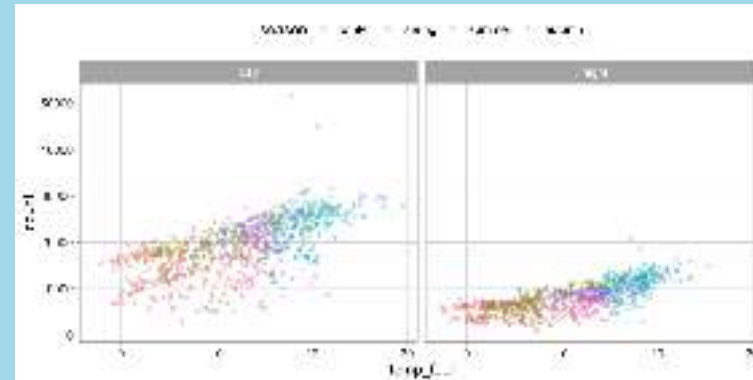
Autumn	Spring
Subset for Autumn	Subset for Spring
Summer	Winter
Subset for Summer	Subset for Winter

facet_grid()

2015	2016	Day
Subset for Day = 2015	Subset for Day = 2016	
Subset for Night = 2015	Subset for Night = 2016	Night

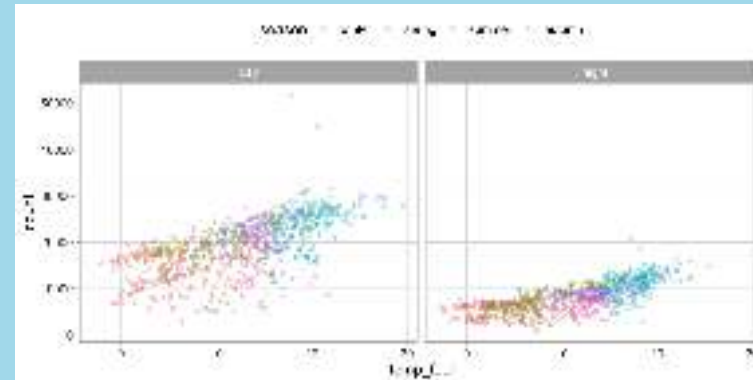
Wrapped Facets

```
1 g <-  
2   ggplot(  
3     bikes,  
4     aes(x = temp_feel, y = count,  
5         color = season)  
6   ) +  
7   geom_point(  
8     alpha = .3,  
9     guide = "none"  
10  )  
11 g +  
12   facet_wrap(  
13     vars(day_night)  
14   )
```



Wrapped Facets

```
1 g +  
2   facet_wrap(  
3     ~ day_night  
4   )
```



Scales

Scales

= translate between variable ranges and property ranges

- feels-like temperature \Leftrightarrow x
- reported bike shares \Leftrightarrow y
- season \Leftrightarrow color
- year \Leftrightarrow 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:

- **positions** via `scale_x_*()` and `scale_y_*()`
- **colors** via `scale_color_*()` and `scale_fill_*()`
- **sizes** via `scale_size_*()` and `scale_radius_*()`
- **shapes** via `scale_shape_*()` and `scale_size_*()`

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().`

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)

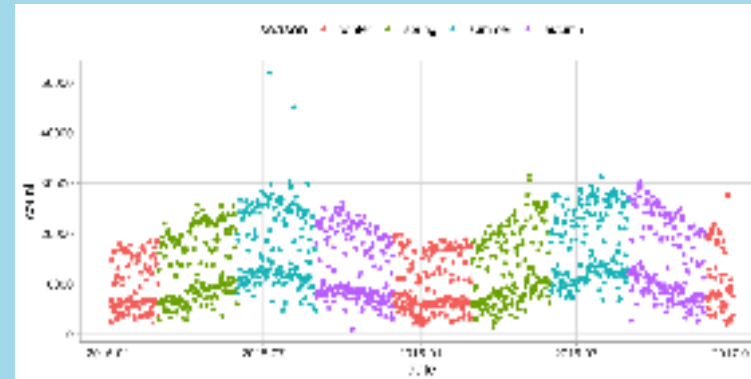
Discrete:

qualitative or categorical data

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

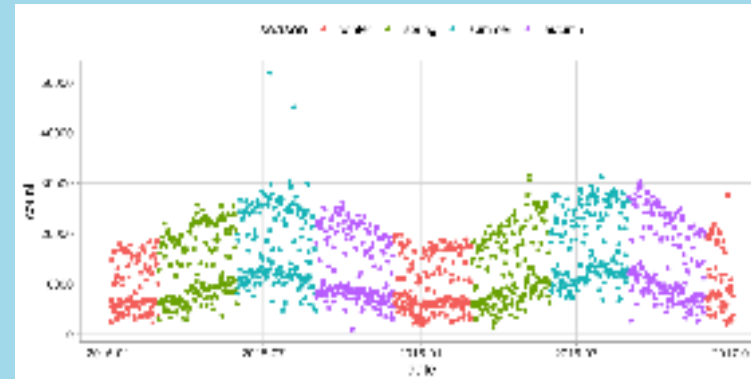
Aesthetics + Scales

```
1 ggplot(  
2   bikes,  
3   aes(x = date, y = count,  
4       color = season)  
5 ) +  
6 geom_point()
```



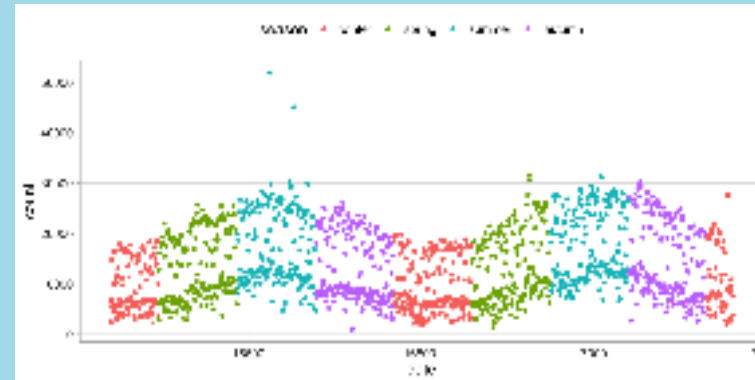
Aesthetics + Scales

```
1 ggplot(  
2   bikes,  
3   aes(x = date, y = count,  
4       color = season)  
5 ) +  
6 geom_point() +  
7 scale_x_date() +  
8 scale_y_continuous() +  
9 scale_color_discrete()
```



Scales

```
1 ggplot(  
2   bikes,  
3   aes(x = date, y = count,  
4       color = season)  
5 ) +  
6 geom_point() +  
7 scale_x_continuous() +  
8 scale_y_continuous() +  
9 scale_color_discrete()
```



Coordinate Systems

= interpret the position aesthetics

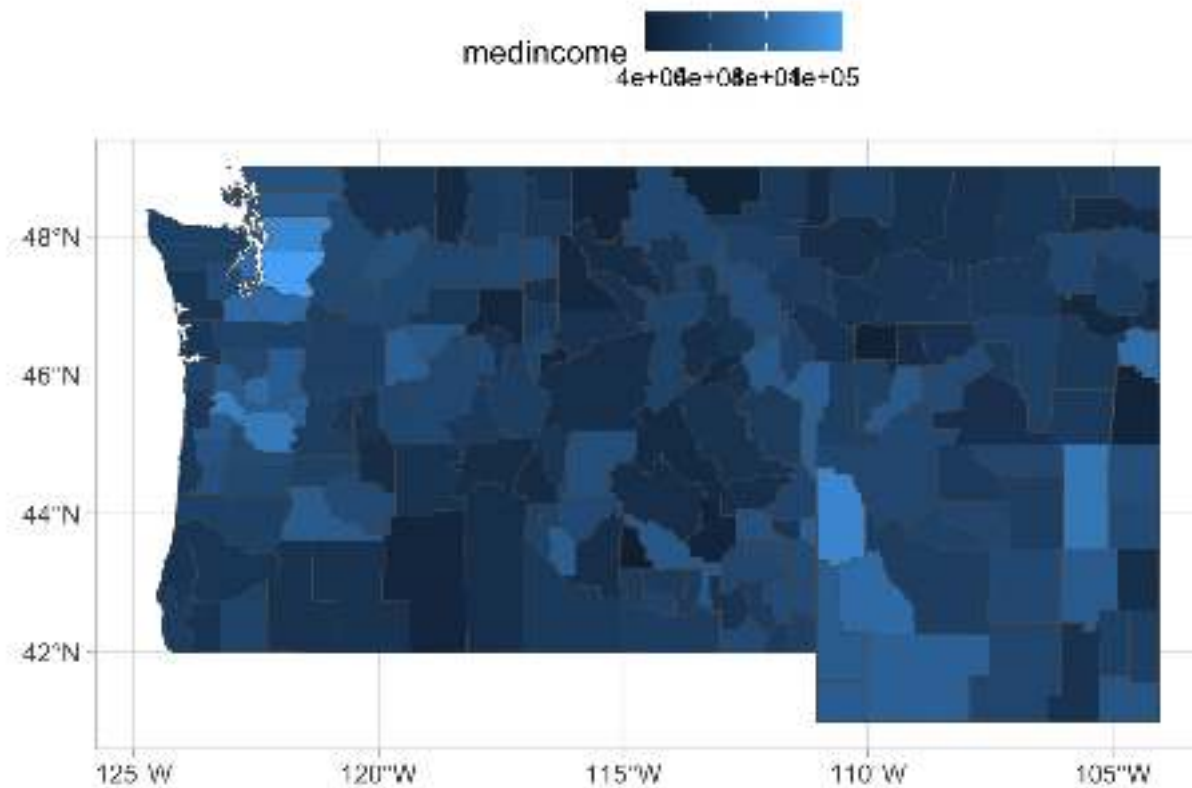
- **linear coordinate systems:** preserve the geometrical shapes
 - `coord_cartesian()`
 - `coord_fixed()`
 - `coord_flip()`
- **non-linear coordinate systems:** likely change the geometrical shapes
 - `coord_polar()`

Building Choropleth Maps

Using ggplot2

```
1 cty.info <- get_acs(geography = "county",
2                     variables = c(pop="B01003_001",
3                                   medincome = "B19013_001"),
4                     survey="acs5",
5                     state = c("WA", "OR", "ID", "MT", "WY"),
6                     geometry = TRUE, key = censkey, progress_bar=FALSE) %
7   select(., -moe) %>%
8   pivot_wider(
9     names_from = "variable",
10    values_from = "estimate"
11  )
12
13 p <- ggplot(data=cty.info) +
14   geom_sf(mapping=aes(fill=medincome))
```

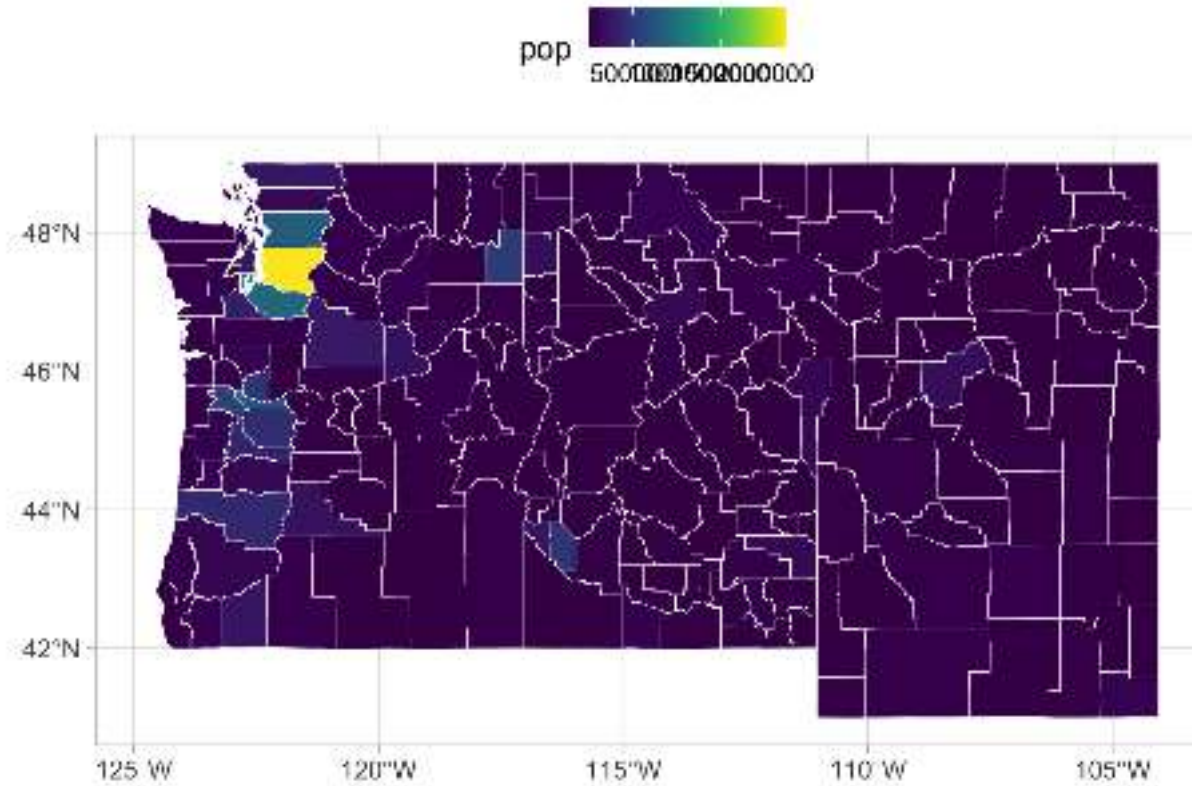
Static Maps with **ggplot2**



Changing aesthetics

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

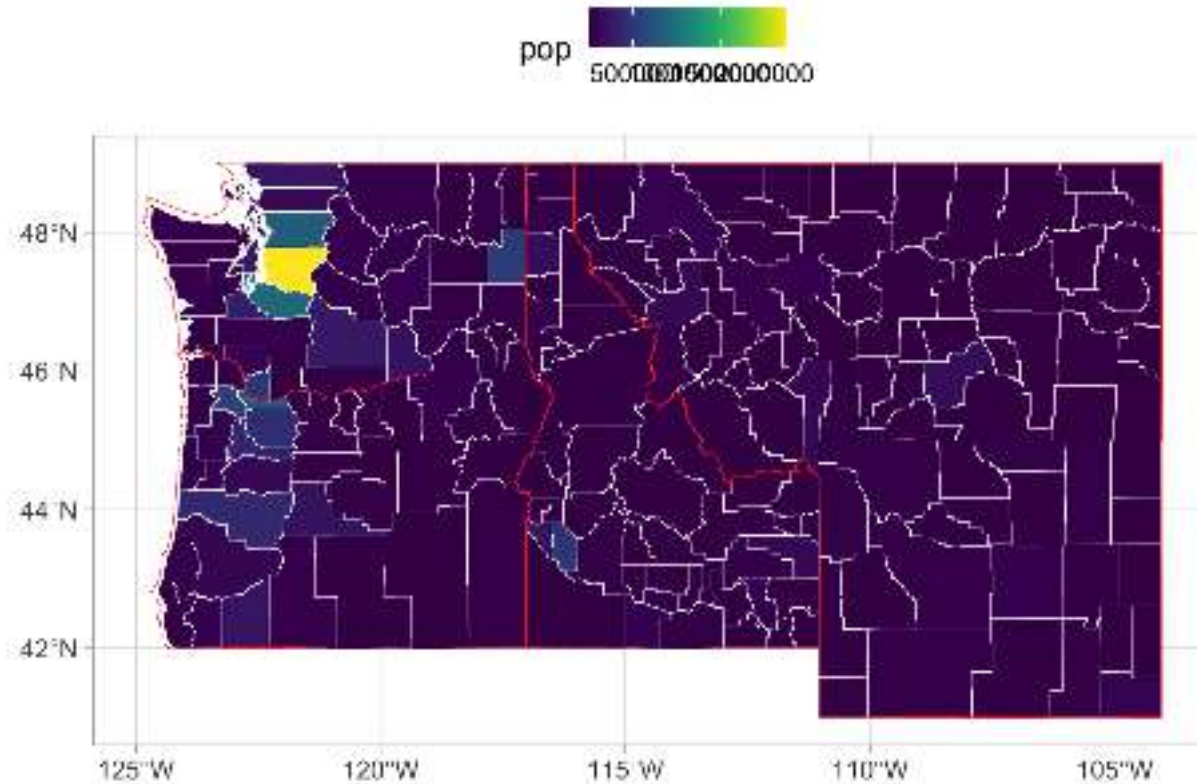
Changing aesthetics



Adding layers

```
1 st <- tigris::states(progress_bar=FALSE) %>%  
2   filter(., STUSPS %in% c("WA", "OR", "ID", "MT", "WY"))  
3  
4 p <- ggplot(data=cty.info) +  
5   geom_sf(mapping=aes(fill=pop), color="white") +  
6   geom_sf(data=st, fill=NA, color="red") +  
7   scale_fill_viridis()
```

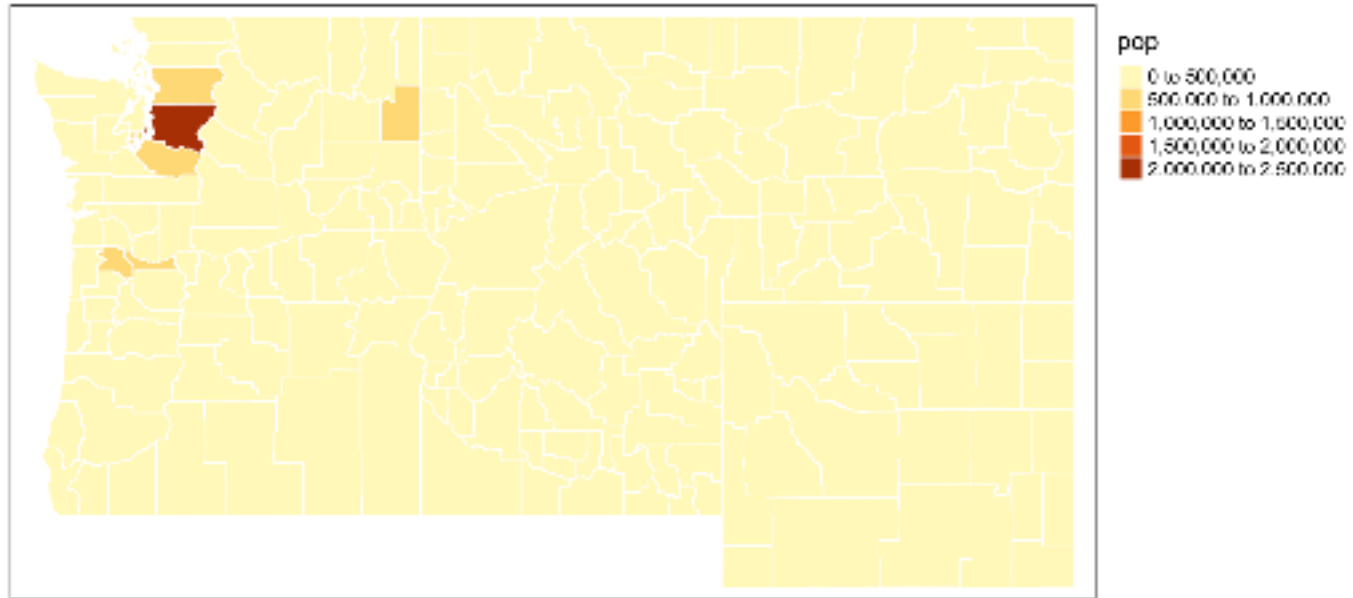
Adding layers



Using **tmap**

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

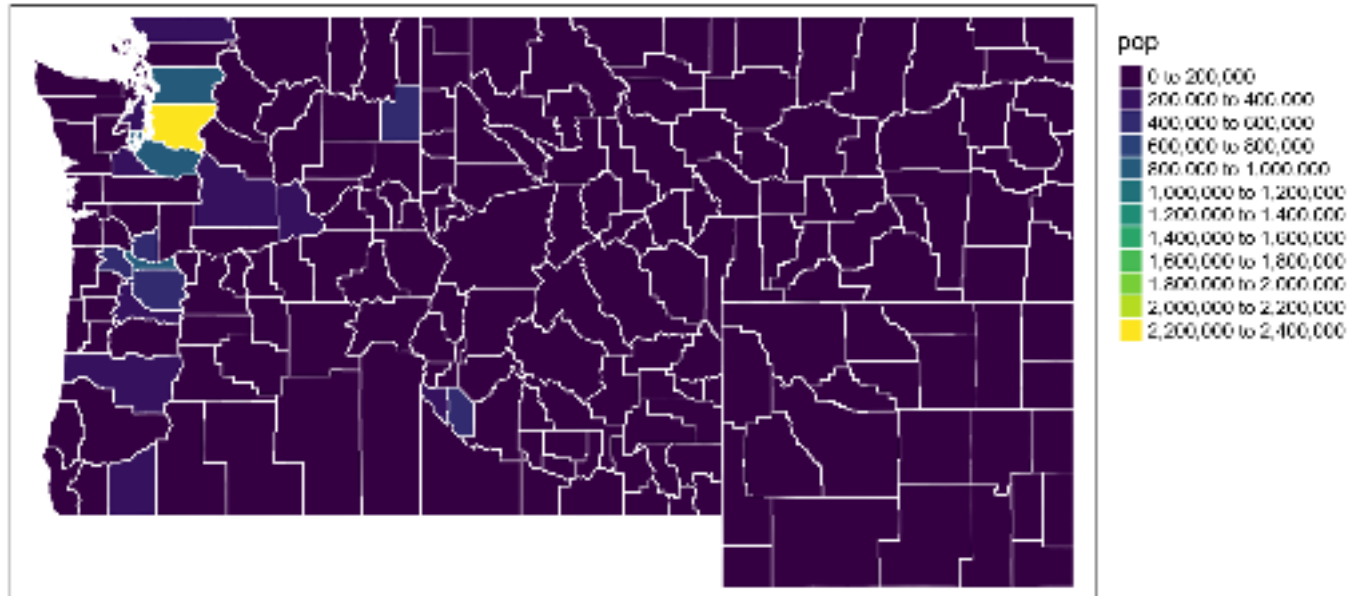
Using **tmap**



Changing aesthetics

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

Changing aesthetics



Adding layers

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

Adding layers

