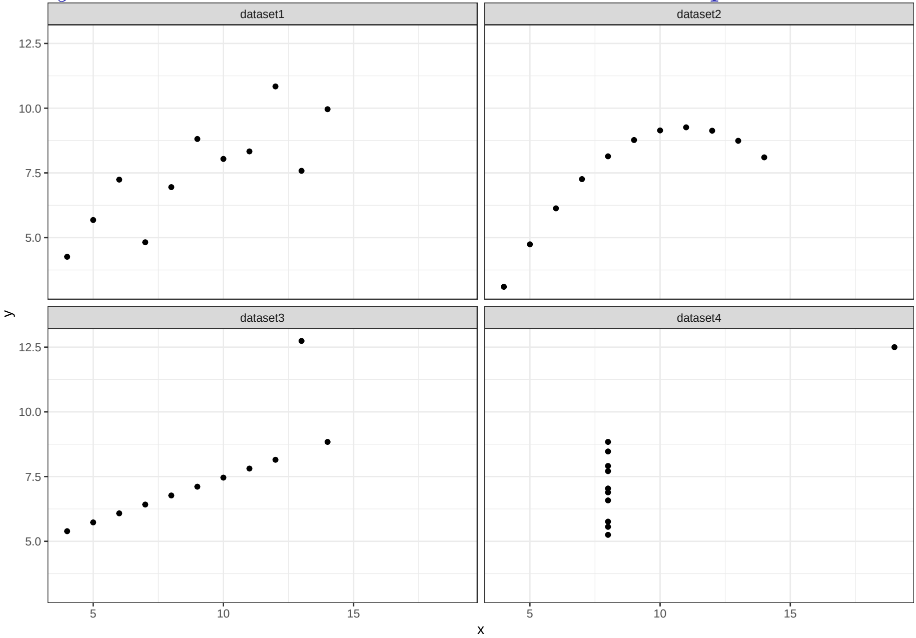
R Data Vis: ggplot

Brian Wright, PhD

Feb, 2020

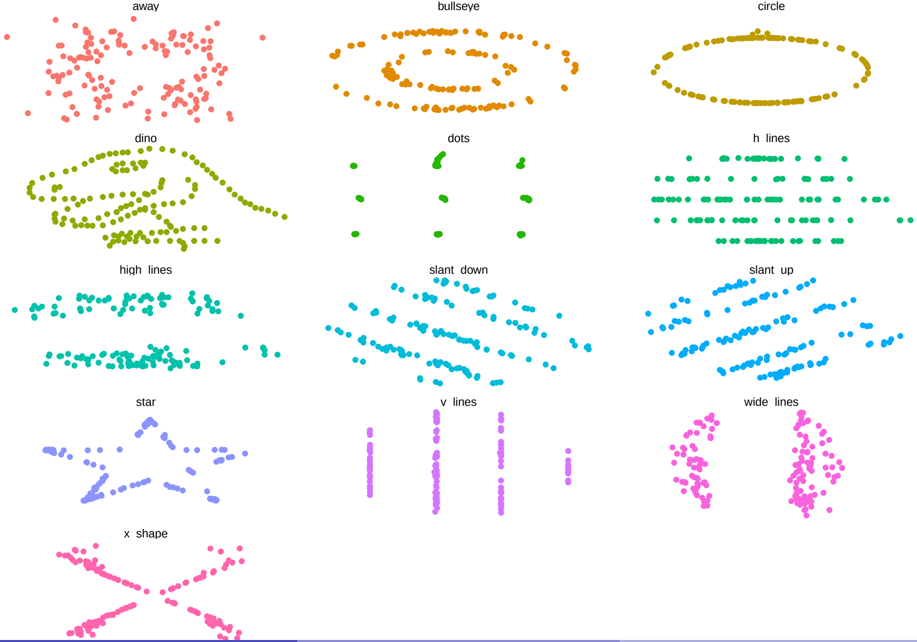
Why visualize data? Anscombe's data example



All Have the Summary Stats

Statistic	Value
`mean(x)`	9
`mean(y)`	7.5
`var(x)`	11
`var(y)`	4.13
`cor(x,y)`	0.82

The DataSaurus dozen



Same Stats

Statistic	Value
`mean(x)`	54.3
`mean(y)`	47.8
`var(x)`	281
`var(y)`	725
`cor(x,y)`	-0.07

Bottom line

- Summary statistics cannot always distinguish datasets
- Take advantage of humans' ability to visually recognize and remember patterns
- Find discrepancies in the data more easily

What is ggplot2?

- A second (and final) iteration of the ggplot
- Implementation of Wilkerson's Grammar of Graphics in R
- Conceptually, a way to layer different elements onto a canvas to create a data visualization
- Started as Dr. Hadley Wickham's PhD thesis (with Dr. Dianne Cook)
- Won the John M. Chambers Statistical Software Award in 2006
- Mimicked in other software platforms
  - `ggplot` and `seaborn` in Python
  - Translated in `plotly`

ggplot2 uses the ***grammar*** *of* ***graphics***

A grammar . . .

- compose and re-use small parts
- build complex structures from simpler units

of graphics

- think of yourself as a painter
- build a visualization using layers on a canvas
- draw layers on top of each other

A dataset

```r
library(tidyverse)
library(rio)
beaches <- import('beaches.csv')
```

```
#>  # A tibble: 6 x 12
#>    date        year month   day season rainfall temperature enterococci day_num
#>    <date>     <int> <int> <int>  <int>    <dbl>       <dbl>       <dbl>   <int>
#>  1 2013-01-02  2013     1     2      1       0          23.4         6.7       2
#>  2 2013-01-06  2013     1     6      1       0          30.3         2         6
#>  3 2013-01-12  2013     1    12      1       0          31.4        69.1      12
#>  4 2013-01-18  2013     1    18      1       0          46.4         9        18
#>  5 2013-01-24  2013     1    24      1       0          27.5        33.9      24
#>  6 2013-01-30  2013     1    30      1       0.6        26.6        26.5      30
#>  # ... with 3 more variables: month_num <int>, month_name <chr>,
#>  #   season_name <chr>
```

Building a graph: Start with a blank canvas

```
ggplot()
```

Add a data set

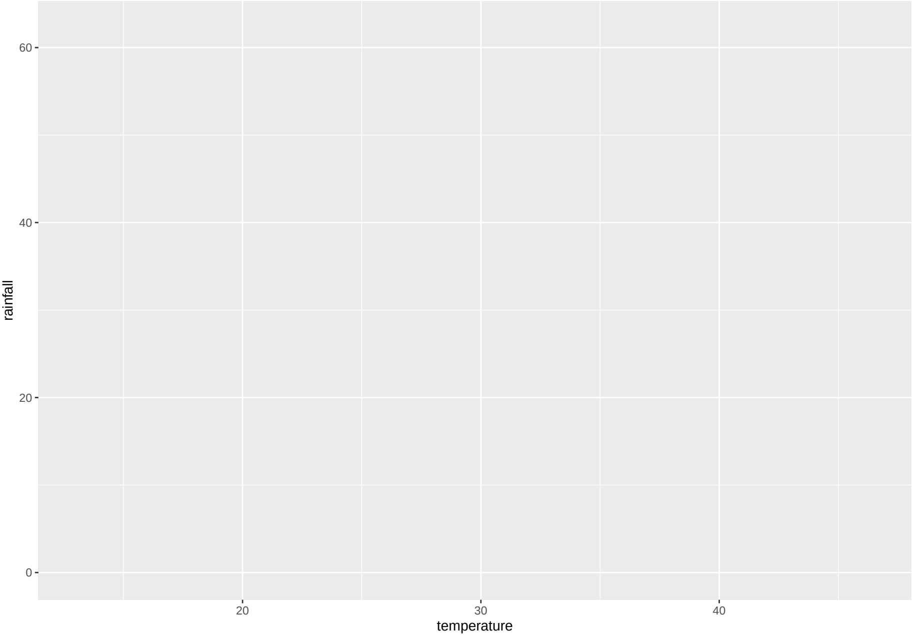
```r
ggplot(
  data = beaches #<< loaded earlier
)
```

Add a mapping from data to elements

```r
ggplot(
  data = beaches,
  mapping = aes( # everytime we add a "data element"
    #we add a aesthetic
    x = temperature,
    y = rainfall
  )
)
```

What goes in

- the x and y axes
- the color of markers
- the shape of markers

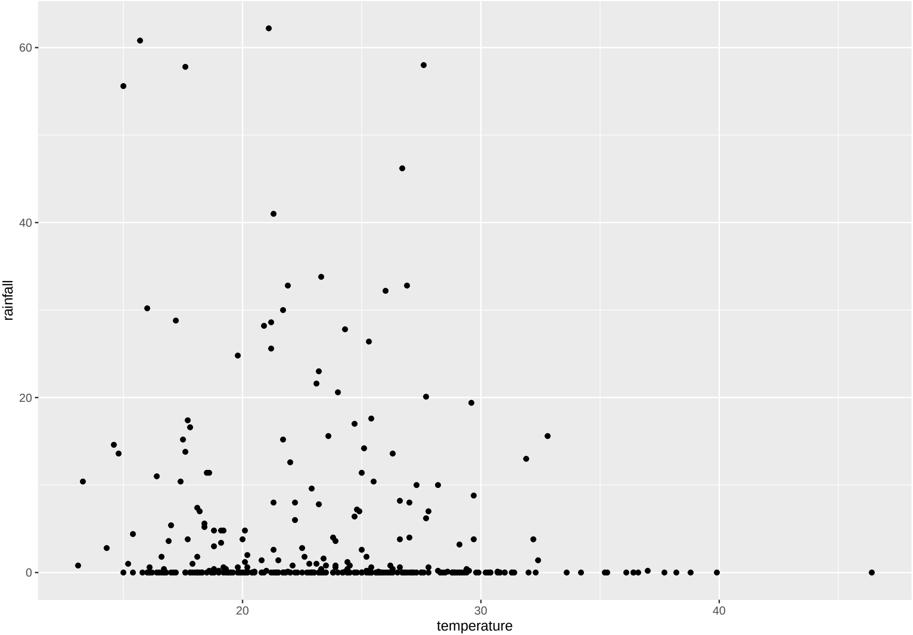

60
40
20
0
rainfall
20
30
40
temperature

Add a geometry to draw

```
ggplot(
  data = beaches,
  mapping = aes(
    x = temperature,
    y = rainfall
  )
) +
  geom_point()#?
```

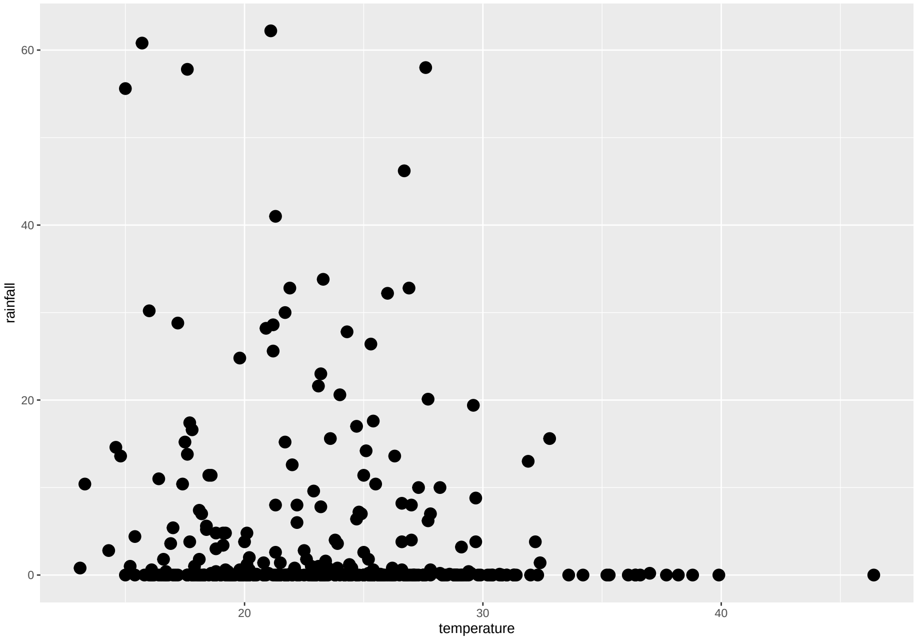
What to draw:

- Points, lines
- Histogram, bars, pies, etc.


rainfall
0
20
40
60
20
30
40
temperature

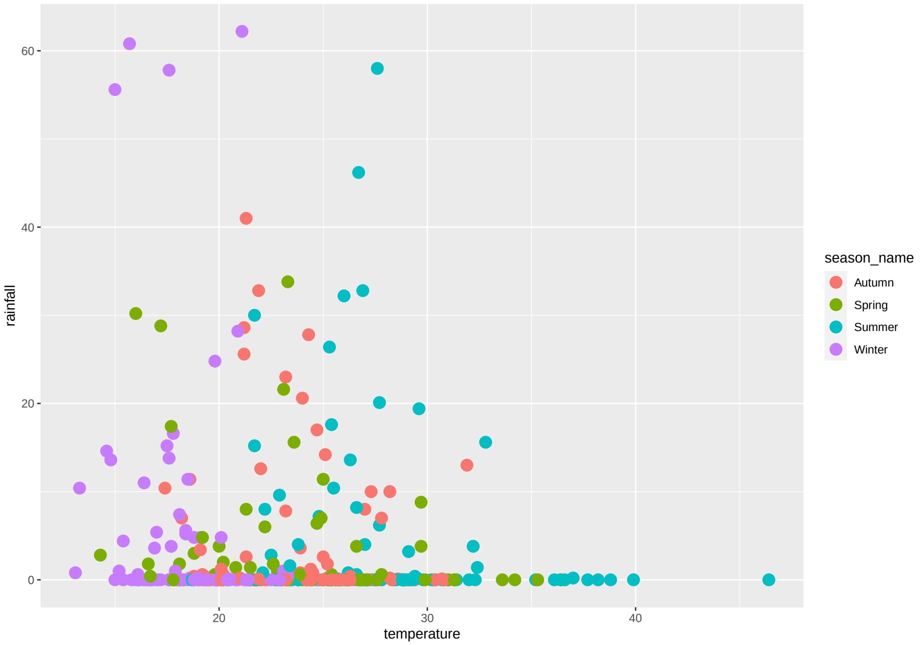
Add options for the geom

```
ggplot(
  data = beaches,
  mapping = aes(
    x = temperature,
    y = rainfall
  )
) +
  geom_point(size = 4)#?
```


60
40
20
0
rainfall
20
30
40
temperature

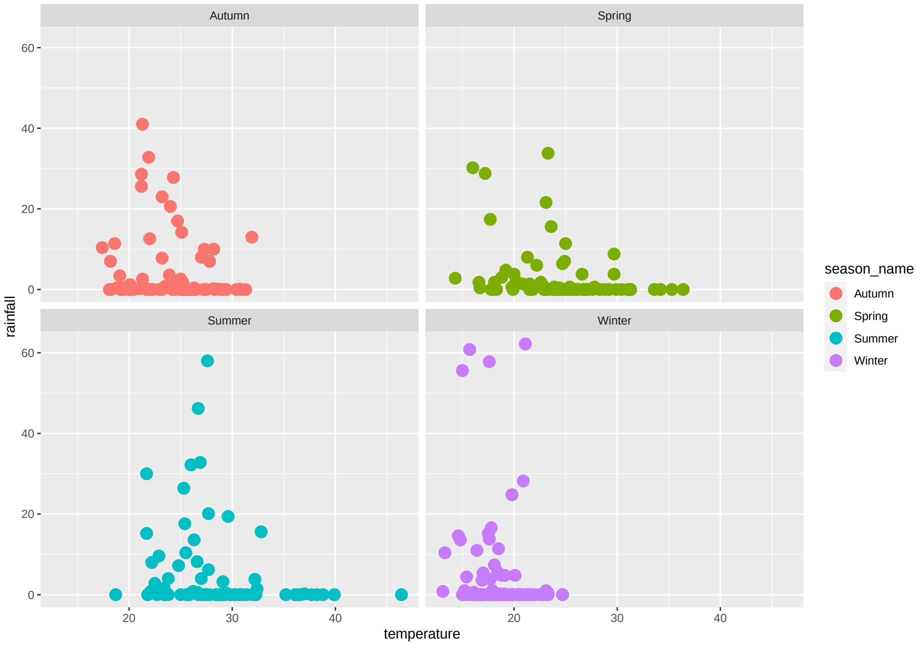
Add a mapping to modify the geom

```r
ggplot(
  data = beaches,
  mapping = aes(
    x = temperature,
    y = rainfall
  )
) +
  geom_point(
    mapping = aes(color = season_name),
    size = 4 #Why do we need the mapping?
  )
```


60
40
20
0
rainfall
20
30
40
temperature
season_name
● Autumn
● Spring
● Summer
● Winter

Split into facets

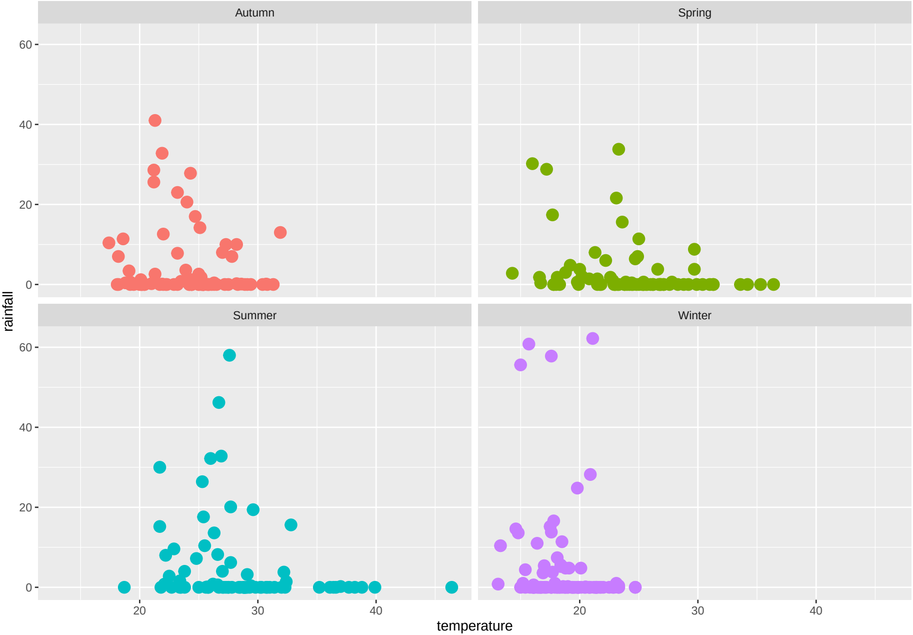
```r
ggplot(
  data = beaches,
  mapping = aes(
    x = temperature,
    y = rainfall
  )
) +
  geom_point(
    mapping = aes(color = season_name),
    size = 4
  ) +
  facet_wrap( ~ season_name)###
```



Autumn
Spring
Summer
Winter
rainfall
0
20
40
60
temperature
20
30
40
season_name
Autumn
Spring
Summer
Winter

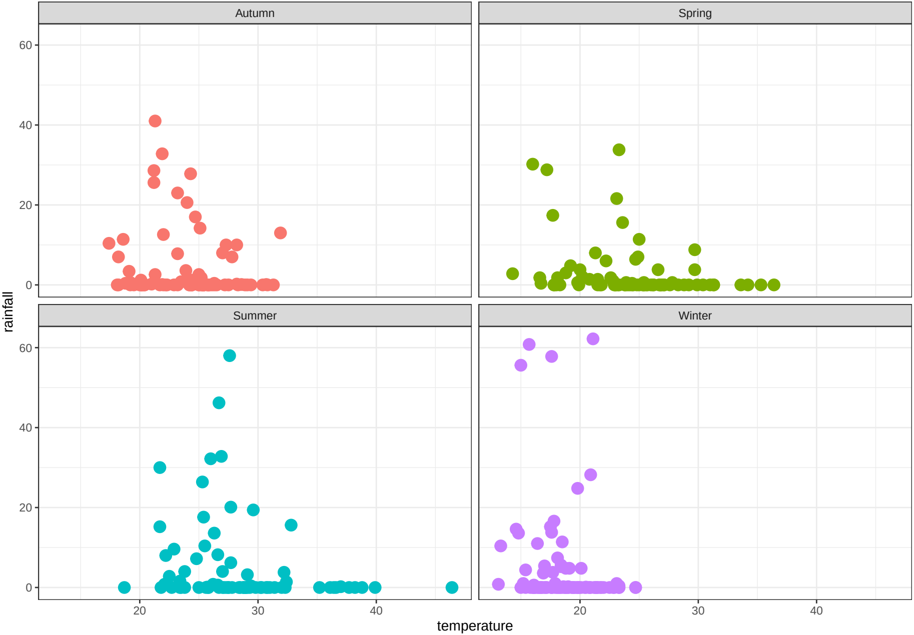
Remove the legend

```
ggplot(
  data = beaches,
  mapping = aes(
    x = temperature,
    y = rainfall
  )
) +
  geom_point(
    mapping = aes(color = season_name),
    size = 4,
    show.legend = FALSE ###
  ) +
  facet_wrap( ~ season_name)
```


Autumn
Spring
Summer
Winter
rainfall
temperature
0
20
40
60
20
30
40

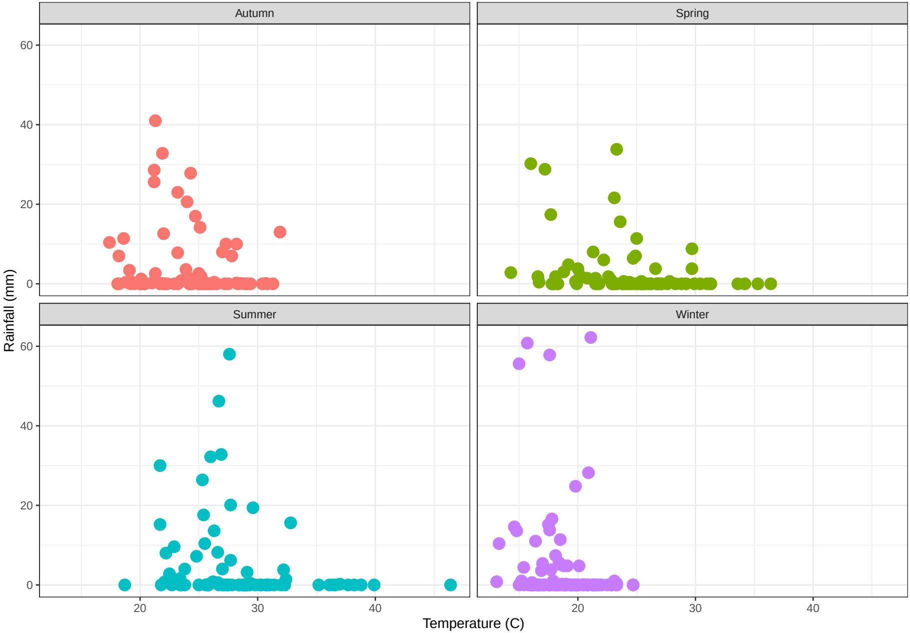
Change the general theme

```
ggplot(
  data = beaches,
  mapping = aes(
    x = temperature,
    y = rainfall
  )
) +
  geom_point(
    mapping = aes(color = season_name),
    size = 4,
    show.legend = FALSE
  ) +
  facet_wrap( ~ season_name) +
  theme_bw() ###
```



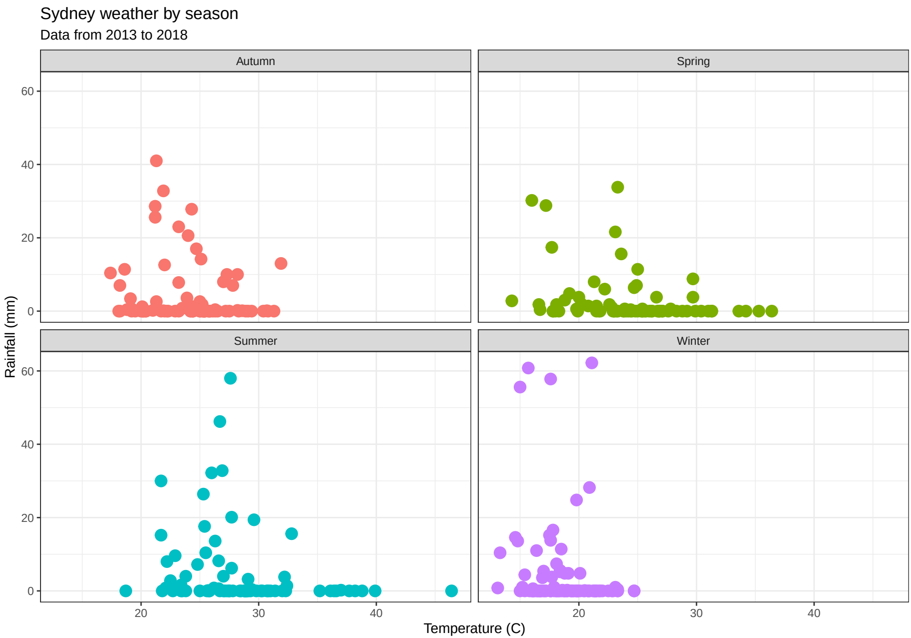
##

Update the labels


Autumn
Spring
Summer
Winter
Rainfall (mm)
60
40
20
0
20
30
40
Temperature (C)

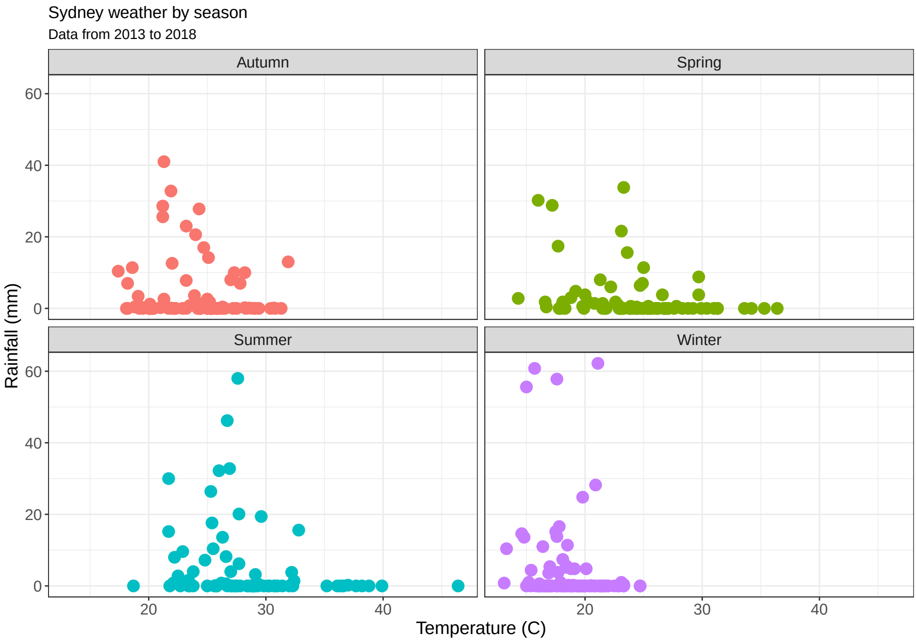
Add titles

```
ggplot(
  data = beaches,
  mapping = aes(
    x = temperature,
    y = rainfall)
) +
  geom_point(
    mapping = aes(color = season_name),
    size = 4,
    show.legend = FALSE
  ) +
  facet_wrap( ~ season_name) +
  theme_bw() +
  labs(x = 'Temperature (C)',
       y = 'Rainfall (mm)',
       title = 'Sydney weather by season', ###
       subtitle = "Data from 2013 to 2018") ###
```


Sydney weather by season
Data from 2013 to 2018
Autumn
Spring
Summer
Winter
Rainfall (mm)
0
20
40
60
Temperature (C)
20
30
40

Customize

```r
ggplot(
  data = beaches,
  mapping = aes(
    x = temperature,
    y = rainfall
  )
) +
  geom_point(
    mapping = aes(color = season_name),
    size = 4,
    show.legend = FALSE
  ) +
  facet_wrap( ~ season_name) +
  theme_bw() +
  labs(x = 'Temperature (C)',
       y = 'Rainfall (mm)',
       title = 'Sydney weather by season',
       subtitle = "Data from 2013 to 2018") +
  theme(axis.title = element_text(size = 14), ###
        axis.text = element_text(size = 12), ###
        strip.text = element_text(size = 12)) ###
```


Sydney weather by season
Data from 2013 to 2018
Autumn
Spring
Summer
Winter
Rainfall (mm)
0
20
40
60
Temperature (C)
20
30
40

The grammar

- Data
- Aesthetics (or aesthetic mappings)
- Geometries (as layers) or Statistics (as computed layers)
- Facets
- Themes
- (Coordinates)
- (Scales)

Peeking under the hood . . .

We input the below items

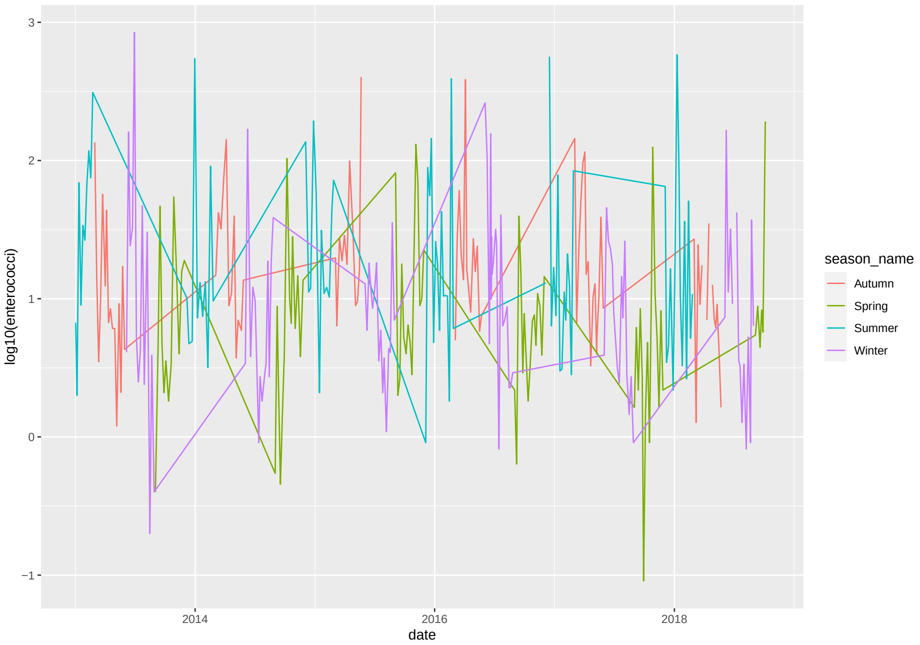
```
ggplot(
  data = beaches,
  aes(x = temperature,
      y = rainfall)
) +
  geom_point()
```

What's really run is . . .

```r
ggplot(
  data = beaches,
  mapping = aes(
    x = temperature, y = rainfall)
  ) +
layer(
  geom = "point",
  stat = "identity",
  position = "identity") +
facet_null() +
theme_grey() +
coord_cartesian() +
scale_x_continuous() +
scale_y_continuous()
```

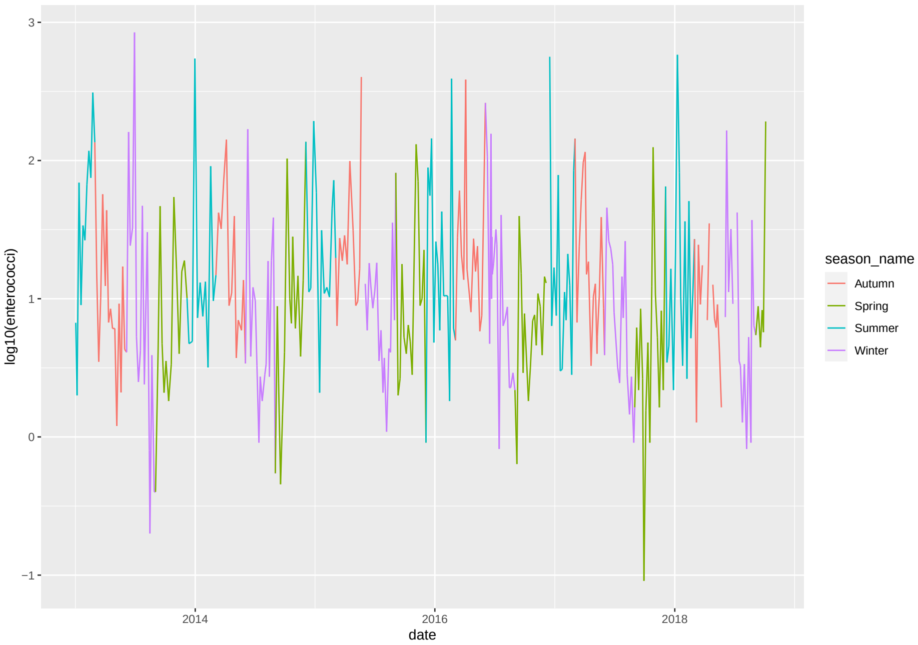
Exploring aesthetics: Mapping color

```r
ggplot(
  data=beaches,
  aes(x = date,
      y = log10(enterococci),
      color=season_name)
  ) +
  geom_line()
```


3
2
1
0
−1
log10(enterococci)
2014
2016
2018
date
season_name
Autumn
Spring
Summer
Winter

Adding groups to the mapping

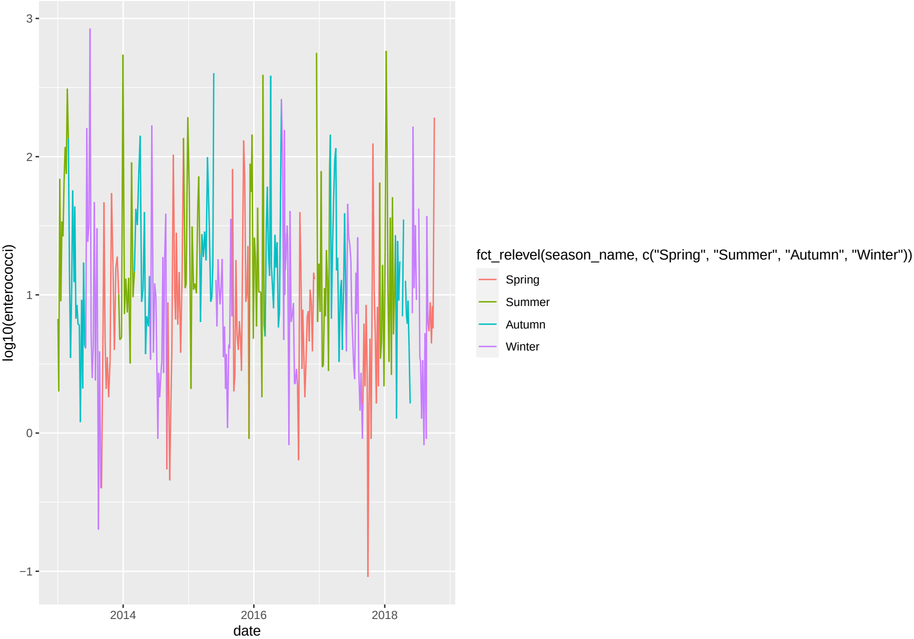
```r
ggplot(
  data=beaches,
  aes(x = date,
      y = log10(enterococci),
      color = season_name,
      group = 1) ###
) +
  geom_line()
```


log10(enterococci)
3
2
1
0
−1
2014
2016
2018
date
season_name
Autumn
Spring
Summer
Winter

Fixing the legend ordering

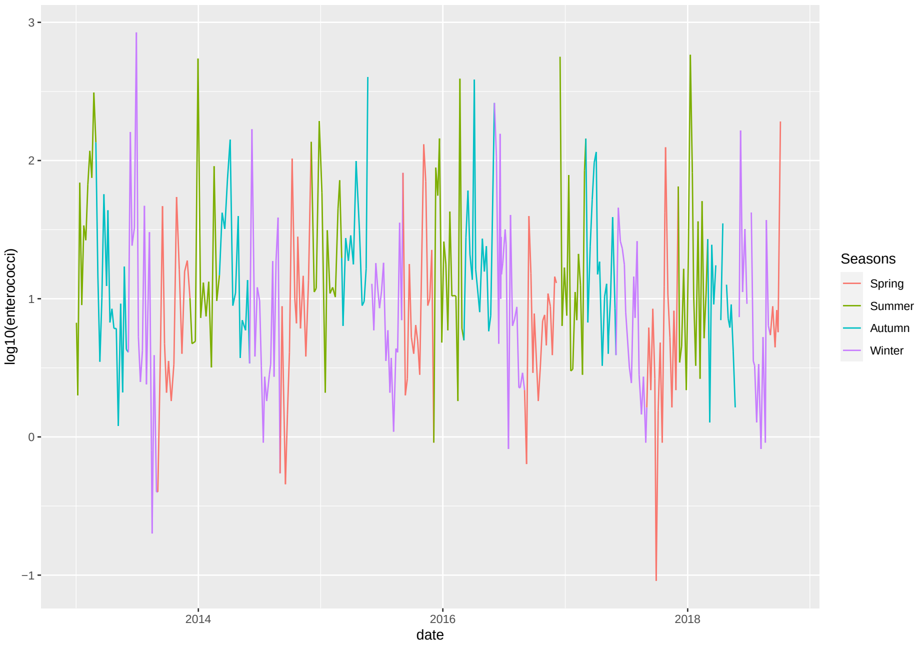
```r
ggplot(
  data=beaches,
  aes(x = date,
      y = log10(enterococci),
      color = fct_relevel(season_name,
                          c('Spring','Summer','Autumn','Winter')),
      group = 1)
) +
  geom_line()
```

Yikes!


3
2
1
0
−1
log10(enterococci)
2014
2016
2018
date
fct_relevel(season_name, c("Spring", "Summer", "Autumn", "Winter"))
Spring
Summer
Autumn
Winter

Fixing the legend ordering

```
ggplot(
  data=beaches,
  aes(x = date,
      y = log10(enterococci),
      color = fct_relevel(season_name,
            c('Spring','Summer','Autumn','Winter')),
      group = 1)
) +
  geom_line()+
  labs(color = 'Seasons') ###
```

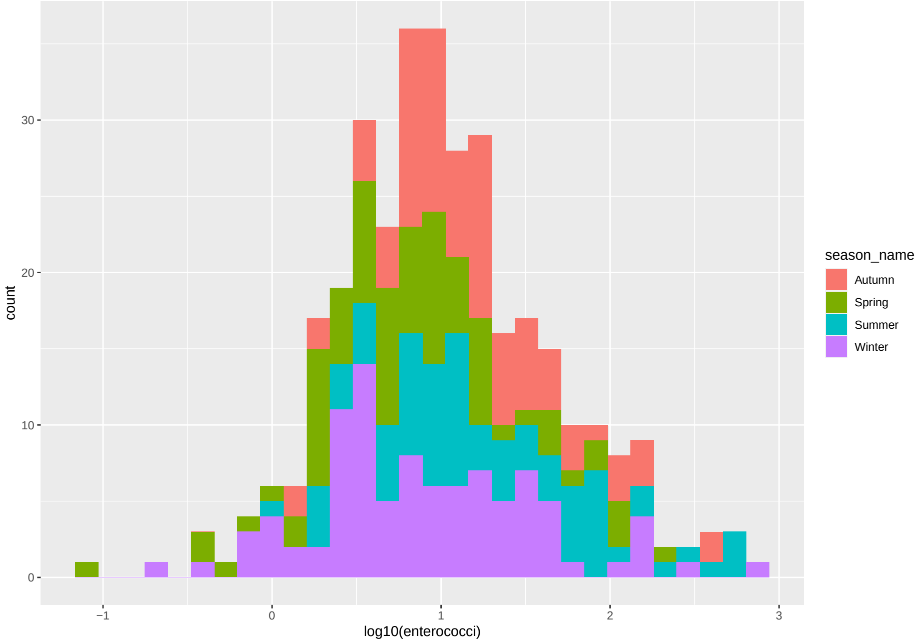


3
2
1
0
−1
log10(enterococci)
2014
2016
2018
date
Seasons
Spring
Summer
Autumn
Winter

You can also fill based on data

```
ggplot(
  data=beaches,
  aes(x = log10(enterococci),
      fill = season_name)
  )+
  geom_histogram()
```

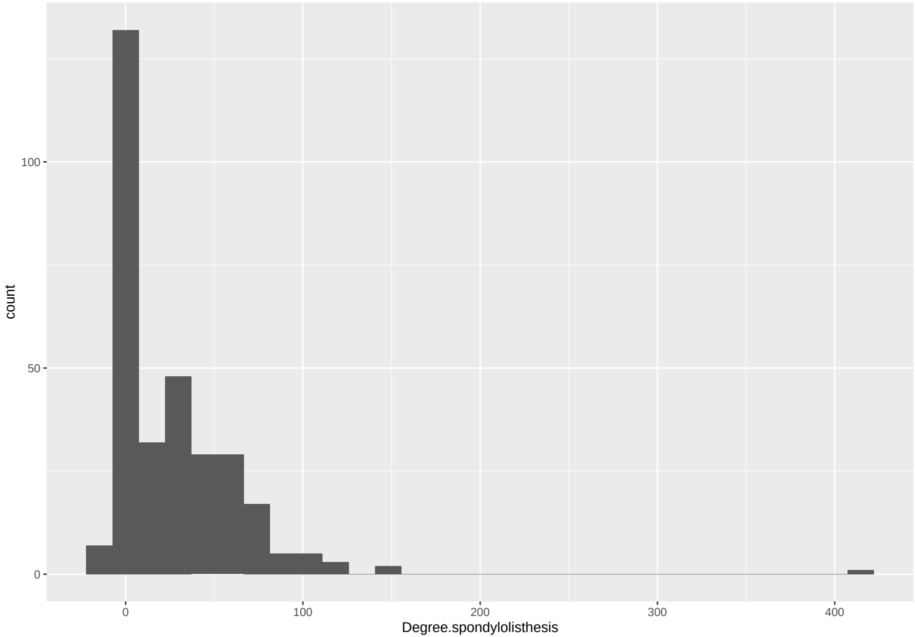
Works a little better



Exploring geometries: Univariate plots

```r
library(tidyverse)
library(rio)
dat_spine <- import('Dataset_spine.csv',
                    check.names = T)

ggplot(
  data=dat_spine,
  aes(x = Degree.spondylolisthesis)
  )+
  geom_histogram()
```

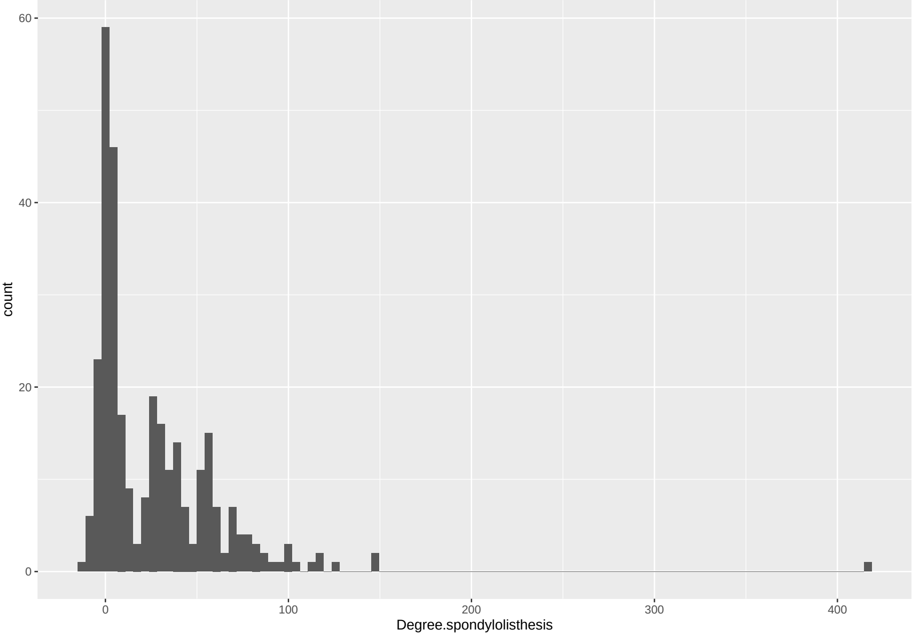
```
#> `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Histograms

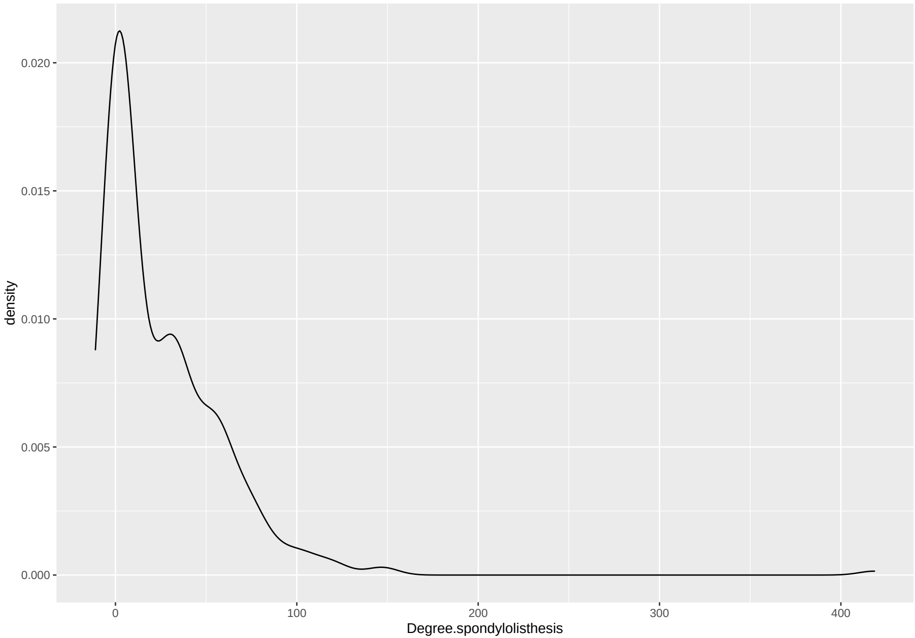
```
ggplot(
  data=dat_spine,
  aes(x = Degree.spondylolisthesis)
  )+
  geom_histogram(bins = 100)
```

Very different view of the data



Density plots

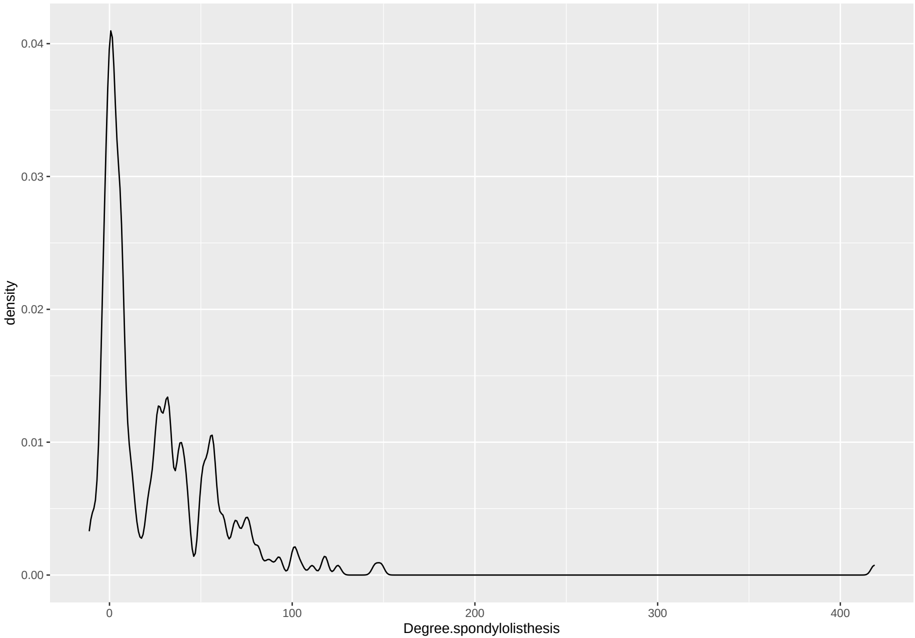
```r
ggplot(
  data=dat_spine,
  aes(x = Degree.spondylolisthesis)
  )+
  geom_density()
```



0.020
0.015
0.010
0.005
0.000
density
0
100
200
300
400
Degree.spondylolisthesis

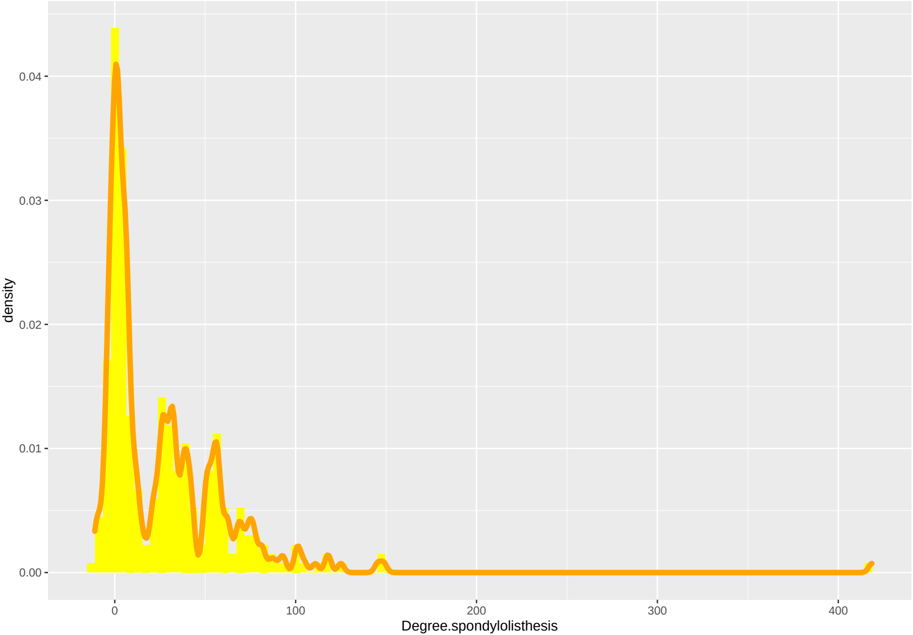
Density plots

```r
ggplot(
  data=dat_spine,
  aes(x = Degree.spondylolisthesis)
  )+
  geom_density(adjust = 1/5) # Use 1/5 the bandwidth
```


0.04
0.03
0.02
0.01
0.00
density
0
100
200
300
400
Degree.spondylolisthesis

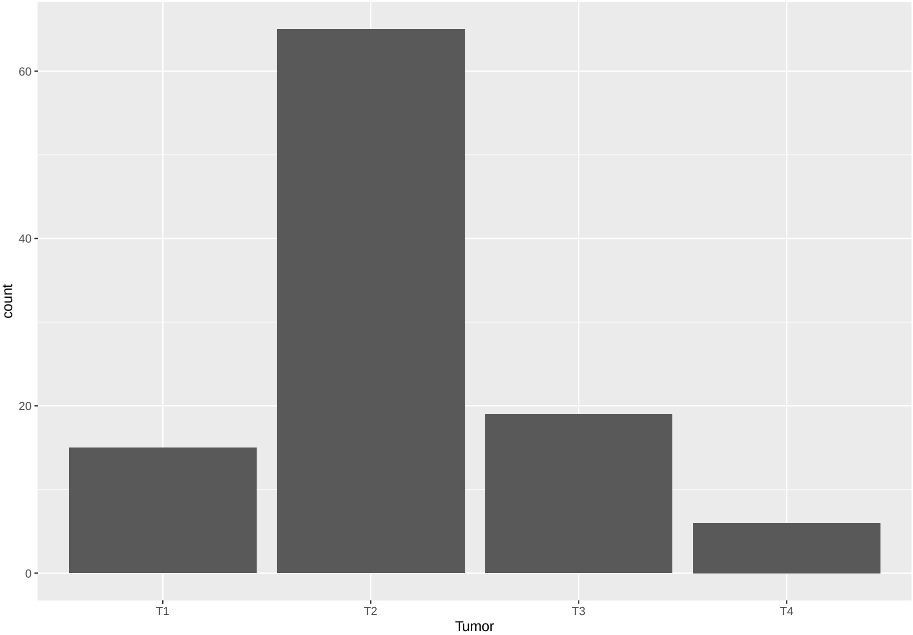
Layering geometries

```r
ggplot(
  data=dat_spine,
  aes(x = Degree.spondylolisthesis,
      y = stat(density))
  )+  # Re-scales histogram
  geom_histogram(bins = 100, fill='yellow') +
  geom_density(adjust = 1/5, color = 'orange', size = 2)
```


density
0.04
0.03
0.02
0.01
0.00
0
100
200
300
400
Degree.spondylolisthesis

Bar plots (categorical variable)

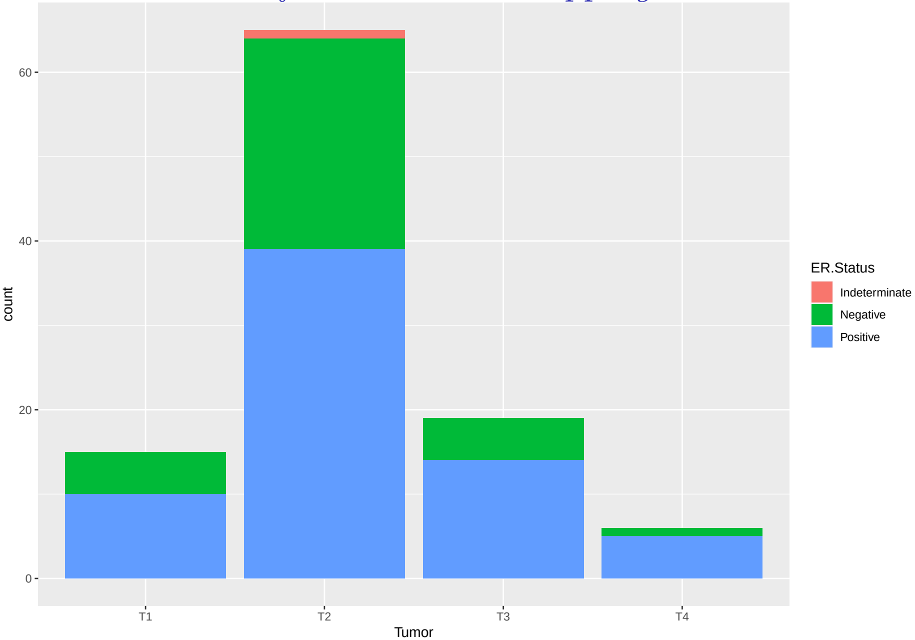
```r
dat_brca <-
  rio::import('clinical_data_breast_cancer_modified.csv',
                         check.names = T)
ggplot(
  data=dat_brca,
  aes(x = Tumor)
  )+
  geom_bar()
```


60
40
20
0
count
T1
T2
T3
T4
Tumor

Bar plots (categorical variable)

```
ggplot(
  data=dat_brca,
  aes(x = Tumor,
      fill = ER.Status)
  )+ #<<
  geom_bar()
```

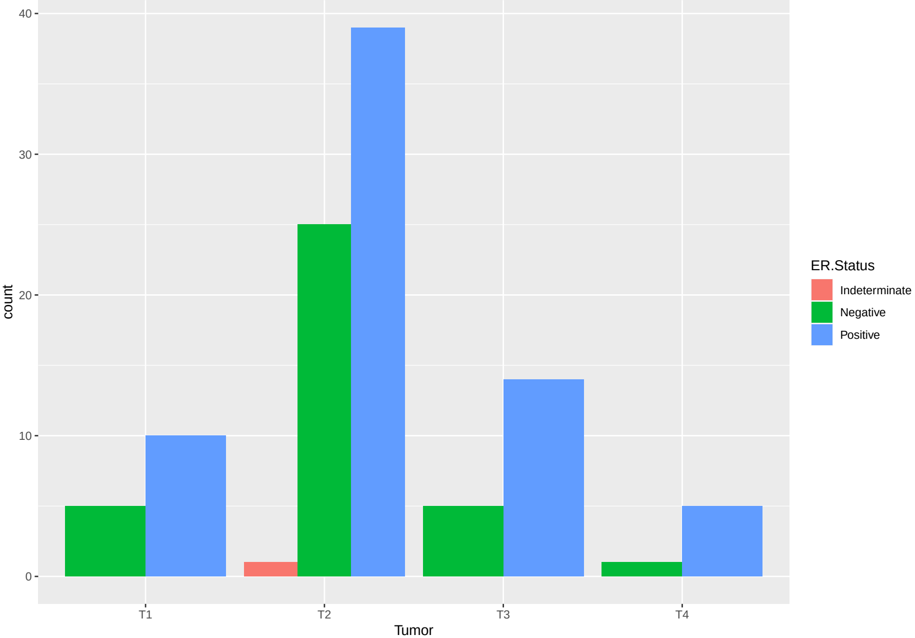
Add additional information via mapping



Bar plots (categorical variable)

```r
ggplot(
  data=dat_brca,
  aes(x = Tumor,
      fill = ER.Status)
  )+
  geom_bar(position = 'dodge')
    # Default is position = "stack"
```

Change the nature of the geometry



Graphing tabulated data

```r
(tabulated <- dat_brca %>% count(Tumor))
```

```
#>  # A tibble: 4 x 2
#>    Tumor     n
#>    <chr> <int>
#>  1 T1       15
#>  2 T2       65
#>  3 T3       19
#>  4 T4        6
```

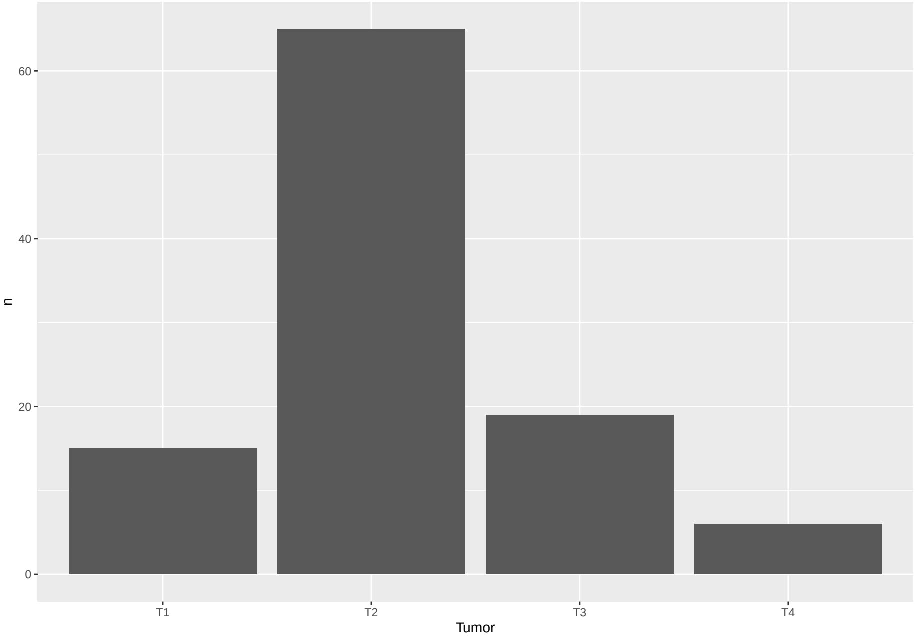
```r
ggplot(
  data = tabulated,
  aes(x = Tumor, y = n)
  ) +
  geom_bar()
```

```
#>  Error: stat_count() can only have an x or y aesthetic.
```

Graphing tabulated data

```
tabulated <- dat_brca %>% count(Tumor)
tabulated

ggplot(
  data = tabulated,
  aes(x = Tumor, y = n)
  ) +
  geom_bar(stat = 'identity') ###
```

- Here we need to change the default computation
- The barplot usually computes the counts (`stat_count`)
- We suppress that here since we have already done the computation


60
40
20
0
n
T1
T2
T3
T4
Tumor

Peeking under the hood

```
plt <- ggplot(
  data = tabulated,
  aes(x = Tumor, y = n)
  ) +
  geom_bar()

plt$layers
```

```
#>  [[1]]
#>  geom_bar: width = NULL, na.rm = FALSE, orientation = NA
#>  stat_count: width = NULL, na.rm = FALSE, orientation = NA
#>  position_stack
```

Peeking under the hood

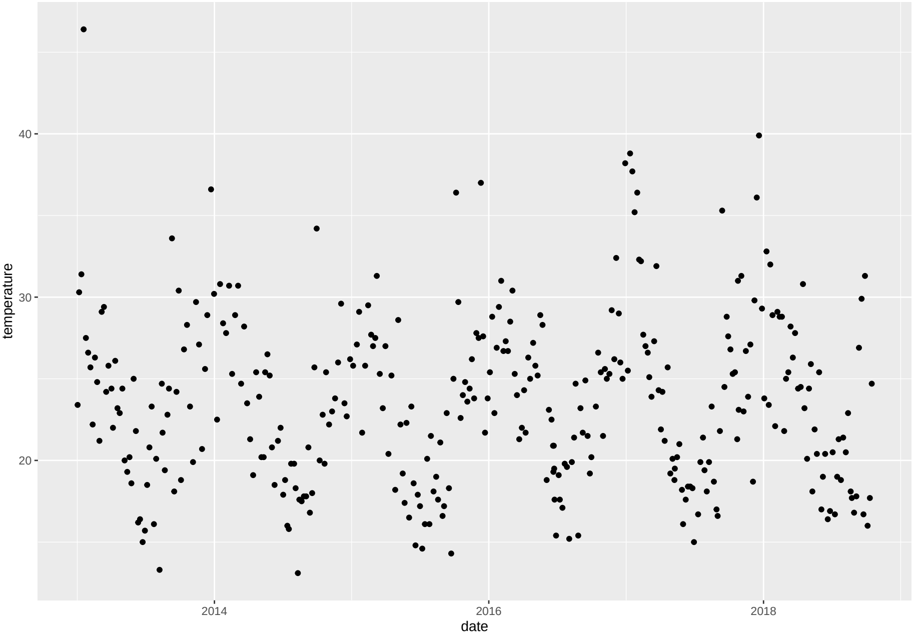
```r
plt <- ggplot(
  data = tabulated,
  aes(x = Tumor, y = n)) +
  geom_bar(stat = 'identity')

plt$layers
```

```
#>  [[1]]
#>  geom_bar: width = NULL, na.rm = FALSE, orientation = NA
#>  stat_identity: na.rm = FALSE
#>  position_stack
```

Each layer has a geometry, statistic and position associated with it

Bivariate plots: Scatter plots

```r
ggplot(
  data = beaches,
  aes(x = date, y = temperature)
  )+
  geom_point()
```

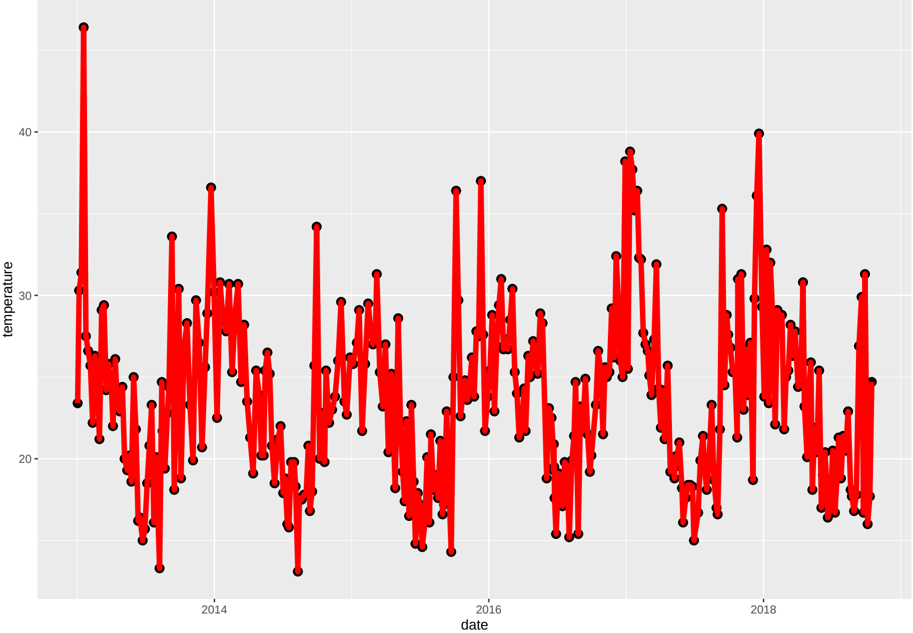


40
30
20
temperature
2014
2016
2018
date

Scatter plots

```
ggplot(
  data = beaches,
  aes(x = date, y = temperature, group=1)#Add the group argu.
  )+
  geom_point(color='black', size = 3) +
  geom_line(color='red',size=2)
```

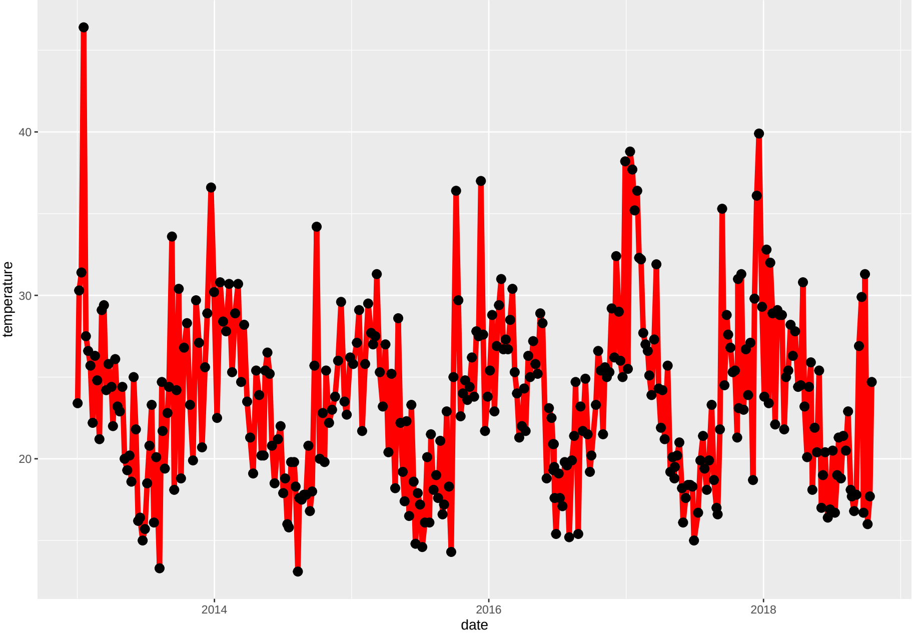
Layer points and lines



Scatter plots

```
ggplot(
  data = beaches,
  aes(x = date, y = temperature,group=1)
  )+
  geom_line(color='red',size=2) + ###
  geom_point(color='black', size = 3) ###
```

Order of laying down geometries matters

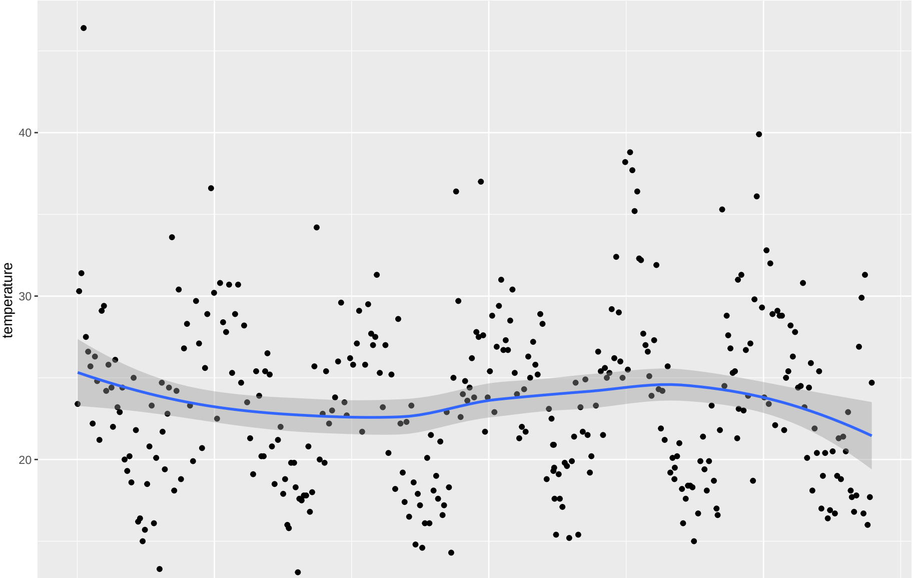


Doing some computations

```
ggplot(
  data = beaches,
  aes(x = date, y = temperature, group=1)
  ) +
  geom_point() +
  geom_smooth(method='loess')
```

Averages over 75% of the data

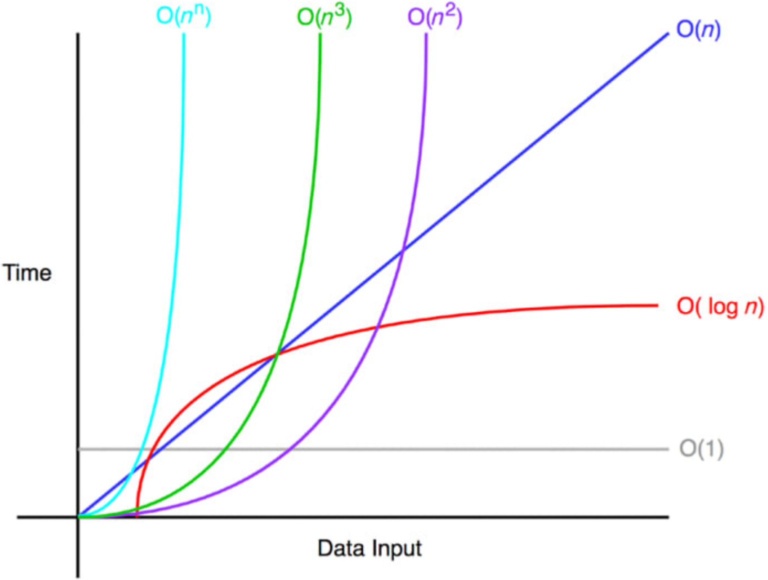
```
#>  `geom_smooth()` using formula 'y ~ x'
```



Doing some computations

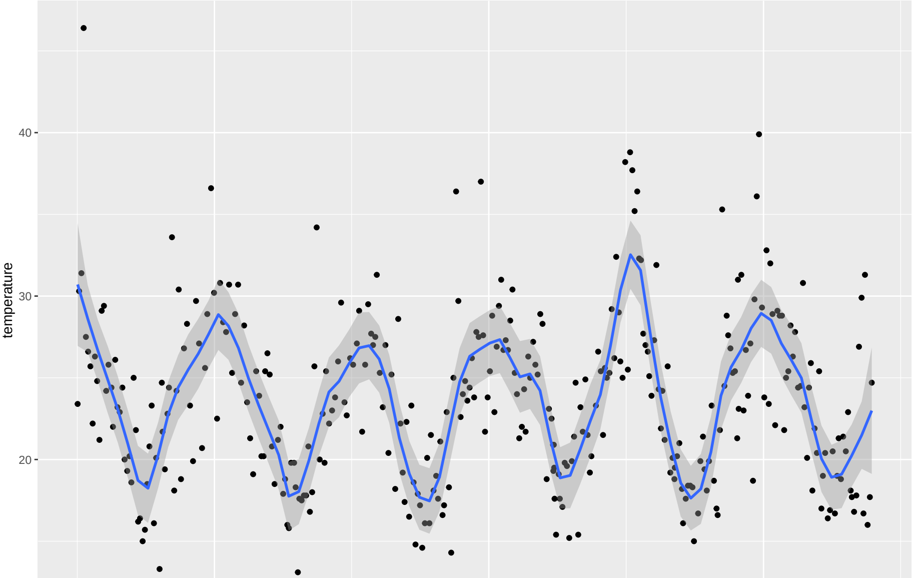
```r
ggplot(
  data = beaches,
  aes(x = date, y = temperature, group=1)
  ) +
  geom_point() +
  geom_smooth(method="loess",span = 0.1) ###
#?geom_smooth, kinda funny...?
```

Big O!



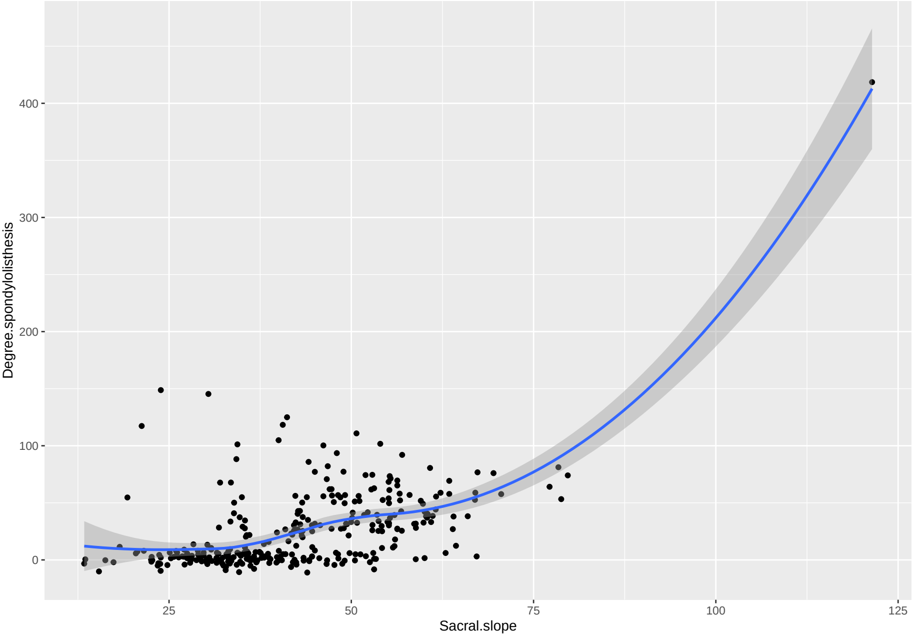
Averages over 10% of the data

```
#>  `geom_smooth()` using formula 'y ~ x'
```



Computations over groups

```r
ggplot(
  data = dat_spine,
  aes(x = Sacral.slope,
      y = Degree.spondylolisthesis)
  ) +
  geom_point() +
  geom_smooth()
```

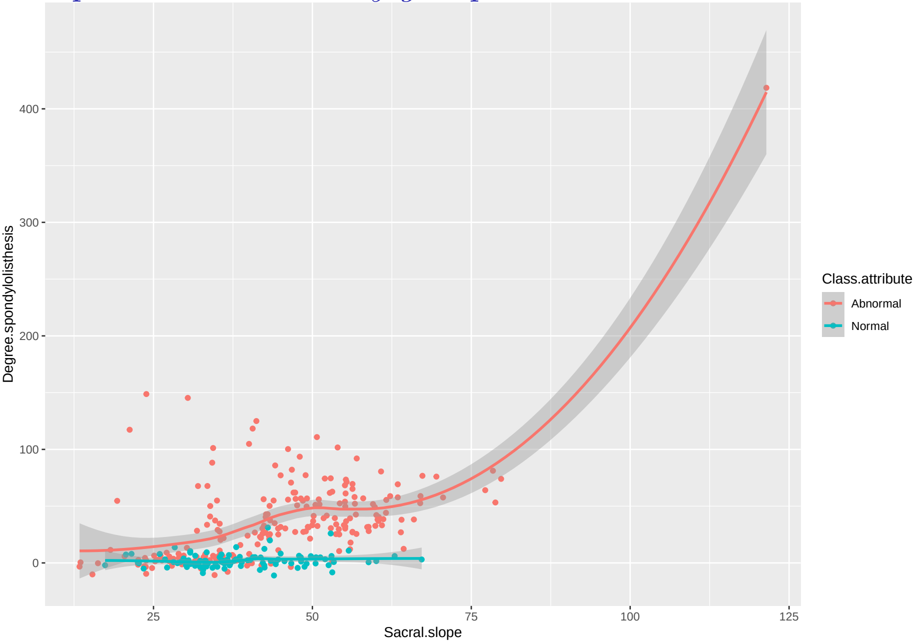


400
300
200
100
0
Degree.spondylolisthesis
25
50
75
100
125
Sacral.slope

Computations over groups

```
ggplot(
  data = dat_spine,
  aes(x = Sacral.slope,
      y = Degree.spondylolisthesis,
      color = Class.attribute) ##
  ) +
  geom_point() +
  geom_smooth()
```

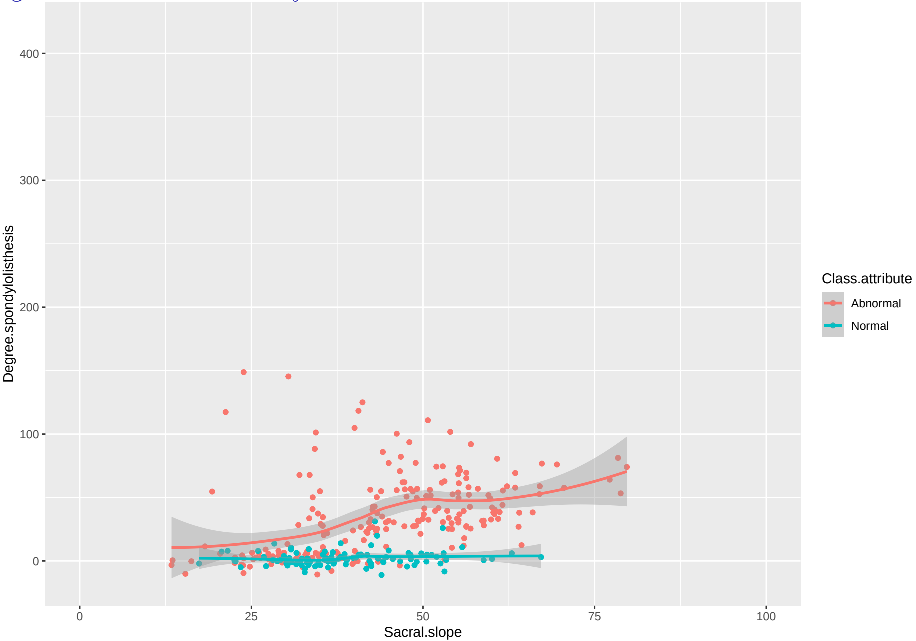
Computation is done by groups



Computations over groups

```r
ggplot(
  data = dat_spine,
  aes(x = Sacral.slope,
      y = Degree.spondylolisthesis,
      color = Class.attribute)
  ) +
  geom_point() +
  geom_smooth() +
  xlim(0, 100)#Changing the demonsions of the graphic
```

Ignore the outlier for now

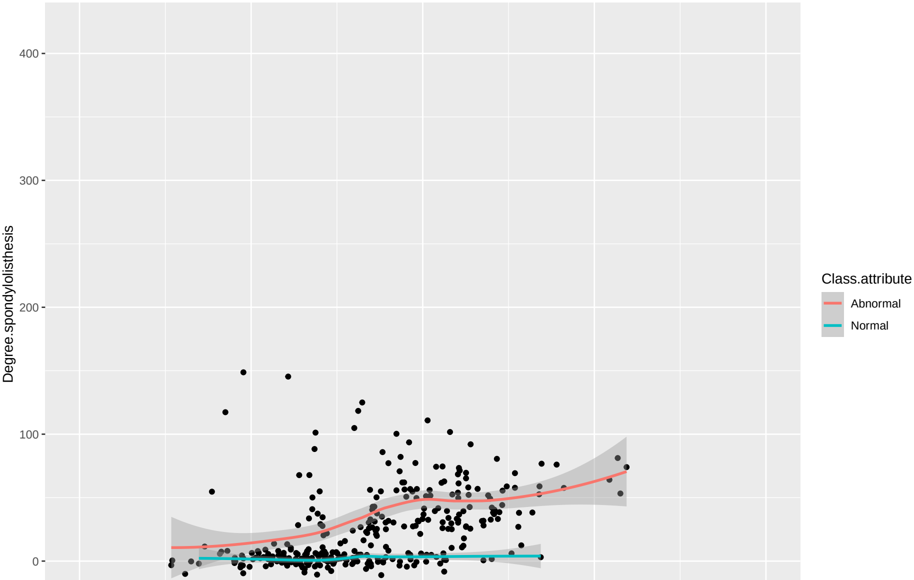


Computations over groups

```r
ggplot(
  data = dat_spine,
  aes(x = Sacral.slope,
      y = Degree.spondylolisthesis)
  ) +
  geom_point() +
  geom_smooth(aes(color = Class.attribute)) + #
  xlim(0, 100)
```

Only color-code the smoothers

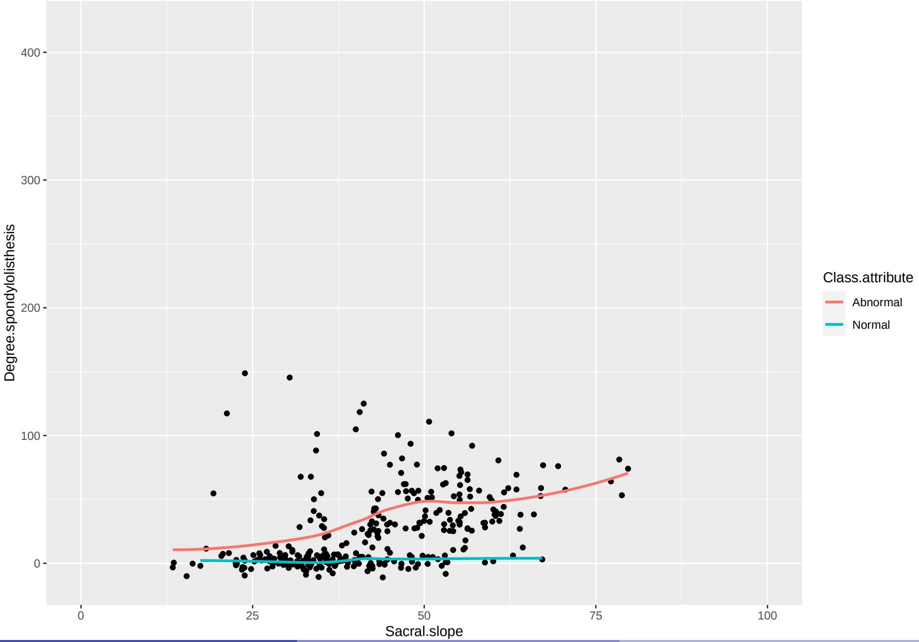
You can change the plot based on where you map the aesthetic



Computations over groups

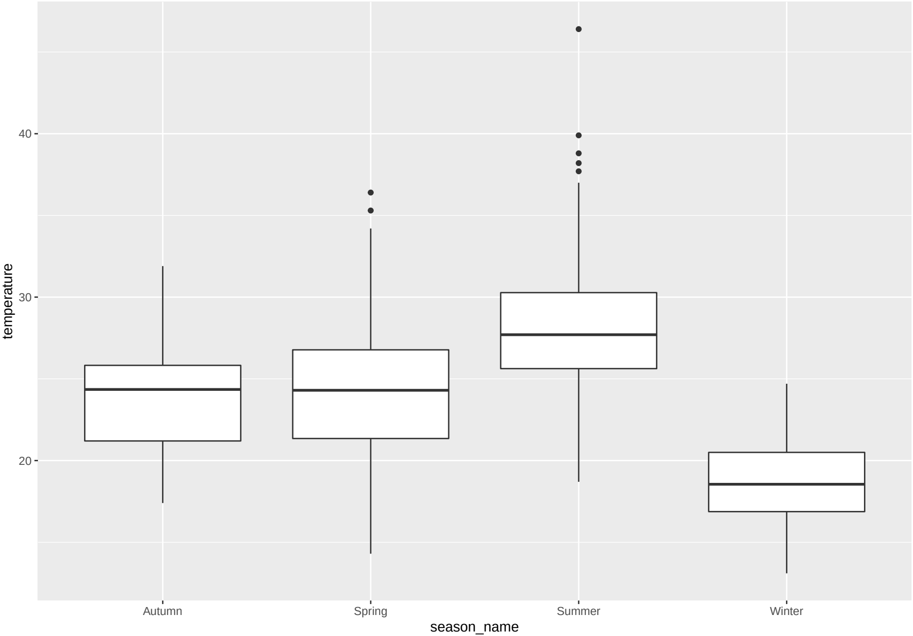
```r
ggplot(
  data = dat_spine,
  aes(x = Sacral.slope,
      y = Degree.spondylolisthesis)
  ) +
  geom_point() +
  geom_smooth(aes(color = Class.attribute),
              se = F) +
  #Turning off the confidence interval
  xlim(0, 100)
```

Looks a little cleaner



Box Plots

```r
ggplot(
  data = beaches,
  aes(x = season_name,
      y = temperature)
  ) +
  geom_boxplot()
```

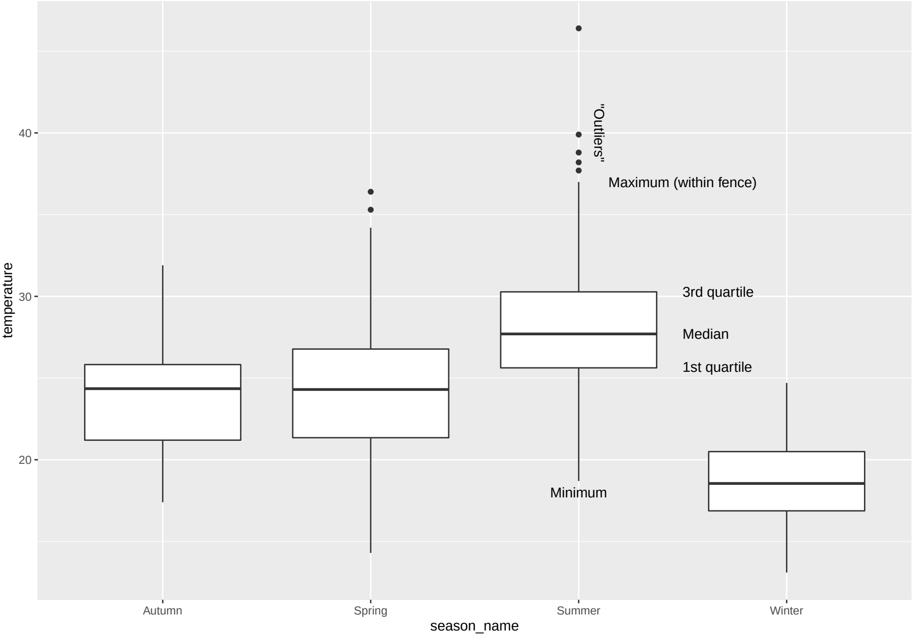


>W

are the components of a boxplot?

Box Plots

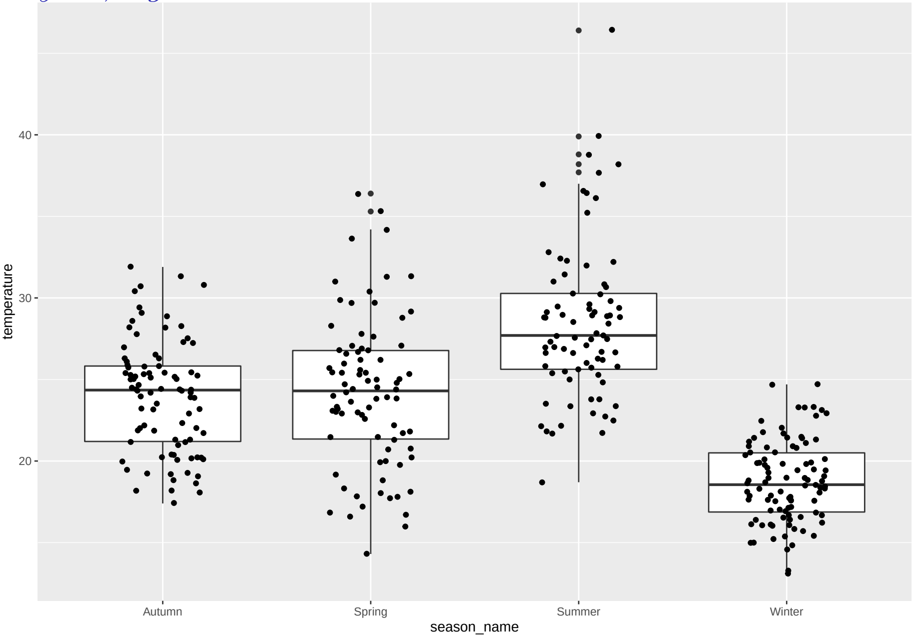
```r
ggplot(
  data = beaches,
  aes(x = season_name,
      y = temperature)
  ) +
  geom_boxplot()
```


temperature
40
30
20
Autumn
Spring
Summer
Winter
season_name
"Outliers"
Maximum (within fence)
3rd quartile
Median
1st quartile
Minimum

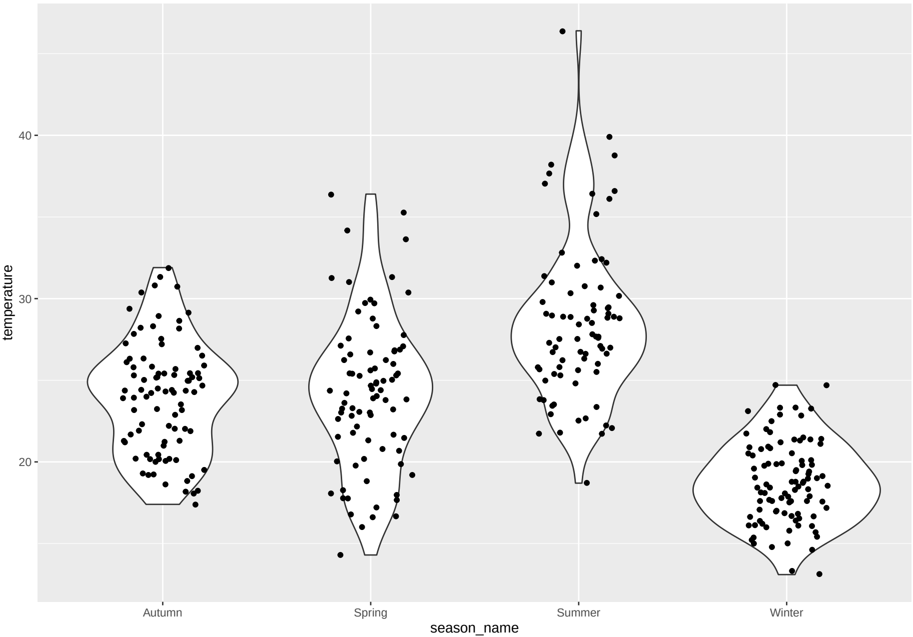
Layers, again

```r
ggplot(
  data = beaches,
  aes(x = season_name,
      y = temperature)
  ) +
  geom_boxplot() +
  geom_jitter(width = 0.2)
```

Layers, again



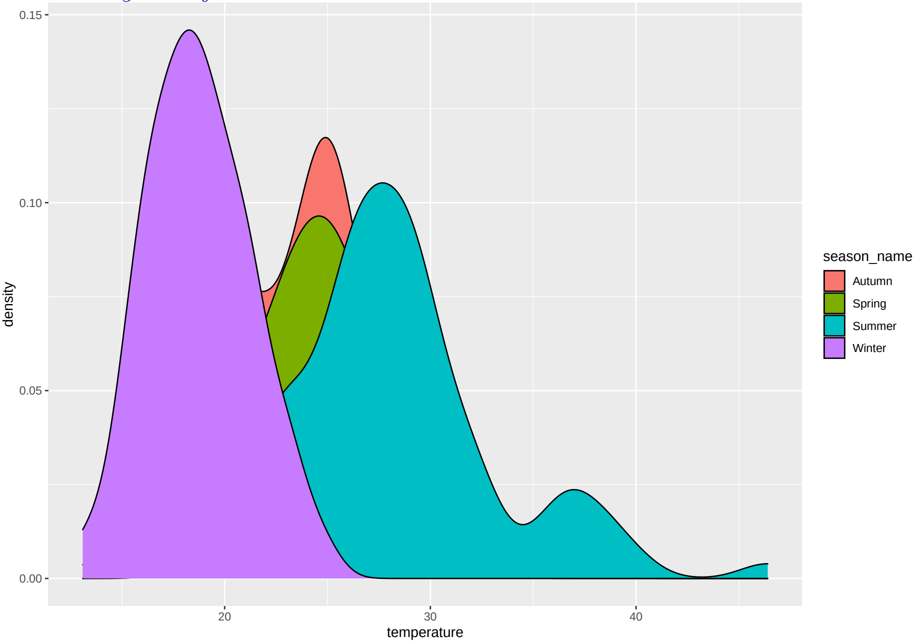
#:


temperature
40
30
20
Autumn
Spring
Summer
Winter
season_name

Exploring grouped data

```
ggplot(
  data = beaches,
  aes(x = temperature,
      fill = season_name)
  ) +
  geom_density()
```

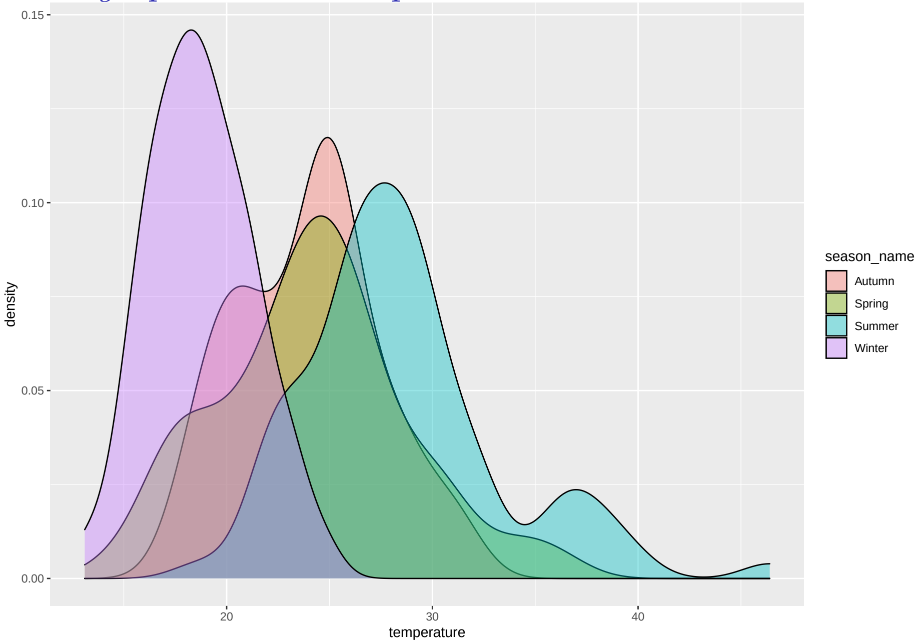
Not very useful



Overlaying graphs

```r
ggplot(
  data = beaches,
  aes(x = temperature,
      fill = season_name)
  ) +
  geom_density(alpha = 0.4)# Changes the transparency
```

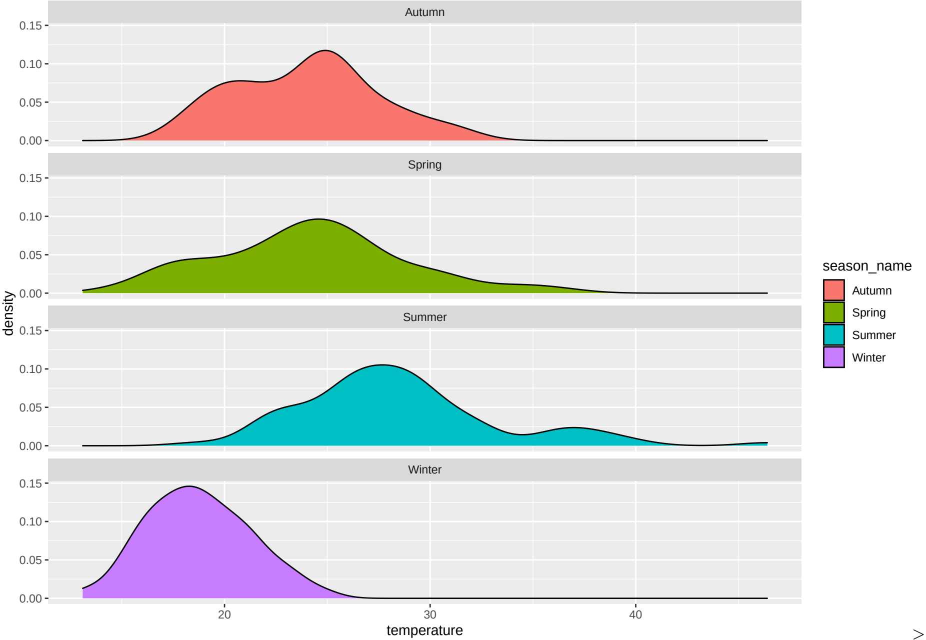
Make graphs more transparent



Exploding graphs

```
ggplot(
  data = beaches,
  aes(x = temperature,
      fill = season_name)
  ) +
  geom_density() +
  facet_wrap(~ season_name, ncol = 1) ###
```

This is called “small multiples” (Tufte)

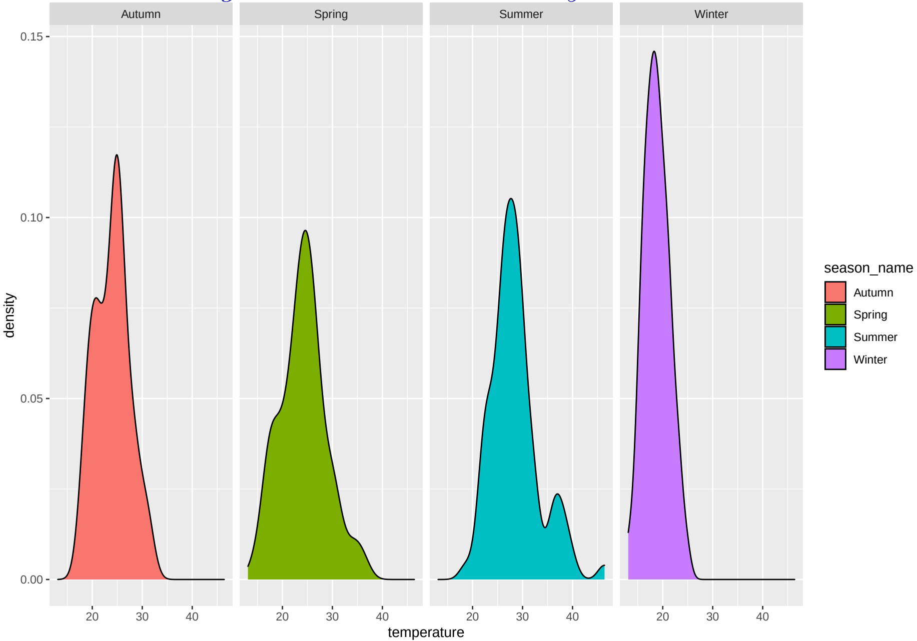


Notice that all the graphs have the same x-axis. This is a good thing

Exploding graphs

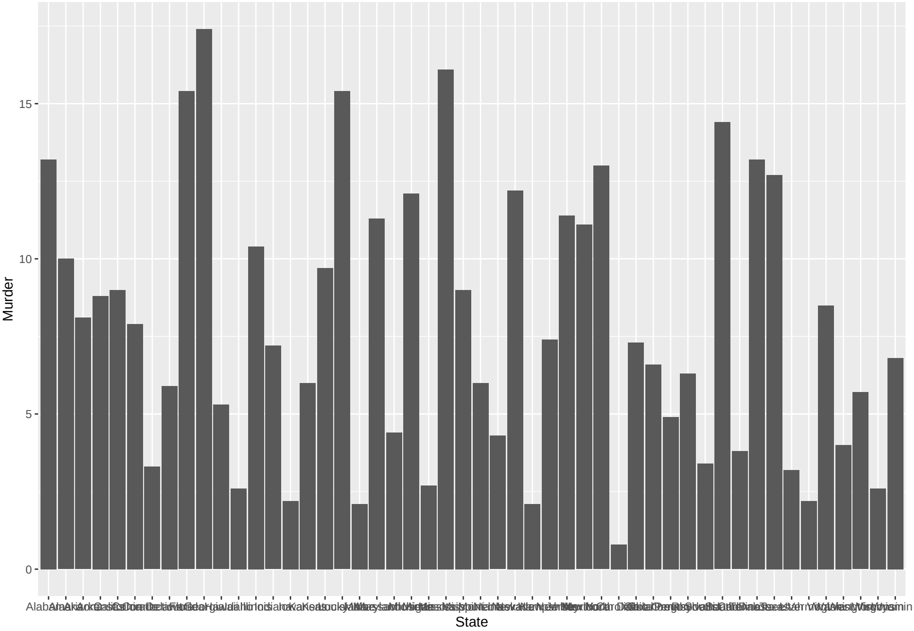
```r
ggplot(
  data = beaches,
  aes(x = temperature,
      fill = season_name)
  ) +
  geom_density() +
  facet_wrap(~ season_name, nrow = 1) ###
```

We can arrange them the other way too



Order and orientation: Arrests in the USA in 1973

```r
arrests <- import('USArrests.csv')
ggplot(
  data = arrests,
  aes(x = State,
      y = Murder)
  ) +
  geom_bar(stat = 'identity')
```



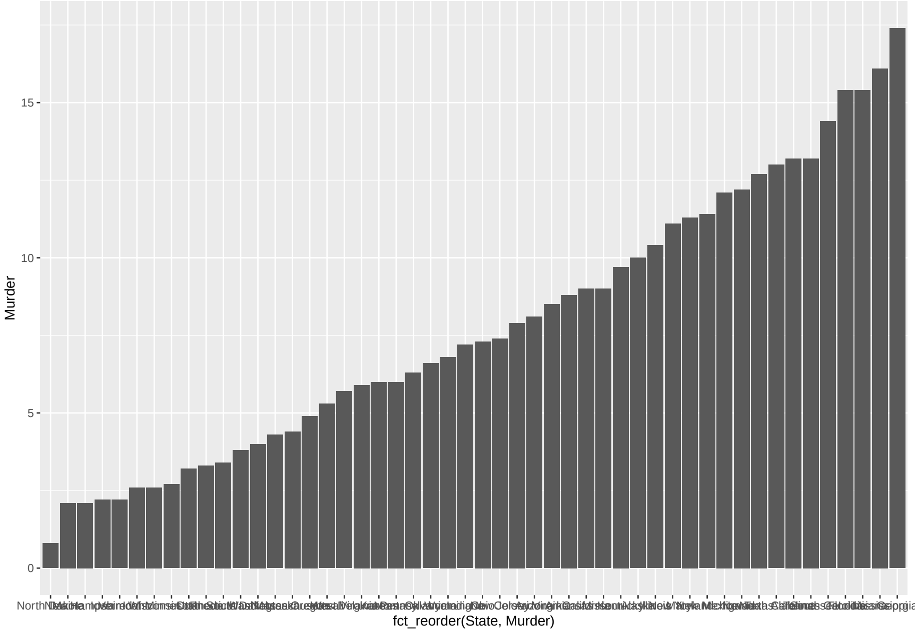
>H

to read, there is no ordering, and x-labels can't be seen

Arrests in the USA in 1973

```r
ggplot(
  data = arrests,
  aes(x = fct_reorder(State, Murder), #Order by murder rate
      y = Murder)
  ) +
  geom_bar(stat = 'identity')
```

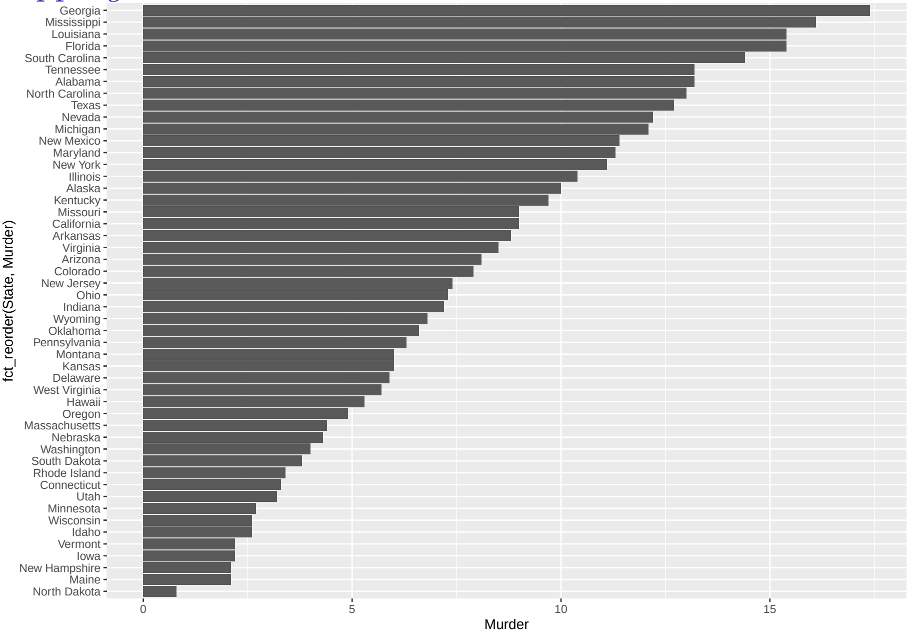
Arrest in the USA in 1973



Arrests in the USA in 1973

```
ggplot(
  data = arrests,
  aes(x = fct_reorder(State, Murder), # Order by murder rate
      y = Murder)
  ) +
  geom_bar(stat = 'identity') +
  coord_flip() #Flipping the coordinates
```

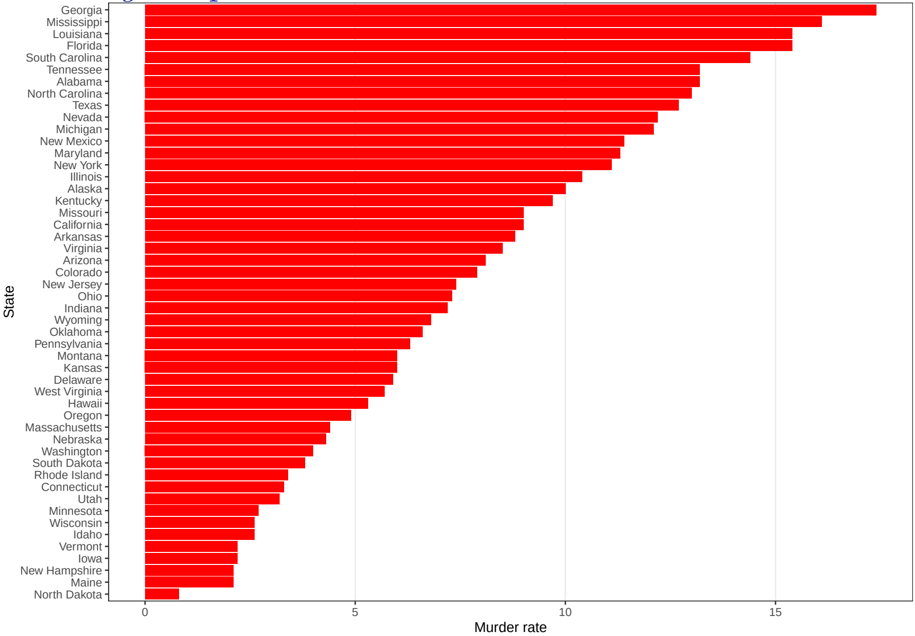
Flipping the axes makes the states readable



Arrests in the USA in 1973

```r
ggplot(
  data = arrests,
  aes(x = fct_reorder(State, Murder), # Order by murder rate
      y = Murder)
  ) +
  geom_bar(stat = 'identity', fill="red") +
  labs(x = 'State', y = 'Murder rate') + # Adding labels
  theme_bw() +# Theme
  theme(panel.grid.major.y = element_blank(),#
        panel.grid.minor.x = element_blank()) +
  coord_flip()# Flip last
```

Cleaning it up a little



Themes

ggplot comes with a default color scheme. There are several other schemes available

- `scale_*_brewer` uses the ColorBrewer palettes
- `scale_*_gradient` uses gradients
- `scale_*_distill` uses the ColorBrewer palettes, for continuous outcomes

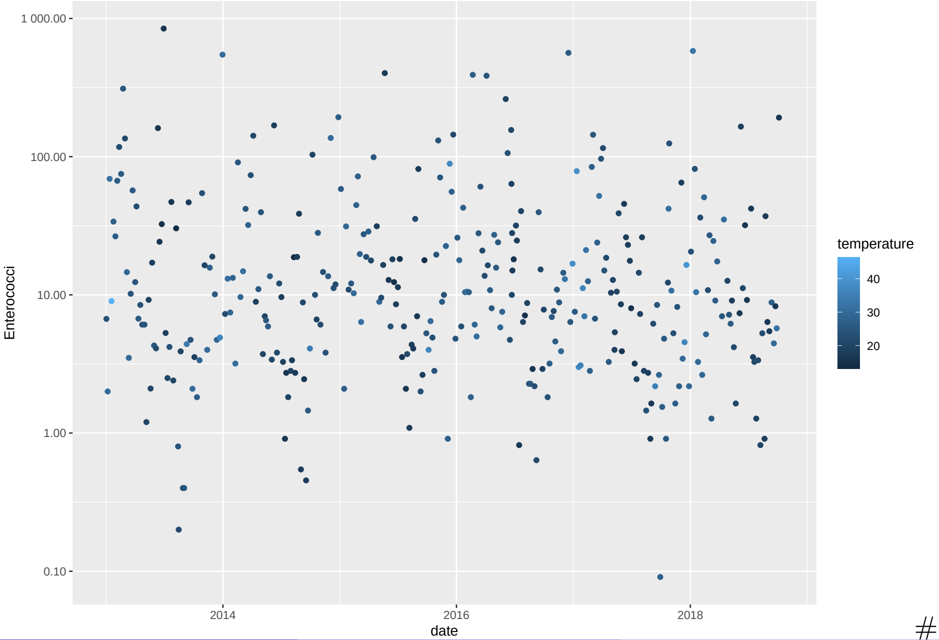
  *Here * can be `color` or `fill`, depending on what you want to color*

  Note `color` refers to the outline, and `fill` refers to the inside

No Theme

```r
ggplot(
  data = beaches,
  aes(x = date, y = enterococci,
      color = temperature)
  ) +
  geom_point() +
  scale_y_log10(name = 'Enterococci',
                label = scales::number_format(digits=3))
```

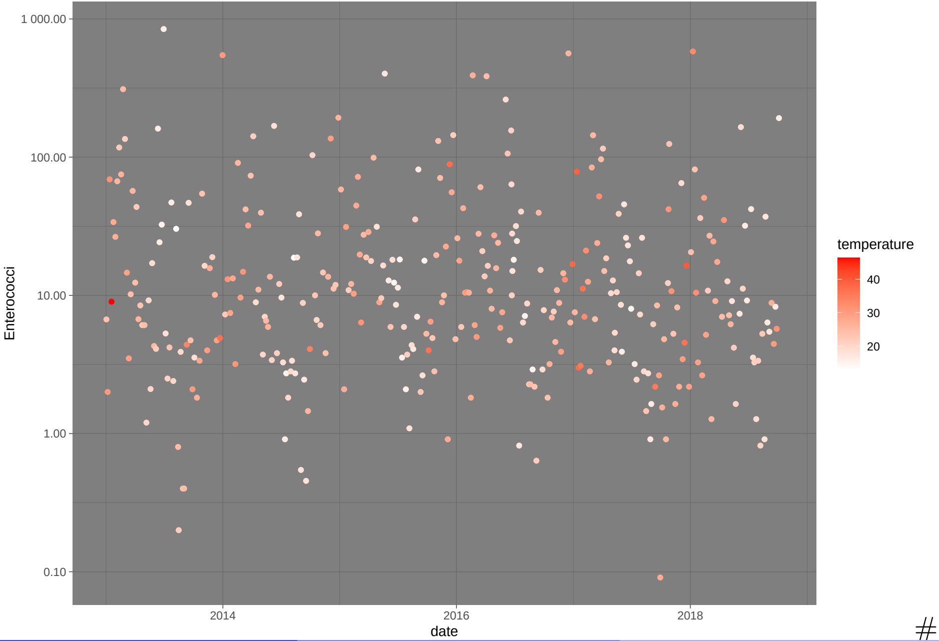
No Theme



1 000.00
100.00
10.00
1.00
0.10
Enterococci
2014
2016
2018
date
temperature
40
30
20

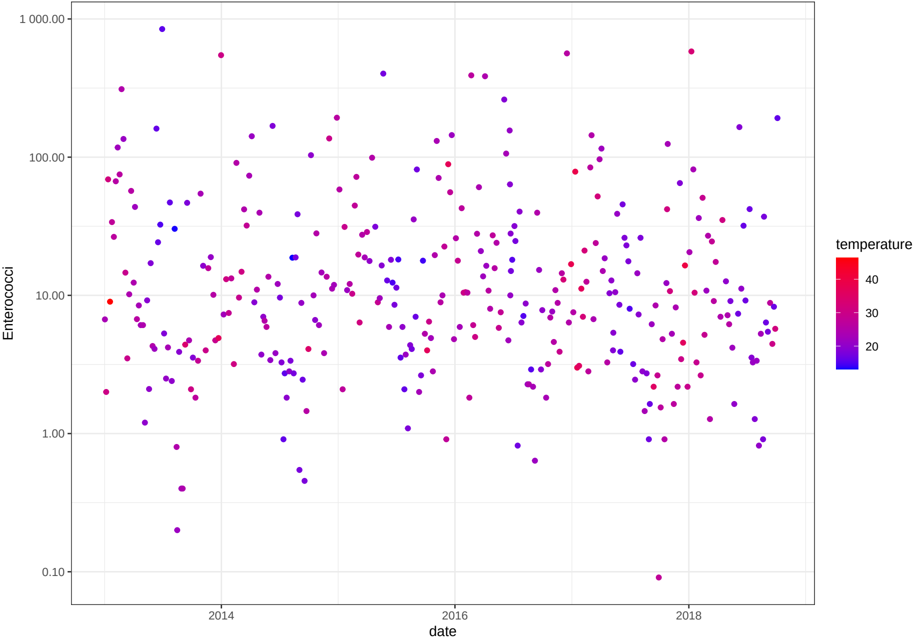
#=

Dark


1 000.00
100.00
10.00
1.00
0.10
Enterococci
2014
2016
2018
date
temperature
40
30
20

#

Black and White



Themes

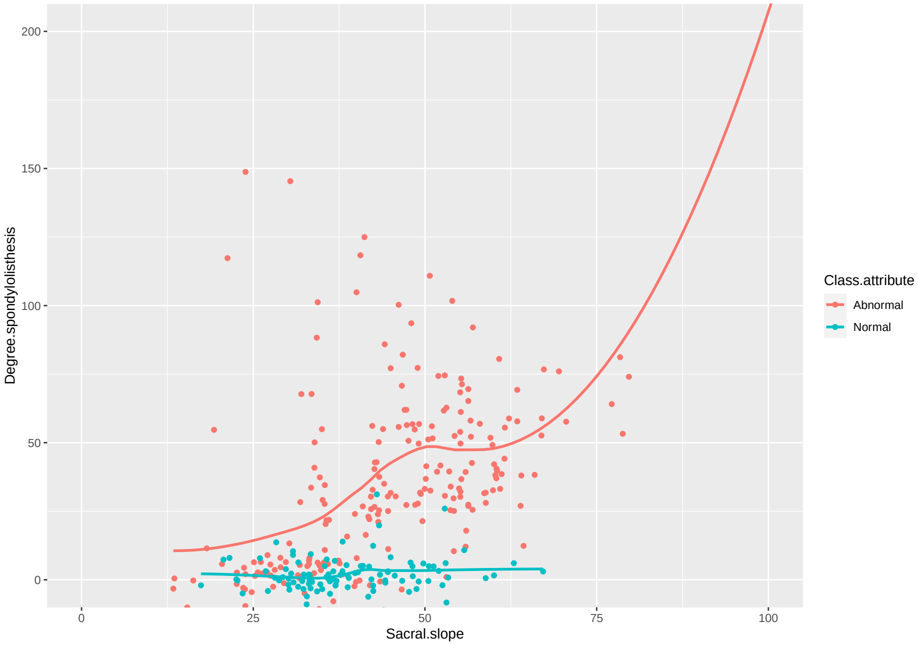
You can create your own custom themes to keep a unified look to your graphs

ggplot comes with

- theme_classic
- theme_bw
- theme_void
- theme_dark
- theme_gray
- theme_light
- theme_minimal

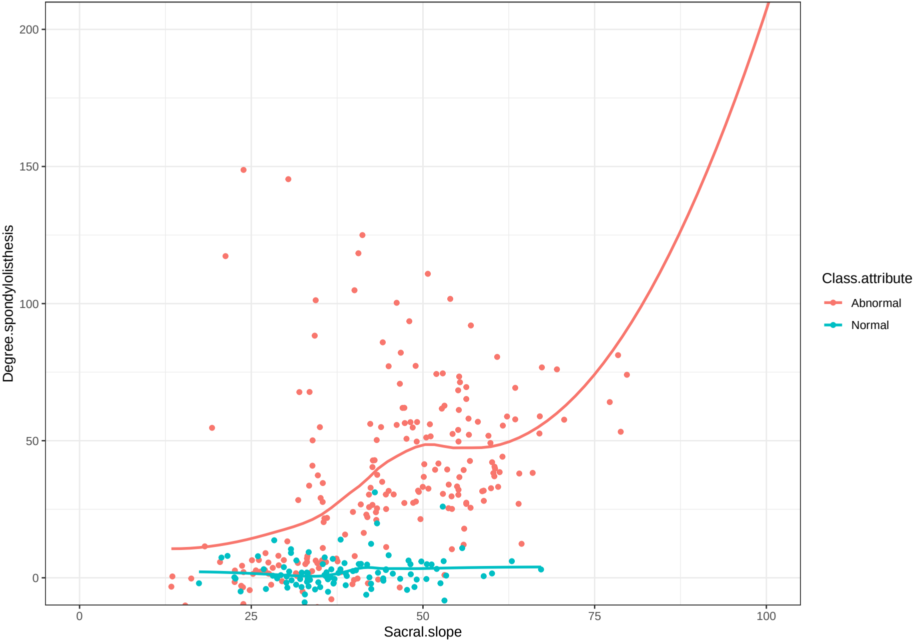
Create your own

```r
ggplot(
  data = dat_spine,
  aes(x = Sacral.slope, y = Degree.spondylolisthesis,
      color = Class.attribute)
  ) +
  geom_point() +
  geom_smooth(se = F) +
  coord_cartesian(xlim = c(0, 100),
                  ylim = c(0,200))
```


Degree.spondylolisthesis
200
150
100
50
0
0
25
50
75
100
Sacral.slope
Class.attribute
Abnormal
Normal

Create your own

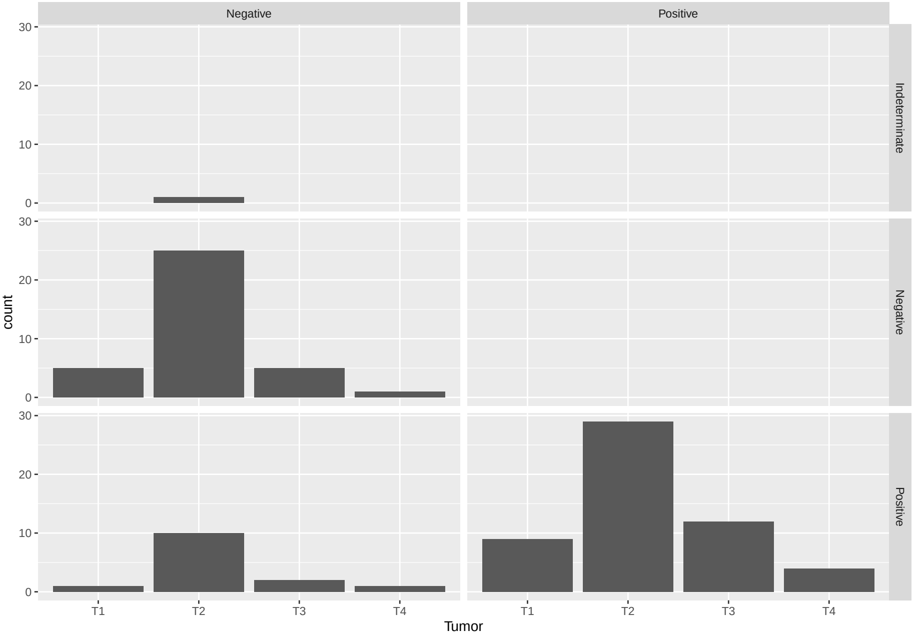
```
my_theme <- function(){
  theme_bw()
}

ggplot(
  data = dat_spine,
  aes(x = Sacral.slope, y = Degree.spondylolisthesis,
      color = Class.attribute)
  ) +
  geom_point() +
  geom_smooth(se = F) +
  coord_cartesian(xlim = c(0, 100),
                  ylim = c(0,200)) +
  my_theme() # Just Black and White
```



Degree.spondylolisthesis
200
150
100
50
0
0
25
50
75
100
Sacral.slope
Class.attribute
Abnormal
Normal

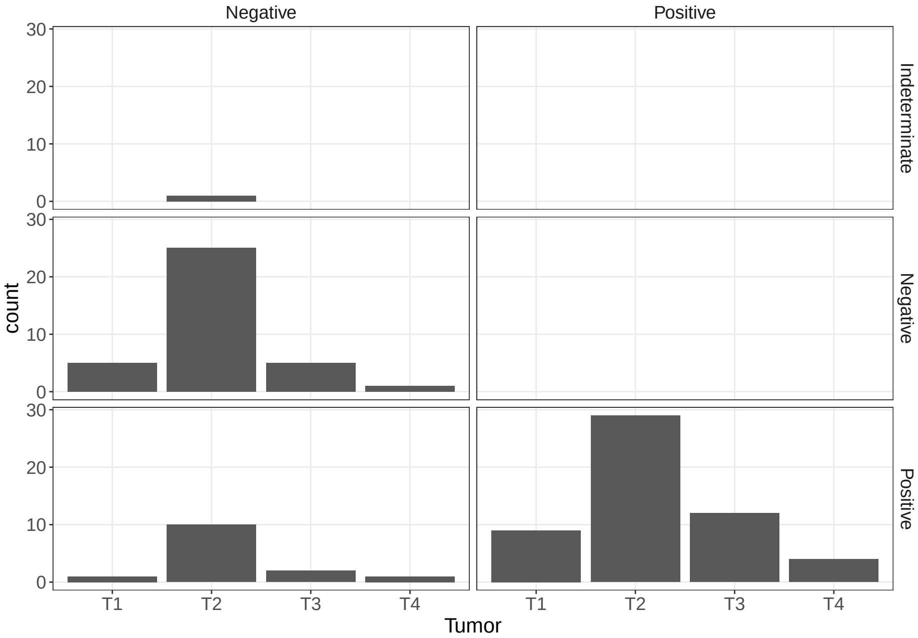
Create your own

```r
my_theme <- function(){
  theme_bw() +
  theme(axis.text = element_text(size = 14),
        axis.title = element_text(size = 16),
        panel.grid.minor = element_blank(),
        strip.text = element_text(size=14),
        strip.background = element_blank())
}

ggplot(
  data = dat_brca,
  aes(x = Tumor))+
  geom_bar() +
  facet_grid(rows = vars(ER.Status),
             cols = vars(PR.Status))
```


Negative
Positive
Indeterminate
Negative
Positive
count
0
10
20
30
T1
T2
T3
T4
Tumor

Create your own

```r
ggplot(
  data = dat_brca,
  aes(x = Tumor)
  )+
  geom_bar() +
  facet_grid(rows = vars(ER.Status),
             cols = vars(PR.Status)) +
  my_theme()
```


Negative
Positive
Indeterminate
Negative
Positive
count
30
20
10
0
T1
T2
T3
T4
Tumor

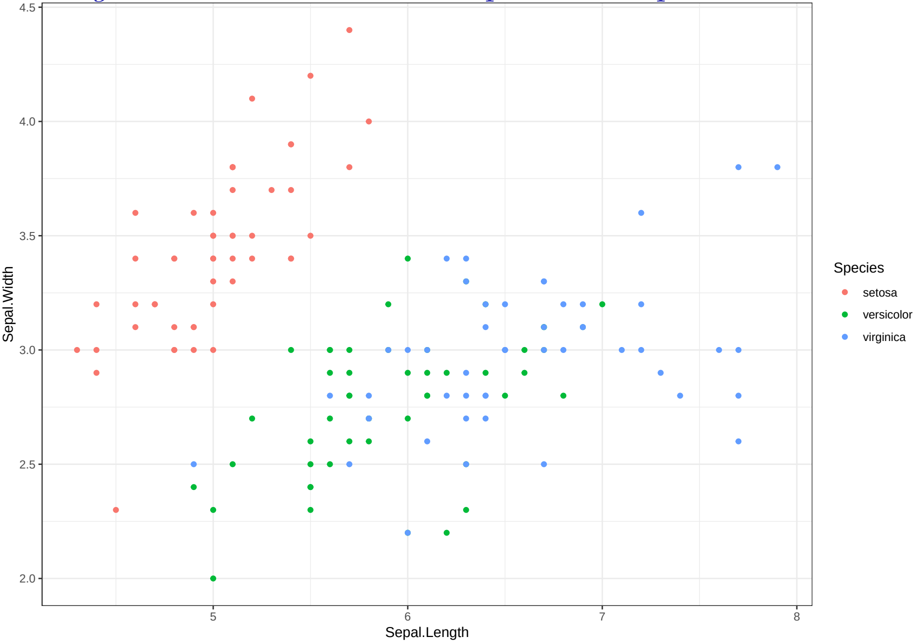
Annotations: Stand-alone stories

- Data visualization to stand on its own
- Relevant information should be placed on the graph
- However, you need to balance the information content with real estate
  - Don't clutter the graph and make it not readable

Adding derived statistics to a plot: Group means

```r
ggplot(iris,
       aes(x = Sepal.Length,
           y = Sepal.Width,
           color = Species)
       )+
  geom_point()+
  theme_bw()
```

Adding derived statistics to a plot: Group means



Adding derived statistics to a plot: Group means

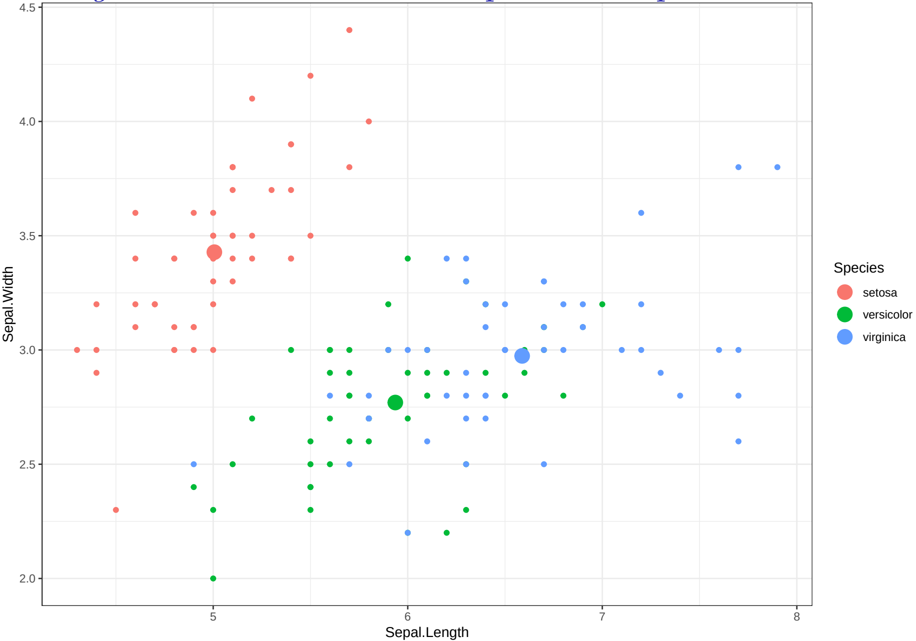
```
means <- iris %>% group_by(Species) %>%
  summarize_at(vars(starts_with('Sepal')),
               mean)
means
```

```
#>  # A tibble: 3 x 3
#>    Species    Sepal.Length Sepal.Width
#>    <fct>             <dbl>       <dbl>
#>  1 setosa             5.01        3.43
#>  2 versicolor         5.94        2.77
#>  3 virginica          6.59        2.97
```

Adding derived statistics to a plot: Group means

```r
ggplot(iris,
       aes(x = Sepal.Length,
           y = Sepal.Width,
           color = Species)
       )+
  geom_point() +
  geom_point(data = means,
             size=5) +
  theme_bw()
```

Adding derived statistics to a plot: Group means



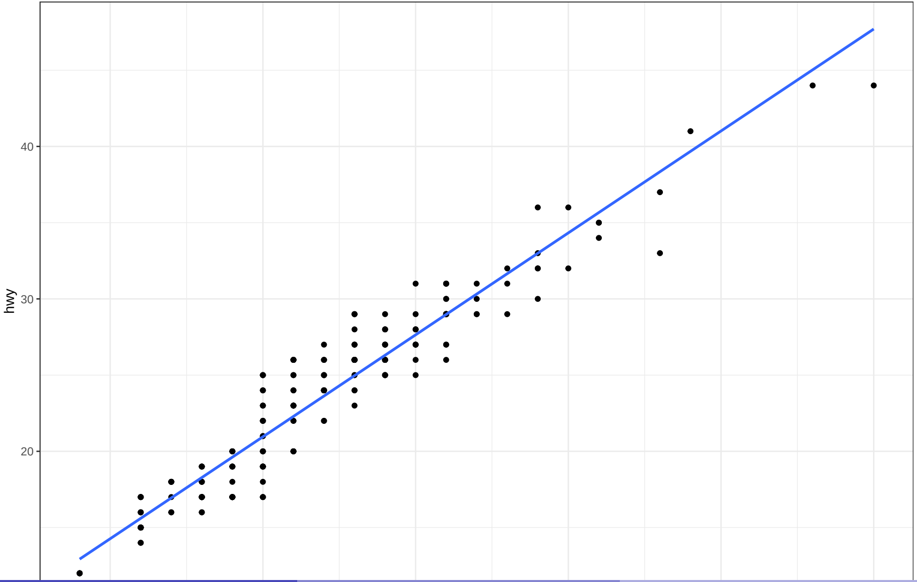
Adding regression metrics

Regress highway mileage on city mileage (data: mpg)

```r
mod1 <- lm(hwy ~ cty, data = mpg)
r2 <- broom::glance(mod1) %>% pull(r.squared)

ggplot(mpg,
       aes(x = cty, y = hwy)
       )+
  geom_point() +
  geom_smooth(method = 'lm', se=F) +
  theme_bw()
```

Adding regression metrics

```
#>  `geom_smooth()` using formula 'y ~ x'
```



Regress highway mileage on city mileage

```
mod1 <- lm(hwy ~ cty, data = mpg)
r2 <- broom::glance(mod1) %>% pull(r.squared) %>%#pull is par
  round(., 2)#part of base R, rounding behind the . by 2

ggplot(mpg,
       aes(x = cty, y = hwy))+
  geom_point() +
  geom_smooth(method = 'lm', se=F)+
  annotate(geom='text',
           x = 15, y = 40,
           label=glue::glue("R^2 == {r}",r=r2),
           size=6,
           parse=T) +
  theme_bw()
```

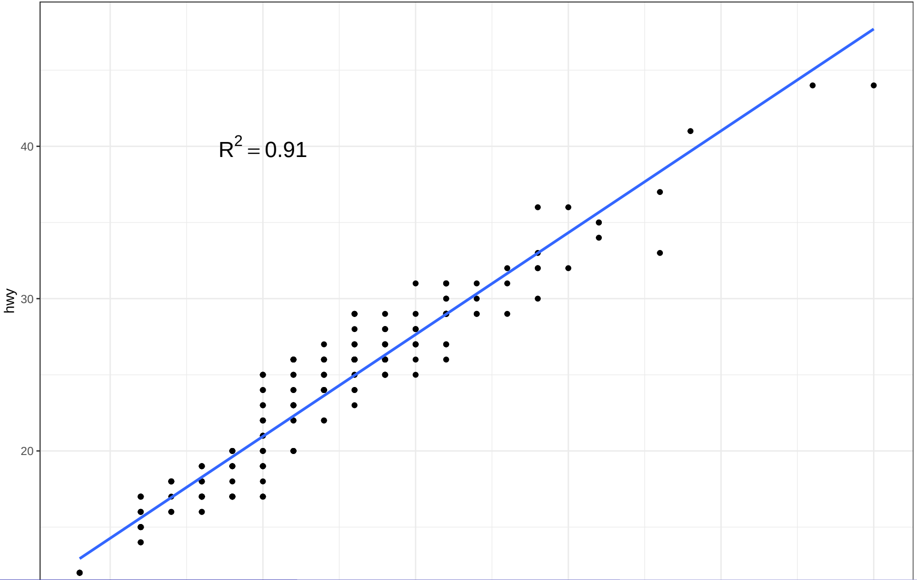
Glance creates a quick summary of the model

```r
broom::glance(mod1)
```

```
#>  # A tibble: 1 x 12
#>    r.squared adj.r.squared sigma statistic   p.value    df logLik   AIC   BIC
#>        <dbl>         <dbl> <dbl>     <dbl>     <dbl> <dbl>  <dbl> <dbl> <dbl>
#>  1     0.914         0.913  1.75     2459. 1.87e-125     1  -462.  931.  941.
#>  # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

Nice Addition

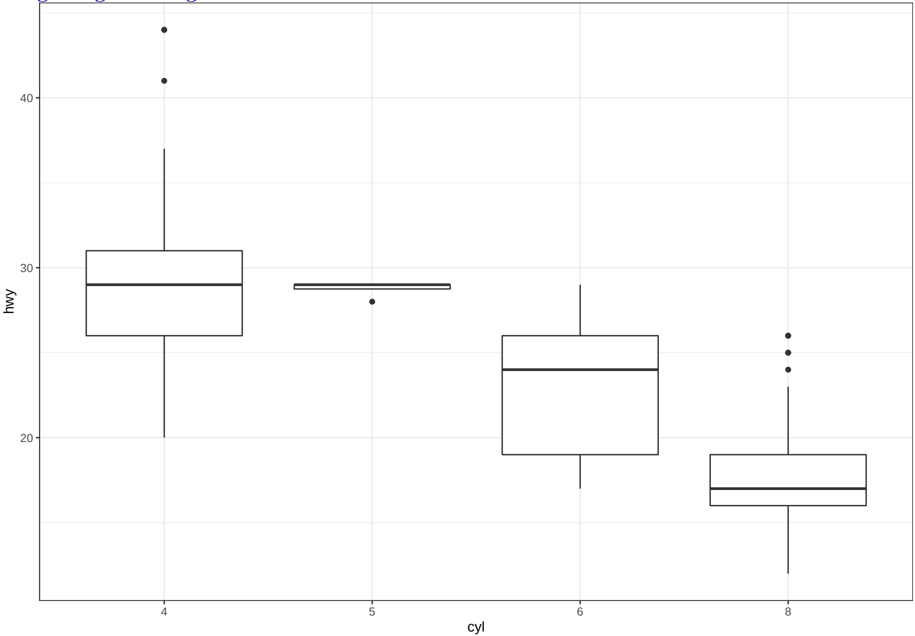
```
#>  `geom_smooth()` using formula 'y ~ x'
```



Highlighting regions

```r
mpg %>%
  mutate(cyl = as.factor(cyl)) %>%
  ggplot(aes(x = cyl, y = hwy)
         ) +
  geom_boxplot() +
  theme_bw()
```

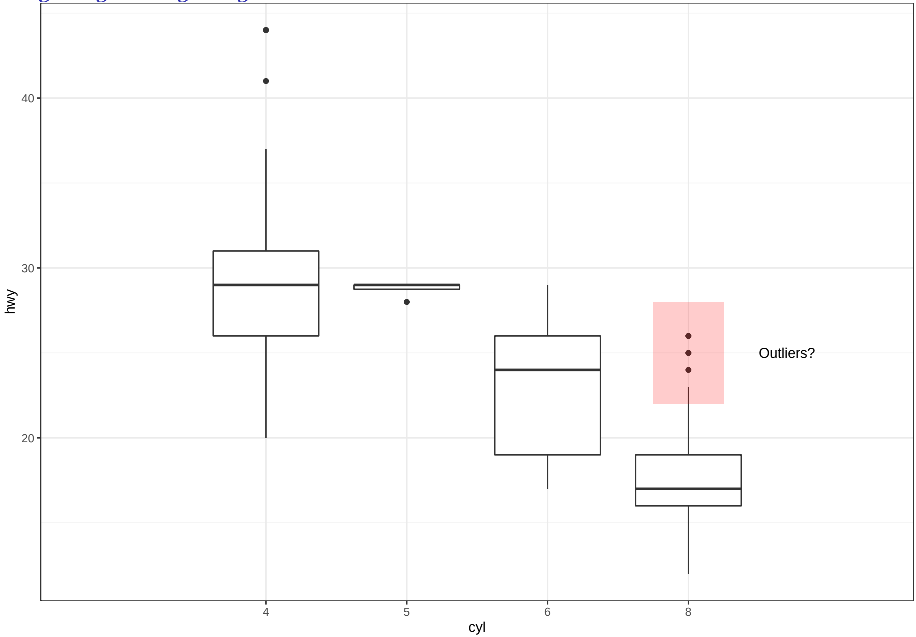
Highlight regions



Highlighting regions

```
mpg %>%
  mutate(cyl = as.factor(cyl)) %>%
  ggplot(aes(x = cyl, y = hwy)
         ) +
  geom_boxplot() +
  theme_bw()+
  annotate(geom = 'rect',
           xmin=3.75,xmax=4.25,
           ymin = 22, ymax = 28,
           fill = 'red',
           alpha = 0.2) +
  annotate('text',
           x = 4.5, y = 25,
           label = 'Outliers?',
           hjust = 0)+
  coord_cartesian(xlim = c(0,5))+
  theme_bw()
```

Highlighting regions


hwy
cyl
Outliers?