Visualization with R's tidyverse

Wilfried Cools & Lara Stas

2023-12-13

square.research.vub.be

Compiled on R 4.3.1

! What-Why-Who

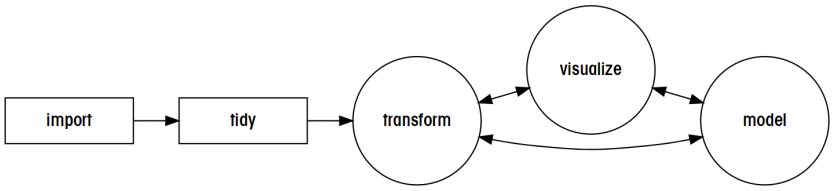
This site aims to introduce researchers to visualization in R with the `ggplot2` package of the `tidyverse` ecosystem.
Our target audience is primarily the research community at VUB / UZ Brussel, those who have some basic experience in R and want to know more.
We invite you to help improve this document by sending us feedback: wilfried.cools@vub.be

Key Message

- Data visualization can focus on data and/or its summarizing statistics
  - to convince yourself, understand data and statistical results
  - to convince others, show results to support inference
- Data visualization is inherent to data analysis
  - more than simply communicating the results
  - no -fit's all data visualization-
  - flexible use of data visualization
    - supports more informative and complete visualizations
    - elicits better data exploration and modeling
- Data visualization is best done with coding
  - maintain structure and transparency, to support reproducibility
- Data visualization is easier and more intuitive with tidy data
  - tidy data: meaning appropriately mapped into structure

        * each row an observation as research unit,
        * each column a variable as property,
        * each cell a particular value, linking row to column
        * note: data can be split into multiple tables (relational data).
    - aim for tidy data registration (avoid tedious manipulations)

- Workflow (Hadley Wickham):



R's tidyverse package: ggplot2

- Current focus on `ggplot2` the visualization package in the `tidyverse` eco-system (Hadley Wickham etal.)
- Different visualization packages exist in R.
    - base R
    - grid
    - trellis/lattice graphics
    - `ggplot2` (tidyverse) & `ggvis`
- `ggplot2` the current default
    - build on idea of Grammar of Graphics (Leland Wilkinson)
        * largely consistent
        * well appreciated defaults
        * easy and intuitive to build (if you get it)
        * without loosing much flexibility
    - explicitly links to tidy data
    - note: requires extensions for 3D plotting and interactive graphics
    - note: does not allow for multiple Y-axes (combination of axes)
- `ggplot2` part of the `tidyverse` ecosystem includes:
    - `dplyr` for manipulating data frames [check Data Manipulation]

  - `tidyr` for tidying data [check Data Representation]
  - `stringr` for dealing with texts
  - `forcats` for dealing with factors
  - …

- Convenient cheat sheets at https://rstudio.com/resources/cheatsheets/.

A quick illustration with ggplot2

toy dataset

- The infamous `iris` data are used
  - observe it's structure with `str( )` and first 6 observations `head( )` function.
  - note: available data with `data( )`

- Have a `tidyverse` look at the data with `glimpse( )`

```
glimpse(iris)
```

```
Rows: 150
Columns: 5
$ Sepal.Length <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0, 4.4, 4.9, 5.4, 4.~
$ Sepal.Width  <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9, 3.1, 3.7, 3.~
$ Petal.Length <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4, 1.5, 1.5, 1.~
$ Petal.Width  <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2, 0.~
$ Species      <fct> setosa, setosa, setosa, setosa, setosa, setosa, setosa, s~
```

- Have a `tidyverse` look at the data with `slice_head( )`

```
iris %>% slice_head(n=6)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

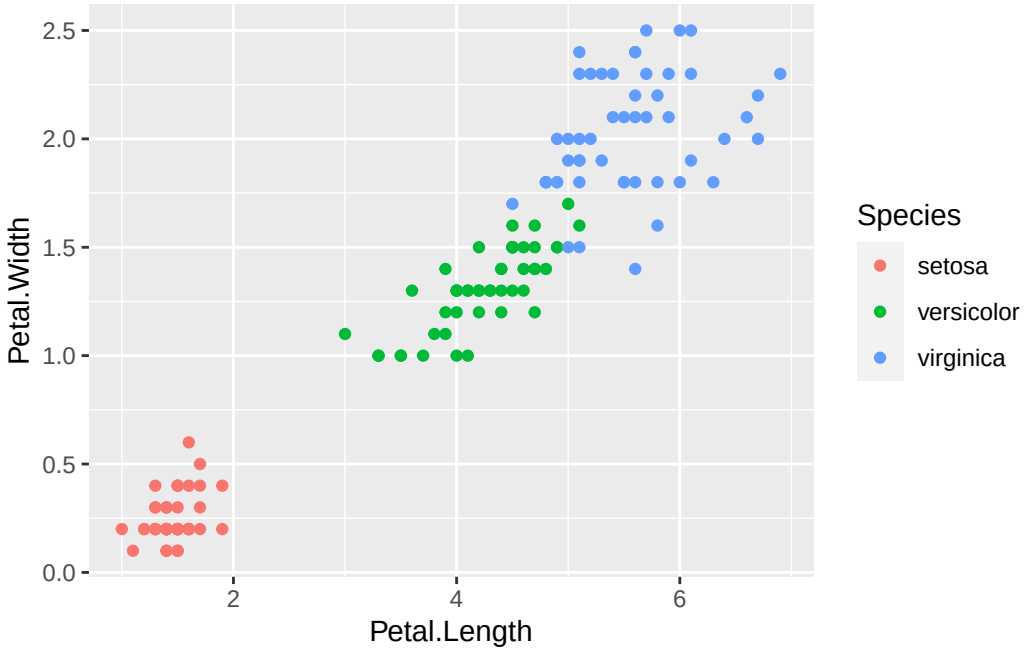
toy illustration

- Make a scatterplot, boxplot, and histogram

```
p1 <- ggplot(data=iris,
    aes(y=Petal.Width,
        x=Petal.Length,
        col=Species)) +
    geom_point()
```

- With the iris data
  - link dimensions y anx x to `Petal` columns
  - link color to column `Species`

```
p1
```

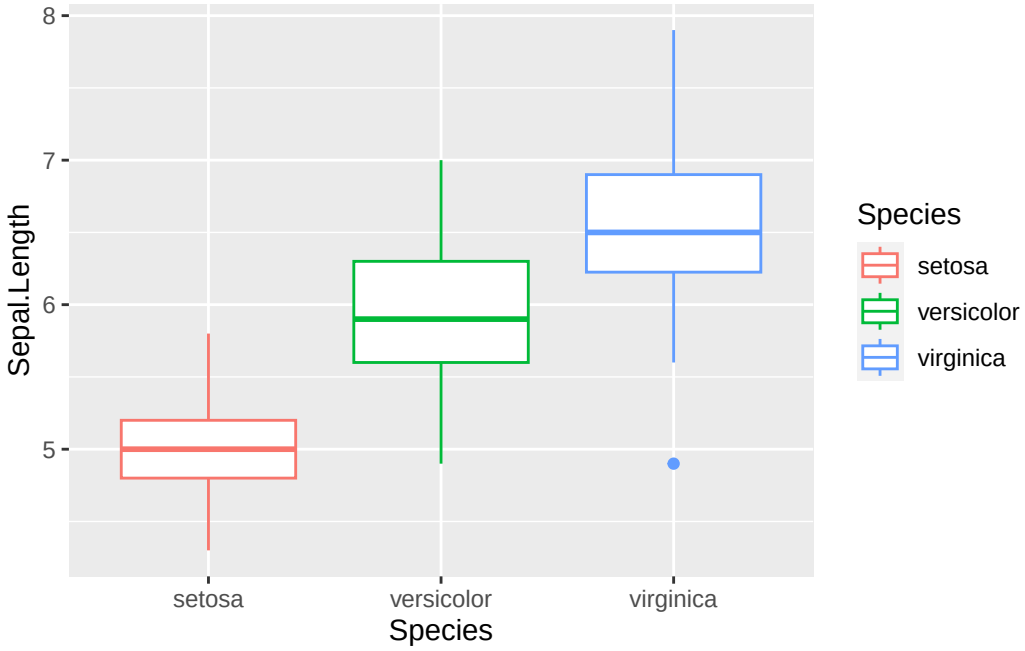



```
p2 <- ggplot(data=iris,
    aes(y=Sepal.Length,
        x=Species,
        col=Species)
```

```
    ) +
  geom_boxplot()
```

- x-axis is now `Species`
- Boxplots show a distribution of values for each `Species`

```
p2
```

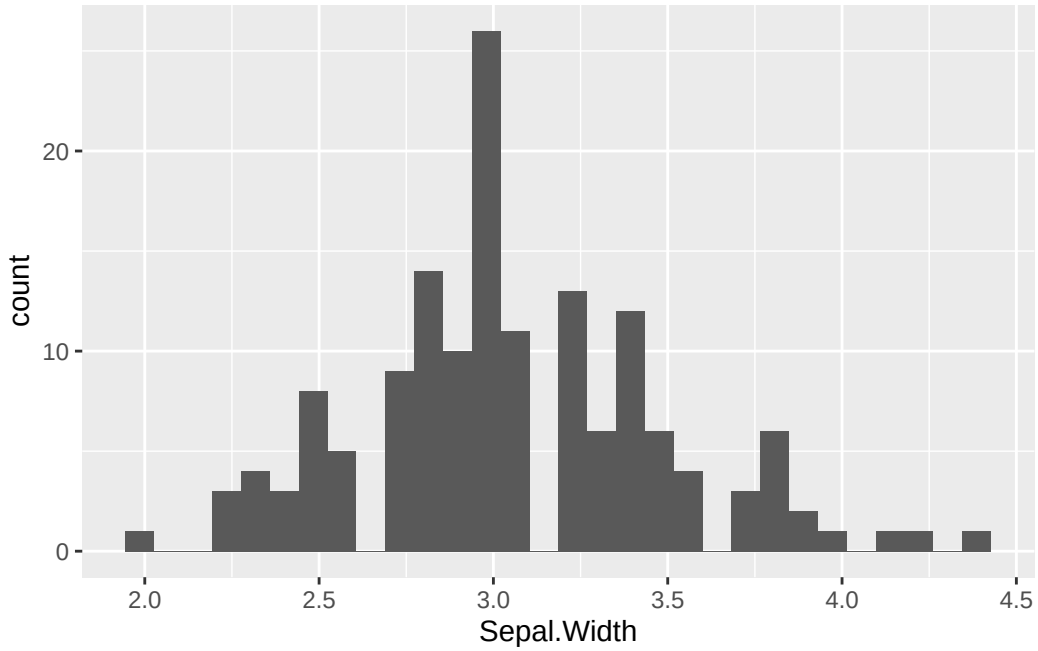



```
p3 <- ggplot(data=iris,
    aes(x=Sepal.Width)) +
    geom_histogram()
```

- x-axis is now Sepal.Width
- A histogram shows the full distribution
- A warning highlights default use of `bin`
  - Check `bins=10`

```
p3
```

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



- `ggsave( )` saves the last made plot

```
ggsave('plotname.png',width=12,height=6)
```

Layered building blocks

- `ggplot` philosophy: -gg- Grammar of Graphics (Leland Wilkinson)
  - build visualization like making a sentence
    - specify building blocks independently
    - combine blocks to create graphical display
  - example of layered building blocks:
    - `ggplot( data=iris, aes(y=Petal.Width,x=Petal.Length,col=Species) ) + geom_point() + geom_smooth()`
- `ggplot( )` creates a `ggplot` object
  - data argument to assign the data frame (or tibble)

  - `aes( )` function to assign variables from the data frame to scales (x-axis, y-axis, color)

- `geom_point( )` to add a scatterplot
  - note: scatterplots require an x and y-axis
  - note: x and y-axis are linked to variables in the data

- `geom_smooth( )` to add smooted averages
  - note: smoothed averages require an x and y-axis
  - note: x and y-axis are linked to variables in the data

- General structure includes functions and arguments
  - functions
    * `ggplot( )` initialize the ggplot object
    * `geom_*( )` visualize geometric objects
    * `stat_*( )` visualize statistical objects
      · largely equivalent to `geom`
    * `facet_*( )` conditional visualization
    * `theme( )`, `guides( )`, `scale_*( )`, `coord_*( )`
  - arguments in `ggplot( )`, `geom_*( )` and `stat_*( )`
    * `data` specify data (=input)
    * `aes( )` specify aesthetic mapping
      · bridging gap input and output
    * `...`

- Grammar of Graphics sparked further developments
  - other packages with -gg- philosophy: `ggforce`, `ggalt`, `ggpubr`, `ggraph`, `tidygraph`, `GGally`, `ggcorrplot`, `ggridges`, ...

Dimensionality

- Think of variables to show as dimensions
  - combine those variables that show your story

- `aes( )` links a variable to a dimension
  - x: categorical or continuous x-axis
  - y: categorical or continuous y-axis
  - color:
    * categorical: set of colors

        * continuous: shades of colors within range
    - shape: categorical (limited number)
    - linewidth: categorical or continuous (dedicated to lines)
    - linetype: categorical (dedicated to lines)
    - …
- Combinations of variables can be linked to one dimension
- Facets: categorical (panels)
    - a panel for each (combination of) value(s)
- Rethink your visualization if this is not sufficient
    - use dimensions aimed at bringing focus

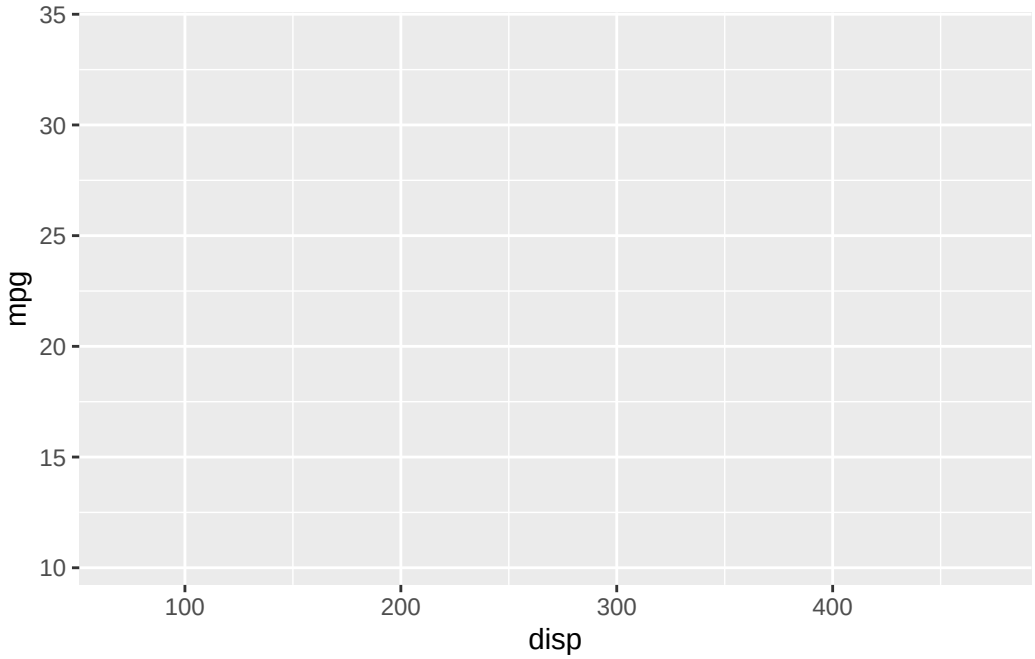
Visualization essentials

- part 1: how to make a visualization
- part 2: how to further refine

step by step example

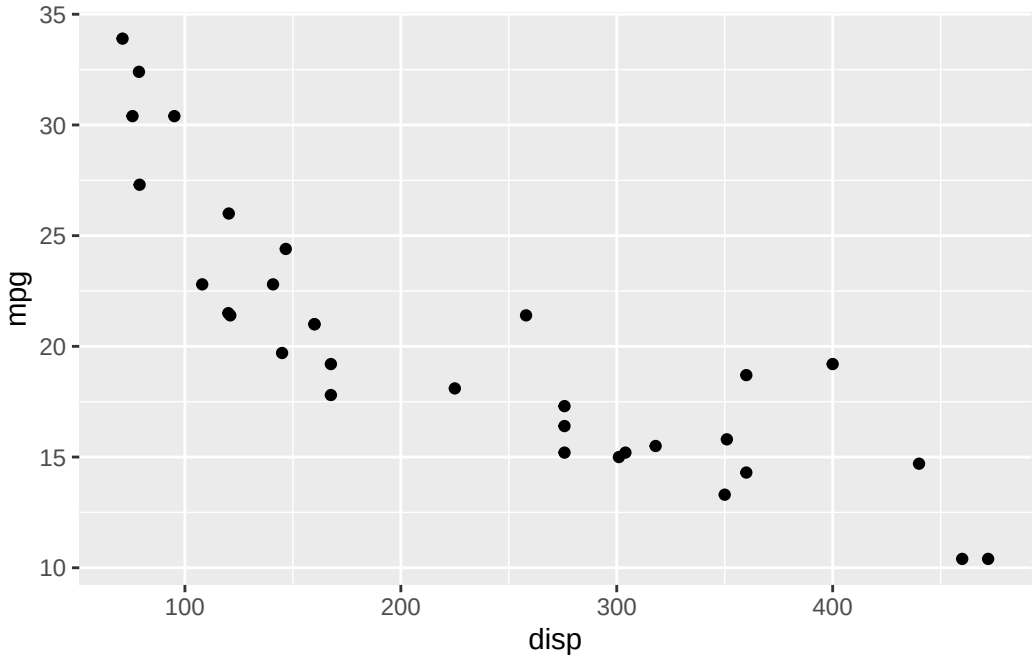
- The `ggplot` object is constructed
    - data is linked to the `mtcars` data
    - x and y aesthetic are linked to it's variables `mpg` and `disp`
    - does not visualize !
- The internal representation does exist
    - ready to extend for visualization
    - aesthetics x and y are given their default values
        * `mpg` and `disp` from `mtcars`
    - includes a legend and scale for both x and y axis
- Any geometric function (object) that at least uses an x and y axis can be added as a layer

```
ggplot(
    data=mtcars,
    aes(y=mpg,x=disp)
)
```



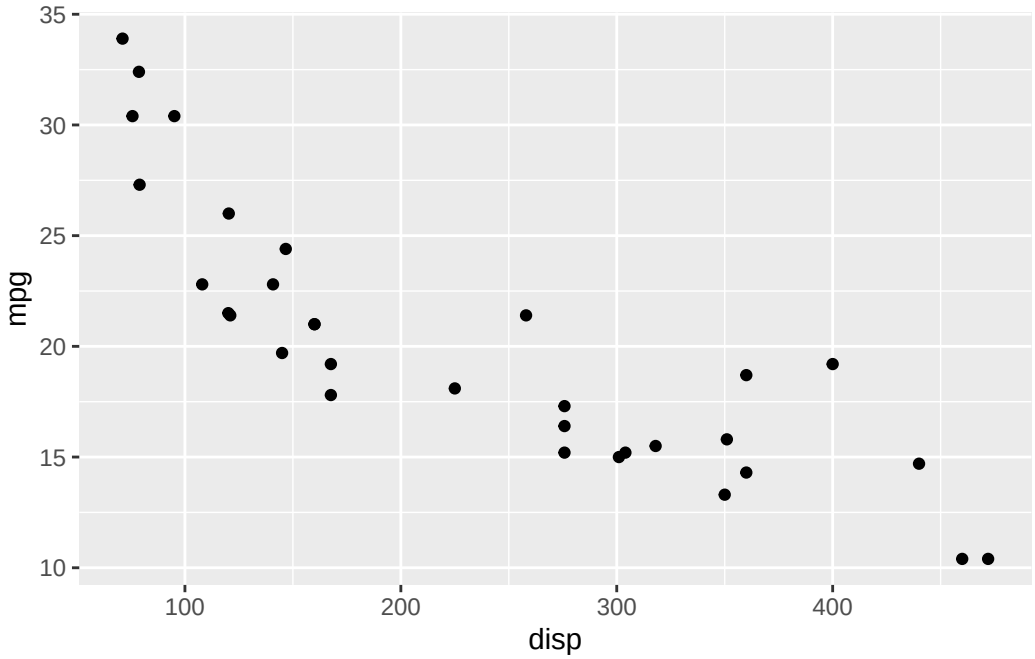
- Add a layer with the `+` sign
  - a layer often is geometric function
  - a geometric function visualizes a `gglot` object
- `geom_point( )`: create a scatterplot
  - geometric function without arguments

```
ggplot(data=mtcars,
    aes(y=mpg,x=disp)
) +
geom_point()
```



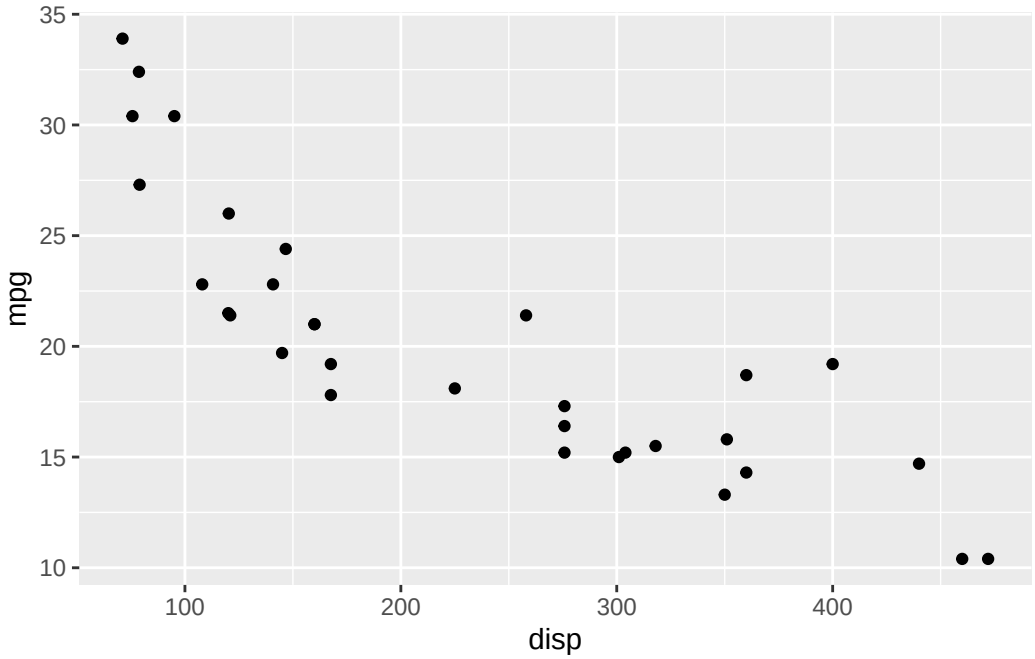
- `geom_point()` requires data, an x and a y-axis
  - if not specified, inherit from `ggplot( )`
  - if specified, ignore `ggplot( )`
- example
  - data is not specified inside, but inherited from `ggplot( )`
    - data is linked to `mtcars`
  - x is not specified inside, but inherited from `ggplot( )`
    - x linked to variable `disp`
  - y is specified inside within `aes( )`
    - y linked to variable `mpg`
- get help using `?`, `?geom_point`

```
ggplot(data=mtcars,
    aes(x=disp)
) +
geom_point(aes(y=mpg))
```



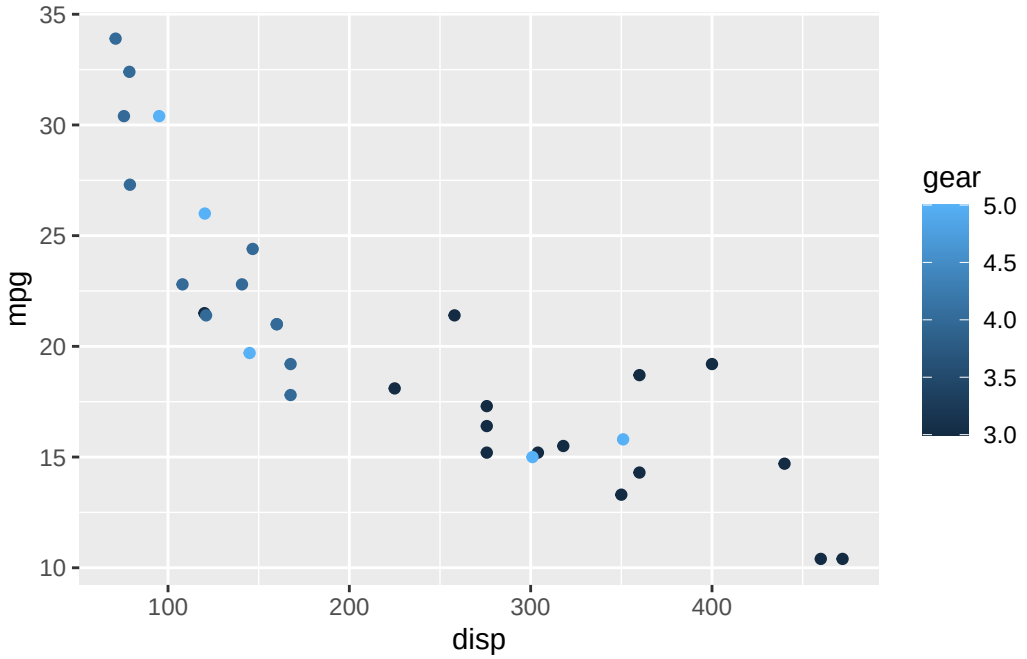
- For this example:
  - specify the dimensions in the constructor
  - add the data when creating the scatterplot

```
ggplot() + geom_point(data=mtcars,aes(x=disp,y=mpg))
```



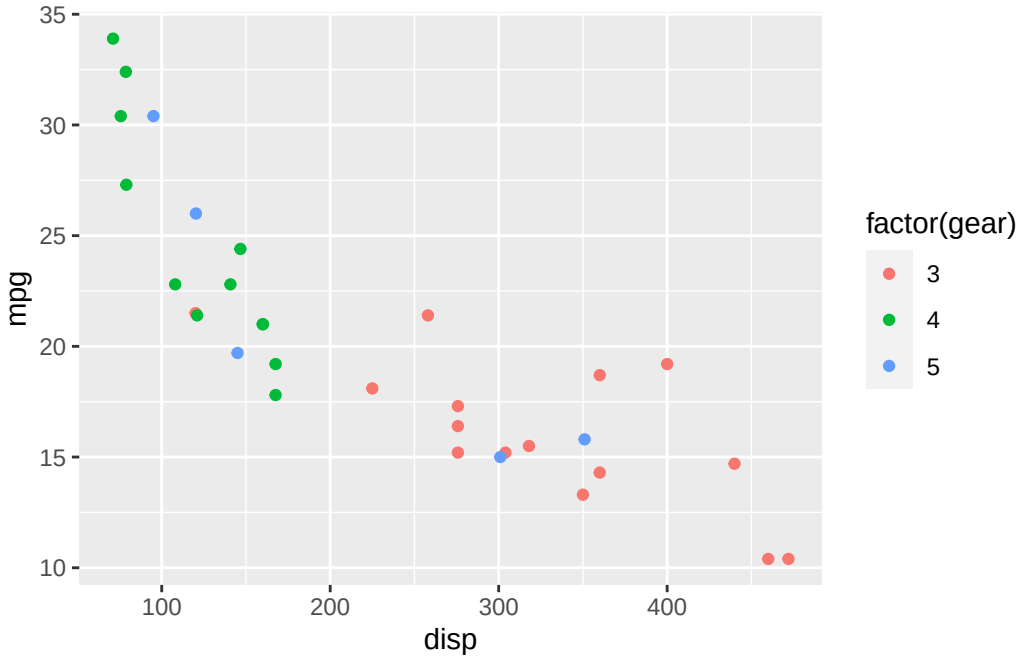
- Non-essential aesthetics like color can be included
  - specify color dependent on data within the `aes( )`
    - color extracted from variable `gear`
  - note: color is a third dimension
- Note: the numerical (continuous) variable gear is assigned a continuous scale of colors
  - a legend for continuous variables is by default included

```
ggplot(data=mtcars,
    aes(y=mpg,x=disp,
        color=gear)
) +
geom_point()
```



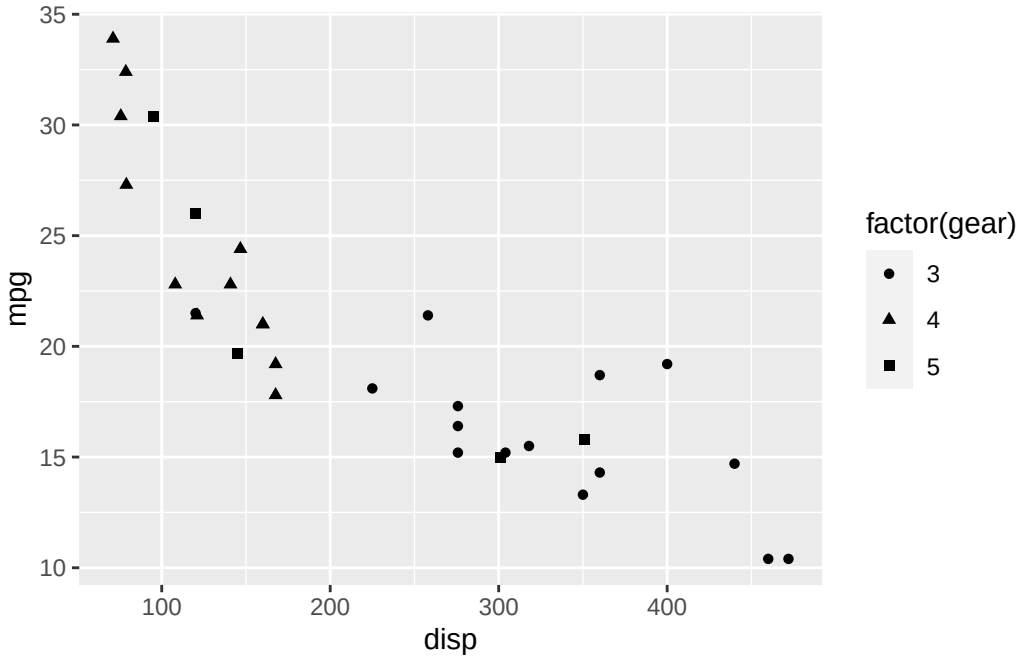
- The type of variable determines how it is visualized
- A categorical version of the same variable is visualized differently
  - the numerical variable is turned into a categorical one (`factor()`)
  - a set of colors is used
  - a categorical legend is included

```
ggplot(data=mtcars,
    aes(y=mpg,x=disp,
    color=factor(gear))
) + geom_point()
```



- Instead of color, use shape
  - not every dimension is equally clear
  - shapes require categorical data

```
ggplot(data=mtcars,
    aes(y=mpg,x=disp,
        shape=factor(gear))
) +
geom_point()
```

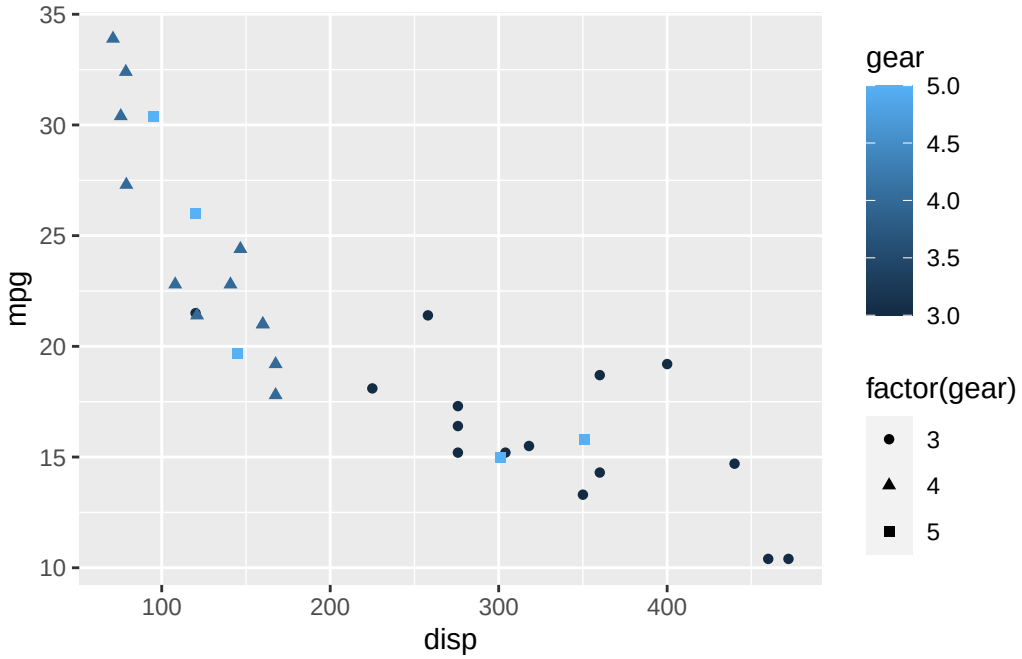


- Default behavior can be overwritten
  - the categorical color variable (geometric function) is used
  - the numerical color variable (constructor) is not considered anymore

```
ggplot(data=mtcars,
    aes(y=mpg,x=disp,color=gear)) +
    geom_point(
        aes(color=factor(gear))
    )
```

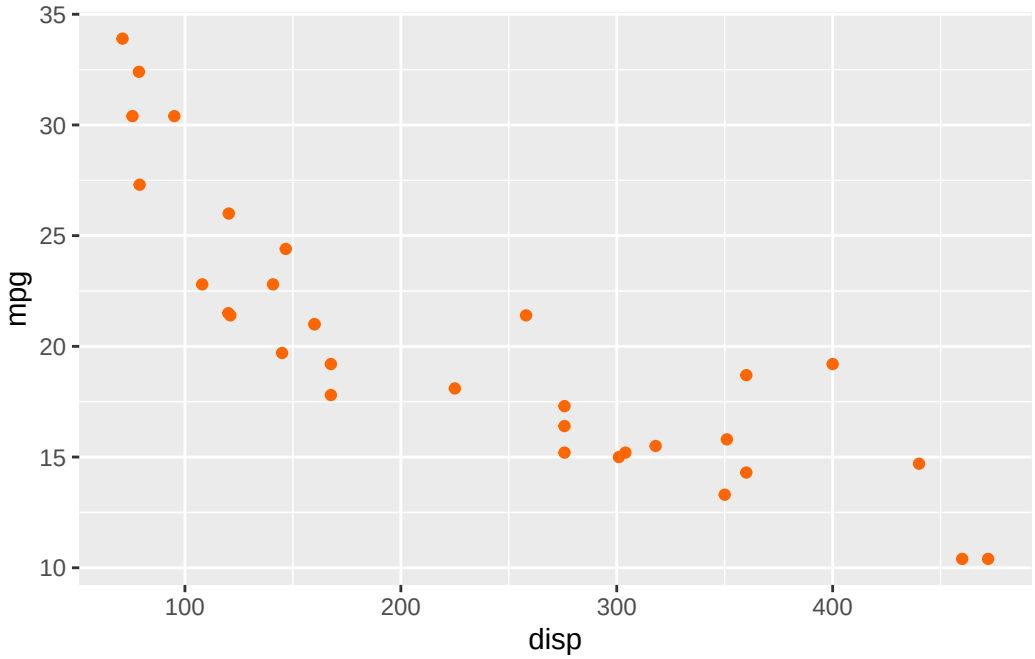
- Turn the color into shape, but keep the continuously colored scale as well

```
ggplot(data=mtcars,
    aes(y=mpg,x=disp,color=gear)) +
    geom_point(
        aes(shape=factor(gear))
    )
```



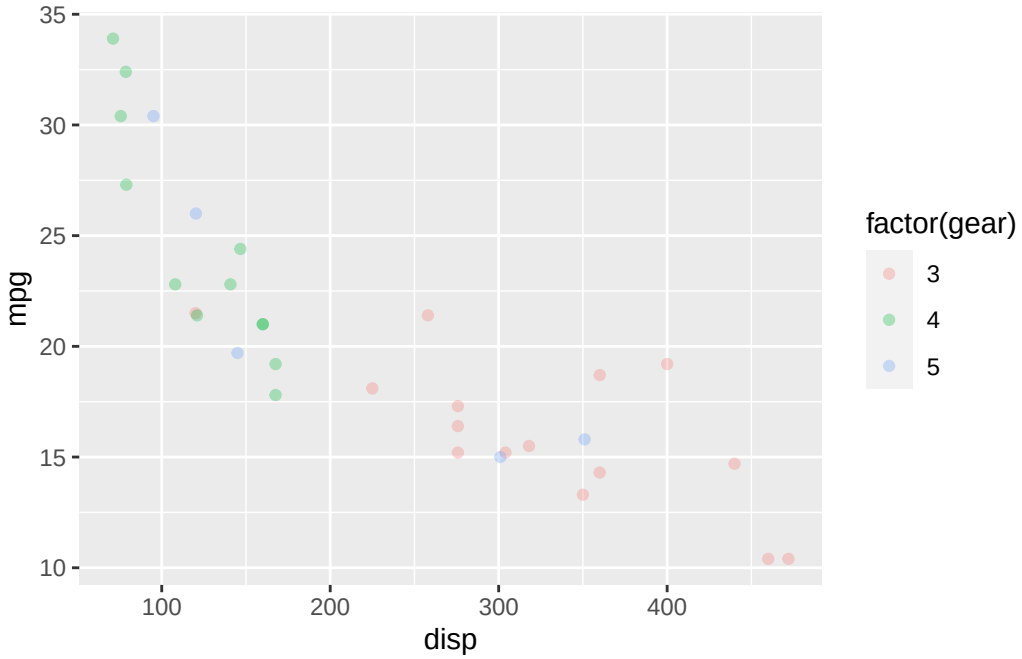
- Dimensions can also be linked to constants (not variables in the data)
- Assign a dimension outside the `aes( )` function
  - `aes( )` only serves to link dimensions to variables in the data
  - color is assigned a value, independent of the data
  - color from the `ggplot( )` is overwritten

```
ggplot(data=mtcars,
    aes(y=mpg,x=disp,color=gear)
) +
geom_point(color='#FF6600')
```



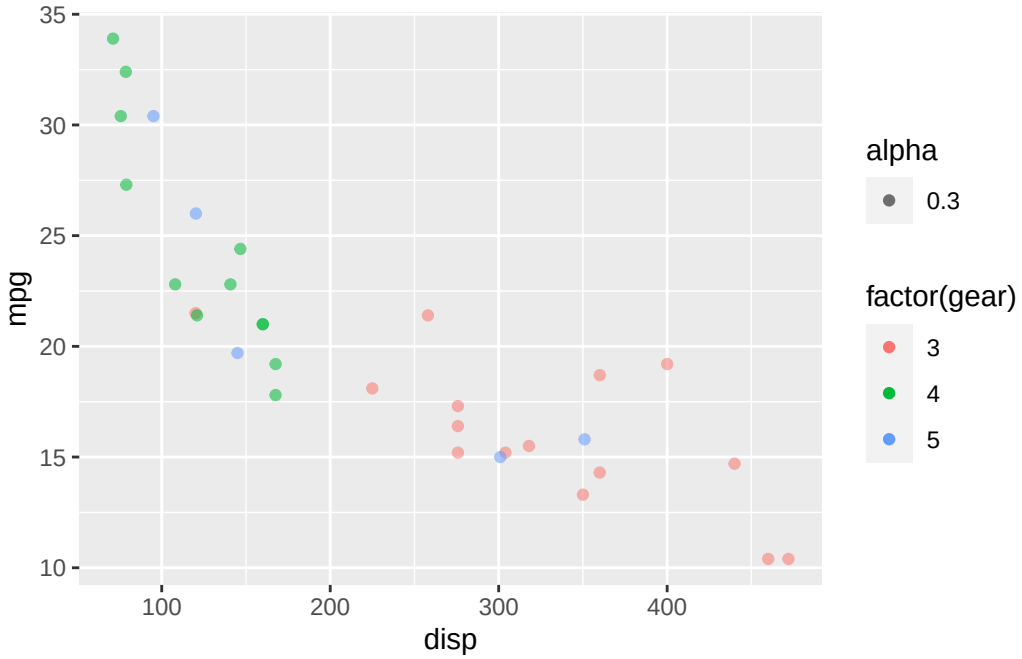
- Multiple aesthetics (dimensions) can be specified, in and outside the `aes( )`
- With color dependent on gear
  - `color` dependent on a variable is defined inside of `aes( )`
  - `alpha` (transparency) is set to .3 (30%), outside of `aes( )`
    - alpha as a percentage (avoid this, it gives unwanted behavior)
    - alpha related to variables with values between 0 and 1

```
ggplot(data=mtcars,
    aes(y=mpg,x=disp)
) +
geom_point(
    aes(color=factor(gear))
    ,alpha=.3)
```



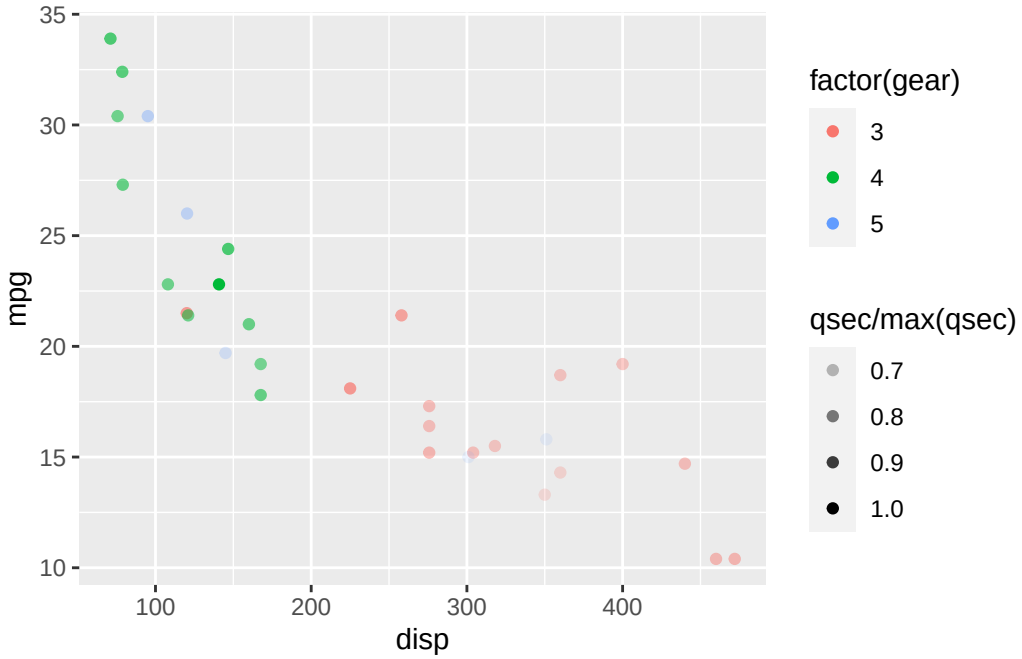
- Note: with alpha (transparency) inside of `aes( )` at .3 (30%)
  - `aes( )` should not be used to link to constants

```
ggplot(data=mtcars,
    aes(y=mpg,x=disp)
) +
geom_point(
    aes(color=factor(gear),alpha=.3)
    )
```



- Use alpha inside of `aes( )` to link it to a variable
  - alpha related to variables with values between 0 and 1

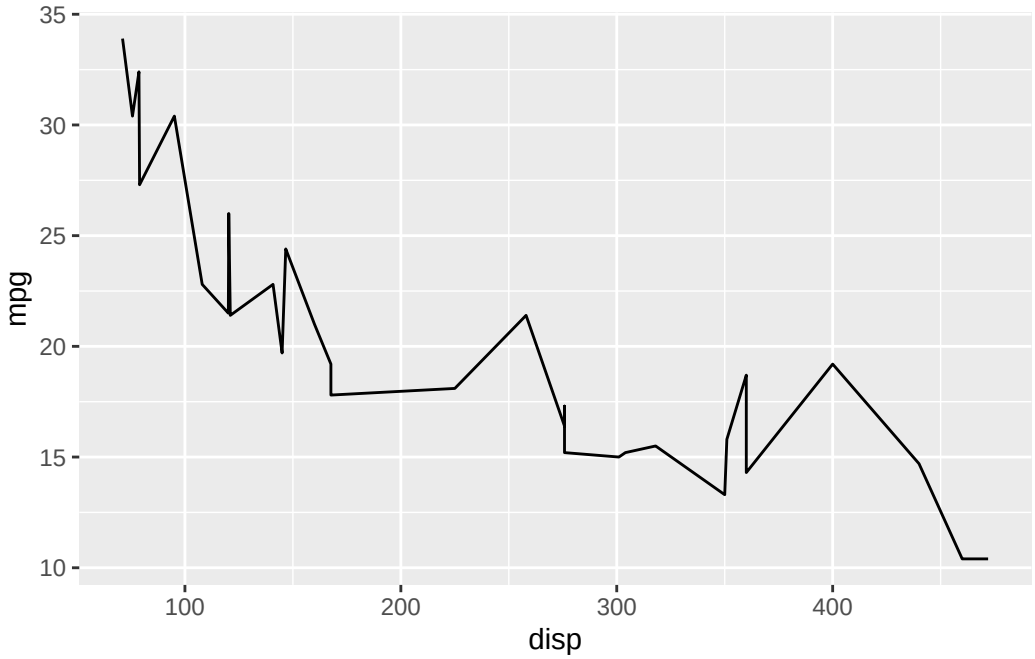
```
ggplot(data=mtcars,
    aes(y=mpg,x=disp)
) +
geom_point(
    aes(color=factor(gear),
        alpha=qsec/max(qsec)
    )
)
```



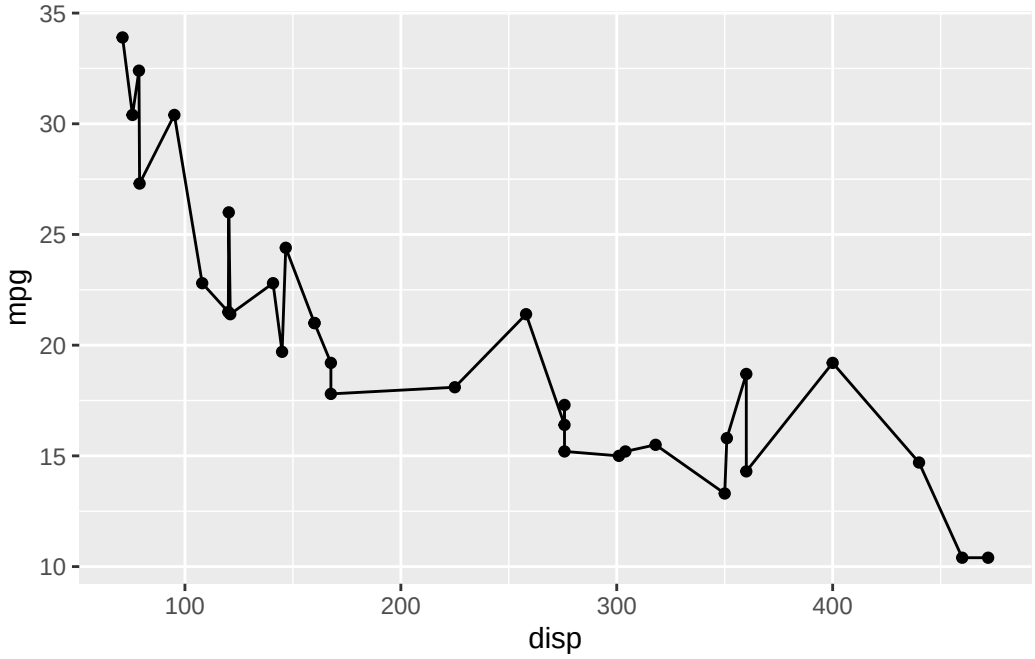
- Multiple geometric functions can be included, layered
  - `geom_point( )` creates the dots, a scatterplot
  - `geom_line( )` connects them over the x-axis
- Note: using the assignment `<-` code can be build stepwise

```
myplot <- ggplot(data=mtcars,
    aes(y=mpg,x=disp)) +
    geom_line()
myplot <- myplot + geom_line()
myplot
```

```
ggplot(data=mtcars,
    aes(y=mpg,x=disp)) +
    geom_line()
```

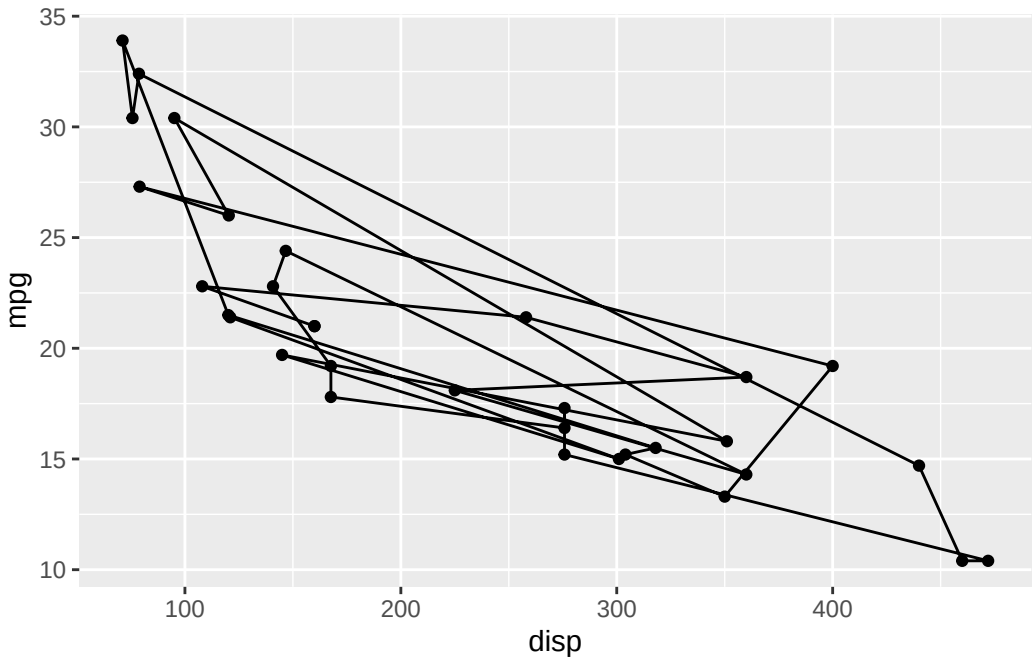



```
ggplot(data=mtcars, aes(y=mpg,x=disp)) +
    geom_point() + geom_line()
```

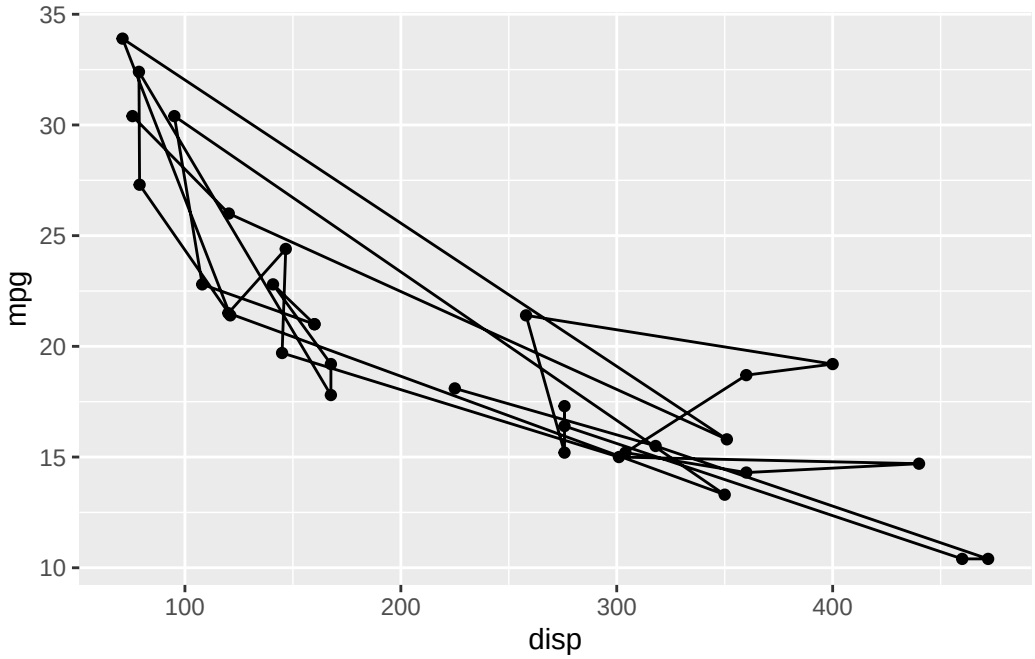


- Use `geom_path( )` to connect subsequent observations
- Note: re-ordering has an effect (`arrange( )`)

```
ggplot(data=mtcars,
    aes(y=mpg,x=disp)
) +
geom_point() + geom_path()
```

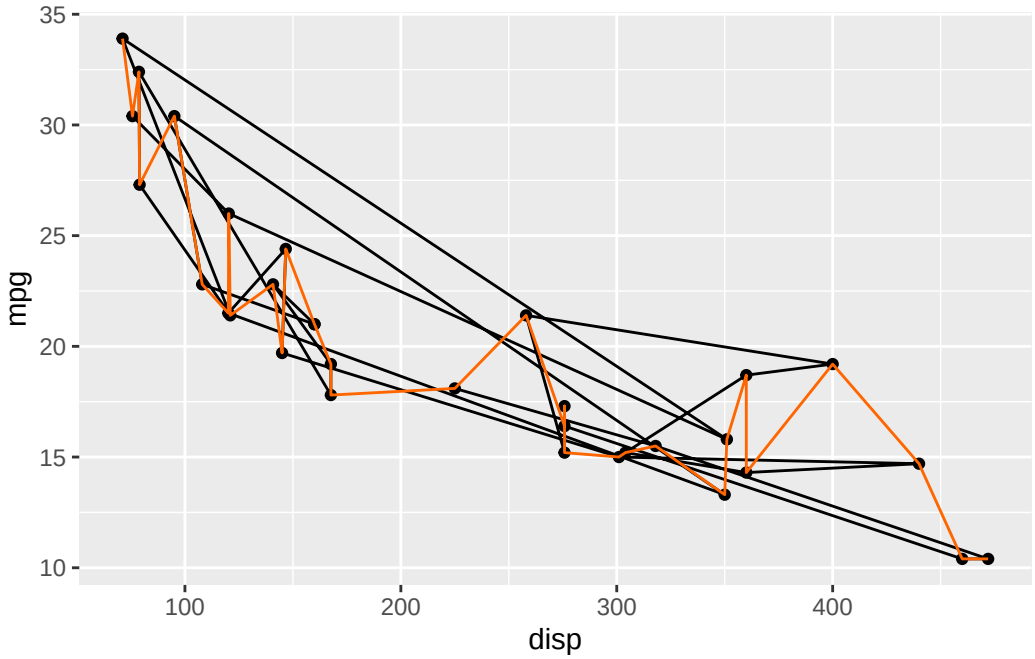



```
ggplot(data=mtcars %>% arrange(drat),
    aes(y=mpg,x=disp)
) +
geom_point() + geom_path()
```



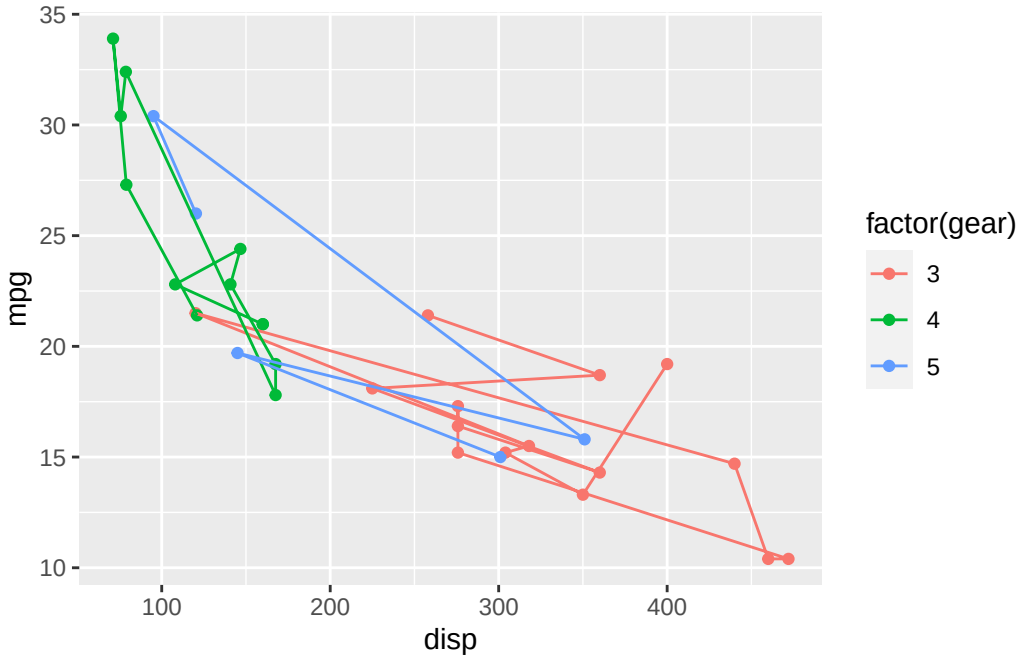
- Add a line over the x-axis, on top of the path
- Make the line orange (#FF6600) to highlight it compared to the path

```
ggplot(data=mtcars %>% arrange(drat),
    aes(y=mpg,x=disp)
) +
geom_point() + geom_path() + geom_line(color="#FF6600")
```



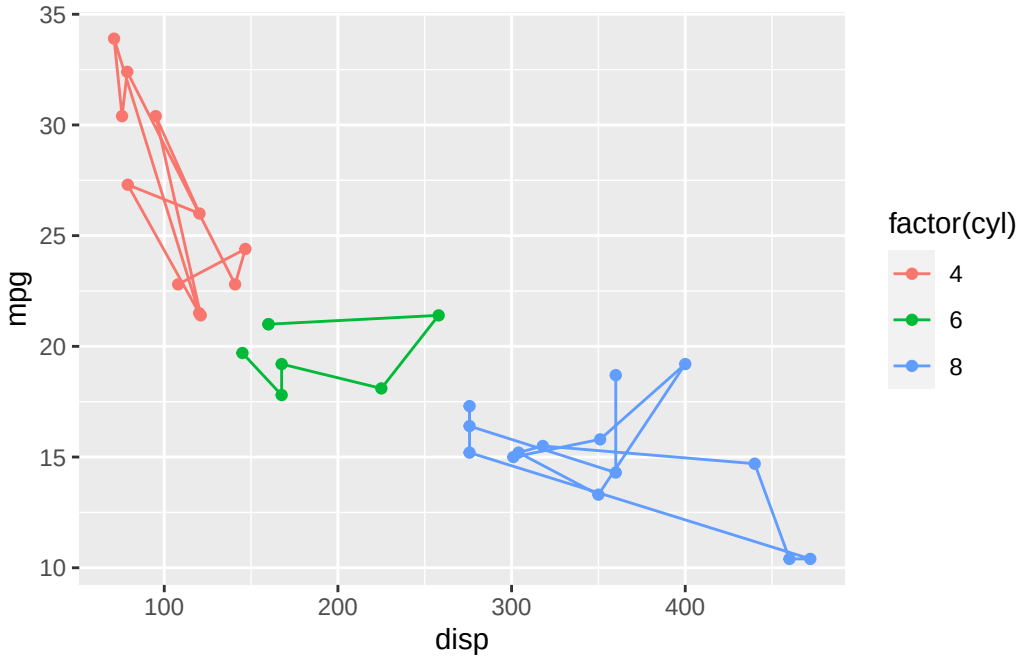
- The type of variables in `aes( )` determines how it is visualized
- Note, assigning a categorical variable to color groups data (lines/path)

```
ggplot(data=mtcars,
    aes(y=mpg,x=disp,
    color=factor(gear))
) + geom_point() + geom_path()
```



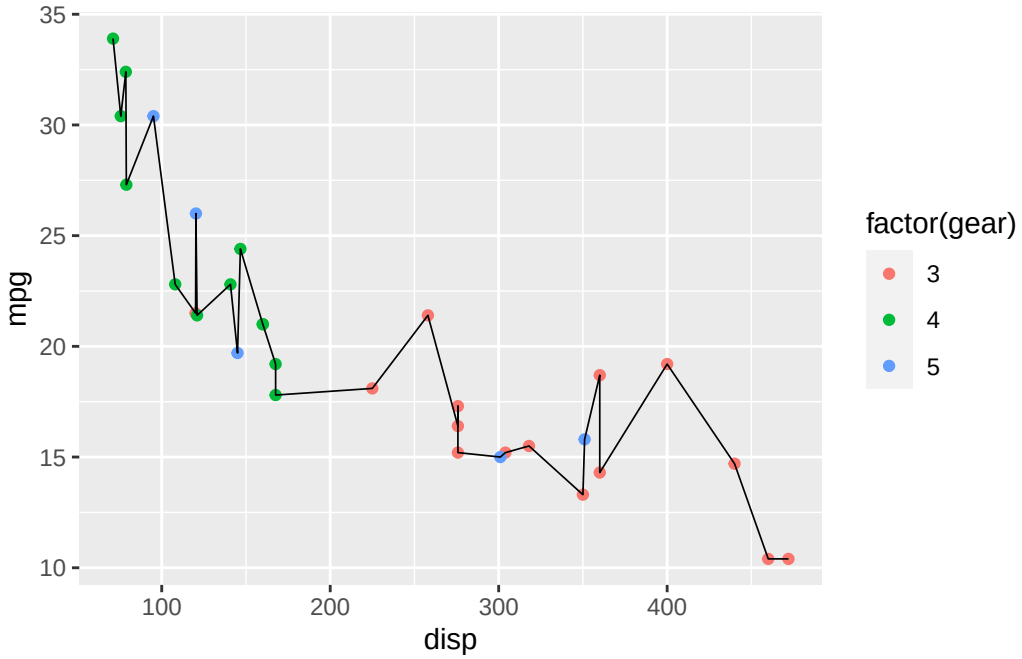
- Note, changing the variable linked to color changes the grouping

```
ggplot(data=mtcars,
    aes(y=mpg,x=disp,
    color=factor(cyl))
) + geom_point() + geom_path()
```



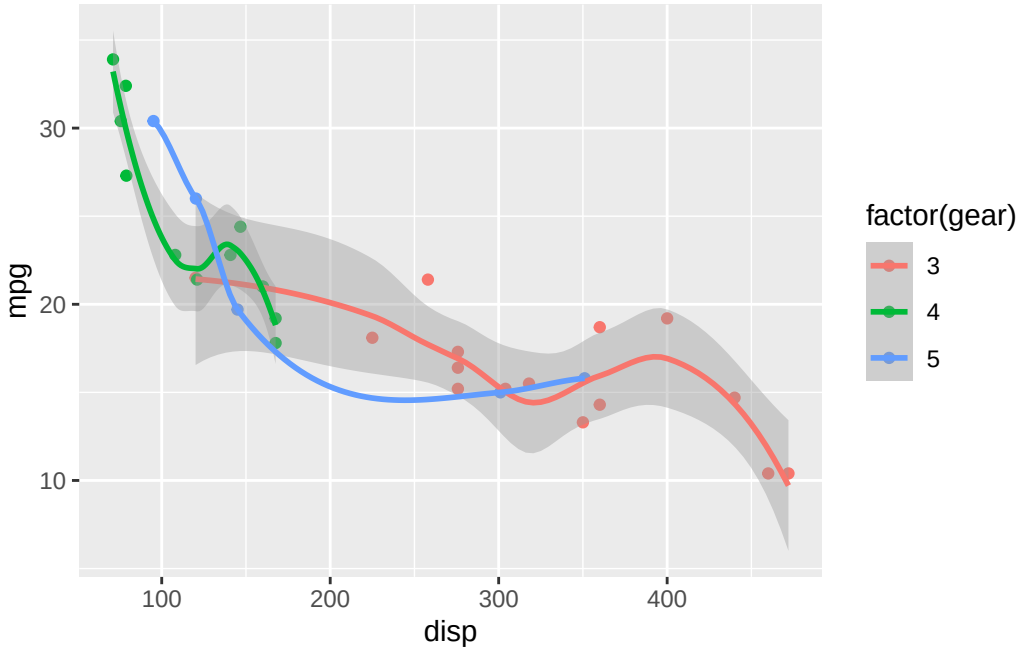
- Specifications in `ggplot( )` offer default behavior
- Specifications inside the `geom_*( )` overwrite defaults
  - note that the `geom_point( )` uses the locally specified color aesthetic
  - note that the `geom_line( )` uses the default black (size made smaller than default)
- Beware: types (discrete/continuous) should agree when overwriting `ggplot(aes( ))`

```
ggplot(data=mtcars,
    aes(y=mpg,x=disp)
) +
geom_point(
    aes(color=factor(gear))
) +
geom_line(linewidth=.3)
```

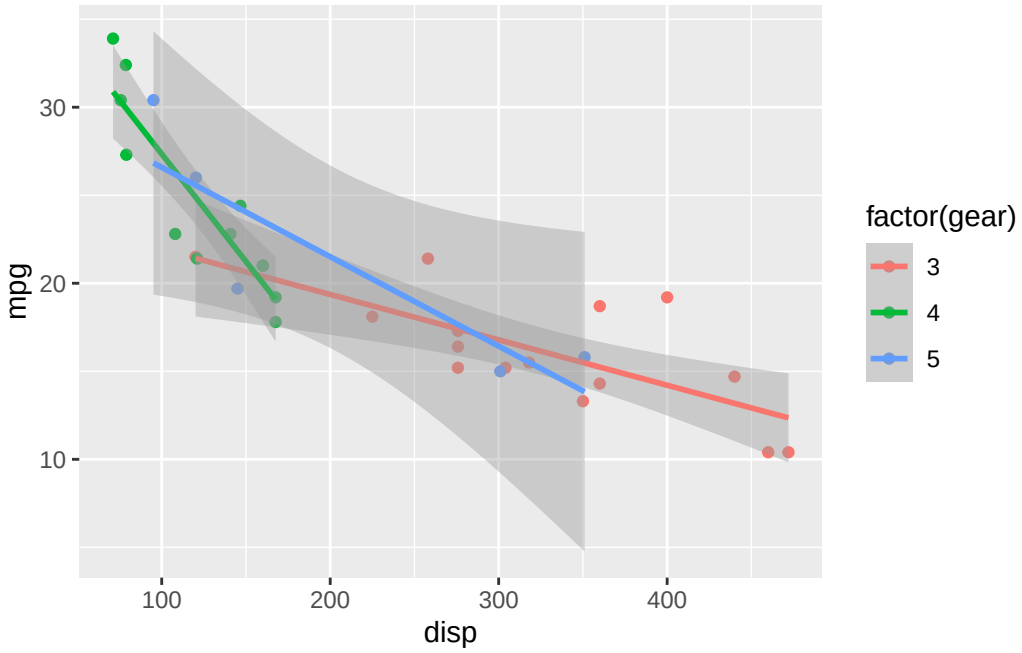


- Geometric functions that add statistics can be included as well
  - `geom_smooth( )` offers averaging and standard errors
    - local averages are default (loess lines)
    - global conditional averages (method='lm')

```
myplot <- ggplot(data=mtcars, aes(y=mpg,x=disp,color=factor(gear)))
myplot + geom_point() + geom_smooth()
```



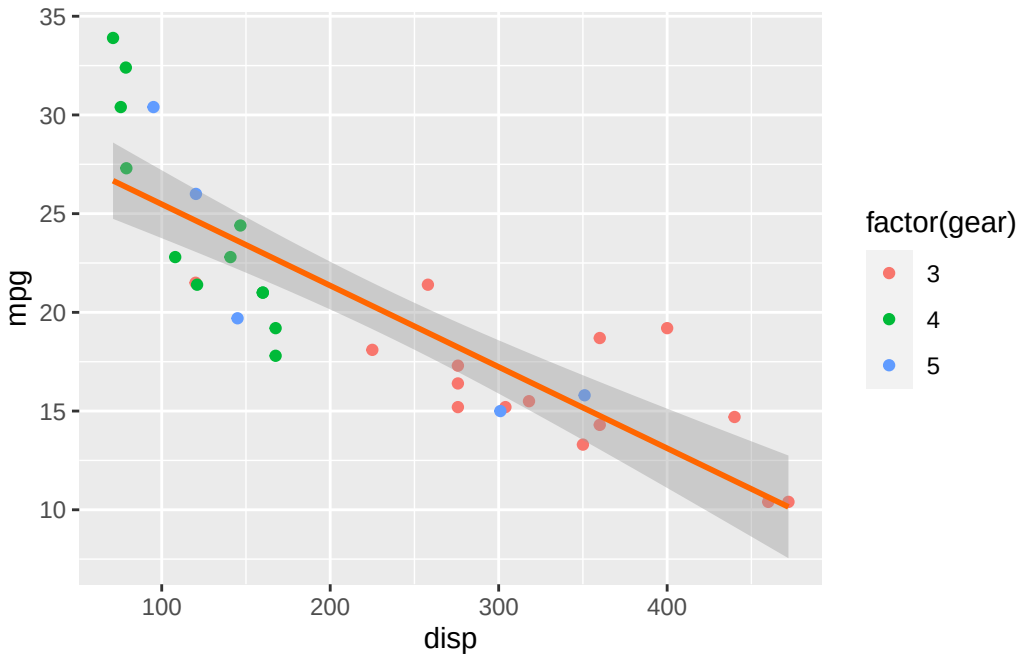

```
myplot + geom_point() + geom_smooth(method='lm')
```



- Grouping through aesthetics works like before
  - the `group` argument overwrites grouping by other aesthetics like color
  - group=1 groups all observations together
- Note: get help `?geom_smooth` for more details

```
ggplot(data=mtcars,
    aes(y=mpg,x=disp,
        color=factor(gear),
        group=1)
) + geom_point() +
geom_smooth(method='lm',color="#FF6600")
```

```
`geom_smooth()` using formula = 'y ~ x'
```



recap ggplot( )

- Function to create a `ggplot` object, ready for visualization.
  - constructor, always required
  - prepares appropriate internal representation

  - can include default data
  - can include default aesthetics

recap aes( )

- Function to link variables (data) to an aesthetic (dimension)
  - used within `ggplot( )` or `geom_*( )`
  - requires variables (part of data)
  - defines dimensions dependent on variable values
    - x and y axes: positions on the axes
    - color: color
    - …
  - possible arguments depent on `geom_*( )` if defined
    - shape: symbols in `geom_point( )`
    - size: size of bullets in `geom_point( )`
    - linewidth: width of lines in `geom_line( )`
    - … check the respective help-files
  - group argument used inside `aes( )` to identify groups of observations
    - allows grouping without assigning an aesthetic
    - use the value `1` to combine all observations into one group
  - automatically assigns a default legend
    - relates to aesthetic
    - depends on variable type (nominal, ordinal, continuous)
  - note: aesthetics defined outside of `aes( )` are independent of the data

recap geom_*( )

- Function to turn an internal `ggplot` representation into a visualization
  - different geoms to create different visualizations
  - different geoms use
    - required aesthetics
      - scatterplots require x and y axis
      - histogram requires x axis
    - optional aesthetics (eg., color)
  - geom-specific arguments overwrite those inherited from `ggplot( )`
    - can include `aes( )`
    - can include data argument
    - useful to add or change aesthetics

Visualization extras

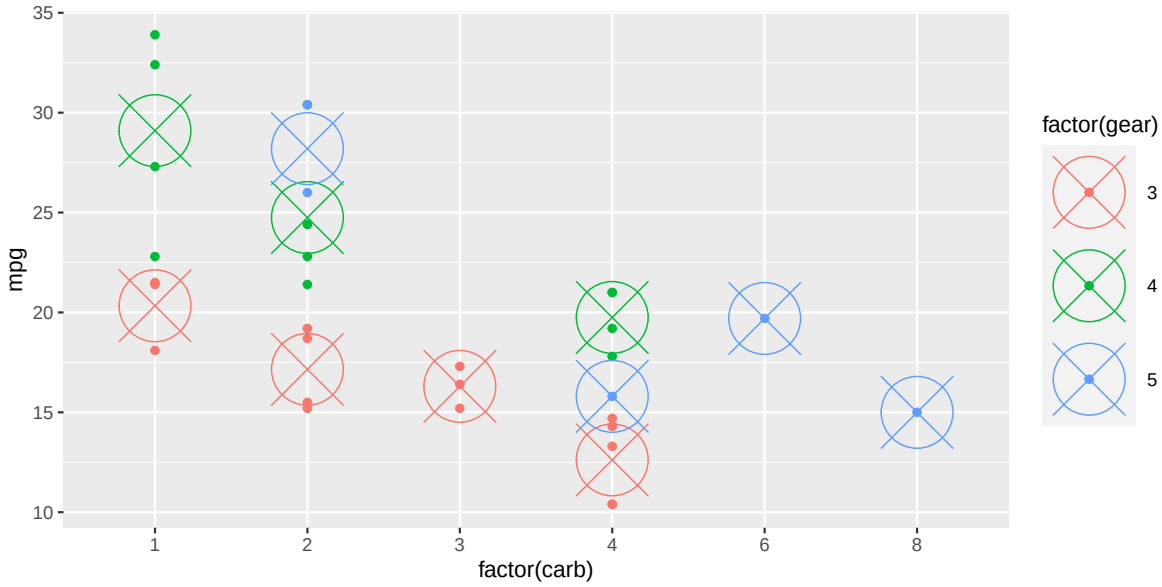
- In addition to the basics, there is much more
  - stat layers
  - scales layers
  - facet layer
  - theme layer
  - coord layer

geom and stat layers

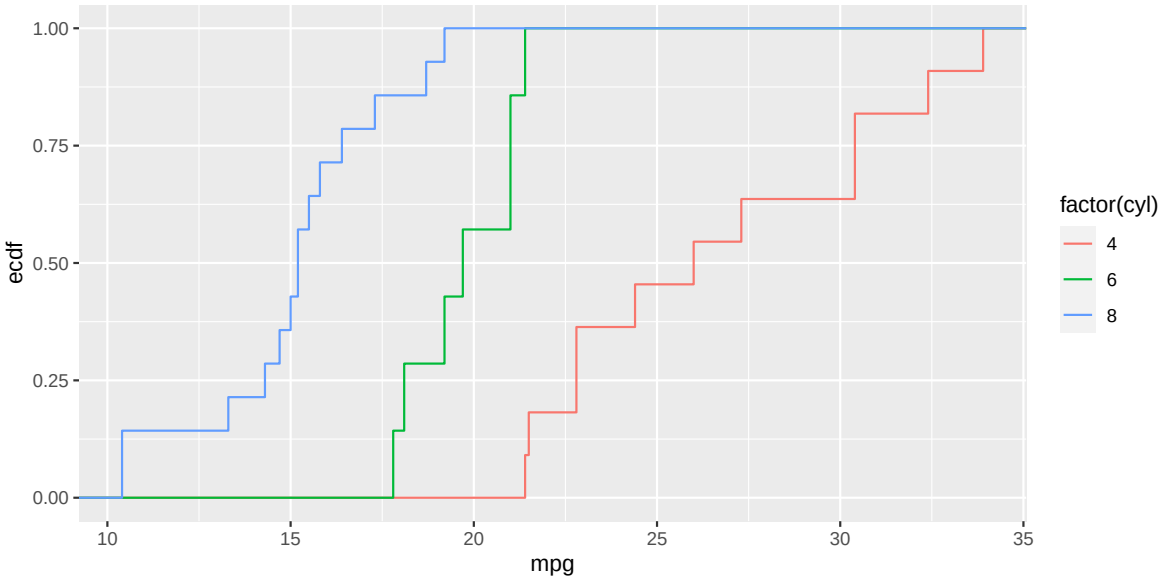
- Geometric functions and statistical transformation
  - each add layers
  - geom_* and stat_* are largely equivalent
- A `geom_*( )` has a default stat argument
  - `geom_smooth(stat="smooth")`
- A `stat_*( )` has a default geom argument
  - `stat_smooth(geom="smooth")`
- Same result
  - `+ geom_point(stat='summary',fun.y='mean',  shape=13,size=16)`
  - `+ stat_summary(geom='point',fun.y='mean',  shape=13,size=16)`
- In most cases stick to `geom_*( )`, occasionally not possible: eg., `stat_ecdf( )`
  - specific `stat_*()` function highly dedicated

```
ggplot(data=mtcars,
    aes(y=mpg,x=factor(carb),
    color=factor(gear))
) +
geom_point() +
stat_summary(geom='point',fun.y='mean',shape=13,size=16)
```

```
Warning: The `fun.y` argument of `stat_summary()` is deprecated as of ggplot2 3.3.0.
i Please use the `fun` argument instead.
```

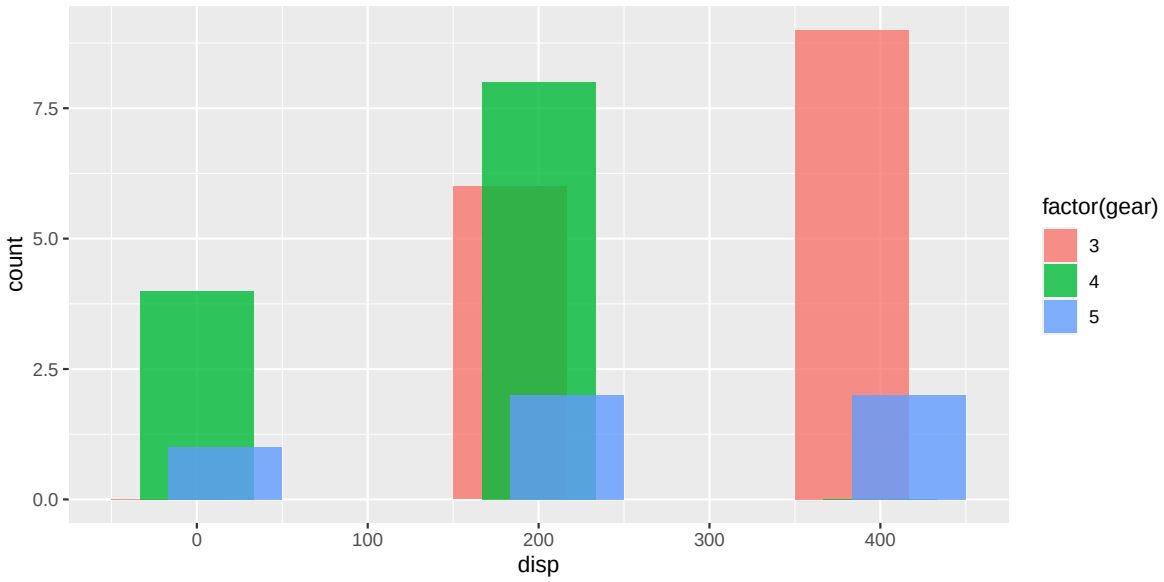



```
ggplot(mtcars,
    aes(mpg)
) + stat_ecdf(
    aes(color=factor(cyl)),
    geom = "step"
)
```



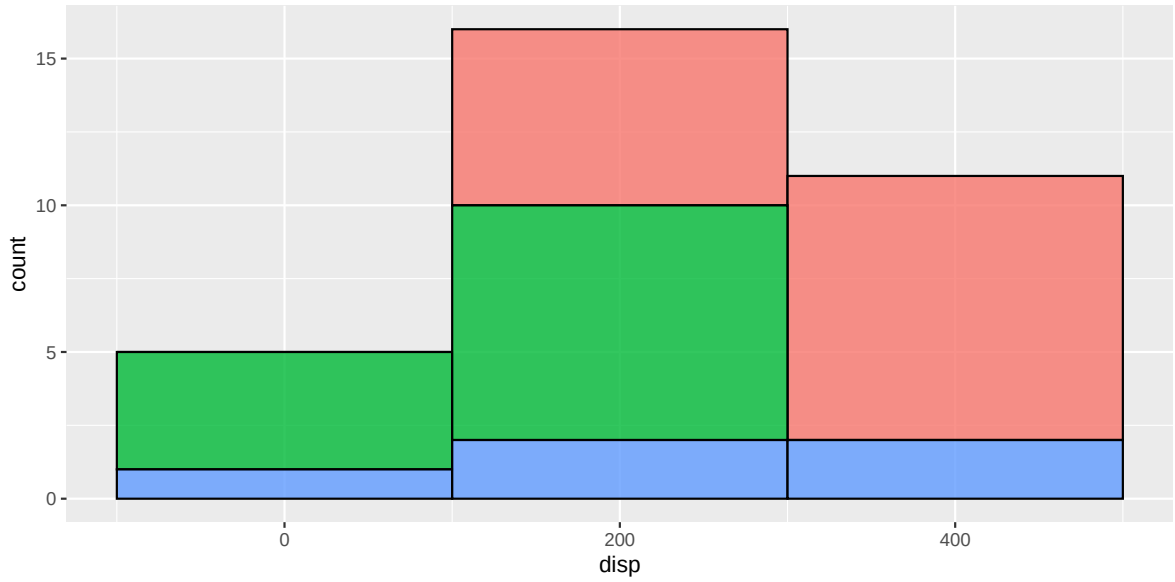
- A layer is a function with arguments
  - data (see before)
  - mapping (aesthetic), defined by the `aes( )` (see before)
  - geom (eg., point or smooth)
  - stat (eg., identity or smooth)
  - position (eg., identity)
- Position adjustment with position argument
  - identity, typically the default
  - jitter convenient for points and lines (random perturbation)
  - stack, fill and dodge convenient for bars (on top or next to)

```
myplot <- ggplot(data=mtcars,
    aes(x=disp,
    fill=factor(gear))
)
myplot + geom_histogram(binwidth=200,
    position = position_dodge(width=50),
    alpha=.8)
```

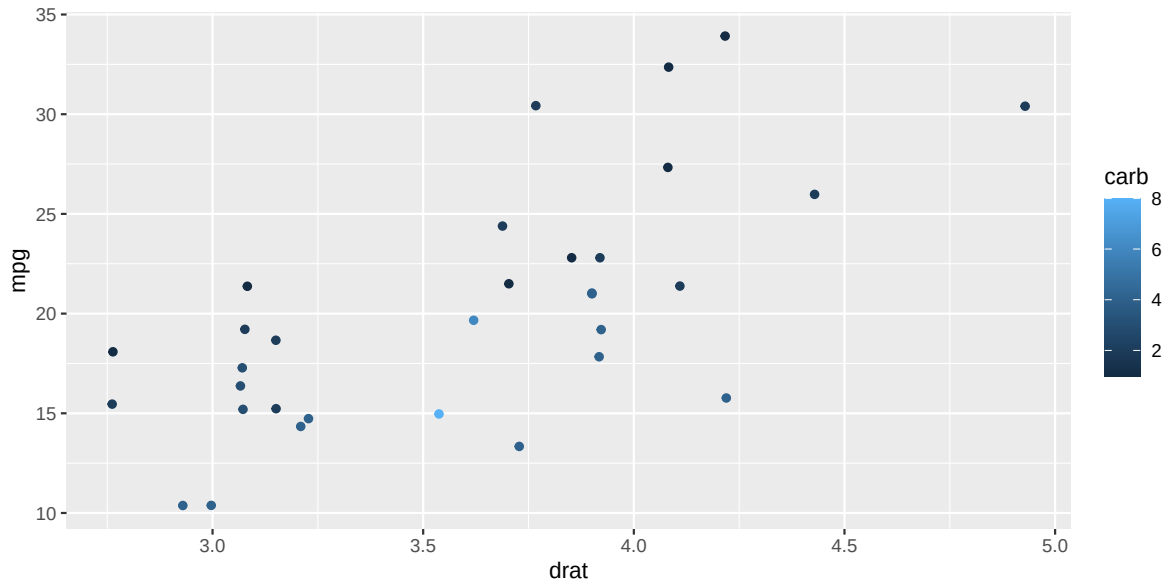



```
myplot + geom_histogram(binwidth=200,
    position = position_stack(),
    alpha=.8,
```

```
    col='black'
) + theme(legend.position='none')
```



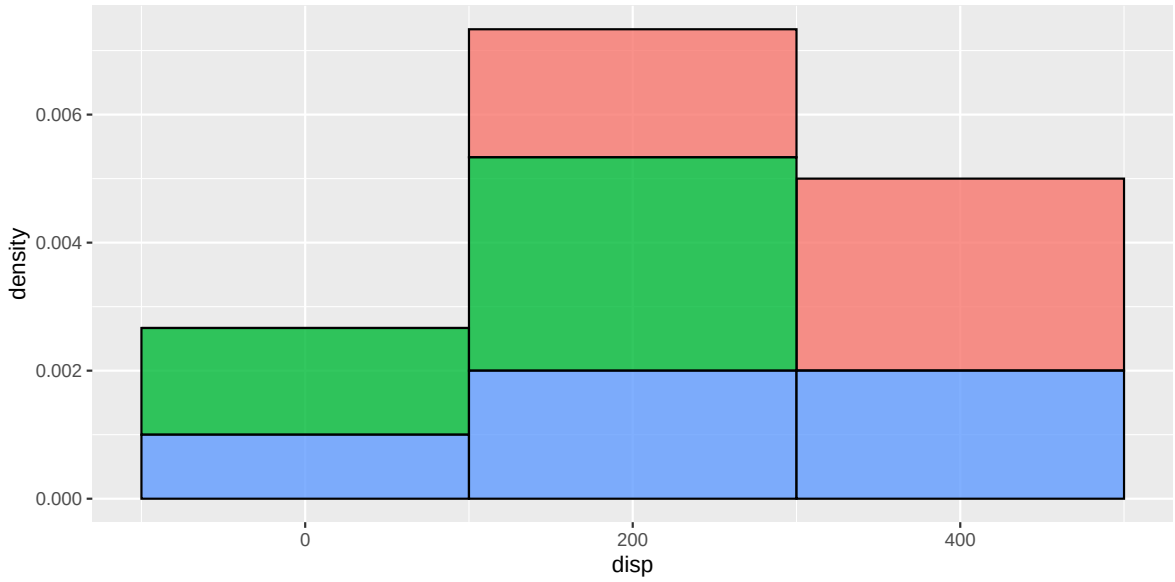

```
ggplot(data=mtcars,
    aes(y=mpg,x=drat,color=carb)
) + geom_jitter()
```



- Generated variables are predefined transformations
  - start and end with `..`
  - exist for some `geom_*( )` and `stat_*( )`
  - example: a histogram uses a `..density..` to avoid counts

```
myplot + geom_histogram(
    aes(y=..density..),
    binwidth=200,
    position = position_stack(),
    alpha=.8,col='black'
) + theme(legend.position='none')
```

```
Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
i Please use `after_stat(density)` instead.
```



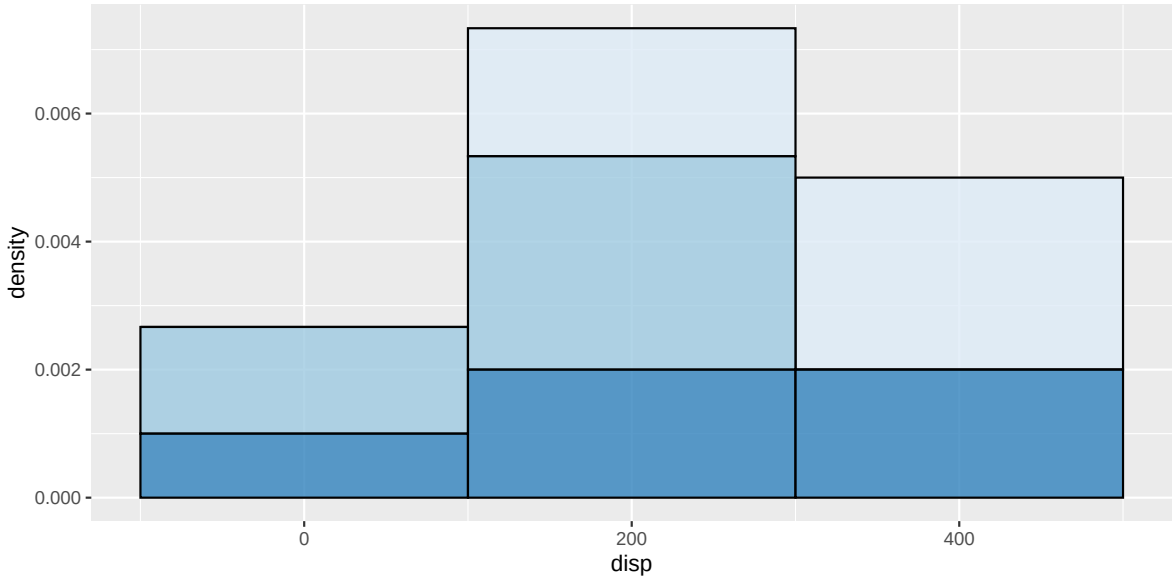
scale_*_*( )

- Each dimension (aesthetic) has a scale
  - serves as a legend
  - helps with interpretation
- Refer to scale with aesthetic and type

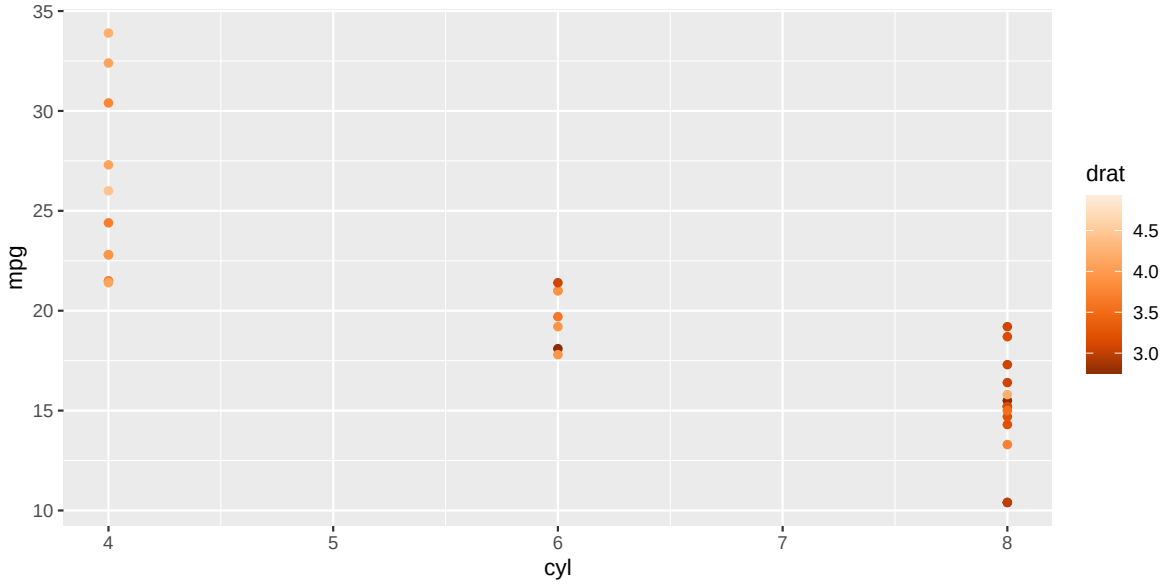
  - structure: scale___
  - examples:
    * scale_x_continuous to assign continuous scale to x-axis aesthetic
    * scale_x_sqrt to square root transform the x-axis aesthetic
    * scale_color_brewer to assign brewer colors to the discrete color aesthetic
    * scale_color_distiller to assign brewer colors to the continuous color aesthetic
    * scale_fill_gradient to assign colors to the fill aesthetic
    * scale_fill_manual to manually assign colors to the fill aesthetic
  - arguments control titles, breaks, labels, limits, … see the help file
  - note: guide argument requires either a name or `guides()` function for additional control

- Scale types have impact on the visualization and legend (continuous vs. categorical)

```
ggplot(data=mtcars,
    aes(x=disp,fill=factor(gear))
) + geom_histogram(
    aes(y=..density..),
    binwidth=200,
    position = position_stack(),
    alpha=.8,
    col='black'
) + theme(legend.position='none') +
scale_fill_brewer()
```



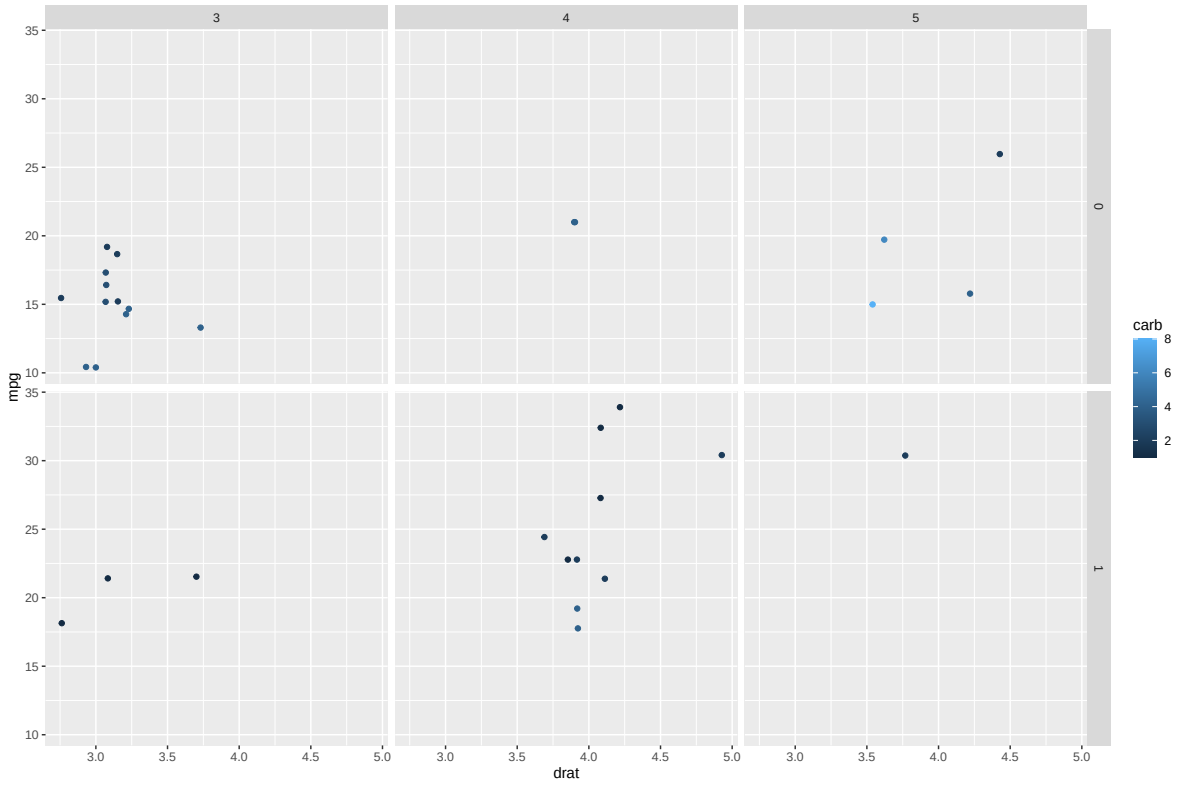
```
ggplot(data=mtcars,
    aes(y=mpg,x=cyl,
    color=drat)
) + geom_point() +
scale_color_distiller(palette='Oranges')
```



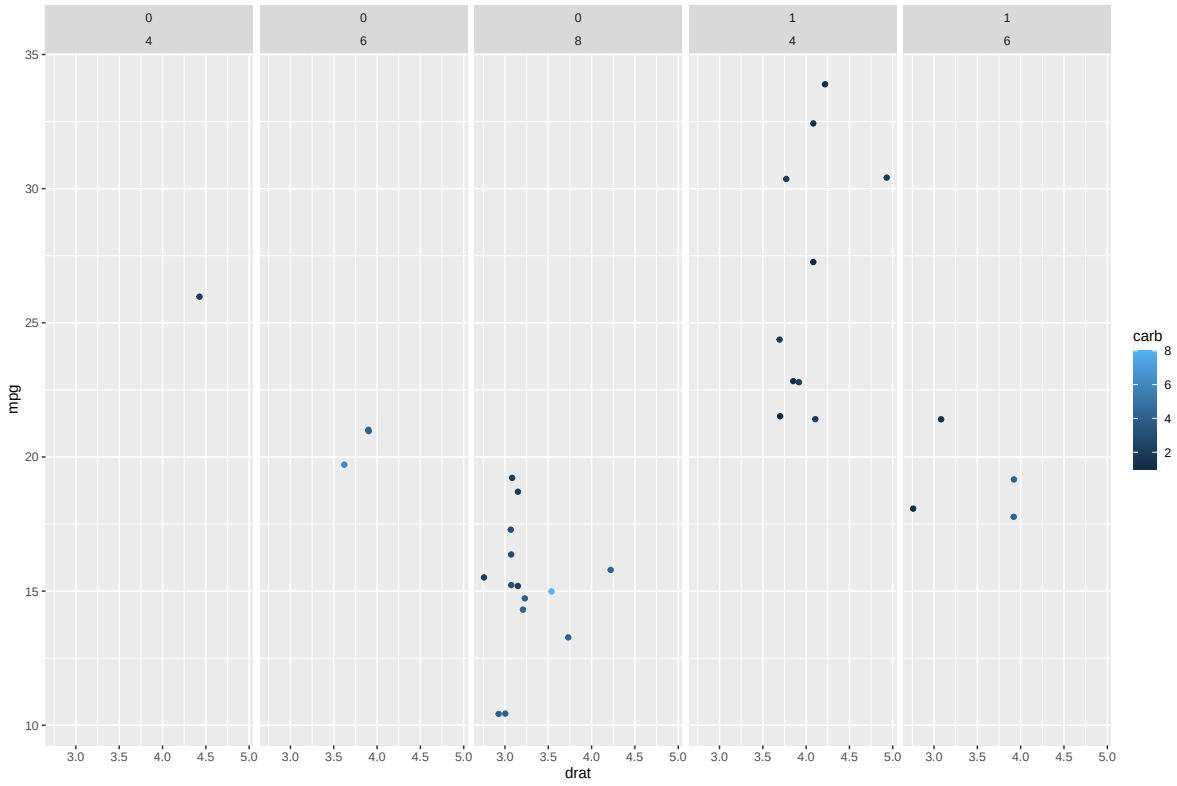
facet_*( )

- Visualizations can be split for subgroups, facilitating conditional comparisons
  - facets can be useful to keep plots simple
  - by default, the axes are kept constant for comparison, changeable
  - `facet_grid( )` uses a grid, `facet_wrap( )` keeps filling space
- Requires a row and/or column specification
  - separate rows and columns by `~`, use `.` if none
  - use `+` to combine multiple row and/or column specification

```
ggplot(data=mtcars,
    aes(y=mpg,x=drat,color=carb)
) +
geom_jitter() +
facet_grid(vs~gear)
```



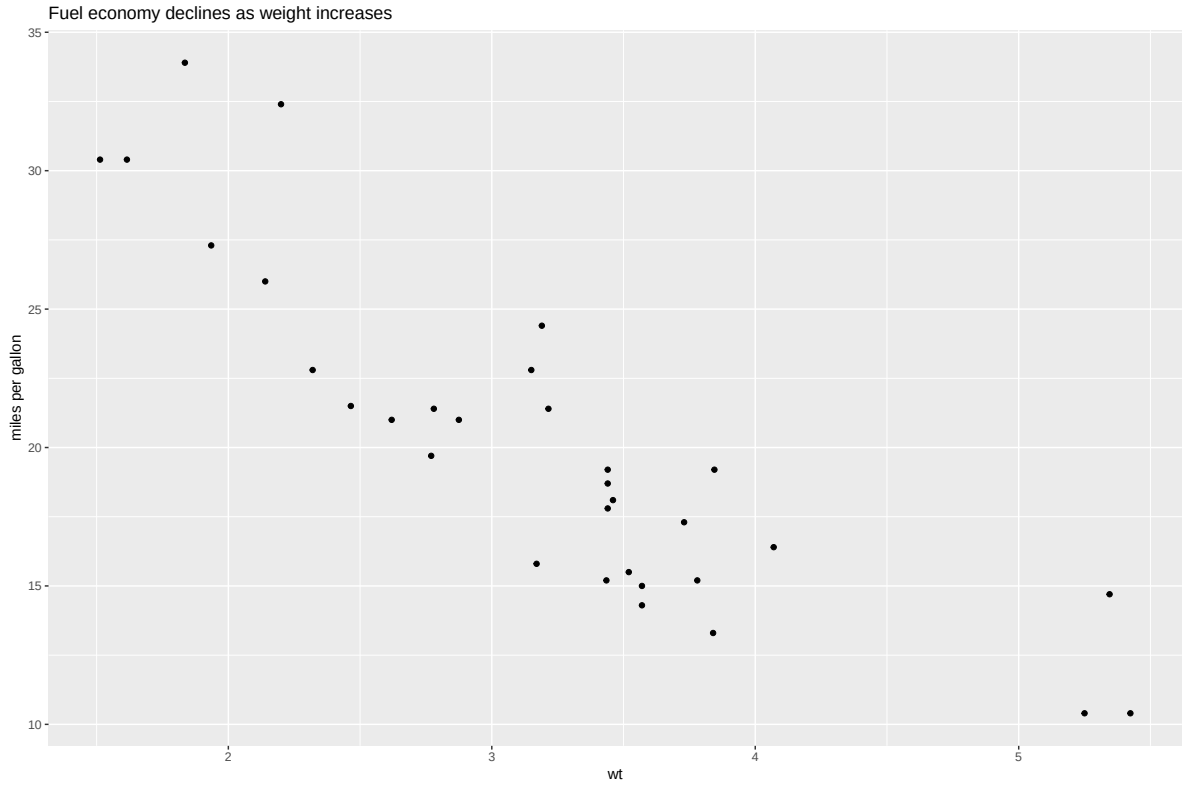

```
ggplot(data=mtcars,
    aes(y=mpg,x=drat,color=carb)
) +
geom_jitter() +
facet_grid(.~vs+cyl)
```



theme( )

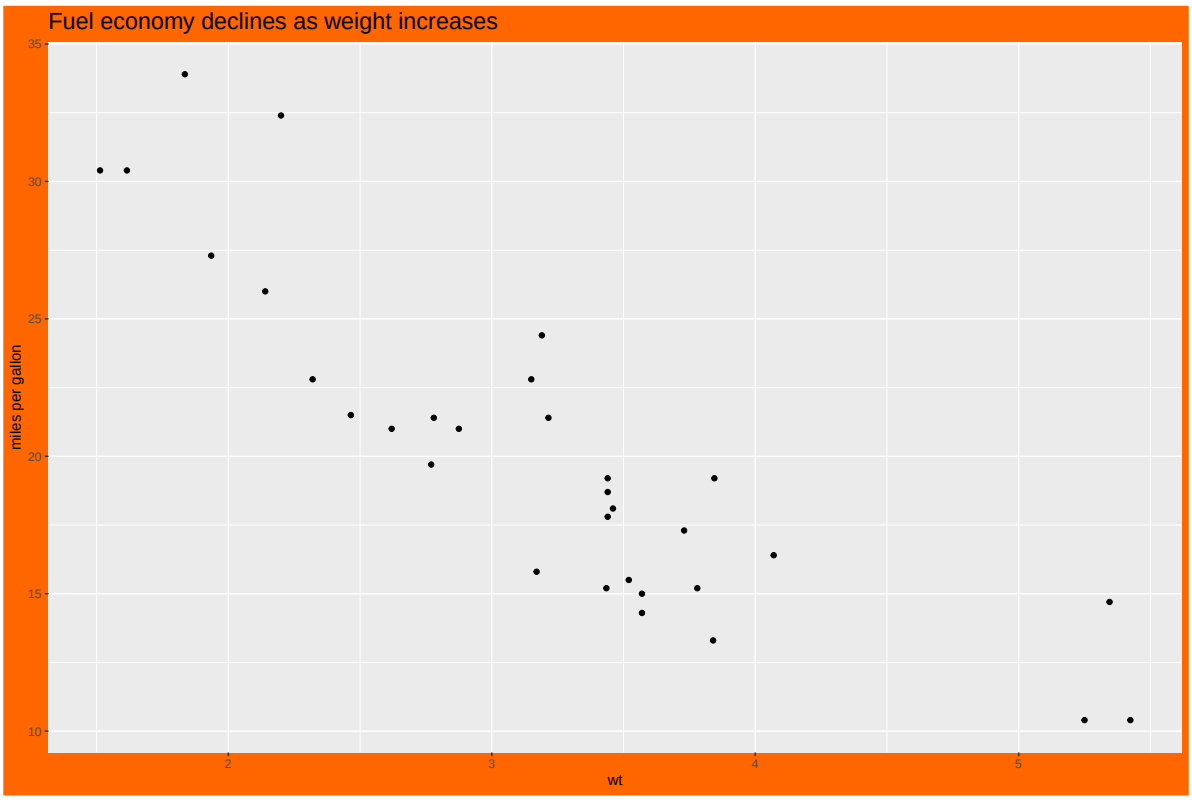
- Themes offer control, unrelated to data
  - themes are layers to (re-)specify theme elements
  - elements are numerous (`?theme`):
    - line, rect, text, title
    - axis: axis.title, axis.text.x.top, axis.ticks.x.bottom, …
    - legend: legend.spacing.y, legend.key.width, legend.justification, …
    - panel: panel.background, panel.grid.major, panel.ontop, …
    - plot: plot.background, plot.caption, plot.tag, plot.margin, …
    - strip: strip.background, strip.text, strip.switch.pad.wrap, …
  - elements are controlled with a theme function, eg., `element_text( )`
    - `element_text( )`, `element_line( )`, `element_rect( )`, `margin( )`, …
  - default themes exist, eg., `theme_minimal( )`, new can be created
  - `labs( )` is a simpler way to specify titles and labels

```
(myplot <- ggplot(mtcars,
    aes(wt, mpg)
) +
geom_point() +
labs(
    title = "Fuel economy declines as weight increases",
    y='miles per gallon')
)
```



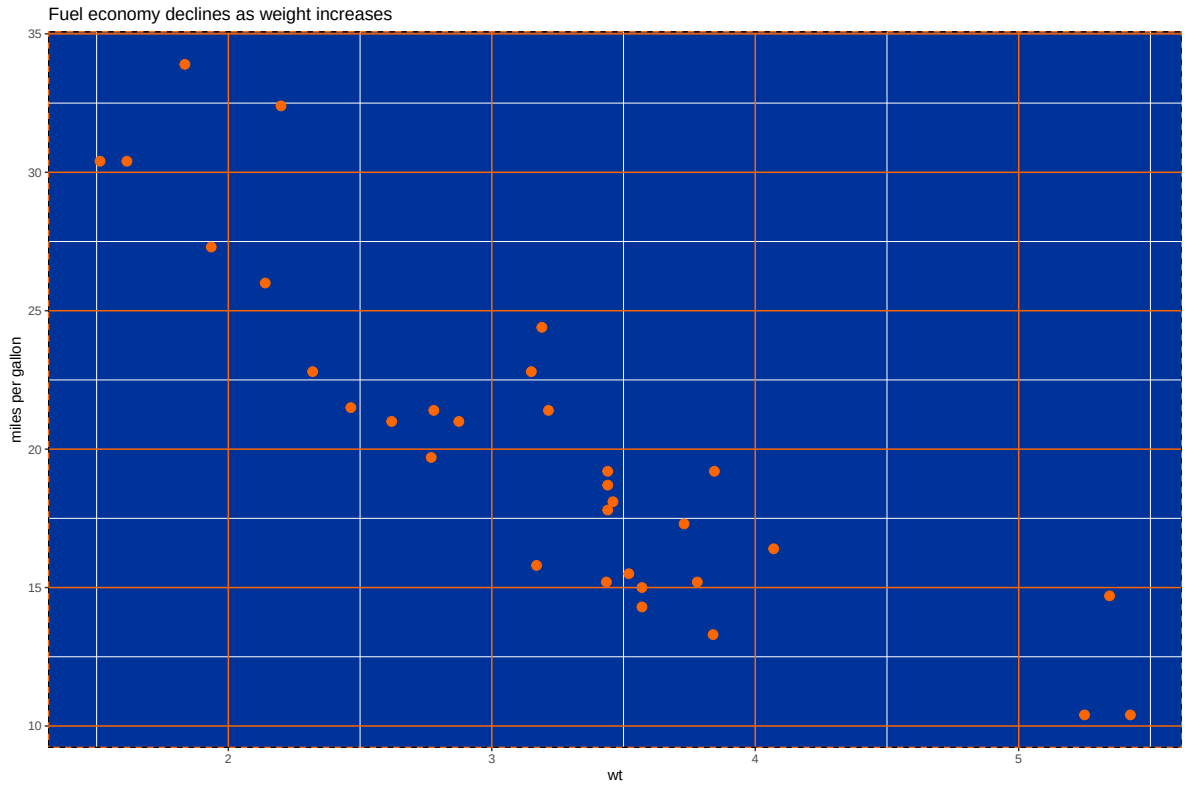
- The size of the plot title is changed with the `element_text( )`, the background with the `element_rect( )`

```
myplot + theme(plot.title = element_text(size=rel(1.5)),
    plot.background=element_rect(fill="#FF6600"))
```

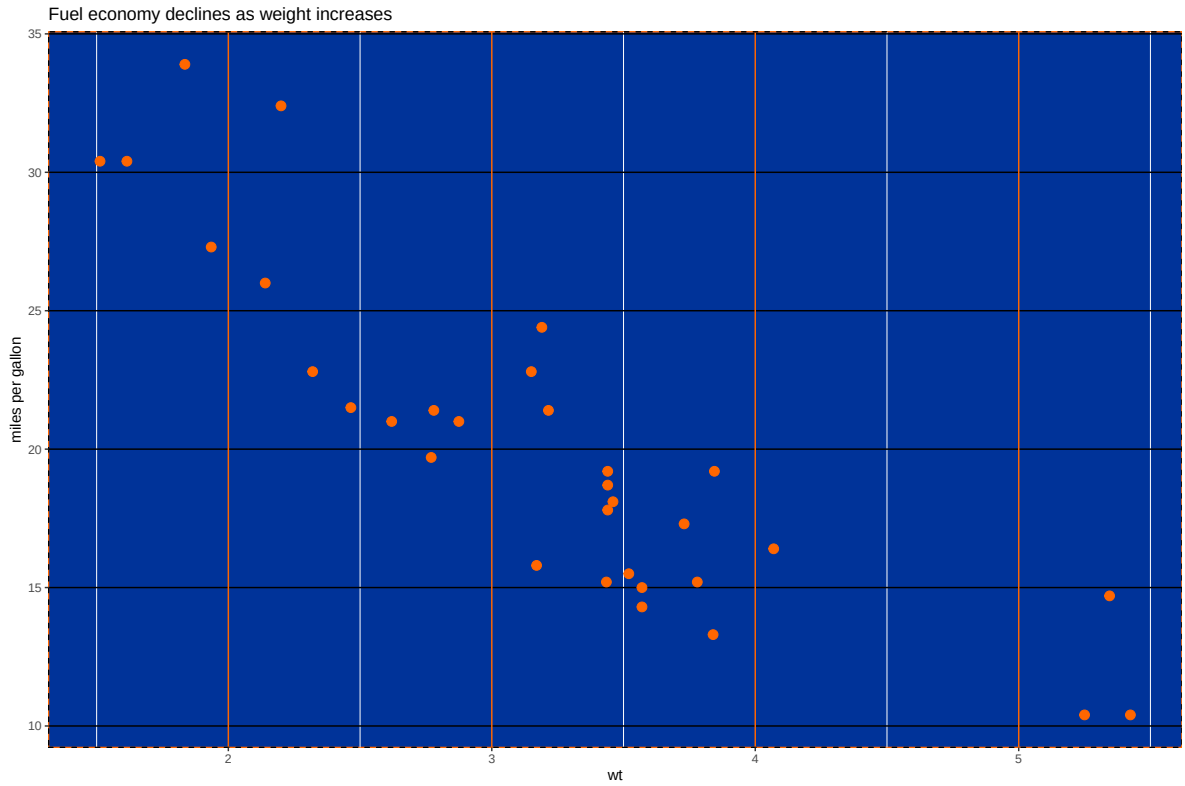


- The inside of the plot is the panel, modifiable too.
  - multiple elements can be specified in one theme
  - multiple themes can be specified by layers

```
(myplot <- myplot +
    geom_point(color='#FF6600',size=3) +
        theme(panel.background =
            element_rect(fill = "#003399",
        colour = "#FF6600")
) +
theme(
    panel.border = element_rect(
        linetype = "dashed",
        fill = NA
    ),
    panel.grid.major = element_line(
        colour = "#FF6600")
    )
)
```




```
myplot + theme(
  panel.grid.major.y = element_line(colour = "black"),
  panel.grid.minor.y = element_blank()
)
```

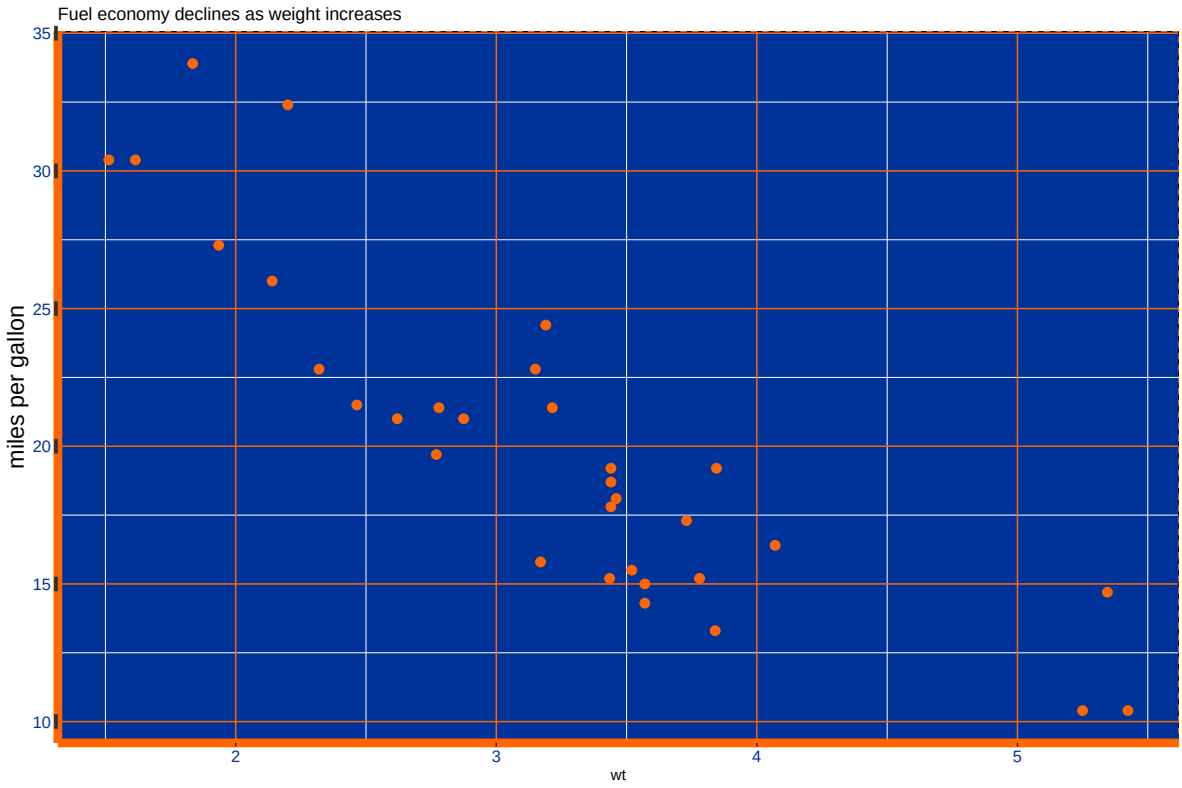


- On the outside are the axes and titles, modifiable too
  - the text, ticks and titles are adjusted for color and size
  - function `element_text( )`, `element_line( )` and `unit( )` are used

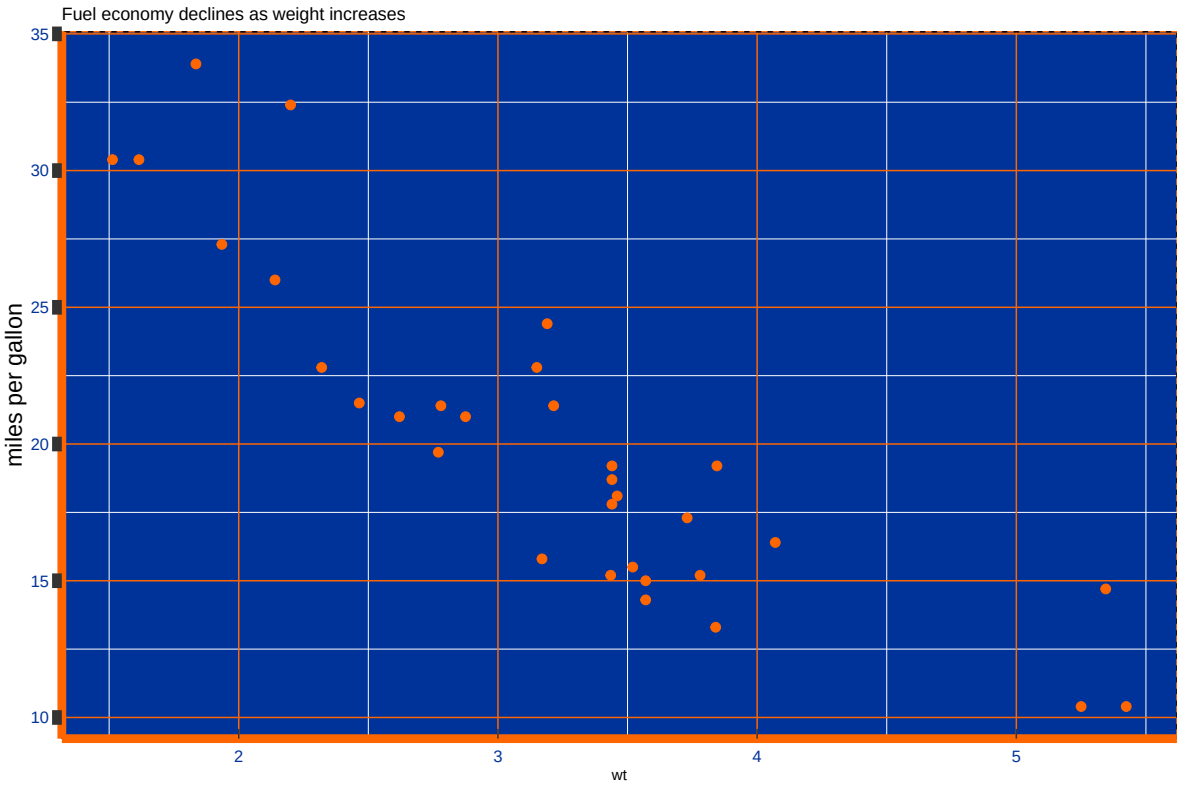
```
(myplot <- myplot + theme(
    axis.line = element_line(
        size = 3,
        colour = "#FF6600")
    ) +
    theme(axis.text = element_text(
        colour = "#003399",
        size=12)
    ) +
    theme(axis.ticks.y =
        element_line(size = 5)
    ) +
    theme(axis.title.y =
        element_text(size = rel(1.5))
    )
```

```
)
```

```
Warning: The `size` argument of `element_line()` is deprecated as of ggplot2 3.4.0.
i Please use the `linewidth` argument instead.
```

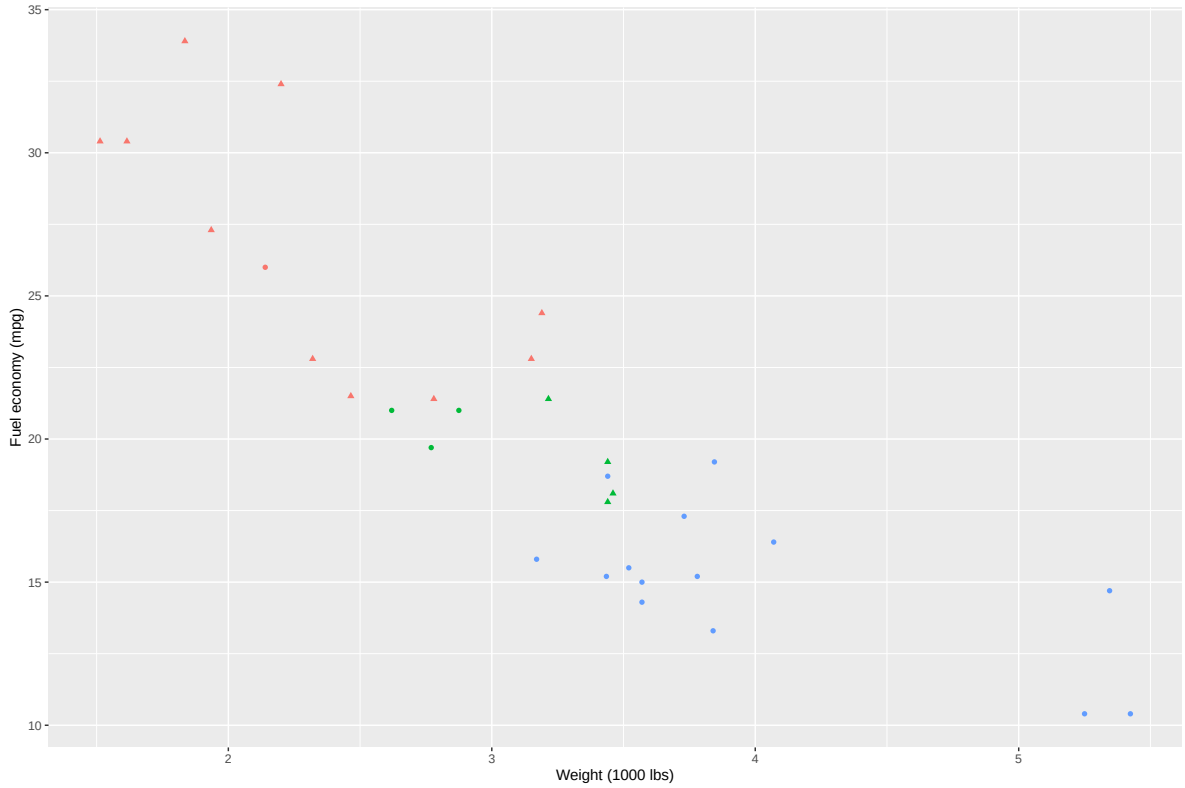



```
myplot + theme(
  axis.ticks.length.y = unit(.25, "cm"),
  axis.ticks.length.x = unit(-.25, "cm"),
  axis.text.x = element_text(
    margin = margin(t = .3, unit = "cm")
    )
)
```

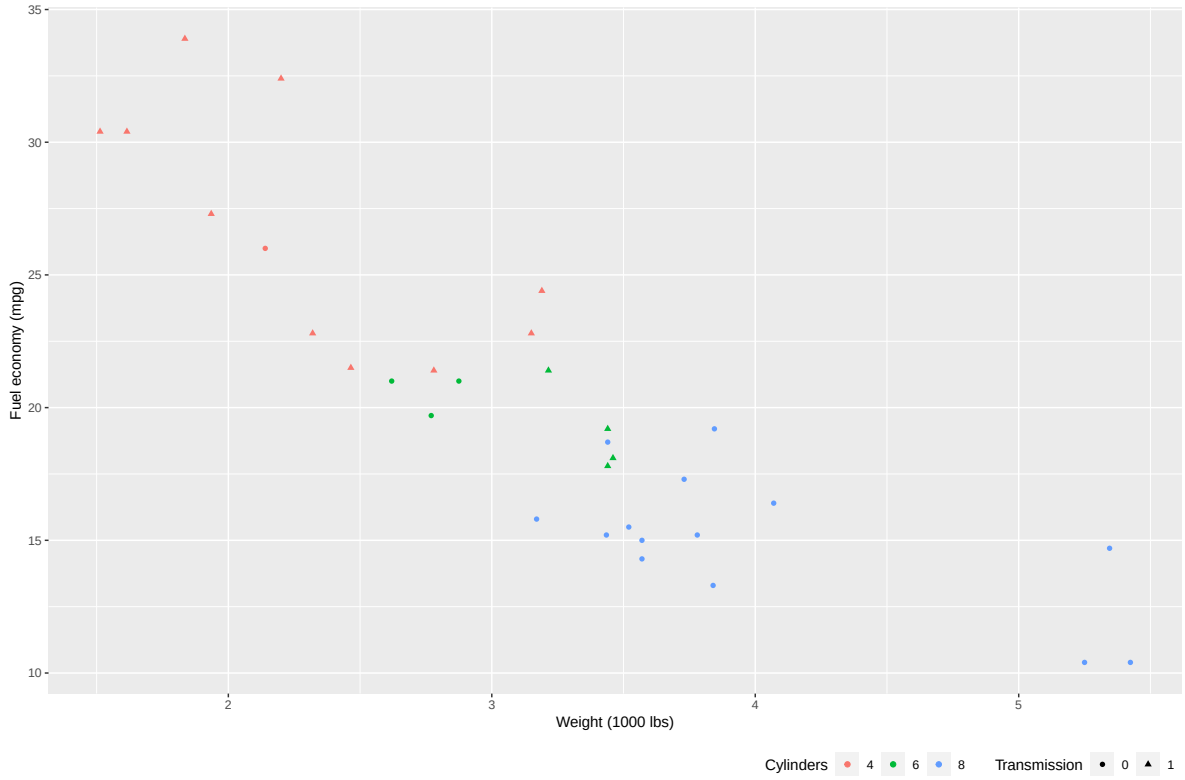


- Scales are represented by legends, modifiable too
  - labs can be used to change multiple legend titles
  - legends can be positioned and formatted

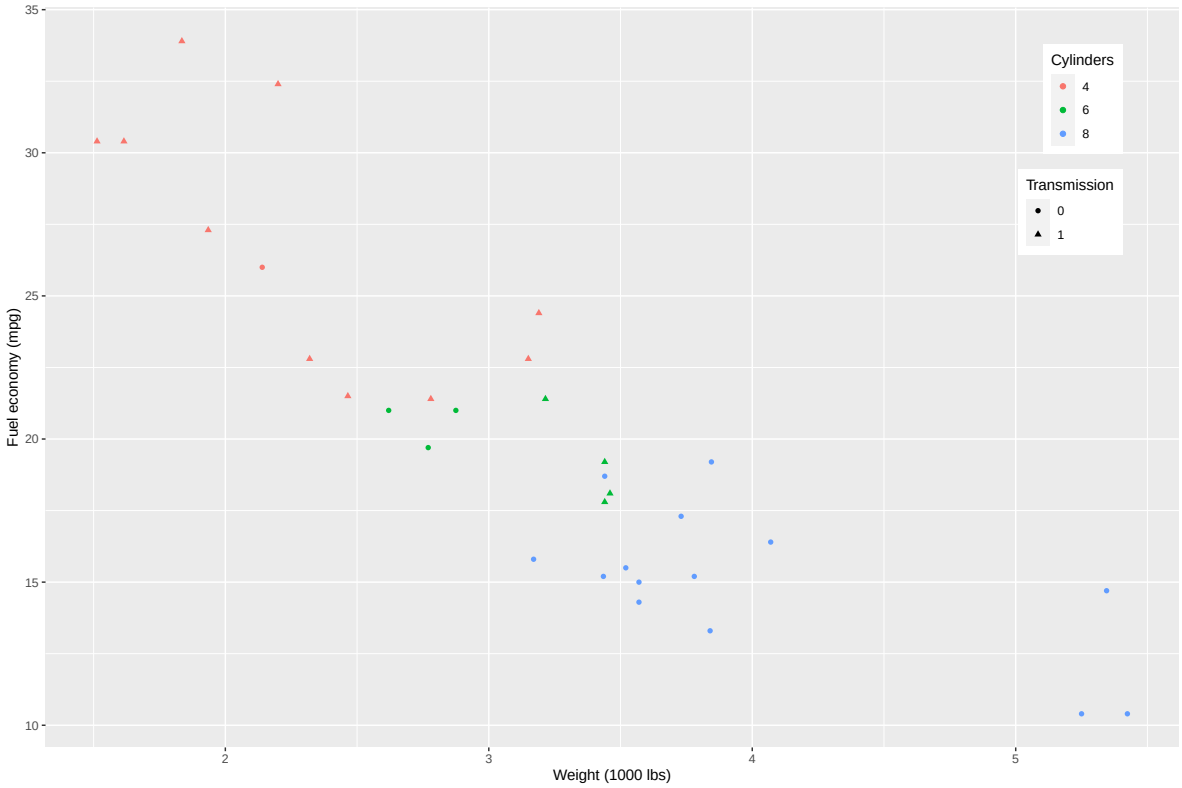
```
myplot <- ggplot(mtcars,
    aes(wt, mpg)) +
    geom_point(
        aes(colour = factor(cyl),
            shape = factor(vs))
    ) +
    labs(
        x = "Weight (1000 lbs)",
        y = "Fuel economy (mpg)",
        colour = "Cylinders",
        shape = "Transmission"
    )

myplot + theme(legend.position='none')
```




```
myplot + theme(
    legend.justification = "right",
    legend.position = "bottom"
    )
```

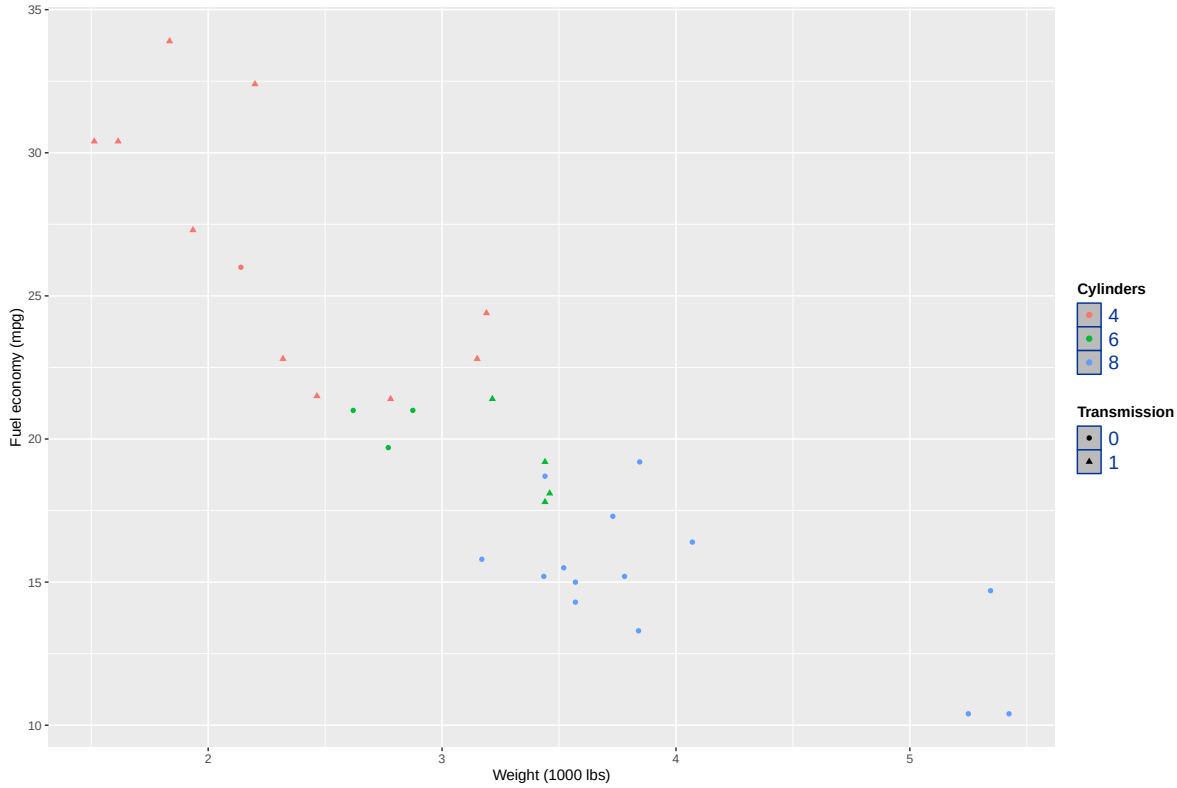



```
myplot + theme(
  legend.position = c(.95, .95),
  legend.justification = c("right", "top"),
  legend.box.just = "right",
  legend.margin = margin(6, 6, 6, 6)
)
```



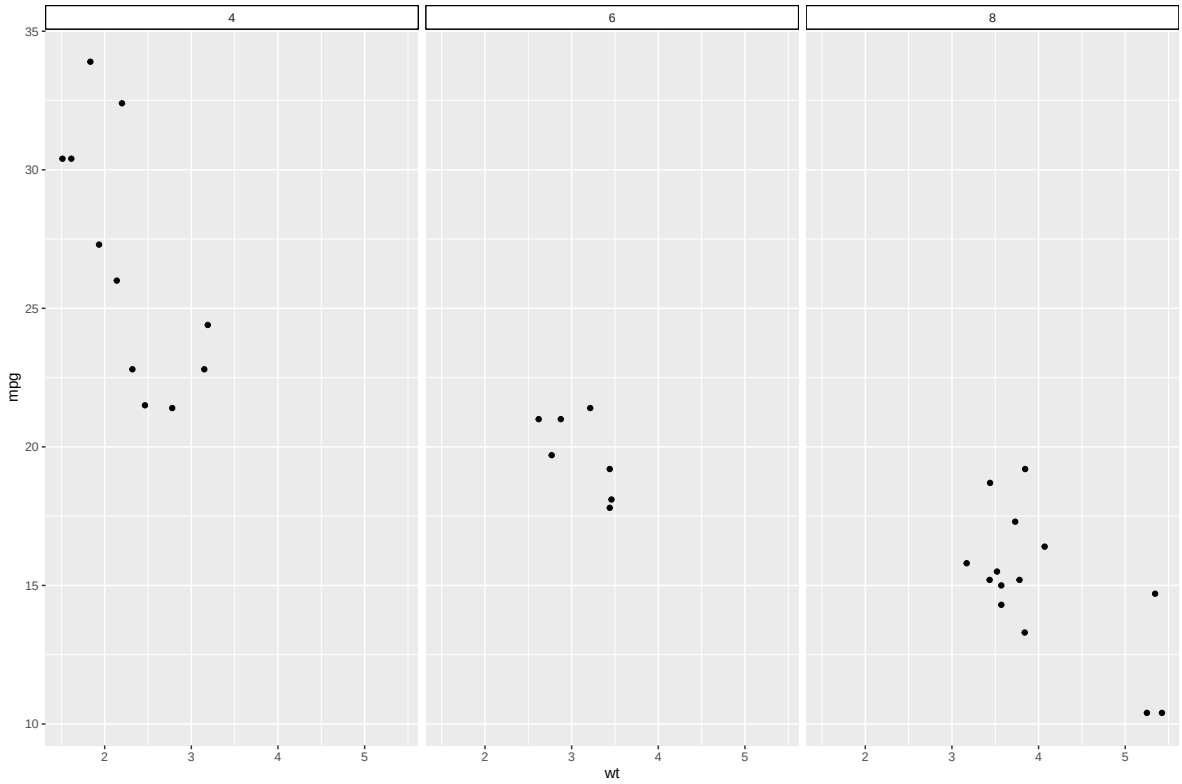
- Keys inside legends are modifiable too
  - labs can be used to change multiple legend titles
  - legends can be positioned and formatted, for key, text and title

```
myplot + theme(
    legend.key = element_rect(
        fill = "#bbbbbb",
        colour = "#003399")
    ) +
    theme(legend.text = element_text(
        size = 14,
        colour = "#003399")
    ) +
    theme(legend.title = element_text(
        face = "bold")
    )
```

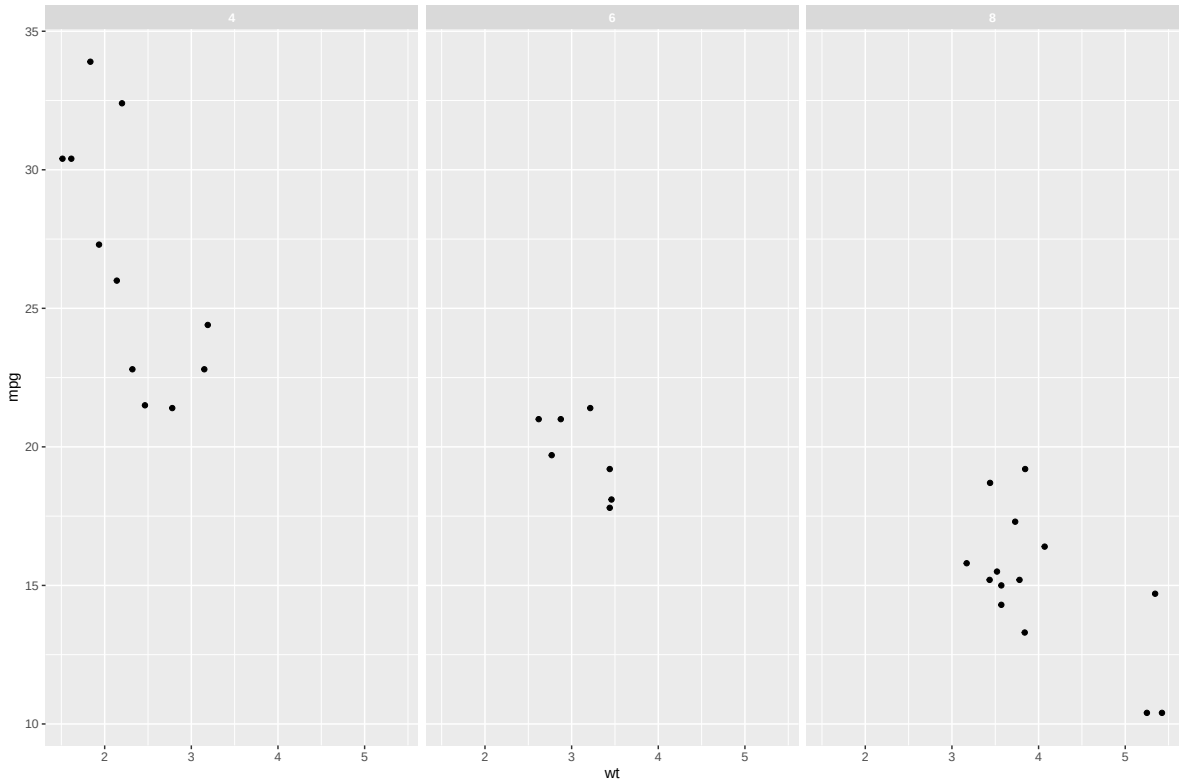


- Themes also work on facets, at which strips are defined

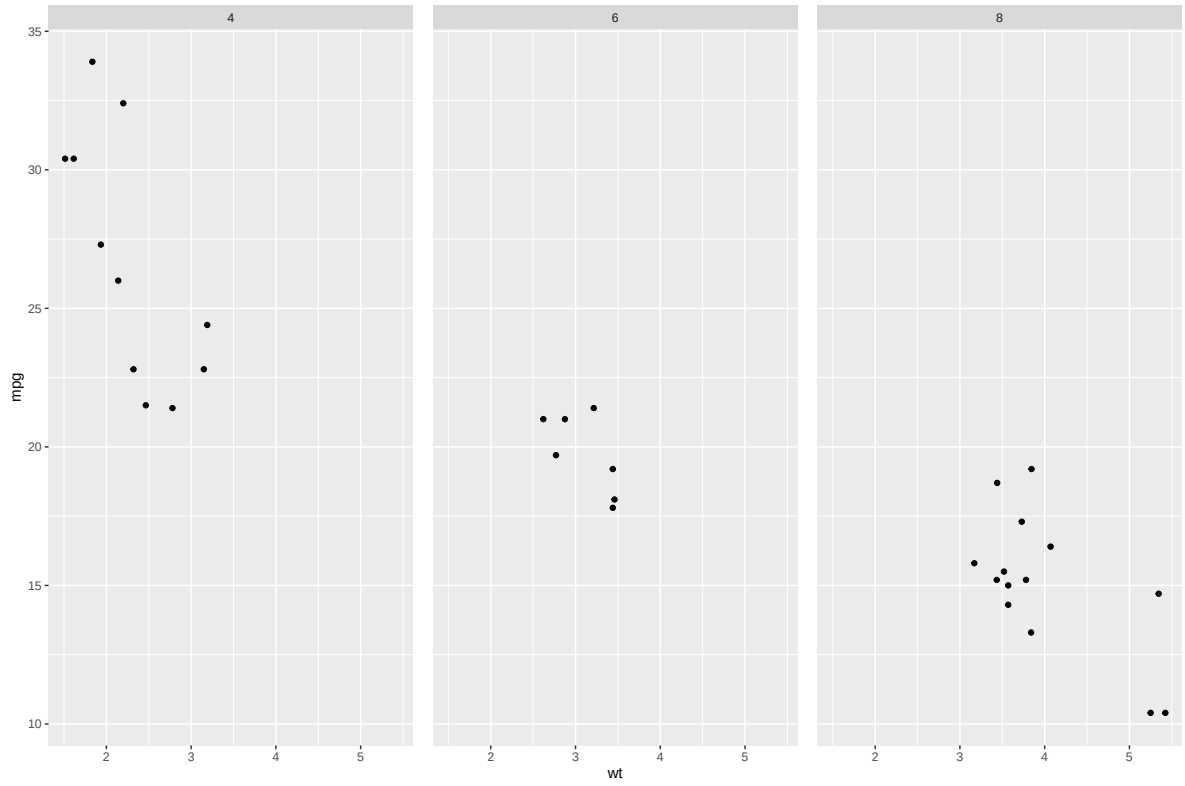
```
myplot <- ggplot(mtcars,
        aes(wt, mpg)
    ) +
    geom_point() +
    facet_wrap(~ cyl)

myplot + theme(strip.background =
    element_rect(colour = "black",
    fill = "white")
)
```




```
myplot + theme(strip.text.x =
    element_text(colour = "white",
    face = "bold")
)
```




```
myplot + theme(panel.spacing =
    unit(1, "lines")
)
```

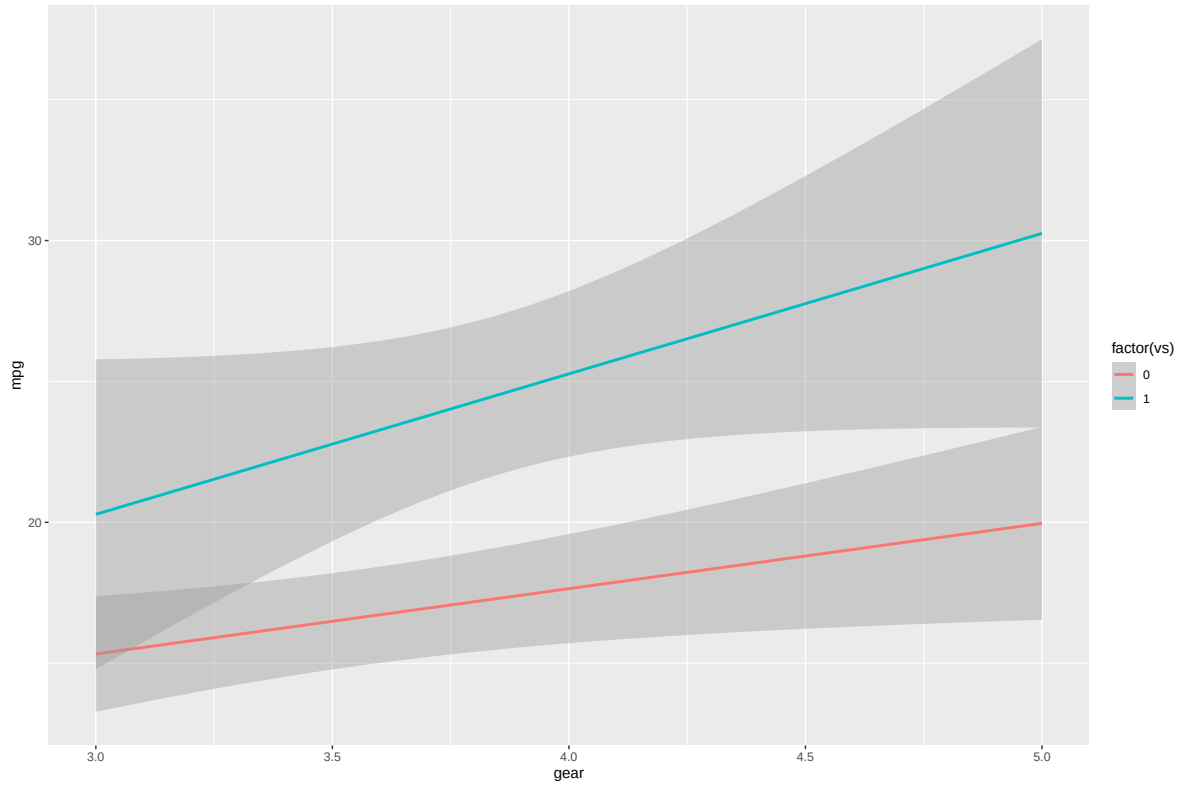


coord_*( )

- Typically the default cartesian coordinate system is used, `coord_cartesian( )`
- Limits of the axes are best specified within the `coord_*( )` function
  - works like a zoom
  - alternatively, use `xlim( )` and `ylim( )`
    - beware that values outside the boundary are treated as missing

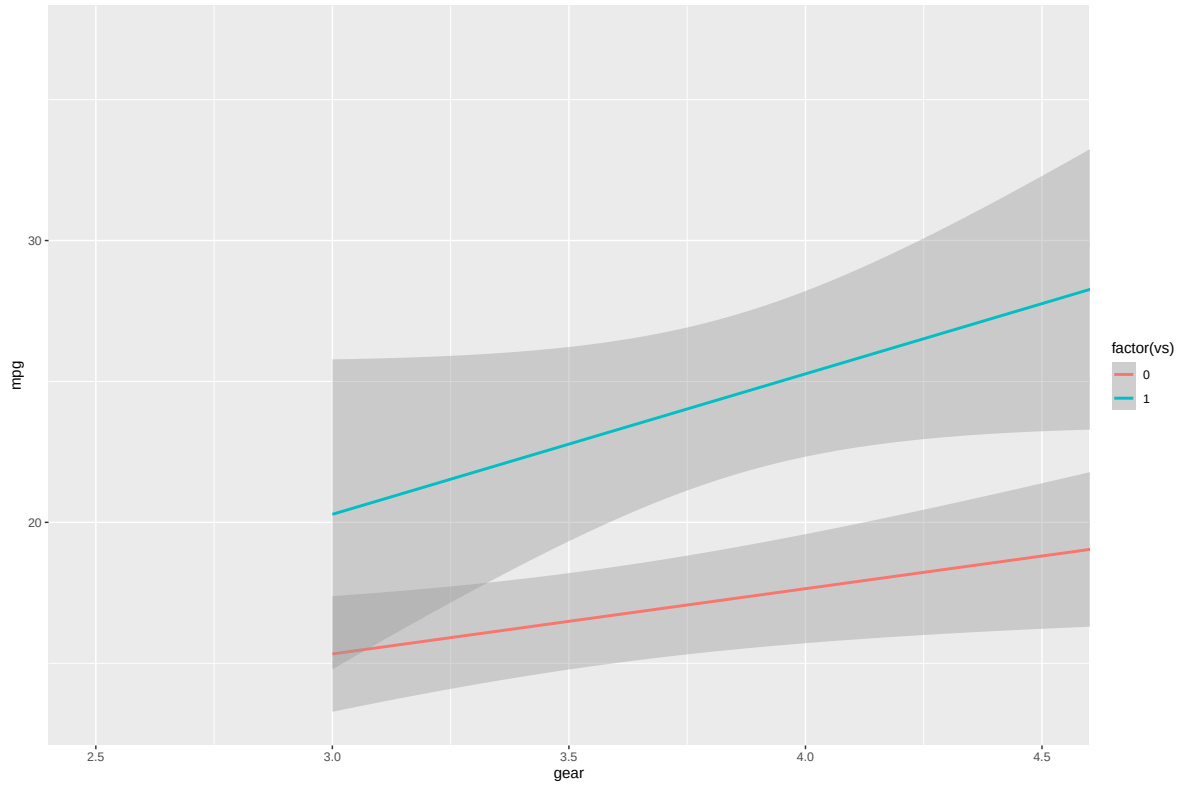
```
myplot <- ggplot(data=mtcars,
    aes(y=mpg,x=gear,
    color=factor(vs),
    group=vs)
)
myplot + geom_smooth(method='lm')
```

```
`geom_smooth()` using formula = 'y ~ x'
```




```
myplot + geom_smooth(method='lm') +
    coord_cartesian(xlim=c(2.5,4.5))
```

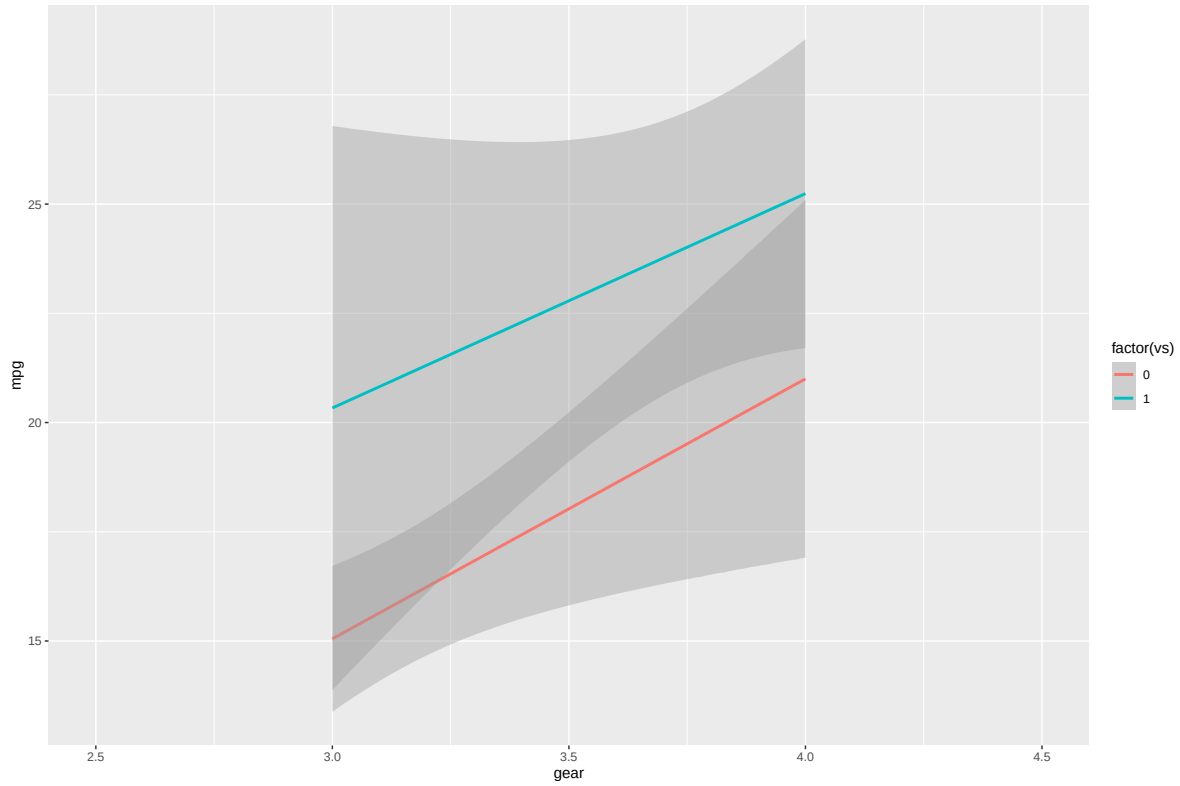
```
`geom_smooth()` using formula = 'y ~ x'
```




```
myplot + geom_smooth(method='lm') +
    xlim(2.5,4.5)
```

```
`geom_smooth()` using formula = 'y ~ x'
```

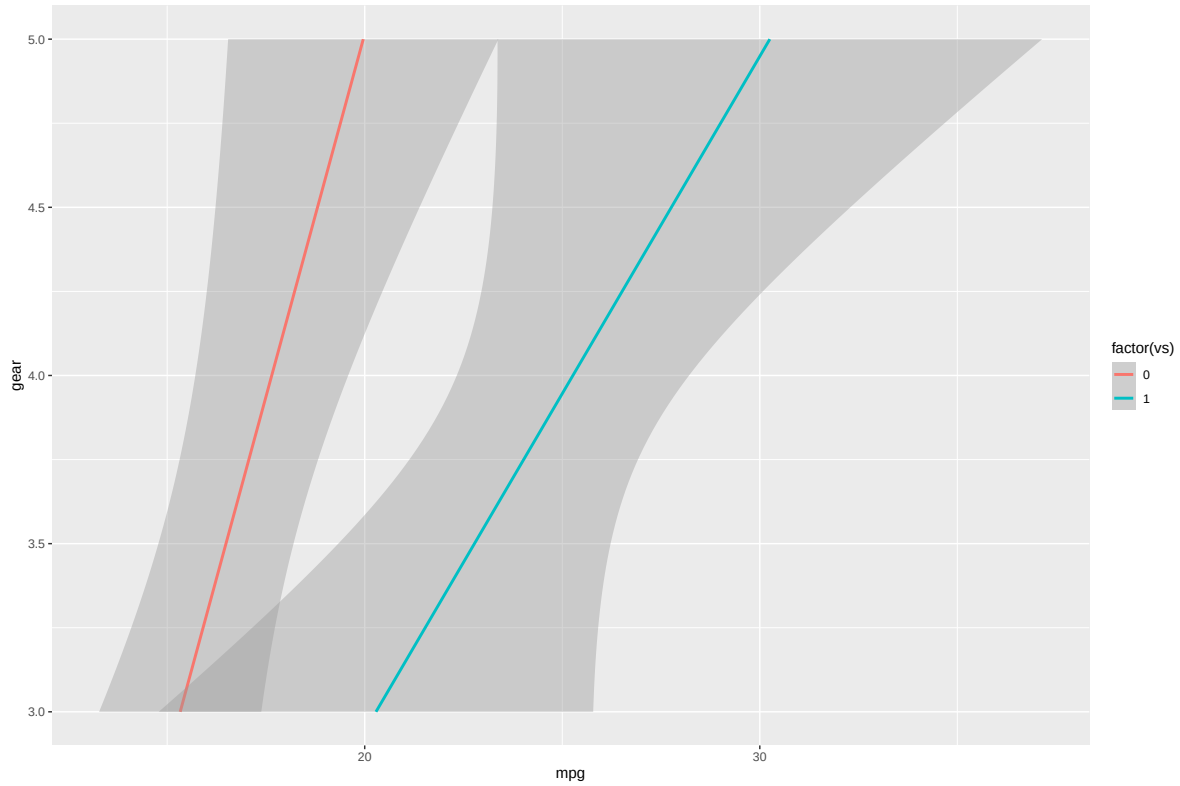
```
Warning: Removed 5 rows containing non-finite values (`stat_smooth()`).
```



- Within the cartesian family alternatives exist
  - `coord_flip( )` switches x and y-axis
  - `coord_fixed( )` sets the ratio for x and y values
    - eg., .1 means 1 unit on x is 10 on y
- Alternatives exist, eg., `coord_polar( )`, `coord_trans( )`, and various map related functions

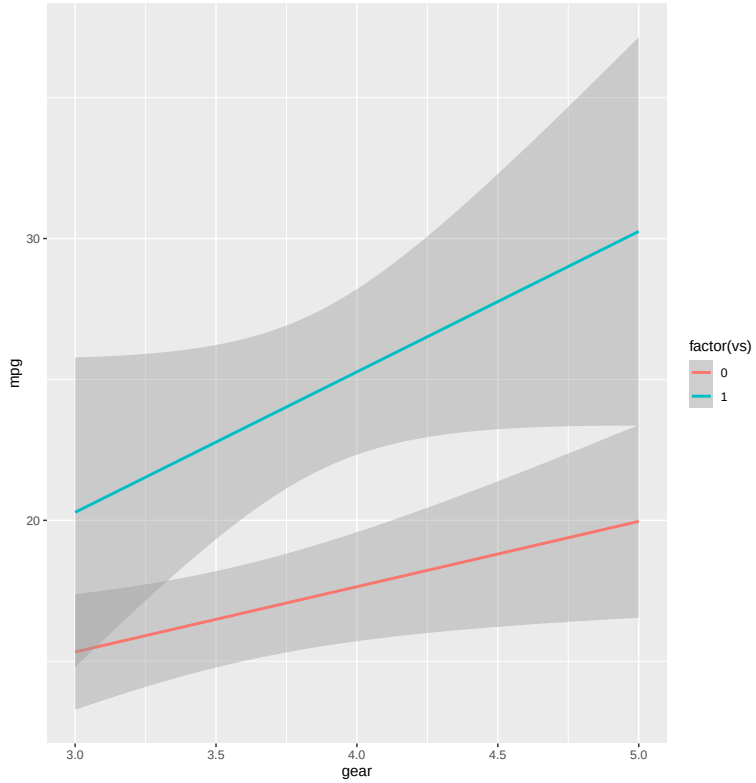
```
myplot + geom_smooth(method='lm') +
    coord_flip()
```

```
`geom_smooth()` using formula = 'y ~ x'
```



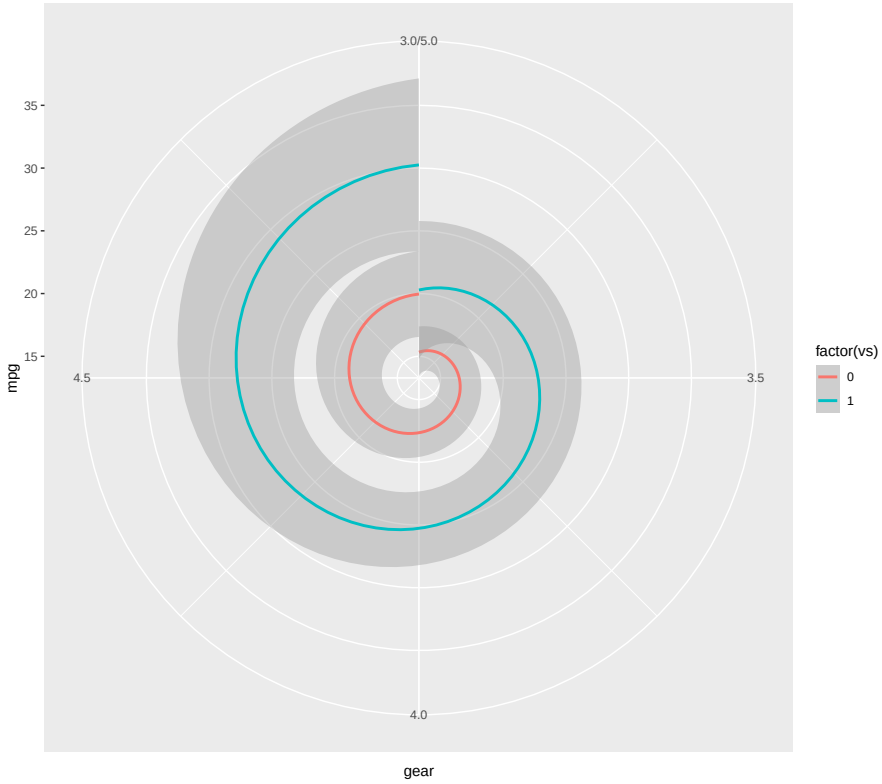

```
myplot + geom_smooth(method='lm') +
    coord_fixed(.1)
```

```
`geom_smooth()` using formula = 'y ~ x'
```




```
myplot + geom_smooth(method='lm') +
    coord_polar()
```

```
`geom_smooth()` using formula = 'y ~ x'
```



summary

- `ggplot` creates a ggplot object, required for visualization
- `aes( )` combines aesthetics and link data to scales, and group them
- `geom_*( )` are geom functions that visualize a ggplot object
- `stat_*( )` are stat functions that also visualize a ggplot object
- `scale_*_*( )` helps fine-tuning visualized dimensions
- `facet_grid( )` or `facet_wrap( )` split dimensions over panels (faceting)
- `coords( )` re-specify the coordinate system, and helps zooming in
- `theme( )` re-specifies data independent characteristics

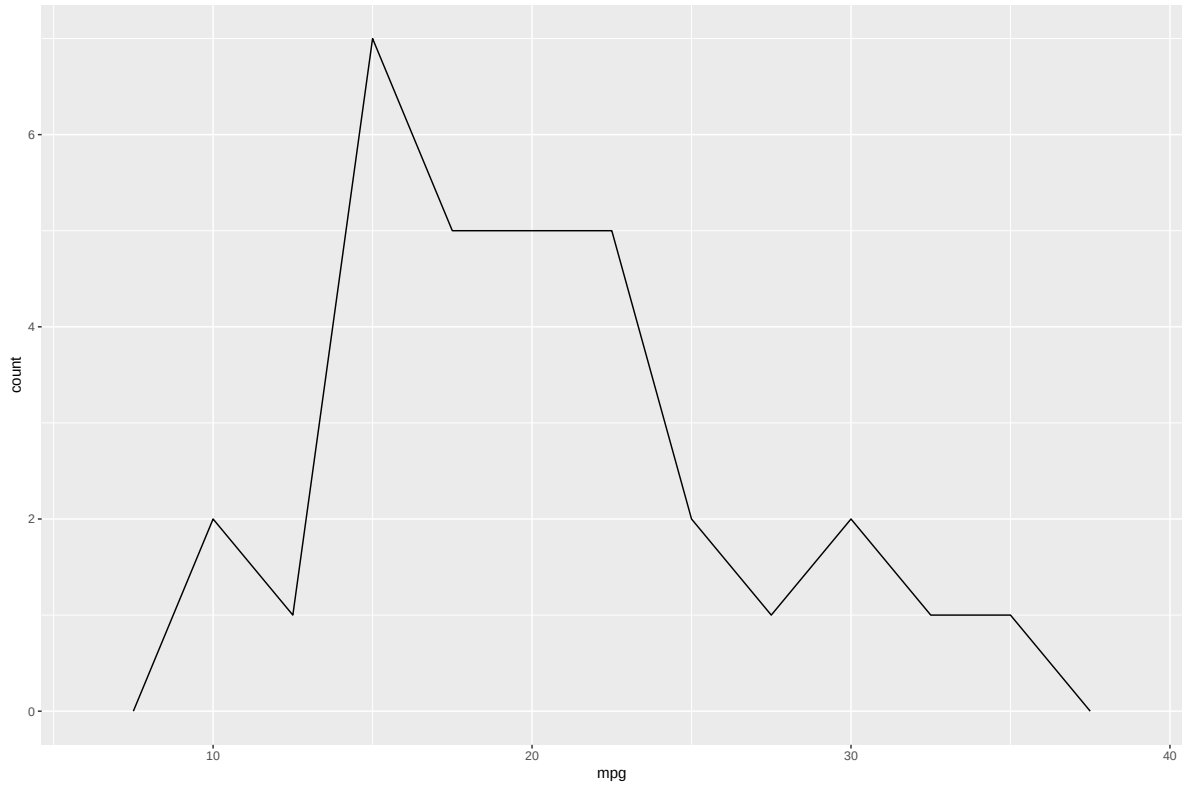
Examples to go into detail

- Through various examples, the cheat sheet is focused upon.
  - https://rstudio.com/resources/cheatsheets/

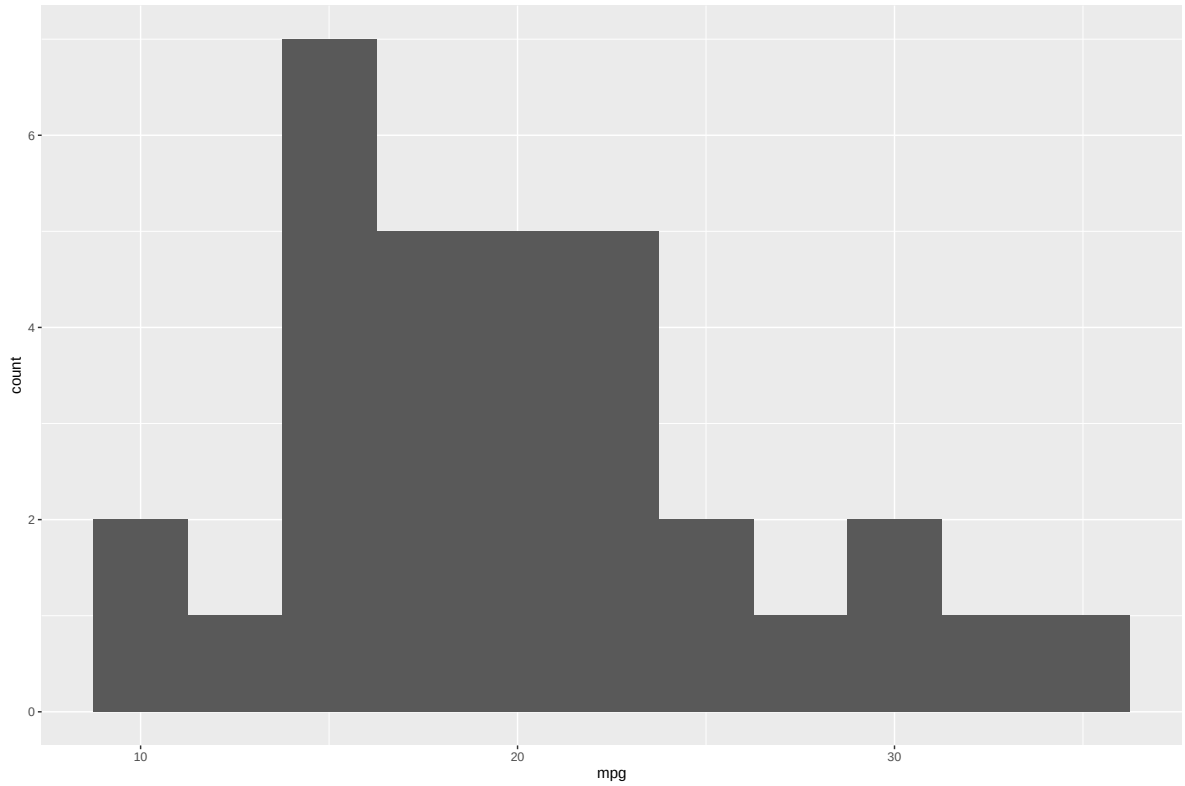
one-variable

- Various visualizations address one particular variable
  - mostly continuous but possibly also discrete
  - continuous variables are typically 'binned'
- Note the frequency polygon, and the histogram.

```
ggplot(data=mtcars,aes(mpg)) +
geom_freqpoly(binwidth=2.5)
```

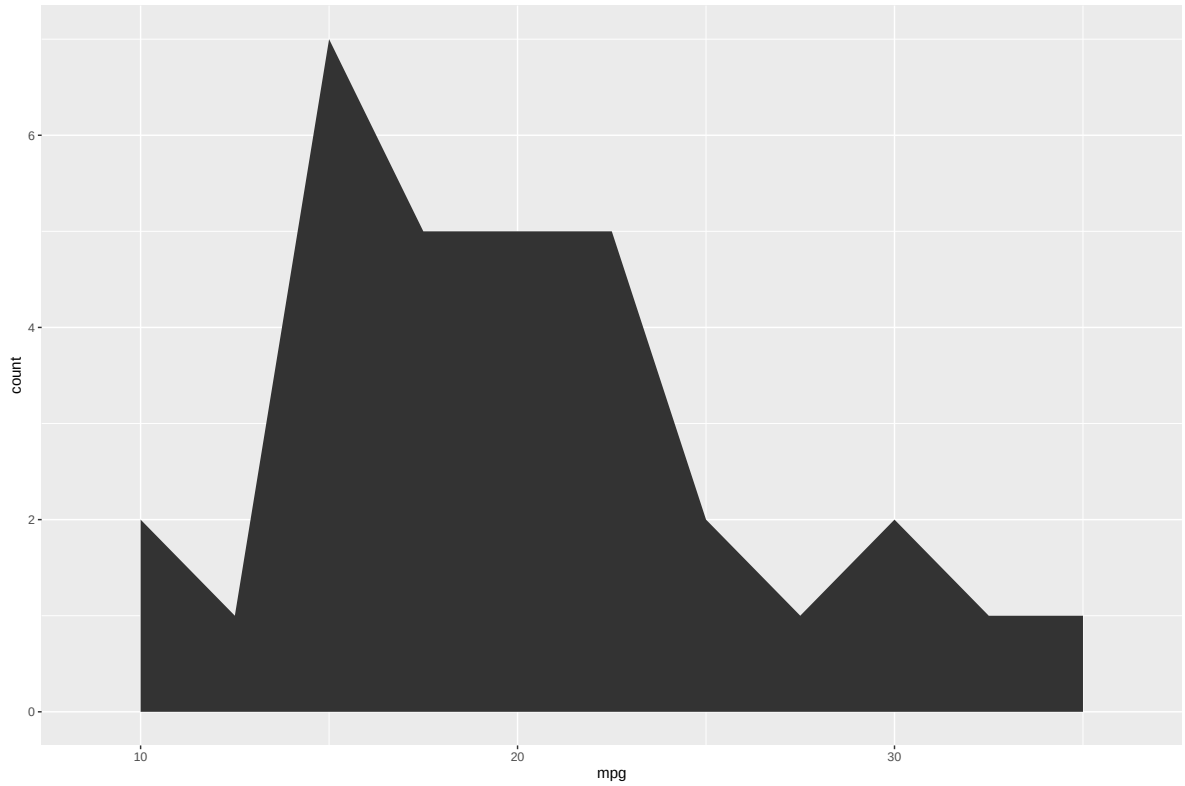



```
ggplot(data=mtcars,aes(mpg)) +
geom_histogram(binwidth=2.5)
```



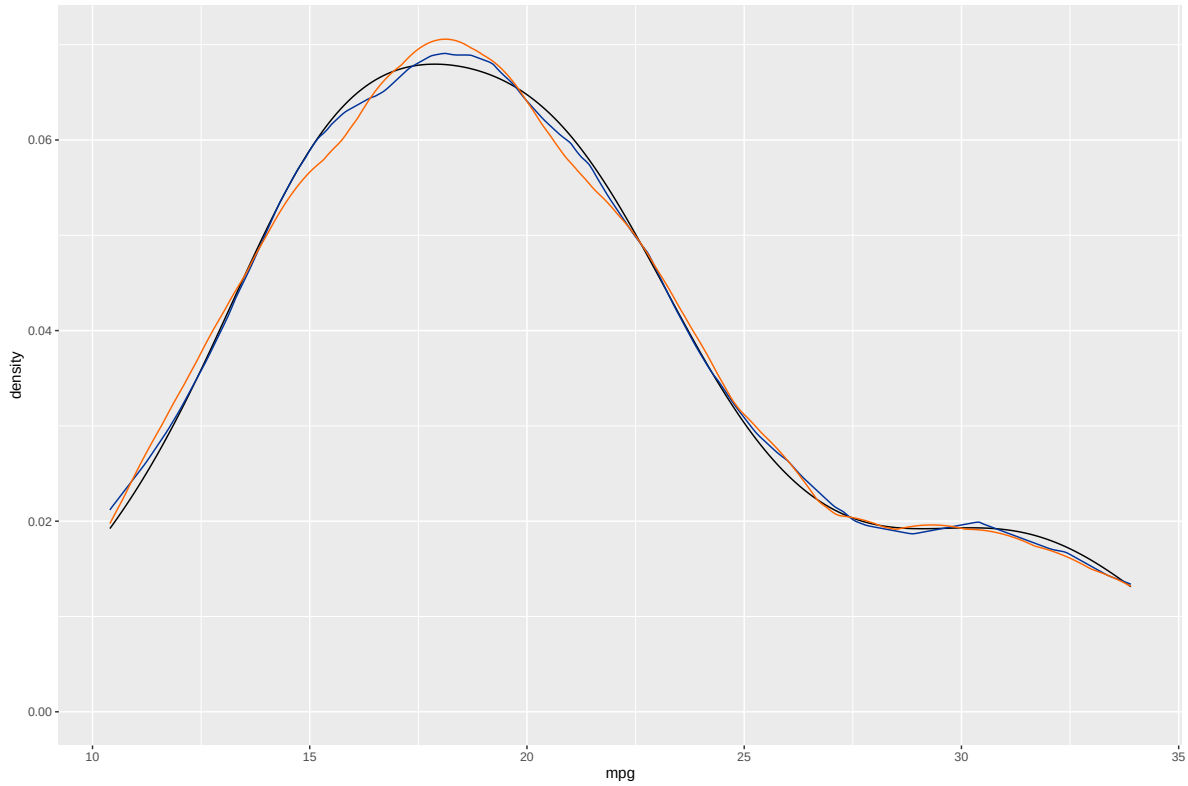
- Using the `geom_area( )`, binning must be explicit as stat argument.

```
ggplot(data=mtcars,aes(mpg)) +
    geom_area(stat='bin',binwidth=2.5)
```



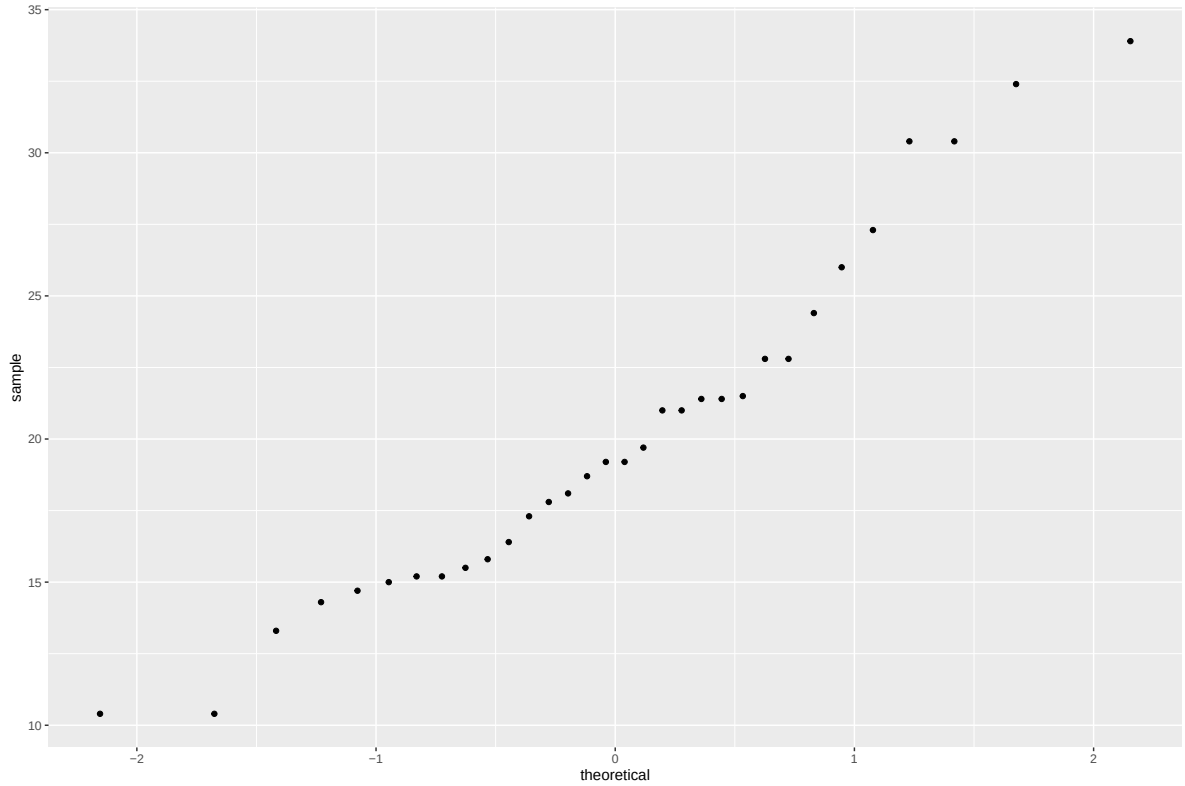
- Continuous variables can also be shown continuously, without binning
- Different types of densities are shown

```
ggplot(data=mtcars,aes(mpg)) + geom_density() +
    geom_density(kernel='triangular',color="#003399") + geom_density(kernel='optcosine',co
```



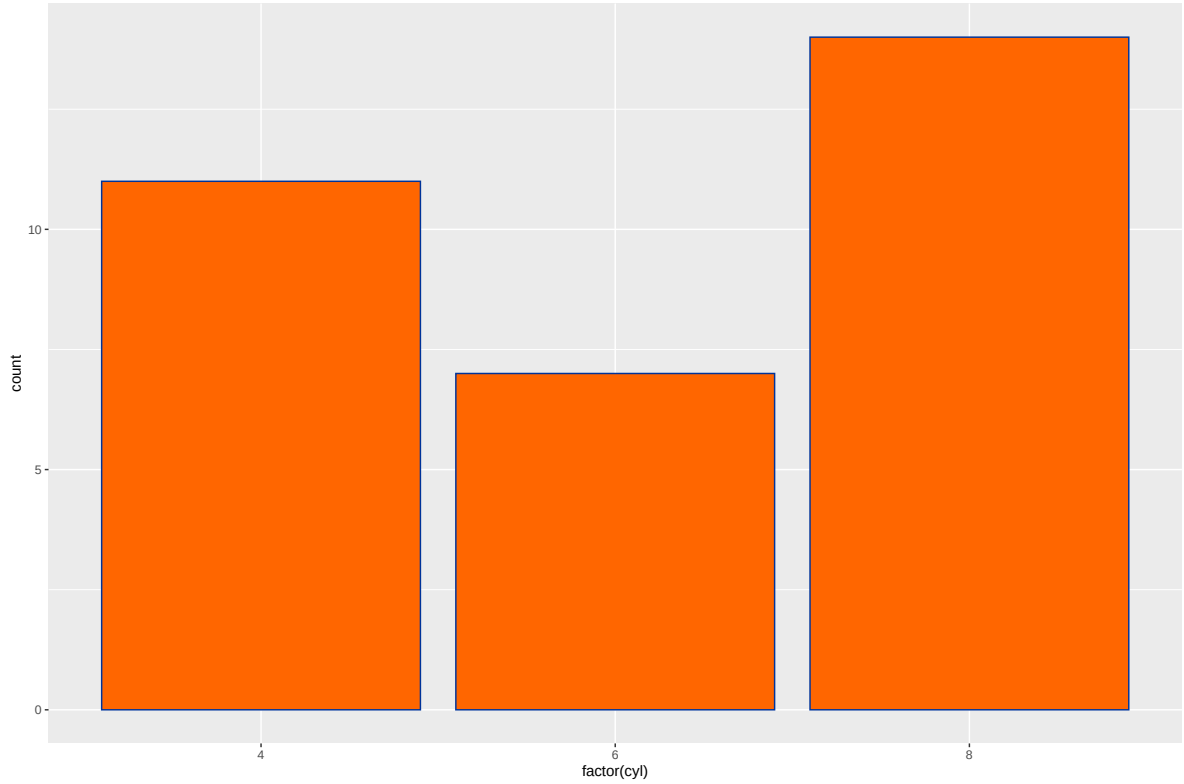
- The `geom_qq( )` requires a sample argument (instead of x)
  - positions are determined by their value

```
ggplot(data=mtcars,
    aes(sample=mpg)) +
    geom_qq()
```



- The bar-plot with `geom_bar( )` is similar to the histogram
  - shows the actual values instead of a count per bin

```
ggplot(data=mtcars,
    aes(factor(cyl))) +
    geom_bar(fill='#FF6600',color='#003399')
```

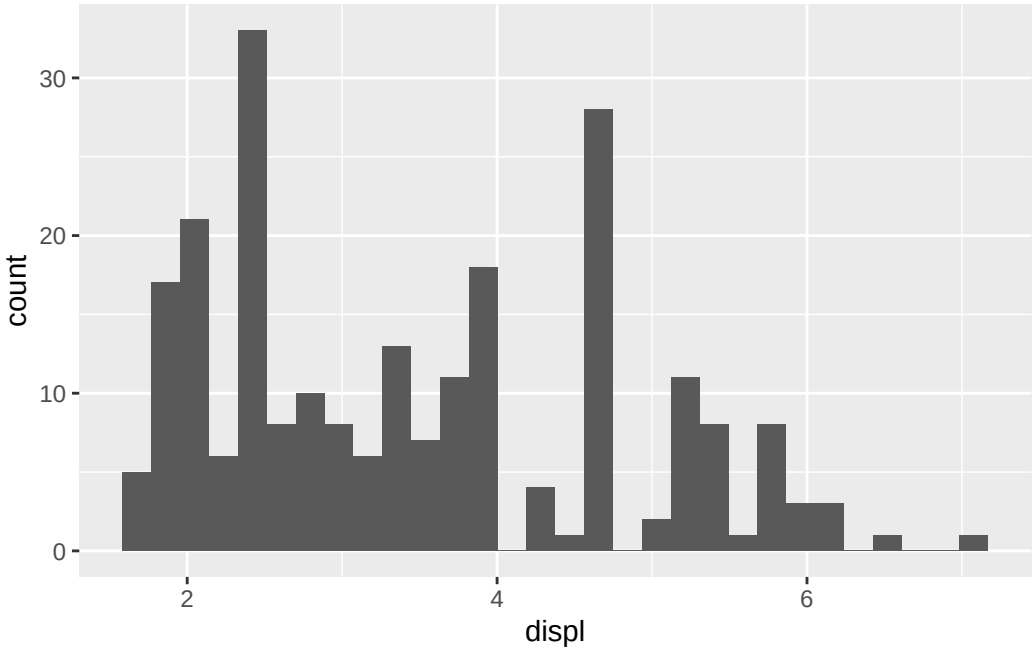


exercises on one variable visualizations

- Make use of the `mpg` dataset again
- Make a histogram for the continuously scaled `displ` variable

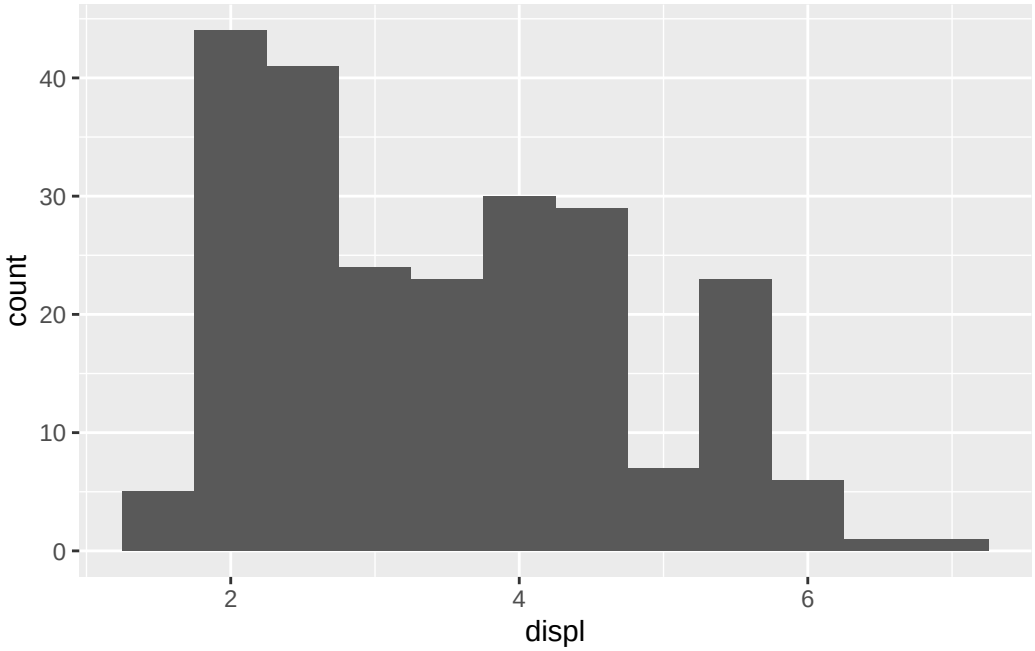
```
ggplot(data=mpg,aes(x=displ)) +
    geom_histogram()
```

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



- Adjust the binwidth to .5
- Have a glimpse at the data, what data-types are included ?

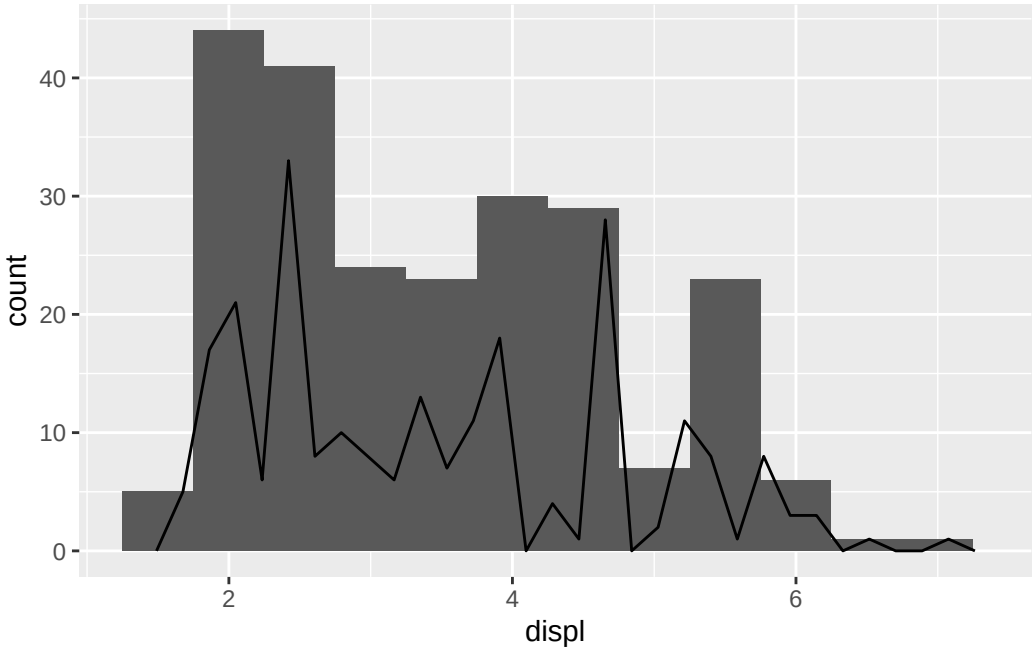
```
ggplot(data=mpg,aes(x=displ)) +
    geom_histogram(binwidth=.5)
```



- Add a frequency polynomial on top (freqpoly)

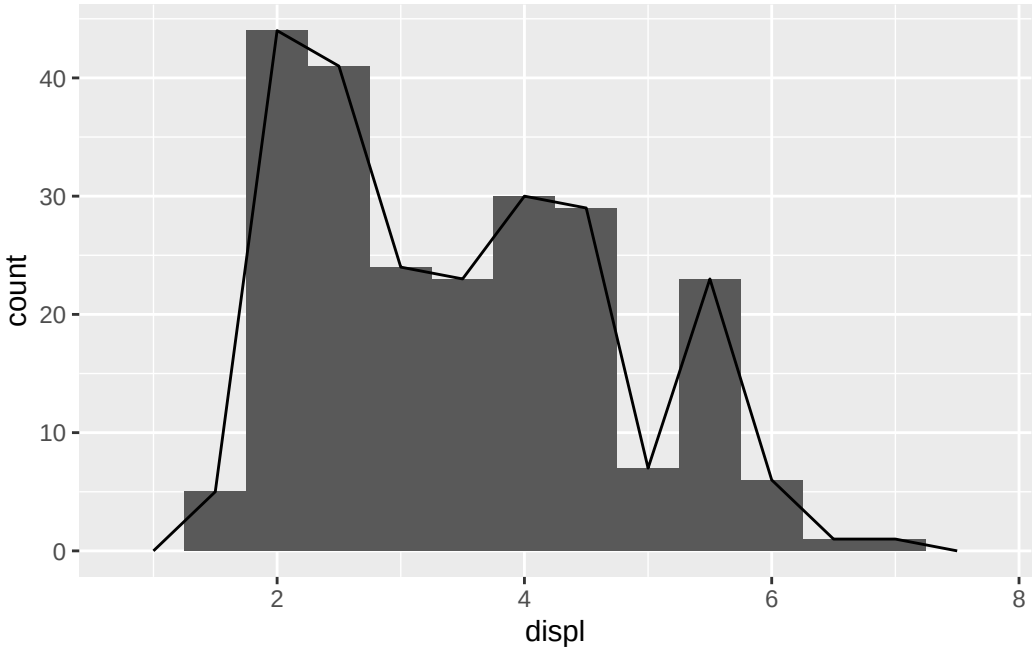
```
ggplot(data=mpg,aes(x=displ)) +
    geom_histogram(binwidth=.5) +
    geom_freqpoly()
```

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



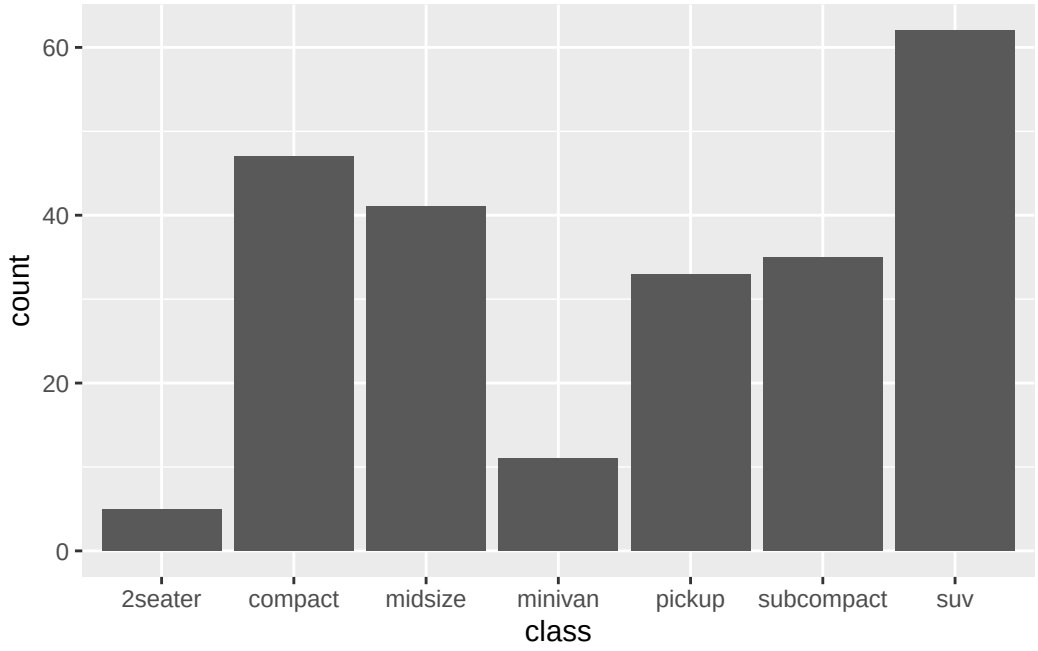
- Notice what happens if the same binwidth is used for the frequency polynomial.

```
ggplot(data=mpg,aes(x=displ)) +
    geom_histogram(binwidth=.5) +
    geom_freqpoly(binwidth=.5)
```



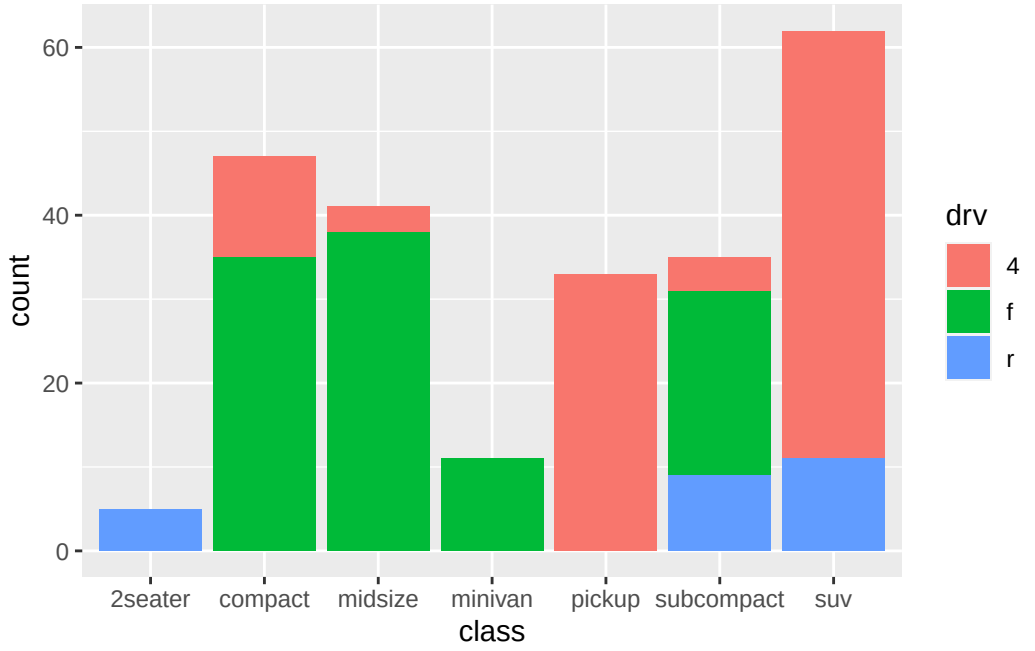
- Make a barplot for the discretely scaled `class` variable

```
ggplot(data=mpg,aes(x=class)) +
    geom_bar()
```



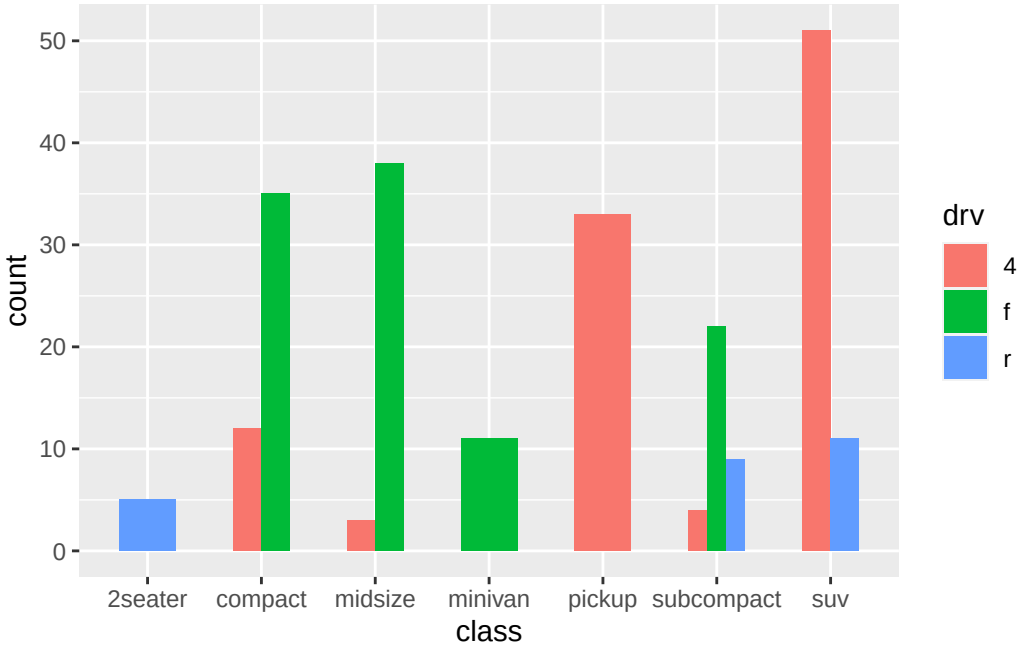
- Group the data by filling in colors dependent on the `drv` variable

```
ggplot(data=mpg,aes(x=class)) +
    geom_bar(aes(fill=drv))
```



- Turn the bars next to one-another
- Reduce their width to .5 to increase space between bars

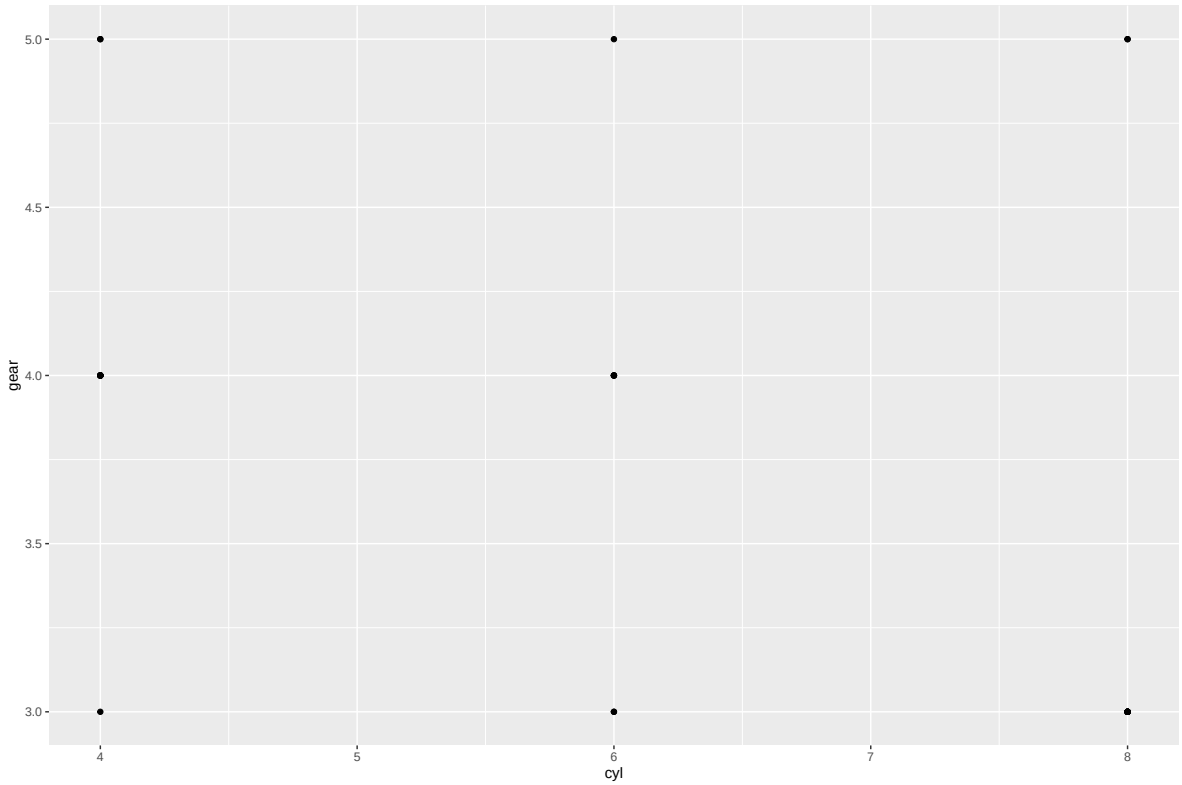
```
ggplot(data=mpg,aes(x=class)) +
    geom_bar(aes(fill=drv),
    position='dodge',width=.5)
```



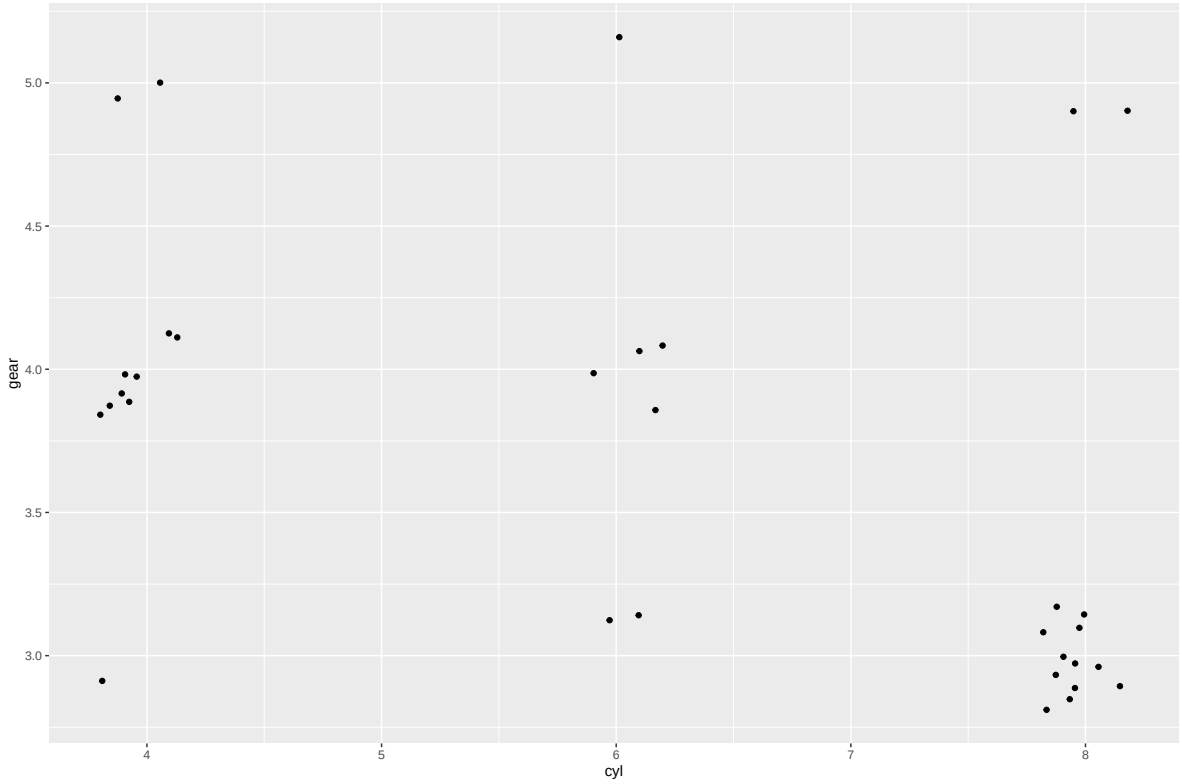
two-variables

- Various visualizations address the relation between two variables, whether discrete and/or continuous.
- Especially for categorical data data could obscure other data.
  - deal with this using the position argument or with the `geom_jitter( )`
  - avoid combining both `geom_point( )` and `geom_jitter( )` as it would draw points each time

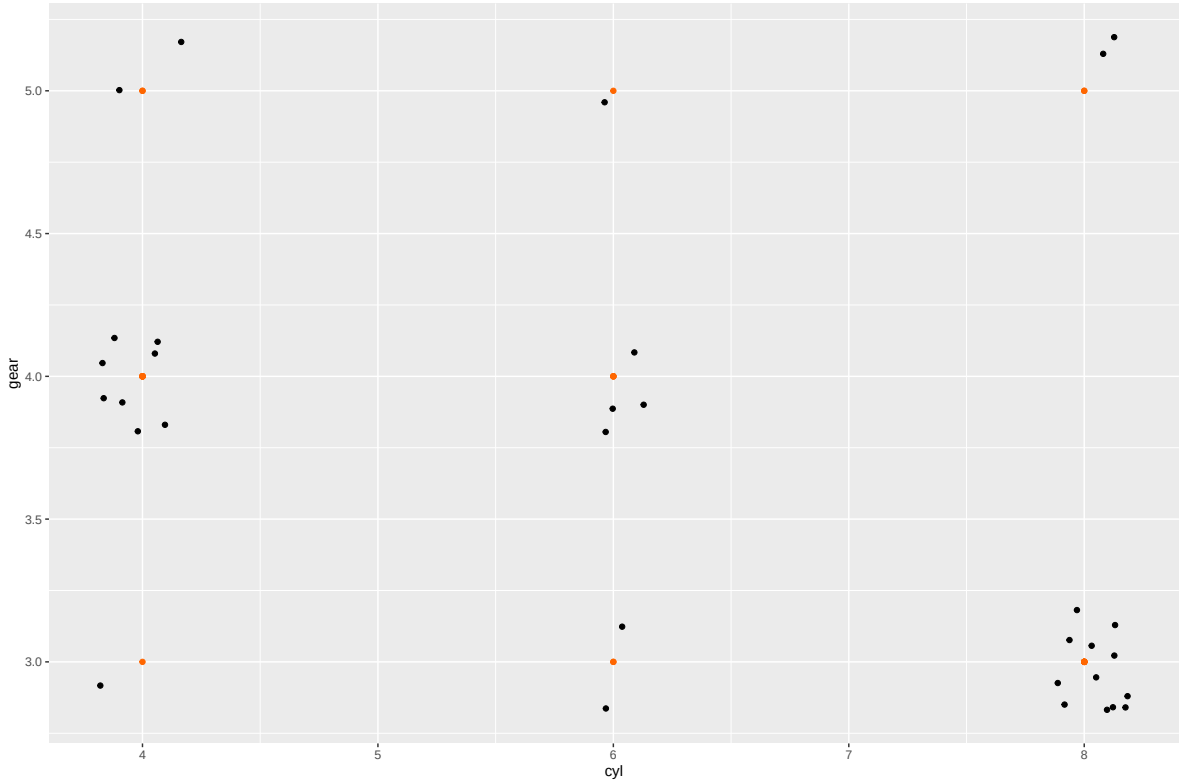
```
ggplot(data=mtcars,aes(cyl,gear)) +
geom_point()
```




```
ggplot(data=mtcars,aes(cyl,gear)) +
geom_jitter(width=.2,height=.2)
```




```
ggplot(data=mtcars,aes(cyl,gear)) +
geom_point(color='#FF6600') +
geom_jitter(width=.2,height=.2)
```

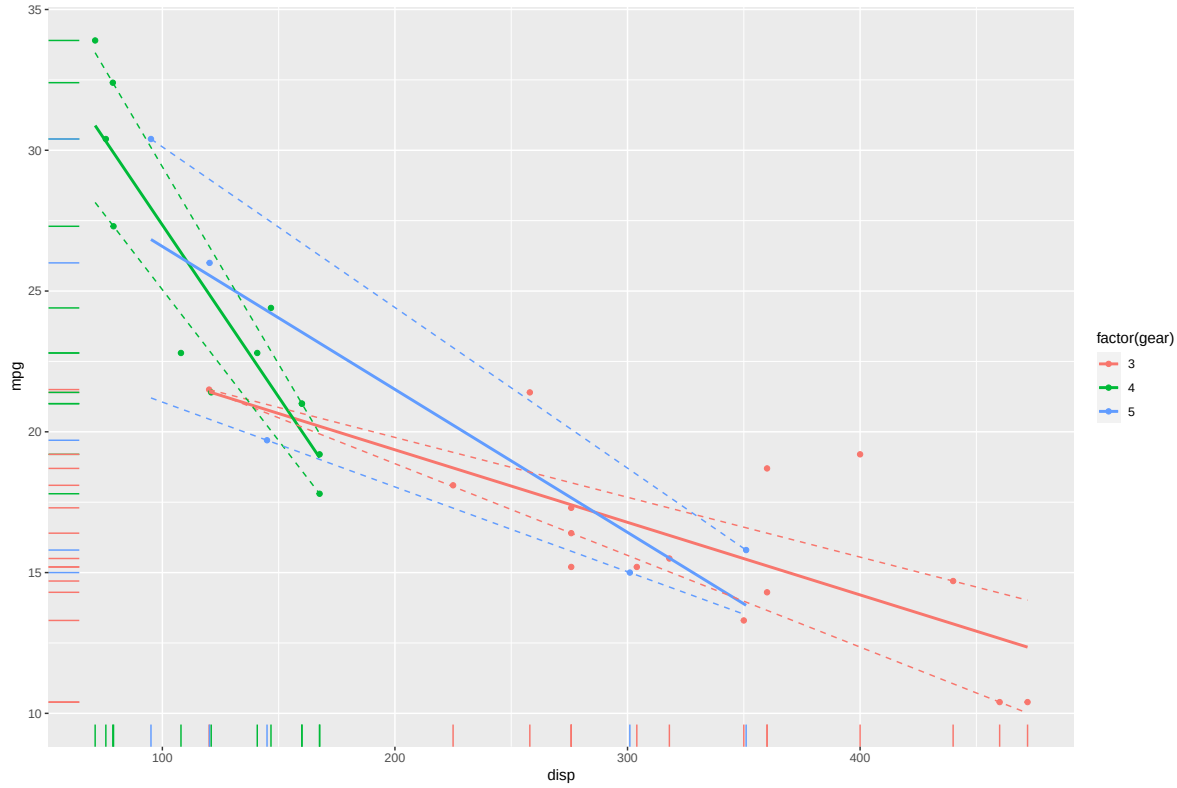


- The smooth function has been shown above
- The confidence band with the middle 50% (quantiles .25 and .75) is used
- A rug at the axes captures the one dimensional distribution.
- Instead of bullet indicators, the row names (or any other set of labels) can be used
  - use the label argument (within the `aes( )` when related to data)
  - some jitter reduces overlap

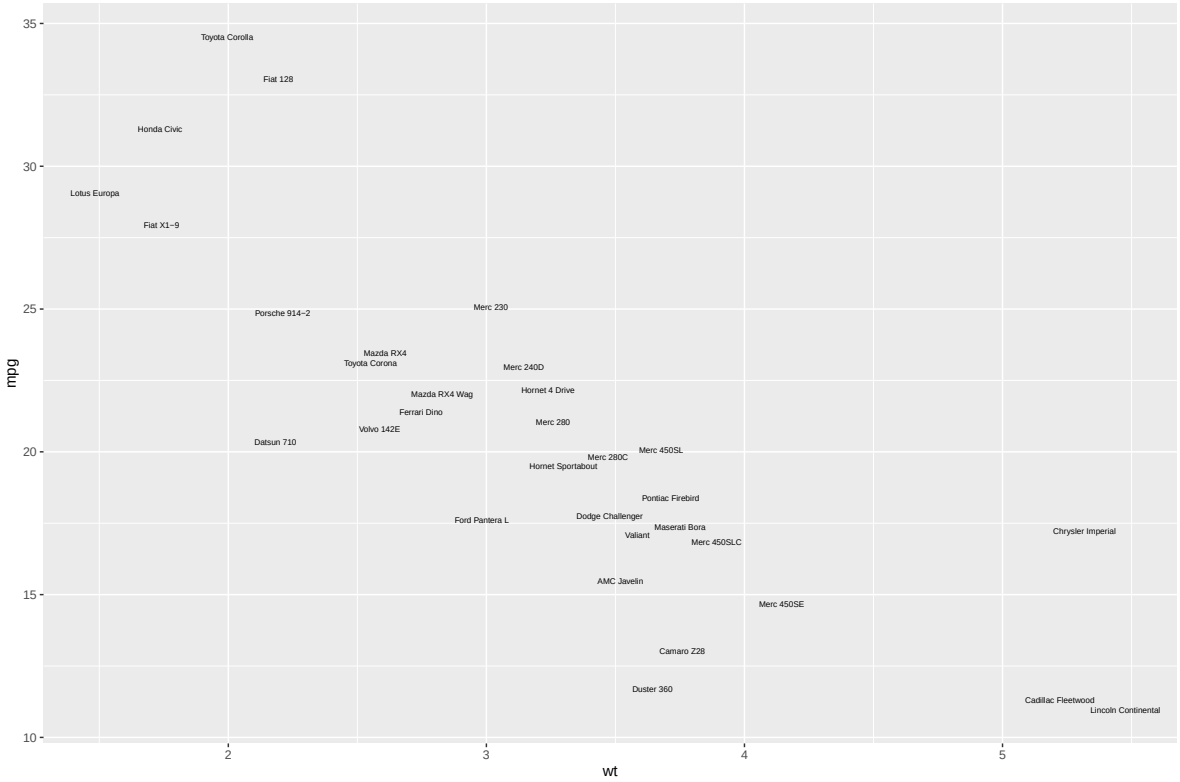
```
ggplot(data=mtcars,
    aes(y=mpg,x=disp,color=factor(gear))) +
    geom_point() +
    geom_smooth(method='lm',se=FALSE) +
    geom_rug() +
    geom_quantile(quantiles=c(.25,.75),linetype=2)
```

```
`geom_smooth()` using formula = 'y ~ x'
Smoothing formula not specified. Using: y ~ x
Smoothing formula not specified. Using: y ~ x
```

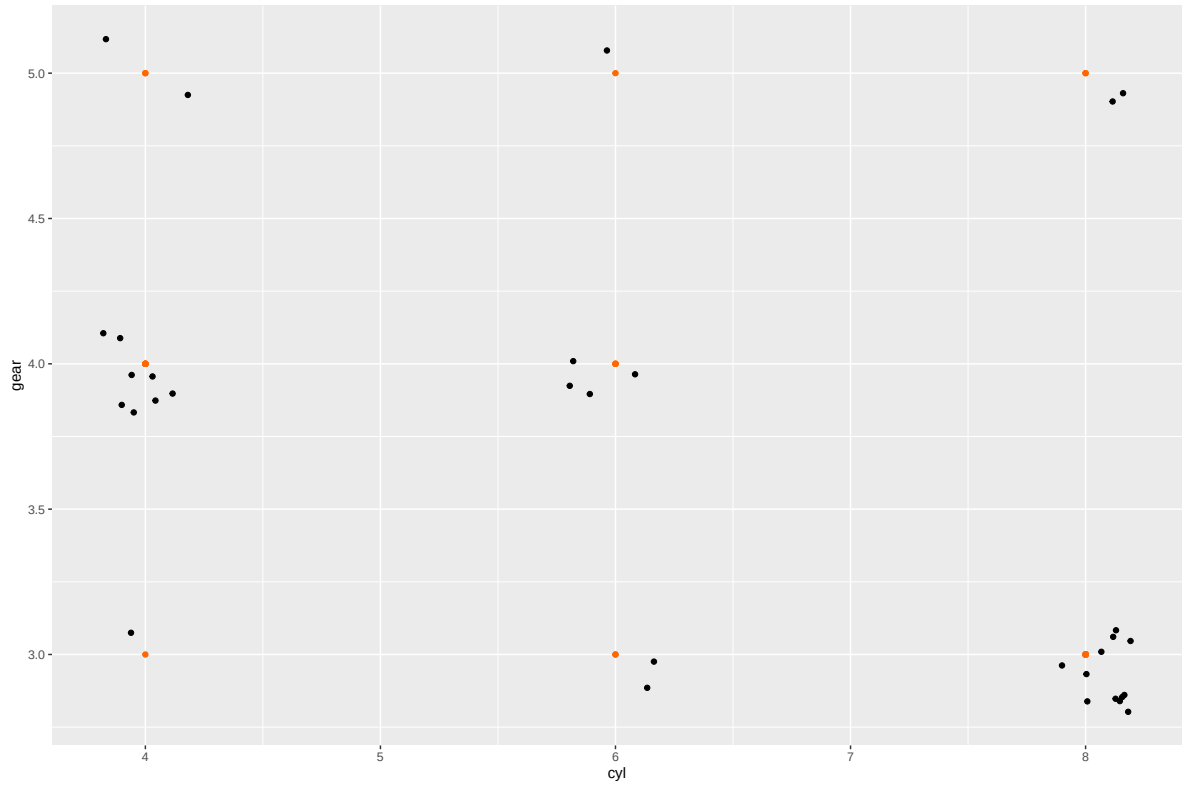
```
Smoothing formula not specified. Using: y ~ x
```




```
ggplot(mtcars, aes(wt, mpg)) +
geom_text(
    aes(label=(rownames(mtcars))),
    size=2,
    position=position_jitter(width = .2,height=3, seed=256)
)
```




```
ggplot(data=mtcars,aes(cyl,gear)) +
geom_point(color='#FF6600') +
geom_jitter(width=.2,height=.2)
```

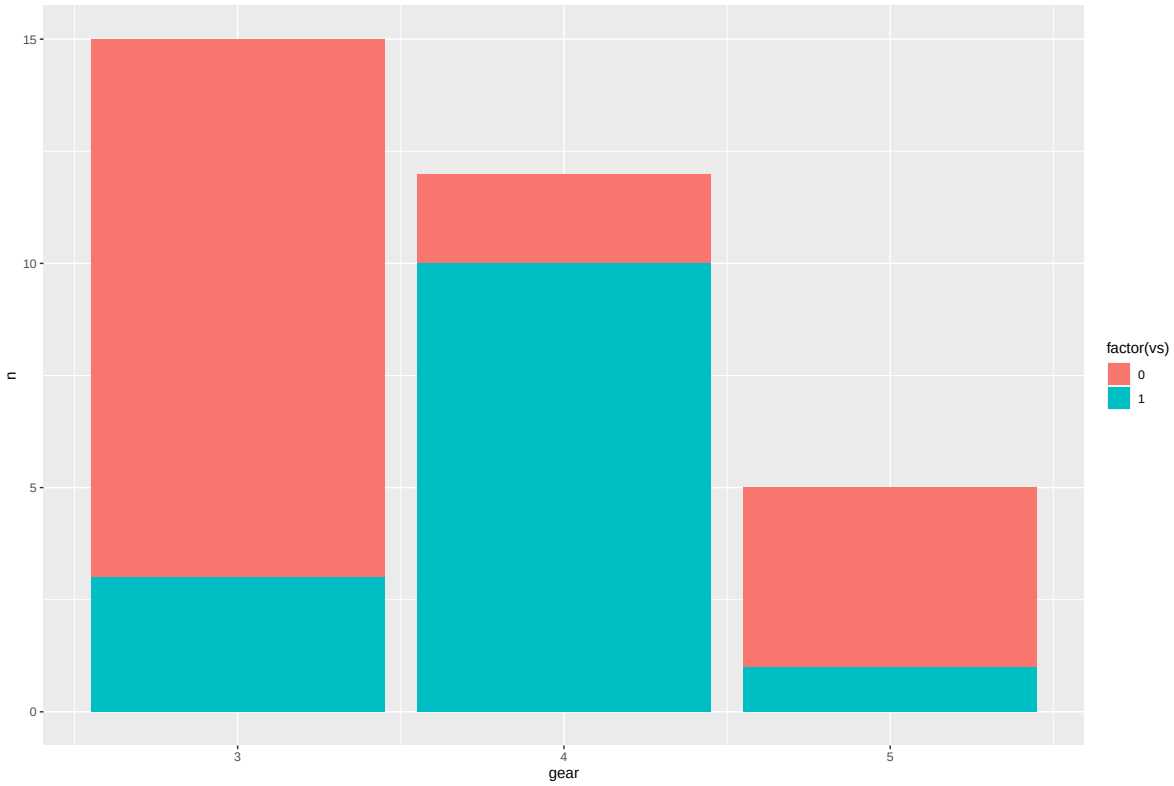


- Bars can be obtained with `geom_col( )`, or with `geom_bar(stat='identity')`
  - use of 'identity' causes the height to depend on the numbers in the data
  - Note: these numbers are summed if not unique, be careful.
- A `count( )` extracts the frequency of the specified grouping

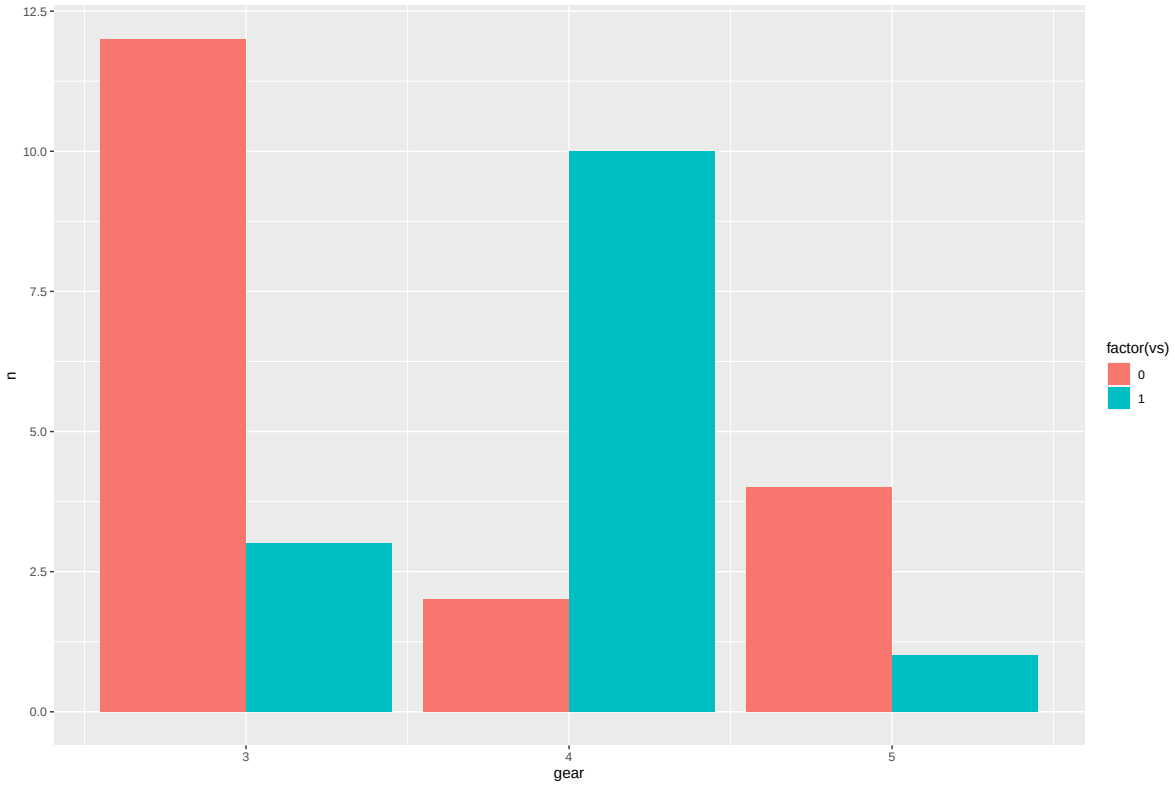
```
(freq_by_group <- mtcars %>% group_by(gear,vs) %>% count())
```

```
# A tibble: 6 x 3
# Groups:   gear, vs [6]
   gear    vs     n
  <dbl> <dbl> <int>
1     3     0    12
2     3     1     3
3     4     0     2
4     4     1    10
5     5     0     4
6     5     1     1
```

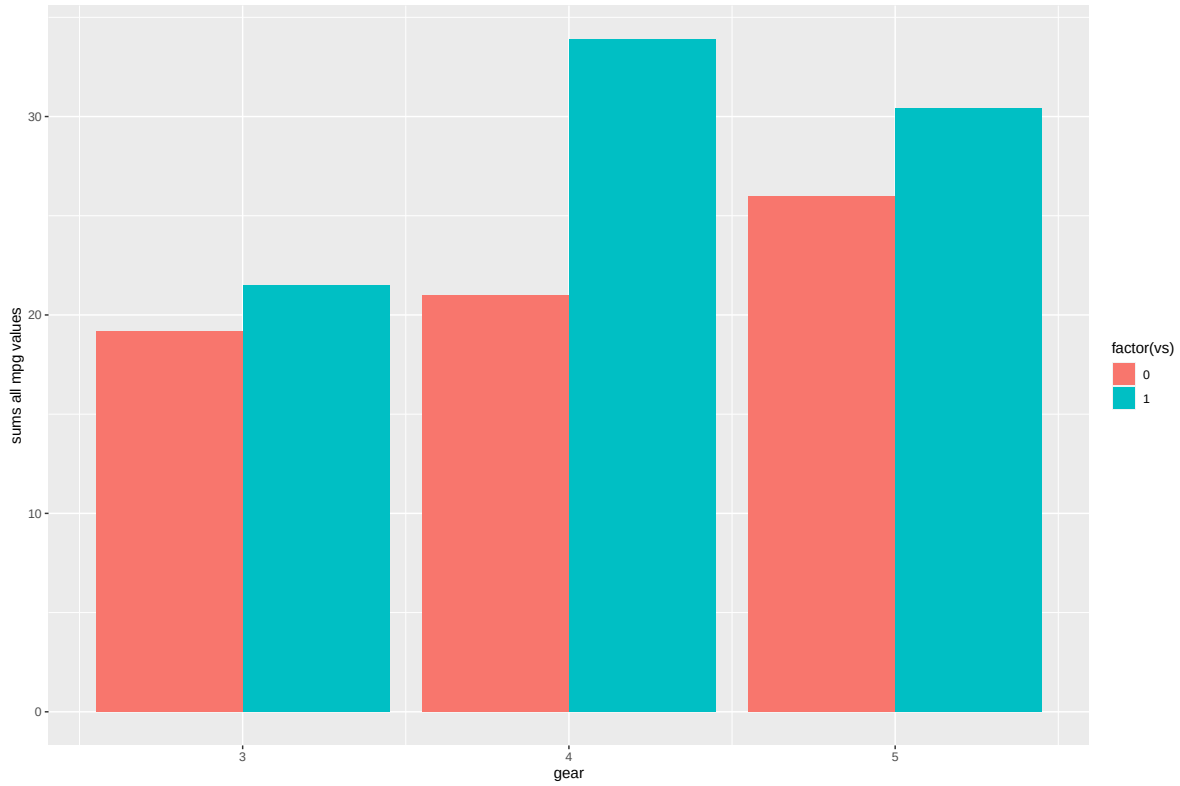
```
ggplot(freq_by_group,
    aes(fill=factor(vs), y=n, x=gear)) +
geom_bar(position="stack", stat="identity")
```




```
ggplot(freq_by_group,
    aes(fill=factor(vs), y=n, x=gear)) +
geom_col(position="dodge")
```

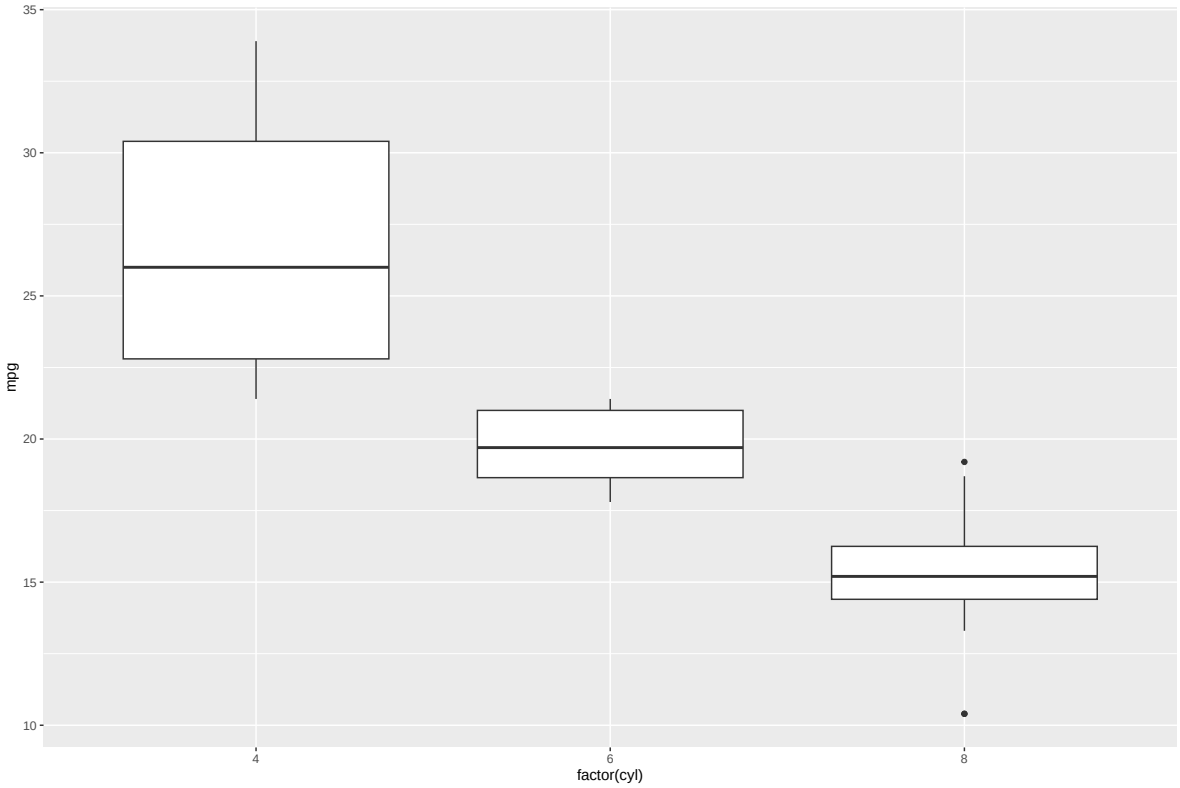



```
ggplot(mtcars,
    aes(fill=factor(vs), y=mpg, x=gear)) +
geom_col(position="dodge") +
labs(y='sums all mpg values')
```

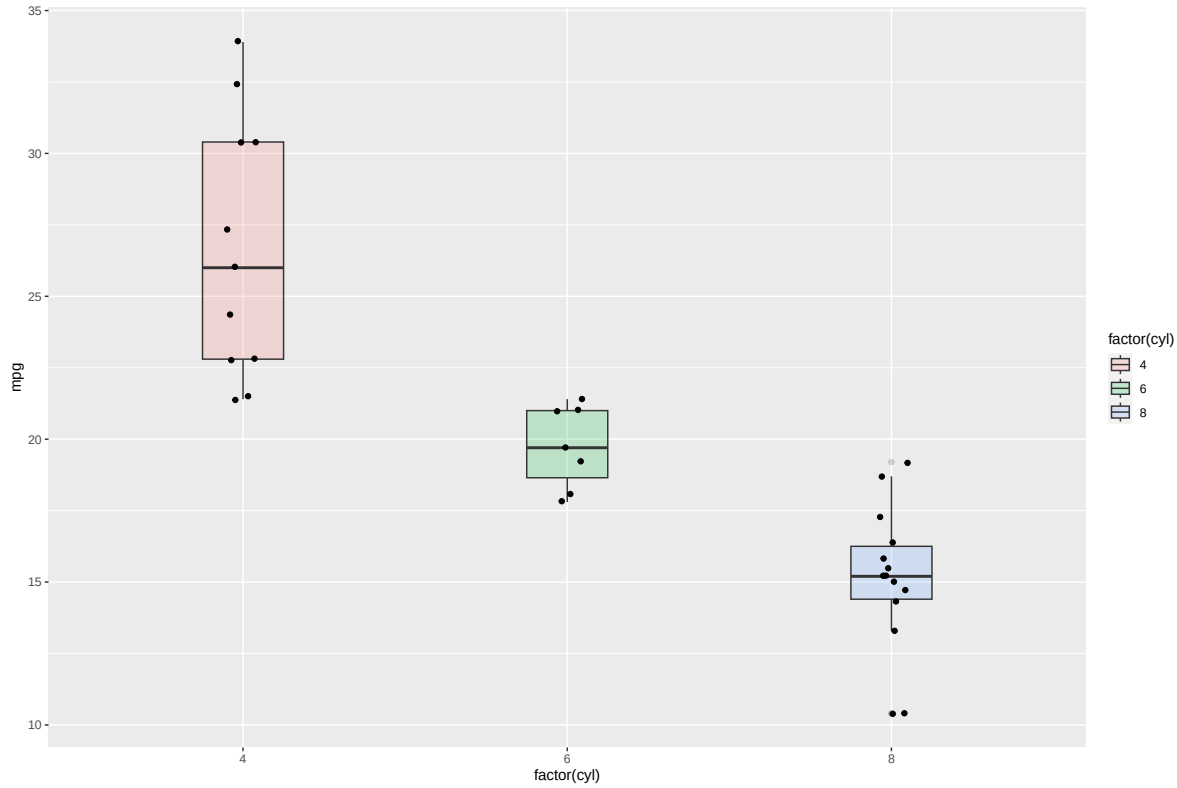


- A boxplot nicely summarizes continuous data, possibly for different groups

```
ggplot(data=mtcars,
    aes(y=mpg,x=factor(cyl))) +
geom_boxplot()
```



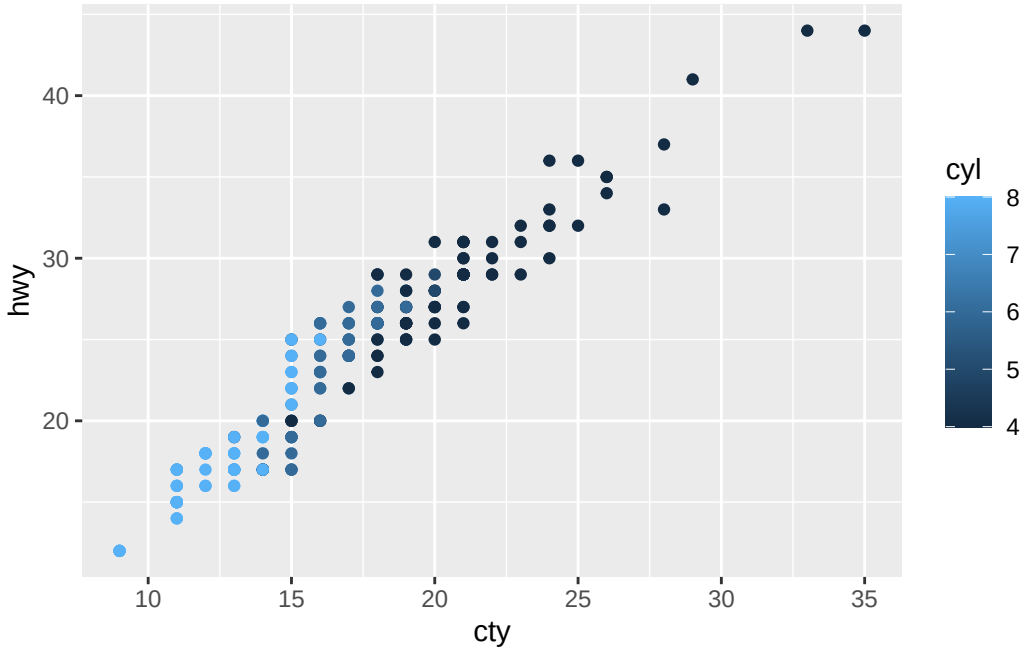

```
ggplot(data=mtcars,
    aes(y=mpg,x=factor(cyl))) +
    geom_boxplot(width=.25,alpha=.2,
        aes(fill=factor(cyl))
    ) +
    geom_jitter(width=.05)
```



exercises on two variable visualizations

- Make a scatterplot for the continuously scaled `hwy` on `cty`, and color by `cyl`

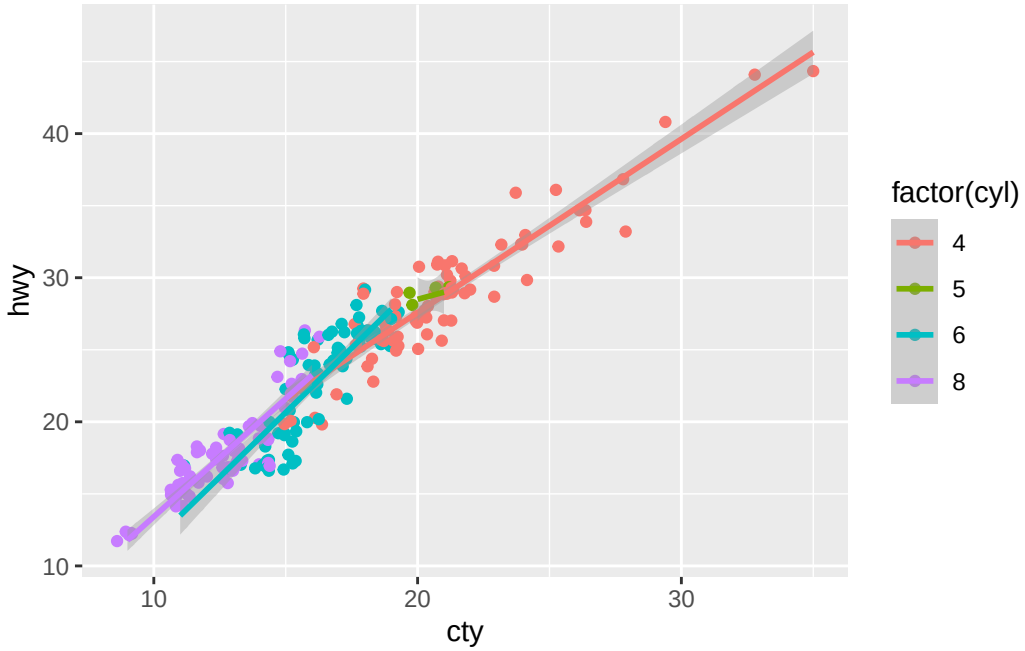
```
ggplot(data=mpg,aes(y=hwy,x=cty,color=cyl)) +
    geom_point()
```



- Jitter the data, so it shows if data points obscure one-another
- Make sure that `cyl` is categorical

```
ggplot(data=mpg,
    aes(y=hwy,x=cty,color=factor(cyl))) +
geom_jitter() +
geom_smooth(method='lm')
```

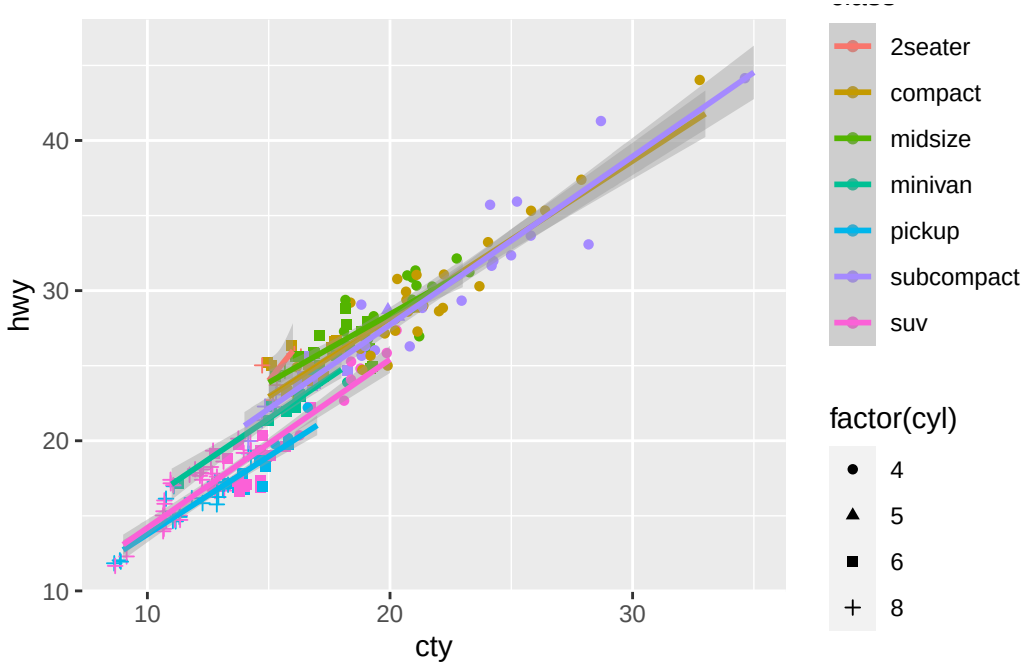
```
`geom_smooth()` using formula = 'y ~ x'
```



- Add a conditional average with `geom_smooth( )`, use the `lm` method

```
ggplot(data=mpg,
    aes(y=hwy,x=cty,color=class)) +
geom_jitter(aes(shape=factor(cyl))) +
geom_smooth(method='lm')
```

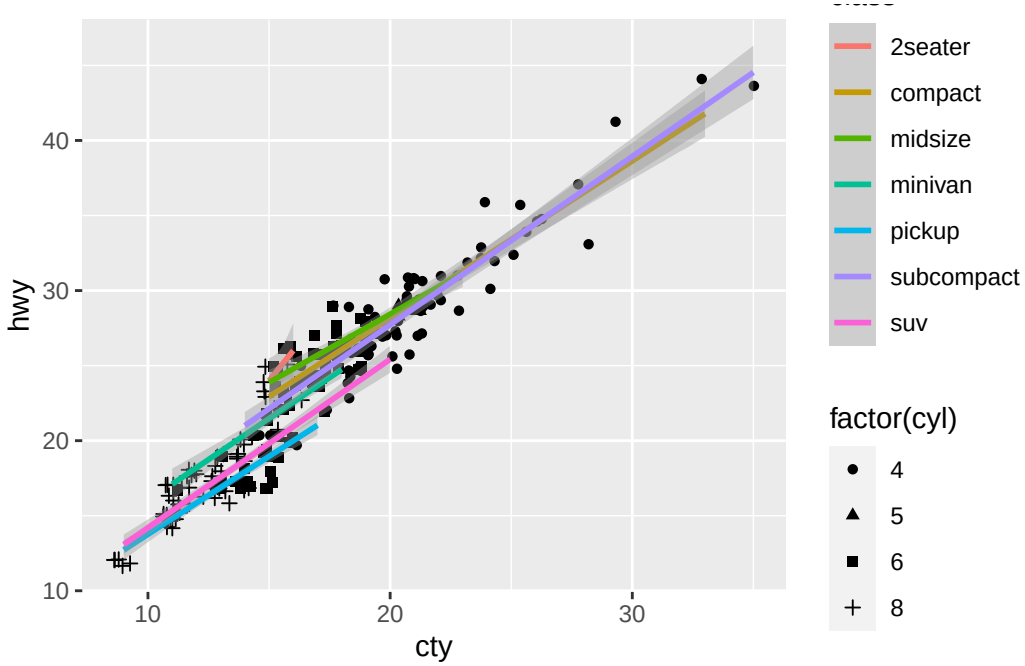
```
`geom_smooth()` using formula = 'y ~ x'
```



- Add a shape dependent on `class`, notice the restriction on the number of shapes

```
ggplot(data=mpg,aes(y=hwy,x=cty)) +
    geom_jitter(
        aes(shape=factor(cyl))
    ) +
    geom_smooth(method='lm',aes(color=class))
```

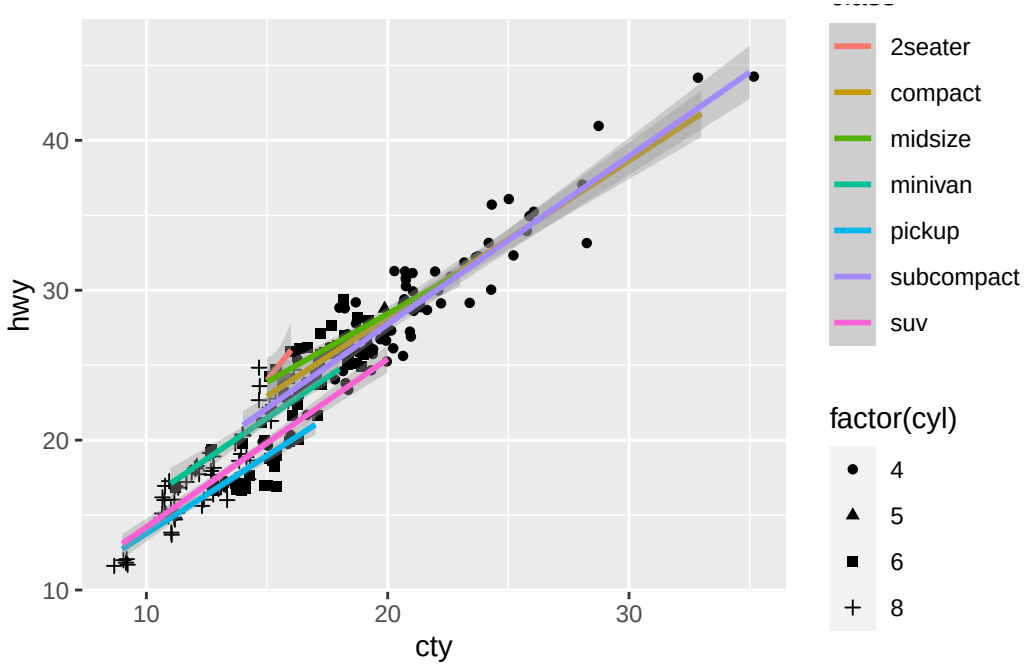
```
`geom_smooth()` using formula = 'y ~ x'
```



- Switch the color and shapes around (color can have more than 6 if it is really necessary)

```
ggplot(data=mpg,aes(y=hwy,x=cty)) +
    geom_jitter(aes(shape=factor(cyl))) + geom_smooth(method='lm',aes(color=class))
```

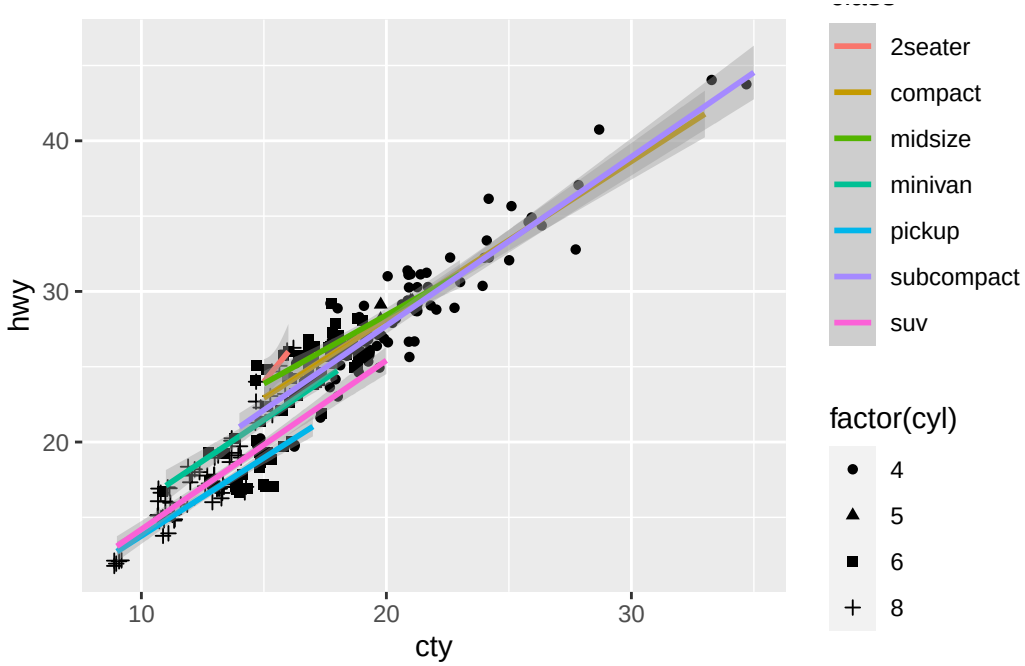
```
`geom_smooth()` using formula = 'y ~ x'
```



- Make sure the symbols do not differ by color (all black), only shape, but keep the regression lines

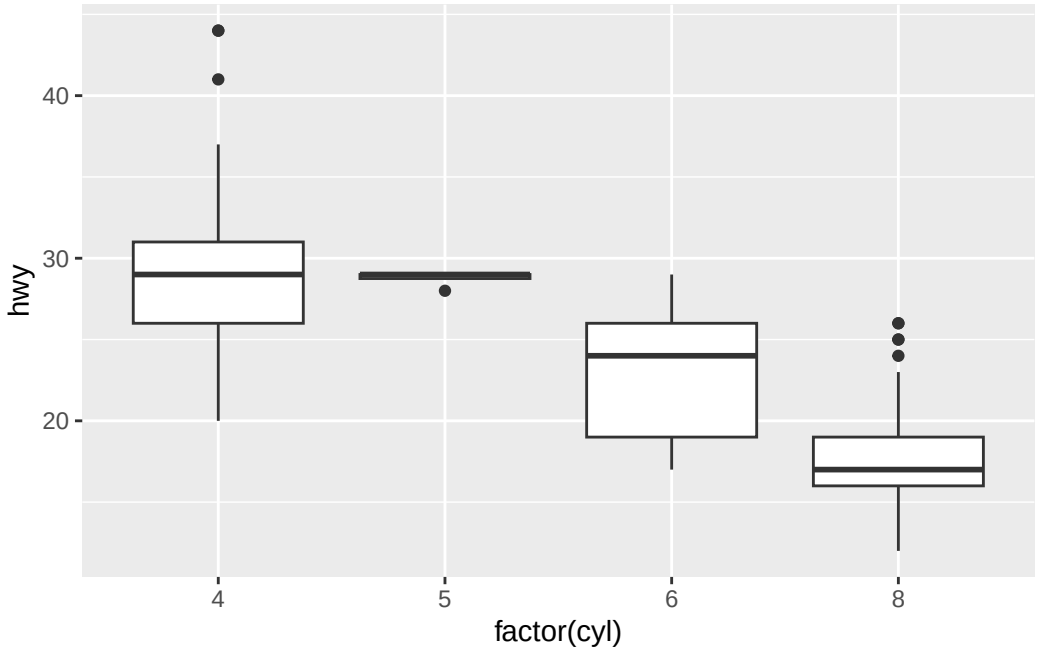
```
ggplot(data=mpg,aes(y=hwy,x=cty)) +
    geom_jitter(aes(shape=factor(cyl))) + geom_smooth(method='lm',aes(color=class))
```

```
`geom_smooth()` using formula = 'y ~ x'
```



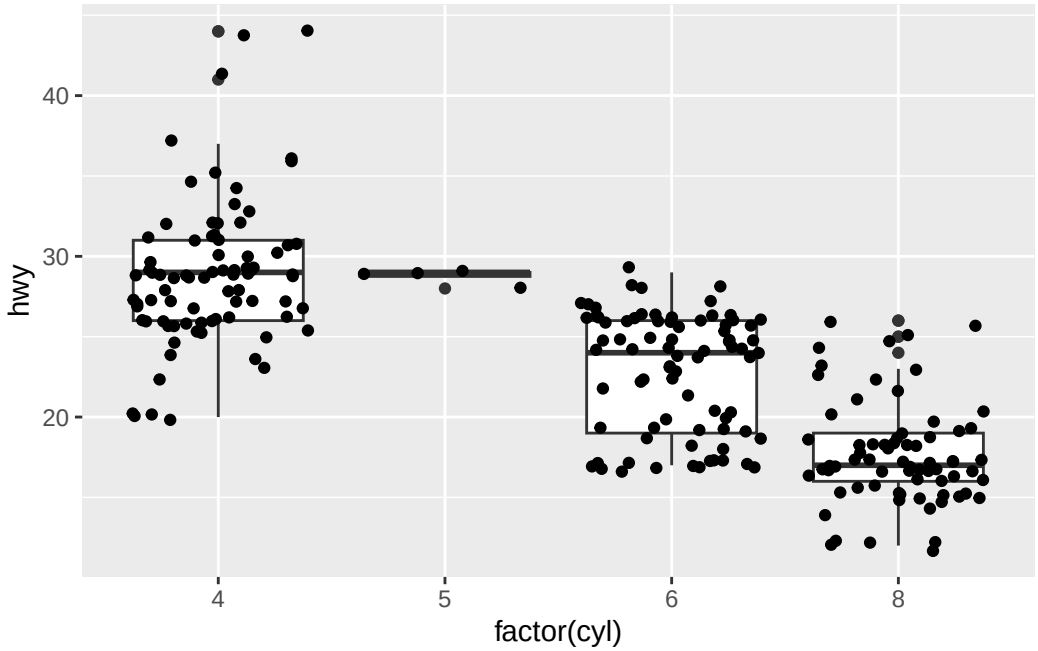
- Show boxplots for the `hwy` for each `cyl`

```
ggplot(data=mpg,
    aes(y=hwy,x=factor(cyl))
) +
geom_boxplot()
```



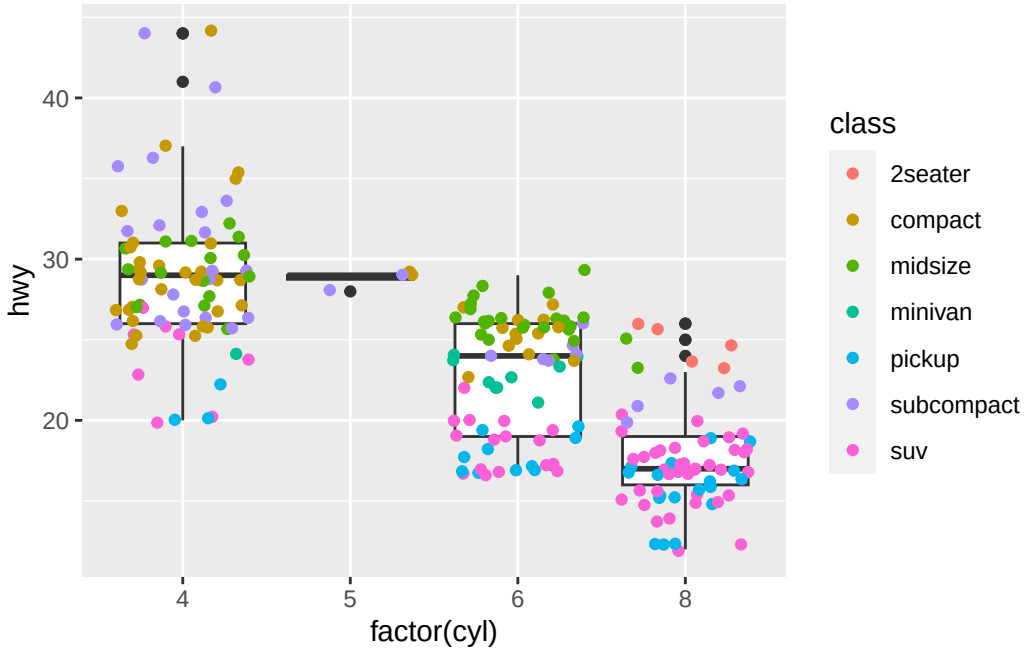
- Add the actual data points and make sure they do not obscure each other

```
ggplot(data=mpg,
    aes(y=hwy,x=factor(cyl))
) +
geom_boxplot() +
geom_jitter()
```



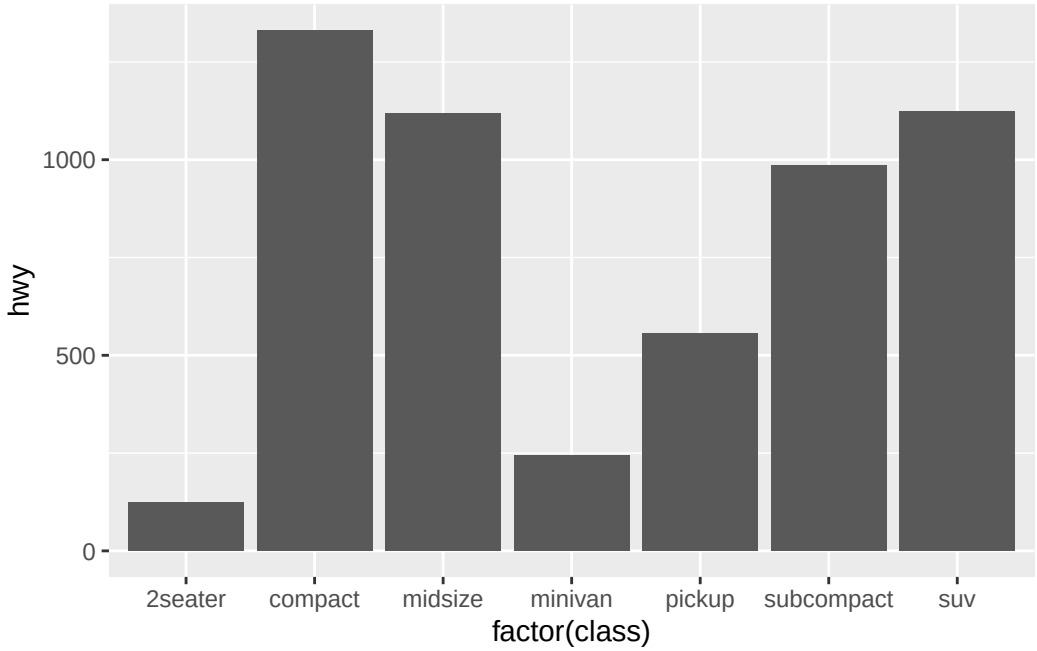
- Give a color to the observations dependent on `class`

```
ggplot(data=mpg,
    aes(y=hwy,x=factor(cyl))
) +
geom_boxplot() +
geom_jitter(aes(color=class))
```



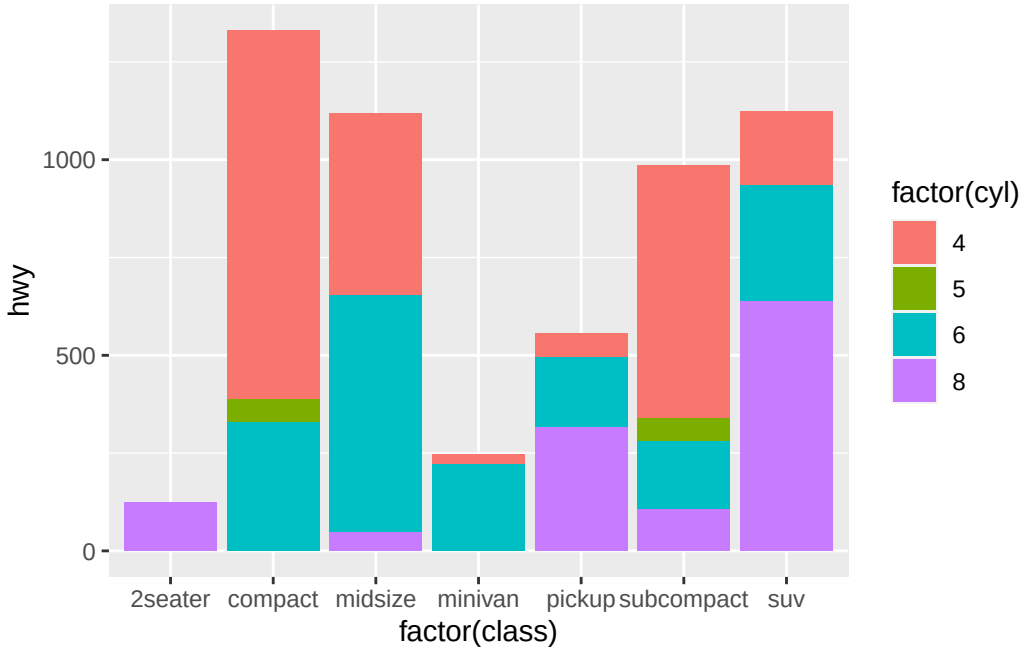
- Make a bar chart that adds all `hwy` values in each `class` group

```
ggplot(data=mpg,
    aes(y=hwy,x=factor(class))
) +
geom_col()
```



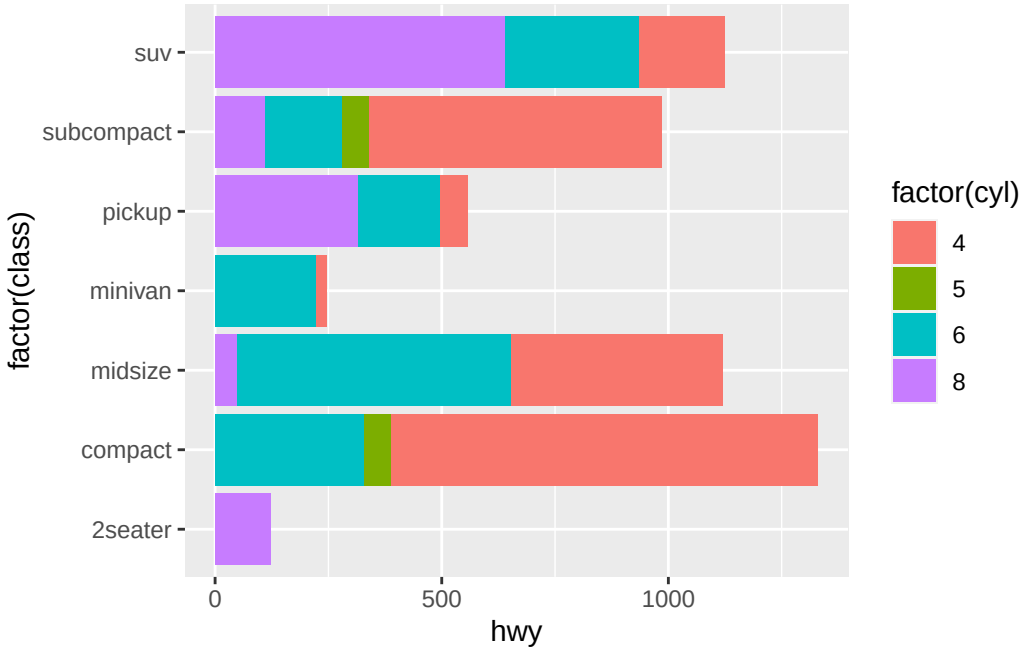
- Use a coloring of the bars to signal the relative contribution of all `cyl` categories

```
ggplot(data=mpg,
    aes(y=hwy,x=factor(class),
    fill=factor(cyl))
) +
geom_col()
```



- Flip the coordinates x and y axis

```
ggplot(data=mpg,
    aes(y=hwy,x=factor(class),
    fill=factor(cyl))
) +
geom_col() +
coord_flip()
```



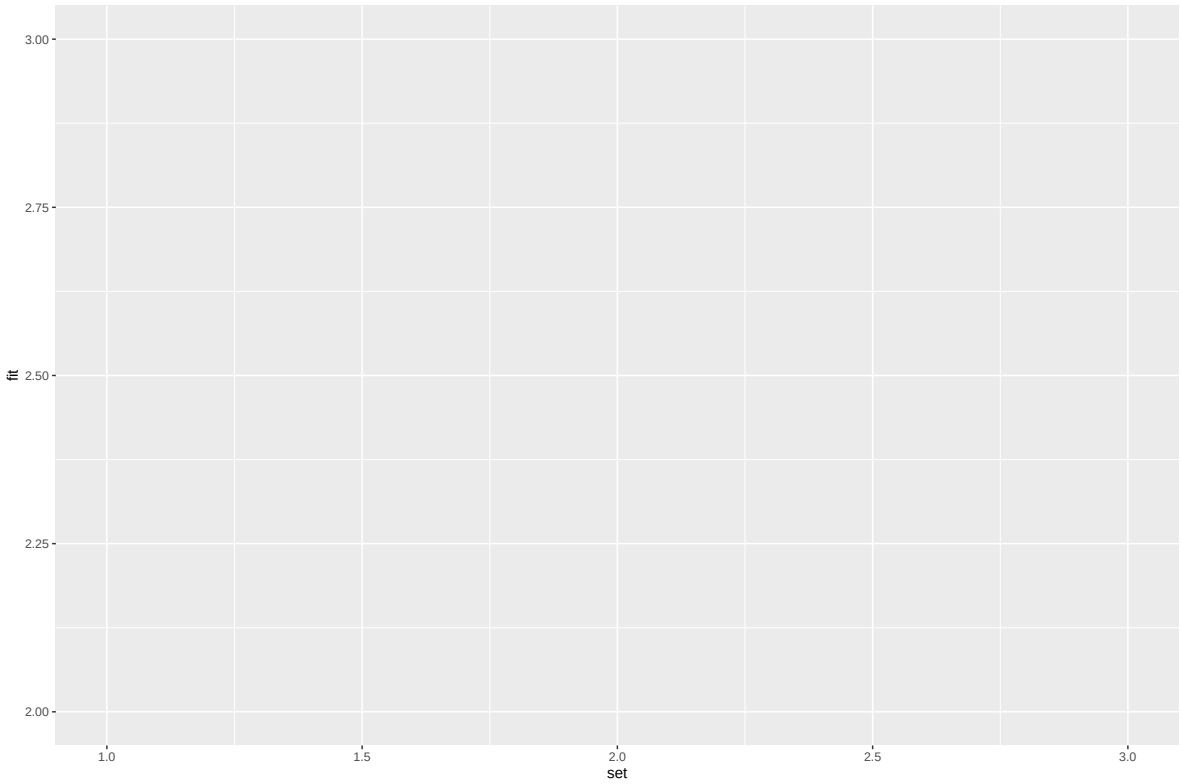
intervals

- Specialized functions facilitate visualization of errors / confidence intervals
- Standard errors or other intervals can be visualized along with fitted values

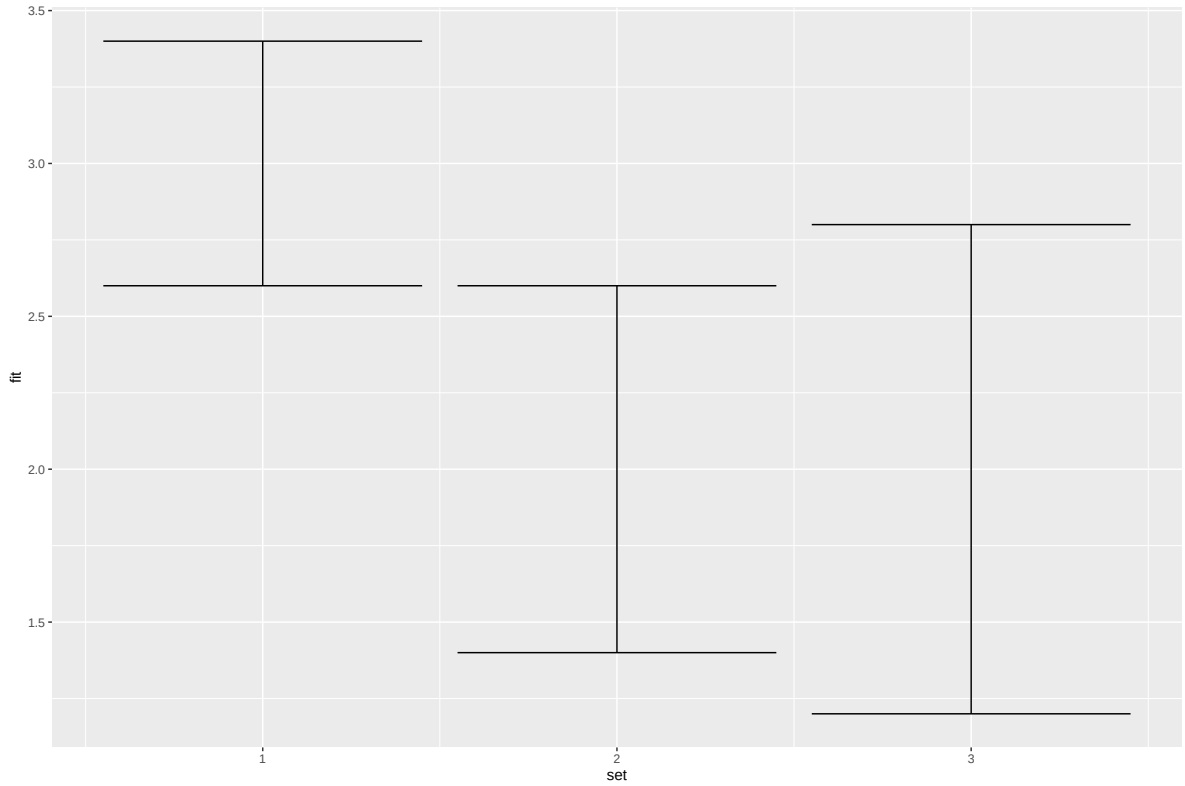
```
(tmp <- tribble(
~set,~fit,~se,
1,3,.2,
2,2,.3,
3,2,.4))
```

```
# A tibble: 3 x 3
    set   fit    se
  <dbl> <dbl> <dbl>
1     1     3   0.2
2     2     2   0.3
3     3     2   0.4
```

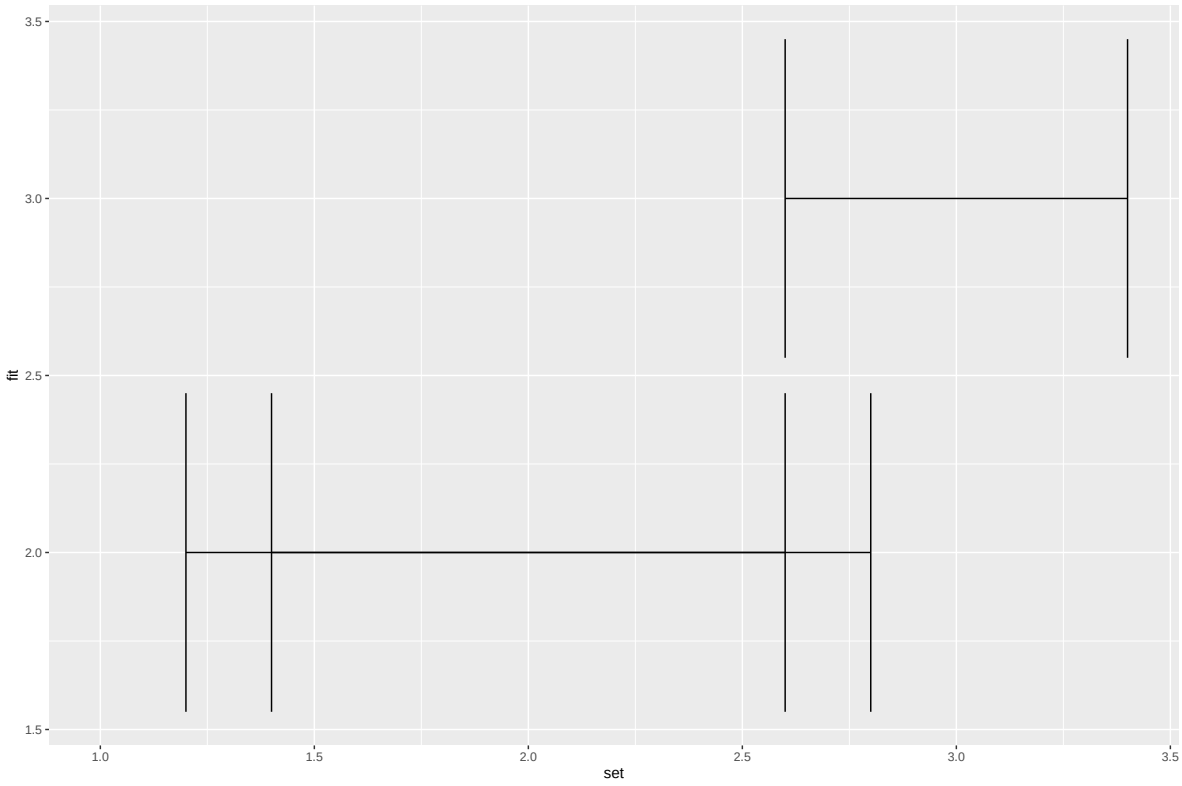
```
(myplot <- ggplot(data=tmp,
    aes(y=fit,x=set)))
```



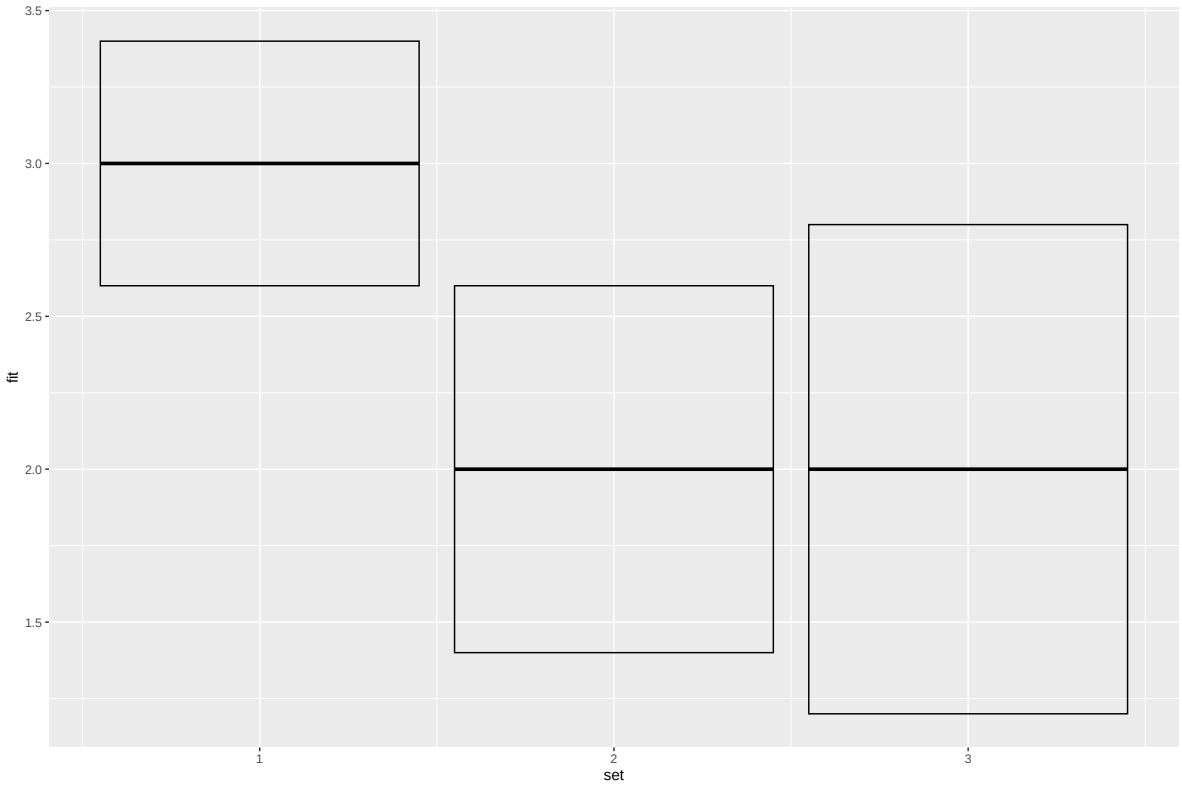

```
myplot + geom_errorbar(
    aes(ymax=fit+2*se,ymin=fit-2*se)
)
```



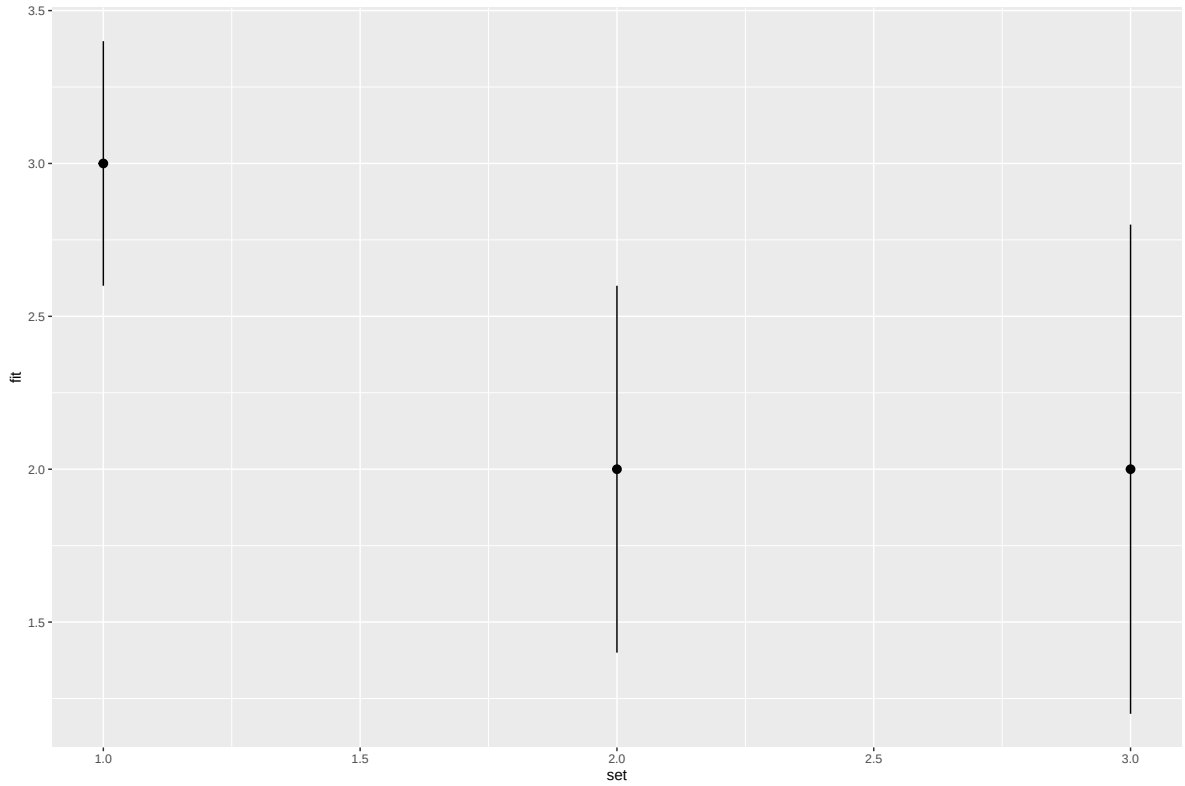

```
myplot + geom_errorbarh(
    aes(xmax=fit+2*se,xmin=fit-2*se)
)
```




```
myplot + geom_crossbar(
    aes(ymax=fit+2*se,ymin=fit-2*se)
)
```



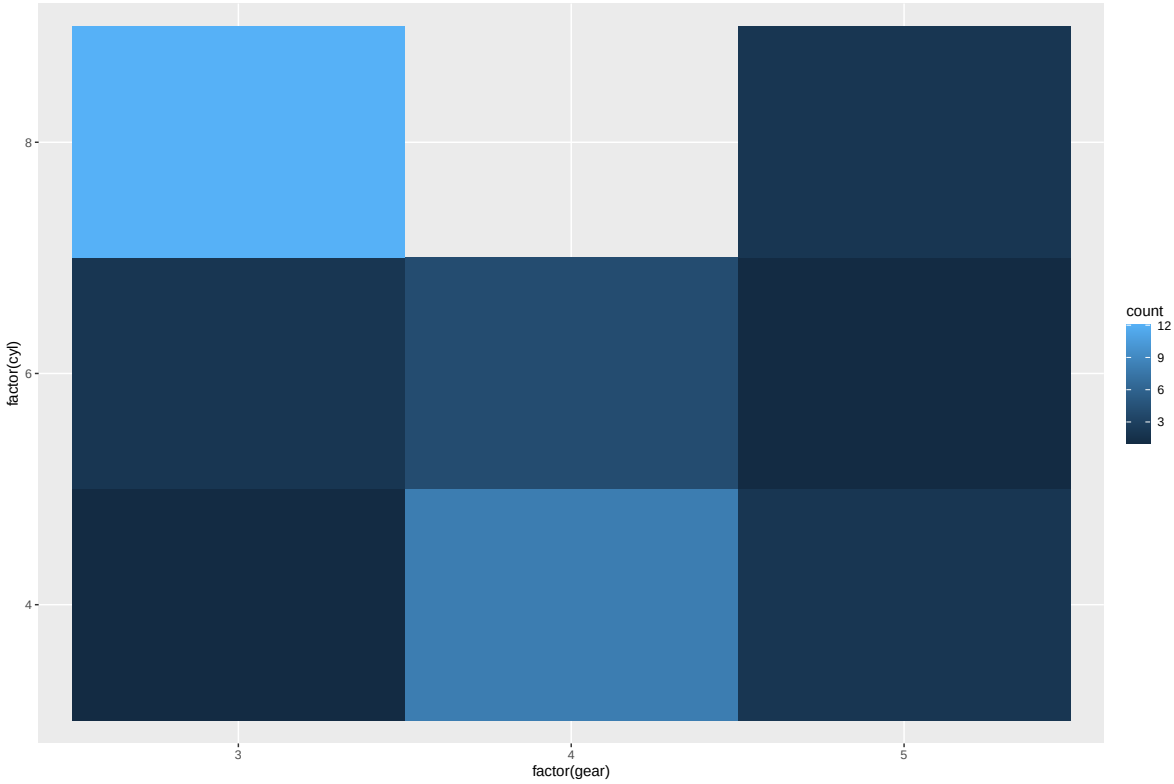

```
myplot + geom_pointrange(
    aes(ymax=fit+2*se,ymin=fit-2*se)
)
```



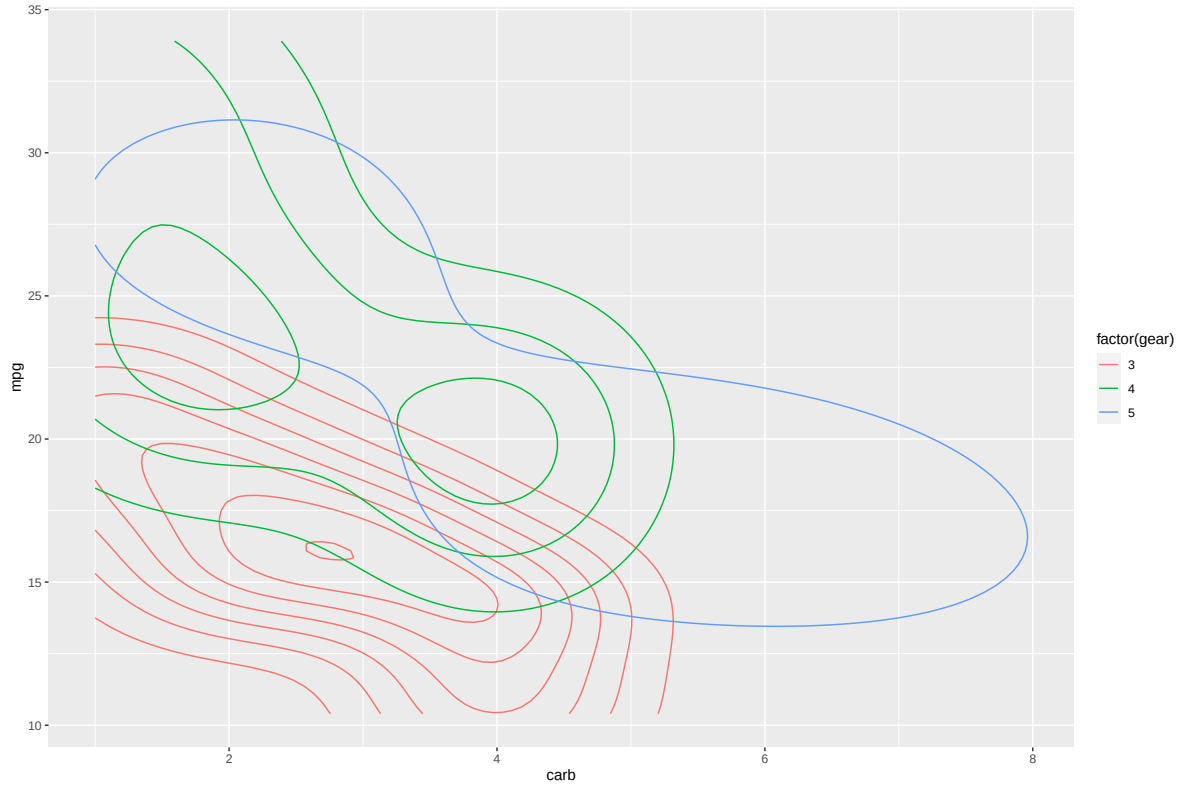
third variable implied

- Frequencies and densities can be obtained for two variables
- With `geom_bin2d( )` the frequency of combinations is obtained
  - in this case for the factors cyl and gear
- Continuous variables can be used with binning
- Contours can be obtained with `geom_density2d( )`

```
ggplot(data=mtcars,
    aes(y=factor(cyl),x=factor(gear))
) +
geom_bin2d()
```




```
ggplot(data=mtcars,
    aes(y=mpg,x=carb)
) + geom_density2d(
    aes(colour = factor(gear))
)
```



three variables

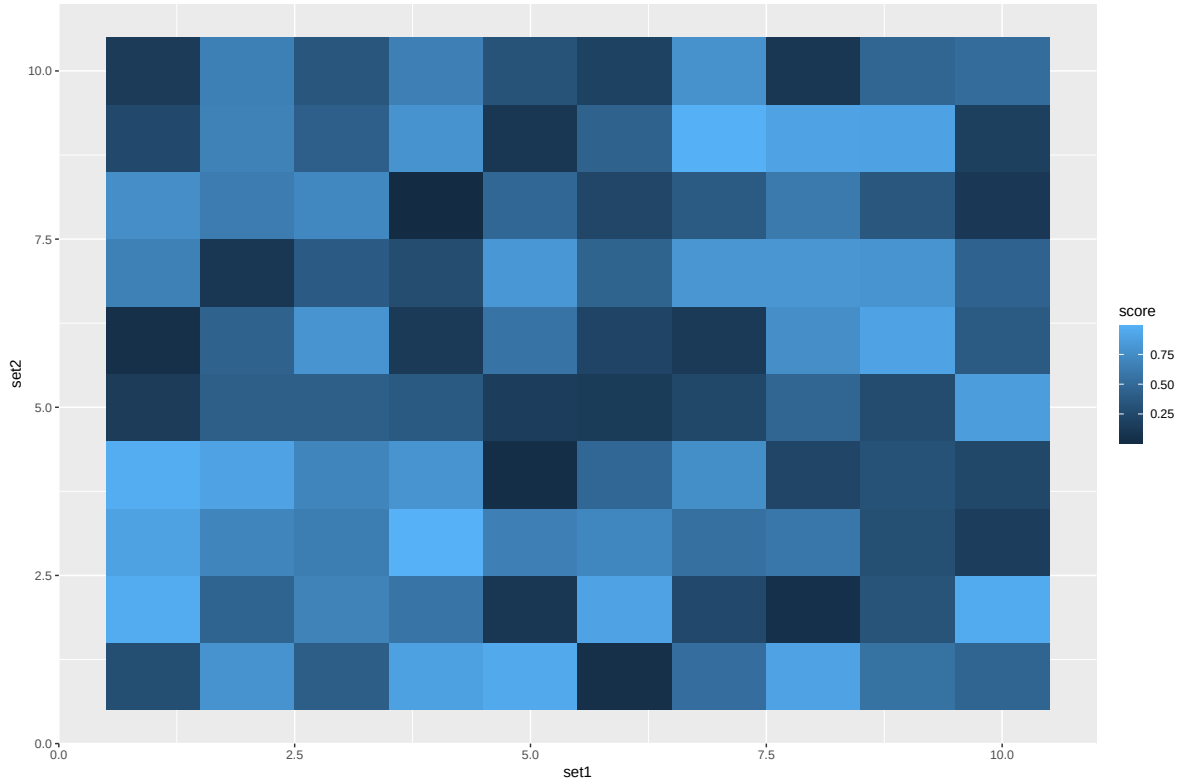
- A z dimension is possible, while typically other aesthetics are used
- A heatmap is the most obvious use
  - to show for example correlations between many variables with colors instead of values
  - a small example is used instead
    - notice the argument for tile is a fill
    - z is the argument for the contour

```
tmp <- expand.grid(set1=1:10,set2=1:10);
set.seed(123);
tmp$score <- runif(100,0,1)
head(tmp)
```

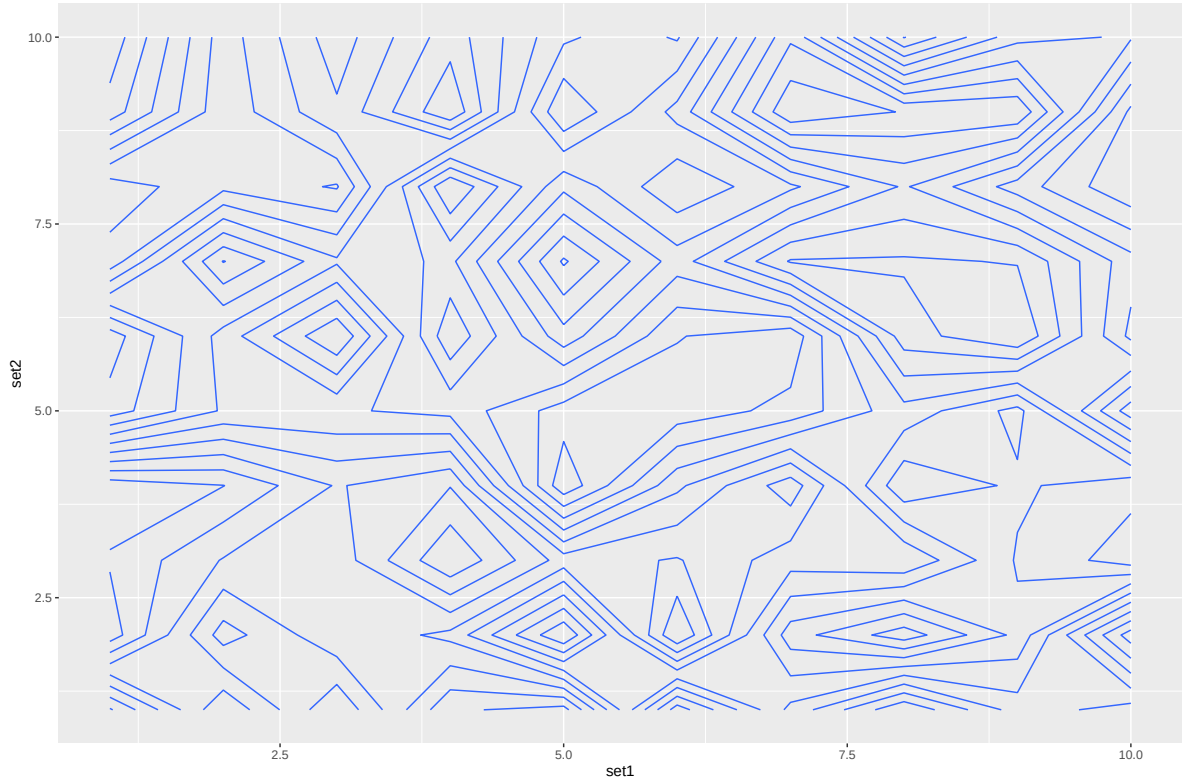
```
  set1 set2     score
1    1    1 0.2875775
```

```
2     2     1 0.7883051
3     3     1 0.4089769
4     4     1 0.8830174
5     5     1 0.9404673
6     6     1 0.0455565
```

```
ggplot(data=tmp,
    aes(y=set2,x=set1)
) +
geom_tile(aes(fill=score))
```




```
ggplot(data=tmp,
    aes(y=set2,x=set1)
) +
geom_contour(aes(z=score))
```



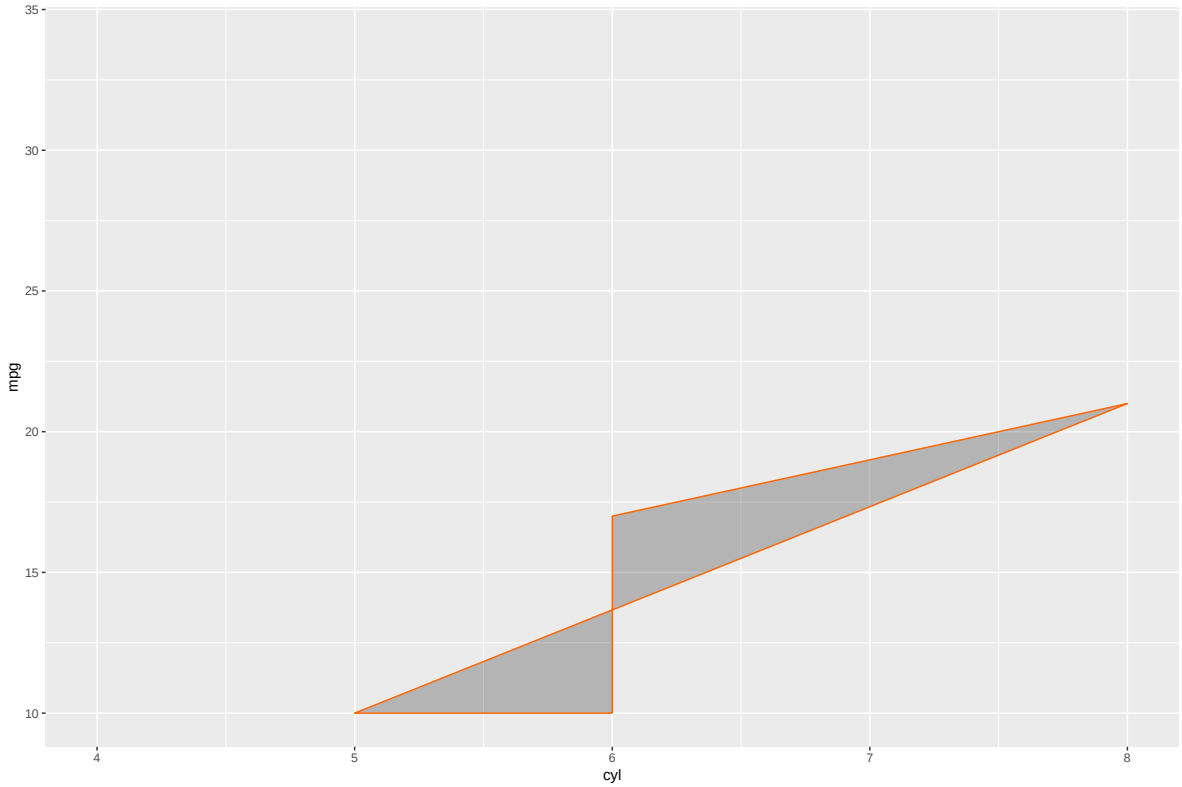
primitives

- Primitives are the basic building blocks
  - several primitives exist: point( ), path( ), polygon( ), segment( ), ribbon( ), rect( ), text( ), blank( )
  - only `geom_point( )` is used very often
  - `geom_ribbon( )` can be interesting for showing intervals
  - typically useful for fine-tuning only
- The `polygon( )` is used, with coordinates created in a separate datafile.
- The same is done with `tiles( )`
- It is possible to create the datafile within the function.
- First a blank plot is drawn, then the rest is added.

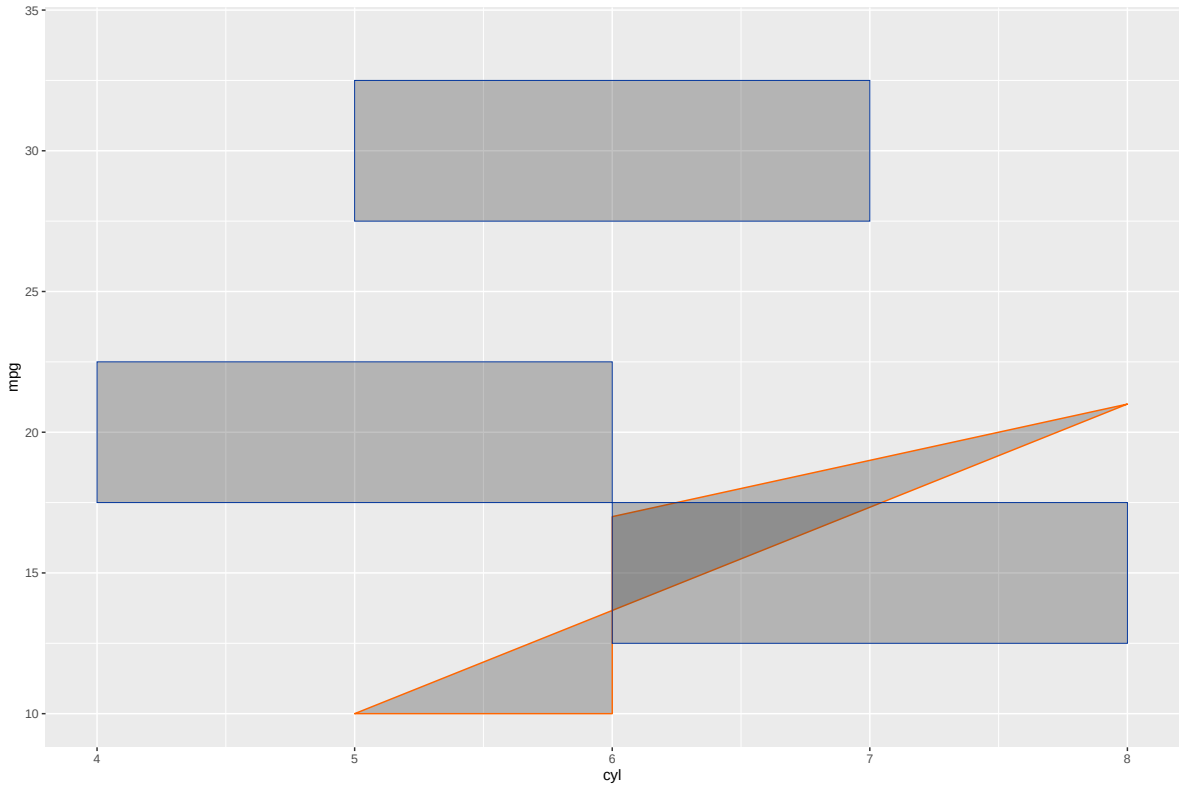
```
blank_plot <- ggplot(mtcars,
    aes(y=mpg,x=cyl)) +
```

```
    geom_blank()

polygon_coordinate_file <- tribble(
~x,~y,
6,17,
6,10,
5,10,
8,21)

tile_coordinate_file <- tribble(
~x,~y,~w,
5,20,2,
6,30,5,
7,15,2)
```

```
(updated_plot <- blank_plot +
    geom_polygon(data=polygon_coordinate_file,
    aes(x=x,y=y),alpha=.3,color="#FF6600"))
```

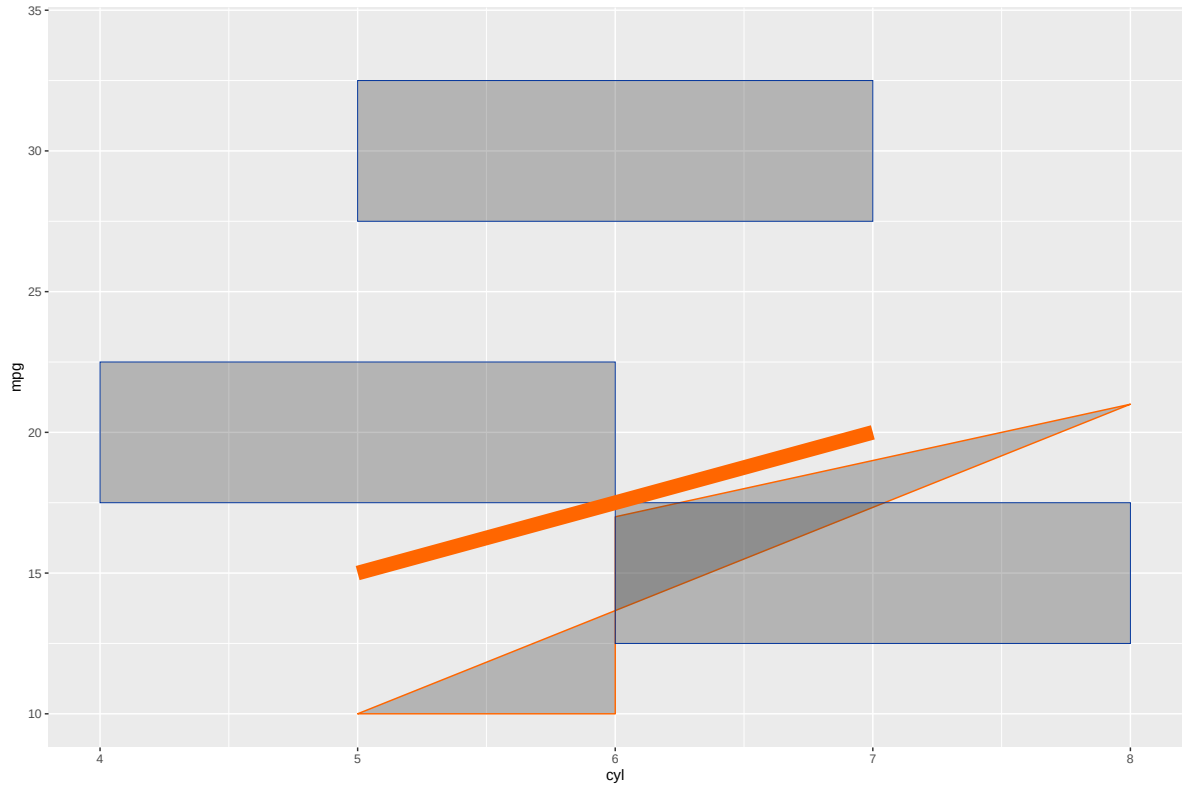


```
(updated_plot <- updated_plot +
    geom_tile(data=tile_coordinate_file,
        aes(x=x,y=y,width=2),alpha=.3,
        color="#003399"))
```




```
updated_plot + geom_segment(data=
    data.frame(x=5,y=15,xend=7,yend=20),
    aes(x=x,y=y,xend=xend,yend=yend),
    size=5,color="#FF6600")
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```



exercises on primitives

- The `mpg` dataset is part of tidyverse's ggplot !

```
data(mpg)
```

- Have a glimpse at the data, what data-types are included ?

```
glimpse(mpg)
```

```
Rows: 234
Columns: 11
$ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi", "~
$ model        <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro", "~
$ displ        <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, 2.~
$ year         <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, 200~
$ cyl          <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 8, 8, ~
$ trans        <chr> "auto(l5)", "manual(m5)", "manual(m6)", "auto(av)", "auto~
```

```
$ drv          <chr> "f", "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", "4~
$ cty          <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17, 1~
$ hwy          <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25, 2~
$ fl           <chr> "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p~
$ class        <chr> "compact", "compact", "compact", "compact", "compact", "c~
```

- Make a scatterplot for `cty` per `hwy`, and assign it to the object name 'current'

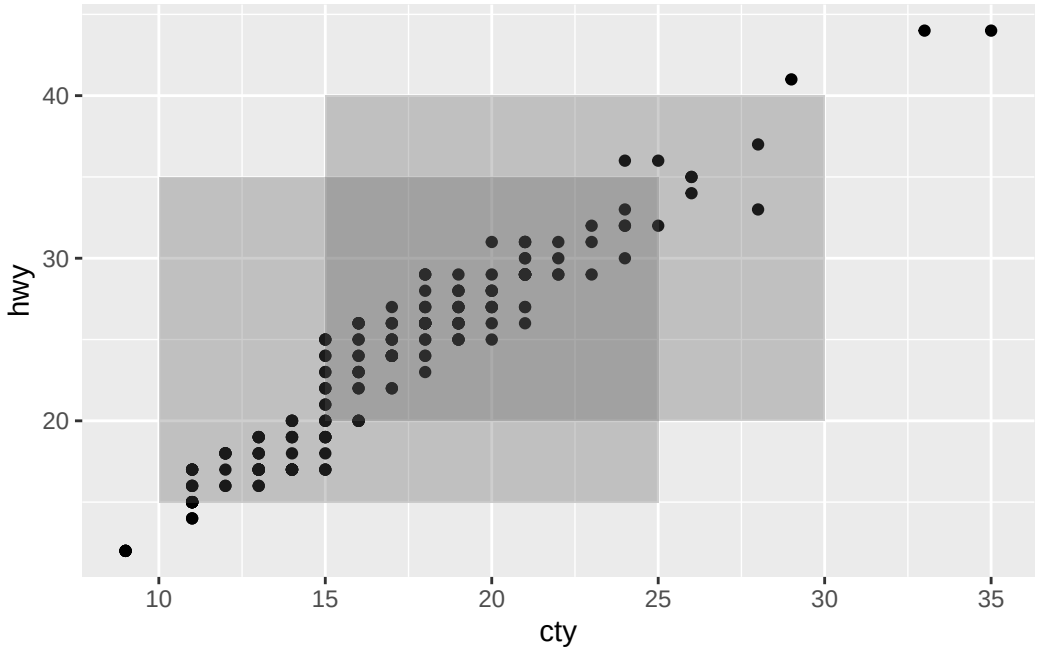
```
current <- ggplot(data=mpg,aes(y=hwy,x=cty)) + geom_point()
```

- Draw 2 rectangles with `geom_rect( )`,
  - one with corners 10,15 and 20,25,
  - one shifted with 5 both dimensions,
  - using 2 additional data frames (tibbles) to store the coordinates.

```
# tmp <- data.frame(
    # c1=c(10,15),
    # c2=c(15,20),
    # c3=c(25,30),
    # c4=c(35,40))
my_2_sets_of_coordinates <- tribble(~xi,~yi,~xa,~ya,
10,15,25,35,
15,20,30,40
)
```

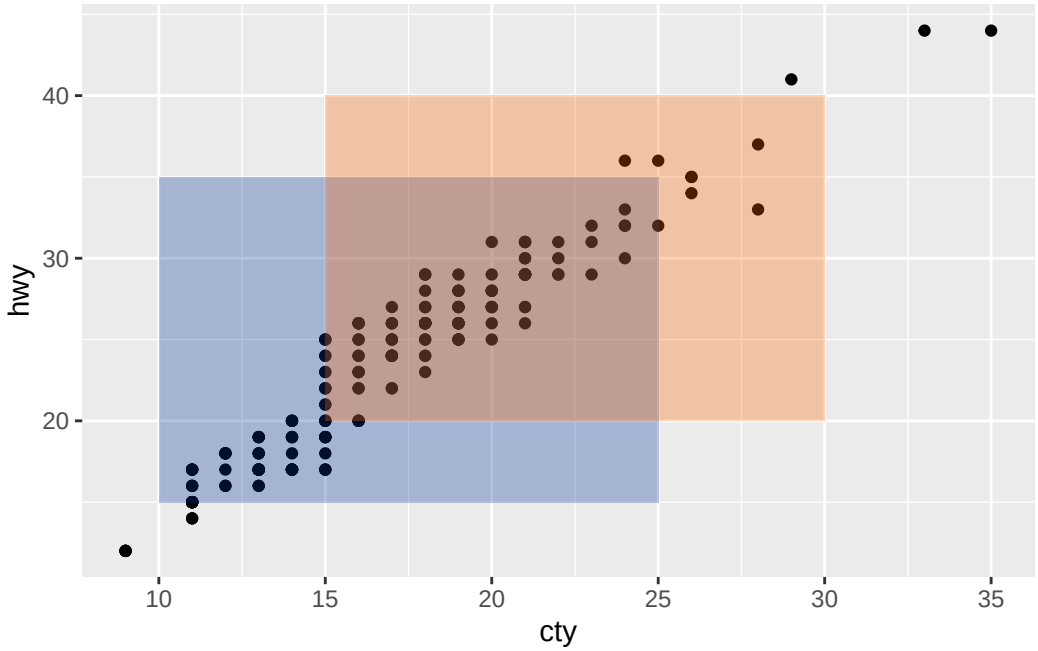
- Plot the coordinates on top of the scatterplot
- Note: the argument inherit.aes=FALSE avoids inheriting aesthetics

```
current + geom_rect(data=my_2_sets_of_coordinates,
    aes(xmin=xi,ymin=yi,xmax=xa,ymax=ya),
    inherit.aes=FALSE,alpha=.3)
```



- Add a different color to each separately, use '#FF6600' and '#003399'

```
current +
    geom_rect(data=my_2_sets_of_coordinates[1,],
        aes(xmin=xi,ymin=yi,xmax=xa,ymax=ya),
        inherit.aes=FALSE,alpha=.3,
        fill="#003399") +
    geom_rect(data=my_2_sets_of_coordinates[2,],
        aes(xmin=xi,ymin=yi,xmax=xa,ymax=ya),
        inherit.aes=FALSE,alpha=.3,
        fill="#FF6600")
```

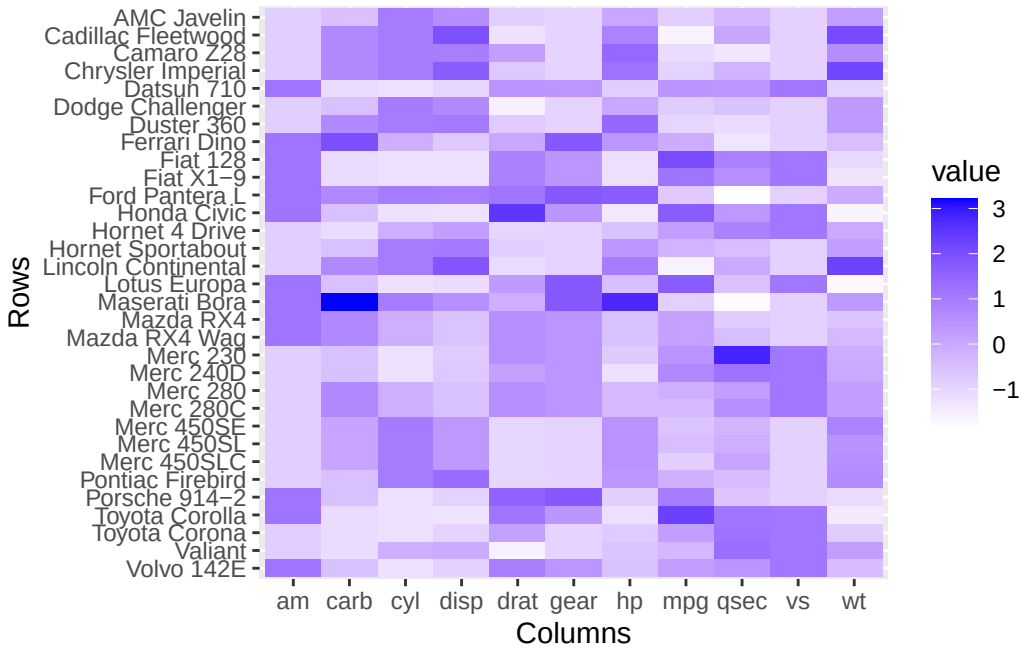


Beyond ggplot2: -gg- packages

- Various extensions have been developed, each showing the same basic structure but typically addressing particular types of visualizations
- Check https://exts.ggplot2.tidyverse.org/gallery/

```
library(ggfortify)
autoplot(scale(mtcars))
```

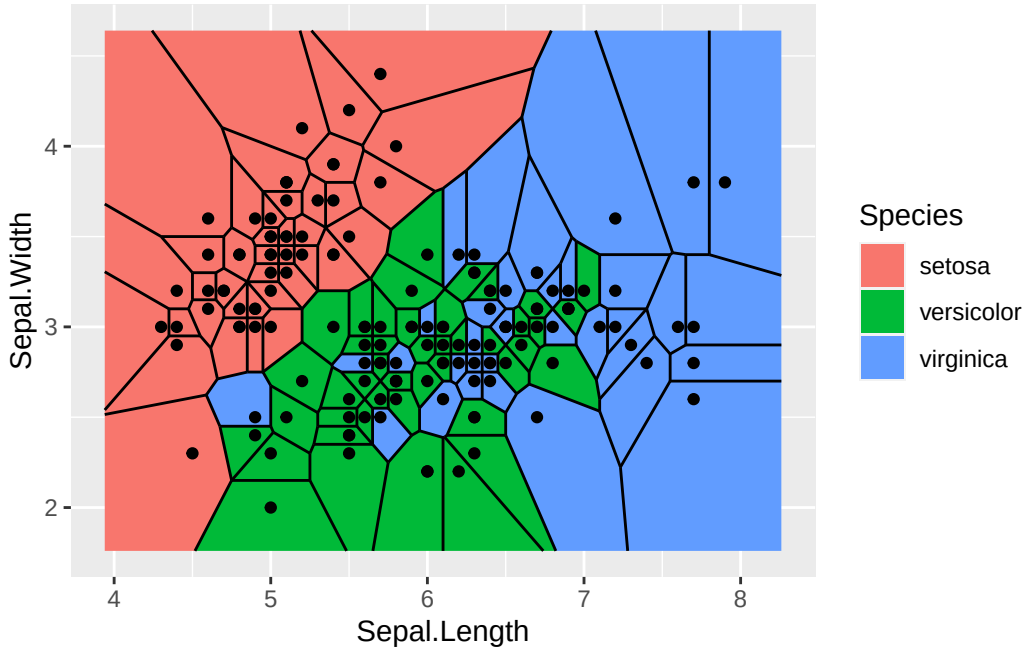
```
Scale for y is already present.
Adding another scale for y, which will replace the existing scale.
```



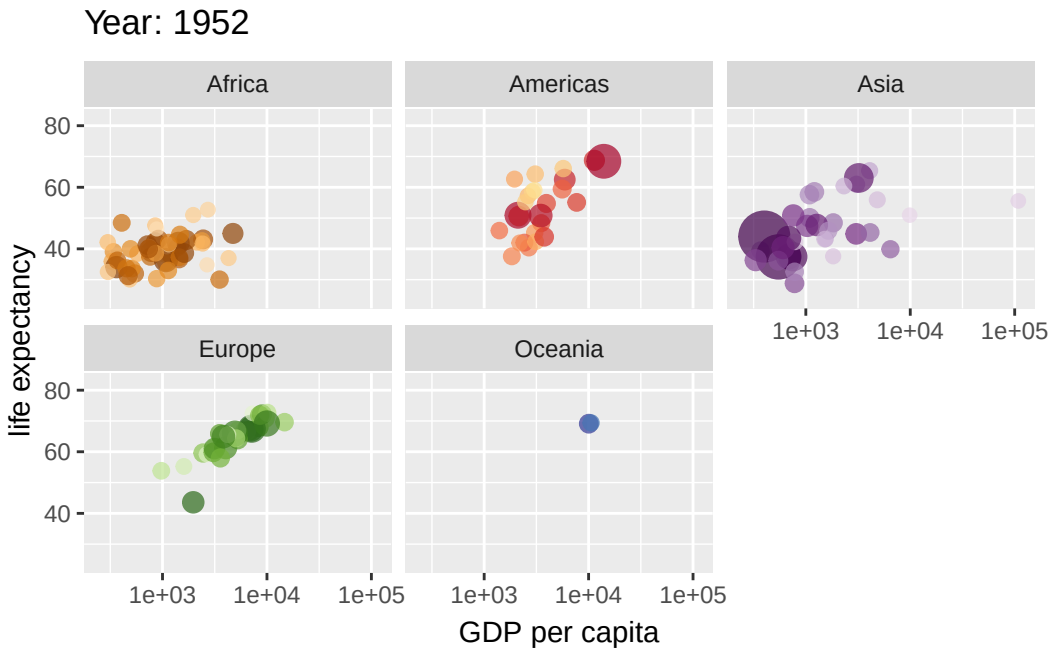

```
library(ggforce)
ggplot(iris, aes(Sepal.Length, Sepal.Width)) +
  geom_voronoi_tile(aes(fill = Species, group = -1L)) +
  geom_voronoi_segment() +
  geom_point()
```

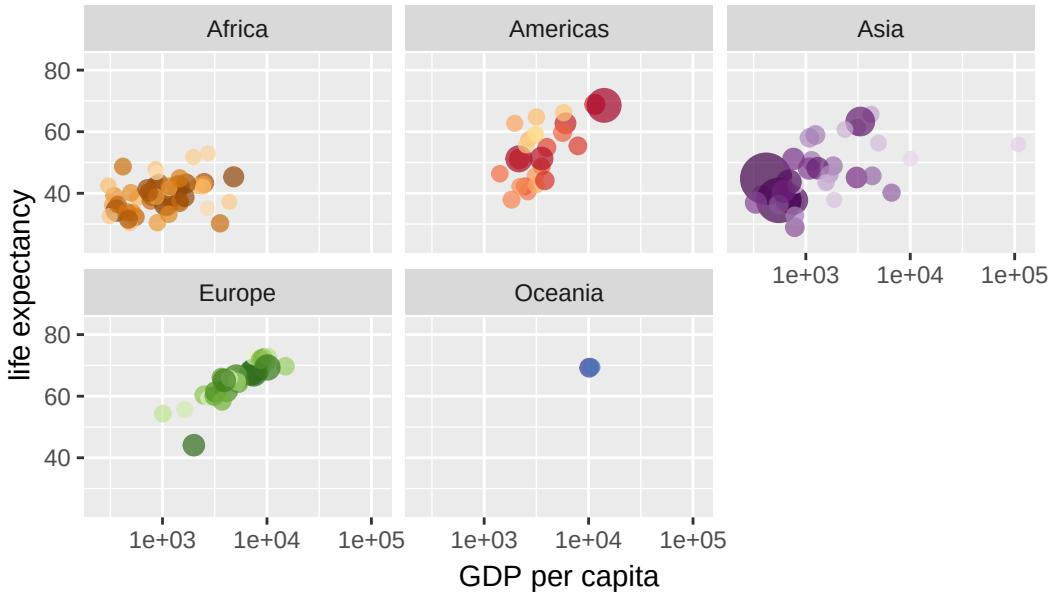
```
Warning: `stat_voronoi_tile()` is dropping duplicated points
```

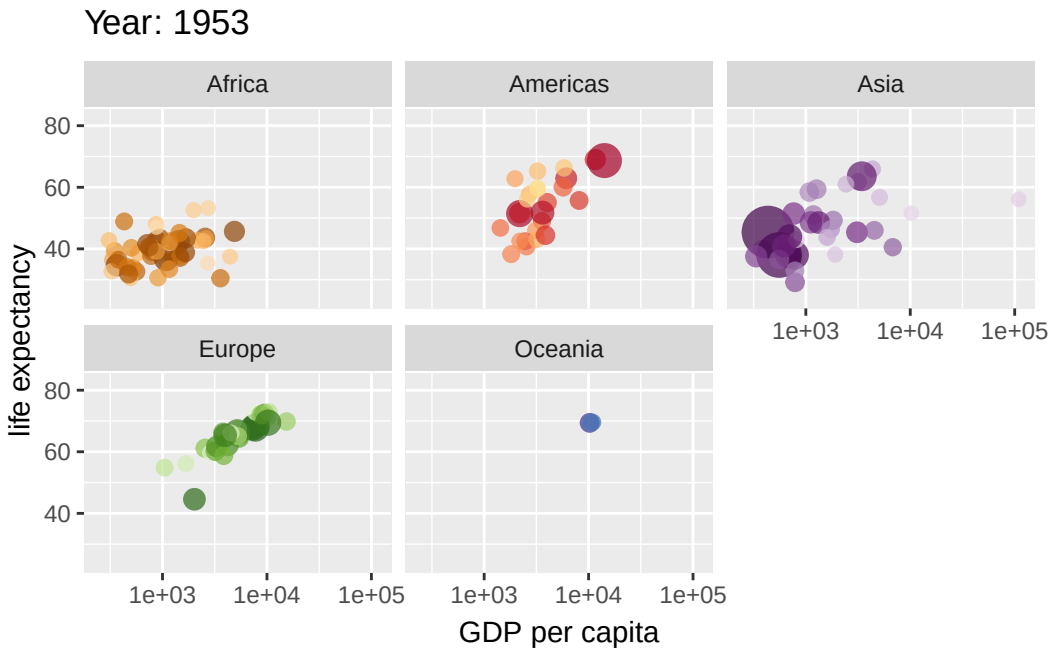
```
Warning: `stat_voronoi_segment()` is dropping duplicated points
```

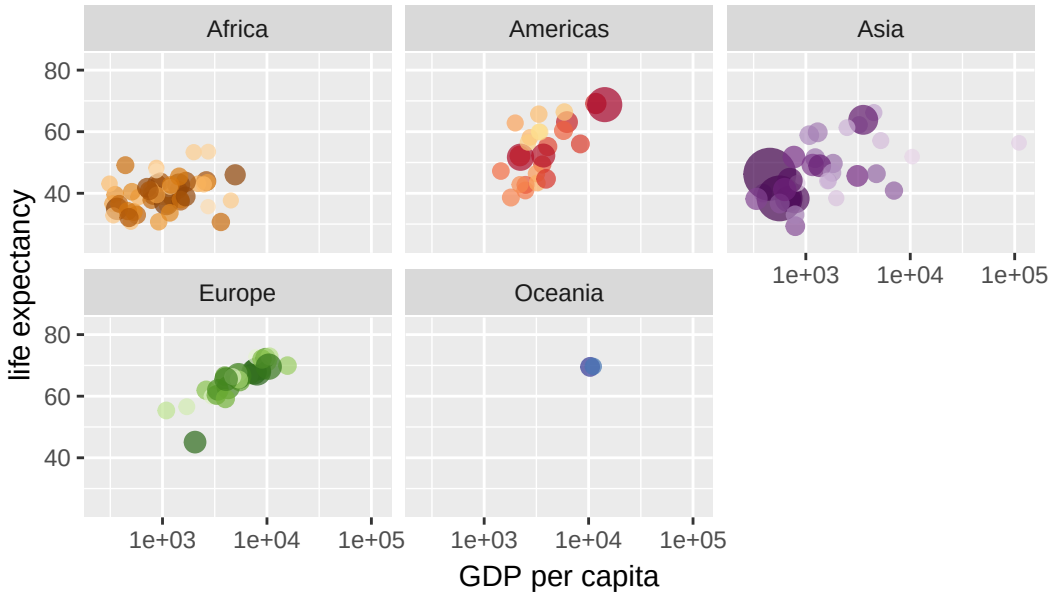


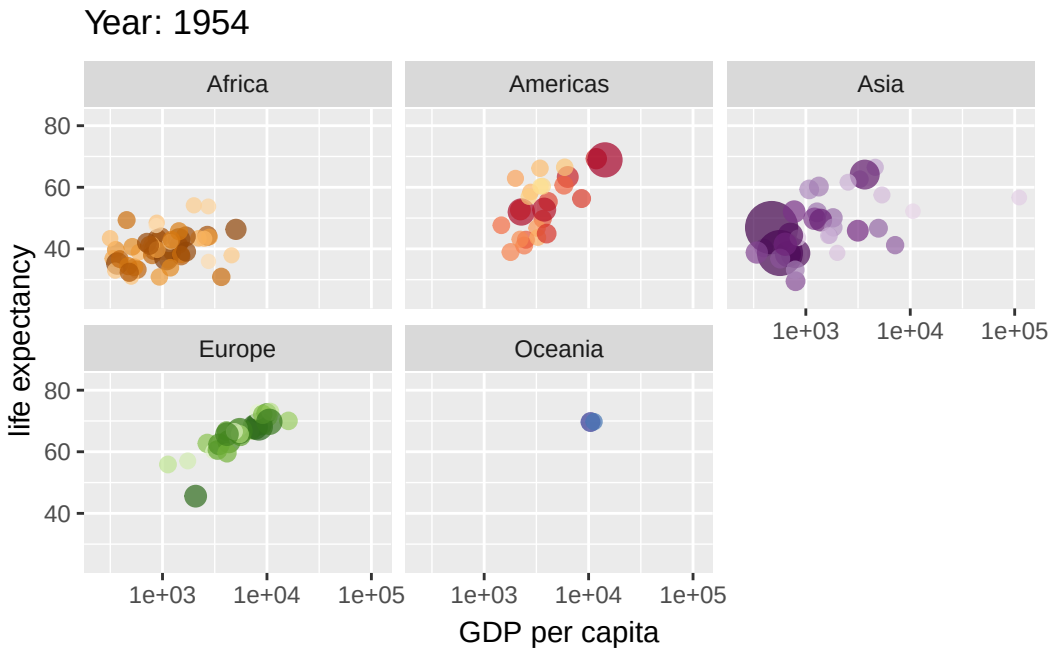

```
library(gganimate)
library(gapminder)
ggplot(gapminder, aes(gdpPercap, lifeExp, size = pop, colour = country)) +
  geom_point(alpha = 0.7, show.legend = FALSE) +
  scale_colour_manual(values = country_colors) +
  scale_size(range = c(2, 12)) +
  scale_x_log10() +
  facet_wrap(~continent) +
  labs(title = 'Year: {frame_time}', x = 'GDP per capita', y = 'life expectancy') +
  transition_time(year) +
  ease_aes('linear')
```

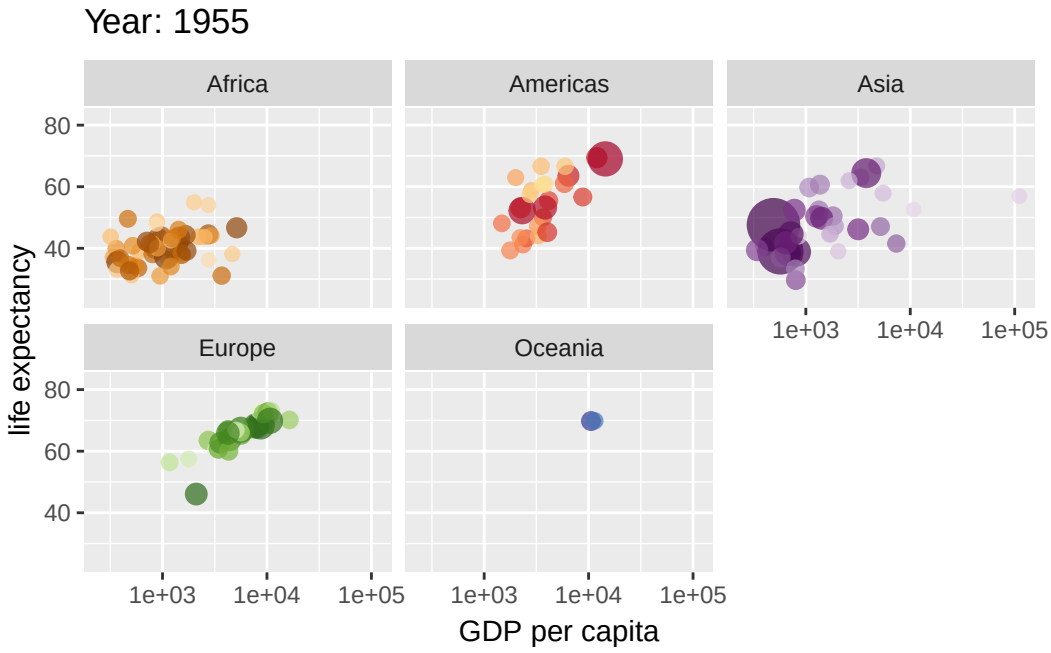

Year: 1952
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

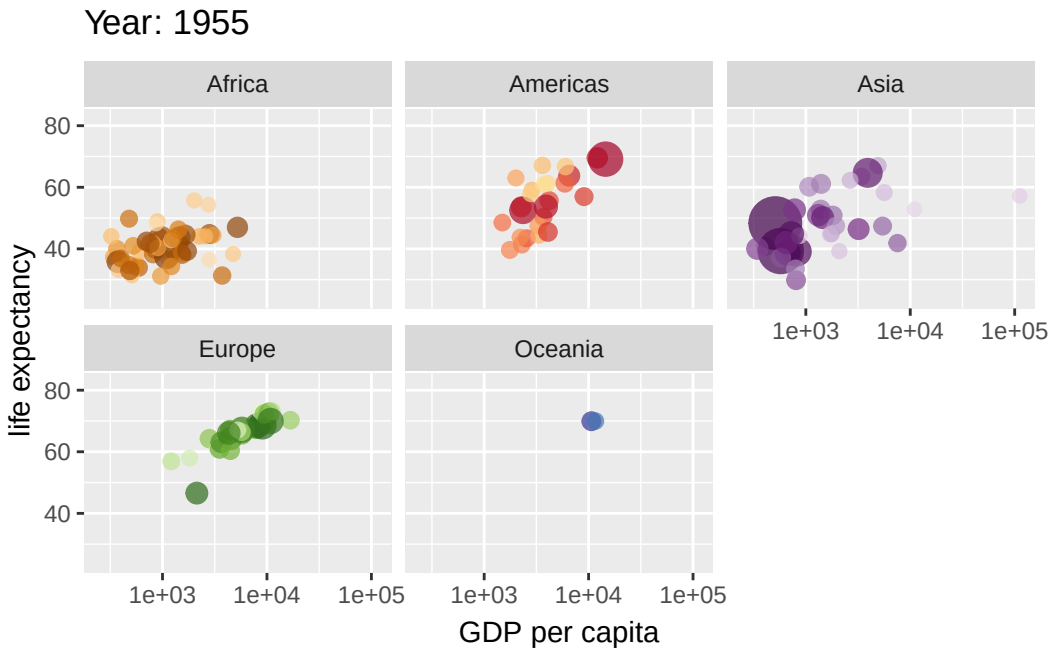

Year: 1953
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

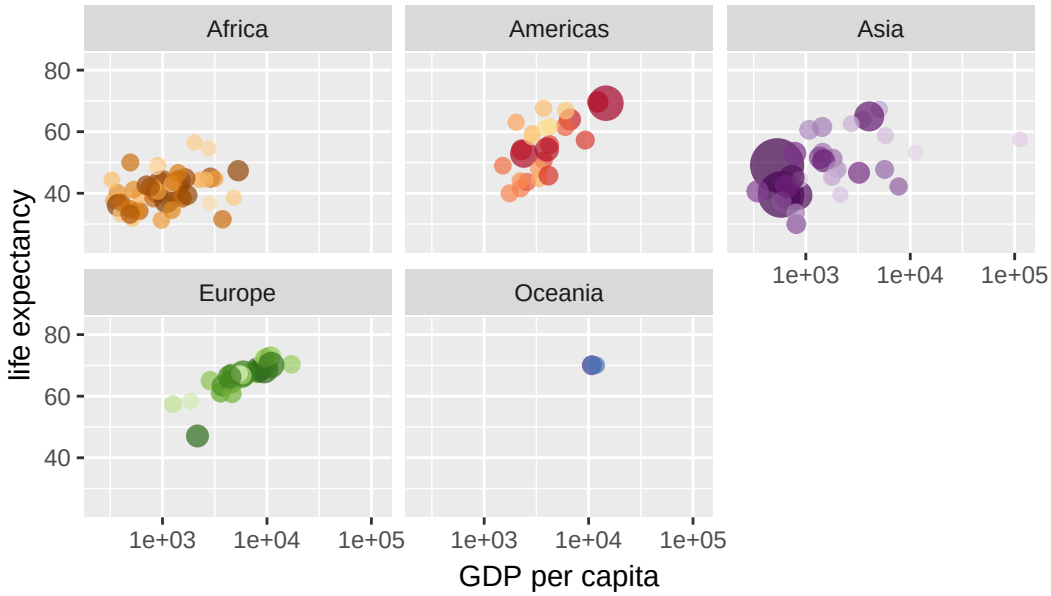

Year: 1953
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

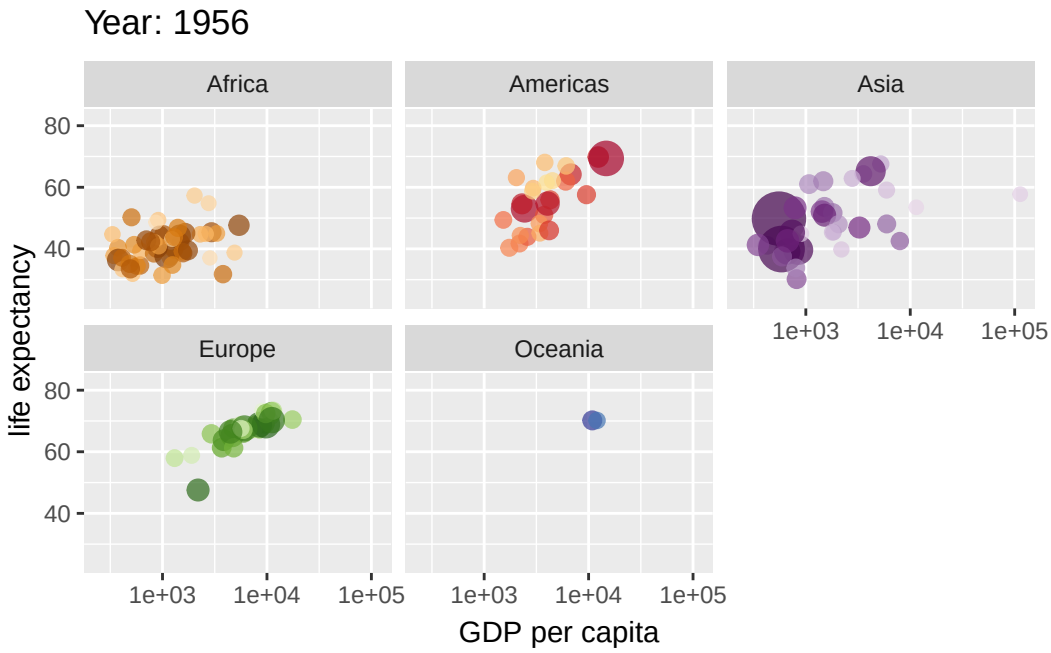

Year: 1954
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

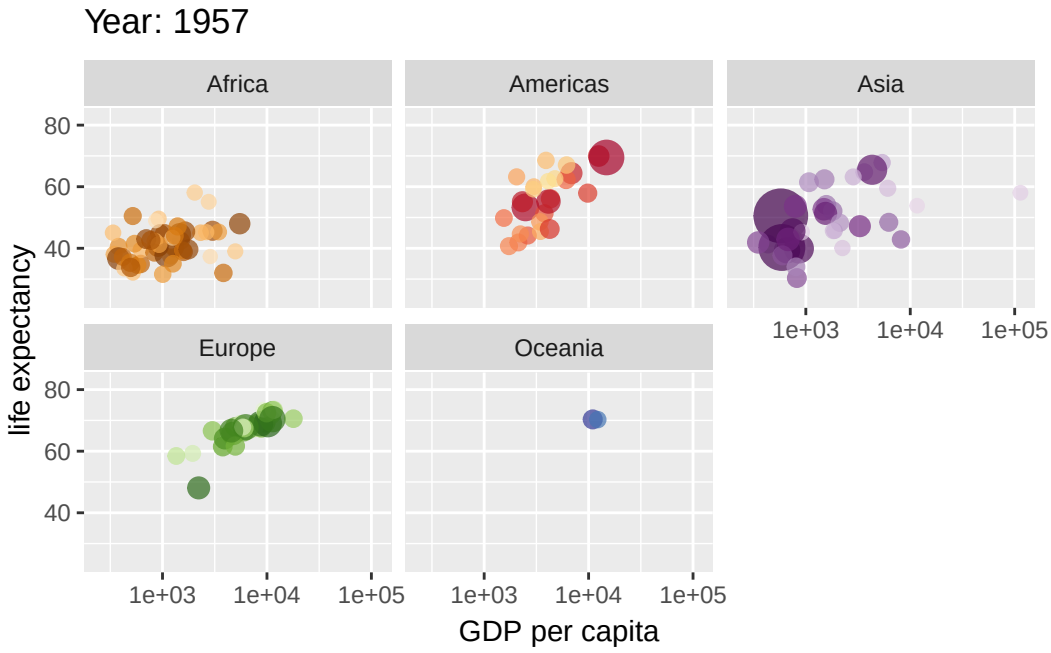

Year: 1954
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

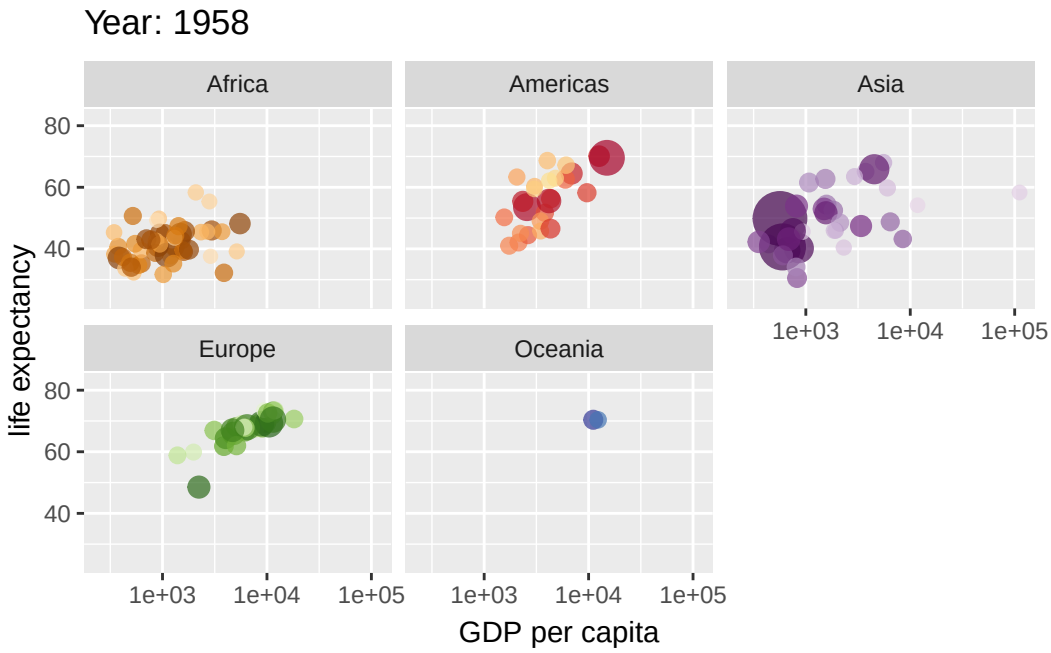

Year: 1955
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

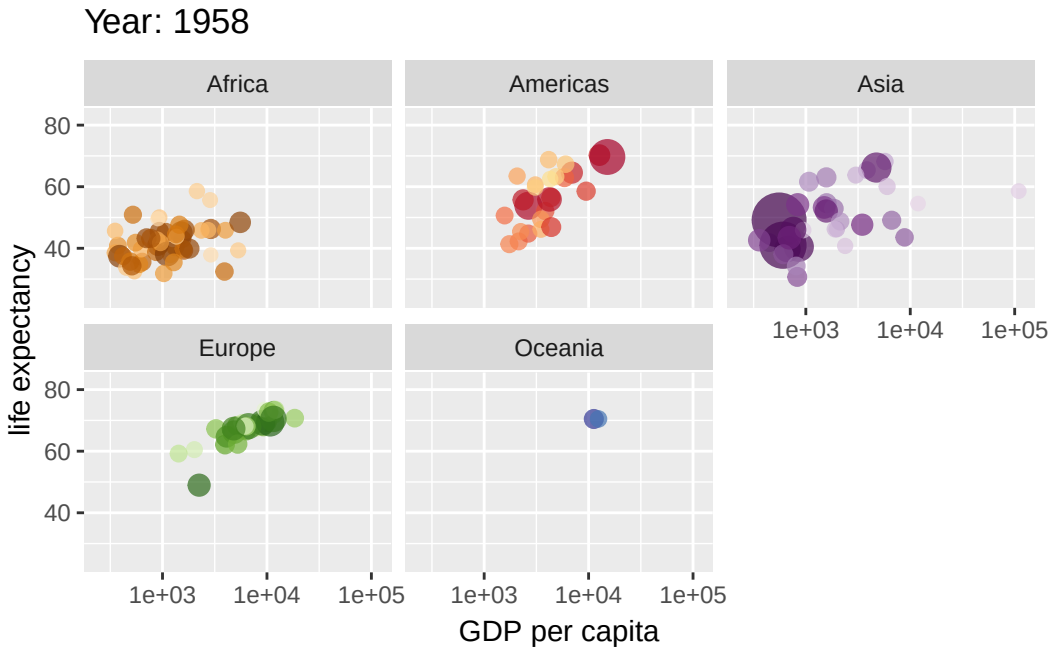

Year: 1955
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

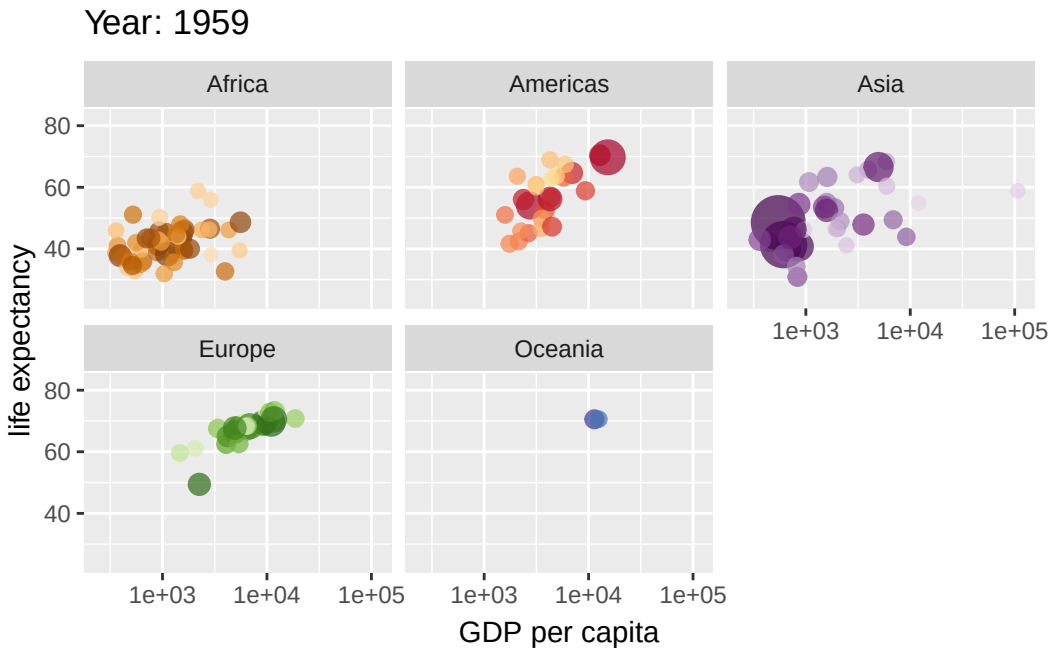

Year: 1956
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

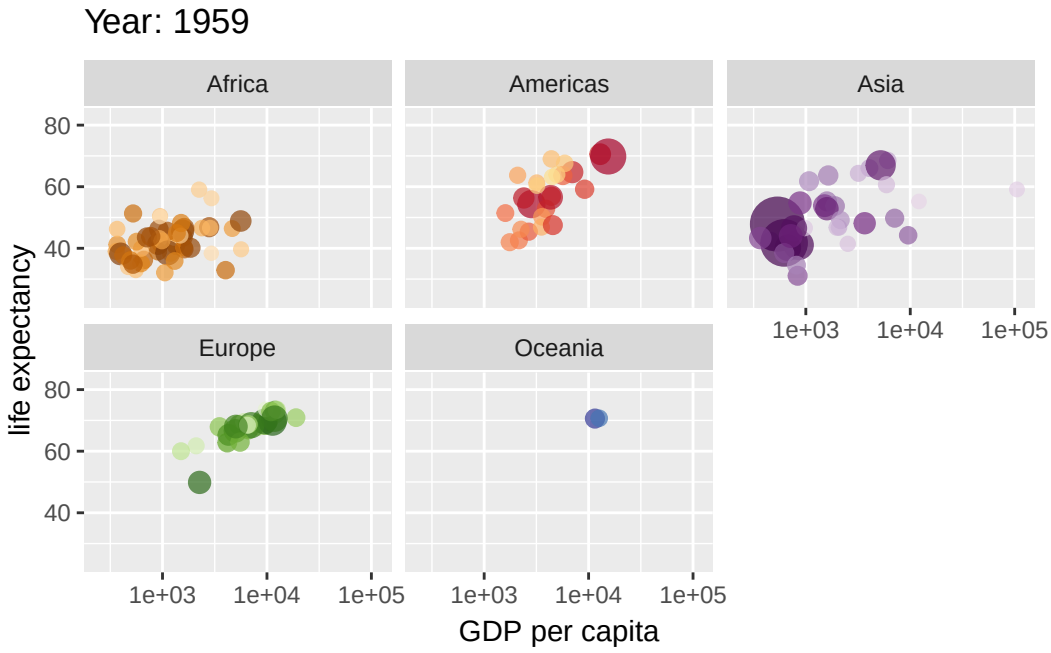

Year: 1956
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

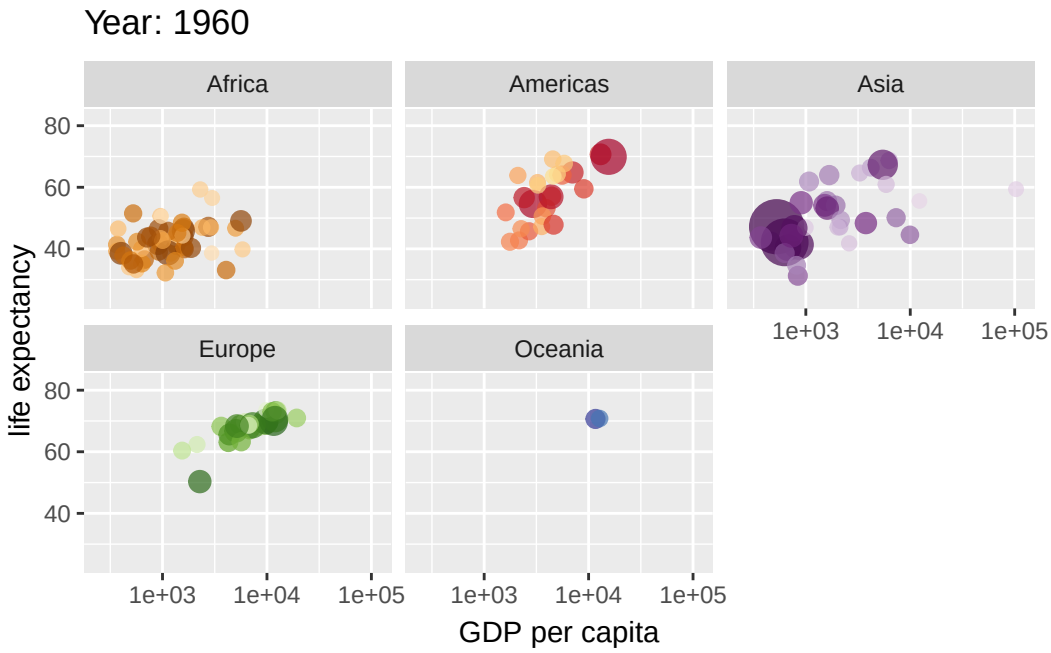

Year: 1957
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

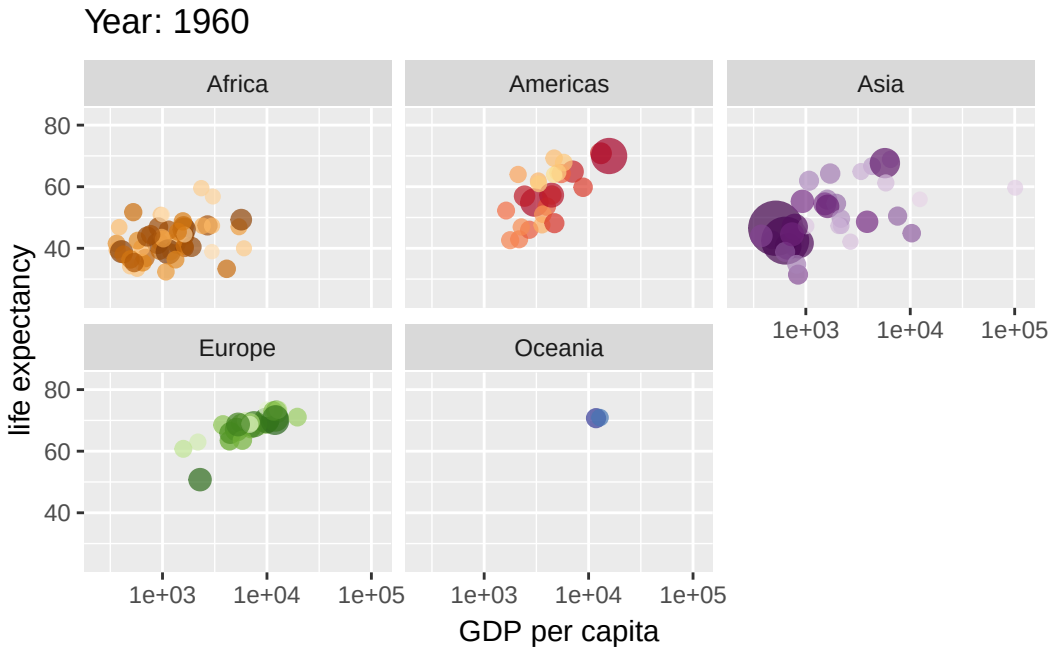

Year: 1958
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

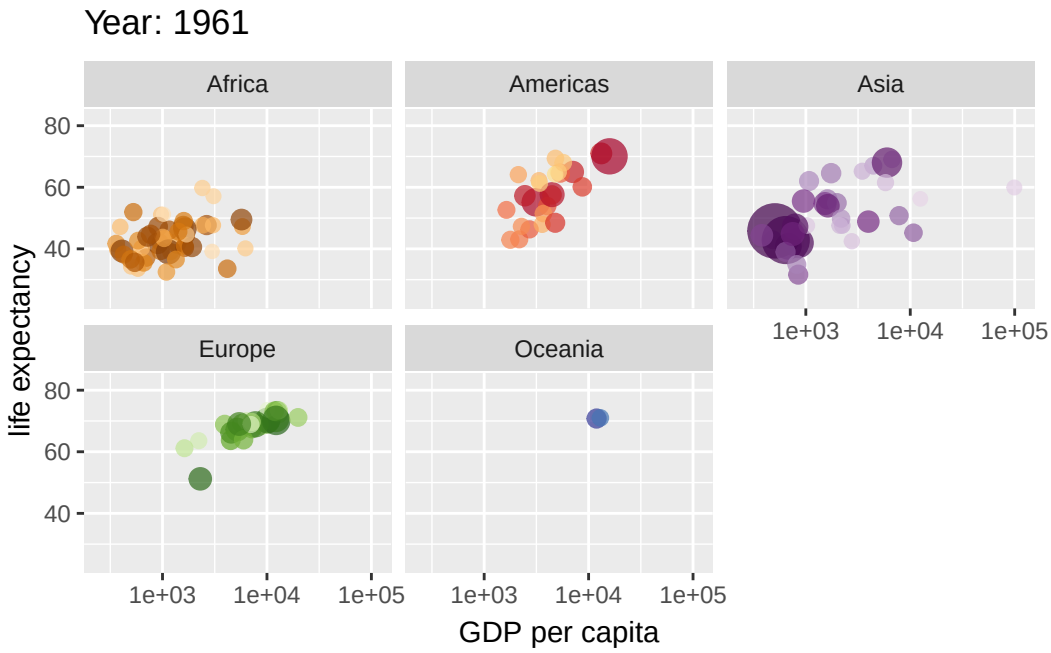

Year: 1958
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

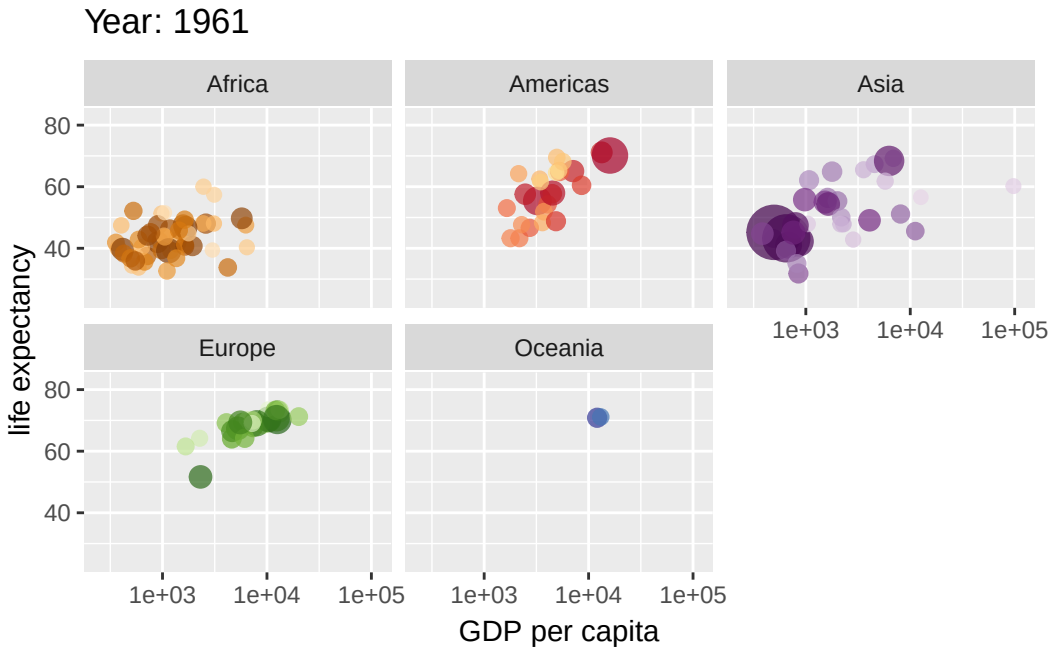

Year: 1959
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

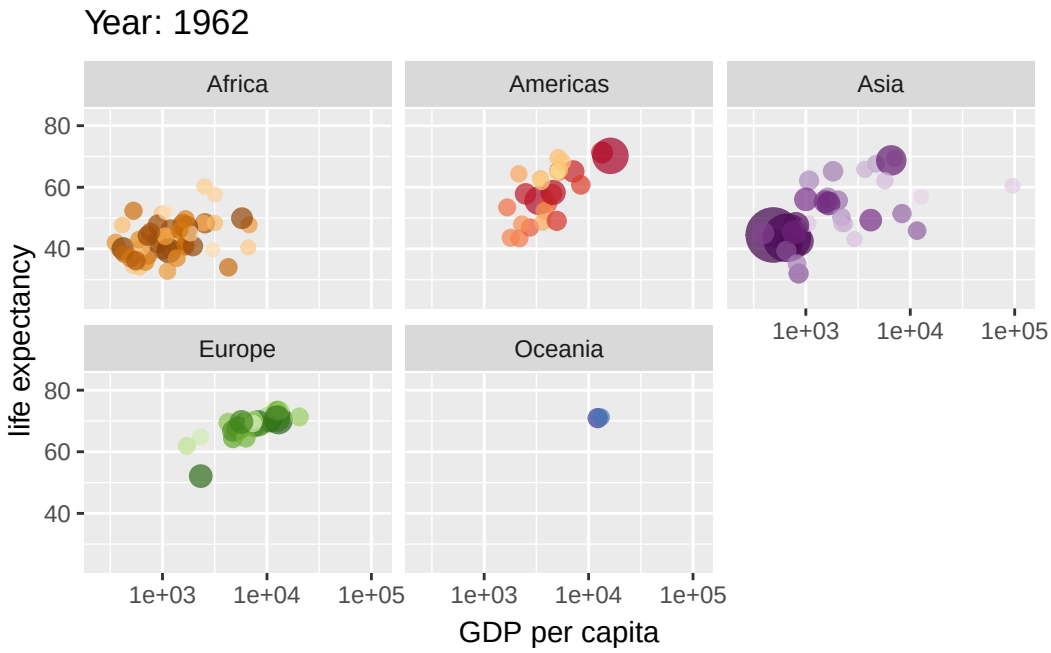

Year: 1959
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

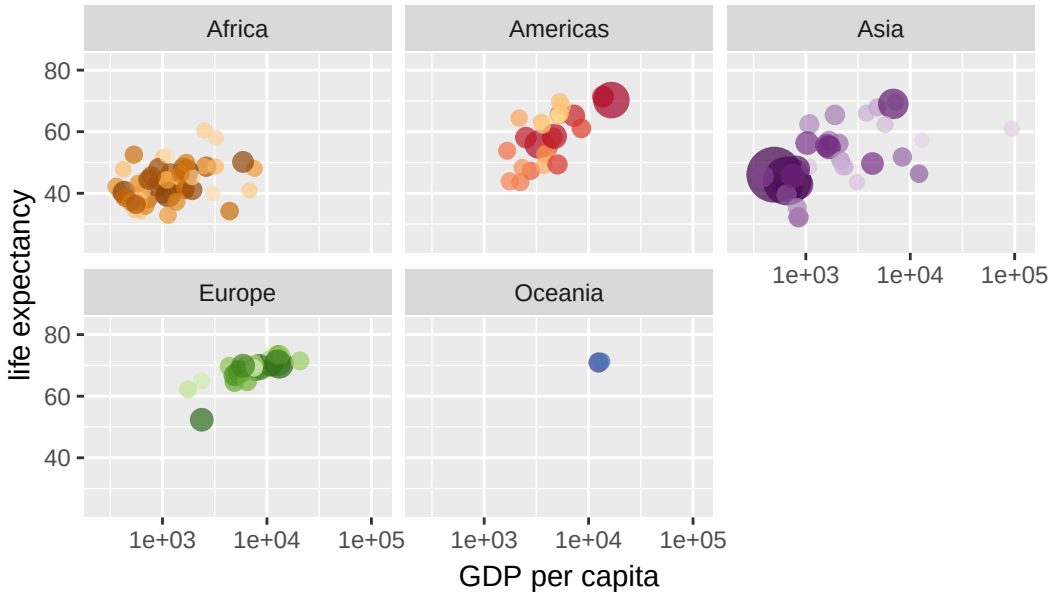

Year: 1960
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

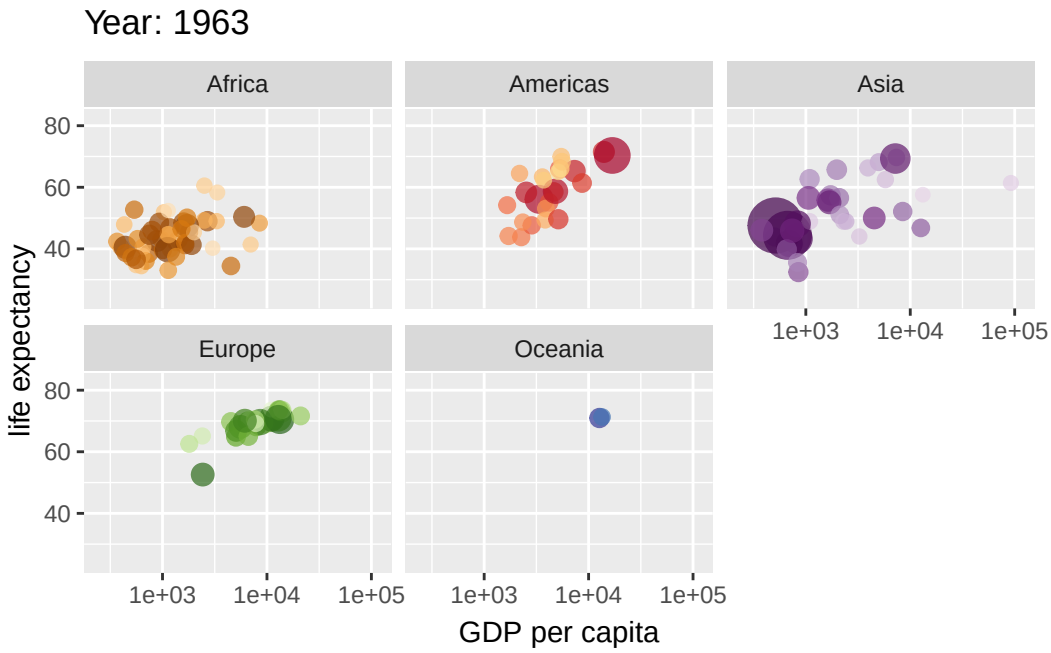

Year: 1960
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

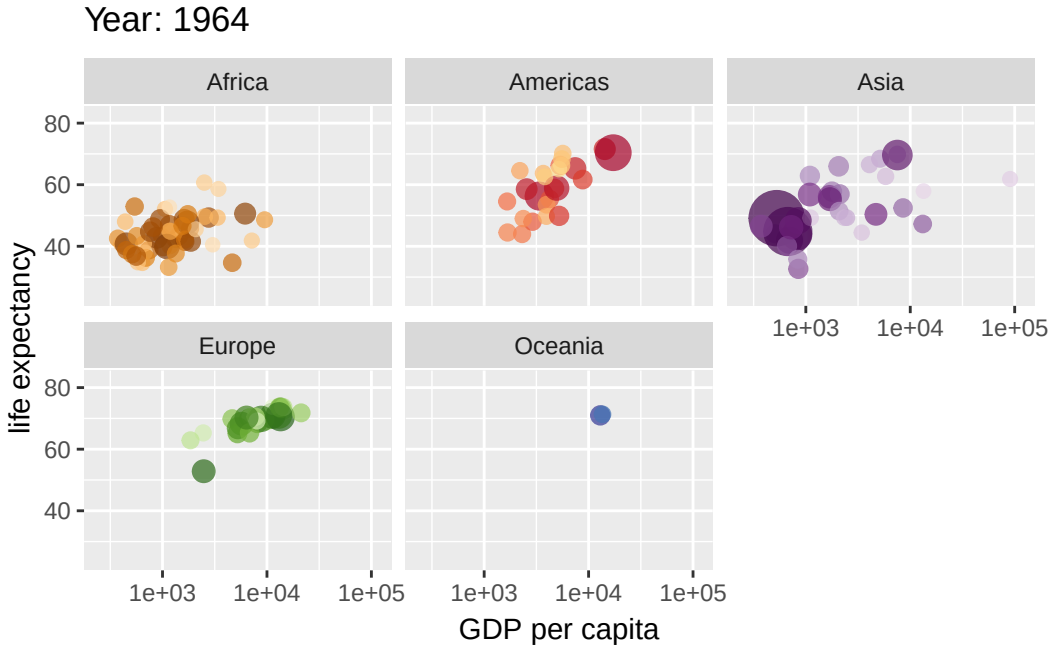

Year: 1961
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

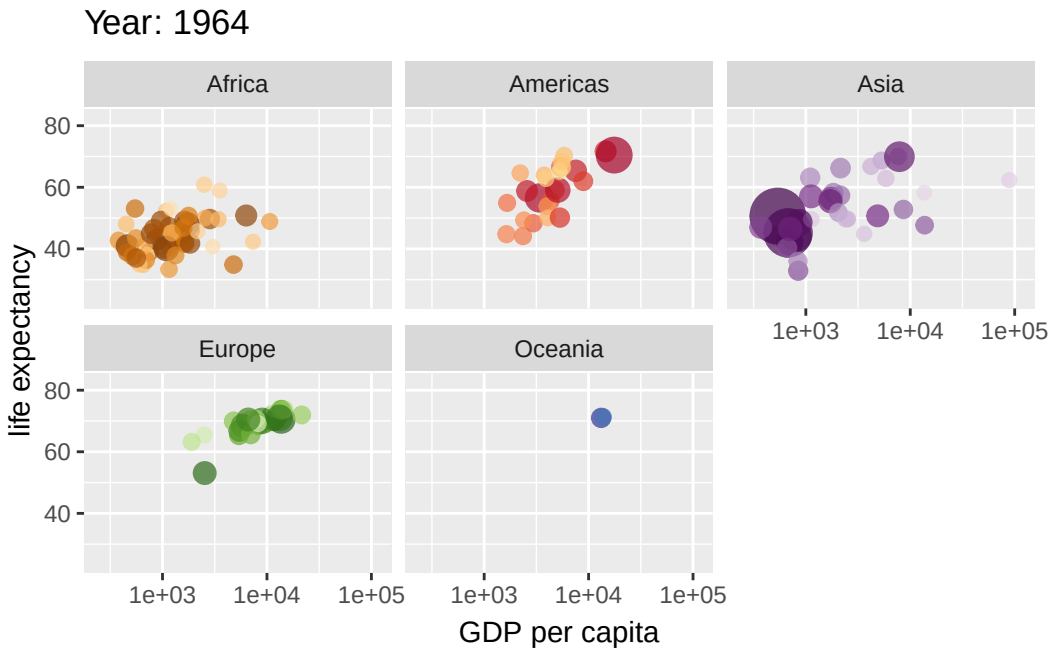

Year: 1961
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

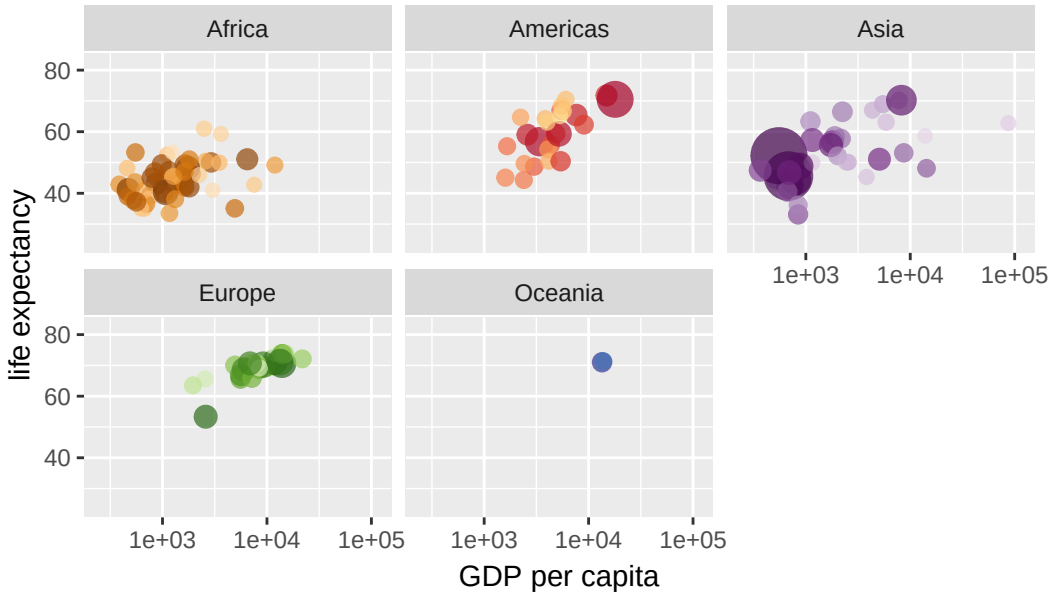

Year: 1962
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

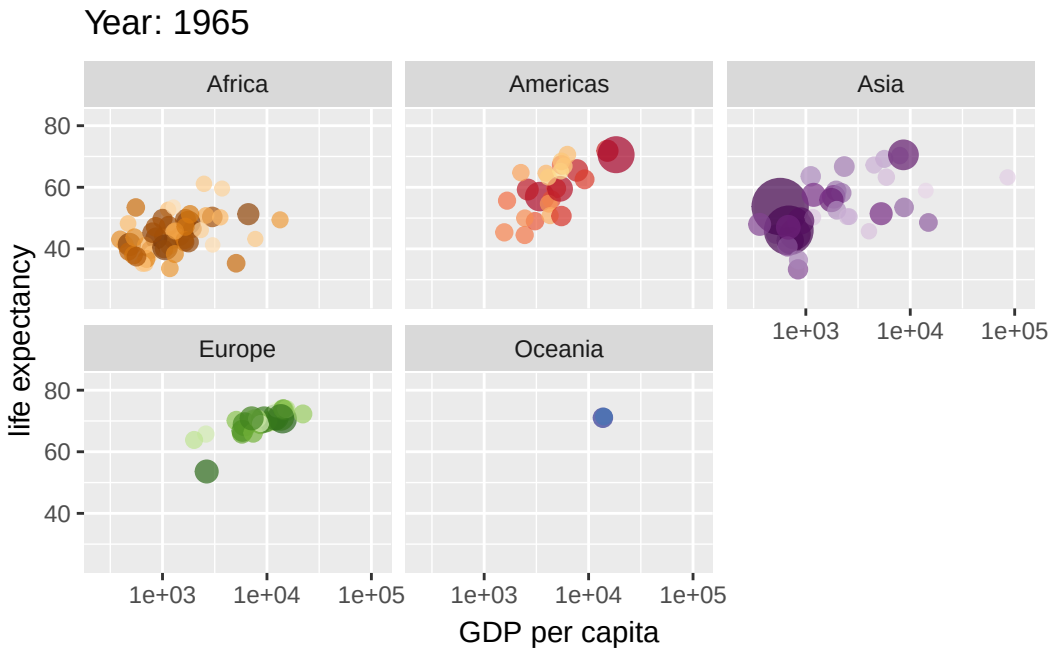

Year: 1963
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

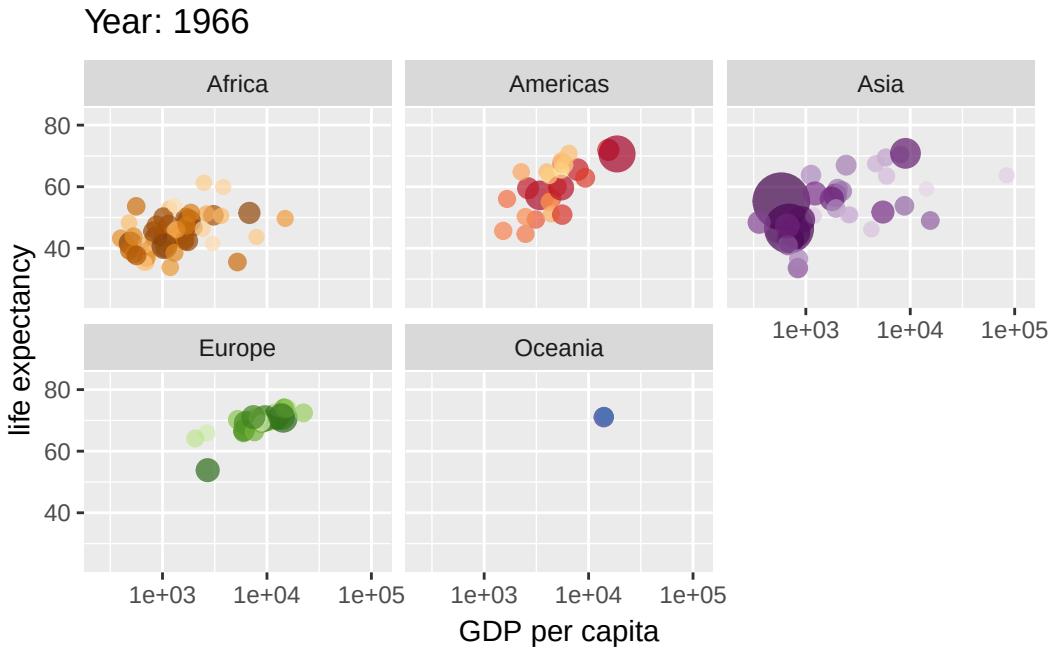

Year: 1963
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

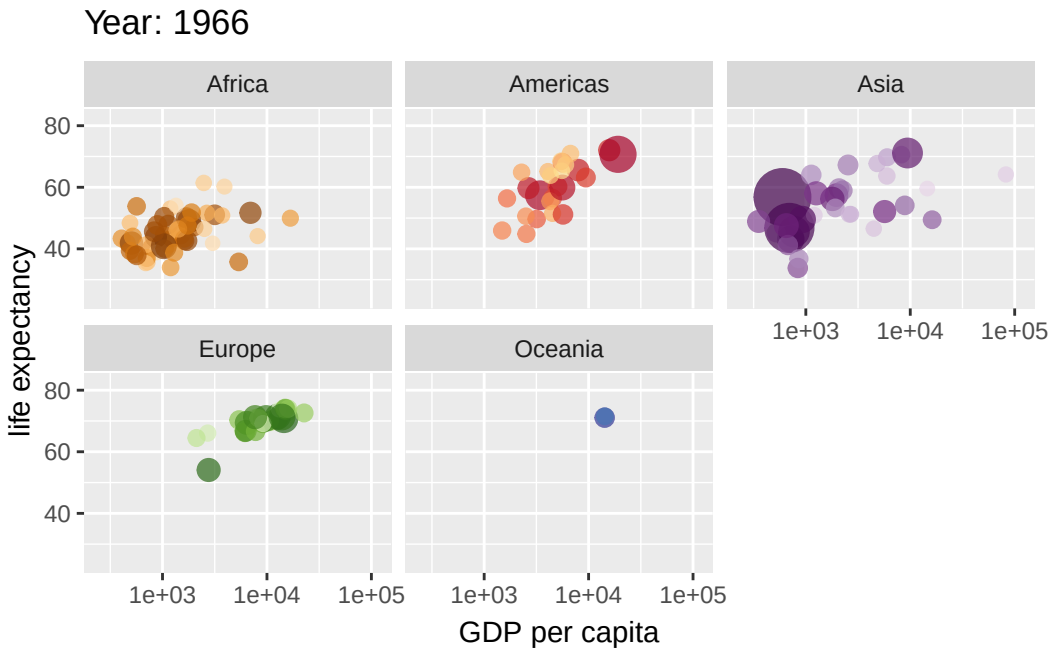

Year: 1964
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

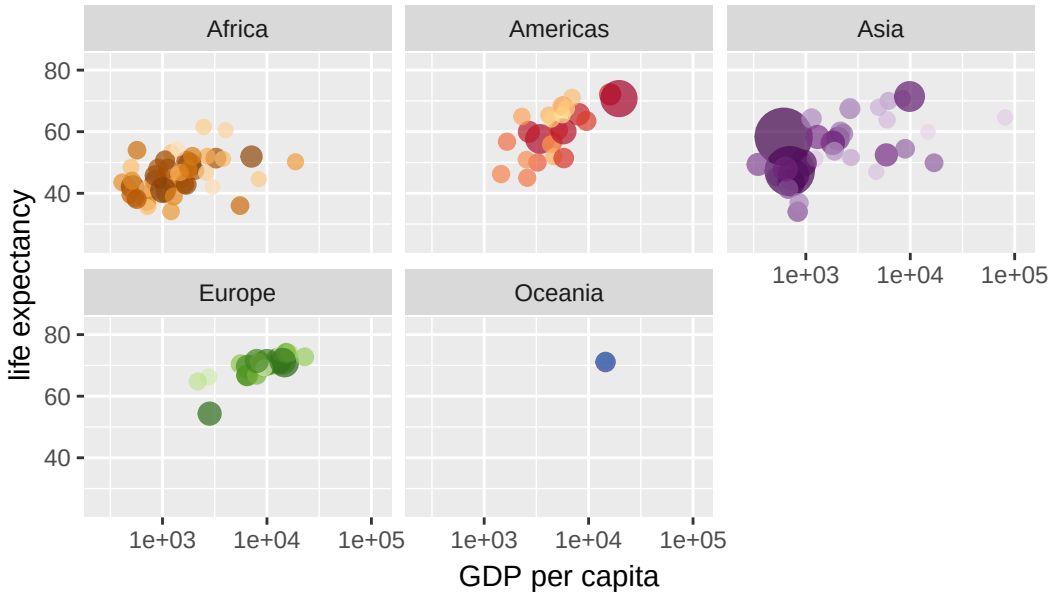

Year: 1964
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

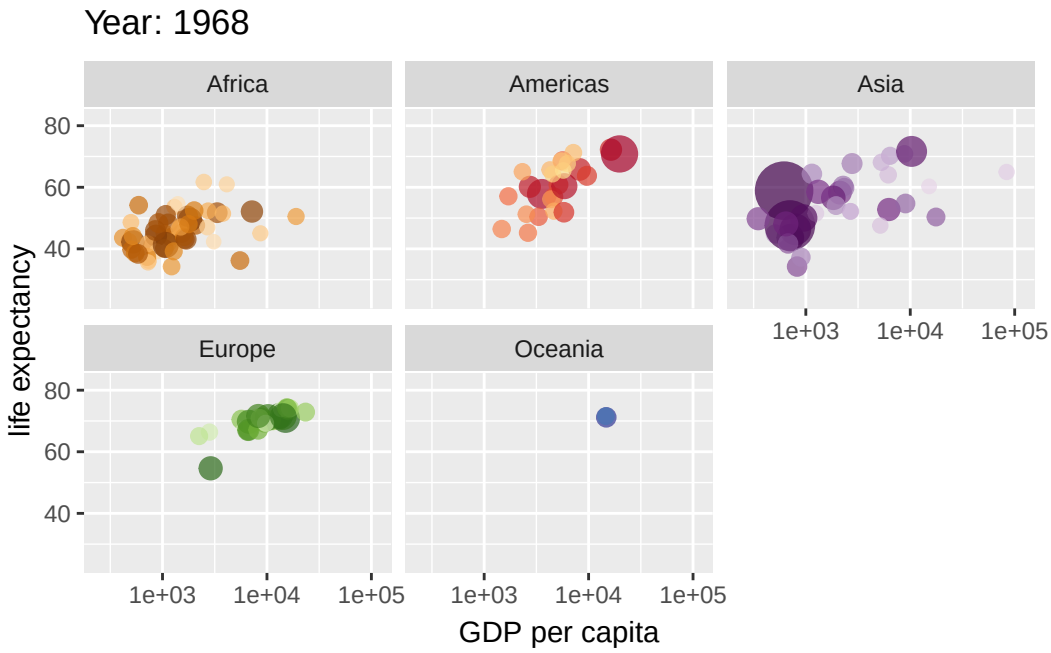

Year: 1965
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

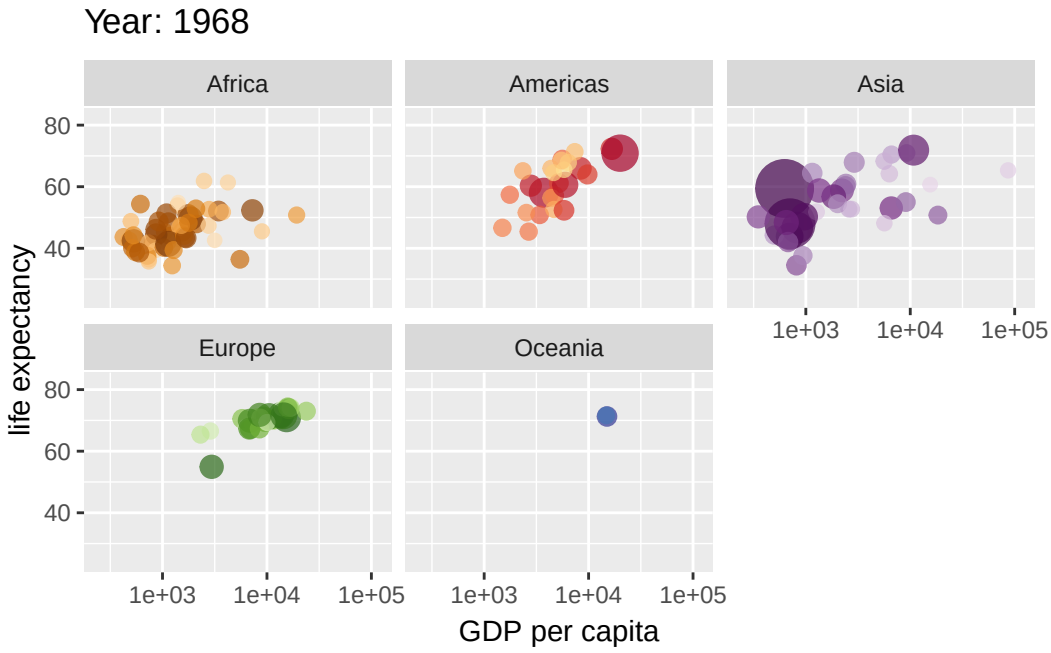

Year: 1965
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

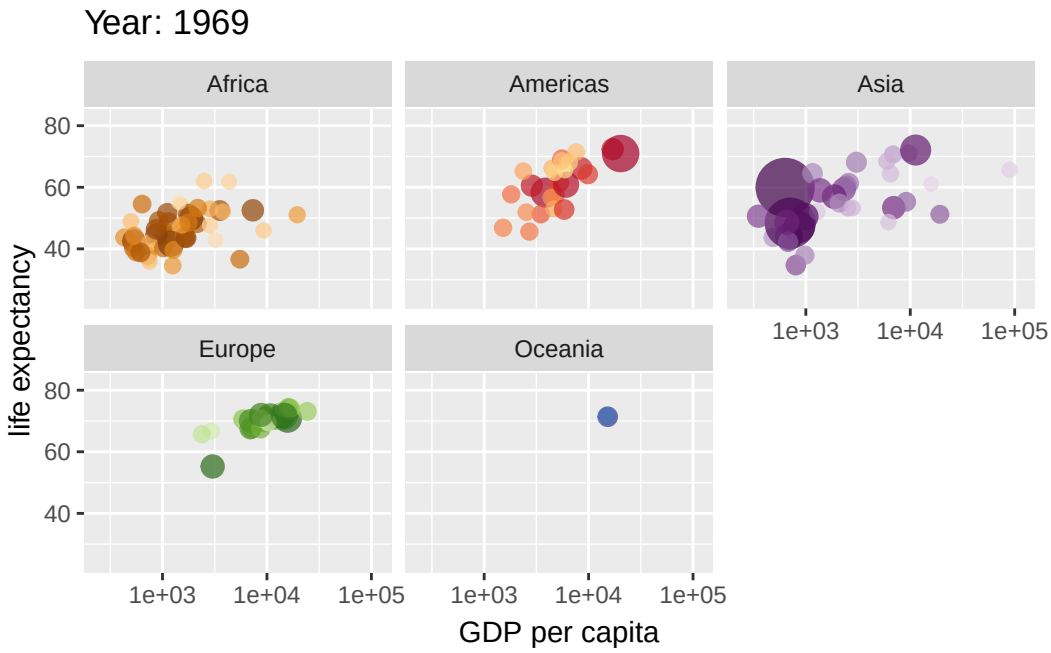

Year: 1966
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

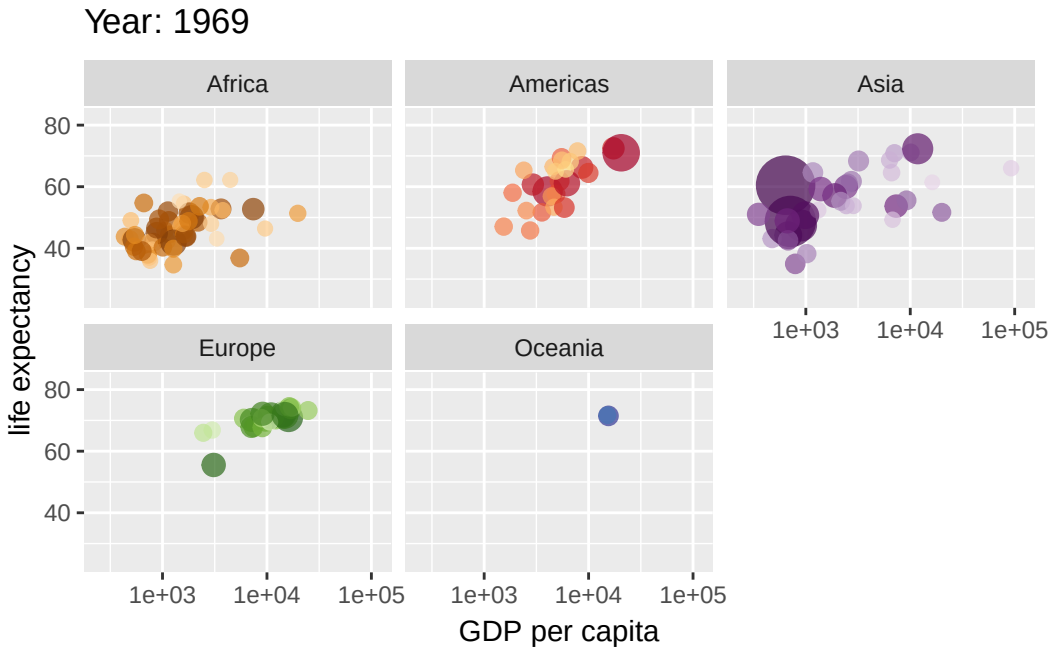

Year: 1966
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

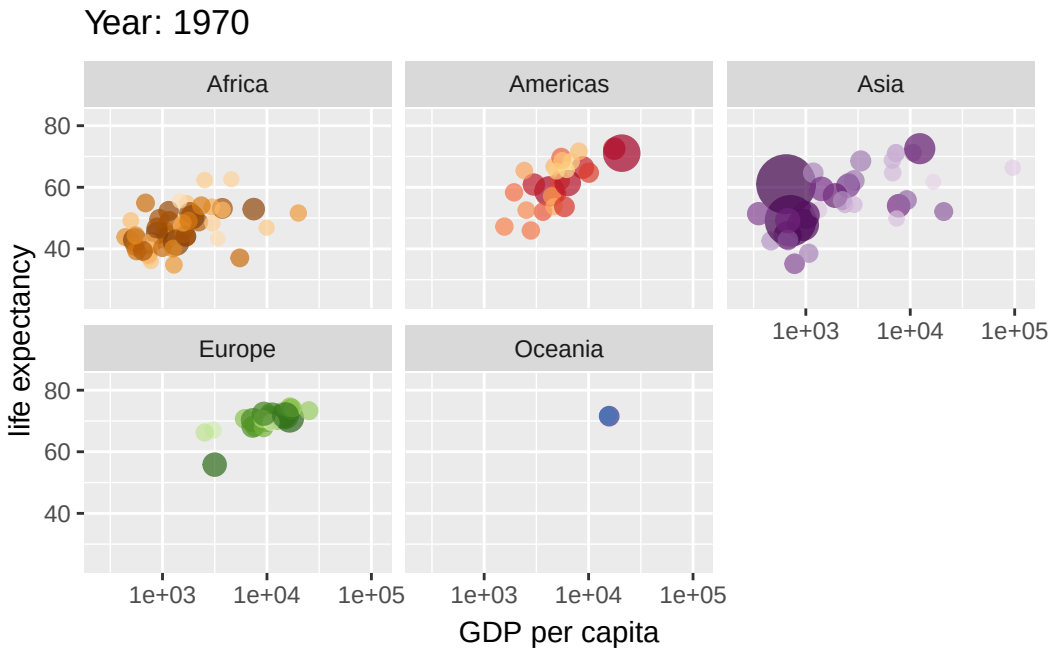

Year: 1967
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

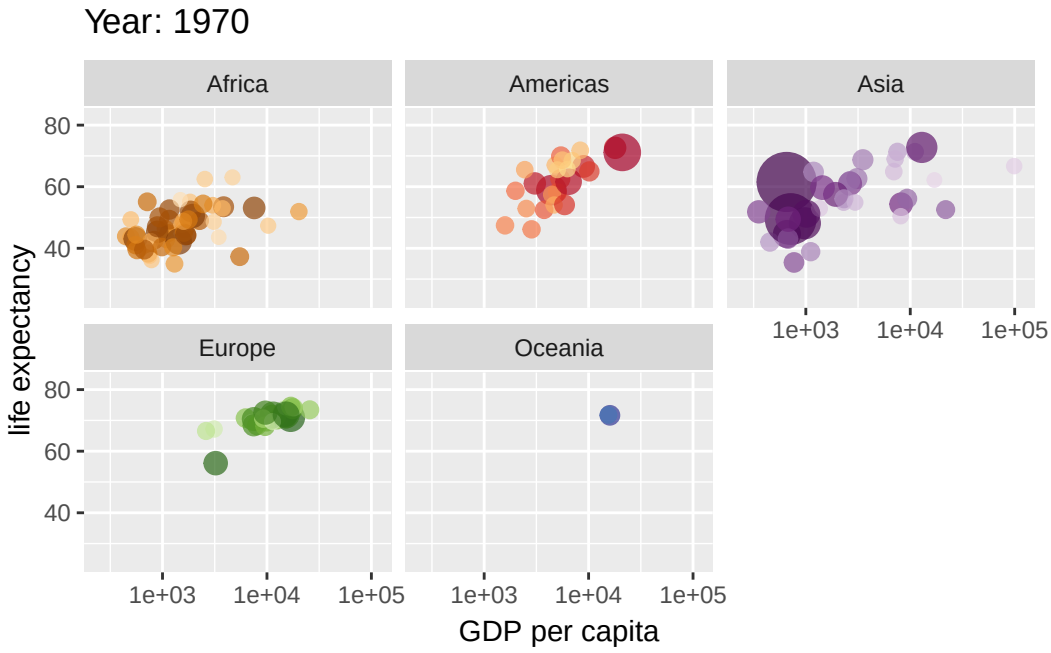

Year: 1968
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

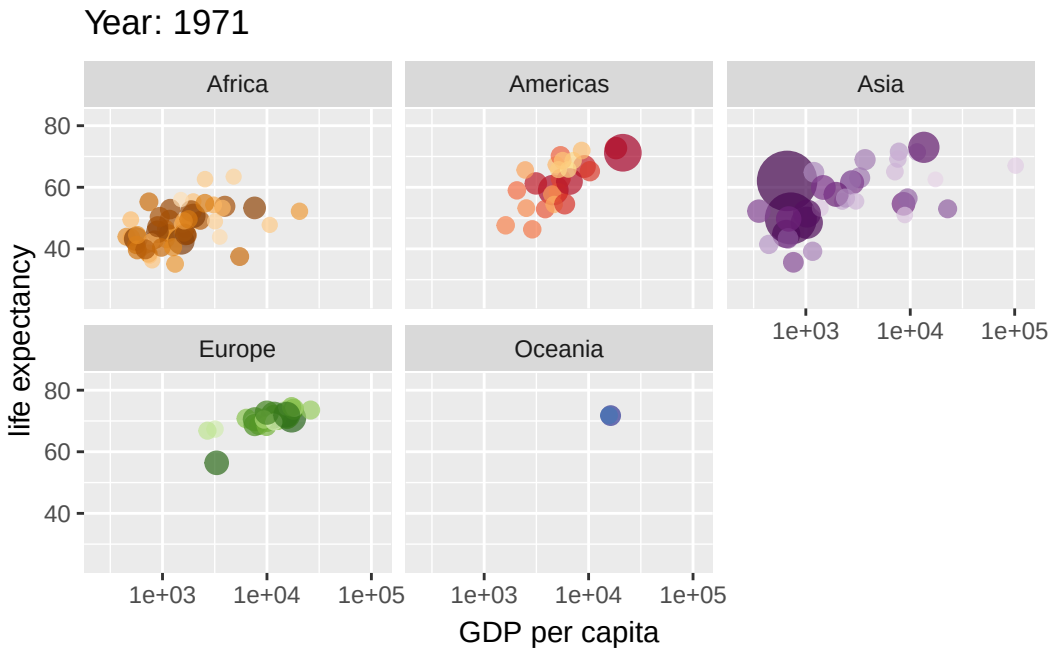

Year: 1968
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

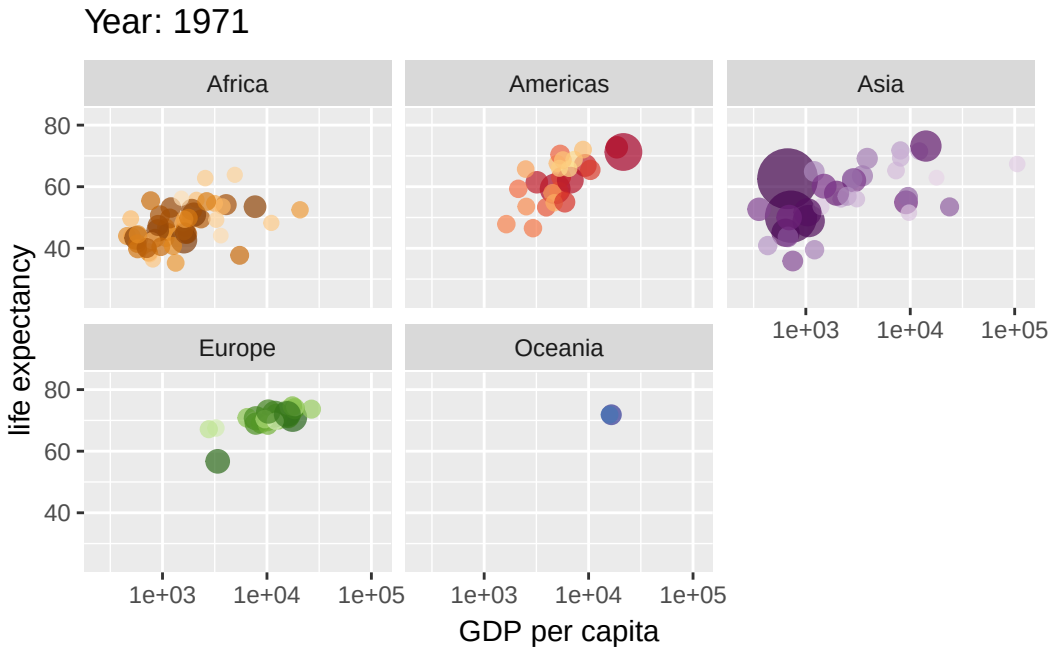

Year: 1969
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

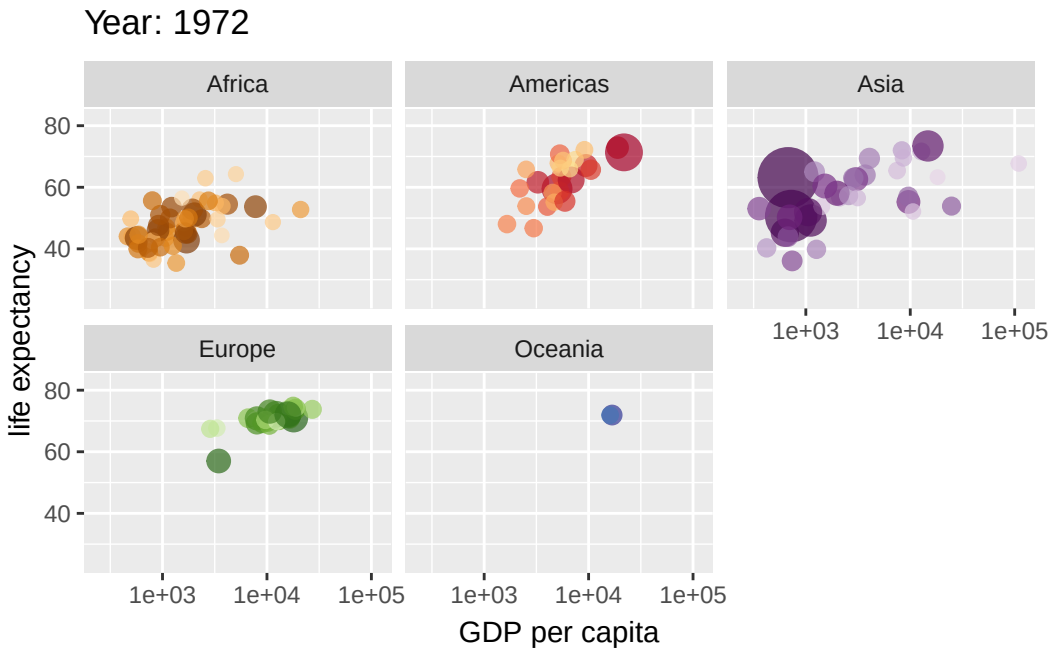

Year: 1969
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

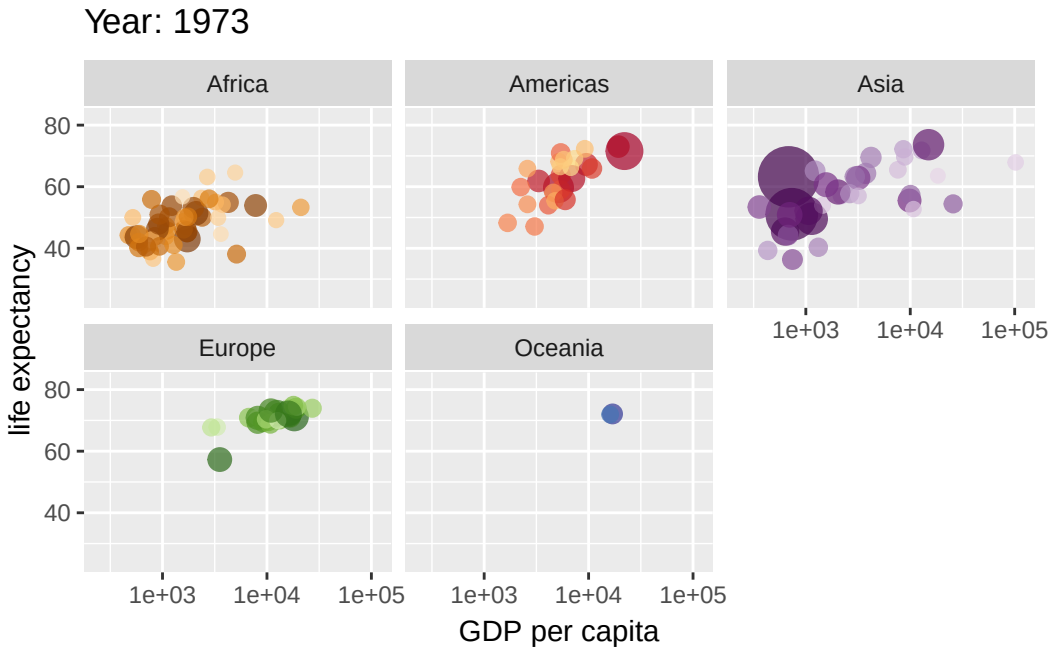

Year: 1970
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

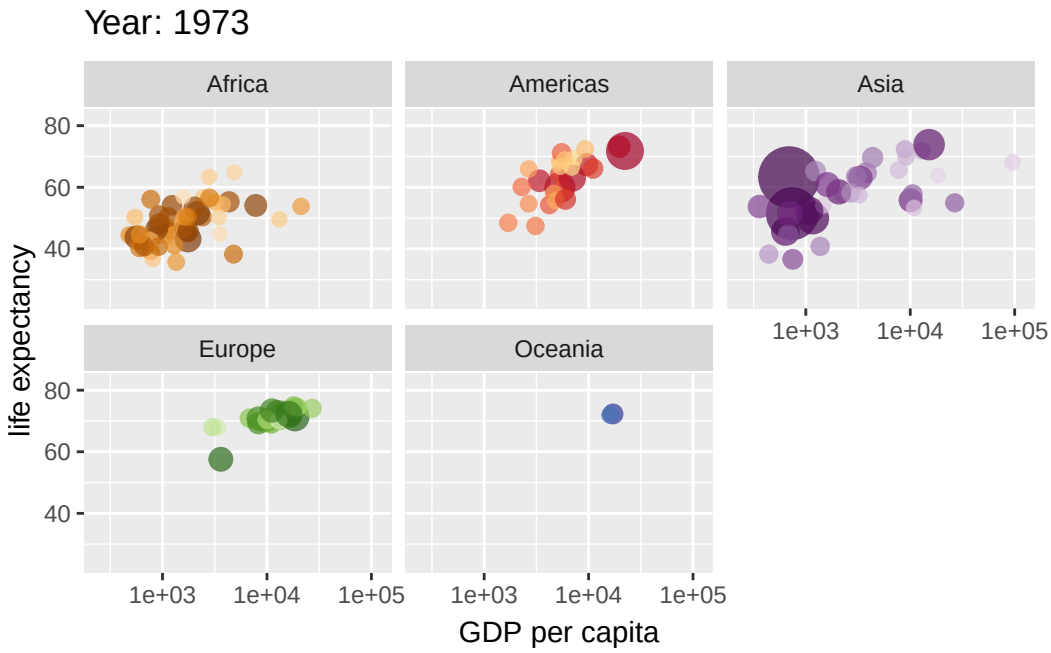

Year: 1970
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

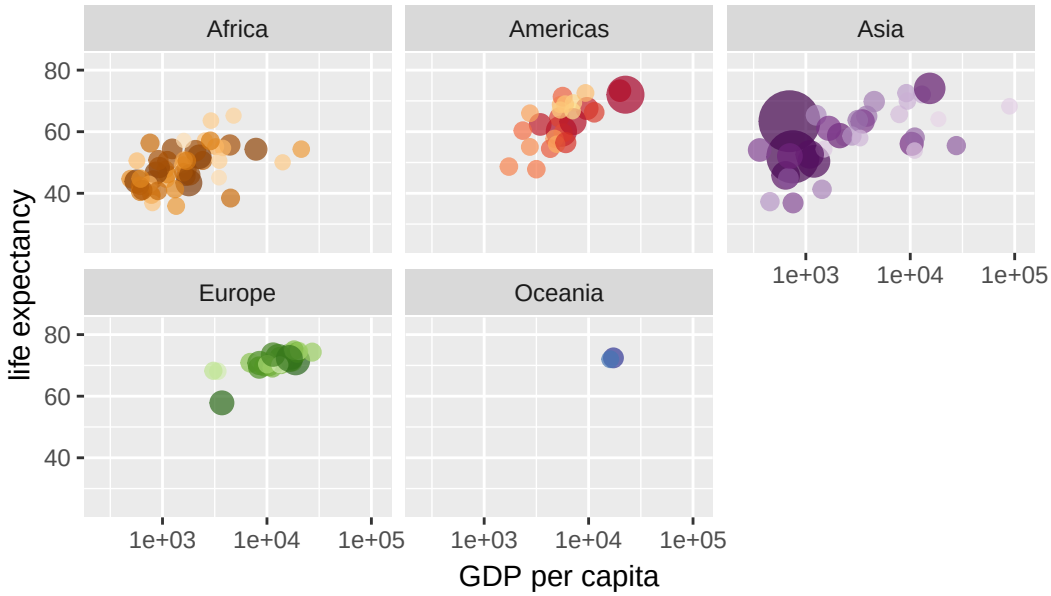

Year: 1971
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita


Year: 1971
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

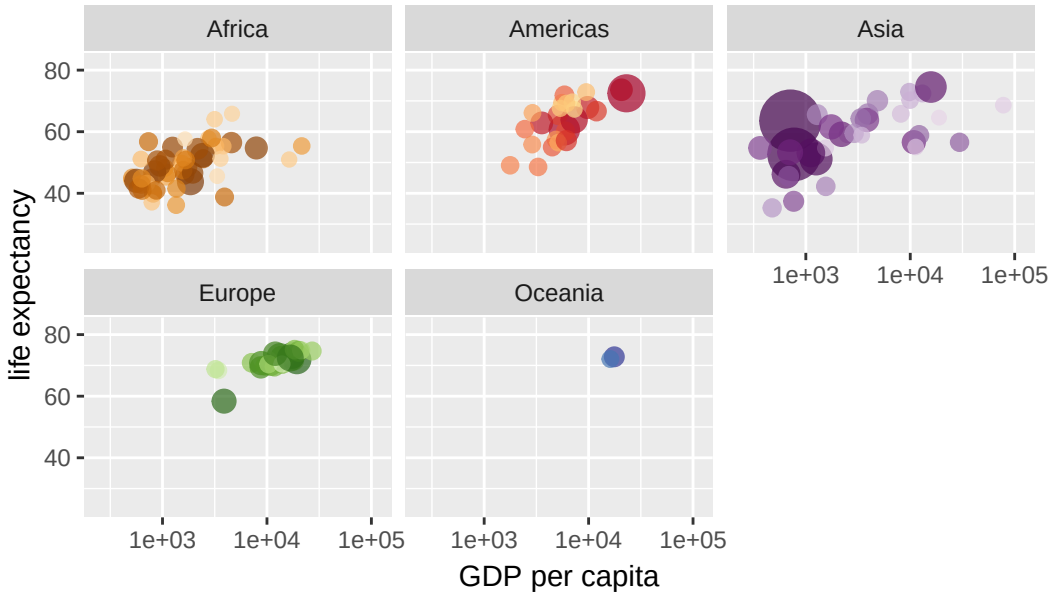

Year: 1972
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

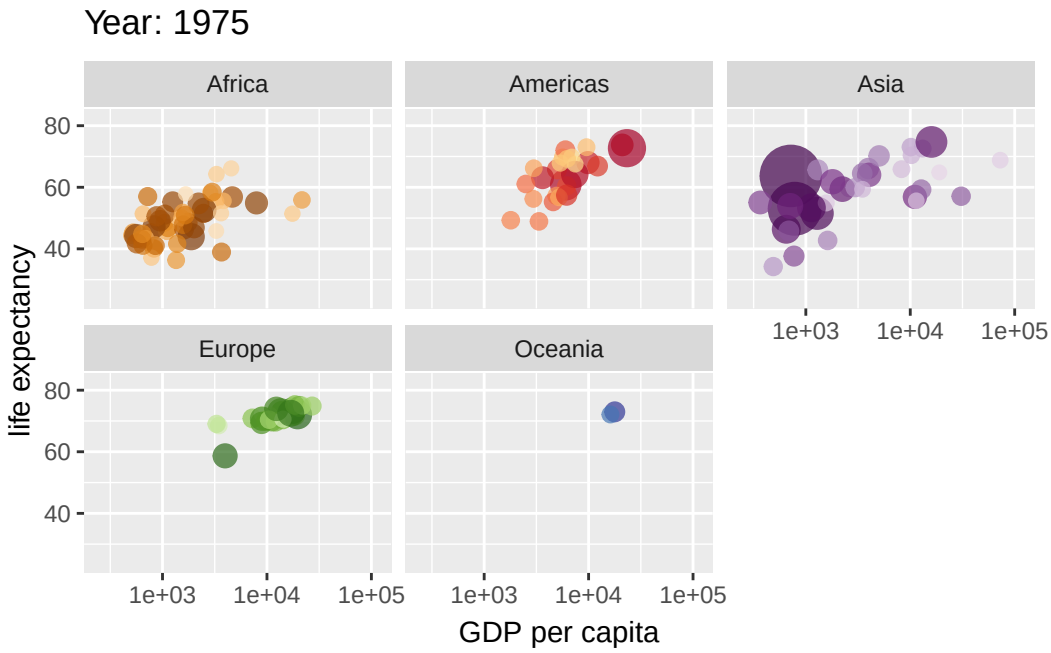

Year: 1973
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

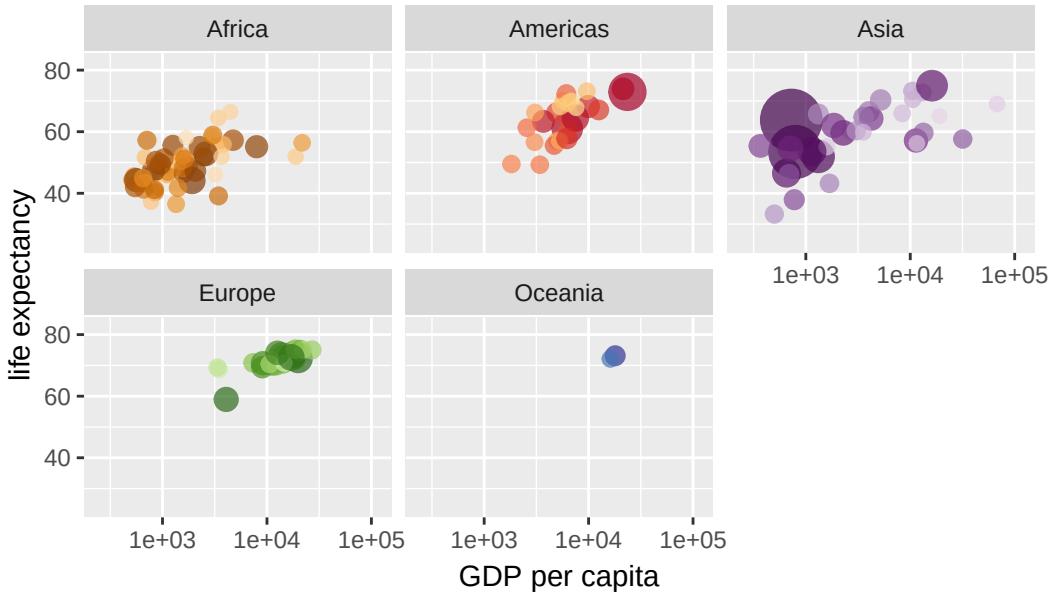

Year: 1973
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

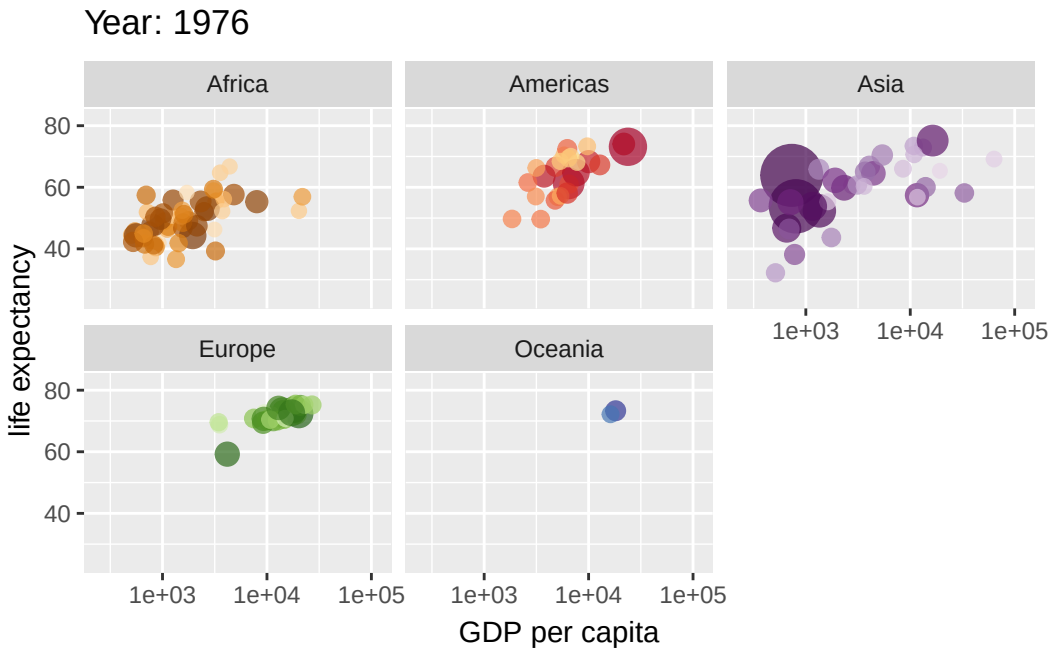

Year: 1974
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

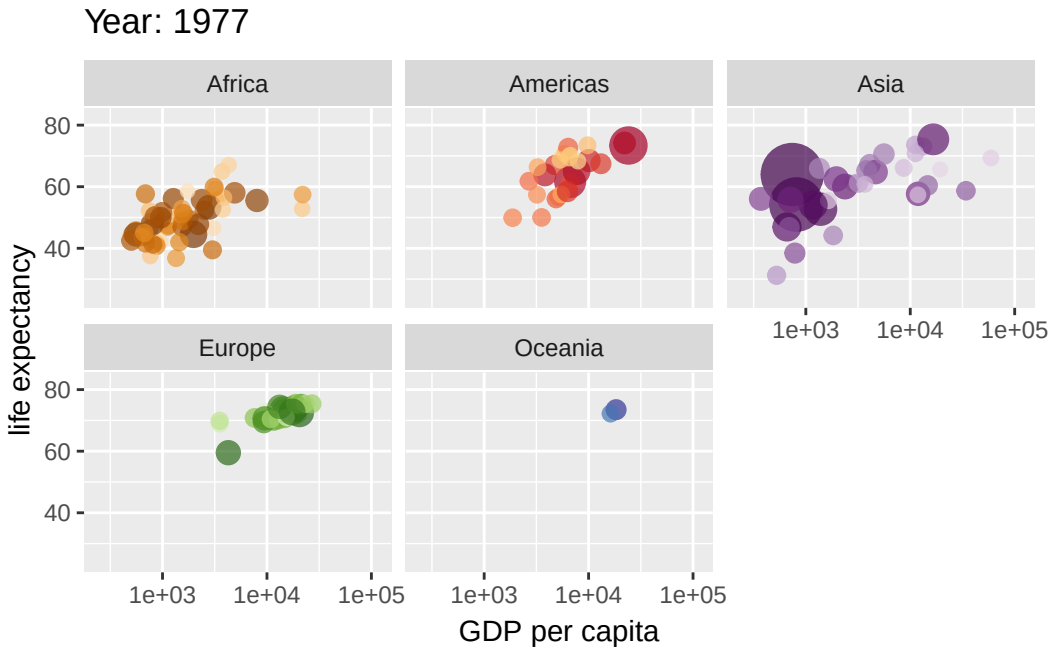

Year: 1974
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

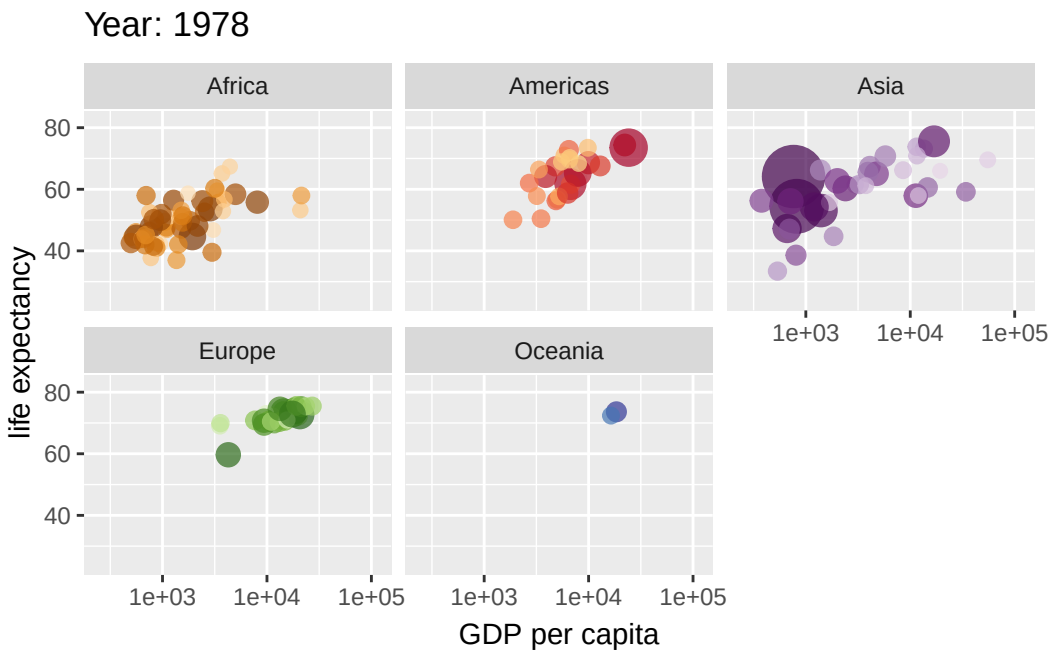

Year: 1975
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

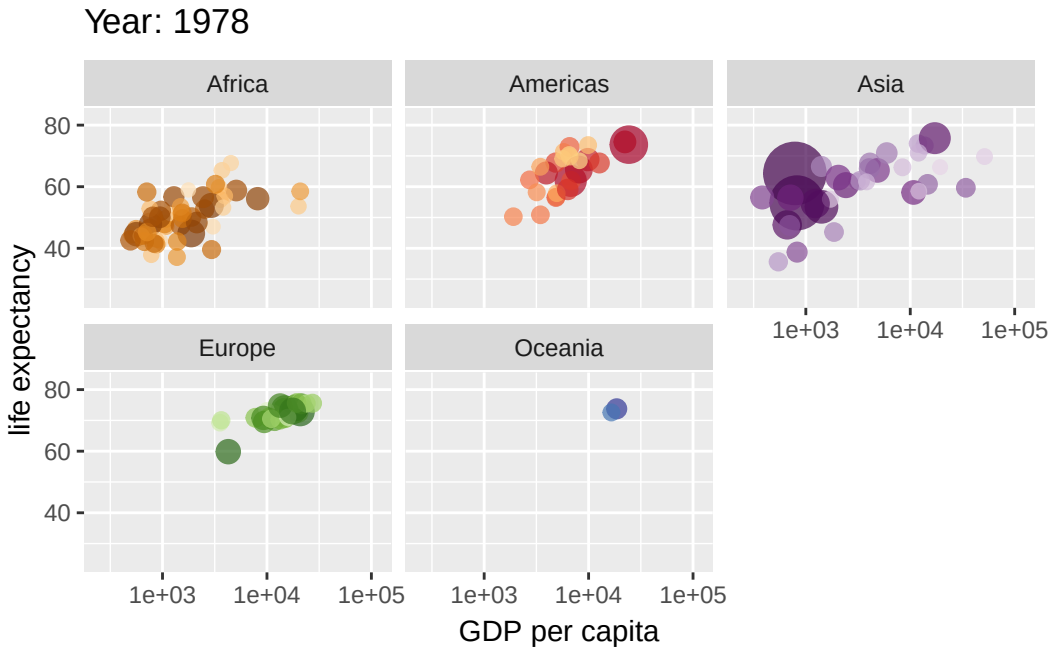

Year: 1975
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

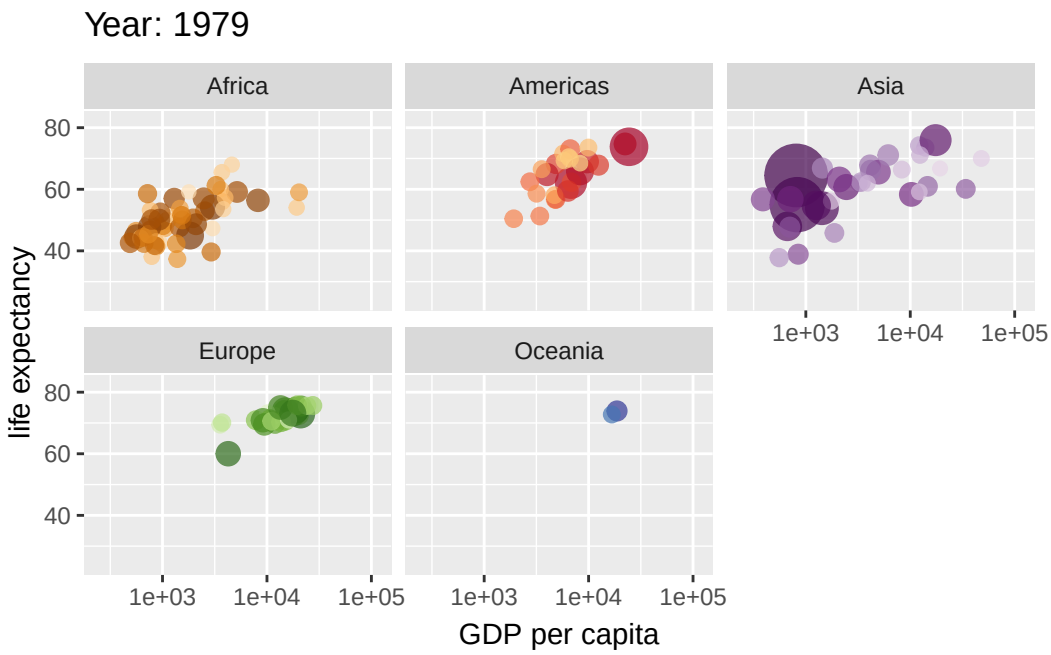

Year: 1976
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

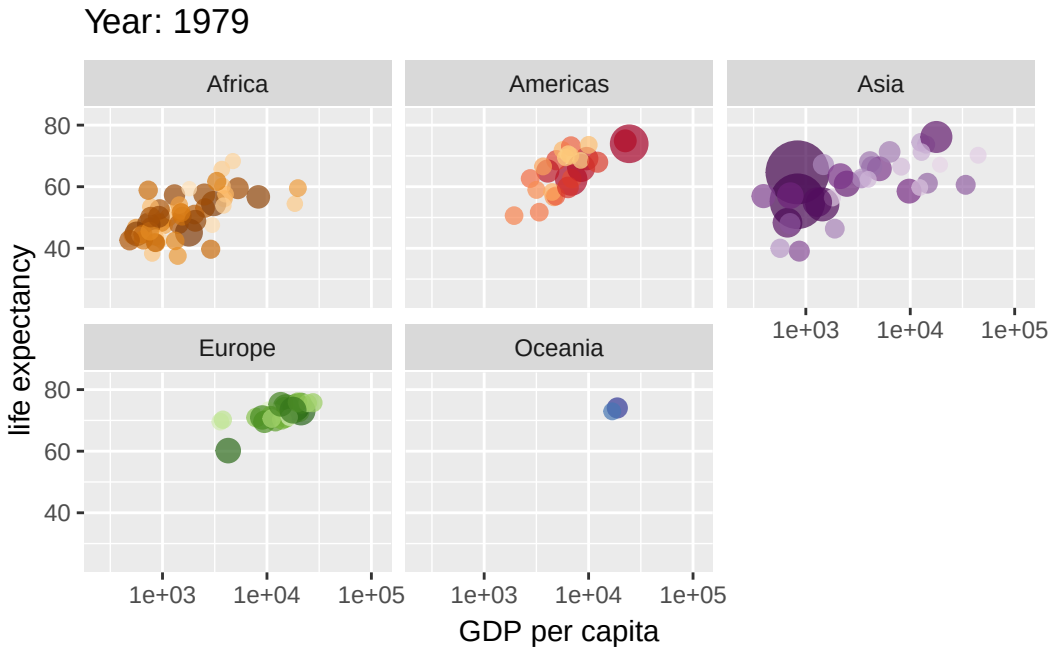

Year: 1976
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

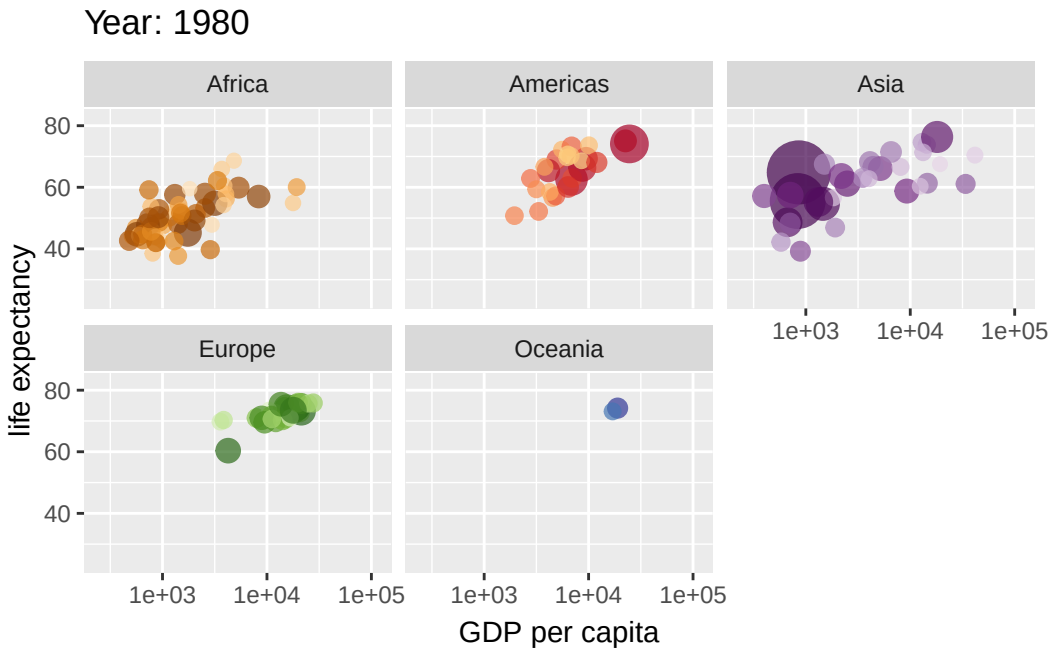

Year: 1977
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

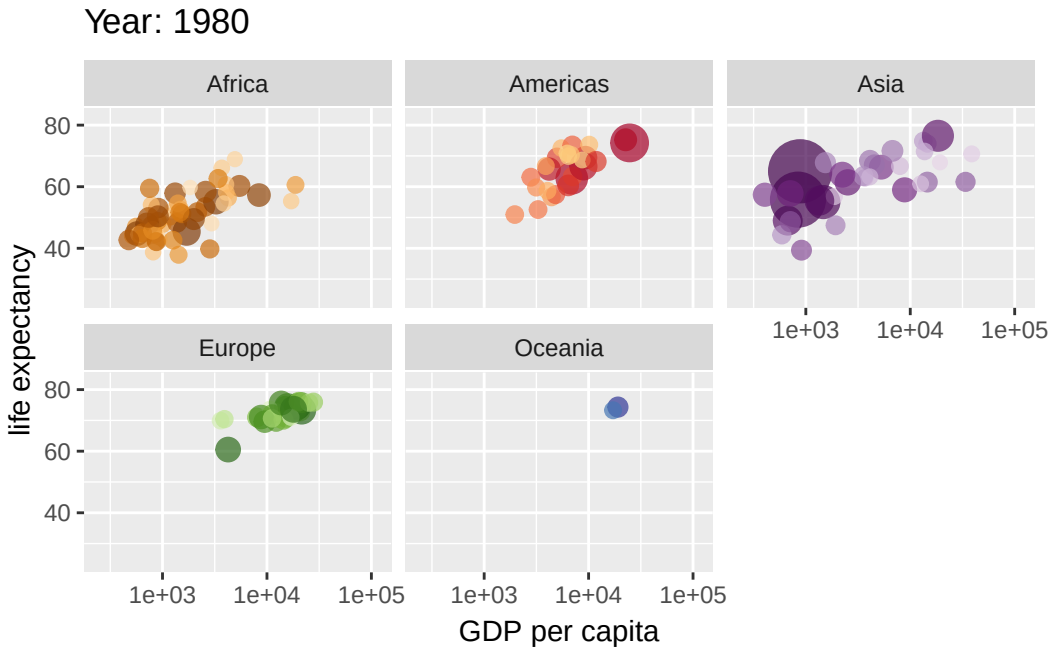

Year: 1978
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

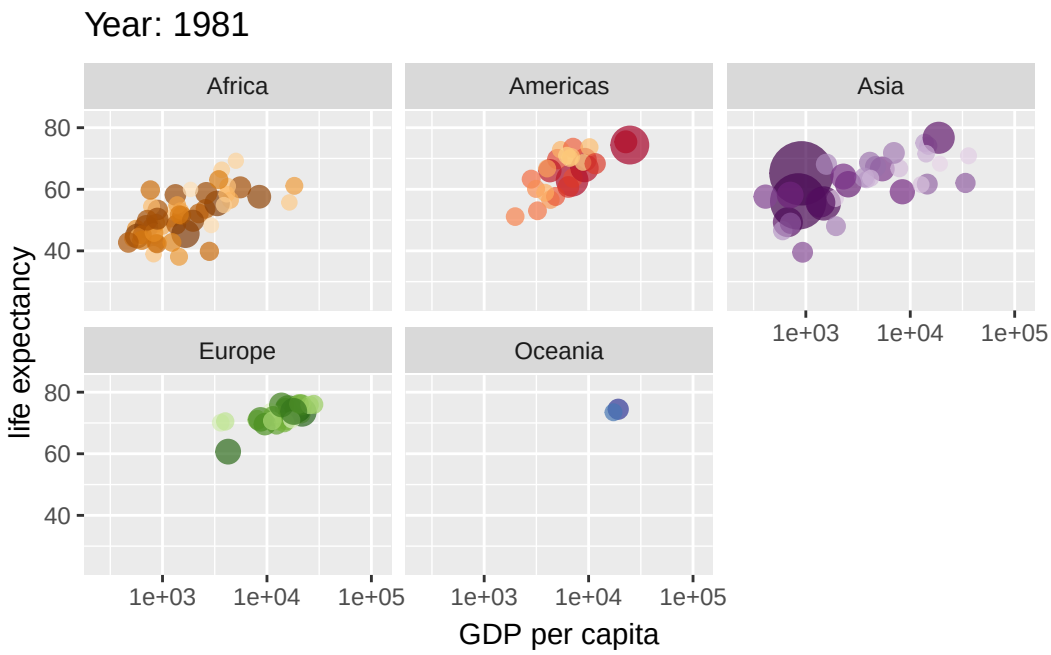

Year: 1978
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

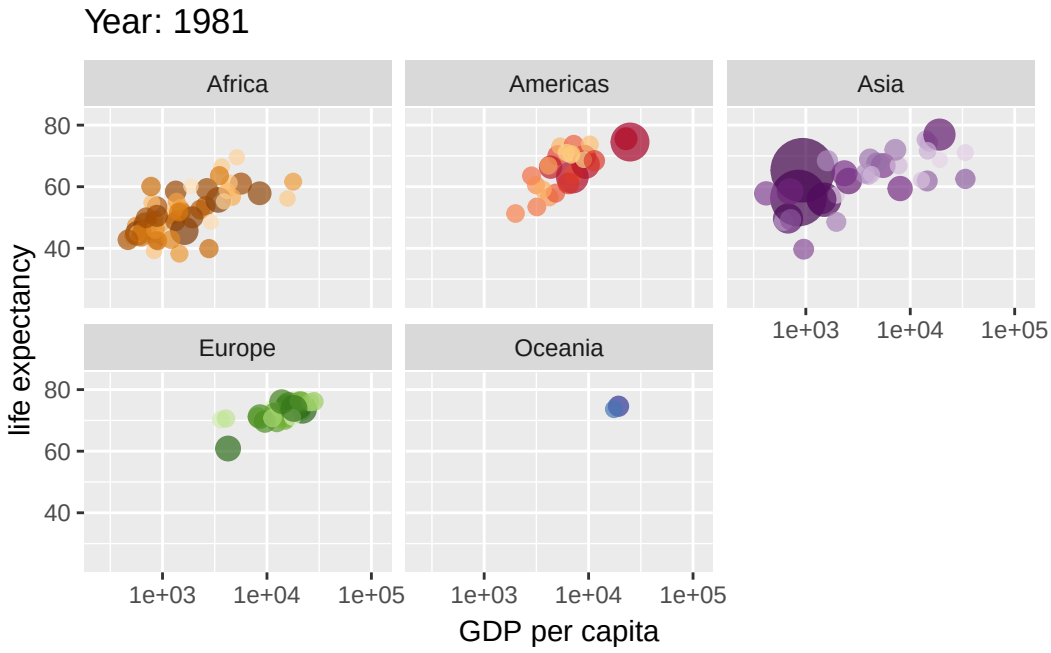

Year: 1979
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

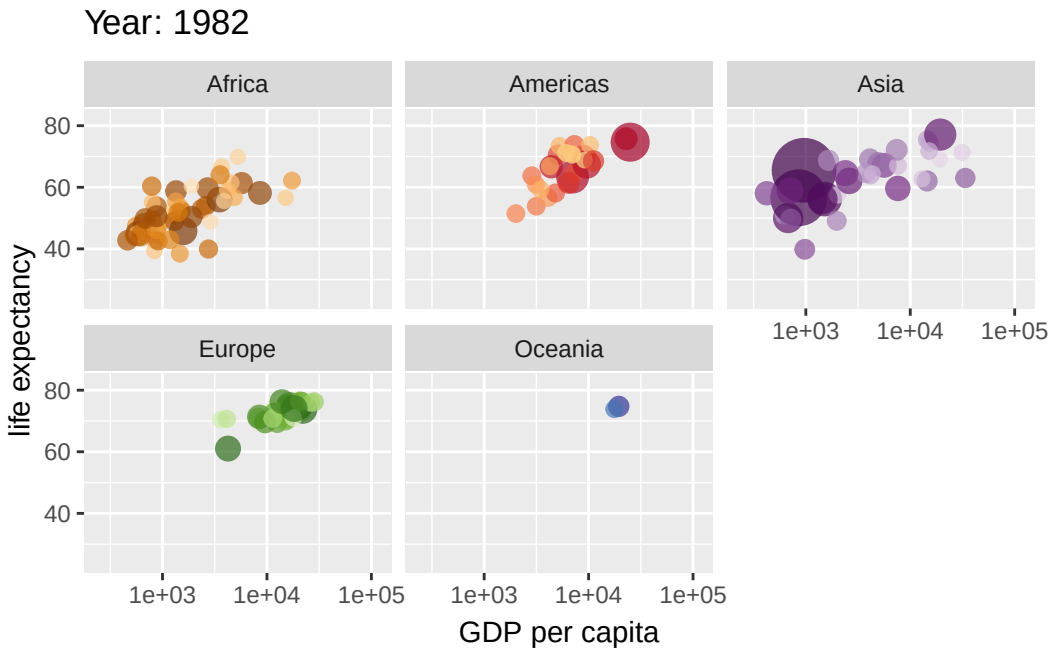

Year: 1979
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

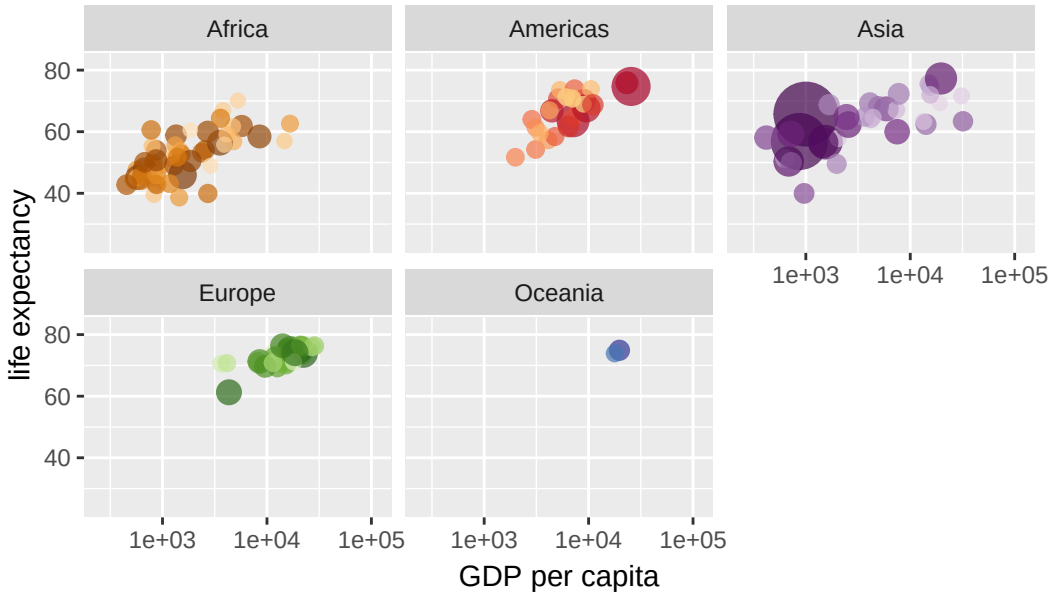

Year: 1980
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

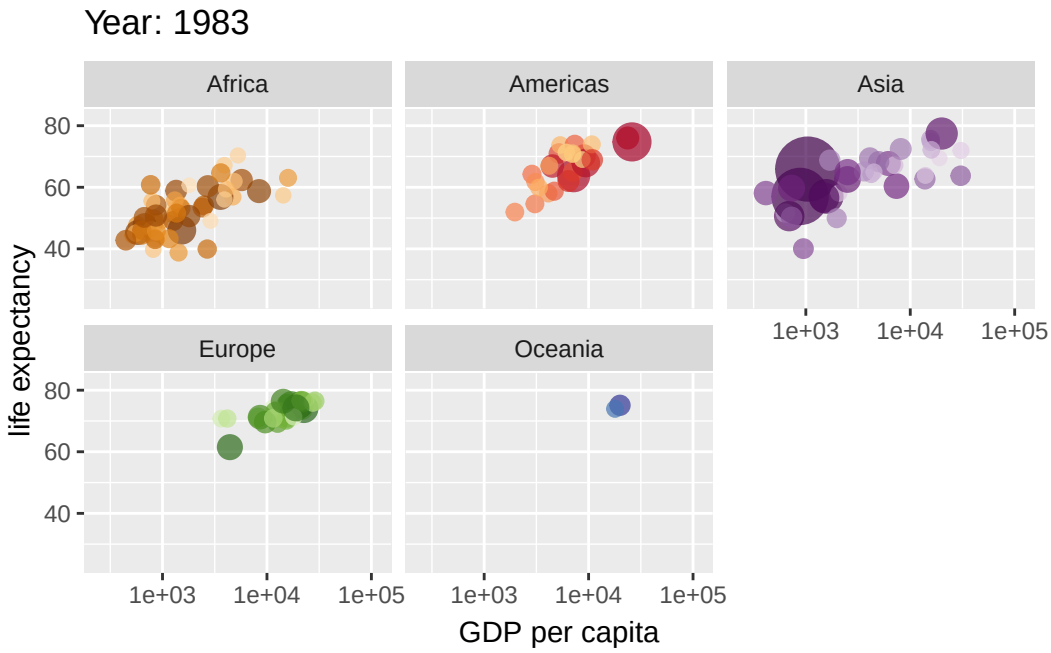

Year: 1980
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

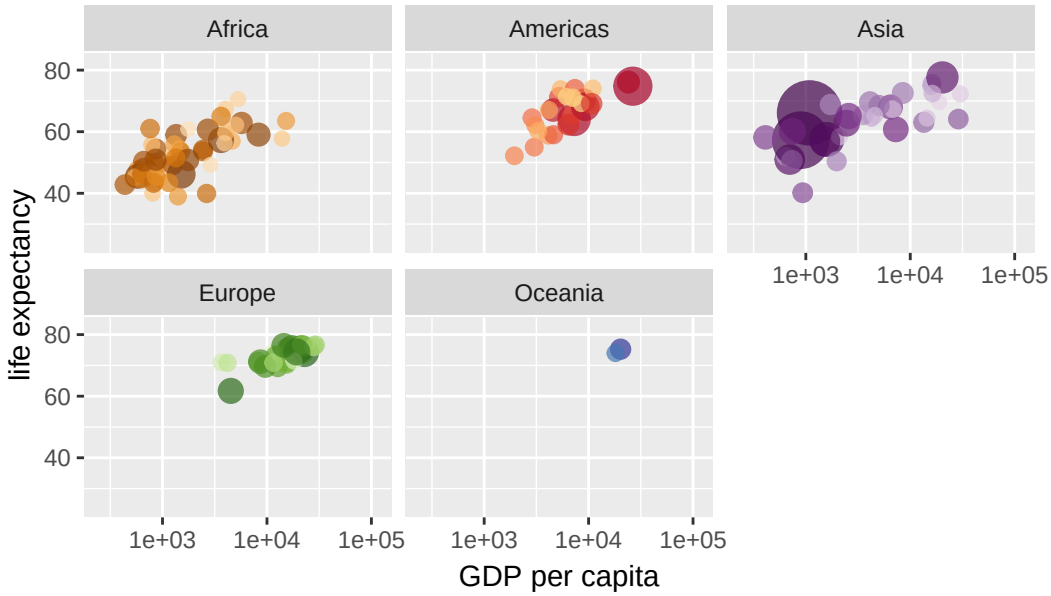

Year: 1981
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

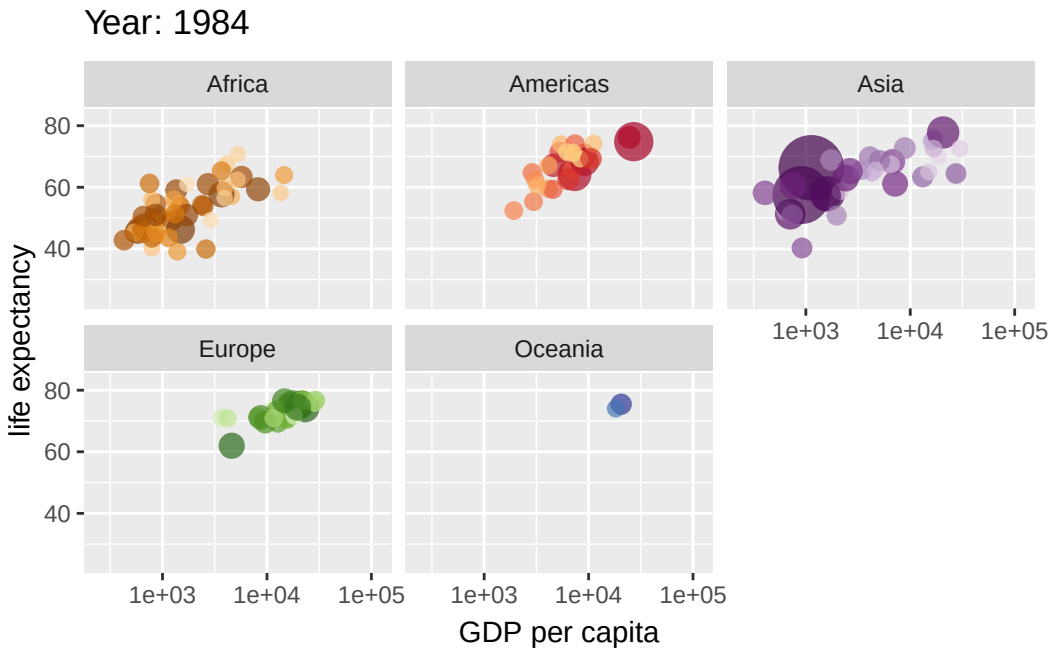

Year: 1981
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

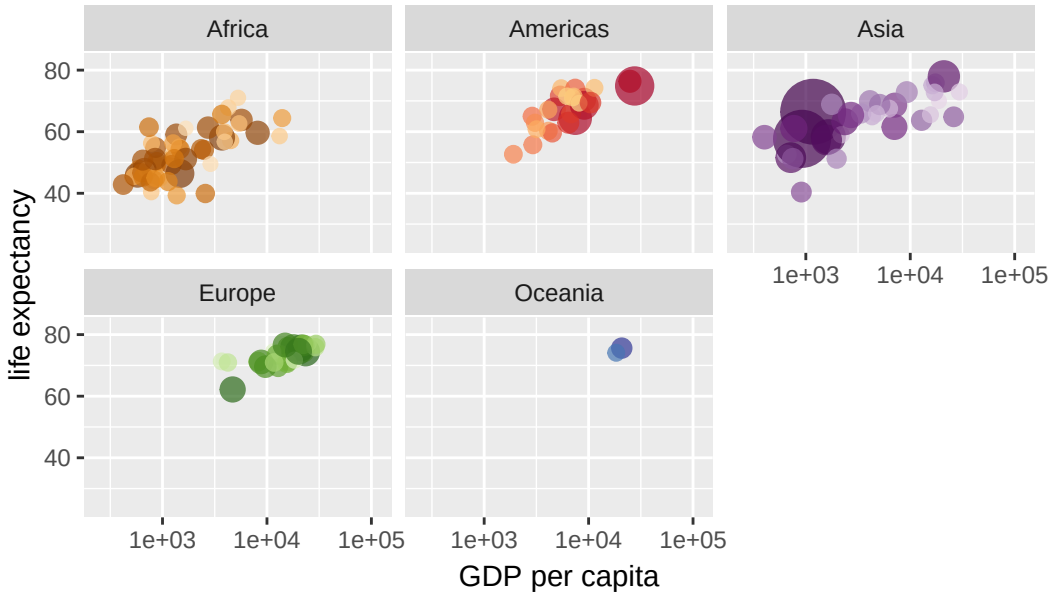

Year: 1982
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

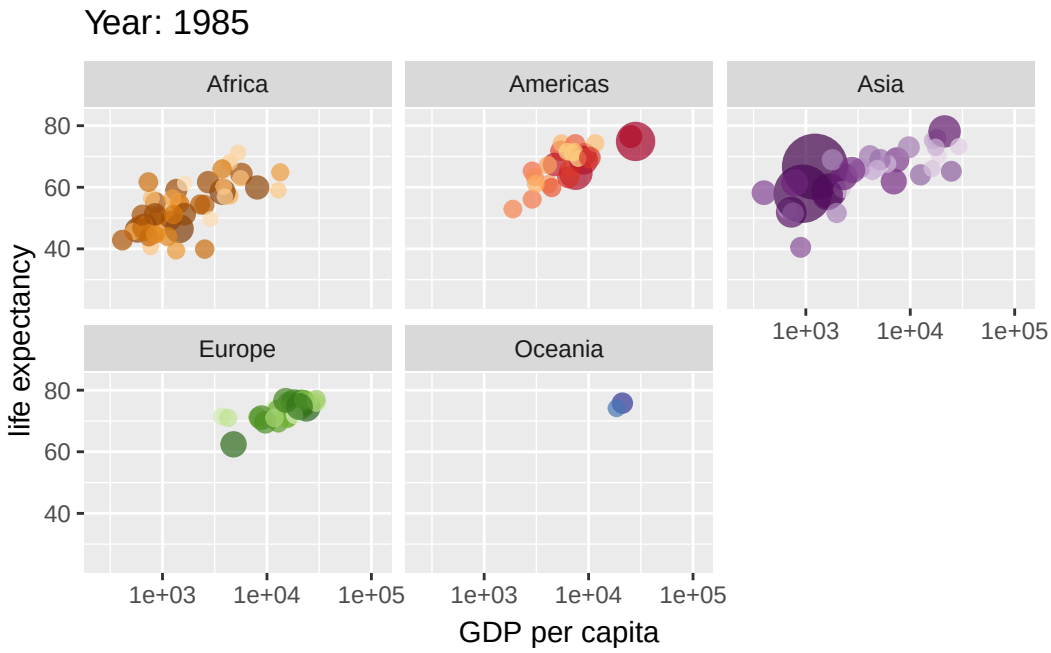

Year: 1983
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

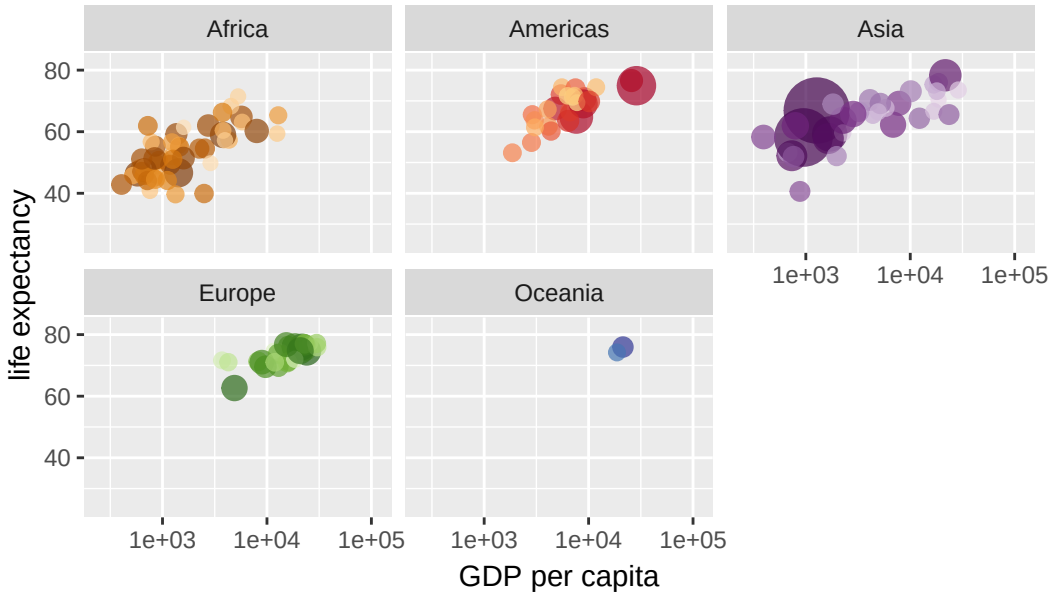

Year: 1983
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

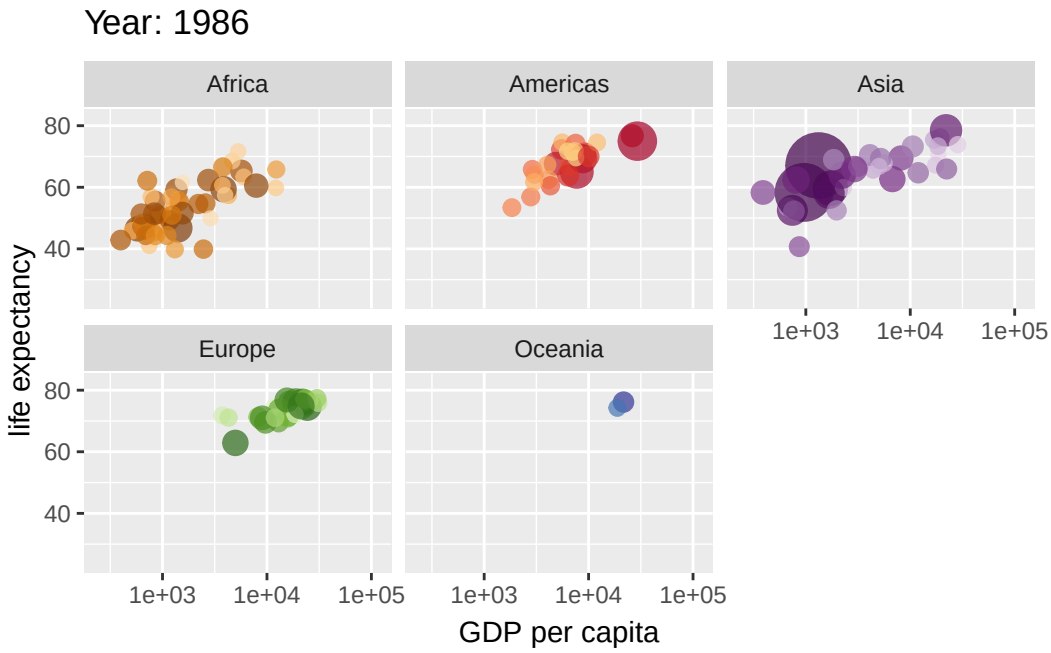

Year: 1984
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

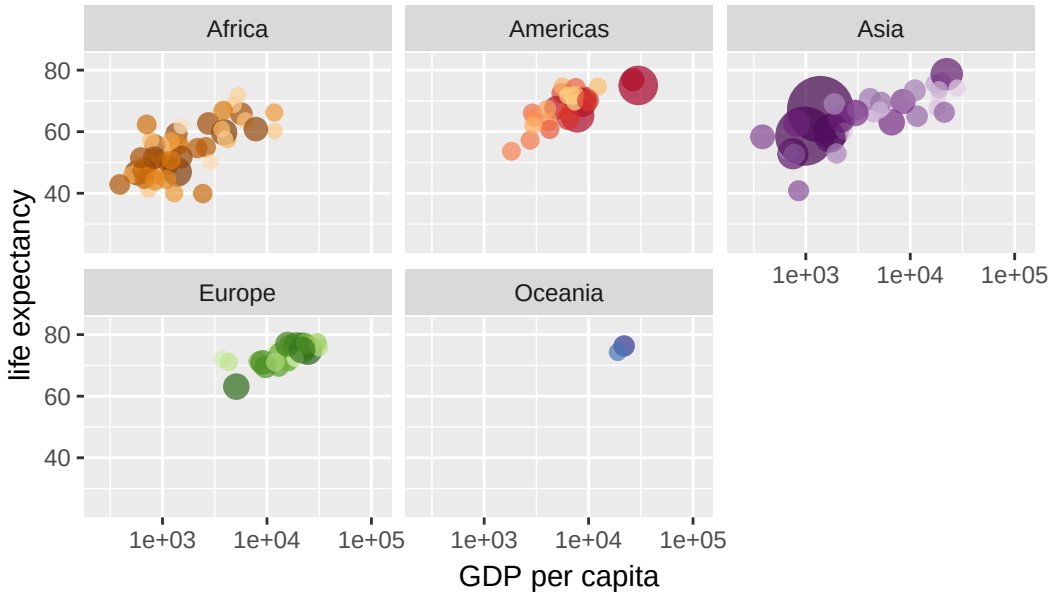

Year: 1984
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

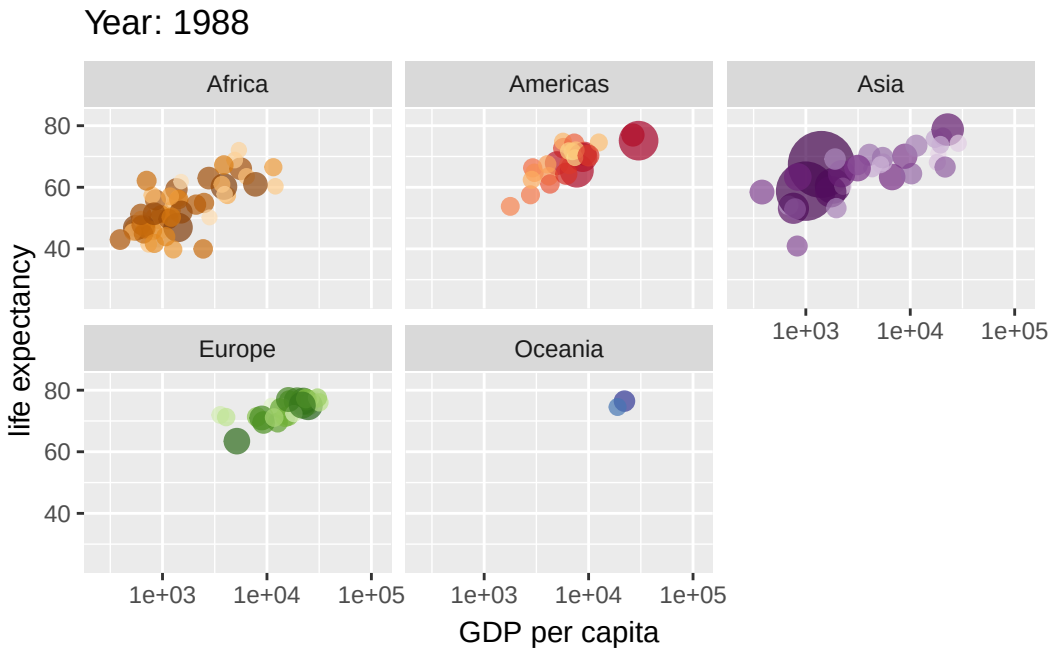

Year: 1985
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

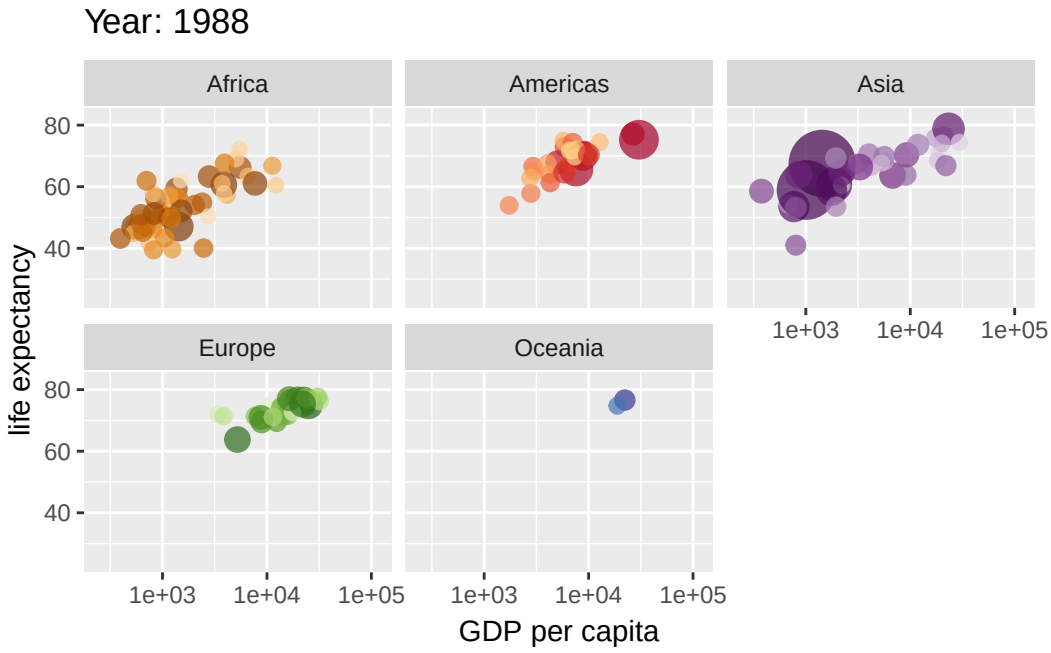

Year: 1985
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

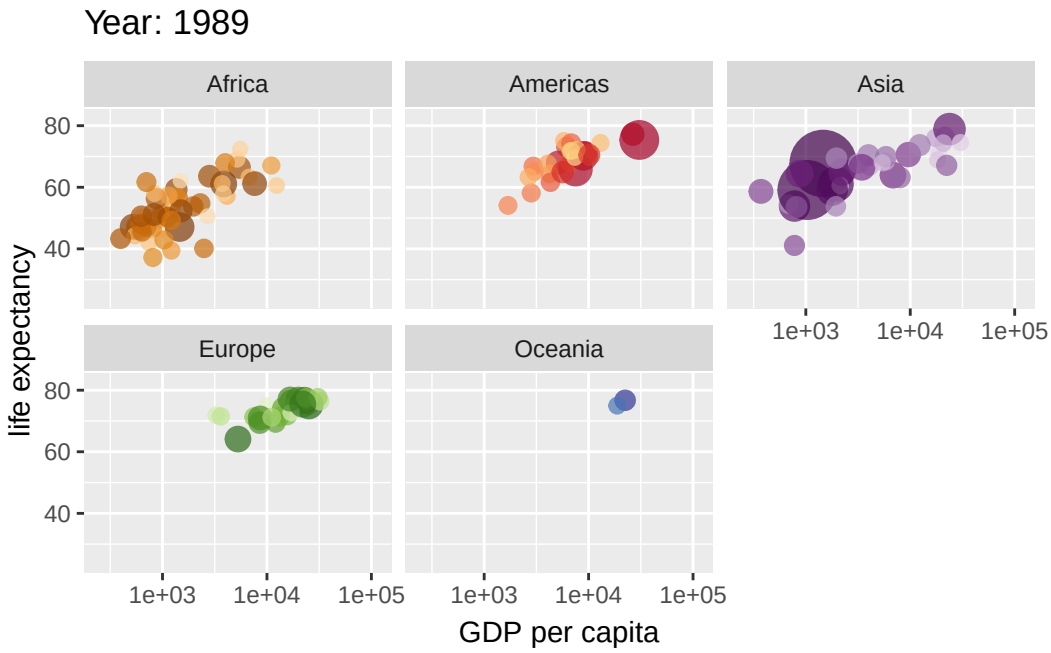

Year: 1986
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

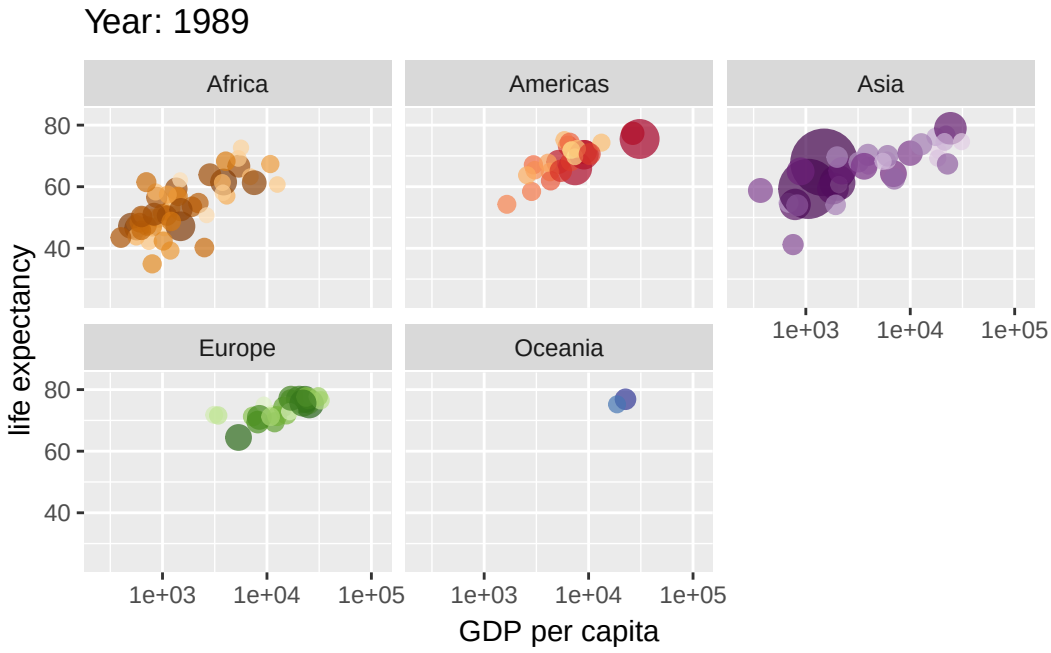

Year: 1986
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

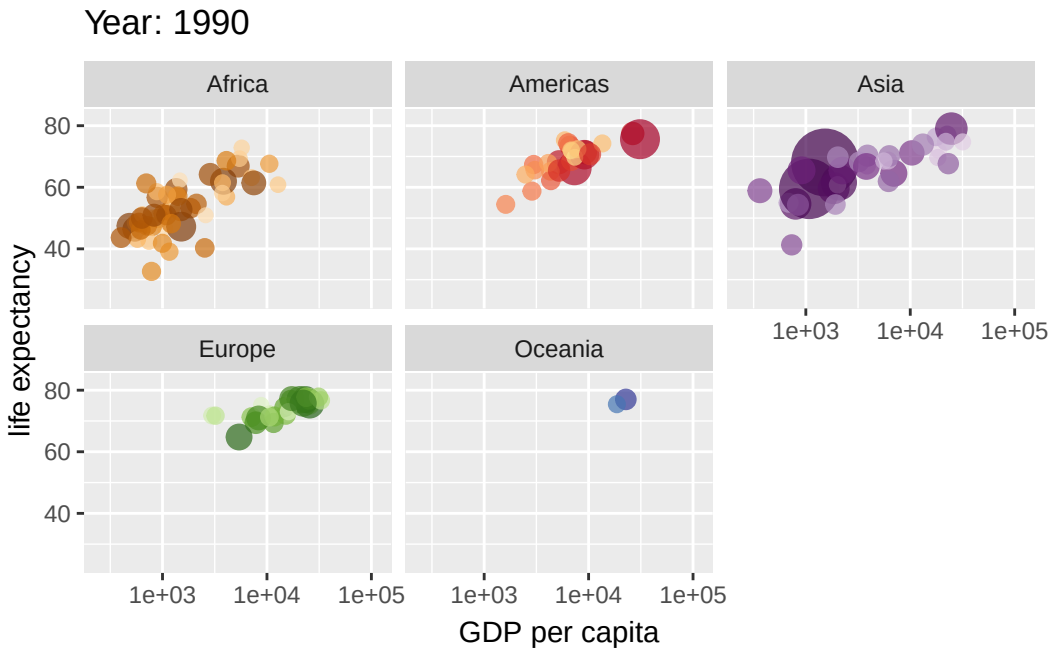

Year: 1987
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

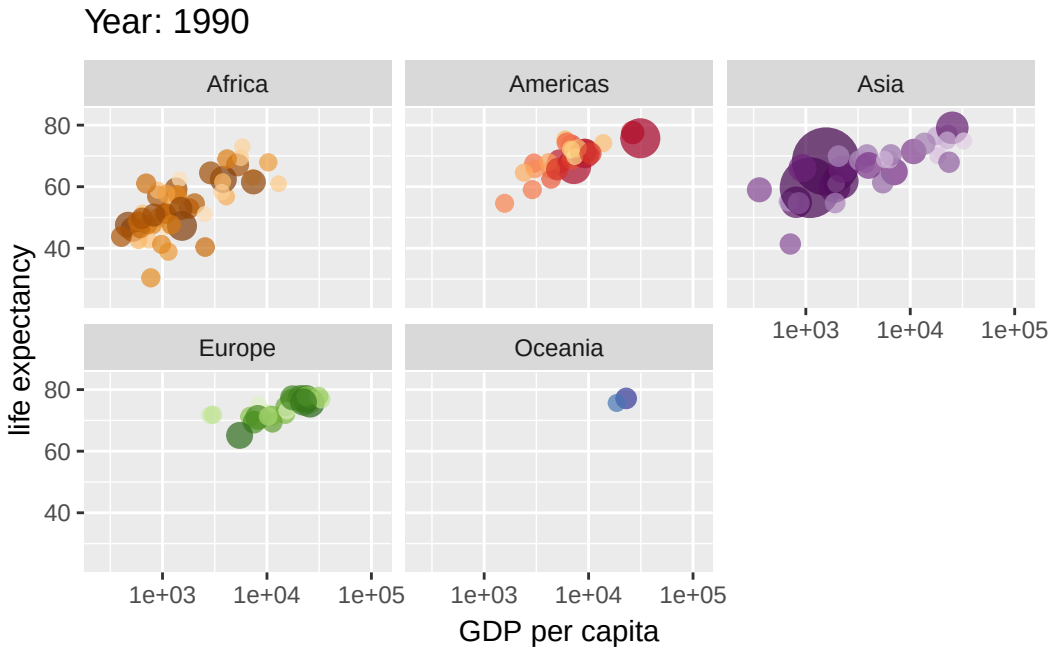

Year: 1988
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

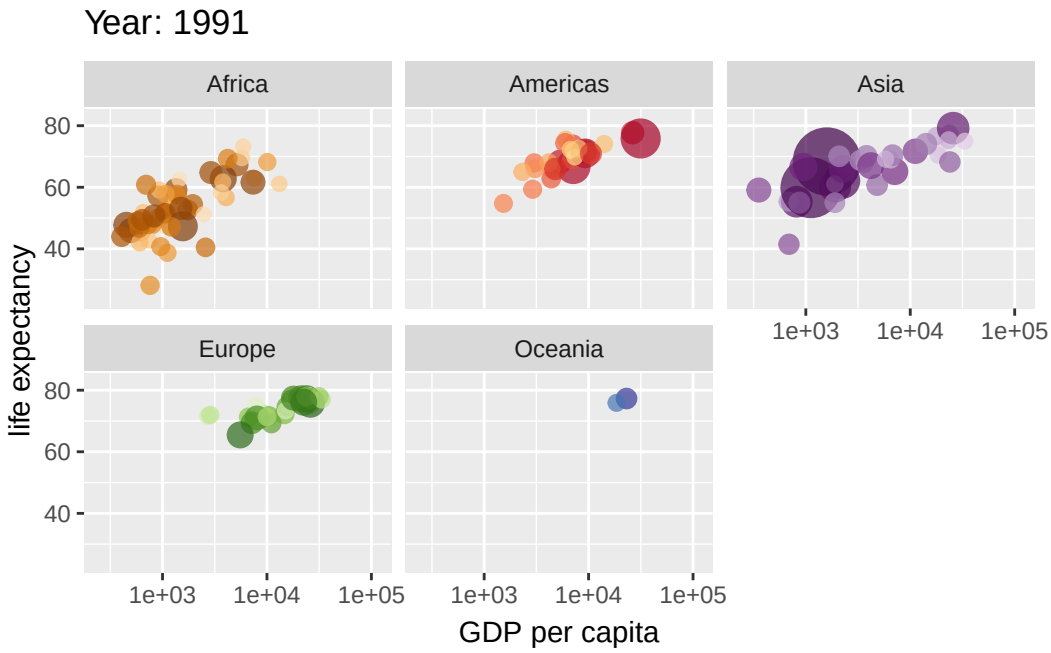

Year: 1988
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

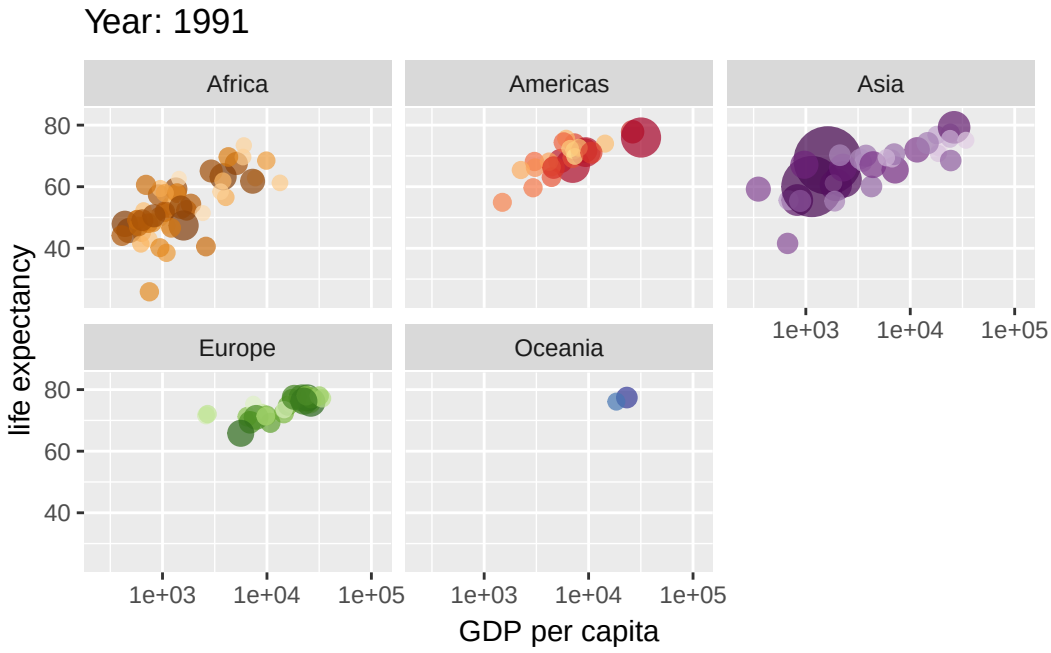

Year: 1989
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

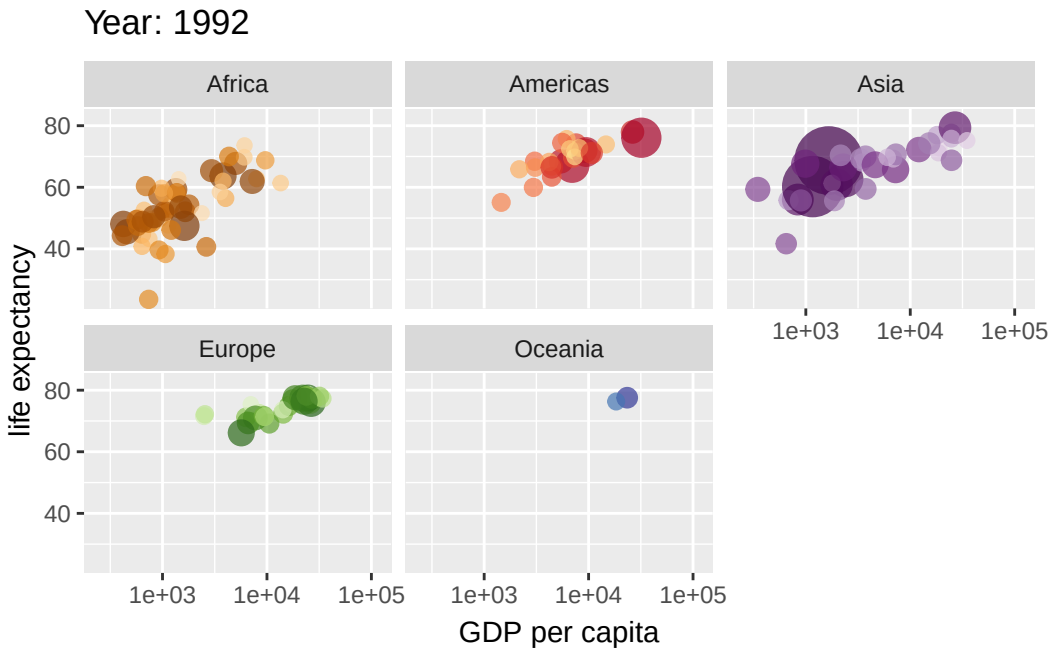

Year: 1989
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

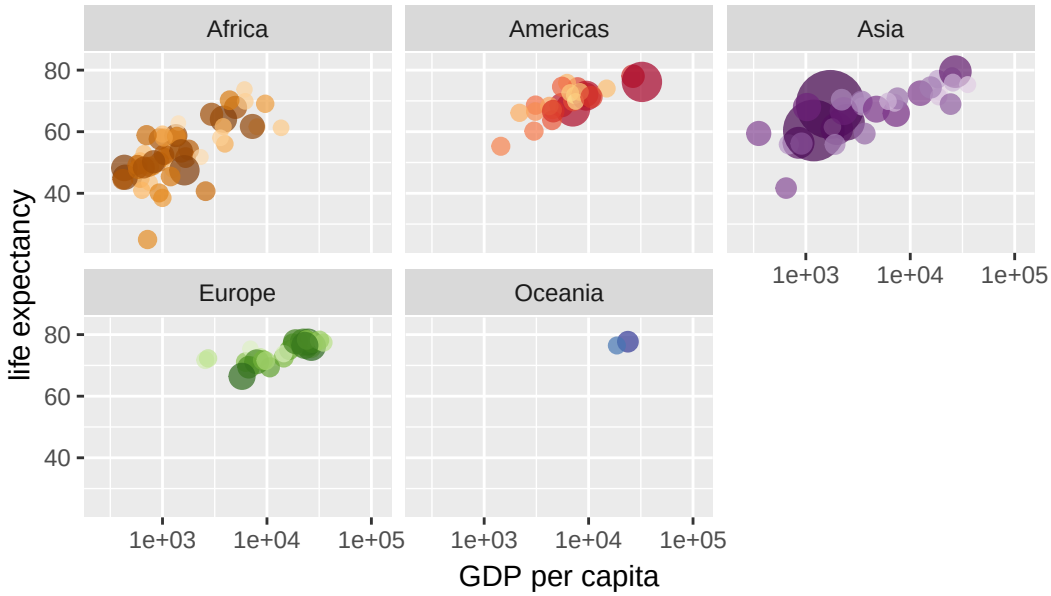

Year: 1990
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

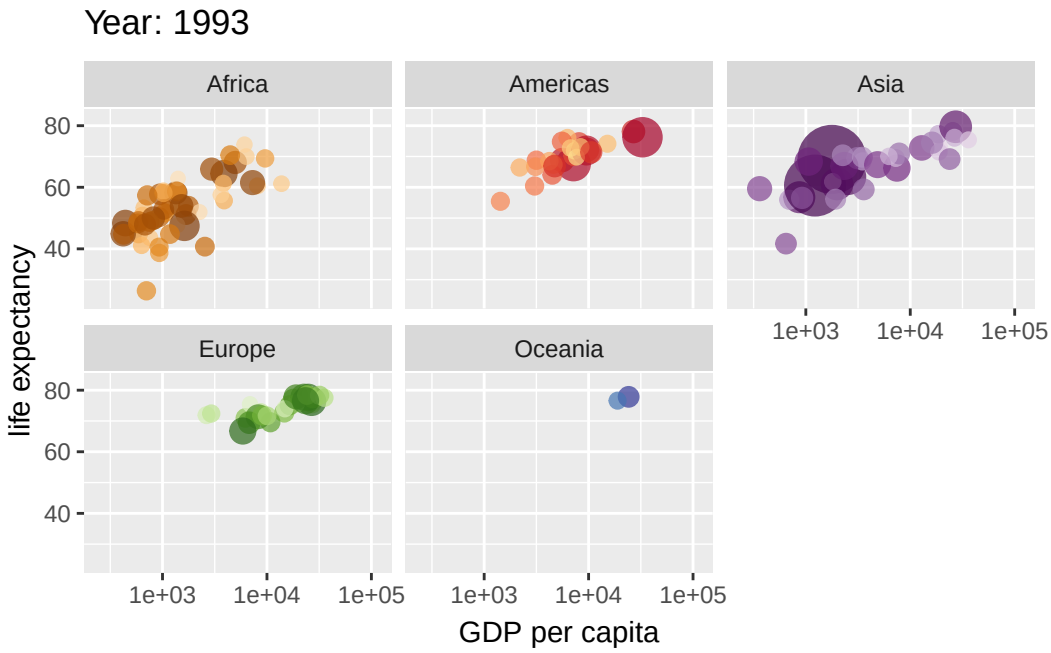

Year: 1990
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

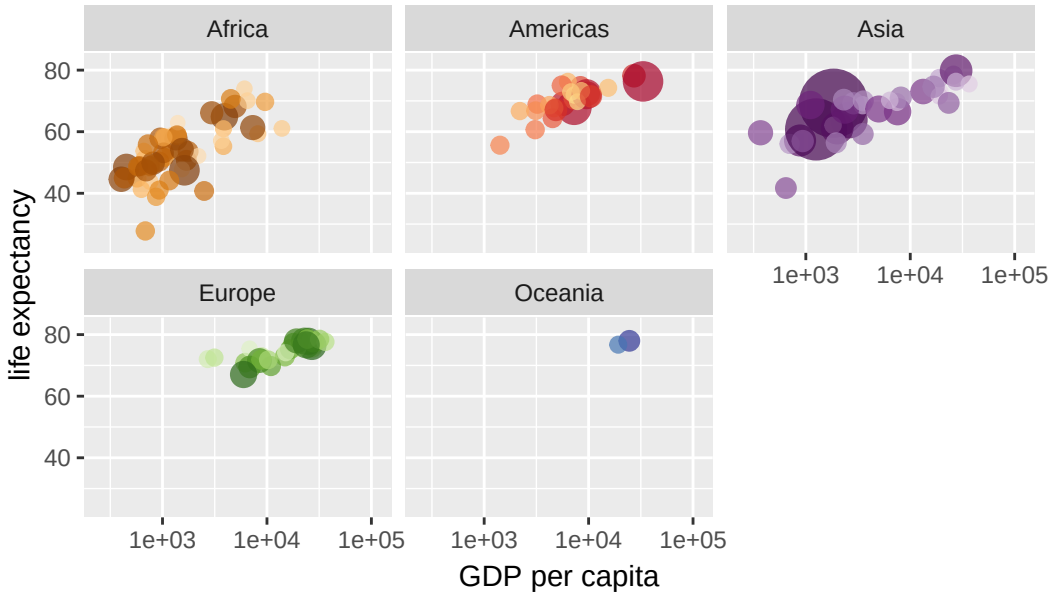

Year: 1991
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

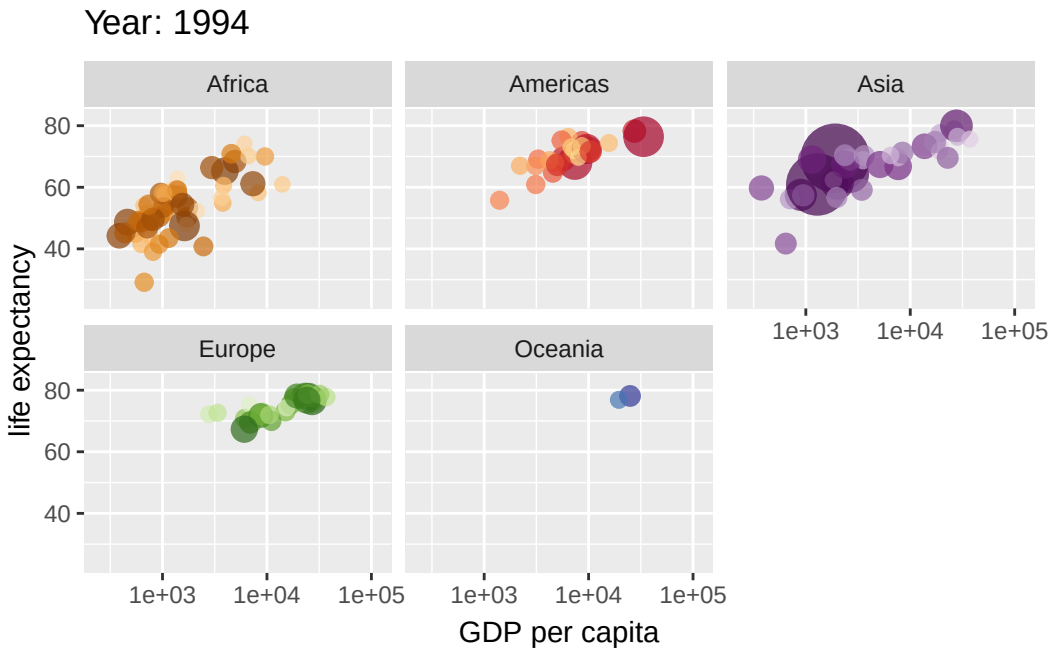

Year: 1991
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

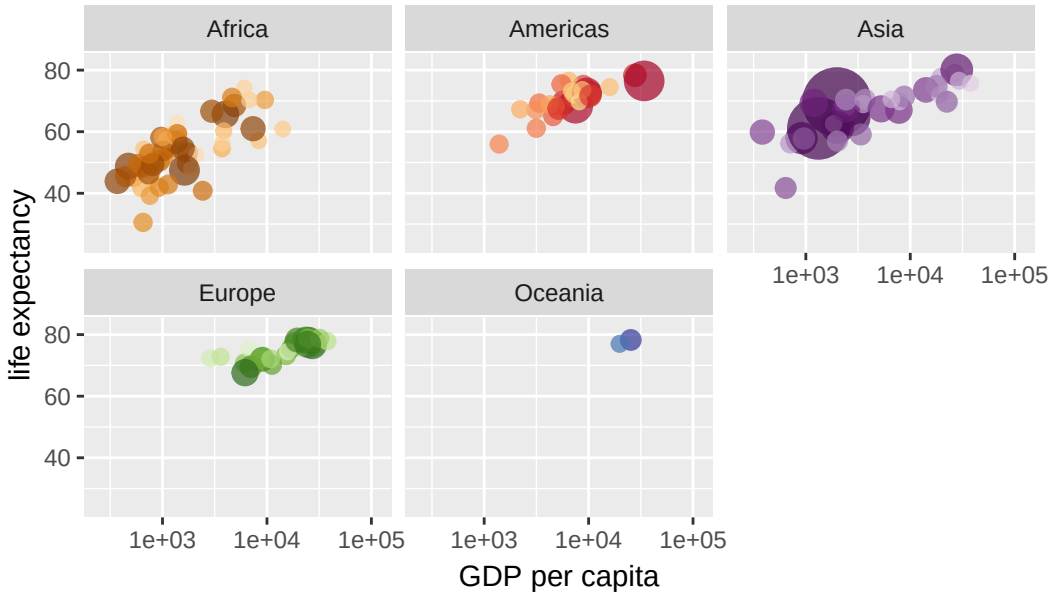

Year: 1992
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

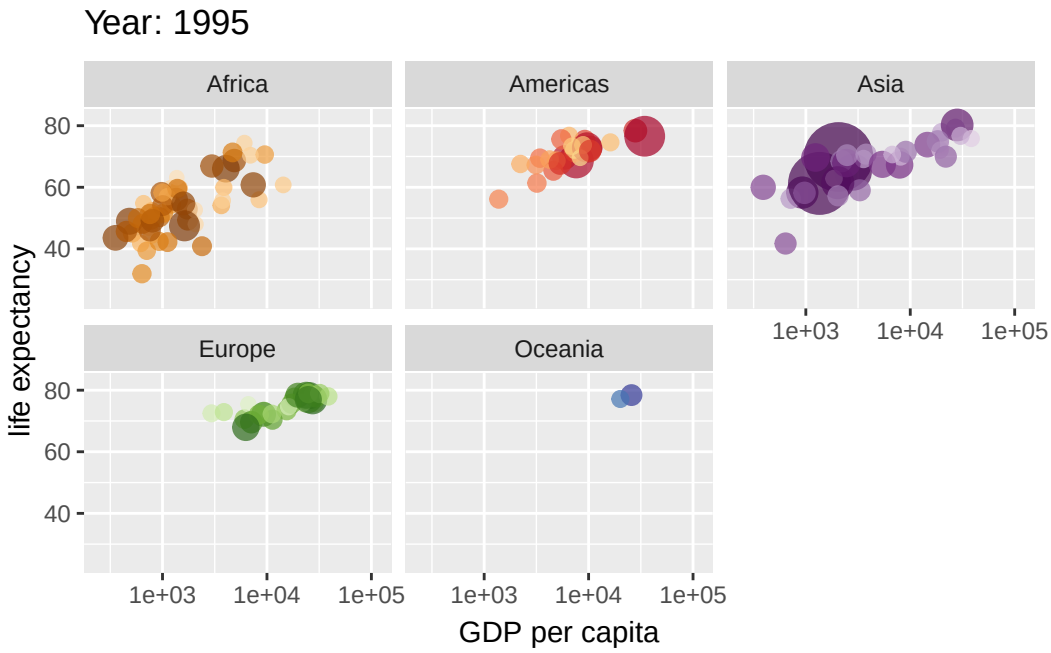

Year: 1993
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

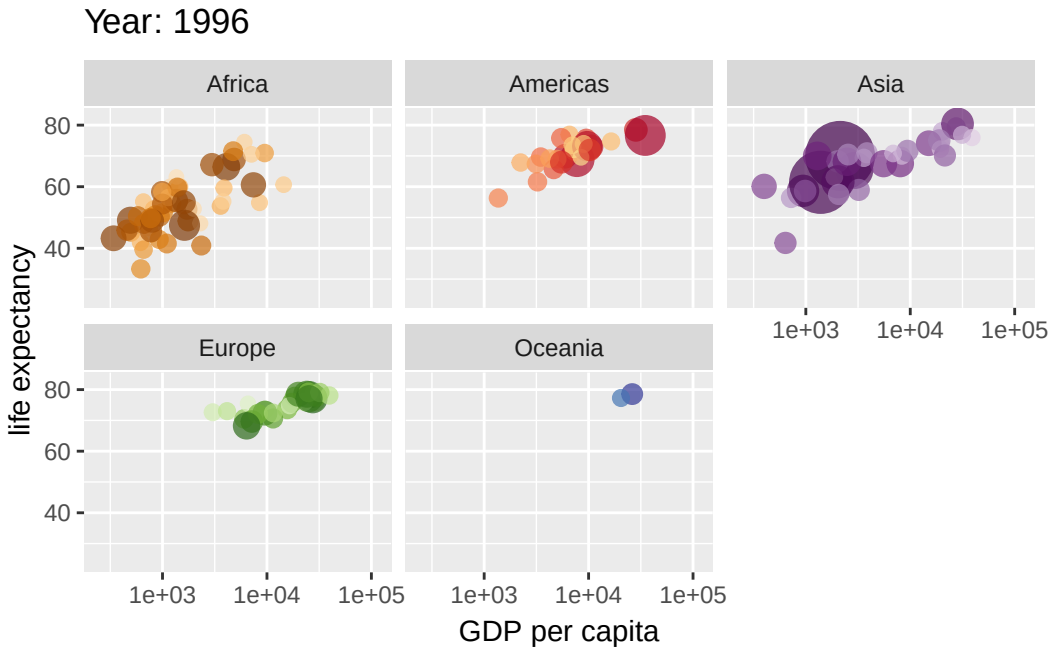

Year: 1993
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

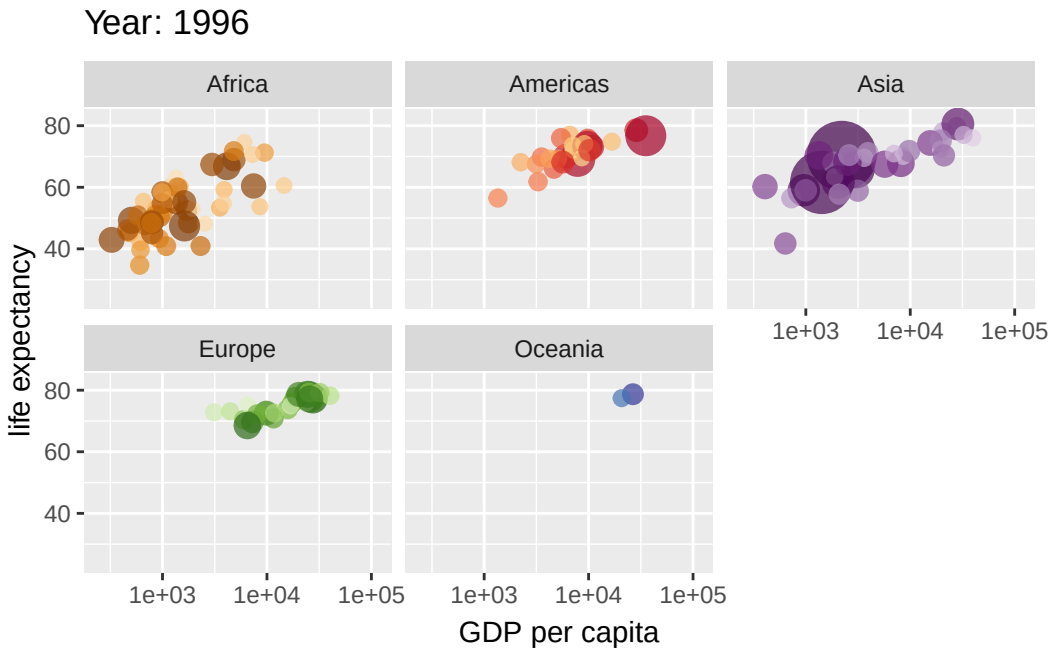

Year: 1994
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

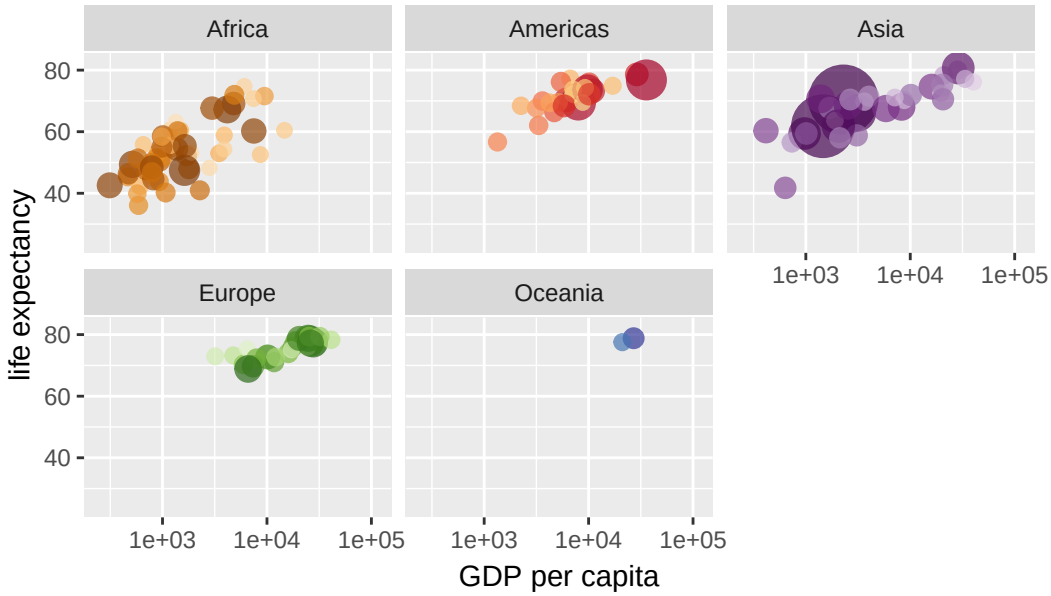

Year: 1994
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

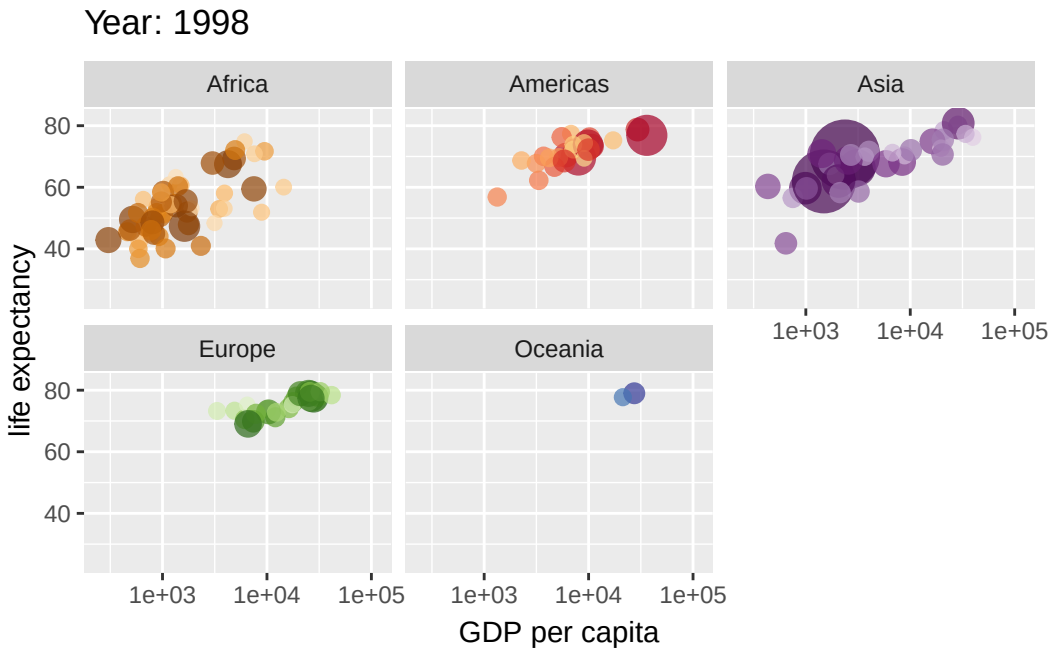

Year: 1995
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

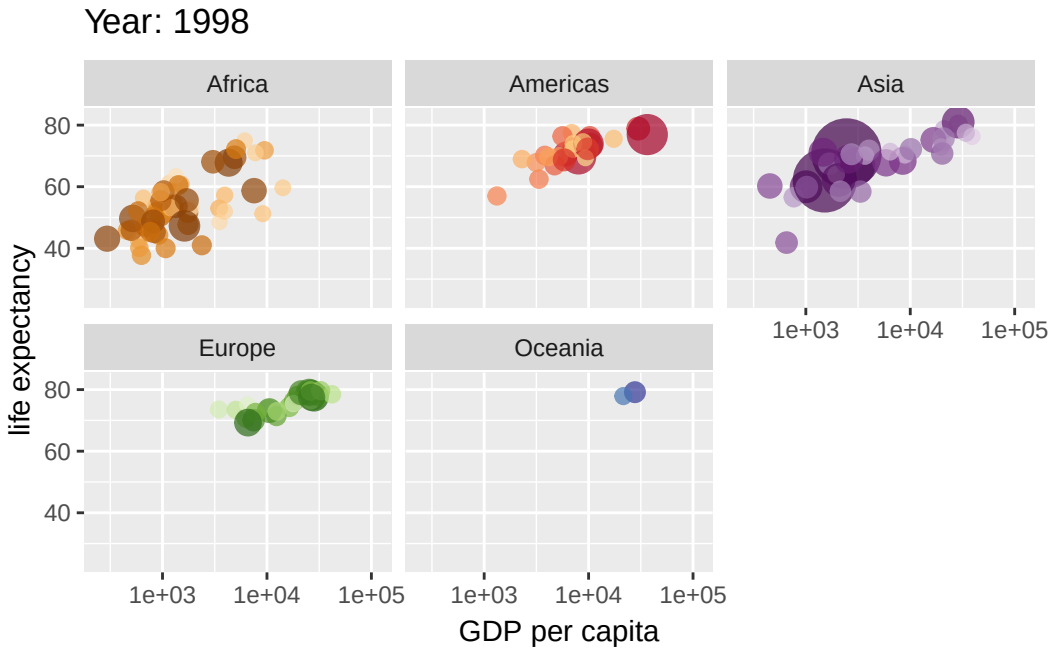

Year: 1995
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

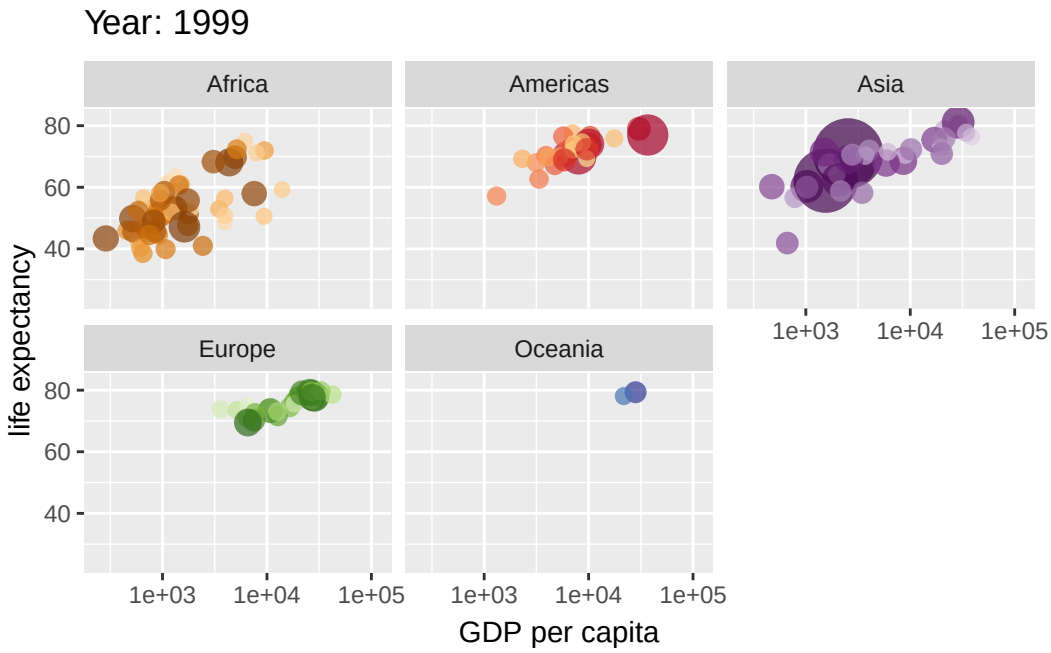

Year: 1996
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

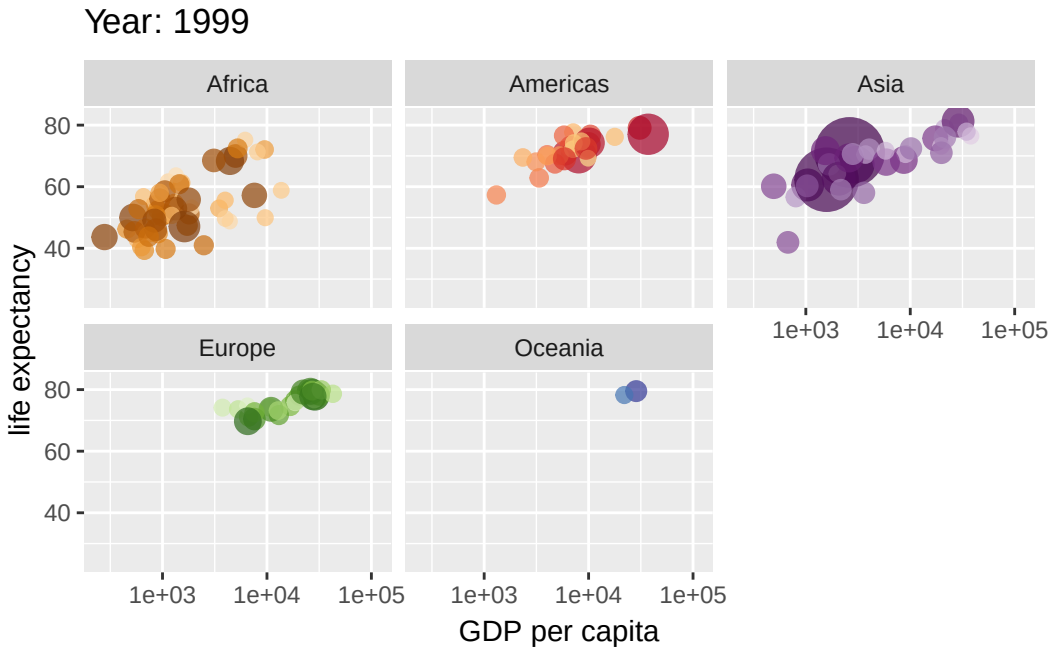

Year: 1996
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

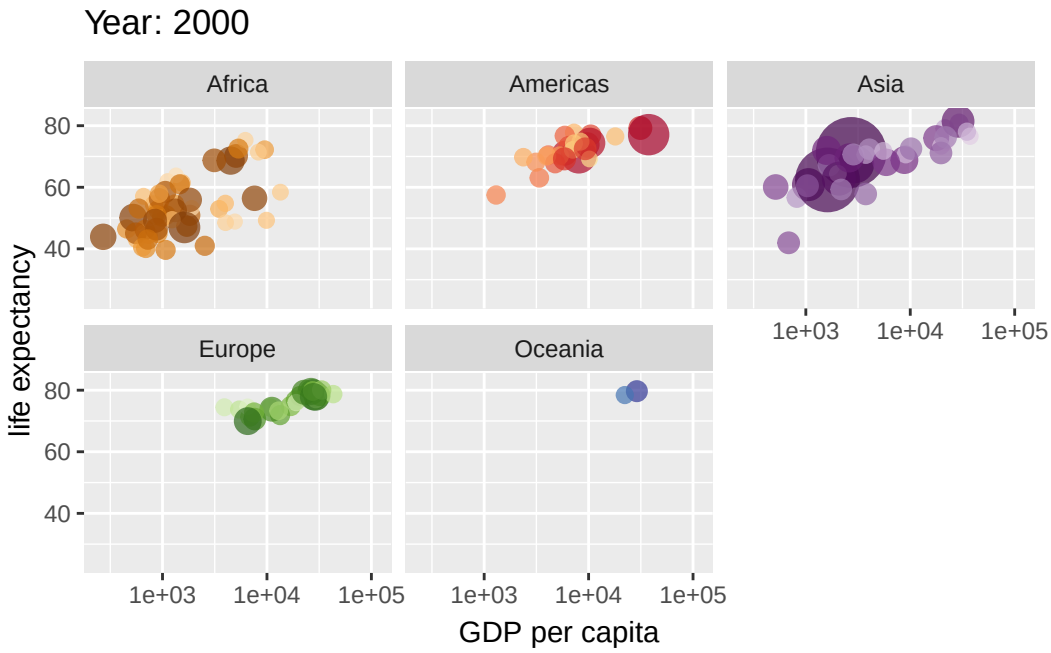

Year: 1997
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

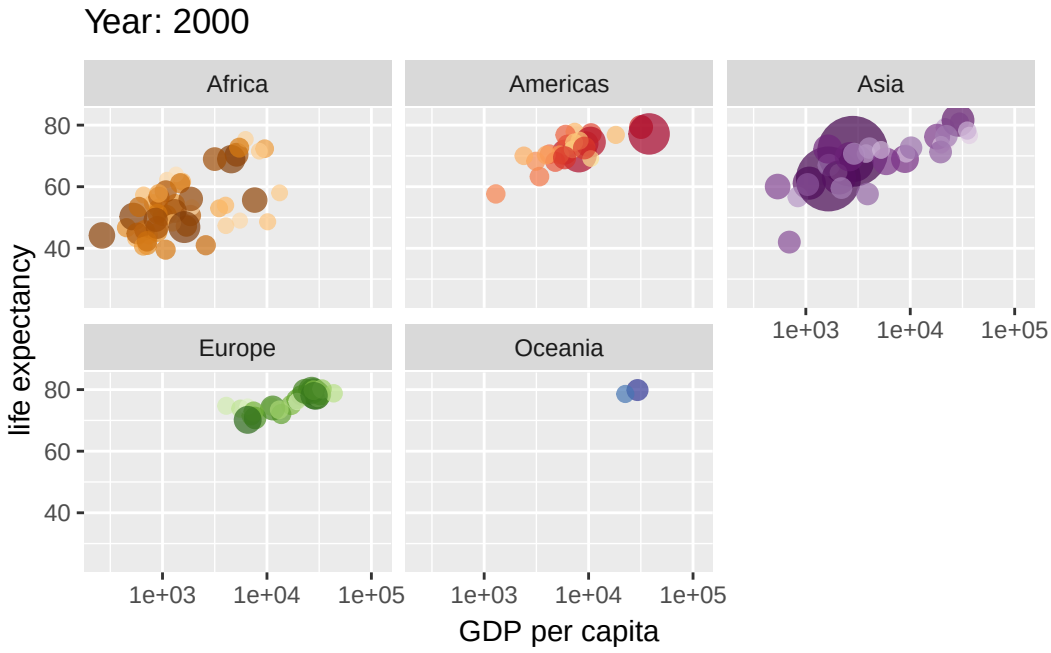

Year: 1998
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

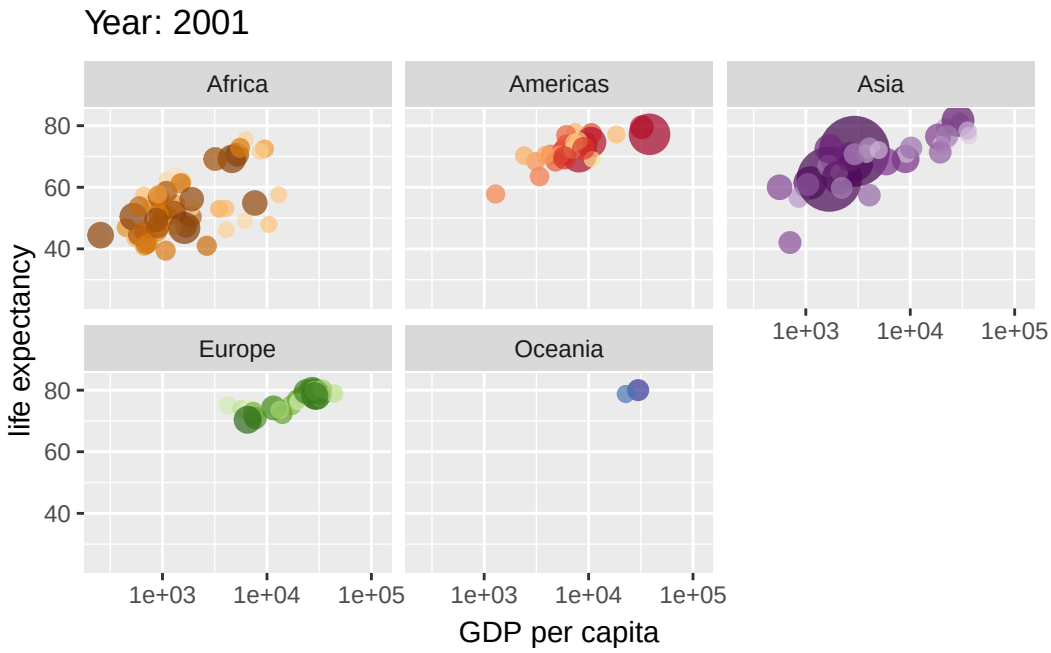

Year: 1998
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

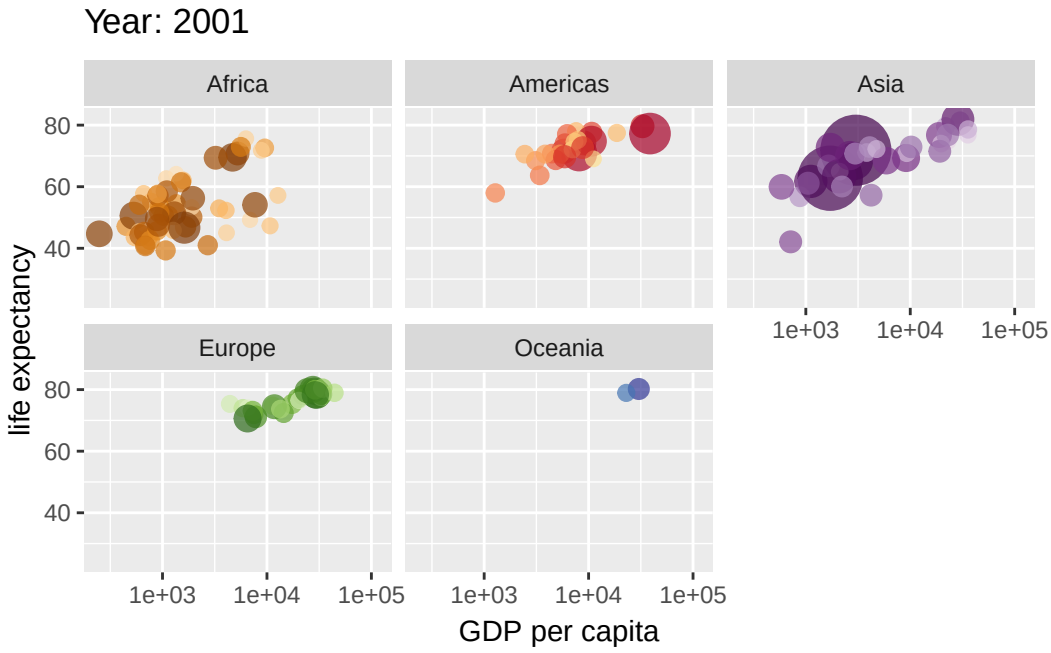

Year: 1999
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

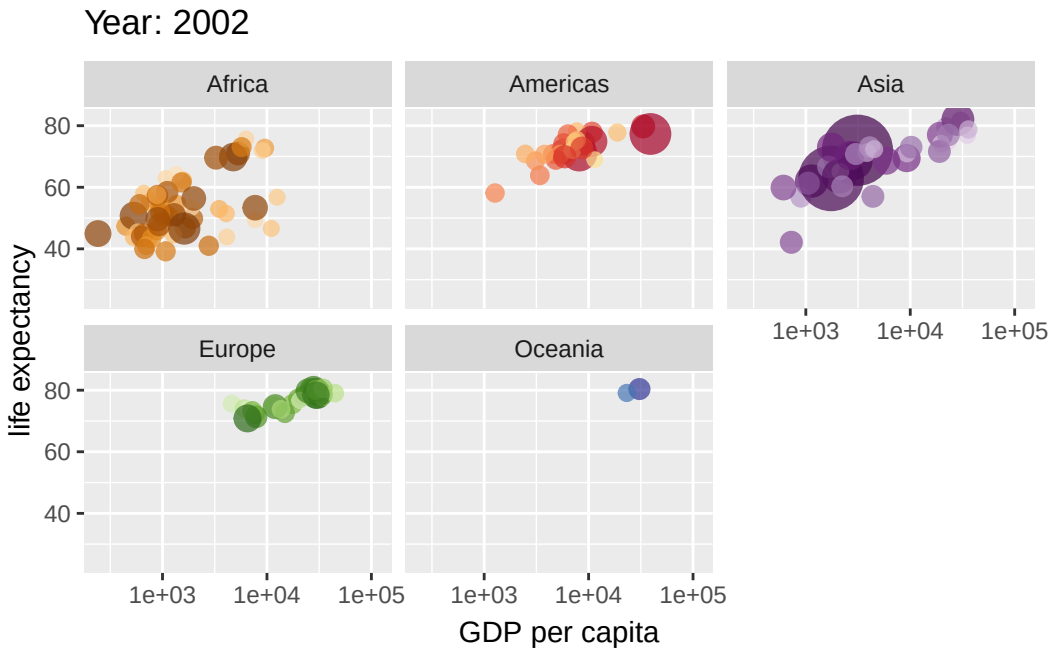

Year: 1999
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

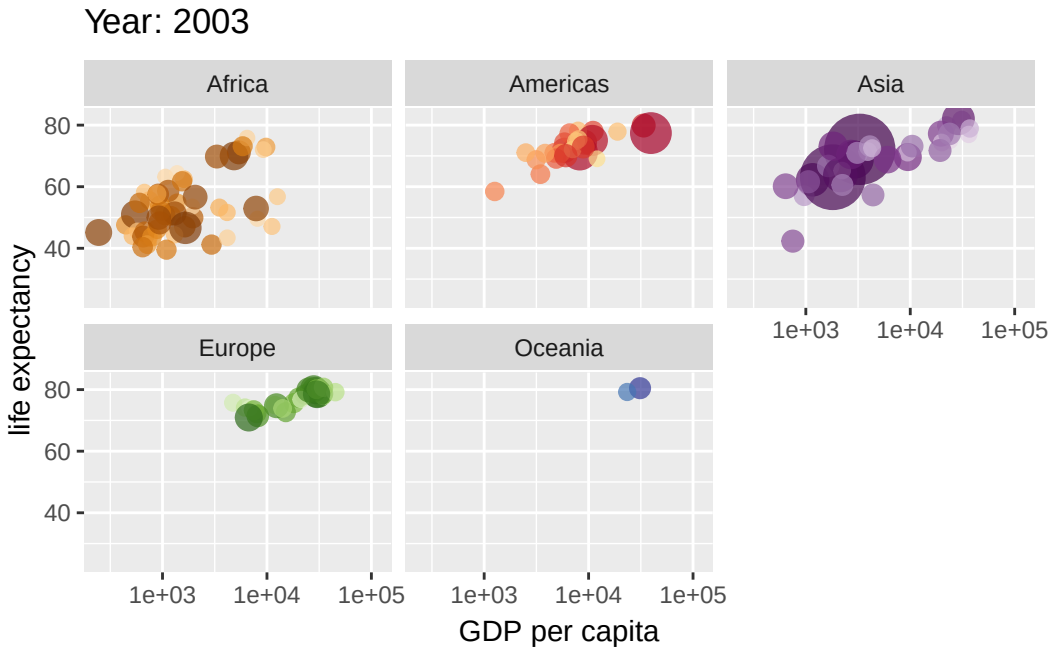

Year: 2000
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

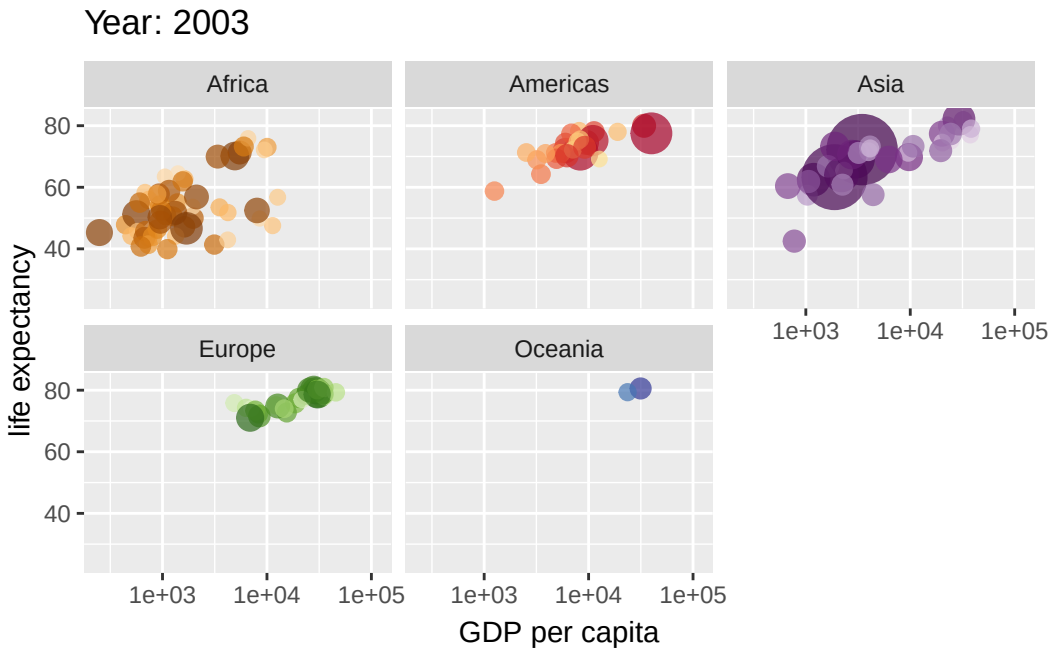

Year: 2000
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

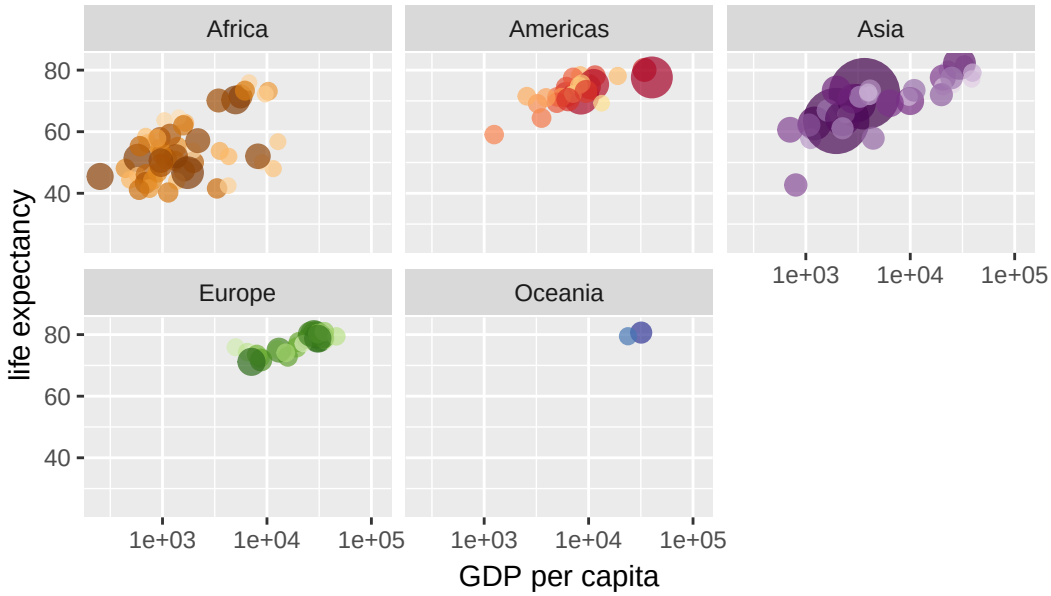

Year: 2001
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

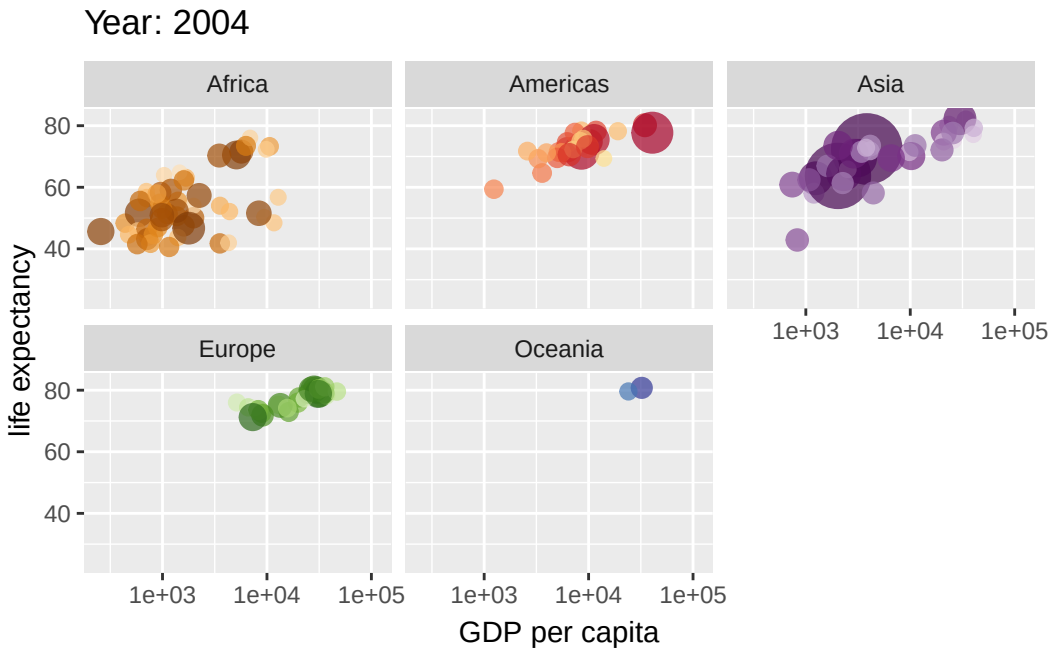

Year: 2001
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

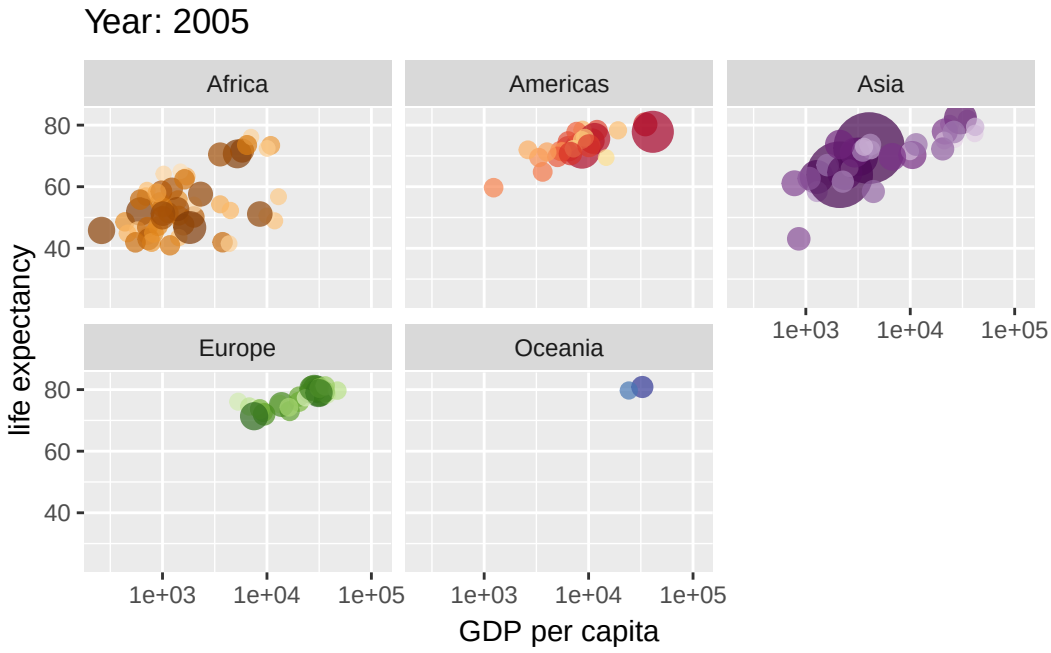

Year: 2002
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

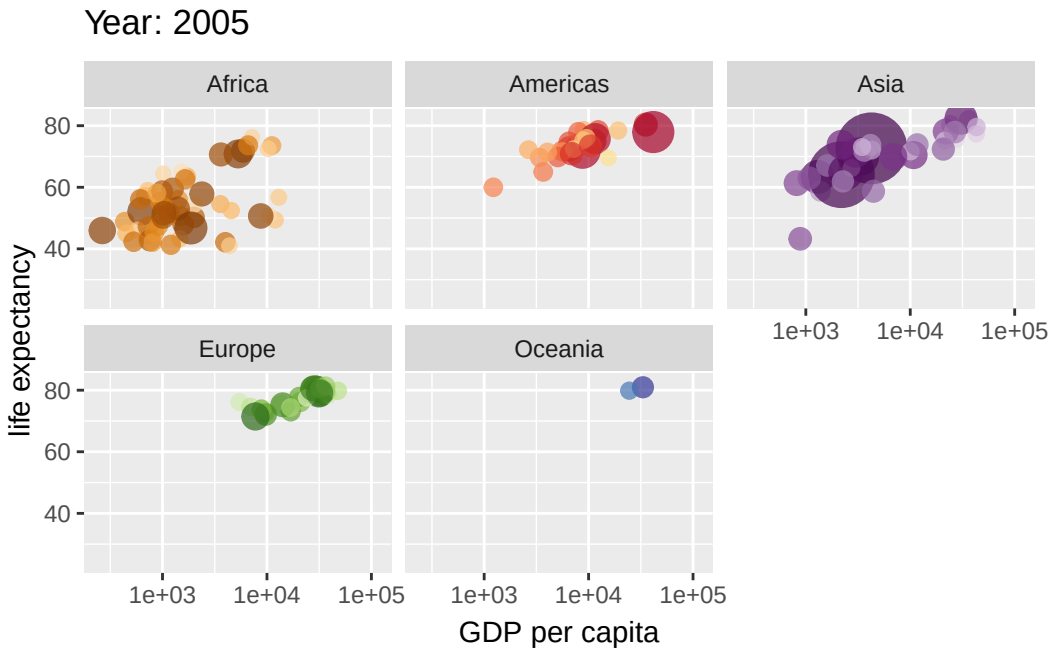

Year: 2003
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

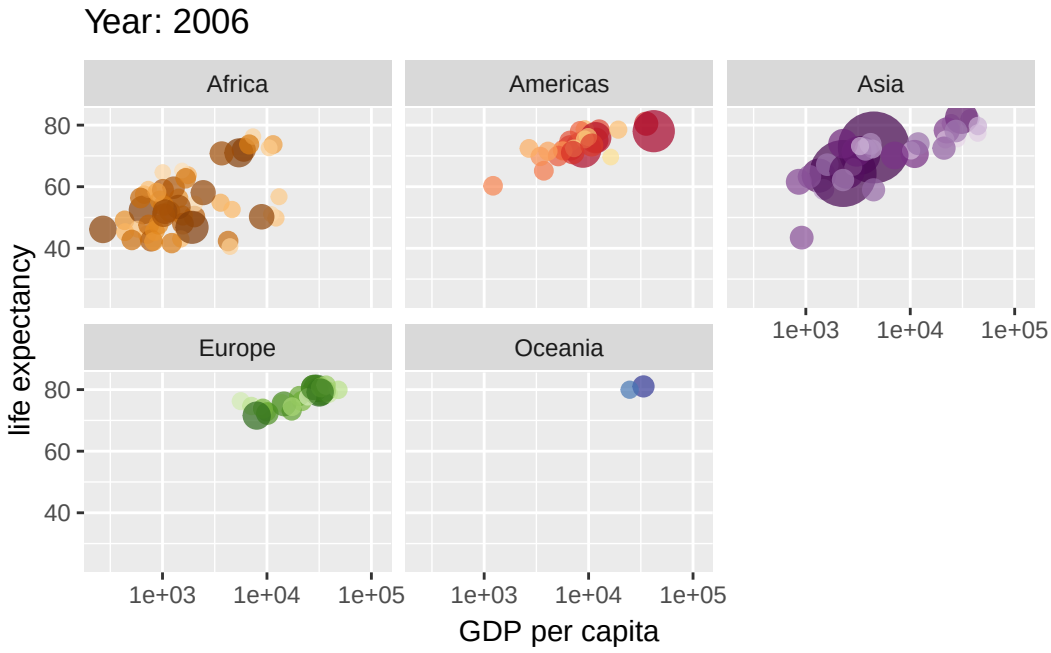

Year: 2003
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

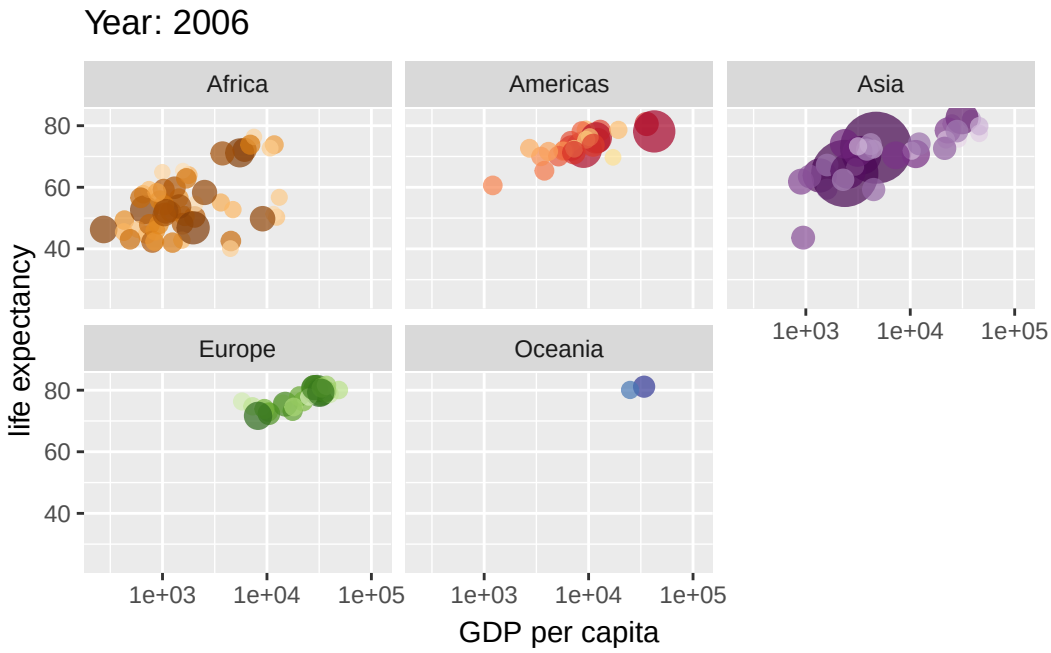

Year: 2004
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

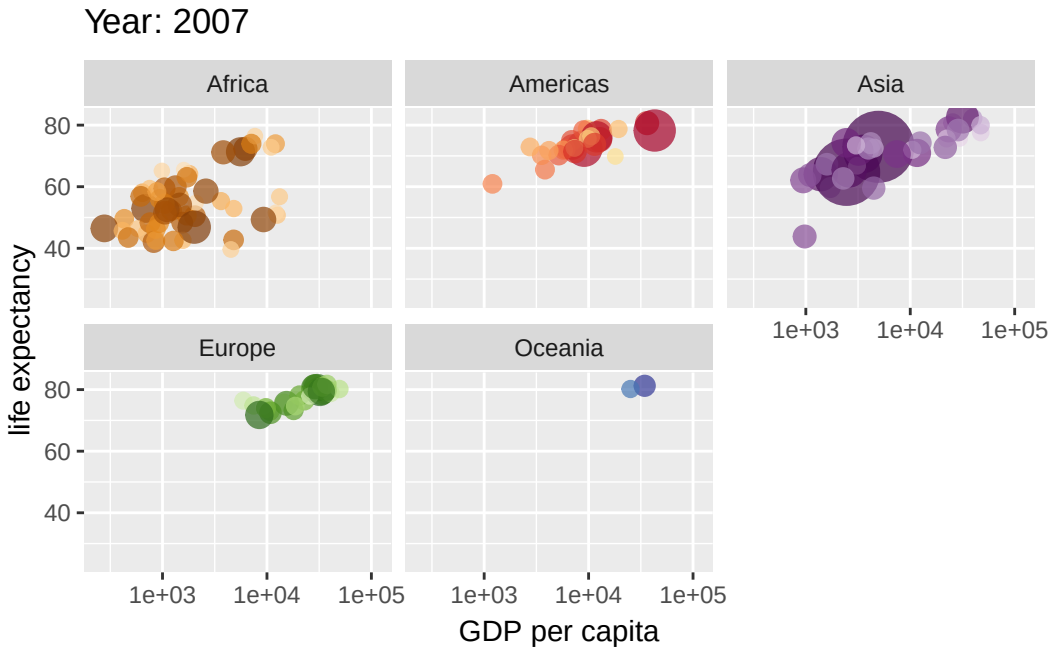

Year: 2004
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita


Year: 2005
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita


Year: 2005
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita


Year: 2006
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita


Year: 2006
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

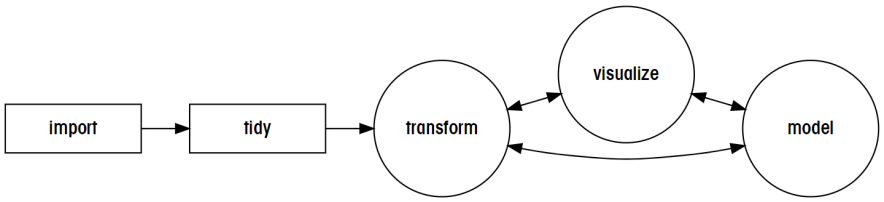

Year: 2007
Africa
Americas
Asia
Europe
Oceania
life expectancy
80
60
40
1e+03
1e+04
1e+05
GDP per capita

Last remarks

Current page provides a primer on visualization, which is a key tool for understanding data and statistics required within the process of data analyses.

Get ahead at https://ggplot2-book.org/

It is strongly advised to play with the techniques discussed above to get some proficiency in using it, as it would add significantly to the flexibility of whatever you want to further do with your data.



Other tidyverse package exist, and within the same framework many more are being developed. The consistency within the tidyverse ecosystem should give you a push though, to study the other packages yourself when of interest.

Base R still is a proper alternative to the tidyverse ecosystem, so be aware that others may do things differently.