CITS4009 Exploratory Data Analysis

Comparing Two Continuous Variables

Dr. Mubashar Hassan

Department of Computer Science and Software Engineering
The University of Western Australia

Semester 2, 2024

Section 1

The Layered Grammar of Graphics

ggplot - the layered grammar of graphics

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(
     mapping = aes(<MAPPINGS>),
     stat = <STAT>,
     position = <POSITION>
  ) +
  <COORDINATE_FUNCTION> +
  <FACET_FUNCTION>
```

This new template takes seven parameters.

You can uniquely describe any plot as a combination of a **dataset**, a **geom**, **a set of mappings**, a **stat**, a **position adjustment**, a **coordinate system**, and a **faceting scheme**.

Chart types - geoms

The chart type in ggplot is referred as types of *geoms*.

A *geom* is the geometrical object that a plot uses to represent data.

- Bar charts use bar geoms, line charts use line geoms, boxplots use boxplot geoms, and so on.
- Scatterplots break the trend; they use the point geom.
- Different geoms can be used to plot the same data.

Section 2

Visually comparing two variables

Packages used

In addition to `tidyverse` and `crayon`, the following packages are used in this set of slides:

```
install.packages("hexbin")
install.packages("gridExtra")
install.packages("maps")
```

Summary of plots useful for two variable comparison

Graph type	Uses
Line plot	Shows the relationship between two continuous variables. Best when that relationship is functional, or nearly so.
Scatter plot	Shows the relationship between two continuous variables. Best when the relationship is too loose or cloud-like to be easily seen on a line plot.
Smoothing curve	Shows underlying "average" relationship, or trend, between two continuous variables. Can also be used to show the relationship between a continuous and a binary or Boolean variable: the fraction of *true* values of the discrete variable as a function of the continuous variable.
Hexbin plot	Shows the relationship between two continuous variables when the data is very dense.
Stacked bar chart	Shows the relationship between two categorical variables (`var1` and `var2`). Highlights the frequencies of each value of `var1`.
Side-by-side bar chart	Shows the relationship between two categorical variables (`var1` and `var2`). Good for comparing the frequencies of each value of `var2` across the values of `var1`. Works best when `var2` is binary.
Filled bar chart	Shows the relationship between two categorical variables (`var1` and `var2`). Good for comparing the relative frequencies of each value of `var2` within each value of `var1`. Works best when `var2` is binary.
Bar chart with faceting	Shows the relationship between two categorical variables (`var1` and `var2`). Best for comparing the relative frequencies of each value of `var2` within each value of `var1` when `var2` takes on more than two values.
Heat map	

Two continuous variables

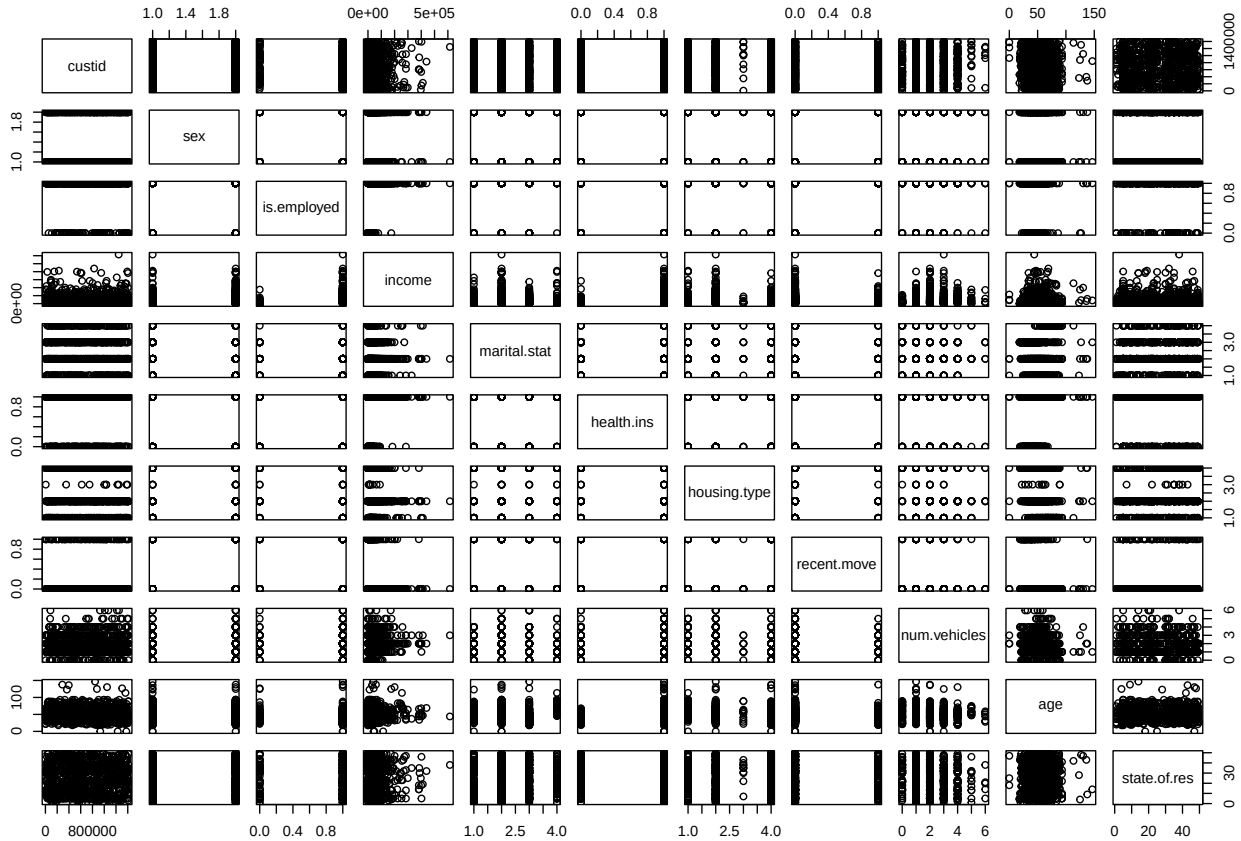
Two categorical variables

Section 3

Scatter plots for two continuous variables

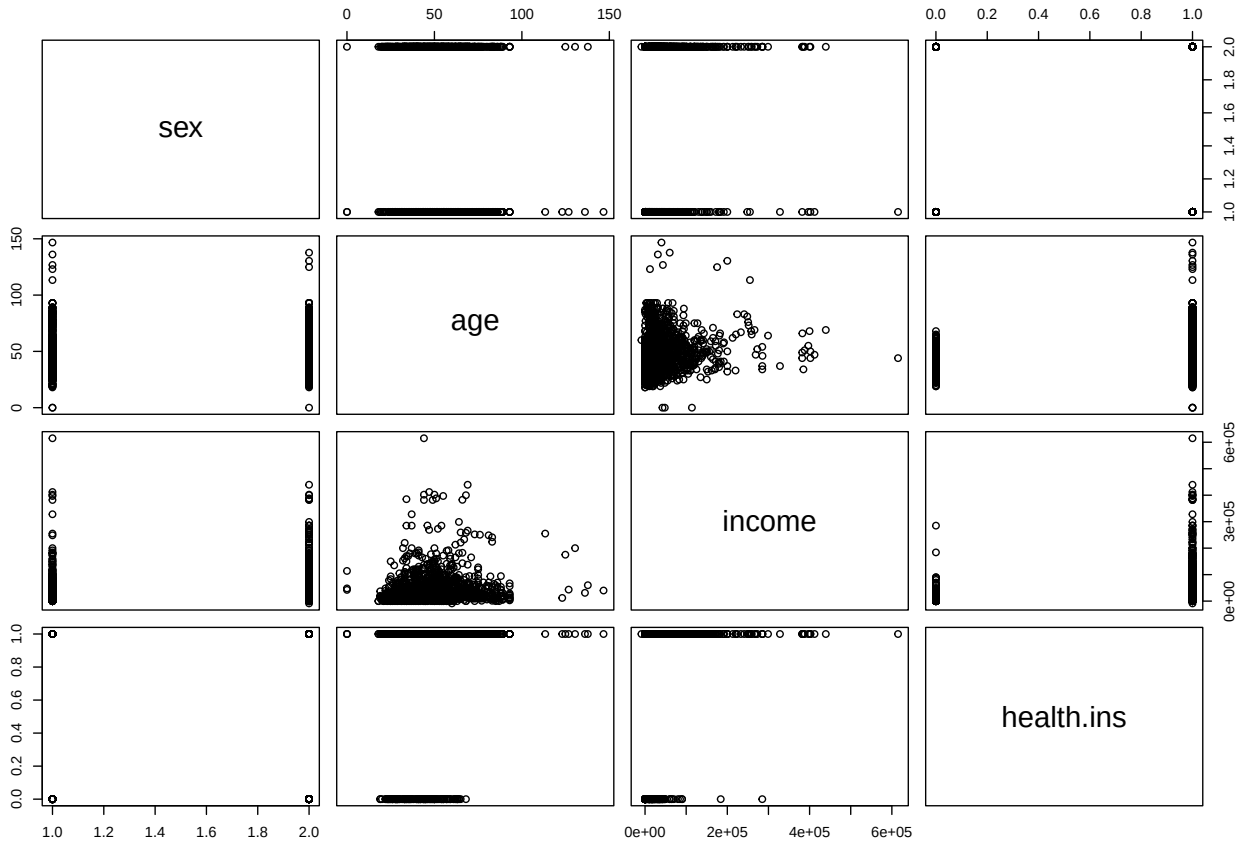
Basic scatter plot matrices (R)

```
custdata <- read.delim('../../data/custdata.tsv', as.is=FALSE)
theme_set(theme_grey(base_size = 18))
pairs(custdata)
```



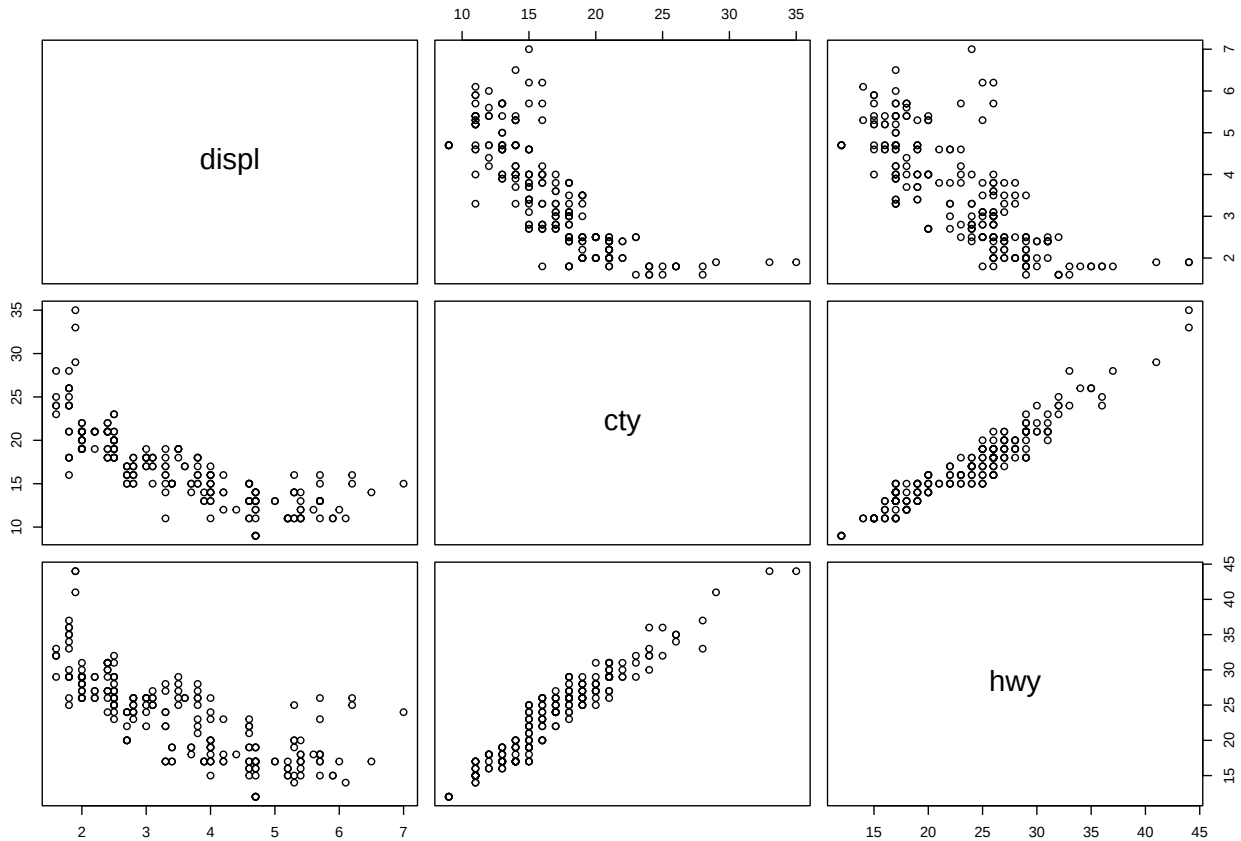
Select the pairs you are interested in

```
pairs(~sex+age+income+health.ins, data=custdata)
```



Select the pairs you are interested in

```
pairs(~displ+cty+hwy, data=mpg)
```



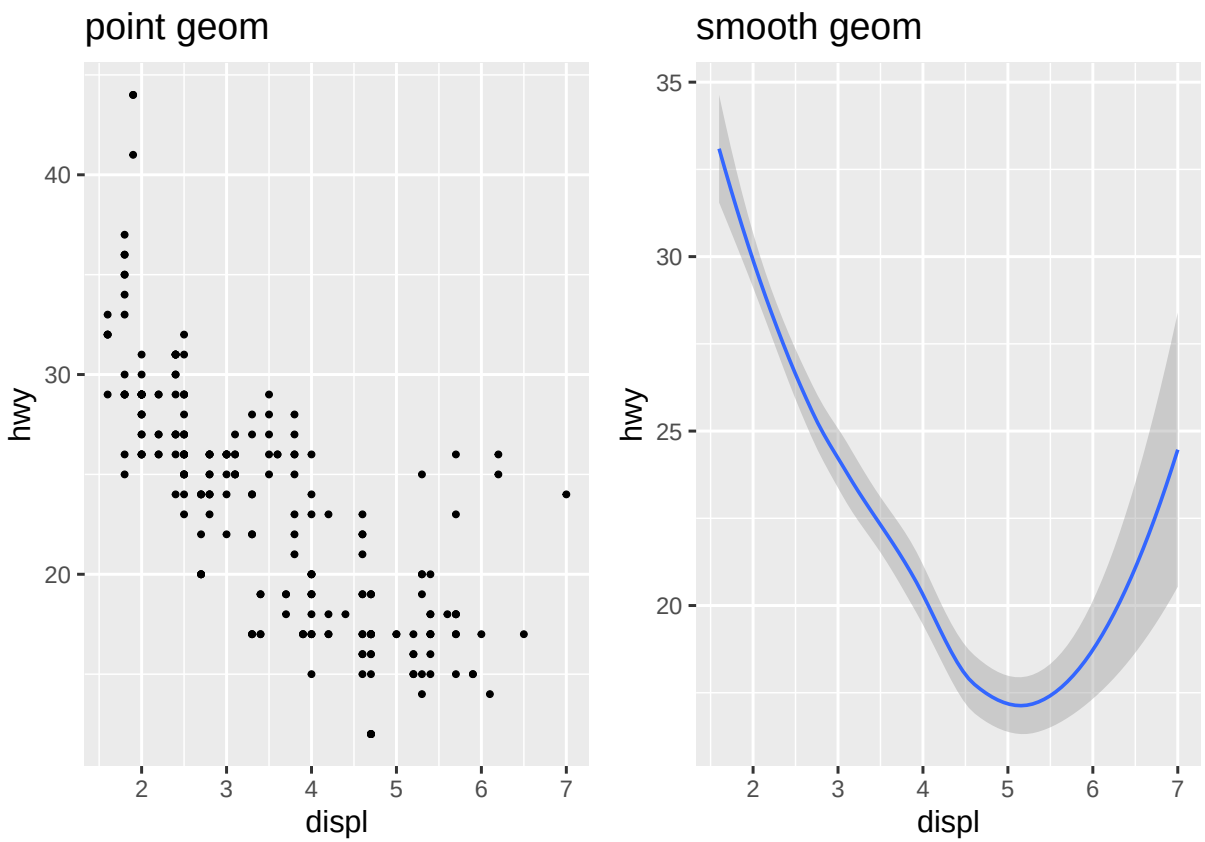
Scatter and Smooth Plots (ggplot) - Code

geom_point and geom_smooth

```
# left
p1 <- ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  labs(title = "point geom")

# right
p2 <- ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy)) +
  labs(title = "smooth geom")
```

Scatter and Smooth Plots (ggplot) - Outputs

```
library(gridExtra)
grid.arrange(p1, p2, ncol=2)
```



The two figures are generated using the same data but different *geom*s.

The Smooth Geom

```
geom_smooth(method="auto", se=TRUE, fullrange=FALSE,
level=0.95)
```

- `method` : smoothing method to be used. Possible values are `lm`, `glm`, `gam`, `loess`, `rlm`.
  - `method = "loess"`: This is the default value for small numbers of observations. It computes *a smooth local regression*. You can read more about `loess` using the R code `?loess`.
  - `method = "lm"`: It fits a linear model. One can also use `lm` to fit non-linear models if the formula is known (e.g., `formula = y ~ poly(x, 3)` will fit a degree 3 polynomial).
- `se` : logical value. If TRUE, confidence interval is displayed.
- `fullrange` : logical value. If TRUE, the fit spans the full range.
- `level` : level of confidence interval to use. Default value is 0.95.

Choose the right aesthetic mapping for comparison

An aesthetic is a visual property of the objects in your plot.

Aesthetics include:

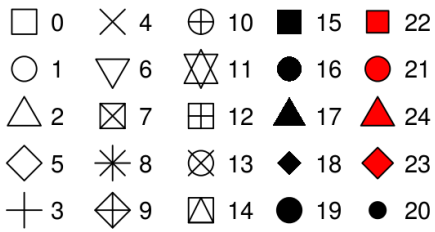
- the coordinates (e.g. `x=`, `y=`), which accepts variables used for each coordinate
- the size (`size=`)
- the colour (`colour=` or `color=`)
- the shape (`shape=`)

You can display a point in different ways by changing the values of its aesthetic properties.

Aesthetic levels

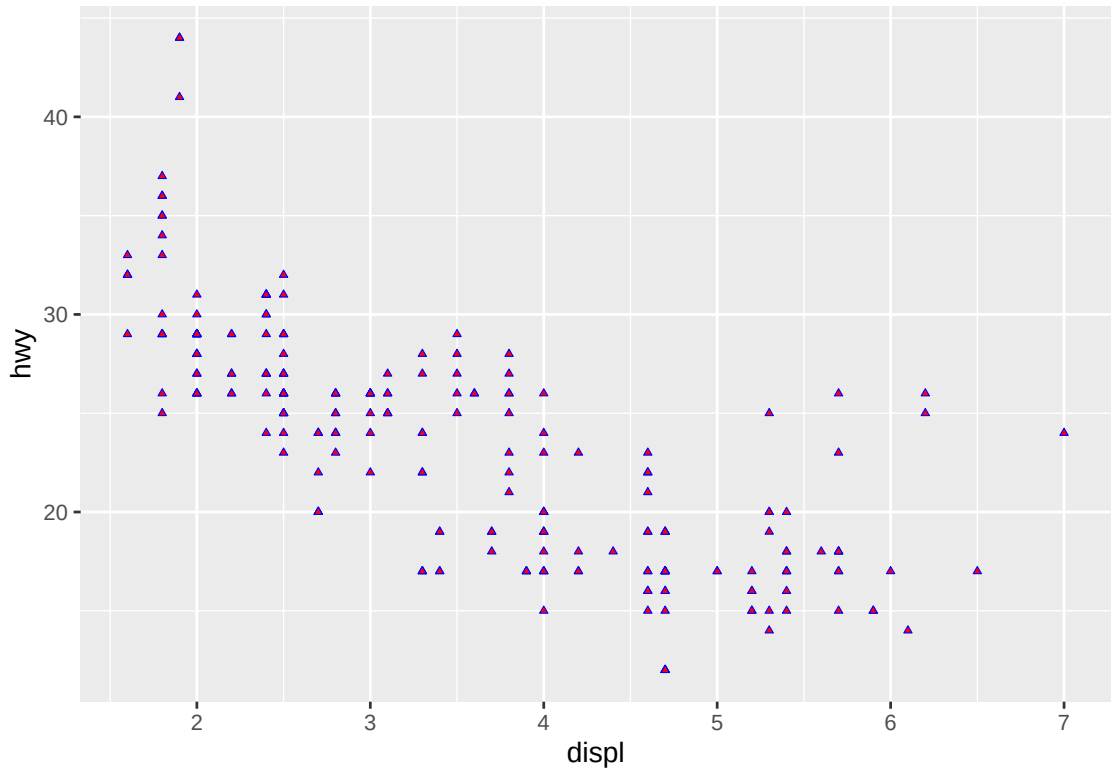
The word "level" is used to describe the values of aesthetic properties. We can change the levels of a point's color, size, and shape:

- `color` is specified using a character string as a color name, e.g. `blue`;
- `size` is measured in mm;
- `shape` is specified as a number.
  - R has 25 built-in shapes that are identified by numbers. Duplicates (e.g., 0, 15, and 22) are differentiated using the interaction of the `colour` and `fill` aesthetics.
    - The hollow shapes (0-14) have a border determined by `colour`;
    - The solid shapes (15-20) are filled with `colour`;
    - The filled shapes (21-24) have a border of `colour` and filled with `fill`.



Manually assigned aesthetic levels

```
ggplot(data = mpg) +
  geom_point(
    mapping = aes(x = displ, y = hwy),
    colour = "blue", shape = 24, fill = "red")
```

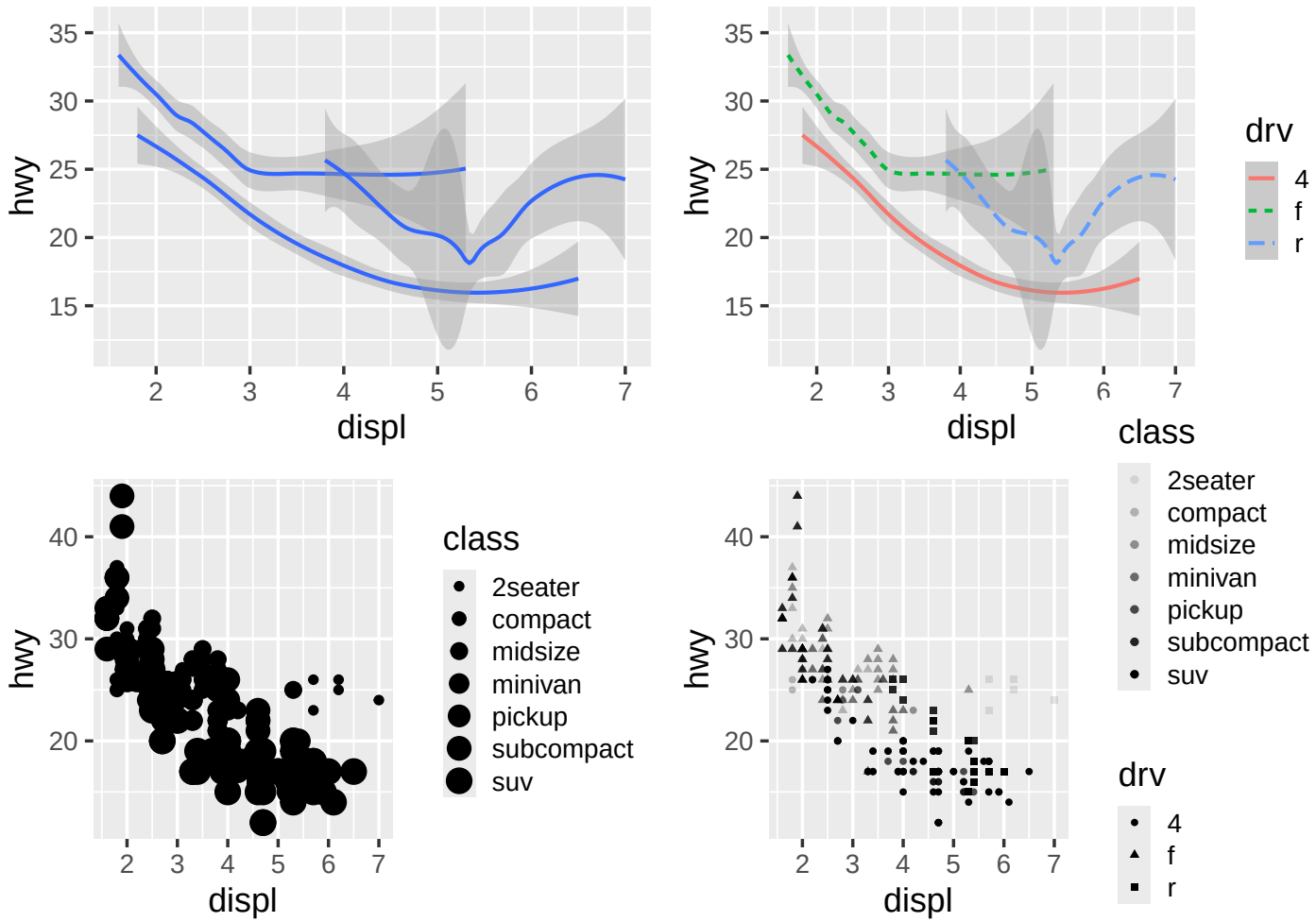


Automatically assigned aesthetic levels (Code)

```
library(gridExtra)
p1 <- ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy, group = drv))

p2 <- ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy, color = drv,
                            linetype = drv))

p3 <- ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, size = class),
             show.legend = TRUE)

p4 <- ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, alpha = class,
                           shape = drv))
```

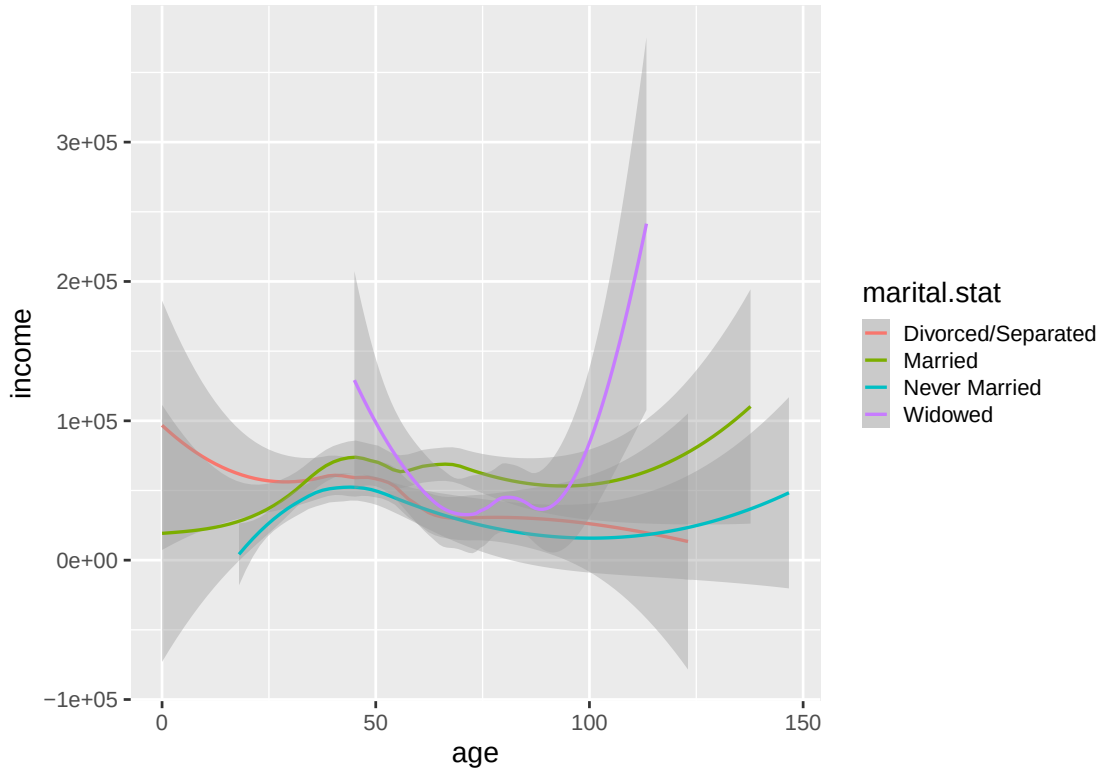
Automatically assigned aesthetic levels (graphs)

```
grid.arrange(p1, p2, p3, p4, ncol=2)
```



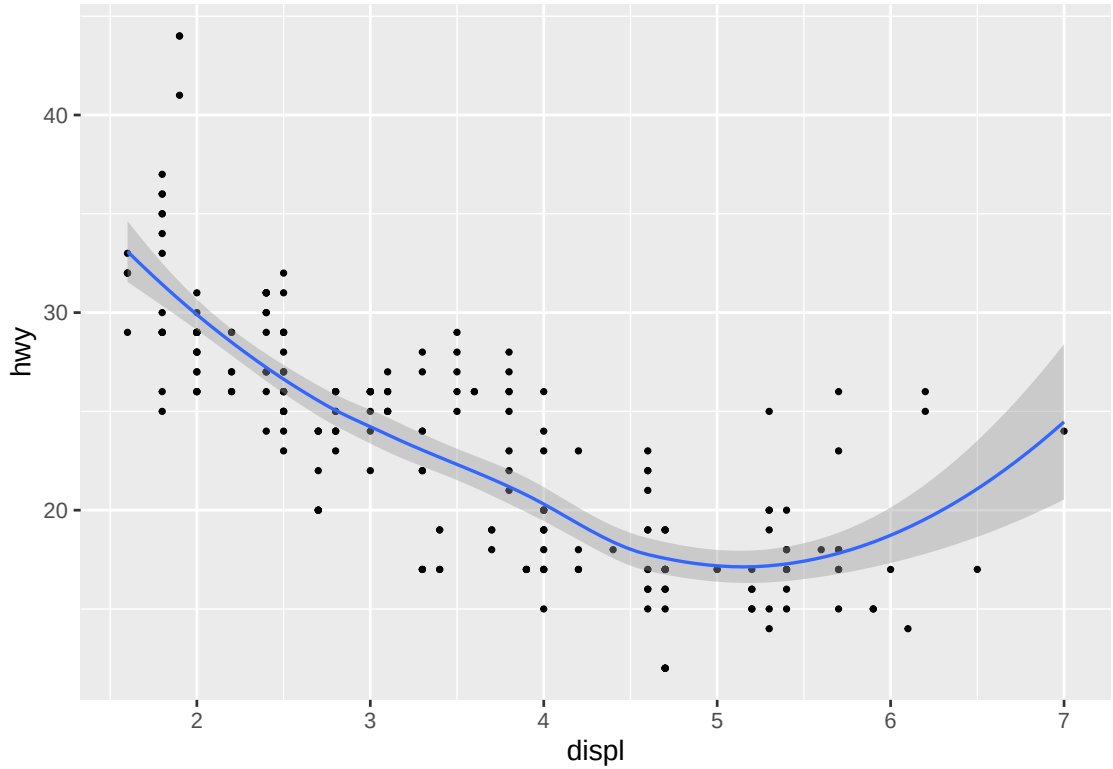
On the customer data

```
ggplot(data = custdata) +
  geom_smooth(mapping = aes(x=age, y=income, color=marital.stat),
              show.legend = TRUE)
```



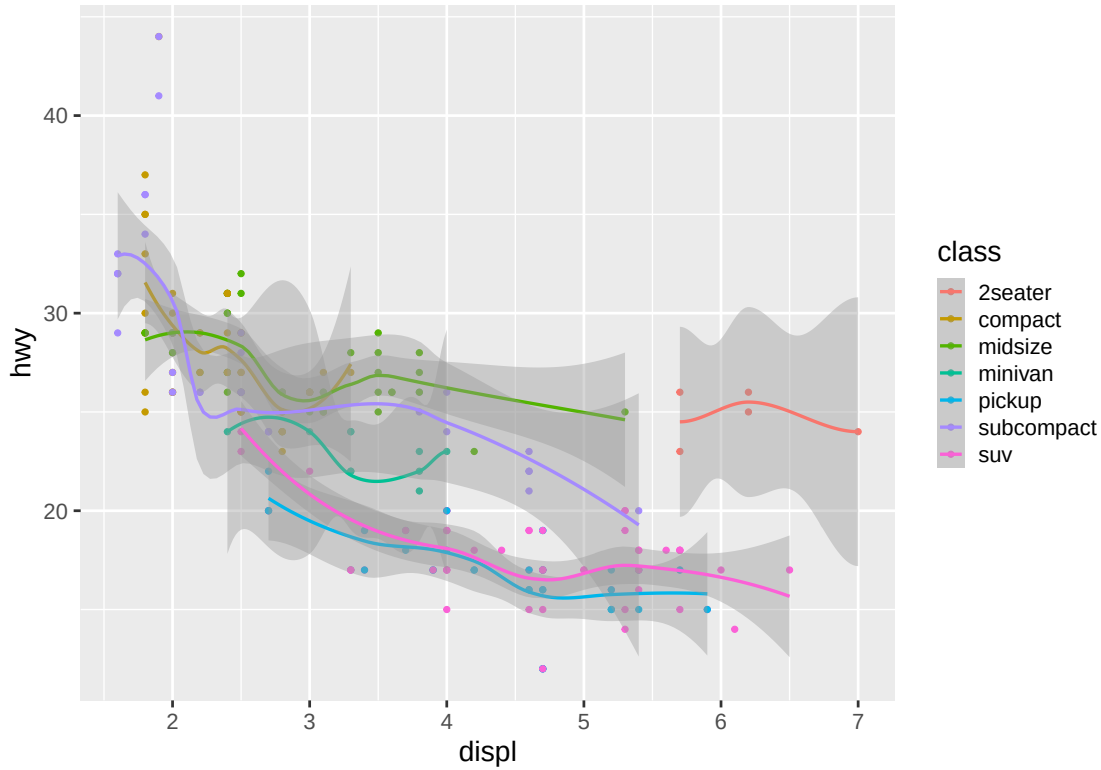
plotting multiple geoms on the same plot

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```



Don't want to specify the data multiple times? Global mappings.

```
ggplot(data = mpg,
       mapping = aes(x = displ, y = hwy, color = class)) +
  geom_point() + geom_smooth()
```



Section 4

Hexbin for two continuous variables

Hexbin - 2D histogram

Install the `hexbin` package first

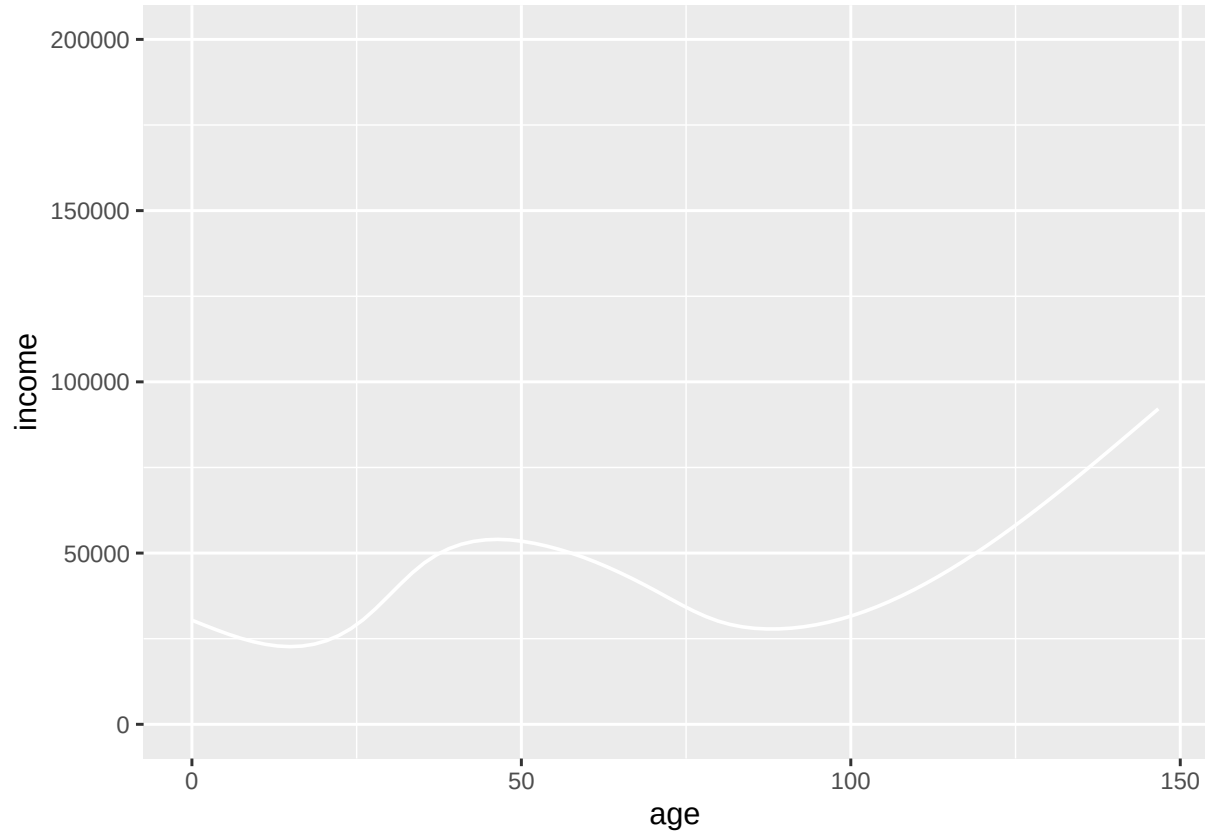
A *hexbin plot* is like a two-dimensional histogram.

The data is divided into bins, and the number of data points in each bin is represented by a color or shading.

Use **hexbin plot** to show the relationship between two **continuous** variables **when the data is very dense**.

Hexbin - 2D histogram example

```
ggplot(custdata, aes(x=age, y=income)) + geom_hex(binwidth=c(5, 10000)) +
  geom_smooth(color="white", se=F) + ylim(0,200000)
```



(compare this with the scatter plot on row 3, column 2 on page 10/26)

References

- **Practical Data Science with R**, *Nina Zumel, John Mount*, Manning, 2nd Ed., 2020 (Chapter 3)
- **R for Data Science**, *Hadley Wickham, Garrett Grolemund*, O'Reilly, 2017 (Chapters 3&7)

CITS4009 Exploratory Data Analysis

Visually Compare Two Categorial Variables

Dr. Mubashar Hassan

Department of Computer Science and Software Engineering
The University of Western Australia

Semester 2, 2024

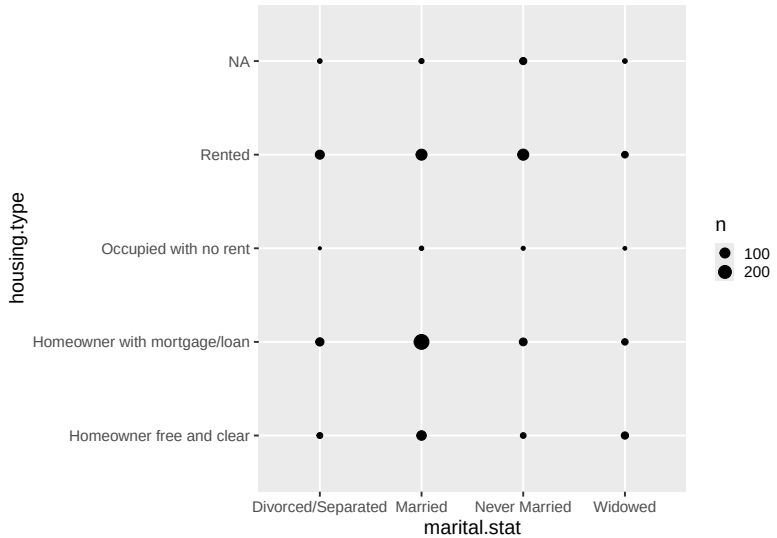
Section 1

Heat Maps

Visualising two categorical variables - geom_count

- The size of each circle in the plot denotes how many observations occurred at each combination of values.
- Covariation will appear as a strong correlation between specific x values and specific y values.

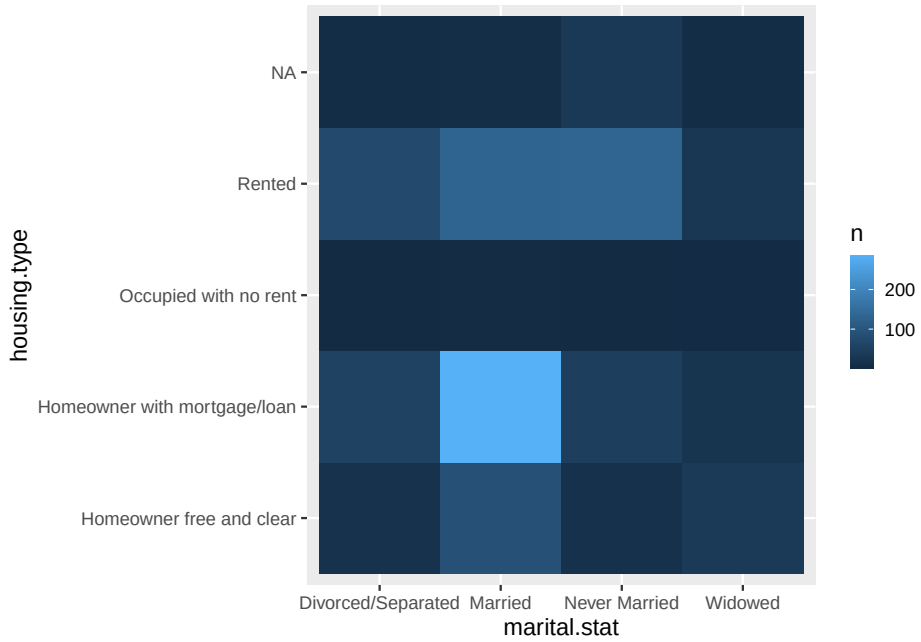
```
ggplot(data = custdata) +
  geom_count(mapping = aes(x=marital.stat, y=housing.type))
```



Note that the counts (frequencies) in the legend are labelled as *n*.

Visualising two categorical variables - geom_tile

```
library(dplyr)
counting <- count(custdata, marital.stat, housing.type)
ggplot(data = counting, mapping = aes(x = marital.stat,
                                      y = housing.type)) +
    geom_tile(mapping = aes(fill = n))
```



Visualising two categorical variables - geom_tile (cont.)

Note that the line

```
counting <- count(custdata, marital.stat, housing.type)
```

produces a data frame:

```
head(counting)
```

```
##         marital.stat                housing.type  n
## 1 Divorced/Separated     Homeowner free and clear 18
## 2 Divorced/Separated Homeowner with mortgage/loan 56
## 3 Divorced/Separated        Occupied with no rent  1
## 4 Divorced/Separated                      Rented 74
## 5 Divorced/Separated                        <NA>  6
## 6              Married     Homeowner free and clear 87
```

So we can use the column `n` as the aesthetic for `geom_tile` on the previous slide.

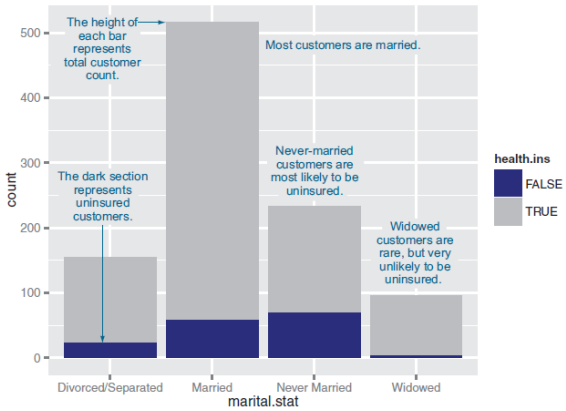
Section 2

Bar Charts and Position Adjustment

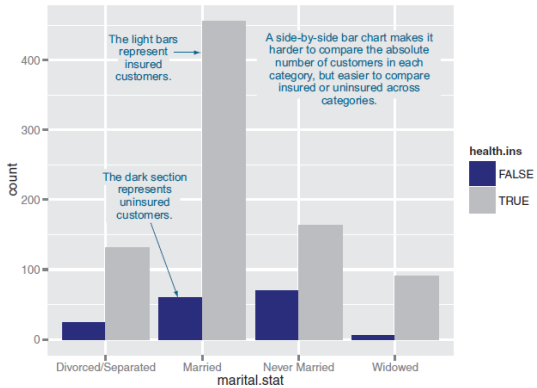
Visualsing two categorical variables using Bar Charts

Let's examine the relationship between marital status and the probability of health insurance coverage.

Stacked bar chart

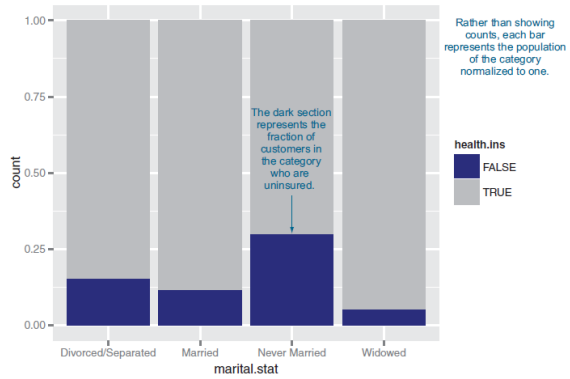


Side-by-side bar chart



(position="dodge")

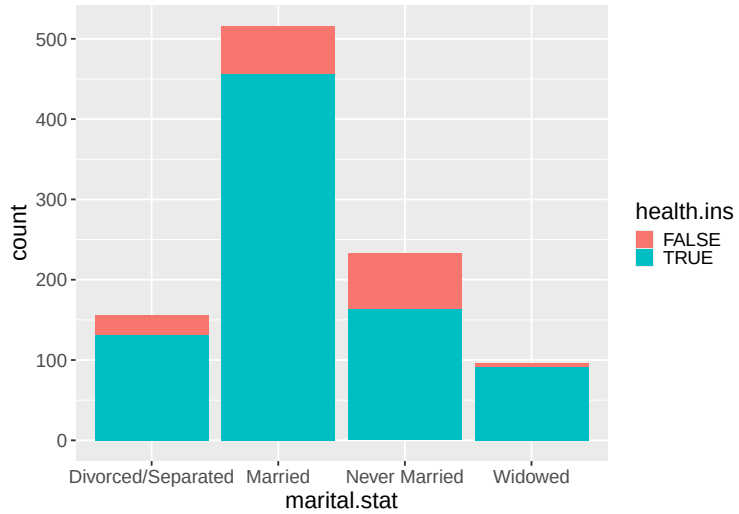
Filled bar chart



(position="fill")

Stacked bar charts

```
ggplot(custdata) + geom_bar(aes(x=marital.stat, fill=health.ins)) +
    theme(text = element_text(size = 22))
```

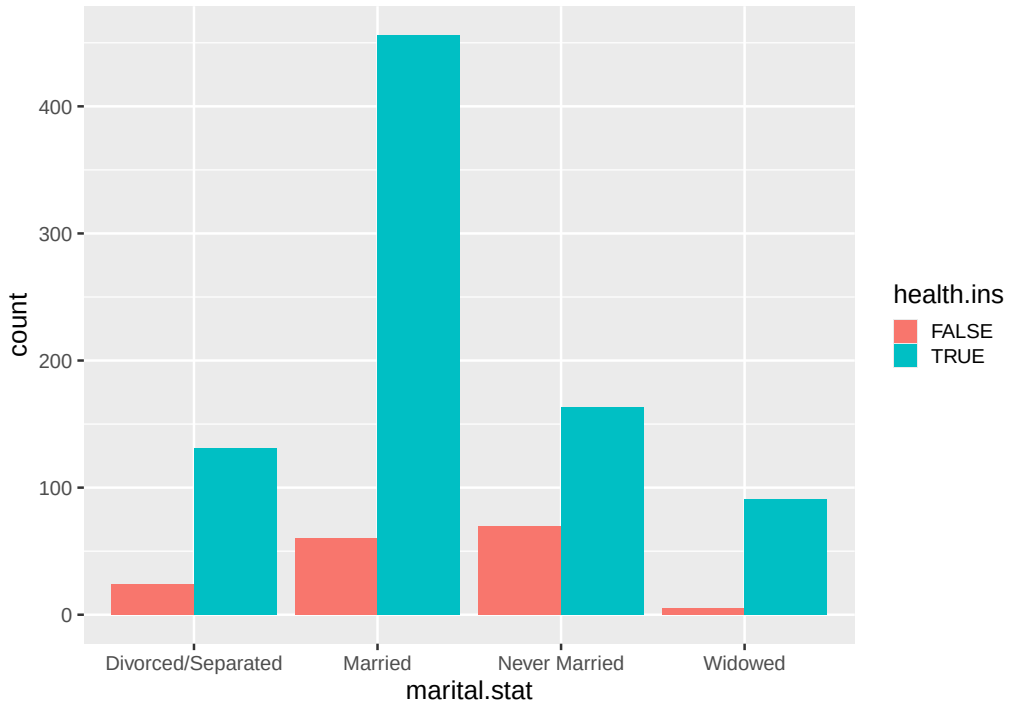


(same as the left diagram on the prvious slide except that "uninsured" is stacked on the top)

The stacked bar chart makes it easy to visualize the distribution of insured and uninsured people in each `marital.stat` category; however, it is difficult to compare the numbers of uninsured people across different `marital.stat` categories as they don't start at the same level.

Side-by-side bar charts

```
ggplot(custdata) +
  geom_bar(aes(x=marital.stat, fill=health.ins), position="dodge")
```



The side-by-side bar chart makes it easier to compare the number of insured (and uninsured) people across different `marital.stat` categories but not the total number of people in each category.

Filled bar charts

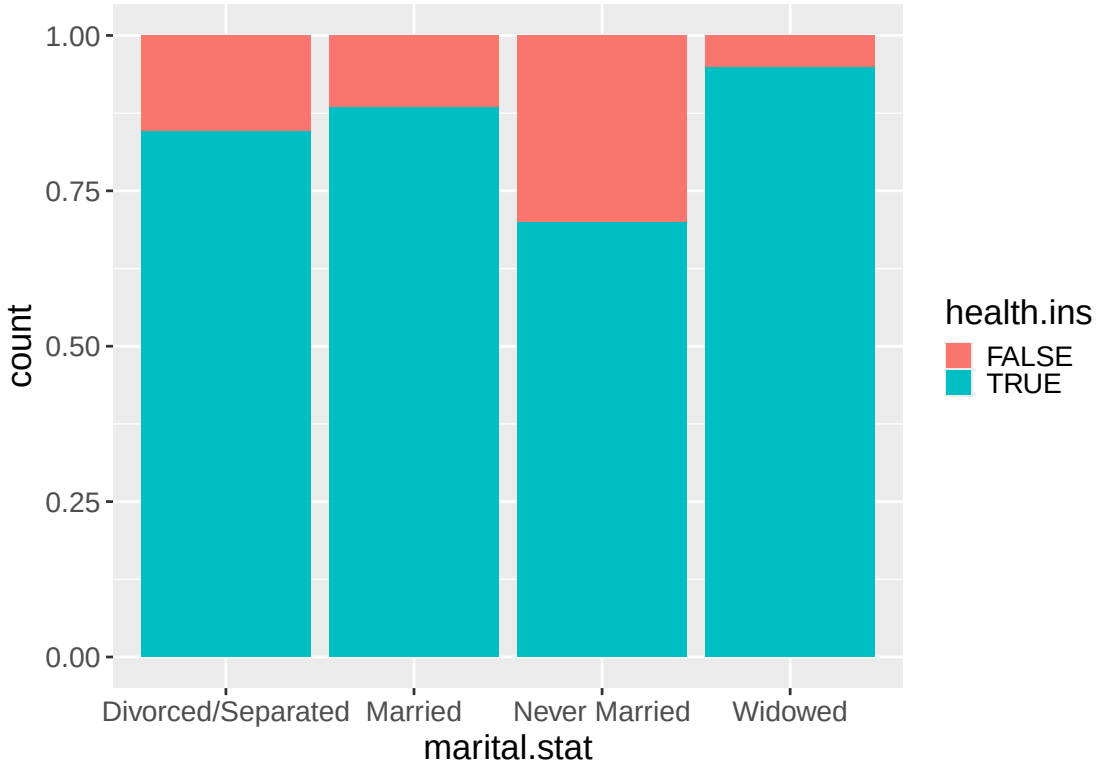
The main shortcoming of both the stacked and side-by-side bar charts is that you can't easily compare the ratios of *insured* to *uninsured* across categories, especially for rare categories like `Widowed`. You can use what `ggplot2` calls a filled bar chart to plot a visualization of the ratios directly.

The filled bar chart makes it obvious to view ratios of second variable for each category.

- For example, the divorced customers are slightly more likely to be uninsured than married ones.
- But we've lost the information that `Widowed`, although highly predictive of insurance coverage, is a rare category.

Filled bar charts

```
ggplot(custdata) +
  geom_bar(aes(x=marital.stat, fill=health.ins), position="fill") +
  theme(text = element_text(size = 22))
```



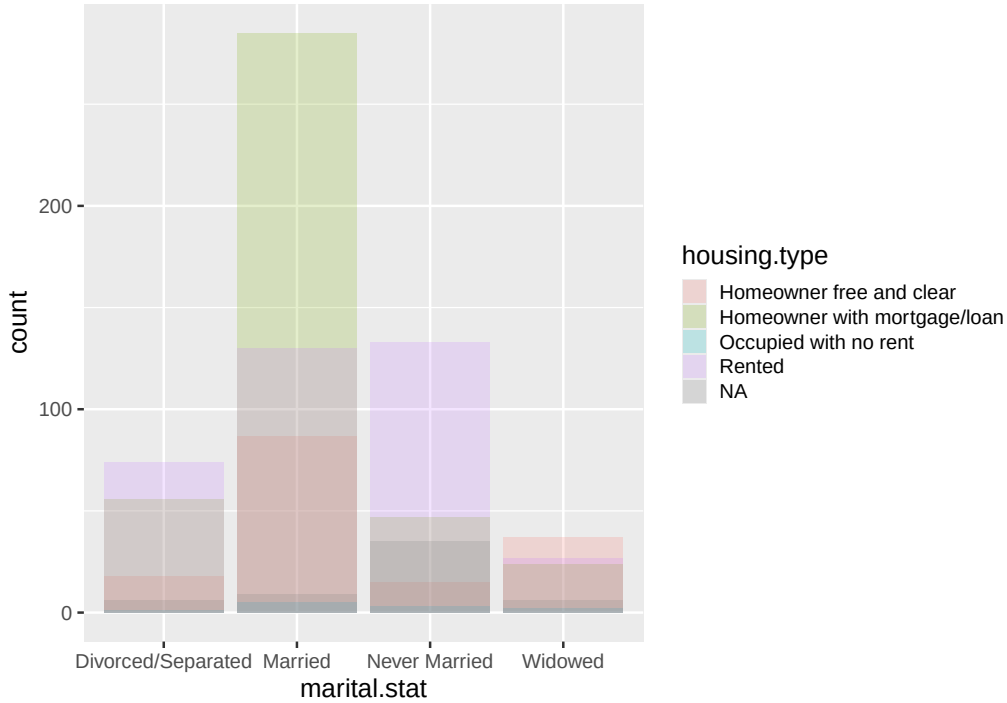
Can you spot a small error in the plot above?

Overlaying bar charts

- The bars are super-imposed onto each other, only the higher values are shown as the lower values are shadowed.
- Better than stacked bars in showing relative size for each category.
- Similar pros and cons to side-by-side bars, but more compact when there are many categories to plot.

Overlaying bar charts (example)

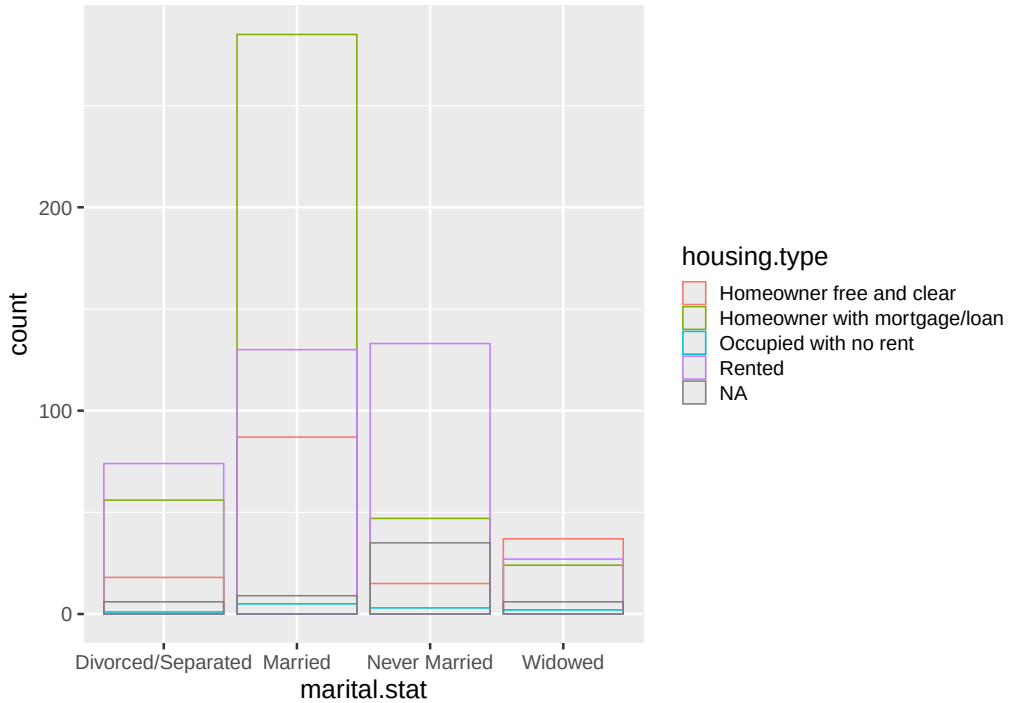
```
ggplot(custdata) + geom_bar(aes(x=marital.stat, fill=housing.type),
                 alpha=0.2,  position="identity") +
  theme(text = element_text(size = 18))
```



(Compare this diagram with the **heat maps** generated using `geom_count` and `geom_tile` on earlier slides)

Overlaying bar charts (example)

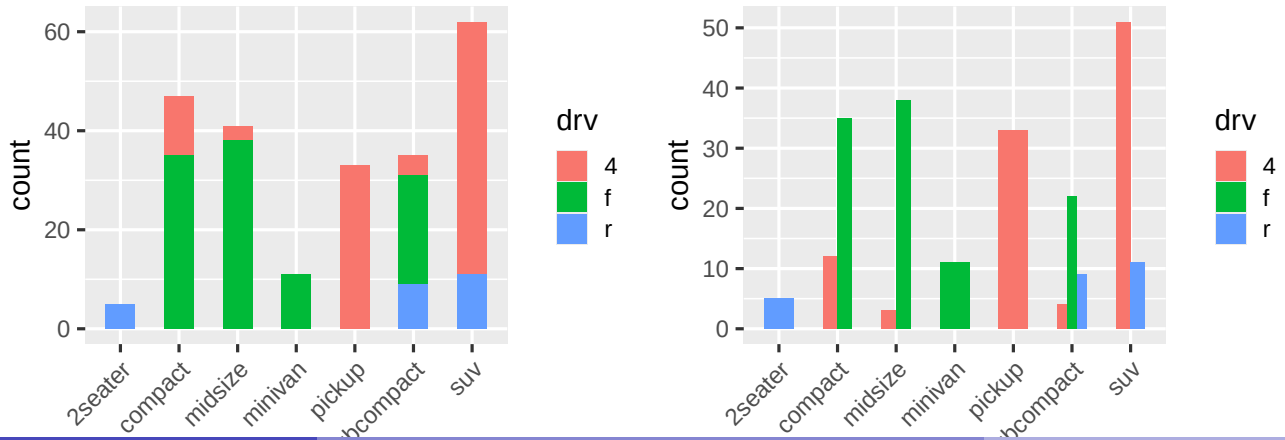
```
ggplot(custdata) +
  geom_bar(aes(x= marital.stat, colour=housing.type),
           fill=NA,  position="identity") +
  theme(text = element_text(size = 18))
```



How to choose the *primary variable*?

All the bar charts shown on the previous slides have two categorical variables. One of these variables should be the *primary variable* (the x-axis) and the other (the *secondary variable*) should be used for aesthetic mapping.
Example: Bar charts of `class` (primary) and `drv` from the `mpg` dataset:

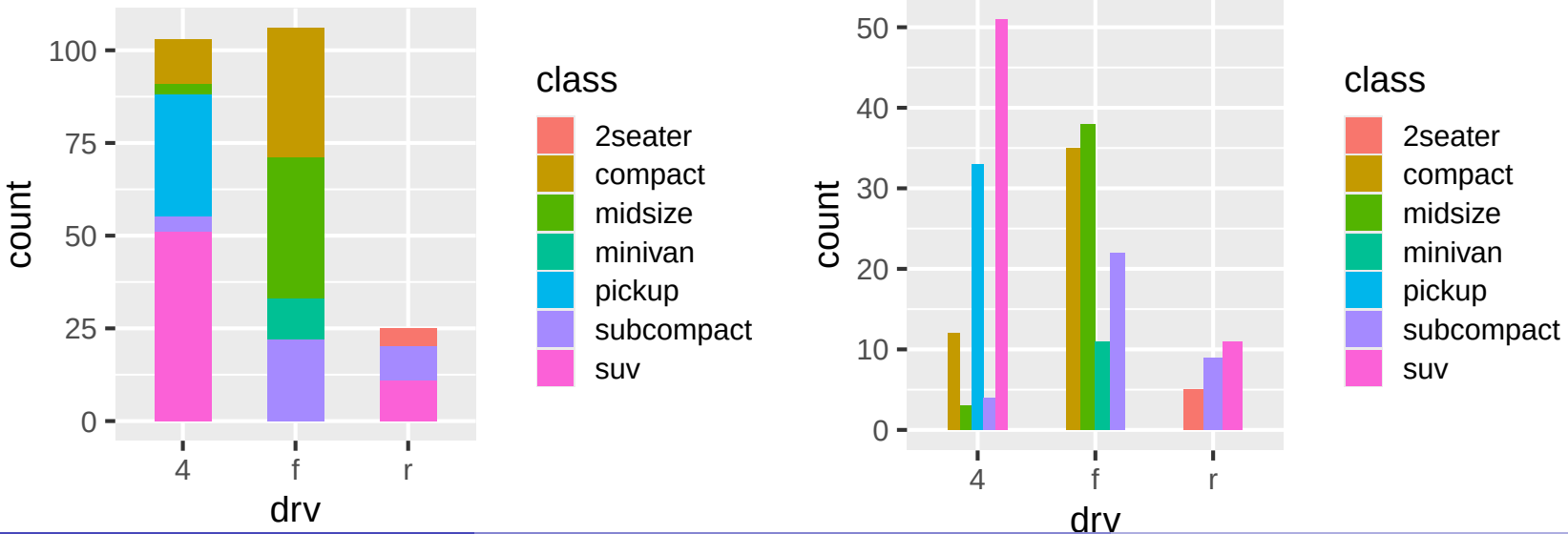
```
fig1 <- ggplot(mpg) + geom_bar(aes(x=class, fill=drv), width=0.5) +
   theme(text=element_text(size=16), axis.text.x=element_text(angle=45, hjust=1), aspect.ratio=0.8)
fig2 <- ggplot(mpg) + geom_bar(aes(x=class, fill=drv), width=0.5, position="dodge") +
   theme(text=element_text(size=16), axis.text.x=element_text(angle=45, hjust=1), aspect.ratio=0.8)
grid.arrange(fig1, fig2, nrow=1)
```



How to choose the *primary variable*? (cont.)

Example: Bar charts of `drv` (primary) and `class` from the `mpg` dataset:

```
fig1 <- ggplot(mpg) + geom_bar(aes(x=drv, fill=class), width=0.5) +
   theme(text=element_text(size=16), aspect.ratio=1.2)
fig2 <- ggplot(mpg) + geom_bar(aes(x=drv, fill=class), width=0.5, position="dodge") +
   theme(text=element_text(size=16), aspect.ratio=1.2)
grid.arrange(fig1, fig2, nrow=1)
```



References

- **Practical Data Science with R**, *Nina Zumel, John Mount*, Manning, 2nd Ed., 2020 (Chapter 3)
- **R for Data Science**, *Hadley Wickham, Garrett Grolemund*, O'Reilly, 2017 (Chapters 3&7)

Advanced Visualisation

CITS4009 Exploratory Data Analysis

Dr. Mubashar Hassan

Department of Computer Science and Software Engineering
The University of Western Australia

Semester 2, 2024

Section 1

Statistical Transformation

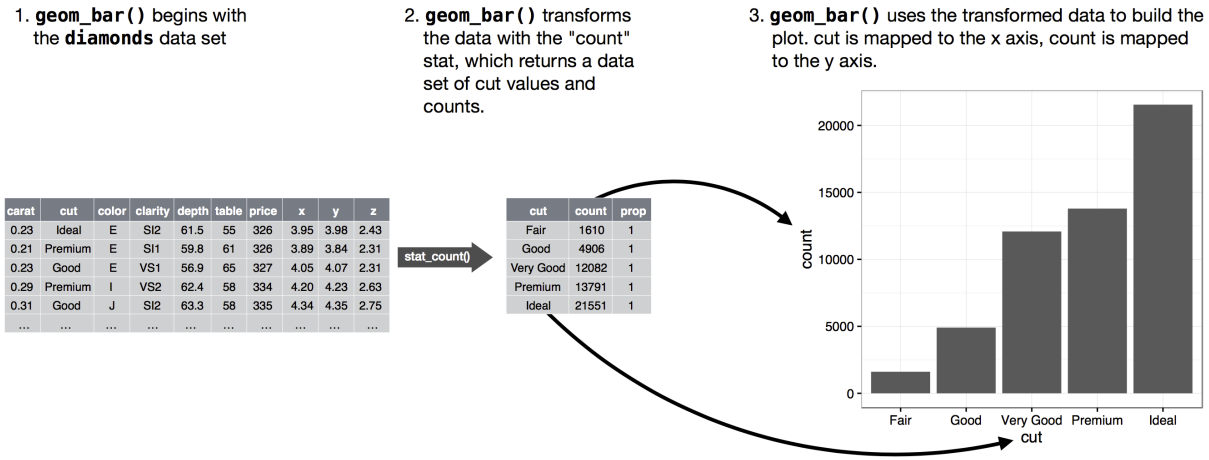
Statistical Transformation - Where does count come from?

Many graphs, like scatterplots, plot the raw values of a dataset. Other graphs, like bar charts, calculate new values to plot:

- *bar charts*, *histograms*, and *frequency polygons* bin your data and then plot the bin counts (i.e., the number of points that fall in each bin).
- *smoothers* fit a model to your data and then plot predictions from the model.
- *boxplots* compute a robust summary of the distribution and then display a specially formatted box.

Use geoms and stats interchangeably

The algorithm used to calculate new values for a graph is called a **stat**, short for *statistical transformation*.



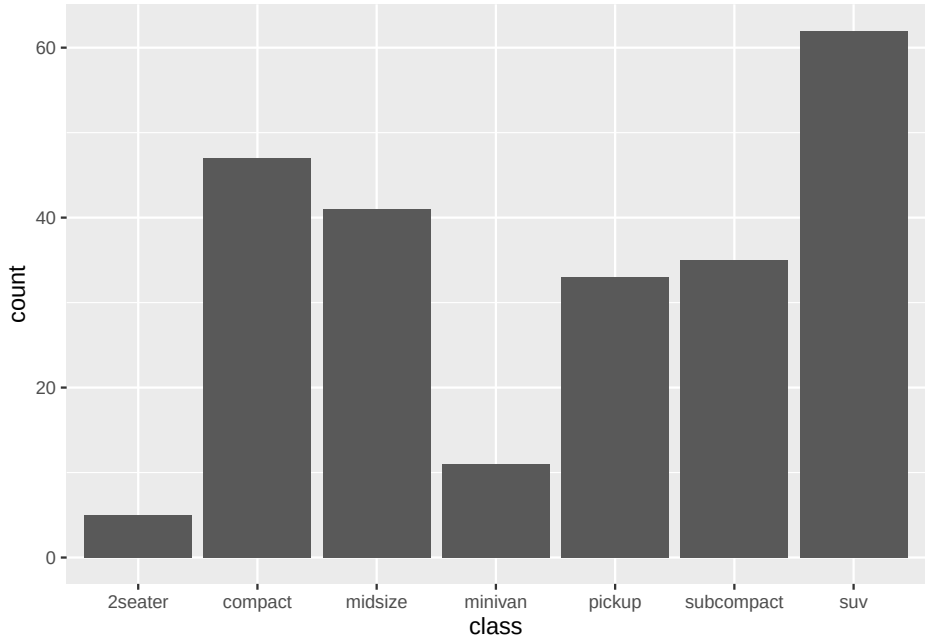
Every *geom* has a default *stat*; and every *stat* has a default *geom*.

- Typically we can use geoms without worrying about the underlying statistical transformation.

Use geoms and stats interchangeably (example)

For example, you can recreate the previous plot using `stat_count()` instead of `geom_bar()`: {(See https://ggplot2.tidyverse.org/reference/)}

```
ggplot(data = mpg) +
  stat_count(mapping = aes(x = class))
```



Overwrite the default stat

Create our own stats for plotting:

```
library(tibble)
demo <- tribble(
  ~cut,          ~freq,
  "Fair",        1610,
  "Good",        4906,
  "Very Good",   12082,
  "Premium",     13791,
  "Ideal",       21551
)
```

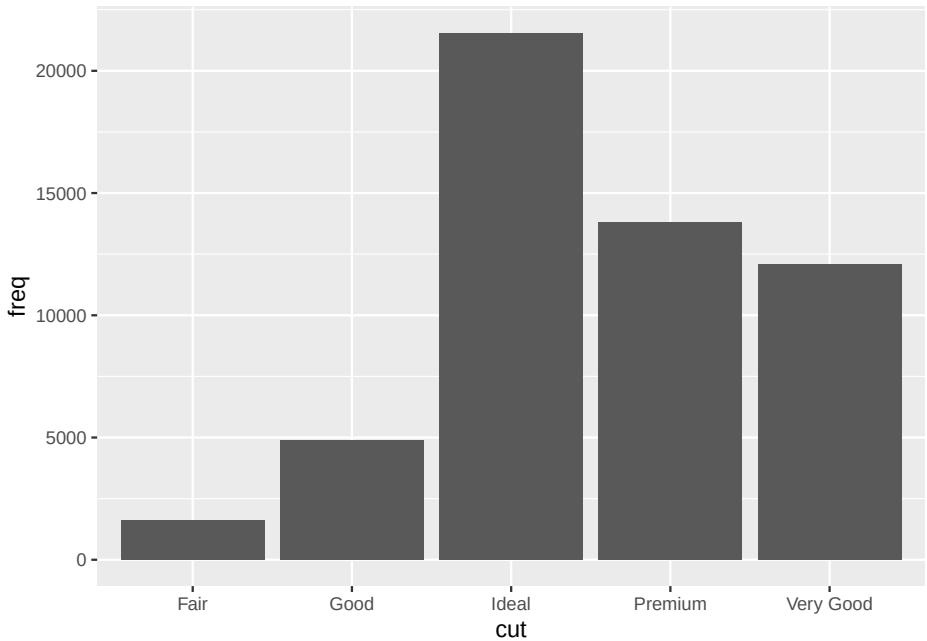
`tribble()` allows you to create small data frames row by row.

The code above creates a small data frame containing two columns whose headings are `cut` and `freq`.

Overwrite the default stat (stat="identity")

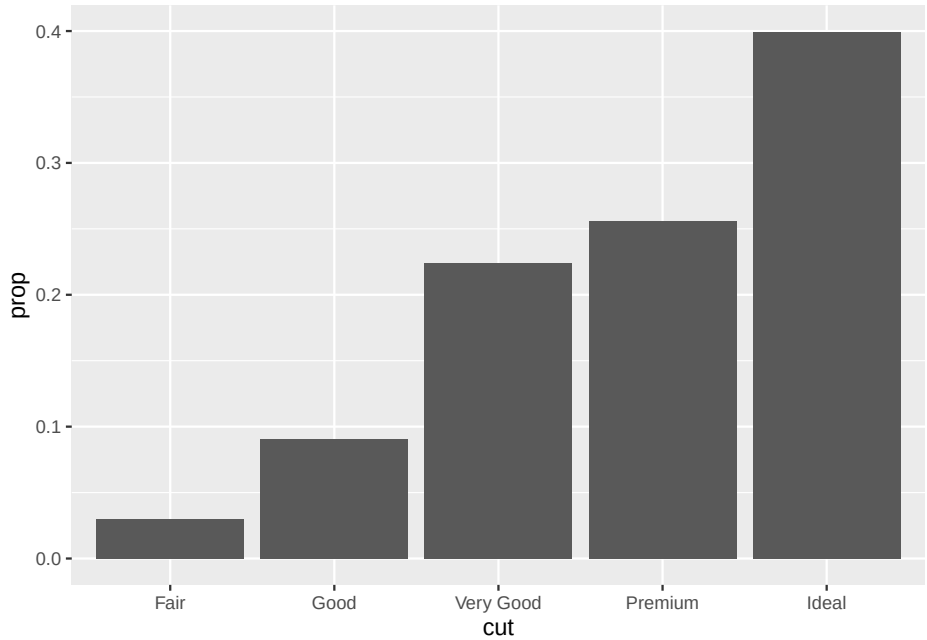
Use `stat = "identity"` to map the height of the bars to the raw values of the y variable.

```
ggplot(data = demo) +
  geom_bar(mapping = aes(x=cut, y=freq), stat = "identity")
```



Display a bar chart of proportions

```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = cut, y = ..prop.., group = 1))
```



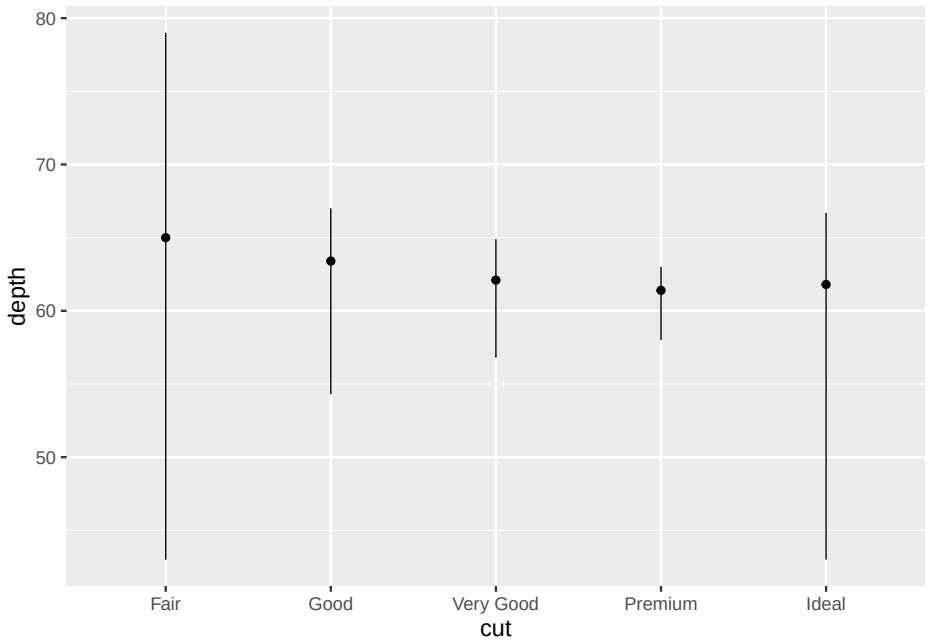
`geom_bar` has two calculated values: `count` and `prop`.

https://github.com/tidyverse/ggplot2/issues/2051

https://www.gl-li.com/2017/08/13/ggplot2-group-overrides-default-grouping/

Plot the stats

```
ggplot(data = diamonds) +
  stat_summary(
    mapping = aes(x = cut, y = depth),
    fun.min = min,
    fun.max = max,
    fun = median
  )
```



(diamonds is a small dataset that comes with the ggplot2 library)

Section 2

Facet subplots

Subplots

Subplots are helpful when you want to show different data presentations in a single view, e.g. a dashboard.

- `pairs()` produces a matrix of scatter plots for pairs of variables.
  - The variables must come from the same dataframe.
- `library(gridExtra)` arranges multiple plots in one view.
  - can be any data frames

What if we want to visually compare the same pair of variables, but over *a subset of the data*?

- use *subsetting* to extract multiple data frames?

Facets - subplots that display subsets of the data

- We have seen that one way to add additional variables is with aesthetics.
- Another way, particularly useful for categorical variables, is to split a plot into *facets*, i.e. *subplots*, each of which displays one subset of the data.

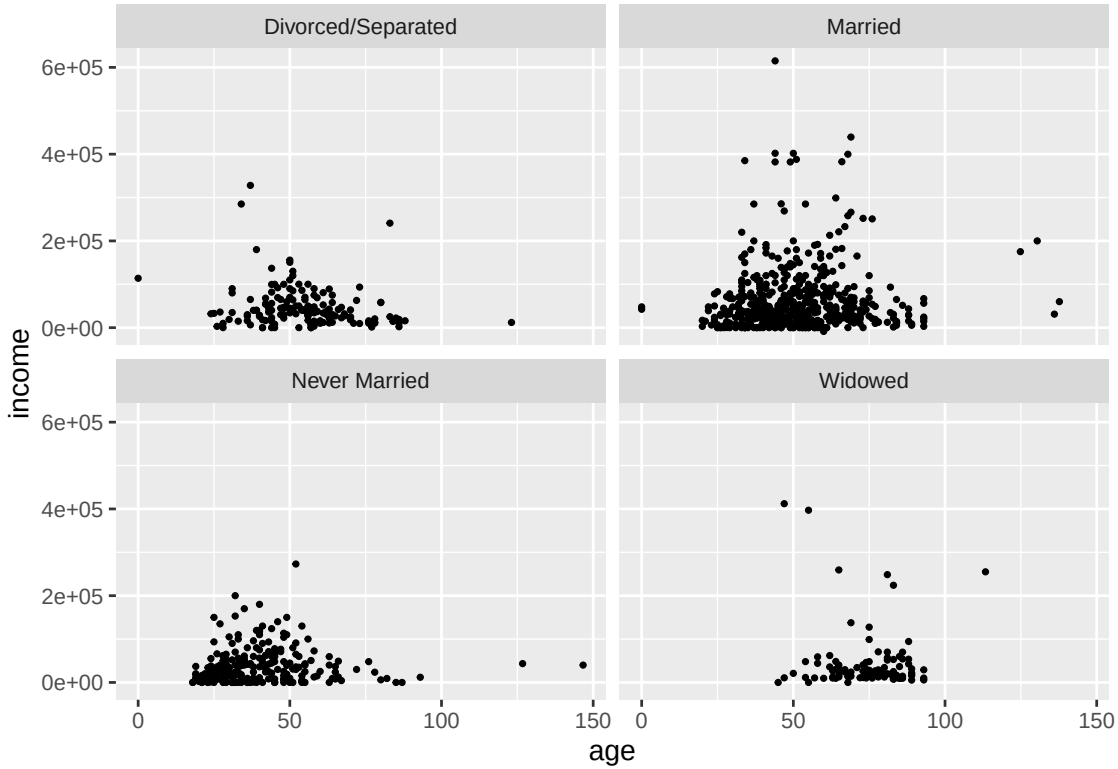
To facet your plot by a single variable, use `facet_wrap()`.

- The first argument of `facet_wrap()` should be a *formula*, created with `~` followed by a variable name.
- Here "formula" is the name of a data structure in R, not a synonym for "equation".
- The variable that you pass to `facet_wrap()` should be discrete.

```
facet_wrap(~ class, nrow = 2)
```

Facets - single variable example

```
ggplot(data = custdata) +
  geom_point(mapping = aes(x = age, y = income)) +
  facet_wrap(~ marital.stat, nrow = 2)
```

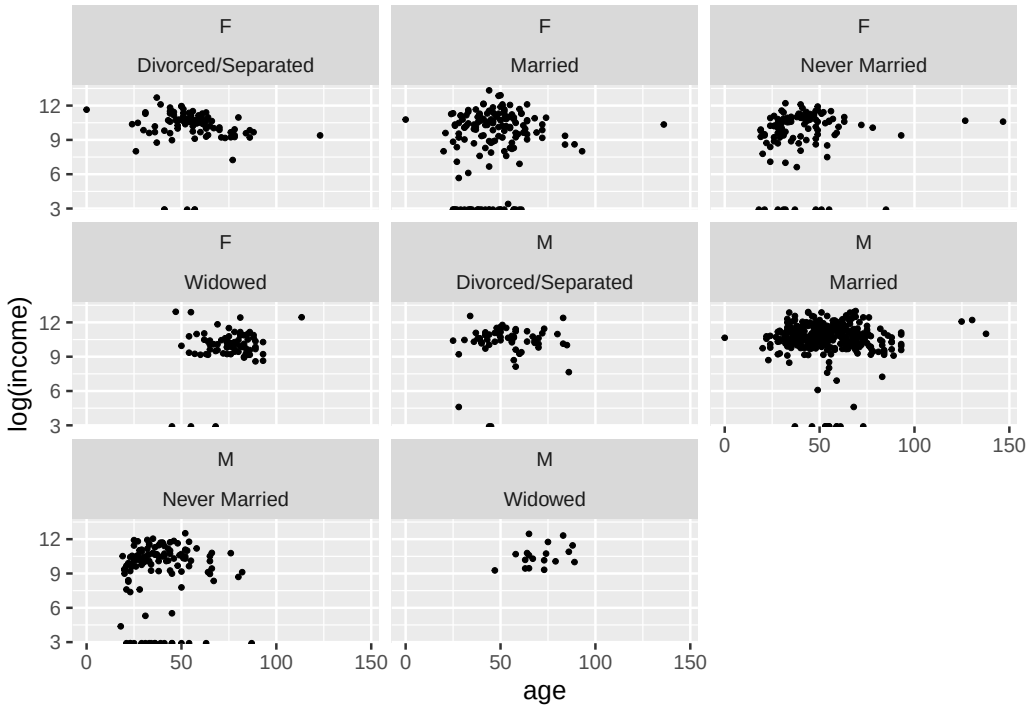


(Compare with the scatter plot for income against age and the hexbin diagram on earlier slides)

Facets - two variable example

Subsetting data using two variables.

```
ggplot(data = custdata) +
  geom_point(mapping = aes(x = age, y = log(income))) +
  facet_wrap( sex ~ marital.stat )
```



(Try `facet_wrap` to see the difference)

Section 3

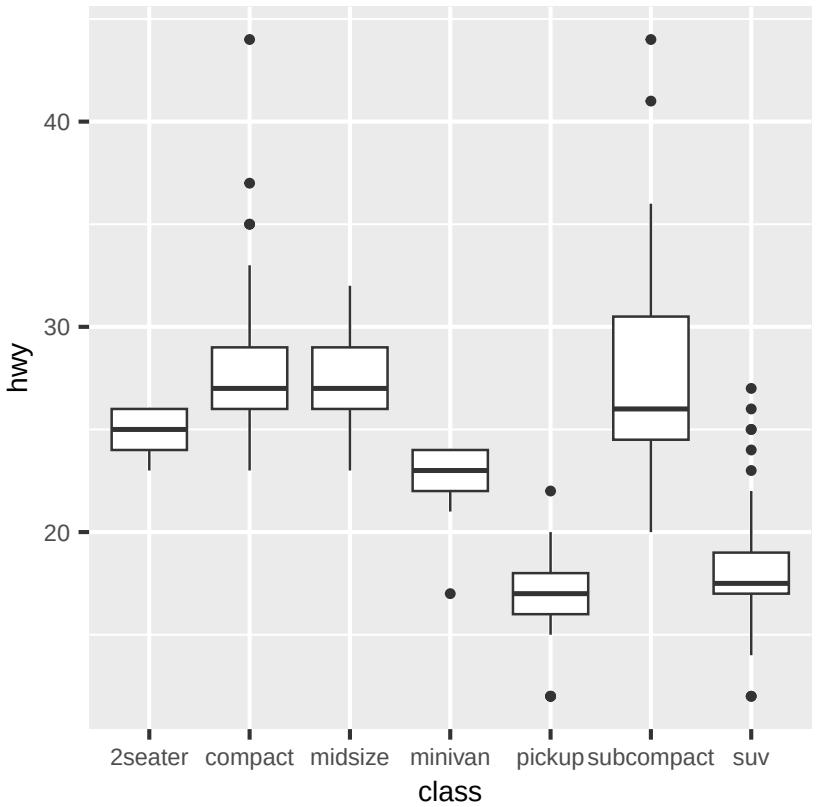
Coordinate systems

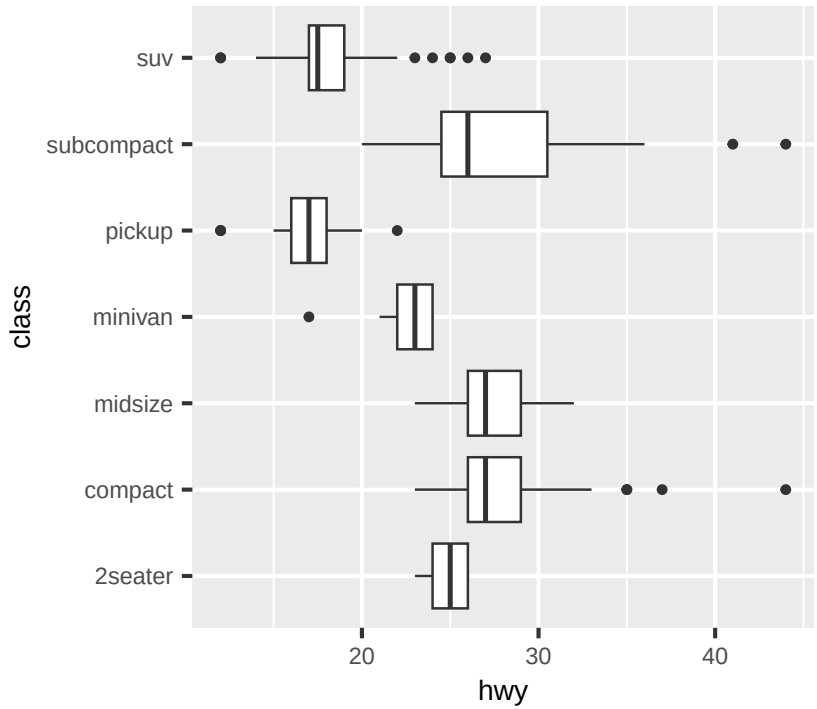
Coordinate flip `coord_flip()`

```
p1 <- ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot() +
  theme(text = element_text(size = 12), aspect.ratio=1)
p2 <- ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot() +
  coord_flip() +
  theme(text = element_text(size = 12), aspect.ratio=1)
```

Coordinate flip example

```
grid.arrange(p1, p2, ncol=2)
```



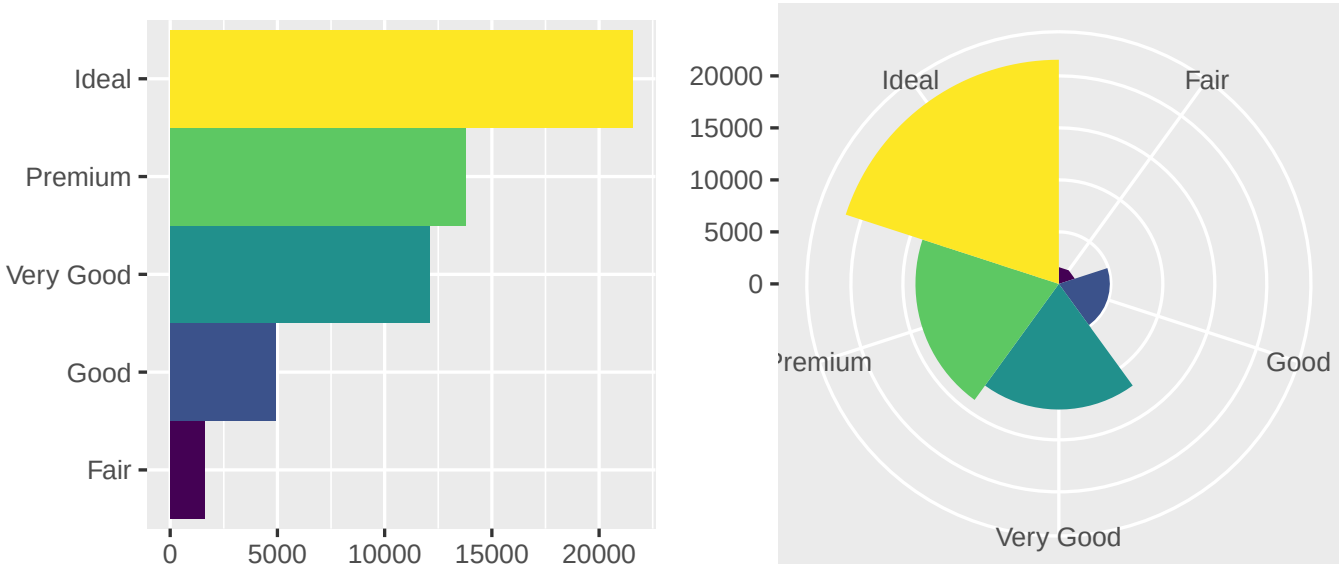


Polar coordinate

```
bar <- ggplot(data = diamonds) +
  geom_bar(
    mapping = aes(x = cut, fill = cut),
    show.legend = FALSE,
    width = 1
  ) +
  theme(aspect.ratio = 1) +
  labs(x = NULL, y = NULL)

p1 <- bar + coord_flip()
p2 <- bar + coord_polar()
```

Polar coordinate plot

```
grid.arrange(p1, p2, ncol=2)
```



Map Coordinates coord_quickmap()

```
library(maps)
w <- map_data("state")

p1 <- ggplot(w, aes(long, lat, group = group)) +
  geom_polygon(fill = "white", colour = "black")

p2 <- ggplot(w, aes(long, lat, group = group)) +
  geom_polygon(fill = "white", colour = "black") +
  coord_quickmap() # coord_sf() is now preferred
```

We can simplify the second assignment statement to:

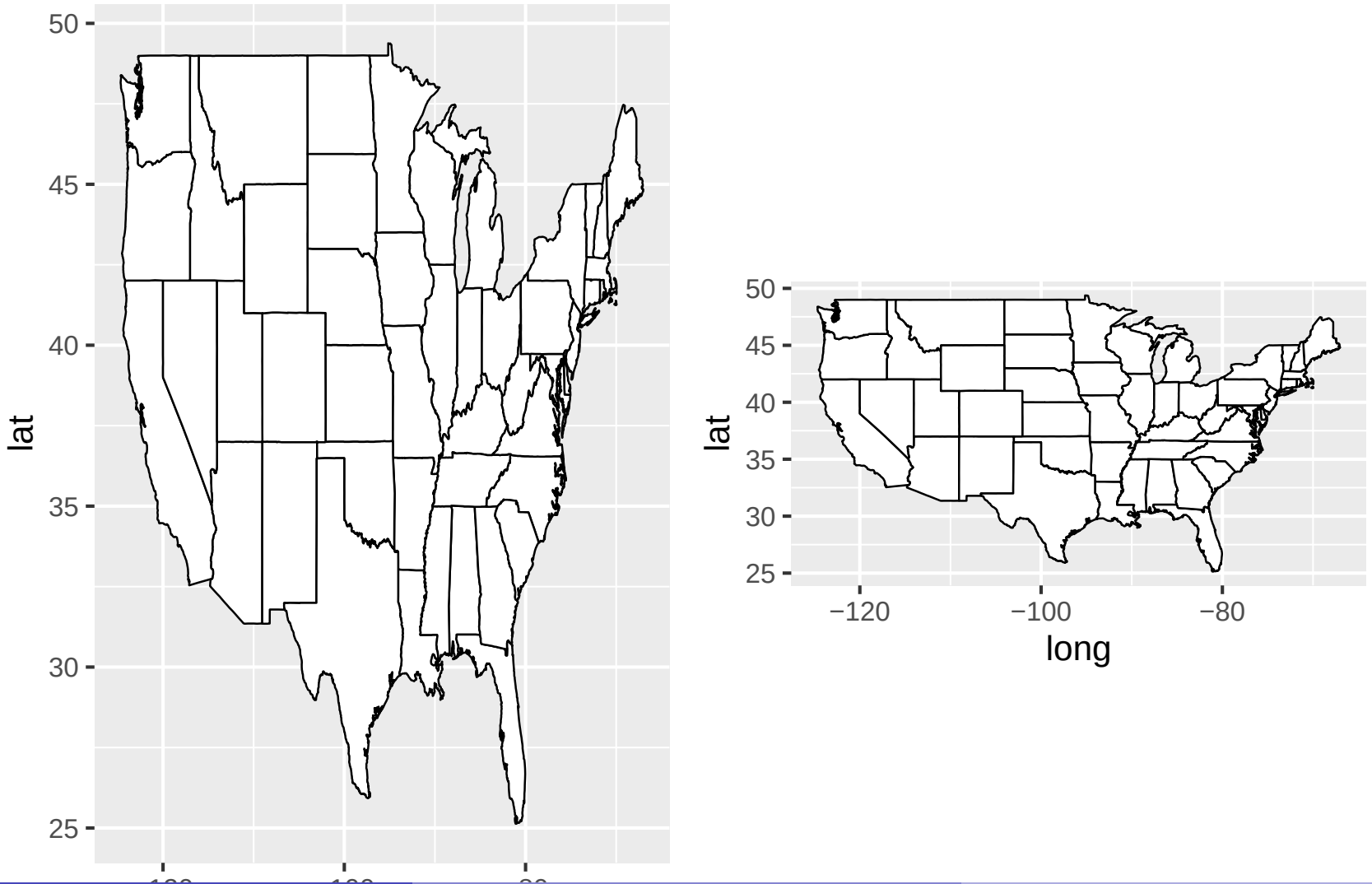
```
p2 <- p1 + coord_quickmap()
```

To use `coord_sf()`, we need

```
install.packages("sf")
```

Map example

```
grid.arrange(p1, p2, ncol=2)
```



Topics covered this Week

- The layer grammar of graphics
  - geom
  - aesthetic mapping
  - position adjustment
  - stat transformation
  - facets
  - coordinate systems
- What plots are suitable for two or more variables exploration?
- Interpreting plots for two or more variable exploration

References

- **Practical Data Science with R**, *Nina Zumel, John Mount*, Manning, 2nd Ed., 2020 (Chapter 3)
- **R for Data Science**, *Hadley Wickham, Garrett Grolemund*, O'Reilly, 2017 (Chapters 3&7)