Data visualization

Lecture 4

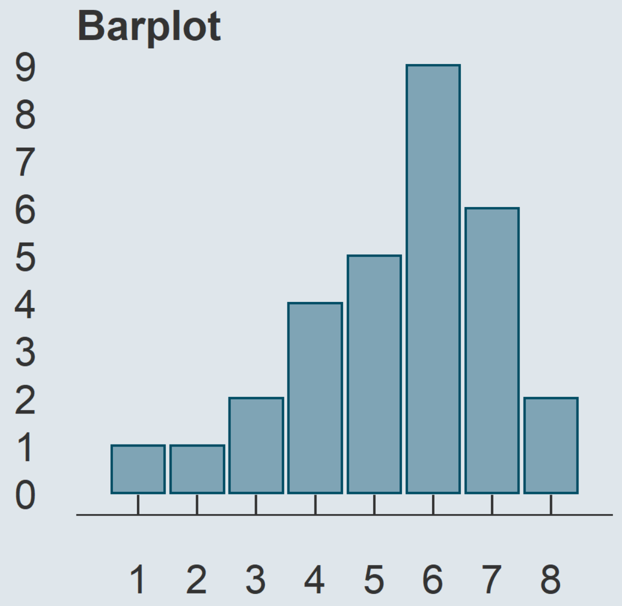
Louis SIRUGUE

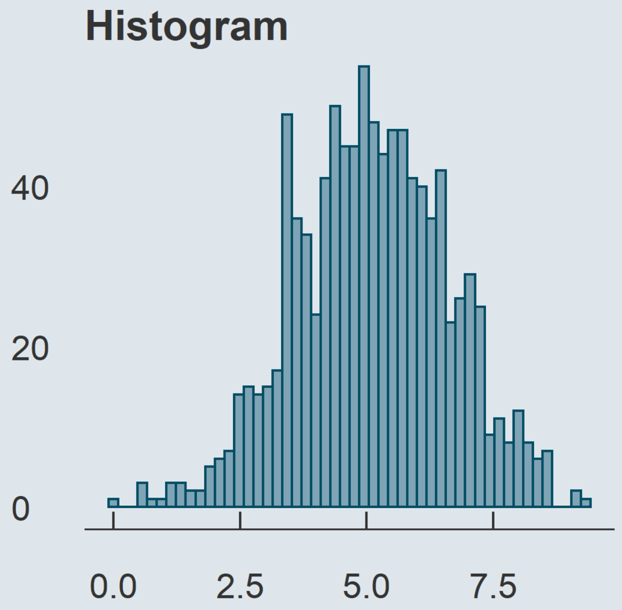
09/2021

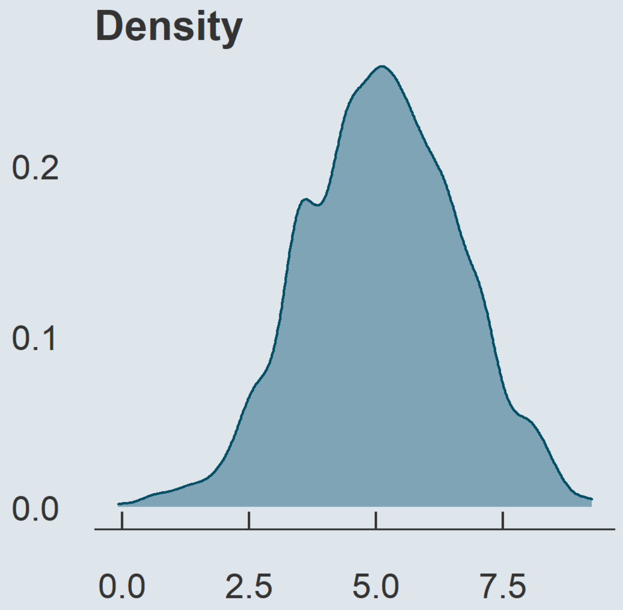
Last time we saw

1. Distributions

- The **distribution** of a variable documents all its possible values and how frequent they are





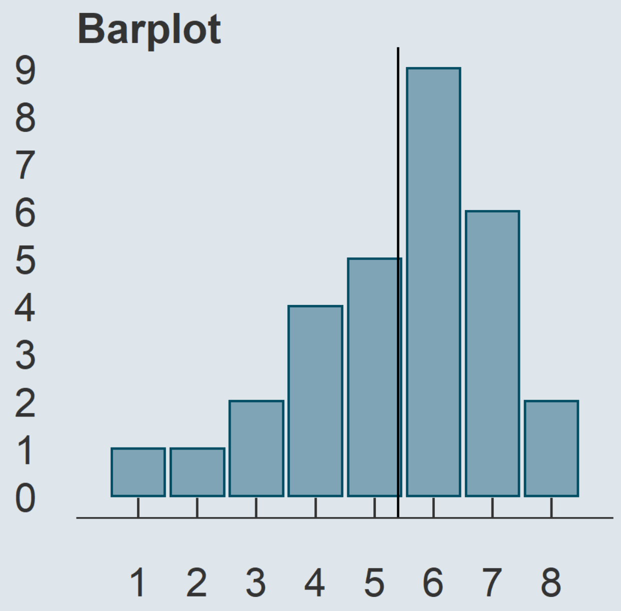


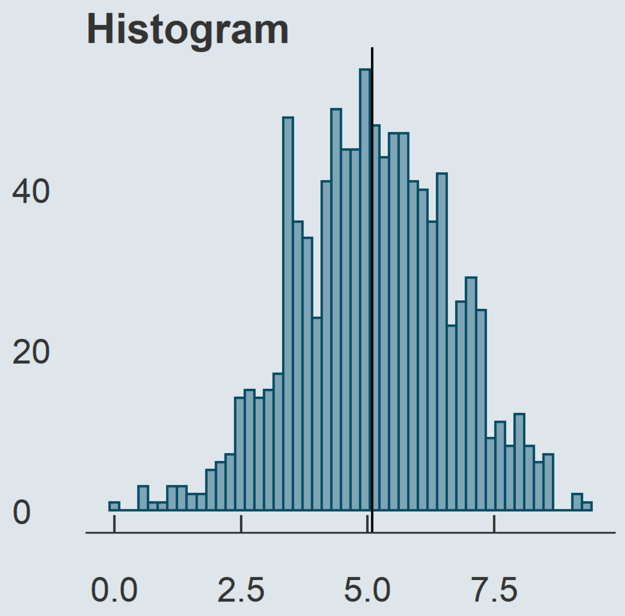
- We can describe a distribution with:

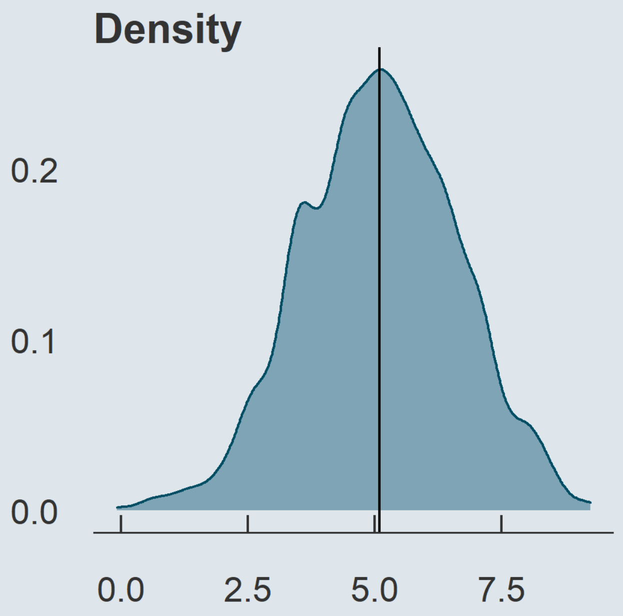
Last time we saw

1. Distributions

- The **distribution** of a variable documents all its possible values and how frequent they are





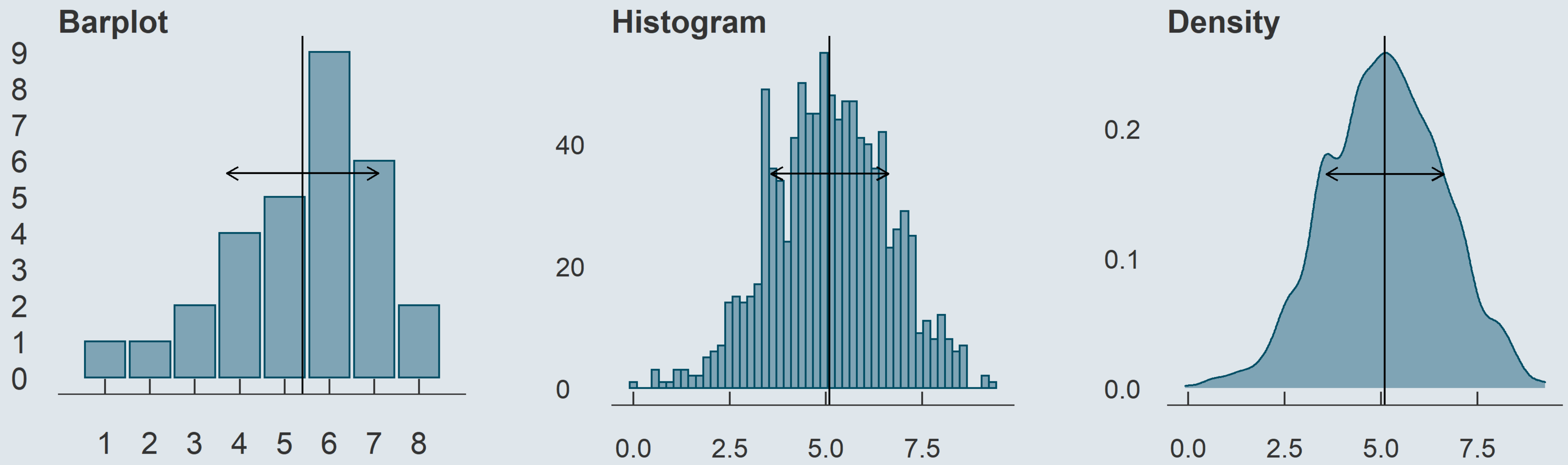


- We can describe a distribution with:
  - Its **central tendency**

Last time we saw

1. Distributions

- The **distribution** of a variable documents all its possible values and how frequent they are



- We can describe a distribution with:
  - Its **central tendency**
  - And its **spread**

Last time we saw

2. Central tendency

- The **mean** is the sum of all values divided by the number of observations

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

- The **median** is the value that divides the (sorted) distribution into two groups of equal size

$$\text{Med}(x) = \begin{cases} x[\frac{N+1}{2}] & \text{if } N \text{ is odd} \\ \frac{x[\frac{N}{2}]+x[\frac{N}{2}+1]}{2} & \text{if } N \text{ is even} \end{cases}$$

3. Spread

- The **standard deviation** is square root of the average squared deviation from the mean

$$\text{SD}(x) = \sqrt{\text{Var}(x)} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

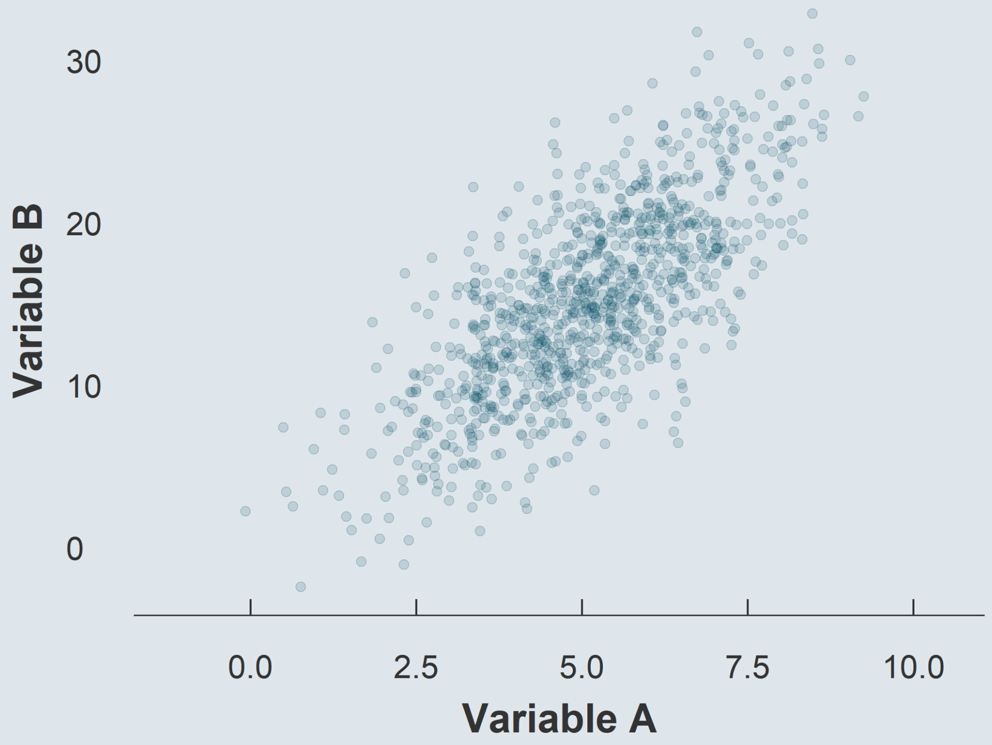
- The **interquartile range** is the difference between the maximum and the minimum value from the middle half of the distribution

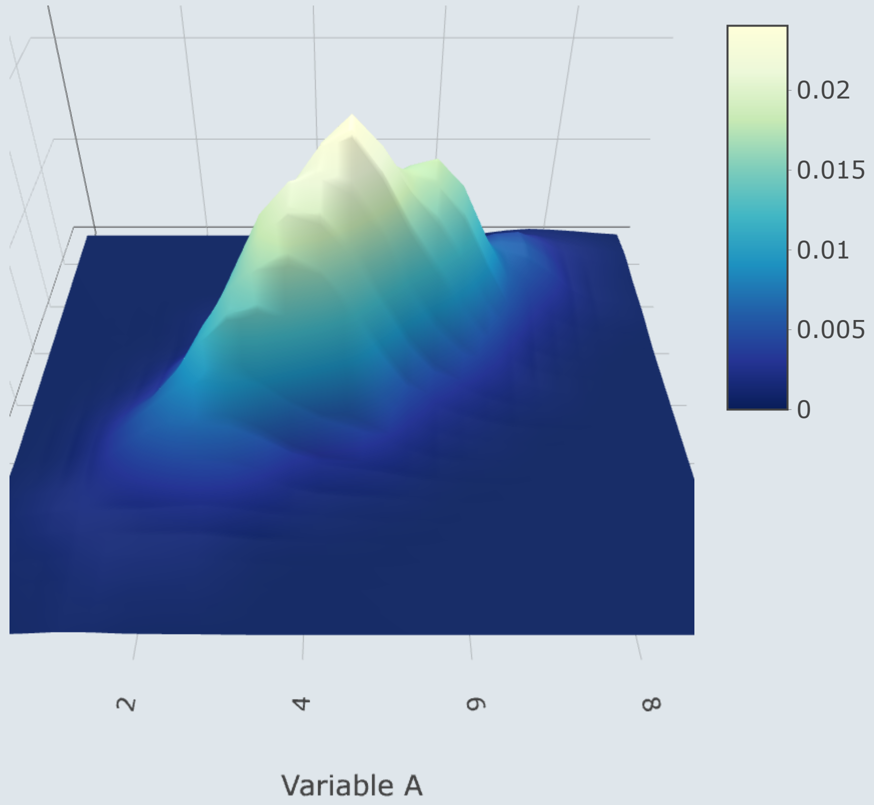
$$\text{IQR} = Q_3 - Q_1$$

Last time we saw

4. Joint distribution

- The **joint distribution** shows the possible values and associated frequencies for two variable simultaneously





Last time we saw

4. Joint distribution

→ ***When describing a joint distribution, we're interested in the relationship between the two variables***

- The **covariance** quantifies the joint deviation of two variables from their respective mean

$$\text{Cov}(x, y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})$$

- The **correlation** is the covariance of two variables divided by the product of their standard deviation

$$\text{Corr}(x, y) = \frac{\text{Cov}(x, y)}{\text{SD}(x) \times \text{SD}(y)}$$

Today we learn how to plot data

Today we learn how to plot data

1. The `ggplot()` function

1.1. Basic structure

`ggplot()` is the function that we're gonna use for all our plots throughout the course This whole lecture is dedicated to that function. The core arguments of the `ggplot()` function are the following.

- Data: the values to plot
- Mapping (aes, for aesthetics): the structure of the plot
- Geometry: the type of plot

Data and mapping should be specified within the parentheses, but the geometry and any other element should be added with a `+` sign

```
ggplot(data, aes) + geometry + anything_else
```

You can also apply the `ggplot()` function to your data with a pipe

```
data %>% ggplot(., aes) + geometry
```

1. The `ggplot()` function

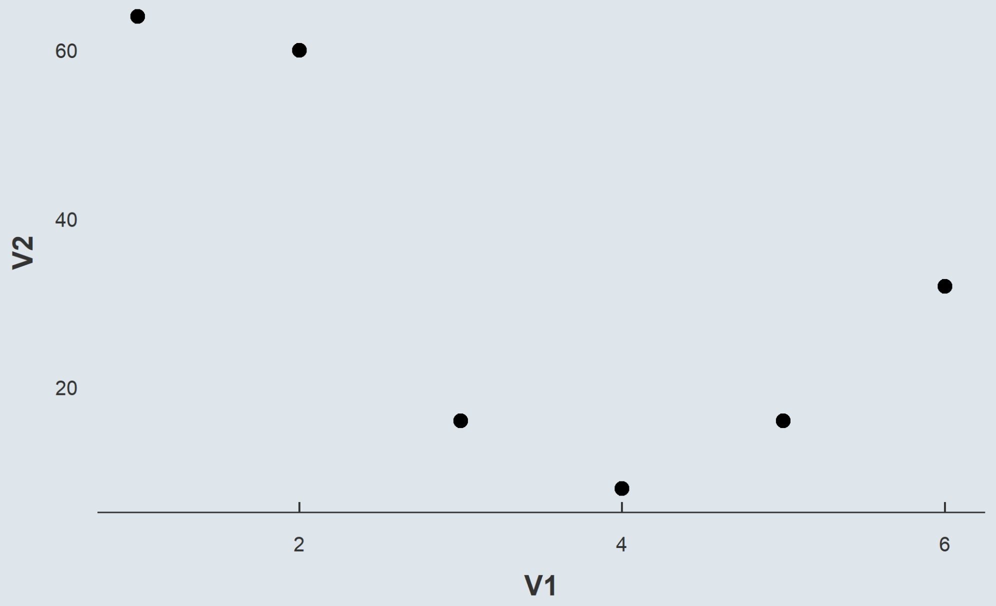
1.1. Basic structure: Example

```
test_data <- tibble(V1 = 1:6, V2 = c(64, 60, 16, 8, 16, 32))
ggplot(test_data, aes(x = V1, y = V2)) + geom_point(size = 3)
```

- We first specified our data:

V1	1	2	3	4	5	6
V2	64	60	16	8	16	32

- Then assigned `V1` to the x-axis and `V2` to the y-axis with `aes()`
- And chose the `point` geometry with a size equal to 3



1. The `ggplot()` function

1.2. Axes and legend

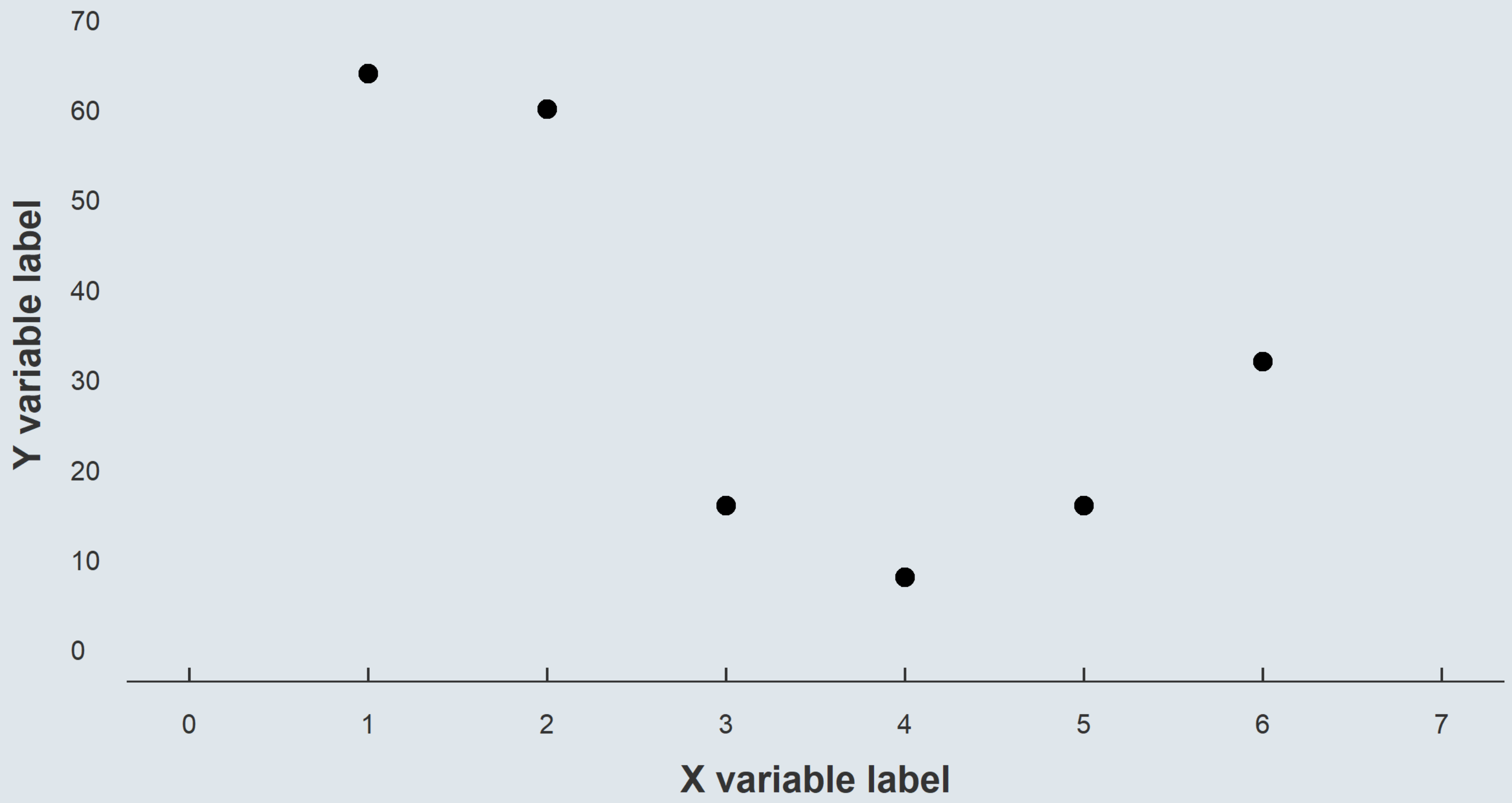
- Axes can be modified using with scale_A_B() functions, where
    - A should indicate the axis: x or y
    - B should indicate the variable type: continuous or discrete

- The following parameters can be modified in these scale functions:
    - **name:** The label of the corresponding axis
    - **limits:** Where the axis should start and end
    - **breaks:** Where to put ticks and values on the axis

```r
ggplot(test_data, aes(x = V1, y = V2)) + geom_point(size = 3) +
  scale_x_continuous(name = "X variable label", limits = c(0, 7), breaks = 0:7) +
  scale_y_continuous(name = "Y variable label", limits = c(0, 70), breaks = seq(0, 70, 10))
```

1. The `ggplot()` function

1.2. Axes and legend



1. The `ggplot()` function

1.2. Axes and legend

- In some cases you would convey information with other means than a position on axis, be it with the color, size of shape of a geometry. For instance if you have two groups:

```
test_data <- test_data %>%
  mutate(Group = paste("Group", c(1, 1, 2, 2, 2, 2)))
```

V1	V2	Group
1	64	Group 1
2	60	Group 1
3	16	Group 2
4	8	Group 2
5	16	Group 2
6	32	Group 2

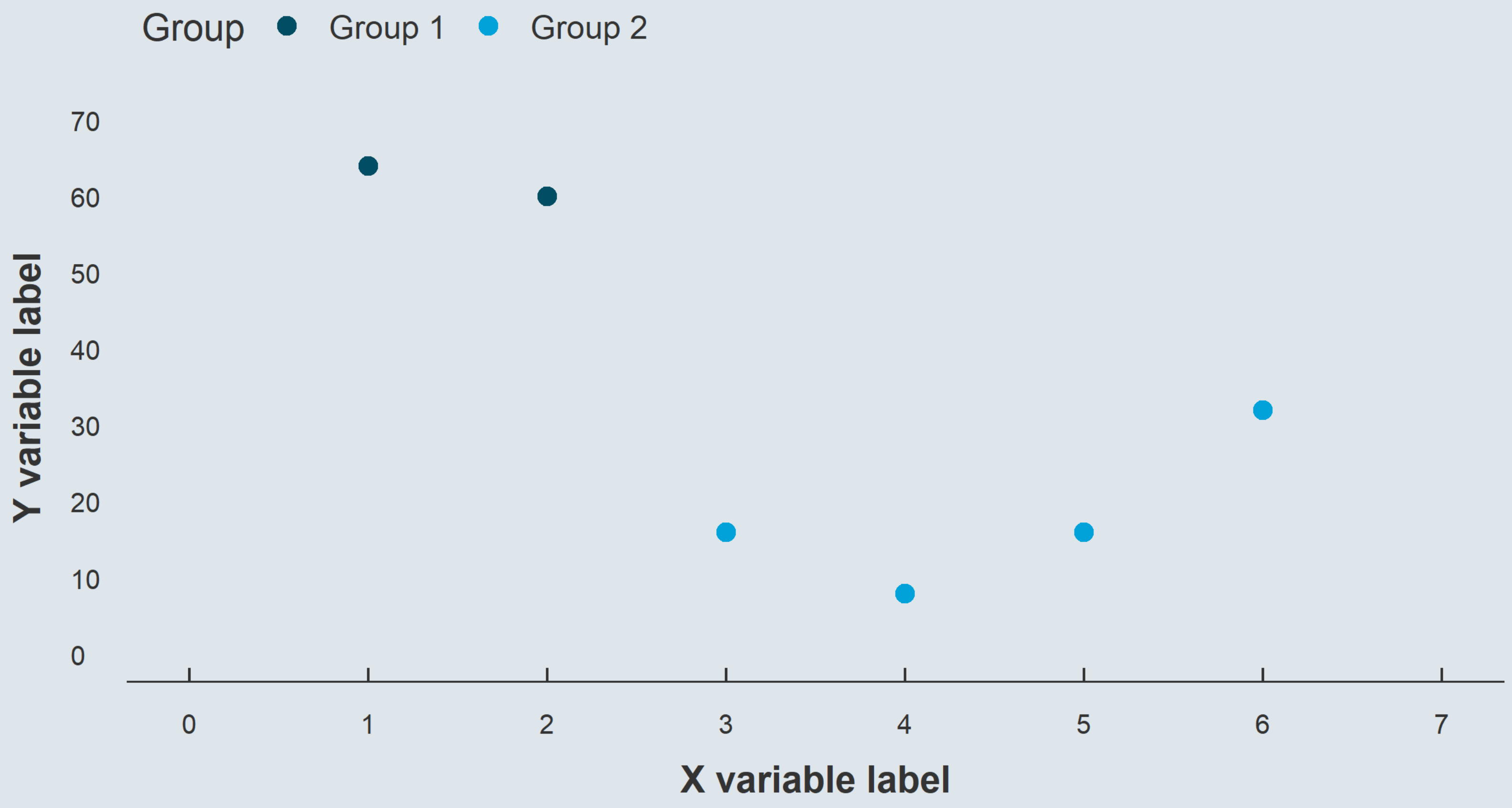
- Just as we assigned the two numeric variables to the x an y axis with aes, we have to assign the group variable to the 'color axis' with aes

```
ggplot(test_data, aes(x = V1, y = V2,
                      color = Group)) + ...
```

- But there is no proper 'color axis', that's why a legend will be generated

1. The `ggplot()` function

1.2. Axes and legend



1. The `ggplot()` function

1.3. Theme()

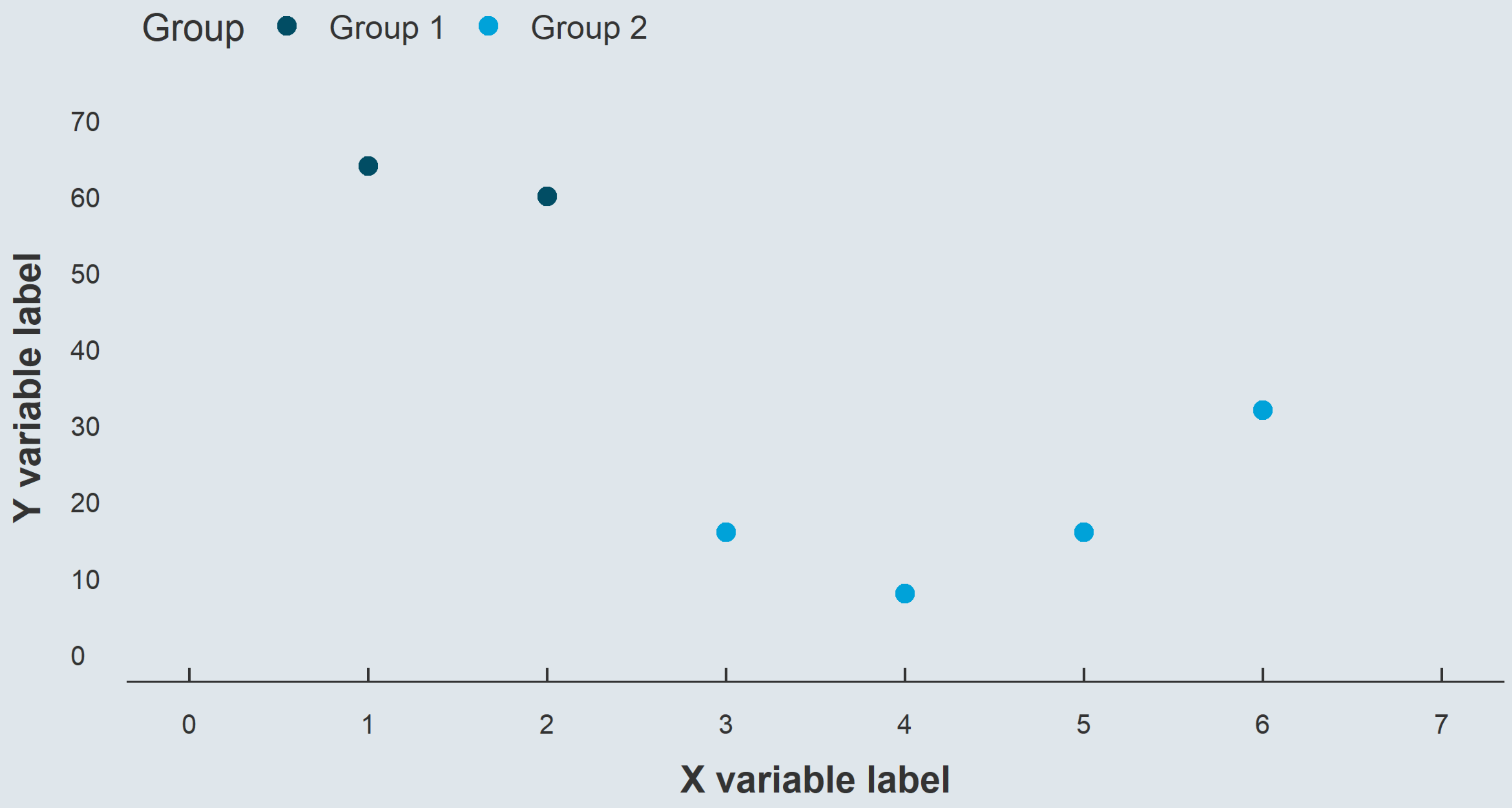
- ggplot() comes with a few default themes
  - `+ theme_bw()`
  - `+ theme_minimal()`
  - `+ theme_dark()`
  - ...

- You can choose the one you prefer and adjust the font size of the graph as follows

```
ggplot(test_data, aes(x = V1, y = V2, color = Group)) + geom_point(size = 3) +
  scale_x_continuous(name = "X variable label", limits = c(0, 7), breaks = 0:7) +
  scale_y_continuous(name = "Y variable label", limits = c(0, 70), breaks = seq(0, 70, 10)) +
  theme_minimal(base_size = 10)
```

1. The `ggplot()` function

1.3. Theme()



1. The `ggplot()` function

1.3. Theme()

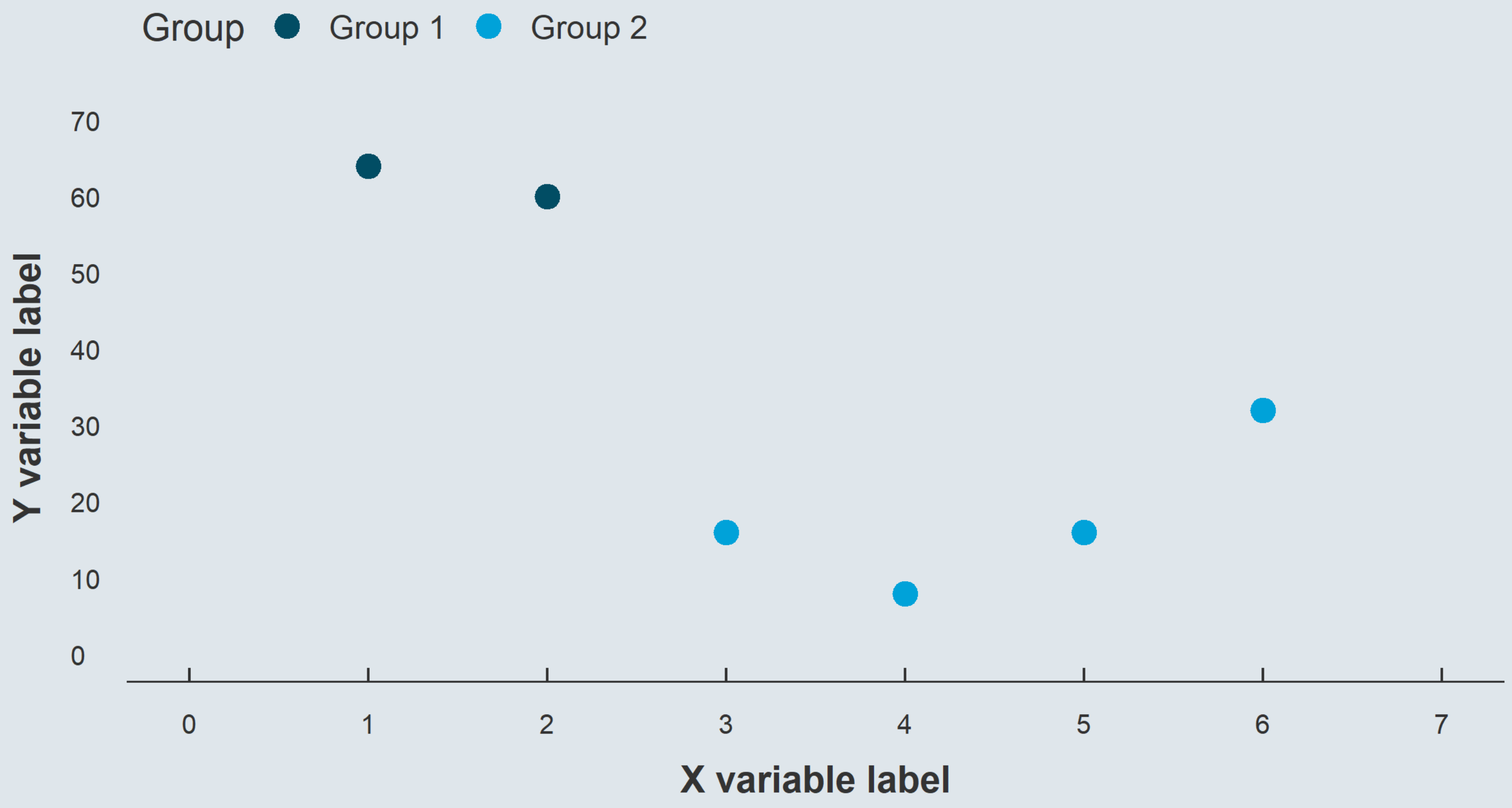
You can also custom your graph using the theme() function

- It allows to custom virtually anything
- Enter `?theme` to see the endless list of possible arguments
- Obviously we won't go through all of theme but here are a few

```r
ggplot(test_data, aes(x = V1, y = V2, color = Group)) + geom_point(size = 4) +
  scale_x_continuous(name = "X variable label", limits = c(0, 7), breaks = 0:7) +
  scale_y_continuous(name = "Y variable label", limits = c(0, 70), breaks = seq(0, 70, 10)) +
  theme_minimal(base_size = 10) + theme(
    plot.background = element_rect(fill = "#DFE6EB",     # Color of the background
                                   colour = "#DFE6EB"), # and of its border
    axis.line = element_line(size = rel(0.8)),          # Size of the axis lines
    legend.background = element_blank(),                # Remove legend background
    legend.position = "top",                            # Put legend at the top of the graph
    legend.direction = "horizontal",                    # Align its elements horizontally
    legend.justification = "left",                      # And left-align it
  )
```

1. The `ggplot()` function

1.3. Theme()



1. The `ggplot()` function

1.3. Theme()

- Geometries can also bu custom
  - **alpha:** opacity from 0 to 1
  - **color:** color of the geometry (for geometries that are filled such as bars, it will color the border)
  - **fill:** fill color for geometries such as bars
  - **size:** size of the geometry
  - **shape:** change shape for geometries like points
  - **linetype:** solid, dashed, dotted, etc., for line geometries
  - ...

- How to modify these elements depends on whether or not it is used to represent variations in variable
  - If not, the style should be assigned uniformly to the whole geometry and the style element can be directly modified in the geom function
  - If so, a variable should be assigned to that style element in aes. Adding parameters on this style assignment can be viewed as customizing an axis, and it should be done with a scale function

1. The `ggplot()` function

1.3. Custom geometry

→ **Case 1:** The style does not depend on the value of a variable

- The style element should be uniform across all data points, so it should be specified within the geometry function

```r
ggplot(test_data, aes(x = V1, y = V2)) +
  geom_point(color = "red", shape = 18)
```

→ **Case 2:** The style element depends on the value of a variable

- The style should depend on the value of the variable it has been assign to in `aes`, so just as for regular axes, modifications should take place in a `scale` function

```r
ggplot(test_data, aes(x = V1, y = V2, color = Group)) +
  scale_color_manual(name = "Group:", values = c("red", "blue"))  +
  geom_point(shape = 18)
```

1. The `ggplot` function

1.4. Annotation

- It is sometimes useful to annotate a graph so that certain things become more salient
  - Separate two groups with a dashed line
  - Add a few words somewhere for clarity
  - Circle a specific group of data points
  - Add labels to data points
  - ...

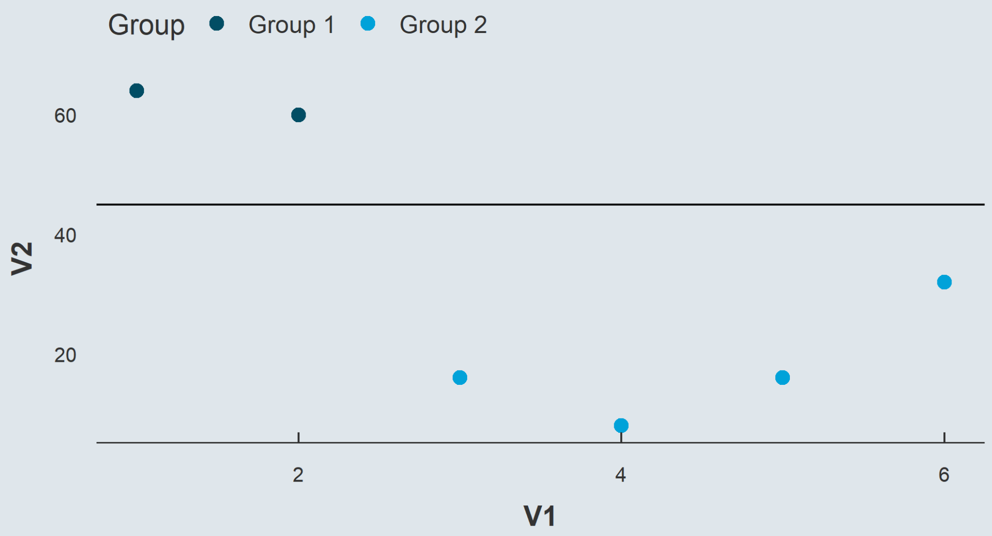
- Here are some handy functions for that purpose:
  - geom_hline()
  - geom_vline()
  - annotate()
  - geom_text()

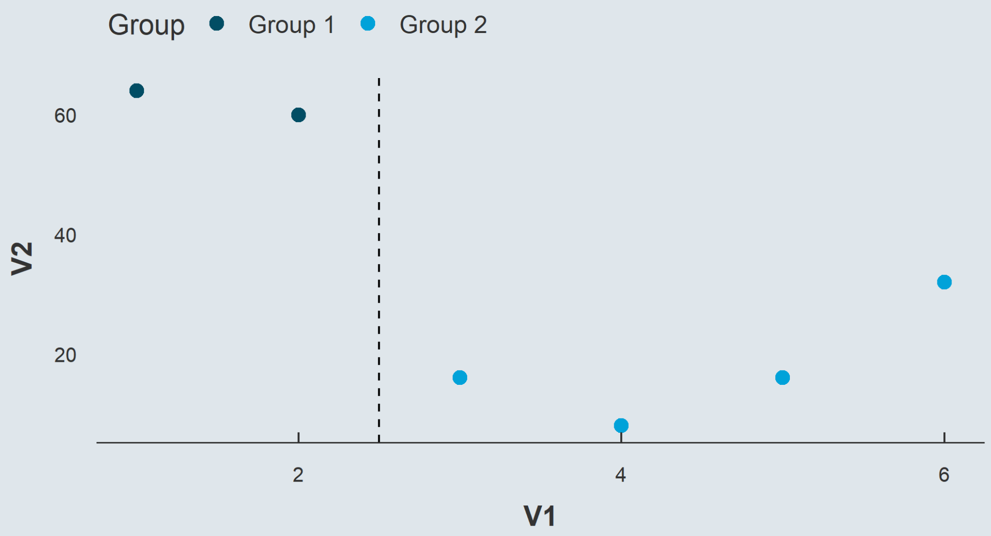
1. The ggplot function

1.4. Annotation: Adding lines

```
ggplot(test_data, aes(x = V1, y = V2,
                      color = Group)) +
  geom_point(size = 3) +
  geom_hline(yintercept = 45)
```

```
ggplot(test_data, aes(x = V1, y = V2,
                      color = Group)) +
  geom_point(size = 3) +
  geom_vline(xintercept = 2.5,
             linetype = "dashed")
```





1. The `ggplot` function

1.4. Annotation: Adding labels

Consider having relevant text information to display next to each data point

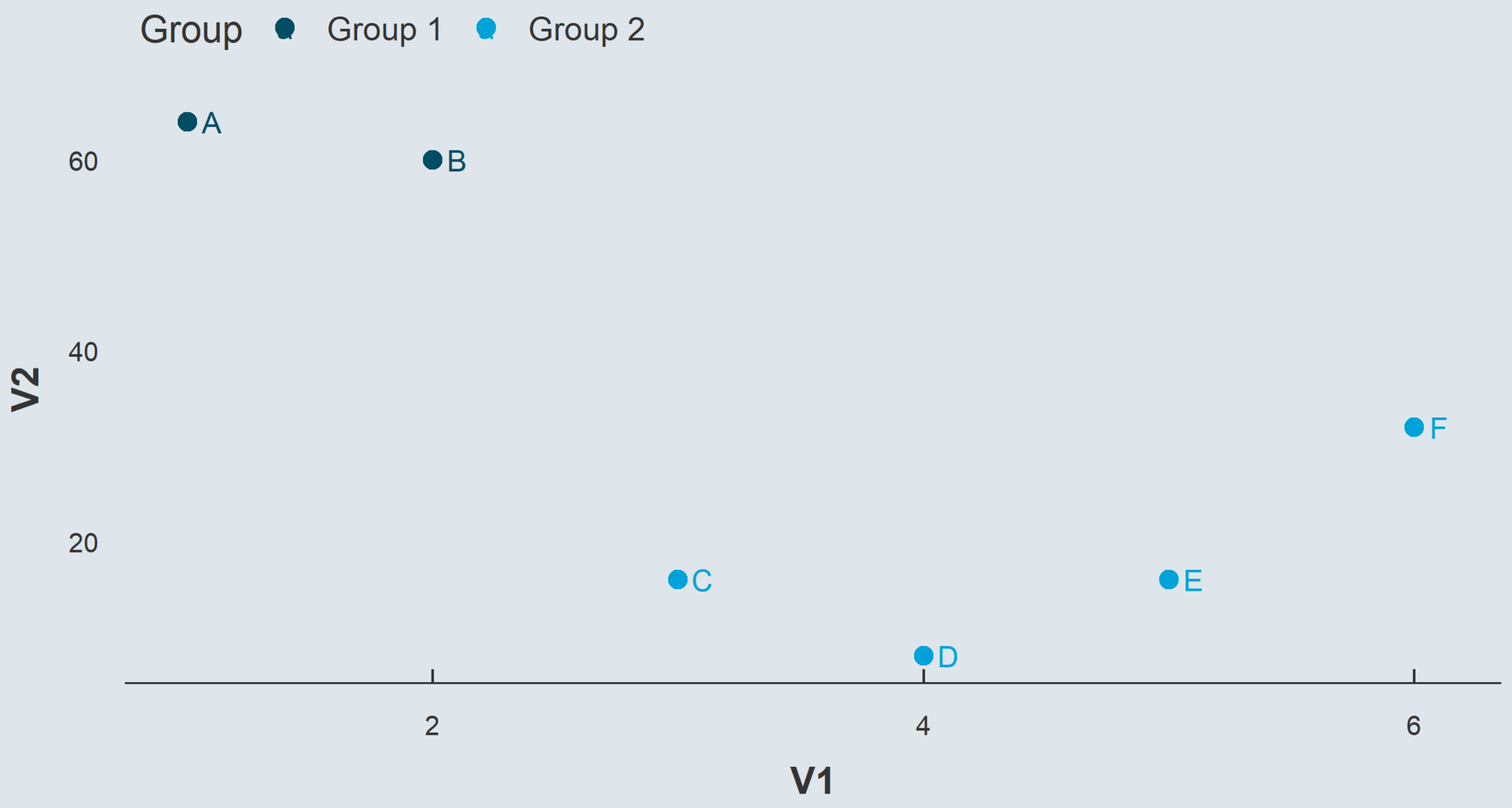
```
test_data <- test_data %>%
  mutate(Name = c("A", "B", "C", "D", "E", "F"))
```

- We can make use of the `geom_text` geometry to display each label next to the corresponding data point
- The label variable should be assigned in aes
- And the style elements that are common to each label can be specified in the `geom` function

V1	V2	Group	Name
1	64	Group 1	A
2	60	Group 1	B
3	16	Group 2	C
4	8	Group 2	D
5	16	Group 2	E
6	32	Group 2	F

1. The `ggplot` function

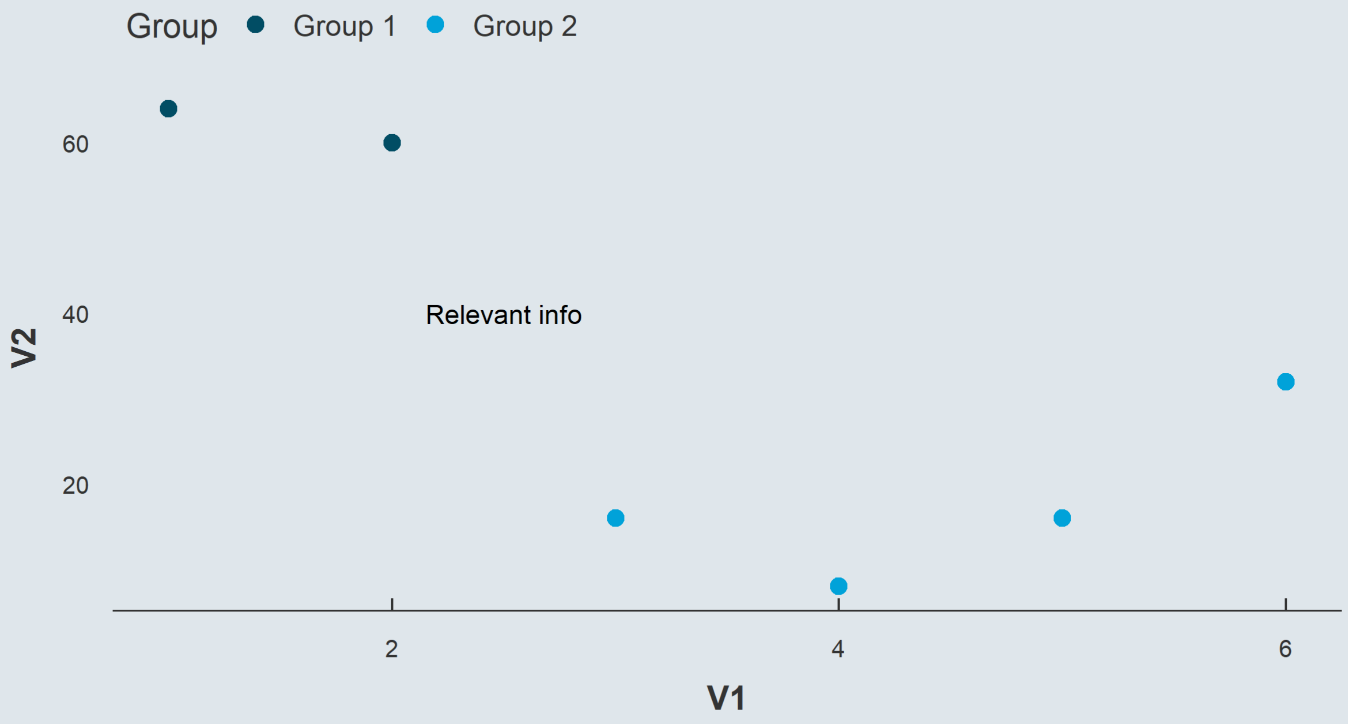
```r
ggplot(test_data, aes(x = V1, y = V2, color = Group, label = Name)) +
  geom_point(size = 3) + geom_text(nudge_x = .1) # Slightly right-shift labels
```



1. The `ggplot` function

1.4. Annotation: Adding text

```r
ggplot(test_data, aes(x = V1, y = V2, color = Group, label = Name)) +
  geom_point(size = 3) + annotate("text", x = 2.5, y = 40, label = "Relevant info")
```



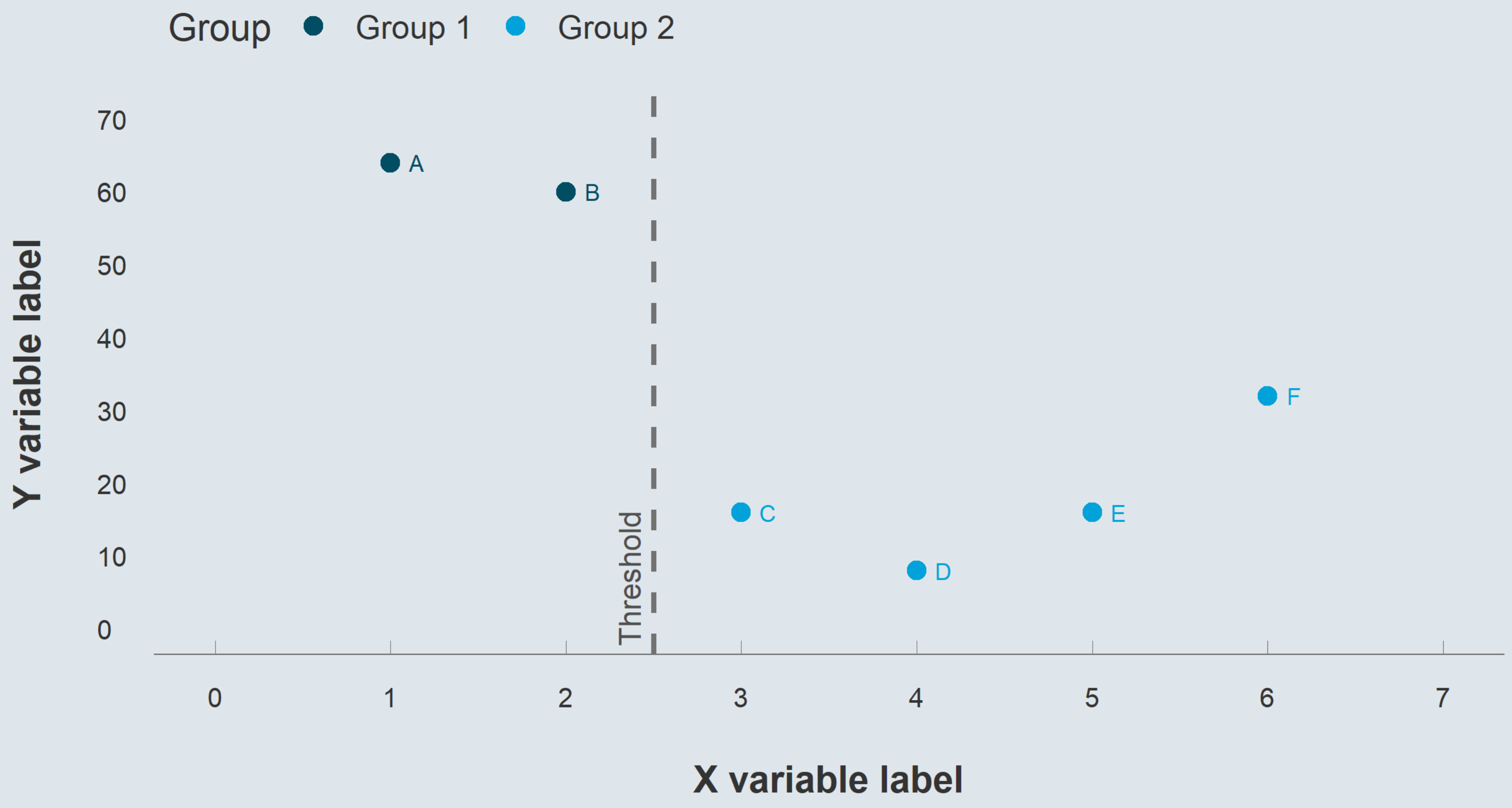
1. The `ggplot` function

Combining everything

```r
ggplot(test_data, aes(x = V1, y = V2, color = Group, label = Name)) +
  geom_point(size = 3) + geom_text(size = 3, show.legend = F, nudge_x = .15) +
  geom_vline(xintercept = 2.5, linetype = "dashed", size = 1, color = "#727272") +
  annotate("text", x = 2.36, y = 7, label = "Threshold", angle = 90, size = 4, color = "#727272") +
  scale_x_continuous(name = "X variable label", limits = c(0, 7), breaks = 0:7) +
  scale_y_continuous(name = "Y variable label", limits = c(0, 70), breaks = seq(0, 70, 10)) +
  theme_minimal(base_size = 10) +
        # Custom background
  theme(plot.background = element_rect(fill = "#DFE6EB", colour = "#DFE6EB"),
        # Custom axes
        axis.line.x = element_line(size = rel(0.8), color = "#727272"),
        axis.ticks.x = element_line(size = rel(0.4), color = "#727272"),
        axis.line.y = element_blank(), axis.ticks.y = element_blank(),
        axis.title.y = element_text(margin = margin(t = 0, r = 20, b = 0, l = 0)),
        axis.title.x = element_text(margin = margin(t = 20, r = 0, b = 0, l = 0)),
        # Custom legend
        legend.background = element_blank(), legend.position = "top",
        legend.direction = "horizontal", legend.justification = "left")
```

1. The `ggplot` function

Combining everything



Overview

Overview

Let's give it a go with actual data

Overview of the data

- We're gonna make use of two datasets:
  - A dataset on birth country of immigrants in France at the year/country of birth level (source: INSEE)
  - A dataset of economic indicators at the country level (source: OECD)

```
insee_data <- read.csv("insee_data.csv")
head(insee_data, 5)
```

country	continent	year	n_immig
Spain	Europe	2006	270
Spain	Europe	2007	263
Spain	Europe	2008	257
Spain	Europe	2009	252
Spain	Europe	2010	248

- The data on birth country of immigrants is in long format:
  - The number of immigrants from a given country in a given year is documented by the variable n_immig (thousands)
  - Country of birth is decomposed into 17 countries/regions
  - The data span from 2006 to 2020
  - There are 255 observations (17 countries $\times$ 15 years)

Let's give it a go with actual data

Overview of the data

- We're gonna make use of two datasets:
  - A dataset on birth country of immigrants in France at the year/country of birth level (source: INSEE)
  - A dataset of economic indicators at the country level (source: OECD)

```r
oecd_data <- read.csv("oecd_data.csv")
head(oecd_data, 5)
```

country	country_iso	continent	n_gen	gini	ratio9010	unemp	f_unemp	m_unemp
Argentina	ARG	South America	6	NA	NA	NA	NA	NA
Australia	AUS	Oceania	4	0.33	4.3	5.71	5.78	5.65
Austria	AUT	Europe	5	0.28	3.5	6.45	5.81	7.04
Belgium	BEL	Europe	4	NA	NA	NA	NA	NA
Brazil	BRA	South America	9	NA	NA	NA	NA	NA

Let's give it a go with actual data

Overview of the data

- The data on country-level economic indicators contains the following variables
  - `n_gen`: An estimate of the number of generations it would take for those born in low-income families to approach the mean income in their society
  - `gini`: The Gini Index of income inequality, ranging from 0 (everybody has the same income) to 1 (a single individual earns everything and everybody else earns nothing)
  - `ratio9010`: The 90/10 income inequality ratio, representing how many times larger is income at the 90th percentile compared to income at the 10th percentile
  - `unemp`: The unemployment rate defined as the share of people of working age who are without work, are available for work, and have taken specific steps to find work
  - `f_unemp` / `m_unemp`: Unemployment rate for females and males
- Their are many `NA`s we'll need to be careful with that

2. Bars and lines

What they're useful for

- Representing the distribution of a variable
  - If it is discrete: the number of observations per category
    - Ex: The number of available countries per continent
  - If it is continuous: how many observations per bin (recall the histogram)
    - Ex: The distribution of the unemployment rate across countries

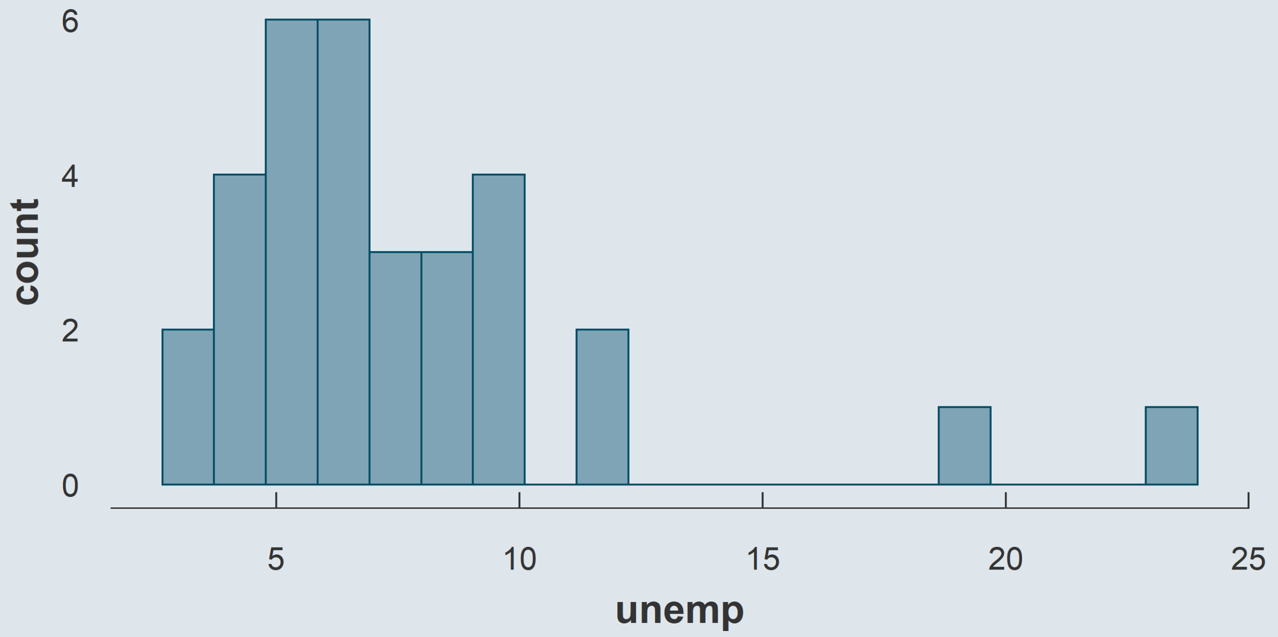
- Showing the value of a continuous variable y for each value of a discrete variable x
  - Ex: The number of generations needed for those born to low income families to converge to approach mean income for each country
  - Ex: The unemployment rate for males and females in each country

- Documenting the share of each category of a discrete variable y for each value of a discrete variable x
  - The distribution of immigrants' countries of birth for each year since 2006

2. Bars and lines

2.1. Histograms

```r
ggplot(oecd_data %>% filter(!is.na(unemp)), aes(x = unemp)) +
  geom_histogram(fill = "#6794A7", color = "#014D64", alpha = .8, bins = 20) +
  theme_minimal(base_size = 16)
```

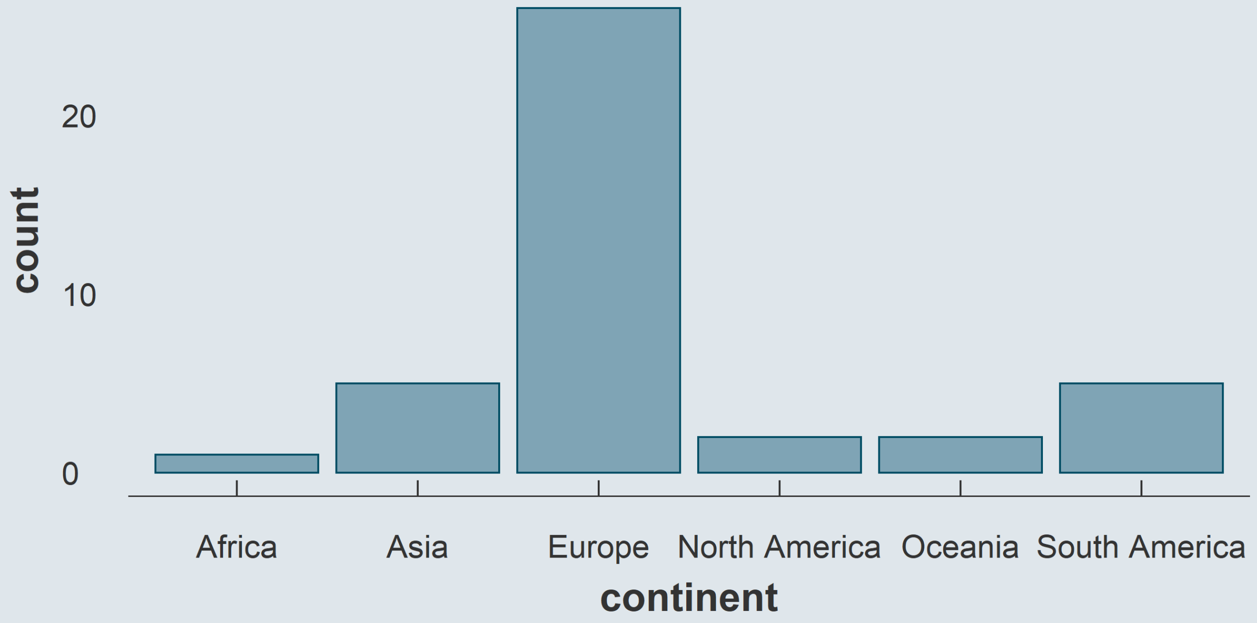


- The dedicated geometry is `geom_histogram()`
  - You just have to specify the x variable you want the histogram of in aes
  - And indicate either the number of `bins=` or the `binwidth=` (see previous lecture)

2. Bars and lines

2.2. geom_bar(stat = "count") for discrete distribution

```r
ggplot(oecd_data, aes(x = continent)) +
  geom_bar(stat = "count", fill = "#6794A7", color = "#014D64", alpha = .8) +
  theme_minimal(base_size = 16)
```

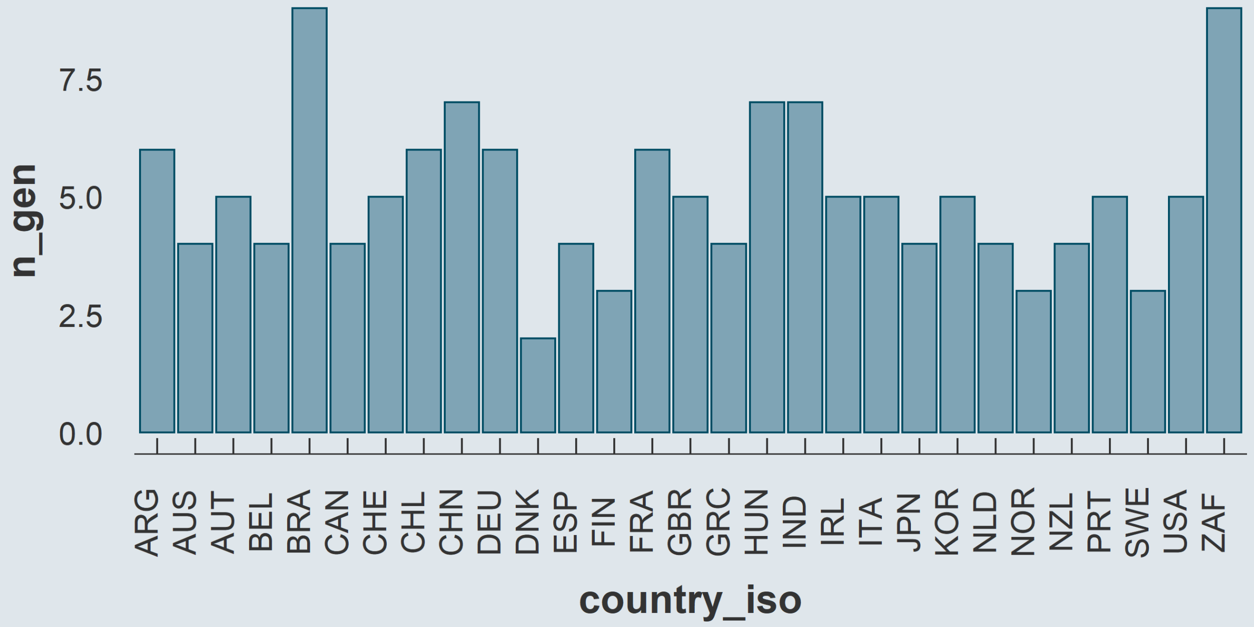


- The `stat = "count"` argument indicates that we want the height of each bar to represent the number of observations for each category of the x variable
  - So no need to set a y variable in aes

2. Bars and lines

2.2. geom_bar(stat = "identity") for y values

```r
ggplot(oecd_data %>% filter(!is.na(n_gen)), aes(x = country_iso, y = n_gen)) +
  geom_bar(stat = "identity", fill = "#6794A7", color = "#014D64", alpha = .8) +
  theme_minimal(base_size = 16) + theme(axis.text.x = element_text(angle = 90))
```

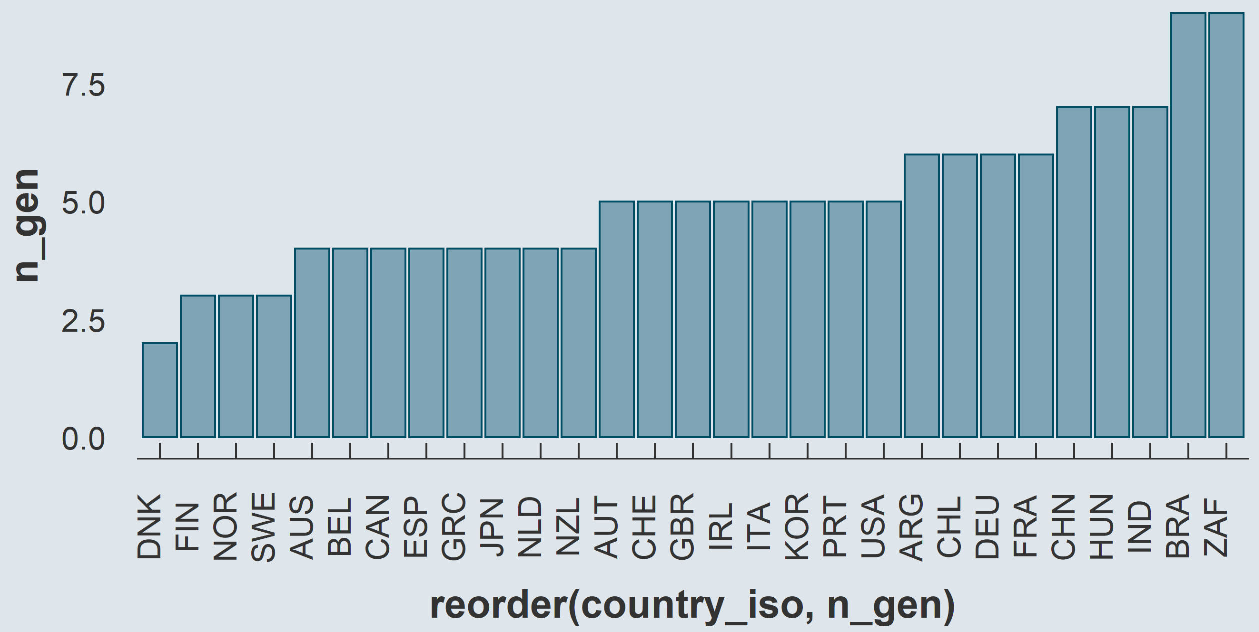


- The `stat = "identity"` argument indicates that we want the height of each bar to represent the value of `y` for each category of the x variable
  - By default the `x` variable is sorted alphabetically, not by value of `y`

2. Bars and lines

2.2. reorder() in aes() to sort bars

```
ggplot(oecd_data %>% filter(!is.na(n_gen)), aes(x = reorder(country_iso, n_gen), y = n_gen)) +
  geom_bar(stat = "identity", fill = "#6794A7", color = "#014D64", alpha = .8) +
  theme_minimal(base_size = 16) + theme(axis.text.x = element_text(angle = 90))
```



- We need to pass the `country_iso` variable through the `reorder()` function and indicate according to which variable it should be sorted
  - Just add `-` before `n_gen` to sort in decreasing order

2. Bars and lines

2.2. geom_bar(position = "dodge") to compare groups

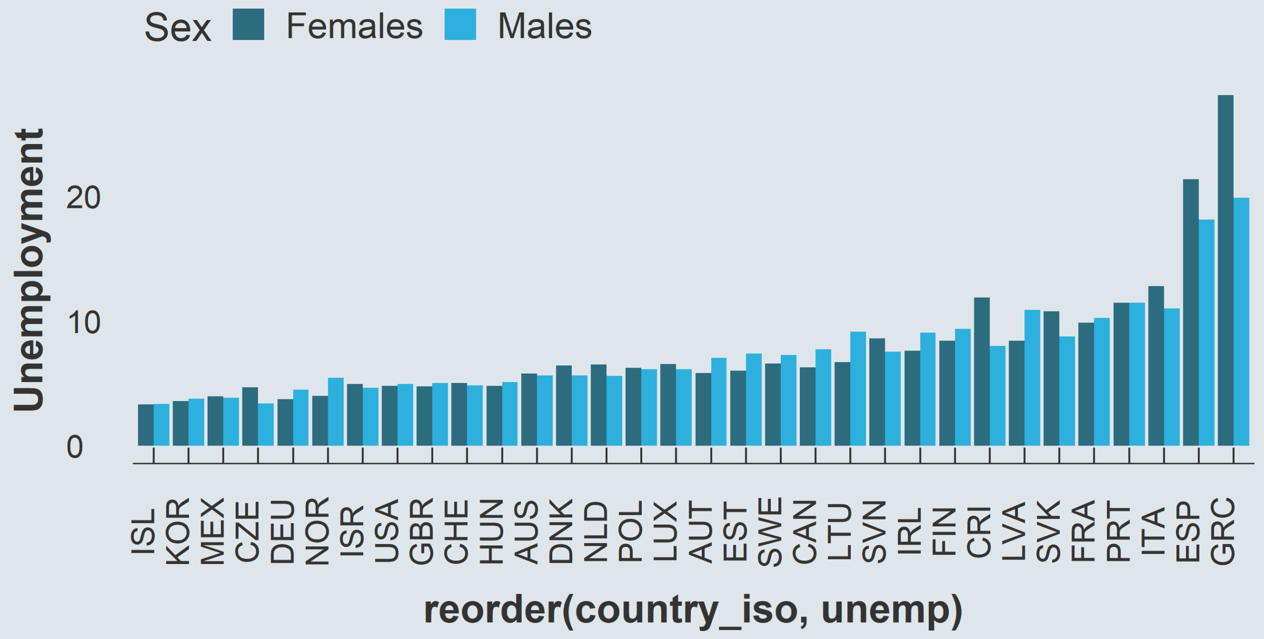
- In some cases we want to have the bars for different groups side to side for each category of an x variable. Ex: unemployment rates for males and females by country.
  - To do so, the grouping variable (here, males/females) should be specified in aes.
  - This implies that our data should be in long format: unemployment rates for males and females should not be side to side in two columns but on top of each other in a single column

```r
unemp_long <- oecd_data %>%
  # Pivot the gender-specific unemployment rates to the long format
  pivot_longer(c(m_unemp, f_unemp), names_to = "Sex", values_to = "Unemployment") %>%
  # Attribute better labels than original variable names to the Sex variable
  mutate(Sex = case_when(Sex == "m_unemp" ~ "Males", Sex == "f_unemp" ~ "Females")) %>%
  # Keep only non-missing observations
  filter(!is.na(Unemployment))
```

2. Bars and lines

2.2. geom_bar(position = "dodge") to compare groups

```
ggplot(unemp_long, aes(x = reorder(country_iso, unemp), y = Unemployment, fill = Sex)) +
  geom_bar(stat = "identity", alpha = .8, position = "dodge") +
  theme_minimal(base_size = 16) + theme(axis.text.x = element_text(angle = 90))
```



- We specify in aes that the fill color should be different according to the value of the variable `Sex`
  - `position = "dodge"` then puts side to side the bars of the grouping variable, here Sex, for each category of the x variable

2. Bars and lines

2.3. geom_bar(position = "stack") for evolutions of discrete distributions

In some cases it can be interesting to see how a distribution evolves over time or between groups.

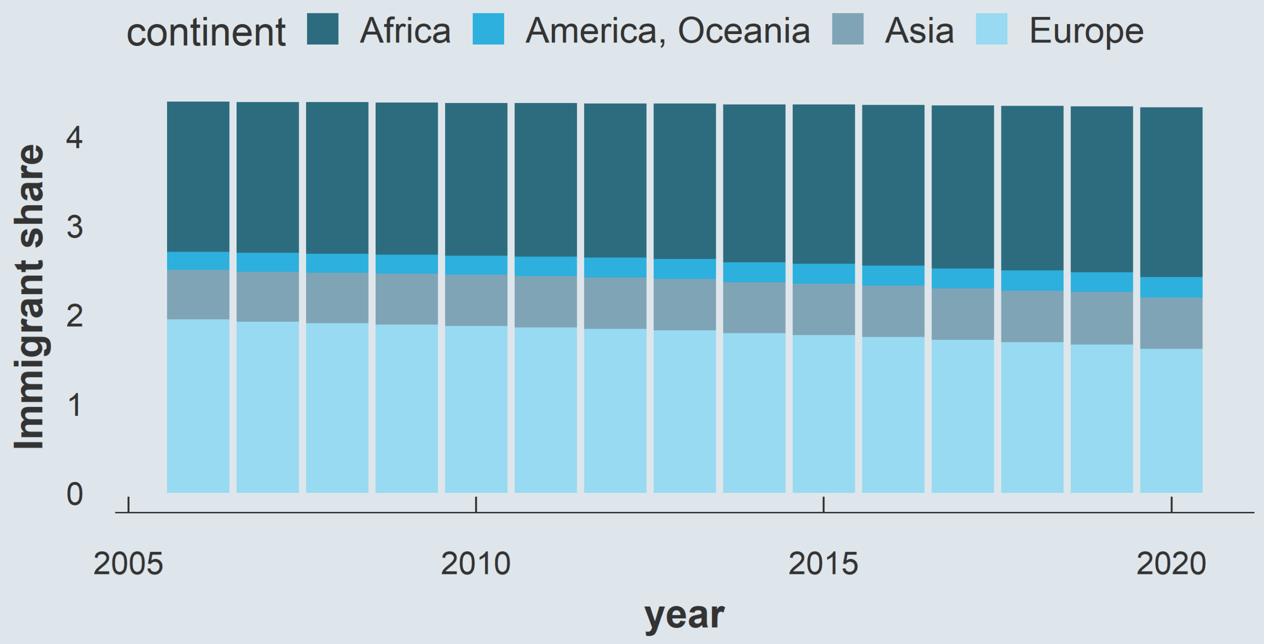
- One way to do it when the variable is discrete is to stack bars on top of others to see how relative share of each category evolves over time. Ex: the distribution of continent of birth of immigrants for each year.
  - To do so, the grouping variable (here, continent) should be specified in aes.
- Before plotting we should compute the share of each continent in the yearly number of immigrants

```r
stack_immig <- insee_data %>%
  # Next modifications should be done separately by value of year
  group_by(year) %>%
  # Compute the yearly number of immigrants
  mutate(tot_immig = sum(n_immig)) %>%
  # Next modifications should be done separately by value of year & continent
  group_by(year, continent) %>%
  # Compute the share of each continent in the yearly number of immigrants
  summarize(`Immigrant share` = sum(n_immig) / tot_immig)
```

2. Bars and lines

2.3. geom_bar(position = "stack") for evolutions of discrete distributions

```
ggplot(stack_immig, aes(x = year, y = `Immigrant share`, fill = continent)) +
  geom_bar(stat = "identity", alpha = .8, position = "stack") +
  theme_minimal(base_size = 16)
```

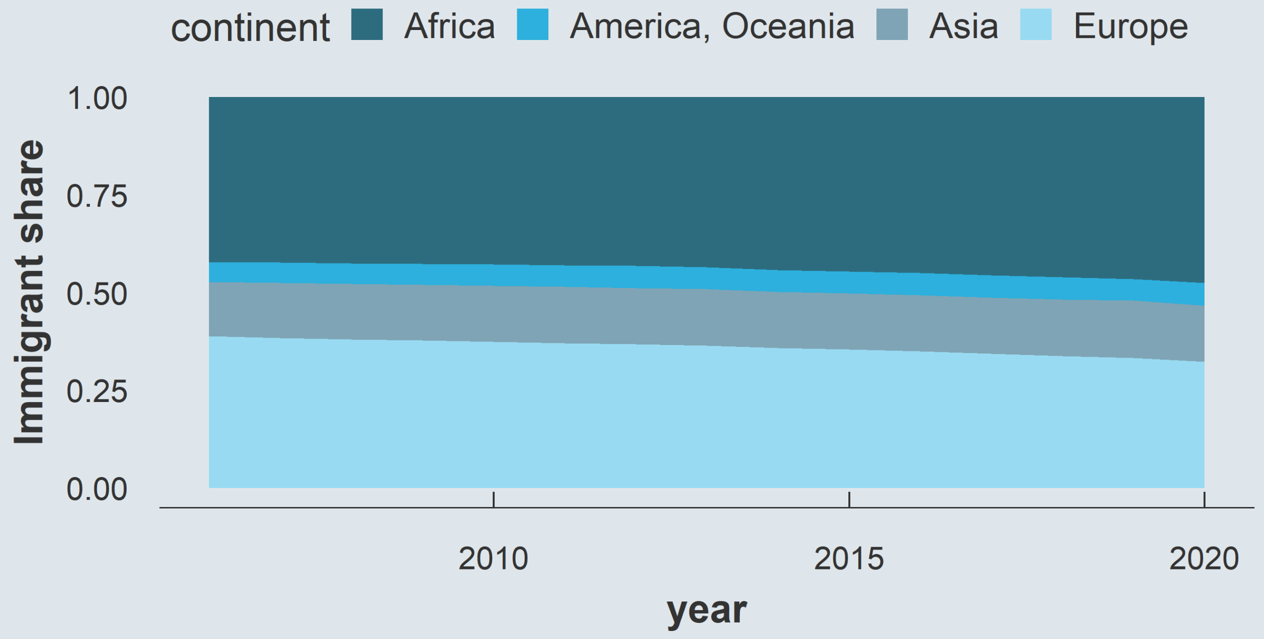


- We specify in aes that the fill color should be different according to the value of the variable `continent`
  - `position = "stack"` then puts bars of the grouping variable on top of others, here continent, for each category of the x variable

2. Bars and lines

2.3. geom_area() for evolutions of discrete distributions

```
ggplot(stack_immig %>% unique(), # Make sure that there's no duplicate
       aes(x = year, y = `Immigrant share`, fill = continent)) +
  geom_area(alpha = .8) + theme_minimal(base_size = 16)
```

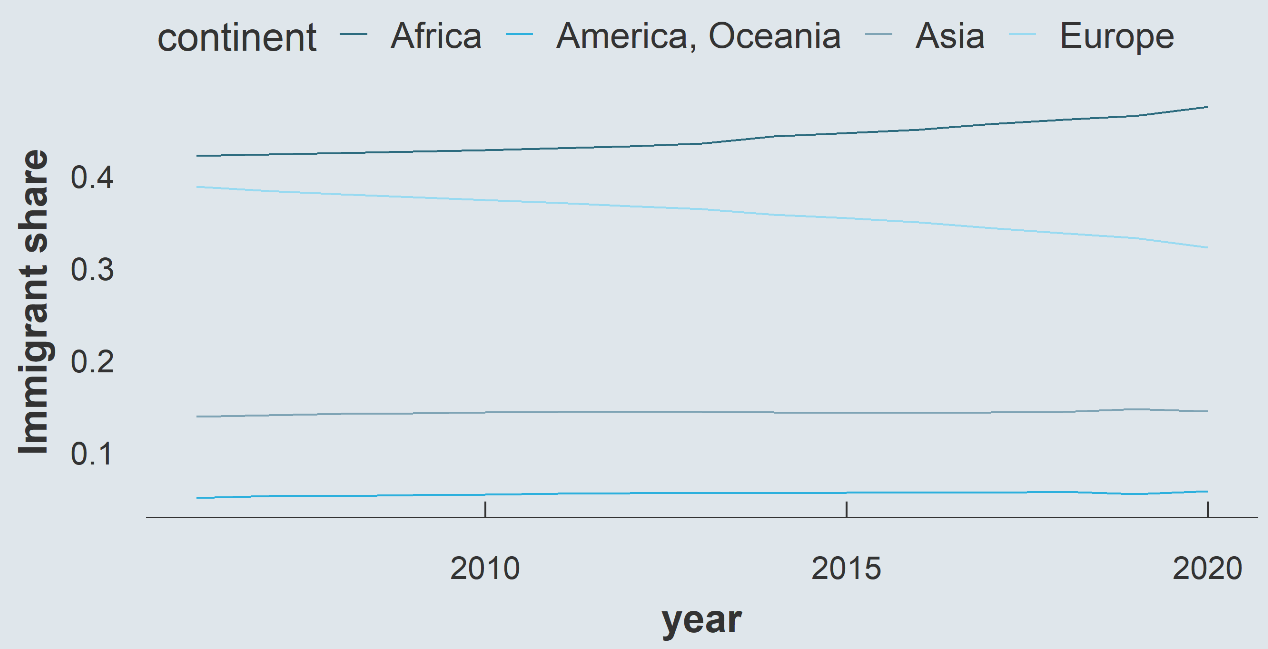


- Instead of stacking bar you can also stack areas with `geom_area()`
    - Do it only if your time variable is "continuous enough", otherwise the plot could be misleading

2. Bars and lines

2.3. geom_line() for evolution of a single variable

```r
ggplot(stack_immig,
       aes(x = year, y = `Immigrant share`, color = continent)) +
  geom_line(alpha = .8) + theme_minimal(base_size = 16)
```

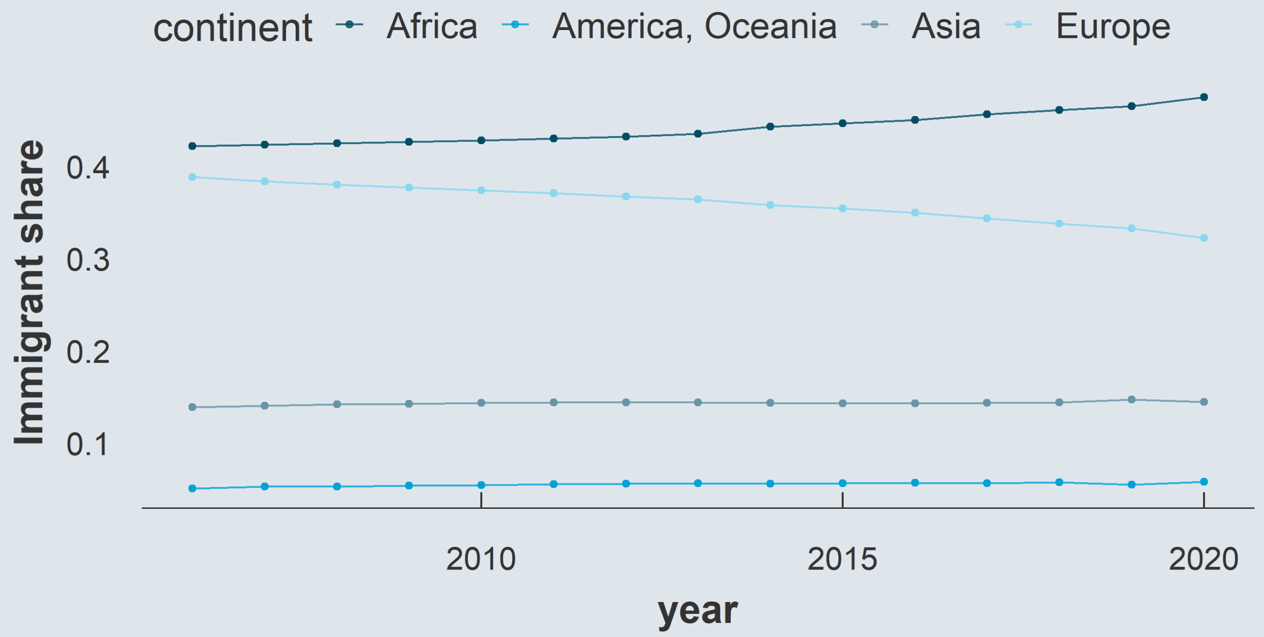


- But with stacked bars/areas it is sometimes difficult to see whether categories in the middle increase or decrease, and to get a sense of their proportion (see Asia in the previous plot)
    - Most of the time simply using lines with `geom_lines()` is better option

2. Bars and lines

2.3. geom_line() + geom_point() for evolution of a single variable

```
ggplot(stack_immig,
       aes(x = year, y = `Immigrant share`, color = continent)) +
  geom_line(alpha = .8) + geom_point(alpha = .8)  + theme_minimal(base_size = 16)
```



- Adding a point geometry is also usually preferred as it makes more salient what is actually interpolated between the point observations
    - The dedicated geometry is `geom_point()`

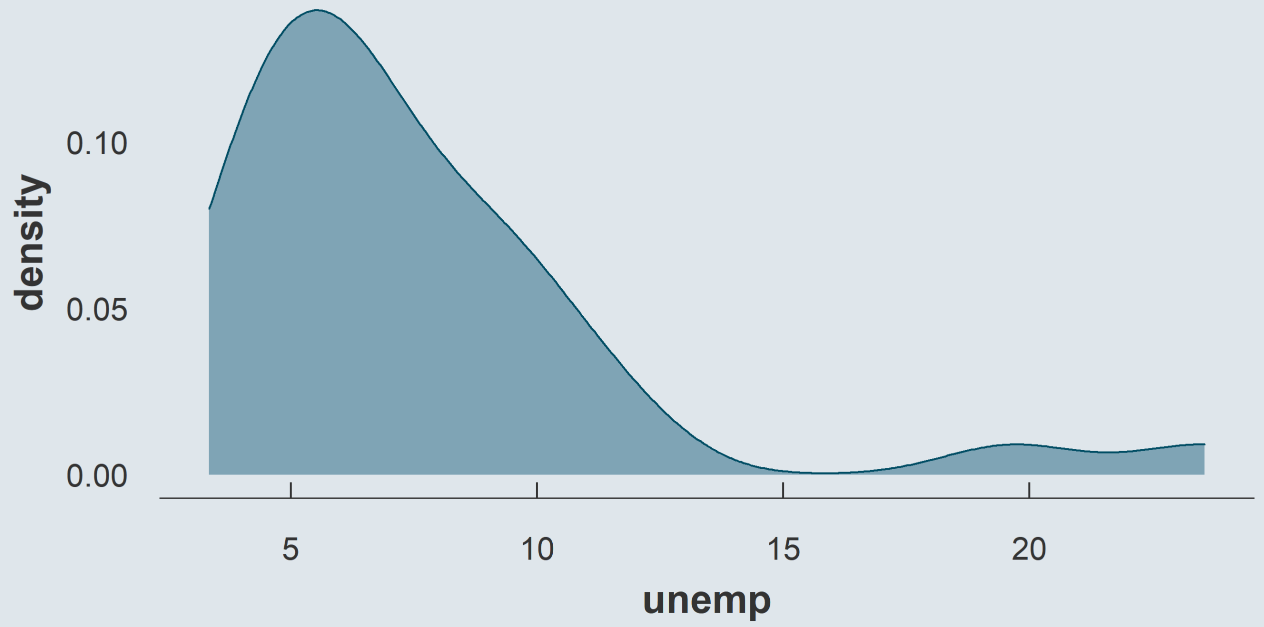
Overview

Overview

3. Densities and boxplots

3.1. Density

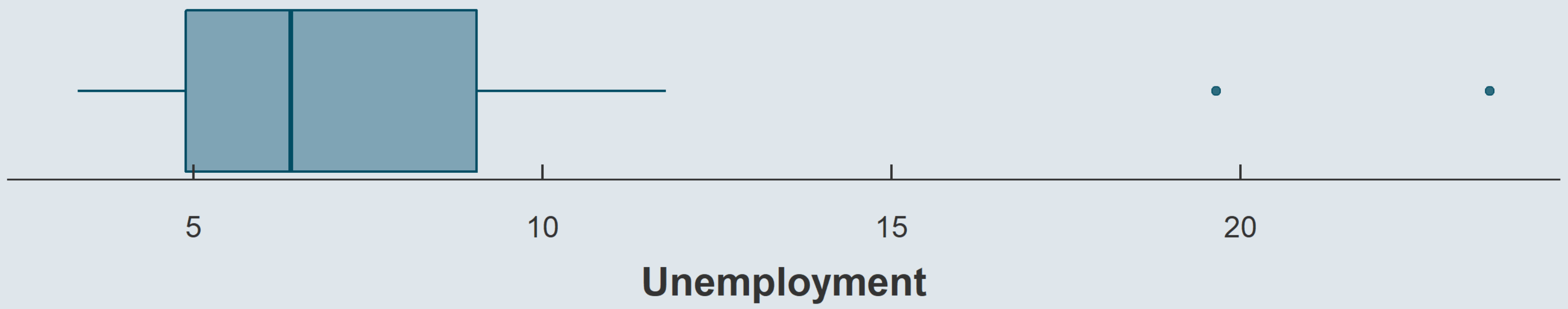
```
ggplot(oecd_data %>% filter(!is.na(unemp)), aes(x = unemp)) +
  geom_density(fill = "#6794A7", color = "#014D64", alpha = .8) +
  theme_minimal(base_size = 16)
```



- The dedicated geometry is `geom_density()`
  - You just have to specify the x variable you want the density of in aes
  - The smoothness of the density can be changed with the argument `bw=`: the larger the smoother

3. Densities and boxplots

3.2. Boxplot



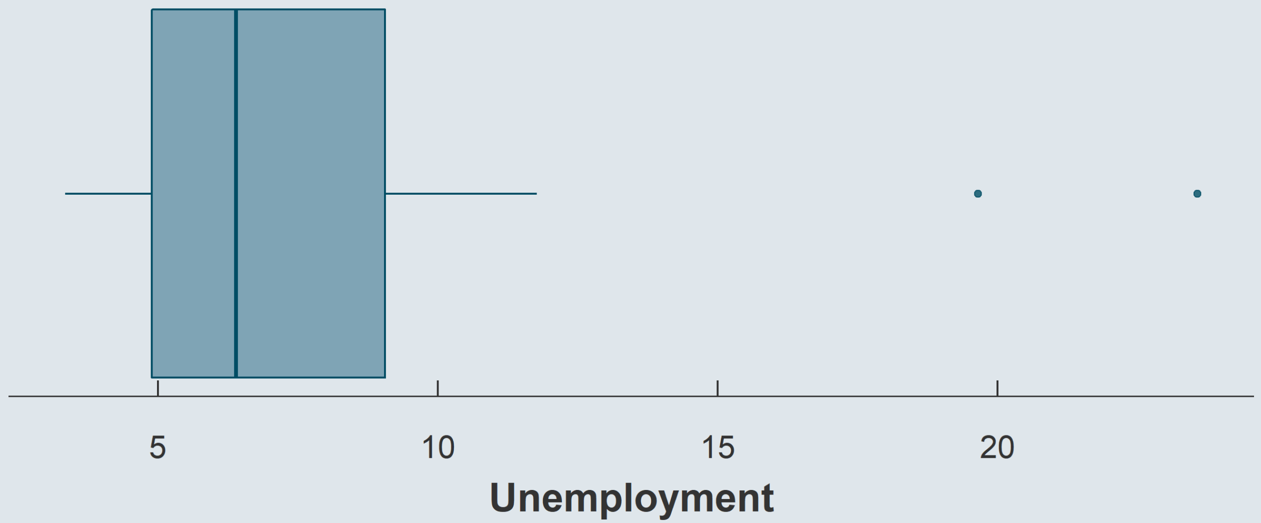
→ *A distribution can also be described with a boxplot:*

- The thick line inside the box is the median
- The bounds of the box are the first and third quartiles
- The lines connect the box to
  - The minimum/maximum value if these are less than 1.5 $\times$ IQR away from their closest quartile
  - 1.5 $\times$ IQR away from the corresponding quartile if some values stand outside this range.
- Dots represent values located more than 1.5 $\times$ IQR away from the closest quartile

3. Densities and boxplots

3.2. Boxplot

```
ggplot(oecd_data %>% filter(!is.na(unemp)), aes(x = unemp)) +
  geom_boxplot(fill = "#6794A7", color = "#014D64", alpha = .8) +
  theme_minimal(base_size = 16) + xlab("Unemployment") +
  theme(axis.text.y = element_blank())
```

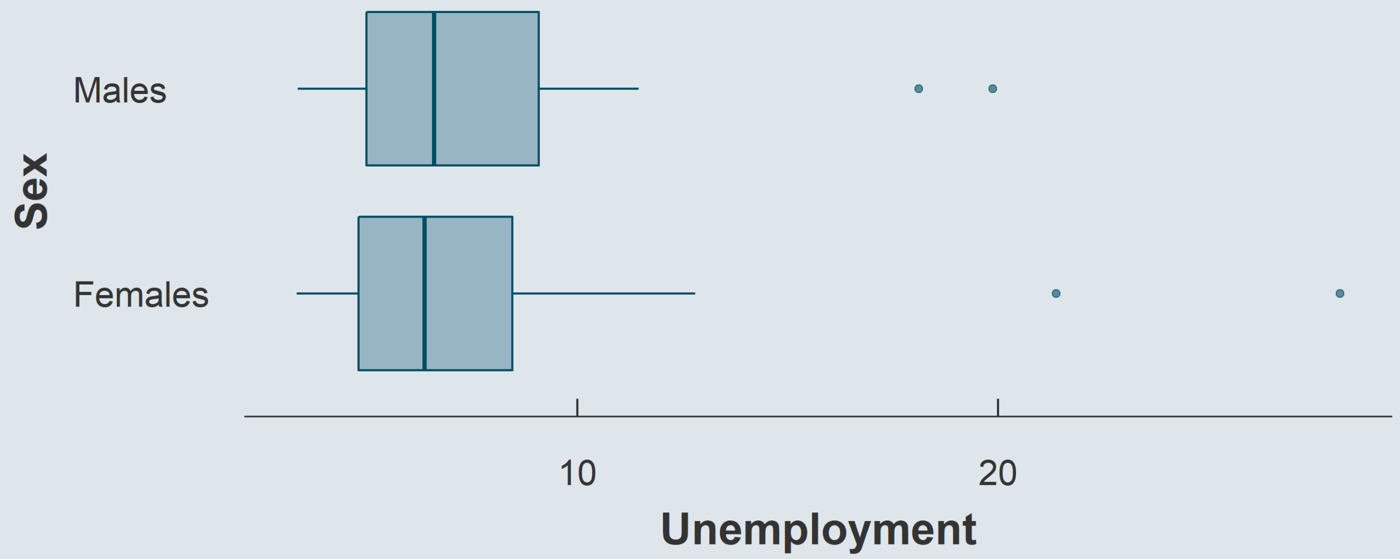


- The dedicated geometry is `geom_boxplot()`
    - Most elements of this geometry can be customized, see the help file `?geom_boxplot` for more details

3. Densities and boxplots

3.2. Boxplots by group

```
ggplot(unemp_long, aes(x = Sex, y = Unemployment)) +
  geom_boxplot(fill = "#6794A7", color = "#014D64", alpha = .6) +
  theme_minimal(base_size = 16) + coord_flip()
```

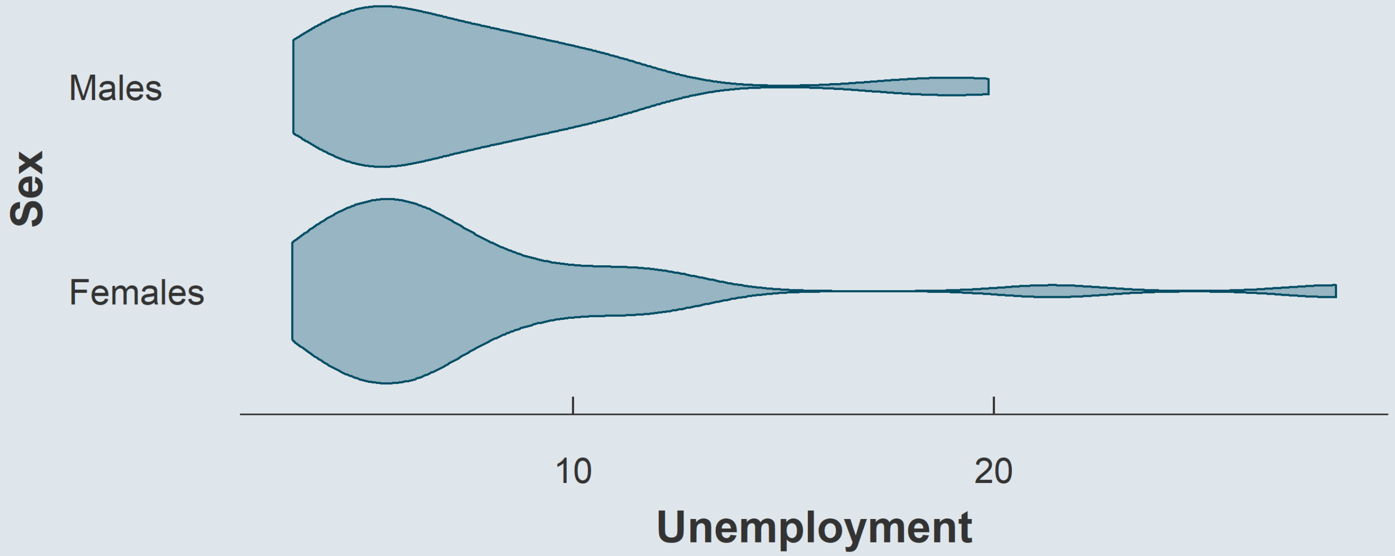


- To have separate boxplots for different groups, you can specify the grouping variable in x and the variable whose density should be display in y in aes
    - For the two boxplots to have different colors, the sex variable should be assign to fill in aes

3. Densities and boxplots

3.3. Violin

```
ggplot(unemp_long, aes(x = Sex, y = Unemployment)) +
  geom_violin(fill = "#6794A7", color = "#014D64", alpha = .6) +
  theme_minimal(base_size = 16) + coord_flip()
```

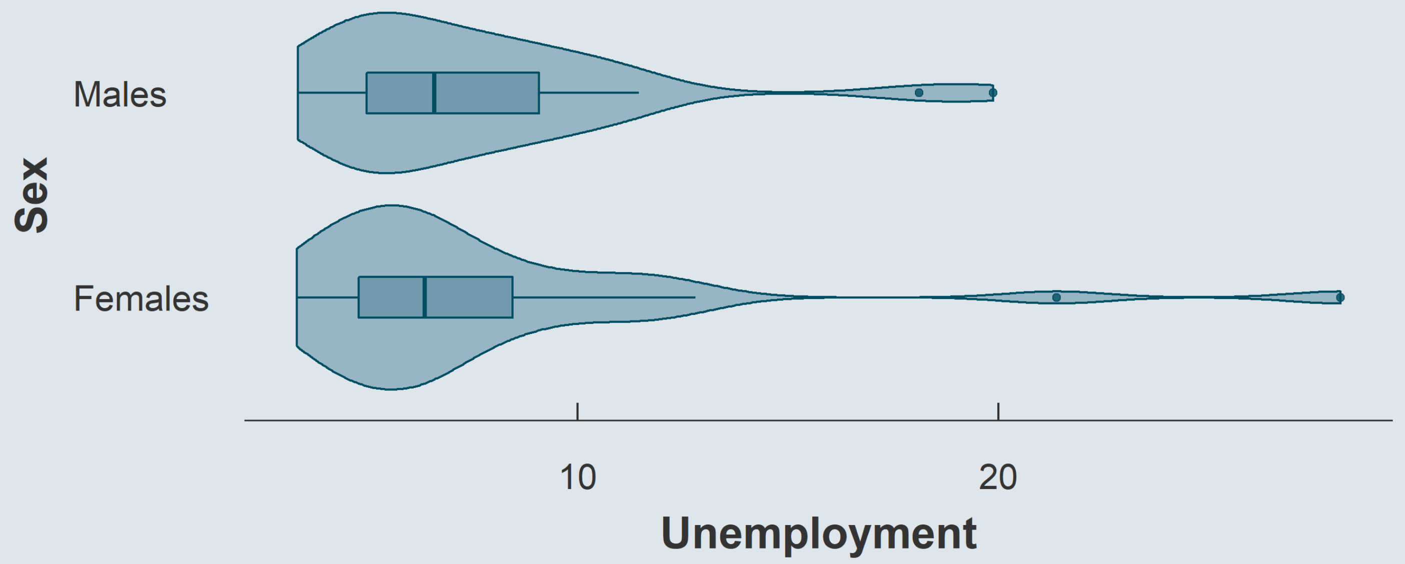


- Violins are a way to compared smooth density distributions for different groups.
  - You simply have to replace `geom_boxplot()` by `geom_violin()`

3. Densities and boxplots

3.3. Violin + boxplot

```r
ggplot(unemp_long, aes(x = Sex, y = Unemployment)) +
  geom_violin(fill = "#6794A7", color = "#014D64", alpha = .6) +
  geom_boxplot(fill = "#6794A7", color = "#014D64", alpha = .8, width = .2) +
  theme_minimal(base_size = 16) + coord_flip()
```



- Violins and boxplots can be combined to provide a more comprehensive depiction of the distribution of a continuous variable
  - Adjust the `width` argument of `geom_boxplot()` to make it fit in the violin

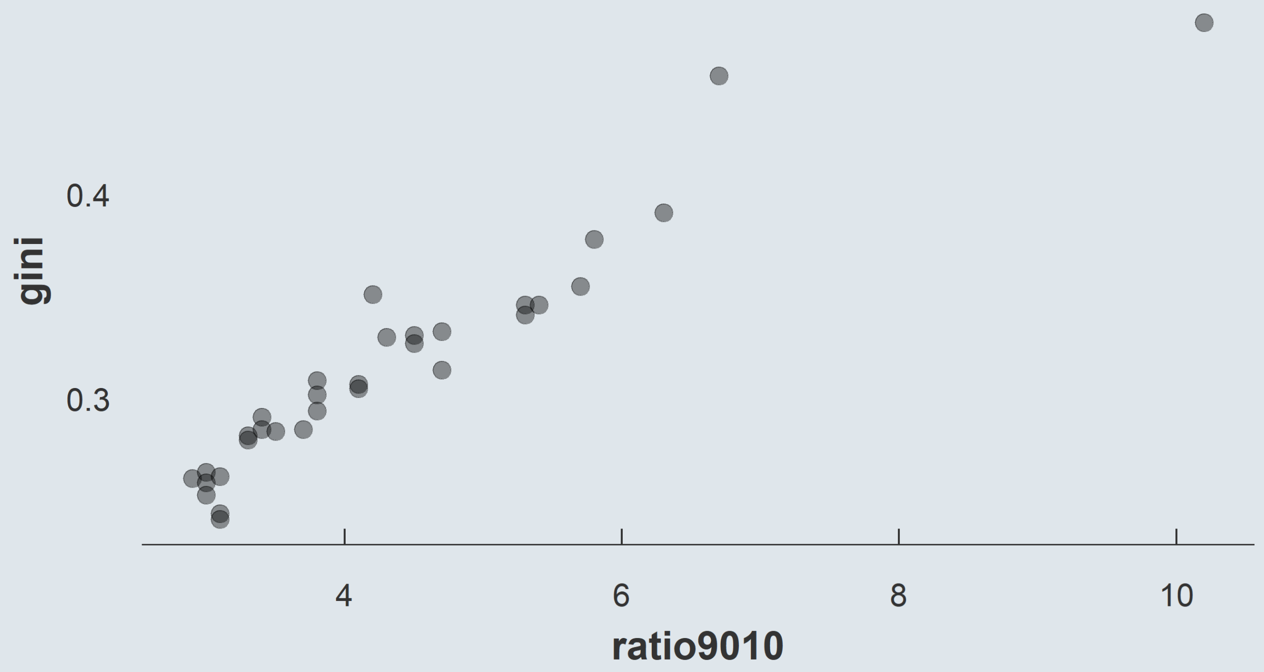
Overview

Overview

4. Scatter plots

Basic scatter plot with geom_point()

```
ggplot(oecd_data, aes(x = ratio9010, y = gini)) +
  geom_point(alpha = .4, size = 4) + theme_minimal(base_size = 16)
```

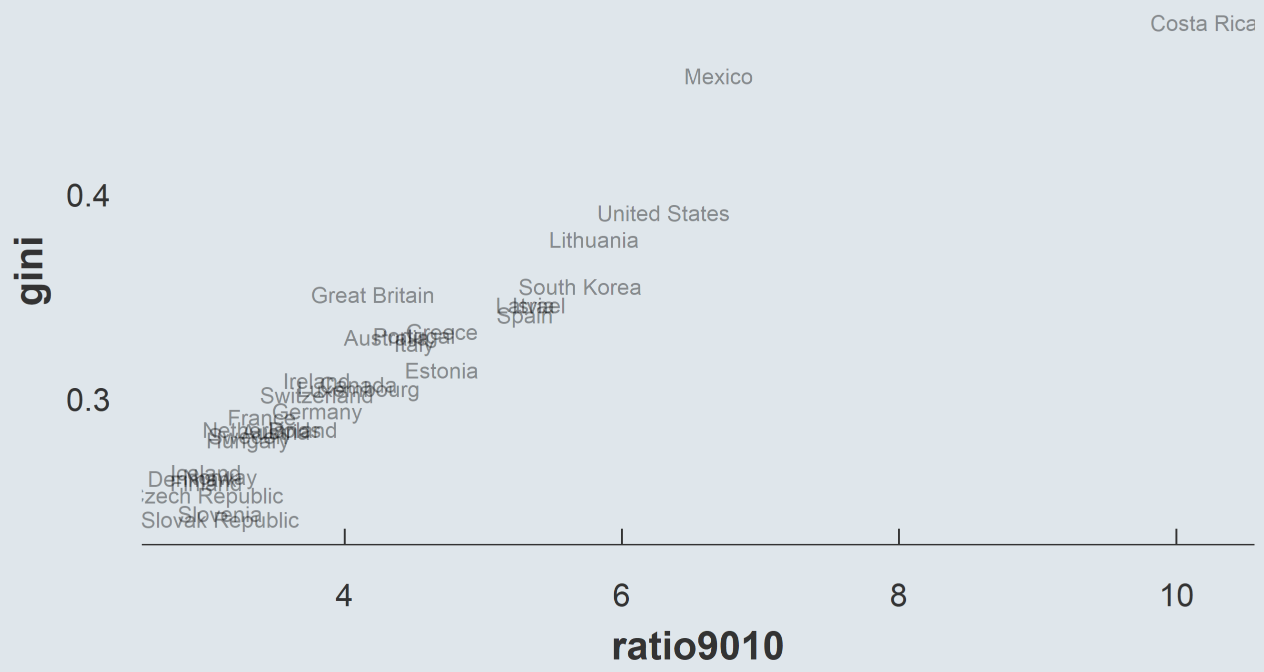


- A scatter plot places a point for every observation at the coordinates given by its x and y values. It is often use to visualize the relationship between two variables
  - The dedicated geometry is `geom_point()`

4. Scatter plots

Label scatter plot geom_text()

```
ggplot(oecd_data, aes(x = ratio9010, y = gini, label = country)) +
  geom_text(alpha = .4) + theme_minimal(base_size = 16)
```

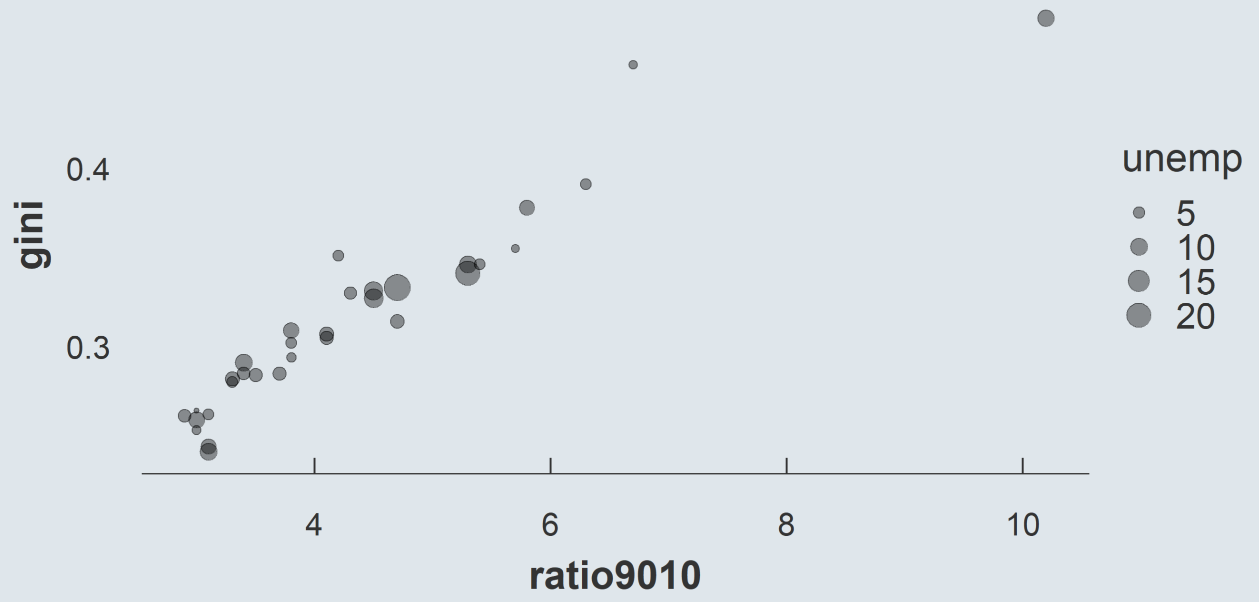


- When observations have labels, these can be used instead of points. By default the label centered at the x and y coordinates.
    - The dedicated geometry is `geom_text()` and the labelling variable should be assigned to the `label` argument in `aes()`

4. Scatter plots

Adding a third dimension with size

```
ggplot(oecd_data, aes(x = ratio9010, y = gini, size = unemp)) +
  geom_point(alpha = .4) + theme_minimal(base_size = 16) +
  theme(legend.position = "right", legend.direction = "vertical")
```

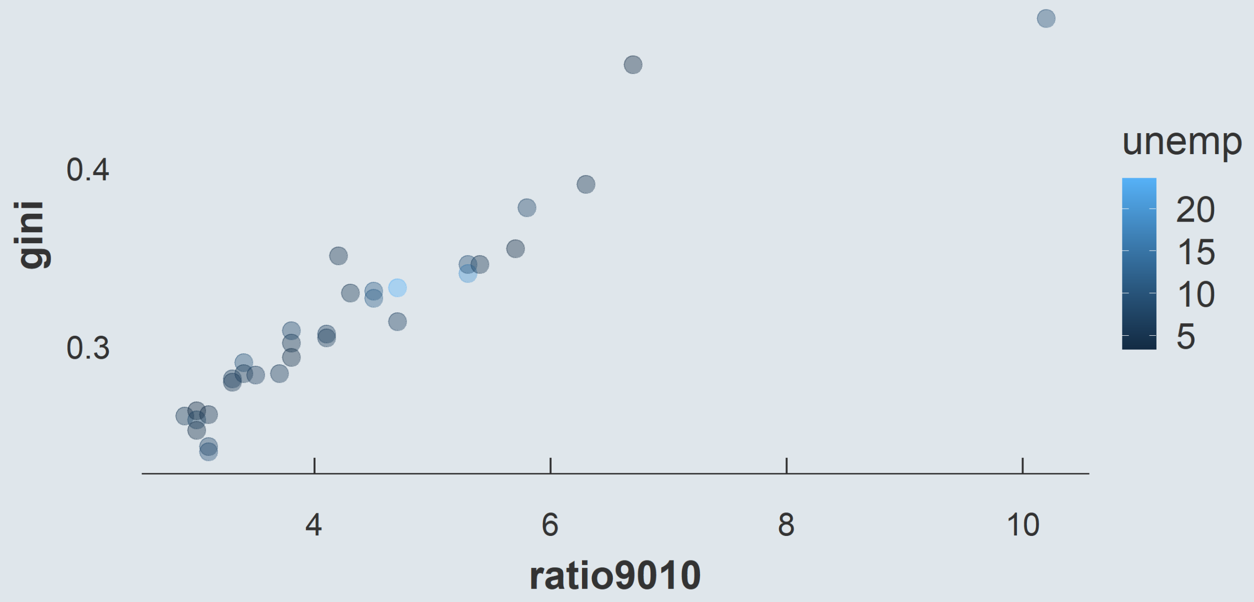


- On a scatter plot, the value of a third variable can be represented using the size of the geometry.
    - The third variable should be assigned to the `size` argument in `aes()`

4. Scatter plots

Adding a third dimension with color

```
ggplot(oecd_data, aes(x = ratio9010, y = gini, color = unemp)) +
 geom_point(alpha = .4, size = 4) + theme_minimal(base_size = 16) +
 theme(legend.position = "right", legend.direction = "vertical")
```

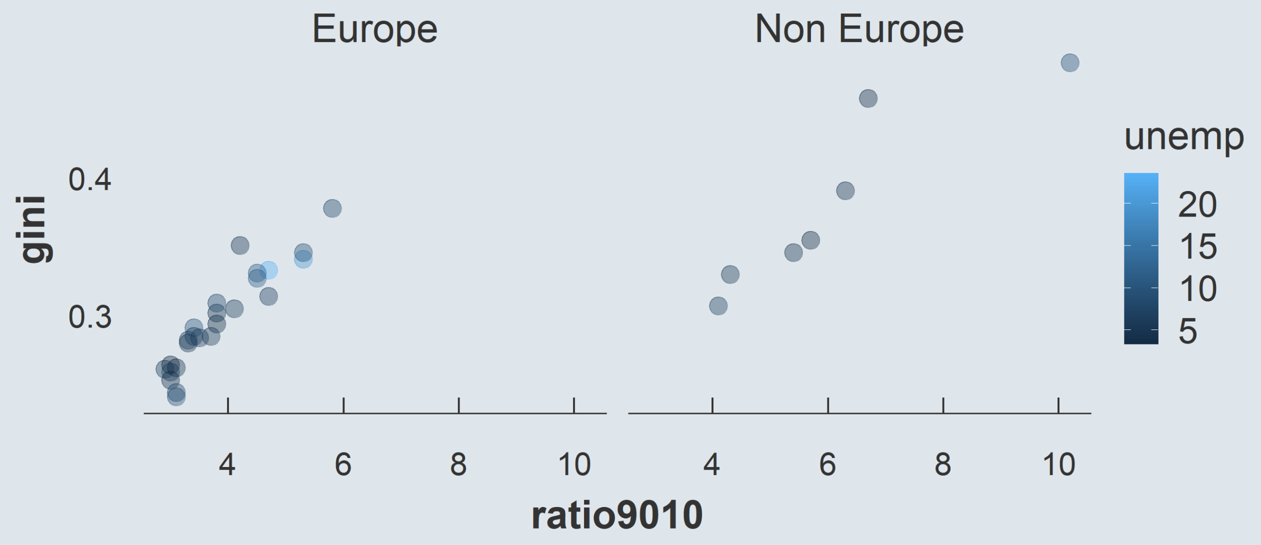


- The value of a third variable can be represented using the color of the geometry.
  - In that case the third variable should be assigned to the `color` argument in `aes()`

4. Scatter plots

Adding a third dimension with facets

```
ggplot(oecd_data %>% mutate(europe = ifelse(continent == "Europe", "Europe", "Non Europe")),
       aes(x = ratio9010, y = gini, color = unemp)) +
  geom_point(alpha = .4, size = 4) + theme_minimal(base_size = 16)  +
  theme(legend.position = "right", legend.direction = "vertical") +
  facet_wrap(~europe)
```

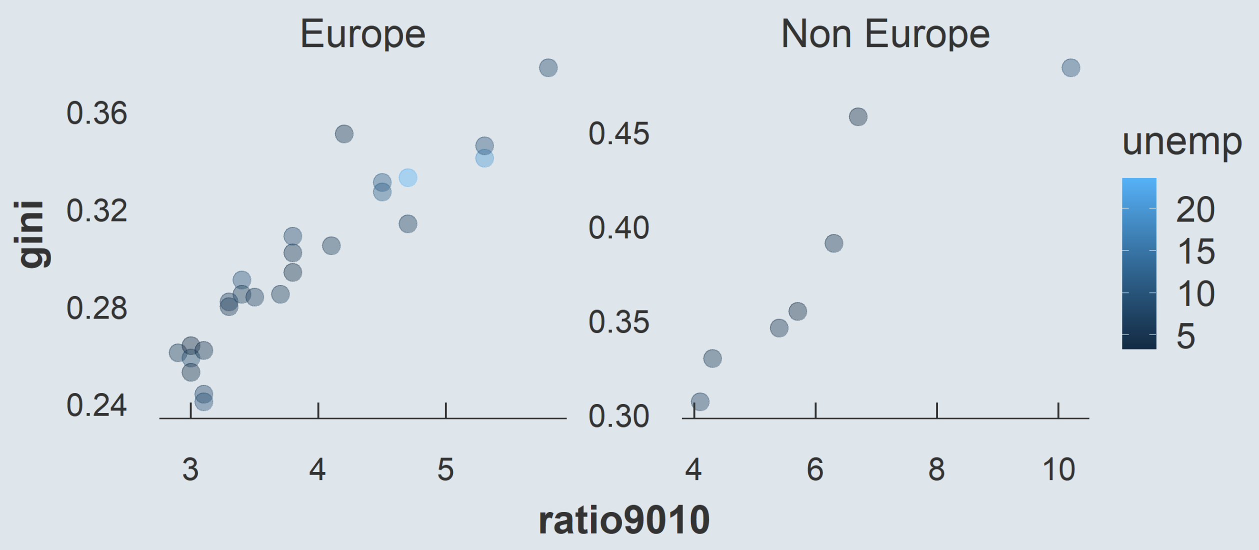


- So far we increased dimensionality with unemployment, a continuous variable. When the third variable is discrete, facets can be used
    - The dedicated function is `facet_wrap()`. The faceting variable should be indicated after a `~`

4. Scatter plots

Adding a third dimension with facets

```
ggplot(oecd_data %>% mutate(europe = ifelse(continent == "Europe", "Europe", "Non Europe")),
       aes(x = ratio9010, y = gini, color = unemp)) +
  geom_point(alpha = .4, size = 4) + theme_minimal(base_size = 16) +
  theme(legend.position = "right", legend.direction = "vertical") +
  facet_wrap(~europe, scales = "free")
```

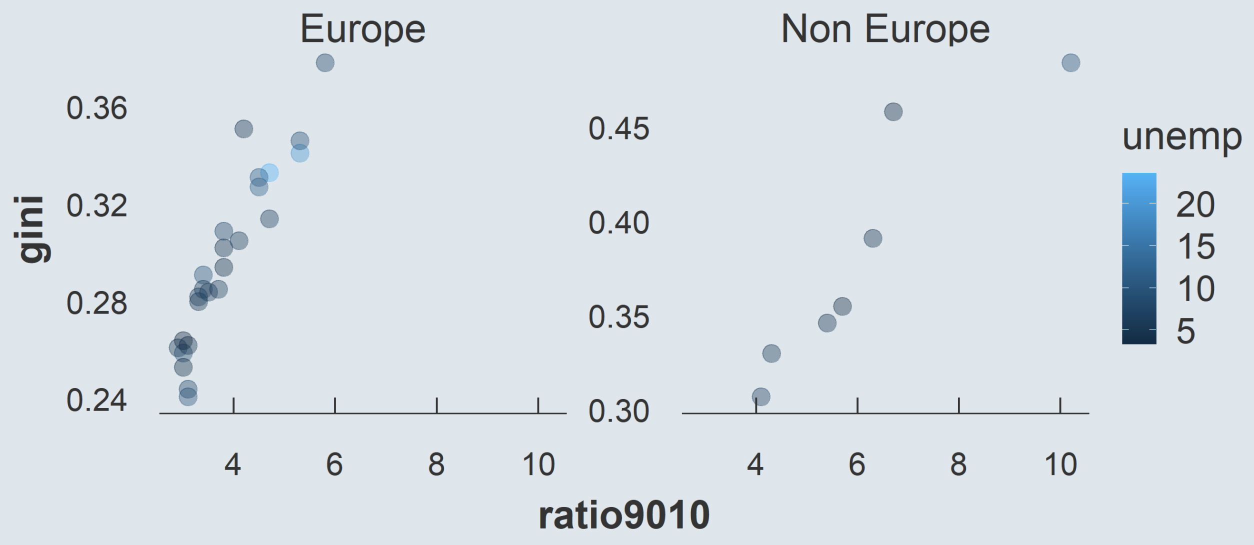


- By default facet axes are fixed, i.e., they are the same for both facets. But in some cases in can be useful to "free" the axes, i.e., to let the axes be facet-specific.
    - This can be done by setting the `scale` argument of `facet_wrap()` to `"free"`

4. Scatter plots

Adding a third dimension with facets

```
ggplot(oecd_data %>% mutate(europe = ifelse(continent == "Europe", "Europe", "Non Europe")),
       aes(x = ratio9010, y = gini, color = unemp)) +
  geom_point(alpha = .4, size = 4) + theme_minimal(base_size = 16)  +
  theme(legend.position = "right", legend.direction = "vertical") +
  facet_wrap(~europe, scales = "free_y")
```



- It is also possible to free a single axis only
    - To free the x or the y axis, set `scale` argument of `facet_wrap()` to `"free_x"` or `"free_y"`

Overview

5. Wrap up!

The 3 core components of the ggplot() function

Component	Contribution	Implementation
Data	Underlying values	ggplot(data, | data %>% ggplot(.,
Mapping	Axis assignment	aes(x = V1, y = V2, ...))
Geometry	Type of plot	+ geom_point() + geom_line() + ...

- Any other element should be added with a `+` sign

```
ggplot(data, aes(x = V1, y = V2)) +
  geom_point() + geom_line() +
  anything_else()
```

5. Wrap up!

Main customization tools

Item to customize	**Main functions**
Axes	scale_[x/y]_[continuous/discrete]
Baseline theme	theme_[bw/void/minimal/.../dark]()
Annotations	geom_[[h/v]line/text](), annotate()
Theme	theme(axis.[line/ticks].[x/y] = ...,
	legend.[position/direction] = ...)

```r
ggplot(data, aes(x = V1, y = V2)) + geom_point() + geom_line() +
  scale_x_continuous(name = "Axis label", limits = c(0, 10), breaks = 0:10) +
  theme_minimal(base_size = 14) + geom_vline(xintercept = 5) +
  annotate("text", x = mean(V1), y = mean(V2), label = "Mean") +
  theme(axis.line.x = element_line(size = 2), axis.ticks = element_blank(),
        legend.position = "top", legend.direction = "horizontal")
```

5. Wrap up!

Main types of aesthetics

Argument	Meaning
alpha	opacity from 0 to 1
color	color of the geometry
fill	fill color of the geometry
size	size of the geometry
shape	shape for geometries like points
linetype	solid, dashed, dotted, etc.

- If specificed in the geometry it will apply uniformly to every all the geometry
- If assigned to a variable in aes it will vary with the variable according to a scale documented in legend

```r
ggplot(data, aes(x = V1, y = V2, size = V3)) +
  geom_point(color = "steelblue", alpha = .6)
```

5. Wrap up!

Main types of geometry

Geometry	Function
Bar plot	geom_bar()
Histogram	geom_histogram()
Area	geom_area()
Line	geom_line()
Density	geom_density()
Boxplot	geom_boxplot()
Violin	geom_violin()
Scatter plot	geom_point()