Lecture 2: Summarizing and Visualizing Data

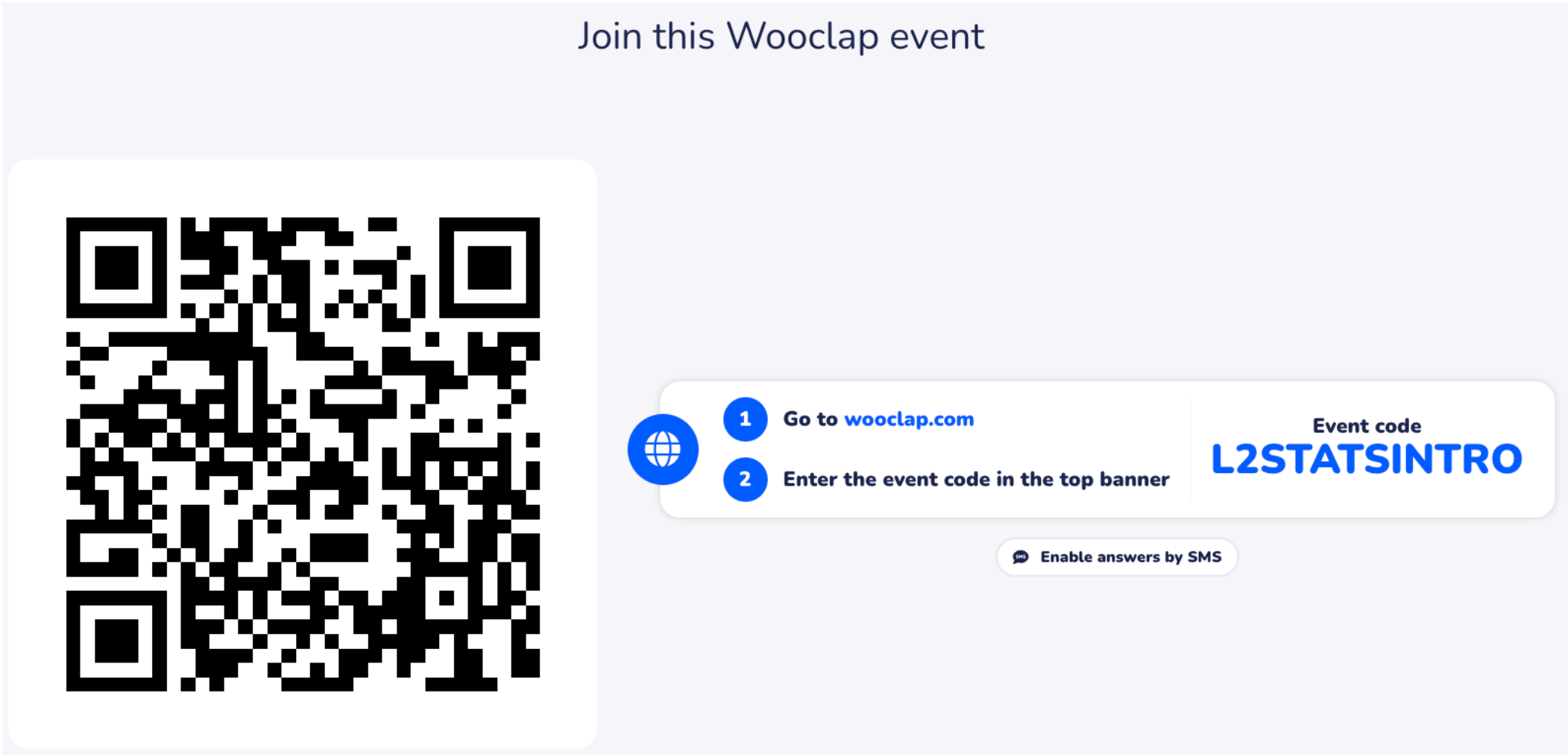
L2 - Statistics

Gustave Kenedi

This lecture is heavily inspired by that of Louis Sirugue.

2026-01-20

Recap quiz



Or click here: link to Wooclap

Today's lecture

1. Summarizing data
    - 1.1 Distributions
    - 1.2 Central tendency
    - 1.3 Spread
    - 1.4 Relationship between variables
2. Manipulating data
    - 2.1 `dplyr` verbs
    - 2.2 `group_by`
    - 2.3 Joining data
3. Visualizing data
    - 3.1 gg is for Grammar of Graphics
    - 3.2 Different types of gaphs

Summarizing data

Distributions

The point of descriptive statistics is to **summarize a big table** of values with a small set of **tractable statistics**

The most comprehensive way to characterize a variable/vector is to compute its **distribution**:

- **What** are the **values** the variable takes?
- **How frequently** does each value appear?

→ **Consider for instance the following variable:**

Variable 1

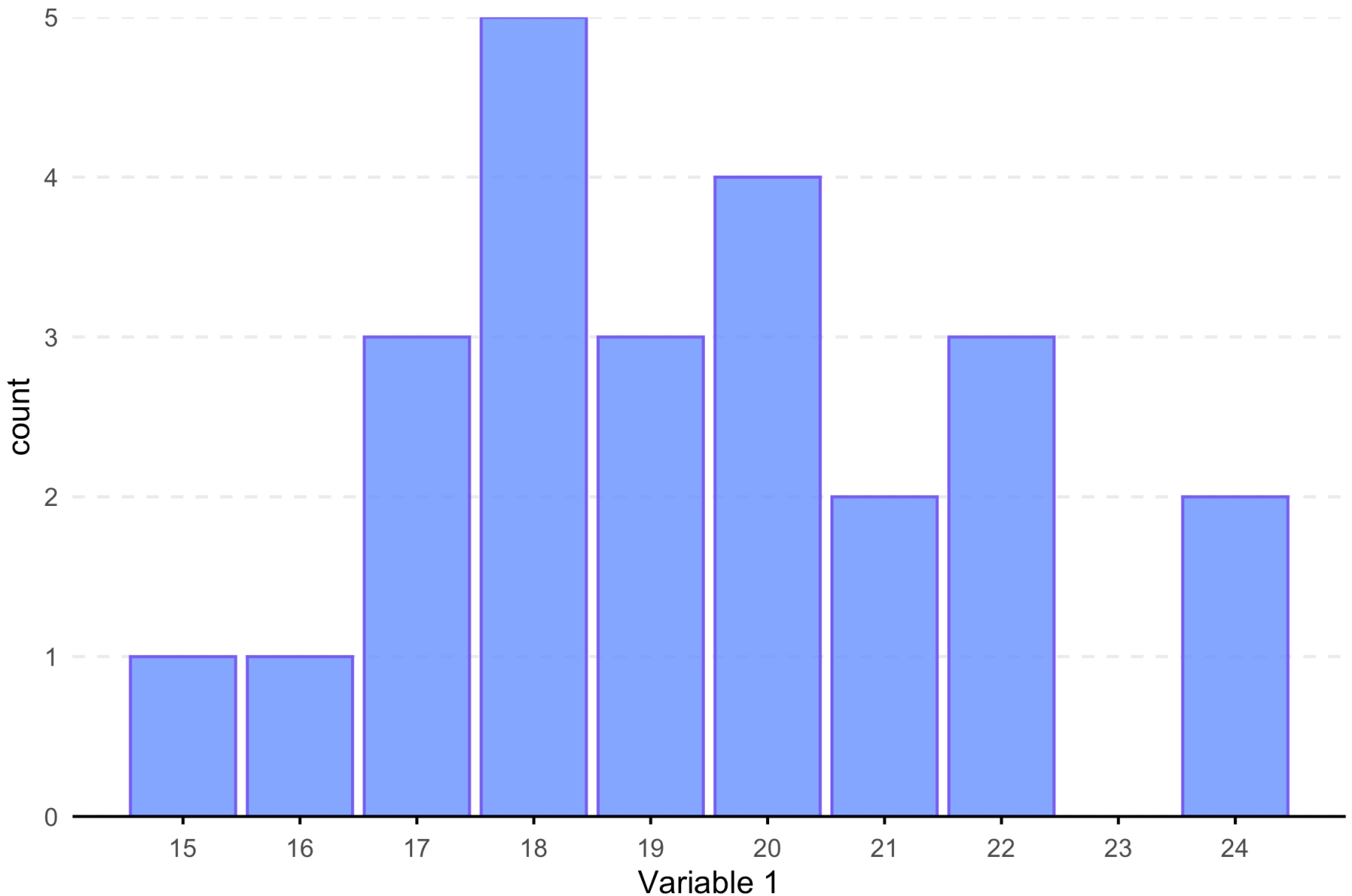
18	17	21	18	17	16	19	24	15	24	18	19
22	20	21	18	17	20	20	22	20	18	19	22

- How many times does each value appear?

Variable 1	15	16	17	18	19	20	21	22	24
n	1	1	3	5	3	4	2	3	2

- We can represent this distribution graphically with a bar plot
  - x-axis: possible values
  - y-axis: number of occurrences

Graphical representation of discrete distributions



Continuous distributions

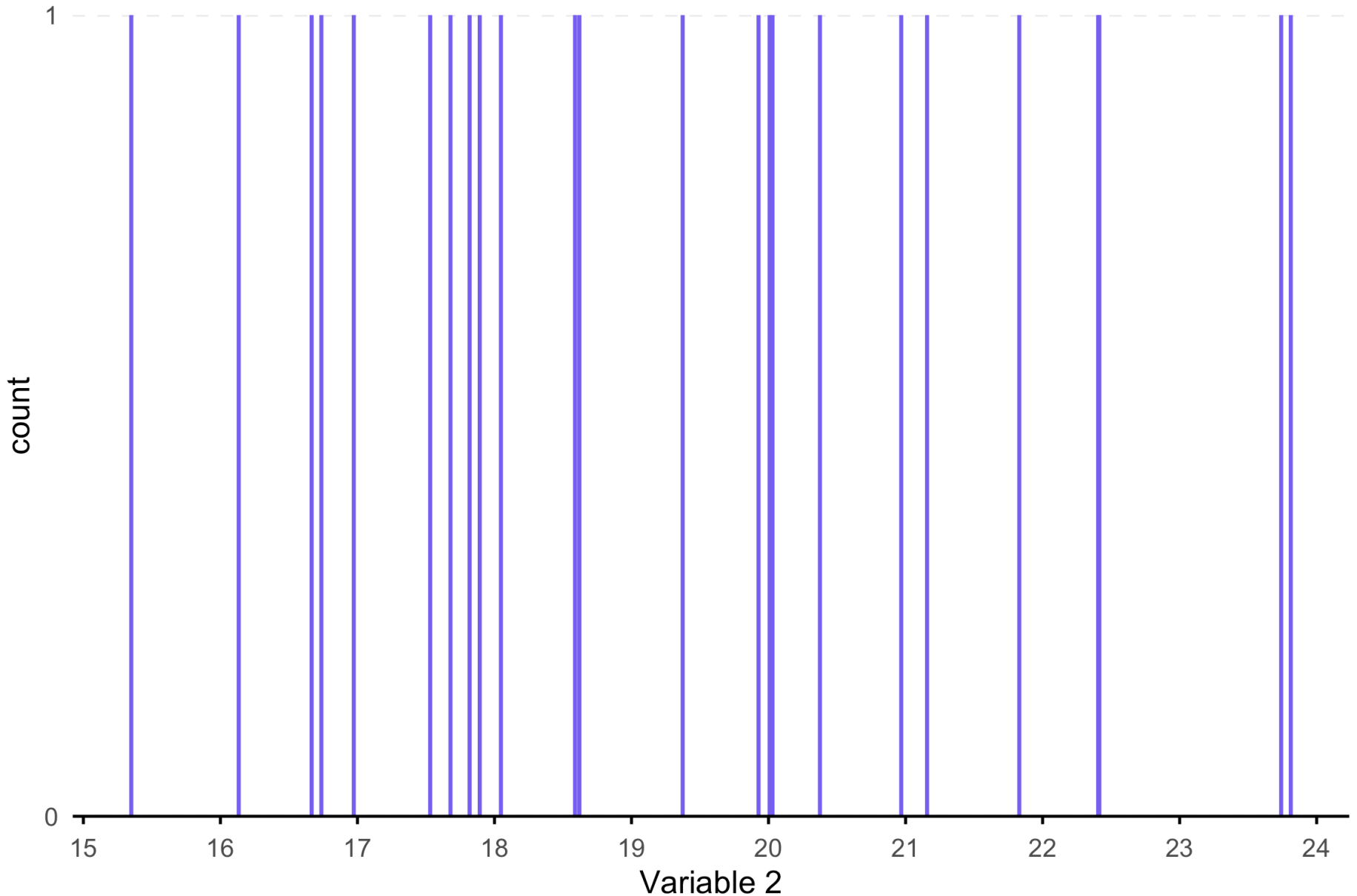
- But what if we would like to do the same thing for the following variable?

Variable 2					
17.53074	21.15684	16.66565	18.58788	15.34944	17.67870
22.40558	20.96827	16.97330	20.00811	19.92690	19.37334
16.73665	17.81842	16.13361	23.81101	23.74077	18.61935
20.37507	17.89235	20.02870	21.82973	18.04681	22.41368

- Each value **appears only once**
  - So the count of each variable does not help summarizing the variable

→***Let's have a look at the corresponding bar plot***

Graphical representation of continuous distributions



Continuous and discrete distributions

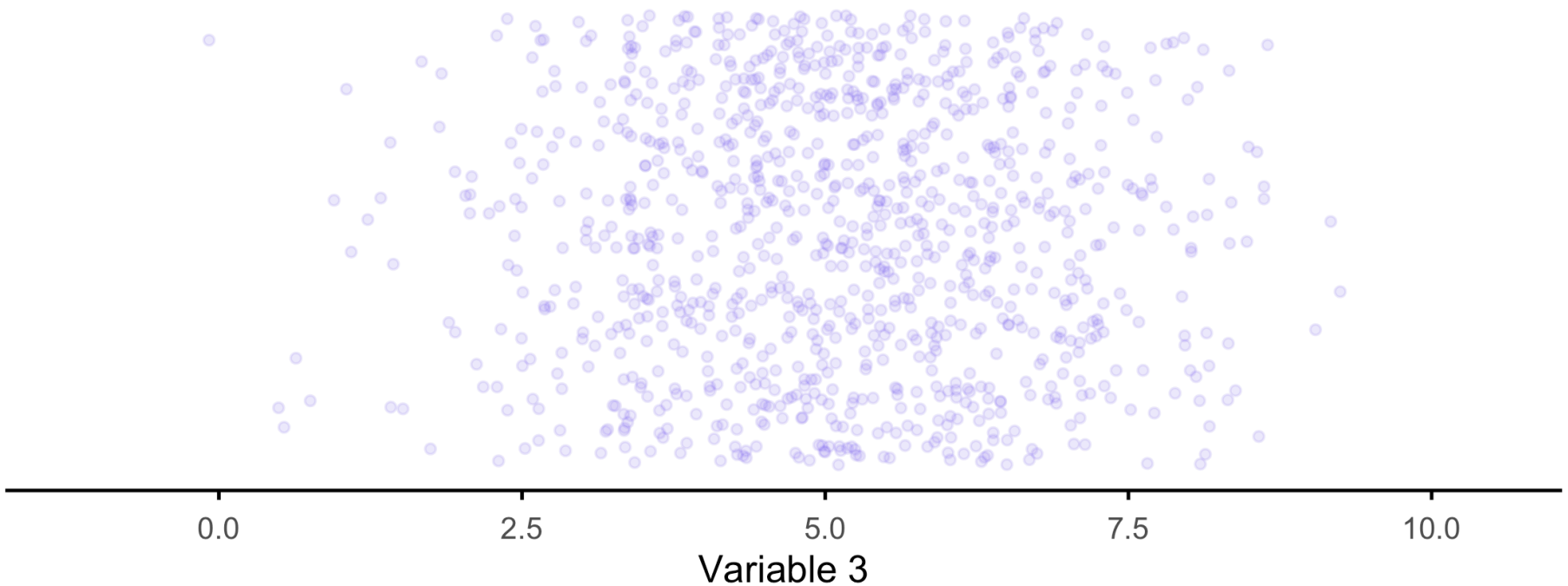
- It does not look good for this variable because it is **continuous**, while the first one was **discrete**
  - **Discrete variables:** variables that take a finite (or sufficiently small) number of values, e.g., number of siblings, eye color, …
  - **Continuous variables:** variables that take an infinite (or sufficiently large) number of values, e.g., annual income, height in centimeters, …

→ *In practice some variables are difficult to classify. E.g.,* ***age (in years)*** *can be viewed* ***as a discrete*** *variable because it takes a finite set of values, but this set is possibly quite wide, one could view it* ***as a continuous variable****. It often depends on the context.*

- To get a sense of the **distribution** of a **continuous variable**: plot a **histogram**
  - Group values into *bins* and plot number values that fall into each bin
  - The bar plots we've seen so far are basically histograms with number of bins = number of possible values

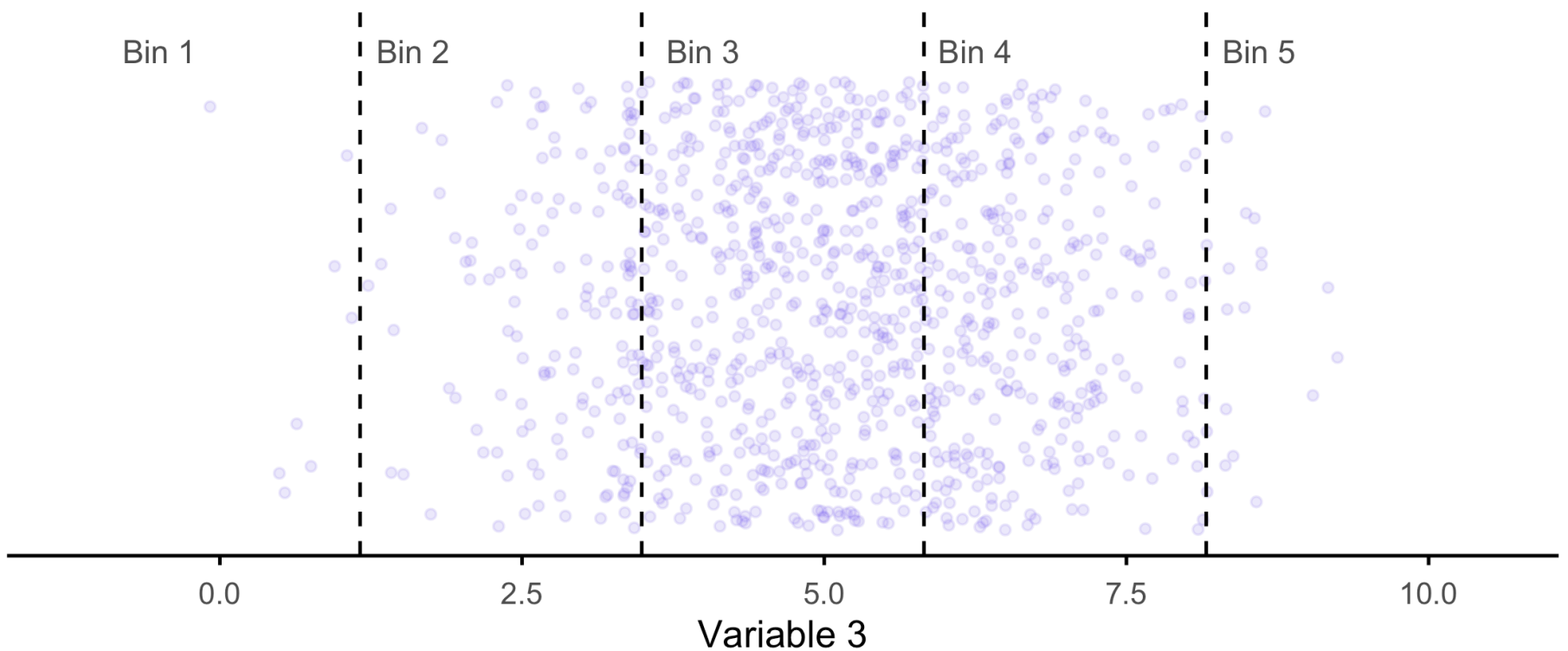
Graphical representation of continuous distributions

- Consider for instance the following variable. For clarity each point is shifted vertically by a random amount



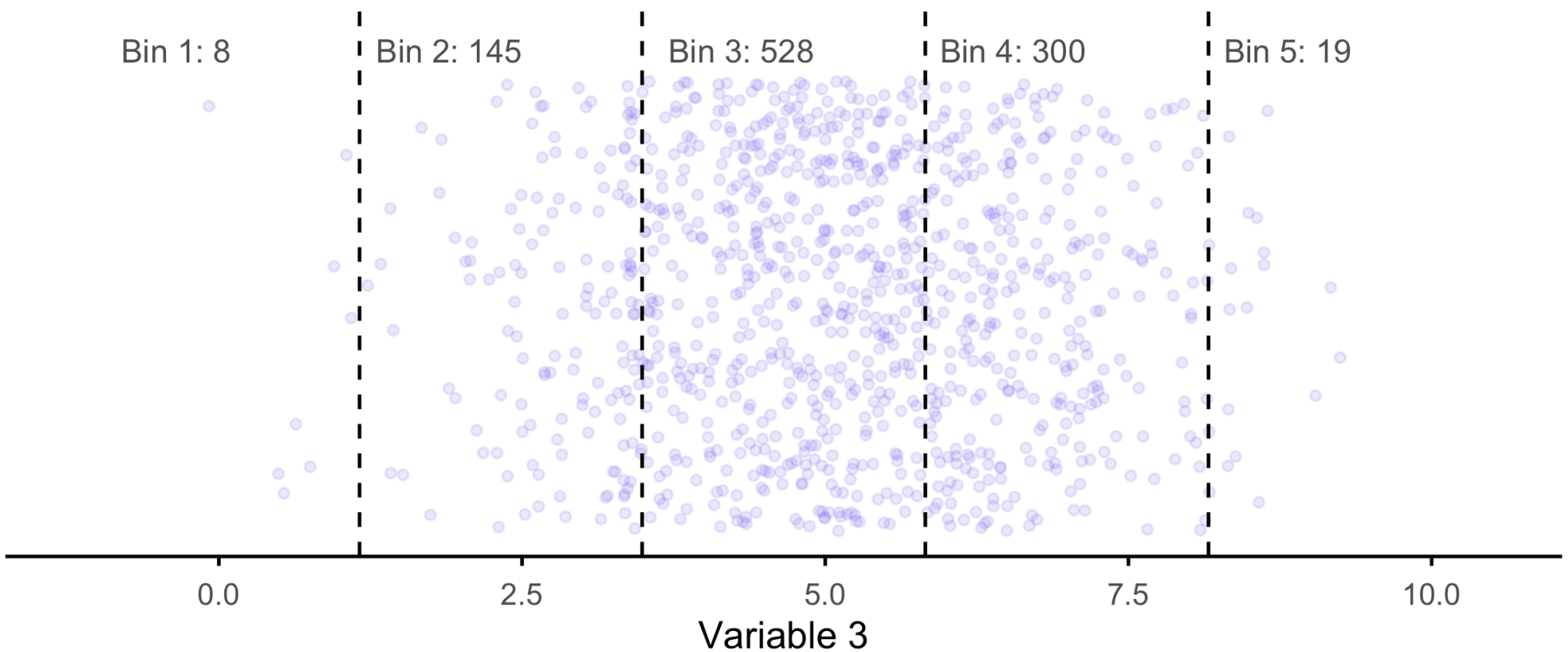
Graphical representation of continuous distributions

- Consider for instance the following variable. For clarity each point is shifted vertically by a random amount
- We can divide the domain of this variable into 5 bins



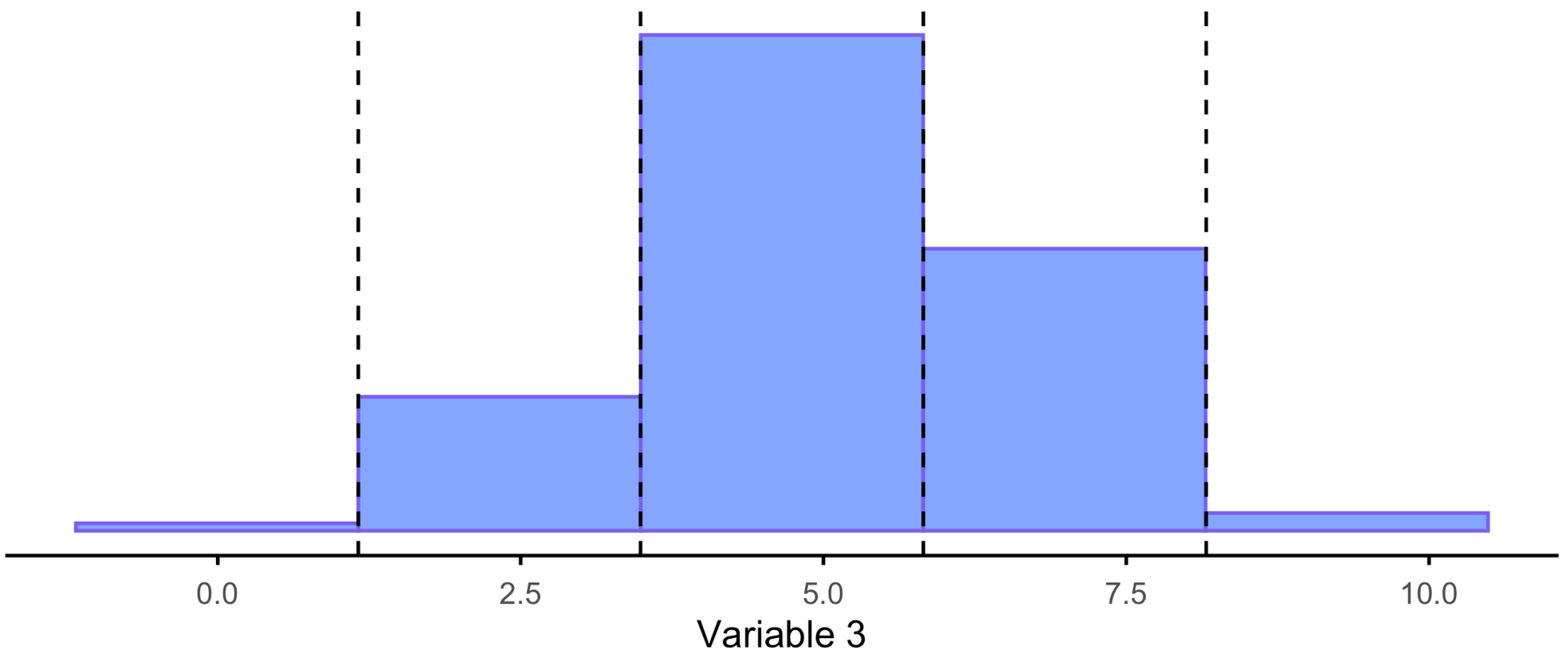
Graphical representation of continuous distributions

- Consider for instance the following variable. For clarity each point is shifted vertically by a random amount
- We can divide the domain of this variable into 5 bins
- And count the number of observations within each bin



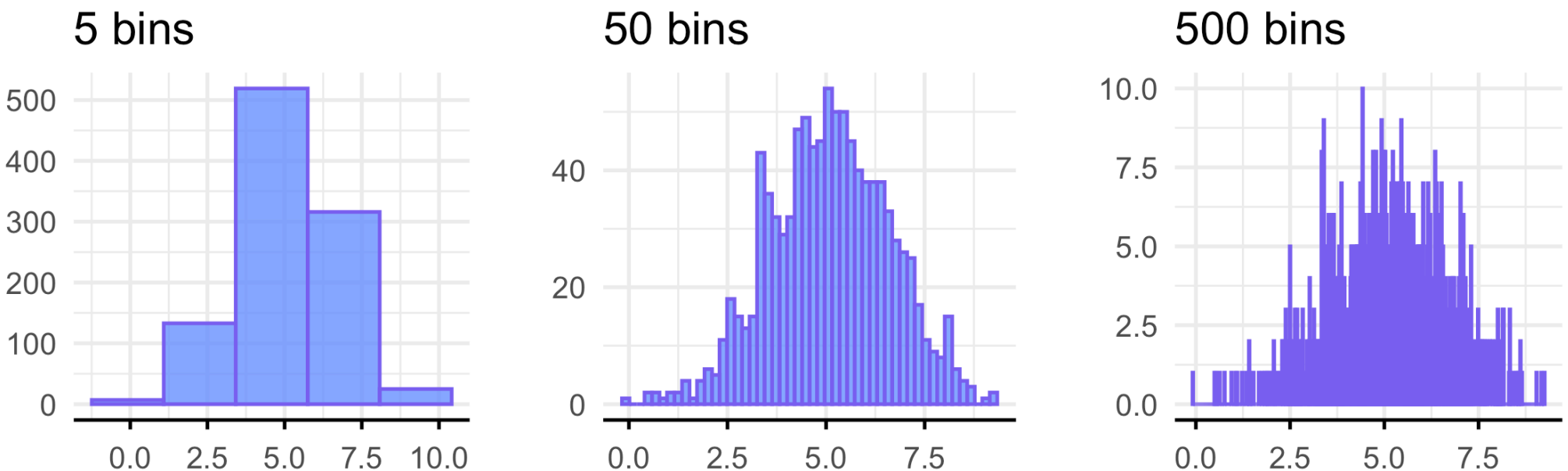
Graphical representation of continuous distributions

- Consider for instance the following variable. For clarity each point is shifted vertically by a random amount
- We can divide the domain of this variable into 5 bins
- And count the number of observations within each bin



Histograms

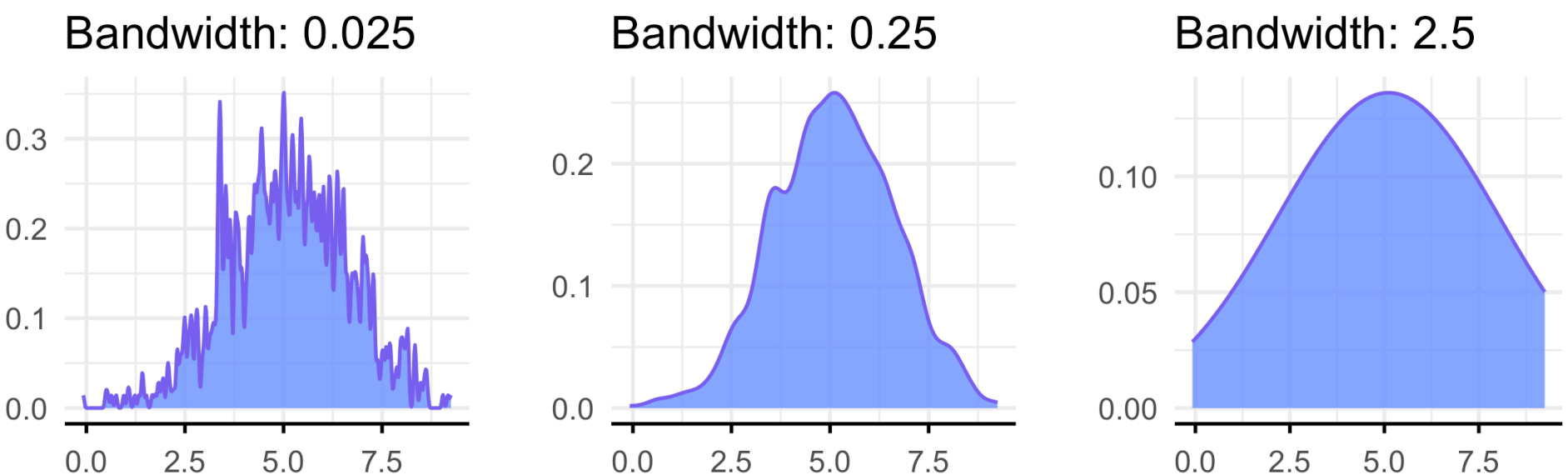
- There's no definitive rule to choose the number of bins
  - But too many or too few can yield misleading histograms



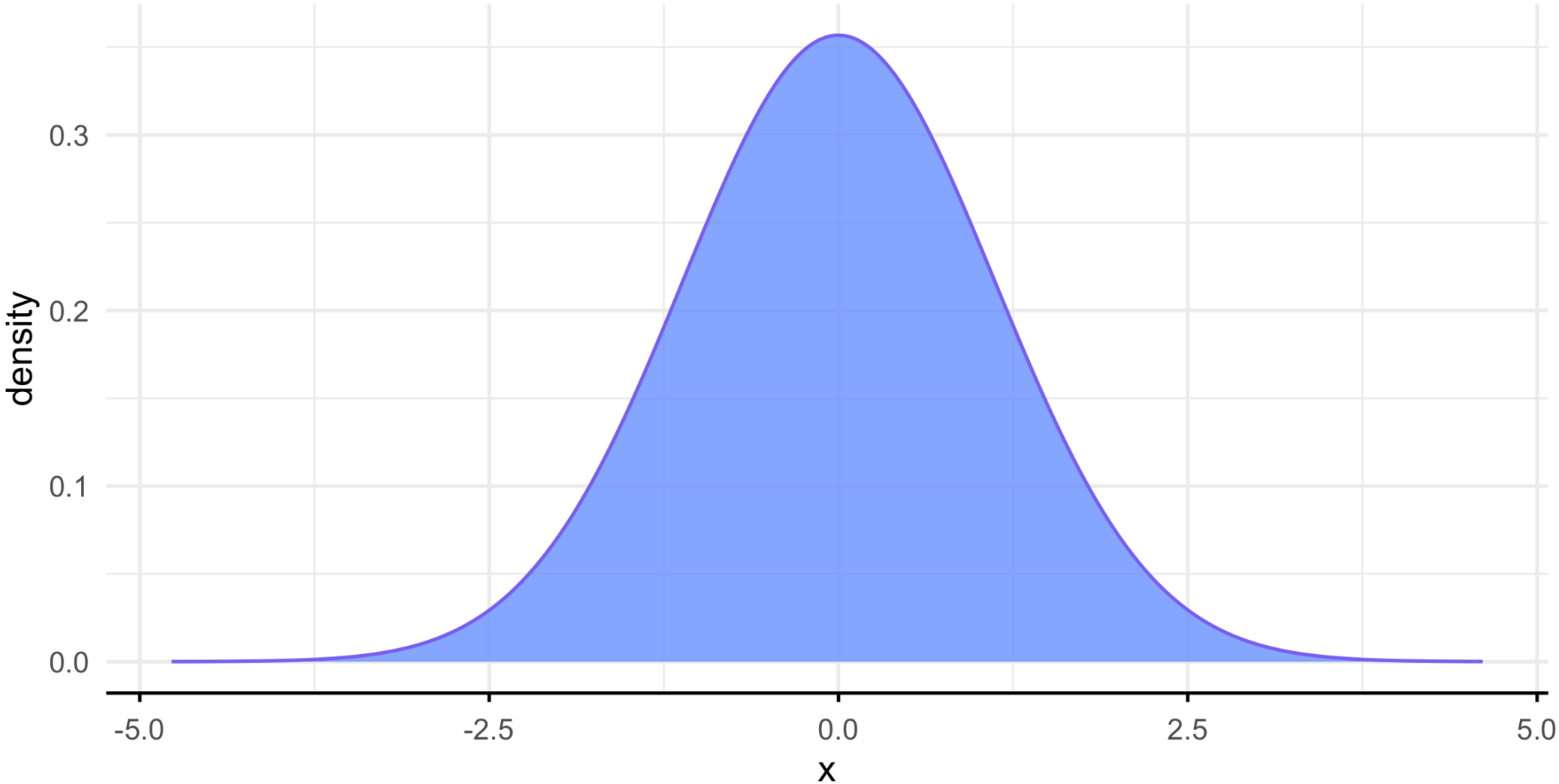
→ ***Note that choosing the number of bins is equivalent to choosing the width of each bin***

Densities

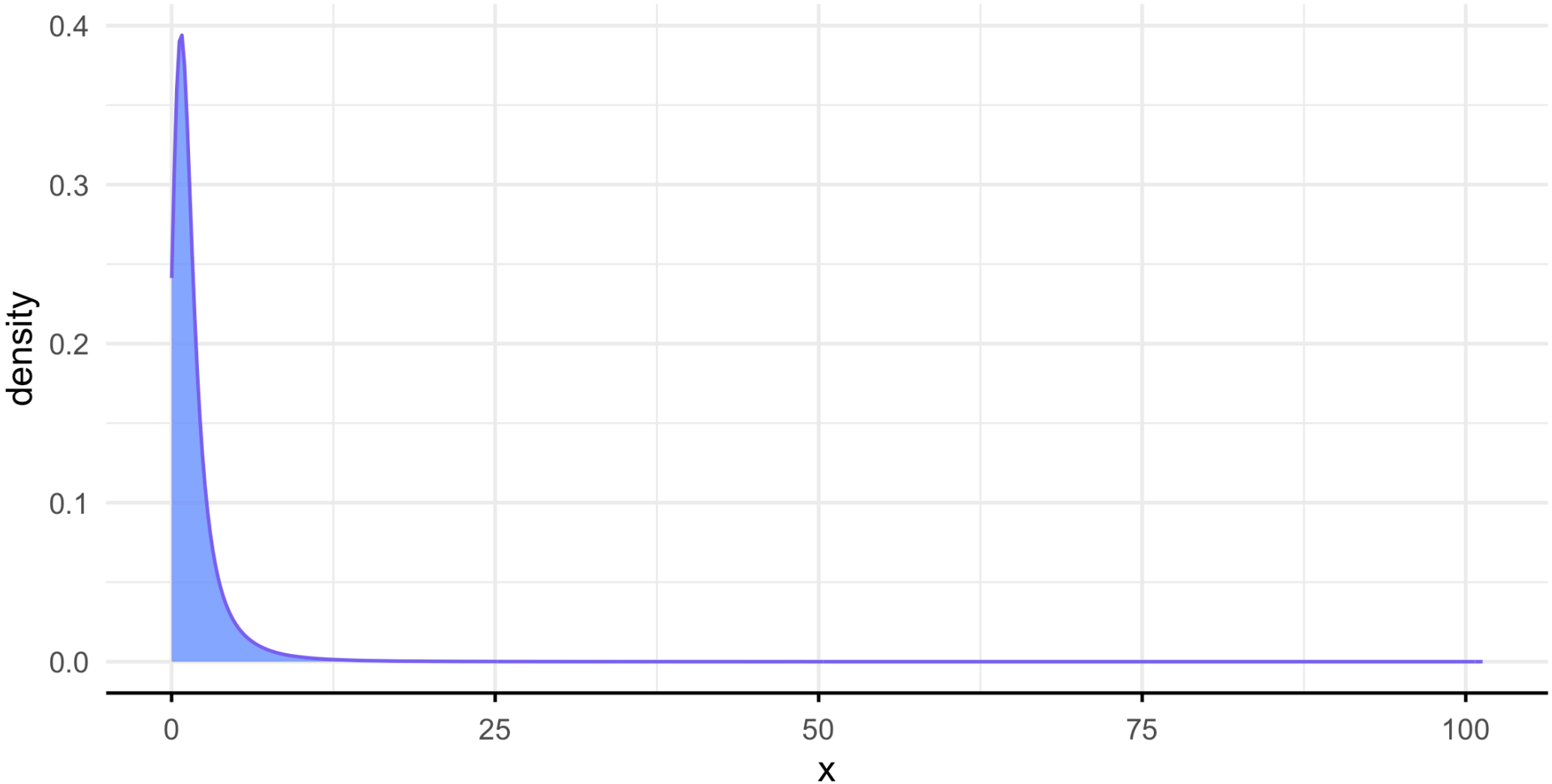
- **Densities** are often used instead of **histograms**
  - Both are based on the **same principle**, but densities are **continuous**
- We won't learn how to derive it in this course but the idea is the same
  - The **higher the value** on the y-axis, the **more observations** there are
- The density's **smoothness** can be tuned with the **bandwidth**:



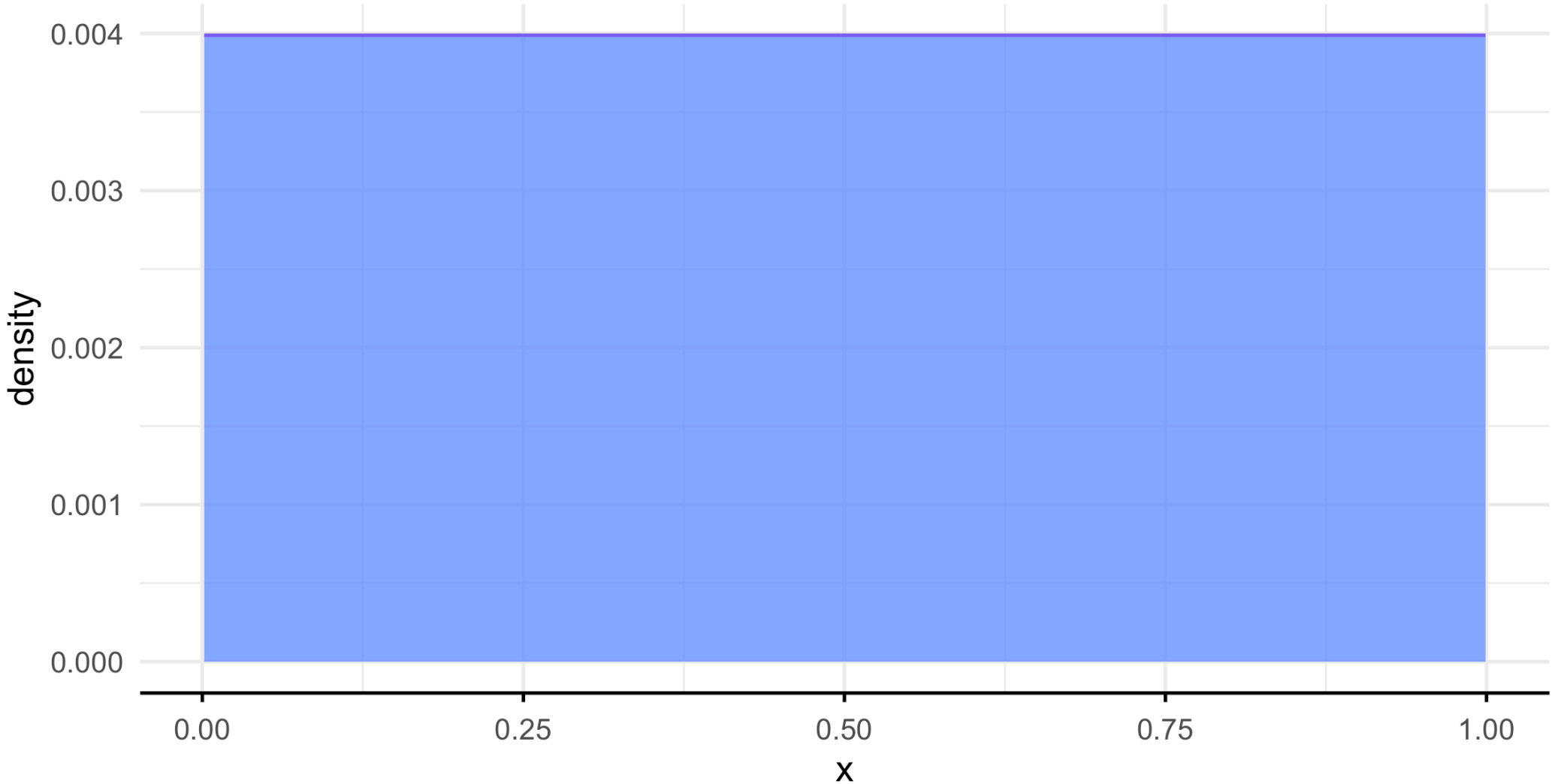
Common distributions: normal distribution



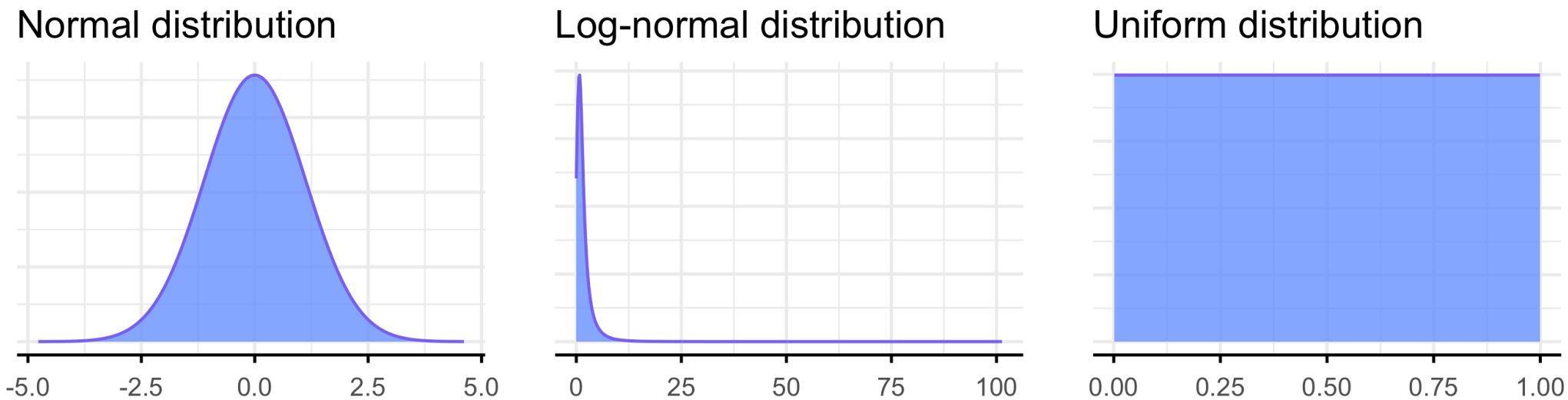
Common distributions: log-normal distribution



Common distributions: uniform distribution

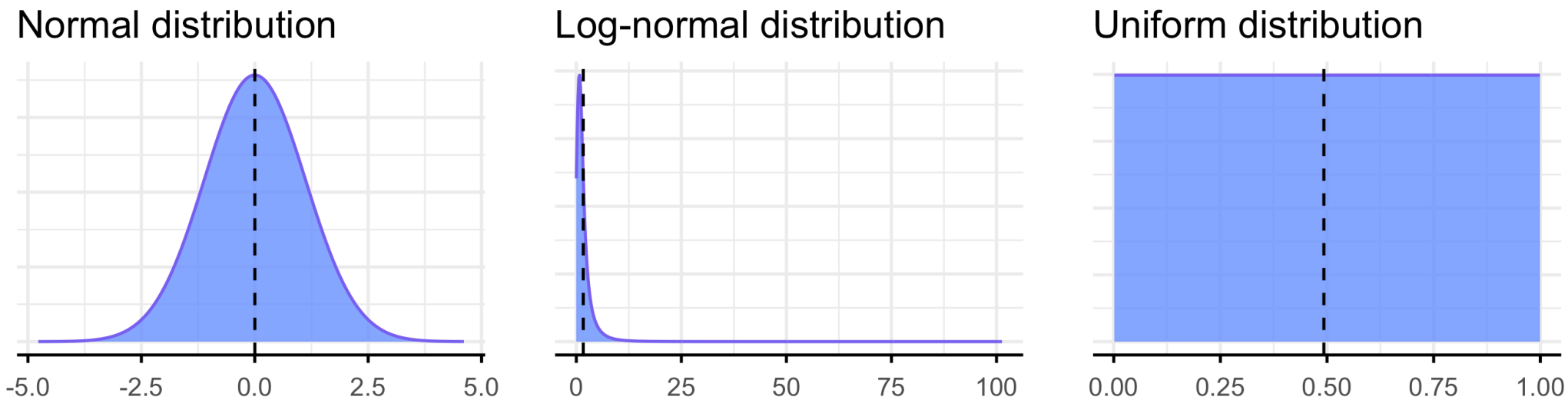


Common distributions: summarizing distributions



How to **summarize** these distributions with simple statistics?

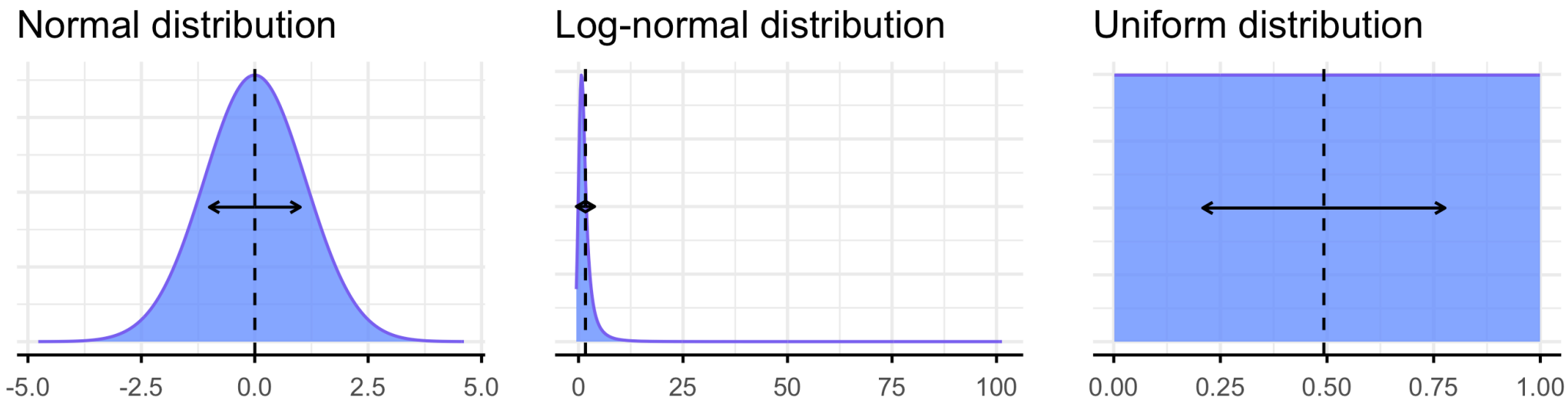
Common distributions: summarizing distributions



How to **summarize** these distributions with simple statistics?

- By describing their **central tendency** (e.g., mean, median)

Common distributions: summarizing distributions



How to **summarize** these distributions with simple statistics?

- By describing their **central tendency** (e.g., mean, median)
- And their **spread** (e.g., standard deviation, inter-quartile range)

Today's lecture

1. Summarizing data

- 1.1 Distributions ✓
- 1.2 Central tendency
- 1.3 Spread
- 1.4 Relationship between variables

2. Manipulating data

- 2.1 `dplyr` verbs
- 2.2 `group_by`
- 2.3 Joining data

3. Visualizing data

- 3.1 gg is for Grammar of Graphics
- 3.2 Different types of gaphs

Central tendency: mean

- The mean is the most common statistic to describe central tendencies
  - Take the number of hours worked in some European countries (Gethin and Saez, 2025):

Number of hours worked in 2023						
30	30	30	41	30	34	28
32	30	31	33	34	28	35

The mean is simply the sum of all the countries' hours of worked divided by the number of countries:

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

$$\frac{30 + 30 + 30 + 41 + 30 + 34 + 28 + 32 + 30 + 31 + 33 + 34 + 28 + 35}{14} = 3$$

Central tendency: mean

- The mean is the most common statistic to describe central tendencies
  - Take the number of hours worked in some European countries (Gethin and Saez, 2025):

Number of hours worked in 2023						
30	30	30	41	30	34	28
32	30	31	33	34	28	35

Note that it can also be expressed as the sum of each value weighted by its proportion in the distribution

$$\bar{x} = \frac{2}{14} \times 28 + \frac{5}{14} \times 30 + \frac{1}{14} \times 31 + \frac{1}{14} \times 32 + \frac{1}{14} \times 33 + \frac{2}{14} \times 34 + \frac{1}{14} \times 35 + \frac{1}{14} \times 41 = 31.86$$

Central tendency: median

- To obtain the median you first need to **sort the values**:

Number of hours worked in 2023

1	2	3	4	5	6	7	8	9	10	11	12	13	14
28	28	30	30	30	30	30	31	32	33	34	34	35	41

- The median is the value that **divides** the distribution into **two halves**
- When there is an even number of observations, the median is the average of the last value of the first half and the first value of the second half

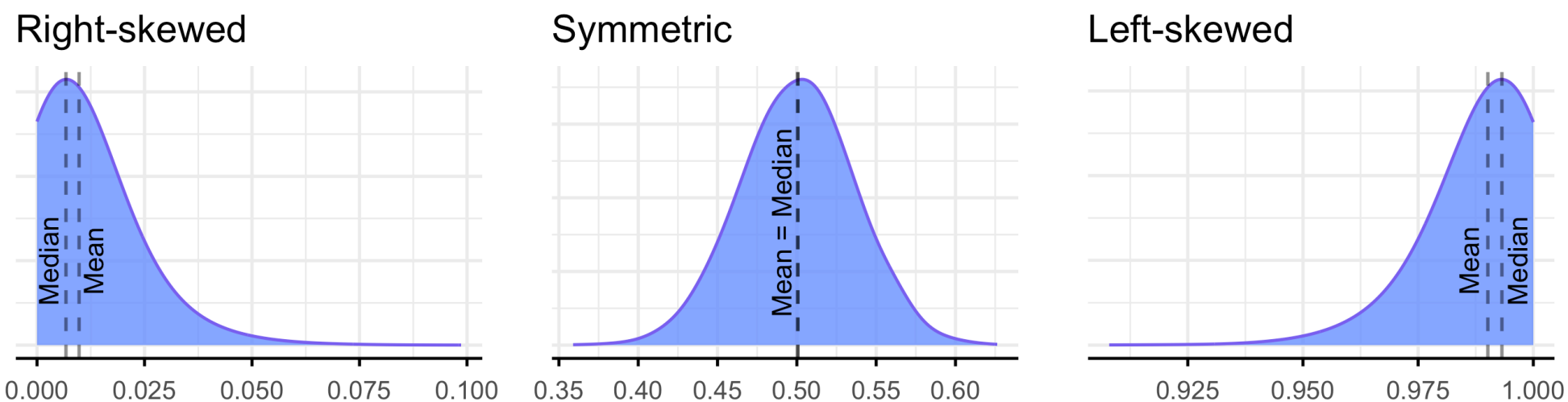
As we have 14 observations, the median is the average of the 7^{th} and the 8^{th} observations:

$$\text{Med}(x) = \begin{cases} x[\frac{N+1}{2}] & \text{if } N \text{ is odd} \\ \frac{x[\frac{N}{2}]+x[\frac{N}{2}+1]}{2} & \text{if } N \text{ is even} \end{cases} = \frac{30+31}{2} = 30.5$$

Mean vs. median: relative magnitude

The **relative magnitude** of the mean and the median depends on the **symmetry of the distribution**:

- The **mean is larger** than the median if the distribution is **right-skewed**
- The mean and the median are **equal** if the distribution is **symmetric**
- The **mean is lower** than the median if the distribution is **left-skewed**



Mean vs. median: relative magnitude

- What's an example of a **right-skewed** distribution?

→ income, wealth

- What's an example of a **symmetric** distribution?

→ height, exam scores (ideally)

- What's an example of a **left-skewed** distribution?

→ age at death, time spent on social media

Mean vs. median: robustness

- The **median** is **less sensitive** than the mean to thick tails and outliers
- For this reason we say that the median is a **robust statistic**

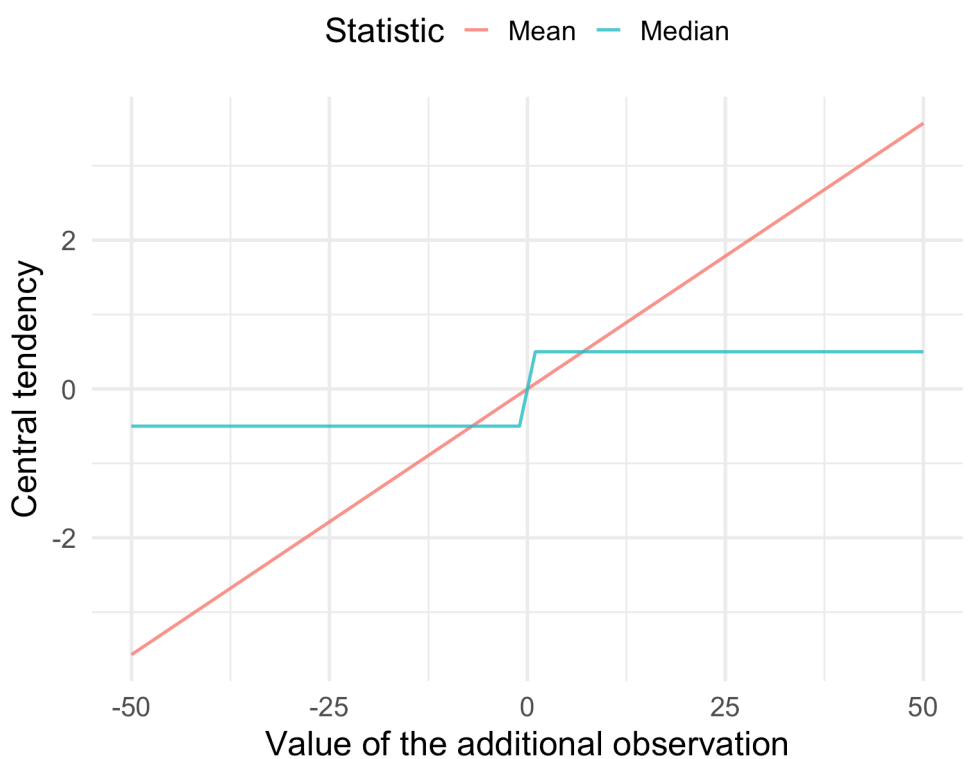
Let's illustrate that with a small example!

Consider the following variable:

-3	-2	-2	-1	-1	-1	0	1	1	1	2	2	3

How would the mean and the median **react** if we were to **add one single observation**?

- We can plot the value of the additional observation on the x axis and the value of the mean and the median on the y axis



Mean vs. median: in R

Both statistics have **dedicated R functions**

→ `mean` for **mean** and `median` for **median**:

```
1 variable <- c(1, 2, 4, 8, 12)
2 c(mean(variable), median(variable))
```

```
[1] 5.4 4.0
```

As always, you should **pay attention to NAs** when using these functions!

```
1 mean(c(1, 2, 3, 4, NA))
```

```
[1] NA
```

```
1 mean(c(1, 2, 3, 4, NA), na.rm = T)
```

```
[1] 2.5
```

Mean vs. median: with binary variable

- A **binary variable** is a variable that can take only **two values** *(e.g., male/female, accepted/rejected)*
  - Any binary variable can be expressed as a sequence of **0s and 1s**

Consider the following binary variable of length 4

0 1 1 1

- The **mean** of a binary variable is equal the the **percentage of 1s**:

$$\frac{0+1+1+1}{4} = \frac{3}{4} = 75\%$$

- The **median** of a binary variable is equal to the **mode** *(mode = most frequent value of a variable)*

$$\frac{1+1}{2} = 1$$

Today's lecture

1. Summarizing data

- 1.1 Distributions ✓
- 1.2 Central tendency ✓
- 1.3 Spread
- 1.4 Relationship between variables

2. Manipulating data

- 2.1 `dplyr` verbs
- 2.2 `group_by`
- 2.3 Joining data

3. Visualizing data

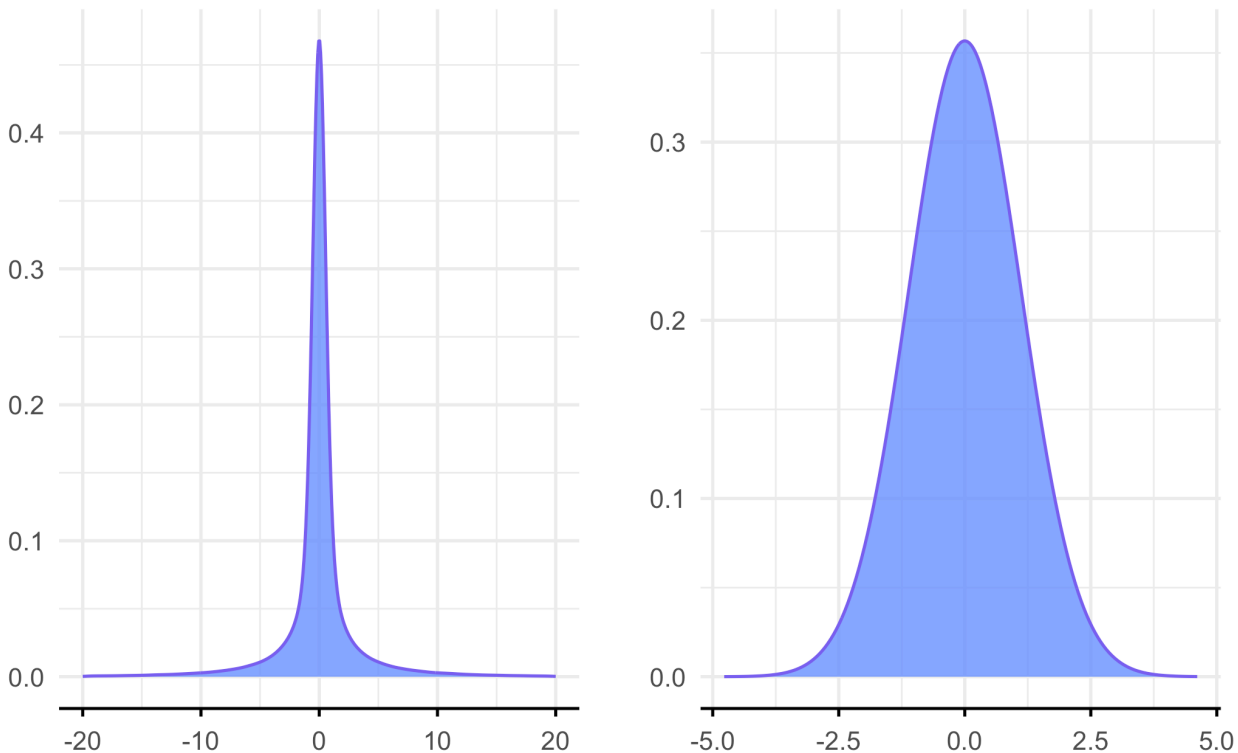
- 3.1 gg is for Grammar of Graphics
- 3.2 Different types of gaphs

Range

The **most intuitive** statistic to describe the spread of a variable is probably

- **The range: the minimum and maximum value it can take**

But consider the following two distributions:

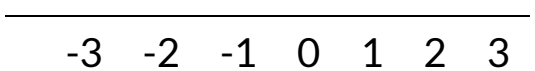


- In the presence of outliers or very skewed distributions, the **full range** of a variable **may not be representative** of what we mean by *'spread'*
- That's why we tend to prefer **inter-quantile** ranges

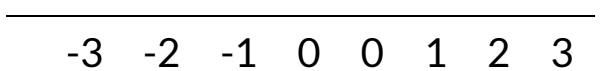
Quantiles

- Quantiles **divide** the population into **groups of equal size**
  - The median divides the population into 2 **groups** of equal size
  - Quartiles divide the population into 4 **groups** of equal size
  - There are also **terciles**, **quintiles**, **deciles**, and so on

- To **compute** quartiles: divide the ordered variable according to the median
  - The lower quartile value is the median of the lower half of the data
  - The upper quartile value is the median of the upper half of the data
  - *If there is an odd number of data points in the original ordered data set, don't include the median in either half*



$$Q_1 = -2, \;\; Q_2 = 0, \;\; Q_3 = 2$$

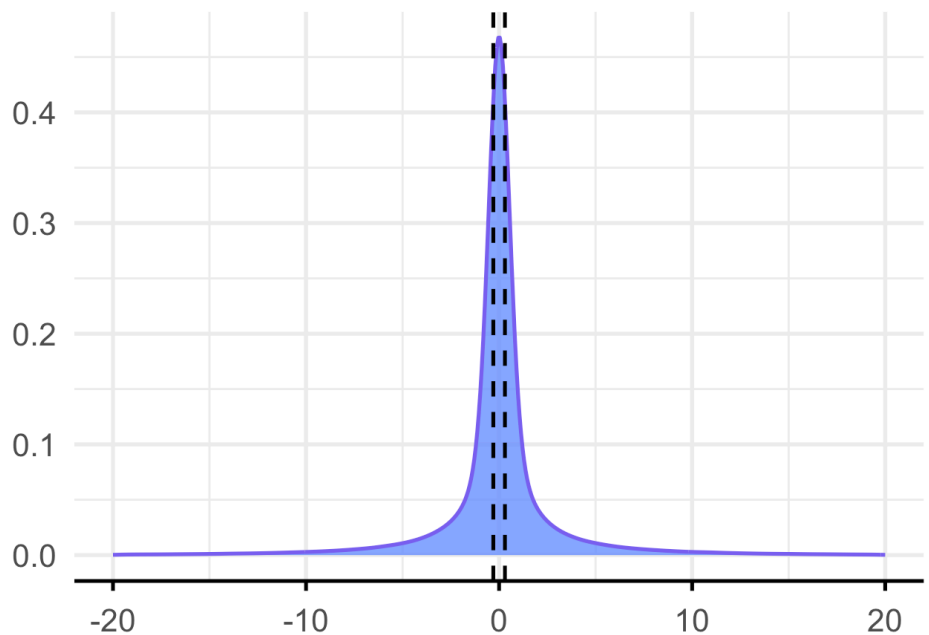


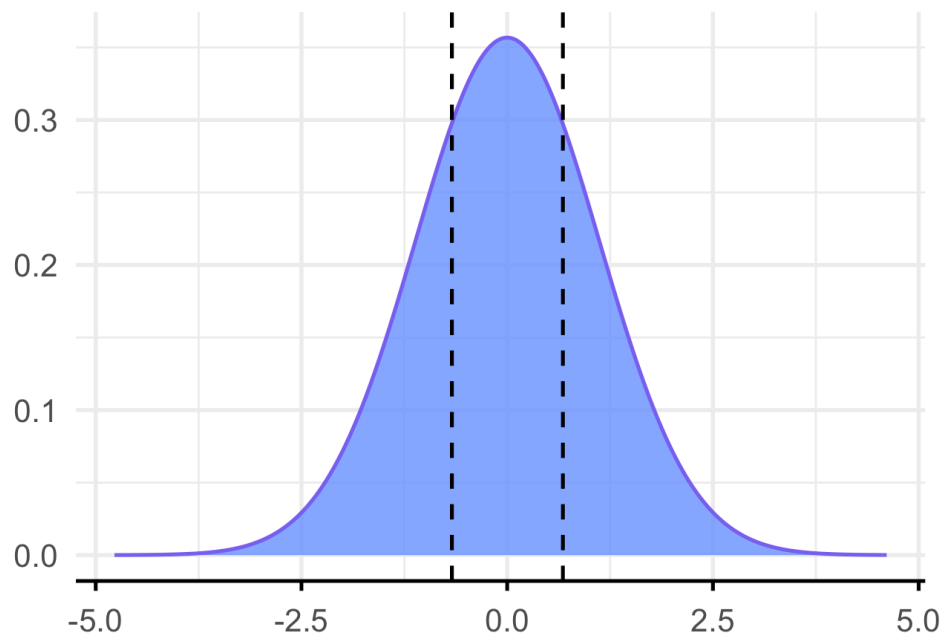
$$Q_1 = -1.5, \;\; Q_2 = 0, \;\; Q_3 = 1.5$$

Interquartile range

The **interquartile range** is the difference between the third and the first quartile:
$\text{IQR} = Q_3 - Q_1$

Put differently, it corresponds to the **bounds** of the set which contains the **middle half** of the distribution





Variance

- The **variance** is a way to quantify how the values of a variable tend to **deviate** from their **mean**
  - If values tend to be **close to the mean**, then the **spread is low**
  - If values tend to be far **from the mean**, then the **spread is large**

Can we just take the **average deviation** from the mean?

x	mean(x)	x - mean(x)
1	2.5	-1.5
4	2.5	1.5
-3	2.5	-5.5
8	2.5	5.5

By construction it would **always be 0**: values above and under the mean compensate

- But we can use the **absolute value** of each deviation: $|x_i - \bar{x}|$
- Or their **square**: $(x_i - \bar{x})^2$

Variance

The **variance** is computed by **averaging the squared deviations from the mean**:

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

Variance and standard deviation

- Because the **variance** is a **sum of squares**, it can get **quite big** compared to the other statistics like the mean, the median or the interquartile range.
- To express the spread in the **same unit** as the data, we can take the **square root** of the variance, which is called the **standard deviation**:

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

- In a way, *the standard deviation is to the mean what the IQR is to the median*

Standard deviation vs. interquartile range

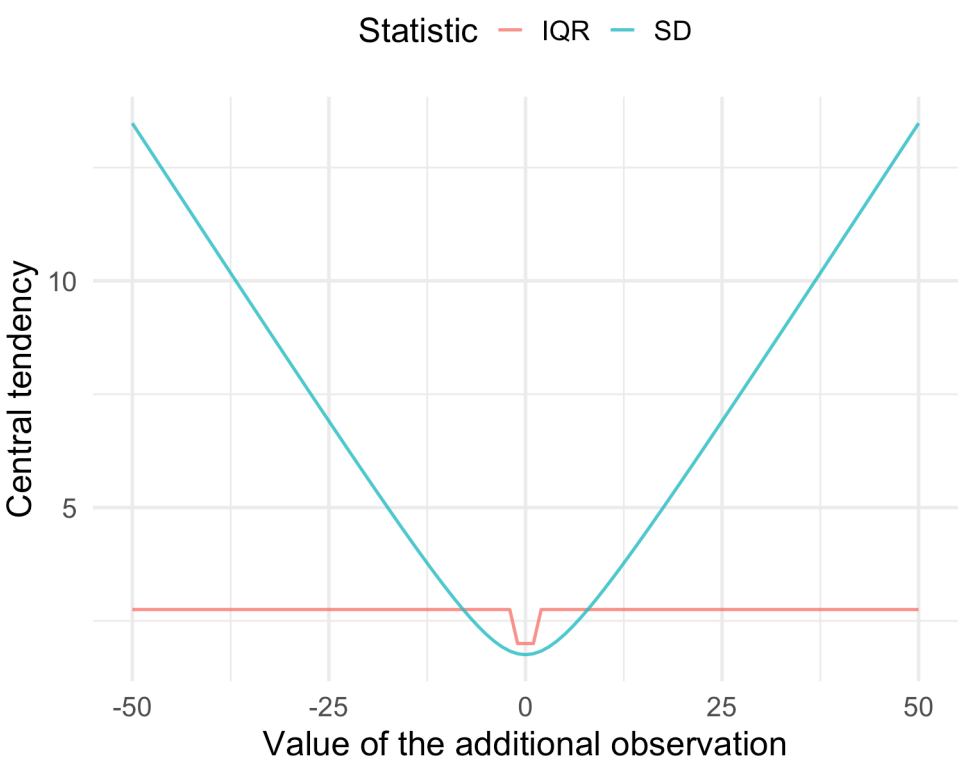
- Remember that the median is **less sensitive** than the mean to thick tails and outliers
- This is also the case for the **IQR** relative to the **standard deviation**

Let's go back to our previous example!

Consider the following variable:

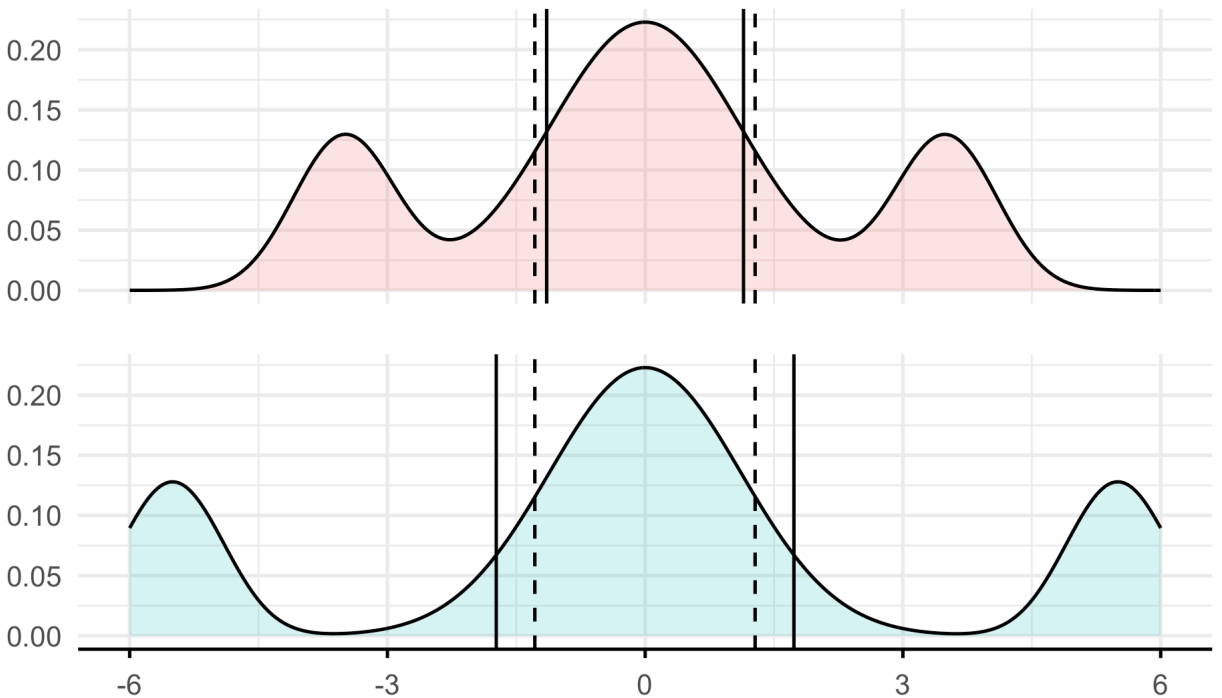
-3	-2	-2	-1	-1	-1	0	1	1	1	2	2	3

- How would the standard deviation and the IQR **react** if we were to **add one single observation**?
  - We can plot the value of the additional observation on the x axis and the value of the mean and the median on the y axis



Standard deviation vs. interquartile range

- But like for the median vs. the mean, it does **not** mean that one is **better than the other**
  - They just **capture different things**



These two distributions

- Have the **same interquartile range**
- Have **different standard deviations**

Standard deviation vs. interquartile range: in R

Both statistics have **dedicated R functions**

→ `sd` for the **standard deviation** and `IQR` for the **interquartile range**

```
1 variable <- c(0, 1, 3, 4, 6, 7, 8, 10, 11)
2 c(sd(variable), IQR(variable))
```

```
[1] 3.844188 5.000000
```

You can obtain the **quantiles** of a variable using the `quantile()` function

```
1 quantile(variable)
```

```
  0%  25%  50%  75% 100% 
   0    3    6    8   11 
```

→ ***See the help file ?quantile() for more info on quantile computation***

Today's lecture

1. Summarizing data

- 1.1 Distributions ✓
- 1.2 Central tendency ✓
- 1.3 Spread ✓
- 1.4 Relationship between variables

2. Manipulating data

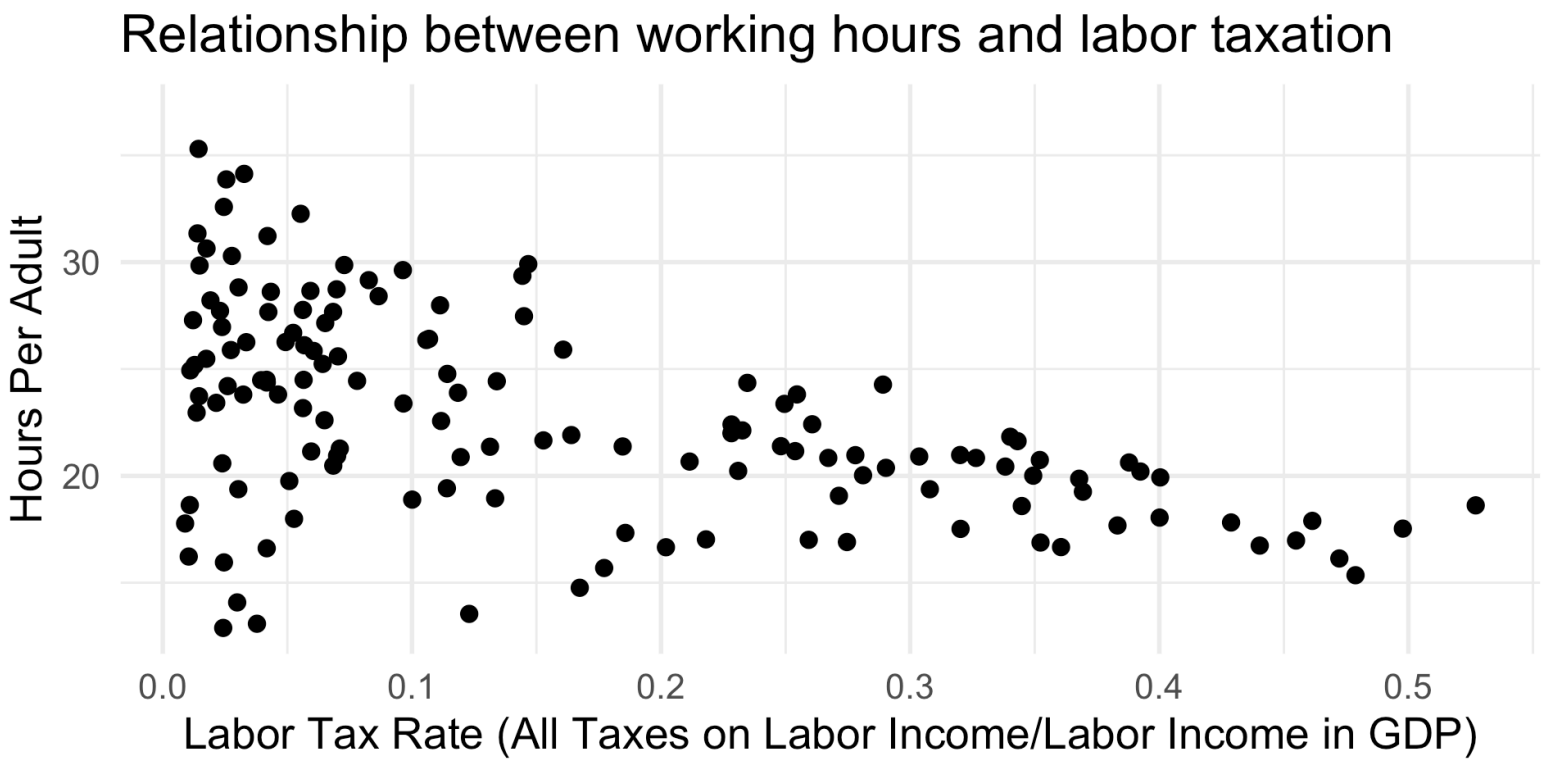
- 2.1 `dplyr` verbs
- 2.2 `group_by`
- 2.3 Joining data

3. Visualizing data

- 3.1 gg is for Grammar of Graphics
- 3.2 Different types of gaphs

Covariance and correlation

How can we characterize the relationship between two variables?



There are two main statistics:

1. The **covariance**
2. The **correation**

Covariance and correlation

- The **covariance** is a measure of **joint variability** between two variables:

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

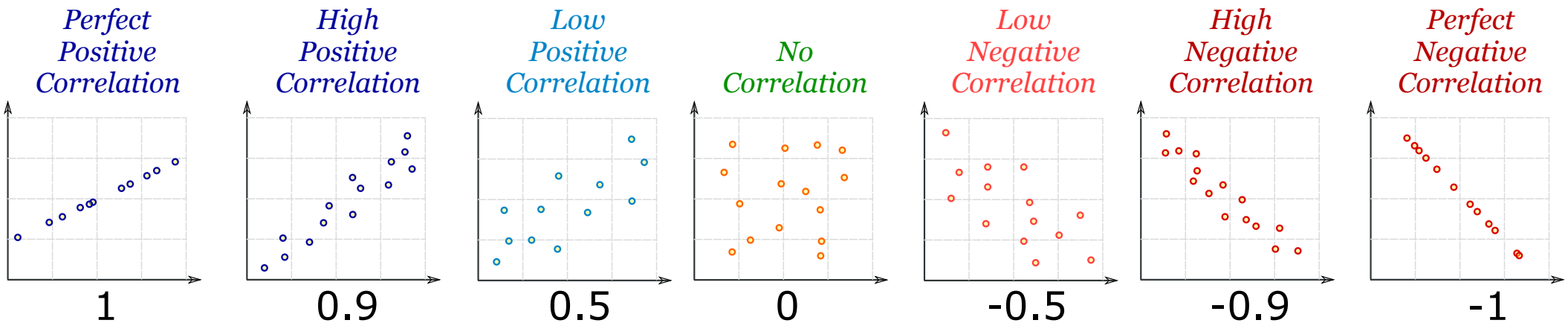
- Difficult to interpret because sensitive to the variables' dispersions from the mean $\rightarrow$ correlation.

- The **correlation** is a measure of the strength of the **linear association** between two variables:

$$Corr(x, y) = \frac{Cov(x, y)}{\sqrt{Var(x)}\sqrt{Var(y)}}$$

Correlation

- Correlation is **always between -1 and 1!**



Covariance and correlation: in R

Both statistics have **dedicated R functions**

→ `cov` for **covariance** and `cor` for **correlation**:

```
1 variables <- data.frame(x = c(1, 2, 4, 8, 12),
2                         y = c( 3, 6, 8, 9, 6))
3 c(cov(variables$x, variables$y),
4   cor(variables$x, variables$y))
```

```
[1] 4.550000 0.433353
```

As always, you should **pay attention to NAs** when using these functions!

```
1 variables <- data.frame(x = c(1, 2, 4, 8, 12, NA),
2                         y = c( 3, 6, 8, 9,NA,  6))
3 c(cov(variables$x, variables$y, use = "complete.obs"),
4   cor(variables$x, variables$y, use = "complete.obs"))
```

```
[1] 7.1666667 0.8750037
```

Your turn! #1

10:00

Load data on global working hours in 2023 from Gethin and Saez (2025). You can find it here.

```
1 global_working_hours <- read.csv("https://www.dropbox.com/scl/fi/ln4ie7ik36opjx5juifqs/hours_worked_c
```

1. Compute the mean of `hours_worked` (hours per adult) and `hours_worker` (hours per worker). Why is there such a difference?
2. Compute the median of `hours_worked` and `hours_worker`. Is there a big difference with the mean? What does this suggest about the distribution of (average) working hours across countries?
3. Compute the three quartiles of the distribution of `hours_worker`. What's the 75th percentile? Try to guess which country this is.
4. Compute the interquartile range of `hours_worker_men` and `hours_worker_women`. What do you notice?
5. Compute the correlation between `hours_worked` and `tax_labor`. (Hint: remember what `NA` stands for.)

Today's lecture

1. Summarizing data ✓
    - 1.1 Distributions ✓
    - 1.2 Central tendency ✓
    - 1.3 Spread ✓
    - 1.4 Relationship between variables ✓
2. **Manipulating data**
    - 2.1 `dplyr` verbs
    - 2.2 `group_by`
    - 2.3 Joining data
3. Visualizing data
    - 3.1 gg is for Grammar of Graphics
    - 3.2 Different types of gaphs

Manipulating data

The `tidyverse` package suit

- The `tidyverse` is a collection of packages that facilitate data manipulation and visualization.
- We will use `dplyr` to manipulate data and `ggplot2` to visualize data.
- You don't need to load each package individually, instead you can just:

```
1  library(tidyverse)
```

- Recall that you **don't need** to re-install a package each time, but you do **need to load it**

`dplyr` verbs

`dplyr` is a **grammar** of data manipulation providing very **user-friendly functions** to handle the most common **data manipulation** tasks:

- `mutate()`: add/modify variables
- `select()`: keep/drop variables (columns)
- `filter()`: keep/drop observations (rows)
- `arrange()`: sort rows according to given variable(s)
- `summarise()`: aggregate data into statistics
- `group_by()`: compute the above verbs by groups

A very handy **operator** to use with the **dplyr** grammar is the **pipe `|>`**[1]:

- You can basically read **df |> mutate()** as *"apply function mutate() to object df"*
- With this operator you can easily **chain the operations** you apply to an object

dplyr: an example

We will use data on global working hours, but this time since 1900. You can find the data here.

```
1 global_working_hours <- read.csv("https://www.dropbox.com/scl/fi/aq6nlnuun9o8bk86h89a2/hours_worked_p
2
3 str(global_working_hours)
```

```
'data.frame':   2565 obs. of  9 variables:
 $ country           : chr  "United Arab Emirates" "United Arab Emirates" "United Arab Emirates" "United
Arab Emirates" ...
 $ region            : chr  "Middle East and North Africa" "Middle East and North Africa" "Middle East
and North Africa" "Middle East and North Africa" ...
 $ year              : int  2017 2018 2019 2022 2023 2007 2008 2011 2012 2013 ...
 $ hours_worked      : num  44.1 40.9 40.8 37.9 37.1 ...
 $ hours_worked_men  : num  49 47.2 47.3 44.5 43.2 ...
 $ hours_worked_women: num  32 26.4 28.5 25.8 25.3 ...
 $ hours_worker      : num  55.9 51.7 52.1 50.4 48.4 ...
 $ hours_worker_men  : num  53.6 51.4 52.1 50.5 48.3 ...
 $ hours_worker_women: num  65.2 53.2 52.2 50.2 48.4 ...
```

`dplyr`: an example

```
1  global_working_hours
```

```
# A tibble: 2,565 × 9
   country          region  year hours_wo
hours_worked_women
   <chr>            <chr>  <int>        <
<dbl>
 1 United Arab Em… Middl…  2017
32.0
 2 United Arab Em… Middl…  2018
26.4
 3 United Arab Em… Middl…  2019
28.5
 4 United Arab Em… Middl…  2022
25.8
 5 United Arab Em… Middl…  2023
25.3
 6 Afghanistan     Middl…  2007
13.1
 7 Afghanistan     Middl…  2008
12.8
```

`dplyr`: an example

```
1  global_working_hours |>
2    select(country, year, hours_worked) # keep/drop certain column
```

```
# A tibble: 2,565 × 3
   country              year hours_work
   <chr>               <int>        <db
 1 United Arab Emirates  2017         44
 2 United Arab Emirates  2018         40
 3 United Arab Emirates  2019         40
 4 United Arab Emirates  2022         37
 5 United Arab Emirates  2023         37
 6 Afghanistan           2007         22
 7 Afghanistan           2008         22
 8 Afghanistan           2011         20
 9 Afghanistan           2012         19
10 Afghanistan           2013         16
# ℹ 2,555 more rows
```

`dplyr`: an example

```
1  global_working_hours |>
2    select(country, year, hours_worker) |>
3    mutate(hours_worked_greater_40 = (hours_worker >= 40)) # creat
```

```
# A tibble: 2,565 × 4
   country                year hours_work
   <chr>                 <int>        <db
 1 United Arab Emirates   2017          55
 2 United Arab Emirates   2018          51
 3 United Arab Emirates   2019          52
 4 United Arab Emirates   2022          50
 5 United Arab Emirates   2023          48
 6 Afghanistan            2007          36
 7 Afghanistan            2008          35
 8 Afghanistan            2011          42
 9 Afghanistan            2012          42
10 Afghanistan            2013          37
# ℹ 2,555 more rows
```

`dplyr`: an example

```
1  global_working_hours |>
2    select(country, year, hours_worker) |>
3    mutate(hours_worked_greater_40 = (hours_worker >= 40)) |>
4    filter(country == "France") # keep/drop certain rows
```

```
# A tibble: 55 × 4
   country  year hours_worker hours_work
   <chr>   <int>        <dbl> <lgl>
 1 France   1968         44.8 TRUE
 2 France   1969         44.2 TRUE
 3 France   1970         43.2 TRUE
 4 France   1971         43.0 TRUE
 5 France   1972         42.9 TRUE
 6 France   1973         42.6 TRUE
 7 France   1974         42.3 TRUE
 8 France   1975         40.4 TRUE
 9 France   1976         40.7 TRUE
10 France   1977         40.5 TRUE
# ℹ 45 more rows
```

`dplyr`: an example

```
1 global_working_hours |>
2   select(country, year, hours_worker) |>
3   mutate(hours_worked_greater_40 = (hours_worker >= 40)) |>
4   filter(country == "France") |>
5   summarise(mean_hours = mean(hours_worker), # aggregate into st
6             num_greater_40 = sum(hours_worked_greater_40))
```

```
# A tibble: 1 × 2
  mean_hours num_greater_40
       <dbl>          <int>
1       35.8             11
```

Useful `mutate` functions

- There are two very handy functions to use within `mutate()`:

ifelse()

```
1 global_working_hours |>
2   select(country, year) |>
3   mutate(post_2000 = ifelse(year >= 2000,
4                             "21th century",
5                             "20th century")) |>
6   head()
```

```
# A tibble: 6 × 3
  country              year post_2000
  <chr>               <int> <chr>
1 United Arab Emirates  2017 21th century
2 United Arab Emirates  2018 21th century
3 United Arab Emirates  2019 21th century
4 United Arab Emirates  2022 21th century
5 United Arab Emirates  2023 21th century
6 Afghanistan           2007 21th century
```

case_when()

```
1 global_working_hours |>
2   select(country, year) |>
3   mutate(year_gp = case_when(year <= 1950 ~ "19
4                              year %in% 1951:199
5                              year >= 2000 ~ "20
6   head()
```

```
# A tibble: 6 × 3
  country              year year_gp
  <chr>               <int> <chr>
1 United Arab Emirates  2017 2000-2023
2 United Arab Emirates  2018 2000-2023
3 United Arab Emirates  2019 2000-2023
4 United Arab Emirates  2022 2000-2023
5 United Arab Emirates  2023 2000-2023
6 Afghanistan           2007 2000-2023
```

Today's lecture

1. Summarizing data ✓
    - 1.1 Distributions ✓
    - 1.2 Central tendency ✓
    - 1.3 Spread ✓
    - 1.4 Relationship between variables ✓
2. **Manipulating data**
    - 2.1 `dplyr` verbs ✓
    - 2.2 `group_by`
    - 2.3 Joining data
3. Visualizing data
    - 3.1 gg is for Grammar of Graphics
    - 3.2 Different types of gaphs

`group_by()` and `mutate`

With `group_by()` you can perform **computations separately** for the **different categories of a variable**

```
1 global_working_hours |>
2   select(year, country, region, hours_worker) |
3   mutate(mean_all = mean(hours_worker)) |>
4   arrange(country, year) |>
5   head(10)
```

```
# A tibble: 10 × 5
    year country     region
hours_worker mean_all
   <int> <chr>       <chr>
<dbl>    <dbl>
 1  2007 Afghanistan Middle East and North Africa
36.5     39.5
 2  2008 Afghanistan Middle East and North Africa
35.8     39.5
 3  2011 Afghanistan Middle East and North Africa
42.9     39.5
 4  2012 Afghanistan Middle East and North Africa
42.3     39.5
 5  2013 Afghanistan Middle East and North Africa
37.4     39.5
 6  2014 Afghanistan Middle East and North Africa
37.7     39.5
 7  2017 Afghanistan Middle East and North Africa
38.0     39.5
```

```
1 global_working_hours |>
2   select(year, country, region, hours_worker) |
3   group_by(region) |>
4   mutate(mean_reg_yr = mean(hours_worker)) |>
5   arrange(country, year) |>
6   head(10)
```

```
# A tibble: 10 × 5
# Groups:   region [2]
    year country     region
hours_worker mean_reg_yr
   <int> <chr>       <chr>
<dbl>       <dbl>
 1  2007 Afghanistan Middle East and North Africa
36.5        44.1
 2  2008 Afghanistan Middle East and North Africa
35.8        44.1
 3  2011 Afghanistan Middle East and North Africa
42.9        44.1
 4  2012 Afghanistan Middle East and North Africa
42.3        44.1
 5  2013 Afghanistan Middle East and North Africa
37.4        44.1
 6  2014 Afghanistan Middle East and North Africa
37.7        44.1
 7  2017 Afghanistan Middle East and North Africa
```

`group_by()` and `mutate`

With `group_by()` you can perform **computations separately** for the **different categories of a variable**

```
global_working_hours |>
  select(year, country, region, hours_worker) |
  mutate(mean_all = mean(hours_worker)) |>
  arrange(country, year) |>
  count(region, mean_all)
```

```
# A tibble: 8 × 3
  region                          mean_all     n
  <chr>                              <dbl> <int>
1 East and Southeast Asia             39.5   246
2 Eastern Europe and ex-USSR          39.5   503
3 Latin America                       39.5   515
4 Middle East and North Africa        39.5   167
5 South Asia                          39.5    64
6 Sub-Saharan Africa                  39.5   159
7 United States                       39.5   123
8 Western Europe and Anglosphere      39.5   788
```

```
global_working_hours |>
  select(year, country, region, hours_worker) |
  group_by(region) |>
  mutate(mean_reg_yr = mean(hours_worker)) |>
  arrange(country, year) |>
  count(region, mean_reg_yr)
```

```
# A tibble: 8 × 3
# Groups:   region [8]
  region                          mean_reg_yr     n
  <chr>                                 <dbl> <int>
1 East and Southeast Asia                43.3   246
2 Eastern Europe and ex-USSR             38.2   503
3 Latin America                          42.5   515
4 Middle East and North Africa           44.1   167
5 South Asia                             45.6    64
6 Sub-Saharan Africa                     41.3   159
7 United States                          42.8   123
8 Western Europe and Anglosphere         34.9   788
```

- `count` is a very useful `dplyr` function for (cross-)tabulation

`group_by()` and `summarise`

- It is particularly **useful with `summarise()`**
  - summarise keeps the grouping variable
  - and computes **statistics for each category**

```
  1  global_working_hours |>
  2    group_by(region) |>
  3    summarise(mean_hours = mean(hours_worker))
```

```
# A tibble: 8 × 2
  region                          mean_hours
  <chr>                                <dbl>
1 East and Southeast Asia               43.3
2 Eastern Europe and ex-USSR            38.2
3 Latin America                         42.5
4 Middle East and North Africa          44.1
5 South Asia                            45.6
6 Sub-Saharan Africa                    41.3
7 United States                         42.8
8 Western Europe and Anglosphere        34.9
```

mutate() $\neq$ summarise()

- `mutate()` takes an operation that converts:
  - **A vector into another vector**
- `summarise()` takes an operation that converts:
  - **A vector into a value**

`group_by()` and `summarise`

- It is particularly **useful with `summarise()`**
  - summarise keeps the grouping variable
  - and computes **statistics for each category**

```
  1  global_working_hours |>
  2    group_by(region) |>
  3    summarise(mean_hours = mean(hours_worker))
```

```
# A tibble: 8 × 2
  region                          mean_hours
  <chr>                                <dbl>
1 East and Southeast Asia               43.3
2 Eastern Europe and ex-USSR            38.2
3 Latin America                         42.5
4 Middle East and North Africa          44.1
5 South Asia                            45.6
6 Sub-Saharan Africa                    41.3
7 United States                         42.8
8 Western Europe and Anglosphere        34.9
```

ungrouping

- `group_by()` applies to **all subsequent operations**
- To cancel its effect you must `ungroup()` the data

```
1  global_working_hours |>
2    group_by(region) |>
3    summarise(mean_hours = mean(hours_worker)) |>
4    ungroup() |>
5    ...
```

`group_by()` using `.by()`

`dplyr` recently introduced the `.by` operator:

```
  1  global_working_hours |>
  2    summarise(mean_hours = mean(hours_worker),
  3              .by = region) |>
  4    head(10)
```

```
# A tibble: 8 × 2
  region                          mean_hours
  <chr>                                <dbl>
1 Middle East and North Africa          44.1
2 Eastern Europe and ex-USSR            38.2
3 Sub-Saharan Africa                    41.3
4 Latin America                         42.5
5 Western Europe and Anglosphere        34.9
6 South Asia                            45.6
7 East and Southeast Asia               43.3
8 United States                         42.8
```

- Clean alternative to `group_by`
- Works with all (relevant) `dplyr` verbs and with multiple variables `c(var1, var2)`
- No need to `ungroup()`

Today's lecture

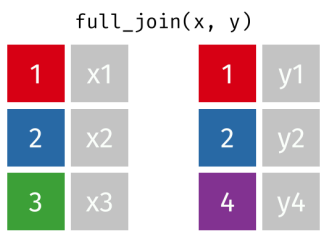
1. Summarizing data ✓
    - 1.1 Distributions ✓
    - 1.2 Central tendency ✓
    - 1.3 Spread ✓
    - 1.4 Relationship between variables ✓
2. **Manipulating data**
    - 2.1 `dplyr` verbs ✓
    - 2.2 `group_by` ✓
    - 2.3 Joining data
3. Visualizing data
    - 3.1 gg is for Grammar of Graphics
    - 3.2 Different types of gaphs

Joining data together: `_join()`

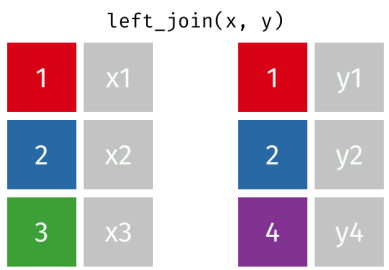
The last `dplyr` verb we will cover enable merging datasets together, a very common operation.

They come in different varieties:

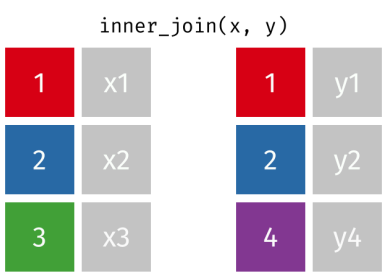
- `full_join()`:



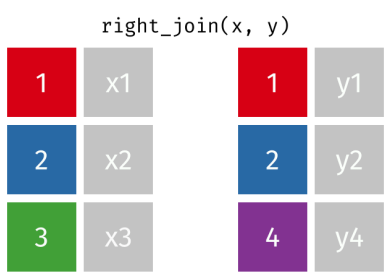
- `left_join()`:



- `inner_join()`:



- `right_join()`:



Joining data together: `_join()`

Imagine you have 2023 working hours in one dataset and GDP in another:

```
1  working_hours <- read.csv("https://www.dropbox.
2
3  gdp <- read.csv("https://www.dropbox.com/scl/fi
4
5  working_hours |>
6    head(10)
```

	country	year	hours_worked
1	United Arab Emirates	2017	44.07154
2	United Arab Emirates	2018	40.94445
3	United Arab Emirates	2019	40.82006
4	United Arab Emirates	2022	37.88909
5	United Arab Emirates	2023	37.12018
6	Afghanistan	2007	22.55852
7	Afghanistan	2008	22.08803
8	Afghanistan	2011	20.15557
9	Afghanistan	2012	19.52261
10	Afghanistan	2013	16.40564

Joining data together: `_join()`

Imagine you have 2023 working hours in one dataset and GDP in another:

```
1  working_hours <- read.csv("https://www.dropbox.
2
3  gdp <- read.csv("https://www.dropbox.com/scl/fi
4
5  gdp |>
6    head(10)
```

```
                country year          gdp
1  United Arab Emirates 2017 704732266496
2  United Arab Emirates 2018 713991258112
3  United Arab Emirates 2019 721905123328
4  United Arab Emirates 2022 772207411200
5  United Arab Emirates 2023 798492196864
6           Afghanistan 2007  71506411520
7           Afghanistan 2008  70340698112
8           Afghanistan 2011  87721918464
9           Afghanistan 2012  96194863104
10          Afghanistan 2013 102418915328
```

Joining data together: `_join()`

Imagine you have 2023 working hours in one dataset and GDP in another:

```
1 working_hours <- read.csv("https://www.dropbox.
2
3 gdp <- read.csv("https://www.dropbox.com/scl/fi
4
5 working_hours |>
6   left_join(gdp) |>
7   head(10)
```

```
                country year hours_worked
gdp
1  United Arab Emirates 2017     44.07154
704732266496
2  United Arab Emirates 2018     40.94445
713991258112
3  United Arab Emirates 2019     40.82006
721905123328
4  United Arab Emirates 2022     37.88909
772207411200
5  United Arab Emirates 2023     37.12018
798492196864
6           Afghanistan 2007     22.55852
71506411520
7           Afghanistan 2008     22.08803
70340698112
8           Afghanistan 2011     20.15557
87721918464
9           Afghanistan 2012     19.52261
```

Last but not least: the `tidylog` package

- `dplyr` is extremely powerful and convenient but it has one drawback: it provides no information when performing operations
- `tidylog` is the solution to this problem!

```
install.packages("tidylog") # only not installe
library(tidylog)
```

```
working_hours |>
  left_join(gdp) |>
  head(10)
```

```
Joining with `by = join_by(country, year)`
left_join: added one column (gdp)
> rows only in working_hours 0
> rows only in gdp ( 0)
> matched rows 2,565
> =======
> rows total 2,565

                country year hours_worked
gdp
1  United Arab Emirates 2017     44.07154
704732266496
2  United Arab Emirates 2018     40.94445
713991258112
3  United Arab Emirates 2019     40.82006
721905123328
4  United Arab Emirates 2022     37.88909
772207411200
5  United Arab Emirates 2023     37.12018
```

Your turn! #2

10:00

Load data on global working hours since 1900 from Gethin and Saez (2025). You can find it here.

```
1 global_working_hours_panel <- read.csv("https://www.dropbox.com/scl/fi/aq6nlnuun9o8bk86h89a2/hours_wo
```

1. Tabulate the number of observations (countries) per year.
2. Which countries had average workers hours per week (`hours_worker`) greater than 48 in 2023?
3. In which years did at least one Latin American country have average workers hours between 30 and 32 hours. What happened that year?
4. Compute the max of `hours_worker` for countries in the `Western Europe and Anglosphere` and `United States` regions, separately, in 2023. (Hint: the `%in%` operator will come in handy.)
5. Compute the average difference in hours worked per adult (`hours_worked`) between men and women in 1990, 2000, 2010 and 2020. What do you observe?

A few words on learning R

When things do not work the way you want, **NAs are the usual suspects**

For instance, this is how the mean function reacts to NAs:

```
1 mean(c(1, 2, NA))
```

```
[1] NA
```

```
1 mean(c(1, 2, NA), na.rm = TRUE)
```

```
[1] 1.5
```

You should systematically **check for NAs!**

```
1 is.na(c(1, 2, NA))
```

```
[1] FALSE FALSE  TRUE
```

A few words on learning R

Don’t pipe blindfolded!

- **Check** that each command does what it’s expected to do
- View or print your data **at each step**

```
  1  global_working_hours |>
  2    select(country, region, year, hours_worked) |>
  3    head(1)
```

```
# A tibble: 1 × 4
  country              region                         year hours_worked
  <chr>                <chr>                         <int>        <dbl>
1 United Arab Emirates Middle East and North Africa  2017         44.1
```

```
  1  global_working_hours |>
  2    select(country, region, year, hours_worked) |>
  3    filter(country == "France" & year == 2023)
```

```
# A tibble: 0 × 4
# ℹ 4 variables: country <chr>, region <chr>, year <int>, hours_worked <dbl>
```

```
  1    head(1)
```

```
[1] 1
```

Where to find help

Oftentimes things don't work either because:

- **You don't understand** a function's argument
- Or **you don't know** that there exists an argument that you should use

This is precisely what **help files** are made for

- Every function has a help file, just enter **?** and the name of your **function** in the console
- The help file will **pop up in the Help tab** of RStudio

```
1  ?mutate
```

Search on the internet/use AI!

- Your question is for sure already asked and answered on Stack Overflow
- Be careful about AI code, often unnecessarily complicated

When it doesn't work at all

Sometimes R breaks and returns an **error** (usually kind of cryptic)

```
1  read.csv("C:\Users\Documents\R")
```

```
Error: '\U' used without hex digits in character string (<input>:1:14)
```

What to do:

1. Look for **keywords** that might help you understand where it comes from
2. Paste it on **Google** with the name of your command

Today's lecture

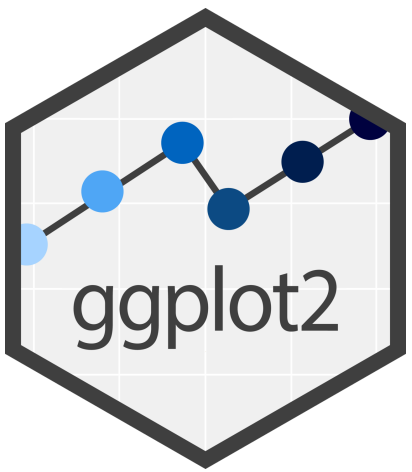
1. Summarizing data ✓
    - 1.1 Distributions ✓
    - 1.2 Central tendency ✓
    - 1.3 Spread ✓
    - 1.4 Relationship between variables ✓
2. Manipulating data ✓
    - 2.1 `dplyr` verbs ✓
    - 2.2 `group_by` ✓
    - 2.3 Joining data ✓
3. **Visualizing data**
    - 3.1 gg is for Grammar of Graphics
    - 3.2 Different types of gaphs

Visualizing data

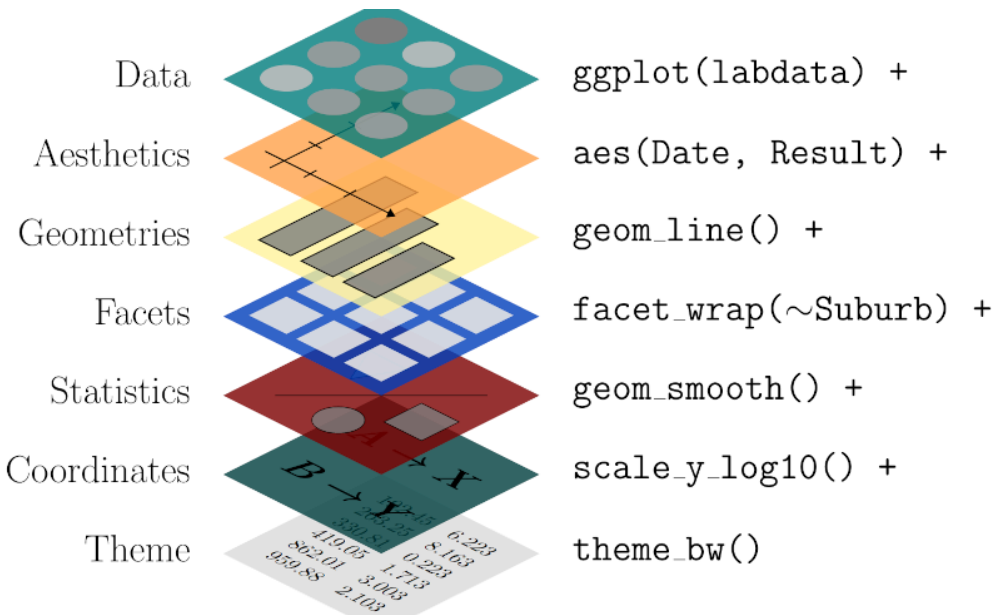
The `ggplot2` package

`ggplot2` is a **grammar of graphics** providing a very **powerful and user-friendly** visualization tool.

We used it in lecture 1, without going through the different elements



Grammar of gaphics



Let's use the global working hours since 1900 data.

gg is for Grammar of Graphics

Data

```
1 data |> ggplot()
```

or

```
1 ggplot(data)
```

Tidy Data

1. Each variable forms a **column**
2. Each observation forms a **row**
3. Each observational unit forms a table

Start by asking

1. What information do I want to use in my visualization?
2. Is that data contained in **one column/row** for a given data point?

gg is for Grammar of Graphics

Data

Aesthetics

```
1  + aes()
```

Map data to visual elements or parameters

- year
- hours worked
- country

gg is for Grammar of Graphics

Data

Aesthetics

```
1  + aes()
```

Map data to visual elements or parameters

- year → **x**
- hours worked → **y**
- country → **shape, color, etc.**

```
1  aes(x = year,
2      y = hours_worked,
3      color = country)
```

gg is for Grammar of Graphics

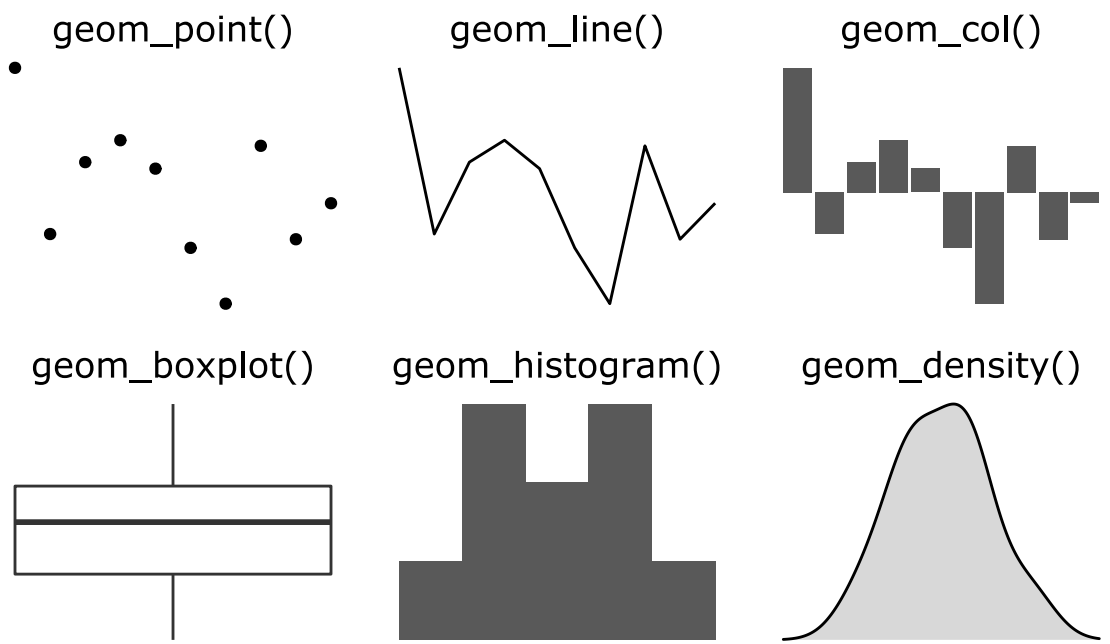
Data

Aesthetics

Geoms

```
1  + geom_*()
```

Geometric objects displayed on the plot



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

```
1  + geom_*()
```

Just start typing `geom_` in `RStudio` to see all the options

```
ggplot(df_geom) +
  aes(x, y) +
```

An example

```
1  global_working_hours_panel
```

```
# A tibble: 2,565 × 9
   country          region  year hours_wo
hours_worked_women
   <chr>            <chr>  <int>        <
<dbl>
 1 United Arab Em… Middl…  2017
32.0
 2 United Arab Em… Middl…  2018
26.4
 3 United Arab Em… Middl…  2019
28.5
 4 United Arab Em… Middl…  2022
25.8
 5 United Arab Em… Middl…  2023
25.3
 6 Afghanistan     Middl…  2007
13.1
 7 Afghanistan     Middl…  2008
12.8
```

An example

```
1  global_working_hours_panel |>
2    filter(country == "France")
```

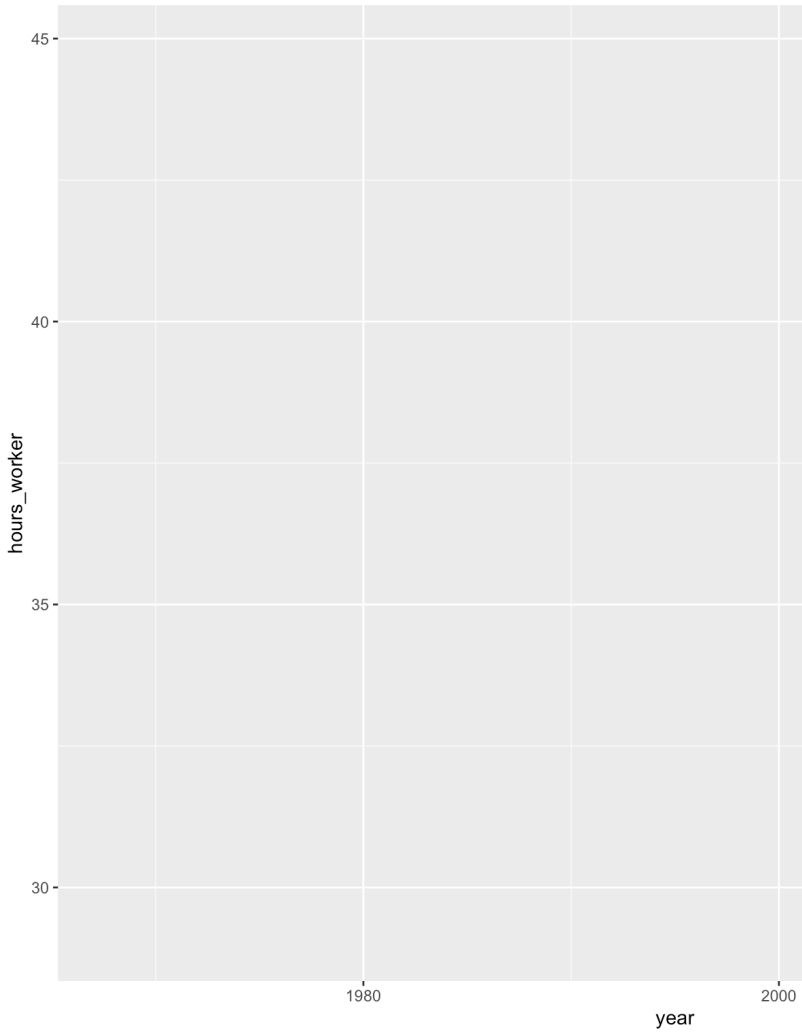
```
# A tibble: 55 × 9
   country region          year hours_wo
hours_worked_women
   <chr>   <chr>          <int>        <
<dbl>
 1 France  Western Europ…  1968
15.3
 2 France  Western Europ…  1969
15.1
 3 France  Western Europ…  1970
14.8
 4 France  Western Europ…  1971
14.5
 5 France  Western Europ…  1972
14.8
 6 France  Western Europ…  1973
14.9
 7 France  Western Europ…  1974
15.0
```

An example

```
1  global_working_hours_panel |>
2    filter(country == "France") |>
3    ggplot()
```

An example

```
1  global_working_hours_panel |>
2    filter(country == "France") |>
3    ggplot(aes(x = year, y = hours_worker))
```



An example

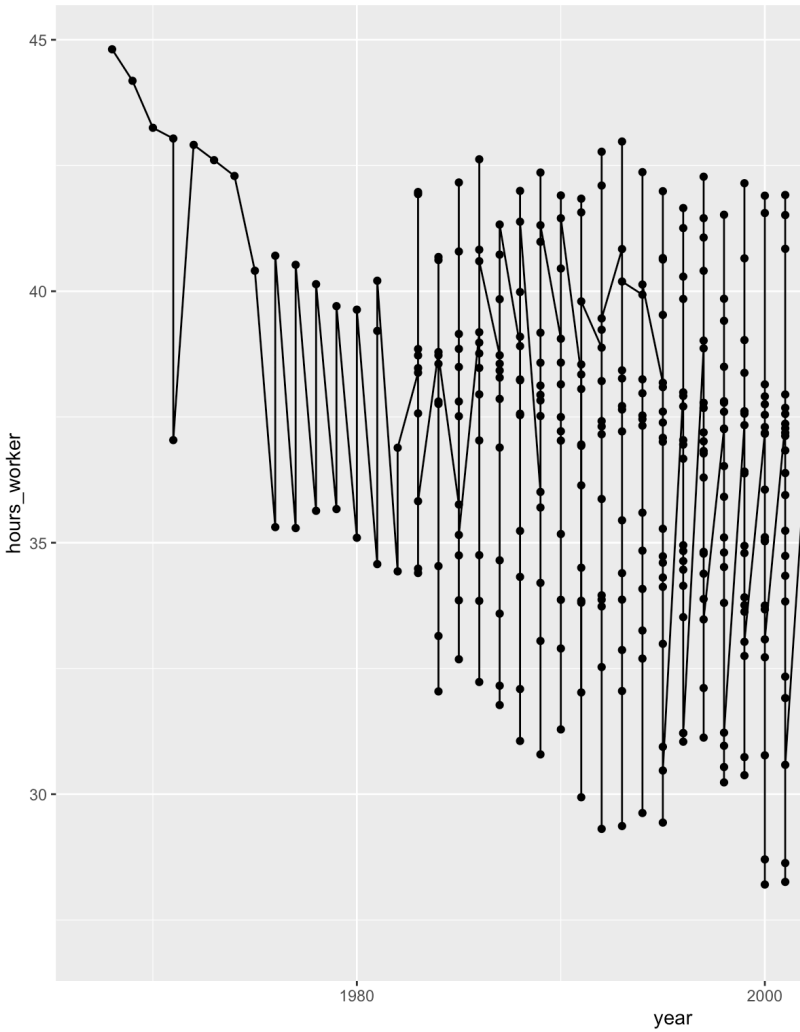
```
1  global_working_hours_panel |>
2    filter(country == "France") |>
3    ggplot(aes(x = year, y = hours_worker)) +
4    geom_line()
```

An example

```
1  global_working_hours_panel |>
2    filter(country == "France") |>
3    ggplot(aes(x = year, y = hours_worker)) +
4    geom_line() +
5    geom_point()
```

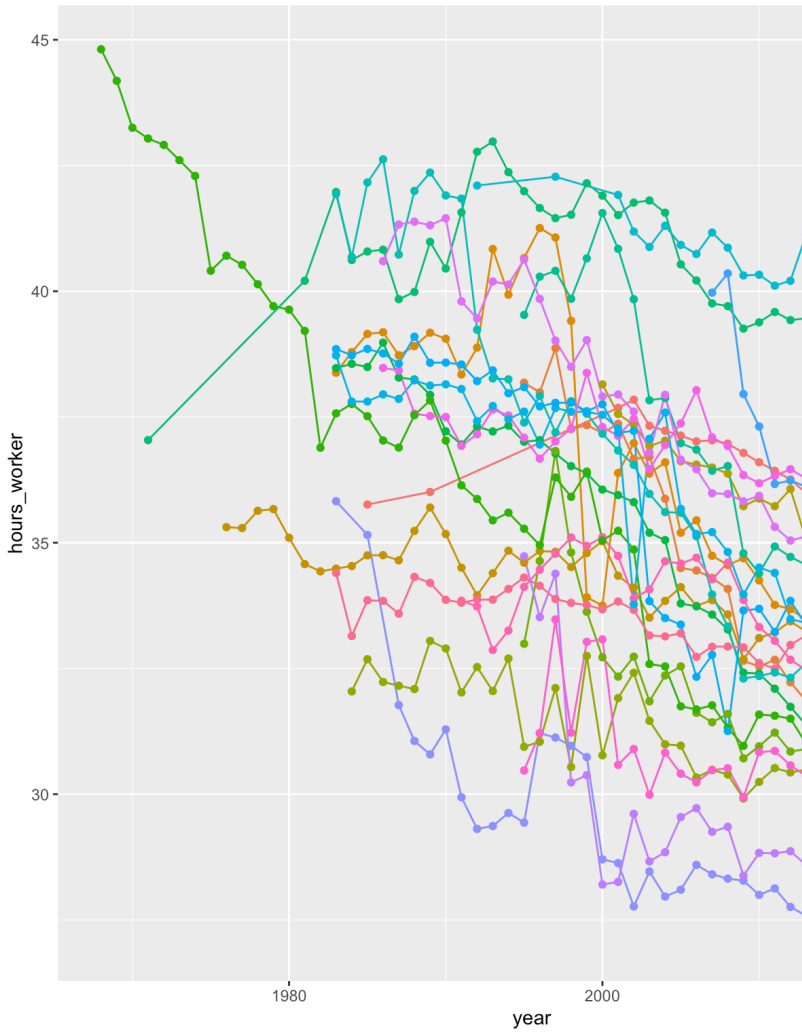
An example

```
1  global_working_hours_panel |>
2    filter(region == "Western Europe and Anglosphere") |>
3    ggplot(aes(x = year, y = hours_worker)) +
4    geom_line() +
5    geom_point()
```



An example

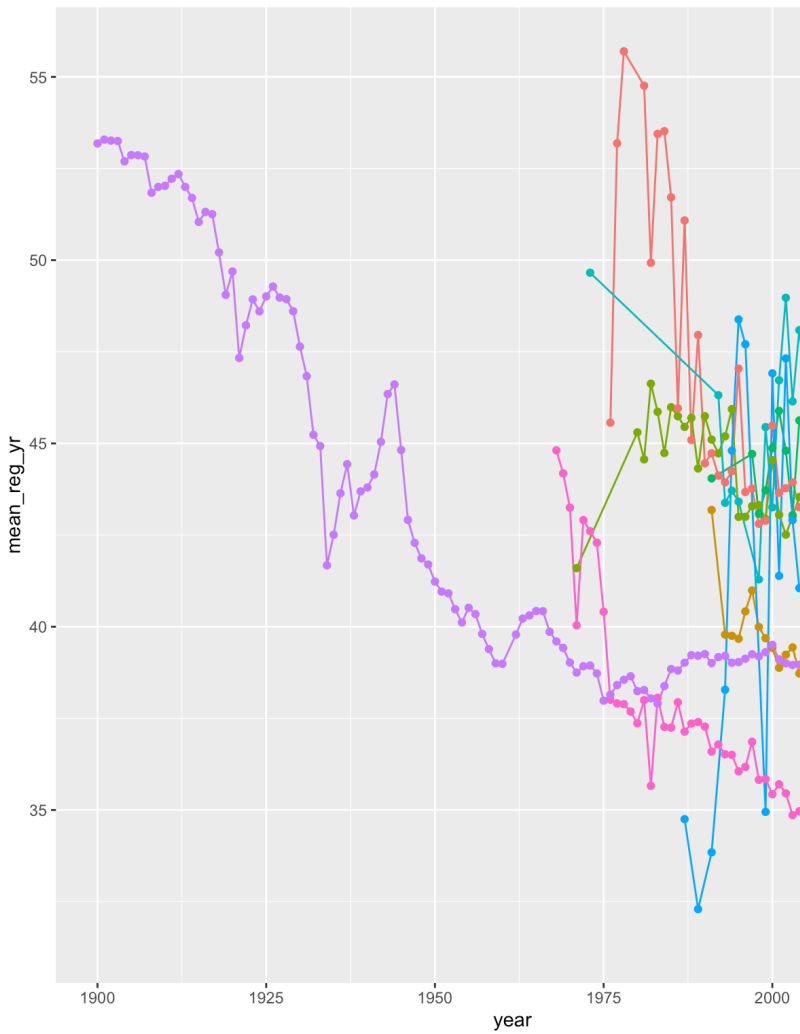
```
1  global_working_hours_panel |>
2    filter(region == "Western Europe and Anglosphere") |>
3    ggplot(aes(x = year, y = hours_worker, color = country)) +
4    geom_line() +
5    geom_point()
```



An example

```
plot = global_working_hours_panel |>
  summarise(mean_reg_yr = mean(hours_worker),
            .by = c(region, year)) |>
  ggplot(aes(x = year, y = mean_reg_yr, color = region)) +
  geom_line() +
  geom_point()

plot # graphs can be saved as objects!
```



gg is for Grammar of Graphics

Data

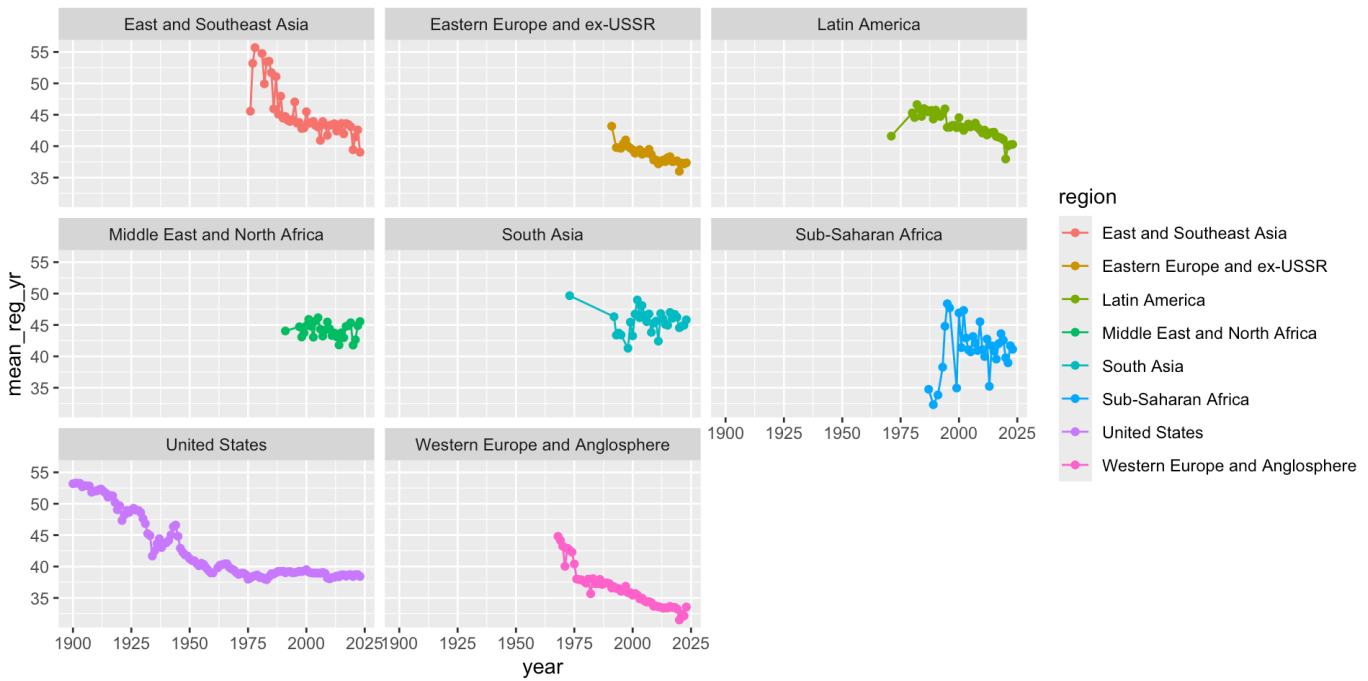
Aesthetics

Geoms

Facet

```
1  + facet_wrap()
2  + facet_grid()
```

```
1  plot +
2    facet_wrap(~ region)
```



gg is for Grammar of Graphics

Data

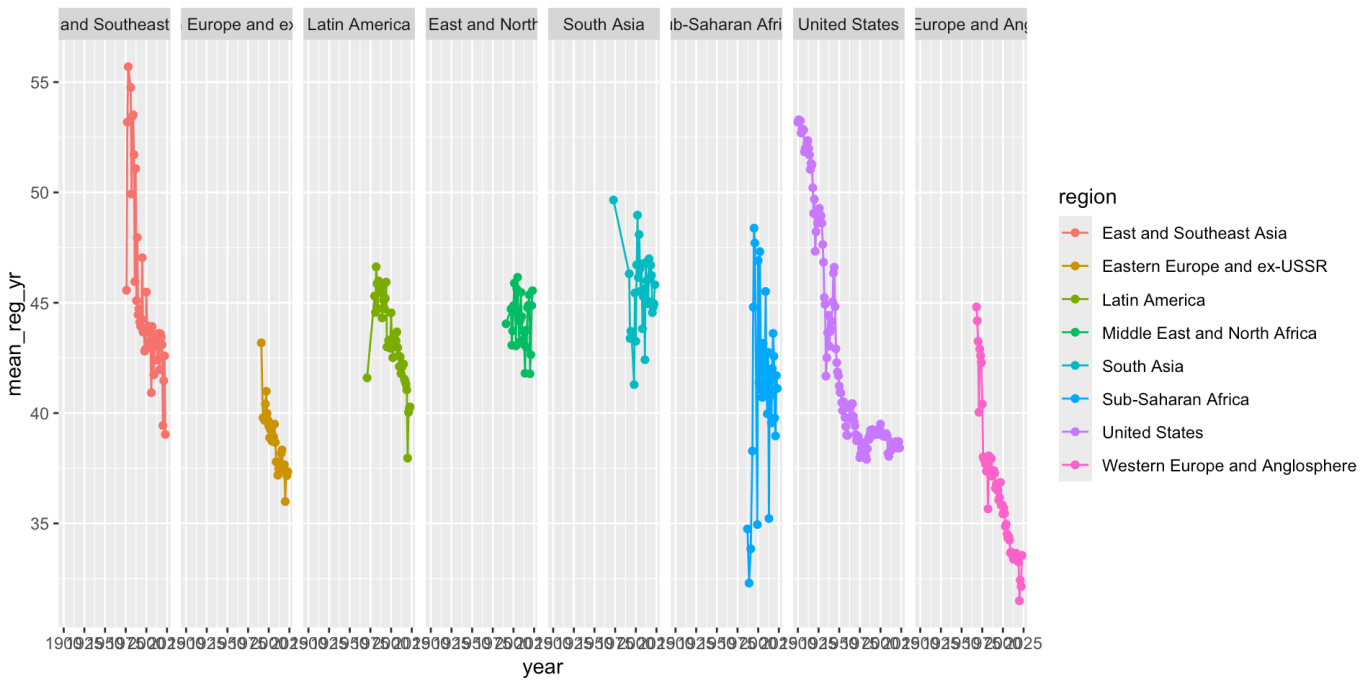
Aesthetics

Geoms

Facet

```
1  + facet_wrap()
2  + facet_grid()
```

```
1  plot +
2    facet_grid(~ region)
```



gg is for Grammar of Graphics

Data

Aesthetics

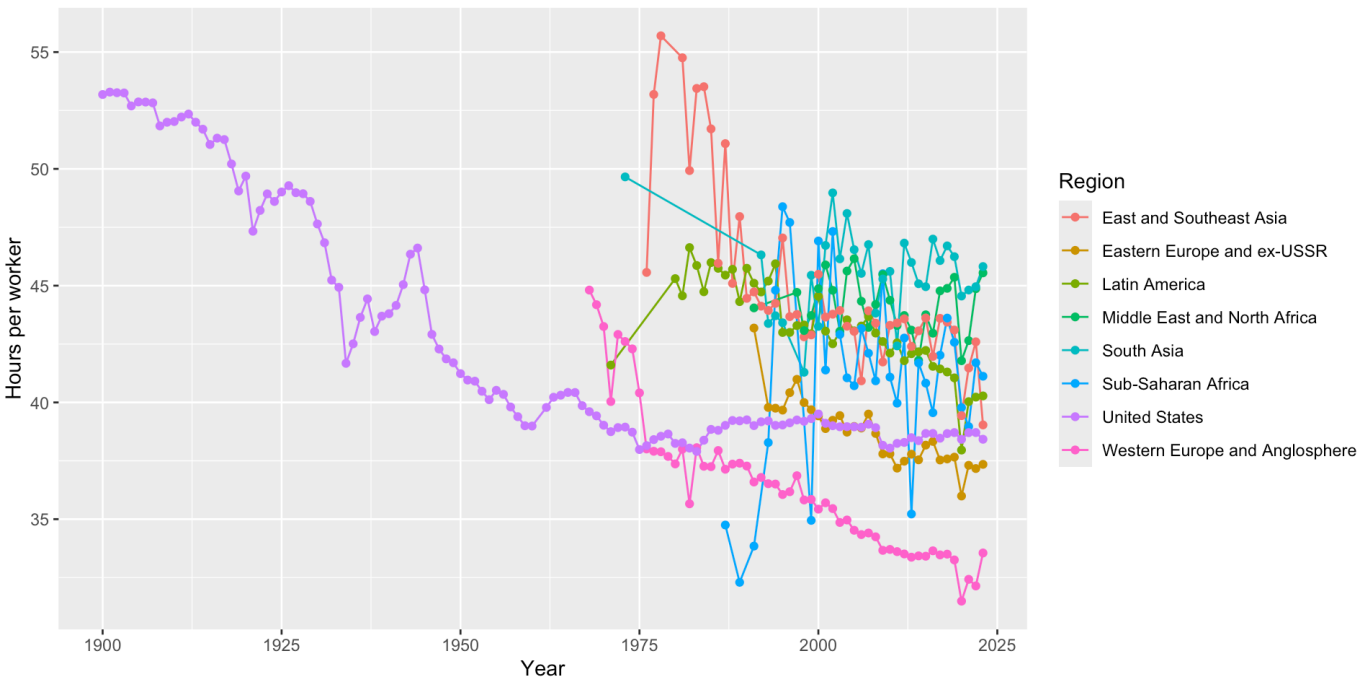
Geoms

Facet

Labels

```
1  + labs
```

```
1  plot +
2    labs(x = "Year",
3         y = "Hours per worker",
4         color = "Region")
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

Facet

Labels

Scales

```
1  + scale_*_*()
```

`scale` + `_` + `<aes>` + `_` + `<type>` + `()`

What parameter do you want to adjust? → `<aes>`

What type is the parameter? → `<type>`

- I want to change my discrete x-axis → `scale_x_discrete()`
- I want to change range of point sizes from continuous variable → `scale_size_continuous()`
- I want to rescale y-axis as log10 → `scale_y_log10()`
- I want to use a different color palette → `scale_fill_discrete()` / `scale_color_manual()`

gg is for Grammar of Graphics

Data

Aesthetics

Geoms

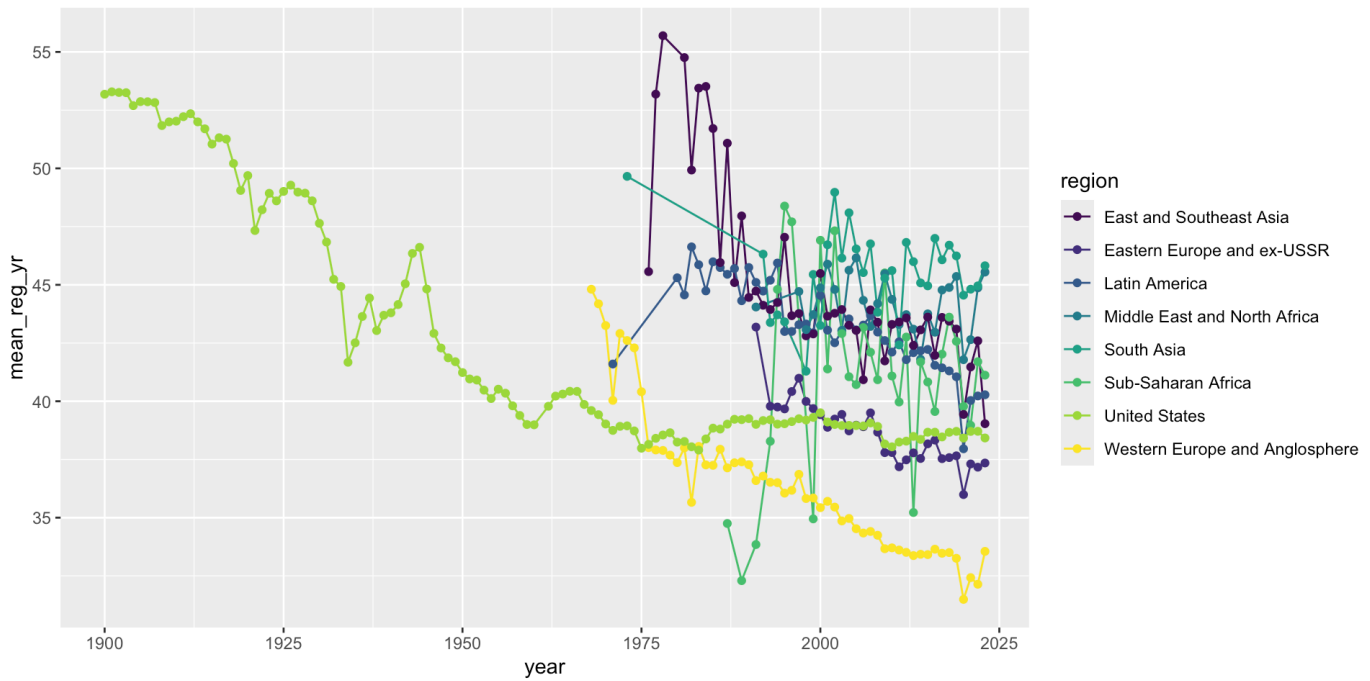
Facet

Labels

Scales

```
1  + scale_*_*()
```

```
1  plot +
2    scale_color_viridis_d()
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

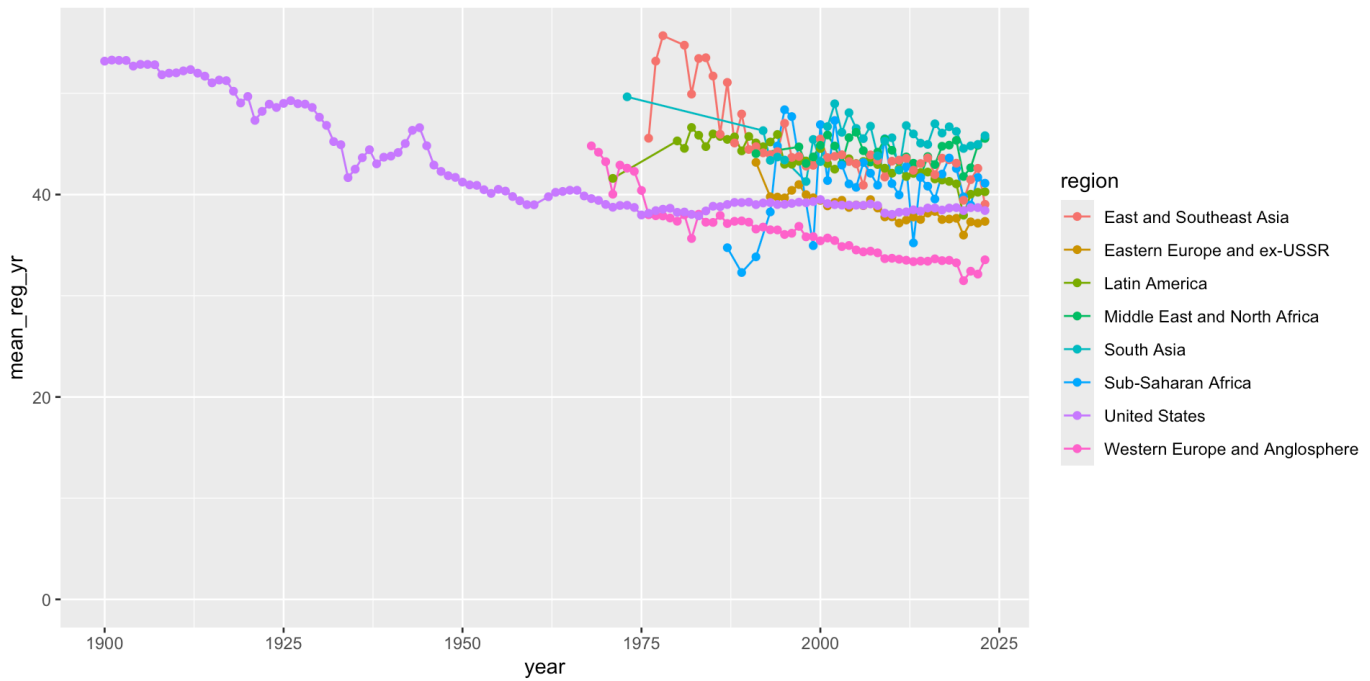
Facet

Labels

Scales

```
1  + scale_*_*()
```

```
1  plot +
2    scale_y_continuous(limits = c(0,NA))
```



gg is for Grammar of Graphics

Data

Aesthetics

Geoms

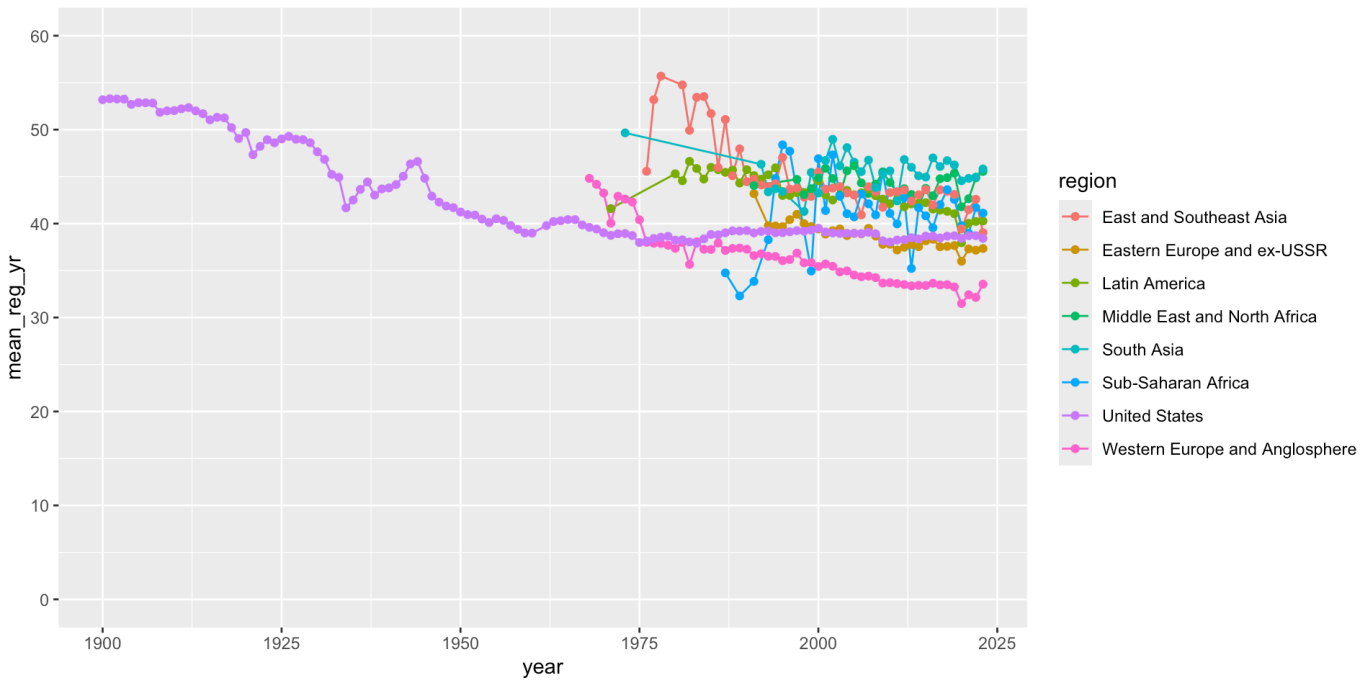
Facet

Labels

Scales

```
1  + scale_*_*()
```

```
1  plot +
2    scale_y_continuous(limits = c(0,60), breaks = seq(0, 60, 10))
```



Devling deeper into `ggplot2`

- Each graph is different and `ggplot2` provides a zillion options to customize your graph to perfection.
- Excellent cheatsheet on project website.
- Cédric Scherer's wonderful *Graphic Design with ggplot2* tutorial

Today's lecture

1. Summarizing data ✓
    - 1.1 Distributions ✓
    - 1.2 Central tendency ✓
    - 1.3 Spread ✓
    - 1.4 Relationship between variables ✓
2. Manipulating data ✓
    - 2.1 `dplyr` verbs ✓
    - 2.2 `group_by` ✓
    - 2.3 Joining data ✓
3. **Visualizing data**
    - 3.1 gg is for Grammar of Graphics ✓
    - 3.2 Different types of gaphs

Different types of graphs

The type of graph you need depends on whether:

1. you are plotting **one variable** or the **relationship between variables**

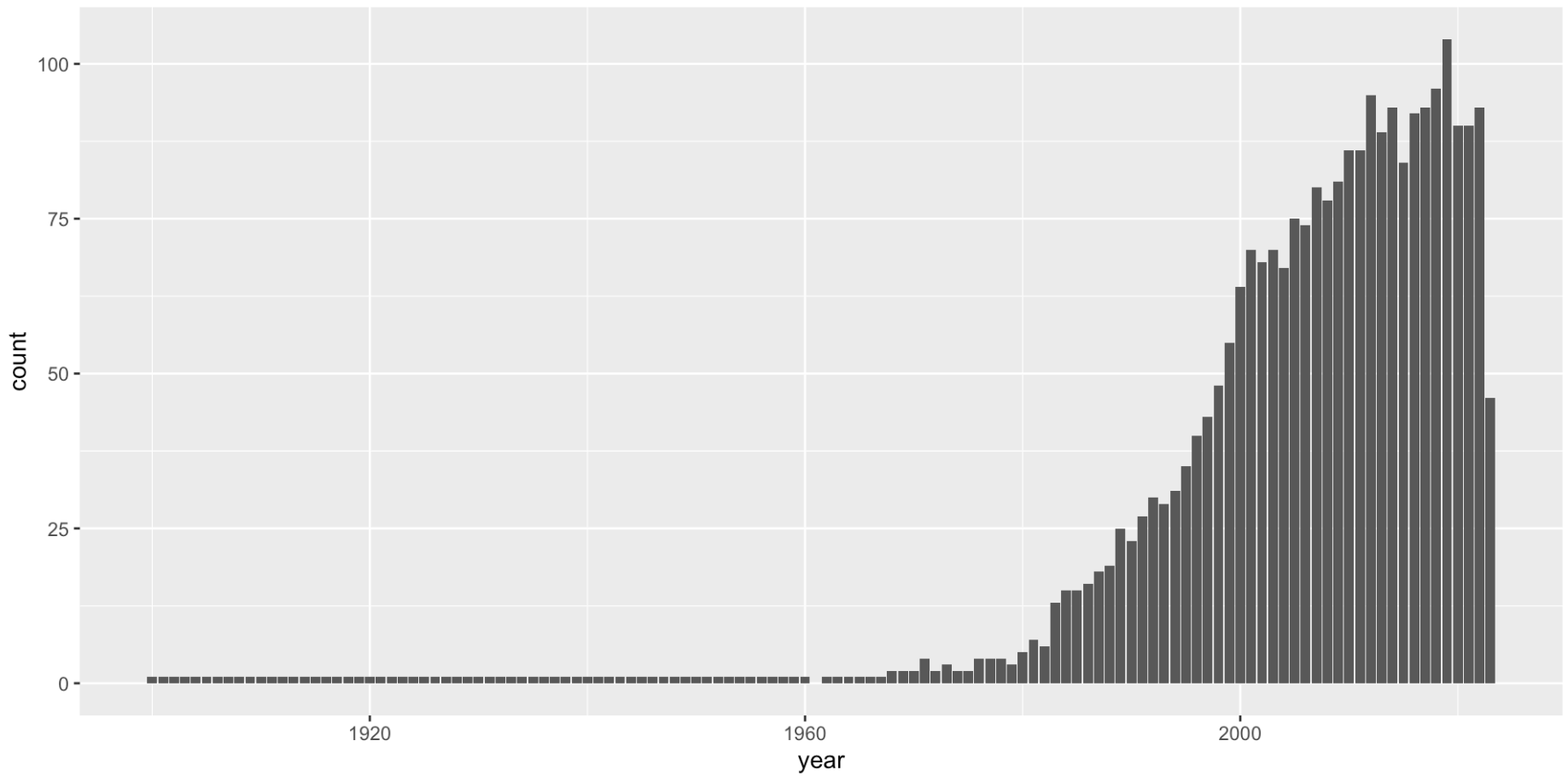
2. the variable(s) are **categorical** or **numerical**.

Let's start with plotting just **one variable**.

Visualizing distributions: frequency plot

How many countries per year?

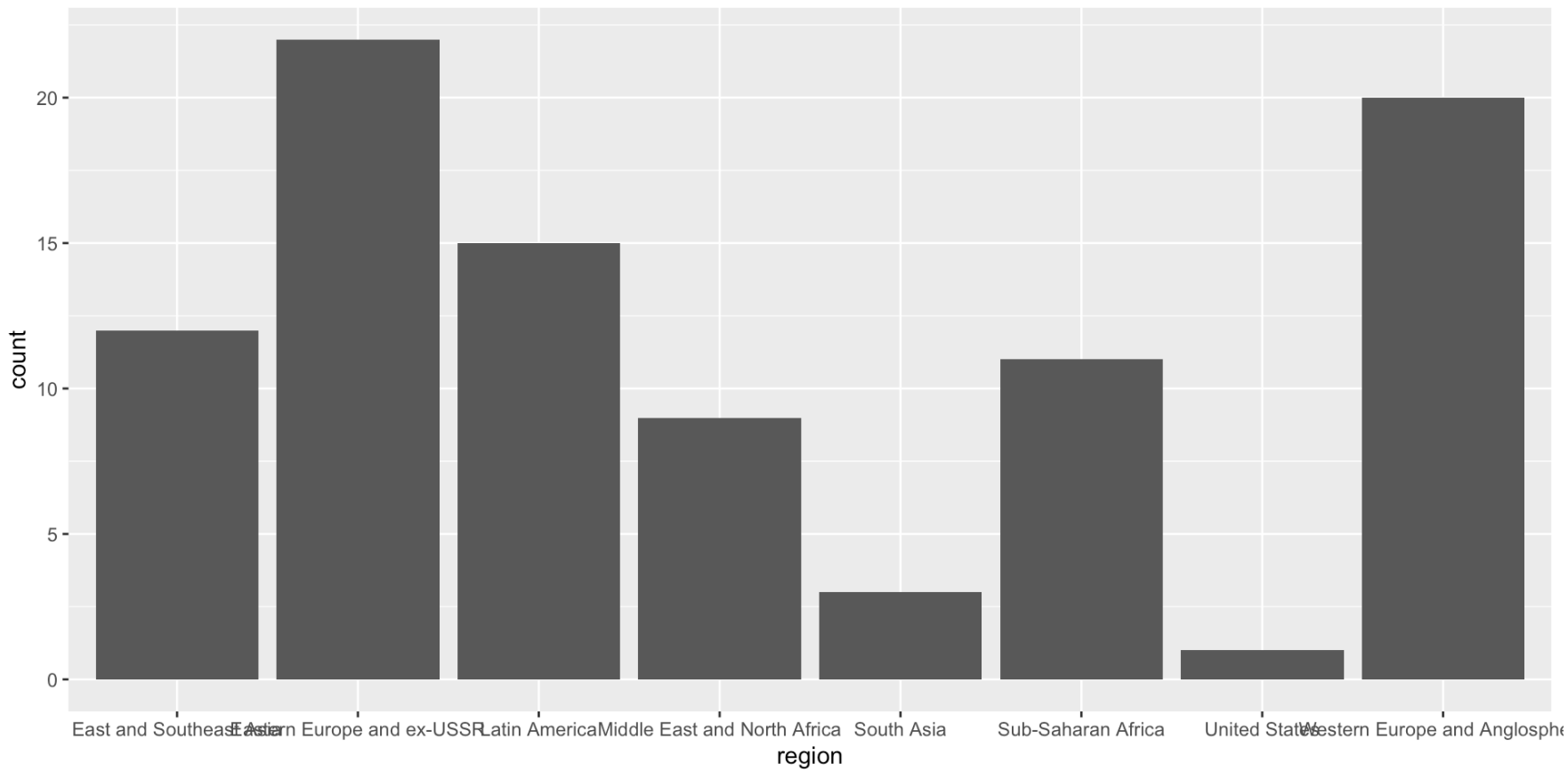
```
1  global_working_hours |>
2    ggplot(aes(x = year)) +
3    geom_bar()
```



Visualizing distributions: frequency plot

How many countries per region?

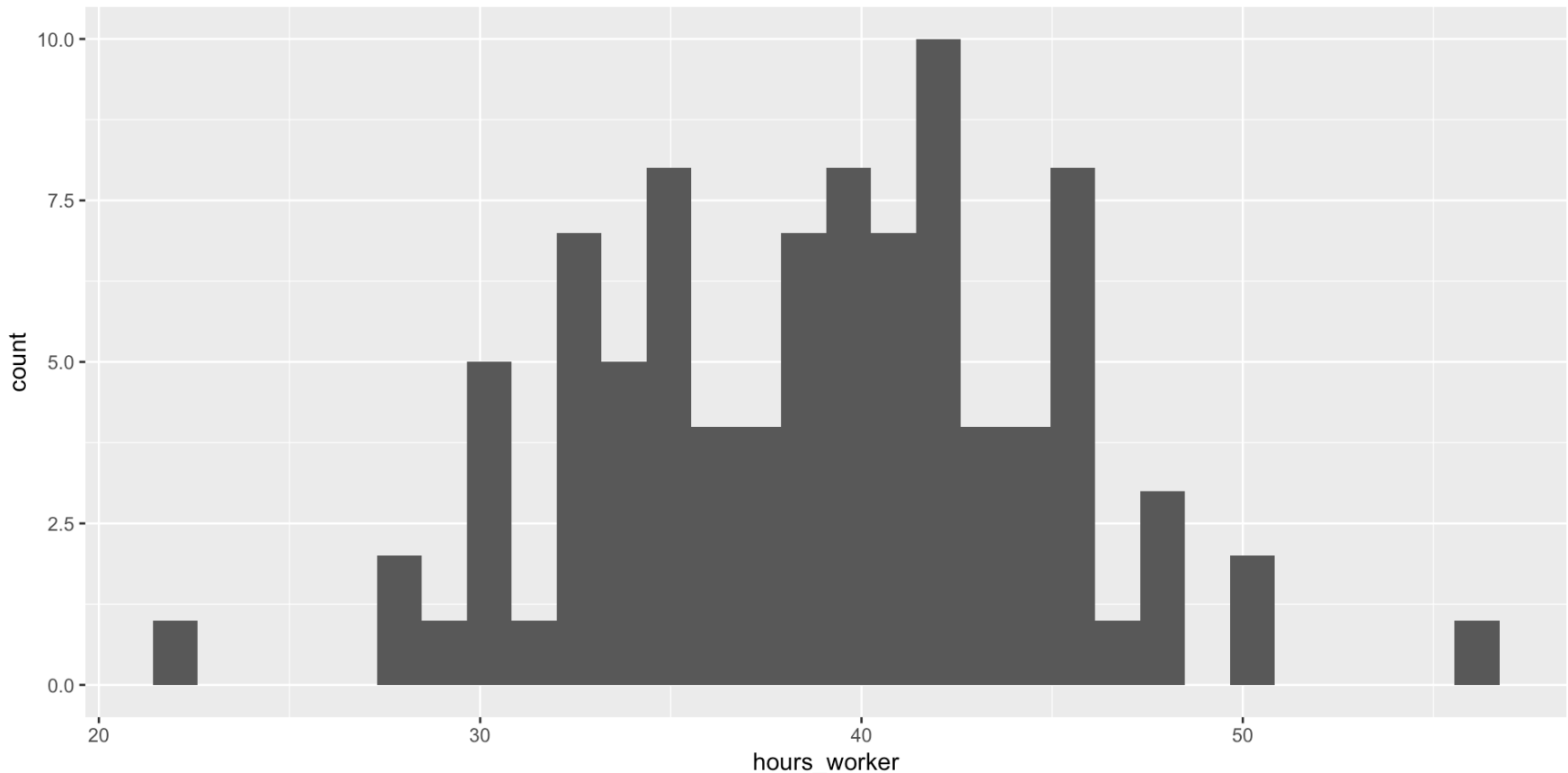
```
1  global_working_hours |>
2    filter(year == 2022) |>
3    ggplot(aes(x = region)) +
4    geom_bar()
```



Visualizing distributions: histogram and density plots

Histogram of hours per worker in 2022

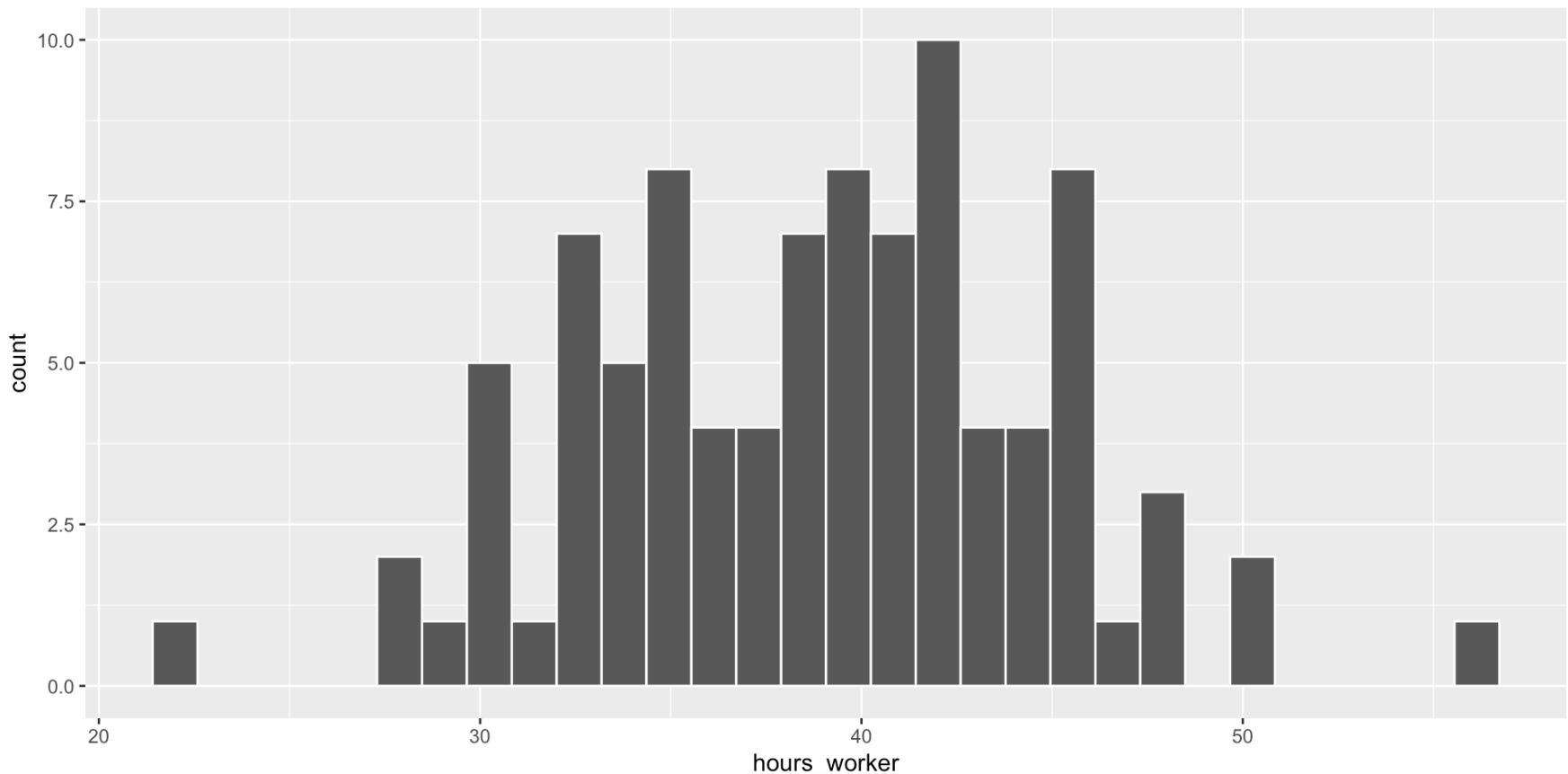
```
1  global_working_hours |>
2    filter(year == 2022) |>
3    ggplot(aes(x = hours_worker)) +
4    geom_histogram()
```



Visualizing distributions: histogram and density plots

Histogram of hours per worker in 2022

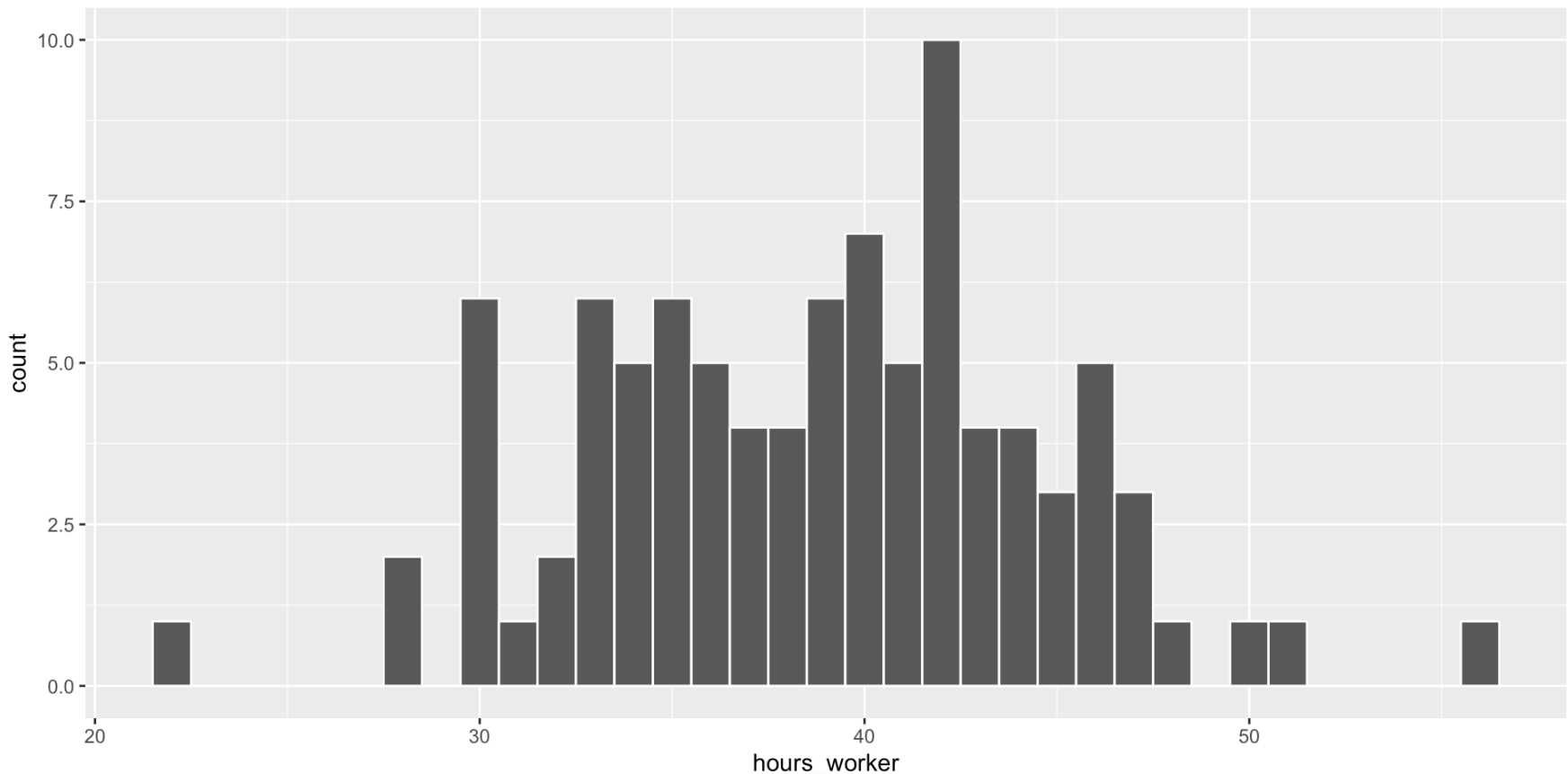
```
1  global_working_hours |>
2    filter(year == 2022) |>
3    ggplot(aes(x = hours_worker)) +
4    geom_histogram(color = "white")
```



Visualizing distributions: histogram and density plots

Histogram of hours per worker in 2022

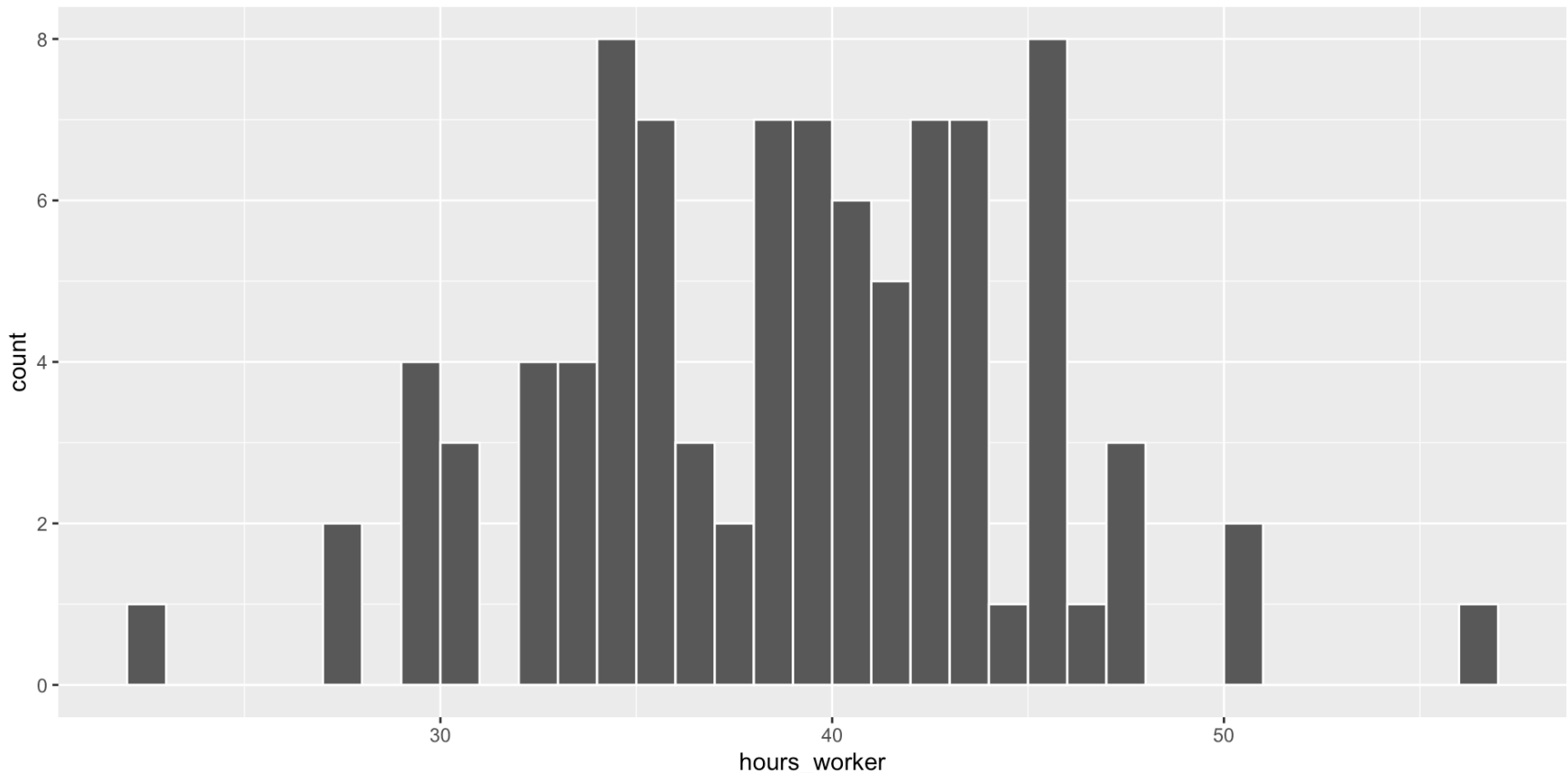
```
1  global_working_hours |>
2    filter(year == 2022) |>
3    ggplot(aes(x = hours_worker)) +
4    geom_histogram(color = "white", binwidth = 1)
```



Visualizing distributions: histogram and density plots

Histogram of hours per worked in 2022

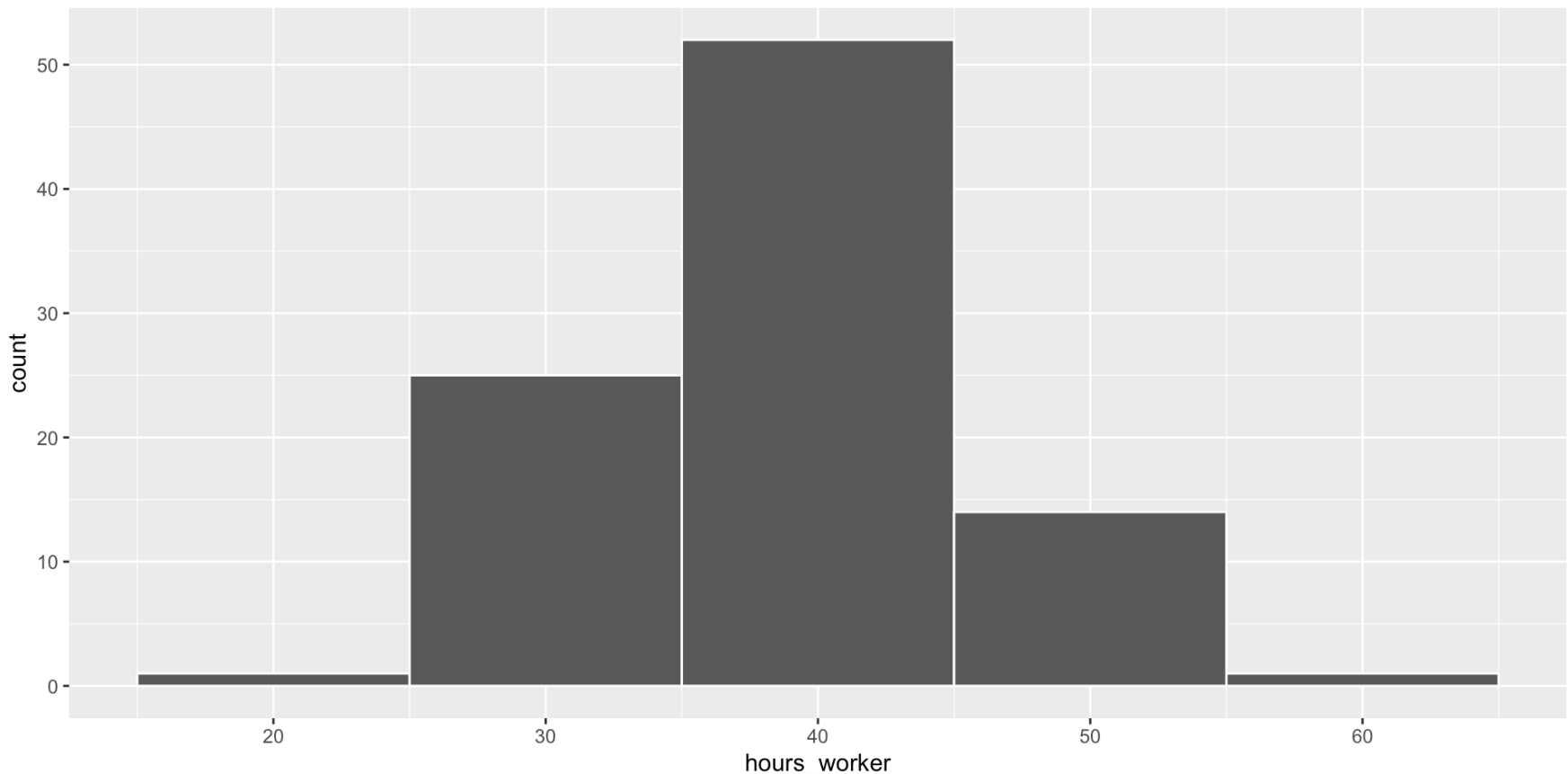
```
1  global_working_hours |>
2    filter(year == 2022) |>
3    ggplot(aes(x = hours_worker)) +
4    geom_histogram(color = "white", binwidth = 1, boundary = 35)
```



Visualizing distributions: histogram and density plots

Histogram of hours per worked in 2022

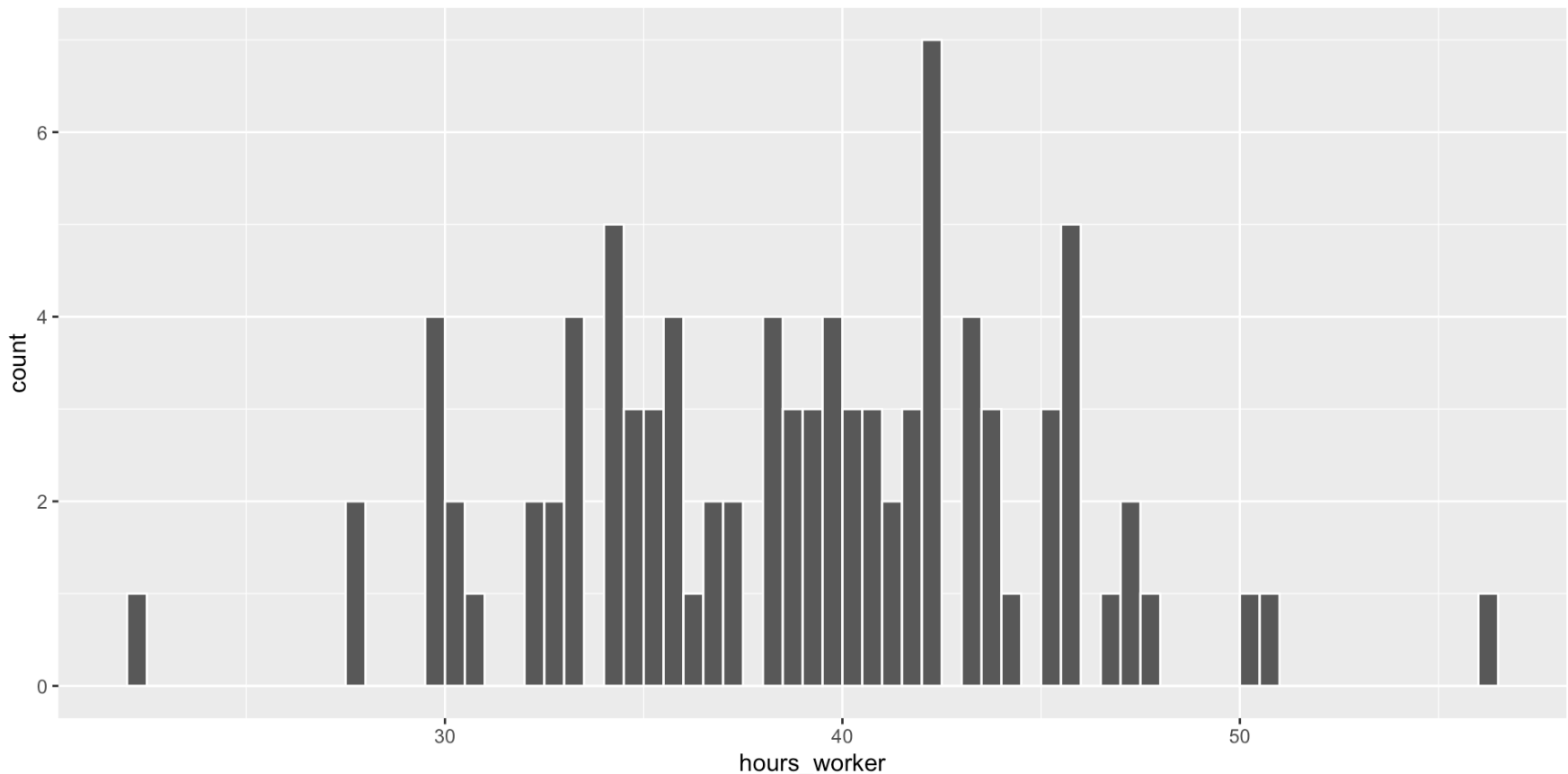
```
1 global_working_hours |>
2   filter(year == 2022) |>
3   ggplot(aes(x = hours_worker)) +
4   geom_histogram(color = "white", binwidth = 10, boundary = 35)
```



Visualizing distributions: histogram and density plots

Histogram of hours per worked in 2022

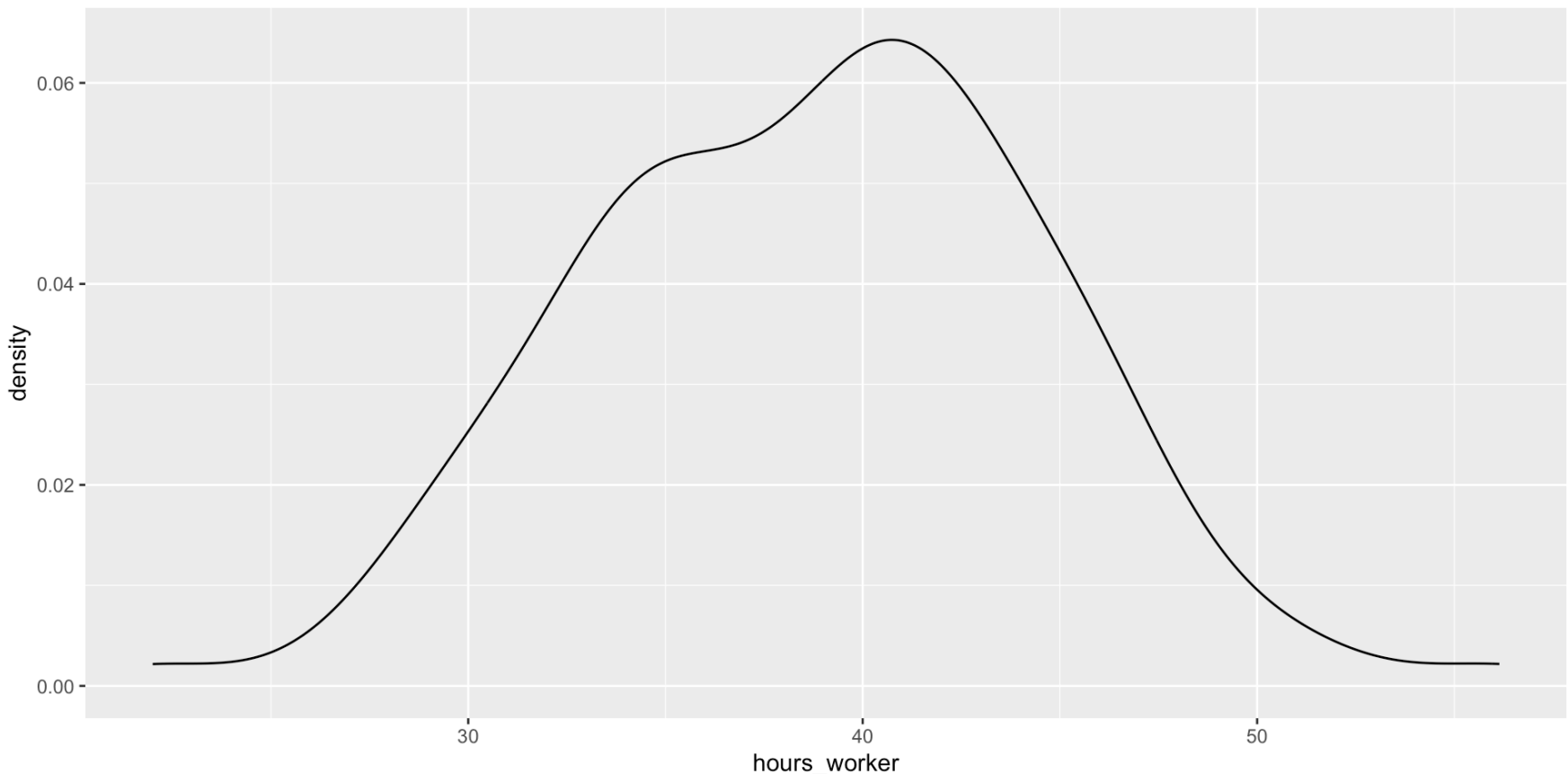
```
1  global_working_hours |>
2    filter(year == 2022) |>
3    ggplot(aes(x = hours_worker)) +
4    geom_histogram(color = "white", binwidth = .5, boundary = 35)
```



Visualizing distributions: histogram and density plots

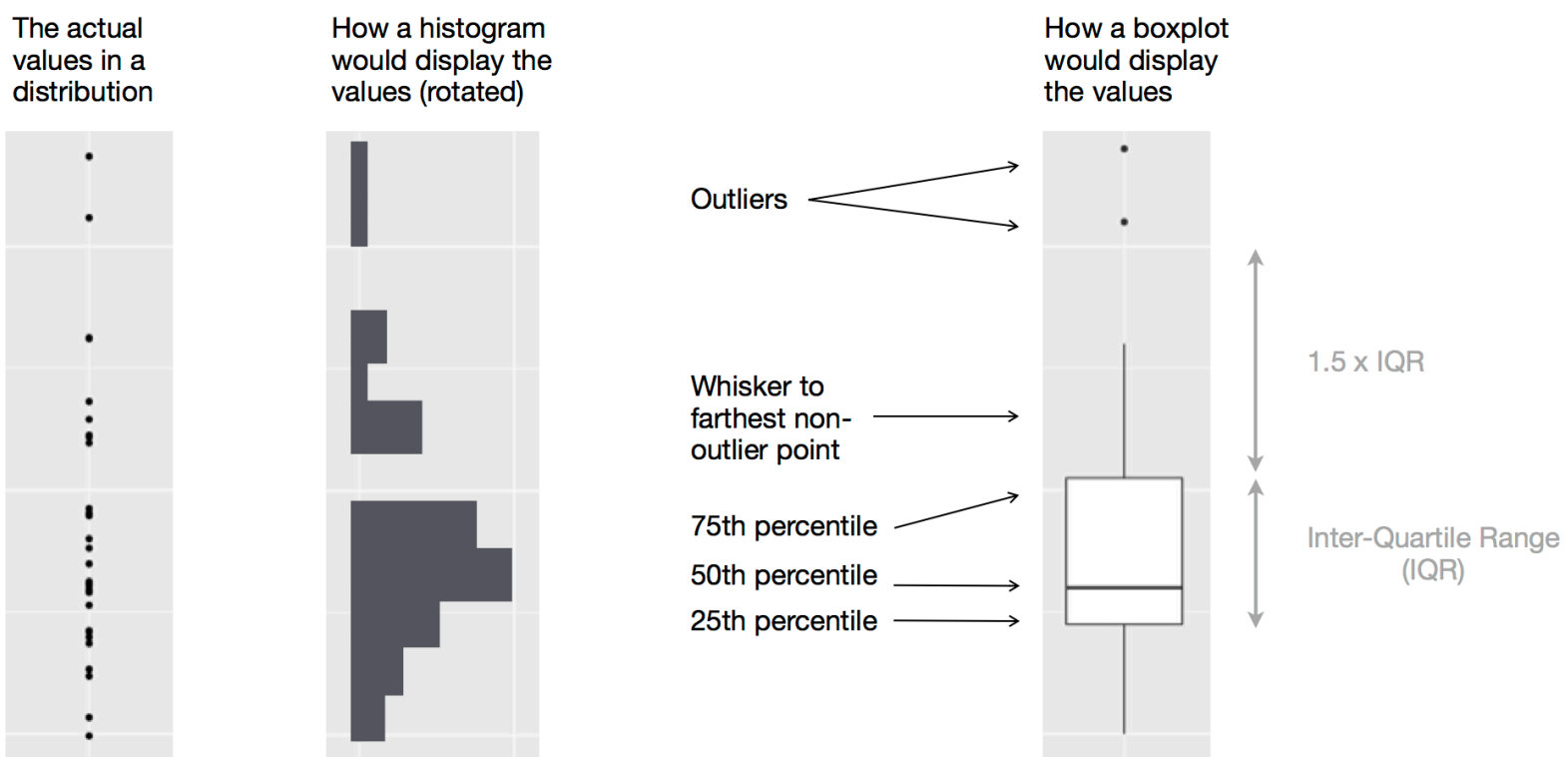
Density of hours per worker in 2022

```
1  global_working_hours |>
2    filter(year == 2022) |>
3    ggplot(aes(x = hours_worker)) +
4    geom_density()
```



Visualizing relationships: box plot

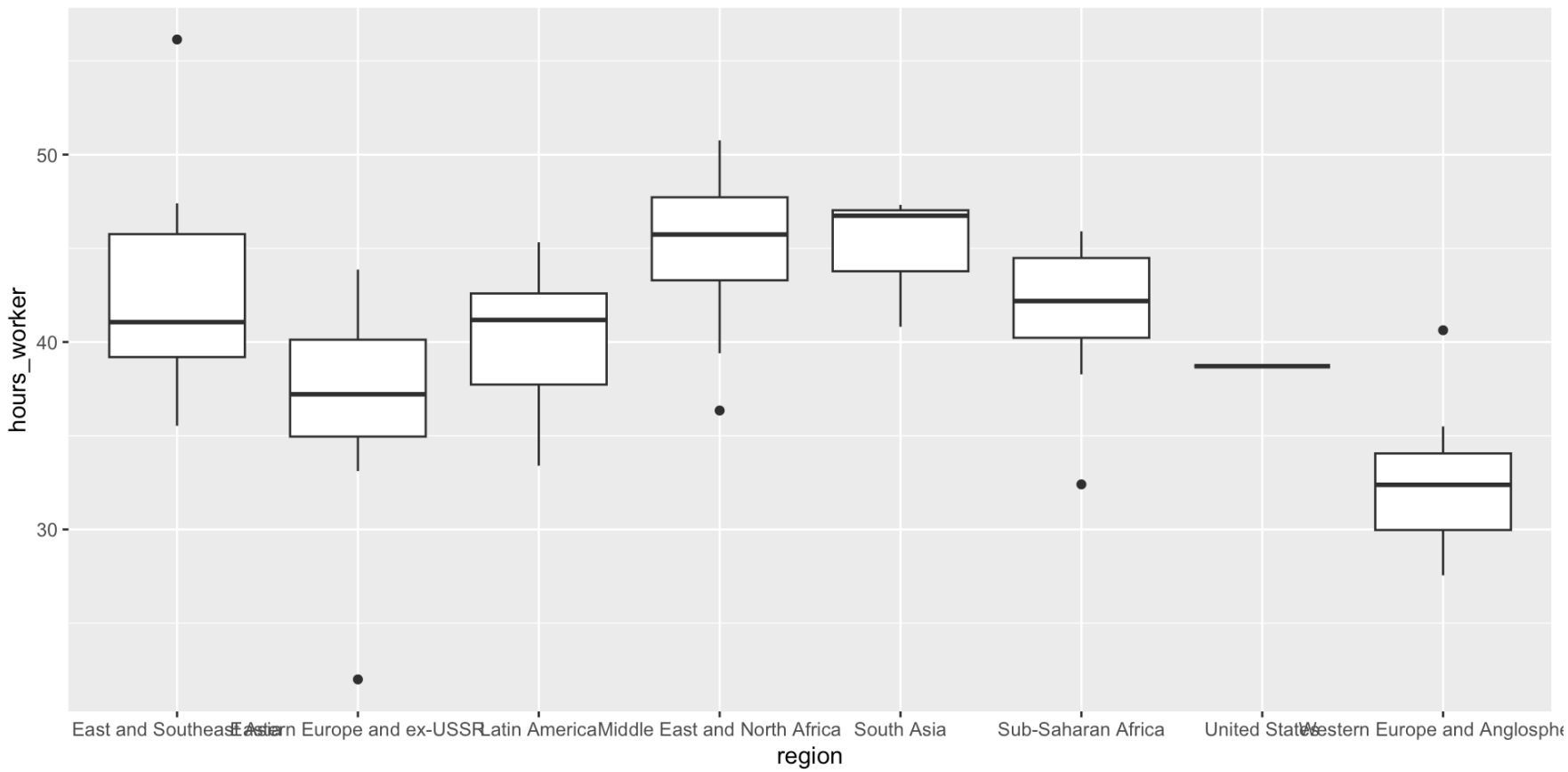
Boxplots: displays the distribution of a variable, with the median, mean and interquartile range.



Visualizing relationships: box plot

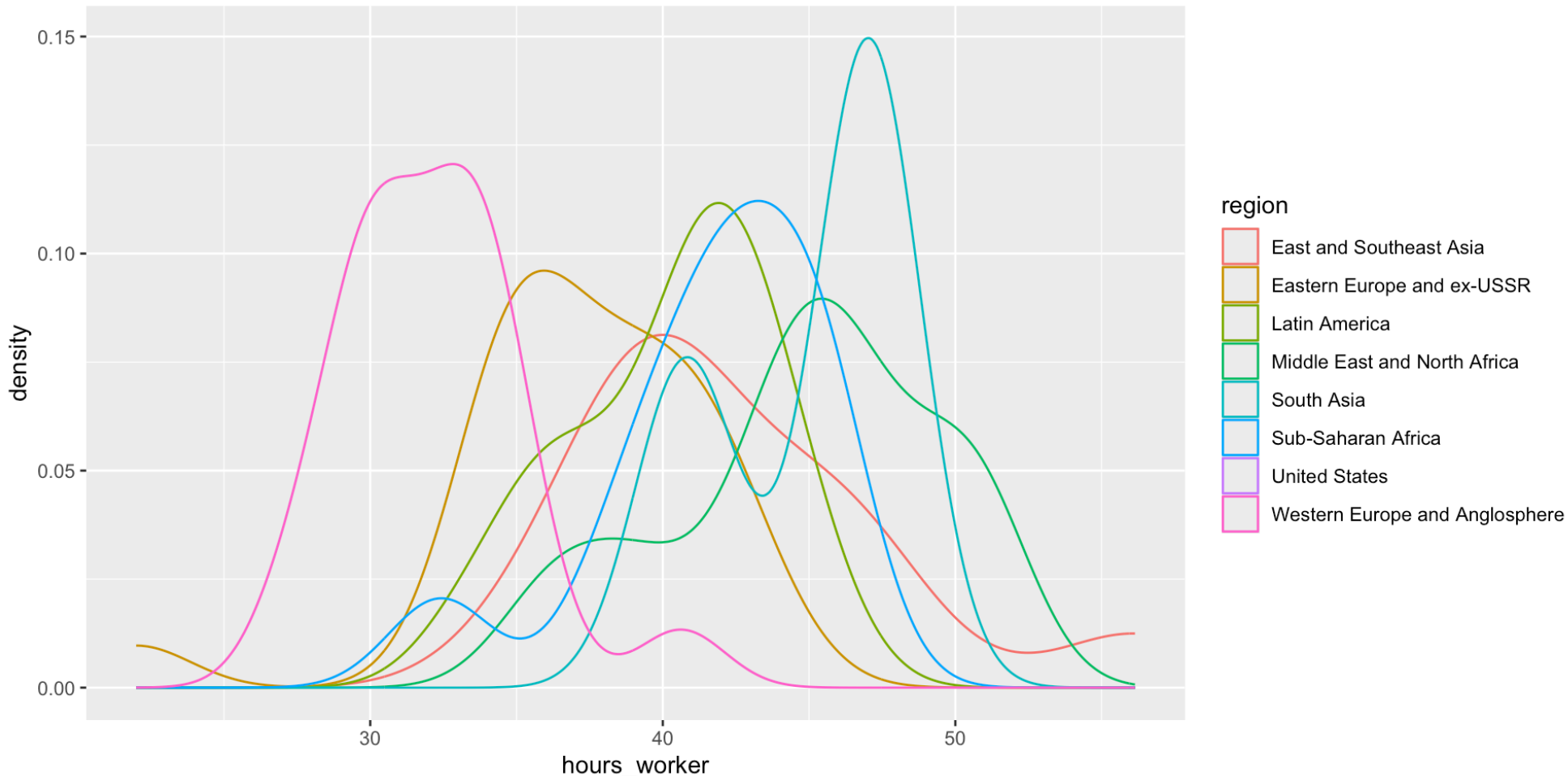
Box plot of hours per worker in 2022 by region

```
1 global_working_hours |>
2   filter(year == 2022) |>
3   ggplot(aes(x = region, y = hours_worker)) +
4   geom_boxplot()
```



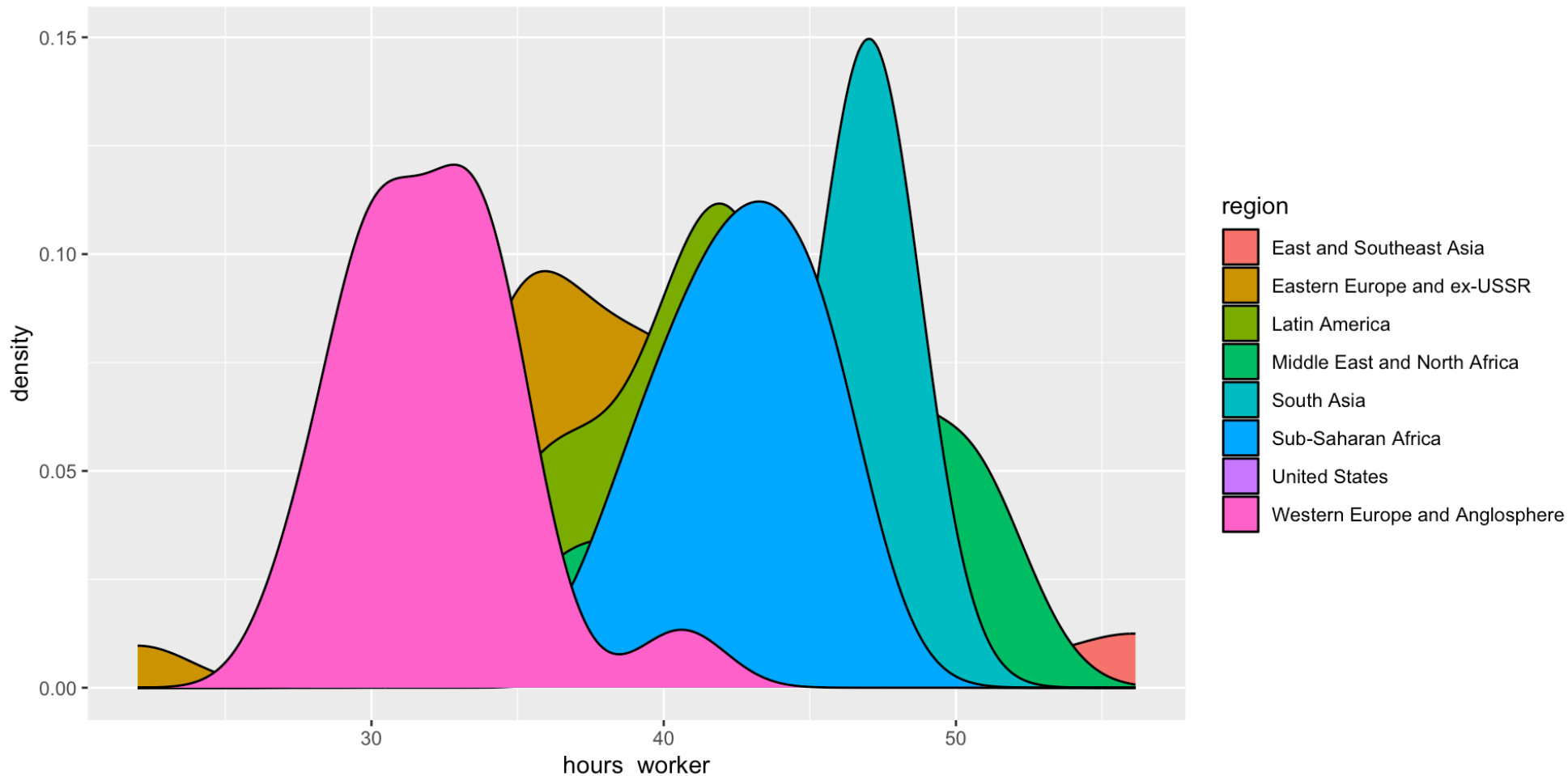
Density plot

```
1  global_working_hours |>
2    filter(year == 2022) |>
3    ggplot(aes(x = hours_worker, color = region)) +
4    geom_density()
```



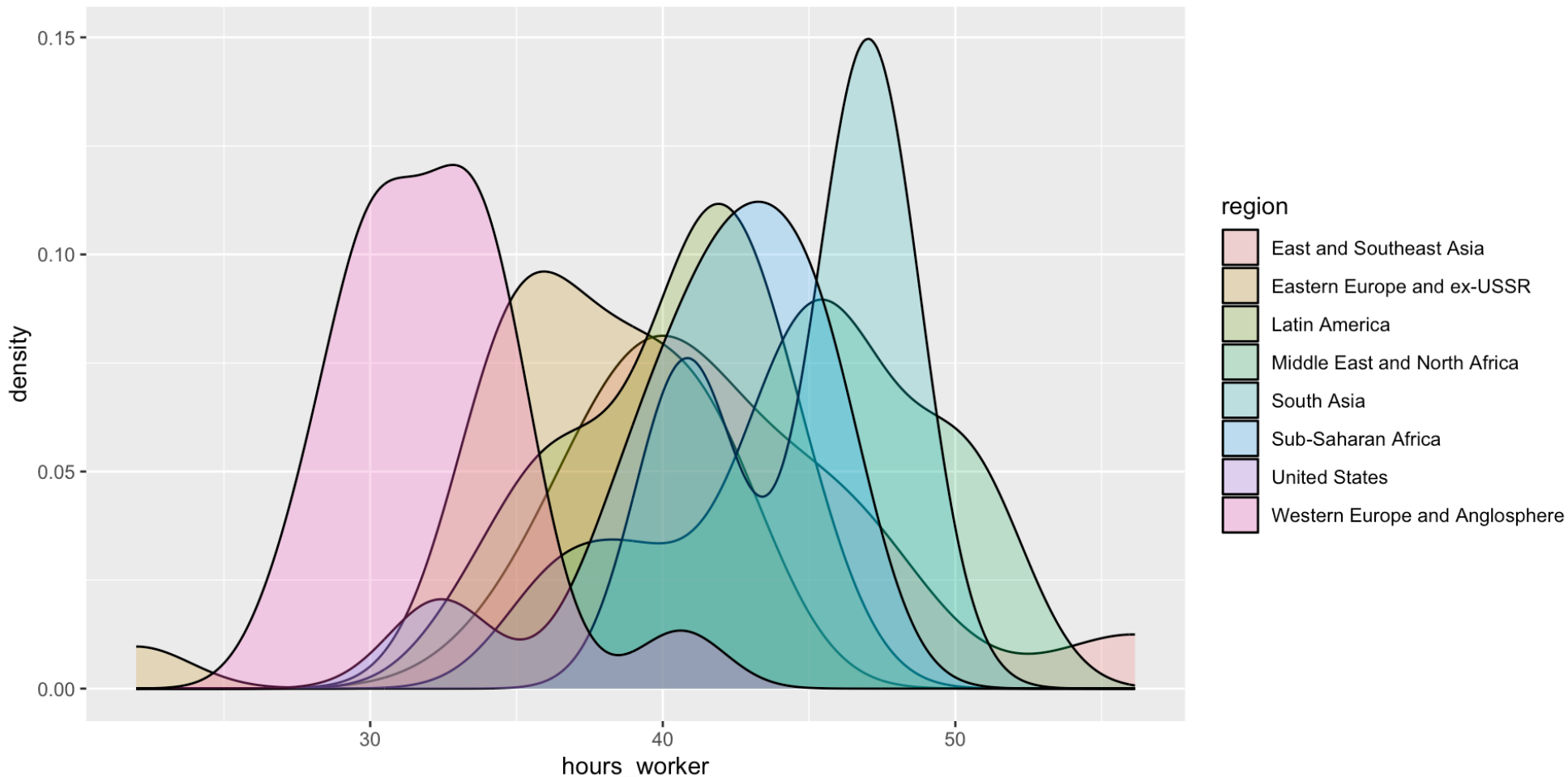
Density plot

```
1  global_working_hours |>
2    filter(year == 2022) |>
3    ggplot(aes(x = hours_worker, fill = region)) +
4    geom_density()
```



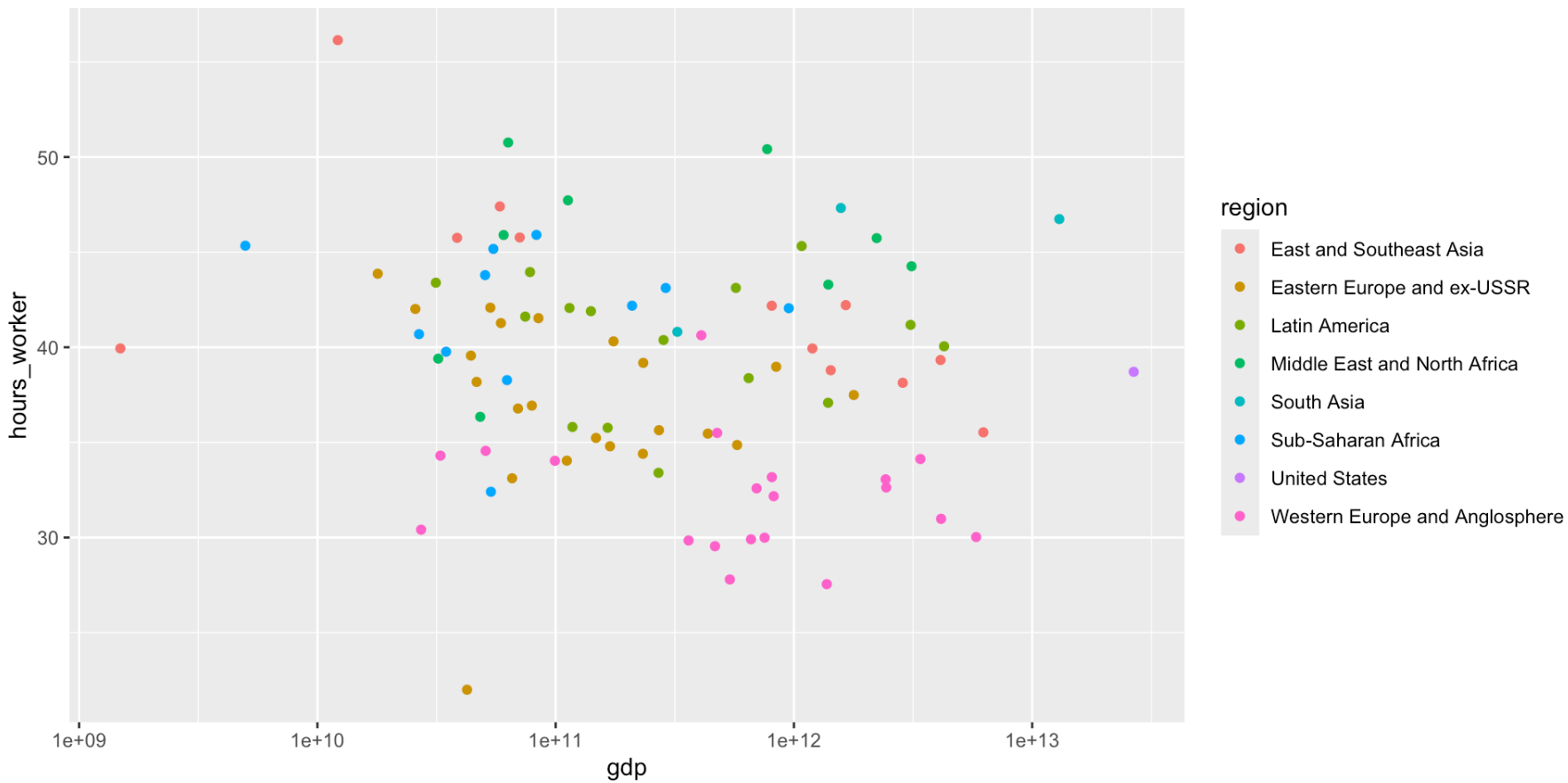
Density plot

```
1  global_working_hours |>
2    filter(year == 2022) |>
3    ggplot(aes(x = hours_worker, fill = region)) +
4    geom_density(alpha = .25)
```



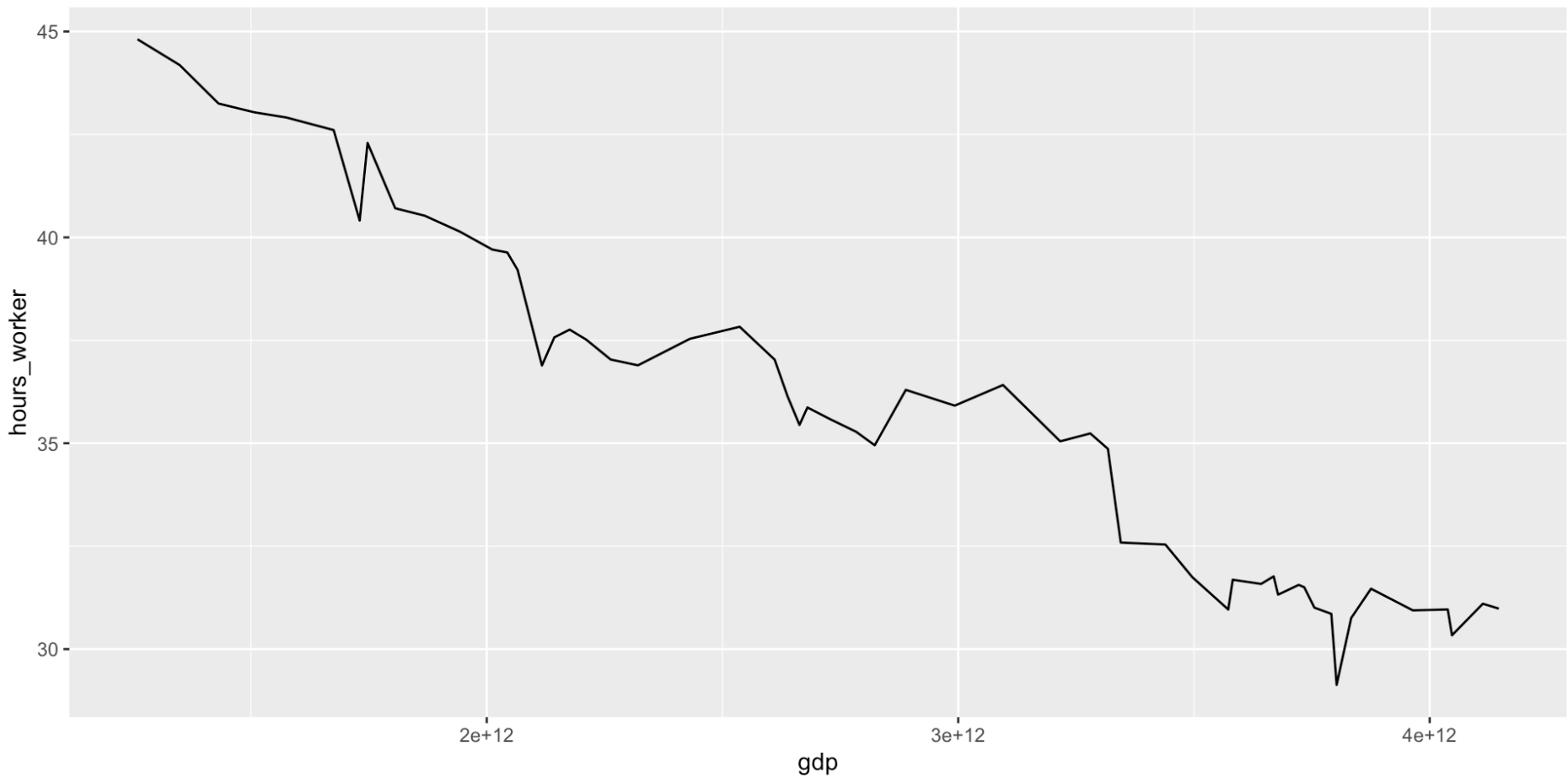
Scatter plot

```
1  global_working_hours |>
2    left_join(gdp) |>
3    filter(year == 2022) |>
4    ggplot(aes(x = gdp, y = hours_worker, color = region)) +
5    geom_point() +
6    scale_x_log10()
```



Line plot

```
1  global_working_hours |>
2    left_join(gdp) |>
3    filter(country == "France") |>
4    ggplot(aes(x = gdp, y = hours_worker)) +
5    geom_line()
```



Your turn! #3

10:00

Use the data from the previous task to produce the following plots:

1. A histogram of hours per female worker (`hours_worker_women`) in 2022. Once you've created the histogram, within the appropriate geom_* set: binwidth to 5, boundary to 30, colour to "white" and fill to "#785EF0". What does each of these options do? *Optional:* using the previous graph, facet it by region such that each region's plot is a new row. (Hint: check the help for facet_grid.)
2. A boxplot of average hours worked `hours_worked` per year by region. Within the appropriate geom_* set: colour to "black" and fill to "#785EF0". (Hint: you need to group by both region and year.)
3. A scatter plot of fertility rate (y-axis) with respect to infant mortality (x-axis) in 2015. Once you've created the scatter plot, within the appropriate geom_* set: size to 3, alpha to 0.5, colour to "#d90502". Add labels (labs) to the plot so that it is cleaner.

Today's lecture

1. Summarizing data ✓
    - 1.1 Distributions ✓
    - 1.2 Central tendency ✓
    - 1.3 Spread ✓
    - 1.4 Relationship between variables ✓
2. Manipulating data ✓
    - 2.1 `dplyr` verbs ✓
    - 2.2 `group_by` ✓
    - 2.3 Joining data ✓
3. Visualizing data
    - 3.1 gg is for Grammar of Graphics ✓
    - 3.2 Different types of gaphs ✓

See you next week!