

R Programming & Descriptive statistics

Lecture 17

Louis SIRUGUE

CPES 2 - Spring 2023



Today: Refresher on R Programming and Descriptive Statistics

1. The basics of R programming

- 1.1. Types of R objects
- 1.2. The dplyr grammar
- 1.3. Data visualization

2. Descriptive statistics

- 2.1. Distributions
- 2.2. Central tendency
- 2.3. Spread
- 2.4. Joint distributions

3. A few words on using R

- 3.1. When it doesn't work the way you want
- 3.2. Where to find help
- 3.3. When it doesn't work at all



Today: Refresher on R Programming and Descriptive Statistics

1. The basics of R programming

- 1.1. Types of R objects
- 1.2. The dplyr grammar
- 1.3. Data visualization



1. The basics of R programming

1.1. Types of R objects

- The most **basic** element in R is just a **value**, an object of dimension 1×1 :

```
a <- 1  
a
```

```
## [1] 1
```

```
b <- "monday"  
b
```

```
## [1] "monday"
```

```
c <- a == b  
c
```

```
## [1] FALSE
```



1. The basics of R programming

1.1. Types of R objects

- Next, there are **vectors**, objects of dimension $n \times 1$:
 - Vectors can be created with **c()**
 - The elements of a vector should be of the **same class**
 - Class can be changed with **as** functions: `as.[numeric/character/logical]()`

Numeric

```
a <- 1:4  
a
```

```
## [1] 1 2 3 4
```

```
a * 2
```

```
## [1] 2 4 6 8
```

Character

```
b <- c("a", "xyz")  
b
```

```
## [1] "a"     "xyz"
```

```
paste0("b", b)
```

```
## [1] "ba"    "bxyz"   ## [1] TRUE FALSE
```

Logical

```
c <- c(F, 1 < 2)  
c
```

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

```
!c
```

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

Factor

```
d <- as.factor(a)  
d
```

```
## [1] 1 2 3 4  
## Levels: 1 2 3 4
```

```
relevel(d, 3)
```

```
## [1] 1 2 3 4  
## Levels: 3 1 2 4
```



1. The basics of R programming

1.1. Types of R objects

- Some useful functions/operators for vectors

```
vec <- c("a", "b", "c", "d")
```

```
length(vec)
```

```
## [1] 4
```

```
match("b", vec)
```

```
## [1] 2
```

```
vec[3]
```

```
## [1] "c"
```



1. The basics of R programming

1.1. Types of R objects

- Finally, there are **tables**, objects of dimension $n \times m$:
 - Gather m vectors (columns) of n observations
 - Several possible classes, e.g., **tibble()** from tidyverse

```
library(tidyverse)

data <- tibble(name = c("Bob", "Tom", "Kim"),
                age = c(43, 19, 27),
                male = c(T, T, F))
```

```
data
```

```
## # A tibble: 3 x 3
##   name     age   male
##   <chr> <dbl> <lgl>
## 1 Bob      43  TRUE
## 2 Tom      19  TRUE
## 3 Kim      27 FALSE
```



1. The basics of R programming

1.1. Types of R objects

- Such datasets can be imported on R with **read functions**
 - There is one read function per **data format** (csv, xls, dta, ...)
 - The main argument is the **path**, with slashes: "C:/User/.../data.csv"

```
data <- read.csv("data/cereals.csv") # Import csv data
data <- as_tibble(data)           # Put in tibble format
head(data, 5)                   # Print first 5 rows
```

```
## # A tibble: 5 x 16
##   name     mfr    type  calories protein    fat sodium fiber carbo sugars potass
##   <chr>    <chr>  <chr>    <int>   <int>  <int>  <dbl>  <dbl>  <int>   <int>
## 1 100% Bran N      C        70       4     1    130    10      5      6    280
## 2 100% Natu~ Q      C       120      3     5     15      2      8      8    135
## 3 All-Bran K      C        70       4     1    260     9      7      5    320
## 4 All-Bran ~ K    C       50       4     0    140    14      8      0    330
## 5 Almond De~ R    C       110      2     2    200     1     14      8     -1
## # ... with 5 more variables: vitamins <dbl>, shelf <int>, weight <dbl>,
## #   cups <dbl>, rating <dbl>
```



1. The basics of R programming

1.2. The dplyr grammar

- `dplyr` provides **useful functions** to manipulate data and the **pipe operator (%>%)** to chain operations

Important functions of the dplyr grammar

Function	Meaning
<code>mutate()</code>	Modify or create a variable
<code>select()</code>	Keep a subset of variables
<code>filter()</code>	Keep a subset of observations
<code>arrange()</code>	Sort the data
<code>group_by()</code>	Group the data
<code>summarise()</code>	Summarizes variables into 1 observation per group
<code>left/right/inner/full_join()</code>	Merge data



1. The basics of R programming

1.2. The dplyr grammar

- We can first subset the data:
 - The type **variable** only takes the value "C", we can remove it with **select()**
 - Some **observations** have negative values of potassium, we can remove them with **filter()**
 - These two operations can be **chained** using the pipe operator `%>%`

```
dim(data) # Dimensions of the data before the operation
```

```
## [1] 77 16
```

```
data <- data %>%
  select(-type) %>%
  filter(potass >= 0)
```

```
dim(data) # Dimensions of the data after the operation
```

```
## [1] 75 15
```



1. The basics of R programming

1.2. The dplyr grammar

- The **mutate()** function allows to modify and create variables
 - Using simple **vector operations**
 - With **ifelse()** to create a binary variable based on a condition
 - With **case_when()** to create a categorical variable

```
data <- data %>%  
  mutate(cal_100g = 100 * (calories / weight),  
        low_cal = ifelse(cal_100g < 100, T, F),  
        mfr = case_when(mfr == "N" ~ "Nestlé",  
                         mfr == "Q" ~ "Quaker Oats",  
                         mfr == "K" ~ "Kellogg's",  
                         mfr %in% c("G", "R") ~ "General Mills",  
                         mfr == "P" ~ "Post Consumer Brands LLC",  
                         mfr == "A" ~ "Maltex Co."))
```



1. The basics of R programming

1.2. The dplyr grammar

- Such computations can also be done **separately** for each value of a variable **with group_by()**

```
data <- data %>%
  group_by(mfr) %>%
  mutate(n_brands = n()) %>%
  ungroup()
```

- Using **summarise()** instead of **mutate()** allows to:
 - Keep only the grouping and summarized variables
 - Keep one value per group (no duplicate row)

```
data %>%
  group_by(mfr) %>%
  summarise(n_brands = n())
```

```
## # A tibble: 6 x 2
##   mfr                n_brands
##   <chr>              <int>
## 1 General Mills        29
## 2 Kellogg's             23
## 3 Maltex Co.            1
## 4 Nestlé                  5
## 5 Post Consumer Brands LLC    9
## 6 Quaker Oats             8
```



1. The basics of R programming

1.2. The dplyr grammar

- `dplyr` also provides functions to:

- Rename variables → `rename()`

```
data <- data %>% rename(manufacturer = mfr)
```

- Sort rows according to the values of one or several variables → `arrange()`

```
data <- data %>% arrange(cal_100g)
```

- Joining another dataset with a common variable → `[left/right/full/inner]_join()`:

```
data <- data %>%  
  left_join(tibble(manufacturer = c("Kellogg's", "Nestlé", "General Mills",  
                                    "Post Consumer Brands LLC", "Quaker Oats", "Maltex Co."),  
             creation = c(1906, 1966, 1928, 1895, 1877, 1899)),  
            by = "manufacturer")
```



1. The basics of R programming

1.3. Data visualization

- The tidyverse packages also gives access to the **ggplot** grammar for data visualization
- 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
- These arguments should be specified as follows:
 - Data and mapping should be specified within the parentheses
 - 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 basics of R programming

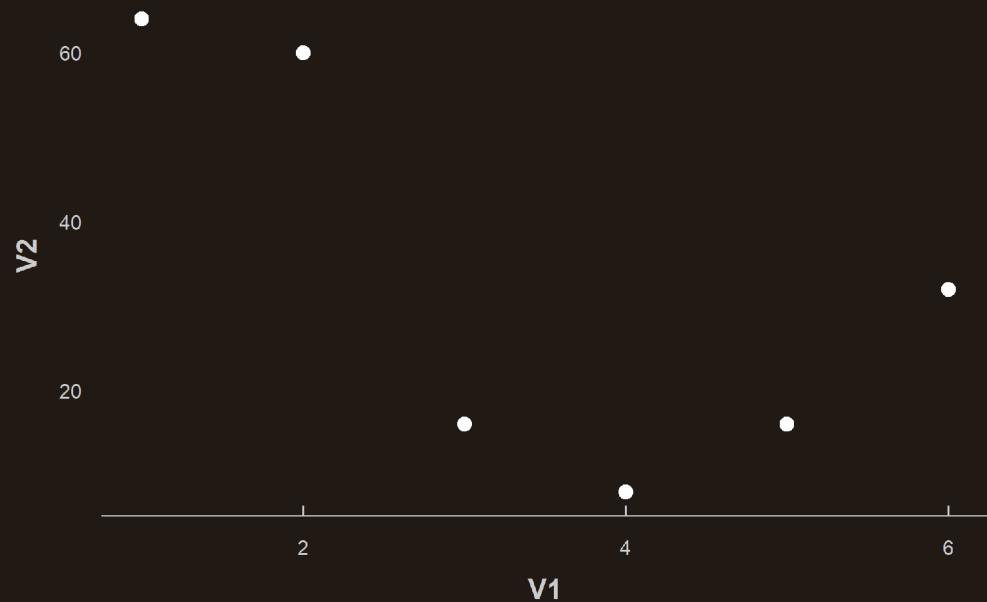
1.3. Data visualization

```
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 of 3





1. The basics of R programming

1.3. Data visualization

- In some cases you would convey information with other means than a position on axis
 - It can be with the color, size or 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 and y axis with aes, we have to assign the group variable to the 'color axis' with aes

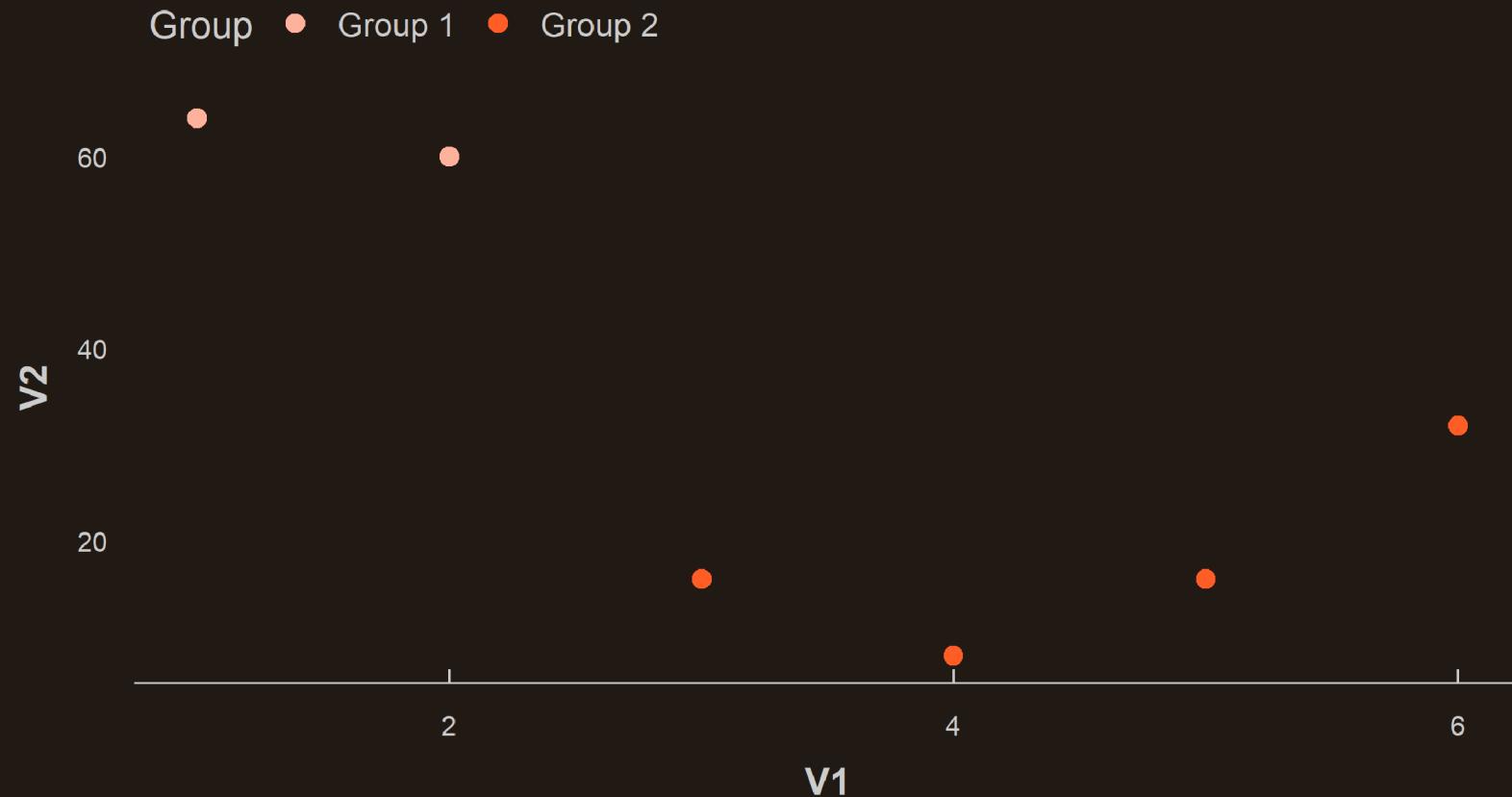
```
ggplot(test_data, aes(x = V1, y = V2,  
                      color = Group)) + ...
```

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



1. The basics of R programming

1.3. Data visualization





1. The basics of R programming

1.3. Data visualization

→ **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

```
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 assigned to in **aes**
 - So just as for regular axes, modifications should take place in a **scale** function

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

Practice

10 : 00

1) Import the dataset `cereals.csv`

2) There is no documentation on the variable `rating`. Use the `summary()` function to deduce the unit of the variable based on its distribution.

3) Generate a scatter plot with `sugars` on the `x` axis and `rating` on the `y` axis to deduce whether the rating was made by nutritionists or consumers

You've got 10 minutes!

Solution

1) Import the dataset `cereals.csv`

```
cereals <- read.csv("C:/User/Documents/cereals.csv")
```

2) There is no documentation on the variable `rating`. Use the `summary()` function to deduce the unit of the variable based on its distribution.

```
summary(cereals$rating)
```

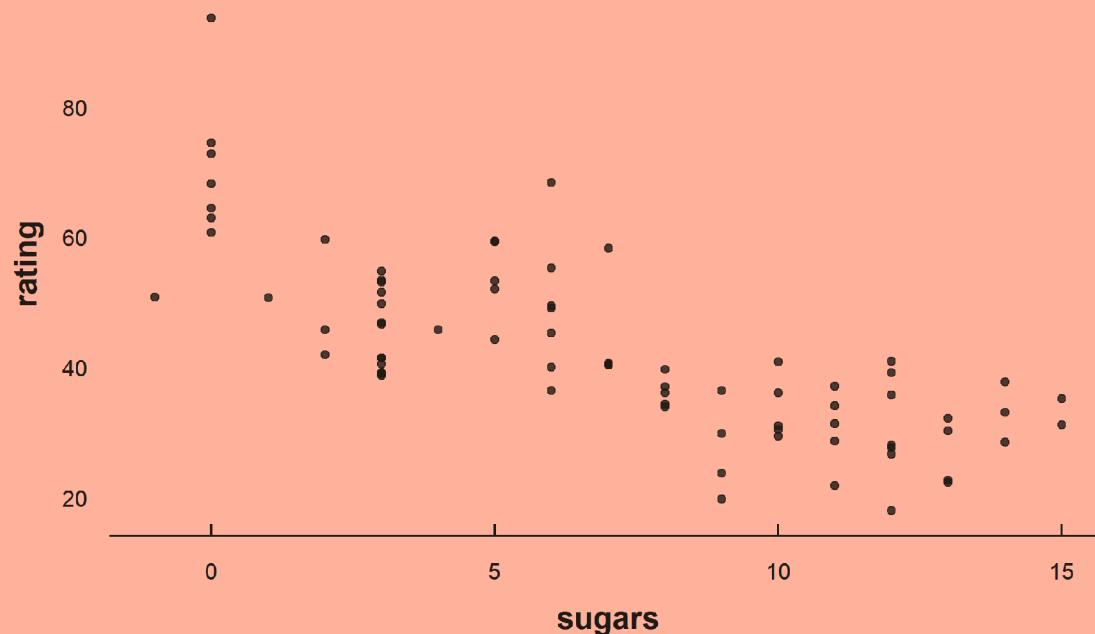
```
##      Min. 1st Qu. Median      Mean 3rd Qu.      Max.
##    18.04    33.17   40.40    42.67   50.83    93.70
```

The variable is probably in percentages

Solution

3) Generate a scatter plot with sugars on the x axis and rating on the y axis to deduce whether the rating was made by nutritionists or consumers

```
ggplot(cereals, aes(x = sugars, y = rating)) +  
  geom_point(alpha = .8)
```



The rating was probably made by nutritionists



Overview

1. The basics of R programming ✓

- 1.1. Types of R objects
- 1.2. The dplyr grammar
- 1.3. Data visualization

2. Descriptive statistics

- 2.1. Distributions
- 2.2. Central tendency
- 2.3. Spread
- 2.4. Joint distributions

3. A few words on using R

- 3.1. When it doesn't work the way you want
- 3.2. Where to find help
- 3.3. When it doesn't work at all



Overview

1. The basics of R programming ✓

- 1.1. Types of R objects
- 1.2. The dplyr grammar
- 1.3. Data visualization

2. Descriptive statistics

- 2.1. Distributions
- 2.2. Central tendency
- 2.3. Spread
- 2.4. Joint distributions



2. Descriptive statistics

2.1. Distributions

- The point of **descriptive statistics** is to **summarize variables** into a small set of tractable statistics.
- The most comprehensive way to characterize a variable is to compute its distribution:
 - What are the values the variable takes?
 - How frequently does each of these values appear?

→ Consider for instance the following variable:

Variable 1
3 5 4 6 5 4 5 7 7 6 1 7 6 7 6 4 7 7 6 6 5 6 6 3 4 5 2 6 8 8

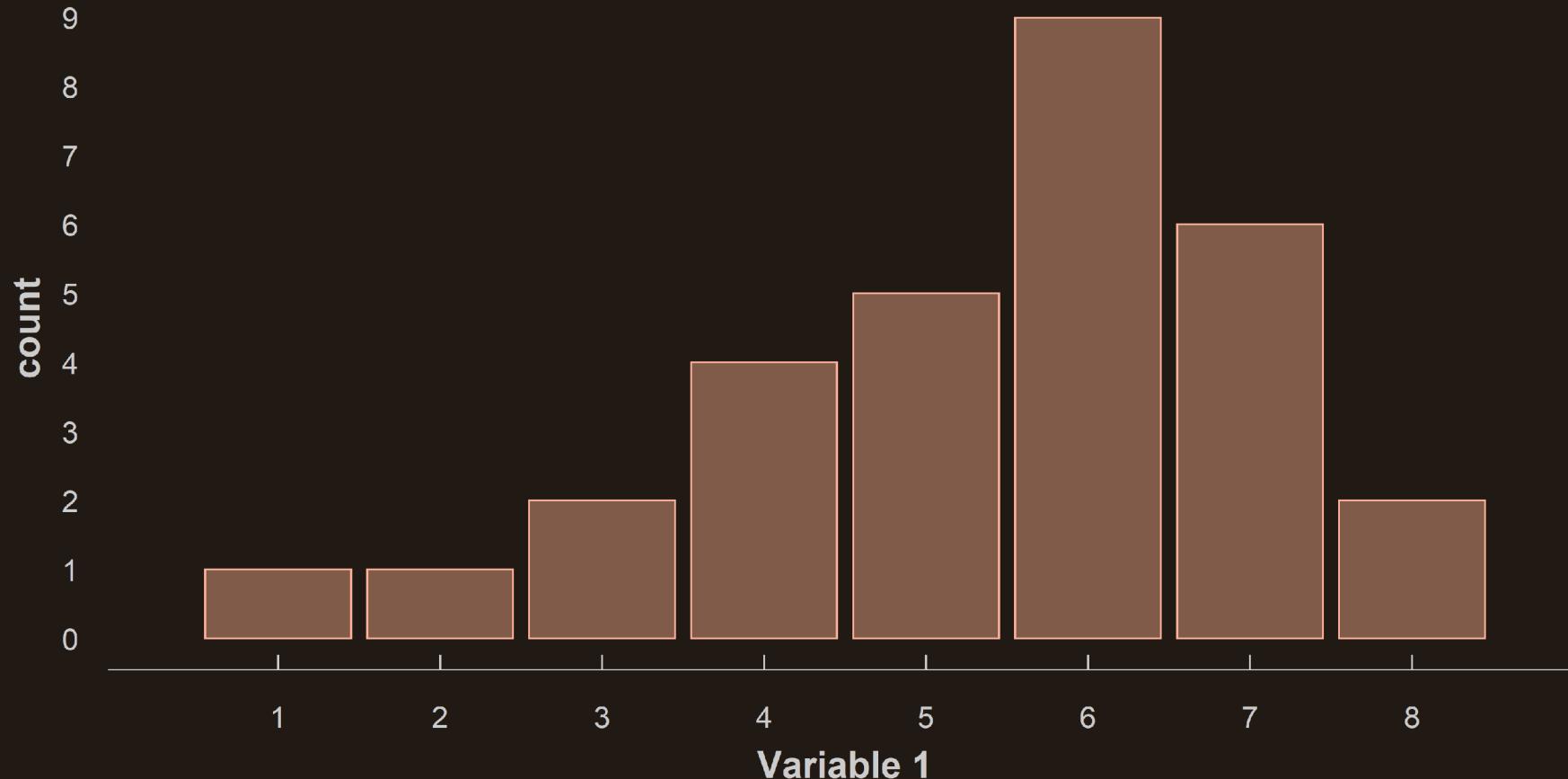
- We can count how many times each value appears
- And we can represent this distribution graphically with a bar plot
 - Each possible value on the x-axis
 - Their number of occurrences on the y-axis

Variable 1	1	2	3	4	5	6	7	8
n	1	1	2	4	5	9	6	2



2. Descriptive statistics

2.1. Distributions





2. Descriptive statistics

2.1. Distributions

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

Variable 2						
5.912877	5.006781	5.517149	5.854849	5.177872	3.815240	
1.666582	4.422721	6.025062	5.411020	5.889811	6.729103	
4.160800	6.519049	6.849172	8.368158	6.167404	2.882974	
6.751888	3.202183	6.390224	3.942039	6.488909	8.195647	
7.073922	4.790039	5.297919	1.218109	5.754213	7.225030	

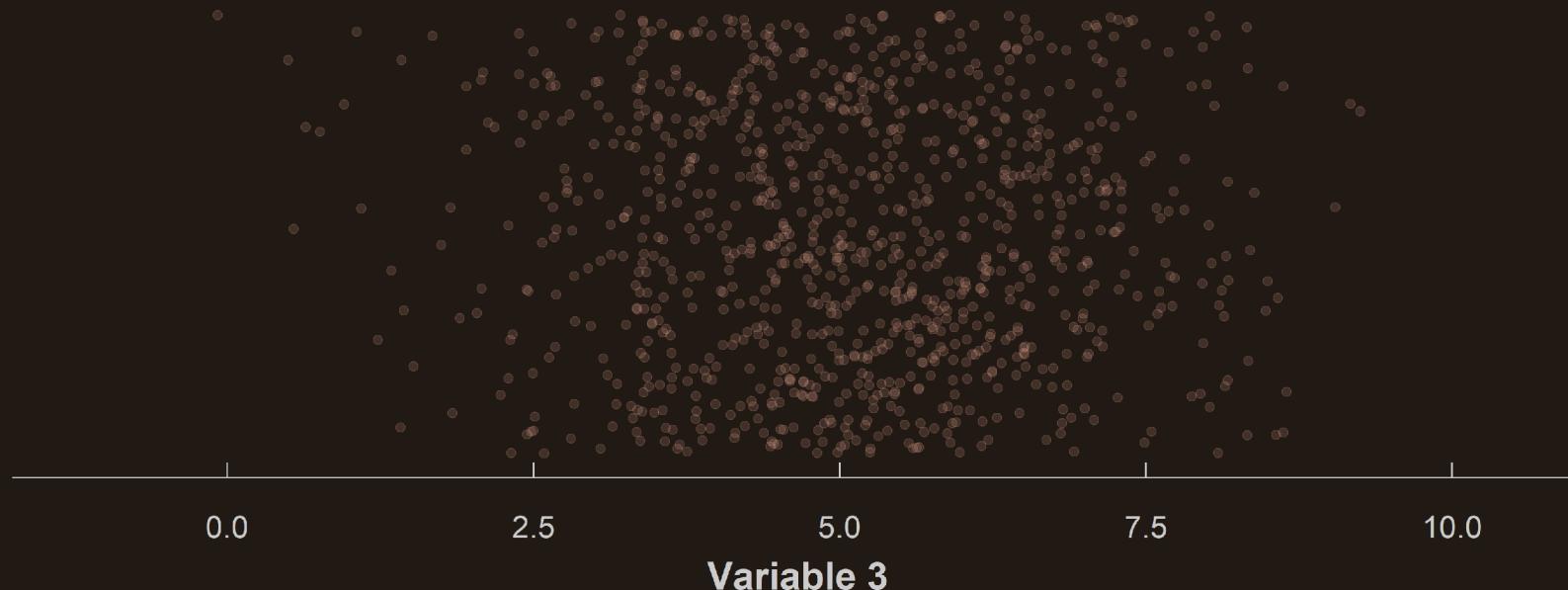
- Each value appears only once
 - So the count of each value does not help summarizing the variable

→ **We should rather do a histogram**

2. Descriptive statistics

2.1. Distributions

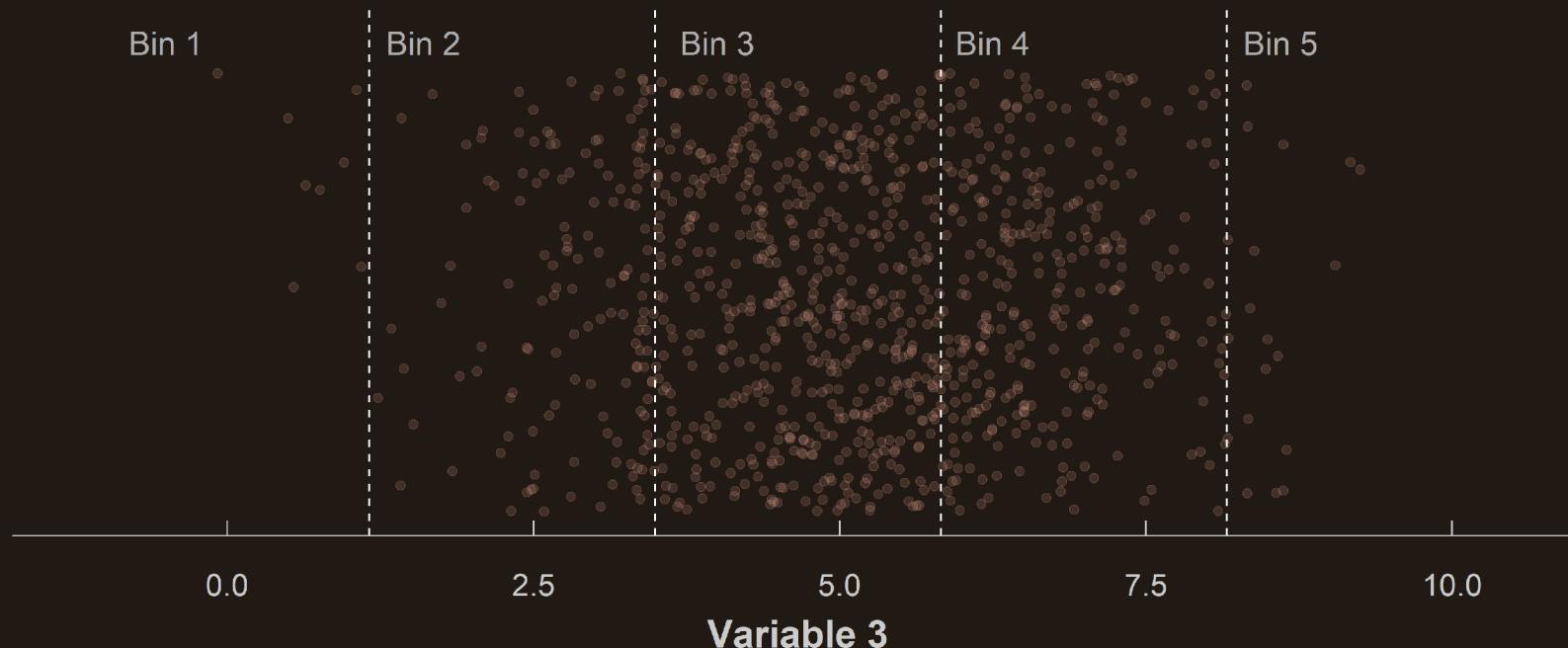
- Consider for instance the following variable. For clarity each point is shifted vertically by a random amount.



2. Descriptive statistics

2.1. Distributions

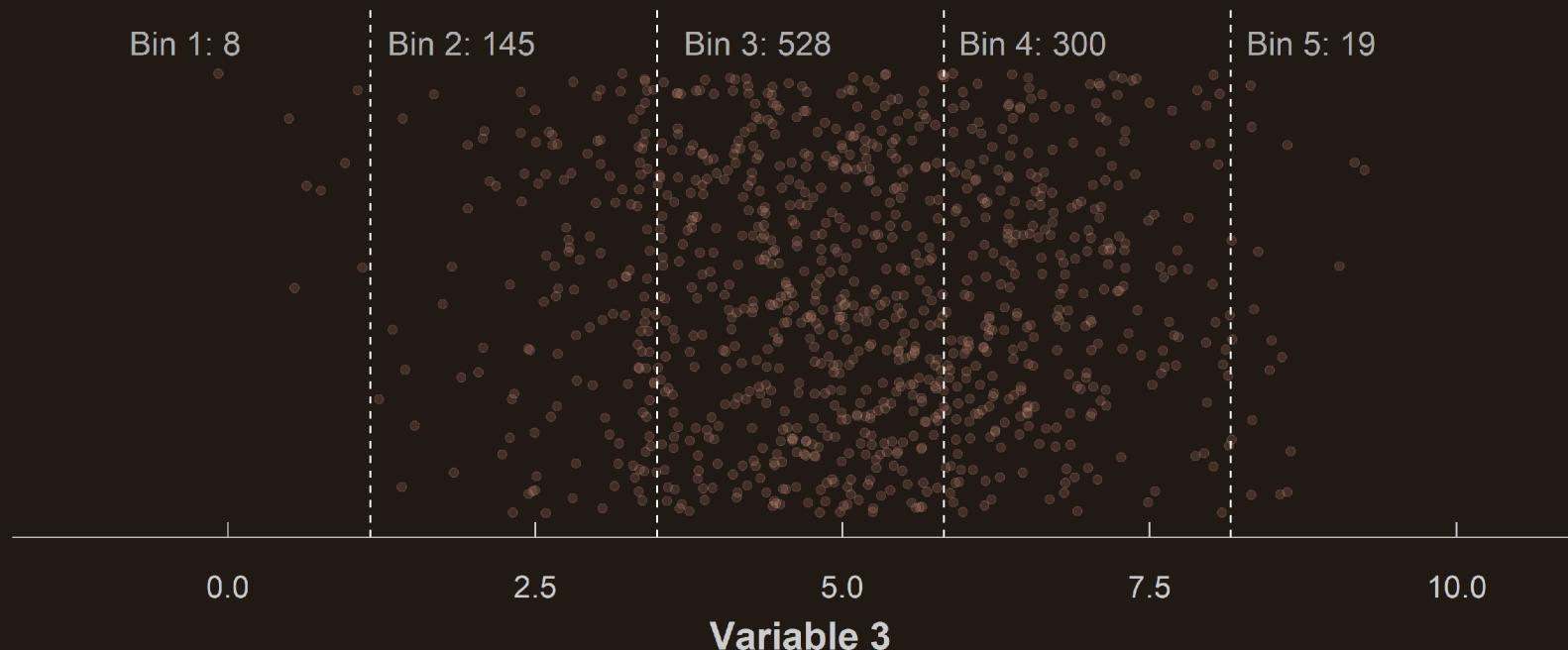
- Consider for instance the following variable. For clarity each point is shifted vertically by a random amount.
 - We can divide the domain of this variable into 5 bins



2. Descriptive statistics

2.1. Distributions

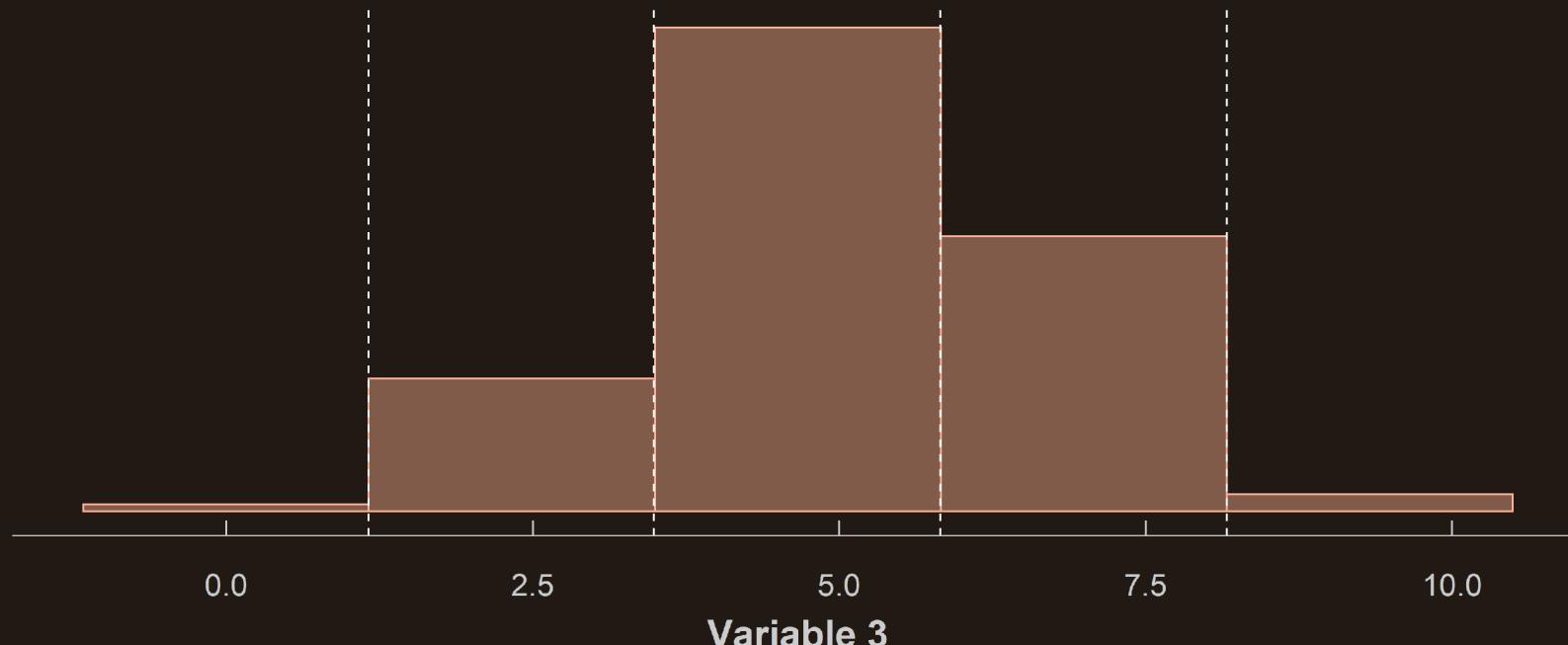
- Consider for instance the following variable. For clarity each point is shifted vertically by a random amount.
 - We can divide the domain of this variable into 5 bins
 - And count the number of observations within each bin



2. Descriptive statistics

2.1. Distributions

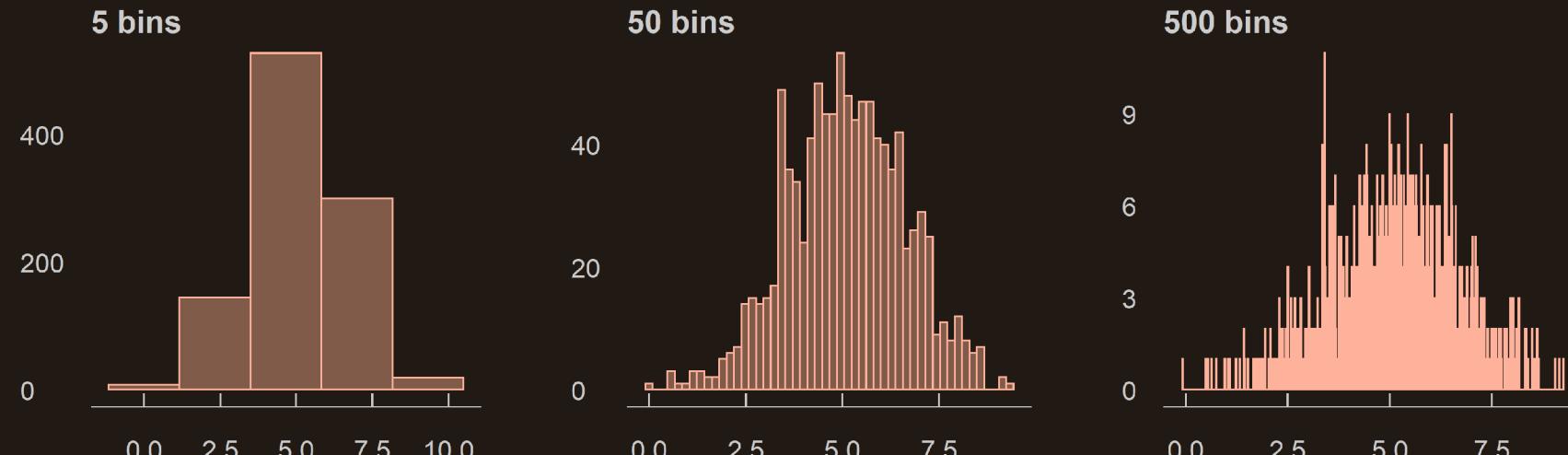
- Consider for instance the following variable. For clarity each point is shifted vertically by a random amount.
 - We can divide the domain of this variable into 5 bins
 - And count the number of observations within each bin



2. Descriptive statistics

2.1. Distributions

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



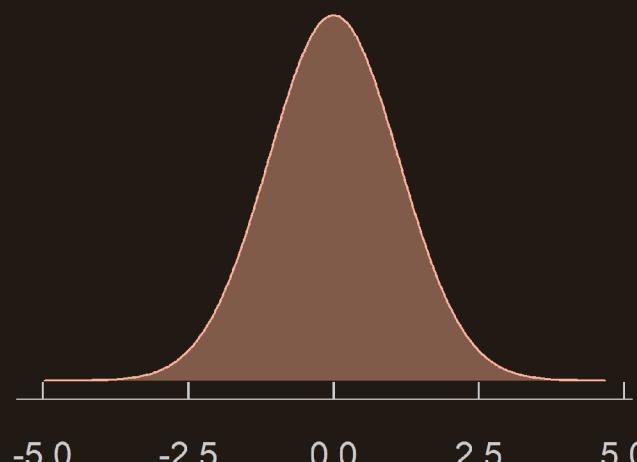
- Densities are often used instead of histograms
 - Both are based on the same principle, but densities are continuous

2. Descriptive statistics

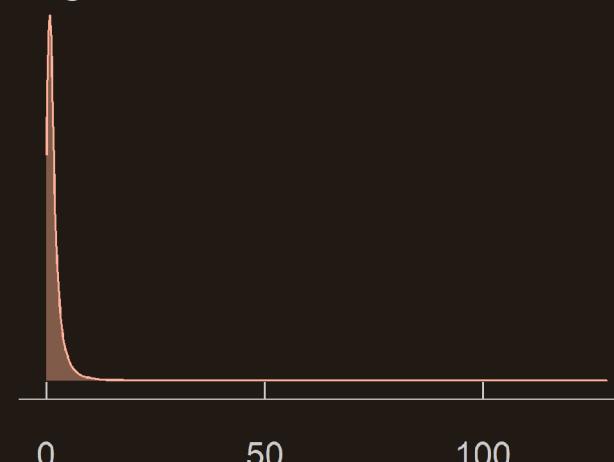
2.1. Distributions

- Distributions are comprehensive representations but not simple statistics

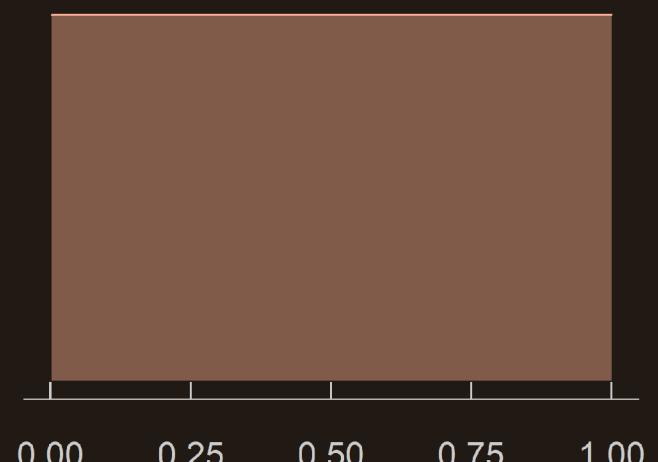
Normal distribution



Log-normal distribution



Uniform distribution

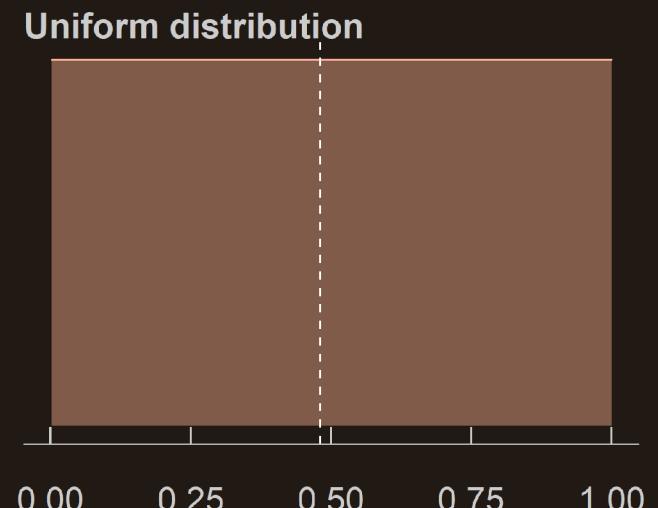
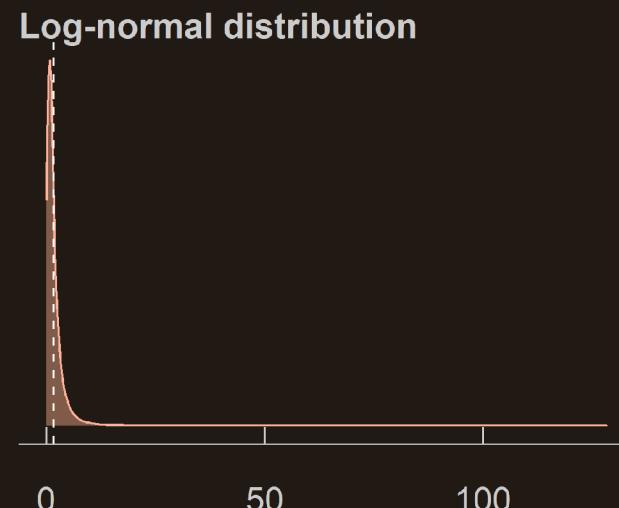
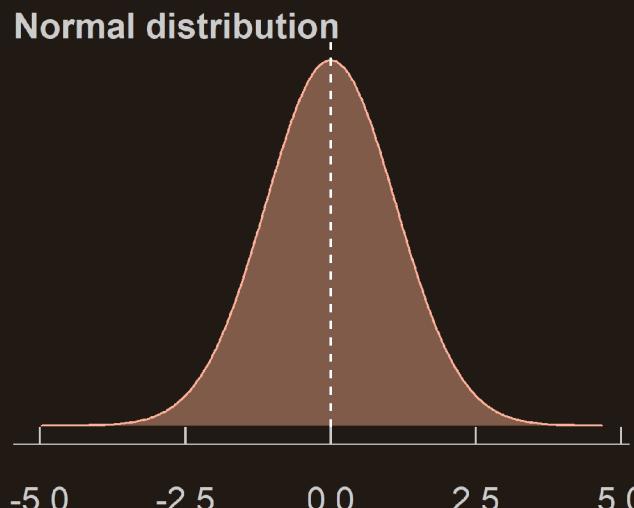


- How to summarize these distributions with simple statistics?

2. Descriptive statistics

2.1. Distributions

- Distributions are comprehensive representations but not simple statistics

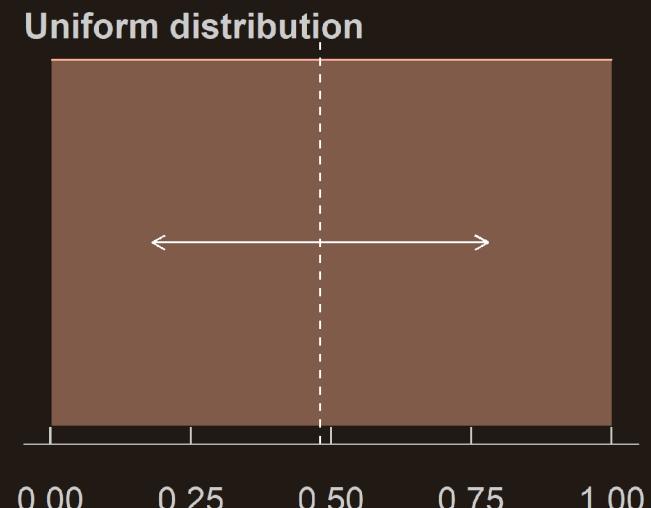
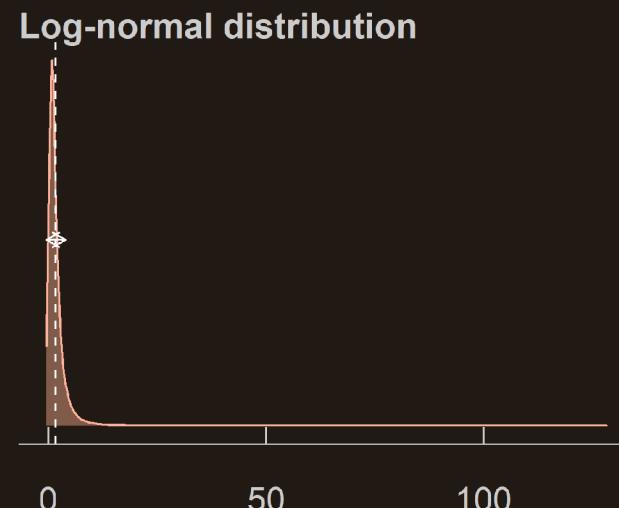
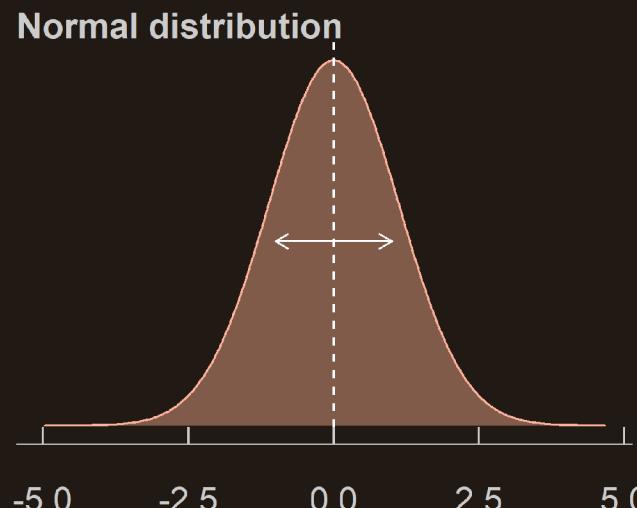


- How to summarize these distributions with simple statistics?
 - By describing their central tendency (e.g., mean, median)

2. Descriptive statistics

2.1. Distributions

- Distributions are comprehensive representations but not simple statistics



- 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)



2. Descriptive statistics

2.2. Central tendency

- The mean is the most common statistic to describe central tendencies
 - Take for instance the grades of group 2 last year for the second-semester final project

Grades of G2 last year							
20	17.5	16	16.0	14.5	19.5	18.5	
20	17.5	16	14.5	19.5	18.5	18.5	

- The mean is simply the sum of all the grades divided by the number of grades:

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

$$\frac{20 + 20 + 17.5 + 17.5 + 16 + 16 + 16 + 14.5 + 14.5 + 19.5 + 19.5 + 18.5 + 18.5 + 18.5}{14} = 17.61$$



2. Descriptive statistics

2.2. Central tendency

- The mean is the most common statistic to describe central tendencies
 - Take for instance the grades of group 2 last year for the second-semester final project

Grades of G2 last year							
20	17.5	16	16.0	14.5	19.5	18.5	
20	17.5	16	14.5	19.5	18.5	18.5	

- It can also be expressed as the average of each possible value weighted by its number of occurrences:

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

$$\bar{x} = \frac{(2 \times 20) + (2 \times 17.5) + (3 \times 16) + (2 \times 14.5) + (2 \times 19.5) + (3 \times 18.5)}{2 + 2 + 3 + 2 + 2 + 3} = 17.61$$

2. Descriptive statistics

2.2. Central tendency

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

Grades of G2 last year														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	
14.5	14.5	16	16	16	17.5	17.5	18.5	18.5	18.5	19.5	19.5	20	20	

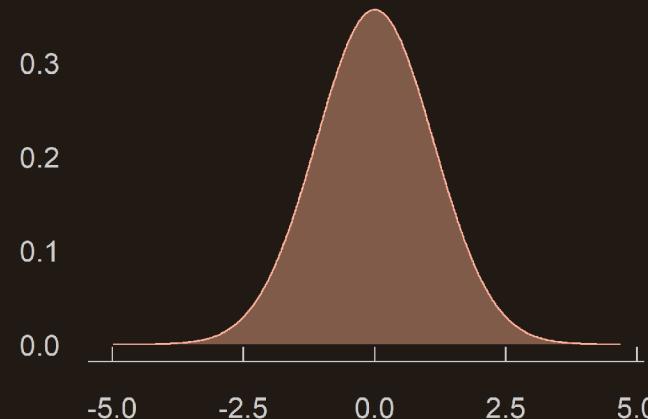
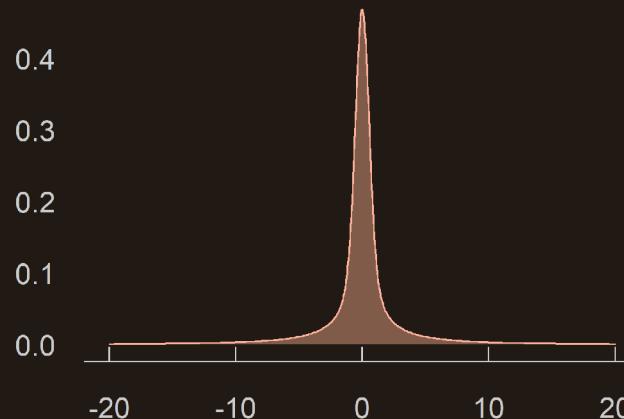
- The median is the value that divides the distribution into two halves
 - With N even: Average of the last value of the first half and the first value of the second half
- As we have 14 observations, here the median is the average of the 7th and the 8th observations:

$$\text{Med}(x) = \begin{cases} x\left[\frac{N+1}{2}\right] & \text{if } N \text{ is odd} \\ \frac{x\left[\frac{N}{2}\right] + x\left[\frac{N}{2}+1\right]}{2} & \text{if } N \text{ is even} \end{cases} = \frac{17.5 + 18.5}{2} = 18$$

2. Descriptive statistics

2.3. Spread

- The most intuitive statistic to describe the spread of a variable is probably its **range**
 - The minimum and maximum value of the distribution
- But consider the following two distributions:

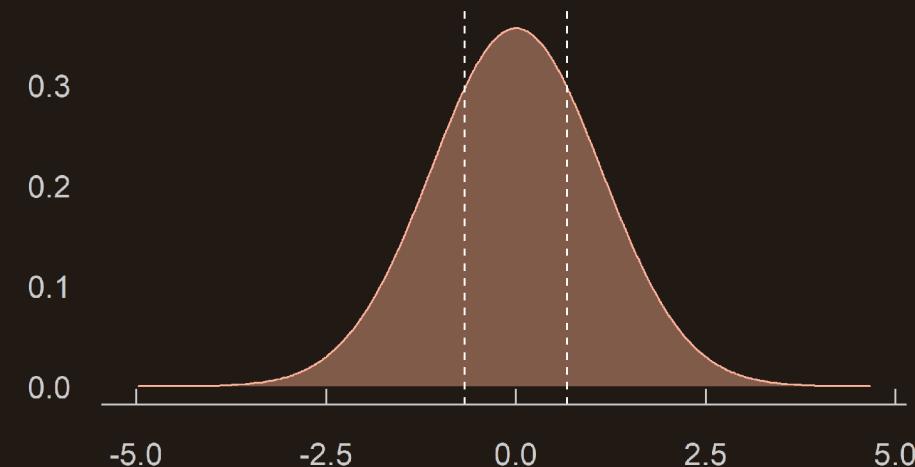
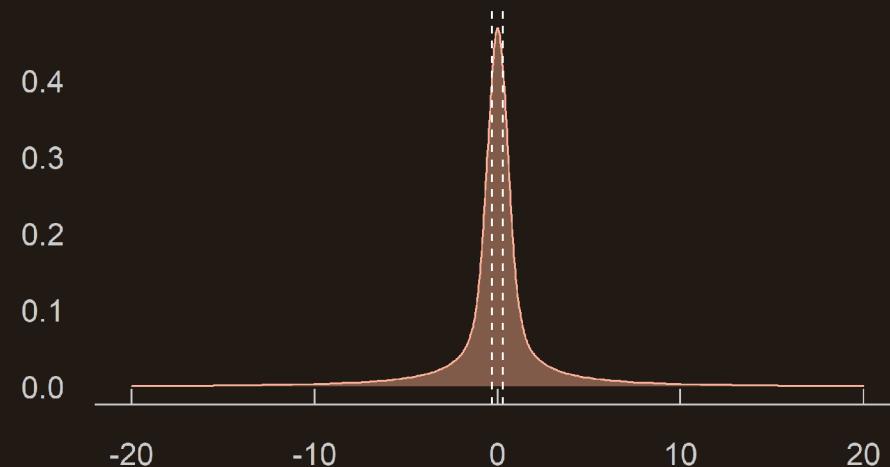


- In the presence of outliers or very skewed distributions, the full range of a variable may not be representative of what we mean by 'spread'
- That's why we tend to prefer:
 - The **inter-quantile range**
 - The **standard deviation**

2. Descriptive statistics

2.3. Spread

- **Quantiles** are observations that 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
- **The interquartile range** is the difference between the third and the first quartile: $IQR = Q_3 - Q_1$
 - Put differently, it corresponds to the bounds of the set which contains the **middle half of the distribution**





2. Descriptive statistics

2.3. Spread

- The **variance** is a way to quantify how 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
- Because deviations from the mean sum to 0, they have to be squared
 - This is how the variance is computed: by **averaging the squared deviations from the mean**

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

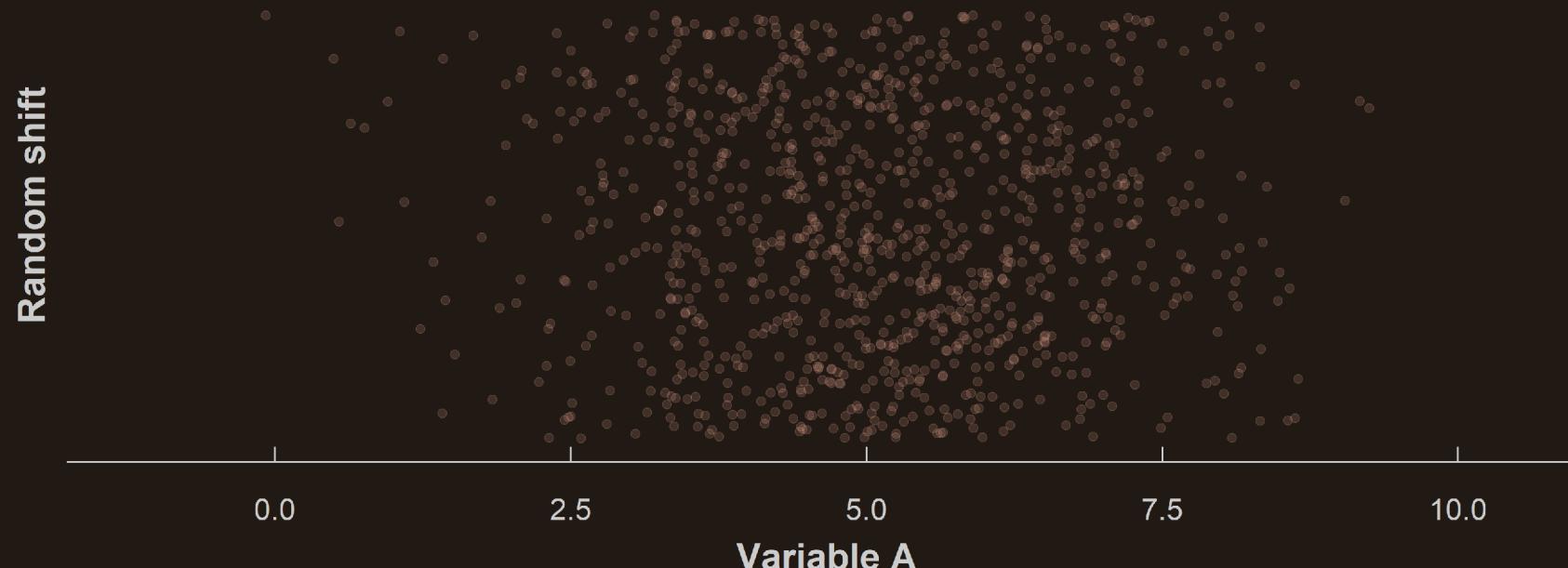
- The variance is a sum of squares, so we have to take its square root to remain in the same unit as the data
 - This is what we call the **standard deviation**

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

2. Descriptive statistics

2.4. Joint distributions

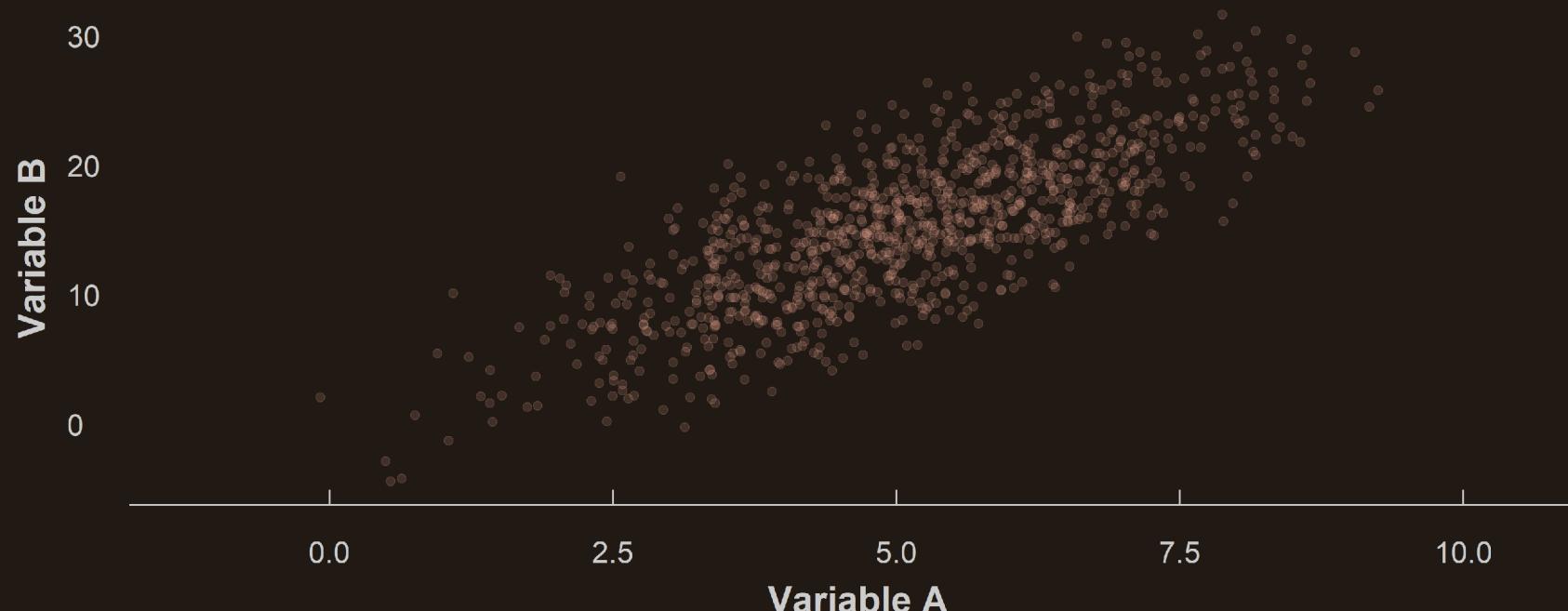
- The joint distribution shows the possible values and associated frequencies for two variables simultaneously
 - Earlier we plotted the observations of a variable on a line, randomly shifted on the vertical axis



2. Descriptive statistics

2.4. Joint distributions

- The joint distribution shows the possible values and associated frequencies for two variable simultaneously
 - Earlier we plotted the observations of a variable on a line, randomly shifted on the vertical axis
 - Instead of shifting observations randomly, vertical coordinates can indicate the value of a second variable





2. Descriptive statistics

2.4. Joint distributions

- When describing a **single distribution**, we're interested in its **spread and central tendency**
- When describing a **joint distribution**, we're interested in the **relationship between the two variables**
 - This can be characterized by the covariance

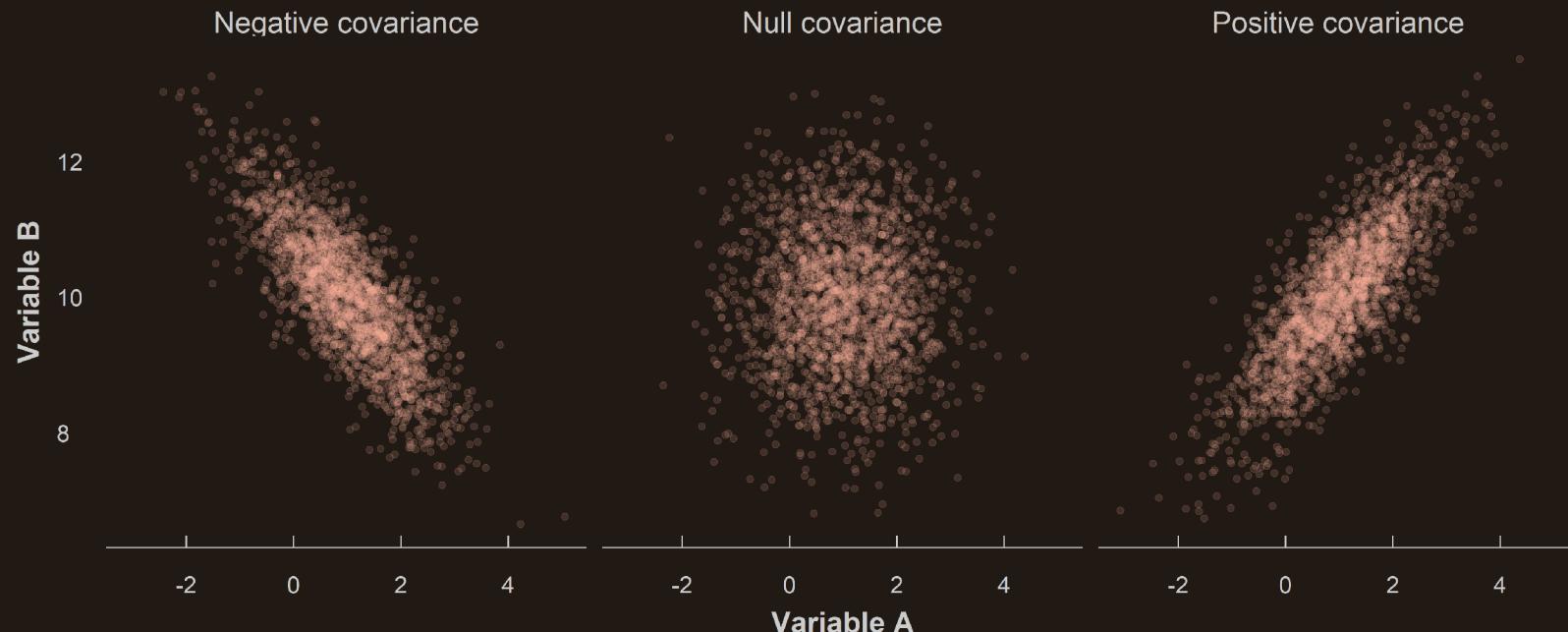
$$\text{Cov}(x, y) = \frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})(y_i - \bar{y})$$

- The contribution of observation i to $\text{Cov}(x, y)$ is:
 - Positive when both x_i and y_i are above their respective mean
 - Positive when both x_i and y_i are below their respective mean
 - Negative when x_i and y_i are on different sides of their respective mean

→ *If y tends to be large relative to its mean when x is large relative to its mean, their covariance is positive. Conversely, if one tends to be large when the other tends to be low, the covariance is negative.*

2. Descriptive statistics

2.4. Joint distributions



- One disadvantage of the **covariance** is that it is **not standardized**
 - You cannot directly compare the covariance of two pairs of completely different variables
 - Theoretically the covariance can take values from $-\infty$ to $+\infty$

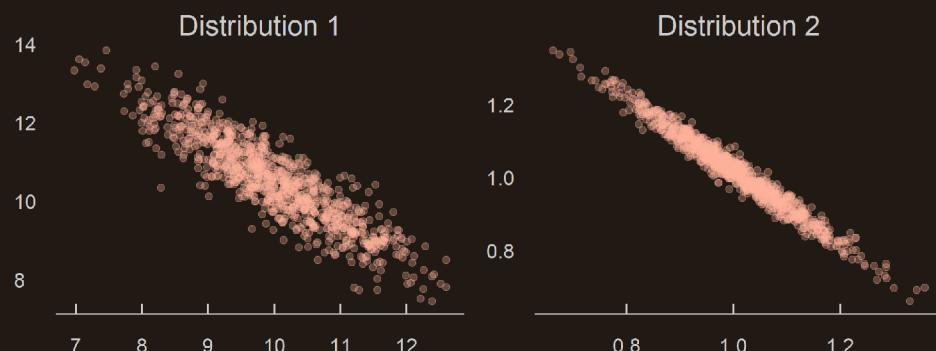
2. Descriptive statistics

2.4. Joint distributions

- This is why we often use the **correlation coefficient**
 - It is obtained by dividing the covariance by the product of the standard deviation of the two variables
 - This allows to **standardize the coefficient** between -1 and 1

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

- Consider for instance the following two distributions:



- Here the association between the two variables feels tighter on the right panel
 - But the covariance is larger for the first relationship because units are larger
 - While the correlation, standardized between 0 and 1, is larger for the second one



Overview

1. The basics of R programming ✓

- 1.1. Types of R objects
- 1.2. The dplyr grammar
- 1.3. Data visualization

2. Descriptive statistics ✓

- 2.1. Distributions
- 2.2. Central tendency
- 2.3. Spread
- 2.4. Joint distributions

3. A few words on using R

- 3.1. When it doesn't work the way you want
- 3.2. Where to find help
- 3.3. When it doesn't work at all



3. A few words on using R

3.1. When it doesn't work the way you want

- 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:

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

```
## [1] NA
```

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

```
## [1] 1.5
```

- Here it is obvious that NAs are the problem, but when chaining operations it's not always that transparent
 - So check your data using **is.na()** to see whether NAs could mess things up

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

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



3. A few words on using R

3.2. Where to find help

- You can find help on **help files**
 - Sometimes things don't work just because you did not understand the arguments of the function
 - Just enter the name of the function preceded by a **?** in your console
 - The help file will appear in the Help tab of R studio

```
?pivot_longer
```

Arguments

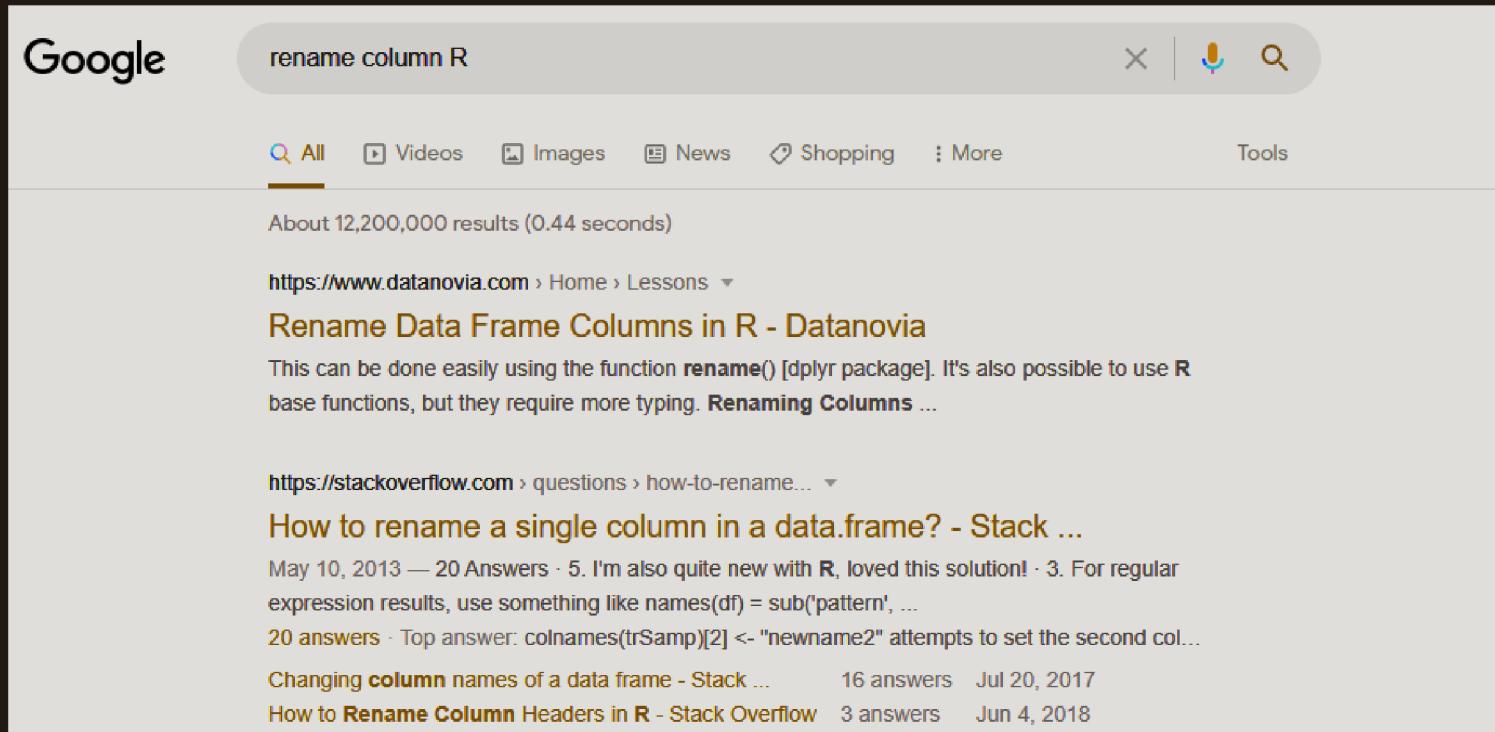
<code>data</code>	A data frame to pivot.
<code>cols</code>	<code><tidy-select></code> Columns to pivot into longer format.
<code>names_to</code>	A string specifying the name of the column to create from the data stored in the column names of <code>data</code> . Can be a character vector, creating multiple columns, if <code>names_sep</code> or <code>names_pattern</code> is provided. In this case, there are two special values you can take advantage of: <ul style="list-style-type: none">• <code>NA</code> will discard that component of the name.• <code>.value</code> indicates that component of the name defines the name of the column containing the cell values, overriding <code>values_to</code>.
<code>names_prefix</code>	A regular expression used to remove matching text from the start of each variable name.



3. A few words on using R

3.2. Where to find help

- When it doesn't work, search on the **internet**
 - **Every question** you might have at that stage is already asked and **answered** at stackoverflow.com



A screenshot of a Google search results page. The search query "rename column R" is entered in the search bar. The results show approximately 12,200,000 results found in 0.44 seconds. The top result is a link to "Rename Data Frame Columns in R - Datanovia". Below it is a link to a Stack Overflow question titled "How to rename a single column in a data.frame? - Stack ...". Other visible links include "Changing column names of a data frame - Stack ..." and "How to Rename Column Headers in R - Stack Overflow".

Google

rename column R

All Videos Images News Shopping More Tools

About 12,200,000 results (0.44 seconds)

<https://www.datanovia.com> › Home › Lessons

Rename Data Frame Columns in R - Datanovia

This can be done easily using the function `rename()` [dplyr package]. It's also possible to use R base functions, but they require more typing. [Renaming Columns ...](#)

<https://stackoverflow.com> › questions › how-to-rename...

How to rename a single column in a data.frame? - Stack ...

May 10, 2013 — 20 Answers · 5. I'm also quite new with R, loved this solution! · 3. For regular expression results, use something like `names(df) = sub('pattern', ...)`

20 answers · Top answer: `colnames(trSamp)[2] <- "newname2"` attempts to set the second col...

Changing column names of a data frame - Stack ... 16 answers Jul 20, 2017

How to Rename Column Headers in R - Stack Overflow 3 answers Jun 4, 2018



3. A few words on using R

3.3. When it doesn't work at all

- Sometimes R breaks and returns an error, which is usually kind of cryptic

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

```
## Error: '\U' non suivi de chiffres hexadécimaux dans la chaîne de caractères débutant ""C:\U"
```

- Try to look for keywords that might help you understand where it comes from
- And paste it in Google with the name of your command, chances are many people already struggled with that

The screenshot shows a Google search results page. The search query in the bar is "'U' used without hex digits in character string starting \"C:\U\"'. The results page indicates there are about 711,000 results. The top result is a link to a Stack Overflow question titled 'starting R: Error: '\U' used without hex digits in character string ...' posted on May 30, 2017, with 8 answers.

Google

'U' used without hex digits in character string starting "C:\U"

All Shopping Videos News Images More Tools

About 711,000 results (0.65 seconds)

<https://stackoverflow.com/questions/44383119/startin...>

starting R: Error: '\U' used without hex digits in character string ...

May 30, 2017 · 8 answers