Лекція 1. Вступ в аналіз даних та R

Данило Тавров

08.02.2023

План лекції

Викладач

- Тавров Данило Юрійович
- Освіта:
  - 2011: кафедра ПМА, бакалавр
  - 2013: кафедра ПМА, магістр
  - 2016: кафедра ПМА, канд. техн. наук
  - 2019: Київська школа економіки, магістр
  - 2022: University of California, Berkeley, магістр економіки
- Увесь час з 2013 р. з перервою на Берклі — на кафедрі ПМА
- Контакти:

Викладач

- Тавров Данило Юрійович
- Освіта:
  - 2011: кафедра ПМА, бакалавр
  - 2013: кафедра ПМА, магістр
  - 2016: кафедра ПМА, канд. техн. наук
  - 2019: Київська школа економіки, магістр
  - 2022: University of California, Berkeley, магістр економіки
- Увесь час з 2013 р. з перервою на Берклі — на кафедрі ПМА
- Контакти:

Викладач

- Тавров Данило Юрійович
- Освіта:
  - 2011: кафедра ПМА, бакалавр
  - 2013: кафедра ПМА, магістр
  - 2016: кафедра ПМА, канд. техн. наук
  - 2019: Київська школа економіки, магістр
  - 2022: University of California, Berkeley, магістр економіки
- Увесь час з 2013 р. з перервою на Берклі — на кафедрі ПМА
- Контакти:

Викладач

- Тавров Данило Юрійович
- Освіта:
  - 2011: кафедра ПМА, бакалавр
  - 2013: кафедра ПМА, магістр
  - 2016: кафедра ПМА, канд. техн. наук
  - 2019: Київська школа економіки, магістр
  - 2022: University of California, Berkeley, магістр економіки
- Увесь час з 2013 р. з перервою на Берклі — на кафедрі ПМА
- Контакти:

Викладач

- Тавров Данило Юрійович
- Освіта:
  - 2011: кафедра ПМА, бакалавр
  - 2013: кафедра ПМА, магістр
  - 2016: кафедра ПМА, канд. техн. наук
  - 2019: Київська школа економіки, магістр
  - 2022: University of California, Berkeley, магістр економіки
- Увесь час з 2013 р. з перервою на Берклі — на кафедрі ПМА
- Контакти:

Викладач

- Тавров Данило Юрійович
- Освіта:
  - 2011: кафедра ПМА, бакалавр
  - 2013: кафедра ПМА, магістр
  - 2016: кафедра ПМА, канд. техн. наук
  - 2019: Київська школа економіки, магістр
  - 2022: University of California, Berkeley, магістр економіки
- Увесь час з 2013 р. з перервою на Берклі — на кафедрі ПМА
- Контакти:

Викладач

- Тавров Данило Юрійович
- Освіта:
  - 2011: кафедра ПМА, бакалавр
  - 2013: кафедра ПМА, магістр
  - 2016: кафедра ПМА, канд. техн. наук
  - 2019: Київська школа економіки, магістр
  - 2022: University of California, Berkeley, магістр економіки
- Увесь час з 2013 р. з перервою на Берклі — на кафедрі ПМА
- Контакти:
  - dtavrov@ukr.net
  - Slack: відповідаю в робочий час
  - Офіс: 11 корпус — за домовленістю

Викладач

- Тавров Данило Юрійович
- Освіта:
  - 2011: кафедра ПМА, бакалавр
  - 2013: кафедра ПМА, магістр
  - 2016: кафедра ПМА, канд. техн. наук
  - 2019: Київська школа економіки, магістр
  - 2022: University of California, Berkeley, магістр економіки
- Увесь час з 2013 р. з перервою на Берклі — на кафедрі ПМА
- Контакти:
  - d.tavrov@kpi.ua
  - Slack: відповідаю в робочий час
  - Офіси і години — за домовленістю

Викладач

- Тавров Данило Юрійович
- Освіта:
  - 2011: кафедра ПМА, бакалавр
  - 2013: кафедра ПМА, магістр
  - 2016: кафедра ПМА, канд. техн. наук
  - 2019: Київська школа економіки, магістр
  - 2022: University of California, Berkeley, магістр економіки
- Увесь час з 2013 р. з перервою на Берклі — на кафедрі ПМА
- Контакти:
  - d.tavrov@kpi.ua
  - Slack: відповідаю в робочий час
  - Офісні години — за домовленістю

Викладач

- Тавров Данило Юрійович
- Освіта:
  - 2011: кафедра ПМА, бакалавр
  - 2013: кафедра ПМА, магістр
  - 2016: кафедра ПМА, канд. техн. наук
  - 2019: Київська школа економіки, магістр
  - 2022: University of California, Berkeley, магістр економіки
- Увесь час з 2013 р. з перервою на Берклі — на кафедрі ПМА
- Контакти:
  - d.tavrov@kpi.ua
  - Slack: відповідаю в робочий час
  - Офісні години — за домовленістю

Викладач

- Тавров Данило Юрійович
- Освіта:
  - 2011: кафедра ПМА, бакалавр
  - 2013: кафедра ПМА, магістр
  - 2016: кафедра ПМА, канд. техн. наук
  - 2019: Київська школа економіки, магістр
  - 2022: University of California, Berkeley, магістр економіки
- Увесь час з 2013 р. з перервою на Берклі — на кафедрі ПМА
- Контакти:
  - d.tavrov@kpi.ua
  - Slack: відповідаю в робочий час
  - Офісні години — за домовленістю

Викладач

- Тавров Данило Юрійович
- Освіта:
  - 2011: кафедра ПМА, бакалавр
  - 2013: кафедра ПМА, магістр
  - 2016: кафедра ПМА, канд. техн. наук
  - 2019: Київська школа економіки, магістр
  - 2022: University of California, Berkeley, магістр економіки
- Увесь час з 2013 р. з перервою на Берклі — на кафедрі ПМА
- Контакти:
  - d.tavrov@kpi.ua
  - Slack: відповідаю в робочий час
  - Офісні години — за домовленістю

Де шукати матеріали до курсу

- Усі матеріали для курсу зібрано на Google-диску
- Там будуть усі лекції та додаткові матеріали
- Усі заняття проходять у Zoom:
  - Лекції — за тими посиланнями
  - Презентація лабораторних робіт — за цими посиланнями

Де шукати матеріали до курсу

- Усі матеріали для курсу зібрано на Google-диску
- Там будуть усі лекції та додаткові матеріали
- Усі заняття проходять у Zoom:
  - Лекції — за цим посиланням
  - Презентація лабораторних робіт — за цим посиланням

Де шукати матеріали до курсу

- Усі матеріали для курсу зібрано на Google-диску
- Там будуть усі лекції та додаткові матеріали
- Усі заняття проходять у Zoom:
  - Лекції — за цим посиланням
  - Презентація лабораторних робіт — за цим посиланням

Де шукати матеріали до курсу

- Усі матеріали для курсу зібрано на Google-диску
- Там будуть усі лекції та додаткові матеріали
- Усі заняття проходять у Zoom:
  - Лекції — за цим посиланням
  - Презентація лабораторних робіт — за цим посиланням

Де шукати матеріали до курсу

- Усі матеріали для курсу зібрано на Google-диску
- Там будуть усі лекції та додаткові матеріали
- Усі заняття проходять у Zoom:
  - Лекції — за цим посиланням
  - Презентація лабораторних робіт — за цим посиланням

Структура курсу

- Вступ в аналіз даних, основи R
- Розвідковий аналіз даних (Exploratory data analysis, EDA)
- Основи статистичного виведення (statistical inference) за допомогою R (статистичні оцінки, тестування гіпотез, бутстреп (bootstrap))
- Регресійний аналіз
- Непараметричні методи (у т.ч. аналіз головних компонент (principle component analysis, PCA), вейвлет-аналіз)
- Беєсівське виведення (Bayesian inference)
- Основи причиново-наслідкового виведення (causal inference)

Структура курсу

- Вступ в аналіз даних, основи R
- Розвідковий аналіз даних (Exploratory data analysis, EDA)
- Основи статистичного виведення (statistical inference) за допомогою R (статистичні оцінки, тестування гіпотез, бутстреп (bootstrap))
- Регресійний аналіз
- Непараметричні методи (у т.ч. аналіз головних компонент (principle component analysis, PCA), вейвлет-аналіз)
- Беєсівське виведення (Bayesian inference)
- Основи причиново-наслідкового виведення (causal inference)

Структура курсу

- Вступ в аналіз даних, основи R
- Розвідковий аналіз даних (Exploratory data analysis, EDA)
- Основи статистичного виведення (statistical inference) за допомогою R (статистичні оцінки, тестування гіпотез, бутстреп (bootstrap))
- Регресійний аналіз
- Непараметричні методи (у т.ч. аналіз головних компонент (principle component analysis, PCA), вейвлет-аналіз)
- Беєсівське виведення (Bayesian inference)
- Основи причиново-наслідкового виведення (causal inference)

Структура курсу

- Вступ в аналіз даних, основи R
- Розвідковий аналіз даних (Exploratory data analysis, EDA)
- Основи статистичного виведення (statistical inference) за допомогою R (статистичні оцінки, тестування гіпотез, бутстреп (bootstrap))
- Регресійний аналіз
- Непараметричні методи (у т.ч. аналіз головних компонент (principle component analysis, PCA), вейвлет-аналіз)
- Беєсівське виведення (Bayesian inference)
- Основи причиново-наслідкового виведення (causal inference)

Структура курсу

- Вступ в аналіз даних, основи R
- Розвідковий аналіз даних (Exploratory data analysis, EDA)
- Основи статистичного виведення (statistical inference) за допомогою R (статистичні оцінки, тестування гіпотез, бутстреп (bootstrap))
- Регресійний аналіз
- Непараметричні методи (у т.ч. аналіз головних компонент (principle component analysis, PCA), вейвлет-аналіз)
- Беєсівське виведення (Bayesian inference)
- Основи причиново-наслідкового виведення (causal inference)

Структура курсу

- Вступ в аналіз даних, основи R
- Розвідковий аналіз даних (Exploratory data analysis, EDA)
- Основи статистичного виведення (statistical inference) за допомогою R (статистичні оцінки, тестування гіпотез, бутстреп (bootstrap))
- Регресійний аналіз
- Непараметричні методи (у т.ч. аналіз головних компонент (principle component analysis, PCA), вейвлет-аналіз)
- Беєсівське виведення (Bayesian inference)
- Основи причиново-наслідкового виведення (causal inference)

Структура курсу

- Вступ в аналіз даних, основи R
- Розвідковий аналіз даних (Exploratory data analysis, EDA)
- Основи статистичного виведення (statistical inference) за допомогою R (статистичні оцінки, тестування гіпотез, бутстреп (bootstrap))
- Регресійний аналіз
- Непараметричні методи (у т.ч. аналіз головних компонент (principle component analysis, PCA), вейвлет-аналіз)
- Беєсівське виведення (Bayesian inference)
- Основи причиново-наслідкового виведення (causal inference)

Рейтингова система оцінювання

Вид роботи	Тематика	Дедлайн	Бали
Лабораторна робота 0	Вибір набору даних	16.02.2023	
Лабораторна робота 1	EDA	02.03.2023	10 балів
Лабораторна робота 2	Статистичне виведення, бутстреп	23.03.2023	10 балів
Лабораторна робота 3	Регресійний аналіз	06.04.2023	10 балів
Модульна контрольна робота		12.04.2022	40 балів
Лабораторна робота 4	Непараметричне виведення, PCA	04.05.2023	10 балів
Лабораторна робота 5	Беєсівське виведення	25.05.2023	10 балів
Лабораторна робота 6	Причиново-наслідкове виведення	06.06.2023	10 балів

- Разом можна набрати 100 балів
- Якщо оцінка не подобається — можна написати залік (1 теоретичне питання і 3 задачі)
- Умова **допуску до заліку** — здано (і зараховано!) всі ЛР та написано МКР
- Умова **атестації** — половина можливих балів

Рейтингова система оцінювання

Вид роботи	Тематика	Дедлайн	Бали
Лабораторна робота 0	Вибір набору даних	16.02.2023	
Лабораторна робота 1	EDA	02.03.2023	10 балів
Лабораторна робота 2	Статистичне виведення, бутстреп	23.03.2023	10 балів
Лабораторна робота 3	Регресійний аналіз	06.04.2023	10 балів
Модульна контрольна робота		12.04.2022	40 балів
Лабораторна робота 4	Непараметричне виведення, PCA	04.05.2023	10 балів
Лабораторна робота 5	Беєсівське виведення	25.05.2023	10 балів
Лабораторна робота 6	Причиново-наслідкове виведення	06.06.2023	10 балів

- Разом можна набрати 100 балів
- Якщо оцінка не подобається — можна написати залік (1 теоретичне питання і 3 задачі)
- Умова **допуску до заліку** — здано (і зараховано!) всі ЛР та написано МКР
- Умова **атестації** — половина можливих балів

Рейтингова система оцінювання

Вид роботи	Тематика	Дедлайн	Бали
Лабораторна робота 0	Вибір набору даних	16.02.2023	
Лабораторна робота 1	EDA	02.03.2023	10 балів
Лабораторна робота 2	Статистичне виведення, бутстреп	23.03.2023	10 балів
Лабораторна робота 3	Регресійний аналіз	06.04.2023	10 балів
Модульна контрольна робота		12.04.2022	40 балів
Лабораторна робота 4	Непараметричне виведення, PCA	04.05.2023	10 балів
Лабораторна робота 5	Беєсівське виведення	25.05.2023	10 балів
Лабораторна робота 6	Причиново-наслідкове виведення	06.06.2023	10 балів

- Разом можна набрати 100 балів
- Якщо оцінка не подобається — можна написати залік (1 теоретичне питання і 3 задачі)
- Умова **допуску до заліку** — здано (і зараховано!) всі ЛР та написано МКР
- Умова **атестації** — половина можливих балів

Рейтингова система оцінювання

Вид роботи	Тематика	Дедлайн	Бали
Лабораторна робота 0	Вибір набору даних	16.02.2023	
Лабораторна робота 1	EDA	02.03.2023	10 балів
Лабораторна робота 2	Статистичне виведення, бутстреп	23.03.2023	10 балів
Лабораторна робота 3	Регресійний аналіз	06.04.2023	10 балів
Модульна контрольна робота		12.04.2022	40 балів
Лабораторна робота 4	Непараметричне виведення, PCA	04.05.2023	10 балів
Лабораторна робота 5	Беєсівське виведення	25.05.2023	10 балів
Лабораторна робота 6	Причиново-наслідкове виведення	06.06.2023	10 балів

- Разом можна набрати 100 балів
- Якщо оцінка не подобається — можна написати залік (1 теоретичне питання і 3 задачі)
- Умова **допуску до заліку** — здано (і зараховано!) всі ЛР та написано МКР
- Умова **атестації** — половина можливих балів

Політика та контроль

- Обов’язковим є відвідання всіх лекцій
- Модульну контрольну роботу та залік кожний студент пише самостійно, консультування з третіми джерелами суворо заборонено
- За порушення правил академічної доброчесности під час МКР чи заліку будуть відповідні наслідки (анулювання результатів)
- Що стосується **лабораторних робіт**:

Політика та контроль

- Обов’язковим є відвідання всіх лекцій
- Модульну контрольну роботу та залік кожний студент пише самостійно, консультування з третіми джерелами суворо заборонено
- За порушення правил академічної доброчесности під час МКР чи заліку будуть відповідні наслідки (анулювання результатів)
- Що стосується **лабораторних робіт**:

Політика та контроль

- Обов'язковим є відвідання всіх лекцій
- Модульну контрольну роботу та залік кожний студент пише самостійно, консультування з третіми джерелами суворо заборонено
- За порушення правил академічної доброчесности під час МКР чи заліку будуть відповідні наслідки (анулювання результатів)
- Що стосується **лабораторних робіт**:
  - Усі лабораторні роботи виконуються на основі **одного й того ж** набору даних
  - Лабораторні роботи студенти виконують мовою R
  - Лабораторні роботи виконуються командами до 4 осіб включно
  - По кожній роботі команда повинна здати код, звіт і презентацію
  - Самі презентації відбуваються усно на лабораторному занятті або за домовленістю
  - На захисті повинні бути присутні всі члени команди
  - Протягом семестру кожний член команди повинен виступити принаймні двічі
  - Презентації записуються та викладаються у спільний доступ
  - На їх основі будуть формулюватися питання МКР

Політика та контроль

- Обов'язковим є відвідання всіх лекцій
- Модульну контрольну роботу та залік кожний студент пише самостійно, консультування з третіми джерелами суворо заборонено
- За порушення правил академічної доброчесности під час МКР чи заліку будуть відповідні наслідки (анулювання результатів)
- Що стосується **лабораторних робіт**:
  - Усі лабораторні роботи виконуються на основі **одного й того ж** набору даних
  - Лабораторні роботи студенти виконують мовою R
  - Лабораторні роботи виконуються **командами до 4 осіб включно**
  - По кожній роботі команда повинна здати **код**, **звіт** і **презентацію**
  - Самі презентації відбуваються **усно** на лабораторному занятті або за домовленістю
  - На захисті повинні бути присутні **всі** члени команди
  - Протягом семестру кожний член команди повинен виступити **принаймні тричі**
  - Презентації **записуються** та викладаються у **спільний доступ**
  - На їх основі будуть формулюватися **питання МКР**

Політика та контроль

- Обов'язковим є відвідання всіх лекцій
- Модульну контрольну роботу та залік кожний студент пише самостійно, консультування з третіми джерелами суворо заборонено
- За порушення правил академічної доброчесности під час МКР чи заліку будуть відповідні наслідки (анулювання результатів)
- Що стосується **лабораторних робіт**:
  - Усі лабораторні роботи виконуються на основі **одного й того ж** набору даних
  - Лабораторні роботи студенти виконують мовою R
  - Лабораторні роботи виконуються **командами до 4 осіб включно**
  - По кожній роботі команда повинна здати **код**, **звіт** і **презентацію**
  - Самі презентації відбуваються **усно** на лабораторному занятті або за домовленістю
  - На захисті повинні бути присутні **всі** члени команди
  - Протягом семестру кожний член команди повинен виступити **принаймні тричі**
  - Презентації **записуються** та викладаються у **спільний доступ**
  - На їх основі будуть формулюватися **питання МКР**

Політика та контроль

- Обов'язковим є відвідання всіх лекцій
- Модульну контрольну роботу та залік кожний студент пише самостійно, консультування з третіми джерелами суворо заборонено
- За порушення правил академічної доброчесности під час МКР чи заліку будуть відповідні наслідки (анулювання результатів)
- Що стосується **лабораторних робіт**:
  - Усі лабораторні роботи виконуються на основі **одного й того ж** набору даних
  - Лабораторні роботи студенти виконують мовою R
  - Лабораторні роботи виконуються **командами до 4 осіб включно**
  - По кожній роботі команда повинна здати **код**, **звіт** і **презентацію**
  - Самі презентації відбуваються **усно** на лабораторному занятті або за домовленістю
  - На захисті повинні бути присутні **всі** члени команди
  - Протягом семестру кожний член команди повинен виступити **принаймні тричі**
  - Презентації **записуються** та викладаються у **спільний доступ**
  - На їх основі будуть формулюватися **питання МКР**

Політика та контроль

- Обов'язковим є відвідання всіх лекцій
- Модульну контрольну роботу та залік кожний студент пише самостійно, консультування з третіми джерелами суворо заборонено
- За порушення правил академічної доброчесности під час МКР чи заліку будуть відповідні наслідки (анулювання результатів)
- Що стосується **лабораторних робіт**:
  - Усі лабораторні роботи виконуються на основі **одного й того ж** набору даних
  - Лабораторні роботи студенти виконують мовою R
  - Лабораторні роботи виконуються **командами до 4 осіб включно**
  - По кожній роботі команда повинна здати **код**, **звіт** і **презентацію**
  - Самі презентації відбуваються **усно** на лабораторному занятті або за домовленістю
  - На захисті повинні бути присутні **всі** члени команди
  - Протягом семестру кожний член команди повинен виступити **принаймні тричі**
  - Презентації **записуються** та викладаються у **спільний доступ**
  - На їх основі будуть формулюватися **питання МКР**

Політика та контроль

- Обов'язковим є відвідання всіх лекцій
- Модульну контрольну роботу та залік кожний студент пише самостійно, консультування з третіми джерелами суворо заборонено
- За порушення правил академічної доброчесности під час МКР чи заліку будуть відповідні наслідки (анулювання результатів)
- Що стосується **лабораторних робіт**:
  - Усі лабораторні роботи виконуються на основі **одного й того ж** набору даних
  - Лабораторні роботи студенти виконують мовою R
  - Лабораторні роботи виконуються **командами до 4 осіб включно**
  - По кожній роботі команда повинна здати **код**, **звіт** і **презентацію**
  - Самі презентації відбуваються **усно** на лабораторному занятті або за домовленістю
  - На захисті повинні бути присутні **всі** члени команди
  - Протягом семестру кожний член команди повинен виступити **принаймні тричі**
  - Презентації **записуються** та викладаються у **спільний доступ**
  - На їх основі будуть формулюватися **питання МКР**

Політика та контроль

- Обов'язковим є відвідання всіх лекцій
- Модульну контрольну роботу та залік кожний студент пише самостійно, консультування з третіми джерелами суворо заборонено
- За порушення правил академічної доброчесности під час МКР чи заліку будуть відповідні наслідки (анулювання результатів)
- Що стосується **лабораторних робіт**:
  - Усі лабораторні роботи виконуються на основі **одного й того ж** набору даних
  - Лабораторні роботи студенти виконують мовою R
  - Лабораторні роботи виконуються **командами до 4 осіб включно**
  - По кожній роботі команда повинна здати **код**, **звіт** і **презентацію**
  - Самі презентації відбуваються **усно** на лабораторному занятті або за домовленістю
  - На захисті повинні бути присутні **всі** члени команди
  - Протягом семестру кожний член команди повинен виступити **принаймні тричі**
  - Презентації **записуються** та викладаються у **спільний доступ**
  - На їх основі будуть формулюватися **питання МКР**

Політика та контроль

- Обов'язковим є відвідання всіх лекцій
- Модульну контрольну роботу та залік кожний студент пише самостійно, консультування з третіми джерелами суворо заборонено
- За порушення правил академічної доброчесности під час МКР чи заліку будуть відповідні наслідки (анулювання результатів)
- Що стосується **лабораторних робіт**:
  - Усі лабораторні роботи виконуються на основі **одного й того ж** набору даних
  - Лабораторні роботи студенти виконують мовою R
  - Лабораторні роботи виконуються **командами до 4 осіб включно**
  - По кожній роботі команда повинна здати **код**, **звіт** і **презентацію**
  - Самі презентації відбуваються **усно** на лабораторному занятті або за домовленістю
  - На захисті повинні бути присутні **всі** члени команди
  - Протягом семестру кожний член команди повинен виступити **принаймні тричі**
  - Презентації **записуються** та викладаються у **спільний доступ**
  - На їх основі будуть формулюватися **питання МКР**

Політика та контроль

- Обов'язковим є відвідання всіх лекцій
- Модульну контрольну роботу та залік кожний студент пише самостійно, консультування з третіми джерелами суворо заборонено
- За порушення правил академічної доброчесности під час МКР чи заліку будуть відповідні наслідки (анулювання результатів)
- Що стосується **лабораторних робіт**:
  - Усі лабораторні роботи виконуються на основі **одного й того ж** набору даних
  - Лабораторні роботи студенти виконують мовою R
  - Лабораторні роботи виконуються **командами до 4 осіб включно**
  - По кожній роботі команда повинна здати **код**, **звіт** і **презентацію**
  - Самі презентації відбуваються **усно** на лабораторному занятті або за домовленістю
  - На захисті повинні бути присутні **всі** члени команди
  - Протягом семестру кожний член команди повинен виступити **принаймні тричі**
  - Презентації **записуються** та викладаються у **спільний доступ**
  - На їх основі будуть формулюватися **питання МКР**

Політика та контроль

- Обов'язковим є відвідання всіх лекцій
- Модульну контрольну роботу та залік кожний студент пише самостійно, консультування з третіми джерелами суворо заборонено
- За порушення правил академічної доброчесности під час МКР чи заліку будуть відповідні наслідки (анулювання результатів)
- Що стосується **лабораторних робіт**:
  - Усі лабораторні роботи виконуються на основі **одного й того ж** набору даних
  - Лабораторні роботи студенти виконують мовою R
  - Лабораторні роботи виконуються **командами до 4 осіб включно**
  - По кожній роботі команда повинна здати **код**, **звіт** і **презентацію**
  - Самі презентації відбуваються **усно** на лабораторному занятті або за домовленістю
  - На захисті повинні бути присутні **всі** члени команди
  - Протягом семестру кожний член команди повинен виступити **принаймні тричі**
  - Презентації **записуються** та викладаються у **спільний доступ**
  - На їх основі будуть формулюватися **питання МКР**

Політика та контроль

- Обов'язковим є відвідання всіх лекцій
- Модульну контрольну роботу та залік кожний студент пише самостійно, консультування з третіми джерелами суворо заборонено
- За порушення правил академічної доброчесности під час МКР чи заліку будуть відповідні наслідки (анулювання результатів)
- Що стосується **лабораторних робіт**:
  - Усі лабораторні роботи виконуються на основі **одного й того ж** набору даних
  - Лабораторні роботи студенти виконують мовою R
  - Лабораторні роботи виконуються **командами до 4 осіб включно**
  - По кожній роботі команда повинна здати **код**, **звіт** і **презентацію**
  - Самі презентації відбуваються **усно** на лабораторному занятті або за домовленістю
  - На захисті повинні бути присутні **всі** члени команди
  - Протягом семестру кожний член команди повинен виступити **принаймні тричі**
  - Презентації **записуються** та викладаються у **спільний доступ**
  - На їх основі будуть формулюватися **питання МКР**

Поради щодо вивчення курсу

- Це не є курс програмування на R!
- R — це просто *один із* можливих інструментів для аналізу даних (поряд із Python, MATLAB, Stata тощо)
- Із таким же успіхом можна було б вичитати цей курс на будь-якій із цих альтернатив
- Студенти, які вже вивчили Python та інші мови програмування, можуть опанувати R дуже швидко
- «Аналіз даних» — доволі розмите поняття, тому наповнення курсу може бути зовсім різне
- Я пропоную своє бачення, що важливо розглянути

Поради щодо вивчення курсу

- Це не є курс програмування на R!
- R — це просто *один із* можливих інструментів для аналізу даних (поряд із Python, MATLAB, Stata тощо)
- Із таким же успіхом можна було б вичитати цей курс на будь-якій із цих альтернатив
- Студенти, які вже вивчили Python та інші мови програмування, можуть опанувати R дуже швидко
- «Аналіз даних» — доволі розмите поняття, тому наповнення курсу може бути зовсім різне
- Я пропоную своє бачення, що важливо розглянути

Поради щодо вивчення курсу

- Це не є курс програмування на R!
- R — це просто *один із* можливих інструментів для аналізу даних (поряд із Python, MATLAB, Stata тощо)
- Із таким же успіхом можна було б вичитати цей курс на будь-якій із цих альтернатив
- Студенти, які вже вивчили Python та інші мови програмування, можуть опанувати R дуже швидко
- «Аналіз даних» — доволі розмите поняття, тому наповнення курсу може бути зовсім різне
- Я пропоную своє бачення, що важливо розглянути

Поради щодо вивчення курсу

- Це не є курс програмування на R!
- R — це просто *один із* можливих інструментів для аналізу даних (поряд із Python, MATLAB, Stata тощо)
- Із таким же успіхом можна було б вичитати цей курс на будь-якій із цих альтернатив
- Студенти, які вже вивчили Python та інші мови програмування, можуть опанувати R дуже швидко
- «Аналіз даних» — доволі розмите поняття, тому наповнення курсу може бути зовсім різне
- Я пропоную своє бачення, що важливо розглянути

Поради щодо вивчення курсу

- Це не є курс програмування на R!
- R — це просто *один із* можливих інструментів для аналізу даних (поряд із Python, MATLAB, Stata тощо)
- Із таким же успіхом можна було б вичитати цей курс на будь-якій із цих альтернатив
- Студенти, які вже вивчили Python та інші мови програмування, можуть опанувати R дуже швидко
- «Аналіз даних» — доволі розмите поняття, тому наповнення курсу може бути зовсім різне
- Я пропоную своє бачення, що важливо розглянути

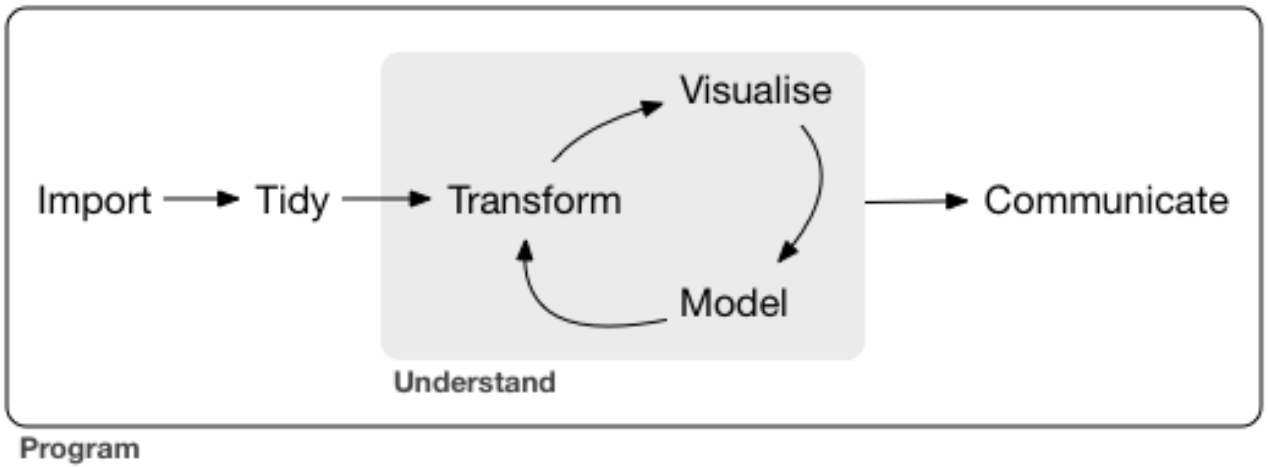
Поради щодо вивчення курсу

- Це не є курс програмування на R!
- R — це просто *один із* можливих інструментів для аналізу даних (поряд із Python, MATLAB, Stata тощо)
- Із таким же успіхом можна було б вичитати цей курс на будь-якій із цих альтернатив
- Студенти, які вже вивчили Python та інші мови програмування, можуть опанувати R дуже швидко
- «Аналіз даних» — доволі розмите поняття, тому наповнення курсу може бути зовсім різне
- Я пропоную своє бачення, що важливо розглянути

План лекції

Життєвий цикл аналізу даних (Data Science lifecycle) (1)

- Одна з популярних версій життєвого циклу аналізу даних [1]:

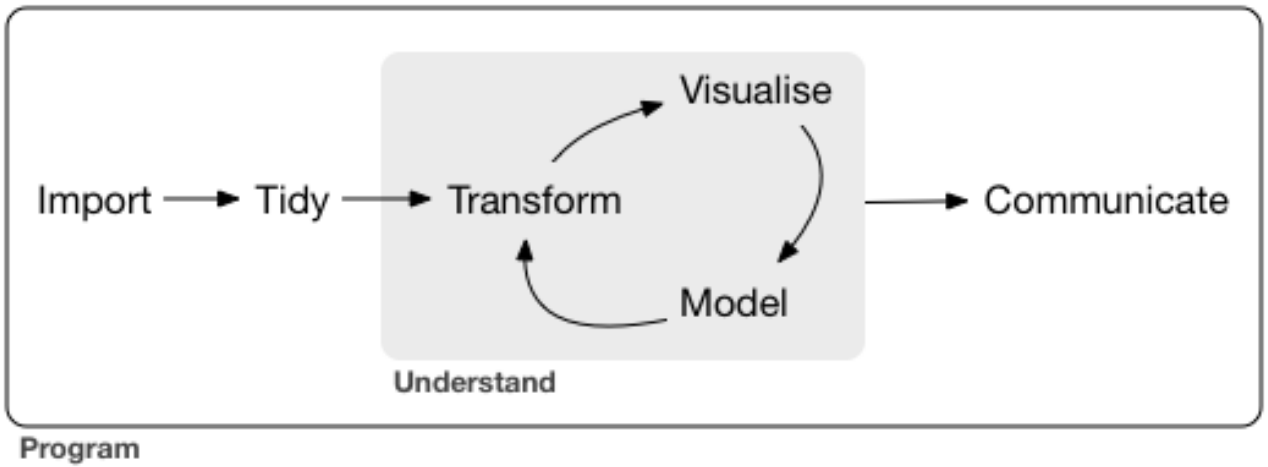


- На самому початку потрібно з'ясувати, **що** і **з якою метою** будемо досліджувати
- Потім потрібно **зібрати відповідні дані**
- Дані потрібно **імпортувати** (з файлу, бази даних, через Інтернет тощо)
- Після цього дані потрібно **почистити** та **перетворити**

[1] https://r4ds.had.co.nz/explore-intro.html

Життєвий цикл аналізу даних (Data Science lifecycle) (1)

- Одна з популярних версій життєвого циклу аналізу даних [1]:

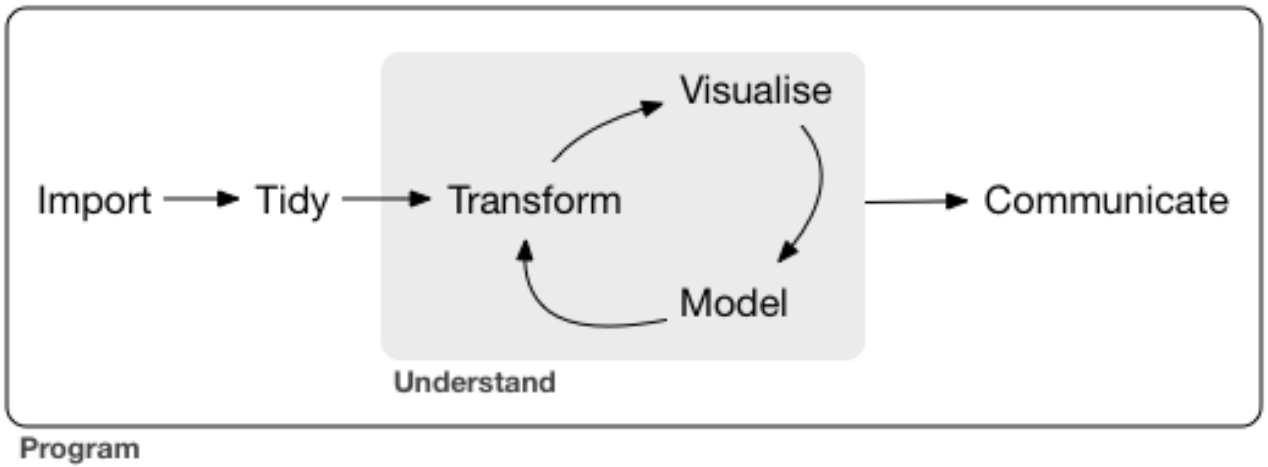


- На самому початку потрібно з'ясувати, **що** і **з якою метою** будемо досліджувати
- Потім потрібно **зібрати відповідні дані**
- Дані потрібно **імпортувати** (з файлу, бази даних, через Інтернет тощо)
- Після цього дані потрібно **почистити** та **перетворити**

[1] https://r4ds.had.co.nz/explore-intro.html

Життєвий цикл аналізу даних (Data Science lifecycle) (1)

- Одна з популярних версій життєвого циклу аналізу даних [1]:

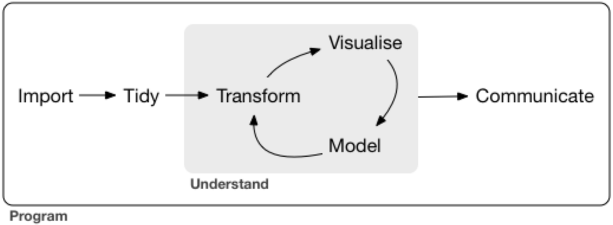


- На самому початку потрібно з'ясувати, **що** і **з якою метою** будемо досліджувати
- Потім потрібно **зібрати відповідні дані**
- Дані потрібно **імпортувати** (з файлу, бази даних, через Інтернет тощо)
- Після цього дані потрібно **почистити** та **перетворити**

[1] https://r4ds.had.co.nz/explore-intro.html

Життєвий цикл аналізу даних (Data Science lifecycle) (1)

- Одна з популярних версій життєвого циклу аналізу даних [1]:

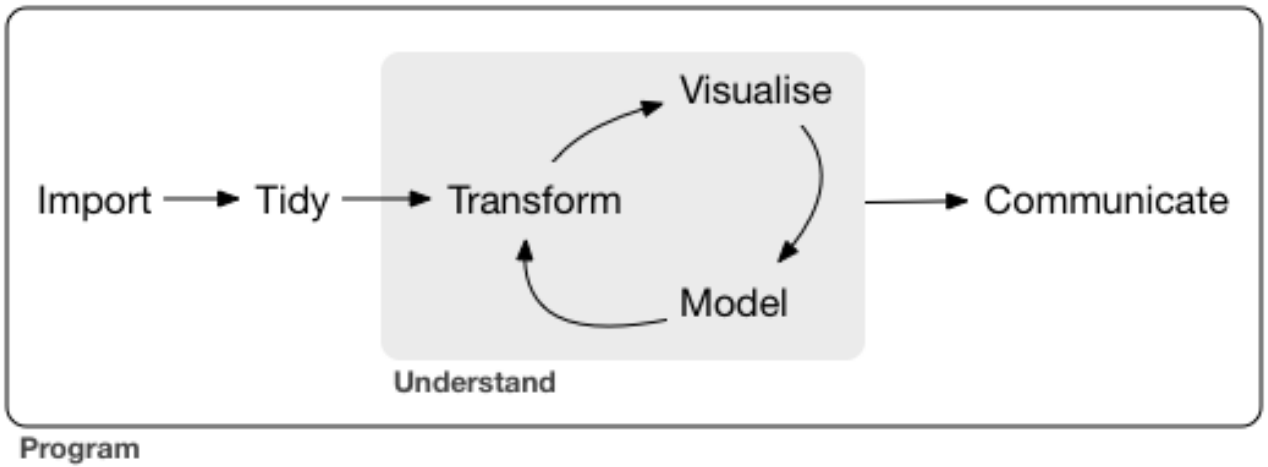


- На самому початку потрібно з'ясувати, **що** і **з якою метою** будемо досліджувати
- Потім потрібно **зібрати відповідні дані**
- Дані потрібно **імпортувати** (з файлу, бази даних, через Інтернет тощо)
- Після цього дані потрібно **почистити** та **перетворити**
  - Реструктуризація даних для приведення їх в охайний вигляд
  - Видалення непотрібних рядків і стовпців
  - Перейменування, перекодування і т.п.

[1] https://r4ds.had.co.nz/explore-intro.html

Життєвий цикл аналізу даних (Data Science lifecycle) (1)

- Одна з популярних версій життєвого циклу аналізу даних [1]:

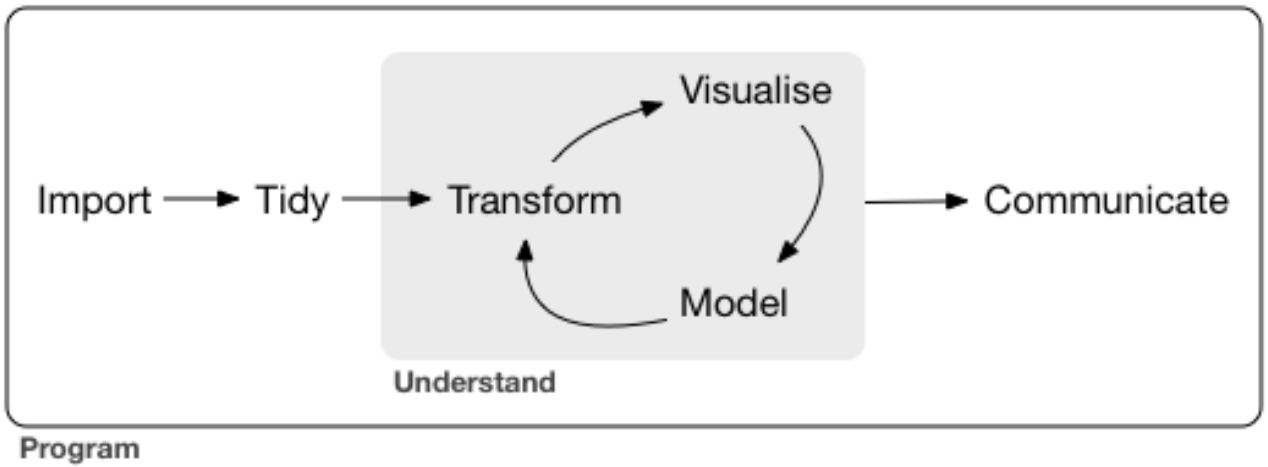


- На самому початку потрібно з'ясувати, **що** і **з якою метою** будемо досліджувати
- Потім потрібно **зібрати відповідні дані**
- Дані потрібно **імпортувати** (з файлу, бази даних, через Інтернет тощо)
- Після цього дані потрібно **почистити** та **перетворити**
  - Реструктуризація даних для приведення їх в охайний вигляд
  - Видалення непотрібних рядків і стовпців
  - Перейменування, перекодування і т.п.
  - Це часто є найдовший і найнудніший, але гранично важливий етап
  - Його часто називають **data wrangling** (дослівно «суперечка з даними»)

[1] https://r4ds.had.co.nz/explore-intro.html

Життєвий цикл аналізу даних (Data Science lifecycle) (1)

- Одна з популярних версій життєвого циклу аналізу даних [1]:

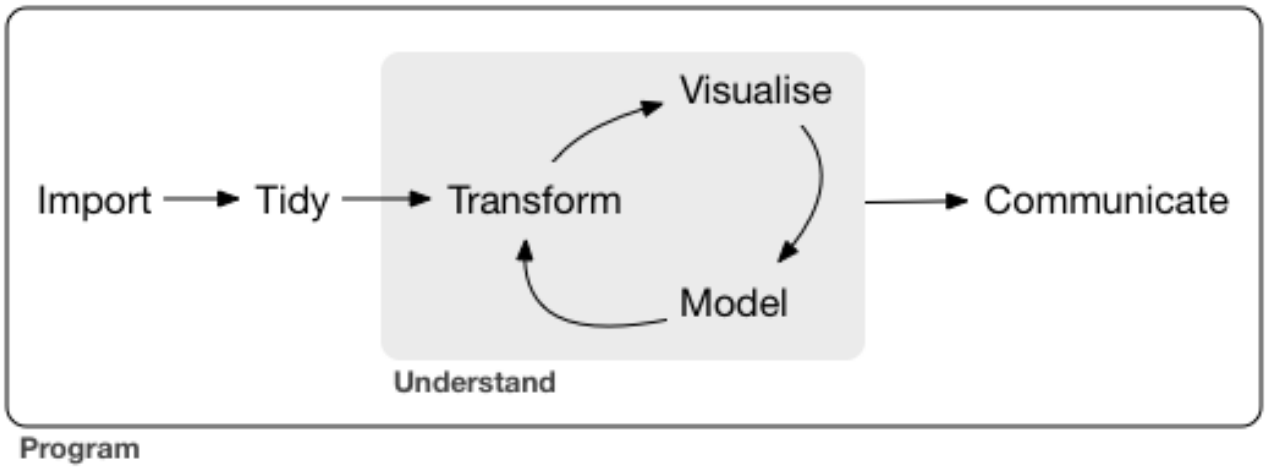


- На самому початку потрібно з'ясувати, **що** і **з якою метою** будемо досліджувати
- Потім потрібно **зібрати відповідні дані**
- Дані потрібно **імпортувати** (з файлу, бази даних, через Інтернет тощо)
- Після цього дані потрібно **почистити** та **перетворити**
  - Реструктуризація даних для приведення їх в охайний вигляд
  - Видалення непотрібних рядків і стовпців
  - Перейменування, перекодування і т.п.
  - Це часто є найдовший і найнудніший, але гранично важливий етап
  - Його часто називають **data wrangling** (дослівно «суперечка з даними»)

[1] https://r4ds.had.co.nz/explore-intro.html

Життєвий цикл аналізу даних (Data Science lifecycle) (1)

- Одна з популярних версій життєвого циклу аналізу даних [1]:

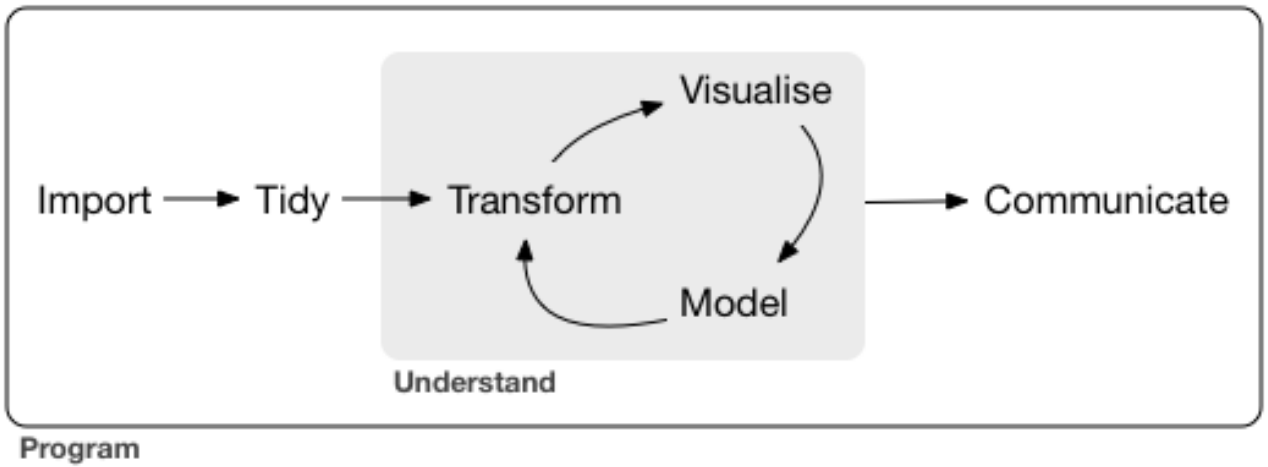


- На самому початку потрібно з'ясувати, **що** і **з якою метою** будемо досліджувати
- Потім потрібно **зібрати відповідні дані**
- Дані потрібно **імпортувати** (з файлу, бази даних, через Інтернет тощо)
- Після цього дані потрібно **почистити** та **перетворити**
  - Реструктуризація даних для приведення їх в охайний вигляд
  - Видалення непотрібних рядків і стовпців
  - Перейменування, перекодування і т.п.
  - Це часто є найдовший і найнудніший, але гранично важливий етап
  - Його часто називають **data wrangling** (дослівно «суперечка з даними»)

[1] https://r4ds.had.co.nz/explore-intro.html

Життєвий цикл аналізу даних (Data Science lifecycle) (1)

- Одна з популярних версій життєвого циклу аналізу даних [1]:

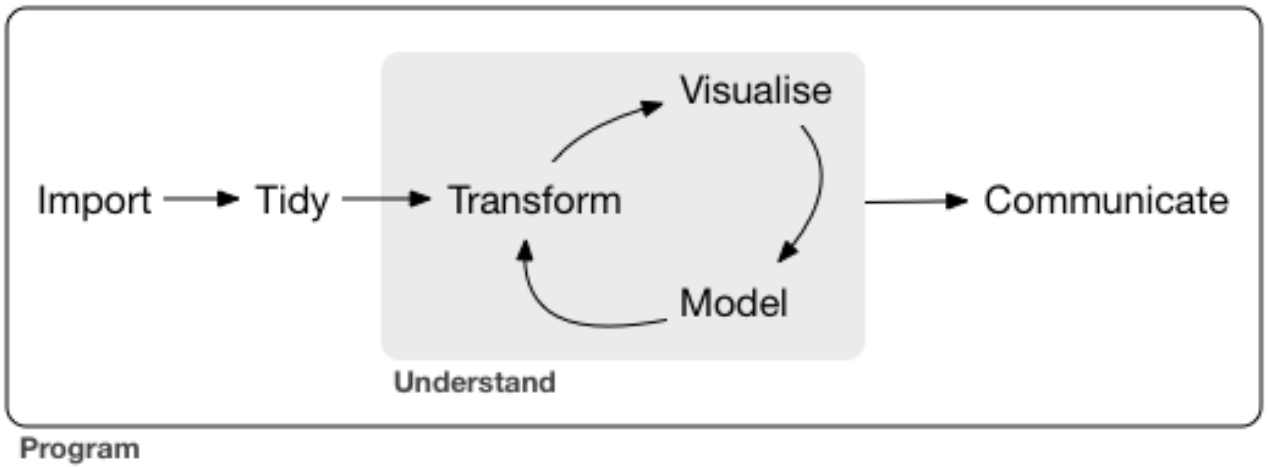


- На самому початку потрібно з'ясувати, **що** і **з якою метою** будемо досліджувати
- Потім потрібно **зібрати відповідні дані**
- Дані потрібно **імпортувати** (з файлу, бази даних, через Інтернет тощо)
- Після цього дані потрібно **почистити** та **перетворити**
  - Реструктуризація даних для приведення їх в охайний вигляд
  - Видалення непотрібних рядків і стовпців
  - Перейменування, перекодування і т.п.
  - Це часто є найдовший і найнудніший, але гранично важливий етап
  - Його часто називають **data wrangling** (дослівно «суперечка з даними»)

[1] https://r4ds.had.co.nz/explore-intro.html

Життєвий цикл аналізу даних (Data Science lifecycle) (1)

- Одна з популярних версій життєвого циклу аналізу даних [1]:

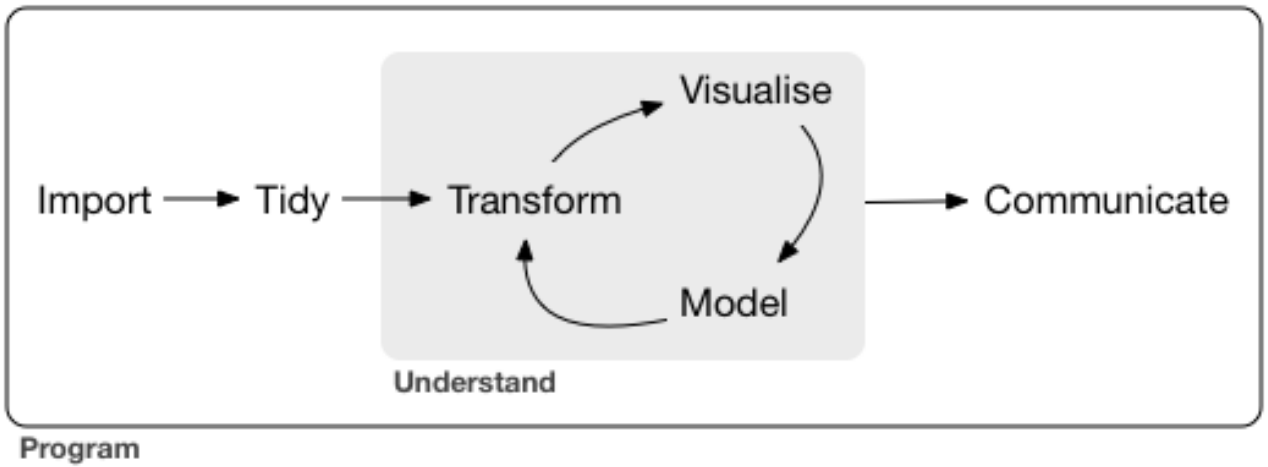


- На самому початку потрібно з'ясувати, **що** і **з якою метою** будемо досліджувати
- Потім потрібно **зібрати відповідні дані**
- Дані потрібно **імпортувати** (з файлу, бази даних, через Інтернет тощо)
- Після цього дані потрібно **почистити** та **перетворити**
  - Реструктуризація даних для приведення їх в охайний вигляд
  - Видалення непотрібних рядків і стовпців
  - Перейменування, перекодування і т.п.
  - Це часто є найдовший і найнудніший, але гранично важливий етап
  - Його часто називають **data wrangling** (дослівно «суперечка з даними»)

[1] https://r4ds.had.co.nz/explore-intro.html

Життєвий цикл аналізу даних (Data Science lifecycle) (1)

- Одна з популярних версій життєвого циклу аналізу даних [1]:



- На самому початку потрібно з'ясувати, **що** і **з якою метою** будемо досліджувати
- Потім потрібно **зібрати відповідні дані**
- Дані потрібно **імпортувати** (з файлу, бази даних, через Інтернет тощо)
- Після цього дані потрібно **почистити** та **перетворити**
  - Реструктуризація даних для приведення їх в охайний вигляд
  - Видалення непотрібних рядків і стовпців
  - Перейменування, перекодування і т.п.
  - Це часто є найдовший і найнудніший, але гранично важливий етап
  - Його часто називають **data wrangling** (дослівно «суперечка з даними»)

[1] https://r4ds.had.co.nz/explore-intro.html

Життєвий цикл аналізу даних (Data Science lifecycle) (2)

- Одна з популярних версій життєвого циклу аналізу даних [2]:

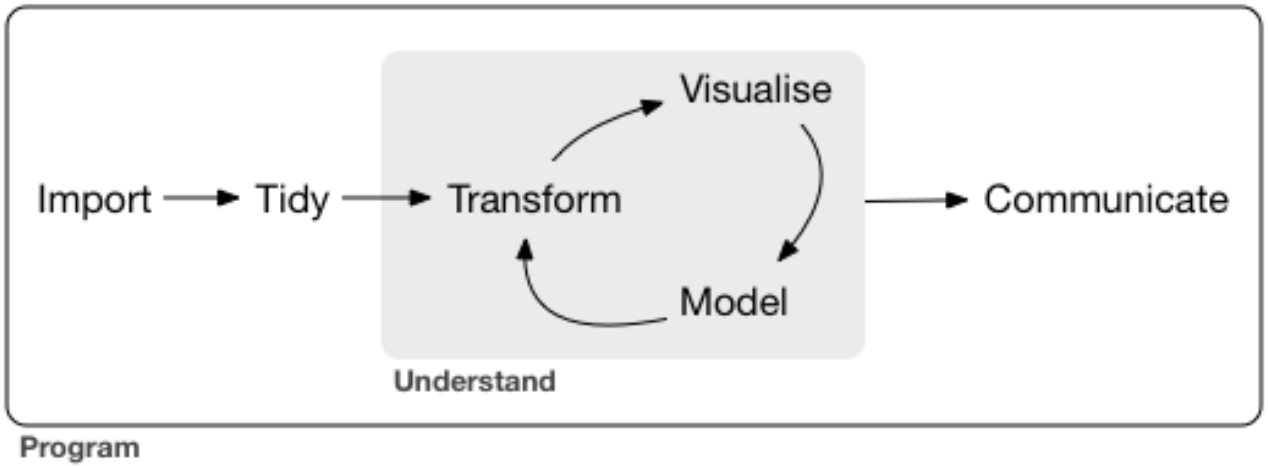


Рис. 2.1: Життєвий цикл аналізу даних

- Після попередньої підготовки даних їх можна **візуалізувати**
  - Корисно для формування гіпотез для дальшої статистичної перевірки
  - Важливо для оформлення результатів аналізу даних у звітах та презентаціях
- На основі теорії, попереднього аналізу даних чи їх візуалізації можна будувати **моделі** (статистичні, так чи інакше), які описують ці дані
- Усі результати аналізу даних потрібно правильно **скомунікувати** у вигляді звіту, презентації тощо

[2] https://r4ds.had.co.nz/explore-intro.html

Життєвий цикл аналізу даних (Data Science lifecycle) (2)

- Одна з популярних версій життєвого циклу аналізу даних [2]:

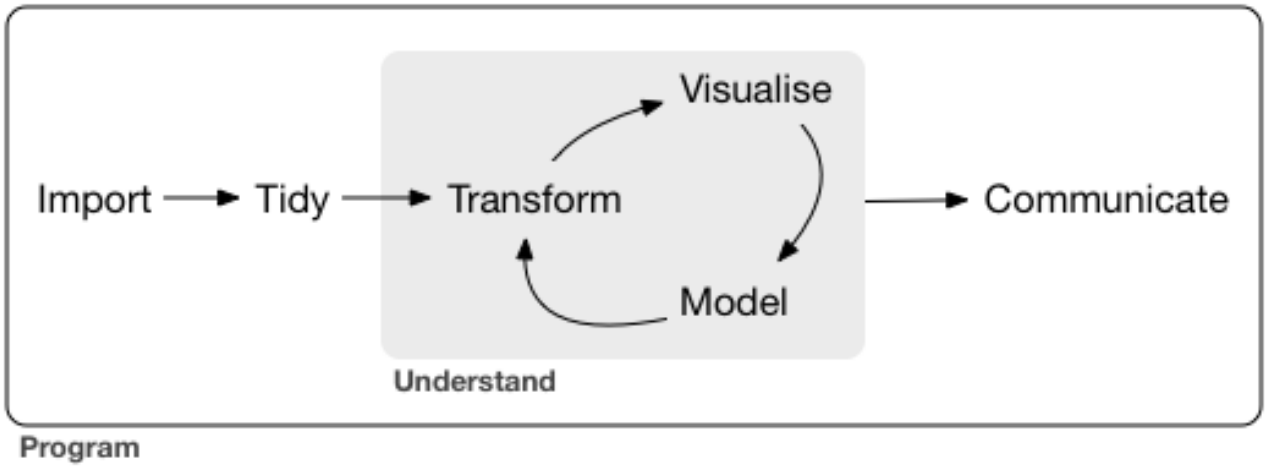


Рис. 2.1: Життєвий цикл аналізу даних

- Після попередньої підготовки даних їх можна **візуалізувати**
  - Корисно для формування гіпотез для дальшої статистичної перевірки
  - Важливо для оформлення результатів аналізу даних у звітах та презентаціях
- На основі теорії, попереднього аналізу даних чи їх візуалізації можна будувати **моделі** (статистичні, так чи інакше), які описують ці дані
- Усі результати аналізу даних потрібно правильно **скомунікувати** у вигляді звіту, презентації тощо

[2] https://r4ds.had.co.nz/explore-intro.html

Життєвий цикл аналізу даних (Data Science lifecycle) (2)

- Одна з популярних версій життєвого циклу аналізу даних [2]:

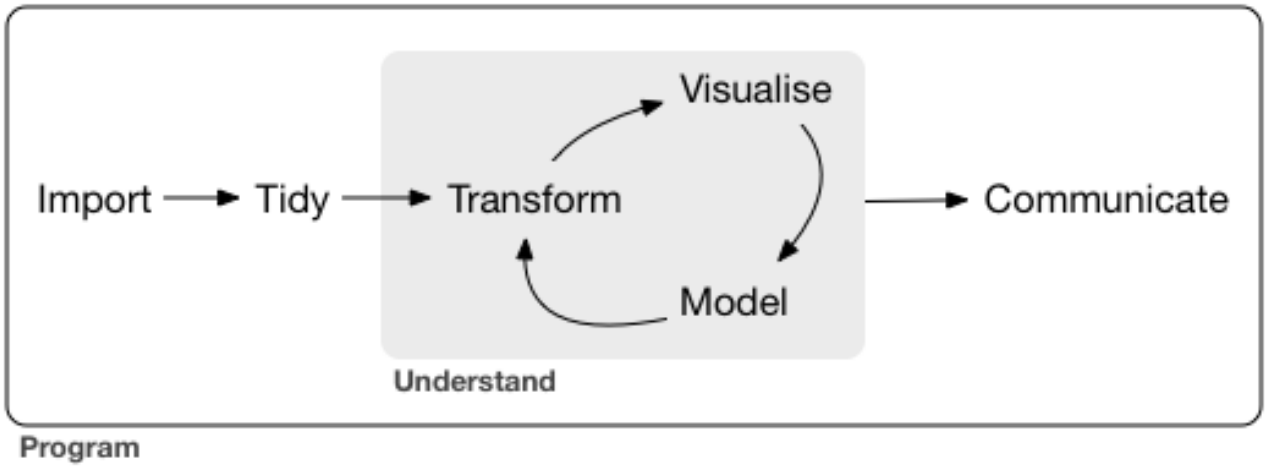


Рис. 2.1: Життєвий цикл аналізу даних

- Після попередньої підготовки даних їх можна **візуалізувати**
  - Корисно для формування гіпотез для дальшої статистичної перевірки
  - Важливо для оформлення результатів аналізу даних у звітах та презентаціях
- На основі теорії, попереднього аналізу даних чи їх візуалізації можна будувати **моделі** (статистичні, так чи інакше), які описують ці дані
- Усі результати аналізу даних потрібно правильно **скомунікувати** у вигляді звіту, презентації тощо

[2] https://r4ds.had.co.nz/explore-intro.html

Життєвий цикл аналізу даних (Data Science lifecycle) (2)

- Одна з популярних версій життєвого циклу аналізу даних [2]:

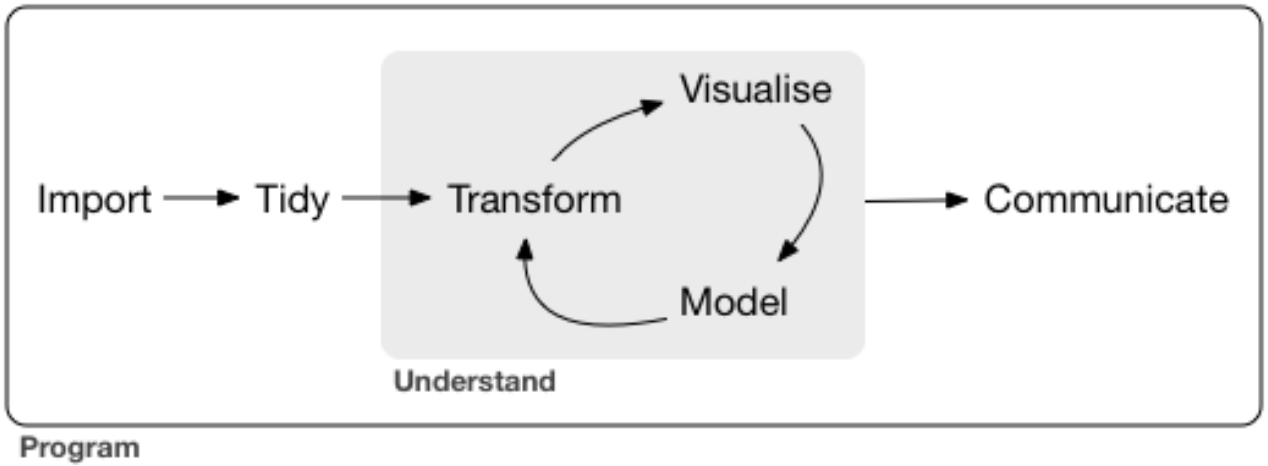


Рис. 2.1: Життєвий цикл аналізу даних

- Після попередньої підготовки даних їх можна **візуалізувати**
  - Корисно для формування гіпотез для дальшої статистичної перевірки
  - Важливо для оформлення результатів аналізу даних у звітах та презентаціях
- На основі теорії, попереднього аналізу даних чи їх візуалізації можна будувати **моделі** (статистичні, так чи інакше), які описують ці дані
  - Цьому в основному й присвячено конкретно наш курс
- Усі результати аналізу даних потрібно правильно **скомунікувати** у вигляді звіту, презентації тощо

[2] https://r4ds.had.co.nz/explore-intro.html

Життєвий цикл аналізу даних (Data Science lifecycle) (2)

- Одна з популярних версій життєвого циклу аналізу даних [2]:

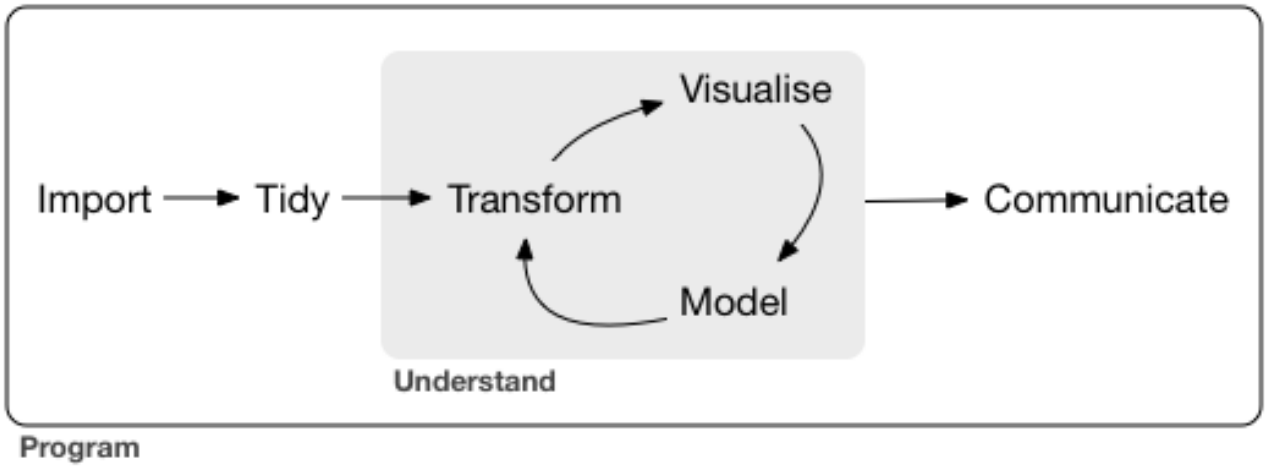


Рис. 2.1: Життєвий цикл аналізу даних

- Після попередньої підготовки даних їх можна **візуалізувати**
  - Корисно для формування гіпотез для дальшої статистичної перевірки
  - Важливо для оформлення результатів аналізу даних у звітах та презентаціях
- На основі теорії, попереднього аналізу даних чи їх візуалізації можна будувати **моделі** (статистичні, так чи інакше), які описують ці дані
  - Цьому в основному й присвячено конкретно **наш курс**
- Усі результати аналізу даних потрібно правильно **скомунікувати** у вигляді звіту, презентації тощо

[2] https://r4ds.had.co.nz/explore-intro.html

Життєвий цикл аналізу даних (Data Science lifecycle) (2)

- Одна з популярних версій життєвого циклу аналізу даних [2]:

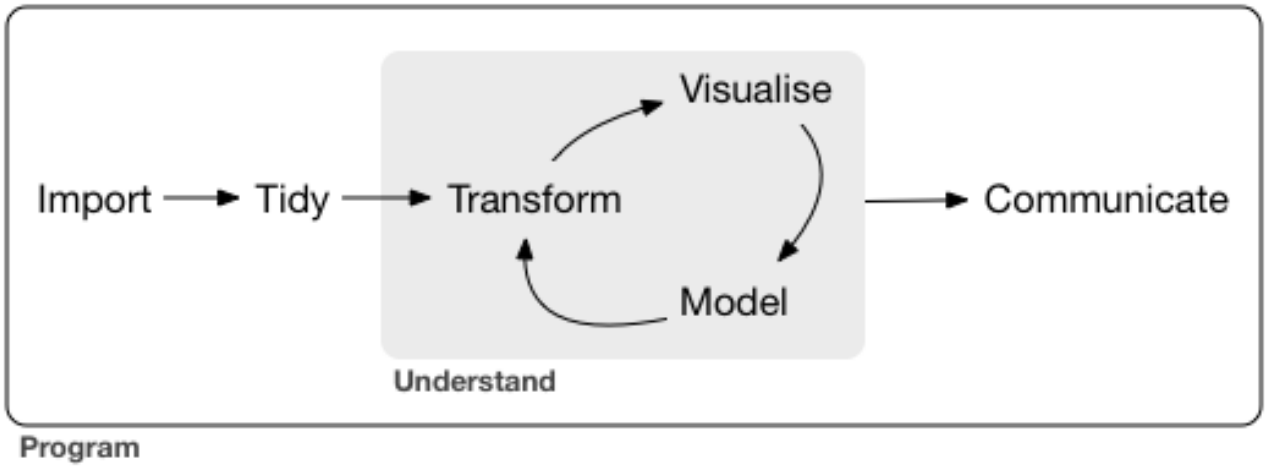


Рис. 2.1: Життєвий цикл аналізу даних

- Після попередньої підготовки даних їх можна **візуалізувати**
  - Корисно для формування гіпотез для дальшої статистичної перевірки
  - Важливо для оформлення результатів аналізу даних у звітах та презентаціях
- На основі теорії, попереднього аналізу даних чи їх візуалізації можна будувати **моделі** (статистичні, так чи інакше), які описують ці дані
  - Цьому в основному й присвячено конкретно **наш курс**
- Усі результати аналізу даних потрібно правильно **скомунікувати** у вигляді звіту, презентації тощо
  - Цим ви займатиметеся в рамках підготовки до захисту своїх лабораторних робіт

[2] https://r4ds.had.co.nz/explore-intro.html

Життєвий цикл аналізу даних (Data Science lifecycle) (2)

- Одна з популярних версій життєвого циклу аналізу даних [2]:

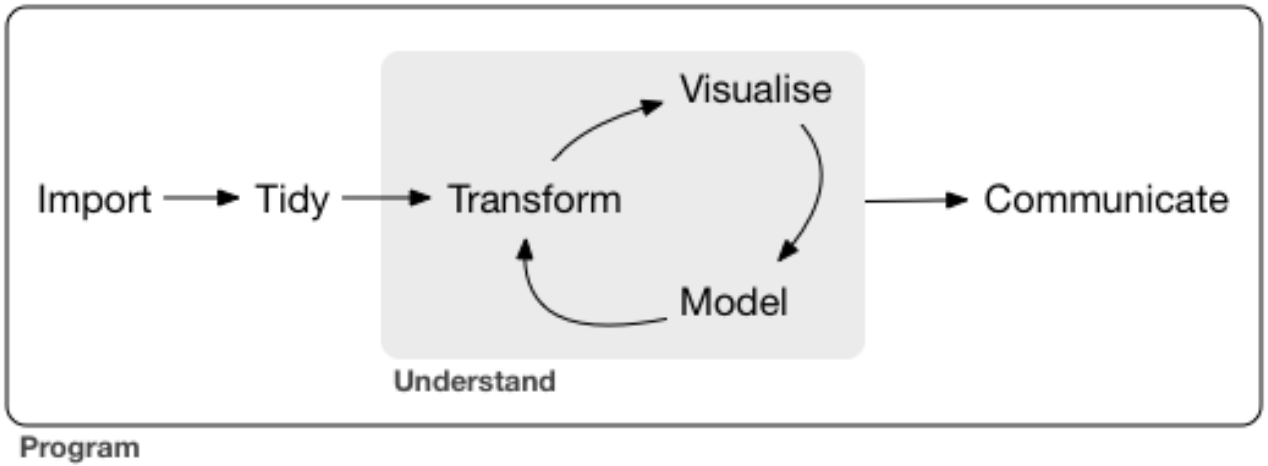


Рис. 2.1: Життєвий цикл аналізу даних

- Після попередньої підготовки даних їх можна **візуалізувати**
  - Корисно для формування гіпотез для дальшої статистичної перевірки
  - Важливо для оформлення результатів аналізу даних у звітах та презентаціях
- На основі теорії, попереднього аналізу даних чи їх візуалізації можна будувати **моделі** (статистичні, так чи інакше), які описують ці дані
  - Цьому в основному й присвячено конкретно **наш курс**
- Усі результати аналізу даних потрібно правильно **скомунікувати** у вигляді звіту, презентації тощо
  - Цим ви займатиметеся в рамках підготовки до захисту своїх **лабораторних робіт**

[2] https://r4ds.had.co.nz/explore-intro.html

Життєвий цикл аналізу даних (Data Science lifecycle) (2)

- Одна з популярних версій життєвого циклу аналізу даних [2]:

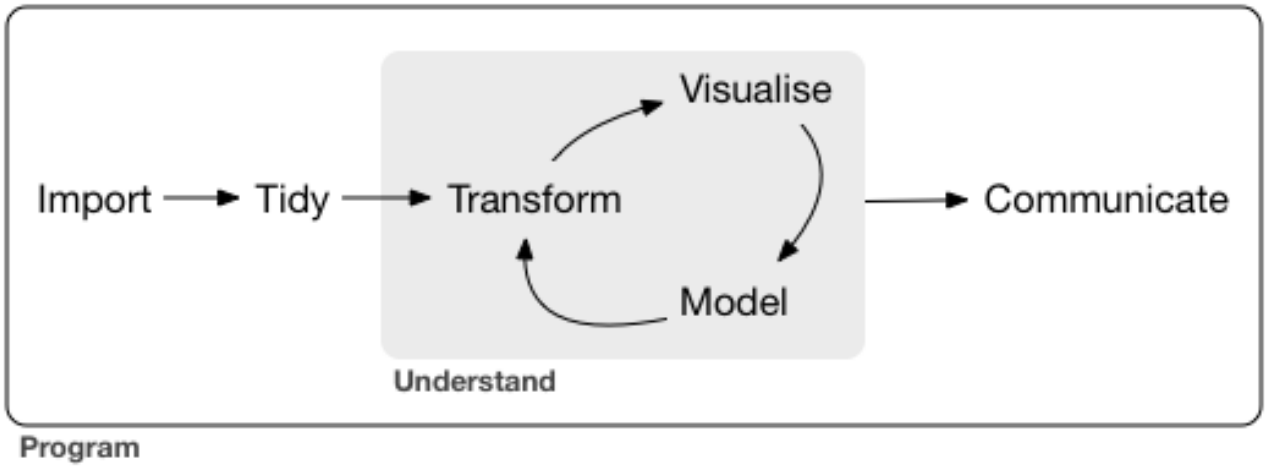


Рис. 2.1: Життєвий цикл аналізу даних

- Після попередньої підготовки даних їх можна **візуалізувати**
  - Корисно для формування гіпотез для дальшої статистичної перевірки
  - Важливо для оформлення результатів аналізу даних у звітах та презентаціях
- На основі теорії, попереднього аналізу даних чи їх візуалізації можна будувати **моделі** (статистичні, так чи інакше), які описують ці дані
  - Цьому в основному й присвячено конкретно **наш курс**
- Усі результати аналізу даних потрібно правильно **скомунікувати** у вигляді звіту, презентації тощо
  - Цим ви займатиметеся в рамках підготовки до захисту своїх **лабораторних робіт**

[2] https://r4ds.had.co.nz/explore-intro.html

Як ставити дослідницьке питання

- Для проведення доброго аналізу даних потрібно думати більше, ніж робити
- Особливо важливо правильно сформулювати дослідницьке питання
- Характеристики доброго дослідницького питання:
  - Воно повинно становити цікавість для аудиторії (принаймні в загальному контексту)
  - Воно не повинно мати відповідей у літературі
  - Воно повинно спиратися на здоровий глузд
  - Відповідь на нього можна дістати (як у гіпотетичному сенсі, так і з практичних міркувань на кшталт наявності даних)
  - Воно повинно бути конкретизовано настільки, щоб на нього можна було дістати вичерпну відповідь
- Нарешті, потрібно заздалегідь передбачати, якого роду відповіді на питання можуть бути, і як їх інтерпретувати
- Розгляньмо основні типи питань, які можуть виникнути

Як ставити дослідницьке питання

- Для проведення доброго аналізу даних потрібно думати більше, ніж робити
- Особливо важливо правильно сформулювати дослідницьке питання
- Характеристики доброго дослідницького питання:
  - Воно повинно становити **цікавість** для авдиторії (природа якої залежить від контексту)
  - Воно не повинно мати відповідей у літературі
  - Воно повинно спиратися на здоровий глузд
  - Відповіді на нього можна дістати (як у теоретичному сенсі, так і практичному, міркуючи на кшталт наявності даних)
  - Воно повинно бути конкретизованим настільки, щоб на нього можна було дати емпіричну відповідь
- Нарешті, потрібно заздалегідь передбачати, якого роду відповіді на питання можуть бути, і як їх інтерпретувати
- Розгляньмо основні типи питань, які можуть виникнути

Як ставити дослідницьке питання

- Для проведення доброго аналізу даних потрібно думати більше, ніж робити
- Особливо важливо правильно сформулювати дослідницьке питання
- Характеристики доброго дослідницького питання:
  - Воно повинно становити **цікавість** для авдиторії (природа якої залежить від контексту)
  - Воно не повинно мати відповідей у літературі
  - Воно повинно спиратися на здоровий глузд
  - Відповідь на нього можна дістати (як у гіпотетичному сенсі, так і з практичних міркувань на кшталт наявности даних)
  - Воно повинно бути конкретизовано настільки, щоб на нього можна було дістати вичерпну відповідь
- Нарешті, потрібно заздалегідь передбачати, якого роду відповіді на питання можуть бути, і як їх інтерпретувати
- Розгляньмо основні типи питань, які можуть виникнути

Як ставити дослідницьке питання

- Для проведення доброго аналізу даних потрібно думати більше, ніж робити
- Особливо важливо правильно сформулювати дослідницьке питання
- Характеристики доброго дослідницького питання:
  - Воно повинно становити **цікавість** для авдиторії (природа якої залежить від контексту)
  - Воно не повинно мати відповідей у літературі
  - Воно повинно спиратися на здоровий глузд
  - Відповідь на нього можна дістати (як у гіпотетичному сенсі, так і з практичних міркувань на кшталт наявности даних)
  - Воно повинно бути конкретизовано настільки, щоб на нього можна було дістати вичерпну відповідь
- Нарешті, потрібно заздалегідь передбачати, якого роду відповіді на питання можуть бути, і як їх інтерпретувати
- Розгляньмо основні типи питань, які можуть виникнути

Як ставити дослідницьке питання

- Для проведення доброго аналізу даних потрібно думати більше, ніж робити
- Особливо важливо правильно сформулювати дослідницьке питання
- Характеристики доброго дослідницького питання:
  - Воно повинно становити **цікавість** для авдиторії (природа якої залежить від контексту)
  - Воно не повинно мати відповідей у літературі
  - Воно повинно спиратися на здоровий глузд
  - Відповідь на нього можна дістати (як у гіпотетичному сенсі, так і з практичних міркувань на кшталт наявности даних)
  - Воно повинно бути конкретизовано настільки, щоб на нього можна було дістати вичерпну відповідь
- Нарешті, потрібно заздалегідь передбачати, якого роду відповіді на питання можуть бути, і як їх інтерпретувати
- Розгляньмо основні типи питань, які можуть виникнути

Як ставити дослідницьке питання

- Для проведення доброго аналізу даних потрібно думати більше, ніж робити
- Особливо важливо правильно сформулювати дослідницьке питання
- Характеристики доброго дослідницького питання:
  - Воно повинно становити **цікавість** для авдиторії (природа якої залежить від контексту)
  - Воно не повинно мати відповідей у літературі
  - Воно повинно спиратися на здоровий глузд
  - Відповідь на нього можна дістати (як у гіпотетичному сенсі, так і з практичних міркувань на кшталт наявности даних)
  - Воно повинно бути конкретизовано настільки, щоб на нього можна було дістати вичерпну відповідь
- Нарешті, потрібно заздалегідь передбачати, якого роду відповіді на питання можуть бути, і як їх інтерпретувати
- Розгляньмо основні типи питань, які можуть виникнути

Як ставити дослідницьке питання

- Для проведення доброго аналізу даних потрібно думати більше, ніж робити
- Особливо важливо правильно сформулювати дослідницьке питання
- Характеристики доброго дослідницького питання:
  - Воно повинно становити **цікавість** для авдиторії (природа якої залежить від контексту)
  - Воно не повинно мати відповідей у літературі
  - Воно повинно спиратися на здоровий глузд
  - Відповідь на нього можна дістати (як у гіпотетичному сенсі, так і з практичних міркувань на кшталт наявности даних)
  - Воно повинно бути конкретизовано настільки, щоб на нього можна було дістати вичерпну відповідь
- Нарешті, потрібно заздалегідь передбачати, якого роду відповіді на питання можуть бути, і як їх інтерпретувати
- Розгляньмо основні типи питань, які можуть виникнути

Як ставити дослідницьке питання

- Для проведення доброго аналізу даних потрібно думати більше, ніж робити
- Особливо важливо правильно сформулювати дослідницьке питання
- Характеристики доброго дослідницького питання:
  - Воно повинно становити **цікавість** для авдиторії (природа якої залежить від контексту)
  - Воно не повинно мати відповідей у літературі
  - Воно повинно спиратися на здоровий глузд
  - Відповідь на нього можна дістати (як у гіпотетичному сенсі, так і з практичних міркувань на кшталт наявности даних)
  - Воно повинно бути конкретизовано настільки, щоб на нього можна було дістати вичерпну відповідь
- Нарешті, потрібно заздалегідь передбачати, якого роду відповіді на питання можуть бути, і як їх інтерпретувати
- Розгляньмо основні типи питань, які можуть виникнути

Як ставити дослідницьке питання

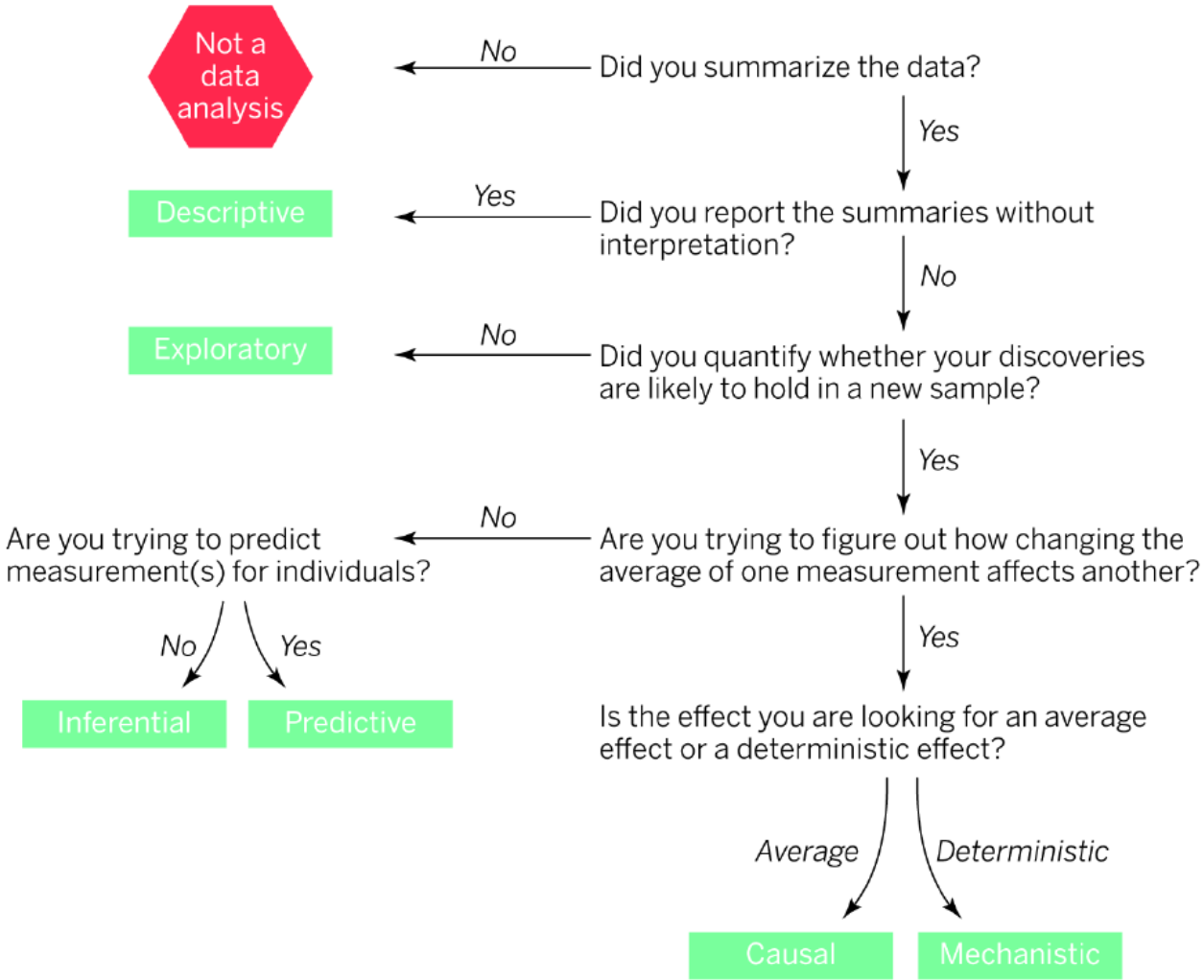
- Для проведення доброго аналізу даних потрібно думати більше, ніж робити
- Особливо важливо правильно сформулювати дослідницьке питання
- Характеристики доброго дослідницького питання:
  - Воно повинно становити **цікавість** для авдиторії (природа якої залежить від контексту)
  - Воно не повинно мати відповідей у літературі
  - Воно повинно спиратися на здоровий глузд
  - Відповідь на нього можна дістати (як у гіпотетичному сенсі, так і з практичних міркувань на кшталт наявности даних)
  - Воно повинно бути конкретизовано настільки, щоб на нього можна було дістати вичерпну відповідь
- Нарешті, потрібно заздалегідь передбачати, якого роду відповіді на питання можуть бути, і як їх інтерпретувати
- Розгляньмо основні типи питань, які можуть виникнути

Як ставити дослідницьке питання

- Для проведення доброго аналізу даних потрібно думати більше, ніж робити
- Особливо важливо правильно сформулювати дослідницьке питання
- Характеристики доброго дослідницького питання:
  - Воно повинно становити **цікавість** для авдиторії (природа якої залежить від контексту)
  - Воно не повинно мати відповідей у літературі
  - Воно повинно спиратися на здоровий глузд
  - Відповідь на нього можна дістати (як у гіпотетичному сенсі, так і з практичних міркувань на кшталт наявности даних)
  - Воно повинно бути конкретизовано настільки, щоб на нього можна було дістати вичерпну відповідь
- Нарешті, потрібно заздалегідь передбачати, якого роду відповіді на питання можуть бути, і як їх інтерпретувати
- Розгляньмо основні типи питань, які можуть виникнути

Типи питань в аналізі даних (1)

- Розгляньмо види аналізу даних, які можуть виринати на практиці [3]



[3] Leek J. T., Peng R. D. What is the question? Science 347(6228), pp. 1314–1315 (2015). https://www.science.org/doi/abs/10.1126/science.aaa6146

Типи питань в аналізі даних (2)

- **Описовий** (descriptive): збір чи підрахунок загальних показників без жодних додаткових інтерпретацій чи пояснень
  - Наприклад, перепис населення, на основі якого можна порахувати населення кожного регіону
- **Розвідковий** (exploratory): пошук трендів, кореляцій, асоціацій тощо між різними змінними з метою формулювання гіпотез, які ще потрібно перевірити
- **Інференційний** (inferential): кількісне вимірювання того, наскільки помічений у даних шаблон повторюватиметься в інших наборах даних

Типи питань в аналізі даних (2)

- **Описовий** (descriptive): збір чи підрахунок загальних показників без жодних додаткових інтерпретацій чи пояснень
  - Наприклад, перепис населення, на основі якого можна порахувати населення кожного регіону
- **Розвідковий** (exploratory): пошук трендів, кореляцій, асоціацій тощо між різними змінними з метою формулювання гіпотез, які ще потрібно перевірити
- **Інференційний** (inferential): кількісне вимірювання того, наскільки помічений у даних шаблон повторюватиметься в інших наборах даних

Типи питань в аналізі даних (2)

- **Описовий** (descriptive): збір чи підрахунок загальних показників без жодних додаткових інтерпретацій чи пояснень
  - Наприклад, перепис населення, на основі якого можна порахувати населення кожного регіону
- **Розвідковий** (exploratory): пошук трендів, кореляцій, асоціацій тощо між різними змінними з метою формулювання гіпотез, які ще потрібно перевірити
- **Інференційний** (inferential): кількісне вимірювання того, наскільки помічений у даних шаблон повторюватиметься в інших наборах даних
  - Фактично мова про оцінку невідомих статистичних параметрів даних для всієї популяції
  - Найпоширеніший тип аналізу в статистиці та економетриці

Типи питань в аналізі даних (2)

- **Описовий** (descriptive): збір чи підрахунок загальних показників без жодних додаткових інтерпретацій чи пояснень
  - Наприклад, перепис населення, на основі якого можна порахувати населення кожного регіону
- **Розвідковий** (exploratory): пошук трендів, кореляцій, асоціацій тощо між різними змінними з метою формулювання гіпотез, які ще потрібно перевірити
- **Інференційний** (inferential): кількісне вимірювання того, наскільки помічений у даних шаблон повторюватиметься в інших наборах даних
  - Фактично мова про оцінку невідомих статистичних параметрів даних для всієї популяції
  - Найпоширеніший тип аналізу в статистиці та економетриці

Типи питань в аналізі даних (2)

- **Описовий** (descriptive): збір чи підрахунок загальних показників без жодних додаткових інтерпретацій чи пояснень
  - Наприклад, перепис населення, на основі якого можна порахувати населення кожного регіону
- **Розвідковий** (exploratory): пошук трендів, кореляцій, асоціацій тощо між різними змінними з метою формулювання гіпотез, які ще потрібно перевірити
- **Інференційний** (inferential): кількісне вимірювання того, наскільки помічений у даних шаблон повторюватиметься в інших наборах даних
  - Фактично мова про оцінку невідомих статистичних параметрів даних для всієї популяції
  - Найпоширеніший тип аналізу в статистиці та економетриці

Типи питань в аналізі даних (2)

- **Описовий** (descriptive): збір чи підрахунок загальних показників без жодних додаткових інтерпретацій чи пояснень
  - Наприклад, перепис населення, на основі якого можна порахувати населення кожного регіону
- **Розвідковий** (exploratory): пошук трендів, кореляцій, асоціацій тощо між різними змінними з метою формулювання гіпотез, які ще потрібно перевірити
- **Інференційний** (inferential): кількісне вимірювання того, наскільки помічений у даних шаблон повторюватиметься в інших наборах даних
  - Фактично мова про оцінку невідомих статистичних параметрів даних для всієї популяції
  - Найпоширеніший тип аналізу в статистиці та економетриці

Типи питань в аналізі даних (3)

- **Прогнозний** (predictive): на основі наявних даних хочемо не просто оцінити параметри деякої модели, що їх породила, а й спрогнозувати нові значення з цієї модели
  - Часто відповідні моделі не дають розуміння, *чому саме* вона правильно прогнозують нові значення
  - Ви їх будете розглядати в рамках машинного навчання
- **Причиново-наслідковий** (causal): ми намагаємося з'ясувати не просто кореляційний зв'язок між деякими змінними, а яка змінна як впливає на інші

Типи питань в аналізі даних (3)

- **Прогнозний** (predictive): на основі наявних даних хочемо не просто оцінити параметри деякої модели, що їх породила, а й спрогнозувати нові значення з цієї модели
  - Часто відповідні моделі не дають розуміння, *чому саме* вона правильно прогнозують нові значення
  - Ви їх будете розглядати в рамках машинного навчання
- **Причиново-наслідковий** (causal): ми намагаємося з'ясувати не просто кореляційний зв'язок між деякими змінними, а яка змінна як впливає на інші

Типи питань в аналізі даних (3)

- **Прогнозний** (predictive): на основі наявних даних хочемо не просто оцінити параметри деякої моделі, що їх породила, а й спрогнозувати нові значення з цієї моделі
  - Часто відповідні моделі не дають розуміння, *чому саме* вона правильно прогнозують нові значення
  - Ви їх будете розглядати в рамках машинного навчання
- **Причиново-наслідковий** (causal): ми намагаємося з'ясувати не просто кореляційний зв'язок між деякими змінними, а яка змінна як впливає на інші
  - Оскільки дані мають статистичний характер, ми можемо встановити прямий зв'язок тільки в середньому
  - Сам дизайн експерименту може свідчити, що інтерпретація буде казуальною
  - Наприклад, якщо дані було зібрано в рамках рандомізованого контрольованого випробування (randomized controlled trial, RCT)
  - В окремих випадках (інженерні задачі тощо), причиново-наслідковий зв'язок можна встановити однозначно (механістичний (mechanistic) аналіз даних)

Типи питань в аналізі даних (3)

- **Прогнозний** (predictive): на основі наявних даних хочемо не просто оцінити параметри деякої модели, що їх породила, а й спрогнозувати нові значення з цієї модели
  - Часто відповідні моделі не дають розуміння, *чому саме* вона правильно прогнозують нові значення
  - Ви їх будете розглядати в рамках машинного навчання
- **Причиново-наслідковий** (causal): ми намагаємося з'ясувати не просто кореляційний зв'язок між деякими змінними, а яка змінна як впливає на інші
  - Оскільки дані мають статистичний характер, ми можемо встановити прямий зв'язок тільки в середньому
  - Сам дизайн експерименту може свідчити, що інтерпретація буде казуальною
  - Наприклад, якщо дані було зібрано в рамках рандомізованого контрольованого випробування (randomized controlled trial, RCT)
  - В окремих випадках (інженерні задачі тощо), причиново-наслідковий зв'язок можна встановити однозначно (механістичний (mechanistic) аналіз даних)

Типи питань в аналізі даних (3)

- **Прогнозний** (predictive): на основі наявних даних хочемо не просто оцінити параметри деякої модели, що їх породила, а й спрогнозувати нові значення з цієї модели
  - Часто відповідні моделі не дають розуміння, *чому саме* вона правильно прогнозують нові значення
  - Ви їх будете розглядати в рамках машинного навчання
- **Причиново-наслідковий** (causal): ми намагаємося з'ясувати не просто кореляційний зв'язок між деякими змінними, а яка змінна як впливає на інші
  - Оскільки дані мають статистичний характер, ми можемо встановити прямий зв'язок тільки в середньому
  - Сам дизайн експерименту може свідчити, що інтерпретація буде казуальною
  - Наприклад, якщо дані було зібрано в рамках рандомізованого контрольованого випробування (randomized controlled trial, RCT)
  - В окремих випадках (інженерні задачі тощо), причиново-наслідковий зв'язок можна встановити однозначно (**механістичний** (mechanistic) аналіз даних)

Типи питань в аналізі даних (3)

- **Прогнозний** (predictive): на основі наявних даних хочемо не просто оцінити параметри деякої модели, що їх породила, а й спрогнозувати нові значення з цієї модели
  - Часто відповідні моделі не дають розуміння, *чому саме* вона правильно прогнозують нові значення
  - Ви їх будете розглядати в рамках машинного навчання
- **Причиново-наслідковий** (causal): ми намагаємося з'ясувати не просто кореляційний зв'язок між деякими змінними, а яка змінна як впливає на інші
  - Оскільки дані мають статистичний характер, ми можемо встановити прямий зв'язок тільки в середньому
  - Сам дизайн експерименту може свідчити, що інтерпретація буде казуальною
  - Наприклад, якщо дані було зібрано в рамках рандомізованого контрольованого випробування (randomized controlled trial, RCT)
  - В окремих випадках (інженерні задачі тощо), причиново-наслідковий зв'язок можна встановити однозначно (**механістичний** (mechanistic) аналіз даних)

Типи питань в аналізі даних (3)

- **Прогнозний** (predictive): на основі наявних даних хочемо не просто оцінити параметри деякої модели, що їх породила, а й спрогнозувати нові значення з цієї модели
  - Часто відповідні моделі не дають розуміння, *чому саме* вона правильно прогнозують нові значення
  - Ви їх будете розглядати в рамках машинного навчання
- **Причиново-наслідковий** (causal): ми намагаємося з'ясувати не просто кореляційний зв'язок між деякими змінними, а яка змінна як впливає на інші
  - Оскільки дані мають статистичний характер, ми можемо встановити прямий зв'язок тільки в середньому
  - Сам дизайн експерименту може свідчити, що інтерпретація буде казуальною
  - Наприклад, якщо дані було зібрано в рамках рандомізованого контрольованого випробування (randomized controlled trial, RCT)
  - В окремих випадках (інженерні задачі тощо), причиново-наслідковий зв'язок можна встановити однозначно (**механістичний** (mechanistic) аналіз даних)

Типи питань в аналізі даних (3)

- **Прогнозний** (predictive): на основі наявних даних хочемо не просто оцінити параметри деякої модели, що їх породила, а й спрогнозувати нові значення з цієї модели
  - Часто відповідні моделі не дають розуміння, *чому саме* вона правильно прогнозують нові значення
  - Ви їх будете розглядати в рамках машинного навчання
- **Причиново-наслідковий** (causal): ми намагаємося з'ясувати не просто кореляційний зв'язок між деякими змінними, а яка змінна як впливає на інші
  - Оскільки дані мають статистичний характер, ми можемо встановити прямий зв'язок тільки в середньому
  - Сам дизайн експерименту може свідчити, що інтерпретація буде казуальною
  - Наприклад, якщо дані було зібрано в рамках рандомізованого контрольованого випробування (randomized controlled trial, RCT)
  - В окремих випадках (інженерні задачі тощо), причиново-наслідковий зв'язок можна встановити однозначно (**механістичний** (mechanistic) аналіз даних)

Неправильна інтепретація видів аналізу

- Розгляньмо типові помилки неправильного визначення природи дослідницького питання [4]
- Видавання інференційного аналізу за причиново-наслідковий
  - Також відоме під назвою «Correlation does not imply causation»
  - Приклади відомих безглуздих кореляцій наведено тут і тут
- Видавання розвідкового аналізу за прогнозний
- Видавання дескриптивного та розвідкового аналізу за інференційний

[4] Leek J. The Elements of Data Analytic Style: A guide for people who want to analyze data. Leanpub (2015)

Неправильна інтепретація видів аналізу

- Розгляньмо типові помилки неправильного визначення природи дослідницького питання [4]
- Видавання інференційного аналізу за причиново-наслідковий
  - Також відоме під назвою «Correlation does not imply causation»
  - Приклади відомих безглуздих кореляцій наведено тут і тут
- Видавання розвідкового аналізу за прогнозний
- Видавання дескриптивного та розвідкового аналізу за інференційний

[4] Leek J. The Elements of Data Analytic Style: A guide for people who want to analyze data. Leanpub (2015)

Неправильна інтепретація видів аналізу

- Розгляньмо типові помилки неправильного визначення природи дослідницького питання [4]
- Видавання інференційного аналізу за причиново-наслідковий
  - Також відоме під назвою «Correlation does not imply causation»
  - Приклади відомих безглуздих кореляцій наведено тут і тут
- Видавання розвідкового аналізу за прогнозний
- Видавання дескриптивного та розвідкового аналізу за інференційний

[4] Leek J. The Elements of Data Analytic Style: A guide for people who want to analyze data. Leanpub (2015)

Неправильна інтепретація видів аналізу

- Розгляньмо типові помилки неправильного визначення природи дослідницького питання [4]
- Видавання інференційного аналізу за причиново-наслідковий
  - Також відоме під назвою «Correlation does not imply causation»
  - Приклади відомих безглуздих кореляцій наведено тут і тут
- Видавання розвідкового аналізу за прогнозний
  - Це тісно пов'язано з поняттям **overfitting** (перенавчання)
  - Увесь набір даних використовують для побудови моделі та її оцінювання
  - Не розбивають на навчальний та тестовий вибірки
- Видавання дескриптивного та розвідкового аналізу за інференційний

[4] Leek J. The Elements of Data Analytic Style: A guide for people who want to analyze data. Leanpub (2015)

Неправильна інтепретація видів аналізу

- Розгляньмо типові помилки неправильного визначення природи дослідницького питання [4]
- Видавання інференційного аналізу за причиново-наслідковий
  - Також відоме під назвою «Correlation does not imply causation»
  - Приклади відомих безглуздих кореляцій наведено тут і тут
- Видавання розвідкового аналізу за прогнозний
  - Це тісно пов'язано з поняттям **overfitting** (перенавчання)
  - Увесь набір даних використовують для побудови модели та її оцінювання
  - Не розбивають на навчальний та тестовий набори
- Видавання дескриптивного та розвідкового аналізу за інференційний

[4] Leek J. The Elements of Data Analytic Style: A guide for people who want to analyze data. Leanpub (2015)

Неправильна інтепретація видів аналізу

- Розгляньмо типові помилки неправильного визначення природи дослідницького питання [4]
- Видавання інференційного аналізу за причиново-наслідковий
  - Також відоме під назвою «Correlation does not imply causation»
  - Приклади відомих безглуздих кореляцій наведено тут і тут
- Видавання розвідкового аналізу за прогнозний
  - Це тісно пов'язано з поняттям **overfitting** (перенавчання)
  - Увесь набір даних використовують для побудови модели та її оцінювання
  - Не розбивають на навчальний та тестовий набори
- Видавання дескриптивного та розвідкового аналізу за інференційний

[4] Leek J. The Elements of Data Analytic Style: A guide for people who want to analyze data. Leanpub (2015)

Неправильна інтепретація видів аналізу

- Розгляньмо типові помилки неправильного визначення природи дослідницького питання [4]
- Видавання інференційного аналізу за причиново-наслідковий
  - Також відоме під назвою «Correlation does not imply causation»
  - Приклади відомих безглуздих кореляцій наведено тут і тут
- Видавання розвідкового аналізу за прогнозний
  - Це тісно пов'язано з поняттям **overfitting** (перенавчання)
  - Увесь набір даних використовують для побудови модели та її оцінювання
  - Не розбивають на навчальний та тестовий набори
- Видавання дескриптивного та розвідкового аналізу за інференційний

[4] Leek J. The Elements of Data Analytic Style: A guide for people who want to analyze data. Leanpub (2015)

Неправильна інтепретація видів аналізу

- Розгляньмо типові помилки неправильного визначення природи дослідницького питання [4]
- Видавання інференційного аналізу за причиново-наслідковий
  - Також відоме під назвою «Correlation does not imply causation»
  - Приклади відомих безглуздих кореляцій наведено тут і тут
- Видавання розвідкового аналізу за прогнозний
  - Це тісно пов'язано з поняттям **overfitting** (перенавчання)
  - Увесь набір даних використовують для побудови моделі та її оцінювання
  - Не розбивають на навчальний та тестовий набори
- Видавання дескриптивного та розвідкового аналізу за інференційний
  - Те, що має місце в одній вибірці, вважають характерним для всієї популяції
  - Якщо на одному наборі будувати багато різних моделей, одна з них суто випадково може показати гарний результат
  - За влучним виразом економіста Рональда Коуза (Ronald Harry Coase, 1910–2013), «Якщо мучити дані достатньо довго, природа в усьому зізнається»

[4] Leek J. The Elements of Data Analytic Style: A guide for people who want to analyze data. Leanpub (2015)

Неправильна інтепретація видів аналізу

- Розгляньмо типові помилки неправильного визначення природи дослідницького питання [4]
- Видавання інференційного аналізу за причиново-наслідковий
  - Також відоме під назвою «Correlation does not imply causation»
  - Приклади відомих безглуздих кореляцій наведено тут і тут
- Видавання розвідкового аналізу за прогнозний
  - Це тісно пов'язано з поняттям **overfitting** (перенавчання)
  - Увесь набір даних використовують для побудови модели та її оцінювання
  - Не розбивають на навчальний та тестовий набори
- Видавання дескриптивного та розвідкового аналізу за інференційний
  - Те, що має місце в одній вибірці, вважають характерним для всієї популяції
  - Якщо на одному наборі будувати багато різних моделей, одна з них суто випадково може показати гарний результат
  - За влучним виразом економіста Рональда Коуза (Ronald Harry Coase, 1910–2013), «Якщо мучити дані достатньо довго, природа в усьому зізнається»

[4] Leek J. The Elements of Data Analytic Style: A guide for people who want to analyze data. Leanpub (2015)

Неправильна інтепретація видів аналізу

- Розгляньмо типові помилки неправильного визначення природи дослідницького питання [4]
- Видавання інференційного аналізу за причиново-наслідковий
  - Також відоме під назвою «Correlation does not imply causation»
  - Приклади відомих безглуздих кореляцій наведено тут і тут
- Видавання розвідкового аналізу за прогнозний
  - Це тісно пов'язано з поняттям **overfitting** (перенавчання)
  - Увесь набір даних використовують для побудови модели та її оцінювання
  - Не розбивають на навчальний та тестовий набори
- Видавання дескриптивного та розвідкового аналізу за інференційний
  - Те, що має місце в одній вибірці, вважають характерним для всієї популяції
  - Якщо на одному наборі будувати багато різних моделей, одна з них суто випадково може показати гарний результат
  - За влучним виразом економіста Рональда Коуза (Ronald Harry Coase, 1910–2013), «Якщо мучити дані достатньо довго, природа в усьому зізнається»

[4] Leek J. The Elements of Data Analytic Style: A guide for people who want to analyze data. Leanpub (2015)

Неправильна інтепретація видів аналізу

- Розгляньмо типові помилки неправильного визначення природи дослідницького питання [4]
- Видавання інференційного аналізу за причиново-наслідковий
  - Також відоме під назвою «Correlation does not imply causation»
  - Приклади відомих безглуздих кореляцій наведено тут і тут
- Видавання розвідкового аналізу за прогнозний
  - Це тісно пов'язано з поняттям **overfitting** (перенавчання)
  - Увесь набір даних використовують для побудови модели та її оцінювання
  - Не розбивають на навчальний та тестовий набори
- Видавання дескриптивного та розвідкового аналізу за інференційний
  - Те, що має місце в одній вибірці, вважають характерним для всієї популяції
  - Якщо на одному наборі будувати багато різних моделей, одна з них суто випадково може показати гарний результат
  - За влучним виразом економіста Рональда Коуза (Ronald Harry Coase, 1910–2013), «Якщо мучити дані достатньо довго, природа в усьому зізнається»

[4] Leek J. The Elements of Data Analytic Style: A guide for people who want to analyze data. Leanpub (2015)

Неправильна інтепретація видів аналізу

- Розгляньмо типові помилки неправильного визначення природи дослідницького питання [4]
- Видавання інференційного аналізу за причиново-наслідковий
  - Також відоме під назвою «Correlation does not imply causation»
  - Приклади відомих безглуздих кореляцій наведено тут і тут
- Видавання розвідкового аналізу за прогнозний
  - Це тісно пов'язано з поняттям **overfitting** (перенавчання)
  - Увесь набір даних використовують для побудови модели та її оцінювання
  - Не розбивають на навчальний та тестовий набори
- Видавання дескриптивного та розвідкового аналізу за інференційний
  - Те, що має місце в одній вибірці, вважають характерним для всієї популяції
  - Якщо на одному наборі будувати багато різних моделей, одна з них суто випадково може показати гарний результат
  - За влучним виразом економіста Рональда Коуза (Ronald Harry Coase, 1910–2013), «Якщо мучити дані достатньо довго, природа в усьому зізнається»

[4] Leek J. The Elements of Data Analytic Style: A guide for people who want to analyze data. Leanpub (2015)

Які можуть бути проблеми з даними (1)

- Вибірка дуже мала
  - Що більше даних, то точніші статистичні оцінки
  - Якщо даних багато, застосовні методи асимптотичної теорії, справедлива центральна гранична теорема
  - Для малих вибірок потрібно застосовувати спеціальні статистичні методи
  - У цьому курсі нас цікавлять тільки **дуже великі** вибірки
- Немає всіх потрібних змінних, замість яких використовують **проксі-змінні** (proxy variables)

Які можуть бути проблеми з даними (1)

- Вибірка дуже мала
  - Що більше даних, то точніші статистичні оцінки
  - Якщо даних багато, застосовні методи асимптотичної теорії, справедлива центральна гранична теорема
  - Для малих вибірок потрібно застосовувати спеціальні статистичні методи
  - У цьому курсі нас цікавлять тільки **дуже великі** вибірки
- Немає всіх потрібних змінних, замість яких використовують **проксі-змінні** (proxy variables)

Які можуть бути проблеми з даними (1)

- Вибірка дуже мала
  - Що більше даних, то точніші статистичні оцінки
  - Якщо даних багато, застосовні методи асимптотичної теорії, справедлива центральна гранична теорема
  - Для малих вибірок потрібно застосовувати спеціальні статистичні методи
  - У цьому курсі нас цікавлять тільки **дуже великі** вибірки
- Немає всіх потрібних змінних, замість яких використовують **проксі-змінні** (proxy variables)

Які можуть бути проблеми з даними (1)

- Вибірка дуже мала
  - Що більше даних, то точніші статистичні оцінки
  - Якщо даних багато, застосовні методи асимптотичної теорії, справедлива центральна гранична теорема
  - Для малих вибірок потрібно застосовувати спеціальні статистичні методи
  - У цьому курсі нас цікавлять тільки **дуже великі** вибірки
- Немає всіх потрібних змінних, замість яких використовують **проксі-змінні** (proxy variables)

Які можуть бути проблеми з даними (1)

- Вибірка дуже мала
  - Що більше даних, то точніші статистичні оцінки
  - Якщо даних багато, застосовні методи асимптотичної теорії, справедлива центральна гранична теорема
  - Для малих вибірок потрібно застосовувати спеціальні статистичні методи
  - У цьому курсі нас цікавлять тільки **дуже великі** вибірки
- Немає всіх потрібних змінних, замість яких використовують **проксі-змінні** (proxy variables)
  - Потрібно оцінити вплив якості життя на народжуваність, але наявні тільки дані про ВВП
  - Потрібно знайти зв'язок між здібностями людини та її доходами, але маємо тільки її оцінки в школі
  - Потрібно з'ясувати вплив куріння на рак легень, але маємо тільки дані про сімейну історію раку

Які можуть бути проблеми з даними (1)

- Вибірка дуже мала
  - Що більше даних, то точніші статистичні оцінки
  - Якщо даних багато, застосовні методи асимптотичної теорії, справедлива центральна гранична теорема
  - Для малих вибірок потрібно застосовувати спеціальні статистичні методи
  - У цьому курсі нас цікавлять тільки **дуже великі** вибірки
- Немає всіх потрібних змінних, замість яких використовують **проксі-змінні** (proxy variables)
  - Потрібно оцінити вплив якости життя на народжуваність, але наявні тільки дані про ВВП
  - Потрібно знайти зв'язок між здібностями людини та її доходами, але маємо тільки її оцінки в школі
  - Потрібно з'ясувати вплив куріння на рак легень, але маємо тільки акцизні збори в деякому районі

Які можуть бути проблеми з даними (1)

- Вибірка дуже мала
  - Що більше даних, то точніші статистичні оцінки
  - Якщо даних багато, застосовні методи асимптотичної теорії, справедлива центральна гранична теорема
  - Для малих вибірок потрібно застосовувати спеціальні статистичні методи
  - У цьому курсі нас цікавлять тільки **дуже великі** вибірки
- Немає всіх потрібних змінних, замість яких використовують **проксі-змінні** (proxy variables)
  - Потрібно оцінити вплив якости життя на народжуваність, але наявні тільки дані про ВВП
  - Потрібно знайти зв'язок між здібностями людини та її доходами, але маємо тільки її оцінки в школі
  - Потрібно з'ясувати вплив куріння на рак легень, але маємо тільки акцизні збори в деякому районі

Які можуть бути проблеми з даними (1)

- Вибірка дуже мала
  - Що більше даних, то точніші статистичні оцінки
  - Якщо даних багато, застосовні методи асимптотичної теорії, справедлива центральна гранична теорема
  - Для малих вибірок потрібно застосовувати спеціальні статистичні методи
  - У цьому курсі нас цікавлять тільки **дуже великі** вибірки
- Немає всіх потрібних змінних, замість яких використовують **проксі-змінні** (proxy variables)
  - Потрібно оцінити вплив якости життя на народжуваність, але наявні тільки дані про ВВП
  - Потрібно знайти зв'язок між здібностями людини та її доходами, але маємо тільки її оцінки в школі
  - Потрібно з'ясувати вплив куріння на рак легень, але маємо тільки акцизні збори в деякому районі

Які можуть бути проблеми з даними (1)

- Вибірка дуже мала
  - Що більше даних, то точніші статистичні оцінки
  - Якщо даних багато, застосовні методи асимптотичної теорії, справедлива центральна гранична теорема
  - Для малих вибірок потрібно застосовувати спеціальні статистичні методи
  - У цьому курсі нас цікавлять тільки **дуже великі** вибірки
- Немає всіх потрібних змінних, замість яких використовують **проксі-змінні** (proxy variables)
  - Потрібно оцінити вплив якости життя на народжуваність, але наявні тільки дані про ВВП
  - Потрібно знайти зв'язок між здібностями людини та її доходами, але маємо тільки її оцінки в школі
  - Потрібно з'ясувати вплив куріння на рак легень, але маємо тільки акцизні збори в деякому районі

Які можуть бути проблеми з даними (2)

- У вибірці відсутні змінні, які можуть бути **конфаундерами** (confounders)
  - Конфаундер впливає одночасно на декілька змінних, і тому видимий зв'язок між ними є хибним
  - Дані можуть свідчити про (додатний) зв'язок уживання алкоголю і раку легень, але причина може бути в тому, що ми не враховуємо куріння
  - Дані можуть свідчити про (від'ємний) зв'язок між рівнем фізичної активности і інфарктом, але причина може бути в тому, що вища активність притаманна молоді
- Дані вибірки не є репрезентативні
- У нашому курсі ви працюватимете з достатньо великими наборами даних, щоб усі ці нюанси було мінімізовано

Які можуть бути проблеми з даними (2)

- У вибірці відсутні змінні, які можуть бути **конфаундерами** (confounders)
  - Конфаундер впливає одночасно на декілька змінних, і тому видимий зв'язок між ними є хибним
  - Дані можуть свідчити про (додатний) зв'язок уживання алкоголю і раку легень, але причина може бути в тому, що ми не враховуємо куріння
  - Дані можуть свідчити про (від'ємний) зв'язок між рівнем фізичної активности і інфарктом, але причина може бути в тому, що вища активність притаманна молоді
- Дані вибірки не є репрезентативні
- У нашому курсі ви працюватимете з достатньо великими наборами даних, щоб усі ці нюанси було мінімізовано

Які можуть бути проблеми з даними (2)

- У вибірці відсутні змінні, які можуть бути **конфаундерами** (confounders)
  - Конфаундер впливає одночасно на декілька змінних, і тому видимий зв'язок між ними є хибним
  - Дані можуть свідчити про (додатний) зв'язок уживання алкоголю і раку легень, але причина може бути в тому, що ми не враховуємо куріння
  - Дані можуть свідчити про (від'ємний) зв'язок між рівнем фізичної активности і інфарктом, але причина може бути в тому, що вища активність притаманна молоді
- Дані вибірки не є репрезентативні
  - В ідеалі, у репрезентативній вибірці частка спостережень із певними характеристиками повинна приблизно дорівнювати ймовірності зустріти їх у популяції
  - Якщо вибірка нерепрезентативна, результати аналізу не можуть бути показовими для всієї популяції (страждає зовнішня валідність (external validity) аналізу)
  - Ми маємо справу з систематичною похибкою відбору (selection bias), яку потрібно враховувати спеціальними методами
  - (Чи можна довіряти дослідженням, у яких респонденти за своїм бажанням заповнюють анкети?)
  - В окремих випадках можна надати різним спостереженням різної ваги, щоб компенсувати непропорційність їх представлення
- У нашому курсі ви працюватимете з достатньо великими наборами даних, щоб усі ці нюанси було мінімізовано

Які можуть бути проблеми з даними (2)

- У вибірці відсутні змінні, які можуть бути **конфаундерами** (confounders)
  - Конфаундер впливає одночасно на декілька змінних, і тому видимий зв'язок між ними є хибним
  - Дані можуть свідчити про (додатний) зв'язок уживання алкоголю і раку легень, але причина може бути в тому, що ми не враховуємо куріння
  - Дані можуть свідчити про (від'ємний) зв'язок між рівнем фізичної активности і інфарктом, але причина може бути в тому, що вища активність притаманна молоді
- Дані вибірки не є репрезентативні
  - В ідеалі, у репрезентативній вибірці частка спостережень із певними характеристиками повинна приблизно дорівнювати ймовірності зустріти їх у популяції
  - Якщо вибірка нерепрезентативна, результати аналізу не можуть бути показові для усієї популяції (страждає зовнішня валідність (external validity) аналізу)
  - Ми маємо справу з систематичною помилкою відбору (selection bias), яку потрібно враховувати спеціальними методами
  - (Чи можна довіряти дослідженням, у яких респонденти за своїм бажанням заповнюють анкети?)
  - В окремих випадках можна надати різним спостереженням різної ваги, щоб компенсувати непропорційність їх представлення
- У нашому курсі ви працюватимете з достатньо великими наборами даних, щоб усі ці нюанси було мінімізовано

Які можуть бути проблеми з даними (2)

- У вибірці відсутні змінні, які можуть бути **конфаундерами** (confounders)
  - Конфаундер впливає одночасно на декілька змінних, і тому видимий зв'язок між ними є хибним
  - Дані можуть свідчити про (додатний) зв'язок уживання алкоголю і раку легень, але причина може бути в тому, що ми не враховуємо куріння
  - Дані можуть свідчити про (від'ємний) зв'язок між рівнем фізичної активности і інфарктом, але причина може бути в тому, що вища активність притаманна молоді
- Дані вибірки не є репрезентативні
  - В ідеалі, у репрезентативній вибірці частка спостережень із певними характеристиками повинна приблизно дорівнювати ймовірності зустріти їх у популяції
  - Якщо вибірка нерепрезентативна, результати аналізу не можуть бути показові для всієї популяції (страждає **зовнішня валідність** (external validity) аналізу)
  - Ми маємо справу з **систематичною похибкою відбору** (selection bias), яку потрібно враховувати спеціальними методами
  - (Чи можна довіряти дослідженням, у яких респонденти **за своїм бажанням** заповнюють анкети?)
  - В окремих випадках можна надати різним спостереженням різної ваги, щоб компенсувати непропорційність їх представлення
- У нашому курсі ви працюватимете з достатньо великими наборами даних, щоб усі ці нюанси було мінімізовано

Які можуть бути проблеми з даними (2)

- У вибірці відсутні змінні, які можуть бути **конфаундерами** (confounders)
  - Конфаундер впливає одночасно на декілька змінних, і тому видимий зв'язок між ними є хибним
  - Дані можуть свідчити про (додатний) зв'язок уживання алкоголю і раку легень, але причина може бути в тому, що ми не враховуємо куріння
  - Дані можуть свідчити про (від'ємний) зв'язок між рівнем фізичної активности і інфарктом, але причина може бути в тому, що вища активність притаманна молоді
- Дані вибірки не є репрезентативні
  - В ідеалі, у репрезентативній вибірці частка спостережень із певними характеристиками повинна приблизно дорівнювати ймовірності зустріти їх у популяції
  - Якщо вибірка нерепрезентативна, результати аналізу не можуть бути показові для всієї популяції (страждає **зовнішня валідність** (external validity) аналізу)
  - Ми маємо справу з **систематичною похибкою відбору** (selection bias), яку потрібно враховувати спеціальними методами
  - (Чи можна довіряти дослідженням, у яких респонденти **за своїм бажанням** заповнюють анкети?)
  - В окремих випадках можна надати різним спостереженням різної ваги, щоб компенсувати непропорційність їх представлення
- У нашому курсі ви працюватимете з достатньо великими наборами даних, щоб усі ці нюанси було мінімізовано

Які можуть бути проблеми з даними (2)

- У вибірці відсутні змінні, які можуть бути **конфаундерами** (confounders)
  - Конфаундер впливає одночасно на декілька змінних, і тому видимий зв'язок між ними є хибним
  - Дані можуть свідчити про (додатний) зв'язок уживання алкоголю і раку легень, але причина може бути в тому, що ми не враховуємо куріння
  - Дані можуть свідчити про (від'ємний) зв'язок між рівнем фізичної активности і інфарктом, але причина може бути в тому, що вища активність притаманна молоді
- Дані вибірки не є репрезентативні
  - В ідеалі, у репрезентативній вибірці частка спостережень із певними характеристиками повинна приблизно дорівнювати ймовірності зустріти їх у популяції
  - Якщо вибірка нерепрезентативна, результати аналізу не можуть бути показові для всієї популяції (страждає **зовнішня валідність** (external validity) аналізу)
  - Ми маємо справу з **систематичною похибкою відбору** (selection bias), яку потрібно враховувати спеціальними методами
  - (Чи можна довіряти дослідженням, у яких респонденти **за своїм бажанням** заповнюють анкети?)
  - В окремих випадках можна надати різним спостереженням різної ваги, щоб компенсувати непропорційність їх представлення
- У нашому курсі ви працюватимете з достатньо великими наборами даних, щоб усі ці нюанси було мінімізовано

Які можуть бути проблеми з даними (2)

- У вибірці відсутні змінні, які можуть бути **конфаундерами** (confounders)
  - Конфаундер впливає одночасно на декілька змінних, і тому видимий зв'язок між ними є хибним
  - Дані можуть свідчити про (додатний) зв'язок уживання алкоголю і раку легень, але причина може бути в тому, що ми не враховуємо куріння
  - Дані можуть свідчити про (від'ємний) зв'язок між рівнем фізичної активности і інфарктом, але причина може бути в тому, що вища активність притаманна молоді
- Дані вибірки не є репрезентативні
  - В ідеалі, у репрезентативній вибірці частка спостережень із певними характеристиками повинна приблизно дорівнювати ймовірності зустріти їх у популяції
  - Якщо вибірка нерепрезентативна, результати аналізу не можуть бути показові для всієї популяції (страждає **зовнішня валідність** (external validity) аналізу)
  - Ми маємо справу з **систематичною похибкою відбору** (selection bias), яку потрібно враховувати спеціальними методами
  - (Чи можна довіряти дослідженням, у яких респонденти **за своїм бажанням** заповнюють анкети?)
  - В окремих випадках можна надати різним спостереженням різної ваги, щоб компенсувати непропорційність їх представлення
- У нашому курсі ви працюватимете з достатньо великими наборами даних, щоб усі ці нюанси було мінімізовано

Які можуть бути проблеми з даними (2)

- У вибірці відсутні змінні, які можуть бути **конфаундерами** (confounders)
  - Конфаундер впливає одночасно на декілька змінних, і тому видимий зв'язок між ними є хибним
  - Дані можуть свідчити про (додатний) зв'язок уживання алкоголю і раку легень, але причина може бути в тому, що ми не враховуємо куріння
  - Дані можуть свідчити про (від'ємний) зв'язок між рівнем фізичної активности і інфарктом, але причина може бути в тому, що вища активність притаманна молоді
- Дані вибірки не є репрезентативні
  - В ідеалі, у репрезентативній вибірці частка спостережень із певними характеристиками повинна приблизно дорівнювати ймовірності зустріти їх у популяції
  - Якщо вибірка нерепрезентативна, результати аналізу не можуть бути показові для всієї популяції (страждає **зовнішня валідність** (external validity) аналізу)
  - Ми маємо справу з **систематичною похибкою відбору** (selection bias), яку потрібно враховувати спеціальними методами
  - (Чи можна довіряти дослідженням, у яких респонденти **за своїм бажанням** заповнюють анкети?)
  - В окремих випадках можна надати різним спостереженням різної ваги, щоб компенсувати непропорційність їх представлення
- У нашому курсі ви працюватимете з достатньо великими наборами даних, щоб усі ці нюанси було мінімізовано

Які можуть бути проблеми з даними (2)

- У вибірці відсутні змінні, які можуть бути **конфаундерами** (confounders)
  - Конфаундер впливає одночасно на декілька змінних, і тому видимий зв'язок між ними є хибним
  - Дані можуть свідчити про (додатний) зв'язок уживання алкоголю і раку легень, але причина може бути в тому, що ми не враховуємо куріння
  - Дані можуть свідчити про (від'ємний) зв'язок між рівнем фізичної активности і інфарктом, але причина може бути в тому, що вища активність притаманна молоді
- Дані вибірки не є репрезентативні
  - В ідеалі, у репрезентативній вибірці частка спостережень із певними характеристиками повинна приблизно дорівнювати ймовірності зустріти їх у популяції
  - Якщо вибірка нерепрезентативна, результати аналізу не можуть бути показові для всієї популяції (страждає **зовнішня валідність** (external validity) аналізу)
  - Ми маємо справу з **систематичною похибкою відбору** (selection bias), яку потрібно враховувати спеціальними методами
  - (Чи можна довіряти дослідженням, у яких респонденти **за своїм бажанням** заповнюють анкети?)
  - В окремих випадках можна надати різним спостереженням різної ваги, щоб компенсувати непропорційність їх представлення
- У нашому курсі ви працюватимете з достатньо великими наборами даних, щоб усі ці нюанси було мінімізовано

Які можуть бути проблеми з даними (2)

- У вибірці відсутні змінні, які можуть бути **конфаундерами** (confounders)
  - Конфаундер впливає одночасно на декілька змінних, і тому видимий зв'язок між ними є хибним
  - Дані можуть свідчити про (додатний) зв'язок уживання алкоголю і раку легень, але причина може бути в тому, що ми не враховуємо куріння
  - Дані можуть свідчити про (від'ємний) зв'язок між рівнем фізичної активности і інфарктом, але причина може бути в тому, що вища активність притаманна молоді
- Дані вибірки не є репрезентативні
  - В ідеалі, у репрезентативній вибірці частка спостережень із певними характеристиками повинна приблизно дорівнювати ймовірності зустріти їх у популяції
  - Якщо вибірка нерепрезентативна, результати аналізу не можуть бути показові для всієї популяції (страждає **зовнішня валідність** (external validity) аналізу)
  - Ми маємо справу з **систематичною похибкою відбору** (selection bias), яку потрібно враховувати спеціальними методами
  - (Чи можна довіряти дослідженням, у яких респонденти **за своїм бажанням** заповнюють анкети?)
  - В окремих випадках можна надати різним спостереженням різної ваги, щоб компенсувати непропорційність їх представлення
- У нашому курсі ви працюватимете з достатньо великими наборами даних, щоб усі ці нюанси було мінімізовано

Приклад формулювання дослідницького питання (1)

- Розгляньмо приклад аналізу даних компанії-виробника фітнес-трекерів на предмет виявлення цільової авдиторії для реклами трекера сну [5]
- У базі даних наявна така інформація: основні демографічні параметри, кількість кроків за день, кількість клітин сходів за день, кількість сидячих годин за день, кількість годин бадьорости та сонливости за день, кількість годин сну за день
- Зазначена мета дослідження не є достатньо конкретною, тому дослідник у спілкуванні з керівництвом формулює деталізованіше питання: *«Які клієнти компанії недосипають?»*
- Дослідник повинен переконатися, що питання відповідає всім критеріям доброго питання

[5] Peng R., Matsui E. The Art of Data Science. Leanpub (2016)

Приклад формулювання дослідницького питання (1)

- Розгляньмо приклад аналізу даних компанії-виробника фітнес-трекерів на предмет виявлення цільової авдиторії для реклами трекера сну [5]
- У базі даних наявна така інформація: основні демографічні параметри, кількість кроків за день, кількість клітин сходів за день, кількість сидячих годин за день, кількість годин бадьорости та сонливости за день, кількість годин сну за день
- Зазначена мета дослідження не є достатньо конкретною, тому дослідник у спілкуванні з керівництвом формулює деталізованіше питання: *«Які клієнти компанії недосипають?»*
- Дослідник повинен переконатися, що питання відповідає всім критеріям доброго питання

[5] Peng R., Matsui E. The Art of Data Science. Leanpub (2016)

Приклад формулювання дослідницького питання (1)

- Розгляньмо приклад аналізу даних компанії-виробника фітнес-трекерів на предмет виявлення цільової авдиторії для реклами трекера сну [5]
- У базі даних наявна така інформація: основні демографічні параметри, кількість кроків за день, кількість клітин сходів за день, кількість сидячих годин за день, кількість годин бадьорости та сонливости за день, кількість годин сну за день
- Зазначена мета дослідження не є достатньо конкретною, тому дослідник у спілкуванні з керівництвом формулює деталізованіше питання: *«Які клієнти компанії недосипають?»*
- Дослідник повинен переконатися, що питання відповідає всім критеріям доброго питання
    - Воно становить цікавість для керівництва
    - Воно, певно, не має відповідей у літературі (інакше керівництво не замовило б це дослідження, хоча воно може помилятися!)
    - Воно спирається на здоровий глузд: клієнти, що недосипають, можуть бути зацікавлені в трекері сну
    - ... Але що означає недосипати?

[5] Peng R., Matsui E. The Art of Data Science. Leanpub (2016)

Приклад формулювання дослідницького питання (1)

- Розгляньмо приклад аналізу даних компанії-виробника фітнес-трекерів на предмет виявлення цільової авдиторії для реклами трекера сну [5]
- У базі даних наявна така інформація: основні демографічні параметри, кількість кроків за день, кількість клітин сходів за день, кількість сидячих годин за день, кількість годин бадьорости та сонливости за день, кількість годин сну за день
- Зазначена мета дослідження не є достатньо конкретною, тому дослідник у спілкуванні з керівництвом формулює деталізованіше питання: *«Які клієнти компанії недосипають?»*
- Дослідник повинен переконатися, що питання відповідає всім критеріям доброго питання
  - Воно становить цікавість для керівництва
  - Воно, **певно**, не має відповідей у літературі (інакше керівництво не замовило б це дослідження, хоча воно може помилятися!)
  - Воно спирається на здоровий глузд: клієнти, що недосипають, можуть бути зацікавлені в трекері сну
  - ... Але що вважати недосипом?

[5] Peng R., Matsui E. The Art of Data Science. Leanpub (2016)

Приклад формулювання дослідницького питання (1)

- Розгляньмо приклад аналізу даних компанії-виробника фітнес-трекерів на предмет виявлення цільової авдиторії для реклами трекера сну [5]
- У базі даних наявна така інформація: основні демографічні параметри, кількість кроків за день, кількість клітин сходів за день, кількість сидячих годин за день, кількість годин бадьорости та сонливости за день, кількість годин сну за день
- Зазначена мета дослідження не є достатньо конкретною, тому дослідник у спілкуванні з керівництвом формулює деталізованіше питання: *«Які клієнти компанії недосипають?»*
- Дослідник повинен переконатися, що питання відповідає всім критеріям доброго питання
  - Воно становить цікавість для керівництва
  - Воно, **певно**, не має відповідей у літературі (інакше керівництво не замовило б це дослідження, хоча воно може помилятися!)
  - Воно спирається на здоровий глузд: клієнти, що недосипають, можуть бути зацікавлені в трекері сну
  - … Але що вважати недосипом?

[5] Peng R., Matsui E. The Art of Data Science. Leanpub (2016)

Приклад формулювання дослідницького питання (1)

- Розгляньмо приклад аналізу даних компанії-виробника фітнес-трекерів на предмет виявлення цільової авдиторії для реклами трекера сну [5]
- У базі даних наявна така інформація: основні демографічні параметри, кількість кроків за день, кількість клітин сходів за день, кількість сидячих годин за день, кількість годин бадьорости та сонливости за день, кількість годин сну за день
- Зазначена мета дослідження не є достатньо конкретною, тому дослідник у спілкуванні з керівництвом формулює деталізованіше питання: *«Які клієнти компанії недосипають?»*
- Дослідник повинен переконатися, що питання відповідає всім критеріям доброго питання
  - Воно становить цікавість для керівництва
  - Воно, **певно**, не має відповідей у літературі (інакше керівництво не замовило б це дослідження, хоча воно може помилятися!)
  - Воно спирається на здоровий глузд: клієнти, що недосипають, можуть бути зацікавлені в трекері сну
  - … Але що вважати недосипом?

[5] Peng R., Matsui E. The Art of Data Science. Leanpub (2016)

Приклад формулювання дослідницького питання (1)

- Розгляньмо приклад аналізу даних компанії-виробника фітнес-трекерів на предмет виявлення цільової авдиторії для реклами трекера сну [5]
- У базі даних наявна така інформація: основні демографічні параметри, кількість кроків за день, кількість клітин сходів за день, кількість сидячих годин за день, кількість годин бадьорости та сонливости за день, кількість годин сну за день
- Зазначена мета дослідження не є достатньо конкретною, тому дослідник у спілкуванні з керівництвом формулює деталізованіше питання: *«Які клієнти компанії недосипають?»*
- Дослідник повинен переконатися, що питання відповідає всім критеріям доброго питання
  - Воно становить цікавість для керівництва
  - Воно, **певно**, не має відповідей у літературі (інакше керівництво не замовило б це дослідження, хоча воно може помилятися!)
  - Воно спирається на здоровий глузд: клієнти, що недосипають, можуть бути зацікавлені в трекері сну
  - … Але що вважати недосипом?

[5] Peng R., Matsui E. The Art of Data Science. Leanpub (2016)

Приклад формулювання дослідницького питання (1)

- Розгляньмо приклад аналізу даних компанії-виробника фітнес-трекерів на предмет виявлення цільової авдиторії для реклами трекера сну [5]
- У базі даних наявна така інформація: основні демографічні параметри, кількість кроків за день, кількість клітин сходів за день, кількість сидячих годин за день, кількість годин бадьорости та сонливости за день, кількість годин сну за день
- Зазначена мета дослідження не є достатньо конкретною, тому дослідник у спілкуванні з керівництвом формулює деталізованіше питання: *«Які клієнти компанії недосипають?»*
- Дослідник повинен переконатися, що питання відповідає всім критеріям доброго питання
  - Воно становить цікавість для керівництва
  - Воно, **певно**, не має відповідей у літературі (інакше керівництво не замовило б це дослідження, хоча воно може помилятися!)
  - Воно спирається на здоровий глузд: клієнти, що недосипають, можуть бути зацікавлені в трекері сну
  - … Але що вважати недосипом?

[5] Peng R., Matsui E. The Art of Data Science. Leanpub (2016)

Приклад формулювання дослідницького питання (2)

- Після спілкування з фахівцями, дослідник усвідомлює, що ліпше аналізувати не кількість сну, а сонливість протягом дня
- Отже питання дістає нове формулювання: *«Які клієнти компанії страждають на сонливість протягом дня?»*
- На щастя, наявні дані дають змогу дати відповідь на таке питання
- Але питання недостатньо конкретизовано
- У спілкуванні з колегами дослідник визнає, що потрібно чіткіше сформулювати, що значить «які клієнти», що таке «сонливість протягом дня» (скільки саме?)
- Остаточне формулювання звучить так: *«Які демографічні та медичні характеристики описують клієнтів, які страждають на хронічну сонливість, визначену як щонайменше один епізод сонливости щонайменше кожного другого дня?»*
- Дослідник очікує два можливі результати аналізу:

Приклад формулювання дослідницького питання (2)

- Після спілкування з фахівцями, дослідник усвідомлює, що ліпше аналізувати не кількість сну, а сонливість протягом дня
- Отже питання дістає нове формулювання: *«Які клієнти компанії страждають на сонливість протягом дня?»*
- На щастя, наявні дані дають змогу дати відповідь на таке питання
- Але питання недостатньо конкретизовано
- У спілкуванні з колегами дослідник визнає, що потрібно чіткіше сформулювати, що значить «які клієнти», що таке «сонливість протягом дня» (скільки саме?)
- Остаточне формулювання звучить так: *«Які демографічні та медичні характеристики описують клієнтів, які страждають на хронічну сонливість, визначену як щонайменше один епізод сонливості щонайменше кожного другого дня?»*
- Дослідник очікує два можливі результати аналізу:

Приклад формулювання дослідницького питання (2)

- Після спілкування з фахівцями, дослідник усвідомлює, що ліпше аналізувати не кількість сну, а сонливість протягом дня
- Отже питання дістає нове формулювання: *«Які клієнти компанії страждають на сонливість протягом дня?»*
- На щастя, наявні дані дають змогу дати відповідь на таке питання
- Але питання недостатньо конкретизовано
- У спілкуванні з колегами дослідник визнає, що потрібно чіткіше сформулювати, що значить «які клієнти», що таке «сонливість протягом дня» (скільки саме?)
- Остаточне формулювання звучить так: *«Які демографічні та медичні характеристики описують клієнтів, які страждають на хронічну сонливість, визначену як щонайменше один епізод сонливости щонайменше кожного другого дня?»*
- Дослідник очікує два можливі результати аналізу:

Приклад формулювання дослідницького питання (2)

- Після спілкування з фахівцями, дослідник усвідомлює, що ліпше аналізувати не кількість сну, а сонливість протягом дня
- Отже питання дістає нове формулювання: *«Які клієнти компанії страждають на сонливість протягом дня?»*
- На щастя, наявні дані дають змогу дати відповідь на таке питання
- Але питання недостатньо конкретизовано
- У спілкуванні з колегами дослідник визнає, що потрібно чіткіше сформулювати, що значить «які клієнти», що таке «сонливість протягом дня» (скільки саме?)
- Остаточне формулювання звучить так: *«Які демографічні та медичні характеристики описують клієнтів, які страждають на хронічну сонливість, визначену як щонайменше один епізод сонливости щонайменше кожного другого дня?»*
- Дослідник очікує два можливі результати аналізу:

Приклад формулювання дослідницького питання (2)

- Після спілкування з фахівцями, дослідник усвідомлює, що ліпше аналізувати не кількість сну, а сонливість протягом дня
- Отже питання дістає нове формулювання: *«Які клієнти компанії страждають на сонливість протягом дня?»*
- На щастя, наявні дані дають змогу дати відповідь на таке питання
- Але питання недостатньо конкретизовано
- У спілкуванні з колегами дослідник визнає, що потрібно чіткіше сформулювати, що значить «які клієнти», що таке «сонливість протягом дня» (скільки саме?)
- Остаточне формулювання звучить так: *«Які демографічні та медичні характеристики описують клієнтів, які страждають на хронічну сонливість, визначену як щонайменше один епізод сонливости щонайменше кожного другого дня?»*
- Дослідник очікує два можливі результати аналізу:
  - Жодних особливих характеристик виявлено не буде. Інтерпретація: таргетована реклама не матиме сенсу
  - Буде знайдено характеристики, що пов'язані з сонливістю. Інтерпретація: можна буде таргетувати рекламу для клієнтів із такими характеристиками

Приклад формулювання дослідницького питання (2)

- Після спілкування з фахівцями, дослідник усвідомлює, що ліпше аналізувати не кількість сну, а сонливість протягом дня
- Отже питання дістає нове формулювання: *«Які клієнти компанії страждають на сонливість протягом дня?»*
- На щастя, наявні дані дають змогу дати відповідь на таке питання
- Але питання недостатньо конкретизовано
- У спілкуванні з колегами дослідник визнає, що потрібно чіткіше сформулювати, що значить «які клієнти», що таке «сонливість протягом дня» (скільки саме?)
- Остаточне формулювання звучить так: *«Які демографічні та медичні характеристики описують клієнтів, які страждають на хронічну сонливість, визначену як щонайменше один епізод сонливости щонайменше кожного другого дня?»*
- Дослідник очікує два можливі результати аналізу:
  - Жодних особливих характеристик виявлено не буде
  - **Інтерпретація:** таргетована реклама не матиме сенсу
  - Буде знайдено характеристики, що пов'язані з сонливістю
  - **Інтерпретація:** можна буде таргетувати рекламу для клієнтів із такими характеристиками

Приклад формулювання дослідницького питання (2)

- Після спілкування з фахівцями, дослідник усвідомлює, що ліпше аналізувати не кількість сну, а сонливість протягом дня
- Отже питання дістає нове формулювання: *«Які клієнти компанії страждають на сонливість протягом дня?»*
- На щастя, наявні дані дають змогу дати відповідь на таке питання
- Але питання недостатньо конкретизовано
- У спілкуванні з колегами дослідник визнає, що потрібно чіткіше сформулювати, що значить «які клієнти», що таке «сонливість протягом дня» (скільки саме?)
- Остаточне формулювання звучить так: *«Які демографічні та медичні характеристики описують клієнтів, які страждають на хронічну сонливість, визначену як щонайменше один епізод сонливости щонайменше кожного другого дня?»*
- Дослідник очікує два можливі результати аналізу:
  - Жодних особливих характеристик виявлено не буде
  - **Інтерпретація**: таргетована реклама не матиме сенсу
  - Буде знайдено характеристики, що пов'язані з сонливістю
  - **Інтерпретація**: можна буде таргетувати рекламу для клієнтів із такими характеристиками

Приклад формулювання дослідницького питання (2)

- Після спілкування з фахівцями, дослідник усвідомлює, що ліпше аналізувати не кількість сну, а сонливість протягом дня
- Отже питання дістає нове формулювання: *«Які клієнти компанії страждають на сонливість протягом дня?»*
- На щастя, наявні дані дають змогу дати відповідь на таке питання
- Але питання недостатньо конкретизовано
- У спілкуванні з колегами дослідник визнає, що потрібно чіткіше сформулювати, що значить «які клієнти», що таке «сонливість протягом дня» (скільки саме?)
- Остаточне формулювання звучить так: *«Які демографічні та медичні характеристики описують клієнтів, які страждають на хронічну сонливість, визначену як щонайменше один епізод сонливости щонайменше кожного другого дня?»*
- Дослідник очікує два можливі результати аналізу:
  - Жодних особливих характеристик виявлено не буде
  - **Інтерпретація**: таргетована реклама не матиме сенсу
  - Буде знайдено характеристики, що пов'язані з сонливістю
  - **Інтерпретація**: можна буде таргетувати рекламу для клієнтів із такими характеристиками

Приклад формулювання дослідницького питання (2)

- Після спілкування з фахівцями, дослідник усвідомлює, що ліпше аналізувати не кількість сну, а сонливість протягом дня
- Отже питання дістає нове формулювання: *«Які клієнти компанії страждають на сонливість протягом дня?»*
- На щастя, наявні дані дають змогу дати відповідь на таке питання
- Але питання недостатньо конкретизовано
- У спілкуванні з колегами дослідник визнає, що потрібно чіткіше сформулювати, що значить «які клієнти», що таке «сонливість протягом дня» (скільки саме?)
- Остаточне формулювання звучить так: *«Які демографічні та медичні характеристики описують клієнтів, які страждають на хронічну сонливість, визначену як щонайменше один епізод сонливости щонайменше кожного другого дня?»*
- Дослідник очікує два можливі результати аналізу:
  - Жодних особливих характеристик виявлено не буде
  - **Інтерпретація**: таргетована реклама не матиме сенсу
  - Буде знайдено характеристики, що пов'язані з сонливістю
  - **Інтерпретація**: можна буде таргетувати рекламу для клієнтів із такими характеристиками

Приклад формулювання дослідницького питання (2)

- Після спілкування з фахівцями, дослідник усвідомлює, що ліпше аналізувати не кількість сну, а сонливість протягом дня
- Отже питання дістає нове формулювання: *«Які клієнти компанії страждають на сонливість протягом дня?»*
- На щастя, наявні дані дають змогу дати відповідь на таке питання
- Але питання недостатньо конкретизовано
- У спілкуванні з колегами дослідник визнає, що потрібно чіткіше сформулювати, що значить «які клієнти», що таке «сонливість протягом дня» (скільки саме?)
- Остаточне формулювання звучить так: *«Які демографічні та медичні характеристики описують клієнтів, які страждають на хронічну сонливість, визначену як щонайменше один епізод сонливости щонайменше кожного другого дня?»*
- Дослідник очікує два можливі результати аналізу:
  - Жодних особливих характеристик виявлено не буде
  - **Інтерпретація**: таргетована реклама не матиме сенсу
  - Буде знайдено характеристики, що пов’язані з сонливістю
  - **Інтерпретація**: можна буде таргетувати рекламу для клієнтів із такими характеристиками

Приклад формулювання дослідницького питання (2)

- Після спілкування з фахівцями, дослідник усвідомлює, що ліпше аналізувати не кількість сну, а сонливість протягом дня
- Отже питання дістає нове формулювання: *«Які клієнти компанії страждають на сонливість протягом дня?»*
- На щастя, наявні дані дають змогу дати відповідь на таке питання
- Але питання недостатньо конкретизовано
- У спілкуванні з колегами дослідник визнає, що потрібно чіткіше сформулювати, що значить «які клієнти», що таке «сонливість протягом дня» (скільки саме?)
- Остаточне формулювання звучить так: *«Які демографічні та медичні характеристики описують клієнтів, які страждають на хронічну сонливість, визначену як щонайменше один епізод сонливости щонайменше кожного другого дня?»*
- Дослідник очікує два можливі результати аналізу:
  - Жодних особливих характеристик виявлено не буде
  - **Інтерпретація**: таргетована реклама не матиме сенсу
  - Буде знайдено характеристики, що пов’язані з сонливістю
  - **Інтерпретація**: можна буде таргетувати рекламу для клієнтів із такими характеристиками

Приклад формулювання дослідницького питання (3)

- Наостанок дослідник повинен з’ясувати, якого типу його питання
- Це дасть змогу зрозуміти, які методи аналізу ліпше використати
- На перший погляд може здатися, що питання має розвідковий характер
- Проте у компанії можуть з’являтися нові клієнти, і тому було б корисно таргетувати рекламу і для них також
- Відтак питання носить **прогнозний** характер
- Тому для відповіді на нього потрібно будувати моделі для прогнозування

Приклад формулювання дослідницького питання (3)

- Наостанок дослідник повинен з'ясувати, якого типу його питання
- Це дасть змогу зрозуміти, які методи аналізу ліпше використати
- На перший погляд може здатися, що питання має розвідковий характер
  - Потрібно подивитися, які існують зв'язки між різними змінними
  - Оскільки ми працюємо з клієнтами однієї компанії, нас не дуже цікавить, чи матиме місце цей зв'язок у всій популяції (тому точно не інференційний)
- Проте у компанії можуть з'являтися нові клієнти, і тому було б корисно таргетувати рекламу і для них також
- Відтак питання носить **прогнозний** характер
- Тому для відповіди на нього потрібно будувати моделі для прогнозування

Приклад формулювання дослідницького питання (3)

- Наостанок дослідник повинен з’ясувати, якого типу його питання
- Це дасть змогу зрозуміти, які методи аналізу ліпше використати
- На перший погляд може здатися, що питання має розвідковий характер
  - Потрібно подивитися, які існують зв’язки між різними змінними
  - Оскільки ми працюємо з клієнтами однієї компанії, нас не дуже цікавить, чи матиме місце цей зв’язок у всій популяції (тому точно не інференційний)
- Проте у компанії можуть з’являтися нові клієнти, і тому було б корисно таргетувати рекламу і для них також
- Відтак питання носить прогнозний характер
- Тому для відповіди на нього потрібно будувати моделі для прогнозування

Приклад формулювання дослідницького питання (3)

- Наостанок дослідник повинен з'ясувати, якого типу його питання
- Це дасть змогу зрозуміти, які методи аналізу ліпше використати
- На перший погляд може здатися, що питання має розвідковий характер
  - Потрібно подивитися, які існують зв'язки між різними змінними
  - Оскільки ми працюємо з клієнтами однієї компанії, нас не дуже цікавить, чи матиме місце цей зв'язок у всій популяції (тому точно не інференційний)
- Проте у компанії можуть з'являтися нові клієнти, і тому було б корисно таргетувати рекламу і для них також
- Відтак питання носить **прогнозний** характер
- Тому для відповіди на нього потрібно будувати моделі для прогнозування

Приклад формулювання дослідницького питання (3)

- Наостанок дослідник повинен з’ясувати, якого типу його питання
- Це дасть змогу зрозуміти, які методи аналізу ліпше використати
- На перший погляд може здатися, що питання має розвідковий характер
  - Потрібно подивитися, які існують зв’язки між різними змінними
  - Оскільки ми працюємо з клієнтами однієї компанії, нас не дуже цікавить, чи матиме місце цей зв’язок у всій популяції (тому точно не інференційний)
- Проте у компанії можуть з’являтися нові клієнти, і тому було б корисно таргетувати рекламу і для них також
- Відтак питання носить **прогнозний** характер
- Тому для відповіди на нього потрібно будувати моделі для прогнозування

Приклад формулювання дослідницького питання (3)

- Наостанок дослідник повинен з'ясувати, якого типу його питання
- Це дасть змогу зрозуміти, які методи аналізу ліпше використати
- На перший погляд може здатися, що питання має розвідковий характер
  - Потрібно подивитися, які існують зв'язки між різними змінними
  - Оскільки ми працюємо з клієнтами однієї компанії, нас не дуже цікавить, чи матиме місце цей зв'язок у всій популяції (тому точно не інференційний)
- Проте у компанії можуть з'являтися нові клієнти, і тому було б корисно таргетувати рекламу і для них також
- Відтак питання носить **прогнозний** характер
- Тому для відповіди на нього потрібно будувати моделі для прогнозування

Приклад формулювання дослідницького питання (3)

- Наостанок дослідник повинен з'ясувати, якого типу його питання
- Це дасть змогу зрозуміти, які методи аналізу ліпше використати
- На перший погляд може здатися, що питання має розвідковий характер
  - Потрібно подивитися, які існують зв'язки між різними змінними
  - Оскільки ми працюємо з клієнтами однієї компанії, нас не дуже цікавить, чи матиме місце цей зв'язок у всій популяції (тому точно не інференційний)
- Проте у компанії можуть з'являтися нові клієнти, і тому було б корисно таргетувати рекламу і для них також
- Відтак питання носить **прогнозний** характер
- Тому для відповіді на нього потрібно будувати моделі для прогнозування

Приклад формулювання дослідницького питання (3)

- Наостанок дослідник повинен з'ясувати, якого типу його питання
- Це дасть змогу зрозуміти, які методи аналізу ліпше використати
- На перший погляд може здатися, що питання має розвідковий характер
  - Потрібно подивитися, які існують зв'язки між різними змінними
  - Оскільки ми працюємо з клієнтами однієї компанії, нас не дуже цікавить, чи матиме місце цей зв'язок у всій популяції (тому точно не інференційний)
- Проте у компанії можуть з'являтися нові клієнти, і тому було б корисно таргетувати рекламу і для них також
- Відтак питання носить **прогнозний** характер
- Тому для відповіди на нього потрібно будувати моделі для прогнозування

План лекції

Вступні зауваги

- Короткий огляд мови R не ставить на меті дати вичерпний аналіз її можливостей
- Це набір окремих типових прикладів
- Докладну інформацію можна дістати з книжки *R for Data Science* (PDF версія доступна на диску)
  - Особливо корисні в першому читанні розділи 4, 5, 6, 10–11 (ви дізнаєтесь), 12, 15, 20
- Інші корисні книжки:
- Додаткову докладну інформацію можна дістати зі шпаргалок (cheat sheets) за посиланням, їх також викладено на диск
- Також завжди можна звернутися до сайтів Stack Overflow та Stack Exchange

Вступні зауваги

- Короткий огляд мови R не ставить на меті дати вичерпний аналіз її можливостей
- Це набір окремих типових прикладів
- Докладну інформацію можна дістати з книжки *R for Data Science* (PDF версія доступна на диску)
  - Особливо корисні в першому читанні розділи 4, 5, 6, 10–11 (по діагоналі), 12, 19, 20
- Інші корисні книжки:
- Додаткову докладну інформацію можна дістати зі шпаргалок (cheat sheets) за посиланням, їх також викладено на диск
- Також завжди можна звернутися до сайтів Stack Overflow та Stack Exchange

Вступні зауваги

- Короткий огляд мови R не ставить на меті дати вичерпний аналіз її можливостей
- Це набір окремих типових прикладів
- Докладну інформацію можна дістати з книжки *R for Data Science* (PDF версія доступна на диску)
  - Особливо корисні в першому читанні розділи 4, 5, 6, 10–11 (по діагоналі), 12, 19, 20
- Інші корисні книжки:
- Додаткову докладну інформацію можна дістати зі шпаргалок (cheat sheets) за посиланням, їх також викладено на диск
- Також завжди можна звернутися до сайтів Stack Overflow та Stack Exchange

Вступні зауваги

- Короткий огляд мови R не ставить на меті дати вичерпний аналіз її можливостей
- Це набір окремих типових прикладів
- Докладну інформацію можна дістати з книжки *R for Data Science* (PDF версія доступна на диску)
    - Особливо корисні в першому читанні розділи 4, 5, 6, 10–11 (по діагоналі), 12, 19, 20
- Інші корисні книжки:
    - Cookbook for R: набір конкретних поширених прикладів
    - Hands-On Programming with R
    - Українською можна почитати розділи 1–2 у підручнику Р. Майбороди
    - Та подивитися відео з двох плейлистів: І. Мірошниченка та КОМІМОВ
- Додаткову докладну інформацію можна дістати зі шпаргалок (cheat sheets) за посиланням, їх також викладено на диск
- Також завжди можна звернутися до сайтів Stack Overflow та Stack Exchange

Вступні зауваги

- Короткий огляд мови R не ставить на меті дати вичерпний аналіз її можливостей
- Це набір окремих типових прикладів
- Докладну інформацію можна дістати з книжки *R for Data Science* (PDF версія доступна на диску)
  - Особливо корисні в першому читанні розділи 4, 5, 6, 10–11 (по діагоналі), 12, 19, 20
- Інші корисні книжки:
  - Cookbook for R: набір конкретних поширених прикладів
  - Hands-On Programming with R
  - Українською можна почитати розділи 1–2 у підручнику Р. Майбороди
  - Та подивитися відео з двох плейлистів: І. Мірошниченка та AGMEMOD
- Додаткову докладну інформацію можна дістати зі шпаргалок (cheat sheets) за посиланням, їх також викладено на диск
- Також завжди можна звернутися до сайтів Stack Overflow та Stack Exchange

Вступні зауваги

- Короткий огляд мови R не ставить на меті дати вичерпний аналіз її можливостей
- Це набір окремих типових прикладів
- Докладну інформацію можна дістати з книжки *R for Data Science* (PDF версія доступна на диску)
  - Особливо корисні в першому читанні розділи 4, 5, 6, 10–11 (по діагоналі), 12, 19, 20
- Інші корисні книжки:
  - Cookbook for R: набір конкретних поширених прикладів
  - Hands-On Programming with R
  - Українською можна почитати розділи 1–2 у підручнику Р. Майбороди
  - Та подивитися відео з двох плейлистів: І. Мірошниченка та AGMEMOD
- Додаткову докладну інформацію можна дістати зі шпаргалок (cheat sheets) за посиланням, їх також викладено на диск
- Також завжди можна звернутися до сайтів Stack Overflow та Stack Exchange

Вступні зауваги

- Короткий огляд мови R не ставить на меті дати вичерпний аналіз її можливостей
- Це набір окремих типових прикладів
- Докладну інформацію можна дістати з книжки *R for Data Science* (PDF версія доступна на диску)
  - Особливо корисні в першому читанні розділи 4, 5, 6, 10–11 (по діагоналі), 12, 19, 20
- Інші корисні книжки:
  - Cookbook for R: набір конкретних поширених прикладів
  - Hands-On Programming with R
  - Українською можна почитати розділи 1–2 у підручнику Р. Майбороди
  - Та подивитися відео з двох плейлистів: І. Мірошниченка та AGMEMOD
- Додаткову докладну інформацію можна дістати зі шпаргалок (cheat sheets) за посиланням, їх також викладено на диск
- Також завжди можна звернутися до сайтів Stack Overflow та Stack Exchange

Вступні зауваги

- Короткий огляд мови R не ставить на меті дати вичерпний аналіз її можливостей
- Це набір окремих типових прикладів
- Докладну інформацію можна дістати з книжки *R for Data Science* (PDF версія доступна на диску)
  - Особливо корисні в першому читанні розділи 4, 5, 6, 10–11 (по діагоналі), 12, 19, 20
- Інші корисні книжки:
  - Cookbook for R: набір конкретних поширених прикладів
  - Hands-On Programming with R
  - Українською можна почитати розділи 1–2 у підручнику Р. Майбороди
  - Та подивитися відео з двох плейлистів: І. Мірошниченка та AGMEMOD
- Додаткову докладну інформацію можна дістати зі шпаргалок (cheat sheets) за посиланням, їх також викладено на диск
- Також завжди можна звернутися до сайтів Stack Overflow та Stack Exchange

Вступні зауваги

- Короткий огляд мови R не ставить на меті дати вичерпний аналіз її можливостей
- Це набір окремих типових прикладів
- Докладну інформацію можна дістати з книжки *R for Data Science* (PDF версія доступна на диску)
  - Особливо корисні в першому читанні розділи 4, 5, 6, 10–11 (по діагоналі), 12, 19, 20
- Інші корисні книжки:
  - Cookbook for R: набір конкретних поширених прикладів
  - Hands-On Programming with R
  - Українською можна почитати розділи 1–2 у підручнику Р. Майбороди
  - Та подивитися відео з двох плейлистів: І. Мірошниченка та AGMEMOD
- Додаткову докладну інформацію можна дістати зі шпаргалок (cheat sheets) за посиланням, їх також викладено на диск
- Також завжди можна звернутися до сайтів Stack Overflow та Stack Exchange

Вступні зауваги

- Короткий огляд мови R не ставить на меті дати вичерпний аналіз її можливостей
- Це набір окремих типових прикладів
- Докладну інформацію можна дістати з книжки *R for Data Science* (PDF версія доступна на диску)
  - Особливо корисні в першому читанні розділи 4, 5, 6, 10–11 (по діагоналі), 12, 19, 20
- Інші корисні книжки:
  - Cookbook for R: набір конкретних поширених прикладів
  - Hands-On Programming with R
  - Українською можна почитати розділи 1–2 у підручнику Р. Майбороди
  - Та подивитися відео з двох плейлистів: І. Мірошниченка та AGMEMOD
- Додаткову докладну інформацію можна дістати зі шпаргалок (cheat sheets) за посиланням, їх також викладено на диск
- Також завжди можна звернутися до сайтів Stack Overflow та Stack Exchange

Вступні зауваги

- Короткий огляд мови R не ставить на меті дати вичерпний аналіз її можливостей
- Це набір окремих типових прикладів
- Докладну інформацію можна дістати з книжки *R for Data Science* (PDF версія доступна на диску)
  - Особливо корисні в першому читанні розділи 4, 5, 6, 10–11 (по діагоналі), 12, 19, 20
- Інші корисні книжки:
  - Cookbook for R: набір конкретних поширених прикладів
  - Hands-On Programming with R
  - Українською можна почитати розділи 1–2 у підручнику Р. Майбороди
  - Та подивитися відео з двох плейлистів: І. Мірошниченка та AGMEMOD
- Додаткову докладну інформацію можна дістати зі шпаргалок (cheat sheets) за посиланням, їх також викладено на диск
- Також завжди можна звернутися до сайтів Stack Overflow та Stack Exchange

Що таке R

- R — це мова та середовище програмування для статистичного аналізу даних та їх візуалізації [6]
- R є некомерційним і вільно розповсюджується за ліцензією GNU General Public Licens
- Чому саме R?

[6] https://www.r-project.org/about.html

Що таке R

- R — це мова та середовище програмування для статистичного аналізу даних та їх візуалізації [6]
- R є некомерційним і вільно розповсюджується за ліцензією GNU General Public Licens
- Чому саме R?
    - Щоб студенти володіли ще одним інструментарієм
    - R заточено саме під статистичний аналіз даних, на відміну від того ж Python, який має ширшу екосистему для машинного навчання
    - Детальніше про відповідні порівняння можна почитати тут і тут
    - «Словнички» між R та Python можна подивитися тут і тут

[6] https://www.r-project.org/about.html

Що таке R

- R — це мова та середовище програмування для статистичного аналізу даних та їх візуалізації [6]
- R є некомерційним і вільно розповсюджується за ліцензією GNU General Public Licens
- Чому саме R?
  - Щоб студенти володіли ще одним інструментарієм
  - R заточено саме під статистичний аналіз даних, на відміну від того ж Python, який має ширшу екосистему для машинного навчання
  - Детальніше про відповідні порівняння можна почитати тут і тут
  - «Словнички» між R та Python можна подивитися тут і тут

[6] https://www.r-project.org/about.html

Що таке R

- R — це мова та середовище програмування для статистичного аналізу даних та їх візуалізації [6]
- R є некомерційним і вільно розповсюджується за ліцензією GNU General Public Licens
- Чому саме R?
  - Щоб студенти володіли ще одним інструментарієм
  - R заточено саме під статистичний аналіз даних, на відміну від того ж Python, який має ширшу екосистему для машинного навчання
  - Детальніше про відповідні порівняння можна почитати тут і тут
  - «Словнички» між R та Python можна подивитися тут і тут

[6] https://www.r-project.org/about.html

Що таке R

- R — це мова та середовище програмування для статистичного аналізу даних та їх візуалізації [6]
- R є некомерційним і вільно розповсюджується за ліцензією GNU General Public Licens
- Чому саме R?
  - Щоб студенти володіли ще одним інструментарієм
  - R заточено саме під статистичний аналіз даних, на відміну від того ж Python, який має ширшу екосистему для машинного навчання
  - Детальніше про відповідні порівняння можна почитати тут і тут
  - «Словнички» між R та Python можна подивитися тут і тут

[6] https://www.r-project.org/about.html

Що таке R

- R — це мова та середовище програмування для статистичного аналізу даних та їх візуалізації [6]
- R є некомерційним і вільно розповсюджується за ліцензією GNU General Public Licens
- Чому саме R?
  - Щоб студенти володіли ще одним інструментарієм
  - R заточено саме під статистичний аналіз даних, на відміну від того ж Python, який має ширшу екосистему для машинного навчання
  - Детальніше про відповідні порівняння можна почитати тут і тут
  - «Словнички» між R та Python можна подивитися тут і тут

[6] https://www.r-project.org/about.html

Що таке R

- R — це мова та середовище програмування для статистичного аналізу даних та їх візуалізації [6]
- R є некомерційним і вільно розповсюджується за ліцензією GNU General Public Licens
- Чому саме R?
  - Щоб студенти володіли ще одним інструментарієм
  - R заточено саме під статистичний аналіз даних, на відміну від того ж Python, який має ширшу екосистему для машинного навчання
  - Детальніше про відповідні порівняння можна почитати тут і тут
  - «Словнички» між R та Python можна подивитися тут і тут

[6] https://www.r-project.org/about.html

Установлення R, RStudio та важливих пакетів

- Для початку роботи з R його потрібно встановити
- Скачати інсталятор можна з сайту CRAN (The Comprehensive R Archive Network)
- Для зручної роботи в R потрібно встановити IDE (integrated development environment), яким у випадку R поза конкуренцією є RStudio
- Як і в Python, значну частину корисних функцій (а також наборів даних та файлів допомоги) реалізовано в **пакетах** (packages), які потрібно встановлювати окремо
- Інсталювати пакети можна або за допомогою інтерфейсу в RStudio, або безпосередньо функцією `install.packages`
- Нам відразу знадобиться пакет `tidyverse`, який можна встановити викликом `install.packages("tidyverse")`
- Інші пакети, які реалізують методи аналізу даних, які ми вивчатимемо далі, інсталюватимемо окремо

Установлення R, RStudio та важливих пакетів

- Для початку роботи з R його потрібно встановити
- Скачати інсталятор можна з сайту CRAN (The Comprehensive R Archive Network)
- Для зручної роботи в R потрібно встановити IDE (integrated development environment), яким у випадку R поза конкуренцією є RStudio
- Як і в Python, значну частину корисних функцій (а також наборів даних та файлів допомоги) реалізовано в **пакетах** (packages), які потрібно встановлювати окремо
- Інсталювати пакети можна або за допомогою інтерфейсу в RStudio, або безпосередньо функцією `install.packages`
- Нам відразу знадобиться пакет `tidyverse`, який можна встановити викликом `install.packages("tidyverse")`
- Інші пакети, які реалізують методи аналізу даних, які ми вивчатимемо далі, інсталюватимемо окремо

Установлення R, RStudio та важливих пакетів

- Для початку роботи з R його потрібно встановити
- Скачати інсталятор можна з сайту CRAN (The Comprehensive R Archive Network)
- Для зручної роботи в R потрібно встановити IDE (integrated development environment), яким у випадку R поза конкуренцією є RStudio
- Як і в Python, значну частину корисних функцій (а також наборів даних та файлів допомоги) реалізовано в **пакетах** (packages), які потрібно встановлювати окремо
- Інсталювати пакети можна або за допомогою інтерфейсу в RStudio, або безпосередньо функцією `install.packages`
- Нам відразу знадобиться пакет `tidyverse`, який можна встановити викликом `install.packages("tidyverse")`
- Інші пакети, які реалізують методи аналізу даних, які ми вивчатимемо далі, інсталюватимемо окремо

Установлення R, RStudio та важливих пакетів

- Для початку роботи з R його потрібно встановити
- Скачати інсталятор можна з сайту CRAN (The Comprehensive R Archive Network)
- Для зручної роботи в R потрібно встановити IDE (integrated development environment), яким у випадку R поза конкуренцією є RStudio
- Як і в Python, значну частину корисних функцій (а також наборів даних та файлів допомоги) реалізовано в **пакетах** (packages), які потрібно встановлювати окремо
- Інсталювати пакети можна або за допомогою інтерфейсу в RStudio, або безпосередньо функцією `install.packages`
- Нам відразу знадобиться пакет `tidyverse`, який можна встановити викликом `install.packages("tidyverse")`
- Інші пакети, які реалізують методи аналізу даних, які ми вивчатимемо далі, інсталюватимемо окремо

Установлення R, RStudio та важливих пакетів

- Для початку роботи з R його потрібно встановити
- Скачати інсталятор можна з сайту CRAN (The Comprehensive R Archive Network)
- Для зручної роботи в R потрібно встановити IDE (integrated development environment), яким у випадку R поза конкуренцією є RStudio
- Як і в Python, значну частину корисних функцій (а також наборів даних та файлів допомоги) реалізовано в **пакетах** (packages), які потрібно встановлювати окремо
- Інсталювати пакети можна або за допомогою інтерфейсу в RStudio, або безпосередньо функцією `install.packages`
- Нам відразу знадобиться пакет `tidyverse`, який можна встановити викликом `install.packages("tidyverse")`
- Інші пакети, які реалізують методи аналізу даних, які ми вивчатимемо далі, інсталюватимемо окремо

Установлення R, RStudio та важливих пакетів

- Для початку роботи з R його потрібно встановити
- Скачати інсталятор можна з сайту CRAN (The Comprehensive R Archive Network)
- Для зручної роботи в R потрібно встановити IDE (integrated development environment), яким у випадку R поза конкуренцією є RStudio
- Як і в Python, значну частину корисних функцій (а також наборів даних та файлів допомоги) реалізовано в **пакетах** (packages), які потрібно встановлювати окремо
- Інсталювати пакети можна або за допомогою інтерфейсу в RStudio, або безпосередньо функцією `install.packages`
- Нам відразу знадобиться пакет `tidyverse`, який можна встановити викликом `install.packages("tidyverse")`
- Інші пакети, які реалізують методи аналізу даних, які ми вивчатимемо далі, інсталюватимемо окремо

Установлення R, RStudio та важливих пакетів

- Для початку роботи з R його потрібно встановити
- Скачати інсталятор можна з сайту CRAN (The Comprehensive R Archive Network)
- Для зручної роботи в R потрібно встановити IDE (integrated development environment), яким у випадку R поза конкуренцією є RStudio
- Як і в Python, значну частину корисних функцій (а також наборів даних та файлів допомоги) реалізовано в **пакетах** (packages), які потрібно встановлювати окремо
- Інсталювати пакети можна або за допомогою інтерфейсу в RStudio, або безпосередньо функцією `install.packages`
- Нам відразу знадобиться пакет `tidyverse`, який можна встановити викликом `install.packages("tidyverse")`
- Інші пакети, які реалізують методи аналізу даних, які ми вивчатимемо далі, інсталюватимемо окремо

Базові операції (1)

- Аритметичні операції:

```
2 + 2
```

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

```
(2 - 2) * 3
```

```
## [1] 0
```

```
2 / (2 ^ 3)
```

```
## [1] 0.25
```

```
sin(pi / 2)
```

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

```
log(exp(1))
```

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

Базові операції (2)

- Логічні операції:

```
0.5 == sqrt(0.25)
```

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

```
5 != (10 / 2)
```

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

```
2 + 3 > 4
```

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

```
5 - 2 < sin(2)
```

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

```
3 >= 10 / 3
```

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

```
1 <= 1 + pi
```

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

```
sqrt(2) ^ 2 == 2
```

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

```
dplyr::near(sqrt(2) ^ 2, 2)
```

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

```
TRUE | FALSE
```

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

```
TRUE & FALSE
```

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

Змінні

- Як і в інших мовах, в R можна створювати **змінні** (variables)

```
a <- 2 + 2
```

- Назви повинні починатися з літер і повинні містити тільки літери, цифри, `_` та `.`
- Існують різні стилі називання змінних: `snake_case`, `camelCase`, або з точками як у `this.function`
- Для виведення вмісту на екран треба просто вказати назву змінної:

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

Змінні

- Як і в інших мовах, в R можна створювати **змінні** (variables)

```
a <- 2 + 2
```

- Назви повинні починатися з літер і повинні містити тільки літери, цифри, _ та .
- Існують різні стилі називання змінних: `snake_case`, `camelCase`, або з точками як у `this.function`
- Для виведення вмісту на екран треба просто вказати назву змінної:

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

Змінні

- Як і в інших мовах, в R можна створювати **змінні** (variables)

```
a <- 2 + 2
```

- Назви повинні починатися з літер і повинні містити тільки літери, цифри, _ та .
- Існують різні стилі називання змінних: `snake_case`, `camelCase`, або з точками як у `this.function`
- Для виведення вмісту на екран треба просто вказати назву змінної:

```
a
```

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

Змінні

- Як і в інших мовах, в R можна створювати **змінні** (variables)

```
a <- 2 + 2
```

- Назви повинні починатися з літер і повинні містити тільки літери, цифри, _ та .
- Існують різні стилі називання змінних: `snake_case`, `camelCase`, або з точками як у `this.function`
- Для виведення вмісту на екран треба просто вказати назву змінної:

```
a
```

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

Вектори (1)

- R — мова, у якій більшість функцій векторизовані
- Створити **вектор** (vector) можна за допомогою функції `c`
- Типи векторів, які нас цікавлять:

Вектори (1)

- R — мова, у якій більшість функцій векторизовані
- Створити **вектор** (vector) можна за допомогою функції `c`
- Типи векторів, які нас цікавлять:
  - **дійсні** (double):

```
vec_num <- c(1.1, 2.3, -0.4)
vec_num
```

```
## [1]  1.1  2.3 -0.4
```

```
typeof(vec_num)
```

```
## [1] "double"
```

  - **цілочисельні** (integer):

```
vec_int <- c(1, 2, 3)
typeof(vec_int)
```

```
## [1] "double"
```

```
vec_int <- c(1L, 2L, 3L)
typeof(vec_int)
```

```
## [1] "integer"
```

  - **логічні** (logical):

```
vec_logic <- c(TRUE, FALSE, TRUE)
typeof(vec_logic)
```

```
## [1] "logical"
```

  - **символьні** (character):

```
vec_char <- c("one", "two", "three")
typeof(vec_char)
```

```
## [1] "character"
```

Вектори (1)

- R — мова, у якій більшість функцій векторизовані
- Створити **вектор** (vector) можна за допомогою функції `c`
- Типи векторів, які нас цікавлять:
  - **дійсні** (double):

```
vec_num <- c(1.1, 2.3, -0.4)
vec_num
```

```
## [1]  1.1  2.3 -0.4
```

```
typeof(vec_num)
```

```
## [1] "double"
```

  - **цілочисельні** (integer):

```
vec_int <- c(1, 2, 3)
typeof(vec_int)
```

```
## [1] "double"
```

```
vec_int <- c(1L, 2L, 3L)
typeof(vec_int)
```

```
## [1] "integer"
```

  - **логічні** (logical):

```
vec_logic <- c(TRUE, FALSE, TRUE)
typeof(vec_logic)
```

```
## [1] "logical"
```

  - **символьні** (character):

```
vec_char <- c("Hello", "world", "!")
typeof(vec_char)
```

```
## [1] "character"
```

Вектори (1)

- R — мова, у якій більшість функцій векторизовані
- Створити **вектор** (vector) можна за допомогою функції `c`
- Типи векторів, які нас цікавлять:
  - **дійсні** (double):

```
vec_num <- c(1.1, 2.3, -0.4)
vec_num
```

```
## [1]  1.1  2.3 -0.4
```

```
typeof(vec_num)
```

```
## [1] "double"
```

  - **цілочисельні** (integer):

```
vec_int <- c(1, 2, 3)
typeof(vec_int)
```

```
## [1] "double"
```

```
vec_int <- c(1L, 2L, 3L)
typeof(vec_int)
```

```
## [1] "integer"
```

  - **логічні** (logical):

```
vec_logic <- c(TRUE, FALSE, TRUE)
typeof(vec_logic)
```

```
## [1] "logical"
```

  - **символьні** (character):

```
vec_char <- c("Hello", "world", "!")
typeof(vec_char)
```

```
## [1] "character"
```

Вектори (1)

- R — мова, у якій більшість функцій векторизовані
- Створити **вектор** (vector) можна за допомогою функції `c`
- Типи векторів, які нас цікавлять:
  - **дійсні** (double):

```
vec_num <- c(1.1, 2.3, -0.4)
vec_num
```

```
## [1]  1.1  2.3 -0.4
```

```
typeof(vec_num)
```

```
## [1] "double"
```

  - **цілочисельні** (integer):

```
vec_int <- c(1, 2, 3)
typeof(vec_int)
```

```
## [1] "double"
```

```
vec_int <- c(1L, 2L, 3L)
typeof(vec_int)
```

```
## [1] "integer"
```

  - **логічні** (logical):

```
vec_logic <- c(TRUE, FALSE, TRUE)
typeof(vec_logic)
```

```
## [1] "logical"
```

  - **символьні** (character):

```
vec_char <- c("Hello", "world", "!")
typeof(vec_char)
```

```
## [1] "character"
```

Вектори (1)

- R — мова, у якій більшість функцій векторизовані
- Створити **вектор** (vector) можна за допомогою функції `c`
- Типи векторів, які нас цікавлять:
  - **дійсні** (double):

```
vec_num <- c(1.1, 2.3, -0.4)
vec_num
```

```
## [1]  1.1  2.3 -0.4
```

```
typeof(vec_num)
```

```
## [1] "double"
```

  - **цілочисельні** (integer):

```
vec_int <- c(1, 2, 3)
typeof(vec_int)
```

```
## [1] "double"
```

```
vec_int <- c(1L, 2L, 3L)
typeof(vec_int)
```

```
## [1] "integer"
```

  - **логічні** (logical):

```
vec_logic <- c(TRUE, FALSE, TRUE)
typeof(vec_logic)
```

```
## [1] "logical"
```

  - **символьні** (character):

```
vec_char <- c("Hello", "world", "!")
typeof(vec_char)
```

```
## [1] "character"
```

Вектори (1)

- R — мова, у якій більшість функцій векторизовані
- Створити **вектор** (vector) можна за допомогою функції `c`
- Типи векторів, які нас цікавлять:
  - **дійсні** (double):

```
vec_num <- c(1.1, 2.3, -0.4)
vec_num
```

```
## [1]  1.1  2.3 -0.4
```

```
typeof(vec_num)
```

```
## [1] "double"
```

  - **цілочисельні** (integer):

```
vec_int <- c(1, 2, 3)
typeof(vec_int)
```

```
## [1] "double"
```

```
vec_int <- c(1L, 2L, 3L)
typeof(vec_int)
```

```
## [1] "integer"
```

  - **логічні** (logical):

```
vec_logic <- c(TRUE, FALSE, TRUE)
typeof(vec_logic)
```

```
## [1] "logical"
```

  - **символьні** (character):

```
vec_char <- c("Hello", "world", "!")
typeof(vec_char)
```

```
## [1] "character"
```

Вектори (2)

- В R переважна більшість функцій є векторизована:

```
vec_num <- c(1.1, 2.3, -0.4)
cos(vec_num)
```

```
## [1]  0.4535961 -0.6662760  0.9210610
```

```
exp(vec_num)
```

```
## [1] 3.004166 9.974182 0.670320
```

```
vec_num * vec_int
```

```
## [1]  1.1  4.6 -1.2
```

- Якщо один вектор коротший, то він повторюватиметься циклічно (т.зв. **recycling**)

```
a <- 1:10
b <- -5:-1
a + b
```

```
##  [1] -4 -2  0  2  4  1  3  5  7  9
```

- Це особливо корисно, якщо один із операндів — скаляр (фактично вектор довжини 1)
- Дуже зручно утворювати індексні вектори

```
a + b > 0
```

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

Вектори (2)

- В R переважна більшість функцій є векторизована:

```
vec_num <- c(1.1, 2.3, -0.4)
cos(vec_num)
```

```
## [1]  0.4535961 -0.6662760  0.9210610
```

```
exp(vec_num)
```

```
## [1] 3.004166 9.974182 0.670320
```

```
vec_num * vec_int
```

```
## [1]  1.1  4.6 -1.2
```

- Якщо один вектор коротший, то він повторюватиметься циклічно (т.зв. **recycling**)

```
a <- 1:10
b <- -5:-1
a + b
```

```
##  [1] -4 -2  0  2  4  1  3  5  7  9
```

- Це особливо корисно, якщо один із операндів — скаляр (фактично вектор довжини 1)
- Дуже зручно утворювати індексні вектори

```
a + b > 0
```

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

Вектори (2)

- В R переважна більшість функцій є векторизована:

```
vec_num <- c(1.1, 2.3, -0.4)
cos(vec_num)
```

```
## [1]  0.4535961 -0.6662760  0.9210610
```

```
exp(vec_num)
```

```
## [1] 3.004166 9.974182 0.670320
```

```
vec_num * vec_int
```

```
## [1]  1.1  4.6 -1.2
```

- Якщо один вектор коротший, то він повторюватиметься циклічно (т.зв. **recycling**)

```
a <- 1:10
b <- -5:-1
a + b
```

```
##  [1] -4 -2  0  2  4  1  3  5  7  9
```

- Це особливо корисно, якщо один із операндів — скаляр (фактично вектор довжини 1)
- Дуже зручно утворювати індексні вектори

```
a + b > 0
```

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

Вектори (2)

- В R переважна більшість функцій є векторизована:

```
vec_num <- c(1.1, 2.3, -0.4)
cos(vec_num)
```

```
## [1]  0.4535961 -0.6662760  0.9210610
```

```
exp(vec_num)
```

```
## [1] 3.004166 9.974182 0.670320
```

```
vec_num * vec_int
```

```
## [1]  1.1  4.6 -1.2
```

- Якщо один вектор коротший, то він повторюватиметься циклічно (т.зв. **recycling**)

```
a <- 1:10
b <- -5:-1
a + b
```

```
##  [1] -4 -2  0  2  4  1  3  5  7  9
```

- Це особливо корисно, якщо один із операндів — скаляр (фактично вектор довжини 1)
- Дуже зручно утворювати індексні вектори

```
a + b > 0
```

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

Спеціальні значення

- Існує спеціальне **логічне** значення `NA` (not available), яке відповідає **пропущеним даним**

```
NA > 5
```
```
## [1] NA
```
```
NA == 5
```
```
## [1] NA
```
```
NA + 5
```
```
## [1] NA
```
```
NA == NA
```
```
## [1] NA
```

- Для перевірки на відсутність застосовуємо функцію `is.na`
- Для дійсних значень також можливі спеціальні значення `Inf` (infinity) та `NaN` (not a number, невизначеність)

```
c(-1, 0, 1) / 0
```
```
## [1] -Inf  NaN  Inf
```

- Для перевірки на нескінченність застосовуємо функції `is.finite`, `is.infinite`
- Для перевірки на невизначеність застосовуємо функцію `is.nan`

```
is.finite(4 / 0)
```
```
## [1] FALSE
```
```
is.nan(Inf - Inf)
```
```
## [1] TRUE
```
```
is.nan(log(-1))
```
```
## [1] TRUE
```

Спеціальні значення

- Існує спеціальне **логічне** значення `NA` (not available), яке відповідає **пропущеним даним**

```
NA > 5
```

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

```
NA == 5
```

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

```
NA + 5
```

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

```
NA == NA
```

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

- Для перевірки на відсутність застосовуємо функцію `is.na`
- Для дійсних значень також можливі спеціальні значення `Inf` (infinity) та `NaN` (not a number, невизначеність)

```
c(-1, 0, 1) / 0
```

```
## [1] -Inf  NaN  Inf
```

- Для перевірки на нескінченність застосовуємо функції `is.finite`, `is.infinite`
- Для перевірки на невизначеність застосовуємо функцію `is.nan`

```
is.finite(4 / 0)
```

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

```
is.nan(Inf - Inf)
```

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

```
is.nan(log(-1))
```

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

Спеціальні значення

- Існує спеціальне **логічне** значення `NA` (not available), яке відповідає **пропущеним даним**

```
NA > 5
```

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

```
NA == 5
```

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

```
NA + 5
```

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

```
NA == NA
```

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

- Для перевірки на відсутність застосовуємо функцію `is.na`
- Для дійсних значень також можливі спеціальні значення `Inf` (infinity) та `NaN` (not a number, невизначеність)

```
c(-1, 0, 1) / 0
```

```
## [1] -Inf  NaN  Inf
```

- Для перевірки на нескінченність застосовуємо функції `is.finite`, `is.infinite`
- Для перевірки на невизначеність застосовуємо функцію `is.nan`

```
is.finite(4 / 0)
```

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

```
is.nan(Inf - Inf)
```

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

```
is.nan(log(-1))
```

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

Спеціальні значення

- Існує спеціальне **логічне** значення `NA` (not available), яке відповідає **пропущеним даним**

```
NA > 5
```

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

```
NA == 5
```

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

```
NA + 5
```

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

```
NA == NA
```

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

- Для перевірки на відсутність застосовуємо функцію `is.na`
- Для дійсних значень також можливі спеціальні значення `Inf` (infinity) та `NaN` (not a number, невизначеність)

```
c(-1, 0, 1) / 0
```

```
## [1] -Inf  NaN  Inf
```

- Для перевірки на нескінченність застосовуємо функції `is.finite`, `is.infinite`
- Для перевірки на невизначеність застосовуємо функцію `is.nan`

```
is.finite(4 / 0)
```

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

```
is.nan(Inf - Inf)
```

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

```
is.nan(log(-1))
```

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

Спеціальні значення

- Існує спеціальне **логічне** значення `NA` (not available), яке відповідає **пропущеним даним**

```
NA > 5
```

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

```
NA == 5
```

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

```
NA + 5
```

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

```
NA == NA
```

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

- Для перевірки на відсутність застосовуємо функцію `is.na`
- Для дійсних значень також можливі спеціальні значення `Inf` (infinity) та `NaN` (not a number, невизначеність)

```
c(-1, 0, 1) / 0
```

```
## [1] -Inf  NaN  Inf
```

- Для перевірки на нескінченність застосовуємо функції `is.finite`, `is.infinite`
- Для перевірки на невизначеність застосовуємо функцію `is.nan`

```
is.finite(4 / 0)
```

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

```
is.nan(Inf - Inf)
```

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

```
is.nan(log(-1))
```

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

Індексація векторів (1)

- Індексація векторів в R починається з 1

```
vec <- 10:1
vec[1]
```

```
## [1] 10
```

- Індексувати можна індексним вектором

```
vec[c(1, 4, 6)]
```

```
## [1] 10  7  5
```

```
vec[vec %% 2 == 0]
```

```
## [1] 10  8  6  4  2
```

```
vec[(vec > 3) & (vec < 7)]
```

```
## [1] 6 5 4
```

- Можна явно вказувати, які елементи викинути

```
vec[-c(1, 4, 6)]
```

```
## [1] 9 8 6 4 3 2 1
```

Індексація векторів (1)

- Індексація векторів в R починається з 1

```
vec <- 10:1
vec[1]
```

```
## [1] 10
```

- Індексувати можна індексним вектором

```
vec[c(1, 4, 6)]
```

```
## [1] 10  7  5
```

```
vec[vec %% 2 == 0]
```

```
## [1] 10  8  6  4  2
```

```
vec[(vec > 3) & (vec < 7)]
```

```
## [1] 6 5 4
```

- Можна явно вказувати, які елементи викинути

```
vec[-c(1, 4, 6)]
```

```
## [1] 9 8 6 4 3 2 1
```

Індексація векторів (1)

- Індексація векторів в R починається з 1

```
vec <- 10:1
vec[1]
```

```
## [1] 10
```

- Індексувати можна індексним вектором

```
vec[c(1, 4, 6)]
```

```
## [1] 10  7  5
```

```
vec[vec %% 2 == 0]
```

```
## [1] 10  8  6  4  2
```

```
vec[(vec > 3) & (vec < 7)]
```

```
## [1] 6 5 4
```

- Можна явно вказувати, які елементи викинути

```
vec[-c(1, 4, 6)]
```

```
## [1] 9 8 6 4 3 2 1
```

Індексація векторів (2)

- Елементам вектора можна присвоїти імена

```
vec <- 1:3
names(vec) <- c("a", "b", "c")
vec
```

```
## a b c
## 1 2 3
```

- Тоді індексацію можна робити також і за іменами

```
vec[c("a", "c")]
```

```
## a c
## 1 3
```

- Індексацію можна використовувати для внесення змін у вектор

```
vec <- 1:10
vec[vec > 5] <- 2
vec
```

```
##  [1] 1 2 3 4 5 2 2 2 2 2
```

Індексація векторів (2)

- Елементам вектора можна присвоїти імена

```
vec <- 1:3
names(vec) <- c("a", "b", "c")
vec
```

```
## a b c
## 1 2 3
```

- Тоді індексацію можна робити також і за іменами

```
vec[c("a", "c")]
```

```
## a c
## 1 3
```

- Індексацію можна використовувати для внесення змін у вектор

```
vec <- 1:10
vec[vec > 5] <- 2
vec
```

```
##  [1] 1 2 3 4 5 2 2 2 2 2
```

Індексація векторів (2)

- Елементам вектора можна присвоїти імена

```
vec <- 1:3
names(vec) <- c("a", "b", "c")
vec
```

```
## a b c
## 1 2 3
```

- Тоді індексацію можна робити також і за іменами

```
vec[c("a", "c")]
```

```
## a c
## 1 3
```

- Індексацію можна використовувати для внесення змін у вектор

```
vec <- 1:10
vec[vec > 5] <- 2
vec
```

```
##  [1] 1 2 3 4 5 2 2 2 2 2
```

Фактори

- Для моделювання **категорійних змінних** (categorical variables) в R застосовують спеціальний вид вектора — **фактор** (factor)
- Фактично це є цілочисельний вектор, але числа кодують окремі категорії (levels)

```
months <- c(
    "Jan", "Feb", "Mar", "Apr", "May", "Jun",
    "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
)
months_factor <- factor(months, levels = months)
months_factor
```

```
##  [1] Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
```

```
levels(months_factor)
```

```
##  [1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov" "Dec"
```

- Можна створити вектор і з частиною категорій

```
factor(c("Dec", "Jan", "Feb"), levels = months)
```

```
## [1] Dec Jan Feb
## Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
```

- Використання факторів унеможливлює описки, дає можливість сортувати категорії та спрощує роботу з ними

Фактори

- Для моделювання **категорійних змінних** (categorical variables) в R застосовують спеціальний вид вектора — **фактор** (factor)
- Фактично це є цілочисельний вектор, але числа кодують окремі категорії (levels)

```
months <- c(
    "Jan", "Feb", "Mar", "Apr", "May", "Jun",
    "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
)
months_factor <- factor(months, levels = months)
months_factor
```

```
##  [1] Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
```

```
levels(months_factor)
```

```
##  [1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov" "Dec"
```

- Можна створити вектор і з частиною категорій

```
factor(c("Dec", "Jan", "Feb"), levels = months)
```

```
## [1] Dec Jan Feb
## Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
```

- Використання факторів унеможливлює описки, дає можливість сортувати категорії та спрощує роботу з ними

Фактори

- Для моделювання **категорійних змінних** (categorical variables) в R застосовують спеціальний вид вектора — **фактор** (factor)
- Фактично це є цілочисельний вектор, але числа кодують окремі категорії (levels)

```
months <- c(
    "Jan", "Feb", "Mar", "Apr", "May", "Jun",
    "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
)
months_factor <- factor(months, levels = months)
months_factor
```

```
##  [1] Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
```

```
levels(months_factor)
```

```
##  [1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov" "Dec"
```

- Можна створити вектор і з частиною категорій

```
factor(c("Dec", "Jan", "Feb"), levels = months)
```

```
## [1] Dec Jan Feb
## Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
```

- Використання факторів унеможливлює описки, дає можливість сортувати категорії та спрощує роботу з ними

Фактори

- Для моделювання **категорійних змінних** (categorical variables) в R застосовують спеціальний вид вектора — **фактор** (factor)
- Фактично це є цілочисельний вектор, але числа кодують окремі категорії (levels)

```
months <- c(
    "Jan", "Feb", "Mar", "Apr", "May", "Jun",
    "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
)
months_factor <- factor(months, levels = months)
months_factor
```

```
##  [1] Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
```

```
levels(months_factor)
```

```
##  [1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov" "Dec"
```

- Можна створити вектор і з частиною категорій

```
factor(c("Dec", "Jan", "Feb"), levels = months)
```

```
## [1] Dec Jan Feb
## Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
```

- Використання факторів унеможливлює описки, дає можливість сортувати категорії та спрощує роботу з ними

Приведення типів (1)

- Вектори можна об’єднувати

```
a <- 1:3
c(a, a, a)
```

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

- Об’єднання векторів різних типів веде до неявного приведення типів

```
typeof(c(TRUE, 1L))
```

```
## [1] "integer"
```

```
typeof(c(1L, 1.5))
```

```
## [1] "double"
```

```
typeof(c(1.5, "a"))
```

```
## [1] "character"
```

- Неявне приведення типів зручно використовувати, щоб рахувати число чи частку елементів

```
x <- c(4, 10, 2, 7, 4, 9)
sum(x > 5)  # скільки чисел перевищує 5
```

```
## [1] 3
```

```
mean(x <= 5)  # частка чисел, не більших 5
```

```
## [1] 0.5
```

Приведення типів (1)

- Вектори можна об’єднувати

```
a <- 1:3
c(a, a, a)
```

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

- Об’єднання векторів різних типів веде до неявного приведення типів

```
typeof(c(TRUE, 1L))
```

```
## [1] "integer"
```

```
typeof(c(1L, 1.5))
```

```
## [1] "double"
```

```
typeof(c(1.5, "a"))
```

```
## [1] "character"
```

- Неявне приведення типів зручно використовувати, щоб рахувати число чи частку елементів

```
x <- c(4, 10, 2, 7, 4, 9)
sum(x > 5)  # скільки чисел перевищує 5
```

```
## [1] 3
```

```
mean(x <= 5)  # частка чисел, не більших 5
```

```
## [1] 0.5
```

Приведення типів (1)

- Вектори можна об'єднувати

```
a <- 1:3
c(a, a, a)
```

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

- Об'єднання векторів різних типів веде до неявного приведення типів

```
typeof(c(TRUE, 1L))
```

```
## [1] "integer"
```

```
typeof(c(1L, 1.5))
```

```
## [1] "double"
```

```
typeof(c(1.5, "a"))
```

```
## [1] "character"
```

- Неявне приведення типів зручно використовувати, щоб рахувати число чи частку елементів

```
x <- c(4, 10, 2, 7, 4, 9)
sum(x > 5)  # скільки чисел перевищує 5
```

```
## [1] 3
```

```
mean(x <= 5)  # частка чисел, не більших 5
```

```
## [1] 0.5
```

Приведення типів (2)

- Приведення типів можна виконати явно

```
vec <- c(1, 0, 1, 0)
as.character(vec)
```

```
## [1] "1" "0" "1" "0"
```

```
as.logical(vec)
```

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

```
vec2 <- c("2", "3.4", "TRUE")
as.numeric(vec2)
```

```
## [1] 2.0 3.4  NA
```

```
vec3 <- c("a", "b")
as.double(vec3)
```

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

- Також можна явно привести вектор до фактора

```
as.factor(c(1, 2, 3, 2, 3, 1))
```

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

Приведення типів (2)

- Приведення типів можна виконати явно

```
vec <- c(1, 0, 1, 0)
as.character(vec)
```

```
## [1] "1" "0" "1" "0"
```

```
as.logical(vec)
```

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

```
vec2 <- c("2", "3.4", "TRUE")
as.numeric(vec2)
```

```
## [1] 2.0 3.4  NA
```

```
vec3 <- c("a", "b")
as.double(vec3)
```

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

- Також можна явно привести вектор до фактора

```
as.factor(c(1, 2, 3, 2, 3, 1))
```

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

Рядки (1)

- Як і в інших мовах, **рядок** (string) в R — це послідовність символів
- Їх можна створювати і одинарними, і подвійними лапками:

```
string1 <- "Hello world!"
string2 <- 'Hello "world"!'
```

- Окремі символи потрібно **екранувати** (escape) за допомогою `\`:

```
c("\"", "\\", "\n", "\t")
```

```
## [1] "\"" "\\" "\n" "\t"
```

- Базовий пакет R містить функції для роботи з рядками, але сучасний підхід полягає у застосуванні функцій із пакету `stringr` (складова пакета `tidyverse`)
- Довжина

```
str_length(c("a", "Data analysis", NA))
```

```
## [1]  1 13 NA
```

- Конкатенація

```
str_c("a", "b")
```

```
## [1] "ab"
```

```
str_c("a", "b", sep = ", ")
```

```
## [1] "a, b"
```

```
str_c("pref-", c("1", NA, "2"), "-suff")
```

```
## [1] "pref-1-suff" NA              "pref-2-suff"
```

```
str_c("pref-", str_replace_na(c("1", NA, "2")), "-suff")
```

```
## [1] "pref-1-suff"  "pref-NA-suff" "pref-2-suff"
```

Рядки (1)

- Як і в інших мовах, **рядок** (string) в R — це послідовність символів
- Їх можна створювати і одинарними, і подвійними лапками:

```
string1 <- "Hello world!"
string2 <- 'Hello "world"!'
```

- Окремі символи потрібно **екранувати** (escape) за допомогою `\`:

```
c("\"", "\\", "\n", "\t")
```

```
## [1] "\"" "\\" "\n" "\t"
```

- Базовий пакет R містить функції для роботи з рядками, але сучасний підхід полягає у застосуванні функцій із пакету `stringr` (складова пакета `tidyverse`)
- Довжина

```
str_length(c("a", "Data analysis", NA))
```

```
## [1]  1 13 NA
```

- Конкатенація

```
str_c("a", "b")
```

```
## [1] "ab"
```

```
str_c("a", "b", sep = ", ")
```

```
## [1] "a, b"
```

```
str_c("pref-", c("1", NA, "2"), "-suff")
```

```
## [1] "pref-1-suff" NA            "pref-2-suff"
```

```
str_c("pref-", str_replace_na(c("1", NA, "2")), "-suff")
```

```
## [1] "pref-1-suff"  "pref-NA-suff" "pref-2-suff"
```

Рядки (1)

- Як і в інших мовах, **рядок** (string) в R — це послідовність символів
- Їх можна створювати і одинарними, і подвійними лапками:

```
string1 <- "Hello world!"
string2 <- 'Hello "world"!'
```

- Окремі символи потрібно **екранувати** (escape) за допомогою \:

```
c("\"", "\\", "\n", "\t")
```

```
## [1] "\"" "\\" "\n" "\t"
```

- Базовий пакет R містить функції для роботи з рядками, але сучасний підхід полягає у застосуванні функцій із пакету `stringr` (складова пакета `tidyverse`)
- Довжина

```
str_length(c("a", "Data analysis", NA))
```

```
## [1]  1 13 NA
```

- Конкатенація

```
str_c("a", "b")
```

```
## [1] "ab"
```

```
str_c("a", "b", sep = ", ")
```

```
## [1] "a, b"
```

```
str_c("pref-", c("1", NA, "2"), "-suff")
```

```
## [1] "pref-1-suff" NA            "pref-2-suff"
```

```
str_c("pref-", str_replace_na(c("1", NA, "2")), "-suff")
```

```
## [1] "pref-1-suff"  "pref-NA-suff" "pref-2-suff"
```

Рядки (1)

- Як і в інших мовах, **рядок** (string) в R — це послідовність символів
- Їх можна створювати і одинарними, і подвійними лапками:

```
string1 <- "Hello world!"
string2 <- 'Hello "world"!'
```

- Окремі символи потрібно **екранувати** (escape) за допомогою \:

```
c("\"", "\\", "\n", "\t")
```

```
## [1] "\"" "\\" "\n" "\t"
```

- Базовий пакет R містить функції для роботи з рядками, але сучасний підхід полягає у застосуванні функцій із пакету `stringr` (складова пакета `tidyverse`)
- Довжина

```
str_length(c("a", "Data analysis", NA))
```

```
## [1]  1 13 NA
```

- Конкатенація

```
str_c("a", "b")
```

```
## [1] "ab"
```

```
str_c("a", "b", sep = ", ")
```

```
## [1] "a, b"
```

```
str_c("pref-", c("1", NA, "2"), "-suff")
```

```
## [1] "pref-1-suff" NA            "pref-2-suff"
```

```
str_c("pref-", str_replace_na(c("1", NA, "2")), "-suff")
```

```
## [1] "pref-1-suff"  "pref-NA-suff" "pref-2-suff"
```

Рядки (1)

- Як і в інших мовах, **рядок** (string) в R — це послідовність символів
- Їх можна створювати і одинарними, і подвійними лапками:

```
string1 <- "Hello world!"
string2 <- 'Hello "world"!'
```

- Окремі символи потрібно **екранувати** (escape) за допомогою \:

```
c("\"", "\\", "\n", "\t")
```

```
## [1] "\"" "\\" "\n" "\t"
```

- Базовий пакет R містить функції для роботи з рядками, але сучасний підхід полягає у застосуванні функцій із пакету `stringr` (складова пакета `tidyverse`)
- Довжина

```
str_length(c("a", "Data analysis", NA))
```

```
## [1]  1 13 NA
```

- Конкатенація

```
str_c("a", "b")
```

```
## [1] "ab"
```

```
str_c("a", "b", sep = ", ")
```

```
## [1] "a, b"
```

```
str_c("pref-", c("1", NA, "2"), "-suff")
```

```
## [1] "pref-1-suff" NA            "pref-2-suff"
```

```
str_c("pref-", str_replace_na(c("1", NA, "2")), "-suff")
```

```
## [1] "pref-1-suff"  "pref-NA-suff" "pref-2-suff"
```

Рядки (1)

- Як і в інших мовах, **рядок** (string) в R — це послідовність символів
- Їх можна створювати і одинарними, і подвійними лапками:

```
string1 <- "Hello world!"
string2 <- 'Hello "world"!'
```

- Окремі символи потрібно **екранувати** (escape) за допомогою \:

```
c("\"", "\\", "\n", "\t")
```

```
## [1] "\"" "\\" "\n" "\t"
```

- Базовий пакет R містить функції для роботи з рядками, але сучасний підхід полягає у застосуванні функцій із пакету `stringr` (складова пакета `tidyverse`)
- Довжина

```
str_length(c("a", "Data analysis", NA))
```

```
## [1]  1 13 NA
```

- Конкатенація

```
str_c("a", "b")
```

```
## [1] "ab"
```

```
str_c("a", "b", sep = ", ")
```

```
## [1] "a, b"
```

```
str_c("pref-", c("1", NA, "2"), "-suff")
```

```
## [1] "pref-1-suff" NA            "pref-2-suff"
```

```
str_c("pref-", str_replace_na(c("1", NA, "2")), "-suff")
```

```
## [1] "pref-1-suff"  "pref-NA-suff" "pref-2-suff"
```

Рядки (2)

- Виділення підрядка

```
x <- c("Яблуко", "Банан")
str_sub(x, 1, 3)
```

```
## [1] "Ябл" "Бан"
```

```
str_sub(x, -3, -1)
```

```
## [1] "уко" "нан"
```

```
str_sub(x, 1, 6)
```

```
## [1] "Яблуко" "Банан"
```

- Зміна регістра

```
x <- c("ЯбЛуко", "БанАН")
str_to_lower(x)
```

```
## [1] "яблуко" "банан"
```

```
str_to_upper(x)
```

```
## [1] "ЯБЛУКО" "БАНАН"
```

```
str_to_title(x)
```

```
## [1] "Яблуко" "Банан"
```

- Сортування

```
x <- c("Ієремія", "Артем", "Яків")
str_sort(x)
```

```
## [1] "Артем"   "Ієремія" "Яків"
```

Рядки (2)

- Виділення підрядка

```
x <- c("Яблуко", "Банан")
str_sub(x, 1, 3)
```

```
## [1] "Ябл" "Бан"
```

```
str_sub(x, -3, -1)
```

```
## [1] "уко" "нан"
```

```
str_sub(x, 1, 6)
```

```
## [1] "Яблуко" "Банан"
```

- Зміна регістра

```
x <- c("ЯбЛуко", "БанАН")
str_to_lower(x)
```

```
## [1] "яблуко" "банан"
```

```
str_to_upper(x)
```

```
## [1] "ЯБЛУКО" "БАНАН"
```

```
str_to_title(x)
```

```
## [1] "Яблуко" "Банан"
```

- Сортування

```
x <- c("Ієремія", "Артем", "Яків")
str_sort(x)
```

```
## [1] "Артем"   "Ієремія" "Яків"
```

Рядки (2)

- Виділення підрядка

```
x <- c("Яблуко", "Банан")
str_sub(x, 1, 3)
```

```
## [1] "Ябл" "Бан"
```

```
str_sub(x, -3, -1)
```

```
## [1] "уко" "нан"
```

```
str_sub(x, 1, 6)
```

```
## [1] "Яблуко" "Банан"
```

- Зміна регістра

```
x <- c("ЯбЛуко", "БанАН")
str_to_lower(x)
```

```
## [1] "яблуко" "банан"
```

```
str_to_upper(x)
```

```
## [1] "ЯБЛУКО" "БАНАН"
```

```
str_to_title(x)
```

```
## [1] "Яблуко" "Банан"
```

- Сортування

```
x <- c("Ієремія", "Артем", "Яків")
str_sort(x)
```

```
## [1] "Артем"   "Ієремія" "Яків"
```

Списки (1)

- **Списки** (lists) в R використовують для зберігання різнотипних даних
- Для перегляду їхньої структури корисною є функція `str`

```
list_exmp <- list(
    "list_1" = list("a", 1L, 1.5, TRUE),
    "list_2" = list(Jan = 1, Feb = 2, Mar = 3),
    "list_3" = 3
)
str(list_exmp)
```

```
## List of 3
##  $ list_1:List of 4
##   ..$ : chr "a"
##   ..$ : int 1
##   ..$ : num 1.5
##   ..$ : logi TRUE
##  $ list_2:List of 3
##   ..$ Jan: num 1
##   ..$ Feb: num 2
##   ..$ Mar: num 3
##  $ list_3: num 3
```

Списки (1)

- **Списки** (lists) в R використовують для зберігання різнотипних даних
- Для перегляду їхньої структури корисною є функція `str`

```
list_exmp <- list(
    "list_1" = list("a", 1L, 1.5, TRUE),
    "list_2" = list(Jan = 1, Feb = 2, Mar = 3),
    "list_3" = 3
)
str(list_exmp)
```

```
## List of 3
##  $ list_1:List of 4
##   ..$ : chr "a"
##   ..$ : int 1
##   ..$ : num 1.5
##   ..$ : logi TRUE
##  $ list_2:List of 3
##   ..$ Jan: num 1
##   ..$ Feb: num 2
##   ..$ Mar: num 3
##  $ list_3: num 3
```

Списки (2)

- Індексування списків можна робити в декілька способів
- За допомогою [ ] можна дістати **список**

```
list_exmp[1:2]
```

```
## $list_1
## $list_1[[1]]
## [1] "a"
##
## $list_1[[2]]
## [1] 1
##
## $list_1[[3]]
## [1] 1.5
##
## $list_1[[4]]
## [1] TRUE
##
##
## $list_2
## $list_2$Jan
## [1] 1
##
## $list_2$Feb
## [1] 2
##
## $list_2$Mar
## [1] 3
```

```
list_exmp[3]
```

```
## $list_3
## [1] 3
```

Списки (2)

- Індексування списків можна робити в декілька способів
- За допомогою [ ] можна дістати **список**

```
list_exmp[1:2]
```

```
## $list_1
## $list_1[[1]]
## [1] "a"
##
## $list_1[[2]]
## [1] 1
##
## $list_1[[3]]
## [1] 1.5
##
## $list_1[[4]]
## [1] TRUE
##
##
## $list_2
## $list_2$Jan
## [1] 1
##
## $list_2$Feb
## [1] 2
##
## $list_2$Mar
## [1] 3
```

```
list_exmp[3]
```

```
## $list_3
## [1] 3
```

Списки (3)

- За допомогою `[ [ ] ]` можна дістати **елемент списку**

```
list_exmp[[2]]
```

```
## $Jan
## [1] 1
##
## $Feb
## [1] 2
##
## $Mar
## [1] 3
```

- `$` працює аналогічно, але особливо корисно для йменованих елементів

```
list_exmp$list_2$Feb
```

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

Списки (3)

- За допомогою [[ ]] можна дістати **елемент списку**

```
list_exmp[[2]]
```

```
## $Jan
## [1] 1
##
## $Feb
## [1] 2
##
## $Mar
## [1] 3
```

- $ працює аналогічно, але особливо корисно для йменованих елементів

```
list_exmp$list_2$Feb
```

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

Функції в R (1)

- R — багато в чому функціональна мова, пакетів із функціями величезна кількість
- Числові операції з векторами

```
vec <- c(-3, 0, 5, 5, -3, 8)
sum(vec)
```

```
## [1] 12
```

```
max(vec)
```

```
## [1] 8
```

```
min(vec)
```

```
## [1] -3
```

```
median(vec)
```

```
## [1] 2.5
```

```
mean(vec)
```

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

```
sd(vec)  # середньоквадратичне відхилення
```

```
## [1] 4.64758
```

```
summary(vec)  # максимум, мінімум, сподівання і квартилі
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -3.00   -2.25    2.50    2.00    5.00    8.00
```

```
vec_na <- c(NA, 1, 2)
mean(vec_na)
```

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

```
mean(vec_na, na.rm = TRUE)
```

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

Функції в R (1)

- R — багато в чому функціональна мова, пакетів із функціями величезна кількість
- Числові операції з векторами

```
vec <- c(-3, 0, 5, 5, -3, 8)
sum(vec)
```

```
## [1] 12
```

```
max(vec)
```

```
## [1] 8
```

```
min(vec)
```

```
## [1] -3
```

```
median(vec)
```

```
## [1] 2.5
```

```
mean(vec)
```

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

```
sd(vec)  # середньоквадратичне відхилення
```

```
## [1] 4.64758
```

```
summary(vec)  # максимум, мінімум, сподівання і квартилі
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -3.00   -2.25    2.50    2.00    5.00    8.00
```

```
vec_na <- c(NA, 1, 2)
mean(vec_na)
```

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

```
mean(vec_na, na.rm = TRUE)
```

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

Функції в R (2)

- Маніпуляції з векторами

```
length(vec)
```

```
## [1] 6
```

```
sort(vec)
```

```
## [1] -3 -3  0  5  5  8
```

```
unique(vec)
```

```
## [1] -3  0  5  8
```

```
rev(vec)  # задом наперед
```

```
## [1]  8 -3  5  5  0 -3
```

```
table(vec)  # скільки яких елементів
```

```
## vec
## -3  0  5  8
##  2  1  2  1
```

- Генерування векторів

```
seq(1, 5)
```

```
## [1] 1 2 3 4 5
```

```
seq(1, 5, by = 2)
```

```
## [1] 1 3 5
```

```
rep("a", 5)
```

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

```
rep(FALSE, 5)
```

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

Функції в R (2)

- Маніпуляції з векторами

```
length(vec)
```

```
## [1] 6
```

```
sort(vec)
```

```
## [1] -3 -3  0  5  5  8
```

```
unique(vec)
```

```
## [1] -3  0  5  8
```

```
rev(vec)  # задом наперед
```

```
## [1]  8 -3  5  5  0 -3
```

```
table(vec)  # скільки яких елементів
```

```
## vec
## -3  0  5  8
##  2  1  2  1
```

- Генерування векторів

```
seq(1, 5)
```

```
## [1] 1 2 3 4 5
```

```
seq(1, 5, by = 2)
```

```
## [1] 1 3 5
```

```
rep("a", 5)
```

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

```
rep(FALSE, 5)
```

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

Функції в R (3)

- Синтаксис для написання власних функцій:

```
add <- function(x, y){
    x + y
}

add(2, 3)
```

```
## [1] 5
```

- { ніколи не може бути першою в рядочку, за нею завжди повинен починатися новий рядок
- } завжди повинен стояти на окремому рядку
- Завжди потрібно робити відступи
- За замовчуванням повертають результат останнього рядка
- Щоб явно вказати, що повертають, потрібно використати `return`

```
add_and_subtract <- function(x, y){
    a <- x + y
    b <- x - y
    return(list("sum" = a, "diff" = b))
}

add_and_subtract(2, 3)
```

```
## $sum
## [1] 5
##
## $diff
## [1] -1
```

Функції в R (3)

- Синтаксис для написання власних функцій:

```
add <- function(x, y){
    x + y
}

add(2, 3)
```

```
## [1] 5
```

- { ніколи не може бути першою в рядочку, за нею завжди повинен починатися новий рядок
- } завжди повинен стояти на окремому рядку
- Завжди потрібно робити відступи
- За замовчуванням повертають результат останнього рядка
- Щоб явно вказати, що повертають, потрібно використати return

```
add_and_subtract <- function(x, y){
    a <- x + y
    b <- x - y
    return(list("sum" = a, "diff" = b))
}

add_and_subtract(2, 3)
```

```
## $sum
## [1] 5
##
## $diff
## [1] -1
```

Функції в R (3)

- Синтаксис для написання власних функцій:

```
add <- function(x, y){
    x + y
}

add(2, 3)
```

```
## [1] 5
```

- { ніколи не може бути першою в рядочку, за нею завжди повинен починатися новий рядок
- } завжди повинен стояти на окремому рядку
- Завжди потрібно робити відступи
- За замовчуванням повертають результат останнього рядка
- Щоб явно вказати, що повертають, потрібно використати `return`

```
add_and_subtract <- function(x, y){
    a <- x + y
    b <- x - y
    return(list("sum" = a, "diff" = b))
}

add_and_subtract(2, 3)
```

```
## $sum
## [1] 5
##
## $diff
## [1] -1
```

Функції в R (3)

- Синтаксис для написання власних функцій:

```
add <- function(x, y){
    x + y
}

add(2, 3)
```

```
## [1] 5
```

- { ніколи не може бути першою в рядочку, за нею завжди повинен починатися новий рядок
- } завжди повинен стояти на окремому рядку
- Завжди потрібно робити відступи
- За замовчуванням повертають результат останнього рядка
- Щоб явно вказати, що повертають, потрібно використати `return`

```
add_and_subtract <- function(x, y){
    a <- x + y
    b <- x - y
    return(list("sum" = a, "diff" = b))
}

add_and_subtract(2, 3)
```

```
## $sum
## [1] 5
##
## $diff
## [1] -1
```

Функції в R (3)

- Синтаксис для написання власних функцій:

```
add <- function(x, y){
    x + y
}

add(2, 3)
```

```
## [1] 5
```

- { ніколи не може бути першою в рядочку, за нею завжди повинен починатися новий рядок
- } завжди повинен стояти на окремому рядку
- Завжди потрібно робити відступи
- За замовчуванням повертають результат останнього рядка
- Щоб явно вказати, що повертають, потрібно використати `return`

```
add_and_subtract <- function(x, y){
    a <- x + y
    b <- x - y
    return(list("sum" = a, "diff" = b))
}

add_and_subtract(2, 3)
```

```
## $sum
## [1] 5
##
## $diff
## [1] -1
```

Функції в R (3)

- Синтаксис для написання власних функцій:

```
add <- function(x, y){
    x + y
}

add(2, 3)
```

```
## [1] 5
```

- { ніколи не може бути першою в рядочку, за нею завжди повинен починатися новий рядок
- } завжди повинен стояти на окремому рядку
- Завжди потрібно робити відступи
- За замовчуванням повертають результат останнього рядка
- Щоб явно вказати, що повертають, потрібно використати `return`

```
add_and_subtract <- function(x, y){
    a <- x + y
    b <- x - y
    return(list("sum" = a, "diff" = b))
}

add_and_subtract(2, 3)
```

```
## $sum
## [1] 5
##
## $diff
## [1] -1
```

Функції в R (4)

- Іменовані аргументи

```
zero_out <- function(vec, threshold){
    vec[abs(vec) > threshold] <- 0
    return(vec)
}

zero_out(c(1, 2, 3), 2)
```

```
## [1] 1 2 0
```

```
zero_out(c(1, 2, 3), threshold = 2)
```

```
## [1] 1 2 0
```

- Значення за замовчуванням

```
zero_out <- function(vec, threshold = 1){
    vec[abs(vec) > threshold] <- 0
    return(vec)
}

zero_out(c(1, 2, 3))
```

```
## [1] 1 0 0
```

```
zero_out(c(1, 2, 3), threshold = 2)
```

```
## [1] 1 2 0
```

- В окремих функціях може бути нефіксована кількість аргументів
- Наприклад, `sum` має сигнатуру `sum(..., na.rm = FALSE)`
- Тому можна викликати і `sum(1, 2, 3)`, і `sum(1:5)` тощо

Функції в R (4)

- Іменовані аргументи

```
zero_out <- function(vec, threshold){
    vec[abs(vec) > threshold] <- 0
    return(vec)
}

zero_out(c(1, 2, 3), 2)
```

```
## [1] 1 2 0
```

```
zero_out(c(1, 2, 3), threshold = 2)
```

```
## [1] 1 2 0
```

- Значення за замовчуванням

```
zero_out <- function(vec, threshold = 1){
    vec[abs(vec) > threshold] <- 0
    return(vec)
}

zero_out(c(1, 2, 3))
```

```
## [1] 1 0 0
```

```
zero_out(c(1, 2, 3), threshold = 2)
```

```
## [1] 1 2 0
```

- В окремих функціях може бути нефіксована кількість аргументів
- Наприклад, `sum` має сигнатуру `sum(..., na.rm = FALSE)`
- Тому можна викликати і `sum(1, 2, 3)`, і `sum(1:5)` тощо

Функції в R (4)

- Іменовані аргументи

```
zero_out <- function(vec, threshold){
    vec[abs(vec) > threshold] <- 0
    return(vec)
}

zero_out(c(1, 2, 3), 2)
```

```
## [1] 1 2 0
```

```
zero_out(c(1, 2, 3), threshold = 2)
```

```
## [1] 1 2 0
```

- Значення за замовчуванням

```
zero_out <- function(vec, threshold = 1){
    vec[abs(vec) > threshold] <- 0
    return(vec)
}

zero_out(c(1, 2, 3))
```

```
## [1] 1 0 0
```

```
zero_out(c(1, 2, 3), threshold = 2)
```

```
## [1] 1 2 0
```

- В окремих функціях може бути нефіксована кількість аргументів
- Наприклад, `sum` має сигнатуру `sum(..., na.rm = FALSE)`
- Тому можна викликати і `sum(1, 2, 3)`, і `sum(1:5)` тощо

Функції в R (4)

- Іменовані аргументи

```
zero_out <- function(vec, threshold){
    vec[abs(vec) > threshold] <- 0
    return(vec)
}

zero_out(c(1, 2, 3), 2)
```

```
## [1] 1 2 0
```

```
zero_out(c(1, 2, 3), threshold = 2)
```

```
## [1] 1 2 0
```

- Значення за замовчуванням

```
zero_out <- function(vec, threshold = 1){
    vec[abs(vec) > threshold] <- 0
    return(vec)
}

zero_out(c(1, 2, 3))
```

```
## [1] 1 0 0
```

```
zero_out(c(1, 2, 3), threshold = 2)
```

```
## [1] 1 2 0
```

- В окремих функціях може бути нефіксована кількість аргументів
- Наприклад, `sum` має сигнатуру `sum(..., na.rm = FALSE)`
- Тому можна викликати і `sum(1, 2, 3)`, і `sum(1:5)` тощо

Функції в R (4)

- Іменовані аргументи

```
zero_out <- function(vec, threshold){
    vec[abs(vec) > threshold] <- 0
    return(vec)
}

zero_out(c(1, 2, 3), 2)
```

```
## [1] 1 2 0
```

```
zero_out(c(1, 2, 3), threshold = 2)
```

```
## [1] 1 2 0
```

- Значення за замовчуванням

```
zero_out <- function(vec, threshold = 1){
    vec[abs(vec) > threshold] <- 0
    return(vec)
}

zero_out(c(1, 2, 3))
```

```
## [1] 1 0 0
```

```
zero_out(c(1, 2, 3), threshold = 2)
```

```
## [1] 1 2 0
```

- В окремих функціях може бути нефіксована кількість аргументів
- Наприклад, `sum` має сигнатуру `sum(..., na.rm = FALSE)`
- Тому можна викликати і `sum(1, 2, 3)`, і `sum(1:5)` тощо

Функції в R: корисні поради

- Ім'я функції повинно бути коротким, але змістовним
  - `remove_outliers` — ліпше, ніж `rem_out`
- Варто притримуватися однакового стилю в іменуваннях
  - `remove_outliers` не краще ніж `removeOutliers` чи `RemoveOutliers`
- Для функцій із однієї сім'ї потрібно використовувати однаковий префікс
- Потрібно бути дуже акуратним, щоб випадково не перевизначити базові речі в R, написавши щось типу `c <- 10`
- Не забувайте рясно коментувати код: початок коментаря позначає символ `#`
- Імена аргументів:

Функції в R: корисні поради

- Ім'я функції повинно бути коротким, але змістовним
  - `remove_outliers` — ліпше, ніж `rem_out`
- Варто притримуватися однакового стилю в іменуваннях
  - `remove_outliers` не може йти поряд з `removeMissingValues`
- Для функцій із однієї сім'ї потрібно використовувати однаковий префікс
- Потрібно бути дуже акуратним, щоб випадково не перевизначити базові речі в R, написавши щось типу `c <- 10`
- Не забувайте рясно коментувати код: початок коментаря позначає символ #
- Імена аргументів:

Функції в R: корисні поради

- Ім'я функції повинно бути коротким, але змістовним
  - `remove_outliers` — ліпше, ніж `rem_out`
- Варто притримуватися однакового стилю в іменуваннях
  - `remove_outliers` не може йти поряд з `removeMissingValues`
- Для функцій із однієї сім'ї потрібно використовувати однаковий префікс
  - `remove_outliers`, `remove_nas`, `remove_bycols` — ok
  - `outliers_remove`, `na_remove`, `bycols_remove` — weird
- Потрібно бути дуже акуратним, щоб випадково не перевизначити базові речі в R, написавши щось типу `c <- 10`
- Не забувайте рясно коментувати код: початок коментаря позначає символ `#`
- Імена аргументів:

Функції в R: корисні поради

- Ім'я функції повинно бути коротким, але змістовним
  - `remove_outliers` — ліпше, ніж `rem_out`
- Варто притримуватися однакового стилю в іменуваннях
  - `remove_outliers` не може йти поряд з `removeMissingValues`
- Для функцій із однієї сім'ї потрібно використовувати однаковий префікс
  - `remove_outliers`, `remove_na`, `remove_typos` — ОК
  - `outliers_remove`, `na_remove`, `typos_remove` — не ОК!
- Потрібно бути дуже акуратним, щоб випадково не перевизначити базові речі в R, написавши щось типу `c <- 10`
- Не забувайте рясно коментувати код: початок коментаря позначає символ `#`
- Імена аргументів:

Функції в R: корисні поради

- Ім'я функції повинно бути коротким, але змістовним
  - `remove_outliers` — ліпше, ніж `rem_out`
- Варто притримуватися однакового стилю в іменуваннях
  - `remove_outliers` не може йти поряд з `removeMissingValues`
- Для функцій із однієї сім'ї потрібно використовувати однаковий префікс
  - `remove_outliers`, `remove_na`, `remove_typos` — ОК
  - `outliers_remove`, `na_remove`, `typos_remove` — не ОК!
- Потрібно бути дуже акуратним, щоб випадково не перевизначити базові речі в R, написавши щось типу `c <- 10`
- Не забувайте рясно коментувати код: початок коментаря позначає символ `#`
- Імена аргументів:

Функції в R: корисні поради

- Ім'я функції повинно бути коротким, але змістовним
  - `remove_outliers` — ліпше, ніж `rem_out`
- Варто притримуватися однакового стилю в іменуваннях
  - `remove_outliers` не може йти поряд з `removeMissingValues`
- Для функцій із однієї сім'ї потрібно використовувати однаковий префікс
  - `remove_outliers`, `remove_na`, `remove_typos` — ОК
  - `outliers_remove`, `na_remove`, `typos_remove` — не ОК!
- Потрібно бути дуже акуратним, щоб випадково не перевизначити базові речі в R, написавши щось типу `c <- 10`
- Не забувайте рясно коментувати код: початок коментаря позначає символ #
- Імена аргументів:

Функції в R: корисні поради

- Ім'я функції повинно бути коротким, але змістовним
  - `remove_outliers` — ліпше, ніж `rem_out`
- Варто притримуватися однакового стилю в іменуваннях
  - `remove_outliers` не може йти поряд з `removeMissingValues`
- Для функцій із однієї сім'ї потрібно використовувати однаковий префікс
  - `remove_outliers`, `remove_na`, `remove_typos` — ОК
  - `outliers_remove`, `na_remove`, `typos_remove` — не ОК!
- Потрібно бути дуже акуратним, щоб випадково не перевизначити базові речі в R, написавши щось типу `c <- 10`
- Не забувайте рясно коментувати код: початок коментаря позначає символ `#`
- Імена аргументів:

Функції в R: корисні поради

- Ім'я функції повинно бути коротким, але змістовним
  - `remove_outliers` — ліпше, ніж `rem_out`
- Варто притримуватися однакового стилю в іменуваннях
  - `remove_outliers` не може йти поряд з `removeMissingValues`
- Для функцій із однієї сім'ї потрібно використовувати однаковий префікс
  - `remove_outliers`, `remove_na`, `remove_typos` — ОК
  - `outliers_remove`, `na_remove`, `typos_remove` — не ОК!
- Потрібно бути дуже акуратним, щоб випадково не перевизначити базові речі в R, написавши щось типу `c <- 10`
- Не забувайте рясно коментувати код: початок коментаря позначає символ #
- Імена аргументів:

Функції в R: корисні поради

- Ім'я функції повинно бути коротким, але змістовним
  - `remove_outliers` — ліпше, ніж `rem_out`
- Варто притримуватися однакового стилю в іменуваннях
  - `remove_outliers` не може йти поряд з `removeMissingValues`
- Для функцій із однієї сім'ї потрібно використовувати однаковий префікс
  - `remove_outliers`, `remove_na`, `remove_typos` — ОК
  - `outliers_remove`, `na_remove`, `typos_remove` — не ОК!
- Потрібно бути дуже акуратним, щоб випадково не перевизначити базові речі в R, написавши щось типу `c <- 10`
- Не забувайте рясно коментувати код: початок коментаря позначає символ `#`
- Імена аргументів:
  - x, y, z: вектори
  - w: вектор вагових коефіцієнтів
  - df: датафрейм (розглядатимемо далі)
  - i, j: індекси (як правило, рядків і стовпців)
  - n: довжина вектора або число рядків
  - p: число стовпців

Функції в R: корисні поради

- Ім'я функції повинно бути коротким, але змістовним
  - `remove_outliers` — ліпше, ніж `rem_out`
- Варто притримуватися однакового стилю в іменуваннях
  - `remove_outliers` не може йти поряд з `removeMissingValues`
- Для функцій із однієї сім'ї потрібно використовувати однаковий префікс
  - `remove_outliers`, `remove_na`, `remove_typos` — ОК
  - `outliers_remove`, `na_remove`, `typos_remove` — не ОК!
- Потрібно бути дуже акуратним, щоб випадково не перевизначити базові речі в R, написавши щось типу `c <- 10`
- Не забувайте рясно коментувати код: початок коментаря позначає символ #
- Імена аргументів:
  - `x`, `y`, `z`: вектори
  - `w`: вектор вагових коефіцієнтів
  - `df`: датафрейм (розглядатимемо далі)
  - `i`, `j`: індекси (як правило, рядок і стовпець)
  - `n`: довжина вектора або число рядків
  - `p`: число стовпців

Функції в R: корисні поради

- Ім'я функції повинно бути коротким, але змістовним
  - `remove_outliers` — ліпше, ніж `rem_out`
- Варто притримуватися однакового стилю в іменуваннях
  - `remove_outliers` не може йти поряд з `removeMissingValues`
- Для функцій із однієї сім'ї потрібно використовувати однаковий префікс
  - `remove_outliers`, `remove_na`, `remove_typos` — ОК
  - `outliers_remove`, `na_remove`, `typos_remove` — не ОК!
- Потрібно бути дуже акуратним, щоб випадково не перевизначити базові речі в R, написавши щось типу `c <- 10`
- Не забувайте рясно коментувати код: початок коментаря позначає символ #
- Імена аргументів:
  - `x`, `y`, `z`: вектори
  - `w`: вектор вагових коефіцієнтів
  - `df`: датафрейм (розглядатимемо далі)
  - `i`, `j`: індекси (як правило, рядок і стовпець)
  - `n`: довжина вектора або число рядків
  - `p`: число стовпців

Функції в R: корисні поради

- Ім'я функції повинно бути коротким, але змістовним
  - `remove_outliers` — ліпше, ніж `rem_out`
- Варто притримуватися однакового стилю в іменуваннях
  - `remove_outliers` не може йти поряд з `removeMissingValues`
- Для функцій із однієї сім'ї потрібно використовувати однаковий префікс
  - `remove_outliers`, `remove_na`, `remove_typos` — ОК
  - `outliers_remove`, `na_remove`, `typos_remove` — не ОК!
- Потрібно бути дуже акуратним, щоб випадково не перевизначити базові речі в R, написавши щось типу `c <- 10`
- Не забувайте рясно коментувати код: початок коментаря позначає символ #
- Імена аргументів:
  - `x`, `y`, `z`: вектори
  - `w`: вектор вагових коефіцієнтів
  - `df`: датафрейм (розглядатимемо далі)
  - `i`, `j`: індекси (як правило, рядок і стовпець)
  - `n`: довжина вектора або число рядків
  - `p`: число стовпців

Функції в R: корисні поради

- Ім'я функції повинно бути коротким, але змістовним
  - `remove_outliers` — ліпше, ніж `rem_out`
- Варто притримуватися однакового стилю в іменуваннях
  - `remove_outliers` не може йти поряд з `removeMissingValues`
- Для функцій із однієї сім'ї потрібно використовувати однаковий префікс
  - `remove_outliers`, `remove_na`, `remove_typos` — ОК
  - `outliers_remove`, `na_remove`, `typos_remove` — не ОК!
- Потрібно бути дуже акуратним, щоб випадково не перевизначити базові речі в R, написавши щось типу `c <- 10`
- Не забувайте рясно коментувати код: початок коментаря позначає символ #
- Імена аргументів:
  - `x`, `y`, `z`: вектори
  - `w`: вектор вагових коефіцієнтів
  - `df`: датафрейм (розглядатимемо далі)
  - `i`, `j`: індекси (як правило, рядок і стовпець)
  - `n`: довжина вектора або число рядків
  - `p`: число стовпців

Функції в R: корисні поради

- Ім'я функції повинно бути коротким, але змістовним
  - `remove_outliers` — ліпше, ніж `rem_out`
- Варто притримуватися однакового стилю в іменуваннях
  - `remove_outliers` не може йти поряд з `removeMissingValues`
- Для функцій із однієї сім'ї потрібно використовувати однаковий префікс
  - `remove_outliers`, `remove_na`, `remove_typos` — ОК
  - `outliers_remove`, `na_remove`, `typos_remove` — не ОК!
- Потрібно бути дуже акуратним, щоб випадково не перевизначити базові речі в R, написавши щось типу `c <- 10`
- Не забувайте рясно коментувати код: початок коментаря позначає символ #
- Імена аргументів:
  - `x`, `y`, `z`: вектори
  - `w`: вектор вагових коефіцієнтів
  - `df`: датафрейм (розглядатимемо далі)
  - `i`, `j`: індекси (як правило, рядок і стовпець)
  - `n`: довжина вектора або число рядків
  - `p`: число стовпців

Функції в R: корисні поради

- Ім'я функції повинно бути коротким, але змістовним
  - `remove_outliers` — ліпше, ніж `rem_out`
- Варто притримуватися однакового стилю в іменуваннях
  - `remove_outliers` не може йти поряд з `removeMissingValues`
- Для функцій із однієї сім'ї потрібно використовувати однаковий префікс
  - `remove_outliers`, `remove_na`, `remove_typos` — ОК
  - `outliers_remove`, `na_remove`, `typos_remove` — не ОК!
- Потрібно бути дуже акуратним, щоб випадково не перевизначити базові речі в R, написавши щось типу `c <- 10`
- Не забувайте рясно коментувати код: початок коментаря позначає символ #
- Імена аргументів:
  - `x`, `y`, `z`: вектори
  - `w`: вектор вагових коефіцієнтів
  - `df`: датафрейм (розглядатимемо далі)
  - `i`, `j`: індекси (як правило, рядок і стовпець)
  - `n`: довжина вектора або число рядків
  - `p`: число стовпців

Функції в R: корисні поради

- Ім'я функції повинно бути коротким, але змістовним
  - `remove_outliers` — ліпше, ніж `rem_out`
- Варто притримуватися однакового стилю в іменуваннях
  - `remove_outliers` не може йти поряд з `removeMissingValues`
- Для функцій із однієї сім'ї потрібно використовувати однаковий префікс
  - `remove_outliers`, `remove_na`, `remove_typos` — ОК
  - `outliers_remove`, `na_remove`, `typos_remove` — не ОК!
- Потрібно бути дуже акуратним, щоб випадково не перевизначити базові речі в R, написавши щось типу `c <- 10`
- Не забувайте рясно коментувати код: початок коментаря позначає символ #
- Імена аргументів:
  - `x`, `y`, `z`: вектори
  - `w`: вектор вагових коефіцієнтів
  - `df`: датафрейм (розглядатимемо далі)
  - `i`, `j`: індекси (як правило, рядок і стовпець)
  - `n`: довжина вектора або число рядків
  - `p`: число стовпців

Умовний оператор (1)

- Як і в інших мовах, умовне виконання коду можна досягти за допомогою виразу `if`

```
x <- 2
if (x > 0) {
    y <- sqrt(2)
} else {
    y <- 0
}

y
```

```
## [1] 1.414214
```

- `{` ніколи не може бути першою в рядочку, за нею завжди повинен починатися новий рядок
- `}` завжди повинен стояти на окремому рядку, якщо тільки за ним не йде `else`
- Для складних умов можна використовувати логічні оператори `&&` (і), `||` (чи)
- Завжди потрібно робити відступи
- Умова повинна бути скалярною
- `==` векторизована, тому порівняння вектора зі скаляром дасть вектор
- Ліпше використовувати `all()` та `any()`

```
x <- c(1, 0, 0)
if (any(x == 0)){
    "Є нульові елементи"
} else {
    "Немає нульових елементів"
}
```

```
## [1] "Є нульові елементи"
```

Умовний оператор (1)

- Як і в інших мовах, умовне виконання коду можна досягти за допомогою виразу `if`

```
x <- 2
if (x > 0) {
    y <- sqrt(2)
} else {
    y <- 0
}

y
```

```
## [1] 1.414214
```

- { ніколи не може бути першою в рядочку, за нею завжди повинен починатися новий рядок
- } завжди повинен стояти на окремому рядку, якщо тільки за ним не йде `else`
- Для складних умов можна використовувати логічні оператори `&&` (і), `||` (чи)
- Завжди потрібно робити відступи
- Умова повинна бути скалярною
- `==` векторизована, тому порівняння вектора зі скаляром дасть вектор
- Ліпше використовувати `all()` та `any()`

```
x <- c(1, 0, 0)
if (any(x == 0)){
    "Є нульові елементи"
} else {
    "Немає нульових елементів"
}
```

```
## [1] "Є нульові елементи"
```

Умовний оператор (1)

- Як і в інших мовах, умовне виконання коду можна досягти за допомогою виразу `if`

```
x <- 2
if (x > 0) {
    y <- sqrt(2)
} else {
    y <- 0
}

y
```

```
## [1] 1.414214
```

- { ніколи не може бути першою в рядочку, за нею завжди повинен починатися новий рядок
- } завжди повинен стояти на окремому рядку, якщо тільки за ним не йде `else`
- Для складних умов можна використовувати логічні оператори `&&` (і), `||` (чи)
- Завжди потрібно робити відступи
- Умова повинна бути скалярною
- `==` векторизована, тому порівняння вектора зі скаляром дасть вектор
- Ліпше використовувати `all()` та `any()`

```
x <- c(1, 0, 0)
if (any(x == 0)){
    "Є нульові елементи"
} else {
    "Немає нульових елементів"
}
```

```
## [1] "Є нульові елементи"
```

Умовний оператор (1)

- Як і в інших мовах, умовне виконання коду можна досягти за допомогою виразу `if`

```
x <- 2
if (x > 0) {
    y <- sqrt(2)
} else {
    y <- 0
}

y
```

```
## [1] 1.414214
```

- { ніколи не може бути першою в рядочку, за нею завжди повинен починатися новий рядок
- } завжди повинен стояти на окремому рядку, якщо тільки за ним не йде `else`
- Для складних умов можна використовувати логічні оператори `&&` (і), `||` (чи)
- Завжди потрібно робити відступи
- Умова повинна бути скалярною
- == векторизована, тому порівняння вектора зі скаляром дасть вектор
- Ліпше використовувати `all()` та `any()`

```
x <- c(1, 0, 0)
if (any(x == 0)){
    "Є нульові елементи"
} else {
    "Немає нульових елементів"
}
```

```
## [1] "Є нульові елементи"
```

Умовний оператор (1)

- Як і в інших мовах, умовне виконання коду можна досягти за допомогою виразу `if`

```
x <- 2
if (x > 0) {
    y <- sqrt(2)
} else {
    y <- 0
}

y
```

```
## [1] 1.414214
```

- { ніколи не може бути першою в рядочку, за нею завжди повинен починатися новий рядок
- } завжди повинен стояти на окремому рядку, якщо тільки за ним не йде `else`
- Для складних умов можна використовувати логічні оператори `&&` (і), `||` (чи)
- Завжди потрібно робити відступи
- Умова повинна бути скалярною
- == векторизована, тому порівняння вектора зі скаляром дасть вектор
- Ліпше використовувати `all()` та `any()`

```
x <- c(1, 0, 0)
if (any(x == 0)){
    "Є нульові елементи"
} else {
    "Немає нульових елементів"
}

## [1] "Є нульові елементи"
```

Умовний оператор (1)

- Як і в інших мовах, умовне виконання коду можна досягти за допомогою виразу `if`

```
x <- 2
if (x > 0) {
    y <- sqrt(2)
} else {
    y <- 0
}

y
```

```
## [1] 1.414214
```

- { ніколи не може бути першою в рядочку, за нею завжди повинен починатися новий рядок
- } завжди повинен стояти на окремому рядку, якщо тільки за ним не йде `else`
- Для складних умов можна використовувати логічні оператори `&&` (і), `||` (чи)
- Завжди потрібно робити відступи
- Умова повинна бути скалярною
- == векторизована, тому порівняння вектора зі скаляром дасть вектор
- Ліпше використовувати `all()` та `any()`

```
x <- c(1, 0, 0)
if (any(x == 0)){
    "Є нульові елементи"
} else {
    "Немає нульових елементів"
}
```

```
## [1] "Є нульові елементи"
```

Умовний оператор (1)

- Як і в інших мовах, умовне виконання коду можна досягти за допомогою виразу `if`

```
x <- 2
if (x > 0) {
    y <- sqrt(2)
} else {
    y <- 0
}

y
```

```
## [1] 1.414214
```

- { ніколи не може бути першою в рядочку, за нею завжди повинен починатися новий рядок
- } завжди повинен стояти на окремому рядку, якщо тільки за ним не йде `else`
- Для складних умов можна використовувати логічні оператори `&&` (і), `||` (чи)
- Завжди потрібно робити відступи
- Умова повинна бути скалярною
- == векторизована, тому порівняння вектора зі скаляром дасть вектор
- Ліпше використовувати `all()` та `any()`

```
x <- c(1, 0, 0)
if (any(x == 0)){
    "Є нульові елементи"
} else {
    "Немає нульових елементів"
}
```

```
## [1] "Є нульові елементи"
```

Умовний оператор (1)

- Як і в інших мовах, умовне виконання коду можна досягти за допомогою виразу `if`

```
x <- 2
if (x > 0) {
    y <- sqrt(2)
} else {
    y <- 0
}

y
```

```
## [1] 1.414214
```

- { ніколи не може бути першою в рядочку, за нею завжди повинен починатися новий рядок
- } завжди повинен стояти на окремому рядку, якщо тільки за ним не йде `else`
- Для складних умов можна використовувати логічні оператори `&&` (і), `||` (чи)
- Завжди потрібно робити відступи
- Умова повинна бути скалярною
- == векторизована, тому порівняння вектора зі скаляром дасть вектор
- Ліпше використовувати `all()` та `any()`

```
x <- c(1, 0, 0)
if (any(x == 0)){
    "Є нульові елементи"
} else {
    "Немає нульових елементів"
}
```

```
## [1] "Є нульові елементи"
```

Умовний оператор (2)

- Якщо умов декілька, можна використовувати або вкладені оператори

```
if (all(x == 0)) {
    "Усі елементи нульові"
} else if (any(x == 0)) {
    "Деякі елементи нульові"
} else {
    "Немає нульових елементів"
}
```

```
## [1] "Деякі елементи нульові"
```

- ...або фукнцію `switch()`

```
centre <- function(x, type) {
    switch(type,
        mean = mean(x),
        median = median(x)
    )
}

x <- c(-1, -1.8, 0.9, -0.4, -0.8, -.1, 1.2, 1.3)
centre(x, "mean")
```

```
## [1] -0.0875
```

```
centre(x, "median")
```

```
## [1] -0.25
```

Умовний оператор (2)

- Якщо умов декілька, можна використовувати або вкладені оператори

```
if (all(x == 0)) {
    "Усі елементи нульові"
} else if (any(x == 0)) {
    "Деякі елементи нульові"
} else {
    "Немає нульових елементів"
}
```

```
## [1] "Деякі елементи нульові"
```

- ...або фукнцію `switch()`

```
centre <- function(x, type) {
    switch(type,
        mean = mean(x),
        median = median(x)
    )
}

x <- c(-1, -1.8, 0.9, -0.4, -0.8, -.1, 1.2, 1.3)
centre(x, "mean")
```

```
## [1] -0.0875
```

```
centre(x, "median")
```

```
## [1] -0.25
```

Цикли

- Як і в інших мовах, R підтримує цикли `for` і `while`

```r
x <- c(1, 2, 3, 4, 5)
for (i in seq_along(x)) {   # те саме, що й "i in 1:length(x)"
    x[i] <- x[i] * 2
}
```

- Але такі цикли зовсім непритаманні R, адже ця мова дуже векторизована
- Значно швидше і зрозуміліше було б написати `x <- 2*x`
- Те саме стосується і складніших структур даних
- Функції можна застосовувати до всіх елементів векторів, списків і т.п. за допомогою спеціальних функцій, наприклад, `lapply`

```r
lapply(list_exmp, length)
```

```
## $list_1
## [1] 4
##
## $list_2
## [1] 3
##
## $list_3
## [1] 1
```

Цикли

- Як і в інших мовах, R підтримує цикли `for` і `while`

```r
x <- c(1, 2, 3, 4, 5)
for (i in seq_along(x)) {   # те саме, що й "i in 1:length(x)"
    x[i] <- x[i] * 2
}
```

- Але такі цикли зовсім непритаманні R, адже ця мова дуже векторизована
- Значно швидше і зрозуміліше було б написати `x <- 2*x`
- Те саме стосується і складніших структур даних
- Функції можна застосовувати до всіх елементів векторів, списків і т.п. за допомогою спеціальних функцій, наприклад, `lapply`

```r
lapply(list_exmp, length)
```

```
## $list_1
## [1] 4
##
## $list_2
## [1] 3
##
## $list_3
## [1] 1
```

Цикли

- Як і в інших мовах, R підтримує цикли `for` і `while`

```
x <- c(1, 2, 3, 4, 5)
for (i in seq_along(x)) {   # те саме, що й "i in 1:length(x)"
    x[i] <- x[i] * 2
}
```

- Але такі цикли зовсім непритаманні R, адже ця мова дуже векторизована
- Значно швидше і зрозуміліше було б написати `x <- 2*x`
- Те саме стосується і складніших структур даних
- Функції можна застосовувати до всіх елементів векторів, списків і т.п. за допомогою спеціальних функцій, наприклад, `lapply`

```
lapply(list_exmp, length)
```

```
## $list_1
## [1] 4
##
## $list_2
## [1] 3
##
## $list_3
## [1] 1
```

Цикли

- Як і в інших мовах, R підтримує цикли `for` і `while`

```
x <- c(1, 2, 3, 4, 5)
for (i in seq_along(x)) {   # те саме, що й "i in 1:length(x)"
    x[i] <- x[i] * 2
}
```

- Але такі цикли зовсім непритаманні R, адже ця мова дуже векторизована
- Значно швидше і зрозуміліше було б написати `x <- 2*x`
- Те саме стосується і складніших структур даних
- Функції можна застосовувати до всіх елементів векторів, списків і т.п. за допомогою спеціальних функцій, наприклад, `lapply`

```
lapply(list_exmp, length)
```

```
## $list_1
## [1] 4
##
## $list_2
## [1] 3
##
## $list_3
## [1] 1
```

Цикли

- Як і в інших мовах, R підтримує цикли `for` і `while`

```
x <- c(1, 2, 3, 4, 5)
for (i in seq_along(x)) {   # те саме, що й "i in 1:length(x)"
    x[i] <- x[i] * 2
}
```

- Але такі цикли зовсім непритаманні R, адже ця мова дуже векторизована
- Значно швидше і зрозуміліше було б написати `x <- 2*x`
- Те саме стосується і складніших структур даних
- Функції можна застосовувати до всіх елементів векторів, списків і т.п. за допомогою спеціальних функцій, наприклад, `lapply`

```
lapply(list_exmp, length)
```

```
## $list_1
## [1] 4
##
## $list_2
## [1] 3
##
## $list_3
## [1] 1
```

Датафрейми (1)

- **Датафрейм** (dataframe) — це фактично список, але з певними обмеженнями
- У датафреймі всі елементи мають різні назви та є векторами однакової довжини
- Тобто фактично датафрейм є таблицею даних різних типів, де стовпці відповідають **змінним** (variables), а рядки — **спостереженням** (observations)
- Створити датафрейм можна, явно вказавши дані, які в ньому містяться

```
df <- data.frame("height" = c(170, 172, 168, 182), "weight" = c(71, 85, 70, 92))
df
```

```
##   height weight
## 1    170     71
## 2    172     85
## 3    168     70
## 4    182     92
```

Датафрейми (1)

- **Датафрейм** (dataframe) — це фактично список, але з певними обмеженнями
- У датафреймі всі елементи мають різні назви та є векторами однакової довжини
- Тобто фактично датафрейм є таблицею даних різних типів, де стовпці відповідають **змінним** (variables), а рядки — **спостереженням** (observations)
- Створити датафрейм можна, явно вказавши дані, які в ньому містяться

```
df <- data.frame("height" = c(170, 172, 168, 182), "weight" = c(71, 85, 70, 92))
df
```

```
##   height weight
## 1    170     71
## 2    172     85
## 3    168     70
## 4    182     92
```

Датафрейми (1)

- **Датафрейм** (dataframe) — це фактично список, але з певними обмеженнями
- У датафреймі всі елементи мають різні назви та є векторами однакової довжини
- Тобто фактично датафрейм є таблицею даних різних типів, де стовпці відповідають **змінним** (variables), а рядки — **спостереженням** (observations)
- Створити датафрейм можна, явно вказавши дані, які в ньому містяться

```
df <- data.frame("height" = c(170, 172, 168, 182), "weight" = c(71, 85, 70, 92))
df
```

```
##   height weight
## 1    170     71
## 2    172     85
## 3    168     70
## 4    182     92
```

Датафрейми (1)

- **Датафрейм** (dataframe) — це фактично список, але з певними обмеженнями
- У датафреймі всі елементи мають різні назви та є векторами однакової довжини
- Тобто фактично датафрейм є таблицею даних різних типів, де стовпці відповідають **змінним** (variables), а рядки — **спостереженням** (observations)
- Створити датафрейм можна, явно вказавши дані, які в ньому містяться

```
df <- data.frame("height" = c(170, 172, 168, 182), "weight" = c(71, 85, 70, 92))
df
```

```
##   height weight
## 1    170     71
## 2    172     85
## 3    168     70
## 4    182     92
```

Датафрейми (2)

- Оскільки це список, окремі стовпці можна дістати за назвою

```
mean(df$weight)
```

```
## [1] 79.5
```

```
median(df$height)
```

```
## [1] 171
```

- Індексацію також можна робити за індексами

```
df[3, 2]  # третій рядок, другий стовпець
```

```
## [1] 70
```

```
df[1, ]  # весь перший рядок
```

```
##   height weight
## 1    170     71
```

```
df[, 2]  # весь другий стовпець
```

```
## [1] 71 85 70 92
```

```
df[2:4, 1:2]  # рядки 2--4 з обох стовпців
```

```
##   height weight
## 2    172     85
## 3    168     70
## 4    182     92
```

- Така індексація не дуже зручна і не дуже прозора

Датафрейми (2)

- Оскільки це список, окремі стовпці можна дістати за назвою

```
mean(df$weight)
```

```
## [1] 79.5
```

```
median(df$height)
```

```
## [1] 171
```

- Індексацію також можна робити за індексами

```
df[3, 2]  # третій рядок, другий стовпець
```

```
## [1] 70
```

```
df[1, ]  # весь перший рядок
```

```
##   height weight
## 1    170     71
```

```
df[, 2]  # весь другий стовпець
```

```
## [1] 71 85 70 92
```

```
df[2:4, 1:2]  # рядки 2--4 з обох стовпців
```

```
##   height weight
## 2    172     85
## 3    168     70
## 4    182     92
```

- Така індексація не дуже зручна і не дуже прозора

Датафрейми (2)

- Оскільки це список, окремі стовпці можна дістати за назвою

```
mean(df$weight)
```

```
## [1] 79.5
```

```
median(df$height)
```

```
## [1] 171
```

- Індексацію також можна робити за індексами

```
df[3, 2]  # третій рядок, другий стовпець
```

```
## [1] 70
```

```
df[1, ]  # весь перший рядок
```

```
##   height weight
## 1    170     71
```

```
df[, 2]  # весь другий стовпець
```

```
## [1] 71 85 70 92
```

```
df[2:4, 1:2]  # рядки 2--4 з обох стовпців
```

```
##   height weight
## 2    172     85
## 3    168     70
## 4    182     92
```

- Така індексація не дуже зручна і не дуже прозора

Деякі операції з датафреймами (1)

- Розмірність датафрейма

```
dim(df)
```

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

```
nrow(df)
```

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

```
ncol(df)
```

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

- Структура датафрейма

```
str(df)
```

```
## 'data.frame':    4 obs. of  2 variables:
##  $ height: num  170 172 168 182
##  $ weight: num  71 85 70 92
```

```
names(df)  # аналогічно colnames
```

```
## [1] "height" "weight"
```

Деякі операції з датафреймами (1)

- Розмірність датафрейма

```
dim(df)
```

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

```
nrow(df)
```

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

```
ncol(df)
```

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

- Структура датафрейма

```
str(df)
```

```
## 'data.frame':    4 obs. of  2 variables:
##  $ height: num  170 172 168 182
##  $ weight: num  71 85 70 92
```

```
names(df)  # аналогічно colnames
```

```
## [1] "height" "weight"
```

Деякі операції з датафреймами (2)

- Виведення на екран перших і останніх рядків

```
head(df)
```

```
##   height weight
## 1    170     71
## 2    172     85
## 3    168     70
## 4    182     92
```

```
tail(df)
```

```
##   height weight
## 1    170     71
## 2    172     85
## 3    168     70
## 4    182     92
```

Додавання нових змінних

- Додавання нових змінних з іншого датафрейма

```
df2 <- data.frame(age = c(35, 32, 33, 34))
df_new <- cbind(df, df2)
df_new
```

```
##   height weight age
## 1    170     71  35
## 2    172     85  32
## 3    168     70  33
## 4    182     92  34
```

- Додавання нового стовпця

```
df_new$sex <- c(1, 1, 0, 0)
df_new
```

```
##   height weight age sex
## 1    170     71  35   1
## 2    172     85  32   1
## 3    168     70  33   0
## 4    182     92  34   0
```

- Нас ці та інші маніпуляції мало цікавитимуть, оскільки ми будемо працювати з сучасною версію датафреймів — так званими **тиблами** (tibbles) з пакету `tidyverse`

Додавання нових змінних

- Додавання нових змінних з іншого датафрейма

```
df2 <- data.frame(age = c(35, 32, 33, 34))
df_new <- cbind(df, df2)
df_new
```

```
##   height weight age
## 1    170     71  35
## 2    172     85  32
## 3    168     70  33
## 4    182     92  34
```

- Додавання нового стовпця

```
df_new$sex <- c(1, 1, 0, 0)
df_new
```

```
##   height weight age sex
## 1    170     71  35   1
## 2    172     85  32   1
## 3    168     70  33   0
## 4    182     92  34   0
```

- Нас ці та інші маніпуляції мало цікавитимуть, оскільки ми будемо працювати з сучасною версію датафреймів — так званими **тиблами** (tibbles) з пакету `tidyverse`

Додавання нових змінних

- Додавання нових змінних з іншого датафрейма

```
df2 <- data.frame(age = c(35, 32, 33, 34))
df_new <- cbind(df, df2)
df_new
```

```
##   height weight age
## 1    170     71  35
## 2    172     85  32
## 3    168     70  33
## 4    182     92  34
```

- Додавання нового стовпця

```
df_new$sex <- c(1, 1, 0, 0)
df_new
```

```
##   height weight age sex
## 1    170     71  35   1
## 2    172     85  32   1
## 3    168     70  33   0
## 4    182     92  34   0
```

- Нас ці та інші маніпуляції мало цікавитимуть, оскільки ми будемо працювати з сучасною версію датафреймів — так званими **тиблами** (tibbles) з пакету `tidyverse`

План лекції

Що таке tidy data («охайні дані»)

- Автор цього поняття — Гедлі Вікем (Hadley Wickham), який і розробив основні пакети з `tidyverse`
- Повну відповідь на це питання можна знайти в цій статті (також викладена на диск)
- Якщо коротко, то дані можна подати в різний спосіб
- Охайні дані є прямокутними, і до того ж:

Що таке tidy data («охайні дані»)

- Автор цього поняття — Гедлі Вікем (Hadley Wickham), який і розробив основні пакети з `tidyverse`
- Повну відповідь на це питання можна знайти в цій статті (також викладена на диск)
- Якщо коротко, то дані можна подати в різний спосіб
- Охайні дані є прямокутними, і до того ж:
  - Кожній змінній (variable) відповідає окремий стовпець
  - Кожному спостереженню (observation) відповідає окремий рядок
  - Кожне окреме значення (value) зберігається в окремій клітинці

Що таке tidy data («охайні дані»)

- Автор цього поняття — Гедлі Вікем (Hadley Wickham), який і розробив основні пакети з `tidyverse`
- Повну відповідь на це питання можна знайти в цій статті (також викладена на диск)
- Якщо коротко, то дані можна подати в різний спосіб
- Охайні дані є прямокутними, і до того ж:
  - Кожній змінній (variable) відповідає окремий стовпець
  - Кожному спостереженню (observation) відповідає окремий рядок
  - Кожне окреме значення (value) зберігається в окремій комірці

Що таке tidy data («охайні дані»)

- Автор цього поняття — Гедлі Вікем (Hadley Wickham), який і розробив основні пакети з `tidyverse`
- Повну відповідь на це питання можна знайти в цій статті (також викладена на диск)
- Якщо коротко, то дані можна подати в різний спосіб
- Охайні дані є прямокутними, і до того ж:
  - Кожній **змінній** (variable) відповідає окремий стовпець
  - Кожному **спостереженню** (observation) відповідає окремий рядок
  - Кожне окреме **значення** (value) зберігається в окремій комірці

Що таке tidy data («охайні дані»)

- Автор цього поняття — Гедлі Вікем (Hadley Wickham), який і розробив основні пакети з `tidyverse`
- Повну відповідь на це питання можна знайти в цій статті (також викладена на диск)
- Якщо коротко, то дані можна подати в різний спосіб
- Охайні дані є прямокутними, і до того ж:
  - Кожній **змінній** (variable) відповідає окремий стовпець
  - Кожному **спостереженню** (observation) відповідає окремий рядок
  - Кожне окреме **значення** (value) зберігається в окремій комірці

Що таке tidy data («охайні дані»)

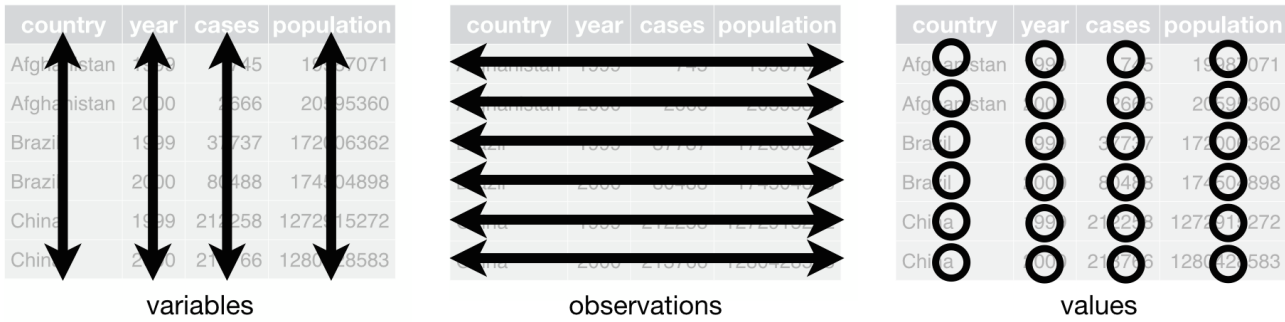
- Автор цього поняття — Гедлі Вікем (Hadley Wickham), який і розробив основні пакети з `tidyverse`
- Повну відповідь на це питання можна знайти в цій статті (також викладена на диск)
- Якщо коротко, то дані можна подати в різний спосіб
- Охайні дані є прямокутними, і до того ж:
  - Кожній **змінній** (variable) відповідає окремий стовпець
  - Кожному **спостереженню** (observation) відповідає окремий рядок
  - Кожне окреме **значення** (value) зберігається в окремій комірці

Що таке tidy data («охайні дані»)

- Автор цього поняття — Гедлі Вікем (Hadley Wickham), який і розробив основні пакети з `tidyverse`
- Повну відповідь на це питання можна знайти в цій статті (також викладена на диск)
- Якщо коротко, то дані можна подати в різний спосіб
- Охайні дані є прямокутними, і до того ж:
  - Кожній **змінній** (variable) відповідає окремий стовпець
  - Кожному **спостереженню** (observation) відповідає окремий рядок
  - Кожне окреме **значення** (value) зберігається в окремій комірці

Що таке tidy data («охайні дані»)

- Автор цього поняття — Гедлі Вікем (Hadley Wickham), який і розробив основні пакети з `tidyverse`
- Повну відповідь на це питання можна знайти в цій статті (також викладена на диск)
- Якщо коротко, то дані можна подати в різний спосіб
- Охайні дані є прямокутними, і до того ж:
  - Кожній **змінній** (variable) відповідає окремий стовпець
  - Кожному **спостереженню** (observation) відповідає окремий рядок
  - Кожне окреме **значення** (value) зберігається в окремій комірці



Приклади організації даних (1)

- Розгляньмо декілька прикладів організації даних із пакету `tidyr`

```
table1
```

```
## # A tibble: 6 x 4
##   country      year  cases population
##   <chr>       <int>  <int>      <int>
## 1 Afghanistan  1999    745   19987071
## 2 Afghanistan  2000   2666   20595360
## 3 Brazil       1999  37737  172006362
## 4 Brazil       2000  80488  174504898
## 5 China        1999 212258 1272915272
## 6 China        2000 213766 1280428583
```

- Цей набір даних є охайним
- Кожній змінній (країна, рік, кількість випадків, населення) відповідає свій стовпець
- Кожному спостереженню (унікальна пара (країна, рік)) — окремий рядок

Приклади організації даних (1)

- Розгляньмо декілька прикладів організації даних із пакету `tidyr`

```
table1
```

```
## # A tibble: 6 x 4
##   country      year  cases population
##   <chr>       <int>  <int>      <int>
## 1 Afghanistan  1999    745   19987071
## 2 Afghanistan  2000   2666   20595360
## 3 Brazil       1999  37737  172006362
## 4 Brazil       2000  80488  174504898
## 5 China        1999 212258 1272915272
## 6 China        2000 213766 1280428583
```

- Цей набір даних є охайним
- Кожній змінній (країна, рік, кількість випадків, населення) відповідає свій стовпець
- Кожному спостереженню (унікальна пара (країна, рік)) — окремий рядок

Приклади організації даних (1)

- Розгляньмо декілька прикладів організації даних із пакету `tidyr`

```
table1
```

```
## # A tibble: 6 x 4
##   country      year  cases population
##   <chr>       <int>  <int>      <int>
## 1 Afghanistan  1999    745   19987071
## 2 Afghanistan  2000   2666   20595360
## 3 Brazil       1999  37737  172006362
## 4 Brazil       2000  80488  174504898
## 5 China        1999 212258 1272915272
## 6 China        2000 213766 1280428583
```

- Цей набір даних є охайним
- Кожній змінній (країна, рік, кількість випадків, населення) відповідає свій стовпець
- Кожному спостереженню (унікальна пара (країна, рік)) — окремий рядок

Приклади організації даних (1)

- Розгляньмо декілька прикладів організації даних із пакету `tidyr`

```
table1
```

```
## # A tibble: 6 x 4
##   country      year  cases population
##   <chr>       <int>  <int>      <int>
## 1 Afghanistan  1999    745   19987071
## 2 Afghanistan  2000   2666   20595360
## 3 Brazil       1999  37737  172006362
## 4 Brazil       2000  80488  174504898
## 5 China        1999 212258 1272915272
## 6 China        2000 213766 1280428583
```

- Цей набір даних є охайним
- Кожній змінній (країна, рік, кількість випадків, населення) відповідає свій стовпець
- Кожному спостереженню (унікальна пара (країна, рік)) — окремий рядок

Приклади організації даних (2)

- Інший варіант:

```
table2
```

```
## # A tibble: 12 x 4
##    country      year type            count
##    <chr>       <int> <chr>           <int>
##  1 Afghanistan  1999 cases             745
##  2 Afghanistan  1999 population   19987071
##  3 Afghanistan  2000 cases            2666
##  4 Afghanistan  2000 population   20595360
##  5 Brazil       1999 cases           37737
##  6 Brazil       1999 population  172006362
##  7 Brazil       2000 cases           80488
##  8 Brazil       2000 population  174504898
##  9 China        1999 cases          212258
## 10 China        1999 population 1272915272
## 11 China        2000 cases          213766
## 12 China        2000 population 1280428583
```

- Тут стовпець `type` не є самостійною змінною, а містить назви двох змінних `cases`, `population`
- Значення цих змінних розміщено в окремому стовпці `count`

Приклади організації даних (2)

- Інший варіант:

```
table2
```

```
## # A tibble: 12 x 4
##    country      year type            count
##    <chr>       <int> <chr>           <int>
##  1 Afghanistan  1999 cases             745
##  2 Afghanistan  1999 population   19987071
##  3 Afghanistan  2000 cases            2666
##  4 Afghanistan  2000 population   20595360
##  5 Brazil       1999 cases           37737
##  6 Brazil       1999 population  172006362
##  7 Brazil       2000 cases           80488
##  8 Brazil       2000 population  174504898
##  9 China        1999 cases          212258
## 10 China        1999 population 1272915272
## 11 China        2000 cases          213766
## 12 China        2000 population 1280428583
```

- Тут стовпець `type` не є самостійною змінною, а містить назви двох змінних `cases`, `population`
- Значення цих змінних розміщено в окремому стовпці `count`

Приклади організації даних (2)

- Інший варіант:

```
table2
```

```
## # A tibble: 12 x 4
##    country      year type            count
##    <chr>       <int> <chr>           <int>
##  1 Afghanistan  1999 cases             745
##  2 Afghanistan  1999 population   19987071
##  3 Afghanistan  2000 cases            2666
##  4 Afghanistan  2000 population   20595360
##  5 Brazil       1999 cases           37737
##  6 Brazil       1999 population  172006362
##  7 Brazil       2000 cases           80488
##  8 Brazil       2000 population  174504898
##  9 China        1999 cases          212258
## 10 China        1999 population 1272915272
## 11 China        2000 cases          213766
## 12 China        2000 population 1280428583
```

- Тут стовпець `type` не є самостійною змінною, а містить назви двох змінних `cases`, `population`
- Значення цих змінних розміщено в окремому стовпці `count`

Приклади організації даних (3)

- Інший варіант:

```
table3
```

```
## # A tibble: 6 x 3
##   country      year rate
## * <chr>       <int> <chr>
## 1 Afghanistan  1999 745/19987071
## 2 Afghanistan  2000 2666/20595360
## 3 Brazil       1999 37737/172006362
## 4 Brazil       2000 80488/174504898
## 5 China        1999 212258/1272915272
## 6 China        2000 213766/1280428583
```

- Цей варіант не є охайним, тому що два показники — число випадків і загальна кількість населення — стиснуто в одному стовпці

Приклади організації даних (3)

- Інший варіант:

```
table3
```

```
## # A tibble: 6 x 3
##   country      year rate
## * <chr>       <int> <chr>
## 1 Afghanistan  1999 745/19987071
## 2 Afghanistan  2000 2666/20595360
## 3 Brazil       1999 37737/172006362
## 4 Brazil       2000 80488/174504898
## 5 China        1999 212258/1272915272
## 6 China        2000 213766/1280428583
```

- Цей варіант не є охайним, тому що два показники — число випадків і загальна кількість населення — стиснуто в одному стовпці

Приклади організації даних (4)

- Інший варіант:

```
table4a
```

```
## # A tibble: 3 x 3
##   country     `1999` `2000`
## * <chr>        <int>  <int>
## 1 Afghanistan    745   2666
## 2 Brazil       37737  80488
## 3 China       212258 213766
```

```
table4b
```

```
## # A tibble: 3 x 3
##   country         `1999`     `2000`
## * <chr>            <int>      <int>
## 1 Afghanistan   19987071   20595360
## 2 Brazil       172006362  174504898
## 3 China       1272915272 1280428583
```

- Цей варіант не є охайним, бо для подання одного набору даних використано два датафрейми
- У кожному датафреймі стовпці відповідають окремим рокам
- Хоча змінною повинно бути не, напр., «Населення в 1999 р.», а просто «Населення»
- Такого роду подання даних можуть бути корисні для публікацій в різних звітах, але зовсім некорисні для роботи з ними та аналізу

Приклади організації даних (4)

- Інший варіант:

```
table4a
```

```
## # A tibble: 3 x 3
##   country     `1999` `2000`
## * <chr>        <int>  <int>
## 1 Afghanistan    745   2666
## 2 Brazil       37737  80488
## 3 China       212258 213766
```

```
table4b
```

```
## # A tibble: 3 x 3
##   country         `1999`     `2000`
## * <chr>            <int>      <int>
## 1 Afghanistan   19987071   20595360
## 2 Brazil       172006362  174504898
## 3 China       1272915272 1280428583
```

- Цей варіант не є охайним, бо для подання одного набору даних використано два датафрейми
- У кожному датафреймі стовпці відповідають окремим рокам
- Хоча змінною повинно бути не, напр., «Населення в 1999 р.», а просто «Населення»
- Такого роду подання даних можуть бути корисні для публікацій в різних звітах, але зовсім некорисні для роботи з ними та аналізу

Приклади організації даних (4)

- Інший варіант:

```
table4a
```

```
## # A tibble: 3 x 3
##   country     `1999` `2000`
## * <chr>        <int>  <int>
## 1 Afghanistan    745   2666
## 2 Brazil       37737  80488
## 3 China       212258 213766
```

```
table4b
```

```
## # A tibble: 3 x 3
##   country         `1999`     `2000`
## * <chr>            <int>      <int>
## 1 Afghanistan   19987071   20595360
## 2 Brazil       172006362  174504898
## 3 China       1272915272 1280428583
```

- Цей варіант не є охайним, бо для подання одного набору даних використано два датафрейми
- У кожному датафреймі стовпці відповідають окремим рокам
- Хоча змінною повинно бути не, напр., «Населення в 1999 р.», а просто «Населення»
- Такого роду подання даних можуть бути корисні для публікацій в різних звітах, але зовсім некорисні для роботи з ними та аналізу

Приклади організації даних (4)

- Інший варіант:

```
table4a
```

```
## # A tibble: 3 x 3
##   country     `1999` `2000`
## * <chr>        <int>  <int>
## 1 Afghanistan    745   2666
## 2 Brazil       37737  80488
## 3 China       212258 213766
```

```
table4b
```

```
## # A tibble: 3 x 3
##   country         `1999`     `2000`
## * <chr>            <int>      <int>
## 1 Afghanistan   19987071   20595360
## 2 Brazil       172006362  174504898
## 3 China       1272915272 1280428583
```

- Цей варіант не є охайним, бо для подання одного набору даних використано два датафрейми
- У кожному датафреймі стовпці відповідають окремим рокам
- Хоча змінною повинно бути не, напр., «Населення в 1999 р.», а просто «Населення»
- Такого роду подання даних можуть бути корисні для публікацій в різних звітах, але зовсім некорисні для роботи з ними та аналізу

Приклади організації даних (4)

- Інший варіант:

```
table4a
```

```
## # A tibble: 3 x 3
##   country     `1999` `2000`
## * <chr>        <int>  <int>
## 1 Afghanistan    745   2666
## 2 Brazil       37737  80488
## 3 China       212258 213766
```

```
table4b
```

```
## # A tibble: 3 x 3
##   country         `1999`     `2000`
## * <chr>            <int>      <int>
## 1 Afghanistan   19987071   20595360
## 2 Brazil       172006362  174504898
## 3 China       1272915272 1280428583
```

- Цей варіант не є охайним, бо для подання одного набору даних використано два датафрейми
- У кожному датафреймі стовпці відповідають окремим рокам
- Хоча змінною повинно бути не, напр., «Населення в 1999 р.», а просто «Населення»
- Такого роду подання даних можуть бути корисні для публікацій в різних звітах, але зовсім некорисні для роботи з ними та аналізу

Робота з охайними даними

- Нас у цьому курсі не так сильно цікавить, як саме приводити дані до охайного вигляду
- Зрештою, це не так складно робити і за потреби можна освоїти самостійно
- Цікаву інформацію можна дістати з розділу *R for Data Science* та інших джерел
- Нас більше цікавить, як можна працювати з охайними даними за допомогою засобів пакету `tidyverse`
- Додаткову інформацію про можливості цього пакету можна дізнатися на офіційному сайті

Робота з охайними даними

- Нас у цьому курсі не так сильно цікавить, як саме приводити дані до охайного вигляду
- Зрештою, це не так складно робити і за потреби можна освоїти самостійно
- Цікаву інформацію можна дістати з розділу *R for Data Science* та інших джерел
- Нас більше цікавить, як можна працювати з охайними даними за допомогою засобів пакету `tidyverse`
- Додаткову інформацію про можливості цього пакету можна дізнатися на офіційному сайті

Робота з охайними даними

- Нас у цьому курсі не так сильно цікавить, як саме приводити дані до охайного вигляду
- Зрештою, це не так складно робити і за потреби можна освоїти самостійно
- Цікаву інформацію можна дістати з розділу *R for Data Science* та інших джерел
- Нас більше цікавить, як можна працювати з охайними даними за допомогою засобів пакету `tidyverse`
- Додаткову інформацію про можливості цього пакету можна дізнатися на офіційному сайті

Робота з охайними даними

- Нас у цьому курсі не так сильно цікавить, як саме приводити дані до охайного вигляду
- Зрештою, це не так складно робити і за потреби можна освоїти самостійно
- Цікаву інформацію можна дістати з розділу *R for Data Science* та інших джерел
- Нас більше цікавить, як можна працювати з охайними даними за допомогою засобів пакету `tidyverse`
- Додаткову інформацію про можливості цього пакету можна дізнатися на офіційному сайті

Робота з охайними даними

- Нас у цьому курсі не так сильно цікавить, як саме приводити дані до охайного вигляду
- Зрештою, це не так складно робити і за потреби можна освоїти самостійно
- Цікаву інформацію можна дістати з розділу *R for Data Science* та інших джерел
- Нас більше цікавить, як можна працювати з охайними даними за допомогою засобів пакету `tidyverse`
- Додаткову інформацію про можливості цього пакету можна дізнатися на офіційному сайті

Приклад

- Як конкретний приклад розгляньмо дані про пасажирів Титаніку, описаний тут
- Після скачування відповідного файлу формату CSV (comma separated values) та розміщення у відповідному каталозі, ми його зчитуємо

```
passengers <- read_csv("data/titanic.csv")
```

```
## Rows: 891 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (5): Name, Sex, Ticket, Cabin, Embarked
## dbl (7): PassengerId, Survived, Pclass, Age, SibSp, Parch, Fare
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
passengers
```

```
## # A tibble: 891 x 12
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1           1        0      3 Braun~ male     22     1     0 A/5 2~  7.25 <NA>
##  2           2        1      1 Cumin~ fema~    38     1     0 PC 17~ 71.3  C85
##  3           3        1      3 Heikk~ fema~    26     0     0 STON/~  7.92 <NA>
##  4           4        1      1 Futre~ fema~    35     1     0 113803 53.1  C123
##  5           5        0      3 Allen~ male     35     0     0 373450  8.05 <NA>
##  6           6        0      3 Moran~ male     NA     0     0 330877  8.46 <NA>
##  7           7        0      1 McCar~ male     54     0     0 17463  51.9  E46
##  8           8        0      3 Palss~ male      2     3     1 349909 21.1  <NA>
##  9           9        1      3 Johns~ fema~    27     0     2 347742 11.1  <NA>
## 10          10        1      2 Nasse~ fema~    14     1     0 237736 30.1  <NA>
## # ... with 881 more rows, and 1 more variable: Embarked <chr>
```

- В описах стовпців можемо зустріти `int` (цілі числа), `dbl` (дійсні числа), `chr` (рядки), `lgl` (логічні), `fctr` (фактори), `dttm` (дата і час) та `date` (дата)

Приклад

- Як конкретний приклад розгляньмо дані про пасажирів Титаніку, описаний тут
- Після скачування відповідного файлу формату CSV (comma separated values) та розміщення у відповідному каталозі, ми його зчитуємо

```
passengers <- read_csv("data/titanic.csv")
```

```
## Rows: 891 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (5): Name, Sex, Ticket, Cabin, Embarked
## dbl (7): PassengerId, Survived, Pclass, Age, SibSp, Parch, Fare
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
passengers
```

```
## # A tibble: 891 x 12
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1           1        0      3 Braun~ male     22     1     0 A/5 2~  7.25 <NA>
##  2           2        1      1 Cumin~ fema~    38     1     0 PC 17~ 71.3  C85
##  3           3        1      3 Heikk~ fema~    26     0     0 STON/~  7.92 <NA>
##  4           4        1      1 Futre~ fema~    35     1     0 113803 53.1  C123
##  5           5        0      3 Allen~ male     35     0     0 373450  8.05 <NA>
##  6           6        0      3 Moran~ male     NA     0     0 330877  8.46 <NA>
##  7           7        0      1 McCar~ male     54     0     0 17463  51.9  E46
##  8           8        0      3 Palss~ male      2     3     1 349909 21.1  <NA>
##  9           9        1      3 Johns~ fema~    27     0     2 347742 11.1  <NA>
## 10          10        1      2 Nasse~ fema~    14     1     0 237736 30.1  <NA>
## # ... with 881 more rows, and 1 more variable: Embarked <chr>
```

- В описах стовпців можемо зустріти `int` (цілі числа), `dbl` (дійсні числа), `chr` (рядки), `lgl` (логічні), `fctr` (фактори), `dttm` (дата і час) та `date` (дата)

Приклад

- Як конкретний приклад розгляньмо дані про пасажирів Титаніку, описаний тут
- Після скачування відповідного файлу формату CSV (comma separated values) та розміщення у відповідному каталозі, ми його зчитуємо

```
passengers <- read_csv("data/titanic.csv")
```

```
## Rows: 891 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (5): Name, Sex, Ticket, Cabin, Embarked
## dbl (7): PassengerId, Survived, Pclass, Age, SibSp, Parch, Fare
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
passengers
```

```
## # A tibble: 891 x 12
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1           1        0      3 Braun~ male     22     1     0 A/5 2~  7.25 <NA>
##  2           2        1      1 Cumin~ fema~    38     1     0 PC 17~ 71.3  C85
##  3           3        1      3 Heikk~ fema~    26     0     0 STON/~  7.92 <NA>
##  4           4        1      1 Futre~ fema~    35     1     0 113803 53.1  C123
##  5           5        0      3 Allen~ male     35     0     0 373450  8.05 <NA>
##  6           6        0      3 Moran~ male     NA     0     0 330877  8.46 <NA>
##  7           7        0      1 McCar~ male     54     0     0 17463  51.9  E46
##  8           8        0      3 Palss~ male      2     3     1 349909 21.1  <NA>
##  9           9        1      3 Johns~ fema~    27     0     2 347742 11.1  <NA>
## 10          10        1      2 Nasse~ fema~    14     1     0 237736 30.1  <NA>
## # ... with 881 more rows, and 1 more variable: Embarked <chr>
```

- В описах стовпців можемо зустріти `int` (цілі числа), `dbl` (дійсні числа), `chr` (рядки), `lgl` (логічні), `fctr` (фактори), `dttm` (дата і час) та `date` (дата)

Відбір рядків за певними критеріями

- Якщо ми хочемо виокремити певну підмножину спостережень, то потрібно використати функцію `filter()`
- Аргументами повинні стати критерії, за якими потрібно здійснювати відбір
- Критерії можна конкатенувати за допомогою логічних операцій, а можна перелічити через кому

```
passengers %>% filter(Sex == "female")
```

```
## # A tibble: 314 x 12
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1           2        1      1 Cumin~ fema~    38     1     0 PC 17~ 71.3  C85
##  2           3        1      3 Heikk~ fema~    26     0     0 STON/~  7.92 <NA>
##  3           4        1      1 Futre~ fema~    35     1     0 113803 53.1  C123
##  4           9        1      3 Johns~ fema~    27     0     2 347742 11.1  <NA>
##  5          10        1      2 Nasse~ fema~    14     1     0 237736 30.1  <NA>
##  6          11        1      3 Sands~ fema~     4     1     1 PP 95~ 16.7  G6
##  7          12        1      1 Bonne~ fema~    58     0     0 113783 26.6  C103
##  8          15        0      3 Vestr~ fema~    14     0     0 350406  7.85 <NA>
##  9          16        1      2 Hewle~ fema~    55     0     0 248706 16    <NA>
## 10          19        0      3 Vande~ fema~    31     1     0 345763 18    <NA>
## # ... with 304 more rows, and 1 more variable: Embarked <chr>
```

- Ми відібрали всіх жінок
- Можна помітити тенденцію, що серед жінок більшість вижили: це варто додаткового аналізу
- Результат відбору можна за потреби записати в окрему змінну, початкові дані при цьому не змінюються

Відбір рядків за певними критеріями

- Якщо ми хочемо виокремити певну підмножину спостережень, то потрібно використати функцію `filter()`
- Аргументами повинні стати критерії, за якими потрібно здійснювати відбір
- Критерії можна конкатенувати за допомогою логічних операцій, а можна перелічити через кому

```
passengers %>% filter(Sex == "female")
```

```
## # A tibble: 314 x 12
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1           2        1      1 Cumin~ fema~    38     1     0 PC 17~ 71.3  C85
##  2           3        1      3 Heikk~ fema~    26     0     0 STON/~  7.92 <NA>
##  3           4        1      1 Futre~ fema~    35     1     0 113803 53.1  C123
##  4           9        1      3 Johns~ fema~    27     0     2 347742 11.1  <NA>
##  5          10        1      2 Nasse~ fema~    14     1     0 237736 30.1  <NA>
##  6          11        1      3 Sands~ fema~     4     1     1 PP 95~ 16.7  G6
##  7          12        1      1 Bonne~ fema~    58     0     0 113783 26.6  C103
##  8          15        0      3 Vestr~ fema~    14     0     0 350406  7.85 <NA>
##  9          16        1      2 Hewle~ fema~    55     0     0 248706 16    <NA>
## 10          19        0      3 Vande~ fema~    31     1     0 345763 18    <NA>
## # ... with 304 more rows, and 1 more variable: Embarked <chr>
```

- Ми відібрали всіх жінок
- Можна помітити тенденцію, що серед жінок більшість вижили: це варто додаткового аналізу
- Результат відбору можна за потреби записати в окрему змінну, початкові дані при цьому не змінюються

Відбір рядків за певними критеріями

- Якщо ми хочемо виокремити певну підмножину спостережень, то потрібно використати функцію `filter()`
- Аргументами повинні стати критерії, за якими потрібно здійснювати відбір
- Критерії можна конкатенувати за допомогою логічних операцій, а можна перелічити через кому

```
passengers %>% filter(Sex == "female")
```

```
## # A tibble: 314 x 12
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1           2        1      1 Cumin~ fema~    38     1     0 PC 17~ 71.3  C85
##  2           3        1      3 Heikk~ fema~    26     0     0 STON/~  7.92 <NA>
##  3           4        1      1 Futre~ fema~    35     1     0 113803 53.1  C123
##  4           9        1      3 Johns~ fema~    27     0     2 347742 11.1  <NA>
##  5          10        1      2 Nasse~ fema~    14     1     0 237736 30.1  <NA>
##  6          11        1      3 Sands~ fema~     4     1     1 PP 95~ 16.7  G6
##  7          12        1      1 Bonne~ fema~    58     0     0 113783 26.6  C103
##  8          15        0      3 Vestr~ fema~    14     0     0 350406  7.85 <NA>
##  9          16        1      2 Hewle~ fema~    55     0     0 248706 16    <NA>
## 10          19        0      3 Vande~ fema~    31     1     0 345763 18    <NA>
## # ... with 304 more rows, and 1 more variable: Embarked <chr>
```

- Ми відібрали всіх жінок
- Можна помітити тенденцію, що серед жінок більшість вижили: це варто додаткового аналізу
- Результат відбору можна за потреби записати в окрему змінну, початкові дані при цьому не змінюються

Відбір рядків за певними критеріями

- Якщо ми хочемо виокремити певну підмножину спостережень, то потрібно використати функцію `filter()`
- Аргументами повинні стати критерії, за якими потрібно здійснювати відбір
- Критерії можна конкатенувати за допомогою логічних операцій, а можна перелічити через кому

```
passengers %>% filter(Sex == "female")
```

```
## # A tibble: 314 x 12
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1           2        1      1 Cumin~ fema~    38     1     0 PC 17~ 71.3  C85
##  2           3        1      3 Heikk~ fema~    26     0     0 STON/~  7.92 <NA>
##  3           4        1      1 Futre~ fema~    35     1     0 113803 53.1  C123
##  4           9        1      3 Johns~ fema~    27     0     2 347742 11.1  <NA>
##  5          10        1      2 Nasse~ fema~    14     1     0 237736 30.1  <NA>
##  6          11        1      3 Sands~ fema~     4     1     1 PP 95~ 16.7  G6
##  7          12        1      1 Bonne~ fema~    58     0     0 113783 26.6  C103
##  8          15        0      3 Vestr~ fema~    14     0     0 350406  7.85 <NA>
##  9          16        1      2 Hewle~ fema~    55     0     0 248706 16    <NA>
## 10          19        0      3 Vande~ fema~    31     1     0 345763 18    <NA>
## # ... with 304 more rows, and 1 more variable: Embarked <chr>
```

- Ми відібрали всіх жінок
- Можна помітити тенденцію, що серед жінок більшість вижили: це варто додаткового аналізу
- Результат відбору можна за потреби записати в окрему змінну, початкові дані при цьому не змінюються

Відбір рядків за певними критеріями

- Якщо ми хочемо виокремити певну підмножину спостережень, то потрібно використати функцію `filter()`
- Аргументами повинні стати критерії, за якими потрібно здійснювати відбір
- Критерії можна конкатенувати за допомогою логічних операцій, а можна перелічити через кому

```
passengers %>% filter(Sex == "female")
```

```
## # A tibble: 314 x 12
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1           2        1      1 Cumin~ fema~    38     1     0 PC 17~ 71.3  C85
##  2           3        1      3 Heikk~ fema~    26     0     0 STON/~  7.92 <NA>
##  3           4        1      1 Futre~ fema~    35     1     0 113803 53.1  C123
##  4           9        1      3 Johns~ fema~    27     0     2 347742 11.1  <NA>
##  5          10        1      2 Nasse~ fema~    14     1     0 237736 30.1  <NA>
##  6          11        1      3 Sands~ fema~     4     1     1 PP 95~ 16.7  G6
##  7          12        1      1 Bonne~ fema~    58     0     0 113783 26.6  C103
##  8          15        0      3 Vestr~ fema~    14     0     0 350406  7.85 <NA>
##  9          16        1      2 Hewle~ fema~    55     0     0 248706 16    <NA>
## 10          19        0      3 Vande~ fema~    31     1     0 345763 18    <NA>
## # ... with 304 more rows, and 1 more variable: Embarked <chr>
```

- Ми відібрали всіх жінок
- Можна помітити тенденцію, що серед жінок більшість вижили: це варто додаткового аналізу
- Результат відбору можна за потреби записати в окрему змінну, початкові дані при цьому не змінюються

Відбір рядків за певними критеріями

- Якщо ми хочемо виокремити певну підмножину спостережень, то потрібно використати функцію `filter()`
- Аргументами повинні стати критерії, за якими потрібно здійснювати відбір
- Критерії можна конкатенувати за допомогою логічних операцій, а можна перелічити через кому

```
passengers %>% filter(Sex == "female")
```

```
## # A tibble: 314 x 12
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1           2        1      1 Cumin~ fema~    38     1     0 PC 17~ 71.3  C85
##  2           3        1      3 Heikk~ fema~    26     0     0 STON/~  7.92 <NA>
##  3           4        1      1 Futre~ fema~    35     1     0 113803 53.1  C123
##  4           9        1      3 Johns~ fema~    27     0     2 347742 11.1  <NA>
##  5          10        1      2 Nasse~ fema~    14     1     0 237736 30.1  <NA>
##  6          11        1      3 Sands~ fema~     4     1     1 PP 95~ 16.7  G6
##  7          12        1      1 Bonne~ fema~    58     0     0 113783 26.6  C103
##  8          15        0      3 Vestr~ fema~    14     0     0 350406  7.85 <NA>
##  9          16        1      2 Hewle~ fema~    55     0     0 248706 16    <NA>
## 10          19        0      3 Vande~ fema~    31     1     0 345763 18    <NA>
## # ... with 304 more rows, and 1 more variable: Embarked <chr>
```

- Ми відібрали всіх жінок
- Можна помітити тенденцію, що серед жінок більшість вижили: це варто додаткового аналізу
- Результат відбору можна за потреби записати в окрему змінну, початкові дані при цьому не змінюються

Сортування рядків за певними критеріями (1)

- Якщо ми хочемо сортувати дані за певними критеріями, то потрібно використати функцію `arrange()`
- Аргументами повинні стати назви стовпців, за якими потрібно сортувати
- Можна вказувати декілька стовпців: тоді спочатку відбуватиметься сортування за першим, далі — за другим тощо

```
passengers %>% arrange(Fare, Age)
```

```
## # A tibble: 891 x 12
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1         303        0      3 "John~ male     19     0     0 LINE       0 <NA>
##  2         272        1      3 "Torn~ male     25     0     0 LINE       0 <NA>
##  3         180        0      3 "Leon~ male     36     0     0 LINE       0 <NA>
##  4         823        0      1 "Reuc~ male     38     0     0 19972      0 <NA>
##  5         807        0      1 "Andr~ male     39     0     0 112050     0 A36
##  6         264        0      1 "Harr~ male     40     0     0 112059     0 B94
##  7         598        0      3 "John~ male     49     0     0 LINE       0 <NA>
##  8         278        0      2 "Park~ male     NA     0     0 239853     0 <NA>
##  9         414        0      2 "Cunn~ male     NA     0     0 239853     0 <NA>
## 10         467        0      2 "Camp~ male     NA     0     0 239853     0 <NA>
## # ... with 881 more rows, and 1 more variable: Embarked <chr>
```

- Ми відсортували всіх пасажирів за вартістю квитка, а у випадку однакової вартости — ще й за віком
- Можна помітити, що пасажири з малою вартістю квитка, як правило, не вижили

Сортування рядків за певними критеріями (1)

- Якщо ми хочемо сортувати дані за певними критеріями, то потрібно використати функцію `arrange()`
- Аргументами повинні стати назви стовпців, за якими потрібно сортувати
- Можна вказувати декілька стовпців: тоді спочатку відбуватиметься сортування за першим, далі — за другим тощо

```
passengers %>% arrange(Fare, Age)
```

```
## # A tibble: 891 x 12
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1         303        0      3 "John~ male     19     0     0 LINE       0 <NA>
##  2         272        1      3 "Torn~ male     25     0     0 LINE       0 <NA>
##  3         180        0      3 "Leon~ male     36     0     0 LINE       0 <NA>
##  4         823        0      1 "Reuc~ male     38     0     0 19972      0 <NA>
##  5         807        0      1 "Andr~ male     39     0     0 112050     0 A36
##  6         264        0      1 "Harr~ male     40     0     0 112059     0 B94
##  7         598        0      3 "John~ male     49     0     0 LINE       0 <NA>
##  8         278        0      2 "Park~ male     NA     0     0 239853     0 <NA>
##  9         414        0      2 "Cunn~ male     NA     0     0 239853     0 <NA>
## 10         467        0      2 "Camp~ male     NA     0     0 239853     0 <NA>
## # ... with 881 more rows, and 1 more variable: Embarked <chr>
```

- Ми відсортували всіх пасажирів за вартістю квитка, а у випадку однакової вартости — ще й за віком
- Можна помітити, що пасажири з малою вартістю квитка, як правило, не вижили

Сортування рядків за певними критеріями (1)

- Якщо ми хочемо сортувати дані за певними критеріями, то потрібно використати функцію `arrange()`
- Аргументами повинні стати назви стовпців, за якими потрібно сортувати
- Можна вказувати декілька стовпців: тоді спочатку відбуватиметься сортування за першим, далі — за другим тощо

```
passengers %>% arrange(Fare, Age)
```

```
## # A tibble: 891 x 12
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1         303        0      3 "John~ male     19     0     0 LINE       0 <NA>
##  2         272        1      3 "Torn~ male     25     0     0 LINE       0 <NA>
##  3         180        0      3 "Leon~ male     36     0     0 LINE       0 <NA>
##  4         823        0      1 "Reuc~ male     38     0     0 19972      0 <NA>
##  5         807        0      1 "Andr~ male     39     0     0 112050     0 A36
##  6         264        0      1 "Harr~ male     40     0     0 112059     0 B94
##  7         598        0      3 "John~ male     49     0     0 LINE       0 <NA>
##  8         278        0      2 "Park~ male     NA     0     0 239853     0 <NA>
##  9         414        0      2 "Cunn~ male     NA     0     0 239853     0 <NA>
## 10         467        0      2 "Camp~ male     NA     0     0 239853     0 <NA>
## # ... with 881 more rows, and 1 more variable: Embarked <chr>
```

- Ми відсортували всіх пасажирів за вартістю квитка, а у випадку однакової вартости — ще й за віком
- Можна помітити, що пасажири з малою вартістю квитка, як правило, не вижили

Сортування рядків за певними критеріями (1)

- Якщо ми хочемо сортувати дані за певними критеріями, то потрібно використати функцію `arrange()`
- Аргументами повинні стати назви стовпців, за якими потрібно сортувати
- Можна вказувати декілька стовпців: тоді спочатку відбуватиметься сортування за першим, далі — за другим тощо

```
passengers %>% arrange(Fare, Age)
```

```
## # A tibble: 891 x 12
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1         303        0      3 "John~ male     19     0     0 LINE       0 <NA>
##  2         272        1      3 "Torn~ male     25     0     0 LINE       0 <NA>
##  3         180        0      3 "Leon~ male     36     0     0 LINE       0 <NA>
##  4         823        0      1 "Reuc~ male     38     0     0 19972      0 <NA>
##  5         807        0      1 "Andr~ male     39     0     0 112050     0 A36
##  6         264        0      1 "Harr~ male     40     0     0 112059     0 B94
##  7         598        0      3 "John~ male     49     0     0 LINE       0 <NA>
##  8         278        0      2 "Park~ male     NA     0     0 239853     0 <NA>
##  9         414        0      2 "Cunn~ male     NA     0     0 239853     0 <NA>
## 10         467        0      2 "Camp~ male     NA     0     0 239853     0 <NA>
## # ... with 881 more rows, and 1 more variable: Embarked <chr>
```

- Ми відсортували всіх пасажирів за вартістю квитка, а у випадку однакової вартости — ще й за віком
- Можна помітити, що пасажири з малою вартістю квитка, як правило, не вижили

Сортування рядків за певними критеріями (1)

- Якщо ми хочемо сортувати дані за певними критеріями, то потрібно використати функцію `arrange()`
- Аргументами повинні стати назви стовпців, за якими потрібно сортувати
- Можна вказувати декілька стовпців: тоді спочатку відбуватиметься сортування за першим, далі — за другим тощо

```
passengers %>% arrange(Fare, Age)
```

```
## # A tibble: 891 x 12
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1         303        0      3 "John~ male     19     0     0 LINE       0 <NA>
##  2         272        1      3 "Torn~ male     25     0     0 LINE       0 <NA>
##  3         180        0      3 "Leon~ male     36     0     0 LINE       0 <NA>
##  4         823        0      1 "Reuc~ male     38     0     0 19972      0 <NA>
##  5         807        0      1 "Andr~ male     39     0     0 112050     0 A36
##  6         264        0      1 "Harr~ male     40     0     0 112059     0 B94
##  7         598        0      3 "John~ male     49     0     0 LINE       0 <NA>
##  8         278        0      2 "Park~ male     NA     0     0 239853     0 <NA>
##  9         414        0      2 "Cunn~ male     NA     0     0 239853     0 <NA>
## 10         467        0      2 "Camp~ male     NA     0     0 239853     0 <NA>
## # ... with 881 more rows, and 1 more variable: Embarked <chr>
```

- Ми відсортували всіх пасажирів за вартістю квитка, а у випадку однакової вартости — ще й за віком
- Можна помітити, що пасажири з малою вартістю квитка, як правило, не вижили

Сортування рядків за певними критеріями (2)

- За замовчуванням сортування йде за зростанням
- Для сортування в спадному порядку, потрібно додати функцію `desc`

```
passengers %>% arrange(desc(Fare), Age)
```

```
## # A tibble: 891 x 12
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1         259        1      1 "Ward~ fema~    35     0     0 PC 17~  512. <NA>
##  2         738        1      1 "Lesu~ male     35     0     0 PC 17~  512. B101
##  3         680        1      1 "Card~ male     36     0     1 PC 17~  512. B51 ~
##  4          28        0      1 "Fort~ male     19     3     2 19950   263  C23 ~
##  5          89        1      1 "Fort~ fema~    23     3     2 19950   263  C23 ~
##  6         342        1      1 "Fort~ fema~    24     3     2 19950   263  C23 ~
##  7         439        0      1 "Fort~ male     64     1     4 19950   263  C23 ~
##  8         312        1      1 "Ryer~ fema~    18     2     2 PC 17~  262. B57 ~
##  9         743        1      1 "Ryer~ fema~    21     2     2 PC 17~  262. B57 ~
## 10         119        0      1 "Baxt~ male     24     0     1 PC 17~  248. B58 ~
## # ... with 881 more rows, and 1 more variable: Embarked <chr>
```

- Справді на перший погляд уцілілих більше серед заможних пасажирів
- Варто зазначити, що пропущені значення `NA` завжди сортуються останні, незалежно від напрямку сортування!

Сортування рядків за певними критеріями (2)

- За замовчуванням сортування йде за зростанням
- Для сортування в спадному порядку, потрібно додати функцію `desc`

```
passengers %>% arrange(desc(Fare), Age)
```

```
## # A tibble: 891 x 12
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1         259        1      1 "Ward~ fema~    35     0     0 PC 17~  512. <NA>
##  2         738        1      1 "Lesu~ male     35     0     0 PC 17~  512. B101
##  3         680        1      1 "Card~ male     36     0     1 PC 17~  512. B51 ~
##  4          28        0      1 "Fort~ male     19     3     2 19950   263  C23 ~
##  5          89        1      1 "Fort~ fema~    23     3     2 19950   263  C23 ~
##  6         342        1      1 "Fort~ fema~    24     3     2 19950   263  C23 ~
##  7         439        0      1 "Fort~ male     64     1     4 19950   263  C23 ~
##  8         312        1      1 "Ryer~ fema~    18     2     2 PC 17~  262. B57 ~
##  9         743        1      1 "Ryer~ fema~    21     2     2 PC 17~  262. B57 ~
## 10         119        0      1 "Baxt~ male     24     0     1 PC 17~  248. B58 ~
## # ... with 881 more rows, and 1 more variable: Embarked <chr>
```

- Справді на перший погляд уцілілих більше серед заможних пасажирів
- Варто зазначити, що пропущені значення NA завжди сортуються останні, незалежно від напрямку сортування!

Сортування рядків за певними критеріями (2)

- За замовчуванням сортування йде за зростанням
- Для сортування в спадному порядку, потрібно додати функцію `desc`

```
passengers %>% arrange(desc(Fare), Age)
```

```
## # A tibble: 891 x 12
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1         259        1      1 "Ward~ fema~    35     0     0 PC 17~  512. <NA>
##  2         738        1      1 "Lesu~ male     35     0     0 PC 17~  512. B101
##  3         680        1      1 "Card~ male     36     0     1 PC 17~  512. B51 ~
##  4          28        0      1 "Fort~ male     19     3     2 19950   263  C23 ~
##  5          89        1      1 "Fort~ fema~    23     3     2 19950   263  C23 ~
##  6         342        1      1 "Fort~ fema~    24     3     2 19950   263  C23 ~
##  7         439        0      1 "Fort~ male     64     1     4 19950   263  C23 ~
##  8         312        1      1 "Ryer~ fema~    18     2     2 PC 17~  262. B57 ~
##  9         743        1      1 "Ryer~ fema~    21     2     2 PC 17~  262. B57 ~
## 10         119        0      1 "Baxt~ male     24     0     1 PC 17~  248. B58 ~
## # ... with 881 more rows, and 1 more variable: Embarked <chr>
```

- Справді на перший погляд уцілілих більше серед заможних пасажирів
- Варто зазначити, що пропущені значення `NA` завжди сортуються останні, незалежно від напрямку сортування!

Сортування рядків за певними критеріями (2)

- За замовчуванням сортування йде за зростанням
- Для сортування в спадному порядку, потрібно додати функцію `desc`

```
passengers %>% arrange(desc(Fare), Age)
```

```
## # A tibble: 891 x 12
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1         259        1      1 "Ward~ fema~    35     0     0 PC 17~  512. <NA>
##  2         738        1      1 "Lesu~ male     35     0     0 PC 17~  512. B101
##  3         680        1      1 "Card~ male     36     0     1 PC 17~  512. B51 ~
##  4          28        0      1 "Fort~ male     19     3     2 19950   263  C23 ~
##  5          89        1      1 "Fort~ fema~    23     3     2 19950   263  C23 ~
##  6         342        1      1 "Fort~ fema~    24     3     2 19950   263  C23 ~
##  7         439        0      1 "Fort~ male     64     1     4 19950   263  C23 ~
##  8         312        1      1 "Ryer~ fema~    18     2     2 PC 17~  262. B57 ~
##  9         743        1      1 "Ryer~ fema~    21     2     2 PC 17~  262. B57 ~
## 10         119        0      1 "Baxt~ male     24     0     1 PC 17~  248. B58 ~
## # ... with 881 more rows, and 1 more variable: Embarked <chr>
```

- Справді на перший погляд уцілілих більше серед заможних пасажирів
- Варто зазначити, що пропущені значення `NA` завжди сортуються останні, незалежно від напрямку сортування!

Додавання нових змінних

- Дуже часто в аналізі даних (і в машинному навчанні) потрібно створювати нові змінні
- Це часто є етапом попередньої підготовки даних, коли з існуючих стовпців потрібно утворити ті, які мають більший сенс
- Часто це потрібно робити на етапі самого аналізу для утворення нових змінних (напр., логаритмування чи піднесення до квадрату тощо)
- Для утворення нових змінних потрібно використати функцію `mutate()`
- Аргументами повинні стати пари *новий стовпець–функція від старих стовпців*

```
passengers %>% mutate(FamSize = Parch + SibSp)
```

```
## # A tibble: 891 x 13
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1           1        0      3 Braun~ male     22     1     0 A/5 2~  7.25 <NA>
##  2           2        1      1 Cumin~ fema~    38     1     0 PC 17~ 71.3  C85
##  3           3        1      3 Heikk~ fema~    26     0     0 STON/~  7.92 <NA>
##  4           4        1      1 Futre~ fema~    35     1     0 113803 53.1  C123
##  5           5        0      3 Allen~ male     35     0     0 373450  8.05 <NA>
##  6           6        0      3 Moran~ male     NA     0     0 330877  8.46 <NA>
##  7           7        0      1 McCar~ male     54     0     0 17463  51.9  E46
##  8           8        0      3 Palss~ male      2     3     1 349909 21.1  <NA>
##  9           9        1      3 Johns~ fema~    27     0     2 347742 11.1  <NA>
## 10          10        1      2 Nasse~ fema~    14     1     0 237736 30.1  <NA>
## # ... with 881 more rows, and 2 more variables: Embarked <chr>, FamSize <dbl>
```

- Ми додали новий стовпець `FamSize` (кількість членів сім'ї) як суму батьків і дітей (`Parch`) та братів і сестер (`SibSp`), які пливли разом із пасажиром
- До змінних можна застосовувати дуже багато різних функцій
- У відповідних джерелах завжди можна знайти, як реалізувати ту чи ту ідею

Додавання нових змінних

- Дуже часто в аналізі даних (і в машинному навчанні) потрібно створювати нові змінні
- Це часто є етапом попередньої підготовки даних, коли з існуючих стовпців потрібно утворити ті, які мають більший сенс
- Часто це потрібно робити на етапі самого аналізу для утворення нових змінних (напр., логаритмування чи піднесення до квадрату тощо)
- Для утворення нових змінних потрібно використати функцію `mutate()`
- Аргументами повинні стати пари *новий стовпець–функція від старих стовпців*

```
passengers %>% mutate(FamSize = Parch + SibSp)
```

```
## # A tibble: 891 x 13
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1           1        0      3 Braun~ male     22     1     0 A/5 2~  7.25 <NA>
##  2           2        1      1 Cumin~ fema~    38     1     0 PC 17~ 71.3  C85
##  3           3        1      3 Heikk~ fema~    26     0     0 STON/~  7.92 <NA>
##  4           4        1      1 Futre~ fema~    35     1     0 113803 53.1  C123
##  5           5        0      3 Allen~ male     35     0     0 373450  8.05 <NA>
##  6           6        0      3 Moran~ male     NA     0     0 330877  8.46 <NA>
##  7           7        0      1 McCar~ male     54     0     0 17463  51.9  E46
##  8           8        0      3 Palss~ male      2     3     1 349909 21.1  <NA>
##  9           9        1      3 Johns~ fema~    27     0     2 347742 11.1  <NA>
## 10          10        1      2 Nasse~ fema~    14     1     0 237736 30.1  <NA>
## # ... with 881 more rows, and 2 more variables: Embarked <chr>, FamSize <dbl>
```

- Ми додали новий стовпець `FamSize` (кількість членів сім'ї) як суму батьків і дітей (`Parch`) та братів і сестер (`SibSp`), які пливли разом із пасажиром
- До змінних можна застосовувати дуже багато різних функцій
- У відповідних джерелах завжди можна знайти, як реалізувати ту чи ту ідею

Додавання нових змінних

- Дуже часто в аналізі даних (і в машинному навчанні) потрібно створювати нові змінні
- Це часто є етапом попередньої підготовки даних, коли з існуючих стовпців потрібно утворити ті, які мають більший сенс
- Часто це потрібно робити на етапі самого аналізу для утворення нових змінних (напр., логаритмування чи піднесення до квадрату тощо)
- Для утворення нових змінних потрібно використати функцію `mutate()`
- Аргументами повинні стати пари *новий стовпець–функція від старих стовпців*

```
passengers %>% mutate(FamSize = Parch + SibSp)
```

```
## # A tibble: 891 x 13
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1           1        0      3 Braun~ male     22     1     0 A/5 2~  7.25 <NA>
##  2           2        1      1 Cumin~ fema~    38     1     0 PC 17~ 71.3  C85
##  3           3        1      3 Heikk~ fema~    26     0     0 STON/~  7.92 <NA>
##  4           4        1      1 Futre~ fema~    35     1     0 113803 53.1  C123
##  5           5        0      3 Allen~ male     35     0     0 373450  8.05 <NA>
##  6           6        0      3 Moran~ male     NA     0     0 330877  8.46 <NA>
##  7           7        0      1 McCar~ male     54     0     0 17463  51.9  E46
##  8           8        0      3 Palss~ male      2     3     1 349909 21.1  <NA>
##  9           9        1      3 Johns~ fema~    27     0     2 347742 11.1  <NA>
## 10          10        1      2 Nasse~ fema~    14     1     0 237736 30.1  <NA>
## # ... with 881 more rows, and 2 more variables: Embarked <chr>, FamSize <dbl>
```

- Ми додали новий стовпець `FamSize` (кількість членів сім'ї) як суму батьків і дітей (`Parch`) та братів і сестер (`SibSp`), які пливли разом із пасажиром
- До змінних можна застосовувати дуже багато різних функцій
- У відповідних джерелах завжди можна знайти, як реалізувати ту чи ту ідею

Додавання нових змінних

- Дуже часто в аналізі даних (і в машинному навчанні) потрібно створювати нові змінні
- Це часто є етапом попередньої підготовки даних, коли з існуючих стовпців потрібно утворити ті, які мають більший сенс
- Часто це потрібно робити на етапі самого аналізу для утворення нових змінних (напр., логаритмування чи піднесення до квадрату тощо)
- Для утворення нових змінних потрібно використати функцію `mutate()`
- Аргументами повинні стати пари *новий стовпець–функція від старих стовпців*

```
passengers %>% mutate(FamSize = Parch + SibSp)
```

```
## # A tibble: 891 x 13
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1           1        0      3 Braun~ male     22     1     0 A/5 2~  7.25 <NA>
##  2           2        1      1 Cumin~ fema~    38     1     0 PC 17~ 71.3  C85
##  3           3        1      3 Heikk~ fema~    26     0     0 STON/~  7.92 <NA>
##  4           4        1      1 Futre~ fema~    35     1     0 113803 53.1  C123
##  5           5        0      3 Allen~ male     35     0     0 373450  8.05 <NA>
##  6           6        0      3 Moran~ male     NA     0     0 330877  8.46 <NA>
##  7           7        0      1 McCar~ male     54     0     0 17463  51.9  E46
##  8           8        0      3 Palss~ male      2     3     1 349909 21.1  <NA>
##  9           9        1      3 Johns~ fema~    27     0     2 347742 11.1  <NA>
## 10          10        1      2 Nasse~ fema~    14     1     0 237736 30.1  <NA>
## # ... with 881 more rows, and 2 more variables: Embarked <chr>, FamSize <dbl>
```

- Ми додали новий стовпець `FamSize` (кількість членів сім'ї) як суму батьків і дітей (`Parch`) та братів і сестер (`SibSp`), які пливли разом із пасажиром
- До змінних можна застосовувати дуже багато різних функцій
- У відповідних джерелах завжди можна знайти, як реалізувати ту чи ту ідею

Додавання нових змінних

- Дуже часто в аналізі даних (і в машинному навчанні) потрібно створювати нові змінні
- Це часто є етапом попередньої підготовки даних, коли з існуючих стовпців потрібно утворити ті, які мають більший сенс
- Часто це потрібно робити на етапі самого аналізу для утворення нових змінних (напр., логаритмування чи піднесення до квадрату тощо)
- Для утворення нових змінних потрібно використати функцію `mutate()`
- Аргументами повинні стати пари *новий стовпець–функція від старих стовпців*

```
passengers %>% mutate(FamSize = Parch + SibSp)
```

```
## # A tibble: 891 x 13
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1           1        0      3 Braun~ male     22     1     0 A/5 2~  7.25 <NA>
##  2           2        1      1 Cumin~ fema~    38     1     0 PC 17~ 71.3  C85
##  3           3        1      3 Heikk~ fema~    26     0     0 STON/~  7.92 <NA>
##  4           4        1      1 Futre~ fema~    35     1     0 113803 53.1  C123
##  5           5        0      3 Allen~ male     35     0     0 373450  8.05 <NA>
##  6           6        0      3 Moran~ male     NA     0     0 330877  8.46 <NA>
##  7           7        0      1 McCar~ male     54     0     0 17463  51.9  E46
##  8           8        0      3 Palss~ male      2     3     1 349909 21.1  <NA>
##  9           9        1      3 Johns~ fema~    27     0     2 347742 11.1  <NA>
## 10          10        1      2 Nasse~ fema~    14     1     0 237736 30.1  <NA>
## # ... with 881 more rows, and 2 more variables: Embarked <chr>, FamSize <dbl>
```

- Ми додали новий стовпець `FamSize` (кількість членів сім'ї) як суму батьків і дітей (`Parch`) та братів і сестер (`SibSp`), які пливли разом із пасажиром
- До змінних можна застосовувати дуже багато різних функцій
- У відповідних джерелах завжди можна знайти, як реалізувати ту чи ту ідею

Додавання нових змінних

- Дуже часто в аналізі даних (і в машинному навчанні) потрібно створювати нові змінні
- Це часто є етапом попередньої підготовки даних, коли з існуючих стовпців потрібно утворити ті, які мають більший сенс
- Часто це потрібно робити на етапі самого аналізу для утворення нових змінних (напр., логаритмування чи піднесення до квадрату тощо)
- Для утворення нових змінних потрібно використати функцію `mutate()`
- Аргументами повинні стати пари *новий стовпець–функція від старих стовпців*

```
passengers %>% mutate(FamSize = Parch + SibSp)
```

```
## # A tibble: 891 x 13
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1           1        0      3 Braun~ male     22     1     0 A/5 2~  7.25 <NA>
##  2           2        1      1 Cumin~ fema~    38     1     0 PC 17~ 71.3  C85
##  3           3        1      3 Heikk~ fema~    26     0     0 STON/~  7.92 <NA>
##  4           4        1      1 Futre~ fema~    35     1     0 113803 53.1  C123
##  5           5        0      3 Allen~ male     35     0     0 373450  8.05 <NA>
##  6           6        0      3 Moran~ male     NA     0     0 330877  8.46 <NA>
##  7           7        0      1 McCar~ male     54     0     0 17463  51.9  E46
##  8           8        0      3 Palss~ male      2     3     1 349909 21.1  <NA>
##  9           9        1      3 Johns~ fema~    27     0     2 347742 11.1  <NA>
## 10          10        1      2 Nasse~ fema~    14     1     0 237736 30.1  <NA>
## # ... with 881 more rows, and 2 more variables: Embarked <chr>, FamSize <dbl>
```

- Ми додали новий стовпець `FamSize` (кількість членів сім'ї) як суму батьків і дітей (`Parch`) та братів і сестер (`SibSp`), які пливли разом із пасажиром
- До змінних можна застосовувати дуже багато різних функцій
- У відповідних джерелах завжди можна знайти, як реалізувати ту чи ту ідею

Додавання нових змінних

- Дуже часто в аналізі даних (і в машинному навчанні) потрібно створювати нові змінні
- Це часто є етапом попередньої підготовки даних, коли з існуючих стовпців потрібно утворити ті, які мають більший сенс
- Часто це потрібно робити на етапі самого аналізу для утворення нових змінних (напр., логаритмування чи піднесення до квадрату тощо)
- Для утворення нових змінних потрібно використати функцію `mutate()`
- Аргументами повинні стати пари *новий стовпець–функція від старих стовпців*

```
passengers %>% mutate(FamSize = Parch + SibSp)
```

```
## # A tibble: 891 x 13
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1           1        0      3 Braun~ male     22     1     0 A/5 2~  7.25 <NA>
##  2           2        1      1 Cumin~ fema~    38     1     0 PC 17~ 71.3  C85
##  3           3        1      3 Heikk~ fema~    26     0     0 STON/~  7.92 <NA>
##  4           4        1      1 Futre~ fema~    35     1     0 113803 53.1  C123
##  5           5        0      3 Allen~ male     35     0     0 373450  8.05 <NA>
##  6           6        0      3 Moran~ male     NA     0     0 330877  8.46 <NA>
##  7           7        0      1 McCar~ male     54     0     0 17463  51.9  E46
##  8           8        0      3 Palss~ male      2     3     1 349909 21.1  <NA>
##  9           9        1      3 Johns~ fema~    27     0     2 347742 11.1  <NA>
## 10          10        1      2 Nasse~ fema~    14     1     0 237736 30.1  <NA>
## # ... with 881 more rows, and 2 more variables: Embarked <chr>, FamSize <dbl>
```

- Ми додали новий стовпець `FamSize` (кількість членів сім'ї) як суму батьків і дітей (`Parch`) та братів і сестер (`SibSp`), які пливли разом із пасажиром
- До змінних можна застосовувати дуже багато різних функцій
- У відповідних джерелах завжди можна знайти, як реалізувати ту чи ту ідею

Додавання нових змінних

- Дуже часто в аналізі даних (і в машинному навчанні) потрібно створювати нові змінні
- Це часто є етапом попередньої підготовки даних, коли з існуючих стовпців потрібно утворити ті, які мають більший сенс
- Часто це потрібно робити на етапі самого аналізу для утворення нових змінних (напр., логаритмування чи піднесення до квадрату тощо)
- Для утворення нових змінних потрібно використати функцію `mutate()`
- Аргументами повинні стати пари *новий стовпець–функція від старих стовпців*

```
passengers %>% mutate(FamSize = Parch + SibSp)
```

```
## # A tibble: 891 x 13
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1           1        0      3 Braun~ male     22     1     0 A/5 2~  7.25 <NA>
##  2           2        1      1 Cumin~ fema~    38     1     0 PC 17~ 71.3  C85
##  3           3        1      3 Heikk~ fema~    26     0     0 STON/~  7.92 <NA>
##  4           4        1      1 Futre~ fema~    35     1     0 113803 53.1  C123
##  5           5        0      3 Allen~ male     35     0     0 373450  8.05 <NA>
##  6           6        0      3 Moran~ male     NA     0     0 330877  8.46 <NA>
##  7           7        0      1 McCar~ male     54     0     0 17463  51.9  E46
##  8           8        0      3 Palss~ male      2     3     1 349909 21.1  <NA>
##  9           9        1      3 Johns~ fema~    27     0     2 347742 11.1  <NA>
## 10          10        1      2 Nasse~ fema~    14     1     0 237736 30.1  <NA>
## # ... with 881 more rows, and 2 more variables: Embarked <chr>, FamSize <dbl>
```

- Ми додали новий стовпець `FamSize` (кількість членів сім'ї) як суму батьків і дітей (`Parch`) та братів і сестер (`SibSp`), які пливли разом із пасажиром
- До змінних можна застосовувати дуже багато різних функцій
- У відповідних джерелах завжди можна знайти, як реалізувати ту чи ту ідею

Комбінування різних операцій

- Майже завжди потрібно застосувати декілька операцій одночасно
- Результат кожної з них можна зберігати в окремій змінній
- Для підвищення читовности коду ці операції ліпше поєднувати за допомогою т.зв. **pipes** (підстановок):

```
passengers %>% mutate(FamSize = Parch + SibSp) %>%
    arrange(desc(FamSize))
```

```
## # A tibble: 891 x 13
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1         160        0      3 "Sage~ male     NA     8     2 CA. 2~  69.6 <NA>
##  2         181        0      3 "Sage~ fema~    NA     8     2 CA. 2~  69.6 <NA>
##  3         202        0      3 "Sage~ male     NA     8     2 CA. 2~  69.6 <NA>
##  4         325        0      3 "Sage~ male     NA     8     2 CA. 2~  69.6 <NA>
##  5         793        0      3 "Sage~ fema~    NA     8     2 CA. 2~  69.6 <NA>
##  6         847        0      3 "Sage~ male     NA     8     2 CA. 2~  69.6 <NA>
##  7         864        0      3 "Sage~ fema~    NA     8     2 CA. 2~  69.6 <NA>
##  8          60        0      3 "Good~ male     11     5     2 CA 21~  46.9 <NA>
##  9          72        0      3 "Good~ fema~    16     5     2 CA 21~  46.9 <NA>
## 10         387        0      3 "Good~ male      1     5     2 CA 21~  46.9 <NA>
## # ... with 881 more rows, and 2 more variables: Embarked <chr>, FamSize <dbl>
```

- Ми створили нову змінну і відразу відсортували датафрейм за спаданням її значень
- Можна помітити, що великі родини мають тенденцію не виживати

Комбінування різних операцій

- Майже завжди потрібно застосувати декілька операцій одночасно
- Результат кожної з них можна зберігати в окремій змінній
- Для підвищення читовности коду ці операції ліпше поєднувати за допомогою т.зв. **pipes** (підстановок):

```
passengers %>% mutate(FamSize = Parch + SibSp) %>%
    arrange(desc(FamSize))
```

```
## # A tibble: 891 x 13
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1         160        0      3 "Sage~ male     NA     8     2 CA. 2~  69.6 <NA>
##  2         181        0      3 "Sage~ fema~    NA     8     2 CA. 2~  69.6 <NA>
##  3         202        0      3 "Sage~ male     NA     8     2 CA. 2~  69.6 <NA>
##  4         325        0      3 "Sage~ male     NA     8     2 CA. 2~  69.6 <NA>
##  5         793        0      3 "Sage~ fema~    NA     8     2 CA. 2~  69.6 <NA>
##  6         847        0      3 "Sage~ male     NA     8     2 CA. 2~  69.6 <NA>
##  7         864        0      3 "Sage~ fema~    NA     8     2 CA. 2~  69.6 <NA>
##  8          60        0      3 "Good~ male     11     5     2 CA 21~  46.9 <NA>
##  9          72        0      3 "Good~ fema~    16     5     2 CA 21~  46.9 <NA>
## 10         387        0      3 "Good~ male      1     5     2 CA 21~  46.9 <NA>
## # ... with 881 more rows, and 2 more variables: Embarked <chr>, FamSize <dbl>
```

- Ми створили нову змінну і відразу відсортували датафрейм за спаданням її значень
- Можна помітити, що великі родини мають тенденцію не виживати

Комбінування різних операцій

- Майже завжди потрібно застосувати декілька операцій одночасно
- Результат кожної з них можна зберігати в окремій змінній
- Для підвищення читовности коду ці операції ліпше поєднувати за допомогою т.зв. **pipes** (підстановок):

```
passengers %>% mutate(FamSize = Parch + SibSp) %>%
    arrange(desc(FamSize))
```

```
## # A tibble: 891 x 13
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1         160        0      3 "Sage~ male     NA     8     2 CA. 2~  69.6 <NA>
##  2         181        0      3 "Sage~ fema~    NA     8     2 CA. 2~  69.6 <NA>
##  3         202        0      3 "Sage~ male     NA     8     2 CA. 2~  69.6 <NA>
##  4         325        0      3 "Sage~ male     NA     8     2 CA. 2~  69.6 <NA>
##  5         793        0      3 "Sage~ fema~    NA     8     2 CA. 2~  69.6 <NA>
##  6         847        0      3 "Sage~ male     NA     8     2 CA. 2~  69.6 <NA>
##  7         864        0      3 "Sage~ fema~    NA     8     2 CA. 2~  69.6 <NA>
##  8          60        0      3 "Good~ male     11     5     2 CA 21~  46.9 <NA>
##  9          72        0      3 "Good~ fema~    16     5     2 CA 21~  46.9 <NA>
## 10         387        0      3 "Good~ male      1     5     2 CA 21~  46.9 <NA>
## # ... with 881 more rows, and 2 more variables: Embarked <chr>, FamSize <dbl>
```

- Ми створили нову змінну і відразу відсортували датафрейм за спаданням її значень
- Можна помітити, що великі родини мають тенденцію не виживати

Комбінування різних операцій

- Майже завжди потрібно застосувати декілька операцій одночасно
- Результат кожної з них можна зберігати в окремій змінній
- Для підвищення читовности коду ці операції ліпше поєднувати за допомогою т.зв. **pipes** (підстановок):

```
passengers %>% mutate(FamSize = Parch + SibSp) %>%
    arrange(desc(FamSize))
```

```
## # A tibble: 891 x 13
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1         160        0      3 "Sage~ male     NA     8     2 CA. 2~  69.6 <NA>
##  2         181        0      3 "Sage~ fema~    NA     8     2 CA. 2~  69.6 <NA>
##  3         202        0      3 "Sage~ male     NA     8     2 CA. 2~  69.6 <NA>
##  4         325        0      3 "Sage~ male     NA     8     2 CA. 2~  69.6 <NA>
##  5         793        0      3 "Sage~ fema~    NA     8     2 CA. 2~  69.6 <NA>
##  6         847        0      3 "Sage~ male     NA     8     2 CA. 2~  69.6 <NA>
##  7         864        0      3 "Sage~ fema~    NA     8     2 CA. 2~  69.6 <NA>
##  8          60        0      3 "Good~ male     11     5     2 CA 21~  46.9 <NA>
##  9          72        0      3 "Good~ fema~    16     5     2 CA 21~  46.9 <NA>
## 10         387        0      3 "Good~ male      1     5     2 CA 21~  46.9 <NA>
## # ... with 881 more rows, and 2 more variables: Embarked <chr>, FamSize <dbl>
```

- Ми створили нову змінну і відразу відсортували датафрейм за спаданням її значень
- Можна помітити, що великі родини мають тенденцію не виживати

Комбінування різних операцій

- Майже завжди потрібно застосувати декілька операцій одночасно
- Результат кожної з них можна зберігати в окремій змінній
- Для підвищення читовности коду ці операції ліпше поєднувати за допомогою т.зв. **pipes** (підстановок):

```
passengers %>% mutate(FamSize = Parch + SibSp) %>%
    arrange(desc(FamSize))
```

```
## # A tibble: 891 x 13
##    PassengerId Survived Pclass Name   Sex     Age SibSp Parch Ticket  Fare Cabin
##          <dbl>    <dbl>  <dbl> <chr>  <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr>
##  1         160        0      3 "Sage~ male     NA     8     2 CA. 2~  69.6 <NA>
##  2         181        0      3 "Sage~ fema~    NA     8     2 CA. 2~  69.6 <NA>
##  3         202        0      3 "Sage~ male     NA     8     2 CA. 2~  69.6 <NA>
##  4         325        0      3 "Sage~ male     NA     8     2 CA. 2~  69.6 <NA>
##  5         793        0      3 "Sage~ fema~    NA     8     2 CA. 2~  69.6 <NA>
##  6         847        0      3 "Sage~ male     NA     8     2 CA. 2~  69.6 <NA>
##  7         864        0      3 "Sage~ fema~    NA     8     2 CA. 2~  69.6 <NA>
##  8          60        0      3 "Good~ male     11     5     2 CA 21~  46.9 <NA>
##  9          72        0      3 "Good~ fema~    16     5     2 CA 21~  46.9 <NA>
## 10         387        0      3 "Good~ male      1     5     2 CA 21~  46.9 <NA>
## # ... with 881 more rows, and 2 more variables: Embarked <chr>, FamSize <dbl>
```

- Ми створили нову змінну і відразу відсортували датафрейм за спаданням її значень
- Можна помітити, що великі родини мають тенденцію не виживати

Відбір окремих стовпців (1)

- Інколи для продовження ефективної роботи з датафреймом буває корисно позбутися зайвих стовпців
- Це можна здійснити за допомогою функції `select()`
- Аргументами повинні стати назви стовпців, які потрібно залишити

```
passengers %>% mutate(FamSize = Parch + SibSp) %>%
    select(Survived, Sex, Age, FamSize)
```

```
## # A tibble: 891 x 4
##    Survived Sex      Age FamSize
##       <dbl> <chr>  <dbl>   <dbl>
##  1        0 male      22       1
##  2        1 female    38       1
##  3        1 female    26       0
##  4        1 female    35       1
##  5        0 male      35       0
##  6        0 male      NA       0
##  7        0 male      54       0
##  8        0 male       2       4
##  9        1 female    27       2
## 10        1 female    14       1
## # ... with 881 more rows
```

Відбір окремих стовпців (1)

- Інколи для продовження ефективної роботи з датафреймом буває корисно позбутися зайвих стовпців
- Це можна здійснити за допомогою функції `select()`
- Аргументами повинні стати назви стовпців, які потрібно залишити

```
passengers %>% mutate(FamSize = Parch + SibSp) %>%
    select(Survived, Sex, Age, FamSize)
```

```
## # A tibble: 891 x 4
##    Survived Sex      Age FamSize
##       <dbl> <chr>  <dbl>   <dbl>
##  1        0 male      22       1
##  2        1 female    38       1
##  3        1 female    26       0
##  4        1 female    35       1
##  5        0 male      35       0
##  6        0 male      NA       0
##  7        0 male      54       0
##  8        0 male       2       4
##  9        1 female    27       2
## 10        1 female    14       1
## # ... with 881 more rows
```

Відбір окремих стовпців (1)

- Інколи для продовження ефективної роботи з датафреймом буває корисно позбутися зайвих стовпців
- Це можна здійснити за допомогою функції `select()`
- Аргументами повинні стати назви стовпців, які потрібно залишити

```
passengers %>% mutate(FamSize = Parch + SibSp) %>%
    select(Survived, Sex, Age, FamSize)
```

```
## # A tibble: 891 x 4
##    Survived Sex      Age FamSize
##       <dbl> <chr>  <dbl>   <dbl>
##  1        0 male      22       1
##  2        1 female    38       1
##  3        1 female    26       0
##  4        1 female    35       1
##  5        0 male      35       0
##  6        0 male      NA       0
##  7        0 male      54       0
##  8        0 male       2       4
##  9        1 female    27       2
## 10        1 female    14       1
## # ... with 881 more rows
```

Відбір окремих стовпців (2)

- Якщо знати порядок розташування стовпців, можна використовувати `:` для вказання першого й останнього стовпця з певного переліку (напр., `ticket:embarked`)
- Якщо стовпців дуже багато, а потрібно викинути тільки деякі, то можна використати від'ємну індексацію

```
passengers %>% mutate(FamSize = Parch + SibSp) %>%
    select(-c(Parch, SibSp))
```

```
## # A tibble: 891 x 11
##    Passeng~1 Survi~2 Pclass Name  Sex     Age Ticket  Fare Cabin Embar~3 FamSize
##        <dbl>   <dbl>  <dbl> <chr> <chr> <dbl> <chr>  <dbl> <chr> <chr>     <dbl>
##  1         1       0      3 Brau~ male     22 A/5 2~  7.25 <NA>  S             1
##  2         2       1      1 Cumi~ fema~    38 PC 17~ 71.3  C85   C             1
##  3         3       1      3 Heik~ fema~    26 STON/~  7.92 <NA>  S             0
##  4         4       1      1 Futr~ fema~    35 113803 53.1  C123  S             1
##  5         5       0      3 Alle~ male     35 373450  8.05 <NA>  S             0
##  6         6       0      3 Mora~ male     NA 330877  8.46 <NA>  Q             0
##  7         7       0      1 McCa~ male     54 17463  51.9  E46   S             0
##  8         8       0      3 Pals~ male      2 349909 21.1  <NA>  S             4
##  9         9       1      3 John~ fema~    27 347742 11.1  <NA>  S             2
## 10        10       1      2 Nass~ fema~    14 237736 30.1  <NA>  C             1
## # ... with 881 more rows, and abbreviated variable names 1: PassengerId,
## #   2: Survived, 3: Embarked
```

Відбір окремих стовпців (2)

- Якщо знати порядок розташування стовпців, можна використовувати `:` для вказання першого й останнього стовпця з певного переліку (напр., `ticket:embarked`)
- Якщо стовпців дуже багато, а потрібно викинути тільки деякі, то можна використати від'ємну індексацію

```
passengers %>% mutate(FamSize = Parch + SibSp) %>%
    select(-c(Parch, SibSp))
```

```
## # A tibble: 891 x 11
##    Passeng~1 Survi~2 Pclass Name  Sex     Age Ticket  Fare Cabin Embar~3 FamSize
##        <dbl>   <dbl>  <dbl> <chr> <chr> <dbl> <chr>  <dbl> <chr> <chr>     <dbl>
##  1         1       0      3 Brau~ male     22 A/5 2~  7.25 <NA>  S             1
##  2         2       1      1 Cumi~ fema~    38 PC 17~ 71.3  C85   C             1
##  3         3       1      3 Heik~ fema~    26 STON/~  7.92 <NA>  S             0
##  4         4       1      1 Futr~ fema~    35 113803 53.1  C123  S             1
##  5         5       0      3 Alle~ male     35 373450  8.05 <NA>  S             0
##  6         6       0      3 Mora~ male     NA 330877  8.46 <NA>  Q             0
##  7         7       0      1 McCa~ male     54 17463  51.9  E46   S             0
##  8         8       0      3 Pals~ male      2 349909 21.1  <NA>  S             4
##  9         9       1      3 John~ fema~    27 347742 11.1  <NA>  S             2
## 10        10       1      2 Nass~ fema~    14 237736 30.1  <NA>  C             1
## # ... with 881 more rows, and abbreviated variable names 1: PassengerId,
## #   2: Survived, 3: Embarked
```

Перейменування змінних

- Для того, щоб перейменувати деяку змінну, можна використати функцію `rename()`
- Аргументами повинні стати пари *нова назва–стара назва*

```
passengers %>% mutate(FamSize = Parch + SibSp) %>%
    rename(ID = PassengerId, Class = Pclass)
```

```
## # A tibble: 891 x 13
##       ID Survived Class Name  Sex     Age SibSp Parch Ticket  Fare Cabin Embar~1
##    <dbl>    <dbl> <dbl> <chr> <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr> <chr>
##  1     1        0     3 Brau~ male     22     1     0 A/5 2~  7.25 <NA>  S
##  2     2        1     1 Cumi~ fema~    38     1     0 PC 17~ 71.3  C85   C
##  3     3        1     3 Heik~ fema~    26     0     0 STON/~  7.92 <NA>  S
##  4     4        1     1 Futr~ fema~    35     1     0 113803 53.1  C123  S
##  5     5        0     3 Alle~ male     35     0     0 373450  8.05 <NA>  S
##  6     6        0     3 Mora~ male     NA     0     0 330877  8.46 <NA>  Q
##  7     7        0     1 McCa~ male     54     0     0 17463  51.9  E46   S
##  8     8        0     3 Pals~ male      2     3     1 349909 21.1  <NA>  S
##  9     9        1     3 John~ fema~    27     0     2 347742 11.1  <NA>  S
## 10    10        1     2 Nass~ fema~    14     1     0 237736 30.1  <NA>  C
## # ... with 881 more rows, 1 more variable: FamSize <dbl>, and abbreviated
## #   variable name 1: Embarked
```

Перейменування змінних

- Для того, щоб перейменувати деяку змінну, можна використати функцію `rename()`
- Аргументами повинні стати пари *нова назва–стара назва*

```
passengers %>% mutate(FamSize = Parch + SibSp) %>%
    rename(ID = PassengerId, Class = Pclass)
```

```
## # A tibble: 891 x 13
##       ID Survived Class Name  Sex     Age SibSp Parch Ticket  Fare Cabin Embar~1
##    <dbl>    <dbl> <dbl> <chr> <chr> <dbl> <dbl> <dbl> <chr>  <dbl> <chr> <chr>
##  1     1        0     3 Brau~ male     22     1     0 A/5 2~  7.25 <NA>  S
##  2     2        1     1 Cumi~ fema~    38     1     0 PC 17~ 71.3  C85   C
##  3     3        1     3 Heik~ fema~    26     0     0 STON/~  7.92 <NA>  S
##  4     4        1     1 Futr~ fema~    35     1     0 113803 53.1  C123  S
##  5     5        0     3 Alle~ male     35     0     0 373450  8.05 <NA>  S
##  6     6        0     3 Mora~ male     NA     0     0 330877  8.46 <NA>  Q
##  7     7        0     1 McCa~ male     54     0     0 17463  51.9  E46   S
##  8     8        0     3 Pals~ male      2     3     1 349909 21.1  <NA>  S
##  9     9        1     3 John~ fema~    27     0     2 347742 11.1  <NA>  S
## 10    10        1     2 Nass~ fema~    14     1     0 237736 30.1  <NA>  C
## # ... with 881 more rows, 1 more variable: FamSize <dbl>, and abbreviated
## #   variable name 1: Embarked
```

Підрахунок підсумкових показників

- Дуже часто потрібно підрахувати деякі підсумкові статистики для наших даних
- До таких належать вибіркове середнє, медіана, середньоквадратичне відхилення та багато інших
- Для роботи з датафреймами потрібно використовувати функцію `summarize()`
- Аргументами повинні стати пари *назва результату–функція від змінних*
- У результаті дістаємо новий датафрейм із відповідними стовпцями та одним рядком

```
passengers %>% summarize(meanFare = mean(Fare), medianFare = median(Fare))
```

```
## # A tibble: 1 x 2
##   meanFare medianFare
##      <dbl>      <dbl>
## 1     32.2       14.5
```

- Звісно, статистики можна рахувати і для деякої підмножини спостережень

```
passengers %>% filter(Sex == "male") %>%
    summarize(meanFare = mean(Fare), medianFare = median(Fare))
```

```
## # A tibble: 1 x 2
##   meanFare medianFare
##      <dbl>      <dbl>
## 1     25.5       10.5
```

Підрахунок підсумкових показників

- Дуже часто потрібно підрахувати деякі підсумкові статистики для наших даних
- До таких належать вибіркове середнє, медіана, середньоквадратичне відхилення та багато інших
- Для роботи з датафреймами потрібно використовувати функцію `summarize()`
- Аргументами повинні стати пари *назва результату–функція від змінних*
- У результаті дістаємо новий датафрейм із відповідними стовпцями та одним рядком

```
passengers %>% summarize(meanFare = mean(Fare), medianFare = median(Fare))
```

```
## # A tibble: 1 x 2
##   meanFare medianFare
##      <dbl>      <dbl>
## 1     32.2       14.5
```

- Звісно, статистики можна рахувати і для деякої підмножини спостережень

```
passengers %>% filter(Sex == "male") %>%
    summarize(meanFare = mean(Fare), medianFare = median(Fare))
```

```
## # A tibble: 1 x 2
##   meanFare medianFare
##      <dbl>      <dbl>
## 1     25.5       10.5
```

Підрахунок підсумкових показників

- Дуже часто потрібно підрахувати деякі підсумкові статистики для наших даних
- До таких належать вибіркове середнє, медіана, середньоквадратичне відхилення та багато інших
- Для роботи з датафреймами потрібно використовувати функцію `summarize()`
- Аргументами повинні стати пари *назва результату–функція від змінних*
- У результаті дістаємо новий датафрейм із відповідними стовпцями та одним рядком

```
passengers %>% summarize(meanFare = mean(Fare), medianFare = median(Fare))
```

```
## # A tibble: 1 x 2
##   meanFare medianFare
##      <dbl>      <dbl>
## 1     32.2       14.5
```

- Звісно, статистики можна рахувати і для деякої підмножини спостережень

```
passengers %>% filter(Sex == "male") %>%
    summarize(meanFare = mean(Fare), medianFare = median(Fare))
```

```
## # A tibble: 1 x 2
##   meanFare medianFare
##      <dbl>      <dbl>
## 1     25.5       10.5
```

Підрахунок підсумкових показників

- Дуже часто потрібно підрахувати деякі підсумкові статистики для наших даних
- До таких належать вибіркове середнє, медіана, середньоквадратичне відхилення та багато інших
- Для роботи з датафреймами потрібно використовувати функцію `summarize()`
- Аргументами повинні стати пари *назва результату–функція від змінних*
- У результаті дістаємо новий датафрейм із відповідними стовпцями та одним рядком

```
passengers %>% summarize(meanFare = mean(Fare), medianFare = median(Fare))
```

```
## # A tibble: 1 x 2
##   meanFare medianFare
##      <dbl>      <dbl>
## 1     32.2       14.5
```

- Звісно, статистики можна рахувати і для деякої підмножини спостережень

```
passengers %>% filter(Sex == "male") %>%
    summarize(meanFare = mean(Fare), medianFare = median(Fare))
```

```
## # A tibble: 1 x 2
##   meanFare medianFare
##      <dbl>      <dbl>
## 1     25.5       10.5
```

Підрахунок підсумкових показників

- Дуже часто потрібно підрахувати деякі підсумкові статистики для наших даних
- До таких належать вибіркове середнє, медіана, середньоквадратичне відхилення та багато інших
- Для роботи з датафреймами потрібно використовувати функцію `summarize()`
- Аргументами повинні стати пари *назва результату–функція від змінних*
- У результаті дістаємо новий датафрейм із відповідними стовпцями та одним рядком

```
passengers %>% summarize(meanFare = mean(Fare), medianFare = median(Fare))
```

```
## # A tibble: 1 x 2
##   meanFare medianFare
##      <dbl>      <dbl>
## 1     32.2       14.5
```

- Звісно, статистики можна рахувати і для деякої підмножини спостережень

```
passengers %>% filter(Sex == "male") %>%
    summarize(meanFare = mean(Fare), medianFare = median(Fare))
```

```
## # A tibble: 1 x 2
##   meanFare medianFare
##      <dbl>      <dbl>
## 1     25.5       10.5
```

Підрахунок підсумкових показників

- Дуже часто потрібно підрахувати деякі підсумкові статистики для наших даних
- До таких належать вибіркове середнє, медіана, середньоквадратичне відхилення та багато інших
- Для роботи з датафреймами потрібно використовувати функцію `summarize()`
- Аргументами повинні стати пари *назва результату–функція від змінних*
- У результаті дістаємо новий датафрейм із відповідними стовпцями та одним рядком

```
passengers %>% summarize(meanFare = mean(Fare), medianFare = median(Fare))
```

```
## # A tibble: 1 x 2
##   meanFare medianFare
##      <dbl>      <dbl>
## 1     32.2       14.5
```

- Звісно, статистики можна рахувати і для деякої підмножини спостережень

```
passengers %>% filter(Sex == "male") %>%
    summarize(meanFare = mean(Fare), medianFare = median(Fare))
```

```
## # A tibble: 1 x 2
##   meanFare medianFare
##      <dbl>      <dbl>
## 1     25.5       10.5
```

Підрахунок підсумкових показників за групами

- Дуже часто виникає потреба рахувати статистики для різних категорій одночасно
- Наприклад, нас можуть цікавити середнє та медіана вартости квитка для чоловіків і для жінок окремо
- Використовувати прийом із попереднього слайду нераціонально
- Тому спочатку дані **групують** за допомогою функції `group_by()`

```
passengers %>% group_by(Sex) %>%
    summarize(meanFare = mean(Fare), medianFare = median(Fare))
```

```
## # A tibble: 2 x 3
##   Sex    meanFare medianFare
##   <chr>     <dbl>      <dbl>
## 1 female     44.5         23
## 2 male       25.5         10.5
```

- У результаті маємо датафрейм, де кожний рядок відповідає окремій категорії
- Особливо корисною є функція `n()`, яка рахує число спостережень у групі

```
passengers %>% group_by(Sex) %>%
    summarize(meanFare = mean(Fare), propSurv = sum(Survived) / n())
```

```
## # A tibble: 2 x 3
##   Sex    meanFare propSurv
##   <chr>     <dbl>    <dbl>
## 1 female     44.5    0.742
## 2 male       25.5    0.189
```

- Ми порахували середню вартість квитка і **частку** вцілілих пасажирів
- Як можна бачити, результати такого **примітивного** аналізу свідчать, що серед жінок уцілілих значно більше, а квитки в них у середньому були дорожчі

Підрахунок підсумкових показників за групами

- Дуже часто виникає потреба рахувати статистики для різних категорій одночасно
- Наприклад, нас можуть цікавити середнє та медіана вартости квитка для чоловіків і для жінок окремо
- Використовувати прийом із попереднього слайду нераціонально
- Тому спочатку дані **групують** за допомогою функції `group_by()`

```
passengers %>% group_by(Sex) %>%
    summarize(meanFare = mean(Fare), medianFare = median(Fare))
```

```
## # A tibble: 2 x 3
##   Sex    meanFare medianFare
##   <chr>     <dbl>      <dbl>
## 1 female     44.5         23
## 2 male       25.5         10.5
```

- У результаті маємо датафрейм, де кожний рядок відповідає окремій категорії
- Особливо корисною є функція `n()`, яка рахує число спостережень у групі

```
passengers %>% group_by(Sex) %>%
    summarize(meanFare = mean(Fare), propSurv = sum(Survived) / n())
```

```
## # A tibble: 2 x 3
##   Sex    meanFare propSurv
##   <chr>     <dbl>    <dbl>
## 1 female     44.5    0.742
## 2 male       25.5    0.189
```

- Ми порахували середню вартість квитка і **частку** вцілілих пасажирів
- Як можна бачити, результати такого **примітивного** аналізу свідчать, що серед жінок уцілілих значно більше, а квитки в них у середньому були дорожчі

Підрахунок підсумкових показників за групами

- Дуже часто виникає потреба рахувати статистики для різних категорій одночасно
- Наприклад, нас можуть цікавити середнє та медіана вартости квитка для чоловіків і для жінок окремо
- Використовувати прийом із попереднього слайду нераціонально
- Тому спочатку дані **групують** за допомогою функції `group_by()`

```
passengers %>% group_by(Sex) %>%
    summarize(meanFare = mean(Fare), medianFare = median(Fare))
```

```
## # A tibble: 2 x 3
##   Sex    meanFare medianFare
##   <chr>     <dbl>      <dbl>
## 1 female     44.5         23
## 2 male       25.5         10.5
```

- У результаті маємо датафрейм, де кожний рядок відповідає окремій категорії
- Особливо корисною є функція `n()`, яка рахує число спостережень у групі

```
passengers %>% group_by(Sex) %>%
    summarize(meanFare = mean(Fare), propSurv = sum(Survived) / n())
```

```
## # A tibble: 2 x 3
##   Sex    meanFare propSurv
##   <chr>     <dbl>    <dbl>
## 1 female     44.5    0.742
## 2 male       25.5    0.189
```

- Ми порахували середню вартість квитка і **частку** вцілілих пасажирів
- Як можна бачити, результати такого **примітивного** аналізу свідчать, що серед жінок уцілілих значно більше, а квитки в них у середньому були дорожчі

Підрахунок підсумкових показників за групами

- Дуже часто виникає потреба рахувати статистики для різних категорій одночасно
- Наприклад, нас можуть цікавити середнє та медіана вартости квитка для чоловіків і для жінок окремо
- Використовувати прийом із попереднього слайду нераціонально
- Тому спочатку дані **групують** за допомогою функції `group_by()`

```
passengers %>% group_by(Sex) %>%
    summarize(meanFare = mean(Fare), medianFare = median(Fare))
```

```
## # A tibble: 2 x 3
##   Sex    meanFare medianFare
##   <chr>     <dbl>      <dbl>
## 1 female     44.5         23
## 2 male       25.5         10.5
```

- У результаті маємо датафрейм, де кожний рядок відповідає окремій категорії
- Особливо корисною є функція `n()`, яка рахує число спостережень у групі

```
passengers %>% group_by(Sex) %>%
    summarize(meanFare = mean(Fare), propSurv = sum(Survived) / n())
```

```
## # A tibble: 2 x 3
##   Sex    meanFare propSurv
##   <chr>     <dbl>    <dbl>
## 1 female     44.5    0.742
## 2 male       25.5    0.189
```

- Ми порахували середню вартість квитка і **частку** вцілілих пасажирів
- Як можна бачити, результати такого **примітивного** аналізу свідчать, що серед жінок уцілілих значно більше, а квитки в них у середньому були дорожчі

Підрахунок підсумкових показників за групами

- Дуже часто виникає потреба рахувати статистики для різних категорій одночасно
- Наприклад, нас можуть цікавити середнє та медіана вартости квитка для чоловіків і для жінок окремо
- Використовувати прийом із попереднього слайду нераціонально
- Тому спочатку дані **групують** за допомогою функції `group_by()`

```
passengers %>% group_by(Sex) %>%
    summarize(meanFare = mean(Fare), medianFare = median(Fare))
```

```
## # A tibble: 2 x 3
##   Sex    meanFare medianFare
##   <chr>     <dbl>      <dbl>
## 1 female     44.5         23
## 2 male       25.5         10.5
```

- У результаті маємо датафрейм, де кожний рядок відповідає окремій категорії
- Особливо корисною є функція `n()`, яка рахує число спостережень у групі

```
passengers %>% group_by(Sex) %>%
    summarize(meanFare = mean(Fare), propSurv = sum(Survived) / n())
```

```
## # A tibble: 2 x 3
##   Sex    meanFare propSurv
##   <chr>     <dbl>    <dbl>
## 1 female     44.5    0.742
## 2 male       25.5    0.189
```

- Ми порахували середню вартість квитка і **частку** вцілілих пасажирів
- Як можна бачити, результати такого **примітивного** аналізу свідчать, що серед жінок уцілілих значно більше, а квитки в них у середньому були дорожчі

Підрахунок підсумкових показників за групами

- Дуже часто виникає потреба рахувати статистики для різних категорій одночасно
- Наприклад, нас можуть цікавити середнє та медіана вартости квитка для чоловіків і для жінок окремо
- Використовувати прийом із попереднього слайду нераціонально
- Тому спочатку дані **групують** за допомогою функції `group_by()`

```
passengers %>% group_by(Sex) %>%
    summarize(meanFare = mean(Fare), medianFare = median(Fare))
```

```
## # A tibble: 2 x 3
##   Sex    meanFare medianFare
##   <chr>     <dbl>      <dbl>
## 1 female     44.5       23
## 2 male       25.5       10.5
```

- У результаті маємо датафрейм, де кожний рядок відповідає окремій категорії
- Особливо корисною є функція `n()`, яка рахує число спостережень у групі

```
passengers %>% group_by(Sex) %>%
    summarize(meanFare = mean(Fare), propSurv = sum(Survived) / n())
```

```
## # A tibble: 2 x 3
##   Sex    meanFare propSurv
##   <chr>     <dbl>    <dbl>
## 1 female     44.5    0.742
## 2 male       25.5    0.189
```

- Ми порахували середню вартість квитка і **частку** вцілілих пасажирів
- Як можна бачити, результати такого **примітивного** аналізу свідчать, що серед жінок уцілілих значно більше, а квитки в них у середньому були дорожчі

Підрахунок підсумкових показників за групами

- Дуже часто виникає потреба рахувати статистики для різних категорій одночасно
- Наприклад, нас можуть цікавити середнє та медіана вартости квитка для чоловіків і для жінок окремо
- Використовувати прийом із попереднього слайду нераціонально
- Тому спочатку дані **групують** за допомогою функції `group_by()`

```
passengers %>% group_by(Sex) %>%
    summarize(meanFare = mean(Fare), medianFare = median(Fare))
```

```
## # A tibble: 2 x 3
##   Sex    meanFare medianFare
##   <chr>     <dbl>      <dbl>
## 1 female     44.5         23
## 2 male       25.5         10.5
```

- У результаті маємо датафрейм, де кожний рядок відповідає окремій категорії
- Особливо корисною є функція `n()`, яка рахує число спостережень у групі

```
passengers %>% group_by(Sex) %>%
    summarize(meanFare = mean(Fare), propSurv = sum(Survived) / n())
```

```
## # A tibble: 2 x 3
##   Sex    meanFare propSurv
##   <chr>     <dbl>    <dbl>
## 1 female     44.5    0.742
## 2 male       25.5    0.189
```

- Ми порахували середню вартість квитка і **частку** вцілілих пасажирів
- Як можна бачити, результати такого **примітивного** аналізу свідчать, що серед жінок уцілілих значно більше, а квитки в них у середньому були дорожчі

Підрахунок підсумкових показників за групами

- Дуже часто виникає потреба рахувати статистики для різних категорій одночасно
- Наприклад, нас можуть цікавити середнє та медіана вартости квитка для чоловіків і для жінок окремо
- Використовувати прийом із попереднього слайду нераціонально
- Тому спочатку дані **групують** за допомогою функції `group_by()`

```
passengers %>% group_by(Sex) %>%
    summarize(meanFare = mean(Fare), medianFare = median(Fare))
```

```
## # A tibble: 2 x 3
##   Sex    meanFare medianFare
##   <chr>     <dbl>      <dbl>
## 1 female     44.5         23
## 2 male       25.5         10.5
```

- У результаті маємо датафрейм, де кожний рядок відповідає окремій категорії
- Особливо корисною є функція `n()`, яка рахує число спостережень у групі

```
passengers %>% group_by(Sex) %>%
    summarize(meanFare = mean(Fare), propSurv = sum(Survived) / n())
```

```
## # A tibble: 2 x 3
##   Sex    meanFare propSurv
##   <chr>     <dbl>    <dbl>
## 1 female     44.5    0.742
## 2 male       25.5    0.189
```

- Ми порахували середню вартість квитка і **частку** вцілілих пасажирів
- Як можна бачити, результати такого **примітивного** аналізу свідчать, що серед жінок уцілілих значно більше, а квитки в них у середньому були дорожчі