МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИИТ)
Кафедра прикладной математики (ПМ)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №11

по дисциплине «Языки программирования для статистической обработки данных»

Студент группы *ИМБО-11-23, Журавлев Ф. А.* _____
(подпись)

Преподаватель *Трушин СМ*

_____
(подпись)

Москва 2025 г.

1) ЦЕЛЬ И ЗАДАЧИ

Цель практической работы:

Освоить методы создания интерактивных графиков в Python, R.

Задачи практической работы:

1. Создать интерактивные графики:
   - Python: использование библиотеки plotly для построения графиков с возможностью взаимодействия (зум, фильтры).
   - R: построение интерактивных панелей с использованием пакета shiny.
2. Работа с большими данными:
   - Использовать фильтры, агрегацию и другие инструменты для упрощения анализа.
3. Сравнить подходы работы с большими данными в Python (pandas, dask), R (data.table).
4. Оптимизировать обработку данных:
   - Анализ производительности каж дого инструмента при работе с большим объёмом данных.

2) РЕЗУЛЬТАТЫ ПРАКТИКИ

Шаг 1) Создание интерактивных графиков и работа с большими данными.

1.1) Создание интерактивных графиков и работа с большими данными в Python.

Для начала работы необходимо установить все необходимые библиотеки, так как их достаточно много, то ниже написан код, который показывает все библиотеки, нужные для практической работы.

Рисунок 1.1 — Необходимые библиотеки.

```
[1] pip install plotly
```

```
Requirement already satisfied: plotly in /usr/local/lib/python3.11/dist-packages (5.24.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.11/dist-packages (from plotly) (9.1.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from plotly) (24.2)
```

```
import plotly.express as px
import plotly.graph_objects as go
import pandas as pd
```

Далее убедитесь, что у нас есть столбцы, из типа «value» и «category». Для дальнейшего построения интерактивных графиков.

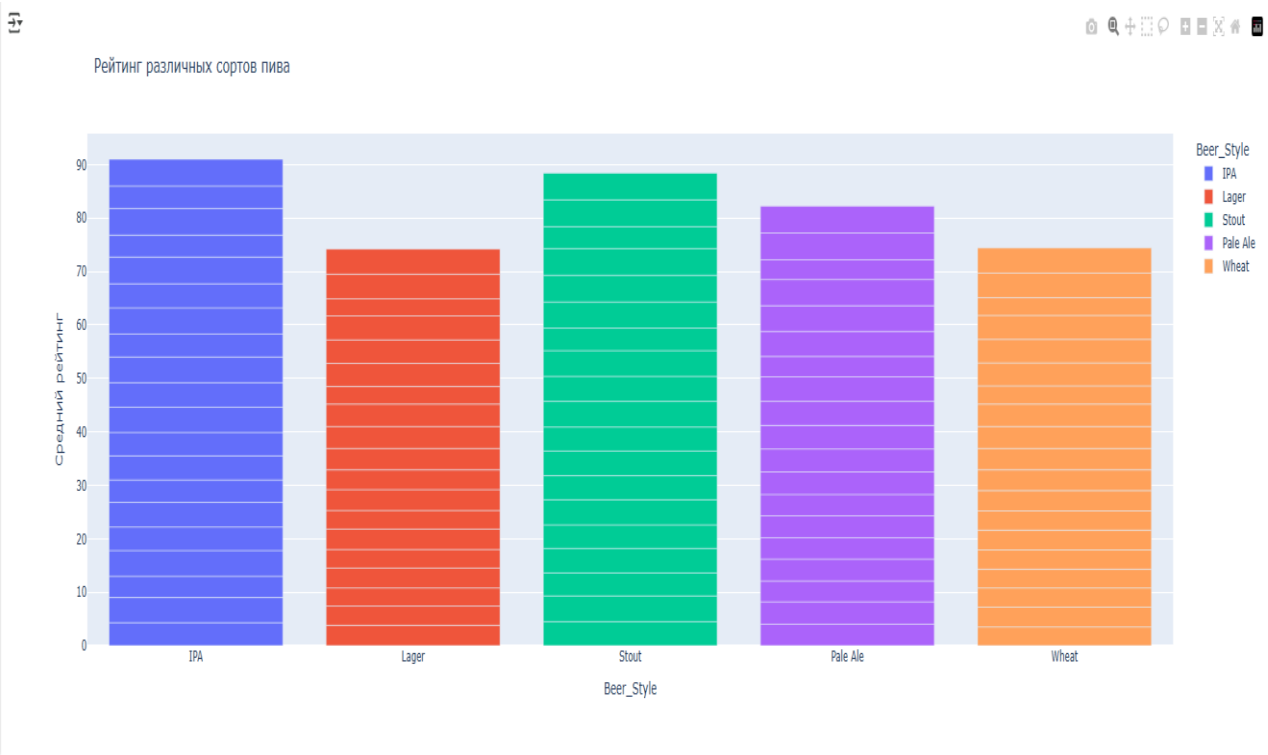
После напишем код, благодаря которому сможем увидеть пример подобного графика:

Рисунок 1.2 – Код для построения графика.

```
fig = px.bar(ds,
             x='Beer_Style',
             y='Rating',
             color='Beer_Style',  # раскрасить столбцы по сорту пива
             labels={'Rating': 'Средний рейтинг'},  # переименовать подписи
             title="Рейтинг различных сортов пива",
             height=600)  # увеличить высоту графика
fig.show()
```

Построим какую красоту нам покажет этот код:

Рисунок 1.3 — Интеративный график.



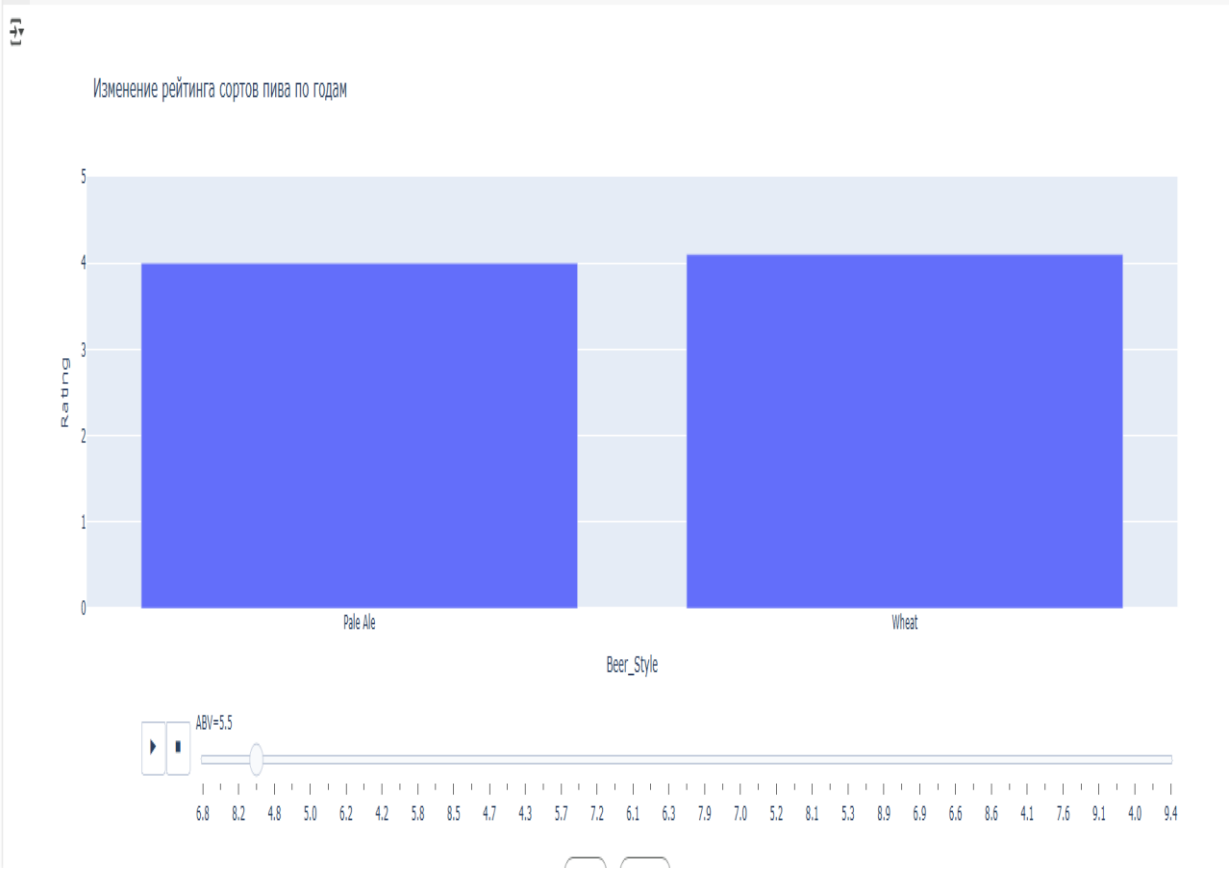
Далее можно написать код, благодаря которому мы сможем **анимировать** наши графики. Ниже представлен максимально простой способ:

Рисунок 1.4 - Код для анимации.

```
import plotly.express as px

# Предположим, в данных есть столбец 'year' с годом оценки
fig = px.bar(ds,
             x='Beer_Style',
             y='Rating',
             animation_frame='ABV',  # анимировать по годам
             title="Изменение рейтинга сортов пива по годам",
             range_y=[0, 5])  # фиксируем шкалу рейтинга (если max=5)
fig.show()
```

Далее посмотрим, какой результат выводит этот код:

Рисунок 1.5 - Анимированный график.



Шаг 2) Создание интерактивных графиков и работа с большими данными.

2.1) Создание интерактивных графиков и работа с большими данными в RStudio.

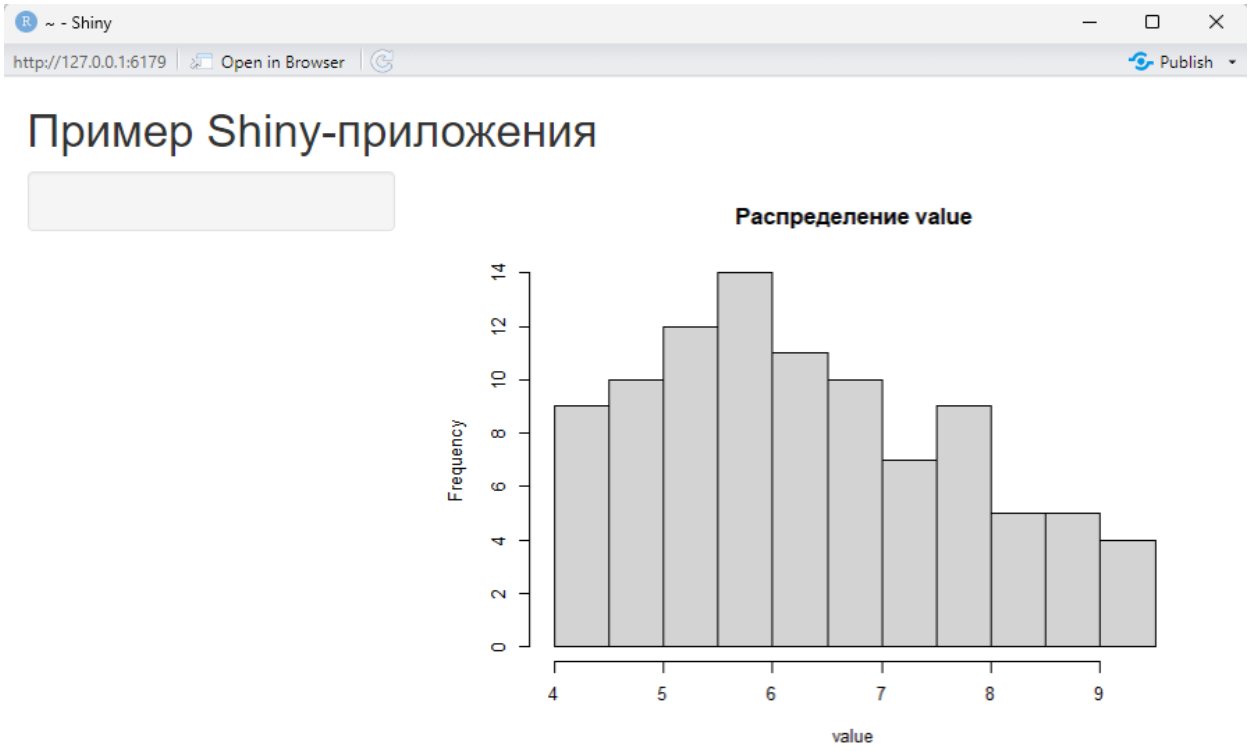
Далее воспользуемся специальным сервером и библиотекой Shiny, для визуализации интерактивых графиков. Ниже представлен максимально простой способ:

Рисунок 2.1 — код в R.

```
# Минимальный шаблон
ui <- fluidPage(
  titlePanel("Пример Shiny-приложения"),
  sidebarLayout(
    sidebarPanel(
      # Элементы управления (input)
    ),
    mainPanel(
      # Вывод графиков (output)
      plotOutput("distPlot")
    )
  )
)

server <- function(input, output) {
  # Загрузка данных (или загрузка до ui/server)
  df <- read.csv(file.choose())

  output$distPlot <- renderPlot({
    # Рисуем базовый график
    hist(df$ABV, main="Распределение value", xlab="value")
  })
}

shinyApp(ui = ui, server = server)
```

Посмотрим, что он нам выводит:

Рисунок 2.2 — Пример Shiny-приложения.



Далее добавим интерактивных элементов:

Рисунок 2.3 — График #1.

Дело в том, что мы не ограничены только стилем “Stout”, мы можем выбирать все категории пива, которые присутствуют в нашем наборе данных в этом приложении. Например вот пример для “Lager”

Рисунок 2.4 — График #2.

3.1) Работа с большими данными в Python.

Далее немного поработаем с большими данными, загрузим дата сет и отфильтруем только нужные столбцы, которые гипотетические будут нужны нам для работы, а также воспользуемся библиотекой dask, которая помогает работать ну прямо с огромными данными, которые разделенные на специальные чанки, ниже представлены различные коды:

Рисунок 3.1 — Выборка нужного из дата сета.

```
# Допустим, нам нужны только столбцы 'beer_style' и 'rating' (аналог usecols)
df = beer_df[['beer_style', 'rating']].copy()  # "usecols"

# Если нужно ограничить количество строк (аналог nrows)
df = df.head(5000)  # Ограничиваем 5000 строками вместо 10M

print(df)
```

```
      beer_style  rating
0           Sour     4.4
1         Kölsch     4.2
2         Saison     4.2
3     Barleywine     2.8
4       Pale Ale     2.5
...          ...     ...
4995     Pilsner     4.3
4996        Bock     3.9
4997     Red Ale     3.5
4998       Stout     4.8
4999     Red Ale     3.4

[5000 rows x 2 columns]
```

Рисунок 3.2 — Выборка с помощью dask.

```
# Использование
ddf = generate_dask_data(n_rows=1_000_000)  # Начнем с 1M строк для теста
result = ddf.groupby('beer_style')['rating'].mean().compute()
print(result)
```

```
beer_style
IPA        3.786399
Pilsner    3.789816
Stout      3.788175
Name: rating, dtype: float64
```

3.2) Работа с большими данными в RStudio.

Для начала установим библиотеку data.table, для того, чтобы быстрее считывать большие данные. Далее с помощью нее отфильтруем нужные нам значения:

Рисунок 3.3 — Фильтрация.

```
dt <- fread("C:/Users/shamk/OneDrive/Рабочий стол/6-8pivo.csv")
# fread() быстрее считывает, чем base read.csv
# Пример группировки:
dt[, .(mean_value = mean(Rating, na.rm=TRUE)), by=Beer_Style]
```

А затем произведем агрегацию, чисто протестировав код для примера:

Рисунок 3.4 — Агрегация.

```
subset_dt <- dt[Rating > 10 & Beer_Style == "Stout"]
# Агрегация
agg_dt <- dt[, .(sum_val = sum(Rating)), by=.(Beer_Style, Season)]
```

В нашем дата сете не оказалось таких значений, но это и не страшно, главное что сам код работает верно и подойдет для различных практических заданий.

Рисунок 3.5 — Результат агрегации.

	Date	Beer_Style	ABV	IBU	SRM	pH	Rating	Season	Production_Volume
No data available in table									

3 ВЫВОДЫ

В результате 11 практической работы мы научились строить интегративные графики в python и r, научились строить shiny-приложение, хоть и самое простое, а также чуть-чуть поработали с большими данными, для наглядности реализовав функции data.table из r, и dask из python.