

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Пензенский государственный университет» (ПГУ)

С. В. Рындина

Бизнес-аналитика на основе
больших данных:
создание приложений
на языках PYTHON и R

Учебно-методическое пособие

Пенза
Издательство ПГУ
2021

УДК 004.6(075)

P95

Р е ц е н з е н т

кандидат технических наук, доцент

А. А. Масленников

Рындина, Светлана Валентиновна.

P95 Бизнес-аналитика на основе больших данных: создание приложений на языках Python и R : учеб.-метод. пособие / С. В. Рындина. – Пенза : Изд-во ПГУ, 2021. – 76 с.

Рассмотрены возможности языков программирования Python и R для создания интерактивных приложений на основе аналитики данных. Приведены задания для лабораторных работ по созданию приложений. Материал пособия соответствует программе дисциплины «Бизнес-аналитика на основе больших данных».

Издание подготовлено на кафедре «Цифровая экономика» ПГУ и предназначено для обучающихся по направлению подготовки 38.03.05 «Бизнес-информатика», а также может быть использовано при изучении дисциплины «Бизнес-прогнозирование» и написании выпускной работы бакалавра.

УДК 004.6(075)

© Пензенский государственный
университет, 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	4
1. ПРЕВРАЩЕНИЕ ДАННЫХ В ЗНАНИЯ.....	5
1.1. Основы визуализации для создания веб-приложений	5
1.2. Microsoft Power BI для визуализации данных	10
2. СОЗДАНИЕ ВЕБ-ПРИЛОЖЕНИЙ НА ЯЗЫКЕ R С ПОМОЩЬЮ СРЕДЫ РАЗРАБОТКИ RSTUDIO	25
2.1. Пакет shiny	25
2.2. Управление выводом таблицы на экран, виджеты.....	28
2.3. Создание графиков	39
2.4. Создание динамичных пользовательских интерфейсов	44
2.5. Загрузка данных	48
2.6. Лабораторная работа «Создание веб-приложения на языке R».....	50
3. СОЗДАНИЕ ВЕБ-ПРИЛОЖЕНИЙ НА ЯЗЫКЕ PYTHON С ПОМОЩЬЮ СРЕДЫ РАЗРАБОТКИ ANACONDA И ТЕРМИНАЛА PROMT	51
3.1. Библиотека streamlit.....	51
3.2. Управление выводом таблицы на экран, виджеты.....	53
3.3. Создание графиков	59
3.4. Построение карт.....	67
3.5. Загрузка данных	69
3.6. Лабораторная работа «Создание веб-приложения на языке Python».....	73
СПИСОК ЛИТЕРАТУРЫ	74

ВВЕДЕНИЕ

В соответствии с учебным планом студенты третьего курса направления подготовки 38.03.05 «Бизнес-информатика» изучают дисциплину «Бизнес-аналитика на основе больших данных».

Важным элементом аналитики является встраивание ее в продукт. Продукты могут быть физические с цифровой компонентой, когда благодаря технологиям интернета вещей сенсоры и датчики, размещенные на физическом объекте или встроенные в него, позволяют собирать данные мониторинга его функционирования или коммуникации с этим продуктом других устройств, конечных пользователей. Сырые данные с регистрирующих устройств (датчиков, сенсоров) поступают в приложение для обработки данных, в котором подвергаются структурированию и из них извлекается полезная информация: о режимах взаимодействия, режимах функционирования и т.п. Продукты могут быть без физической составляющей, изначально цифровые и спроектированные для цифровой среды, логирование взаимодействия с ними пользователей генерирует огромные объемы потенциально ценных данных, из которых при обработке можно извлечь полезные закономерности и зависимости.

Задачи управления таким продуктом или задачи развития продукта решаются не напрямую, т.е. алгоритмически, а с помощью машинного обучения (machine learning, ML) на основе анализа имеющихся данных.

Серьезные продуктовые разработки включают не только реализацию ML-управления приложений и устройств (ML означает под управлением машинного обучения). Цифровой продукт с ML-управлением в самой начальной комплектации можно создать, не привлекая программирования на языках C++, Java и т.п., не реализуя html разметку, не используя JavaScript и php, а только на основе пакетных решений и библиотек Python и R, работа с которыми незначительно отличается для бизнес-аналитика от реализации моделей машинного обучения.

В учебно-методическом пособии рассматривается, как на основе библиотек и пакетов на языках программирования Python и R создать веб-приложение для визуализации работы с данными.

1. ПРЕВРАЩЕНИЕ ДАННЫХ В ЗНАНИЯ

Данные, собираемые бизнесом, на начальном этапе – сырые, т.е. исходные необработанные данные в своем первоначальном виде, которые получены вследствие функционирования или наблюдения за некоторой системой, объектом, группой объектов.

Визуализация данных – коммуникационное средство для понимания информации, которая содержится в данных. Извлечение выгодных для бизнеса зависимостей, важных сигналов становится достижимой задачей, если спроектировать удобный и полезный информационный продукт на основе данных.

Для бизнеса при исследовании данных имеют значение три параметра:

- информативность: возможность получить новое знание и понимание этого знания;
- ценность: возможность использовать полученное знание для уменьшения бизнес-рисков, решения бизнес-проблем, получения бизнес-выгоды;
- использование: возможность превратить ценность в конкретную последовательность действий, реализовать ценность, которая заложена в полученном знании.

1.1. Основы визуализации для создания веб-приложений

Стивен Фью (Stephen Few) [1] определяет три типа сигналов:

- заметные изменения в паттернах показателей (изменение трендов);
- заметные изменения в размахе/величине;
- появление уже известного и заметного паттерна.

Если в данных появляются новые закономерности – это потенциальный сигнал. Если диапазон значений показателя резко изменился (увеличился или уменьшился) – это потенциальный сигнал. Если в данных мы начинаем видеть признаки хороших или плохих перемен – это тоже потенциальный сигнал.

Ключевые способы сравнения данных [2]:

- сравнение по категориям:
 - изменения по категории;
 - изменения во времени;
 - изменения в пространстве;

- отношения между категориями;
- сравнение по показателям:
 - изменения показателя;
 - отношения между показателями.

Исследуя данные согласно основным видам сравнения, можно получить представление о наборе категорий и показателей, которое дает фундамент для поиска дальнейших сигналов.

К визуализации относится не только дашборд (dashboard, приборная панель) – интерактивный отчет в реальном времени по всем метрикам и каналам, включающий обычно различные варианты графического представления данных, но и отображение данных в виде таблицы. По мнению Стивена Фью, для отображения небольшого числа значений вполне эффективно использовать таблицу.

Дашборд состоит из следующих элементов:

- заголовок;
- текст (пояснения);
- визуализации (графики, таблицы и т.п.);
- селекторы или фильтры (элементы для взаимодействия с данными).

Также на дашборд можно добавить индикаторы – важнейшие числовые показатели на панели визуализации данных, к которым нужно привлечь повышенное внимание (итоговые суммы, наибольшие значения показателя и т.п.).

Для создания схемы и структуры дашборда необходимо определиться с целевой аудиторией, для которой он предназначен. Далее необходимо сформулировать вопросы, которые интересуют целевого пользователя, отсортировать их по приоритетности и определить, какие визуализации отвечают на поставленные вопросы.

Желательно, чтобы вся важная информация отображалась на одном экране. Если это невозможно, то нужно спроектировать последовательность дашбордов.

Для исследования данных наиболее интересны дашборды категории Аналитика & slice-n-dice (slice-n-dice – нарезка кубиками).

В приложении, основанном на дашборде Аналитика & slice-n-dice, на одном экране отображаются все необходимые данные и предоставляются интерфейсные элементы для их изучения и фильтрации. Это основа композиции приложения для аналитики и поиска интересных закономерностей.

При проектировании композиции дашборда необходимо учитывать точки притяжения человеческого внимания: первичного, вторичного и т.д.

Привычка начинать просмотр с левого верхнего угла у пользователей связана с базовым паттерном нашего поведения, сформированным благодаря правилам чтения, присутствующим во многих языках.

Однако в некоторых языках привычнее работать справа налево или сверху вниз, по тому же паттерну носители этих языков знакомятся и с информацией на экране. Поэтому композиция дашбордов с целевой аудиторией, говорящей на таком языке, должна быть скорректирована.

Так как высокий уровень взаимодействия пользователей наблюдается для элементов дашборда, расположенных слева и сверху, то именно в этой области обычно отображают панель селекторов для интерактивного взаимодействия с исходными данными.

Легкость считывания визуализации не случайна, диаграммы основаны на специальных визуальных атрибутах, подчиняющихся принципам гештальта.

Гештальт – это термин психологии, который буквально означает «единое целое» [3] и связан с теориями визуального восприятия, разработанными немецкими психологами в 20-х гг. прошлого столетия. Эти теории стремились описать, как люди склонны организовывать визуальные элементы в группы или целостные структуры, по каким принципам они их объединяют.

К важнейшим факторам (или принципам) образования гештальта относят: сходство, однородность, близость, замкнутость, смыкание, непрерывность.

Принцип сходства: похожие по одному признаку элементы воспринимаются как единое целое, даже если по другим признакам эти элементы отличаются. Основные объединяющие признаки – форма, цвет, размер.

Принцип близости: элементы, которые расположены близко друг к другу, воспринимаются как связанные и дифференцируются от элементов, которые отделены друг от друга.

Принцип замкнутости: элементы, находящиеся в одной и той же замкнутой области, воспринимаются как схожие.

Принцип непрерывности: элементы группируются как схожие, если они находятся на воображаемой линии, кривой или последовательности форм, так как человеческий глаз привык формировать маршруты и следовать по ним.

Принцип смыкания: мозг автоматически заполняет промежутки между элементами, чтобы воспринимать неполные объекты как целое изображение.

Для каждого вида сравнения данных можно подобрать наиболее подходящую визуализацию на основе перечисленных принципов [4].

Сравнение категорий

Для сравнения категорий используют столбиковые диаграммы в разных вариациях. Столбцы можно отсортировать по убыванию или возрастанию величины – это упрощает сравнение (рис. 1.1).

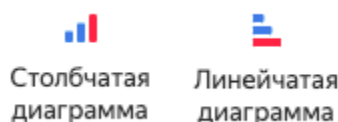


Рис. 1.1. Столбчатая и линейчатая диаграммы

Отношения между показателями

Для отображения возможных взаимосвязей между двумя показателями используется точечная (или пузырьковая) диаграмма, которая показывает наличие или отсутствие зависимости двух переменных (рис. 1.2).

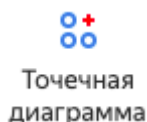


Рис. 1.2. Точечная диаграмма

Части целого

Для отображения деления целого на составные части используются круговая, древовидная и накопительные линейчатые диаграммы (рис. 1.3). Обычно сегменты показателя (его части) также сортируют от большего к меньшему, чтобы визуальное упростить считывание данных.

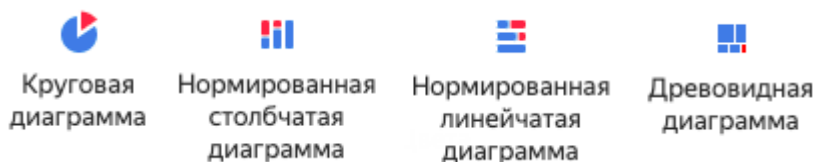


Рис. 1.3. Круговая, древовидная и накопительные линейчатые диаграммы

Распределение величины

Для отображения частотности и распределения данных в пределах определенного интервала или по выделенным группам используется столбчатая диаграмма (гистограмма). Каждая полоса на

гистограмме представляет частотность значения за определенный интервал, т.е. сколько раз то или иное наблюдение встречается в данных (рис. 1.4).



Рис. 1.4. Столбчатая диаграмма

Изменение на местности

Для отображения изменения показателя на местности используются карты, на которых откладываются точки координат или географические слои (рис. 1.5). Сам показатель может выражаться с помощью фоновой заливки элементов карты (областей) или с помощью точек разного размера (чем больше точка, тем больше показатель).



Рис. 1.5. Карта

Визуальные атрибуты, которые используются в графике, представлены на рис. 1.6 [4].

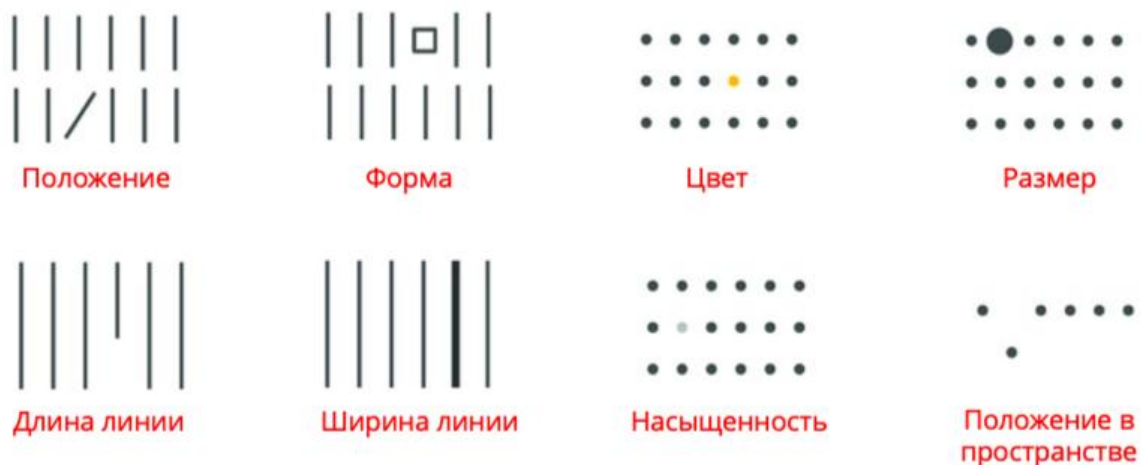


Рис. 1.6. Визуальные атрибуты

Необходимо также продумать подписи на графиках, добавить на графики заголовки и подзаголовки, легенду и т.п. Это улучшает восприятие графической информации, так как сообщает пользователю контекст.

Данные на графиках можно улучшить, добавив нормальные, ожидаемые или целевые значения, позволяющие пользователю лучше ориентироваться.

Добавление информации о выбросах, нетипичных значениях показателей позволяет ускорить время реакции на инциденты.

1.2. Microsoft Power BI для визуализации данных

Business Intelligence, или BI-системы – это набор инструментов и технологий для сбора, анализа и обработки данных.

Microsoft Power BI состоит из классического приложения для Microsoft Windows – Power BI Desktop, веб-службы SaaS (программное обеспечение как услуга), называемой Power BI service, и мобильных приложений Power BI Mobile, доступных на смартфонах и планшетах Windows, а также на устройствах Apple iOS и Google Android [5].

Общая последовательность работы в Power BI начинается с создания отчета в Power BI Desktop. Затем этот отчет публикуется в Power BI service, после чего с этими данными могут работать пользователи мобильных приложений Power BI Mobile.

В службе Power BI service (app.powerbi.com) можно зарегистрироваться с корпоративной почты (рис. 1.7).

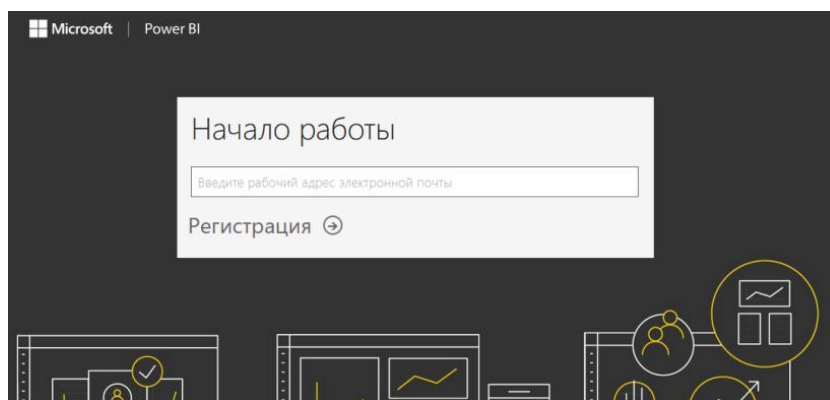


Рис. 1.7. Регистрация в Power BI service

Окно Power BI service, открытое в браузере после авторизации, имеет вид, как на рис. 1.8.

После выбора ссылки Получить данные происходит переход к диалоговому окну поиска и создания содержимого (рис. 1.9).

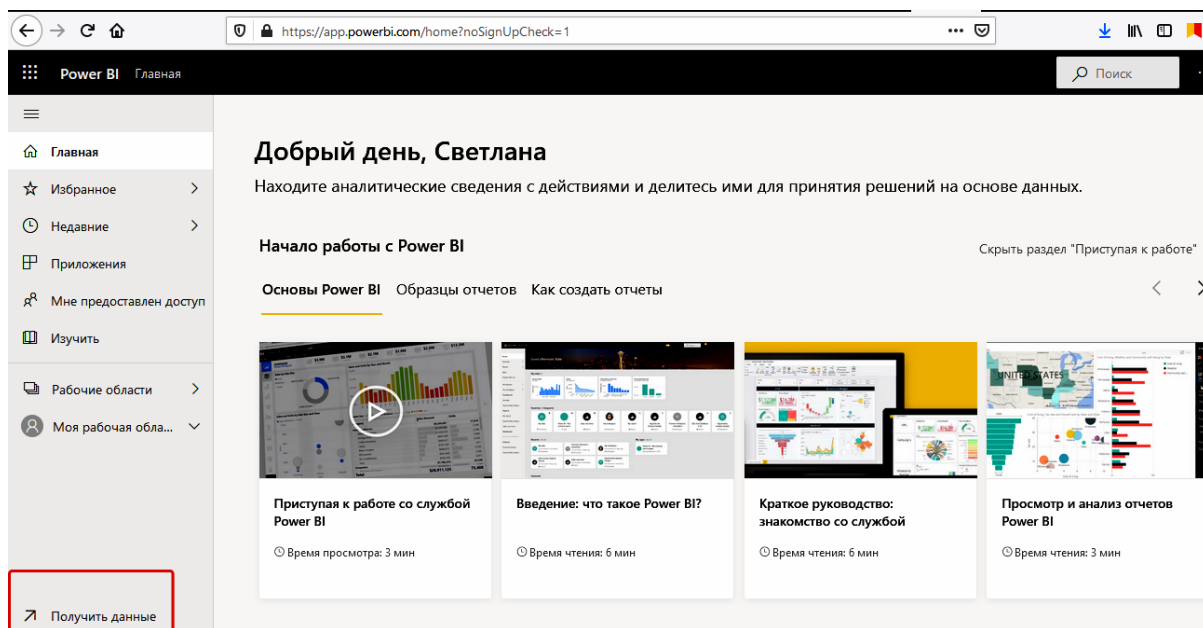


Рис. 1.8. Главная страница Power BI service

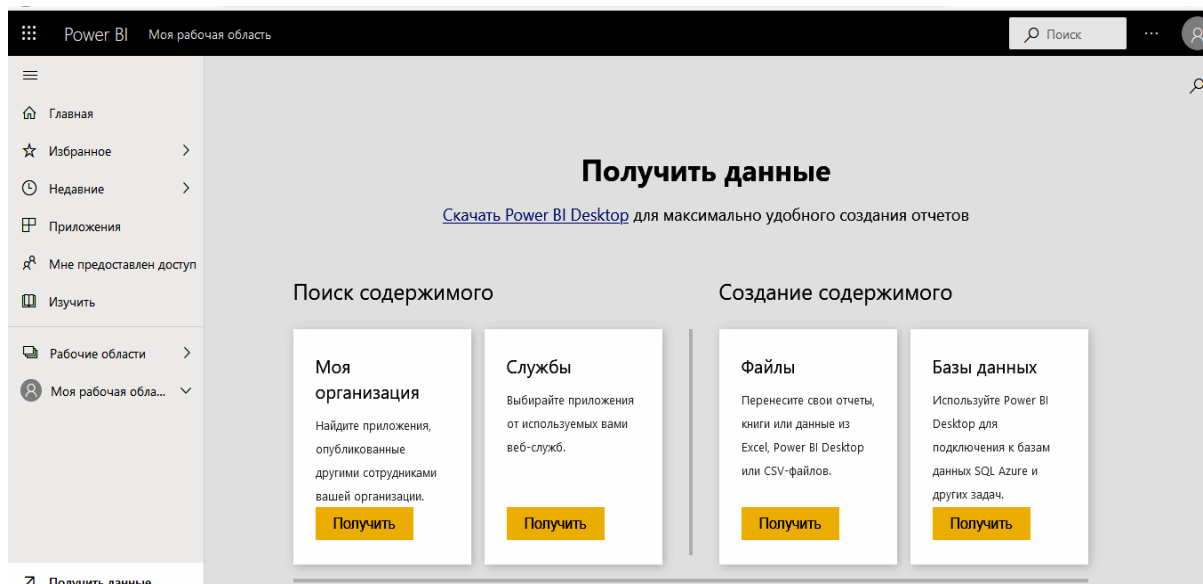


Рис. 1.9. Диалоговое окно Получить данные в Power BI service

Далее необходимо установить на компьютер пользователя приложение Power BI Desktop [6]. Окно приложения представлено на рис. 1.10.

Открытый файл безымянный. В рабочей панели открыт отчет, который в Power BI Desktop представляет собой набор визуализаций. Настройка типа визуализации, селекторов и выбор полей из таблицы данных происходят справа от рабочей области.

Так как уже была пройдена регистрация в Power BI service, то возможна авторизация и в Power BI Desktop (рис. 1.11).

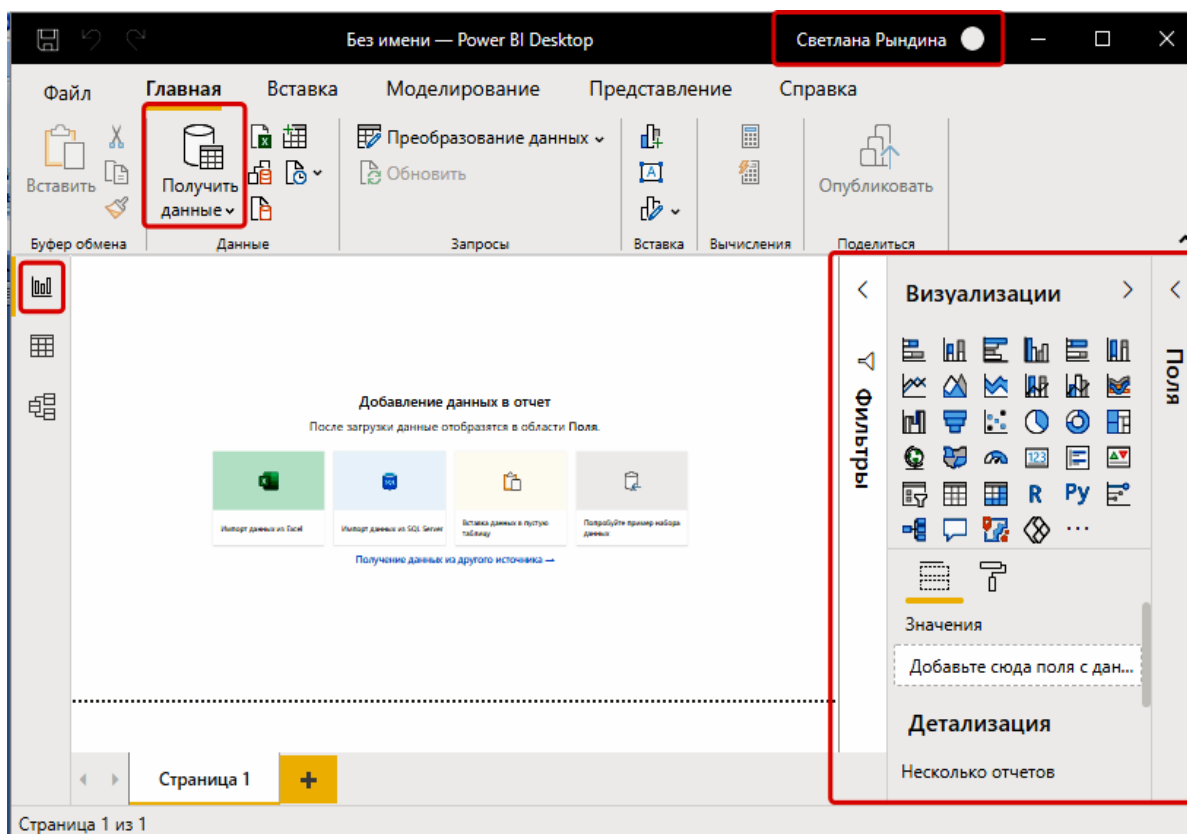


Рис. 1.10. Приложение Power BI Desktop

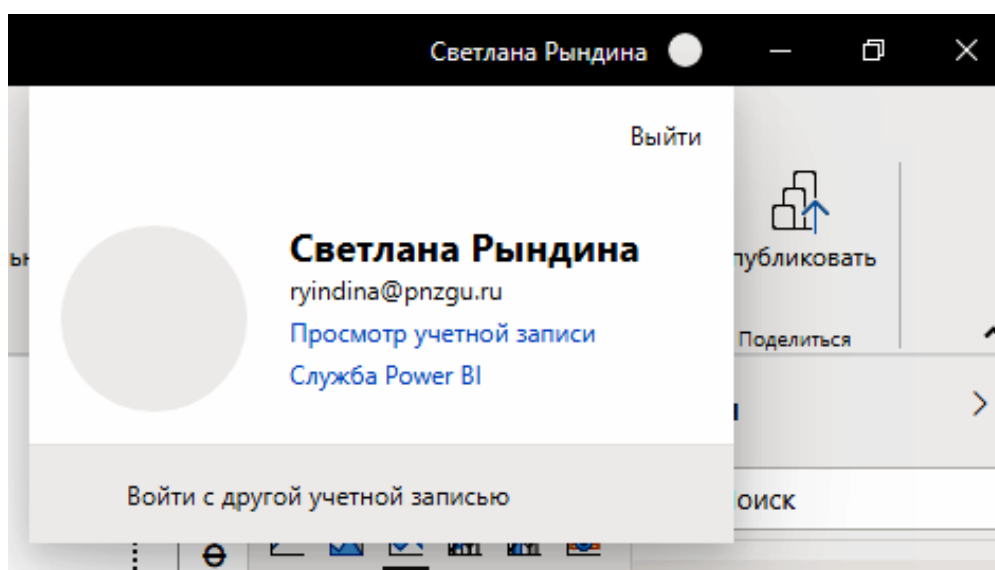


Рис. 1.11. Авторизация в Power BI Desktop с учетной записью из Power BI service

Прежде всего необходимо загрузить данные (рис. 1.12) и выбрать файл для загрузки данных (рис. 1.13).

Выберем для загрузки файл insurance.csv (скачать файл можно по ссылке: URL: <https://www.kaggle.com/mirichoi0218/insurance>).

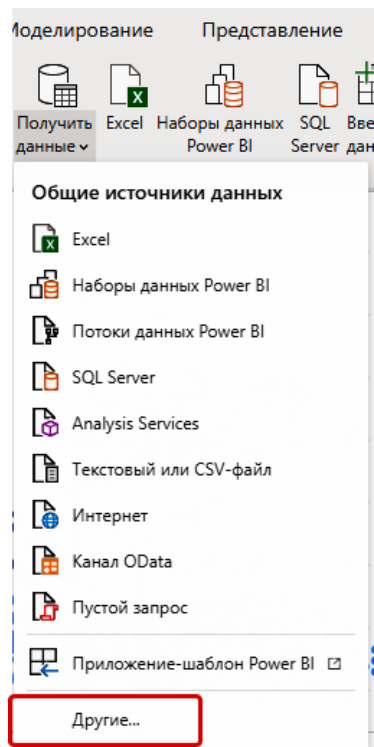


Рис. 1.12. Список пиктограммы Получить данные на панели инструментов Power BI Desktop

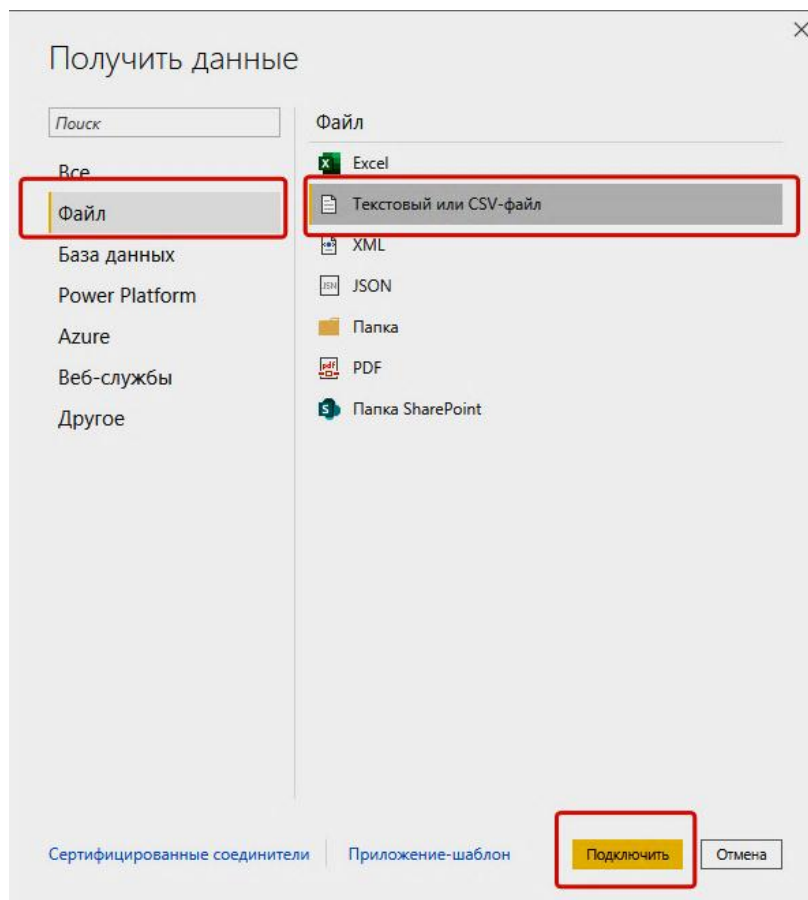


Рис. 1.13. Настройка источника данных и типа данных

Используемые данные – расходы на медицинское обслуживание тех, кто имеет медицинскую страховку:

- age: возраст основного бенефициара;
- sex: пол застрахованного;
- bmi: индекс массы тела;
- children: число детей, охваченных медицинским страхованием / число иждивенцев;
- smoker: курит ли застрахованный;
- region: жилой район получателя в США – Северо-Восток, Юго-Восток, Юго-Запад, Северо-Запад;
- charges: индивидуальные медицинские расходы, оплачиваемые страховкой.

Содержимое начальных строк файла отображается в диалоговом окне перед загрузкой (рис. 1.14). Как видим, два столбца bmi и charges определились как текстовые. Это значит, что необходимо перед загрузкой Преобразовать данные (рис. 1.15), заменив разделитель десятичных знаков с точки на запятую (рис. 1.16).

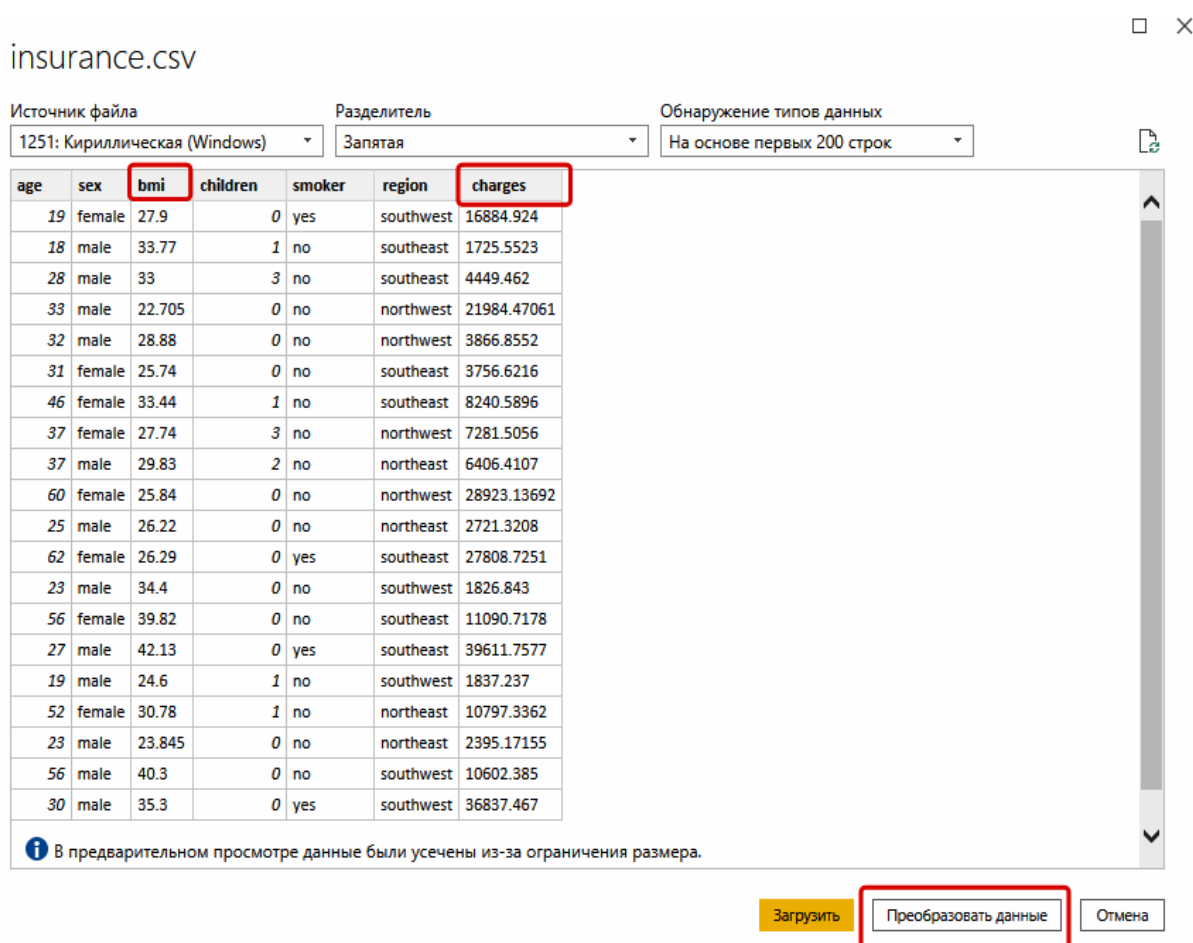


Рис. 1.14. Отображение данных из файла insurance.csv

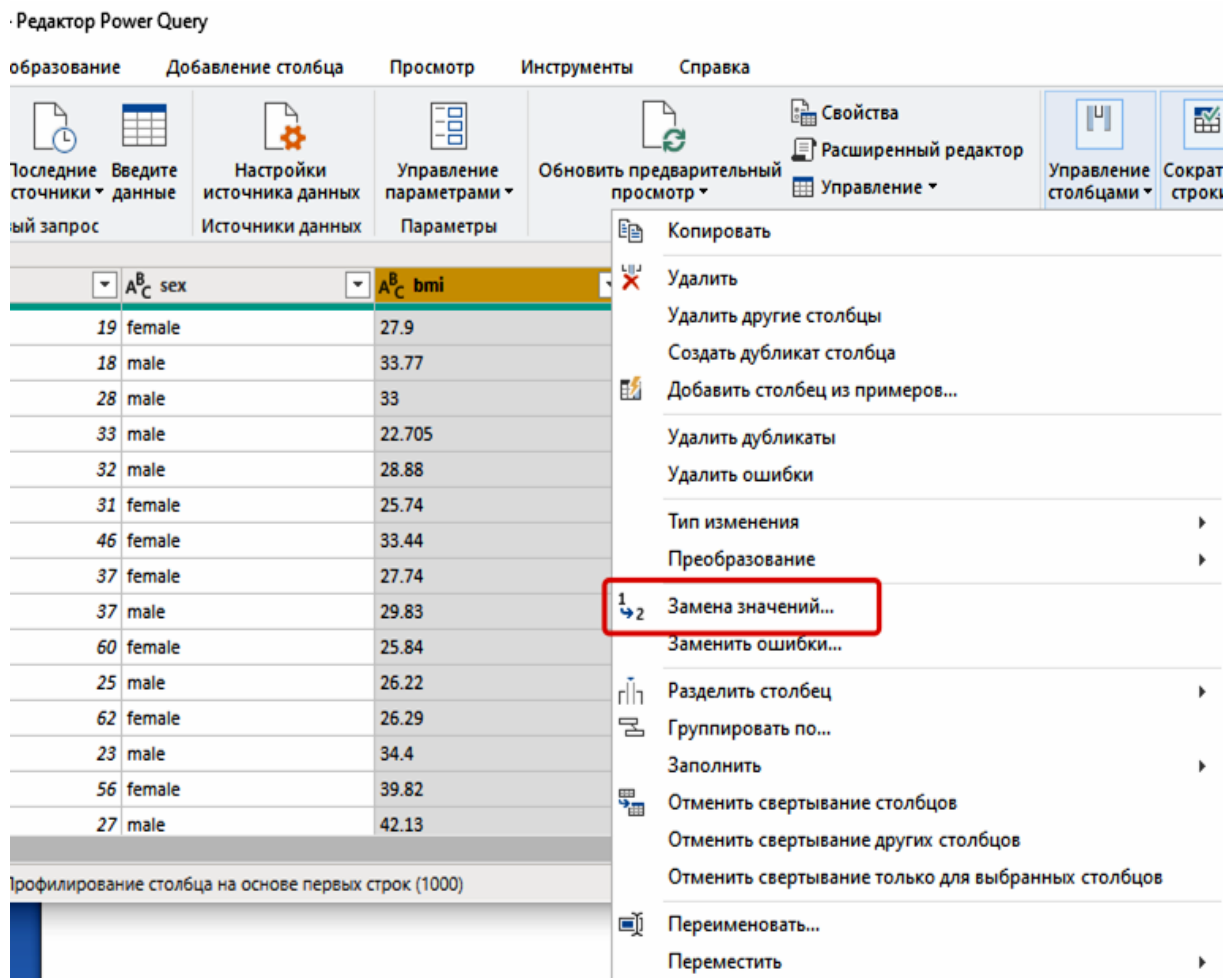


Рис. 1.15. Выбор опции Замена значений для столбца bmi

Замена значений

Заменить одно значение другим в выбранных столбцах.

Значение для поиска

Заменить на

Расширенные параметры

ОК

Отмена

Рис. 1.16. Проведение замены разделителя десятичных знаков

Сохраним проект под именем insurance_1.pbix. Откроем на просмотр таблицу в рабочем окне (рис. 1.17) и убедимся, что тип данных в каждом столбце определен верно.

insurance_1 — Power BI Desktop Светлана Рындина

Файл Главная Справка Средства работы с таблицами

Имя: insurance

Отметить как таблицу дат Управление связями Создать меру Быстрая мера Создать столбец Создать таблицу

Структура Календари Связи Вычисления

age	sex	bmi	children	smoker	region	charges
33	male	22,705	0	no	northwest	21984,47
32	male	28,88	0	no	northwest	3866,86
31	female	25,74	0	no	southeast	3756,62
60	female	25,84	0	no	northwest	28923,14
25	male	26,22	0	no	northeast	2721,32
23	male	34,4	0	no	southwest	1826,84
56	female	39,82	0	no	southeast	11090,72
23	male	23,845	0	no	northeast	2395,17
56	male	40,3	0	no	southwest	10602,39
60	female	36,005	0	no	northeast	13228,85
18	male	34,1	0	no	southeast	1137,01
63	female	23,085	0	no	northeast	14451,84
18	female	26,315	0	no	northeast	2198,19
63	male	28,31	0	no	northwest	13770,10

Поля: insurance, age, bmi, charges, children, region, sex, smoker

Таблица: insurance (строк: 1,338)

Рис. 1.17. Таблица insurance

После загрузки данных можно приступить к построению отчета. Выбрана визуализация – точечный график, данные по оси X – bmi, по оси Y – charges, условные обозначения, т.е. показатель, отвечающий за цвет точек на диаграмме, – smoker (рис. 1.18).

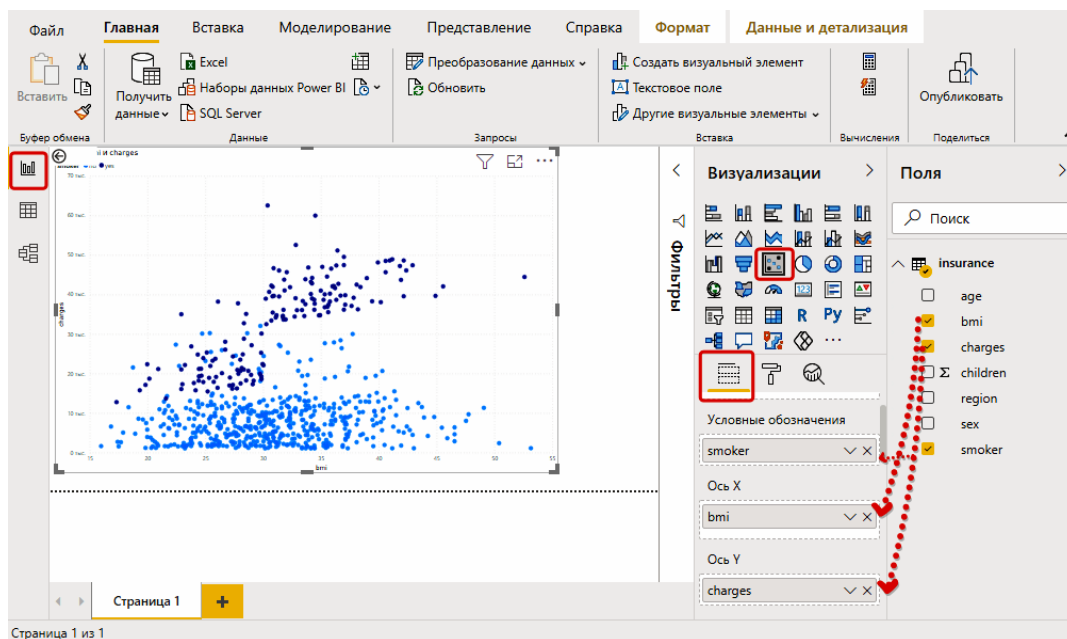


Рис. 1.18. Построение визуализации в отчете

При смещении ползунка прокрутки в окне настройки визуализации получаем доступ к настройке дополнительной опции Детализация, включив в нее показатель sex, по которому будет происходить фильтрация строк для отображения (рис. 1.19).

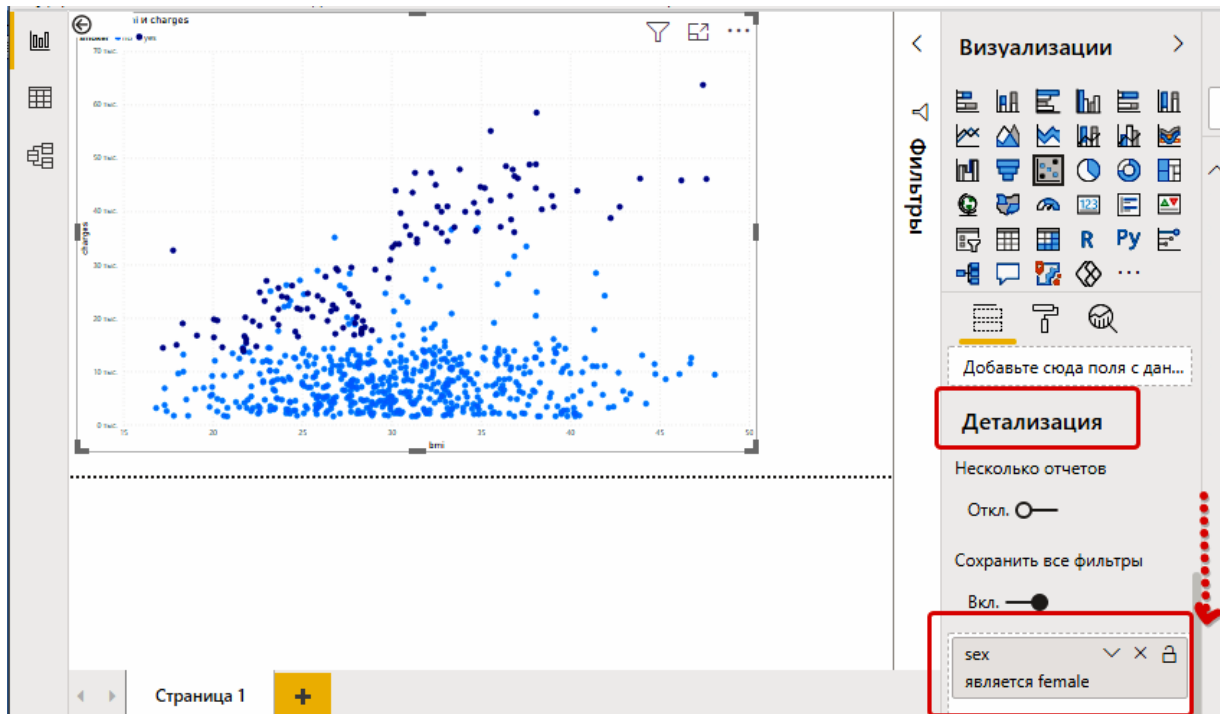


Рис. 1.19. Настройка Детализации в визуализации

Наш первый отчет готов к передаче в Power BI service. Щелкаем в командном меню на опцию Файл и выбираем Опубликовать (рис. 1.20).

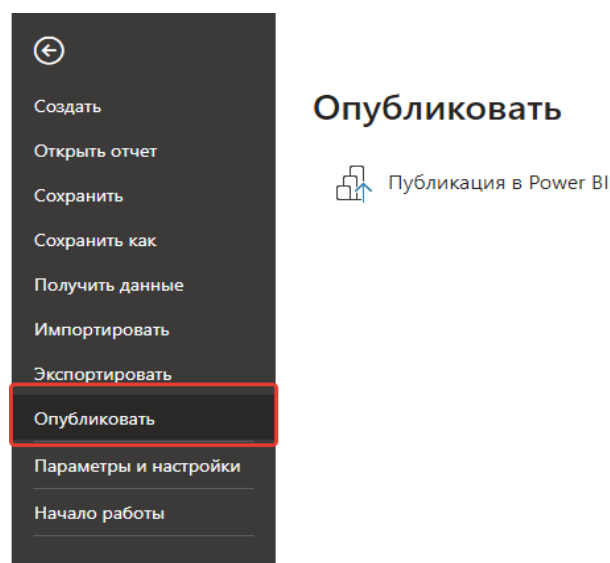


Рис. 1.20. Публикация отчета и связанного набора данных в Power BI service

Это возможно только при наличии регистрации в Power BI service и авторизации в Power BI Desktop. Наш отчет сразу передается в рабочую область Power BI service (рис. 1.21).

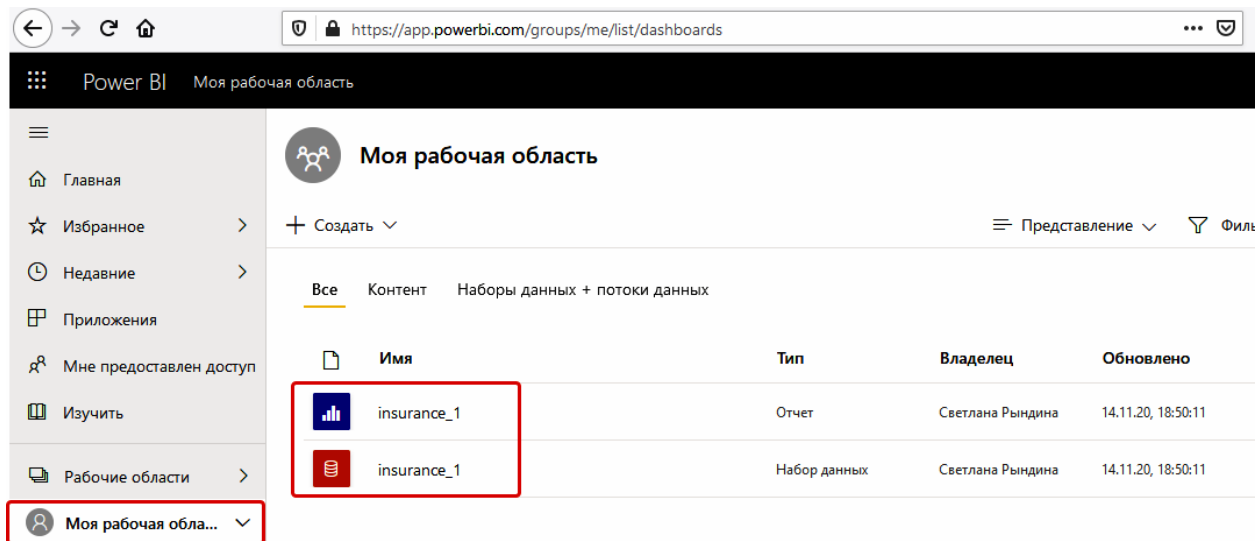


Рис. 1.21. Рабочая область Power BI service

Откроем отчет insurance_1 в Power BI service в режиме фокусировки (переход осуществляется по кнопке панели, расположенной в правом нижнем углу области визуализации – сразу под графиком). Возврат к исходному представлению доступен по кнопке Назад к отчету (рис. 1.22). Также кнопкой на панели под визуализацией можно добавить таблицу с данными, которые использовались в отчете. Справа от визуализации отображается возможность выбрать настройки фильтра по параметру sex.

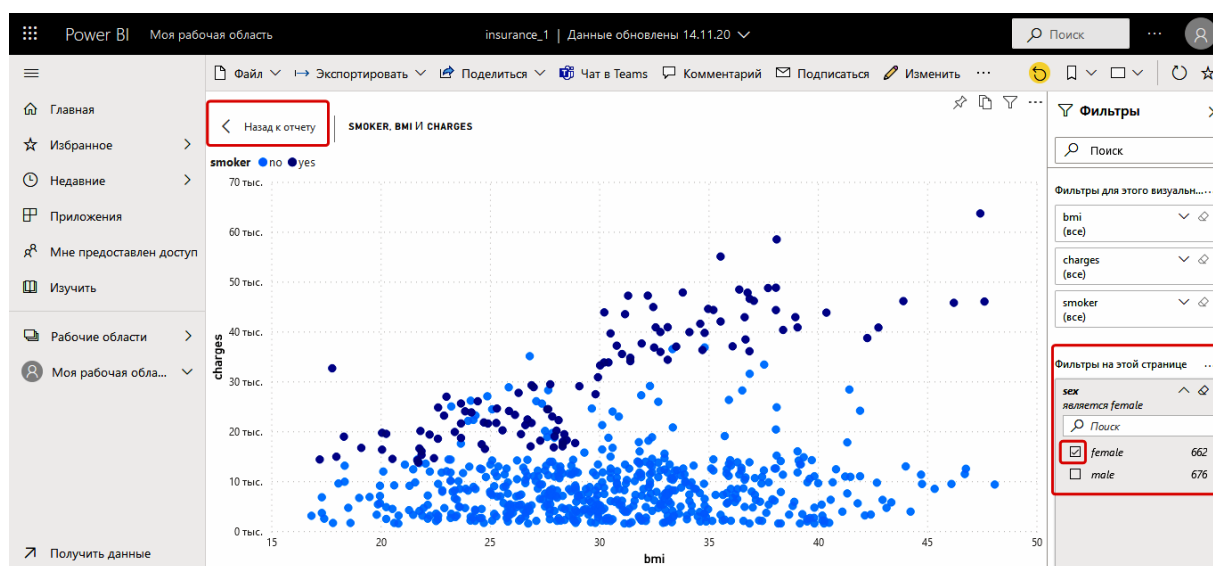


Рис. 1.22. Отчет insurance_1 в режиме фокусировки

Построим новый отчет. Загрузим в Power BI Desktop данные, которые предлагались участникам Марафона от DataYoga [2] во второй день – это файл Kinopoisk.csv.

Датасет содержит выгрузку по фильмам и сериалам, которые были выпущены с 1980 г. и относятся только к наиболее популярным жанрам (драма, комедия, мелодрама...).

В датасете, помимо жанра и страны производства, есть данные по длительности фильма или одной серии в сериале, среднему рейтингу, выставленному на Кинопоиске, и количеству предоставленных оценок:

Тип – тип видеоконтента: фильм или сериал.

Год начала – год, в котором выпущен фильм или начат показ сериала.

Жанр – основной жанр.

Жанр 2, Жанр 3 – дополнительные жанры.

Страна производства – основной производитель фильма.

Название – название фильма или сериала.

Количество голосов – сколько зарегистрированных пользователей Кинопоиска проголосовало за фильм.

Длительность – продолжительность фильма или сериала в минутах.

Средний рейтинг – рейтинг картины на площадке Кинопоиск.

Год окончания – год, в который закончился показ сериала (поле, пустое для фильмов).

Начало таблицы представлено на рис. 1.23.

Год начала	Тип	Жанр	Жанр 2	Жанр 3	Страна производства	Название	Количество голосов	Длительность (мин)	Средний рейтинг
2000	сериал	боевик	триллер		США	Секретные агенты	77	43.0	6.579
2012	фильм	комедия	спорт		США	Толстяк на ринге	33511	105.0	6.955999999999995
2008	сериал	драма	мелодрама	комедия	США	Бeverly-Хиллз 90210: Новое поколение	8704	43.0	7.137000000000005
2008	фильм	боевик	детектив	приключения	Германия	В поисках сокровищ нибелунгов	135	120.0	6.148
1987	фильм	мультфильм	короткометражка		СССР	Белая арена	58	9.0	6.22
2008	фильм	комедия			США	Знакомство со спартацами	30719	87.0	3.909
2006	фильм	ужасы	триллер	драма	Колумбия	В конце спектра	207	92.0	4.830999999999995
2003	фильм	мультфильм	фантастика	драма	США	Ноль один	65	85.0	3.925
2012	фильм	драма	комедия	история	Великобритания	Гайд-Парк на Гудзоне	2331	94.0	5.684
2010	фильм	документальный			Германия	Зеленая волна	75	80.0	5.674

Рис. 1.23. Таблица файла Kinopoisk.csv

Создадим несколько визуализаций, использующих агрегированные показатели для данных датасета.

В Power BI Desktop на первой странице создадим три визуализации (рис. 1.24).

Для первой активной визуализации выберем режим фокусировки, чтобы подробнее рассмотреть параметры ее настройки (рис. 1.25).

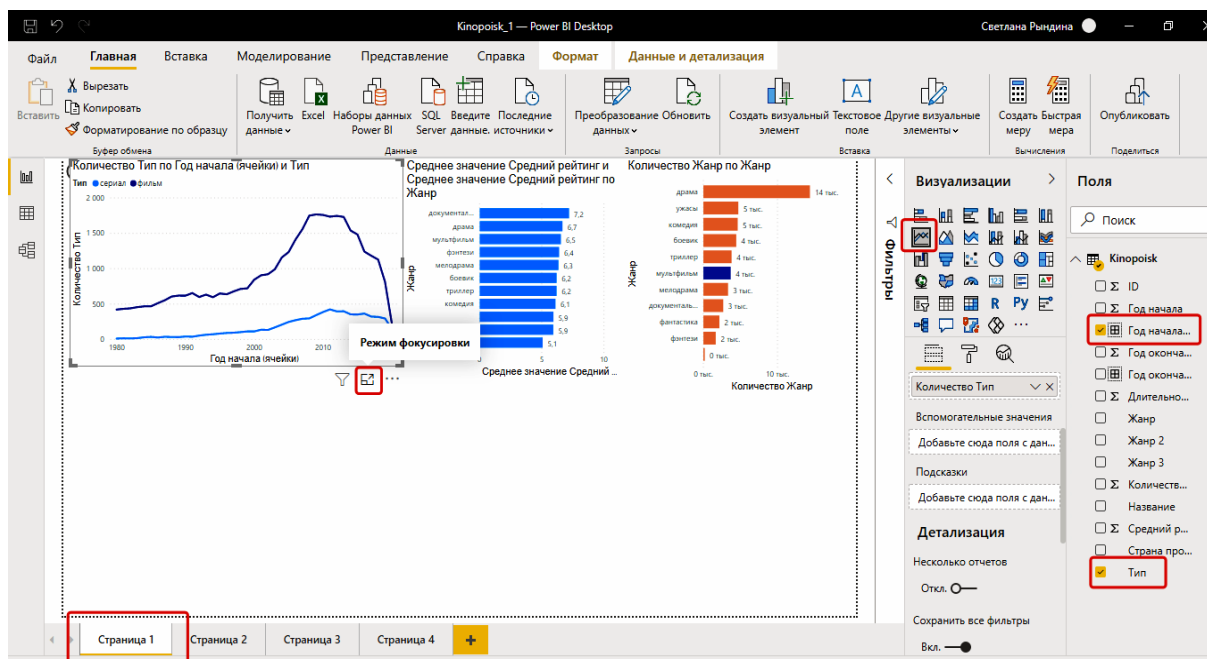


Рис. 1.24. Визуализации для данных Kinopoisk

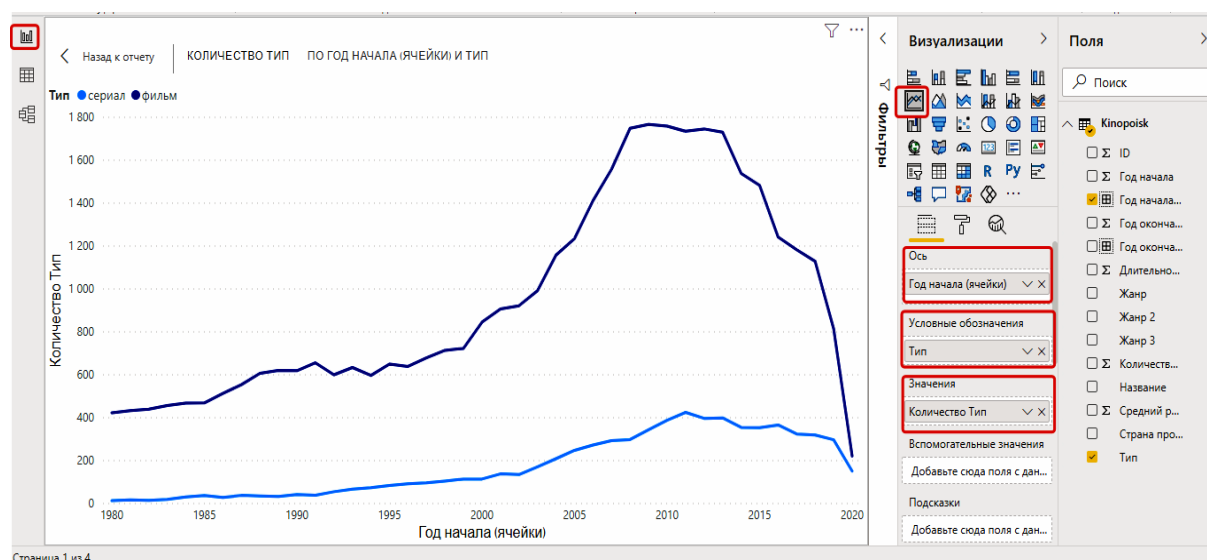


Рис. 1.25. Визуализация количества фильмов и сериалов, выпущенных на экраны по годам начала выпуска

Для второй визуализации также выберем режим фокусировки (рис. 1.26). В этой визуализации дополнительно на вкладке **Формат** в панели **Визуализация** было настроено отображение значений среднего для среднего рейтинга по жанрам.

Для третьей визуализации дополнительно на вкладке **Формат** в панели **Визуализация** были настроены позиции **Цвет** и **Метки** для отображения значений количества фильмов и сериалов в разных жанрах (рис. 1.27).

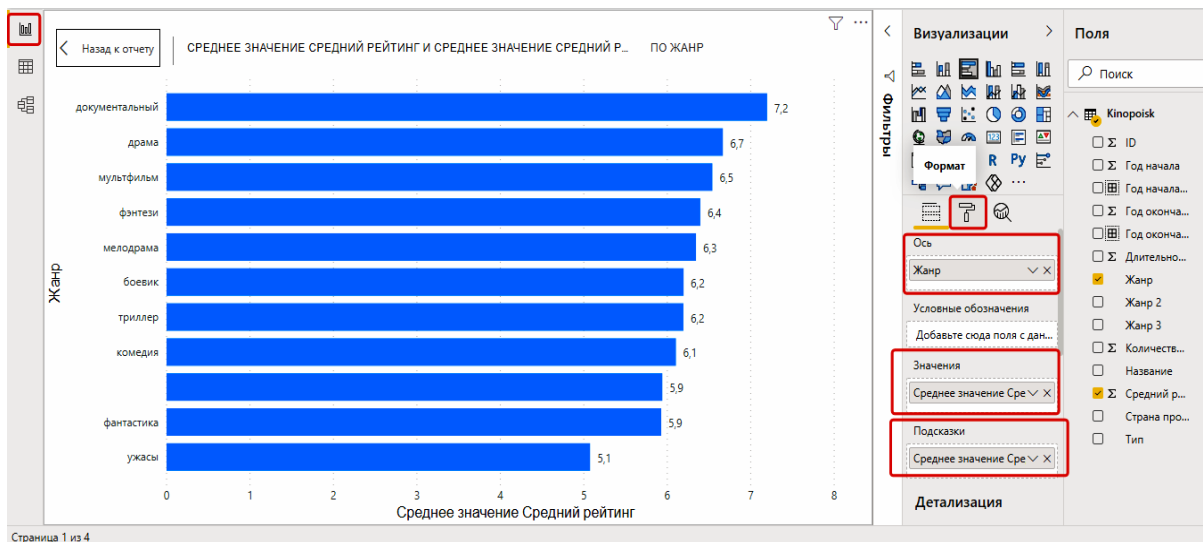


Рис. 1.26. Визуализация среднего значения для среднего рейтинга фильмов и сериалов

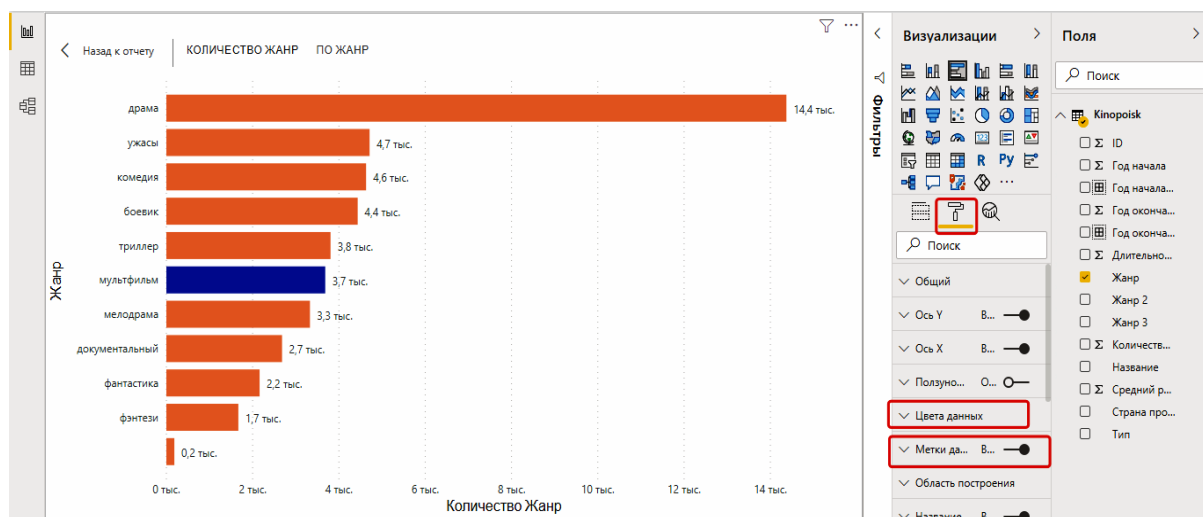


Рис. 1.27. Визуализация количества фильмов и сериалов в разных жанрах

На следующих страницах файла Kinopoisk_1.pbix повторим дашборд Романа Бунина – руководителя команды визуализации данных Yandex.Go [2] – и опубликуем новый файл Kinopoisk_1.pbix в Power BI service.

На рис. 1.28 представлен отчет Kinopoisk_1 в Power BI service с отображением на экране первой страницы отчета.

Для удобства свернем боковую панель меню и навигацию по страницам (рис. 1.29). Этот вариант отображает дашборд в полноэкранный режим, скрывает часть интерфейсных элементов и увеличивает область отображения визуализации на экране.

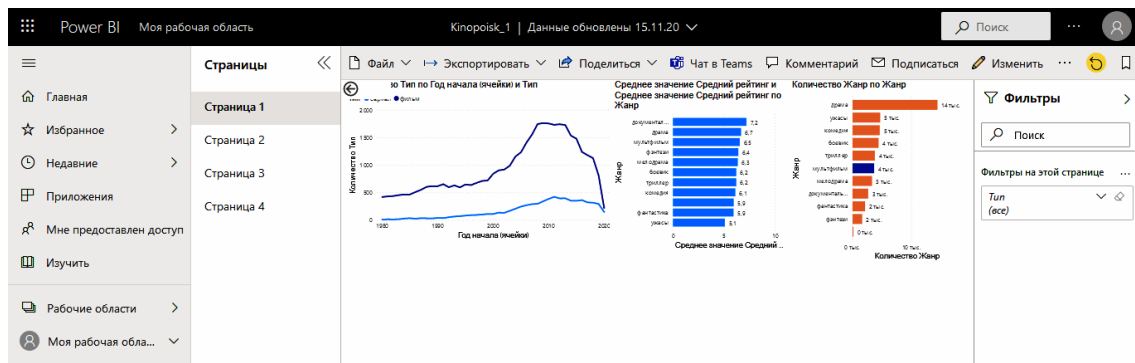


Рис. 1.28. Отчет Kinopoisk_1

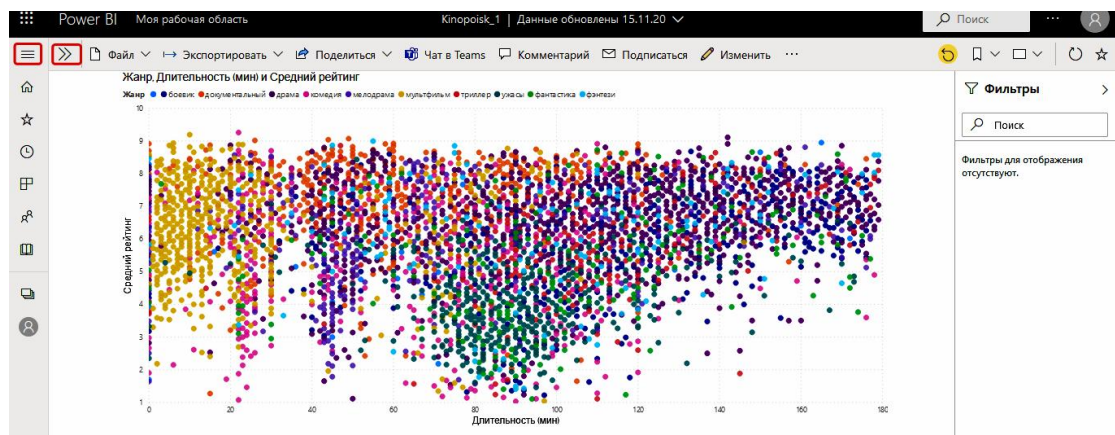


Рис. 1.29. Визуализация зависимости среднего рейтинга от длительности фильма/сериала (одной серии) с учетом жанра

На следующей визуализации отчета демонстрируется, как перемещение в области графика по оси X позволяет получить информационную сводку по отображаемым в этой точке данным (рис. 1.30).

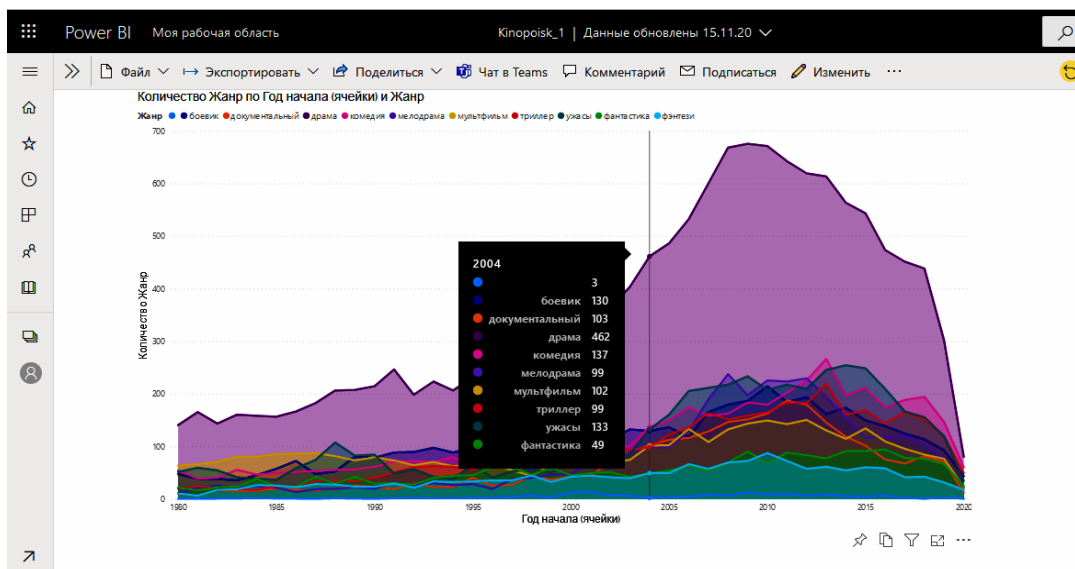


Рис. 1.30. Визуализация количества снятых фильмов и сериалов по жанрам во времени

Для визуализации изменения среднего рейтинга фильмов и сериалов во времени с учетом жанра использовались настройки, позволяющие отобразить только самую популярную видеопroduкцию: фильтр по количеству голосов настроен на значение больше 1000 (рис. 1.31).

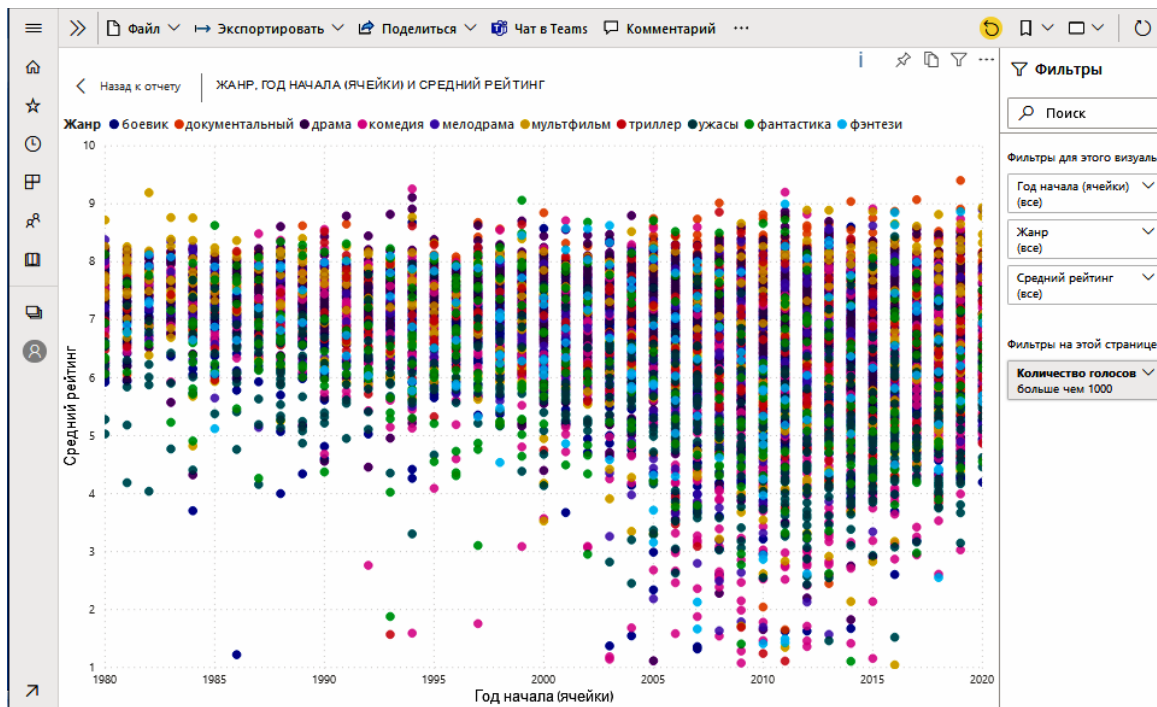


Рис. 1.31. Визуализация среднего рейтинга фильмов и сериалов по жанрам во времени с селектором: максимально популярные

Так как в Power BI service происходила интенсивная работа, то главное окно теперь примет вид, как показано на рис. 1.32.

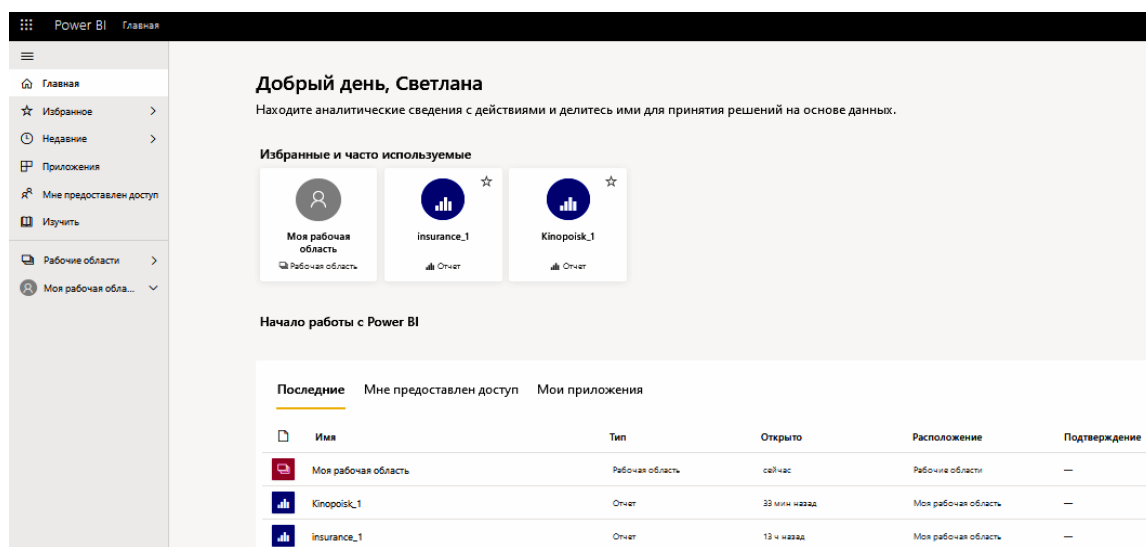


Рис. 1.32. Главная страница Power BI service

К сожалению, многие важные возможности по совместной работе и дополнительные инструменты визуализации доступны только в версии Power BI Pro.

На представленном в пособии инструментарии возможности бесплатной версии Power BI не заканчиваются. Для визуализации данных в Power BI Desktop реализована поддержка скриптов на языках R и Python, разработке веб-приложений на которых посвящено дальнейшее содержание пособия.

Изучение возможностей платформы аналитики данных Power BI можно продолжить самостоятельно [7].

2. СОЗДАНИЕ ВЕБ-ПРИЛОЖЕНИЙ НА ЯЗЫКЕ R С ПОМОЩЬЮ СРЕДЫ РАЗРАБОТКИ RSTUDIO

2.1. Пакет shiny

Наука о данных и веб-разработка объединяются благодаря пакету shiny. Организуем взаимодействие мира данных и Интернета: совместим разработанные алгоритмы анализа данных с их размещением в веб-приложениях.

Для создания веб-приложений на основе данных на языке R используется пакет shiny [8–10]. Подключить этот пакет можно в консоли RStudio:

```
install.packages("shiny")
```

Создадим веб-приложение, воспользовавшись командным меню в RStudio (рис. 2.1) и выбрав в опции File – New file – Shiny Web App...

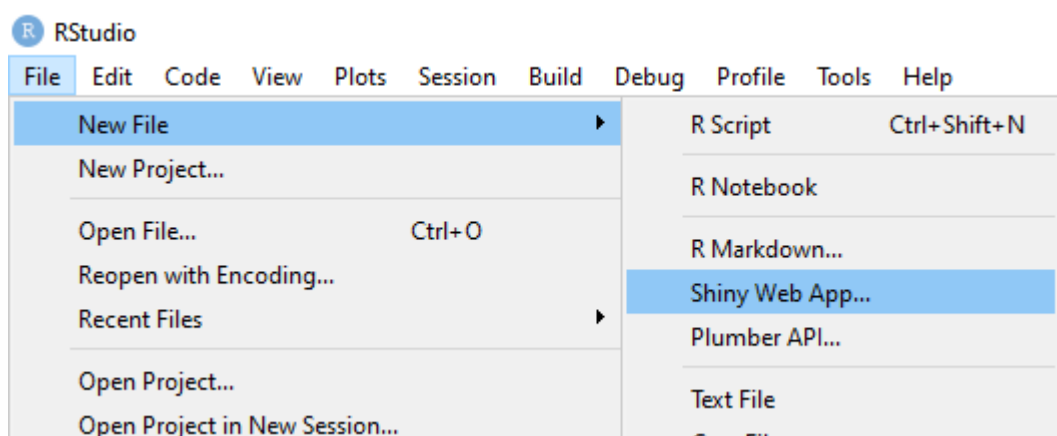


Рис. 2.1. Создание Shiny Web App

Выберем вариант с двухкомпонентным приложением (рис. 2.2) и дадим ему имя. В папке с именем приложения создаются два файла: скрипт с интерфейсом пользователя (user interface, UI) – файл ui.R, который управляет внешним видом приложения и представляет собой клиентскую часть, и скрипт для сервера – файл server.R, управляющий процессом создания приложения и его поведением.

В пользовательском скрипте настраиваются элементы интерфейса (кнопки, поясняющий текст и т.п.). Скрипт для сервера определяет сценарии его работы: методы манипулирования данными, логику работы приложения.

Опция **Single File (app.R)** объединяет сценарии ui.R и server.R в один (см. рис. 2.2). Разделение этих двух составляющих приложения по разным скриптам позволяет не перегружать код.

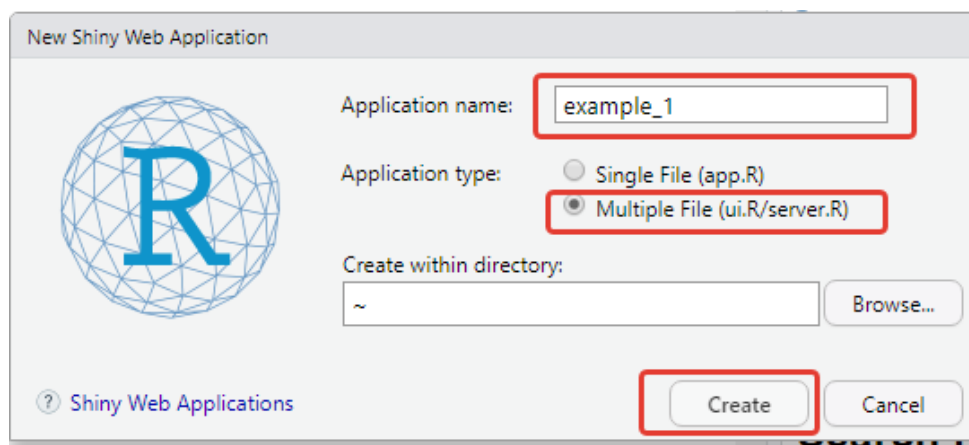


Рис. 2.2. Создание двухкомпонентного приложения с именем example_1

Рассмотрим основные составляющие ui.R на дефолтном примере, который генерируется по умолчанию при создании файла (пояснения переведены на русский, в оригинале комментарии создаются на английском):

```
library(shiny)
# Определение пользовательского интерфейса для приложения,
# которое рисует гистограмму
shinyUI(fluidPage(
  # Заголовок приложения
  titlePanel("Old Faithful Geyser Data"),
  # Боковая панель с ползунком ввода количества баров
  # (числа интервалов) для построения гистограммы
  sidebarLayout(
    sidebarPanel(
      sliderInput("bins",
        "Number of bins:",
        min = 1,
        max = 50,
        value = 30)
    ),
    # Вывод графика сгенерированного распределения
    mainPanel(
      plotOutput("distPlot")
    )
  )
))
```

Содержимое на панели заголовков (Title Panel) отображается как метаданные, обычно это имя приложения и другая поясняющая информация.

Боковая панель (Sidebar Layout) принимает данные от пользователя в различных формах, таких как ввод текста, установка флажка, переключение радиокнопки, выпадающий список и т.д.

Все элементы интерфейса пользователя обернуты в функцию `fluidPage()`, которая отвечает за «жидкий макет» страницы с внедренными в него элементами.

Основная панель (Main Panel) предоставляет вывод результата выполнения набора операций из файла `server.R`.

Файл `server.R`, который генерируется по умолчанию при создании приложения:

```
library(shiny)
# Определение логики сервера,
# необходимой для построения гистограммы
shinyServer(function(input, output) {
  output$distPlot <- renderPlot({
    # создать бары на основе переменной input$bins из файла ui.R
    x <- faithful[, 2]
    bins <- seq(min(x), max(x), length.out = input$bins + 1)
    # отрисовка гистограммы с заданным количеством баров
    hist(x, breaks = bins, col = 'darkgray', border = 'white')
  })
})
```

Запустить созданное приложение с именем `example_1` можно:

- функцией `runApp("application_name")`, которую нужно запустить на выполнение в консоли;
- кнопкой **Run**, которая расположена в верхнем правом углу редактора, при открытом файле приложения (рис. 2.3).

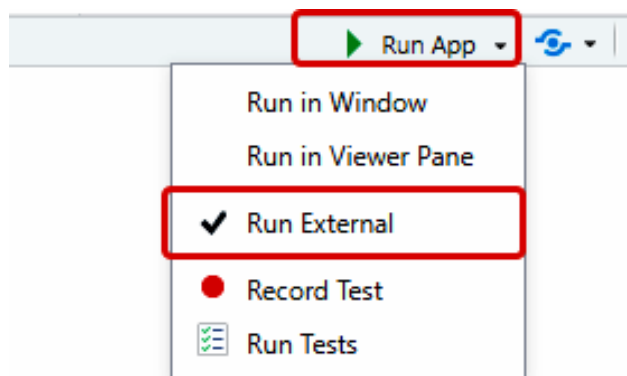


Рис. 2.3. Запуск приложения

При выборе опции **Run External** откроется окно веб-браузера (рис. 2.4) со страницей созданного приложения.

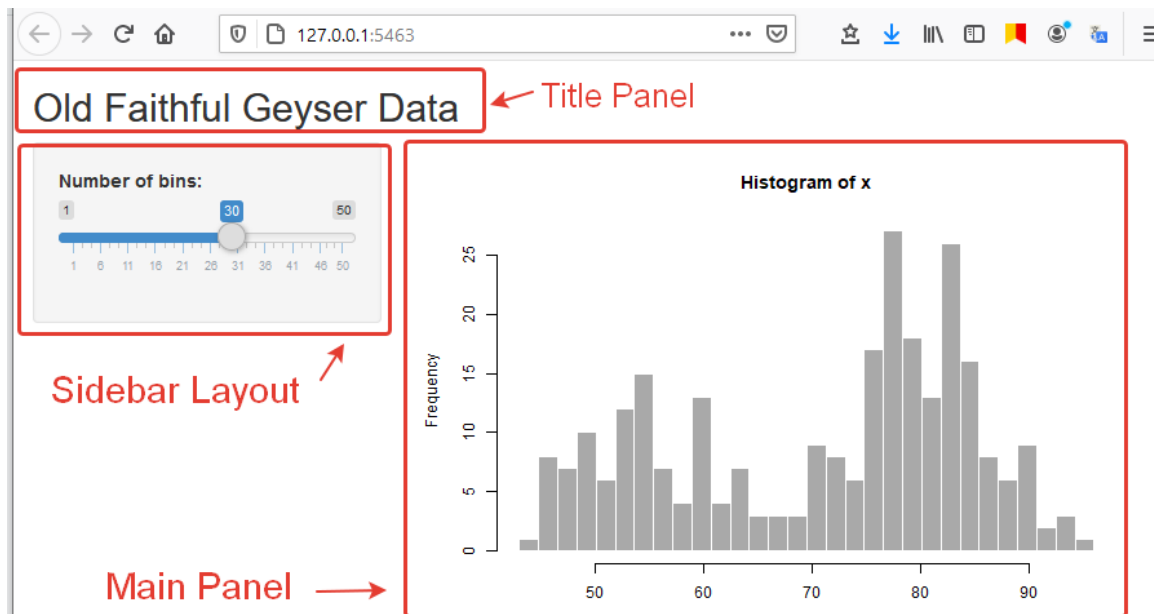


Рис. 2.4. Вид приложения в браузере

Чтобы закрыть приложение, нужно нажать Esc (Stop на консоли).

2.2. Управление выводом таблицы на экран, виджеты

Изменим файлы созданного приложения `example_1`, чтобы отображалось содержимое таблицы данных из файла. В папке `example_1` имеются два скрипта `ui.R` и `server.R` (с дефолтным содержанием кода), там же сохраним файл данных `insurance.csv`.

Изменим дефолтный код, чтобы приложение могло отобразить таблицу из файла `insurance.csv`:

ui.R

```
insurance<-read.table('insurance.csv', sep=',', header=T)
shinyUI(fluidPage(
  titlePanel("Insurance"),
  mainPanel(tableOutput("data"))))
```

server.R

```
shinyServer(function(input, output) {
  output$data = renderTable({insurance})
})
```

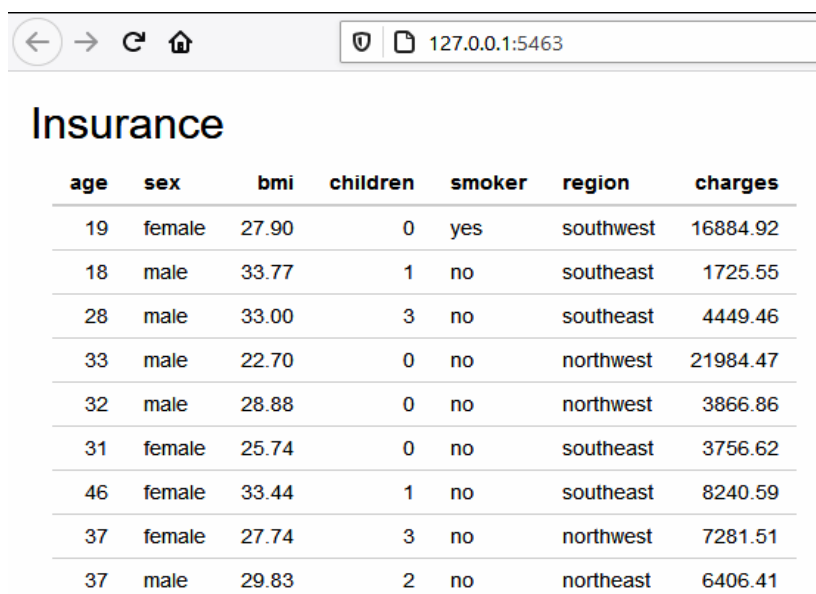
Хотя переменная `insurance` используется в скрипте сервера, чтение в переменную данных из файла происходит в скрипте клиентской части. Описание используемых функций приведем в табл. 2.1.

Функции для отображения таблицы

Скрипт	Функция	Назначение	Аргумент	Назначение
ui.R	tableOutput (outputId)	Создает выходной элемент: таблицу	outputId	Выходная переменная для чтения таблицы
server.R	renderTable()	Создает реактивную таблицу, назначаемую переменной вывода: output\$outputId	{ dataframe }	Имя переменной типа dataframe с данными таблицы
			hover = FALSE, bordered = FALSE, width = "auto", align = NULL, rownames = FALSE, colnames = TRUE, digits = NULL, na = "NA" ...	Аргументы, определяющие параметры вывода таблицы: выравнивание текста, отображение имен столбцов и строк и т.п.

В примере outputId = "data".

В браузере страница веб-приложения примет вид, как показано на рис. 2.5.



age	sex	bmi	children	smoker	region	charges
19	female	27.90	0	yes	southwest	16884.92
18	male	33.77	1	no	southeast	1725.55
28	male	33.00	3	no	southeast	4449.46
33	male	22.70	0	no	northwest	21984.47
32	male	28.88	0	no	northwest	3866.86
31	female	25.74	0	no	southeast	3756.62
46	female	33.44	1	no	southeast	8240.59
37	female	27.74	3	no	northwest	7281.51
37	male	29.83	2	no	northeast	6406.41

Рис. 2.5. Вывод таблицы из файла на страницу приложения

Добавим таблице реактивности, т.е. реализуем возможность динамически менять содержимое страницы в ответ на действия пользователей (поменяли что-то во фрагменте клиентской части – мгновенно изменения отразились на всех связанных местах без перезагрузки страниц).

Дополним боковую панель пользовательского интерфейса виджетами (widgets). Они используются в клиентской части приложения для передачи значений входных данных в серверную функцию, т.е. для управления выводом серверной функции на основании выбора пользователя в интерфейсе приложения. Список виджетов представлен в табл. 2.2.

Таблица 2.2

Виджеты для внедрения в ui.R

Функция	Назначение	Аргумент	Назначение
1	2	3	4
Slider Input()	Создает виджет слайдера для выбора числового значения из диапазона	inputId	Входная переменная, которая будет использоваться для доступа к введенному значению
		label	Метка, отображаемая на экране рядом с вводимым значением (необязательный аргумент)
		min	Минимальное значение, которое может быть выбрано
		max	Максимальное значение, которое может быть выбрано
		value	Начальное значение ползунка
		step = NULL, round = FALSE, format = NULL, locale = NULL, ticks = TRUE, animate = FALSE, width = NULL, sep = ", "	Параметры, которые определяют вид слайдера (количество делений на нем и т.п.)

1	2	3	4
numeric Input()	Создает виджет слайдера для выбора числового значения из диапазона	inputId	Входная переменная, которая будет использоваться для доступа к введенному значению
		label	Метка, отображаемая на экране рядом с вводимым значением (необязательный аргумент)
		min	Минимальное значение, которое может быть выбрано (необязательный аргумент)
		max	Максимальное значение, которое может быть выбрано (необязательный аргумент)
		value	Начальное значение (необязательный аргумент)
		step = NA, width = NULL	Параметры, которые определяют шаг для отображения значений в линейке выбора и ширину этой линейки
selectInput()	Создает список выбора, который можно использовать для выбора одного или нескольких элементов из списка значений	inputId	Входная переменная, которая будет использоваться для доступа к введенному значению
		label	Метка, отображаемая на экране рядом с вводимым значением (необязательный аргумент)
		choices	Список значений для выбора. Если элементы списка имеют имена, то пользователю отображается это имя, а не значение
			Кроме того, можно сгруппировать связанные входные данные, предоставив именованный список, элементами которого являются (именованные или неназванные) списки, векторы или факторы
		selected = NULL, multiple = FALSE, selectize = TRUE, width = NULL, size = NULL	Параметры отображения списка выбора. Например, selected задает первоначально выбранное значение (или несколько значений, если multiple = TRUE). Если не указано, то по умолчанию используется первое значение для списков с одним выбором и никаких значений для списков с несколькими вариантами выбора
		inputId	Входная переменная, которая будет использоваться для доступа к введенному значению

1	2	3	4
radio Buttons()	Создает набор переключателей, используемых для выбора элемента из списка	label	Метка, отображаемая на экране рядом с вводимым значением (необязательный аргумент)
		choices	Список значений для выбора (если элементы списка имеют имена, то пользователю отображается это имя, а не значение). Если этот аргумент указан, то choiceNames и choiceValues не должны быть указаны, и наоборот. Значения должны быть строками; другие типы (например, логические и числовые) будут принудительно преобразованы в строки (необязательный аргумент)
		selected = NULL, inline = FALSE, width = NULL, choiceNames = NULL, choiceValues = NULL	Параметры отображения списка выбора
		inputId	Входная переменная, которая будет использоваться для доступа к введенному значению
checkbox Input()	Создает флаг для ввода логических значений	label	Метка, отображаемая на экране рядом с вводимым значением (необязательный аргумент)
		value = FALSE,	Начальное значение (TRUE or FALSE) (необязательный аргумент)
		width = NULL	Параметры отображения списка выбора
		inputId	Входная переменная, которая будет использоваться для доступа к введенному значению
file Input()	Создает элемент управления загрузкой файлов, который можно использовать для загрузки одного или нескольких файлов	label	Метка, отображаемая на экране рядом с вводимым значением (необязательный аргумент)
		multiple = FALSE, accept = NULL, width = NULL, buttonLabel = "Browse...", placeholder = "No file selected"	Отображение дополнительной информации о загрузке файла

1	2	3	4
		inputId	Входная переменная, которая будет использоваться для доступа к введенному значению
date Input()	Создает поле для ввода, при нажатии на которое появляется календарь	label	Метка, отображаемая на экране рядом с вводимым значением (необязательный аргумент)
		value = NULL, min = NULL, max = NULL, format = "yyyy-mm-dd", startview = "month", weekstart =0, language = "en", width = NULL, ...	Параметры отображения даты

Создадим реактивное веб-приложение, в котором с помощью внедренных виджетов на боковой панели пользователь будет управлять параметрами отображения данных в таблице.

Кроме внедрения виджетов в файл `ui.R`, изменения должны быть внесены и в скрипт `server.R` (на сервере введенные значения будут отфильтровывать содержимое таблицы для отображения на веб-странице).

Внедряем виджеты на боковую панель и используем введенные значения для фильтрации данных исходной таблицы, изменяя код на следующий:

ui.R

```
insurance<-read.table('insurance.csv', sep=',', header=T)
shinyUI(fluidPage(
  titlePanel("Insurance"),
  sidebarLayout(sidebarPanel(
    numericInput(inputId = "age1",
      label = "Age>",
      min = 18,
```

```

        max = 90,
        value = 18
      ),
      numericInput(inputId = "age2",
        label = "Age<",
        min = 18,
        max = 67,
        value = 65
      ),
      selectInput(inputId = "smoker",
        label = "Smoker:",
        choices = c("all", unique(as.character(df_film$smoker)))
      ),
      radioButtons(inputId = "sex",
        label = "Sex:",
        choices = list(
          "all" = "all",
          "female" = "female",
          "male" = "male"
        ),
        selected = "all"
      ),
      selectInput(inputId = "region",
        label = "Region:",
        choices = c("all", unique(as.character(insurance$region)))
      ),
      selectInput(inputId = "children",
        label = "Children:",
        choices = c("all", unique(as.character(insurance$children)))
      ),
      sliderInput(inputId = "bmi",
        label = "Bmi",
        min = min(insurance$bmi),
        max = max(insurance$bmi),
        value = c(20,25)
      ),
    ),
    mainPanel(tableOutput("data")))))

```

server.R

```

shinyServer(function(input, output) {
  output$data = renderTable({
    data <- insurance
    data <- data[(data$age > input$age1) & (data$age < input$age2),]
    if (input$sex != "all") {
      data <- data[data$sex == input$sex,]
    }
    if (input$children != "all") {
      data <- data[data$children == input$children,]
    }
  })

```

```

    }
    if (input$smoker != "all") {
      data <- data[data$smoker == input$smoker,]
    }
    if (input$region != "all") {
      data <- data[data$region == input$region,]
    }
    data <- data[(data$bmi >= min(input$bmi))&(data$bmi <= max(input$bmi)),]
    data
  })
})

```

В браузере веб-страница примет вид, как показано на рис. 2.6.

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5463'. The page title is 'Insurance'. The interface features several filters on the left side: 'Age' with a range from 18 to 65, 'Smoker' set to 'all', 'Sex' with radio buttons for 'all', 'female', and 'male', 'Region' set to 'all', 'Children' set to 'all', and 'Bmi' with a slider ranging from 15.96 to 53.13. On the right side, a table displays the filtered data.

age	sex	bmi	children	smoker	region	charges
33	male	22.70	0	no	northwest	21984.47
19	male	24.60	1	no	southwest	1837.24
23	male	23.84	0	no	northeast	2395.17
63	female	23.09	0	no	northeast	14451.84
19	male	20.43	0	no	northwest	1625.43
26	male	20.80	0	no	southwest	2302.30
41	male	21.78	1	no	southeast	6272.48
60	female	24.53	0	no	southeast	12629.90
53	female	22.88	1	yes	southeast	23244.79
64	male	24.70	1	no	northwest	30166.62
20	female	22.42	0	yes	northwest	14711.74
28	male	23.98	3	yes	southeast	17663.14
27	female	24.75	0	yes	southeast	16577.78
45	male	22.89	2	yes	northwest	21098.55
53	female	24.80	1	no	northwest	10942.13
37	female	23.37	2	no	northwest	6686.43
61	female	22.04	0	no	northeast	13616.36
34	male	22.42	2	no	northeast	27375.90
35	male	24.13	1	no	northwest	5125.22
42	female	23.37	0	yes	northeast	19964.75

Рис. 2.6. Новый вид веб-страницы, отображающей таблицу данных

Можно улучшить отображение таблицы, используя функции, представленные в табл. 2.3.

Использование функций из табл. 2.3 дополнит таблицу на веб-странице окном поиска, возможностью выбора количества одновременно выводимых на одном экране строк и кнопками сортировки значений в столбце по возрастанию или убыванию.

Функции для отображения таблицы

Скрипт	Функция	Назначение	Аргумент	Назначение
ui.R	tableOutput(outputId)	Создает выходной элемент: таблицу	outputId	Выходная переменная для чтения таблицы
server.R	renderDataTable()	<p>Вывод таблицы с помощью библиотеки JavaScript DataTables.</p> <p>Создает реактивную версию данной функции, возвращающую фрейм данных (или матрицу), который будет визуализироваться с помощью библиотеки DataTables.</p> <p>Подкачка, поиск, фильтрация и сортировка могут выполняться на стороне R с использованием Shiny в качестве серверной инфраструктуры</p>	{ data.frame }	Имя переменной типа data frame с данными таблицы
			options = NULL, searchDelay = 500, callback = "function(oTable) { }", escape = TRUE, env = parent.frame(), quoted = FALSE, outputArgs = list()	Аргументы, определяющие параметры вывода таблицы
ui.R	DT::dataTableOutput(outputId)	<p>Функция из пакета DT (используется с парной функцией того же пакета для сервера).</p> <p>Создает выходной элемент: таблицу</p>	outputId	Выходная переменная для чтения таблицы
server.R	DT::renderDataTable()	<p>Функция из пакета DT (имеет дополнительные возможности по работе с таблицами на стороне сервера и клиента)</p>	DT::datatable({ data.frame })	Имя переменной типа data frame с данными таблицы

Вместо боковой панели можно отобразить виджеты над таблицей, используя функцию для создания «жидких макетов» `fluidRow()`. «Жидкий макет» – это общий термин, который включает набор определенных правил «жидкой» страницы: масштабирование, повторное выравнивание по центру, ориентирование по направляющей и объекту. Строки включают элементы, которые отображаются в одной строке (если браузер имеет достаточную ширину). Для перехода на новую строку нужно использовать новую функцию `fluidRow()`. Аргументом функции будет функция столбца `column()`, которая определяет, сколько горизонтального пространства в пределах сетки шириной 12 единиц должны занимать ее элементы. «Жидкие» страницы масштабируют свои компоненты в реальном времени, чтобы заполнить всю доступную ширину браузера.

Изменим код на следующий:

ui.R

```
insurance<-read.table('insurance.csv', sep=',', header=T)
fluidPage(
  titlePanel("Insurance"),
  # Создание строки для ввода значений параметров
  fluidRow(
    column(1,
      numericInput(inputId = "age1",
        label = "Age>",
        min = 18,
        max = 90,
        value = 18)
    ),
    column(1,
      numericInput(inputId = "age2",
        label = "Age<",
        min = 18,
        max = 67,
        value = 65)
    ),
    column(1,
      selectInput(inputId = "smoker",
        label = "Smoker:",
        choices = c("all",unique(as.character(df_film$smoker))))
    ),
    column(1,
      radioButtons(inputId = "sex",
        label = "Sex:",
        choices = list(
          "all" = "all",
          "female" = "female",
```

```

        "male" = "male"
      ),
      selected = "all")
    ),
    column(2,
      selectInput(inputId = "region",
        label = "Region:",
        choices = c("all", unique(as.character(insurance$region))))
    ),
    column(1,
      selectInput(inputId = "children",
        label = "Children:",
        choices = c("all", unique(as.character(insurance$children))))
    ),
    column(5,
      sliderInput(inputId = "bmi",
        label = "Bmi",
        min = min(insurance$bmi),
        max = max(insurance$bmi),
        value = c(20,25))
    )
  ),
  # Создание таблицы с новой строки
  DT::dataTableOutput("table")
)

```

server.R

```

function(input, output) {
  # Фильтрация данных, на основе введенных пользователем значений
  output$table <- DT::renderDataTable(DT::datatable({
    data <- insurance
    data <- data[(data$age > input$age1) & (data$age < input$age2),]
    if (input$sex != "all") {
      data <- data[data$sex == input$sex,]
    }
    if (input$children != "all") {
      data <- data[data$children == input$children,]
    }
    if (input$smoker != "all") {
      data <- data[data$smoker == input$smoker,]
    }
    if (input$region != "all") {
      data <- data[data$region == input$region,]
    }
    data <- data[(data$bmi >= min(input$bmi)) & (data$bmi <=
max(input$bmi)),]
    data
  })))
}

```

В браузере веб-страница примет вид, как на рис. 2.7.

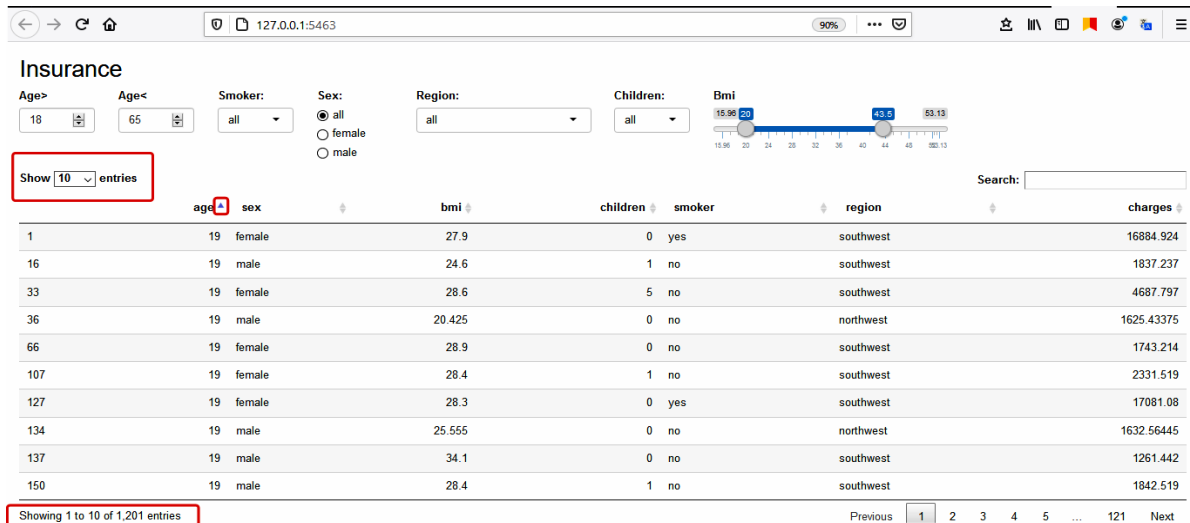


Рис. 2.7. Новый вид веб-страницы, отображающей таблицу данных

2.3. Создание графиков

Создадим новое веб-приложение `example_2`. Для таблицы данных `insurance` в интерфейс пользователя включим возможность выбора числового показателя, а на главную панель выведем график `boxplot()` – «ящик с усами»:

ui.R

```
insurance<-read.table('insurance.csv', sep=',', header=T)
shinyUI(fluidPage(
  titlePanel("Insurance"),
  sidebarLayout(sidebarPanel(
    radioButtons(
      inputId = "characterstic",
      label = "Select the characterstic",
      choices = c(
        "Age" = "age",
        "Bmi" = "bmi",
        "Charges" = "charges"
      ),
      selected = "charges"
    )
  ),
  mainPanel(plotOutput("myplot"))
))
```

server.R

```
shinyServer(function(input, output) {
  output$myplot = renderPlot({
    boxplot(insurance[, input$characterstic], main = "Boxplot")
  })
})
```

В браузере веб-страница примет вид, как на рис. 2.8.

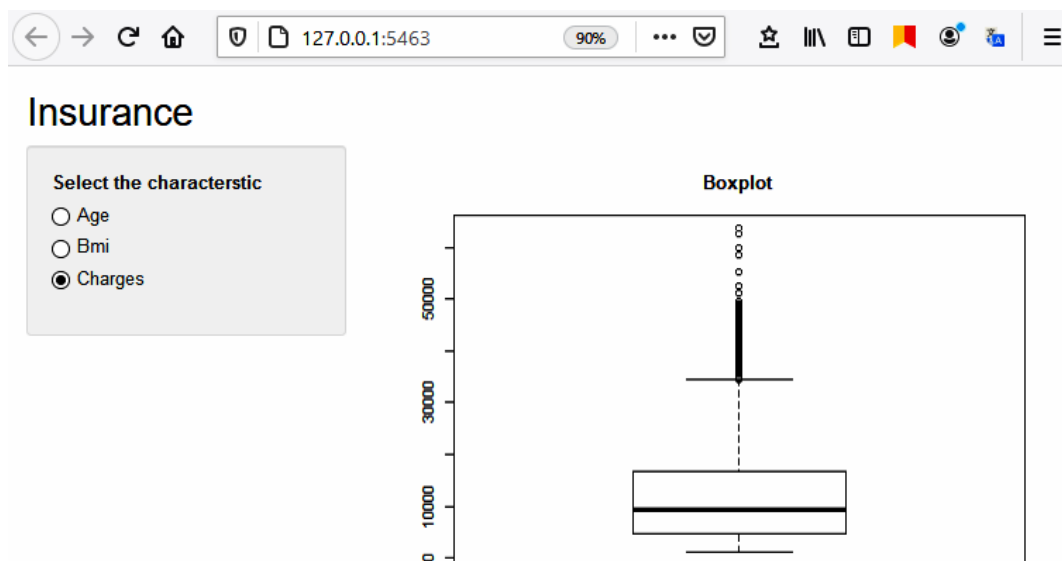


Рис. 2.8. Веб-страница с построенным графиком для столбца таблицы данных charges

Описание используемых функций приведем в табл. 2.4.

Таблица 2.4

Функции для отображения графиков

Скрипт	Функция	Назначение	Аргумент	Назначение
ui.R	plotOutput(outputId)	Создает выходной элемент: график	outputId	Выходная переменная для чтения графика
server.R	renderPlot()	Создает реактивный график, назначаемый переменной вывода: output\$outputId	{expr}	Функция для построения графика на основе некоторых данных. Например, <code>expr = plot()</code> или <code>expr = hist()</code>
			width = "auto", height = "auto", res = 72, ..., env = parent.frame(), quoted = FALSE, execOnResize = FALSE, outputArgs = list()	Аргументы, определяющие параметры вывода графика

Включим в веб-страницу отображение описательных статистик для выбранного столбца. Для этого достаточно изменить код следующим образом: в `ui.R` внесем изменения в функцию `mainPanel()`, добавив еще один вывод `verbatimTextOutput("mysummary")`, а в `server.R` внесем изменения в `function(input, output){}`, добавив отображение сводной статистики для значений выбранного столбца таблицы данных. Приведем ниже только строки, требующие изменения:

ui.R

```
mainPanel(verbatimTextOutput("mysummary"), plotOutput("myplot"))
```

server.R

```
output$mysummary = renderPrint({
  summary(insurance[, input$characterstic])
})
```

На рис. 2.9 представлен вид веб-страницы после изменений.

Insurance

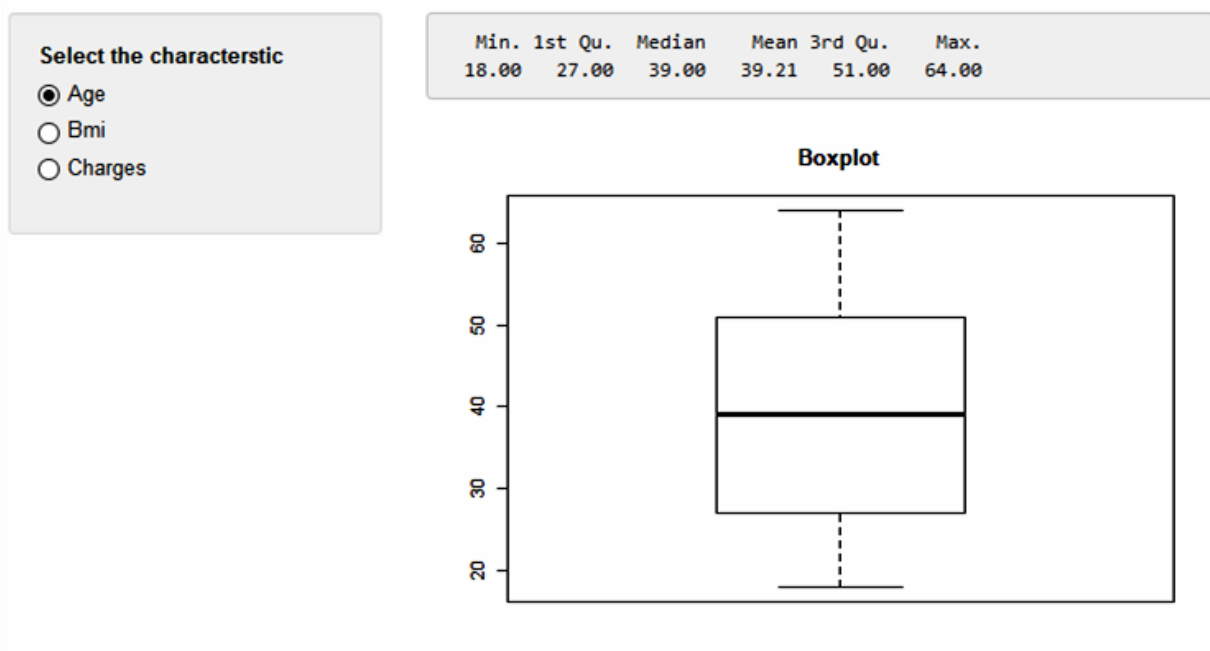


Рис. 2.9. Веб-страница с построенным графиком для столбца таблицы данных `age` и описательными статистиками для этого столбца

На странице можно организовать несколько вкладок, перемещаясь по которым можно изменять вид отображаемой на экране информации. Для этого необходимо в файле `ui.R` для `mainPanel()` использовать функцию `tabsetPanel(tabPanel(), tabPanel())`:

ui.R

```
mainPanel(tabsetPanel(  
  tabPanel("Summary", verbatimTextOutput("mysummary")),  
  tabPanel("Boxplot", plotOutput("myplot"))  
)
```

Новый вид веб-страницы с активными вкладками Boxplot и Summary представлен на рис. 2.10 и 2.11.

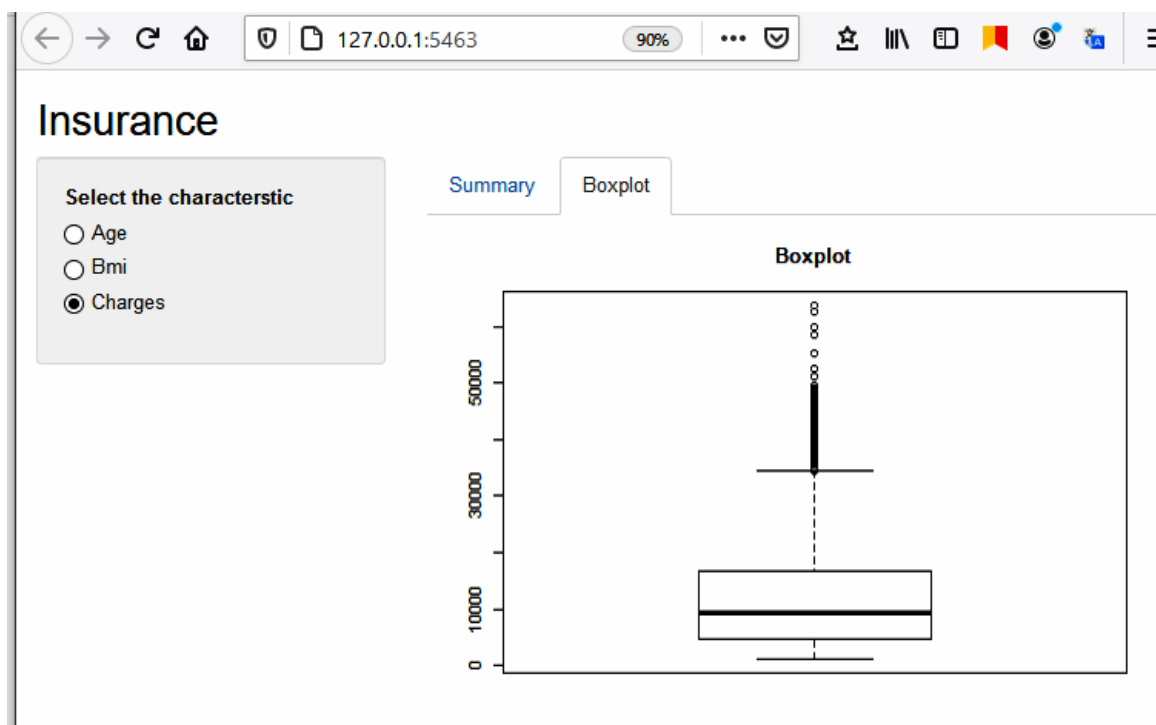


Рис. 2.10. Вкладка Boxplot для столбца данных charges

Insurance

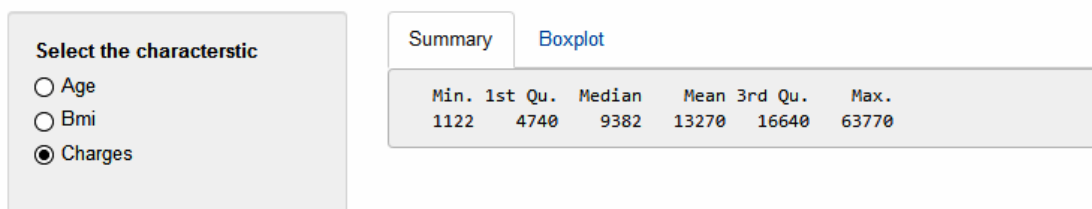


Рис. 2.11. Вкладка Summary для столбца данных charges

Создадим новое веб-приложение example_3. В интерфейсе пользователя для таблицы данных insurance включим возможность выбирать факторный показатель для построения графика boxplot() по переменной charges отдельно по каждому из значений факторной переменной.

ui.R

```
insurance<-read.table('insurance.csv', sep=',', header=T)
shinyUI(fluidPage(
  titlePanel("Insurance"),
  sidebarLayout(sidebarPanel(
    radioButtons(
      inputId = "characterstic",
      label = "Select the characterstic",
      choices = c(
        "Sex" = "sex",
        "Smoker" = "smoker"
      ),
      selected = "sex"
    )
  ),
  mainPanel(plotOutput("Boxplot"))
))
```

server.R

```
shinyServer(function(input, output) {
  output$Boxplot <- renderPlot({
    boxplot(insurance[, "charges"]~factor(insurance[,input$characterstic]),
      xlab = input$characterstic, ylab = "charges")
  })
})
```

Веб-страница с возможностью выбора факторов smoker и sex для построения графика boxplot() по переменной charges представлена на рис. 2.12 и 2.13.

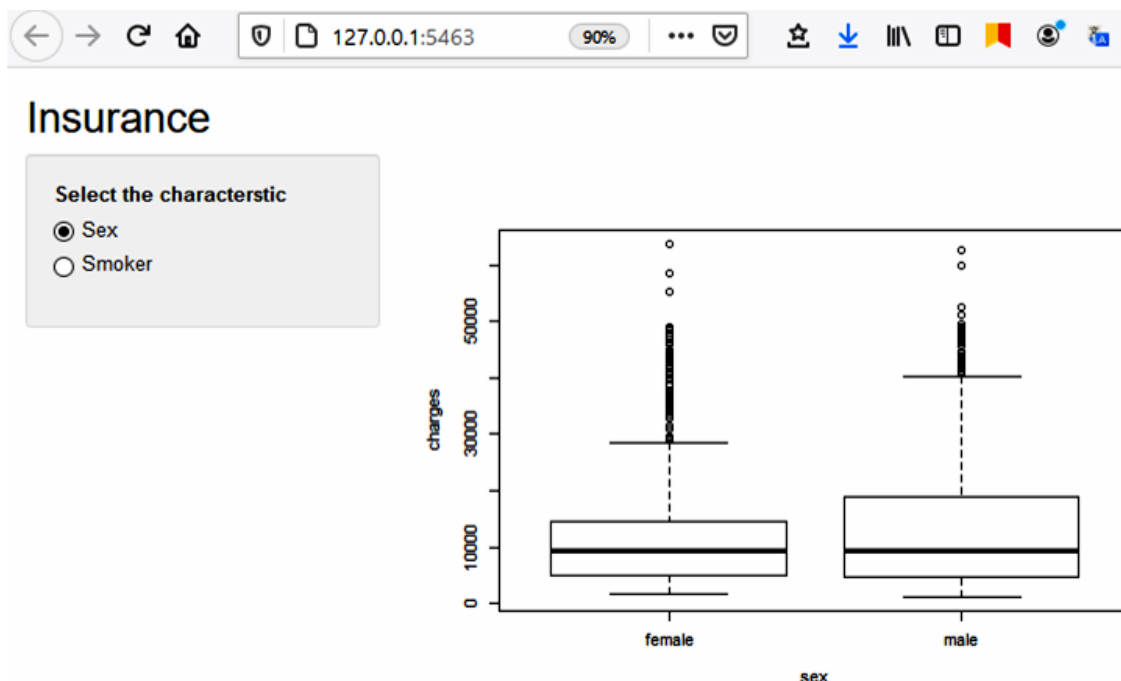


Рис. 2.12. Визуализация группировки расходов по значению переменной *пол*

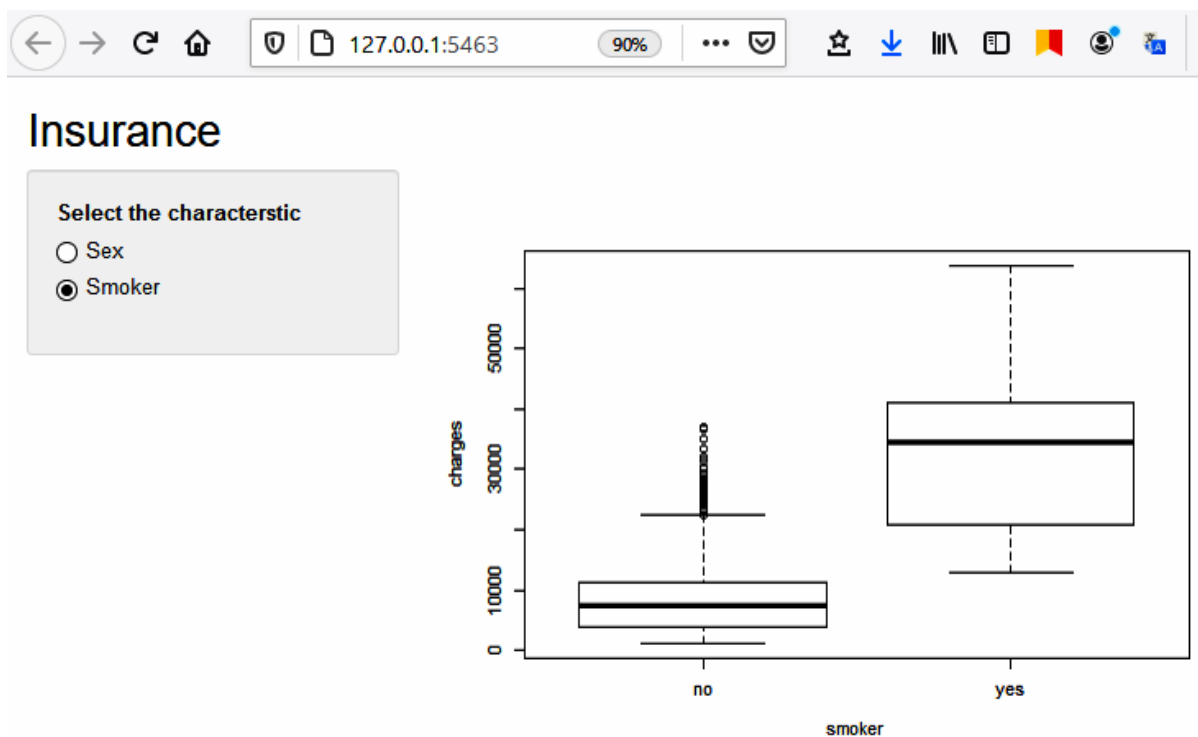


Рис. 2.13. Визуализация группировки расходов по значению переменной *наличие вредной привычки*

2.4. Создание динамичных пользовательских интерфейсов

Функции `reactive()` и `observe()` – это функции обертки, которые позволяют приложению `shiny` реагировать на изменения входных данных в выражениях, к которым применены эти функции. Однако реактивные выражения не выполняются сразу же, когда в их зависимостях что-то меняется, а ждут, когда будут вызваны, чтобы повторно выполниться с новыми значениями аргументов. Это позволяет избежать многократного пересчета, если результат изменений не требуется.

Функция `observe()` в отличие от `reactive()` повторно выполняется сразу же, как только изменяются зависимости. Также `observe()` в отличие от `reactive()` не возвращает результат и не может использоваться в качестве входных данных для других реактивных выражений.

Для отображения результатов выполнения скриптов на веб-странице на основе некоторых входных данных использовались функции `textOutput()`, `tableOutput()` и `plotOutput()`.

Теперь попробуем реализовать интерфейс, в котором пользователь может выбрать тип вывода с помощью радиокнопки.

Этот вид динамического интерфейса может быть создан с помощью функций `conditionalPanel()` и `renderUI()`.

Функция `conditionalPanel()` используется в `ui.R` и оборачивает набор элементов пользовательского интерфейса, которые должны быть динамически показаны или скрыты.

Функция `renderUI()` используется в `server.R` совместно с функцией `uiOutput()` в `ui.R`.

Создадим новое веб-приложение `example_4`, в котором для таблицы данных `insurance` пользователь выбирает переменную, с которой хочет работать, и его выбор сохраняется в 'а' с помощью функции `reactive()`. Виджет, внедренный с помощью `uiOutput()`, дает пользователю возможность выбрать, какой тип отображения информации его интересует: таблица, сводка или график. На основе выбранной пользователем опции создаются условные панели для сводки описательной статистики, просмотра данных и построения графика.

ui.R

```
insurance<-read.table('insurance.csv', sep=',', header=T)
ui = fluidPage(titlePanel(em("Conditional panels")),
  sidebarLayout(
    sidebarPanel(
      selectInput(
        "Choice1",
        "Select the variable",
        choices = colnames(insurance)[1:7],
        selected = "age"
      ),
      uiOutput("Out1")
    ),
    mainPanel(
      conditionalPanel("input.Choice2 === 'Summary'",
        verbatimTextOutput("Out2")),
      conditionalPanel("input.Choice2 === 'View data'",
        tableOutput("Out3")),
      conditionalPanel("input.Choice2 === 'Plot'",
        plotOutput("Out4"))
    )
  )
)
```

server.R

```
server = function(input, output) {
  a = reactive({
```

```

insurance[, colnames(insurance) == input$Choice1]
})
output$Out1 = renderUI({
  radioButtons(
    "Choice2",
    "What do you want to do?",
    choices = c("Summary", "View data", "Plot"),
    selected = "Summary"
  )
})
output$Out2 = renderPrint({
  summary(a())
})
output$Out3 = renderTable({
  return(a())
})
output$Out4 = renderPlot({
  return(plot(a(), insurance[, "charges"],
    main = "Plot", xlab = input$Choice1, ylab = "charges"))
})
}

```

Веб-страница с динамичным пользовательским интерфейсом представлена на рис. 2.14–2.17.

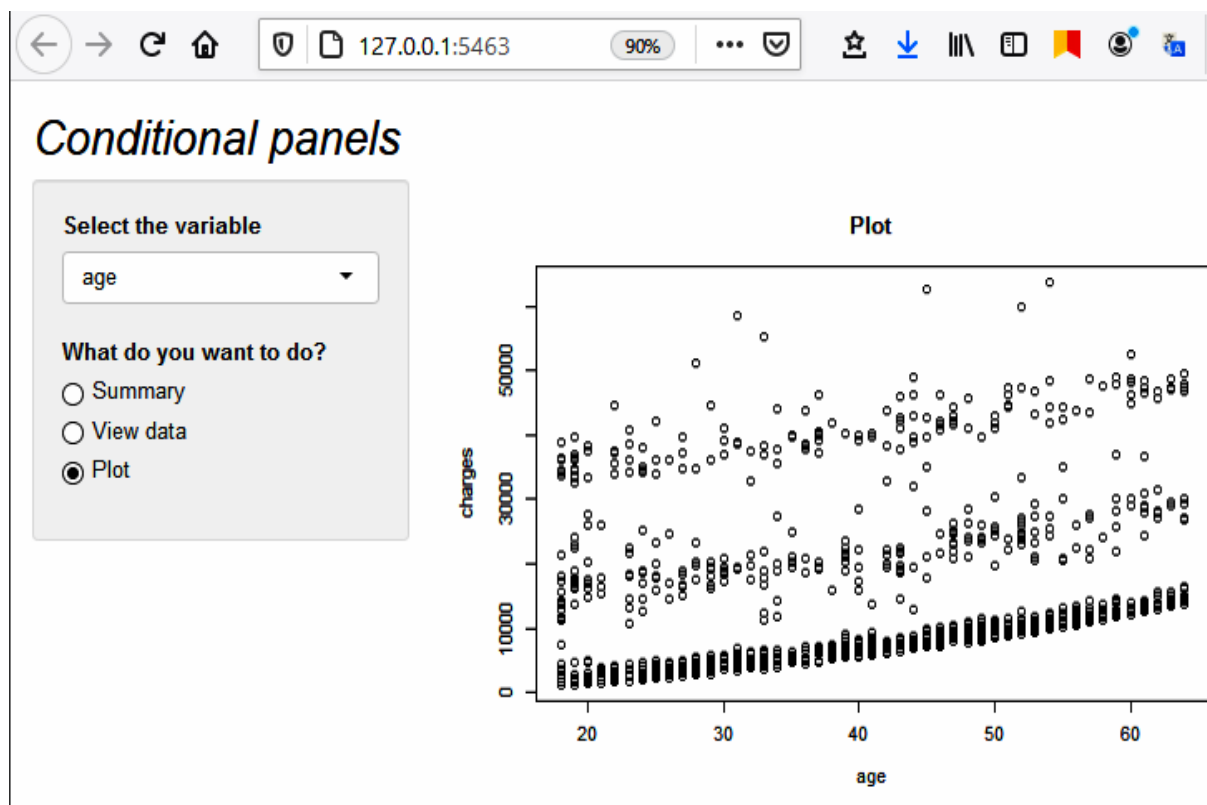


Рис. 2.14. Выбор переменной age и построения графика

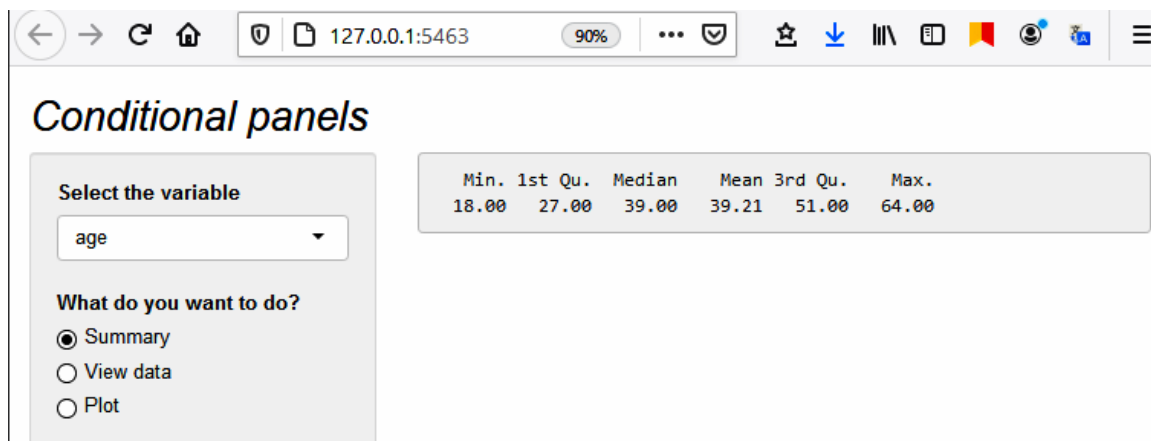


Рис. 2.15. Выбор переменной age и вывода описательной статистики



Рис. 2.16. Выбор переменной smoker и вывода описательной статистики

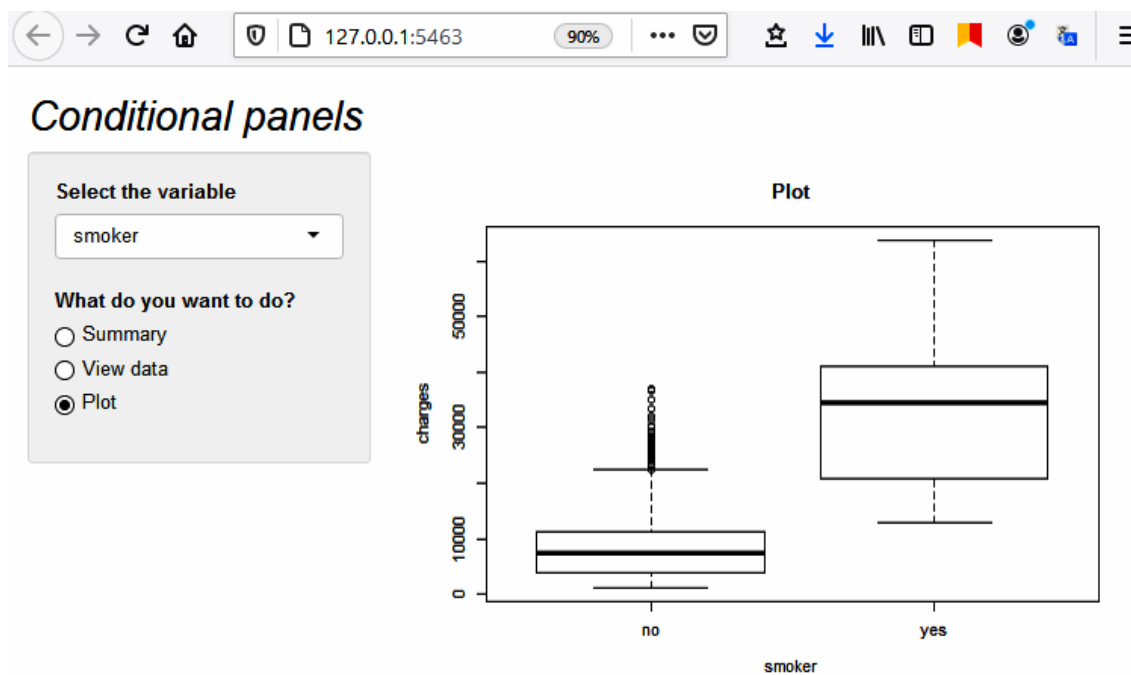


Рис. 2.17. Выбор переменной smoker и построения графика

Демонстрацию возможностей визуализации с помощью пакета shiny можно увидеть в Галерее [11].

2.5. Загрузка данных

Создадим новое приложение example_5. Функция fileInput() в ui.R позволяет пользователям загружать свой собственный файл. Аргумент multiple = F означает, что пользователь может загрузить только один файл, а аргумент accept = csv определяет тип файлов, которые могут быть загружены.

Также в приложении пользователь может выбрать, хочет ли он просмотреть заголовок данных или весь набор данных, который затем просматривается с помощью функции renderTable().

ui.R

```
ui = fluidPage(titlePanel("Uploading file"),
  sidebarLayout(
    sidebarPanel(
      fileInput(
        inputId = "myfile",
        label = "Choose CSV File",
        multiple = F,
        accept = ".csv"
      ),
      checkboxInput("header", "Header", TRUE),
      radioButtons(
        "choice",
        "Display",
        choices = c(Head = "head",
                    All = "all"),
        selected = "head"
      )
    ),
    mainPanel(tableOutput("contents"))
  )
)
```

server.R

```
server = function(input, output) {
  output$contents = renderTable({
    req(input$myfile)
    data = read.csv(input$myfile$datapath,
                    header = input$header)
    if (input$choice == "head") {
      return(head(data))
    }
  })
}
```



```

    }
    else {
        return(data)
    }
  })
}

```

Веб-страница с возможностью загрузки файла представлена на рис. 2.18.

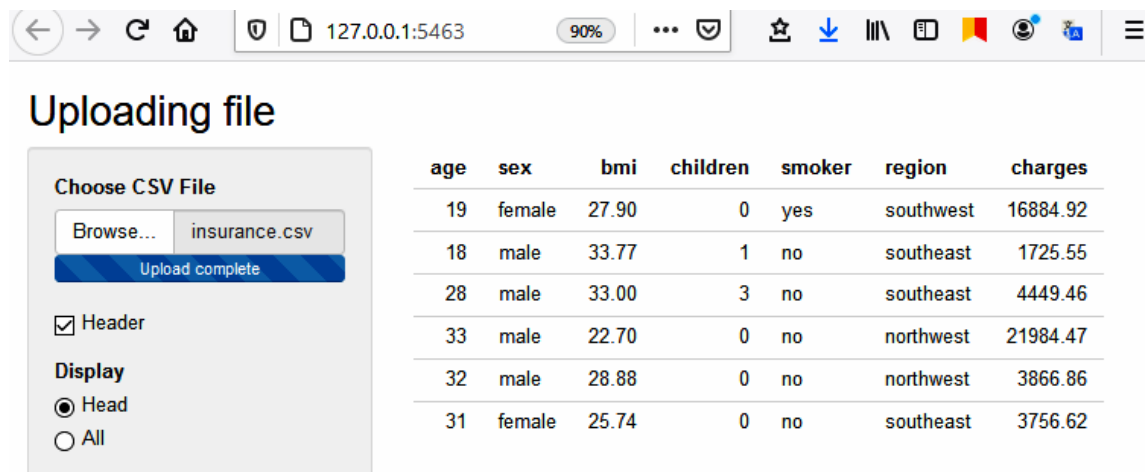


Рис. 2.18. Загрузка файла insurance

Описание используемых функций приведем в табл. 2.5.

Таблица 2.5

Функция для загрузки файлов

Скрипт	Функция	Назначение	Аргумент	Назначение
ui.R	fileInput()	Создает выходной элемент: график	inputId	Входная переменная для доступа к значению
			label	Метка, отображаемая на экране рядом с загружаемым файлом (необязательный аргумент)
			multiple = FALSE, accept = NULL, width = NULL, buttonLabel = "Browse...", placeholder = "No file selected"	Аргументы, определяющие параметры загрузки файла

2.6. Лабораторная работа

«Создание веб-приложения на языке R»

Цель работы: научиться создавать реактивные приложения на основе данных.

Формируемые знания, умения и навыки: знать основные возможности пакета `shiny` для создания веб-приложений на языке R; получить навыки создания приложений на языке R.

Необходимо:

1. С сайта: <https://www.kaggle.com/> импортировать один из наборов.
2. Используя возможности пакета `shiny` на языке R, создать веб-приложение с возможностью выбора данных для анализа, с различными вариантами визуализации.

Контрольные вопросы и задания

1. С помощью каких функций можно создать «жидкий» макет?
2. Как отобразить в приложении таблицу данных?
3. Как отобразить в приложении график?
4. Как отобразить в приложении текст?
5. Какие виджеты есть в пакете `shiny`, для чего они предназначены?
6. Из каких элементов состоит `ui.R`?
7. Из каких элементов состоит `server.R`?

3. СОЗДАНИЕ ВЕБ-ПРИЛОЖЕНИЙ НА ЯЗЫКЕ PYTHON С ПОМОЩЬЮ СРЕДЫ РАЗРАБОТКИ ANACONDA И ТЕРМИНАЛА PROMT

Для создания приложений на языке Python установим Anaconda с официального сайта: <https://www.anaconda.com/distribution/> На момент написания пособия актуальна версия Anaconda для Python 3.7 [12].

Код скриптов будем создавать в редакторе кода Visual Studio Code – open source продукте от компании Microsoft [13]. Однако в качестве редактора кода может быть выбран иной, например, Sublime Text [14].

3.1. Библиотека streamlit

В Python для создания веб-приложений используется библиотека streamlit [15].

Для ее подключения в сеансе работы запустим терминал Promt в составе Anaconda (рис. 3.1).

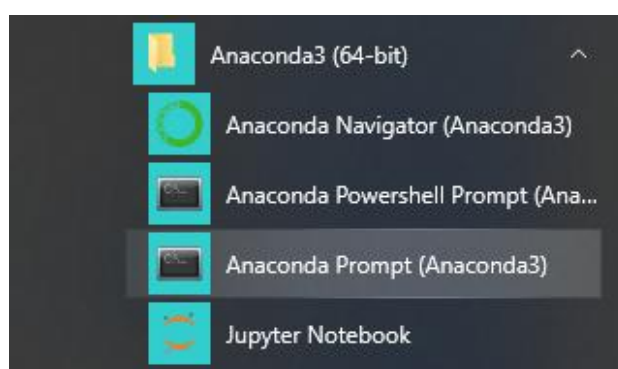


Рис. 3.1. Запуск терминала

В терминале наберем команду, устанавливающую библиотеку streamlit:

```
pip install streamlit
```

Создадим веб-приложение с помощью редактора кода Visual Studio Code, сохранив его с именем first_app.py в папку, которая содержит папку с именем .anaconda. В этом случае для запуска файла на выполнение из терминала достаточно будет указать его имя, а не полный путь. В этой же папке (на одном уровне с файлом приложения) сохраним файл данных insurance.csv, к которому обращается приложение для считывания данных из файла.

first_app.py

```
import streamlit as st
import pandas as pd

st.title('My first app')
df = pd.read_csv("insurance.csv")
st.write(df)
```

Как выглядит код в редакторе кода, показано на рис. 3.2.

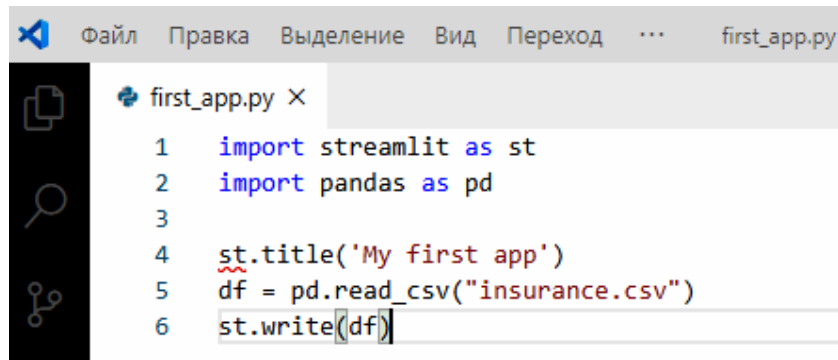


Рис. 3.2. Файл first_app.py в редакторе кода

После чего запустим файл на выполнение в терминале:

```
streamlit run first_app.py
```

Откроется вкладка браузера, и приложение будет иметь вид, как на рис. 3.3.

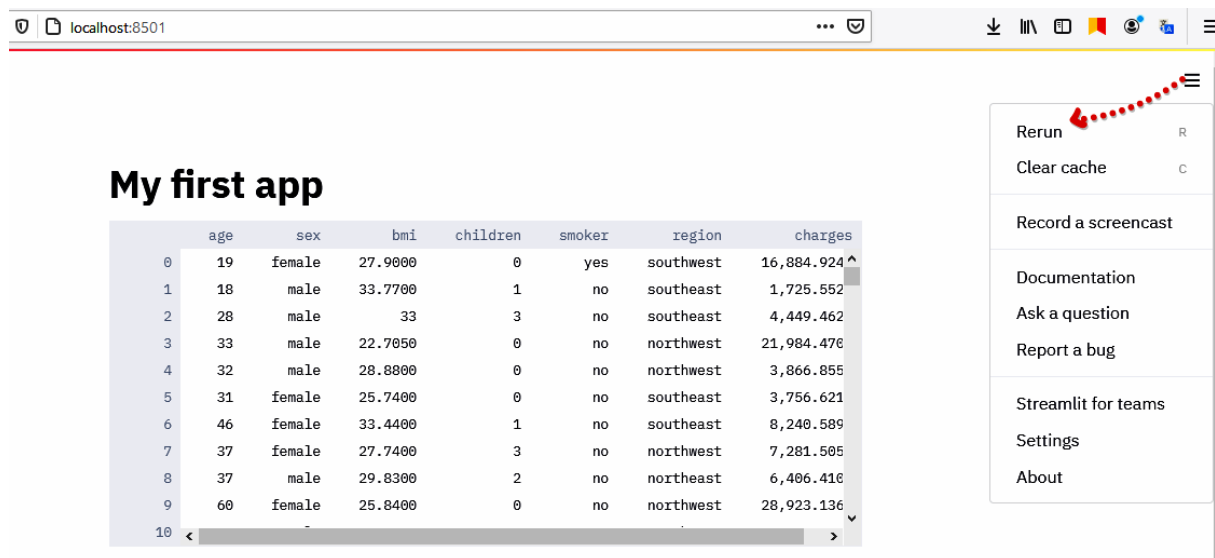


Рис. 3.3. Приложение first_app.py

Если приложение не открылось, то можно в адресной строке набрать локальный URL для приложения, который выведется в терминале после запуска приложения first_app.py (рис 3.4).

```
(base) C:\Users\Svetlana>streamlit run first_app.py

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://82.209.99.86:8501
```

Рис. 3.4. Запуск приложения в терминале

После внесения изменений и сохранения файла приложения оно перезапускается на выполнение в браузере.

Остановить приложение можно в терминале, нажав одновременно Ctrl+c.

Созданное приложение считывает файл данных в переменную `df` как `dataFrame` библиотеки `pandas` и выводит на экран заголовок приложения с помощью метода `st.title()` и таблицу данных с помощью метода `st.write()`. Как видим, к базовым методам Python просто добавляется `st.` (т.е. объявленное при подключении библиотеки `streamlit` сокращение для ее имени), что позволяет методам отрабатывать функционал в веб-приложении.

3.2. Управление выводом таблицы на экран, виджеты

Дополним интерфейс страницы с выводом таблицы виджетами (табл. 3.1).

Таблица 3.1

Виджеты [16]

Виджет	Назначение	Параметры	Описание
1	2	3	4
streamlit.checkbox(label, value=False, key=None)	Отображение виджета флаг (элемент графического пользовательского интерфейса, позволяющий пользователю управлять параметром с двумя состояниями – <input checked="" type="checkbox"/> включено и <input type="checkbox"/> отключено)	label (str)	Метка, объясняющая пользователю, для чего предназначен этот флажок
		value (bool)	Значение, на котором будет установлен флажок при запуске приложения
		key (str)	Необязательный параметр, используется в качестве уникального ключа для виджета. Если этот параметр опущен, то для виджета будет сгенерирован ключ на основе его содержимого

1	2	3	4
streamlit.button(label, key=None)	Отображение виджета кнопка	label (str)	Метка, объясняющая пользователю, для чего предназначена эта кнопка
		key (str)	Необязательный параметр, используется в качестве уникального ключа для виджета
streamlit.radio(label, options, index=0, format_func=<class 'str'>, key=None)	Отображение виджета радиокнопка (элемент интерфейса, который позволяет пользователю выбрать одну опцию (пункт) из предопределенного набора (группы))	label (str)	Метка, объясняющая пользователю, для чего предназначена эта группа переключателей
		options (list, tuple, numpy.ndarray, pandas.Series, or pandas.DataFrame)	Метки для переключателей
		index (int)	Индекс переключателя, который будет выбран при первом запуске приложения
		format_func (function)	Функция для изменения отображения переключателей
		key (str)	Необязательный параметр, используется в качестве уникального ключа для виджета
streamlit.slider(label, min_value=None, max_value=None, value=None, step=None, format=None, key=None)	Отображение виджета слайдера поддерживает типы int, float, date, time и datetime. Это также позволяет визуализировать ползунок диапазона, передавая в качестве значения двухэлементный кортеж или список. Разница между st.slider() и st.select_slider() заключается в том, что slider	label (str or None)	Метка, объясняющая пользователю, для чего предназначен этот слайдер
		min_value	Минимально допустимое значение. По умолчанию 0, если значение является int; 0.0 – если float; value – timedelta (days=14), если date/datetime
		max_value	Максимально допустимое значение. По умолчанию 100, если значение является int; 1.0 – если float; value + timedelta (days=14), если date/datetime
		value	Значение, на котором будет установлен ползунок при его первом отображении. Если передается список из двух значений, то отображаются два ползунка диапазона с этими границами

1	2	3	4
	принимает только числовые данные или данные даты/времени и диапазон в качестве входных данных, в то время как select_slider принимает любой тип данных и итеративный набор параметров	step (int/float/timedelta or None)	Интервал шага. По умолчанию используется значение 1, если значение равно int; 0.01 – если float; timedelta(days=1), если дата/datetime
		format (str or None)	Влияет на отображение чисел на экране
		key (str)	Необязательный параметр, используется в качестве уникального ключа для виджета
streamlit.number_input (label, min_value=None, max_value=None, value=<streamlit.elements.utils.NoValue object>, step=None, format=None, key=None)	Отображение виджета числового ввода	label (str or None)	Метка, объясняющая пользователю, для чего предназначен этот ввод
		min_value	Минимально допустимое значение. Если не задано, то не будет и минимума
		max_value	Максимально допустимое значение. Если не задано, то не будет и максимума
		value	Значение, на котором будет установлен виджет при его первом отображении. По умолчанию используется значение min_value или 0.0, если значение min_value равно None
		step (int/float or None)	По умолчанию используется значение 1, если значение равно int, и 0.01 – в противном случае
		format (str or None)	Влияет на отображение чисел на экране
		key (str)	Необязательный параметр, используется в качестве уникального ключа для виджета

Создадим в редакторе кода файл example_1.py, в котором добавим таблице реактивности.

example_1.py

```
import streamlit as st
import pandas as pd
```

```

st.title('Insurance')
df = pd.read_csv("insurance.csv")
Region = st.multiselect('Choose regions', df['region'].unique())
Smoker = st.multiselect('Choose smoker', df['smoker'].unique())
Sex = st.radio('Choose a gender', df['sex'].unique())
Age = st.slider('Choose age', min_value=17, max_value=66, value=(20, 50))
Children = st.number_input('Choose the number of children', min_value= 0,
                           max_value=5, value =1, step=1)

new_df = df[(df['region'].isin(Region)) & (df['smoker'].isin(Smoker)) & (df['sex'] =
= Sex)
            & (df['age']> Age[0]) & (df['age']< Age[1])
            & (df['children'] == Children)]
st.write(new_df)

```

Запустим приложение в терминале:

```
streamlit run example_1.py
```

Начальный вид веб-страницы представлен на рис. 3.5. Таблица отображается пустой, так как для показателей регион (region) и наличие вредной привычки (smoker) пока не выбрано ни одного значения для отображения. Метод `st.multiselect()` позволяет выбрать сразу несколько значений, а метод `st.number_input()` – ввести только одно, метод `st.radio()` также позволяет выбрать только одно значение из группы.

Рис. 3.5. Веб-приложение с начальными установками (при первом запуске)

При выборе значений для показателей регион (region), наличие вредной привычки (smoker), изменение числа детей в семье получим отображение таблицы с соответствующими пользовательскому выбору строками данных (рис. 3.6). Также можно перемещать ползунки слайдера для выбора интервала для возрастной группы отображаемых в таблице клиентов.

The screenshot shows a web browser at localhost:8501 displaying an application titled "Insurance". The interface includes several filters:

- Choose regions:** Two buttons labeled "southwest" and "northwest", both with an 'X' to remove them.
- Choose smoker:** Two buttons labeled "yes" and "no", both with an 'X' to remove them.
- Choose a gender:** Two radio buttons, "female" (selected) and "male".
- Choose age:** A range slider with a minimum value of 17 and a maximum value of 66. The current range is from 20 to 50.
- Choose the number of children:** A numeric input field showing "0", with minus and plus buttons to adjust the value.

Below the filters is a table of data. The table has 8 columns: an index column, and columns for age, sex, bmi, children, smoker, region, and charges. The table is filtered to show 10 rows of data.

	age	sex	bmi	children	smoker	region	charges
47	28	female	34.7700	0	no	northwest	3,556.92
79	41	female	32.9650	0	no	northwest	6,571.02
88	46	female	27.7400	0	no	northwest	8,026.66
100	41	female	31.6000	0	no	southwest	6,186.12
113	21	female	35.7200	0	no	northwest	2,404.78
139	22	female	36	0	no	southwest	2,166.78
160	42	female	26.6000	0	yes	northwest	21,348.76
174	24	female	33.3450	0	no	northwest	2,855.43
183	44	female	26.4100	0	no	northwest	7,419.47
191	36	female	26.2000	0	no	southwest	4,883.86

Рис. 3.6. Отображение таблицы данных в веб-приложении с пользовательскими настройками для значений отображаемых строк

В этом приложении совсем не используется пространство слева. Изменим пользовательский интерфейс, организовав боковую панель с перенесенными на нее виджетами для выбора значений для отображения.

Для этого используем метод `st.sidebar[element_name]`. Внесем изменения в код:

example_1.py

```
import streamlit as st
import pandas as pd

st.title('Insurance')
df = pd.read_csv("insurance.csv")
Region = st.sidebar.multiselect('Choose regions', df['region'].unique())
Smoker = st.sidebar.multiselect('Choose smoker', df['smoker'].unique())
Sex = st.sidebar.radio('Choose a gender', df['sex'].unique())
Age = st.sidebar.slider('Choose age', min_value=17, max_value=66, value=(20, 50))
Children = st.sidebar.number_input('Choose the number of children', min_value= 0,
                                  max_value=5, value = 1, step=1)

new_df = df[(df['region'].isin(Region)) & (df['smoker'].isin(Smoker)) & (df['sex'] = Sex)
            & (df['age']> Age[0]) & (df['age']< Age[1]) & (df['children'] == Children)]
st.write(new_df)
```

Сохранив новую версию файла в редакторе кода, увидим в браузере в правом верхнем углу кнопки для перезапуска приложения (рис. 3.7). Можно выбрать опцию всегда перезапускать при сохранении изменений в файле приложения.

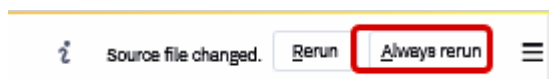


Рис. 3.7. Кнопки для перезапуска измененного файла приложения

Новый вид приложения с боковой панелью представлен на рис. 3.8 (есть возможность убрать/возвратить боковую панель на экран).

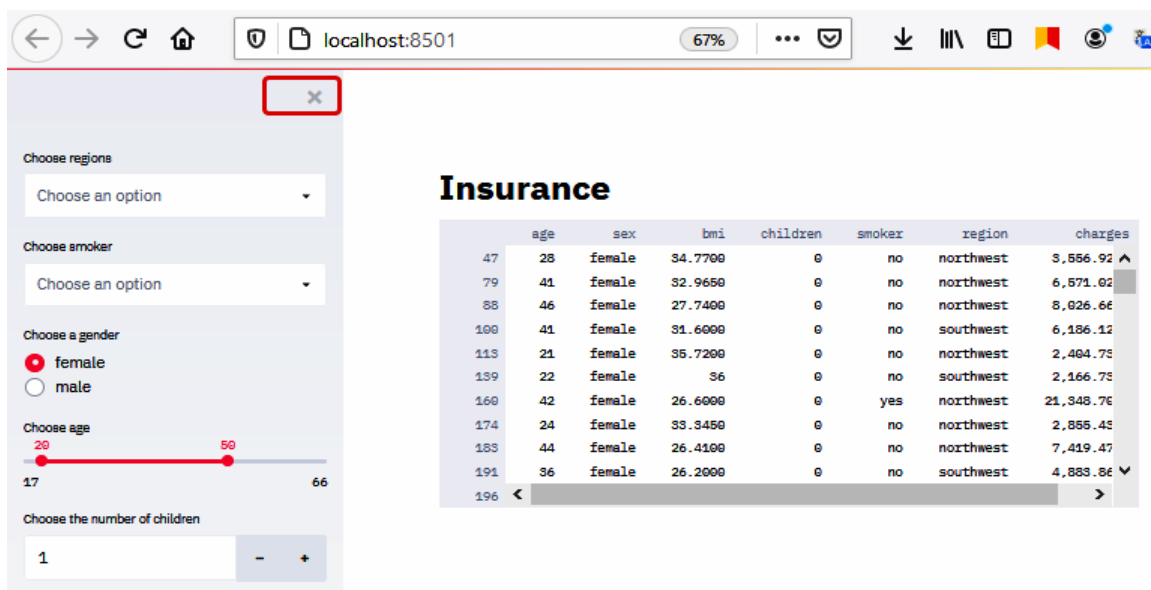


Рис. 3.8. Приложение с боковой панелью выбора пользовательских настроек для отображения данных таблицы

3.3. Создание графиков

Создадим новое веб-приложение `example_2.py`. Для таблицы данных `insurance` в интерфейс пользователя включим возможность выбора числового показателя, а на главную панель выведем график `boxplot()` – «ящик с усами» – и описательные статистики для выбранного показателя:

example_2.py

```
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt

st.title('Insurance')
df = pd.read_csv("insurance.csv")
variable = st.sidebar.radio('Choose variable', ('bmi', 'age', 'charges'))

new_df = df[variable]
st.write(df)
fig, ax = plt.subplots()
ax.boxplot(new_df)
st.pyplot(fig)
st.write(new_df.describe())
```

Для построения графика использовалась библиотека `matplotlib` (методы `pyplot`). В Python есть библиотеки с большими возможностями визуализации данных, но пока библиотека `streamlit` из коробки работает только со следующими библиотеками: `matplotlib` (основная библиотека для встраивания графики в приложения `streamlit`), `altair`, `deck.gl` (карты и 3D-графики) и некоторыми другими (`streamlit` развивается, и, возможно, поддержка со временем включит и другие популярные библиотеки).

Для анализа данных использовались возможности библиотеки `pandas` [17], также `streamlit` поддерживает библиотеку `numpy` [18].

В браузере веб-страница примет вид, как на рис. 3.9.

Описание используемых функций приведем в табл. 3.2.

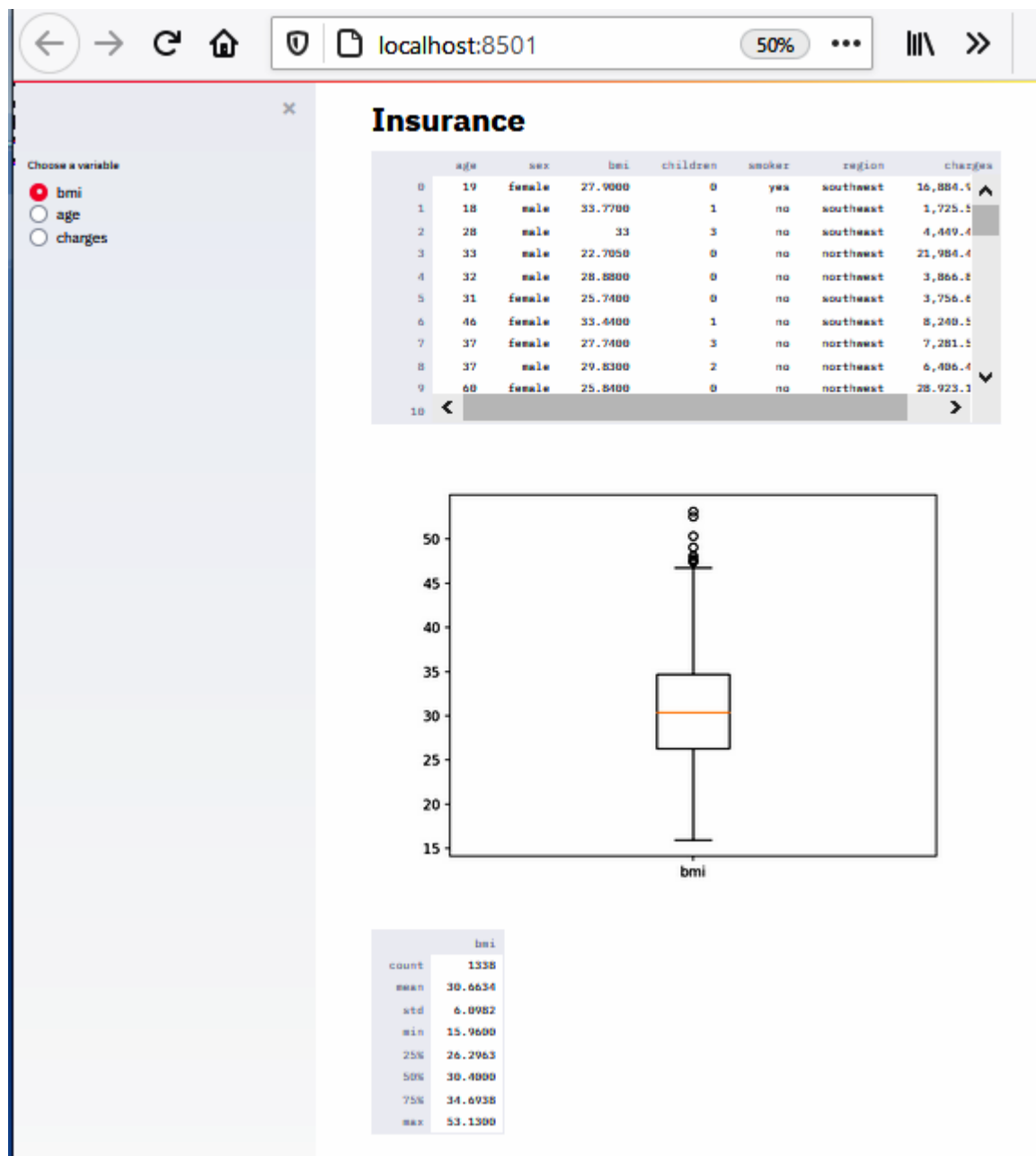


Рис. 3.9. Веб-страница с построенным графиком для столбца таблицы данных bmi

Таблица 3.2

Функции для отображения графиков [19]

Библиотека	Функция	Назначение	Аргумент	Назначение
1	2	3	4	5
streamlit	streamlit.line_chart (data=None, width=0, height=0, use_container_ width=True)	Отображение линейного графика	data (pandas. DataFrame, pandas.Styler, numpy.ndarray, Iterable, dict)	Данные для построения графика

1	2	3	4	5
			width (int) height (int) use_container_ width (bool)	Параметры, задающие ширину и высоту диаграммы
streamlit	streamlit.area_chart (data=None, width=0, height=0, use_container_width=True)	Отображение диаграммы области	data (pandas.DataFrame, pandas.Styler, numpy.ndarray, Iterable, dict)	Данные для построения графика
			width (int) height (int) use_container_ width (bool)	Параметры, задающие ширину и высоту диаграммы
streamlit	streamlit.bar_chart (data=None, width=0, height=0, use_container_width=True)	Отображение гистограммы	data (pandas.DataFrame, pandas.Styler, numpy.ndarray, Iterable, dict)	Данные для построения графика
			width (int) height (int) use_container_ width (bool)	Параметры, задающие ширину и высоту диаграммы
streamlit	streamlit.map(data=None, zoom=None, use_container_width=True)	Отображение карты с точками на ней	data (pandas.DataFrame, pandas.Styler, numpy.ndarray, Iterable, dict,)	Данные, которые должны быть показаны. Имена столбцов должны быть: 'lat', 'lon', 'latitude', or 'longitude' (долгота, широта – в ед. или мн. числе)
			zoom (int)	Масштаб
matplotlib	streamlit.pyplot (fig=None, clear_figure=None, **kwargs)	Отображение графика matplotlib.pyplot	fig (Matplotlib Figure)	Фигура для построения графика
			**kwargs (any)	Аргументы для передачи в функцию matplotlib.pyplot

Графические возможности `matplotlib.pyplot`, которые интегрируются в приложение с помощью функции `streamlit.pyplot()`, приведем в табл. 3.3.

Таблица 3.3

Графические возможности `matplotlib.pyplot` [20]

Функция	Назначение	Аргумент	Назначение
<code>boxplot()</code>	Отображение графика «ящик с усами»	x	Данные для построения графика
		<i>notch=None, sym=None, vert=None, whis=None, positions=None, widths=None, ...</i>	Параметры, задающие отображение графика
<code>hist()</code>	Отображение гистограммы	x	Данные для построения графика
		<i>bins=None, range=None, density=None, weights=None, cumulative=False, ...</i>	Параметры, задающие отображение графика
<code>scatter()</code>	Отображение точечной диаграммы с переменными значениями размера маркеров и/или цвета	x, y	Данные для построения графика
		<i>s=None, c=None, marker=None, cmap=None, norm=None, vmin=None, vmax=None, ...</i>	Параметры, задающие отображение графика

Для построения гистограммы вместо «ящика с усами» в код вносятся минимальные изменения: строка

```
ax.boxplot(new_df)
```

заменяется на строку

```
ax.hist(new_df)
```

Создадим новое веб-приложение `example_3.py`. Для таблицы данных `insurance` в интерфейс пользователя включим возможность выбора двух числовых показателей, а на главную панель выведем график `scatter()`.

```
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt
```

```

st.title('Insurance')
df = pd.read_csv("insurance.csv")
variable_1 = st.sidebar.radio('Choose a variable', ('bmi', 'age', 'charges'))
variable_2 = st.sidebar.radio('Choose a variable', ('bmi', 'age', 'charges'),
index=2)

fig, ax = plt.subplots()
ax.scatter(df[variable_1], df[variable_2])
plt.xlabel(variable_1)
plt.ylabel(variable_2)
st.pyplot(fig)

```

Обратите внимание, что функции для выбора значений `variable_1` и `variable_2` отличаются только наличием параметра `index=2`, если бы аргументы этих функций были полностью идентичны, то для них сгенерировались бы одинаковые ключи и выполнение кода завершилось бы ошибкой. Для использования в одинаковых функциях одних и тех же значений аргументов необходимо задавать и значение необязательного параметра `key`.

В браузере веб-страница примет вид, как на рис. 3.10.

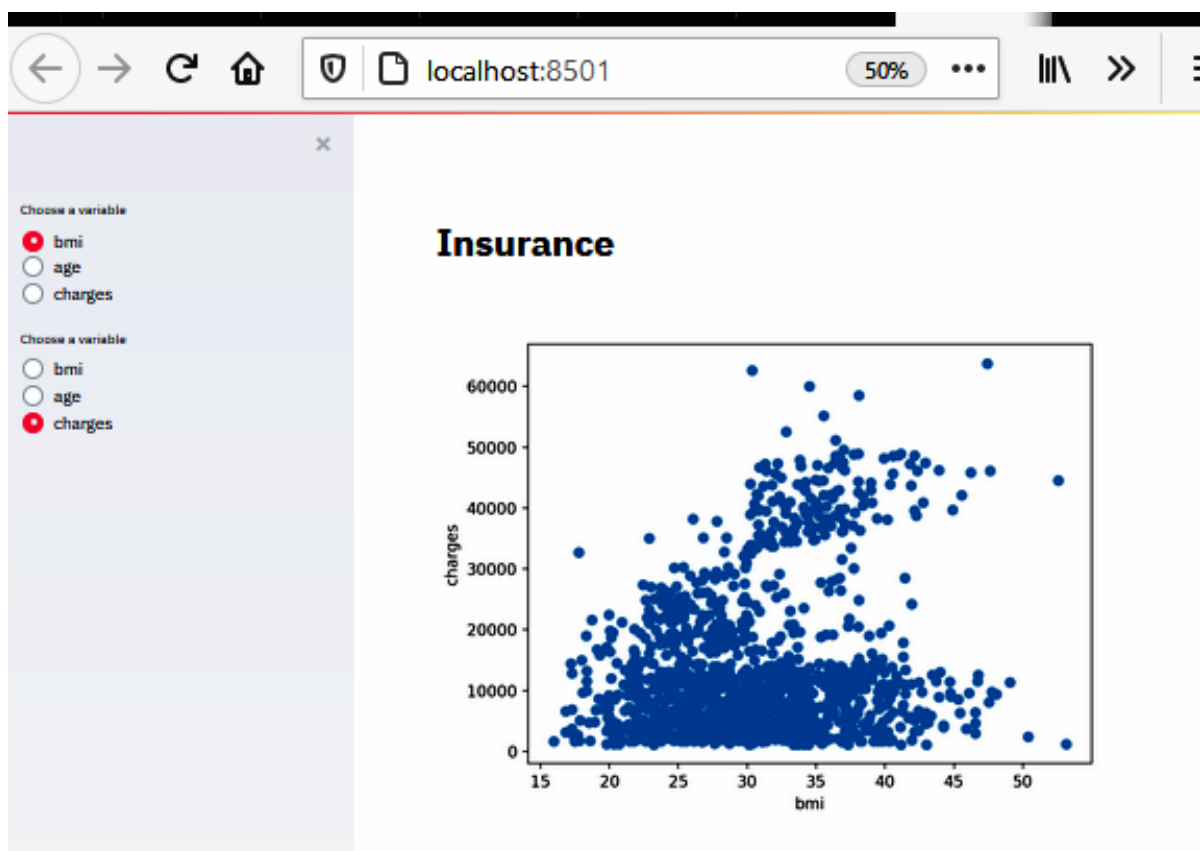


Рис. 3.10. Точечный график для показателей `bmi` и `charges`

На боковой панели можно изменить форму отображения графика и/или описательной статистики для выбранного столбца данных. Для этого используем виджет чекбокс и условный оператор if:

example_4.py

```
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt

st.title('Insurance')
df = pd.read_csv("insurance.csv")
variable = st.sidebar.radio('Choose variable', ('bmi', 'age', 'charges'))
st.sidebar.text('Show')
new_df = df[variable]
fig, ax = plt.subplots()
ax.boxplot(new_df)

if (st.sidebar.checkbox('statistics')):
    st.write(new_df.describe())
if (st.sidebar.checkbox('plot')):
    st.write(st.pyplot(fig))
```

Новый вид веб-страницы с возможностью выбора отображения графика и/или описательной статистики представлен на рис. 3.11 и 3.12.

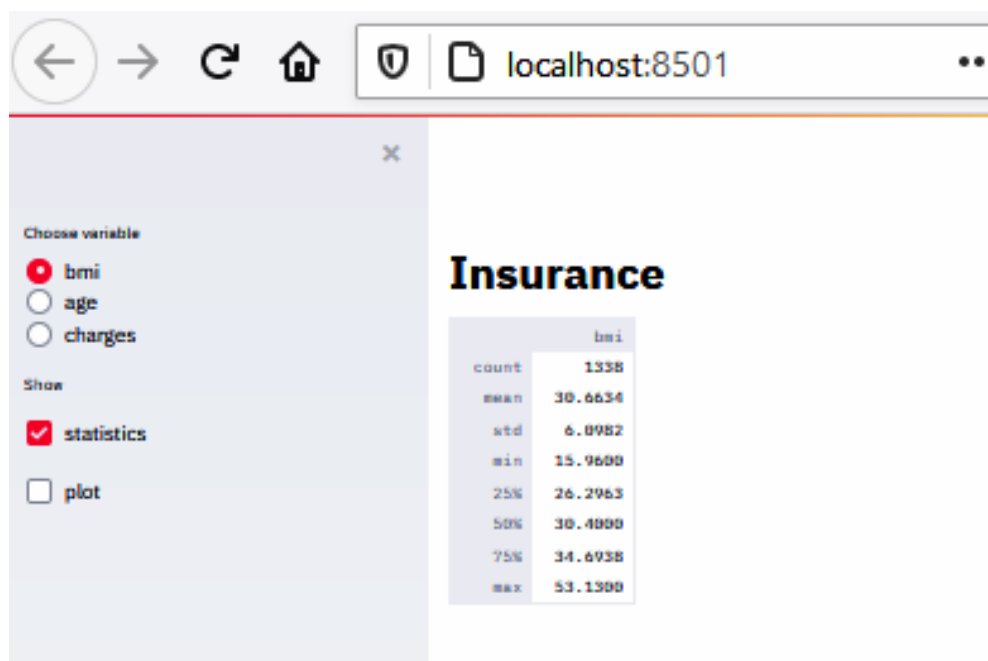


Рис. 3.11. Выбор отображения описательной статистики для столбца данных bmi

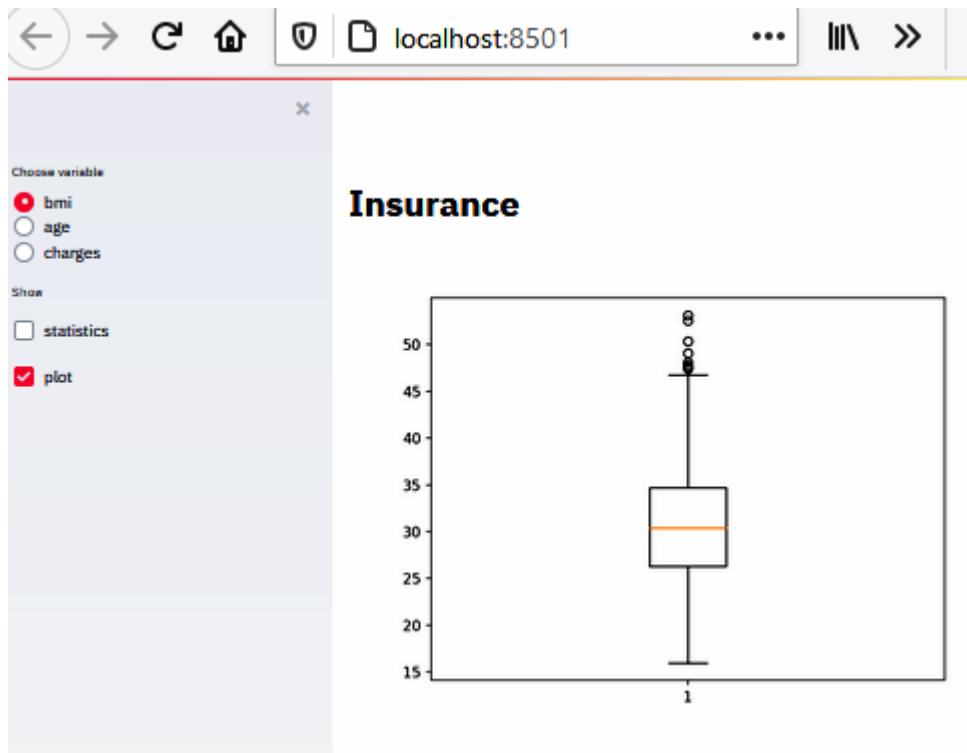


Рис. 3.12. Выбор отображения графика для столбца данных bmi

Создадим новое веб-приложение `example_5.py`. В интерфейсе пользователя для таблицы данных `insurance` включим возможность выбирать факторный показатель для построения графика `boxplot()` по переменным `age`, `bmi`, `charges` отдельно по каждому из значений факторной переменной. Таблица с двумя выбранными показателями (факторным и количественным) по желанию пользователя будет отображаться на экране.

example_5.py

```
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt

st.title('Insurance')
df = pd.read_csv("insurance.csv")
variable_1 = st.sidebar.radio('Choose a variable', ('bmi', 'age', 'charges'))
variable_2 = st.sidebar.radio('Choose a grouping variable', ('sex', 'smoker'))
st.sidebar.text('Show')
new_df = df[[variable_1, variable_2]]

y = df[variable_2].unique()
X1 = new_df[df[variable_2] == y[0]]
X2 = new_df[df[variable_2] == y[1]]
X1 = X1.iloc[:, 0]
X2 = X2.iloc[:, 0]
fig, ax = plt.subplots()
ax.boxplot((X1, X2), labels = [y[0], y[1]])
```

```
if (st.sidebar.checkbox('data')):
    st.write(new_df)
if (st.sidebar.checkbox('plot')):
    st.write(st.pyplot(fig))
```

Веб-страница с возможностью выбора факторов *smoker* и *sex* для построения графика `boxplot()` по одной из переменных *age*, *bmi*, *charges* представлена на рис. 3.13–3.15.

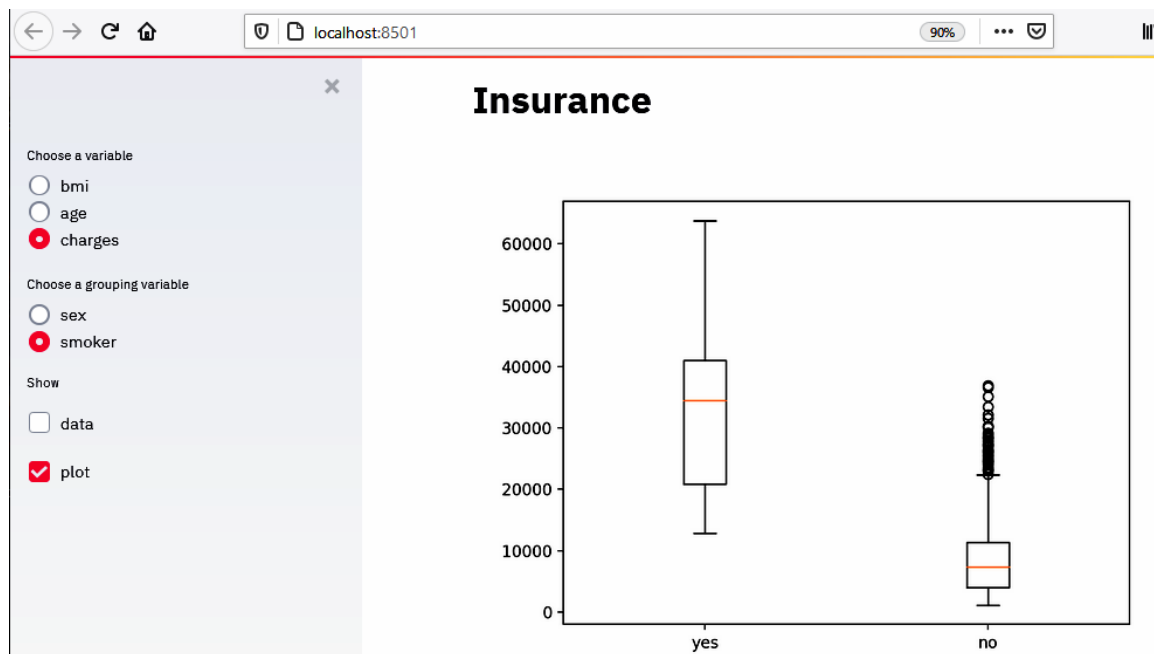


Рис. 3.13. Визуализация группировки расходов по значению переменной *наличие вредной привычки (smoker)*

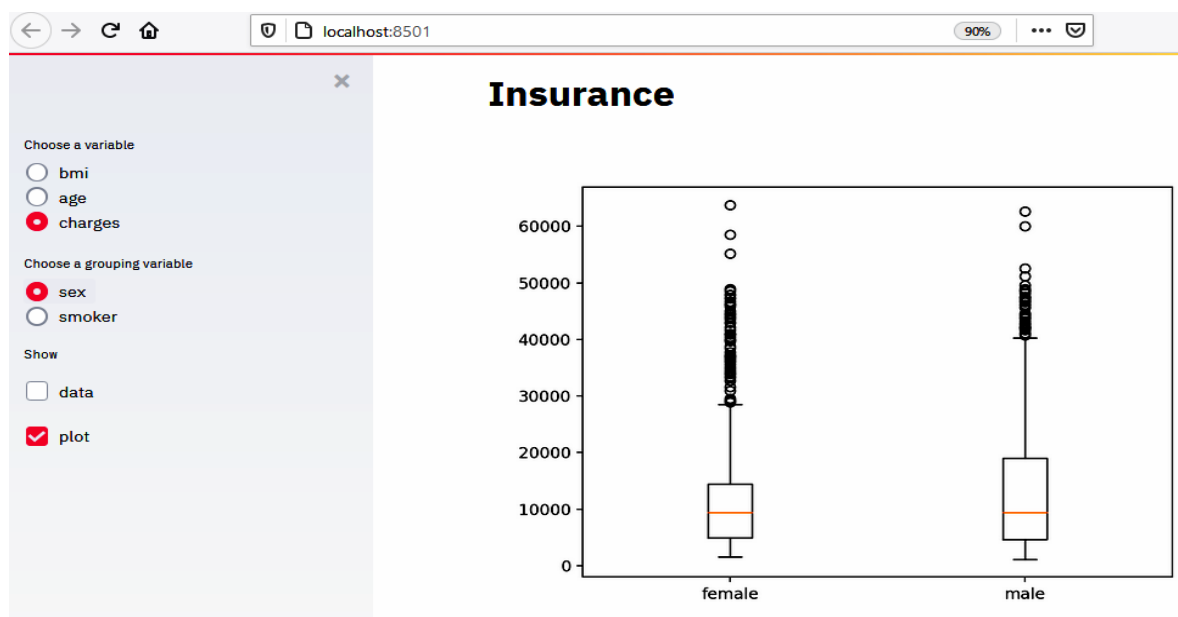


Рис. 3.14. Визуализация группировки расходов по значению переменной *пол (sex)*



Рис. 3.15. Отображение совместно с графиком таблицы данных с двумя выбранными показателями *пол (sex)* и *расходы (charges)*

3.4. Построение карт

Проведем визуализацию геоданных. Рассмотрим файл CV_LatLon_21Jan_12Mar.csv, который представляет собой агрегированную версию набора данных 2019-nCoV, собранного Университетом Джона Хопкинса (URL: <https://www.kaggle.com/greublin/coronavirus-latlon-dataset>). Из этого файла нам понадобятся данные о широте и долготе (столбцы 'lat' и 'lon'). Файл переименован в LatLon.csv и перемещен в папку, в которой находятся все файлы приложений streamlit.

example_6.py

```
import streamlit as st
import pandas as pd
```

```
# Loading the data for showing visualization spread of covid on the world map.
plotData = pd.read_csv("LatLon.csv")
```

```

st.write(plotData)
Data = pd.DataFrame()
Data['lat'] = plotData['lat']
Data['lon'] = plotData['lon']
st.subheader("2019-nCoV, 21Jan_12Mar")
st.map(Data, zoom = 14)

```

Веб-страница с отображением точек геоданных на карте мира из файла LatLon.csv с помощью функции `streamlit.map()` представлена на рис. 3.16. Карту мира можно перемещать в окне отображения, увеличивая/уменьшая масштаб (рис. 3.17).

	country	lat	lon	date	confirmed	recovered	de
0	Thailand	15	101	1/22/20	2	0	^
1	Japan	36	138	1/22/20	2	0	
2	Singapore	1.2833	103.8333	1/22/20	0	0	
3	Nepal	28.1667	84.2500	1/22/20	0	0	
4	Malaysia	2.5000	112.5000	1/22/20	0	0	
5	Canada	49.2827	-123.1207	1/22/20	0	0	
6	Australia	-33.8688	151.2093	1/22/20	0	0	
7	Australia	-37.8136	144.9631	1/22/20	0	0	
8	Australia	-28.0167	153.4000	1/22/20	0	0	
9	Cambodia	11.5500	104.9167	1/22/20	0	0	v
10							< >

2019-nCoV, 21Jan_12Mar

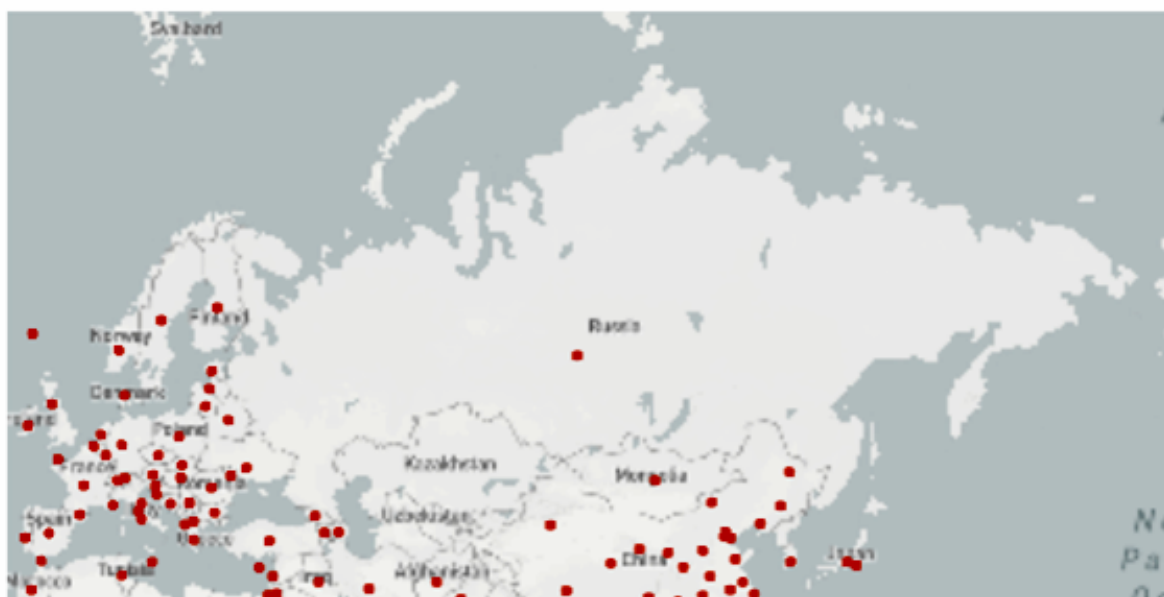


Рис. 3.16. Отображение карты мира с геоданными и таблицы данных

2019-nCoV, 21Jan_12Mar



Рис. 3.17. Изменение области видимости на карте и масштаба отображения

3.5. Загрузка данных

Создадим новое приложение `example_7.py` с загрузкой файлов по выбору пользователя и отображением их содержимого на странице.

example_7.py

```
import streamlit as st
import pandas as pd
uploaded_file = st.file_uploader("Choose a file .csv", type='csv')
dataframe = pd.read_csv(uploaded_file)
st.write(dataframe)
```

Веб-страница с возможностью загрузки файла представлена на рис. 3.18.

Choose a file .csv

insurance.csv

[browse files](#)

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.9000	0	yes	southwest	16,884.92
1	18	male	33.7700	1	no	southeast	1,725.55
2	28	male	33	3	no	southeast	4,449.46
3	33	male	22.7050	0	no	northwest	21,984.47
4	32	male	28.8800	0	no	northwest	3,866.85
5	31	female	25.7400	0	no	southeast	3,756.62
6	46	female	33.4400	1	no	southeast	8,240.58
7	37	female	27.7400	3	no	northwest	7,281.50
8	37	male	29.8300	2	no	northeast	6,406.41
9	60	female	25.8400	0	no	northwest	28,923.13
10							

Рис. 3.18. Загрузка файла insurance.csv

Щелчок на кнопке 'browse files' позволяет загрузить новый файл (рис 3.19). Так как задан параметр type='csv', то для загрузки отображаются только файлы с заданным расширением (рис. 3.20).

Choose a file .csv

travel.csv

[browse files](#)

	Agency	Agency Type	Distribution Channel	Product Name	Clai
0	CBH	Travel Agency	Offline	Comprehensive Plan	
1	CBH	Travel Agency	Offline	Comprehensive Plan	
2	CWT	Travel Agency	Online	Rental Vehicle Excess _	
3	CWT	Travel Agency	Online	Rental Vehicle Excess _	
4	CWT	Travel Agency	Online	Rental Vehicle Excess _	
5	JZI	Airlines	Online	Value Plan	
6	CWT	Travel Agency	Online	Rental Vehicle Excess _	
7	CWT	Travel Agency	Online	Rental Vehicle Excess _	
8	CWT	Travel Agency	Online	Rental Vehicle Excess _	
9	CWT	Travel Agency	Online	Rental Vehicle Excess _	
10					

Рис. 3.19. Загрузка файла travel.csv

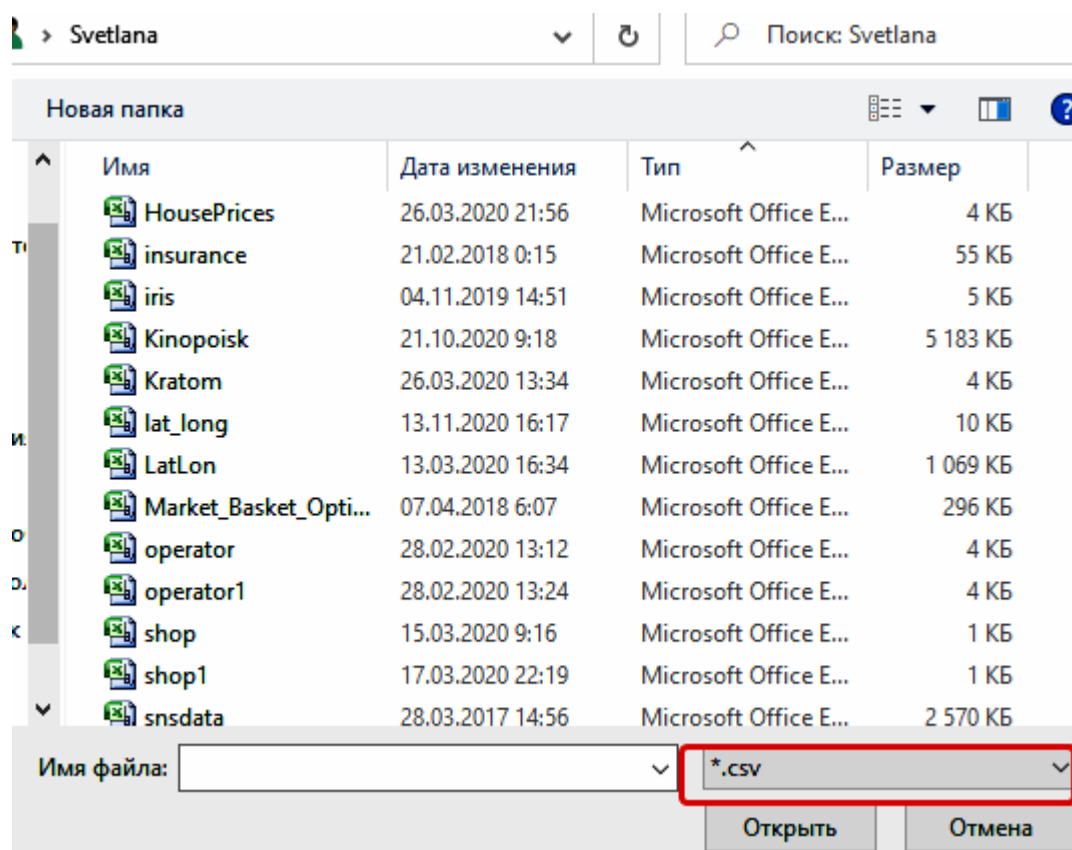


Рис. 3.20. Выбор файла для загрузки

Описание функции для загрузки файлов приведем в табл. 3.4.

Таблица 3.4

Функция для загрузки файлов

Функция	Назначение	Параметр	Назначение
streamlit.file_uploader (label, type=None, accept_multiple_ files=False, key=None, **kwargs)	Отображение виджета загрузчика файлов. По умолчанию загруженные файлы ограничены размером 200 МБ	label	Метка, объясняющая пользователю, для чего предназначен этот загрузчик файлов
		type (str or list of str or None)	Массив разрешенных расширений ['png', 'jpg'], по умолчанию нет, это означает, что все расширения разрешены
		accept_multiple_files (bool)	Если True, то позволяет пользователю загружать несколько файлов одновременно, и в этом случае возвращаемым значением будет список файлов. По умолчанию: False

Изменим приложение example_7.py, чтобы пользователь мог выбрать: отображать ли только первые десять строк файла или сразу весь загруженный файл.

example_7.py

```
import streamlit as st
import pandas as pd
uploaded_file = st.file_uploader("Choose a file .csv", type='csv')
dataframe = pd.read_csv(uploaded_file)

if (st.sidebar.checkbox('data')):
    st.write(dataframe)
if (st.sidebar.checkbox('data.head')):
    st.write(dataframe.head(10))
```

Новый вид приложения с дополнительными возможностями представлен на рис. 3.21–3.23.

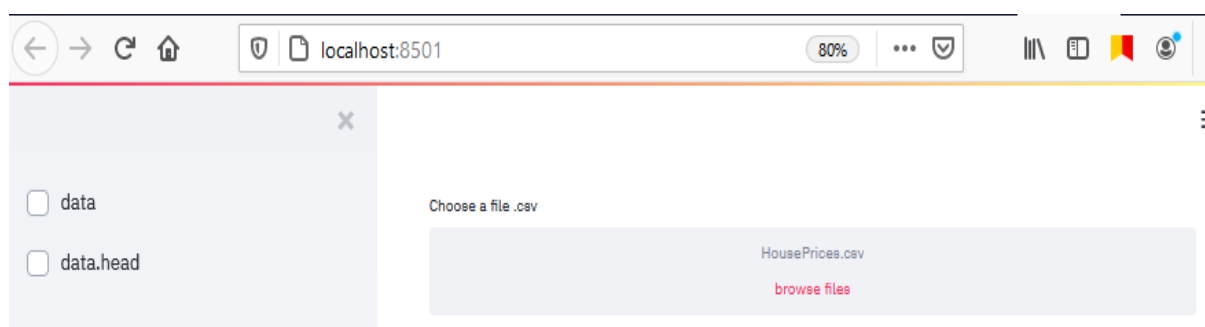
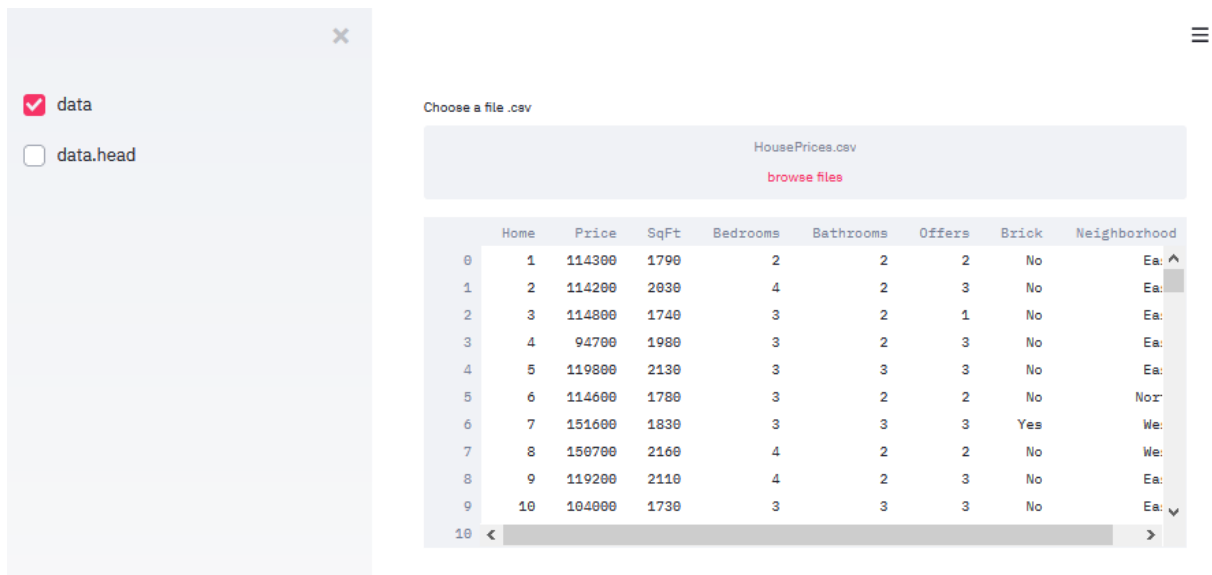


Рис. 3.21. Загрузка файла HousePrices.csv

	Home	Price	SqFt	Bedrooms	Bathrooms	Offers	Brick	Neighborhood
0	1	114300	1790	2	2	2	No	East
1	2	114200	2030	4	2	3	No	East
2	3	114800	1740	3	2	1	No	East
3	4	94700	1980	3	2	3	No	East
4	5	119800	2130	3	3	3	No	East
5	6	114600	1780	3	2	2	No	North
6	7	151600	1830	3	3	3	Yes	West
7	8	150700	2160	4	2	2	No	West
8	9	119200	2110	4	2	3	No	East
9	10	104000	1730	3	3	3	No	East

Рис. 3.22. Отображение первых десяти наблюдений



The screenshot shows a Streamlit web application interface. On the left, there is a sidebar with two checkboxes: 'data' (checked) and 'data.head' (unchecked). The main area displays a file upload section with the text 'Choose a file .csv' and a button 'browse files'. Below this, a table titled 'HousePrices.csv' is shown, displaying 10 rows of data. The table has columns: Home, Price, SqFt, Bedrooms, Bathrooms, Offers, Brick, and Neighborhood. The data is as follows:

	Home	Price	SqFt	Bedrooms	Bathrooms	Offers	Brick	Neighborhood
0	1	114300	1790	2	2	2	No	Ea:
1	2	114200	2030	4	2	3	No	Ea:
2	3	114800	1740	3	2	1	No	Ea:
3	4	94700	1980	3	2	3	No	Ea:
4	5	119800	2130	3	3	3	No	Ea:
5	6	114600	1780	3	2	2	No	Nor
6	7	151600	1830	3	3	3	Yes	We:
7	8	150700	2160	4	2	2	No	We:
8	9	119200	2110	4	2	3	No	Ea:
9	10	104000	1730	3	3	3	No	Ea:

Рис. 3.23. Отображение всего массива данных

В сети есть примеры проектов с использованием библиотеки streamlit [21, 22].

3.6. Лабораторная работа «Создание веб-приложения на языке Python»

Цель работы: научиться создавать реактивные приложения на основе данных.

Формируемые знания, умения и навыки: знать основные возможности библиотеки streamlit для создания веб-приложений на языке Python; получить навыки создания приложений на языке Python.

Необходимо:

1. С сайта <https://www.kaggle.com/> импортировать один из наборов.
2. Используя возможности библиотеки streamlit на языке Python, создать веб-приложение с возможностью выбора данных для анализа, с различными вариантами визуализации.

Контрольные вопросы и задания

1. Как отобразить в приложении таблицу данных?
2. Как отобразить в приложении график?
3. Как отобразить в приложении текст?
4. Какие виджеты есть в библиотеке streamlit, для чего они предназначены?
5. Какие библиотеки поддерживает streamlit?

СПИСОК ЛИТЕРАТУРЫ

1. Few Stephen Visual Business Intelligence. A blog by Stephen Few. URL: <http://www.perceptualedge.com/blog/> (дата обращения: 10.10.2020).
2. Марафон DataLens. Облачные дашборды. URL: <https://datayoga.ru/datalens02> (дата обращения: 10.10.2020).
3. Гештальт и его базовые принципы. Сайт Пси-фактор. URL: <https://psyfactor.org/hist/gestalt-principle.htm> (дата обращения: 10.10.2020).
4. Марафон DataLens. Привет, чарт! URL: <https://datayoga.ru/datalens01> (дата обращения: 10.10.2020).
5. Quickstart – Getting around in Power BI service. URL: <https://docs.microsoft.com/en-us/power-bi/consumer/end-user-experience> (дата обращения: 10.10.2020).
6. Официальный сайт для установки Microsoft Power BI Desktop URL: <https://www.microsoft.com/ru-RU/download/details.aspx?id=58494> (дата обращения: 10.10.2020).
7. Power BI documentation. URL: <https://docs.microsoft.com/en-us/power-bi/> (дата обращения: 10.10.2020).
8. Learn Shiny URL: <https://shiny.rstudio.com/tutorial/> (дата обращения: 10.10.2020).
9. R Documentantion. URL: <https://www.rdocumentation.org/packages/shiny/versions/1.5.0> (дата обращения: 10.10.2020).
10. Build Web App with Shiny R. URL: <https://www.listen-data.com/2018/02/shiny-tutorial-r.html> (дата обращения: 10.10.2020).
11. Gallery. URL: <https://shiny.rstudio.com/gallery/> (дата обращения: 10.10.2020).
12. Официальный сайт Anaconda. URL: <https://www.anaconda.com/> (дата обращения: 10.10.2020).
13. Официальный сайт для установки Visual Studio Code. URL: <https://code.visualstudio.com/download> (дата обращения: 10.10.2020).
14. Официальный сайт для установки Sublime Text. URL: <https://www.sublimetext.com/3> (дата обращения: 10.10.2020).
15. Начало работы с библиотекой streamlit. Get started. URL: https://docs.streamlit.io/en/stable/getting_started.html#create-your-first-streamlit-app (дата обращения: 10.10.2020).
16. Описание виджетов библиотеки streamlit. Display interactive widgets. URL: <https://docs.streamlit.io/en/stable/api.html#display-interactive-widgets> (дата обращения: 10.10.2020).
17. Описание библиотеки pandas. URL: <https://pandas.ydata.rg/> (дата обращения: 10.10.2020).

18. Описание библиотеки NumPy. URL: <https://numpy.org/> (дата обращения: 10.10.2020).

19. Описание графических возможностей библиотеки streamlit. Display charts. URL: <https://docs.streamlit.io/en/stable/api.html#display-charts> (дата обращения: 10.10.2020).

20. Описание графических возможностей библиотеки matplotlib. Методы matplotlib.pyplot. URL: https://matplotlib.org/3.1.0/api/_as_gen/matplotlib.pyplot.html (дата обращения: 10.10.2020).

21. Streamlit – разверните модель машинного обучения, не изучая веб-фреймворк. URL: <https://www.machinelearningmastery.ru/streamlit-deploy-a-machine-learning-model-without-learning-any-web-framework-e8fb86079c61/> (дата обращения: 10.10.2020).

22. Учебный проект на Python: интерфейс в 40 строк кода. Ч. 2. URL: <https://habr.com/ru/company/skillfactory/blog/509340/> (дата обращения: 10.10.2020).

Учебное издание

Рындина Светлана Валентиновна

**БИЗНЕС-АНАЛИТИКА НА ОСНОВЕ БОЛЬШИХ ДАННЫХ:
СОЗДАНИЕ ПРИЛОЖЕНИЙ НА ЯЗЫКАХ PYTHON И R**

Редактор *Н. А. Сидельникова*
Технический редактор *Р. Б. Бердникова*
Компьютерная верстка *Р. Б. Бердниковой*

Подписано в печать 07.04.2021. Формат 60×84¹/₁₆

Усл. печ. л. 4,41.

Заказ № 105. Тираж 3.

440026, Пенза, Красная, 40. Издательство ПГУ
Тел.: (8412) 66-60-49, 66-67-77; e-mail: iic@pnzgu.ru