ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

Федеральное государственное образовательное бюджетное учреждение высшего профессионального образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ им. проф. М. А. БОНЧ-БРУЕВИЧА»

Ф.В. Филиппов

ТЕХНОЛОГИИ ОБРАБОТКИ ИНФОРМАЦИИ

Пособие для практических занятий

САНКТ-ПЕТЕРБУРГ
2015

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель настоящего пособия состоит в том, чтобы познакомить студентов с основами использования современных сред обработки информации и разработки веб приложений. Язык R и среда RStudio первоначально являвшимися основным средством статистической обработки, постепенно завоевывают прочное место среди ИТ-специалистов, занимающихся всесторонним анализом данных и разработкой интерактивных информационных систем. В частности, систем ориентированных на веб технологии использующие реактивные объекты.

Основная задача пособия заключается в развитии у студентов практических навыков использования эффективных программных пакетов обработки данных и разработки интерактивных веб приложений, удовлетворяющих самым высоким требованиям сегодняшнего дня. В этой связи представленные материалы отражают последние достижения в указанной сфере. Такие программные пакеты, как Shiny и knitr являются высокоэффективными средствами обеспечивающими технологии создания информационных веб приложений на базе реактивных выражений.

Пособие включает необходимый методический материал для подготовки к практическим занятиям, а также контрольные задания для закрепления изученного материала. Для успешного выполнения контрольных заданий рекомендуется по мере знакомства с материалом выполнять описываемые команды и анализировать результаты их выполнения. Команды выделены в тексте следующим образом:

```
> команда
```

Пособие включает три основных раздела. В первом разделе описывается среда разработки RStudio.

Второй раздел посвящен описанию использования популярных технологий *data mining* для обработки различных информационных ресурсов с целью поиска, выделения и визуализации данных.

Наконец, третий раздел описывает технологии создания интерактивных веб приложений, в частности на базе реактивных выражений.

Контрольные задания размещены в конце основных разделов пособия и содержат ссылки на ресурсы интернета, которые необходимы для их выполнения.

1. Среда разработки RStudio

1.1. Язык программирования R

R является свободно распространяемым программным обеспечением (язык программирования и среда разработки) для решения широкого класса задач в различных областях научных исследований. Современной мощной средой разработки на основе R служит RStudio, которая доступна для Windows, Linux и MacOS.

R широко используется в подавляющем большинстве университетов мира в учебном процессе, в научных кругах для проведения исследований, а также в бизнес приложениях.

В настоящее время репозитарий CRAN (Comprehensive R Archive Network) для языка программирования R включает более шести тысяч доступных пакетов. Их количество постоянно растет, а назначение расширяется, по мере роста задач в различных научных областях. Полную информация о доступных пакетах всегда можно получить на [1].

Для первого знакомства с R весьма полезна информация из [2, 3]. Более подробная информация доступна на [4], а веб-сайт [5] содержит ссылки на некоторые онлайн документы для R.

R постоянно меняется - как в плане возможностей самого языка, так и платформы. Обновления, содержащие новые и пересмотренные пакеты появляются несколько раз в неделю. Чтобы оставаться в курсе этих многочисленных изменений нужно периодически обращаться к некоторым интернет-ресурсам, которые постоянно информируют о том, что происходит в мире R.

Конечно основой является сайт [5], а на сайте [6] собирается информация обо всех новых и обновленных пакетах, и содержатся ссылки на CRAN для каждого из них. Еще следует отметить сайт планета R [7], который является отличным агрегатором, и включает в себя информацию из широкого диапазона источников. Сайт [8] является центральным узлом (блог агрегатором) для сбора контента от блоггеров, пишущих о R. На нем публикуется несколько новых статей каждый день – это отличное пособие для изучения новых методов из области аналитики данных и программирования.

Наконец, существенную помощь можно получить на сайте [9], где содержится основной список рассылки и который является лучшим местом, чтобы задать вопросы о R.

1.2. Инсталляция R и RStudio

Инсталляторы R и RStudio доступны для всех распространенных операционных систем, в частности Windows XP/Vista/7, Mac OS X 10.5, Debian 6+/Ubuntu 10.04, Linux и Fedora 13.

Инсталляцию лучше всего производить следующим образом. Сначала обратитесь на официальный сайт [10] проекта RStudio. Там имеется последняя версия RStudio и указание на то, какая для нее минимальная версия R требуется. Именно эту, или более позднюю версию R, следует сначала инсталлировать.

Далее, можно установить версию RStudio, которая представляет собой бесплатную интегрированную среду разработки для R. Благодаря ряду своих особенностей этот активно развивающийся программный продукт делает работу с R очень удобной.

Инсталляция происходит стандартно, без каких-либо особенностей. После того, как среда разработки R и RStudio установлены, можно погружаться в изучение их интересных и весьма полезных для ИТ-специалиста возможностей.

Отметим, что все примеры представленные в данном учебном пособии апробированы в следующей среде: версия R 3.1.1 GUI 1.65 Snow Leopard и версия RStudio 0.98.1056.

1.3. Интерфейс среды разработки

Целью этого раздела является знакомство с языком R и RStudio на примере использования фундаментальных понятий: интерфейс среды разработки, чтение данных и основные команды обработки данных.

Интерфейс состоит из четырех панелей (рис. 1).

Панель в правом верхнем углу содержит две закладки - рабочее пространство (Environment) и история команд (History). По закладке Environment содержится информация обо всех созданных в процессе работы объектах. По закладке History, можно посмотреть все команды, которые вводились и исполнялись в процессе работы.

Панель в правом нижнем углу содержит четыре закладки: Files, Plots, Packages и Help. По закладке Files осуществляется доступ к файловой системе компьютера. Пространство закладки Plots используется для вывода графиков и изображений. По закладке Packages осуществляется доступ к любому пакету системной библиотеки R. Двойной щелчок по названию пакета осуществляет доступ к описанию всех его функций, а щелчок по иконке Install позволяет выбрать и инсталлировать из интернета любой отсутствующий пакет. Наконец закладка Help служит для вывода справочной информации.

Панель слева состоит из двух частей.

Верхняя часть панели слева называется Source и служит для размещения

кода скриптов, которые подлежат отладке и исполнению. Кроме того, там можно посмотреть содержимое некоторых объектов, например фреймов, имена которых в панели Environment снабжены специальными иконками (справа).

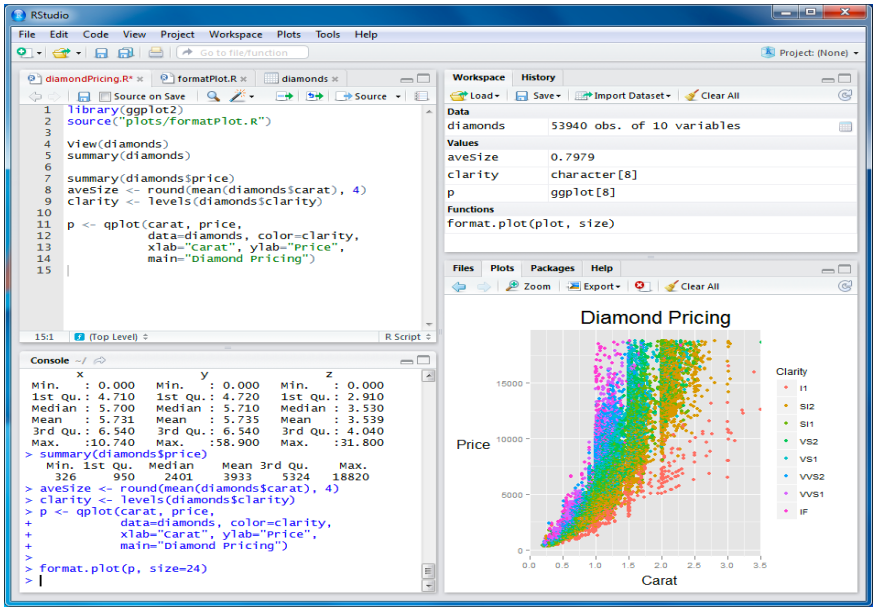


Рис.1. Панели среды разработки RStudio

Нижняя часть панели слева называется консоль. Каждый раз при запуске RStudio на ней будет указана версия языка R и его краткая характеристика. Ниже находится приглашение (командная строка), где вводятся команды для R интерпретатора. Эти команды и их синтаксис развивались в течение десятилетий и в настоящее время обеспечивают то, что огромное число пользователей во всем мире считает очень эффективным способом доступа к данным, их организации, анализа, визуализации и всего того, что в настоящее время называют data mining.

Для начала введите в командную строку следующую команду, не забудьте нажать клавишу Enter для запуска команды:

```
> source("http://www.openintro.org/stat/data/present.R")
```

По команде интерпретатор R получит доступ к сайту www.openintro.org и загрузит набор данных. При этом в рабочей зоне, в правом верхнем углу окна RStudio, отобразится характеристика этого набора данных с именем present (рис. 2).

Набор данных сохранится в таблице с именем present. Формат хранения – фрейм, объем данных 2,4 килобайта, в таблице 3 столбца (variables) и 63 строки (observations).

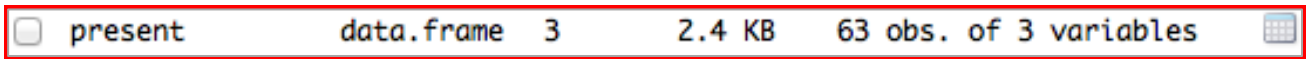


Рис.2. Характеристика набора данных present

Данные Present

Настоящий набор данных содержит количество рождений мальчиков и девочек в США с 1940 по 2002 год. Можно взглянуть на данные прямо в консоли, введя команду present (рис. 3).

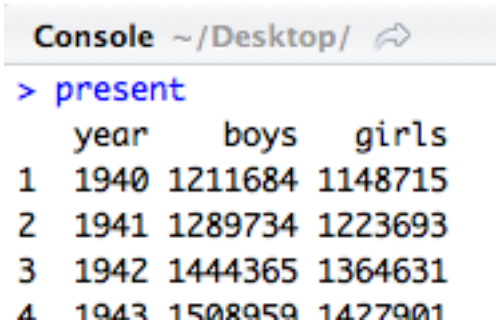


Рис. 3. Представление данных в консоли

Для изучения полного набора данных служит полоса прокрутки на правой стороне окна консоли. Обратите внимание, что номера строк в первом столбце не являются частью настоящего массива данных. Интерпретатор R добавляет их для удобства. Можно рассматривать их как индекс. В самом деле, сравнение с электронной таблицей обычно может быть полезным. Интерпретатор R сохранил данные в своего рода таблицу называемую фреймом. Можно узнать размерность фрейма, введя:

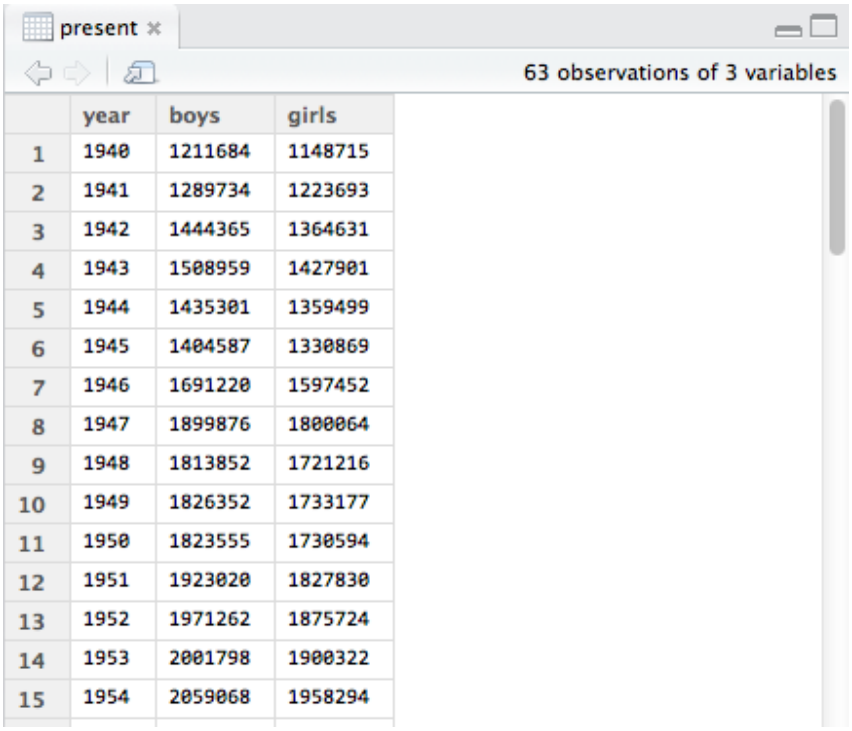
```
> dim(present)
```

Эта команда должна вывести [1] 63 3, указывая, что во фрейме - 63 строки и 3 колонки (столбца). Увидеть имена этих столбцов (или переменных), можно введя команду:

```
> names(present)
```

Видно, что фрейм данных содержит колонки year, boys и girls. На этом этапе можно заметить, что команды в R похожи на математические функции, и вызов R команд означает выполнение функции с некоторым числом аргументов. Команды dim и names, например, имеют единственный аргумент - название фрейма данных.

Одним из преимуществ RStudio является встроенныё просмотрщик (viewer) данных. Посмотреть содержимое фрейма можно нажав на иконку расположенную справа от характеристики переменной на рис. 2. Содержимое фрейма появится на левой верхней панели (рис. 4).



	year	boys	girls
1	1940	1211684	1148715
2	1941	1289734	1223693
3	1942	1444365	1364631
4	1943	1508959	1427901
5	1944	1435301	1359499
6	1945	1404587	1330869
7	1946	1691220	1597452
8	1947	1899876	1800064
9	1948	1813852	1721216
10	1949	1826352	1733177
11	1950	1823555	1730594
12	1951	1923020	1827830
13	1952	1971262	1875724
14	1953	2001798	1900322
15	1954	2059068	1958294

Рис. 4. Содержимое фрейма present

Обратите внимание, что при клике по иконке, в командной строке автоматически сформировалась команда `> View(present)`. Закрыть просмотр данных можно, нажав на «x» в верхнем левом углу.

Небольшое исследование

Начнем изучать данные более внимательно. Можно получить доступ отдельно к данным из одной колонки фрейма, используя команду:

```
> present$boys
[1] 1211684 1289734 1444365 1508959 1435301 1404587 1691220 1899876 1813852 1826352 1823555
[12] 1923020 1971262 2001798 2059068 2073719 2133588 2179960 2152546 2173638 2179708 2186274
[23] 2132466 2101632 2060162 1927054 1845862 1803388 1796326 1846572 1915378 1822910 1669927
[34] 1608326 1622114 1613135 1624436 1705916 1709394 1791267 1852616 1860272 1885676 1865553
[45] 1879490 1927983 1924868 1951153 2002424 2069490 2129495 2101518 2082097 2048861 2022589
[56] 1996355 1990480 1985596 2016205 2026854 2076969 2057922 2057979
```

Эта команда показывает количество только мальчиков по годам. Обратите внимание, что интерпретатор R напечатал эти данные в другом формате. Теперь данные не структурированы в виде таблицы с другими переменными, поэтому

они отображаются один за другим. Объекты, которые печатаются в этом случае, называются векторами; они представляют собой набор цифр. Интерпретатор R добавил числа в скобках [] в левой части распечатки, чтобы указать местоположение в пределах вектора. Например, за индексом [1] следует 1211684, что указывает на то, что 1211684 является первым элементом в векторе. И если индекс [45] начинает строчку, то это означает - первое число в этой строке будет представлять 45-ю запись в векторе.

Интерпретатор R имеет несколько мощных функций для создания графики. Мы можем создать простой график количества ежегодно рождаемых девочек с помощью команды:

```
> plot(x = present$year, y = present$girls)
```

По умолчанию интерпретатор R создает точечный график, рисуя кружок для каждой пары x, y. График сам по себе должен появиться на вкладке Plots в нижней правой панели RStudio. Обратите внимание, что команда снова выглядит как функция, на этот раз с двумя аргументами, разделенными запятой. Первый аргумент в функции задает переменную для оси абсцисс x, а второй - для оси ординат y. Если нужно, чтобы точки были соединены в линию (рис. 5), можно добавить третий аргумент, буква " l " для линии:

```
> plot(x = present$year, y = present$girls, type = "l").
```

Как же узнать, что можно было добавить третий аргумент. К счастью, в R все функции хорошо задокументированы. Чтобы изучить, что делает функция и узнать о ее аргументах нужно просто ввести вопросительный знак и имя функции. Например, попробуйте следующее:

```
> ?plot
```

Обратите внимание, что файл справки заменяет график в нижней правой панели. Можно переключаться между графиком и справкой (Plots и Help) с использованием вкладок в верхней части этой панели.

Теперь предположим, что нужно построить график общего количества рождений. Чтобы вычислить это, можно сложить:

```
> 1211684 + 1148715
```

чтобы увидеть общее количество рождений в 1940 году и повторить это для каждого года, но есть более быстрый способ.

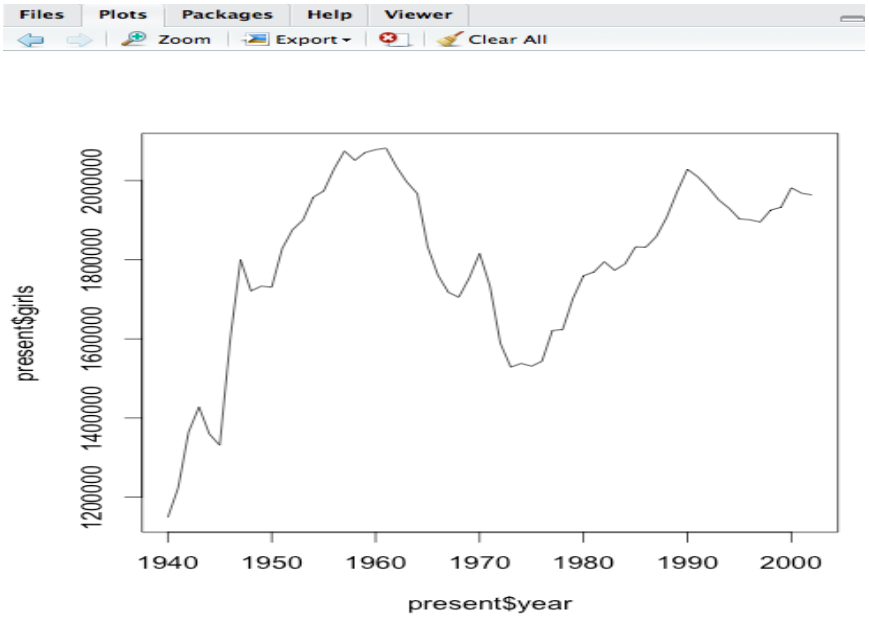


Рис. 5. Линейный график

Если сложить вектора рождения для мальчиков и девочек, интерпретатор R вычисляет все суммы одновременно:

```
> present$boys + present$girls
```

В результате получится 63 числа, каждое из которых представляет собой сумму количества мальчиков и девочек, рожденных в соответствующий год. Можно сделать график общего количества родившихся по годам с помощью команды:

```
> plot(present$year, present$boys + present$girls, type = "l")
```

Обратите внимание, что здесь не указаны имена первых двух аргументов. Потому, что файл справки показывает, что по умолчанию для plot первым аргументом является переменная x, а вторым аргументом - переменная y.

Подобно тому, как рассчитывалась сумма новорожденных мальчиков и девочек, можно вычислить отношение числа мальчиков к числу девочек:

```
> present$boys/present$girls
```

или долю новорожденных мальчиков:

```
> present$boys/(present$boys + present$girls)
```

В дополнение к таким операциям как вычитание и деление, можно выполнять операции сравнения, такие как больше, меньше и другие. Например, мож-

но узнать, превосходят ли мальчики девочек по рождаемости в каждом году с помощью выражения:

```
> present$boys > present$girls
```

Эта команда возвращает 63 значения либо TRUE, если в этом году было больше мальчиков, чем девочек, либо FALSE, если в этом году это не так. Этот вывод показывает данные иного рода, чем мы рассматривали до сих пор. Во фрейме данных present значения численные (год, количество мальчиков и девочек). Здесь, интерпретатор R создал логические данные, где значения являются TRUE или FALSE. В общем, анализ данных будет включать в себя различные типы, и одна из причин для использования интерпретатора R заключается в том, что он способен представлять и выполнять операции вычисления со многими из них.

Наборы данных

В примерах этого раздела использовался набор данных present в формате фрейма. Для знакомства с другими форматами представления данных и способами их обработки удобно использовать встроенные наборы данных. Например, набор данных diamonds, содержит цены и другие атрибуты почти 54 тысяч алмазов. Для выбора подходящего набора можно воспользоваться командой:

```
> library(help = "datasets").
```

Результатом выполнения этой команды является информация по пакету datasets, включающему более ста наборов данных от AirPassengers (ежемесячное число авиапассажиров с 1949 по 1960) до women (средний рост и вес американких женщин). Некоторые из этих наборов будут использованы в контрольных заданиях по разделу.

КОНТРОЛЬНЫЕ ЗАДАНИЯ

Задание 1

Используя фрейм present**,** создайте график доли мальчиков с течением времени, и на основе графика определите, является ли следующее утверждение истинным или ложным: доля мальчиков, родившихся в США , уменьшилась с течением времени.

Совет: с помощью клавиш со стрелками вверх и вниз можно просматривать все предыдущие команды. Также можно получить доступ к истории команд, нажав на вкладку **History** в верхней правой панели. Это позволит сэкономить много времени в будущем.

Задание 2

Используя фрейм present**,** создайте график, который отображает соотношение мальчик-к-девочке за каждый год. Что вы видите? Опишите тенденцию.

Задание 3

Используя фрейм present**,** вычислите абсолютные различия между количеством мальчиков и девочек, родившихся в каждом году, и определите, в каком году была самая большая абсолютная разница в количествах новорожденных девочек и мальчиков?

Задание 4

Используя фрейм diamonds, определите все возможные сочетания качества огранки (cut), цвета (color) и чистоты (clarity) алмазов. Результат представьте в таблице со столбцами: № п/п, cut, color, clarity.

Задание 5

Используя фрейм diamonds, определите 100 самых больших алмазов, с указанием их цены (price), величины (carat) и размеров по трем измерениям (x, y, z).

Задание 6

Используя набор данных nottem, постройте график изменения средней температуры воздуха в городе Ноттенгейм за период 1920-1939 год, отдельно для каждого месяца с января по декабрь.

Задание 7

Используя набор данных precip, определите города США с минимальным и максимальным уровнем годовых осадков.

Задание 8

Используя набор данных pressure, постройте график зависимости давления паров ртути от температуры.

2. Обработка информации

Технология *Data Mining* - это процесс обнаружения в сырых данных ранее неизвестных, нетривиальных, практически полезных и доступных интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности. Суть и цель технологии *Data Mining* можно охарактеризовать так: это технология, которая предназначена для поиска в больших объемах данных *неочевидных*, *объективных* и *полезных* на практике закономерностей.

Неочевидных - это значит, что найденные закономерности не обнаруживаются стандартными методами обработки информации или экспертным путем.

Объективных - это значит, что обнаруженные закономерности будут полностью соответствовать действительности, в отличие от экспертного мнения, которое всегда является субъективным.

Практически полезных - это значит, что выводы имеют конкретное значение, которому можно найти практическое применение.

Настоящий раздел включает описание технологий обработки информации, позволяющих осуществлять анализ и выделение из данных различной природы неявной и неструктурированной информации и представлять ее в виде, пригодном для использования. В основе представленных технологий лежат мощные программные пакеты языка R.

2.1. Импортирование данных

Возможности системы R по вводу и редактированию данных весьма многообразны. Импортирование данных в систему R часто вызывает проблемы у тех, кто только начинает работать с этой программой. Тем не менее, ничего сложного в этом нет. Ниже будут подробно рассмотрены наиболее распространенные способы импорта таблиц данных в рабочую среду R, однако сначала стоит ознакомиться с правилами подготовки загружаемых файлов:

- В импортируемой таблице с данными не должно быть пустых ячеек. Если некоторые значения по тем или иным причинам отсутствуют, вместо них следует ввести NA.
- Импортируемую таблицу с данными рекомендуется преобразовать в простой текстовый файл с одним из допустимых расширений. На практике обычно используются файлы с расширением txt, в которых значения переменных разделены знаками табуляции (*tab-delimited files*), а также файлы с расширением.csv (*comma separated values*), в которых значения переменных разделены запятыми.

- В качестве первой строки в импортируемой таблице рекомендуется ввести заголовки столбцов-переменных. Такая строка – удобный, но не обязательный элемент загружаемого файла. Если она отсутствует, то об этом необходимо сообщить в описании команды, которая будет управлять загрузкой файла (например, read.table() – см. ниже). Все последующие строки файла в качестве первого элемента содержат заголовки строк (если таковые предусмотрены), после которых следуют значения каждой из имеющихся в таблице переменных. В именах столбцов таблицы не допускается наличие пробелов. Кроме того, имена столбцов (так же как и имена строк) не должны начинаться с точки или чисел. Во избежание связанных с кодировкой проблем все текстовые величины в импортируемых файлах рекомендуется создавать с использованием букв латинского алфавита.
- Подлежащий импортированию файл рекомендуется поместить в т.н. рабочую папку программы, т.е. папку, в которой интерпретатор R по умолчанию будет "пытаться найти" этот файл. Чтобы выяснить путь к рабочей папке R на своем компьютере используйте команду getwd() (*get working directory* – узнать рабочий директорий), например:

```
> getwd()
[1] "C:/Temp/"
```

Изменить рабочий директорий можно при помощи команды **setwd**() (*set working directory* – создать рабочий директорий):

```
> setwd("C:/MyDocuments/")
# при выполнении приведенной команды внешне ничего не произойдет,
# однако последующее применение команды getwd() покажет,
# что путь к рабочей папке изменился:
> getwd()
[1] "C:/My Documents/"
```

Ниже приведен фрагмент типичной таблицы данных (табл. 1), которая может быть успешно загружена для анализа в среду R. Используйте этот фрагмент в качестве образца при оформлении своих таблиц с данными.

Таблица 1. Фрагмент типичной таблицы данных

	Group	Variable1	Variable2	Variable3
Ivan	A	102	1.3	14
Vitaliy	A	98	1.4	11
Sergey	B	45	NA	8
Mikhail	B	50	3.2	6

Как видим, приведенный фрагмент имеет размерность 5x5, т.е. состоит из пяти строк и пяти столбцов. В первой строке представлены заголовки всех имеющихся в таблице столбцов, за исключением первого. Первый столбец, хотя и не имеет собственного заголовка, не является пустым – он содержит имена добровольцев, участвовавших в некотором эксперименте (Ivan, Vitaliy, и т.д.). Второй столбец имеет заголовок Group и содержит метки, по которым можно выяснить принадлежность испытуемых к той или иной экспериментальной группе (A, B, и т.д.). В терминах языка R переменная Group называется фактором. В последующих столбцах (с заголовками Variable1,Variable2, и т.д.) содержатся значения измеренных в ходе исследования переменных. В приведенном фрагменте таблицы имеется одно отсутствующее значение, вместо которого введено NA.

Пожалуй, одним из наиболее доступных и удобных средств подготовки данных для их последующего анализа при помощи R, является программа Microsoft Excel. Для сохранения Excel-таблиц в виде txt- или csv-файлов используйте опцию *Сохранить как* (*Save as*) в разделе *Файл* (*File*) главного меню этой программы.

2.1.1. Функция read.table()

Основной функцией для импортирования данных в рабочую среду R является read.table(). Эта мощная функция позволяет достаточно тонко настроить процесс загрузки внешних файлов, в связи с чем она имеет большое количество управляющих аргументов. Наиболее важные из этих аргументов перечислены ниже в таблице (табл. 2). Подробнее смотри файл помощи, доступный в RStudio по команде ?read.table.

Таблица 2. Аргументы функции read.table()

Аргумент	Назначение
file	Служит для указания пути к импортируемому файлу. В качестве имени можно также указывать полную URL-ссылку на файл, который предполагается загрузить из интернета . Начиная с версии R 2.10, появилась возможность импортировать архивированные файлы в zip-формате.
header	Служит для сообщения программе о наличии в загружаемом файле строки с заголовками столбцов. По умолчанию принимает значение FALSE. Если строка с заголовками столбцов имеется, этому аргументу следует присвоить значение TRUE

Окончание табл. 2

Аргумент	Назначение
row.names	Служит для указания номера столбца, в котором содержатся имена строк (например, в рассмотренном выше примере это был первый столбец, поэтому row.names = 1). Важно помнить, что все имена строк должны быть уникальными, т.е. одинаковые имена для двух или более строк не допускаются.
sep	Служит для указания используемого в файле разделителя значений переменных (*separator* – разделитель). По умолчанию предполагается, что значения переменных разделены "пустым пространством", например, в виде пробела или знака табуляции (sep = ""). В файлах формата csv значения переменных разделены запятыми, и поэтому для них sep = ",".
dec	Служит для указания знака, используемого в файле для отделения целой части числа от дроби. По умолчанию sep = ".". Однако во многих странах в качестве десятичного знака применяют запятую, о чем важно вспомнить перед загрузкой файла и, при необходимости, использовать dec = ",".
nrows	Выражается целым числом, указывающим количество строк, которое должно быть считано из загружаемой таблицы. Отрицательные и иные значения игнорируются. Пример: nrows = 100.
skip	Выражается целым числом, указывающим количество строк в файле, которое должно быть пропущено перед началом импортирования. Пример: skip = 5

Для загрузки тщательно подготовленных файлов (см. правила выше) достаточно использовать минимальный набор аргументов функции read.table(). В качестве примера предположим, что нам необходимо загрузить файл new_data.txt, который хранится в рабочей папке R. Загружаемую таблицу данных мы намерены сохранить в виде объекта с именем data_1. Функции read.table() в этом случае может быть применена следующим образом:

```
> data_1 = read.table(file = "new_data.txt", header = TRUE)
```

Как отмечено выше, часто импортируемые в R файлы имеют формат csv. Для их загрузки можно воспользоваться той же функцией read.table(), но при этом следует указать, что в качестве разделителя значений переменных в файле используется запятая:

```
> data_1 = read.table(file = "new_data.csv", header = TRUE, sep = ",")
```

2.1.2. Функция read.csv()

Аналогом функции read.table() для считывания csv-файлов является функция read.csv():

```
> data_1 = read.csv(file = "new_data.csv", header = TRUE)
```

Если подлежащий загрузке файл хранится в папке, отличной от рабочей папки R, то следует указать полный путь к нему. При этом пользователям операционных систем Windows необходимо помнить, что для указания полных путей к файлам в программе R используется не обратный одинарный слеш (\), а прямой одинарный (/) либо двойной обратный слеш (\\). Например, следующие две команды будут успешно восприняты R и приведут к идентичному результату – загрузке файла new_data.txt и сохранению его в виде объекта data_1:

```
> data_1= read.csv(file = "D:\\Documents\\data_1.txt", header = TRUE)
> data_1 = read.csv(file = "D:/Documents/data_1.txt", header = TRUE)
```

Для интерактивного выбора загружаемого файла, который хранится вне рабочей папки R, можно применить вспомогательную функцию file.choose() (*выбрать файл*). Выполнение этой команды приводит к открытию обычного диалогового окна операционной системы Windows, в котором пользователь выбирает папку с необходимым файлом. Очень удобно совмещать file.choose() с командами read.table() или read.csv(), например:

```
> data_1 = read.table(file = file.choose(), header = TRUE, sep = ",")
```

2.2. Фреймы

Изучая интерфейс среды разработки, мы частично познакомились с понятием фрейма. Исходя из того, что эти объекты чрезвычайно часто используются при обработке данных рассмотрим их более детально.

Фреймы используются для хранения данных в виде таблиц. Они представляют собой набор векторов одинаковой длины, причем типы данных каждого вектора могут быть разными. Например, сформируем три вектора a, b и c:

```
> a = c(1, 2, 3)
> b = c(“x”, ”y”, ”z”)
> c =  c(FALSE, TRUE, FALSE)
```

и с помощью команды data.frame() создадим фрейм df:

```
> df = data.frame(a, b. c)
> df
   a  b  c
1  1  x  FALSE
2  2  y  TRUE
3  3  z   FALSE
```

Для демонстрации основных приемов работы с фреймами используем встроенный в R фрейм с именем mtcars. Посмотрим три первые строки фрейма mtcars:

```
> mtcars[1:3,]
                  mpg cyl disp  hp drat      wt  qsec  vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0    1      4     4
Mazda RX4 Wag  21.0   6  160 110 3.90 2.875 17.02  0    1      4     4
Datsun 710        22.8   4  108   93 3.85 2.320 18.61  1    1      4     1
```

Верхняя строка таблицы, называемая заголовок (header), содержит имена столбцов фрейма. Для доступа к различным ячейкам фрейма можно использовать их координаты [строка,столбец] в квадратных скобках. Координаты можно задавать как номерами соответствующей строки и столбца:

```
> mtcars[2,6]
[1] 2.875
```

так и с помощью их имен:

```
> mtcars["Mazda RX4 Wag","wt"]
[1] 2.875
```

Число строк и столбцов фрейма можно узнать с помощью следующих двух команд:

```
> nrow(mtcars)
[1] 32
> ncol(mtcars)
[1] 11
```

Видно, что встроенный в R фрейм с именем mtcars имеет размерность 32x11. Подробную информацию о данных, размещенных во фрейме mtcars можно получить с помощью команды: > help(mtcars).

2.2.1. Доступ к столбцам фрейма

Существует много способов доступа к векторам, размещенным в столбцах фрейма. Например вектор в столбце cyl можно получить с помощью любой из команд: mtcars$cyl, mtcars[[2]], mtcars[["cyl "]] или mtcars[,"cyl "], например:

```
> mtcars[[2]]
 [1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 8 6 8 4
```

В этом случае, вектор представляется как массив значений. Если необходимо получить вектор столбца с указанием его имени и имени строк, можно использовать любую из следующих команд: mtcars[2], mtcars[“cyl”], например:

```
> mtcars[2]
                  cyl
Mazda RX4           6
Mazda RX4 Wag       6
Datsun 710          4
  . . .
```

Можно получить любой фрагмент фрейма из нескольких интересующих столбцов, используя перечисление их номеров или имен в векторе c() внутри квадратных скобок, например:

```
> mtcars[c(2,7)]
                  cyl   qsec
Mazda RX4           6  16.46
Mazda RX4 Wag       6  17.02
Datsun 710          4  18.61
```

2.2.2. Доступ к строкам фрейма

Можно извлекать строки фрейма также используя одинарные квадратные скобки, но при этом после номера (или имени) интересующей строки нужно обязательно ставить запятую. Например, команды mtcars[19,] и mtcars["Honda Civic",] приведут к одному результату:

```
> mtcars["Honda Civic",]
             mpg cyl  disp hp drat    wt  qsec vs am gear carb
Honda Civic 30.4   4 75.7 52 4.93 1.615 18.52  1  1    4    2
```

Можно извлекать любое сочетание строк фрейма, перечисляя их имена или номера в векторе c(). Следующие команды: mtcars[c(3, 24),] и mtcars[c("Datsun 710", "Camaro Z28"),] приведут к одному результату:

```
> mtcars[c("Datsun 710", "Camaro Z28"),]
             mpg cyl disp  hp  drat    wt  qsec vs am gear carb
Datsun 710  22.8   4  108  93  3.85 2.32 18.61  1  1    4    1
Camaro Z28 13.3   8  350 245 3.73 3.84 15.41  0  0    3    4
```

Существует также способ логической индексации строк, позволяющий выбирать из фрейма только те строки, которые удовлетворяют некоторому условию. Если какой-то из столбцов включает логические значения (0 или 1), указывающие на наличие или отсутствие некоторого свойства объекта, то его можно использовать для логической индексации. Например, сформируем логический вектор AT, принимающий значение TRUE если автомобиль снабжен автоматической коробкой передач и FALSE, в противном случае:

```
> AT = mtcars$am == 0
> AT
 [1] FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[16]  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE
[31] FALSE FALSE
```

На основе вектора AT можно получить перечень всех автомобилей из фрейма mtcars с автоматической коробкой передач, следующим образом:

```
> mtcars[AT,]
                  mpg cyl  disp  hp  drat    wt  qsec vs am gear carb
Hornet 4 Drive   21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
Valiant          18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
…
```

таких автомобилей всего 19. Ниже представлены данные о расходе горючего этих автомобилей (в виде вектора):

```
> mtcars[AT,]$mpg
```

```
[1] 21.4   18.7   18.1   14.3   24.4   . . .
```

2.3. Анализ XML документов

Основным пакетом используемым для анализа XML документов служит пакет XML. Рассмотрим некоторые особенности использования функций xmlName(), xmlSize(), xmlValue(), xmlInternalTreeParse() и xmlRoot() из этого пакета на примере создания пользовательской функции visitNode().

2.3.1. Функция visitNode()

Ниже приведен пример рекурсивной функции visitNode() для анализа XML-дерева и обработки вершин модели DOM:

```
library(XML)
visitNode = function(node) {
 if (is.null(node)) {
  # последняя вершина достигнута, выход
  return()
 }
 print(paste("Node: ", xmlName(node)))
 num.children = xmlSize(node)
 if(num.children == 0 ) {
  print( paste("   ", xmlValue(node)))
 }
 # погружаемся на следующий уровень
 for (i in 1 : num.children) {
  visitNode(node[[i]]) # i-й потомок
 }
}
```

Предварительно загружается необходимый пакет XML. Рекурсия продолжается до тех пор, пока не достигнута последняя вершина дерева node=0 модели DOM. На каждом текущем уровне выводится название вершины (тега) с помощью функции xmlName(node), например "Node: AUTHOR". Если вершина является листом дерева xmlSize(node)=0, то для нее выводится контент, например "Mark Twain".

Рассмотрим использование функции visitNode() для анализа XML документа. Ниже приведена последовательность команд с необходимыми комментариями:

```
# Загружаем имя анализируемого XML файла
xmlfile = "Inventory.XML"
# Сохраняем XML дерево в память в объекте xtree
xtree = xmlInternalTreeParse(xmlfile)
# Переходим к корню дерева
root = xmlRoot(xtree)
# Обрабатываем вершины дерева функцией
visitNode(root)
```

В приведенном примере использован XML документ из файла Inventory.xml, его содержимое мы узнаем из следующего раздела. Для практического изучения работы приведенной выше рекурсивной функции, а также функций из пакета XML, скопируйте два предыдущих скрипта друг за другом в левую верхнюю панель RStudio. Не забудьте изменить имя файла в команде xmlfile = "Inventory.XML" на имя своего XML файла. Напомним, если файл располагается не в текущем рабочем директории (см. раздел Импортирование данных в R), то для него нужно указать полный путь.

Когда все скопировано и путь к анализируемому файлу указан можно изучать процесс интерпретации скрипта по-разному.

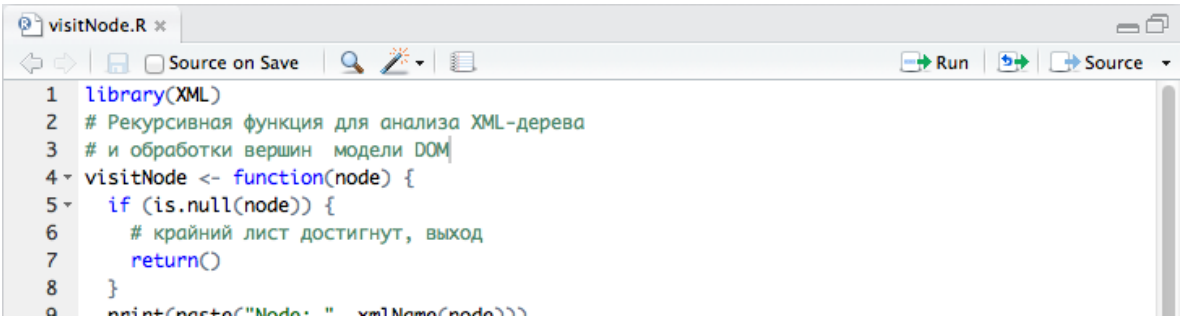


Рис. 6. Скрипт в панели Source

Для пошаговой (построчной) интерпретации кода нужно использовать иконку -> Run в правом верхнем углу панели Source. При каждом нажатии на нее в панели Source будет отмечаться интерпретируемая строка кода, а процесс интерпретации будет отображаться на нижней левой панели (Console) RStudio. Такой режим обработки удобен при отладке кода.

Можно выполнить весь код полностью использовав иконку -> Source также в правом верхнем углу.

2.3.2. Функция xmlSApply()

Когда стоит задача получить наглядное представление XML документа, полезно использовать функцию xmlSApply(). Ниже представлен скрипт, который обрабатывает некоторый XML файл Inventory.xml:

```
> url = "Inventory.xml"
> xmlfile = xmlTreeParse(url)
> class(xmlfile)
[1] "XMLDocument"         "XMLAbstractDocument"
> top = xmlRoot(xmlfile)
> print(top)

<INVENTORY>
 <BOOK>
  <TITLE>The Adventures of Huckleberry Finn</TITLE>
  <AUTHOR>Mark Twain</AUTHOR>
  <BINDING>mass market paperback</BINDING>
  <PAGES>298</PAGES>
  <PRICE>$5.49</PRICE>
 </BOOK>
 <BOOK>
  <TITLE>Leaves of Grass</TITLE>
  <AUTHOR>Walt Whitman</AUTHOR>
  <BINDING>hardcover</BINDING>
  <PAGES>462</PAGES>
  <PRICE>$7.75</PRICE>
 </BOOK>
 <BOOK>
  <TITLE>The Legend of Sleepy Hollow</TITLE>
  <AUTHOR>Washington Irving</AUTHOR>
  <BINDING>mass market paperback</BINDING>
  <PAGES>98</PAGES>
  <PRICE>$2.95</PRICE>
 </BOOK>
 <BOOK>
  <TITLE>The Marble Faun</TITLE>
  <AUTHOR>Nathaniel Hawthorne</AUTHOR>
  <BINDING>trade paperback</BINDING>
  <PAGES>473</PAGES>
  <PRICE>$10.95</PRICE>
 </BOOK>
</INVENTORY>
```

Предыдущие команды выполняют следующие действия: xmlfile = xmlTreeParse(url) - считывает файл Inventory.xml в рабочую область, class(xmlfile) - показывает что он является объектом класса "XMLDocument", xmlRoot(xmlfile) - переходит к корню документа и print(top) - распечатывает его.

```
> content = xmlSApply(top, function(x) xmlSApply(x, xmlValue))
> content_frame = data.frame(t(content),row.names=NULL)
> content_frame
```

	TITLE	AUTHOR	BINDING	PAGES	PRICE
1	The Adventures of Huckleberry Finn	Mark Twain	mass market paperback	298	$5.49
2	Leaves of Grass	Walt Whitman	hardcover	462	$7.75
3	The Legend of Sleepy Hollow	Washington Irving	mass market paperback	98	$2.95
4	The Marble Faun Nathaniel	Hawthorne	trade paperback	473	$10.95

2.4. Анализ HTML контента

2.4.1. Парсинг

Синтаксический анализ (*парсинг*) в информатике — процесс сопоставления линейной последовательности слов естественного или формального языка с его формальной грамматикой. Результатом обычно является дерево разбора или синтаксическое дерево (рис. 7).

Синтаксический анализатор (*парсер*) — это программа, выполняющая синтаксический анализ.

В ходе синтаксического анализа исходный текст преобразуется в структуру данных, обычно — в дерево, которое отражает синтаксическую структуру входной последовательности и хорошо подходит для дальнейшей обработки.

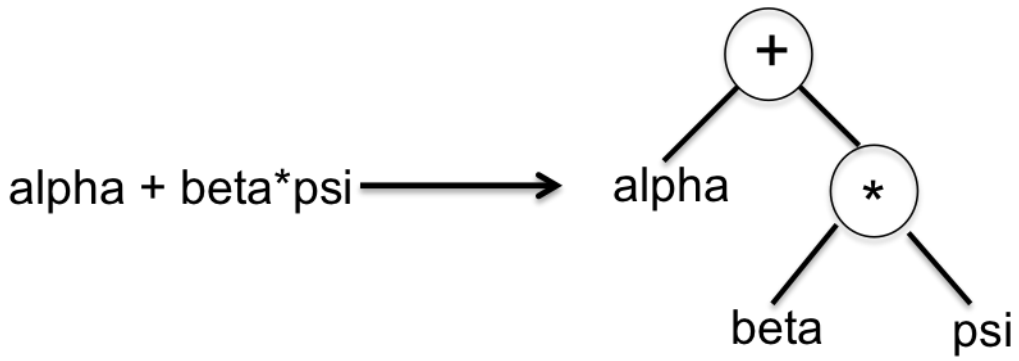


Рис. 7. Пример разбора выражения в дерево

Приведенные выше определения относятся к анализу обычного текста. Подобная технология парсинга используется для анализа отдельных частей контента, например содержимого тега <p>. При работе с контентом веба, в качестве исходного материала может выступать HTML представление.

Прежде всего для анализа контента его нужно загрузить его в рабочую область RStudio. Ниже представлен набор команд, который использует для этой цели две основные библиотеки RCurl и XML:

```
> require(RCurl)
> require(XML)
> url = "http://en.wikipedia.org/wiki/Euromaidan"
> SOURCE =  getURL(url,encoding="UTF-8")
```

С помощью функции substring() можно прочитать любой фрагмент загруженного сайта. В качестве параметров функции используется имя объекта, хранящего содержимое сайта, а также номера символа начала и конца фрагмента, при этом, результатом первичной загрузки ресурса будет просто строковая переменная:

```
> substring (SOURCE,1,200)
[1] "<!DOCTYPE html>\n<html lang=\"en\" dir=\"ltr\" class=\"client-nojs\">\n<head>\n<meta charset=\"UTF-8\" />\n<title>Euromaidan - Wikipedia, the free encyclopedia</title>\n<meta name=\"generator\" content=\"MediaWiki 1."
```

С помощью функции htmlParse() можно отформатировать эту переменную:

```
> parsed <- htmlParse(substring (SOURCE,1,200))
> parsed
<!DOCTYPE html>
<html lang="en" dir="ltr" class="client-nojs"><head>
<meta charset="UTF-8">
<title>Euromaidan - Wikipedia, the free encyclopedia</title>
<meta name="generator" content="MediaWiki 1.">
</head></html>
```

В данном случае, функция htmlParse() осуществляет структурирование исходной строки путем обработки и удаления вспомогательных символов.

Примеры анализа

Для выделения содержимого тэгов HTML можно использовать XPath выражения [15], например для тега заголовка h1 получим:

```
> xpathSApply(PARSED, "//h1")
[[1]]
<h1 id="firstHeading" class="firstHeading" lang="en">Euromaidan</h1>
```

Для выделения смыслового содержания контента тэгов HTML можно в команде xpathSApply() использовать атрибут xmlValue:

```
> xpathSApply(PARSED, "//h1",xmlValue)
[1] "Euromaidan"
```

Аналогично можно выделять содержимое любого тега HTML, в частности для заголовков h3 получаем:

```
> xpathSApply(PARSED, "//h3",xmlValue)
 [1] "Name history[edit]"
 [2] "Initial causes[edit]"
 [3] "Public opinion about Euromaidan[edit]"
  . . . . . .
```

2.4.2. Веб скрапинг

Веб скрапинг (от scraping - выскабливание) представляет собой технологию программного извлечения информации из сайтов [11]. Эта технология фокусируется на трансформации неструктурированных данных веба в структурированные данные, которые могут быть сохранены и проанализированы в базе данных, таблице, графике, диаграмме или изображении. Т.е. по сути, решается две задачи – извлечения нужной информации и ее визуализация. Таким образом, скрапинг решает более широкую задачу, нежели парсинг, который является составной частью технологии веб скрапинга.

В последнее время разработано много автоматизированных средств для реализации скрапинга. Например, фреймворк Scrapy [12] позволяет сконфигурировать «паука», выполняющего GET-запросы, искать нужные данные и экспортировать их в другой формат. Таким образом, с помощью Scrapy можно быстро извлечь нужную информацию с сайта и сформировать поток данных для обработки и использования в другом веб-приложении.

Наша задача состоит в изучении основных процедур, составляющих технологию скрапинга. К этим процедурам относится формирование URL ссылок на ресурсы, извлечение данных в доступном формате и их визуализация.

Формирование URL ссылок

Любая процедура скрапинга начинается с определения универсальных идентификаторов тех ресурсов веба, которые предстоит обработать. Подчеркнем, что здесь будет показано только то, как автоматизировать процесс формирования URL адресов, а не как находить эти ресурсы.

Пусть, например, стоит задача определения популярности какой-либо предметной области (скажем «web_scraping») у пользователей интернета, за какой-либо период времени (скажем, 2012-2014 год). Не секрет, что Wikipedia чаще других дает ответы на интересующие вопросы, поэтому анализ обращений к ней помог бы решить эту задачу.

Известно, что статистика трафика запросов по темам в Wikipedia формируется на ресурсе с URL http://stats.grok.se/. Анализ этого ресурса показывает, что доступ к нужной информации можно осуществить путем формирования URL следующего формата:

```
http://stats.grok.se/json/en/201201/web_scraping, где
- json……формат данных
- en.....язык
- 201201.....дата
- web_scraping.....тема
```

Таким образом, трафик запросов дается по месяцам и для решения поставленной задачи, прежде всего необходимо сформировать следующие URL:

```
http://stats.grok.se/json/en/201201/web_scraping
http://stats.grok.se/json/en/201202/web_scraping
http://stats.grok.se/json/en/201203/data_scraping
...
http://stats.grok.se/json/en/201412/web_scraping
```

это ни много, ни мало – 36 адресов - ссылок на веб ресурсы с необходимой информацией. Анализ этих ссылок показывает, что изменяющаяся часть адреса включает только четыре последних цифры даты от 20*1201* до 20*1412*. Введем переменную var = 201201 и сформируем первый URL, с помощью функции paste():

```
var=201201
```

```
paste("http://stats.grok.se/json/en/",var,"/web_scraping",sep="")

[1] "http://stats.grok.se/json/en/201201/web_scraping"
```

Теперь нам необходимо составить два цикла, один внутренний - для автоматического формирования месяцев с 01 по 12 и второй внешний - для формирования лет с 2012 года по 2014 год:

```
for (year in 2012:2014){
  for (month in 1:9){
print(paste("http://stats.grok.se/json/en/",year,0,month,"/web_scraping",sep
=""))
    }
   for (month in 10:12){
print(paste("http://stats.grok.se/json/en/",year,month,"/web_scraping",sep="
"))
   }
}
```

Результат выполнения скрипта:

```
[1] "http://stats.grok.se/json/en/201201/web_scraping"
[1] "http://stats.grok.se/json/en/201202/web_scraping"
[1] "http://stats.grok.se/json/en/201203/web_scraping"
…
[1] "http://stats.grok.se/json/en/201401/web_scraping"
[1] http://stats.grok.se/json/en/201412/web_scraping
```

Отметим, что внутренний цикл пришлось разбить на две части – одна для месяцев с 1 по 9 и другая для месяцев с 10 по 12 для того, чтобы правильно сформировать две последние цифры в датах.

Итак, мы автоматизировали процесс формирования URL адресов и перейдем к вопросу извлечения данных.

Извлечение данных

Прежде всего загрузим данные для одной даты, например январь 2014 года:

```
var=201401
url=paste("http://stats.grok.se/json/en/",var,"/web_scraping",sep="")
raw.data <- readLines(url, warn="F") # чтение данных
```

```
raw.data                                    # просмотр данных
[1] "{\"daily_views\": {\"2014-01-15\": 779, \"2014-01-14\": 806, \"2014-01-17\": 827,
\"2014-01-16\": 981, \"2014-01-11\": 489, \"2014-01-10\": 782, \"2014-01-13\": 756,
\"2014-01-12\": 476, \"2014-01-19\": 507, \"2014-01-18\": 473, \"2014-01-28\": 789,
\"2014-01-29\": 799, \"2014-01-20\": 816, \"2014-01-21\": 857, \"2014-01-22\": 899,
\"2014-01-23\": 792, \"2014-01-24\": 749, \"2014-01-25\": 508, \"2014-01-26\": 488,
\"2014-01-27\": 769, \"2014-01-06\": 0, \"2014-01-07\": 786, \"2014-01-04\": 456, \"2014-
01-05\": 77, \"2014-01-02\": 674, \"2014-01-03\": 586, \"2014-01-01\": 348, \"2014-01-
08\": 765, \"2014-01-09\": 787, \"2014-01-31\": 874, \"2014-01-30\": 1159}, \"project\":
\"en\", \"month\": \"201401\", \"rank\": -1, \"title\": \"web_scraping\"}"
```

Как и предполагалось, данные представлены в формате JSON и для выделения только нужной информации необходимо их предварительно обработать. Здесь в первую очередь, нас интересует только ключ daily_views и все пары ключ-значение внутри него, т.е. 2014-01-15: 779, 2014-01-14: 806, …, 2014-01-30: 1159. Для выделения этой информации используем следующий скрипт:

```
require(rjson)                  # загружаем необходимый пакет
rd  <- fromJSON(raw.data)       # разбиваем данные по ключам
rd.views <- rd$daily_views      # выбираем пары из daily_views
rd.unlst <- unlist(rd.views)    # преобразуем в вектор значений
df <- as.data.frame(rd.unlst)   # формируем фрейм df
df
            rd.unlst
2014-01-15     779
2014-01-14     806
.   .   .
2014-01-30    1159
```

Для лучшего понимания отдельных шагов преобразований посмотрите форматы представления каждой переменной rd, rd.views и rd.unlst.

Совместив загрузку и выделение нужных данных для произвольной даты можно использовать скрипт для извлечения данных во фрейм:

```
rd <- fromJSON(readLines(url, warn="F")) # чтение данных
rd.views <- rd$daily_views               # выбор нужных данных
df <- as.data.frame(unlist(rd.views))    # формирование фрейма
```

Визуализация данных

Исходные данные находятся во фрейме df. Для их визуализации загрузим необходимые библиотеки, сформируем столбцы со значениями и построим график (рис. 8):

```
require(ggplot2)                    # загружаем необходимые пакеты
require(lubridate)
df$date <-  as.Date(rownames(df))   # формируем столбец с датами
colnames(df) <- c(«views»,»date»)   # присваиваем имена столбцам
ggplot(df,aes(date,views))+         # строим график
  geom_line()+
  geom_smooth()+
  theme_bw(base_size=20)
```

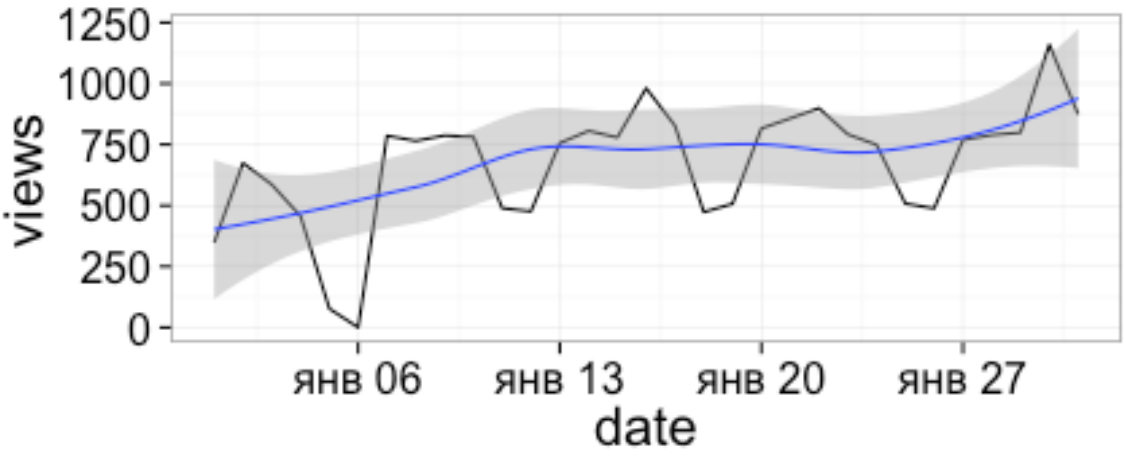


Рис. 8. Статистика трафика запросов по теме web_scraping за январь 2014 года

Для изучения особенностей использования функции ggplot() рекомендуем обратиться к встроенной системе справочной информации. Выберите закладку Help в правой нижней панели RStudio, введите запрос ggplot и нажмите Enter.

Рис. 9. Справка для функции ggplot() в закладке Help

Подобную практику изучения весьма полезно применять и в других случаях для уточнения влияния значений параметров функции на результат.

Построение функции

В заключение раздела рассмотрим решение задачи определения популярности какой-либо предметной области у пользователей интернета, за заданный интервал времени. Логично будет построить универсальную функцию getStat(theme,from,to), входными параметрами которой будут – интересующая тема (theme) и интервал времени (from,to), а возвращаемым параметром – статистика запросов по этой теме. Причем статистику запроса будем формировать в файле stat.txt в формате:

```
   year_month  how
1     2008_01 6282
2     2008_02 5638
3     2008_03 6219
```

где how – количество запросов по теме в данном месяце.

Ниже приведен полный код функции getStat(), со всеми необходимыми комментариями:

```
getStat <- function(theme,from,to){

        # функция чтения данных из URL
        getData <- function(url){
         rd <- fromJSON(readLines(url, warn=»F»)) # чтение данных
         rd.views <- rd$daily_views                # выбор числа запросов
         how_daily<-as.numeric(rd.views)
          how=0
          for(i in 1:length(how_daily)){           # суммирование
            how=how+how_daily[i]                   # запросов за месяц
          }
          return(how)
        }
# формирование статистики (year_month how)
 for (year in from:to){
  for (month in 1:9){       # формирование URL для месяцев 1 – 9
    url=paste(«http://stats.grok.se/json/en/»,year,0,month,»/»,theme,sep=»»)
    ddd=getData(url)        # получение числа запросов за месяц (how)
```

```
  month0=paste(0,month,sep=»»)
  var=c(year,month0)
  var=paste(var,collapse=»_») # формирование (year_month)
  p=c(c(var),c(ddd))              # формирование (year_month how)
  q=as.character(p)
  write(q,»stat.txt»,append=TRUE) # запись (year_month how) в stat.txt
 }
  for (month in 10:12){   # формирование URL для месяцев 10 – 12
  url=paste(«http://stats.grok.se/json/en/»,year,month,»/»,theme,sep=»»)
  ddd=getData(url)        # получение числа запросов за месяц (how)
  var=c(year,month)
  var=paste(var,collapse=»_») # формирование (year_month)
  p=c(c(var),c(ddd))              # формирование (year_month how)
  q=as.character(p)
  write(q,»stat.txt»,append=TRUE) # запись (year_month how) в stat.txt
 }}}
```

Скопируйте приведенный выше код в Rstudio, добавьте вызов функции getStat(«web scraping»,2014,2015) и посмотрите содержимое файла stat.txt. Отметим, что файл со статистикой сформируется в текущем рабочем директории.

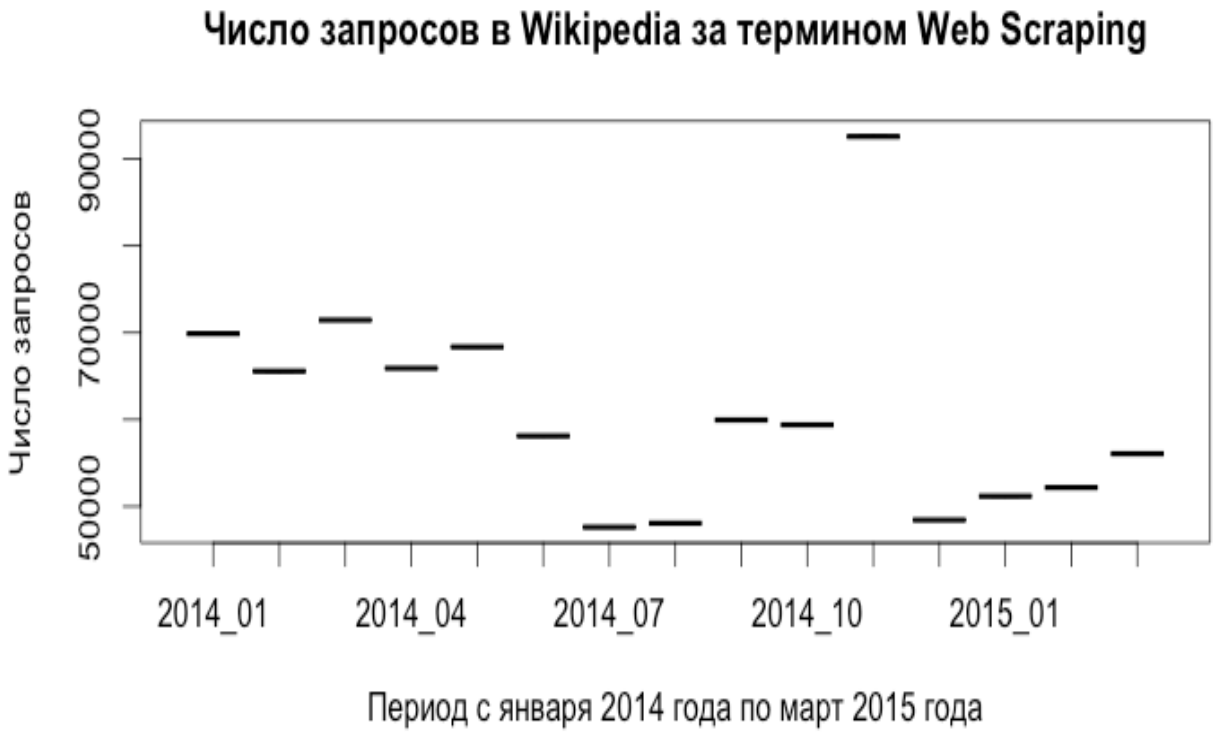


Рис. 10. Статистика трафика запросов по теме web_scraping
с января 2014 по март 2015 года

Для визуализации полученных данных прочитаем содержимое файла с помощью функции scan(), представим их в формате фрейма и построим график, используя функцию plot():

```
data = scan(«stat.txt», what=list («»,0))
fr=as.data.frame(data)
plot(fr,main=«Число запросов в Wikipedia за термином Web Scraping»,xlab=«Период с января 2014 года по март 2015 года»,ylab=«Число запросов»)
```

Из приведенного графика статистики запросов видно, что параметр main – определяет заголовок графика, xlab – надпись под осью x и ylab – надпись у оси y. Пик трафика запросов в указанный период приходится на ноябрь 2014 года.

2.5. Интерактивный JavaScript

Известно, что язык программирования JavaScript занимает прочное место среди ИТ-специалистов. Поэтому интересно будет узнать о возможностях использования этого языка в среде RStudio. Для поддержки интерактивной консоли JavaScript служит пакет V8 версии 0.3. Использование интерактивной консоли предоставляет удобное средство как для отладки скриптов, так и просто для изучения языка JavaScript.

2.5.1. Создание консоли

Рассмотрим особенности использования пакета V8 на конкретном примере из [13]:

```
library(V8)
data(diamonds, package="ggplot2")
# Создание JavaScript сессии
ct <- new_context()
ct$assign("diamonds", diamonds)
# Загрузка crossfilter JavaScript библиотеки
ct$source("http://cdnjs.cloudflare.com/ajax/libs/crossfilter/1.3.11/crossfilter.min.js")
```

Представленный выше код загружает необходимую библиотеку V8 и набор данных diamond из пакета ggplot2, а также связывает их с контекстом новой сессии JavaScript. Кроме того, осуществляется загрузка crossfilter JavaScript библиотеки. Теперь можно использовать консольный метод для интерактивного режима работы с JavaScript для этой сессии:

```
ct$console()
# This is V8 version 3.14.5.10. Press ESC or CTRL+C to exit.
# ~
```

Курсор ~ показывает, что нас обслуживает V8 и можно вводить JavaScript команды.

2.5.2. Пример использования

Решим задачу выбора 10 бриллиантов highest depth в ценовом диапазоне от 2000 до 3000:

```
//теперь мы в javasript :)
var cf = crossfilter(diamonds)
var price = cf.dimension(function(x){return x.price})
var depth = cf.dimension(function(x){return x.depth})
price.filter([2000, 3000])
output = depth.top(10)
```

Для вывода результата в R поступим следующим образом. Нажмем клавишу ESC и попадем обратно в R консоль с курсором >. Здесь считаем объект output с помощью функции ct$get() и выведем результат:

```
# Нажимаем ESC и уходим с консоли V8
result <- ct$get("output")
print(result)
```

Аналогичным образом, можно было загрузить весь этот скрипт в панель Source и, нажимая на иконку Run в правом верхнем углу панели, осуществить пошаговую отладку.

Отметим, что наилучший способ использования JavaScript скриптов в R, размещать их в отдельном файле (например, myscript.js) в рабочем директории и затем загружать его в R с помощью команды ct$source(“myscript.js”).

2.6. Анализ текстовых документов

В этом разделе представлены два основных пакета wordcloud и igraph, которые удобно использовать для анализа текстовых документов. Наряду с этим использованы программные пакеты XML, tm (от *text mining*) и RColorBrewer, играющие вспомогательную роль при решении задач анализа текстов.

2.6.1. Построение облака слов

Для первого знакомства с пакетом wordcloud в качестве исходного текста для анализа возьмем описание 6539 доступных пакетов с сайта http://cran.r-project.org/web/packages:

```
require(XML)
u = "http://cran.r-project.org/web/packages/available_packages_by_date.html"
t = readHTMLTable(u)[[1]]
```

Кроме того, нам нужен будет пакет tm (от text mining), имеющий широкие возможности для анализа текстовой информации. Обработка текста потребует нескольких функций из этого пакета. Первым шагом должно стать создание так называемого "корпуса", т.е. сведение всех имеющихся небольших текстов (описаний) в один объект:

```
require(tm)
corpus <- Corpus(DataframeSource(data.frame(as.character(t[,3]))))
```

С корпусом текстов можно выполнять разнообразные операции, что осуществляется через интерфейс функции tm_map() из пакета tm. Для начала сделаем так, чтобы все слова в анализируемых сообщениях были представлены прописными буквами:

```
corpus <- tm_map(corpus, tolower)
```

Следующий шаг - очень важный. Он состоит в том, чтобы удалить из сообщений как можно больше так называемых "стоп-слов", или "шумовых слов", т.е. слов, не несущих смысловой нагрузки. Для работы с англоязычными текстами, используется команда:

```
corpus <- tm_map(corpus, function(x) removeWords(x, stopwords("english"))).
```

Следует отметить, что в R включен также набор стоп-слов для работы с русскоязычными текстами stopwords("russian") состоящий из 159 слов.Теперь преобразуем "корпус" в матрицу индексируемых слов из сообщений. Подробнее о структуре и назначении этой матрицы можно узнать из [1]:

```
tdm <- TermDocumentMatrix(corpus,control = list(minWordLength = 1))
m <- as.matrix(tdm)
```

Подсчитываем частоту слов:

```
v <- sort(rowSums(m),decreasing=TRUE)
df <- data.frame(word = names(v),freq=v)
```

и в результате получаем фрейм df , в котором слова упорядочены по убыванию частоты повторяемости (табл. 3).

Таблица 3. Первые десять слов фрейма df

	word	freq
1	data	961
2	analysis	775
3	models	539
4	functions	392
5	regression	320
6	estimation	308
7	package	298
8	using	276
9	tools	270
10	model	265

Устанавливаем цветовую гамму и генерируем облако слов:

```
require(RColorBrewer)
require(wordcloud)
pal2 <- brewer.pal(8,"Dark2")
wordcloud(df$word,df$freq, scale=c(8,.2),min.freq=3,
      max.words=100, random.order=FALSE, rot.per=.15, colors=pal2)
```

Рис. 11. Облако слов из имен программных пакетов ресурса CRAN

Как видно из рисунка, одним из ключевых слов является data и далее лидируют слова занимающие первые места во фрейме df.

2.6.2. Графовый анализ текстов

Изучим пример графового анализа с использованием пакета igraph. В качестве данных для анализа возьмем данные из файла termDocMatrix.rdata, который может быть загружен с веб-страницы [14]. При составлении этого файла использовалось более 150 статей взятых из твиттера. Файл представляет из себя матрицу со строками соответствующими терминам использованным в статьях, и столбцами - номерам статей. Будем строить граф определяющий взаимосвязи терминов из этих статей на основе их совместной встречаемости.

Во-первых, загрузим данные для анализа в RStudio. После этого трансформируем их в матрицу смежности, на основе которой построим сеть в виде графа взаимосвязи терминов. Вершины графа будут соответствовать терминам из статей, а ребра определять взаимосвязи между терминами. Далее введем определенные коррективы в представление графа, чтобы сделать его более удобным для восприятия, установив цвета, размеры шрифта и прозрачности вершин и ребер.

Загрузка данных

```
> load("termDocMatrix.rdata")
# посмотрим на часть анализируемых данных
> termDocMatrix[1:5,1:20]
```

```
                Docs
Terms            1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
  analysis       0 0 0 0 0 0 0 0 0  0  0  0  0  0  0
  applications   0 0 0 0 0 0 0 0 0  0  0  0  0  0  0
  code           0 0 0 0 0 0 0 0 0  0  0  0  0  0  0
  computing      0 0 1 1 0 1 1 1 1  1  0  1  0  0  0
  data           1 1 0 0 1 0 0 0 0  0  1  1  1  1  1
```

Отметим, что представленная выше матрица termDocMatrix является стандартной матрицей, а не termDocument матрицей, используемой в рамках интеллектуального анализа текста с помощью пакета tm (от text mining).

Трансформация данных в матрицу смежности

Для того, чтобы привести матрицу termDocMatrix к виду, соответствующему построенной с помощью пакета tm необходимо выполнить следующую команду:

```
> termDocMatrix <- as.matrix(termDocMatrix)
```

Наконец, чтобы получить матрицу смежности необходимоо выполнить следующие преобразования:

```
# преобразование к булевой матрице
> termDocMatrix[termDocMatrix>=1] <- 1
# преобразование к матрице смежности
> termMatrix <- termDocMatrix %*% t(termDocMatrix)
# посмотрим на термины с 1 по 5
> termMatrix[1:5,1:5]
```

Terms	Terms				
	analysis	applications	code	computing	data
analysis	23	0	1	0	4
applications	0	9	0	0	7
code	1	0	9	0	1
computing	0	0	0	10	1
data	4	7	1	1	53

В построенной матрице смежности, строки и столбцы представляют термины, а элементы матрицы определяют число совместных вхождений двух терминов.

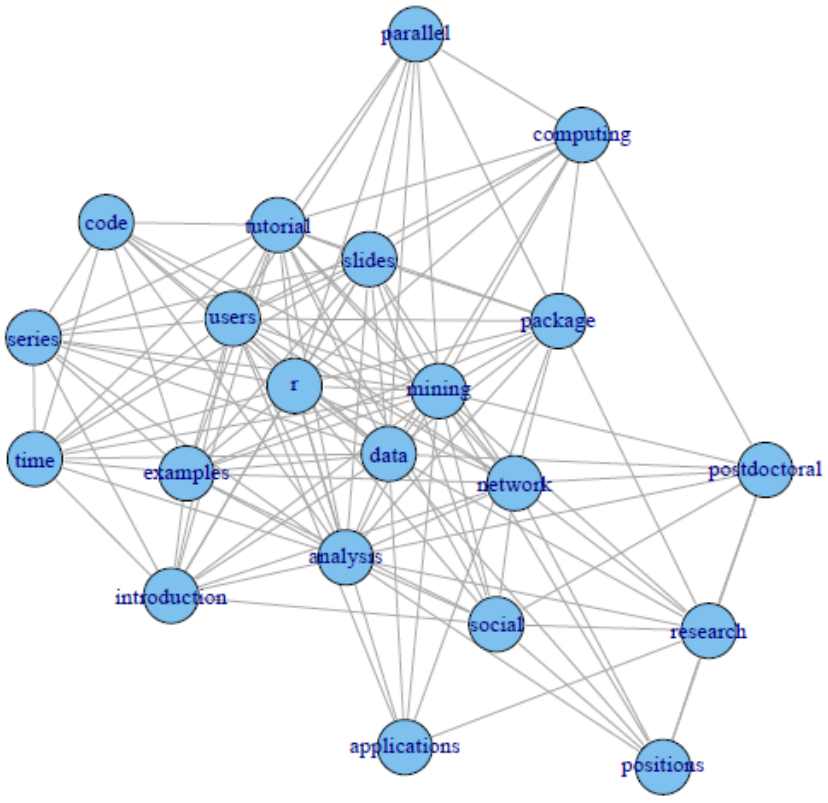


Рис. 12. Граф смежности

Построение графа смежности

Теперь можно построить граф с помощью функции graph.adjacency() из пакета igraph:

```
> library(igraph)
# строим граф из матрицы смежности
> g <- graph.adjacency(termMatrix, weighted=T, mode = "undirected")
```

```
# убираем петли
> g <- simplify(g)
# помечаем вершины терминами
> V(g)$label <- V(g)$name
# задаем веса вершин
> V(g)$degree <- degree(g)
> set.seed(3952)
> layout1 <- layout.fruchterman.reingold(g)
> plot(g, layout=layout1)
```

Другой вариант представления графа может быть получен с помощью команды:

```
> plot(g, layout=layout.kamada.kawai)
```

Следующая команда создает интерактивное приложение (рис. 13), которое позволит вручную изменять макет графа:

```
> tkplot(g, layout=layout.kamada.kawai)
```

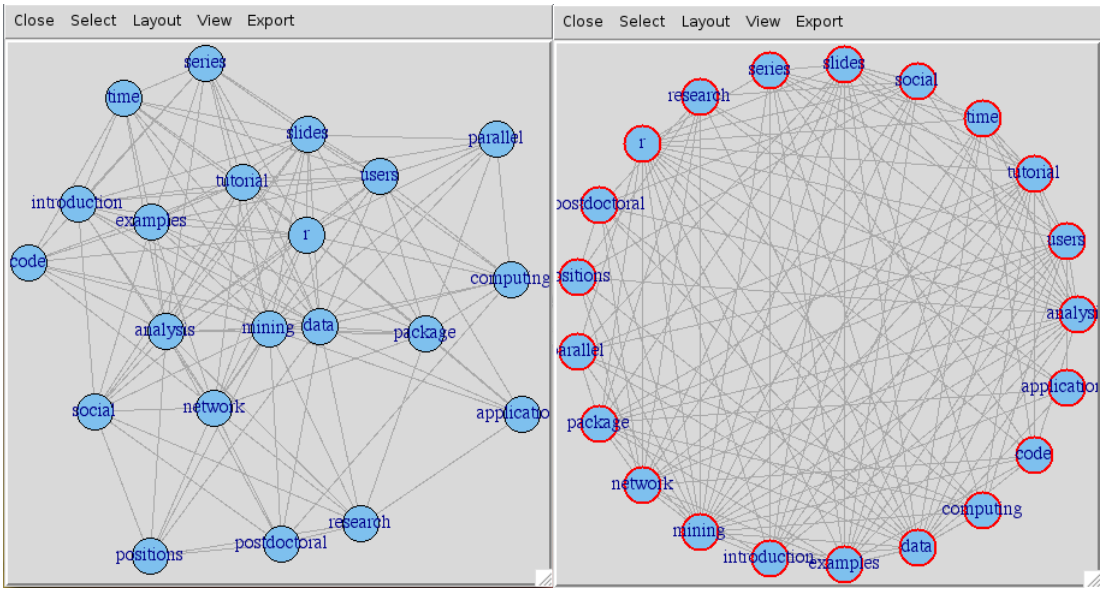


Рис. 13. Интерактивное приложение
для манипуляции с графом смежности

На рис. 13 слева изображен исходный граф, а справа – граф полученный путем выбора пункта Circle в меню Layout созданного приложения.

2.7. Кластерный анализ

Во многих исследованиях возникает задача выделения кластеров — групп похожих между собой результатов наблюдений. Одним из инструментов такого рода кластеризации служат функции kmeans() и hklust() из пакета stats. Функции обеспечивают кластерный анализ, т. е. позволяют количественно оценить меру принадлежности объекта к каждой из выделяемых групп. Мы рассмотрим примеры их использования, но сначала разберемся в различии подходов к решению задачи кластеризации.

Поясним подходы на простейшем примере. Положим у нас есть набор данных и у каждого элемента данных есть набор характеристик (например, если речь идет о людях: имя, пол, возраст, образование, место проживания и уровень дохода).

Таблица 4. Исходные данные для кластеризации

Имя	Пол	Возраст	Город	Доход
Михаил	м	25	Москва	50000
Андрей	м	34	Санкт-Петербург	120000
Елена	ж	27	Нижний Новгород	15000
Наталья	ж	36	Санкт-Петербург	35000
Ярослав	м	53	Москва	180000
Ольга	ж	24	Москва	40000

Каким образом можно разбить эти данные на кластеры? Можно по зарплате: Андрей и Ярослав в один кластер, Михаил, Наталья и Ольга во второй, а Елена в третий. Можно по возрасту: Михаил, Елена и Ольга в один, Андрей и Наталья во второй и Ярослав в третий. Проще всего разбить эти данные по месту жительства или полу. И всякий раз результат будет получаться разный.

Какой результат лучше? Ответить можно только зная задачу, для чего именно проводится кластеризация. Данные сами по себе никак не определяют возможные группировки. Чтобы получить однозначный результат необходимо ввести понятие расстояния. Причем, можно использовать сразу несколько характеристик объектов для определения расстояния между ними, например зарплату, возраст и пол.

Существуют различные меры расстояний, но для нас интуитивно понятным является классическое Евклидово расстояние и, в большинстве задач оно более чем достаточно. Важно лишь правильно нормализовать данные. Как в нашем примере: зарплаты измеряются в тысячах, а возраст в годах. Непосредст-

венно сравнивать эти две величины нельзя. А вот если предварительно привести их к диапазону от 0 до 1 то они станут вполне сравнимыми.

Таким образом, кластерный анализ, в первую очередь состоит из работы над данными. Требуется выбрать интересующие нас характеристики, нормализовать их и выбрать подходящую меру расстояний. И только после этого можно переходить к алгоритмам кластерного анализа.

Алгоритмы кластерного анализа

Известно много алгоритмов кластерного анализа. Далеко не полная и не единственная возможная классификация представлена на рис. 14.

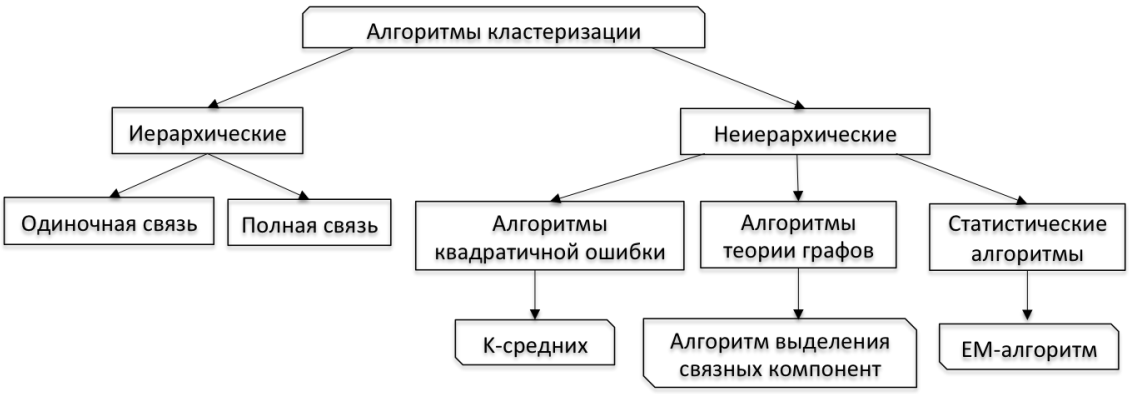


Рис. 14. Классификация алгоритмов кластерного анализа

Итак, все алгоритмы кластерного анализа делятся на две группы: иерархические и неиерархические. Первые строят не просто разбиение на классы, а иерархию разбиений. Результатом их работы, как правило, является дендрограмма, на основе которой, пользователь может сам выбрать желаемое разбиение.

Неиерархические алгоритмы, напротив, в результате работы выдают некоторое конкретное разбиение и имеют ряд параметров, позволяющих настраивать алгоритм для имеющихся данных. Естественно, вторые работают быстрее чем первые. Все методы кластеризации работают с данными в виде векторов в многомерном пространстве. Каждый вектор определяется значениями нескольких направлений, а направления и есть известные нам характеристики (пол, возраст, образование). Характеристики могут быть как количественные, так и качественные и искусство специалиста по data mining состоит в том, чтобы правильно отобрать и нормализовать эти характеристики, а потом выбрать подходящую меру расстояний. И только после этого в игру вступают алгоритмы кластеризации.

К числу наиболее популярных, неиерархических алгоритмов относится алгоритм k-средних. Он особенно популярен в силу простоты реализации и ско-

рости работы. Главным его недостатком является сходимость к локальному минимуму и зависимость результата от начального распределения. Кроме того, требуется заранее знать предполагаемое число кластеров k.

Алгоритм k-средних

Итак, сам алгоритм:

1) Выбрать k случайных центров в пространстве, содержащем исходные данные.

2) Приписать каждый объект из множества исходных данных кластеру исходя из того, какой центр к нему ближе.

3) Пересчитать центры кластеров используя полученное распределение объектов.

4) Если алгоритм не сошелся, то перейти к п. 2.

Типичные критерии схождения алгоритма это либо среднеквадратичная ошибка, либо отсутствие перемещений объектов из кластера в кластер. Существует множество вариаций этого алгоритма, некоторые из них пытаются выбирать наилучшее начальное распределение, другие позволяют кластерам разбиваться на части и объединяться.

Иерархическая кластеризация

По своему подходу, все алгоритмы иерархической кластеризации делятся на два типа: сверху-вниз и снизу-вверх. Первые начинают с одного большого кластера состоящего из всех элементов, а затем, шаг за шагом разбивают его на все более и более мелкие кластеры. Вторые, напротив, начинают с отдельных элементов постепенно объединяя их в более и более крупные кластеры. Для пользователя, принципиальной разницы между этими двумя подходами нет, куда более значимо то, каким способом интерпретируется и вычисляется расстояния между кластерами. По этому признаку иерархические алгоритмы делятся на алгоритмы с одиночной связью и с полной связью (существуют и другие подходы, но они менее популярны). В алгоритмах с одиночной связью расстоянием между двумя кластерами считается минимальное расстояние между всеми парами элементов из этих двух кластеров. В алгоритмах с полной связью расстоянием считается максимальное расстояние между всеми парами элементов из двух кластеров.

Таким образом, алгоритмы с полной связью склонны находить более компактные кластеры. Алгоритмы с одиночной связью, напротив, склонны выявлять сильно вытянутые и сложны формы. Но часто страдают из-за шумов. На рис. 15 приведен классический пример работы алгоритма одиночной и полной связи. Нам даны два кластера, соединенных "дорожкой шумов". Слева результат работы алгоритма с одиночной связью, а справа алгоритма с полной связью.

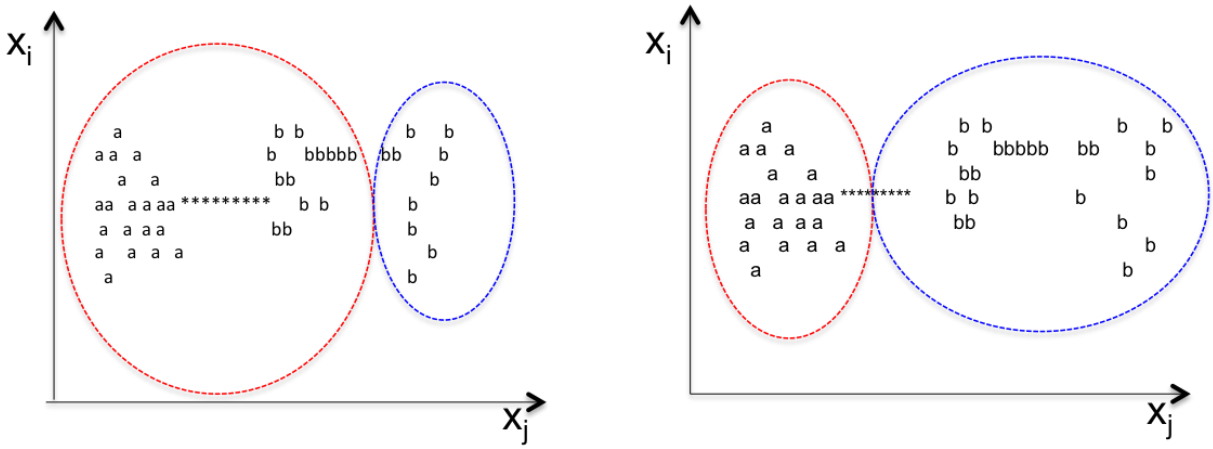


Рис. 15. Результаты работы алгоритмов кластеризации

Видно, что алгоритм с одиночной связью дал не самый адекватный результат. С другой стороны, алгоритмы с полной связью не способны выявить концентрические кластеры (рис. 16).

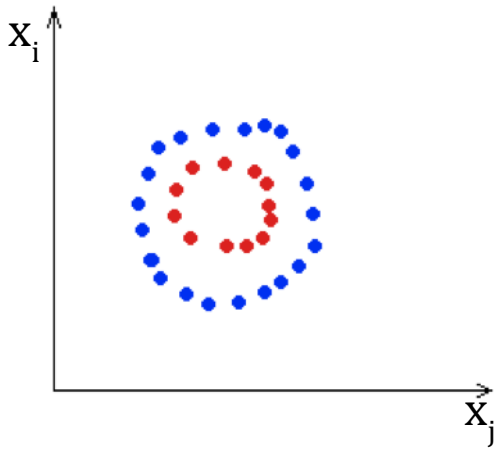


Рис. 16. Концентрическик кластеры

Таким образом, выбор алгоритма определяется конкретной задачей. Но на практике, кластеры сложной формы встречаются редко, а шумы часто и поэтому чаще применяются алгоритмы с полной связью.

Алгоритмы основанные на теории графов

На практике, в чистом виде, такие алгоритмы применяются редко, т.к. их вычислительная сложность не позволяет обрабатывать большие графы. Чаще используются в сочетании с другими алгоритмами, такими как k-средних. Клас-

сический пример, это алгоритм минимального покрывающего дерева. Идея состоит в том, чтобы представить весь набор данных в виде графа, вершины которого элементы данных, а вес каждого ребра равен расстоянию между соответствующими элементами. Здесь тоже, сначала придется определить понятие расстояния, тогда, мы можем построить минимальное покрывающее дерево на данном графе, а затем последовательно удалять ребра с наибольшим весом. Кластером считается множество элементов, соединенных "остатком" дерева. С каждым убранным ребром, количество кластеров увеличивается.

Статистические алгоритмы

Этот подход очень популярен в области распознавания изображений. Идея такова: предположим, что каждый кластер в наших данных, подчиняется некоторому распределению, как правило предполагается Гауссовское распределение.

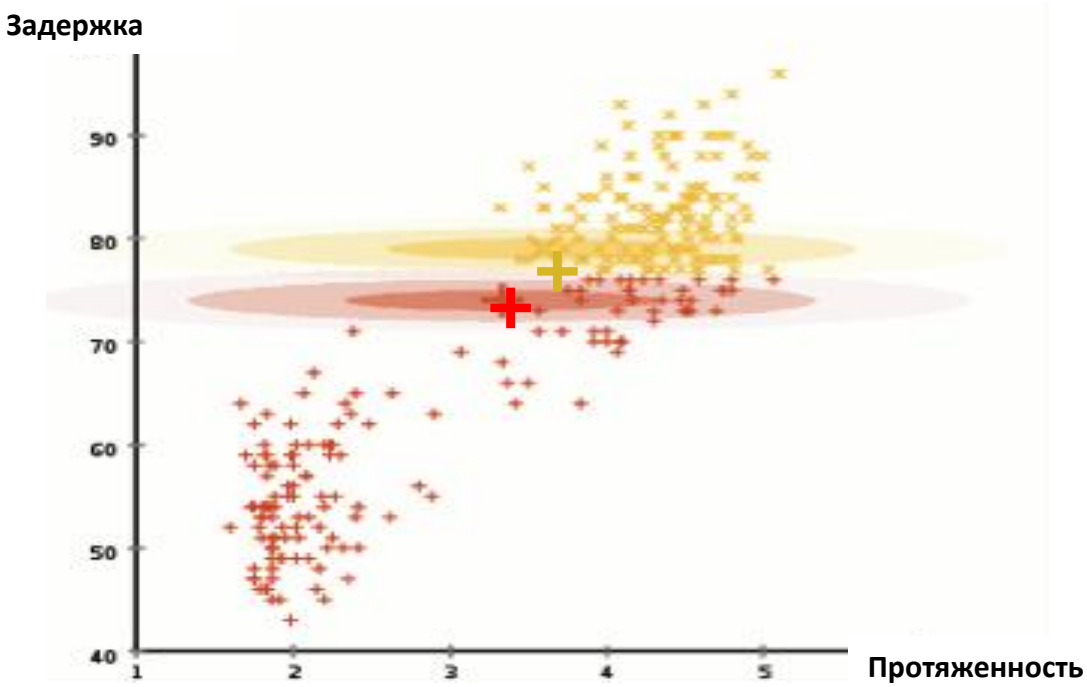


Рис. 17. Центры допустимых распределений

Далее, методами статистики, мы определяем параметры допустимых распределений и их центры. На этом основан, например, алгоритм EM-кластеризации.

2.7.1. Функция kmeans()

Для демонстрации использования неиерархического алгоритма k-средних приведем достаточно показательный пример из области финансовых рынков [15]. Допустим, у нас есть акции большого количества разных компаний. Одни из них растут, другие падают и мы накопили большую историю изменения их цен. Можем ли мы на основе этих данных объединить акции в группы? Логично было бы предположить, что акции из одного сектора рынка растут или падают вместе. И было бы логично если полученные группы это как-то отразили.

В качестве исходного набора данных загрузим ежедневные котировки акций примерно 20 российских компаний из разных областей, за два года. Для удобства, все они собраны в одном файле и доступны в [15]. Попробуем проанализировать их средствами кластерного анализа. Поскольку цены разных акций отличаются весьма сильно и абсолютная величина цены для нас не важна, а интересно только относительное изменение, приведем все цены к общему знаменателю: а именно, вместо самой цены возьмем логарифмическое изменение цены: $x_i = \log(c_i) - \log(c_{i-1})$, где c_i - цена закрытия в i-ый день, а c_{i-1} – в предыдущий:

```
# 1 - читаем файл с ценами
x = read.table(“stocks.csv”, sep = “,” , header = TRUE)
# 2 – логарифмируем цены
x = log(x)
# 3 – отбрасываем колонку с датой
x = x[-1]
# 4 – вычисляем разницу между соседними элементами
x = apply(x, 2, diff)
# 5 – транспонируем таблицу
x = t(x)
# 6 – разбиваем на 5 кластеров, максимум 1 миллион итераций
kmeans(x , 5, 1000000)
```

Получаем результат представлены в табл. 5. Сразу заметна тенденция к группировке в кластеры компаний из одного сектора: автопроизводители попали в пятый кластер, нефтедобывающие компании и крупные банки в третий, все генерирующие компании в четвертый. Если запустить этот алгоритм снова, то результат может получится другой, но тенденция, тем не менее, сохранится. Почему было решено разбивать именно на 5 кластеров, а не 6 или 4? К сожалению, решение вопроса о количестве кластеров не алгоритмизируется, в конечном счете это всегда должен решать пользователь.

Таблица 5. Результаты кластеризации

1	2	3	4	5
RASP	AFLT, PMTL, PLZL, MMBM, MTSI, VZRZ	GAZP, LKOH, VTBR, URKA, TRNFP, SIBN, SBER, ROSN, GMKN, SNGS	OGK1, OGK2, OGK5	KMAZ, AVAZ

Предлагаем обязательно проанализировать работу представленного скрипта, скопировав его в панель Source и используя пошаговый режим. Попробуйте изменить параметры функции kmeans() и посмотрите к каким изменениям результата это приведет.

2.7.2. Функция hclust()

Использование иерархической кластеризации позволяет отложить на потом решение вопроса о количестве кластеров, поскольку предоставляет выбор из многих уровней. Если Вы уже проанализировали работу предыдущего скрипта, то легко проанализируете следующий алгоритм. Для этого в предыдущем скрипте нужно оставить неизменными все строки до строки с комментарием #6 а вместо него вставить:

```
# 1 - читаем файл с ценами
x = read.table("stocks.csv", sep = "," , header = TRUE)
# 2 – логарифмируем цены
x = log(x)
# 3 – отбрасываем колонку с датой
x = x[-1]
# 4 – вычисляем разницу между соседними элементами
x = apply(x, 2, diff)
# 5 – транспонируем таблицу
x = t(x)
# 6 – используем функцию hclust() и строим график
hc = hclust(dist(x))
plot(hc)
```

Нам понадобилось заменить всего одну строчку в прежнем примере, чтобы получит вместо таблицы график (рис. 18).

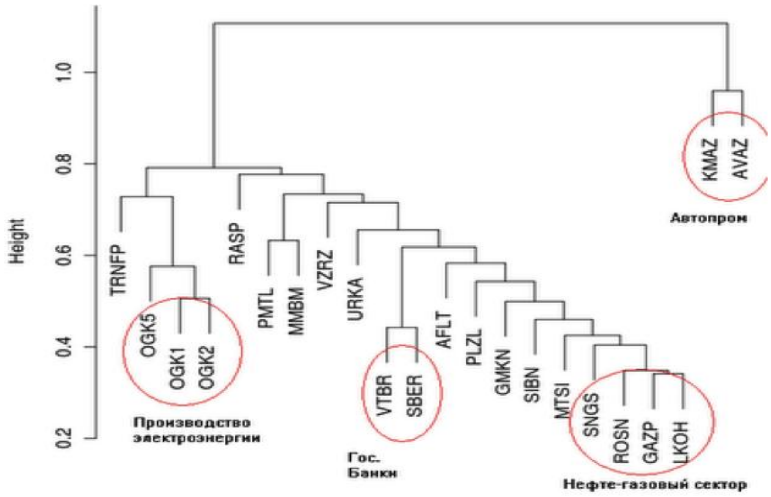


Рис. 18. Кластеризация предприятий

На этом рис. 18 уже гораздо лучше видны традиционные секторы нашей экономики. Видны и «выколотые точки», которые могут стать предметом особого исследования, ведь если компания ведет себя не как все в её отрасли, значит она что-то делает по-другому. Знание подобной информации может дать дополнительное преимущество.

2.8. Отдыхай с функцией getXKSD()

Мировое сообщество пользователей и почитателей платформы языка R весьма велико. Среди них найдется немало любителей хорошей шутки, анекдота, комикса, особенно в своей предметной области - технологии обработки информации. Примером тому может послужить специально созданный для этих целей веб-комикс сайт: http://xkcd.com/.

Хорошо поработав, можно приятно отдохнуть, вчитываясь и всматриваясь в материалы этого сайта. Для доступа к нему служит функция getXKSD() из пакета пакет RXKSD. Ниже приведен скрипт, который загрузит в панель Plot комикс под номером 1314:

```
> install.packages("RXKCD")
> library("RXKCD")
> q=getXKCD(1314, display=TRUE)
```

Рис. 19. Комикс 1314 из пакета RXKSD

- Программа, написанная на языке Haskell, с гарантией не имеет никаких побочных эффектов.

- … это потому, что никто не сможет даже запустить ее?

Попробуйте запустить другие комиксы, их там немало.

Делу время – потехе час! Пора приниматься за контрольные задания.

КОНТРОЛЬНЫЕ ЗАДАНИЯ

Задание 1

Исходные данные

American Community Survey предоставляет для скачивания данные, взятые из материалов различных обследований в Соединенных Штатах. Скачайте данные из опроса о жилье в штате Айдахо в 2006 г. с сайта:

https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2Fss06hid.csv

Загрузите эти данные в RStudio с помощью команды read.csv(). Книга кодирования, описывающая термины используемых в таблице переменных находится на:

https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2FPUMSDataDict06.pdf

Задача

Используя исходные данные определите общее количество предложений недвижимости с кодом расположения PUMA = 800 (Public Use Microdata Area code).

Ответ

Укажите правильный ответ: (231 1077 851 104)

Задание 2

Исходные данные

Скачать Excel таблицу из данных Natural Gas Aquisition Program по адресу: http://www.gsa.gov/dg/pbs/DATA.gov_NGAP.xlsx (оригинальный источник данных: http://catalog.data.gov/dataset/natural-gas-acquisition-program)

Задача

Используя исходные данные запишите строки 18-22 и столбцы 7-12 в переменную **dat.** Используя оператор for(), найдите максимальное значение в столбце NA..5 фрейма **dat.**

Задание 3

Исходные данные

Загрузите XML данные о ресторанах Балтимора в переменную **res** с сайта: https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2Frestaurants.xml

Задача

Сколько ресторанов описано в загруженных данных? В переменную **zip** прочитайте все zipcode. Посчитайте число ресторанов с zipcode = 21218?

Ответ

Укажите правильный ответ: (96 83 69 122)

Задание 4

Исходные данные

Известно, что статистика трафика запросов по темам в Wikipedia формируется на ресурсе с URL http://stats.grok.se/. Доступ к нужной информации можно осуществить путем формирования URL следующего формата:

http://stats.grok.se/json/en/201201/xxxxxxx, где
- json……формат данных
- en.....язык
- 201201.....дата
- xxxxxxx.....тема

Задача

Визуализируйте данные по обращению к страницам Wikipedia в феврале 2014 по теме "Sochi games" .

Сравните число этих обращений к русскоязычным страницам.

Задание 5

Показать статистику трафика запросов к Wikipedia (см. исходные данные к предыдущему заданию) по темам "web scaraping" и "semantic web" за период 2012-2014 год на одном графике. Какая из тем превалирует и почему?

Задание 6

Создайте консоль для работы с JavaScript, как показано в разделе 2.5.1. Разместите код на JavaScript в отдельном файле (например, myscript.js) в рабочем директории и затем загрузите его в R с помощью команды ct$source(“myscript.js”).

Продемонстрируйте работу кода в пошаговом режиме. Содержимое кода может быть произвольным.

Задание 7

Скачайте тексты трех русских народных сказок. Сформируйте облако слов, используя stopwords("russian"). Какие слова следует исключить? Добавьте эти слова в вектор исключаемых слов и сформируйте новое облако.

Продемоннстрируйте приемы формирования различной цветовой гаммы.

Задание 8

Используя термины из трех русских сказок задания 7, постройте граф смежности с помощью функции graph.adjacency() из пакета igraph. Создайте интерактивное приложение с помощью функции tkplot() и продемонстрируйте манипуляции по изменению макета графа.

Задание 9

Средствами кластерного анализа осуществитете разбиение марок автомобилей и их владельцев на классы, каждый из которых соответствует определенной рисковой группе. Наблюдения, попавшие в одну группу, характеризуются одинаковой вероятностью наступления страхового случая, которая впоследствии оценивается страховщиком.

Исходные данные взять с из http://www.statsoft.ru/solutions/ExamplesBase/.

Задание 10

Используя функцию fanny() из пакета cluster постройте два кластера для набора данных cars. Изобразите эти кластеры графически с помощью функции clusplot(). Объясните полученный результат.

Задание 11

Дана матрица случайных чисел вида X = [rand(10,3); rand(10,3)+1.2; rand(10,3)+2.5]. Построить кластеры по трем алгоритмам кластеризации - иерархический, k-среднее(k-means) и FCM(c-means), вывести графики и текстовый результат анализа.

Задание 12

Используя информацию из http://profitraders.com/Rlang/Rintro.html с помощью пакета quantmod построить график котировок акций фирмы Майкрософт (MFST) за три последних месяца с помощью функции chartSeries().

Построить аналогичный график на белом фоне в виде «баров», с помощью функции barChart().

Задание 13

На последний график задания 13 добавьте два индикатора с помощью функций addMACD() и addBBands() из пакета TTR. Объясните назначение индикаторов MACD и полос Боллинджера.

Задание 14

Используя информацию из http://profitraders.com/Rlang/Rintro.html выведите таблицу котировок акций Microsoft за все дни января 2015-го года. Постройте график изменения значений MSFT.Ajusted.

Задание 15

Используя информацию из http://profitraders.com/Rlang/Rintro.html выведите таблицу котировок акций Microsoft за последние шесть месяцев. Сравните эти данные с данными за аналогичные шесть месяцев прошлого года.

Приведите результаты сравнения данных.

3. Разработка веб приложений

Для визуализации информации удобно использовать веб приложения по ряду причин.

- не требуется специальных программных средств, кроме браузера;
- такое приложение может быть просто опубликовано в вебе;
- веб приложения допускают

3.1. Разработка реактивных приложений

Пакет Shiny позволяет легко создавать интерактивные веб-приложения прямо из RStudio. Пакет включает одиннадцать встроенных примеров, каждый из которых является автономным приложением демонстрирующим, как работает Shiny и служит хорошим пособием для изучения Shiny.

Пример приложения 01_hello (рис. 20) показывает построение гистограммы для представления табличных данных с заданным количеством прямоугольников.

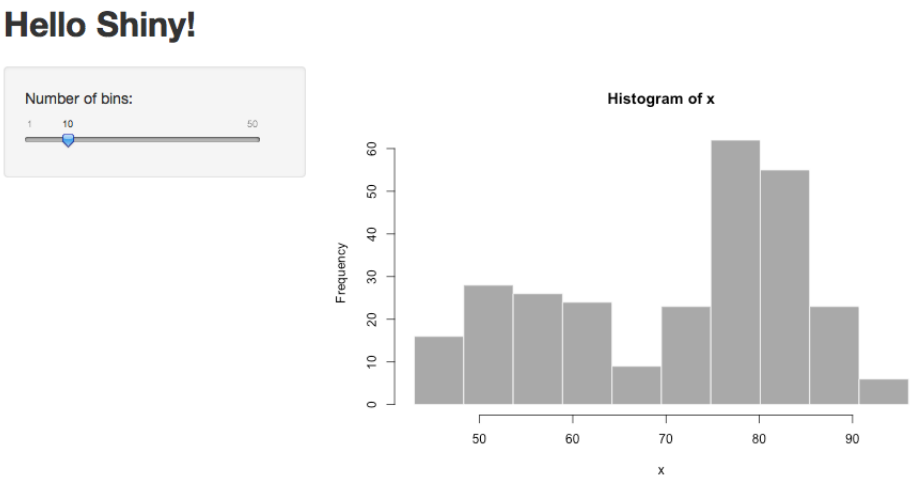


Рис. 20. Интерфейс приложения 01_hello

Чтобы запустить приложение 01_hello , необходимо ввести две команды:

```
> library(shiny)
> runExample("01_hello")
```

Первой командой загружается библиотека, обслуживающая пакет Shiny, а второй запускается приложение.

Пользователи могут изменять количество прямоугольников с помощью ползунка, и приложение будет немедленно реагировать на это изменение. Будем использовать этот пример для изучения структуры приложений пакета Shiny и создания своих приложений.

3.1.1. Структура Shiny приложения

Shiny приложение имеет две составляющие: *скрипт пользовательского интерфейса* и *серверный скрипт*.

Скрипт пользовательского интерфейса управляет форматом и внешним видом приложения. Он определяется в файле с именем ui.R. Вот сценарий ui.R приложения 01_hello:

ui.R

```
# Определение пользовательского интерфейса для приложения
shinyUI(fluidPage(
   # Заголовок приложения
  titlePanel("Hello Shiny!"),
   # Определение ползунка для установки числа прямоугольников
sidebarLayout(
    sidebarPanel(
      sliderInput("bins",
                  "Number of bins:",
                  min = 1,
                  max = 50,
                  value = 30)
    ),
     # Демонстрация сгенерированной гистограммы
    mainPanel(
      plotOutput("distPlot")
    ))  ))
```

Серверный скрипт server.R содержит команды необходимые компьютеру для построения приложения. Ниже приведен текст этого скрипта:

server.R

```
# Определение логического сервера для вывода гистограммы
shinyServer(function(input, output) {
# Выражение для формирования гистограммы определяет, что:
```

```
  #  1) надо динамично пересчитывать число прямоугольников
  #  2) выводить гистограмму в виде графически
   output$distPlot <- renderPlot({
    x    <- faithful[, 2]
    bins <- seq(min(x), max(x), length.out = input$bins + 1)
     # Вывод гистограммы с заданным числом прямоугольников
    hist(x, breaks = bins, col = 'darkgray', border = 'white')
  })
 })
```

Фактически, сценарий server.R приложения 01_hello достаточно прост: выполняются некоторые вычисления, а затем строится гистограмма с необходимым числом прямоугольников. Тем не менее, необходимо правильно задать параметры функции renderPlot().

Создание пользовательского интерфейса

Рассмотрим как построить пользовательский интерфейс для некоторого приложения. Любое приложение строится на основе как минимум двух файлов с фиксированными именами ui.R и server.R. Таким образом, целесообразно для каждого приложения создавать свою папку и в них размещать эти файлы. Будем использовать шаблоны готовых скриптов server.R и ui.R следующего содержания:

ui.R

```
shinyUI(fluidPage(
))
```

server.R

```
shinyServer(function(input, output) {
})
```

Этот код - минимум, который необходим для создания Shiny приложения. Результатом является пустое приложение с пустым пользовательским интерфейсом, которое будет нашей отправной точкой.

Использование в скрипте ui.R функции fluidPage(), позволит создать окно приложения, которое автоматически приспосабливается к размерам окна браузера пользователя. Например, следующий сценарий:

```
# ui.R
shinyUI(fluidPage(
  titlePanel("Заголовок"),
  sidebarLayout(
```

```
    sidebarPanel( "Боковая панель"),
    mainPanel("Основная панель")
  )
))
```

создаст пользовательский интерфейс (рис. 21), который имеет панель заголовка titlePanel(), и макет боковой панели sidebarLayout(), включающей в себя боковую sidebarPanel() и основную панели sidebarPanel(). Следует отметить, что эти элементы расположены внутри функции fluidPage().

Функции titlePanel() и sidebarLayout() являются двумя наиболее популярными элементами в fluidPage(). Они создают основу Shiny приложения с боковой панелью. Функция sidebarLayout() всегда имеет два аргумента: функцию выхода sidebarPanel() и mainPanel(). Эти две функции позволяют разместить контент в любой из панелей. Боковая панель появится на левой стороне вашего приложения по умолчанию. Можно переместить ее в правую сторону, дополнив sidebarLayout() дополнительным параметром position = "right".

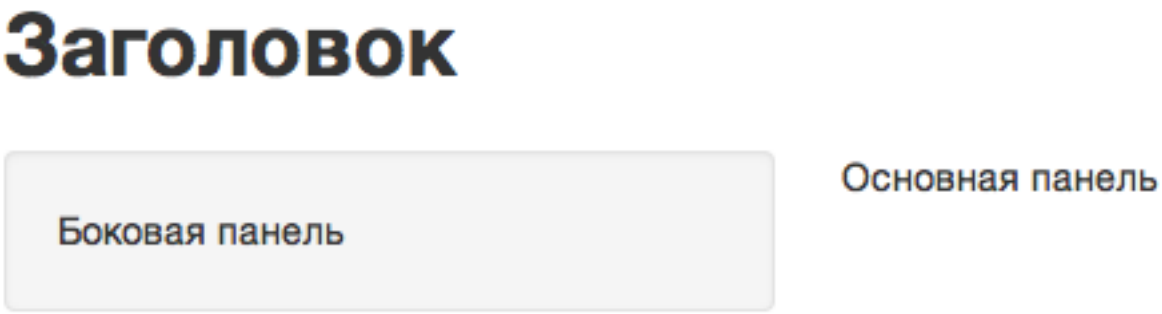


Рис. 21. Простейший пользовательский интерфейс

Используя другие функции можно создавать более продвинутые макеты приложения. Например, функция navbarPage() позволяет создавать многостраничные приложения, содержащие панель навигации. Кроме того, доступны к применению функции – налоги HTML тегов, используемых в веб дизайне. Они также, могут существенно изменить внешний вид пользовательского интерфейса там, где это необходимо.

Использование тегов

Чтобы сформировать продвинутый контент в любой панели, можно использовать функции - аналоги HTML тегов.

Таблица 6. Функции – аналоги HTML тегов

Функция Shiny	Тег HTML	Формирует
p	<p>	Абзац текста
h1 – h6	<h1> - <h6>	Заголовки от 1 до 6 уровня
a	<a>	Гиперссылка
br	 	Перенос строки
div		Абзац текста с уникальным стилем
span		Часть текста в строке с уникальным стилем
pre	<pre>	Текст с фиксированной шириной символов
code	<code>	Форматированный блок кода
img	<img>	Изображение
strong		Жирный текст
em		Наклонный текст
HTML		Обрабатывает строку символов как HTML код

Вот пример использования некоторых из них:

```
# ui.R
shinyUI(fluidPage(
 titlePanel("Новое приложение"),
 sidebarLayout(
  sidebarPanel(),
  mainPanel(
   p("Функция p() создает абзац текста"),
   p("Новая функция p() образует новый абзац. Добавочный атрибут стиля меняет шрифт всего абзаца.",style = "font-family: 'times'; font-si30pt"),
   strong("Функция strong() делает текст «жирным»."),
   br(),
   em(" Функция em() делает текст наклонным."),
   br(),
   code("Функция code() отображает текст похожим на компьютерный код"),
   div("Функция div() создает сегменты текста идентичного стиля. Этот сегмент текста голубого цвета, потому что добавлен аргумент style к этому div", style = "color:blue"),
   br(),
   p("Функция span() похожа на функцию div(), но она работает",
    span("с группой слов ", style = "color:blue"),
    "которые содержатся внутри абзаца.")
```

```
  )
 )))
```

Результат работы этого скрипта приведен на рис. 22.

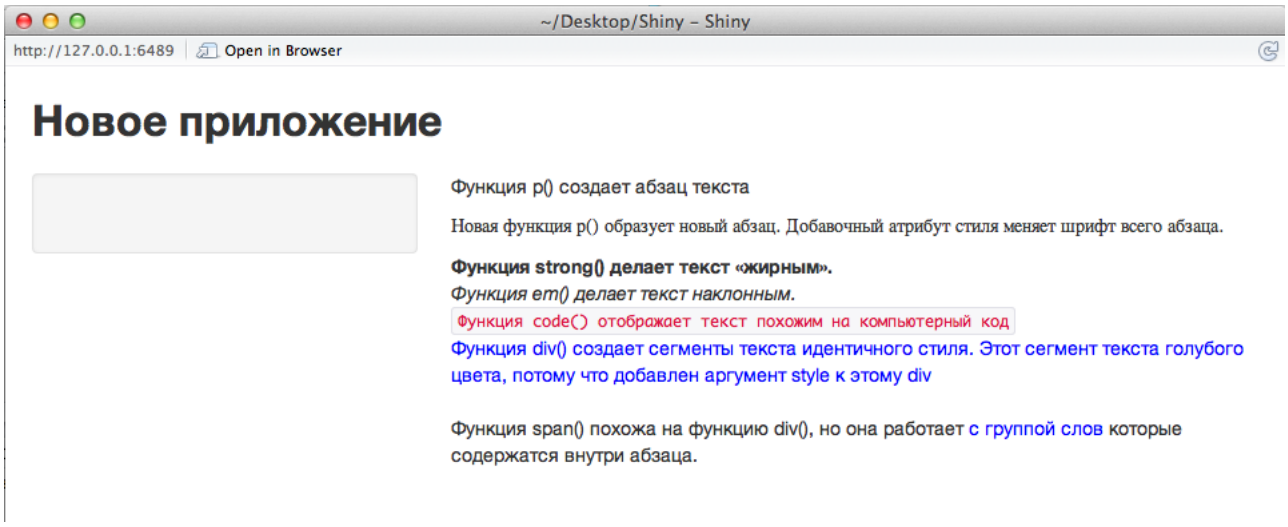


Рис. 22. Результат работы скрипта «Новое приложение»

Здесь рассмотрена лишь небольшая часть Shiny функций – аналогов HTML тегов. Общее количество этих функций равно 110 и об особенностях использования любой из них можно узнать на http://shiny.rstudio.com/articles/tag-glossary.html.

Вставка изображений

Изображения могут улучшить внешний вид приложения и помогут пользователям лучше понять его содержание. Для вставки изображения используется функция IMG и имя файла в качестве аргумента SRC (например, IMG (SRC = "my_image.png")). Можно также включать другие HTML параметры, такие как высота и ширина img(src = "my_image.png", height = 72, width = 72). Пример скрипта с картинкой:

```
shinyUI(fluidPage(
 titlePanel("Shiny с картинкой"),
 sidebarLayout(
  sidebarPanel(),
  mainPanel(
   img(src="Hall.gif", height = 200, width = 200)
  )
```

```
  )
))
```

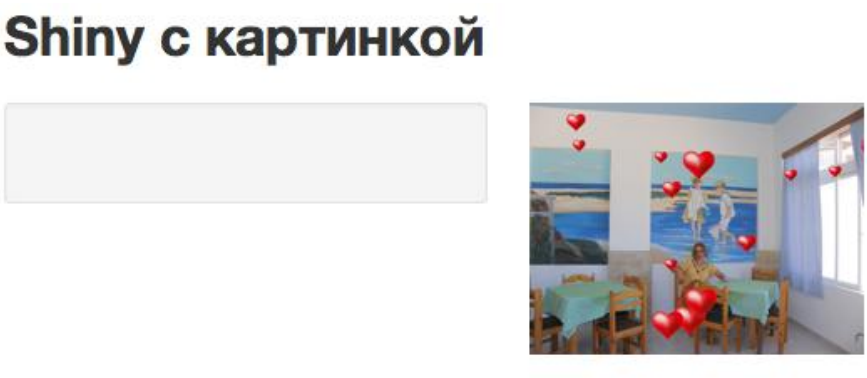


Рис. 23. Результат работы скрипта «Shiny с картинкой»

Обратите внимание, что высота и ширина задается в пикселах. Функция IMG всегда ищет файл изображения в папке под названием WWW, которая должна быть расположена в том же каталоге, что и сценарий ui.R.

Приведем еще один пример приложения **Shiny_App**, сочетающий описанные выше возможности (рис.24):

```
# ui.R
shinyUI(fluidPage(
 titlePanel("Shiny_App"),
 sidebarLayout(
  sidebarPanel(
   h3("Инсталляция"),
   p("Пакет Shiny размещен на CRAN, поэтому его можно установить в
RStudio обычным образом:"),
   code('install.packages("shiny")'),
   br(),
   br(),
   br(),
   br(),
   img(src = "rs3.jpeg", height = 72, width = 72),
   "  Shiny - продукт  ",
    span("RStudio", style = "color:blue")
  ),
  mainPanel(
   h1("ЧТО ТАКОЕ Shiny?"),
```

```
            p("Shiny новый программный пакет от RStudio, который позволяет ",
              em("чрезвычайно просто"),
              " создавать интерактивные динамичные веб приложения на языке
R."),
            br(),
            p("Для знакомства с пакетом и живыми примерами посетите ",
              a("Домашние страницы Shiny.",
                href = "http://www.rstudio.com/shiny")),
            br(),
            h2("ОТЛИЧИТЕЛЬНЫЕ ОСОБЕННОСТИ"),
            p("* Позволяет создавать полезные веб приложения с помощью не-
скольких строчек кода без использования JavaScript."),
            p("* Приложение Shiny автоматически “живет”, также как ",
              strong("представления"),
              " в СУБД. Выходные значения изменяются тотчас, как только пользо-
ватель изменяет входные, без необходимости перезапуска браузера.")
          ))))
        # server.R
        shinyServer(function(input, output) {
        })
```

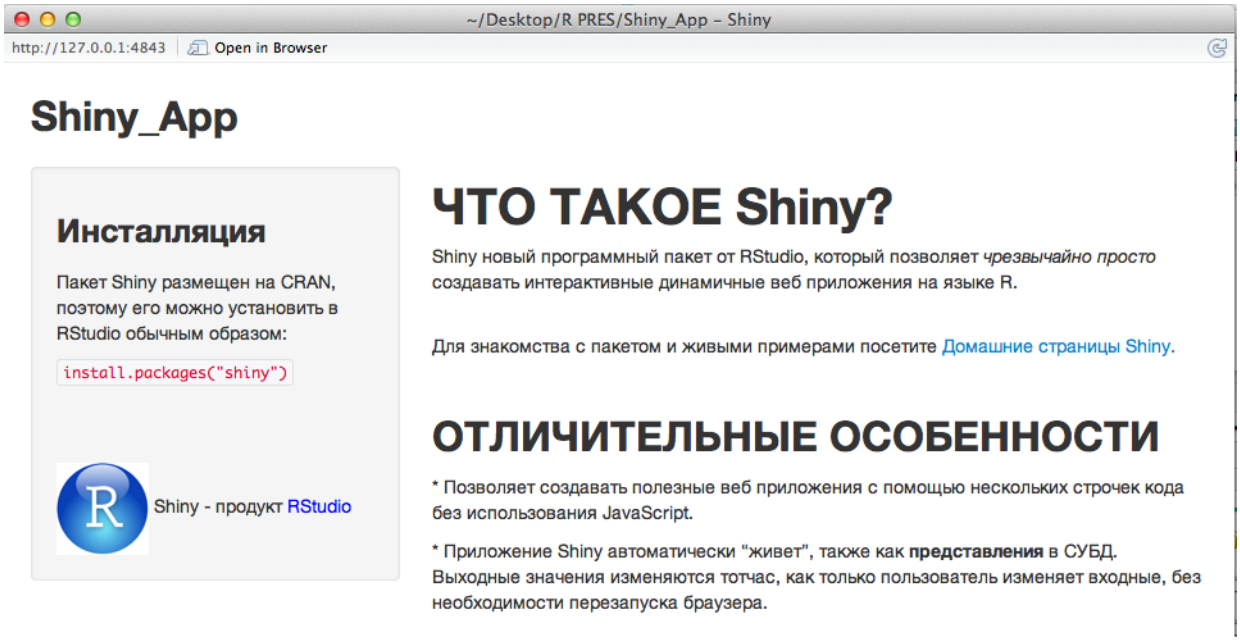


Рис. 24. Пример Shiny приложения

Создайте свое приложение. Скопируйте два эти скрипта в отдельную папку, не забудьте также создать в ней папку с именем WWW и поместить туда какое-либо изображение. Запустив это приложение, Вы получите нечто похожее на рис. 24.

Возможно у Вас не получилось изображение – это потому, что надо поменять имя изображения src = "**rs3.jpeg**" на выбранное Вами имя.

Виджеты управления

Рассмотрим возможности Shiny по обеспечению интерактивного взаимодействия пользователя с приложением. На рис. 5 представлены основные виджеты управления.

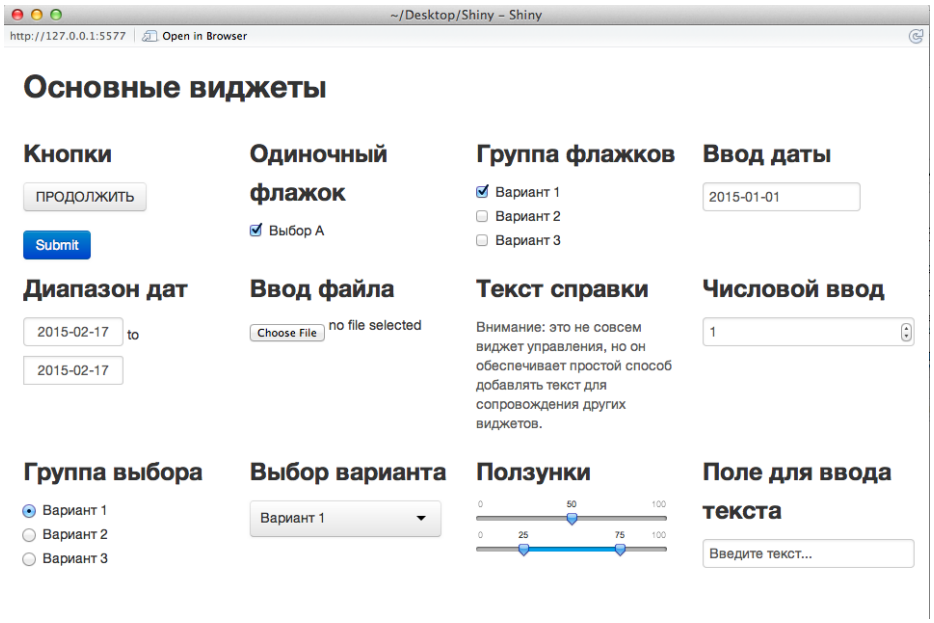


Рис. 25. Результат работы скрипта «Основные виджеты»

В пакет Shiny включен большой набор готовых виджетов, каждому из которых соответствует функция R с «понятным» названием. Например, функция с именем actionButton(), создает кнопку действия, а dateInput() – окно ввода даты. В табл. 7 перечислены функции создания стандартных виджетов управления.

Таблица 7. Функции создания виджетов

Функция	Виджет
actionButton	Кнопка действия
checkboxGroupInput	Группа флажков
checkboxInput	Одиночный флажок
dateInput	Ввод даты

Окончание табл.7

Функция	Виджет
dateRangeInput	Ввод диапазона дат
fileInput	Ввод файла
helpText	Текст справки
numericInput	Числовой ввод
radioButtons	Группа выбора
selectInput	Выбор варианта
sliderInput	Ползунок
submitButton	Кнопка подтверждения
textInput	Поле для ввода текста

Добавить виджеты на веб-страницу можно таким же образом, как добавлялись другие виды контента HTML. Чтобы добавить виджет в приложение достаточно поместить соответствующую функцию виджета в sidebarPanel() или mainPanel() в ui.R файла.

Каждая виджет-функция включает несколько аргументов. Первыми двумя аргументами для каждого виджета являются:

• **Имя** виджета. Пользователь не увидит это имя, но его можно использовать для доступа к значению виджет-функции.

• **Метка**. Она будет отображаться с виджетом в приложении. Метка должна быть строкой символов, но это может быть и пустой строка "".

Например, у виджет-функции actionButton("action_1", label = "ПРОДОЛЖИТЬ") **имя** "action_1" и **метка** "ПРОДОЛЖИТЬ":

Остальные аргументы варьируются от виджета к виджету, в зависимости от того, как виджет должен работать. Они включают такие параметры, как начальные значения, диапазоны и шаги. О них можно узнать на странице справки о виджет-функции (например,? SelectInput).

Ниже представлен сценарий ui.R, формирующий приложение «Основные виджеты» (рис. 5):

```
shinyUI(fluidPage(
 titlePanel("Основные виджеты"),

 fluidRow(

  column(3,
   h3("Кнопки"),
```

```
    actionButton("action", label = "ПРОДОЛЖИТЬ"),
    br(),
    br(),
    submitButton("Submit")),

  column(3,
    h3("Одиночный флажок"),
    checkboxInput("checkbox", label = "Выбор A", value = TRUE)),

  column(3,
    checkboxGroupInput("checkGroup",
      label = h3("Группа флажков"),
      choices = list("Вариант 1" = 1,
        "Вариант 2" = 2, "Вариант 3" = 3),
      selected = 1)),

  column(3,
    dateInput("date",
      label = h3("Ввод даты"),
      value = "2015-01-01"))
),

fluidRow(

  column(3,
    dateRangeInput("dates", label = h3("Диапазон дат"))),

  column(3,
    fileInput("file", label = h3("Ввод файла"))),

  column(3,
    h3("Текст справки"),
    helpText("Внимание: это не совсем виджет управления,",
      "но он обеспечивает простой способ добавлять текст",
      "для сопровождения других виджетов.")),

  column(3,
    numericInput("num",
      label = h3("Числовой ввод"),
```

```
      value = 1))
  ),

  fluidRow(

    column(3,
      radioButtons("radio", label = h3("Группа выбора"),
        choices = list("Вариант 1" = 1, "Вариант 2" = 2,
                       "Вариант 3" = 3),selected = 1)),

    column(3,
      selectInput("select", label = h3("Выбор варианта"),
        choices = list("Вариант 1" = 1, "Вариант 2" = 2,
                       "Вариант 3" = 3), selected = 1)),

    column(3,
      sliderInput("slider1", label = h3("Ползунки"),
        min = 0, max = 100, value = 50),
      sliderInput("slider2", "",
        min = 0, max = 100, value = c(25, 75))
      ),

    column(3,
      textInput("text", label = h3("Поле для ввода текста"),
        value = "Введите текст..."))
  )

))
```

Для изучения особенностей работы виджетов управления запустите приложение «Основные виджеты» с приведенным выше сценарием ui.R и попробуйте каждый виджет в действии.

3.1.2. Отображение реакции объектов

Рассмотрим, как осуществить автоматическое реагирование Shiny приложения на различные действия пользователя, когда он воздействует на объекты, обладающие реакцией (реактивные объекты), т.е. меняющие свои выходные значения при изменении входных воздействий.

Это делается в два этапа:

- добавляются нужные реактивные объекты в пользовательский интерфейс ui.R;
- пишется код сценария server.R, где для вывода используются выходные значения объектов.

Этап 1:

Shiny включает целое семейство функций вывода, которые формируют выходные значения объекта (табл. 8). Каждая функция формирует определенный тип выходного значения.

Выходные значения можно добавлять к пользовательскому интерфейсу аналогично тому, как добавлялись HTML элементы или виджеты. Размещать функции вывода внутри sidebarPanel() или mainPanel() в ui.R сценарии.

Таблица 8. Функции вывода выходных значений объекта

Функция вывода	Выводит
htmlOutput	чистый HTML
imageOutput	изображение
plotOutput	график
tableOutput	таблица
textOutput	текст
uiOutput	чистый HTML
verbatimTextOutput	текст

Например, в сценарии ui.R используется функция вывода textOutput() для добавления реакции в виде текстовой строки в основную панель mainPanel() приложения censusVis:

```
#ui.R
shinyUI(fluidPage(
 titlePanel("censusVis"),

 sidebarLayout(
  sidebarPanel(
   helpText("Формирование демографической карты
   по данным переписи населения США 2010 года."),

   selectInput("var",
    label = "Выберите переменную для отображения",
    choices = c("Процент белых", "Процент черных",
```

```
      "Процент испанцев", "Процент азиатов"),
    selected = "Процент белых"),

  sliderInput("range",
    label = "Диапазон интересов:",
    min = 0, max = 100, value = c(0, 100))
 ),

 mainPanel(
  textOutput("text1"),
  textOutput("text2")
 ))))
```

Отметим, что функция textOutput() использует аргументы “text1” и “text2”. Каждая функция вывода использует один аргумент в виде строковой переменной и Shiny использует ее как имя реакции объекта.

Этап 2:

Размещая функции вывода в ui.R мы просто указываем место, где будет отображаться реакция того или иного объекта. Для отображения реакции строится код в server.R. Код должен располагаться в “безымянной” функции располагаемой внутри shinyServer() в server.R сценарии. “Безымянные” функции играют специальную роль – они позволяют выстраивать объекты друг за другом путем указания нужных аргументов. Каждый объект описывается отдельно. Имена объектов должны соответствовать именам аргументов введенных в ui.R. Например, в сценарии server.R output$text1 соответствует textOutput("text1") в скрипте ui.R:

```
#server.R
shinyServer(
 function(input, output) {

  output$text1 <- renderText({
   paste("Вы выбрали: ", input$var)
})

   output$text2 <- renderText({
    paste("Выбран диапазон от", input$range[1], "до", input$range[2])
 })
  })
```

Каждое обращение к output должно осуществляться с помощью render* функций. Эти функции перехватывают R выражения и выполняют нужную предобработку. Необходимо использовать render* функции, которые соответствуют типу реакции объекта.

Каждая render* функция имеет один аргумент: R выражение заключенное в фигурные скобки {}. Выражение может состоять из одной строки кода, а может включать много команд. Его можно рассматривать как набор инструкций данных Shiny для запоминания. Shiny запустит эти инструкции когда пользователь первый раз запустит приложение и будет перезапускать их каждый раз когда выходные значения объекта изменятся при изменении входных значений пользователем.

Таблица 9. Соответствие render функций и создаваемых объектов

Render функция	Создает
renderImage	изображение (сохраняет ссылку на файл)
renderPlot	график
renderPrint	вывод на печать
renderTable	фрейм данных, матрица, другая структурированная таблица
renderText	символьная строка
renderUI	Shiny тег, объект или HTML

На рис. 26 представлен внешний вид приложения censusVis.

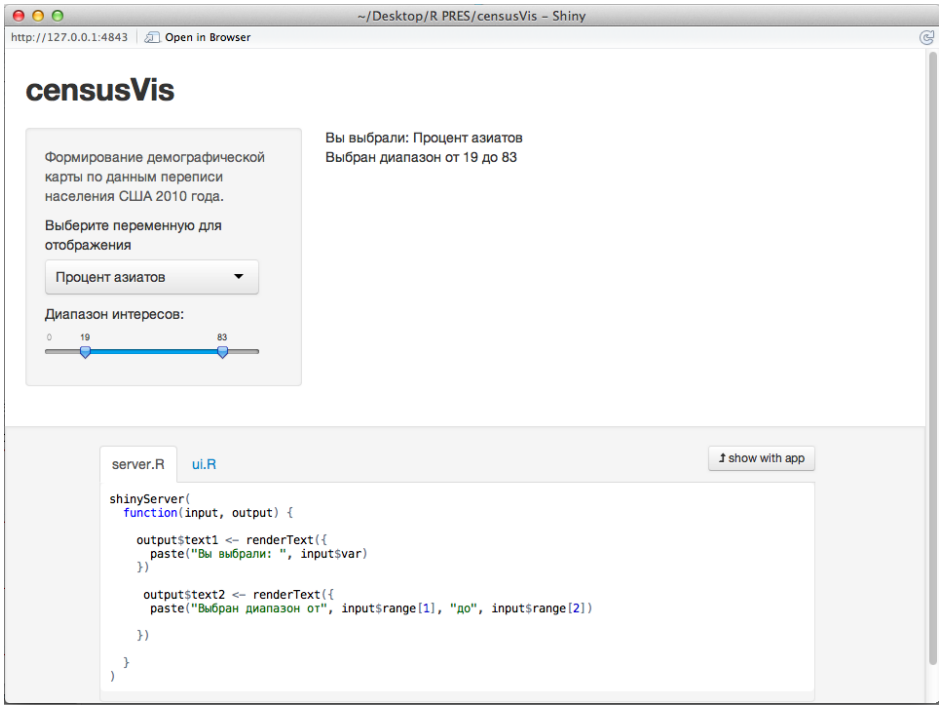


Рис. 26. Приложение censusVis

Для того, чтобы это работало, выражение должно возвращать правильный тип выходного значения (текст, график, фрейм данных и т.п.). В случае, если объект не возвратит значения или оно будет не соответствовать указанному типу, будет выведено сообщение об ошибке.

3.1.3. R скрипты и данные

Здесь рассмотрим как загружать данные, R скрипты и пакеты для того, чтобы использовать их в приложении. Сделаем это на примере построения достаточно сложного приложения, которое будет визуализировать данные переписи населения США .

Набор данных counties.rds

Будем использовать данные из counties.rds, которые содержат демографические показатели для каждого округа США, собранные в виде набор данных UScensus2010 R. Этот набор можно скачать из [16].

После скачивания набора данных создайте новую папку проекта, например census-app, в ней папку с именем data и загрузите туда файл counties.rds. Набор данных в counties.rds включает:

- название каждого округа США;
- общую численность населения округа;
- процент жителей в округе - белых, черных, латиноамериканцев или азиатов.

Проверить содержимое файла counties.rds можно следующим образом:

```
> counties <- readRDS("census-app/data/counties.rds")
> head(counties)
```

Таблица 10. Демографические данные для округов США (первые пять строк)

	name	total.pop	white	black	hispanic	asian
1	alabama,autauga	54571	77.2	19.3	2.4	0.9
2	alabama,baldwin	182265	83.5	10.9	4.4	0.7
3	alabama,barbour	27457	46.8	47.8	5.1	0.4
4	alabama,bibb	22915	75.0	22.9	1.8	0.1
5	alabama,blount	57322	88.9	2.5	8.1	0.2
6	alabama,bullock	10914	21.9	71.0	7.1	0.2

Скрипт helpers.R

Для создания приложения нам еще потребуется R скрипт helpers.R, который помогает формировать тематические карты. Эти карты позволяют закрашивать отдельные регионы различными цветами, в зависимости от некоторого параметра. В нашем случае helpers.R сформирует карту с помощью функции percent_map(), которая позволит раскрасить округи США в соответствии с процентами жителей из counties.rds. Загрузить helpers.R можно отсюда [17]. Сохраните скрипт helpers.R в папке census-app.

Скрипт helpers.R использует в RStudio два пакета: maps и mapproj. Поэтому, если они не установлены ранее, то их нужно установить:

```
> install.packages(c("maps", "mapproj"))
```

Функция percent_map() из helpers.R имеет пять аргументов (табл. 11).

Таблица 11. Аргументы функции percent_map()

Аргумент	Вход
var	вектор столбец из набора данных counties.rds
color	название используемого цвета
legend.title	заголовок для обозначений
max	диапазон затемнения (по умолчанию 100)
min	диапазон затемнения (по умолчанию 0)

Можно использовать функцию percent_map() в командной строке для построения тематической карты для столбца white из таблицы counties.rds следующим образом:

```
> library(maps)
> library(mapproj)
> source("census-app/helpers.R")
> counties <- readRDS("census-app/data/counties.rds")
> percent_map(counties$white, "darkgreen", "% white")
```

В результате функция percent_map() сформирует тематическую карту (рис. 27). На ней изображены проценты белого населения counties$white в округах США, с использованием темно-зеленого цвета "darkgreen", и заголовком обозначений "% white".

Загрузка файлов и указание путей

Обратите внимание на приведенный выше код. Для того, чтобы использо-

вать функцию percent_map() необходимо было сначала запустить скрипт helpers.R, включающим эту функцию, и затем загрузить counties.rds с функцией readRDS(). Также необходимо было запустить библиотеки library(maps) и library(mapproj).

Очевидно, необходимо сообщить Shiny вызывать те же функции перед использованием percent_map() в приложении когда пользователь будет изменять входные значения. Для этого нужно указать пути к файлам где эти функции описаны. Когда Shiny запускает команды сценария server.R, то все пути к файлам рассматриваются по отношению к директорию server.R. Другими словами, директорий где хранится server.R становится рабочим директорием всего приложения.

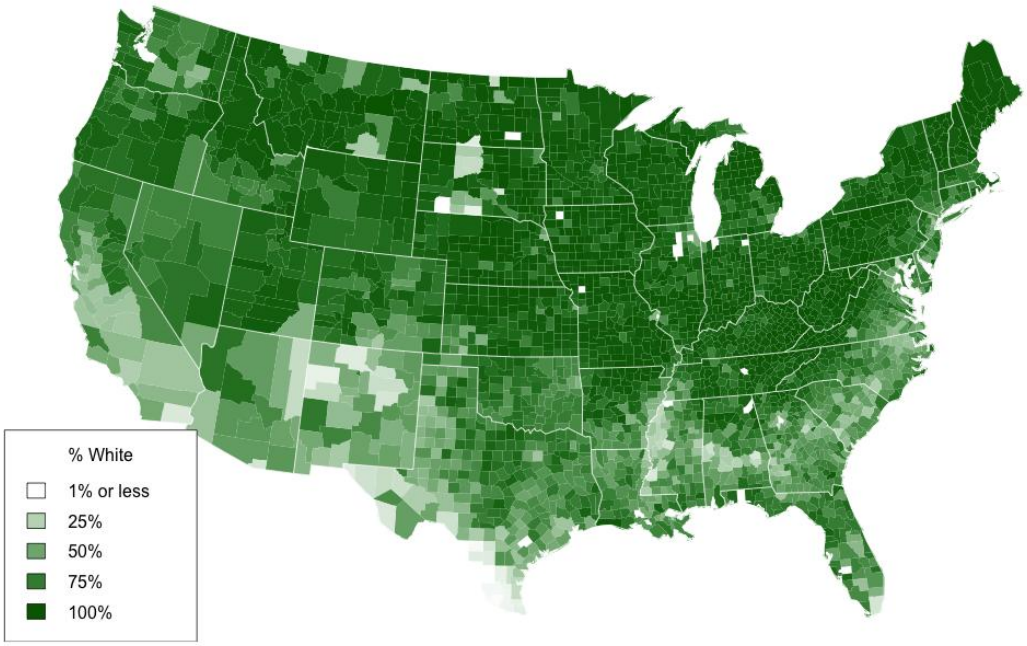


Рис. 27. Тематическая карта населения США

Поскольку helpers.R сохранен в том же директории, что и server.R, нужно сообщить об этом Shiny следующим образом:

```
> source("helpers.R")
```

Поскольку файл counties.rds находится в поддиректории (поименованном data) директория где находится server.R, то его загружаем так:

```
> counties <- readRDS("data/counties.rds")
```

Пакеты maps и mapproj загружаем как обычно:

```
> library(maps)
> library(mapproj)
```

здесь не требуется указывать пути к файлам.

Особенности выполнения скриптов

Shiny будет выполнять все команды, размещенные в сценарии server.R. Однако, от того где их поместить в server.R, будет зависеть сколько раз они будут запускаться (или перезапускаться), что, в свою очередь, влияет на производительность приложения.

Shiny будет запускать некоторые секции сценария server.R чаще, чем другие. Когда приложение вызывается в первый раз будет запущен весь сценарий. Shiny выполнит shinyServer() и сохранит «безымянную» функцию до следующего пользователя приложения (рис. 28). Каждый раз, когда пользователь обращается к приложению, Shiny запускает «безымянную» функцию. Таким образом эта функция «помогает» выстраивать различные наборы реактивных объектов для каждого пользователя.

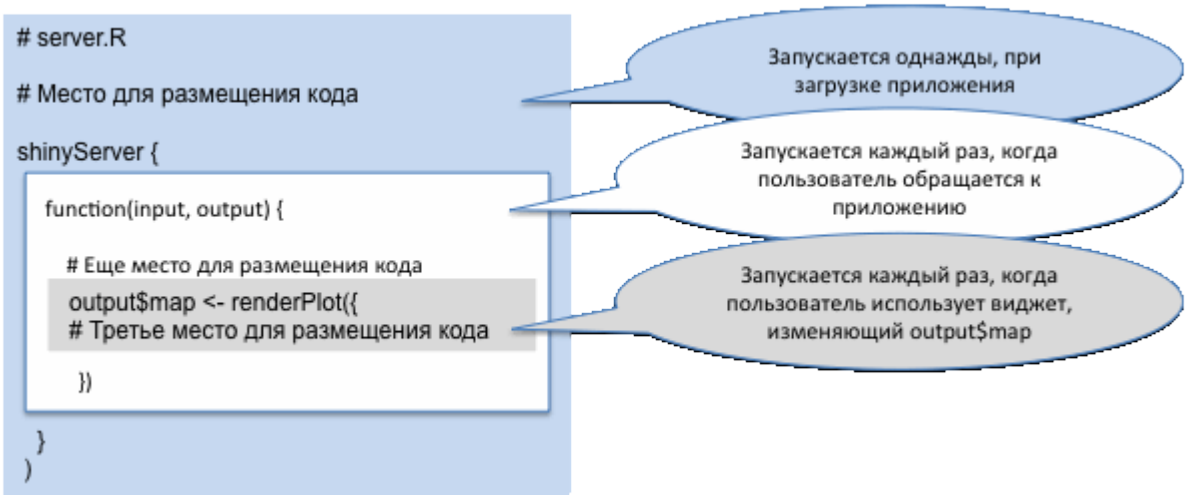


Рис. 28. Запуск различных секций сценария server.R

Наконец, когда пользователь будет использовать виджет для управления объектом, Shiny будет перезапускать выражения, соответствующие каждому объекту. Если пользователь очень активен, эти выражения могут быть повторно запущены много-много раз в секунду.

Покажем, как следует использовать эту информацию.

Исходные скрипты, нужные библиотеки и наборы данных необходимо загрузить в начале сценария server.R, за пределами функции shinyServer(). Shiny запустит их один раз и настроит сервер для запуска R выражений, содержащихся в shinyServer().

Определить конкретного пользователя объекта внутри «безымянной» функции function(), но за пределами каких-либо render*() вызовов. Они будут объектами, закрепленными за конкретными пользователями. Например, объект, который записывает информацию о сеансе пользователя. Этот код будет выполняться один раз для каждого пользователя.

Наконец, код который нужно часто перезапускать следует разместить внутри render*() функции. Shiny будет перезапускать отдельные «куски» этого кода каждый раз, когда пользователь использует для управления виджет упомянутый в этом «куске». Это может быть довольно часто.

Следует вообще избегать размещения кодов внутри render*() функции, которые не должны быть там. Такой код будет замедлять работу всего приложения.

Завершение построения приложения

Скопируйте и вставьте следующие сценарии ui.R и server.R в папку переписи приложения census-app:

```
#ui.R
shinyUI(fluidPage(
 titlePanel("censusVis"),

 sidebarLayout(
  sidebarPanel(
   helpText("Демографическая карта составлена
    на основе данных переписи 2010"),

   selectInput("var",
    label = "Выберите переменную для отображения на карте",
    choices = c("Белые", "Черные",
     "Латиноамериканцы", "Азиаты"),
    selected = "Белые"),

   sliderInput("range",
    label = "Диапазон представления:",
    min = 0, max = 100, value = c(0, 100))
  ),
```

```
   mainPanel(plotOutput("map"))
  )
))

#server.R
library(maps)
library(mapproj)
counties <- readRDS("data/counties.rds")
source("helpers.R")

shinyServer(
  function(input, output) {
    output$map <- renderPlot({
      data <- switch(input$var,
        "Белые" = counties$white,
        "Черные" = counties$black,
        "Латиноамериканцы" = counties$hispanic,
        "Азиаты" = counties$asian)

     color <- switch(input$var,
        "Белые" = "darkgreen",
        "Черные" = "black",
        "Латиноамериканцы" = "darkorange",
        "Азиаты" = "darkred")

      legend <- switch(input$var,
        "Белые" = "% БЕЛЫХ",
        "Черные" = "% ЧЕРНЫХ",
        "Латиноамериканцы" = "% ЛАТИНОАМЕРИКАНЦЕВ",
        "Азиаты" = "% АЗИАТОВ")

      percent_map(var = data,
        color = color,
        legend.title = legend,
        max = input$range[2],
        min = input$range[1])
    })
  })
```

Приложение censusVis имеет один реактивный объект – карта (map). Сценарий построен с использованием функции percent_map(), которая рисует карту и имеет пять аргументов.

Первые три аргумента var, color и legend.title зависят от значения выбора виджета окна. Последние два аргумента max и min определяются минимальным и максимальным значением ползунка виджета.

В сценарии server.R показан простейший способов формирования первых трех аргументов для функции percent_map(), основанный на одном выборе значения input$var.

3.1.4. Использование реактивных выражений

Shiny приложение просто поражает пользователей мгновенным быстродействием. Но что делать, если приложение вынуждено ожидать реакцию от некоторых «заторможенных» объектов?

Покажем, как можно модернизировать Shiny приложение путем использования реактивных выражений. Реактивные выражения позволяют управлять тем, когда и какие части приложения требует обновления, а также когда и какие обновления доступны, что предотвращает неэффективные затраты времени.

Для демонстрации этих возможностей создадим приложение stockVis. Создадим новую папку с именем stockVis в рабочем каталоге RStudio и поместим в нее файлы: ui.R, server.R, helpers.R:

```
#ui.R
shinyUI(fluidPage(
  titlePanel("stockVis"),
    sidebarLayout(
    sidebarPanel(
      helpText("Выберите акции для анализа.
        Информация предоставляется фирмой Yahoo finance."),
        textInput("symb", "ВВЕДИТЕ ТИКЕР АКЦИИ", "SPY"),
      dateRangeInput("dates",
        "ВЫБЕРИТЕ ДИАПАЗОН ДАТ",
        start = "2013-01-01",
        end = as.character(Sys.Date())),
      br(),
      checkboxInput("log", "Логарифмическая шкала по Y",
        value = FALSE),
      checkboxInput("adjust",
```

```
      "Адаптировать с учетом инфляции", value = FALSE)
   ),
  mainPanel(plotOutput("plot"))
 )
))
```

 # server.R

```
library(quantmod)
source("helpers.R")
shinyServer(function(input, output) {
 output$plot <- renderPlot({
  data <- getSymbols(input$symb, src = "yahoo",
   from = input$dates[1],
   to = input$dates[2],
   auto.assign = FALSE)
   chartSeries(data, theme = chartTheme("white"),
   type = "line", log.scale = input$log, TA = NULL)
 })
})
```

#helpers.R

```
if (!exists(".inflation")) {
 .inflation <- getSymbols('CPIAUCNS', src = 'FRED',
   auto.assign = FALSE)
}
adjust <- function(data) {
   latestcpi <- last(.inflation)[[1]]
   inf.latest <- time(last(.inflation))
   months <- split(data)
   adjust_month <- function(month) {
    date <- substr(min(time(month[1]), inf.latest), 1, 7)
    coredata(month) * latestcpi / .inflation[date][[1]]
   }
   adjs <- lapply(months, adjust_month)
   adj <- do.call("rbind", adjs)
   axts <- xts(adj, order.by = time(data))
   axts[ , 5] <- Vo(data)
   axts
}
```

Запустим приложение stockVis:

```
runApp("stockVis")
```

и получим «картинку» (рис. 29). Приложение выводит линейный график изменения стоимости акций выбранной компании за выбранный интервал дат. Используемые виджеты управления позволяют:

1. Выбрать компанию, набрав тикер акции.
2. Задать диапазон дат.
3. Выбрать логарифмический масштаб представления по оси Y.
4. Выбрать адаптацию цен с учетом инфляции.

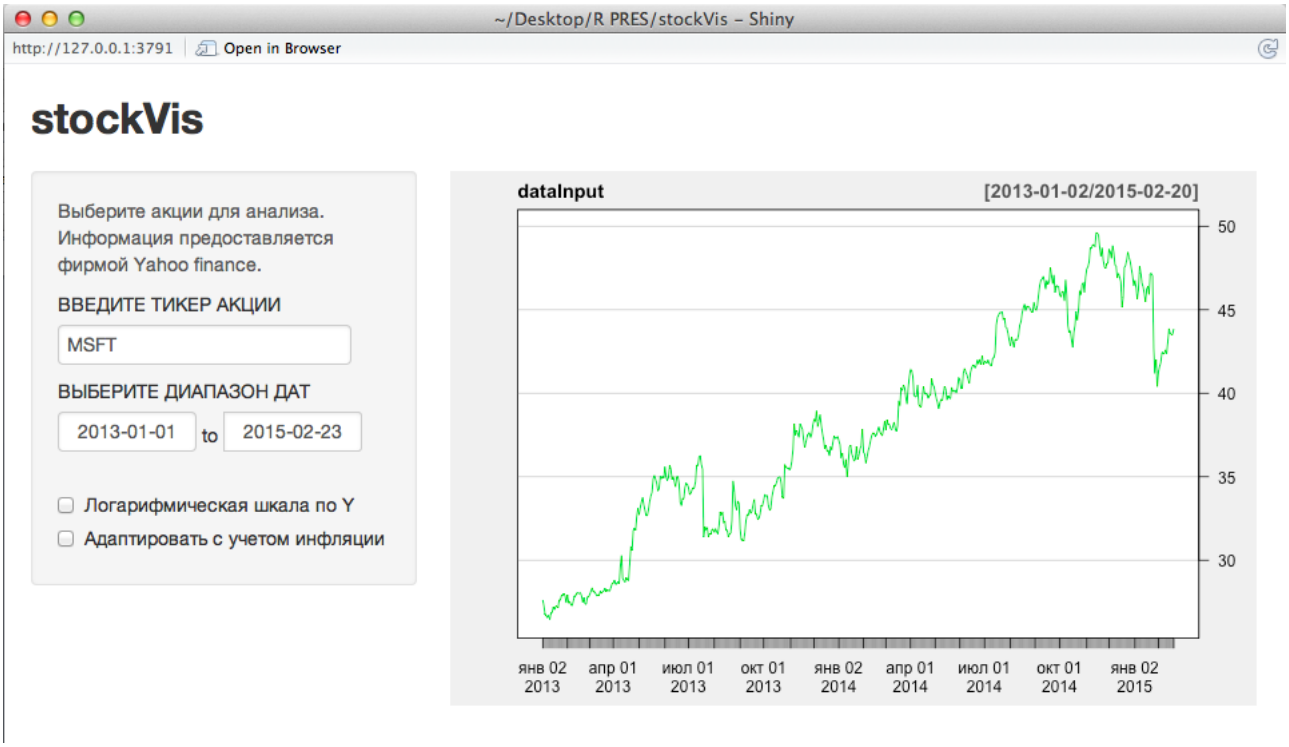


Рис. 29. Внешний вид приложения stockVis

Отметим, что флажок «Адаптация цен с учетом инфляции» пока не работает, для него введем соответствующий код позже.

По умолчанию stockVis отображает график для компании Майкрософт (тикер акции MSFT). Для получения информации о других компаниях необходимо ввести соответствующий тикер в поле ввода. Необходимо, чтобы тикер акции был распознан на сайте фирмы Yahoo finance. Найти перечень всех тикеров акций используемых Yahoo можно здесь [18]. Вот некоторые тикеры обще-

известных компаний: AAPL (Apple), AMZN (Amazon), KO (Coca-Cola), CSCO (Cisco), F(Ford) и GOOG (Google).

В сценарии server.R приложения StockVis используется две функции из пакета quantmod:

- getSymbols() – обеспечивает скачивание финансовые данных прямо в RStudio с таких веб-сайтов, как Yahoo finance [18] Yahoo финансов и Federal Reserve Bank of St. Louis [19];
- ChartSeries() - для отображения цен в привлекательном графике.

Сценарий helpers.R приложения StockVis содержит функцию, которая позволяет регулирует цены на акции с учетом инфляции. Эту функцию мы в дальнейшем используем для обслуживания флажка «Адаптация цен с учетом инфляции».

Флажки и диапазоны дат

Приложение StockVis использует несколько новых виджетов - селектор диапазона дат, созданный с помощью функции dateRangeInput() и пара флажков, обслуживаемых функцией checkboxInput().

Виджеты флажков использовать очень просто - они возвращают значение TRUE, когда флажок взведен и FALSE, когда флажок не установлен. В сценарии ui.R флажки названы log и adjust, что означает возможность их использования как input$log и input$adjust в качестве входных значений в сценарии server.R.

Оптимизация вычислений

Приложение StockVis имеет следующий недостаток. Посмотрите, что произойдет после установки флажка «Логарифмическая шкала по Y» - это вызовет изменение значения input$log и перезапуск всего выражения в renderPlot():

```
output$plot <- renderPlot({
 data <- getSymbols(input$symb, src = "yahoo",
  from = input$dates[1],
  to = input$dates[2],
  auto.assign = FALSE)

 chartSeries(data, theme = chartTheme("white"),
  type = "line", log.scale = input$log, TA = NULL)
      })
```

Недостаток заключается в том, что renderPlot() автоматически перезапускается каждый раз, когда:

- пользователь выбирает данные из Yahoo Finance с помощью функции getSymbols();

- пользователь выбирает логарифмический масштаб оси Y с помощью функции chartSeries().

Это плохо, потому что при изменении масштаба оси совсем не нужно заново обновлять данные о стоимости акций. В самом деле, Yahoo Finance будет отказывать в обслуживании, если слишком часто повторно запрашивать данные (это выглядит как атака робота). Но что еще более важно, повторный запуск getSymbols() вообще не требуется, он просто приведет к замедлению работы приложения и уменьшит пропускную способность сервера.

Реактивные выражения

Реактивным выражением называется R выражение (функция), которое используется для обслуживания виджета ввода, возвращающего значение. Реактивные выражения позволяют ограничить повторный запуск определенных функций. Оно вычисляется только в том случае, когда изменяется выходное значение виджета.

Для создания реактивного выражения используется reactive функция, R выражение которой заключается в фигурные скобки {}. Например, реактивное выражение, которое использует виджеты stockVis для извлечения данных из Yahoo:

```
dataInput <- reactive({
  getSymbols(input$symb, src = "yahoo",
    from = input$dates[1],
    to = input$dates[2],
    auto.assign = FALSE)
})
```

При запуске этого выражения запускается функция getSymbols(), которая возвращает диапазон дат. В функции renderPlot() это выражение для доступа к данным о диапазоне дат используется как вызов dataInput():

```
output$plot <- renderPlot({
  chartSeries(dataInput(), theme = chartTheme("white"),
    type = "line", log.scale = input$log, TA = NULL)
})
```

Реактивные выражения немного «умнее» регулярных функций R. Они кэшируют выходные значения и «узнают», когда эти значения устарели. Это означает, что первый раз, когда запускается реактивное выражение, оно сохраняет результат в памяти компьютера. В следующий раз при вызове реактивного выражения, оно может вернуть этот сохраненный результат, не делая никаких вычислений (что сделает приложение быстрее).

Реактивное выражение вернет сохраненный результат, если «знает», что он актуален. Если реактивному выражению «стало известно», что результат является устаревшим (потому что виджет изменил его значение), выражение пересчитает результат, а затем возвратит новый результат и сохранит его новую копию. Реактивное выражение будет использовать эту новую копию, пока та также не устареет.

Понятно, что можно использовать это поведение, чтобы предотвратить Shiny от повторного запуска ненужного кода. Рассмотрим, как реактивное выражение будет работать в новом stockVis приложение:

```
# server.R
shinyServer(function(input, output) {
  dataInput <- reactive({
   getSymbols(input$symb, src = "yahoo",
     from = input$dates[1],
     to = input$dates[2],
     auto.assign = FALSE)
  })

  output$plot <- renderPlot({
   chartSeries(dataInput(), theme = chartTheme("white"),
     type = "line", log.scale = input$log, TA = NULL)
  })
        })
```

При установке флажка «Логарифмическая шкала по Y» значение input$log изменится и функция renderPlot() будет повторно выполняться, при этом: renderPlot() вызовет функцию dataInput (), которая проверит, что даты не изменились и вернет свои сохраненные данные без повторного запроса к серверу Yahoo. Функция renderPlot() перерисует график с коррекцией формата шкалы оси Y.

Зависимости

Что произойдет, если пользователь изменяет содержимое виджета тикер акции, т.е. значение переменой symb?

Это сделает график, нарисованный функцией renderPlot() не актуальным, но renderPlot() не будет вызывать input$symb. Тем не менее, Shiny перерисует график, благодаря тому что отслеживает, от какого виджета зависит реактивные выражения объекта вывода. Shiny автоматически заново строит объект, если:

- входное значение у объектов из render* функции изменяется;
- реактивное выражение у объектов из render* функции устаревает.

Реактивные выражения можно представить, как звенья одной цепи, которые связывают входные и выходные значения объектов. Выходные значения объектов будет реагировать на изменения, внесенные в любом месте ниже по течению в цепочке. (Можете составить длинную цепочку, потому что реактивные выражения могут вызывать другие реактивные выражения).

Реактивные выражения можно вызывать только внутри reactive или render* функции. Потому, что только эти функции способны обслуживать реактивные выходы, которые могут быть изменены без предупреждения.

Оптимизация обслуживания флажка

Приступим к «оживлению» флажка «Адаптация цен с учетом инфляции». Пользователь должен иметь возможность для переключения между ценами с учетом инфляции и не скорректированными ценами.

Функция регулировки в helpers.R использует данные индекса потребительских цен, предусмотренных федеральным резервным банком Сент-Луиса, чтобы преобразовать цены. Посмотрим, как ее можно реализовать в приложении. Вот одно из решений, которое не является идеальным:

```
# server.R
shinyServer(function(input, output) {
 dataInput <- reactive({
  getSymbols(input$symb, src = "yahoo",
     from = input$dates[1],
     to = input$dates[2],
     auto.assign = FALSE)
 })
 output$plot <- renderPlot({
  data <- dataInput()
  if (input$adjust) data <- adjust(dataInput())
  chartSeries(data, theme = chartTheme("white"),
     type = "line", log.scale = input$log, TA = NULL)
 })
       })
```

Функция adjust() вызывается внутри renderPlot(). Если флажок установлен, приложение будет преобразовывать все цены каждый раз.

Устранить эту проблему можно путем добавления нового реактивного выражения. Реактивное выражение должно взять значение dataInput() и вернуть преобразованную или непреобразованную копию данных:

```
# server.R
```

```
shinyServer(function(input, output) {
 dataInput <- reactive({
    getSymbols(input$symb, src = "yahoo",
       from = input$dates[1],
       to = input$dates[2],
       auto.assign = FALSE)
 })
 finalInput <- reactive({
  if (!input$adjust) return(dataInput())
  adjust(dataInput())
 })
 output$plot <- renderPlot({
  chartSeries(finalInput(), theme = chartTheme("white"),
      type = "line", log.scale = input$log, TA = NULL)
 })
       })
```

Теперь каждый вход выделен в собственном реактивном выражении или render* функции. Если вход изменяется, то повторно запускаться будут только устаревшие выражения.

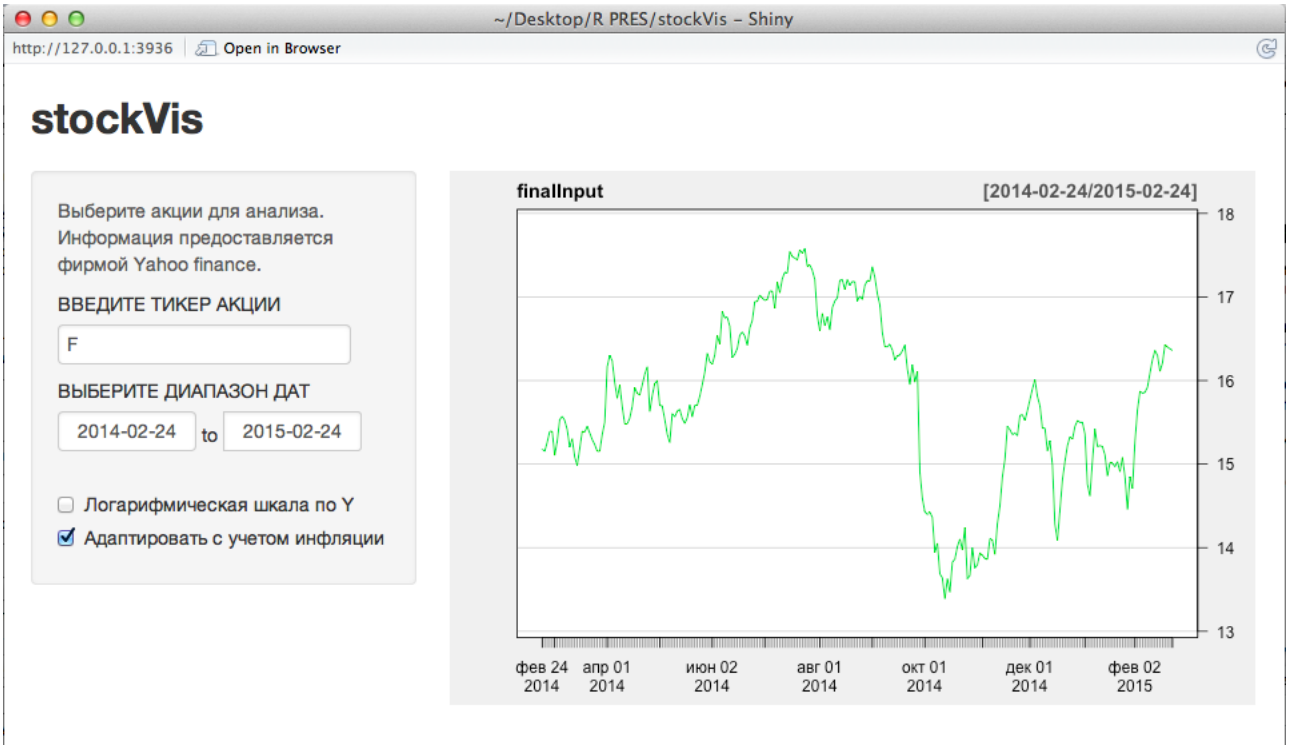


Рис. 30. Работа флажка «Адаптация цен с учетом инфляции»

Вот пример последовательности работы этого приложения:

- устанавливаем флажок «Логарифмическая шкала по Y» ;
- renderPlot() перезапускается;
- в renderPlot() вызывается finalInput();.
- finalInput() проверяет dataInput() и input$adjust;
- если ничего ни изменилось, finalInput() возвращает сохраненное значение;
- если что-либо изменилось, finalInput() вычисляет новое значение с текущими входами, после чего вычисляется новое значение renderPlot() и сохраняется для будущих запросов.

3.1.5. Использование Shiny приложений

После того, как приложение создано оно может быть использовано для различных целей. Возможны два основных варианта его использования:

- в виде двух файлов: server.R и ui.R, что потребует наличие среды R;
- в виде веб-страницы, которая будет доступна для любого браузера.

В заключении данного раздела обратим внимание на интернет ресурс [20], где можно найти много полезной дополнительной информации о пакете Shiny.

3.2. Разработка веб презентаций

3.2.1. Использование пакета knitr

Этот пакет предоставляет инструмент общего назначения для динамической генерации отчетов и презентаций в R, в которых могут быть использованы любые языки разметки, в том числе Sweave, HTML, Markdown, ReStructuredText, AsciiDoc и Textile [21]. Полученный документ может демонстрироваться непосредственно в браузере. Пакет knirt обеспечивает целый ряд полезных свойств:

- свойство прозрачности, которое означает, что пользователь имеет полный доступ к каждой части документа при его формировании [22];
- knitr автоматически запускает R код при демонстрации документа, как в среде Rstudio, так и в браузере;
- дизайн knitr позволяет использовать разные языки сценариев (например, R, Python и AWK) и языки выходной разметки (например, LaTeX, HTML, Markdown, AsciiDoc и ReStructuredText) [23].

Для разработки веб презентаций требуется RStudio v0.98 или более поздняя версия. Начиная с этой версии, пользователям предоставляется возможность очень просто создавать HTML5 презентации, без знания HTML и JavaScript. R презентации базируются на R Markdown и характеризуются следующими особенностями:

- очень простой синтаксис (Markdown);
- автоматический вывод результатов выполнения включенного в слайд R кода (в том числе графиков и изображений);
- поддержка LaTeX уравнений с использованием MathJax;
- гибкая раскладка контента слайда на два столбца;
- множество вариантов для переходов между слайдами и слайд - навигации;
- возможность настраивать внешний вид слайдов с использованием CSS;
- возможность предварительного просмотра в рамках RStudio;
- может воспроизводится внутри RStudio или как автономные презентации HTML5 в веб-браузере;
- может быть легко опубликована для общего веб доступа в RPubs.

Цель R презентаций, состоит в том, чтобы сделать создание авторских слайдов, которые используют R код и LaTeX уравнения так просто, как это возможно. Они особенно полезны для учебного использования, поскольку их можно демонстрировать непосредственно из среды, которая используется для написания и изучения кода.

Начало разработки

Чтобы создать новую R презентацию нужно открыть новый файл **New File -> R Presentation**:

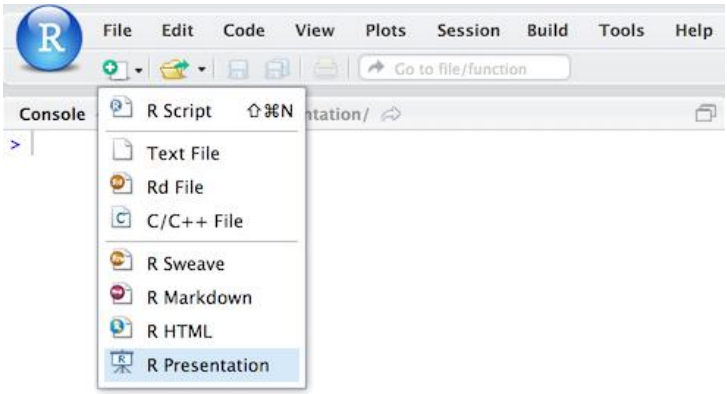


Рис. 31. Создание новой презентации

После указания места для сохранения презентации и ее названия, будет создан шаблон новой презентации и ее предварительный просмотр

будет доступен в правом верхнем окне RStudio на вкладке Presentation. Например, создадим новую презентацию с именем Example, для нее автоматически сформируется шаблон:

````
Example
========================================================
author:
date:

First Slide
========================================================
For more details on authoring R presentations click the
**Help** button on the toolbar.
- Bullet 1
- Bullet 2
- Bullet 3

Slide With Code
========================================================
```{r}
summary(cars)
```

Slide With Plot
========================================================
```{r, echo=FALSE}
plot(cars)
```
````

Шаблон появится в панели Source (левый верхний угол RStudio), а также в созданном файле Example.Rpres. Как видно из приведенного примера, шаблон состоит из четырех слайдов:

- Example (титульный слайд с названием презентации).
- First Slide (с примером ненумерованного списка).
- Slide With Code (с примером R кода).
- Slide With Plot (с примером графика).

Можно изменить названия слайдов и их содержимое и посмотреть, как RStudio «отрабатывает» эти изменения в правом верхнем окне RStudio на вкладке Presentation. Для того, чтобы посмотреть «отработку», после изменении содержимого шаблона нажмите иконку Preview.

Основы формирования слайдов

Заголовки слайдов от их содержимого разделяются с помощью =====. Вот, пример первого слайда:

```
Обработка информации в среде RStudio
==========================================================
![image](authors.png) <br>
Александр Николаевич Губин <br>
Феликс Васильевич Филиппов <br>
ИСиТ, кафедра ИУС, spb.sut.ius@mail.ru
```

и его отображение на вкладке Presentation (рис. 32)

Рис. 32. Титульный слайд презентации

Название презентации автоматически отображается в формате заголовка H1 и слайд окрашен специальным цветовым фоном. Здесь также показано, как можно вставить изображение (![image]) и «заставить» писать текст с новой строки (<br>).

Форматирование текста

Контент в R презентациях форматируется с помощью стандартного синтаксиса Markdown. Слайд шаблона First Slide (с примером ненумерованного списка) приведенный выше, является одним из примеров этого синтаксиса. Синтаксис Markdown очень прост и с ним можно ознакомиться прямо в RStudio, выбрав в меню HELP пункт Markdown Quick Reference.

Фрагменты R кода

R презентации могут включать в себя фрагменты R кода. Код может быть использован для иллюстрации способов программирования, а также для автоматической визуализации результата его выполнения. Примером может служить слайд шаблона Slide With Code (с примером R кода), приведенный выше.

Для отображения только результата выполнения R кода без отображения самого кода нужно указать значение атрибута echo = FALSE. Примером может служить слайд шаблона Slide With Plot (с примером графика), приведенный выше.

Наконец, для демонстрации только самого кода, без результатов его выполнения, следует использовать значение атрибута eval=FALSE, например:

```
Слайд с одним кодом
==========
```{r, eval=FALSE}
summary(cars)
```
```

Отметим, что при добавлении любого атрибута к {r} уже после ввода запятой RStudio автоматически высвечивает набор доступных атрибутов и при выборе конкретного атрибута предоставляет на выбор перечень его возможных значений.

Наконец, подчеркнем, что фрагменты кода могут быть также использованы для включения графиков в презентации. Примером может служить слайд шаблона Slide With Plot (с примером графика)), приведенный выше.

Кэширование

Каждый раз, когда презентация сохраняется она автоматически обновляется, повторным запуском knitr для презентации. Если это начинает отнимать много времени из-за длинных вычислений или прорисовки больших графиков, то можно использовать knitr кэширование для повышения производительности. В документации по knitr описано, как работает кэширование. а кэш примеры демонстрируют дополнительные подробности.

Если необходимо включить кэширование глобально для всей презентации, можно включить фрагмент следующего кода, в начале документа:

```
```{r setup, include=FALSE}
opts_chunk$set(cache=TRUE)
```
```

Изображения

Вставлять изображения в пределах слайдов можно с использованием стандартного markdown синтаксиса: ![alt text](myimage.png). Если необходимо,

чтобы изображение было влево или вправо от текста, просто разбейте слайд на колонки (макеты двух колонок обсуждается более подробно ниже), например:

```
Слайд с изображением слева
==================================
![alt text](myimage.png)
***

Этот текст появится справа.
```

Есть несколько специальных правил относительно того, как изображения в пределах презентаций располагаются:

- в колонке, изображения заполняют все доступное пространство;
- когда слайд содержит только изображение, оно будет заполнять все свободное пространство;
- изображения, которые появляются вместе с текстом ограничены до 50% по вертикали.

Обратите внимание, что эти правила также регулируют встраивание в слайд изображений, создаваемых «на лету» во фрагменте R кода.

Формулы с MathJax

Можно вставлять LaTeX или MathML формулы в R презентации, используя следующий синтаксис:

- $формула$ для формул, встроенных в текстовую строку (обратите внимание, что не должно быть пробелов, прилегающих к $ разделителям);
- $$ формула $$ для отображения формул вне текстовых строк.

Ниже приведены примеры синтаксиса, используемого для формул встроенных в текстовую строку:

```
Среднее арифметическое $x_(cp)=\frac1{n}\sum_{i=1}^{n}x_{i}$
равно сумме n членов, деленной на n.
```

и отдельных формул вне строк:

```
$$
\begin{aligned}
\dot{x}&=\sigma(y-x)\\
\dot{y}&=\gamma - y - xz\\
\dot{z}&=-\beta z + xy
\end{aligned}
$$
```

Более подробную информацию о встраивании формул в R презентации можно найти в [24].

Макеты двух колонок

Можно создать макет слайда из двух столбцов с помощью markdown горизонтальной линии (***). Например:

```
Слайд с двумя равными колонками
==================================
Первая колонка
***
Вторая колонка
```

По умолчанию столбцы разделяют пространство слайда поровну. Если необходимо, чтобы один из столбцов получал больше места, можно это указать с помощью атрибута left (или right), например:

```
Слайд с двумя неравными колонками
==================================
left: 70%
Слева занято 70% пространства
***
Справа 30%
```

Переходы между слайдами

По умолчанию смена слайдов "линейная" - обычное передвижение слайдов справа налево. Можно указать альтернативный стиль перехода, используя атрибут перехода transition. Например:

```
Название презентации
======================================
Автор: Михаил Иванов
Дата: 26 февраля 2015
transition: zoom
```

Допустимые значения для атрибута перехода transition включают в себя:

. none
. linear
. rotate

. fade
. zoom
. concave

Чтобы указать стиль перехода глобально, для всего набора слайдов, значение атрибута transition задается на первом слайде. Можно также указывать различные значения атрибута перехода на отдельных слайдах.

Кроме стиля, можете задавать скорость переходов (глобально или на каждого слайда), используя атрибут transition-speed. Допустимые значения для transition-speed включают в себя:

- default
- slow
- fast

Обратите внимание, что переходы между слайдами, не регулируются при просмотре презентации в рамках RStudio (они формируются при просмотре в браузере).

Типы слайдов

Для каждого слайда может быть указан его тип для, который изменяет по умолчанию внешний вид слайда. Есть четыре встроенных типа:

- section
- sub-section
- prompt
- alert

Например:

```
Новый раздел
==================================
type: section
```

или

```
Слайд подсказка
==================================
type: prompt
```

Типы раздел и подраздел используют особый фон и цвет шрифта, немного больший размер текста заголовка. Типы подсказка и предупреждение также отличаются по цветам фона, шрифтам и размещению заголовков.

Инкрементальный показ

Можно задать для отображения контента режим постепенного добавления фрагментов слайда, установкой значения true атрибута incremental. Например, слайд со списком, в котором установлено значение true для этого атрибута, будет отображать отдельные пункты списка постепенно, по нажатию «следующий»:

```
Слайд со списком
==================================
incremental: true
- Пункт 1
- Пункт 2
- Пункт 3
```

К фрагментам слайда относятся: заголовки, параграфы, цитаты, блоки кода, и элементы списка (первый пункт любого слайда всегда отображается сразу). Установка значения true атрибута `incremental` на титульном слайде задает это поведение по умолчанию для всей презентации.

Навигация

По умолчанию пользователи могут перемещаться к любому слайду в презентации с помощью меню навигации. Чтобы ограничить такую возможность можно использовать атрибут навигации navigation на первом слайде. Допустимые значения атрибута navigation:

- none — навигация невозможна;
- section — пользователи могут перейти только к разделам;
- slide — пользователи могут перейти к любому слайду.

Добавлять гиперссылки на слайды можно, используя стандартный синтаксис markdown: [Фраза относящаяся к ссылке] (http://example.com). Можно также добавить атрибут id на слайде, который позволяет указать конкретную точку для ссылки. Например, если на каком-либо слайде презентации добавить атрибут id:show, то с любого другого слайда туда можно перейти так: [Перейти к просмотру] (# / show).

Открытие исходных файлов

Можно внешний файл с R кодом открыть автоматически для показа, добавив на слайде атрибут источника source с указанием имени файла, например source: example.R.

Для того, чтобы подсветить нужную строку в файле с кодом при показе, достаточно указать номер нужной строки через пробел: source: example.R 7.

3.2.2. Настройка шрифтов и внешнего вида

Шрифты

По умолчанию в R презентации используется шрифт Lato, однако можно настроить другой шрифт, используя атрибут font-family в первом слайде, например, указав font-family: 'Helvetica'. Семантика здесь такая же, как и для CSS семейства шрифтов (например, можно указать список разделенных запятыми альтернативных шрифтов).

Обратите внимание, что если вы используете по умолчанию шрифт Lato, то презентация гарантированно всегда будет демонстрироваться с этим шрифтом (так как он встроен в презентации). Если ли же указать альтернативный шрифт, то он должен быть доступен на той системе, где презентация отображается (в противном случае будет использован резервный шрифт).

Можно также импортировать веб-шрифты из пользовательского URL с помощью атрибута font-import. Например, это можно сделать так:

```
font-import: http://fonts.googleapis.com/css?family=Risque
font-family: 'Risque'
```

Следует иметь ввиду, что при импорте веб-шрифта требуется гарантированное подключение к интернету.

Если необходимо уменьшить размер букв для некоторых фрагментов текста, можете заключить этот фрагмент в теги <small>.

Например, <small> Это предложение будет выглядеть меньше. </small >.

Задание стилей

Для настройки отображения отдельных слайдов или фрагментов текста можно использовать CSS. Если таблицу стилей разместить в том же директории что и презентации, и с тем же базовым именем, то она будет автоматически импортирована. Например, если файл презентации ТОИ.Rpres то файл ТОИ.css будет импортирован, если он существует.

Также можете явно указать файл стилей, используя атрибут CSS на титульном слайде, например CSS: custom.css

Наконец, можно включить стили прямо в исходный файл презентации, разместив их в начале, перед титульным слайдом.

Применение стилей

Для применения стиля класса, определенного в CSS файле к отдельному слайду, надо добавить атрибут class, например:

```
Слайд
==================================
class: illustration
```

Для применения стиля класса к фрагменту текста, используется тег span, например:

```
Слайд
==================================
<span class="emphasized">Внимание!</span>
```

Если какой-либо стиль нужно применить ко многим фрагментам текста, можно определить его для тега del, а затем нужные фрагменты выделять с помощью двойных «~~» ограничителей. Например, в файле стилей определяем красный цвет для тег del:

```
.reveal section del {
  color: red;
```

Теперь следующий текст будет отображаться красным цветом:

~~ Этот текст будет красным ~~

Пользовательские типы

Как отмечалось выше, можно использовать четыре типа слайдов section, sub-section, prompt и alert, которые отличаются по цвету фона, шрифтам и размещению заголовков. Однако, можно создавать собственные типы слайдов, используя соответствующее описание в CSS. Например, создадим новый тип с названием schooldesk, для которого определим черный фон и белые буквы:

```
.schooldesk .reveal .state-background {
  background: black;
}
. schooldesk .reveal h1,
. schooldesk .reveal h2,
. schooldesk .reveal p {
  color: white;
}
```

Теперь можно задать слайду этот тип, используя следующий синтаксис:

```
Слайд
========================================
type: schooldesk
```

Следует отметить, что единственный способ, чтобы настроить цвет фона (или изображение) слайда является определение пользовательского типа.

КОНТРОЛЬНЫЕ ЗАДАНИЯ

Задание 1

Создайте веб презентацию, включающую три слайда следующего типа:

- Example (титульный слайд с названием презентации).
- Slide With Code (с примером R кода).
- Slide With Plot (с примером графика).

В качестве R кода возьмите пример из раздела «Фреймы», а в качестве графика пример из раздела «Интерфейс среды разработки».

Задание 2

Создайте веб презентацию демонстрирующую анимационные возможности файлов *.svg – масштабируемой векторной графики и файлов *gif анимации. Примеры файлов svg можно взять в http://www.03www.ru/2013-02/svg/index.html.

Задание 3

Создайте веб приложение Shiny с заголовком «Вэб юмор!», использующее на «Боковой панели» виджет «Числовой ввод». При вводе номера комикса приложение в «Основной панели» должно выводить соответствующий комикс с сайта: http://xkcd.com/.

Задание 4

Создайте веб приложение Shiny с заголовком «Анализ HTML контента», использующее на «Боковой панели» виджеты «Ввод файла» и «Выбор варианта». При каждом вводе нового файла с расширением HTML и выборе одного из

тегов <h1>, <h2>, <h3> или <p>, приложение в «Основной панели» должно выводить соответствующий контент.

Задание 5

Создайте Shiny приложение, в котором пользовательский интерфейс будет включать виджет «Ввод файла» и два виджета «Ползунок» в боковой панели. При выборе пользователем любого текстового файла приложение должно строить облако слов (Word Cloud) и выводить его на основную панель. Виджеты «Ползунок» должны позволять изменять параметры облака слов – минимальная частота повторения и количество слов.

Задание 6

Создайте Shiny приложение, в котором пользовательский интерфейс будет включать виджеты «Поле ввода» и «Группа выбора» в боковой панели. После ввода URL ресурса и выбора типа тега, приложение должно выводить смысловое содержимое выбранных тегов.

В группе выбора должны быть указаны следующие типы тегов: <a>,<p>, <h1>, <h2>, <h3>.

Задание 7

Создайте веб презентацию, включающую как минимум 7 слайдов, посвященных семи чудесам света. Каждый слайд должен содержать географическую карту места расположения чуда, время его создания и назначение сооружения.

Для формирования карты воспользуйтесь API Google, размещенном по адресу: https://developers.google.com/maps/documentation/staticmaps/.

Задание 8

Создайте веб приложение Shiny с заголовком «Золотое кольцо России», использующее на «Боковой панели» виджет «Выбор варианта» со списком всех городов, входящих в состав золотого кольца и плюс вариант «показать все кольцо» .

При выборе города в основную панель должна выводится карта города, а при выборе варианта «показать все кольцо» - карта всего кольца.

Для формирования карты воспользуйтесь API Google, размещенном по адресу: https://developers.google.com/maps/documentation/staticmaps/.

Задание 9

Создайте веб приложение Shiny, в котором после ввода названия геогра-

фического объекта в «Поле ввода», в основную панель выводятся его географические координаты.

Для решения задачи воспользуйтесь API Google и информацией по адресу: http://habrahabr.ru/post/110460/.

Задание 10

Создайте веб приложение Shiny, в котором сформированы два виджета типа «Поле ввода» - один для ввода географической широты, другой для ввода географической долготы. Виджет «Кнопка» должен служить для показа карты, соответствующей введенным географическим координатам.

Для решения задачи воспользуйтесь API Google и информацией по адресу: http://habrahabr.ru/post/110460/.

Задание 11

Создайте веб презентацию, включающую как минимум 7 слайдов, посвященных семи чудесам света. Каждый слайд должен содержать географическую карту места расположения чуда, время его создания и назначение сооружения.

Для формирования карты воспользуйтесь API Yandex, размещенном по адресу: https://tech.yandex.ru/maps/doc/staticapi/1.x/dg/concepts/input_params-docpage/.

Задание 12

Создайте веб приложение Shiny с заголовком «Золотое кольцо России», использующее на «Боковой панели» виджет «Выбор варианта» со списком всех городов, входящих в состав золотого кольца и плюс вариант «показать все кольцо» .

При выборе города в основную панель должна выводится карта города, а при выборе варианта «показать все кольцо» - карта всего кольца.

Для формирования карты воспользуйтесь API Yandex, размещенном по адресу: https://tech.yandex.ru/maps/doc/staticapi/1.x/dg/concepts/input_params-docpage/.

ИСПОЛЬЗОВАННЫЕ ИСТОЧНИКИ

1. http://cran.r-project.org/web/packages/
2. http://cran.r-project.org/doc/manuals/r-release/R-intro.pdf
3. http://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf
4. http://cran.r-project.org/manuals.html
5. http://cran.r-project.org
6. http://dirk.eddelbuettel.com/cranberries/
7. http://planetr.stderr.org
8. www.r-bloggers.com
9. stat.ethz.ch/mailman/listinfo/r
10. http://rstudio.org/download/desktop
11. http://en.wikipedia.org/wiki/Web_scraping
12. http://scrapy.org/
13. http://www.r-bloggers.com/interactive-javascript-in-r-with-v8-a-crossfilter-example
14. http://www.rdatamining.com/data
15. http://www.algorithmist.ru/2011/05/clustering-with-example-in-r.html
16. http://shiny.rstudio.com/tutoril/lesson5/census-app/data/counties.rds
17. http://shiny.rstudio.com/tutoril/lesson5/census-app/data/helpers.r
18. http://finance.yahoo.com/lookup
19. http://research.stlouisfed.org/fred2/
20. http://shiny.rstudio.com
21. http://cran.r-project.org/web/packages/knitr/index.html
22. http://yihui.name/knitr/
23. Yihui Xie «Dynamic Documents with R and knitr» ,Chapman & Hall, 2013
24. https://support.rstudio.com/hc/en-us/articles/200486328-Equations-in-R-Markdown