Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ УПРАВЛЕНИЯ»

Институт информационных систем
Кафедра математических методов в экономике и управлении

Утверждено
проректором
доц. А.В. Троицким
28 июня 2019 г.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторных работ по учебной дисциплине

«ПРАКТИКУМ НА ЭВМ 4»

для подготовки бакалавров по направлению
01.03.02 Прикладная математика и информатика

Москва – 2019

УДК 378.147.88:004(072)
6Н1
М54
Методические указания к выполнению лабораторных работ по учебной дисциплине «Практикум на ЭВМ 4» : для подготовки бакалавров по направлению 01.03.02 Прикладная математика и информатика [Текст] / Государственный университет управления, Институт информационных систем ГУУ, кафедра математических методов в экономике и управлении ; [сост. С.А. Суязова]. – М. : Издательский дом ГУУ, 2019. – 58 с.

Составитель
старший преподаватель кафедры математических методов
в экономике и управлении
С.А. СУЯЗОВА

Ответственный редактор
заведующий кафедрой математических методов в экономике и управлении,
кандидат экономических наук, доцент
О.М. ПИСАРЕВА

Обсуждено
на заседании кафедры математических методов в экономике и управлении
27 мая 2019 г.

Обсуждено и одобрено
на заседании Методического совета
Института информационных систем ГУУ
18 июня 2019 г.

Рецензент
доцент кафедры информационных систем,
кандидат экономических наук, доцент
А.В. БАДЬИНА
(Государственный университет управления)

ВВЕДЕНИЕ

Методические указания к выполнению лабораторных работ по учебной дисциплине «Практикум на ЭВМ 4» предназначен для обучающихся по направлению подготовки бакалавров 01.03.02 Прикладная математика и информатика образовательная программа (ОП) Прикладная математика и информатика. Язык R [1] – это один из наиболее популярных инструментов визуализации, статистического анализа и моделирования. По результатам анализа статей, опубликованных в Google Scholar за 2018 год, R поднялся на второе место среди инструментов Data Science, уступив только SPSS[1].

Настоящее пособие предназначено для студентов, которые владеют синтаксисом языка R (с основами знакомит пособие [2]). Лабораторные работы по дисциплине «Практикум на ЭВМ 4» направлены на формирование навыков описательного анализа кросс-секционных данных и построения простых линейных моделей. При этом важной частью каждой работы является создание скрипта обновляемого отчёта на языке RMarkdown, который включает в себя и расчёты на языке R, и описательный текст. Приобретённые навыки полезны как в практике создания автоматизированных отчётов, так и в самостоятельных научных исследованиях. Рассматриваемые программные средства: свободно распространяемые R и RStudio, а также пакет «knitr» для создания отчётов внутри R, – широко применяются в различных сферах науки, от биологии до экономики, и зарекомендовали себя в качестве одного из стандартов обмена результатами обработки данных и моделирования.

Содержательно лабораторные работы по дисциплине «Практикум на ЭВМ 4» связаны с курсом «Эконометрика», читаемым в том же семестре. Однако отчёты по лабораторным работам сосредоточены на лаконичной подаче результатов и воспроизводимости. Предполагается, что усилия студентов при выполнении работ распределяются поровну между содержанием и оформлением (и публикацией) результатов. В качестве платформы для публикации используются:

- RPubs (https://rpubs.com) – сайт для размещения публичных отчётов и презентаций, созданных в RStudio;
- GitHub (https://github.com/) – портал, предоставляющий средства для совместной разработки и контроля версий на базе Git более чем 37 миллионам[2] пользователей по всему миру.

[1] Robert A. Data Science Software Used in Journals: Stat Packages Declining (including R), AI/ML Software Growing. URL: http://r4stats.com/2019/04/01/scholarly-datasci-popularity-2019/
[2] По данным Википедии. URL: https://en.wikipedia.org/wiki/GitHub

Примеры выполнены R версии 3.6.0, «Planting of a Tree», версия RStudio: 1.2.1335. Все ссылки действительны на 14 июня 2019 г. Репозиторий со скриптами из этого пособия: github.com/aksyuk/R-Practice-basics.

Лабораторные работы по дисциплине «Практикум на ЭВМ 4» формируют у студента следующие компетенции:

- **ОПК-3** – способность к разработке алгоритмических и программных решений в области системного и прикладного программирования, математических, информационных и имитационных моделей, созданию информационных ресурсов глобальных сетей, образовательного контента, прикладных баз данных, тестов и средств тестирования систем и средств на соответствие стандартам и исходным требованиям. Планируется, что по итогам успешного выполнения лабораторных работ студент будет ***знать*** принципы создания и оформления скриптов на языке R для вычислений, оформления отчетов и графического представления результатов исследования; ***уметь*** применять возможности языка R в области математического моделирования; ***владеть*** языком разметки Markdown для написания скриптов к отчетам по результатам вычислений; навыками решения практических задач в области теории вероятностей и математической статистики, эконометрики средствами R.
- **ПК-3** – способность критически переосмысливать накопленный опыт, изменять при необходимости вид и характер своей профессиональной деятельности. Планируется, что по итогам успешного выполнения лабораторных работ студент будет ***уметь*** формулировать проблемы и находить их решения на открытых форумов разработчиков программного обеспечения; ***владеть*** средствами коллективной работы (на примере github.com), средствами размещения отчетов (на примере rpubs.com).

Лабораторные работы охватывают следующие разделы программы дисциплины «Практикум на ЭВМ 4»:

1. **Раздел 4.** Предварительный анализ данных на примере кросс-секционной выборки. Графическое представление пространственных данных. Поиск аномальных наблюдений. Создание отчёта с помощью пакета «knitr».
2. **Раздел 5**. Построение регрессионных моделей в среде R.
3. **Раздел 6**. Тестирование остатков регрессионных моделей в среде R.

Условные обозначения

В этом пособии программный код выделен моноширинным шрифтом. Комментарии начинаются с символа решётки, а вывод консоли – с решётки и угловой скобки.

```
# описательные статистики по первым двум переменным
#  фрейма данных airquality
summary(airquality[, 1:2])
#>
#>      Ozone           Solar.R
#>  Min.   :  1.00   Min.   :  7.0
#>  1st Qu.: 18.00   1st Qu.:115.8
#>  Median : 31.50   Median :205.0
#>  Mean   : 42.13   Mean   :185.9
#>  3rd Qu.: 63.25   3rd Qu.:258.8
#>  Max.   :168.00   Max.   :334.0
#>  NA's   :37       NA's   :7
```

Неинформативные в примере строки скрипта будем заменять многоточием в угловых скобках (<…>). Скрипты ко всем главам и необходимые файлы данных находятся в свободном доступе по адресу: https://github.com/aksyuk/R-Practice-basics/tree/master/RScripts/manual_labs.

Код ниже демонстрирует версию R, в которой тестировались примеры этого руководства (ОС Windows 7).

```
R.version
#>
#> platform       x86_64-w64-mingw32
#> arch           x86_64
#> os             mingw32
#> system         x86_64, mingw32
#> status
#> major          3
#> minor          6.0
#> year           2019
#> month          04
#> day            26
#> svn rev        76424
#> language       R
#> version.string R version 3.6.0 (2019-04-26)
#> nickname       Planting of a Tree
```

1. СОЗДАНИЕ ОТЧЁТА С ПОМОЩЬЮ ПАКЕТА «knitr» В RSTUDIO

Для работы нам понадобятся:

- дистрибутив R, который можно скачать с официального сайта проекта. Версию под Windows можно скачать по ссылке: https://cran.r-project.org/bin/windows/base/ (бесплатен, распространяется по свободной лицензии);
- дистрибутив RStudio с официального сайта: https://www.rstudio.com/products/rstudio/download/#download (версия Desktop бесплатна).

Далее под Windows достаточно запустить скачанные файлы и следовать диалогу установки, оставив по умолчанию все настройки, кроме пути к папкам, в которые устанавливаются R и RStudio. Пути не должны быть очень длинными и не должны содержать символов кириллицы[3].

В настройках RStudio (меню «Tools» → «Global Options»…) есть раздел, который относится к скриптам отчётов: «Sweave» (рисунок 1). Исторически сложилось, что пакет «sweave» для интеграции кода R в LaTeX и LyX документы появился первым. Однако после появления альтернативного пакета «knitr» большинство пользователей переключились на него, в том числе благодаря возможностям публиковать отчёты в файлы Ms Word и html-страницы. Убедимся, что в первом пункте «Weave Rnw files using:» выбран «knitr». Вторым пунктом стоит выбор дистрибутива LaTeX, который необходим для создания отчётов в виде pdf-файлов. В Windows его нужно устанавливать отдельно.

1.1 Встроенный пример отчёта «knitr»

Установим и включим пакет «knitr»:

```
install.packages('knitr')          # установка
library('knitr')                   # загрузка функций в память
```

Исходный код отчёта необходимо записать в файл RMarkdown с расширением `.Rmd`. В меню RStudio выберем пункт «File» → New File → RMarkdown… В диалоговом окне введём заголовок отчёта (поле «Title»), имя автора («Author») и выберем формат «Word» (рисунок 2).

[3] Подробная инструкция по установке R и RStudio под Windows, macos и linux доступна на гитхабе Б.Б. Демешева: https://bdemeshev.github.io/installation/r/R_installation.html

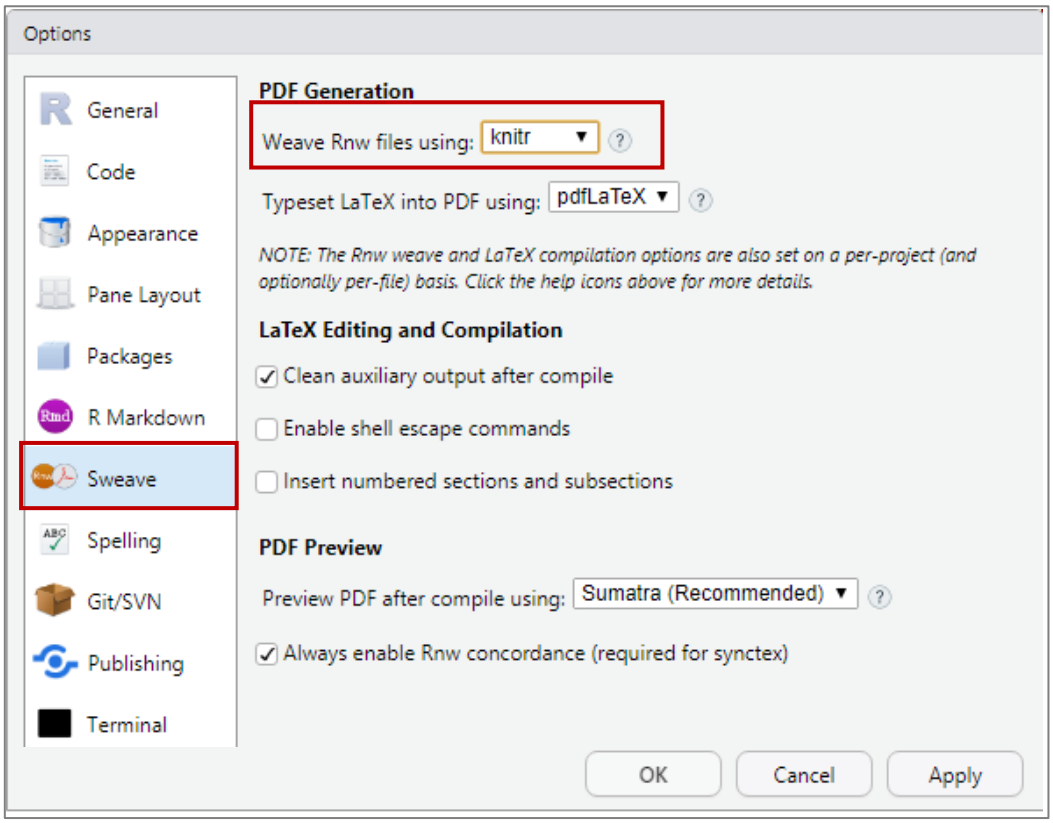

Рис. 1. Выбор пакета для связывания отчётов в RStudio

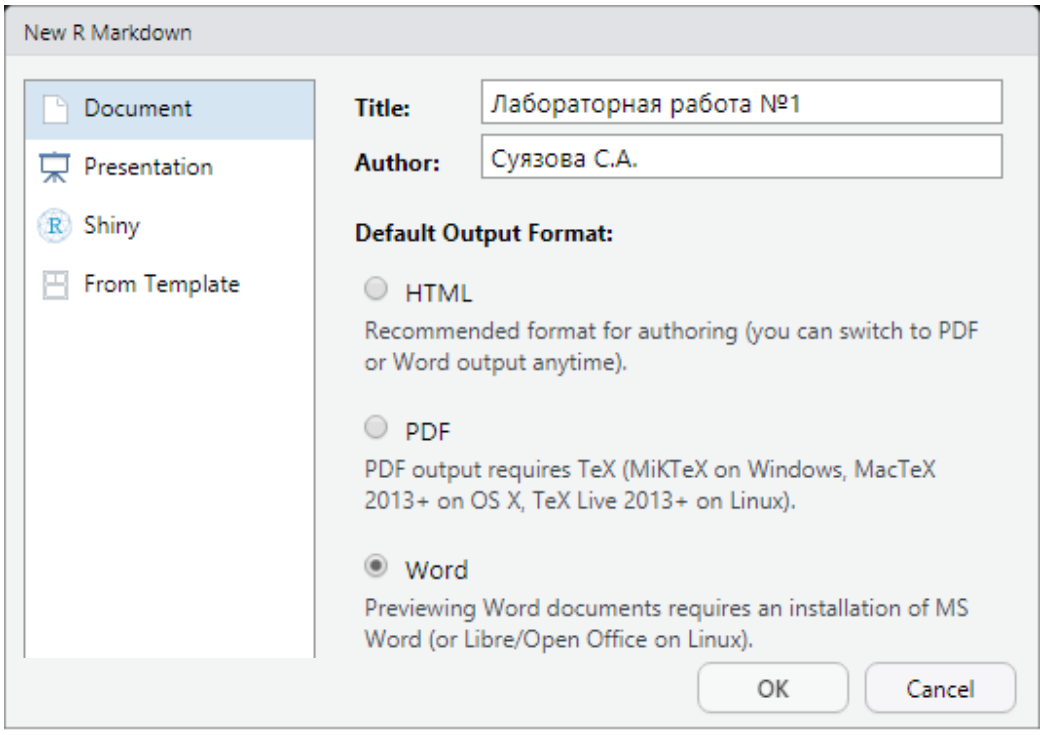

Рис. 2. Диалог создания скрипта отчёта (файла .Rmd)

Новый файл уже содержит готовый пример отчёта (рисунок 3). Основной текст записывается в разметке Markdown. Блоки кода на языке R приводятся в окружении тройных обратных кавычек (буква «ё» в английской раскладке).

Чтобы получить файл «.doc» с отчётом, нужно нажать на кнопку «Knit» над окном редактора кода. При этом файл требуется предварительно сохранить; если этого не было сделано, появится диалоговое окно сохранения. Дадим отчёту имя «test.Rmd» и сохраним в рабочей директории (расширение нужно указывать явно).

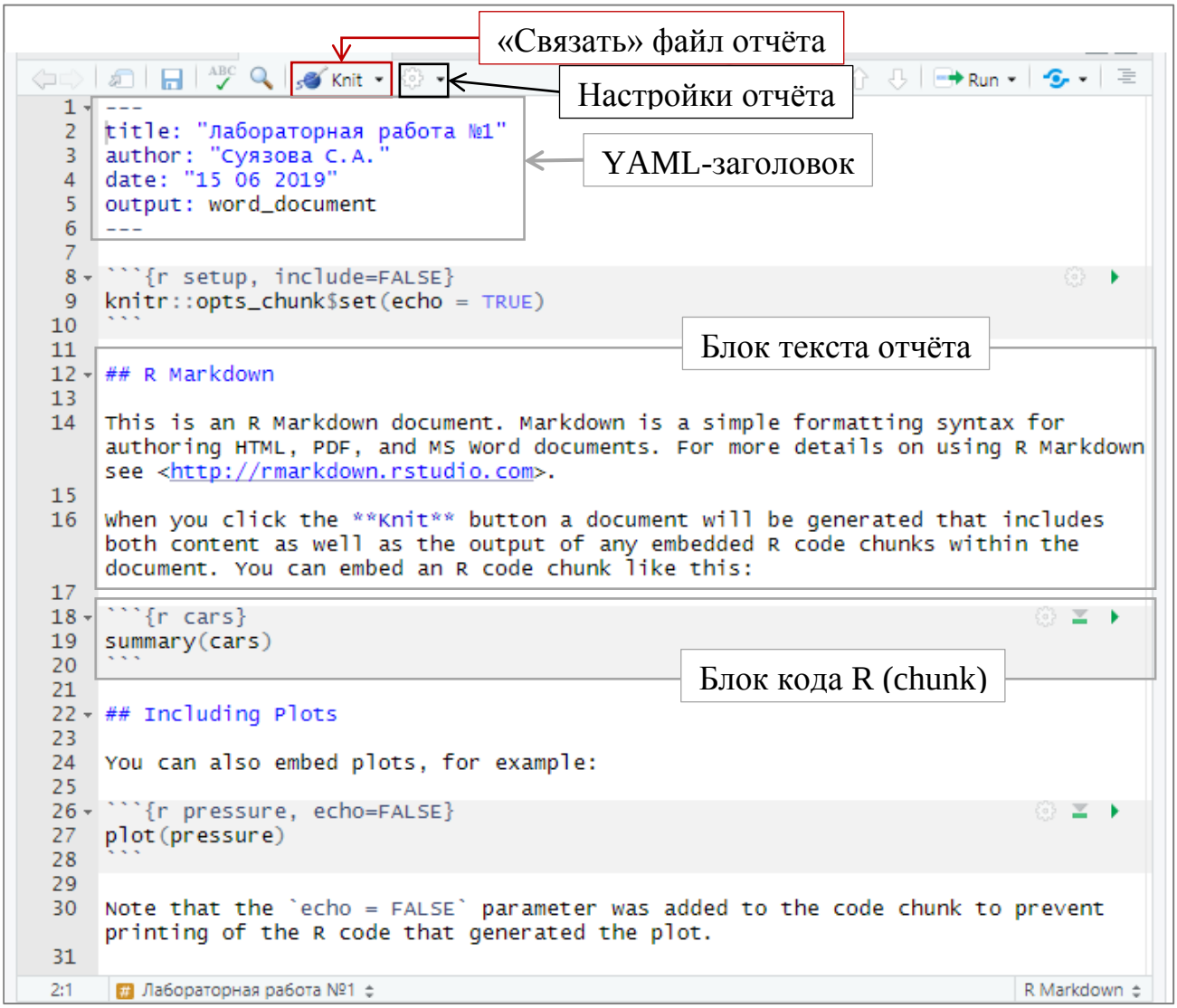

Рис. 3. Скрипт отчёта по умолчанию

Не всегда в готовый отчёт требуется включать код на R. Вывод кода можно отключить с помощью параметров блоков кода, перечисленных в таблице 1. Параметры добавляют после запятой в объявление блока. Например, третий блок примера отчёта называется «pressure» и строит график, не показывая код:

```
```{r pressure, echo = FALSE}
plot(pressure)
```
```

Таблица 1 – Параметры блоков кода [2]

Опция	Выполнение кода	Отображение кода	Результаты	Графики	Сообщения	Предупреждения
`eval = FALSE`	X	-	X	X	X	X
`include = FALSE`	-	X	X	X	X	X
`echo = FALSE`	-	X	-	-	-	-
`results = "hide"`	-	-	X	-	-	-
`fig.show = "hide"`	-	-	-	X	-	-
`message = FALSE`	-	-	-	-	X	-
`warning = FALSE`	-	-	-	-	-	X

Перечисленные в таблице параметры можно выставить по умолчанию для всех блоков функцией `opts_current$set()` в самом первом блоке кода. Этот блок удобно назвать «setup», тогда он будет выполняться при каждом запуске любого блока кода. Так, блок «setup» скрипта, который сгенерировал данное руководство, включает код:

````
```{r setup, include = FALSE}
knitr::opts_chunk$set(
     echo = TRUE,
     comment = "#>",
     collapse = TRUE)
```
````

Опция `comment = "#>"` устанавливает символ начала строки для вывода консоли, а опция `collapse = TRUE` объединяет весь вывод одного блока кода в единый абзац.

1.2 Коротко о языке Markdown

Markdown – облегчённый язык разметки, созданный с целью написания максимально читабельного и удобного для правки текста, но пригодного для преобразования в языки для продвинутых публикаций[4]. Его синтаксис описан в [4, 5] (последняя ссылка – англоязычный первоисточник от одного из создателей языка).

Простые элементы разметки Markdown:

1. 1. После двойного пробела текст переносится на следующую строку.

[4] Markdown / Википедия – свободная энциклопедия. URL: ru.wikipedia.org/wiki/Markdown

2. 1. Один из способов выделить заголовок – поставить символы решётки в начале строки. Количество решёток соответствует уровню заголовка.
3. 1. Нумерованные списки обозначаются цифрой и точкой, как в этом списке, только цифрам необязательно идти по порядку. При интерпретации файла список будет пронумерован автоматически.
4. 1. Позиции маркированных списков начинаются со звёздочки.
   - * Для выделения жирным шрифтом текст заключают в двойные звёздочки: ****жирный текст****.
   - * Для выделения курсивом – в одинарные: **курсив**.
   - * Тройные косые кавычки (буква ё в английской раскладке) обрамляют текст, написанный буквами одной ширины, как в программном коде: `` ```моноширинный шрифт``` ``.
5. 1. Наконец, один из способов написать гиперссылку такой: `[краткий справочник по Markdown] (http://rukeba.com/by-the-way/markdown-sintaksis-po-russki/)`. Что даст следующий результат: [краткий справочник по Markdown](http://rukeba.com/by-the-way/markdown-sintaksis-po-russki/).

В RStudio скрипты на R также можно снабжать заголовками на Markdown:

```
# Заголовок первого уровня ------------------------------

# Заголовок второго уровня ==============================

# Заголовок третьего уровня ##############################
```

После чего можно легко перемещаться по разделам кода, используя навигатор в нижней части окна редактора скриптов (рисунок 4).

Каждому блоку можно дать уникальное имя, указав его в заголовке: `` `{r <имя блока>}` ``. Тогда в окне редактора кода RStudio будет удобно переключаться между ними так же, как показано на рисунке 4. Имена блоков можно писать и латиницей, и кириллицей, но без пробелов и спецсимволов.

Код R можно включать прямо в строку текста, заключая его в обратные кавычки. Например, выражение `` `r x` `` в тексте при генерации отчёта будет заменено на значение переменной `x`.

Добавим, что при написании кода на R в отчёте следует придерживаться рекомендаций по стилю программирования на R от Google [7].

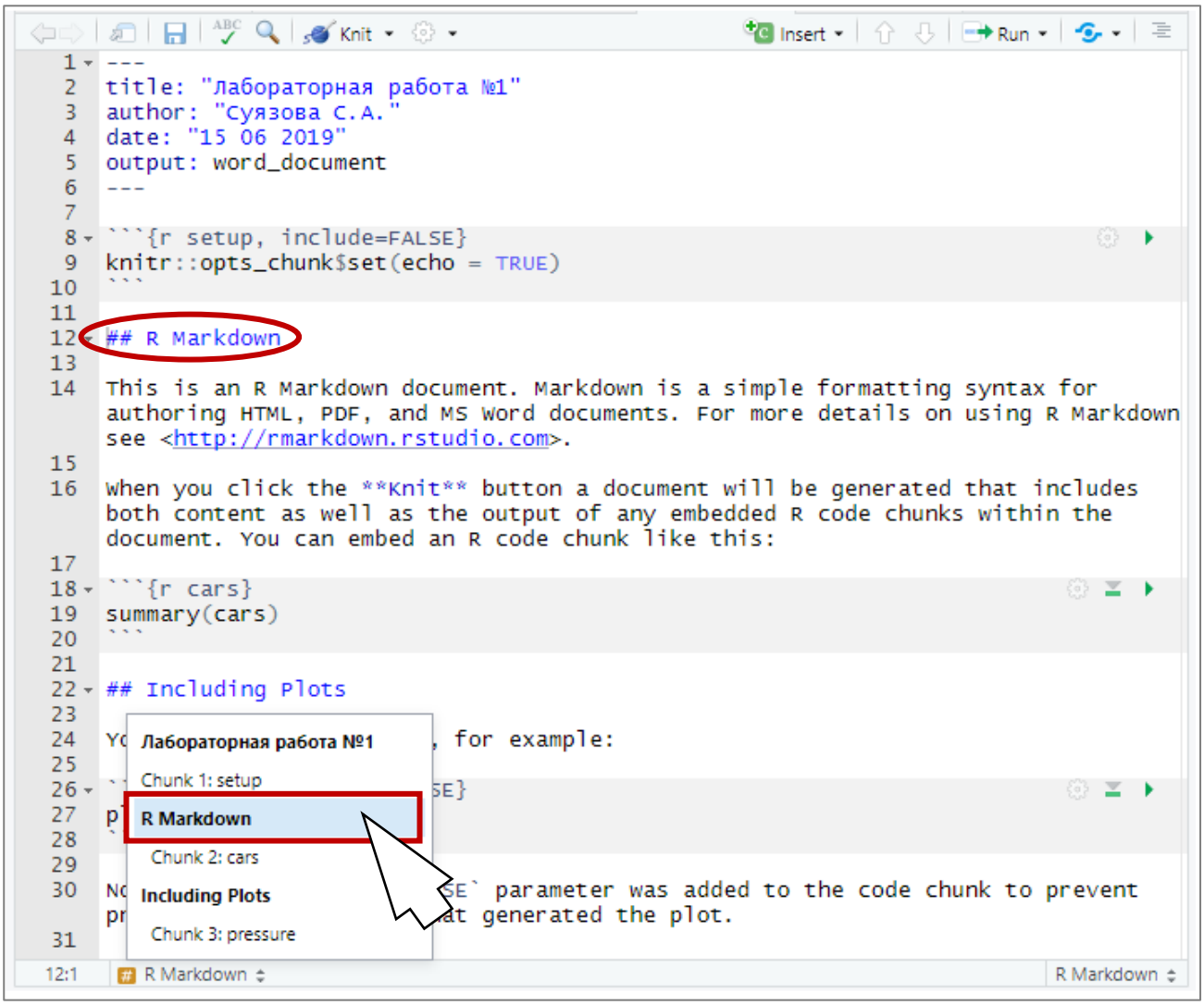

Рис. 4. Навигация по заголовкам в скрипте отчёта

Стоит упомянуть, что при генерации отчёта все объекты создаются внутри его собственного пространства имён. Поэтому после того, как текст отчёта «связан», объекты и графики не сохраняются в глобальном пространстве имён. Поэтому на этапе отладки бывает полезно запускать весь код отчёта либо его части, не создавая сам файл отчёта. Для этого в «RStudio» удобно пользоваться разделом меню «Code» → «Run Region»:

- **Run Setup Chunk** – запустить код блока «setup» с общими настройками;
- **Run All Chunks Above** – запустить все блоки выше курсора;
- **Run All Chunks Below** – запустить все блоки ниже курсора;
- **Run Current Chunk** – запустить блок, в котором стоит курсор;
- **Run Next Chunk** – запустить следующий блок после курсора;
- **Run Function Definition** – запустить код определения функции;
- **Run All** – запустить весь код в отчёте.

2. УКАЗАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ №1: ПРЕДВАРИТЕЛЬНЫЙ АНАЛИЗ ДАННЫХ

Предварительный анализ предшествует построению эконометрических моделей и преследует несколько задач, среди которых:

- Построение гистограмм разброса и расчёт описательных статистик исходных показателей для сопоставления с нормальных законом и выявления аномальных наблюдений.
- Графическое представление исходных данных для поиска взаимосвязей между показателями.
- Корреляционный анализ, проверка гипотез о значимости выявленных взаимосвязей.

В лабораторной работе рассматривается несколько простых инструментов предварительного анализа данных в R. Используются данные:

- статистика продаж ликёроводочной продукции и несколько показателей социально-экономического развития регионов за 2011 год из файла «Пример_алкоголь-2011.csv» (файл находится в репозитории github.com/aksyuk/R-data/).

Используются пакеты:

```
library('Hmisc')          # для расчёта корреляционной матрицы
library('corrplot')       # визуализация корреляционных матриц
library('knitr')          # для генерации отчёта
library('nortest')        # тест на нормальность: ad.test()
```

2.1 Импорт данных

Исходные данные для работы хранятся в файле «Пример_алкоголь-2011.csv». Это розничные продажи ликёроводочной продукции и ещё три показателя, взаимосвязь которых с продажами необходимо проверить, по регионам РФ за 2011 год. Убедимся, что файл лежит в рабочей директории. Удобно разместить файл с данными в той же директории, в которой сохранён скрипт. Установить в качестве рабочей директории расположение скрипта, открытого в RStudio, можно в меню «Session» → «Set Working Directory» → «To Source File Location».

```
# проверка содержимого рабочей директории
dir()
<…>
#> [9] "Пример_алкоголь.RData"
```

На рисунке 5 приводится выдержка из файла данных.

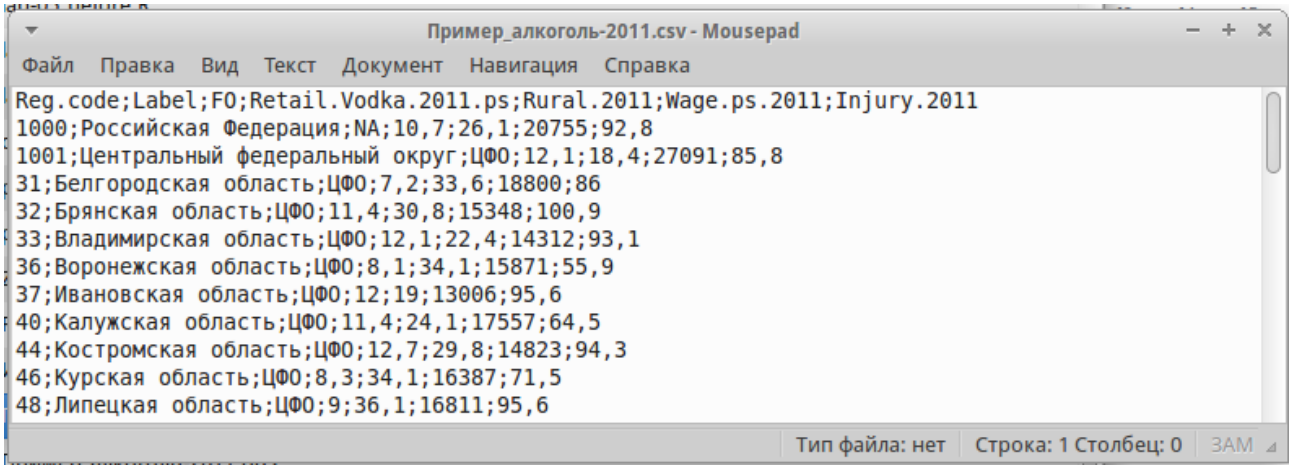

Рис. 5. Содержимое файла исходных данных

Импортируем таблицу во фрейм:

```
# импорт данных из .csv
DF <- read.table('Пример_алкоголь-2011.csv',
                 header = T,        # заголовок в первой строке
                 dec = ',',         # разделитель целой и дробной части
                 sep = ';',         # разделитель значений в строке
                 row.names = 1,     # первый столбец – названия наблюдений
                 as.is = T,              # не делать факторы
                 na.strings = 'NA')      # символы пропущенных значений
```

Посмотрим на результат:

```
dim(DF)
#> [1] 92  9

# структура фрейма
str(DF)
#> 'data.frame':    92 obs. of  9 variables:
#>  $ Label                : chr  "Российская Федерация" ...
#>  $ FO                   : chr  NA "ЦФО" "ЦФО" "ЦФО" ...
#>  $ Retail.Vodka.2011.ps: num  10.7 12.1 7.2 11.4 12.1 8.1...
#>  $ Rural.2011           : num  26.1 18.4 33.6 30.8 22.4 34.1 ...
#>  $ Wage.ps.2011         : int  20755 27091 18800 15348 14312 ...
#>  $ Injury.2011          : num  92.8 85.8 86 100.9 93.1 ...
```

Метки наблюдений – номера регионов согласно тексту Конституции РФ[5]. Первый столбец называется «Label» и содержит наименования регионов. Второй столбец («FO») – принадлежность региона к федеральному округу.

Столбцы с третьего по шестой содержат количественные показатели:

- **Retail.Vodka.2011.ps** – розничная продажа ликёроводочных изделий за год, литров на человека[6].
- **Rural.2011** – удельный вес сельского населения, в процентах, оценка на конец года[7].
- **Wage.ps.2011** – среднедушевые денежные доходы населения в месяц, рублей[8].
- **Injury.2011** – заболеваемость на 1000 человек населения: травмы, отравления и некоторые другие последствия воздействия внешних причин[9].

Сделаем из столбца «FO» фактор (порядок округов неважен).

```
# делаем из столбца "FO" фактор
DF$FO <- as.factor(DF$FO)
str(DF)
#> 'data.frame':    92 obs. of  9 variables:
#> <…>
#>  $ FO     : Factor w/ 8 levels "ДВФО","ПФО",..: NA 7 7 7 7 ...
```

В таблице 97 строк и 6 столбцов, нет необходимости экономить оперативную память. Оставим исходную таблицу без изменений, создав фрейм «`reg.df`», в который войдут только нужные нам строки и столбцы.

```
# оставляем только регионы и выбрасываем столбец меток,
#  чтобы удобнее было считать
reg.df <- DF[as.numeric(row.names(DF)) < 1000, -1]
# выбрасываем пропущенные
reg.df <- na.omit(reg.df)
```

[5] Коды субъектов Российской Федерации / Википедия – свободная энциклопедия. URL: https://ru.wikipedia.org/wiki/ Коды_субъектов_Российской_Федерации/

[6] Статистический сборник «Основные показатели, характеризующие рынок алкогольной продукции в 2011 году» / сайт Федеральной службы по регулированию алкогольного рынка. URL: http://fsrar.ru/industry/ 1261678438828/statisticheskii-sbornik-osnovnye-pokazateli--harak

[7] «Регионы России. Социально-экономические показатели – 2012г.» / сайт Росстата. URL: http://www.gks.ru/wps/wcm/connect/rosstat_main/rosstat/ru/statistics/publications/catalog/doc_1138623506156

[8] Там же

[9] Там же

```
# первые пять строк фрейма
head(reg.df)
#>      FO Retail.Vodka.2011.ps Rural.2011 Wage.ps.2011 Injury.2011
#> 31 ЦФО                  7.2       33.6        18800        86.0
#> 32 ЦФО                 11.4       30.8        15348       100.9
#> 33 ЦФО                 12.1       22.4        14312        93.1
#> 36 ЦФО                  8.1       34.1        15871        55.9
#> 37 ЦФО                 12.0       19.0        13006        95.6
#> 40 ЦФО                 11.4       24.1        17557        64.5

# последние пять строк фрейма
tail(reg.df)
#>      FO Retail.Vodka.2011.ps Rural.2011 Wage.ps.2011 Injury.2011
#> 27 ДВФО                 16.7       18.5        23766       100.2
#> 28 ДВФО                 12.3       33.0        17790        93.5
#> 49 ДВФО                 17.3        4.3        30452       112.2
#> 65 ДВФО                 20.0       19.7        32268        65.0
#> 79 ДВФО                 12.3       31.9        16525        94.3
#> 87 ДВФО                  3.7       34.0        43049       124.1
```

2.2 Расчёт описательных статистик

Пример №1. Встроенная функция `summary()` для аргумента-фрейма возвращает таблицу описательных статистик.

```
# встроенная функция расчёта описательных статистик
summary(reg.df, digits = 2)
#>        FO     Retail.Vodka.2011.ps   Rural.2011    Wage.ps.2011
#>  ЦФО    :17   Min.   : 0.0         Min.   : 4.3   Min.   : 8829
#>  ПФО    :14   1st Qu.: 8.3         1st Qu.:22.6   1st Qu.:14322
#>  СФО    :12   Median :11.2         Median :29.8   Median :15990
#>  ДВФО   : 9   Mean   :10.5         Mean   :31.5   Mean   :17488
#>  СЗФО   : 9   3rd Qu.:13.1         3rd Qu.:36.2   3rd Qu.:18712
#>  СКФО   : 7   Max.   :20.0         Max.   :71.3   Max.   :43049
#>  (Other):10
#>   Injury.2011
#>  Min.   : 37
#>  1st Qu.: 75
#>  Median : 93
#>  Mean   : 90
#>  3rd Qu.:106
#>  Max.   :128
```

Такой набор даёт представление о поведении показателей, но не всегда нужен в тексте отчёта, где для подкрепления выводов достаточно нескольких

основных статистик. Рассчитаем только среднее арифметическое, стандартное отклонение и коэффициент вариации для четырёх количественных показателей. Используем функции:

- **mean(x)** – для расчёта среднего арифметического вектора *x*;
- **sd(x)** – для расчёта стандартного отклонения вектора *x*.

Эти функции применяются к числовым векторам, и для короткого расчёта статистик по фрейму, который является списком векторов, нам понадобится:

- **apply(*df*, *dim*, *fun*)** – функция, которая применяет к измерению *dim* фрейма *df* функцию *fun*. В нашем случае нужно посчитать описательные статистики по столбцам, поэтому аргумент *dim* равен 2 (для строк этот аргумент равен 1). При этом мы выбросим первый столбец фрейма, в котором записаны коды федеральных округов (`reg.df[, -1]`).

```
#  средние арифметические
mns <- round(apply(reg.df[, 2:5], 2, mean), 1)
mns
#> Retail.Vodka.2011.ps           Rural.2011         Wage.ps.2011
#>                 10.5                 31.5              17488.4
#>          Injury.2011
#>                 89.6

#  стандартные отклонения
sds <- round(apply(reg.df[, 2:5], 2, sd), 1)
sds
#> Retail.Vodka.2011.ps           Rural.2011         Wage.ps.2011
#>                  4.2                 12.4               5489.4
#>          Injury.2011
#>                 21.2

#  коэффициенты вариации
coef.vars <- round(sds/mns*100, 1)
coef.vars
#> Retail.Vodka.2011.ps           Rural.2011         Wage.ps.2011
#>                 40.0                 39.4                 31.4
#>          Injury.2011
#>                 23.7

# делаем свою таблицу только с нужными статистиками
#  по количественным показателям: среднее, СКО, коэфф. вариации
smm <- rbind(mns, sds, coef.vars)     # соединить как строки
```

```
# названия статистик -- заголовки строк
row.names(smm) <- c('Среднее', 'Стандартное отклонение',
                    'Коэффициент вариации, %')
smm
#>                     Retail.Vodka.2011.ps Rural.2011 Wage.ps.2011
#> Среднее                             10.5       31.5      17488.4
#> Стандартное отклонение               4.2       12.4       5489.4
#> Коэффициент вариации, %             40.0       39.4         31.4
#>                           Injury.2011
#> Среднее                          89.6
#> Стандартное отклонение           21.2
#> Коэффициент вариации, %          23.7
```

В итоге из полученной таблицы видно, что данные относительно однородны (коэффициенты вариации в пределах или ненамного превышают 33%).

2.3 Анализ распределения показателей

Пример №2. Посмотрим, насколько нормально распределение исходных показателей. Для этого построим гистограммы плотности распределения и проведём статистический тест Шапиро-Уилка на нормальность[10].

```
# строим гистограммы на одном полотне
par(mfrow = c(2, 2))            # разбить полотно на 4 части, 2x2
par(oma = c(0, 0, 1.5, 0))      # внешние поля общего полотна
par(mar = c(4, 4, 0.5, 0.5))    # внутренние поля каждого графика

# цикл по номерам столбцов с количественными переменными
for (i in 2:5) {
    # данные -- i-ый столбец фрейма
    x <- reg.df[, i]
    # гистограмма
    hist(x,
         freq = F,              # по вертикали - плотность (доля)
         col = 'wheat',         # цвет заливки
         xlab = colnames(reg.df)[i],    # ось X - название столбца
         ylab = 'Плотность',            # ось Y
         main = '')                     # без заголовка
    # теоретическая плотность
    curve(dnorm(x, mean = mean(x), sd = sd(x)), col = 'darkblue',
          lwd = 2, add = TRUE)}
```

[10] Критерий Шапиро-Уилка / Портал MachineLearning.ru. URL: http://www.machinelearning.ru/wiki/index.php?title=Критерий_Шапиро-Уилка

```
# общий заголовок для всех графиков
title(main = 'Гистограммы распределения показателей',
      outer = TRUE, cex = 1.5)
par(mfrow = c(1, 1))                # вернуть настройки обратно, 1x1
```

Результат показан на рисунке 6. Судя по гистограммам, распределения всех показателей несколько скошены (асимметричны) по сравнению с нормальным законом. Особенно это проявляется у «Rural.2011» и «Wage.ps.2011». Кроме того, показатели имеют более островершинные распределения по сравнению с нормальным законом.

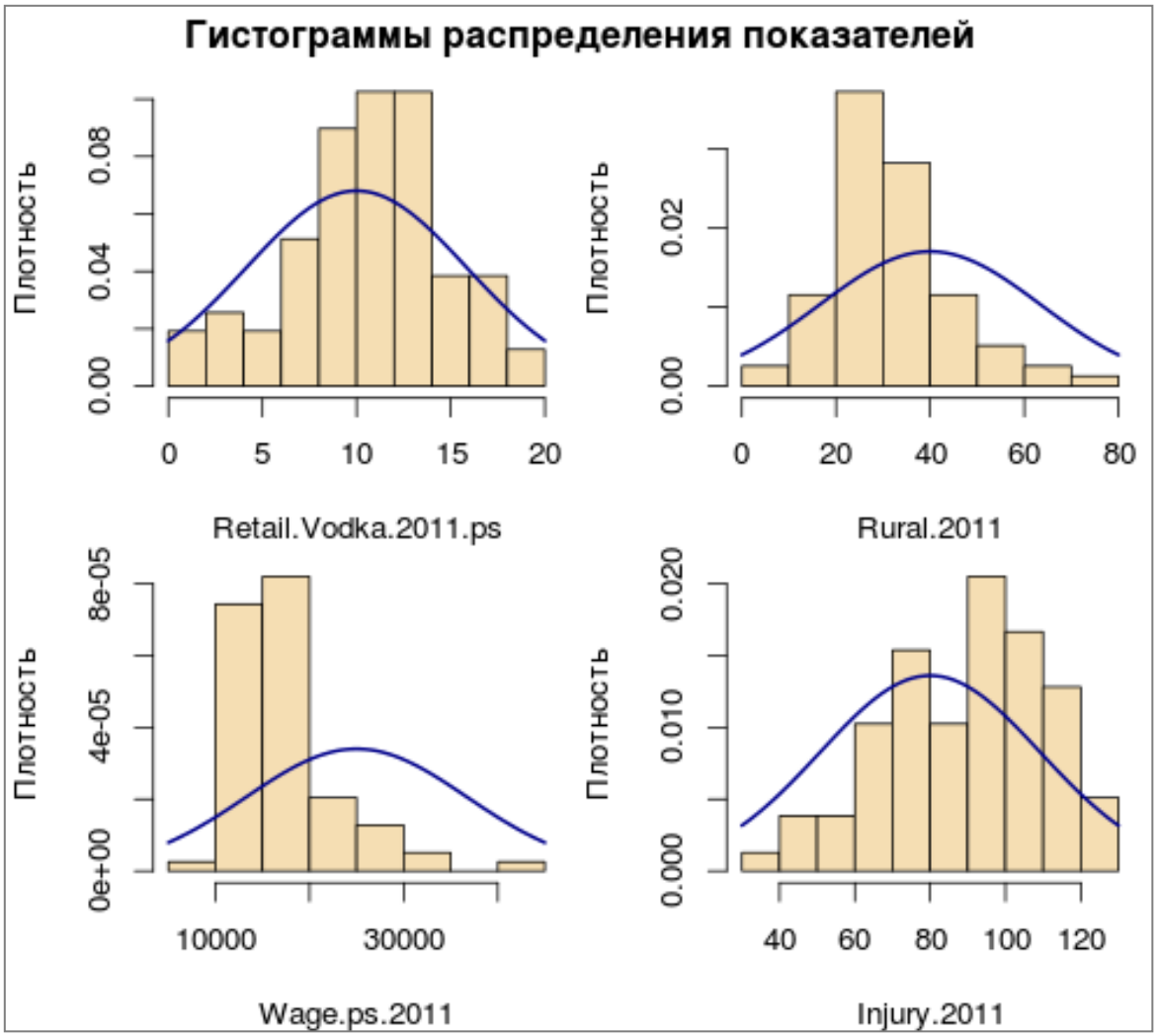

Рис. 6. Гистограммы распределения переменных примера

Протестируем показатели на нормальность. Нулевая гипотеза теста Шапиро-Уилка: распределение соответствует нормальному закону. Альтернативная: распределение ненормально. Проверим гипотезу для переменной `Retail.Vodka.2011.ps` на уровне значимости 0,05.

```
# тест Шапиро-Уилка на нормальность Retail.Vodka.2011.ps
shapiro.test(reg.df$Retail.Vodka.2011.ps)
#>  Shapiro-Wilk normality test
#>
#> data:  reg.df$Retail.Vodka.2011.ps
#> W = 0.98686, p-value = 0.616
```

P-значения больше чем 0,05, следовательно, нулевая гипотеза не отвергается: распределение нормально. Вызывать функцию от каждого столбца фрейма по отдельности неудобно, и в итоге из всего вывода нам нужно только одно число. Оптимизируем процедуру с помощью функции `apply()`:

```
# применяем ко всем столбцам с помощью apply()
apply(reg.df[, 2:5], 2, shapiro.test)
#> $Retail.Vodka.2011.ps
#>
#>  Shapiro-Wilk normality test
#>
#> data:  newX[, i]
#> W = 0.98686, p-value = 0.616
#> <…>
```

Результат выводится в консоль в виде списка, который не подходит для представления в текстовом отчёте. Стоит пойти дальше и извлечь из объекта результатом каждого теста нужную статистику, чтобы затем представить всё в одной таблице. Для начала необходимо понять структуру объекта с результатом:

```
# структура объекта
str(shapiro.test(reg.df$Retail.Vodka.2011.ps))
#> List of 4
#>  $ statistic: Named num 0.987
#>   ..- attr(*, "names")= chr "W"
#>  $ p.value  : num 0.616
#>  $ method   : chr "Shapiro-Wilk normality test"
#>  $ data.name: chr "reg.df$Retail.Vodka.2011.ps"
#>  - attr(*, "class")= chr "htest"
```

Это список из четырёх элементов, среди которых: «statistic» – расчётное значение критерия, «p.value» – соответствующее p-значение. Изменим вызов функции `apply()`, прописав пользовательскую функцию, которая извлекает из отчёта по тесту значение статистики и округляет до двух знаков после запятой.

```
# применяем ко всем столбцам и вытаскиваем только тестовую
статистику
apply(reg.df[, 2:5], 2, function (x) {
    round(shapiro.test(x)$statistic, 2)
})
#> Retail.Vodka.2011.ps           Rural.2011         Wage.ps.2011
#>                 0.98                 0.95                 0.82
#>          Injury.2011
#>                 0.97
```

Ещё один тест на нормальность, тест Андерсона-Дарлинга, реализован в пакете «nortest» и работает аналогичным образом (гипотезы аналогичны):

```
# Тест Андерсона-Дарлинга на нормальность распределения
# пример для одного столбца
ad.test(reg.df$Retail.Vodka.2011.ps)
#>
#>  Anderson-Darling normality test
#>
#> data:  reg.df$Retail.Vodka.2011.ps
#> A = 0.42136, p-value = 0.3155
```

2.4 Анализ взаимосвязей показателей

Пример №3. Для выявления взаимосвязей между парами показателей удобно использовать графики взаимного разброса. Построим их с помощью функции `pairs()` базовой графической системы R (рисунок 7).

```
# графики взаимного разброса
pairs(reg.df[, -1],      # фрейм без первого столбца-фактора
      pch = 21,          # тип символов для точек
      col = rgb(0, 0, 1, alpha = 0.4),    # цвет заливки точек
      bg = rgb(0, 0, 1, alpha = 0.4),     # цвет границы точек
      cex = 1.1)                  # масштаб символов для точек
```

Необходимо в первую очередь сделать вывод о взаимосвязи «Retail.Vodka.2011.ps» с остальными показателями, поэтому сначала анализируют первую строку матрицы. На графиках в этой строке по вертикали отложен показатель «Retail.Vodka.2011.ps».

Судя по графикам, между показателем «Retail.Vodka.2011.ps» и «Rural.2011» есть слабая обратная линейная связь (точки разбросаны вдоль воображаемой прямой с отрицательным наклоном). Если исключить несколько отдельно стоящих точек, вероятно, будет слабая прямая связь между

«Retail.Vodka.2011.ps» и «Injury.2011». На втором графике разброса в первой строке центр тяжести точек лежит вдоль практически вертикальной прямой, что может указывать на отсутствие связи между «Retail.Vodka.2011.ps» и «Wage.ps.2011».

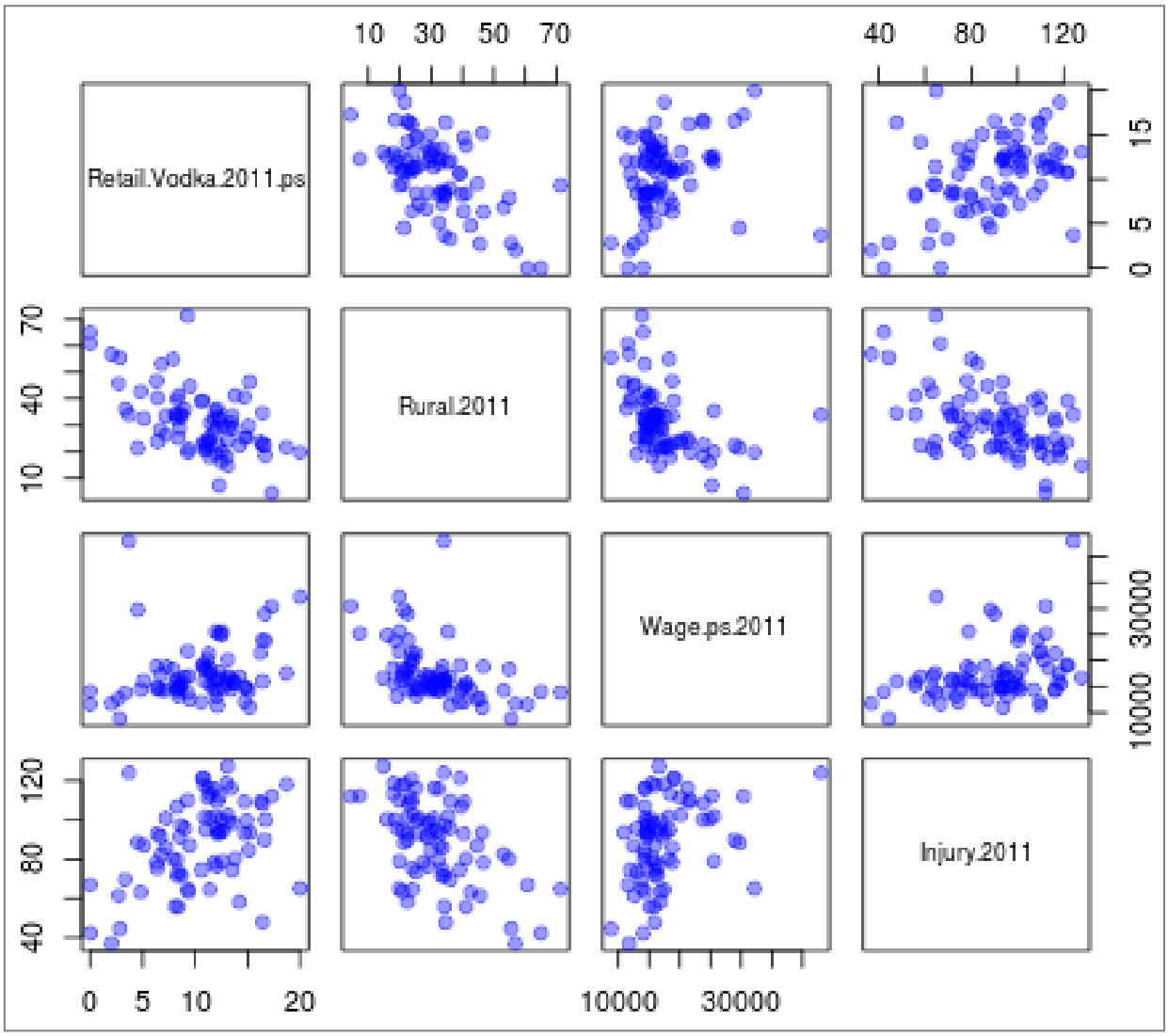

Рис. 7. Графики взаимного разброса показателей

Чтобы подтвердить или опровергнуть эти предположения, рассчитаем корреляционную матрицу и проверим на значимость коэффициенты корреляции (см. код на следующей странице).

Судя по P-значениям, все коэффициенты корреляции значимы с вероятностью 0,95 (меньше уровня значимости 0,05). Между всеми парами показателей обнаружены линейные взаимосвязи, но они не являются сильными: ни один из коэффициентов корреляции по модулю не больше 0,6[11].

[11] Величина и сила коэффициента корреляции. URL: https://statpsy.ru/correlation/velicina/

Анализ корреляционной матрицы можно облегчить, представив её графически. Для этого воспользуемся функцией `corrplot()` пакета «corrplot». Результат показан на рисунке 8.

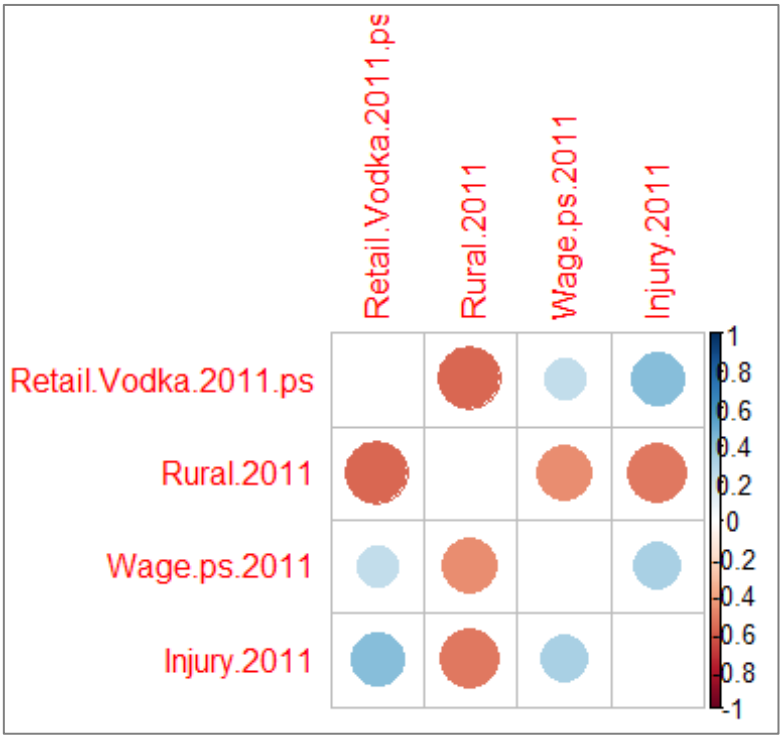

Рис. 8. Визуализация корреляционной матрицы

Аргументы функции `corrplot()` позволяют делать довольно сложные визуализации. Много примеров содержится в справке по этой функции, в разделе примеров («Examples»).

2.5 Сохранение рабочего пространства

Сохраним объекты, которые понадобятся в следующей практике, в файл «Пример_алкоголь.RData» в рабочей директории.

```
# убираем промежуточные объекты
rm(i, smm, coef.vars, sds, mns, matrix.cor, matrix.p, x)

# сохраняем рабочее пространство в файл
save.image('Пример_алкоголь.RData')
```

```
# коэффициенты Пирсона с P-значениями
rcorr(as.matrix(reg.df[, -1]))
#>
#>                      Retail.Vodka.2011.ps Rural.2011 Wage.ps.2011 Injury.2011
#> Retail.Vodka.2011.ps                 1.00      -0.58         0.25        0.42
#> Rural.2011                          -0.58       1.00        -0.46       -0.52
#> Wage.ps.2011                         0.25      -0.46         1.00        0.33
#> Injury.2011                          0.42      -0.52         0.33        1.00
#>
#>n= 78
#>
#>
#>P
#>                     Retail.Vodka.2011.ps Rural.2011 Wage.ps.2011 Injury.2011
#>Retail.Vodka.2011.ps                      0.0000     0.0295       0.0001
#>Rural.2011          0.0000                           0.0000       0.0000
#>Wage.ps.2011        0.0295                0.0000                  0.0036
#>Injury.2011         0.0001                0.0000     0.0036

# сохраняем корреляционную матрицу
matrix.cor <- cor(reg.df[, -1])
# сохраняем p-значения
matrix.p <- rcorr(as.matrix(reg.df[, -1]))$P

# изображаем матрицу графически
corrplot(matrix.cor,          # сама корреляционная матрица
         order = 'original',  # порядок отображения показателей
                              # в матрице
         diag = F,            # не отображать значения на главной
                              # диагонали
         p.mat = matrix.p,    # p-значения
         insig = 'blank',     # метод отображения незначимых
         sig.level = 0.05)    # уровень значимости
```

3. УКАЗАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ №2: РЕГРЕССИОННЫЕ МОДЕЛИ

Во второй лабораторной мы продолжим работу с примером данных по реализации ликёро-водочных изделий в РФ в натуральном выражении за 2011 год и оценим параметры моделей, объясняющих поведение данного показателя несколькими факторами. Зависимая переменная модели:

- **Retail.Vodka.2011.ps** – розничная продажа ликёроводочных изделий за год, литров на человека.

Независимые переменные модели:

- **Rural.2011** – удельный вес сельского населения, в процентах, оценка на конец года.
- **Wage.ps.2011** – среднедушевые денежные доходы населения в месяц, рублей.
- **Injury.2011** – заболеваемость на 1000 человек населения: травмы, отравления и некоторые другие последствия воздействия внешних причин.

Факторы были подобраны, чтобы проверить следующие гипотезы:

1. Подтверждают ли показатели Retail.Vodka.2011.ps и Rural.2011 тезис о том, что в сельской местности пьют больше, чем в городской?
2. Низкий уровень доходов может быть как фактором, так и следствием систематического потребления алкоголя. Зависит ли объём реализованной ликёроводочной продукции от среднедушевых ежемесячных доходов населения по региону?
3. Связано ли потребление водки с травмами и отравлениями? Пара показателей Retail.Vodka.2011.ps как зависимая переменная и Injury.2011 как независимая на самом деле служит примером перевёрнутой причинно-следственной связи. Однако если значимая взаимосвязь существует, она будет работать в обе стороны, и переменная Injury.2011 может оказаться фактором, который увеличит процент объяснённой дисперсии модели.

В лабораторной работе рассматривается построение линейных регрессионных моделей в R. Используются данные:

- файл рабочего пространства R «Пример_алкоголь-2011.RData».

Используются пакеты:

```
library('Hmisc')          # для расчёта корреляционной матрицы
```

3.1 Импорт данных

Исходные данные для работы были загружены в R и сохранены в файле рабочего пространства «Пример_алкоголь-2011.RData». Убедимся, что файл лежит в рабочей директории, и загрузим его.

```
dir()                  # список файлов рабочей директории
#> [1] "Пример_алкоголь.RData"
# загрузка объектов из сохранённого рабочего пространства
load('Пример_алкоголь.RData')

# просмотр списка объектов
ls()
#> [1] "DF"     "reg.df"
```

Фрейм «DF» – это статистика по показателям из сборников «Регионы России» по всем субъектам России, включая федеральные округа и РФ в целом, за 2011 год. Фрейм «reg.df» содержит данные, подготовленные для анализа: только регионы без пропущенных наблюдений.

```
# размерность фрейма с данными по регионам
dim(reg.df)

#> [1] 78  5

# первые строки фрейма данных
head(reg.df)

#>      FO Retail.Vodka.2011.ps Rural.2011 Wage.ps.2011 Injury.2011
#> 31 ЦФО                  7.2       33.6        18800        86.0
#> 32 ЦФО                 11.4       30.8        15348       100.9
#> 33 ЦФО                 12.1       22.4        14312        93.1
#> 36 ЦФО                  8.1       34.1        15871        55.9
```

Согласно результатам корреляционного анализа, между зависимой и объясняющими переменными существует статистическая взаимосвязь. Построим модель множественной регрессии вида:

$$Retail.Vodka.2011.ps = a_0 + a_1*Rural.2011 + a_2*Wage.ps.2011 + a_3*Injury.2011$$

3.2 Оценка параметров модели линейной регрессии

Пример №4. Функция `lm()` (от англ. *linear model*) оценивает параметры линейной модели регрессии. По умолчанию для оценки используется метод

наименьших квадратов (МНК). В качестве аргументов укажем формулу взаимосвязи переменных в регрессии: `Retail.Vodka.2011.ps` зависит от факторов `Rural.2011`, `Wage.ps.2011` и `Injury.2011` – и фрейм с данными (`data`).

```
# Оценка параметров моделей ======================================
fit.1 <- lm(Retail.Vodka.2011.ps ~ Rural.2011 + Wage.ps.2011
            + Injury.2011,       # формула взаимосвязей переменных
            data = reg.df        # фрейм с исходными данными
            )
summary(fit.1)  # сводка по построенной модели

#> Call:
#> lm(formula = Retail.Vodka.2011.ps ~ Rural.2011 + Wage.ps.2011 +
#>     Injury.2011, data = reg.df)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -7.3270 -2.3200 -0.4987  1.8928  8.7583
#>
#> Coefficients:
#>                Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   1.352e+01  3.243e+00   4.169 8.22e-05 ***
#> Rural.2011   -1.718e-01  3.987e-02  -4.308 4.99e-05 ***
#> Wage.ps.2011 -3.268e-05  8.138e-05  -0.402    0.689
#> Injury.2011   3.325e-02  2.197e-02   1.513    0.134
#> ---
#> Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
#>
#> Residual standard error: 3.455 on 74 degrees of freedom
#> Multiple R-squared:  0.3523,   Adjusted R-squared:  0.3261
#> F-statistic: 13.42 on 3 and 74 DF,  p-value: 4.41e-07
```

Отчёт по модели включает сведения о вызове функции `lm()` (секция «Call»), описательные статистики остатков модели («Residuals»), таблицу коэффициентов («Coefficients») и характеристики качества модели: стандартную ошибку модели («Residual standard error»), множественный R-квадрат («Multiple R-squared»), скорректированный R-квадрат («Adjusted R-squared»), F-расчётное и соответствующее P-значение для проверки гипотезы о значимости модели в целом («F-statistic»). Модель значима (P-значение = $4{,}41*10^{-7}$ < 0,05), однако объясняет всего 32,61% разброса зависимой переменной с поправкой на степени свободы. Главной проблемой являются незначимые коэффициенты при факторах `Wage.ps.2011` и `Injury.2011`.

Согласно процедуре последовательного исключения незначимых факторов, первым следует убрать из модели наименее значимый фактор (с наибольшим P-значением или наименьшей по модулю t-статистикой). Это переменная «Wage.ps.2011». Перестраиваем модель без неё:

```
# модель без Wage.ps.2011
fit.2 <- lm(Retail.Vodka.2011.ps ~ Rural.2011 + Injury.2011,
            data = reg.df)

round(summary(fit.2)$coef, 4)  # только таблица с коэффициентами

#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  12.8523     2.7700  4.6398   0.0000
#> Rural.2011   -0.1660     0.0370 -4.4904   0.0000
#> Injury.2011   0.0323     0.0217  1.4862   0.1414
```

Коэффициент при `Injury.2011` снова оказался незначимым (P-значение = 0,1414 > 0,05). Исключаем его и перестраиваем модель. Теперь в ней остаётся только один фактор, поэтому назовём модель «`fit.X1`».

```
# модель парной регрессии
fit.X1 <- lm(Retail.Vodka.2011.ps ~ Rural.2011,
             data = reg.df)

summary(fit.X1)    # сводка по построенной модели

#> Call:
#> lm(formula = Retail.Vodka.2011.ps ~ Rural.2011, data = reg.df)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -8.0028 -2.2965 -0.3443  2.1272  7.5870
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 16.65235    1.07414  15.503  < 2e-16 ***
#> Rural.2011  -0.19481    0.03171  -6.143 3.43e-08 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> Residual standard error: 3.463 on 76 degrees of freedom
#> Multiple R-squared:  0.3318,    Adjusted R-squared:  0.323
#> F-statistic: 37.74 on 1 and 76 DF,  p-value: 3.43e-08
```

Можно убедиться, что исключение факторов из модели закономерно снизило скорректированный R-квадрат до 32,3%, однако значимость модели в целом возросла (F-статистика = 37,74).

Проверим, улучшит ли ситуацию включение в модель фиктивных переменных. Построим модель с переменной структурой, используя принадлежность каждого региона к одному из восьми федеральных округов. Включим фиктивные переменные как в константу, так и в коэффициенты. Общий вид модели с переменной структурой.

```
# модель с переменной структурой
fit.X1.fo <- lm(Retail.Vodka.2011.ps ~ Rural.2011 * FO,
                data = reg.df)
summary(fit.X1.fo)

#> Call:
#> lm(formula = Retail.Vodka.2011.ps ~ Rural.2011 * FO,
      data = reg.df)
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -6.8306 -1.4812  0.1891  1.4246  5.3084
#> Coefficients:
#>                     Estimate Std. Error t value Pr(>|t|)
#> (Intercept)         21.51912    2.39878   8.971 8.34e-13 ***
#> Rural.2011          -0.32319    0.09043  -3.574 0.000688 ***
#> FOПФО              -10.81045    3.81951  -2.830 0.006260 **
#> FOСЗФО              -8.11327    3.83103  -2.118 0.038211 *
#> FOСКФО             -12.60047    5.77786  -2.181 0.033000 *
#> FOСФО               -9.62085    3.04424  -3.160 0.002437 **
#> FOУФО               -9.56620    4.23692  -2.258 0.027488 *
#> FOЦФО               -4.65017    3.63340  -1.280 0.205369
#> FOЮФО              -12.73389    4.51155  -2.823 0.006396 **
#> Rural.2011:FOПФО    0.32666    0.13120   2.490 0.015479 *
#> Rural.2011:FOСЗФО   0.39322    0.14768   2.663 0.009863 **
#> Rural.2011:FOСКФО   0.20795    0.13473   1.543 0.127803
#> Rural.2011:FOСФО    0.31124    0.10363   3.003 0.003846 **
#> Rural.2011:FOУФО    0.20598    0.16418   1.255 0.214336
#> Rural.2011:FOЦФО    0.09908    0.13051   0.759 0.450599
#> Rural.2011:FOЮФО    0.25542    0.12755   2.003 0.049601 *
#> ---
#> Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
#>
#> Residual standard error: 2.569 on 62 degrees of freedom
#> Multiple R-squared:    0.7,    Adjusted R-squared:  0.6274
#> F-statistic: 9.644 on 15 and 62 DF,  p-value: 3.656e-11
```

Модель в целом значима, и скорректированный коэффициент детерминации у неё выше, чем у модели по всем регионам (62,74%). Однако у неё много незначимых параметров. Исключать их последовательно вручную трудоёмко, поэтому мы воспользуемся пользовательской функцией, которая проводит процедуру последовательного исключения регрессоров.

Сначала сгенерируем матрицу независимых переменных функцией `model.matrix()`.

```
# создаём фрейм со всеми переменными-факторами (создаём фиктивные)
X.matrix <- model.matrix(Retail.Vodka.2011.ps ~ Rural.2011 * FO,
                         data = reg.df)

# присоединяем независимую переменную
data.fit <- cbind(Retail.Vodka.2011.ps = reg.df$Retail.Vodka.2011.ps,
                  data.frame(X.matrix)[, -1])
```

Теперь загружаем функцию для исключения незначимых регрессоров из файла «removeFactorsByPValue.R» в рабочей директории и применяем её к модели с переменной структурой.

```
# функция с последовательным исключением аномальных
source('removeFactorsByPValue.R')
# доводим до значимости
fit.X1.fo <- removeFactorsByPValue(data = data.fit,
                                   y.var.name = 'Retail.Vodka.2011.ps')
summary(fit.X1.fo) # результат
<…>
#> Coefficients:
#>                    Estimate Std. Error t value Pr(>|t|)
#> (Intercept)        13.77202    0.97188  14.170  < 2e-16 ***
#> Rural.2011         -0.08867    0.03126  -2.837 0.005893 **
#> FOCKФO             -6.22800    1.32565  -4.698  1.2e-05 ***
#> FOЮФO              -4.13310    1.25261  -3.300 0.001499 **
#> Rural.2011.FOCЗФO   0.14499    0.03887   3.730 0.000375 ***
#> ---
#> Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
#>
#> Residual standard error: 2.782 on 73 degrees of freedom
#> Multiple R-squared:  0.5858,    Adjusted R-squared:  0.5631
#> F-statistic: 25.81 on 4 and 73 DF,  p-value: 2.382e-13
```

Значимы константы для Южного и Северо-Кавказского федеральных округов, а также коэффициент при независимой переменной для Северо-Западного федерального округа.

Для сравнения проделаем аналогичные шаги для второго по силе связи фактора: переменной `Injury.2011`.

```
# строим ПЛР на второй по силе корреляции фактор
fit.X2 <- lm(Retail.Vodka.2011.ps ~ Injury.2011, data = reg.df)
summary(fit.X2)      # результат

# модель с переменной структурой
fit.X2.fo <- lm(Retail.Vodka.2011.ps ~ FO + Injury.2011,
             data = reg.df)
summary(fit.X2.fo)   # результат

# доводим до состояния значимости
# создаём фрейм со всеми переменными-факторами (создаём фиктивные)
X.matrix <- model.matrix(Retail.Vodka.2011.ps ~ Injury.2011 * FO,
                         data = reg.df)
data.fit <- cbind(Retail.Vodka.2011.ps =
                                    reg.df$Retail.Vodka.2011.ps,
                  data.frame(X.matrix)[, -1])
head(data.fit)       # результат

# доводим до значимости с помощью пользовательской функции
fit.X2.fo <- removeFactorsByPValue(data = data.fit,
                                   y.var.name = 'Retail.Vodka.2011.ps')
summary(fit.X2.fo)   # результат
```

3.3 Сравнение нескольких моделей по качеству

Пример №5. Используем функцию `anova()`, чтобы проверить гипотезы об эквивалентности построенных моделей. Процедура проверяет нулевую гипотезу: пара следующих друг за другом моделей не отличается по качеству аппроксимации – против альтернативной: пара моделей значимо отличается.

```
# модели с фактором Rural.2011
anova(fit.X1, fit.X1.fo)
#>
#> Analysis of Variance Table
#> <…>
#>   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
#> 1     76 911.45
#> 2     73 564.95  3     346.5 14.924 1.136e-07 ***
```

```
# модели с фактором Injury.2011
anova(fit.X2, fit.X2.fo)

#>
#> Analysis of Variance Table
#> <…>
#>   Res.Df     RSS Df Sum of Sq      F    Pr(>F)
#> 1     76 1123.41
#> 2     62  379.87 14    743.54 8.6682 5.444e-10 ***
#> ---
#> Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Видно, что для двух рассматриваемых факторов добавление фиктивных переменных уменьшает остаточную дисперсию (RSS) и тем самым значимо повышает качество моделей (p-значения меньше 0,05).

Составим таблицу с некоторыми характеристиками качества четырёх построенных моделей. Включим в неё скорректированный R-квадрат, F-расчётное и стандартную ошибку. Считать эти показатели не нужно, достаточно извлечь их из объектов, содержащих построенные модели. Посмотрим на структуру любого объекта этого типа. Это список из 11 элементов, из которых нас интересуют три.

```
# ищем характеристики качества аппроксимации в отчёте
str(summary(fit.X1))

#>
#> List of 11
#> <…>
#> $ sigma         : num 3.46
#> $ adj.r.squared: num 0.323
#> $ fstatistic   : Named num [1:3] 37.7 1 76
#> <…>
```

Покажем на примере F-статистики, как извлечь данные.

```
# F-статистика: только значение F-расчётного
summary(fit.X1)$fstatistic[1]
#>    value
#> 37.74008
```

Извлечём эти статистики из всех построенных моделей регрессии.

```
# список построенных моделей
models.list <- list(fit.X1, fit.X1.fo, fit.X2, fit.X2.fo)
names(models.list) <- c('fit.X1', 'fit.X1.fo', 'fit.X2',
                        'fit.X2.fo')

# создаём фрейм под характеристики четырёх моделей
df.goodness.of.fit <- data.frame(Модель = names(models.list),
                        R.2.скорр = rep(0, length(models.list)),
                        F.расч = rep(0, length(models.list)),
                        Станд.Ошибка = rep(0, length(models.list)))
# заполняем его
for (i in 1:length(models.list)){
    # скорректированный R-квадрат
    df.goodness.of.fit[i, 'R.2.скорр'] <-
                round(summary(models.list[[i]])$adj.r.squared, 3)
    # F расчётное
    df.goodness.of.fit[i, 'F.расч'] <-
                round(summary(models.list[[i]])$fstatistic[1], 2)
    # стандартная ошибка
    df.goodness.of.fit[i, 'Станд.Ошибка'] <-
                        round(summary(models.list[[i]])$sigma, 1)
}

# результат
df.goodness.of.fit

#>      Модель R.2.скорр F.расч Станд.Ошибка
#> 1    fit.X1     0.323  37.74          3.5
#> 2 fit.X1.fo     0.563  25.81          2.8
#> 3    fit.X2     0.166  16.28          3.8
#> 4 fit.X2.fo     0.654  10.71          2.5
```

3.4 Сохранение рабочего пространства

Сохраним объекты, которые понадобятся в следующей практике, в файл «Пример_алкоголь_модели.RData» в рабочей директории.

```
# убираем промежуточные объекты
rm(i, df.goodness.of.fit, data.fit, X.matrix,
   fit.1, fit.2, fit.X1, fit.X2, fit.X1.fo, fit.X2.fo,
   SKFO.coef, SZFO.coef,
   removeFactorsByPValue)

# сохраняем рабочее пространство в файл
save.image('Пример_алкоголь_модели.RData')
```

4. УКАЗАНИЯ К ЛАБОРАТОРНОЙ РАБОТЕ №3: ТЕСТЫ ОСТАТКОВ

В третьей лабораторной мы продолжим работу с примером данных по реализации ликёро-водочных изделий в РФ в натуральном выражении за 2011 год и оценим проверим, удовлетворяют ли остатки четырёх построенных моделей условиям Гаусса-Маркова. Рассматриваются модели:

- **`fit.X1`** – зависимость розничной продажи ликёроводочных изделий (литров на человека за год; переменная Retail.Vodka.2011.ps), от удельного веса сельского населения (в процентах, оценка на конец года; переменная Rural.2011).
- **`fit.X1.fo`** – модель `fit.X1` со значимыми на уровне 0,05 фиктивными переменными, задающими принадлежность регионов к федеральным округам.
- **`fit.X2`** – зависимость розничной продажи ликёроводочных изделий (литров на человека за год), от заболеваемости на 1000 человек населения: травмы, отравления и некоторые другие последствия воздействия внешних причин (переменная Injury.2011).
- **`fit.X2.fo`** – модель `fit.X2` со значимыми на уровне 0,05 фиктивными переменными, задающими принадлежность регионов к федеральным округам.

В лабораторной работе рассматривается тесты остатков линейных регрессионных моделей в R. Используются данные:

- файл рабочего пространства R «Пример_алкоголь_модели.RData».

Используются пакеты:

```
library('lmtest')      # тесты остатков: bptest(), dwtest()
library('broom')       # трансформации данных: augment()
library('car')         # тест на мультиколинеарность: vif()
library('sandwich')    # оценки модели с поправкой на
                       #  гетероскедастичность: vcovHC()
```

4.1 Импорт данных

Исходные данные для работы были загружены в R и сохранены в файле рабочего пространства «Пример_алкоголь_модели.RData». Убедимся, что файл лежит в рабочей директории, и загрузим его.

```
# список файлов рабочей директории
dir()
```

```
#> <…>
#> [1] "Пример_алкоголь_модели.RData"

# загрузка объектов из сохранённого рабочего пространства
load('Пример_алкоголь_модели.RData')
# просмотр списка объектов
ls()
#> [1] "data.fit.X1.fo" "data.fit.X2.fo" "DF"
#> [4] "models.list"    "reg.df"
```

Построенные нами модели хранятся в списке `models.list`:

```
# названия моделей в списке
names(models.list)
#> [1] "fit.X1"    "fit.X1.fo" "fit.X2"    "fit.X2.fo"
```

Нам необходимо исследовать остатки этих моделей и определить, не нарушаются ли условия Гаусса-Маркова:

1. Равенство среднего остатков нулю.
2. Постоянство дисперсии остатков (гомоскедастичность).
3. Отсутствие автокорреляции (независимость).

Кроме того, модели множественной регрессии тестируют на отсутствие мультиколлинеарности факторов (их значимой взаимосвязи друг с другом). Мы проведём эту проверку исключительно в учебных целях, поскольку в нашем случае модели множественной регрессии содержат фиктивные переменные, которые априори взаимосвязаны с количественной независимой переменной.

4.2 Графики остатков и поиск влияющих наблюдений

Пример №6. Представим графически остатки четырёх моделей. Функция `plot()` от аргумента типа «модель регрессии», построенного функцией `lm()`, строит шесть графиков остатков (значение второго аргумента – номер графика):

1. «Residuals versus Fitted» – график зависимости остатков модели (по вертикали) от модельных значений (по горизонтали). При наличии явных нарушений условий Гаусса-Маркова они проявятся на этом графике.
2. «Normal Q-Q» – график «квантиль-квантиль», который сопоставляет квантили фактического распределения стандартизованных остатков (по вертикали) с соответствующими теоретическими значениями квантилей нормального распределения (по горизонтали). Чем ближе все точки в прямой, тем «нормальнее» распределение остатков.

3. «Scale-Location» – график, на котором по вертикали отложены корни из стандартизованных значений остатков, а по горизонтали – модельные значения. Этот график служит для поиска гетероскедастичности (непостоянства дисперсии) остатков.
4. «Cook's Distance» – график расстояний Кука. С помощью расстояния Кука оценивают влияние отдельного наблюдения выборки на модель регрессии. Эта величина показывает разницу между вычисленными коэффициентами уравнения регрессии и значениями, которые получились бы при исключении соответствующего наблюдения:

$$D_i = \frac{\sum_{j=1}^{n}\left(\hat{Y}_j - \hat{Y}_{j(i)}\right)^2}{p \cdot MSE}$$

где $\hat{Y}_j$ – прогноз по модели, построенной по всей выборке, для j-ого наблюдения; $\hat{Y}_{j(i)}$ – предсказание регрессионной модели, построенной по выборке без i-ого наблюдения, получаемое для j-ого наблюдения; p – количество параметров модели; MSE – среднеквадратичная ошибка модели[12].
5. «Residuals versus Leverage» – зависимость стандартизованных остатков модели (по вертикали) от оценок их влияния на модель регрессии. Эти оценки («leverage») рассчитываются по формуле:

$$h_i = x_i^T (X^T X)^{-1} x_i$$

где x_i – вектор-столбец значений независимых переменных модели в i-м наблюдении, X – матрица независимых переменных[13].
6. «Cook's Distance versus Leverage» – зависимость расстояния Кука для каждого наблюдения (вертикальная ось) от оценки влияния на модель (по горизонтали).

Некоторое представление о видах влияющих наблюдений на примере графика парной линейной регрессии даёт рисунок 9. Очевидно, что наблюдение может быть аномальным по величине Y и X (случай c), и при этом укладываться в модель, не оказывая на неё заметного влияния. Самое большое искажение возникает в случае b, когда влияющее наблюдение изменяет и константу, и коэффициент модели. Это характерно для наблюдений, аномальных по X. Если же влияющее наблюдение аномально только по Y, то, вероятно, пострадает только константа (случай a).

[12] Расстояние Кука / Портал machinelearning.ru. URL: http://www.machinelearning.ru/wiki/index.php?title=Расстояние_Кука
[13] Cook's Distance / From Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Cook's_distance

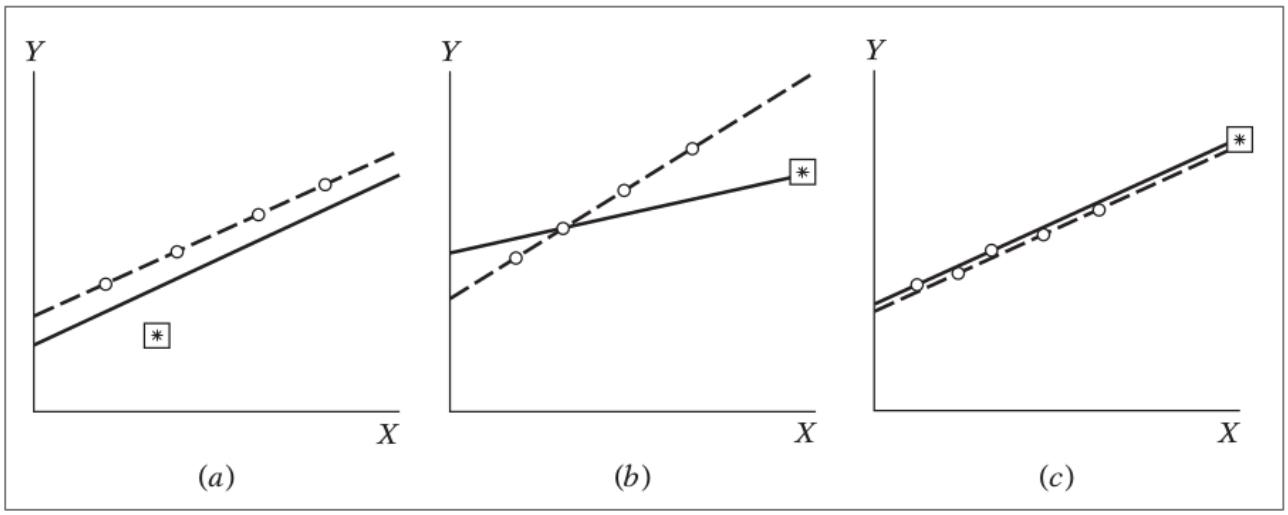

Рис. 9. Три случая влияющих наблюдений; жирная прямая – модель по всем наблюдениям; пунктир – модель без влияющего наблюдения (обозначено звёздочкой) [8]

Построим эти четыре графика для остатков каждой из четырёх моделей. Читателю предлагается построить и проанализировать эти графики самостоятельно.

```
# графики остатков: цикл по моделям в списке models.list
for (i in 1:length(models.list)) {
    png(paste('RPlot', i, '.png', sep = ''),
        height = 500, width = 500) # открываем вывод в файл
    par(mfrow = c(2, 2))           # делим полотно на 4 части

    # рисуем 4 графика для одной и той же модели
    plot(models.list[[i]], 1)
    plot(models.list[[i]], 2)
    plot(models.list[[i]], 3)
    plot(models.list[[i]], 5)

    # добавляем общий заголовок с названием модели
    mtext(paste('Остатки модели ',
                names(models.list)[i], sep = ''),
          side = 3, line = -2, outer = TRUE, cex = 1.2)
    par(mfrow = c(1, 1))
    dev.off()                      # закрываем вывод в файл
}
```

На графиках остатков не видно явных ошибок спецификации. В последней модели обнаруживаются два влияющих наблюдения: регионы с номерами 72 и 8. Это Тюменская область и Руспублика Калмыкия:

```
# Регионы с номерами 72 и 8
DF[rownames(DF) %in% c(8, 72), c('Label', 'FO')]

#>                     Label     FO
#> 8   Республика Калмыкия    ЮФО
#> 72    Тюменская область    УФО
```

Чтобы понять, нужно ли исключать эти наблюдения, нужно установить, насколько сильно они отличаются от всех остальных. Можно использовать тот факт, что расстояние Кука распределено по закону Фишера со степенями свободы $\nu_1 = p,\ \nu_2 = n - p$ (p – количество параметров модели, n – число наблюдений) и проверить гипотезу:

H_0: отличие *i*-го наблюдения от остальных статистически незначимо.

H_1: i-е наблюдение значимо отличается от остальных (и его нужно убрать как влияющее на модель).

Для проверки гипотезы достаточно сравнить расстояния Кука для влияющих наблюдений с квантилем F-распределения, соответствующим заданному уровню значимости. Если расстояния меньше квантиля (критической границы), нулевая гипотеза не отклоняется. Для модели №4:

```
#  1. добавляем переменные, рассчитанные по модели, к данным
h <- augment(models.list[[4]], reg.df)
rownames(h) <- rownames(reg.df)

#  2. вытаскиваем расстояния Кука во влияющих наблюдениях
lev <- h[rownames(reg.df) %in% c(8, 72), '.cooksd', drop = F]

# критическая граница F-распределения
n <- nrow(reg.df)
k <- nrow(summary(models.list[[4]])$coeff)
f.tabl <- qf(1 - 0.05, df1 = k, df2 = n - k)
f.tabl
#> [1] 1.809466

# 3. сравниваем расчётные значения с критической границей
cbind(leverage = round(lev, 2),
    f.tabl = round(f.tabl, 2),
    p.val = round(pf(lev$.cooksd, df1 = p, df2 = n - p), 4))
#>    .cooksd f.tabl  p.val
#> 8     0.92   1.81 0.4519
#> 72    0.51   1.81 0.0692
```

Поскольку расстояния Кука не превосходят критическую границу (и, следовательно, соответствующие p-значения больше 0,05), принимаем нулевые гипотезы для регионов 8 и 72: их не стоит убирать из выборки для построения четвёртой модели.

4.3 Тесты на равенство среднего остатков нулю

Пример №7. Проверим гипотезу о равенстве среднего остатков модели нулю. Эта гипотеза проверяется с помощью t-критерия.

H_0: среднее остатков равно 0 в генеральной совокупности.

H_1: среднее остатков не равно 0 в генеральной совокупности.

Если p-значение для статистики теста больше уровня значимости (0,05), нулевая гипотеза не отклоняется.

```
# номер модели
i <- 1
# t-тест для среднего
t.test(models.list[[i]]$residuals, mu = 0,
       alternative = 'two.sided')

#> One Sample t-test
#> data:  models.list[[i]]$residuals
#> t = -3.5039e-16, df = 77, p-value = 1
#> alternative hypothesis: true mean is not equal to 0
#> 95 percent confidence interval:
#>  -0.7757137  0.7757137
#> sample estimates:
#>     mean of x
#> -1.364983e-16
```

Для первой модели нулевая гипотеза не отклоняется. Несложно убедиться, что остатки других моделей также проходят этот тест.

4.4 Тесты на гетероскедастичность

Пример №8. Проверим остатки на постоянство дисперсии четырьмя тестами:

1) тест Бройша-Пагана[14];
2) тест Уайта0[15];

[14] Тест Бройша-Пагана / Википедия — свободная энциклопедия. URL: https://ru.wikipedia.org/wiki/ Тест_Бройша_—_Пагана

[15] Тест Уайта / Википедия — свободная энциклопедия. URL: https://ru.wikipedia.org/wiki/Тест_Уайта

3) тест Голдфельда-Квандта[16];
4) тест Глейзера[17].

Нулевые гипотезы у всех тестов совпадает по смыслу.

H_0: остатки модели гомоскедастичны.

H_1: в остатках модели есть гетероскедастичность.

Если p-значение для статистики теста больше уровня значимости (0,05), нулевая гипотеза не отклоняется.

```
# номер модели в списке
i <- 1

# тест Бройша-Пагана на гетероскедастичность
bptest(models.list[[i]])

#> studentized Breusch-Pagan test
#> data:  models.list[[i]]
#> BP = 2.4022, df = 1, p-value = 0.1212
```

Результат первого тест говорит о том, что в остатках первой модели нет гетероскедастичности. Для следующего теста нам понадобится создать вспомогательную таблицу, присоединив ко фрейму с исходными данными характеристики наблюдений, полученные по модели. Сделаем это с помощью функции `augment()`, которая извлекает из объекта типа «линейная модель» переменные, среди которых модельные значения, остатки, оценки влияния, расстояния Кука и присоединяет их к указанному фрейму.

В новом фрейме (объект **`h`**):

1) **`.fitted`** – модельные значения для каждого наблюдения;
2) **`.se.fit`** – стандартные ошибки прогноза каждого наблюдения;
3) **`.resid`** – остатки модели;
4) **`.resid`** – оценки влияния (h_i);
5) **`.sigma`** – оценка стандартной ошибки для этой модели, из которой убрали данное наблюдение;
6) **`.cooksd`** – расстояния Кука;
7) **`.std.resid`** – стандартизованные остатки.

```
# добавляем в исходную таблицу h прогнозы, остатки из модели model
```

[16] Тест Голдфелда — Куандта / Википедия — свободная энциклопедия. URL: https://ru.wikipedia.org/wiki/Тест_Голдфелда_—_Куандта

[17] Тест Глейзера / Википедия — свободная энциклопедия. URL: https://ru.wikipedia.org/wiki/Тест_Глейзера

```
h <- augment(models.list[[i]], reg.df)
str(h) # смотрим, что добавилось в таблицу h
#> 'data.frame':    78 obs. of  13 variables:
#>  $ .rownames           : chr  "31" "32" "33" "36" ...
#>  $ FO                  : Factor w/ 8 levels "ДВФО",..: 7 7...
#>  $ Retail.Vodka.2011.ps: num  7.2 11.4 12.1 8.1 12 11.4 ...
#>  $ Rural.2011          : num  33.6 30.8 22.4 34.1 19 24.1 ...
#>  $ Wage.ps.2011        : int  18800 15348 14312 15871 ...
#>  $ Injury.2011         : num  86 100.9 93.1 55.9 95.6 ...
#>  $ .fitted             : num  10.1 10.7 12.3 10 13 ...
#>  $ .se.fit             : num  0.398 0.393 0.488 0.4 0.558 ...
#>  $ .resid              : num  -2.907 0.748 -0.189 -1.909 ...
#>  $ .hat                : num  0.0132 0.0129 0.0198 0.0134 ...
#>  $ .sigma              : num  3.47 3.48 3.49 3.48 3.48 ...
#>  $ .cooksd             : num  4.77e-03 3.08e-04 3.06e-05 ...
#>  $ .std.resid          : num  -0.845 0.217 -0.055 -0.555 ...
```

Тест Уайта проведём той же функцией, что и тест Бройша-Пагана, указав форму взаимосвязи для вспомогательной регрессии в аргументе `varformula`.

```
# тест Уайта
# Во вспомогательной регрессии e^2 зависят от X и X^2
#  для моделей 1-2 X: Rural.2011;
#  для моделей 3-4 X: Injury.2011
bptest(models.list[[i]], data = h,
       varformula = ~ Rural.2011 + I(Rural.2011^2))
#> studentized Breusch-Pagan test
#> data:  models.list[[i]]
#> BP = 2.1348, df = 2, p-value = 0.3439
```

Тест Уайта также говорит нам, что гетероскедастичности в остатках нет.

Тот же результат по тесту Голдфельда-Квандта (аргумент `fraction` указывает, какую долю упорядоченных по возрастанию фактора наблюдений выбросить перед сравнением дисперсий в начале и в конце выборки).

```
# тест Голдфельда-Квандта
# для моделей 1-2: Rural.2011; для моделей 3-4: Injury.2011
gqtest(models.list[[i]], order.by = ~ Rural.2011,
       data = h, fraction = 0.2)
#> Goldfeld-Quandt test
#> GQ = 1.2936, df1 = 30, df2 = 29, p-value = 0.2452
```

Наконец, тест Глейзера подразумевает построение нескольких вспомогательных регрессий, которые отличаются степенью (β), в которую

возводится независимая переменная – ведущий фактор модели. Сохраним последовательность степеней в векторе и запустим цикл по элементам этого вектора. В случае, если среди вспомогательных регрессий обнаружится значимая, выведем значение степени β и p-значение для коэффициента модели.

```
# Тест Глейзера
# вектор степеней независимой переменной
beta.vector <- seq(-1, 1.5, by = 0.05)
# убираем значение степени 0
beta.vector <- beta.vector[beta.vector != 0]

# строим вспомогательные регрессии, и если коэффициент модели
#  значим, выводим p-значение и степень.
#    для моделей 1-2 X: Rural.2011;
#    для моделей 3-4 X: Injury.2011
for (j in 1:length(beta.vector)){
    # оценка вспомогательной парной регрессии
    gl.test <- lm(abs(.resid) ~ I(Rural.2011^beta.vector[j]),
                  data = h)
    if(summary(gl.test)$coef[2, 4] < 0.05){
        # собираем сообщение и выводим в консоль
        print(paste('beta =', beta.vector[j],
                    'p-value =',
                    round(summary(gl.test)$coef[2, 4], 4)))
    }
}
```

Поскольку ни одна из вспомогательных регрессий для тест Глейзера не оказалась значимой, нулевая гипотеза (остатки гомоскедастичны) не отвергается.

Читателю предлагается убедиться, что тесты показывают наличие гетероскедастичности в остатках третьей модели (тест Бройша-Пагана: p-значение = 0,0291; тест Глейзера: минимальное p-значение = 0,0081).

4.5 Тесты на автокорреляцию

Пример №9. Протестируем остатки моделей на автокорреляцию. Все модели содержат константу, поэтому можно проверить остатки на наличие автокорреляции первого порядка с помощью статистики Дарбина-Уотсона[18].

H_0: в остатках отсутствует автокорреляция первого порядка.

[18] Критерий Дарбина — Уотсона / Википедия — свободная энциклопедия. URL: https://ru.wikipedia.org/wiki/Критерий_Дарбина_—_Уотсона

H_1: автокорреляция первого порядка в остатках есть (двусторонняя альтернативная гипотеза, `alternative = 'two.sided'`).

Если p-значение > 0,05, нулевая гипотеза не отклоняется.

```
# номер модели в списке
i <- 1
# тест Дарбина-Уотсона на автокорреляцию
dwtest(models.list[[i]], alternative = 'two.sided')
#> Durbin-Watson test
#> data:  models.list[[i]]
#> DW = 1.1492, p-value = 5.985e-05
#> alternative hypothesis: true autocorrelation is not 0
```

Таким образом, в остатках первой модели обнаружена автокорреляция. Можно убедиться, что автокорреляция отсутствует только в остатках четвёртой модели (p-значение = 0,4655). Другой тест на автокорреляцию состоит в расчёте и проверке на значимость автокорреляционного коэффициента первого порядка для остатков модели.

H_0: коэффициент автокорреляции незначим, в остатках отсутствует автокорреляция первого порядка.

H_1: автокорреляция первого порядка в остатках есть.

Если p-значение > 0,05, нулевая гипотеза не отклоняется.

```
# количество наблюдений = числу строк во фрейме с данными
n <- nrow(reg.df)
# автокорреляционный коэффициент первого порядка для остатков
cor.test(x = models.list[[i]]$residuals[1:(n-1)],
         y = models.list[[i]]$residuals[2:n])
#>
#> Pearson's product-moment correlation
#> <…>
#> t = 3.8888, df = 75, p-value = 0.000216
#>alternative hypothesis: true correlation is not equal to 0
#> 95 percent confidence interval:
#>  0.2044158 0.5803690
#> sample estimates:
#>       cor
#> 0.4096394
```

Результаты этого теста совпадают с результатами теста на основе статистики Дабрина-Уотсона на уровне значимости 0,10.

4.6 Переоценка параметров с учётом ошибок спецификации

Главный негативный результат наличия гетероскедастичности и автокорреляции в остатках состоит в том, что традиционным критериям значимости параметров и модели в целом нельзя доверять, поскольку оценки стандартных ошибок параметров смещены. Поэтому в случае, если в остатках обнаружены ошибки, следует повторно провести оценку стандартных ошибок, используя специальные методы, в основе которых лежит взвешивание наблюдений с помощью ковариационной матрицы. В R реализован ряд алгоритмов оценки этой матрицы.

Пример №10. Пересчитаем ошибки параметров первой модели и перепроверим гипотезы об их значимости с учётом автокорреляции. Воспользуемся функцией `coeftest()` с аргументом `vcov.` – ковариационная матрица, которую можно оценить функциями:

- **`vcovHC()`** – оценка ковариационной матрицы, устойчивая к гетероскедастичности;
- **`vcovHAC()`** – оценка ковариационной матрицы, устойчивая к гетероскедастичности и автокорреляции;

Исходные оценки параметров модели и их стандартных ошибок без поправок, таковы:

```
# оценки параметров модели 1 по МНК
coeftest(models.list[[1]])
#>
#> t test of coefficients:
#>              Estimate Std. Error t value  Pr(>|t|)
#> (Intercept) 16.652353   1.074142 15.5029 < 2.2e-16 ***
#> Rural.2011  -0.194814   0.031712 -6.1433  3.43e-08 ***
#> ---
#> Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Рассчитаем ошибки параметров с поправками:

```
# робастные (т.е. устойчивые) оценки стандартных ошибок моделей
coeftest(models.list[[1]], vcov. = vcovHAC(models.list[[1]]))
#>
#> t test of coefficients:
#>              Estimate Std. Error t value  Pr(>|t|)
#> (Intercept) 16.652353   1.718587  9.6896 6.445e-15 ***
#> Rural.2011  -0.194814   0.057802 -3.3704  0.001182 **
#> ---
#> Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Видно, что скорректированные стандартные ошибки параметров (Столбец «Std. Error») больше исходных. Соответственно, уменьшаются t-статистики и P-значения. Однако параметры остаются значимыми. Это означает, что с поправкой на автокорреляцию модель остаётся значимой и ей можно пользоваться для интерпретации зависимости *Y* от факторов и для прогноза. Читатель может самостоятельно убедиться, что коэффициенты моделей №2 и 3, в остатках которых также была обнаружена гетероскедастичность, после переоценки ошибок остаются значимыми на уровне 0,05.

4.7 Тест на мультиколлинеарность факторов модели

Для проверки модели на мультиколлинеарность используются VIF-коэффициенты, которые в R рассчитывает одноимённая функция `vif()`.

Пример №11. В учебных целях протестируем факторы второй модели на мультиколлинеарность (из-за того, что среди факторов всего один количественный показатель, а все остальные – фиктивные переменные, высокая мультиколлинеарность в некоторых случаях неизбежна).

```
# VIF-тест на мультиколлинеарность факторов
#  (ПРИМЕНЯЕТСЯ ДЛЯ МНОЖЕСТВЕННОЙ РЕГРЕССИИ)
round(vif(models.list[[2]]), 2)
#>
#>           Rural.2011             FOSKFO             FOUFO
#>                 1.51               1.45              1.12
#>   Rural.2011.FOSZFO
#>                1.03
```

Поскольку значения VIF-коэффициентов близки к единице, в модели нет мультиколлинеарности факторов.

5. ПУБЛИКАЦИЯ ОТЧЁТА

Результаты выполнения лабораторной работы:

1. Скрипт с кодом на RMarkdown (расширение «.Rmd»), генерирующий отчёт. Скрипт необходимо разместить в репозитории на GitHub. В репозитории также должен присутствовать файл с исходными данными для лабораторной работы.
2. Сгенерированный файл отчёта с расширением «.html», размещённый на RPubs. Блоки кода скрыты, в тексте должны быть список переменных, размерность данных, постановка задачи. Отчёт должен содержать как минимум один блок кода, встроенный в строку текста.

5.1 Публикация отчёта на RPubs

Для начала необходимо создать профиль на rpubs.com, для этого требуется адрес электронной почты. Далее, после того как отчёт «связан» в html-файл (рисунок 10), он открывается в окне просмотрщика RStudio. На правой верхней рамке окна есть кнопка «опубликовать» (рисунок 11). В диалоговом окне необходимо выбрать вариант «RPubs». Появится окно с предупреждением о том, что всё, опубликованное на rpubs.com, попадает в публичный доступ – нажмите «Publish» (рисунок 12).

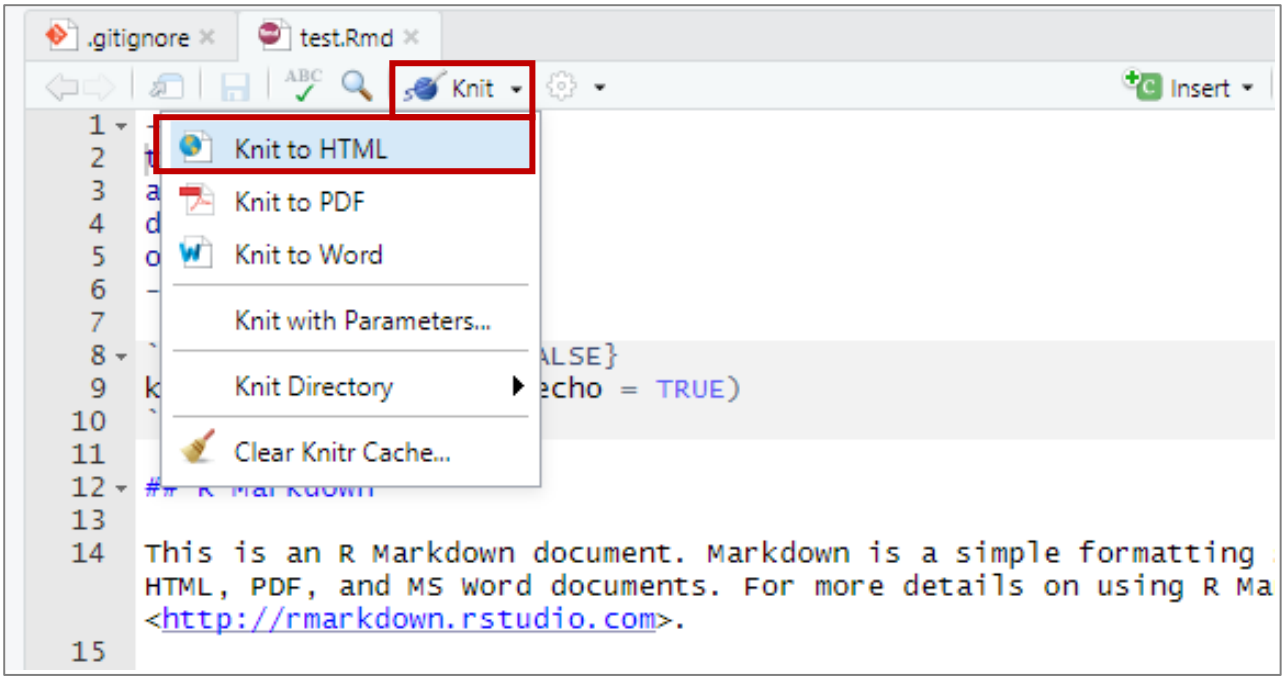

Рис. 10. Связываем html-отчёт для публикации на RPubs

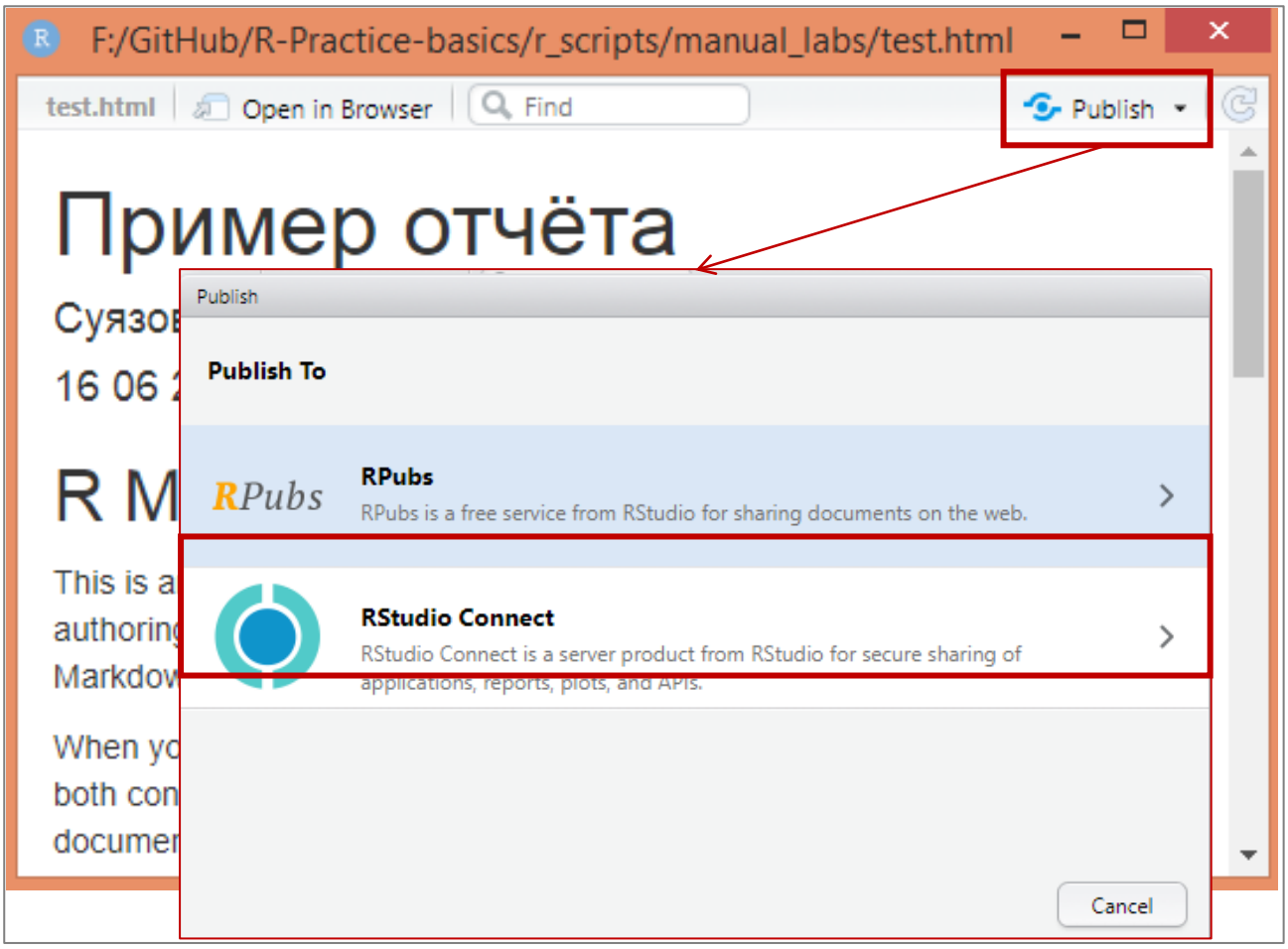

Рис. 11. Публикуем html-отчёт на RPubs (шаг 1)

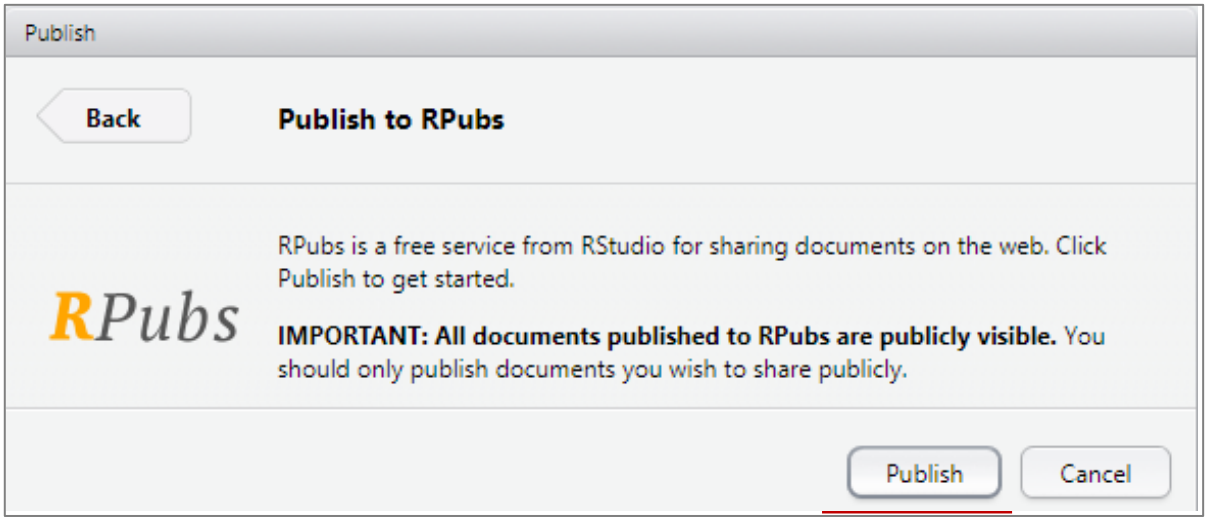

Рис. 12. Публикуем html-отчёт на RPubs (шаг 2)

После чего RStudio передаст управление браузеру, откроется страница авторизации на rpubs.com. После авторизации необходимо ввести название отчёта, путь к нему и короткий комментарий к содержимому в веб-форму (рисунок 13).

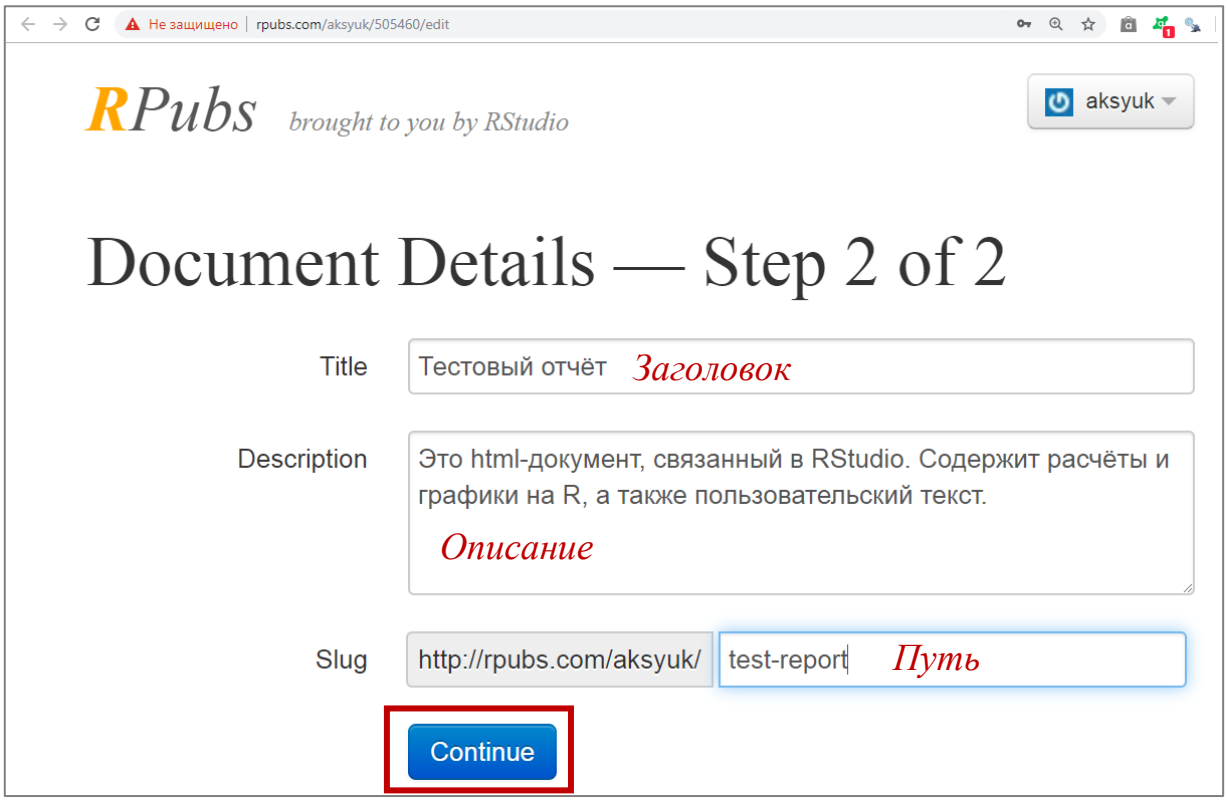

Рис. 13. Публикуем html-отчёт на RPubs (шаг 3)

После публикации по заданному адресу появится веб-страница с отчётом.

5.2 Размещение кода в репозитории GitHub

Github – это, прежде всего, система контроля версий, используемая для разработки программного обеспечения. Ещё Github – это универсальная открытая площадка для работы с данными, учебными материалами и программными кодами, и мы будем пользоваться ей именно в таком ключе.
Для работы нужна регистрация на сайте. Это бесплатная процедура, для которой понадобится только электронный почтовый ящик.

После регистрации можно создавать свои репозитории – аналоги папок, в которых размещаются файлы с кодами, данными, отчётами по проекту. Обратите внимание, что у Github есть ограничение на размер хранимых файлов. Репозиторий – это версия проекта, которая хранится на сервере. В то время как для непосредственной работы с проектом используется локальная версия репозитория. Чтобы получить её, нужно *клонировать* (команда «clone») репозиторий на локальный компьютер.

После клонирования репозитория все файлы внутри папки с его локальной версией будут постоянно проверяться на обновления. Это касается любых действий, будь то добавление, удаление, изменение или создание файлов и каталогов внутри локального репозитория.

Когда изменения, сделанные локально, нужно закрепить на сервере, осуществляется *коммит* (команда «commit») – закрепление изменений и их подготовка к отправке на сервер. После чего изменения отправляются на сервер командой «push».

Ключевой функционал git – это *ветвление* проекта («fork»), и запрос на внесение изменений («pull request») которое, собственно, и позволяют вести совместную разработку. Ветвление – это создание своей версии чужого репозитория. При этом файлы репозитория-источника полностью переносятся в новый репозиторий вашего аккаунта. Если внесённые в ветвь изменения значимы для первоначального проекта, вы можете запросить автора исходного репозитория внести сделанные вами правки обратно.

Все действия по индексации изменений, а также по перемещению файлов между локальным компьютером и сервером осуществляются с помощью программы git, версии которой есть для всех популярных операционных систем.

В данной практике для работы с Git будет достаточно интерфейса сайта https://github.com/. Опишем основные действия, необходимые для выполнения упражнения 1. Для начала нужно зарегистрироваться на Github (рисунок 14) и войти под своим аккаунтом.

Сделать свою ветвь чужого репозитория. Репозиторий с кодами для данных методических указаний находится по адресу: https://github.com/aksyuk/R-Practice-basics. На рисунке 15 показано, как сделать форк, или другими словами, скопировать его в свой аккаунт.

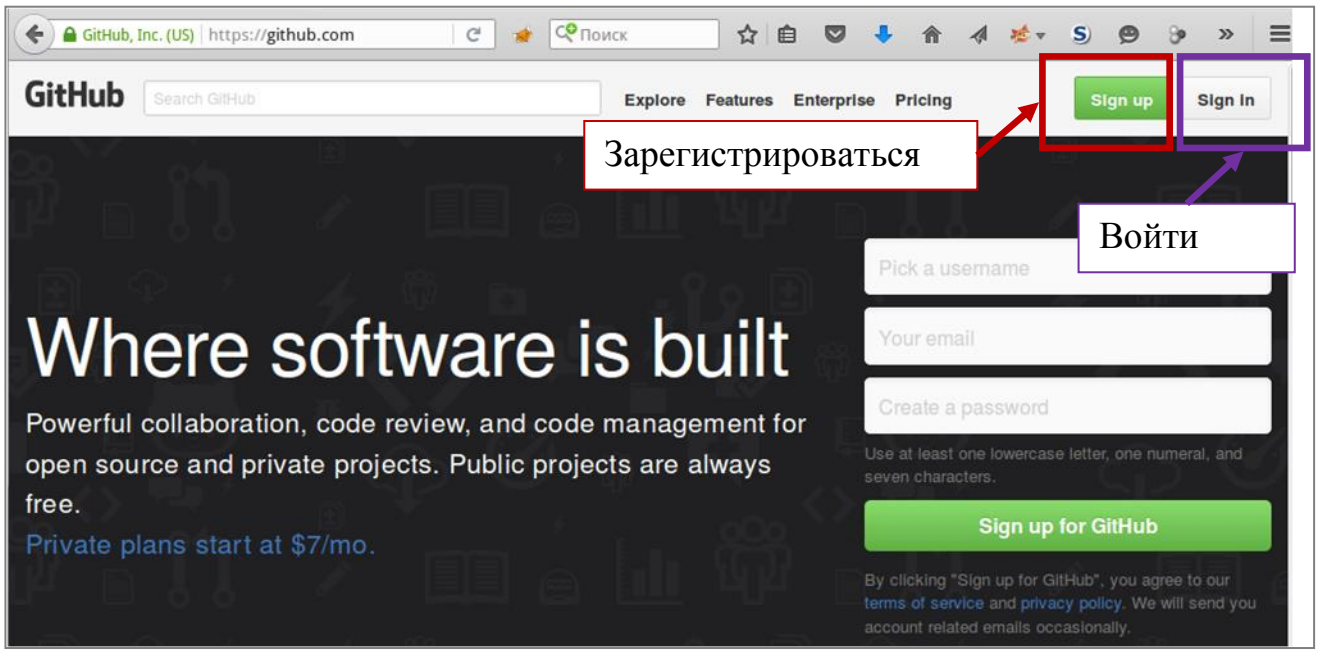

Рис. 14. Окно приветствия Github с кнопкой регистрации

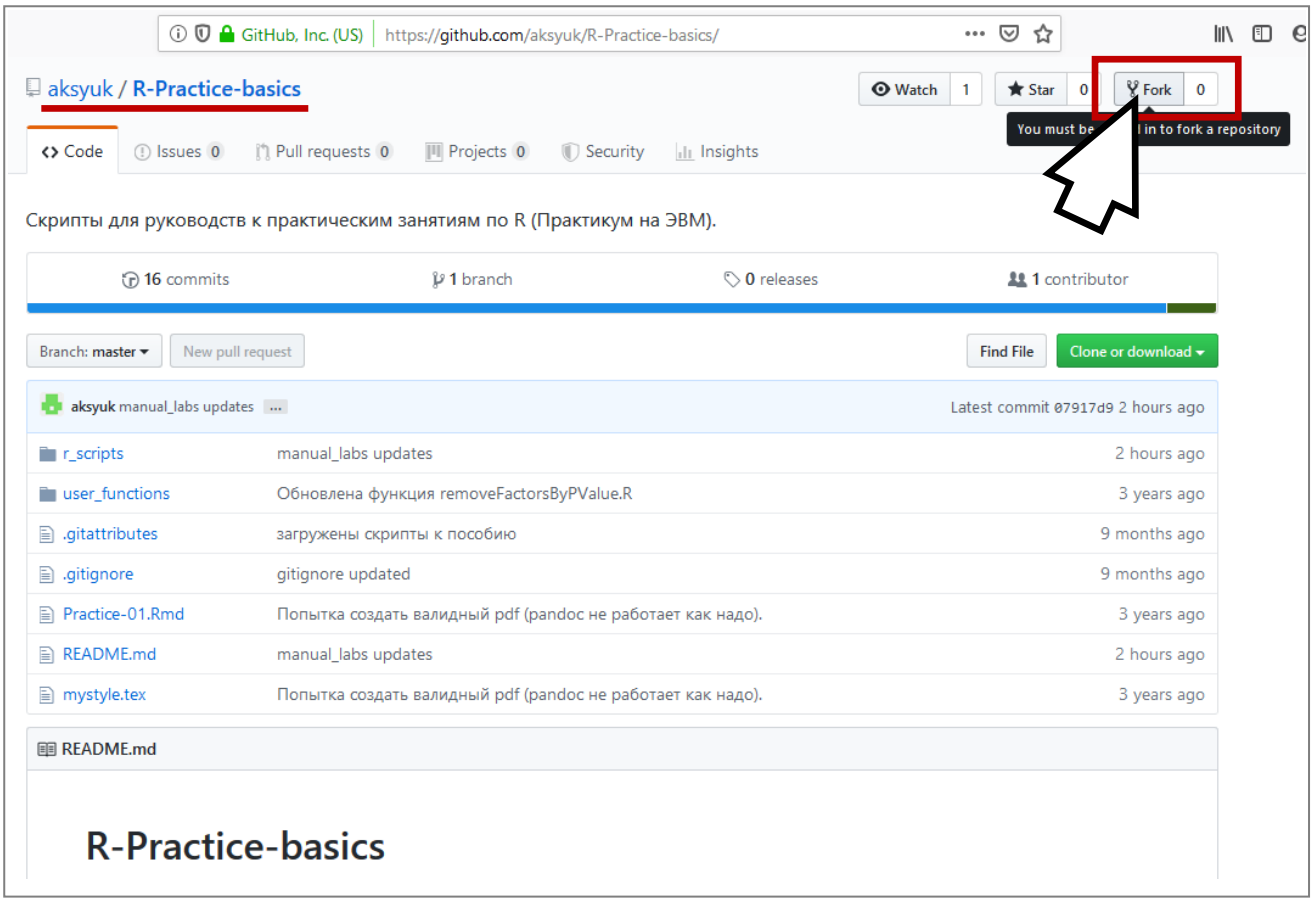

Рис. 15. Создать форк репозитория

Обновить файлы своего репозитория через веб-интерфейс. В случае если необходимо внести минимальные правки, не загружая файлы репозитория на локальный компьютер, можно воспользоваться веб-интерфейсом. Можно открыть исходник («raw»), например, README.md (рисунки 16 – 18).

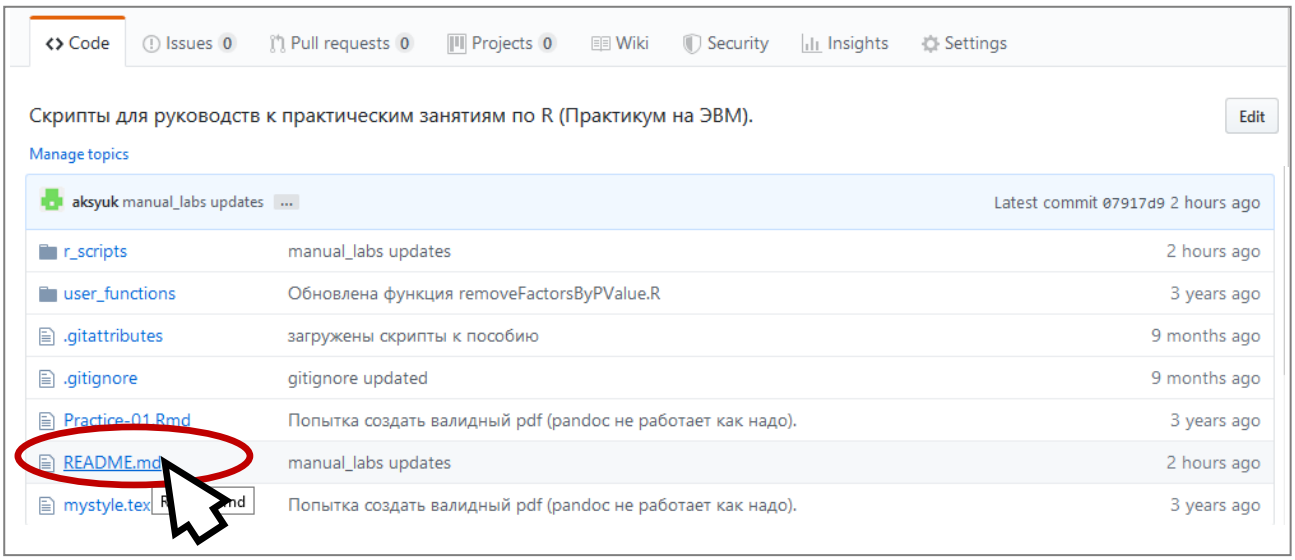

Рис. 16. Выбрать файл README.md для редактирования в веб-интерфейсе

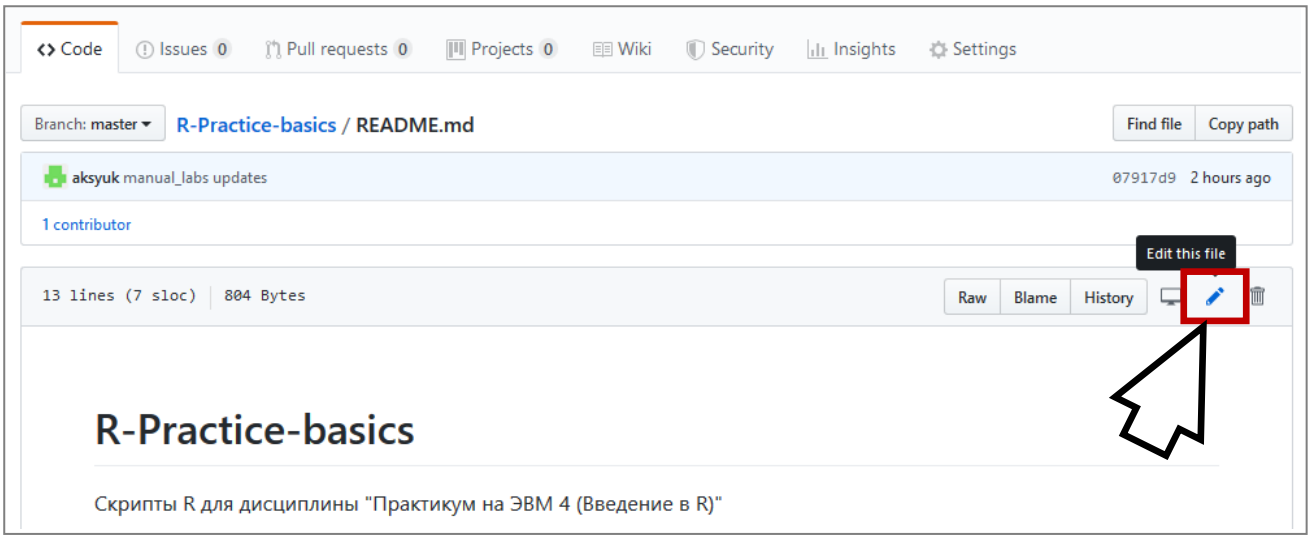

Рис. 17. Перейти в режим редактирования файла на сайте

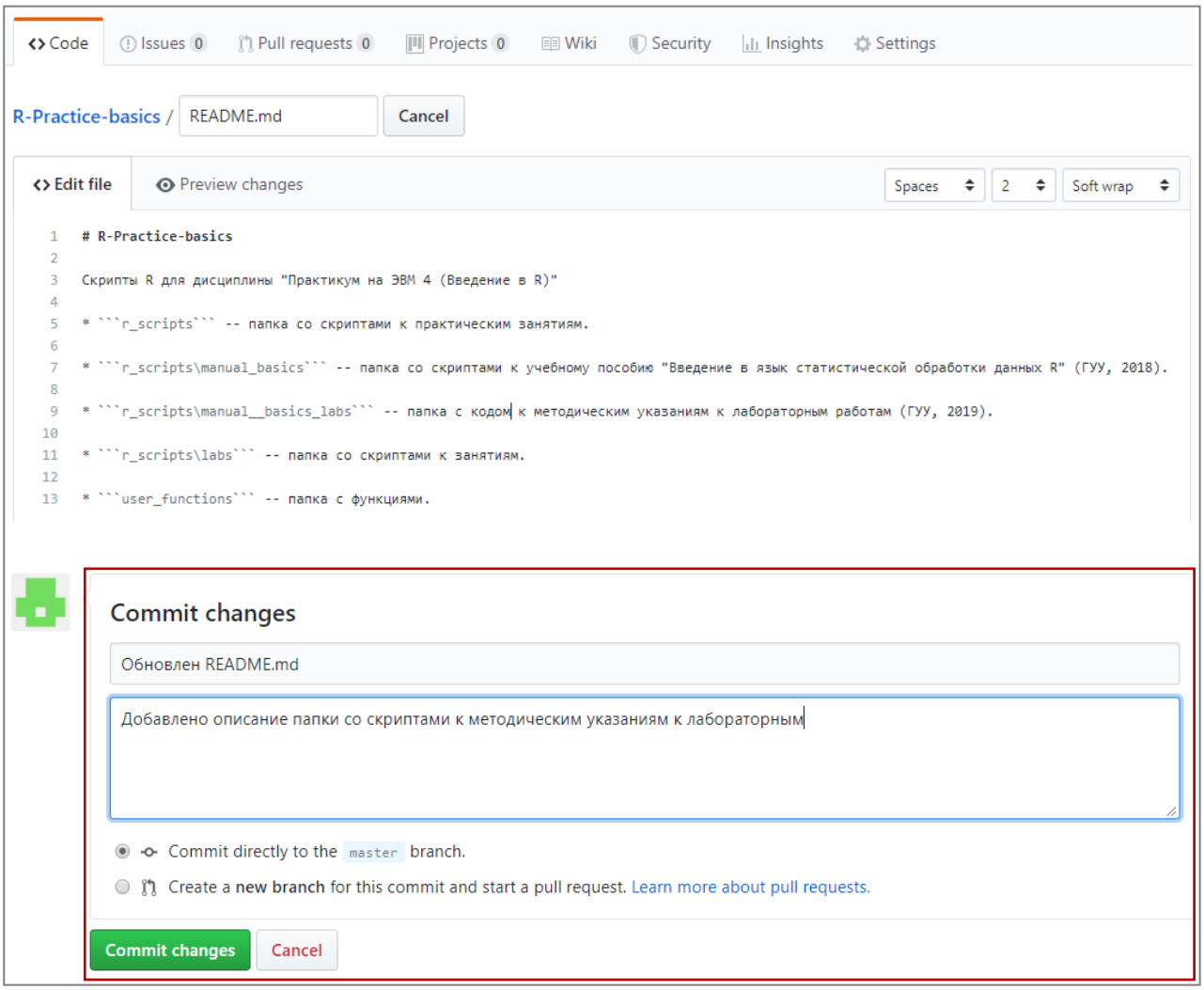

Рис. 18. Отредактировать файл через веб-интерфейс
и сделать коммит изменений

После внесения правок файл необходимо сохранить, сделав коммит. Не игнорируйте комментарий к коммиту, даже если вносимые изменения минимальны: это привычка, которая выручает при работе над большими проектами, и просто правило хорошего тона.

Итак, возможностей сайта Github достаточно, чтобы делать простые операции по размещению файлов лабораторных, созданных локально. Однако программа GitHub Desktop даёт более широкие возможности и удобнее в использовании.

5.3 Обзор GitHub Desktop

Скачать дистрибутив программы можно по адресу: https://desktop.github.com/. Установить программу на компьютер под управлением Windows можно и без прав администратора. После установки необходимо ввести данные своего аккаунта на github.com.

С помощью GitHub Desktop можно работать локально с копиями своих репозиториев, программа автоматически отслеживает изменения по сравнению с версиями репозиториев, размещёнными на серверах GitHub. Клонируем свою копию репозитория на локальный компьютер: File → Clone Repository… (рисунок 19).

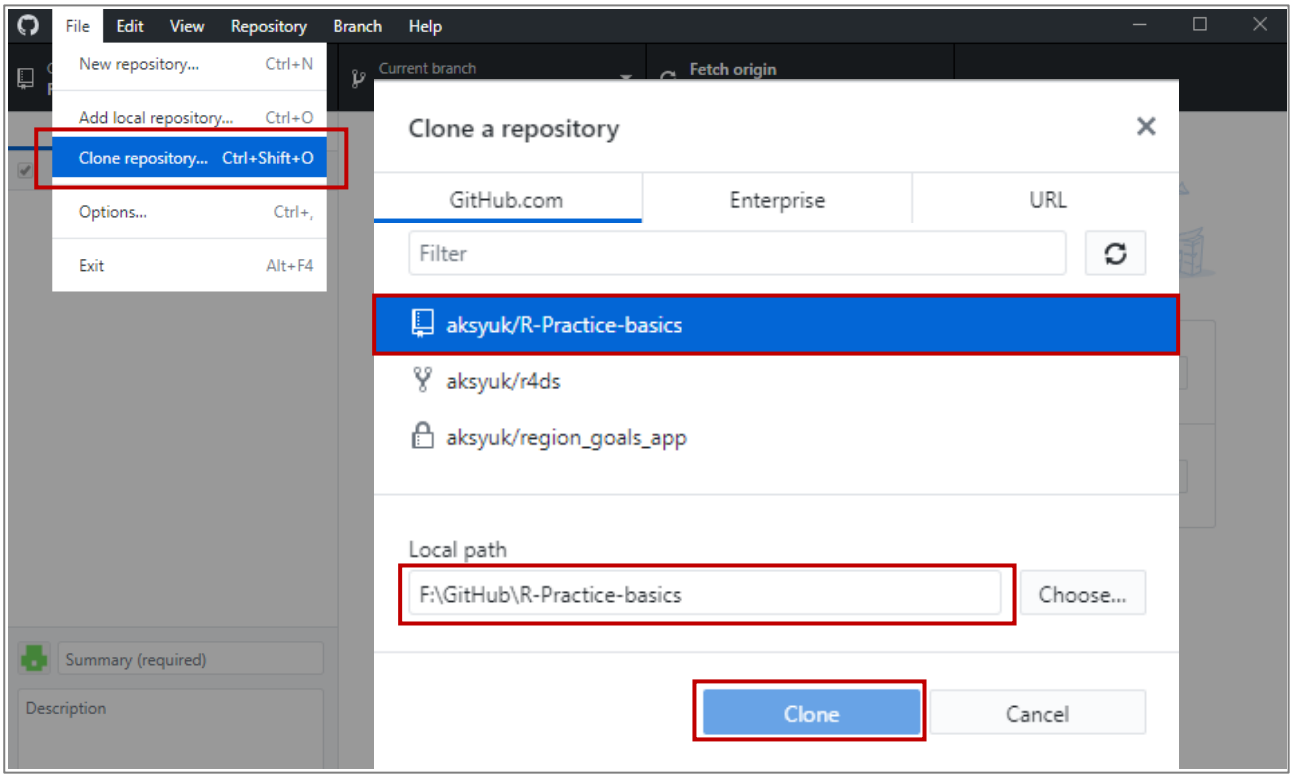

Рис. 19. Клонирование репозитория в Github Desktop

Любые изменения по сравнению с версией на сервере отображаются в окне репозитория (рисунок 20). В левой части окна появляется список файлов, в которые были внесены изменения. Сняв галочку с файла, его можно исключить из коммита на сервер, т.е. не изменять его в удалённом репозитории. Изменения в текстовых файлах (скрипты .R, .Rmd, .md и другие) можно видеть в основной части окна. Добавленные строки выделены зелёным фоном, удалённые – красным.

Обновление удалённого репозитория после внесения локальных изменений происходит по схеме: фиксация изменений в коммите → Push-запрос. Вид окна репозитория после коммита показа на рисунке 21. Кнопка «Push origin» отправит изменения на сервер, кнопка «Undo» в левом нижнем углу отменит зафиксированные в коммите изменения.

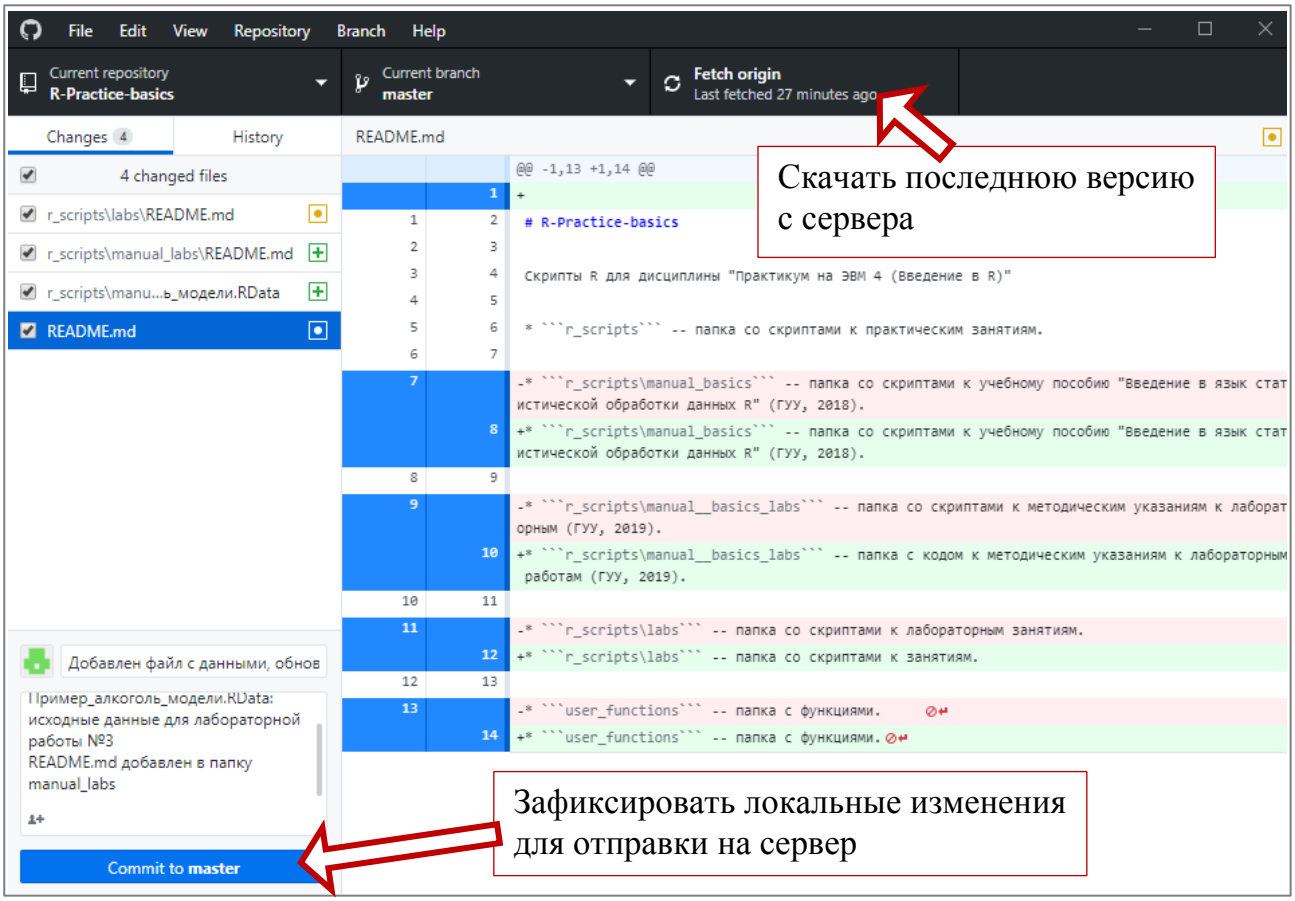

Рис. 20. Коммит локальных изменений

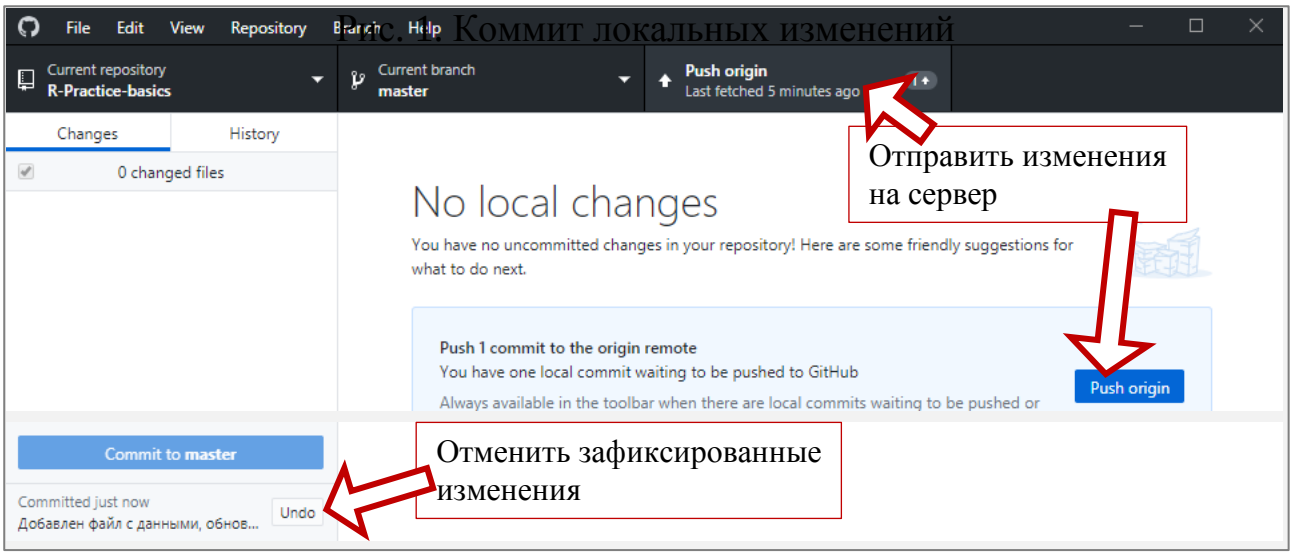

Рис. 21. Отправка локальных изменений на сервер (push-запрос)

Не все локальные файлы стоит отправлять на сервер. Например, в публикации не нуждаются отчёты в формате html, doc и pdf, так как они не отображаются в репозитории и обновляются интерактивно, по запуску сгенерировавших их скриптов. Чтобы не снимать галочки с файлов в каждом коммите, можно прописать, какие файлы нужно игнорировать, в файле «.gitignore» в директории репозитория. Этот файл является скрытым, и чтобы его увидеть, в Windows придётся сначала включить отображение скрытых файлов, как показано на рисунке 22.

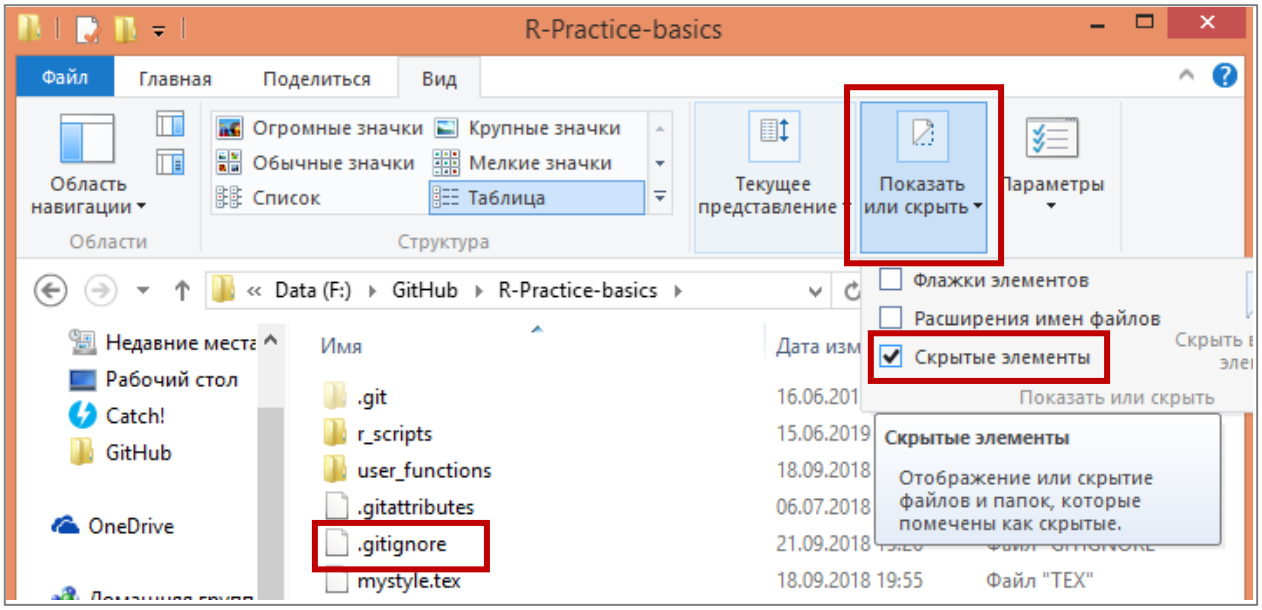

Рис. 22. Включаем отображение скрытых файлов в Windows 10, чтобы увидеть «.gitignore»[19]

[19] Как это сделать в других версиях Windows, описано в справке Microsoft: https://support.microsoft.com/ru-ru/help/14201/windows-show-hidden-files

Теперь «.gitignore» можно отредактировать в любом текстовом редакторе. Используем для этого RStudio и создадим правила игнорирования. Синтаксис файла должен отвечать следующим требованиям:

- каждая строка определяет одно правило;
- пустые строки игнорируются;
- комментарии начинаются с символа решётки в начале строки;
- символ «/» в начале строки указывает, что правило применяется только к файлам и папкам, которые располагаются в той же папке, что и сам файл .gitignore;
- символ звёздочки (*) заменяет любое количество символов (ноль или больше);
- знак вопроса (?) заменяет от нуля до одного символа;
- две звёздочки (**) используются для указания любого количества поддиректорий;
- восклицательный знак (!) в начале строки означает инвертирование правила, необходим для указания исключений из правил игнорирования;
- символ «\» используется для экранирования спецсимволов, например, чтобы игнорировать файл с именем "!readme!.txt", нужно написать правило: "\!readme!.txt";
- для игнорирования директории правило должно оканчиваться на слэш («/»), в противном случае правило считается именем файла[20].

Ниже приводится код файла «.gitignore», где эти правила перечислены с комментариями.

```
# AUTHOR:    github.com/aksyuk
# FILE:      .gitignore
#-----------------------------------------------------------------
# список файлов, добавление / изменение которых нужно
# игнорировать

# исходники ======================================================
*.com
*.class
*.dll
*.exe
*.o
*.so
```

[20] Michael Sokolov Правила синтаксиса файла .gitignore. URL: https://support.rdb24.com/hc/ru/articles/115000463769-Правила-синтаксиса-файла-gitignore

```
# архивы ================================================================
*.7z
*.dmg
*.gz
*.iso
*.jar
*.rar
*.tar
*.zip

# логи и базы данных ====================================================
*.log
*.sql
*.sqlite
*.Rhistory

# файлы, сгенерированные системой =======================================
*.DS_Store
*.DS_Store?
*._*
*.Spotlight-V100
*.Trashes
ehthumbs.db
Thumbs.db

# сгенерированные отчёты ================================================
*.html
*.htm
*.pdf
*.doc
*.docx
```

Содержимое «.gitignore» редактируется с учётом особенностей конкретного репозитория. К примеру, нужно игнорировать «сырые» данные, которые лучше вынести в отдельную директорию. В большинстве случаев стоит игнорировать графики, которые также удобно размещать в своей папке.

5.4 Подробнее о Git

Интерактивный тур по работе с Git на русском языке можно найти по адресу: https://githowto.com/ru. Более подробный учебник по Git с теорией и также на русском языке можно найти здесь: https://git-scm.com/book/ru/v1. Для освоения курса глав «Введение» и «Основы Git» будет достаточно.

6. ЗАДАНИЯ НА ЛАБОРАТОРНЫЕ РАБОТЫ

Задание на лабораторную работу №1

Цель работы: провести предварительный анализ данных в соответствии со своим вариантом и написать скрипт (файл .Rmd), который генерирует короткий отчёт. Номер варианта – номер студента в списке группы.

Исходные данные и задания на описательный анализ данных: варианты исходных данных и задания по описательному анализу размещены в репозитории на github.com: https://github.com/aksyuk/R-Practice-basics/blob/master/RScripts/manual_labs/TASK_lab_01.md.

Задание на лабораторную работу №2

Цель работы: построить линейные регрессионные модели, в т.ч. с переменной структурой, в соответствии со своим вариантом и написать скрипт (файл .Rmd), который генерирует короткий отчёт. Номер варианта – номер студента в списке группы.

Исходные данные и задания на моделирование: варианты исходных данных и задания на построение моделей размещены в репозитории на github.com: https://github.com/aksyuk/R-Practice-basics/blob/master/RScripts/manual_labs/TASK_lab_02.md.

Задание на лабораторную работу №3

Цель работы: протестировать остатки моделей, построенных во второй лабораторной, в соответствии со своим вариантом и написать скрипт (файл .Rmd), который генерирует короткий отчёт. Номер варианта – номер студента в списке группы.

Исходные данные и задания на тестирование остатков: варианты исходных данных и задания на тестирование моделей размещены в репозитории на github.com: https://github.com/aksyuk/R-Practice-basics/blob/master/RScripts/manual_labs/TASK_lab_03.md.

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

Основная литература

1. R Core Team (**2015**). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/
2. *Суязова С.А.* Введение в язык статистической обработки данных R: учебное пособие для подготовки бакалавров по направлению 01.03.02 Прикладная математика и информатика / Государственный университет управления, Институт информационных систем, кафедра математических методов в экономике и управлении. – М.: ГУУ, **2018**. – 67 с.
3. *Х.Уикем*, *Г.Гроулмунд* Язык R в задачах науки о данных: импорт, подготовка, обработка, визуализация и моделирование данных.: Пер. с англ. – Спб.: ООО "Альфа-книга", **2018**. – 592 с.
4. Markdown / Википедия – свободная энциклопедия. URL: https://ru.wikipedia.org/wiki/Markdown
5. Markdown синтаксис по-русски / Сайт http://rukeba.com/. URL: http://rukeba.com/by-the-way/markdown-sintaksis-po-russki/
6. Сайт проекта Markdown. URL: http://daringfireball.net/projects/markdown/
7. Программирование в стиле R (руководство от Google), перевод на русский язык: https://sites.google.com/a/kiber-guu.ru/r-practice/links/R_style_from_Google.pdf?attredirects=0&d=1
8. *Damodar N. Gujarati* Basic Econometrics. The McGraw-Hill Companies, **2004**.

Дополнительная литература

9. *А.Б. Шипунов, Е.М. Балдин, П.А. Волкова, А.И. Коробейников, С.А. Назарова, С.В. Петров, В.Г. Суфиянов* Наглядная статистика. Используем R! – М.: ДМК пресс, 2012. – 298 с.
10. *Мастицкий С.Э.*, *Шитиков В.К.* Статистический анализ в визуализация данных с помощью R. – М.: ДМК Пресс, 2015. – 496 с.: цв. ил.
11. *Роберт И. Кабаков* R в действии. Анализ и визуализация данных на языке R. – М.: ДМК Пресс, 2014. – 588 с.
12. *Patrick Burns* The R Inferno, version 30th April 2011. – URL: http://www.burns-stat.com/pages/Tutor/R_inferno.pdf, 2011. – 125 с.
13. *W. J. Owen* The R Guide version 2.5, Department of Mathematics and Computer Science University of Richmond, 2010 – 57 с.

СОДЕРЖАНИЕ

Введение ... 3
1. Создание отчёта с помощью пакета «knitr» в RStudio ... 6
1.1 Встроенный пример отчёта «knitr» ... 6
1.2 Коротко о языке Markdown ... 9
2. Указания к Лабораторной работе №1: Предварительный анализ данных ... 12
2.1 Импорт данных ... 12
2.2 Расчёт описательных статистик ... 15
2.3 Анализ распределения показателей ... 17
2.4 Анализ взаимосвязей показателей ... 20
2.5 Сохранение рабочего пространства ... 22
3. Указания к Лабораторной работе №2: Регрессионные модели ... 24
3.1 Импорт данных ... 25
3.2 Оценка параметров модели линейной регрессии ... 25
3.3 Сравнение нескольких моделей по качеству ... 30
3.4 Сохранение рабочего пространства ... 32
4. Указания к Лабораторной работе №3: Тесты остатков ... 33
4.1 Импорт данных ... 33
4.2 Графики остатков и поиск влияющих наблюдений ... 34
4.3 Тесты на равенство среднего остатков нулю ... 38
4.4 Тесты на гетероскедастичность ... 38
4.5 Тесты на автокорреляцию ... 41
4.6 Переоценка параметров с учётом ошибок спецификации ... 43
4.7 Тест на мультиколлинеарность факторов модели ... 44
5. Публикация отчёта ... 45
5.1 Публикация отчёта на RPubs ... 45
5.2 Размещение кода в репозитории GitHub ... 47
5.3 Обзор GitHub Desktop ... 51
5.4 Подробнее о Git ... 55
6. Задания на лабораторные работы ... 56
Список рекомендуемой литературы ... 57

Учебно-методическое издание

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к выполнению лабораторных работ по учебной дисциплине
«ПРАКТИКУМ НА ЭВМ 4»

СУЯЗОВА Светлана Андреевна

Материал издается в авторской редакции с представленного оригинал-макета.
Ответственность за сведения, представленные в издании, несет составитель.

Подготовка макета к изданию *Ю.Р. Шамильева*
Дизайн обложки *Е.А. Чеканова*
Тематический план внутривузовских изданий ГУУ 2019 г.

Подп. в печ. 19.08.2019. Формат 60x90/16. Объем 3,75 п.л.
Бумага офисная. Печать цифровая. Гарнитура Times New Roman.
Уч.-изд. л. 2,07. Изд. № 254/2019. Тираж 30 экз. Заказ № 714.

ФГБОУ ВО «Государственный университет управления»
Издательский дом ФГБОУ ВО ГУУ
109542, Москва, Рязанский проспект, 99, учебный корпус, ауд. 106
e-mail: id@guu.ru, roguu115@gmail.com
www.id.guu.ru, www.guu.ru