# Заголовок 1

Составитель: С.А.  Аксюк, [s.aksuk@kiber-guu.ru](mailto:s.aksuk@kiber-guu.ru)

*Ключевые слова: R, project R, проект R [1]*

Версия 1.0.3 (2018)

Все примеры выполнены R версии 3.4.3 (2017-11-30), «Kite-Eating Tree».

Версия RStudio: 1.1.423.

Все ссылки действительны на 12 февраля 2018 г.

Файл с макетом кода для этой практики: [lab\_2-1\_before.R](https://sites.google.com/a/kiber-guu.ru/r-practice/dpv/lab_2-1_before.R?attredirects=0&d=1)

## Заголовок 2

Одним из преимуществ R как языка обработки данных является воспроизводимость результатов. Технически, если код описывает всё, что происходит на разных стадиях исследования, от их загрузки до генерации отчёта, то любой пользователь может выполнить его на своём компьютере и получить такой же отчёт. Разумеется, для этого необходимо выполнение ряда условий, в частности, инструкции должны быть универсальными, а источники данных и пакеты, использованные для их обработки, – открытыми.

Рассмотрим несколько вариантов загрузки данных из открытых источников. При этом рекомендуем придерживаться нескольких простых правил, которые помогают добиться разумного уровня воспроизводимости:

1. *Не задавайте явно рабочую директорию*. Пока код не является самодостаточным пакетом или отдельным приложением, мы считаем, что он адресован пользователям, знакомым с азами R. Поэтому рабочую директорию пользователь задаст без нашего участия, и она, вероятно, не совпадёт с нашей. Однако,
2. *Сохраняйте данные в отдельную директорию внутри рабочей*. Это поможет отделить «сырые» данные и сохранить их на случай, если не будет возможности перезагрузить файл. Далее во всех примерах данные загружаются в папку «data» внутри рабочей директории.
3. *Сохраняйте время и дату загрузки*. Это облегчает контроль версий и просто даёт представление о том, как давно всё произошло. Далее в примерах эта информация сохраняется в текстовом файле внутри директории «data».
4. *Снабжайте данные описанием*. Обычно после предварительной обработки, которая может включать переименование столбцов, заполнение пропусков, изменение самого макета таблицы, файл данных отличается от исходного. В этом случае хорошим тоном будет составить короткий справочник с описанием проделанных трансформаций и с итоговым списком переменных (столбцов), с обязательным указанием их единиц измерения. В англоязычных источниках такой справочник носит название «code book», дословно – кодовая книга. Его назначение в том, чтобы составленная вами таблица данных не превращалась для стороннего человека в шифровку. Подобными кодовыми книгами, или справочниками, как они будут называться ниже, снабжены все встроенные в R наборы данных. Чтобы убедиться в этом, достаточно вызвать справку по файлу данных, например: «?mtcars» или «?iris».

### Заголовок 3

**Пример №.** Первый и самый простой способ получить данные – загрузить их в виде файла с известного адреса. Загрузим таблицу со статистикой импорта сливочного масла в РФ за 2010 – 2016 гг. Источник данных – база UN COMTRADE (<http://comtrade.un.org/data/>). Данные сохранены в репозитории на github.com и доступны по ссылке: <https://raw.githubusercontent.com/aksyuk/R-data/master/COMTRADE/040510-Imp-RF-comtrade.csv>.

Создадим директорию «data» внутри рабочей директории с помощью функции «dir.create()» и файл для записи лога загрузок с помощью функции «file.create()». Чтобы не перезаписывать их при повторных прогонах кода, добавим проверку условия. При загрузке и чтении данных полезны следующие функции R:

* file.exists('*путь\_к\_файлу*') возвращает TRUE, если указанный файл существует, и FALSE в противном случае.
* exists('*имя\_объекта*') возвращает TRUE, если указанный объект существует в рабочем пространстве R, и FALSE в противном случае.

Проверка условия существования файла (объекта) перед загрузкой (чтением) существенно экономит время при работе с таблицами большой размерности.

# создаём директорию для данных, если она ещё не существует:

if (!file.exists('./data')) {

dir.create('./data')

}

# создаём файл с логом загрузок, если он ещё не существует:

if (!file.exists('./data/download.log')) {

file.create('./data/download.log')

}

На этапе непосредственной загрузки файла используем функции:

* download.file(*URL\_файла, имя\_файла\_для\_сохранения*) загружает файл и сохраняет под указанным именем. Второй аргумент может быть именем файла, тогда он будет сохранён в рабочую директорию, а также абсолютным или относительным путём к файлу. Мы используем относительную ссылку на директорию с данными: «./data/имя\_файла», где точка означает «в текущей (рабочей) директории».
* write(*текст\_для\_записи,* file = *имя\_файла,* append = TRUE) записывает текст в указанный файл. Аргумент append = T означает, что новая строка будет добавлена в конец файла.

# адрес файла

fileURL <- 'https://raw.githubusercontent.com/aksyuk/R-data/master/COMTRADE/040510-Imp-RF-comtrade.csv'

# загружаем файл, если он ещё не существует,

# и делаем запись о загрузке в лог:

if (!file.exists('./data/040510-Imp-RF-comtrade.csv')) {

download.file(fileURL,

'./data/040510-Imp-RF-comtrade.csv')

# сделать запись в лог

write(paste('Файл "040510-Imp-RF-comtrade.csv" загружен',

Sys.time()),

file = './data/download.log', append = T)

}

Наконец, чтение данных из загруженного файла во фрейм и просмотр содержимого. Помните, что в случае если таблица содержит текстовые переменные, R автоматически сделает их факторами, присвоив каждому уникальному текстовому значению порядковый номер. Если требуется прочесть заранее неизвестную таблицу, нужно запретить такое преобразование, указав в функции read.csv() аргумент as.is = TRUE. Буквально это будет означать: читать текстовые столбцы «как есть».

# читаем данные из загруженного .csv во фрейм,

# если он ещё не существует

if (!exists('DF.import')){

DF.import <- read.csv('./data/040510-Imp-RF-comtrade.csv', as.is = T)

}# предварительный просмотр

dim(DF.import) # размерность таблицы

str(DF.import) # структура (характеристики столбцов)

head(DF.import) # первые несколько строк таблицы

tail(DF.import) # последние несколько строк таблицы

Это простой пример, поскольку загруженная таблица уже очищена от пустых столбцов и приведена к аккуратному виду, то есть:

* каждая строка содержит одно наблюдение;
* каждому столбцу соответствует одна переменная;
* каждый тип наблюдений (объектов) хранится в отдельной таблице.

Это список требований к тому, что принято называть «tidy data», или дословно – «аккуратные данные». Термин предложен Хэдли Уикхэмом в одноимённой статье в 2014 году[[1]](#footnote-1). В дополнение к этому, по заголовкам столбцов этого файла можно понять, что за переменные в нём содержатся. Кроме того, для данных уже написан справочник: <https://github.com/aksyuk/R-data/blob/master/COMTRADE/CodeBook_040510-Imp-RF-comtrade.md>.

Подчеркнём, что данные примера №1 уже подверглись предварительной обработке. Мы вернёмся к этому примеру в разделе «Очистка и предобработка данных», чтобы подробно поговорить об этом.

## Публикация результатов

### Сервис Github

На разных этапах работы с R мы будем использовать различные методы публикации полученных результатов. В данном практическом занятии рассмотрим Github (<https://github.com/>). 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.

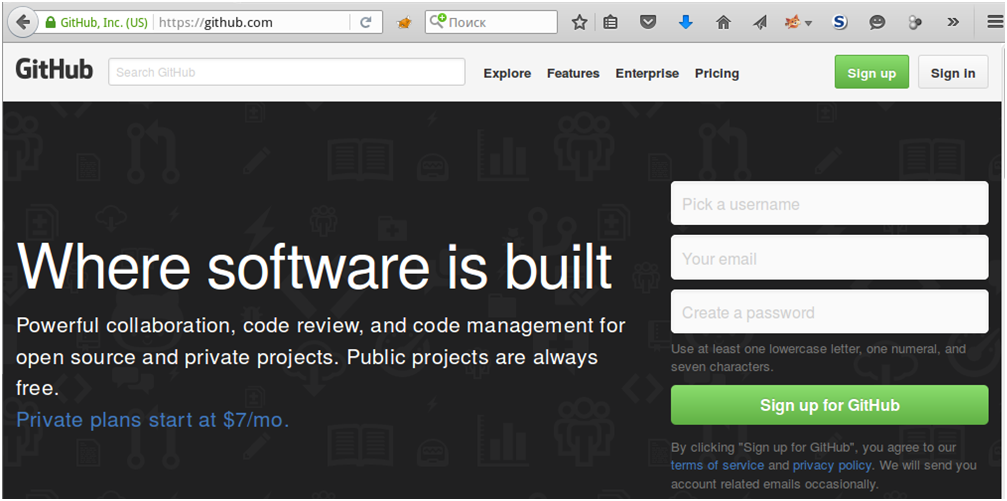
Когда изменения, сделанные локально, нужно закрепить на сервере, осуществляется *коммит* (команда «commit») – закрепление изменений и их подготовка к отправке на сервер. После чего изменения отправляются на сервер командой «push».

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 (Рис. 6) и войти под своим аккаунтом.

1. **Сделать свою ветвь чужого репозитория**. Исходный репозиторий находится по ссылке: <https://github.com/aksyuk/web-timeline>. На Рис. 7 показано, как сделать ветвь (другими словами, скопировать его) в свой аккаунт.



Зарегистрироваться

Войти

Рис. . Название объекта

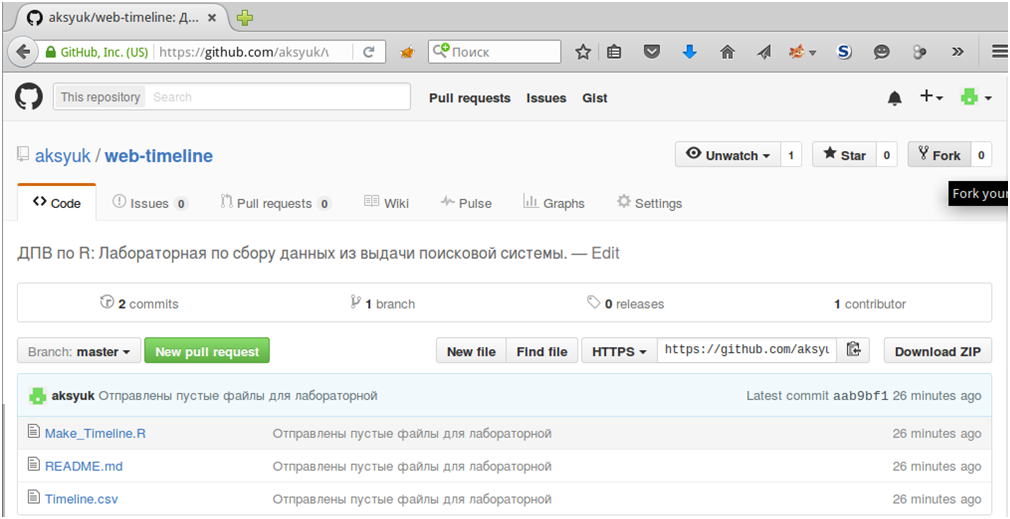


Рис. . Создать ветвь репозитория

1. **Обновить файлы своего репозитория через веб-интерфейс**. Для выполнения упражнения №1 из исходных файлов репозитория вам понадобится только «README.md». Скопировать его содержимое, чтобы отредактировать локально, можно, открыв исходник, или «Raw» (Рис. 8 – Рис. 10). Аналогично можно вставить в файл код или данные из файла на локальном компьютере.

## Упражнения

### Упражнение 1

В онлайн комиксе «xkcd[[2]](#footnote-2)» есть выпуск, который называется «Future Timeline», или «Хронология будущего»: <https://xkcd.com/887/>. Автор комикса проанализировал первые страницы выдачи поисковой системы Google по запросам вида «к <*году*>», «в течение <*года*>», «в <*году*>» и т.д., где год менялся от 2012 до 2101. Результаты – заголовки статей – собраны в единую временную линию, из которой можно судить об ожиданиях человечества (или, по крайней мере, англоязычной его части) относительного грядущего.

Необходимо реализовать тот же принцип, используя запросы, которые интересны Вам. Требования к выполнению и результатам:

1. Горизонт, за который просматриваются новости, должен быть не менее 10 лет. Распространяться он может как в будущее, так и в прошлое.
2. Количество формулировок запросов для каждого года – не менее 3. Например: «к 2018 импорт\*», «импорт\* на 2018», «импорт\* за 2018». Не забывайте, что поисковая система имеет свой синтаксис запросов, с помощью которого можно искать не только слово, но и его формы, либо исключать из выдачи статьи с определёнными словами. Большое количество запросов может привести к тому, что поисковая система потребует ввести капчу. Поэтому стоит встраивать таймер. Подробнее смотрите справку конкретной поисковой системы.
3. Поисковая система для выполнения упражнения выбирается в соответствии с вариантом. Номер варианта – номер в списке группы.
4. Результаты всех запросов за все годы нужно сохранить в файл «**Timeline.csv**» (при записи используйте аргумент row.names = FALSE) с тремя столбцами:

* Year – год, по которому сделан запрос.
* Header – заголовок статьи.
* Source – источник новости (по аналогии с примером 4, это адрес информационного источника, который виден на странице выдачи поисковика).
* URL – полная ссылка на источник (т.е. куда ведёт ссылка на заголовке статьи).

1. Код R, с помощью которого было выполнено задание, сохраните в файле «**Make\_Timeline.R**» с кодировкой UTF-8. Код должен быть оформлен в соответствии с рекомендациями Google [29] и снабжён комментариями.
2. Файлы «**Timeline.csv**» и «**Make\_Timeline.R**» должны быть размещены в репозитории на Github, который является ветвью репозитория <https://github.com/aksyuk/web-timeline>. В файле «README.md» вашего репозитория должна быть коротко описана постановка задания, структура запросов, а также временной горизонт. Ссылку на свой репозиторий вместе с ФИО студента высылайте на почту: [s.aksuk@kiber-guu.ru](file:///Z:\home\light\Dropbox\R-Дисциплины\03_ДПВ%20Аналитический%20пакет%20R\s.aksuk@kiber-guu.ru).

Поисковые системы для выполнения упражнения по вариантам:

1. [www.yandex.ru](http://www.yandex.ru)
2. [www.google.ru](file:///Z:\home\light\Dropbox\R-Дисциплины\03_ДПВ%20Аналитический%20пакет%20R\www.google.ru)
3. [www.nigma.ru](http://www.nigma.ru)
4. [search.yahoo.com](https://search.yahoo.com/)
5. <http://www.bing.com>
6. [http://www.rambler.ru](http://www.rambler.ru/)
7. [www.yandex.ru](http://www.yandex.ru)
8. [www.google.ru](file:///Z:\home\light\Dropbox\%5bR%5d\www.google.ru)
9. [www.nigma.ru](http://www.nigma.ru)
10. [search.yahoo.com](https://search.yahoo.com/)
11. <http://www.bing.com>
12. [http://www.rambler.ru](http://www.rambler.ru/)
13. [www.yandex.ru](http://www.yandex.ru)
14. [www.google.ru](file:///Z:\home\light\Dropbox\%5bR%5d\www.google.ru)
15. [www.nigma.ru](http://www.nigma.ru)
16. [search.yahoo.com](https://search.yahoo.com/)
17. <http://www.bing.com>
18. [http://www.rambler.ru](http://www.rambler.ru/)

Срок выполнения упражнения – две недели. Критерии оценки:

1. Наличие ссылки на репозиторий Github и наличие изменений в репозитории студента по сравнению с исходным.
2. Соответствие файла «**Timeline.csv**» требованиям к заданию.
3. Соответствие кода «**Make\_Timeline.R**» рекомендациям к оформлению.
4. Воспроизводим ли код из «**Make\_Timeline.R**», то есть охватывает ли он полностью все операции от загрузки данных до записи результатов.
5. Выдаёт ли код из «**Make\_Timeline.R**» результат, записанный «**Timeline.csv**».

## Источники

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. Jeffrey Leek. Материалы курса «Getting and Cleaning Data» Университета Джонса Хопкинса на портале coursera.org, доступные в репозитории на github.com: <https://github.com/jtleek/modules/tree/master/03_GettingData>.
3. XPath Syntax. URL: <http://www.w3schools.com/xsl/xpath_syntax.asp>.
4. Duncan Temple Lang and the CRAN Team (2015). XML: Tools for Parsing and Generating XML Within R and S-Plus. R package version 3.98-1.3. <https://CRAN.R-project.org/package=XML>.
5. Jeff Gentry (2015). twitteR: R Based Twitter Client. R package version 1.1.9. <https://CRAN.R-project.org/package=twitteR>.
6. Pablo Barbera and Michael Piccirilli (2015). Rfacebook: Access to Facebook API via R. R package version 0.6. <https://CRAN.R-project.org/package=Rfacebook>.
7. Raymond McTaggart, Gergely Daroczi and Clement Leung (2015). Quandl: API Wrapper for Quandl.com. R package version 2.7.0. <https://CRAN.R-project.org/package=Quandl>.
8. Jeffrey A. Ryan (2015). quantmod: Quantitative Financial Modelling Framework. R package version 0.4-5. <https://CRAN.R-project.org/package=quantmod>.
9. Введение в JSON. URL: <http://www.json.org/json-ru.html>.
10. Alex Couture-Beil (2014). rjson: JSON for R. R package version 0.2.15. <https://CRAN.R-project.org/package=rjson>.
11. Adrian A. Dragulescu (2014). xlsx: Read, write, format Excel 2007 and Excel 97/2000/XP/2003 files. R package version 0.5.7. <https://CRAN.R-project.org/package=xlsx>.
12. Gerrit-Jan Schutten (2014). readODS: Read ODS files and puts them into data frames. R package version 1.4. <https://CRAN.R-project.org/package=readODS>
13. R Core Team (2015). foreign: Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, Weka, dBase, .... R package version 0.8-66. <https://CRAN.R-project.org/package=foreign>.
14. Jeroen Ooms, David James, Saikat DebRoy, Hadley Wickham and Jeffrey Horner (2015). RMySQL: Database Interface and 'MySQL' Driver for R. R package version 0.10.7. <https://CRAN.R-project.org/package=RMySQL>
15. Brian Ripley and Michael Lapsley (2015). RODBC: ODBC Database Access. R package version 1.3-12. <https://CRAN.R-project.org/package=RODBC>.
16. Tommy Chheng (2013). RMongo: MongoDB Client for R. R package version 0.0.25. <https://CRAN.R-project.org/package=RMongo>.
17. Ben Stabler (2013). shapefiles: Read and Write ESRI Shapefiles. R package version 0.7. <https://CRAN.R-project.org/package=shapefiles>.
18. Roger Bivand, Tim Keitt and Barry Rowlingson (2015). rgdal: Bindings for the Geospatial Data Abstraction Library. R package version 1.1-3. <https://CRAN.R-project.org/package=rgdal>.
19. Roger Bivand and Colin Rundel (2015). rgeos: Interface to Geometry Engine - Open Source (GEOS). R package version 0.3-15. <https://CRAN.R-project.org/package=rgeos>.
20. Robert J. Hijmans (2015). raster: Geographic Data Analysis and Modeling. R package version 2.5-2. <https://CRAN.R-project.org/package=raster>.
21. Simon Urbanek (2014). jpeg: Read and write JPEG images. R package version 0.1-8. <https://CRAN.R-project.org/package=jpeg>.
22. Simon Urbanek (2013). png: Read and write PNG images. R package version 0.1-7. <https://CRAN.R-project.org/package=png>.
23. Gregory Jefferis (2014). readbitmap: Simple Unified Interface to Read Bitmap Images (BMP,JPEG,PNG). R package version 0.1-4. <https://CRAN.R-project.org/package=readbitmap>.
24. Uwe Ligges, Sebastian Krey, Olaf Mersmann, and Sarah Schnackenberg (2013). tuneR: Analysis of music. URL: <http://r-forge.r-project.org/projects/tuner/>.
25. Sueur J., Aubin T., Simonis C. (2008). Seewave: a free modular tool for sound analysis and synthesis. Bioacoustics, 18: 213-226. URL: <https://CRAN.R-project.org/package=seewave>.
26. M Dowle, A Srinivasan, T Short, S Lianoglou with contributions from R Saporta and E Antonyan (2015). data.table: Extension of Data.frame. R package version 1.9.6. <https://CRAN.R-project.org/package=data.table>.
27. Hadley Wickham and Romain Francois (2015). dplyr: A Grammar of Data Manipulation. R package version 0.4.3. <https://CRAN.R-project.org/package=dplyr>.
28. M Dowle, A Srinivasan, T Short, S Lianoglou with contributions from R Saporta, E Antonyan. Package 'data.table' Reference Manual, September 19, 2015. URL: <https://cran.r-project.org/web/packages/data.table/data.table.pdf>.
29. Markdown / Википедия – свободная энциклопедия. URL: <https://ru.wikipedia.org/wiki/Markdown>
30. Markdown синтаксис по-русски / Сайт <http://rukeba.com/>. URL: <http://rukeba.com/by-the-way/markdown-sintaksis-po-russki/>
31. Сайт проекта Markdown. URL: <http://daringfireball.net/projects/markdown/>
32. Руководство по стилю программирования на R от Google. URL: <https://sites.google.com/a/kiber-guu.ru/r-practice/links/R_style_from_Google.pdf?attredirects=0&d=1>

1. Hadley Wickham. Tidy data. The Journal of Statistical Software, vol. 59, 2014. [↑](#footnote-ref-1)
2. xkcd: A webcomic of romance, sarcasm, math, and language. <https://xkcd.com>. [↑](#footnote-ref-2)