Аналитический пакет R. Загрузка и предобработка данных

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

12 фев, 2019

Ключевые слова: R, r-project, RStudio

Версия 2.0.0 (2019)

Примеры выполнены R версии 3.5.1, «Feather Spray».

Версия RStudio: 1.1.453.

Все ссылки действительны на 12 февраля 2019 г.

Файл с макетом кода для этой практики: [lab\_2-1\_before.R]

## Несколько полезных привычек

     В любой работе есть важные вещи, которые быстро становятся рутиной. В аналитике это работа с данными на начальном этапе, ещё до того, как нам представится случай проявить своё творческое начало в визуализации переменных и в интерпретации того, что мы видим. Часто загрузка данных – это рутина, но если к этому этапу отнестить неаккуратно, в ходе анализа можно создать себе неприятные проблемы. Поэтому специалисту по работе с данными лучше сразу формировать у себя полезные привычки, которые, к слову, пригодятся не только в R.

     Одно из преимуществ R как языка обработки данных – 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. В этой практике мы рассмотрим несколько вариантов загрузки данных из открытых источников. При этом будем придерживаться нескольких простых правил, которые помогают избежать многих проблем.

*Не задавайте явно рабочую директорию*. Пока код не является самодостаточным пакетом или отдельным приложением, мы считаем, что он адресован пользователям, знакомым с азами R. Рабочую директорию адресат задаёт без нашего участия, и её имя и расположение, очевидно, не совпадут с нашими.

     Однако, *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.

*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».

*Снабжайте данные описанием*. Обычно после предварительной обработки, которая может включать переименование столбцов, заполнение пропусков, изменение 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.

     Наработав собственный опыт, читатель, разумеется, прибавит к этому минимальному списку свои правила, которые, при должной практике, превратятся в полезные привычки. Главная цель здесь, с одной стороны, настроить рабочее окружение под себя, а с другой – в случае совместной работы над проектом свести необходимость дополнительных разъяснений к минимуму.

## Загрузка файла .csv из сети

**Пример №1**. 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 2010-2018 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 = TRUE 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 stringsAsFactors = 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',   
 stringsAsFactors = 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. 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.

## Парсинг XML

     Значительная часть открытых данных в интернете содержится не в файлах .xls или .csv, а в виде таблиц на веб-страницах. Анализ текста веб-страницы с целью извлечь нужную информацию, ориентируясь по тегам разметки, носит название парсинга (от англ. parse – разбор, структурный анализ). Легче всего понять технологию парсинга веб-страниц в R на примере разбора XML, хотя на практике сайты, написанные на чистом XML, в настоящее время очень редки.

**Пример №2**. Рассмотрим учебную XML-страницу: <http://www.w3schools.com/xml/simple.xml> [[2]](#footnote-2). На Рис. 1 показана структура этого файла.

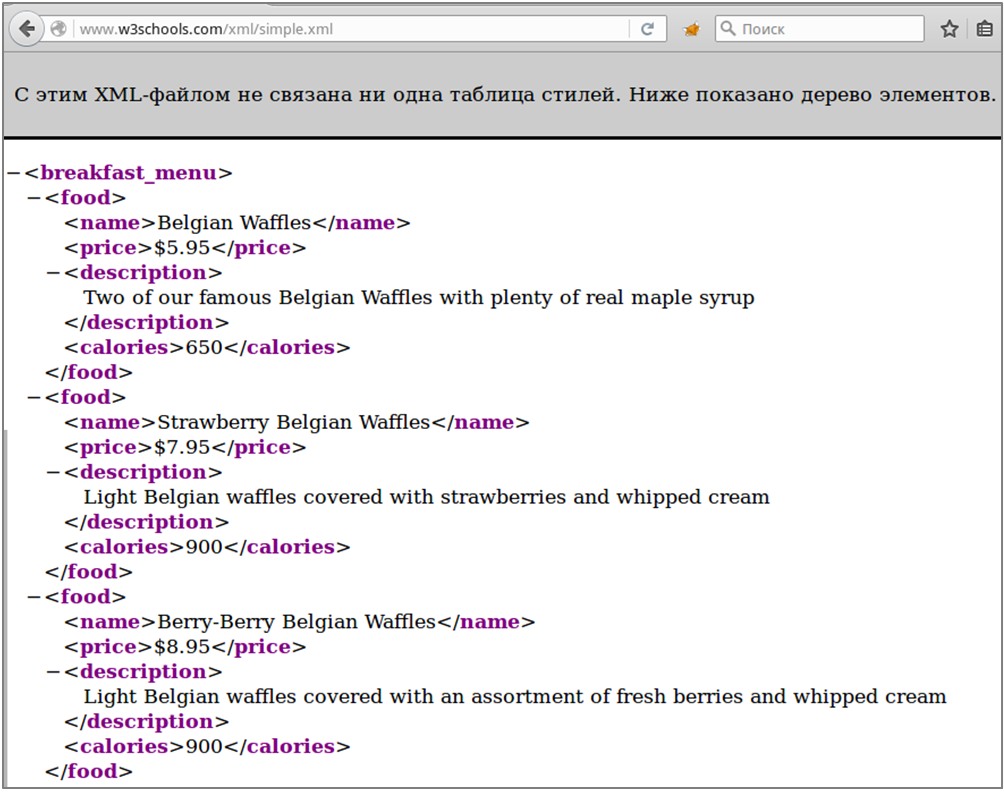


Рис.1. Структура XML-файла из примера 2

Для разбора XML-страниц в R служит пакет «XML».

# Загрузка пакетов  
library('RCurl') # работа с URL  
#> Warning: package 'RCurl' was built under R version 3.5.2  
#> Loading required package: bitops  
library('XML') # разбор XML-файлов  
#> Warning: package 'XML' was built under R version 3.5.2  
  
# адрес XML-страницы  
fileURL <- 'https://www.w3schools.com/xml/simple.xml'  
  
# загружаем содержимое в объект doc  
doc <- getURL(fileURL)  
  
# разбираем объект как XML  
doc <- xmlTreeParse(doc, useInternalNodes = T)

1. Hadley Wickham. Tidy data. The Journal of Statistical Software, vol. 59, 2014. [↑](#footnote-ref-1)
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>. [↑](#footnote-ref-2)