1 Чтение данных из электронных таблиц Excel

В огромном мире анализа данных Excel — ваш верный помощник, упрощающий процесс вычислений, организации и представления информации. Благодаря интуитивно понятному интерфейсу и широкому распространению Excel получил статус основного инструмента в мире бизнеса. Однако объем данных и их сложность непрерывно растут, из-за чего возможности Excel становятся все более ограниченными. Решить эту проблему призвано сближение миров Excel, R и Python. В книге вы увидите, насколько эффективна синергия этих языков программирования и Excel, расширите возможности этой программы и научитесь легко решать задачи, связанные с данными. Мы подробно обсудим способы интеграции Excel с R и Python, которые позволяют раскрыть потенциал этой программы, извлекать ценные сведения, автоматизировать процессы и использовать все возможности анализа данных.

Пакет Microsoft Excel появился на рынке в 1985 году и до сих пор остается популярным программным обеспечением (ПО) для работы с электронными таблицами. Изначально приложение было известно под названием MultiPlan. Microsoft Excel и базы данных в целом имеют некоторые общие черты в плане организации и управления данными, хотя и служат разным целям. Excel — программа для работы с электронными таблицами, которая позволяет пользователям хранить данные в табличном формате и выполнять с ними различные действия. Таблица состоит из строк и столбцов, а каждая ее ячейка может содержать текст, числа или формулы. Аналогичным образом база данных представляет собой структурированную коллекцию данных, хранящихся в таблицах, состоящих из строк и столбцов.

И Excel, и базы данных позволяют хранить и извлекать данные. В Excel можно вводить данные, выполнять вычисления, создавать диаграммы и графики. Точно так же базы данных позволяют хранить большие объемы структурированных данных и управлять ими, выполняя запросы, сортировку и фильтрацию. Кроме того, Excel и базы данных поддерживают концепцию отношений. В Excel можно связывать ячейки или диапазоны ячеек на разных листах, устанавливая связи между

данными. Базы данных используют отношения для связывания таблиц на основе общих полей, что позволяет извлекать связанные данные из нескольких таблиц.

Цель этой главы — познакомить вас со способами чтения файлов Excel в среде R и выполнения некоторых операций над ними. В частности, мы рассмотрим следующие темы:

- обработка данных Excel с помощью пакетов R;
- чтение файлов Excel в среду R;
- чтение нескольких листов Excel с помощью readxl и пользовательской функции;
- пакеты Python для работы с Excel;
- открытие листа Excel из среды Python и чтение данных;
- чтение нескольких листов с помощью Python (openpyxl и пользовательских функций).

Технические требования

Во время написания книги мы использовали следующее программное обеспечение:

- R 4.2.1;
- версия RStudio 2023.03.1+446 Cherry Blossom для Windows.

Изучить материал этой главы вы сможете, установив следующие пакеты:

- `readxl`;
- `openxlsx`;
- `xlsx`.

Для выполнения кода Python, приведенного в этой главе, мы будем использовать:

- Python 3.11;
- `pandas`;
- `openpyxl`;
- Excel-файл `iris.xlsx`, доступный в репозитории GitHub для этой книги.

Описание процесса настройки среды Python выходит за рамки данной книги, но сделать это несложно. Необходимые пакеты можно установить, выполнив следующие команды:

```
python -m pip install pandas==2.0.1
python -m pip install openpyxl==3.1.2
```

Обратите внимание, что эти команды следует запускать из терминала, а не из сценария Python. Запустите их из папки, в которой находится файл `requirements.txt`, или укажите полный путь к нему.

В репозитории GitHub к этой книге также содержится файл `requirements.txt`, который вы можете использовать для установки всех зависимостей. Для этого выполните следующую команду:

```
python -m pip install -r requirements.txt
```

Она устанавливает все пакеты, которые будут использоваться в текущей главе, избавляя вас от необходимости устанавливать их по одному. Вдобавок она гарантирует, что дерево зависимостей будет полностью совпадать с тем, которое использовали мы, авторы данной книги.

В качестве альтернативы при использовании блокнотов Jupyter можно применить следующие команды:

```
%pip install pandas==2.0.1
%pip install openpyxl==3.1.2
```

Все примеры кода из книги представлены на GitHub по адресу https://github.com/PacktPublishing/Extending-Excel-with-Python-and-R. Примеры из каждой главы содержатся в соответствующей папке: примеры из текущей главы можно найти в папке Chapter1.

ПРИМЕЧАНИЕ

Технические требования к Python перечислены в файле requirements.txt, доступном в репозитории GitHub по адресу https://github.com/PacktPublishing/Extending-Excel-with-Python-and-R/blob/main/requirements.txt. Если вы установите эти зависимости, то вам будет проще писать код и легче читать книгу. Обязательно установите их все, прежде чем приступать к выполнению упражнений.

Обработка данных Excel с помощью пакетов R

На ресурсах CRAN и GitHub доступно несколько пакетов, позволяющих читать файлы Excel и выполнять с ними различные действия. В этом разделе мы рассмотрим пакеты `readxl`, `openxlsx` и `xlsx`. В них есть функции для чтения файлов Excel:

- `readxl::read_excel()`;
- `openxlsx::read.xlsx()`;
- `xlsx::read.xlsx()`.

Каждая функция предусматривает свой набор параметров и соглашений, которые необходимо соблюдать. Пакет `readxl` — часть коллекции пакетов `tidyverse`, поэтому он придерживается ее соглашений и после чтения файла возвращает объект `tibble` (тиббл) — современную версию `data.frame` языка R, которая представляет собой своего рода электронную таблицу в среде R. Данный объект служит строительным блоком при выполнении большинства видов анализа. Что касается пакетов `openxlsx` и `xlsx`, то они оба возвращают базовый объект R `data.frame`, причем `xlsx` может возвращать еще и объект `list`. Возможно, вам интересно, как это связано с обработкой реального файла Excel. Начнем вот с чего: чтобы получить возможность работать с данными в среде R, их необходимо сначала считать. В указанных выше пакетах есть различные функции для работы с файлами Excel, а также для чтения данных такими способами, которые позволяют проводить их дальнейший анализ. Имейте в виду, что пакет `xlsx` требует установки Java.

Обсудив пакеты R для работы с Excel, поговорим об эффективном способе считывания файлов Excel в среду R, который предоставляет еще больше возможностей для анализа данных и работы с ними.

Чтение файлов Excel в среду R

В этом разделе мы будем считывать данные из Excel с помощью различных библиотек R. Это необходимо сделать до выполнения каких-либо действий с данными, содержащимися в листах файлов Excel, или до их анализа.

Как уже было сказано в разделе «Технические требования» выше, для считывания данных в среду R мы будем использовать пакеты `readxl`, `openxlsx` и `xlsx`.

Установка и загрузка библиотек

Мы будем использовать библиотеки `openxlsx`, `xlsx`, `readxl` и `readxlsb`. Если вы еще не установили и не загрузили их, выполните следующий блок кода:

```
pkgs <- c("openxlsx", "xlsx", "readxl")
install.packages(pkgs, dependencies = TRUE)
lapply(pkgs, library, character.only = TRUE)
```

Функция `lapply()` в языке R — это универсальный инструмент для применения функции к каждому элементу списка, вектора данных или датафрейма (`DataFrame`). Она принимает два аргумента: `x` и `FUN`, где `x` — это список, а `FUN` — функция, которая применяется к объекту `x`.

Теперь, когда библиотеки установлены, можем приступить к работе. Для этого мы считаем электронную таблицу, созданную на основе встроенного в R набора

данных `Iris`. Мы прочитаем файл с помощью трех разных библиотек, а затем создадим пользовательскую функцию для работы с библиотекой `readxl`, которая будет считывать все листы файла Excel. Мы назовем эту функцию `read_excel_sheets()`.

Первой библиотекой, которой мы воспользуемся для открытия файла Excel, будет `openxlsx`. Чтобы прочитать нужный файл Excel, вы можете запустить код из папки `chapter1` репозитория GitHub под названием `ch1_create_iris_dataset.R`. Процесс считывания файла в среду R показан на рис. 1.1.

```
> # Use openxlsx
> openxlsx::read.xlsx(f_path) |> head(5)
  sepal_length sepal_width petal_length petal_width species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
> openxlsx::read.xlsx(f_path, sheet = "iris") |> head(5)
  sepal_length sepal_width petal_length petal_width species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
```

Рис. 1.1. Использование пакета openxlsx для чтения файла Excel

Обратите внимание на переменную `f_path`. Она содержит путь к месту сохранения набора данных Iris в виде файла Excel, например `C:/User/UserName/Documents/iris_data.xlsx`.

В этом примере предполагается, что для создания файла Excel вы использовали файл `ch1_create_iris_datase.R`. В действительности вы можете прочитать любой нужный вам файл Excel.

Теперь мы выполним ту же операцию, но уже с помощью библиотеки `xlsx`. На рис. 1.2 показан тот же метод чтения, что и в случае с пакетом `openxlsx`.

Наконец, воспользуемся библиотекой `readxl`, которая является частью коллекции пакетов `tidyverse` (рис. 1.3).

```
> # Use xlsx
> xlsx::read.xlsx(file = f_path, sheetIndex = 1) |> head(5)
  sepal_length sepal_width petal_length petal_width species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
> xlsx::read.xlsx(file = f_path, sheetName = "iris") |> head(5)
  sepal_length sepal_width petal_length petal_width species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
```

Рис. 1.2. Использование библиотеки xlsx и функции read.xlsx() для открытия созданного нами файла Excel

```
> # Use readxl
> readxl::read_excel(f_path) |> head(5)
# A tibble: 5 × 5
  sepal_length sepal_width petal_length petal_width species
         <dbl>       <dbl>        <dbl>       <dbl> <chr>
1          5.1         3.5          1.4         0.2 setosa
2          4.9         3            1.4         0.2 setosa
3          4.7         3.2          1.3         0.2 setosa
4          4.6         3.1          1.5         0.2 setosa
5          5           3.6          1.4         0.2 setosa
> readxl::read_excel(f_path, "iris") |> head(5)
# A tibble: 5 × 5
  sepal_length sepal_width petal_length petal_width species
         <dbl>       <dbl>        <dbl>       <dbl> <chr>
1          5.1         3.5          1.4         0.2 setosa
2          4.9         3            1.4         0.2 setosa
3          4.7         3.2          1.3         0.2 setosa
4          4.6         3.1          1.5         0.2 setosa
5          5           3.6          1.4         0.2 setosa
```

Рис. 1.3. Использование библиотеки readxl и функции read_excel() для считывания файла Excel в память

В этом разделе вы научились считывать файлы Excel в среду R с помощью нескольких различных пакетов. Эти пакеты позволяют выполнять множество других операций, однако в данном случае нас интересовало именно чтение файлов. Итак, вы знаете, как использовать функции `readxl::read_excel()`, `xlsx::read.xlsx()` и `openxlsx::read.xlsx()`.

Теперь мы можем двигаться дальше и поговорить об эффективном извлечении данных из нескольких листов файла Excel.

Чтение нескольких листов с помощью readxl и пользовательской функции

При работе в Excel мы часто имеем дело с рабочими книгами, включающими несколько листов. Они могут содержать представленные в определенном формате статистические данные за разные месяцы или какие-то другие периоды. По тем или иным причинам мы можем захотеть прочитать все листы, содержащиеся в файле, не вызывая при этом функцию чтения для каждого отдельного листа. Можно использовать R, чтобы выполнить эту операцию в цикле с помощью пакета `purrr`.

Создадим пользовательскую функцию. Для этого загрузим функцию `readxl`, если она еще не загружена. Однако если библиотека установлена, но вы не хотите загружать ее в память, можете вызвать функцию `excel_sheets()` с помощью `readxl::excel_sheets()` (рис. 1.4).

Этот новый код можно разделить так:

```
read_excel_sheets <- function(filename, single_tbl) {
```

Эта строка определяет функцию `read_excel_sheets`, принимающую два аргумента: `filename` (имя файла Excel, который нужно прочитать) и `single_tbl` (логическое значение, указывающее на то, что именно должна возвращать функция: одну таблицу или список таблиц).

Далее следует такая строка:

```
sheets <- readxl::excel_sheets(filename)
```

Здесь пакет `readxl` используется для извлечения имен всех листов из файла Excel с именем, определяемым параметром `filename`. Имена листов сохраняются в переменной `sheets`.

```
read_excel_sheets <- function(filename, single_tbl = FALSE) {
  sheets <- readxl::excel_sheets(filename)

  if (single_tbl){
    x <- purrr::map_df(sheets, readxl::read_excel, path = filename)
  } else {
    x <- purrr::map(sheets, ~ readxl::read_excel(filename, sheet = .x))
    purrr::set_names(x, sheets)
  }

  x
}
```

Рис. 1.4. Создание функции read_excel_sheets() для одновременного считывания всех листов из файла Excel

Следующая строка выглядит так:

```
if (single_tbl) {
```

Она запускает инструкцию `if`, которая проверяет значение аргумента `single_tbl`.

Далее следует строка:

```
x <- purrr::map_df(sheets, read_excel, path = filename)
```

Если значением `single_tbl` является `TRUE`, то вызывается функция `map_df` пакета `purrr` для перебора имен листов, хранящихся в переменной `sheets`, и чтения соответствующих листов с помощью функции `read_excel` из пакета `readxl`. Полученные объекты `DataFrame` объединяются в таблицу, которая присваивается переменной `x`.

Следующая строка выглядит так:

```
} else {
```

Это начало блока `else` инструкции `if`. Если значением `single_tbl` является `FALSE`, то выполняется код, содержащийся в этом блоке.

Затем идет такая строка:

```
x <- purrr::map(sheets, ~ readxl::read_excel(filename, sheet = .x))
```

Здесь функция `map` пакета `purrr` используется для перебора имен листов, хранящихся в переменной `sheets`. Для чтения каждого листа файла Excel, имя которого определяется параметром `filename`, вызывается функция `read_excel` из пакета `readxl`. Полученные объекты `DataFrame` сохраняются в списке, который присваивается переменной `x`.

Далее следует строка:

```
purrr::set_names(x, sheets)
```

В ней используется функция `set_names` из пакета `purrr` для присвоения элементам списка `x` имен листов, хранящихся в переменной `sheets`.

Предпоследняя строка выглядит так:

```
x
```

Она возвращает из функции значение `x`, которое будет представлять собой либо одну таблицу (`data.frame`), если значением `single_tbl` является `TRUE`, либо список таблиц (`data.frame`), если значением `single_tbl` является `FALSE`.

Итак, функция `read_excel_sheets` принимает имя файла Excel и логическое значение, указывающее на то, что именно необходимо возвратить: одну таблицу или их список. Функция использует пакет `readxl` для извлечения имен листов из файла Excel, а затем считывает соответствующие листы либо в одну таблицу (если значением `single_tbl` является `TRUE`), либо в их список (если значением `single_tbl` является `FALSE`). Полученные данные возвращаются в качестве результата работы функции. Чтобы увидеть, как работает эта схема, рассмотрим следующий пример.

У нас есть электронная таблица с четырьмя вкладками — по одной для каждого вида ирисов из набора данных Iris, и еще один лист под названием `iris`, который содержит весь набор данных.

Как показано на рис. 1.5, функция `read_excel_sheets()` прочитала все четыре листа файла Excel. Кроме того, мы видим, что эта функция импортировала листы в виде объекта-списка и присвоила каждому его элементу имя соответствующей вкладки в файле Excel. Важно отметить, что для корректной работы данной функции все листы должны иметь одинаковые имена столбцов и структуру.

В этом разделе мы показали процесс написания функции, которая может прочитать все листы в любом файле Excel и возвратить их в виде списка элементов, имена которых соответствуют названиям вкладок в исходном файле.

Теперь, когда вы научились считывать листы Excel в среду R, посмотрим, как делать то же самое с помощью языка Python.

```
> read_excel_sheets(f, T)
# A tibble: 300 × 5
   sepal_length sepal_width petal_length petal_width species
          <dbl>       <dbl>        <dbl>       <dbl> <chr>
 1          5.1         3.5          1.4         0.2 setosa
 2          4.9         3            1.4         0.2 setosa

> read_excel_sheets(f, F)
[[1]]
# A tibble: 50 × 5
   sepal_length sepal_width petal_length petal_width species
          <dbl>       <dbl>        <dbl>       <dbl> <chr>
 1          5.1         3.5          1.4         0.2 setosa
 2          4.9         3            1.4         0.2 setosa

[[2]]
# A tibble: 50 × 5
   sepal_length sepal_width petal_length petal_width species
          <dbl>       <dbl>        <dbl>       <dbl> <chr>
 1          7           3.2          4.7         1.4 versicolor
 2          6.4         3.2          4.5         1.5 versicolor

[[3]]
# A tibble: 50 × 5
   sepal_length sepal_width petal_length petal_width species
          <dbl>       <dbl>        <dbl>       <dbl> <chr>
 1          6.3         3.3          6           2.5 virginica
 2          5.8         2.7          5.1         1.9 virginica

[[4]]
# A tibble: 150 × 5
   sepal_length sepal_width petal_length petal_width species
          <dbl>       <dbl>        <dbl>       <dbl> <chr>
 1          5.1         3.5          1.4         0.2 setosa
 2          4.9         3            1.4         0.2 setosa
```

Рис. 1.5. Файл Excel, прочитанный с помощью функции read_excel_sheets()

Пакеты Python для работы с Excel

Один из ключевых аспектов работы с файлами Excel в Python — наличие подходящего набора пакетов, обеспечивающих необходимую функциональность. В этом разделе мы обсудим пакеты Python, наиболее часто используемые при работе с Excel, и расскажем об их преимуществах и особенностях.