Оглавление

Об авторе

Тиаго Антао – биоинформатик, который в настоящее время работает в области геномики. Бывший ученый, работавший в области информатики, Тиаго перешел в вычислительную биологию, получив степень магистра биоинформатики на факультете естественных наук Университета Порту, Португалия, и докторскую степень за работу по теме «Распространение лекарственно-устойчивой малярии» в Ливерпульской школе тропической медицины, Великобритания. После защиты докторской диссертации Тиаго работал с наборами данных по человеку в Кембриджском университете, Великобритания, и с данными полногеномного секвенирования комаров в Оксфордском университете, Великобритания, а затем помог создать инфраструктуру биоинформатики в Университете Монтаны, США. В настоящее время он работает дата-инженером в области биотехнологии в Бостоне, штат Массачусетс. Он является одним из соавторов Biopython, основного программного пакета биоинформатики, написанного на Python.

О рецензентах

Урминдер Сингх – биоинформатик, ученый в области информатики и разработчик нескольких инструментов биоинформатики с открытым исходным кодом. Его образование включает степени в области физики, информатики и вычислительной биологии, в том числе степень доктора наук по биоинформатике Университета штата Айова, США.

Его разнообразные исследовательские интересы включают эволюцию новых генов, точную медицину, социогеномику, машинное обучение в медицине и разработку инструментов и алгоритмов для больших гетерогенных данных. Вы можете посетить его онлайн-страницу по адресу urmi-21.github.io.

Тиффани Хо работает сотрудником по биоинформатике в Embark Veterinary. Она имеет степень бакалавра Калифорнийского университета в Дэвисе в области генетики и геномики, а также степень магистра в области селекции растений и генетики Корнельского университета.

Предисловие

Биоинформатика – это активная область исследований, в которой используется ряд простых и сложных вычислений для извлечения ценной информации из биологических данных, и эта книга покажет вам, как управлять этими задачами с помощью Python.

Это обновленное издание книги рецептов по биоинформатике с Python начинается с краткого обзора различных инструментов и библиотек в программном окружении Python, которые помогут вам преобразовывать, анализировать и визуализировать наборы биологических данных. По мере продвижения по главам вы познакомитесь с важнейшими методами секвенирования следующего поколения, анализа отдельных клеток, геномики, метагеномики, популяционной генетики, филогенетики и протеомики на основе реальных примеров. Вы узнаете, как работать с лучшими конвейерными системами, такими как серверы Galaxy и Snakemake, освоите различные модули Python для функционального и асинхронного программирования. В числе прочего эта книга поможет вам изучить такие темы, как обнаружение SNP с использованием статистических подходов в высокопроизводительных вычислительных средах, включая Dask и Spark, и применение алгоритмов машинного обучения в биоинформатике.

К концу этой книги по биоинформатике Python вы приобретете знания для реализации новейших методов и сред программирования, что позволит вам работать с данными биоинформатики на любом уровне.

Для кого эта книга

Эта книга предназначена для аналитиков в области биоинформатики, специалистов по данным, вычислительных биологов, исследователей, разработчиков Python, которые хотят решать проблемы биологии и биоинформатики от среднего до высокого уровня. Приветствуется практическое знание языка программирования Python. Также не помешали бы базовые знания по биологии.

Какие темы охватывает эта книга

В главе 1 «*Python и окружающее программное обеспечение*» рассказывается, как создать современную биоинформационную среду с помощью Python. В этой главе обсуждается, как развертывать программное обеспечение с помощью Docker, взаимодействовать с R и с Jupyter Notebooks.

В главе 2 «*Знакомство с NumPy, pandas, Arrow и Matplotlib*» представлены основные библиотеки Python для обработки данных: NumPy для обработки массивов и матриц; Pandas для работы с табличными данными; Arrow для оптимизации обработки Pandas и Matplotlib для построения диаграмм.

В главе 3 «*Секвенирование следующего поколения*» представлены конкретные решения для работы с данными секвенирования на уровне следующего поколения. В этой главе вы узнаете, как работать с большими файлами FASTQ, BAM и VCF. Также обсуждается фильтрация данных.

В главе 4 «*Продвинутый процессинг данных NGS*» рассматриваются расширенные методы программирования для фильтрации данных NGS. Они включают в себя использование менделевских наборов данных, которые затем анализируются стандартной статистикой. Также мы вводим метагеномный анализ.

В главе 5 «*Работа с геномами*» рассматриваются не только высококачественные образцы, такие как геном человека, но также обсуждается, как анализировать другие, низкокачественные образцы, характерные для немодельных видов. Эта глава знакомит вас с обработкой GFF, учит анализировать информацию о геномных особенностях и обсуждает, как использовать генные онтологии.

В главе 6 «*Популяционная генетика*» описывается, как проводить популяционно-генетический анализ эмпирических наборов данных. Например, в Python мы можем выполнять анализ основных компонентов (PCA), компьютерный индекс фиксации (FST) или диаграммы примесных структур.

В главе 7 «*Филогенетика*» используются полные последовательности недавно секвенированных вирусов Эбола для проведения полноценного филогенетического анализа, который включает в себя реконструкцию филогенетического дерева и сравнение последовательностей. В этой главе обсуждаются рекурсивные алгоритмы для обработки древовидных структур.

Глава 8 «*Использование Protein Data Bank*» посвящена обработке файлов банка данных о белках (PDB), например выполнению геометрического анализа структуры белков. В этой главе рассматривается визуализация структуры белков.

В главе 9 «*Конвейеры биоинформатики*» представлены два типа конвейеров обработки. Первый тип конвейера – это Galaxy на основе Python, широко используемая система с веб-интерфейсом, ориентированная в основном на пользователей, не занимающихся программированием, хотя биоинформатикам, возможно, все же придется взаимодействовать с ней программно. Второй тип основан на snakemake и nextflow, типе конвейера, предназначенном для программистов.

Глава 10 «*Машинное обучение в биоинформатике*» знакомит с машинным обучением, использующим интуитивный подход к решению задач вычислительной биологии. В этой главе рассматриваются анализ основных компонентов, кластеризация, деревья принятия решений (Decision Trees) и случайные леса (Random Forests).

Глава 11 «*Параллельная обработка с помощью Dask и Zarr*» познакомит вас с методами работы с очень большими наборами данных и алгоритмами, требующими больших вычислительных ресурсов. В этой главе объясняется, как использовать параллельные вычисления на многих компьютерах (в кластере или облаке). Мы также обсудим эффективное хранение биологических данных.

Глава 12 «*Функциональное программирование для биоинформатики*» познакомит вас с функциональным программированием. Оно позволяет разрабатывать более сложные программы на Python, которые благодаря «ленивому программированию» и немутабельности легче развертывать в параллельных средах со сложными алгоритмами.

Чтобы получить максимальную отдачу от этой книги

Программное и аппаратное обеспечение, описанное в книге	Требования к ОС
Python 3.9	Windows, Mac OS X и Linux (предпочтительно)
Numpy, Pandas, Matplolib	
Biopython	
Dask, zarr, scikit-learn	

Если вы используете цифровую версию этой книги, мы советуем вам ввести код самостоятельно или получить доступ к коду через репозиторий GitHub (ссылка доступна в следующем разделе). Это поможет вам избежать возможных ошибок, связанных с копированием и вставкой кода.

Загрузите файлы примеров кода

Вы можете загрузить примеры файлов кода для этой книги с GitHub по адресу https://github.com/PacktPublishing/Bioinformatics-with-Python-Cookbook-Third-edition. В случае обновления кода он будет обновлен в существующем репозитории GitHub.

У нас также есть другие пакеты кода из нашего богатого каталога книг и видео, доступных по адресу https://github.com/PacktPublishing/. Ознакомьтесь с ними!

Загрузите цветные изображения

Мы также предоставляем PDF-файл с цветными изображениями скриншотов и диаграмм, использованных в этой книге. Скачать его можно здесь: https://packt.link/3KQQO.

Используемые соглашения

В этой книге используется ряд текстовых соглашений.

`Код в тексте`: указывает кодовые слова в тексте, имена таблиц базы данных, имена папок, имена файлов, расширения файлов, пути, фиктивные URL-адреса, пользовательский ввод и дескрипторы Twitter. Вот пример: «`call_genotype` имеет вид 56,241×1,1198,2, то есть размерные варианты, образцы, `ploidy`».

Блок кода устанавливается следующим образом:

```
from Bio import SeqIO
genome_name = 'PlasmoDB-9.3_Pfalciparum3D7_Genome.fasta'
recs = SeqIO.parse(genome_name, 'fasta')
for rec in recs:
    print(rec.description)
```

Когда мы хотим привлечь ваше внимание к определенной части блока кода, соответствующие строки или элементы выделяются жирным шрифтом:

```
AgamP4_2L | organism=Anopheles_gambiae_PEST | version=AgamP4 | length=49364325
| SO=chromosome
AgamP4_2R | organism=Anopheles_gambiae_PEST | version=AgamP4 | length=61545105
| SO=chromosome
```

Жирный: отмечает новый термин, важное слово или слова, которые вы видите на экране. Например, слова в меню или диалоговых окнах отображаются в тексте следующим образом. Вот пример: «По столбцу **Chunk** см. главу 11, но пока вы можете спокойно это игнорировать».

> **Советы или важные примечания**
>
> Представлены так.

Разделы

В этой книге вы найдете несколько часто встречающихся заголовков (*Подготовка, Как это сделать..., Как это работает..., Дополнительно и См. также*).

Чтобы получить четкие инструкции по завершению рецепта, используйте эти разделы следующим образом:

Подготовка

В этом разделе рассказывается, чего ожидать от рецепта, и описывается, как установить нужное программное обеспечение или сделать предварительные настройки, необходимые для выполнения рецепта.

Как это сделать...

Этот раздел содержит шаги, необходимые для выполнения рецепта.

Как это работает...

Этот раздел обычно состоит из подробного объяснения того, что произошло в предыдущем разделе.

Дополнительно...

Этот раздел содержит дополнительную информацию о рецепте, чтобы вы могли больше о нем узнать.

Смотрите также

Этот раздел содержит полезные ссылки на другую полезную информацию по рецепту.

Глава 1

Python и окружающее программное обеспечение

Мы начнем с установки основного программного обеспечения, необходимого для большей части этой книги. Оно включает в себя дистрибутив **Python**, некоторые фундаментальные библиотеки Python и внешнее программное обеспечение для биоинформатики. Мы также рассмотрим мир за пределами Python. В биоинформатике и больших данных важную роль играет **R**; поэтому вы узнаете, как взаимодействовать с ним через **rpy2**, который представляет собой мост от Python к R. Кроме того, мы изучим преимущества, которые может дать нам фреймворк **IPython** (через Jupyter Lab) для эффективного взаимодействия с R. Учитывая, что управление исходным кодом с помощью **Git** и **GitHub** широко распространено, мы позаботимся о том, чтобы наш сетап хорошо работал с ними. Эта глава даст вам основу для всех процессов вычислительной биологии, которые мы будем выполнять в оставшейся части книги.

Поскольку у разных пользователей разные требования, мы рассмотрим два разных подхода к установке программного обеспечения. Один подход заключается в использовании дистрибутива **Anaconda** Python (http://docs.continuum.io/anaconda/), а другой – в установке программного обеспечения через **Docker** (это метод виртуализации сервера, основанный на контейнерах, использующих одно и то же ядро операционной системы; пожалуйста, обратитесь к https://www.docker.com/). Это также установит вам Anaconda, но уже внутри контейнера. Если вы используете операционную систему Windows, мы настоятельно рекомендуем вам рассмотреть возможность поменять операционную систему или использовать Docker с помощью некоторых существующих опций в Windows. В macOS вы можете установить большую часть программного обеспечения непосредственно, хотя Docker также доступен. Обучение с использованием локального дистрибутива (Anaconda или чего-то другого) проще, чем с Docker, но, учитывая, что управление пакетами в Python может быть сложным, образы Docker обеспечивают определенный уровень стабильности.

В этой главе мы рассмотрим следующие рецепты:

- установка необходимого программного обеспечения с Anaconda;
- установка необходимого программного обеспечения с Docker;
- взаимодействие с R через rpy2;
- выполнение R magic с Jupyter.

Установка необходимого базового программного обеспечения с помощью Anaconda

Прежде чем мы начнем, нам нужно установить необходимое базовое программное обеспечение. Следующие разделы познакомят вас с программным обеспечением и шагами, необходимыми для его установки. Каждая глава и раздел могут предъявить дополнительные требования – мы укажем их, когда достигнем соответствующего момента в изложении. Альтернативный способ начать работу – использовать рецепт Docker, после чего вас обеспечат всем нужным через контейнер Docker.

Если вы уже используете другой дистрибутив Python, мы настоятельно рекомендуем вам рассмотреть Anaconda, поскольку он стал стандартом де-факто для науки о данных и биоинформатики. Также именно этот дистрибутив позволит вам установить программное обеспечение от **Bioconda** (https://bioconda.github.io/).

Подготовка

Python можно запускать поверх разных сред. Например, вы можете использовать Python внутри виртуальной машины Java (Java Virtual Machine, **JVM**) (через **Python** или с .NET через **IronPython**). Однако здесь нас интересует не только Python, но и вся среда программного обеспечения вокруг него. Поэтому мы будем использовать стандартную реализацию **CPython**, так как версии JVM и .NET существуют в основном для взаимодействия с оригинальными библиотеками этих платформ.

Для создания наших кодов мы будем использовать Python 3.10. Если вы начинали с Python и биоинформатики, подойдет любая операционная система. Но здесь нас в основном интересует промежуточное и продвинутое применение. Таким образом, если вы используете Windows и macOS, имейте в виду, что большую часть сложного ресурсозатратного анализа придется выполнять в Linux (вероятно, на Linux high-performance computing (высокопроизводительных вычислениях Linux) или кластере **HPC**). Анализ данных секвенирования следующего поколения (Next-generation sequencing, **NGS**) и сложное машинное обучение в основном выполняются на кластерах Linux.

Если вы работаете в Windows, следует подумать об обновлении до Linux для вашей работы в области биоинформатики, поскольку большинство современных программ для биоинформатики не будут работать в Windows. Обратите внимание, что macOS подойдет почти для всех анализов, если только вы не планируете использовать компьютерный кластер, который, скорее всего, будет основан на Linux.

Если вы работаете в Windows или macOS и у вас нет простого доступа к Linux, не беспокойтесь. Вам на помощь придет современное программное обеспечение виртуализации (такое как **VirtualBox** и Docker), которое позволит установить виртуальную Linux на вашу операционную систему. Если вы работаете с Windows и решили, что хотите перейти на оригинальный язык и не использовать Anaconda, будьте осторожны с выбором библиотек; вы, вероятно, будете в большей безопасности, если установите 32-битную версию для всего (включая сам Python).

> **Примечание**
>
> Если вы работаете в Windows, многие инструменты будут вам недоступны.

Биоинформатика и наука о данных развиваются с головокружительной скоростью; это не реклама, это реальность. При установке программных библиотек выбор версии может причинить проблемы. В зависимости от кода, который у вас есть, он может не работать с некоторыми старыми версиями или, возможно, даже не работать с более новой версией. Будем надеяться, что любой код, который вы используете, покажет правильные соответствия, хотя это и не гарантируется. В этой книге мы установим точные версии всех программных пакетов и позаботимся о том, чтобы код работал с ними. Вполне естественно, что код может нуждаться в согласовании с другими версиями пакета.

Программное обеспечение, разработанное для этой книги, доступно по адресу https://github.com/PacktPublishing/Bioinformatics-with-Python-Cookbook-Third-edition. Чтобы получить к нему доступ, вам нужно будет установить Git. Опыт работы с ним имеет свою пользу, потому что с помощью Git разрабатывается множество программных пакетов для научных вычислений.

Прежде чем правильно установить стек Python, вам необходимо установить все внешнее программное обеспечение, которое требуется для Python, с которым вы будете взаимодействовать. Список будет варьироваться от главы к главе, и все программные пакеты будут объяснены в соответствующих главах. К счастью, со времени выхода предыдущих изданий этой книги большая часть программного обеспечения для биоинформатики стала доступна через проект Bioconda; поэтому инсталляция, как правило, не вызовет проблем.

Вам нужно будет установить некоторые компиляторы и библиотеки проектирования, все они бесплатны. В Ubuntu рассмотрите возможность установки пакета build-essential (`apt-get install build-essential`), а для macOS подумайте о **Xcode** (https://developer.apple.com/xcode/).

На рис. 1.1 вы найдете список наиболее важного программного обеспечения для разработки биоинформатики с помощью Python.

Мы будем использовать `pandas` для обработки большинства табличных данных. Альтернативой может быть использование только стандартного Python. Библиотека `pandas` стала настолько распространенной в науке о данных, что, вероятно, имеет смысл обрабатывать с ее помощью все табличные данные (если они помещаются в памяти).

Вся наша работа будет производиться внутри проекта Jupyter, а именно Jupyter Lab. Jupyter *де-факто* стал стандартом для написания скриптов интерактивного анализа данных. К сожалению, формат по умолчанию для Jupyter Notebooks основан на JSON. Этот формат трудно читать, трудно сравнивать, и его необходимо экспортировать, чтобы передать в обычный интерпретатор Python. Чтобы устранить эту проблему, расширим Jupyter с помощью `jupytext` (https://jupytext.readthedocs.io/), что позволит нам сохранять блокноты Jupyter как обычные программы Python.

Имя	Использование	URL-адрес	Цель
Project Jupyter	Все главы	https://jupyter.org/	Интерактивные вычисления
pandas	Все главы	https://pandas.pydata.org/	Обработка данных
NumPy	Все главы	http://www.numpy.org/	Обработка массивов/матриц
SciPy	Все главы	https://www.scipy.org/	Научные вычисления
Biopython	Все главы	https://biopython.org/	Библиотека биоинформатики
seaborn	Все главы	http://seaborn.pydata.org/	Библиотека статистических диаграмм
R	Биоинформатика и статистика	https://www.r-project.org/	Язык статистических вычислений
rpy2	R-подключение	https://rpy2.readthedocs.io	Интерфейс R
PyVCF	NGS	https://pyvcf.readthedocs.io	Обработка VCF
Pysam	NGS	https://github.com/pysam-developers/pysam	Обработка SAM/BAM
HTSeq	NGS/геномы	https://htseq.readthedocs.io	Обработка NGS
DendroPY	Филогенетика	https://dendropy.org/	Филогенетика
PyMol	Протеомика	https://pymol.org	Молекулярная визуализация
scikit-learn	Машинное обучение	http://scikit-learn.org	Библиотека машинного обучения
Cython	Большие данные	http://cython.org/	Высокая производительность
Numba	Большие данные	https://numba.pydata.org/	Высокая производительность
Dask	Большие данные	http://dask.pydata.org	Параллельная обработка

Рис. 1.1. Таблица, показывающая различные программные пакеты, полезные в биоинформатике

Как это сделать...

Для начала ознакомьтесь со следующими шагами:

1. Начните с загрузки дистрибутива Anaconda с https://www.anaconda. com/products/individual. Мы будем использовать версию 21.05, хотя вам, вероятно, подойдет и более поздняя, если она вам доступна. Вы можете принять все параметры инсталляции по умолчанию, но можете убедиться,

что двоичные файлы conda находятся в вашем пути (не забудьте открыть новое окно, чтобы можно было обновить путь). Если у вас есть другой дистрибутив Python, будьте осторожны с PYTHONPATH и существующими библиотеками Python. Вероятно, лучше отключить ваш PYTHONPATH. Насколько это возможно, удалите все другие версии Python и его библиотеки, установленные ранее.

2. Займемся библиотеками. Теперь мы создадим новую среду conda под названием bioinformatics_base с biopython=1.70, как показано в следующей команде:

```
conda create -n bioinformatics_base python=3.10
```

3. Давайте активируем среду следующим образом:

```
conda activate bioinformatics_base
```

4. Давайте добавим каналы bioconda и conda-forge в наш список источников:

```
conda config --add channels bioconda
conda config --add channels conda-forge
```

5. Также установим основные пакеты:

```
conda install \
biopython==1.79 \
jupyterlab==3.2.1 \
jupytext==1.13 \
matplotlib==3.4.3 \
numpy==1.21.3 \
pandas==1.3.4 \
scipy==1.7.1
```

6. Теперь сохраним нашу среду, чтобы мы могли использовать ее позже для создания новых сред на других машинах или если вам нужно будеть очистить базовую среду.

```
conda list –explicit > bioinformatics_base.txt
```

7. Можно даже установить R из conda:

```
conda install rpy2 r-essentials r-gridextra
```

Обратите внимание, что r-essentials устанавливает множество пакетов R, включая ggplot2, который мы будем использовать позже. Кроме того, мы устанавливаем r-gridextra, так как будем использовать его в Notebook.

Дополнительно...

Если вы предпочитаете не использовать Anaconda, вы сможете установить разные библиотеки Python через `pip`, используя любой выбранный вами дистрибутив. Вам, вероятно, понадобится довольно много компиляторов и инструментов сборки[1] – не только компиляторы C, но также C++ и Fortran.

Мы не будем использовать среду, созданную на предыдущих шагах. Вместо этого мы будем использовать ее в качестве базы для клонирования рабочих сред. Это связано с тем, что управление средой с помощью Python – даже с помощью системы пакетов `conda` – может причинить головную боль. Таким образом, мы создадим чистую среду, которую мы никогда не испортим и от которой мы сможем избавиться, если она станет неуправляемой.

Например, представьте, что вы хотите создать среду для машинного обучения с помощью `scikit-learn`. Можно сделать следующее.

1. Создайте клон исходной среды:

```
conda create -n scikit-learn --clone bioinformatics_base
```

2. Добавьте `scikit-learn`:

```
conda activate scikit-learn
conda install scikit-learn
```

Находясь внутри JupyterLab, мы должны открывать наши файлы jupytext с помощью блокнота Notebook, а не текстового редактора. Поскольку файлы jupytext имеют то же расширение, что и файлы Python – это особенность, а не ошибка, – по умолчанию JupyterLab будет использовать обычный текстовый редактор. Когда мы открываем файл jupytext, нам нужно переписать установки по умолчанию. Открыв его, щелкните правой кнопкой мыши и выберите **Notebook**, как показано на рис. 1.2.

Наши файлы jupytext не будут сохранять графические данные, и это допустимо для данной книги. Если вы хотите иметь версию с изображениями, это возможно с помощью парных блокнотов. Для получения дополнительной информации посетите страницу Jupytext (https://github.com/mwouts/jupytext).

> **Предупреждение**
>
> Поскольку наш код предназначен для запуска внутри Jupyter, много раз в этой книге я не буду использовать `print` для вывода содержимого, поскольку последняя строка ячейки будет отображаться автоматически. Если вы не используете блокноты, сделайте `print`.

[1] Сборка включает в себя компиляцию, компоновку и упаковку кода в пригодную для использования или исполняемую форму. – *Прим. ред.*

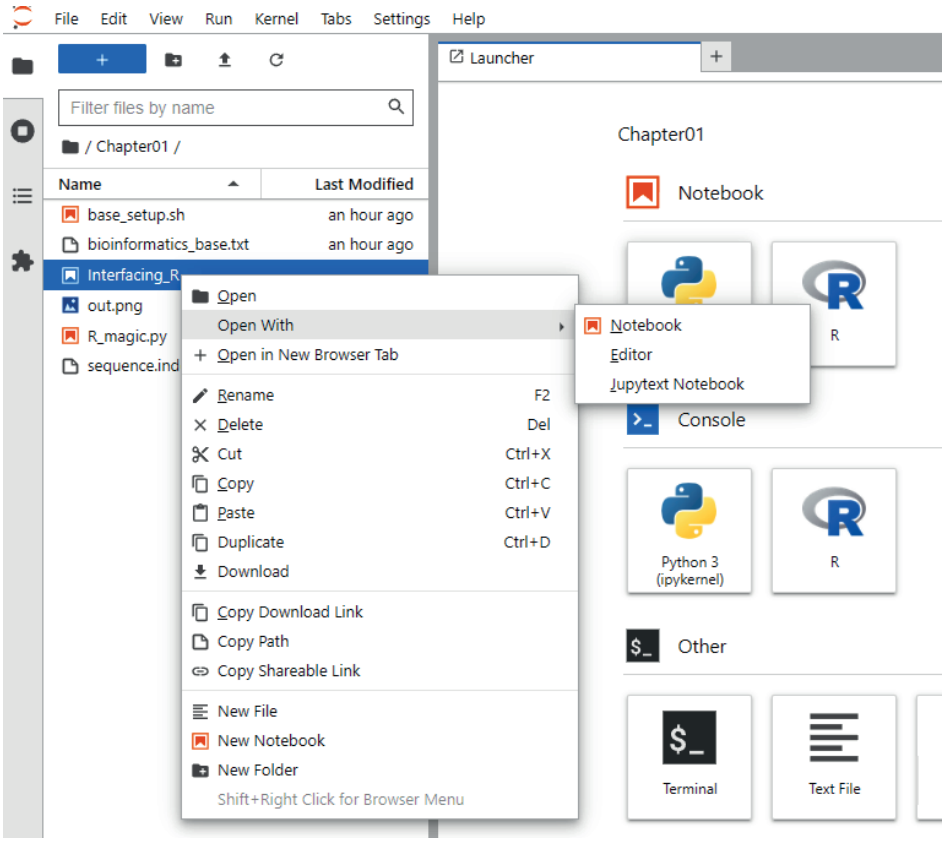

Рис. 1.2. Открытие файла jupytext в Notebook

Установка необходимого программного обеспечения с помощью Docker

Docker – наиболее широко используемый фреймворк для реализации виртуализации на уровне операционной системы. Эта технология позволяет вам иметь независимый контейнер: уровень, который легче, чем виртуальная машина, но все же позволяет компартментализировать программное обеспечение. Он практически изолирует все процессы, создавая ощущение, что каждый контейнер является виртуальной машиной.

Docker хорошо работает на обоих краях разработки: это удобный способ настроить содержание данной книги в учебных целях, и он может стать вашей предпочтительной платформой для развертывания приложений в сложных средах. Этот рецепт является альтернативой предыдущему рецепту.

Однако для сред проектирования, рассчитанных на долгую историю, лучше всего подойдет что-то вроде предыдущего рецепта, хотя это может повлечь за собой более трудоемкую первоначальную настройку.

Подготовка

Если вы используете Linux, первое, что вам нужно сделать, – это установить Docker. Самое безопасное решение – получить последнюю версию с https://www.docker.com/. Хотя в вашем дистрибутиве Linux может быть пакет Docker, он может быть слишком старым и содержать ошибки.

Если у вас Windows или macOS, не отчаивайтесь; давайте посмотрим на сайт Docker. Существуют различные варианты, чтобы спасти вас, но четкой формулы нет, поскольку Docker довольно быстро продвигается на этих платформах. Для запуска нашей 64-битной виртуальной машины необходим достаточно новый компьютер. Если у вас возникли проблемы, перезагрузите компьютер и убедитесь, что BIOS, VT-X или AMD-V доступны. Вам понадобится по крайней мере 6 ГБ оперативной памяти, а лучше больше.

> **Примечание**
>
> Для этого потребуется скачать очень большой объем данных из интернета, поэтому убедитесь, что у вас достаточная скорость соединения с интернетом. Также будьте готовы долго ждать.

Как это сделать...

Чтобы начать, выполните следующие действия.

1. Используйте такую команду в вашей оболочке Docker:

```
docker build -t bio https://raw.githubusercontent.com/PacktPublishing/
Bioinformatics-with-Python-Cookbook-third-edition/main/docker/main/
Dockerfile
```

   В Linux вам потребуются либо привилегии root, либо добавление в группу Docker Unix.

2. Теперь вы готовы запустить контейнер следующим образом:

```
docker run -ti -p 9875:9875 -v YOUR_DIRECTORY:/data bio
```

3. Замените директорию `YOUR_DIRECTORY` на каталог в вашей операционной системе. Он будет использоваться вашей операционной системой хоста совместно с контейнером Docker. Каталог `YOUR_DIRECTORY` будет виден в контейнере в `/data`, и наоборот.

   Установка `-p 9875:9875` выставит TCP-порт контейнера 9875 на порт хост-компьютера, `9875`.

   Убедитесь, что ваш каталог действительно виден в среде оболочки Docker, особенно в Windows (и, возможно, в macOS). Если нет, ознакомьтесь с официальной документацией Docker о том, как открывать каталоги.

4. Теперь вы готовы к использованию системы. Укажите в браузере http://localhost:9875, и вы должны получить среду Jupyter.

Если это не работает в Windows, ознакомьтесь с официальной документацией Docker (https://docs.docker.com/), чтобы узнать, как открывать порты.

Смотрите также

Также стоит знать следующее:

- Docker является наиболее широко используемым программным обеспечением для контейнеризации[1], и в последнее время его использование значительно возросло. Подробнее об этом можно прочитать на https://www.docker.com/;
- альтернативой Docker с точки зрения безопасности является **rkt**, который можно найти по адресу: https://coreos.com/rkt/;
- если вы не можете использовать Docker, например если у вас нет необходимых допусков, как это происходит на большинстве компьютерных кластеров, тогда давайте рассмотрим Singularity по адресу https://www.sylabs.io/singularity/.

Взаимодействие с R через rpy2

Если вам нужна какая-то определенная функциональность, но вы не можете найти ее в библиотеке Python, вам в первую очередь нужно проверить, реализована ли она в R. Для статистических методов R по-прежнему является наиболее полной структурой; кроме того, некоторые функции биоинформатики доступны только в R и, вероятно, предлагаются в виде пакета, принадлежащего проекту Bioconductor.

Пакет **rpy2** предоставляет декларативный интерфейс от Python к R. Как вы увидите, вы сможете написать очень элегантный код Python для выполнения процесса взаимодействия. Чтобы показать интерфейс (и опробовать одну из самых распространенных структур данных R, DataFrame, и одну из самых популярных библиотек R, `ggplot2`), мы загрузим его метаданные из проекта «Human 1000 Genomes Project» (http://www.1000genomes.org/). Это не книга по R, но мы хотим предоставить интересные и функциональные примеры.

Подготовка

Вам нужно будет получить файл метаданных из указателя последовательности «Проекта 1000 геномов». Пожалуйста, проверьте https://github.com/PacktPublishing/Bioinformatics-with-Python-Cookbook-Third-edition/blob/main/Datasets.py и загрузите файл `sequence.index`. Если вы используете Jupyter Notebook, откройте файл `Chapter01/Interfacing_R.py` и просто выполните команду `wget` сверху.

Этот файл содержит информацию обо всех файлах FASTQ в проекте (мы будем использовать данные из проекта «1000 геномов человека» в следующих главах). Он включает в себя файл FASTQ, идентификатор образца, исходную популяцию и важную статистическую информацию для каждой дорожки (lane)[2], такую как количество ридов (reads, чтений, прочтений) и количество прочитанных оснований ДНК.

---

[1] Метод виртуализации, при котором ядро операционной системы поддерживает несколько изолированных экземпляров пространства пользователя вместо одного. – *Прим. ред.*

[2] Lane – это физическая «дорожка» на проточной кювете, которую вводят в машину для секвенирования. Дорожки могут использоваться для разделения различных экспериментов. Вы можете секвенировать свой образец на нескольких дорожках, но в итоге все равно получите один файл. Или секвенировать несколько образцов на одной дорожке и затем разделить их. – *Прим. ред.*

Чтобы настроить Anaconda, вы можете запустить следующее:

```
conda create -n bioinformatics_r --clone bioinformatics_base
conda activate bioinformatics_r
conda install r-ggplot2=3.3.5 r-lazyeval r-gridextra rpy2
```

С помощью Docker вы можете запустить:

```
docker run -ti -p 9875:9875 -v YOUR_DIRECTORY:/data tiagoantao/bioinformatics_r
```

Теперь мы можем начать.

Как это сделать...

Чтобы начать, выполните следующие действия.

1. Давайте начнем с импорта:

```
import os
from IPython.display import Image
import rpy2.robjects as robjects
import rpy2.robjects.lib.ggplot2 as ggplot2
from rpy2.robjects.functions import SignatureTranslatedFunction
import pandas as pd
import rpy2.robjects as ro
from rpy2.robjects import pandas2ri
from rpy2.robjects import local_converter
```

   Мы будем использовать `pandas` на стороне Python. R DataFrames очень хорошо сопоставляется[1] с `pandas`.

2. Прочитаем данные из нашего файла, используя функцию R `read.delim`:

```
read_delim = robjects.r('read.delim')
seq_data = read_delim ('sequence.index', header=True, stringsAsFactors=False)
#In R:
# seq.data <- read.delim('sequence.index', header=TRUE,
stringsAsFactors=FALSE)
```

   Первое, что мы делаем после импорта, – это обращаемся к R-функции `read.delim`, которая позволяет читать файлы. Спецификация языка R позволяет ставить точки в именах объектов. Поэтому нам нужно преобразовать имя функции в `read_delim`. Затем мы называем функции правильным именем; обратите внимание на следующие признаки. Во-первых,

[1] Сопоставление (mapping) заключается в применении функции преобразования к итерируемому объекту для создания нового объекта. Элементы в новой итерации создаются путем вызова функции преобразования для каждого элемента в исходной итерации. – *Прим. ред.*

большинство атомарных объектов, таких как строки, можно передавать без преобразования. Во-вторых, имена аргументов преобразуются без проблем (за исключением проблемы с точкой). Наконец, объекты доступны в пространстве имен Python (однако объекты фактически недоступны в пространстве имен R; мы обсудим это позже).

Для примера я привожу соответствующий код R. Надеюсь, понятно, что это простое преобразование. Объект `seq_data` – это DataFrame. Если вы знакомы с основами R или `pandas`, вы, вероятно, знакомы с этим типом структуры данных. Если нет, то это, по сути, таблица, то есть последовательность строк, где каждый столбец имеет один и тот же тип.

3. Давайте проведем базовую проверку этого DataFrame следующим образом:

```
print('This dataframe has %d columns and %d rows' % (seq_data.ncol, seq_
data.nrow))

print(seq_data.colnames)

#In R:

# print(colnames(seq.data))

# print(nrow(seq.data))

# print(ncol(seq.data))
```

Опять же, обратите внимание на сходство кода.

4. Вы даже можете смешивать стили, используя такой код:

```
my_cols = robjects.r.ncol(seq_data)

print(my_cols)
```

Вы можете вызывать функции R напрямую; в этом случае мы будем вызывать `ncol`, если в их имени нет точек; однако будьте осторожны. Это отобразит вывод не 26 столбцов, а [26] – вектора, состоящего из 26 элементов. Это связано с тем, что по умолчанию большинство операций в R возвращают векторы. Если вам нужно некоторое количество столбцов, вы должны выполнить `my_cols[0]`. Кроме того, говоря о подводных камнях, обратите внимание, что индексация массива R начинается с 1, тогда как в Python она начинается с 0.

5. Теперь нам нужно выполнить некоторую очистку данных. Например, некоторые столбцы должны интерпретироваться как числа, но вместо этого они читаются как строки:

```
as_integer = robjects.r('as.integer')

match = robjects.r.match

my_col = match('READ_COUNT', seq_data.colnames)[0] # vector returned

print('Type of read count before as.integer: %s' % seq_data [my_col
1].rclass[0])

seq_data [my_col - 1] = as_integer (seq_data[my_col - 1])

print('Type of read count after as.integer: %s' % seq_data [my_col -
1].rclass[0])
```

Функция `match` чем-то похожа на метод `index` в списках Python. Как и ожидается, она возвращает вектор, чтобы мы могли извлечь элемент 0. Он также имеет индекс 1, поэтому мы вычитаем 1 при работе с Python. Функция `as_integer` преобразует столбец в целые числа. Первый принт будет отображать строки (то есть значения, заключенные в кавычки, «"»), тогда как второй принт покажет числа.

6. Нам нужно будет еще немного потрясти этот стол; подробности об этом можно найти в блокноте. Здесь мы завершим получение DataFrame в R (помните, что хотя это объект R, он фактически виден в пространстве имен Python):

```
robjects.r.assign('seq.data', seq_data)
```

Это создаст переменную в пространстве имен R с именем `seq.data`, с содержимым DataFrame из пространства имен Python. Обратите внимание, что после этой операции оба объекта будут независимыми (изменение одного не будет отражаться на другом).

> **Примечание**
>
> Хотя вы можете выполнять построение диаграмм на Python, R имеет встроенные функции построения диаграмм по умолчанию (которые мы здесь проигнорируем). Он также имеет библиотеку под названием `ggplot2`, которая реализует **Grammar of Graphics** («Грамматика графики», декларативный язык для спецификации статистических диаграмм).

7. Что касается нашего конкретного примера, основанного на проекте «1000 геномов человека», сначала мы построим гистограмму с распределением названий центров, где были созданы все дорожки секвенирования. Для этого воспользуемся `ggplot`:

```
from rpy2.robjects.functions import
SignatureTranslatedFunction
ggplot2.theme = SignatureTranslatedFunction(ggplot2.theme,
init_prm_translate = \{'axis_text_x': 'axis.text.x'})
bar = ggplot2.ggplot (seq_data) + ggplot2.geom_bar() +
    ggplot2.aes_string(x='CENTER_NAME') +
    ggplot2.theme(axis_text_x=ggplot2.element_text(angle=90, hjust=1))
robjects.r.png('out.png', type='cairo-png')
bar.plot()
dev_off = robjects.r('dev.off')
dev_off()
```

Вторая строка не очень интересна, но является важной частью стандартного кода. Одна из функций R, которую мы будем вызывать, имеет пара-

метр с точкой в имени. Поскольку вызовы функций Python не могут быть такими, мы должны сопоставить имя параметра `axis.text.x` R с именем `axis_text_r` Python в теме функции. Мы исправляем его (то есть заменяем `ggplot2.theme` на исправленную версию).

Затем мы строим саму диаграмму. Обратите внимание на декларативный характер `ggplot2`, когда мы добавляем данные на диаграмму. Сначала мы указываем кадр данных `seq_data`, затем используем гистограмму с именем `geom_bar`. После этого аннотируем переменную x (`CENTER_NAME`). Окончательно мы поворачиваем текст по *оси X*, меняя тему. Завершаем, закрывая устройство печати R.

8. Теперь мы можем распечатать изображение в Jupyter Notebook:

```
Image(filename='out.png')
```

Получается следующая диаграмма:

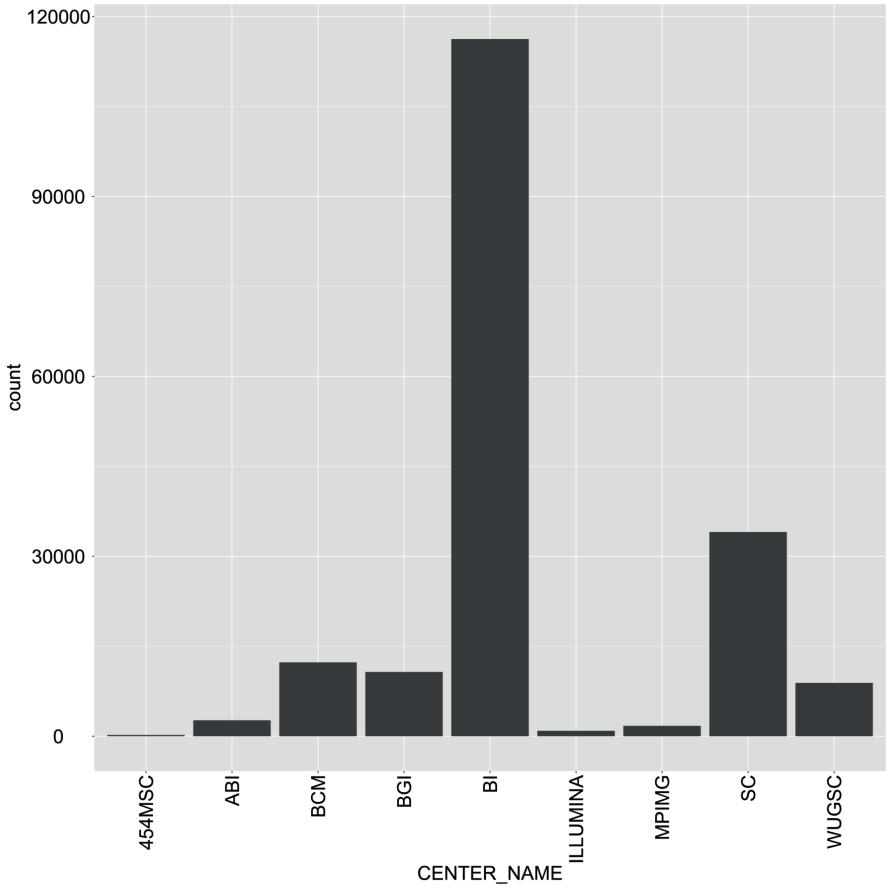

Рис. 1.3. Сгенерированная ggplot2 гистограмма названий центров, которые отвечают за секвенирование дорожек геномных данных человека из проекта «1000 геномов»

9. В качестве последнего примера сделаем точечную диаграмму ридов и количества оснований для всех секвенированных дорожек народа йоруба (YRI) и жителей Юты, происходящих из Северной и Западной Европы (CEU), используя проект «1000 геномов человека» (резюме данных этого проекта, которые мы будем тщательно использовать, можно увидеть в рецепте *«Работа с современными форматами последовательностей»* в главе 3 *«Секвенирование следующего поколения»*). Кроме того, нас интересуют различия между разными типами секвенирования (например, захват экзом[1], высокий захват и низкий захват). Во-первых, мы создаем DataFrame только с дорожками YRI и CEU и ограничиваем максимальное количество оснований и ридов:

```
robjects.r('yri_ceu <- seq.data[seq.data$POPULATION

%in% c("YRI", "CEU") & seq.data$BASE_COUNT < 2E9 & seq.data$READ_COUNT <
3E7, ]')

yri_ceu = robjects.r('yri_ceu')
```

10. Теперь мы готовы построить:

```
scatter = ggplot2.ggplot (yri_ceu) + ggplot2. aes_string(x='BASE_COUNT',
y='READ_COUNT', shape='factor(POPULATION)', col='factor(ANALYSIS_GROUP)')
+ ggplot2.geom_point()

robjects.r.png('out.png')

scatter.plot()
```

Надеемся, что этот пример (пожалуйста, обратитесь к следующему рисунку) проясняет возможности метода Grammar of Graphics. Мы начнем с объявления DataFrame и типа используемой диаграммы (то есть точечной диаграммы, реализованной geom_point).

Обратите внимание, как легко выразить, что форма каждой точки зависит от переменной POPULATION и что цвет зависит от переменной ANALYSIS_GROUP:

[1] Экзом – часть генома, которая отвечает за синтез белка в организме. Экзом составляет всего 1 % генома, но при этом включает более 85 % всех клинически значимых мутаций. – *Прим. ред.*

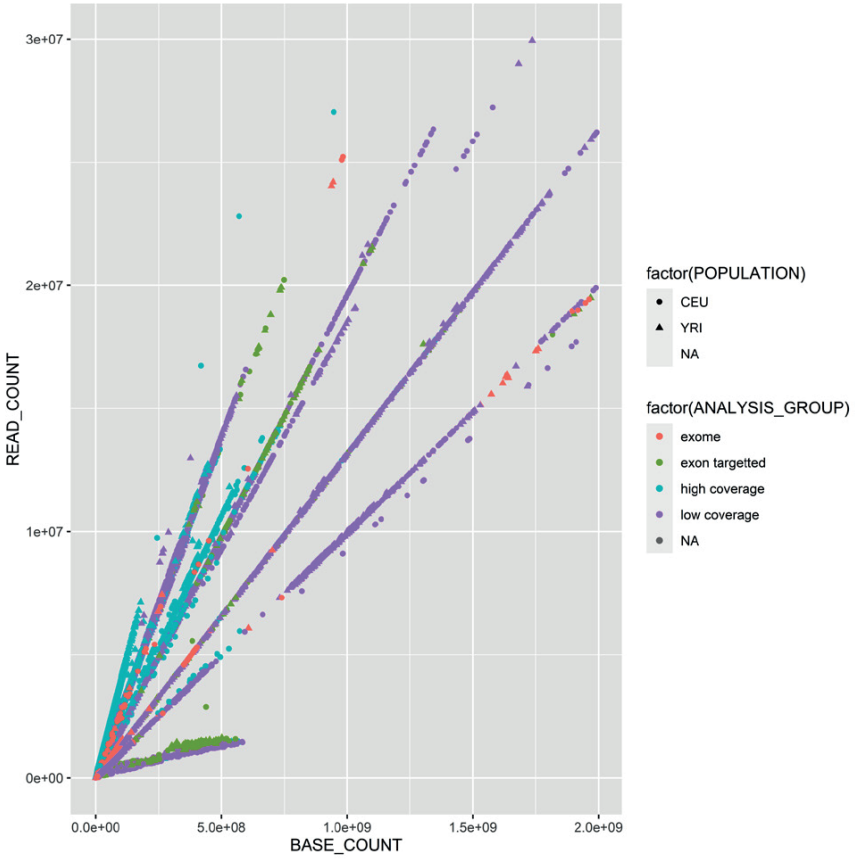

Рис. 1.4. Диаграмма рассеяния, созданная с помощью ggplot2, с количеством оснований и количеством ридов для всех прочитанных дорожек секвенирования; цвет и форма каждой точки отражают категориальные данные (популяцию и тип секвенированных данных)

11. Поскольку R DataFrame очень близок к `pandas`, имеет смысл сделать конвертацию между ними, так как это поддерживается `rpy2`:

```
import rpy2.robjects as ro
from rpy2.robjects import pandas2ri
from rpy2.robjects.conversion import localconverter
with localconverter(ro.default_converter + pandas2ri.converter):
pd_yri_ceu = ro.conversion.rpy2py(yri_ceu)
del pd_yri_ceu['PAIRED_FASTQ']
with localconverter(ro.default_converter + pandas2ri.converter):
no_paired = ro.conversion.py2rpy(pd_yri_ceu)
robjects.r.assign('no.paired', no_paired)
robjects.r("print(colnames(no.paired))")
```

Начнем с импорта необходимого модуля преобразования – `rpy2` предоставляет множество стратегий для преобразования данных из R в Python. Здесь нас интересует преобразование фрейма данных. Затем мы преобразуем R DataFrame (обратите внимание, что мы делаем преобразование `yri_ceu` в пространстве имен R, а не в пространстве имен Python). Мы удаляем столбец, который указывает имя парного файла FASTQ в DataFrame `pandas`, и копируем его обратно в пространство имен R. Если вы напечатаете имена столбцов нового R DataFrame, вы увидите, что `PAIRED_FASTQ` отсутствует.

Дополнительно...

Достойно упоминания, что прогресс в среде программного обеспечения, окружающей Python, происходит с головокружительной скоростью. Это означает, что если какая-то функциональность недоступна сегодня, она может быть разработана в ближайшем будущем. Итак, если вы работаете над новым проектом, обязательно ознакомьтесь с самыми последними разработками в области Python, прежде чем использовать функциональные возможности пакета R.

Существует множество пакетов R для биоинформатики в проекте Bioconductor (http://www.biconductor.org/). Это, вероятно, должно быть вашим первым портом запроса в мире R для функциональных возможностей биоинформатики. Однако обратите внимание, что многих пакетов R Bioinformatics нет в Bioconductor, поэтому обязательно ищите более широкие пакеты R в сети комплексных архивов R (Comprehensive R Archive Network, **CRAN**) (см. CRAN по адресу http://cran.rproject.org/).

Для Python сделано множество графических библиотек. Matplotlib – самая распространенная библиотека, но у вас также есть много других вариантов. В контексте R стоит отметить, что существует ggplot2-подобная реализация для Python, основанная на языке описания Grammar of Graphics для диаграмм, и – сюрприз, сюрприз – она называется `ggplot!` (http://yhat.github.io/ggpy/).

Смотрите также

Чтобы узнать больше об этих темах, обратитесь к следующим ресурсам:

- существует множество руководств и книг по R; посмотрите документацию на веб-странице R (http://www.r-project.org/);
- для Bioconductor см. документацию по адресу http://manuals.bioinformatics. ucr.edu/home/R_BioCondManual;
- если вы работаете с NGS, вы также можете ознакомиться с высокопроизводительным анализом последовательностей с помощью Bioconductor на странице http://manuals.bioinformatics.ucr.edu/home/ht-seq;
- документация по библиотеке `rpy` – это ваш мост от Python к R, ее можно найти по адресу https://rpy2.bitbucket.io/;
- подход *Grammar of Graphics* описан в книге Леланда Уилкинсона, Springer, под названием *«Грамматика графики»*;
- с точки зрения структур данных, функции, аналогичные R, можно найти в библиотеке `pandas`. Некоторые учебные пособия можно найти по адресу http://pandas.pydata.org/pandas-docs/dev/tutorials.html. Книга *«Python для анализа данных»* Уэса МакКинни, O'Reilly Media, также является альтернативой для рассмотрения. В следующей главе мы обсудим `pandas` и будем использовать ее на протяжении всей книги.

Демонстрация R magic с Jupyter

Jupyter предоставляет несколько дополнительных функций по сравнению со стандартным Python. Среди этих функций он предоставляет структуру расширяемых команд, называемых **magic** (на самом деле это работает только с ядром IPython Jupyter, поскольку в действительности это функция IPython, но это то, что нас интересует). Magic позволяют расширить язык многими полезными способами. Существуют функции magic, которые вы можете использовать для работы с R. Как вы увидите в нашем примере, это значительно упрощает взаимодействие с R и делает его более декларативным. В этом рецепте не будут представлены какие-либо новые функциональные возможности R, но, надеюсь, он прояснит, как IPython может дать серьезное повышение производительности научных вычислений в этом отношении.

Подготовка

Вам нужно будет следовать предыдущим шагам раздела *«Подготовка»* в рецепте *«Взаимодействие с R через rpy2»*. Блокнот – `Chapter01/R_magic.py`. Он более полный, чем представленный здесь рецепт, и содержит больше примеров диаграмм. Для краткости мы сконцентрируемся только на фундаментальных конструктах для взаимодействия с R с использованием magic. Если вы используете Docker, вы можете задать следующее:

```
docker run -ti -p 9875:9875 -v YOUR_DIRECTORY:/data tiagoantao/bioinformatics_r
```

Как это сделать...

Этот рецепт является агрессивным упрощением предыдущего, потому что он иллюстрирует лаконичность и элегантность R magic.

1. Первое, что вам нужно сделать, – это загрузить R magic и `ggplot2`:

   ```
   import rpy2.robjects as robjects
   import rpy2.robjects.lib.ggplot2 as ggplot2
   %load_ext rpy2.ipython
   ```

   Обратите внимание, что `%` запускает специфичную для IPython директиву. В качестве простого примера вы можете написать `%R print(c(1, 2))` в ячейку Jupyter.

   Посмотрите, как легко выполнить код R без использования пакета `robjects`. На самом деле `rpy2` используется для того, чтобы «заглянуть под капот».

2. Прочитаем файл `sequence.index`, загруженный в предыдущем рецепте:

   ```
   %%R
   seq.data <- read.delim('sequence.index', header=TRUE, stringsAsFactors=FALSE)
   seq.data$READ_COUNT <- as.integer(seq.data$READ_COUNT)
   seq.data$BASE_COUNT <- as.integer(seq.data$BASE_COUNT)
   ```

Затем вы можете указать, что вся ячейка должна интерпретироваться как код R, используя `%%R` (обратите внимание на двойной `%%`).

3. Теперь мы можем перенести переменную в пространство имен Python:

```
seq_data = %R seq.data

print(type(seq_data)) # pandas dataframe!
```

Тип DataFrame – это не стандартный объект Python, а `pandas` DataFrame. Это отход от предыдущих версий интерфейса R magic.

4. Поскольку у нас есть фрейм данных `pandas`, мы можем легко работать с ним, используя интерфейс `pandas`:

```
my_col = list(seq_data.columns).index ("CENTER_NAME")

seq_data ['CENTER_NAME'] = seq_data['CENTER_NAME'].

apply(lambda` x: x.upper())
```

5. Давайте поместим этот DataFrame обратно в пространство имен R следующим образом:

```
%R -i seq_data

%R print(colnames(seq_data))
```

Аргумент `-i` информирует систему magic о том, что переменная, следующая за пространством Python, должна быть скопирована в пространство имен R. Вторая строка просто показывает, что DataFrame действительно доступен в R. Используемое нами имя отличается от исходного – это `seq_data`, а не `seq.data`.

6. Выполним окончательную очистку (подробности см. в предыдущем рецепте) и распечатаем ту же гистограмму, что и раньше:

```
%%R

bar <- ggplot (seq_data) + aes(factor(CENTER_NAME)) + geom_bar() +
theme(axis.text.x = element_text(angle = 90, hjust = 1))

print(bar)
```

Кроме того, система R magic позволяет сократить код, так как изменяет поведение при взаимодействии R с IPython. Например, в коде `ggplot2` предыдущего рецепта вам не нужно использовать R-функции `.png` и `dev.off`, так как система magic позаботится об этом за вас. Когда вы даете команду R распечатать диаграмму, она магическим образом появляется в вашем блокноте или графической консоли.

Дополнительно...

R magic, похоже, со временем сильно изменился с точки зрения интерфейса. Например, я несколько раз обновлял код R для первого издания этой книги. Текущая версия назначения DataFrame возвращает объекты `pandas`, что является серьезным изменением.

Смотрите также

Для получения дополнительной информации ознакомьтесь с этими ссылками:

- основные инструкции по IPython magic см. на странице https://ipython.readthedocs.io/en/stable/interactive/magics.html;
- список сторонних расширений для IPython, в том числе magic, можно найти по адресу https://github.com/ipython/ipython/wiki/Extensions-Index.