Работа с растровыми данными в R: rgdal

Краткое описание возможностей R по работе с пространственными растровыми данными без конвертации

Обсудить в форуме Комментариев — 14

R — статистический пакет с открытым кодом, обладающий огромным набором функций, которые несомненно могли бы пригодится любым специалистам имеющим дело с анализом данных, в том числе и пространственных.

Целью данной статьи является демонстрация того, как можно осуществлять статистические операции с многоканальными растровыми данными, с которыми мы повседневно работаем в GIS, не переводя их для этого в табличный формат. Работа напрямую с растром экономит время, способствует более чистой логике процесса и значительно облегчает программирование. В качестве примера для тестов можно использовать композитный снимок AVHRR использованный для иллюстраций в статье.

Работу с растрами напрямую делает возможной пакет RGDAL. Перед использованием примера необходимо убедиться что библиотека rgdal установлена, для этого из меню необходимо выбрать Packages\Set CRAN Mirror..., Packages\Install package и загружена Packages\Load package....

Примечание: если выбор этих пунктов из меню приводит к ошибке Error: unexpected input in "packages\", следует, значит вы выбираете не из меню программы, а пытаетесь ввести эти операции как команды в командную строку.

Другим способом загрузки и установки пакета является работа из командной строки:

```
#выбор зеркала загрузки
chooseCRANmirror()
#выбор пакета и его инсталляция
utils:::menuInstallPkgs()
#загрузка пакета для работы
library(rgdal)
```

Чтение растровых данных

Введем в командной строке R следующие команды и выберем файл для анализа, например вот этот.

```
test = file.choose()1
x = new("GDALReadOnlyDataset", test)
```

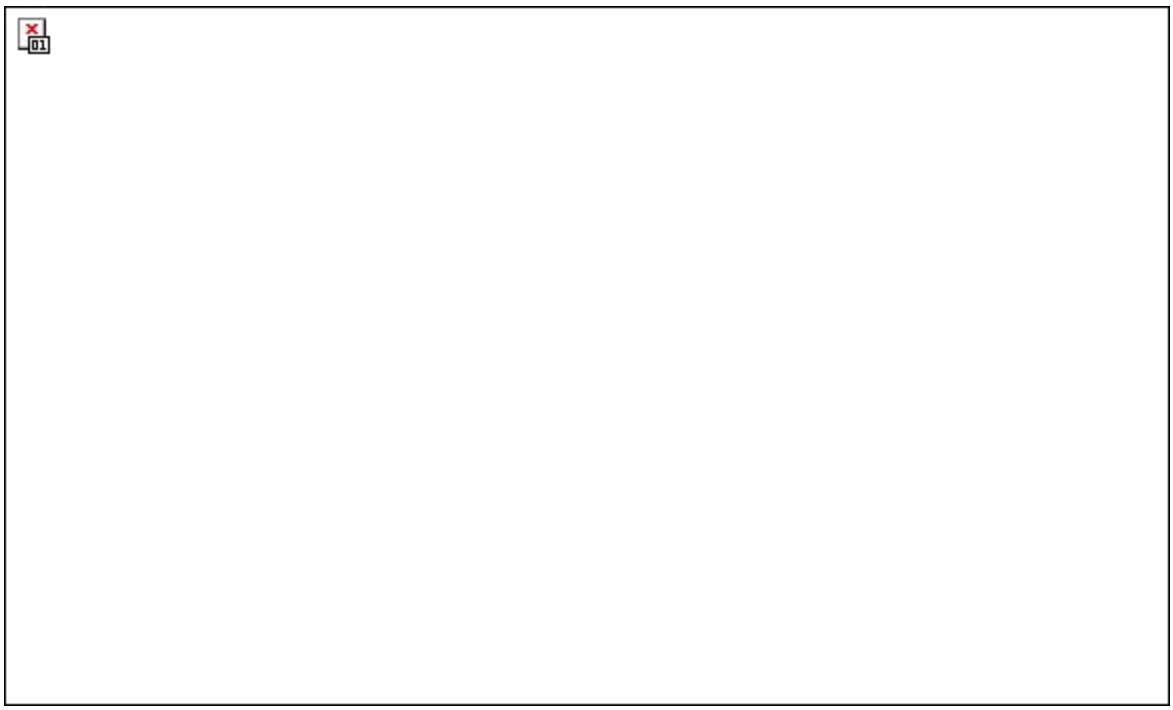
Сразу получим размеры изображения, они понадобятся в дальнейшем для формирования матрицы:

```
height = dim(x)1
width = dim(x)2
```

Работа с растровыми данными

Для начала, посмотрим на распределение данных:

```
plot(density(getRasterTable(x)$band1))
```


Пример графика распределения яркостей в первом канале тестового изображения

Построим регрессию и изучим ее результаты:

```
lm1 = lm(getRasterTable(x)$band2~getRasterTable(x)$band1)
summary(lm1)
```

Для того, чтобы получить из изображения фрейм данных (dataframe в терминологии R) необходимый для некоторых статистический операций можно указать все каналы изображения (в данном случае их 15) следующим образом:

```
imagedata = data.frame(getRasterTable(x)1:15)
```

Как видно из примеров, все операции работают с растром напрямую, без обычного преобразования в другие форматы, что удобно.

Сохранение данных

Самый простой вариант - сохранение данных в текстовый файл с разделителями

```
outputfile="c:\\temp\\1\\test.csv"
write.table(imagedata,file=outputfile,sep=",",row.names=F,col.names=T)
```

Если в результате некоторой операции у вас получилась матрица значений, ее можно сохранить как растр. Такая матрица часто является результатом обработки исходных растровых данных. Если результатом некой операции (над импортированным растром или новыми данными) является вектор (как например при применении деревьев классификации), то его легко сконвертировать в матрицу с нужными размерами, вот так:

```
#для простоты представим в качестве результирующего вектора просто копию первого канала растра
resultvector = getRasterTable(x)$band1
resmtx = matrix(resultvector,width,height)
```

Где resultvector - результирующий вектор, height - высота (количество рядов) матрицы, width - ширина (количество колонок).

rgdal позволяет читать и сохранять растры со следующей разрядностью согласно спецификации GeoTIFF:

- Byte - Eight bit unsigned integer

- UInt16 - Sixteen bit unsigned integer
- Int16 - Sixteen bit signed integer
- UInt32 - Thirty two bit unsigned integer
- Int32 - Thirty two bit signed integer
- Float32 - Thirty two bit floating point
- Float64 - Sixty four bit floating point
- CInt16 - Complex Int16
- CInt32 - Complex Int32
- CFloat32 - Complex Float32
- CFloat64 - Complex Float64

rgdal позволяет читать и сохранять данные в следующих растровых форматах:

- GeoTIFF (GTiff)
- Geosoft GXF (GXF)
- Erdas Imagine (HFA)
- CEOS (CEOS)
- ELAS (ELAS)
- Arc/Info Binary Grid (AIGrid) - только чтение
- SDTS Raster DEM (SDTS)
- OGDI (OGDI)
- ESRI Labelled BIL (EHdr)
- PCI .aux Labelled Raw Raster (PAux)
- HDF4 Hierachal Data Format Release 4
- HDF5 Hierachal Data Format Release 5
- GSAG Golden Software ASCII Grid
- GSBG Golden Software Binary Grid

Для сохранения данных необходимо инициализировать драйвер соответствующего формата и создать растр с разрядностью данных соответствующей данным в матрице:

```
tif_driver = new("GDALDriver", "GTiff")
tif2 = new("GDALTransientDataset", tif_driver, height, width, 1, 'Byte')
```

Если матрица одна то и канал у результирующего растра будет только один:

```
bnd1 = putRasterData(tif2, resmtx)
```

Для многоканальных данных, необходимо задать их количество и несколько видоизменить команду помещения данных в канал, например для 3-х канального растра (b1-b3 векторы с данными):

```
tifraster <- new("GDALTransientDataset", tif_driver, 77, 101, 3, 'Byte')
bnd1 <- putRasterData(tifraster, b1, band=1)
bnd2 <- putRasterData(tifraster, b2, band=2)
bnd3 <- putRasterData(tifraster, b3, band=3)
```

В приведенном выше выражении, переменные bnd1-3 имеют чисто утилитарное значение, в них передается объект результата, который также помещается в канал изображения. Подобное выражение может быть записано и без перенаправления:

```
putRasterData(tifraster, b1, band=1)
```

В этом случае результат будет записан в канал, а имя объекта выведено в консоль, а не в переменную.

После этого остается только назвать результирующий файл и сохранить его:

```
tif_file = "D:\\Programming\\R\\gdal\\test.tif"
saveDataset(tif2, tif_file)
```

После окончания работы стоит также удалить использованные объекты: файл с данными и драйвер формата:

```
GDAL.close(tif2)
GDAL.close(tif_driver)
```

Сохранение файла привязки

Самый простой вариант сохранения привязки - перенести привязку из импортированного файла в новый с помощью создания world-файла. Для этого сначала, считаем данные о привязки, размер пикселя и координаты верхнего угла с помощью GDALinfo:

```
pixsize = as.numeric(GDALinfo(test)6)
originx = as.numeric(GDALinfo(test)4)
originy = as.numeric(GDALinfo(test)5)
tfwinfo = rbind(pixsize,0,0,-1*pixsize,originx,originy)
```

А потом сохраним в текстовый файл с таким же названием, как растр и расширением tfw:

```
tif_file = "D:\\Programming\\R\\raster-open-save-georef\\output.tif"
tfw_file = tif_file
substr(tfw_file, nchar(tfw_file)-2, nchar(tfw_file)) <- "tfw"
write(tfwinfo,tfw_file,sep="\n")
```

Заключение

Таким образом, данный пример иллюстрирует, как применить всю математическую мощность популярного бесплатного пакета для работы с данными для обработки данных в растровых форматах. Язык R легко позволяет работать со скриптами, что позволяет еще больше увеличить производительность.

Если у вас есть свои интересные примеры использования R в анализе пространственных данных, пожалуйста, пишите, мы с удовольствием их опубликуем у нас на сайте.

Обсудить в форуме Комментариев — 14

Ссылки по теме

- Возможности работы с пространственными данными статистического пакета R
- Подробнее о пакете rgdal

Последнее обновление: March 01 2011

Дата создания: 06.11.2007
Автор(ы): Максим Дубинин