The major advantage of Python is that it is a *general-purpose* programming language. It is used in many domains, including desktop software, computer games, websites and data science. Python is often the only shared language between different (geocomputation) communities and can be seen as the 'glue' that holds many GIS programs together. Many geoalgorithms, including those in QGIS and ArcMap, can be accessed from the Python command line, making it well-suited as a starter language for command-line GIS.[17]

For spatial statistics and predictive modeling, however, R is second-to-none. This does not mean you must choose either R or Python: Python supports most common statistical techniques (though R tends to support new developments in spatial statistics earlier) and many concepts learned from Python can be applied to the R world. Like R, Python also supports geographic data analysis and manipulation with packages such as **osgeo**, **Shapely**, **NumPy** and **PyGeoProcessing** (Garrard, 2016).

1.4 R's spatial ecosystem

There are many ways to handle geographic data in R, with dozens of packages in the area.[18] In this book we endeavor to teach the state-of-the-art in the field whilst ensuring that the methods are future-proof. Like many areas of software development, R's spatial ecosystem is rapidly evolving (Figure 1.2). Because R is open source, these developments can easily build on previous work, by 'standing on the shoulders of giants', as Isaac Newton put it in 1675[19]. This approach is advantageous because it encourages collaboration and avoids 'reinventing the wheel'. The package **sf** (covered in Chapter 2), for example, builds on its predecessor **sp**.

A surge in development time (and interest) in 'R-spatial' has followed the award of a grant by the R Consortium for the development of support for Simple Features, an open-source standard and model to store and access vector geometries. This resulted in the **sf** package (covered in Section 2.2.1). Multiple places reflect the immense interest in **sf**. This is especially true for the R-sig-Geo Archives[20], a long-standing open access email list containing much R-spatial wisdom accumulated over the years.

It is noteworthy that shifts in the wider R community, as exemplified by the data

[17]Python modules providing access to geoalgorithms include `grass.script` for GRASS, `saga-python` for SAGA-GIS, `processing` for QGIS and `arcpy` for ArcGIS.

[18]An overview of R's spatial ecosystem can be found in the CRAN Task View on the Analysis of Spatial Data (see `https://cran.r-project.org/web/views/Spatial.html`).

[19]`http://digitallibrary.hsp.org/index.php/Detail/Object/Show/object_id/9285`

[20]`https://stat.ethz.ch/pipermail/r-sig-geo/`

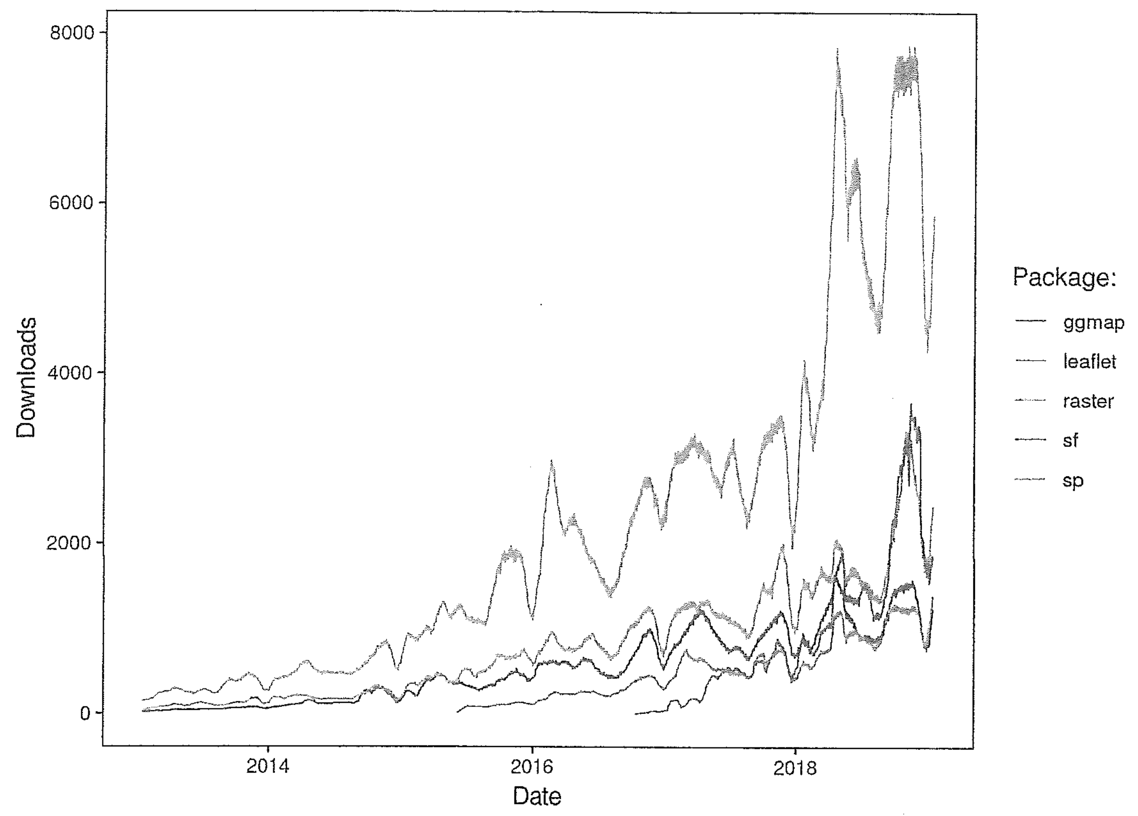

FIGURE 1.2: The popularity of spatial packages in R. The y-axis shows average number of downloads per day, within a 30-day rolling window, of prominent spatial packages.

processing package **dplyr** (released in 2014[21]) influenced shifts in R's spatial ecosystem. Alongside other packages that have a shared style and emphasis on 'tidy data' (including, e.g., **ggplot2**), **dplyr** was placed in the **tidyverse** 'metapackage' in late 2016[22]. The **tidyverse** approach, with its focus on long-form data and fast intuitively named functions, has become immensely popular. This has led to a demand for 'tidy geographic data' which has been partly met by **sf** and other approaches such as **tabularaster**. An obvious feature of the **tidyverse** is the tendency for packages to work in harmony. There is no equivalent **geoverse**, but there are attempts at harmonization between packages hosted in the r-spatial[23] organization and a growing number of packages use **sf** (Table 1.2).

[21] `https://cran.r-project.org/src/contrib/Archive/dplyr/`

[22] `https://cran.r-project.org/src/contrib/Archive/tidyverse/`

[23] `https://github.com/r-spatial/discuss/issues/11`

TABLE 1.2: The top 5 most downloaded packages that depend on sf, in terms of average number of downloads per day over the previous month. As of 2019-01-29 there are 117 packages which import sf.

package	Downloads
ggplot2	22593
plotly	3628
raster	2433
leaflet	1391
spData	1174

1.5 The history of R-spatial

There are many benefits of using recent spatial packages such as **sf**, but it also important to be aware of the history of R's spatial capabilities: many functions, use-cases and teaching material are contained in older packages. These can still be useful today, provided you know where to look.

R's spatial capabilities originated in early spatial packages in the S language (Bivand and Gebhardt, 2000). The 1990s saw the development of numerous S scripts and a handful of packages for spatial statistics. R packages arose from these and by 2000 there were R packages for various spatial methods "point pattern analysis, geostatistics, exploratory spatial data analysis and spatial econometrics", according to an article[24] presented at GeoComputation 2000 (Bivand and Neteler, 2000). Some of these, notably **spatial**, **sgeostat** and **splancs** are still available on CRAN (Rowlingson and Diggle, 1993, 2017; Venables and Ripley, 2002; Majure and Gebhardt, 2016).

A subsequent article in R News (the predecessor of The R Journal[25]) contained an overview of spatial statistical software in R at the time, much of which was based on previous code written for S/S-PLUS (Ripley, 2001). This overview described packages for spatial smoothing and interpolation, including **akima** and **geoR** (Akima and Gebhardt, 2016; Jr and Diggle, 2016), and point pattern analysis, including **splancs** (Rowlingson and Diggle, 2017) and **spatstat** (Baddeley et al., 2015).

The following R News issue (Volume 1/3) put spatial packages in the spotlight again, with a more detailed introduction to **splancs** and a commentary on future prospects regarding spatial statistics (Bivand, 2001). Additionally, the issue introduced two packages for testing spatial autocorrelation that eventually became part of **spdep** (Bivand, 2017). Notably, the commentary mentions

[24] http://www.geocomputation.org/2000/GC009/Gc009.htm

the need for standardization of spatial interfaces, efficient mechanisms for exchanging data with GIS, and handling of spatial metadata such as coordinate reference systems (CRS).

maptools (written by Nicholas Lewin-Koh; Bivand and Lewin-Koh, 2017) is another important package from this time. Initially **maptools** just contained a wrapper around shapelib[26] and permitted the reading of ESRI Shapefiles into geometry nested lists. The corresponding and nowadays obsolete S3 class called "Map" stored this list alongside an attribute data frame. The work on the "Map" class representation was nevertheless important since it directly fed into **sp** prior to its publication on CRAN.

In 2003 Roger Bivand published an extended review of spatial packages. It proposed a class system to support the "data objects offered by GDAL", including 'fundamental' point, line, polygon, and raster types. Furthermore, it suggested interfaces to external libraries should form the basis of modular R packages (Bivand, 2003). To a large extent these ideas were realized in the packages **rgdal** and **sp**. These provided a foundation for spatial data analysis with R, as described in *Applied Spatial Data Analysis with R* (ASDAR) (Bivand et al., 2013), first published in 2008. Ten years later, R's spatial capabilities have evolved substantially but they still build on ideas set-out by Bivand (2003): interfaces to GDAL and PROJ, for example, still power R's high-performance geographic data I/O and CRS transformation capabilities (see Chapters 6 and 7, respectively).

rgdal, released in 2003, provided GDAL bindings for R which greatly enhanced its ability to import data from previously unavailable geographic data formats. The initial release supported only raster drivers but subsequent enhancements provided support for coordinate reference systems (via the PROJ library), reprojections and import of vector file formats (see Chapter 7 for more on file formats). Many of these additional capabilities were developed by Barry Rowlingson and released in the **rgdal** codebase in 2006 (see Rowlingson et al., 2003, and the R-help[27] email list for context).

sp, released in 2005, overcame R's inability to distinguish spatial and non-spatial objects (Pebesma and Bivand, 2005). **sp** grew from a workshop[28] in Vienna in 2003 and was hosted at sourceforge before migrating to R-Forge[29]. Prior to 2005, geographic coordinates were generally treated like any other number. **sp** changed this with its classes and generic methods supporting points, lines, polygons and grids, and attribute data.

sp stores information such as bounding box, coordinate reference system and attributes in slots in `Spatial` objects using the S4 class system, enabling data operations to work on geographic data (see Section 2.2.2). Further, **sp**

[26] `http://shapelib.maptools.org/`

[27] `https://stat.ethz.ch/pipermail/r-help/2003-January/028413.html`

[28] `http://spatial.nhh.no/meetings/vienna/index.html`

[29] `https://r-forge.r-project.org`

provides generic methods such as `summary()` and `plot()` for geographic data. In the following decade, **sp** classes rapidly became popular for geographic data in R and the number of packages that depended on it increased from around 20 in 2008 to over 100 in 2013 (Bivand et al., 2013). As of 2018 almost 500 packages rely on **sp**, making it an important part of the R ecosystem. Prominent R packages using **sp** include: **gstat**, for spatial and spatio-temporal geostatistics; **geosphere**, for spherical trigonometry; and **adehabitat** used for the analysis of habitat selection by animals (Pebesma and Graeler, 2018; Calenge, 2006; Hijmans, 2016).

While **rgdal** and **sp** solved many spatial issues, R still lacked the ability to do geometric operations (see Chapter 5). Colin Rundel addressed this issue by developing **rgeos**, an R interface to the open-source geometry library (GEOS) during a Google Summer of Code project in 2010 (Bivand and Rundel, 2018). **rgeos** enabled GEOS to manipulate **sp** objects, with functions such as `gIntersection()`.

Another limitation of **sp** — its limited support for raster data — was overcome by **raster**, first released in 2010 (Hijmans, 2017). Its class system and functions support a range of raster operations as outlined in Section 2.3. A key feature of **raster** is its ability to work with datasets that are too large to fit into RAM (R's interface to PostGIS supports off-disc operations on vector geographic data). **raster** also supports map algebra (see Section 4.3.2).

In parallel with these developments of class systems and methods came the support for R as an interface to dedicated GIS software. **GRASS** (Bivand, 2000) and follow-on packages **spgrass6** and **rgrass7** (for GRASS GIS 6 and 7, respectively) were prominent examples in this direction (Bivand, 2016a,b). Other examples of bridges between R and GIS include **RSAGA** (Brenning et al., 2018, first published in 2008), **RPyGeo** (Brenning, 2012a, first published in 2008), and **RQGIS** (Muenchow et al., 2017, first published in 2016) (see Chapter 9).

Visualization was not a focus initially, with the bulk of R-spatial development focused on analysis and geographic operations. **sp** provided methods for map making using both the base and lattice plotting system but demand was growing for advanced map making capabilities, especially after the release of **ggplot2** in 2007. **ggmap** extended **ggplot2**'s spatial capabilities (Kahle and Wickham, 2013), by facilitating access to 'basemap' tiles from online services such as Google Maps. Though **ggmap** facilitated map-making with **ggplot2**, its utility was limited by the need to `fortify` spatial objects, which means converting them into long data frames. While this works well for points it is computationally inefficient for lines and polygons, since each coordinate (vertex) is converted into a row, leading to huge data frames to represent complex geometries. Although geographic visualization tended to focus on vector data, raster visualization is supported in **raster** and received a boost with the release of **rasterVis**, which is described in a book on the subject of

ıphic data. In
raphic data in
ı around 20 in
. 500 packages
Prominent R
.l geostatistics;
ɔr the analysis
Calenge, 2006;

the ability to
ssed this issue
metry library
ıd and Rundel,
.ctions such as

was overcome
. and functions
A key feature
ɔ fit into RAM
.or geographic

ıods came the
ASS (Bivand,
.SS GIS 6 and
rand, 2016a,b).
GA (Brenning
first published
l in 2016) (see

ıl development
:thods for map
; demand was
: the release of
ies (Kahle and
online services
with **ggplot2**,
, which means
ll for points it
ıch coordinate
:s to represent
ɔd to focus on
:ceived a boost
the subject of

spatial and temporal data visualization (Lamigueiro, 2018). As of 2018 map making in R is a hot topic with dedicated packages such as **tmap**, **leaflet** and **mapview** all supporting the class system provided by **sf**, the focus of the next chapter (see Chapter 8 for more on visualization).

1.6 Exercises

1. Think about the terms 'GIS', 'GDS' and 'geocomputation' described above. Which (if any) best describes the work you would like to do using geo* methods and software and why?
2. Provide three reasons for using a scriptable language such as R for geocomputation instead of using an established GIS program such as QGIS.
3. Name two advantages and two disadvantages of using mature vs recent packages for geographic data analysis (for example **sp** vs **sf**).

Geocomputation with R

Robin Lovelace
Jakub Nowosad
Jannes Muenchow

© 2019

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business