8 Interfaces to the Outside World

Watch a video of this chapter

Data are read in using ***connection*** interfaces. Connections can be made to files (most common) or to other more exotic things.

- `file` , opens a connection to a file
- `gzfile` , opens a connection to a file compressed with gzip
- `bzfile` , opens a connection to a file compressed with bzip2
- `url` , opens a connection to a webpage

In general, connections are powerful tools that let you navigate files or other external objects. Connections can be thought of as a translator that lets you talk to objects that are outside of R. Those outside objects could be anything from a data base, a simple text file, or a a web service API. Connections allow R functions to talk to all these different external objects without you having to write custom code for each object.

8.1 File Connections

Connections to text files can be created with the `file()` function.

```
> str(file)
function (description = "", open = "", blocking = TRUE, encoding = getOption("encod
    raw = FALSE, method = getOption("url.method", "default"))
```

The `file()` function has a number of arguments that are common to many other connection functions so it's worth going into a little detail here.

- `description` is the name of the file
- `open` is a code indicating what mode the file should be opened in

The `open` argument allows for the following options:

- “r” open file in read only mode
- “w” open a file for writing (and initializing a new file)
- “a” open a file for appending
- “rb”, “wb”, “ab” reading, writing, or appending in binary mode (Windows)

In practice, we often don’t need to deal with the connection interface directly as many functions for reading and writing data just deal with it in the background.

For example, if one were to explicitly use connections to read a CSV file in to R, it might look like this,

```
> ## Create a connection to 'foo.txt'
> con <- file("foo.txt")
>
> ## Open connection to 'foo.txt' in read-only mode
> open(con, "r")
>
> ## Read from the connection
> data <- read.csv(con)
>
> ## Close the connection
> close(con)
```

which is the same as

```
> data <- read.csv("foo.txt")
```

In the background, `read.csv()` opens a connection to the file `foo.txt`, reads from it, and closes the connection when it’s done.

The above example shows the basic approach to using connections. Connections must be opened, then the are read from or written to, and then they are closed.

8.2 Reading Lines of a Text File

Text files can be read line by line using the `readLines()` function. This function is useful for reading text files that may be unstructured or contain non-standard data.

```
> ## Open connection to gz-compressed text file
> con <- gzfile("words.gz")
> x <- readLines(con, 10)
> x
 [1] "1080"     "10-point" "10th"     "11-point" "12-point" "16-point"
 [7] "18-point" "1st"      "2"        "20-point"
```

For more structured text data like CSV files or tab-delimited files, there are other functions like `read.csv()` or `read.table()`.

The above example used the `gzfile()` function which is used to create a connection to files compressed using the gzip algorithm. This approach is useful because it allows you to read from a file without having to uncompress the file first, which would be a waste of space and time.

There is a complementary function `writeLines()` that takes a character vector and writes each element of the vector one line at a time to a text file.

8.3 Reading From a URL Connection

The `readLines()` function can be useful for reading in lines of webpages. Since web pages are basically text files that are stored on a remote server, there is conceptually not much difference between a web page and a local text file. However, we need R to negotiate the communication between your computer and the web server. This is what the `url()` function can do for you, by creating a `url` connection to a web server.

This code might take time depending on your connection speed.

```
> ## Open a URL connection for reading
> con <- url("http://www.jhsph.edu", "r")
>
> ## Read the web page
> x <- readLines(con)
>
> ## Print out the first few lines
> head(x)
[1] "<!DOCTYPE html>"
[2] "<html lang=\"en\">"
[3] ""
[4] "<head>"
[5] "<meta charset=\"utf-8\" />"
[6] "<title>Johns Hopkins Bloomberg School of Public Health</title>"
```

While reading in a simple web page is sometimes useful, particularly if data are embedded in the web page somewhere. However, more commonly we can use URL connection to read in specific data files that are stored on web servers.

Using URL connections can be useful for producing a reproducible analysis, because the code essentially documents where the data came from and how they were obtained. This is approach is preferable to opening a web browser and downloading a dataset by hand. Of course, the code you write with connections may not be executable at a later date if things on the server side are changed or reorganized.