R Reference Manual

Contents

1 Introduction **5**

2 Import **7**

2.1 Create Data 7

2.2 Import Data from a Local Drive 8

2.3 Import Data from the Internet 9

2.4 Import Data from a Database 10

2.5 Reference Material 10

3 Tidy **11**

3.1 Explore Raw Data 11

3.2 Tidy Data 12

3.3 Prepare Data for Analysis 13

4 Transform **17**

4.1 Arithmetic & Summary Statistics 17

4.2 Create New Variables or Modify Existing Ones 18

4.3 Dates and Datetimes 18

4.4 Merge or Append Data 19

4.5 Narrow in on Observations of Interest 20

4.6 Test 21

5 Visualize **23**

5.1 Flowcharts 23

5.2 Charts 23

5.3 Interfaces 23

6 Model **25**

6.1 25

7 Communicate **27**

7.1 Format Output . . . 27

7.2 Export . . . 27

8 Program **29**

8.1 Conditionals . . . 29

8.2 Environment and Workspace . . . 29

8.3 Evaluation (Standard and Non-standard) . . . 31

8.4 Functionals . . . 31

8.5 Functions . . . 32

8.6 Learn About an Object . . . 32

8.7 Optimization . . . 32

8.8 Pipes . . . 32

8.9 Selecting & Subsetting . . . 33

8.10 Version Control . . . 33

9 Referenced Packages **35**

9.1 A-D . . . 35

9.2 E-H . . . 35

9.3 I-L . . . 36

9.4 M-P . . . 36

9.5 Q-T . . . 37

9.6 U-Z . . . 37

10 References & Resources **39**

Chapter 1

Introduction

The purpose of this manual is to increase my ability to use R well, which means that I must be able to get things done. My vision is that this manual will allow me to document the material I learn in a way that will help me work more effectively and efficiently. Most of the documentation here will link to existing online material so that my notes supplement rather than duplicate.

In order to help me reference material in the context I would use it, I organized the material according to Garrett Grolemund and Hadley Wickham's conception of the tools needed to tackle about 80% of the tasks required in a typical data science project ("Introduction", *R for Data Science*). Those tools are: import, tidy, transform, visualize, model, communicate, and program.

Chapter 2

Import

2.1 Create Data

2.1.1 Data Structures

2.1.1.1 Atomic Vector

`base`

- `c()`: Combine values into a vector or list.
- `factor()`: Create a factor variable.
- `seq()`: Generate a sequence.
- `vector()`: Create a vector.

2.1.1.2 Matrix

`base`

- `matrix()`: Create a matrix.

2.1.1.3 Array

`base`

- `array()`: Create an array.

2.1.1.4 List

`base`

- `c()`: Combine values into a vector or list.

`tibble`

- `tibble()`: Create a data frame or list.

2.1.1.5 Data Frame

`base`

- `data.frame()`: Create a data frame.
  - Prefer `tibble::tibble()` over `base::data.frame()`.
- `list()`: Create a list.

`tibble`

- `tibble()`: Create a data frame or list.

2.1.1.6 References

- "Vectors" (Grolemund & Wickham, *R for Data Science*)
- "Data structures" (Wickham, *Advanced R*)

2.1.2 Other

`stats`

- `rnorm()`: Generate a random normal distribution.

2.2 Import Data from a Local Drive

`base`

- `attach()`: Attach a set of R objects to the search path.
  - Allows objects in the database to be accessed by giving their names (e.g., `height` rather than `women$height`).
- `file.choose()`: Choose a file interactively.
  - Use as `file = file.choose()` in `read.table()` and similar functions.
- `load()`: Reload datasets saved with `save()`.
- `readRDS()`: Restore an R object written with `saveRDS()`.

`data.table`

- `fread()`: Similar to `read.table()`, but faster and more convenient for large data sets.

`foreign`

- `read.spss()`: Read an SPSS data file.

`haven`

- `read_sas()`: Read and write SAS files.

`readr`

- `read_delim()`: Read a delimited (including csv and txv) file.
  - Child functions: `read_csv()`, `read_csv2()`, `read_tsv()`.

`readxl`

- `excel_sheets()`: List all sheets in an Excel spreadsheet.

`utils`

- `data()`: Load specified data sets, or list the available data sets.
  - Use this function to load the data sets that accompany R packages, such as `openintro`'s `hsb2` and `email50` and `gapminder`'s `gapminder`.
- `read.table()`: Read a file in table format.
  - Child functions: `read.csv()`, `read.csv2()`, `read.delim()`, `read.delim2()`.

`XLConnect`

- `readWorksheetFromFile()`: Read data from worksheets in an Excel file.

2.3 Import Data from the Internet

`httr`

- `GET()`: Get a URL.

`jsonlite`

- `read_json()`: Read and write JSON.

`readr`

- `read_delim()`: Read a delimited file (including csv & tsv) into a tibble.
  - Child functions: `read_csv()`, `read_csv2()`, `read_tsv()`.

`rjson`

- `fromJSON()`: Convert JSON to R.

`utils`

- `download.file()`: Download a file from the Internet.

2.4 Import Data from a Database

`DBI`

- `dbBind()`: Bind values to a parameterized/prepared statement.
- `dbClearResult()`: Free all resources (local and remote) associated with a result set.
- `dbConnect()`: Connect to a DBMS.
- `dbDataType()`: Determine the SQL data type of an object.
- `dbDisconnect()`: Disconnect (close) a connection to a DBMS.
- `dbFetch()`: Fetch records from a previously executed query.
- `dbGetQuery()`: Send query, retrieve the results, and then clear result set.
- `dbListTables()`: List remote tables.
- `dbReadTable()`: Copy data frames to and from database tables.
- `dbSendQuery()`: Execute a query on a given database connection.
- `dbSendStatement()`: Execute a data manipulation statement on a given database connection.

2.5 Reference Material

The `openintro` package contains data sets useful for practicing and teaching.

Chapter 3

Tidy

"Tidying your data means storing it in a consistent form that matches the semantics of the dataset with the way it is stored. In brief, when your data is tidy, each column is a variable and each row is an observation. Tidying data is important because the consistent structure lets you focus your struggle on questions about the data, not fighting to get the data into the right form for different functions."
- Garrett Grolemund & Hadley Wickham, *R for Data Science*

3.1 Explore Raw Data

3.1.1 Understand the Structure of the Data

`base`

- `class()`: Get or set the class attribute of an object.
- `colnames()`: Retrieve or set the column names of a matrix-like object.
- `dim()`: Retrieve or set the dimension of an object.
- `dimnames()`: Retrieve or set the dimension names of an object.
- `format()`: Format an R object.
- `length()`: Get or set the length of an object.
- `levels()`: Get or set the (factor) levels of a variable.
- `mode()`: Get or set the storage mode attribute of an object.
  - Modes include logical, numeric (the mode equivalent of `typeof()`'s integer and double), complex, character, raw, and list.
- `names()`: Get or set the names of an object.
- `rownames()`: Retrieve or set the row names of a matrix-like object.
- `typeof()`: Display the R internal type of an object.
  - Types include logical, integer, double, complex, character, raw, and list.

`tibble`

- `glimpse()`: Get a glimpse of the data.

  - Similar to `utils::str()`.

`utils`

- `str()`: Display the structure of an R object.
  - Similar to `tibble::glimpse()`.

3.1.2 Look at the Data

`base`

- `names()`: Get or set the names of an object.
- `summary()`: Summarize the object.

`utils`

- `head()`: View the first observations in a data frame.
- `tail()`: View the last observations in a data frame.

3.1.3 Visualize the Data

`graphics`

- `hist()`: Create a histogram.
- `plot()`: Create an x-y plot.

3.2 Tidy Data

3.2.1 Manage Columns and Observations

`splitstackshape`

- `cSplit()`: Split concatencated values into separate values.

`tidyr`

- `gather()`: Gather columns.
- `separate()`: Separate one column into multiple columns.
- `spread()`: Spread across multiple columns.
- `unite()`: Unite multiple columns into one.

3.2.2 Transpose

`purrr`

- `transpose()`: Turn a list-of-lists inside-out.

3.3 Prepare Data for Analysis

3.3.1 Coerce Data

`base`

- `as.*()` functions:
  - `as.array()`: Coerce to array.
  - `as.data.frame()`: Coerce to data frame.
    - Prefer`tibble::as_tibble()` to `base::as.data.frame()`.
  - `as.Date()`: Coerce to date.
  - `as.factor()`: Coerce to factor.
  - `as.list()`: Coerce to list.
  - `as.matrix()`: Coerce to matrix.
  - `as.POSIX*()`: Coerce to POSIXlt or POSIXct.
- `unclass()`: Remove the class attribute of an object.

`methods`

- `as()`: Force an object to belong to a class.

`tibble`

- `as_tibble()`: Coerce lists and matrices to data frames.
  - Preferable to `base::as.data.frame()`.

3.3.2 Dates and Datetimes

`anytime`

- `anytime()`: Parse POSIXct or Date objects from input data.

`base`

- `as.Date()`: Date conversion to and from character.
- `as.POSIX*()`: Date-time conversion for POSIXct and POSIXlt.
  - `as.POSIXct()`: Setting default for UTC and 1970.
- `strptime()`: Date-time conversion to and from character.
- `Sys.timezone()`: Return the name of the current time zone.
  - `OlsonNames()` displays available time zones.

`fasttime`

- `fastPOSIXct()`: Convert strings into POSICct object (string must be in year, month, day, hour, minute, second format.)

`hms`

- `hms()`: Store time-of-day values as `hms` class.
  - Child functions: `as.hms()`, 'is.hms().

`lubridate`

- `parse_date_time()`: User friendly date-time parsing functions that can accomodate parsing multiple dates in different formats.
  - `fast_strptime()`: Fast C parser of numeric formats only that accepts explicit format arguments, just as `base::strptime()`.
    - Note that the format argument must match the input exactly, including any non-white space characters (such as "T" and "Z").
  - `make_date()`: Create dates from numeric representations.
  - `make_datetime()`: Create date-times from numeric representations.
  - `parse_date_time2()`: Fast C parser of numeric orders.
  - `parse_date_time()` can be slow because it is designed to be forgiving and flexible. If the dates you are working with are in a consistent format (ideally ISO 8601), use one of the following: `fasttime::fastPOSIXct()`
- `ymd()`: Parse dates with year, month, and day components. + Related formats: `ydm()`, `mdy()`, `myd()`, `dmy()`, `dym()`, `yq()`.
- `ymd_hms()`: Parse date-times with year, month, day, hour, minute, and second components. + Related formats: `ymd_hm()`, `ymd_h()`, `dmy_hms()`, `dmy_hm()`, `dmy_h()`, `mdy_hms()`, `mdy_hm()`, `mdy_h()`, `ydm_hms()`, `ydm_hm()`, `ydm_h()`.

3.3.3 Factors and Levels

`base`

- `factor()`: Get and set factors.
  - Rearrange the order of factors by using the `levels` argument. For example, rearrange the order of "Bad,"Good," and "Neutral" using 'levels = c("Bad", "Neutral", "Good").

3.3.4 Filter and Remove Data

`purrr`

- `keep()`: Keep or discard elements using a predicate function.

`stats`

- `na.omit()`: Remove rows with `NA` values.

3.3.5 Strings

`base`

- `cat()`: Concatenate and print.
- `chartr()`: Change certain characters.
- `grep()`: Pattern matching and replacement.
  - `grep()` family functions: `grepl()`, `sub()`, `gsub()`, `regexpr()`, `gregexpr()`, and `regexec()`.
- `tolower()`: Convert to lowercase.
  - `stringr::str_to_lower()` is an alternative.
- `toupper()`: Convert to uppercase.
  - `stringr::str_to_upper()` is an alternative.

`stringr`

- `str_detect()`: Detect the presence or absence of a pattern in a string.
  - Control the `pattern` argument options with `regex()` (i.e., `str_detect(x, regex(pattern, ignore_case = TRUE))`).
- `str_to_lower()`: Convert to lower case.
- `str_to_title()`: Capitalize the first letter.
- `str_to_upper()`: Convert to upper case.

3.3.6 Test Data

`base`

- `all()`: Are all values true?
- `any()`: Are any values true?
  - Use `any(is.na(data.frame))` to determine if there are any NA values in a data frame.
- `exists()`: Check whether an R object exists.
- `is.*()` functions:
  - `is.array()`: Test whether an object is an array.
  - `is.data.frame()`: Test whether an object is a data frame.
  - `is.matrix()`: Test whether an object is a matrix.
  - `is.vector()`: Test whether an object is a vector.
- `setequal()`: Check two vectors for equality.

`purrr`

- `every()`: Do every or some elements of a list satisfy a predicate?

`stats`

- `complete.cases()`: Find complete cases (i.e., rows without `NA` values).

`tibble`

- `is_tibble()`: Test whether an object is a tibble.

Chapter 4

Transform

"Transformation includes narrowing in on observations of interest (like all people in one city, or all data from the last year), creating new variables that are functions of existing variables (like computing velocity from speed and time), and calculating a set of summary statistics (like counts or means)."
- Garrett Grolemund & Hadley Wickham, *R for Data Science*

4.1 Arithmetic & Summary Statistics

`base`

- `abs()`: Compute absolute value.
- `colMeans()`: Compute the column mean.
- `colSums()`: Compute the column sum.
- Comparison Operators: Binary operators which allow the comparison of values in atomic vectors.
  - `<`, `>`, `<=`, `>=`, `==`, `!=`.
  - `identical()` and `all.equal()` rather than `==` and `!=` should be used in tests where a single `TRUE` or `FALSE` is required (such as `if` expressions).
- `diff()`: Compute the difference between two objects.
- `max()`: Return the maximum value.
- `mean()`: Compute the mean.
- `min()`: Return the minimum value.
- `round()`: Round numbers.
- `rowMeans()`: Compute the row mean.
- `rowSums()`: Compute the row sum.
- `sqrt()`: Compute square root.
- `sum()`: Sum elements.

`dplyr`

- `count()`: Count/tally observations by group.
- `group_by()`: Group by one or more variables.

- `n()`: Get the number of observations in a current group.
  - Must be used within `summarise()`, `mutate()`, or `filter()`).
- `n_distinct()`: Count the number of unique values in a vector.
- `summarise()`: Reduce multiple values to a single value.
- `tally()`: An alternative to `count()`.

`stats`

- `aggregate()`: Compute summary statistics of data subsets.
- `cor()`: Correlation.
- `cov()`: Covariance.
- `rnorm()`: Generate a random normal distribution.
- `var()`: Variance.

4.2 Create New Variables or Modify Existing Ones

`dplyr`

- `mutate()`: Add new variables.
  - `mutate()` can also be used to modify existing variables. To change the case of a character variable, for example, do something like:

```
df <-
  df %>%
  mutate(var_name = str_to_lower(var_name))
```

4.3 Dates and Datetimes

`base`

- `date()`: Get the current system date and time.
- `difftime()`: Time intervals and differences.
  - `difftime()` is the function behind the `-` operator when used with dates and datetimes (e.g., `time_1 - time_2` is equivalent to `difftime(time_1, time_2)`). The advantage of using `difftime()` over `-`, however, is the `units` argument because it allows you to specify the unit of time in which the difference is calculated.
- `months()`: Extract the month names.
- `quarters()`: Extract the calendar quarters.
- `Sys.Date()`: Get the current date in the current time zone.
- `Sys.time()`: Get the absolute date-time value (which can be converted to various time zones and may return different days).
- `weekdays()`: Extract weekday names.

`lubridate`

- `date()`: Get or set the date component of a date-time.
- `day()`: Get or set the day component of a datetime.
- `month()`: Get or set the month component of a datetime.
- `now()`: The current time (as a POSIXct object).
- `quarter()`: Get or set the fiscal quarter or semester component of a datetime.
- `round_date()`: Round the datetime to the nearest datetime.
  - Child functions: `ceiling_date()`, `floor_date()`.
- Time spans: Duration:
  - `dseconds()`, `dminutes()`, `dhours()`, `ddays()`, `dweeks()`, `dyears()`.
  - Use when you are interested in seconds elapsed.
- Time spans: Interval:
  - `interval()`, `%--%`, `is.interval()`, `int_start()`, `int_end()`, `int_length()`, `int_flip()`, `int_shift()`, `int_overlaps()`, `int_standardize()`, `int_aligns()`, `int_diff()`.
  - Use when you have a start and end.
- Time spans: Period:
  - `seconds()`, `minutes()`, `hours()`, `days()`, `weeks()`, `months()`, `years()`.
  - Use when you are interested in human units.
- Time zones:
  - `force_tz()`: Change the time zone without changing the clock time.
  - `tz()`: Extract the time zone from a datetime.
  - `with_tz()`: View the same instant in a different time zone.
- `today()`: The current date (as a Date object).
- `%m+%` & `%m-%`: Add and subtract months to a date without exceeding the last day of the new month.
- `%within%`: Test whether a date or interval falls within an interval.
- `year()`: Get or set the year component of a datetime.

4.4 Merge or Append Data

`base`

- `append()`: Add elements to a vector.
- `cbind()`: Combine objects by column.
- `intersect()`: Combine data shared in common between two datasets.
  - Similar to `dplyr::semi_join()`.
- `merge()`: Merge two data frames.
  - `dplyr::join` functions are an alternative to `merge()`.
- `rbind()`: Combine objects by row.
- `setdiff()`: Find the difference between two vectors.
  - Similar to `dplyr::anti_join()`.
- `union()`: Combine two datasets without duplicating values.

`dplyr`

- `bind()`: Bind multiple data frames by row and column.
  - Child functions: bind_rows(), bind_cols(), combine().
- Join Functions: Join two tables.
  - Filtering Joins:
    - anti_join(): Return all rows from `x` where there are not matching values in `y`, keeping just columns from `x`.
    - semi_join(): Return all rows from `x` where there are matching values in `y`, keeping just columns from `x`. A semi join differs from an inner join because an inner join will return one row of `x` for each matching row of `y`, where a semi join will never duplicate rows of `x`.
  - Mutating Joins:
    - full_join(): Return all rows and all columns from both `x` and `y`. Where there are not matching values, returns `NA` for the one missing.
    - inner_join(): Return all rows from `x` where there are matching values in `y`, and all columns from `x` and `y`. If there are multiple matches between `x` and `y`, all combination of the matches are returned.
    - left_join(): Return all rows from `x`, and all columns from `x` and `y`. Rows in `x` with no match in `y` will have `NA` values in the new columns. If there are multiple matches between `x` and `y`, all combinations of the matches are returned.
    - right_join(): Return all rows from `y`, and all columns from `x` and `y`. Rows in `x` with no match in `y` will have `NA` values in the new columns. If there are multiple matches between `y` and `x`, all combinations of the matches are returned.

`tibble`

- `add_column()`: Add columns to a data frame.
- `add_row()`: Add rows to a data frame.

4.5 Narrow in on Observations of Interest

`base`

- `droplevels()`: Drop unused levels from factors.
  - This function will keep levels that have even 1 or 2 counts. If you want to remove levels with low counts from a data set in order to simplify your analysis, first `filter()` out those rows and then use `droplevels()`.
- `prop.table()`: Express table entries as proportions of the marginal table.
  - The input is a table produced by `table()`.
  - As these are proportions of the whole, `sum(prop.table(table_name))` = 1.
  - Specify conditional proportions on rows or columns by using the `margin` argument.
- `table()`: Build a table of the counts at each combination of factor levels.
  - Use `prop.table()` to see the table entries expressed as proportions.

`dplyr`

- `arrange()`: Arrange rows by variable, in ascending order.
- `distinct()`: Select distinct rows.
- `filter()`: Return rows with matching conditions.
- `rename()`: Rename variables by name (a modification of `select()`).
- `sample_n()`: Sample n rows from a table.
- `select()`: Select/rename variables.
  - Helper functions include: `starts_with()`, `ends_with()`, `contains()`, `matches()`, `num_range()`, and `one_of()`.

4.6 Test

`base`

- `setequal()`: Check two vectors for equality.

Chapter 5

Visualize

"Visualisation is a fundamentally human activity. A good visualisation will show you things that you did not expect, or raise new questions about the data. A good visualisation might also hint that you're asking the wrong question, or you need to collect different data. Visualisations can surprise you, but don't scale particularly well because they require a human to interpret them."
- Garrett Grolemund & Hadley Wickham, *R for Data Science*

5.1 Flowcharts

`diagram`

`DiagrammeR`

5.2 Charts

`ggplot2`

- `ggplot()`: Create a plot.
- `facet_wrap()`: Wrap a 1D ribbon of panels into 2D (observe a variable, conditional on another variable).

`graphics`

- `boxplot()`: Create a box-and-whisker plot.

5.3 Interfaces

`shiny`

Chapter 6

Model

"Models are complementary tools to visualisation. Once you have made your questions sufficiently precise, you can use a model to answer them. Models are a fundamentally mathematical or computational tool, so they generally scale well. … But every model makes assumptions, and by its very nature a model cannot question its own assumptions. That means a model cannot fundamentally surprise you. - Garrett Grolemund & Hadley Wickham, *R for Data Science*"

6.1

`broom`

- `tidy()`: Construct a data frame that summarizes the model's statistical findings.

`dplyr`

- `sample_n()`: Sample n rows from a table.

`stats`: Statistical functions.

- `coef()`: Extract model coefficients.

Chapter 7

Communicate

"The last step of data science is communication, an absolutely critical part of any data analysis project. It doesn't matter how well your models and visualisation have led you to understand the data unless you can also communicate your results to others."
- Garrett Grolemund & Hadley Wickham, *R for Data Science*

7.1 Format Output

`base`

- `format()`: Format an object for pretty printing.

`lubridate`

- `stamp()`: Format dates and times based on human-friendly templates.

`scales`: Scale functions for visualization.

- `dollar()`: Round to the nearest cent and display dollar sign.

7.2 Export

`base`

- `file.path()`: Construct a file path.
- `save()`: Save R objects.
- `saveRDS()`: Save a single R object.

  - See "A better way of saving and loading objects in R" to understand the differences between `save()` and `saveRDS()`.

`readr`

- `write_delim()`: Write a data frame to a delimited file.
  - About twice as fast as `write.csv()` and never writes row names.
  - Child functions: `write_csv()`, `write_excel_csv()`, `write_tsv()`.

`utils`

- `write.table()`: Data output.
  - Prefer `readr::write_delim()` to `utils::write.table()`.
  - Child functions: `write.csv()`, `write.csv2()`.

`XLConnect`: Read, write, and format Excel data.

Chapter 8

Program

"Surrounding [the tools for importing, tidying, transforming, visualising, modeling, and communicating data] is programming. Programming is a cross-cutting tool that you use in every part of a project. You don't need to be an expert programmer to be a data scientist, but learning more about programming pays off because becoming a better programmer allows you to automate common tasks, and solve new problems with greater ease."
- Garrett Grolemund & Hadley Wickham, *R for Data Science*

8.1 Conditionals

`base`

- 'ifelse()': Conditional element selection.

`dplyr`

- `case_when()`: A general vectorized if.

8.2 Environment and Workspace

`base`

- `dir()`: List the files in a directory/folder.
- Environments
  - `baseenv()`: The environment of the `base` package, it's enclosing environment ("parent environment") is the empty environment.
  - `emptyenv()`: The empty environment, which is the ancestor of all environments and the only environment without an enclosing environment.

  - `environment()`: The current environment.
  - `globalenv()`: The environment in which you normally work, it's enclosing environment is the last package attached with `library()` or `require()`.
  - `new.env()`: Create a new environment.

- `exists()`: Look for an R object of the given name and possibly return it.
  - Must use quotations to name the object.
- `getwd()`: Get the working directory.
- `list.files()`: List the files in a directory/folder.
- `ls()`: List objects in the specified environment.
- `options()`: Set and examine global options.
  - `getOption()`: Set and examine global options.
- `rm()`: Remove objects from a specified environment.
- `search()`: Return a list of attached packages and R objects.
  - `searchpaths()`: Return the path to attached packages.

`gdata`

- `object.size()`: Report the space allocated for an object.
  - See also `utils::object.size()`.

`installr`

- `updateR()`: Check for the latest R version; downloads and installs new R versions.

`pryr`

- `where()`: Find where a name is defined.

`utils`

- `object.size()`: Report the space allocated for an object.
  - See also `gdata::object.size()`.

References:

- "Environments" (Hadley Wickham, *Advanced R*)

8.3 Evaluation (Standard and Non-standard)

`base`

- `cat()`: Concatenate and print.
- `quote()`: Return the argument, unevaluated.
- `writeLines()`: Display quotes and backslashes as they would be read, rather than as R stores them (i.e., see the raw contents of the string, as the `print()` representation is not the same as the string itself).

`rlang`

- Quosures
  - `enquo()`, `new_quosure()`, `quo()`.

References:

- "Non-standard evaluation" (Hadley Wickham, *Advanced R*)
- "Non-standard evaluation" (Hadley Wickham, `lazyeval` package vignette)
- "Programming with dplyr" (dplyr.tidyverse.org)

8.4 Functionals

`base`

- Apply Functions
  - `apply()`: Apply functions over array margins.
  - `lapply()`: Apply a function over a list or vector.
  - `sapply()`: Apply a function over a list or vector and return a vector or matrix.
  - `vapply()`: A safer version of `sapply()`, as it requires the output type to be predetermined.
  - `mapply()`: Apply a function to multiple list or vector arguments.
  - `rapply()`: Recursively apply a function to a list.
  - `tapply()`: Apply a function over a ragged array.

`purrr`

- `map()`: Apply a function to each element of a vector.
- `map2()`: Map over multiple inputs simultaneously.
- `safely()`: Capture side effects.
- `transpose()`: Transpose a list (turn a list-of-lists inside-out).

8.5 Functions

`base`

- `do.call()`): Execute a function call from a name or a function and a list of arguments to be passed to the function.
- `message()`: Generage a diagnostic message.
- `unlist()`: Flatten lists.
  - Useful when using `purrr`'s `map()` functions, which return objects as type `list`.

8.6 Learn About an Object

`base`

- `args()`: Display the argument names and default values of a function.
- `attributes()`: View or assign an objects attributes (e.g., `class()`, `dim()`, `dimnames()`, `names()`, `row.names()`).
- `body()`: Get or set the body of a function.
- `colnames()`: Retrieve or set column names.
- `dim()`: Retrieve or set the dimnames of an object.
- `dimnames()`: Retrieve or set the dimension names of an object.
- `formals()`: Get or set the formal arguments of a function.
- `help()`: Get the topic documentation.
- `vignette()`: View a specified package vignette.
- ?object_name
- ??object_name
- `rownames()`: Retrieve or set row names.

8.7 Optimization

`microbenchmark`

- `microbenchmark()`: Sub-millisecond accurate timing of expression evaluations.
  - A more accurate replacement of `system.time(replicate(1000, expr))`.

8.8 Pipes

`magrittr`: Forward-pipe operator for R.

- `freduce()`: Apply a list of functions sequentially.

- `%<>%`: Compound assignment-pipe operator.
- `%>%`: Forward-pipe operator.
- `%$%`: Expositions-pipe operator.

8.9 Selecting & Subsetting

`base`

- `subset()`: Subset vectors, matrices, and data frames.

`dplyr`

- `first()`: Extract the first element of a vector.
- `last()`: Extract the last element of a vector.
- 'nth(): Extract the nth element of a vector.
- `select()`: Select/rename variables.
  - Helper functions include: `starts_with()`, `ends_with()`, `contains()`, `matches()`, `num_range()`, and `one_of()`.
  - A closely-related function is `dplyr::rename()`.

References:

- "Indexing lists in #rstats. Inspired by Residence Inn" (Hadley Wickham, Twitter, 14 September 2015)

8.10 Version Control

Git

- Git
- *Pro Git* by Scott Chacon and Ben Straub
- *Git and GitHub* by Hadley Wickham
- *Happy Git and GitHub for the useR* by Jenny Bryan

`packrat`

- See note on the `packrat` package in the "Referenced Packages" section.
- `snapshot()`: Capture and store the packages and versions in use.
- `restore()`: Load the most recent snapshot to the project's private library.

Chapter 9

Referenced Packages

9.1 A-D

`anytime`: Date converter.

`base`: Base R functions.

`bookdown`: Author books and technical documents with R Markdown. + See Yihui Xie's *bookdown: Authoring Books and Technical Documents with R Markdown.*

`broom`: Convert statistical analysis objects into tidy data frames.

`chron`: Chronological objects which can handle dates and times.

`data.table`: For large data.

`DBI`: Database interface.

- For MySQL documentation, see the MySQL Reference Manual.
- Use with the `odbc` package.

`diagram`: Visualize simple graphs (networks); create plot flow diagrams.

`DiagrammeR`: Graph/network visualization. + DiagrammeR uses the GraphViz language.

`dplyr`: Data manipulation.

- See also dplyr.tidyverse.org.

9.2 E-H

`fasttime`: Fast utilit function for time parsing and conversion.

`forcats`: Tools for working with categorical variables.

`gdata`: Data manipulation.

`ggplot2`: Create elegant data visualizations.

`graphics`: R functions for base graphics.

`hms`: Times without dates.

`httr`: Tools for working with HTTP.

9.3 I-L

`installr`: Install and update stuff (such as R, Rtools, Rstudio, Git).

`jsonlite`: A robust, high performance JSON parser and generator.

`knitr`: Dynamic report generation in R using Literate Programming techniques. + See Yihui Xie's *knitr*.

`lubridate`: Functions to work with date-times and time-spans.

- `lubridate` uses character formatting similar to `strptime()`, though there are some differences. To see `lubridate`'s formatting, type `?parse_date_time` into the R Console.

9.4 M-P

`magrittr`: Forward-pipe operator for R.

`methods`: Formal methods and classes.

`microbenchmark`: Measure and compare the execution time of R expressions.

`odbc`: Connect to ODBC compatible databases using the DBI Interface.

`packrat`: Manage and document the versions of packages used in an R program.

- `packrat` tends to cause more trouble than it prevents, so avoid using it unless necessary or until it is improved.

`pryr`: Tools to pry back the covers of R and understand the language at a deeper level.

`purrr`: Functional programming tools.

- See also purr.tidyverse.org.

9.5 Q-T

`readr`: Read rectangular text data.

- See also readr.tidyverse.org.

`readxl`: Read Excel files.

`rjson`: Convert between R and JSON objects.

`rlang`: Functions for base types and Core R and Tidyverse features.

`RMarkdown`: Save and execute code; generate high quality reports.

- See also "R Markdown: The Definitive Guide").

`scales`: Scale functions for visualization.

`shiny`: Web application framework.

`splitstackshape`: Stack and reshape datasets after splitting concatenated values.

`stats`: Statistical functions.

`stringr`: Working with strings.

- See also stringr.tidyverse.org).

`tibble`: Simple data frames with stricter checking and better formatting than the traditional data frame.

`tidyr`: Tidy data.

- See also tidyr.tidyverse.org.

`tinytex`: Compile LaTeX Documents. + Required to compile and build a `bookdown` book. + See https://yihui.name/tinytex/

9.6 U-Z

`utils`: Various programming utilities.

`XLConnect`: Read, write, and format Excel data.

`xts`: Provide for uniform handling of R's different time-based data classes by extending zoo.

`zoo`: For regular and irregular time series.

Chapter 10

References & Resources

- For an introduction to the R programming language, see the R Project for Statistical Computing's "What is R?" and Wikipedia's "R (programming language)."
- To download R, go to r-project.org and choose the cloud CRAN Mirror option.
- To program in the R language on a user-friendly platform, download the RStudio IDE.
- The R Project for Statistical Computing
  - Library of R Packages
  - *Getting Help with R*
  - *The R Manuals*
  - *Frequently Asked Questions*
  - *Books Related to R*
  - *Documentation*
- RStudio
  - *RStudio Cheat Sheets*
  - *Webinars and Videos On Demand*
  - *Online learning*
  - *RStudio Blog*
- Online Manuals
  - *R for Data Science*
  - *Advanced R* by Hadley Wickham
    - Hadley's second edition draft is available here.
  - *R Packages* by Hadley Wickham
  - *The tidyverse style guide*
  - *Efficient R Programming*
- Other Online Resources
  - DataCamp
  - RDocumentation
  - R Bloggers
    - "Tutorials for learning R"