1 R Programming

List

- `[[idx]]`: get element in a list
- `str(ls)`: get **str**ucture of a list (similar to summary)
- `saveRDS` and `loadRDS`

Recycling Rule

- shorter vectors are recycled until they match the length of the longest vector
- the length of the longest vector must be a multiple of the shorter vector in arithmetic operations!

Useful functions

- `sample(x, size, replace, prob)`
  - `size`: length of output vector
  - `replace`: if `TRUE`, then sampling is with replacement
  - `prob`: a vector of probability weights
- `rep(x, times, length.out)`
- `table()`
- `args(func)`: list the arguments of a function
- `seq(from, to, by, length)`
- `paste(v1, v2, sep)`: concatenate vectors after converting them to characters
  - `sep`: separator between elements of `v1` and `v2`
  - The recycling rule applies when `length(v1) != length(v2)`
- apply function family: apply function to each row (1) or column (2)
  - `apply(X, margin, func, ...)`
    * Note that `X` must be a **matrix** or **df** in apply
  - sapply returns a vector or a matrix, **input must be 1 dimensional!**
  - `lapply` returns a list, useful when the output of the function may not be all of the same length/type, **input must be 1 dimensional!**
  - `replicate(n, func)`: replicate anonymous function n number of times (especially useful for random number generations)

Function debugging

- `cat("...")`: used to print statements
- `browser()`: debugging with breakpoint

Important classes

Strings

- Start by importing `tidyverse` and `stringr`
- Library functions
  - `str_length`: returns vector of string lengths
  - `str_c(..., sep)`: concatenate strings with optional separator
  - `str_sub(string, start, end)`: returns vector of substrings
- Regular expressions (`str_view()` to test out regex), *Tidyverse Article*
  - to match an **a** at the beginning of a string
    `str_view(x, "^a")`
  - to match an **a** at the end of a string
    `str_view(x, "a$")`
  - to match an **a** or **e** at the end of a string
    `str_view(x, "[ae]$")`
  - to match a string of 3 chars with **a** in the middle
    `str_view(x, ".a.")`
- `str_detect(vec, regex)`: returns a boolean vector
- `str_extract(vec, regex)`: returns a vector of strings, particularly helpful for `".a."` regex

Factors

`factor(vec, levels=c(...))`: convert `vec` to factors with fixes levels
`unique(vec)`: returns a vector with unique values

Date

- `as.Date(x, format)`: convert string x to Date object
  e.g. `as.Date("2014/02/22", "%Y/%m/%d")`
- `months(d)`: what month of the year is the date in?
- `weekdays(d)`: what day of the week is the date on?
- `Sys.Date()`
- `cut(x, breaks, labels)`: usually used to group dates that fall into a month/week/quarter
  - `breaks`: numeric vector/string (`"month"`, `"week"`)
  - `labels`: if `TRUE`, return a label vector

Basic Plotting

`plot()`

- `pch`: abbr. for plotting character

```r
# show all pch characters
example(pch)
```

- `col`:

```r
# show all preset colours
colours()
# set custom colour, alpha is
  transparency
col <- rgb(..., alpha=?)
```

- `cex`: abbr. for character expansion
- `bty`: change box borders
- **!!** `?par` shows all parameters for `plot()`
- use `points()` or `lines()` to add more stuff to an existing plot

`barplot()`

`hist()`

- `freq`: makes the y-axis a proportion of all the total shit (count/total), not total count using integer

2 Importing Data

CSV Files

`read.csv()`: main arguments:

- `file`: filename/path
- `skip`: skip lines?
- `header`: default is `TRUE`
- `row.names`
- `stringsAsFactors`
- `na.strings`: what are the `NA` values
- `colClasses`: what classes are the columns (in terms of class names vector)

Procedure when dealing with CSV:

- `apply(salaries, 2, function(x) sum(is.na(x)))` (check if any column has missing values)
- if `read.csv` doesn't work, can try `readLines` and `str_split` to split commas

Excel Files

- import `readxl`, data is in the form of a tibble
- `read_excel`