Strings and Dates in R

Sean Hellingman ©

Data Visualization and Manipulation through Scripting (ADSC1010)

shellingman@tru.ca

Fall 2024

THOMPSON RIVERS UNIVERSITY

Topics

Introduction

- **Strings** or character strings are a commonly used data class in R.
- So far, we have used string values but have not learned how to manipulate them.
- Use strings as both variables and labels.
- Strings and dates behave a bit differently in R.

Length of a String

- The `nchar()` function returns the length of a string.
    - `nchar("ADSC1010")`
    - `nchar("Hello, World!")`
- The `length()` function will not work for this.
    - `length("ADSC1010")`
    - `length("Hello, World!")`

Concatenating Strings I

- The `paste()` function allows you to combine two or more strings.
  - `paste("Everybody", "loves", "Data Science.")`
  - `paste("Everybody", "loves", "Data Science.", sep = "-")`
  - `paste("Everybody", "loves", "Data Science.", sep = "")`
- The `paste()` function will convert non-string arguments to strings.
  - `paste("Four plus four is:  ", 6)`

Concatenating Strings II

- The `paste()` function will generate all combinations of the arguments if one or more of those arguments are vectors.
  - `Students <- c("Cox", "Gauss", "Bayes", "Fisher")`
  - `paste(Students, "Loves", "Data Science.")`

- We can also create one big string:
  - `paste(Students, "Loves", "Data", "Science", sep=" ", collapse = " ")`

Example 1

- Write a function that concatenates two strings if their combined length is over 10 characters.
- If their combined length is 10 or fewer characters your function should return: `"Combine them yourself, they are too short"`

Extracting Substrings

- Use `substr(string, start, end)` to extract the substring that begins at `start` and ends at `end`.
  - `substr("Statistics", 1, 5)` extracts the first 5 characters
  - `substr("Statistics", 7, 10)` extracts the last 4 characters
- Can apply on a vector, will be applied to every string of a given vector:
  - `str_vec <- c("Statistics", "Mathematics", "Science")`
  - `substr(str_vec, 1, 3)`

Splitting a String with a Delimiter

- A string can be split into substrings if there are delimiters within the string.
- Use the `strsplit()` function.
  - `temp.str <- "Three times two equals to 6"`
  - `strsplit(temp.str, " ")`

  - `path <- ".../home/data/ADSC1010/project.csv"`
  - `strsplit(path, "/")`
- It is split into a list object.

Replacing Substrings

- Within a string, we can replace one substring with another using the `sub()` or `gsub()` function.
  - `sub(old, new, string)` replaces the first instance of a substring.
  - `gsub(old, new, string)` replaces the all instances of a substring.
- Example:
  - `s <- "Data science is fun.  Data science is very useful."`
  - `sub("Data science", "Statistics", s)`
  - `gsub("Data science", "Statistics", s)`

Example 2

- Create the following string: `"I study at TRU and I am excited for the winter."`
- Conduct the following modifications on your string:
  - Split the string into two strings that form sentences. *Hint*: use `nchar()` to give you lengths of the substrings you want.
  - Replace you word *winter* with *summer*.
  - Combine the strings back together but in opposite order `"I am excited for the summer and I study at TRU."`

Combinations

- If we have two sets of strings and want to generate all combinations from the two sets use the `outer()` function.
- This will generate a matrix of all possible combinations:
  - `m <- outer(string1, string2, paste, sep="")`
- Example:
  - `locations <- c("NY", "LA", "CHI", "HOU")`
  - `treatments <- c("T1", "T2", "T3")`
  - `outer(locations, treatments, paste, sep="-")`
  - `outer(locations, locations, paste, sep="-")`

Dates in R

Introduction

- Date objects often resemble strings in R but behave differently.
- We can store date values as character strings.
- Used in many different fields.

Today's Date

- Use the following command to get the current date (today's date):
  - `Sys.Date()`

- Be sure to know the difference between a string and a date. Although sometimes they appear the same.
  - `today <- Sys.Date()`
  - `class(today)`

  - `today.1 <- "2024-10-11"`
  - `class(today.1)`

Convert a Date into a String

- When we want to print the date, we need to convert a Date object into a character string.
  - `format(Sys.Date())`
  - `as.character(Sys.Date())`
- Both functions allow a `format` argument that controls the formatting, for example:
  - `format(Sys.Date(), format="%m/%d/%Y")`
- See the help page for the `strftime` function for a complete list of formatting codes.

Convert a String to a Date

- When working with real world data, for example, daily stock prices, we will need to use date object.
  - Often the dates in the data appear to be represented as a date object.
  - They may just be string values, such as `"2010-12-31"`.
- Need to convert the strings to date object using `as.Date`.
  - `as.Date("2010-12-31")`
- If the string is in other format, such as `mm/dd/yyyy`, we must provide a `format` argument:
  - `as.Date("12/31/2010")` will produce an error.
  - `as.Date("12/31/2010", format="%m/%d/%Y")` will return `"2010-12-31"`.

Merging into Date Objects

- If we have a date represented by its year, month, and day separately, the functions below can merge these elements into a Date object.
  - `ISOdate(year, month, day)` prints both date and time.
  - `as.Date(ISOdate(year, month, day))` only prints out date.
- Will result in `NA` if the date is not valid:
  - `as.Date(ISOdate(2013,2,29))`
- Can also specify hour, minute, and second:
  - `ISOdate(year, month, day, hour, minute, second)`

Example 3

- Use the following vectors to generate date objects in R:
  - `years <- c(2010, 2011, 2012, 2013, 2014)`
  - `months <- rep(1,5)`
  - `days <- c(3, 5, 8, 10, 23)`

Extracting Parts of Dates

- Extract a date part such as the day of the week or year, the calendar day or month, etc.
  - `d <- as.Date("2023-07-24")`
  - `p <- as.POSIXlt(d)`
  - `POSIXlt` is a class that stores date and time information.

`POSIXlt` Dates

- `p$sec` *prints the second*
- `p$min` *prints the minute*
- `p$hour` *prints the hour*
- `p$mday` *prints day of the month*
- `p$mon` *prints the month (0 represents January)*
- `p$year + 1900` *prints the year*
- `p$wday` *prints day of the week*
- `p$yday` *prints day of the year*

Sequence of Dates

- You can generate a sequence of dates using the `seq()` function.
- Example:

```
start_date <- as.Date("2023-09-01")
end_date <- as.Date("2023-09-10")
date_sequence <- seq(from = start_date, to = end_date, by
= "day")
```

Differences in Dates

- You can calculate the differences in dates using the `difftime()` function.
- Example:

```
date1 <- as.Date("2023-09-15")
date2 <- as.Date("2023-09-20")
date_diff <- difftime(date2, date1, units = "days")
```

Exercise 1

- Write a function in R that splits a character string in half (you will have to contend with even and odd lengths) and re-orders the two halves.
- For example:
  - `"Hello ADSC1010"` becomes `"DSC1010hello A"`

Exercise 2

- Recall the following example from ADSC1000 (Intro. to experimental design):
  - An experiment was carried out on 18 patients to determine the effect of gold alloys and the sintering process on the hardness (measured by Diamond Pyramid Hardness) of dental fillings.
  - Two gold alloys were used: Au 97-1-1-1 and AuCa.
  - Fillings were sintered (fused and hardened) at three different temperatures: 1500° F, 1600° F, and 1700° F.
- **Use R to generate all possible treatment combinations.**

Exercise 3

- Use the following vectors to generate date objects in R:
    - `years <- c(2014, 2015, 2016, 2017, 2018)`
    - `months <- rep(1,5)`
    - `days <- c(3, 5, 8, 10, 23)`
- Calculate the differences between at least two of the dates in days.
- Create a vector of just the month and year dates.

References & Resources

- Strings
- POSIXlt
- Dates and Times
- lubridate