SIOB 296 Introduction to Programming with R

Eric Archer (eric.archer@noaa.gov)

Week 05: February 3, 2020

string manipulation, date/time objects

Character and string manipulation

paste

To create strings from combinations of strings (or numbers) we use `paste()`. This function takes a set of vectors, and pastes the elements together using recycling:

```
# vectors are equal length
paste(letters[1:6], 1:6)
```

```
[1] "a 1" "b 2" "c 3" "d 4" "e 5" "f 6"
```

```
# one vector is a multiple of the other
paste(letters[1:6], 1:2)
```

```
[1] "a 1" "b 2" "c 1" "d 2" "e 1" "f 2"
```

```
# one vector is not a multiple of the other
paste(letters[1:6], 1:4)
```

```
[1] "a 1" "b 2" "c 3" "d 4" "e 1" "f 2"
```

The argument `sep` determines what character is used as a separator between the characters:

```
paste(letters[1:6], 1:2, sep = "-")
```

```
[1] "a-1" "b-2" "c-1" "d-2" "e-1" "f-2"
```

If you do not want a separator character, either set `sep = ""` or use `paste0()`:

```
paste0(letters[1:6], 1:2)
```

```
[1] "a1" "b2" "c1" "d2" "e1" "f2"
```

If you want to paste all of the arguments to create a single element vector, set the `collapse` argument:

```
paste(letters[1:6], 1:2, sep = "-", collapse = "#")
```

```
[1] "a-1#b-2#c-1#d-2#e-1#f-2"
```

nchar

A character vector is a vector where every element is a character string of any length. The `length()` of a character vector is the number of elements in it:

```r
x <- c("This is a sentence", "Hello World!", "This is the third element")
length(x)
```

```
[1] 3
```

To get the number of characters in each element, use `nchar()`:

```r
nchar(x)
```

```
[1] 18 12 25
```

substr

Strings can be extracted from elements using `substr()`. You specify the first and last characters to be extracted from each string:

```r
# get the first three characters from every string
substr(x, 1, 3)
```

```
[1] "Thi" "Hel" "Thi"
```

```r
# get the 3rd character from every string
substr(x, 3, 3)
```

```
[1] "i" "l" "i"
```

`substr` can also be used to replace values within strings by assigning:

```r
substr(x, 1, 4) <- "That"
x
```

```
[1] "That is a sentence"        "Thato World!"
[3] "That is the third element"
```

strsplit

Strings can be split based on some common delimiter using `strsplit()`:

```r
# split based on spaces
x.split <- strsplit(x, " ")
x.split
```

```
[[1]]
[1] "That"     "is"       "a"        "sentence"

[[2]]
[1] "Thato"  "World!"

[[3]]
[1] "That"    "is"      "the"     "third"   "element"
```

```r
str(x.split)
```

```
List of 3
 $ : chr [1:4] "That" "is" "a" "sentence"
 $ : chr [1:2] "Thato" "World!"
 $ : chr [1:5] "That" "is" "the" "third" ...
```

Note that the return value from `strsplit` is a list. Each element in the list corresponds to a vector resulting from splitting every element in the orginal vector

```
x.split[[1]]
```

```
[1] "That"     "is"       "a"        "sentence"
```

tolower, toupper

Character case can be changed with `tolower` and `toupper`:

```
tolower(x)
```

```
[1] "that is a sentence"        "thato world!"
[3] "that is the third element"
```

```
toupper(x)
```

```
[1] "THAT IS A SENTENCE"        "THATO WORLD!"
[3] "THAT IS THE THIRD ELEMENT"
```

Regular Expressions

For finer control on searching and replacing text within strings, you will have to turn to "regular expressions", which is a kind of syntax of its own and is common across several platforms. The help page for regular expressions in R is `?regex`. The functions that are most commonly used with regular expressions are given in `grep`. The most commonly used on this page are:

`grep` and `grepl`: Identify elements that have the sought after pattern `sub` and `gsub`: Replace a desired pattern with other text

```
x <- c("Here is some text", "This is more text", "I have the number 1", "22 is the number I have")
# which elements have the word "text"?
grep("text", x)
```

```
[1] 1 2
```

```
# which elements have numbers?
grep("[[:digit:]]", x)
```

```
[1] 3 4
```

```
# replace the word "This" with "That"
gsub("This", "That", x)
```

```
[1] "Here is some text"       "That is more text"
[3] "I have the number 1"     "22 is the number I have"
```

Dates

Dates in base R are usually represented by classes related to `POSIXxt` (Portable Operating System Interface for uniX). There are other packages and classes for manipulating dates/times (`chron`, `date`, `gdata`, `lubridate`, `stringi`), but `POSIXxt` dates and functions are sufficient for most uses.

`POSIXlt` stores dates in a list and is good for easily working with components of the dates:

```r
dt.lt <- as.POSIXlt("2011/08/23 6:05")
is.list(dt.lt)
```

```
[1] TRUE
```

```r
str(dt.lt)
```

```
 POSIXlt[1:1], format: "2011-08-23 06:05:00"
```

```r
dt.lt$mon
```

```
[1] 7
```

```r
dt.lt$year
```

```
[1] 111
```

`POSIXct` stores dates as characters and is used for storing dates in data frames:

```r
dt.ct <- as.POSIXct("2011/08/23 6:05")
str(dt.ct)
```

```
 POSIXct[1:1], format: "2011-08-23 06:05:00"
```

```r
dt.ct
```

```
[1] "2011-08-23 06:05:00 PDT"
```

Note that you cannot access directly access elements of a POSIXct date:

```r
dt.ct$mon
```

```
Error in dt.ct$mon: $ operator is invalid for atomic vectors
```

If we have the components of dates and times as different vectors, we can create `POSIXct` representations with `ISOdatetime` or `ISOdate`:

```r
dt.iso <- ISOdatetime(2011, 8, 23, 6, 5, 0)
str(dt.iso)
```

```
 POSIXct[1:1], format: "2011-08-23 06:05:00"
```

```r
dt.iso <- ISOdate(
  year = rep(2011, 4),
  month = c(1, 2, 3, 4),
  day = rep(c(15, 20), 2)
)
dt.iso
```

```
[1] "2011-01-15 12:00:00 GMT" "2011-02-20 12:00:00 GMT"
[3] "2011-03-15 12:00:00 GMT" "2011-04-20 12:00:00 GMT"
```

If the date is a formatted character string, we convert it to `POSIXlt` with `strptime`. We have to use a character string that details the way the formatting is to be read. The codes for this string can be found in the Details section of `?strptime`.

```r
xmas <- strptime("12/25/2019", format = "%m/%d/%Y")
xmas
```

```
[1] "2019-12-25 PST"
```

We can convert from POSIXlt to character using `strftime`:

```r
xmas.ct <- strftime(xmas, format = "%Y-%m-%d %H hours, %M minutes")
str(xmas.ct)
```

```
 chr "2019-12-25 00 hours, 00 minutes"
```

```r
xmas.ct
```

```
[1] "2019-12-25 00 hours, 00 minutes"
```

Alternatively, we can use `format` to convert either `POSIXxt` to a character vector:

```r
format(dt.ct, "Year: %Y, Month: %m, Day: %d at %H%M", tz = "EST")
```

```
[1] "Year: 2011, Month: 08, Day: 23 at 0805"
```

Today's date and time can be reported as a `POSIXct` object with `Sys.time()`:

```r
today.ct <- Sys.time()
today.ct
```

```
[1] "2020-01-28 09:16:08 PST"
```

See `?Sys.timezone` for help on setting timezones (usually the `tz` argument of a function).

The difference between two date/time objects can also be computed.

```r
days2go <- xmas - today.ct
str(days2go)
```

```
 'difftime' num -34.3862045396285
 - attr(*, "units")= chr "days"
```

```r
days2go
```

```
Time difference of -34.3862 days
```

Note that this creates an `difftime` object and displays the difference using an automatically selected set of units. Time differences can be explicitly selected with the `difftime()` function:

```r
sec.of.year <- difftime(today.ct, as.POSIXct("2019/1/1"), units = "secs")
str(sec.of.year)
```

```
 'difftime' num 33902168.0722239
 - attr(*, "units")= chr "secs"
```

```r
sec.of.year
```

```
Time difference of 33902168 secs
```

The units of a `difftime` object can be extracted:

```r
units(sec.of.year)
```

```
[1] "secs"
```

The units can also be converted:

```r
units(sec.of.year) <- "weeks"
str(sec.of.year)
```

```
 'difftime' num 56.0551720770898
 - attr(*, "units")= chr "weeks"
```

```r
sec.of.year
```

```
Time difference of 56.05517 weeks
```

Time units can also be added to a date:

```r
dt.ct + as.difftime(2, units = "weeks")
```

```
[1] "2011-09-06 06:05:00 PDT"
```

```r
dt.ct + as.difftime(3.5, units = "hours")
```

```
[1] "2011-08-23 09:35:00 PDT"
```

Finally, there are a handful of functions for extracting various values from dates:

```r
weekdays(dt.iso)
```

```
[1] "Saturday"  "Sunday"    "Tuesday"   "Wednesday"
```

```r
months(dt.ct)
```

```
[1] "August"
```

```r
quarters(xmas)
```

```
[1] "Q4"
```

```r
julian(today.ct, "2011-1-1")
```

```
Time difference of 3314.386 days
```