UiO : PROMENTA
Det samfunnsvitenskapelige fakultet

Working with Character Variables

E. F. Haghish

25/8/2022

Main idea

- Understanding character variables
- Learn important printing functions
  - `paste()`, `cat()`, `paste0()`
- Learn to manipulate character data
  - Using R base
    - `substring()`, `strsplit()`, `...`
  - Using `stringr` package

Displaying and Concatenating Character Strings

- you can display a character object by typing the name in the console
- you can also display an object using the `print`, `paste`, or `cat` function

`cat()`

- the `cat` function Concatenates and Prints outputs
- the `cat` function can include "`\n`" for adding an empty line
- the `cat` function can write text to a file using `file=` argument
- `cat` often used for creating output in functions

```r
a = 100
b = 200
cat("The value of a is", a, "and the value of b is", b,"\n")
```

```
## The value of a is 100 and the value of b is 200
```

- in contrast to `print` and `paste`, what is returned from `cat` is not assignable by default

```r
a = print("hi")
b = paste("hi")
d = cat("hi")
```

```r
a
```

```
## [1] "hi"
```

```r
b
```

```
## [1] "hi"
```

```r
d
```

```
## NULL
```

paste() function

- If any object passed to paste is not of mode character, it is converted to character

```
paste('one', 2, 'three', 4, 'five')
```

```
## [1] "one 2 three 4 five"
```

```
paste('one', 2, 'three', 4, 'five', sep = ";")
```

```
## [1] "one;2;three;4;five"
```

`paste0()` function

- does not add empty space between the elements

```
paste0('one', 2, 'three', 4, 'five')
```

```
## [1] "one2three4five"
```

it is equivilant of setting `sep = ""` in `paste()` function:

```
paste('one', 2, 'three', 4, 'five', sep = "")
```

```
## [1] "one2three4five"
```

- you can `collapse=` a character vector to change what appears between the members of the elements after concatenation

```
(a = paste("my vector", c('one','two','three','four')))
```

```
## [1] "my vector one"   "my vector two"   "my vector three" "my vector fo
```

```
length(a)
```

```
## [1] 4
```

```r
(a = paste("my vector", c('one','two','three','four'), collapse=';'))
```

```
## [1] "my vector one;my vector two;my vector three;my vector four"
```

```r
length(a)
```

```
## [1] 1
```

- When multiple arguments are passed to paste, it will vectorize the operation

```r
paste('X', 1:5, sep='')
```

```
## [1] "X1" "X2" "X3" "X4" "X5"
```

```r
paste(c('X','Y'), 1:5, sep='')
```

```
## [1] "X1" "Y2" "X3" "Y4" "X5"
```

Working with Parts of Character Values

- we can also access words or single characters in text
- the `substring()` function can be used either to extract parts of character strings, or to change the values of parts of character strings.
- check out **`help(substring)`**
- it can operate with vectors

```
state.name #names of US states, stored in R
```

```
##  [1] "Alabama"        "Alaska"         "Arizona"        "Arkansas"
##  [5] "California"     "Colorado"       "Connecticut"    "Delaware"
##  [9] "Florida"        "Georgia"        "Hawaii"         "Idaho"
## [13] "Illinois"       "Indiana"        "Iowa"           "Kansas"
## [17] "Kentucky"       "Louisiana"      "Maine"          "Maryland"
## [21] "Massachusetts"  "Michigan"       "Minnesota"      "Mississippi"
## [25] "Missouri"       "Montana"        "Nebraska"       "Nevada"
## [29] "New Hampshire"  "New Jersey"     "New Mexico"     "New York"
## [33] "North Carolina" "North Dakota"   "Ohio"           "Oklahoma"
## [37] "Oregon"         "Pennsylvania"   "Rhode Island"   "South Carolina"
## [41] "South Dakota"   "Tennessee"      "Texas"          "Utah"
## [45] "Vermont"        "Virginia"       "Washington"     "West Virginia"
## [49] "Wisconsin"      "Wyoming"
```

```
substring(state.name,1,3)
```

```
##  [1] "Ala" "Ala" "Ari" "Ark" "Cal" "Col" "Con" "Del" "Flo" "Geo" "Haw" "Ida"
## [13] "Ill" "Ind" "Iow" "Kan" "Ken" "Lou" "Mai" "Mar" "Mas" "Mic" "Min" "Mis"
## [25] "Mis" "Mon" "Neb" "Nev" "New" "New" "New" "New" "Nor" "Nor" "Ohi" "Okl"
```

can you explain what is happenning here?

```r
mystring = 'dog cat duck'
substring(mystring, c(1,5,9), c(3,7,12))
```

```
## [1] "dog"  "cat"  "duck"
```

- you can do the same using the `strsplit()` function

```r
strsplit("break this string based on the defined separator", split = " ")
```

```
## [[1]]
## [1] "break"     "this"      "string"    "based"     "on"        "the"
## [7] "defined"   "separator"
```

`nchar()` vs `length()`

Note:

- `length()` returns the number of elements in a vector
- `nchar()` returns the number of character in an element

- you can also break the characters using the same methods

```
a = "my name"
strsplit(a, split = "")
```

```
## [[1]]
## [1] "m" "y" " " "n" "a" "m" "e"
```

- alternatively, I can:

```
n = nchar(a)
(b = substring(a, 1:n, 1:n))
```

```
## [1] "m" "y" " " "n" "a" "m" "e"
```

- use the `which` function to get a particular character

```
which(b == 'm')
```

```
## [1] 1 6
```

Manipulate a character

- changing a character vector is simple

```
a = c("dog","cat","duck")
a[2] = "rat"
a
```

```
## [1] "dog"  "rat"  "duck"
```

- you can change a part of the string using `substring`

```
mystring = 'dog cat duck'
substring(mystring,5,7) = 'rat'
mystring
```

```
## [1] "dog rat duck"
```

`strsplit` function

- divides character string into smaller pieces
- it returns a list

```
sentence = 'R is a free software environment for statistical computing'
(parts = strsplit(sentence,' '))
```

```
## [[1]]
## [1] "R"           "is"          "a"           "free"          "software"
## [6] "environment" "for"         "statistical" "computing"
```

- To access the results, the first element of the list must be used

```
length(parts)
```

```
## [1] 1
```

```
length(parts[[1]])
```

```
## [1] 9
```

`sapply` function

- very useful for repeating a function on several inputs:
- it’s the simple version of `lapply` function
- the syntax is `sapply(x, function, ...)`

example

run `sapply` on a character vector and evaluate the length of each vector

```
a = c("my name", "some text goes here", "the thirs character input")
sapply(a, nchar)
```

```
##                   my name       some text goes here the thirs character
##                         7                        19
```

stringr package

Grammar

- There are 7 main verbs that you need to learn
  - `str_detect()`
  - `str_count()`
  - `str_subset()`
  - `str_locate()`
  - `str_extract()`
  - `str_match()`
  - `str_replace()`

how to do `nchar()`, `concatenate`, and `substring()` with stringr

```r
library(stringr)
x <- c("why", "video", "cross", "extra", "deal", "authority")
str_length(x)
```

```
## [1] 3 5 5 5 4 9
```

```r
str_c(x, collapse = ", ")
```

```
## [1] "why, video, cross, extra, deal, authority"
```

```r
str_sub(x, 1, 2)
```

```
## [1] "wh" "vi" "cr" "ex" "de" "au"
```

Regular expressions

You can search for particular characters or a combination of letters in a string:

```
str_subset(x, "[aeiou]")
```

```
## [1] "video"      "cross"      "extra"      "deal"       "authority"
```

The nicest way to learn regular expressions is `regexplain` package

```
# install.packages("devtools")
devtools::install_github("gadenbuie/regexplain")

# run the Shiny App
regexplain::regexplain_addin()
```

TASK: Let's work with `regexplain`

Explore the RegEx library for detecting numbers, words, etc…