Non-Numeric Values

Pedro Fonseca

Introduction to R

May 9, 2020

Creating character values

- Character strings are used to represent text, and should be inside single or double quotes:

```
> foo <- "hello world"
> foo
[1] "hello world"

> foo2 <- 'hello world'
> foo2
[1] "hello world"
```

Basic functions for characters

- Character strings are used to represent text, and should be inside single or double quotes:

```
> foo <- "hello world"
> foo
[1] "hello world"

> length(foo)
[1] 1

> nchar(foo)
[1] 11
```

Common use of characters in R

- Provide arguments to functions
- Directories
- Create factors
- Create names (vectors, matrices, lists, data frames)

Introduction

- When writing strings, you can insert single quotes in a string with double quotes, and vice versa:

```
# single quotes within double quotes
ex1 <- "The 'R' project for statistical computing"

# double quotes within single quotes
ex2 <- 'The "R" project for statistical computing'
```

- You cannot directly insert single quotes in a string with single quotes, neither you can insert double quotes in a string with double quotes

Introduction

- If you really want to include a double quote as part of the string, you need to escape the double quote using a backslash before it:

  ```
  "The \"R\" project for statistical computing"
  ```

The *paste* and *paste0* functions

```
PI <- paste("The life of", pi)
PI
> [1] "The life of 3.14159265358979"
```

The *paste* and *paste0* functions

```
# paste with objects of the same lengths
IloveR <- paste("I", "love", "R", sep = "-")
IloveR
> [1] "I-love-R"

> paste(c(3, 2, 1), c("a", "b", "c"), sep = "_")
[1] "3_a" "2_b" "1_c"

# paste with objects of different lengths
paste("X", 1:5, sep = ".")
> [1] "X.1" "X.2" "X.3" "X.4" "X.5"
```

The *paste* and *paste0* functions

```
# paste with collapsing
paste(1:3, c("!","?","+"), sep = "", collapse = "")
> [1] "1!2?3+"

> paste(1:3, c("!","?","+"), sep = "$", collapse = "")
[1] "1$!2$?3$+"

# paste without collapsing
paste(1:3, c("!","?","+"), sep = "")
> [1] "1!" "2?" "3+"
```

The *paste* and *paste0* functions

- One of the potential problems with *paste* is that it coerces NAs into the character "NA"

```
# with NA
evalue <- paste("the value of 'e' is", exp(1), NA)

evalue
> [1] "the value of 'e' is 2.71828182845905 NA"
```

The *paste* and *paste0* functions

- In addition to paste(), there's also the function *paste0* which is the equivalent of *paste*(..., sep = "")

```
# collapsing with paste0
paste0("let's", "collapse", "all", "these", "words")
> [1] "let'scollapseallthesewords"

> paste("let's", "collapse", "all", "these", "words")
[1] "let's collapse all these words"
```

The *paste* and *paste0* functions

- *paste* and *paste0* can be useful to generate vector names:

```
> paste("y", 1:length(y), sep = "")
[1] "y1" "y2" "y3"

> paste("name", 1:length(y), sep = "_")
[1] "name_1" "name_2" "name_3"

> paste("year", 1990:1993, sep = "-")
[1] "year-1990" "year-1991" "year-1992" "year-1993"

> paste0("X", 1:5)
[1] "X1" "X2" "X3" "X4" "X5"
```

The *paste* and *paste0* functions

```
> vec <- c("awesome","R","is")
>
> my_opinion <- paste(vec[2],vec[3],"totally",vec[1],"!")
> my_opinion
[1] "R is totally awesome !"
```

The *cat* function

```
> vec <- c("awesome","R","is")

> cat(vec[2],vec[3],"totally",vec[1],"!")
R is totally awesome !
```

- *cat* outputs the object but does not store it nor does it return anything
- Useful to print objects in functions

Operations with characters

- It is not possible to make operations with characters:

```
> zag <- c("23", "4")
> zag * 5
Error in zag * 5 : non-numeric argument to binary operator

> bar <- c("23", "4", "some-random-string")
> length(bar)
[1] 3
> nchar(bar) # number of characters
[1]  2  1 18
> zag[2] # subsetting works as usual
[1] "4"
```

Equality test

```
> "alpha"=="alpha"
[1] TRUE

> "alpha"!="beta"
[1] TRUE

> c("alpha","beta","gamma") == "beta"
[1] FALSE TRUE FALSE

> "beta" %in% c("alpha","beta","gamma")
[1] TRUE
```

Logical comparisons

- Alphabetical order matters:

```
> "alpha"<="beta"
[1] TRUE
> "gamma">"Alpha"
[1] TRUE
```

- Uppercase letters also matters:

```
> "Alpha">"alpha"
[1] TRUE
> "beta">="bEtA"
[1] FALSE
```