Introduction to R Software

Swayam Prabha

Lecture 28

Search in Strings and Other Data Operations

Shalabh

Department of Mathematics and Statistics

Indian Institute of Technology Kanpur

Slides can be downloaded from

http://home.iitk.ac.in/~shalab/sp

Operations with Strings

`sub` and `gsub` Functions:

Within a string, we want to replace one substring with another.

```
sub(old, new, string)
```

The `sub` function finds the first instance of the old substring within string and replaces it with the new substring.

`gsub` does the same thing, but it replaces all instances of the substring (a global replace), not just the first.

```
gsub(old, new, string)
```

Operations with Strings

Examples:

```
>  y <- "Mr. Bhatia is the smart one. Mr.
Bhatia is funny, too."

> sub("Mr. Bhatia ","Professor Bose", y)
[1] "Professor Bose is the smart one. Mr.
Bhatia is funny, too."

> gsub("Mr. Bhatia","Professor Bose", y)
[1] "Professor Bose is the smart one. Professor
Bose is funny, too."
```

Operations with Strings

R has various functions for regular expression based match and replaces.

Some functions (e.g., `grep`, `grepl`, etc.) are used for searching for matches and functions whereas `sub` and `gsub` are used for performing replacement.

`grep(x)` returns a vector of indices of the character strings in `x` that contains the pattern.

`grepl(x)` returns `TRUE` when a pattern is found in the character string `x`.

Operations with Strings

`grep` function:

The `grep` function is used for searching the matches.

(`sub` and `gsub` are used for performing replacement.)

`grep:` Globally search regular expression and print it

`grep(pattern, x)` search for matches to argument `pattern` within each element of a character vector `x`.

It returns an integer vector of the indices of the elements of x that yielded a match

Operations with Strings

`grep(pattern, x, value = FALSE)` returns an integer vector of the indices of the elements of x that yielded a match

`value = FALSE` is default.

```
> str <- c("R Course", "assigments", "include
examples of R language")

> grep("as", str, value=F)
[1] 2

> grep("as", str)
[1] 2
```

Operations with Strings

`grep(pattern, x, value = TRUE)` returns a character vector containing the selected elements of x.

```
> str <- c("R Course", "assigments", "include
examples of R language")

> grep("as", str, value=T)
[1] "assigments"
```

Operations with Strings

R Console

```
> str <- c("R Course", "assigments", "include examples of R language")
> grep("as", str, value=T)
[1] "assigments"
>
> grep("assi", str, value=F)
[1] 2
>
```

Operations with Strings

Example:

```
> x <- "R course 01.06.2020"
> y <- "Number of participants: 50"

> c(x,y)  # Combine the two strings
[1] "R course 01.06.2020"  "Number of
participants: 50"

> grep("our", c(x,y) )
[1] 1
```

"our" is in the 1st element (in the word "course"), therefore in x. There is no "our" in y.

Operations with Strings

Example:

```
x <- "R course 01.06.2020"
y <- "Number of participants: 50"

> c(x,y)  # Combine the two strings
[1] "R course 01.06.2020"  "Number of
participants: 50"

> grep("Num", c(x,y) )
[1] 2
```

"Num" is in the 2nd element (in the word "Number"), therefore in y.

There is no "Num" in x.

Operations with Strings

`grep` function:

R Console

```
> x <- "R course 01.06.2020"
> y <- "Number of participants: 50"
> c(x,y)
[1] "R course 01.06.2020"        "Number of participants: 50"
>
> grep("our", c(x,y) )
[1] 1
>
> grep("Num", c(x,y) )
[1] 2
>
```

Operations with Strings

`grepl` function:

The `grepl` function is used for searching the matches.

`grepl()` searches for matches of certain character pattern in a vector of character strings and returns a logical vector indicating which elements of the vector contained a match.

It returns TRUE when a pattern is found in the character string .

Operations with Strings

`grepl` function:

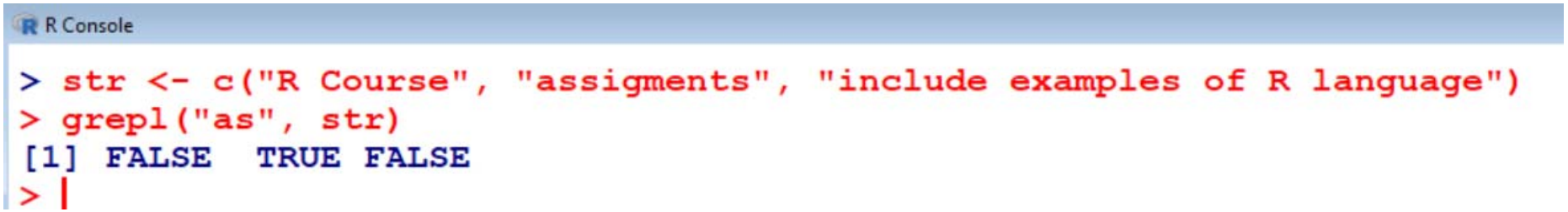
`grepl(pattern, x)` search for matches to argument `pattern` within each element of a character vector `x`.

It returns a logical **TRUE** when a pattern is found in the character string `x`.

Operations with Strings

```
> str <- c("R Course", "assigments", "include
examples of R language")

> grepl("as", str)
[1] FALSE  TRUE FALSE
```



Operations with Strings

Example:

```
x <- "R course 01.06.2020"
y <- "Number of participants: 50"

> c(x,y)  # Combine the two strings
[1] "R course 01.06.2020"  "Number of
participants: 50"

> grepl("Num", c(x,y) )
[1] FALSE  TRUE
```

"Num" is in the 2nd element (in the word "Number"), therefore in y and so it is TRUE

There is no "Num" in x, hence FALSE.

Operations with Strings

`eval` **function:**

`eval` **function evaluates an (Unevaluated) R expression in a specified environment.**

Example:

```
> eval(2 ^ 3 ^ 5)
[1] 256
```

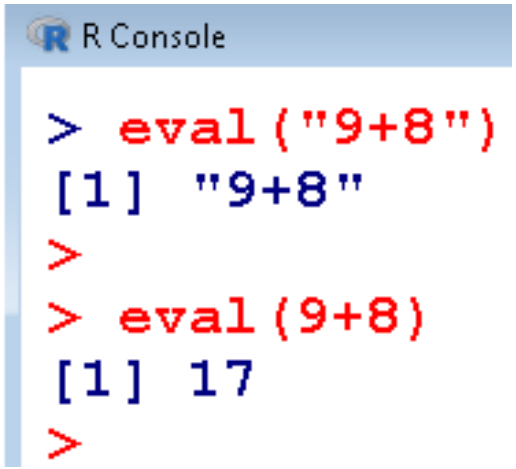
Operations with Strings

`eval` function:

Example:

```
> eval("9+8")
[1] "9+8"
```

```
> eval(9+8)
[1] 17
```



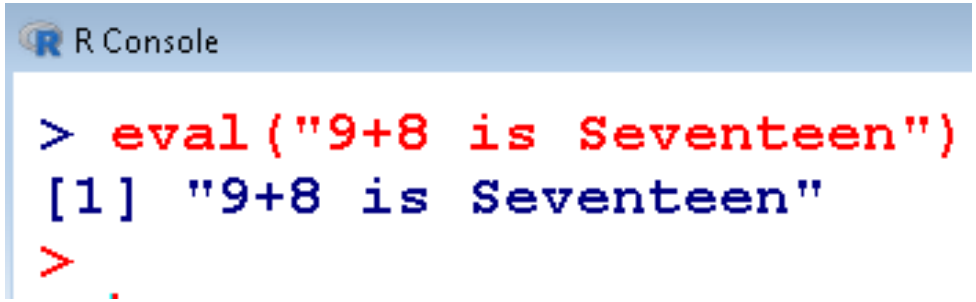
The `eval()` function evaluates an expression, but `"9+8"` is a string, not an expression whereas `9+8` is not an expression.

Operations with Strings

`eval` **function:**

Example:

```
> eval("9+8 is Seventeen" )
[1] "9+8 is Seventeen"
```



Operations with Strings

`parse` function:

`parse()` with `text=string` is used to change the string into an expression.

Example:

```
> eval("9+8")
[1] "9+8"

> mode("9+8")
[1] "character"

> eval(parse(text="9+8"))
[1] 17

> mode(parse(text="9+8"))
[1] "expression"
```

Operations with Strings

`parse` function:

R Console

```
> eval("9+8")
[1] "9+8"
> mode("9+8")
[1] "character"
>
> eval(parse(text="9+8"))
[1] 17
> mode(eval(parse(text="9+8")))
[1] "numeric"
>
```