Foundations of R Software

Lecture 37

Strings – Display and Formatting

:::

Paste Function

Shalabh

Department of Mathematics and Statistics

Indian Institute of Technology Kanpur

Formatting and Display of Strings: `paste`

- The `paste()` function concatenates several strings together.

- It creates a new string by joining the given strings end to end.

- The result of `paste()` can be assigned to a variable

  (in contrast to the function `cat()`).

Formatting and Display of Strings

Usage

```
paste (..., sep = " ", collapse = NULL).
```

`collapse` **is an optional character string to separate the results.**

- **The parameter** `sep` **is a string that serves as a separation between the strings that are given as input.**
- `paste` **inserts a single space between pairs of strings.**
- **A desired line break can be achieved with** `"\n"` **(newline).**

Formatting and Display of Strings: `paste`

- `paste` converts its arguments to character strings (via `as.character`) , and concatenates them (separating them by the string given by `sep`).

- If the arguments are vectors, they are concatenated term-by-term to give a character vector result.

- If a value is specified for `collapse`, the values in the result are then concatenated into a single string, with the elements being separated by the value of `collapse`.

Formatting and Display of Strings: `paste`

- The `paste()` function concatenates several strings together.

- It creates a new string by joining the given strings end to end.

- The result of `paste()` can be assigned to a variable

  (in contrast to the function `cat()`).

`paste0(..., collapse)`

is equivalent to `paste(..., sep = "", collapse)`,

slightly more efficiently.

Formatting and Display of Strings

Example: `paste` converts its arguments to character strings (via `as.character`), and concatenates them

```
> paste(1:12)
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"
"9"  "10" "11" "12"

> as.character(1:12)  #Alternative to paste
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"
"9"  "10" "11" "12"
```

Formatting and Display of Strings

```
> paste("Everybody", "loves", "R Programming.")
[1] "Everybody loves R Programming."

> paste("Everybody", "loves", "R Programming.",
sep="*")
[1] "Everybody*loves*R Programming."

> paste("Everybody", "loves", "R Programming.",
sep="===")
[1] "Everybody===loves===R Programming."
```

Formatting and Display of Strings

R Console

```
> paste("Everybody", "loves", "R Programming.")
[1] "Everybody loves R Programming."
>
> paste("Everybody", "loves", "R Programming.", sep="*")
[1] "Everybody*loves*R Programming."
>
> paste("Everybody", "loves", "R Programming.", sep="===")
[1] "Everybody===loves===R Programming."
```

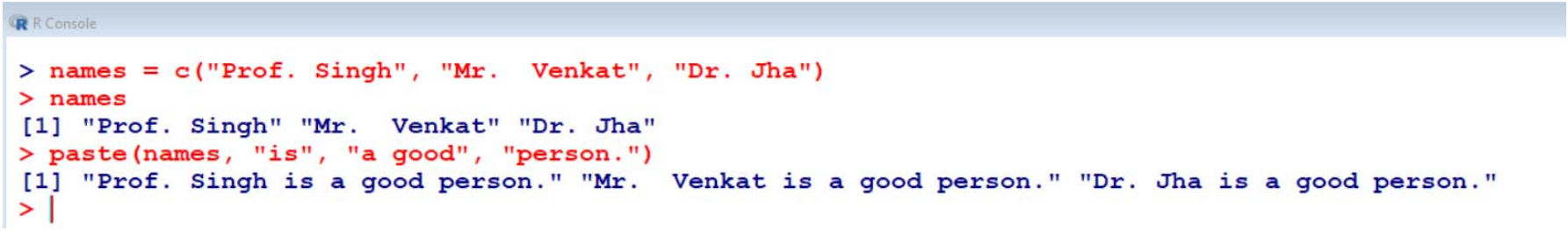
Formatting and Display of Strings

If one or more arguments are vectors of strings, `paste` will generate all combinations of the arguments:

```
> names = c("Prof. Singh", "Mr.  Venkat", "Dr. Jha")

> names
[1] "Prof. Singh" "Mr.  Venkat" "Dr. Jha"

> paste(names, "is", "a good", "person.")
[1] "Prof. Singh is a good person." "Mr.  Venkat is a good person." "Dr. Jha is a good person."
```



Formatting and Display of Strings

When we want to join even those combinations into one, big string.

The `collapse` parameter defines a top-level separator and instructs `paste` to concatenate the generated strings using that separator:

```
> names = c("Prof. Singh", "Mr.  Venkat", "Dr. Jha")

> paste(names, "is", "a good", "person.", collapse=", and ")
[1] "Prof. Singh is a good person., and Mr. Venkat is a good person., and Dr. Jha is a good person."
```

Formatting and Display of Strings

R Console

```
> names = c("Prof. Singh", "Mr.  Venkat", "Dr. Jha")
>
> paste(names, "is", "a good", "person.", collapse=", and ")
[1] "Prof. Singh is a good person., and Mr.  Venkat is a good person., and Dr. Jha is a good person."
>
```

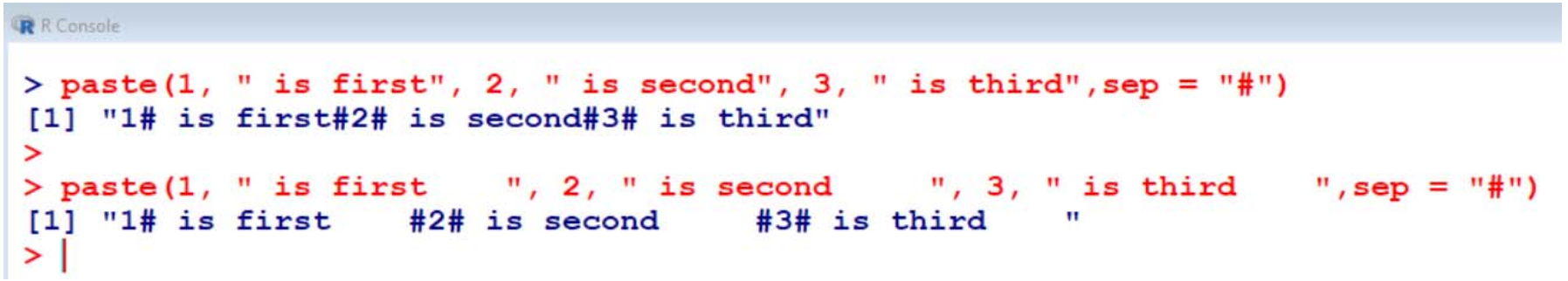
Operations with Strings

Example:

```
> paste(1, " is first", 2, " is second", 3, " is third",
sep = "#")
[1] "1# is first#2# is second#3# is third"
```

Observe the role of blank space inside double quotes

```
> paste(1, " is first    ", 2, " is second     ", 3, " is
third    ", sep = "#")
[1] "1# is first    #2# is second     #3# is third    "
```



Operations with Strings

Example:

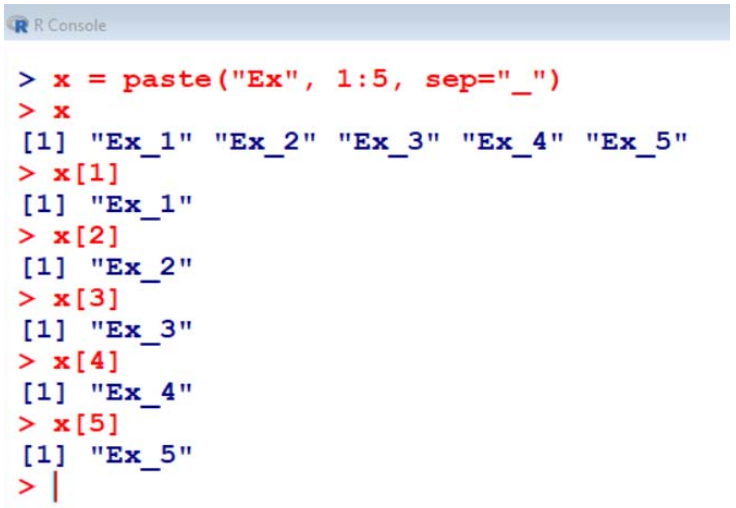
```
> x = paste("Ex", 1:5, sep="_")
>x
[1] "Ex_1" "Ex_2" "Ex_3" "Ex_4" "Ex_5"

> x[1]
[1] "Ex_1"

> x[2]
[1] "Ex_2"

> x[3]
[1] "Ex_3"

> x[5]
[1] "Ex_5"
```

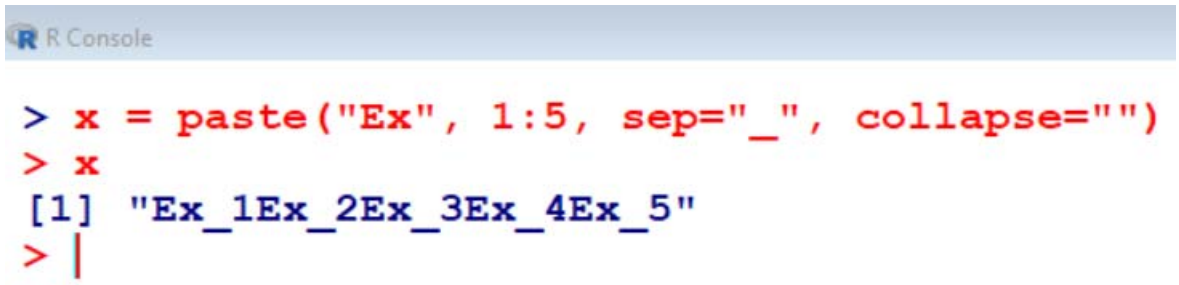


Operations with Strings

x is a vector of strings.

If we use the parameter `collapse`, a single string, instead of a vector of strings, is created:

```
> x = paste("Ex", 1:5, sep="_", collapse="")
> x[1]
[1] "Ex_1Ex_2Ex_3Ex_4Ex_5"
```



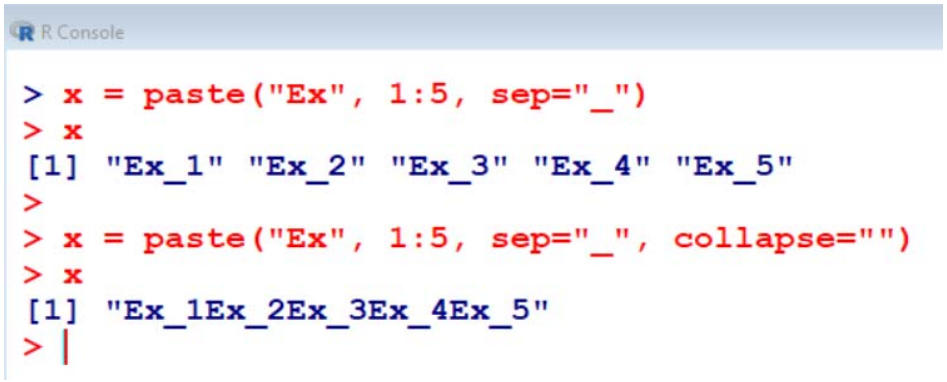
Operations with Strings

Note the difference between

```
x = paste("Ex", 1:5, sep="_")
> x
[1] "Ex_1" "Ex_2" "Ex_3" "Ex_4" "Ex_5"
```

and

```
x = paste("Ex", 1:5, sep="_", collapse="")
> x
[1] "Ex_1Ex_2Ex_3Ex_4Ex_5"
```



Operations with Strings

When using a single vector, `paste0` and `paste` have the same outcome and they work the same `as.character`.

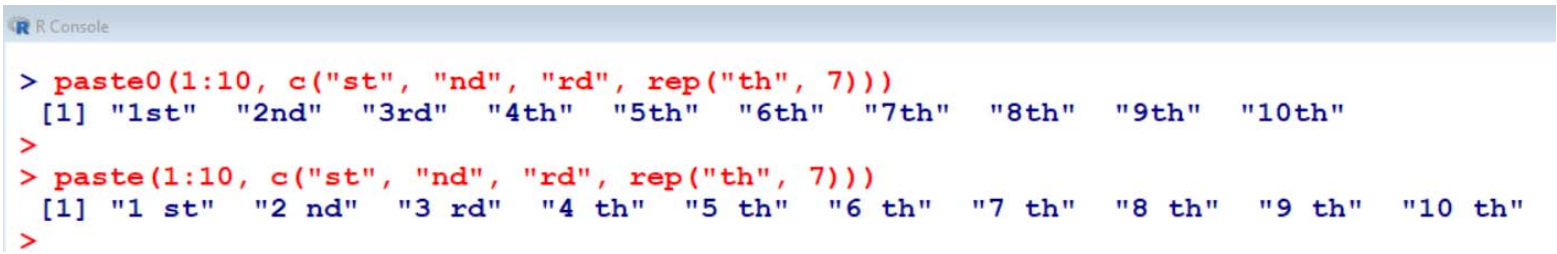
```
> paste0(1:10)
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"
"9"  "10"

> paste(1:10)
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"
"9"  "10"
```

Operations with Strings

When we use more than one vectors to `paste0` then they concatenate in a vectorized way.

```
> paste0(1:10, c("st", "nd", "rd", rep("th", 7)))
 [1] "1st"  "2nd"  "3rd"  "4th"  "5th"  "6th"  "7th"
"8th"  "9th"  "10th"
```

Observe the role of blank space

```
> paste(1:10, c("st", "nd", "rd", rep("th", 7)))
 [1] "1 st"  "2 nd"  "3 rd"  "4 th"  "5 th"  "6 th"
"7 th"  "8 th"  "9 th"  "10 th"
```