Introduction to R: **String Functions**

Session 1, Part C

Nick Graetz[1]

[1] University of Pennsylvania, Population Studies Center

9/4/2020

In this lecture

1. Pasting
2. Pattern matching
3. Substitution
4. Regular expressions
5. Other string commands

Pasting

`paste()` and `paste0()` are used to combine 2 or more vectors into a single character vector:

```
> day <- c(14, 18:20)
> month <- "September"
> year <- 2018
```

```
> paste0(month, day, year)
[1] "September142018" "September182018" "September192018"
[4] "September202018"
```

```
> paste(month, day, year)
[1] "September 14 2018" "September 18 2018"
[3] "September 19 2018" "September 20 2018"
```

```
> paste(month, day, year, sep="-")
[1] "September-14-2018" "September-18-2018"
[3] "September-19-2018" "September-20-2018"
```

Pasting

One very common use of `paste0()` is to construct file paths:

```
> main_dir <- "J:/temp/bootcamp_r_training/"
> paste0(main_dir, "data/us_state_cigarette_data.rdata")
[1] "J:/temp/bootcamp_r_training/data/us_state_cigarette_data.
```

Pattern matching

R has several functions for matching patterns in character vectors, including `grepl()`, which returns a logical vector telling you where there are matches:

```
> states <- c("North Carolina", "North Dakota", "South Dakota"
> grepl("Dakota", states)
[1] FALSE  TRUE  TRUE
```

Pattern matching

R has several functions for matching patterns in character vectors, including `grepl()`, which returns a logical vector telling you where there are matches:

```
> states <- c("North Carolina", "North Dakota", "South Dakota"
> grepl("Dakota", states)
[1] FALSE  TRUE  TRUE
```

and `grep()`, which returns the indices of any matches (or the actual matches, if `value=T`):

```
> grep("Dakota", states)
[1] 2 3
> grep("Dakota", states, value=T)
[1] "North Dakota" "South Dakota"
```

Pattern matching

`grepl()` and `grep()` are extremely useful for subsetting data:

```
> mmr <- c(31.66, 33.02, 81.42, 79.54, 88.74, 52.57,
+          50.42, 1246.75, 419.74, 489.17, 779.53)
> location_name <- c("Chile", "United Kingdom", "Guatemala", "Iraq",
+   "Bangladesh", "China", "Cambodia", "Central African Republic",
+   "Uganda", "Botswana", "Nigeria")
> region_name <- c("Southern Latin America", "Western Europe",
+   "Central Latin America", "North Africa and Middle East",
+   "South Asia", "East Asia", "Southeast Asia",
+   "Central Sub-Saharan Africa", "Eastern Sub-Saharan Africa",
+   "Southern Sub-Saharan Africa", "Western Sub-Saharan Africa")
> super_region_name <- c("High-income", "High-income",
+   "Latin America and Caribbean", "North Africa and Middle East",
+   "South Asia", "Southeast Asia, East Asia, and Oceania",
+   "Southeast Asia, East Asia, and Oceania", "Sub-Saharan Africa",
+   "Sub-Saharan Africa", "Sub-Saharan Africa", "Sub-Saharan Africa")
> options(width=60)
```

Substitution

A related function, `gsub()`, uses similar logic to identify and then replace patterns in a character vector

```
> region_name
 [1] "Southern Latin America"
 [2] "Western Europe"
 [3] "Central Latin America"
 [4] "North Africa and Middle East"
 [5] "South Asia"
 [6] "East Asia"
 [7] "Southeast Asia"
 [8] "Central Sub-Saharan Africa"
 [9] "Eastern Sub-Saharan Africa"
[10] "Southern Sub-Saharan Africa"
[11] "Western Sub-Saharan Africa"

> region_name <- gsub("South", "S.", region_name)
> region_name <- gsub("East|east", "E.", region_name)
> region_name
 [1] "S.ern Latin America"
 [2] "Western Europe"
 [3] "Central Latin America"
 [4] "North Africa and Middle E."
 [5] "S. Asia"
 [6] "E. Asia"
 [7] "S.E. Asia"
 [8] "Central Sub-Saharan Africa"
 [9] "E.ern Sub-Saharan Africa"
[10] "S.ern Sub-Saharan Africa"
[11] "Western Sub-Saharan Africa"
```

Regular expressions

More complicated pattern matching can be done using regular expressions (see `help(regexp)` for details).

So in addition to matching strings directly...

```
> colors <- c("red_blue_green", "red_green_orange", "orange_blue_red", "red_oran
> grep("orange", colors, value=T)
[1] "red_green_orange" "orange_blue_red"  "red_orange_green"
```

You can match the beginning or end of a string:

```
> grep("^orange", colors, value=T)
[1] "orange_blue_red"
> grep("orange$", colors, value=T)
[1] "red_green_orange"
```

Or add wildcards:

```
> grep("red_.*_green", colors, value=T)
[1] "red_blue_green"   "red_orange_green"
```

Other string commands

Beyond pattern matching, there are many functions that act specifically on character vectors, e.g.,

To force them to lower or upper case:

```
> tolower(states)
[1] "north carolina" "north dakota"   "south dakota"
> toupper(states)
[1] "NORTH CAROLINA" "NORTH DAKOTA"   "SOUTH DAKOTA"
```

To truncate them in some way:

```
> substr(states, 1, 5)
[1] "North" "North" "South"
```

Or to figure out how many characters they contain:

```
> nchar(region_name)
 [1] 19 14 21 26  7  7  9 26 24 24 26
```