Introduction to R: **String Functions**

Day 1, Part D

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 <- "C:/Users/ngraetz/Documents/repos/r_training_penn/"
> paste0(main_dir, "data/us_state_cigarette_data.rdata")
[1] "C:/Users/ngraetz/Documents/repos/r_training_penn/data/us_state_cig
```

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")
> data <- data.frame(mmr, location_name, region_name, super_region_name)
> head(data)
    mmr  location_name                region_name                      super_region_name
1 31.66          Chile       Southern Latin America                           High-income
2 33.02 United Kingdom               Western Europe                           High-income
3 81.42      Guatemala        Central Latin America          Latin America and Caribbean
4 79.54           Iraq North Africa and Middle East          North Africa and Middle East
5 88.74     Bangladesh                   South Asia                            South Asia
6 52.57          China                    East Asia Southeast Asia, East Asia, and Oceania
```

Pattern matching

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

```
> data <- data[grepl("Asia", data$region_name), ]
> head(data)
    mmr location_name    region_name                        super_region_name
5 88.74    Bangladesh     South Asia                                South Asia
6 52.57         China      East Asia Southeast Asia, East Asia, and Oceania
7 50.42      Cambodia Southeast Asia Southeast Asia, East Asia, and Oceania
```

Pattern matching

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

```
> data <- data[, grep("name", names(data))]
> head(data)
  location_name    region_name
5    Bangladesh     South Asia
6         China      East Asia
7      Cambodia Southeast Asia
                      super_region_name
5                            South Asia
6 Southeast Asia, East Asia, and Oceania
7 Southeast Asia, East Asia, and Oceania
```

Substitution

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

```
> data$region_name
[1] South Asia     East Asia      Southeast Asia
11 Levels: Central Latin America ...
```

```
> data$region_name <- gsub("South", "S.", data$region_name)
> data$region_name <- gsub("East|east", "E.", data$region_name)
> data$region_name
[1] "S. Asia"   "E. Asia"   "S.E. Asia"
```

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_orange_green")
> 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(data$region_name)
[1] 7 7 9
```