Strings in R

Barry Snowe

11/28/2016

Handling and Processing Strings in R

Chapter One notes

Preliminaries

Let's load the **stringr** library, which we'll need later:

```
library(stringr)
```

We will use the **USArrests** data set included in base R:

```
head(USArrests)
```

```
##            Murder Assault UrbanPop Rape
## Alabama      13.2     236       58 21.2
## Alaska       10.0     263       48 44.5
## Arizona       8.1     294       80 31.0
## Arkansas      8.8     190       50 19.5
## California    9.0     276       91 40.6
## Colorado      7.9     204       78 38.7
```

Let's get the names of the states:

```
states <- rownames(USArrests)
```

Locating values in strings

We would like to get the abbreviations of all of the state names. First we will try this using **substr()**

```
substr(x=states, start=1, stop=4)
```

```
##  [1] "Alab" "Alas" "Ariz" "Arka" "Cali" "Colo" "Conn" "Dela" "Flor" "Geor"
## [11] "Hawa" "Idah" "Illi" "Indi" "Iowa" "Kans" "Kent" "Loui" "Main" "Mary"
## [21] "Mass" "Mich" "Minn" "Miss" "Miss" "Mont" "Nebr" "Neva" "New " "New "
## [31] "New " "New " "Nort" "Nort" "Ohio" "Okla" "Oreg" "Penn" "Rhod" "Sout"
## [41] "Sout" "Tenn" "Texa" "Utah" "Verm" "Virg" "Wash" "West" "Wisc" "Wyom"
```

This leaves something to be desired.

Now we try again, using the **abbreviate()** function.

```
states2 <- abbreviate(states)
```

Remove vector names for convenience:

```
names(states2) = NULL
```

Have a look at the revised data:

```
states2
```

```
##  [1] "Albm" "Alsk" "Arzn" "Arkn" "Clfr" "Clrd" "Cnnc" "Dlwr" "Flrd" "Gerg"
## [11] "Hawa" "Idah" "Illn" "Indn" "Iowa" "Knss" "Kntc" "Losn" "Main" "Mryl"
## [21] "Mssc" "Mchg" "Mnns" "Msss" "Mssr" "Mntn" "Nbrs" "Nevd" "NwHm" "NwJr"
## [31] "NwMx" "NwYr" "NrtC" "NrtD" "Ohio" "Oklh" "Orgn" "Pnns" "RhdI" "SthC"
## [41] "SthD" "Tnns" "Texs" "Utah" "Vrmn" "Vrgn" "Wshn" "WstV" "Wscn" "Wymn"
```

If we want an abbreviation with more letters, we can change the argument *minlength*.

```
# abbreviate state names with 5 letters
abbreviate(states, minlength=5)
```

```
##        Alabama         Alaska        Arizona       Arkansas     California
##        "Alabm"        "Alask"        "Arizn"        "Arkns"        "Clfrn"
##       Colorado    Connecticut       Delaware        Florida        Georgia
##        "Colrd"        "Cnnct"        "Delwr"        "Flord"        "Georg"
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##        "Hawai"        "Idaho"        "Illns"        "Indin"        "Iowa"
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##        "Kanss"        "Kntck"        "Lousn"        "Maine"        "Mryln"
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##        "Mssch"        "Mchgn"        "Mnnst"        "Mssss"        "Missr"
##        Montana       Nebraska         Nevada  New Hampshire     New Jersey
##        "Montn"        "Nbrsk"        "Nevad"        "NwHmp"        "NwJrs"
##     New Mexico       New York North Carolina   North Dakota           Ohio
##        "NwMxc"        "NwYrk"        "NrthC"        "NrthD"        "Ohio"
##       Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
##        "Oklhm"        "Oregn"        "Pnnsy"        "RhdIs"        "SthCr"
##   South Dakota      Tennessee          Texas           Utah        Vermont
##        "SthDk"        "Tnnss"        "Texas"         "Utah"        "Vrmnt"
##       Virginia     Washington  West Virginia      Wisconsin        Wyoming
##        "Virgn"        "Wshng"        "WstVr"        "Wscns"        "Wymng"
```

Getting the longest name

We need to count the letters in each name. We could use the function **nchar()** for this.

```
# size (in characters) of each state
state_chars = nchar(states)
```

Display the longest name:

```
states[which(state_chars==max(state_chars))]
```

```
## [1] "North Carolina" "South Carolina"
```

Select just those states containing the letter "k". We can use the function **grep()** for this. We need to indicate pattern="k" in the arguments to **grep()**:

```
grep(pattern="k", x=states, value=TRUE)
```

```
## [1] "Alaska"       "Arkansas"     "Kentucky"     "Nebraska"
## [5] "New York"     "North Dakota" "Oklahoma"     "South Dakota"
```

To get the states containing "w":

```
grep(pattern="w", x=states, value=TRUE)
```

```
## [1] "Delaware"      "Hawaii"        "Iowa"          "New Hampshire"
## [5] "New Jersey"    "New Mexico"    "New York"
```

Notice that we only got states with *lower case* "w". Now how about getting states containing either upper or lower case "w"/"W"? There are a few options for dealing with this.

We could specify the searched pattern as a character class "[wW]":

```
grep(pattern="[wW]", x=states, value=TRUE)
```

```
##  [1] "Delaware"      "Hawaii"        "Iowa"          "New Hampshire"
##  [5] "New Jersey"    "New Mexico"    "New York"      "Washington"
##  [9] "West Virginia" "Wisconsin"     "Wyoming"
```

We could convert the state names to lower case, then look for "w":

```
grep(pattern="w", x=tolower(states), value=TRUE)
```

```
##  [1] "delaware"      "hawaii"        "iowa"          "new hampshire"
##  [5] "new jersey"    "new mexico"    "new york"      "washington"
##  [9] "west virginia" "wisconsin"     "wyoming"
```

Or, similarly, we could convert them to uppercase, then look for "W":

```
grep(pattern="W", x=toupper(states), value=TRUE)
```

```
##  [1] "DELAWARE"      "HAWAII"        "IOWA"          "NEW HAMPSHIRE"
##  [5] "NEW JERSEY"    "NEW MEXICO"    "NEW YORK"      "WASHINGTON"
##  [9] "WEST VIRGINIA" "WISCONSIN"     "WYOMING"
```

We could also specify the argument *ignore.case=TRUE* inside **grep()**

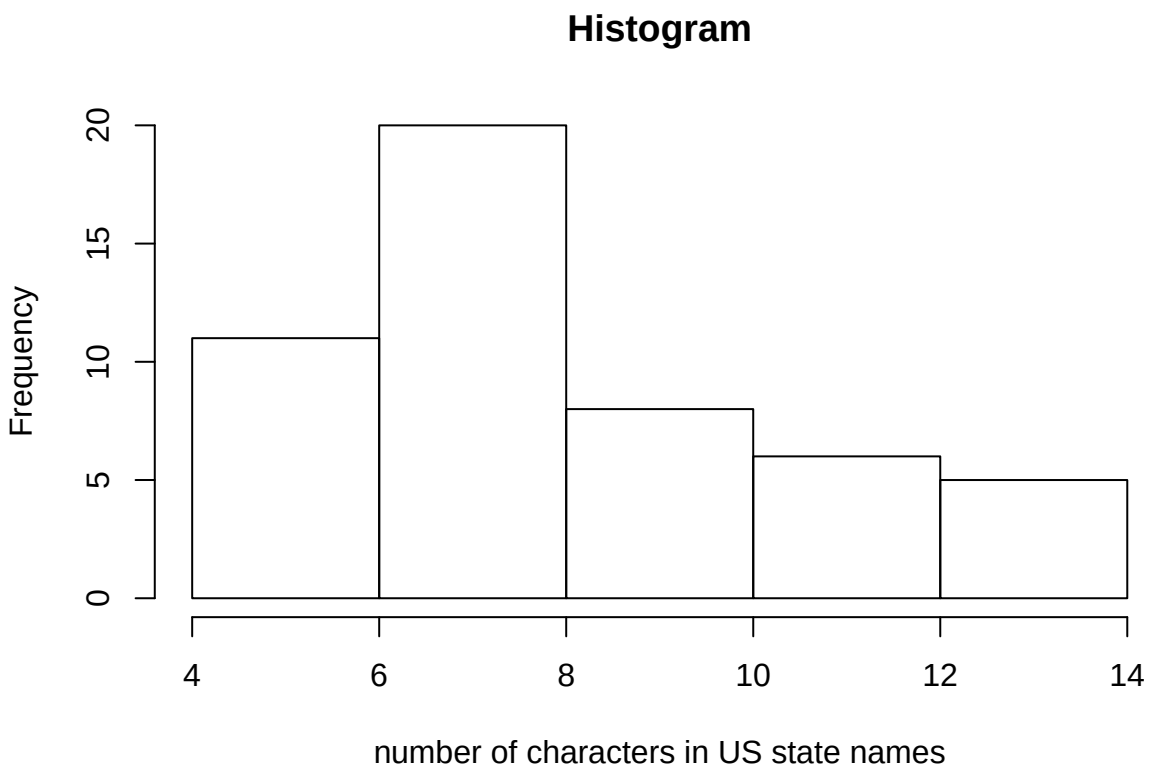
```
grep(pattern="w", x=states, value=TRUE, ignore.case=TRUE)
```

```
##  [1] "Delaware"      "Hawaii"        "Iowa"          "New Hampshire"
##  [5] "New Jersey"    "New Mexico"    "New York"      "Washington"
##  [9] "West Virginia" "Wisconsin"     "Wyoming"
```

Computations with string data

Make a histogram of the lengths in characters of the state names:

```
hist(nchar(states), main="Histogram",
     xlab="number of characters in US state names")
```



New question: what's the distribution of vowels in the names of the states? We can start with a simple case - the number of "a"'s in each name. To do this, we use the function **gregexpr()** to get the number of times that a searched pattern is found in a character vector. If there is no match you will get "-1"" returned.

```
# position of a's
position_a <- gregexpr(pattern="a", text=states, ignore.case=TRUE)
```

We'll use **sapply()** to replace those "-1"s with zeros, then we'll display the resulting vector object.

```
# how many a's?
num_a <- sapply(position_a, function(x) ifelse(x[1] > 0, length(x), 0))
num_a
```

```
##  [1] 4 3 2 3 2 1 0 2 1 1 2 1 0 2 1 2 0 2 1 2 2 1 1 0 0 2 2 2 1 0 0 0 2 2 0
## [36] 2 0 2 1 2 2 0 1 1 0 1 1 1 0 0
```

```
# this returns 0, not -1, for no match.
```

However, we could also use the **stringr** library's **str_count** function to get the counts. First, the total number of a's:

```
str_count(states, "a")
```

```
##  [1] 3 2 1 2 2 1 0 2 1 1 2 1 0 2 1 2 0 2 1 2 2 1 1 0 0 2 2 2 1 0 0 0 2 2 0
## [36] 2 0 2 1 2 2 0 1 1 0 1 1 1 0 0
```

Great, but we need to specify case. **str_count()** doesn't have an *ignore-case* argument. So let's use **tolower()**:

```
# total number of a's, either case
str_count(tolower(states), "a")
```

```
##  [1] 4 3 2 3 2 1 0 2 1 1 2 1 0 2 1 2 0 2 1 2 2 1 1 0 0 2 2 2 1 0 0 0 2 2 0
## [36] 2 0 2 1 2 2 0 1 1 0 1 1 1 0 0
```

Now that we can do it for one vowel, we can do it for all of them:

First, we create a vector of vowels:

```
vowels <- c('a', 'e', 'i', 'o', 'u')
```

Next, we create a vector for storing results:

```
num_vowels <- vector(mode="integer", length=5)
```

Next, calculate the number of vowels in each name:

```
for (j in seq_along(vowels)) {
  num_aux <- str_count(tolower(states), vowels[j])
  num_vowels[j] = sum(num_aux)
}
```

Next, add vowel names to **num_vowels**:

```
names(num_vowels) = vowels
```

Now, display the total number of vowels:

```
num_vowels
```

```
##  a  e  i  o  u
## 61 28 44 36  8
```

Sort them in decreasing order:

```
sort(num_vowels, decreasing=TRUE)
```

```
##  a  i  o  e  u
## 61 44 36 28  8
```

Finally, we can visualize this distribution with a barplot:

```
barplot(num_vowels, main="Number of vowels in US state names",
        border=NA, ylim=c(0, 80))
```