Demonstration of foofactors new capabilities

Elijah Willie

Contents

Introduction

Hello my name is `Elijah Willie` and welcome to a modified version of the `foofactor` package. In this document, I will attempt to show the new functionalities of this package.

Installtion

Below is how one would go about installing this package

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(devtools)
# devtools::install_github("STAT545-UBC-students/hw07-ecool50")
```

Functions in foofactor

Factors are a very useful type of variable in R, but they can also drive you nuts. Especially the "stealth factor" that you think of as character.

Can we soften some of their sharp edges?

fbind

Binding two factors via `fbind()`:

```
library(foofactors)
a <- factor(c("character", "hits", "your", "eyeballs"))
b <- factor(c("but", "integer", "where it", "counts"))
```

Simply catenating two factors leads to a result that most don't expect.

```
c(a, b)
```

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

The `fbind()` function glues two factors together and returns factor.

```
fbind(a, b)
```

```
## [1] character hits      your      eyeballs  but       integer   where it
## [8] counts
## Levels: but character counts eyeballs hits integer where it your
```

Often we want a table of frequencies for the levels of a factor. The base `table()` function returns an object of class `table`, which can be inconvenient for downstream work. Processing with `as.data.frame()` can be helpful but it's a bit clunky.

```
set.seed(1234)
x <- factor(sample(letters[1:5], size = 100, replace = TRUE))
table(x)
```

```
## x
##  a  b  c  d  e
## 25 26 17 17 15
```

```
as.data.frame(table(x))
```

```
##   x Freq
## 1 a   25
## 2 b   26
## 3 c   17
## 4 d   17
## 5 e   15
```

freq_out

The `freq_out()` function returns a frequency table as a well-named `tbl_df`:

```
freq_out(x)
```

```
## # A tibble: 5 x 2
##   x         n
##   <fct> <int>
## 1 a        25
## 2 b        26
## 3 c        17
## 4 d        17
```

```
## 5 e        15
```

isFactor

One of the new functionalities is the `isFactor` function. This function checks to see if a factor could be represented as a string. If the number of levels of the factor is equal to the length of the factor, then it is most likely a character rather than a factor.

```
#show the case where it is indeed a factor
x <- c('X', 'X', 'Y')
x_new <- factor(x)
y <- c('W', 'X')
y_new <- factor(y)
isFactor(x_new)
```

```
## [1] FALSE
```

```
isFactor(y_new)
```

```
## [1] TRUE
```

We see that for the first case it returns `False` and for the second case it returns `True`

Factor_Reorder

The `Factor_Reorder` function takes in a factor and returns the levels of the factor reordered. The reordering is done in descending order.

```
#create a factor variable
x <- c('X', 'X', 'Y', 'Q', 'P')
x_new <- factor(x)
#get the levesl of x
x_levels <- levels(x_new)

#now reorder the factor levels
x_reord <- Factor_Reorder(x_new)

#print the results
print("Original factor levels")
```

```
## [1] "Original factor levels"
```

```
x_levels
```

```
## [1] "P" "Q" "X" "Y"
```

```
print("Reordered factor levels")
```

```
## [1] "Reordered factor levels"
```

```
x_reord
```

```
## [1] "Y" "X" "Q" "P"
```

Factor_Order

The function `Factor_Order` takes in a factor and returns the levels of the factor in the same order as they appear in the dataset.

```
#create a factor variable
x <- c('X', 'X', 'Y', 'Q', 'P')
x_new <- factor(x)

print("the original factor levels")
```

```
## [1] "the original factor levels"
```

```
levels(x_new)
```

```
## [1] "P" "Q" "X" "Y"
```

```
print("After setting the factors as is")
```

```
## [1] "After setting the factors as is"
```

```
Factor_Order(x_new)
```

```
## [1] Q Q P Y X
## Levels: X Y Q P
```

We see that the factors are returned as is in the original dataset.

Read_textFile

Now I will demonstrate reading in a textfile and automatically detecting its delimeter

```
#set the data source
datae_source <- "http://www.stat.ubc.ca/~jenny/notOcto/STAT545A/examples/gapminder/data/gapminderDataFi

#read in the data
df <- Read_textFile(datae_source)

#show it
knitr::kable(head(df, 10))
```

country	year	pop	continent	lifeExp	gdpPercap
Afghanistan	1952	8425333	Asia	28.801	779.4453
Afghanistan	1957	9240934	Asia	30.332	820.8530
Afghanistan	1962	10267083	Asia	31.997	853.1007
Afghanistan	1967	11537966	Asia	34.020	836.1971
Afghanistan	1972	13079460	Asia	36.088	739.9811
Afghanistan	1977	14880372	Asia	38.438	786.1134
Afghanistan	1982	12881816	Asia	39.854	978.0114
Afghanistan	1987	13867957	Asia	40.822	852.3959
Afghanistan	1992	16317921	Asia	41.674	649.3414
Afghanistan	1997	22227415	Asia	41.763	635.3414

```
#get some summary
str(df)
```

```
## 'data.frame':    1704 obs. of  6 variables:
##  $ country  : Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ year     : int  1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
##  $ pop      : num  8425333 9240934 10267083 11537966 13079460 ...
##  $ continent: Factor w/ 5 levels "Africa","Americas",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ lifeExp  : num  28.8 30.3 32 34 36.1 ...
##  $ gdpPercap: num  779 821 853 836 740 ...
```

Looking at the structure of the data, we can see that both country and continent was read in as factors as desired.

Write_textFile

Finally, I will demonstrate the writing to text file functionality.

```
library(gapminder)
#create a new dataframe to write
df_new <- data.frame(gapminder$country, gapminder$lifeExp)

#write it to a textfile
Write_textFile(df_new, name = "Gapminder_cont_lifeExp.txt", delim = "\t")

#read it back to double check
df_gap <- Read_textFile("Gapminder_cont_lifeExp.txt")

#double check it was written correctly
knitr::kable(head(df_gap, 10))
```

gapminder.country	gapminder.lifeExp
Afghanistan	28.801
Afghanistan	30.332
Afghanistan	31.997
Afghanistan	34.020
Afghanistan	36.088
Afghanistan	38.438
Afghanistan	39.854
Afghanistan	40.822
Afghanistan	41.674
Afghanistan	41.763

Fabulous! everything works!

Credits

- Some of the work here may have been borrowed and modified from other sources.
- This was purely done to illustrate some ideas
- I take no credit for parts of the work that are not mine
- Let credit be given where it is due.
- The foofactor package was originally written by `Jennifer Bryan` and can be found here