Tidybooster

2022-08-09

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.2.1
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## Warning: package 'readr' was built under R version 4.2.1
```

```
## Warning: package 'forcats' was built under R version 4.2.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(janitor)
```

```
## Warning: package 'janitor' was built under R version 4.2.1
```

```
##
## Attaching package: 'janitor'
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
library(fs)
```

```
## Warning: package 'fs' was built under R version 4.2.1
```

1 How to improve reading files with read_* function

En janitor podemos usar clean_names() para convertir todos los nombres a lowercase

```
mpg_new <- read_csv("data/mpg_uppercase.csv") %>% clean_names()
```

```
## Rows: 234 Columns: 11
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (6): MANUFACTURER, MODEL, TRANS, DRV, FL, CLASS
## dbl (5): DISPL, YEAR, CYL, CTY, HWY
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Tambien es posible lograr esto usando name_repair option de read_csv

```
read_csv("data/mpg_uppercase.csv",
show_col_types = FALSE,
name_repair = make_clean_names) %>%
glimpse()
```

```
## Rows: 234
## Columns: 11
## $ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi", "~
## $ model        <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro", "~
## $ displ        <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, 2.~
## $ year         <dbl> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, 200~
## $ cyl          <dbl> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 8, 8, ~
## $ trans        <chr> "auto(l5)", "manual(m5)", "manual(m6)", "auto(av)", "auto~
## $ drv          <chr> "f", "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", "4~
## $ cty          <dbl> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17, 1~
## $ hwy          <dbl> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25, 2~
## $ fl           <chr> "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p~
## $ class        <chr> "compact", "compact", "compact", "compact", "compact", "c~
```

1.2 Reemplazar y remover caracteres en los nombres de tus columnas

por ejemplo, para reemplazar % con _percent en este vector

```
make_clean_names(c("A","B%","C"),replace = c("%"="_percent"))
```

```
## [1] "a"         "b_percent" "c"
```

De la misma forma podemos usar expresiones regulares

```
make_clean_names(c("A_1","B_1","C_1"),replace = c("^A_"="a"))
```

```
## [1] "a1"  "b_1" "c_1"
```

1.3 Using a specific naming convention

podemos modificar la funcion make_clean names para algun case especifico

```
make_clean_names(c("myHouse", "MyGarden"),
case = "snake")
```

```
## [1] "my_house"  "my_garden"
```

Asi, podemos hacer

```
read_csv("data/mpg_uppercase.csv",
show_col_types = FALSE,
name_repair = ~ make_clean_names(., case = "upper_camel")) %>%
glimpse()
```

```
## Rows: 234
## Columns: 11
## $ Manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi", "~
## $ Model        <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro", "~
## $ Displ        <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, 2.~
## $ Year         <dbl> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, 200~
## $ Cyl          <dbl> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 8, 8, ~
## $ Trans        <chr> "auto(l5)", "manual(m5)", "manual(m6)", "auto(av)", "auto~
## $ Drv          <chr> "f", "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", "4~
```

```
## $ Cty          <dbl> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17, 1~
## $ Hwy          <dbl> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25, 2~
## $ Fl           <chr> "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p~
## $ Class        <chr> "compact", "compact", "compact", "compact", "compact", "c~
```

El punto en make-clean_names es tidyevaluation, indicando los columnames del dataset

tambien podemos escoger las columnas usando col_select, ahorrando tiempo de ejecucion

```
read_csv("data/mpg_uppercase.csv",
show_col_types = FALSE,
name_repair = make_clean_names,
col_select = c(manufacturer, model)) %>%
glimpse()
```

```
## Rows: 234
## Columns: 2
## $ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi", "~
## $ model        <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro", "~
```

1.2 read many files into R

Es muy comun leer data esparcida en muchos archivos. Buscaremos automatizar esto

Para este ejemplo crearemos 25 archivos csv tomando muestras de los datos mpg

```
mpg_samples <- map(1:25, ~ slice_sample(mpg,n =20))
iwalk(mpg_samples, ~ write_csv(.,paste0("many_files/",.y,".csv")))
```

Antes de leer los archivos creamos un vector con todos los file paths. Hay muchas opciones, podemos usar list.files, o podemos usar dir_ls

```
(csv_files_list_files <- list.files(path = "many_files/", pattern = "csv", full.names = TRUE))
```

```
##  [1] "many_files/1.csv"  "many_files/10.csv" "many_files/11.csv"
##  [4] "many_files/12.csv" "many_files/13.csv" "many_files/14.csv"
##  [7] "many_files/15.csv" "many_files/16.csv" "many_files/17.csv"
## [10] "many_files/18.csv" "many_files/19.csv" "many_files/2.csv"
## [13] "many_files/20.csv" "many_files/21.csv" "many_files/22.csv"
## [16] "many_files/23.csv" "many_files/24.csv" "many_files/25.csv"
## [19] "many_files/3.csv"  "many_files/4.csv"  "many_files/5.csv"
## [22] "many_files/6.csv"  "many_files/7.csv"  "many_files/8.csv"
## [25] "many_files/9.csv"
```

Para leer todos estos archivos usamos la funcion map_dfr de la libreria purrr

```
data_frames <- map_dfr(csv_files_list_files, ~ read_csv(.x,show_col_types = FALSE))
```

podemos lograr lo mismo pasando el vector de archivos a la funcionn read_css

```
read_csv(csv_files_list_files, id = "filename", show_col_types = FALSE) %>% glimpse()
```

```
## Rows: 500
## Columns: 12
## $ filename     <chr> "many_files/1.csv", "many_files/1.csv", "many_files/1.csv~
## $ manufacturer <chr> "nissan", "subaru", "nissan", "ford", "honda", "chevrolet~
## $ model        <chr> "altima", "impreza awd", "pathfinder 4wd", "f150 pickup 4~
## $ displ        <dbl> 2.4, 2.2, 3.3, 5.4, 1.6, 6.0, 4.6, 5.3, 5.3, 1.8, 2.0, 5.~
## $ year         <dbl> 1999, 1999, 1999, 2008, 1999, 2008, 1999, 2008, 2008, 199~
## $ cyl          <dbl> 4, 4, 6, 8, 4, 8, 8, 8, 8, 4, 4, 8, 8, 6, 8, 8, 4, 8, 6, ~
```

```
## $ trans        <chr> "manual(m5)", "auto(l4)", "auto(l4)", "auto(l4)", "manual~
## $ drv          <chr> "f", "4", "4", "4", "f", "r", "r", "r", "f", "f", "f", "4~
## $ cty          <dbl> 21, 21, 14, 13, 25, 12, 15, 14, 16, 21, 19, 11, 11, 15, 1~
## $ hwy          <dbl> 29, 26, 17, 17, 32, 17, 21, 20, 25, 29, 26, 15, 15, 18, 2~
## $ fl           <chr> "r", "r", "r", "r", "r", "r", "r", "r", "p", "p", "r", "r~
## $ class        <chr> "compact", "subcompact", "suv", "pickup", "subcompact", "~
```

podemos meter el archivo de origen usando el primer metodo agrefando una linea de codigo

```
map_dfr(csv_files_list_files,
~ read_csv(.x, , show_col_types = FALSE) %>%
mutate(filename = .x)) %>%
glimpse()
```

```
## Rows: 500
## Columns: 12
## $ manufacturer <chr> "nissan", "subaru", "nissan", "ford", "honda", "chevrolet~
## $ model        <chr> "altima", "impreza awd", "pathfinder 4wd", "f150 pickup 4~
## $ displ        <dbl> 2.4, 2.2, 3.3, 5.4, 1.6, 6.0, 4.6, 5.3, 5.3, 1.8, 2.0, 5.~
## $ year         <dbl> 1999, 1999, 1999, 2008, 1999, 2008, 1999, 2008, 2008, 199~
## $ cyl          <dbl> 4, 4, 6, 8, 4, 8, 8, 8, 8, 4, 4, 8, 8, 6, 8, 8, 4, 8, 6, ~
## $ trans        <chr> "manual(m5)", "auto(l4)", "auto(l4)", "auto(l4)", "manual~
## $ drv          <chr> "f", "4", "4", "4", "f", "r", "r", "r", "f", "f", "f", "4~
## $ cty          <dbl> 21, 21, 14, 13, 25, 12, 15, 14, 16, 21, 19, 11, 11, 15, 1~
## $ hwy          <dbl> 29, 26, 17, 17, 32, 17, 21, 20, 25, 29, 26, 15, 15, 18, 2~
## $ fl           <chr> "r", "r", "r", "r", "r", "r", "r", "r", "p", "p", "r", "r~
## $ class        <chr> "compact", "subcompact", "suv", "pickup", "subcompact", "~
## $ filename     <chr> "many_files/1.csv", "many_files/1.csv", "many_files/1.csv~
```

pero que sucede cuando los nombres de las columnas son incosistentes?

```
mpg_samples <- map(1:10, ~ slice_sample(mpg, n = 20))
inconsistent_dframes <- map(mpg_samples,~ janitor::clean_names(dat = .x, case = "random"))
```

Para mejorar las cosas elijamos un conjunto de columnas aleatorios

```
inconsistent_dframes <- map(inconsistent_dframes,
~ .x[sample(1:length(.x), sample(1:length(.x), 1))])
map(inconsistent_dframes, ~ colnames(.x)) %>%
head
```

```
## [[1]]
## [1] "cyl"          "DrV"          "Fl"           "YeAr"         "TraNs"
## [6] "MAnuFACTUrER"
##
## [[2]]
## [1] "DrV"
##
## [[3]]
## [1] "MoDel" "cYL"
##
## [[4]]
## [1] "HwY" "cYL"
##
## [[5]]
## [1] "Cyl"   "diSPL" "clasS" "Hwy"   "YEar"
##
```

```
## [[6]]
## [1] "drv"          "moDEl"        "DISPL"        "ctY"          "YEAR"
## [6] "ManUfActUrer" "cyl"          "cLasS"        "fl"
```

```
iwalk(inconsistent_dframes,
~ write_csv(.x, paste0("unclean_files/", .y, ".csv")))
```

Para leer todos estos archivos sin terminar con un monton de nombres repetidos e incosistentes podemos agregar una convencion de tipado para la lectura de las columnas

```
many_columns_data_frame <- dir_ls(path = "unclean_files/",
glob = "*.csv", type = "file") %>%
map_dfr(~ read_csv(.x, name_repair = tolower, show_col_types = FALSE) %>%
mutate(filename = .x))
many_columns_data_frame %>% glimpse()
```

```
## Rows: 200
## Columns: 12
## $ cyl          <dbl> 8, 8, 6, 6, 8, 4, 8, 5, 4, 6, 4, 4, 8, 6, 4, 4, 4, 8, 6, ~
## $ drv          <chr> "4", "4", "4", "4", "4", "f", "r", "f", "f", "f", "4", "4~
## $ fl           <chr> "r", "r", "r", "p", "r", "d", "r", "r", "r", "r", "p", "r~
## $ year         <dbl> 1999, 2008, 1999, 1999, 2008, 1999, 2008, 2008, 1999, 200~
## $ trans        <chr> "auto(l4)", "auto(l5)", "manual(m5)", "auto(l5)", "auto(l~
## $ manufacturer <chr> "chevrolet", "jeep", "toyota", "audi", "ford", "volkswage~
## $ filename     <fs::path> "unclean_files/1.csv", "unclean_files/1.csv", "uncle~
## $ displ        <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ model        <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ hwy          <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ class        <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ cty          <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
```

σ