1 How to improve reading files with the `read_*` functions

> **What will this tutorial cover?**
>
> In this tutorial you will learn how to clean column names, replace strings from column names, and select columns directly in the `read_*` functions.

> **Who do I have to thank?**
>
> I came across this trick from a blog post by Jim Hester on tidyverse.org. If you want to dive deeper, read the fantastic blog post.

It is rare to read a CSV file without any data cleaning. Suppose I want to convert the column names of this CSV file to lowercase and select only columns that start with the letter "m":

```
MANUFACTURER,MODEL,DISPL,YEAR,CYL,TRANS,DRV,CTY,HWY,FL,CLASS
audi,a4,1.8,1999,4,auto(l5),f,18,29,p,compact
audi,a4,1.8,1999,4,manual(m5),f,21,29,p,compact
audi,a4,2,2008,4,manual(m6),f,20,31,p,compact
audi,a4,2,2008,4,auto(av),f,21,30,p,compact
audi,a4,2.8,1999,6,auto(l5),f,16,26,p,compact
audi,a4,2.8,1999,6,manual(m5),f,18,26,p,compact
```

Most of us would probably read the CSV file first and then do the data cleaning. For example, using the clean_names function from the janitor package (fyi: I use the `show_col_types` argument here to hide the output. You don't need to use this argument):

```r
library(tidyverse)
library(janitor)

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

```r
  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", "~
```

This approach is perfectly fine. It turns out, however, that the `read_*` functions have some data cleaning arguments build in. These arguments don't allow you to do something new, but they do allow you to encapsulate the reading of the data with the data cleaning. Let's see how.

1.1 Converting column names to lowercase

In my previous example, I have used the `clean_names` function from the janitor package to convert the column names to lowercase. The same can be achieved inside `read_csv` with the function make_clean_names for the `name_repair` argument:

```r
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~
```

As you can see, I use the `make_clean_names` function here and not `clean_names`. This is because `clean_names` does not work with vectors, but `make_clean_names` does.

1.2 Replacing and removing character strings in your column names

With `make_clean_names` you can also replace certain characters from the column names. Suppose we want to replace the character "%" with the actual word "_percent":

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

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

If you are familiar with regular expressions, you can make more complex replacements. For example, you could remove the underscore for all column names that start with the letter "A":

```
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 for column names

You may have noticed that in the last example `make_clean_names` converted the column names to lowercase. That's because the function uses the snake naming convention by default. Snake converts all names to lowercase and separates words with an underscore:

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

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

If you do not want to change the naming convention of your column names at all, use "none" for the case:

```
make_clean_names(c("myHouse", "MyGarden"),
                 case = "none")
```

```
[1] "myHouse"  "MyGarden"
```

Here is a list of all naming conventions you can use:

Naming conventions

Naming Convention	example1	example2
snake	myHouse -> my_house	MyGarden -> my_garden
small_camel	myHouse -> myHouse	MyGarden -> myGarden
big_camel	myHouse -> MyHouse	MyGarden -> MyGarden
screaming_snake	myHouse -> MY_HOUSE	MyGarden -> MY_GARDEN
parsed	myHouse -> my_House	MyGarden -> My_Garden
mixed	myHouse -> my_House	MyGarden -> My_Garden
lower_upper	myHouse -> myHOUSE	MyGarden -> myGARDEN
upper_lower	myHouse -> MYhouse	MyGarden -> MYgarden
swap	myHouse -> MYhOUSE	MyGarden -> mYgARDEN
all_caps	myHouse -> MY_HOUSE	MyGarden -> MY_GARDEN
lower_camel	myHouse -> myHouse	MyGarden -> myGarden
upper_camel	myHouse -> MyHouse	MyGarden -> MyGarden
internal_parsing	myHouse -> my_House	MyGarden -> My_Garden
none	myHouse -> myHouse	MyGarden -> MyGarden
flip	myHouse -> MYhOUSE	MyGarden -> mYgARDEN
sentence	myHouse -> My house	MyGarden -> My garden
random	myHouse -> mYHoUsE	MyGarden -> MYgArDEn
title	myHouse -> My House	MyGarden -> My Garden

For example, in our dataset we could change the column names to `upper_camel`:

```
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~
```

The dot . in `make_clean_names` denotes the vector of column names.

Selecting specific columns

Apart from cleaning your column names, you can also select columns directly from `read_csv` using the `col_select` argument:

```
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", "~
```

In this example, I have explicitly selected the columns. You can also use the tidyselect functions, which we will cover in another tutorial in this course.

> **Summary**
>
> Here's what you can take away from this tutorial:
>
> - You can clean column names, replace strings in columns, and select columns directly in your `read_*` functions
> - With the function `make_clean_names` you can convert your column names to certain naming conventions (e.g., small camel)
> - You can use regular expressions with the replace argument in `make_clean_names` to remove or replace specific characters from your column names
> - You can use tidyselect functions to select a subset of columns with the `col_select` argument