What we'll be covering:

- Row-wise operation the column-wise way with `pivot_longer`
- Row-wise data operation with `transpose`
- Row-wise operations with `rowSums`
- Row-wise operations with `pmap` and `apply`
- ID columns for doing row-wise operations the column-wise way

Get the #tidytuesday Data

```
library(tidyverse)
library(tidytuesdayR)

plastic <- tidytuesdayR::tt_download(tt_load_gh("2021-01-26"))
plastic[[1]]

## # A tibble: 13,380 x 14
##    country  year parent_company empty  hdpe  ldpe     o   pet    pp
ps   pvc
##
##  1 Argent…  2019 Grand Total        0   215    55   607  1376   281
116    18
##  2 Argent…  2019 Unbranded          0   155    50   532   848   122
114    17
##  3 Argent…  2019 The Coca-Cola…     0     0     0     0   222    35
0     0
##  4 Argent…  2019 Secco              0     0     0     0    39     4
0     0
##  5 Argent…  2019 Doble Cola         0     0     0     0    38     0
0     0
##  6 Argent…  2019 Pritty             0     0     0     0    22     7
0     0
##  7 Argent…  2019 PepsiCo            0     0     0     0    21     6
0     0
##  8 Argent…  2019 Casoni             0     0     0     0    26     0
0     0
##  9 Argent…  2019 Villa Del Sur…     0     0     0     0    19     1
0     0
## 10 Argent…  2019 Manaos             0     0     0     0    14     4
0     0
## # … with 13,370 more rows, and 3 more variables: grand_total ,
## #   num_events , volunteers
```

We can see that there are certain names for different plastic types in the columns. The columns that count the different types of plastic are from `empty` until `pvc`. There is also a `grand_total` column that sums up all the plastic. Let's do a little quality check and see if the `grand_total` column has all the plastic from column `empty` until column `pvc` summed up.

For the approach, we will be using the `pivot_longer` function from the `tidyr` package. We will demonstrate some other approaches later that perform some row-wise operations. However, I have a better mental model of the data and what I want to accomplish when I am thinking column-wise.

Row-wise operation the column-wise way with pivot_longer()

```
plastic[[1]] %>%
  tidyr::pivot_longer(empty:pvc) %>%
  dplyr::group_by(country, year, parent_company) %>%
  dplyr::mutate(grand_total_check = sum(value, na.rm = T)) %>%
  dplyr::ungroup() %>%
  tidyr::pivot_wider() %>%
  dplyr::filter(grand_total_check != grand_total)
```

```
## # A tibble: 2 x 15
##   country  year parent_company grand_total num_events volunteers
##
## 1 Korea    2020 Dongsuh                 44         26         NA
## 2 United…  2020 null                  4037        134        511
## # … with 9 more variables: grand_total_check , empty , hdpe ,
## #   ldpe , o , pet , pp , ps , pvc
```

With the code above, we are pivoting longer on all the plastic categories and then are identifying the row by grouping by `country`, `year`, and `parent_company`. The we can do any operation we wish to do in the `summarise` function.

And we find some rows where the sum in the plastic columns does not match the total.

Row-wise Data Operations With transpose

```
plastic[[1]] %>%
  dplyr::mutate(
    grand_total_check = plastic[[1]] %>%
      dplyr::select(empty:pvc) %>%
      purrr::transpose() %>%
      purrr::map_dbl(~ flatten_dbl(.) %>%
                       sum(na.rm = T))
  ) %>%
  dplyr::filter(grand_total_check != grand_total)
```

```
## # A tibble: 2 x 15
##   country  year parent_company empty  hdpe  ldpe     o   pet    pp
ps   pvc
##
## 1 Korea    2020 Dongsuh            0     0    32     4     2     0
0     0
## 2 United…  2020 null             102   165   275   752   788  1668
243    41
## # … with 4 more variables: grand_total , num_events ,
## #   volunteers , grand_total_check
```

It basically turns the data frame “inside-out” and then we can add up the rows with `map_dbl`.

Row-wise Operations with rowSums

We can also just simply with the base R `rowSums` function. However, this operation is specific to adding up numbers so if you would like to do other row-wise operations, you have to use another function.

```
plastic[[1]] %>%
  dplyr::mutate(
    grand_total_check = dplyr::select(
      ., empty:pvc
    ) %>% base::rowSums(na.rm = T)
  ) %>%
  dplyr::filter(grand_total_check != grand_total)

plastic[[1]] %>%
  dplyr::mutate(
    grand_total_check = dplyr::select(
      ., empty:pvc
    ) %>% base::rowSums(na.rm = T)
  ) %>%
  dplyr::filter(grand_total_check != grand_total)
```

Row-wise Operations With pmap and apply

Other ways to do row-wise operations are with `purrr's pmap` function and the base R apply function.

```
plastic[[1]] %>%
  dplyr::mutate(
    grand_total_check = dplyr::select(
      ., empty:pvc
    ) %>% purrr::pmap_dbl(sum, na.rm = T)
  ) %>%
  dplyr::filter(grand_total_check != grand_total)

## # A tibble: 2 x 15
##   country  year parent_company empty  hdpe  ldpe     o   pet    pp
ps   pvc
##
## 1 Korea    2020 Dongsuh            0     0    32     4     2     0
0     0
## 2 United…  2020 null             102   165   275   752   788  1668
243    41
## # … with 4 more variables: grand_total , num_events ,
## #   volunteers , grand_total_check
plastic[[1]] %>%
  dplyr::mutate(
    grand_total_check = apply(dplyr::select(plastic[[1]], empty:pvc),
1,
                              function(x) {sum(x, na.rm = TRUE)})
  ) %>%
  dplyr::filter(grand_total_check != grand_total)

## # A tibble: 2 x 15
##   country  year parent_company empty  hdpe  ldpe     o   pet    pp
ps   pvc
##
## 1 Korea    2020 Dongsuh            0     0    32     4     2     0
0     0
## 2 United…  2020 null             102   165   275   752   788  1668
243    41
## # … with 4 more variables: grand_total , num_events ,
## #   volunteers , grand_total_check
```

All these approaches work. However, lately, I have been finding myself using the `pivot_longer` and `pivot_wider` functions a lot from the `tidyr` package to do row-wise operations.

ID Columns for Doing Row-wise Operations the Column-wise Way

Sometimes, you have to first add an id to do row-wise operations column-wise. For example, when you would like to sum up all the rows where the columns are numeric in the `mtcars` data set, you can add an id, `pivot_wider` and then group by id (the row previously) and then sum up the value.

```
mtcars %>%
  dplyr::select_if(is.numeric) %>%
  dplyr::mutate(id = dplyr::row_number()) %>%
  tidyr::pivot_longer(-id) %>%
  dplyr::group_by(id) %>%
  dplyr::summarise(total = sum(value))
```

```
## # A tibble: 32 x 2
##       id total
##   *
##  1     1  329.
##  2     2  330.
##  3     3  260.
##  4     4  426.
##  5     5  590.
##  6     6  386.
##  7     7  657.
##  8     8  271.
##  9     9  300.
## 10    10  350.
## # … with 22 more rows
```

Additional Resources

- A tutorial about `pivot_wider()` and `pivot_longer()` with a #tidytuesday data set
- Row-wise data operation's with `purrr` and `pmap`
- Another `purrr` `apply` example