Border Crossing Entry Data

Paul Perez

12/1/2019

Library

```
## -- Attaching packages ------------------------------------------------------------------- tidyverse 1

## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

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

Packages

```
library(tibble)
library(dplyr)
```

Data

We're going to use a 'Border Crossing' data set from The Bureau of Transportation Statistics (BTS) which provides summary statistics for inbound crossings at the U.S. - Canada and the U.S. - Mexico border. For more info, check out the kaggle link.

Opening a .csv with the utils package function, read.csv()

```
data <- read.csv('Border_Crossing_Entry_Data.csv')
head(data)
```

```
##       Port.Name      State Port.Code           Border                   Date
## 1 Calexico East California      2507 US-Mexico Border 03/01/2019 12:00:00 AM
## 2     Van Buren      Maine       108 US-Canada Border 03/01/2019 12:00:00 AM
## 3     Otay Mesa California      2506 US-Mexico Border 03/01/2019 12:00:00 AM
## 4       Nogales    Arizona      2604 US-Mexico Border 03/01/2019 12:00:00 AM
## 5   Trout River   New York       715 US-Canada Border 03/01/2019 12:00:00 AM
## 6     Madawaska      Maine       109 US-Canada Border 03/01/2019 12:00:00 AM
##                         Measure Value                              Location
## 1                        Trucks 34447  POINT (-115.48433000000001 32.67524)
## 2        Rail Containers Full   428            POINT (-67.94271 47.16207)
## 3                        Trucks 81217           POINT (-117.05333 32.57333)
## 4                        Trains    62 POINT (-110.93361 31.340279999999996)
## 5 Personal Vehicle Passengers 16377  POINT (-73.44253 44.990010000000005)
## 6                        Trucks   179             POINT (-68.3271 47.35446)
```

```
class(data)
```

```
## [1] "data.frame"
```

We're reading the head of the dataframe to preview the dataset. We can see that the column names that have a space automatically replace the space with a `.` and we can also see the datatypes for each column right below.

Opening a .csv with the readr package function, read_csv()

```
data2 <- read_csv('Border_Crossing_Entry_Data.csv')
```

```
## Parsed with column specification:
## cols(
##   `Port Name` = col_character(),
##   State = col_character(),
##   `Port Code` = col_double(),
##   Border = col_character(),
##   Date = col_character(),
##   Measure = col_character(),
##   Value = col_double(),
##   Location = col_character()
## )
```

```
head(data2)
```

```
## # A tibble: 6 x 8
##   `Port Name`  State  `Port Code` Border   Date    Measure    Value Location
##   <chr>        <chr>        <dbl> <chr>    <chr>   <chr>      <dbl> <chr>
## 1 Calexico Ea~ Calif~        2507 US-Mexi~ 03/01/~ Trucks     34447 POINT (-115~
## 2 Van Buren    Maine          108 US-Cana~ 03/01/~ Rail Cont~   428 POINT (-67.~
## 3 Otay Mesa    Calif~        2506 US-Mexi~ 03/01/~ Trucks     81217 POINT (-117~
## 4 Nogales      Arizo~        2604 US-Mexi~ 03/01/~ Trains        62 POINT (-110~
## 5 Trout River  New Y~         715 US-Cana~ 03/01/~ Personal ~ 16377 POINT (-73.~
## 6 Madawaska    Maine          109 US-Cana~ 03/01/~ Trucks       179 POINT (-68.~
```

```
class(data2)
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

We'll notice a very similar output as the one above, but with slight variations. Our column names are true to the file in which there are spaces between words. Again, we see datatypes below the column names, but this time they are different types. They are no longer factors wherever the integers are. Since the `Date` column is not a date type, we'll convert that.

```
data2$Date <- as.Date(data2$Date, "%m/%d/%Y")
head(data2)
```

```
## # A tibble: 6 x 8
##   `Port Name`  State  `Port Code` Border  Date       Measure   Value Location
##   <chr>        <chr>        <dbl> <chr>   <date>     <chr>     <dbl> <chr>
## 1 Calexico Ea~ Calif~        2507 US-Mex~ 2019-03-01 Trucks    34447 POINT (-11~
## 2 Van Buren    Maine          108 US-Can~ 2019-03-01 Rail Con~   428 POINT (-67~
## 3 Otay Mesa    Calif~        2506 US-Mex~ 2019-03-01 Trucks    81217 POINT (-11~
## 4 Nogales      Arizo~        2604 US-Mex~ 2019-03-01 Trains       62 POINT (-11~
## 5 Trout River  New Y~         715 US-Can~ 2019-03-01 Personal~ 16377 POINT (-73~
## 6 Madawaska    Maine          109 US-Can~ 2019-03-01 Trucks      179 POINT (-68~
```

Select Function

Next, we'll use the `select()` function of the `dplyr` package to select specific and set of columns or variables desired.

We'll `select()` the Border, State, Port Name, Port Code.

Note: Since `Port Name` and `Port Code` have spaces inbetween them, we'll need to add quotes around those columns

```
border_data <- select(data2, Border, State, 'Port Name', 'Port Code')
head(border_data)
```

```
## # A tibble: 6 x 4
##   Border           State      `Port Name`   `Port Code`
##   <chr>            <chr>      <chr>               <dbl>
## 1 US-Mexico Border California Calexico East        2507
## 2 US-Canada Border Maine      Van Buren             108
## 3 US-Mexico Border California Otay Mesa            2506
## 4 US-Mexico Border Arizona    Nogales              2604
## 5 US-Canada Border New York   Trout River           715
## 6 US-Canada Border Maine      Madawaska             109
```

Filter Function

If we wanted to only look at data from this set where the border was `US-Mexico Border` only, then we can use the `filter()` function.

```
us_mex_data <- filter(data2, Border=='US-Mexico Border')
head(us_mex_data)
```

```
## # A tibble: 6 x 8
##   `Port Name`  State  `Port Code` Border   Date       Measure  Value Location
##   <chr>        <chr>        <dbl> <chr>    <date>     <chr>    <dbl> <chr>
## 1 Calexico Ea~ Calif~        2507 US-Mexi~ 2019-03-01 Trucks   34447 POINT (-11~
## 2 Otay Mesa    Calif~        2506 US-Mexi~ 2019-03-01 Trucks   81217 POINT (-11~
## 3 Nogales      Arizo~        2604 US-Mexi~ 2019-03-01 Trains      62 POINT (-11~
## 4 Progreso     Texas         2309 US-Mexi~ 2019-03-01 Truck C~  1808 POINT (-97~
## 5 San Ysidro   Calif~        2504 US-Mexi~ 2019-03-01 Bus Pas~  7779 POINT (-11~
## 6 Tecate       Calif~        2505 US-Mexi~ 2019-03-01 Truck C~  1993 POINT (-11~
```

Taking it a step further, if we wanted to filter for a specific date range, like April of 2014, then we can add , in the `filter()` function. Since this data set has rolled up data at a monthly level, we'll need to use an == statement for the April, 2014.

```
us_mex_data_2014_04 <- filter(data2, Border=='US-Mexico Border', Date == '2014-04-01')
head(us_mex_data_2014_04)
```

```
## # A tibble: 6 x 8
##   `Port Name` State  `Port Code` Border   Date       Measure   Value Location
##   <chr>       <chr>        <dbl> <chr>    <date>     <chr>     <dbl> <chr>
## 1 San Luis    Arizo~        2608 US-Mexi~ 2014-04-01 Bus Pass~     0 POINT (-11~
## 2 Hidalgo     Texas         2305 US-Mexi~ 2014-04-01 Rail Con~     0 POINT (-98~
## 3 Laredo      Texas         2304 US-Mexi~ 2014-04-01 Rail Con~ 20736 POINT (-99~
## 4 Roma        Texas         2310 US-Mexi~ 2014-04-01 Trucks      663 POINT (-99~
## 5 Del Rio     Texas         2302 US-Mexi~ 2014-04-01 Truck Co~  4393 POINT (-10~
## 6 Andrade     Calif~        2502 US-Mexi~ 2014-04-01 Buses         0 POINT (-11~
```

Arange Function

Next, we can arrange the data how we want in ascending or descending order using the `arrange()` function.

```
us_mex_data_2014_04 <- arrange(us_mex_data_2014_04, State)
head(us_mex_data_2014_04)
```

```
## # A tibble: 6 x 8
##   `Port Name` State  `Port Code` Border  Date       Measure   Value Location
##   <chr>       <chr>        <dbl> <chr>   <date>     <chr>     <dbl> <chr>
## 1 San Luis    Arizo~        2608 US-Mex~ 2014-04-01 Bus Pass~     0 POINT (-114~
## 2 Douglas     Arizo~        2601 US-Mex~ 2014-04-01 Buses       180 POINT (-109~
## 3 Lukeville   Arizo~        2602 US-Mex~ 2014-04-01 Bus Pass~   238 POINT (-112~
## 4 Douglas     Arizo~        2601 US-Mex~ 2014-04-01 Trains        0 POINT (-109~
## 5 Naco        Arizo~        2603 US-Mex~ 2014-04-01 Truck Co~   283 POINT (-109~
## 6 Lukeville   Arizo~        2602 US-Mex~ 2014-04-01 Pedestri~  4098 POINT (-112~
```

Piping multiple functions into one statement

We'll take everything we went through so far and combine it into one statement using piping %>% for the `US-Canada Border` in January, 2015.

```
us_can_data_2015_01 <- data2 %>% select(Border, State, 'Port Name', 'Port Code', Date) %>% filter(Borde

head(us_can_data_2015_01)
```

```
## # A tibble: 6 x 5
##   Border           State  `Port Name`  `Port Code` Date
##   <chr>            <chr>  <chr>              <dbl> <date>
## 1 US-Canada Border Alaska Dalton Cache        3106 2015-01-01
## 2 US-Canada Border Alaska Ketchikan           3102 2015-01-01
## 3 US-Canada Border Alaska Alcan               3104 2015-01-01
## 4 US-Canada Border Alaska Ketchikan           3102 2015-01-01
## 5 US-Canada Border Alaska Ketchikan           3102 2015-01-01
## 6 US-Canada Border Alaska Ketchikan           3102 2015-01-01
```