STAT 408: Week 8

Strings, Dates and Factors

3/8/2022

Baltimore Towing Data

An Overview

This dataset contains information on vehicles towed in Baltimore, MD:

- A larger version of this dataset along with additional descriptions can be found at: https://data.baltimorecity.gov/Transportation/DOT-Towing/k78j-azhn.
- The full version of the dataset contains 61,000 rows and 36 columns, where each row corresponds to a vehicle and the columns are information pertaining to the vehicle.
- We will be working with a smaller dataset with approximately 30,000 rows and 5 columns.

The dataset

First read in the data set which is available at: http://www.math.montana.edu/ahoegh/teaching/stat408/datasets/BaltimoreTowing.cs

```
baltimore_tow <- read_csv('http://www.math.montana.edu/ahoegh/teaching/stat40
str(baltimore_tow)
```

```
## spec_tbl_df [30,263 × 5] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ vehicleType      : chr [1:30263] "Van" "Car" "Car" "Car" ...
##  $ vehicleMake      : chr [1:30263] "LEXUS" "Mercedes" "Chysler" "Chevrolet
##  $ vehicleModel     : chr [1:30263] NA NA "Cirrus" "Cavalier" ...
##  $ receivingDateTime: chr [1:30263] "10/24/2010 12:41:00 PM" "04/28/2015 0
##  $ totalPaid        : chr [1:30263] "$322.00" "$130.00" "$280.00" "$1057.0
##  - attr(*, "spec")=
##   .. cols(
##   ..   vehicleType = col_character(),
##   ..   vehicleMake = col_character(),
##   ..   vehicleModel = col_character(),
##   ..   receivingDateTime = col_character(),
##   ..   totalPaid = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Information for a few vehicles

vehicleType	vehicleMake	vehicleModel	receivingDateTime	totalPaid
Van	LEXUS	NA	10/24/2010 12:41:00 PM	$322.00
Car	Mercedes	NA	04/28/2015 09:27:00 AM	$130.00
Car	Chysler	Cirrus	07/23/2015 07:55:00 AM	$280.00
Car	Chevrolet	Cavalier	10/23/2010 11:35:00 AM	$1057.00
Car	Hyundai	Tiburon	10/25/2010 02:49:00 PM	$469.00
SUV	Toyota	RAV4	10/25/2010 11:12:00 AM	$305.00
Car	Bmw	325	10/23/2012 07:50:00 PM	$220.00
Car	Honda	Accord	10/25/2010 02:53:00 PM	$327.00

Data Wrangling Concepts

Data Wrangling Concepts

- Dealing with strings
- Dealing with date/time objects
- Dealing with factors

Goal 1. Average Towing Cost by Month

Motivating Exercise: `group_by()`

Now compute the average towing cost grouped by month.

Not Solution: `group_by()`

Now compute the average towing cost grouped by month.

```
baltimore_tow %>%
  group_by(month) %>%
  summarize(mean.cost = mean(totalPaid))
```

Motivating Exercise: `group_by()`

Now compute the average towing cost grouped by month.

vehicleType	vehicleMake	vehicleModel	receivingDateTime	totalPaid
Van	LEXUS	NA	10/24/2010 12:41:00 PM	$322.00
Car	Mercedes	NA	04/28/2015 09:27:00 AM	$130.00
Car	Chysler	Cirrus	07/23/2015 07:55:00 AM	$280.00
Car	Chevrolet	Cavalier	10/23/2010 11:35:00 AM	$1057.00

`substr()` function

Consider adding a column for year to the data set. This can be done using `substr()`.

Usage: substr(x, start, stop)

Arguments:

- x, text a character vector.
- start, first integer. The first element to be extracted
- stop, last integer. The last element to be extracted

Exercise: Using the `substr()` function

Use the `substr()` function to extract month and create a new variable in R.

Solution: Using the `substr()` function

Use the `substr()` function to extract month and create a new variable in R.

```
baltimore_tow$month <- substr(baltimore_tow$receivingDateTime, 0, 2)
head(baltimore_tow$month)
```

```
## [1] "10" "04" "07" "10" "10" "10"
```

Motivating Exercise: `group_by()`

Now compute the average towing cost grouped by month.

```
baltimore_tow %>%
  group_by(month) %>%
  summarize(mean.cost = mean(totalPaid))
```

```
## # A tibble: 12 × 2
##    month mean.cost
##    <chr>     <dbl>
##  1 01           NA
##  2 02           NA
##  3 03           NA
##  4 04           NA
##  5 05           NA
##  6 06           NA
##  7 07           NA
##  8 08           NA
##  9 09           NA
## 10 10           NA
## 11 11           NA
## 12 12           NA
```


`strsplit()` function

In many situations, the year could be in a different position so the `substr()` might not work. For example month the date could be coded `4/1/2015` rather than `04/01/2015` So consider, using `strsplit()` instead.

Usage: strsplit(x, split)

Arguments:

- x: character vector, each element of which is to be split. Other inputs, including a factor, will give an error.
- split: character vector (or object which can be coerced to such) containing regular expression(s) (unless fixed = TRUE) to use for splitting.

Exercise: Using the `strsplit()` function

Use the `strsplit()` function to remove the dollar sign from the cost.

Solution: Using the `strsplit()` function

Use the `strsplit()` function to remove the dollar sign from the cost.

```
## example for one row
strsplit(baltimore_tow$totalPaid[1],'$', fixed = T)[[1]][2]
```

```
## [1] "322.00"
```

Lists

Data structure overview (review)

The base data structures in R can be organized by dimensionality and whether they are homogenous.

Dimension	Homogeneous	Heterogeneous
1d	Vector	List
2d	Matrix	Data Frame
no d	Array	

Lists

Consider the two lists

```
msu.info <- list(
  name = c('Waded Cruzado', 'Stacey Hancock'),
  degree.from = c('University of Texas at Arlington',
                  'Colorado State University'),
  job.title = c('President',
                'Associate Professor of Statistics'))

msu.info2 <- list(
  c('Waded Cruzado',
    'University of Texas at Arlington',
    'President'),
  c('Stacey Hancock',
    'Colorado State University',
    'Associate Professor of Statistics'))
```


List Output

```
msu.info
```

```
## $name
## [1] "Waded Cruzado"  "Stacey Hancock"
##
## $degree.from
## [1] "University of Texas at Arlington" "Colorado State University"
##
## $job.title
## [1] "President"                        "Associate Professor of Statistics"
```

```
msu.info2
```

```
## [[1]]
## [1] "Waded Cruzado"                    "University of Texas at Arlington"
## [3] "President"
##
## [[2]]
## [1] "Stacey Hancock"                   "Colorado State University"
## [3] "Associate Professor of Statistics"
```

Lists - indexing

With the current lists we can index elements using the double bracket `[[ ]]` notation or if names have been initialized, those can be used too.

So the first element of each list can be indexed

```
msu.info[[1]]
```

```
## [1] "Waded Cruzado"  "Stacey Hancock"
```

```
msu.info$name
```

```
## [1] "Waded Cruzado"  "Stacey Hancock"
```


Exercise: Lists

Explore the indexing with these commands.

```
msu.info[1]
msu.info[[1]]
msu.info$name[2]
msu.info[1:2]
unlist(msu.info)
```

Indexing lists

"If list `x` is a train carrying objects, then `x[[5]]` is the object in car 5; `x[4:6]` is a train of cars 4-6."

— @RLangTip

Source: http://adv-r.had.co.nz/Subsetting.html

Elements of lists need not be the same class or even dimension!

```
list(c("Jan","Feb","Mar"),
     matrix(c(3,9,5,1,-2,8), nrow = 2),
     list("green", 12.3)
     )
```

```
## [[1]]
## [1] "Jan" "Feb" "Mar"
##
## [[2]]
##      [,1] [,2] [,3]
## [1,]    3    5   -2
## [2,]    9    1    8
##
## [[3]]
## [[3]][[1]]
## [1] "green"
##
## [[3]][[2]]
## [1] 12.3
```

Solution: Using the `strsplit()` function (revisited)

Use the `strsplit()` function to remove the dollar sign from the cost.

```
strsplit(baltimore_tow$totalPaid[1:2], '$', fixed = T)[[1]][2]
```

```
## [1] "322.00"
```


lubridate package

lubridate package

lubridate is a tidyverse package for manipulating date objects. There is a nice website with a cheatsheet.

Date objects for Baltimore tow

```
library(lubridate) # loads with tidyverse
class(baltimore_tow$receivingDateTime)
```

```
## [1] "character"
```

```
baltimore_tow <- baltimore_tow %>%
  mutate(date_time = mdy_hms(receivingDateTime))
class(baltimore_tow$date_time)
```

```
## [1] "POSIXct" "POSIXt"
```

Date objects for Baltimore tow

```
head(month(baltimore_tow$date_time))
```

```
## [1] 10  4  7 10 10 10
```

```
head(year(baltimore_tow$date_time))
```

```
## [1] 2010 2015 2015 2010 2010 2010
```


Stringr Package

The `stringr` package (cheat sheet) provides a nice set of tools. There is also an information page.

Exercise: Stringr approach

Use the `stringr` package to remove (replace) the dollar sign. Note that a dollar sign is a special character, so you'll need to use `\\$`.

Solution: Stringr approach

Use the `stringr` package to remove (replace) the dollar sign

```
library(stringr)
baltimore_tow$cost <-
  as.numeric(str_replace(baltimore_tow$totalPaid, '\\$',''))
```

Motivating Exercise: `group_by()`

Now compute the average towing cost grouped by month.

```
baltimore_tow %>%
  group_by(month) %>%
  summarize(mean.cost = mean(cost), .groups = 'keep')
```

```
## # A tibble: 12 × 2
## # Groups:   month [12]
##    month mean.cost
##    <chr>     <dbl>
##  1 01         353.
##  2 02         349.
##  3 03         363.
##  4 04         347.
##  5 05         357.
##  6 06         346.
##  7 07         350.
##  8 08         350.
##  9 09         359.
## 10 10         343.
## 11 11         342.
## 12 12         344.
```


Goal 2. Type of Vehicles Towed by Month

Goal 2. Type of Vehicles Towed by Month

Next we wish to compute how many vehicles were towed for each vehicle type.

However, we want to take a close look at the vehicle types in the data set and perhaps create more useful groups.

`unique` function – how to group vehicles

First examine the unique types of vehicles in this data set.

```
unique(baltimore_tow$vehicleType)
```

```
##  [1] "Van"                          "Car"
##  [3] "SUV"                          "Pick-up Truck"
##  [5] "Motor Cycle (Street Bike)"    "Dirt Bike"
##  [7] "Commercial Truck"             "Trailer"
##  [9] "Station Wagon"                "Truck"
## [11] "Taxi"                         "Pickup Truck"
## [13] "Convertible"                  "Tractor Trailer"
## [15] "Tow Truck"                    "All terrain - 4 wheel bike"
## [17] "Mini-Bike"                    "Golf Cart"
## [19] "Boat"                         "Tractor"
## [21] "Construction Equipment"       "Sport Utility Vehicle"
```

Grouping

First consider reasonable groups for vehicle types.

1. Cars - (Car, convertible)
2. Large Cars - (SUV, Station Wagon, Sport Utility Vehicle, Van, Taxi)
3. Trucks - (Pick-up Truck, Pickup Truck)
4. Large Trucks - (Truck, Tractor Trailer, Tow Truck, Tractor, Construction Equipment, Commercial Truck)
5. Bikes - (Motor Cycle (Street Bike), Dirt Bike, All terrain - 4 wheel bike, Mini-Bike)
6. Misc (delete) - (Boat, Golf Cart, Trailer)

Messy Data: Grouping

Next examine values in some of these groups, we will just look at the vehicle type of 'Truck'.

```
unique(baltimore_tow$vehicleMake[baltimore_tow$vehicleType == 'Truck'])
```

```
##  [1] "GMC"           "Ford"          "Dodge"         "Freightliner"
##  [5] "Chevrolet"     "Izuzu"         "Toyota"        "Chevy"
##  [9] "Peterbilt"     "International" "Kenworth"      "Nissan"
## [13] "Mercedes"      "Isuzu"         "Frightliner"   "Mack"
## [17] "Sterling"      "Internantional" "Peterbelt"    "Pete"
## [21] "Hummer"        "Hino"
```

Note that there are several spelling errors in this data set. How do we combine them?

Messy Data: Data Cleaning

Spelling errors can be addressed, by reassigning vehicles to the correct spelling.

```
baltimore_tow$vehicleMake[baltimore_tow$vehicleMake ==
                          'Peterbelt'] <- 'Peterbilt'
baltimore_tow$vehicleMake[baltimore_tow$vehicleMake ==
                          'Internantional'] <- 'International'
baltimore_tow$vehicleMake <-
  str_replace(baltimore_tow$vehicleMake,'Izuzu','Isuzu')
baltimore_tow$vehicleMake <-
  str_replace(baltimore_tow$vehicleMake,'Frightliner','Freightliner')
```

Also note that many of the groupings have mis-classified vehicles, but we will not focus on that yet.

Exercise: Delete Misc. Type Vehicles

First we will delete golf carts, boats, and trailers. There are several ways to do this, consider making a new data frame called `balt_tow_small` that does not include golf carts, boats, and trailers.

Solution: Delete Misc. Type Vehicles

First we will delete golf carts, boats, and trailers.

```
balt_tow_small <- baltimore_tow %>%
  filter(!(vehicleType %in% c("Golf Cart", "Boat", "Trailer")))
```


Exercise: Create Additional Groups

Now we need to create a variable for the additional groups below.

1. Cars - (Car, convertible)
2. Large Cars - (SUV, Station Wagon, Sport Utility Vehicle, Van, Taxi)
3. Trucks - (Pick-up Truck, Pickup Truck)
4. Large Trucks - (Truck, Tractor Trailer, Tow Truck, Tractor, Construction Equipment, Commercial Truck)
5. Bikes - (Motor Cycle (Street Bike), Dirt Bike, All terrain - 4 wheel bike, Mini-Bike)

Solution: Create Additional Groups

One way to create groups is by creating a new variable

```
balt_tow_small$Group <- '' # Creates empty string for all rows in data set

balt_tow_small$Group[balt_tow_small$vehicleType %in%
                       c('Car','Convertible')] <- 'Cars'

balt_tow_small$Group[balt_tow_small$vehicleType %in% c('SUV',
 'Station Wagon','Sport Utility Vehicle','Van','Taxi')] <- 'Large Cars'

balt_tow_small$Group[balt_tow_small$vehicleType %in%
                       c('Pick-up Truck','Pickup Truck')] <- 'Trucks'

balt_tow_small$Group[balt_tow_small$vehicleType %in%
    c('Truck','Tractor Trailer','Tow Truck','Tractor',
      'Construction Equipment','Commercial Truck')] <- 'Large Trucks'

balt_tow_small$Group[balt_tow_small$vehicleType %in%
    c('Motor Cycle (Street Bike)','Dirt Bike','Mini-Bike',
      'All terrain - 4 wheel bike')] <- 'Bikes'
```


Solution:

Next we wish to compute how many vehicles were towed for each vehicle type

```
balt_tow_small %>% count(Group)
```

```
## # A tibble: 5 × 2
##   Group            n
##   <chr>        <int>
## 1 Bikes          383
## 2 Cars         19675
## 3 Large Cars    8575
## 4 Large Trucks   211
## 5 Trucks        1378
```

Factors

Factors…

Factors are a specific way to store categorical data. Using factors results in a more efficient data storage process, but can be cumbersome.

Factors can be necessary for making plots and fitting models in R.

forcats Package

The `forcats` package, website, is a tidyverse package designed for dealing with categorical factors.

Character Values

```
favorite_day <- c('Friday', 'Saturday', 'Sunday', 'Tuesday', 'Saturday', 'Satu
class(favorite_day)
```

```
## [1] "character"
```

Creating Factors

```
day_factor <- as.factor(favorite_day)
class(day_factor)
```

```
## [1] "factor"
```

```
sort(day_factor)
```

```
## [1] Friday   Saturday Saturday Saturday Sunday   Tuesday
## Levels: Friday Saturday Sunday Tuesday
```


Reordering Factors

```
library(forcats)
day_factor <- fct_relevel(day_factor,
                          c('Sunday','Tuesday','Friday','Saturday'))
sort(day_factor)
```

```
## [1] Sunday   Tuesday  Friday   Saturday Saturday Saturday
## Levels: Sunday Tuesday Friday Saturday
```

Creating Factors

Rather than coercing a class variable to be a factor, the factor can be created directly.

```
day_factor2 <- factor(c('Friday', 'Saturday', 'Sunday', 'Monday'),
                      levels = c('Sunday', 'Monday', 'Tuesday',
                        'Wednesday','Thursday','Friday','Saturday'))
sort(day_factor2)
```

```
## [1] Sunday   Monday   Friday   Saturday
## Levels: Sunday Monday Tuesday Wednesday Thursday Friday Saturday
```


Collapsing Factors

Factors can also easily be collapsed with `forcats`

```
balt_tow_small %>%
  mutate(Group2 = fct_collapse(vehicleType,
           Cars = c('Car','Convertible'),
           Large_Cars = c('SUV', 'Station Wagon',
                          'Sport Utility Vehicle',
                          'Van','Taxi'),
           Trucks = c('Pick-up Truck','Pickup Truck'),
           Large_Trucks = c('Truck', 'Tractor Trailer',
                            'Tow Truck', 'Tractor',
                            'Construction Equipment',
                            'Commercial Truck'),
           Bikes = c('Motor Cycle (Street Bike)', 'Dirt Bike',
                     'Mini-Bike','All terrain - 4 wheel bike')
           )
         ) %>%
  mutate(Group2 = fct_infreq(Group2)) %>%
  group_by(Group2) %>%
  summarize(ave_cost = mean(cost), .groups = 'drop')
```

```
## # A tibble: 5 × 2
##   Group2       ave_cost
##   <fct>           <dbl>
## 1 Cars             354.
```