# Dataset segmentation

Daniel Paliura

4/27/2021

## Purpose

Data-set “EPA CO daily summary” seems to not satisfy third principle of tidy data because it contains several entities in single table. There I gonna logically define which columns refers to special entities and separate the table in the file ‘epa_co_daily_summary.csv’ into few much smaller tables.

## Diagnostic

I previously viewed data so I set some numeric variables as factors and, also, figured out with dates.

I will rely on data-set description at first. It is already available on GitHub here and also at Kaggle data-set page here. Data preview is useful and so it is downwards.

```
## Classes 'data.table' and 'data.frame':   8064820 obs. of  29 variables:
##  $ state_code         : Factor w/ 53 levels "01","02","04",..: 5 36 27 47 5 29 31 20 44 7 ...
##  $ county_code        : Factor w/ 111 levels "001","002","003",..: 42 64 21 10 38 21 25 3 101 3 ...
##  $ site_num           : Factor w/ 273 levels "0001","0002",..: 5 28 16 20 260 141 3 173 210 17 ...
##  $ parameter_code     : Factor w/ 1 level "42101": 1 1 1 1 1 1 1 1 1 1 ...
##  $ poc                : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ latitude           : num  33.2 39.8 44.7 38.9 34 ...
##  $ longitude          : num  -117.4 -84.2 -111.1 -77.1 -117.4 ...
##  $ datum              : Factor w/ 4 levels "NAD27","NAD83",..: 1 2 4 4 4 4 4 2 2 4 ...
##  $ parameter_name     : Factor w/ 1 level "Carbon monoxide": 1 1 1 1 1 1 1 1 1 1 ...
##  $ sample_duration    : Factor w/ 2 levels "1 HOUR","8-HR RUN AVG END HOUR": 1 1 1 1 1 1 2 2 2 2 ...
##  $ pollutant_standard : Factor w/ 2 levels "CO 1-hour 1971",..: 1 1 1 1 1 1 2 2 2 2 ...
##  $ date_local         : Date, format: "1994-02-01" "1996-04-29" ...
##  $ units_of_measure   : Factor w/ 1 level "Parts per million": 1 1 1 1 1 1 1 1 1 1 ...
##  $ event_type         : Factor w/ 3 levels "Excluded","Included",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ observation_count  : int  24 24 23 24 23 24 24 24 24 24 ...
##  $ observation_percent: num  100 100 96 100 96 100 100 100 100 100 ...
##  $ arithmetic_mean    : num  1.129 0.533 0.157 0.287 0.322 ...
##  $ first_max_value    : num  1.9 1.2 0.3 0.8 0.6 0.8 0.4 0.2 0.2 1.3 ...
##  $ first_max_hour     : int  8 17 22 5 5 6 0 0 0 0 ...
##  $ aqi                : int  NA NA NA NA NA NA 5 2 2 15 ...
##  $ method_code        : Factor w/ 22 levels "","008","011",..: 12 13 15 13 13 15 1 1 1 1 ...
##  $ method_name        : Factor w/ 11 levels "-","INSTRUMENTAL - DUAL ISOTOPE FLORESCENCE",..: 8 8 3
##  $ local_site_name    : Factor w/ 781 levels "",".7 MILES E FROM OLD SITE ON S SIDE OF ST CHAS ROCK
##  $ address            : Factor w/ 1166 levels "??? QUEEN STREET AT MILLER PARK",..: 196 722 759 1072
##  $ state_name         : Factor w/ 53 levels "Alabama","Alaska",..: 5 37 28 49 5 30 32 21 46 7 ...
##  $ county_name        : Factor w/ 341 levels "Ada","Adair",..: 263 202 107 14 250 325 316 15 306 124
##  $ city_name          : Factor w/ 514 levels "Akron","Albany",..: 326 117 493 14 390 432 139 363 14
##  $ cbsa_name          : Factor w/ 248 levels "","Akron OH",..: 195 64 30 237 181 179 150 1 62 99 ...
##  $ date_of_last_change: Date, format: "2016-04-27" "2016-04-26" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

We can see that all numeric variables are about measurements of CO. Also it looks like many factor variables are about location (sites location). I read data-set description a few times and found out that all geodata is about sites where monitors placed.

There is such an hierarchy: Somewhere in some country there is site, which holds monitors fixing values of different substances (parameters). This site is placed in some county, in some city. Site can have many monitors to measure different parameters. And also it can be more than single monitor fixing same parameter, for insurance I guess. And this last defines variable `poc` which refers to different monitors for measuring same parameter.

Variable `parameter_name`, as `parameter_name`, refers to measurements (measured parameter). They has single value, but I guess it's just because following data-set was taken from EPA data base. And they are measuring not only CO. Variable `datum` still appears to be strange. I read about its purposes and it's useful. It's about site location. Variables `sample_duration` and `polutant_standard` are about the same and go to observations table. I shall make a remark: under observations and measurements I understand almost the same - fixed data about CO levels and about how it was fixed, so observations is not rows in my sense. Column `cbsa_name` says about site itself. And last one `date_of_last_change` is something about measurement values changing in data base. It was said in description that it's about last change of numeric values. If codes are meant as not numeric, so it is right conclusion.

Not described variables aren't need to be described. They say themselves for themselves.

## Division with key variables

I gonna divide data-set into two tables at first: observations summary and sites info. Observations summary table will contain only data about measurements. It is clearer to say that sites info will represent only information about unique sites making measurements. It will contain information about sites placements from name of state to latitude and longitude.

Also I must say about relation between tables. As it is did with data bases, two related tables must be tied with some key. It is needed because if I want to restore information about site, which measured some interesting value, than I have to know exactly which site it is. There is no such a variables as it would be in data base like `site_id` and `measurement_id`. Second one is excessive because relation is one to many from sites table to observations table but I won't create such a variable, because I have fixed data. So I just have to find sensible set of variables providing uniqueness in table sites and add this variables also to table observations to provide relationship.

### Main division

So I decided such a division:

1) Table observations has 17 variables:

    - `parameter_code`
    - `poc`
    - `parameter_name`
    - `sample_duration`
    - `pollutant_standard`
    - `date_local`
    - `units_of_measure`

- `event_type`
- `observation_count`
- `observation_percent`
- `arithmetic_mean`
- `first_max_value`
- `first_max_hour`
- `aqi`
- `method_code`
- `method_name`
- `date_of_last_change`

2) Table sites has 12 variables:

- `state_code`
- `county_code`
- `site_num`
- `latitude`
- `longitude`
- `datum`
- `local_site_name`
- `address`
- `state_name`
- `county_name`
- `city_name`
- `cbsa_name`

OK. Now I have to found set of variables suitable as primary key of table sites. I guess it could be `state_code, county_code, site_num` or `latitude, longitude`. To see if so I will create a sub-able with mentioned 12 variables and unique rows. After that I will try to find such a combinations of variables that provide same number of unique observations as created subtable. Let's start:

```
# Uses dplyr package
sites <- unique(DT %>% select(state_code, county_code, site_num,
                              latitude, longitude, datum,
                              local_site_name, address,
                              state_name, county_name, city_name,
                              cbsa_name))
dim(sites)
```

```
## [1] 1304   12
```

Let's check mentioned two combinations, whether they cover all sites. I shall try use `latitude` and `longitude` without and with `datum` to check how is it needed in key.

```
first_key_uniq <- unique(DT %>% select(state_code, county_code, site_num))
second_key_uniq <- unique(DT %>% select(latitude, longitude))

dim(first_key_uniq)
```

```
## [1] 1162    3
```

```
dim(second_key_uniq)
```

```
## [1] 1180    2
```

They both don't cover all unique elements. I propose to use also `datum` variable because it has short values for difference to `address` and different names.

```
first_key_uniq <- unique(DT %>% select(state_code, county_code, site_num, datum))
second_key_uniq <- unique(DT %>% select(latitude, longitude, datum))

dim(first_key_uniq)
```

```
## [1] 1299    4
```

```
dim(second_key_uniq)
```

```
## [1] 1298    3
```

It wasn't expected that `datum` variable in key variables set will make set with codes better than set with coordinates. Now I can check such an observations in sites table that are different but has not unique key variables. I decided to choose set of variables with codes as key due to pair of reasons:

1) Logical purposes: codes could form something like and ID or hash, so it makes sense. And also it is intuitive - everyone, I guess, will search site in sites table by state and county codes an site number instead of location coordinates.
2) Storage purpose: It might not be a rule, but let's take a look on how key variables are written inside .csv file: `state_code, county_code, site_num`: "XX,XXX,XXXX" - 11 characters; `latitude, longitude`: "???X.XXXXXX,???X.XXXXXX" - from 15 to 21 characters. Here 'X' means a number, '?' means optional symbol (could be sign '-' or number). `datum` is ignored because it does not make any valuable difference in both cases. So set `state_code, county_code, site_num, datum` will be primary key to chose unique site and will be used as foreign key in observations table. Now we have to find out why this set doesn't provide complete uniqueness.

To find it out, let's take a look at different sites with single key: To do so I will make a loop over all unique values for chosen set, I mean, over data-frame `first_key_uniq`. I shall rename it. Let's do so and see searched observations (I will call them `confuses`):

```
##     state_code county_code site_num latitude longitude datum
##  1:         44         007     1010 41.84157 -71.36077 NAD83
##  2:         44         007     1010 41.84104 -71.36097 NAD83
##  3:         47         157     0024 35.15119 -90.04156 WGS84
##  4:         47         157     0024 35.15119 -90.04156 WGS84
##  5:         12         057     3002 27.96565 -82.23040 WGS84
##  6:         12         057     3002 27.96565 -82.23040 WGS84
##  7:         29         510     0085 38.65650 -90.19865 NAD83
##  8:         29         510     0085 38.65643 -90.19835 NAD83
##  9:         12         086     0034 25.68333 -80.39972 NAD83
## 10:         12         086     0034 25.68333 -80.40000 NAD83
##                   local_site_name
##  1: FRANCIS SCHOOL East Providence
##  2: FRANCIS SCHOOL East Providence
##  3:
##  4:          Alabama Ave. Station
##  5:                        SYDNEY
##  6:                        SYDNEY
##  7:                  Blair Street
##  8:                  Blair Street
##  9:                       KENDALL
## 10:
##                                                        address   state_name
##  1:                               FRANCIS SCHOOL 64 BOURNE AVE Rhode Island
##  2:                               FRANCIS SCHOOL 64 BOURNE AVE Rhode Island
##  3:                                         416 ALABAMA AVENUE    Tennessee
##  4:                                         416 ALABAMA AVENUE    Tennessee
##  5:                         1167 N Dover Road Valrico FL 33527      Florida
##  6:                                     1167 NORTH DOVER ROAD      Florida
##  7:      BLAIR STREET: 3247 Blair Street St. Louis MO 63107     Missouri
##  8:      BLAIR STREET: 3247 Blair Street St. Louis MO 63107     Missouri
##  9:                            9015 SW 127th Ave Miami FL 33186      Florida
## 10: NW CORNER OF INTERSECTION OF SW 88 ST AND SW 127 AVENUE      Florida
##        county_name       city_name                                cbsa_name
##  1:     Providence East Providence                Providence-Warwick RI-MA
##  2:     Providence East Providence                Providence-Warwick RI-MA
##  3:         Shelby         Memphis                        Memphis TN-MS-AR
##  4:         Shelby         Memphis                        Memphis TN-MS-AR
##  5:   Hillsborough         Valrico       Tampa-St. Petersburg-Clearwater FL
##  6:   Hillsborough         Valrico       Tampa-St. Petersburg-Clearwater FL
##  7: St. Louis City       St. Louis                           St. Louis MO-IL
##  8: St. Louis City       St. Louis                           St. Louis MO-IL
##  9:     Miami-Dade   Not in a city Miami-Fort Lauderdale-West Palm Beach FL
## 10:     Miami-Dade   Not in a city Miami-Fort Lauderdale-West Palm Beach FL
```

There is just an errors of data-set formation. Let's fix data successively. First pair differs only by `latitude` and `longitude`. We have to check which value is mistake

```
## Warning in min.default(structure(numeric(0), class = "Date"), na.rm = FALSE): no
## non-missing arguments to min; returning Inf

## Warning in max.default(structure(numeric(0), class = "Date"), na.rm = FALSE): no
## non-missing arguments to max; returning -Inf

## # A tibble: 0 x 4
## # ... with 4 variables: latitude <chr>, count <int>, first_date <date>,
## #   last_date <date>
```

I guess, latitude changed due to tectonic plates movements and it was fixed at very beginning of 2012. I guess it won't make a big problem if I set coordinates as old values. I checked this site in Google Maps and it says that it placed 41.84188 by latitude nowadays. It seems to be not a big issue. So I will change latitude and longitude after 2011 to values as before 2012.

```
cond1 <- DT$state_code==44 & DT$county_code==7 &
         DT$site_num==1010 & DT$datum=="NAD83"
new_latitude <- DT$latitude
```

```
new_latitude[cond1] <- 41.841573
new_longitude <- DT$longitude
new_longitude[cond1] <- -71.360770
```

Second pair is OK, but first observation don't have a `local_site_name` value, so I just gonna add it:

```
cond2 <- DT$state_code==47 & DT$county_code==157 &
         DT$site_num==24 & DT$datum=="WGS84"
new_local_site_name <- DT$local_site_name
new_local_site_name[cond2] <- "Alabama Ave. Station"
```

Third pair has different addresses. Such and addresses looks very confusing to me. I will try to figure out more actual one. Also Google Maps just confusing me somewhy. I guess it would be correct something like

```
## Warning in min.default(structure(numeric(0), class = "Date"), na.rm = FALSE): no
## non-missing arguments to min; returning Inf
```

```
## Warning in max.default(structure(numeric(0), class = "Date"), na.rm = FALSE): no
## non-missing arguments to max; returning -Inf
```

```
## # A tibble: 0 x 4
## # ... with 4 variables: address <fct>, count <int>, first_date <date>,
## #   last_date <date>
```

I gonna use last address. Also it has number 32527, which referred to more actual point according to coordinates, when searching "1167 NORTH DOVER ROAD". So I change address to latest:

```
cond3 <- DT$state_code==12 & DT$county_code==57 &
         DT$site_num==3002 & DT$datum=="WGS84"
new_address <- DT$address
new_address[cond3] <- "1167 N Dover Road Valrico FL 33527"
```

Fourth pair has regular coordinates confuse. Bless God for not many such ones.

```
## Warning in min.default(structure(numeric(0), class = "Date"), na.rm = FALSE): no
## non-missing arguments to min; returning Inf
```

```
## Warning in max.default(structure(numeric(0), class = "Date"), na.rm = FALSE): no
## non-missing arguments to max; returning -Inf
```

```
## # A tibble: 0 x 4
## # ... with 4 variables: latitude <chr>, count <int>, first_date <date>,
## #   last_date <date>
```

Difference is small and It seems to me that it doesn't matter what change: old values to new or new values to old. Difference is at fifth and fourth numbers after point. I will do as I done in previous case: change old values to a new values.

```r
cond4 <- DT$state_code==29 & DT$county_code==510 &
         DT$site_num==85 & DT$datum=="NAD83"
new_latitude[cond4] <- 38.656498
new_longitude[cond4] <- -90.198646
```

And last one pair. Google Maps shows different places and it can be because of monitor relocation. Let's see summary statistics:

```
## Warning in min.default(structure(numeric(0), class = "Date"), na.rm = FALSE): no
## non-missing arguments to min; returning Inf

## Warning in max.default(structure(numeric(0), class = "Date"), na.rm = FALSE): no
## non-missing arguments to max; returning -Inf

## # A tibble: 0 x 4
## # ... with 4 variables: longitude <chr>, count <int>, first_date <date>,
## #   last_date <date>
```

It is more observations are before 2012, but in this case I gonna set values of 2012 and after years because it is set up an local site name and address is foundable via Google Maps, so:

```r
cond5 <- DT$state_code==12 & DT$county_code==86 &
         DT$site_num==34 & DT$datum=="NAD83"
new_longitude[cond5] <- -80.399722
new_local_site_name[cond5] <- "KENDALL"
new_address[cond5] <- "9015 SW 127th Ave Miami FL 33186"
```

And last thing - set new values into table and divide table into two:

```r
DT$latitude <- new_latitude
DT$longitude <- new_longitude
DT$local_site_name <- new_local_site_name
DT$address <- new_address

sites <- DT %>%
    select(state_code, county_code, site_num,
           latitude, longitude, datum,
           local_site_name, address,
           state_name, county_name, city_name, cbsa_name) %>%
    unique

observations <- DT %>%
    select(-latitude, -longitude, -datum,
           -local_site_name, -address,
           -state_name, -county_name, -city_name, -cbsa_name)
```

check now for unique sites:

```r
dim(sites)
```

```
## [1] 1304   12
```

```r
dim(unique(select(sites,
                  state_code, county_code, site_num, datum)))
```

```
## [1] 1299    4
```

Numbers of rows are the same so I separated table of sites. All code will be moved into script. Current R Markdown does not write any files.

#### Subdivision

Second step: to divide table of observations into two. I found out that table contains information both of measurements and about method of measurements. I expect that variable `method_code` will be enough to divide observations table into mentioned twain. So check it:

```r
methods <- observations %>%
    select(method_code, method_name,
           parameter_code, parameter_name, units_of_measure,
           sample_duration, pollutant_standard) %>%
    unique

print(dim(methods))
```

```
## [1] 22  7
```

```r
print(length(unique(methods$method_code)))
```

```
## [1] 22
```

So `method_code` variable is enough. Let's write observations table itself withouts methods info:

## Division with ID's

This division will appear first in script and it will save data untouched. I will appoint unique numbers to unique rows in sites table just before changing values in columns. And also I won't include codes into observations table. I will add special column `site_id` instead. And same thing I will do for methods. I will replace `method_code` with `method_id`.

## Conclusion

I made two data-set divisions. In first one I violated principle of data tiding. I rewrote some values in table. To fix such a violation I made also data division using IDs. To figure out sense of this division I reveal comaprisons.

## Comparison

And what we have now? Let's see dimensions:

```
## [1] 8064820      29
```

```
## [1] 8064820      14
```

```
## [1] 1304   12
```

```
## [1] 22  7
```

Size of data reduced twice in sense of dimensions. What about memory?

```
## 1161639328 bytes
```

```
## 612962160 bytes
```

```
## 367840 bytes
```

```
## 9304 bytes
```

The size decreased almost twice. It was worth it, I think. I expect better work with smaller data-set.