Repeated sales/rent walkthrough

THIS FILE ONLY COMPILES DURING THE PIPELINE; ERRORS IF DONE MANUALLY! Introduction outlining the idea and process..

Details of reading/reprocessing the RED data can be found in R/read__/, specifically read__RED.R and prepare__RED.R. These should be self explanatory. Note that the classification algorithm is designed to be run in parallel.[1] This is achieved by grouping the RED data on "blid", i.e. one group for each federal state. These groups are than classified in parallel for significant speedups. This classification digs down to coordinate-level, which is where i will start explaining the actual procedure.

Also dropped filtered and dropped balkon— The example data is taken from a coordinate with 25 observations of federal state Bremen. Since the data is already subset to the required dissolution for the next steps, the variables "blid" and "latlon__utm" were dropped beforehand (See R/misc/make__example__markdown__data.R for details).

```
## load example data
tar_load(example_markdown_data)
#make this dynamic
#tar_load(example_markdown_data, store ="N:/FDZ/Intern/HiWi-Praktikanten/Mitarbeiter/Thorben/repeated o
```

#General makeup of the data:

```
# show head
head(example_markdown_data)
```

```
##    wohnflaeche zimmeranzahl etage counting_id amonths emonths price_var
## 1:          71            3     2     1026832   24086   24088    135200
## 2:          56            2     1     1026833   24086   24088    102300
## 3:          76            3     6     1026834   24086   24089    139900
## 4:          71            3     2     1027667   24089   24100    142900
## 5:          56            2     1     1027668   24089   24100    107800
## 6:          82            3     2     1029808   24095   24097    164900
```

```
# show summary
summary(example_markdown_data)
```

```
##   wohnflaeche     zimmeranzahl      etage       counting_id
##  Min.   : 56.0   Min.   :2.00   Min.   :0.00   Min.   :1026832
##  1st Qu.: 71.0   1st Qu.:3.00   1st Qu.:1.00   1st Qu.:1030454
##  Median : 80.0   Median :3.00   Median :2.00   Median :1032333
##  Mean   : 80.0   Mean   :2.96   Mean   :2.24   Mean   :1033371
##  3rd Qu.: 81.0   3rd Qu.:3.00   3rd Qu.:3.00   3rd Qu.:1034542
##  Max.   :149.9   Max.   :5.00   Max.   :6.00   Max.   :1045696
##     amonths         emonths        price_var
##  Min.   :24086   Min.   :24088   Min.   :102300
##  1st Qu.:24097   1st Qu.:24100   1st Qu.:138000
##  Median :24102   Median :24104   Median :139900
##  Mean   :24104   Mean   :24108   Mean   :155220
```

[1] I recommend using 'tar__make__future(workers = n)', where n is the number of cores. $n = 4$ is a decent value, when no one else is using the server the number can be set higher. More than 8 is overkill, since Berlin alone typically bottlenecks. If speed becomes a big issue with growing data consider using "foreach()" on the coordinate level.

```
##  3rd Qu.:24107   3rd Qu.:24117   3rd Qu.:155000
##  Max.   :24134   Max.   :24135   Max.   :335700
```

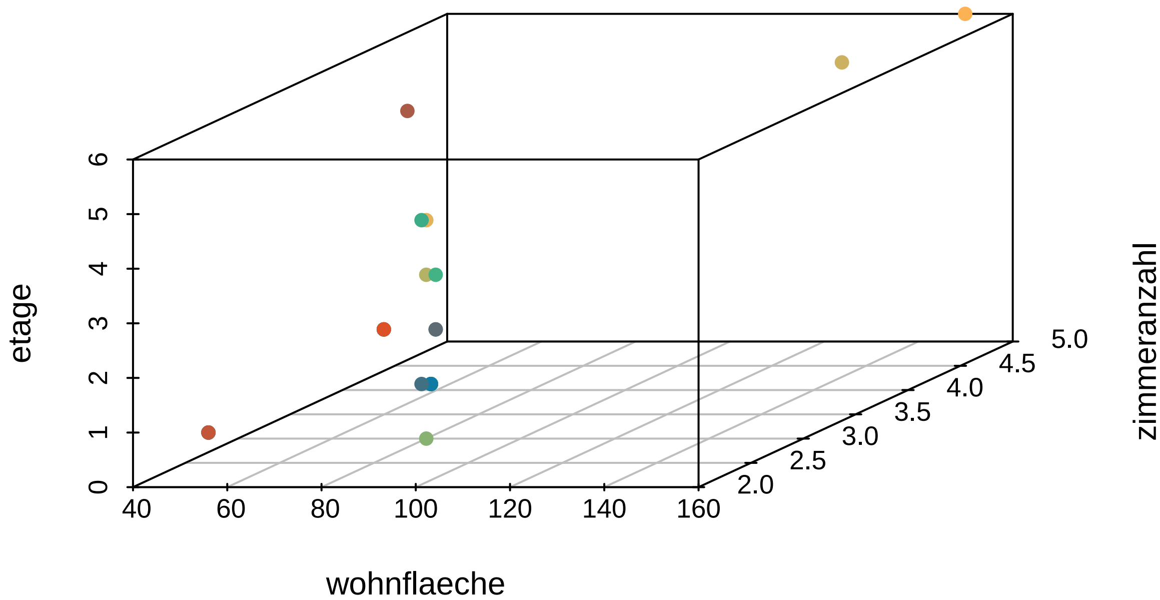
There are two types of similarity which will be considered: resembling and exact. The former allows for slightly larger deviations, the latter is quite restrictive. "r_o" refers to resembeling offset and "e_o" to exact offset.

parameter used for classification:

```r
# these are globally defined in _targets.R
print(exportJSON)
```

```
##    RED_type RED_version   categories wohnflaeche_r_o etage_r_o zimmeranzahl_r_o
## 1:       WK          v9  wohnflaeche             0.1         1              0.5
## 2:       WK          v9        etage             0.1         1              0.5
## 3:       WK          v9 zimmeranzahl             0.1         1              0.5
##    wohnflaeche_e_o zimmeranzahl_e_o time_offset
## 1:            0.05              0.5           6
## 2:            0.05              0.5           6
## 3:            0.05              0.5           6
```

```r
# make color blind friendly palette
etage_colors = MetBrewer::met.brewer("Egypt", n = length(example_markdown_data$etage))
with(
  example_markdown_data,
  scatterplot3d(x = wohnflaeche , y = zimmeranzahl, z =  etage , color = etage_colors, pch = 16)
)
```



Note that direct overlaps naturally arent visible. The challenge now is to make a decision: are the lisings for two different apartments or are they the same apartment? To make this decision for any number of characteristic combinations, I use a modified version of k-nearest neighbors clustering. Before we can get to this stage however, it is necessary to define and approach the issue formally. Visually classifying each combination at each coordinate would take quite some time (and would have to be re-done for every new wave, more on that later).

We will proceed in two overarching dimensions: the characteristics dimension (classifying similarity) and the subsequent time dimension ('classifying' non list reason):

characteristics dimension

```
# save ordering of ids
occurence_ids <- example_markdown_data[, counting_id]

similarity_lists = make_similarity_lists(example_markdown_data,occurence_ids)
```

index

```
similarity_index_list = similarity_lists[[1]]
similarity_index_list[1:5,1:5]
```

```
##    1026832 1026833 1026834 1027667 1027668
```

```
## 1:       0      NA      NA       0      NA
## 2:      NA       0      NA      NA       0
## 3:      NA      NA       0      NA      NA
## 4:       0      NA      NA       0      NA
## 5:      NA       0      NA      NA       0
```

similarity distances

```r
similarity_dist_list = similarity_lists[[2]]
similarity_dist_list[1:5,1:5]
```

```
##       1026832    1026833   1026834   1027667    1027668
## 1: 0.00000000 0.82169095 0.5761836 0.0538924 0.79771737
## 2: 0.57102726 0.00000000 0.8733848 0.5895578 0.05102946
## 3: 1.33564426 2.60050791 0.0000000 1.3353572 2.59156730
## 4: 0.05696078 0.85396470 0.5756037 0.0000000 0.82326469
## 5: 0.55491380 0.05377204 0.8620993 0.5720039 0.00000000
```

```r
    similarity_dist_list[is.na(similarity_index_list)] = NA

    # setup and run the actual clustering
    clustering <- cluster$new(
      cluster_options = similarity_index_list,
      distance = similarity_dist_list
    )
    clustering$determine_cluster_centers()
    clustering$centers |> head(n = 10)
```

```
##     counting_id  parent   sim_dist sim_index
##  1:     1026832 1026832 0.00000000         0
##  2:     1027667 1026832 0.05696078         0
##  3:     1031914 1026832 0.03512732         0
##  4:     1034541 1026832 0.03559309         0
##  5:     1038183 1026832 0.01579443         0
##  6:     1026832 1027667 0.05389240         0
##  7:     1027667 1027667 0.00000000         0
##  8:     1031914 1027667 0.02140384         0
##  9:     1034541 1027667 0.02205500         0
## 10:     1038183 1027667 0.04460554         0
```

```r
clustering$centers <- clustering$centers[
        ,
        similarity_cost_function(.SD)
      ]
clustering$centers |>  head(n = 10)
```

```
##     counting_id  parent    sim_dist sim_index
##  1:     1026832 1026832 0.000000000         0
##  2:     1027667 1027667 0.000000000         0
##  3:     1031914 1034541 0.002578610         0
##  4:     1034541 1034541 0.000000000         0
##  5:     1038183 1038183 0.000000000         0
##  6:     1026833 1026833 0.000000000         0
##  7:     1027668 1027668 0.000000000         0
```

```
## 8:     1031915 1034542 0.002599517         0
## 9:     1034542 1034542 0.000000000         0
## 10:    1026834 1026834 0.000000000         0
```

```
# example_markdown_data is equvialent to geo_grouped_data in the code
out <- example_markdown_data[
    clustering$centers,
    on = .(counting_id)
  ]
out |>  head(n = 10)
```

```
##     wohnflaeche zimmeranzahl etage counting_id amonths emonths price_var
##  1:          71            3     3     1026832   24086   24088    135200
##  2:          71            3     3     1027667   24089   24100    142900
##  3:          71            3     3     1031914   24101   24106    139900
##  4:          71            3     3     1034541   24107   24115    139900
##  5:          71            3     3     1038183   24116   24127    136700
##  6:          56            2     2     1026833   24086   24088    102300
##  7:          56            2     2     1027668   24089   24100    107800
##  8:          56            2     2     1031915   24101   24106    106800
##  9:          56            2     2     1034542   24107   24118    106800
## 10:          76            3     7     1026834   24086   24089    139900
##       parent    sim_dist sim_index
##  1: 1026832 0.000000000         0
##  2: 1027667 0.000000000         0
##  3: 1034541 0.002578610         0
##  4: 1034541 0.000000000         0
##  5: 1038183 0.000000000         0
##  6: 1026833 0.000000000         0
##  7: 1027668 0.000000000         0
##  8: 1034542 0.002599517         0
##  9: 1034542 0.000000000         0
## 10: 1026834 0.000000000         0
```