06 Data Transformation

Fabian Blasch

06/01/2022

1 Load Data

```
# source AUX
source("./../Misc/Auxilliary.R")

# packages
get.package(c("bizdays", "lubridate", "tidytext", "stopwords", "stringi",
              "SnowballC", "wordcloud"))

# load data
auctions <- readRDS("./../../Data/Bid Tab RDS/Bid_Tab_Stem.RDS")
```

2 Tabularizing the Data

Firstly, we may construct the data that is easily obtainable via the table and text data.

```
# over years
lapply(auctions, \(y){

  lapply(y, \(a){

    # apply function
    d_transform(a) |> try() # apparently there are three auctions with empty tables
                            # that slipped through
  })

}) -> res
```

```
## Error in data.frame(..., check.names = FALSE) :
##   Argumente implizieren unterschiedliche Anzahl Zeilen: 1, 0
## Error in data.frame(..., check.names = FALSE) :
##   Argumente implizieren unterschiedliche Anzahl Zeilen: 1, 0
## Error in data.frame(..., check.names = FALSE) :
##   Argumente implizieren unterschiedliche Anzahl Zeilen: 1, 0
```

```
# remove auctions that slipped
# over years
lapply(res, \(y){

  sapply(y, \(a){

    # apply function
    class(a) != "try-error" # apparently there are three auctions with empty tables
                            # that slipped through
```

```
  })

}) -> ind_slip

# remove from auctions list
Map(\(au, ind) au[ind], auctions, ind_slip) -> auctions

# remove from table list
Map(\(au, ind) au[ind], res, ind_slip) -> res

# rbind to dataframe
do.call(rbind, lapply(res, \(x) do.call(rbind, x))) -> dat_bids
```

3 Adding the Description via Stemwords

To represent the description, each stem word will be added as a factor.

```
# fetch vector of stemmed words from list
lapply(auctions, \(y){

  # over auctions
  lapply(y, \(a) a[["Stem"]])

}) -> stems

# generate all unique words ordered by frequency
do.call(c, lapply(stems, \(x) do.call(c, x))) |> table() |>
  sort(decreasing = TRUE) -> stem_tab

# remove all that occur in only one auction
stem_tab <- stem_tab[stem_tab > 1] # we don't want those words to be auction identifying

# loop over years
do.call(rbind, lapply(stems, \(y){

  # loop over auctions
  do.call(rbind,lapply(y, \(a){

    # match
    names(stem_tab) %in% a

  }))

})) |> as.data.frame() |> setNames(names(stem_tab)) -> stem_vars

# use rownames (contract ID for merging)
stem_vars <- cbind("Contract_ID" = row.names(stem_vars), stem_vars)

# merge on Contract ID
dat_bids <- merge(dat_bids, stem_vars, by = "Contract_ID", all.x = TRUE)
```

4 Variable Classes

```
# classes
# sapply(dat_bids, class)

# to factor
f1 <- names(dat_bids) %in% c("Contract_ID", "County", "Letting_Month", "Letting_Year",
                             "Vendor_ID", "Vendor_Name")
# asssign
dat_bids[, f1] <- lapply(dat_bids[, f1], as.factor)

# convert all logical to numeric
dat_bids <- rapply(dat_bids, as.numeric, classes = "logical", how = "replace")

# contract time to numerc
dat_bids[, "Contract_Time"] <- as.numeric(dat_bids[, "Contract_Time"])

# write
# saveRDS(dat_bids, "./../../Data/Bid Tab RDS/Bids_df.RDS")
```

5 Training and Test Set

For final out of sample performance evaluation 20% of the bids will be sampled from the data set.

```
# seed
{set.seed(33)
ind <- sample(1:nrow(dat_bids), replace = FALSE,
              size = floor(nrow(dat_bids) * 0.2))
}

# train and test
dat_bids_split <- list("Train" = dat_bids[!(1:nrow(dat_bids) %in% ind), ],
                       "Test" = dat_bids[ind , ])

# write
# saveRDS(dat_bids, "./../../Data/Bid Tab RDS/Bids_df_split.RDS")
```