# 06 Data Transformation

Fabian Blasch

06/10/2022

## 1 Load Data

```r
# 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 Data Set for Winning Bid Prediction

### 2.1 Tabularizing the Data

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

```r
# 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
```

## 2.2 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
```

```r
  }))

})) |> 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)
```

## 2.3 Variable Classes

```r
# 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")
```

## 2.4 Training and Test Set

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

```r
# 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_split, "./../../Data/Bid Tab RDS/Bids_df_split.RDS")
```

## 3 Data Set for Adjusted Bid Spread Prediction

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

  lapply(y, \(a){

    # apply function
    d_transform_2(a) |> try() # apparently there are three auctions with empty tables
                              # that slipped through
  })

}) -> tmp_a

# dat auc
do.call(rbind, lapply(tmp_a, \(x) do.call(rbind, x))) |> as.data.frame() -> dat_auc
```

```
## Warning in (function (..., deparse.level = 1) : number of columns of result is
## not a multiple of vector length (arg 41)

## Warning in (function (..., deparse.level = 1) : number of columns of result is
## not a multiple of vector length (arg 6)

## Warning in (function (..., deparse.level = 1) : number of columns of result is
## not a multiple of vector length (arg 12)

## Warning in (function (..., deparse.level = 1) : number of columns of result is
## not a multiple of vector length (arg 17)
```

```r
# fetch bidder ids
lapply(auctions, \(y){

  # over auctions
  lapply(y, \(a){

    # bidders
    tmp <- a[["Table"]][, "Vendor_ID"]
    tmp[!grepl(".*EST.*", tmp)]

  })

}) -> ids

# to single vector
do.call(c, lapply(ids, \(x) do.call(c, x))) |> unique() -> id_vec


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

    # ids without est.
    tmp <- a[["Table"]][, "Vendor_ID"]
    tmp[!grepl(".*EST.*", tmp)]

    # match
    id_vec %in% tmp

  }))

})) |> as.data.frame() |> setNames(id_vec) -> id_vars

# generate dataset
test <- auctions[["2015"]][[41]] # has no second place bid ? - remove

# rm
dat_auc <- dat_auc[-41, ]

# incomplete cases
dat_auc_c <- dat_auc[complete.cases(dat_auc), ]

# unrealistic results
dat_auc_c <- dat_auc_c[as.numeric(dat_auc_c$MLOT) < 1, ]
```

## 3.1 Merge

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

# merge on Contract ID
dat_auc_c <- merge(dat_auc_c, id_vars, by = "Contract_ID", all.x = TRUE)

# merge stems
dat_auc_c <- merge(dat_auc_c, stem_vars, by = "Contract_ID", all.x = TRUE)
```

## 3.2 Classes

```r
# to factor
f1 <- names(dat_auc_c) %in% c("Contract_ID", "County", "Letting_Month", "Letting_Year")

# asssign
dat_auc_c[, f1] <- lapply(dat_auc_c[, f1], as.factor)

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

# to numeric
```

```r
n1 <- c("Contract_Time", "N_Firm", "Eng_Est", "EW_Diff","MLOT")

# contract time to numerc
dat_auc_c[, n1] <- lapply(dat_auc_c[, n1], as.numeric)

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

### 3.3 Save

```r
# write full
# saveRDS(dat_auc_c, "./../../Data/Bid Tab RDS/Aucs_df.RDS")

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

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

# write
# saveRDS(dat_aucs_split, "./../../Data/Bid Tab RDS/Aucs_df_split.RDS")
```

## 4 Competitor Extension to Bids Data Set

```r
# set logical to numeric
id_vars_c <- data.frame("Contract_ID" = id_vars[, 1],
                        lapply(id_vars[, -1], as.numeric))
# rm x
colnames(id_vars_c) <- stringr::str_remove(colnames(id_vars_c), "X")


# add identification to bids dataset
dat_bids_id <- merge(dat_bids, id_vars_c, by = "Contract_ID", all.x = TRUE)

# remove bidders from competition factors
sub <- dat_bids_id[, c(279:ncol(dat_bids_id))]
VID <- dat_bids_id[, "Vendor_ID"]

# correct foactor levels
Map(\(x, n){

  # if VID is col.name then remove Vendor ID as this is the bidder and not
  # a competition
  ifelse(VID == n, 0, x)
```

```r
}, sub, colnames(sub)) |> as.data.frame() -> cor_id

# rm x
colnames(cor_id) <- stringr::str_remove(colnames(cor_id), "X")

# check
# data.frame(VID[which(VID == "1557A")], sub[which(VID == "1557A"), 1],
#            cor_id[which(VID == "1557A"), 1])

# merge
dat_bids_id <- data.frame(dat_bids, cor_id)

# colnames again ...
colnames(dat_bids_id)[279:ncol(dat_bids_id)] <- stringr::str_remove(colnames(dat_bids_id)[279:ncol(dat_

# write
# saveRDS(dat_bids_id, "./../../Data/Bid Tab RDS/Bids_id_df.RDS")
```

## 4.1 Split Train and Test

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

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

# write
# saveRDS(dat_bids_id_split, "./../../Data/Bid Tab RDS/Bids_id_df_split.RDS")
```