04 Exploratory Analysis

Fabian Blasch

05/30/2022

1 Load Data

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

# packages
get.package(c("bizdays", "lubridate"))
```

```
## Lade nötiges Paket: bizdays
```

```
## Warning: Paket 'bizdays' wurde unter R Version 4.1.3 erstellt
```

```
##
## Attache Paket: 'bizdays'
```

```
## Das folgende Objekt ist maskiert 'package:stats':
##
##     offset
```

```
## Lade nötiges Paket: lubridate
```

```
## Warning: Paket 'lubridate' wurde unter R Version 4.1.3 erstellt
```

```
##
## Attache Paket: 'lubridate'
```

```
## Die folgenden Objekte sind maskiert von 'package:base':
##
##     date, intersect, setdiff, union
```

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

2 Required Data and Missingness

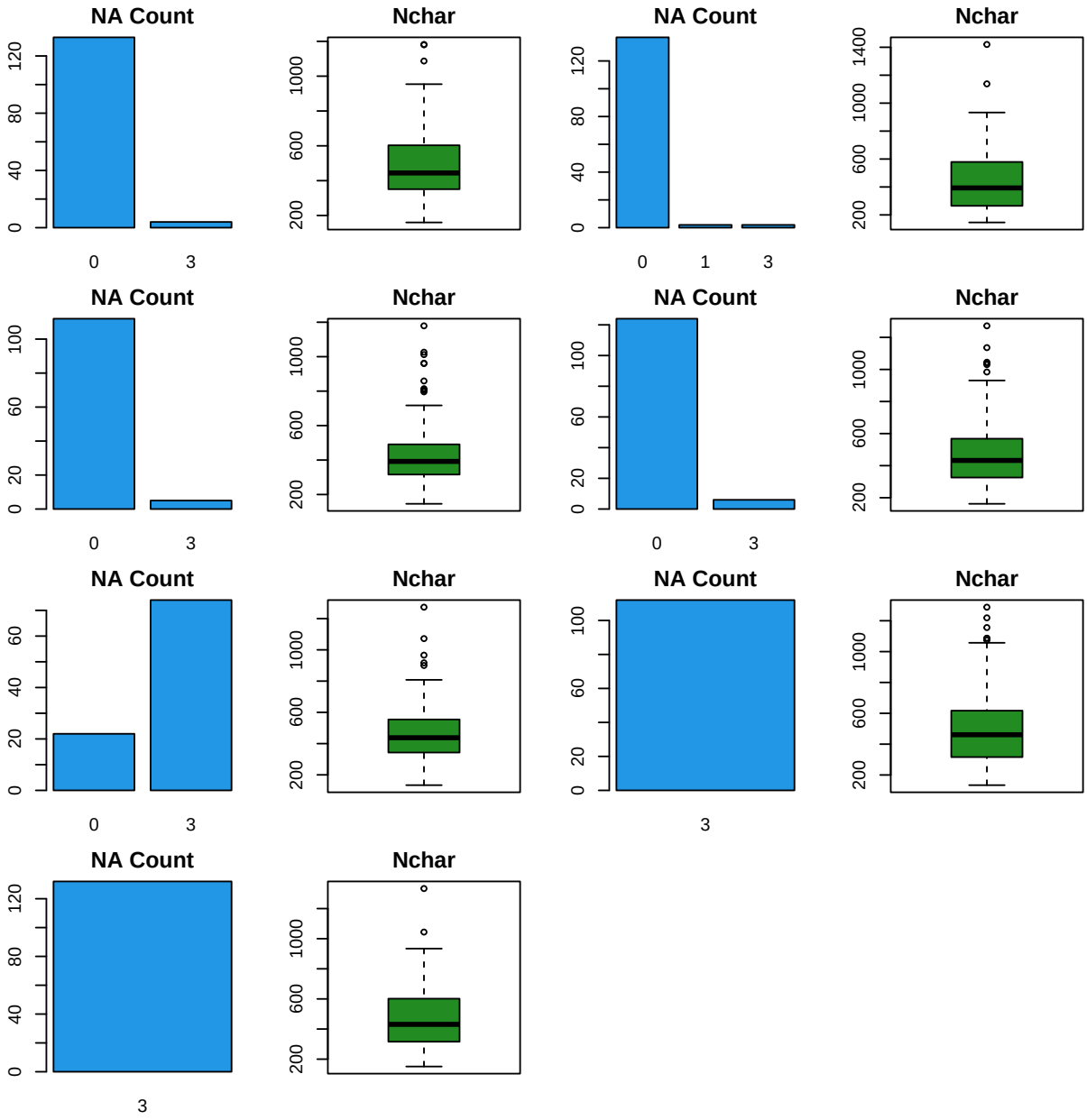
Firstly, we want to find the number of auctions that feature all characteristics required for the estimation procedure outlined in Krasnokutskaya 2012, i.e., letting date, completion time, location, tasks involved (description), identity of all bidders, their bids and an engineer's estimate.

```
# number of auctions available
sapply(auctions, length) |> sum()
```

```
## [1] 865
```

```
# align plots
par(mfrow = c(4, 4), mar = c(2, 2, 2, 2) + 0.1)

# over different years
```

```r
invis.lapply(auctions, \(x){

  # over project ID
  lapply(x, \(y){

    # check
    NAs <- y$Text[c("Letting Date", "Contract Time", "Counties")] |> is.na()

    # no NAs
    na_check <- all(!NAs)

    # Description length
    Dlen <- y$Text["Contract Description"] |> nchar()

    # required min char count
    dlen_check <- Dlen > 200

    # Check for both conditions
    check <- na_check & dlen_check

    # return
    list("NAs" = NAs, "Description" = Dlen,
         "NAcheck" = na_check, "Dlcheck" = dlen_check,
         "Check" = check)

  }) -> tmp

  # display missigness in the three variables
  barplot(table(do.call(cbind, lapply(tmp, "[[", 1)) |> colSums()), xlab = "",
       ylab = "NAs", pch = 19, col = 4, main = "NA Count")

  # display nchar
  boxplot(sapply(tmp, "[[", 2), main = "Nchar", xlab = "", col = "forestgreen")

  # return
  return(tmp)

}) -> filter_dat
```



The plots are aligned by year from left to right and from top to bottom. We observe that the NA count for the variables of interest is 100% for 2021 and 2020. In 2019 some of th auctions seem to carry the required information but most do not. Accordingly, the subset of the data that we will pursue our estimation with will be limited to auctions from 2015 until 2019. Further, we observe that the character count of the description is quite consistent over time, with the median being very close to 400 characters for all years observed. In order to remove descriptions that are not particularly informative observations with a character count below 200 will be removed.

3 Subset of Auctions for Estimation

```
# fetch index for subsetting

# over years
```

```
lapply(filter_dat, \(x){

  # over project ID
  sapply(x, "[[", "Check")

}) -> ind_check

# subset
Map(\(au, ind) au[ind], auctions, ind_check) -> auctions_checked

# number of remaining auctions
sapply(auctions_checked, length) |> sum()
```

```
## [1] 478
```

```
# remove empty lists
auctions_checked[c("2020", "2021")] <- NULL
```

4 Further Data Cleaning

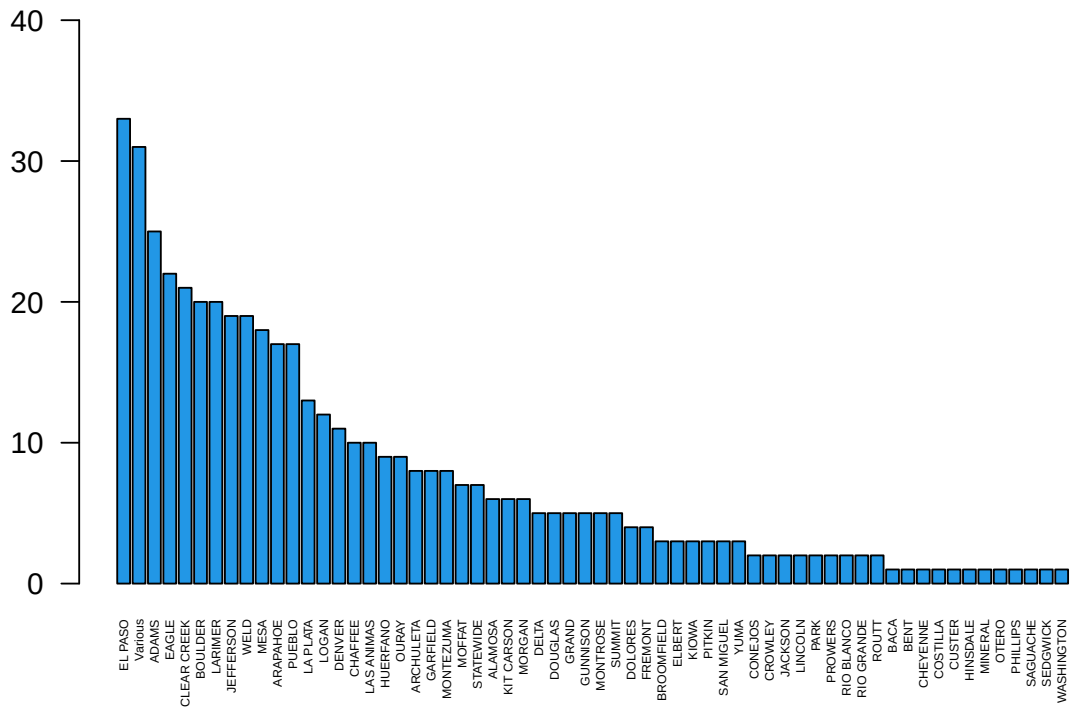
4.1 Counties

The county data that was scraped via *pdftools* still suffers to some textual impurities. Further, counties are split into sub-regions, accordingly to reduce the levels of the factor from 133 to 62, the county specific regions are removed.

```
# pull counties from all auctions
counties <- pull_varT(auctions_checked, "Counties")

# factor levels
counties |> table() |> length()
```

```
## [1] 133
```

```
# remove in county regional separation
counties_alt <- strsplit(counties, ",", fixed = TRUE) |> sapply(FUN = "[[", 1)

# result
barplot(sort(table(counties_alt), decreasing = TRUE), las = 2, cex.names = 0.4,
        ylim = c(0, 40), col = 4)
```




```
# apply transformation to all auctions in the list
auctions_checked <- alt_varT(auctions_checked, "Counties",
                             Fun = \(x) strsplit(x, ",", fixed = TRUE) |>
                             sapply(FUN = "[[", 1))
```

4.2 Letting Date

```
# pull
Ldate <- pull_varT(auctions_checked, "Letting Date")

# date is in a weird format
head(Ldate)
```

```
## 2015.C19942.Letting Date 2015.C20362.Letting Date 2015.C19916.Letting Date
##         "March 12, 2015"         "April 16, 2015"         "April 30, 2015"
## 2015.C19384.Letting Date 2015.C20263.Letting Date 2015.C19918.Letting Date
##      "February 05, 2015"      "December 10, 2015"         "March 05, 2015"
```

```
# transform using lubridate
auctions_checked <- alt_varT(auctions_checked, "Letting Date",
                             Fun = \(x) lubridate::mdy(x) |> as.character())
```

4.3 Contract Time

Unfortunately, contract time is not consistently defined among auctions. Most of the time the amount of working days are provided until the project must be completed. On the website of the Colorado Department

for Transportation it is stated that between the letting date and the beginning of construction, 60 days pass. Accordingly, for all cases where the finishing date is given the amount of working days until project completion are calculated as finishing date - letting date + 60. Where the difference between the two dates only business days will be considered.

```
# pull
Ctime <- pull_varT(auctions_checked, "Contract Time")

# glimpse
head(Ctime)
```

```
## 2015.C19942.Contract Time 2015.C20362.Contract Time 2015.C19916.Contract Time
##                      "90"                      "20"                     "123"
## 2015.C19384.Contract Time 2015.C20263.Contract Time 2015.C19918.Contract Time
##                "11/01/15"                     "170"                     "110"
```

```
# create Calender for Colorado (ran once)
# holidays <- tabulizer::extract_areas("./../../Data/Misc Data/Colorado_Holidays.pdf") |>
#             as.data.frame()

# fix two row entries (ran once)
# sapply(holidays, \(x){
#
#   # paste
#   x[c(9, 11, 13)] <- mapply(\(y, z) paste(y, z), x[c(9, 11, 13)], x[c(9, 11, 13) + 1])
#
#   # assign
#   x <- x[-c(10, 12, 14)]
#
#   # paste year
#   y <- paste0(x[-1], ", ", x[1])
#
#   # remove *
#   y <- stringr::str_remove_all(y, "\\*")
#
#   # convert and return
#   lubridate::mdy(y) |> as.character()
#
# }) -> colorado_holidays

# save (ran once)
# saveRDS(colorado_holidays, "./../../Data/Misc Data/Colorado_Holidays.RDS")

# read RDS
holidays <- readRDS("./../../Data/Misc Data/Colorado_Holidays.RDS")

# manually add 2020 (no file available)
holidays <- cbind(holidays,
      c("2020-01-01", "2020-01-20", "2020-02-17", "2020-04-25", "2020-07-03",
        "2020-09-07", "2020-11-11", "2020-11-26", "2019-12-25", "2019-10-05"))

# to date
holidates <- as.Date(c(holidays))

# create calender
```

```
cal <- bizdays::create.calendar("Colorado",
                                holidays = holidates,
                                weekdays = c("saturday", "sunday"))

# test
bizdays::bizdays("2015-01-01", "2016-01-01", cal) # works great!
```

```
## [1] 251
```

```
# change in place
  # over years
  for(i in seq_along(auctions_checked)){

    # over contract ID
    for(j in seq_along(auctions_checked[[i]])){

      # var
      inp <- auctions_checked[[i]][[j]][["Text"]]["Contract Time"]
      tmp <- auctions_checked[[i]][[j]][["Text"]]["Letting Date"]

      # if string is date not number than
      if(stringr::str_detect(inp, "/")){

        # add 60 days to letting date
        start <- as.Date(tmp) + 60

        # convert finishing date
        stop <- lubridate::mdy(inp)

        # assign
        auctions_checked[[i]][[j]][["Text"]]["Contract Time"] <- bizdays::bizdays(start,
                                                                                  stop,
                                                                                  cal)
      }

    }

  }
```

4.4 Contract Description

In order to cluster the contracts at a later stage, the description text needs to be cleaned further, i.e., remove line breaks and repetitive information.

```
# pull
Cdesc <- pull_varT(auctions_checked, "Contract Description")

# glimpse
tmp <- head(Cdesc, 20)

# function to apply to all
desc_clean <- \(desc){
  stringr::str_remove(desc, "Contract Description:\n") |>
  stringr::str_remove("Contract Description:\n\n") |>
  tolower() |>
```

```
  stringr::str_replace_all("\n", " ") |>
  stringr::str_replace_all("  ", " ") |>
  stringr::str_remove("cdot will only be accepting electronic bids.*") |>
  stringr::str_trim()
}

# apply to all descriptions
auctions_checked <- alt_varT(auctions_checked, "Contract Description",
                             Fun = desc_clean)

# write cleaned file
# saveRDS(auctions_checked, "./../../Data/Bid Tab RDS/Bid_Tabs_Cleaned.RDS")
```