JTR-Data Documentation

Christian 'Becko' Beck

13 April 2017

Introduction

In this short script i will explain how i loaded, transformed, cleaned and amended the given JTR-data. The resulting data set the basis for the following analysis. It is also free to use for everybody else for further analysis-ideas. Please just give credit to the source of this data-set.

Licensing

Data Loading

The source is a flat CSV - a snapshot directly drawn from the database - provided by Ace. It comprises detailed torunament informations as well as participant informations within each row.

```
library(readr)

JTR.jtr <- read_csv("jtr_export.csv", col_names = FALSE,
                    col_types = cols(X2 = col_datetime(format = "%Y-%m-%d %H:%M:%S"), X3 = col_datetime
```

Cleaning and Transformations

In the source all characters are coded in HTML-symbols. So 'ü' becomes ü. For a clean data-set i prefer the original names over those codings that are only relevant for presentation. So i reverse this coding with the help of the XML-package.

```
##############################################################################################
## SOURCE: http://stackoverflow.com/questions/5060076/convert-html-character-entity-encoding-in-r
# added NA check
# load XML package
library(XML)

# Convenience function to convert html codes
html2txt <- function(str) {
  if (is.na(str)) {
    ''
  } else {
    xpathApply(htmlParse(str, asText=TRUE),
               "//body//text()",
               xmlValue)[[1]]
  }
}
##############################################################################################
```

```
library(dplyr, warn.conflicts = FALSE)

# fix char encoding
JTR.jtr$X6[JTR.jtr$X6 == '-'] <- NA
JTR.jtr <- JTR.jtr %>% mutate(X1  = sapply(X1 , FUN = html2txt),
                              X5  = sapply(X5 , FUN = html2txt),
                              X6  = sapply(X6 , FUN = html2txt),
                              X7  = sapply(X7 , FUN = html2txt),
                              X8  = sapply(X8 , FUN = html2txt),
                              X9  = sapply(X9 , FUN = html2txt),
                              X11 = sapply(X11, FUN = html2txt),
                              X12 = sapply(X12, FUN = html2txt),
                              X13 = sapply(X13, FUN = html2txt))
# clean PLZ
JTR.jtr <- JTR.jtr %>% mutate(X6 = substr(X6,1,5))
```

Postal codes are somewhat messy in the dataset, since they are entered by hand. Sometimes they are appended by the city name or sometimes they are not given, but instead '-' is entered. I fix those occurences in the aboce snippet.

Finally the whole dataset gets transformed into one densly coded (with the help of factors) data frame. Note that the factor-encoding for city and country for torunaments and participants are based on all country and city entries in both cases. This increases ease of matching between both conditions in later analysis.

```
# code tournament and team together
Country <- union(JTR.jtr$X5, JTR.jtr$X12)
City    <- union(JTR.jtr$X7, JTR.jtr$X13)
JTR.jtr <- JTR.jtr %>% transmute(TournamentName = as.factor(X1),
                                 TournamentStart = X2,
                                 TournamentEnd = X3,
                                 maxParticipants = X4,
                                 TournamentCountry = factor(X5, levels = Country),
                                 TournamentPostalCode = as.factor(X6),
                                 TournamentCity = factor(X7, levels = City),
                                 TournamentStreet = as.factor(X8),
                                 TournamentPlace = as.factor(X9),
                                 Rank = X10,
                                 TeamName = as.factor(X11),
                                 TeamCountry = factor(X12, levels = Country),
                                 TeamCity = factor(X13, levels = City))
```

Extract and amend tournament information

Separate tournament informations are useful for several analysis scenarios. So they get extracted from the whole dataset above.

```
JTR.Tournaments <- JTR.jtr %>% group_by(TournamentName, TournamentStart, TournamentEnd, TournamentCount
                               summarise(nParticipants = n()) %>% arrange(TournamentStart, TournamentNam

JTR.Tournaments <- JTR.Tournaments %>% ungroup() %>% mutate(TournamentID = row_number())
```

In the above statements i add a TournamentID as a surrogate key - since i don't have access to the original key from the database. This key will be used later to connect results and teams to a torunament.

In the grouping statement i refrain from only using the name of the tournament as group label (instead i'm

somewhat lazy and use all tournament core information). I suspected that while most tournament names are individual due to counting numbers or unique names, that there might be some who aren't. Let's see if i was right.

There are 217 with 213 unique names!

```
# tournament name collisions?
#length(levels(JTR.Tournaments$TournamentName)) #!!!

TournamentsNameMult <- JTR.Tournaments %>% group_by(TournamentName) %>% summarise(n = n()) %>% filter(n
knitr::kable(JTR.Tournaments %>% filter(TournamentName %in% TournamentsNameMult$TournamentName) %>% arra
```

TournamentName	TournamentStart	TournamentEnd	TournamentCountry	Tournam
Ligaqualifikation Nord-West	2015-06-06 09:30:00	2015-06-06 18:00:00	Deutschland	30167
Ligaqualifikation Nord-West	2016-06-18 10:00:00	2016-06-18 20:00:00	Deutschland	30167
OJL Finals / Freiburg Cup	2014-10-11 09:30:00	2014-10-11 18:00:00	Deutschland	79110
OJL Finals / Freiburg Cup	2015-09-19 08:00:00	2015-09-19 18:00:00	Deutschland	79379
Rotenburger Stadtklopperei -Jugendturnier-	2015-04-19 09:00:00	2015-04-19 19:00:00	Deutschland	36199
Rotenburger Stadtklopperei -Jugendturnier-	2016-02-21 10:00:00	2016-02-21 19:00:00	Deutschland	36199
Rotenburger Stadtklopperei -Jugendturnier-	2017-02-12 10:00:00	2017-02-12 19:00:00	Deutschland	36199

Next item on the agenda: geo-information! We got country, city and road - a complete adress - for most tournaments. So why not get the geo coordinates for those places. For this task the most convenient solution would be to load ggmap and use the geocode function. geocode has two source: the Data Science Toolkit (DSK, http://www.datasciencetoolkit.org/about) and Google. The DSK has good coverage in the US and UK, but otherwise bad to no coverage - with the main protion of tournaments located in germany DSK is not an option. Google has good coverage of almost every place (at least thats what they are aiming for), but i don't want to rely to much on the behemoth that google is. So the next logical choice is Open Street Map (http://www.openstreetmap.org/) - it' open, like really open, free and has a very involved community of contributers. They don't have data on every place, but their coverage our regions is quit excellent. So a good alternative to Google i would say.

```
##############################################################################################################
# get geo information
# BASED ON: https://www.r-bloggers.com/search-and-draw-cities-on-a-map-using-openstreetmap-and-r/
library(RJSONIO)

geoCodeOSM <- function(Street, City, Country) {
  cleanCityName  <- gsub(' ', '%20', City)
  if (!is.na(Street)) {
    cleanStreeName <- gsub(' ', '%20', Street)
    url <- paste(
      "http://nominatim.openstreetmap.org/search?"
      , "limit=9&format=json"
      , "&street="
      , cleanStreeName
      , "&city="
      , cleanCityName
      , "&country="
      , Country
      , sep="")
  } else {
    url <- paste(
      "http://nominatim.openstreetmap.org/search?"
```

```
      , "limit=9&format=json"
      , "&city="
      , cleanCityName
      , "&country="
      , Country
      , sep="")
  }
  # return(url);
  resOSM <- fromJSON(url)
  if (length(resOSM) > 0) {
    return(c(resOSM[[1]]$lon, resOSM[[1]]$lat))
  } else return(rep(NA,2))
}
###############################################################################################

geocodeTournaments <- apply(JTR.Tournaments, 1, FUN = function(trow) { geoCodeOSM(trow[7], trow[6], trou
save(geocodeTournaments, file = "geocodes.RData")

JTR.Tournaments$TournamentLongitude <- as.numeric(geocodeTournaments[1, ])
JTR.Tournaments$TournamentLatitude  <- as.numeric(geocodeTournaments[2, ])
```

Note: OSM is free but has costs for hosting and bandwith. Do not run the above code, wehn you don't need it. And considere donating to the project ;) (btw. while testing the above code i was careless and quickly exeeded their bandwidth limit and was blocked for some time)

Do we have coordinates for every tournament?

No surprise: We don't. There are 22 (10.138249 %) tournaments without geo-information.

```
# missing geo?
#sum(is.na(JTR.Tournaments$TournamentLatitude)) / nrow(JTR.Tournaments)

knitr::kable(JTR.Tournaments %>% filter(is.na(TournamentLatitude)))
```

TournamentName	TournamentStart	TournamentEnd	TournamentCountry	Tou
2. Oldenburger Kreismeisteschaft	2009-07-11 09:00:00	2009-07-11 20:00:00	Deutschland	261
1. Hessische Meisterschaft	2009-07-18 09:00:00	2009-07-19 23:59:00	Deutschland	
5. Saarländische Meisterschaft	2009-08-29 11:00:00	2009-08-30 16:00:00	Deutschland	664
12. Deutsche Meisterschaft	2009-09-12 11:00:00	2009-09-13 16:00:00	Deutschland	100
5. Moshen im Park	2009-10-11 10:00:00	2009-10-11 18:00:00	Deutschland	238
2. Freundschaftliche Winterspiele	2009-12-05 10:00:00	2009-12-06 16:00:00	Deutschland	213
2. Hessische Meisterschaft	2010-07-24 09:00:00	2010-07-24 17:00:00	Deutschland	
13. Deutsche Meisterschaft	2010-09-10 11:00:00	2010-09-12 14:00:00	Deutschland	100
6. Moshen im Park	2010-10-09 10:00:00	2010-10-09 18:00:00	Deutschland	238
5. Berliner Juggerpokal	2011-05-07 10:00:00	2011-05-08 18:00:00	Deutschland	
3. Hessische Meisterschaft	2011-07-02 09:00:00	2011-07-02 18:00:00	Deutschland	
2. Berliner Meisterschaft	2011-08-27 11:00:00	2011-08-28 16:00:00	Deutschland	100
7. Moshen im Park	2011-10-02 10:00:00	2011-10-02 18:00:00	Deutschland	238
1. Freundschaftliches Jugger Turnier zu Hannover	2012-09-01 10:00:00	2012-09-02 18:00:00	Deutschland	301
2. Hallesche Stadtbalgerei	2013-05-31 16:00:00	2013-06-02 19:00:00	Deutschland	061
4. NRW Turnier	2014-04-05 10:00:00	2014-04-05 18:00:00	Deutschland	472
Holt euch die Banane! 2015	2015-06-20 09:00:00	2015-06-20 17:00:00	Deutschland	922
1. Rheinland-Pfälzische Meisterschaft	2015-08-01 09:00:00	2015-08-01 19:00:00	Deutschland	
1. Hallenturnier in Oldenburg	2016-03-19 10:00:00	2016-03-20 16:00:00	Deutschland	
Holt euch die Banane! 2016	2016-05-28 08:00:00	2016-05-28 19:00:00	Deutschland	922

TournamentName	TournamentStart	TournamentEnd	TournamentCountry	Tou
2. Rheinland-Pfälzische Meisterschaft	2016-06-04 09:00:00	2016-06-05 15:00:00	Deutschland	
1. Dorfturnier Röttenbach	2016-09-17 09:00:00	2016-09-17 17:00:00	Deutschland	913

Well, obviously missing values are caused by faulty entry of adress information. There is virtually no way to fix this, so we just have to live with it - no big problem anyways.

Extract and amend Team information

The next logical step is to create a similar table for each team.

```
##############################################################################################

JTR.Teams <- JTR.jtr %>% group_by(TeamName, TeamCountry, TeamCity) %>% summarise(nParticipations = n())
JTR.Teams <- JTR.Teams %>% ungroup() %>% mutate(TeamID = row_number())
```

Again i opt for using the full team information in the JTR source to identify a team - just to be on the safe side. However no team-name collision is present in the current data-set, so just by name would be as accurate. For some use cases (e.g. a team relocating) the chosen approach might even be a problem, but there is no example for such a process in the data.

Note that i again add a surrogate Team ID for later analysis purpose.

And again geocoding is possible.

```
geocodeTeams <- apply(JTR.Teams, 1, FUN = function(trow) { geoCodeOSM(Street = NA, trow[3], trow[2]) })
save(geocodeTournaments,geocodeTeams, file = "geocodes.RData")

JTR.Teams$TeamLongitude <- as.numeric(geocodeTeams[1, ])
JTR.Teams$TeamLatitude  <- as.numeric(geocodeTeams[2, ])
# rm(geocodeTeams)
```

A Team has additional information of country and city. Not very specific, but still as an approximation good enough. This information might not be as useful or accurate as the geocoding for the tournament location, but it still has some uses.

Do we have coordinates for every team?

No surprise: We don't. There are 34 (7.172996 %) teams without valid geo-information.

```
# missing geo?
#sum(is.na(JTR.Teams$TeamLatitude)) / nrow(JTR.Teams)

knitr::kable(JTR.Teams %>% filter(is.na(TeamLatitude)))
```

TeamName	TeamCountry	TeamCity	nParticipations	TeamID
AH Mixedup	Deutschland	Mixteam	1	10
Assis unter Palmen	Deutschland	Mainz - Darmstadt	1	21
Auenland Pudel	Deutschland	Heidelberg-Darmstadt	1	23
Blaubären	Deutschland	Hannover/Paderborn	1	44
Chimera Kraken	Deutschland	Mixteam	1	58
Die Kreischwürste	Deutschland	3 Citys	1	85
Die Kurzen "pure strenght"	Deutschland	Geilenhausen	3	87
Five Finger Death Pompfer	Deutschland	Trusetal/Schmalkalden	2	122
Flying Wolfmen	Deutschland	Bonn/Bielefeld	1	133

TeamName	TeamCountry	TeamCity	nParticipations	TeamID
Freilos	Deutschland	Mixteam	1	136
Gossenmob	Deutschland	Mixteam	1	154
JNV Westfalia Thüle 69 - Alte Herren II	Deutschland	BÄÄMudadreieck	1	189
Jugger Pöbelblock	Deutschland	Hamburg/Berlin	1	215
Julia und die Räuber	Deutschland	Mixteam	1	240
KlosterKätzchen	Deutschland	Mixteam	1	255
Lahnpinkchen	Deutschland	Mixteam	1	261
MadBears	Deutschland	Mixteam	1	275
Mixed	Deutschland	mixed	1	288
Mix-Team	Deutschland	Mixteam	1	292
Nauteraner	Deutschland	Mixteam	2	306
OlRoburg	Deutschland	Mixteam	2	315
Oranjes	Nederland	Randstad	1	317
Pinke Patrolleure	Deutschland	Mainz - Darmstadt	2	329
Resterampe	Deutschland	Mixteam	1	356
Ruby Rabauken	Deutschland	Mainz - Darmstadt	1	370
S.G.A.D	Deutschland	Mixteam	1	389
S.H.I.E.L.D.	Deutschland	Bundesweit	1	390
Silver Horde	Deutschland	3 Citys	6	392
SpaceShuttle17	United States of America	Cape Canaveral	1	396
Swimming Pool	Deutschland	Mixteam	2	409
Team im MIxer	Deutschland	Mixteam	1	414
The Avengers	United States of America	NY	3	419
Waffelnauts	Deutschland	Mixteam	1	455
ZonenSpaten	Deutschland	mixed	1	473

Well, obviously missing values are caused by faulty entry of adress information and this time this seems to be wanted - some teams are not affiliated with any city, but are a mixed from several cities.

Extract and amend Results

Lastly i'd like to have a seperate results table, just like you would write into a database. Given the prior created IDs this table can be created without any hassle.

```
JTR.jtr <- JTR.jtr %>%  left_join(JTR.Tournaments) %>%
                        left_join(JTR.Teams)
```

```
## Joining, by = c("TournamentName", "TournamentStart", "TournamentEnd", "maxParticipants", "TournamentC
```

```
## Joining, by = c("TeamName", "TeamCountry", "TeamCity")
```

```
JTR.Results <- JTR.jtr %>% transmute(TournamentID, TeamID, Rank)
```

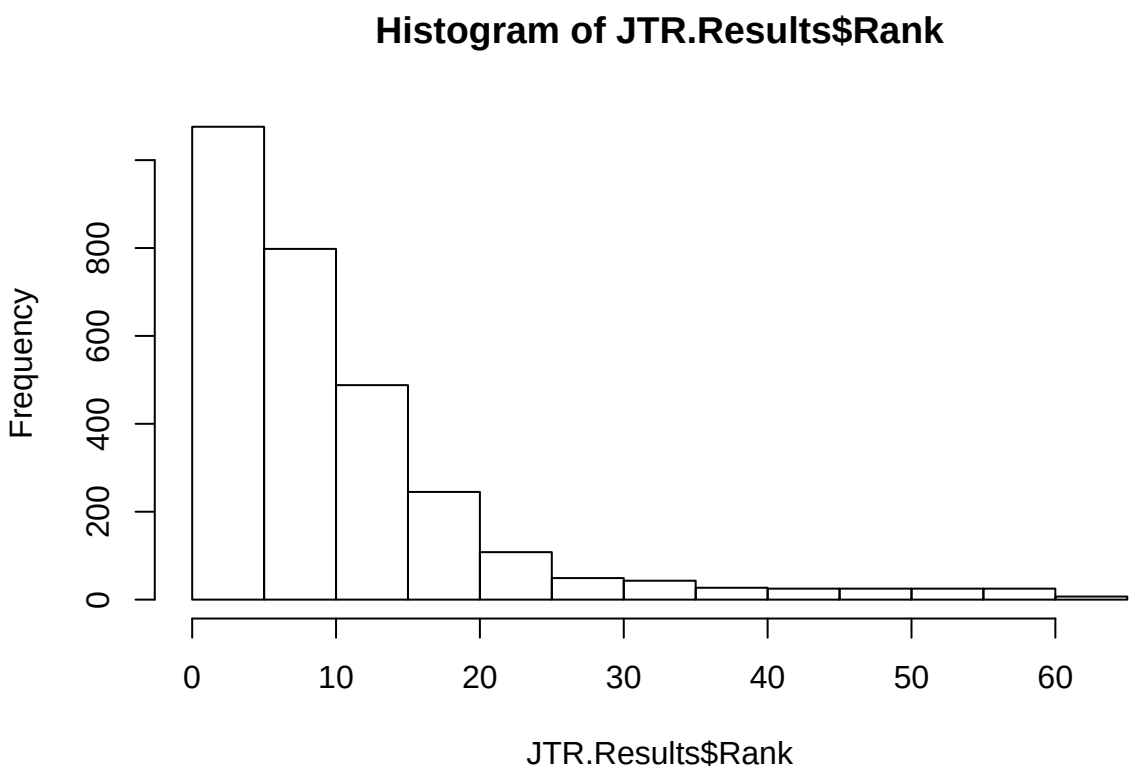
Just a quick look on the new dataset:

```
summary(JTR.Results)
```

```
##   TournamentID       TeamID           Rank
##  Min.   :  1.0   Min.   :  1.0   Min.   : 1.0
##  1st Qu.: 63.0   1st Qu.:129.0   1st Qu.: 4.0
##  Median :113.0   Median :256.0   Median : 8.0
##  Mean   :112.7   Mean   :246.1   Mean   :10.8
##  3rd Qu.:165.0   3rd Qu.:359.0   3rd Qu.:14.0
```

```
##  Max.   :217.0   Max.   :474.0   Max.   :64.0
```

```
hist(JTR.Results$Rank)
```



No outliers. Nice! And - as expected - lots of torunaments with a low number of participants and few with very many teams (64 Teams, DM2014). No surprises here.

```
knitr::kable(JTR.Results %>% group_by(TournamentID) %>% summarise(n = n(), max = max(Rank), ranks = len
```

TournamentID	n	max	ranks
9	8	5	5
14	21	20	19
18	3	2	2
87	15	14	14
107	36	35	20
129	8	5	5
167	9	6	6
171	16	15	15

There is a small number of torunaments which at a first glance are somewhat problematic. Those tournaments have all less result ranks than participating teams. But those are all due to ties in the result set (sometimes even intended by the tournament organiser). Just keep this in mind when analysing torunament results - either find some logic for those cases or remove those few tournaments.

Finishing

All four sets - jtr, tournaments, teams and results - can be used for any kind of analysis. In the following snippet i store them in as RData and as flat-files, so that anyone can use those data-sets for any analysis purpose.

```
devtools::use_data(JTR.jtr, JTR.Tournaments, JTR.Teams, JTR.Results, overwrite = TRUE)
```

```
## Saving JTR.jtr, JTR.Tournaments, JTR.Teams, JTR.Results as JTR.jtr.rda, JTR.Tournaments.rda, JTR.Tea
```

```
save(JTR.jtr, JTR.Tournaments, JTR.Teams, JTR.Results, file = "jtr.RData")
```