How to use the quantqual package for web-scraping, wrangling, plotting and analyzing data.

2019-01-08

On the following pages you will learn some basics on how to use the `quantqual` package for scraping, wrangling, plotting and analyzing data. Let's start by installing and loading the `quantqual` package.

```
devtools::install_github("AndreasFischer1985/quantqual")
library(quantqual)
```

Web-Scraping

In an empirical study on "How to identify hot topics in psychology using topic modeling" Bittermann & Fischer (2018) recently identified the so-called "refugee crisis" and related contents (e.g., Cross-Cultural Differences, Human Migration, Cross-Cultural Communication, Cultural Sensitivity, Cross-Cultural Treatment, Multiculturalism, Expatriates, Transcultural Psychiatry, International Organizations, Cross-Cultural Counseling, Globalization, Multicultural Education, Foreign Workers, Acculturation, Racial And Ethnic Differences) as one of the hot topics in today's psychological research.

Thus, let's download a pdf-document with up-to-date data on asylum seekers in Germany (provided by the Federal Office for Migration and Refugees) and use the `extractTable`-function to extract a table with data on the number of asylum seekers from different countries of origin. The table in this pdf-document is surrounded by text, so we have to specify a regular expression to identify the first line of the table (`reg.up`), the first line below the table (`reg.down`), as well as the start (`reg.left`) and end-point (`reg.right`) of the first line. As the table contains information we don't need, we also specify, which pattern to exclude (`reg.fix`).

```
#Get pdf document from an url and save it to the working directory:
url=paste0("http://www.bamf.de/SharedDocs/Anlagen/DE/Publikationen/Flyer/",
"flyer-schluesselzahlen-asyl-halbjahr-2018.pdf?__blob=publicationFile");
fn=paste0(gsub("[/?.:]","",url),".pdf");
doc=getFile(url,fn);

#Extract table:
tab=extractTable(strsplit(doc,"\r\n")[[1]],
reg.up="Staatsangeh.rigkeit",reg.down="Entscheidungen",reg.left="Staatsangeh.rigkeit",
reg.right="$",reg.fix="(^[ ]*[0-9]{1,2}[ ]+|[ ]+[0-9][ ]*$)",correctNotation=T,convert=T)
```

In addition to information on applications for asylum, let's download some data on family reunification of immigrants. This time, we'll extract the tables from the appenix of a multi-page document (with one table per page), which makes extracting the table a lot easier.

```
#Get pdf document from an url and save it to the working directory:
url=paste0("http://www.bamf.de/SharedDocs/Anlagen/EN/Publikationen/EMN/Studien/",
"wp73-emn-familiennachzug-drittstaatsangehoerige-deutschland.pdf?__blob=publicationFile")
doc=getFile(url,paste0(gsub("[/?.:]","",url),".pdf"))
doc=strsplit(doc,"\r\n")[[1]]

#Extract tables (skipping dots):
tab2015=extractTable(
doc[(last(which(grepl("Table 4",doc)),1)+4):(last(which(grepl("Table 5",doc)),1)-3)],reg.fix="[.]")

tab2014=extractTable(
doc[(last(which(grepl("Table 5",doc)),1)+5):(last(which(grepl("Table 6",doc)),1)-3)],reg.fix="[.]")
```

```
tab2013=extractTable(
doc[(last(which(grepl("Table 6",doc)),1)+4):(last(which(grepl("Table 7",doc)),1)-3)],reg.fix="[.]")

tab2012=extractTable(
doc[(last(which(grepl("Table 7",doc)),1)+5):(last(which(grepl("Table 8",doc)),1)-3)],reg.fix="[.]")

tab2011=extractTable(
doc[(last(which(grepl("Table 8",doc)),1)+4):(last(which(grepl("Table 9",doc)),1)-3)],reg.fix="[.]")

tab2010=extractTable(
doc[(last(which(grepl("Table 9",doc)),1)+5):(last(which(grepl("Table 10",doc)),1)-3)],reg.fix="[.]")
```

Wrangling

Now we'll extract an interesting subset from the first table we extracted (the number of first time applications per year for each country of origin) for further analysis, translate rownames from German to English, replace missing values with 0 (which makes plotting a lot more comfortable) and bring some order to the data.

```
dat=tab[c(1:16,19),-c(1,4,6)]
rownames(dat)=c("Afghanistan","Albania","Eritrea","Georgia",
"Iraq","Iran","Kosovo","Macedonia","Nigeria","Pakistan",
"Russ.Fed.","Serbia","Somalia","Syria","Turkey","Unknown","Total")
colnames(dat)=c("2015","2016","2017","1st half of 2018")
dat[is.na(dat)]=0
dat=dat[order(rowSums(dat,na.rm=T),decreasing=T),]
dat
```

```
##               2015   2016   2017 1st half of 2018
## Total       441899 722370 198317            81765
## Syria       158657 266250  48974            21587
## Afghanistan  31382 127012  16423             5138
## Iraq         29784  96116  21930             8259
## Albania      53805  14853      0                0
## Eritrea      10876  18854  10226             3535
## Iran             0  26426   8608             4283
## Kosovo       33427      0      0                0
## Unknown      11721  14659   4067             2109
## Nigeria          0  12709   7811             5734
## Pakistan      8199  14484      0                0
## Serbia       16700      0      0                0
## Russ.Fed.        0  10985   4884                0
## Turkey           0      0   8027             4089
## Somalia          0      0   6836             2912
## Macedonia     9083      0      0                0
## Georgia          0      0      0             2450
```

From the other tables (from tab2010 to tab2015) we'll extract the total numbers of immigrations for family reunification purposes, and combine them in a `data.frame` `dat2`.

```
#Select first and last column of each table:
t15=tab2015[,c(1,dim(tab2015)[2])]
t14=tab2014[,c(1,dim(tab2014)[2])]
t13=tab2013[,c(1,dim(tab2013)[2])]
t12=tab2012[,c(1,dim(tab2012)[2])]
t11=tab2011[,c(1,dim(tab2011)[2])]
```

```
t10=tab2010[,c(1,dim(tab2010)[2])]

#Combine the last columns of all tables to a matrix
dat2=cbind(
"2015"=as.numeric(t15[,2]),
"2014"=as.numeric(t14[match(t15[,1],t14[,1]),2]),
"2013"=as.numeric(t13[match(t15[,1],t13[,1]),2]),
"2012"=as.numeric(t12[match(t15[,1],t12[,1]),2]),
"2011"=as.numeric(t11[match(t15[,1],t11[,1]),2]),
"2010"=as.numeric(t10[match(t15[,1],t10[,1]),2]))
rownames(dat2)=t15[,1]

dat2 #looks like lines 13:15 need a little face-lifting...
```

```
##                           2015  2014  2013  2012  2011  2010
##  Syria                   15956  3025   861    NA    NA    NA
##  Turkey                   7720  7317  6966  7332  8363  8366
##  Russian Federation       4726  4286  4108  3926  3733  3646
##  India                    4605  3992  3542  3634  2970  2613
##  Kosovo                   3808  3766  3337  2835  2770  2875
##  USA                      3098  3075  2942  3090  3254  2849
##  Ukraine                  2693  2642  2140  1937  1772  1569
##  China                    2635  2418  2114  1974  1790  1527
##  Iraq                     1800    NA    NA   757  1034  2555
##  Bosnia and Herzegovina   1775  1425  1183  1019   894   771
##  Japan                    1743  1650  1674  1844  1870  1669
##  Morocco                  1672  1504  1475  1527  1441  1456
##  Serbia (incl former        NA    NA    NA    NA    NA    NA
##                           1617  1417  1391  1455  1282  1228
##  Serbia & Montenegro)       NA    NA    NA    NA    NA    NA
##  Pakistan                 1543  1798  1092   794   860   850
##  Thailand                 1437  1416  1526  1513  1584  1728
##  Brazil                   1432  1064   953  1075  1071  1083
##  Macedonia                1174  1005   891   760   709    NA
##  Tunisia                  1171  1142  1010   945   862   870
##  Vietnam                  1127  1055   933   898   905   983
##  Iran                     1063  1080   924   845   798   748
##  All countries           82440 63677 56046 54816 53495 54036
```

```
dat2=dat2[-c(13,15),];rownames(dat2)[13]="Serbia"
dat2[is.na(dat2)]=0
dat2=dat2[order(rowSums(dat2,na.rm=T),decreasing=T),]
dat2
```

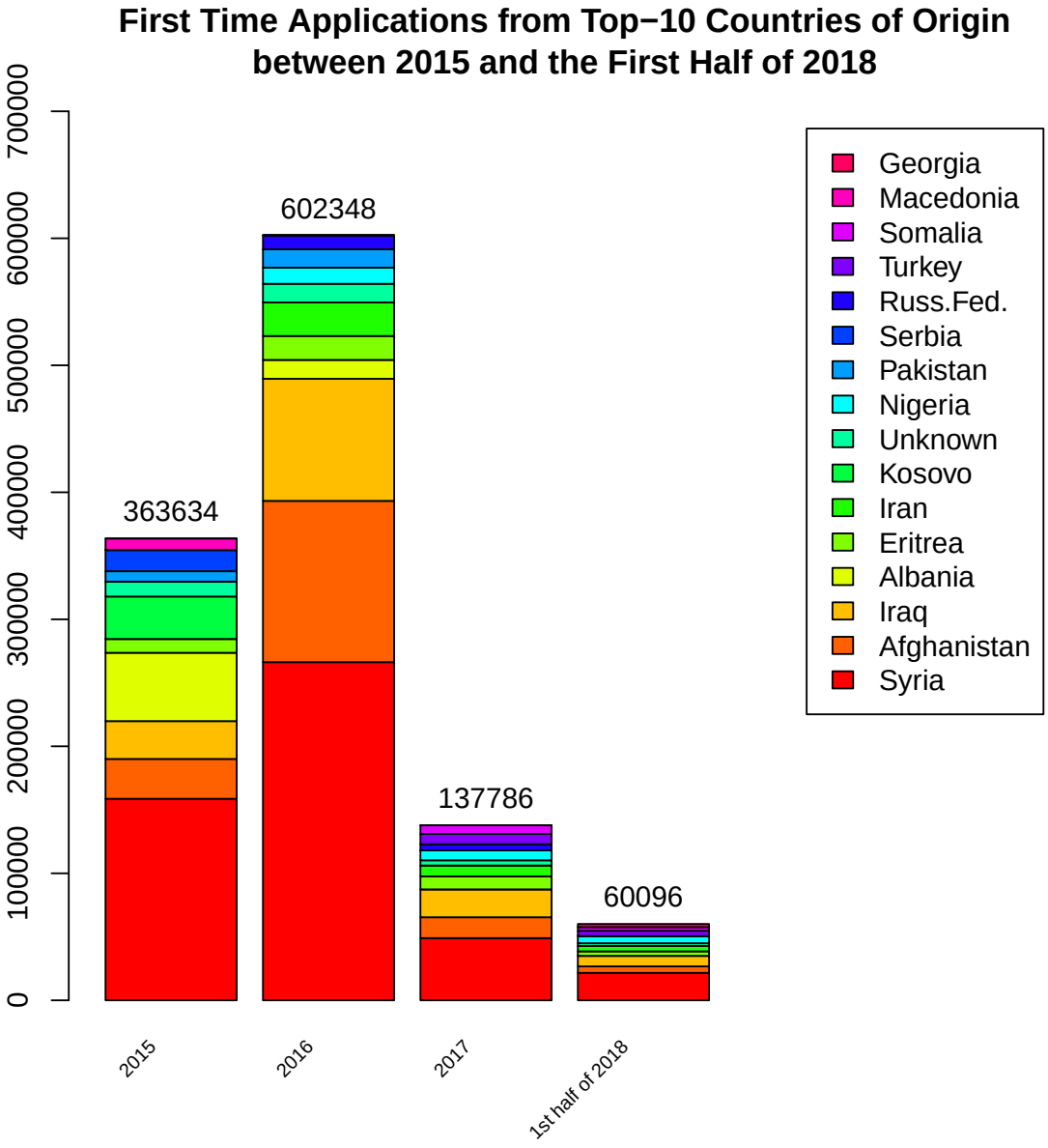
```
##                           2015  2014  2013  2012  2011  2010
##  All countries           82440 63677 56046 54816 53495 54036
##  Turkey                   7720  7317  6966  7332  8363  8366
##  Russian Federation       4726  4286  4108  3926  3733  3646
##  India                    4605  3992  3542  3634  2970  2613
##  Syria                   15956  3025   861     0     0     0
##  Kosovo                   3808  3766  3337  2835  2770  2875
##  USA                      3098  3075  2942  3090  3254  2849
##  Ukraine                  2693  2642  2140  1937  1772  1569
##  China                    2635  2418  2114  1974  1790  1527
##  Japan                    1743  1650  1674  1844  1870  1669
```

```
##  Thailand                1437  1416  1526  1513  1584  1728
##  Morocco                 1672  1504  1475  1527  1441  1456
## Serbia                   1617  1417  1391  1455  1282  1228
##  Bosnia and Herzegovina  1775  1425  1183  1019   894   771
##  Pakistan                1543  1798  1092   794   860   850
##  Brazil                  1432  1064   953  1075  1071  1083
##  Iraq                    1800     0     0   757  1034  2555
##  Tunisia                 1171  1142  1010   945   862   870
##  Vietnam                 1127  1055   933   898   905   983
##  Iran                    1063  1080   924   845   798   748
##  Macedonia               1174  1005   891   760   709     0
```

Plotting

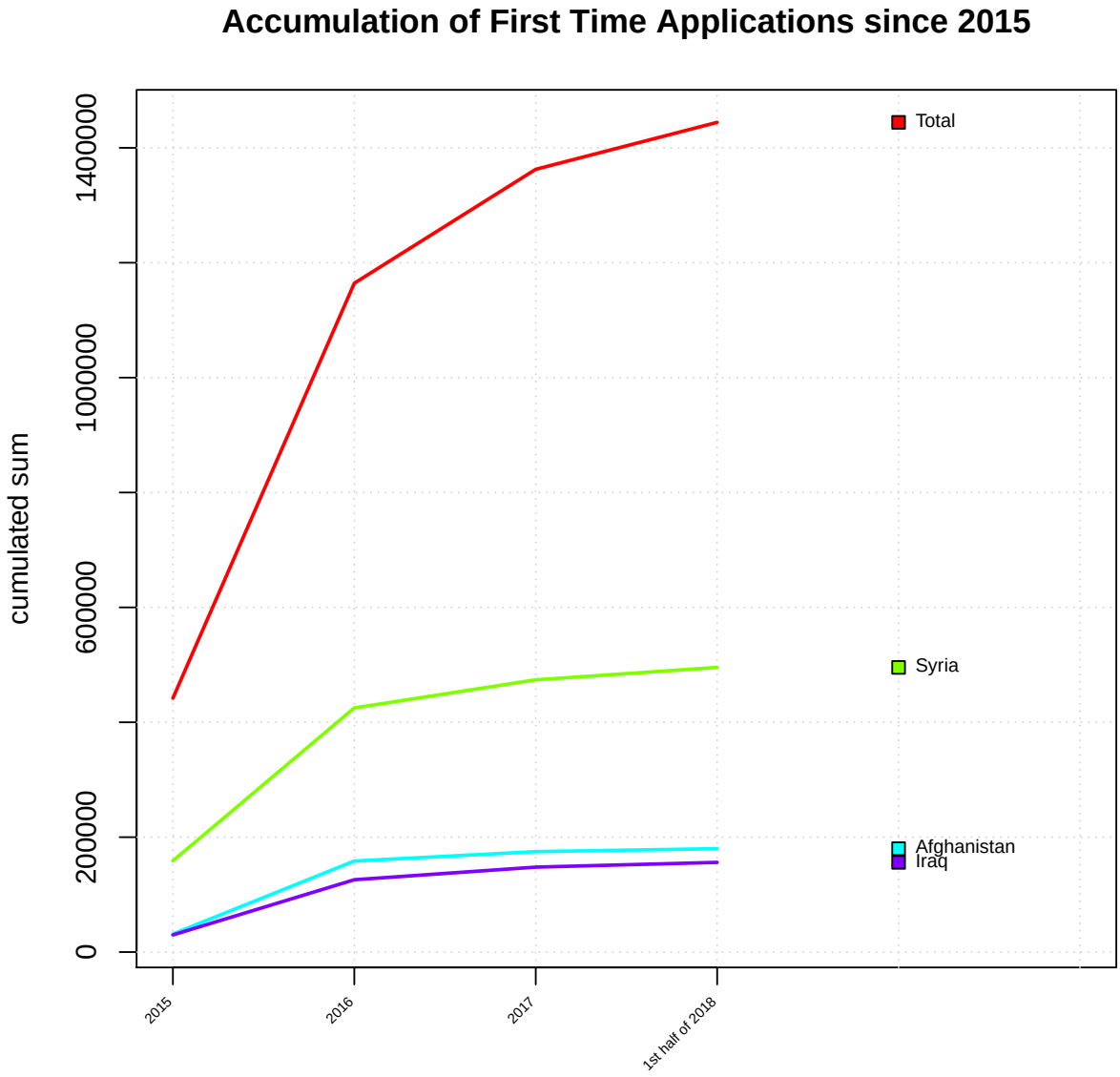
First, let's have a look at the number of first time applications for asylum in Germany from different countries of origin using the `bp`-function.

```
options(scipen=5)
b=bp(dat[-1,],beside=F,
main=paste0("First Time Applications from Top-10 Countries of Origin\n",
"between 2015 and the First Half of 2018"),ylim=c(0,700000))
text(b,colSums(dat[-1,]),colSums(dat[-1,]),pos=3,xpd=T)
```



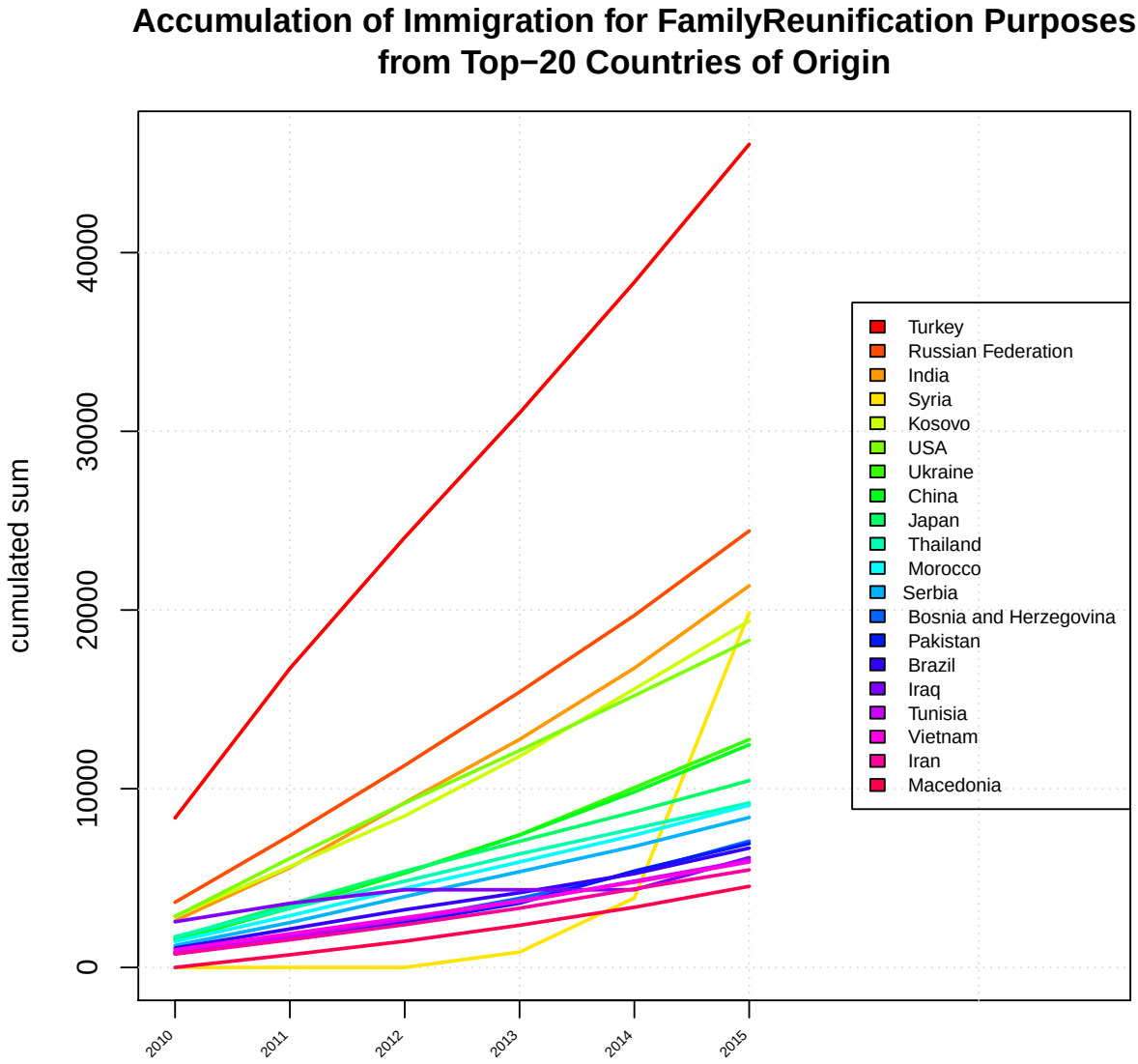
Let's plot the numbers of first time applications for the Top-3 countries of origin over time using the `plotMAT`-function.

```
plotMAT(dat[1:4,],"Accumulation of First Time Applications since 2015",show.legend=F)
```



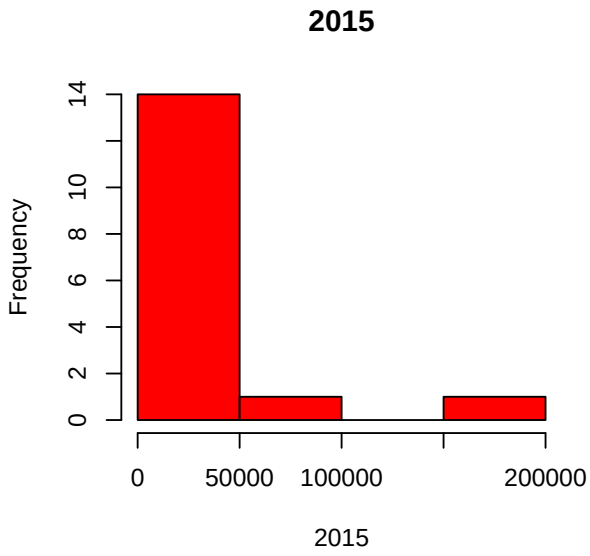
Next, let's have a look at how the immigration for the purpose of family reunification accumulated over the years for different countries of origin.

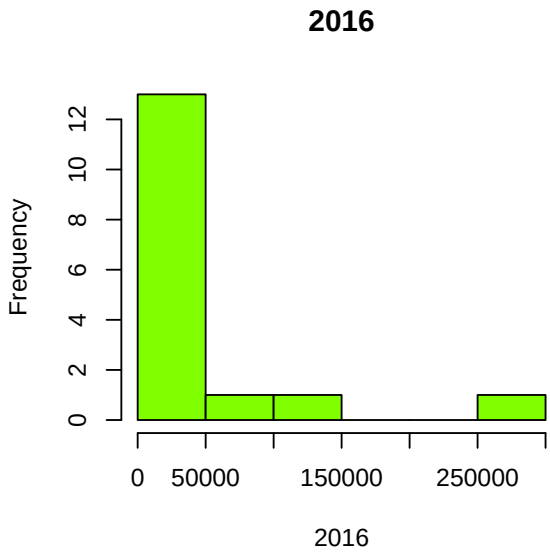
```
plotMAT(dat2[2:21,6:1],paste0("Accumulation of Immigration for Family",
"Reunification Purposes\nfrom Top-20 Countries of Origin"))
```

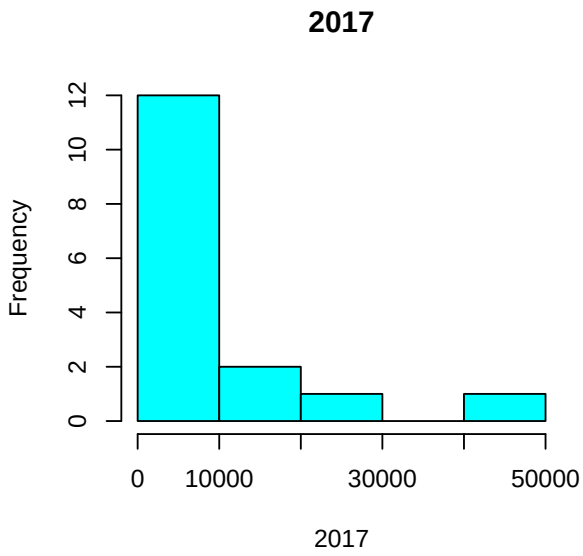


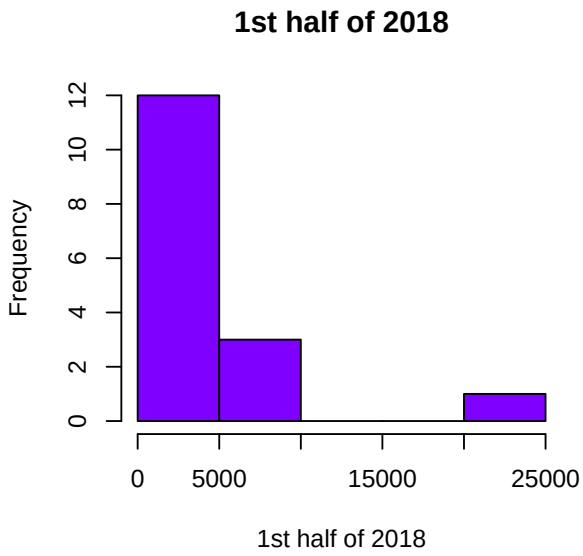
For a quick overview over our data, we can also use `plotDF` to plot a histogram per column.

```
plotDF(dat[-1,])
```

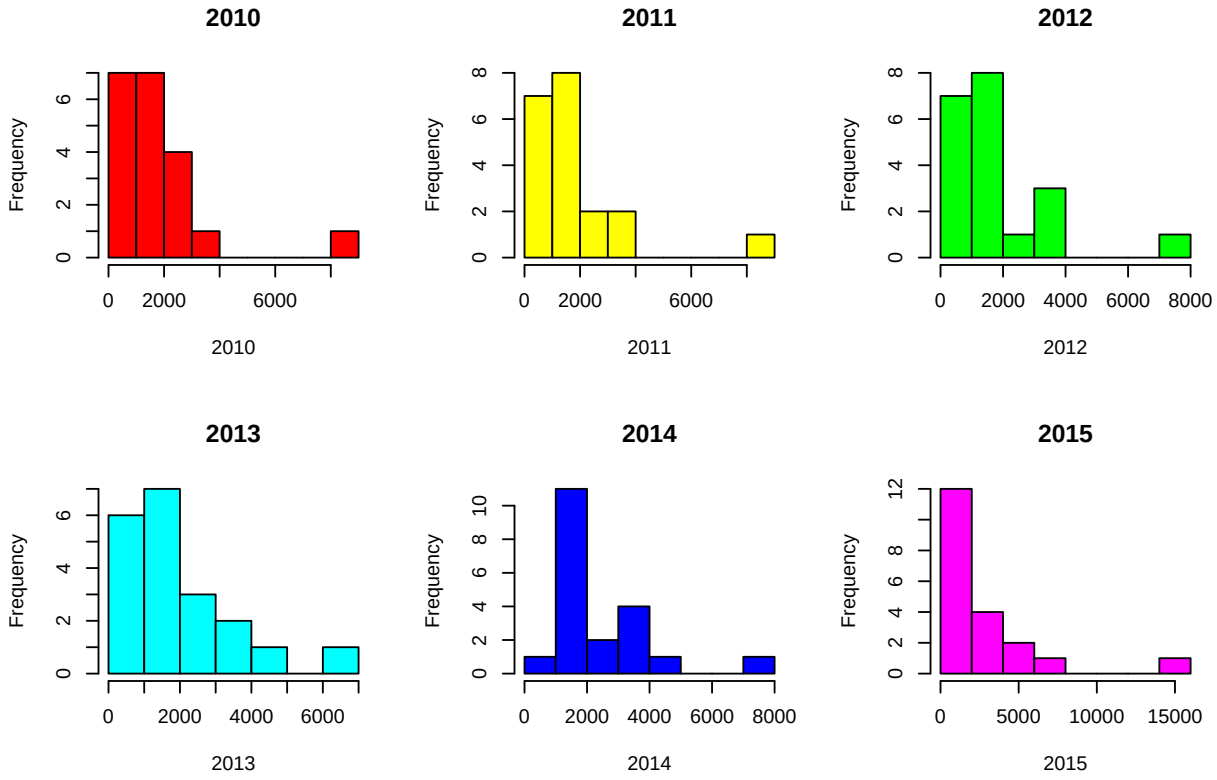






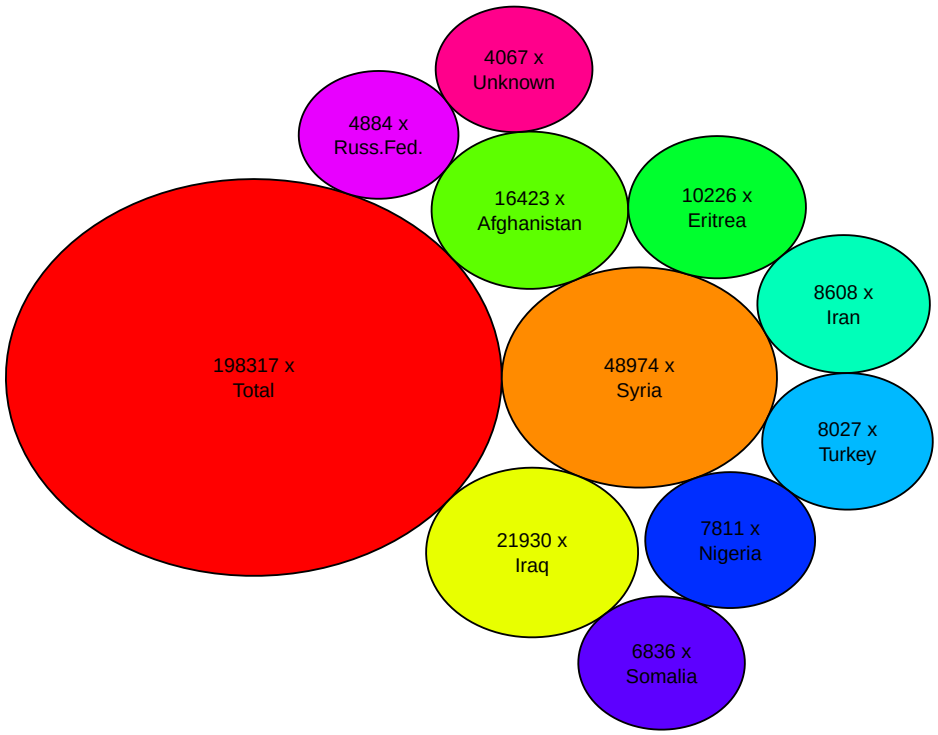



```
plotDF(dat2[2:21,6:1])
```



Now we'll plot the number of applications from the Top-Ten different countries in 2017 using the function `packedBubbleChart`.

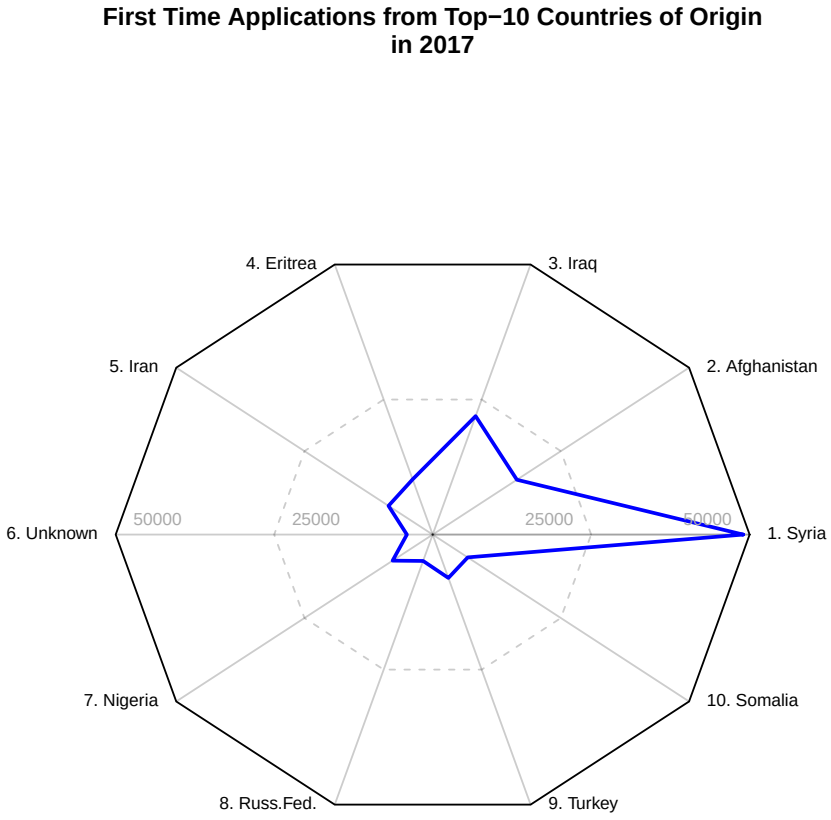
```
packedBubbleChart(dat[dat[,"2017"]>0,"2017"],break.names=T,
main="\n\nFirst Time Applications for Asylum\nfrom Top-10 Countries of Origin in 2017",cex=.7)
```




```
##             x         y   radius
## 1  -5.6418958  0.000000 5.641896
## 2   3.1314470  0.000000 3.131447
## 3   0.6901689 -4.976893 2.411955
## 4   0.6401824  4.755699 2.237265
## 5   4.8993104  4.841505 2.022727
## 6   7.7790265  2.086127 1.962856
## 7   7.8673313 -1.816636 1.940906
## 8   5.1977344 -4.623404 1.932683
## 9   3.6366485 -8.118409 1.895117
## 10 -2.8025568  6.897964 1.817578
## 11  0.2799192  8.760918 1.784124
```

Maybe a `spiderplot` is better suited for comparing the numbers of different countries? Let's try with a single line of code.

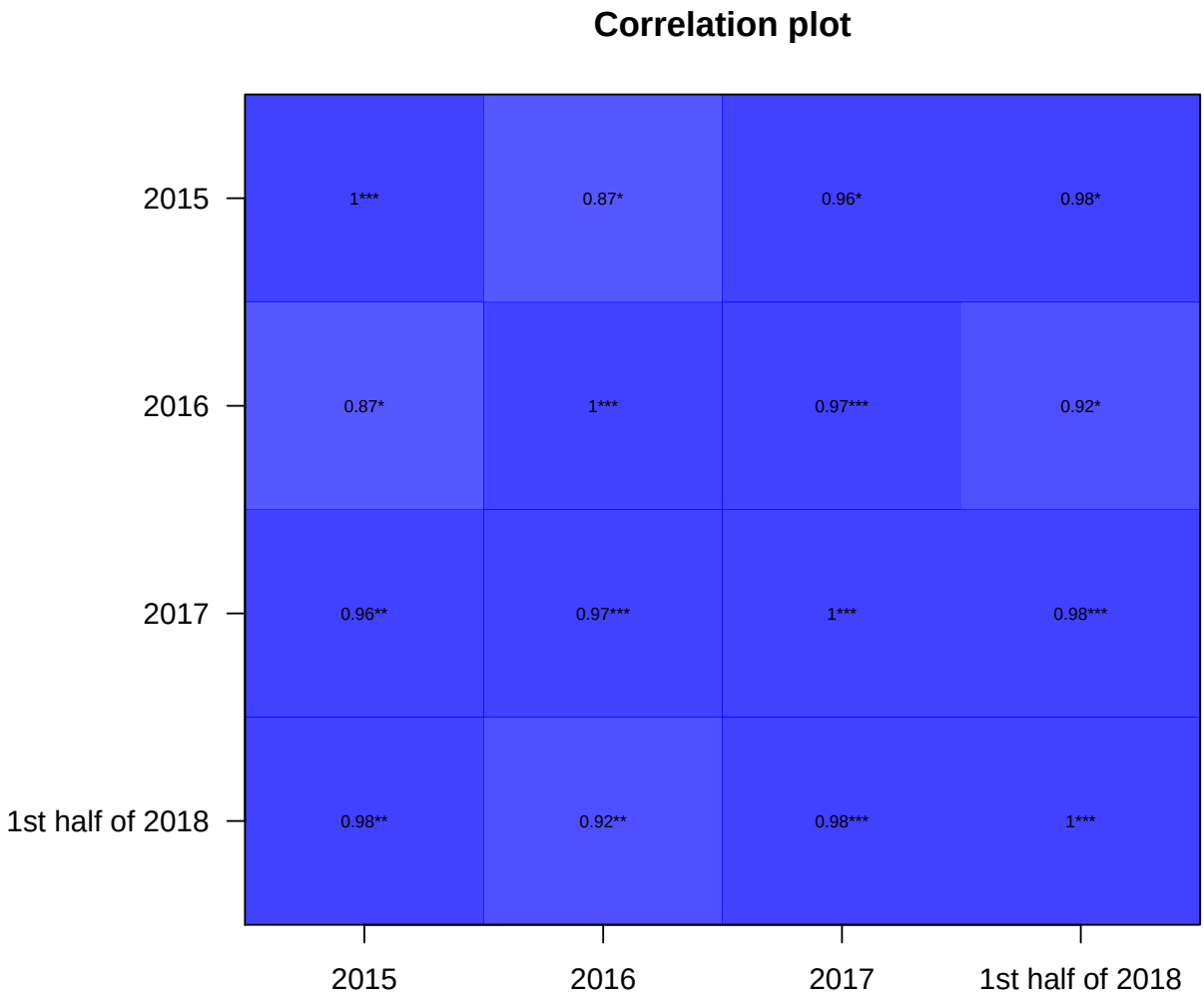
```
spiderplot(dat[dat[,"2017"]>0,"2017"][-1],max=50000,main=
"\n\nFirst Time Applications from Top-10 Countries of Origin\nin 2017")
```



Analyzing

Let's have a look at all the correlations between years (over countries) using the `correlationplot`-function (and skipping observations with missing values).

```
dat[dat==0]=NA;
correlationplot(dat[-1,])
```

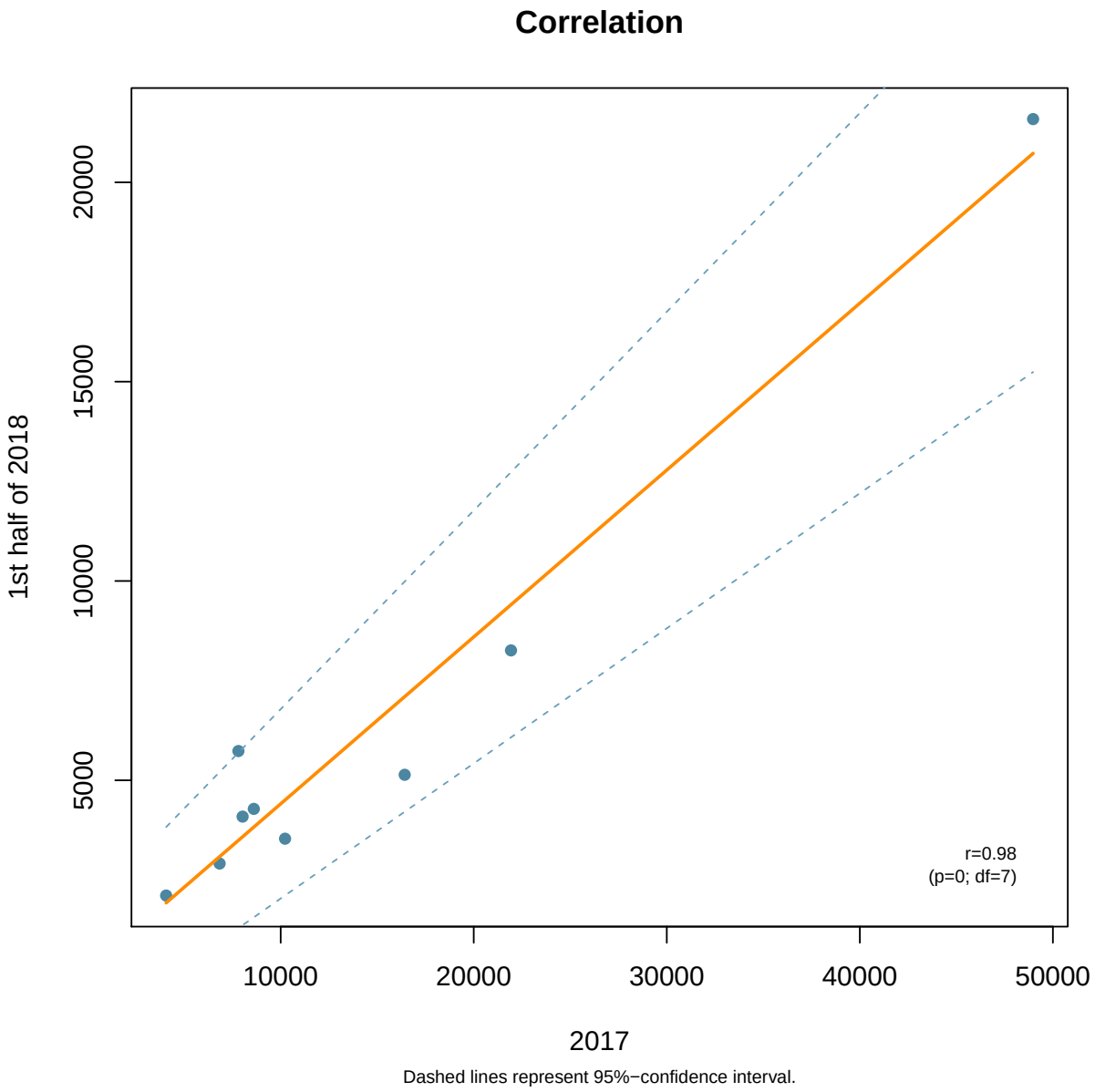



```
## Call:corr.test(x = data)
## Correlation matrix
##                 2015 2016 2017 1st half of 2018
## 2015            1.00 0.87 0.96            0.98
## 2016            0.87 1.00 0.97            0.92
## 2017            0.96 0.97 1.00            0.98
## 1st half of 2018 0.98 0.92 0.98            1.00
## Sample Size
##                 2015 2016 2017 1st half of 2018
## 2015              10    7    5                5
## 2016               7   10    8                7
## 2017               5    8   10                9
## 1st half of 2018    5    7    9               10
## Probability values (Entries above the diagonal are adjusted for multiple tests.)
```

```
##                  2015 2016 2017 1st half of 2018
## 2015             0.00 0.02 0.02             0.01
## 2016             0.01 0.00 0.00             0.01
## 2017             0.01 0.00 0.00             0.00
## 1st half of 2018 0.00 0.00 0.00             0.00
##
##  To see confidence intervals of the correlations, print with the short=FALSE option
```

If we want to examine a single correlation in more detail (let's say the correlation between 2017 and the first half of 2018) we could use the `plotXY`-function (skipping observations with missing values).

```
plotXY(dat[-1,3],dat[-1,4],xlab="2017",ylab="1st half of 2018")
```

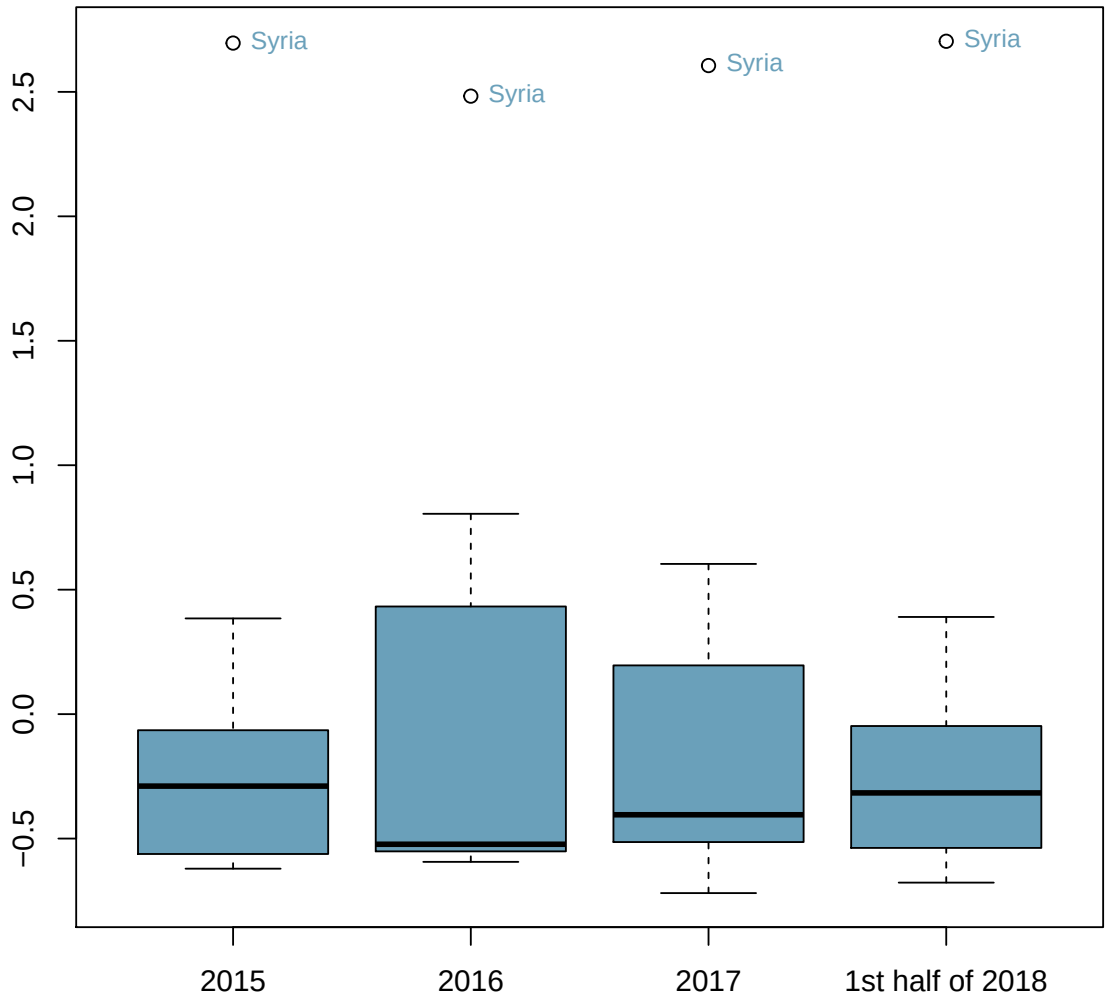



```
##
## Call:
```

```
## lm(formula = data[, 2] ~ data[, 1])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1960.5  -969.0   183.5   505.6  2241.0
##
## Coefficients:
##              Estimate Std. Error t value   Pr(>|t|)
## (Intercept) 222.85493  666.95953   0.334      0.748
## data[, 1]     0.41866    0.03374  12.408 0.00000508 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1330 on 7 degrees of freedom
## Multiple R-squared:  0.9565, Adjusted R-squared:  0.9503
## F-statistic:   154 on 1 and 7 DF,  p-value: 0.00000508
```

Based on the plots presented above we can say that the numbers of applications from Syrian refugees stand out quite a bit (compared to the rest of the data). Let's check this in a more formal manner by conducting an outlier analysis. The function `outlier.analysis` gives us boxplots of scaled variables (to detect univariate outliers) and - as an attribute of the boxplot-object - the mahalanobis distance for each observation (to detect multivariate outliers).

```
outlier.analysis(dat[-1,])
```




```
## $stats
##             [,1]       [,2]       [,3]        [,4]
## [1,] -0.62091078 -0.5935752 -0.7190310 -0.67693082
## [2,] -0.56189379 -0.5514041 -0.5140188 -0.53757394
## [3,] -0.28927365 -0.5228461 -0.4043309 -0.31647722
## [4,] -0.06473571  0.4324523  0.1957871 -0.04782909
## [5,]  0.38451654  0.8048213  0.6035164  0.39037281
##
## $n
## [1] 10 10 10 10
##
## $conf
##             [,1]        [,2]        [,3]        [,4]
## [1,] -0.53767365 -1.01441995 -0.75897825 -0.56117328
```

```
## [2,] -0.04087365 -0.03127218 -0.04968364 -0.07178116
##
## $out
## [1] 2.696078 2.482965 2.605810 2.703385
##
## $group
## [1] 1 2 3 4
##
## $names
## [1] "2015"             "2016"             "2017"
## [4] "1st half of 2018"
##
## attr(,"mahalanobis")
##       Syria Afghanistan        Iraq     Albania     Eritrea        Iran
##        7.63       36.48        4.47          NA       -0.02          NA
##      Kosovo     Unknown     Nigeria    Pakistan      Serbia   Russ.Fed.
##          NA        0.29          NA          NA          NA          NA
##      Turkey     Somalia   Macedonia     Georgia
##          NA          NA          NA          NA
```

As expected, the number of Syrian applications is beyond the inter-quartile range (with regard to the other numbers examined). From a multivariate perspective, however, applications from Afghanistan are even more outstanding.

Last but not least, let's return to our document on family reunification and have a short look at its content by building a `wordcloud` of the most frequent terms (omitting stopwords and numbers from 1 to 100).

```
tdm=vecToTDM(doc,plot=F)
fre=sort(rowSums(tdm))
fre=fre[is.na(match(names(fre),c(0:100,quanteda::stopwords("English"))))]
set.seed(0);wordcloud::wordcloud(names(fre),fre,min.freq=30,random.order=F)
```

References

- Bittermann, A., & Fischer, A. (2018). How to identify hot topics in psychology using topic modeling. *Zeitschrift fuer Psychologie, 226*, 3-13.
- Fischer, A., Holt, D., & Funke, J. (2018). Web-Scraping the JDDM Database: Citations, Reads and Downloads. *Journal of Dynamic Decision Making, 4* (4), 1-5.