Data preparation

The following section describes the steps of scraping and cleaning as well as the explorative data analysis. In this context scraping is the step of automated downloading raw HTML-files from a specific news-source. Cleaning describes the extraction of selected text elements, i. e. the article itself, the title or date it was published. Although it is possible to combine the steps and directly extract the article we chose this method to be able to access the raw HTML-files in case of errors during the scraping process.

Since this project aims to do research not on one specific newspaper homepage but rather a whole selection (Year of revolution: An overview of the Arab Spring demonstrations), the process of scraping homepages and extracting data or cleaning differs for each site.
Since the process is very similar, we decided to choose Hespress to document the code in depth. The remaining commeneted functions are to be found in the code folder.

The packages used during scraping are mainly rvest and tidyr. Rvest provides functionalities to download and examine homepages as XML. Tidyr is used to pipe the output of one R command into another command, usually to narrow down XML-elements in a given homepage. This makes the code shorter and easier to read. Both packages are used during scraping and cleaning.

```
libs<-c("rvest","stringr","tidyr","curl")
for(i in 1:length(libs)){
  suppressPackageStartupMessages(library(libs[i], character.only = TRUE))
}
```

Scrape day

After loading the packages and the functions implemented in D.1 a time sequence is set which is to be downloaded from the specific news source (function is left out). This sequence consists out of date strings. Hespress organizes its' articles in the following way: The URL is composed out of a root, the date an the string 'index.$.html', since there is more than one index page for each day. The strings can be concatinated by sapply.

```
days.to.scrape<-f.generateTimeSequence("2010/1/1","2010/12/31")
hp.base<-"http://www.hespress.com/archive/"
days.to.scrape.url.v<-sapply(hp.base,paste,days.to.scrape,"/index.1.html",sep="")
```

In the following, the function f.scrape.day.hespress is called with sapply. sapply applies a function on each element of a vector, in this case holding links to homepages.

```
target.folder <- "~/Documents/hespress"
# sapply(days.to.scrape.url.v, f.scrape.day.hespress, target.folder)
```

The function navigates to all top level index pages which have links to articles and saves them in a variable day.links. This vector is passed to the function f.scrape.index.hespress which does the actual scraping. The algorithm stops when all of the index pages have been checked and their articles have been downloaded. I.e. the next page is empty and thus does not contain the string 'index'.

```
f.scrape.day.hespress <- function(hespress.url, target.folder) {
  while (length(grep("index", hespress.url)) > 0) {
    tryCatch({
      homepage <- read_html(hespress.url)
      link.css.element <- "h2.section_title a"
      day.links <- homepage %>% html_nodes(link.css.element) %>% html_attr("href")
      ## Each index page has several articles which are to be saved.
      f.scrape.index.hespress(day.links, target.folder)
```

```
      # go to next index ('increment index')
      hespress.url <- homepage %>% html_node("span.page_active+a") %>% html_attr("href")
    }, error = function(e) {
      write(hespress.url,
            paste(target.folder, "missed.log", sep = ""),
            append = TRUE)
    })  # end of tryCatch
  }  # end of while-loop
}  # end of scrape.day.hespress
```

In the next step these links are loaded as XML and the resulting homepage is saved in a target folder. The name of these files corresponds to the URL of the article homepage. In this way homepages which are linked more than once are downloaded just once.

```
f.scrape.index.hespress <- function(day.links, target.folder) {

  for (link in day.links) {
    file <- paste(target.folder, gsub("/", "_", link), sep = "")
    ## if the file is already downloaded, skip it.
    if (!file.exists(file)) {
      tryCatch({
        article.url <- paste("http://www.hespress.com", link, sep = "")
        article.homepage <- read_html( article.url, encoding = "UTF-8")
        write_xml(article.homepage, file)
        sleep(0.1)
      } , error = function(e) {
        write(link,
              paste(target.folder, "missed.log", sep = ""),
              append = TRUE)
      })  # end of tryCatch
    }  # end of if file.exists
  }  # end of for-loop
}  # end of scrape.index.hespress
```

To keep the data format as easy as possible, we only extracted each article and corresponding date. This enables us to identify each article with an uri, an uniform resource identifier consisting out of a short form of the newspaper as shown in table (), the date as concatenated string without separators and an index starting at the first day of our chosen timespan. HP20101201$1 would identify an article written on the 1st of December 2010. Al-Masri Al-Yawm only uses Arabic numerals making it difficult to extract them as numbers. In this case only an index is used to identify an article. Other pages like Ahram used month names instead of numbers, thus had to be converted using a self written function replaceMonthNames (D.3). The identifiers and the articles are saved in the cross platform format comma-separated value (csv).

```
source.folder<- target.folder
# clean.hespress(source.folder)
```

Having saved all corresponding data in the target folder the data is ready to be extracted. The extraction-function expects a source-folder. Each file in this folder is first loaded as XML into the R environment. Then the script navigates to preselected HTML-nodes and extract the article fragments as well as the date. In this example there is but one article structure and each element remains at the same place making it easy to split the date string and extract the article. The elements are appended to a target csv file. Any files which do not contain an article are noted in a log file.

```
f.clean.hespress <- function(source.folder) {

  my.target.csv <- paste(source.folder, ".csv", sep = "")
```

```
  article.css.element <- "div#article_body p"
  title.css.element <- "h1.page_title"
  date.css.element <- "span.story_date"

  for (my.filename in dir(source.folder, full.names = TRUE, no.. = TRUE)) {
    tryCatch({
      article.homepage <- read_html(my.filename, encoding = "UTF-8")

      date.c <- article.homepage %>% html_nodes(date.css.element) %>% html_text()
      date <- str_split(date.c, " ")
      month <- date[[1]][3]
      day <- date[[1]][2]
      year <- date[[1]][4]

      text.c <- article.homepage %>% html_nodes(article.css.element) %>% html_text()
      text.c <- gsub("\"", "", text.c)  # remove quotes

      write.table(data.frame(year, month, day, text.c),
        col.names = FALSE, row.names = FALSE,
        file = my.target.csv, sep = ",", fileEncoding = "UTF-8",
        append = T )
    }, error = function(e) {
      write(paste("page", my.filename, "could not be loaded", sep = " "),
            "log", append = TRUE)
    })  # end of tryCatch
  }  # end of for-loop
}  # end of f.clean.hespress
```

The final renaming which results in the format explained in the previous paragraph can now be applied. This was done in an extra step, because we decided to narrow down the timespan and change the format of the Uri. After loading the hespress corpus file articles with no characters are sorted out. To include the month into the Uri and select articles written in a specific month, the month names are replaced by digits.

```
source.file <- "/home/tobias/Schreibtisch/hespress2010.csv"
hespress.corpus <- read.csv(
  file = source.file,
  fileEncoding = "UTF-8", sep = ",",
  header = FALSE, stringsAsFactors = F
)
hespress.corpus <- hespress.corpus[hespress.corpus$V4 != "",]
hespress.corpus <- f.replaceMonthNames(hespress.corpus, month.col = 2)
```

In the following all articles written in December 2010 are selected. If the csv file contains articles of several years, i.e. 2010 and 2011, these years have to be preselected, too.

```
dec <- hespress.corpus[which(hespress.corpus$V2 == 12),]
ids <- 1:dim(dec)[1]
# Prepending the short form of the news-source to date and id.
uri <- paste("HP", dec$V1, dec$V2, SPRINTF(dec$V3), "$", ids, sep = "")

new.corpus <- data.frame(uri, dec$V4 , stringsAsFactors = F)
# write.table( new.corpus,
#   col.names = FALSE, row.names = FALSE,
#   "/home/tobias/Dropbox/Dokumente/islamicate2.0/reduced/hespress.csv",
#   sep = ",", fileEncoding = "UTF-8", append = F
```

```
# )
```