Appendix 1 - Commented example code

The example code below illustrates a routine to automatically scrape the reactions to a post of a public Facebook page (such as the page of a politician) in R. Unlike *profiles*, which are to be regarded as the private online-personas of Facebook users, pages are public entities of which posts are visible to all Facebook users, regardless of whether they are connected to the page (via "liking" it) or not.

To run, the code requires a recent version of R (it was tested on version 3.4.4) and of Google Chrome (version 67.0.3396.62).

Install the requested dependencies

The code below installs (if they are not already installed) the packages wdman, binman, XML, devtools, getPass, and RSelenium, and loads them.

```
#================================================================
==========
# INSTALL/LOAD PACKAGES
#================================================================
==========
want = c("XML", "devtools","getPass")
have = want %in% rownames(installed.packages())
if ( any(!have) ) { install.packages( want[!have] ) }

# Check for RSelenium (not on the CRAN)
if ( !"RSelenium" %in% rownames(installed.packages()) ) {
    devtools::install_github("ropensci/RSelenium")
    devtools::install_github("ropensci/wdman")
    devtools::install_github("ropensci/binman")
}
want <- c(want, "RSelenium")

# Load all packages
junk <- lapply(want, library, character.only = TRUE)
rm(have, want, junk)
```

Set up the parameters for the scraping

The code below will save some useful parameters that will be used for the scraping. Specifically:

- `user`: the Facebook ID of the user performing the scraping.
- `user_path`: a folder on the local hard drive in which the user will insert the newly-created Chrome profile, e.g. "C: \User\Desktop".
- `post_id`: the numeric ID of the public post the reactions to which the user wants to scrape. For this example, we will use a post by the Italian politician Giulio Cavalli which can be found at this link.
- `timeout`: the minimum and maximum "sleep" time –that is, the time in which the program will be idle, to avoid clogging the Facebook server with repeated requests.
- `seed`: the number used to initialize the pseudo-randomization for the MD5 hash conversion of the user links (that is, the unique user IDs that can be used to trace individual users within the network of the collected data). Once user links are encrypted, it is impossible to reverse the process and go back to the original user links. However, if the seed number is the same across runs of the function (e.g. to scrape reactions to different posts) the encrypted ID will be the same for each individual user.

The password to the Facebook profile will be asked directly while the routine is running, so it is not necessary to store it into an object.

```
user <- "user@email.com"
user_path <- "folder"
post_id <- "10155919157297756"
timeout <- c(2,4)
seed <- 1234
```

Create a new Chrome profile

The code below will create a brand-new Google Chrome profile, necessary to store all the login information that will be used to automatically access Facebook via browser.

```
#================================================================
==========
# CREATES PROFILE FOLDER
#================================================================
==========
# Loads the right version of chromedriver and passes the profile
```

```
options.
# It creates the folder needed to scrape the data.

cDrv <- wdman::chrome(version = "2.40", verbose = FALSE, check =
TRUE)
eCaps <- RSelenium::getChromeProfile(dataDir = user_path,
                                     profileDir = "Profile 1")
eCaps$chromeOptions$args[[3]] <- "--disable-notifications"
remDr <- RSelenium::remoteDriver(remoteServerAddr = "localhost",
                                 browserName = "chrome",
                                 port = 4567L,
                                 extraCapabilities = eCaps)

# Gets the password
pass <- getPass::getPass("Enter your Facebook profile's password: ")

# Opens an automatic Chrome session and saver user and password in
the folder
tryCatch({
    suppressMessages({

        remDr$open()
        remDr$navigate("http://www.facebook.com")

        remDr$findElement("id",
"email")$sendKeysToElement(list(user))
        remDr$findElement("id",
"pass")$sendKeysToElement(list(pass))
        remDr$findElements("id", "loginbutton")[[1]]$clickElement()

    })
},
error = function(e) {
    remDr$close()
    cDrv$stop()
    stop("something went wrong. Check your internet connection and
try again.",
         call. = FALSE)
}
)
string <- substr(remDr$getCurrentUrl()[[1]], 1, 49)
if (string == "https://www.facebook.com/login.php?login_attempt=") {
    remDr$close()
    cDrv$stop()
    stop(paste0("something went wrong. Check your e-mail and
password
                and try again.\n
                Please erase the folder in ", user_path),
         call. = FALSE)
```

```
} else {
    # Close ports
    remDr$close()
    cDrv$stop()
}
```

Scrape the reactions to a post from a public page

In the code below, we instruct the headless browser to go on the page showing the reactions to the post, find the "See more" button in case the reactions are not all loaded, click on the button until it disappears, and download the whole HTML of the page. Then, the code parses the HTML to obtain the list of reactions by type, together with the URL of the users performing the reaction. The latter is particularly valuable, as it will be used to trace the users among different posts in the data.

```
#================================================================
==========
# SCRAPES REACTIONS
#================================================================
==========
# Loads the right version of chromedriver
# and passes the profile options from 'user_path'
cDrv <- wdman::chrome(version = "2.40", verbose = FALSE, check =
TRUE)
eCaps <- RSelenium::getChromeProfile(dataDir = user_path,
                                     profileDir = "Profile 1")
eCaps$chromeOptions$args[[3]] <- "--disable-notifications"
remDr<- RSelenium::remoteDriver(remoteServerAddr = "localhost",
                                browserName = "chrome",
                                port = 4567L,
                                extraCapabilities = eCaps)


# Opens a session and goes to the chosen page
remDr$open(silent = TRUE)
fr <-
"https://www.facebook.com/ufi/reaction/profile/browser/?ft_ent_ident
ifier="
url_id <- paste0(fr, post_id)
tryCatch({
    remDr$navigate(url_id)
},
error = function(e) {
    remDr$close()
```

```
    cDrv$stop()
    stop("something went wrong. Check your internet connection and
try again.",
         call. = FALSE)

}
)

# Scrapes the html by type of reaction
var <- remDr$findElements("class name", "_5i_p")

# Prepare lists
elemtxt <- list(NULL)
elemxml <- list(NULL)
link_list <- list(NULL)

# Loop around reaction types
i <- 1
while(i <= length(var)) {
    elemtxt[[i]] <- var[[i]]$getElementAttribute("outerHTML")[[1]]
    reac <- as.numeric(sub(".*reaction_profile_browser(\\d+).*",
"\\1",
                           elemtxt[[i]]))
    reac_t <- ifelse(reac == 1, "Like",
                     ifelse(reac == 8, "Angry",
                            ifelse(reac == 7, "Sad",
                                   ifelse(reac == 3, "Wow",
                                          ifelse(reac == 2, "Love",
                                                 ifelse(reac == 4,
"Haha",
                                                        ""))))))

    print(reac_t)
    elemxml[[i]] <- XML::htmlTreeParse(elemtxt[[i]],
                                       useInternalNodes = T,
                                       encoding = "UTF-8")
    repeat{
        delayedAssign("break.func", {break})
        tryCatch({
            suppressMessages({

                # Find "See more" button
                butt <- paste0('//a[contains(@href,
"reaction_type=',
                               reac,
                               '")]')

                seemorecounter <- remDr$findElement(using = 'xpath',
butt)
```

```
                seemorecounter$clickElement()
                Sys.sleep(stats::runif(1, min = min(timeout),
                                       max = max(timeout)))
            })
        },
        error = function(e) {
            force(break.func)
        }
        )
    }
    var <- remDr$findElements("class name", "_5i_p")
    elemtxt[[i]] <- var[[i]]$getElementAttribute("outerHTML")[[1]]
    elemxml[[i]] <- XML::htmlTreeParse(elemtxt[[i]],
                                       useInternalNodes = TRUE,
                                       encoding = "UTF-8")
    ## Parse list of user links
    prof_link <- unique(XML::xpathSApply(elemxml[[i]], "//a",
                                         XML::xmlGetAttr,
                                         "href"))
    prof_link <- prof_link[!grepl("/browse/mutual_friends/",
prof_link)]
    prof_link <- prof_link[prof_link!="#"]
    prof_link <- gsub("profile.php\\?id=", "", prof_link)
    prof_link <-  sub("[?].*$", "", prof_link)
    prof_link <-  sub("[&].*$", "", prof_link)
    link_list[[i]] <- prof_link
    names(link_list)[[i]] <- reac_t
    i <- i + 1
}

## [1] "Like"
## [1] "Love"

# Close ports
remDr$close()
cDrv$stop()

## [1] TRUE
```

As reported by the output, the post in this example had only “Like” and “Love” reactions.

Encrypt the data and put them into a data frame

In the code below we put all the information together into a data frame, i.e. an R table object which can be saved in various formats and used for subsequent analyses. Importantly, the code randomizes the order of the vector of scraped reactions (so

e.g. the first user reacting with a “Like” in the data is not the first user reacting with a “Like” in the reactions page) and performs the pseudonymization by applying MD5 encryption to the user URLs, based on the seed specified at the beginning.

```
# Randomize the order of the scraped reactions so even by going to
the post URL
# it is not possible to match the users who performed the reactions
with those
# in the data
link_list <- lapply(link_list, function(x) sample(x, length(x)))

# Make data frame and encrypt the user links
link_data <- data.frame(post_id = post_id,
                        user_id =
sapply(as.character(unlist(link_list)),
                                           function(x)
digest::digest(x, "md5",

seed = seed)),
                        reaction = rep(names(link_list),
                                       sapply(link_list, length)),
                        stringsAsFactors = FALSE)
row.names(link_data) <- 1:nrow(link_data)

# Show the data
head(link_data)
```

```
##             post_id                           user_id reaction
## 1 10155919157297756 1fca391750e53b9c30fee19063c4991a     Like
## 2 10155919157297756 1a8d52ad7f1088ecdc57f95fec0be9c5     Like
## 3 10155919157297756 7f8a5fcdc91cd0dc35077f5da8001237     Like
## 4 10155919157297756 1d87d97ac2cda3087a966f3fee780cbc     Like
## 5 10155919157297756 9d59e137505e45ed61defeec1979cf05     Like
## 6 10155919157297756 fa2073ed23231847660e91a021abc8bc     Like
```