Scraping useR2017! attendee data by Romain Francois

The following code is a good scraping example created by Romain Francois per here.

We can just play with it and learn from the great example provided.

Romain Francois' original code

Here is Romain Francois' original code:

```
library(rvest)
library(purrr)
library(tibble)
library(dplyr)
library(stringr)

attendees <- function(page) {
  extract <- function(class) html_node( persons, class ) %>% html_text() %>% str_trim()

  url <- paste0( 'https://user2017.sched.com/directory/attendees/', page )
  persons <- read_html(url) %>%
    html_nodes(".sched-person")
  tibble(
    position = extract(".sched-event-details-position"),
    company  = extract(".sched-event-details-company"),
    id = extract("h2:nth-child(2) a"),
    img = html_node(persons, "img") %>% html_attr("src")
  )
}

data <- map_df( 1:6, attendees )
data %>%
  summarise( profile = sum(!is.na(img)) / n() )
```

A few things stand out about this elegant code:

- The `tidyverse` set of packages are used to derive consistent coding practices
- The pipe operator is used to control the flow of the program
- The `purrr` package is used to map (as opposed to loop) through the multiple webpages and derive the combined dataset

Some minor modifications

I stepped through this code line by line and realized there was not much to improve! Here is the minimally revised code and outputs:

```
# ipak function: install and load multiple R packages.
# check to see if packages are installed. Install them if they are not, then load them into the R sessi
# source: https://gist.github.com/stevenworthington/3178163

ipak <- function(pkg){
```

```
    new.pkg <- pkg[!(pkg %in% installed.packages()[, "Package"])]
    if (length(new.pkg))
        install.packages(new.pkg, dependencies = TRUE)
    sapply(pkg, require, character.only = TRUE)
}

pckgs <- c("rvest", "tidyverse", "stringr", "xml2")
ipak(pckgs)
```

```
## Loading required package: rvest
## Loading required package: xml2
## Loading required package: tidyverse
## Loading tidyverse: ggplot2
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
## Loading tidyverse: dplyr
## Conflicts with tidy packages ----------------------------------------------
## filter(): dplyr, stats
## lag():    dplyr, stats
## Loading required package: stringr
##     rvest tidyverse   stringr      xml2
##      TRUE      TRUE      TRUE      TRUE
```

```
URL_BASE_USER_CONF <- "https://user2017.sched.com/directory/attendees/"

# The function by Romain Francois to obtain the required
# attendee data from the useR! 2017 conference

user_conf_2017_data <- function(page_num){

    # An auxillary function to extract and trim the
    # relevant html class from parsed html
    aux_extract <- function(html_class){
        html_node(persons, html_class) %>%
            html_text() %>%
            str_trim()
        }

    # Get the specific attendee list page for a single
    # page id
    url_user_conf  <- stringr::str_c(URL_BASE_USER_CONF
                                     , page_num)

    # Now that we have read the required html correctly we can
    # do the parsing
    persons <- xml2::read_html(url_user_conf) %>%
        rvest::html_nodes(".sched-person")

    # Create the tibble filled with user details
```

```
    attdee_data <- tibble(
        # Which page did the attendees profile appear
        attdee_page_num = page_num,
        attdee_id       = aux_extract("h2:nth-child(2) a"),
        attdee_company  = aux_extract(".sched-event-details-company"),
        attdee_position = aux_extract(".sched-event-details-position"),
        # "h2", or "h2 a" seems to work here, wonder why Romain used
        # used "h2:nth-child(2) a", seems to be unnecessarily complex
        attdee_img      = html_node(persons, "img") %>% html_attr("src"))

    return(attdee_data)
}

# Now get the data from all 6 attendee pages in a combined tibble!
attdee_data <- purrr::map_df(.x = 1:6
                             , .f = user_conf_2017_data) %>%
               print()
```

```
## # A tibble: 962 × 5
##    attdee_page_num       attdee_id attdee_company attdee_position
##              <int>           <chr>          <chr>           <chr>
## 1                1          a.levy
## 2                1            almd
## 3                1        bpiccolo
## 4                1           cderv
## 5                1 dario.bonaretti
## 6                1         dreznik
## 7                1     edwardhywel
## 8                1           gijsvk
## 9                1         janverc
## 10               1        philninh
## # ... with 952 more rows, and 1 more variables: attdee_img <chr>
```

```
# Let's look at the data
glimpse(attdee_data) %>% print()
```

```
## Observations: 962
## Variables: 5
## $ attdee_page_num <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ attdee_id       <chr> "a.levy", "almd", "bpiccolo", "cderv", "dario....
## $ attdee_company  <chr> "", "", "", "", "", "", "", "", "", "", "", ""...
## $ attdee_position <chr> "", "", "", "", "", "", "", "", "", "", "", ""...
## $ attdee_img      <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
## # A tibble: 962 × 5
##    attdee_page_num       attdee_id attdee_company attdee_position
##              <int>           <chr>          <chr>           <chr>
## 1                1          a.levy
## 2                1            almd
## 3                1        bpiccolo
## 4                1           cderv
## 5                1 dario.bonaretti
## 6                1         dreznik
## 7                1     edwardhywel
## 8                1           gijsvk
```

```
## 9                1         janverc
## 10               1        philninh
## # ... with 952 more rows, and 1 more variables: attdee_img <chr>
```

```
# From Romain's code, count the proportion of users
# with an image
attdee_data %>%
    dplyr::summarise(profile =
                         sum(!is.na(attdee_img))/ n()) %>%
    print()
```

```
## # A tibble: 1 × 1
##     profile
##       <dbl>
## 1 0.4230769
```

```
# We can also get the counts of attendees displayed
# on each of the 6 pages!
attdee_data %>%
    dplyr::group_by(attdee_page_num) %>%
    dplyr::summarise(count_id = n()) %>%
    dplyr::ungroup() %>%
    print()
```

```
## # A tibble: 6 × 2
##   attdee_page_num count_id
##             <int>    <int>
## 1               1      179
## 2               2      188
## 3               3      195
## 4               4      191
## 5               5      187
## 6               6       22
```

Some very minor improvements include:

- Defining the global base url of the useR2017! conference outside of the function. Then it was referenced inside the function as a variable makes the code seem easier to read and potentially more flexible for future conferences
- Adding in a Attendee page id reference to examine later the counts of attendees per page as a cross check of the compiled results
- Renamed the `extract` function to `aux_extract` to make it clear that this is not the `extract` function from any other package. This confused me up initially.
- Referenced the `tidyverse` packages directly to ensure a more concise loading of the required packages
- Explicit namespace referencing of the required packages to make it clear where each function comes from e.g. `dplyr::summarize` instead of just `summarize`. Reduces ambiguity and makes it easier for new users to clearly lear from the code.