Some time ago, Maëlle Salmon published a [very nice post](https://masalmon.eu/2018/06/18/mathtree/) showing how she scraped her mathematical family tree from the [Mathematics Genealogy Project](https://www.genealogy.math.ndsu.nodak.edu/). Of course I immediately wanted to produce my own! I am not a mathematician myself, but one of my PhD supervisor has a PhD in mathematics. Which makes me the indirect descendant of a long lineage of famous mathematicians! As Maëlle kindly invited me to share my tree on Twitter, I decided to write this post to show how I customized the appearance of my tree to make it more sexy. [It is actually part of my job to plot sexy phylogenetic tree…]. So, this post is a remix of Maëlle’s post which was a remix of Nathalie Vialaneix’s [post](http://tuxette.nathalievilla.org/?p=1292) 

**1. Scraping more data**

First I modified a bit Maëlle’s scraping function to collect more data. I added Xpath selectors for the degree, the university, the country, the year and the title of the thesis. I am not going to use all these informations but now they are here if you need them.

library(magrittr)

library(DiagrammeR)

library(dplyr)

library(igraph)

library(purrr)

library(stringr)

library(rvest)

library(xml2)

.get\_advisors <- function(id\_string = "id.php?id=143630", sleep\_time, terminal = FALSE){

# small break to be nice

Sys.sleep(sleep\_time)

# try to get the page

page <- glue::glue("https://www.genealogy.math.ndsu.nodak.edu/{id\_string}") %>%

httr::GET()

# try until it works but not more than 5 times

try <- 1

while(httr::status\_code(page) != 200 & try <= 5){

Sys.sleep(sleep\_time)

page <- glue::glue("https://www.genealogy.math.ndsu.nodak.edu/{id\_string}") %>%

httr::GET()

try = try + 1

}

# Now get student's data

student\_name <- httr::content(page) %>%

rvest::xml\_nodes(xpath = '//h2[@style="text-align: center; margin-bottom: 0.5ex; margin-top: 1ex"]') %>%

rvest::html\_text() %>%

stringr::str\_remove("\\\n")

degree <- httr::content(page) %>%

rvest::xml\_nodes(xpath = '//span[@style="margin-right: 0.5em"]/span/preceding-sibling::text()') %>%

rvest::html\_text()

university <- httr::content(page) %>%

rvest::xml\_nodes(xpath = '//span[@style="margin-right: 0.5em"]/span') %>%

rvest::html\_text()

year <- httr::content(page) %>%

rvest::xml\_nodes(xpath = '//span[@style="margin-right: 0.5em"]/span/following-sibling::text()') %>%

rvest::html\_text() %>%

stringr::str\_trim()

country <- httr::content(page) %>%

rvest::xml\_nodes(xpath = '//div[@style="line-height: 30px; text-align: center; margin-bottom: 1ex"]/img') %>%

rvest::html\_attr("title")

thesis\_title <- httr::content(page) %>%

rvest::xml\_nodes(xpath = '//span[@id="thesisTitle"]') %>%

rvest::html\_text() %>%

stringr::str\_remove\_all("\\\n")

# Get all nodes corresponding to advisors

# Thanks to their... formatting but it works

all\_advisors <- httr::content(page) %>%

rvest::xml\_nodes(xpath = "//p[@style='text-align: center; line-height: 2.75ex']") %>%

rvest::html\_nodes("a")

if(terminal){

name <- NA

id\_string\_advisors <- NA

} else {

name <- purrr::map\_chr(all\_advisors, rvest::html\_text)

id\_string\_advisors <- purrr::map\_chr(all\_advisors, rvest::html\_attr,

"href")

}

# Export results

tibble::tibble(student\_name = student\_name,

degree = list(degree),

university = list(university),

year = list(year),

country = list(country),

thesis\_title = list(thesis\_title),

id\_string\_student = id\_string,

name = name,

id\_string = id\_string\_advisors)

}

get\_advisors <- memoise::memoise(.get\_advisors)

#### Download and prepare data ####

# initial data.frame

df <- get\_advisors(me, 5)

new\_df <- df

keep\_growing <- TRUE

while(keep\_growing){

# get size to compare to size after a bit more scraping

nrow1 <- nrow(df)

# get advisors for all new lines

# from the previous iterations

new\_df <- purrr::map\_df(new\_df$id\_string, get\_advisors, sleep\_time = 30)

df <- unique(rbind(df, new\_df))

# if the data.frame didn't grow, stop

if(nrow(df) == nrow1){

keep\_growing <- FALSE

}

}

df <- df %>% mutate(student\_name = stringr::str\_trim(student\_name),

name = stringr::str\_trim(name))

terminal\_df <- setdiff(df$id\_string, df$id\_string\_student) %>%

map(get\_advisors, terminal = TRUE, sleep\_time = 30) %>%

bind\_rows() %>%

mutate(student\_name = stringr::str\_trim(student\_name),

name = stringr::str\_trim(name))

**2. Labels**

I wanted to create labels with the name of the mathematician, the flag of the country and the year. Displaying a picture in a node with graphviz (using DiagrammeR) was not simple. Apparently it is possible to use some kind of basic HTML to format nodes but I failed to include images. Finally I decided to use emojis via package emo ,it was fairly easy.

#### Construct the graph ####

# create nodes

labels <- unique(c(df$student\_name, df$name))

nodes\_df <- create\_node\_df(n = length(labels))

nodes\_df$label <- labels

df\_red <- df %>%

bind\_rows(terminal\_df) %>%

filter(!duplicated(student\_name)) %>%

right\_join(nodes\_df, by = c("student\_name" = "label"))

# create edges

edges\_df <- df[, c("name", "student\_name")]

edges\_df <- dplyr::left\_join(edges\_df, nodes\_df,

by = c("name" = "label"))

edges\_df <- dplyr::rename(edges\_df, from = id)

edges\_df <- dplyr::left\_join(edges\_df, nodes\_df,

by = c("student\_name" = "label"))

edges\_df <- dplyr::rename(edges\_df, to = id)

edges\_df <- mutate(edges\_df, rel = "a", color = "BurlyWood4")

# Create labels

years <- map\_chr(df\_red$year, paste, collapse = " ") %>%

map\_chr(str\_trim) %>%

str\_replace\_all("(?<=[0-9]{4}) (?=[0-9]{4})", ", ") %>%

str\_replace\_all(" /", ", ") %>%

paste0("(", ., ")") %>%

str\_replace\_all("\\(\\)", "")

df\_red$country <-

map\_if(df\_red$country,

(map\_lgl(df\_red$country,~ is.null(.x)) | map\_lgl(df\_red$country,~ length(.x) == 0)),

function(x) "") %>%

map(stringr::str\_replace, "UnitedKingdom", "United Kingdom")

country\_flag <- map\_if(df\_red$country,

map\_lgl(df\_red$country,~ any(!.x == "")),

function(x) map\_chr(x, ~ as.character(emo::flag(paste0("^", .x, "$"))))) %>%

map\_chr(paste, collapse = " ⋅ ")

label <- paste0(nodes\_df$label, "\n", years, "")

label <- paste0(nodes\_df$label, "\n", country\_flag, "\n", years, "")

nodes\_df$label <- label %>%

str\_replace\_all("'", " ") %>%

str\_replace\_all("[[:space:]]{2,}", " ")

**3. Adding my self**

nodes\_df <- bind\_rows(nodes\_df,

tibble(id = nrow(nodes\_df) + 1,

type = NA, label = "François Keck\n\n(2016)"))

edges\_df <- bind\_rows(edges\_df,

tibble(name = "AF", student\_name = "FK",

from = 1, type.x = NA,

to = nrow(nodes\_df), type.y = NA,

rel = "a", color = "BurlyWood4"))

**4. Customizing the style of the nodes**

I changed the color and shape of the nodes. For some obscure reasons the rectangle-based shapes of graphiz are not correctly rendered in Firefox on Ubuntu (labels are overlapping). It worked on Windows but then I couldn’t display my flags with colored emojis The only solution I found was to fix the font size manually (see next point).

# Customizing the nodes

nodes\_df <- mutate(nodes\_df,

shape = "note",

color = "Tan",

fillcolor = "Moccasin",

style = "filled")

# create a DiagrammeR dgr\_graph object

dgr <- create\_graph(nodes\_df = nodes\_df,

edges\_df = edges\_df[, c(-1, -2)],

directed = TRUE)

# export the object to igraph format

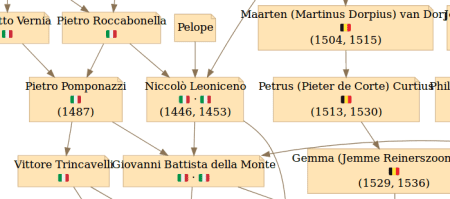
# and then write it to a GraphViz DOT file

DiagrammeR::to\_igraph(dgr) %>%

igraph::write.graph(file = "graph.dot",

format = "dot")

DiagrammeR::grViz("graph.dot", width = 4000, height = 5000)

[](http://www.pieceofk.fr/?attachment_id=585)

**5. The final touch**

The viz.js library which stands behind DiagrammeR renders DOT objects in the browser via SVG. SVG files are XML-based and therefore can be directly processed with R. This gives us great power to manipulate the look of our tree.

First, we need to convert the widget to static HTML/SVG. We can do that in command line using Chromium in headerless mode to render the widget page.

DiagrammeR::grViz("graph.dot", width = 4000, height = 5000) %>%

htmlwidgets::saveWidget("index.html")

system("chromium-browser --headless --dump-dom index.html > genealogy.html")

Finally, I used R and [xml2](https://cran.r-project.org/web/packages/xml2/index.html) to edit directly the SVG content and improve the look of the tree. In the code below I show how to fill the page and labels backgrounds with a texture image, how to fix the text size, and how to add a shadow effect on labels.

# Load and clean html

html <- read\_html("index.html")

xml\_find\_all(html, '//script') %>%

xml\_remove()

# Background

xml\_find\_all(html, '/html/body') %>%

xml\_set\_attr('style', 'background-image: url("ricepaper2.png"); margin: 0px; padding: 40px;')

xml\_find\_all(html, '/html/body/div/div/svg/g/polygon') %>%

xml\_set\_attr('fill', 'transparent')

# Labels text size

xml\_find\_all(html, '//text') %>%

xml\_text() %>%

str\_detect("(^[A-Z])|(^\\()") %>%

extract(xml\_find\_all(html, '//text'), .) %>%

xml\_set\_attr('font-size', '12')

# Labels background

xnodes <- xml\_find\_all(html, '//polygon[@fill="#ffe4b5"]')

xml\_set\_attr(xnodes, 'id', 'paper\_tag')

xml\_set\_attr(xnodes, 'fill', 'url(#img1)')

xml\_find\_all(html, '/html/body/div/div/svg') %>%

xml\_add\_child(read\_xml('

'), .where = 0)

# Labels shadow fx

xml\_find\_all(html, '//polygon[@id="paper\_tag"]') %>%

xml\_set\_attr('filter', 'url(#f3)')

xnodes <- xml\_find\_all(html, '//\*[@stroke="#d2b48c"]')

xml\_set\_attr(xnodes, 'stroke', '#000000')

xml\_set\_attr(xnodes, 'stroke-width', '0.5')

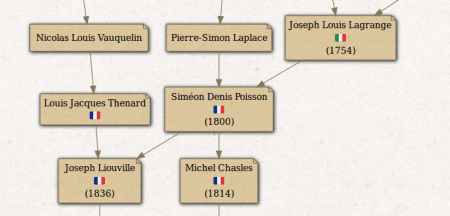
xml\_find\_all(html, '/html/body/div/div/svg') %>%

xml\_add\_child(read\_xml('

'), .where = 0)

write\_html(html, "genealogy.html")

I find the result pretty nice 

[](http://www.pieceofk.fr/?attachment_id=577)

And this is my mathematical family tree! I recognize some illustrious names here! Do you?

[](https://i1.wp.com/www.pieceofk.fr/wp-content/uploads/2018/07/Screenshot_2018-07-08-grViz.jpg)My tree

**Bonus:**

For a better parchment look, we can use the calligraphic font Tangerine for the labels. Note that some glyphs are unfortunately not supported by this font.

# Labels font.

xml\_find\_all(html, '//text') %>%

xml\_text() %>%

str\_detect("(^[A-Z])|(^\\()") %>%

extract(xml\_find\_all(html, '//text'), .) %>%

xml\_set\_attr('font-size', '20')

xml\_find\_all(html, '//text') %>%

xml\_text() %>%

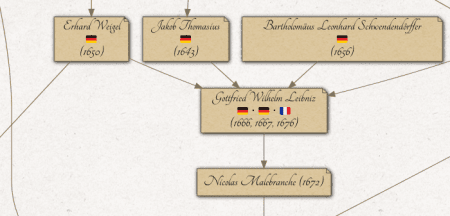
str\_detect("(^[A-Z])|(^\\()") %>%

extract(xml\_find\_all(html, '//text'), .) %>%

xml\_set\_attr('font-family', 'Tangerine')

xml\_find\_all(html, '/html/head') %>%

xml\_add\_child(read\_xml(''))

[](http://www.pieceofk.fr/?attachment_id=576)

[Facebook](http://www.facebook.com/sharer.php?u=http%3A%2F%2Fwww.pieceofk.fr%2F%3Fp%3D566&t=The%20Mathematics%20Genealogy%20Project%3A%20Customizing%20my%20mathematical%20family%20tree&%23038;s=100&%23038;p&%23091;url&%23093;=http%3A%2F%2Fwww.pieceofk.fr%2F%3Fp%3D566&%23038;p&%23091;images&%23093;&%23091;0&%23093;=http%3A%2F%2Fwww.pieceofk.fr%2Fwp-content%2Fuploads%2F2018%2F07%2FScreenshot_2018-07-10-grViz.png&%23038;p&%23091;title&%23093;=The%20Mathematics%20Genealogy%20Project%3A%20Customizing%20my%20mathematical%20family%20tree)