Understanding who you are is strongly related to understanding your family history. Discovering ancestors is now a popular hobby, as many archives are available on the internet. The GEDCOM format provides a standardised way to store information about ancestors. This article shows how to develop a GEDCOM reader using the R language.

**The GEDCOM format**

The GEDCOM format is not an ideal way to store information, but it has become the de-facto standard for family history. This format includes metadata and two sets of data. The file contains a list of the individuals, and it lists the families to which they belong.

The basic principle is that each line has a level, indicated by the first digit. At level zero, we find metadata and the individuals and their family. At level one, we see the various types of data, such as births, deaths and marriages. The deeper levels provide the data for these events.

The GEDCOM format is not only old in the way it stores data, but it is also limited in the types of human relationships. These files only store genetic relationships between people and assume that these relationships are marriages between a wife and a husband. Human relationships are, however, a lot more complicated than the genetic relationships between children and their parents, grandparents and ancestors.

These issues aside, all genealogical software can export a file to GEDCOM. The next section shows how to create a basic GEDCOM reader using the stringr, tibble and dplyr packages from the Tidyverse.

**GEDCOM Reader**

The read.gedcom() function takes a GEDCOM file as input and delivers a data frame (tibble) with basic information:

* ID
* Full name
* Gender
* Birthdate and place
* Father
* Mother
* Death date and place

This code only can be easily expanded to include further fields by adding lines in the while-loops and including the fields in the data frame.

The first lines read the file and setup the data frame. The extract() function extracts an individual’s ID from a line in the file. The for loop runs through each line of the GEDCOM file. When the start of a new individual is detected, the GEDCOM reader collects the relevant information.

Births and christenings are considered equal to simplify the data. In older data, we often only know one or the other. The function looks for the start of a family. It extracts the husband and wife and assigns these as parents to each of the children. The last section cleans the data and returns the result.

## Read GEDCOM file

## The Devil is in the Data

## Dr Peter Prevos

read.gedcom <- function(gedcom.loc) {

require(stringr)

require(tibble)

require(dplyr)

gedcom <- str\_squish(readLines(gedcom.loc))

idv <- sum(grepl("^0.\*INDI$", gedcom))

fam <- sum(grepl("^0.\*FAM$", gedcom))

cat(paste("Individuals: ", idv, "\n"))

cat(paste("Families: ", fam, "\n"))

family <- tibble(id = NA,

Full\_Name = NA,

Gender = NA,

Birth\_Date = NA,

Birth\_Place = NA,

Father\_id = NA,

Mother\_id = NA,

Death\_Date = NA,

Death\_Place = NA)

## Extract data

extract <- function(line, type) {

str\_trim(str\_sub(line, str\_locate(line, type)[2] + 1))

}

id <- 0

for (l in 1:length(gedcom)) {

if (str\_detect(gedcom[l], "^0") & str\_detect(gedcom[l], "INDI$")) {

id <- id + 1

family[id, "id"] <- unlist(str\_split(gedcom[l], "@"))[2]

l <- l + 1

while(!str\_detect(gedcom[l], "^0")) {

if (grepl("NAME", gedcom[l]))

family[id, "Full\_Name"] <- extract(gedcom[l], "NAME")

if (grepl("SEX", gedcom[l]))

family[id, "Gender"] <- extract(gedcom[l], "SEX")

l <- l + 1

if (grepl("BIRT|CHR", gedcom[l])) {

l <- l + 1

while (!str\_detect(gedcom[l], "^1")) {

if (grepl("DATE", gedcom[l]))

family[id, "Birth\_Date"] <- extract(gedcom[l], "DATE")

if (grepl("PLAC", gedcom[l]))

family[id, "Birth\_Place"] <- extract(gedcom[l], "PLAC")

l <- l + 1

}

}

if (grepl("DEAT|BURI", gedcom[l])) {

l <- l + 1

while (!str\_detect(gedcom[l], "^1")) {

if (grepl("DATE", gedcom[l]))

family[id, "Death\_Date"] <- extract(gedcom[l], "DATE")

if (grepl("PLAC", gedcom[l]))

family[id, "Death\_Place"] <- extract(gedcom[l], "PLAC")

l <- l + 1

}

}

}

}

if (str\_detect(gedcom[l], "^0") & str\_detect(gedcom[l], "FAM")) {

l <- l + 1

while(!str\_detect(gedcom[l], "^0")) {

if (grepl("HUSB", gedcom[l]))

husband <- unlist(str\_split(gedcom[l], "@"))[2]

if (grepl("WIFE", gedcom[l]))

wife <- unlist(str\_split(gedcom[l], "@"))[2]

if (grepl("CHIL", gedcom[l])) {

child <- which(family$id == unlist(str\_split(gedcom[l], "@"))[2])

family[child, "Father\_id"] <- husband

family[child, "Mother\_id"] <- wife

}

l <- l + 1

}

}

}

family %>%

mutate(Full\_Name = gsub("/", "", str\_trim(Full\_Name)),

Birth\_Date = as.Date(family$Birth\_Date, format = "%d %b %Y"),

Death\_Date = as.Date(family$Death\_Date, format = "%d %b %Y")) %>%

return()

}

**Analysing the data**

There are many websites with GEDCOM files of family histories of famous and not so famous people. The [Famous GEDCOMs](https://webtreeprint.com/tp_famous_gedcoms.php) website has a few useful examples to test the GEDCOM reader.

Once the data is in a data frame, you can analyse it any way you please. The code below downloads a file with the presidents of the US, with their ancestors and descendants. The alive() function filters people who are alive at a certain date. For people without birth date, it sets a maximum age of 100 years.

The histogram shows the distribution of ages at time of death of all the people in the presidents file.

These are just some random examples of how to analyse family history data with this GEDCOM reader. The next article will explain how to plot a population pyramid using this data. A future article will discuss how to visualise the structure of family history.

Birth years of people of in the Presidents file

## Basic family history statistics

library(tidyverse)

library(lubridate)

source("read.gedcom.R")

presidents <- read.gedcom("<https://webtreeprint.com/tp_downloader.php?path=famous_gedcoms/pres.ged>")

filter(presidents, grepl("Jefferson", Full\_Name))

mutate(presidents, Year = year(Birth\_Date)) %>%

ggplot(aes(Year)) +

geom\_histogram(binwidth = 10, fill = "#6A6A9D", col = "white") +

labs(title = "Birth years in the presidents file")

ggsave("../../Genealogy/years.png")

alive <- function(population, census\_date){

max\_date <- census\_date + 100 \* 365.25

filter(people, ([is.na](http://is.na)(Birth\_Date) & (Death\_Date <= max\_date &

Death\_Date >= census\_date)) |

(Birth\_Date <= census\_date & Death\_Date >= census\_date)) %>%

arrange(Birth\_Date) %>%

mutate(Age = as.numeric(census\_date - Birth\_Date) / 365.25) %>%

return()

}

alive(presidents, as.Date("1840-03-07"))