# Estimating Williams College Faculty Age

*Zafer Cesur ('19)*

*2016-02-08*

---

**Abstract** This paper describes the R package **ager**. The package is focused on gathering the graduation data of Williams College faculty in order to construct statistical models for faculty age. The paper outlines the functions for selection and organization of the data, which are designed to require little to no manual manipulation during pre-processing and transformation thereof and be as applicable as possible so that it can easily be used for future years. The usage of these functions is illustrated and the results are demonstrated with descriptive histograms.

---

## Introduction:

The package **ager** provides the functionality for easily accessing educational background of Williams College faculty members that consist of professors, lecturers and postdoctoral fellows. This is useful because assuming that most people graduate when they are 22 years old, we can estimate faculty's age with a small margin of error.

Brief academic history of faculty members is available on the course catalogs that can be accessed online on Williams' website. Let us explore and compare some excerpts taken from these catalogs. For instance, this is how it looked in 2009-2010:

> Colin C. Adams, Thomas T. Read Professor of Mathematics
> B.S. (1978) M.I.T.; Ph.D. (1983) University of Wisconsin

And this in 2013-2014:

> Colin C. Adams, Thomas T. Read Professor of Mathematics, 1978, BS, MA Institute of Technology, 1983, PHD, University of WI, Madison

And finally, in 2015-2016:

> Adams,Colin C., Mathematics and Statistics

As we can observe, the format has significantly changed over the years. Also notice that in the most recent year, we do not get informed about the educational background of the professor, but rather only on the faculty he belongs to. This also holds true for the previous year, 2014-2015. The lack of information and the unpredictability of the presentation format urged us to devise a standard structure so that the data can be represented in a concise way and to develop an indirect method so that the desired information can be extracted.

The rest of the paper is organized as follows. First we are going to explain how we organized the data. Then, we are going to introduce the methods we used to extract missing data. The last section concludes the article by presenting histograms and plots.

---

## Methods

This section presents the methodology used in the package **ager**.

### 1. Accessing and converting archived documents

The catalog files are downloaded from the Williams' website in PDF format. Then, using A-PDF Text Extractor, the selected pages are converted into TXT files. The desired pages vary from year to year based on the content of the catalog; for the academic year 2015-2016 we need pages from 396 through 398. Here is the workflow that we used to prepare the data we need:

**1.** Run A-PDF Text Extractor
**2.** Open bulletin2015_16.pdf
**3.** In the options tab, select pages from 396 to 398 and make sure *In PDF Order* is chosen as the method for extraction.
**4.** Extract text
**5.** Edit the lines that end prematurely, by hand. There were 2 manual editings for this document:

- Gibson,Matthew, Economics Department
- Das,Joanna Dee, Dance Department

The workflow for other years is the same. The text files for the most recent three academic years is available at */inst/extdata*.

### 2. Cleaning

The next step is removing irrelevant parts of the text file.

```r
# Open up a txt file and store lines of information as character strings of the
# vector 'flat_file'
flat_file <- paste("2015-16", ".txt", sep = "") %>%
  system.file("extdata", ., package = "ager") %>%
  readLines(skipNul = TRUE)
  head(flat_file, 12)
```

```
##  [1] "ÿþ"                    ""                     ""
##  [4] ""                      "  "                   ""
##  [7] "FACULTY 2015-2016  " ""                     "  "
## [10] ""                      ""                     ""
```

As we can see, there is a myriad of unwanted lines in the beginning of the document. We also have to get rid of empty lines inbetween and of the elements containing page numbers. We can remove these with the following set of codes.

```r
# Remove leftovers from the previous page. In this case, we do not have any, however
# there are, for instance in the catalog from the year 2010-2011, some.
relevant_line <- grep("FACULTY", flat_file)[1]
flat_file <- flat_file[-seq(1, relevant_line - 1)]

# Remove all other irrelevant data at the beginning of the document until coming
# across a line containing a comma
relevant_line <- grep(",", flat_file)[1]
```

```r
flat_file <- flat_file[-seq(1, relevant_line - 1)]

# Remove the strings which are of length 0 and those containing page numbers. 4 is not
# chosen arbitrarily but rather based on other years' documents. For instance, in the
# previous 2 years, the strings that contain pages are formatted as 3-digit integer and
# a space.
flat_file <- setdiff(flat_file, flat_file[nchar(flat_file) <= 4])
head(flat_file)
```

```
## [1] "Aalberts,Daniel P., Physics Department   "
## [2] "+Acha,Beverly D., Art Department   "
## [3] "Adams,Colin C., Mathematics and Statistics   "
## [4] "Adhami,Zaid, Religion Department   "
## [5] "Adler Mandelbaum,Sara E., Economics Department   "
## [6] "Albrecht,Jeannie R, Computer Science Department   "
```

After the basic structure is established, we need to make sure no character string is stored as 2 separate ones. This can be caused while converting PDF documents if soft line breaks are converted to hard line breaks. For this document we do not need to worry about it as the lines are already very short. But for the year 2013-2014, for example, we need to run the following as the last step of cleaning process to make sure it flows smoothly.

```r
short_lines <- order(nchar(flat_file), decreasing = FALSE)[1:15]
long_lines <- order(nchar(flat_file), decreasing = TRUE)[1:15]

# Evaluate which of the long lines are immediately followed by a short line
cut_lines <- short_lines[(short_lines - 1) %in% long_lines]

# Join the cut off line with the line above
flat_file[cut_lines - 1] <- paste(flat_file[cut_lines - 1], flat_file[cut_lines], sep=" ")

# Remove cut off lines
flat_file <- setdiff(flat_file, flat_file[cut_lines])
```

Of course, this step requires an assumption and an arbitrary integer to be chosen and there might be a better algorithm or program suited for this, but I have observed that it is precise enough for our purposes.

This section can be reproduced simply with the *scrub_catalog* function.

```r
academic_year <- "2015-16"
flat_file <- scrub_catalog(academic_year)
head(flat_file)
```

```
## [1] "Aalberts,Daniel P., Physics Department   "
## [2] "+Acha,Beverly D., Art Department   "
## [3] "Adams,Colin C., Mathematics and Statistics   "
## [4] "Adhami,Zaid, Religion Department   "
## [5] "Adler Mandelbaum,Sara E., Economics Department   "
## [6] "Albrecht,Jeannie R, Computer Science Department   "
```

## 3. Collecting data

### 3.1 Names

For the third part, it is crucial that we collect the names of faculty members from the flat file we have just recently obtained.

```
# Obtain first names using regular expressions
first_names <-
  ",(| )(.*?)," %>%
  regexpr(flat_file) %>%
  regmatches(flat_file, .) %>%
  gsub(",(| )|,", "", .)
head(first_names)
```

```
## [1] "Daniel P."  "Beverly D." "Colin C."   "Zaid"       "Sara E."
## [6] "Jeannie R"
```

It finds and returns the first match inbetween two commas. Although there is no space after the first comma in this document, (| ) is added to the pattern in order not to lose generality.

```
  # Similarly,
last_names <-
  "[A-Z](.*?)," %>%
  regexpr(flat_file) %>%
  regmatches(flat_file, .) %>%
  gsub(",.*", "", .)
head(last_names)
```

```
## [1] "Aalberts"          "Acha"              "Adams"
## [4] "Adhami"            "Adler Mandelbaum" "Albrecht"
```

This one explicitly indicates that the match should start with a capital letter. This is in order to get rid of special characters that represent if a faculty member is visiting or on leave, and also to get rid of spaces that are present in some documents.

```
  names <- paste(first_names, last_names, sep = " ")
head(names)
```

```
## [1] "Daniel P. Aalberts"       "Beverly D. Acha"
## [3] "Colin C. Adams"           "Zaid Adhami"
## [5] "Sara E. Adler Mandelbaum" "Jeannie R Albrecht"
```

This section can be reproduced with the function *collect_names* by passing *TRUE* value to its *reformat* argument.

```
names <- collect_names(flat_file, reformat = TRUE)
```

### 3.2 Departments

We can also collect the department data from the year 2015-2016 in a similar way.

```
departments <- collect_departments(flat_file)
unique(departments)
```

```
##  [1] "Physics"                          "Art"
##  [3] "Mathematics and Statistics"       "Religion"
##  [5] "Economics"                        "Computer Science"
##  [7] "German and Russian"               "Music"
##  [9] "Comparative Literature"           "Geosciences"
## [11] "Athletics"                        "Biology"
## [13] "Theatre"                          "Romance Languages"
## [15] "Chemistry"                        "English"
## [17] "Philosophy"                       "History of Science"
## [19] "History"                          "Africana Studies"
## [21] "Dance"                            "Ctr-Environmental Studies"
## [23] "Latina/o Studies"                 "Leadership Studies"
## [25] "Asian Studies"                    "Classics"
## [27] "Graduate-Art History"             "Psychology"
## [29] "Gender & Sexuality Stdy"          "Political Science"
## [31] "Astronomy"                        "Humanities"
## [33] "Anthropology and Sociology"       "Williams-Mystic"
## [35] "American Studies"                 "Neuroscience"
## [37] "Oakley Ctr for Human & Soc Sci" "Global Studies"
```

## 4. Extracting data

### 4.1 From another catalog

Now using the names we have gathered, we can obtain information from another document that has the desired content. For instance, so as to access the information we need for the analysis of the academic year 2015-16, we can extract data from the catalog of the year 2013-14 using the names of the faculty members who are common. We store the source catalog in another flat file called *data_source*, but we do not need to collect names from this data source because it is unnecessary.

```
data_source <- scrub_catalog("2013-14")
pattern <- "\\d{4}, (B\\w+|AB), (.*?),"
academic_data <- sapply(names, function(x){
  vector <-
    regexpr(pattern, data_source[grep(x, data_source)]) %>%
    regmatches(data_source[grep(x, data_source)], .)
  if(length(vector) == 0){
    return(NA)
  }else{
    return(vector)
  }
})
attr(academic_data[1:3], "names")
```

```
## [1] "Daniel P. Aalberts" "Beverly D. Acha"    "Colin C. Adams"
```

```
unname(academic_data[1:3])
```

```
## [1] "1989, BS, MA Institute of Technology,"
## [2] NA
## [3] "1978, BS, MA Institute of Technology,"
```

### 4.2 From the web directory

Using the names we have gathered in **3.1**, we can create search queries to further investigate desired data online by making use of Williams' directory, which can generically accessed with the URL format http://www.williams.edu/people/?s_directory=firstname+lastname without middle names. We need the following lines of codes.

```
# The carrot makes sure that we only get the first word of each names element.
first_names <- regmatches(names, regexpr("^\\w+", names))

# The optional \\w+[-] pattern makes sure that we take hypenated last names such as
# 'Robert Baker-White' into account.
last_names <- regmatches(names, regexpr("(\\w+[-])?\\w+$", names))

# Generate search queries in order to access data on the web directory
search_queries <- paste(first_names, "+", last_names, sep = "")
search_links <- paste("http://www.williams.edu/people/?s_directory=", search_queries,
                sep = "")
head(search_links)
```

```
## [1] "http://www.williams.edu/people/?s_directory=Daniel+Aalberts"
## [2] "http://www.williams.edu/people/?s_directory=Beverly+Acha"
## [3] "http://www.williams.edu/people/?s_directory=Colin+Adams"
## [4] "http://www.williams.edu/people/?s_directory=Zaid+Adhami"
## [5] "http://www.williams.edu/people/?s_directory=Sara+Mandelbaum"
## [6] "http://www.williams.edu/people/?s_directory=Jeannie+Albrecht"
```

Unfortunately, the search page only gives out limited information that consists of profession, department, office, phone number and e-mail address. It does not contain any links that we can follow, either. Therefore, we need to take an additional step and create a generic profile URL since we can extract Unix ID's from the e-mail address. We observed that the profile URL is in the format http://www.williams.edu/profile/unixid. Let us take Robert E. Baker-White, who is in the 18th row of our *names* vector, as an example.

```
academic_data[18]
```

```
## Robert E. Baker-White
##                    NA
```

```
unix_ids <-
  read_html(search_links[18]) %>%
  html_nodes(css = ".phone+ .email a") %>%
  html_attr("href") %>%
  gsub("mailto:|@williams.edu", "", .)
print(unix_ids)
```

```
## [1] "rbakerwh"
```

To select the CSS, we used SelectorGadget as Hadley Wickham also suggests in his package **rvest**. Next, we create profile URLs using the Unix IDs we have obtained and repeat the step above.

```
  profile_data <-
    paste("http://www.williams.edu/profile/", unix_ids, sep="") %>%
    read_html() %>%
    html_nodes(css = ".profile-education .profile-subsection") %>%
    html_text()
  print(profile_data)
```

```
## [1] "B.A. Williams College (1980)M.F.A. University of Washington, Theatre (1983)Ph.D. Stanford Unive
```

Similarly, we extract the desired data as we did in **4.1**, but with a slightly different pattern.

```
 pattern <- "(B\\.(.*?)\\.|A\\.B\\.|Diploma) .*? \\(\\d{4}\\)"
 academic_data[18] <- regmatches(profile_data, regexpr(pattern, profile_data))
   academic_data[18]
```

```
##          Robert E. Baker-White
## "B.A. Williams College (1980)"
```

Nevertheless, we need an if-statement to deal with missing people, i.e., postdoctoral fellows and some visiting lecturers who do not appear on the directory search. We can do this by simply defining a variable named *email_nodes* and wrapping around our code with the following.

```
  email_nodes <-
    read_html(search_links[k]) %>%
    html_nodes(css = ".phone+ .email a")

  # If the person is not under the directory, we can safely assume the length of the variable
  # we have defined to be zero.
  if (length(email_nodes) != 0){
}
```

There are also two conditions that can stop the iteration. In order to handle such exceptions without breaking our for-loop and to see where they occur, we need to use the function **tryCatch**.

**1.** There may be multiple people with the same first and last name, in which case we obtain two different Unix IDs and they are both stored in the same vector *unix_id*. However, the function *read_html* takes only a single value and if a vector containing multiple URLs is provided, an error arises. Running the code with the *tryCatch* function around it shows us that there are two such cases: 'Matthew Gibson' and 'Joel Lee'. In each case, there is a professor and a student with the same first and last name. Since faculty members are always shown above students and other staff, we could command it to take the first value assuming it is unlikely that two professors share the same name. Just to be safe, we add an if-statement to check manually if our assumption is true and remove the *tryCatch* function because it is redundant. To demonstrate:

```
  names[132]
```

```
## [1] "Matthew Gibson"
```

```
  unix_ids <-
    read_html(search_links[132]) %>%
    html_nodes(css = ".phone+ .email a") %>%
    html_attr("href") %>%
    gsub("mailto:|@williams.edu", "", .)
  if (length(unix_ids) > 1){
    print(cat("Warning: More than one people with the name", print(names[132]), "\n"))
  }
```

```
## [1] "Matthew Gibson"
## Warning: More than one people with the name Matthew Gibson
## NULL
```

```
  return(unix_ids[1])
```

```
## [1] "mg17"
```

**2.** There may be some faculty members who have web profile pages, but their undergraduate degree information is missing. With the exact method as above (i.e., by using *tryCatch*), we observe that there are 5 such cases:

- Nicole S. Desrosiers
- Wang Guowei
- Mamoru Hatakeyama
- Christophe A. Kone
- Julia L. Kowalski

```
  tryCatch({

  }, error=function(e){cat("Missing information on", print(names[i]), "\n")})
```

This section can be reproduced simply with the function *extract_data* as follows.

```
  academic_data <- extract_data(names, data_source, useInternet = TRUE)
```

For the sake of efficiency, I have stored the data we need in the folder */inst/extdata* in advance. Let us take a look at it:

```
academic_data <- dget(file = system.file("extdata", "academic_data1516.txt",
  package = "ager"))
  head(academic_data)
```

```
##                            Daniel P. Aalberts
##       "1989, BS, MA Institute of Technology,"
##                               Beverly D. Acha
##                                            NA
##                                Colin C. Adams
##       "1978, BS, MA Institute of Technology,"
##                                  Zaid Adhami
##             "B.A. Stanford University (2010)"
##                     Sara E. Adler Mandelbaum
## "B.A. Case Western Reserve University (2009)"
##                           Jeannie R Albrecht
##                "2001, BS, Gettysburg College,"
```

## 5. Transforming data

Having all the information we need, we can process our data and prepare a data frame. First, we need to convert academic degrees into a single format, which has dots inbetween capital letters.

```
by1 <- c("BA", "BS", "BM", "BFA", "BPhil", "AB", "BE", "A.B.")
by2 <- c("B.A.", "B.S.", "B.M.", "B.F.A.", "B.Phil.", "B.A.", "B.E.", "B.A.")
for (i in 1:8){
  academic_data <- sub(by1[i], by2[i], academic_data)
}
```

Next, we need to decompose the data into proper, relevant partitions.

```
  grad_years <- as.numeric(sub(".*(\\d{4}).*", "\\1", academic_data))
  degrees <- as.factor(sub(".*(B\\.(.*?)\\.|A\\.B\\.|Diploma).*", "\\1", academic_data))
```

Finally, we can form our data frame.

```
   faculty1516_df <- data.frame(Name = names(academic_data),
                                Graduation.Year = grad_years,
                                Degree = degrees,
                                Age = (2016 + 22 - grad_years),
                                Academic.Year = academic_year,
                                stringsAsFactors = FALSE,
                                row.names = NULL
   )
  head(faculty1516_df)
```

```
##                       Name Graduation.Year Degree Age Academic.Year
## 1       Daniel P. Aalberts            1989   B.S.  49       2015-16
## 2          Beverly D. Acha              NA   <NA>  NA       2015-16
## 3           Colin C. Adams            1978   B.S.  60       2015-16
## 4              Zaid Adhami            2010   B.A.  28       2015-16
## 5 Sara E. Adler Mandelbaum            2009   B.A.  29       2015-16
## 6       Jeannie R Albrecht            2001   B.S.  37       2015-16
```

```
  str(faculty1516_df)
```

```
## 'data.frame':    438 obs. of  5 variables:
##  $ Name           : chr  "Daniel P. Aalberts" "Beverly D. Acha" "Colin C. Adams" "Zaid Adhami" ...
##  $ Graduation.Year: num  1989 NA 1978 2010 2009 ...
##  $ Degree         : Factor w/ 6 levels "B.A.","B.F.",..: 5 NA 5 1 1 5 1 1 3 1 ...
##  $ Age            : num  49 NA 60 28 29 37 47 38 40 46 ...
##  $ Academic.Year  : chr  "2015-16" "2015-16" "2015-16" "2015-16" ...
```

This section can be reproduced with the function *transform_data* as follows.

```
  academic_year <- "2015-16"
  faculty1516_df <- transform_data(academic_data)
```

## 6. Loading data

The final step is to integrate data from different sources into a single data warehouse, which is pretty trivial. First we need data frames from other years. Let us see the necessary steps for one of them and simply load the other one from */inst/extdata*.

```
academic_year <- "2013-14"
flat_file <- scrub_catalog(academic_year)
data_source <- scrub_catalog("2013-14")
names <- collect_names(flat_file, reformat = FALSE)
academic_data <- extract_data(names, data_source, useInternet = FALSE)
faculty1314_df <- transform_data(academic_data)
head(faculty1314_df)
```

```
##                   Name Graduation.Year Degree Age Academic.Year
## 1  Daniel P. Aalberts            1989   B.S.  49       2013-14
## 2           Sayaka Abe            1997   B.A.  41       2013-14
## 3       Colin C. Adams            1978   B.S.  60       2013-14
## 4  Jeannie R Albrecht            2001   B.S.  37       2013-14
## 5           Laylah Ali            1991   B.A.  47       2013-14
## 6 Kristopher D. Allen            1998   B.M.  40       2013-14
```

```
academic_year <- "2014-15"
academic_data <- dget(file = system.file("extdata", "academic_data1415.txt",
package = "ager"))
faculty1415_df <- transform_data(academic_data)
head(faculty1415_df)
```

```
##                  Name Graduation.Year Degree Age Academic.Year
## 1 Daniel P. Aalberts            1989   B.S.  49       2014-15
## 2     Colin C. Adams            1978   B.S.  60       2014-15
## 3        Zaid Adhami            2010   B.A.  28       2014-15
## 4 Jeannie R Albrecht            2001   B.S.  37       2014-15
## 5         Laylah Ali            1991   B.A.  47       2014-15
## 6         Kris Allen            1998   B.M.  40       2014-15
```

Now that we have 3 data frames, we can bind them together and create some plots. Note that this does not merge multiple rows with the same *Name* into one, because they have different *Academic.Year* values, which is exactly what we want.

```
data_warehouse_df <- rbind(faculty1516_df,
                           faculty1415_df,
                           faculty1314_df)
str(data_warehouse_df)
```

```
## 'data.frame':    1277 obs. of  5 variables:
##  $ Name           : chr  "Daniel P. Aalberts" "Beverly D. Acha" "Colin C. Adams" "Zaid Adhami" ...
##  $ Graduation.Year: num  1989 NA 1978 2010 2009 ...
##  $ Degree         : Factor w/ 7 levels "B.A.","B.F.",..: 5 NA 5 1 1 5 1 1 3 1 ...
##  $ Age            : num  49 NA 60 28 29 37 47 38 40 46 ...
##  $ Academic.Year  : chr  "2015-16" "2015-16" "2015-16" "2015-16" ...
```

## Results

```
# Mean
aggregate(Age ~ Academic.Year, data_warehouse_df, function(x){
  round(mean(x, na.rm = TRUE))
})
```

```
##   Academic.Year Age
## 1       2013-14  51
## 2       2014-15  51
## 3       2015-16  50
```

```
# Range
aggregate(Age ~ Academic.Year, data_warehouse_df, range, na.rm = TRUE)
```

```
##   Academic.Year Age.1 Age.2
## 1       2013-14    27    80
## 2       2014-15    28    80
## 3       2015-16    28    80
```

```
# Number of college faculty members
aggregate(Name ~ Academic.Year, data_warehouse_df, length)
```

```
##   Academic.Year Name
## 1       2013-14  418
## 2       2014-15  421
## 3       2015-16  438
```

```
# Youngest faculty member(s)
min <- aggregate(Age ~ Academic.Year, data_warehouse_df, min, na.rm = TRUE)
data_warehouse_df[which(data_warehouse_df$Age == min[ ,2]),
                  c("Academic.Year", "Name", "Age")]
```

```
##      Academic.Year               Name Age
## 174        2015-16  Scott D. Honecker  28
## 236        2015-16      Kelsey Levine  28
## 441        2014-15        Zaid Adhami  28
## 602        2014-15  Scott D. Honecker  28
## 1123       2013-14 Sarah A. Mirseyedi  27
```

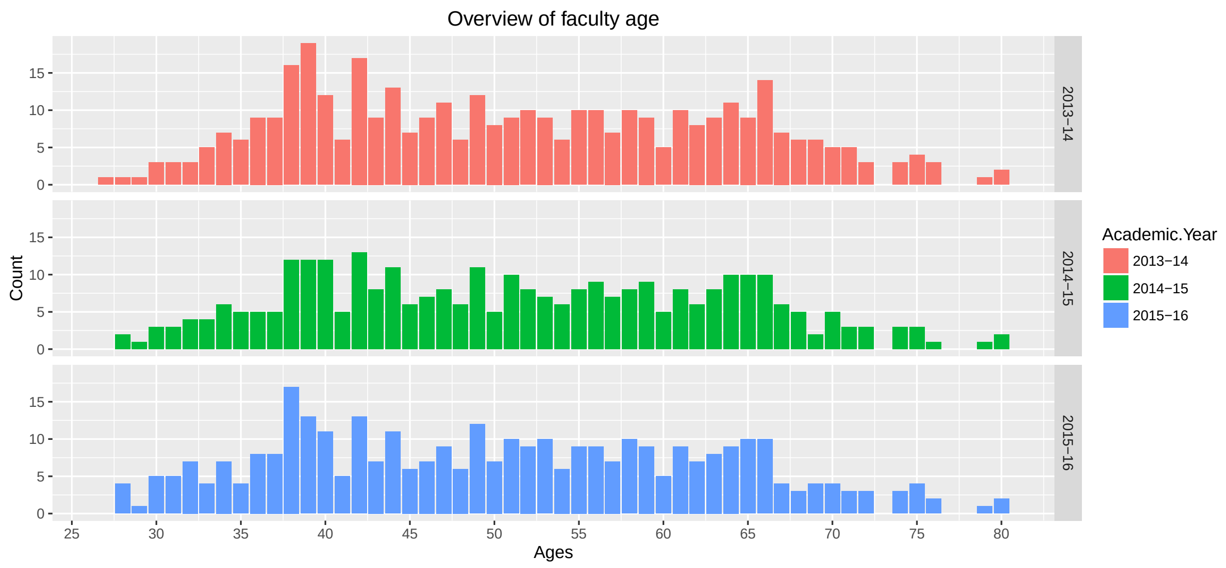
```
# Oldest faculty member(s)
max <- aggregate(Age ~ Academic.Year, data_warehouse_df, max, na.rm = TRUE)
data_warehouse_df[which(data_warehouse_df$Age == max[ ,2]),
                  c("Academic.Year", "Name", "Age")]
```

```
##     Academic.Year               Name Age
## 28        2015-16 Donald deB. Beaver  80
## 94        2015-16     Charles B. Dew  80
## 464       2014-15 Donald deB. Beaver  80
## 524       2014-15     Charles B. Dew  80
## 879       2013-14 Donald deB. Beaver  80
## 948       2013-14     Charles B. Dew  80
```

Finally, here are some demonstrations that represent the data in various ways.

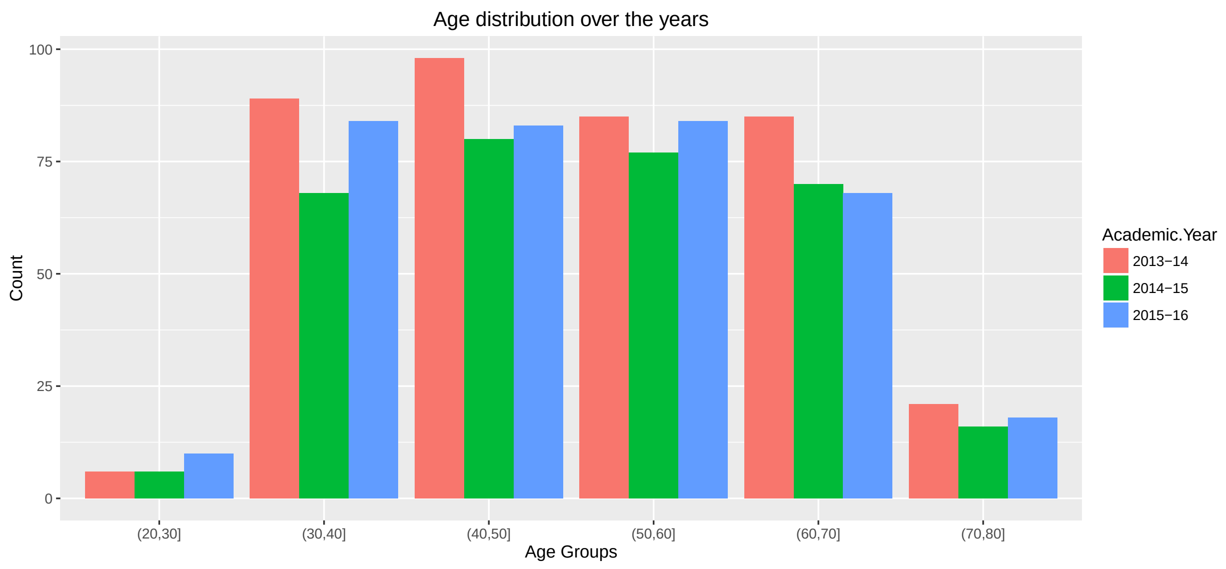
```
  # First get rid of missing data
  data_warehouse_df <- na.omit(data_warehouse_df)

  ggplot(data_warehouse_df) +
    aes(x = Age, fill = Academic.Year) +
    facet_grid(Academic.Year~.) +
    geom_bar() +
    scale_x_continuous(breaks = seq(25, 85, 5), labels = seq(25, 85, 5)) +
    xlab("Ages") + ylab("Count") +
    ggtitle("Overview of faculty age")
```

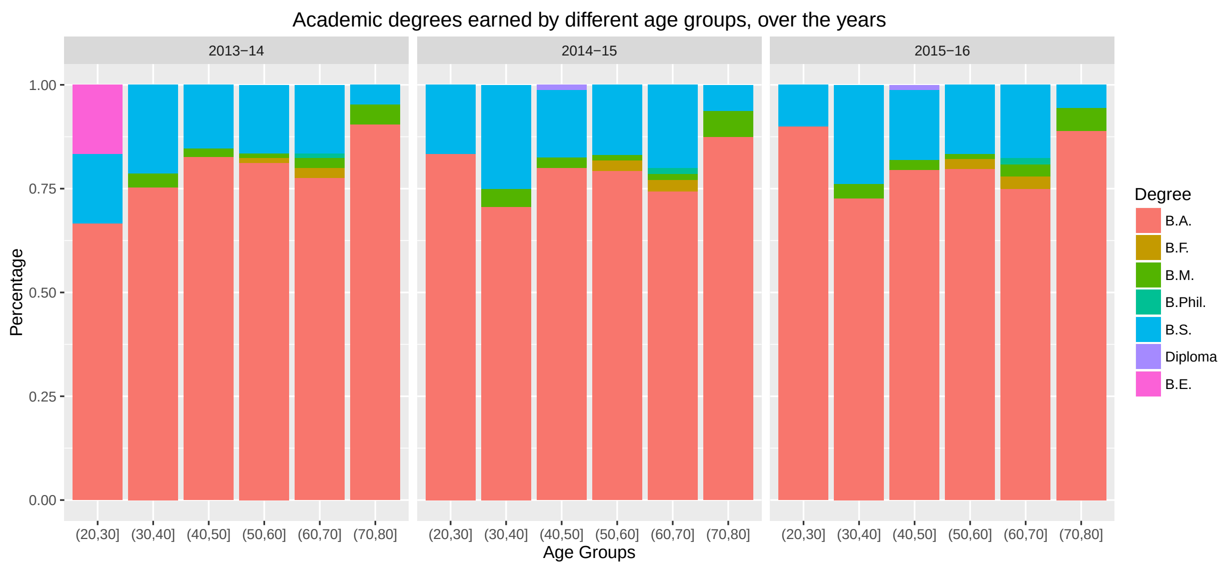


```
# Create age groups
data_warehouse_df <- mutate(data_warehouse_df,
                            Age.Group = cut(Age, breaks = seq(20, 80, by = 10)))

ggplot(data_warehouse_df) +
    aes(x = Age.Group, fill = Academic.Year) +
    geom_bar(position = "dodge") + xlab("Age Groups") + ylab("Count") +
    ggtitle("Age distribution over the years")
```

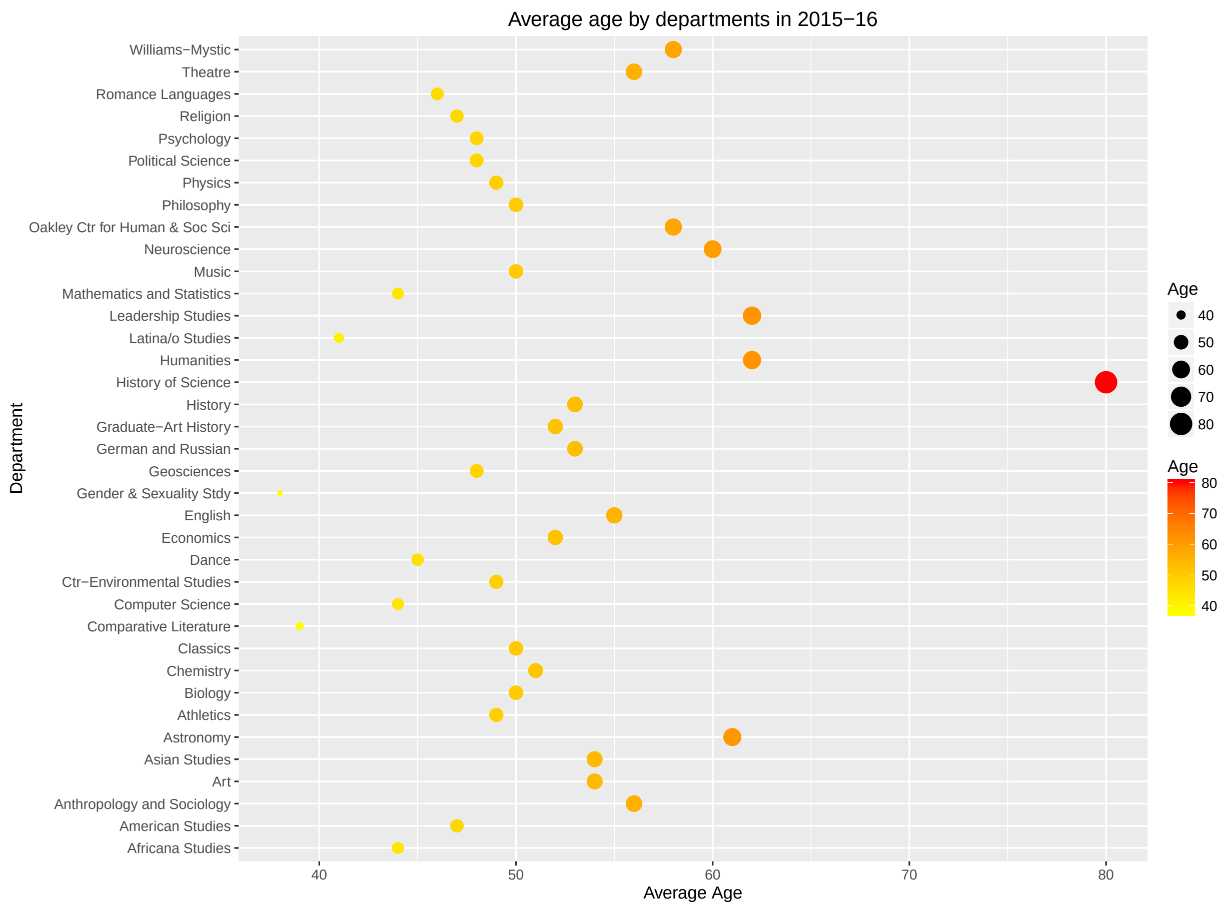


```
ggplot(data_warehouse_df) +
    aes(x = Age.Group, fill = Degree) + facet_grid(. ~ Academic.Year) +
    geom_bar(position = "fill") + xlab("Age Groups") + ylab("Percentage") +
    ggtitle("Academic degrees earned by different age groups, over the years")
```



```
# Create department column
faculty1516_df$Department <- departments
department_average_df <- aggregate(Age~Department, faculty1516_df, function(x){
  round(mean(x, na.rm = TRUE))
})

ggplot(department_average_df) +
    aes(x = Age, y = Department) +
    geom_point(aes(color = Age, size = Age)) +
    scale_color_gradient(low = "yellow", high = "red") +
    xlab("Average Age") +
    ggtitle("Average age by departments in 2015-16")
```


Average age by departments in 2015−16
Williams−Mystic
Theatre
Romance Languages
Religion
Psychology
Political Science
Physics
Philosophy
Oakley Ctr for Human & Soc Sci
Neuroscience
Music
Mathematics and Statistics
Leadership Studies
Latina/o Studies
Humanities
History of Science
History
Graduate−Art History
German and Russian
Geosciences
Gender & Sexuality Stdy
English
Economics
Dance
Ctr−Environmental Studies
Computer Science
Comparative Literature
Classics
Chemistry
Biology
Athletics
Astronomy
Asian Studies
Art
Anthropology and Sociology
American Studies
Africana Studies
Department
Average Age
Age
Age