

Data Cleaning Tutorials

Eilif Mikkelsen

Contents

1	Introduction	1
2	What is Presumed About the Reader	1
3	Setup	2
4	Topics Covered	2
5	Initial Look	2
5.1	Commands Used	2
6	Standardizing Text (string) Columns	4
6.1	Commands Used:	4
6.2	drinks_type: A Demonstration in Data Context	6
7	DateTime Parsing and Computation	7
7.1	Commands Used	7
7.2	Packages Used	7
7.3	A Brief Introduction to Why Data Types Matter	7
7.4	Setting data type for collection_date	8
7.5	A Brief Aside on Regular Expressions (Regex)	9
7.6	Using Regex to Sort Out age	9
8	Including Plots	17

1 Introduction

In an academic setting data sets are often presented as clean and orderly sets of information. For most real-world applications this couldn't be farther from the truth. When talking to data scientists across industries it is commonly noted that the vast majority of their time is spent cleaning and structuring data before they can even consider models.

In this tutorial we will cover some core data cleaning methods. This is by no means exhaustive however it should leave the reader with a strong set of data manipulation skills. For context, I wrote this document to hone my data manipulation skills in R after spending years working with the Python Pandas data library which provides data structures and features similar to the R `data.frame` object. Periodically, comments detailing the `pandas` equivalents will be included looking something like `# Py/Pandas: df.head()`.

This document was originally written in R Markdown, a format for generating mixed format documents combining code and text. Where it seems useful, I will include links to relevant packages or functions. Here is a link to a cheat sheet on the R Markdown syntax.

2 What is Presumed About the Reader

- The reader has a very basic understanding of simple programming concepts.

- The reader is has cursory knowledge of the `data.frame` tabular data structure and its methods.
- The reader remembers that arrays in R are 1-indexed rather than 0-indexed.

I will use `=` rather than `<-` out of habit from my work in Python. So far as I can tell, they are equivalent.

3 Setup

Here is the output information on the operating system and R version used to generate this tutorial. The tools used in this tutorial are elemental functions of R as such is it not expected that this tutorial will become out-of-date. If you find issues, please feel free to edit this file and submit a pull request to this repository. Where specific packages are required, additional notes will be made.

```
version

##
## platform      x86_64-apple-darwin13.4.0
## arch          x86_64
## os            darwin13.4.0
## system        x86_64, darwin13.4.0
## status
## major         3
## minor         3.3
## year          2017
## month         03
## day           06
## svn rev       72310
## language      R
## version.string R version 3.3.3 (2017-03-06)
## nickname      Another Canoe
```

4 Topics Covered

- Cleaning string categories
- Context aware string categorization
- Context aware unit standardization
- Datetime parsing

5 Initial Look

I once had a professor that said that no matter how good and experienced and smart we are, there is no substitute for viewing and plotting your data!

Let's start by loading and looking at the first few rows of the data. The `ix` column is an arbitrary row number that was created along with the dataset. We will use this to uniquely identify rows. Additionally, we want all string columns to be treated as characters rather than "factors", a special R data type.

5.1 Commands Used

- `hist`
- `read.csv`

- row.names
- head
- nrow
- ncol

```
# Loading the CSV
# Py/Pandas: pd.read_csv('./horrible_data.csv', index=0)
in_data = read.csv('./horrible_data.csv', stringsAsFactors=FALSE)

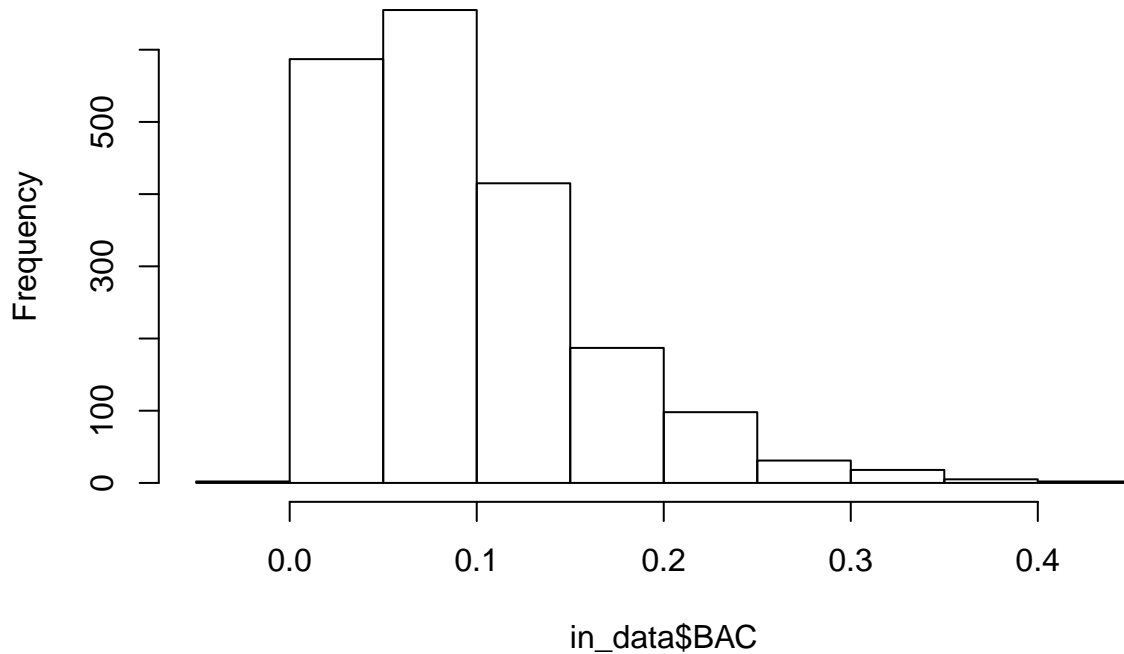
# Let's duplicate the data so we can keep a copy of the original. When we update data we will update wrk_data
wrk_data = cbind(in_data)
# Setting the row names to the `ix` column
row.names(in_data) = in_data$ix

# Py/Pandas: df.head()
head(in_data)
```

```
##   ix      BAC    age      collection_date      drink_type
## 0  0 0.05725438 1/7/66 2018-01-29T12:21:46.232643      Merlot
## 1  1 0.15269702 8/19/68 2018-01-29T12:21:46.233977 Franziskaner Weissbier
## 2  2 0.05564855    50 2018-01-29T12:21:46.234572      Wine
## 3  3 0.03288376   671 2018-01-29T12:21:46.235095      Wine
## 4  4 0.03005566 4/2/60 2018-01-29T12:21:46.235582      Vin
## 5  5 0.02840692 9/14/74 2018-01-29T12:21:46.236121      whiskey
##   num_drinks    sex units volume_consumed  weight
## 0         2.4      M      12.0000 177.8200
## 1         4.6      F metric      1634.5534 237.3310
## 2         2.5  Male      12.5000 185.4400
## 3         0.9 female metric      134.6799 265.0237
## 4         1.7  meal      SI      253.6016 315.3754
## 5         0.6 Female metric      28.0873 225.4100
```

```
# Let's use the super handy built-in to generate a histogram
hist(in_data$BAC)
```

Histogram of in_data\$BAC



```
# Let's also look at the overall shape of the dataset.
```

```
nrow(in_data)
```

```
## [1] 2000
```

```
ncol(in_data)
```

```
## [1] 10
```

6 Standardizing Text (string) Columns

When faced with text variables, figuring out how to group and organize the values into usable data can be extremely time consuming and difficult.

Immediately upon observing the first 6 rows of the data we see that there are several variations of the same category for both the **units** and **sex** column. The goal is to identify what values in the column correspond to what categories. The **drink_type** column also needs to be cleaned in a similar manner but it requires additional discussion.

I used this review of R subsetting methods to brush up for this tutorial. Be sure to also remember that a dataframe has two dimensions, rows and columns. Let **rdf** be an R data frame. If we wish to select the first column for all observations (rows) we would call **rdf[, 1]** as dataframe selecting follows [row, column] convention.

6.1 Commands Used:

- **unique**
- **which**
- **c**

Let's print out the list of unique items in the sex column

```
sex_vals = unique(in_data$sex)
sex_vals
```

```
## [1] "M"      "F"      "Male"   "female" "meal"   "Female" "femeal" ""
## [9] "male"
```

Conditional indexing allows for the selection of rows based on matching column conditions. For those readers familiar with general programming methods, this is essentially the creation of a binary mask. Here we will look for the rows where the `sex` column equals "Male". We can make these masks as elaborate as required by combining boolean criteria with `&` and `|`.

```
# Selecting rows where
head(in_data[which(in_data$sex == "Male"), ])
```

```
##      ix      BAC age      collection_date      drink_type
## 2    2 0.05564855 50 2018-01-29T12:21:46.234572      Wine
## 13   13 0.09123496 746 2018-01-29T12:21:46.237715 Franziskaner Weissbier
## 20   20 0.04443760 53 2018-01-29T12:21:46.238809      Scotch
## 22   22 0.01751559 774 2018-01-29T12:21:46.239154      Warsteiner
## 24   24 0.06256844 584 2018-01-29T12:21:46.239470      IPA
## 41   41 0.09546701 556 2018-01-29T12:21:46.242135      Merlot
##      num_drinks sex units volume_consumed weight
## 2              2.5 Male              12.5000 185.4400
## 13             3.6 Male              43.2000 171.7100
## 20             1.4 Male              2.1000 167.3100
## 22             0.4 Male metric      140.5923 280.8853
## 24             3.1 Male              37.2000 231.9700
## 41             3.3 Male metric      488.3116 238.5470
```

What is the `which` command doing? It is applying the boolean filters provided and returning the row numbers/names of the rows for which the filters return TRUE. If we print out the results of the `which` command we can see this.

```
head(which(in_data$sex == "Male"))
```

```
## [1] 3 14 21 23 25 42
```

We see from the output that this dataset has two correct options for `sex`, Male and Female. We also observe that the dataset contains a variety of abbreviations and typos for the two categories. To standardize the values such that all observations that should be recorded as Male are updated accordingly. We will perform the same operation for Female.

To accomplish this task, we will need to use the `%in%` operator. The `%in%` operator checks if a value is in the supplied set. In a pseudocode example let `s = {1, 3, 6}`. The evaluation of `1 in s` returns TRUE while `2 in s` returns FALSE.

```
# A vector of all the values that should be classified as "Male"
male_values = c("Male", "meal", "male", "M")
```

```
# Now we will use the `%in%` operator to select all rows where the `sex` column value is in the set of
# possible values.
head(in_data[which(in_data$sex %in% male_values), c("BAC", "sex")], n = 10)
```

```
##      BAC sex
## 0 0.057254381 M
## 2 0.055648547 Male
## 4 0.030055655 meal
```

```
## 9 0.055791317 meal
## 13 0.091234964 Male
## 14 0.043404374 male
## 15 0.005654698 M
## 18 0.075756832 M
## 20 0.044437597 Male
## 21 0.017537210 M
```

Now that we know how to select the rows that need to be updates, we can update wrk_data directly. We can set a column of the filtered dataframe to a column and only the selected rows will be updates.

```
wrk_data[which(wrk_data$sex %in% male_values), "sex"] = "Male"
```

```
# Let's do the same thing for "Female"
```

```
female_values = c("F", "female", "Female", "femeal")
```

```
wrk_data[which(wrk_data$sex %in% female_values), "sex"] = "Female"
```

```
# SUCCESS! If we examine the unique set of values for `sex` in wrk_data we see that the only option for
unique(wrk_data$sex)
```

```
## [1] "Male" "Female" ""
```

Alright! We have learned how to update rows/columns in a dataframe using boolean masking. The units column has similar issues. I won't go through the explanation again but the steps are the same as those for sex above.

```
# Let's print out the list of unique items in the units column
```

```
unique(in_data$units)
```

```
## [1] "" "metric" "SI"
```

```
metric_values = c("metric", "SI")
```

```
wrk_data[which(wrk_data$units %in% metric_values), "units"] = "metric"
```

```
# It's bad practice to have the absense of information imply a definitive value.
```

```
# I know (because I made the dataset) that all values that are not marked as metric
```

```
# use the America/imperial units.
```

```
wrk_data[which(!(wrk_data$units %in% metric_values)), "units"] = "imperial"
```

```
# Let's check our work.
```

```
unique(wrk_data$units)
```

```
## [1] "imperial" "metric"
```

6.2 drinks_type: A Demonstration in Data Context

Before I go into what the values for this column mean, let's start by just looking at them. What do you notice?

```
unique(wrk_data$drink_type)
```

```
## [1] "Merlot" "Franziskaner Weissbier"
## [3] "Wine" "Vin"
## [5] "whiskey" "Warsteiner"
## [7] "shots" "Vino"
## [9] "Scotch" "IPA"
## [11] "Beer" "vidka"
```

```
## [13] "Bud Light"          "Cab"
```

What we have here is known in the industry as a “hot mess”. If you didn’t guess, the values are a mix of alcoholic beverage brands and types in a variety of languages. When encountering this sort of data, it’s often worth taking the 20 mins. to go on Google each item. I am not well read in the boozes of the world. A quick Google search reveals that “Warsteiner” and “Franziskaner Weissbier” are beer. Once we know what everything is we can generate our list of categories and assign the options to a specific category as we did with `sex` and `units`. The resulting code would look something like this.

```
# Vectors for each type of alcoholic beverage.
wine_options = c("Merlot", "Vino", "Vin", "Cab")
beer_options = c("Bud Light", "IPA", "Beer", "Franziskaner Weissbier", "Warsteiner")
hard_liquor_options = c("shots", "Scotch", "whiskey", "vidka")

wrk_data[which(wrk_data$drink_type %in% wine_options), "drink_type"] = "Wine"

wrk_data[which(wrk_data$drink_type %in% beer_options), "drink_type"] = "Beer"

wrk_data[which(wrk_data$drink_type %in% hard_liquor_options), "drink_type"] = "Hard Liquor"

unique(wrk_data$drink_type)

## [1] "Wine"          "Beer"          "Hard Liquor"
```

7 DateTime Parsing and Computation

In this section we will clean up the `collection_date` and `age` columns, discuss the importance of correct data typing, and demonstrate datetime operations in R.

7.1 Commands Used

- `anytime`
- `typeof`
- `as.POSIXct`
- `as.Date`

7.2 Packages Used

- `anytime`. This can be installed from your favorite R session using `install.packages("anytime")`

7.3 A Brief Introduction to Why Data Types Matter

Every column in the loaded dataframe `in_data` has a data type. Without going too far into what this means know that the type of a column influences how R, and the code we write for it, process the data. Imagine we have the data value 01776. This set of digits/characters could mean a variety of completely unrelated things. For example 01776 could be:

- The string Zip Code for Sudbury, Massachusetts
- The year in which the Declaration of Independence of the United States of America was signed but for some reason the number was padded with a zero.
- An arbitrary integer measurement where the padded zero was included to indicate that the maximum value is 99999.

If this value was a Zip Code then we must tell R that under no circumstance should the numerical value be considered. The string 01778 is a code of characters, not a number. If the value was a year, we may consider forcing R to treat it as a string, date, or integer depending on the application. If the value was a measurement, we would want it to be considered a float or an integer.

Before we can perform operations on entire columns we must first unify the data types within the column.

There is a lot more than can be said on data types and “type safety” in programming. If you’re interested in reading more about this be prepared for a fierce internet debate between programming communities.

Back to the examples at hand...

7.4 Setting data type for `collection_date`

In this example dataset we have two columns that need additional attention to their types, `collection_date`, the data of data collection, and `age`, the age of the participant at the time of the study. If we examine the `collection_date` column we can observe that the values are “string encoded dates” or a representation of a date and time using a standard string format. Here, the dates are represented using the ISO 8601 date format. We will use the `anytime` package which is excellent for automatically parsing standard date formats into the datetime datatypes in R. We will also be using `lubridate` and a custom function.

```
# Load the anytime library
library(anytime)

# Examine the type of the collection_date column. Note that R currently classifies it as "character"
typeof(in_data$collection_date)

## [1] "character"

# Use anytime to convert the column to POSIXct times.
wrk_data[, 'collection_date'] = anydate(in_data$collection_date)

# Note that the type has changed to "double", don't worry, this is really a datetime.
typeof(wrk_data$collection_date)

## [1] "double"

wrk_data$collection_date[1]

## [1] "2018-01-29"
```

Why does this matter? Supposed I wanted to compute the time between a value in the column and some other date. I could try to type out the dates as strings and pray that the compute knows what I mean. This won’t work because the subtraction of two string types has no idea what you intended to do! You could also transform the dates from your colloquial format to some relative time unit like “fortnights since 1/1/2000” but this will quickly become cumbersome and error prone. By having two values that are typed as datetime, R knows about all the leap years, time zones, and other quirks and is able to correctly perform time operations without any effort from the user.

```
# Let's just type out the dates and try to subtract them.... ERROR
# "2011-03-25" - "2001-03-25"

# Now let's try a computation of two datetime objects.
as.Date("2018-03-19") - wrk_data$collection_date[1]

## Time difference of 49 days
```

Tada! Using datetime objects gives a correct and useful result.

7.5 A Brief Aside on Regular Expressions (Regex)

7.6 Using Regex to Sort Out age

Looking at the first 10 values in the `age` column we can see immediately that the type is mixed. Some of the values are the age in **months** while others are the birthday. Yikes!

```
head(in_data$age, n = 10)
```

```
## [1] "1/7/66" "8/19/68" "50" "671" "4/2/60" "9/14/74" "6/21/59"
## [8] "389" "549" "35"
```

To select the rows where the `age` column contains a birthdate rather than an age, we need to match the values that follow the pattern MM/DD/YYYY. In regex for R, this is `'^(\d{2}|\d{2})\d{2}/(\d{1}|\d{2})\d{2}/(\d{2})'`.

For these examples we will be exploring the features of the `lubridate` package.

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
## The following object is masked from 'package:base':
##
## date
# If we run grep along the column, it returns the row number of the matching columns
grep('^(\d{1}|\d{2})\d{2}/(\d{1}|\d{2})\d{2}/(\d{2})', in_data$age, perl = TRUE)
```

```
## [1] 1 2 5 6 7 12 13 20 41 48 49 56 63 65
## [15] 67 68 70 71 72 80 81 89 92 94 95 96 97 109
## [29] 116 121 134 150 158 163 164 166 173 174 176 178 180 182
## [43] 187 195 200 209 210 214 216 217 220 222 225 239 240 249
## [57] 254 258 261 266 269 272 273 276 285 294 295 296 309 314
## [71] 317 318 321 332 333 336 338 339 346 347 352 354 360 365
## [85] 367 396 397 398 399 403 408 413 416 420 421 423 429 431
## [99] 437 438 440 441 443 448 453 454 455 457 464 469 470 472
## [113] 479 488 492 493 495 508 511 517 519 520 528 545 551 559
## [127] 561 572 573 576 577 578 582 584 585 591 592 595 596 599
## [141] 600 610 615 620 621 626 627 630 638 639 641 643 650 651
## [155] 653 657 670 675 689 694 696 699 712 716 718 723 730 732
## [169] 735 740 749 751 756 758 761 766 767 768 772 787 789 791
## [183] 796 804 805 812 815 817 820 824 826 831 834 835 837 841
## [197] 844 848 853 860 865 871 875 889 901 913 916 924 928 932
## [211] 937 940 942 945 946 948 949 951 952 953 956 959 964 965
## [225] 969 971 974 986 987 989 991 993 995 999 1009 1010 1033 1038
## [239] 1039 1043 1044 1045 1048 1055 1056 1057 1059 1065 1069 1074 1075 1080
## [253] 1083 1085 1088 1095 1098 1100 1101 1102 1103 1112 1114 1117 1118 1124
## [267] 1125 1126 1127 1136 1139 1144 1148 1150 1151 1156 1157 1162 1163 1165
## [281] 1169 1175 1176 1177 1184 1189 1191 1195 1196 1197 1208 1212 1219 1226
## [295] 1229 1234 1245 1248 1252 1255 1259 1261 1264 1272 1273 1275 1276 1278
## [309] 1281 1284 1293 1297 1306 1307 1312 1314 1315 1317 1319 1320 1326 1328
## [323] 1331 1334 1337 1343 1345 1346 1348 1351 1352 1364 1368 1374 1375 1378
## [337] 1379 1387 1393 1394 1395 1397 1402 1406 1407 1411 1415 1420 1422 1427
## [351] 1429 1433 1445 1453 1458 1468 1469 1471 1473 1475 1486 1487 1495 1497
## [365] 1499 1500 1505 1513 1515 1517 1518 1525 1535 1541 1547 1552 1557 1559
## [379] 1560 1562 1564 1571 1572 1576 1580 1588 1590 1591 1592 1606 1607 1608
```

```
## [393] 1615 1616 1619 1620 1621 1628 1639 1642 1653 1654 1660 1665 1669 1673
## [407] 1674 1676 1680 1682 1685 1693 1699 1700 1707 1708 1710 1719 1722 1723
## [421] 1725 1729 1732 1733 1736 1748 1750 1757 1758 1760 1769 1773 1778 1781
## [435] 1783 1785 1786 1790 1793 1794 1797 1800 1805 1814 1819 1820 1822 1823
## [449] 1824 1825 1828 1831 1833 1842 1844 1845 1847 1856 1864 1867 1868 1869
## [463] 1871 1876 1880 1886 1887 1890 1894 1895 1901 1902 1912 1913 1914 1923
## [477] 1928 1929 1930 1931 1933 1936 1940 1945 1948 1951 1960 1961 1970 1974
## [491] 1975 1980 1982 1987 1994 1995
```

```
in_data$age
```

```
##      [1] "1/7/66"      "8/19/68"      "50"           "671"          "4/2/60"       "9/14/74"
##      [7] "6/21/59"     "389"          "549"          "35"           "336"          "5/31/82"
##     [13] "8/5/62"      "746"          "633"          "391"          "351"          "429"
##     [19] "49"          "1/7/54"       "53"           "388"          "774"          "709"
##     [25] "584"         "709"          "531"          "460"          "489"          "46"
##     [31] "24"          "50"           "260"          "23"           "650"          "61"
##     [37] "26"          "56"           "660"          "743"          "5/1/55"       "556"
##     [43] "592"         "43"           "59"           "632"          "272"          "1/13/62"
##     [49] "1/3/61"      "498"          "567"          "61"           "323"          "625"
##     [55] "51"          "3/6/87"       "556"          "431"          "741"          "480"
##     [61] "618"         "31"           "10/3/63"      "388"          "3/22/73"      "28"
##     [67] "12/26/73"    "8/31/84"      "45"           "6/19/53"      "12/8/91"      "11/14/68"
##     [73] "53"          "443"          "34"           "737"          "628"          "51"
##     [79] "449"         "6/30/92"      "5/7/92"       "643"          "499"          "631"
##     [85] "568"         "269"          "31"           "542"          "11/11/75"     "375"
##     [91] "33"          "10/14/58"     "696"          "8/20/57"      "1/15/65"      "1/6/57"
##     [97] "2/6/85"      "36"           "29"           "614"          "265"          "684"
##    [103] "675"         "38"           "717"          "395"          "44"           "605"
##    [109] "12/5/88"     "612"          "58"           "666"          "40"           "55"
##    [115] "367"         "3/23/83"      "596"          "29"           "560"          "366"
##    [121] "11/20/96"    "45"           "346"          "329"          "336"          "255"
##    [127] "276"         "591"          "26"           "600"          "21"           "544"
##    [133] "33"          "2/6/74"       "374"          "413"          "588"          "482"
##    [139] "51"          "41"           "730"          "28"           "32"           "566"
##    [145] "55"          "52"           "631"          "645"          "31"           "10/8/85"
##    [151] "546"         "275"          "59"           "34"           "381"          "34"
##    [157] "303"         "7/12/71"      "392"          "346"          "633"          "43"
##    [163] "3/3/62"      "6/4/61"       "473"          "2/21/72"      "406"          "334"
##    [169] "290"         "307"          "715"          "721"          "12/6/87"      "1/27/67"
##    [175] "338"         "2/29/88"      "283"          "5/13/91"      "630"          "7/27/73"
##    [181] "43"          "8/5/61"       "761"          "395"          "664"          "32"
##    [187] "4/7/71"      "47"           "293"          "756"          "426"          "742"
##    [193] "570"         "633"          "6/24/82"      "341"          "662"          "55"
##    [199] "525"         "11/30/60"     "37"           "628"          "383"          "415"
##    [205] "639"         "49"           "726"          "606"          "9/18/59"      "9/23/81"
##    [211] "483"         "60"           "612"          "1/25/61"      "402"          "12/14/91"
##    [217] "8/30/57"     "33"           "743"          "11/15/81"     "35"           "7/30/54"
##    [223] "545"         "463"          "12/19/65"     "51"           "688"          "55"
##    [229] "394"         "34"           "334"          "742"          "754"          "21"
##    [235] "432"         "592"          "601"          "379"          "7/5/82"       "1/22/80"
##    [241] "58"          "543"          "57"           "22"           "49"           "41"
##    [247] "772"         "432"          "10/19/74"     "353"          "599"          "596"
##    [253] "57"          "4/12/89"      "536"          "519"          "31"           "2/17/68"
##    [259] "522"         "474"          "7/18/92"      "59"           "43"           "312"
```

##	[265]	"600"	"10/5/76"	"526"	"737"	"4/21/64"	"425"
##	[271]	"470"	"12/30/57"	"8/8/71"	"438"	"261"	"10/2/77"
##	[277]	"384"	"670"	"54"	"462"	"277"	"575"
##	[283]	"24"	"387"	"2/1/96"	"23"	"433"	"675"
##	[289]	"541"	"64"	"368"	"436"	"683"	"12/23/57"
##	[295]	"9/11/58"	"9/6/63"	"446"	"767"	"64"	"532"
##	[301]	"45"	"618"	"262"	"406"	"564"	"280"
##	[307]	"26"	"39"	"6/26/68"	"254"	"464"	"679"
##	[313]	"692"	"4/22/95"	"400"	"21"	"1/27/64"	"3/5/91"
##	[319]	"23"	"754"	"6/10/81"	"737"	"46"	"56"
##	[325]	"429"	"63"	"38"	"590"	"49"	"60"
##	[331]	"409"	"8/15/78"	"11/21/58"	"643"	"271"	"5/29/87"
##	[337]	"22"	"1/3/88"	"11/13/75"	"632"	"593"	"40"
##	[343]	"602"	"727"	"649"	"11/21/88"	"10/21/73"	"27"
##	[349]	"457"	"649"	"63"	"11/7/72"	"270"	"3/2/56"
##	[355]	"26"	"302"	"61"	"54"	"575"	"6/21/77"
##	[361]	"559"	"483"	"55"	"37"	"1/22/91"	"52"
##	[367]	"5/23/95"	"685"	"39"	"57"	"338"	"361"
##	[373]	"29"	"698"	"53"	"27"	"583"	"61"
##	[379]	"56"	"580"	"387"	"314"	"61"	"661"
##	[385]	"308"	"624"	"340"	"642"	"49"	"698"
##	[391]	"59"	"316"	"437"	"760"	"429"	"11/18/71"
##	[397]	"6/21/84"	"5/30/90"	"6/30/86"	"473"	"653"	"369"
##	[403]	"6/14/87"	"573"	"705"	"59"	"464"	"10/4/71"
##	[409]	"607"	"521"	"674"	"463"	"9/14/67"	"270"
##	[415]	"47"	"11/30/62"	"470"	"294"	"306"	"8/1/91"
##	[421]	"9/15/69"	"294"	"9/15/59"	"40"	"532"	"748"
##	[427]	"694"	"473"	"9/25/53"	"41"	"11/3/69"	"39"
##	[433]	"408"	"44"	"265"	"33"	"5/22/55"	"4/19/82"
##	[439]	"54"	"7/24/77"	"7/26/64"	"611"	"9/4/89"	"45"
##	[445]	"432"	"523"	"36"	"2/25/55"	"32"	"29"
##	[451]	"763"	"748"	"1/31/79"	"12/11/56"	"5/2/86"	"21"
##	[457]	"11/30/54"	"459"	"63"	"354"	"434"	"656"
##	[463]	"30"	"11/26/95"	"44"	"629"	"322"	"61"
##	[469]	"12/13/94"	"12/30/66"	"27"	"9/28/92"	"50"	"36"
##	[475]	"54"	"476"	"259"	"573"	"6/6/66"	"399"
##	[481]	"34"	"412"	"518"	"343"	"665"	"442"
##	[487]	"348"	"9/15/56"	"411"	"294"	"282"	"4/16/62"
##	[493]	"10/6/54"	"294"	"3/4/92"	"309"	"472"	"62"
##	[499]	"370"	"770"	"47"	"686"	"653"	"309"
##	[505]	"564"	"455"	"610"	"9/7/71"	"41"	"725"
##	[511]	"9/26/58"	"254"	"37"	"653"	"61"	"634"
##	[517]	"10/31/81"	"271"	"2/24/60"	"10/21/62"	"596"	"351"
##	[523]	"563"	"328"	"638"	"28"	"43"	"10/13/90"
##	[529]	"750"	"33"	"526"	"486"	"510"	"734"
##	[535]	"38"	"533"	"325"	"23"	"26"	"30"
##	[541]	"480"	"659"	"348"	"29"	"7/1/94"	"36"
##	[547]	"53"	"55"	"628"	"285"	"11/26/82"	"720"
##	[553]	"772"	"399"	"433"	"41"	"735"	"700"
##	[559]	"9/21/96"	"492"	"9/18/94"	"493"	"24"	"334"
##	[565]	"628"	"707"	"770"	"437"	"56"	"263"
##	[571]	"32"	"1/15/72"	"3/8/75"	"562"	"546"	"11/30/72"
##	[577]	"8/11/71"	"1/14/65"	"595"	"51"	"64"	"8/16/95"
##	[583]	"27"	"6/22/87"	"2/27/70"	"575"	"50"	"51"

##	[589]	"348"	"409"	"3/11/59"	"1/31/93"	"22"	"709"
##	[595]	"10/24/72"	"2/26/76"	"726"	"640"	"2/18/67"	"1/27/72"
##	[601]	"28"	"63"	"51"	"564"	"338"	"36"
##	[607]	"38"	"274"	"24"	"3/28/95"	"719"	"577"
##	[613]	"621"	"629"	"2/8/94"	"513"	"433"	"478"
##	[619]	"705"	"4/15/94"	"10/28/81"	"408"	"524"	"568"
##	[625]	"291"	"12/12/75"	"1/21/83"	"351"	"33"	"3/28/62"
##	[631]	"23"	"469"	"581"	"57"	"551"	"24"
##	[637]	"397"	"4/25/91"	"3/26/88"	"683"	"3/6/95"	"497"
##	[643]	"11/17/95"	"372"	"400"	"60"	"390"	"38"
##	[649]	"340"	"12/15/92"	"12/24/60"	"435"	"8/15/85"	"535"
##	[655]	"322"	"51"	"12/20/77"	"780"	"601"	"43"
##	[661]	"530"	"598"	"653"	"768"	"43"	"41"
##	[667]	"531"	"481"	"628"	"6/22/95"	"23"	"667"
##	[673]	"36"	"299"	"1/7/61"	"44"	"63"	"322"
##	[679]	"696"	"542"	"444"	"563"	"351"	"418"
##	[685]	"647"	"551"	"441"	"53"	"4/17/77"	"668"
##	[691]	"442"	"47"	"395"	"10/26/87"	"552"	"2/5/91"
##	[697]	"477"	"520"	"8/26/86"	"472"	"42"	"687"
##	[703]	"58"	"56"	"370"	"409"	"479"	"394"
##	[709]	"456"	"37"	"465"	"3/20/77"	"486"	"695"
##	[715]	"48"	"12/17/93"	"385"	"1/4/86"	"25"	"773"
##	[721]	"38"	"293"	"10/18/53"	"573"	"565"	"26"
##	[727]	"25"	"21"	"458"	"1/5/65"	"629"	"6/22/90"
##	[733]	"694"	"779"	"12/11/75"	"472"	"21"	"325"
##	[739]	"394"	"3/19/89"	"48"	"32"	"293"	"26"
##	[745]	"437"	"561"	"532"	"456"	"4/21/62"	"645"
##	[751]	"12/16/69"	"527"	"776"	"52"	"27"	"7/18/58"
##	[757]	"31"	"11/24/70"	"352"	"528"	"10/27/78"	"461"
##	[763]	"50"	"601"	"58"	"6/11/59"	"4/28/54"	"9/25/56"
##	[769]	"22"	"640"	"735"	"5/9/66"	"279"	"48"
##	[775]	"37"	"52"	"37"	"480"	"704"	"52"
##	[781]	"44"	"58"	"23"	"684"	"550"	"385"
##	[787]	"9/28/87"	"712"	"1/2/71"	"30"	"4/28/70"	"572"
##	[793]	"748"	"50"	"22"	"9/30/96"	"50"	"479"
##	[799]	"471"	"39"	"675"	"658"	"618"	"3/27/83"
##	[805]	"10/3/81"	"39"	"428"	"32"	"305"	"41"
##	[811]	"640"	"4/5/85"	"594"	"59"	"12/27/55"	"510"
##	[817]	"3/26/66"	"45"	"261"	"9/14/63"	"611"	"670"
##	[823]	"41"	"5/13/93"	"303"	"2/20/86"	"285"	"29"
##	[829]	"597"	"313"	"5/20/68"	"24"	"710"	"10/29/55"
##	[835]	"4/16/75"	"32"	"9/28/62"	"724"	"362"	"278"
##	[841]	"3/29/55"	"273"	"64"	"1/15/56"	"64"	"627"
##	[847]	"45"	"7/2/85"	"314"	"725"	"335"	"391"
##	[853]	"9/4/69"	"47"	"38"	"410"	"39"	"22"
##	[859]	"519"	"8/10/92"	"279"	"59"	"667"	"439"
##	[865]	"1/28/89"	"507"	"590"	"295"	"610"	"45"
##	[871]	"11/14/77"	"573"	"403"	"59"	"1/1/85"	"419"
##	[877]	"317"	"496"	"44"	"437"	"63"	"323"
##	[883]	"25"	"32"	"486"	"423"	"375"	"28"
##	[889]	"12/6/81"	"522"	"21"	"531"	"731"	"487"
##	[895]	"380"	"26"	"648"	"63"	"462"	"682"
##	[901]	"8/22/69"	"500"	"615"	"608"	"56"	"56"
##	[907]	"281"	"31"	"25"	"36"	"21"	"395"

##	[913]	"12/3/91"	"42"	"704"	"12/29/86"	"49"	"57"
##	[919]	"662"	"295"	"63"	"49"	"38"	"11/10/91"
##	[925]	"711"	"271"	"40"	"11/26/64"	"43"	"47"
##	[931]	"593"	"4/8/65"	"317"	"524"	"359"	"21"
##	[937]	"12/21/53"	"21"	"48"	"8/8/80"	"25"	"12/25/75"
##	[943]	"702"	"55"	"7/10/90"	"2/12/61"	"52"	"11/27/94"
##	[949]	"2/22/62"	"58"	"4/17/80"	"9/13/55"	"1/3/68"	"629"
##	[955]	"42"	"8/16/63"	"304"	"609"	"8/18/74"	"360"
##	[961]	"28"	"638"	"62"	"6/19/77"	"5/20/94"	"22"
##	[967]	"551"	"320"	"7/2/84"	"390"	"4/10/84"	"22"
##	[973]	"721"	"1/30/92"	"488"	"21"	"468"	"64"
##	[979]	"295"	"476"	"28"	"59"	"368"	"691"
##	[985]	"380"	"11/8/68"	"3/10/90"	"384"	"2/11/93"	"559"
##	[991]	"4/21/80"	"610"	"11/16/85"	"41"	"1/27/93"	"737"
##	[997]	"294"	"407"	"7/24/71"	"53"	"638"	"52"
##	[1003]	"52"	"671"	"670"	"292"	"62"	"762"
##	[1009]	"12/14/68"	"5/24/54"	"749"	"718"	"737"	"632"
##	[1015]	"705"	"637"	"255"	"274"	"739"	"495"
##	[1021]	"28"	"522"	"407"	"321"	"507"	"25"
##	[1027]	"755"	"729"	"605"	"567"	"305"	"38"
##	[1033]	"3/15/71"	"44"	"614"	"681"	"440"	"3/23/58"
##	[1039]	"11/5/95"	"29"	"55"	"701"	"11/23/80"	"2/14/53"
##	[1045]	"7/15/55"	"23"	"332"	"7/5/75"	"61"	"430"
##	[1051]	"323"	"24"	"264"	"35"	"1/23/70"	"9/30/80"
##	[1057]	"10/15/70"	"549"	"12/19/77"	"566"	"35"	"24"
##	[1063]	"701"	"776"	"6/17/70"	"594"	"44"	"499"
##	[1069]	"1/15/94"	"765"	"627"	"641"	"589"	"7/20/84"
##	[1075]	"8/8/73"	"543"	"670"	"697"	"268"	"12/3/89"
##	[1081]	"400"	"658"	"6/15/61"	"519"	"7/11/74"	"606"
##	[1087]	"487"	"7/18/76"	"39"	"393"	"549"	"268"
##	[1093]	"559"	"691"	"2/19/68"	"681"	"444"	"10/6/69"
##	[1099]	"458"	"5/17/65"	"5/24/88"	"9/1/67"	"7/18/87"	"762"
##	[1105]	"591"	"346"	"474"	"653"	"303"	"561"
##	[1111]	"729"	"3/26/58"	"483"	"2/17/71"	"351"	"498"
##	[1117]	"9/17/68"	"9/30/90"	"448"	"474"	"537"	"573"
##	[1123]	"580"	"5/17/89"	"4/13/59"	"8/5/95"	"1/4/56"	"39"
##	[1129]	"642"	"38"	"651"	"649"	"560"	"58"
##	[1135]	"39"	"7/3/86"	"325"	"292"	"6/25/91"	"587"
##	[1141]	"508"	"47"	"56"	"4/6/73"	"56"	"319"
##	[1147]	"572"	"7/8/87"	"26"	"10/15/71"	"1/10/94"	"438"
##	[1153]	"485"	"501"	"717"	"7/30/95"	"4/5/74"	"53"
##	[1159]	"410"	"735"	"649"	"8/31/92"	"5/8/81"	"488"
##	[1165]	"8/19/95"	"533"	"41"	"402"	"4/4/59"	"695"
##	[1171]	"272"	"691"	"378"	"57"	"12/1/77"	"11/7/90"
##	[1177]	"2/22/80"	"62"	"732"	"268"	"58"	"23"
##	[1183]	"30"	"9/13/76"	"711"	"25"	"58"	"541"
##	[1189]	"7/3/56"	"450"	"7/3/61"	"596"	"594"	"44"
##	[1195]	"7/30/94"	"2/27/59"	"6/10/87"	"293"	"301"	"38"
##	[1201]	"391"	"612"	"475"	"536"	"27"	"354"
##	[1207]	"413"	"9/29/64"	"271"	"539"	"652"	"11/14/92"
##	[1213]	"302"	"597"	"582"	"674"	"35"	"728"
##	[1219]	"5/15/93"	"50"	"281"	"414"	"657"	"26"
##	[1225]	"381"	"12/19/53"	"770"	"31"	"4/14/90"	"33"
##	[1231]	"448"	"687"	"364"	"10/25/73"	"26"	"713"

##	[1237]	"733"	"673"	"755"	"53"	"56"	"54"
##	[1243]	"386"	"720"	"7/21/68"	"598"	"25"	"3/31/85"
##	[1249]	"34"	"326"	"489"	"10/20/61"	"595"	"58"
##	[1255]	"2/14/53"	"359"	"274"	"64"	"1/2/57"	"271"
##	[1261]	"6/10/78"	"29"	"408"	"9/8/66"	"46"	"695"
##	[1267]	"35"	"332"	"356"	"58"	"48"	"6/3/85"
##	[1273]	"7/19/55"	"43"	"2/19/64"	"8/5/83"	"779"	"2/28/69"
##	[1279]	"260"	"370"	"7/16/77"	"262"	"51"	"5/16/64"
##	[1285]	"54"	"686"	"57"	"773"	"463"	"670"
##	[1291]	"765"	"649"	"1/4/94"	"452"	"284"	"549"
##	[1297]	"9/8/75"	"634"	"28"	"357"	"21"	"396"
##	[1303]	"63"	"55"	"28"	"1/15/76"	"5/26/62"	"740"
##	[1309]	"414"	"41"	"33"	"7/11/82"	"396"	"12/10/78"
##	[1315]	"5/22/58"	"47"	"5/9/77"	"295"	"8/10/75"	"12/22/89"
##	[1321]	"329"	"502"	"425"	"50"	"334"	"3/15/69"
##	[1327]	"634"	"11/24/61"	"472"	"30"	"10/23/66"	"38"
##	[1333]	"506"	"6/4/76"	"61"	"315"	"6/23/91"	"496"
##	[1339]	"59"	"609"	"519"	"763"	"5/24/75"	"54"
##	[1345]	"3/15/72"	"7/18/55"	"377"	"3/15/64"	"621"	"284"
##	[1351]	"3/21/94"	"9/20/70"	"24"	"59"	"554"	"651"
##	[1357]	"45"	"315"	"727"	"391"	"469"	"55"
##	[1363]	"61"	"8/18/62"	"709"	"353"	"49"	"6/25/83"
##	[1369]	"497"	"496"	"34"	"548"	"297"	"1/13/64"
##	[1375]	"6/20/83"	"740"	"48"	"11/28/54"	"8/7/92"	"44"
##	[1381]	"47"	"56"	"715"	"359"	"578"	"655"
##	[1387]	"2/19/69"	"32"	"606"	"46"	"44"	"730"
##	[1393]	"12/29/96"	"5/2/93"	"8/21/68"	"733"	"5/22/60"	"656"
##	[1399]	"41"	"476"	"61"	"10/3/60"	"412"	"31"
##	[1405]	"541"	"8/30/65"	"2/7/64"	"45"	"513"	"52"
##	[1411]	"9/20/77"	"533"	"48"	"408"	"10/3/80"	"61"
##	[1417]	"768"	"760"	"262"	"3/25/72"	"515"	"2/12/58"
##	[1423]	"459"	"33"	"428"	"23"	"1/29/86"	"605"
##	[1429]	"6/9/75"	"60"	"499"	"262"	"10/7/66"	"55"
##	[1435]	"723"	"317"	"378"	"778"	"48"	"532"
##	[1441]	"288"	"44"	"599"	"560"	"1/17/76"	"47"
##	[1447]	"702"	"719"	"47"	"62"	"24"	"273"
##	[1453]	"9/25/59"	"32"	"49"	"60"	"45"	"7/5/65"
##	[1459]	"315"	"672"	"736"	"44"	"529"	"414"
##	[1465]	"690"	"64"	"38"	"5/2/78"	"5/19/95"	"43"
##	[1471]	"5/16/86"	"429"	"1/10/68"	"47"	"5/19/54"	"25"
##	[1477]	"319"	"48"	"456"	"61"	"31"	"616"
##	[1483]	"379"	"530"	"474"	"2/20/90"	"11/21/80"	"487"
##	[1489]	"572"	"607"	"23"	"55"	"49"	"672"
##	[1495]	"11/27/59"	"286"	"5/4/84"	"600"	"5/8/77"	"12/28/88"
##	[1501]	"678"	"264"	"370"	"43"	"3/1/76"	"572"
##	[1507]	"645"	"584"	"706"	"698"	"564"	"35"
##	[1513]	"2/5/81"	"342"	"9/15/60"	"37"	"7/25/76"	"6/5/92"
##	[1519]	"510"	"749"	"569"	"742"	"545"	"316"
##	[1525]	"11/9/69"	"664"	"583"	"354"	"459"	"681"
##	[1531]	"421"	"701"	"44"	"722"	"12/8/75"	"35"
##	[1537]	"268"	"33"	"28"	"569"	"4/2/66"	"429"
##	[1543]	"743"	"555"	"262"	"400"	"12/26/88"	"404"
##	[1549]	"34"	"412"	"46"	"11/3/63"	"455"	"560"
##	[1555]	"621"	"655"	"9/21/95"	"470"	"6/11/61"	"9/13/84"

## [1561]	"58"	"11/5/92"	"312"	"3/15/64"	"28"	"348"
## [1567]	"710"	"453"	"54"	"23"	"1/1/71"	"1/28/91"
## [1573]	"351"	"44"	"42"	"12/30/77"	"766"	"55"
## [1579]	"272"	"7/12/53"	"708"	"306"	"453"	"31"
## [1585]	"254"	"463"	"707"	"7/25/71"	"595"	"2/10/76"
## [1591]	"8/5/86"	"4/1/53"	"451"	"363"	"689"	"343"
## [1597]	"26"	"64"	"58"	"55"	"57"	"620"
## [1603]	"556"	"550"	"369"	"9/17/89"	"9/30/55"	"1/28/70"
## [1609]	"537"	"405"	"29"	"48"	"542"	"23"
## [1615]	"5/14/68"	"10/15/79"	"731"	"50"	"5/8/78"	"11/11/67"
## [1621]	"4/1/60"	"425"	"506"	"700"	"609"	"338"
## [1627]	"541"	"11/14/61"	"29"	"554"	"736"	"536"
## [1633]	"33"	"27"	"335"	"447"	"596"	"55"
## [1639]	"9/4/77"	"746"	"483"	"9/4/68"	"44"	"44"
## [1645]	"23"	"300"	"751"	"391"	"697"	"48"
## [1651]	"750"	"459"	"11/11/59"	"6/4/94"	"515"	"47"
## [1657]	"387"	"49"	"46"	"1/23/75"	"27"	"49"
## [1663]	"39"	"25"	"2/11/65"	"527"	"419"	"50"
## [1669]	"12/1/87"	"60"	"511"	"41"	"4/7/81"	"12/23/93"
## [1675]	"45"	"5/20/77"	"327"	"322"	"45"	"3/12/95"
## [1681]	"531"	"8/31/81"	"480"	"338"	"3/27/81"	"725"
## [1687]	"31"	"516"	"58"	"52"	"544"	"22"
## [1693]	"11/28/83"	"254"	"558"	"271"	"463"	"40"
## [1699]	"8/23/61"	"11/2/54"	"63"	"357"	"43"	"487"
## [1705]	"302"	"315"	"10/19/87"	"4/23/65"	"31"	"7/31/78"
## [1711]	"48"	"570"	"596"	"56"	"48"	"44"
## [1717]	"398"	"43"	"8/14/53"	"57"	"762"	"8/19/94"
## [1723]	"7/11/75"	"362"	"6/1/81"	"57"	"28"	"57"
## [1729]	"8/28/95"	"443"	"330"	"7/18/77"	"2/13/53"	"704"
## [1735]	"50"	"12/5/88"	"554"	"43"	"609"	"482"
## [1741]	"720"	"25"	"684"	"37"	"57"	"595"
## [1747]	"756"	"9/10/80"	"778"	"3/25/86"	"424"	"542"
## [1753]	"57"	"320"	"404"	"39"	"8/4/83"	"5/27/87"
## [1759]	"31"	"7/14/57"	"393"	"318"	"36"	"641"
## [1765]	"343"	"21"	"310"	"566"	"8/25/70"	"488"
## [1771]	"60"	"604"	"12/19/80"	"276"	"715"	"344"
## [1777]	"269"	"3/21/95"	"717"	"314"	"8/13/55"	"26"
## [1783]	"11/21/65"	"458"	"12/13/84"	"11/8/86"	"594"	"44"
## [1789]	"372"	"12/11/94"	"625"	"361"	"1/18/55"	"8/26/77"
## [1795]	"388"	"292"	"3/14/75"	"55"	"593"	"1/4/79"
## [1801]	"663"	"691"	"621"	"38"	"12/3/69"	"346"
## [1807]	"64"	"774"	"58"	"631"	"412"	"497"
## [1813]	"695"	"7/6/66"	"257"	"728"	"23"	"520"
## [1819]	"6/11/86"	"8/28/77"	"402"	"3/8/68"	"12/20/89"	"11/14/66"
## [1825]	"1/10/66"	"556"	"29"	"11/28/79"	"552"	"542"
## [1831]	"9/29/55"	"27"	"9/12/72"	"336"	"608"	"470"
## [1837]	"647"	"351"	"39"	"516"	"457"	"10/16/93"
## [1843]	"45"	"11/15/53"	"8/7/74"	"41"	"5/22/53"	"55"
## [1849]	"26"	"63"	"62"	"469"	"350"	"511"
## [1855]	"747"	"7/15/55"	"37"	"568"	"468"	"325"
## [1861]	"487"	"627"	"28"	"3/5/56"	"685"	"727"
## [1867]	"12/3/91"	"10/7/62"	"5/5/53"	"53"	"9/12/57"	"27"
## [1873]	"556"	"455"	"558"	"12/25/60"	"705"	"496"
## [1879]	"53"	"1/9/85"	"442"	"27"	"426"	"745"

```
## [1885] "39"      "1/22/96" "2/3/63"  "333"     "42"      "2/2/87"
## [1891] "395"     "257"     "44"      "12/5/81" "2/27/65" "448"
## [1897] "578"     "594"     "58"      "512"     "8/3/63"  "8/10/78"
## [1903] "470"     "40"      "437"     "778"     "716"     "549"
## [1909] "51"      "393"     "45"      "7/11/61" "1/5/58"  "10/11/62"
## [1915] "700"     "661"     "282"     "653"     "31"      "339"
## [1921] "309"     "679"     "7/29/59" "55"      "621"     "42"
## [1927] "40"      "12/15/59" "11/24/79" "9/1/84"  "5/1/54"  "370"
## [1933] "5/10/88" "667"     "696"     "9/15/74" "37"      "24"
## [1939] "348"     "9/30/79" "43"      "45"      "45"      "273"
## [1945] "12/8/58" "293"     "63"      "5/9/54"  "281"     "31"
## [1951] "5/10/71" "292"     "498"     "676"     "532"     "53"
## [1957] "392"     "627"     "728"     "3/23/58" "2/20/60" "761"
## [1963] "366"     "30"      "326"     "309"     "492"     "45"
## [1969] "531"     "9/21/72" "647"     "344"     "598"     "4/24/61"
## [1975] "11/29/59" "599"     "452"     "365"     "743"     "6/11/59"
## [1981] "48"      "6/11/72" "466"     "23"      "315"     "710"
## [1987] "4/9/54"  "765"     "23"      "547"     "36"      "433"
## [1993] "470"     "10/14/96" "7/17/56" "62"      "678"     "632"
## [1999] "424"     "50"
```

Let's make a vector of these

```
birthdate_rows = grep('^((\\d{1}|\\d{2})\\/(\\d{1}|\\d{2})\\/(\\d{2}))', in_data$age, perl = TRUE)
```

Let's look at the values

```
in_data[birthdate_rows, "age"][1:10]
```

```
## [1] "1/7/66" "8/19/68" "4/2/60" "9/14/74" "6/21/59" "5/31/82" "8/5/62"
## [8] "1/7/54" "5/1/55" "1/13/62"
```

Here is what happens when we try to parse the MM/DD/YY format...

```
as.Date(in_data[birthdate_rows, "age"][1:10], format = '%d/%m/%y')
```

```
## [1] "2066-07-01" NA      "2060-02-04" NA      NA
## [6] NA      "2062-05-08" "2054-07-01" "2055-01-05" NA
```

And the same thing using a wrapper in lubridate

```
mdy(in_data[birthdate_rows, "age"][1:10])
```

```
## [1] "2066-01-07" "2068-08-19" "2060-04-02" "1974-09-14" "2059-06-21"
## [6] "1982-05-31" "2062-08-05" "2054-01-07" "2055-05-01" "2062-01-13"
```

UH OH! The dates are in the future! LET THIS BE A LESSON TO ALWAYS USE 4 YEAR DATES. If we know that the dates are all 1900s dates we can write our own function that converts the date strings to dates in the 1900s. I won't go too far into the syntax here but it should be simple enough to follow along.

```
nineteen_hundreds_bday <- function(x, year=2017){
  # Parse as MM/DD/YY date
  x <- mdy(x)

  # Get the last two digits of the date
  m <- year(x) %% 100

  # Set the year to the 1900s year with that two digit suffix
  year(x) <- 1900+m
  return(x)
}
```



```
# Test it
nineteen_hundreds_bday(c("10/3/63", "12/26/73", "12/8/91"))
```

```
## [1] "1963-10-03" "1973-12-26" "1991-12-08"
```

We will also need a function to convert the time between the birthday and collection date to months.

```
months_between <- function(end_date, start_date) {
  end_dt <- as.POSIXlt(end_date)
  start_dt <- as.POSIXlt(start_date)

  # 12 times the elapsed years + the elapsed months intrayear
  return(12 * (end_dt$year - start_dt$year) + (end_dt$mon - start_dt$mon))
}
```

```
# Test it
months_between("2016-02-23", "2015-01-28")
```

```
## [1] 13
```

Phew! Now let's put it all together.

```
head(wrk_data$age)
```

```
## [1] "1/7/66" "8/19/68" "50" "671" "4/2/60" "9/14/74"
```

```
wrk_data$age = in_data$age
```

```
wrk_data[birthdate_rows, ]$age = as.character(months_between(in_data[birthdate_rows, "collection_date"]
wrk_data[, 'age'] = as.integer(wrk_data$age)
```

8 Including Plots

You can also embed plots, for example:

```
# {r pressure, echo=FALSE}
plot(pressure)
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.