TAs’ Birthdays:
A Data Cleaning ICA

Data Cleaning Libraries

```
> install.packages("stringr")
> install.packages("lubridate")
```

Setup

Create an R markdown file, and include the following as your setup:

````
```{r setup, include = FALSE}
library(lubridate)
library(stringr)
library(dplyr)
```
````

Birthdays

At the beginning of the year, the TAs filled out a questionnaire about themselves. Unfortunately, they failed to specify a standard date format, so they all entered their birthdays in different ways.

Today, we will clean their birthday entries, so they can be easily processed in R!

The Dataset

```
tas <- read.csv("https://cs.brown.edu/courses/cs100/studios/

data/5/birthdays.csv", stringsAsFactors = FALSE)
```

- Add this line to your setup chunk to read in the data
- `stringsAsFactors = FALSE` keeps the strings as strings, instead of converting them to factors (categorical variables)

```
> tas
      Name    Birthdate Birthyear
1     Erin         3.10      1998
2     Juho       31-May      1996
3     Iris  August 29th      1998
4      Ben     Sept. 20      1998
5     Anna    June 11th      1998
6  Shivani                     NA
7    David         4/26      1999
8  Sarah C       16-Nov      1997
9  Sarah F       Dec 18      1997
10   Will  November 3rd      1997
```

NAs

One of the TAs failed to enter their birthday!
So we will remove that observation from the dataset using `na.omit()`

```
# Omit NAs
tas <- na.omit(tas)
```

Data Types

Now, to clean the remaining data, let's combine the "Birthdate" and "Birthyear" variables into one new variable called "Birthdays", and then let's standardize the format of all birthday entries.

For starters, we need to choose a datatype for birthdays. "Birthdate" and "Birthyear" are both strings. But storing dates as strings is not that useful, because you cannot easily do arithmetic with strings, and you often want to do arithmetic with dates, for example, to calculate ages.

R has a special data type called `Date`.

Paste

For starters, paste the contents of “Birthdate” and “Birthyear” together and view the new variable in the console:

```
# Combine variables
fullbirthday <- paste(tas$Birthdate, tas$Birthyear)
```

```
> fullbirthday
[1] "3.10 1998"          "31-May 1996"        "August 29th 1998"  "Sept. 20 1998"
[5] "June 11th 1998"     "4/26 1999"          "16-Nov 1997"       "Dec 18 1997"
[9] "November 3rd 1997"
```

`str_replace()`

Because these data are so messy (the format of each entry is different than the last), we have to clean/standardize them manually, one at a time.

For concreteness, let's go with `yyyymmdd`, meaning a four-digit year, followed by two digits for the month, and then two more digits for the day of the month, with no spaces or other separators between any of the digits.

One way to proceed is with the `str_replace` function in the stringr package.

str_replace()

```
# Standardize birthday formats
bd <- str_replace(fullbirthday, "3.10 1998", "19980310")
```

Repeat this step for all the TAs' birthdays.

```
> bd
[1] "19980310" "19960531" "19980929" "19980920" "19980611" "19990426" "19971116"
[8] "19971218" "19971103"
```

ymd

The `ymd` function in the lubridate package takes in a vector of strings and converts each one to a `Date`.

```
birthdays <- ymd(bd)
```

Add `birthdays` to the TAs data frame

Add the `birthdays` vector to your `tas` data frame as a new variable called "birthdays".

```
tas$birthdays <- birthdays
```

Congratulations! Your data are clean!

Now, we can perform calculations with the TA's birthdays, like calculate their average age. To calculate an age, we can the lubridate function, `today`.

Try typing `today` in the console to see what it does. Then, add an "age" variable to the TAs data frame, and calculate the TAs' average age.

```
tas$age <- (today() - tas$birthdays) / 365
mean(tas$age)
```

Add Joon

Joon is a former CS100 TA. His birthday is April 10th, 1993.
What is the average age of the CS100 TA staff, with Joon?

Insert Joon into the data frame, using the `rbind()` function.
Then recalculate the average TA age.

```
joon <- c("Joon", "19930410")
tas <- rbind(tas, joon)
```

Save the clean data

Finally, we should save our cleaned-up version of the data. We can do so in several ways: e.g., as a `.csv` file, a `.tsv` file, etc. But we can also save our data as an `.rds` file, which is R's unique data format.

The benefit of using `.rds` is that it can preserve the variables' R data types. That way, if you importa your data frame into R later, nothing will have to be converted from one type to another.

A `.csv` file is just a text file, so saving as a `.csv` does not store R specific data types. Saving to a `.csv` file will store all `Dates` as strings instead.

Save the clean data

To save as a `.csv` file:

```
filename_csv <- paste("cleanBirthdays", today(), ".csv", sep = "")
write.csv(tas, file = filename_csv)
```

To save as an `.rds` file:

```
filename_rds <- paste("cleanBirthdays", today(), ".rds", sep = "")
saveRDS(tas, file = filename_rds)
```