Manipulation of Time-Based Data

Hannah Spinner

Introduction & Motivation

What day of the week did Sputnik launch? How does the temperature in Basel, Switzerland change over the course of a day? How have crime rates changed over the last decade?

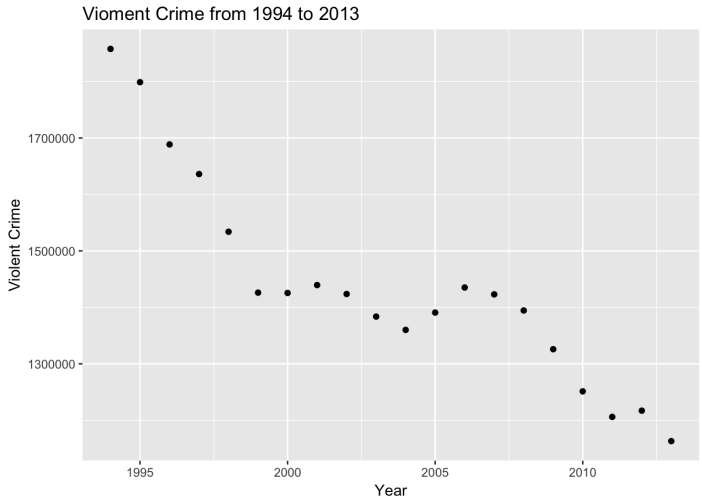
All of these questions involve time and time manipulation. Many data analyses projects involve manipulations of seconds, hours, minutes, days, weeks, months, and/or years. Positions in time are very important for many data sets, including the US Census data[1], FBI crime over the past 20 years[2], CDC public heatlh information[3], Bureau of Labor Statistics data sets[4], Weather data from weather stations[5], and the list goes on and on. Seeing how trends change over time is immensely important to understanding patters in data.

For example, we can use the FBI data set to see how the amount of violent crime has changed from 1994 to 2013:

```
violent_crime_by_year <- read_csv(
  '/Users/hspinner/Desktop/stat133/stat133-hws-fall17/post01/data/fbi_data.csv')
```

```
## Parsed with column specification:
## cols(
##   year = col_integer(),
##   violent_crime = col_number(),
##   vioment_crime_rate = col_double(),
##   murder_and_nonnegligent_manslaughter = col_number(),
##   murder_and_nonnegligent_manslaughter_rate = col_double(),
##   rape = col_number(),
##   rape_rate = col_double(),
##   robbery = col_number(),
##   robbery_rate = col_double(),
##   aggravated_assult = col_number(),
##   aggravated_assult_rate = col_double(),
##   property_crime = col_number(),
##   property_crime_rate = col_number(),
##   burgalry = col_number(),
##   burgalry_rate = col_number(),
##   larceny_theft = col_number(),
##   larceny_theft_rate = col_number(),
##   motor_vehicle_theft = col_number(),
##   motor_vehicle_theft_rate = col_double()
## )
```

```
crime_plot <- ggplot(data = violent_crime_by_year, aes(x = year, y = violent_crime)) +
  geom_point() +
  labs(title = "Vioment Crime from 1994 to 2013", x = "Year", y = "Violent Crime")
crime_plot
```



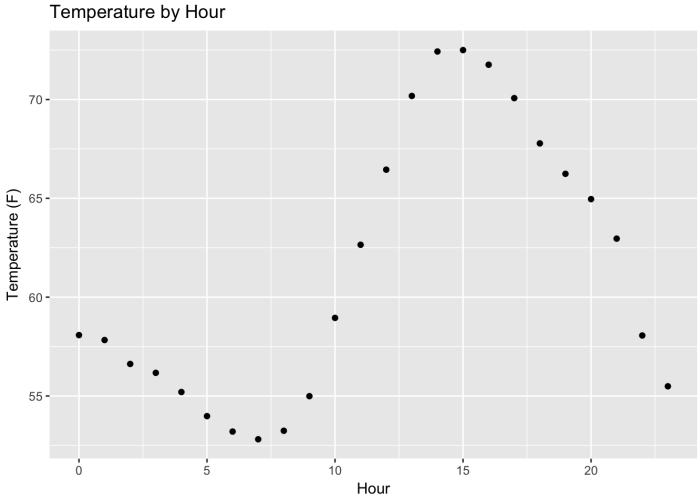
Here, you can see that the general pattern is decreasing crime over time; the amount of violent crime is generally going down over the years. However, this graph only involves years as the measure of time.

Let's look at something slightly more complex… Weather! MeteoBlue is a website that stores weather data for different places every single hour. They have all sorts of data such as temperature, humidity, pressure, precipitation, snowfall, cloud cover, sunshine, solar radiation, and wind speed and direction. For example, let's look at how temperature in Basel, Switzerland changes by hour over the course of one day, October 26, 2017:

```
temperature_by_hour <- read_csv(
  '/Users/hspinner/Desktop/stat133/stat133-hws-fall17/post01/data/weather.csv')
```

```
## Parsed with column specification:
## cols(
##   Year = col_integer(),
##   Month = col_integer(),
##   Day = col_integer(),
##   Hour = col_integer(),
##   Minute = col_integer(),
##   Temperature = col_double(),
##   Relative_humidity = col_integer()
## )
```

```
weather_plot <- ggplot(data = temperature_by_hour, aes(x = Hour, y = Temperature)) +
  geom_point() +
  labs(title = "Temperature by Hour", x = "Hour", y = "Temperature (F)")
weather_plot
```



Here, you can see that the temperature initially decreases until about 7am, and then quickly rises throughout the morning and afternoon until a peak at about 3pm. After this, it quickly drops throughout the evening.

Because of trends like this and general trends one can glean from time-based data, it is important to have a way of manipulating time data accurately. One thing we might want to do is get a range of time data. For instance, when looking at weather patterns or crime rates, one might want to know how many days are there between dates like October 20th and 30th. One way of getting the range between two dates is as follows:

```
date_a <- 2017-10-20
date_b <- 2017-10-30
diff(c(date_b, date_a))
```

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

However, this is cumbersome and would be challenging to do for a large quantity of dates. More importantly, this method is not always accurate. Finding the range between two months proves a more formidable task:

```
date_a <- 2017-10-31
date_b <- 2017-11-1
diff(c(date_b, date_a))
```

```
## [1] -29
```

The range between these two dates should be 1 since there is one day between October 31st and November 1st, but R is telling us that there are -29 days between the two dates. This is due to improper assignment of dates and not using the proper formatting (which we will get into later). This method of processing time data is messy and inaccurate. There must be a better way! Luckily, there is!

The motivation of this post is to explore some packages in R that make time data manipulation easy, efficient, and accurate. The package lubridate makes data analysis of time much more straightforward.

Background

There are two sources[6,7] that describe the package “lubridate” in great detail. In the first source, “Dates and Times Made Easy with libridate” by Garrett Grolemund and Hadley Wickham, it describes lubridate in a very broad, practical sense. This published paper describes all of the benefits of using lubridate over the base functions in R. The second source is a cran file entitled “Package ‘lubridate’” also written by Garrett Grolemund, Hadley Wickham, and several others. In contrast to the paper, this cran post describes every single function in lubridate as it would appear if someone typed:

```
?am
```

The description of the “am” function in the Help tab of R appears exactly the same as the description of “am” in the cran post.

The package lubridate assists data scientists who work with dates and times since these tend to cause technical issues when analyzing data.

Examples

1. Inputting Dates

First, let’s consider the simple scenario of inputting a date-time datum into R. In the base R package, that would look like this:

```
oct_27_2017 <- as.POSIXct("27-10-2017", format = "%d-%m-%Y", tz = "UTC")
oct_27_2017
```

```
## [1] "2017-10-27 UTC"
```

Note: throughout this post we will use the standard of Coordinated Universal Time, as it is the standard time zone for time data manipulation.

In lubridate, all of that would simply look like this:

```
oct_26_2017 <- dmy("26-10-2017")
oct_26_2017
```

```
## [1] "2017-10-26"
```

In base R, you have to input the date, the format of said date (where “d” = day, “m” = month, and “Y” = year), and the time zone. The “Y” must be capitalized or the function will not work, which is a pain.

In contrast, lubridate syntax is much cleaner and easier to interpret. Days, months, and years get assigned by the function used to import them. This way, you can very easily import dates in any format; there are options for ymd(), myd(), and dym(). If the data set includes hours and minutes, there are functions for those too! These include ymd_h, ymd_hm, and ymd_hms. Additionally, lubricate will always output the date as year-month-day.

2. Changing Dates

Now, say that we wanted to change one of the elements within the date. If we wanted to change 2017 to 2016 in the R base package, it would be much more cumbersome than in lubridate. With the base R, you would manually have to change the year to 2016. It would look like this:

```
as.numeric(format(oct_27_2017, "%Y"))
```

```
## [1] 2017
```

```
oct_27_2016 <- as.POSIXct(format(oct_27_2017, "2016-%m-%d"), tz = "UTC")
oct_27_2016
```

```
## [1] "2016-10-27 UTC"
```

However, with lubridate, you would simply code:

```
year(oct_26_2017) <- 2016
oct_26_2017
```

```
## [1] "2016-10-26"
```

Again, it is painfully obvious how much easier lubridate is when it comes to manipulating dates!

3. Extracting Date Info

Another great thing that lubridate does is enable us to efficiently extract information from the date and specify exactly how we want that output.

For example, say you wanted to know the year, month, week, day of the year, day of the month, or day of the week from the date above:

```
year(oct_26_2017)
```

```
## [1] 2016
```

```
month(oct_26_2017)
```

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

```
week(oct_26_2017)
```

```
## [1] 43
```

```
yday(oct_26_2017)
```

```
## [1] 300
```

```
mday(oct_26_2017)
```

```
## [1] 26
```

```
wday(oct_26_2017)
```

```
## [1] 4
```

With these you can also specify what you want these outputs to look like; you can specify if you want numeric output, abbreviated, or full name. For example:

```
month(oct_26_2017, label = FALSE, abbr = FALSE)
```

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

```
month(oct_26_2017, label = TRUE, abbr = FALSE)
```

```
## [1] October
## 12 Levels: January < February < March < April < May < June < ... < December
```

```
month(oct_26_2017, label = TRUE, abbr = TRUE)
```

```
## [1] Oct
## 12 Levels: Jan < Feb < Mar < Apr < May < Jun < Jul < Aug < Sep < ... < Dec
```

This is actually a very cool functionality of lubridate! You can see which day of the week something is, what day of the year it is, and so much more. These functions save a lot of head scratching when one is trying to gather information about a data set from long ago. For example, one fun date[8] is Oct 4th, 1957 when they launched Sputnik into space! I wonder what day of the week that was…if I was a space agency, would I want my shuttle to launch on a Monday at the start of the week? On hump-day? On Friday to kick off the weekend? Let's use libridate to find out!

```
sputnik <- dmy("4-10-1957")
wday(sputnik)
```

```
## [1] 6
```

Wow! It was on day 6 of the week, or on a Saturday.

4. Ranges of Dates

Like mentioned at the beginning of this post, what if we want to find the range between two dates? How much time is there between October 31st and November 1? It seems trivial, but let's find out!

```
date1 <- mdy("10-31-2017")
date2 <- mdy("11-1-2017")
span <- date2 - date1
span
```

```
## Time difference of 1 days
```

Yay, we got one day this time! By using lubridate, we were able to accurately glean the span between these two dates and avoid the cumbersome syntax of base R. Try finding the span between the day Sputnik launched and Halloween this year! (The answer is 21942 days, or 60 years and 27 days.)

Discusion & Conclusions

As seen by the examples above, the package lubridate is a fast and easy way to manipulate date data in R. It can help with practical things like finding date ranges, but also some fun things like finding out what day of the week historical events occured on!

The main take-home message of this post is that lubridate is a great package for manipulating time data in R. I hope you find it useful in your future coding projects!

References

[1]US Census data: https://www2.census.gov/acs2013_1yr/summaryfile/2013_ACSSF_All_In_1_Giant_File(Experienced-Users-Only)/

[2]FBI Crime data: https://ucr.fbi.gov/crime-in-the-u.s/2013/crime-in-the-u.s.-2013/tables/1tabledatadecoverviewpdf/table_1_crime_in_the_united_states_by_volume_and_rate_per_100000_inhabitants_1994-2013.xls

[3]CDC public health data: https://wonder.cdc.gov

[4]Bureau of Labor Statistics: https://www.bls.gov/data/#prices

[5]Weather: https://www.meteoblue.com/en/weather/archive/export/basel_switzerland_2661604?daterange=2017-10-19+to+2017-10-26&params=&params%5B%5D=11%3B2+m+above+gnd&params%5B%5D=52%3B2+m+above+gnd&utc_offset=1&aggregation=hourly&temperatureunit=FAHRENHEIT&windspeed

[6] Journal of Statistical Software lubridate: https://www.jstatsoft.org/article/view/v040i03/v40i03.pdf

[7] cran Lubridate: https://cran.r-project.org/web/packages/lubridate/lubridate.pdf

[8]Fun dates! https://www.shmoop.com/1960s/timeline.html