I’ve been using data.table quite a lot in my spare time, and I want to make a few notes of some things I’ve learned along the way. I would say I’m no longer a beginner, not yet an expert, but getting there..

As noted in a previous post, I’ve been doing the PreppinData challenges, or at least I was – things have fallen a bit by the wayside. A recent challenge allows me to demonstrate some ways to use data.table that you may not have been aware of.

You can see the details of the challenge here:

PreppinData Week 8

First of all, you may know that you can use `setDT` to convert an existing data.frame or tibble to a data.table.

As part of the recent challenged I had to read in some Excel files , which were tibbles as a result of using readxl, then change them to data.tables.

Here was my code – read the files in, then assign them with `setDT`

```
# Input the data
choices <- read_excel("karaoke.xlsx", col_types = c("date","text",
"text"))

customers <- read_excel("karaoke.xlsx", sheet = "Customers",
                        col_types = c("text", "date"))

setDT(choices)
setDT(customers)
```

But, you can actually make this a bit slicker, by wrapping the call to `setDT` around the call to `read_excel`

```
choices <- setDT(read_excel("karaoke.xlsx",
                            col_types = c("date","text", "text")))

customers <- setDT(read_excel("karaoke.xlsx", sheet = "Customers",
                        col_types = c("text", "date")))
```

At this point, I will back up a little bit, and explain the challenge, and why data.table was the right choice.

There were 2 datasets which needed to be joined - a list of karaoke songs (date, artist and song title), and a list of customers (customer ID and date of entry).

The requirements were to:

- Input the data
- Calculate the time between songs
- Determine the number of sessions - If the time between songs is greater than (or equal to) 59 minutes, flag this as being a new session

- Number the songs in order for each session
- Match the customers to the correct session, based on their entry time
- The Customer ID field should be null if there were no customers who arrived 10 minutes (or less) before the start of the session
- Output the data

Knowing that I need to join these tables together, one useful little hack with data.table is to copy the columns you want to join, and make sure they have the same name, like so:

```
# create columns to join on, then sort them  by set a key on both
columns
choices[,join_time := Date]
customers[,join_time := `Entry Time`]

#order with setkey
setkey(choices, join_time)
setkey(customers, join_time)
```

When I first do some work with data.table, I like to be really explicit about what I'm doing, and I often use empty square brackets at the end of each line, so that I know my code is doing what I expect. (The empty brackets force data.table to update in the console so that you see the current state of the data.table)

Here I check to see if the time between songs is greater than 59 minutes

```
# If the time between songs is greater than (or equal to) 59 minutes,
# flag this as being a new session
choices[,lag_date := shift(Date, type = 'lag', fill = 1L)][]
choices[,new_session := difftime(Date,lag_date,'mins')>= 59][]
```

The next step was to identify each session, and number the songs in each session

```
# Create a session number field
# Number the songs in order for each session
choices[,session_number := cumsum(new_session == TRUE)][]
choices[,song_number := rowid(session_number)][]

# delete columns we no longer need
choices[,`:=`(new_session = NULL, lag_date = NULL)][]
```

Now the cool part. We need to match these two tables, but even though we have a date time column in both, they are not containing values that you would expect to join on.

For example, how do we match '27/12/2020 06:55:00' with a value like '22/12/2020 13:59:59'?

How? Rolling joins to the rescue.

```
# Match the customers to the correct session, based on their entry time
merged <- customers[choices, roll = TRUE][,join_time := NULL][]
```

What this does is join customers to the session date that is closest to their time of entry. The time of entry will be either a match or they will have come in prior to the start of the session.

Rolling joins can be specified in either direction (e.g. I could have searched for those who came in after the start of the session) and the order of the tables is also important.

Data.table does right joins by default, so its always `lookup_table[main_data_table,]` rather than `main[lookup,]`

The final step is to check clear out any customer IDs if they did not arrive at least 10 mins before the start of a session. There aren't very many of them.

```
# The Customer ID field should be null if there were no customers who
# arrived 10 minutes (or less) before the start of the session
merged[,session_start := min(Date), by = session_number][]
merged[,entry_check := session_start - lubridate::minutes(10)][]
merged[!between(`Entry Time`,entry_check,session_start), `Customer ID`
:= NA][]

# Output the data
setkey(merged,'Date','session_number','Customer ID','song_number')
output <- merged[,c('session_number','Customer ID',
'song_number','Date','Artist','Song')][]
fwrite(output,"final_output.csv")
```

With a bit of tweaking, the code can be condensed a fair bit:

```
choices <- setDT(read_excel("karaoke.xlsx",
                            col_types = c("date","text", "text")))

customers <- setDT(read_excel("karaoke.xlsx", sheet = "Customers",
                        col_types = c("text", "date")))

# create columns to join on, then sort
choices[,join_time := Date]; setkey(choices, join_time)
customers[, join_time := `Entry Time`]; setkey(customers, join_time)

# add session number & check entry time within 10 minutes of session
start
choices[,lag_date := shift(Date, type = 'lag', fill = 1L)
        ][,new_session := difftime(Date,lag_date,'mins')>= 59
          ][,session_number := cumsum(new_session == TRUE)
            ][,song_number := rowid(session_number)
              ][,session_start := min(Date), by = session_number
                ][,entry_check := session_start -
lubridate::minutes(10)
                  ][,`:=`(new_session = NULL, lag_date = NULL)]

merged <- customers[choices, roll = TRUE]
merged[!between(`Entry Time`,entry_check,session_start), `Customer ID`
:= NA]

setkey(merged,'Date','session_number','Customer ID','song_number')
```

```
output <- fwrite(merged[,c('session_number','Customer ID',
                                    'song_number','Date','Artist',
'Song')],"final_output.csv")…
```