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..

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")…