In our previous series on [Animating Data Transformations](https://omnianalytics.io/2018/08/30/animating-the-data-transformation-process-with-gganimate/), we showed you how to use *gganimate* to construct an animation which illustrates the process of going between tall and wide representations of data. Today, we will show the same procedure for constructing an animation of the *unnest()* function.

The *unnest()* function takes a tibble containing a list column and converts it to a tibble such that each element in the list comprises a single row. Think of it as “unpacking” the list column into a more standard vector column in R.

We will create a toy dataset using the sleep data, and ensure that it contains a list column:

library(tidyverse)

library(gganimate)

sleep\_data <- sleep %>%

mutate(group = rep(1:4, each = 5),

ID = rep(1:4, each = 5)) %>%

select(id = ID, group, data = extra)

sleep\_data

# A tibble: 20 x 3

id group data

1 1 1 0.7

2 1 1 -1.6

3 1 1 -0.2

4 1 1 -1.2

5 1 1 -0.1

...

You can see that we have a 20 row by 3 column tibble in R. Next, we perform the following routine in order to nest the data column into a new *nesteddata* column:

sleep\_nested <- sleep\_data %>%

group\_by(group) %>%

summarise(id = id[1], nesteddata = list(data))  
  
sleep\_nested

# A tibble: 4 x 3

group id nesteddata

1 1 1

2 2 2

3 3 3

4 4 4

Next, we perform a similar routine to the previous blog and combine the two datasets into one dataset which will be used to build the animation:

longDat <- function(x) {

names(x) %>%

rbind(x) %>%

setNames(seq\_len(ncol(x))) %>%

mutate(row = row\_number()) %>%

tidyr::gather(column, value, -row) %>%

mutate(column = as.integer(column)) %>%

ungroup() %>%

arrange(column, row)

}

long\_tables <- map(list(sleep\_nested, sleep\_data), longDat)

nested\_table <- long\_tables[[1]] %>%

mutate(tstep = "a",

value = sapply(value, paste, collapse = ", "))

unnested\_table <- long\_tables[[2]] %>%

mutate(tstep = "b")

both\_tables <- bind\_rows(nested\_table, unnested\_table)

both\_tables$celltype[both\_tables$column == 1] <- c("header", rep("id", 4), "header", rep("id", 20))

both\_tables$celltype[both\_tables$column == 2] <- c("header", rep(1:4, each = 1), "header", rep(1:4, each = 5))

both\_tables$celltype[both\_tables$column == 3] <- c("header", rep("nesteddata", 4), "header", rep("data", 20))

both\_tables

# A tibble: 78 x 5

row column value tstep celltype

1 1 1 group a header

2 2 1 1 a id

3 3 1 2 a id

4 4 1 3 a id

5 5 1 4 a id

6 1 2 id a header

7 2 2 1 a 1

8 3 2 2 a 2

9 4 2 3 a 3

10 5 2 4 a 4

# … with 68 more rows

From this, we can produce static versions of the two images which will form the basis for the animation:

base\_plot <- ggplot(both\_tables, aes(column, -row, fill = celltype)) +

geom\_tile(color = "black") +

theme\_void() +

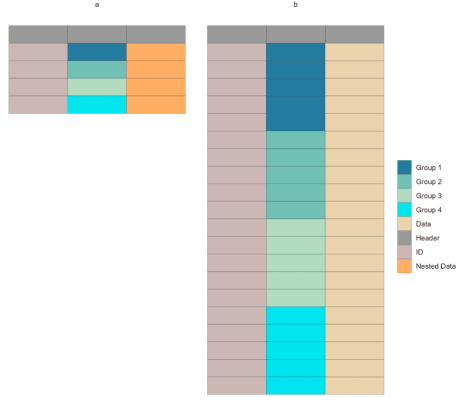
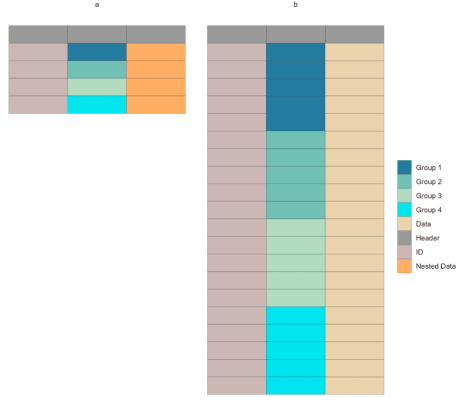
scale\_fill\_manual(values = c("#247ba0","#70c1b3","#b2dbbf","turquoise2", "#ead2ac", "grey60", "mistyrose3", "#ffae63"),

name = "",

labels = c("Group 1", "Group 2", "Group 3", "Group 4", "Data", "Header", "ID", "Nested Data"))

base\_plot +

facet\_wrap(~tstep)

The static plot that will be used to generate the animation

Finally, we use gganimate to build the final animation!

p1 <- base\_plot +

transition\_states(

states = tstep,

transition\_length = 1,

state\_length = 1

) +

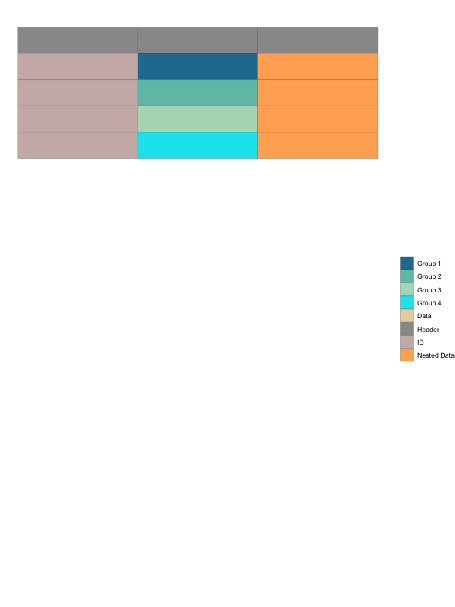
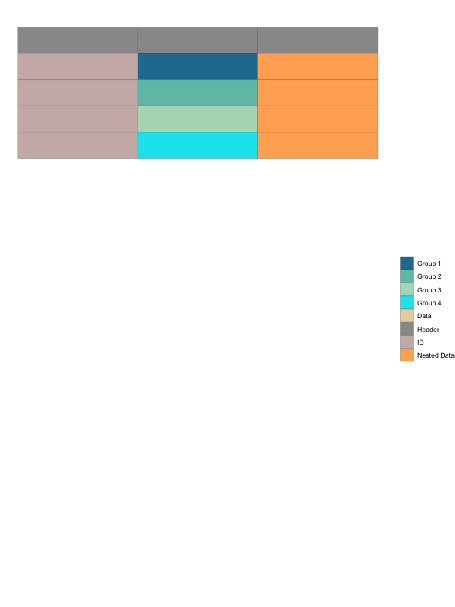
enter\_fade() +

exit\_fade() +

ease\_aes('sine-in-out')

p1\_animate <- animate(p1, height = 800, width = 600, fps = 20, duration = 10)

anim\_save("unnest\_animate.gif")



And there you have it! We hope this was helpful both in learning how to produce data transformation animations, and in terms of learning the unnest() operation itself. If you have any requests for more data transformation animations, please let us know, and be on the look out for future posts in this series!