gganimate hits CRAN –

At the NHS\_R conference back in October, I showed a few ways of building animations using gifski – I also wrote up the.

However – the gganimate file in the repo is now out of date, so use the code down below.

Now at the time of the conference, the method I used to build animations involved a fair bit of slicing, dicing and a bit of purrr to build up individual plots and stitch them together. It was a workaround because gganimate was on github at the time, and installing packages from github onto NHS computers is likely to be highly problematic, to put it mildly.

But, the New Year comes, and despite the fact the country is going completely to pot, there are still things to celebrate because gganimate has now hit CRAN.

So, join me as I take my mind off it all by updating some of the code I used with this wonderful new package.

Fair Warning – regular reader(s) will have seen these plots before.Sorry.However, I still think it’s (vaguely) worthwhile writing this because while there were / are quite a few examples of how to use gganimate with points, I couldn’t find one that shows how to keep the old points on screen while new points are added (Disclaimer – I haven’t put too much effort into additional checking of the latest examples before writing this).

If you are using lines and or segments in your plots, then transition\_reveal keeps the history of the line over time, but it did not work with geom\_point.

Here’s the setup :

source("1\_setup.R")

====================

|  |
| --- |
| library(magrittr) |
|  | library(tidyverse) |
|  | library(lubridate) |
|  | library(scales) |
|  | library(readxl) |
|  | library(gifski) |
|  | library(plotly) |
|  | library(ggExtra) |
|  | library(here) |
|  | #library(officer) |
|  |  |
|  | setwd(here::here()) |
|  |  |
|  |  |
|  | data <- readxl::read\_xlsx("RedGreenGreyDots.xlsx", sheet = 1) #read raw data from Excel |

source("2\_data\_wrangling.R")

|  |
| --- |
| plot\_data <- data %>% |
|  | mutate(Movement15 = lubridate::floor\_date(MovementDateTime,"15 minutes")) %>% |
|  | group\_by(IN\_OUT, Movement\_Type,Staging\_Post,Movement15) %>% |
|  | mutate( counter = if\_else(IN\_OUT == 'IN',1,-1), |
|  | Movement\_15\_SEQNO = cumsum(counter)) %>% |
|  | ungroup() |
|  |  |
|  |  |
|  |  |
|  | # Change "Tranfer In" or "Transfer Out" to "Transfer" |
|  | plot\_data$Movement\_Type <- gsub("Transfer.\*","Transfer",x = plot\_data$Movement\_Type) |
|  |  |
|  | # Set limits for plotting |
|  | lims <- as.POSIXct(strptime(c("2014-09-03 00:00","2014-09-04 01:00") |
|  | , format = "%Y-%m-%d %H:%M")) |
|  |  |
|  | #set colours for points |
|  |  |
|  | plot\_colours <- c("orangered1","royalblue2","grey60") |

library(gganimate)

# Change "Tranfer In" or "Transfer Out" to "Transfer"

plot\_data$Movement\_Type <- gsub("Transfer.\*","Transfer",x = plot\_data$Movement\_Type)

#convert Movement Type to factor, as the first sequence of dots turns red instead of green

plot\_data$Movement\_Type <- as\_factor(plot\_data$Movement\_Type)

#check the levels

levels(plot\_data$Movement\_Type)

plot\_data$Movement\_Type <- forcats::fct\_rev(plot\_data$Movement\_Type)

levels(plot\_data$Movement\_Type)

lims <- as.POSIXct(strptime(c("2014-09-03 00:00","2014-09-04 01:00")

, format = "%Y-%m-%d %H:%M"))

p <- ggplot(plot\_data,aes(Movement15,Movement\_15\_SEQNO, colour = Movement\_Type)) +

geom\_jitter(width = 0.10) +

scale\_colour\_manual(values = plot\_colours) +

facet\_grid(Staging\_Post~.,switch = "y") +

# facet\_wrap(vars(Staging\_Post), ncol = 1) +

scale\_x\_datetime(date\_labels = "%H:%M",date\_breaks = "3 hours",

limits = lims,

timezone = "GMT",

expand = c(0,0)) +

ggtitle(label = "Anytown General Hospital | Wednesday 3rd September 2014 00:00 to 23:59\n",

subtitle = "A&E AND INPATIENT ARRIVALS, DEPARTURES AND TRANSFERS") +

labs(x = NULL, y = NULL,

caption = "@\_johnmackintosh | johnmackintosh.com Source: Neil Pettinger | @kurtstat | kurtosis.co.uk") +

theme\_minimal(base\_size = 11) +

theme(axis.text.y = element\_blank(),

axis.ticks.y = element\_blank()) +

theme(axis.text.x = element\_text(size = 7)) +

theme(axis.ticks.x = element\_blank()) +

theme(legend.position = "bottom") +

theme(panel.grid.minor = element\_blank(),

panel.grid.major = element\_blank()) +

theme(strip.text.y = element\_text(angle = 180)) +

guides(color = guide\_legend("Movement Type")) +

ggExtra::removeGrid()

You will need to hit up my repo to grab the source files, and obviously you’ll want to change the titles and labels.

Now, here is the gganimate bit:

p <- p +

labs(title = 'Time: {frame\_time}', x = 'Time In/ Out', y = NULL) +

transition\_time(MovementDateTime) +

shadow\_mark(past = TRUE, future = FALSE) +

ease\_aes('linear')

p

#OR

#animate(p, fps = 5, width = 1000, height = 600)

#save it if you want to:

#anim\_save("newgganimate.gif")

Transition\_time, by itself, plotted each dot where I expected it, but did not keep the previous points on screen.  
I experimented with some other options (ie. shadow\_wake and shadow\_trail), but it seems

shadow\_mark(past = TRUE, future = FALSE)

is the magic sauce for this particular task.  
I could be completely wrong about that, and maybe there is a better way, in which case, let me know in the comments.

Here is the final gif:

Also at the event, I didn’t get a chance to demonstrate the Shiny app I built.  
It was my first ever attempt, and while admittedly it’s nothing earth shattering, you have to start somewhere right?  
It’s over here, but please lower your expectations.