# Riddler 27th April 2018

*Robert Hickman*

*2018-05-01*

I've been looking for small programming problems to practice on while running experiments. One such source is Fivethirtyeight's Riddler column which posts conundrums weekly. This week one problem focus on one of life's universal problems: how many urinals are needed in any bathroom for all patrons to use it without awkwardness.

Formally this is phrased as:

> Some number, N, of people need to pee, and there is some number, M, of urinals in a row in a men's room. The people always follow a rule for which urinal they select: The first person goes to one on either far end of the row, and the rest try to maximize the number of urinals between them and any other person. So the second person will go on the other far end, the third person in the middle, and so on. They continue to occupy the urinals until one person would have to go directly next to another person, at which point that person decides not to go at all.
>
> What's the minimum number, M, of urinals required to accommodate all the N people at the same time?

Which is perhaps easiest explained using the 'urinal etiquette' meme:



Luckily, this sort of problem is extremely tractable in R to get an estimate of the function for any 1:N people with a few simple loops:

```
#just going to use dplyr and purr
#data.table might be faster but not too worried- verbose programming anyway
library(dplyr)
library(purrr)

#a tip from colin fay
#https://tinyurl.com/colin-fay-purrr
`%not_in%` <- negate(`%in%`)

#start with n = 1 and with a bathroom with 1 urinal
n <- 1
urinal_number <- 1

#create a df with 1 urinal which is unoccupied
urinals_df <- data.frame(urinal = 1:urinal_number,
                         occupied = rep(NA, urinal_number))

#for how many n do we want to solve
while(n < 10) {
  #whilst not all n have a urinal to use loop through
  while(sum(urinals_df$occupied, na.rm = TRUE) < n) {
    #when all are unoccupied take the first urinal
    if(sum(urinals_df$occupied, na.rm = TRUE) == 0) {
      urinals_df$occupied[1] <- 1
    #when all but 1 are unoccupied and there are more than 2 urinals
    #take the opposite end one next
```

```
  } else if(sum(urinals_df$occupied, na.rm = TRUE) == 1 &
            nrow(urinals_df) > 2) {
    urinals_df$occupied[nrow(urinals_df)] <- 1
  #otherwise work out the most isolated free urinal
  } else {
    #get the distances from each urinal to all the occupied urinals
    urinal_distances <- abs(1:nrow(urinals_df) -
                              rep(which(!is.na(urinals_df$occupied)), each = nrow(urinals_df))) %>%
      matrix(., nrow = length(!is.na(urinals_df$occupied)))
    #index
    rownames(urinal_distances) <- 1:nrow(urinal_distances)

    #awkward urinals are ones that are either taken or next to taken urinals
    #don't want to urinate there
    awkward <- c(which(urinal_distances == 1, arr.ind = TRUE)[,1],
                 which(urinal_distances == 0, arr.ind = TRUE)[,1]) %>%
      unique()

    #use %not_in% to find free urinals that aren't in an awkward position
    possible_urinals <- which(rownames(urinal_distances) %not_in% awkward)

    #if only one remains use this urinal
    if(length(possible_urinals) == 1) {
      taken_urinal <- possible_urinals
    } else if(length(possible_urinals) > 1) {
      #for the remaining possible urinals find how far the closest taken urinal is
      #initialise a small nameless func
      closest_distance <- lapply(seq(nrow(urinal_distances)), function(x){
        row <- urinal_distances[x,]
        min <- min(row)
        }) %>%
        unlist()

      #use the urinal that has the maximum distance to its closest urinal
      taken_urinal <- as.numeric(rownames(urinal_distances)[which.max(closest_distance)])
    } else if(length(possible_urinals) == 0) {
      #if there are no free urinals break the loop
      #and add one to the urinal number in the hypothetical bathroom
      urinal_number <- urinal_number + 1
      break
    }
    #occupy the chosen urinal
    urinals_df$occupied[taken_urinal] <- 1
  }
}

#if completed
#i.e. if all users have found a satisfactory free urinal
if(sum(urinals_df$occupied, na.rm = TRUE) == n) {
  if(n == 1) {
    #when n = 1 initial a df to hold the results per n
    results_df <- data.frame(n = 1,
                             urinals_required = urinal_number)
```

```
    } else {
      #otherwise add in a new row to results_df
      results_df <- rbind(results_df, data.frame(n = n, urinals_required = urinal_number))
    }
      #increase n to the next number of patrons
      n <- n + 1
      #start with at least n urinals in the next bathroom
      #this is the bare minimum we would need
      urinal_number <- n
  }

  #reintialise the bathroom to see if it is big enough for the n patrons
  urinals_df <- data.frame(urinal = 1:urinal_number,
                           occupied = rep(NA, urinal_number))

}
```

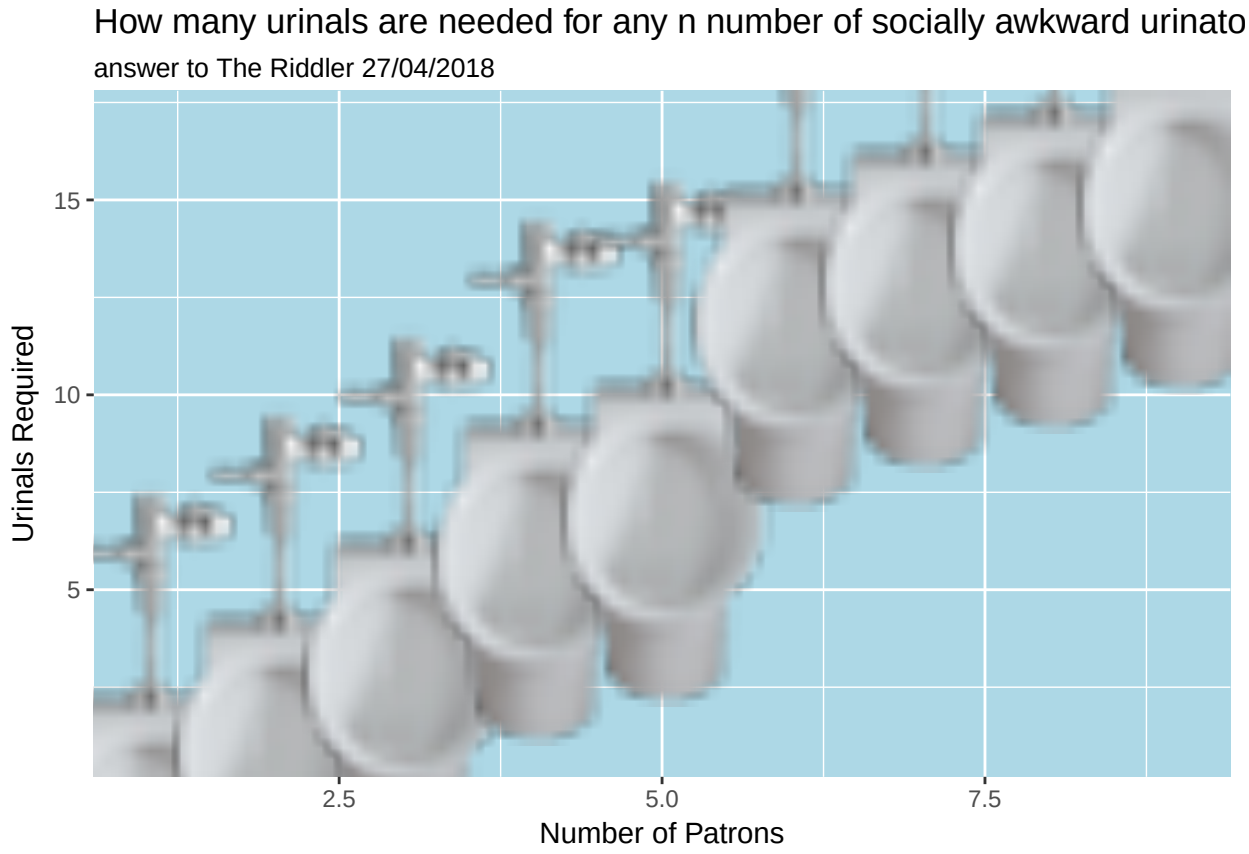
We can then plot this. I decided to add a little flair to the plot using annotate_custom which is a nice little trick to spice up ggplots

```
#load the libraries for plotting
library(ggplot2)
library(png)
library(grid)

#a nice png of a urinal I found online
urinal_image <- readPNG("../../static/img/urinal.png") %>%
  rasterGrob()

#plot the number of urinals needed for any n number of patrons
urinals_plot <- ggplot(data = results_df, aes(x = n, y = urinals_required)) +
  geom_point() +
  #mapply a function to paste the urinal image as an annotation to the graph
  #takes the x and y arguments from the ggplot aesthetic
  mapply(function(x, y, size) {
    annotation_custom(urinal_image,
                      xmin = x - size, xmax = x + size,
                      ymin = y - size, ymax = y + size) },
    x = results_df$n, y = results_df$urinals_required, size = 7) +
  #labelling and etc.
  ylab("Urinals Required") +
  xlab("Number of Patrons") +
  ggtitle("How many urinals are needed for any n number of socially awkward urinators",
          subtitle = "answer to The Riddler 27/04/2018") +
  theme(panel.background = element_rect(fill = 'lightblue'))

urinals_plot
```




which gives a surprisingly complex function! I had assume it would be some simple function of x but clearly something more complex is going on.

Why this happens become clear if you plot out why M urinals are needed for N people. Optimally each person would be separated by 1 urinal, but as the number of urinals increases they become less efficiently packed, with 2 urinals (neither of which can be used without standing next to someone) between each urinating person. This eventually reaches a breaking point and the number of urinals necessary jumps upward.

The formula is known as 'The Pay Phone Packing Sequence' (where users of pay phones don't want to be overheard) and is summarised at https://oeis.org/A185456. The formula itself is:

$f(n) = n + 2^(1 + floor(log(n - 2)))$

That's all for this weeks riddler.

Franz Ferdinand and Sparks - Piss Off