Generative Art with Truchet Tiles in R

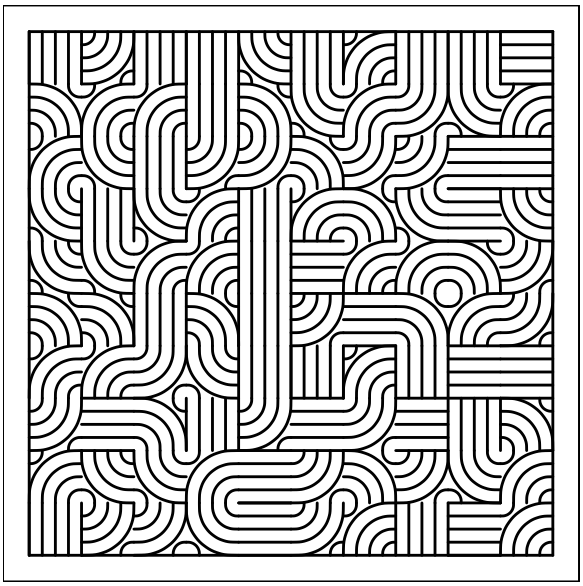
Brian Boyle

05/03/2022

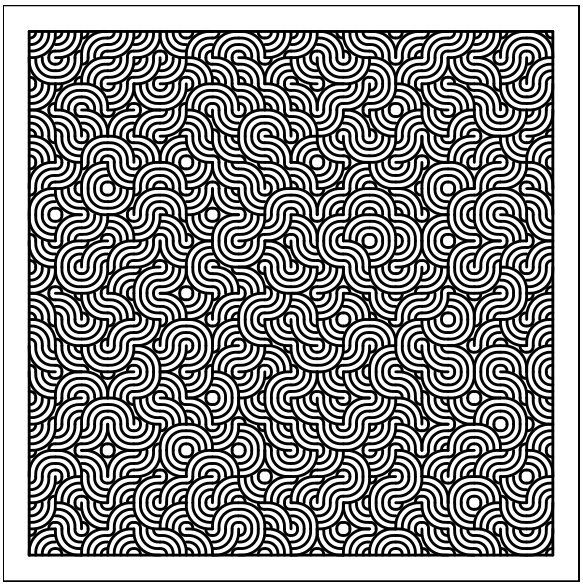
```
## Function arguments
# grid_max    number of tiles on each axis (e.g. 5 = 5x5 grid)
# seed        set seed for random tile selection
# all_tiles   T ==  straight and curved tiles, F == curved tiles only
# line_col    line colour
# line_size   line size
# line_alpha  line alpha value
# bg_col      background colour
# line_end    shape at end of each line (butt, square, round)
```

Plot examples

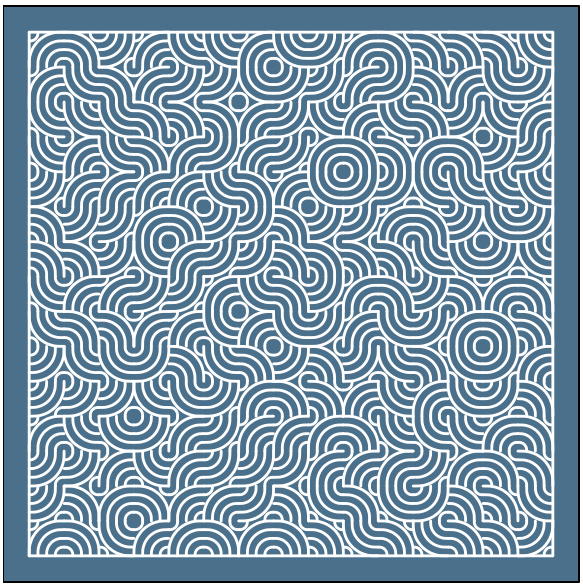
```
## Example plots using the function
# Plot 10x10 grid
plot.truchet(grid_max = 10, seed = 16)
```



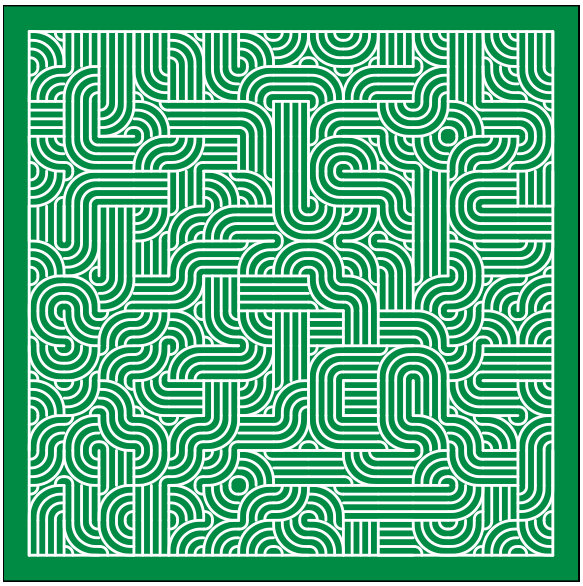
```
# Curved tiles only
plot.truchet(grid_max = 20, seed = 16,
             all_tiles = F)
```



```
# Change colours
plot.truchet(grid_max = 15, seed = 16,
             all_tiles = F,
             line_size = .5,
             line_col = 'white',
             bg_col = 'skyblue4')
```



```
# All tile types
plot.truchet(grid_max = 15, seed = 16,
             all_tiles = T,
             line_size = .5,
             line_col = 'white',
             bg_col = 'springgreen4')
```



Code to create function

```
## Packages used
# install.packages('dplyr')
# install.packages('magrittr')
# install.packages('ggplot2')
library(dplyr)
library(magrittr)
library(ggplot2)


## Custom Function

# Create function to plot grid of truchet tiles
plot.truchet <- function(grid_max, seed, all_tiles = T,
                         line_col = 'black',
                         line_size = 0.5,
                         line_alpha = 1,
                         bg_col = 'white',
                         line_end = 'round'){

  # Set seed for random tile selection
  set.seed(seed)

  # Create set of tiles to sample from. Tiles can be either:
  # 1) Vertical or horizontal straight lines
  # 2) Curved lines, where each tile is rotated 90 degrees

  # If all_tiles is true, include all tiles.
  # If false, only include curved tiles
  if(all_tiles == T){
    tiles <- c('V', 'H', 'C1', 'C2', 'C3', 'C4')
  } else {
    tiles <- c('C1', 'C2', 'C3', 'C4')
  }


  # Create initial dataframe for plotting, where each row is a single line
  dat <-
    data.frame(xs = NA, # X axis value for line start
               xe = NA, # X axis value for line end
               ys = NA, # Y axis value for line start
               ye = NA, # Y axis value for line end
               x = rep(rep(1:grid_max, grid_max),5), # tile X position
               y = rep(rep(1:grid_max, each = grid_max), 5)) %>% # tile Y position
    # unique XY cordinate for each tile
    mutate(xy = paste0(formatC(x,flag=0,width=4), formatC(y,flag=0,width=4))) %>%
    arrange(xy) %>%
    # each tile contains 5 lines, so label these 1:5
    mutate(line_no = rep(1:5, grid_max*grid_max)) %>%
    # for each tile, randomly select a tile type
    mutate(type = rep(sample(tiles, grid_max*grid_max, replace = T), each = 5))
```

```
# Now we input start and end coordinates for each line
# depending on the type of tile that has been selected

# For each tile
for(i in unique(dat$xy)){
  if(any(k <- which(dat$xy == i & dat$type == 'V'))){

    dat$xs[k] = c(dat$x[k][1]*10 - 10,
                  dat$x[k][1]*10 - 7.5,
                  dat$x[k][1]*10 - 5,
                  dat$x[k][1]*10 - 2.5,
                  dat$x[k][1]*10)

    dat$xe[k] = c(dat$x[k][1]*10 - 10,
                  dat$x[k][1]*10 - 7.5,
                  dat$x[k][1]*10 - 5,
                  dat$x[k][1]*10 - 2.5,
                  dat$x[k][1]*10)

    dat$ys[k] = dat$y[k]*10 -10
    dat$ye[k] = dat$y[k]*10

  }
  if(any(k <- which(dat$xy == i & dat$type == 'H'))){

    dat$ys[k] = c(dat$y[k][1]*10 - 10,
                  dat$y[k][1]*10 - 7.5,
                  dat$y[k][1]*10 - 5,
                  dat$y[k][1]*10 - 2.5,
                  dat$y[k][1]*10)

    dat$ye[k] = c(dat$y[k][1]*10 - 10,
                  dat$y[k][1]*10 - 7.5,
                  dat$y[k][1]*10 - 5,
                  dat$y[k][1]*10 - 2.5,
                  dat$y[k][1]*10)

    dat$xs[k] = dat$x[k]*10 -10
    dat$xe[k] = dat$x[k]*10

  }
  if(any(k <- which(dat$xy == i & dat$type == 'C1'))){

    dat$xs[k] = c(dat$x[k][1]*10 - 10,
                  dat$x[k][1]*10 - 7.5,
                  dat$x[k][1]*10 - 5,
                  dat$x[k][1]*10 - 2.5,
                  dat$x[k][1]*10 - 10)

    dat$xe[k] = c(rep(dat$x[k][1]*10, 4),
                  dat$x[k][1]*10 - 7.5)
```

```
    dat$ys[k] = c(rep(dat$y[k][1]*10 -10, 4),
                  dat$y[k][1]*10 -2.5)


    dat$ye[k] = c(dat$y[k][1]*10,
                  dat$y[k][1]*10 - 2.5,
                  dat$y[k][1]*10 - 5,
                  dat$y[k][1]*10 - 7.5,
                  dat$y[k][1]*10)

  }

  if(any(k <- which(dat$xy == i & dat$type == 'C2'))){

    dat$xe[k] = c(dat$x[k][1]*10,
                  dat$x[k][1]*10 - 2.5,
                  dat$x[k][1]*10 - 5,
                  dat$x[k][1]*10 - 7.5,
                  dat$x[k][1]*10)

    dat$xs[k] = c(rep(dat$x[k][1]*10 -10, 4),
                  dat$x[k][1]*10 - 2.5)

    dat$ye[k] = c(rep(dat$y[k][1]*10 -10, 4),
                  dat$y[k][1]*10 -2.5)


    dat$ys[k] = c(dat$y[k][1]*10,
                  dat$y[k][1]*10 - 2.5,
                  dat$y[k][1]*10 - 5,
                  dat$y[k][1]*10 - 7.5,
                  dat$y[k][1]*10)

  }

  if(any(k <- which(dat$xy == i & dat$type == 'C3'))){

    dat$xs[k] = c(rep(dat$x[k][1]*10 -10, 4),
                  dat$x[k][1]*10 - 2.5)

    dat$xe[k] = c(dat$x[k][1]*10 - 7.5,
                  dat$x[k][1]*10 - 5,
                  dat$x[k][1]*10 - 2.5,
                  dat$x[k][1]*10,
                  dat$x[k][1]*10)



    dat$ye[k] = c(rep(dat$y[k][1]*10, 4),
                  dat$y[k][1]*10 - 7.5)


    dat$ys[k] = c(dat$y[k][1]*10 - 2.5,
```

```
                  dat$y[k][1]*10 - 5,
                  dat$y[k][1]*10 - 7.5,
                  dat$y[k][1]*10 - 10,
                  dat$y[k][1]*10 - 10)

  }

  if(any(k <- which(dat$xy == i & dat$type == 'C4'))){

    dat$xs[k] = c(dat$x[k][1]*10 - 10,
                  dat$x[k][1]*10 - 7.5,
                  dat$x[k][1]*10 - 5,
                  dat$x[k][1]*10 - 2.5,
                  dat$x[k][1]*10 - 10)

    dat$xe[k] = c(rep(dat$x[k][1]*10, 4),
                  dat$x[k][1]*10 - 7.5)

    dat$ys[k] = c(rep(dat$y[k][1]*10, 4),
                  dat$y[k][1]*10 - 7.5)


    dat$ye[k] = c(dat$y[k][1]*10 - 10,
                  dat$y[k][1]*10 - 7.5,
                  dat$y[k][1]*10 - 5,
                  dat$y[k][1]*10 - 2.5,
                  dat$y[k][1]*10 - 10)

  }
}



# Now the line values are present, we can create the plot

ggplot() +
  geom_segment(aes(x = c(0, grid_max*10, grid_max*10, 0), # plot grid border
                   y = c(0, 0, grid_max*10, grid_max*10),
                   xend = c(grid_max*10, grid_max*10, 0, 0),
                   yend = c(0, grid_max*10, grid_max*10, 0)),
               colour = line_col,
               size = line_size,
               alpha = line_alpha,
               lineend = line_end) +
  xlim(0,grid_max*10) +
  ylim(0,grid_max*10) +
  # plot straight vertical and horizontal tiles
  geom_segment(data = filter(dat, !grepl('C', type)),
               aes(x = xs,
                   y = ys,
                   xend = xe,
                   yend = ye),
               colour = line_col,
```

```
                size = line_size,
                alpha = line_alpha,
                lineend = line_end) +
    # Plot curved lines for tile type 1 & 2, apart from lines number 5
    geom_curve(data = filter(dat, type %in% c('C1', 'C2'), line_no == 5),
               aes(x = xs,
                   y = ys,
                   xend = xe,
                   yend = ye),
               curvature = .5,
               colour = line_col,
               size = line_size,
               alpha = line_alpha,
               lineend = line_end) +
    # We plot line 5 separately because it curves in the opposite direction
    geom_curve(data = filter(dat, type %in% c('C1', 'C2'), line_no != 5),
               aes(x = xs,
                   y = ys,
                   xend = xe,
                   yend = ye),
               curvature = -.5,
               colour = line_col,
               size = line_size,
               alpha = line_alpha,
               lineend = line_end) +
    # Repeat for the remaining two curved tiles (curve direction reversed)
    geom_curve(data = filter(dat, type %in% c('C3', 'C4'), line_no != 5),
               aes(x = xs,
                   y = ys,
                   xend = xe,
                   yend = ye),
               curvature = .5,
               colour = line_col,
               size = line_size,
               alpha = line_alpha,
               lineend = line_end) +
    geom_curve(data = filter(dat, type %in% c('C3', 'C4'), line_no == 5),
               aes(x = xs,
                   y = ys,
                   xend = xe,
                   yend = ye),
               curvature = -.5,
               colour = line_col,
               size = line_size,
               alpha = line_alpha,
               lineend = line_end) +
    theme_void() +
    # Add in background colour
    theme(panel.background = element_rect(fill = bg_col))

}
```