Code a Valentine!

International Love Data Week

2025-02-11

This document contains instructions for creating images for a Valentine's card using `R` code. We have lots of helper here to walk you through how to run and modify the code. Please raise your hand if you need help!

You should experiment and get creative with all of the options before deciding which one to include on your Valentine's card. Each option includes an example and then a place for you to modify the code and create something that is unique to you!

But First: How to Run and Modify Code!

The following gray box is called an `R` **chunk**.

- To run all of the code in the gray box, click on the green triangle.
- To run a single line of code, put the cursor on the line you want to run and hit Command+Enter (for a Mac) and Control+Enter (for Windows).

```
2 + 2
```

```
[1] 4
```

```
5 - 2
```

```
[1] 3
```

Option 1: Heart of Hearts!

Example

Run the following code to create a heart of hearts. The created image is stored in a file called "option1_example.png". Make sure to look both in the Plot window and the output image as the size of the hearts and text may differ between the two. The output image is what you will be putting in your Valentine's card.

```
# Load packages
library(tidyverse)
library(emojifont)

# Ensure the exported images are the same size in the Plots window
library(showtext)
showtext_opts(dpi = 1000)

# Add more font options
font_add_google("Merriweather")
font_add_google("Lora")

# Generate dataset with heart coordinates
heart_x <- function(angle) {
  x <- (16 * sin(angle) ^ 3) / 17
  return(x - mean(x))
}

heart_y <- function(angle) {
  y <- (13 * cos(angle) - 5 * cos(2 * angle) - 2 * cos(3 * angle) -
          cos(4 * angle)) / 17
  return(y - mean(y))
}

heart_data <- tibble(
  angle = seq(0, 2 * pi, length.out = 50),
  x = heart_x(angle),
  y = heart_y(angle)
) %>%
  mutate(heart = fontawesome("fa-heart"))
```

```r
#########
# Experiment with the following items
#########

# Try changing the colors
color_heart = sample(c("deeppink", "purple3", "red3"),
                     size = 50, replace = TRUE)
color_text = "deeppink3"
# Add your valentine's name
valentines_name = "Dad!"

# Play around with the size of the hearts
heart_size = 18

# Make the hearts more or less opaque
how_opaque = 0.6

# Font
font_writing = "Lora"

# Font size
font_size = 15

#########

# Graph the hearts and name of valentine
ggplot(data = heart_data,
       mapping = aes(x = x, y = y, label = heart)) +
  geom_text(alpha = how_opaque,
            color = color_heart,
            size = heart_size,
            family = 'fontawesome-webfont') +
  theme_void() +
  annotate("text", x = 0, y = 0,
           label = "Happy Valentine's Day,",
           size = font_size,
           color = color_text,
           family = font_writing) +
  annotate("text", x = 0, y = -0.25,
           label = valentines_name,
           size = font_size,
           color = color_text,
```

```
          family = font_writing)
```


```
ggsave("option1_example.png", dpi = 1000, width = 7, height = 5,
       units = "in")
```

Your turn!

Tip: The following website is a good place to find hex codes for colors: https://coolors.co/palettes/trending

```
#########
# Experiment with the following items
#########

# Try changing the colors
color_heart = "red3"
color_text = "deeppink3"

# Add your valentine's name
valentines_name = "Insert Name!"
```

```r
# Play around with the size of the hearts
heart_size = 18

# Make the hearts more or less opaque
how_opaque = 0.6

# Change the font
# Add more font options from Google Fonts
font_add_google("Lora")
font_writing = "Lora"

# Font size
font_size = 15

#########

# Graph the hearts and name of valentine
ggplot(data = heart_data,
       mapping = aes(x = x, y = y, label = heart)) +
  geom_text(alpha = how_opaque,
            color = color_heart,
            size = heart_size,
            family = 'fontawesome-webfont') +
  theme_void() +
  annotate("text", x = 0, y = 0,
           label = "Happy Valentine's Day,",
           size = font_size,
           color = color_text,
           family = font_writing) +
  annotate("text", x = 0, y = -0.25,
           label = valentines_name,
           size = font_size,
           color = color_text,
           family = font_writing)
```


```
ggsave("option1_example.png", dpi = 1000, width = 7, height = 5,
       units = "in")
```

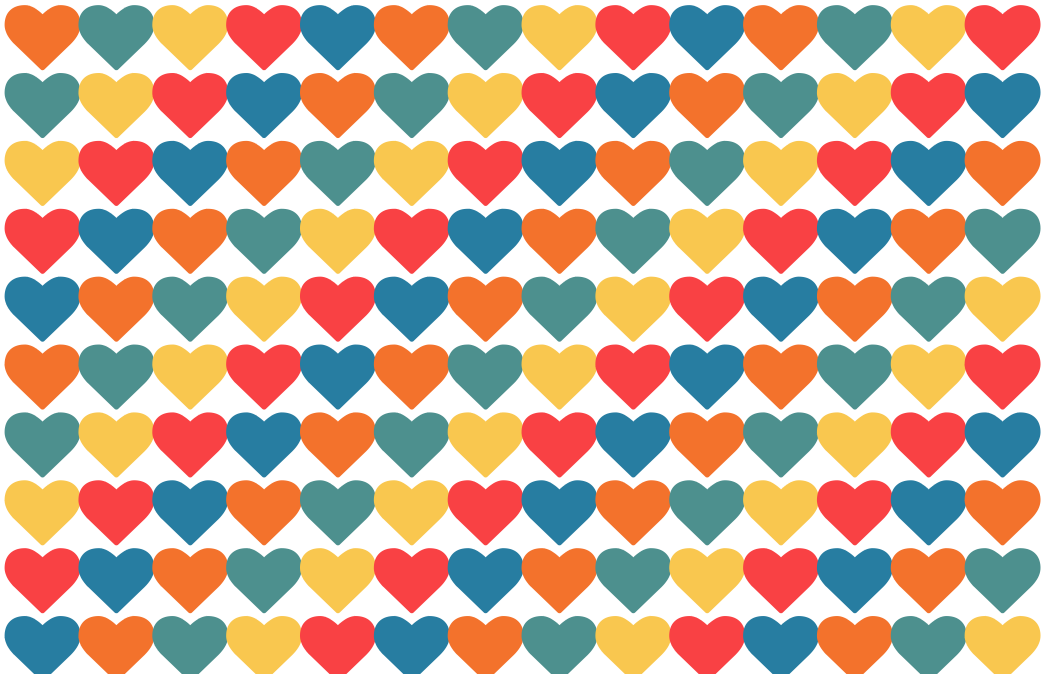
Option 2: Many Hearts: Two Ways

Example

Run the following code to create a wall of hearts, two ways.

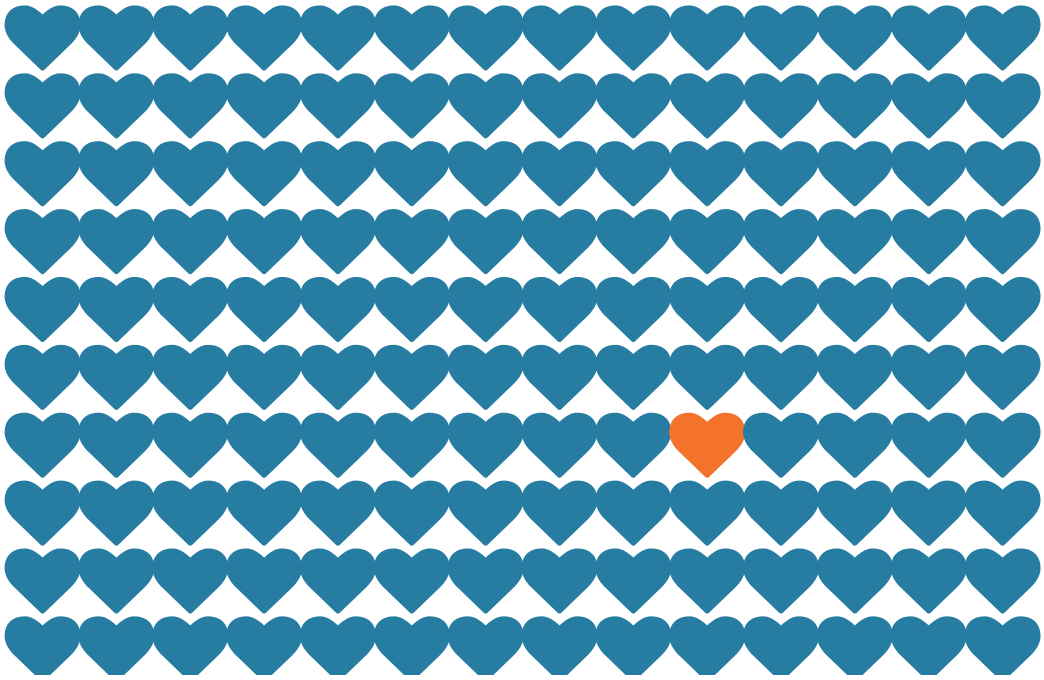
```
# Create grid for hearts
hearts <- expand.grid(x = 1:14, y = 1:10) %>%
  mutate(heart = fontawesome("fa-heart")) %>%
  mutate(special = if_else(x == 10 & y == 4, "yes", "no")) %>%
  mutate(cycle = rep(letters[1:5], 28))


# Graph the hearts with a fun pattern
ggplot(data = hearts,
       mapping = aes(x = x, y = y,
                     label = heart,
                     color = cycle)) +
```

```
  geom_text(size = 10,
            family = 'fontawesome-webfont') +
  scale_color_manual(values = c("#277da1", "#f3722c",
                                "#4d908e", "#f9c74f",
                                "#f94144")) +
  guides(color = "none") +
  theme_void()
```



```
ggsave("option2_example.png", dpi = 1000, width = 7, height = 5,
       units = "in")
```

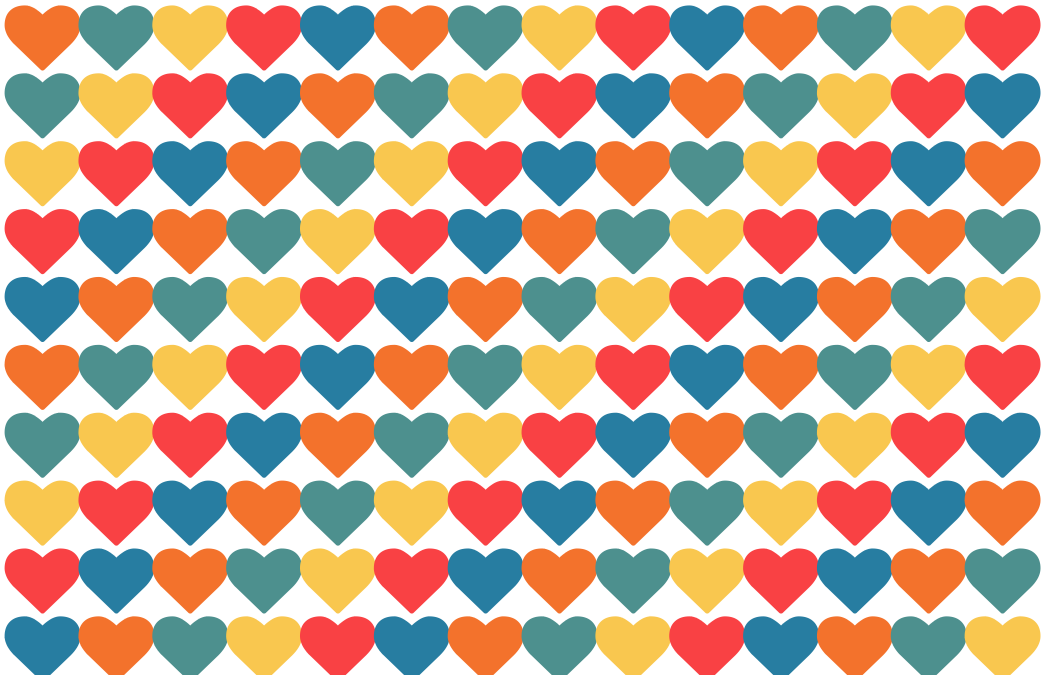
```
# Graph the hearts with one special heart
ggplot(data = hearts,
       mapping = aes(x = x, y = y,
                     label = heart,
                     color = special)) +
  geom_text(size = 10,
            family = 'fontawesome-webfont') +
  scale_color_manual(values = c("#277da1", "#f3722c")) +
  guides(color = "none") +
  theme_void()
```



```
ggsave("option3_example.png", dpi = 1000, width = 7, height = 5,
       units = "in")
```

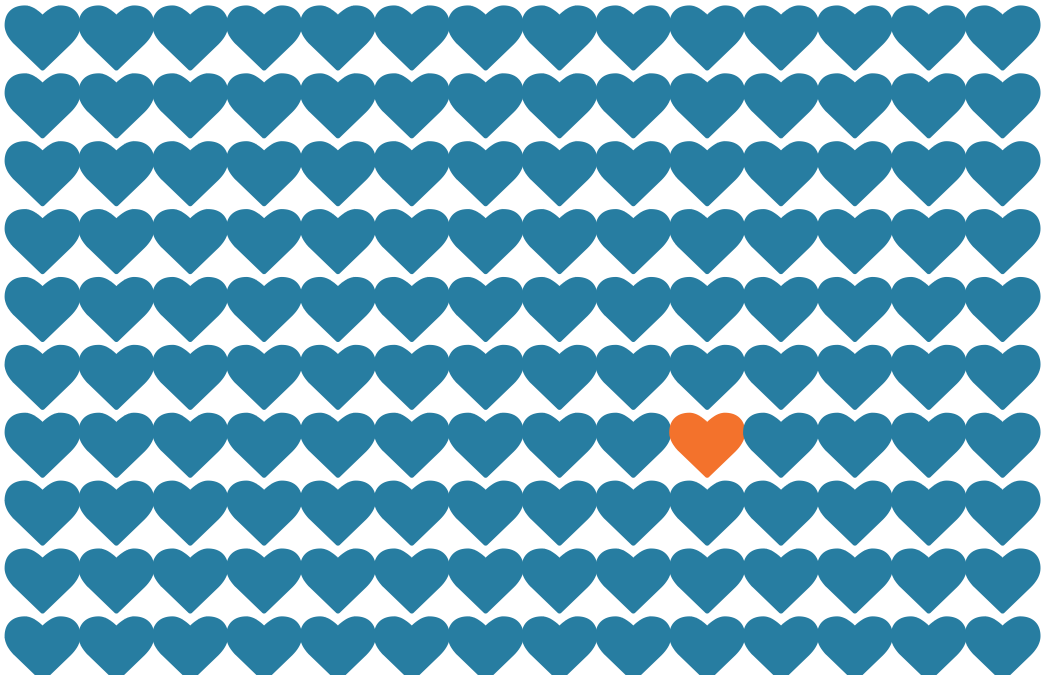
Your turn!

```
############
# Pick 5 new colors
colors = c("#277da1", "#f3722c", "#4d908e", "#f9c74f", "#f94144")
############


# Graph the hearts with a fun pattern
ggplot(data = hearts,
       mapping = aes(x = x, y = y,
                     label = heart,
                     color = cycle)) +
  geom_text(size = 10,
            family = 'fontawesome-webfont') +
  scale_color_manual(values = colors) +
  guides(color = "none") +
  theme_void()
```



```
ggsave("option2_example.png", dpi = 1000, width = 7, height = 5,
       units = "in")
```

```
############
# Pick 2 new colors
colors = c("#277da1", "#f3722c")
############


# Graph the hearts with one special heart
ggplot(data = hearts,
       mapping = aes(x = x, y = y,
                     label = heart,
                     color = special)) +
  geom_text(size = 10,
            family = 'fontawesome-webfont') +
  scale_color_manual(values = colors) +
  guides(color = "none") +
  theme_void()
```



```
ggsave("option3_example.png", dpi = 1000, width = 7, height = 5,
       units = "in")
```

Option 3: Grow a Forest

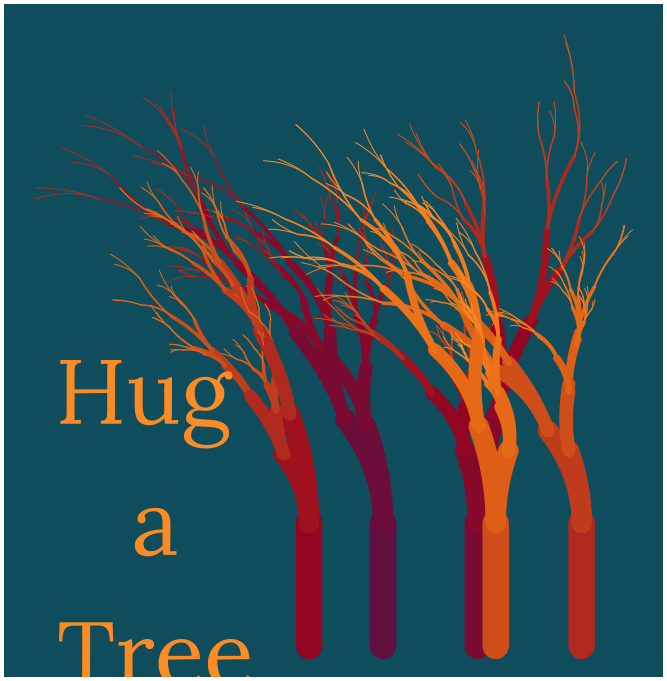
The next two options come from Danielle Navarro, a data scientist and generative artist. She licenses all her code for creating generative art in R using creative common licenses so that others can use her code to create their own art!

Example

Run the following code to grow your own forest.

```
# Load Danielle's package
library(flametree)

# Pick some colors
shades = c("#5F0F40", "#9A031E", "#E36414", "#FB8B24")

# Grow the trees
```

```
dat = flametree_grow(seed = 8,
                     time = 6,
                     trees = 5)

# Draw the trees
dat %>%
  flametree_plot(
    background = "#0F4C5C",
    palette = shades
  ) +
  annotate("text", x = -2.5, y = 1,
           label = "Hug\n a\n Tree",
           size = 12, color = "#FB8B24",
           family = font_writing)
```

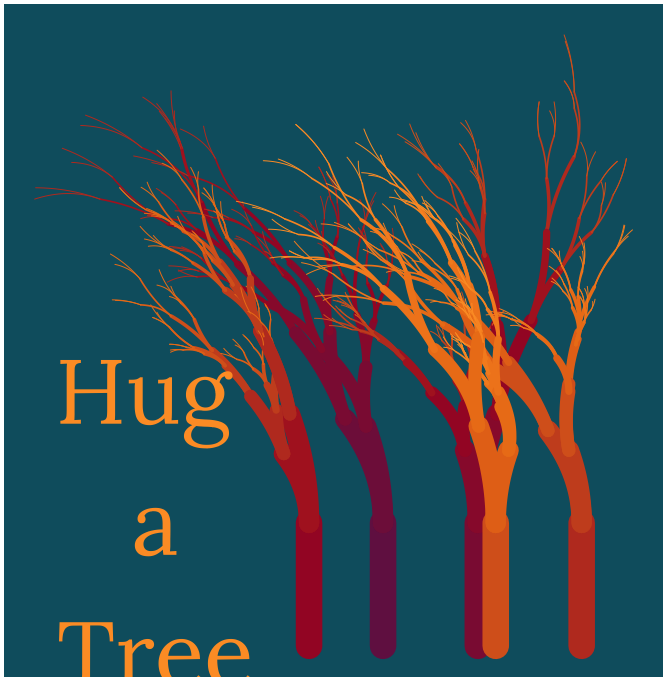



```
ggsave("option4_example.png", dpi = 500)
```

Your turn!

```
#########
# Change the following
#########

# Determine how many trees you want
number_trees = 5

# Pick a random number for growing the trees
# Impacts their location
number_random = 8

# Determine how much growing (larger number = more growth)
number_growing = 6

# Pick some new colors
shades = c("#5F0F40", "#9A031E", "#E36414", "#FB8B24")

# Pick a background color
background_color = "#0F4C5C"

# Play around with the style of trees
# Options: "plain", "voronoi", "wisp", "nativeflora", "minimal"
tree_style = "plain"

# Add a message
# Tips: "" = no message
# \n = new line
message = "Hug\n a\n Tree"

# Pick the size of the message
message_size = 12

# Pick the x location and y location of the message
message_x = -2.5
message_y = 1

# Pick the color of the message
message_color = "#FB8B24"

# Change the font
# Add more font options from Google Fonts
font_add_google("Lora")
```

```
message_font = "Lora"

#########

# Grow the trees
dat <- flametree_grow(seed = number_random,
                      time = number_growing,
                      trees = number_trees)

# draw the plot
dat %>%
  flametree_plot(
    background = background_color,
    palette = shades,
    style = tree_style
  ) +
  annotate("text",
           label = message,
           size = message_size,
           x = message_x, y = message_y,
           color = message_color,
           family = message_font)
```



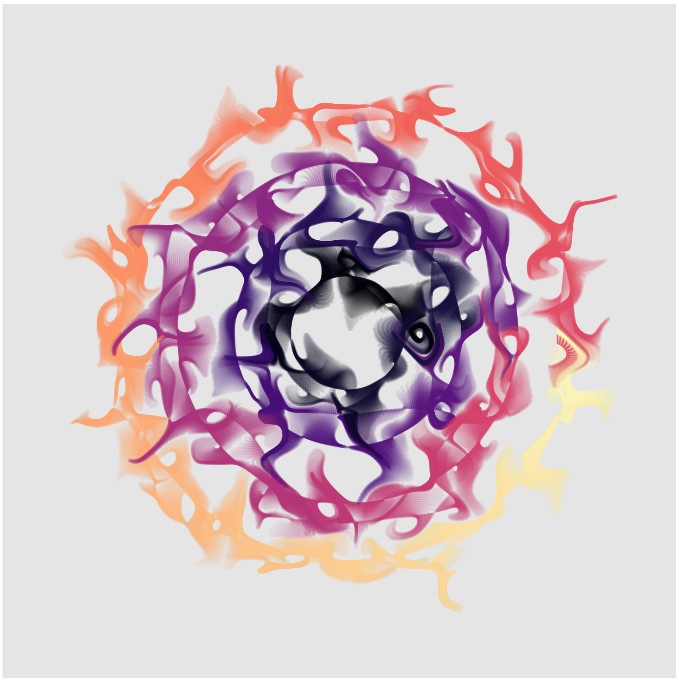
```
ggsave("option4_example.png", dpi = 500)
```

Option 4: Create Art

The `jasmines` package allows you to create interesting shapes that have some stochastic elements.

Example

```
# Load package
library(jasmines)

# Create flowy rings
use_seed(4) %>%
  scene_discs(
    rings = 4, points = 5000, size = 6
  ) %>%
  mutate(ind = 1:n()) %>%
  unfold_warp(
    iterations = 1,
    scale = .5,
    output = "layer"
  ) %>%
  unfold_tempest(
    iterations = 20,
    scale = .011
  ) %>%
  style_ribbon(
    palette = "magma",
    colour = "ind",
    alpha = c(.5,.1),
    background = "gray90"
  )
```

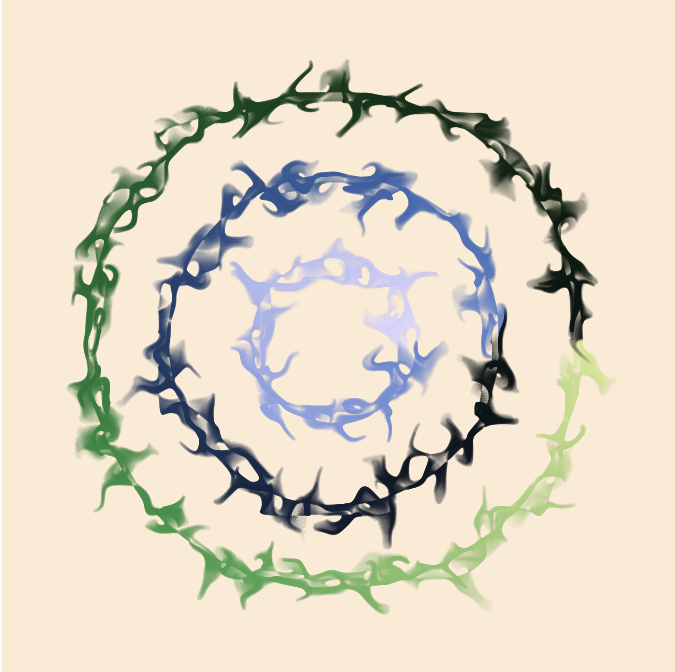


```
ggsave("option5_example.png", dpi = 500)
```

Your turn!

```
#########
# Change the following
#########

# Pick the number of rings
number_rings = 3

# Pick the number of interior points
number_points = 5000

# Pick a diameter for the outermost ring
diameter = 10

# Pick a random number, it impacts the distortions in the rings
number_random = 8

# Pick a background color
```

```
# Great options: "antiquewhite", "wheat", "black"
background_color = "antiquewhite"

# Pick a color palette
# Some of the options: "magma", "vik", "acton", "bamako", "batlow", "berlin", "bilbao",
# "broc", "buda", "cork", "davos", "devon", "grayC", "hawaii",
# "imola", "lajolla", "lapaz", "lisbon", "nuuk", "oleron",
# "oslo", "roma", "tofino", "tokyo", "turku", "vik"
color_palette = "tofino"

########

# Create flowy rings
use_seed(number_random) %>%
  scene_discs(
    rings = number_rings,
    points = number_points,
    size = 10
  ) %>%
  mutate(ind = 1:n()) %>%
  unfold_warp(
    iterations = 1,
    scale = 0.5,
    output = "layer"
  ) %>%
  unfold_tempest(
    iterations = 20,
    scale = .011
  ) %>%
  style_ribbon(
    palette = color_palette,
    colour = "ind",
    alpha = c(.5, .1),
    background = background_color
  )
```



```
ggsave("option5_example.png", dpi = 500)
```

There are lots of other shapes and distortions you can play around with in the `jasmines` package: https://jasmines.djnavarro.net/index.html.