1. Let's create some aRt!

   (a) Install the `aRtsy` package. Provide the code in an R chunk that does not run. You only need to install it one time.
   Solution:

   ```
   install.packages("aRtsy")
   ```

   (b) Load the `aRtsy` package. Provide the code in an R chunk that does run. We need to load the library each time it is run.
   Solution:

   ```
   library(aRtsy)
   ```

   (c) Running `demo("aRtsy")` or `vignette("aRtsy")` don't return any helpful demos or tutorials. However, if you run `help("aRtsy")` you will find a link to a tutorial. Recreate the first figure they make using `canvas_collatz()`. Make sure to update the caption.
   Solution:

   ```
   set.seed(1)
   canvas_collatz(colors = colorPalette("tuscany3"))
   ```

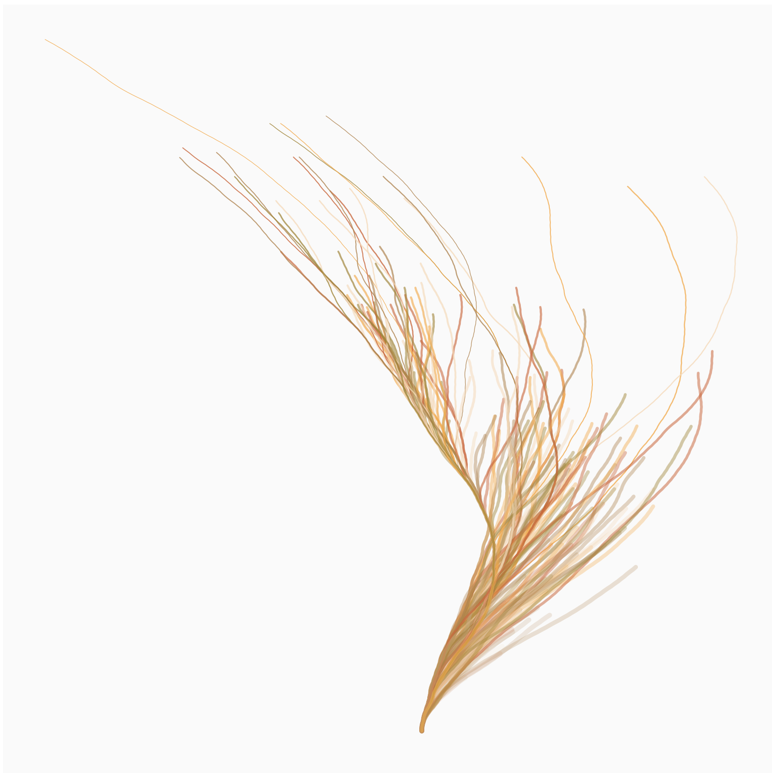

Figure 1: The original copied collatz plot

   (d) Change the randomization seed to 1313, which will change the random numbers generated to create the plot. Can you see the difference? Make sure to update the caption.
   Solution:

   ```
   set.seed(1313)
   canvas_collatz(colors = colorPalette("tuscany3"))
   ```

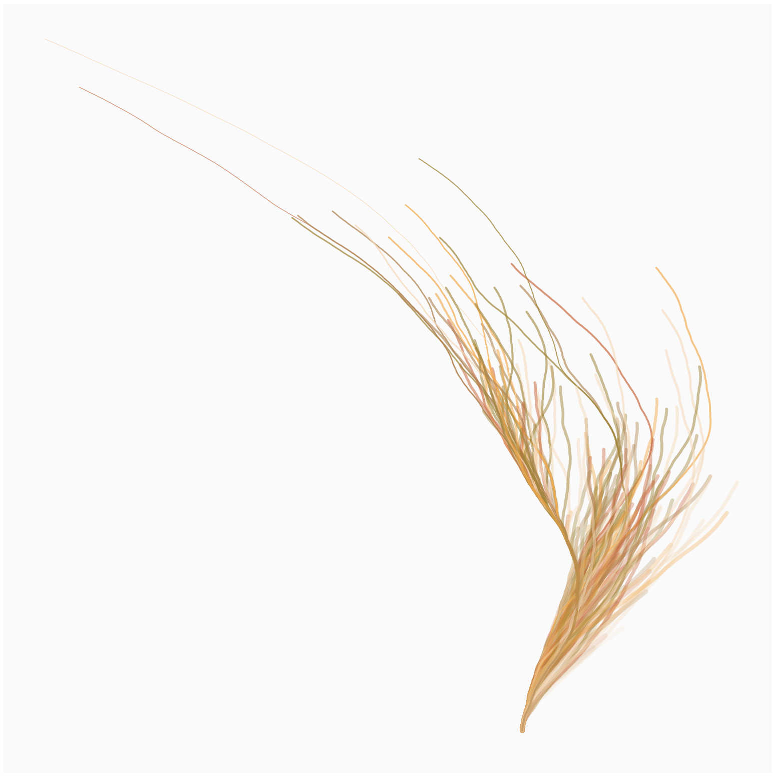

Figure 2: Slightly tweaked seeded plot

(e) Now, create a new Collatz conjecture plot by specifying the following arguments. Note you will find the help file for the `canvas_collatz()` function to be rather helpful. Make sure to update the caption.

- Use the `vrolik4` color palette. Note you can find other by running `?colorPalette` in the console.
- Make the background grey. Note a hexcode for grey is `#dbdbdb`.
- Specify that there should be 72 strands.
- Specify the angle used for bending the sequence for odd numbers as -0.05.
- Specify the angle used for bending the sequence for even numbers as 0.0145 (note this is the default).

Solution:

```r
set.seed(1)
canvas_collatz(colors = colorPalette("vrolik4"), background = "#dbdbdb", n= 72, angle.odd = -0.05)
```

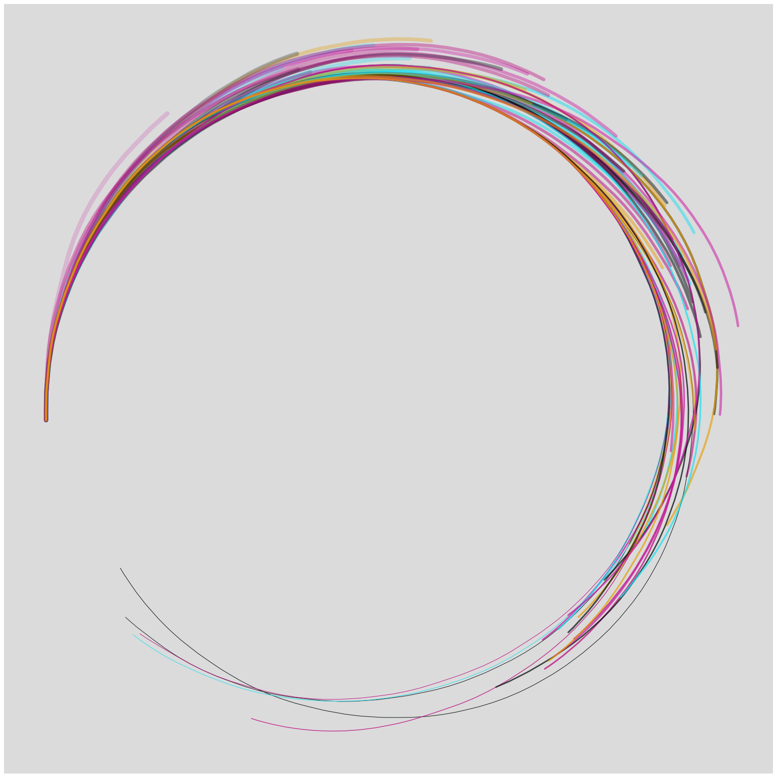

Figure 3: Circular beauty

(f) Make another plot using the tutorial – feel free to be creative here! Note that I leave creating the R chunk and figure environment to you here. Make sure that your code is well-formatted and your plot is appropriately scaled.
Solution:

```
set.seed(1)
canvas_collatz(colors = colorPalette("blossom"), background = "#000000", n= 300, angle.odd = -0.3)
```

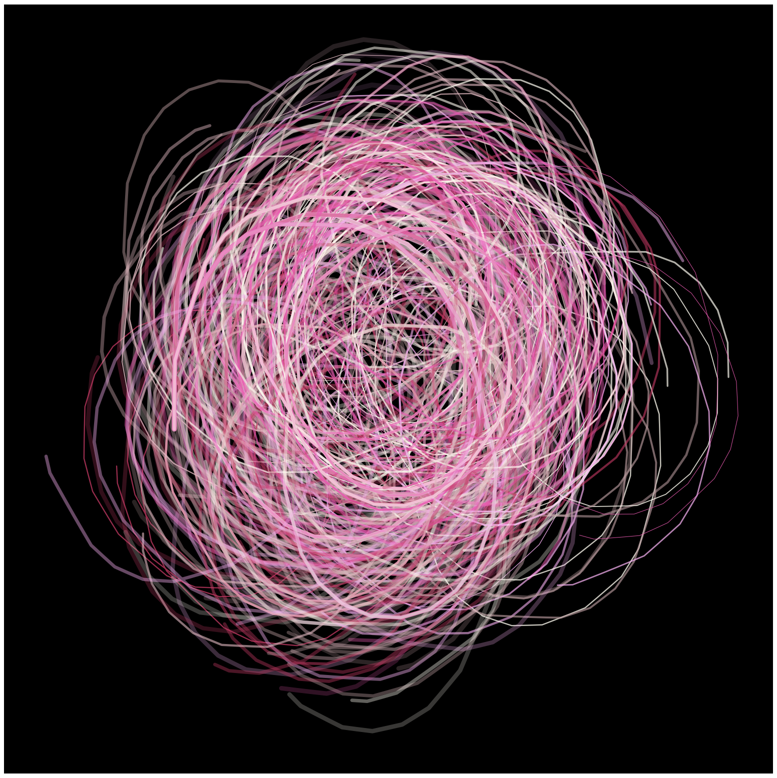

Figure 4: Cotton candy plate

(g) Use `citation()` to get the BiBTeX citation for the `aRtsy` package and use `\citep{}` to add a parenthetical citation to the end of the sentence below.
Solution: We created the generative art in Question 1 using the `aRtsy` package for `R` (Derks, 2024).

References

Derks, K. (2024). *aRtsy: Generative Art with 'ggplot2'*. R package version 1.0.0.

2. Suppose we wanted to solve $2^{x+1} + 2^{x-1} = 40$ for x. While this is a pretty straightforward algebra problem, it's useful for demonstrating the use of objects in R.

   (a) Create a numeric vector containing the integers from 0 to 10 inclusive. Hint – the solution to this problem is one of these values.
   Solution:

```
numbers <- 1:10
numbers

##  [1]  1  2  3  4  5  6  7  8  9 10
```

   (b) Complete the algebra to compute $2^{x+1} + 2^{x-1}$ for each value in the numerical vector created in step 1. Make sure to save the result to a new numeric vector.
   Solution:

```
computed <- rep(x =NA, times = length(numbers))
for (num in 1:length(numbers)){
  current <- numbers[num]
  computed[num] <- 2^(current+1) + 2^(current-1)
}
computed

##  [1]    5   10   20   40   80  160  320  640 1280 2560
```

   (c) Use the which() function to ask which result is 40.
   Solution:

```
which(computed == 40)

## [1] 4
```

   (d) What is the solution? That is, what value of x yields $2^{x+1} + 2^{x-1} = 40$?
   Solution: The answer is 4

   (e) Explain why this approach wouldn't work for something like $3^{x+2} + 5(3^x) = 84$ where the solution is $x \approx 1.6309$.
   Solution: This approach wouldn't work well because the vectors are created by incrementing by a specific value. Getting a precise number like 1.6309 would take a long time to narrow down what values to put into the values. Doing 10,000 computations from 1 to 2 to find this approximate value would be super inefficient.