# Art in Many Forms

Creative Coding in R

Thomas Wise; 6664202

## Introduction

As someone passionate about the arts in all their forms, be them live, self-reflective or simply the doodles you draw whilst bored on the train home. As such, I have been slowly building and developing my own ability to create (and teach) artist skills through using code. As such, I will use this markup script to document my progress and demonstrate these skills to you, the humble reader.

## Packages

For those familiar with *R*, you will know the pleasure that is the packages we use within everyday analysis and development. Below you will find those required for the reproduction of these images, examples and more!

```
## Core Packages
  library(tidyverse)      ## In particular ggplot2
  library(RColorBrewer)   ## For colour experimentation
  library(Rcpp)           ## For CPP intergration


## Additional Packages
  # Those used within publication of this pdf
  library(kableExtra)
  library(rmarkdown)
```

## Simulations

For the creation of a basic art piece, data is required. Although data is all around us, due to the Internet of Everything (IoE), sometimes it is easiest to simulate and mathematically generate this data.

For this, we can simply simulate 100 data points from a standard normal distribution.

```
## Set a unique seed (I personally like to use random dates)
## This allows the simulated data to be reproducible

set.seed(160920) ## Today's date

## For personal use let us set our population, mean and sd
mean.val = 0
```

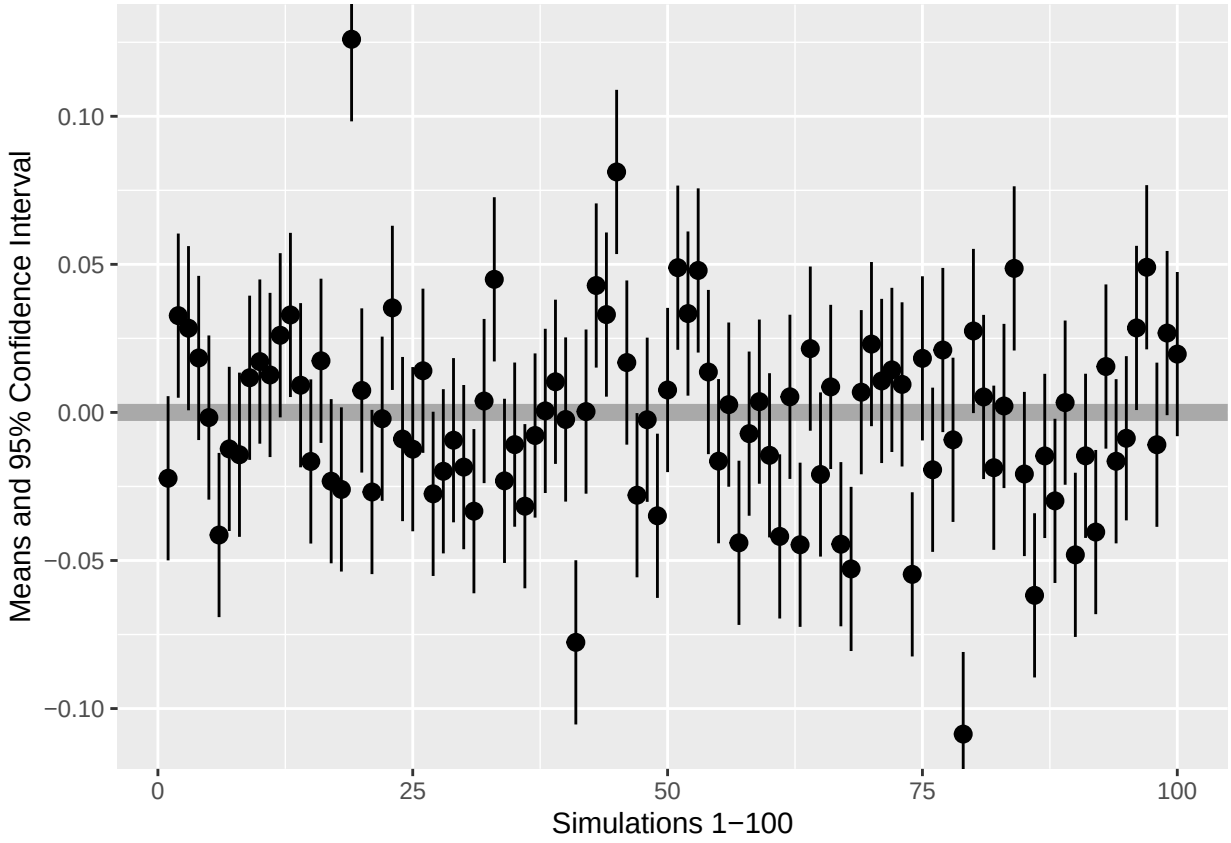
```r
sd.val = 1
pop.val = 5000 #n

## Generate an initial test data set (to confirm everything works)
samp.norm.dat <- rnorm(n = pop.val, mean = mean.val, sd = sd.val)

## Lets check out the top 6 values
head(samp.norm.dat)
```

```
## [1] -0.3450794 -1.4032334 -0.8702486  0.8956729 -0.6775004  2.1820632
```

```r
## Lets make things a little more complex to add some flavour to our outcome
samp.samp.norm.dat <- replicate(pop.val,
                                rnorm(1000, mean = mean.val, sd = sd.val))

## From this, we have produced 1000 samples from 100 different normal distirbutions
## We are also able to produce various sample statistics from these

## Absolute Bias
  ## Determined as: The Sample Mean - True Mean
  ## In this case this can be calculated as the Column Mean - 0 (given in the initial case)

colmean.samp.dat <- colMeans(samp.samp.norm.dat)
ab.samp.dat <- colmean.samp.dat - mean.val

## Standard Error
  ## Determined as True SD / sqrt(N)

se.samp.dat <- sd.val / sqrt((pop.val))

## 95% Confidence Interval
  ## For this, we can calculate the inital boundaries
  df.val = pop.val - 1
  bound.samp.dat <- (qt(0.975, df = df.val) * se.samp.dat)

  upper.b.samp.dat <- colmean.samp.dat + bound.samp.dat
  lower.b.samp.dat <- colmean.samp.dat - bound.samp.dat

## This can be integrated into a plot
  ## Firstly let us join them into one data frame
  samp.df <- as.data.frame(cbind(colmean.samp.dat,
                                 ab.samp.dat,
                                 rep_len(se.samp.dat, length.out = pop.val),
                                 upper.b.samp.dat,
                                 lower.b.samp.dat))
  colnames(samp.df) <- c("Mean", "Bias", "Std.Error", "Upper", "Lower")

  ## Then these can be created into a visual plot
  ggplot(data = samp.df,
         mapping = aes(x = 1:pop.val, y = Mean)) +
    geom_hline(aes(yintercept = mean.val), colour = "darkgrey", size = 3) +
    geom_pointrange(ymax = samp.df$Upper, ymin = samp.df$Lower) +
    xlim(1, 100) +
```

```
    labs(x = "Simulations 1-100",
         y = "Means and 95% Confidence Interval")
```

```
## Warning: Removed 4900 rows containing missing values (geom_pointrange).
```




Although this simulation is interesting, we can kick it up a notch in order to be able to create artistic designs from it.

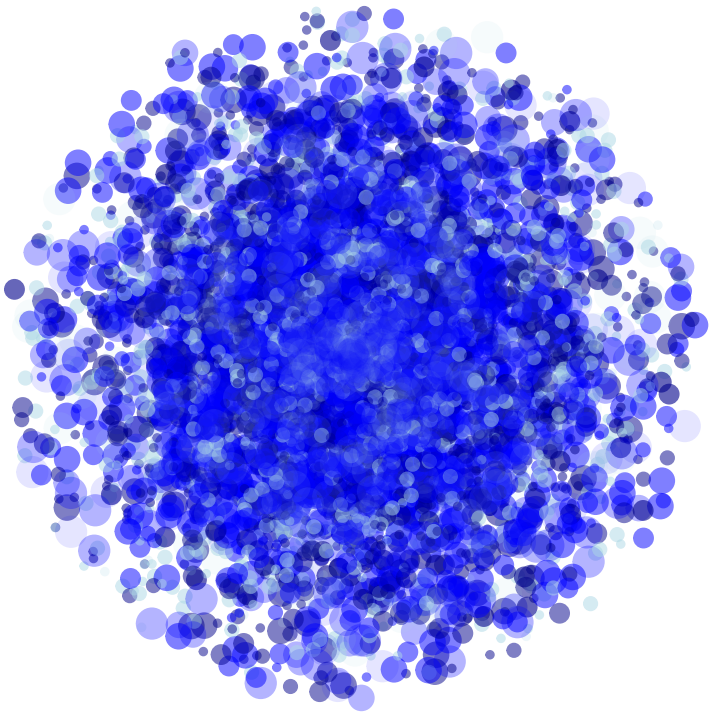
## From Simulations to Art

So from our previous examination, we generated a normal distribution based around the mean value of 0, with an Standard Deviation of 1. This when visualized as a density on a Cartesian coordinate system highlights that the most common values fall around the mean. This can easily be converted to the polar based coordinate system (denoted through the ggplot function \textit{coord_polar()}), and present high *clustering* around the mean before reducing in frequency as it moves away (both to the upper and lower bounds of the confidence intervals as previously discussed). This has the potential for some beautiful biological-esc patterns, such as water drops or waves. Wherein, in enough points are generated it could be seen at a high level of detail.

If we therefore use the same generation structure as before, wherein, we generate 25 different simulations of this normal distribution using the means and standard deviation of 0 & 1 respectively. We are able to generate data which clusters around this mean of 0. Although this can be changed we can start at this point.

```r
## Unit Specification
pop2.val = 360
mean.2.val = 0
sd.2.val = 1

## Data Generation
samp.norm.2.dat <- replicate(25,
                             rnorm(pop2.val, mean = mean.2.val, sd = sd.2.val))
samp.norm.2.dat <- as.data.frame(samp.norm.2.dat)

## Data Plotting and Graphing
ggplot(data = samp.norm.2.dat) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V1), colour = "darkblue",
             alpha = 0.5, size = 1) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V2), colour = "blue",
             alpha = 0.5, size = 3) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V3), colour = "lightblue",
             alpha = 0.5, size = 2) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V4), colour = "darkblue",
             alpha = 0.5, size = 4) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V5), colour = "blue",
             alpha = 0.5, size = 2) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V6), colour = "lightblue",
             alpha = 0.5, size = 1) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V7), colour = "darkblue",
             alpha = 0.5, size = 2) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V8), colour = "blue",
             alpha = 0.5, size = 4) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V9), colour = "lightblue",
             alpha = 0.5, size = 2) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V10), colour = "darkblue",
             alpha = 0.5, size = 1) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V11), colour = "blue",
             alpha = 0.3, size = 4) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V12), colour = "lightblue",
             alpha = 0.5, size = 2) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V13), colour = "darkblue",
             alpha = 0.6, size = 3) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V14), colour = "blue",
             alpha = 0.5, size = 4) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V15), colour = "lightblue",
             alpha = 0.2, size = 1) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V16), colour = "darkblue",
             alpha = 0.5, size = 2) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V17), colour = "blue",
             alpha = 0.5, size = 1) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V18), colour = "lightblue",
             alpha = 0.5, size = 1) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V19), colour = "darkblue",
             alpha = 0.5, size = 3) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V20), colour = "blue",
             alpha = 0.3, size = 5) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V21), colour = "lightblue",
```

```
            alpha = 0.5, size = 2) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V22), colour = "darkblue",
            alpha = 0.5, size = 1) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V23), colour = "blue",
            alpha = 0.5, size = 3) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V24), colour = "lightblue",
            alpha = 0.1, size = 5) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V25), colour = "blue",
            alpha = 0.1, size = 5) +
  coord_polar() +
  ylim(-1, 2) +
  theme_void() +
  theme(legend.position = "none")
```

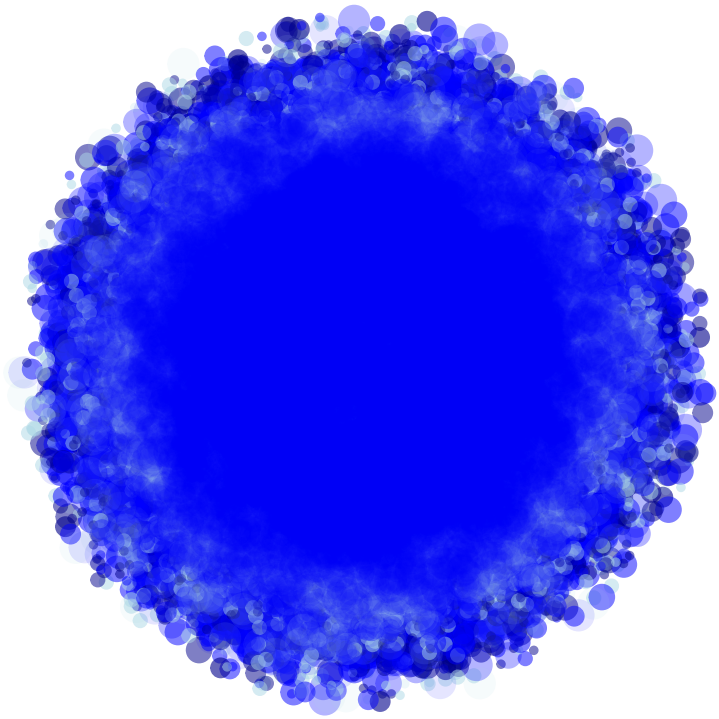


From this, it can be observed that this actively generates a water like graph. Personally though I think we can go more complex. As such, the following adaptions can be made.

```
## Copy and Pasted code from above
## Unit Specification
## Note this file was originally run with 1 million data points...
  # And it broke the pdf and rmd file.
pop2.val = 10000
mean.2.val = 10
sd.2.val = 12

## Data Generation
samp.norm.2.dat <- replicate(25,
                             rnorm(pop2.val, mean = mean.2.val, sd = sd.2.val))
samp.norm.2.dat <- as.data.frame(samp.norm.2.dat)
```

```r
## Data Plotting and Graphing
ggplot(data = samp.norm.2.dat) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V1), colour = "darkblue",
             alpha = 0.5, size = 1) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V2), colour = "blue",
             alpha = 0.5, size = 3) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V3), colour = "lightblue",
             alpha = 0.5, size = 2) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V4), colour = "darkblue",
             alpha = 0.5, size = 4) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V5), colour = "blue",
             alpha = 0.5, size = 2) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V6), colour = "lightblue",
             alpha = 0.5, size = 1) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V7), colour = "darkblue",
             alpha = 0.5, size = 2) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V8), colour = "blue",
             alpha = 0.5, size = 4) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V9), colour = "lightblue",
             alpha = 0.5, size = 2) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V10), colour = "darkblue",
             alpha = 0.5, size = 1) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V11), colour = "blue",
             alpha = 0.3, size = 4) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V12), colour = "lightblue",
             alpha = 0.5, size = 2) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V13), colour = "darkblue",
             alpha = 0.6, size = 3) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V14), colour = "blue",
             alpha = 0.5, size = 4) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V15), colour = "lightblue",
             alpha = 0.2, size = 1) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V16), colour = "darkblue",
             alpha = 0.5, size = 2) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V17), colour = "blue",
             alpha = 0.5, size = 1) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V18), colour = "lightblue",
             alpha = 0.5, size = 1) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V19), colour = "darkblue",
             alpha = 0.5, size = 3) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V20), colour = "blue",
             alpha = 0.3, size = 5) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V21), colour = "lightblue",
             alpha = 0.5, size = 2) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V22), colour = "darkblue",
             alpha = 0.5, size = 1) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V23), colour = "blue",
             alpha = 0.5, size = 3) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V24), colour = "lightblue",
             alpha = 0.1, size = 5) +
  geom_point(mapping = aes(x = 1:pop2.val, y = V25), colour = "blue",
             alpha = 0.1, size = 5) +
  coord_polar() +
```

```
  ylim(-20, 50) +
  theme_void() +
  theme(legend.position = "none")
```



This from this example, creates a much more highly defined graphic, which creates almost a sponged-esc art style. This can be furthered as different techniques are used.