Working in the R Ecosystem
Building Applications & Content for Your Gateway

Derrick Kearney
dsk@rstudio.com

Slides and Examples
https://github.com/dskard/sgci-201805
Background
Background

RStudio IDE

RStudio, Shiny, and R Markdown are trademarks of RStudio, Inc. Images from rstudio.com
Background

RStudio IDE

Open Source

RStudio, Shiny, and R Markdown are trademarks of RStudio, Inc. Images from rstudio.com
Background

RStudio IDE

Open Source

- tidyverse
- ggplot2
- Shiny
- rmarkdown
- dplyr
- plumber

Enterprise-Ready Professional Products

- RStudio Server Pro and Commercial Desktop
- RStudio Connect
- RStudio Shiny Server Pro
- Shinyapps.io

RStudio, Shiny, and R Markdown are trademarks of RStudio, Inc.

Images from rstudio.com
Who is Using the R Language

- Statistics
- Analysis
- Biology
- Economics
- Ecology
- Chemical Engineering
- Computational Anthropology
Who is Using the R Language

- Statistics
- Analysis
- Biology
- Economics
- Ecology
- Chemical Engineering
- Computational Anthropology

- Wrapper around math libraries
- Raises the level of thinking
The “AGE of the WEB”
The “AGE of the WEB”
The “AGE of the WEB” 2.0
The “AGE of the WEB” 2.0 (or 1.x)
The “AGE of the WEB” 2.0 (or 1.x)

Fun with Spirographs

The spirograph equation for three of more wheels can be generalized as follows:

\[ z(t) = \sum_{k=1}^{n} a_k e^{i(t+\alpha_k)} \]

This program solves these equations for three wheels, assuming all of the \( a \) coefficients are 1 and \( \alpha \) coefficients are 0. Find more details online at [http://frink.sourceforge.net/131/buns.html](http://frink.sourceforge.net/131/buns.html).

```r
spiro <- function(a1,a2,a3) {
  t <- seq(0,1,length.out=1000)
  z <- exp(i*t*a1)+exp(i*t*a2)+exp(i*t*a3)
  result <- data.frame(x=Re(z),y=Im(z))
  return(result)
}

result <- spiro(13,-7,4)
```

The “AGE of the WEB” 2.0 (or 1.x)
The “AGE of the WEB” 2.0 (or 1.x)
The “AGE of the WEB” 2.0 (or 1.x)
Today’s Goals

Three ways to build reproducible, R based, content that can be published on a science gateway.

Applications

APIs

Articles

Spirographs... For Ages 3+

Plotting the spirograph equations with ‘gnuplot’

By Victor Lanz
Universidad de Oviedo, Departamento de Química Físico y Analítica, E-33006 Oviedo, Spain.

The author had specifically requested that we keep the large font used in his article in order to match the font size of the essay images.

The program, called ‘Spirograph’, utilizes the graphical capabilities of the ‘gnuplot’ package to create intricate and beautiful designs. The program takes as input a set of parameters that define the size of the spirograph, such as the radius of the inner and outer circles, the distance between the centers of the circles, and the number of times the pen is drawn around the circumference of the outer circle.

A typical use of the program is to create art for display or printing.

1. Introduction

Imagine the movement of a small circle that rolls, without slipping, on the inside of a big circle. Imagine now that the small circle has an arm, rigidly attached, with a point/paper fixed at some point. That is a recipe for drawing the hypotrochoids, a number of large family of curves including epicycloids (the moving circle rolls on the outside of the fixed one), and cycloids (the pen is on the edge of the rolling circle), and envelopes (surfaces formed by many different types of curves) in general.

The concept of wheels rolling on wheels is, in fact, an example of the cycloidal motions, a complex motion, known as a cycloidal motion, named after the cycloidal paths. Cycloidal motion is not only important in mathematics but also in physics, as it is essential in the design of gears and chains. The cycloidal motion is also important in the study of planetary motion, as it is the path that planets follow when they move around the sun.

2. The hypotrochoid and some related curves

Figure 1 shows the geometry of the hypotrochoid and will help us determine the parametric equations for the curve. Three lengths determine the shape of the curve. The radius of the fixed circle, , the radius of the moving circle, , and , the distance from the pen to the moving circle center. The center of the fixed circle, point O, will serve as the origin of the coordinate system. Points O’ and P’ designate the current position of the rolling circle center and the pen, respectively.

![Fig. 1. Geometry for the hypotrochoid equations. The grayed figure corresponds to R=4, r=2, and d=3.](https://linuxgazette.net/133/luana.html)
1. We can model spirographs as a sum of exponentials:

\[ z(t) = \sum_{k=1}^{n} a_k e^{i2\pi(n_k t + \theta_k)}, t \in [0, 1] \]
1. We can model spirographs as a sum of exponentials:

\[ z(t) = \sum_{k=1}^{n} a_k e^{i2\pi(n_k t + \theta_k)}, \quad t \in [0, 1] \]

2. We can model a 3-wheeled system, where \( \alpha_k = 1 \) and \( \theta_k = 0 \), with this equation:

\[ z(t) = e^{i2\pi(n_1 t)} + e^{i2\pi(n_2 t)} + e^{i2\pi(n_3 t)}, \quad t \in [0, 1] \]
**TL;DR:**

1. We can model spirographs as a sum of exponentials:
   \[ z(t) = \sum_{k=1}^{n} a_k e^{i2\pi(n_k t + \theta_k)}, \quad t \in [0, 1] \]

2. We can model a 3-wheeled system, where \( \alpha_k = 1 \) and \( \theta_k = 0 \), with this equation:
   \[ z(t) = e^{i2\pi(n_1 t)} + e^{i2\pi(n_2 t)} + e^{i2\pi(n_3 t)}, \quad t \in [0, 1] \]

3. Need three inputs from the user: \( n_1, n_2, \) and \( n_3 \)

---

Spirographs... For Ages 3+

---

https://linuxgazette.net/133/luana.html
Spirographs... For Ages 3+

**TL;DR:**

1. We can model spirographs as a sum of exponentials:

   \[
   z(t) = \sum_{k=1}^{n} a_k e^{i2\pi (n_k t + \theta_k)}, t \in [0, 1]
   \]

2. We can model a 3-wheeled system, where \(\alpha_k = 1\) and \(\theta_k = 0\), with this equation:

   \[
   z(t) = e^{i2\pi(n_1 t)} + e^{i2\pi(n_2 t)} + e^{i2\pi(n_3 t)}, t \in [0, 1]
   \]

3. Need three inputs from the user: \(n_1\), \(n_2\), and \(n_3\)

---

```r
spiro <- function(n1, n2, n3) {
  t <- seq(0, 1, length.out=1000)
  z <- exp(1i*2*pi*n1*t) + 
       exp(1i*2*pi*n2*t) + 
       exp(1i*2*pi*n3*t)
  result <- tibble(x=Re(z), y=Im(z))
  return (result)
}
```
spiro <- function(n1, n2, n3) {
  t <- seq(0, 1, length.out=1000)
  z <- exp(1i*2*pi*n1*t) +
      exp(1i*2*pi*n2*t) +
      exp(1i*2*pi*n3*t)
  result <- tibble(x=Re(z), y=Im(z))
  return (result)
}

TL;DR:

1. We can model spirographs as a sum of exponentials:

\[
    z(t) = \sum_{k=1}^{n} a_k e^{i2\pi(n_k t+\theta_k)}, t \in [0, 1]
\]

2. We can model a 3-wheeled system, where \( a_k = 1 \) and \( \theta_k = 0 \), with this equation:

\[
    z(t) = e^{i2\pi(n_1 t)} + e^{i2\pi(n_2 t)} + e^{i2\pi(n_3 t)}, t \in [0, 1]
\]

3. Need three inputs from the user: \( n_1 \), \( n_2 \), and \( n_3 \)
Spirographs… For Ages 3+

TL;DR:

1. We can model spirographs as a sum of exponentials:

   \[ z(t) = \sum_{k=1}^{n} a_k e^{i2\pi(n_k t + \theta_k)}, \quad t \in [0, 1] \]

2. We can model a 3-wheeled system, where \( a_k = 1 \) and \( \theta_k = 0 \), with this equation:

   \[ z(t) = e^{i2\pi(n_1 t)} + e^{i2\pi(n_2 t)} + e^{i2\pi(n_3 t)}, \quad t \in [0, 1] \]

3. Need three inputs from the user: \( n_1, n_2, \) and \( n_3 \)

```r
spiro <- function(n1, n2, n3) {
  t <- seq(0, 1, length.out=1000)
  z <- exp(1i*2*pi*n1*t) +
       exp(1i*2*pi*n2*t) +
       exp(1i*2*pi*n3*t)
  result <- tibble(x=Re(z), y=Im(z))
  return (result)
}
```
Spirographs… For Ages 3+

```r
spiro <- function(n1, n2, n3) {
  t <- seq(0, 1, length.out=1000)
  z <- exp(1i*2*pi*n1*t) +
       exp(1i*2*pi*n2*t) +
       exp(1i*2*pi*n3*t)
  result <- tibble(x=Re(z), y=Im(z))
  return (result)
}
```

**TL;DR:**

1. We can model spirographs as a sum of exponentials:
   \[
   z(t) = \sum_{k=1}^{n} \alpha_k e^{i2\pi(n_k t + \theta_k)}, \quad t \in [0, 1]
   \]

2. We can model a 3-wheeled system, where \(\alpha_k = 1\) and \(\theta_k = 0\), with this equation:
   \[
   z(t) = e^{i2\pi(n_1 t)} + e^{i2\pi(n_2 t)} + e^{i2\pi(n_3 t)}, \quad t \in [0, 1]
   \]

3. Need three inputs from the user: \(n_1\), \(n_2\), and \(n_3\)
Spirographs… For Ages 3+

TL;DR:

1. We can model spirographs as a sum of exponentials:

\[ z(t) = \sum_{k=1}^{n} a_k e^{i2\pi(n_k t + \theta_k)}, t \in [0, 1] \]

2. We can model a 3-wheeled system, where \( \alpha_k = 1 \) and \( \theta_k = 0 \), with this equation:

\[ z(t) = e^{i2\pi(n_1 t)} + e^{i2\pi(n_2 t)} + e^{i2\pi(n_3 t)}, t \in [0, 1] \]

3. Need three inputs from the user: \( n_1, n_2, \) and \( n_3 \)
spo <- function(n1, n2, n3) {
  t <- seq(0, 1, length.out = 1000)
  z <- exp(i*2*pi*n1*t) +
  exp(i*2*pi*n2*t) +
  exp(i*2*pi*n3*t)
  result <- tibble(x=Re(z), y=Im(z))
  return (result)
}

> z <- spiro(13, -7, -3)
> z
# A tibble: 1,000 x 2
   x       y
  <dbl>   <dbl>
1   3 0
2 3.00 0.0188
3 2.98 0.0371
4 2.96 0.0546
5 2.93 0.0707
6 2.89 0.0850
7 2.84 0.0971
8 2.78 0.107
9 2.72 0.113
10 2.65 0.116
# ... with 990 more rows
> ggplot(...)

Spirographs... For Ages 3+
Spirographs… For Ages 3+

spiro <- function(n1, n2, n3) {
  t <- seq(0, 1, length.out=1000)
  z <- exp(1i*2*pi*n1*t) +
      exp(1i*2*pi*n2*t) +
      exp(1i*2*pi*n3*t)
  result <- tibble(x=Re(z), y=Im(z))
  return (result)
}

> z <- spiro(13, -7, -3)
> z
# A tibble: 1,000 x 2
   x    y
  <dbl> <dbl>
1  3  0
2  3.00 0.0188
3  2.98 0.0371
4  2.96 0.0546
5  2.93 0.0707
6  2.89 0.0850
7  2.84 0.0971
8  2.78 0.107
9  2.72 0.113
10 2.65 0.116
# ... with 990 more rows
> ggplot(...)

> z <- spiro(0, 10, -3)
> z
# A tibble: 1,000 x 2
   x    y
  <dbl> <dbl>
1  3  0
2  3.00 0.0440
3  2.99 0.0877
4  2.98 0.131
5  2.97 0.174
6  2.95 0.215
7  2.92 0.256
8  2.90 0.294
9  2.86 0.332
10 2.83 0.367
# ... with 990 more rows
> ggplot(...)
Web Applications

POST "/"  
localhost:3000

app.R

Web API
Building Web Applications with Shiny

```r
spiro <- function(n1, n2, n3) {
  t <- seq(0, 1, length.out=1000)
  z <- exp(1i * 2 * pi * n1 * t) +
       exp(1i * 2 * pi * n2 * t) +
       exp(1i * 2 * pi * n3 * t)
  result <- tibble(x=Re(z), y=Im(z))
  return(result)
}

# Define UI for spirograph application
ui <- fluidPage(
  # Input and Output
  # User Interface Widgets
)

# Define server logic required to draw
# a spirograph
server <- function(input, output) {
  # Call spiro()
  # Return x,y points
}

# Run the application
shinyApp(ui, server)
```

POST “/”
localhost:3000
Building Web Applications with Shiny

### Function Definition

```r
spiro <- function(n1,n2,n3) {
  t <- seq(0,1,length.out=1000)
  z <- exp(1i*2*pi*n1*t) +
      exp(1i*2*pi*n2*t) +
      exp(1i*2*pi*n3*t)
  result <- tibble(x=Re(z),y=Im(z))
  return (result)
}
```

### UI for Spirograph Application

```r
# Define UI for spirograph application
ui <- fluidPage(
  # Input and Output
  # User Interface Widgets
)

# Define server logic required to draw a spirograph
server <- function(input, output) {
  # Call spiro()
  # Return x,y points
}

# Run the application
shinyApp(ui, server)
```
Building Web Applications with Shiny

Server glues widgets to science code

✅ Reactivity - responds to user changes
✅ Brushing - linking widgets together

```r
spiro <- function(n1, n2, n3) {
  t <- seq(0, 1, length.out=1000)
  z <- exp(1i*2*pi*n1*t) +
      exp(1i*2*pi*n2*t) +
      exp(1i*2*pi*n3*t)
  result <- tibble(x=Re(z), y=Im(z))
  return (result)
}

# Define UI for spirograph application
ui <- fluidPage(
  # Input and Output
  # User Interface Widgets
)

# Define server logic required to draw
# a spirograph
server <- function(input, output) {
  # Call spiro()
  # Return x,y points
}

# Run the application
shinyApp(ui, server)
```
Building Web Applications
with Shiny

```
spiro <- function(n1, n2, n3) {
  t <- seq(0, 1, length.out=1000)
  z <- exp(1i * 2 * pi * n1 * t) +
       exp(1i * 2 * pi * n2 * t) +
       exp(1i * 2 * pi * n3 * t)
  result <- tibble(x=Re(z), y=Im(z))
  return (result)
}
```

```r
# Define UI for spirograph application
ui <- fluidPage(
  # Input and Output
  # User Interface Widgets
)

# Define server logic required to draw
# a spirograph
server <- function(input, output) {
  # Call spiro()
  # Return x,y points
}
```

# Run the application
shinyApp(ui, server)
Getting Started with Shiny

https://shiny.rstudio.com

https://shiny.rstudio.com/articles/basics.html

https://htmlwidgets.org

https://rstudio.github.io/shinydashboard
Web Application Basics

POST "/'
localhost:3000

Web Application

app.R
Web API Basics

Web API

POST "/spiro/calc"
localhost:3000

app.R

client.py

client.js

client.R

Python logo is a trademark of Python Software Foundation
Node.js is a trademark of Joyent, Inc.
Still sending messages
Focused on sharing information
Communication with other programs
Web API Basics

Who's Using Web APIs?

POST “/spiro/calc”
localhost:3000

✅ Still sending messages
✅ Focused on sharing information
✅ Communication with other programs

Python logo is a trademark of Python Software Foundation
Node.js is a trademark of Joyent, Inc.
Who’s Using Web APIs?

Web API Basics

✅ Still sending messages
✅ Focused on sharing information
✅ Communication with other programs

Node.js is a trademark of Joyent, Inc.
Python logo is a trademark of Python Software Foundation
Node.js is a trademark of Joyent, Inc.
Web API Basics

Who's Using Web APIs?

- Still sending messages
- Focused on sharing information
- Communication with other programs

POST "/spiro/calc"
localhost:3000

Python logo is a trademark of Python Software Foundation
Node.js is a trademark of Joyent, Inc.
Who Using Web APIs?

POST "/spiro/calc"
localhost:3000

Still sending messages
Focused on sharing information
Communication with other programs

✅

Node.js is a trademark of Joyent, Inc.
Python logo is a trademark of Python Software Foundation
Node.js is a trademark of Joyent, Inc.
Who’s Using Web APIs?

- POST “/spiro/calc”
- localhost:3000

Still sending messages
Focused on sharing information
Communication with other programs
Web API Basics

Who's Using Web APIs?

- SWAPI
- OpenFEC API Documentation

Web API

app.R

client.py

client.js

client.R

POST “/spiro/calc”
localhost:3000

✅ Still sending messages
✅ Focused on sharing information
✅ Communication with other programs

Still sending messages
Focused on sharing information
Communication with other programs

Python logo is a trademark of Python Software Foundation
Node.js is a trademark of Joyent, Inc.
Web API Basics

Who's Using Web APIs?

Still sending messages
Focused on sharing information
Communication with other programs

POST "/spiro/calc"
localhost:3000

Python logo is a trademark of Python Software Foundation
Node.js is a trademark of Joyent, Inc.
Building Web APIs with Plumber

```r
spiro <- function(n1=13, n2=-7, n3=-3) {
  t <- seq(0, 1, length.out=1000)
  z <- exp(1i*2*pi*n1*t) +
    exp(1i*2*pi*n2*t) +
    exp(1i*2*pi*n3*t)
  result <- tibble(x=Re(z), y=Im(z))
  return(result)
}
```

// Spirograph with custom n1, n2, n3
// @post /spiro/calc
// @param n3 Characteristics for the third wheel
// @param n2 Characteristics for the second wheel
// @param n1 Characteristics for the first wheel

```js
function(n1, n2, n3) {
  return(spiro(as.numeric(n1), as.numeric(n2), as.numeric(n3))
}
```

// Spirograph plot
// @get /spiro/plot
// @param n3 Characteristics for the third wheel
// @param n2 Characteristics for the second wheel
// @param n1 Characteristics for the first wheel

```js
result <- spiro(as.numeric(n1), as.numeric(n2), as.numeric(n3))
plot(result$x, result$y, xlab="Real(z)", ylab="Imag(z)"
```

POST “/spiro/calc”
localhost:3000

Python logo is a trademark of Python Software Foundation
Node.js is a trademark of Joyent, Inc.
spiro <- function(n1=13,n2=-7,n3=-3) {
  t <- seq(0,1,length.out=1000)
  z <- exp(1i*2*pi*n1*t) +
       exp(1i*2*pi*n2*t) +
       exp(1i*2*pi*n3*t)
  result <- tibble(x=Re(z),y=Im(z))
  return(result)
}

#* Spirograph with custom n1, n2, n3
#* @post /spiro/calc
#* @param n3 Characteristics for the third wheel
#* @param n2 Characteristics for the second wheel
#* @param n1 Characteristics for the first wheel
#* @json
function(n1,n2,n3){
  return(spiro(as.numeric(n1),
               as.numeric(n2),
               as.numeric(n3)))
}
spiro <- function(n1=13, n2=-7, n3=-3) {
  t <- seq(0,1,length.out=1000)
  z <- exp(1i*2*pi*n1*t) +
      exp(1i*2*pi*n2*t) +
      exp(1i*2*pi*n3*t)
  result <- tibble(x=Re(z), y=Im(z))
  return(result)
}

#* Spirograph with custom n1, n2, n3
#* @post /spiro/calc
#* @param n3 Characteristics for the third wheel
#* @param n2 Characteristics for the second wheel
#* @param n1 Characteristics for the first wheel
#* @json
function(n1, n2, n3){
  return(spiro(as.numeric(n1),
               as.numeric(n2),
               as.numeric(n3)))
}
Building Web APIs with Plumber

```r
spiro <- function(n1 = 13, n2 = -7, n3 = -3) {
  t <- seq(0, 1, length.out = 1000)
  z <- exp(1i * 2 * pi * n1 * t) +
       exp(1i * 2 * pi * n2 * t) +
       exp(1i * 2 * pi * n3 * t)
  result <- tibble(x = Re(z), y = Im(z))
  return(result)
}

#* Spirograph with custom n1, n2, n3
#* @post /spiro/calc
#* @param n3 Characteristics for the third wheel
#* @param n2 Characteristics for the second wheel
#* @param n1 Characteristics for the first wheel
#* @json
function(n1, n2, n3) {
  return(spiro(as.numeric(n1),
               as.numeric(n2),
               as.numeric(n3)))
}

#* Spirograph plot
#* @get /spiro/plot
#* @param n3 Characteristics for the third wheel
#* @param n2 Characteristics for the second wheel
#* @param n1 Characteristics for the first wheel
#* @png
function(n1, n2, n3) {
  result <- spiro(as.numeric(n1),
                  as.numeric(n2),
                  as.numeric(n3))
  plot(result$x, result$y,
       xlab = "Real(z)",
       ylab = "Imag(z)")
}
```

GET “/spiro/plot”
localhost:3000
function(n1=13,n2=-7,n3=-3) {
  t <- seq(0,1,length.out=1000)
  z <- exp(1i*2*pi*n1*t) +
      exp(1i*2*pi*n2*t) +
      exp(1i*2*pi*n3*t)
  result <- tibble(x=Re(z),y=Im(z))
  return(result)
}

#* Spirograph with custom n1, n2, n3
#* @post /spiro/calc
#* @param n3 Characteristics for the third wheel
#* @param n2 Characteristics for the second wheel
#* @param n1 Characteristics for the first wheel
#* @json
function(n1,n2,n3){
  return(spiro(as.numeric(n1),
                as.numeric(n2),
                as.numeric(n3)))
}

#* Spirograph plot
#* @get /spiro/plot
#* @param n3 Characteristics for the third wheel
#* @param n2 Characteristics for the second wheel
#* @param n1 Characteristics for the first wheel
#* @png
function(n1,n2,n3){
  result <- spiro(as.numeric(n1),
                  as.numeric(n2),
                  as.numeric(n3))
  plot(result$x, result$y,
        xlab="Real(z)",
        ylab="Imag(z)")
}
**Building Web APIs with Plumber**

Try it: [https://beta.rstudioconnect.com/connect/#/apps/3533](https://beta.rstudioconnect.com/connect/#/apps/3533)

```r
spiro <- function(n1=13, n2=-7, n3=-3) {
  t <- seq(0,1,length.out=1000)
  z <- exp(1i*2*pi*n1*t) +
      exp(1i*2*pi*n2*t) +
      exp(1i*2*pi*n3*t)
  result <- tibble(x=Re(z), y=Im(z))
  return(result)
}

#* Spirograph with custom n1, n2, n3
#* @post /spiro/calc
#* @param n3 Characteristics for the third wheel
#* @param n2 Characteristics for the second wheel
#* @param n1 Characteristics for the first wheel
#* @json
function(n1,n2,n3) {
  return(spiro(as.numeric(n1),
               as.numeric(n2),
               as.numeric(n3)))
}

#* Spirograph plot
#* @get /spiro/plot
#* @param n3 Characteristics for the third wheel
#* @param n2 Characteristics for the second wheel
#* @param n1 Characteristics for the first wheel
#* @png
function(n1,n2,n3) {
  result <- spiro(as.numeric(n1),
                  as.numeric(n2),
                  as.numeric(n3))
  plot(result$x, result$y,
       xlab="Real(z)",
       ylab="Imag(z)")
}
```
Getting Started with Plumber

Keep in mind…

1. APIs are good for sharing access to datasets you put together.
2. Watch out for long running processes.
3. Need an R process to run the server and background processes.

https://www.rplumber.io
R + Markdown

✅ Write reproducible content using the Markdown syntax.

# H1
## H2
### H3

*Emphasis*

**Bold**

1. Ordered
2. Lists
3. Items

[inline links](https://sciencegateways.org)

H1
H2
H3

Emphasis

Bold

inline links
R + Markdown

✓ Write reproducible content using the Markdown syntax.

✓ Embed chunks of code between lines of narrative in literate programming style.
R + Markdown

- Write reproducible content using the Markdown syntax.
- Embed chunks of code between lines of narrative in literate programming style.
- Output to multiple formats.
Getting Started with R Markdown

https://rmarkdown.rstudio.com

https://rmarkdown.rstudio.com/flexdashboard

https://bookdown.org

https://rmarkdown.rstudio.com/developer_parameterized_reports.html
Learn More

Three ways to build reproducible, R based, content that can be published on a science gateway.

Applications

APIs

https://www.rplumber.io

Articles

https://rmarkdown.rstudio.com

Slides and Examples

https://github.com/dskard/sgci-201805