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

≠

Background

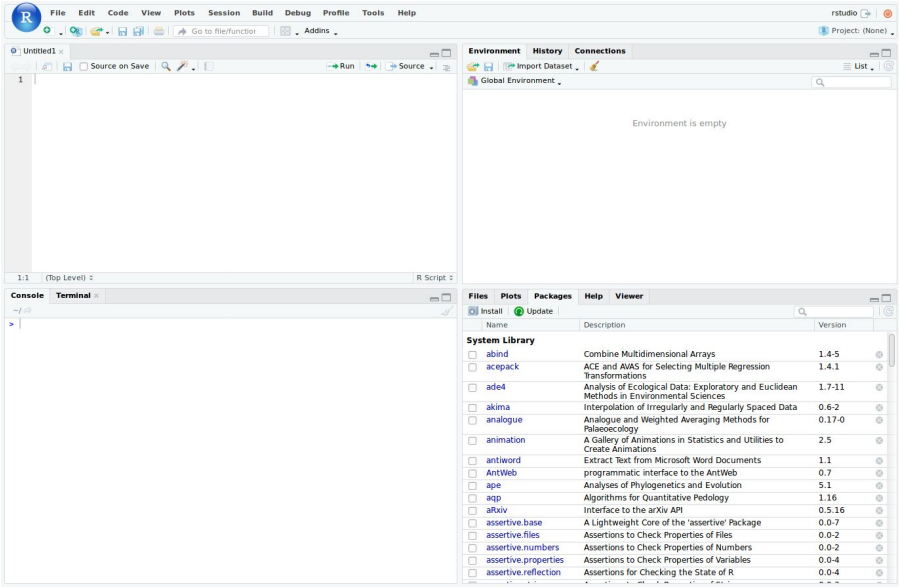
≅

+ . . .

Background

RStudio ≅ R + . . .

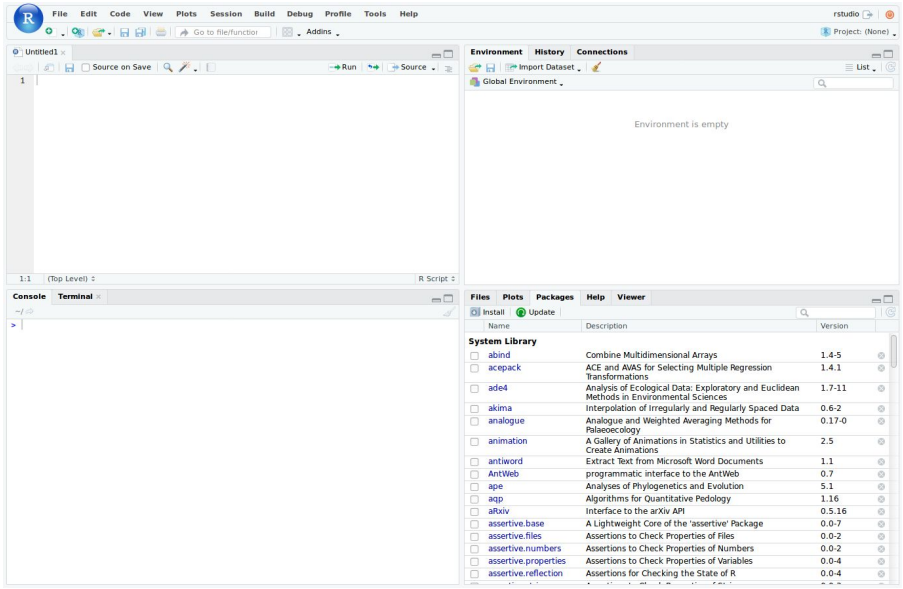
RStudio IDE



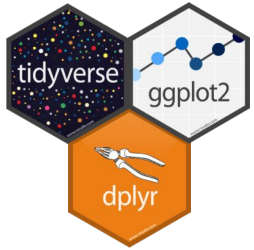
Background

RStudio ≅ R + . . .

RStudio IDE



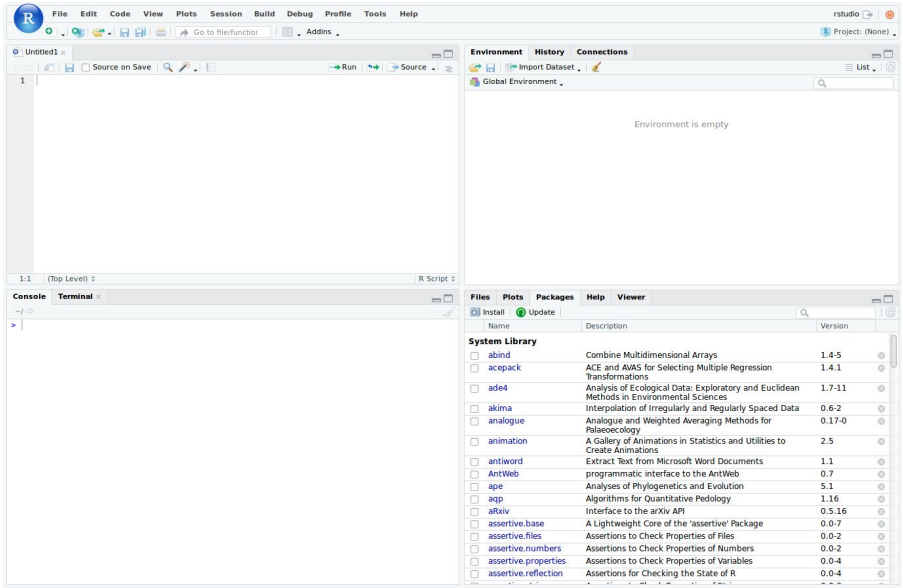
Open Source



Background

RStudio ≅ R + . . .

RStudio IDE

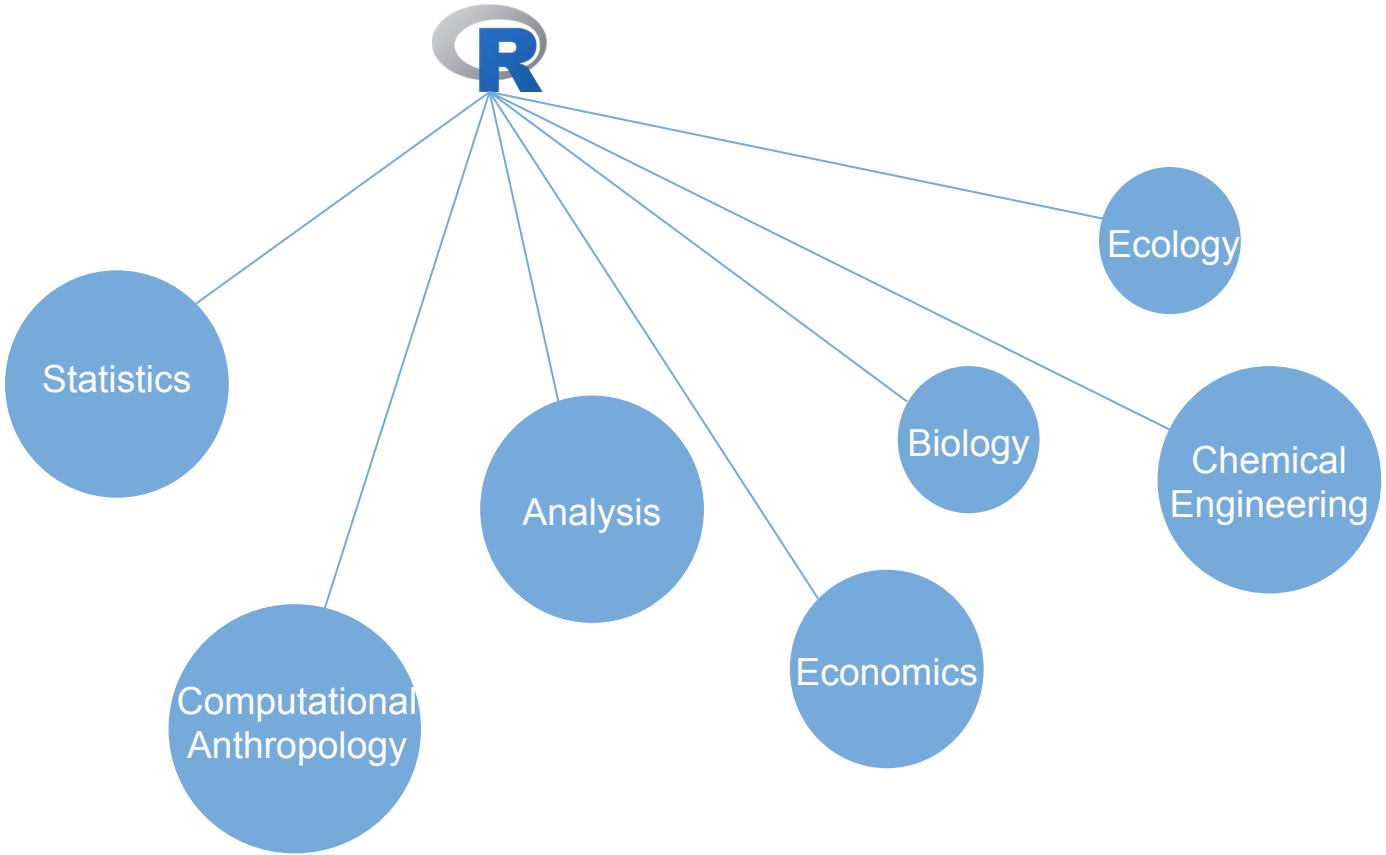


Open Source

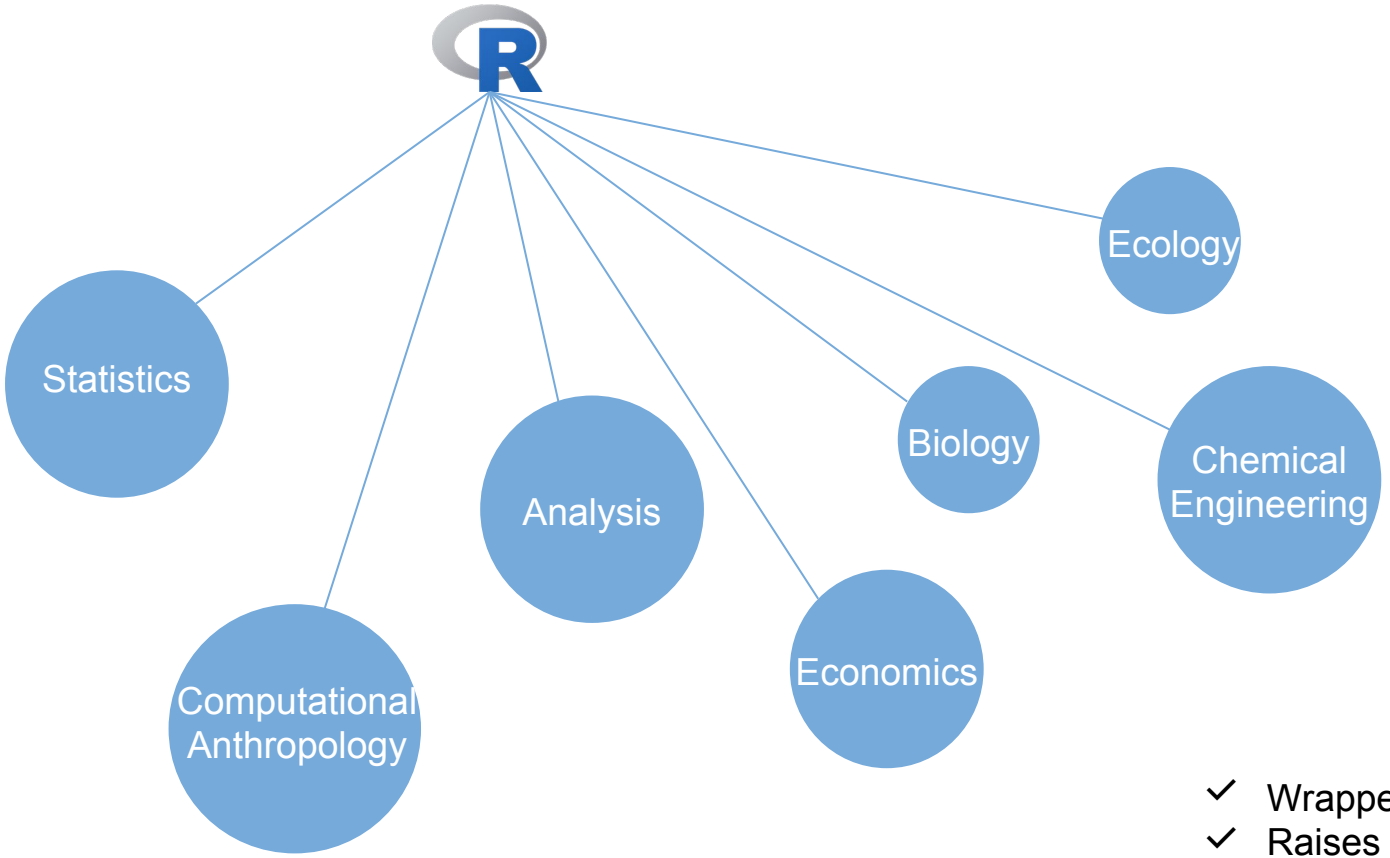
Enterprise-Ready Professional Products

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

Who is Using the R Language



Who is Using the R Language



- ✓ Wrapper around math libraries
- ✓ Raises the level of thinking

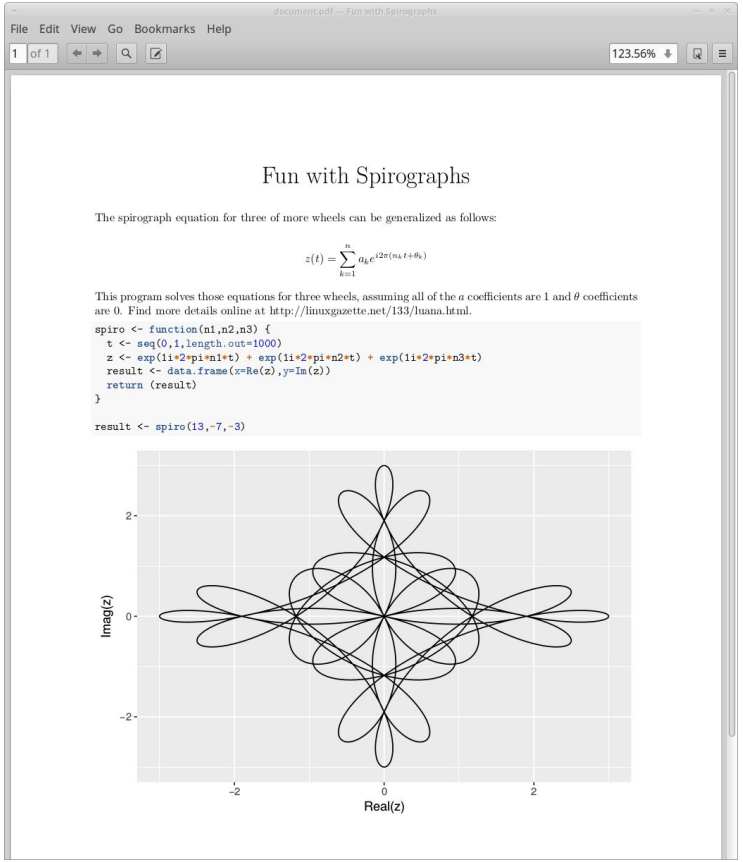
The “AGE of the WEB”

The “AGE of the WEB”

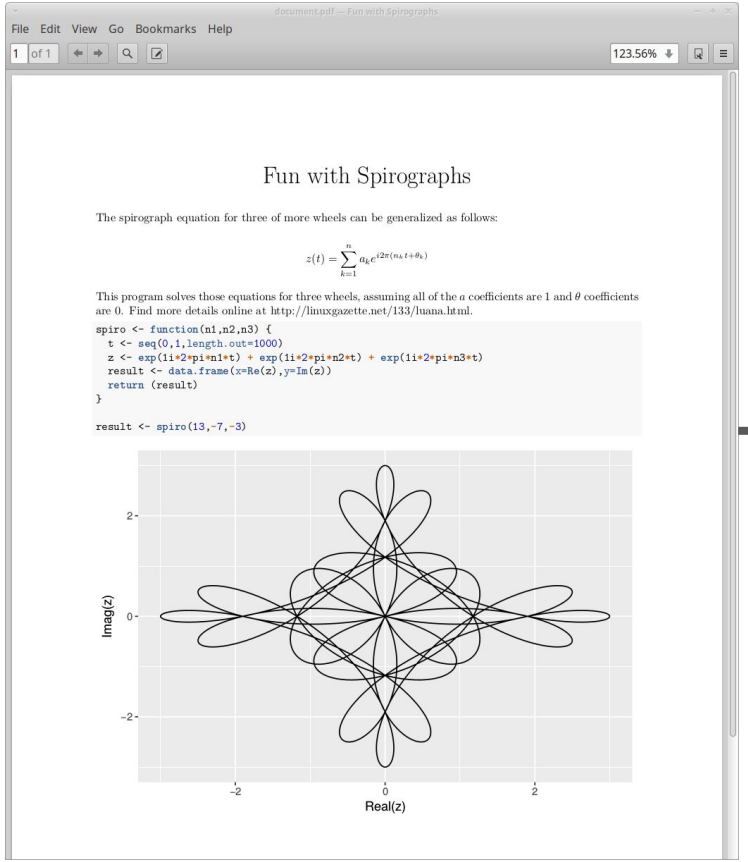
The “AGE of the WEB” 2.0

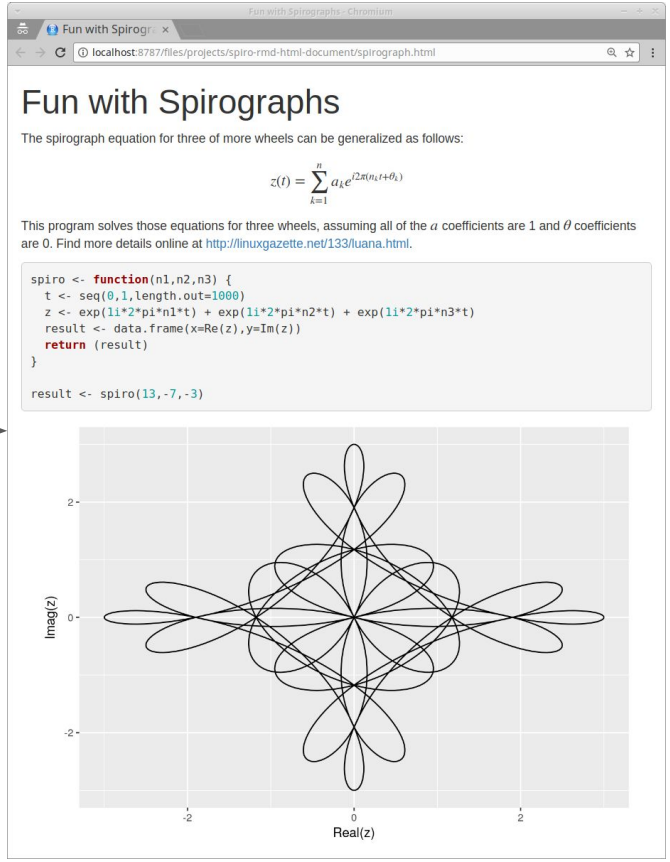
The “AGE of the WEB” 2.0 (or 1.x)

The “AGE of the WEB” 2.0 (or 1.x)

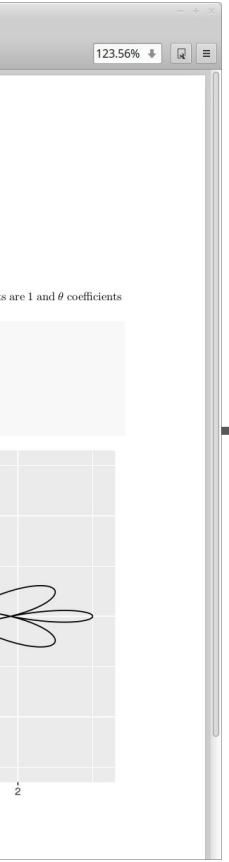


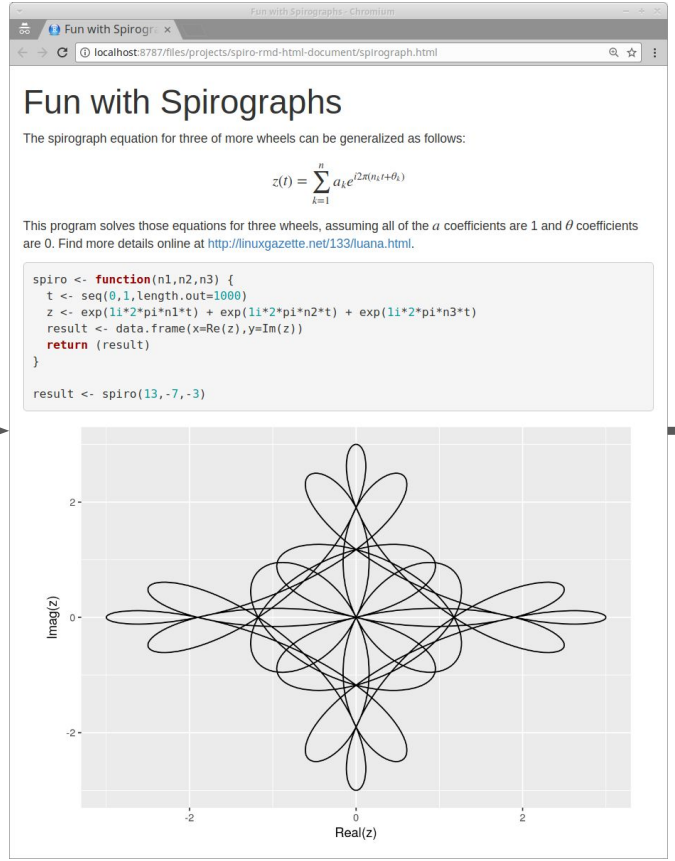
The “AGE of the WEB” 2.0 (or 1.x)

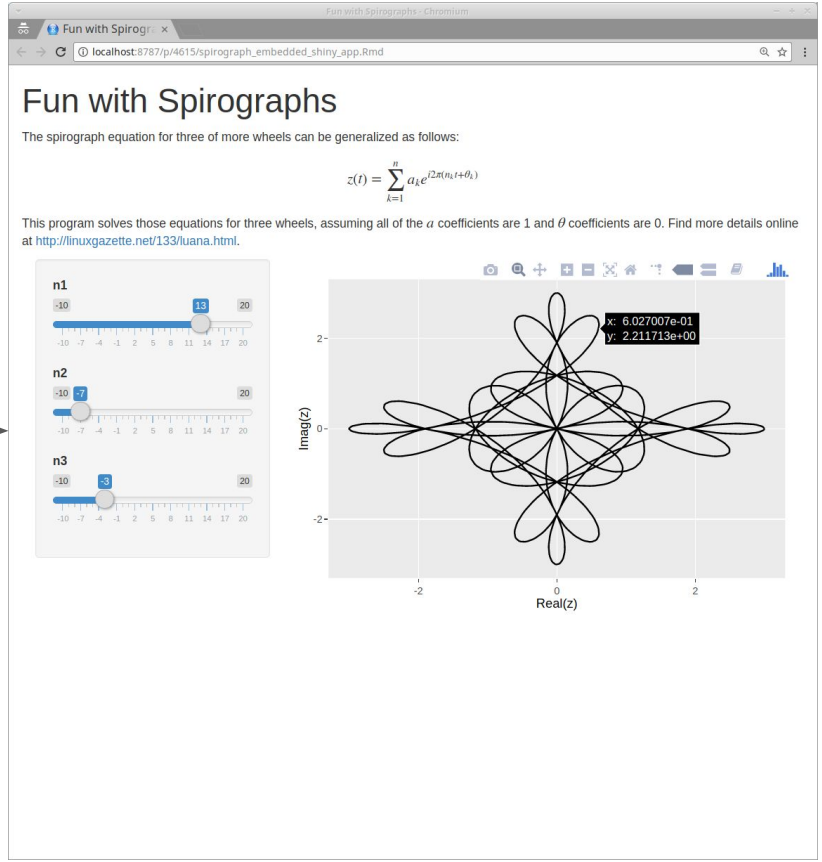




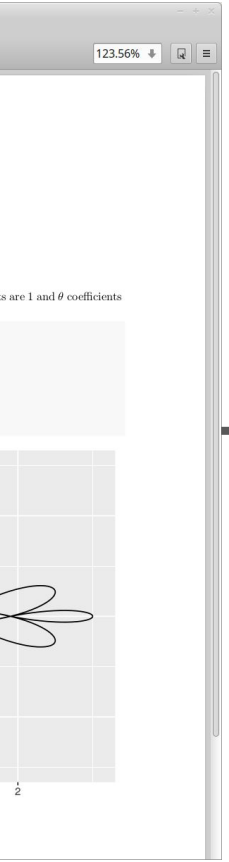
The “AGE of the WEB” 2.0 (or 1.x)

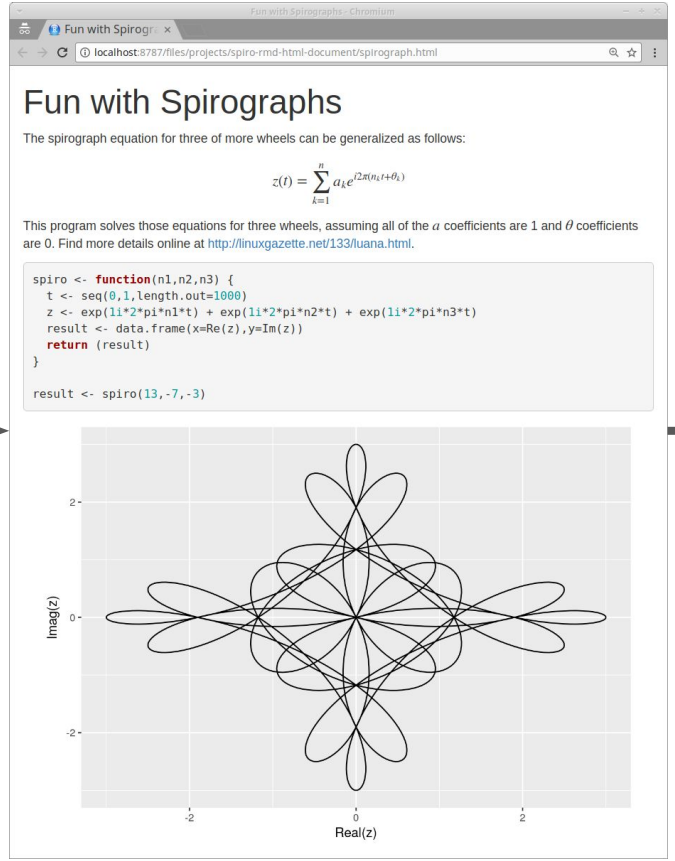


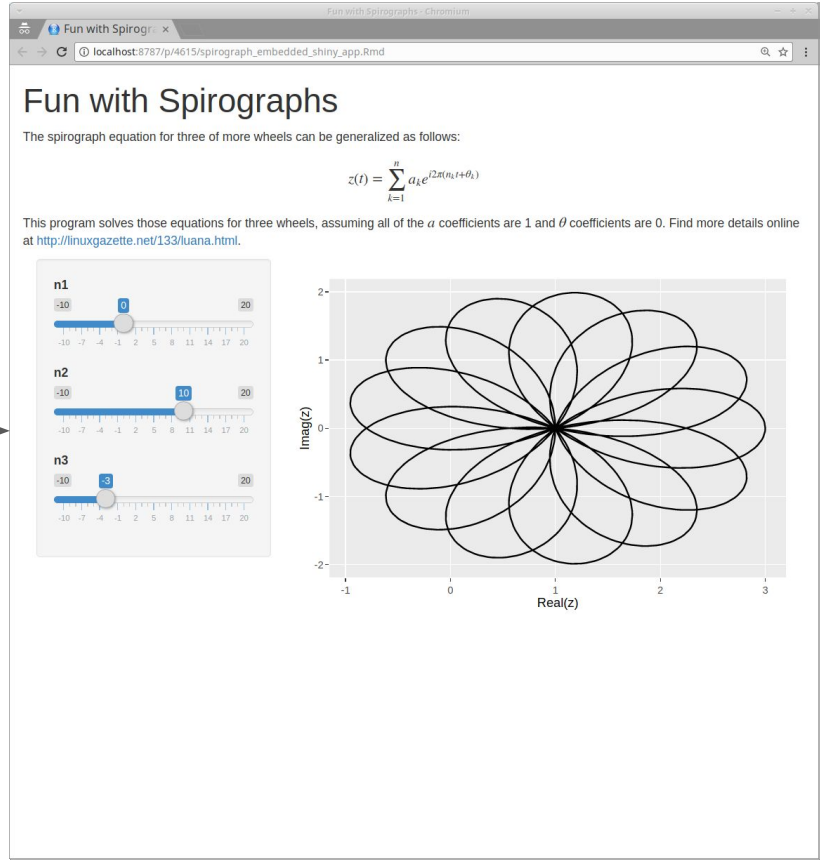




The “AGE of the WEB” 2.0 (or 1.x)







Today's Goals

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

Applications

https://shiny.rstudio.com

APIs

https://www.rplumber.io

Articles

https://rmarkdown.rstudio.com

Spirographs… For Ages 3+

Spirographs… For Ages 3+

<-- prev | next -->

Plotting the spirograph equations with 'gnuplot'

By Víctor Luaña

Universidad de Oviedo, Departamento de Química Física 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 equation images; I agreed, since the two would look disproportionate otherwise. My apologies to anyone whose eyeballs exploded due to the rapid decompression. -- Ben]

GNUPLOT'S internal programming capabilities are used to plot the continuous and segmented versions of the spirograph equations. The segmented version, in particular, stretches the program model and requires the emulation of internal loops and conditional sentences. As a final exercise, we will develop an extensible mini-language, mixing GAWK and GNUPLOT programming, that lets the user combine any number of generalized spirographic patterns in a design.

A PDF version of this article is available for archiving and printing.

I. Introduction

Imagine the movement of a small circle that rolls, without slipping, on the inside of a rigid circle. Imagine now that the small circle has an arm, rigidly attached, with a plotting pen fixed at some point. That is a recipe for drawing the hypotrochoid, a member of a large family of curves including epitrochoids (the moving circle rolls on the outside of the fixed one), cycloids (the pen is on the edge of the rolling circle), and roulettes (several forms rolling on many different types of curves) in general.

The concept of wheels rolling on wheels can, in fact, be generalized to any number of embedded elements. Complex lathe engines, known as *Guilloché* machines, have been used since the 17th or 18th century for engraving beautiful designs onto watches, jewels, and other items of fine craftsmanship. Many sources attribute the first use of Guilloché engravings on a watch to Abraham-Louis Breguet in 1786, but the technique was already in use on jewelry. Ancient machines are still being used, and can be seen at the RGM Watch Company Web pages. Intricate Guilloché patterns are usually incorporated on bank notes and official documents to prevent forgery. The name "Spirograph" comes, actually, from the trade name of a toy invented in 1962 by Denys Fisher, a British electronic engineer, and licensed to several toy companies over the years.

Our purpose, however, is not to explore the history or even the mathematical aspects of the Spirograph decorations: our interest is centered on the techniques needed to use GNUPLOT as the drawing engine of the cycloid-related curves.

Section II presents a simple derivation for the hypotrochoid equations and discusses a generalization to any number of rolling wheels described by F. Farris. Section III describes the techniques required to draw the cycloid-related curves with GNUPLOT. From the use of complex arithmetic to the simulation of an implicit do loop and the recursive definition of user functions, GNUPLOT offers a large capability for the creation of algorithmic designs. The techniques discussed in Section III are embedded within a simple GAWK filter that reads a formal description of a cycloid pattern and uses GNUPLOT to produce the final plot. The design of this filter is the subject of Section IV.

II. The hypotrochoid and some related curves

Figure 1 shows the formation of a hypotrochoid and will help us in determining the parametric equations for the curve. Three lengths determine the shape of the curve: R, the radius of the fixed circle; r, the radius of the moving circle; and p, 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 of the pen, respectively.

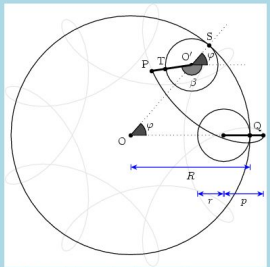


Figure 1 Geometry for the hypotrochoid equations. The grayed figure corresponds to R=9, r=2, and p=3.

https://linuxgazette.net/133/luana.html

Spirographs… For Ages 3+

<-- prev | next -->

Plotting the spirograph equations with 'gnuplot'

By Víctor Luaña

Universidad de Oviedo, Departamento de Química Física 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 equation images; I agreed, since the two would look disproportionate otherwise. My apologies to anyone whose eyeballs exploded due to the rapid decompression. -- Ben]

GNUPLOT'S internal programming capabilities are used to plot the continuous and segmented versions of the spirograph equations. The segmented version, in particular, stretches the program model and requires the emulation of internal loops and conditional sentences. As a final exercise, we will develop an extensible mini-language, mixing GAWK and GNUPLOT programming, that lets the user combine any number of generalized spirographic patterns in a design.

A PDF version of this article is available for archiving and printing.

I. Introduction

Imagine the movement of a small circle that rolls, without slipping, on the inside of a rigid circle. Imagine now that the small circle has an arm, rigidly attached, with a plotting pen fixed at some point. That is a recipe for drawing the hypotrochoid, a member of a large family of curves including epitrochoids (the moving circle rolls on the outside of the fixed one), cycloids (the pen is on the edge of the rolling circle), and roulettes (several forms rolling on many different types of curves) in general.

The concept of wheels rolling on wheels can, in fact, be generalized to any number of embedded elements. Complex lathe engines, known as *Guilloché* machines, have been used since the 17th or 18th century for engraving beautiful designs onto watches, jewels, and other items of fine craftsmanship. Many sources attribute the first use of Gilloché engravings on a watch to Abraham-Louis Breguet in 1786, but the technique was already in use on jewelry. Ancient machines are still being used, and can be seen at the RGM Watch Company Web pages. Intricate Guilloché patterns are usually incorporated on bank notes and official documents to prevent forgery. The name "Spirograph" comes, actually, from the trade name of a toy invented in 1962 by Denys Fisher, a British electronic engineer, and licensed to several toy companies over the years.

Our purpose, however, is not to explore the history or even the mathematical aspects of the Spirograph decorations: our interest is centered on the techniques needed to use GNUPLOT as the drawing engine of the cycloid-related curves.

Section II presents a simple derivation for the hypotrochoid equations and discusses a generalization to any number of rolling wheels described by F. Farris. Section III describes the techniques required to draw the cycloid-related curves with GNUPLOT. From the use of complex arithmetic to the simulation of an implicit do loop and the recursive definition of user functions, GNUPLOT offers a large capability for the creation of algorithmic designs. The techniques discussed in Section III are embedded within a simple GAWK filter that reads a formal description of a cycloid pattern and uses GNUPLOT to produce the final plot. The design of this filter is the subject of Section IV.

II. The hypotrochoid and some related curves

Figure 1 shows the formation of a hypotrochoid and will help us in determining the parametric equations for the curve. Three lengths determine the shape of the curve: R, the radius of the fixed circle; r, the radius of the moving circle; and p, 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 of the pen, respectively.

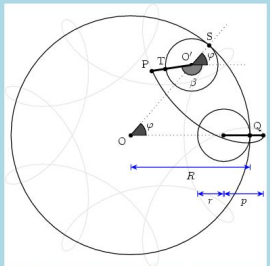


Figure 1 Geometry for the hypotrochoid equations. The grayed figure corresponds to R=9, r=2, and p=3.

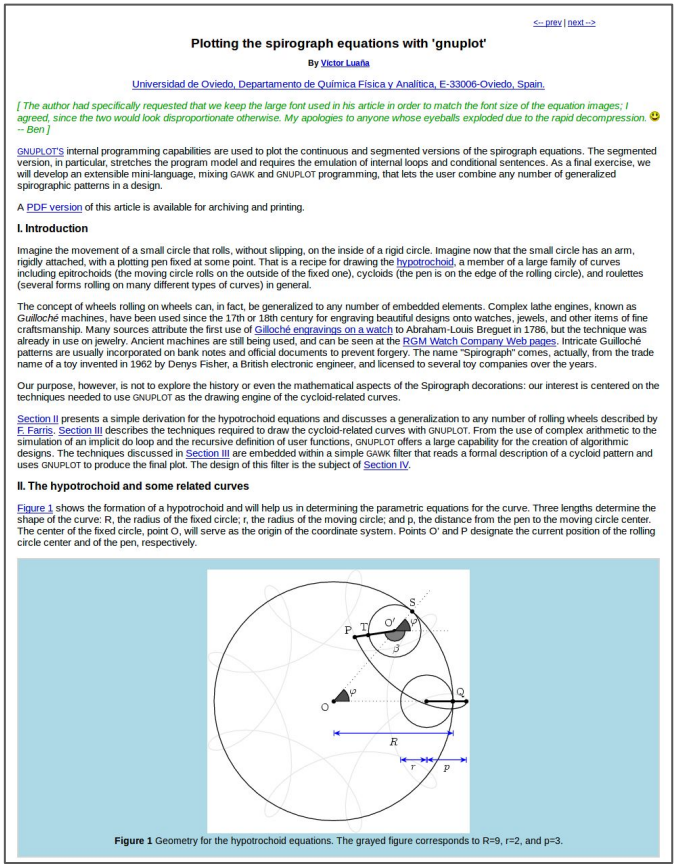
https://linuxgazette.net/133/luana.html

TL;DR:

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

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

Spirographs… For Ages 3+



<-- prev | next -->

Plotting the spirograph equations with 'gnuplot'

By Víctor Luaña

Universidad de Oviedo, Departamento de Química Física 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 equation images; I agreed, since the two would look disproportionate otherwise. My apologies to anyone whose eyeballs exploded due to the rapid decompression. -- Ben]

GNUPLOT'S internal programming capabilities are used to plot the continuous and segmented versions of the spirograph equations. The segmented version, in particular, stretches the program model and requires the emulation of internal loops and conditional sentences. As a final exercise, we will develop an extensible mini-language, mixing GAWK and GNUPLOT programming, that lets the user combine any number of generalized spirographic patterns in a design.

A PDF version of this article is available for archiving and printing.

I. Introduction

Imagine the movement of a small circle that rolls, without slipping, on the inside of a rigid circle. Imagine now that the small circle has an arm, rigidly attached, with a plotting pen fixed at some point. That is a recipe for drawing the hypotrochoid, a member of a large family of curves including epitrochoids (the moving circle rolls on the outside of the fixed one), cycloids (the pen is on the edge of the rolling circle), and roulettes (several forms rolling on many different types of curves) in general.

The concept of wheels rolling on wheels can, in fact, be generalized to any number of embedded elements. Complex lathe engines, known as *Guilloché* machines, have been used since the 17th or 18th century for engraving beautiful designs onto watches, jewels, and other items of fine craftsmanship. Many sources attribute the first use of Gilloché engravings on a watch to Abraham-Louis Breguet in 1786, but the technique was already in use on jewelry. Ancient machines are still being used, and can be seen at the RGM Watch Company Web pages. Intricate Guilloché patterns are usually incorporated on bank notes and official documents to prevent forgery. The name "Spirograph" comes, actually, from the trade name of a toy invented in 1962 by Denys Fisher, a British electronic engineer, and licensed to several toy companies over the years.

Our purpose, however, is not to explore the history or even the mathematical aspects of the Spirograph decorations: our interest is centered on the techniques needed to use GNUPLOT as the drawing engine of the cycloid-related curves.

Section II presents a simple derivation for the hypotrochoid equations and discusses a generalization to any number of rolling wheels described by F. Farris. Section III describes the techniques required to draw the cycloid-related curves with GNUPLOT. From the use of complex arithmetic to the simulation of an implicit do loop and the recursive definition of user functions, GNUPLOT offers a large capability for the creation of algorithmic designs. The techniques discussed in Section III are embedded within a simple GAWK filter that reads a formal description of a cycloid pattern and uses GNUPLOT to produce the final plot. The design of this filter is the subject of Section IV.

II. The hypotrochoid and some related curves

Figure 1 shows the formation of a hypotrochoid and will help us in determining the parametric equations for the curve. Three lengths determine the shape of the curve: R, the radius of the fixed circle; r, the radius of the moving circle; and p, 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 of the pen, respectively.

Figure 1 Geometry for the hypotrochoid equations. The grayed figure corresponds to R=9, r=2, and p=3.

https://linuxgazette.net/133/luana.html

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 α_k = 1 and θ_k = 0, with this equation:

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

Spirographs… For Ages 3+

<-- prev | next -->

Plotting the spirograph equations with 'gnuplot'

By Víctor Luaña

Universidad de Oviedo, Departamento de Química Física 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 equation images; I agreed, since the two would look disproportionate otherwise. My apologies to anyone whose eyeballs exploded due to the rapid decompression. -- Ben]

GNUPLOT'S internal programming capabilities are used to plot the continuous and segmented versions of the spirograph equations. The segmented version, in particular, stretches the program model and requires the emulation of internal loops and conditional sentences. As a final exercise, we will develop an extensible mini-language, mixing GAWK and GNUPLOT programming, that lets the user combine any number of generalized spirographic patterns in a design.

A PDF version of this article is available for archiving and printing.

I. Introduction

Imagine the movement of a small circle that rolls, without slipping, on the inside of a rigid circle. Imagine now that the small circle has an arm, rigidly attached, with a plotting pen fixed at some point. That is a recipe for drawing the hypotrochoid, a member of a large family of curves including epitrochoids (the moving circle rolls on the outside of the fixed one), cycloids (the pen is on the edge of the rolling circle), and roulettes (several forms rolling on many different types of curves) in general.

The concept of wheels rolling on wheels can, in fact, be generalized to any number of embedded elements. Complex lathe engines, known as *Guilloché* machines, have been used since the 17th or 18th century for engraving beautiful designs onto watches, jewels, and other items of fine craftsmanship. Many sources attribute the first use of Gilloché engravings on a watch to Abraham-Louis Breguet in 1786, but the technique was already in use on jewelry. Ancient machines are still being used, and can be seen at the RGM Watch Company Web pages. Intricate Guilloché patterns are usually incorporated on bank notes and official documents to prevent forgery. The name "Spirograph" comes, actually, from the trade name of a toy invented in 1962 by Denys Fisher, a British electronic engineer, and licensed to several toy companies over the years.

Our purpose, however, is not to explore the history or even the mathematical aspects of the Spirograph decorations: our interest is centered on the techniques needed to use GNUPLOT as the drawing engine of the cycloid-related curves.

Section II presents a simple derivation for the hypotrochoid equations and discusses a generalization to any number of rolling wheels described by F. Farris. Section III describes the techniques required to draw the cycloid-related curves with GNUPLOT. From the use of complex arithmetic to the simulation of an implicit do loop and the recursive definition of user functions, GNUPLOT offers a large capability for the creation of algorithmic designs. The techniques discussed in Section III are embedded within a simple GAWK filter that reads a formal description of a cycloid pattern and uses GNUPLOT to produce the final plot. The design of this filter is the subject of Section IV.

II. The hypotrochoid and some related curves

Figure 1 shows the formation of a hypotrochoid and will help us in determining the parametric equations for the curve. Three lengths determine the shape of the curve: R, the radius of the fixed circle; r, the radius of the moving circle; and p, 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 of the pen, respectively.

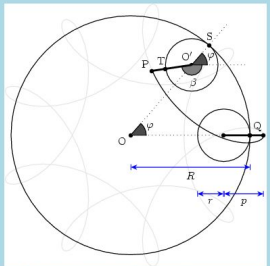


Figure 1 Geometry for the hypotrochoid equations. The grayed figure corresponds to R=9, r=2, and p=3.

https://linuxgazette.net/133/luana.html

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: $\mathbf{n_1}$, $\mathbf{n_2}$, and $\mathbf{n_3}$

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)
}
```

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 α_k = 1 and θ_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+

```
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 α_k = 1 and θ_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+

```
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 α_k = 1 and θ_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+

```
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 α_k = 1 and θ_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+

```
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 α_k = 1 and θ_k = 0, with this equation:

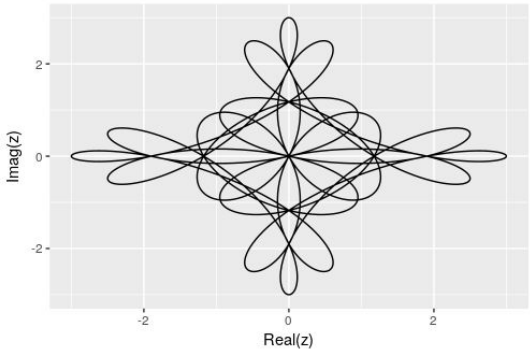
$$z(t) = e^{i2\pi(n_1 t)} + e^{i2\pi(n_2 t)} + e^{i2\pi(n_3 t)}, t \in [0, 1]$$

3. Need three inputs from the user: **n_1**, **n_2**, and **n_3**

Spirographs… For Ages 3+

```
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)
}
```

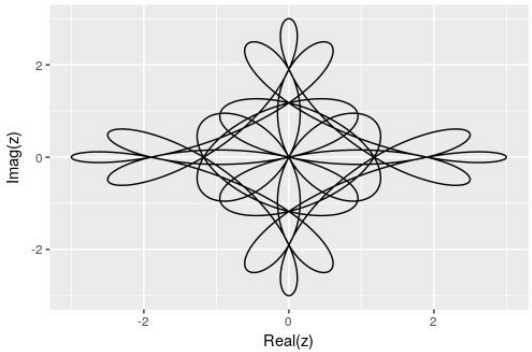
```
> z <- spiro(13,-7,-3)
> z
# A tibble: 1,000 x 2
      x      y
  <dbl>  <dbl>
 1  3     0
 2  3.00  0.0188
 3  2.98  0.0371
 4  2.96  0.0546
 5  2.93  0.0707
 6  2.89  0.0850
 7  2.84  0.0971
 8  2.78  0.107
 9  2.72  0.113
10  2.65  0.116
# ... with 990 more rows
> ggplot(...)
```



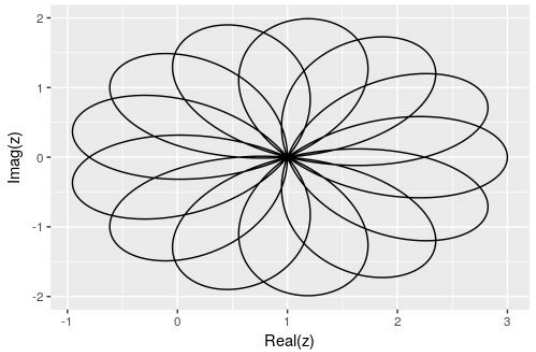
Spirographs… For Ages 3+

```
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)
}
```

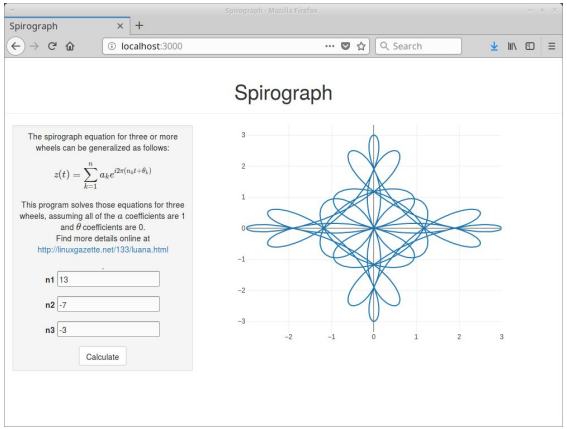
```
> z <- spiro(13,-7,-3)
> z
# A tibble: 1,000 x 2
      x      y
  <dbl>  <dbl>
 1  3     0
 2  3.00  0.0188
 3  2.98  0.0371
 4  2.96  0.0546
 5  2.93  0.0707
 6  2.89  0.0850
 7  2.84  0.0971
 8  2.78  0.107
 9  2.72  0.113
10  2.65  0.116
# ... with 990 more rows
> ggplot(...)
```



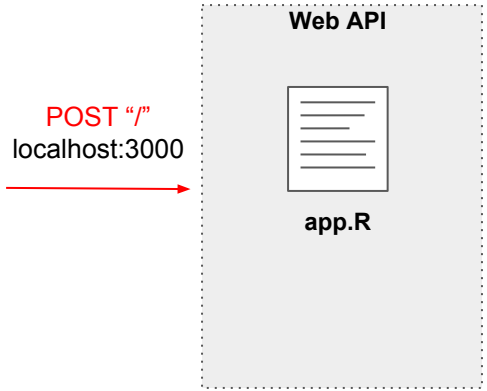

```
> z <- spiro(0,10,-3)
> z
# A tibble: 1,000 x 2
      x      y
  <dbl>  <dbl>
 1  3     0
 2  3.00  0.0440
 3  2.99  0.0877
 4  2.98  0.131
 5  2.97  0.174
 6  2.95  0.215
 7  2.92  0.256
 8  2.90  0.294
 9  2.86  0.332
10  2.83  0.367
# ... with 990 more rows
> ggplot(...)
```



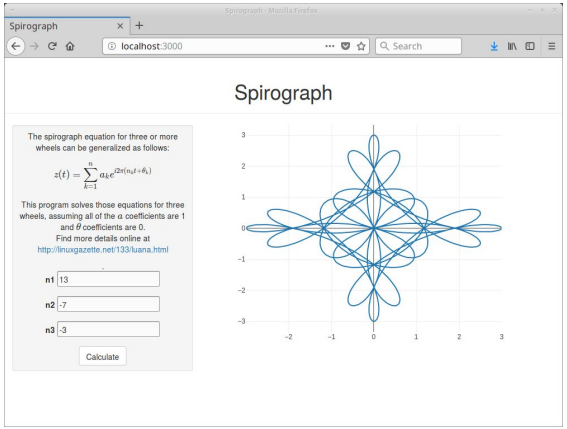
Web Applications



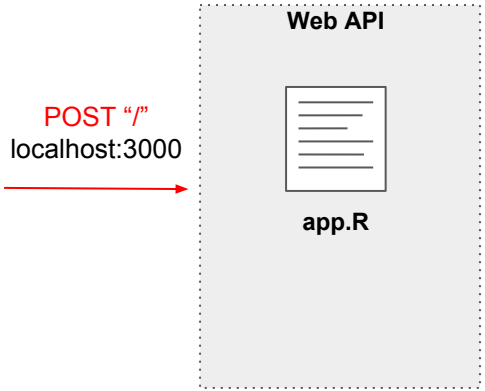
POST “/”
localhost:3000



Building Web Applications with Shiny



POST "/"
localhost:3000

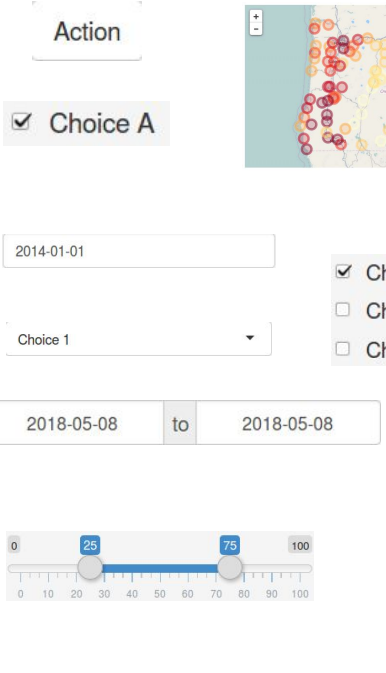



```
spiro <- function(n1,n2,n3) {
  t <- seq(0,1,length.out=1000)
  z <- exp(1i*2*pi*n1*t) +
       exp(1i*2*pi*n2*t) +
       exp(1i*2*pi*n3*t)
  result <- tibble(x=Re(z),y=Im(z))
  return (result)
}
```

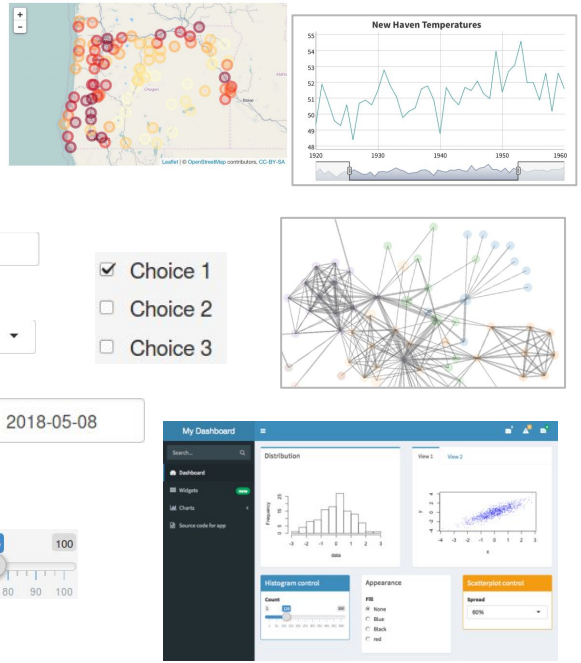
```
# Define UI for spirograph application
ui <- fluidPage(

  # 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

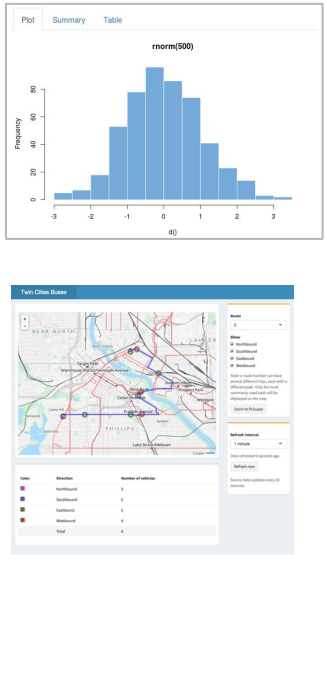
Inputs



Outputs



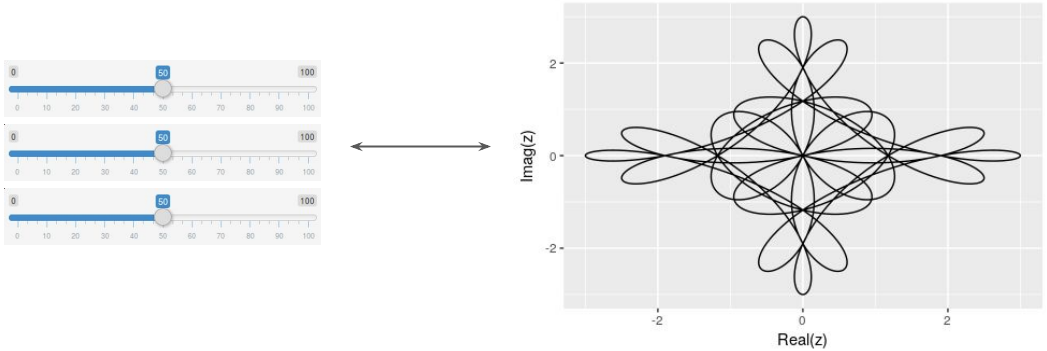
Layouts




```
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

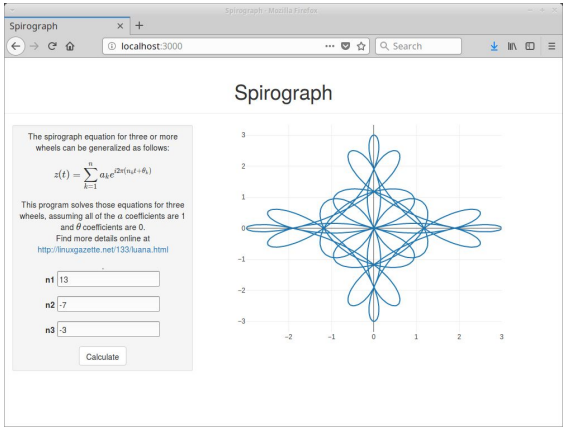
Server glues widgets to science code



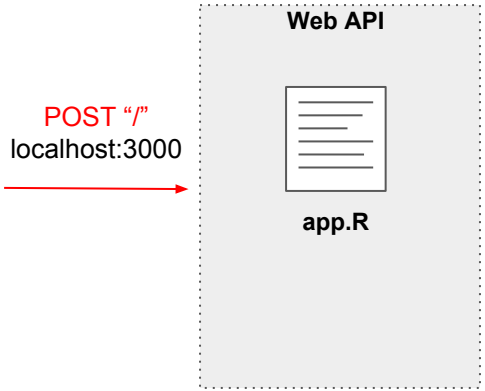

```
spiro <- function(n1,n2,n3) {
  t <- seq(0,1,length.out=1000)
  z <- exp(1i*2*pi*n1*t) +
       exp(1i*2*pi*n2*t) +
       exp(1i*2*pi*n3*t)
  result <- tibble(x=Re(z),y=Im(z))
  return (result)
}

# 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)
```

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

Building Web Applications with Shiny

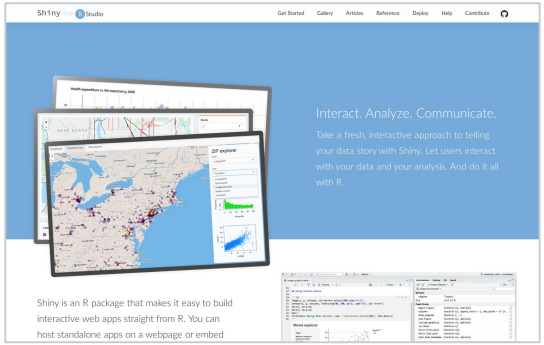


POST "/"
localhost:3000

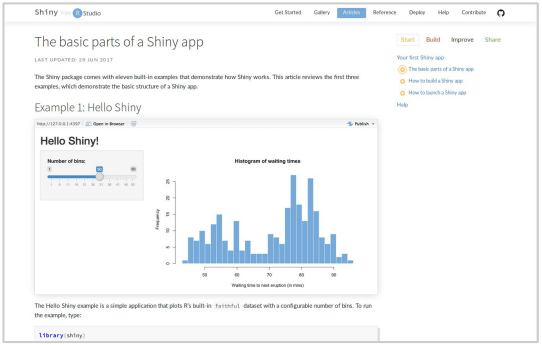



```r
spiro <- function(n1,n2,n3) {
  t <- seq(0,1,length.out=1000)
  z <- exp(1i*2*pi*n1*t) +
       exp(1i*2*pi*n2*t) +
       exp(1i*2*pi*n3*t)
  result <- tibble(x=Re(z),y=Im(z))
  return (result)
}

# 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)
```

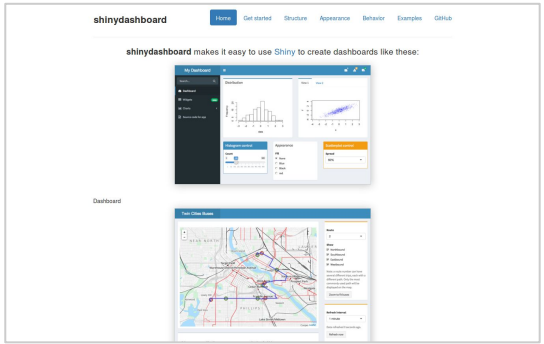
Getting Started with Shiny



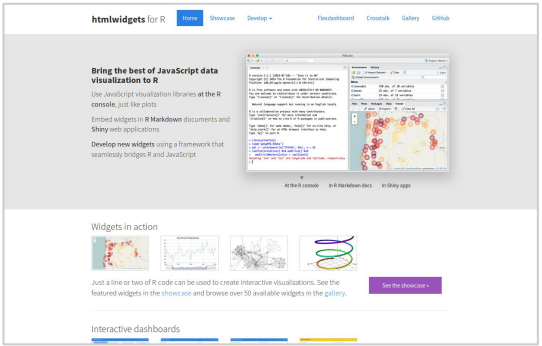
https://shiny.rstudio.com



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

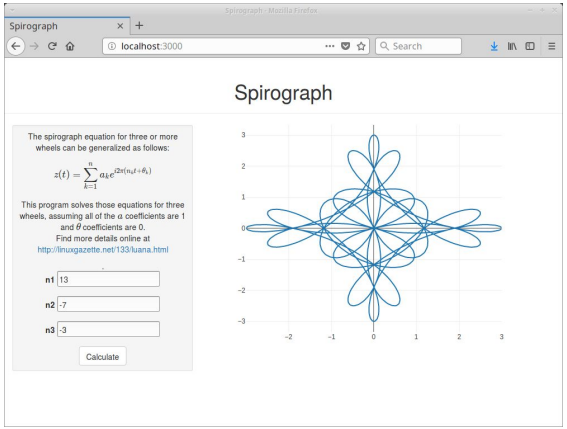


https://htmlwidgets.org

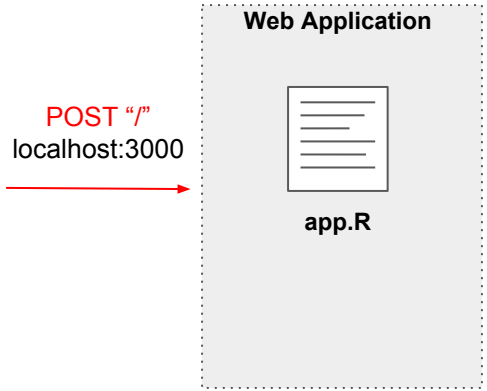


https://rstudio.github.io/shinydashboard

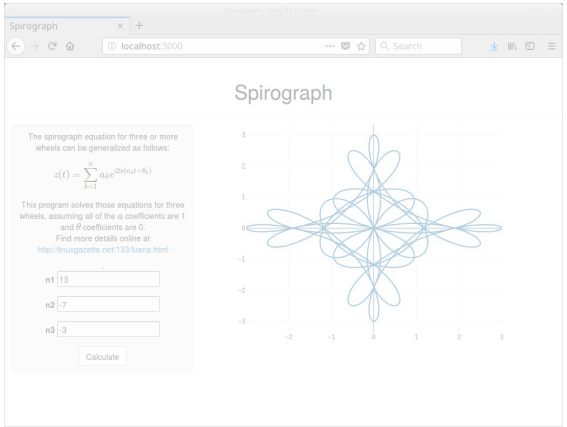
Web Application Basics

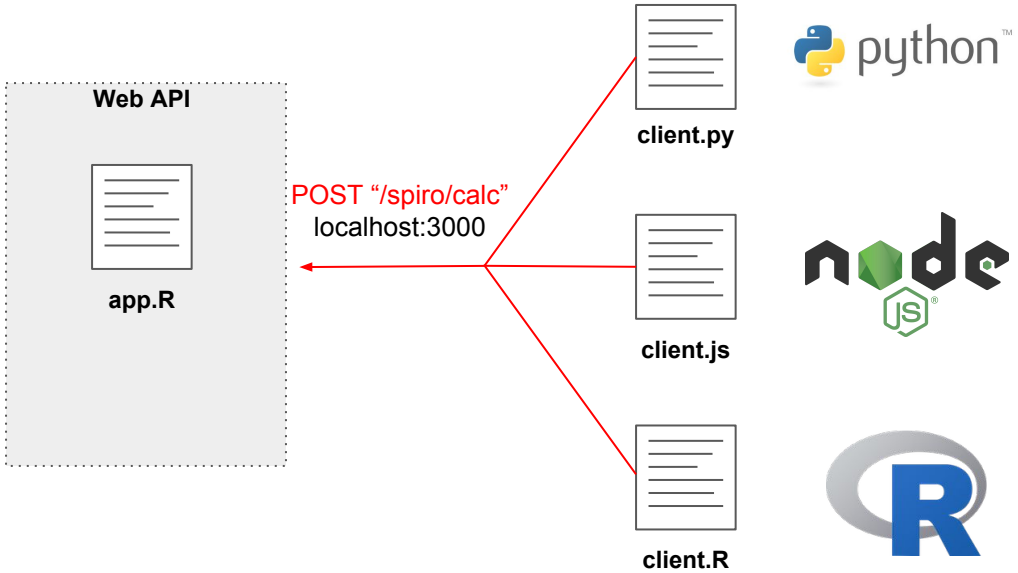


POST "/"
localhost:3000



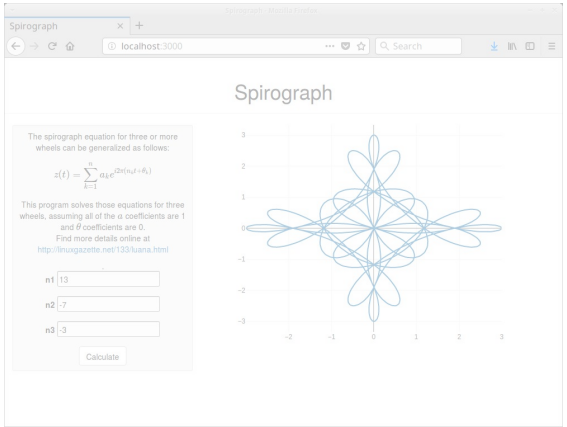
Web API Basics

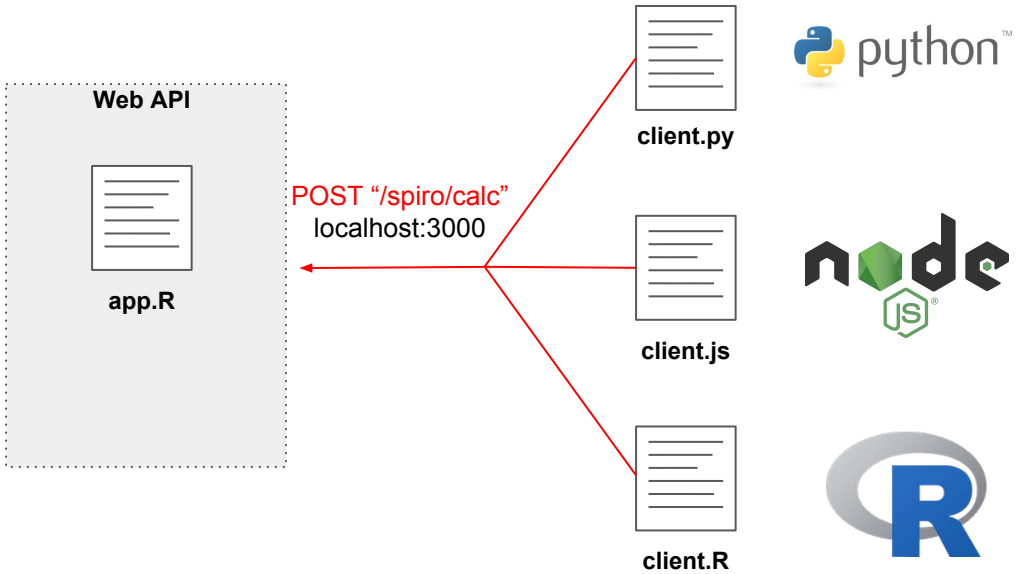




Python logo is a trademark of Python Software Foundation
Node.js is a trademark of Joyent, Inc.

Web API Basics

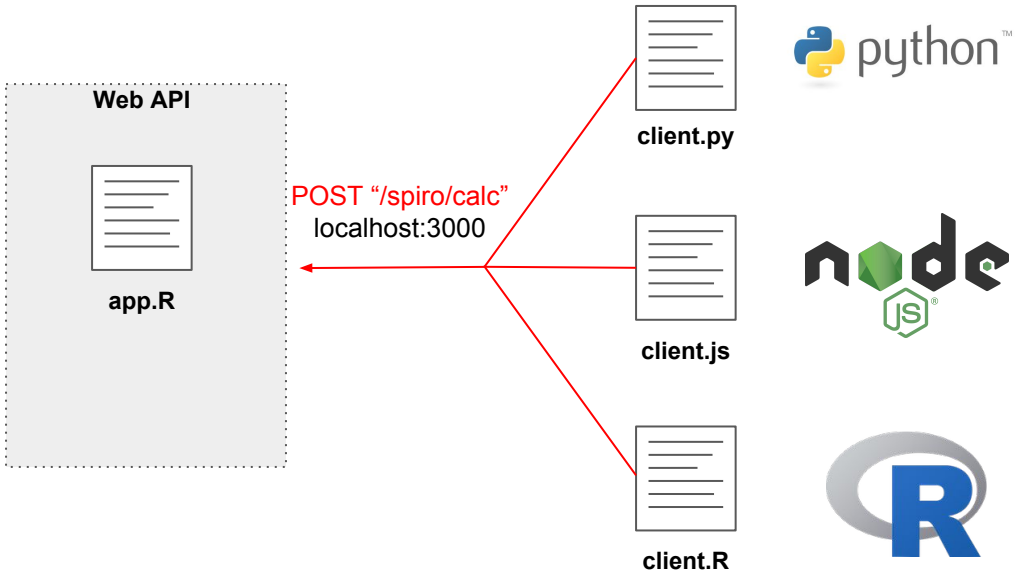




- ✓ Still sending messages
- ✓ Focused on sharing information
- ✓ Communication with other programs

Web API Basics

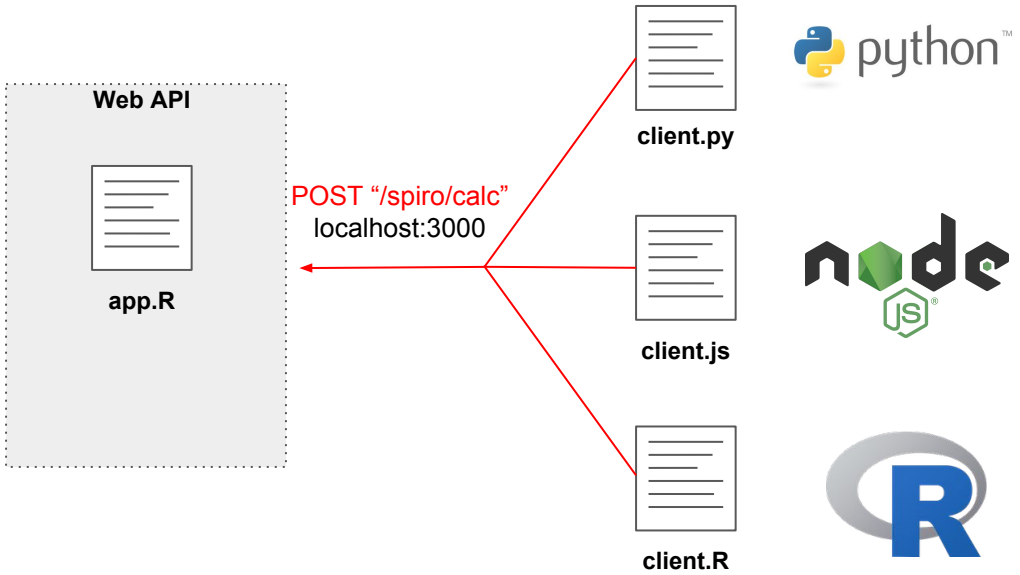
Who's Using Web APIs?



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

Web API Basics

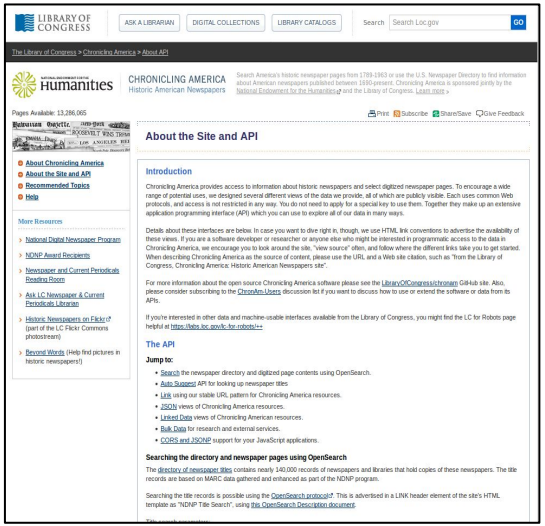
Who Isn't Using Web APIs?

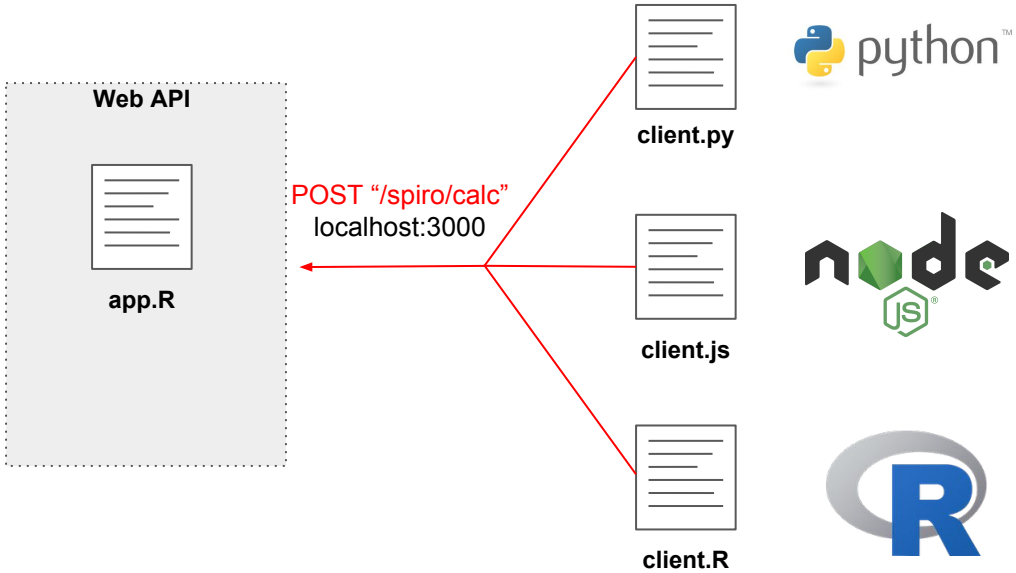


- ✓ Still sending messages
- ✓ Focused on sharing information
- ✓ Communication with other programs

Web API Basics

Who ~~Isn't~~ ^ Using Web APIs?

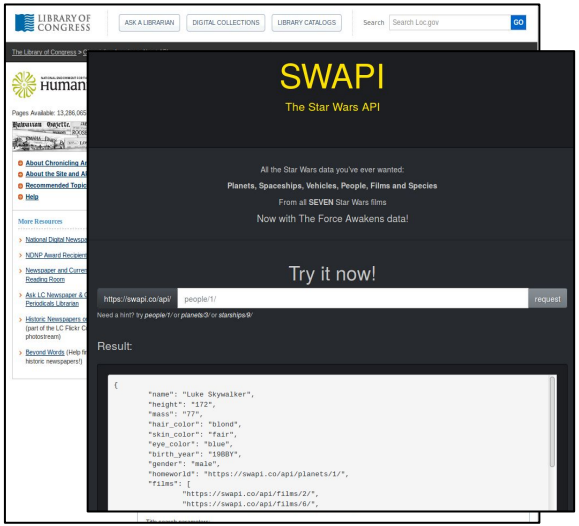


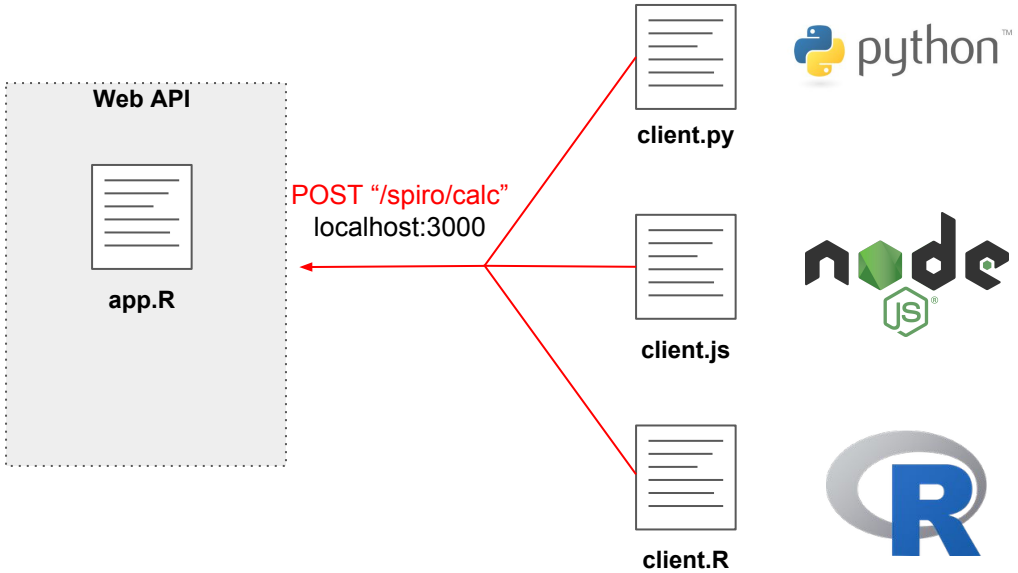


- ✓ Still sending messages
- ✓ Focused on sharing information
- ✓ Communication with other programs

Web API Basics

Who ~~Isn't~~ ^ Using Web APIs?

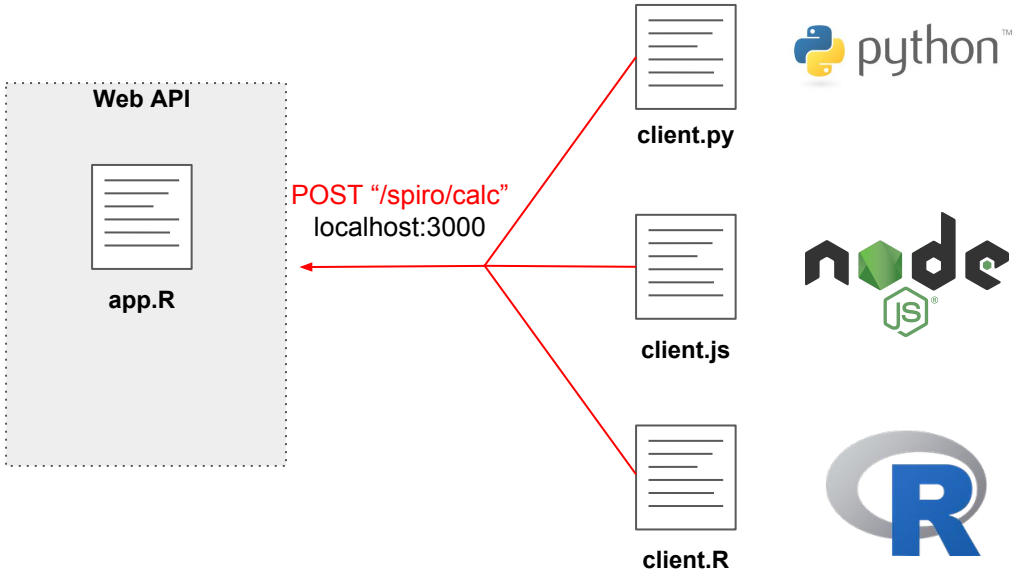




- ✓ Still sending messages
- ✓ Focused on sharing information
- ✓ Communication with other programs

Web API Basics

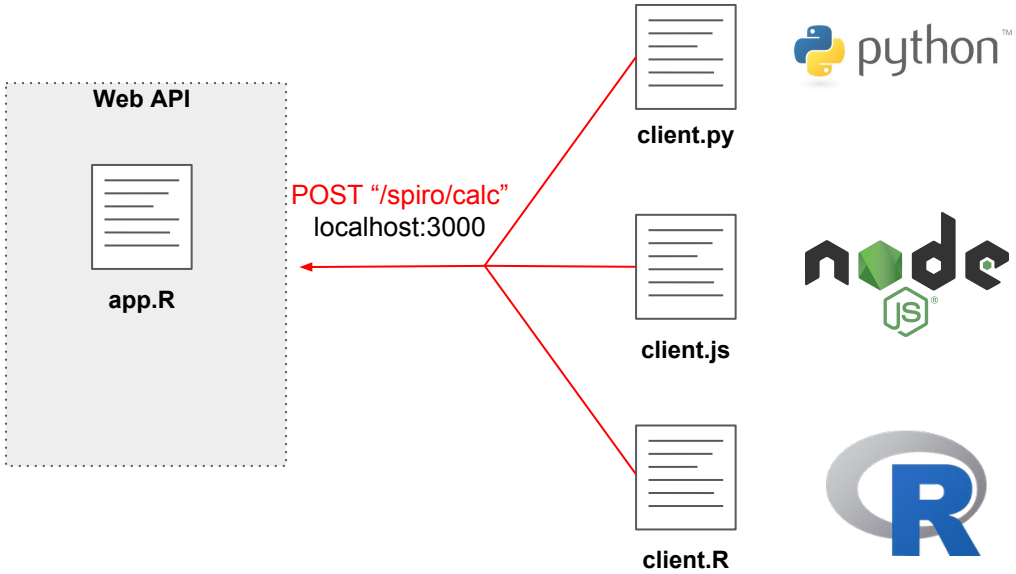
Who Isn't Using Web APIs?



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

Web API Basics

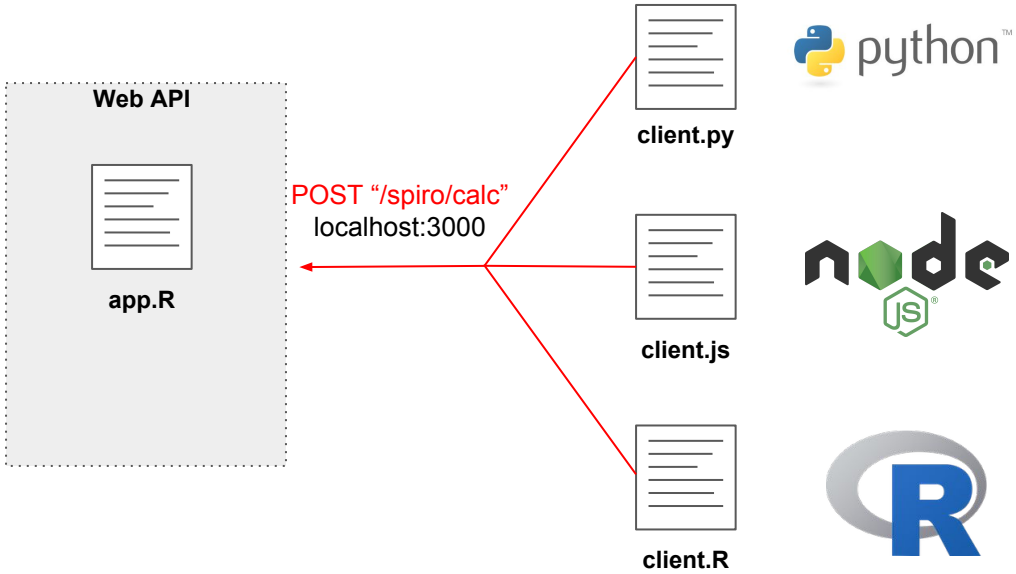
Who ~~Isn't~~ ^ Using Web APIs?



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

Web API Basics

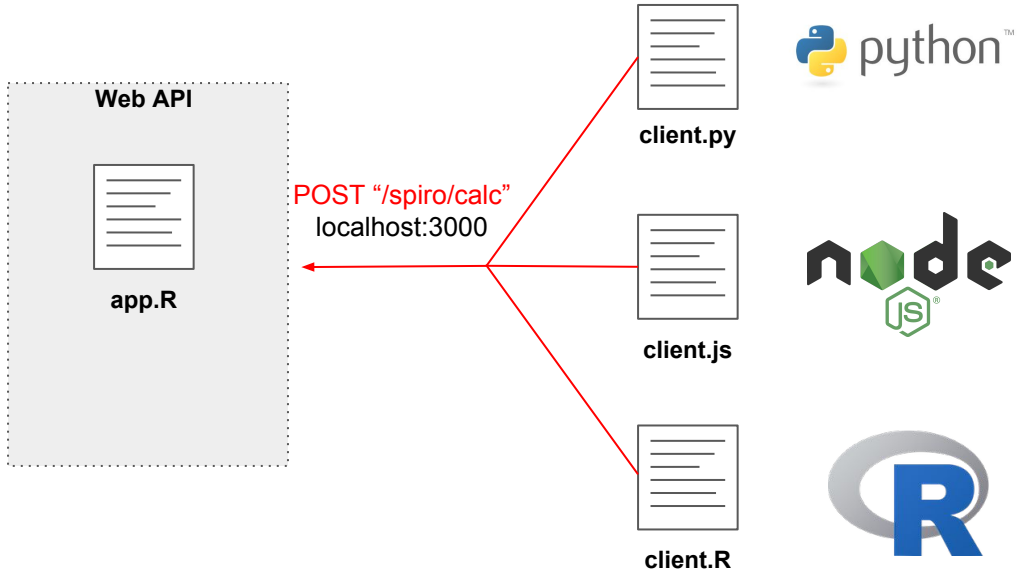
Who ~~Isn't~~ ^ Using Web APIs?



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

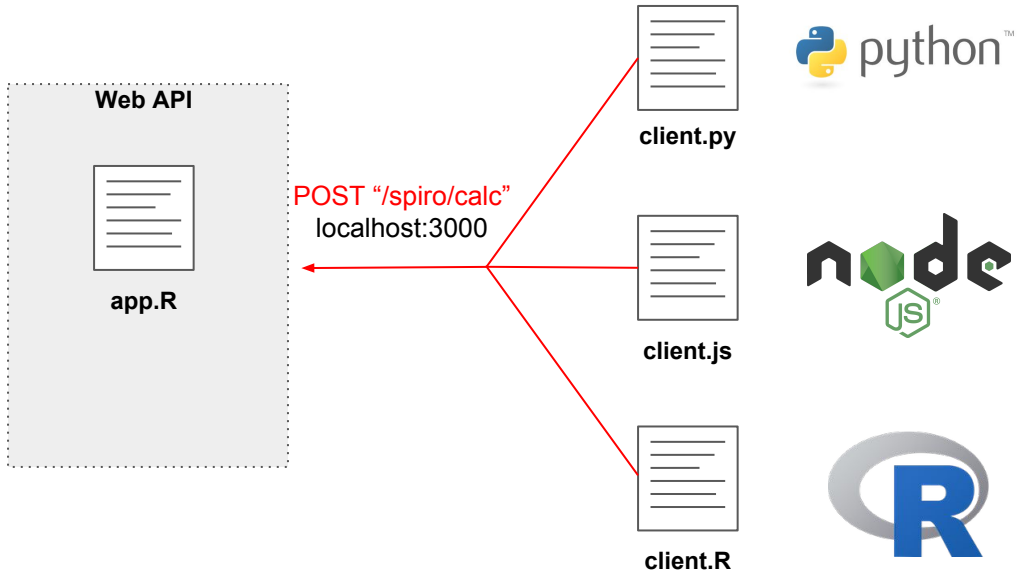
```r
spiro <- function(n1=13,n2=-7,n3=-3) {
  t <- seq(0,1,length.out=1000)
  z <- exp(1i*2*pi*n1*t) +
       exp(1i*2*pi*n2*t) +
       exp(1i*2*pi*n3*t)
  result <- tibble(x=Re(z),y=Im(z))
  return(result)
}
```

Building Web APIs with Plumber



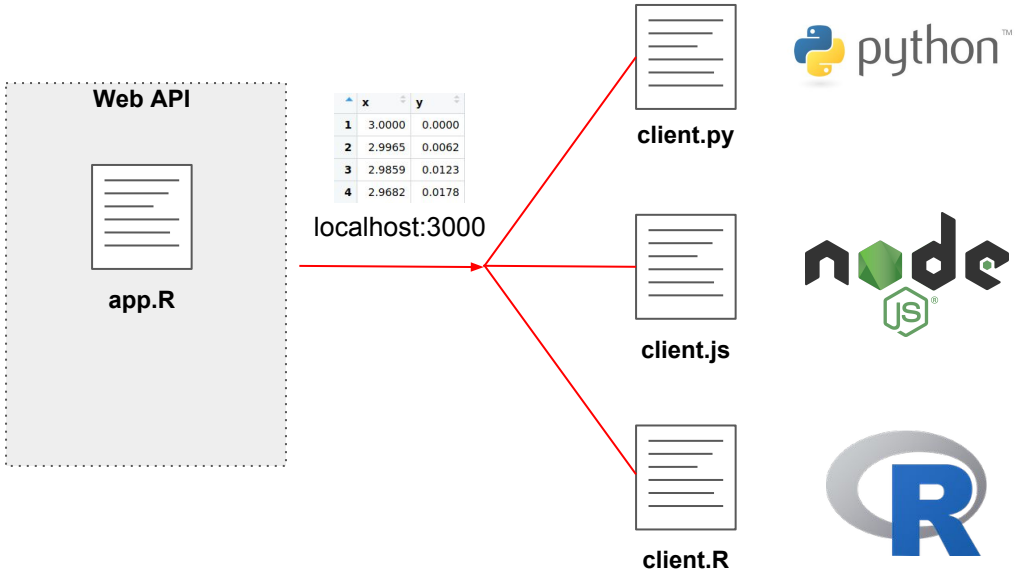
Building Web APIs with Plumber

```
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)))
}
```



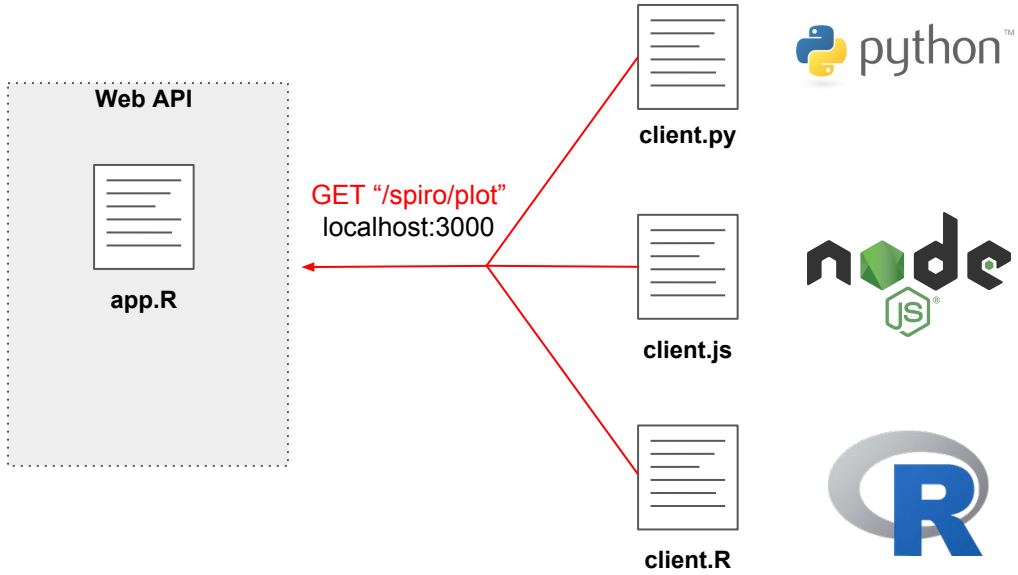
Building Web APIs with Plumber

```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)))
}
```



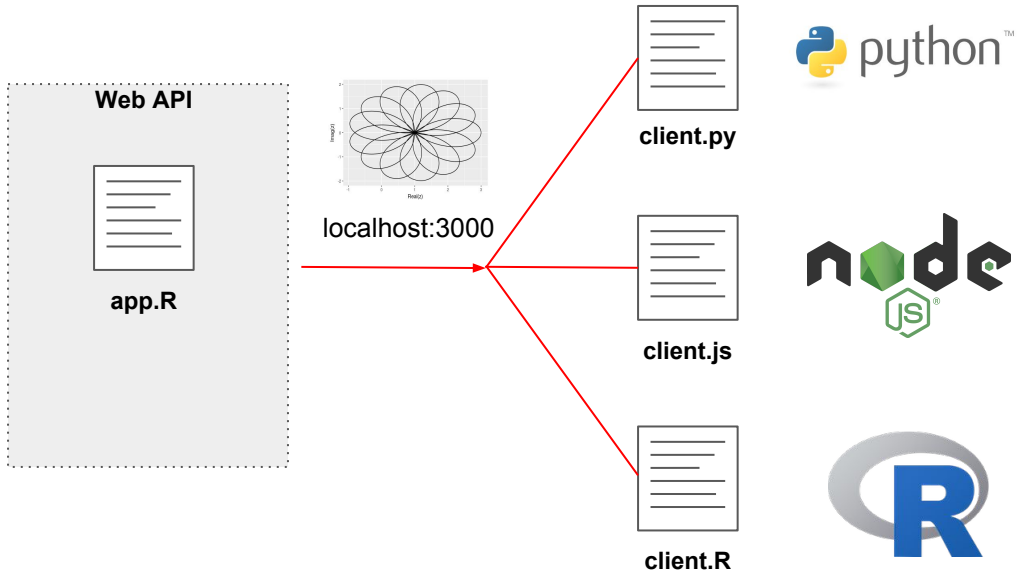
Building Web APIs with Plumber

```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)")
}
```



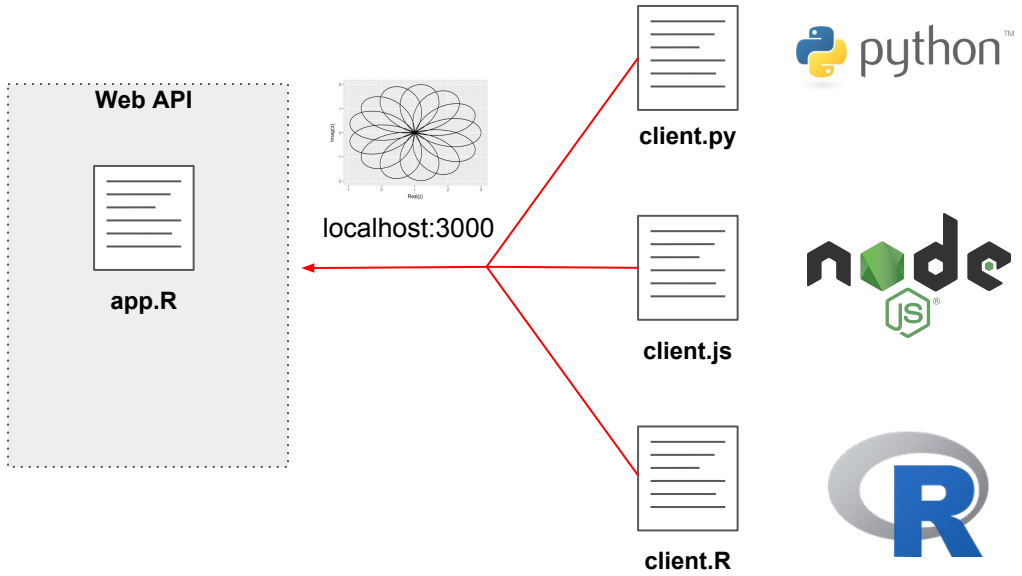
Building Web APIs with Plumber

```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)")
}
```



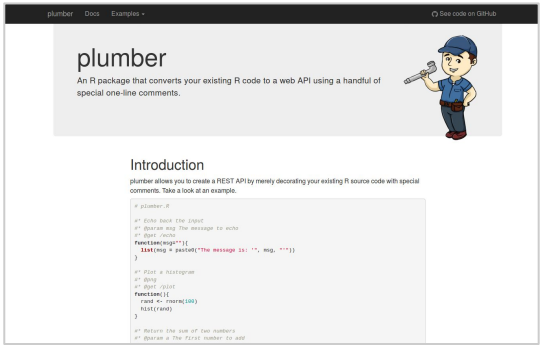
Building Web APIs with Plumber

```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)")
}
```



Try it: https://beta.rstudioconnect.com/connect/#/apps/3533

Getting Started with Plumber



https://www.rplumber.io

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.

Publishing with R Markdown

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

1. Ordered
2. Lists
3. Items

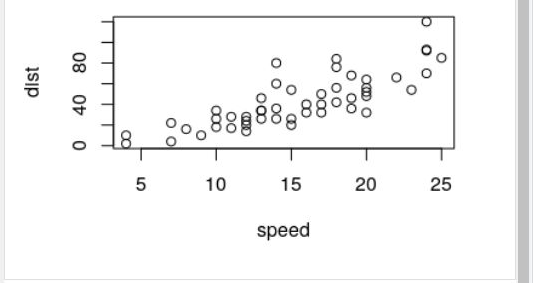
inline links

Publishing with R Markdown

R + Markdown

- ✓ Write reproducible content using the Markdown syntax.
- ✓ Embed chunks of code between lines of narrative in literate programming style.

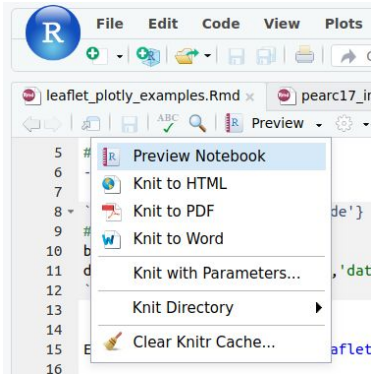
````
---
title: "R Notebook"
output: html_notebook
---

Explore how speed affects stopping distance with the cars dataset.

```{r}
plot(cars)
```
````

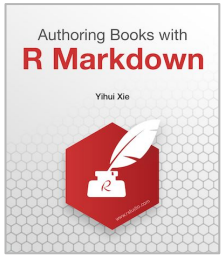


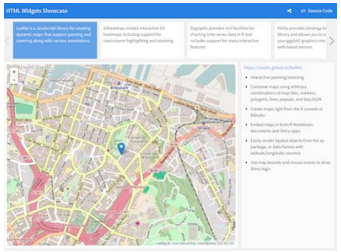
Publishing with R Markdown

R + Markdown

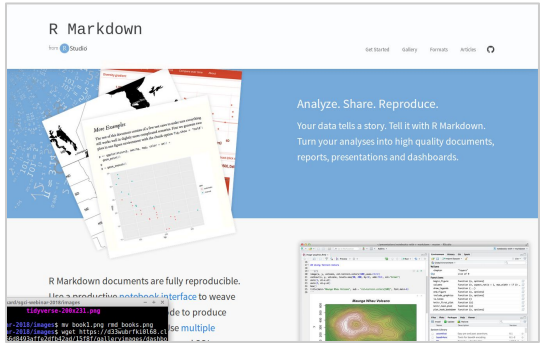
- ✓ Write reproducible content using the Markdown syntax.
- ✓ Embed chunks of code between lines of narrative in literate programming style.
- ✓ Output to multiple formats.



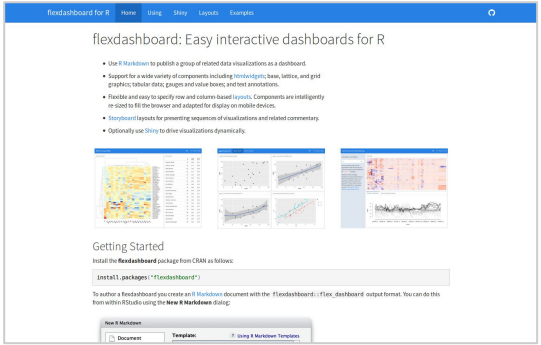




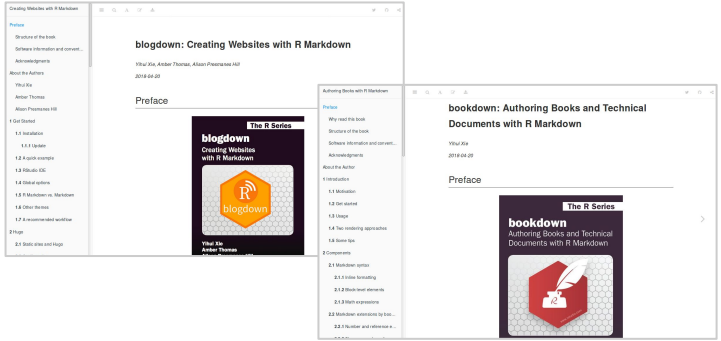
Getting Started with R Markdown



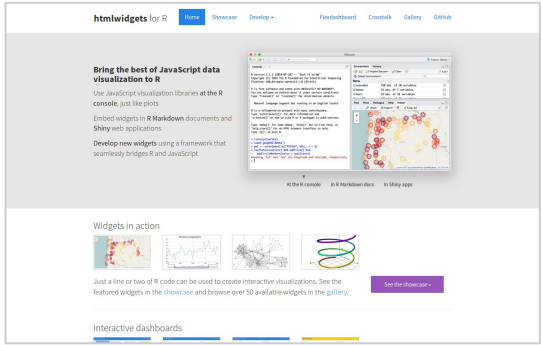
https://rmarkdown.rstudio.com



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

https://shiny.rstudio.com

APIs

https://www.rplumber.io

Articles

https://rmarkdown.rstudio.com

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