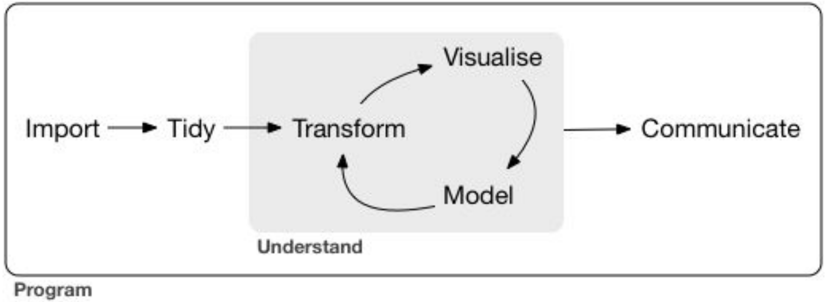
Turning Your Analysis into an API with Plumber

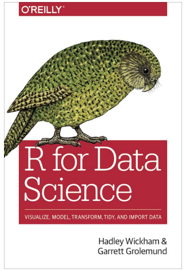
Derrick Kearney

Slides and Examples
https://github.com/dskard/indy-use-r-201805

How to Data Science

Data Science Tools

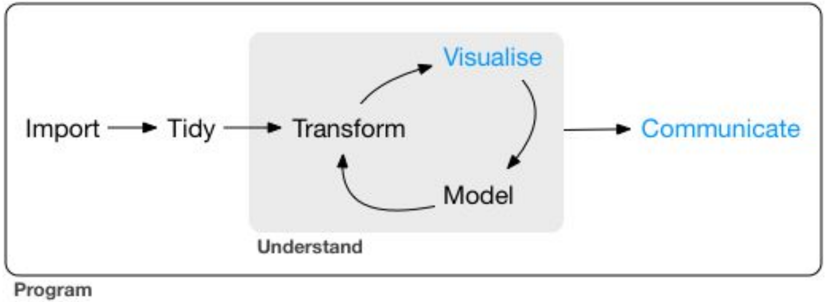


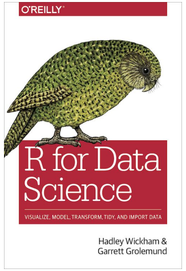


http://r4ds.had.co.nz

How to Data Science

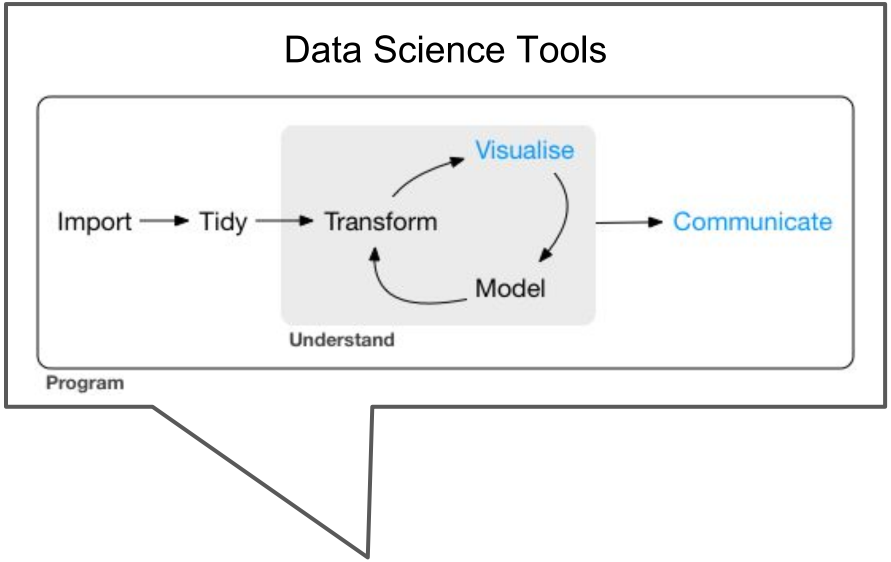
Data Science Tools

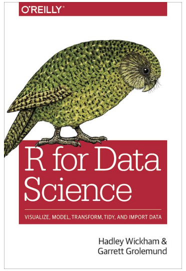




http://r4ds.had.co.nz

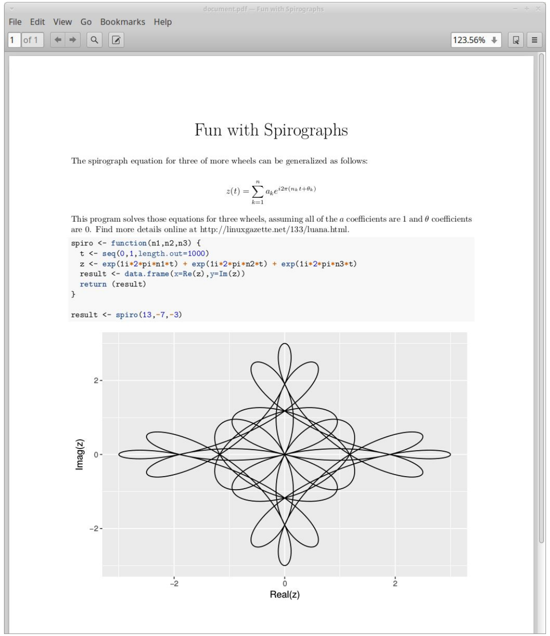
Ways We Communicate





http://r4ds.had.co.nz

✓ Articles / Reports - Sweave, knitr, R Markdown



Fun with Spirographs

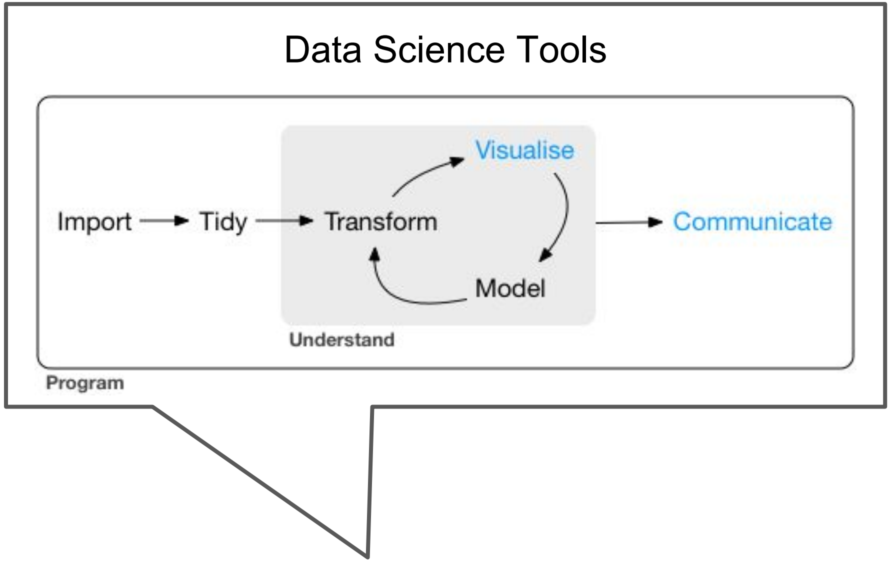
The spirograph equation for three of more wheels can be generalized as follows:

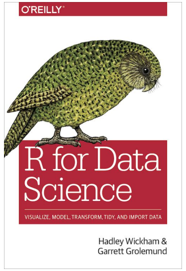
$$z(t) = \sum_{k=1}^{n} a_k e^{i2\pi(n_k t+\theta_k)}$$

This program solves those equations for three wheels, assuming all of the a coefficients are 1 and θ coefficients are 0. Find more details online at http://linuxgazette.net/133/luana.html.

```
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 <- data.frame(x=Re(z),y=Im(z))
  return (result)
}

result <- spiro(13,-7,-3)
```

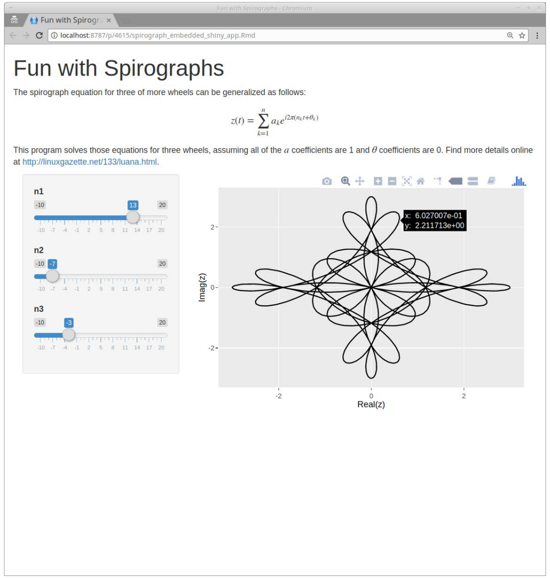
Ways We Communicate





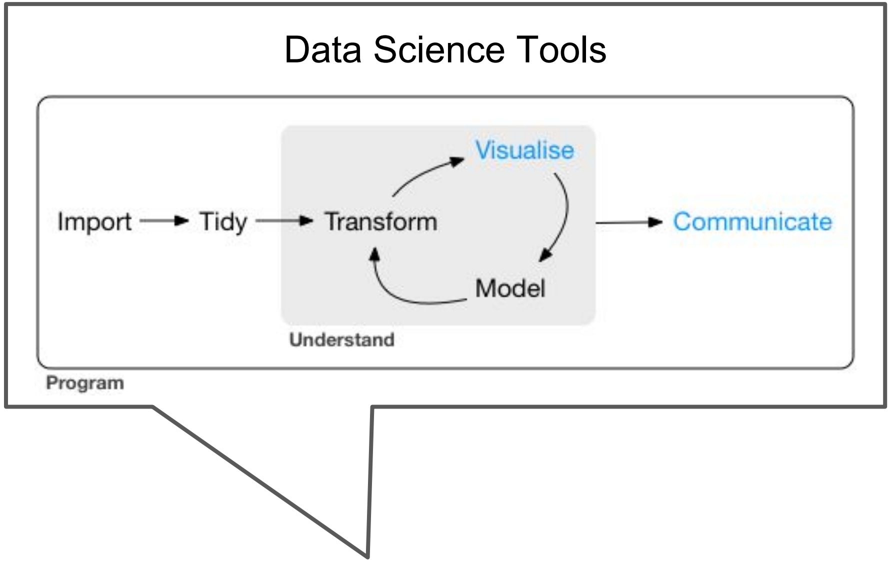
http://r4ds.had.co.nz

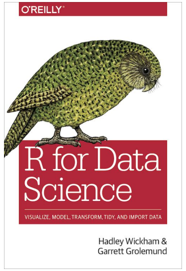
- ✓ Articles / Reports - Sweave, knitr, R Markdown
- ✓ Applications - Shiny



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

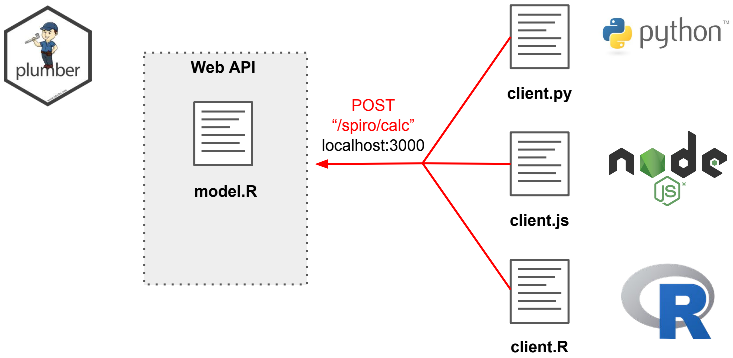
Ways We Communicate





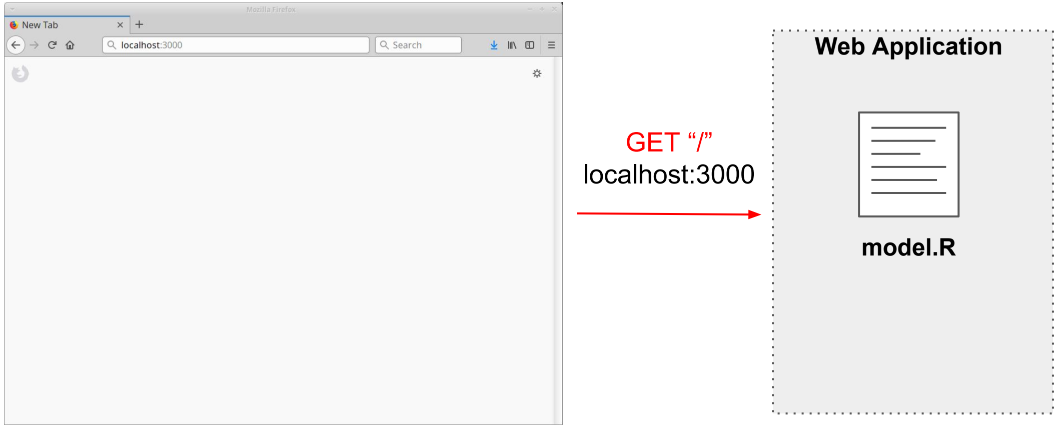
http://r4ds.had.co.nz

- ✓ Articles / Reports - Sweave, knitr, R Markdown
- ✓ Applications - Shiny
- ✓ APIs - Plumber

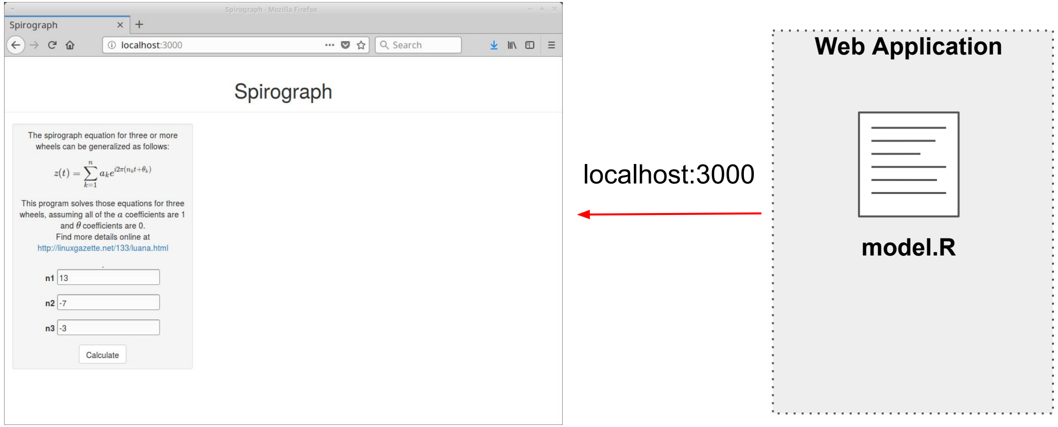


RStudio, Shiny, and R Markdown are trademarks of RStudio, Inc.
Hex Images from rstudio.com
Python logo is a trademark of Python Software Foundation
Node.js is a trademark of Joyent, Inc.

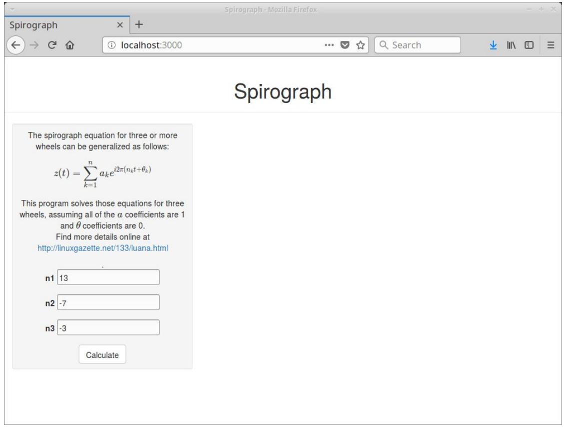
Web Application Basics

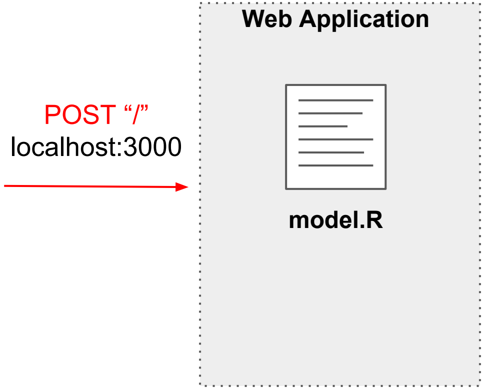


Web Application Basics

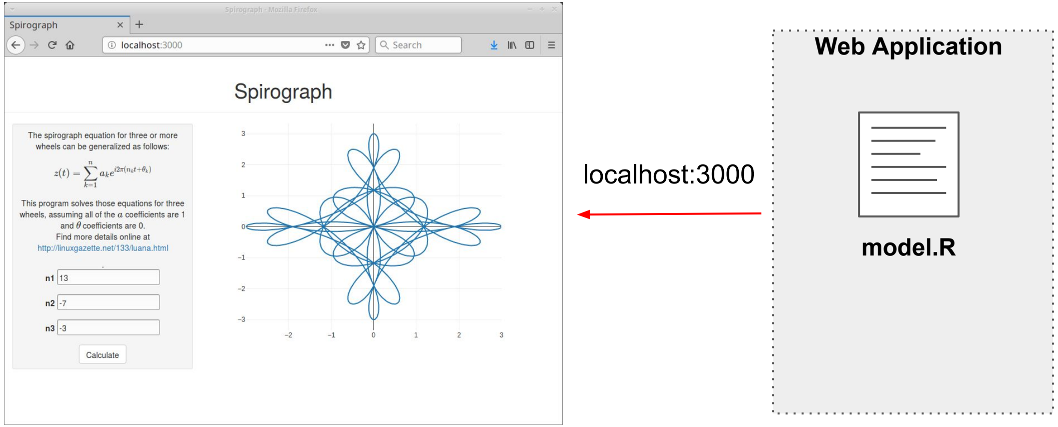


Web Application Basics

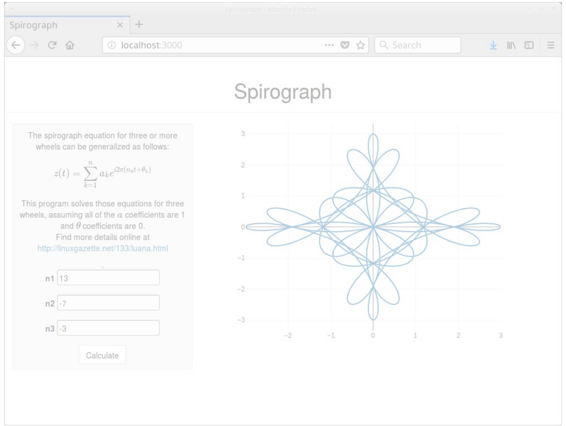


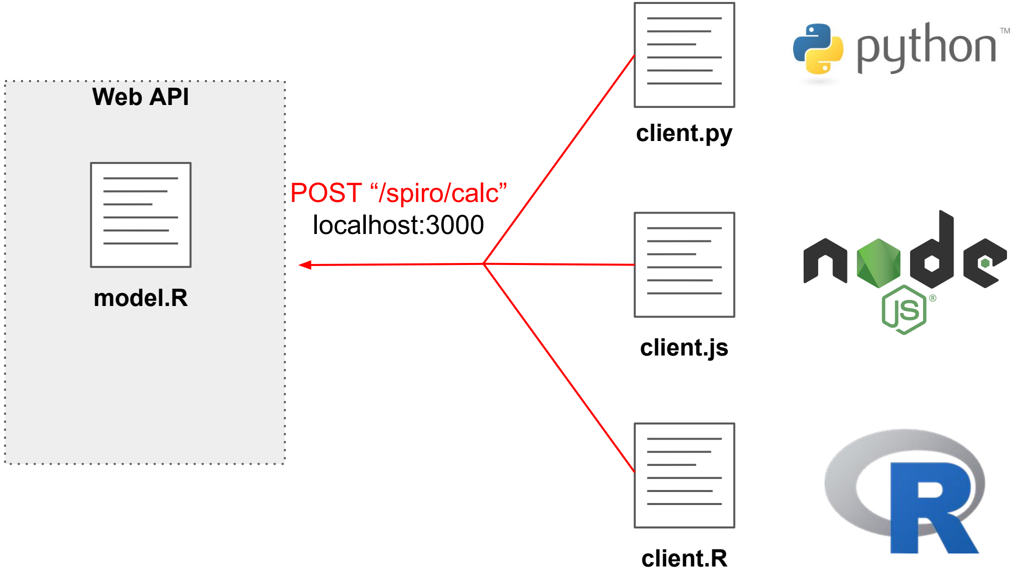


Web Application Basics



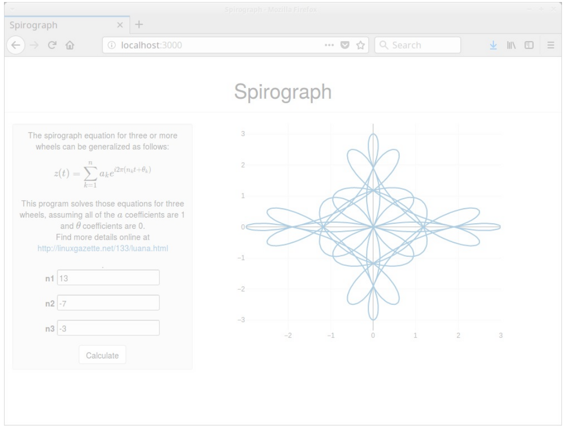
Web API Basics

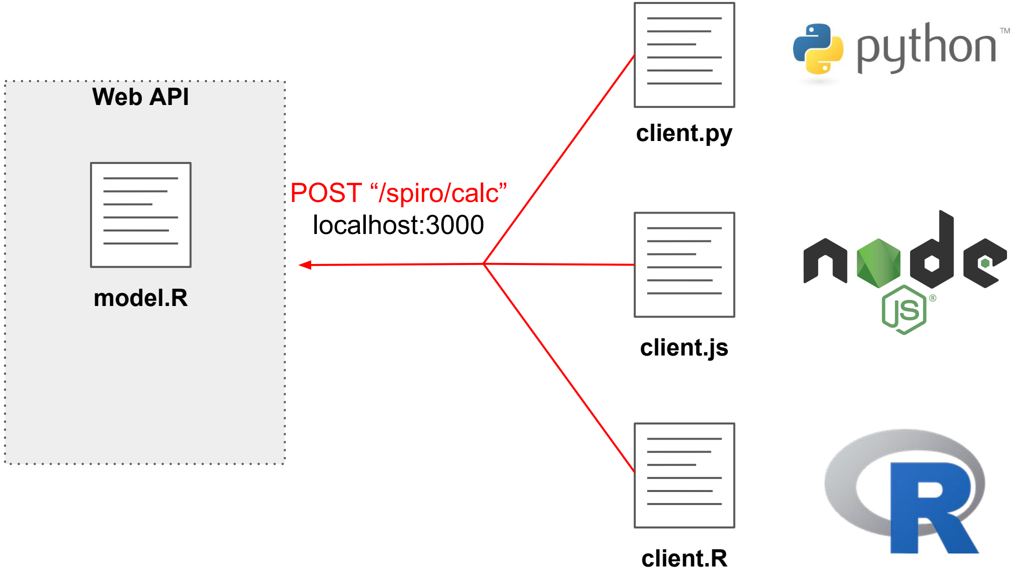




Python logo is a trademark of Python Software Foundation
Node.js is a trademark of Joyent, Inc.

Web API Basics

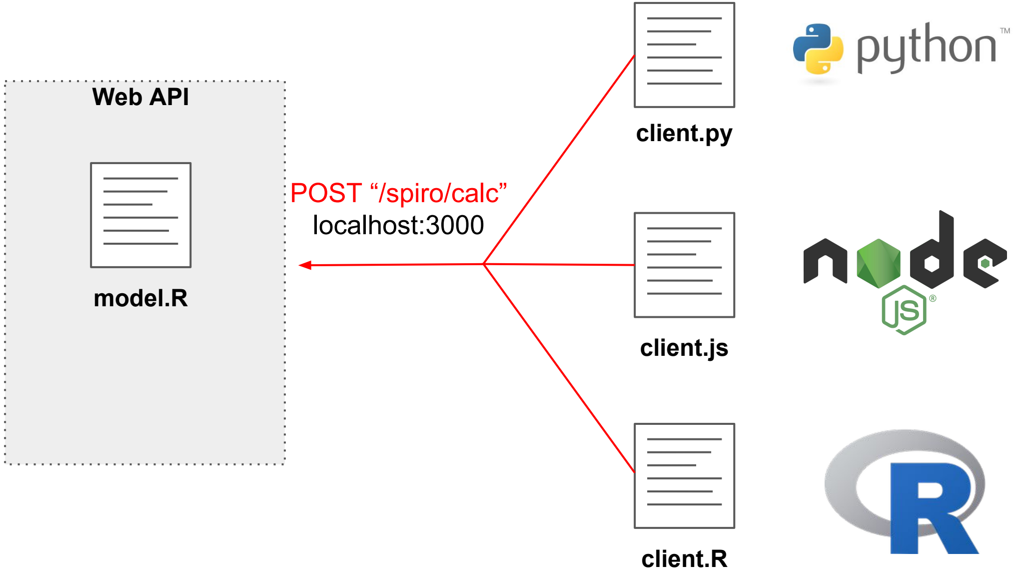




- ✓ Still sending messages
- ✓ Focused on sharing information
- ✓ Communication with other programs

Web API Basics

Who’s Using Web APIs?

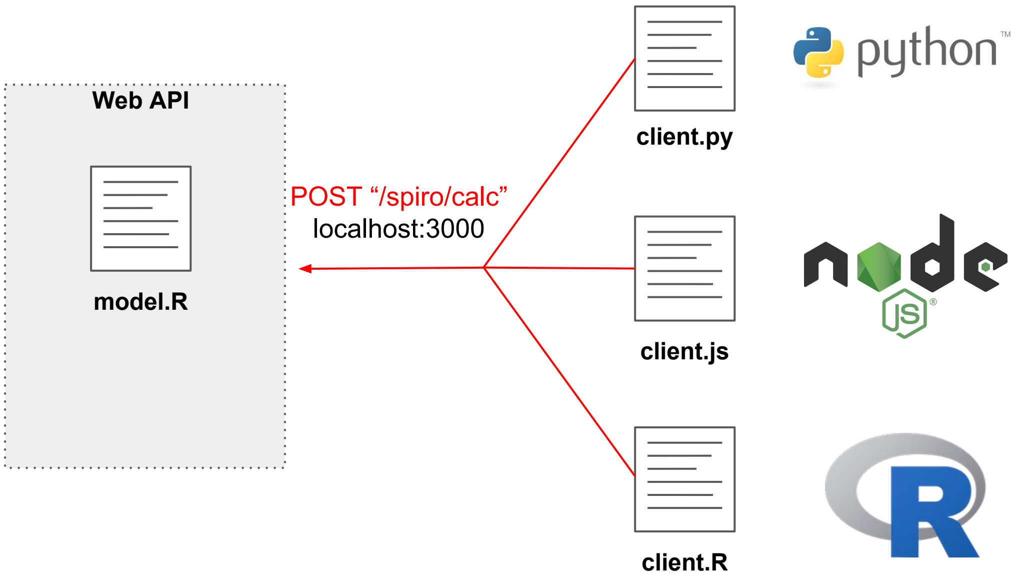


- ✓ Still sending messages
- ✓ Focused on sharing information
- ✓ Communication with other programs

Web API Basics

Isn’t

<u>Who ^ Using Web APIs?</u>

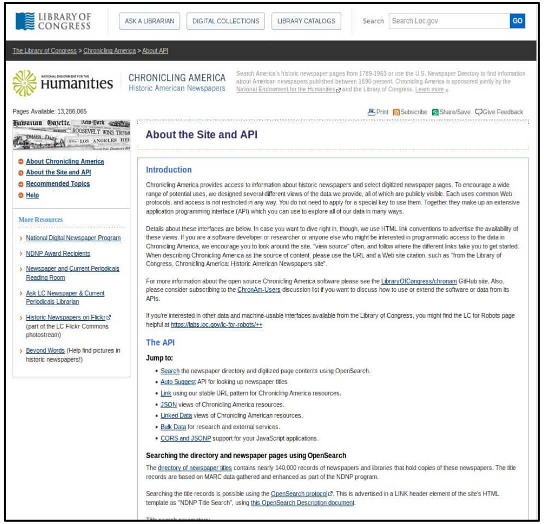


- ✓ Still sending messages
- ✓ Focused on sharing information
- ✓ Communication with other programs

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

Web API Basics

Who Isn't ^ Using Web APIs?


LIBRARY OF CONGRESS | ASK A LIBRARIAN | DIGITAL COLLECTIONS | LIBRARY CATALOGS | Search | Search Loc.gov | GO

The Library of Congress > Chronicling America > About API

National Endowment for the Humanities

CHRONICLING AMERICA
Historic American Newspapers

Search America's historic newspaper pages from 1789-1963 or use the U.S. Newspaper Directory to find information about American newspapers published between 1690-present. Chronicling America is sponsored jointly by the National Endowment for the Humanities and the Library of Congress. Learn more »

Pages Available: 13,286,065

Print | Subscribe | Share/Save | Give Feedback

- About Chronicling America
- About the Site and API
- Recommended Topics
- Help

More Resources

- National Digital Newspaper Program
- NDNP Award Recipients
- Newspaper and Current Periodicals Reading Room
- Ask LC Newspaper & Current Periodicals Librarian
- Historic Newspapers on Flickr (part of the LC Flickr Commons photostream)
- Beyond Words (Help find pictures in historic newspapers!)

About the Site and API

Introduction

Chronicling America provides access to information about historic newspapers and select digitized newspaper pages. To encourage a wide range of potential uses, we designed several different views of the data we provide, all of which are publicly visible. Each uses common Web protocols, and access is not restricted in any way. You do not need to apply for a special key to use them. Together they make up an extensive application programming interface (API) which you can use to explore all of our data in many ways.

Details about these interfaces are below. In case you want to dive right in, though, we use HTML link conventions to advertise the availability of these views. If you are a software developer or researcher or anyone else who might be interested in programmatic access to the data in Chronicling America, we encourage you to look around the site, "view source" often, and follow where the different links take you to get started. When describing Chronicling America as the source of content, please use the URL and a Web site citation, such as "from the Library of Congress, Chronicling America: Historic American Newspapers site".

For more information about the open source Chronicling America software please see the LibraryOfCongress/chronam GitHub site. Also, please consider subscribing to the ChronAm-Users discussion list if you want to discuss how to use or extend the software or data from its APIs.

If you're interested in other data and machine-usable interfaces available from the Library of Congress, you might find the LC for Robots page helpful at https://labs.loc.gov/lc-for-robots/++

The API

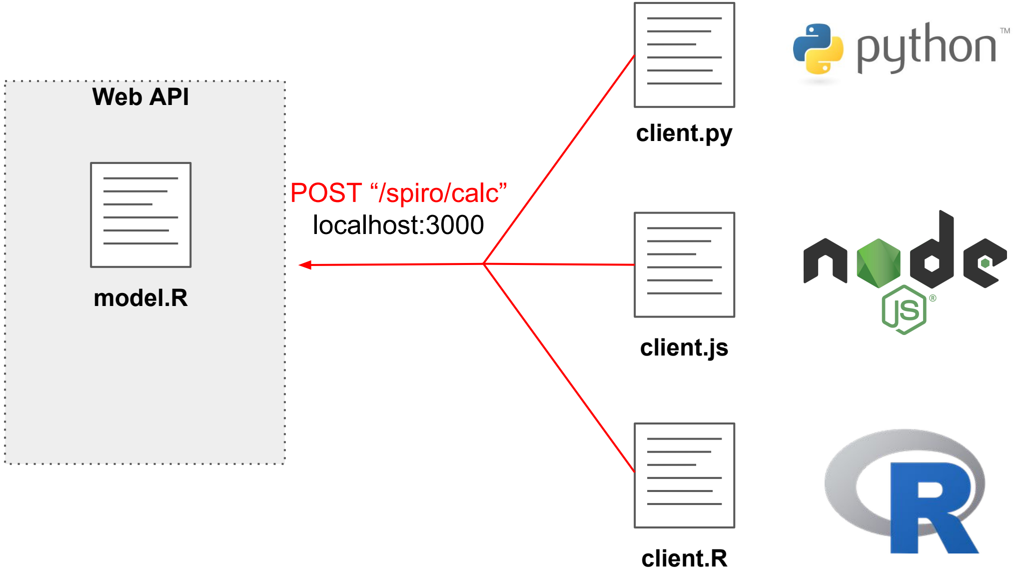
Jump to:

- Search the newspaper directory and digitized page contents using OpenSearch.
- Auto Suggest API for looking up newspaper titles
- Link using our stable URL pattern for Chronicling America resources.
- JSON views of Chronicling America resources.
- Linked Data views of Chronicling American resources.
- Bulk Data for research and external services.
- CORS and JSONP support for your JavaScript applications.

Searching the directory and newspaper pages using OpenSearch

The directory of newspaper titles contains nearly 140,000 records of newspapers and libraries that hold copies of these newspapers. The title records are based on MARC data gathered and enhanced as part of the NDNP program.

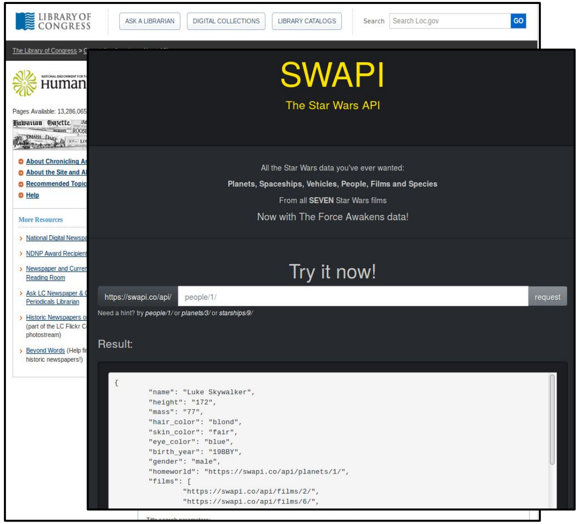
Searching the title records is possible using the OpenSearch protocol. This is advertised in a LINK header element of the site's HTML template as "NDNP Title Search", using this OpenSearch Description document.

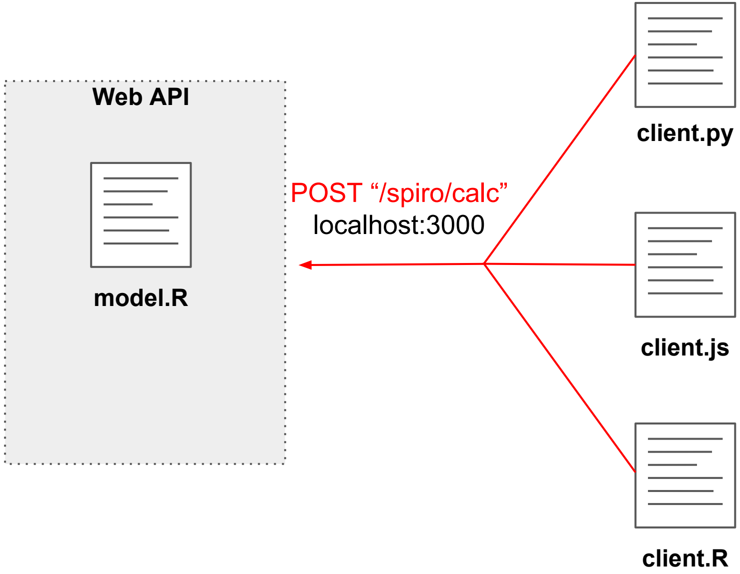


- ✓ Still sending messages
- ✓ Focused on sharing information
- ✓ Communication with other programs

Web API Basics

Who Isn't ^ Using Web APIs?

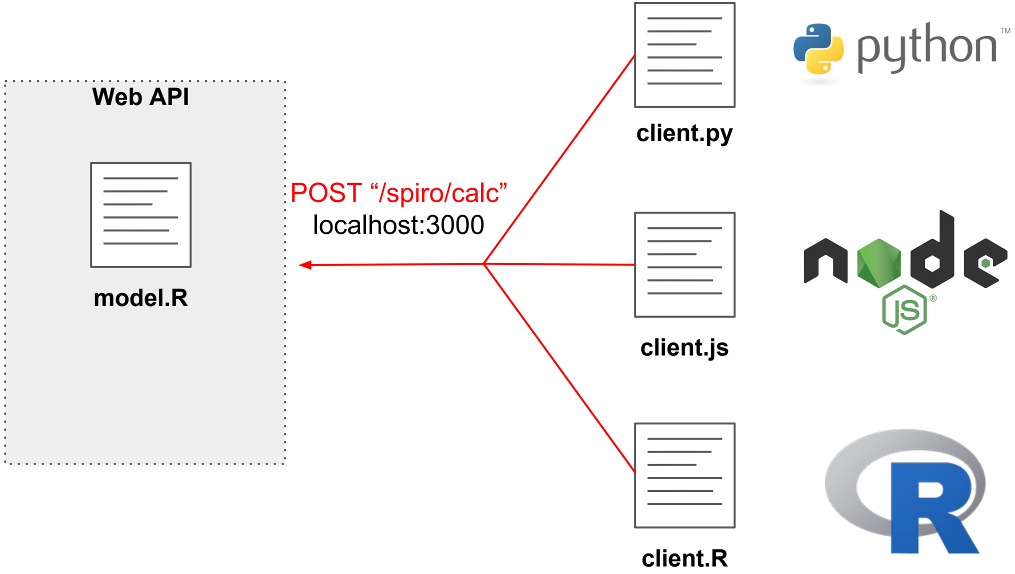




- ✓ Still sending messages
- ✓ Focused on sharing information
- ✓ Communication with other programs

Web API Basics

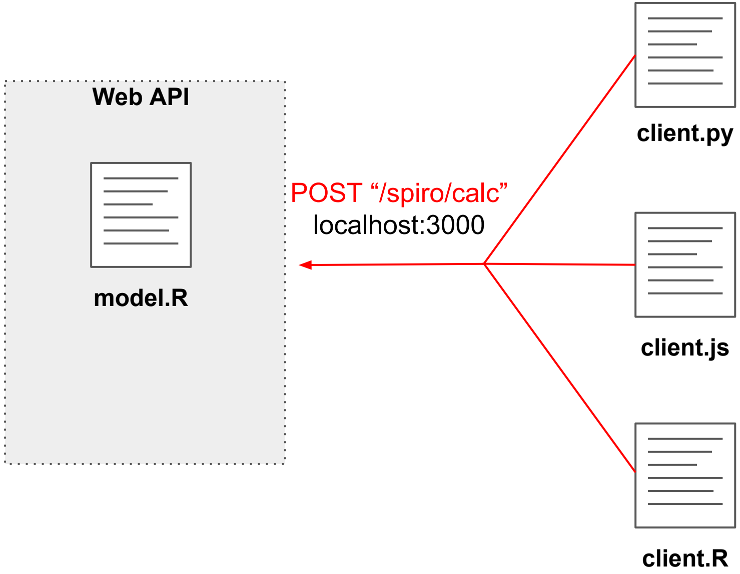
Who ^Isn't Using Web APIs?



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

Web API Basics

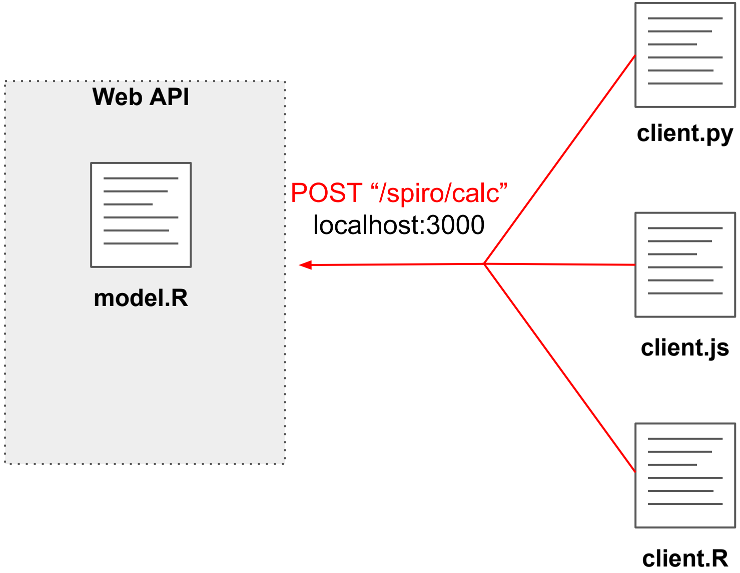
Who Isn't ^ Using Web APIs?



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

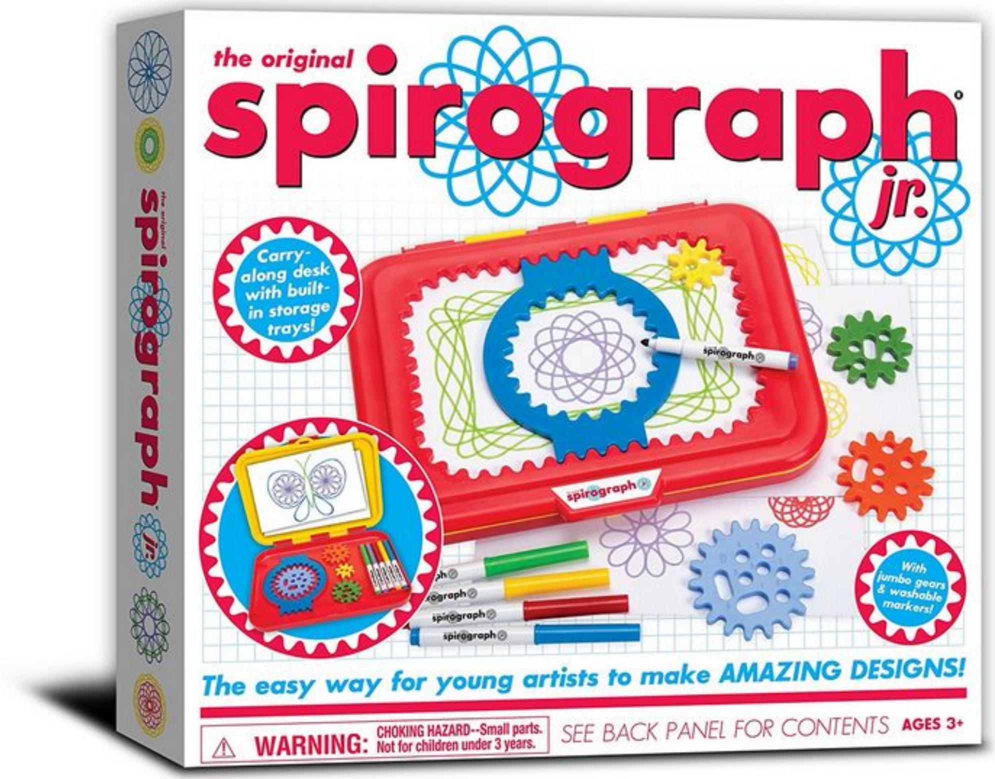
Web API Basics

Who Isn't ^ Using Web APIs?



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

Spirographs… For Ages 3+



Spirographs… For Ages 3+

<-- prev | next -->

Plotting the spirograph equations with 'gnuplot'

By Victor 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.

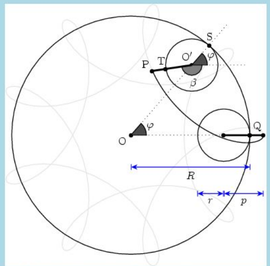


Figure 1 Geometry for the hypotrochoid equations. The grayed figure corresponds to R=9, r=2, and p=3.

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

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.

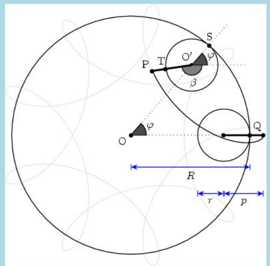


Figure 1 Geometry for the hypotrochoid equations. The grayed figure corresponds to R=9, r=2, and p=3.

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.

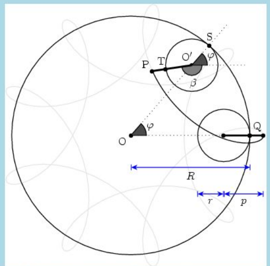


Figure 1 Geometry for the hypotrochoid equations. The grayed figure corresponds to R=9, r=2, and p=3.

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

TL;DR:

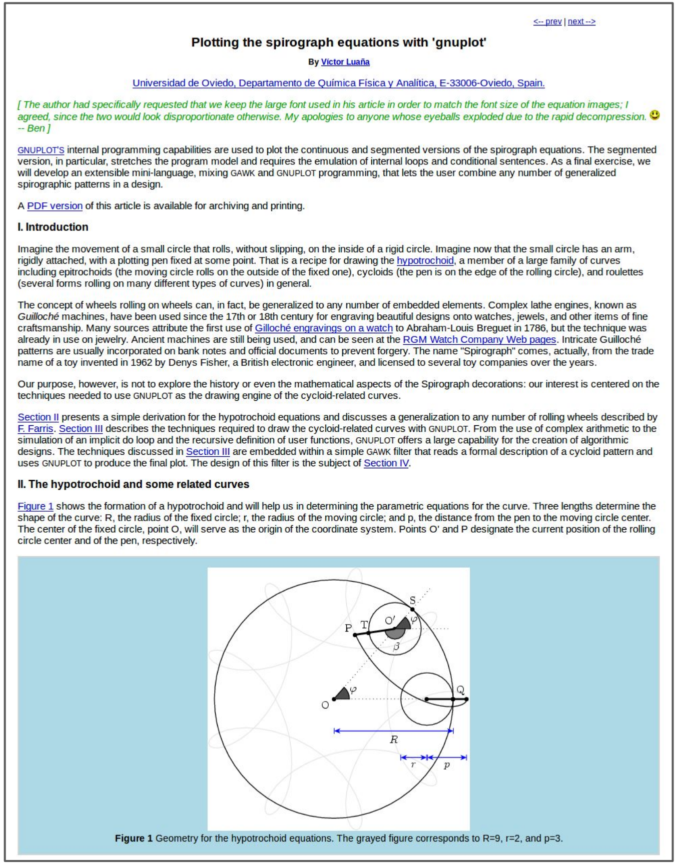
1. We can model spirographs as a sum of exponentials:

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

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

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:

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

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

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

2. We can model a 3-wheeled system, where α_k = 1 and θ_k = 0, with this equation:

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

3. Need three inputs from the user: $\mathbf{n_1}$, $\mathbf{n_2}$, and $\mathbf{n_3}$

Spirographs… For Ages 3+

```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:

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

2. We can model a 3-wheeled system, where α_k = 1 and θ_k = 0, with this equation:

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

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:

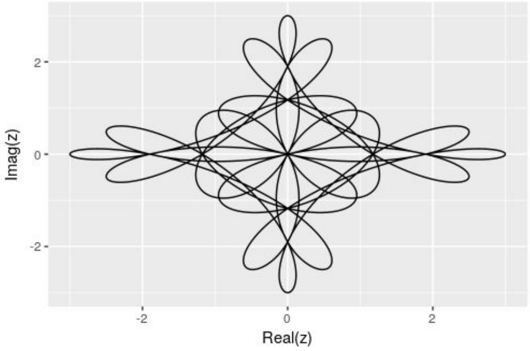
$$z(t) = e^{i2\pi(n_1 t)} + e^{i2\pi(n_2 t)} + e^{i2\pi(n_3 t)}, t \in [0,1]$$

3. Need three inputs from the user: $\mathbf{n_1}$, $\mathbf{n_2}$, 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)
}
```

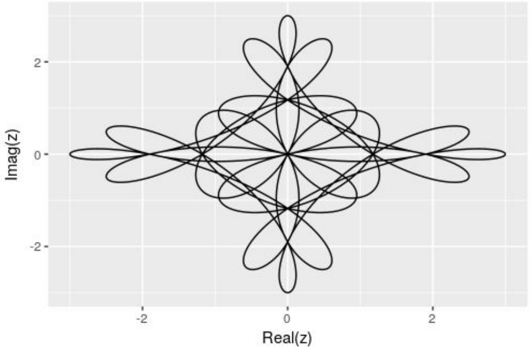
```
> z <- spiro(13,-7,-3)
> z
# A tibble: 1,000 x 2
       x      y
   <dbl>  <dbl>
 1  3     0
 2  3.00 0.0188
 3  2.98 0.0371
 4  2.96 0.0546
 5  2.93 0.0707
 6  2.89 0.0850
 7  2.84 0.0971
 8  2.78 0.107
 9  2.72 0.113
10  2.65 0.116
# ... with 990 more rows
> ggplot(...)
```



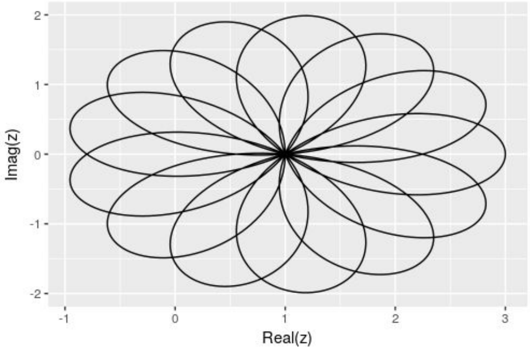
Spirographs… For Ages 3+

```
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(...)
```

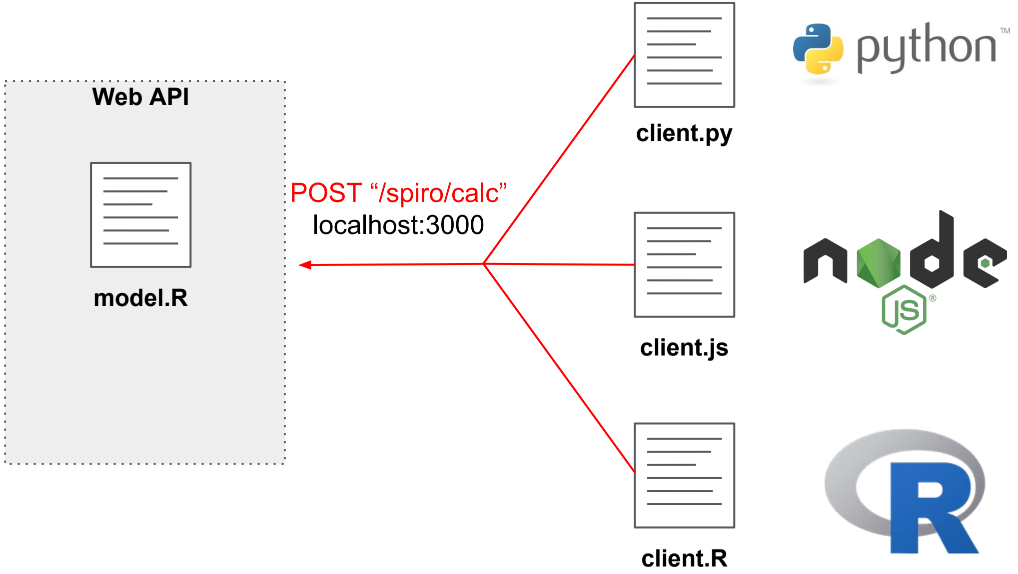



```
> z <- spiro(0,10,-3)
> z
# A tibble: 1,000 x 2
        x      y
    <dbl>  <dbl>
 1  3     0
 2  3.00 0.0440
 3  2.99 0.0877
 4  2.98 0.131
 5  2.97 0.174
 6  2.95 0.215
 7  2.92 0.256
 8  2.90 0.294
 9  2.86 0.332
10  2.83 0.367
# ... with 990 more rows
> ggplot(...)
```



```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

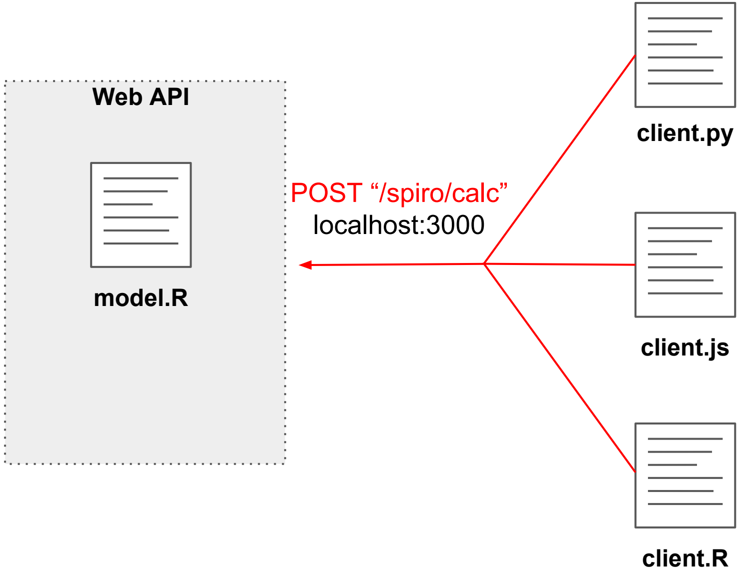


Building Web APIs with Plumber

```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)
}
```

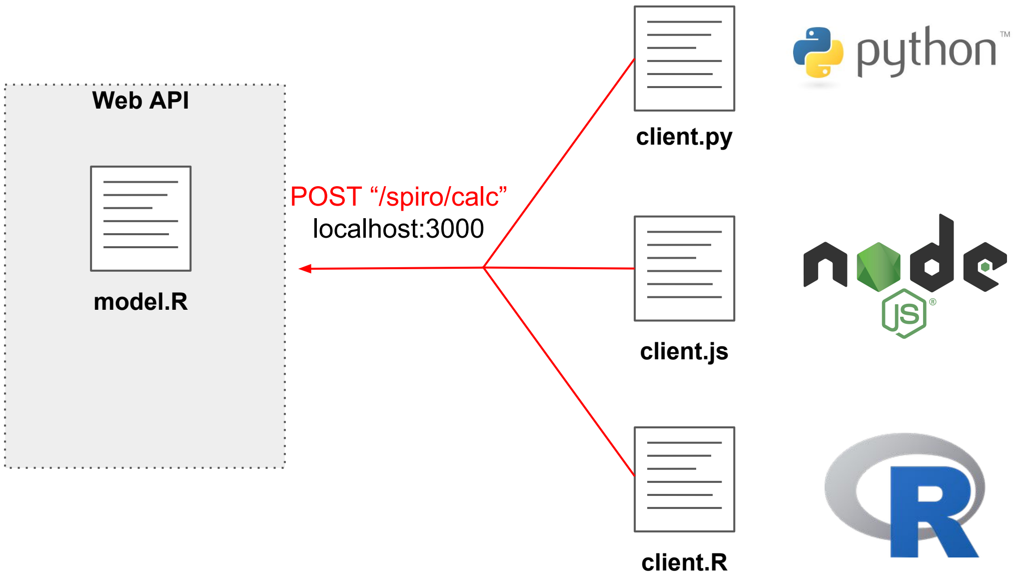
```r
#* Spirograph with custom n1, n2, n3
#* @post /spiro/calc
#* @param n3 Characteristics for the third wheel
#* @param n2 Characteristics for the second wheel
#* @param n1 Characteristics for the first wheel
#* @json
function(n1,n2,n3){
  return(spiro(as.numeric(n1),
               as.numeric(n2),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

Describe the Endpoint



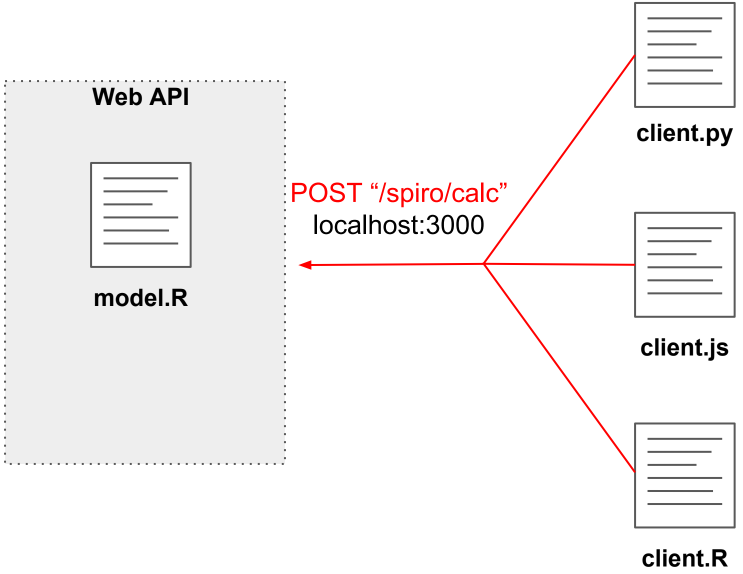
Building Web APIs with Plumber

```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

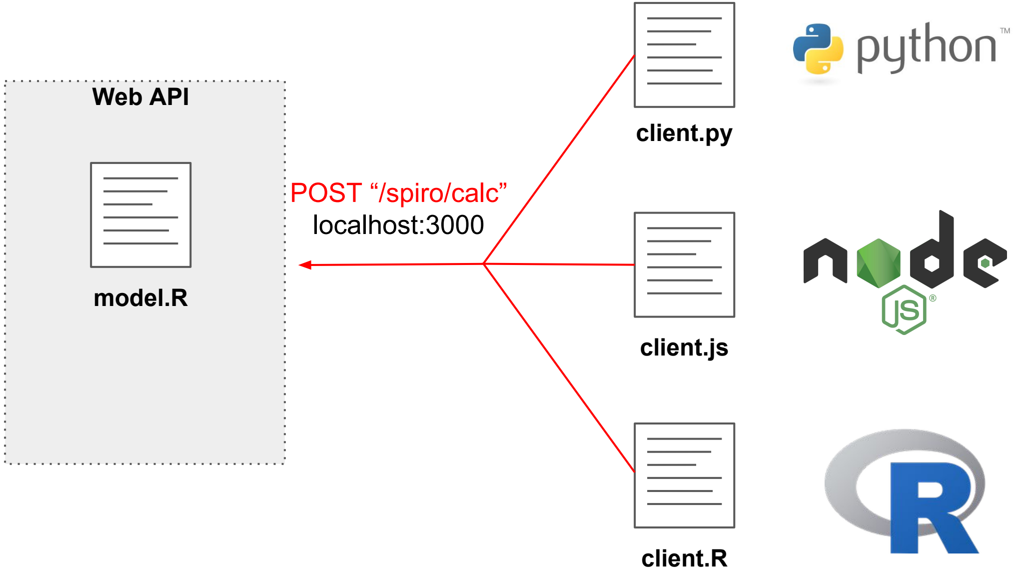
```r
spiro <- function(n1=13,n2=-7,n3=-3) {
  t <- seq(0,1,length.out=1000)
  z <- exp(1i*2*pi*n1*t) +
       exp(1i*2*pi*n2*t) +
       exp(1i*2*pi*n3*t)
  result <- tibble(x=Re(z),y=Im(z))
  return(result)
}

#* 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)))
}
```



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

Building Web APIs with Plumber

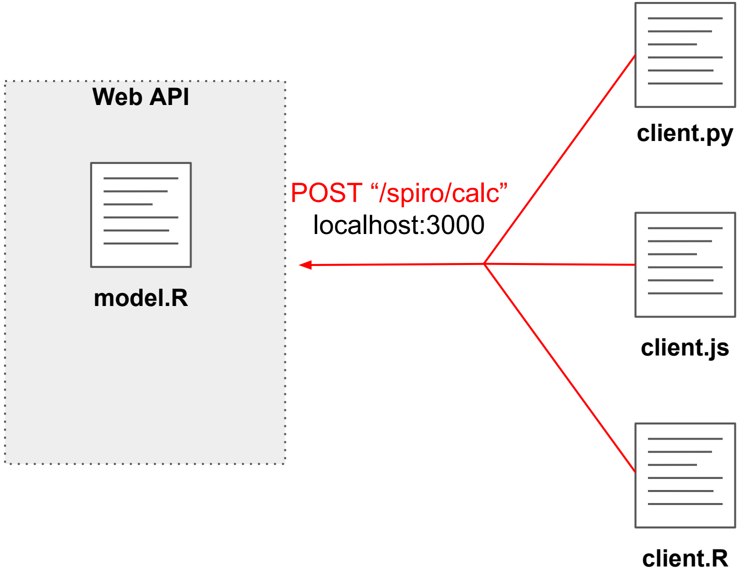
```r
spiro <- function(n1=13,n2=-7,n3=-3) {
  t <- seq(0,1,length.out=1000)
  z <- exp(1i*2*pi*n1*t) +
       exp(1i*2*pi*n2*t) +
       exp(1i*2*pi*n3*t)
  result <- tibble(x=Re(z),y=Im(z))
  return(result)
}

#* 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
#* @json
function(n1,n2,n3){
  return(spiro(as.numeric(n1),
               as.numeric(n2),
               as.numeric(n3)))
}
```



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

Building Web APIs with Plumber

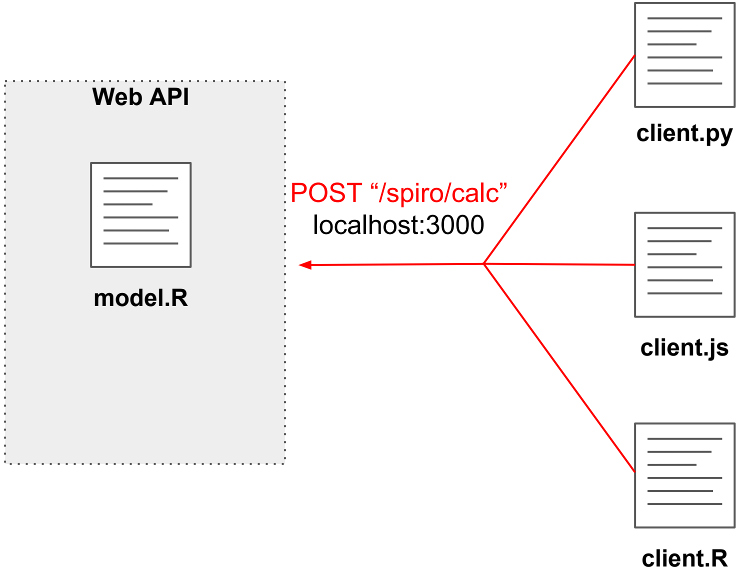
```r
spiro <- function(n1=13,n2=-7,n3=-3) {
  t <- seq(0,1,length.out=1000)
  z <- exp(1i*2*pi*n1*t) +
       exp(1i*2*pi*n2*t) +
       exp(1i*2*pi*n3*t)
  result <- tibble(x=Re(z),y=Im(z))
  return(result)
}

#* 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
#* @json
function(n1,n2,n3){
  return(spiro(as.numeric(n1),
               as.numeric(n2),
               as.numeric(n3)))
}
```



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

Building Web APIs with Plumber

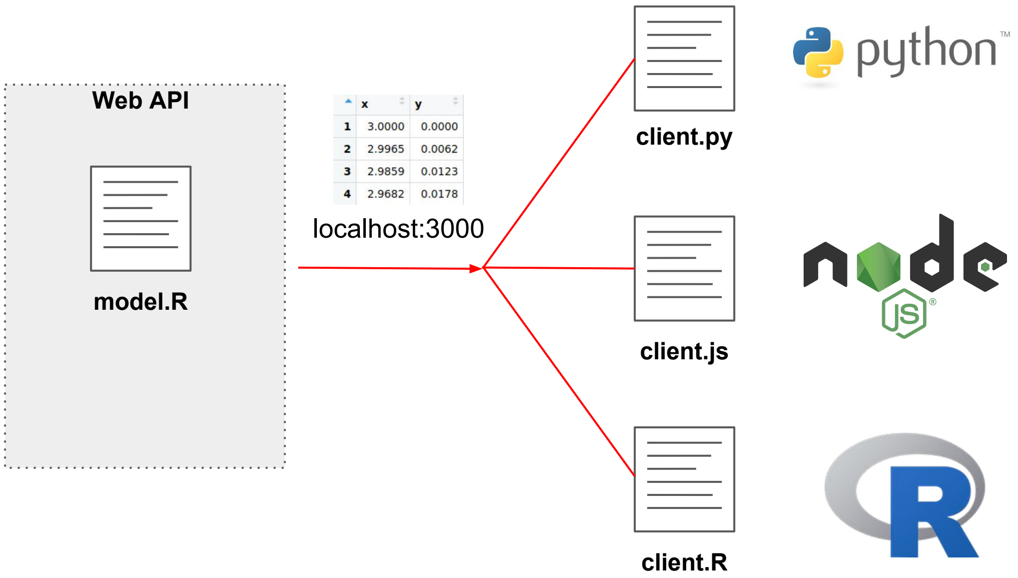
```r
spiro <- function(n1=13,n2=-7,n3=-3) {
  t <- seq(0,1,length.out=1000)
  z <- exp(1i*2*pi*n1*t) +
       exp(1i*2*pi*n2*t) +
       exp(1i*2*pi*n3*t)
  result <- tibble(x=Re(z),y=Im(z))
  return(result)
}

#* 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
#* @json
function(n1,n2,n3){
  return(spiro(as.numeric(n1),
               as.numeric(n2),
               as.numeric(n3)))
}
```



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

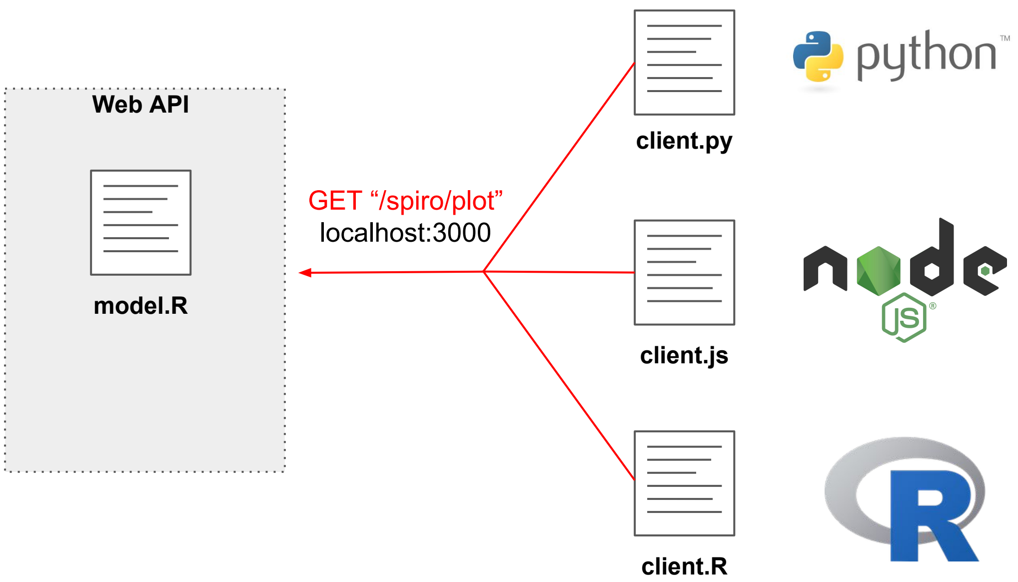
Building Web APIs with Plumber

```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
#* @json
function(n1,n2,n3){
  return(spiro(as.numeric(n1),
               as.numeric(n2),
               as.numeric(n3)))
}
```



Building Web APIs with Plumber

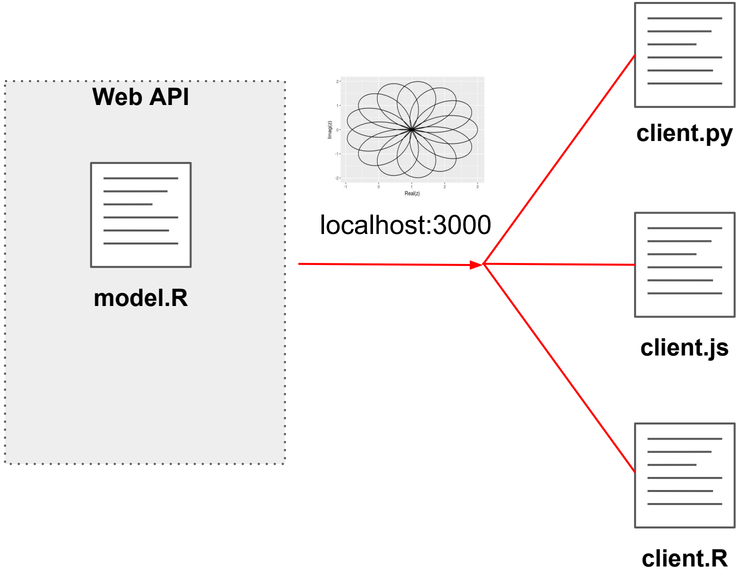
```r
spiro <- function(n1=13,n2=-7,n3=-3) {
  t <- seq(0,1,length.out=1000)
  z <- exp(1i*2*pi*n1*t) +
       exp(1i*2*pi*n2*t) +
       exp(1i*2*pi*n3*t)
  result <- tibble(x=Re(z),y=Im(z))
  return(result)
}

#* 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)")
}
```



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

Building Web APIs with Plumber

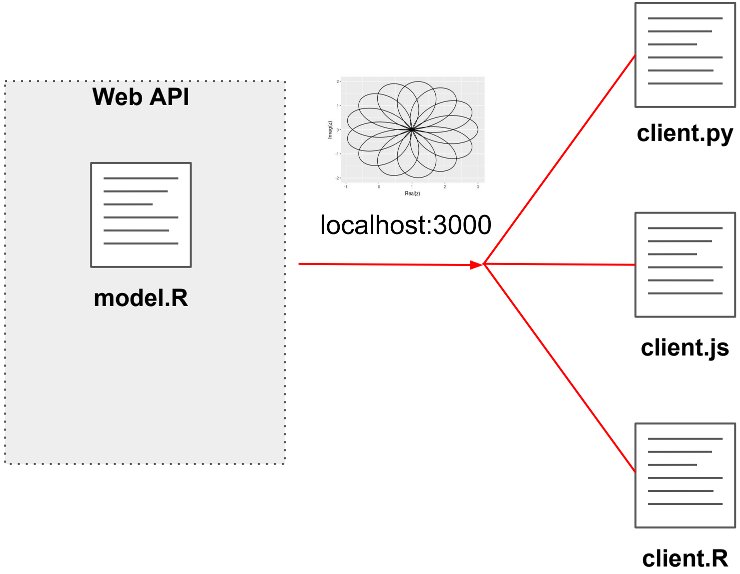
```r
spiro <- function(n1=13,n2=-7,n3=-3) {
  t <- seq(0,1,length.out=1000)
  z <- exp(1i*2*pi*n1*t) +
       exp(1i*2*pi*n2*t) +
       exp(1i*2*pi*n3*t)
  result <- tibble(x=Re(z),y=Im(z))
  return(result)
}

#* 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
#* @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)")
}
```



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

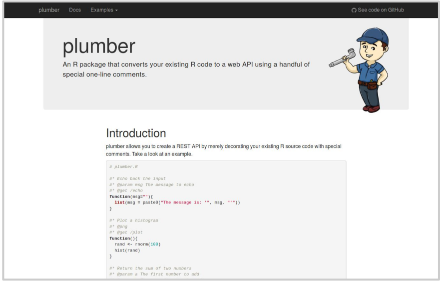
Building Web APIs with Plumber

```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
#* @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)")
}
```



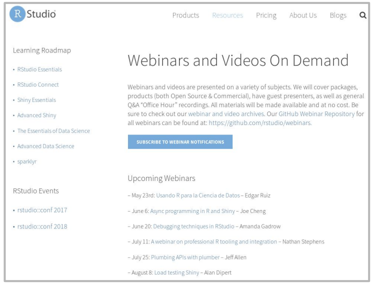
Try it: spiro_clients.Rmd from github.com:dskard/spiro-plumber
https://beta.rstudioconnect.com/connect/#/apps/3533

Learn More About Plumber



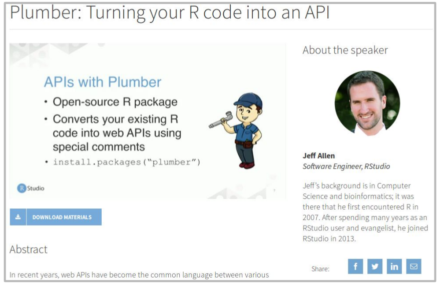
https://www.rplumber.io

Plumber Documentation



https://rstudio.com/resources/webinars

Upcoming Webinar about Plumber APIs: July 25, 2018



https://www.rstudio.com/resources/videos/plumber-turning-your-r-code-into-an-api/

Jeff Allen's rstudio::conf(2018) Talk