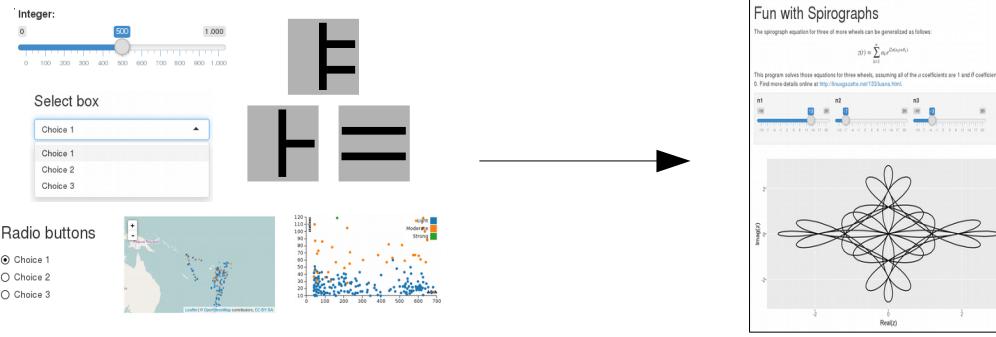


Using Rmarkdown to show Research



Derrick Kearney

*HUBzero® Platform for Scientific Collaboration
Purdue University*

This work licensed under
Creative Commons



See license online:
by-nc-sa/3.0

Some people write programs like this...

```
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)  
}
```

Some people write programs like this...

```
# Comments Suck
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) \
}

# Learn to read code!
result <- data.frame(x=Re(z),y=Im(z))
return (result)
}
```

Other people write programs like this...

```
# Function to produce a spirograph
# Accepts 3 parameters describing the size of our wheels

spiro <- function(n1,n2,n3) {
  t <- seq(0,1,length.out=1000)

  # Spirographs can be approximated with this formula
  z <- exp(1i*2*pi*n1*t) \
    + exp(1i*2*pi*n2*t) \
    + exp(1i*2*pi*n3*t) \

  # store the results for the user
  result <- data.frame(x=Re(z),y=Im(z))
  return (result)
}
```

And a few people write programs like this...

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 **\$\theta\$** coefficients are **0**. Find more details online at [<http://linuxgazette.net/133/luana.html>](http://linuxgazette.net/133/luana.html).

```
```{r echo=TRUE}
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)
}
ggplot(data=spiro(13,-7,-3),aes(x=x,y=y)) + geom_path()
```

```

When they share their programs, it looks like this:

Fun with Spirographs

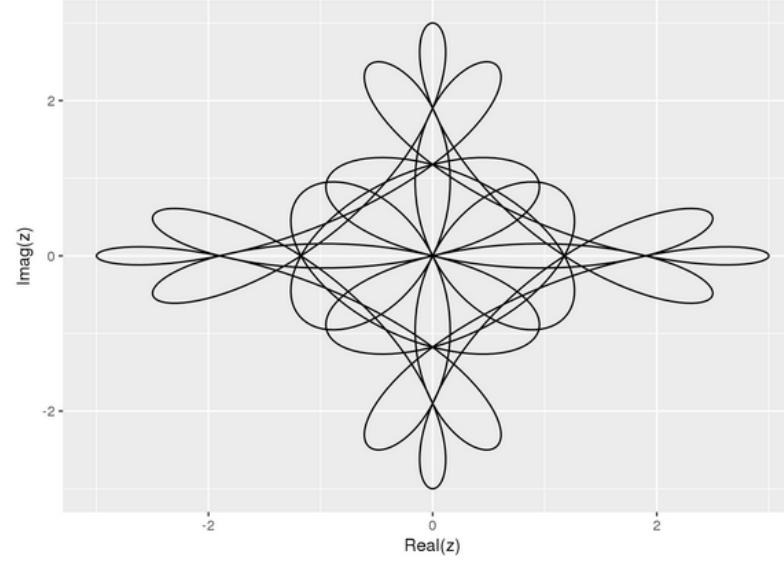
The spirograph equation for three or more wheels can be generalized as follows:

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

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)
```



Literate Programming

Donald Knuth. "Literate Programming (1984)" in Literate Programming. CSLI, 1992, pg. 99.

I believe that the time is ripe for significantly better documentation of programs, and that we can best achieve this by considering programs to be works of literature. Hence, my title: "Literate Programming."

Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.

The practitioner of literate programming can be regarded as an essayist, whose main concern is with exposition and excellence of style. Such an author, with thesaurus in hand, chooses the names of variables carefully and explains what each variable means. He or she strives for a program that is comprehensible because its concepts have been introduced in an order that is best for human understanding, using a mixture of formal and informal methods that reinforce each other.

literateprogramming.com (retrieved Apr 29, 2017)

What are Interactive Notebooks?

Executable Code
& Calculations

```
In [3]:
```

```
def plot_spiral(n1, n2, n3):
    t = linspace(0, 1, 1000)
    z = exp(1j*2*pi*n1*t) \
        + exp(1j*2*pi*n2*t) \
        + exp(1j*2*pi*n3*t)
    plt.plot(real(z), imag(z))
    plt.show()
```

```
In [3]:
```

```
def plot_spiral(n1, n2, n3):
    t = linspace(0, 1, 1000)
    z = exp(1j*2*pi*n1*t) \
        + exp(1j*2*pi*n2*t) \
        + exp(1j*2*pi*n3*t)
    plt.plot(real(z), imag(z))
    plt.show()
```



MDTraj <http://mdtraj.org/>
MDTraj is a python library that allows users to manipulate molecular dynamics simulation trajectories. Features include:

- Read MD trajectories in various formats: xtc, trr, dcd, binpos, netcdf, mdtraj, prmtop, and more.
- Efficient trajectory reading (~4x the speed of the original Theobald GROMOS).
- Extensive analysis functions including those that compute bonds, angles, dihedrals, hydrogen bonds, secondary structure, and NMR scalar coupling.
- Lightweight, Pythonic API.

Jupyter Notebook
calculations.ipynb

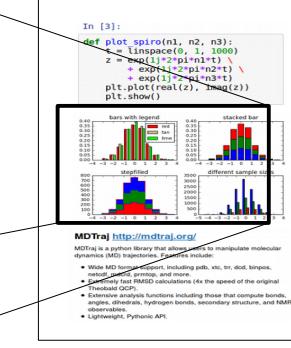
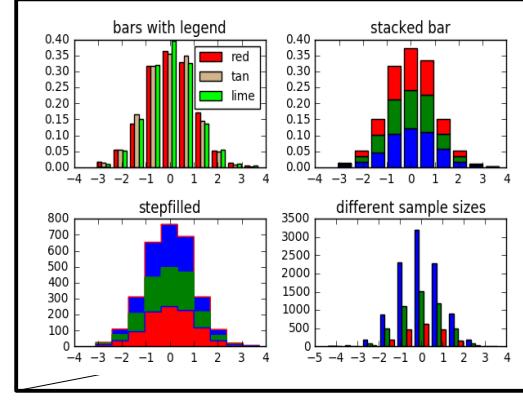
What are Interactive Notebooks?

Executable Code
& Calculations

Data Visualizations

In [3]:

```
def plot_spiro(n1, n2, n3):
    t = linspace(0, 1, 1000)
    z = exp(1j*2*pi*n1*t) \
        + exp(1j*2*pi*n2*t) \
        + exp(1j*2*pi*n3*t)
    plt.plot(real(z), imag(z))
    plt.show()
```



Jupyter Notebook
calculations.ipynb

What are Interactive Notebooks?

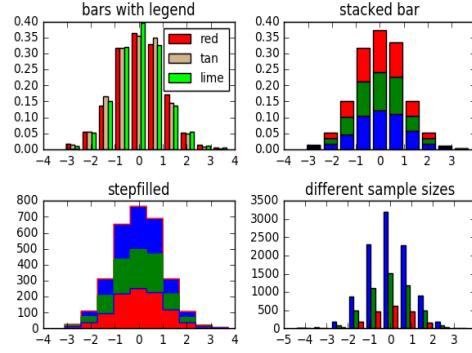
Executable Code
& Calculations

Data Visualizations

Narrative

In [3]:

```
def plot_spiro(n1, n2, n3):
    t = linspace(0, 1, 1000)
    z = exp(1j*2*pi*n1*t) \
        + exp(1j*2*pi*n2*t) \
        + exp(1j*2*pi*n3*t)
    plt.plot(real(z), imag(z))
    plt.show()
```



MDTraj <http://mdtraj.org/>

MDTraj is a python library that allows users to manipulate molecular dynamics (MD) trajectories. Features include:

- Wide MD format support, including pdb, xtc, trr, dcd, binpos, netcdf, mdcrd, prmtop, and more.
- Extremely fast RMSD calculations (4x the speed of the original Theobald QCP).
- Extensive analysis functions including those that compute bonds, angles, dihedrals, hydrogen bonds, secondary structure, and NMR observables.
- Lightweight, Pythonic API.

```
In [3]:
def plot_spiro(n1, n2, n3):
    t = linspace(0, 1, 1000)
    z = exp(1j*2*pi*n1*t) \
        + exp(1j*2*pi*n2*t) \
        + exp(1j*2*pi*n3*t)
    plt.plot(real(z), imag(z))
    plt.show()
```

MDTraj <http://mdtraj.org/>
 MDTraj is a python library that allows users to manipulate molecular dynamics (MD) trajectories. Features include:
 • Wide MD format support, including pdb, xtc, trr, dcd, binpos, netcdf, mdcrd, prmtop, and more.
 • Extremely fast RMSD calculations (4x the speed of the original Theobald QCP).
 • Extensive analysis functions including those that compute bonds, angles, dihedrals, hydrogen bonds, secondary structure, and NMR observables.
 • Lightweight, Pythonic API.

Jupyter Notebook
calculations.ipynb

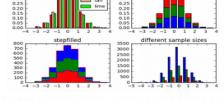
Sharing Notebooks



Sharing Made Easy

- Text based file format
- Outputs stored with document

```
In [3]:
def plot_spiral(n1, n2, n3):
    t = linspace(0, 1, 1000)
    z = exp(1j*2*pi*n1*t) \
        + exp(1j*2*pi*n2*t) \
        + exp(1j*2*pi*n3*t)
    plt.plot(real(z), imag(z))
    plt.show()



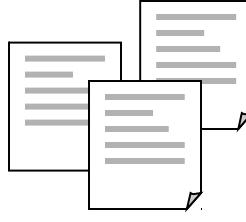
```

MDTraj <http://mdtraj.org/>

MDTraj is a python library that allows users to manipulate molecular dynamics simulation trajectories. Features include:

- Reading MD trajectories in various formats: trr, dcd, binpos, netcdf, mdtraj, prmtop, and more.
- Writing MD trajectories in various formats (4x the speed of the original Threaded OpenMM).
- Computing various properties and functions including those that compute bonds, angles, dihedrals, hydrogen bonds, secondary structure, and NMR relaxation times.
- Lightweight, Pythonic API.

Sharing Notebooks

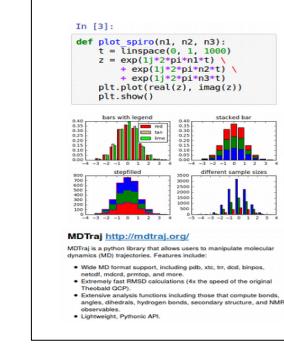


Sharing Made Easy

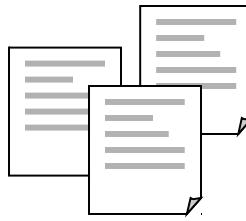
- Text based file format
- Outputs stored with document

Export Document to:

- PDF, Word, HTML,
- LaTeX, Slides,
- Markdown, Wiki, EPub



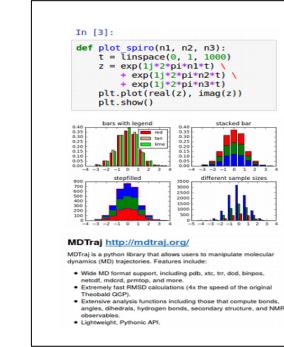
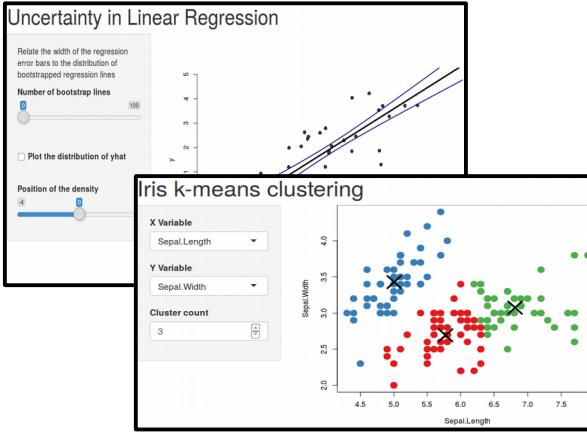
Sharing Notebooks



Sharing Made Easy

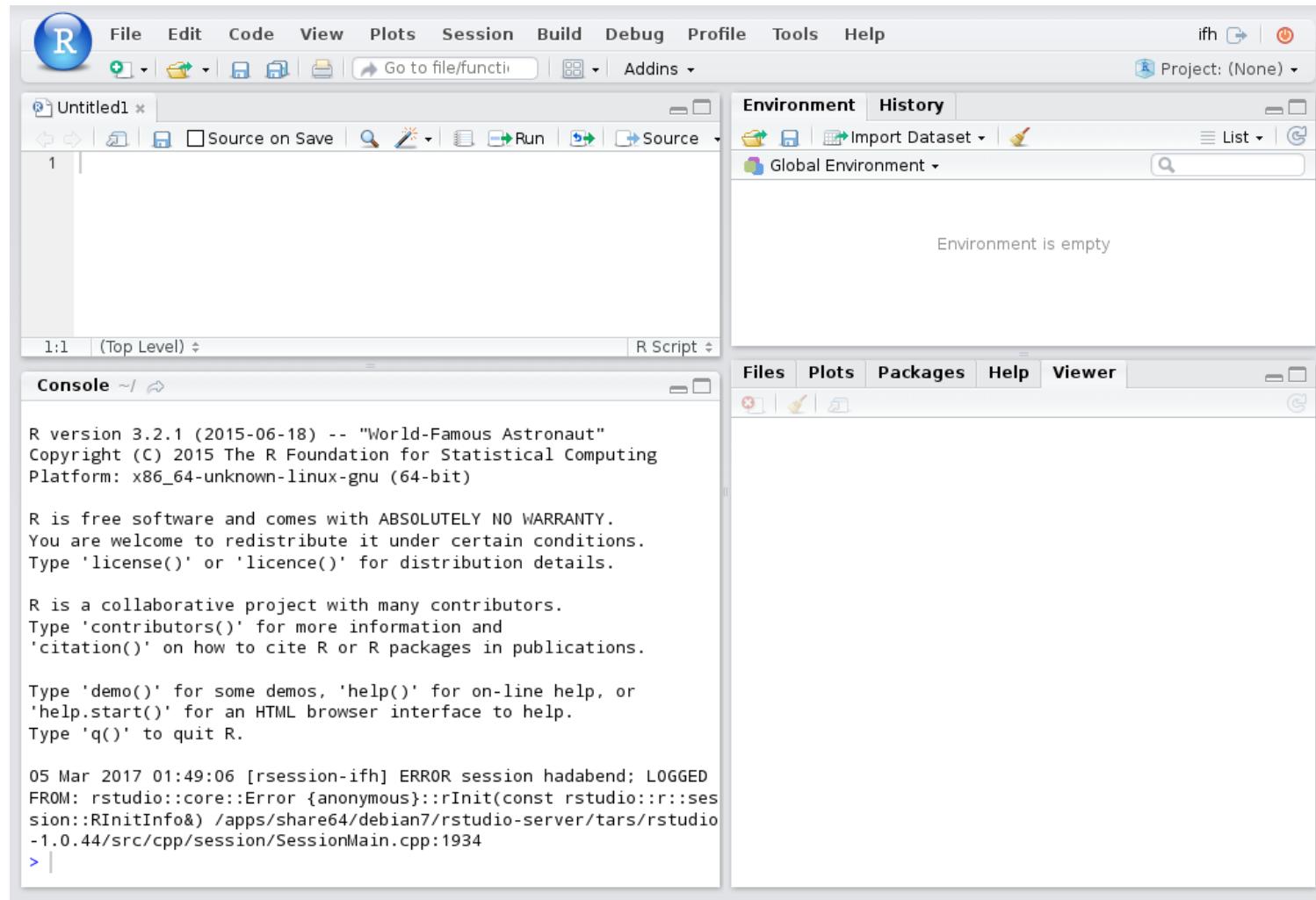
- Text based file format
- Outputs stored with document

Export Document to:
 PDF, Word, HTML,
 LaTeX, Slides,
 Markdown, Wiki, EPub

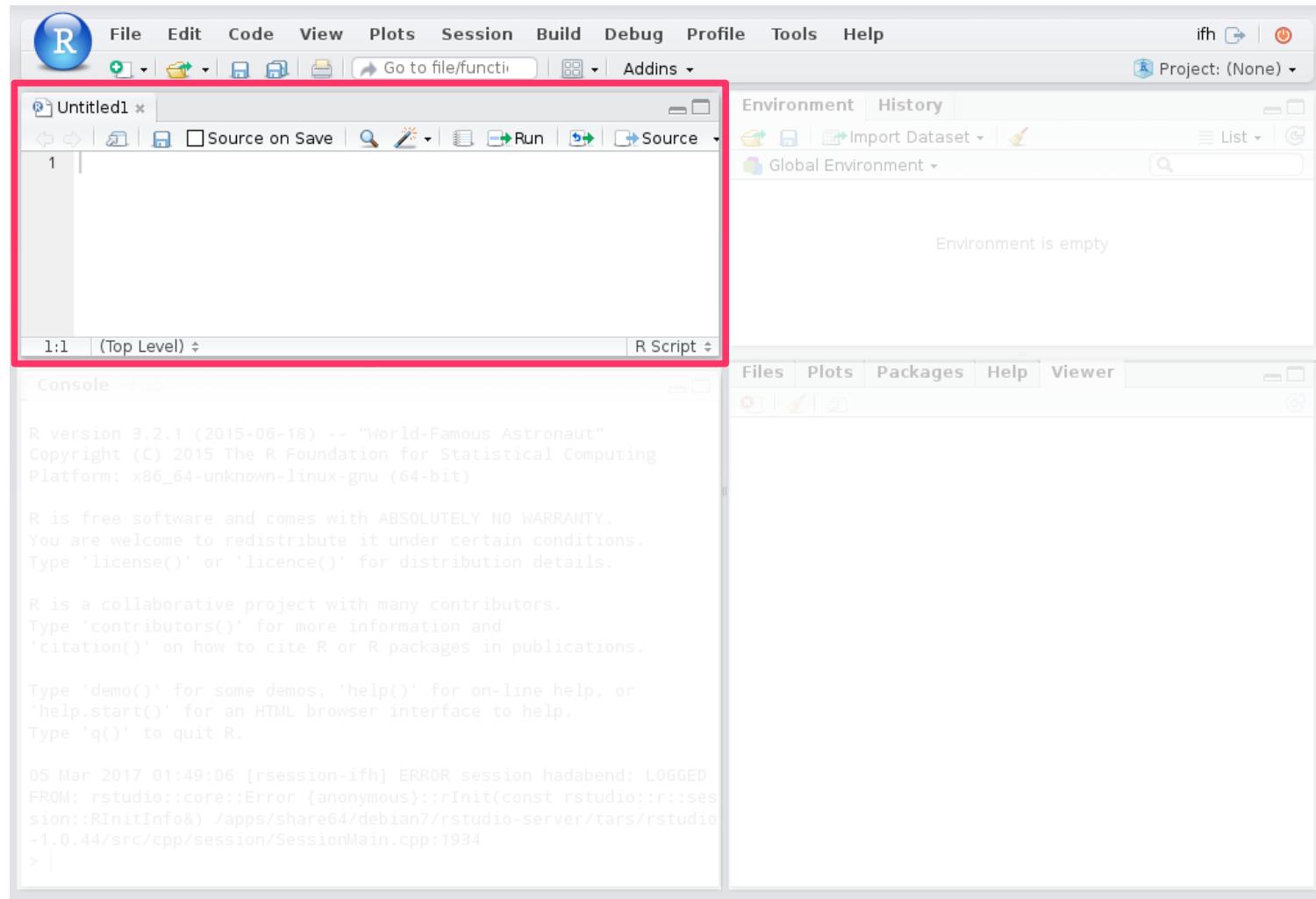


Embed Interactive
 Web Applications

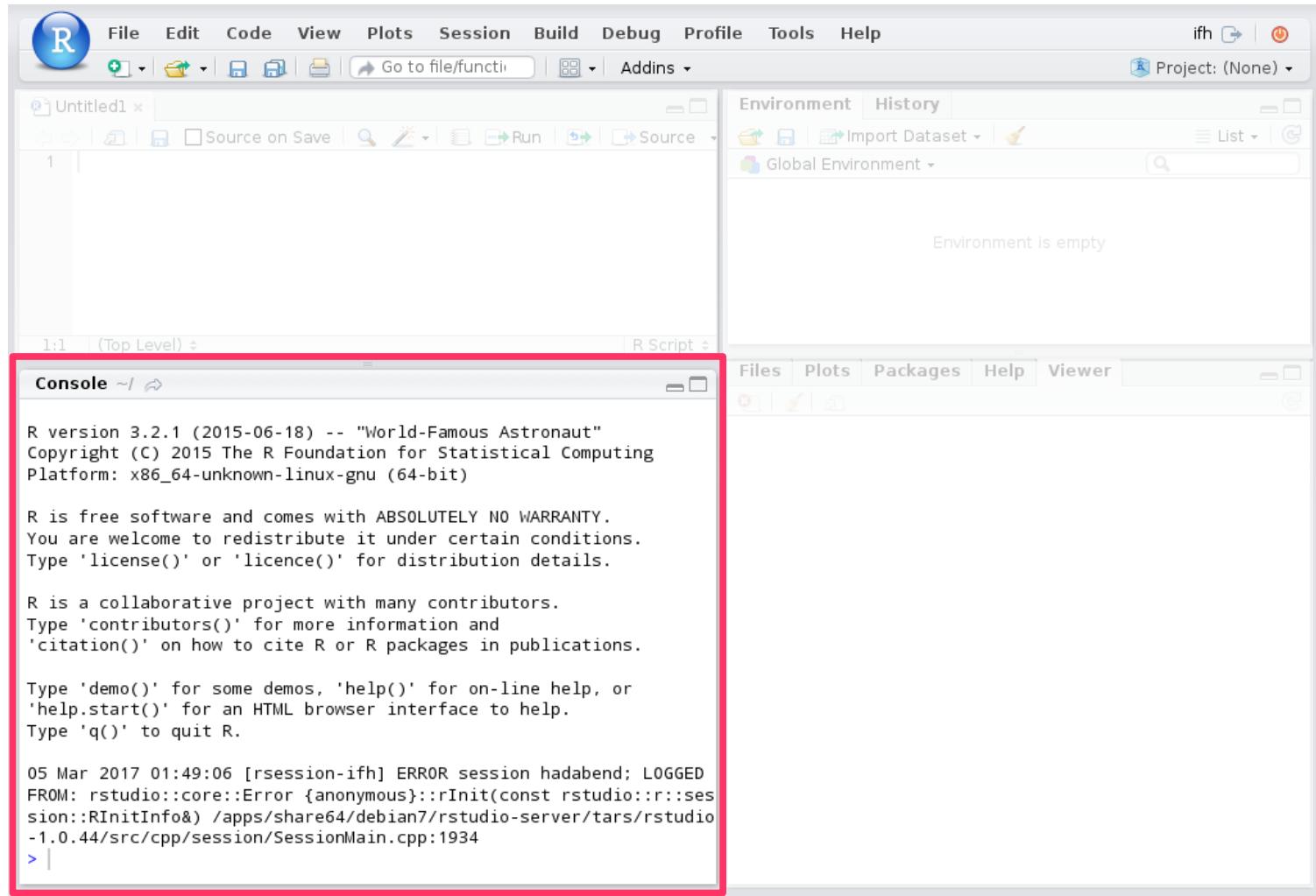
Launching RStudio IDE



RStudio Interface Tour – Source Pane



RStudio Interface Tour – Console Pane



The screenshot shows the RStudio interface with the following components:

- Top Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project Bar:** Project: (None).
- Environment Tab:** Shows "Environment" and "History" tabs. The Environment tab displays "Environment is empty".
- Console Tab:** Shows the R startup message and a log entry from March 5, 2017.
- Bottom Tabs:** Files, Plots, Packages, Help, Viewer.

Console Output:

```
R version 3.2.1 (2015-06-18) -- "World-Famous Astronaut"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-unknown-linux-gnu (64-bit)

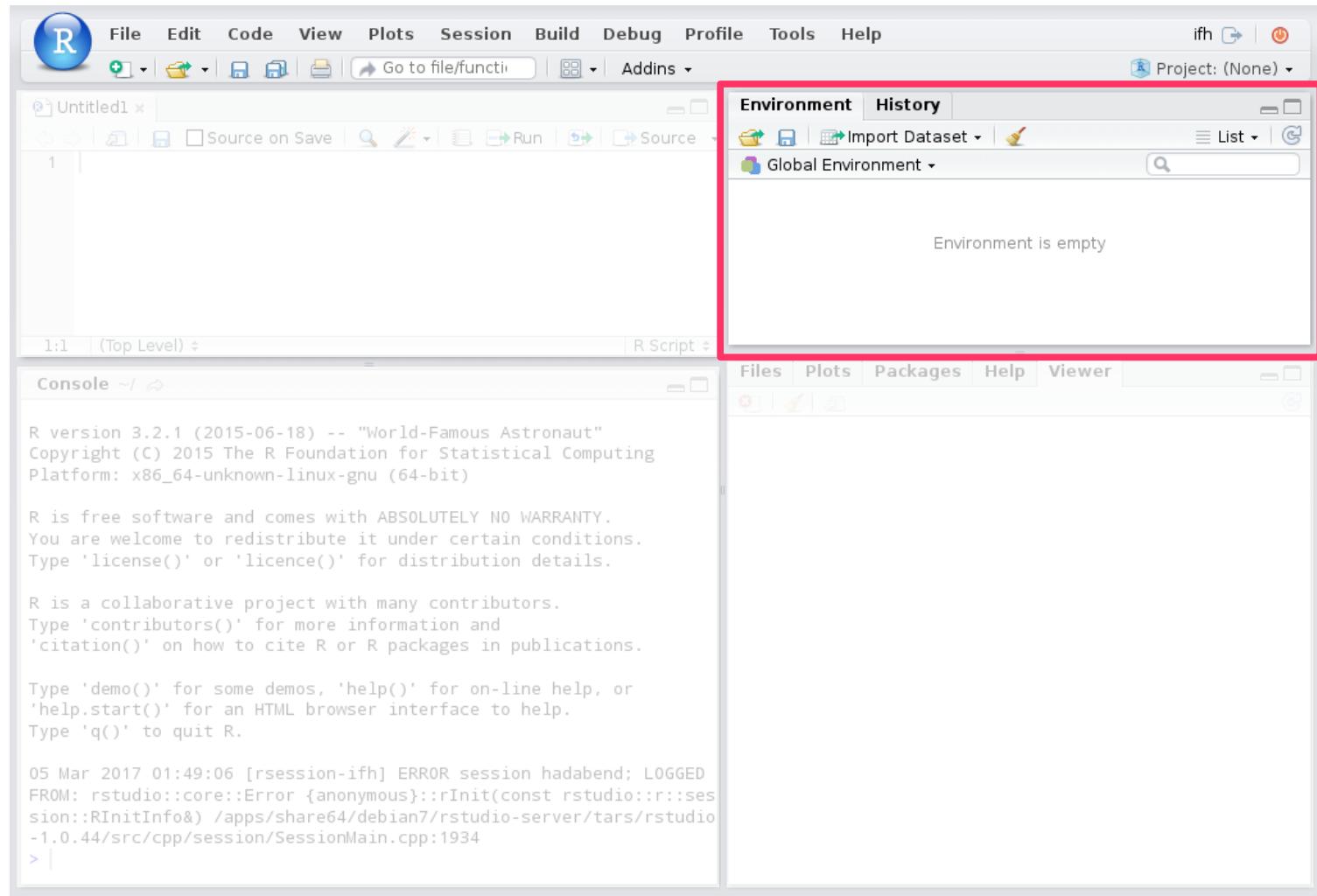
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

05 Mar 2017 01:49:06 [rsession-ifh] ERROR session hadabend; LOGGED
FROM: rstudio::core::Error {anonymous}::rInit(const rstudio::r::ses
sion::RInitInfo& ) /apps/share64/debian7/rstudio-server/tars/rstudio
-1.0.44/src/cpp/session/SessionMain.cpp:1934
> |
```

RStudio Interface Tour – Environment Pane



The screenshot shows the RStudio interface with the Environment pane highlighted by a red box. The Environment pane title bar says "Environment | History". It contains a toolbar with icons for Import Dataset, Global Environment, and a search bar. Below the toolbar, a message says "Environment is empty". The rest of the interface includes the top menu bar, a file browser, a code editor, a console window displaying R startup messages, and a bottom navigation bar.

Environment | History

Import Dataset | Global Environment |

Environment is empty

Console

```
R version 3.2.1 (2015-06-18) -- "World-Famous Astronaut"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-unknown-linux-gnu (64-bit)

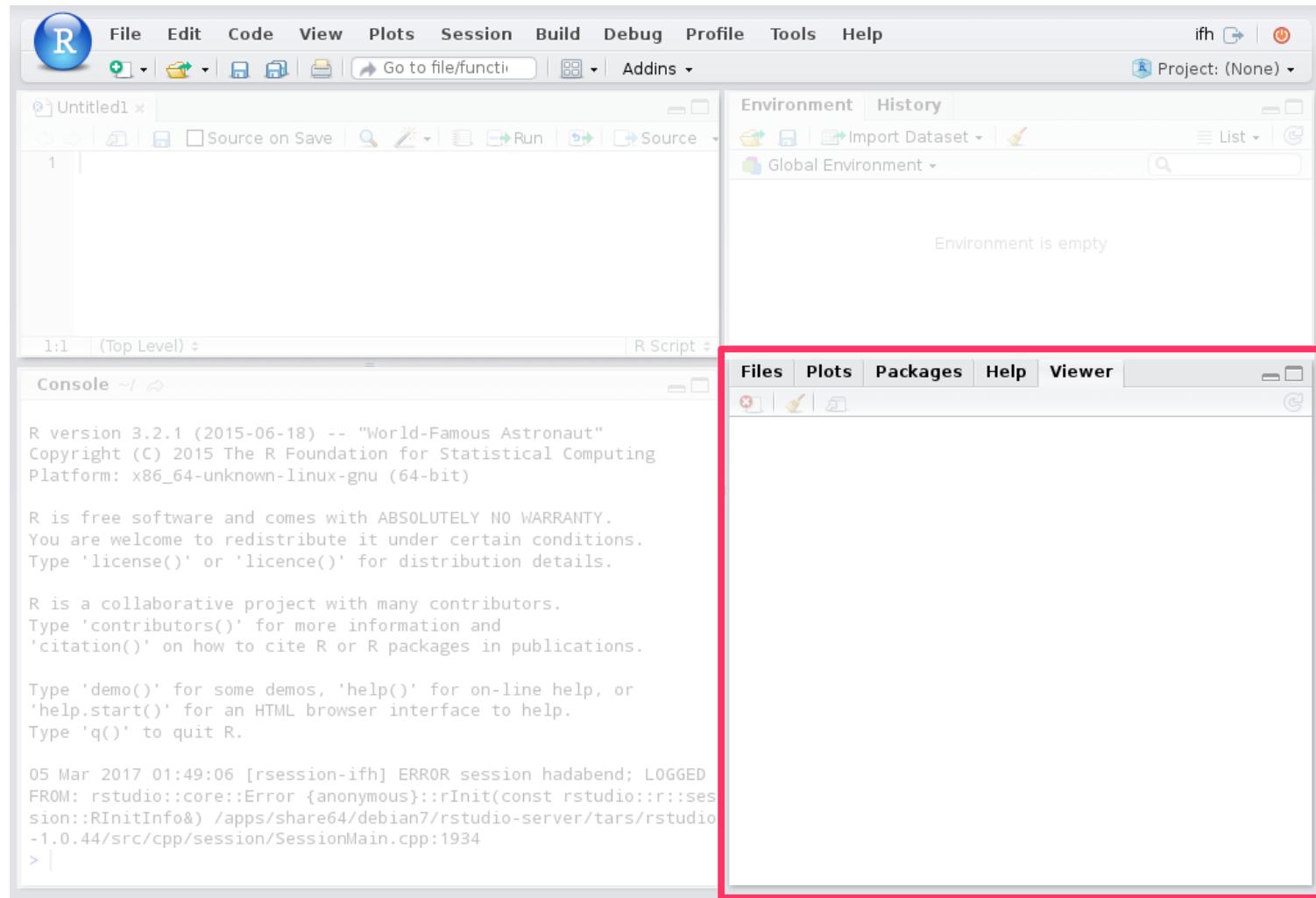
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

05 Mar 2017 01:49:06 [rsession-ifh] ERROR session hadabend; LOGGED
FROM: rstudio:::core:::Error {anonymous}::rInit(const rstudio:::session:::RInitInfo&)
/apps/share64/debian7/rstudio-server/tars/rstudio
-1.0.44/src/cpp/session/SessionMain.cpp:1934
> |
```

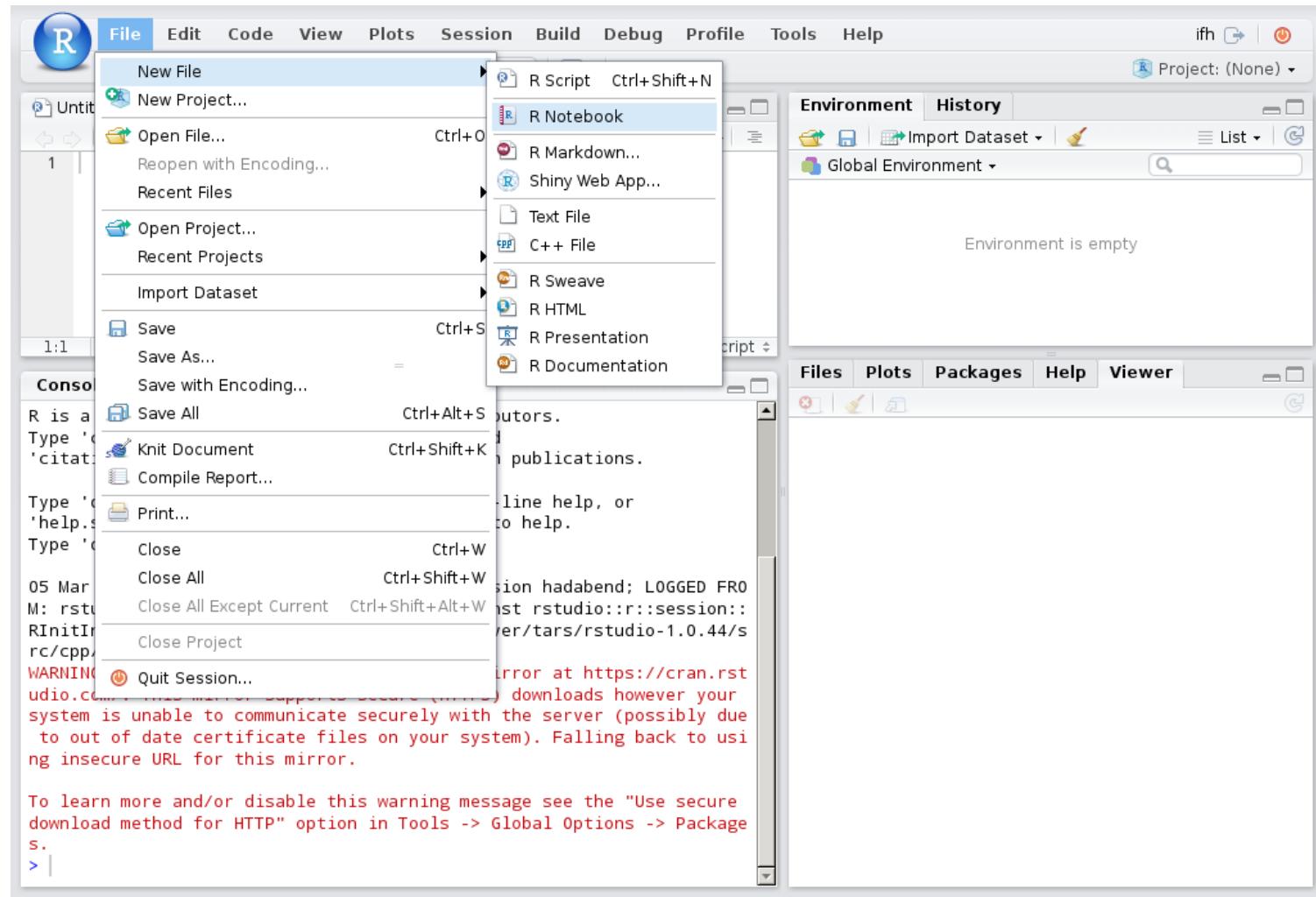
RStudio Interface Tour – Viewer Pane



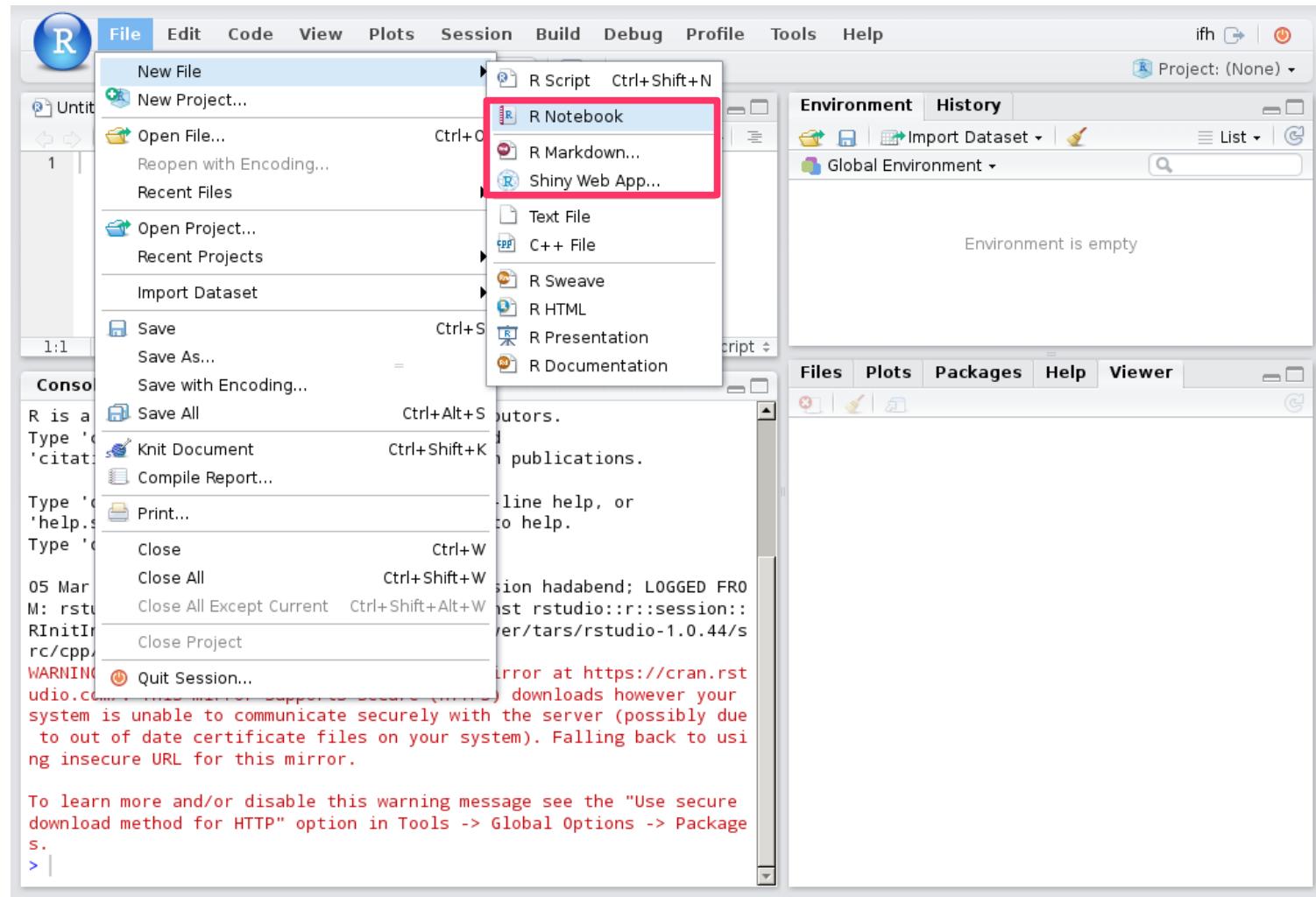
The screenshot shows the RStudio interface with the following components:

- Top Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project Bar:** Project: (None).
- Environment Tab:** Shows "Environment" and "History" tabs. The Environment tab displays "Environment is empty".
- Console Tab:** Displays the R startup message and session logs.
- Viewer Tab:** The tab bar includes Files, Plots, Packages, Help, and **Viewer**, which is highlighted with a red border.
- Viewer Area:** The main pane below the tab bar is currently empty.

Writing R Code



Writing R Code



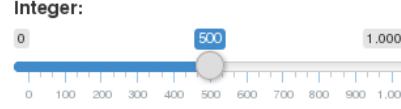
Building Applications and Interactive Reports



RStudio and Shiny are trademarks of RStudio, Inc.



Building Applications and Interactive Reports



Select box

Choice 1

▼

Choice 1

Choice 2

Choice 3

Radio buttons

- Choice 1
- Choice 2
- Choice 3

Building Applications and Interactive Reports



Integer:

0 100 200 300 400 500 600 700 800 900 1.000

Select box

Choice 1

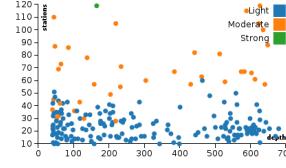
Choice 1

Choice 2

Choice 3

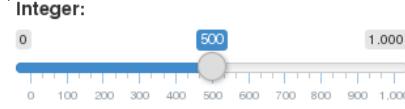
Radio buttons

- Choice 1
- Choice 2
- Choice 3



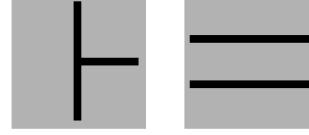
RStudio and Shiny are trademarks of RStudio, Inc.

Building Applications and Interactive Reports



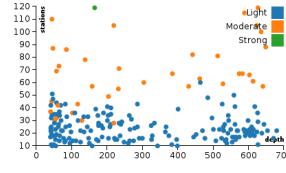
Select box

Choice 1
 Choice 1
 Choice 2
 Choice 3



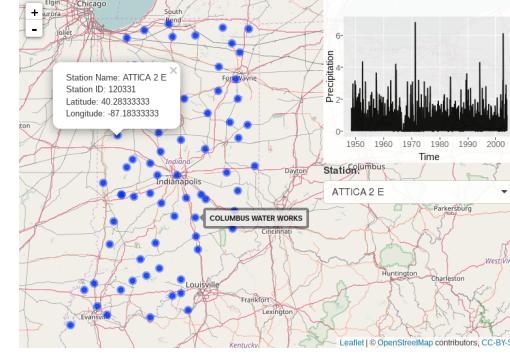
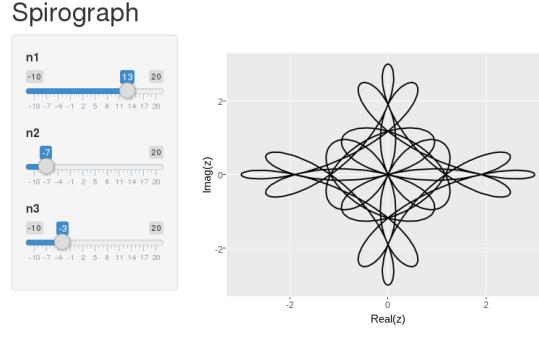
Radio buttons

- Choice 1
- Choice 2
- Choice 3



RStudio and Shiny are trademarks of RStudio, Inc.

Shiny Example 1 - Spirograph



Spirograph Example

<https://github.com/codedsk/hubzero-tool-spiro-shiny-1>

Indiana Precip Example

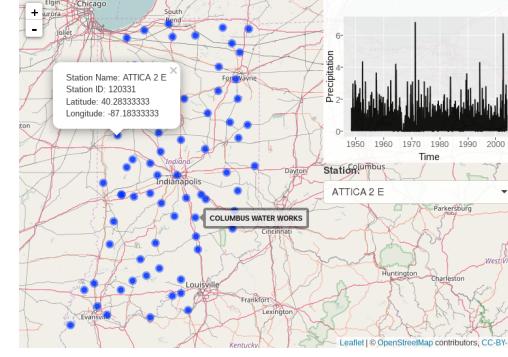
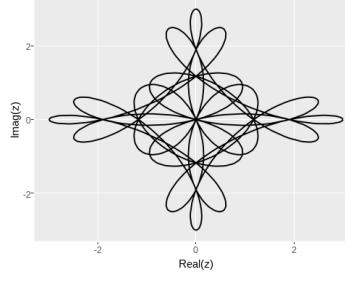
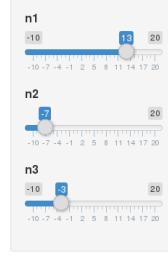
<https://github.com/codedsk/rplay/mapping/indiana-precip-shiny>

RStudio and Shiny are trademarks of RStudio, Inc.

Shiny Example 2 – Indiana Precipitation



Spirograph



Spirograph Example

<https://github.com/codedsk/hubzero-tool-spiro-shiny-1>

Indiana Precip Example

<https://github.com/codedsk/rplay/mapping/iniana-precip-shiny>

RStudio and Shiny are trademarks of RStudio, Inc.

Building Applications and Interactive Reports

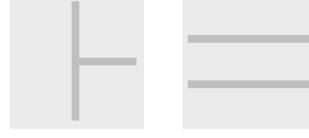


Integer:



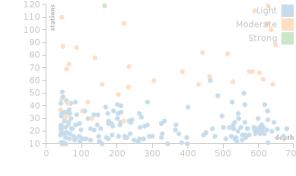
Select box

- Choice 1
- Choice 1
- Choice 2
- Choice 3



Radio buttons

- Choice 1
- Choice 2
- Choice 3



```

9- ``{r setup, include=FALSE}
10 knitr::opts_chunk$set(echo = TRUE)
11
12
13
14 The spirograph equation for three of more wheels can be generalized
as follows:
15
16 $$z(t) = \sum_{k=1}^n a_k e^{i2\pi(n_k t + \theta_k)}$$
17
18 This program solves those equations for three wheels, assuming all
of the $a_k$ coefficients are 1 and $\theta_k$ coefficients are 0. Find
more details online at <http://linuxgazette.net/133/luana.html>.
19
20
21- ``{echo=TRUE}
22 spiro <- function(n1,n2,n3) {
23   t <- seq(0,1,length.out=1000)
24   z <- exp(1i*2*pi*n1*t) + exp(1i*2*pi*n2*t) + exp(1i*2*pi*n3*t)
25   result <- data.frame(x=Re(z),y=Im(z))
26   return (result)
27 }
28
29 result <- spiro(13,-7,-3)

```

RStudio and Shiny are trademarks of RStudio, Inc.

Building Applications and Interactive Reports



Integer:



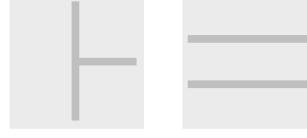
Select box

Choice 1

Choice 1

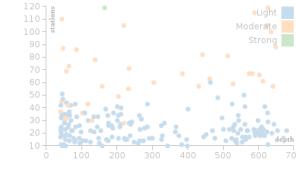
Choice 2

Choice 3



Radio buttons

- Choice 1
- Choice 2
- Choice 3



```
9- `` {r setup, include=FALSE}
10 knitr::opts_chunk$set(echo = TRUE)
11
12
13
```

14 The spirograph equation
as follows:

```
15 $\$z(t) = \sum_{k=1}^n a_k e^{j2\pi(f_k t + \theta_k)}
```

```
17
18 This program solves tho
of the \$a\$ coefficients
more details online at
```

```
19
20
21- `` {r echo=TRUE}
22 spiro <- function(n1,n2)
23 t <- seq(0,1,length.out=1000)
24 z <- exp((1i*2*pi*n1*t) + exp(1i*2*pi*n2*t))
25 result <- data.frame(x=Re(z),y=Im(z))
26 return (result)
27 }
28
29 result <- spiro(13,-7,-3)
```

Fun with Spirographs

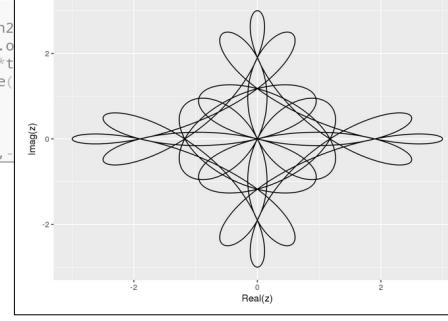
The spirograph equation for three or more wheels can be generalized as follows:

$$z(t) = \sum_{k=1}^n a_k e^{j2\pi(f_k t + \theta_k)}$$

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://linuzgazette.net/133/usana.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)
```



RStudio and Shiny are trademarks of RStudio, Inc.

Rmd Example 1 - Markup



```

9  # (r, setup, include=FALSE)
10 knitr::opts_chunk$set(echo = TRUE)
11 ...
12 
13 
14 # The spirograph equation for three of more wheels can be generalized as follows:
15 
16 ##z(t) = \sum_{k=1}^n a_k e^{ik\pi(n_k t)}
17 
18 This program solves those equations for three wheels, assuming all of the a coefficients are 1 and # coefficients are 1. Find more details online at http://linuxgazette.net/123spiro.html.
19 
20 
21 ## (r echo=TRUE)
22 spiro <- function(n1,n2,n3) {
23   t <- seq(0,1,length.out=1000)
24   z <- exp(1i*2*pi*n1*t) + exp(1i*2*pi*n3*t)
25   result <- data.frame(x=Re(z),y=Im(z))
26   return(result)
27 }
28 
29 result <- spiro(13,-7,-3)

```

Fun with Spirographs

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

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

This program solves those equations for three wheels, assuming all of the a coefficients are 1 and # coefficients are 1. Find more details online at <http://linuxgazette.net/123spiro.html>.

```

15 Examples from the [RStudio leaflet](https://rstudio.github.io/leaflet/).
16 
17 Introduction
18 
19 Build a basic map, using tiles from OpenStreetMap and adding a
20 marker for Purdue University.
21 
22 ## (r warning=FALSE, message=FALSE)
23 library(leaflet)
24 pu_lng = -86.9212
25 pu_lat = 40.4237
26 
27 m <- leaflet() %>%
28   addTiles() %>%
29   addMarkers lng=pu_lng, lat=pu_lat, popup="Purdue
30 
31 m
32

```

Fun with Spirographs

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

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

This program solves those equations for three wheels, assuming all of the a coefficients are 1 and # coefficients are 1. Find more details online at <http://linuxgazette.net/123spiro.html>.

Markup Example

Notebook w/ HTML Widgets

RMarkdown w/ Shiny Widgets

<https://github.com/codedsk/rplay>

RStudio and Shiny are trademarks of RStudio, Inc.

Rmd Example 2 – Notebook w/ HTML Widgets



```

9  # (r, setup, include=FALSE)
10 knitr::opts_chunk$set(echo = TRUE)
11 ...
12 
13
14 The spirograph equation for three or more wheels can be generalized as follows:
15
16 
$$z(t) = \sum_{k=1}^n a_k e^{i(2\pi n_k t)}$$

17 This program solves those equations for three wheels, assuming all of the  $a$  coefficients are 1 and  $\theta$  coefficients are 1. Find more details online at http://linuxgazette.net/123spiro.html.
18
19
20
21 ... (r echo=TRUE)
22 spiro <- function(n1,n2,n3) {
23   t <- seq(0,1,length.out=1000)
24   z <- exp(i*2*pi*n1*t) + exp(i*2*pi*n2*t) + exp(i*2*pi*n3*t)
25   result <- data.frame(x=Re(z),y=Im(z))
26   return(result)
27 }
28
29 result <- spiro(13,-7,-3)

```

Fun with Spirographs

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

$$z(t) = \sum_{k=1}^n a_k e^{i(2\pi n_k t)}$$

This program solves those equations for three wheels, assuming all of the a coefficients are 1 and θ coefficients are 1. Find more details online at <http://linuxgazette.net/123spiro.html>.

```

15 Examples from the RStudio leaflet
16 tutorial(https://rstudio.github.io/leaflet/)
17 Introduction
18 Build a basic map, using tiles from OpenStreetMap and adding a
19 marker for Purdue University.
20
21 ... (r warning=FALSE, message=FALSE)
22 library(leaflet)
23
24 spiro <- function(n1,n2,n3) {
25   t <- seq(0,1,length.out=1000)
26   z <- exp(i*2*pi*n1*t) + exp(i*2*pi*n2*t) + exp(i*2*pi*n3*t)
27   result <- data.frame(x=Re(z),y=Im(z))
28   return(result)
29 }
30
31
32

```

Fun with Spirographs

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

$$z(t) = \sum_{k=1}^n a_k e^{i(2\pi n_k t)}$$

This program solves those equations for three wheels, assuming all of the a coefficients are 1 and θ coefficients are 1. Find more details online at <http://linuxgazette.net/123spiro.html>.

Markup Example

Notebook w/ HTML Widgets

```

15 Examples from the RStudio leaflet
16 tutorial(https://rstudio.github.io/leaflet/)
17 Introduction
18 Build a basic map, using tiles from OpenStreetMap and adding a
19 marker for Purdue University.
20
21 ... (r warning=FALSE, message=FALSE)
22 library(leaflet)
23
24 spiro <- function(n1,n2,n3) {
25   t <- seq(0,1,length.out=1000)
26   z <- exp(i*2*pi*n1*t) + exp(i*2*pi*n2*t) + exp(i*2*pi*n3*t)
27   result <- data.frame(x=Re(z),y=Im(z))
28   return(result)
29 }
30
31
32

```

Fun with Spirographs

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

$$z(t) = \sum_{k=1}^n a_k e^{i(2\pi n_k t)}$$

This program solves those equations for three wheels, assuming all of the a coefficients are 1 and θ coefficients are 1. Find more details online at <http://linuxgazette.net/123spiro.html>.

RMarkdown w/ Shiny Widgets

<https://github.com/codedsk/rplay>

RStudio and Shiny are trademarks of RStudio, Inc.

Rmd Example 3 – Rmd w/ Shiny Widgets



9 `(r setup, include=FALSE)`
 10 `knitr::opts_chunk$set(echo = TRUE)`
 11 `---`
 12
 13
 14 The spirograph equation for three or more wheels can be generalized as follows:
 15

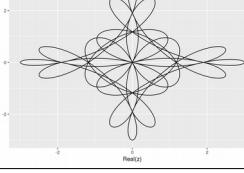
$$z(t) = \sum_{k=1}^n a_k e^{i(2\pi n_k t)}$$

 16 This program solves those equations for three wheels, assuming all of the a coefficients are 1 and #
 coefficients are 1 and #
 17 This program solves those equations for three wheels, assuming all of the a coefficients are 1 and #
 coefficients are 1 and #
 18 The `$s3` coefficients are 1 and #
 19 More details online at <http://linuxgazette.net>
 20
 21 `## (r echo=TRUE)`
 22 `spiro <- function(n1,n2,n3) {`
 23 `t <- seq(0,1,length.out=1000)`
 24 `z <- exp(i*2*pi*n1*t) + exp(i*2*pi*n2*t) + exp(i*2*pi*n3*t)`
 25 `result <- data.frame(x=Re(z),y=Im(z))`
 26 `return(result)`
 27 `}`
 28
 29 `result <- spiro(13,-7,-3)`

Fun with Spirographs
 The spirograph equation for three or more wheels can be generalized as follows:

$$z(t) = \sum_{k=1}^n a_k e^{i(2\pi n_k t)}$$

 This program solves those equations for three wheels, assuming all of the a coefficients are 1 and #
 coefficients are 1 and #
 This program solves those equations for three wheels, assuming all of the a coefficients are 1 and #
 coefficients are 1 and #
 The `$s3` coefficients are 1 and #
 More details online at <http://linuxgazette.net>



Markup Example

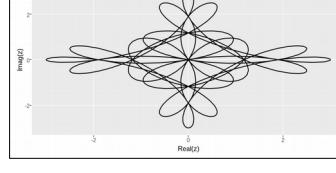
Notebook w/ HTML Widgets

15 Examples from the RStudio leaflet tutorial (<https://rstudio.github.io/leaflet/>)
 16 Introduction
 17 Build a basic map, using tiles from OpenStreetMap and adding a marker for Purdue University.
 18
 19 `library(leaflet)`
 20 `## (r warning=FALSE, message=FALSE)`
 21 `library(leaflet)`
 22 `pu_lng = -86.9212`
 23 `pu_lat = 40.4237`
 24
 25 `n <- leaflet() %>%`
 26 `addTiles() %>% # Add default OpenStreetMap map tiles`
 `addMarkers(lng=pu_lng, lat=pu_lat, popup="Purdue University")`
 27
 28
 29
 30
 31
 32

Fun with Spirographs
 The spirograph equation for three or more wheels can be generalized as follows:

$$z(t) = \sum_{k=1}^n a_k e^{i(2\pi n_k t)}$$

 This program solves those equations for three wheels, assuming all of the a coefficients are 1 and #
 coefficients are 1 and #
 This program solves those equations for three wheels, assuming all of the a coefficients are 1 and #
 coefficients are 1 and #
 The `$s3` coefficients are 1 and #
 More details online at <http://linuxgazette.net>

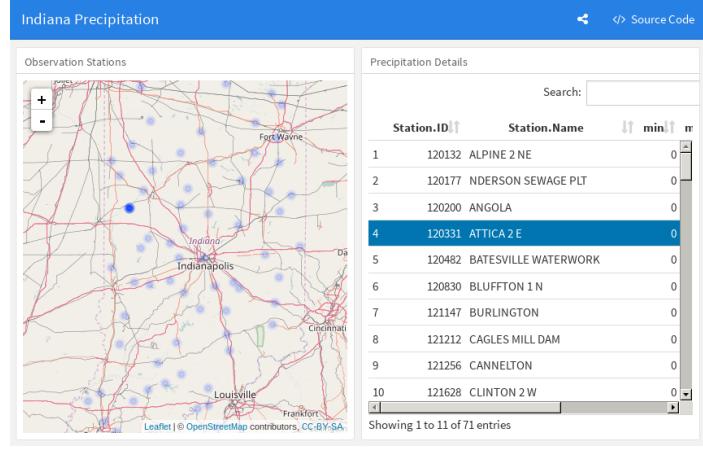



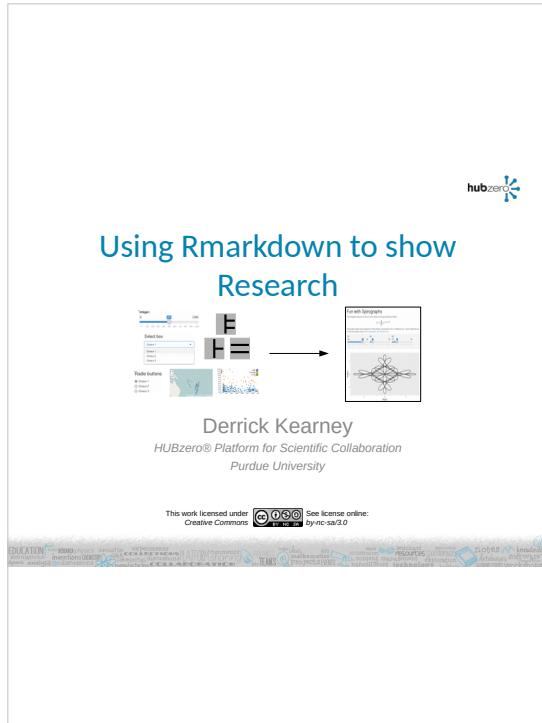
RMarkdown w/ Shiny Widgets

<https://github.com/codedsk/rplay>

RStudio and Shiny are trademarks of RStudio, Inc.

Layout and widget communication w/o Shiny





Literate computing has a long history with contributions to the idea reaching from its creator Donald Knuth, to Mathematica, and Matlab. More recently we've seen the introduction of Jupyter Notebooks.

Rstudio Inc, a small company focused on the R programming language has built an ecosystem of tools that bring Literate Programming capabilities to the R programming language, while nicely incorporating the reporting and analysis tools that have been a hallmark of the R programming language.

Today we'll get a brief introduction to a few of these tools and see how they can be deployed as resources on the HUB.

Some people write programs like this...



```
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)  
}
```



You'll notice this doesn't launch the normal vnc viewer. Instead, the user is served a javascript based user interface.

This is the Rstudio IDE. There are 4 main parts to the IDE

Some people write programs like this...



```
# Comments Suck
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) +
}

# Learn to read code!
result <- data.frame(x=Re(z),y=Im(z))
return (result)
}
```



You'll notice this doesn't launch the normal vnc viewer. Instead, the user is served a javascript based user interface.

This is the Rstudio IDE. There are 4 main parts to the IDE

Other people write programs like this...



```
# Function to produce a spirograph
# Accepts 3 parameters describing the size of our wheels

spiro <- function(n1,n2,n3) {
  t <- seq(0,1,length.out=1000)

  # Spirographs can be approximated with this formula
  z <- exp(1i*2*pi*n1*t) \
    + exp(1i*2*pi*n2*t) \
    + exp(1i*2*pi*n3*t) \
}

# store the results for the user
result <- data.frame(x=Re(z),y=Im(z))
return (result)
}
```

EDUCATION | RStudio IDE | RStudio Server | RStudio Cloud | RStudio Connect | RStudio Tools | RStudio Projects | RStudio Shiny | RStudio Notebooks | RStudio Databases | RStudio Reference

You'll notice this doesn't launch the normal vnc viewer. Instead, the user is served a javascript based user interface.

This is the Rstudio IDE. There are 4 main parts to the IDE

And a few people write programs like this... 

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

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

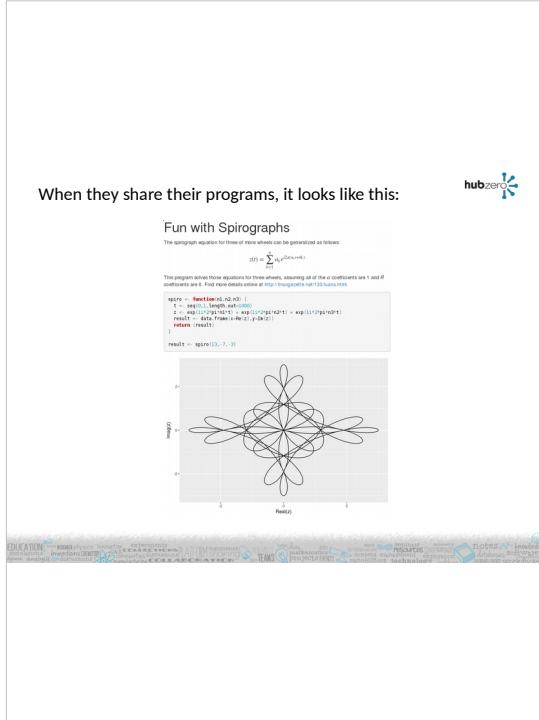
This program solves those equations **for** three wheels, assuming all of the **a**s coefficients are **1** and **\theta**s coefficients are **0**. Find more details online at <<http://linuxgazette.net/133/luana.html>>.

```
'''{r echo=TRUE}
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)
}
ggplot(data=spiro(13,-7,-3),aes(x=x,y=y)) + geom_path()
'''
```



You'll notice this doesn't launch the normal vnc viewer. Instead, the user is served a javascript based user interface.

This is the Rstudio IDE. There are 4 main parts to the IDE



You'll notice this doesn't launch the normal vnc viewer. Instead, the user is served a javascript based user interface.

This is the Rstudio IDE. There are 4 main parts to the IDE

Literate Programming

Donald Knuth. "Literate Programming (1984)" in *Literate Programming*. CSLI, 1992, pg. 99.

I believe that the time is ripe for significantly better documentation of programs, and that we can best achieve this by considering programs to be works of literature. Hence, my title: "Literate Programming."

Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.

The practitioner of literate programming can be regarded as an essayist, whose main concern is with exposition and excellence of style. Such an author, with thesaurus in hand, chooses the names of variables carefully and explains what each variable means. He or she strives for a program that is comprehensible because its concepts have been introduced in an order that is best for human understanding, using a mixture of formal and informal methods that reinforce each other.

literateprogramming.com (retrieved Apr 29, 2017)

There are a number of applications working in this space.

Mathematica

Jupyter Notebooks

Sweave for R

Apache Zeppelin

Rstudio with RMarkdown

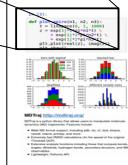
What are Interactive Notebooks?



Executable Code & Calculations

In [3]:

```
def plot_spiral(n1, n2, n3):
    t = linspace(0, 1, 1000)
    z = exp(j*2*pi*n1*t) \
        + exp(j*2*pi*n2*t) \
        + exp(j*2*pi*n3*t)
    plt.plot(real(z), imag(z))
    plt.show()
```



Jupyter Notebook
calculations.ipynb



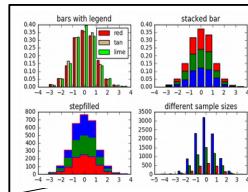
The most recent interpretations of this include Interactive Notebooks

What are Interactive Notebooks?

Executable Code & Calculations

Data Visualizations

```
In [3]:
def plot_spiral(n1, n2, n3):
    t = linspace(0, 1, 1000)
    z = exp(j*2*pi*n1*t) \
        + exp(j*2*pi*n2*t) \
        + exp(j*2*pi*n3*t)
    plt.plot(real(z), imag(z))
    plt.show()
```



Jupyter Notebook
calculations.ipynb

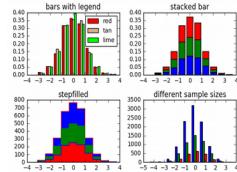


What are Interactive Notebooks?

Executable Code & Calculations

```
In [3]:
def plot_spiral(n1, n2, n3):
    t = linspace(0, 1, 1000)
    z = exp(j*2*pi*n1*t) \
        + exp(j*2*pi*n2*t) \
        + exp(j*2*pi*n3*t)
    plt.plot(real(z), imag(z))
    plt.show()
```

Data Visualizations



Jupyter Notebook calculations.ipynb

Narrative

MDTraj <http://mdtraj.org/>

MDTraj is a python library that allows users to manipulate molecular dynamics (MD) trajectories. Features include:

- Wide MD format support, including pdb, xtc, trr, dcd, binpos, netoff, mdcrd, pmtop, and more.
- Extremely fast RMSD calculations (4x the speed of the original Theobald GCP).
- Extensive analysis functions including those that compute bonds, angles, dihedrals, hydrogen bonds, secondary structure, and NMR observables.
- Lightweight, Pythonic API.

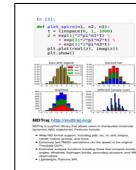


Sharing Notebooks



Sharing Made Easy

- Text based file format
- Outputs stored with document



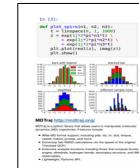
Sharing Notebooks



Sharing Made Easy

- Text based file format
- Outputs stored with document

Export Document to:
PDF, Word, HTML,
LaTeX, Slides,
Markdown, Wiki, EPub

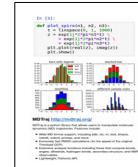


Sharing Notebooks



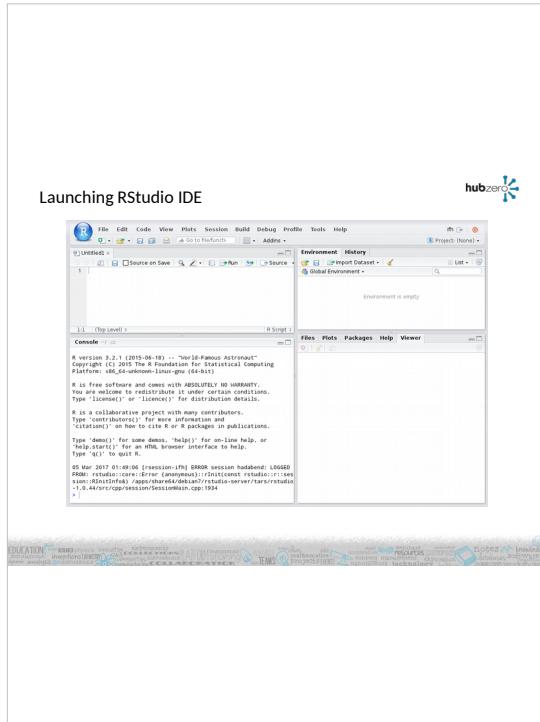
Sharing Made Easy
- Text based file format
- Outputs stored with document

Export Document to:
PDF, Word, HTML,
LaTeX, Slides,
Markdown, Wiki, EPub



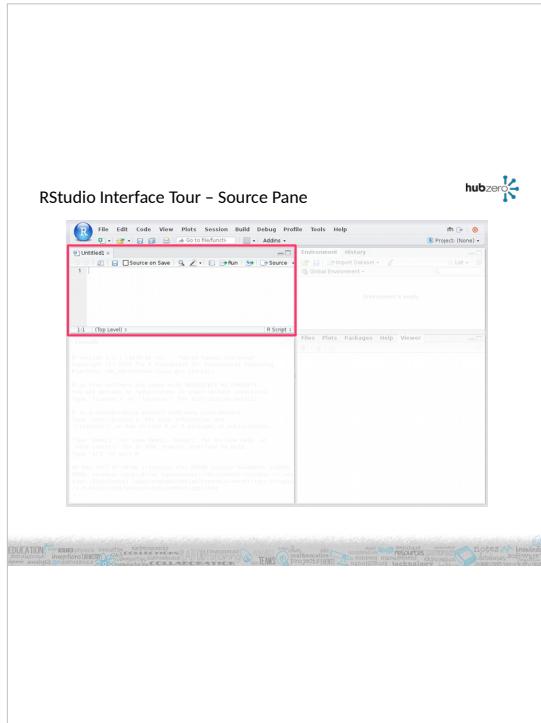
Embed Interactive
Web Applications





You'll notice this doesn't launch the normal vnc viewer. Instead, the user is served a javascript based user interface.

This is the Rstudio IDE. There are 4 main parts to the IDE



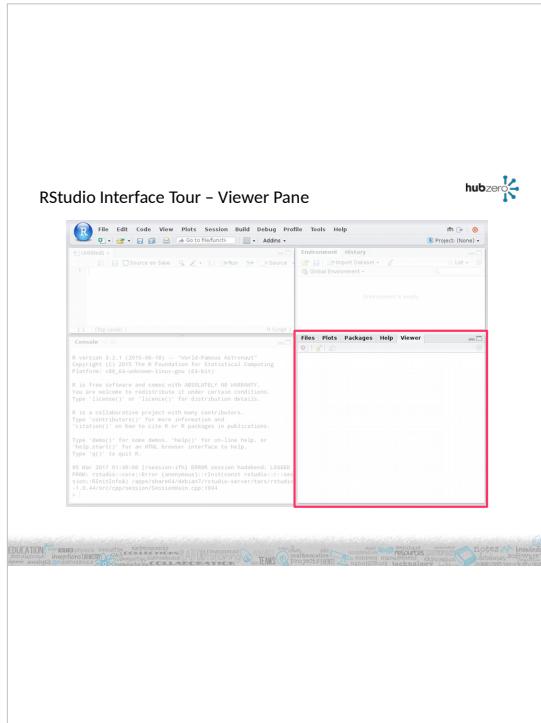
The Code Pane is where users manipulate source code



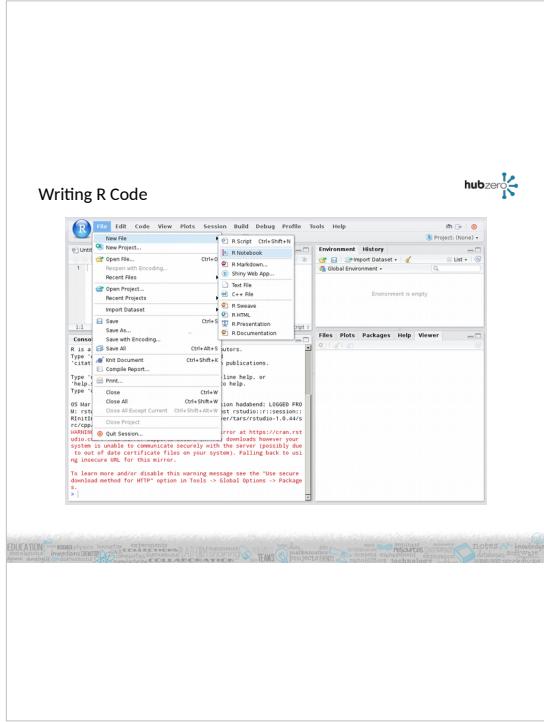
The console is where R commands are executed



The Environment / History pane shows variables and files that have been loaded into the ide environment.



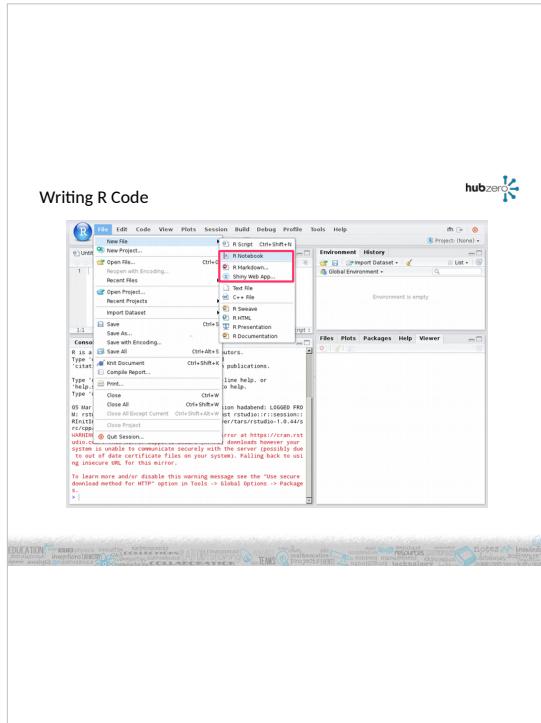
And the viewer pane shows output that can be visualized



To write code, use the **File → New** menu item to open a new file.

Notice there are a number of different types of files you can open.

You would probably expect to be able to open an R file to write a regular program in R, but there are three other types of files I'd like to highlight today.



Shiny Web Apps, RMarkdown, and the closely related R Notebooks

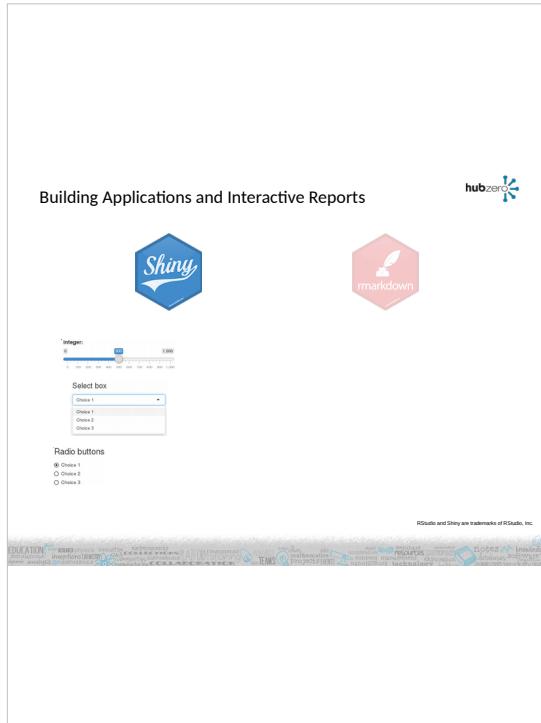


The R language has a history of allowing people to easily generate reports. People have been using Sweave to combine R code and plots with LaTeX for years.

RStudio, the company behind the graphical user interface, has been focusing on building ways for people to more easily disseminate their research results and analysis, particularly on the web.

Two packages they developed to make this happen include Shiny and RMarkdown.

Shiny is a toolkit that allows users to put together web apps.



It includes things like the normal input widgets you would expect, sliders, radio buttons, dropdown menus.



And output widgets like text boxes and static plots.

It also includes a number of interactive output widgets backed by JavaScript libraries like leaflet for mapping, Plotly for plots, and data tables.



And it allows users to either fully describe their layout using a grid system or one of their predefined templates.

Shiny applications are backed by a shiny server, which produces the HTML, CSS, and JavaScript that the user renders in their web browser.

Let's check out a couple of examples.

Shiny Example 1 - Spirograph

hubzero



Spirograph Example

<https://github.com/codedsk/hubzero-tool-spiro-shiny-1>

Indiana Precip Example

<https://github.com/codedsk/rplay/mapping/indiana-precip-shiny>

RSudio and shiny are trademarks of RStudio, Inc.

EDUCATION | R | RStudio | RStudio Server | RStudio Server Pro | RStudio IDE | RStudio Connect | RStudio Tools | RStudio Projects | RStudio Team | RStudio Analytics | RStudio DataScience | RStudio DataScience Reference

<https://github.com/codedsk/hubzero-tool-spiro-shiny-1>

Shiny Example 2 – Indiana Precipitation

hubzero

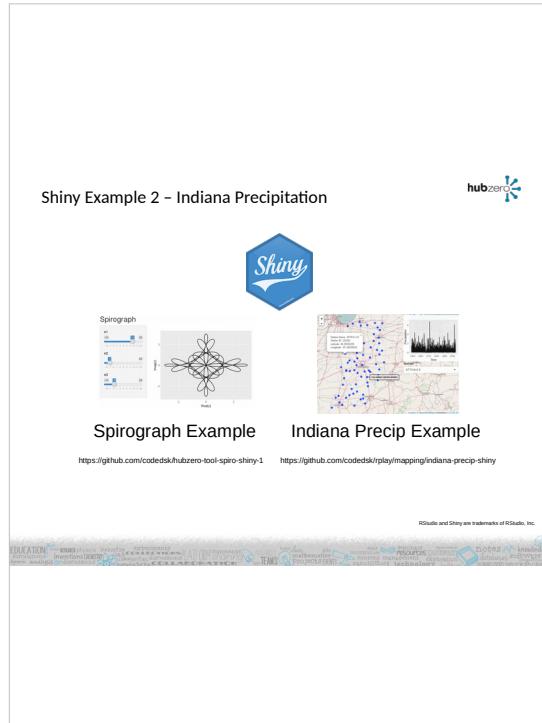


Spirograph Example **Indiana Precip Example**

<https://github.com/codedsk/hubzero-tool-spiro-shiny-1> <https://github.com/codedsk/rplay/mapping/indiana-precip-shiny>

RSudio and Shiny are trademarks of RStudio, Inc.

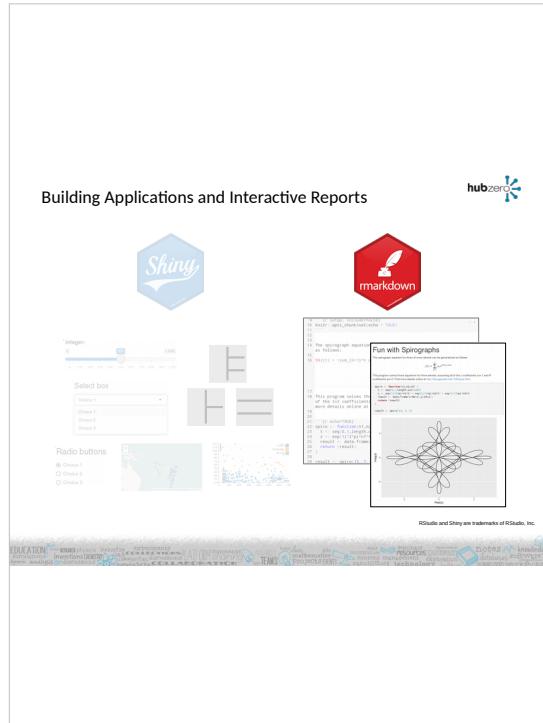
EDUCATION TEAM PROJECTS LIBRARIES SUPPORT



<https://github.com/codedsk/rplay/mapping/indiana-precip-shiny>



RMarkdown is a type of markup language, based on the markdown syntax that holds a few additional features including being able to hold R code.



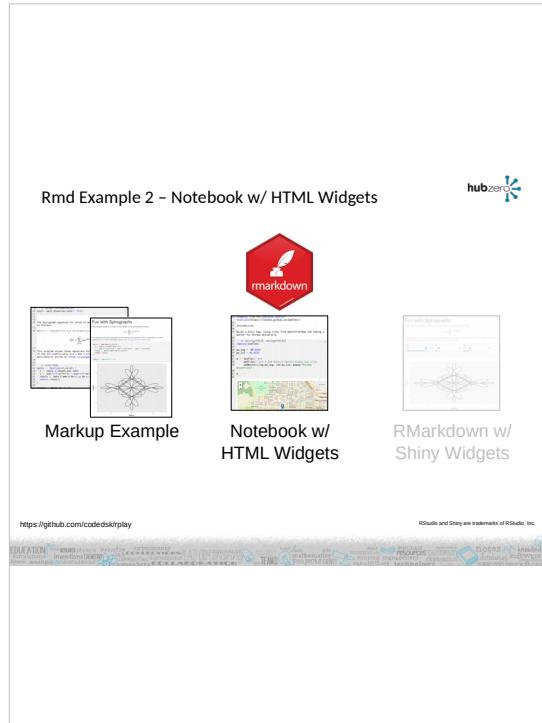
RMarkdown files can be converted to HTML files for distribution on the web or PDF files. Additionally they can be converted to Word document, LaTeX documents and a handful of other formats.

That's one of the strengths of the system.

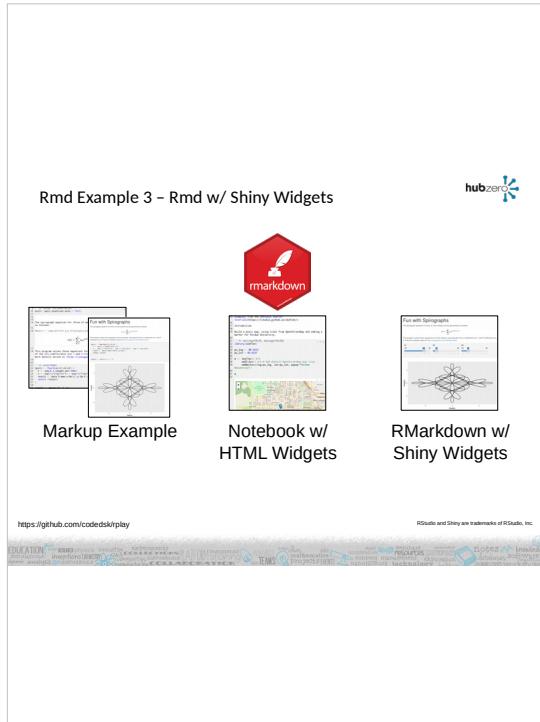


Let's check out some examples

http://github.com/codedsk/rplay/rmarkdown/plain_markup.Rmd



[http://github.com/codedsk/rplay/mapping/notebook_htmlwidgets_example.Rmd](https://github.com/codedsk/rplay/mapping/notebook_htmlwidgets_example.Rmd)



[http://github.com/codedsk/hubzero-tool-spiro-rmd-html-rshiny-widgets/src/spirograph_with_shiny_widgets.Rmd](https://github.com/codedsk/hubzero-tool-spiro-rmd-html-rshiny-widgets/src/spirograph_with_shiny_widgets.Rmd).



https://github.com/codedsk/rplay/mapping/indiana_precip_dashboard_1.Rmd

Shiny apps require a server and provide great flexibility to control how widgets react to changes to each other.

The flexdashboard module allows users to layout a dashboard using html widgets.

The crosstalk module allows you to coordinate communication between the widgets so as you filter with say a map widget, an xy plot also sees the changes and reacts accordingly.