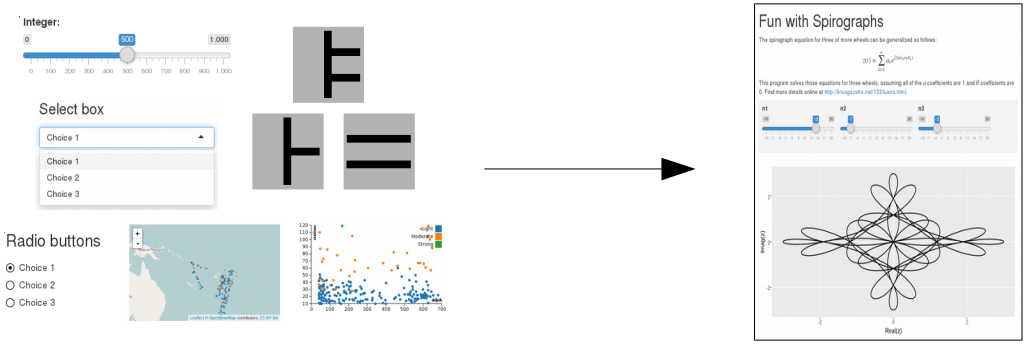
Using Rmarkdown to show Research



Derrick Kearney

HUBzero® Platform for Scientific Collaboration

Purdue University

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**:**

$$**z(t) **=** sum_{*k*=1}**^**n a_k e**^**{i2pi(n_kt**+**theta_k)}**$$

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**>**.

```{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

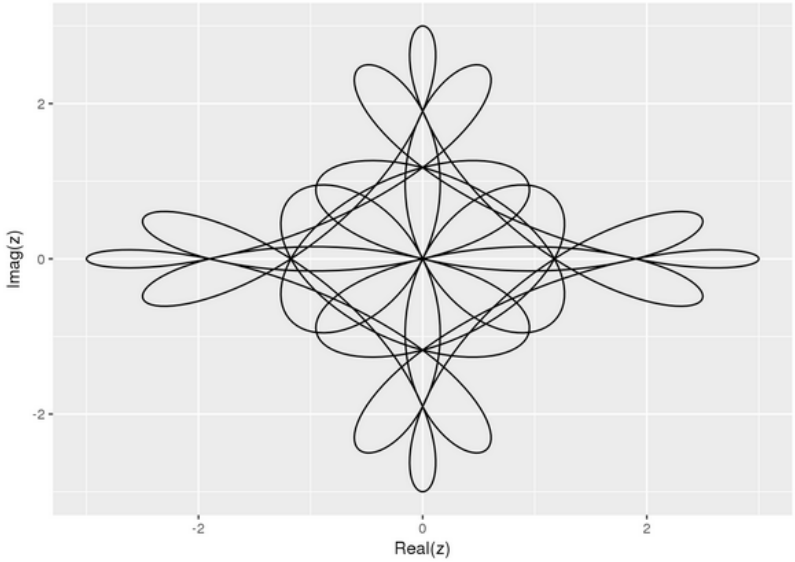
The spirograph equation for three of more wheels can be generalized as follows:

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

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)
```



Literate Programming

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

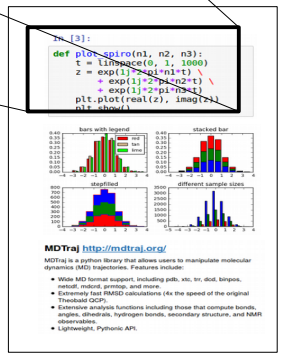
> I believe that the time is ripe for significantly better documentation of programs, and that we can best achieve this by considering programs to be works of literature. Hence, my title: "Literate Programming."
>
> Let us change our traditional attitude to the construction of programs: Instead of imagining 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_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()
```



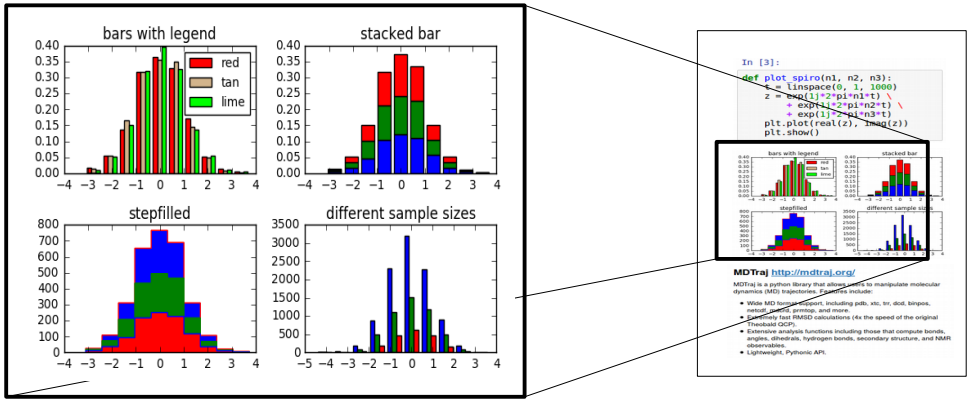
Jupyter Notebook
calculations.ipynb

What are Interactive Notebooks?

Executable Code
& Calculations

Data Visualizations

```
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

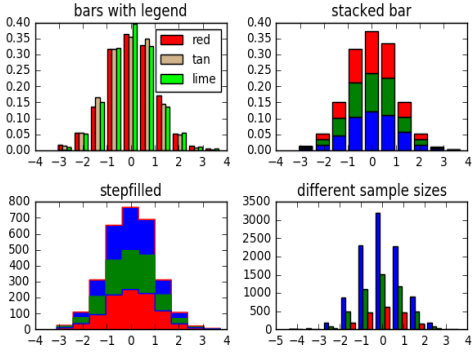
What are Interactive Notebooks?

Executable Code
& Calculations

Data Visualizations

Narrative

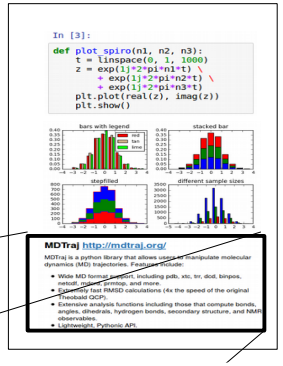
```
In [3]:
def plot_spiro(n1, n2, n3):
    t = linspace(0, 1, 1000)
    z = exp(1j*2*pi*n1*t) \
        + exp(1j*2*pi*n2*t) \
        + exp(1j*2*pi*n3*t)
    plt.plot(real(z), imag(z))
    plt.show()
```



MDTraj http://mdtraj.org/

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

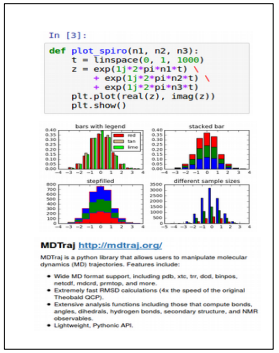
- Wide MD format support, including pdb, xtc, trr, dcd, binpos, netcdf, mdcrd, prmtop, and more.
- Extremely fast RMSD calculations (4x the speed of the original Theobald QCP).
- Extensive analysis functions including those that compute bonds, angles, dihedrals, hydrogen bonds, secondary structure, and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



Jupyter Notebook
calculations.ipynb

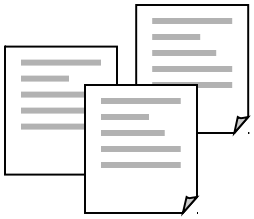
Sharing Notebooks

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

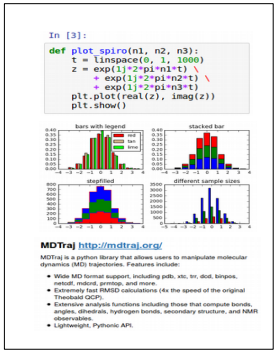


Sharing Notebooks

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

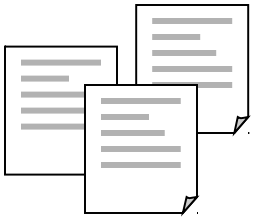


Export Document to:
PDF, Word, HTML, LaTeX, Slides, Markdown, Wiki, EPub

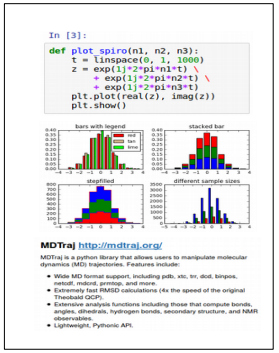


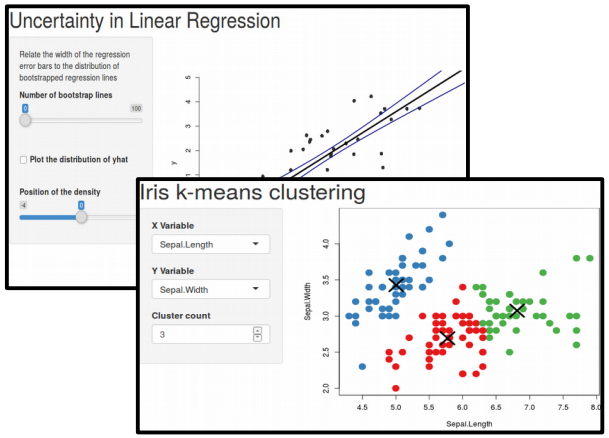
Sharing Notebooks

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



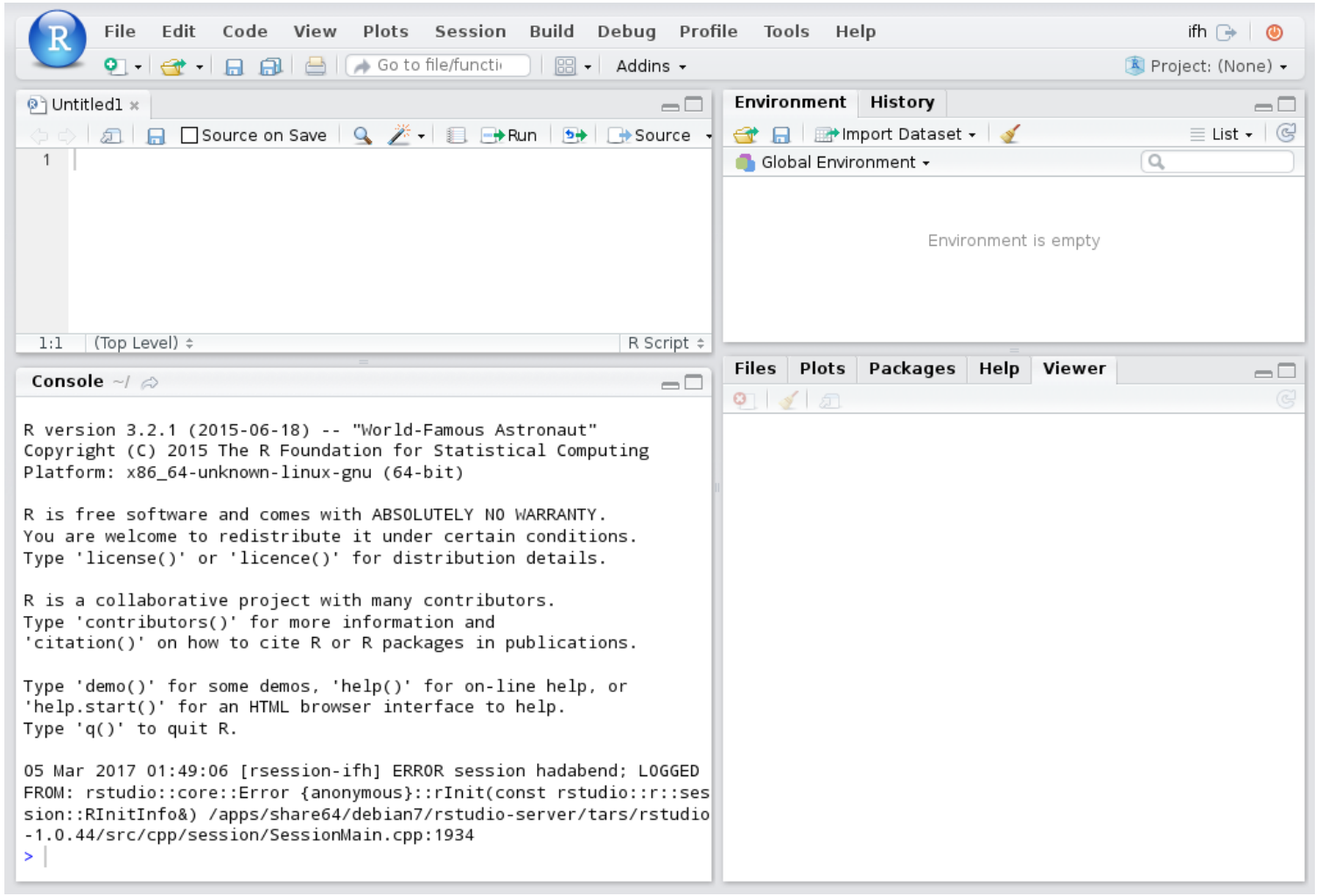
Export Document to:
PDF, Word, HTML, LaTeX, Slides, Markdown, Wiki, EPub



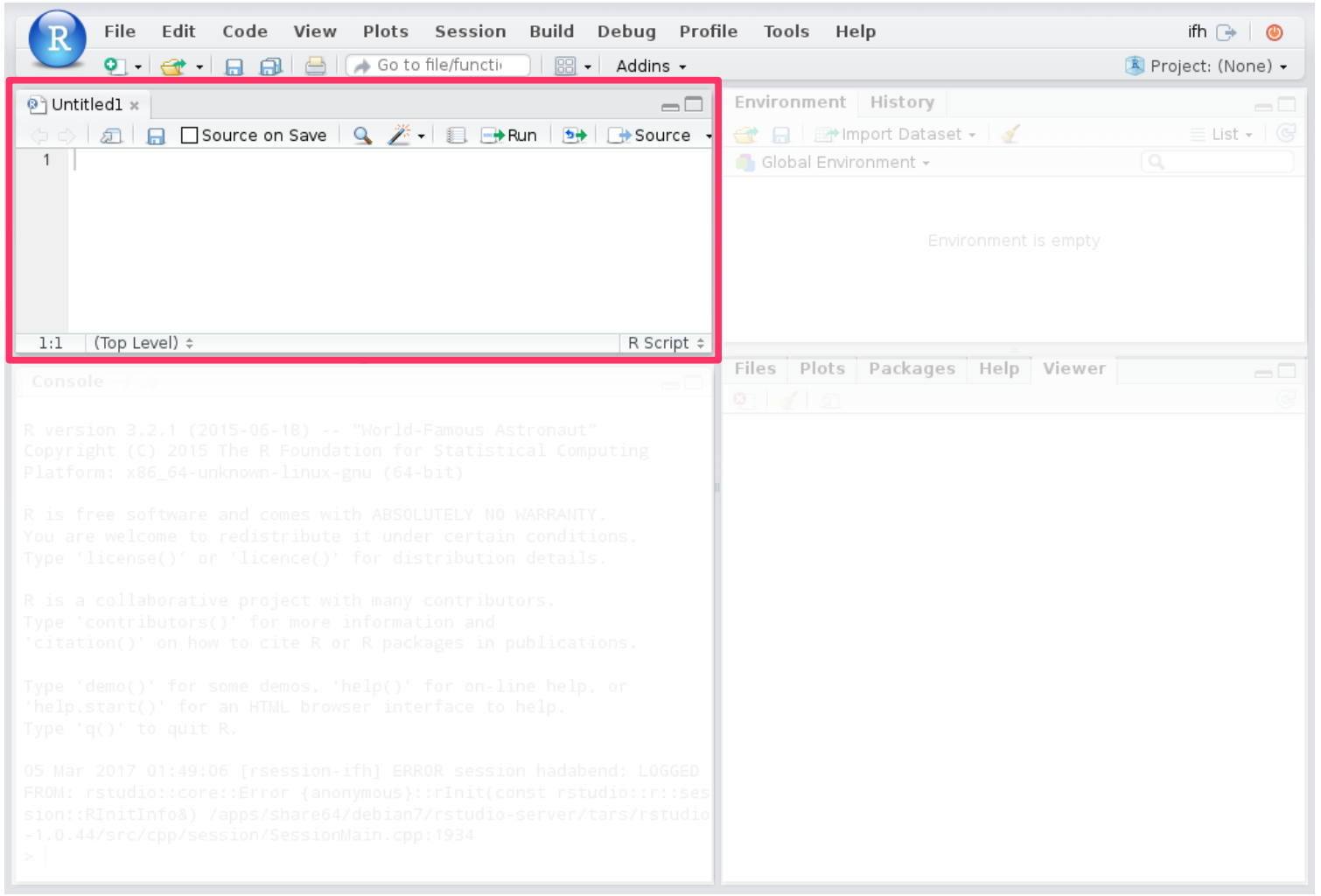


Embed Interactive Web Applications

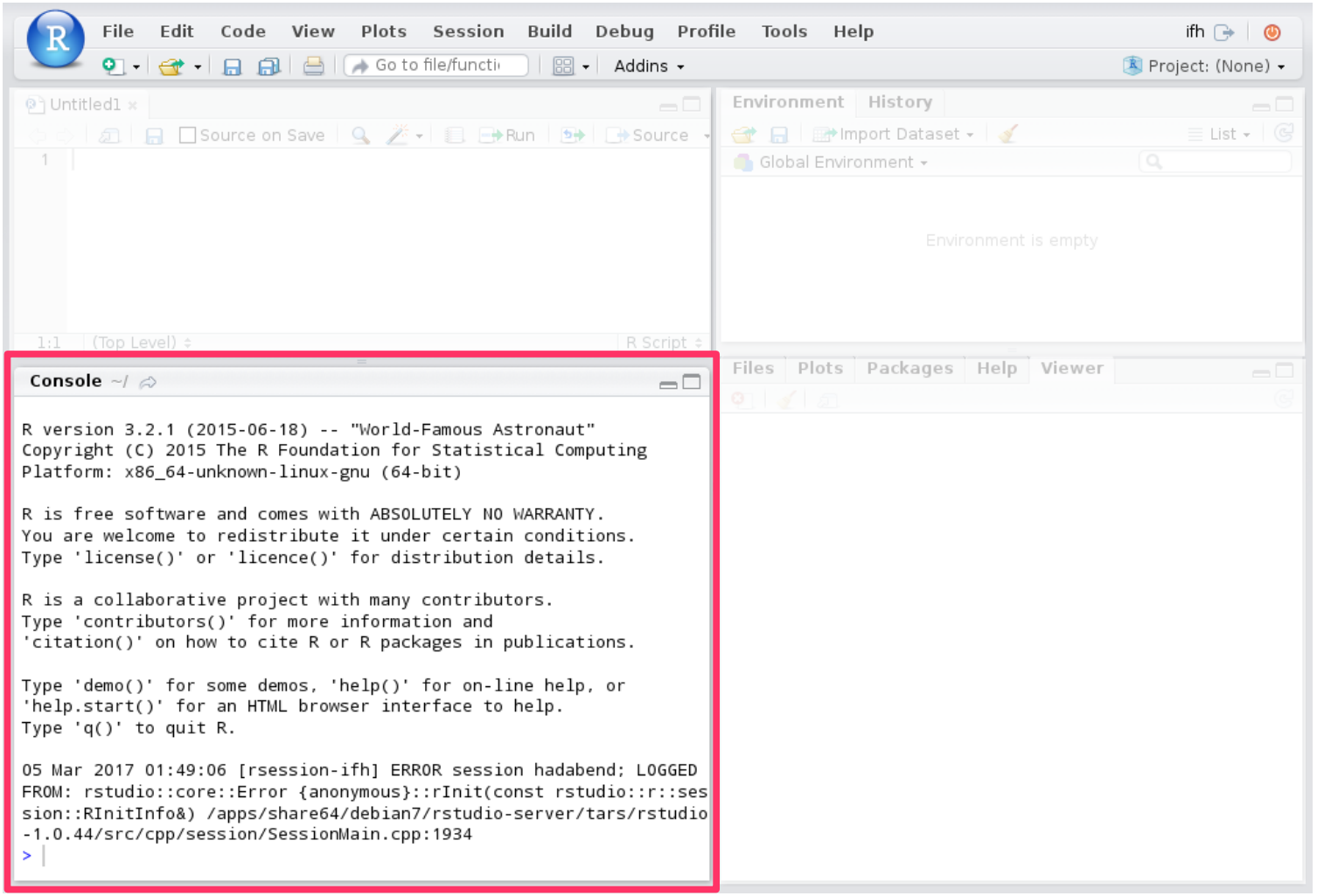
Launching RStudio IDE



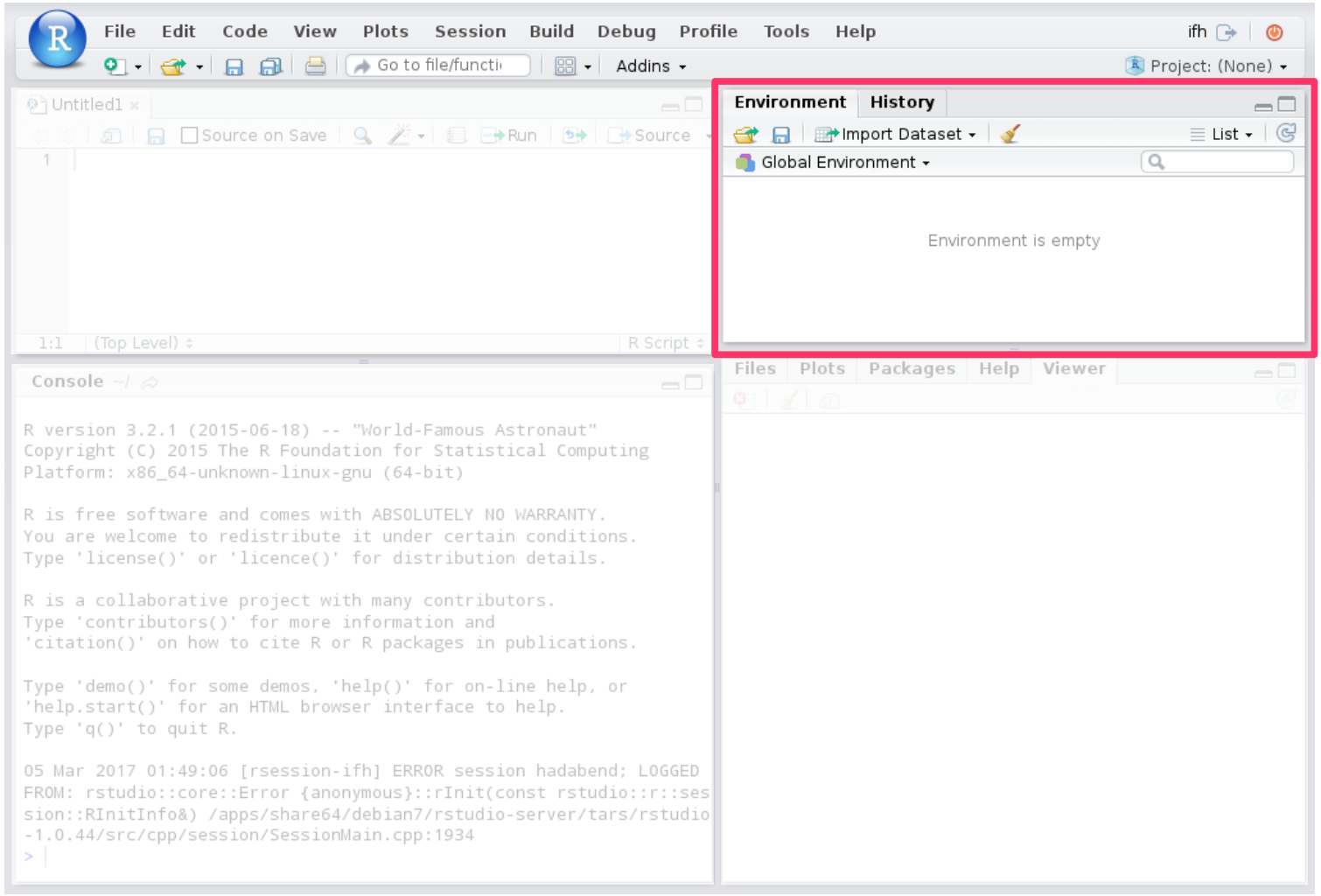
RStudio Interface Tour – Source Pane



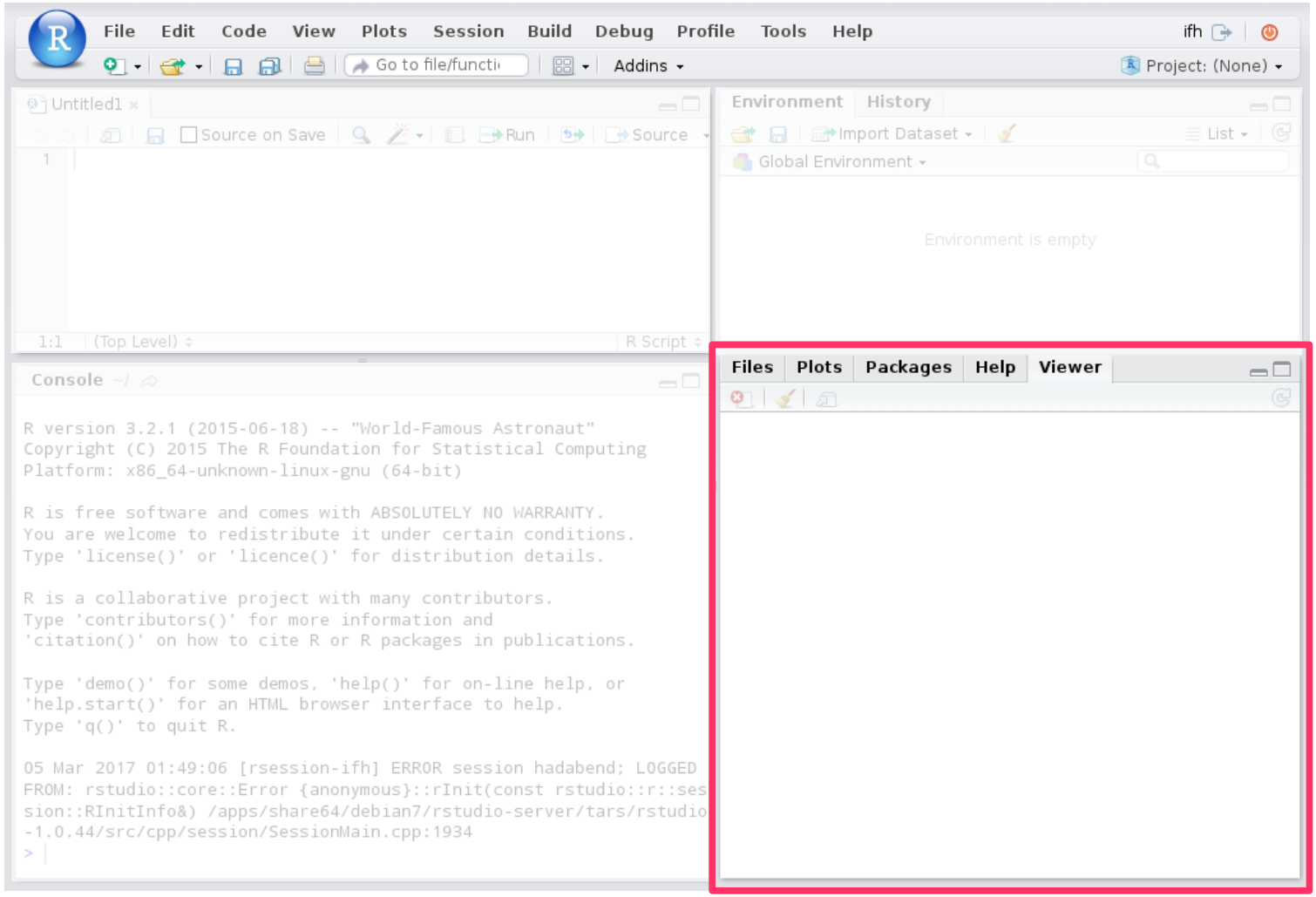
RStudio Interface Tour – Console Pane



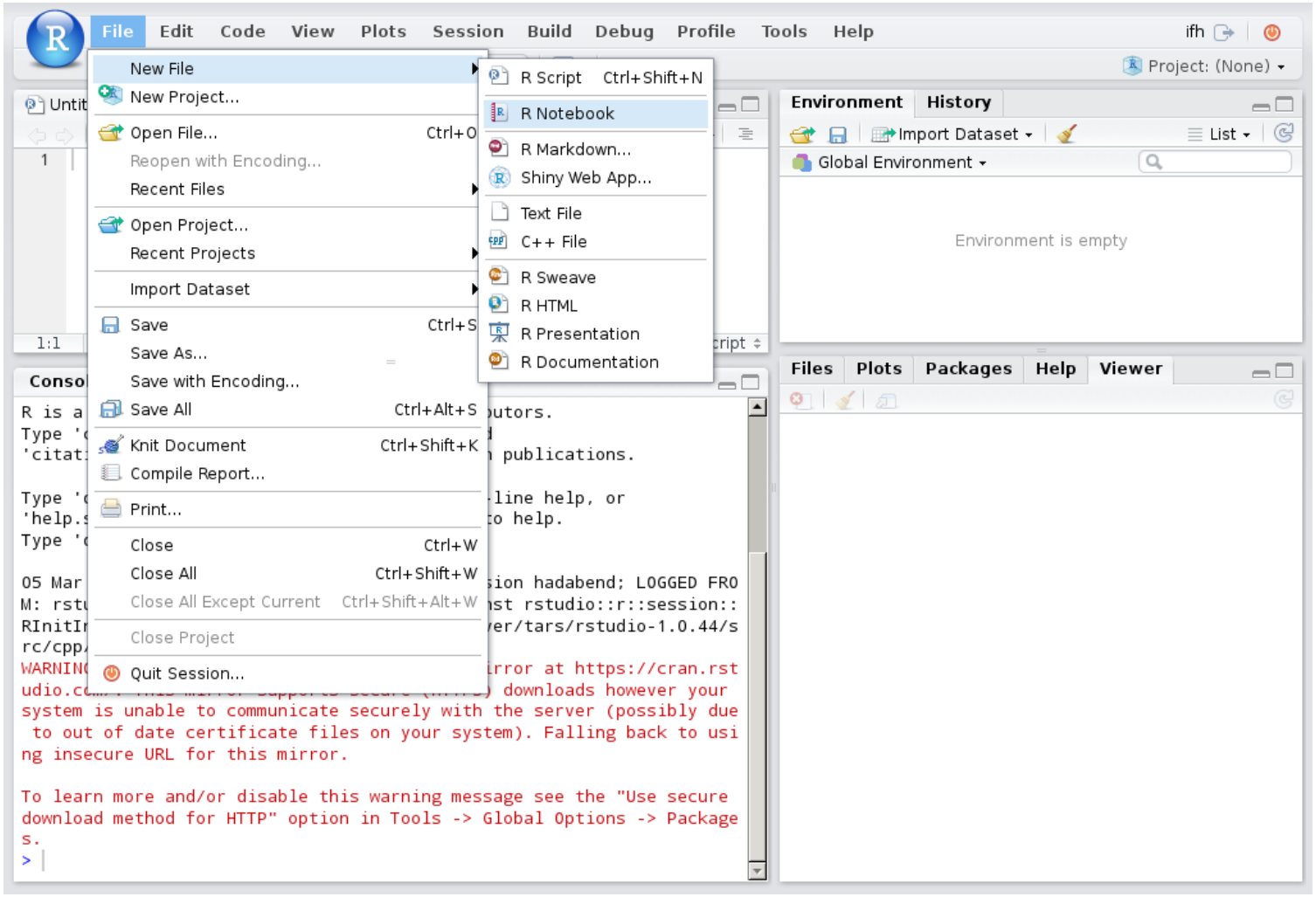
RStudio Interface Tour – Environment Pane



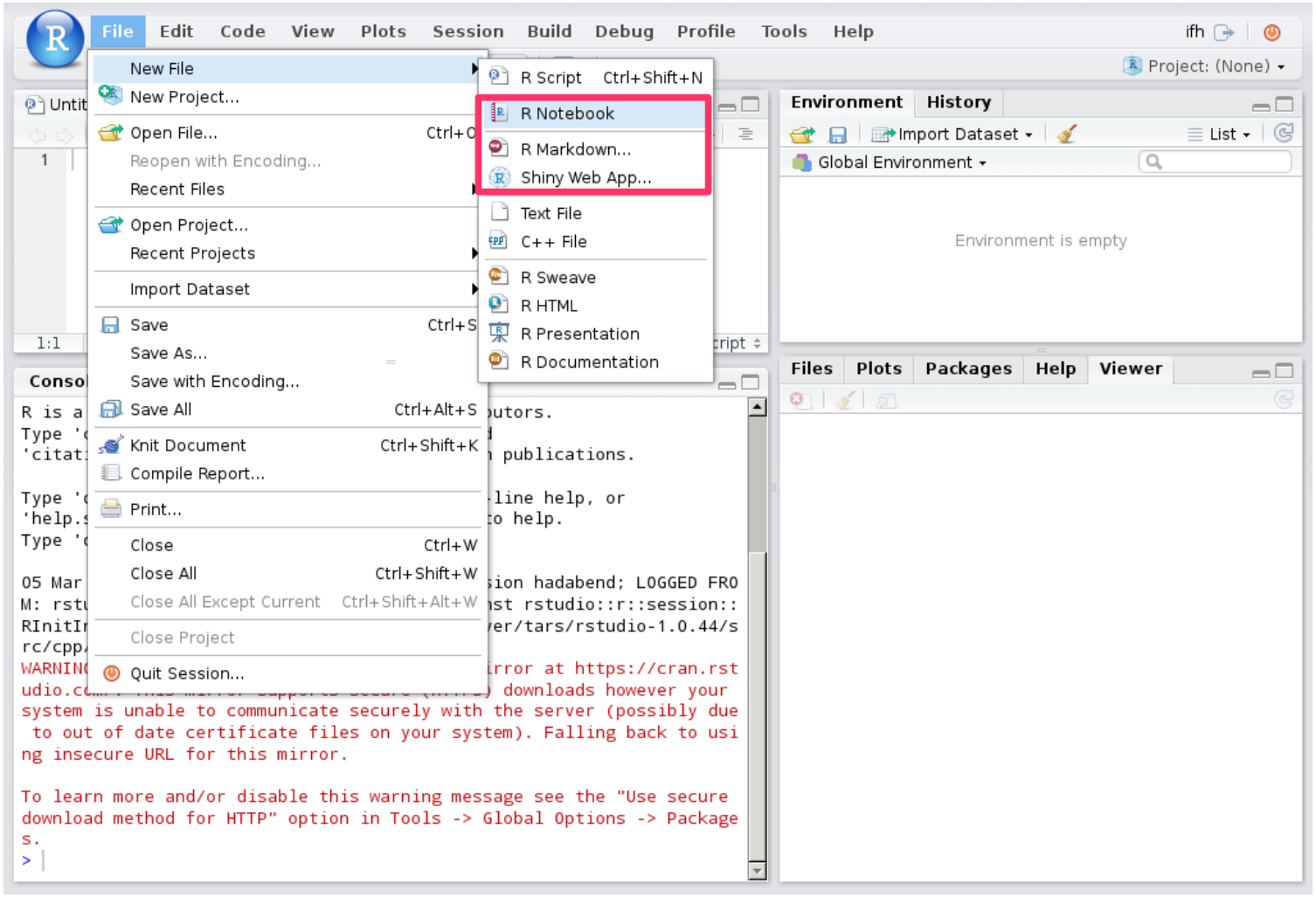
RStudio Interface Tour – Viewer Pane



Writing R Code

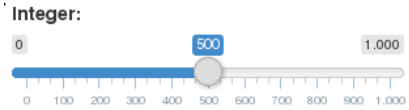


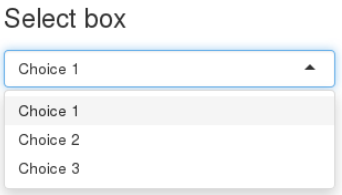
Writing R Code

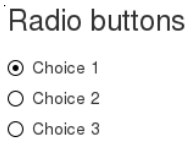


Building Applications and Interactive Reports

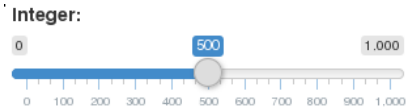
Building Applications and Interactive Reports

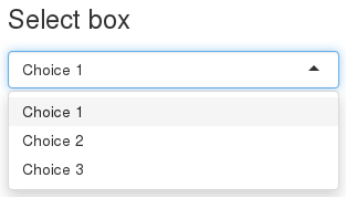


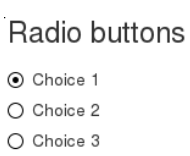


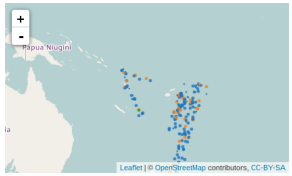


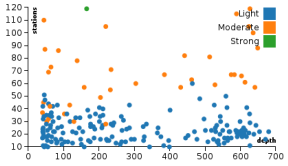
Building Applications and Interactive Reports



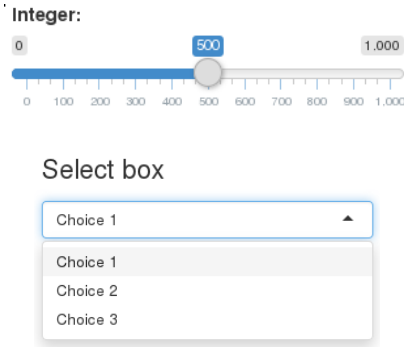


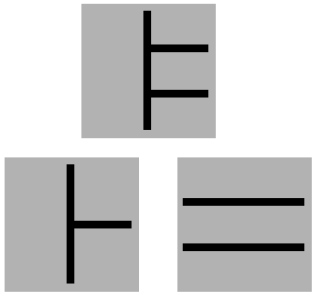


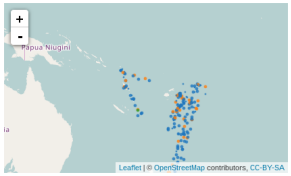


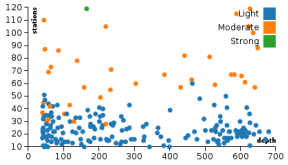


Building Applications and Interactive Reports

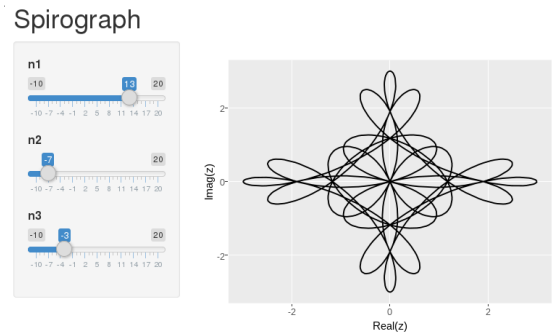


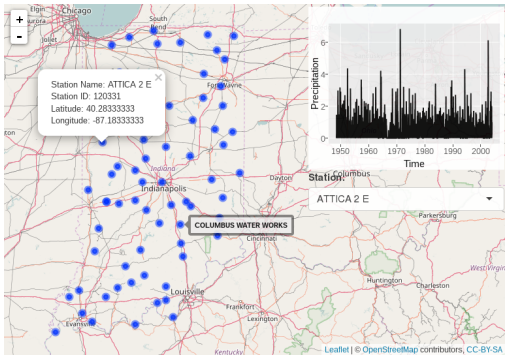






Shiny Example 1 - Spirograph





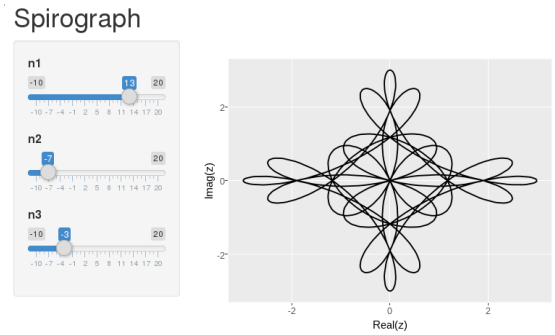
Spirograph Example

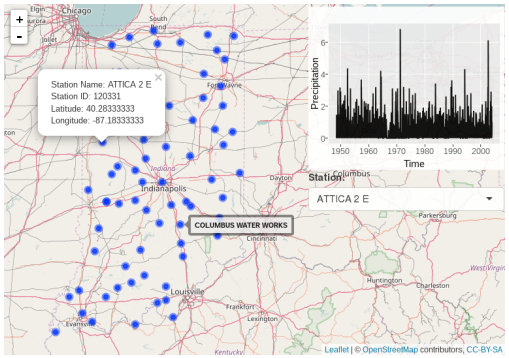
https://github.com/codedsk/hubzero-tool-spiro-shiny-1

Indiana Precip Example

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

Shiny Example 2 – Indiana Precipitation





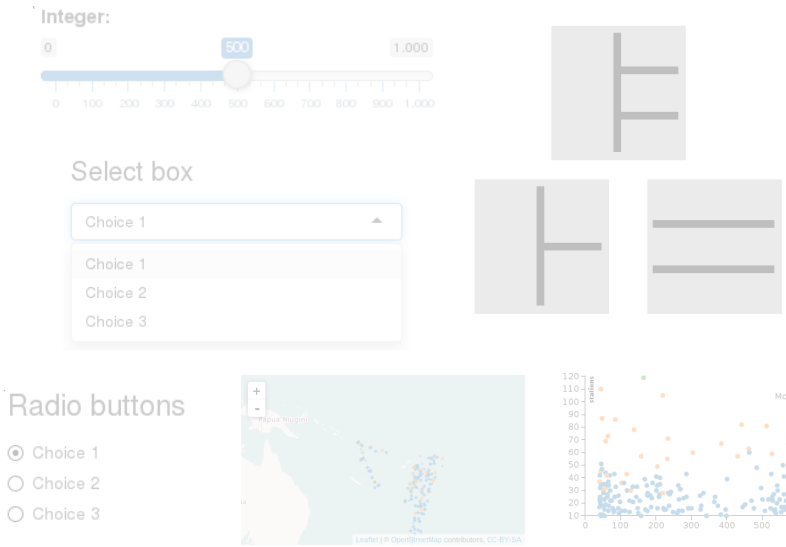
Spirograph Example

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

Indiana Precip Example

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

Building Applications and Interactive Reports




````
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```


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

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

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

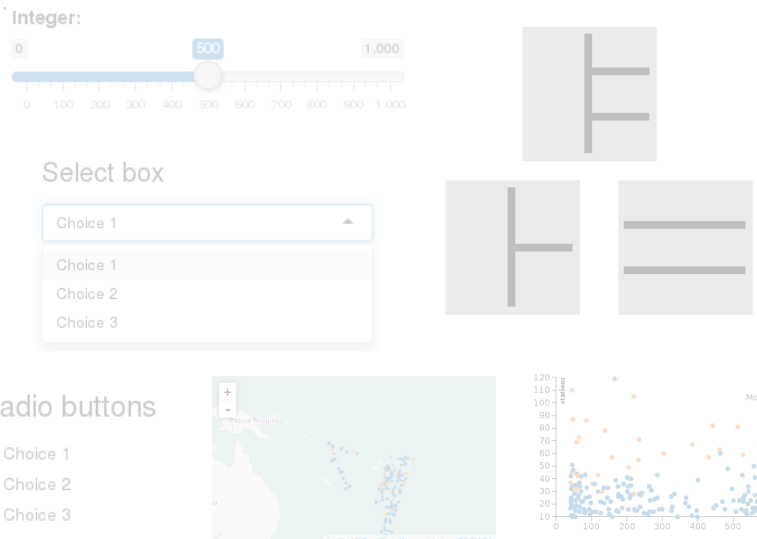
This program solves those equations for three wheels, assuming all of the $a$ coefficients are 1 and $\theta$ coefficients are 0. Find more details online at <http://linuxgazette.net/133/luana.html>.

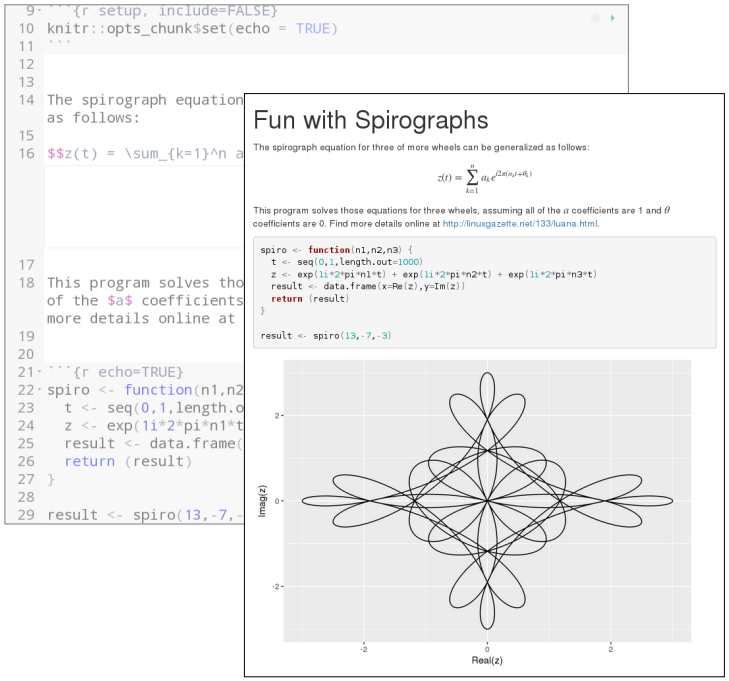

```{r echo=TRUE}
spiro <- function(n1,n2,n3) {
  t <- seq(0,1,length.out=1000)
  z <- exp(1i*2*pi*n1*t) + exp(1i*2*pi*n2*t) + exp(1i*2*pi*n3*t)
  result <- data.frame(x=Re(z),y=Im(z))
  return (result)
}

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

RStudio and Shiny are trademarks of RStudio, Inc.
````

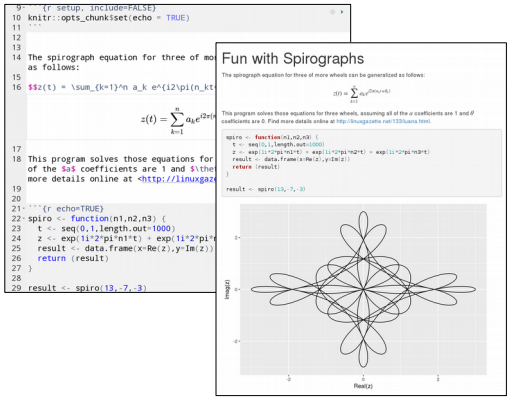
Building Applications and Interactive Reports





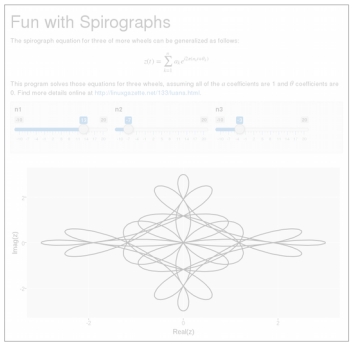
RStudio and Shiny are trademarks of RStudio, Inc.

Rmd Example 1 – Markup



Markup Example

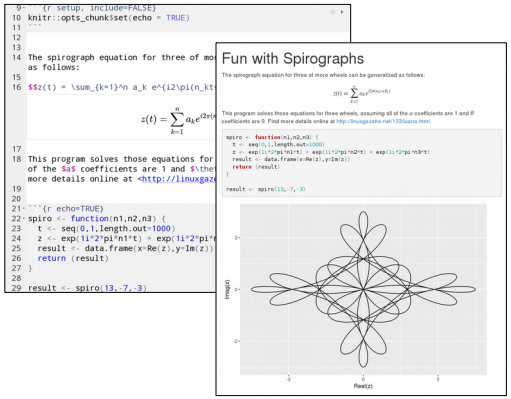
Notebook w/ HTML Widgets



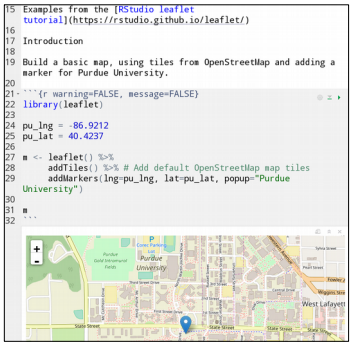
RMarkdown w/ Shiny Widgets

https://github.com/codedsk/rplay

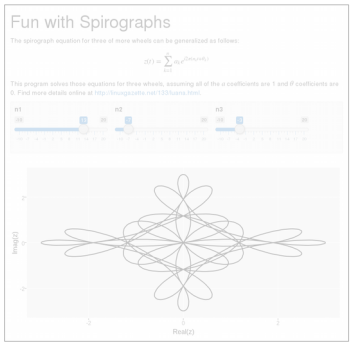
Rmd Example 2 – Notebook w/ HTML Widgets



Markup Example



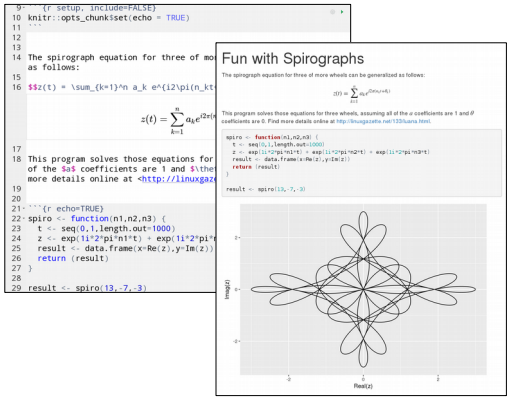
Notebook w/ HTML Widgets



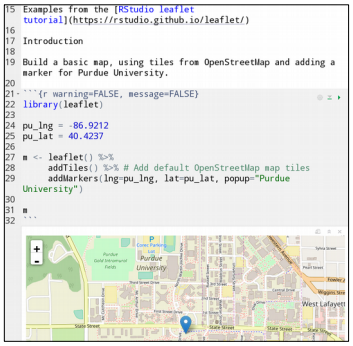
RMarkdown w/ Shiny Widgets

https://github.com/codedsk/rplay

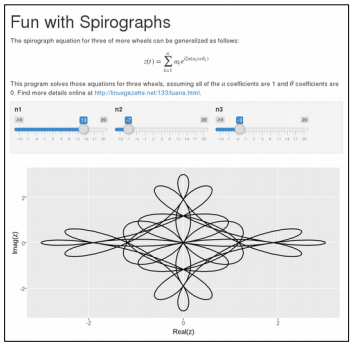
Rmd Example 3 – Rmd w/ Shiny Widgets



Markup Example



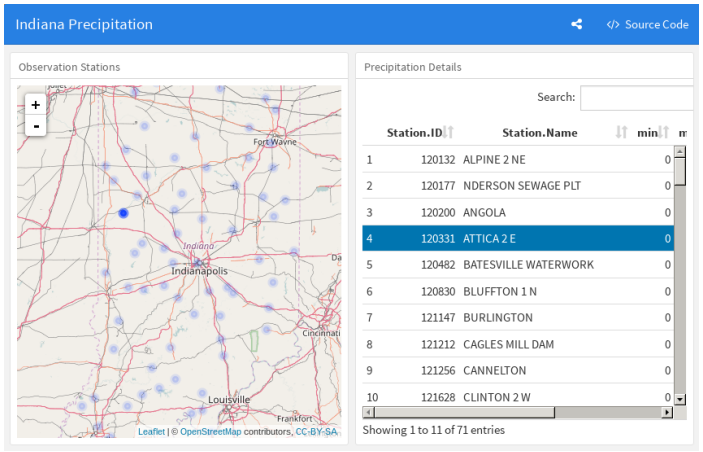
Notebook w/ HTML Widgets

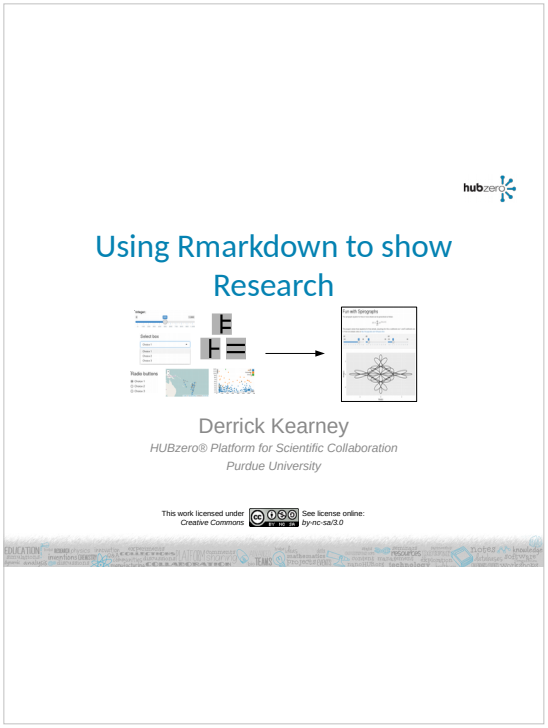


RMarkdown w/ Shiny Widgets

https://github.com/codedsk/rplay

Layout and widget communication w/o Shiny

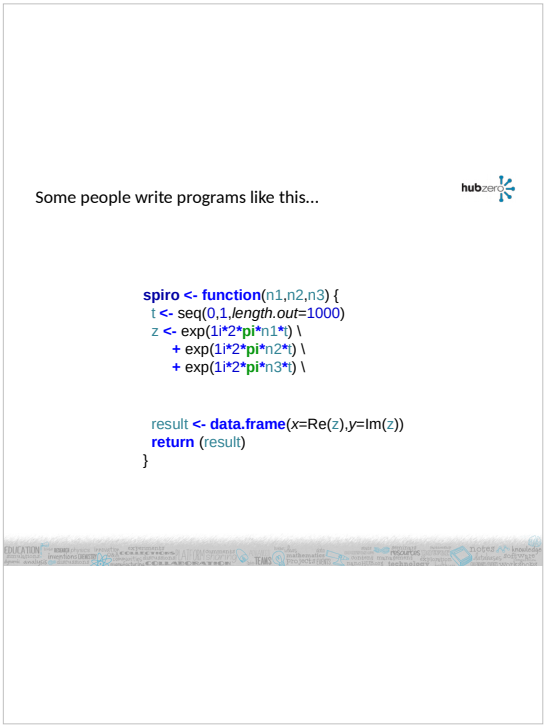




Literate computing has a long history with contributions to the idea reaching from its creator Donald Knuth, to Mathematica, and Matlab. More recently we've seen the introduction of Jupyter Notebooks.

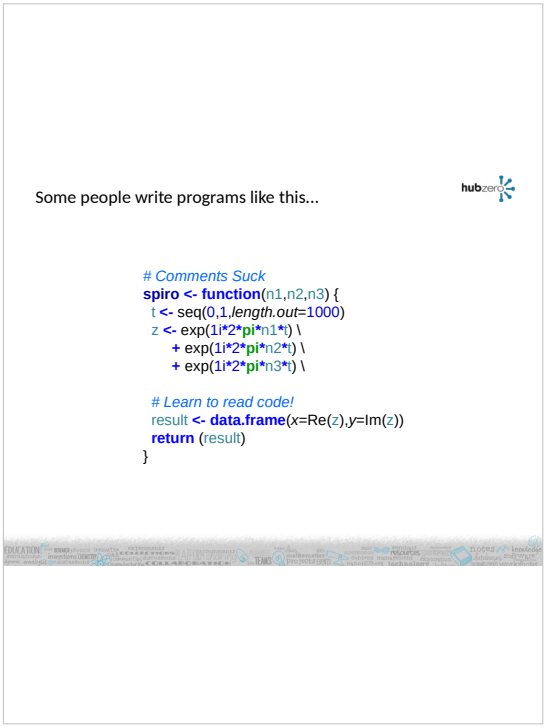
Rstudio Inc, a small company focused on the R programming language has built an ecosystem of tools that bring Literate Programming capabilities to the R programming language, while nicely incorporating the reporting and analysis tools that have been a halmark of the R programming language.

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.



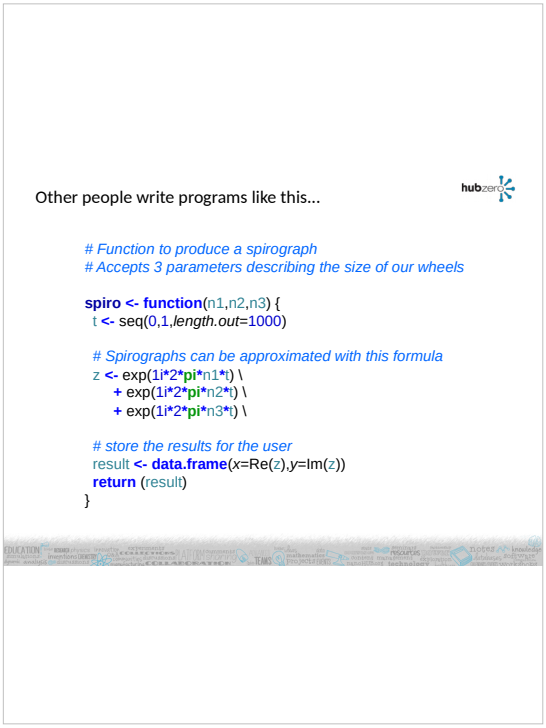
You'll notice this doesn't launch the normal vnc viewer. Instead, the user is served a javascript based user interface.

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



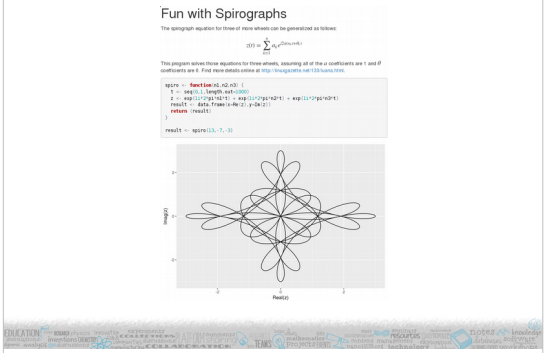
You'll notice this doesn't launch the normal vnc viewer. Instead, the user is served a javascript based user interface.

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.

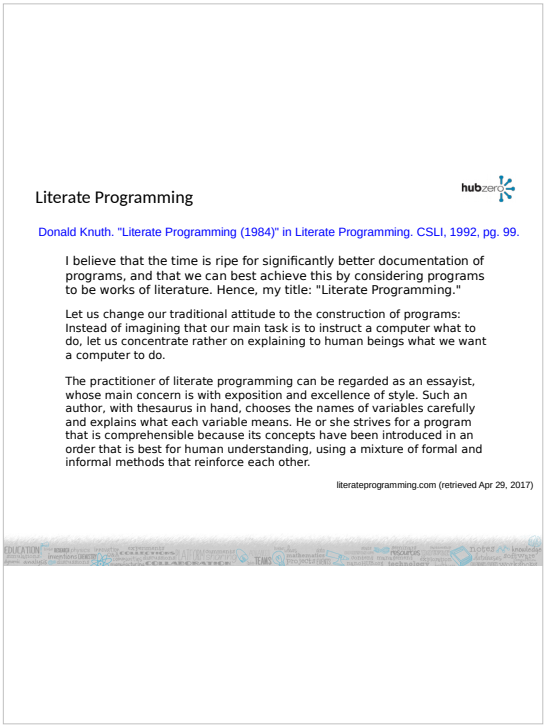
This is the Rstudio IDE. There are 4 main parts to the IDE

When they share their programs, it looks like this:



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



There are a number of applications working in this space.

Mathematica

Jupyter Notebooks

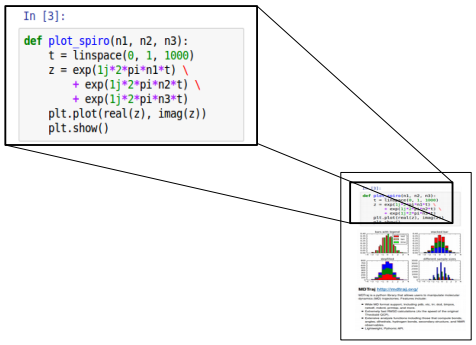
Sweave for R

Apache Zepplin

Rstudio with RMarkdown

What are Interactive Notebooks?

Executable Code
& Calculations



Jupyter Notebook
calculations.ipynb

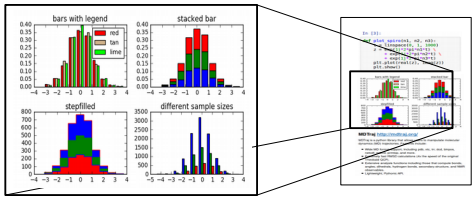
The most recent interpretations of this include Interactive Notebooks

What are Interactive Notebooks?

Executable Code
& Calculations

```
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()
```

Data Visualizations



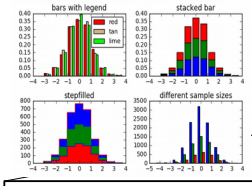
Jupyter Notebook
calculations.ipynb

What are Interactive Notebooks?

Executable Code & Calculations

```
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()
```

Data Visualizations



Narrative

MDTraj http://mdtraj.org/

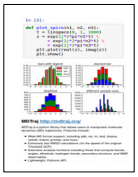
MDTraj is a python library that allows users to manipulate molecular dynamics (MD) trajectories. Features include:

- Wide MD format support, including pdb, xtc, trr, dcd, binpos, 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

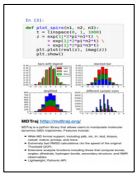
Sharing Made Easy
- Text based file format
- Outputs stored with document



Sharing Notebooks

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

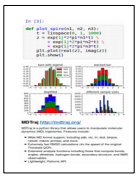
Export Document to:
PDF, Word, HTML, LaTeX, Slides, Markdown, Wiki, EPub



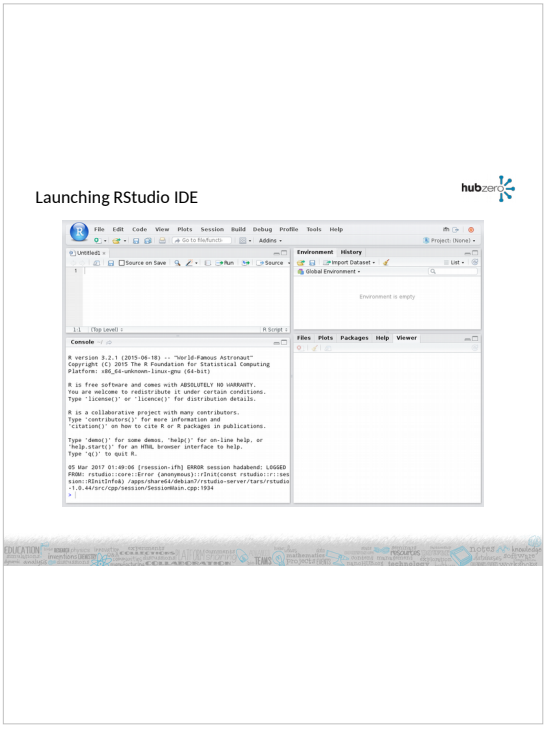
Sharing Notebooks

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

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

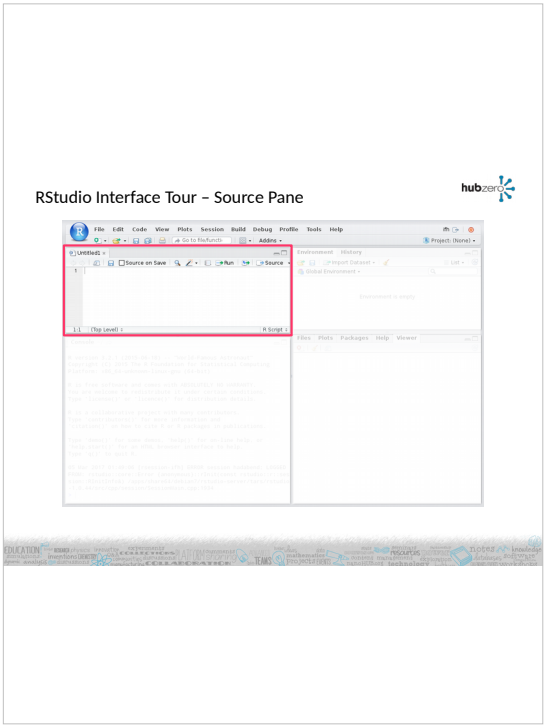


Embed Interactive Web Applications

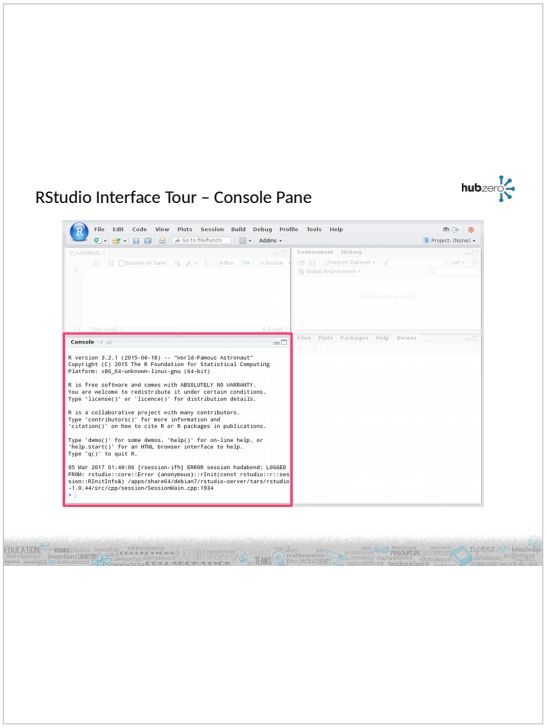


You'll notice this doesn't launch the normal vnc viewer. Instead, the user is served a javascript based user interface.

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



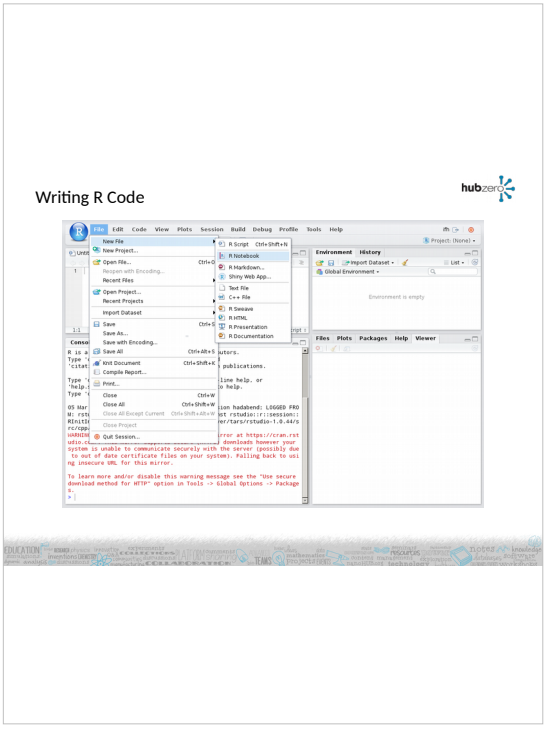
The Code Pane is where users manipulate source code



The console is where R commands are executed

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

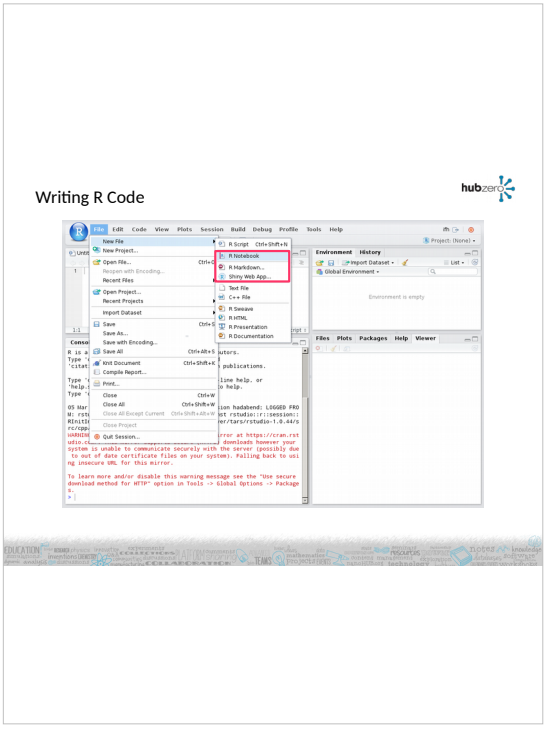
And the viewer pane shows output that can be visualized



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.



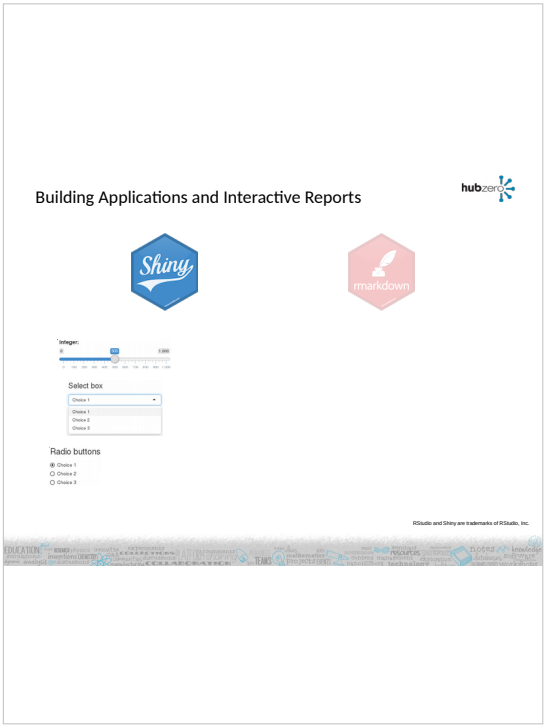
Shiny Web Apps, RMarkdown, and the closely related R Notebooks

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.

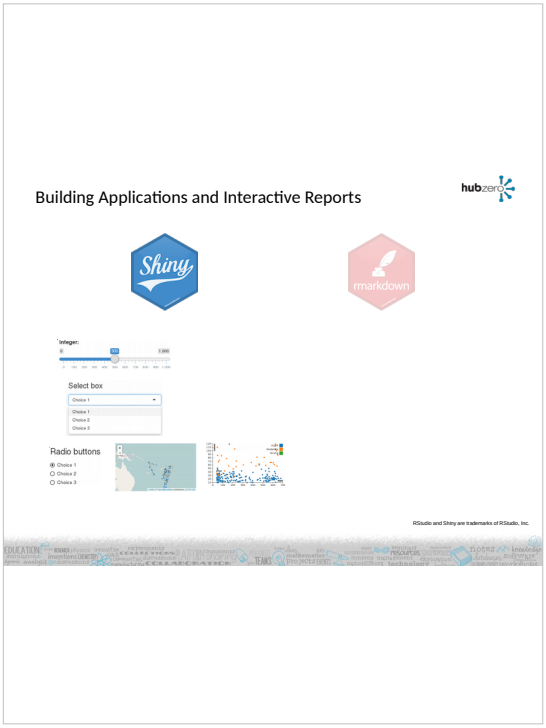
RStudio, the company behind the graphical user interface, has been focusing on building ways for people to more easily disseminate their research results and analysis, particularly on the web.

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.



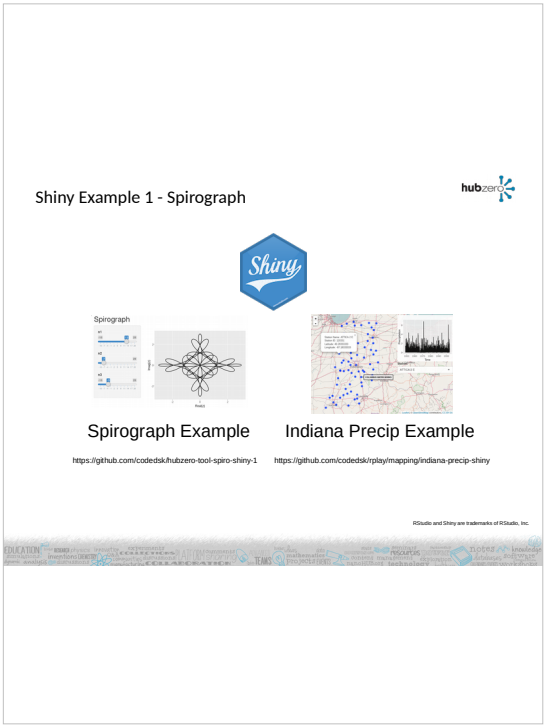
And output widgets like text boxes and static plots.

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

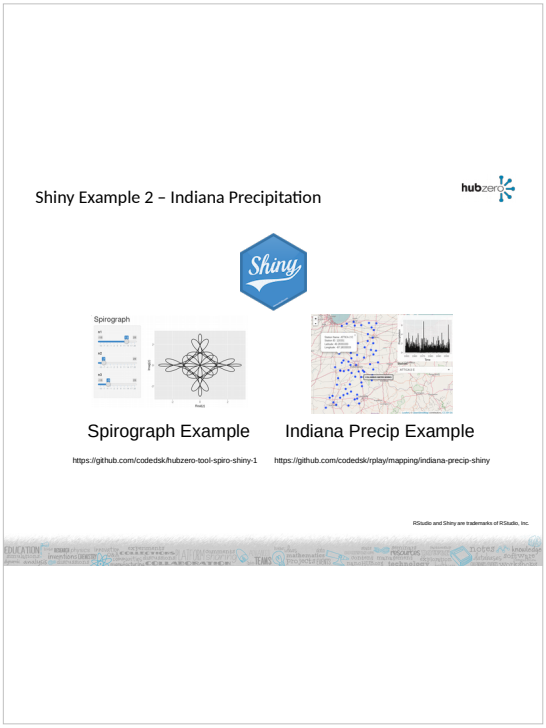
And it allows users to either fully describe their layout using a grid system or one of their predefined templates.

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.



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

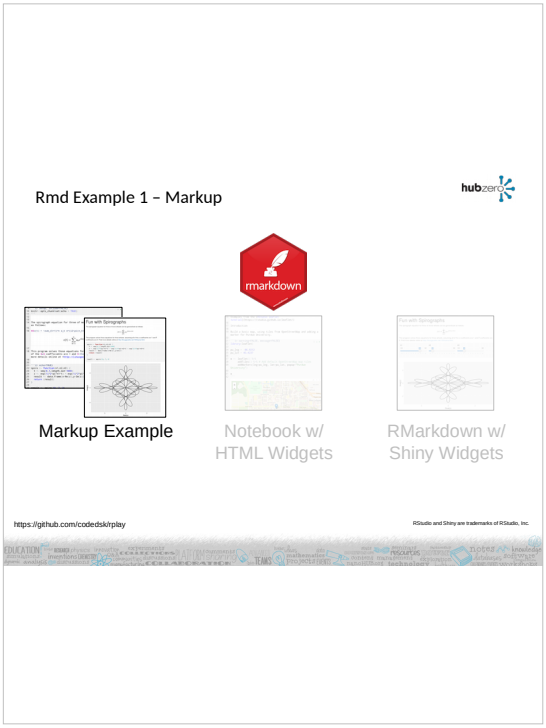


https://github.com/codedsk/rplay/mapping/indiana-precip-shiny

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.

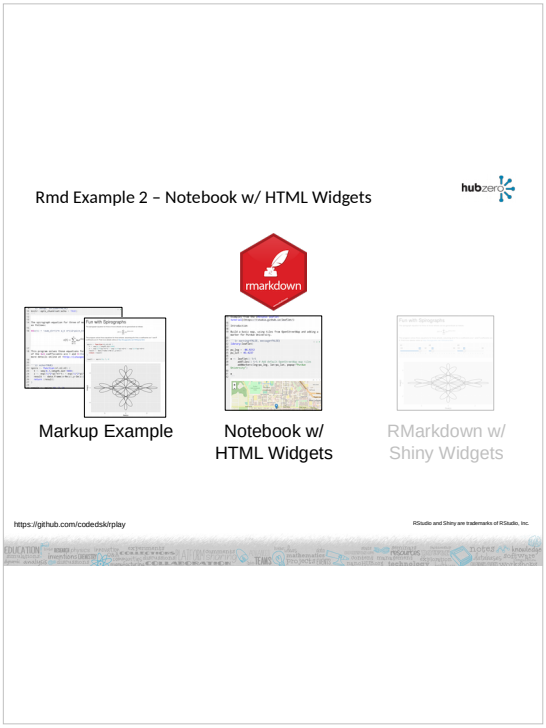
RMarkdown files can be converted to HTML files for distribution on the web or PDF files. Additionally they can be converted to Word document, LaTeX documents and a handful of other formats.

Thats one of the strengths of the system.

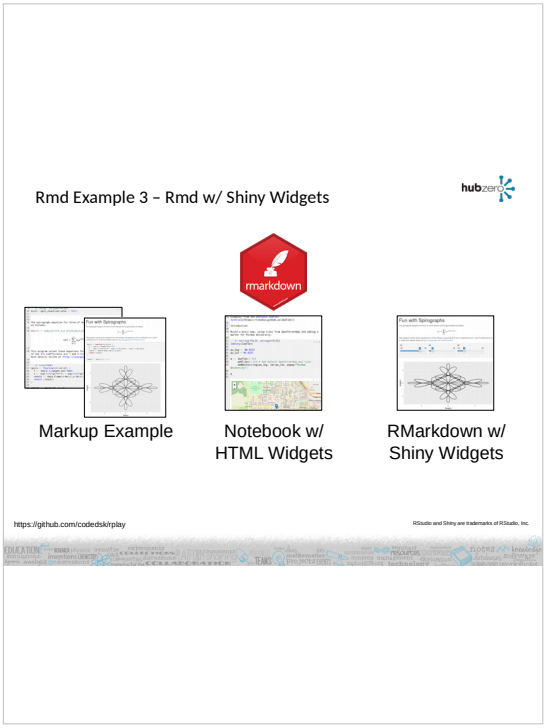


Let’s check out some examples

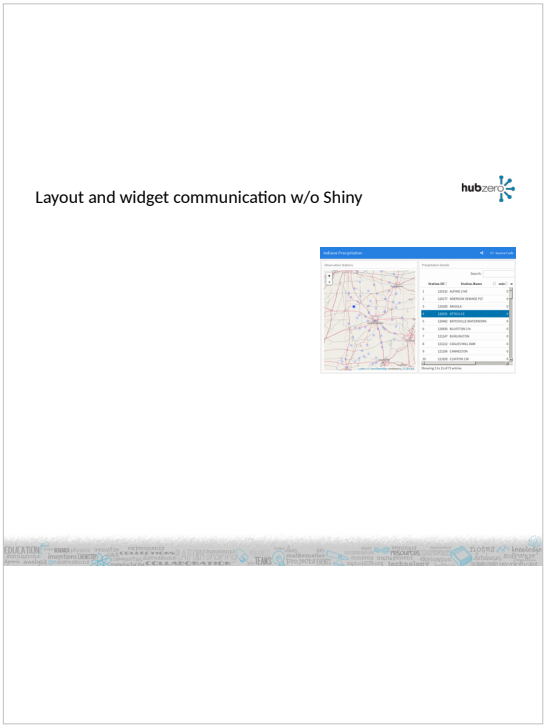
http://github.com/codedsk/rplay/rmarkdown/plain_markup.Rmd



http://github.com/codedsk/rplay/mapping/notebook_htmlwidgets_example.Rmd



http://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.