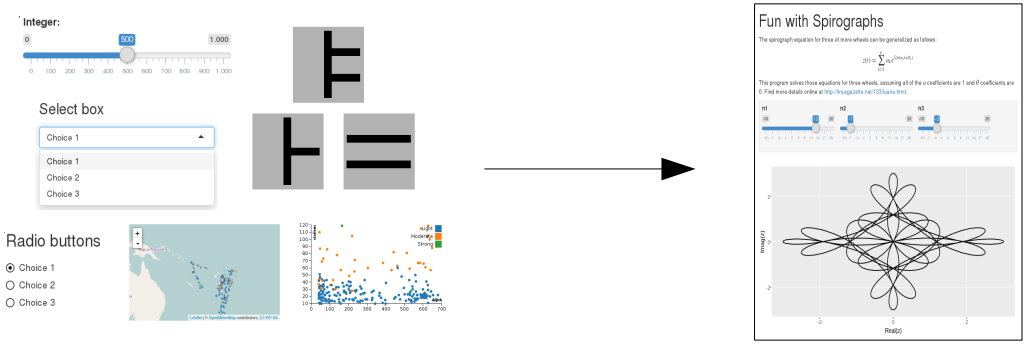
Developing with R on HUBzero

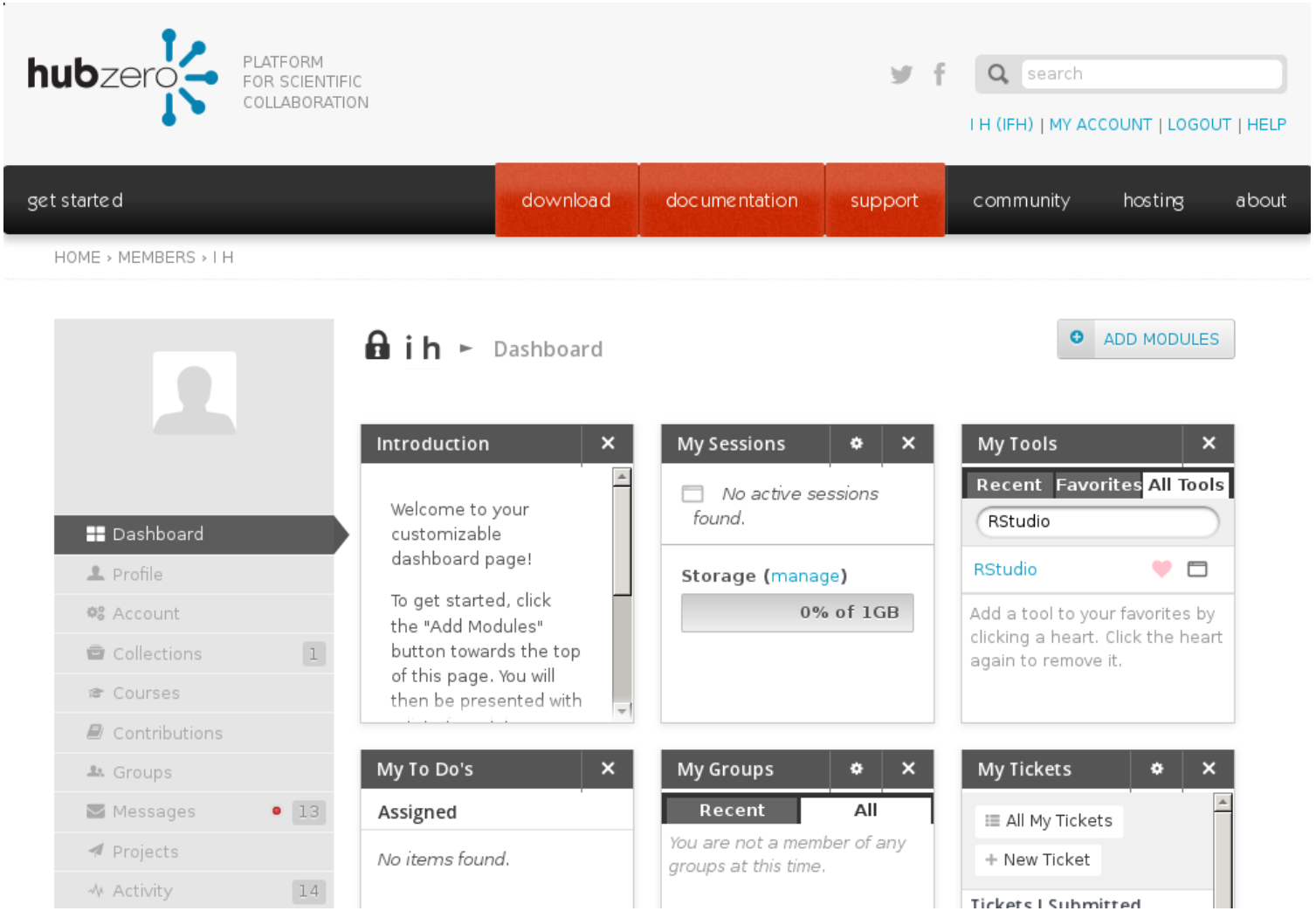


Derrick Kearney

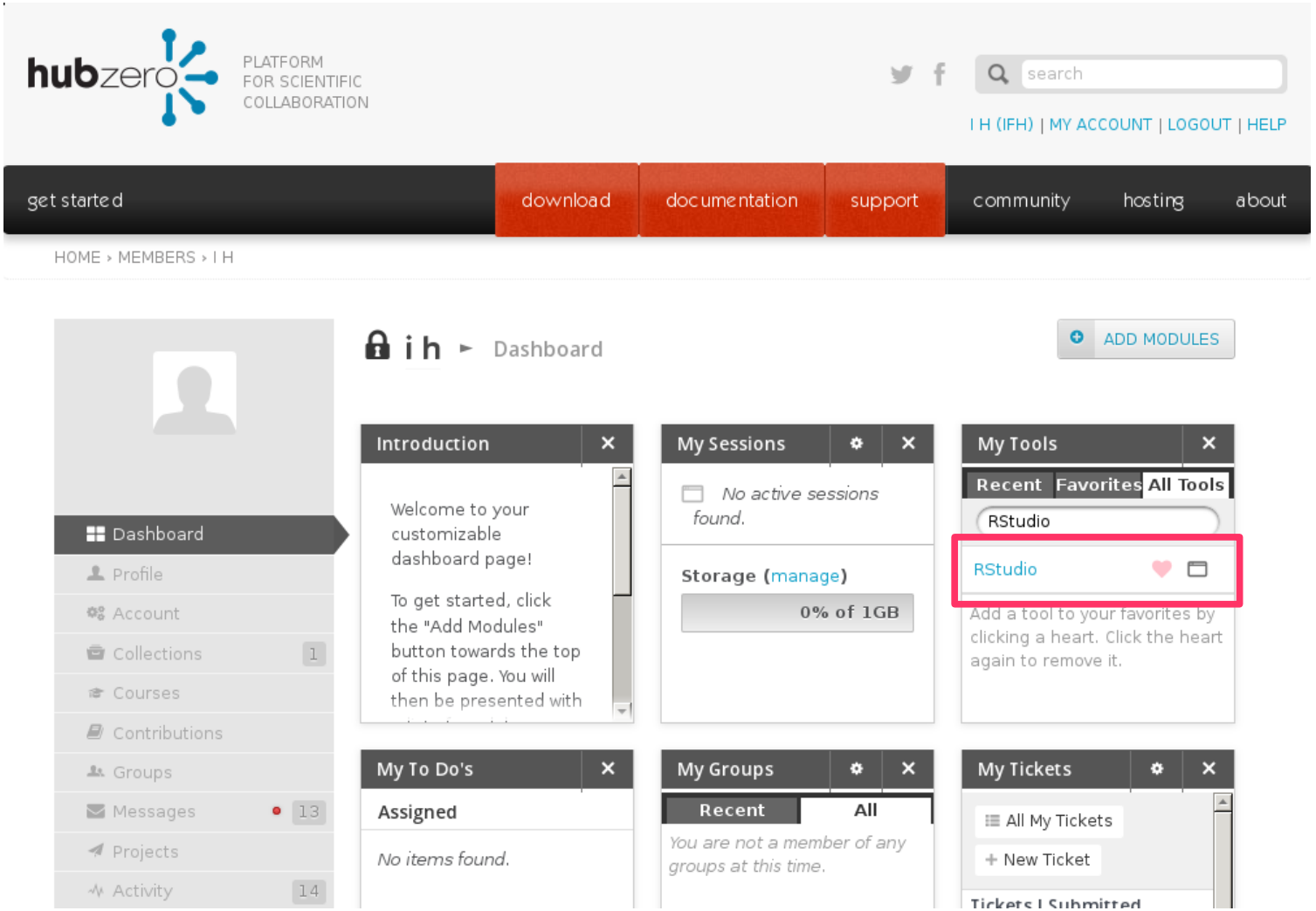
HUBzero® Platform for Scientific Collaboration

Purdue University

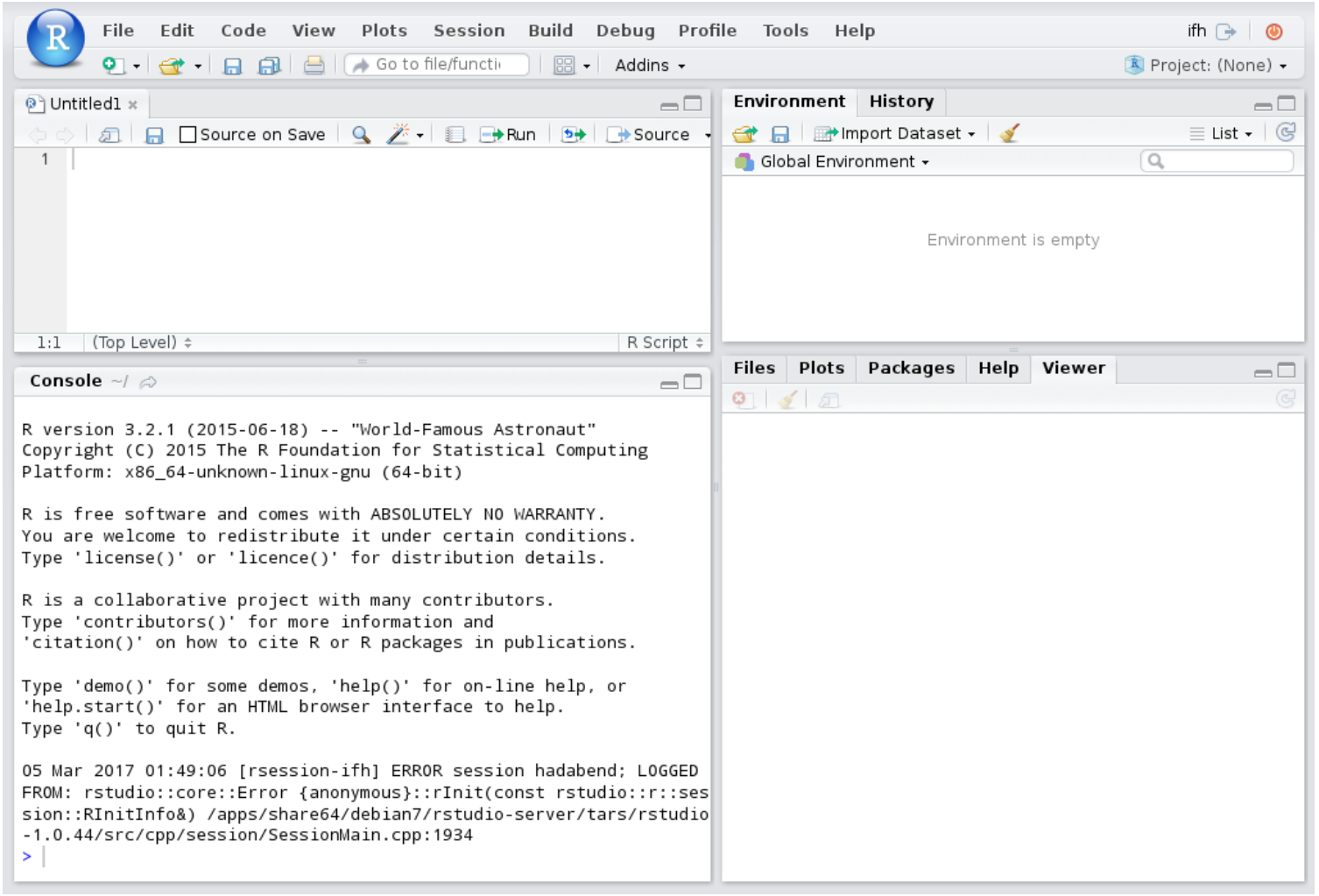
Launching RStudio From HUBzero



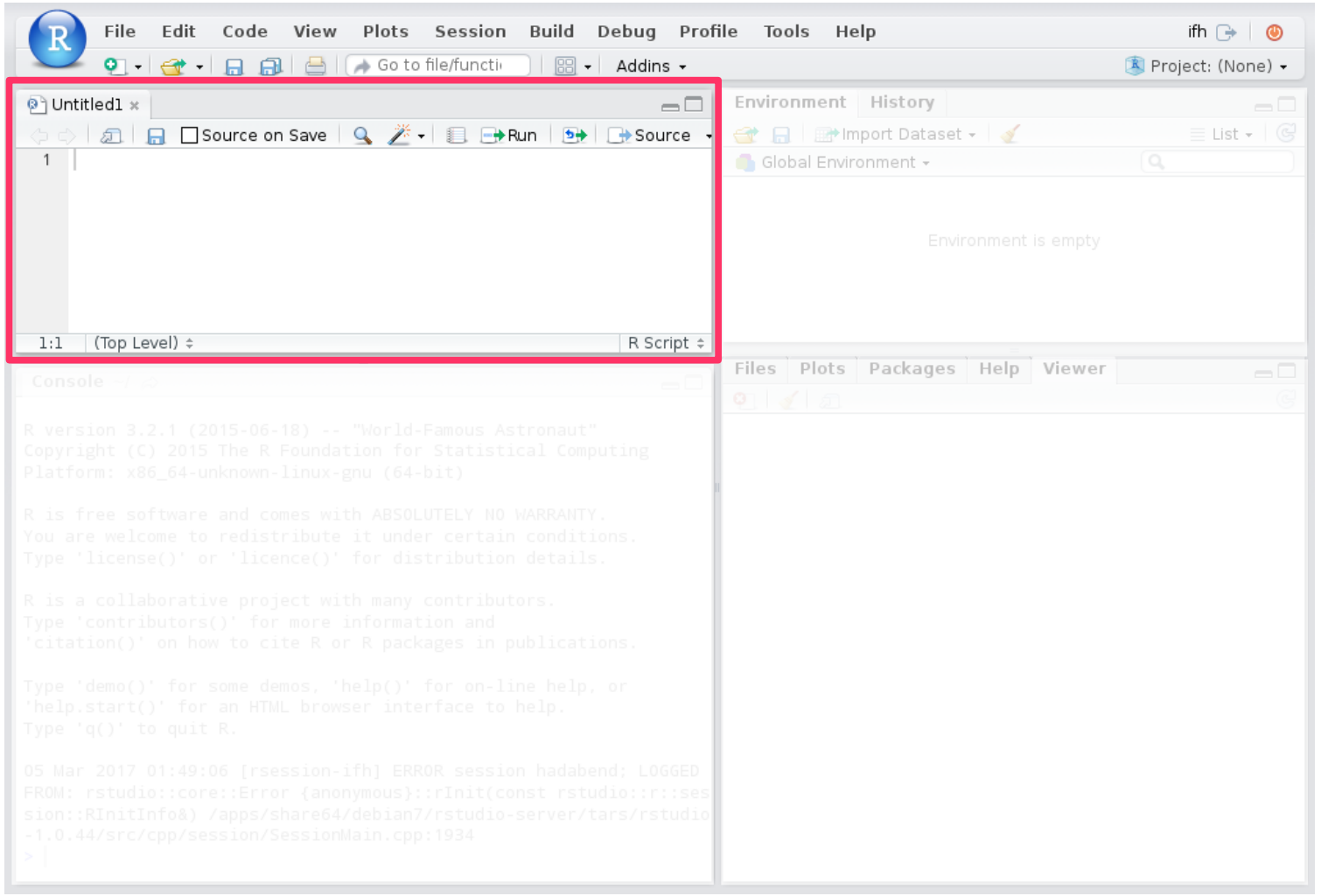
Launching RStudio From HUBzero



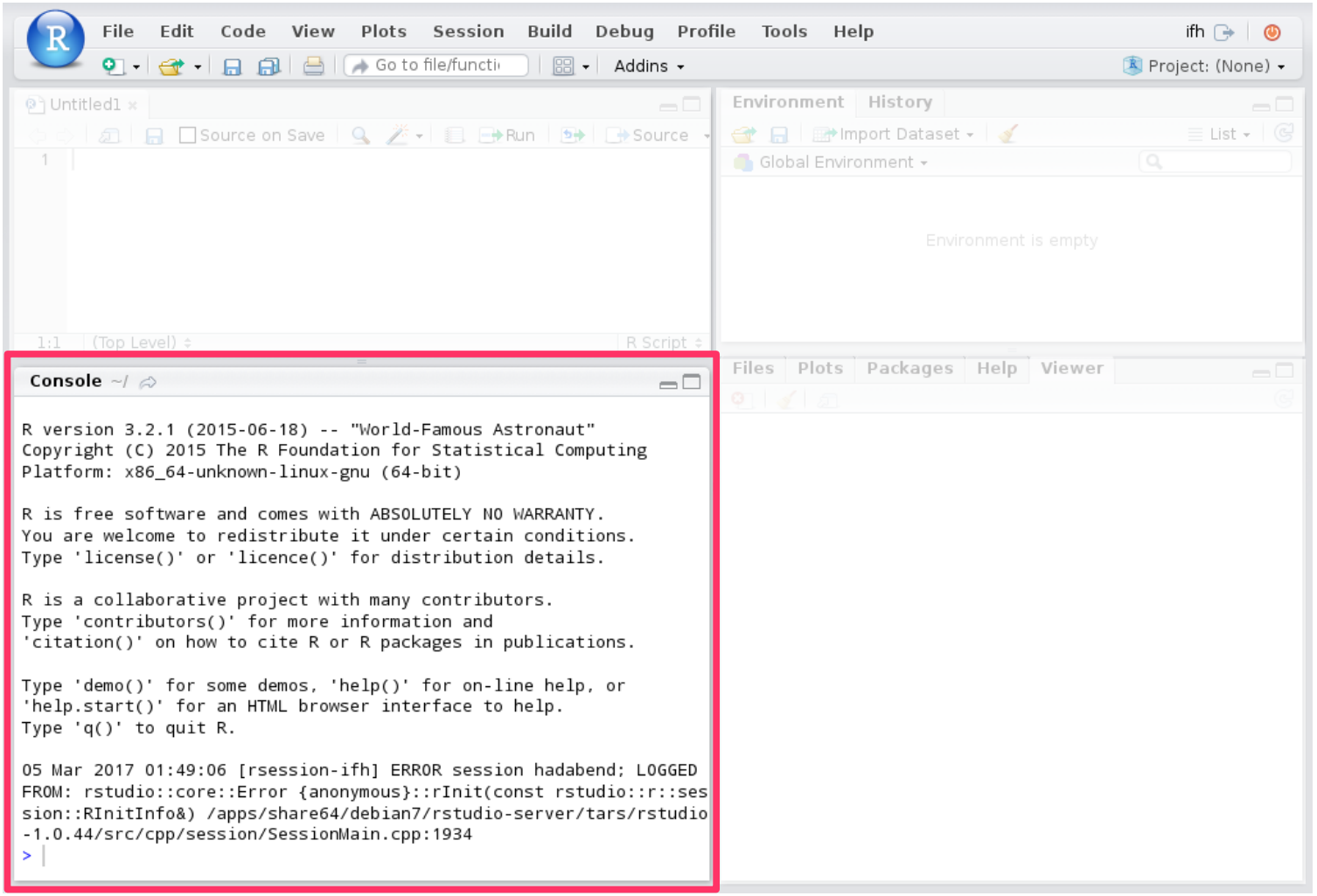
Launching RStudio From HUBzero



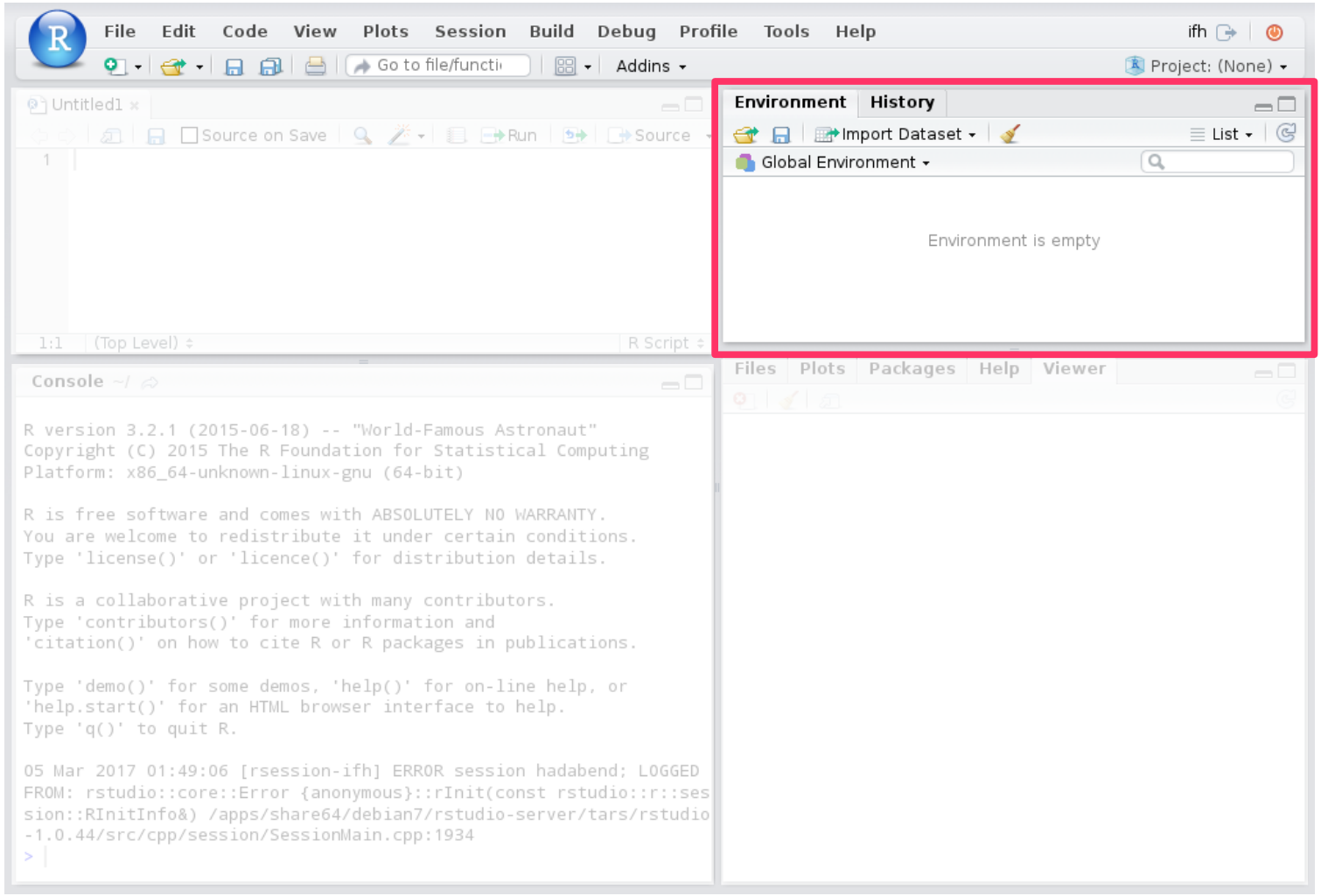
RStudio Interface Tour – Source Pane



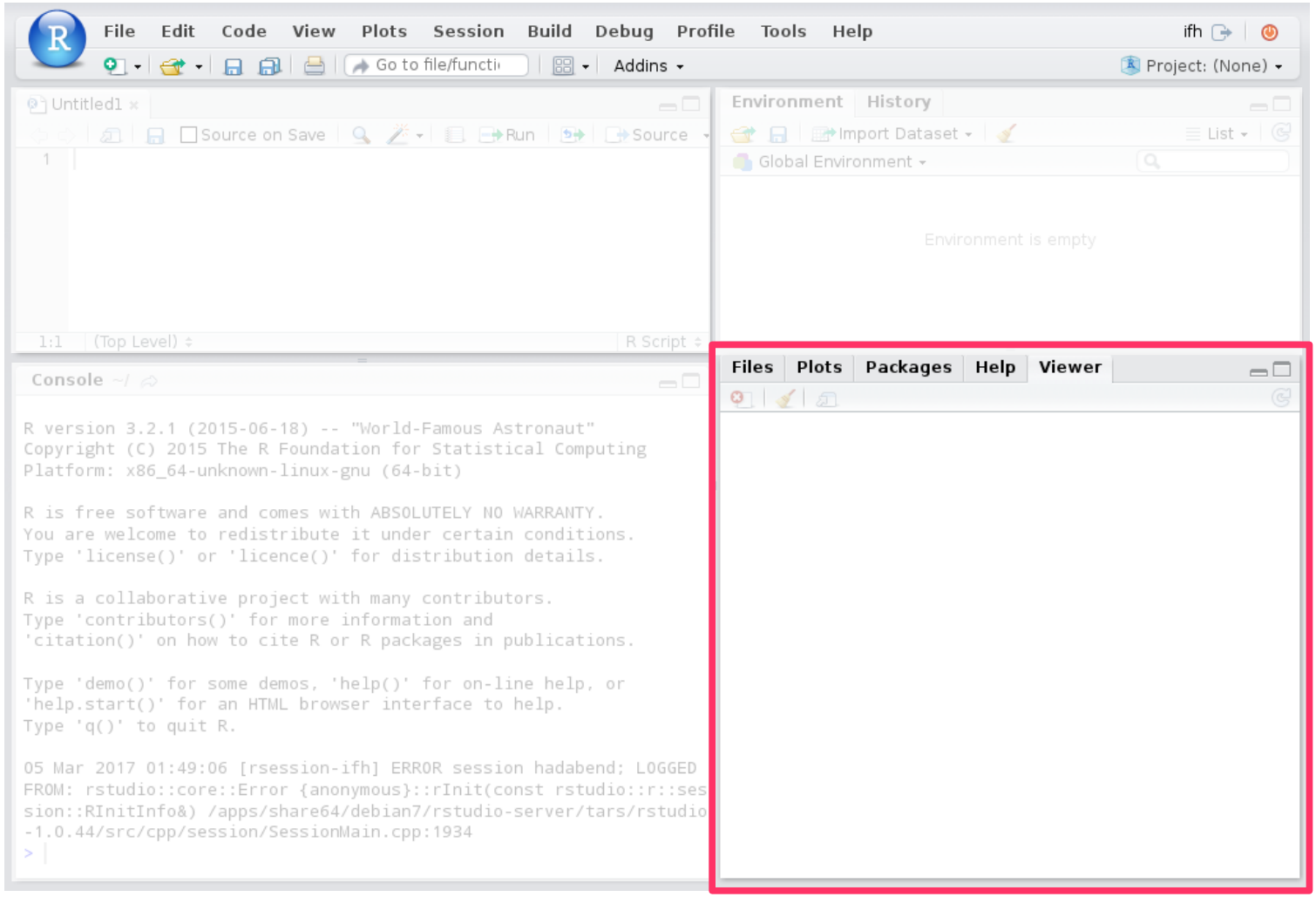
RStudio Interface Tour – Console Pane



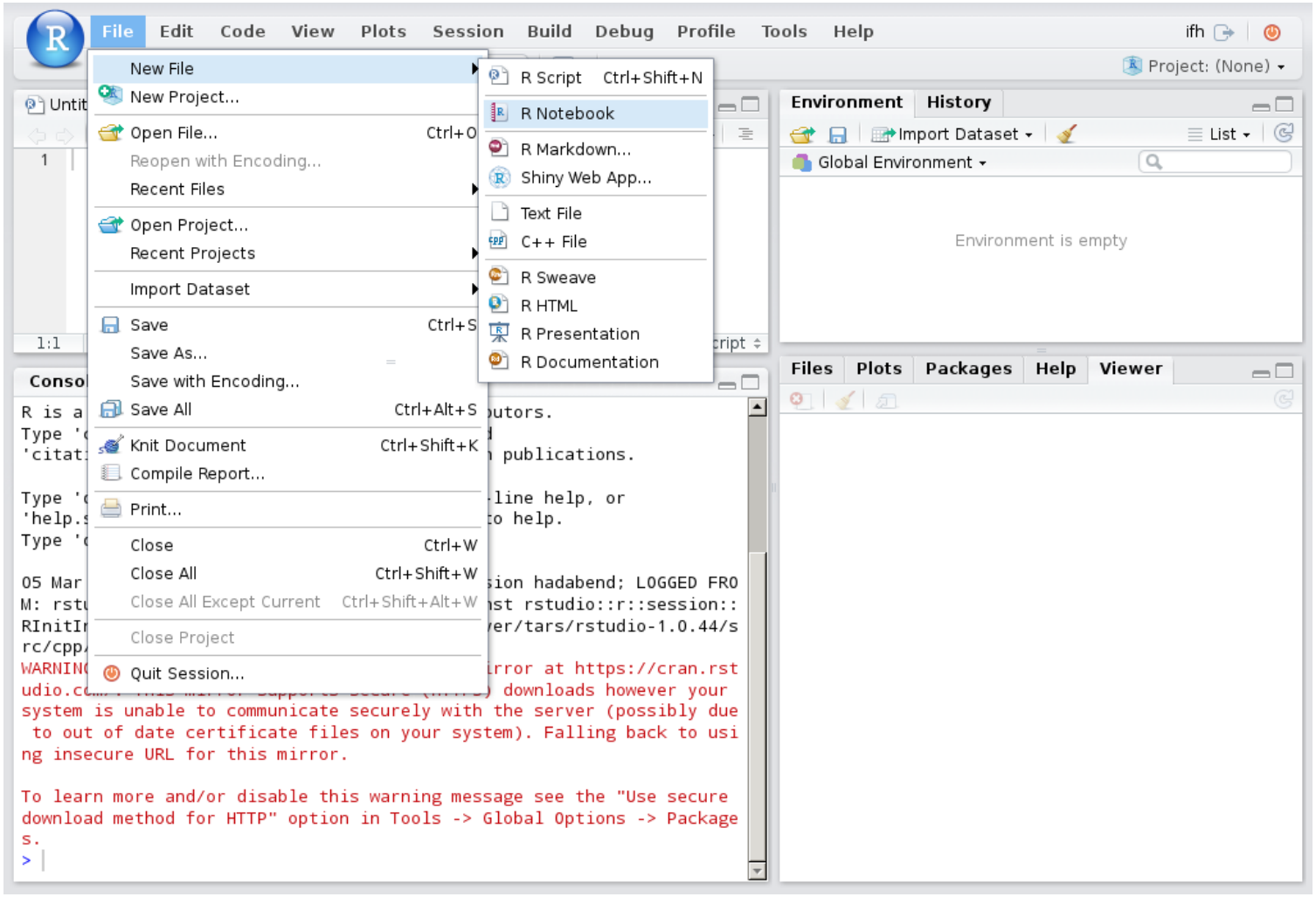
RStudio Interface Tour – Environment Pane



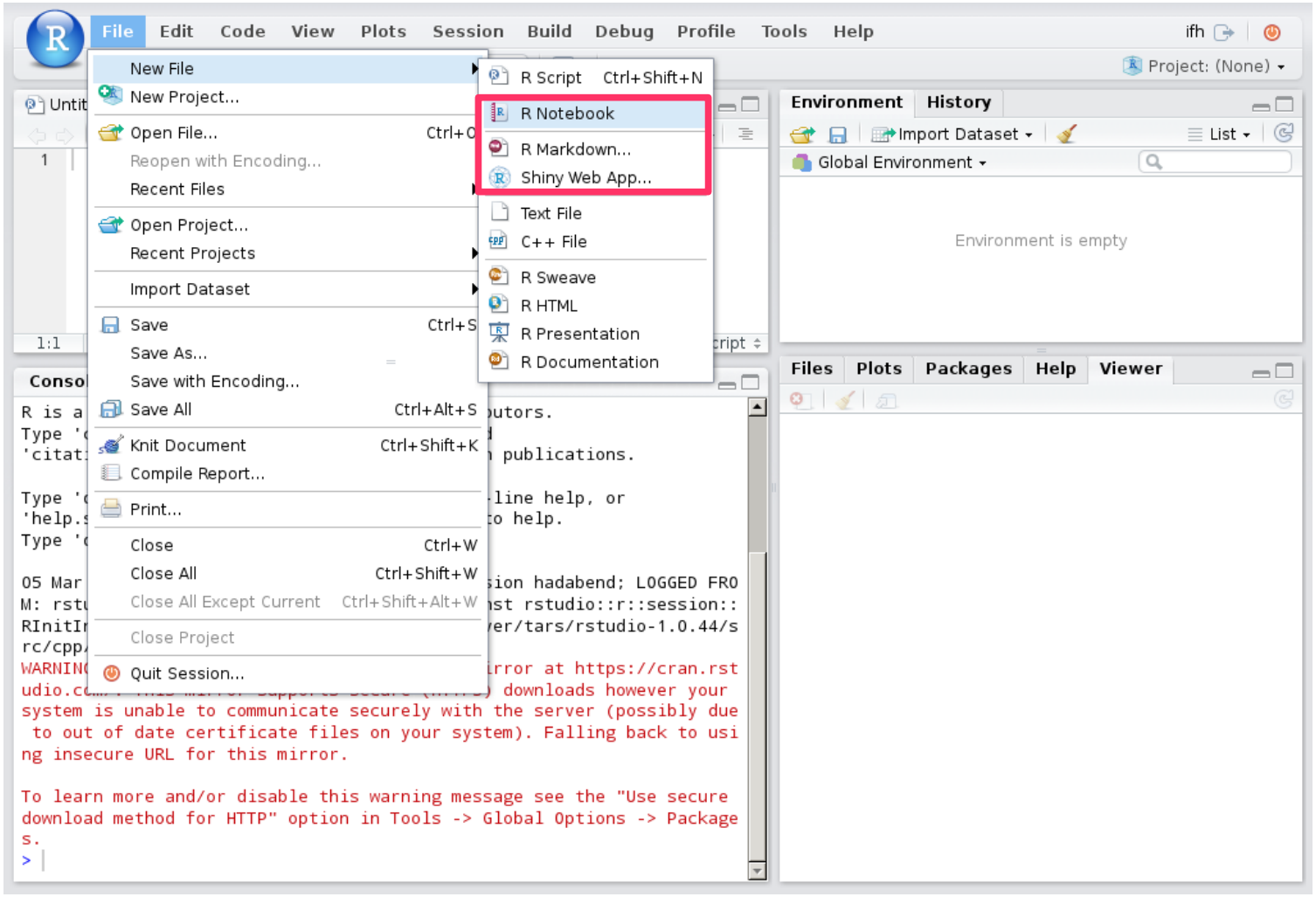
RStudio Interface Tour – Viewer Pane



Writing R Code

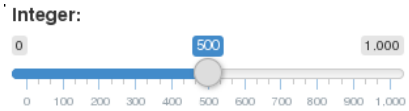


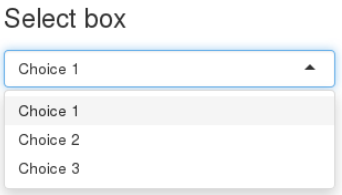
Writing R Code

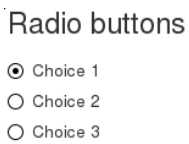


Building Applications and Interactive Reports

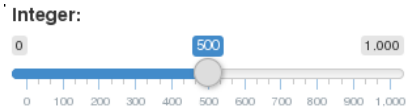
Building Applications and Interactive Reports

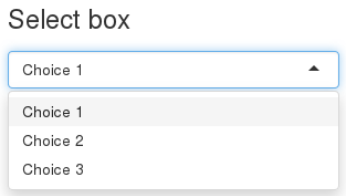


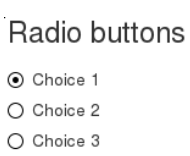


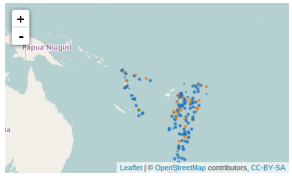


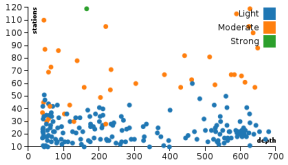
Building Applications and Interactive Reports



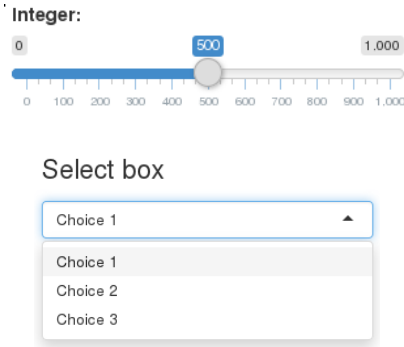


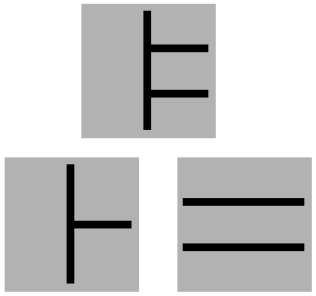


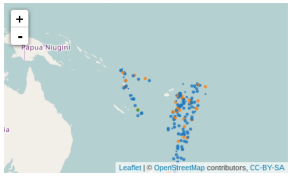


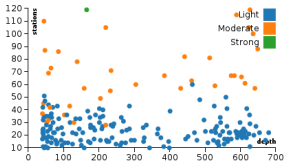


Building Applications and Interactive Reports

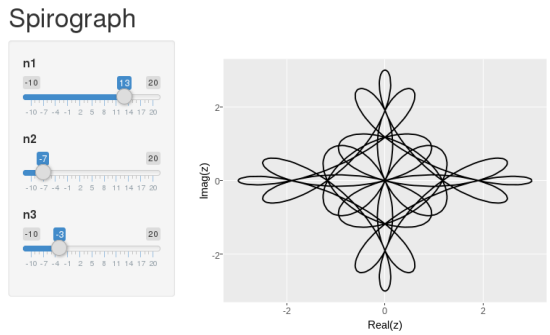


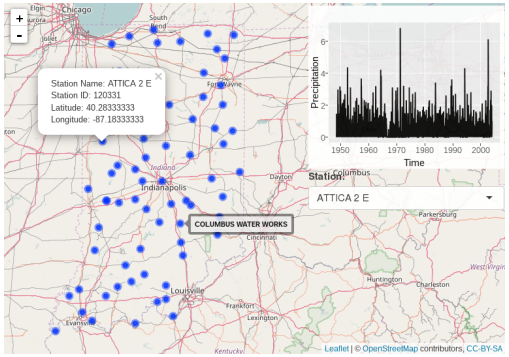






Shiny Example 1 - Spirograph





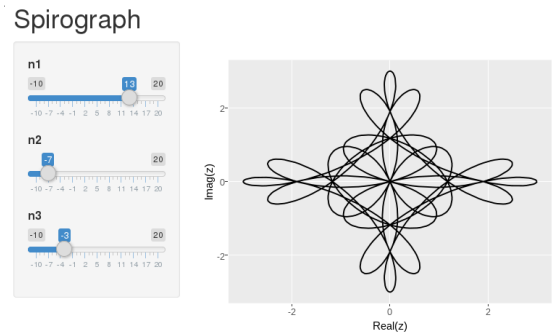
Spirograph Example

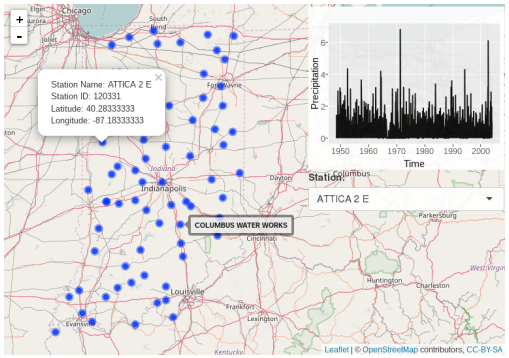
https://github.com/codedsk/hubzero-tool-spiro-shiny-1

Indiana Precip Example

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

Shiny Example 2 – Indiana Precipitation





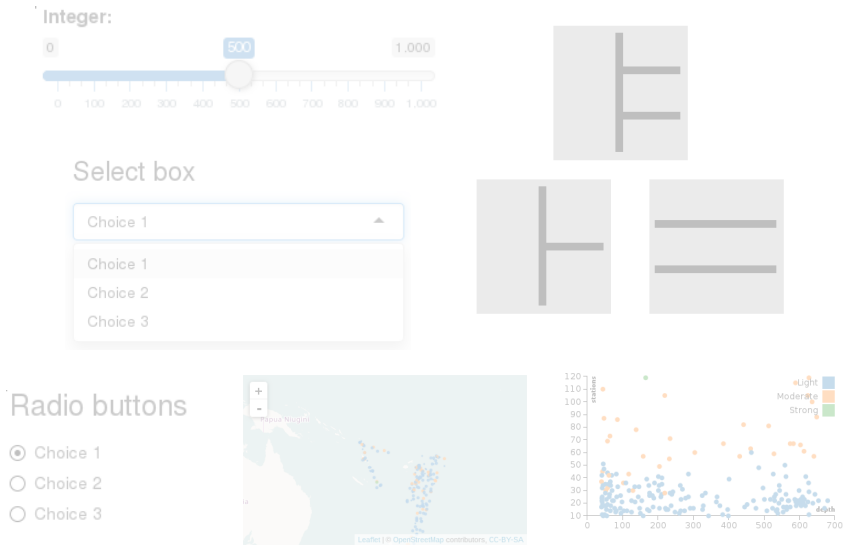
Spirograph Example

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:

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

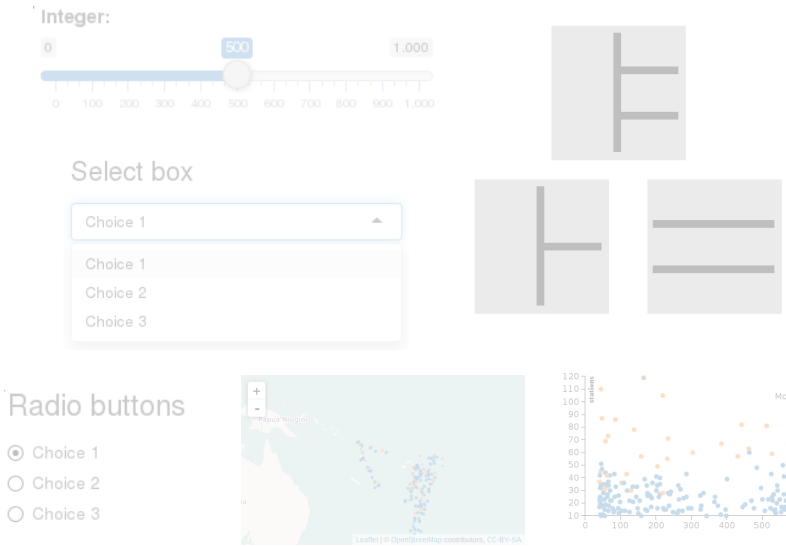
$$z(t) = \sum_{k=1}^{n} a_k e^{i2\pi(n_k t+\theta_k)}$$

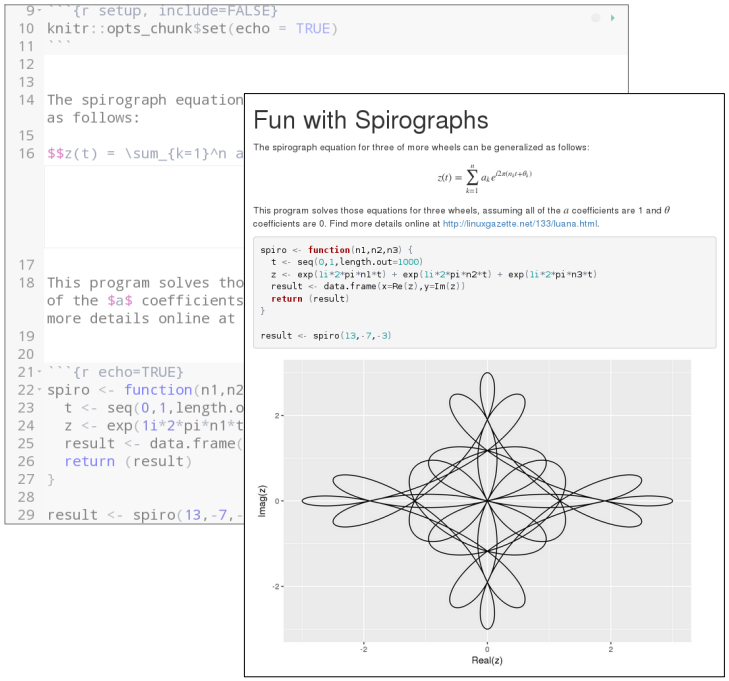
This program solves those equations for three wheels, assuming all of the a coefficients are 1 and θ coefficients are 0. Find more details online at <http://linuxgazette.net/133/luana.html>.


```{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)
```
```

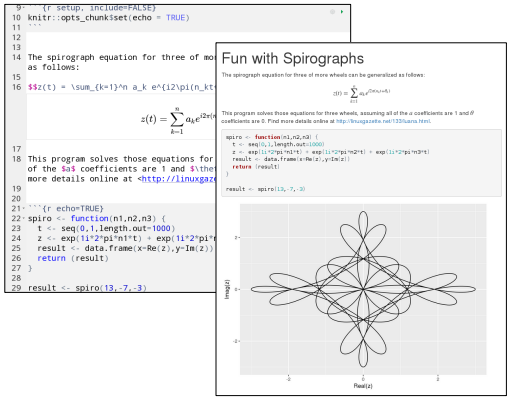
Building Applications and Interactive Reports



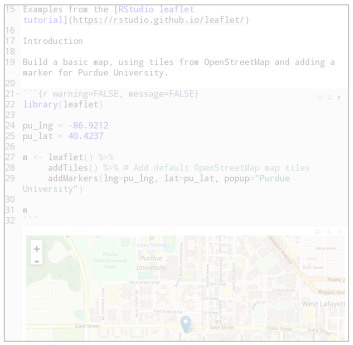


RStudio and Shiny are trademarks of RStudio, Inc.

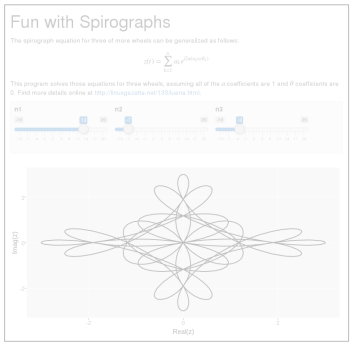
Rmd Example 1 – Markup



Markup Example



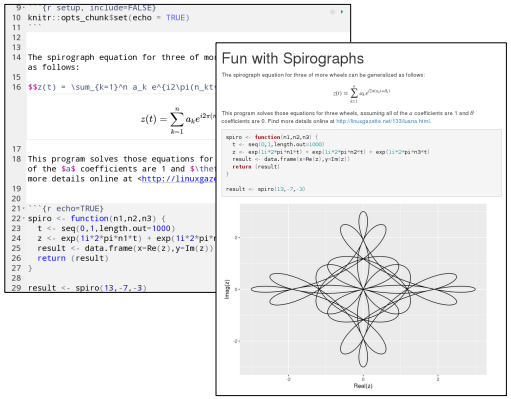
Notebook w/ HTML Widgets



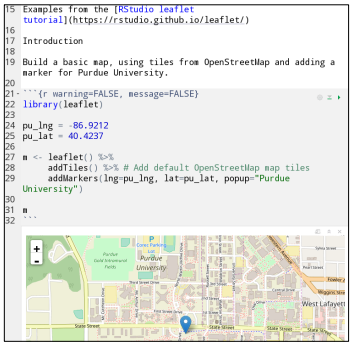
RMarkdown w/ Shiny Widgets

https://github.com/codedsk/rplay

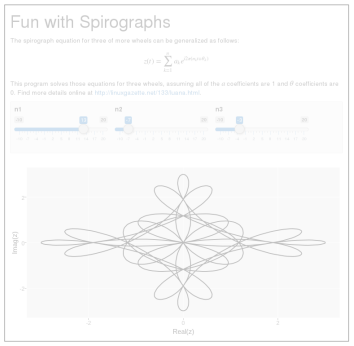
Rmd Example 2 – Notebook w/ HTML Widgets



Markup Example



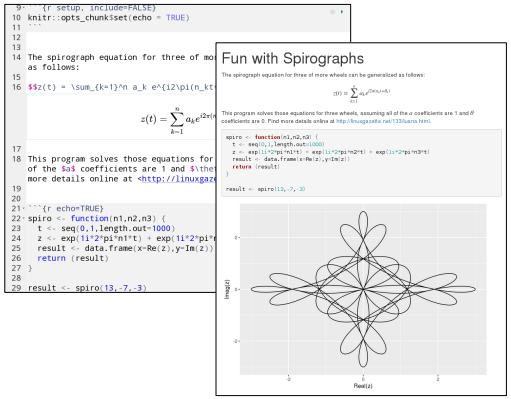
Notebook w/ HTML Widgets



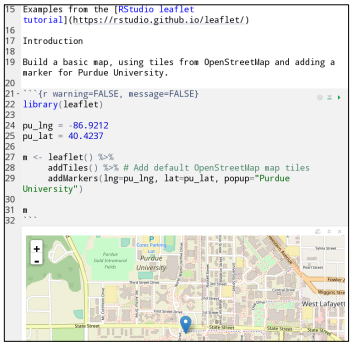
RMarkdown w/ Shiny Widgets

https://github.com/codedsk/rplay

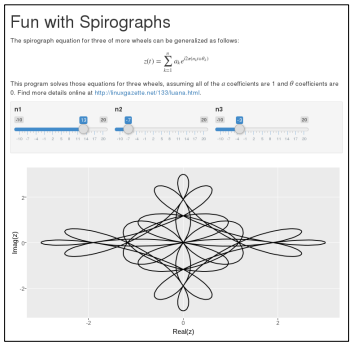
Rmd Example 3 – Rmd w/ Shiny Widgets



Markup Example



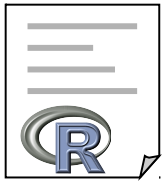
Notebook w/ HTML Widgets



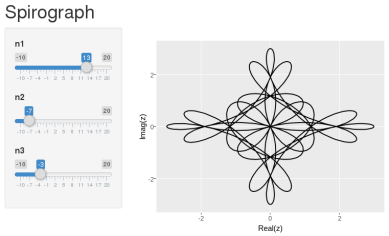
RMarkdown w/ Shiny Widgets

https://github.com/codedsk/rplay

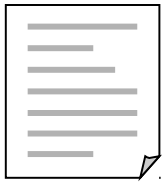
Publishing on the HUB



Source Code



Graphical User Interface

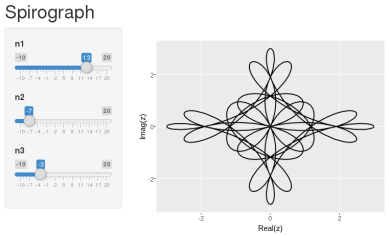


Makefile

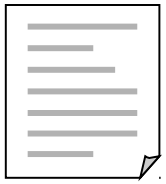
Invoke Script

Publishing on the HUB

Source Code

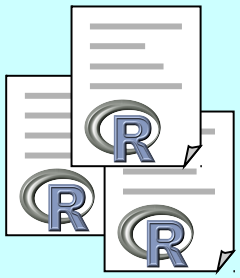


Graphical User Interface

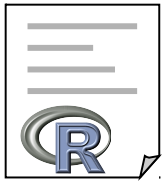


Makefile

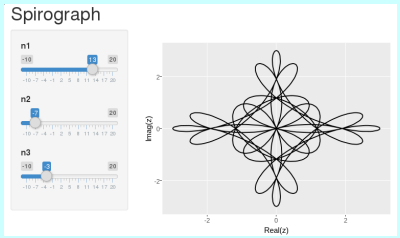
Invoke Script



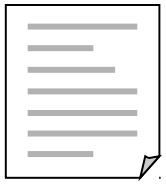
Publishing on the HUB



Source Code

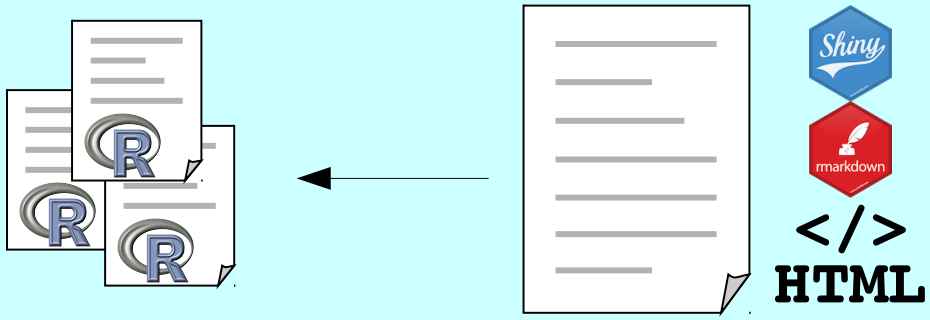


Graphical User Interface

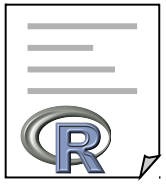


Makefile

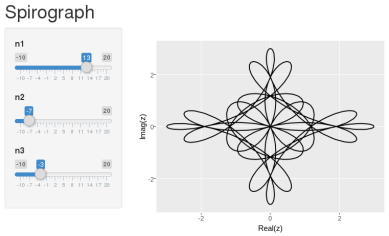
Invoke Script



Publishing on the HUB



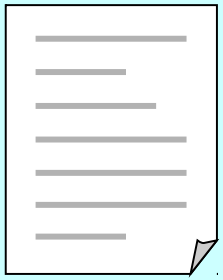
Source Code



Graphical User Interface

Makefile

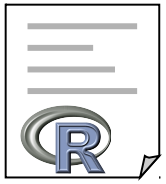
Invoke Script



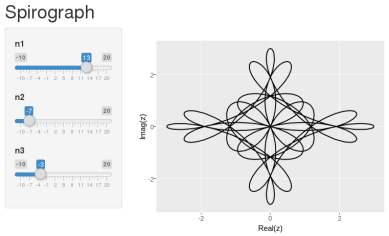
```
all:
    @. /etc/environ.sh; \
        use -e -r R-3.2.1; \
        use -e -r rstudio-server-1.0.44; \
        Rscript -e "rmarkdown::render('${RMDFILE}')"

install:
    install --mode 0444 -D ${HTMLFILE} ${WWWDIR}/${HTMLFILE}
    ln -s ${HTMLFILE} ${WWWDIR}/index.html
```

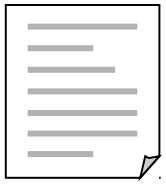
Publishing on the HUB



Source Code

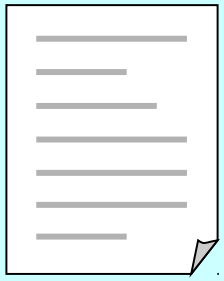


Graphical User Interface



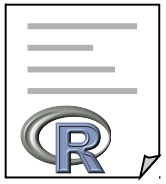
Makefile

Invoke Script

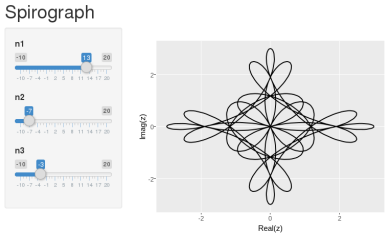


```
/usr/bin/invoke_app "$@" -t spirormddoc \
    -u R-3.2.5 \
    -u rstudio-server-1.0.44 \
    -u wrwroxy-0.1 \
    -u wrapprun-0.1 \
    -c "wrwroxy --listenPort 8000 --forwardPort 8001 --stream-log --logfile /dev/null" \
    -C "wrapprun --host 0.0.0.0 --port 8001 @tool/www/index.html"
```

Publishing on the HUB



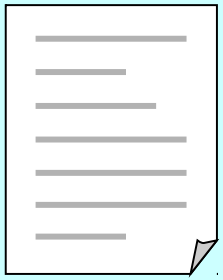
Source Code



Graphical User Interface

Makefile

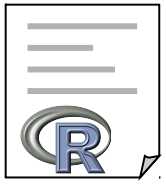
Invoke Script



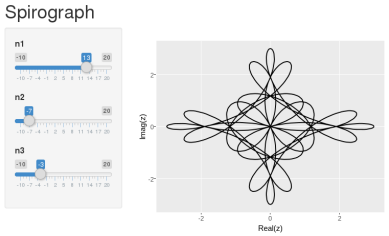
```
all:

install:
    install --mode 0444 -D ${RMDFILE} ${BINDIR}/${RMDFILE}
```

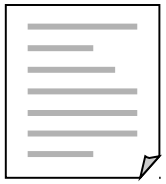
Publishing on the HUB



Source Code

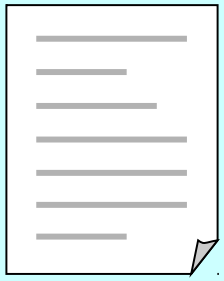


Graphical User Interface



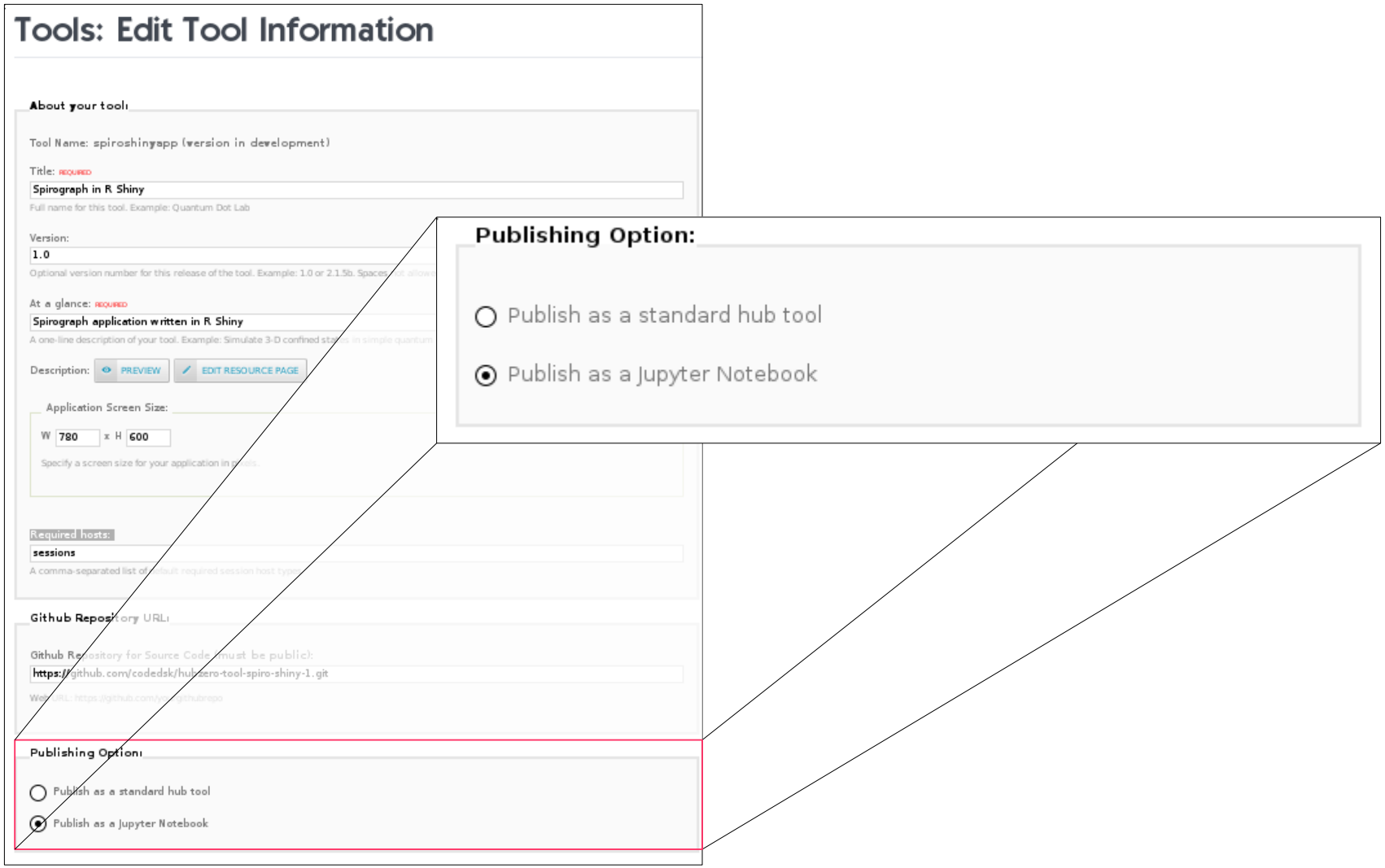
Makefile

Invoke Script



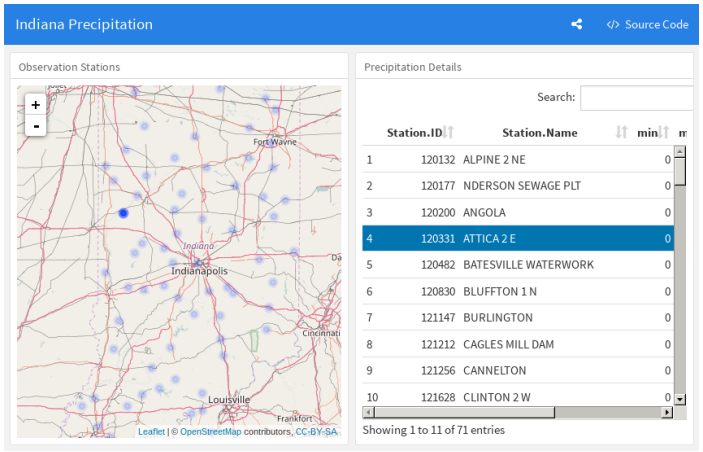
```
/usr/bin/invoke_app "$@" -t spirormddoc \
    -u R-3.2.5 \
    -u rstudio-server-1.0.44 \
    -u wrwroxy-0.1 \
    -u wrapprun-0.1 \
    -c "wrwroxy --listenPort 8000 --forwardPort 8001 --stream-log --logfile /dev/null" \
    -C "wrapprun --host 0.0.0.0 --port 8001 @tool/bin/spirograph.Rmd"
```

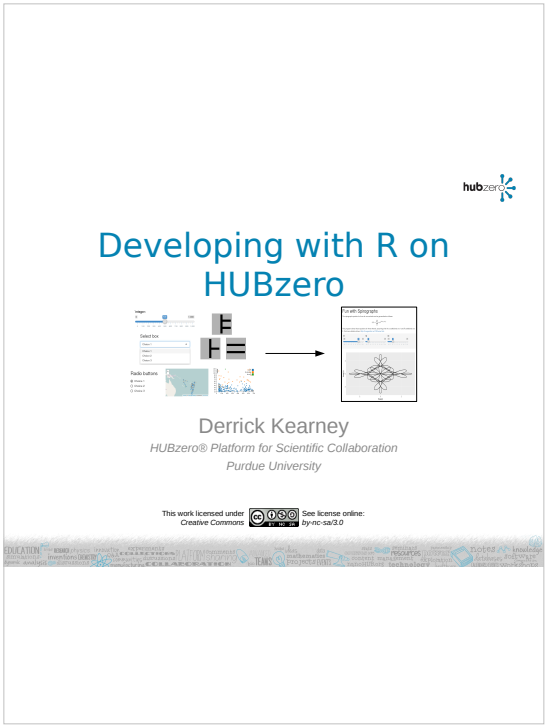
Contribtool: View as Notebook




EDUCATION
simulations
dynamic
analysis
hubs
RESEARCH
physics
inventions
CHEMISTRY
discussions
innovation
Q&A
communities
manufacturing
experiments
COLLECTIONS
discussions
COLLABORATION
PLATFORM
comments
sharing
ADVANCED
bio
TEAMS
hubs
ideas
mathematics
projects
data
EVENTS
stats
courses
seminars
resources
content
management
nanoHUB.org
Partnership
power
exploration
technology
notes
knowledge
databases
software
computation
CALENDARS/EVENTS
workshops

Layout and widget communication w/o Shiny

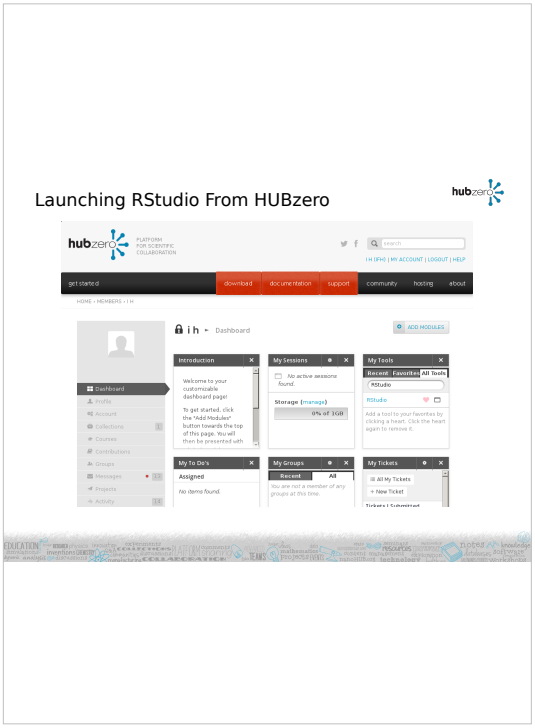




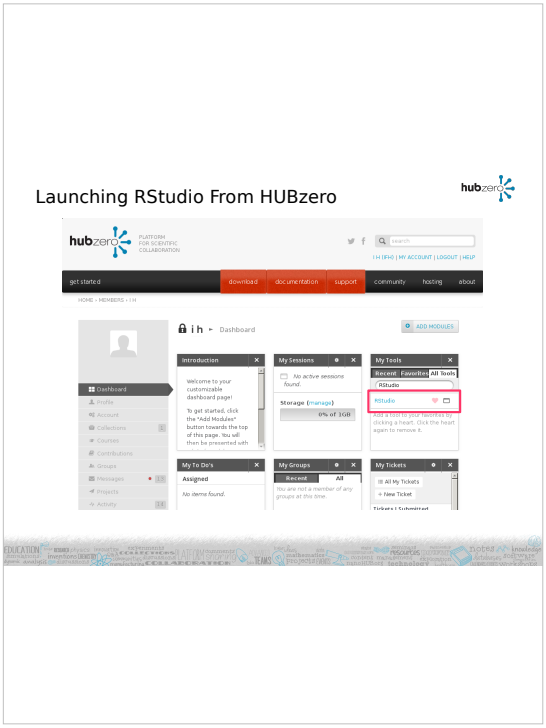
Literate computing has a long history with contributions to the idea reaching from its creator Donald Knuth, to Mathematica, and Matlab. More recently we've seen the introduction of Jupyter Notebooks.

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.

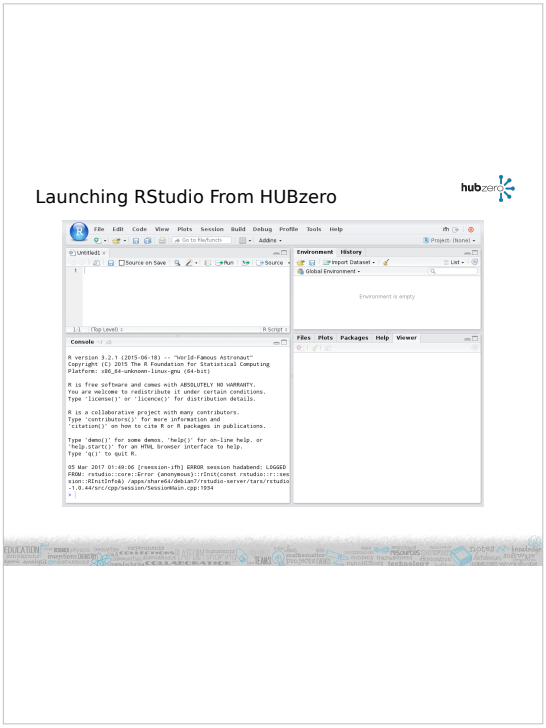


The RstudioIDE is available on the HUB



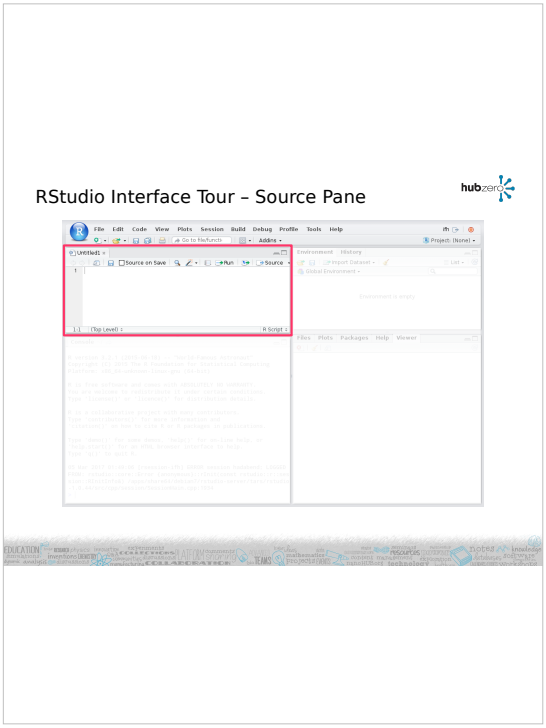
You can search for it under the My Tools module

When you click the launch tool button,



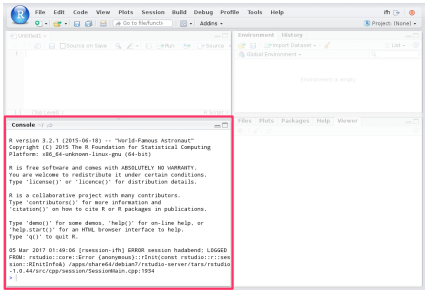
You'll notice this doesn't launch the normal vnc viewer. Instead, the user is served a javascript based user interface.

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

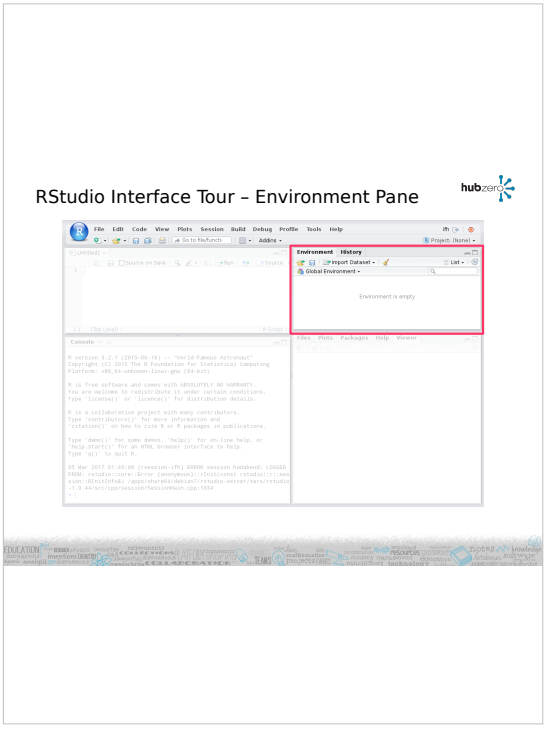


The Code Pane is where users manipulate source code

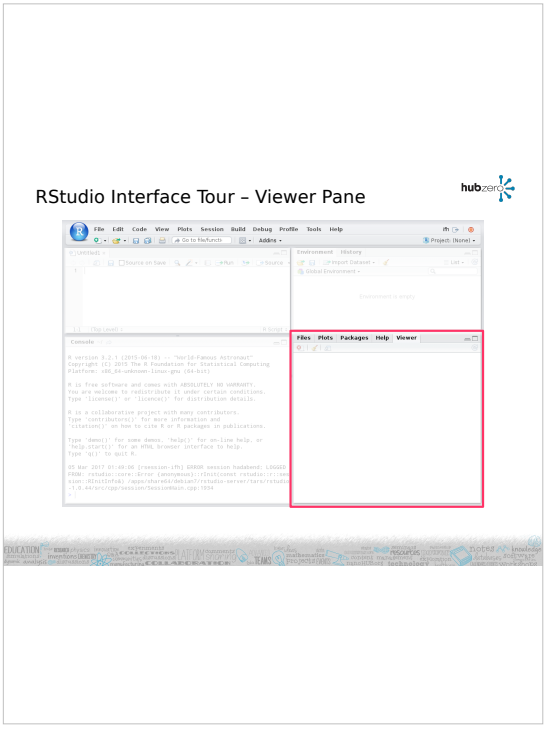
RStudio Interface Tour – Console Pane



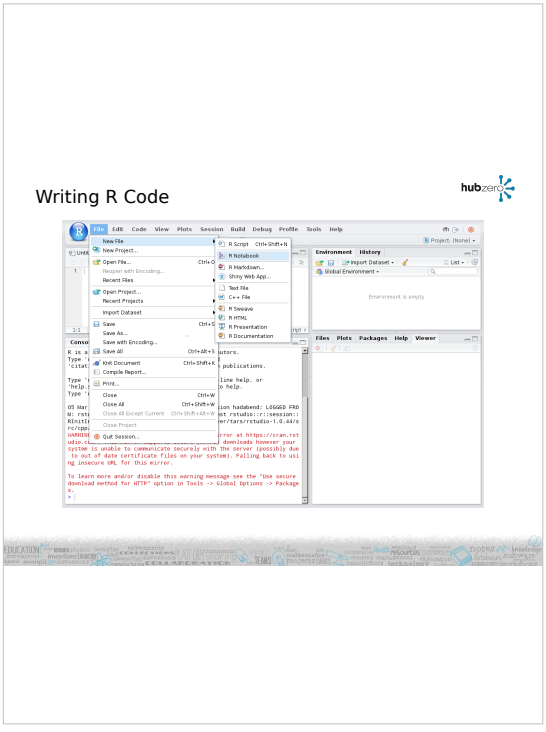
The console is where R commands are executed



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



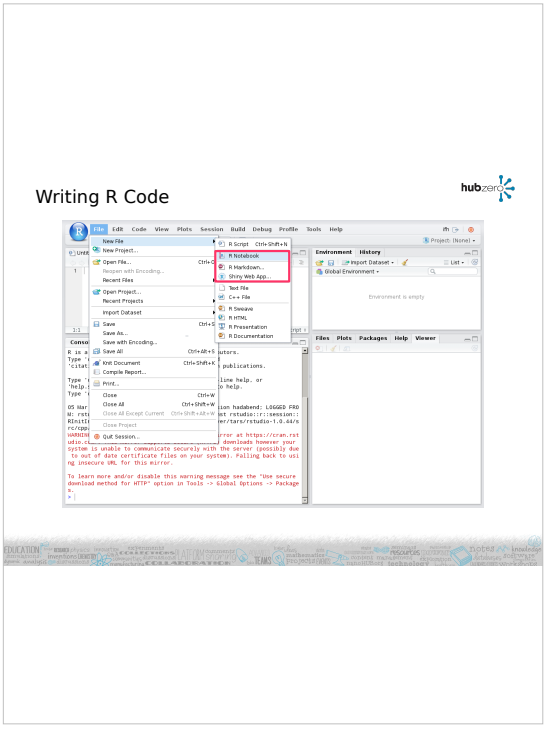
And the viewer pane shows output that can be visualized



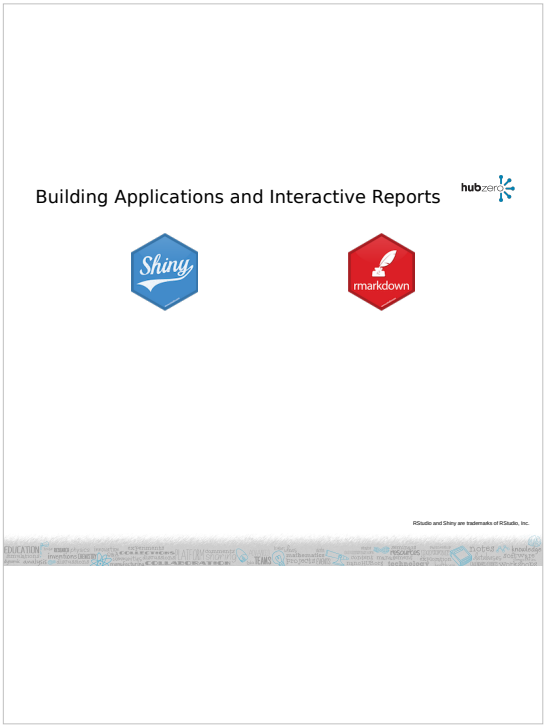
To write code, use the File → New menu item to open a new file.

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.

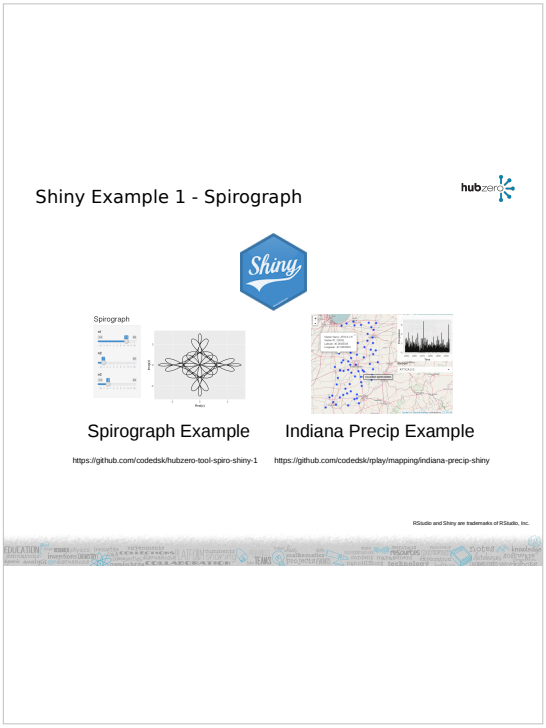
And output widgets like text boxes and static plots.

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

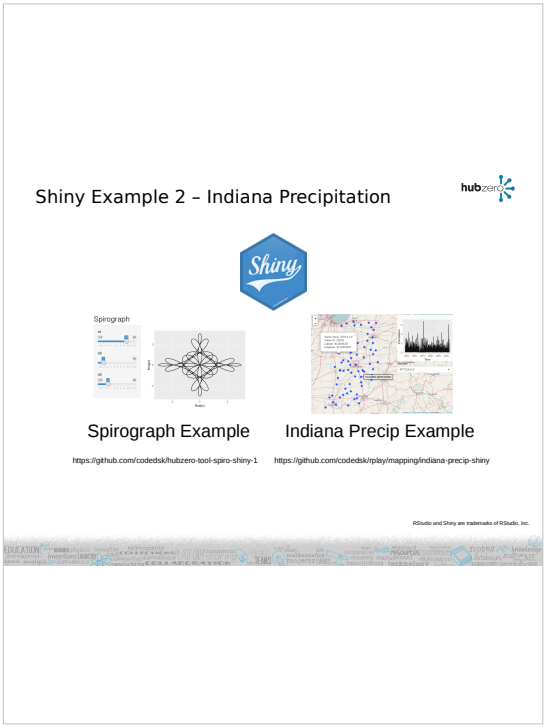
And it allows users to either fully describe their layout using a grid system or one of their predefined templates.

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.

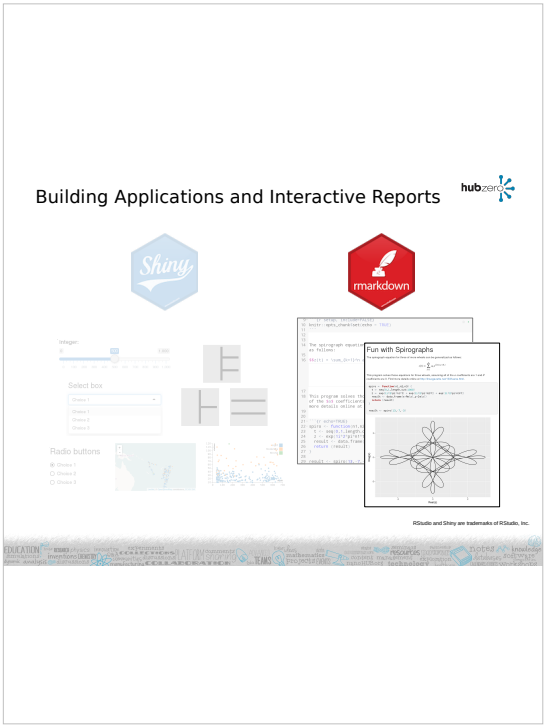


https://github.com/codedsk/hubzero-tool-spiro-shiny-1



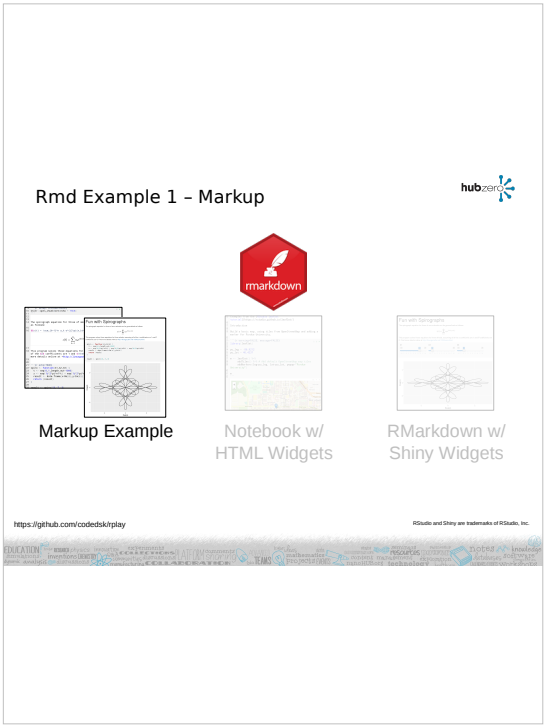
https://github.com/codedsk/rplay/mapping/indiana-precip-shiny

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.



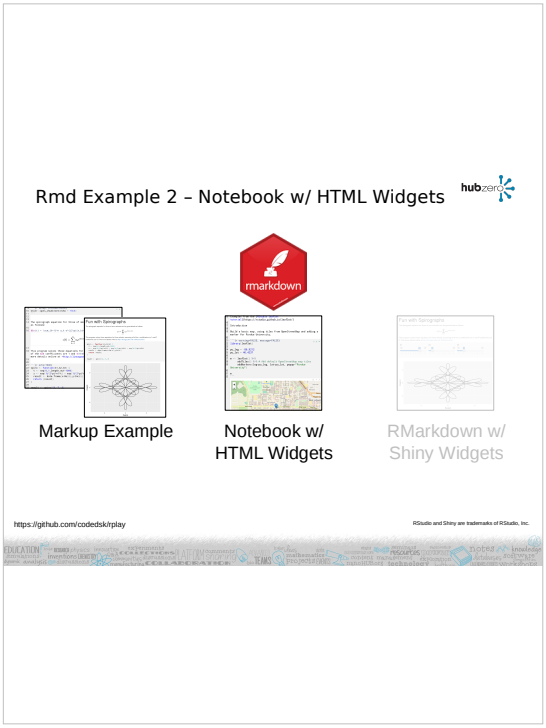
RMarkdown files can be converted to HTML files for distribution on the web or PDF files. Additionally they can be converted to Word document, LaTeX documents and a handful of other formats.

Thats one of the strengths of the system.

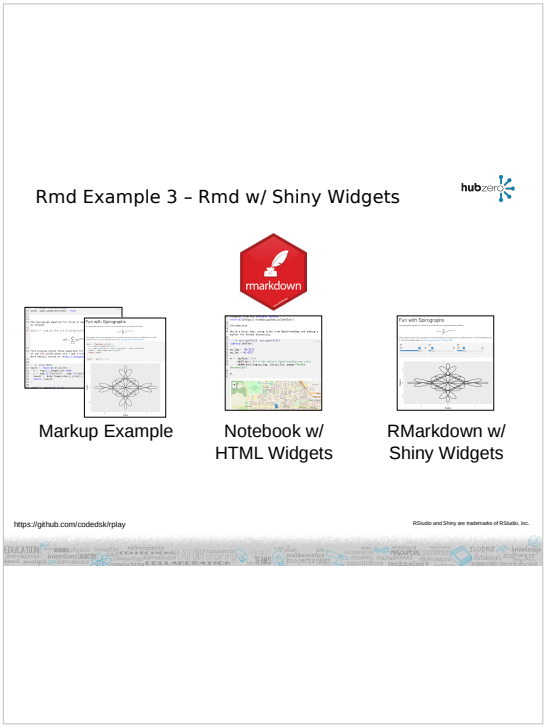


Let’s check out some examples

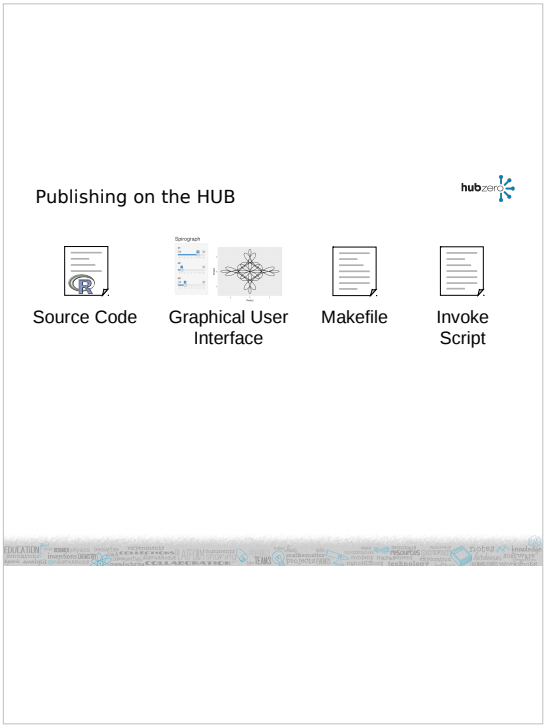
http://github.com/codedsk/rplay/rmarkdown/plain_markup.Rmd



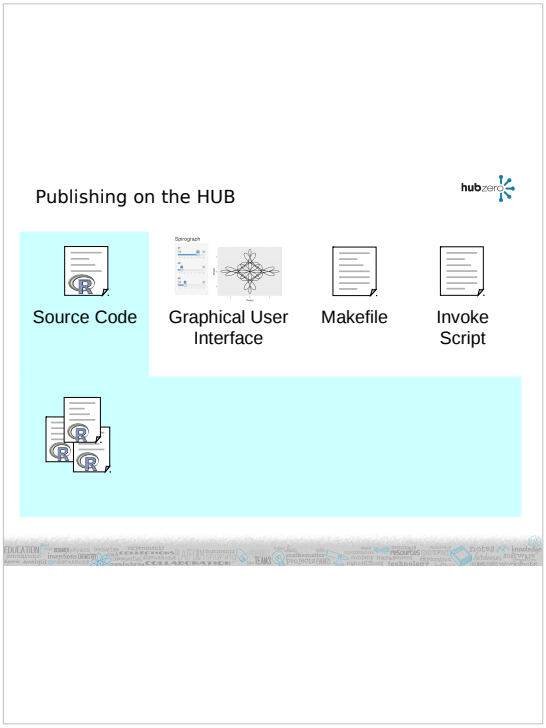
http://github.com/codedsk/rplay/mapping/notebook_htmlwidgets_example.Rmd



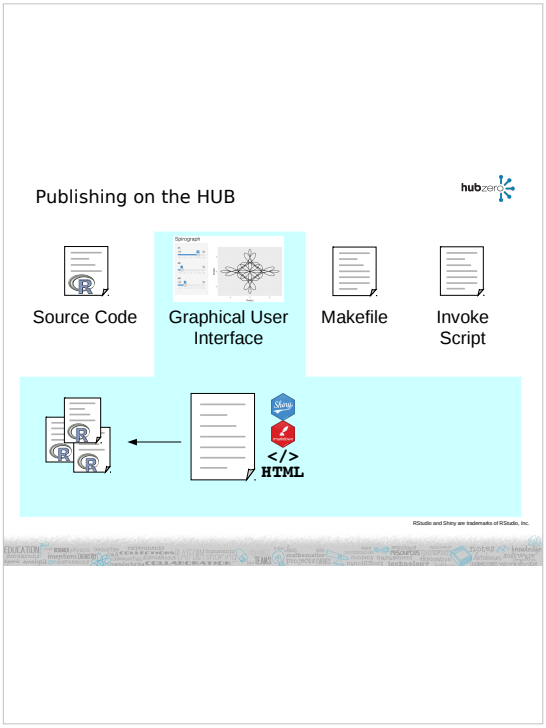
http://github.com/codedsk/hubzero-tool-spiro-rmd-html-rshiny-widgets/src/spirograph_with_shiny_widgets.Rmd.



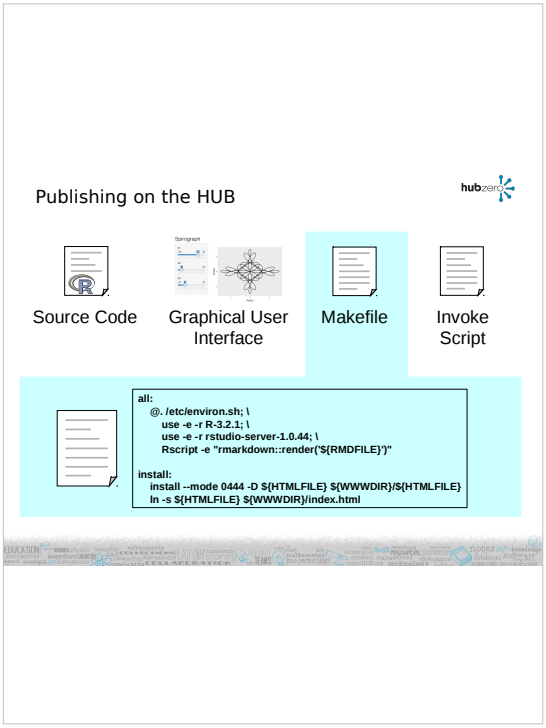
If you are familiar with the tool contribution process on the hub, you know we require 4 things to get your application published. Publishing Shiny and RMarkdown applications is no different.



We need to have the source code in a source code repository

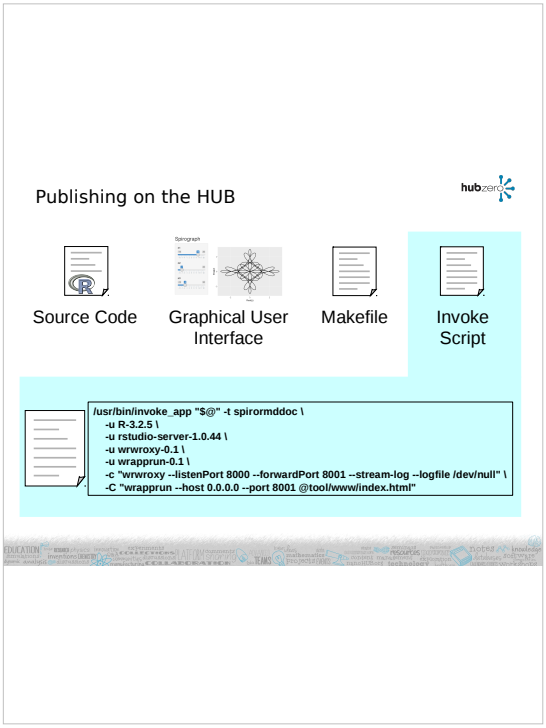


There needs to be a graphical user interface. Since these are web applications and web pages, they will have a HTML and JavaScript based user interface.

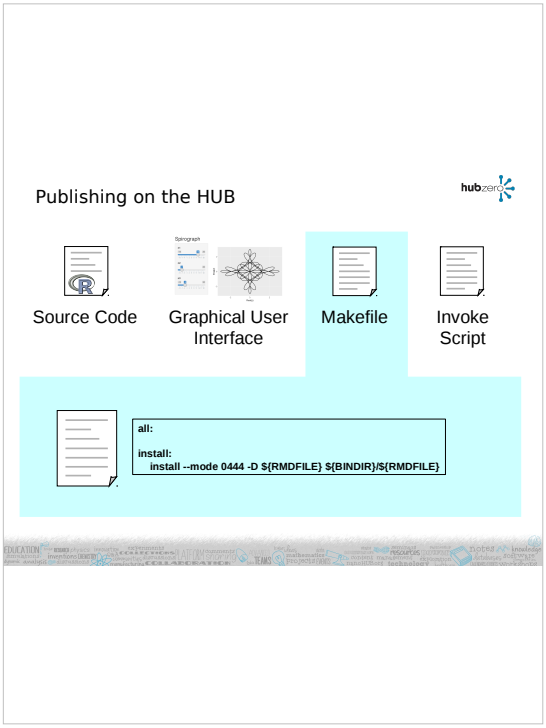


It's always a good idea to include a Makefile that describes how to compile and install the application.

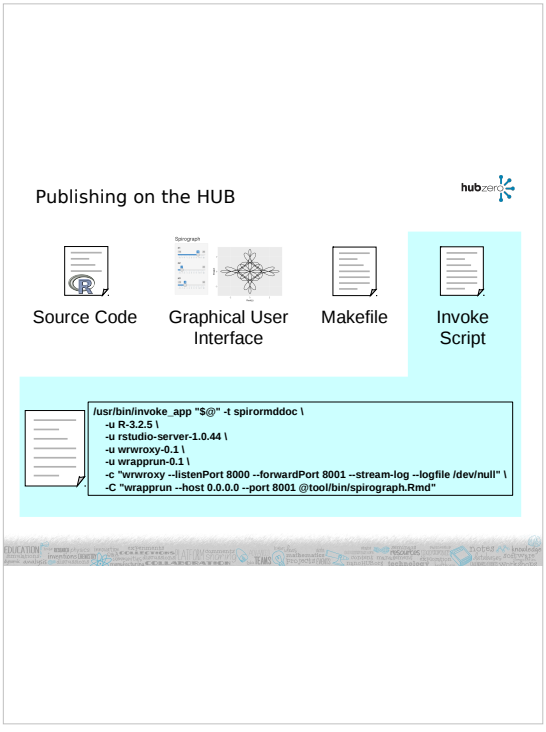
In the case where the application is a static web page, install files in the www directory.



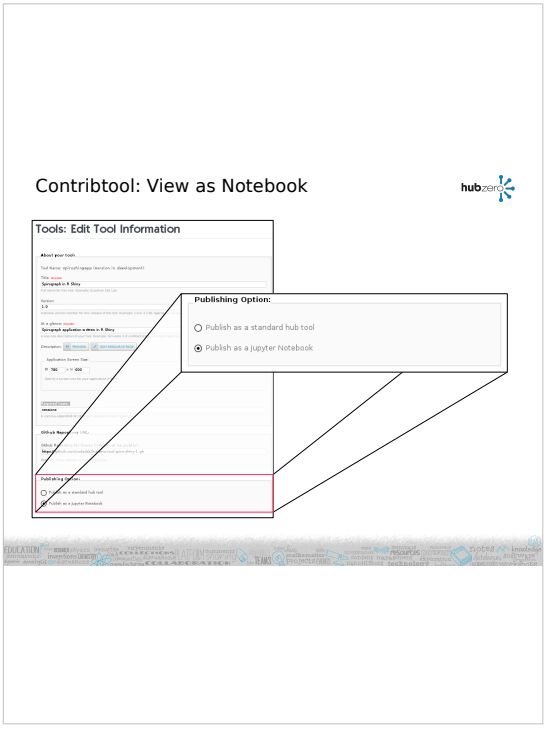
Lastly, the invoke script tells us how to launch the application. For Shiny and RMarkdown applications, we generally launch a rewrite proxy in front of the application. Users can call wraprun to launch their Shiny and RMarkdown application



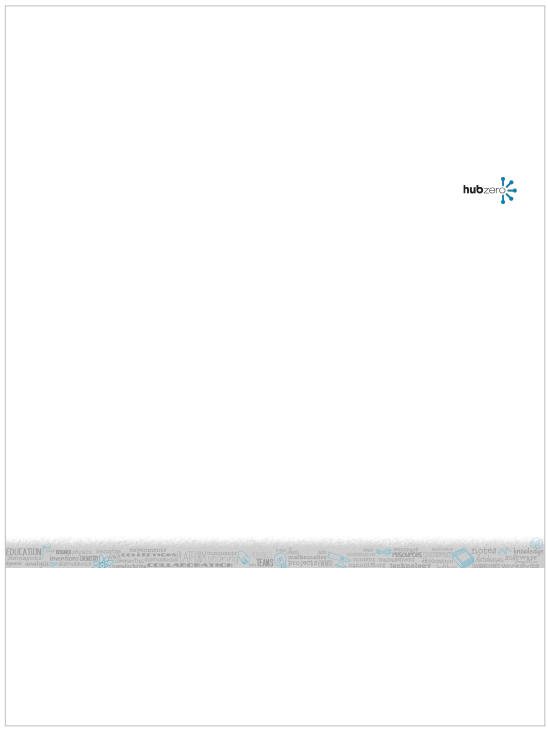
For Shiny and RMarkdown files can be installed in the bin directory or the www directory

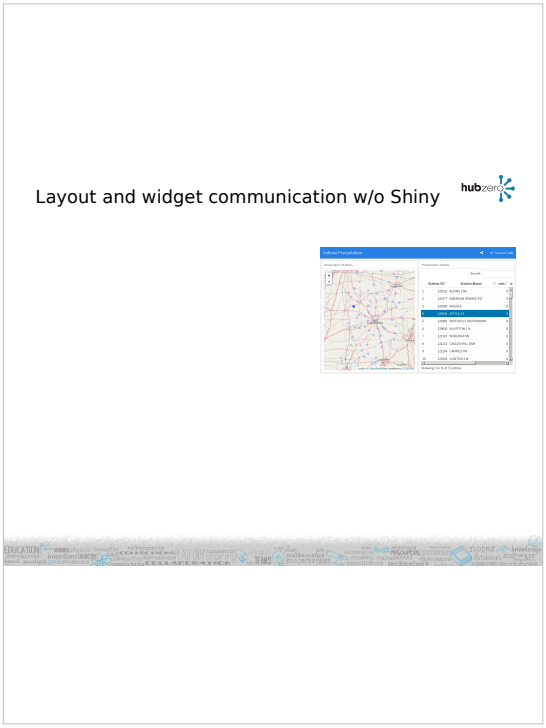


And the files can be handed to wraprun to launch the application.



In the Contribtool interface, we added a switch that tells the middleware to launch the application as a Weber (or Jupyter Notebook) application and to stream the HTML, JavaScript, and CSS back to the user's web browser, instead of using the VNC based viewer.





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.