DSECOP Module Implementation

A Guinea Pig Speaks

Jacob Hale

DePauw University

DSECOP Goals

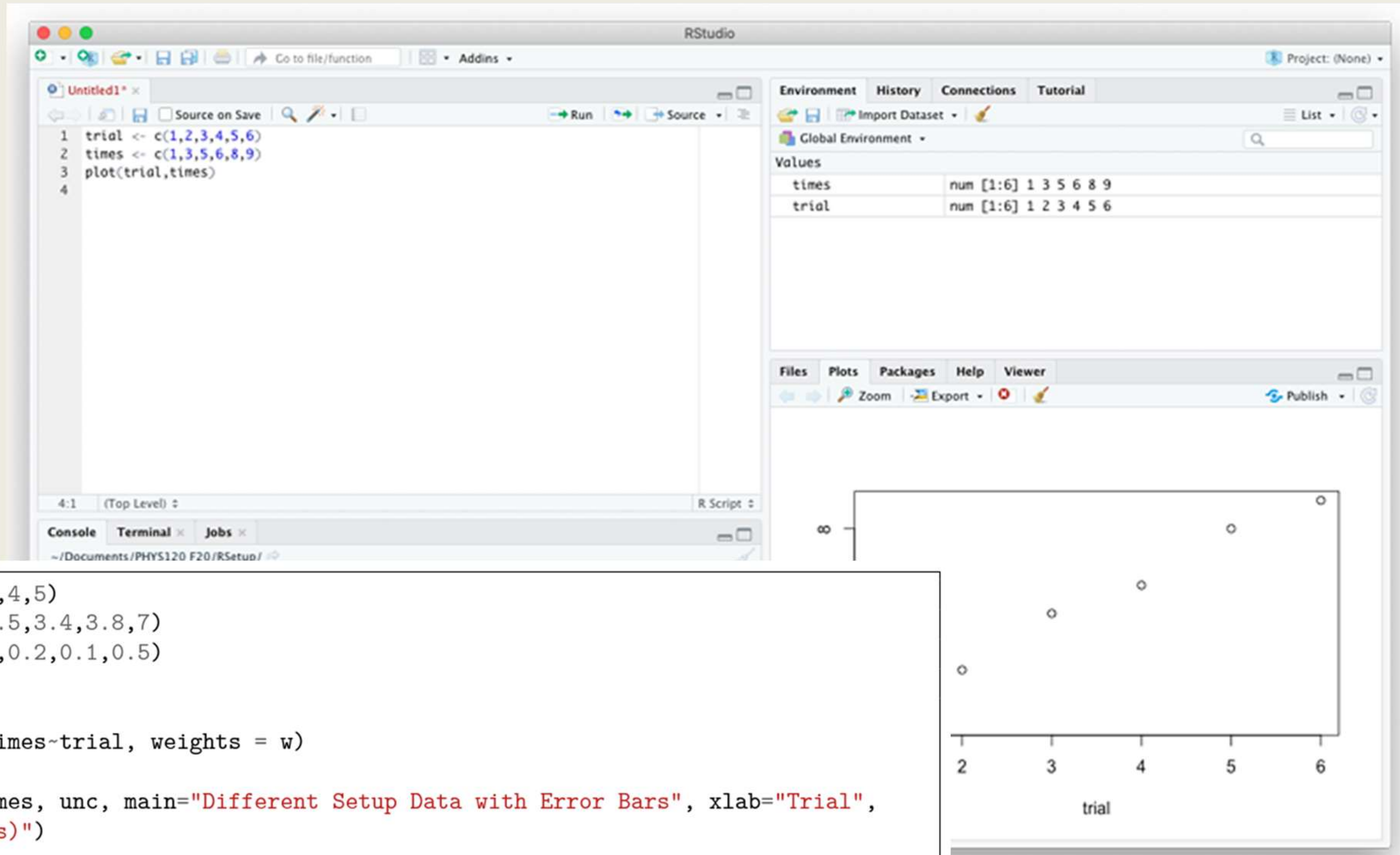
- Accessible to a broad range of schools/student experiences
- DS is a tool
  - Familiar to the Physics problem-solving process
  - Strengthens ability to do physics
  - Broadens students skill sets for career variation
  - I saw this particularly for my audience of only 1/3 going to grad school

Physics 280 Audience

- DePauw: PUI, liberal arts (BA), small, rural, residential
- Physics Department: graduate 2-5/year, 5 → 3 faculty
- 280 Students: majors or minors, broad preparation-math/coding/experimentation
- 280 the Course: collect and analyze data, communicate results
  - Randomly distributed dataset then linear data
  - Used to use Mathematica but moved to Python a year before this implementation

Department uses R in Intro Sequence

- Linear fits with errors: direct, proportional, logarithmic



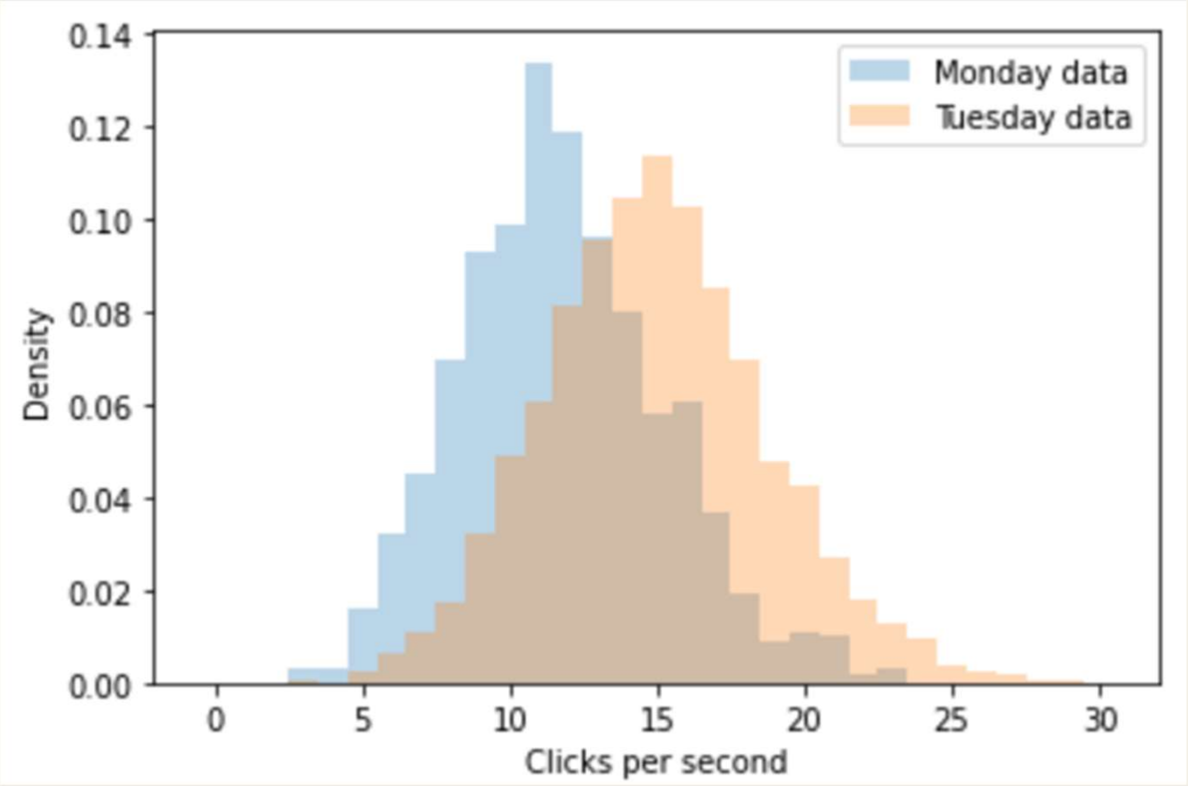

```r
trial <- c(1,2,3,4,5)
times <- c(1.1,2.5,3.4,3.8,7)
unc <- c(0.3,0.2,0.2,0.1,0.5)
w <- 1/unc^2

ourModel <- lm(times~trial, weights = w)

plotCI(trial, times, unc, main="Different Setup Data with Error Bars", xlab="Trial",
↪  ylab="Time (s)")

summary(ourModel)
abline(0.5453,0.8590)

legend("topleft",c("slp: 0.8590 ± 0.2319","int: 0.5453 ± 0.8142"),cex=0.8)
```

How Python was implemented in 280

- Introduction to Data Processing with Histograms module by Radha Mastandrea
  - Notebook 1: bring in data, make a brute force histogram, use the plt.hist function, plot multiple histograms with formatting
  - Notebook 2: brute force minimization of loss function, use the curve.fit function, discuss standard deviation
  - Notebook 3: calculate χ^2 and p-value
  - Notebook 4: create and interpret pull-plots

How Python was implemented in 280

- Introduction to Data Processing with Histograms module by Radha Mastandrea
  - Notebook 1: bring in data, make a brute force histogram, use the plt.hist function, plot multiple histograms with formatting



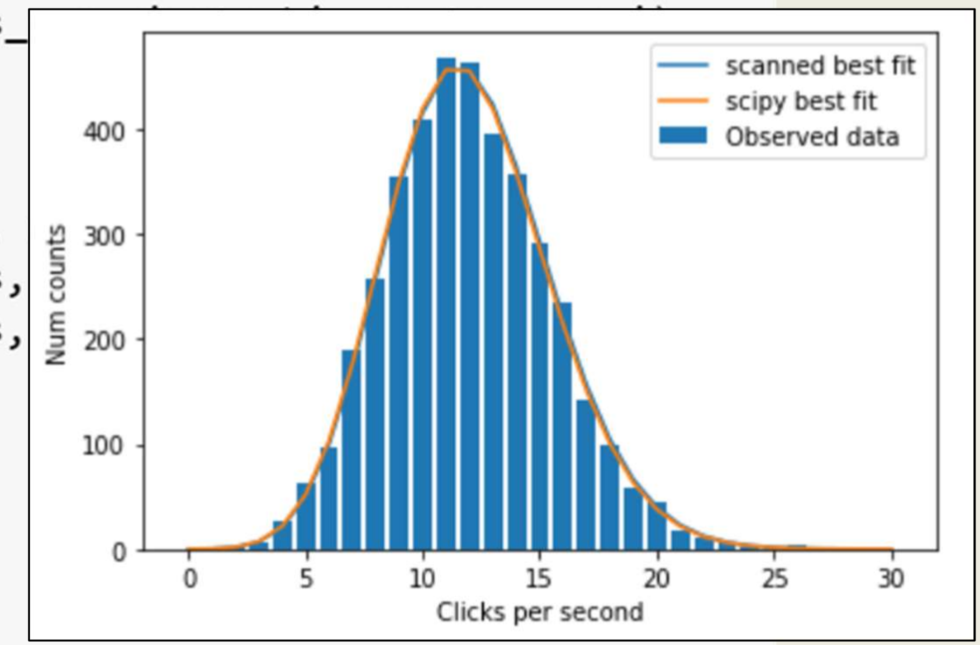
How Python was implemented in 280

- Introduction to Data Processing with Histograms module by Radha Mastandrea
  - Notebook 1: bring in data, make a brute force histogram, use the plt.hist function, plot multiple histograms with formatting
  - Notebook 2: brute force minimization of loss function, use the curve.fit function, discuss standard deviation
  - Notebook 3: calculate χ^2 and p-value
  - Notebook 4: create and interpret pull-plots

How Python was implemented in 280

- Introduction to Data Processing with Histograms module by Radha Mastandrea
  - Notebook 2: brute force minimization of loss function, use the curve.fit function, discuss standard deviation

```
from scipy.optimize import curve_fit


popt, pcov = curve_fit(poisson, clicks_


plt.figure()
plt.bar(clicks_per_minute_bin_centers,
plt.plot(clicks_per_minute_bin_centers,
plt.plot(clicks_per_minute_bin_centers,

plt.legend()

plt.xlabel("Clicks per minute")
plt.ylabel("Num counts")
plt.show()
```



How Python was implemented in 280

- Introduction to Data Processing with Histograms module by Radha Mastandrea
  - Notebook 1: bring in data, make a brute force histogram, use the plt.hist function, plot multiple histograms with formatting
  - Notebook 2: brute force minimization of loss function, use the curve.fit function, discuss standard deviation
  - Notebook 3: calculate χ^2 and p-value
  - Notebook 4: create and interpret pull-plots

How Python was implemented in 280

- Introduction to Data Processing with Histograms module by Radha Mastandrea
  - Notebook 3: calculate χ^2 and p-value

```
from scipy.stats import chisquare

result_gaussian = chisquare(n_i, exp_gaussian, 2)
result_poisson = chisquare(n_i, exp_poisson, 1)

print("$\Chi^2$ for gaussian fit =", result_gaussian)
print("$\Chi^2$ for poisson fit =", result_poisson)
```

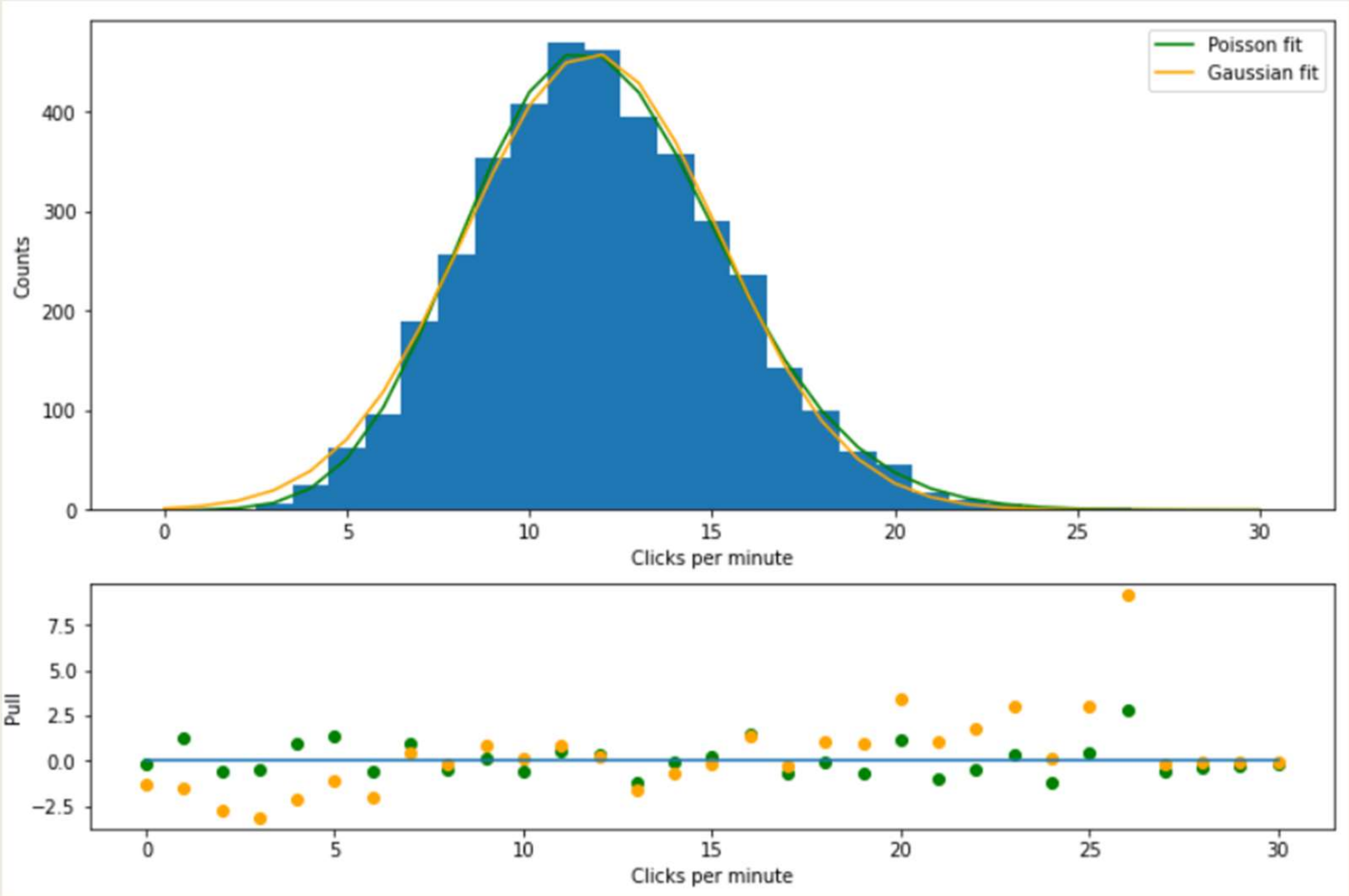
```
$\Chi^2$ for gaussian fit = Power_divergenceResult(statistic=158.5802040119228, pvalue=3.442260037741459e-20)
$\Chi^2$ for poisson fit = Power_divergenceResult(statistic=24.961948378337947, pvalue=0.6802205648039448)
```

How Python was implemented in 280

- Introduction to Data Processing with Histograms module by Radha Mastandrea
  - Notebook 1: bring in data, make a brute force histogram, use the plt.hist function, plot multiple histograms with formatting
  - Notebook 2: brute force minimization of loss function, use the curve.fit function, discuss standard deviation
  - Notebook 3: calculate χ^2 and p-value
  - Notebook 4: create and interpret pull-plots

How Python was implemented in 280

- Introduction to Data Processing with Histograms module by Radha Mastandrea
  - Notebook 4: create and interpret pull-plots



How Python was implemented in 280

- Class time, Day 1
  - Reviewed statistical theory from book
    - Taylor, An Introduction to Error Analysis
- Homework 1
  - Tutorial to install Anaconda for Spyder and Jupyter Notebooks
    - Lots of Google sources
  - Very brief intro to Jupyter Notebook usage
  - Very brief intro to Python tutorial to prep for Module
    - Bringing in libraries
    - Creating or bringing in data
    - Inspecting and modifying the data
    - Simple plotting
    - Custom functions: def

How Python was implemented in 280

Generated random dataset

```
rng=np.random.default_rng()#create a Random number generator
N=1000#pick how many trials
A0=10#pick a true average value for a gaussian distribution
std0=2#pick a standard deviation for a gaussian distribution
x=rng.normal(A0,std0,N)#create a data set that is normally distributed
np.savetxt('randomA10S2N1000.csv',x)
```

Manipulating the data

Now that we have data we can do things to it. Run the following code, determine what it is doing, and evaluate the output.

```
data=x
dataave=np.average(data)
datastdev=np.std(data)
N=len(data)
print('The average and standard deviation of the data is '+ f'{dataave:.2f}' +' +/- ' + '{:4.2f}'.format(datastdev) + '.')
```

```
The average and standard deviation of the data is 9.99 +/- 1.96.
```

As you can see, I used commands within `numpy` to calculate the average and standard deviation of the dataset. I also used a standard Python command, `len`, to count how many values were in the dataset, or its length.

The first line in the above code isn't necessary but it allows us to efficiently analyze different datasets as desired. Instead of typing y or zarray in each of the above commands, we assigned a generic name to the preferred dataset and use that name in the rest of the code.

In printing the output I showed two ways how to format the number so it wasn't so long. I encourage you to look up the format command on Google to understand how it works.

How Python was implemented in 280

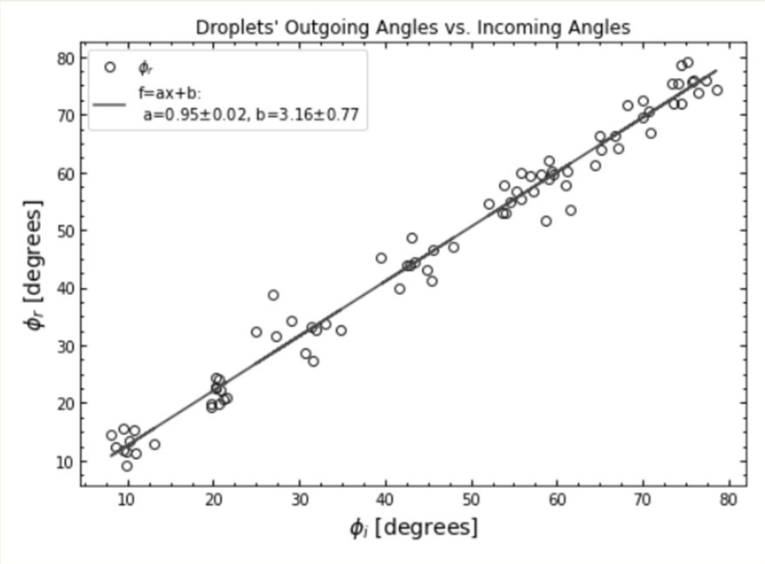
- Class time, Day 2
  - Demonstrated basic Python coding
  - Students had to comment Notebook 1 and 2 starting in class
  - Encouraged "Googling it" habit
- Homework 2
  - Finish commenting on Notebook 1 and 2

How Python was implemented in 280

- Class time, Day 3
  - Review Notebook 1 and 2 code
  - Students get started to create a new notebook to create a histogram of their data with fit
  - Reviewed χ^2 theory
  - Introduced code for fitting
- Homework 3
  - Finish histogram of data
  - Comment on Notebook 3 and 4 OR fill in needed code
  - Calculate χ^2 for their data, show data complete with fit, χ^2, and pull plot

How Python was implemented in 280

- Class time, Day 4
  - (After collecting data exploring one parameter)
  - Review linear fitting theory
  - Students apply fitting ideas from Module to find linear fit with errors to their data
- Homework 4
  - Create plot of data with linear fit line and values with error in legend



Reflections

- Diverse audience is a challenge but modules are flexible and “modular”
- Would modify modules to meet specific needs of course: “customizable” with experience
- Increase the number of exercises/check points for in class active learning

Other Audiences

- What is your student population?
- How can you implement this Module or other Modules to your students and content?