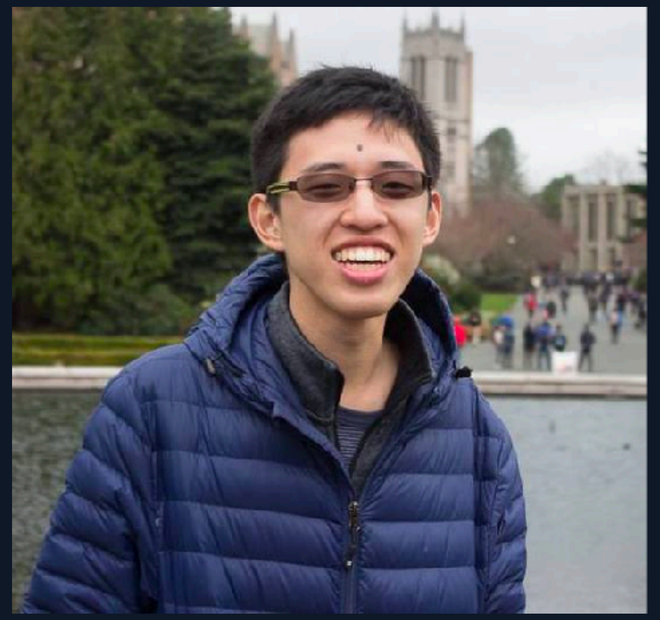
nbinteract

Interactive web pages from notebooks

Who We Are

Sam Lau
UC Berkeley, UC San Diego



Caleb Siu
UC Berkeley

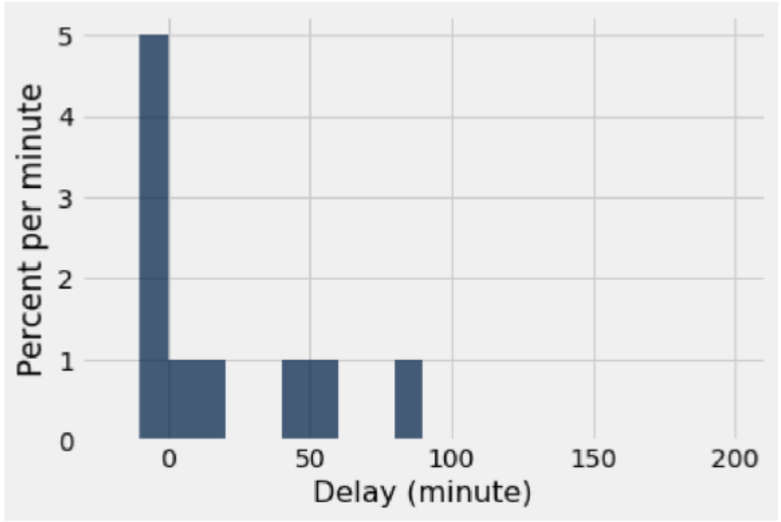
Motivation

Writing a data science textbook

As sample size goes up,
sample looks like the population

As we saw with the dice, as the sample size increases, the empirical histogram of the sample more closely resembles the histogram of the population. Compare these histograms to the population histogram above.

```
empirical_hist_delay(10)
```



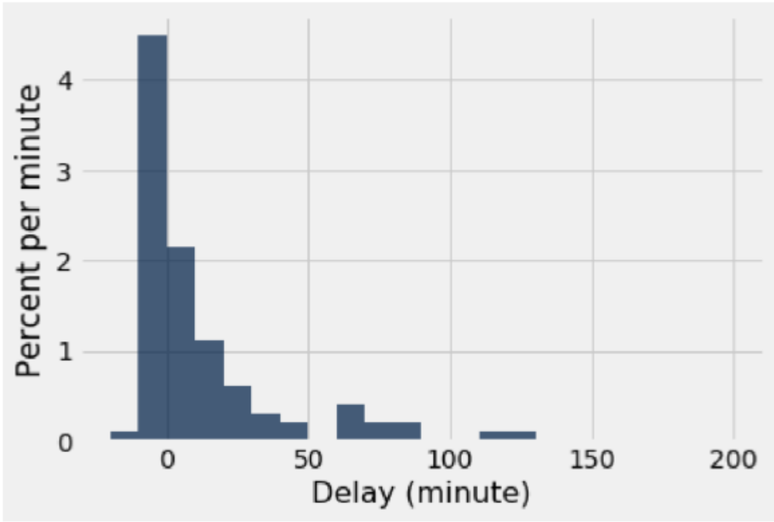
https://www.inferentialthinking.com/chapters/10/1/Empirical_Distributions

Writing a data science textbook

As sample size goes up,
sample looks like the population

https://www.inferentialthinking.com/chapters/10/1/Empirical_Distributions

```
empirical_hist_delay(100)
```



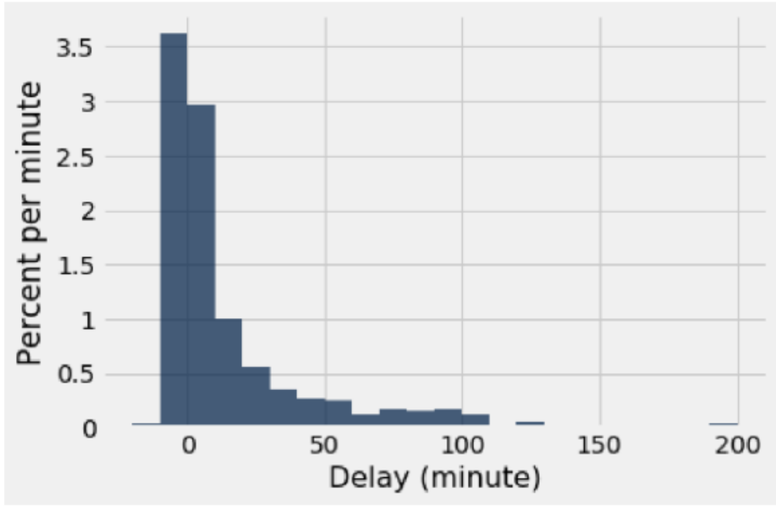
Writing a data science textbook

As sample size goes up, sample looks like the population

https://www.inferentialthinking.com/chapters/10/1/Empirical_Distributions

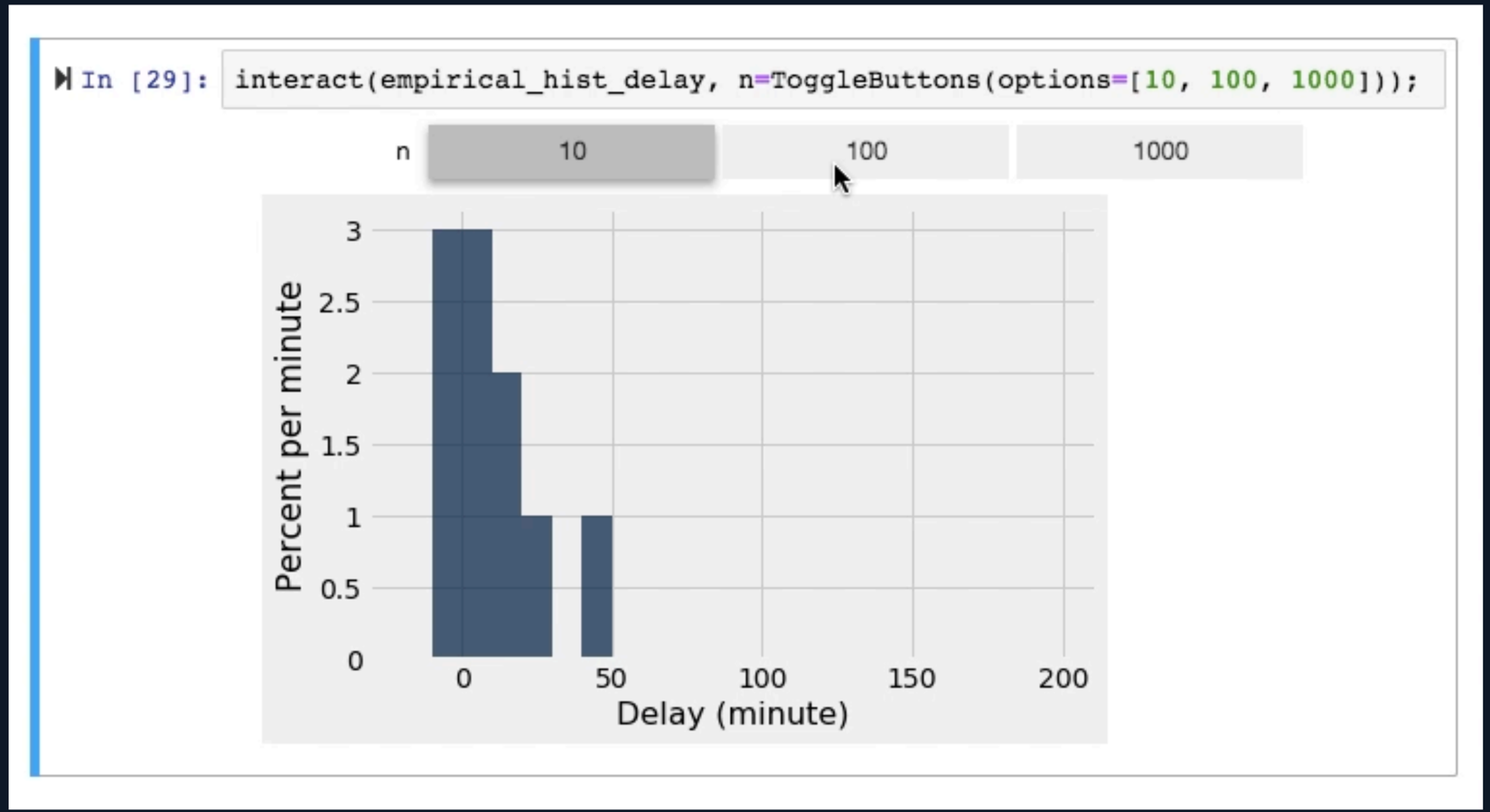
The most consistently visible discrepancies are among the values that are rare in the population. In our example, those values are in the the right hand tail of the distribution. But as the sample size increases, even those values begin to appear in the sample in roughly the correct proportions.

```
empirical_hist_delay(1000)
```



ipywidgets

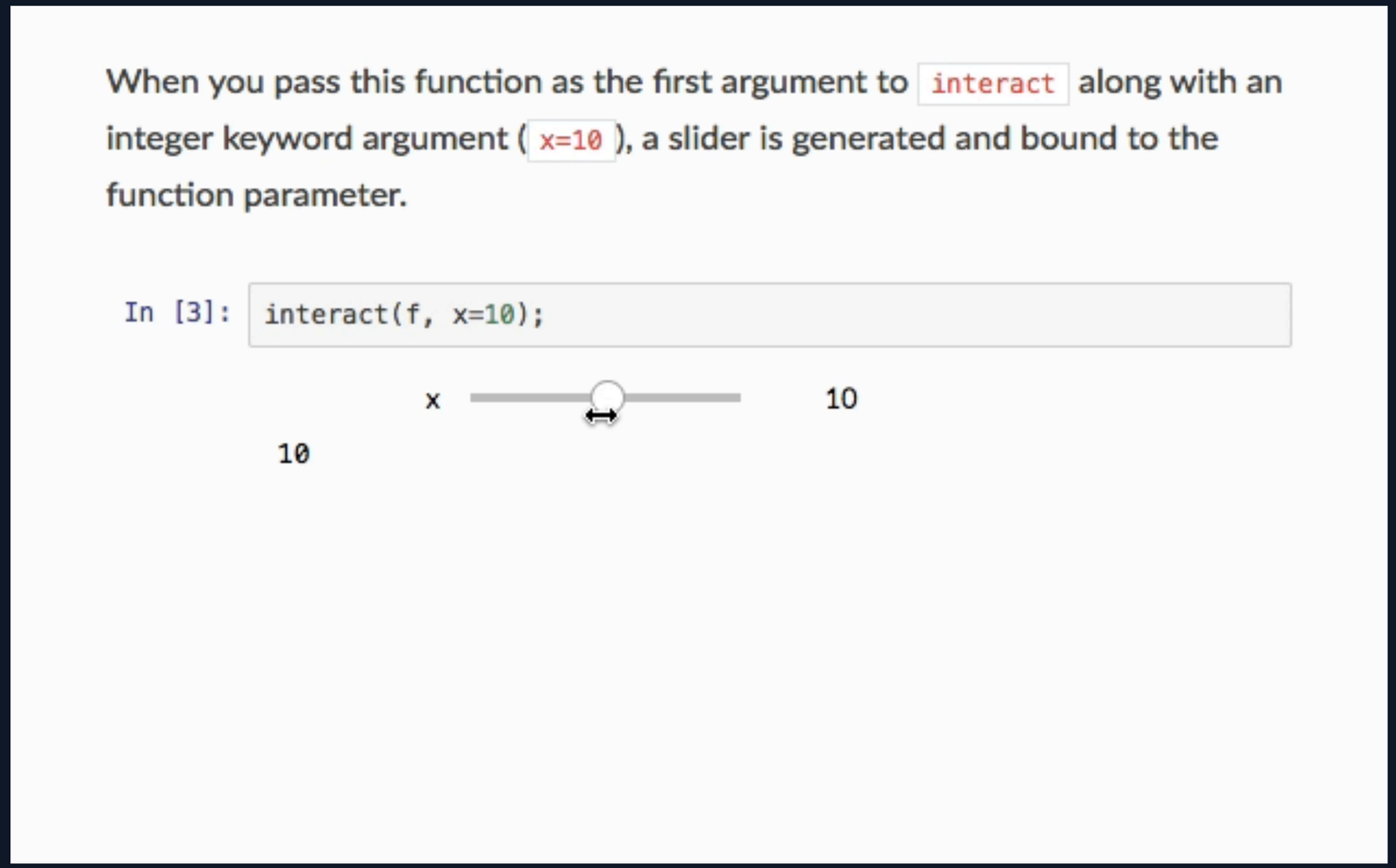
Quickly make interactive content in Jupyter!...



ipywidgets

...but can't export to HTML pages

(This example from ipywidget docs)



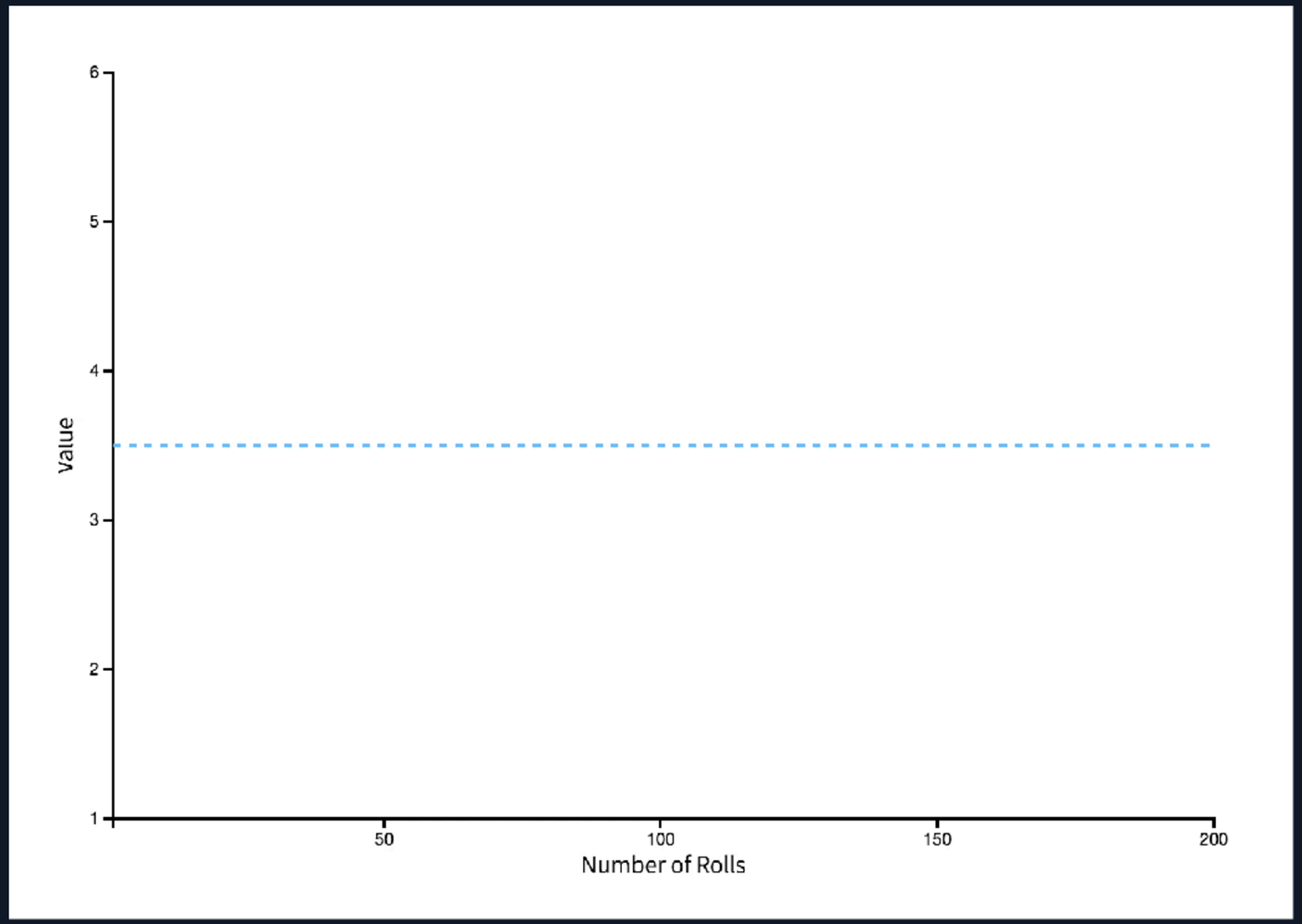
https://ipywidgets.readthedocs.io/en/stable/examples/Using%20Interact.html

The nbinteract CLI generates interactive webpages from Jupyter notebooks.

Exporting widgets to HTML

(demo)

Using JavaScript



Great effect, but…

https://students.brown.edu/seeing-theory/basic-probability/index.html#section1

300 lines of D3.js

10 lines of logic

https://github.com/danielkunin/Seeing-Theory/blob/ea98a1f9898994432be6d88a63de85a24813a828/basic-probability/basic-probability.js#L185-L466

```
function expectation() {
  //Constants
  var probDie = [{p:1/6},{p:1/6},{p:1/6},{p:1/6},{p:1/6},{p:1/6}];
  var countDie = [0,0,0,0,0,1];
  var expectedData = [average(countDie)];

  //Create SVG and SVG elements
  var svgDie = d3.select("#barDie").append("svg");

  //Create Container
  var containerDie = svgDie.append("g").attr('class','True');

  //yScale
  var yScaleDie = d3.scale.linear().domain([0,1]);

  //xScale
  var xScaleDie = d3.scale.ordinal().domain([1,2,3,4,5,6]);

  //xAxis
  var xAxisDie = d3.svg.axis().scale(xScaleDie).orient("bottom").ticks(0);
  var axisDie = svgDie.append("g").attr("class", "x axis");

  //Drag function for coin bar chart
  var dragDie = d3.behavior.drag()
          .origin(function(d,i) { return {x: 0, y: yScaleDie(d.p)};})
          .on('drag', function(d,i) {
                  var y = Math.min(1,Math.max(0,yScaleDie.invert(d3.event.y)));
                  var oldV = probDie[i].p;
                  var change = y-oldV;
                  probDie.map(function(x,index){
                      if(index==i) x.p=y;
                      else {
                        if(oldV==1) x.p= -change/5;
                        else x.p= x.p-change*x.p/(1-oldV);
                      }});
                  updateDie();
                  tipDie.show(d,this);
                  countDie = [0,0,0,0,0,0];
                  expectedData = [];
                  maxXExpected = 200;
                  expectation(expectedData,expectationCalc(probDie));
                })

  //Create Rects
  var expectedRects = containerDie.selectAll("rect").data(probDie).enter().append("rect");

  //Create Labels
  var dieFaces = svgDie.select("g.axis").selectAll("g.tick").data(probDie).enter().append("image");

  //Tool Tip
  var tipDie = d3.tip().attr('id', 'tipDie').attr('class', 'd3-tip').offset([-10, 0]);

  //Update Coin Bar Chart
  function updateDie() {

      tipDie.html(function(d,i) { return round(d.p,2);});


    expectedRects
        .attr("x",function(d,i) {return xScaleDie(i+1);})
        .attr("y",function(d,i) {return yScaleDie(d.p);})
        .attr("height",function(d,i) {return yScaleDie(1-d.p);})
        .attr("width",xScaleDie.rangeBand())
        .attr("id",function(d,i) {return i;})
        .on('mousedown', function(d){tipDie.show(d,this)})
        .on('mouseover', function(d){tipDie.show(d,this)})
        .on('mouseout', tipDie.hide)
        .call(dragDie);

    $('#barDie').parent().on('mouseup', tipDie.hide);

    svgDie.select(".axis").selectAll(".tick").remove();
    dieFaces
          .attr("xlink:href", function(d,i) { return "../img/dice_"+(i+1)+".png"; })
          .attr("x", function(d,i) {return xScaleDie(i+1)-1/4*xScaleDie.rangeBand();})
          .attr("y", 0)
          .attr("width", 3/2*xScaleDie.rangeBand())
          .attr("height", 3/2*xScaleDie.rangeBand());
  }

  //Handles Die Roll
  function roll(die){
    var num = Math.random();
    var cumProb = cumsum(probDie);
    if (num<cumProb[0]) {
      die.css("background-image", "url(../img/dice_1.png");
      countDie[0] = countDie[0] + 1;
    } else if (num<cumProb[1]) {
      die.css("background-image", "url(../img/dice_2.png");
      countDie[1] = countDie[1] + 1;
    } else if (num<cumProb[2]) {
      die.css("background-image", "url(../img/dice_3.png");
      countDie[2] = countDie[2] + 1;
    } else if (num<cumProb[3]) {
```

Exporting plots with widgets

(demo)

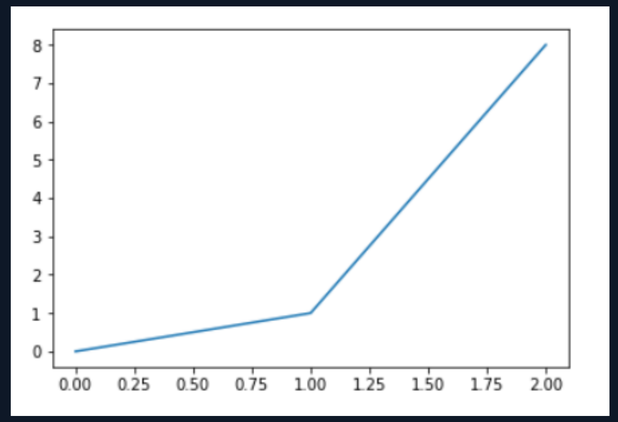
nbinteract CLI Usage

```
$ pip install nbinteract
$ nbinteract -s {BINDER_SPEC}   {NOTEBOOKS}

# Produces hello.html
$ nbinteract -s DS-100/textbook hello.ipynb

# After running nbinteract init
$ nbinteract hello.ipynb
```

Why Spec?

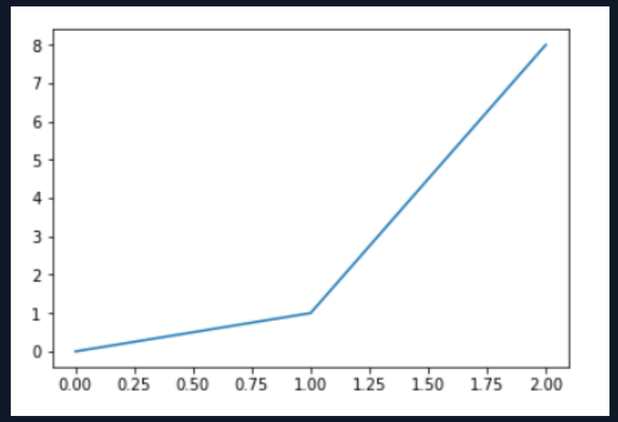
Widgets call Python functions on interaction

Usually, a local notebook server runs the code



Why Spec?

Widgets call Python functions on interaction

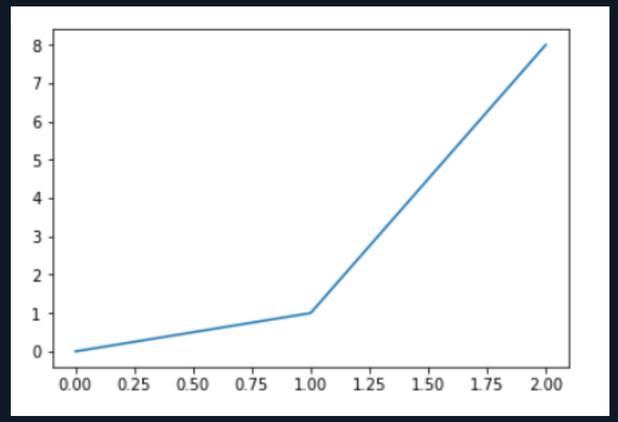


nbinteract uses a Binder server to run your code instead

https://mybinder.org/

Why Spec?

Widgets call Python functions on interaction



...which requires configuration for what packages to include.

`requirements.txt`

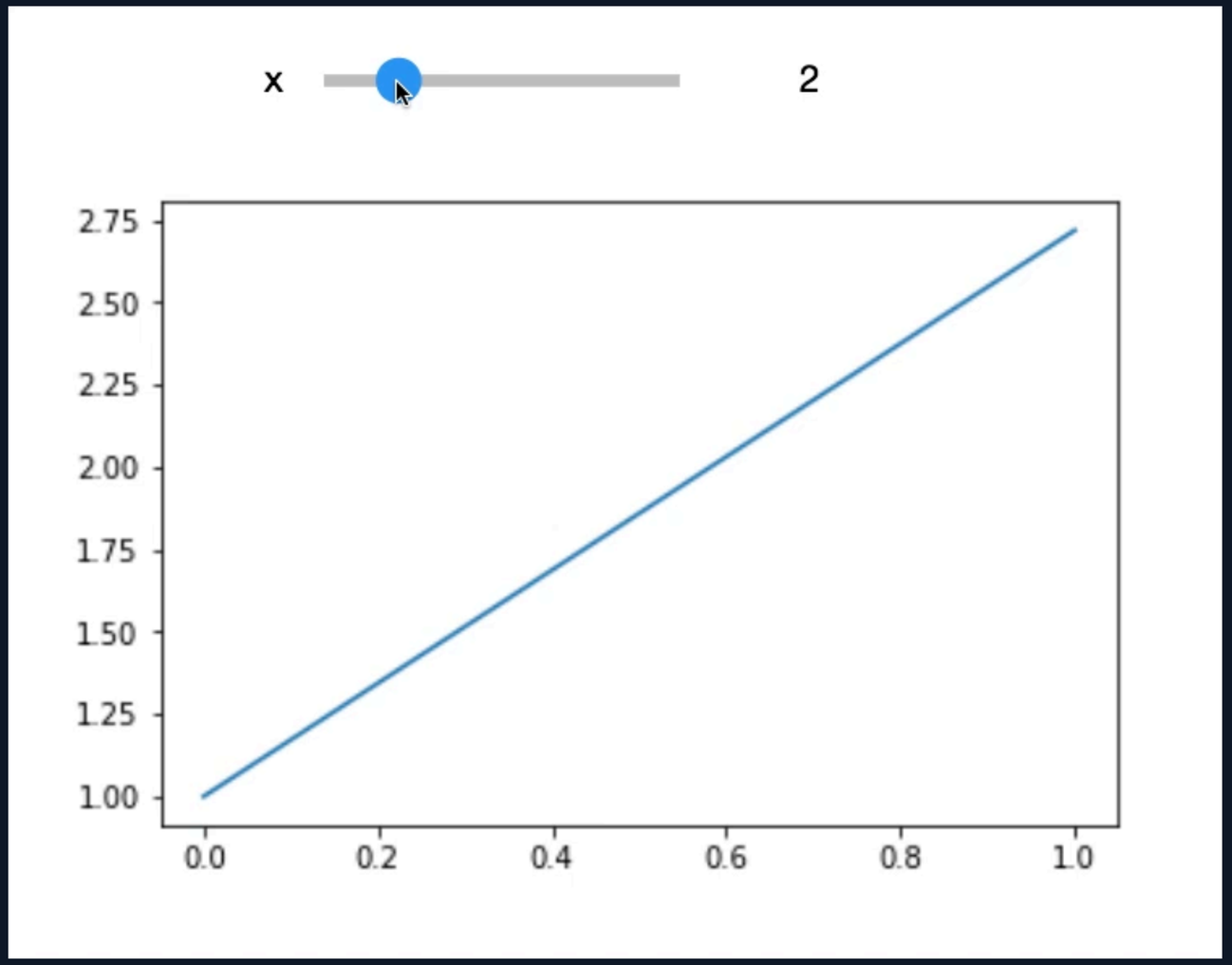
Okay, how do I get started?

Do this...	So that...
Create GitHub repo	Binder can create a server for you
`nbinteract init`	Your repo has a requirements.txt file for Binder to install packages. You can omit the Binder spec when converting notebooks.
`nbinteract foo.ipynb`	You convert your notebook to HTML

The nbinteract CLI generates interactive webpages from Jupyter notebooks.

The nbinteract Python package provides plotting functions for smooth interactions.

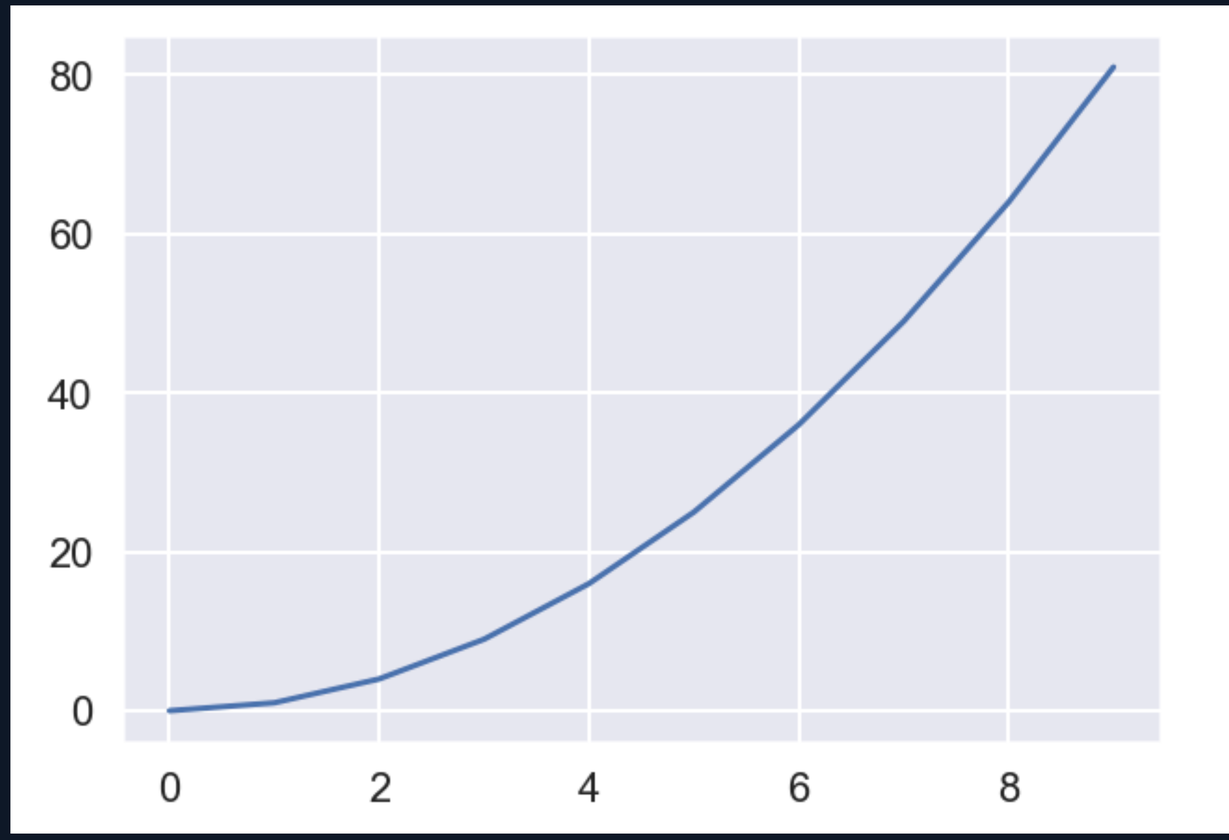
matplotlib Interactions Are Slow



Because the library generates a new image every interaction.

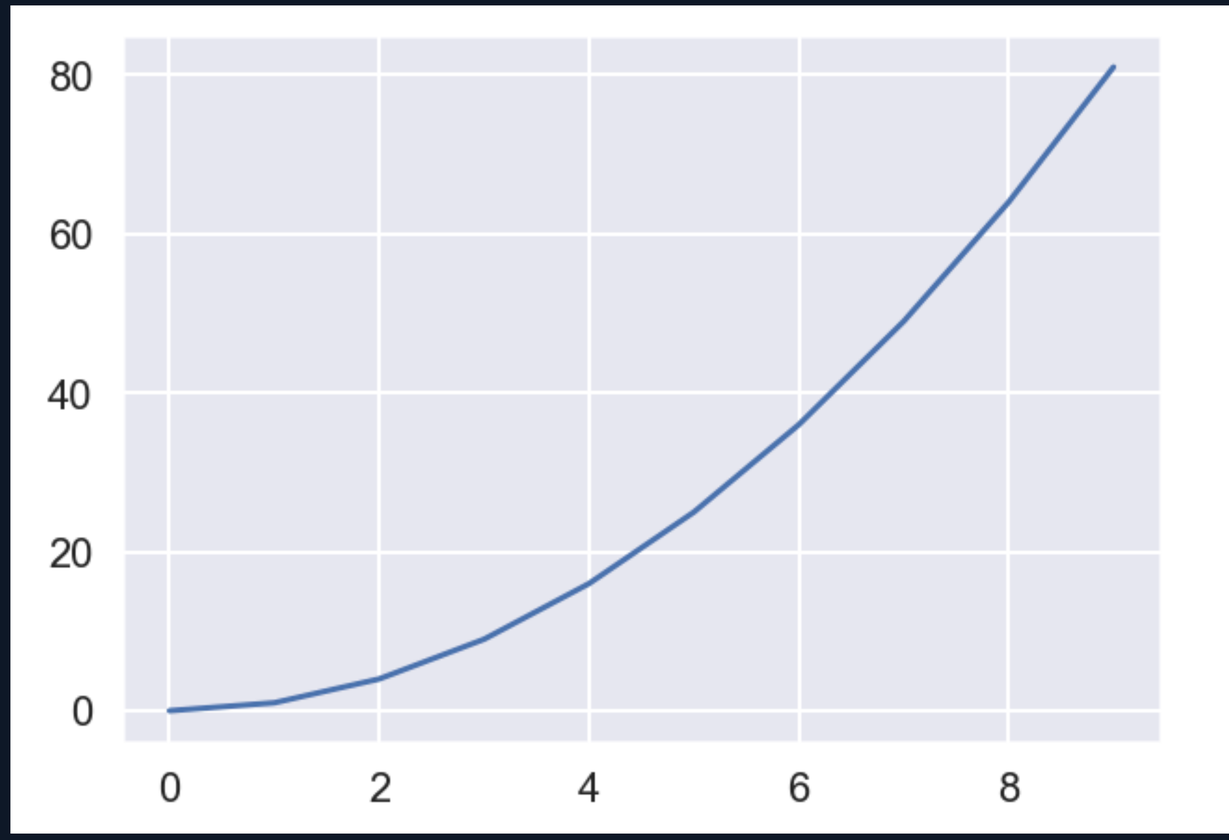
nbinteract plotting

```
x_vals = np.arange(10)

y_vals = x_vals ** 2


plt.plot(x_vals, y_vals)
```



nbinteract plotting

```
x_vals = np.arange(10)

y_vals = x_vals ** 2


plt.plot(x_vals, y_vals)
```



nbinteract plotting

```
x_vals = np.arange(10)

y_vals = x_vals ** 2


nbi.line(x_vals, y_vals)
```

nbinteract plotting

```
x_vals = np.arange(10)

y_vals = x_vals ** 2


nbi.line(x_vals, y_vals)
```

nbinteract plotting

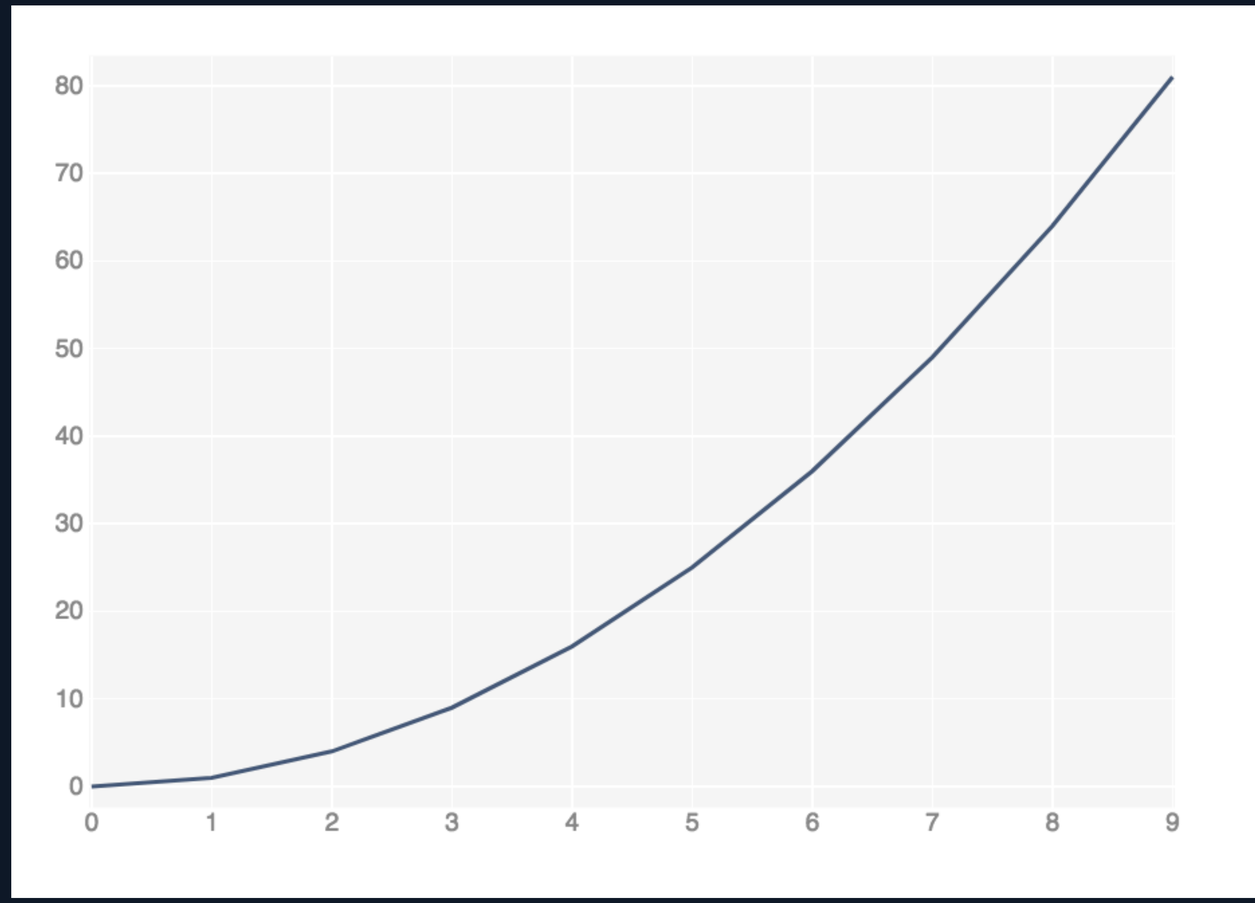
```
def x_vals():
    return np.arange(10)

y_vals = x_vals ** 2



nbi.line(x_vals, y_vals)
```

nbinteract plotting

```
def x_vals():
    return np.arange(10)

y_vals = x_vals ** 2



nbi.line(x_vals, y_vals)
```

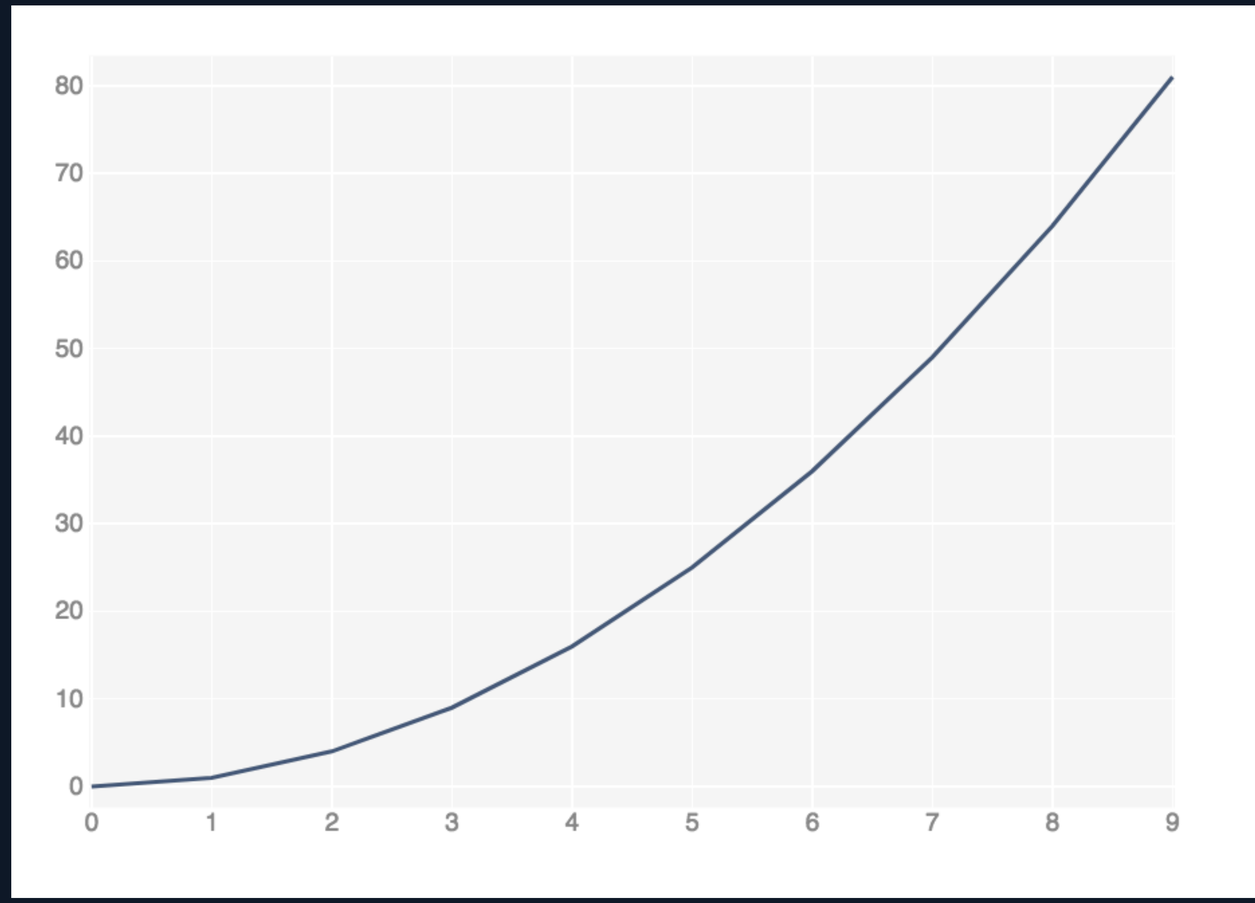
nbinteract plotting

```
def x_vals():
    return np.arange(10)

def y_vals(xs):
    return xs ** 2


nbi.line(x_vals, y_vals)
```



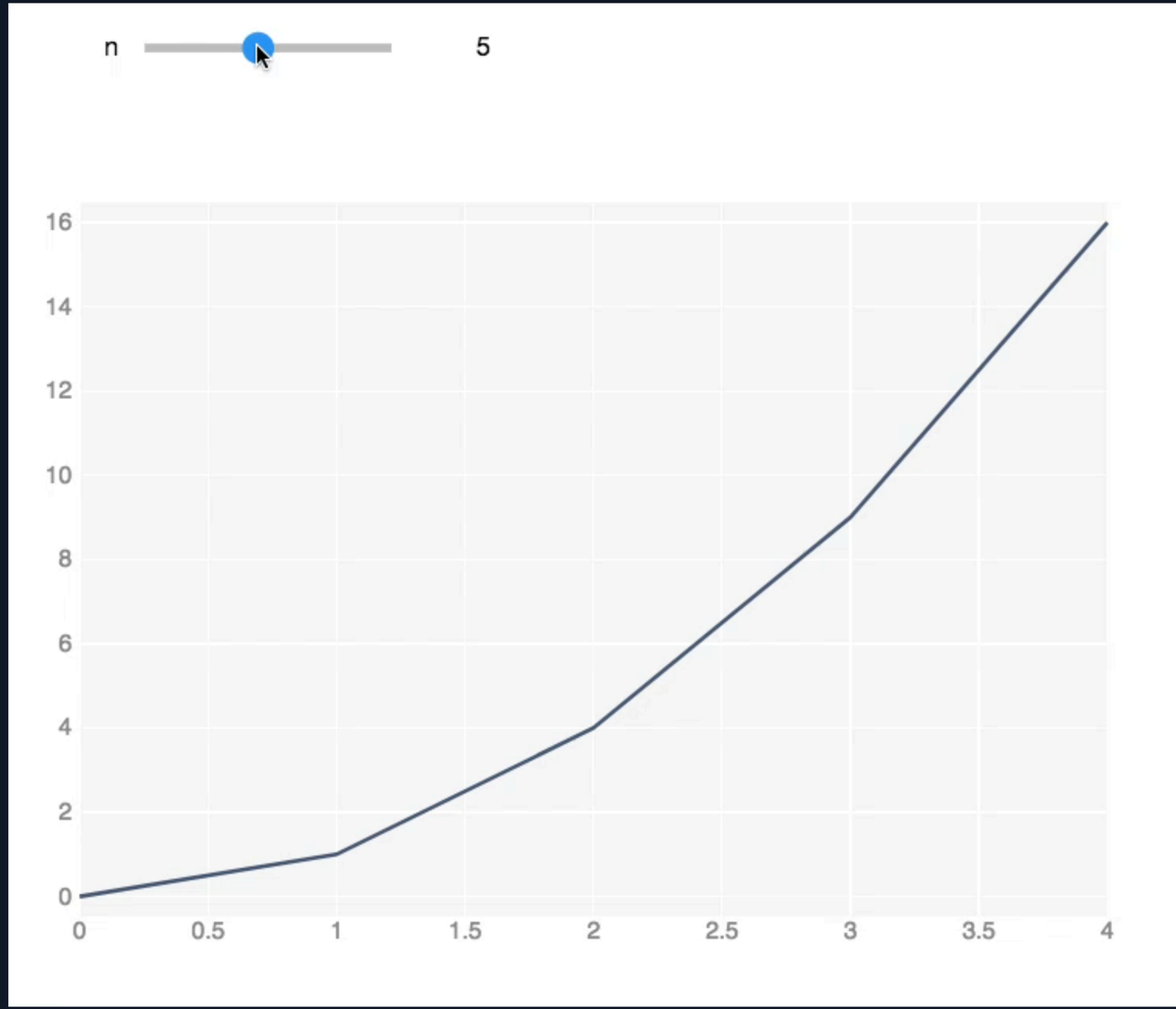
nbinteract plotting

```
def x_vals():
    return np.arange(10)

def y_vals(xs):
    return xs ** 2


nbi.line(x_vals, y_vals)
```



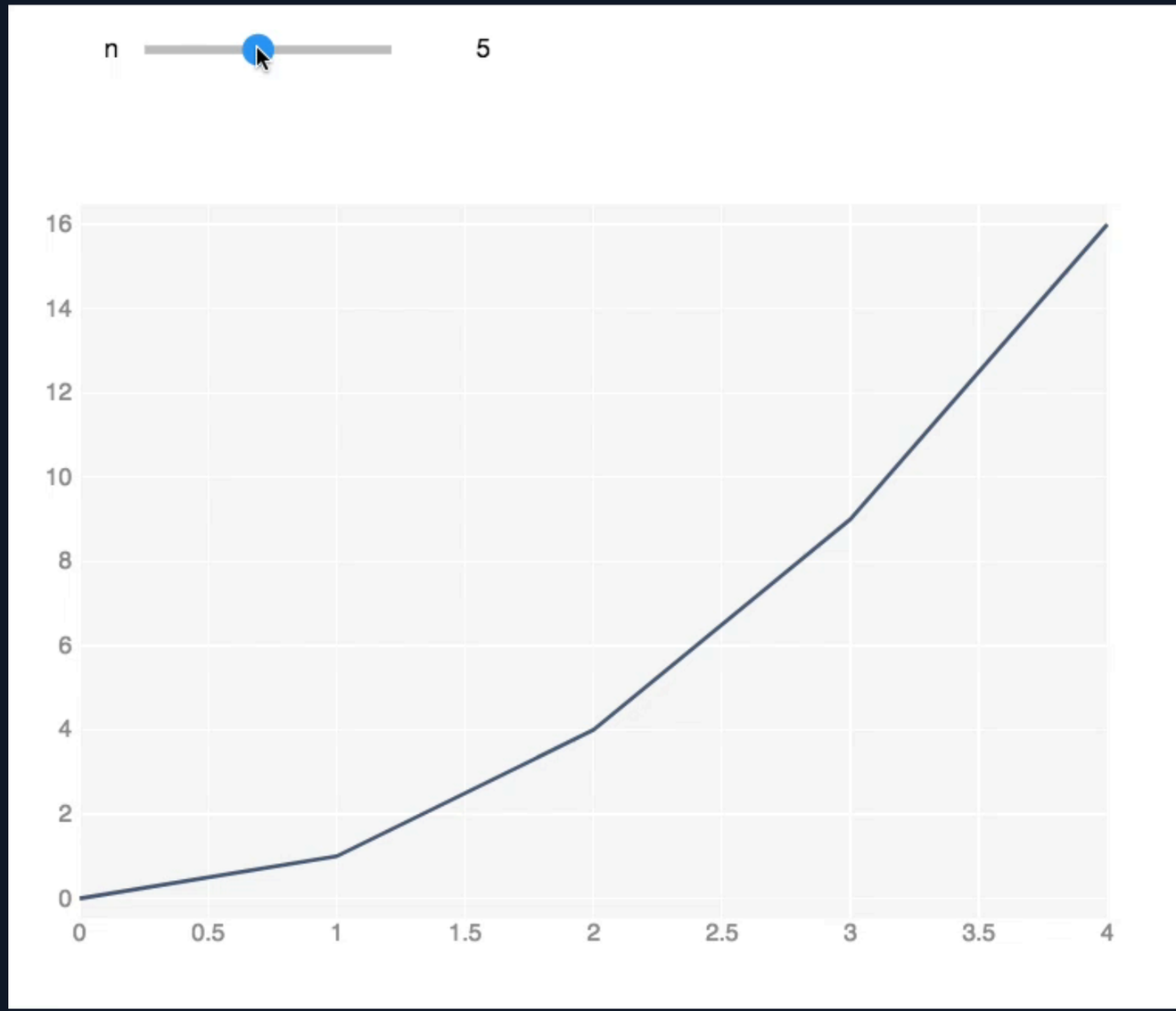
nbinteract plotting

```
def x_vals(n):
    return np.arange(n)

def y_vals(xs):
    return xs ** 2


nbi.line(x_vals, y_vals,
         n=(1, 10))
```



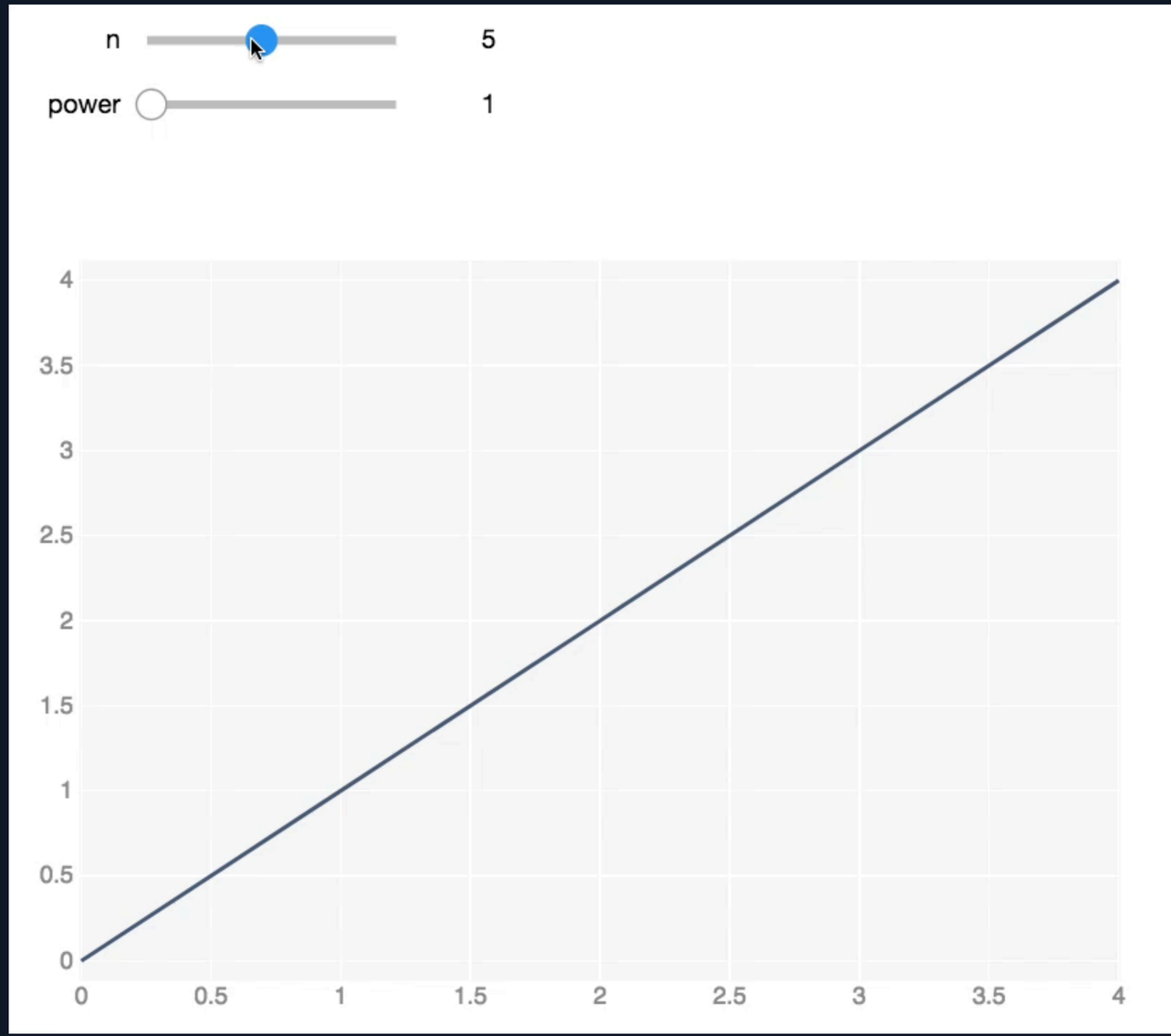
nbinteract plotting

```
def x_vals(n):
    return np.arange(n)

def y_vals(xs):
    return xs ** 2


nbi.line(x_vals, y_vals,
         n=(1, 10))
```



nbinteract plotting

```
def x_vals(n):
    return np.arange(n)

def y_vals(xs, power):
    return xs ** power


nbi.line(x_vals, y_vals,
         n=(1, 10),
         power=(1, 5))
```



Available Plot Functions

Function	Description
`nbi.hist(vals)`	**Interactive histogram.**
`nbi.bar(x_vals, y_vals)`	**Interactive bar chart.**
`nbi.scatter(x_vals, y_vals)`	**Interactive scatter plot.**
`nbi.line(x_vals, y_vals)`	**Interactive line plot.**

https://www.nbinteract.com/recipes/recipes_graphing.html

Our First Example

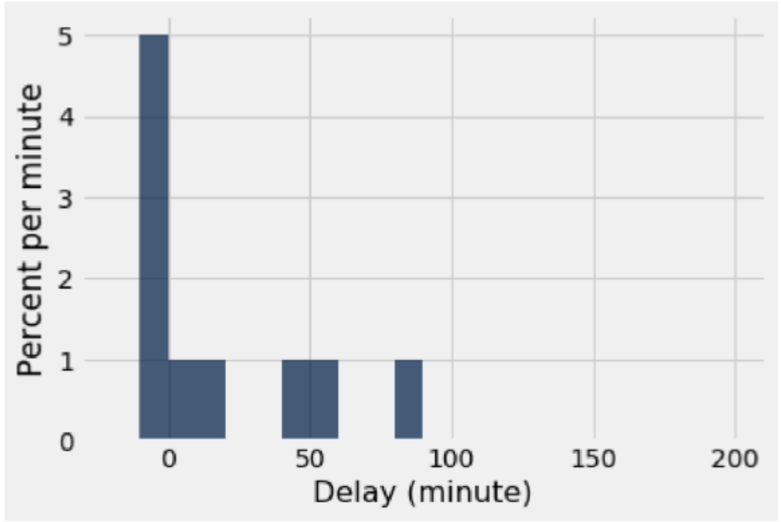
Writing a data science textbook

As sample size goes up,
sample looks like the population

https://www.inferentialthinking.com/chapters/10/1/Empirical_Distributions

As we saw with the dice, as the sample size increases, the empirical histogram of the sample more closely resembles the histogram of the population. Compare these histograms to the population histogram above.

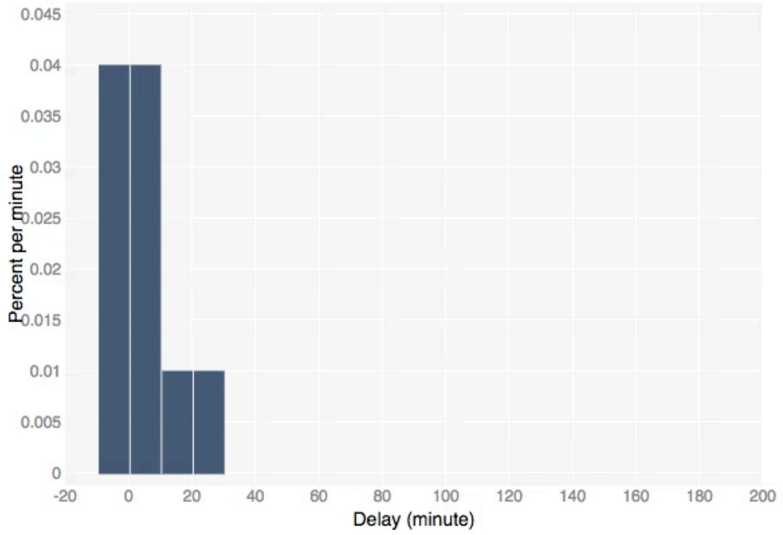
```
empirical_hist_delay(10)
```



As we saw with the dice, as the sample size increases, the empirical histogram of the sample more closely resembles the histogram of the population. Compare these histograms to the population histogram above.

```
nbi.hist(empirical_hist_delay,
         options=delay_opts,
         sample_size=widgets.ToggleButtons(options=[10, 100, 1000, 10000], descriptio
```

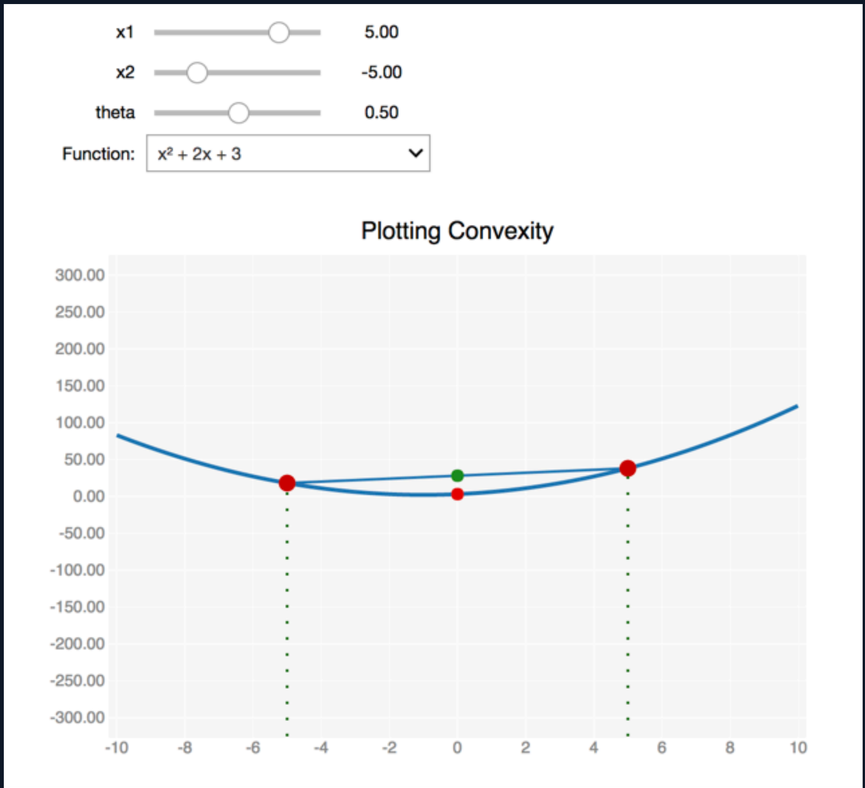
Sample Size: 10 100 1000 10000



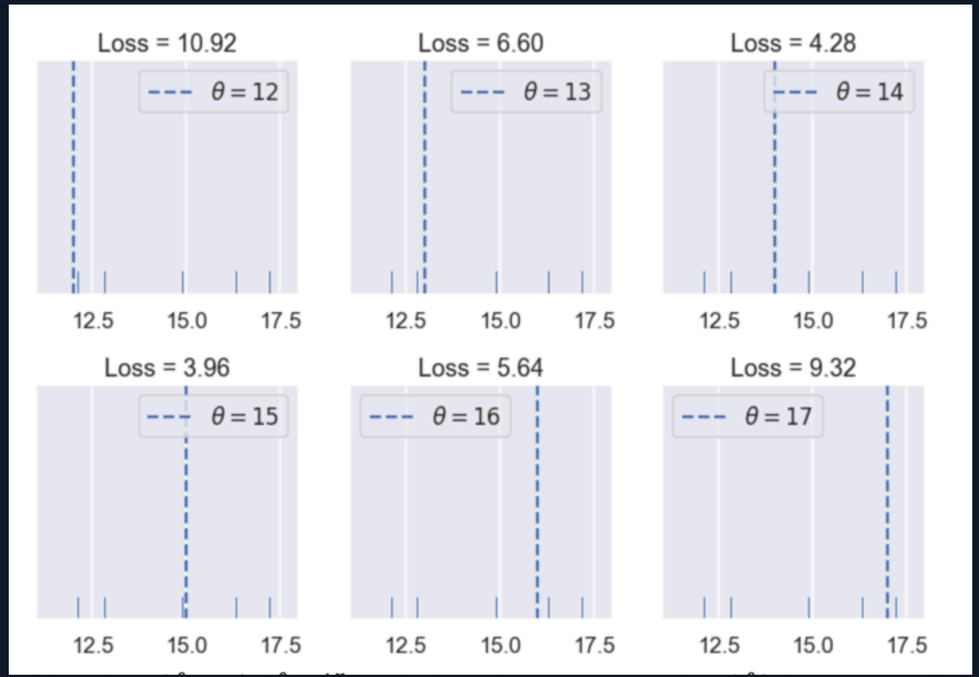
The most consistently visible discrepancies are among the values that are rare in the population. In our example, those values are in the the right hand tail of the distribution. But as the sample size increases,

https://www.nbinteract.com/examples/examples_sampling_from_a_population.html

nbinteract can't make these (use matplotlib)



Multiple marks



Subplots

Use Cases

Displaying a Large Table

```
# Load the dataset and drop N/A values to make plot function calls simpler
ti = sns.load_dataset('titanic').dropna().reset_index(drop=True)

# This table is too large to fit onto a page so we'll output sliders to
# pan through different sections.
df_interact(ti)
```

row 100

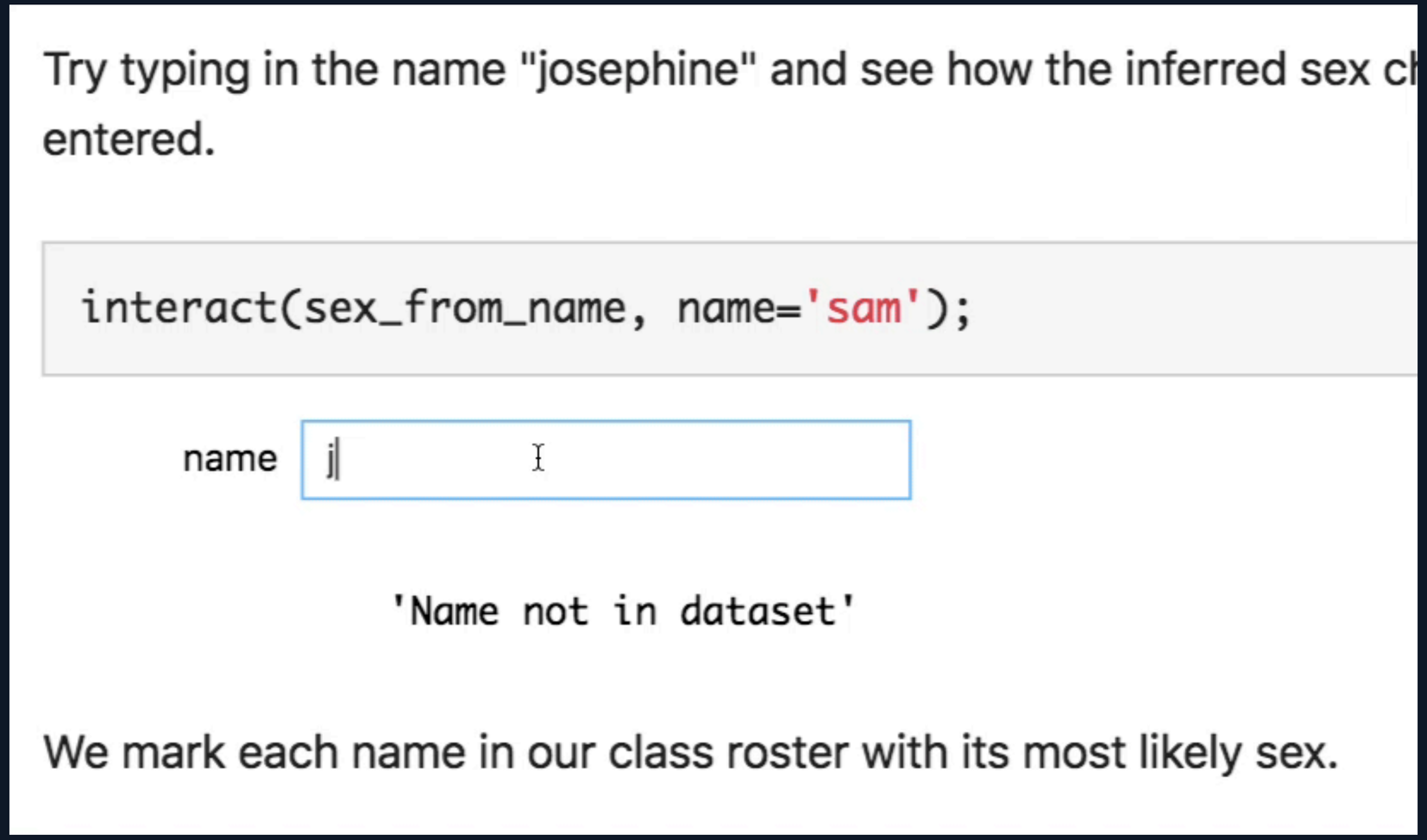
col 1

	pclass	sex	age	sibsp	parch	fare	embarked	class
100	1	female	16.0	0	0	86.5000	S	First
101	1	male	18.0	1	0	108.9000	C	First
102	1	male	36.0	0	0	26.2875	S	First
103	1	male	47.0	0	0	34.0208	S	First
104	2	female	34.0	0	0	10.5000	S	Second

```
(182 rows, 15 columns) total
```

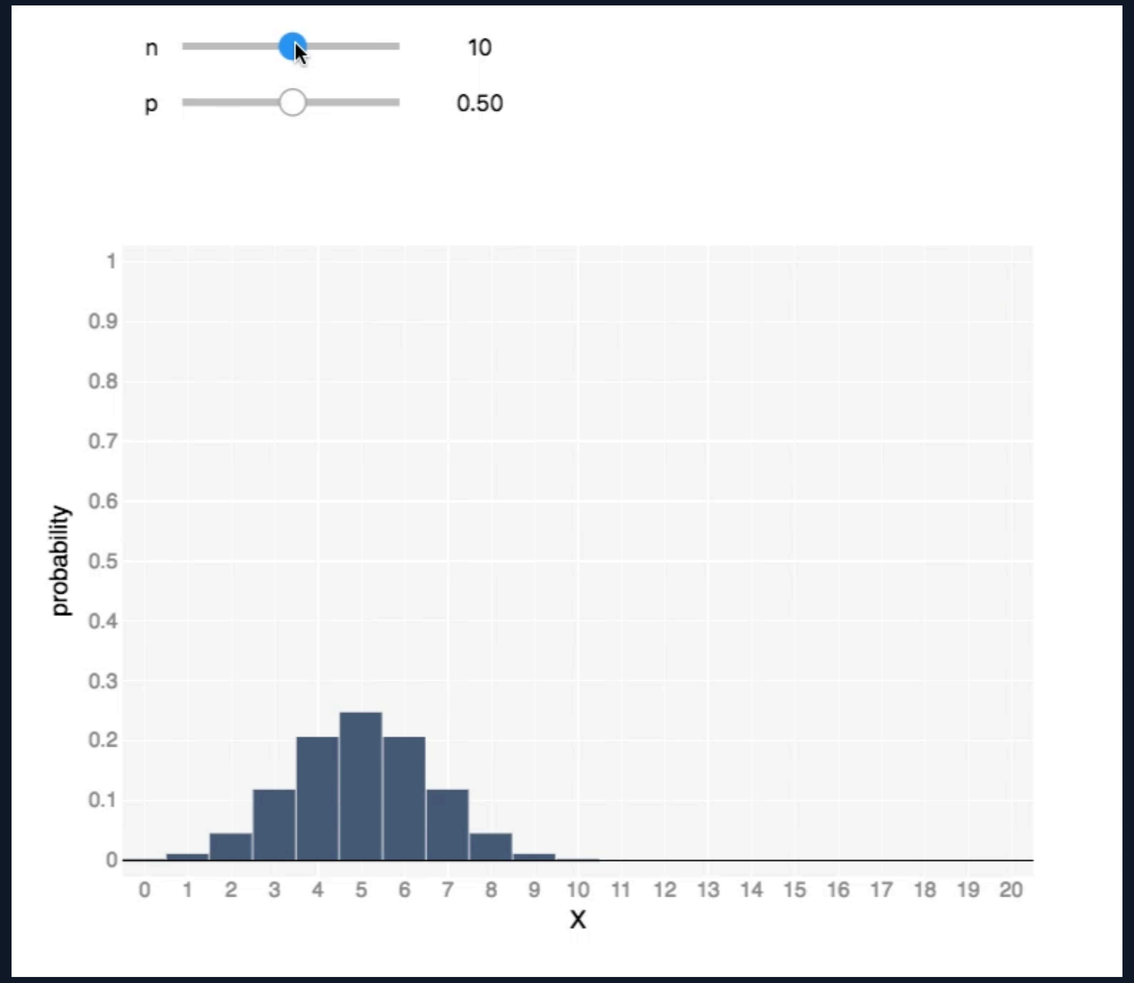
http://www.textbook.ds100.org/ch/06/viz_quantitative.html

Calling a Python Function



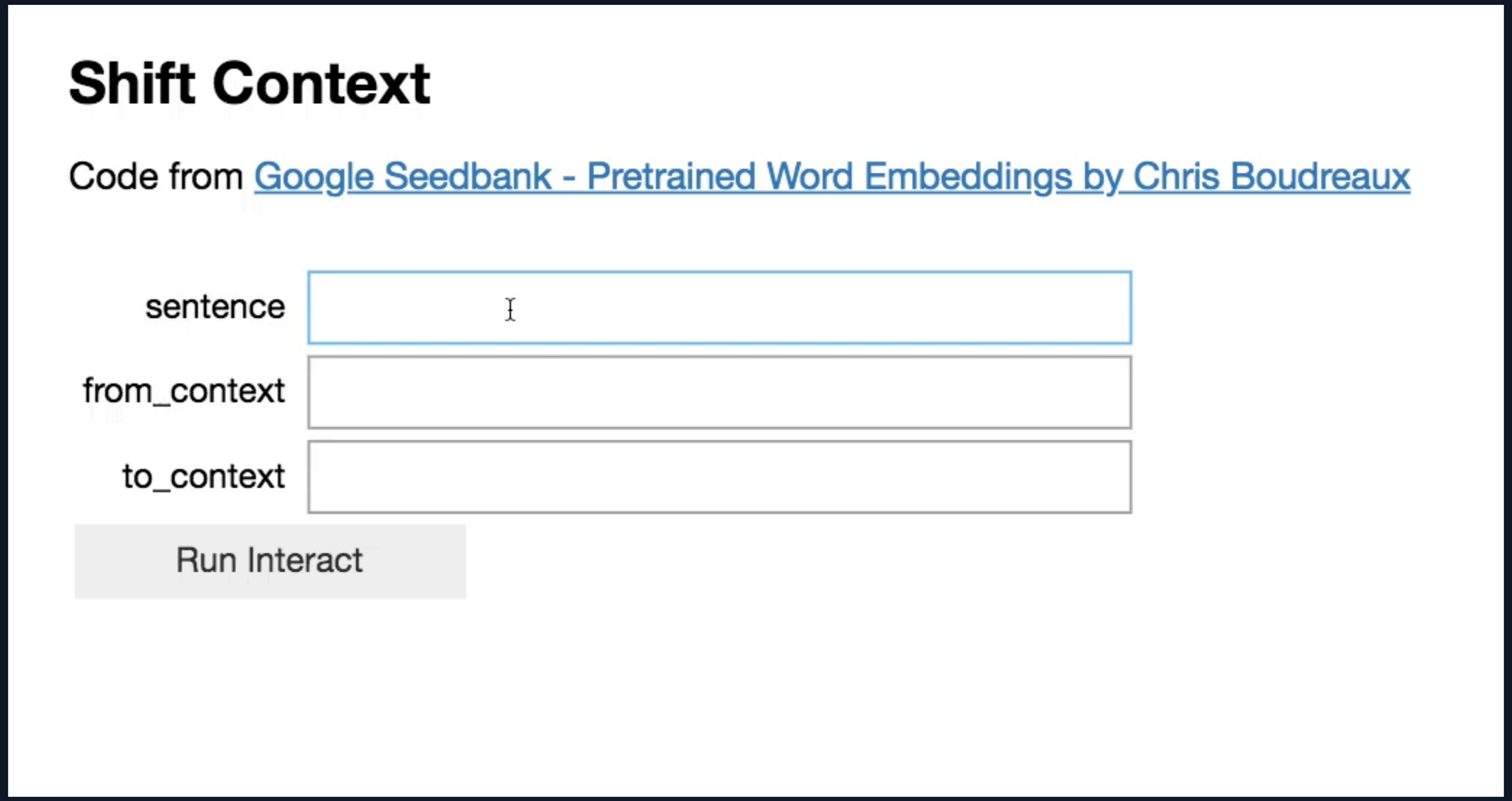
http://www.textbook.ds100.org/ch/01/lifecycle_students_3.html

Displaying an Interactive Plot



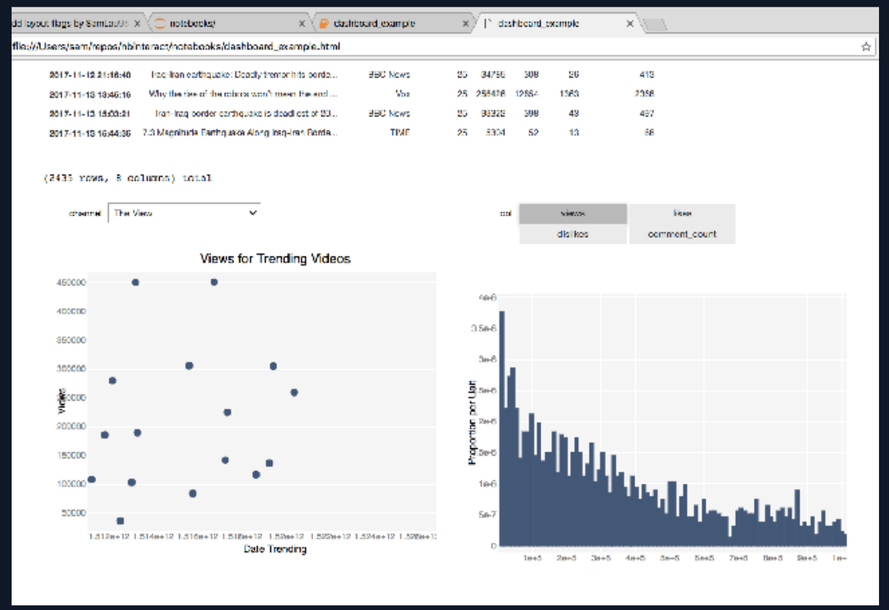
https://www.nbinteract.com/examples/examples_probability_distribution_plots.html

Demo a Library

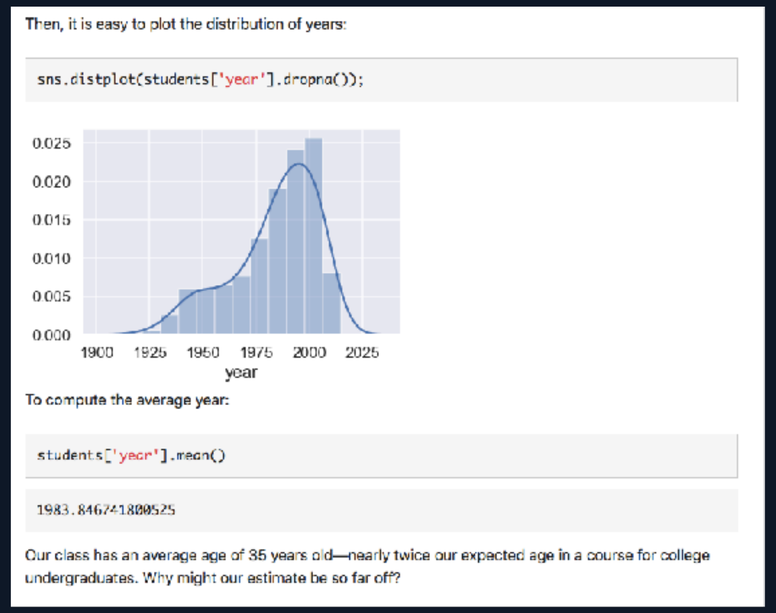


https://ethically.ai/words-embedding/

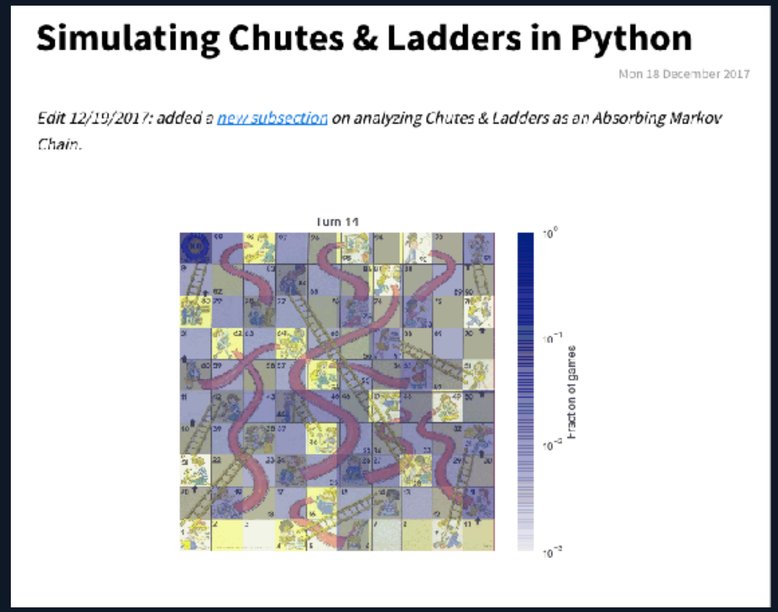
What are your use cases?



Dashboard?



Book?

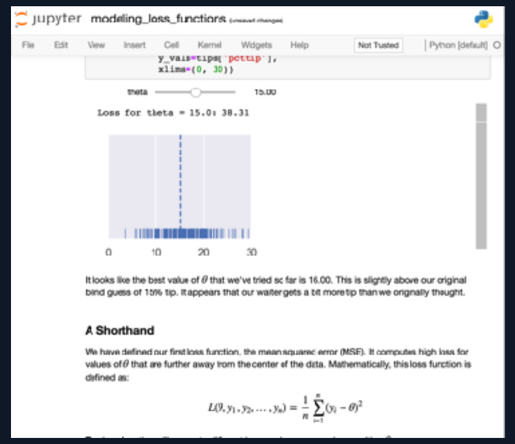


Blog post?

Related Projects

Name	Description	URL
Data 8 Textbook	Intro to data science	http://inferentialthinking.com/
Data 100 Textbook	Upper-division data science	http://www.textbook.ds100.org/
nbconvert	Converts notebooks	https://nbconvert.readthedocs.io/
ipywidgets	Interactive widgets	https://ipywidgets.readthedocs.io/en/stable/
Binder	Notebook servers	https://mybinder.org/
thebelab	Embed notebook cells in HTML	https://github.com/minrk/thebelab

nbinteract creates interactive web pages from notebooks




```
$ nbinteract modeling.ipynb
```

Docs: http://nbinteract.com/

GitHub: https://github.com/SamLau95/nbinteract

Chat room: https://gitter.im/nbinteract/Lobby/

Thanks!