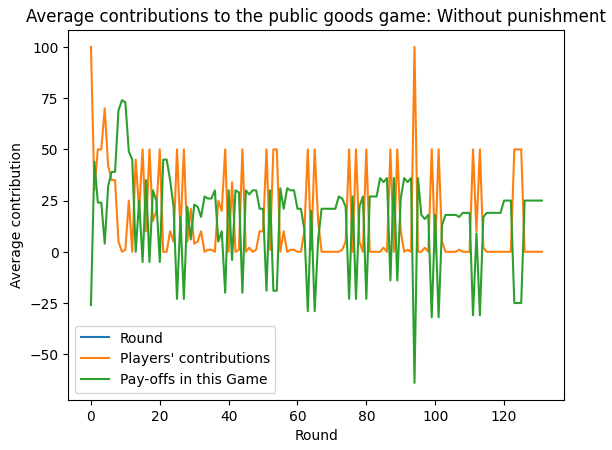
# **Empirical Project 1: Working in Python**

## **Empirical Project 1 Working in Python**

To [download the code chunks](https://tinyco.re/9501186) used in this project, right-click on the download link and select ‘Save Link As…’. You’ll need to save the code download to your working directory and open it in Python.

## **Part 2.1 Collecting Data by Playing**

#### Make a line chart with average contribution as the vertical axis variable, and period (from 1 to 10) on the horizontal axis. Describe how average contributions have changed over the course of the game.



### **Python walk-through 2.1 Plotting a line chart with multiple variables**

Use the data from your own experiment to answer Question 1. As an example, we will use the data for the first three cities of the dataset that will be introduced in Part 2.2.

from google.colab import files

uploaded = files.upload()

df = pd.read\_excel("base\_juego\_taller4O.xlsx")

# Mostrar las primeras filas del DataFrame

df.head()

Now we need to plot the data. Note that, with data in ‘wide’ format (one column per city) and with an index, simply calling **.**plot on a pandas dataframe will create a matplotlib line chart. We could also use the lets\_plot package to make this kind of chart, but it expects data in ‘tidy’ or ‘long’ format—and for that, we would have to reshape the data so that the city names were values in a single column called ‘city’ or similar. Let’s just use matplotlib for now.

*# Plot the data*

fig**,** ax **=** plt**.**subplots**()**

df**.**plot**(**ax**=**ax**)**

ax**.**set\_title**(**"Average contributions to the public goods game: Without punishment"**)**

ax**.**set\_ylabel**(**"Average contribution"**)**

ax**.**set\_xlabel**(**"Round"**);**

#### Compare your line chart with Figure 3 of Herrmann et al. (2008).**1** Comment on any similarities or differences between the results (for example, the amount contributed at the start and end, or the change in average contributions over the course of the game).

R// En la Figura 3 de Herrmann et al. (2008), las contribuciones disminuyen gradualmente a lo largo de los 10 períodos en la mayoría de las ciudades mientras que en el segundo gráfico (sin castigo), las contribuciones fluctúan fuertemente en las primeras rondas y luego parecen estabilizarse en niveles más bajos.

#### Can you think of any reasons why your results are similar to (or different from) those in Figure 3? You may find it helpful to read the ‘Experiments’ section of the [Herrmann et al. (2008) study](https://tinyco.re/3969457) for a more detailed description of how the experiments were conducted.

R// Los resultados muestran una tendencia similar en la disminución de contribuciones, pero con más variabilidad en el gráfico sin castigo. Esto podría explicarse por diferencias en el control experimental, el número de participantes o el contexto cultural de los jugadores.

## **Part 2.1 Describing the data**

We will now use the data used in Figures 2A and 3 of Herrmann et al. (2008), and evaluate the effect of the punishment option on average contributions. Rather than compare two charts showing all of the data from each experiment, as the authors of the study did, we will use summary measures to compare the data, and show the data from both experiments (with and without punishment) on the same chart.

First, download and save [the data](https://tinyco.re/6011816). The spreadsheet contains two tables:

* The first table shows average contributions in a public goods game without punishment (Figure 3).
* The second table shows average contributions in a public goods game with punishment (Figure 2A).

You can see that in each period (row), the average contribution varies across countries, in other words, there is a distribution of average contributions in each period.

### **Python walk-through 2.2 Importing the datafile into Python**

Both the tables you need are in a single Excel worksheet. Note down the cell ranges of each table, in this case A2:Q12 for the without punishment data and A16:Q26 for the punishment data. We will use this range information to import the data into two dataframes (data\_n and data\_p, respectively).

In the code below, we’ll use the **.**copy method, which we’ll explain more about in a moment.

from google.colab import files

uploaded = files.upload()

data\_np = pd.read\_excel(

    "doing-economics-datafile-working-in-excel-project-2.xlsx",

    usecols="A:Q",

    header=1,

    index\_col="Period",

)

data\_n = data\_np.iloc[:10, :].copy()

data\_p = data\_np.iloc[14:24, :].copy()

When loading the data from Excel, you may see an error message about an ‘unknown extension’. Note that this particular Excel file has some issues that mean pandas will warn you about an ‘unknown extension’: an Excel file is actually a bundle of files tied up to look like one file, and what has happened here is that pandas doesn’t recognise one of the files in the bundle. Despite this issue, we can still import the data we need in the worksheets.

In the code above, we used **.**copy and you may be wondering what it does. When a new object (say, data\_p) is created from an *existing* object (here data\_np), programming languages have a few different options for how to do it. In this case, Python has two options: it could allocate some entirely new memory to store the new variable, data\_p, or it could just create a link to the *existing* bit of memory where some of data\_np is stored.

The two different approaches behave differently. Under the former, changes to data\_p won’t affect data\_np because data\_p gets its own bit of memory and is entirely independent of the existing variable. But in the latter case, any changes to data\_p will also be applied to data\_np! This is because, underneath it all, they’re both ‘pointing’ to the same bit of computer memory. Indeed, that is why variables that do this are sometimes called *pointers*. They’re common to most programming languages and pandas tends to use them by default because they save on memory. This case is just an example of a situation where we don’t want to change data\_np by changing data\_p, so we use the **.**copy method to allocate new memory and avoid creating a pointer.

Let’s see a simple example of how this **.**copy method works:

test\_data **=** **{**

"City A"**:** **[14.1,** **14.1,** **13.7],**

"City B"**:** **[11.0,** **12.6,** **12.1],**

**}**

*# Original dataframe*

test\_df **=** pd**.**DataFrame**.**from\_dict**(**test\_data**)**

*# A copy of the dataframe*

test\_copy **=** test\_df**.**copy**()**

*# A pointer to the dataframe*

test\_pointer **=** test\_df

test\_pointer**.**iloc**[1,** **1]** **=** **99**

Now, even though we only modified test\_pointer, we can look at both the original data frame and the copy that we took earlier:

**print(**"test\_df="**)**

**print(**f"{test\_df}\n"**)**

**print(**"test\_copy="**)**

**print(**f"{test\_copy}\n"**)**

We see that test\_df has changed because test\_pointer pointed to it, but our pure copy, test\_copy, hasn’t changed.

As well as importing the correct data, we’re going to ensure it is of the correct *datatype*. Common datatypes include ‘double’ and ‘integer’ (for numbers), string (for words), and ‘category’ (for variables that take on a fixed number of categories, like ethnicity or educational attainment). We can check the datatypes of the data we just read in using data\_n**.**info**()** (you can do the same for data\_p).

data\_n**.**info**()**

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Copenhagen 10 non-null object

1 Dnipropetrovs’k 10 non-null object

2 Minsk 10 non-null object

3 St. Gallen 10 non-null object

4 Muscat 10 non-null object

5 Samara 10 non-null object

6 Zurich 10 non-null object

7 Boston 10 non-null object

8 Bonn 10 non-null object

9 Chengdu 10 non-null object

10 Seoul 10 non-null object

11 Riyadh 10 non-null object

12 Nottingham 10 non-null object

13 Athens 10 non-null object

14 Istanbul 10 non-null object

15 Melbourne 10 non-null object

dtypes: object(16)

memory usage: 1.3+ KB

All of the columns are of the ‘object’ type, which is Python’s default when it’s not clear which datatype to use.

We have continuous real numbers in the columns of data\_n and data\_p here, so we’ll set the datatypes to be double, which is a datatype used for continuous real numbers.

data\_n **=** data\_n**.**astype**(**"double"**)**

data\_p **=** data\_p**.**astype**(**"double"**)**

You can look at the data either by opening the dataframes from the Environment window or by typing data\_n or data\_p into the interactive Python window.

You can see that in each row, the average contribution varies across countries; in other words, there is a distribution of average contributions in each period.

The **meanmean** A summary statistic for a set of observations, calculated by adding all values in the set and dividing by the number of observations.close⁠ and **variancevariance** A measure of dispersion in a frequency distribution, equal to the mean of the squares of the deviations from the arithmetic mean of the distribution. The variance is used to indicate how ‘spread out’ the data is. A higher variance means that the data is more spread out.Example: The set of numbers 1, 1, 1 has zero variance (no variation), while the set of numbers 1, 1, 999 has a high variance of 221,334 (large spread).close⁠ are two ways to summarize distributions. We will now use these measures, along with other measures (range and standard deviation) to summarize and compare the distribution of contributions in both experiments.

#### Using the data for Figures 2A and 3 of Herrmann et al. (2008):

#### Calculate the mean contribution in each period (row) separately for both experiments.

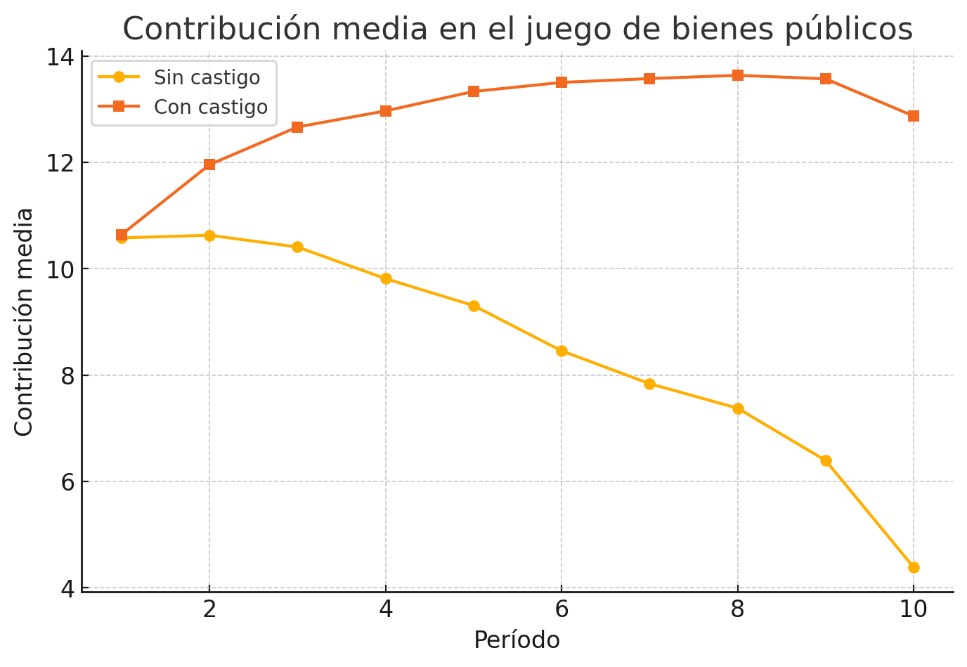
#### Plot a line chart of mean contribution on the vertical axis and time period (from 1 to 10) on the horizontal axis (with a separate line for each experiment). Make sure the lines in the legend are clearly labelled according to the experiment (with punishment or without punishment).

#### Describe any differences and similarities you see in the mean contribution over time in both experiments.

R// a)







1. En ambas se evidencia una contribución inicial relativamente alta y en diferencias por la parte de sin castigo la contribución disminuye progresivamente con cada período, lo que sugiere que los jugadores aportan menos a medida que avanza el juego y con castigo la contribución aumenta en los primeros períodos y luego se mantiene relativamente estable, indicando que la posibilidad de castigo incentiva a los jugadores a mantener su cooperación.

### **Python walk-through 2.3 Calculating the mean using the .mean() or the agg functions**

We calculate the mean using two different methods to illustrate that there are usually many ways of achieving the same thing. We apply the first method on data\_n, which uses the built-in **.**mean**()** function to calculate the average separately over each column except the first. We use the second method (the agg function) on data\_p.

mean\_n\_c **=** data\_n**.**mean**(**axis**=1)**

mean\_p\_c **=** data\_p**.**agg**(**np**.**mean**,** axis**=1)**

As the name suggests, the agg function applies an aggregation function (the mean function in this case) to all rows or columns in a dataframe. The second input, axis**=1**, applies the specified function to all rows in data\_p, so we are taking the average over cities for each period.

Typing axis**=0** would have calculated column means instead, that is, it would have averaged over periods to produce one value per city (run this code to see for yourself). Type help**(**pd**.**DataFrame**.**agg**)** in your interactive Python window for more details, or see [Python walk-through 2.5](https://www.core-econ.org/doing-economics/book/text/02-06.html#python-walk-through-25-calculating-and-understanding-the-standard-deviation) for further practice.

*Plot the mean contribution*

Now we will produce a line chart showing the mean contributions.

fig**,** ax **=** plt**.**subplots**()**

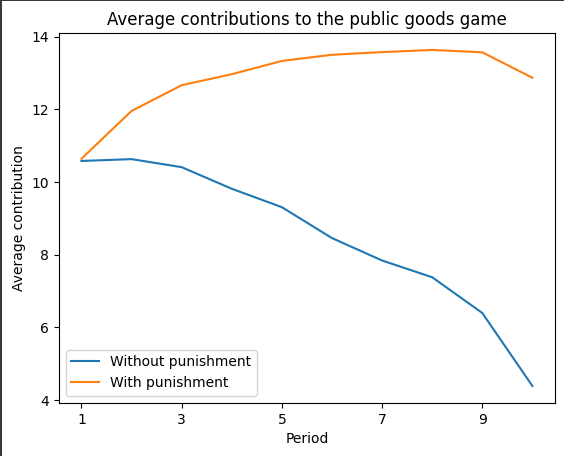
mean\_n\_c**.**plot**(**ax**=**ax**,** label**=**"Without punishment"**)**

mean\_p\_c**.**plot**(**ax**=**ax**,** label**=**"With punishment"**)**

ax**.**set\_title**(**"Average contributions to the public goods game"**)**

ax**.**set\_ylabel**(**"Average contribution"**)**

ax**.**legend**();**



The difference between experiments is stark, as the contributions increase and then stabilise at around 13 in the case where there is punishment, but decrease consistently from around 11 to 4 across the rounds when there is no punishment.

#### Instead of looking at all periods, we can focus on contributions in the first and last period. Plot a column chart showing the mean contribution in the first and last period for both experiments. Your chart should look like [Figure 2.3](https://www.core-econ.org/doing-economics/book/text/02-06.html#figure-2-3).

### **Python walk-through 2.4 Drawing a column chart to compare two groups**

To do this next part, we’re going to use something called a ‘list comprehension’, which is a special kind of loop. Loops are very useful in programming when you have the same task that you want to execute for a sequence of values. You could use a loop to find the squares of the first 10 numbers, for example.

A list comprehension is a way of writing a loop that creates a Python list. The loops it creates tend to be quick to run, too.

As a specific example, let’s say we wanted to add the first name ‘John’ to a list of names. Using a list comprehension, the code would be:

partial\_names\_list **=** **[**"F. Kennedy"**,** "Lennon"**,** "Maynard Keynes"**,** "Wayne"**]**

**[**"John " **+** name **for** name **in** partial\_names\_list**]**

The second line shows the syntax: square bracket (which usually signifies a list), then an operation (here "John" **+** name), and then **for** name\_of\_thing **in** name\_of\_list (replace name\_of\_thing and name\_of\_list with the thing you would like to apply the loop to, and your list name).

To make a column chart, we will use the **.**plot**.**bar**()** function. We first extract the four data points we need (Periods 1 and 10, with and without punishment) and place them into another dataframe (called compare\_grps).

*# Create new dataframe with bars in*

compare\_grps **=** pd**.**DataFrame**(**

**[**mean\_n\_c**.**loc**[[1,** **10]],** mean\_p\_c**.**loc**[[1,** **10]]],**

index**=[**"Without punishment"**,** "With punishment"**],**

**)**

*# Rename columns to have 'round' in them*

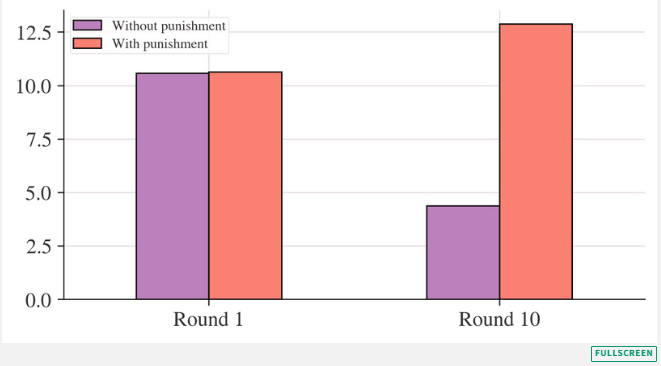
compare\_grps**.**columns **=** **[**"Round " **+** str**(**i**)** **for** i **in** compare\_grps**.**columns**]**

*# Swap the column and index variables around with the transpose function, ready for plotting (.T is transpose)*

compare\_grps **=** compare\_grps**.**T

*# Make a bar chart*

compare\_grps**.**plot**.**bar**(**rot**=0);**



The mean is one useful measure of the ‘middle’ of a distribution, but is not a complete description of what our data looks like. We also need to know how ‘spread out’ the data is in order to get a clearer picture and make comparisons between distributions. The **variance**⁠ is one way to measure spread: the higher the variance, the more spread out the data is.

A similar measure is **standard deviationstandard deviation** A measure of dispersion in a frequency distribution, equal to the square root of the variance. The standard deviation has a similar interpretation to the variance. A larger standard deviation means that the data is more spread out.Example: The set of numbers 1, 1, 1 has a standard deviation of zero (no variation or spread), while the set of numbers 1, 1, 999 has a standard deviation of 46.7 (large spread).close⁠, which is the square root of the variance and is commonly used because there is a handy rule of thumb for large datasets, which is that most of the data (95%, if there are many observations) will be less than two standard deviations away from the mean.

#### Using the data for Figures 2A and 3 of Herrmann et al. (2008):

#### Calculate the standard deviation for Periods 1 and 10 separately, for both experiments. Does the rule of thumb apply? (In other words, are most values within two standard deviations of the mean?)

#### As shown in [Figure 2.3](https://www.core-econ.org/doing-economics/book/text/02-06.html#figure-2-3), the mean contribution for both experiments was 10.6 in Period 1. With reference to your standard deviation calculations, explain whether this means that the two sets of data are the same.

### **Python walk-through 2.5 Calculating and understanding standard deviation**

In order to calculate these standard deviations and variances, we will use the agg function, which we introduced in Python walk-through 2.3. As we saw, agg is a command that asks pandas to aggregate a set of rows or columns of the dataframe using a particular aggregation function. The basic structure is as follows: dataframe\_name**.**agg**([**function1**,** function2**,** **...],** rows**/**columns**)**. So to calculate the variances and more, we use the following command:

n\_c **=** data\_n**.**agg**([**"std"**,** "var"**,** "mean"**],** **1)**

n\_c

Here we take data\_n and apply the "var" and "std" functions to each row (recall that the second input **1** does this; **0** would indicate columns). Note that the index column, which contains the period numbers, is automatically excluded from the calculation. The result is saved as a new variable called n\_c.

We then apply the same principle to the data\_p dataframe.

p\_c **=** data\_p**.**agg**([**"std"**,** "var"**,** "mean"**],** **1)**

*Aside:* In the next chart, we will use another kind of loop. The syntax for this one is **for** 'thing' **in** list of things, then a colon (**:**), then an indented operation that uses thing.

To determine whether 95% of the observations fall within two standard deviations of the mean, we can use a line chart. As we have 16 countries in every period, we would expect about one observation (0.05 × 16 = 0.8) to fall outside this interval.

fig**,** ax **=** plt**.**subplots**()**

n\_c**[**"mean"**].**plot**(**ax**=**ax**,** label**=**"mean"**)**

*# mean + 2 standard deviations*

**(**n\_c**[**"mean"**]** **+** **2** **\*** n\_c**[**"std"**]).**plot**(**ax**=**ax**,** ylim**=(0,** None**),** color**=**"red"**,** label**=**"±2 s.d."**)**

*# mean - 2 standard deviations*

**(**n\_c**[**"mean"**]** **-** **2** **\*** n\_c**[**"std"**]).**plot**(**ax**=**ax**,** ylim**=(0,** None**),** color**=**"red"**,** label**=**""**)**

**for** i **in** range**(**len**(**data\_n**.**columns**)):**

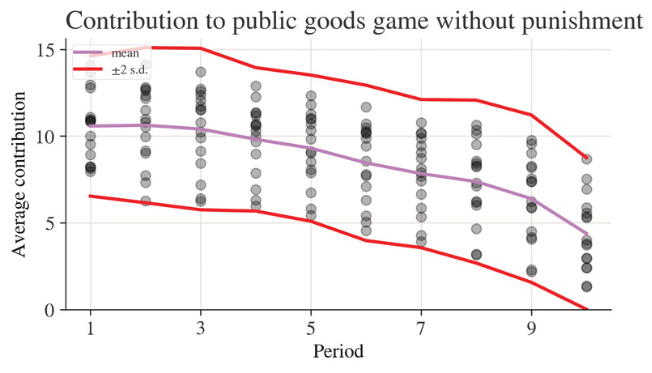
ax**.**scatter**(**x**=**data\_n**.**index**,** y**=**data\_n**.**iloc**[:,** i**],** color**=**"k"**,** alpha**=0.3)**

ax**.**legend**()**

ax**.**set\_ylabel**(**"Average contribution"**)**

ax**.**set\_title**(**"Contribution to public goods game without punishment"**)**

plt**.**show**();**



None of the observations fall outside the mean ± two standard deviations interval for the public goods game without punishment. Let’s plot the equivalent chart for the version with punishment.

fig**,** ax **=** plt**.**subplots**()**

p\_c**[**"mean"**].**plot**(**ax**=**ax**,** label**=**"mean"**)**

*# mean + 2 sd*

**(**p\_c**[**"mean"**]** **+** **2** **\*** p\_c**[**"std"**]).**plot**(**ax**=**ax**,** ylim**=(0,** None**),** color**=**"red"**,** label**=**"±2 s.d."**)**

*# mean - 2 sd*

**(**p\_c**[**"mean"**]** **-** **2** **\*** p\_c**[**"std"**]).**plot**(**ax**=**ax**,** ylim**=(0,** None**),** color**=**"red"**,** label**=**""**)**

**for** i **in** range**(**len**(**data\_p**.**columns**)):**

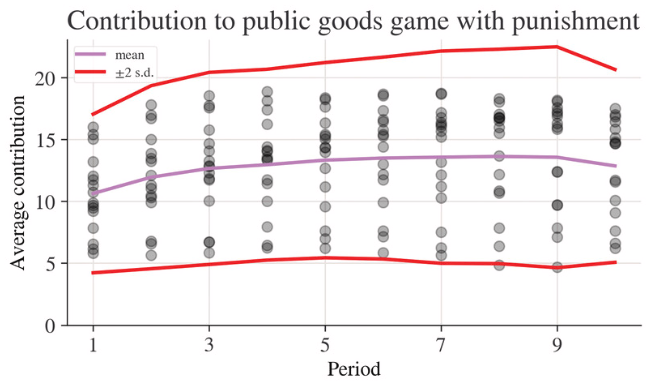
ax**.**scatter**(**x**=**data\_p**.**index**,** y**=**data\_p**.**iloc**[:,** i**],** color**=**"k"**,** alpha**=0.3)**

ax**.**legend**()**

ax**.**set\_ylabel**(**"Average contribution"**)**

ax**.**set\_title**(**"Contribution to public goods game with punishment"**)**

plt**.**show**();**



Here, we only have one observation outside the interval (in Period 8). In that aspect the two experiments look similar. However, from comparing these two charts, the game with punishment displays a greater variation of responses than the game without punishment. In other words, there is a larger standard deviation and variance for the observations coming from the game with punishment.

Another measure of spread is the **rangerange** The interval formed by the smallest (minimum) and the largest (maximum) value of a particular variable. The range shows the two most extreme values in the distribution, and can be used to check whether there are any outliers in the data. (Outliers are a few observations in the data that are very different from the rest of the observations.)close⁠, which is the interval formed by the smallest (minimum) and the largest (maximum) values of a particular variable. For example, we might say that the number of periods in the public goods experiment ranges from 1 to 10. Once we know the most extreme values in our dataset, we have a better picture of what our data looks like.

#### Calculate the maximum and minimum value for Periods 1 and 10 separately, for both experiments.

### **Python walk-through 2.6 Finding the minimum, maximum, and range of a variable**

We’re now going to see one of our first *functions*. A function takes inputs, does some operations on them, and returns outputs.

You can imagine functions as vending machines: for them to work you need some inputs (money, and a choice of snack or drink), then an operation happens (your drink or snack is dropped into the tray), and finally there is an output (your drink or snack as you grab it).

Functions are incredibly useful in programming because they are separate units that can be tested in isolation, re-used, and given helpful ‘dressing’ (such as information on how they work) that make code more readable.

To calculate the range for both experiments and for all periods, we will use an apply method in combination with the max and min methods that apply to a column or row. We’ll also use a *lambda function* to bring these all together. In our case, it’s going to look like this:

data\_p**.**apply**(lambda** x**:** x**.**max**()** **-** x**.**min**(),** axis**=1)**

Period

**1** **10.199675**

**2** **12.185065**

**3** **12.689935**

**4** **12.625000**

**5** **12.140375**

**6** **12.827541**

**7** **13.098931**

**8** **13.482621**

**9** **13.496754**

**10** **11.307360**

dtype**:** float64

This lambda function tells Python to take the difference between the maximum and minimum of each row.

A lambda function is an idea in programming (and mathematics) that has a long and interesting history. You don’t need to know all that, but it is instructive to look at a more general example of a lambda function:

*# A lambda function accepting three inputs, a, b, and c, and calculating the sum of the squares*

test\_function **=** **lambda** a**,** b**,** c**:** a**\*\*2** **+** b**\*\*2** **+** c**\*\*2**

*# Now we apply the function by handing over (in parenthesis) the following inputs: a=3, b=4 and c=5*

test\_function**(3,** **4,** **5)**

Above, we defined a lambda function that looked like **lambda** x**:** x**.**max**()** **-** x**.**min**()**. It accepts one input, x (which could be a row or column), and returns the range of x. Because making code reusable is good programming practice, we will define this function and give it a name using a separate line of code like this:

range\_function **=** **lambda** x**:** x**.**max**()** **-** x**.**min**()**

When we call data\_p**.**apply**(**range\_function**,** axis**=1)**, the following will happen: data\_p contains the experimental data (with punishment). We will apply the range\_function to that data. As data\_p has two dimensions, we also need to let Python know over which dimension it should calculate the minimum and maximum. The axis**=1** option in the apply function tells the apply function that it should apply the range\_function over rows rather than columns (to get columns, it would be axis**=0**, which is also the default if you don’t specify the axis keyword argument).

range\_function **=** **lambda** x**:** x**.**max**()** **-** x**.**min**()**

range\_p **=** data\_p**.**apply**(**range\_function**,** axis**=1)**

range\_n **=** data\_n**.**apply**(**range\_function**,** axis**=1)**

Let’s create a chart of the ranges for both experiments for all periods in order to compare them.

fig**,** ax **=** plt**.**subplots**()**

range\_p**.**plot**(**ax**=**ax**,** label**=**"With punishment"**)**

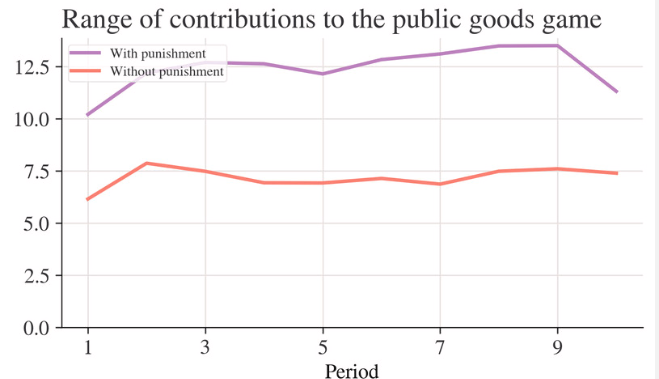
range\_n**.**plot**(**ax**=**ax**,** label**=**"Without punishment"**)**

ax**.**set\_ylim**(0,** None**)**

ax**.**legend**()**

ax**.**set\_title**(**"Range of contributions to the public goods game"**)**

plt**.**show**();**



This chart confirms what we found in Python walk-through 2.5, which is that there is a greater spread (variation) of contributions in the game with punishment.

#### A concise way to describe the data is in a summary table. With just four numbers (mean, standard deviation, minimum value, maximum value), we can get a general idea of what the data looks like.

#### Create a table of summary statistics that displays mean, variance, standard deviation, minimum, maximum and range for Periods 1 and 10 and for both experiments.

#### Comment on any similarities and differences in the distributions, both across time and across experiments.

### **Python walk-through 2.7 Creating a table of summary statistics**

We have already done most of the work for creating this summary table in Python walk-through 2.6. Since we also want to display the minimum and maximum values, we should create these too. And it’s convenient to add in std and mean using the same syntax (even though we created a separate mean earlier), so we have all the information in one place. We’ll call our new summary statistics summ\_p and summ\_n.

funcs\_to\_apply **=** **[**range\_function**,** "max"**,** "min"**,** "std"**,** "mean"**]**

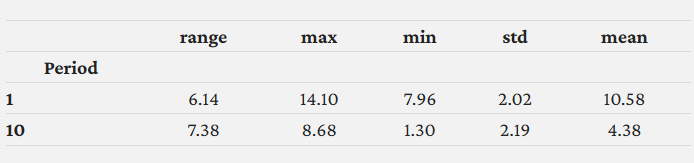
summ\_p **=** data\_p**.**apply**(**funcs\_to\_apply**,** axis**=1).**rename**(**columns**={**"<lambda>"**:** "range"**})**

summ\_n **=** data\_n**.**apply**(**funcs\_to\_apply**,** axis**=1).**rename**(**columns**={**"<lambda>"**:** "range"**})**

Note that as well as applying all of the functions in the list funcs\_to\_apply, we also renamed the first function using the rename method. Because the range isn’t a built-in aggregation function and we defined it, it is automatically given a column name—and because the range function we supplied is a lambda function, the name it gets is "<lambda>". Using rename**(**columns**=**, we change this name to "range" using a dictionary object (**{** **:** **}**) that maps the old name to the new name.

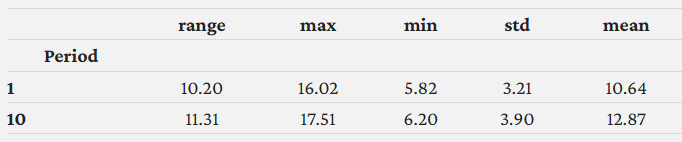
Now we display the summary statistics in a table. We use the round method, which reduces the number of digits displayed after the decimal point (**2** in our case) and makes the table easier to read. We’re only interested in periods 1 and 10, so we pass a list, **[1,** **10]**, to the **.**loc selector in the first position (which corresponds to rows and the index). We want all columns, so we pass **:** to the second position of the **.**loc selector.

summ\_n**.**loc**[[1,** **10],** **:].**round**(2)**



Now we do the same for the version with punishment.

summ\_p**.**loc**[[1,** **10],** **:].**round**(2)**



## **Part 2.3 How did changing the rules of the game affect behaviour?**

The punishment option was introduced into the public goods game in order to see whether it could help sustain contributions, compared to the game without a punishment option. We will now use a calculation called a p-value to compare the results from both experiments more formally.

By comparing the results in Period 10 of both experiments, we can see that the mean contribution in the experiment with punishment is 8.5 units higher than in the experiment without punishment (see [Figure 2.6](https://www.core-econ.org/doing-economics/book/text/02-06.html#figure-2-6) in Part 2.2). Is it more likely that this behaviour is due to chance, or is it more likely to be due to the difference in experimental conditions?

#### You can conduct another experiment to understand why we might see differences in behaviour that are due to chance.

#### First, flip a coin six times, using one hand only, and record the results (for example, Heads, Heads, Tails, etc.). Then, using the same hand, flip a coin six times and record the results again.

#### Compare the outcomes from Question 1(a). Did you get the same number of heads in both cases? Even if you did, was the sequence of the outcomes (for example, Heads, Tails, Tails …) the same in both cases?

R//

a) Los resultados fueron:

* **Moneda 1:** cruz, cara, cruz, cruz, cruz, cruz
* **Moneda 2:** cara, cruz, cruz, cruz, cara, cara

b)

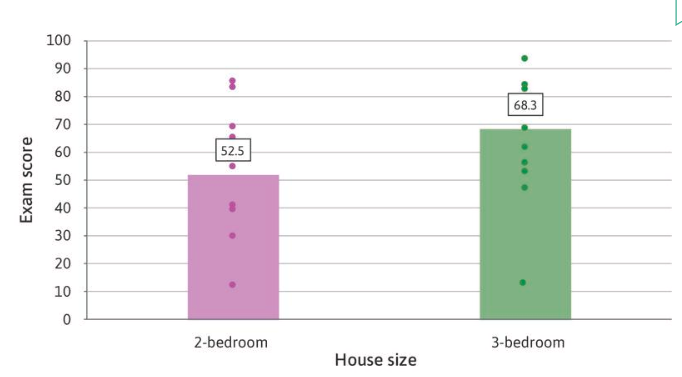
En la cantidad de caras en el primer caso fue de 1 mientras que en el segundo caso fue de 3 veces. Esto muestra que incluso con el mismo proceso, los resultados pueden diferir debido al azar.

The important point to note is that even when we conduct experiments under the same controlled conditions, due to an element of randomness, we may not observe the exact same behavior each time we do the experiment.

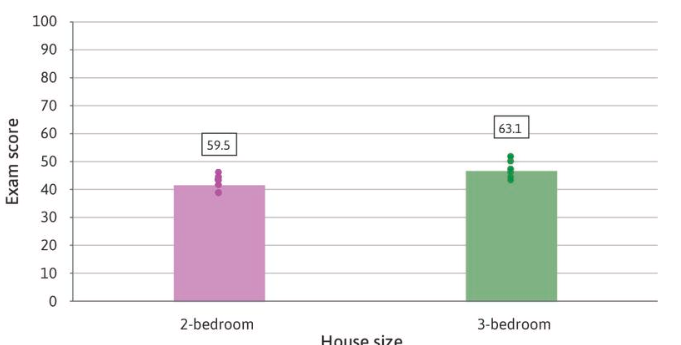
Randomness arises because the statistical analysis is conducted on a sample of data (for example, a small group of people from the entire population), and the sample we observe is only one of many possible samples. Whatever differences we calculate between two samples would almost certainly change if we had observed another pair of samples. Importantly, economists aren’t really interested in whether two samples are actually different, but rather whether the underlying populations, from which the samples were drawn, differ in the characteristics we are interested in (for example, age, income, contributions to the public good). And this is the challenge faced by the empirical economist.

When we are interested in whether a treatment works—in this case, whether having the punishment option makes a difference in how much people contribute to the public good. So, we want a way to check whether any observed differences could just be due to sample variation.

The size of the difference alone cannot tell us whether it might just be due to chance. Even if the observed difference seems large, it could be small relative to how much the data vary. Figures 2.7 and 2.8 show the mean exam score of two groups of high school students and the size of house in which they live (represented by the height of the columns, and reported in the boxes above the columns), with the dots representing the underlying data. Figure 2.7 shows a relatively large difference in means that could have arisen by chance because the data is widely spread out (the standard deviation is large), while Figure 2.8 shows a relatively small difference that looks unlikely to be due to chance because the data is tightly clustered together (the standard deviation is very small). Note that we are looking at two distinct questions here: first, is there a large or small difference in exam score associated with the size of house of the student and second, is that difference likely to have arisen by chance. A social scientist is interested in the answer to both questions. If the difference is large but could easily have occurred by chance or if the difference is very small and unlikely to have occurred by chance, then the results are not suggestive of an important relationship between size of house and exam grade.



**Figure 2.7**An example of a large difference in means that is likely to have happened by chance.



**Figure 2.8**An example of a small difference in means that is unlikely to have happened by chance.

To help us decide, we consider the hypothesis that the difference occurred by chance – in other words, we start by hypothesizing that house size does not matter for exam scores. Then we ask how likely it is that we would observe differences at least as extreme as those we actually observe in our sample groups, *assuming* that our hypothesis is true. The answer to this question is called a **p-value**⁠. The smaller the p-value, the less likely that we would observe differences at least as extreme as those we did, given our hypothesis. So the smaller this p-value, the smaller our confidence will be in the hypothesis that in the population house size does not matter for exam grades.

Notice that the p-value is not the probability that the hypothesis is correct – the data cannot tell us that probability. It is the probability that we would find a difference as big as the one we have observed *if* the hypothesis were correct.

We can estimate the p-value from the data, using the sample means and sample deviations. It is calculated by comparing the difference in the means with the amount of variation in the data as measured by the standard deviations. This is a well-established method, although some other statistical assumptions, which we do not discuss, are required to ensure that it gives a good estimate.

When we look at the data in Figure 2.7, we cannot be absolutely certain that there really is a link between house size and exam scores. But if the p-value for the difference in means is very small (for example, 0.02) then we know that there would only be a 2% probability of seeing differences at least as extreme as those we did observe in the sample, given our hypothesis that in the population there was no relationship between house size and exam scores.

We will calculate the p-value and use it to assess how likely it is that the differences we observe are due to chance.

#### Using the data for Figures 2A and 3:

#### Use the ttest function to calculate the p-value for the difference in means in Period 1 (with and without punishment).

#### What does this p-value tell us about the difference in means in Period 1?

R//

import pandas as pd

from scipy.stats import ttest\_ind

# Cargar datos

df = pd.read\_excel("doing-economics-datafile-working-in-excel-project-2.xlsx", sheet\_name="Public goods contributions").dropna()

# Extraer datos del Período 1

period\_1\_no\_punishment = df.iloc[1, 1:].astype(float)

period\_1\_punishment = df.iloc[15, 1:].astype(float)

# Prueba t

t\_stat, p\_value = ttest\_ind(period\_1\_no\_punishment, period\_1\_punishment, equal\_var=False)

# Mostrar resultados

print(f"t: {t\_stat}, p: {p\_value}")

### **Python walk-through 2.8Calculating the p-value for the difference in means**

We need to extract the observations in Period 1 for the data for with and without punishment, and then feed the observations into a function that performs a t-test. We’ll use the statistics package pingouin for this, which you will need to install on the command line using pip install pingouin. Once installed, import it using **import** pingouin **as** pg, just like we did at the start of the project.

*Tip:* you can open up the command line, also known as the *terminal* or *command prompt*, in order to install packages in multiple ways. If you’re working within Visual Studio Code use the ⌃ + \` keyboard shortcut (Mac) or CTRL + \` (Windows and Linux), or click ‘View > Terminal’. If you want to open up the command line independently of Visual Studio Code, search for ‘Terminal’ on Mac and Linux, and ‘Anaconda Prompt’ on Windows.

pingouin’s t-test function is called ttest. The ttest function is extremely flexible: if you input two variables (x and y) as shown below, it will automatically test whether the difference in means is likely to be due to chance or not (formally speaking, it tests the null hypothesis that the means of both variables are equal).

Note that the ttest function will only accept one series of data, not multiple data series. By subsetting (iloc**[1,** **:]**), we are passing in the 0th row (the first period) for all columns (cities).

pg**.**ttest**(**x**=**data\_n**.**iloc**[0,** **:],** y**=**data\_p**.**iloc**[0,** **:])**

Note that as well as the t-statistic (T), the p-value (p**-**val), the degrees of freedom (dof), the alternative hypothesis (two**-**sided) and the confidence interval (CI95**%**), we get some other variables that help us put the main results into context.

This result delivers a p-value of 0.9496. This means it is very likely that the assumption that there are no differences in the populations is likely to be true (formally speaking, we cannot reject the null hypothesis).

The ttest function automatically assumes that both variables were generated by different groups of people. When calculating the p-value, it assumes that the observed differences are partly due to some variation in characteristics between these two groups, and not just the differences in experimental conditions. However, in this case, the same groups of people did both experiments, so there will not be any variation in characteristics between the groups. When calculating the p-value, we account for this fact with the paired**=**True option.

pg**.**ttest**(**x**=**data\_n**.**iloc**[0,** **:],** y**=**data\_p**.**iloc**[0,** **:],** paired**=**True**)**

The p-value becomes smaller as we can attribute more of the differences to the ‘with punishment’ treatment, but the p-value is still very large (0.8828), so we still conclude that the differences in Period 1 are likely to be due to chance.

#### Using the data for Period 10:

#### Use the ttest function to calculate the p-value for the difference in means in Period 10 (with and without punishment).

#### What does this p-value tell us about the relationship between punishment, and behaviour in the public goods game?

#### With reference to [Figure 2.7](https://www.core-econ.org/doing-economics/book/text/02-06.html#figure-2-7) and [Figure 2.8](https://www.core-econ.org/doing-economics/book/text/02-06.html#figure-2-8), explain why we cannot use the size of the difference to directly conclude whether the difference could be due to chance.

R//

a)

import pandas as pd

from scipy.stats import ttest\_ind

# Cargar datos

df = pd.read\_excel("doing-economics-datafile-working-in-excel-project-2.xlsx", sheet\_name="Public goods contributions").dropna()

# Extraer datos del Período 10

period\_10\_data = df.iloc[df[df.iloc[:, 0] == 10].index[0], 1:].astype(float)

period\_10\_punishment\_data = df.iloc[24, 1:].astype(float) # Datos con castigo

# Prueba t

t\_stat, p\_value = ttest\_ind(period\_10\_data, period\_10\_punishment\_data, equal\_var=False)

# Mostrar resultados

print(f"t: {t\_stat}, p: {p\_value}")

b) El resultado de p es (8.76 × 10⁻⁸) lo que indica que la diferencia en las contribuciones entre el grupo con castigo y sin castigo en el Período 10 es estadísticamente significativa. Esto sugiere que la introducción del castigo tuvo un efecto importante en el comportamiento de los participantes en el juego de los bienes públicos, probablemente incentivando una mayor cooperación.

* Por Variabilidad de datos en la Figura 2.7, la diferencia en medias es grande (52.5 vs. 68.3), pero la dispersión de los datos (representada por los puntos) también parece amplia. Esto sugiere que la diferencia podría haber surgido por azar debido a una alta variabilidad.
* La significancia estadística depende no solo del tamaño de la diferencia sino también del número de observaciones y la dispersión de los datos.
* Si la muestra es pequeña, incluso una gran diferencia en medias podría ser simplemente una fluctuación aleatoria.

En conclusión, no podemos usar únicamente el tamaño de la diferencia de medias para decidir si una diferencia es significativa, sino que debemos tener más factores en cuenta para una mayor precisión.

An important point to note is that calculating p-values may not tell us anything about causation. The example of house size and exam scores shown in Figure 2.8, gives us evidence that some kind of relationship between house size and exam scores is very likely. However, we would not conclude that building an extra room automatically makes someone smarter. P-values cannot help us detect these **spurious correlationsspurious correlation** A strong linear association between two variables that does not result from any direct relationship, but instead may be due to coincidence or to another unseen factor.close⁠.

However, calculating p-values for experimental evidence can help us determine whether there is a causal link between two variables. If we conduct an experiment and find a difference in outcomes with a low p-value, then we may conclude that the change in experimental conditions is likely to have caused the difference.

#### Refer to the results from the public goods games.

#### Which characteristics of the experimental setting make it likely that the with punishment option was the cause of the change in behaviour?

#### Using [Figure 2.6](https://www.core-econ.org/doing-economics/book/text/02-06.html#figure-2-6), explain why we need to compare the two groups in Period 1 in order to conclude that there is a causal link between the with punishment option and behaviour in the game.

R//

* + 1. Características del entorno experimental que hacen probable que el castigo sea la causa del cambio de comportamiento:
* **Asignación aleatoria**: Los participantes se asignan aleatoriamente a cada grupo, reduciendo sesgos externos.
* **Reglas claras y consistentes:** Todos los jugadores conocen las reglas del juego y la posibilidad de ser castigados en una de las condiciones.
  + 1. **Importancia de comparar los dos grupos en el Período 1:**
       - En la Figura 2.6, se observa que en el **Período 1** ambos grupos tienen contribuciones similares lo que nos indica que las diferencias en períodos posteriores no se deben a factores iniciales preexistentes, sino al efecto del castigo.

Experiments can be useful for identifying causal links. However, if people’s behaviour in experimental conditions were different from their behaviour in the real world, our results would not be applicable anywhere outside the experiment.

#### Discuss some limitations of experiments, and suggest some ways to address (or partially address) them. (You may find pages 158–171 of the paper [‘What do laboratory experiments measuring social preferences reveal about the real world?’](https://tinyco.re/2593698) helpful, as well as the discussion on free riding and altruism in [Section 2.6](https://tinyco.re/7103820) of Economy, Society, and Public Policy.)

R//

1. Una de las limitaciones puede ser que los participantes saben que están en un entorno experimental, lo que puede afectar su comportamiento entonces una posible solución podría ser usar incentivos monetarios que representen mejor la realidad económica.

2. Otra limitación puede ser la composición de la muestra ya que muchos estudios se realizan con estudiantes universitarios, lo que puede no representar a la población general entonces esto se podría solucionar incluyendo participantes de diversos contextos socioeconómicos y culturales.