Lecture 24: Plotting

Comp 102

Fakhir Shaheen

Forman Christian University

Today: Five Stories

1



Water Pump

London 1854

2



Cold Launch

Florida 1986

3



24 Galaxies

California 1929

4



Army Hospital

Istanbul 1855

5



13 Tables

2017

Each story has **real numbers** and one **question**.

Today: Five Stories

1



Water Pump

London 1854

2



Cold Launch

Florida 1986

3



24 Galaxies

California 1929

4



Army Hospital

Istanbul 1855

5



13 Tables

2017

Each story has **real numbers** and one **question**.

Each time, a **picture** gave the answer.

Today: Five Stories

1



Water Pump

London 1854

2



Cold Launch

Florida 1986

3



24 Galaxies

California 1929

4



Army Hospital

Istanbul 1855

5



13 Tables

2017

Each story has **real numbers** and one **question**.

Each time, a **picture** gave the answer.

We draw every picture in Python, with `matplotlib`.

Story 1 of 5: The Water Pump

1



Water Pump

London 1854

2



Cold Launch

Florida 1986

3



24 Galaxies

California 1929

4



Army Hospital

Istanbul 1855

5



13 Tables

2017

New tools: `scatter` `plot` `title` `xlabel` `ylabel`

London, 1854

- Soho, a busy part of London.



a water pump

No taps at home.
Each street has a pump.

London, 1854

- Soho, a busy part of London.
- People get very sick: **cholera**.

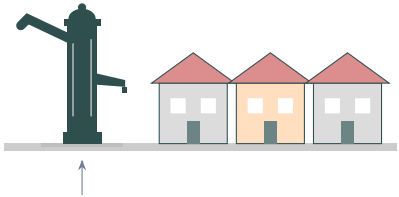


a water pump

No taps at home.
Each street has a pump.

London, 1854

- Soho, a busy part of London.
- People get very sick: **cholera**.
- In 10 days, more than **500** die.



a water pump

No taps at home.
Each street has a pump.

London, 1854

- Soho, a busy part of London.
- People get very sick: **cholera**.
- In 10 days, more than **500** die.



a water pump

No taps at home.
Each street has a pump.

Doctors say: cholera comes from **bad air**.
Nobody knows about germs yet.

Dr John Snow Makes a List

Broad St
Broad St
Poland St
Cambridge St
Broad St
Berwick St
Broad St
Silver St
Broad St
Dean St
⋮

- He thinks: it is the **water**, not the air.

Dr John Snow Makes a List

Broad St
Broad St
Poland St
Cambridge St
Broad St
Berwick St
Broad St
Silver St
Broad St
Dean St
⋮

- He thinks: it is the **water**, not the air.
- He writes down where each dead person lived.

Dr John Snow Makes a List

Broad St
Broad St
Poland St
Cambridge St
Broad St
Berwick St
Broad St
Silver St
Broad St
Dean St
⋮

- He thinks: it is the **water**, not the air.
- He writes down where each dead person lived.
- **578** deaths. A long list.

Dr John Snow Makes a List

Broad St
Broad St
Poland St
Cambridge St
Broad St
Berwick St
Broad St
Silver St
Broad St
Dean St
⋮

- He thinks: it is the **water**, not the air.
- He writes down where each dead person lived.
- **578** deaths. A long list.

Can you see the cause in a **list**?

Dr John Snow Makes a List

Broad St
Broad St
Poland St
Cambridge St
Broad St
Berwick St
Broad St
Silver St
Broad St
Dean St
⋮

- He thinks: it is the **water**, not the air.
- He writes down where each dead person lived.
- **578** deaths. A long list.

Can you see the cause in a **list**?

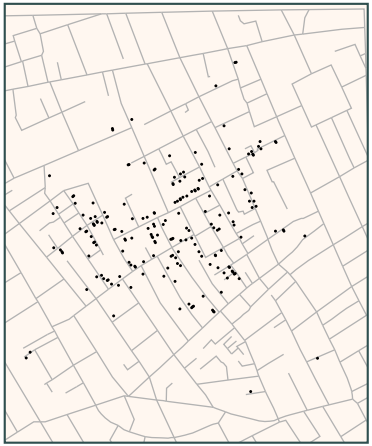
So he draws a map.

Snow's Map



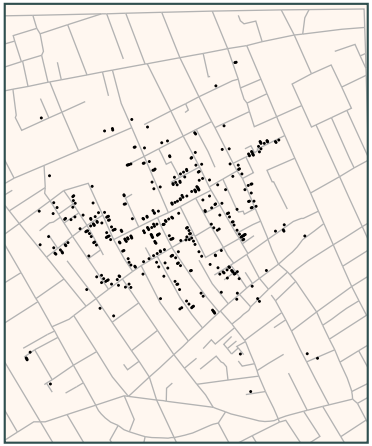
The streets of Soho, from above.

Snow's Map



He puts one dot on the map
for **each death**,
at the house of that person.

Snow's Map



He puts one dot on the map
for **each death**,
at the house of that person.

Snow's Map

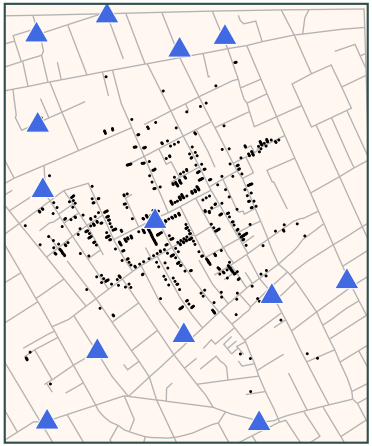


All **578** deaths.

Some streets are full of dots.

Some have none.

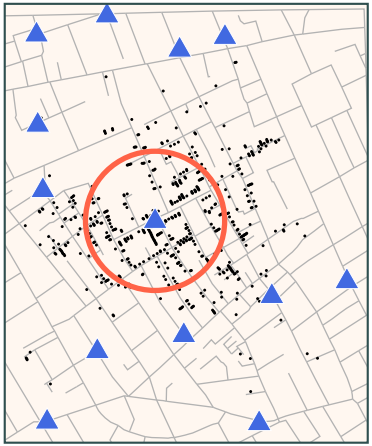
Snow's Map



Every ▲ is a **water pump**.

There are 13 of them.

Snow's Map

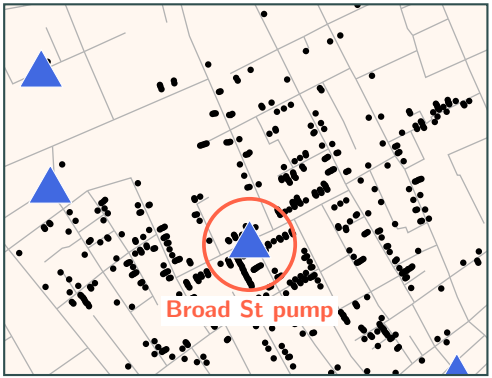


The dots crowd around **one** pump:

Broad Street.

The list had the same 578 addresses.
Only the **picture** shows the crowd.

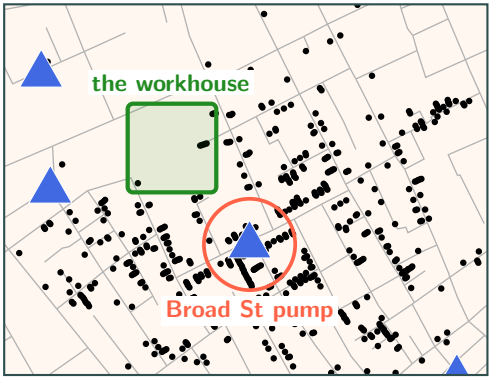
Two More Clues



Look closer at the map.

Around the pump, the dots are thick.

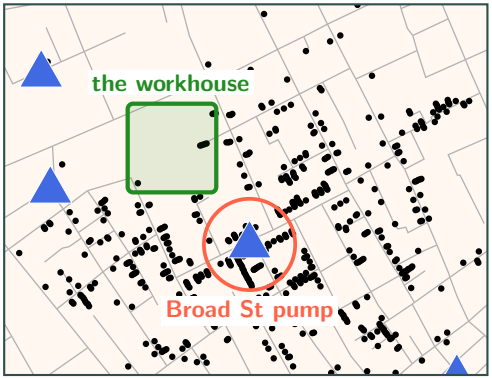
Two More Clues



1. The workhouse

535 poor people live in this block, next to the pump.

Two More Clues



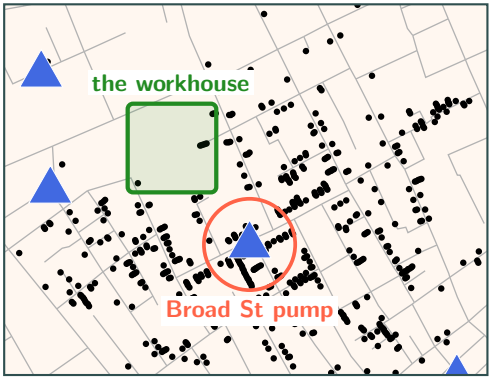
1. The workhouse

535 poor people live in this block, next to the pump.

Count the dots inside: only **5**.

A block of this size around the pump has 65.

Two More Clues



1. The workhouse

535 poor people live in this block, next to the pump.

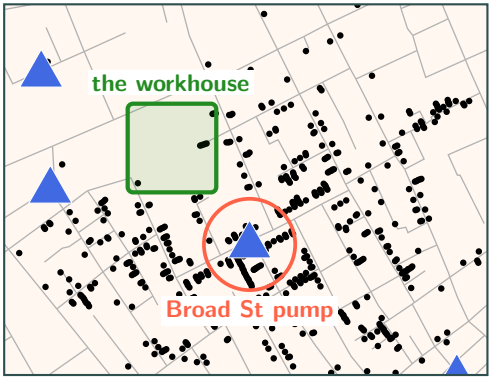
Count the dots inside: only **5**.

A block of this size around the pump has 65.

Why? The workhouse has **its own well**.

Two More Clues

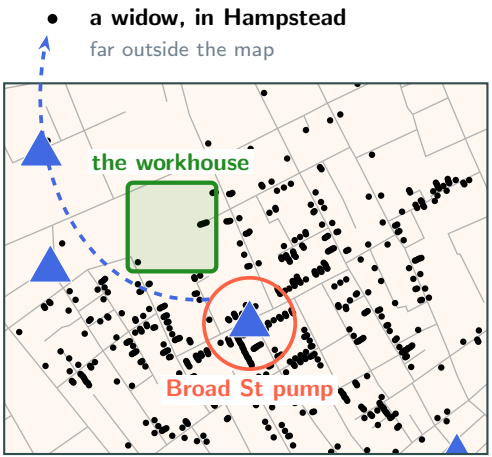
- **a widow, in Hampstead**
  far outside the map



2. The widow

She lives far away, in clean air. She **dies** of cholera too.

Two More Clues

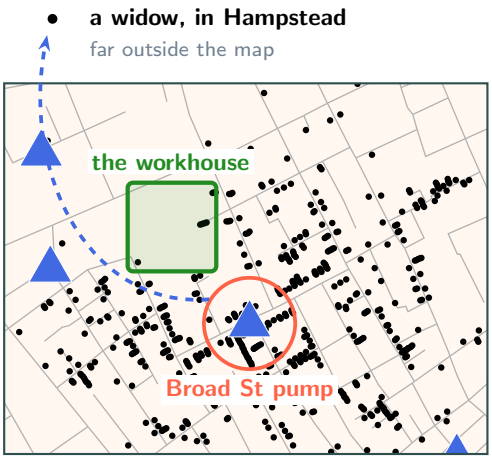


2. The widow

She lives far away, in clean air. She **dies** of cholera too.

Why? She liked Broad Street water. A cart brought her **a bottle** every day.

Two More Clues



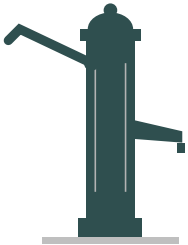
2. The widow

She lives far away, in clean air. She **dies** of cholera too.

Why? She liked Broad Street water. A cart brought her **a bottle** every day.

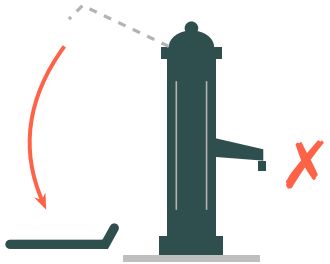
It is the water, not the air.

Take Off the Handle



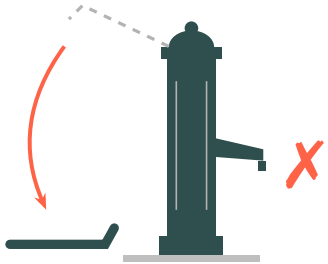
- Snow tells the council: it is **this** pump.

Take Off the Handle



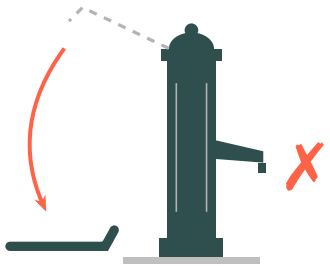
- Snow tells the council: it is **this** pump.
- 8 September: its handle comes **off**.

Take Off the Handle



- Snow tells the council: it is **this** pump.
- 8 September: its handle comes **off**.
- No one can drink that water now.

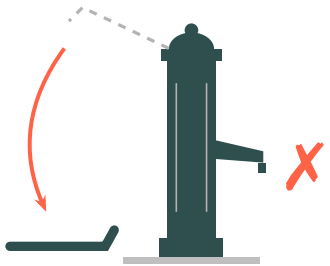
Take Off the Handle



- Snow tells the council: it is **this** pump.
- 8 September: its handle comes **off**.
- No one can drink that water now.

Later they find it: a dirty drain,
less than one metre from the well.

Take Off the Handle



- Snow tells the council: it is **this** pump.
- 8 September: its handle comes **off**.
- No one can drink that water now.

Later they find it: a dirty drain,
less than one metre from the well.

Now we draw his map, in Python.

Get the Tool

`matplotlib`: the Python tool that draws plots.

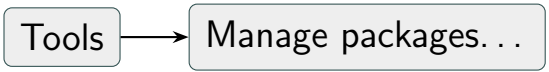
in Thonny

Tools

Get the Tool

`matplotlib`: the Python tool that draws plots.

in Thonny



Get the Tool

`matplotlib`: the Python tool that draws plots.

in Thonny



once per computer

Get the Tool

`matplotlib`: the Python tool that draws plots.

in Thonny



once per computer

```
import matplotlib.pyplot as plt
```

a short name

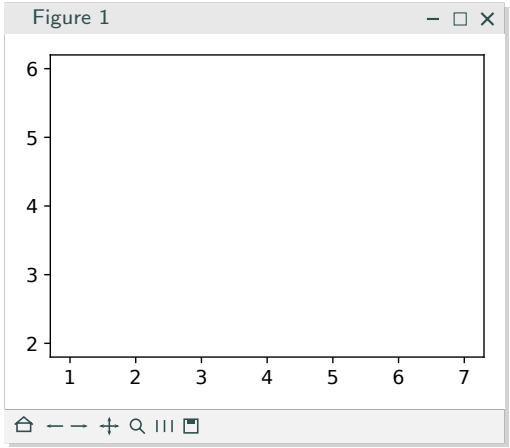
The first line of every plotting program.

Two Lists, One Dot Each

```
xs = [1, 2, 4, 5, 7]
ys = [3, 5, 2, 6, 4]
```

x	1	2	4	5	7
y	3	5	2	6	4

Two lists of the same length.



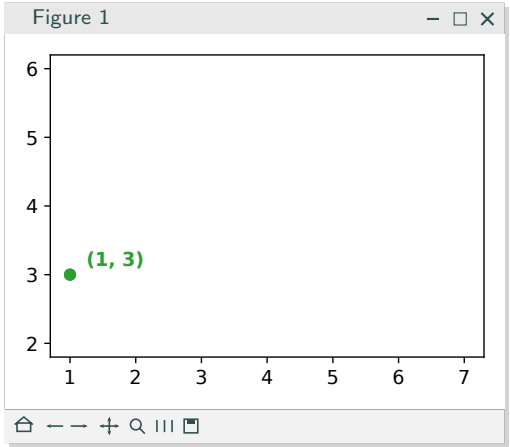
Two Lists, One Dot Each

```
xs = [1, 2, 4, 5, 7]
ys = [3, 5, 2, 6, 4]
plt.scatter(xs, ys)
```

x	1	2	4	5	7
y	3	5	2	6	4

One pair = **one dot**.

x: how far **across**. y: how far **up**.



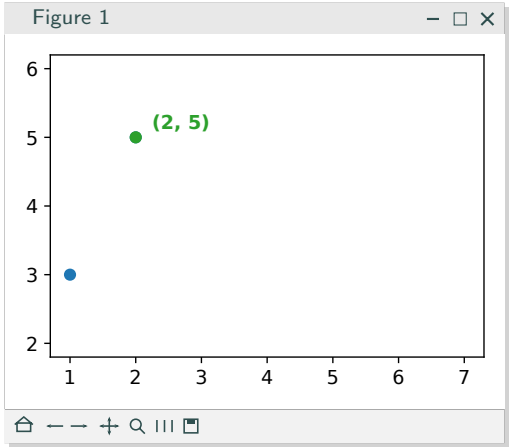
Two Lists, One Dot Each

```
xs = [1, 2, 4, 5, 7]
ys = [3, 5, 2, 6, 4]
plt.scatter(xs, ys)
```

x	1	2	4	5	7
y	3	5	2	6	4

One pair = **one dot**.

x: how far **across**. y: how far **up**.



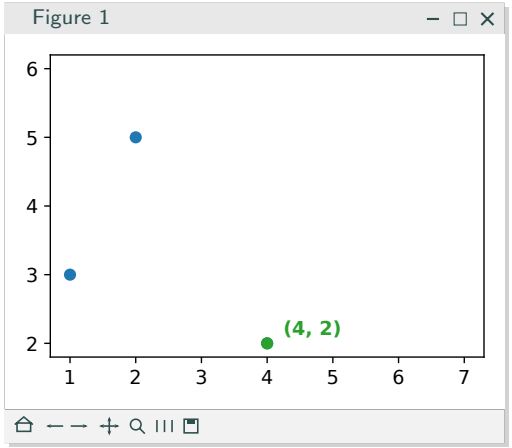
Two Lists, One Dot Each

```
xs = [1, 2, 4, 5, 7]
ys = [3, 5, 2, 6, 4]
plt.scatter(xs, ys)
```

x	1	2	4	5	7
y	3	5	2	6	4

One pair = **one dot**.

x: how far **across**. y: how far **up**.



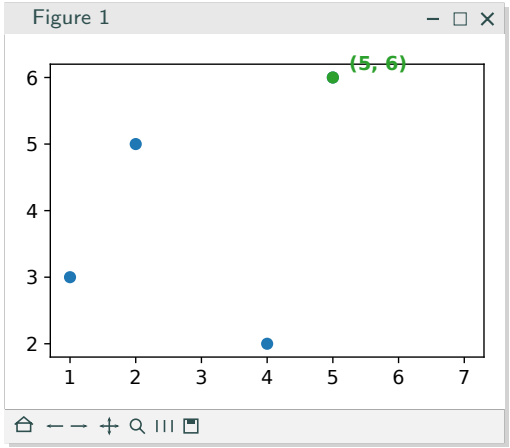
Two Lists, One Dot Each

```
xs = [1, 2, 4, 5, 7]
ys = [3, 5, 2, 6, 4]
plt.scatter(xs, ys)
```

x	1	2	4	5	7
y	3	5	2	6	4

One pair = **one dot**.

x: how far **across**. y: how far **up**.



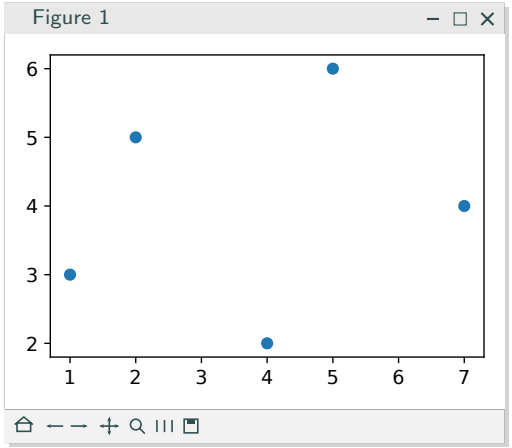
Two Lists, One Dot Each

```
xs = [1, 2, 4, 5, 7]
ys = [3, 5, 2, 6, 4]
plt.scatter(xs, ys)
```

x	1	2	4	5	7
y	3	5	2	6	4

One pair = **one dot**.

x: how far **across**. y: how far **up**.

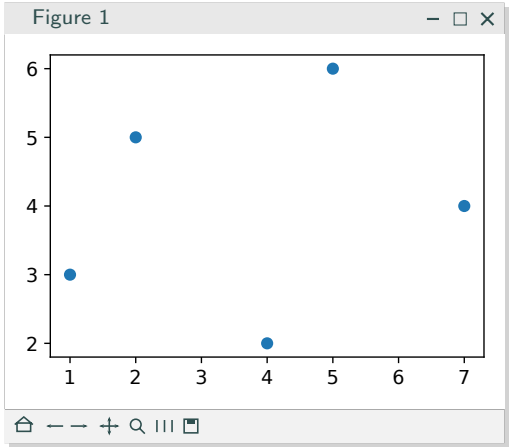


Two Lists, One Dot Each

```
xs = [1, 2, 4, 5, 7]
ys = [3, 5, 2, 6, 4]
plt.scatter(xs, ys)
plt.show()
```

x	1	2	4	5	7
y	3	5	2	6	4

`plt.show()` opens the window.
Close the window, and the program goes on.



Snow's Map in Python

```
plt.scatter(death_xs, death_ys,
            s=4)
```

```
plt.show()
```

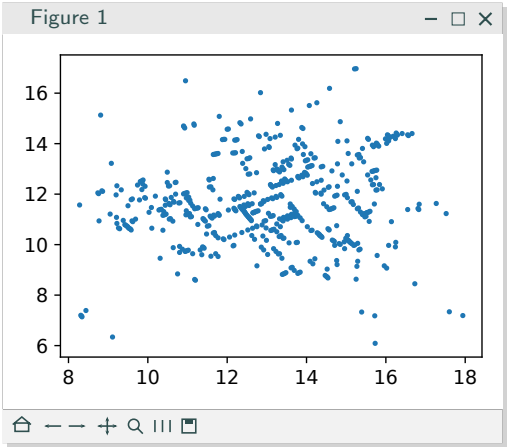
`death_xs` 578 numbers: across

`death_ys` 578 numbers: up

`s=4` the size of a dot: small

578 dots: one for each death.

The real numbers are in `1ec24_data.py`.

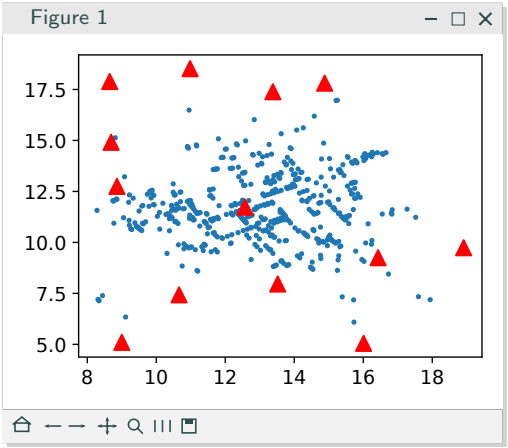


Snow's Map in Python

```
plt.scatter(death_xs, death_ys,
            s=4)
plt.scatter(pump_xs, pump_ys,
            s=80, marker='^',
            color='red')
plt.show()
```

`s=80`	bigger marks
`marker='^'`	a triangle, not a dot
`color='red'`	its colour

Two `scatter` lines draw on the **same** plot.

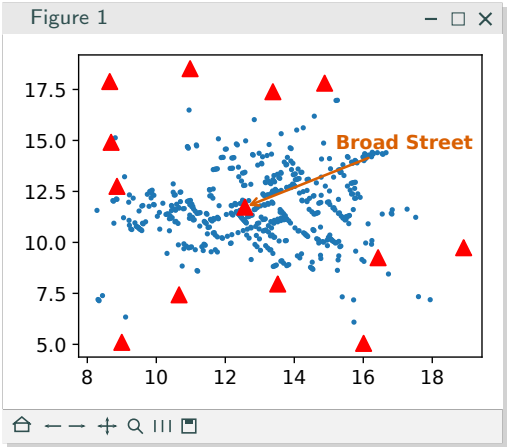


Snow's Map in Python

```
plt.scatter(death_xs, death_ys,
            s=4)
plt.scatter(pump_xs, pump_ys,
            s=80, marker='^',
            color='red')
plt.show()
```

`s=80`	bigger marks
`marker='^'`	a triangle, not a dot
`color='red'`	its colour

Read it: the dots are thickest around **one** triangle, the Broad Street pump.



Day by Day

```
days  = [1, 2, 3,  4,   5,   6, ...]
cases = [1, 1, 8, 56, 143, 116, ...]
```

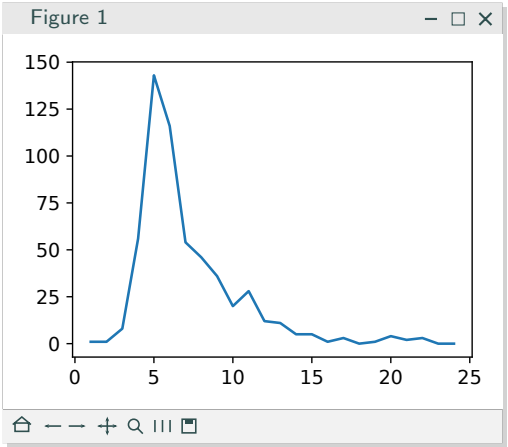
cases: how many people fell sick on each day.

24 days, from 28 August.

Day by Day

```
days  = [1, 2, 3,  4,   5,   6, ...]
cases = [1, 1, 8, 56, 143, 116, ...]
plt.plot(days, cases)
plt.show()
```

`plot` makes the same points as `scatter`, then **joins** them with a line, in list order.

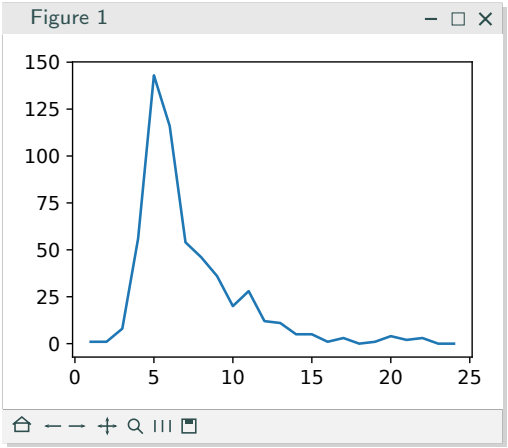


Day by Day

```
days  = [1, 2, 3,  4,   5,   6, ...]
cases = [1, 1, 8, 56, 143, 116, ...]
plt.plot(days, cases)
plt.show()
```

A line is good for
how something changes.

Read it: a fast climb, a peak, then a slow fall.



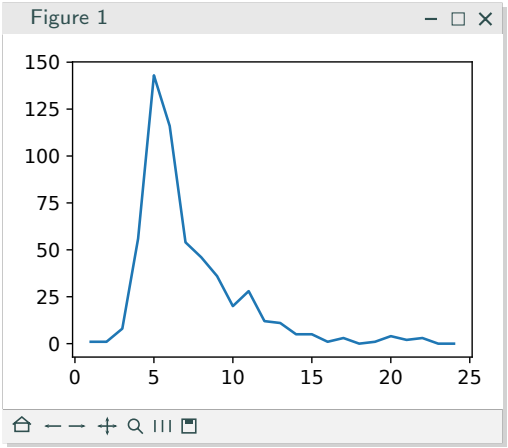
Day by Day

```
days  = [1, 2, 3,  4,   5,   6, ...]
cases = [1, 1, 8, 56, 143, 116, ...]
plt.plot(days, cases)
plt.show()
```

A line is good for
how something changes.

Read it: a fast climb, a peak, then a slow fall.

But: 140 of what? Day 5 of what?

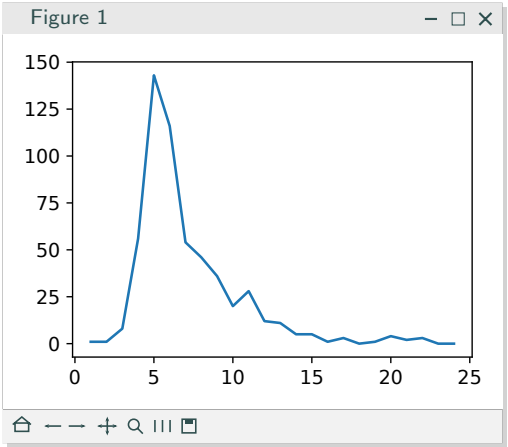


A Plot Needs Words

```
plt.plot(days, cases)
```

```
plt.show()
```

Without words, a plot is a puzzle.

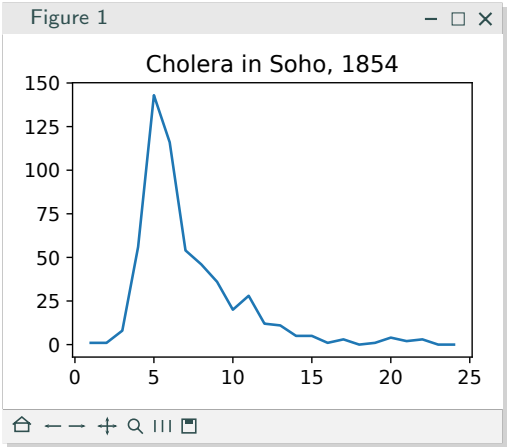


A Plot Needs Words

```
plt.plot(days, cases)
plt.title('Cholera in Soho, 1854')


plt.show()
```

`title`: the words on top.

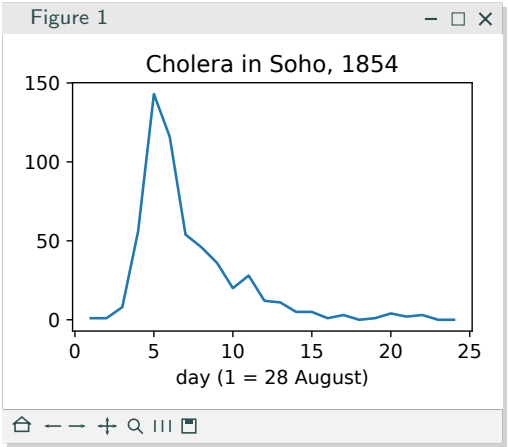


A Plot Needs Words

```
plt.plot(days, cases)
plt.title('Cholera in Soho, 1854')
plt.xlabel('day (1 = 28 August)')

plt.show()
```

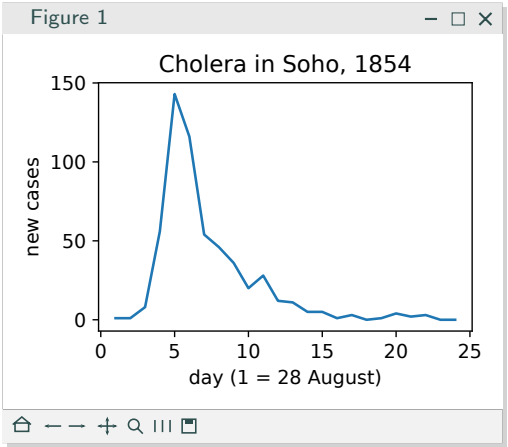
`xlabel`: what the numbers along the bottom mean.



A Plot Needs Words

```
plt.plot(days, cases)
plt.title('Cholera in Soho, 1854')
plt.xlabel('day (1 = 28 August)')
plt.ylabel('new cases')
plt.show()
```

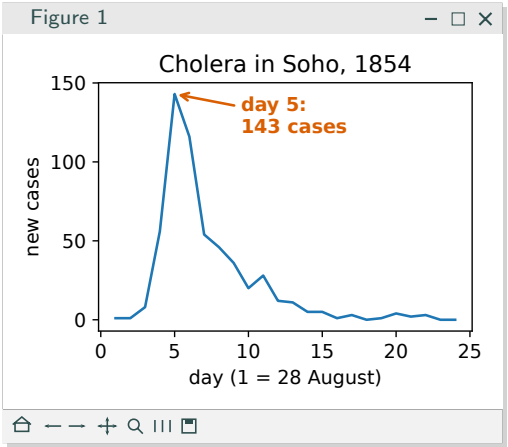
`ylabel`: what the numbers up the side mean.



A Plot Needs Words

```
plt.plot(days, cases)
plt.title('Cholera in Soho, 1854')
plt.xlabel('day (1 = 28 August)')
plt.ylabel('new cases')
plt.show()
```

Read it: on day 5 (1 September), 143 people fell sick.

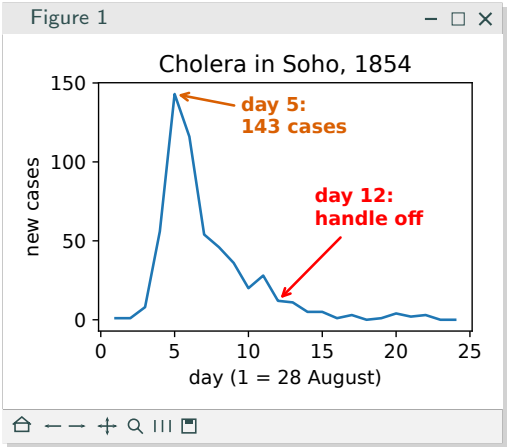


A Plot Needs Words

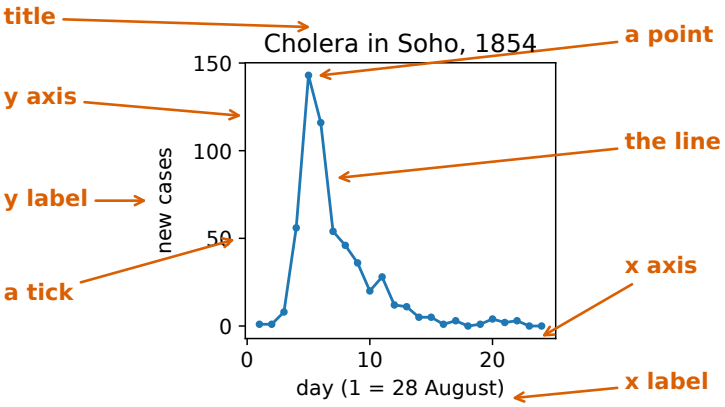
```
plt.plot(days, cases)
plt.title('Cholera in Soho, 1854')
plt.xlabel('day (1 = 28 August)')
plt.ylabel('new cases')
plt.show()
```

Read it: the handle came off on day 12. The cases were **already falling**.

Snow wrote this himself. A plot must be read **honestly**.



The Parts of a Plot



Every plot today has these parts. A tick: a small mark with its number.

You Try! *(2 min)*

(a) What does this draw?

```
plt.scatter([1, 2, 3], [2, 2, 2])
plt.show()
```

(b) Which tool joins the points with a line?

(c) Write the line that puts the word `year` under the plot.

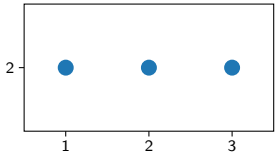
You Try! *(2 min)*

(a) What does this draw?

```
plt.scatter([1, 2, 3], [2, 2, 2])
plt.show()
```

(b) Which tool joins the points with a line?

(c) Write the line that puts the word `year` under the plot.



(a) three dots in a flat row

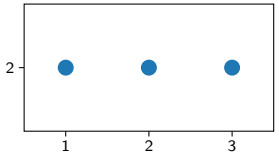
You Try! *(2 min)*

(a) What does this draw?

```
plt.scatter([1, 2, 3], [2, 2, 2])
plt.show()
```

(b) Which tool joins the points with a line?

(c) Write the line that puts the word `year` under the plot.



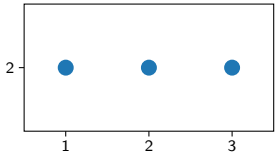
(a) three dots in a flat row

(b) `plt.plot`

You Try! *(2 min)*

(a) What does this draw?

```
plt.scatter([1, 2, 3], [2, 2, 2])
plt.show()
```



(a) three dots in a flat row

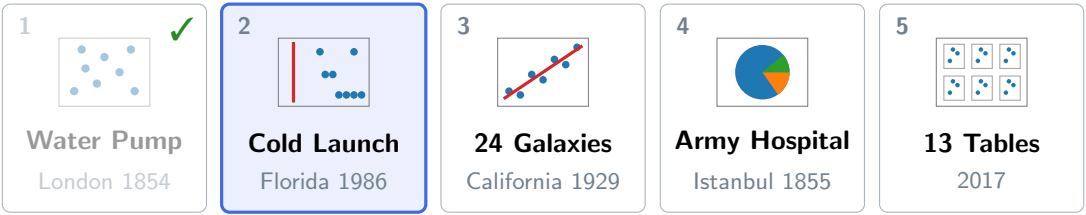
(b) Which tool joins the points with a line?

(b) `plt.plot`

(c) Write the line that puts the word `year` under the plot.

(c) `plt.xlabel('year')`

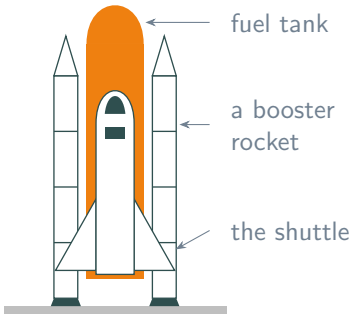
Story 2 of 5: The Cold Launch



New tools: `alpha=` `xlim` `axvline` `text`

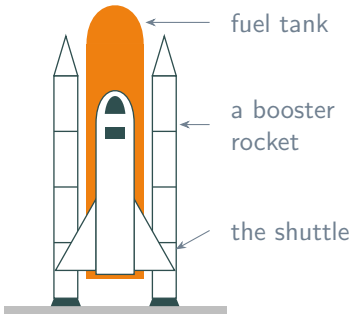
Florida, January 1986

- The space shuttle **Challenger** will fly tomorrow morning.



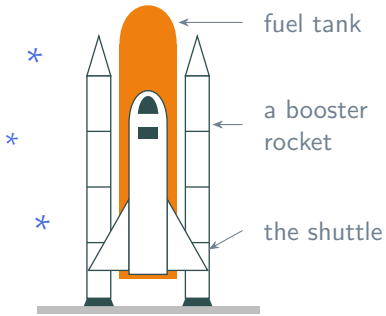
Florida, January 1986

- The space shuttle **Challenger** will fly tomorrow morning.
- **7** astronauts are on board. One is a school teacher.



Florida, January 1986

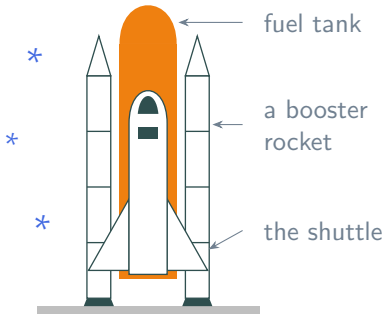
- The space shuttle **Challenger** will fly tomorrow morning.
- **7** astronauts are on board. One is a school teacher.
- The night is very cold: about $-1\,^{\circ}\mathbf{C}$.



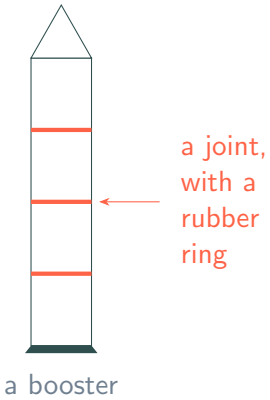
Florida, January 1986

- The space shuttle **Challenger** will fly tomorrow morning.
- **7** astronauts are on board. One is a school teacher.
- The night is very cold: about $-1\,^{\circ}\mathbf{C}$.

Some engineers are worried.
They phone NASA: **do not launch**.

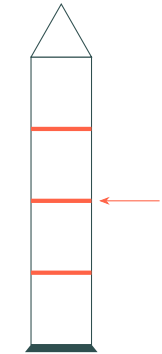


The Rubber Rings



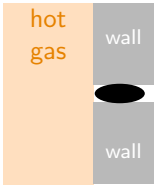
A booster is built in **pieces**. A **rubber ring** seals each joint, so the hot gas stays inside.

The Rubber Rings



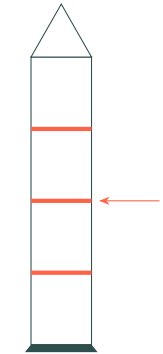
a booster

A booster is built in **pieces**. A **rubber ring** seals each joint, so the hot gas stays inside.



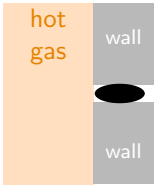
warm: soft ring
it fills the gap

The Rubber Rings

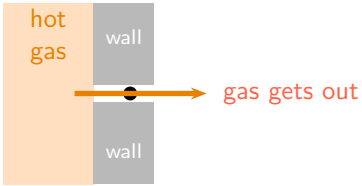


a booster

A booster is built in **pieces**. A **rubber ring** seals each joint, so the hot gas stays inside.

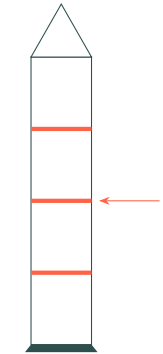


warm: soft ring
it fills the gap

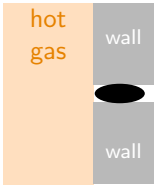


cold: hard ring
it leaves a gap

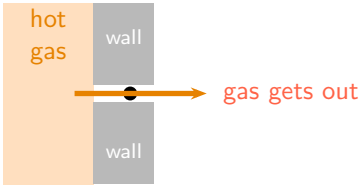
The Rubber Rings



A booster is built in **pieces**. A **rubber ring** seals each joint, so the hot gas stays inside.



warm: soft ring
it fills the gap



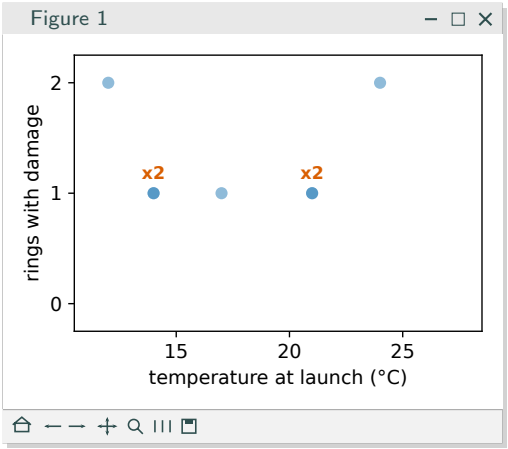
cold: hard ring
it leaves a gap

After each launch, engineers count the rings **with damage**.

The Night Before: 7 Launches

The engineers show the **7** earlier launches that had ring damage.

°C at launch	21	14	17	21	**12**	**24**	14
rings with damage	1	1	1	1	**2**	**2**	1



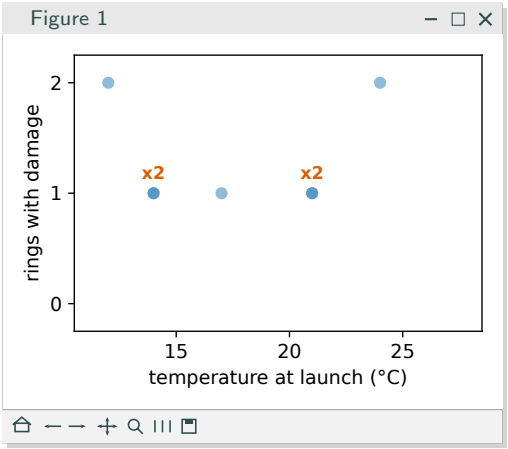
x2 = 2 launches at that spot

The Night Before: 7 Launches

The engineers show the **7** earlier launches that had ring damage.

°C at launch	21	14	17	21	**12**	**24**	14
rings with damage	1	1	1	1	**2**	**2**	1

Read it: damage on a cold day (12 °C).
Damage on a warm day (24 °C).



x2 = 2 launches at that spot

The Night Before: 7 Launches

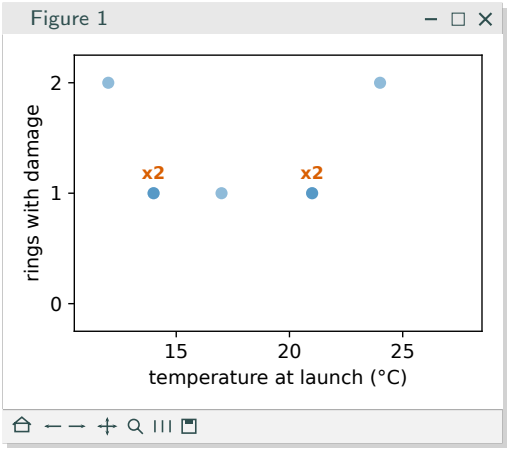
The engineers show the **7** earlier launches that had ring damage.

°C at launch	21	14	17	21	**12**	**24**	14
rings with damage	1	1	1	1	**2**	**2**	1

Read it: damage on a cold day (12 °C).
Damage on a warm day (24 °C).

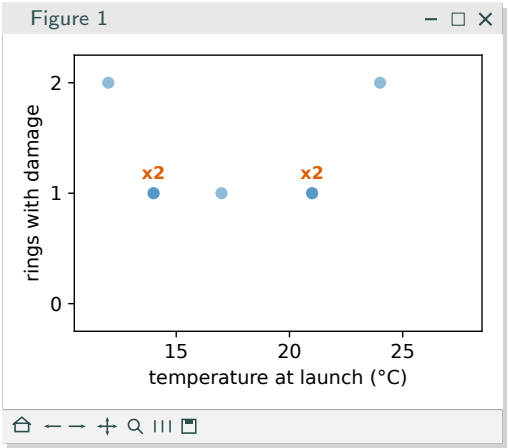
The managers: "**No pattern.** We launch."

x2 = 2 launches at that spot



All 23 Launches

There were **23** launches. The other **16** had **no** damage. They were left out!



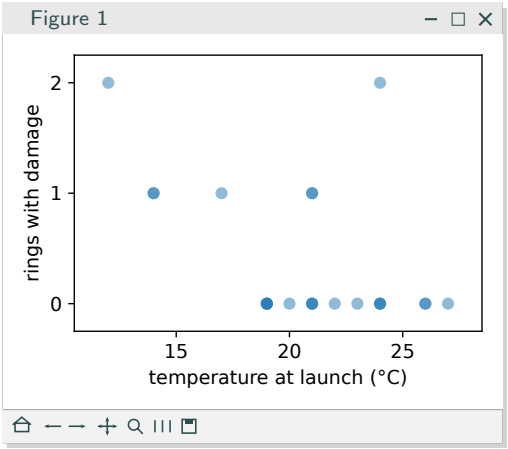
All 23 Launches

There were **23** launches. The other **16** had **no** damage. They were left out!

```
plt.scatter(temps, damage, alpha=0.5)
plt.xlabel('temperature at launch (°C)')
plt.ylabel('rings with damage')
plt.show()
```

The 16 new dots sit on the bottom line: **0** damage.

`alpha=0.5`: see-through dots. A **darker** dot is more than one launch.

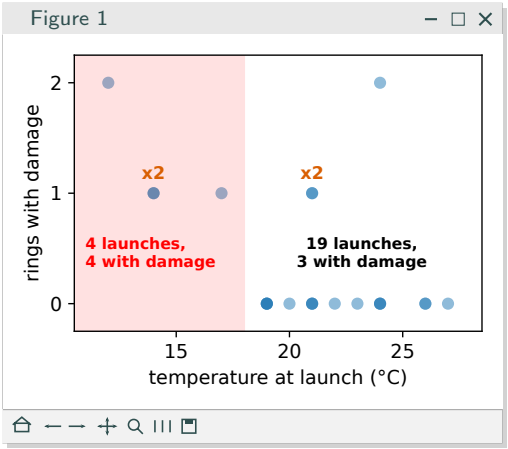


All 23 Launches

There were **23** launches. The other **16** had **no** damage. They were left out!

```
plt.scatter(temps, damage, alpha=0.5)
plt.xlabel('temperature at launch (°C)')
plt.ylabel('rings with damage')
plt.show()
```

	launches	with damage
below 18 °C	4	4
19 °C or warmer	19	**3**



All 23 Launches

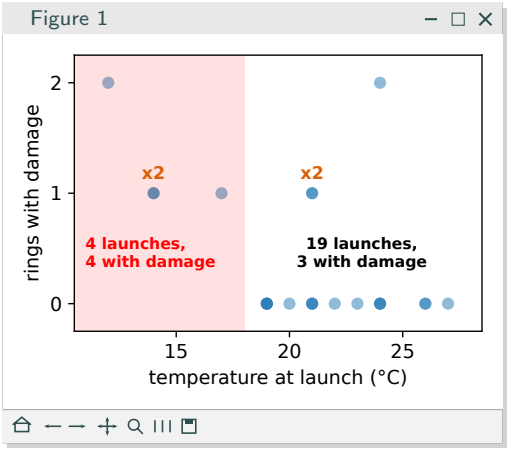
There were **23** launches. The other **16** had **no** damage. They were left out!

```
plt.scatter(temps, damage, alpha=0.5)
plt.xlabel('temperature at launch (°C)')
plt.ylabel('rings with damage')
plt.show()
```

	launches	with damage
below 18 °C	4	**4**
19 °C or warmer	19	**3**

All 16 clean launches were warm.

Leave them out, and cold looks harmless.



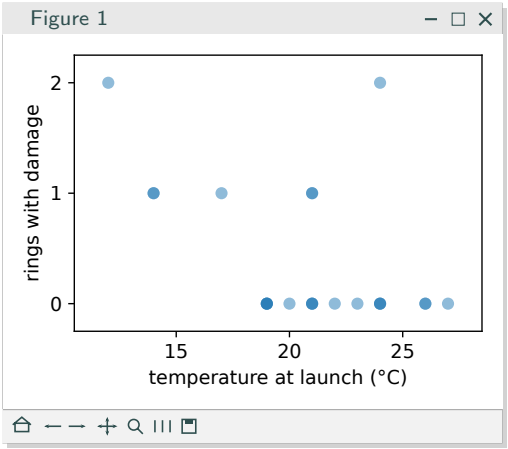
Where Is Launch Day?

```
plt.scatter(temps, damage, alpha=0.5)
```

```
plt.show()
```

Launch day: about −1 °C.

It is not on this plot. The x axis stops near 10.



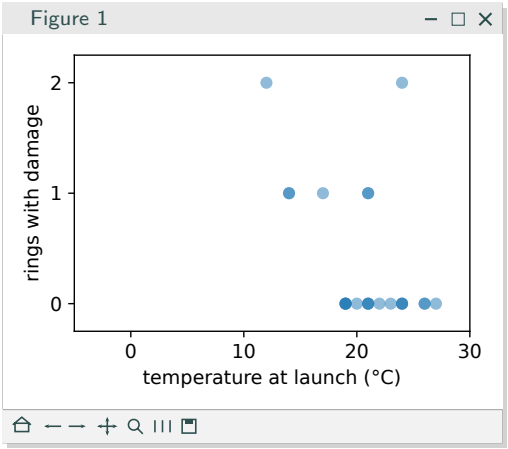
Where Is Launch Day?

```
plt.scatter(temps, damage, alpha=0.5)
plt.xlim(-5, 30)
```

```
plt.show()
```

`xlim(-5, 30)`: the x axis now goes from -5 to 30.

The dots are the same. We see **more** of the axis.

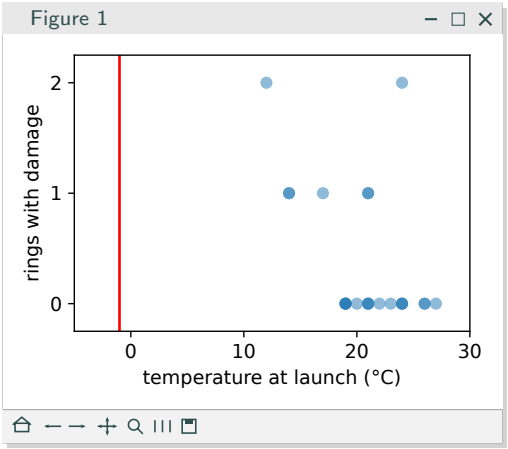


Where Is Launch Day?

```
plt.scatter(temps, damage, alpha=0.5)
plt.xlim(-5, 30)
plt.axvline(-1, color='red')

plt.show()
```

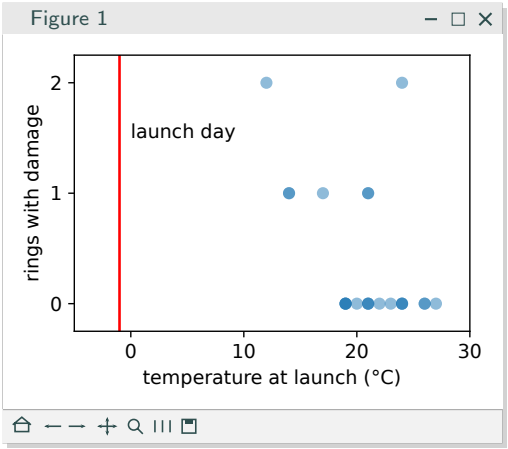
`axvline(-1)`: a line straight up at $x = -1$.



Where Is Launch Day?

```
plt.scatter(temps, damage, alpha=0.5)
plt.xlim(-5, 30)
plt.axvline(-1, color='red')
plt.text(0, 1.5, 'launch day')
plt.show()
```

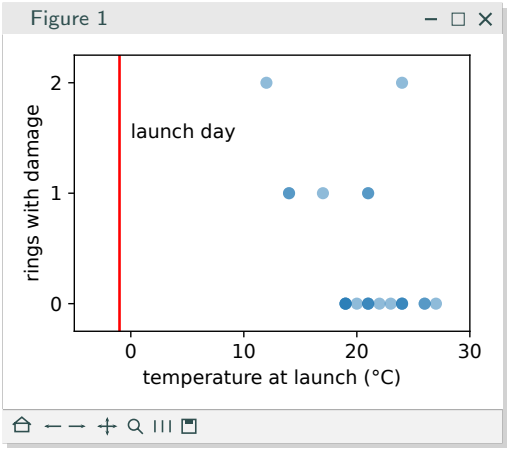
`text(0, 1.5, ...)`: words inside the plot, at the point (0, 1.5).



Where Is Launch Day?

```
plt.scatter(temps, damage, alpha=0.5)
plt.xlim(-5, 30)
plt.axvline(-1, color='red')
plt.text(0, 1.5, 'launch day')
plt.show()
```

Read it: the coldest launch before was 12 °C. Launch day is **13 degrees colder**.

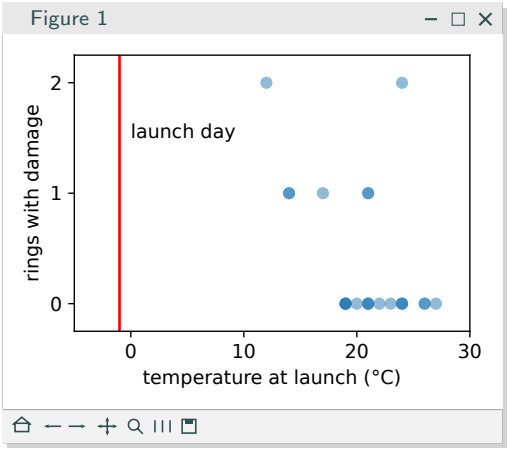


Where Is Launch Day?

```
plt.scatter(temps, damage, alpha=0.5)
plt.xlim(-5, 30)
plt.axvline(-1, color='red')
plt.text(0, 1.5, 'launch day')
plt.show()
```

Read it: the coldest launch before was 12 °C. Launch day is **13 degrees colder**.

Every launch below 18 °C had damage. **No** launch showed that −1 °C was safe.



28 January 1986

- Challenger is launched.

28 January 1986

- Challenger is launched.
- A cold rubber ring does not seal.

28 January 1986

- Challenger is launched.
- A cold rubber ring does not seal.
- After **73 seconds** the shuttle breaks apart. All 7 astronauts die.

28 January 1986

- Challenger is launched.
- A cold rubber ring does not seal.
- After **73 seconds** the shuttle breaks apart. All 7 astronauts die.

The numbers were all in the engineers' tables.

The plot was drawn **after** the accident.

28 January 1986

- Challenger is launched.
- A cold rubber ring does not seal.
- After **73 seconds** the shuttle breaks apart. All 7 astronauts die.

The numbers were all in the engineers' tables.

The plot was drawn **after** the accident.

Plot **all** the data, also the zeros.

Then ask: where is **my** case on this plot?

You Try! *(2 min)*

For 30 students: `hours` of study and `marks`.

```
plt.scatter(hours, marks)
plt.xlim(0, 12)
plt.axvline(8, color='red')
plt.text(8.3, 20, 'me')
plt.show()
```

(a) Which line sets the x axis from 0 to 12?

(b) Where is the red line?

(c) Where is the word `me`?

(d) The lists hold only the students who passed. What is missing?

You Try! *(2 min)*

For 30 students: `hours` of study and `marks`.

```
plt.scatter(hours, marks)
plt.xlim(0, 12)
plt.axvline(8, color='red')
plt.text(8.3, 20, 'me')
plt.show()
```

(a) Which line sets the x axis from 0 to 12?
`plt.xlim(0, 12)`

(b) Where is the red line?

(c) Where is the word `me`?

(d) The lists hold only the students who passed. What is missing?

You Try! *(2 min)*

For 30 students: `hours` of study and `marks`.

```
plt.scatter(hours, marks)
plt.xlim(0, 12)
plt.axvline(8, color='red')
plt.text(8.3, 20, 'me')
plt.show()
```

(a) Which line sets the x axis from 0 to 12?
`plt.xlim(0, 12)`

(b) Where is the red line? **at x = 8**

(c) Where is the word `me`?

(d) The lists hold only the students who passed. What is missing?

You Try! *(2 min)*

For 30 students: `hours` of study and `marks`.

```
plt.scatter(hours, marks)
plt.xlim(0, 12)
plt.axvline(8, color='red')
plt.text(8.3, 20, 'me')
plt.show()
```

(a) Which line sets the x axis from 0 to 12?
`plt.xlim(0, 12)`

(b) Where is the red line? **at x = 8**

(c) Where is the word `me`? **at (8.3, 20)**

(d) The lists hold only the students who passed. What is missing?

You Try! *(2 min)*

For 30 students: `hours` of study and `marks`.

```
plt.scatter(hours, marks)
plt.xlim(0, 12)
plt.axvline(8, color='red')
plt.text(8.3, 20, 'me')
plt.show()
```

(a) Which line sets the x axis from 0 to 12?
`plt.xlim(0, 12)`

(b) Where is the red line? **at x = 8**

(c) Where is the word `me`? **at (8.3, 20)**

(d) The lists hold only the students who passed. What is missing?
those who failed: the zeros

Story 3 of 5: 24 Galaxies

1 ✓

Water Pump
London 1854

2 ✓

Cold Launch
Florida 1986

3

24 Galaxies
California 1929

4

Army Hospital
Istanbul 1855

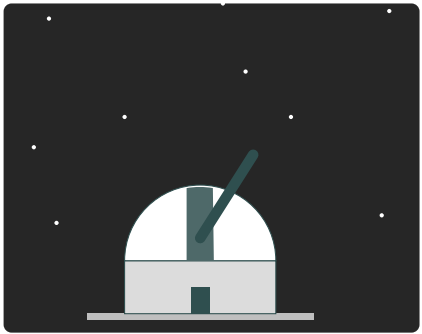
5

13 Tables
2017

New tools: `plot` a line `color=` `label=` `legend`

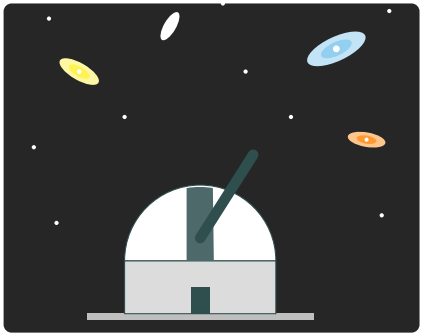
California, 1929

- **Edwin Hubble** works at the biggest telescope in the world.



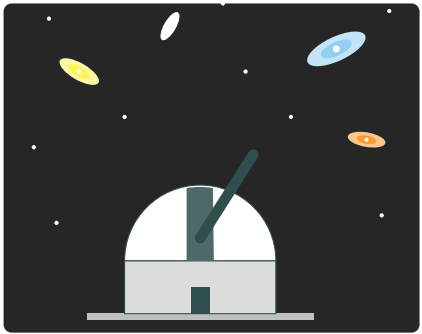
California, 1929

- **Edwin Hubble** works at the biggest telescope in the world.
- He studies **galaxies**: far islands of billions of stars.



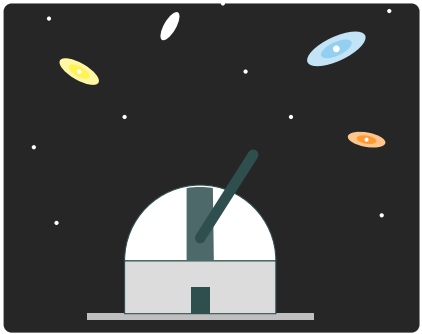
California, 1929

- **Edwin Hubble** works at the biggest telescope in the world.
- He studies **galaxies**: far islands of billions of stars.
- For **24** galaxies he has two numbers: how **far**? how **fast** does it move?



California, 1929

- **Edwin Hubble** works at the biggest telescope in the world.
- He studies **galaxies**: far islands of billions of stars.
- For **24** galaxies he has two numbers: how **far**? how **fast** does it move?



In 1929 almost everyone thinks: the universe **stands still**.
Even Einstein.

24 Dots

galaxy	distance	speed
Small Magellanic Cloud	0.03	+170
Andromeda	0.28	−220
NGC 5194	0.5	+270
NGC 4258	1.4	+500
NGC 4649	2.0	+1090

. . . 24 rows

Speed above 0: moving **away** from us.
Below 0: coming **closer**.

1 parsec is about 3 light years.

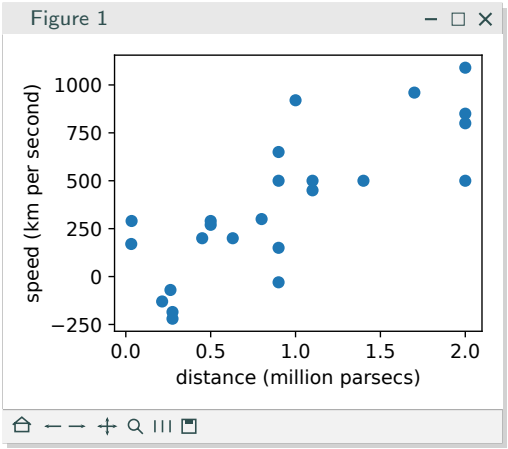
24 Dots

galaxy	distance	speed
Small Magellanic Cloud	0.03	+170
Andromeda	0.28	−220
NGC 5194	0.5	+270
NGC 4258	1.4	+500
NGC 4649	2.0	+1090

... 24 rows

```
plt.scatter(distances, speeds)
plt.xlabel('distance (million parsecs)')
plt.ylabel('speed (km per second)')
```

To the right: farther. Up: faster away.



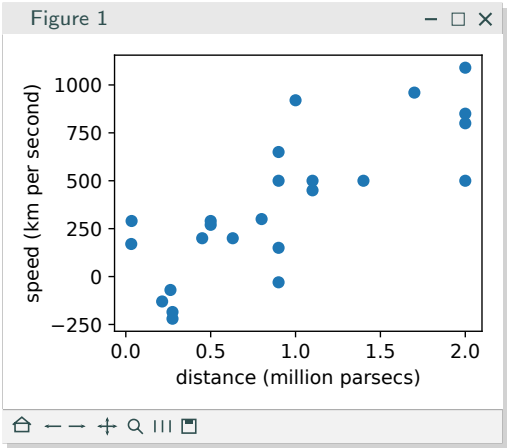
24 Dots

galaxy	distance	speed
Small Magellanic Cloud	0.03	+170
Andromeda	0.28	−220
NGC 5194	0.5	+270
NGC 4258	1.4	+500
NGC 4649	2.0	+1090

... 24 rows

```
plt.scatter(distances, speeds)
plt.xlabel('distance (million parsecs)')
plt.ylabel('speed (km per second)')
```

Read it: the dots climb. Far galaxies move away **faster**.

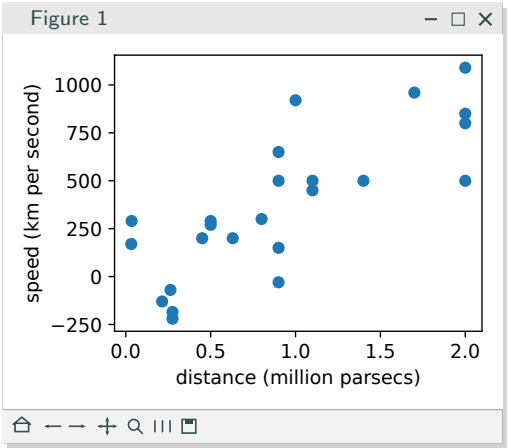


A Line Through the Dots

```
plt.scatter(distances, speeds)
```

```
plt.show()
```

Hubble draws a straight line through the cloud of dots.



A Line Through the Dots

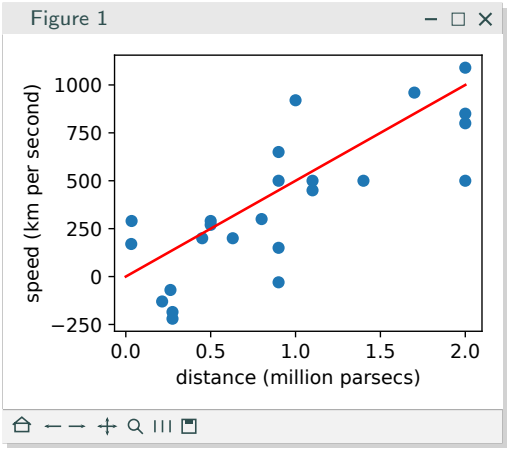
```
plt.scatter(distances, speeds)

plt.plot([0, 2], [0, 1000],
         color='red')

plt.show()
```

The two points are (0, 0) and (2, 1000).

A straight line needs only **two points**.

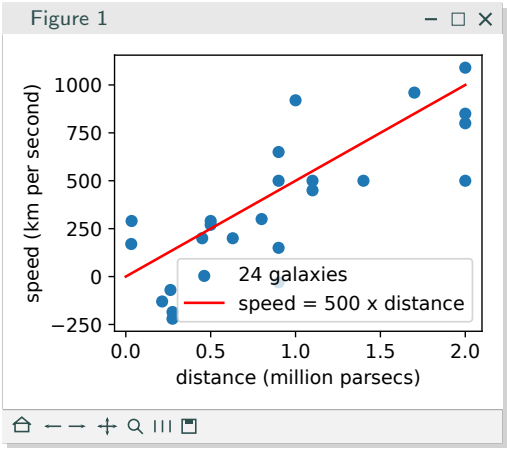


A Line Through the Dots

```
plt.scatter(distances, speeds,
            label='24 galaxies')
plt.plot([0, 2], [0, 1000],
         color='red',
         label='speed = 500 x distance')
plt.legend()
plt.show()
```

`label=` names each layer.

`legend()` draws the box of names.



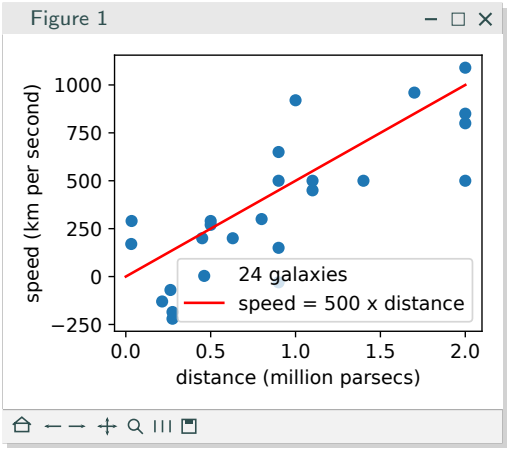
A Line Through the Dots

```
plt.scatter(distances, speeds,
            label='24 galaxies')
plt.plot([0, 2], [0, 1000],
         color='red',
         label='speed = 500 x distance')
plt.legend()
plt.show()
```

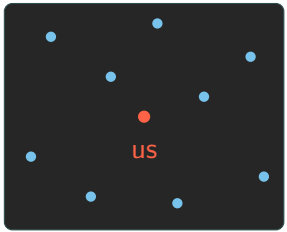
`label=` names each layer.

`legend()` draws the box of names.

Read it: twice as far, twice as fast.



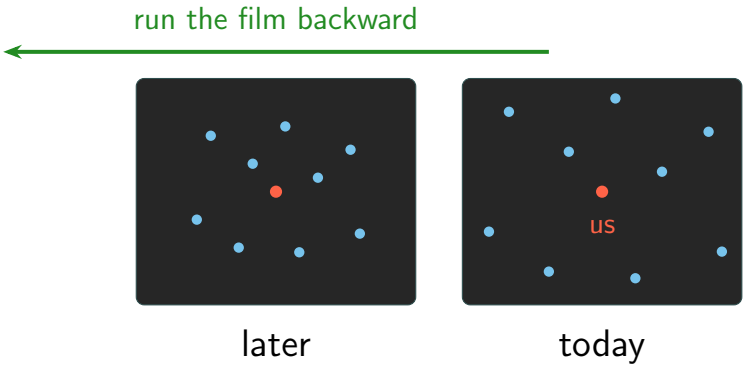
What the Line Says



today

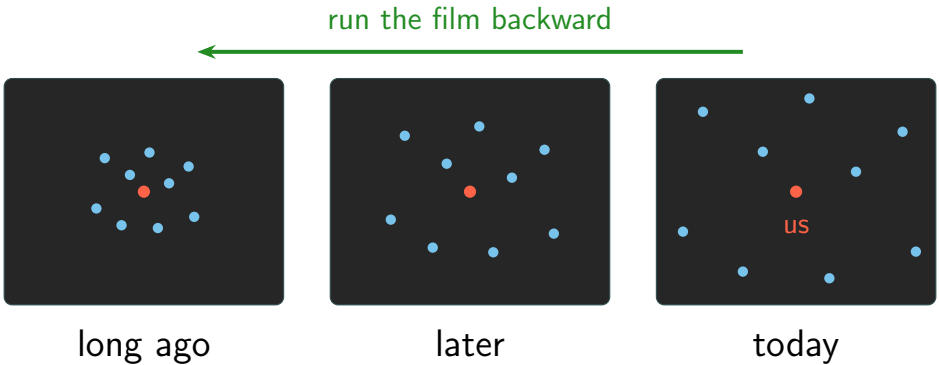
Every galaxy moves away from every other one: the universe **grows**.

What the Line Says



Go back in time: the galaxies come closer . . .

What the Line Says



. . . and closer. The universe had a **beginning**.

24 dots and one line changed how we see everything.

You Try! *(sketch it on paper, 1 min)*

What does this draw?

```
plt.plot([0, 2, 4], [0, 4, 0])
plt.show()
```

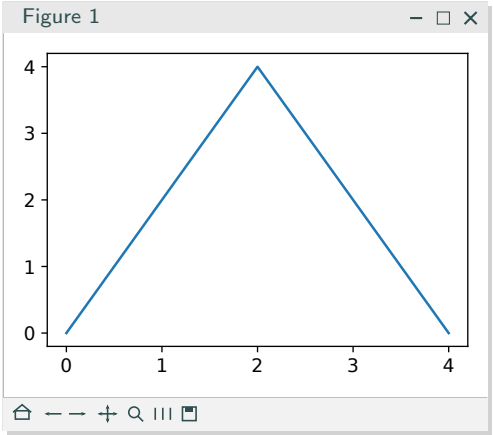
You Try! *(sketch it on paper, 1 min)*

What does this draw?

```
plt.plot([0, 2, 4], [0, 4, 0])
plt.show()
```

A tent. Three points:
(0, 0), then (2, 4), then (4, 0).

`plot` joins them **in list order**.



Story 4 of 5: The Army Hospital

1 ✓



Water Pump

London 1854

2 ✓



Cold Launch

Florida 1986

3 ✓



24 Galaxies

California 1929

4



Army Hospital

Istanbul 1855

5



13 Tables

2017

New tools: `pie` `bar` `bottom=` `polar`

Istanbul, 1854

- Britain is at war in Crimea. Its army hospital is in **Üsküdar**, Istanbul.



Istanbul, 1854

- Britain is at war in Crimea. Its army hospital is in **Üsküdar**, Istanbul.
- A nurse arrives: **Florence Nightingale**.



Istanbul, 1854

- Britain is at war in Crimea. Its army hospital is in **Üsküdar**, Istanbul.
- A nurse arrives: **Florence Nightingale**.
- The hospital is **dirty**: blocked drains, no fresh air.



Istanbul, 1854

- Britain is at war in Crimea. Its army hospital is in **Üsküdar**, Istanbul.
- A nurse arrives: **Florence Nightingale**.
- The hospital is **dirty**: blocked drains, no fresh air.
- Soldiers die every day. The army says: that is war.



Istanbul, 1854

- Britain is at war in Crimea. Its army hospital is in **Üsküdar**, Istanbul.
- A nurse arrives: **Florence Nightingale**.
- The hospital is **dirty**: blocked drains, no fresh air.
- Soldiers die every day. The army says: that is war.



She **counts**: every month, who died, and **of what**.

What Killed the Soldiers?

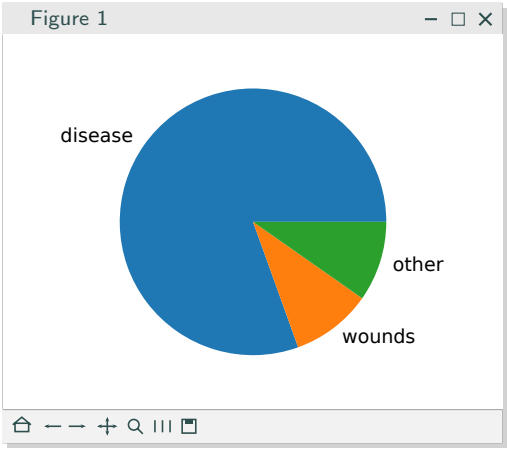
deaths in 2 years	
disease	**14,476**
wounds	1,758
other	1,748

What Killed the Soldiers?

deaths in 2 years	
disease	**14,476**
wounds	1,758
other	1,748

```
counts = [14476, 1758, 1748]
names = ['disease', 'wounds', 'other']
plt.pie(counts, labels=names)
plt.show()
```

`pie`: one slice for each count.
A bigger count gets a bigger slice.
Good for the **parts of one whole**.

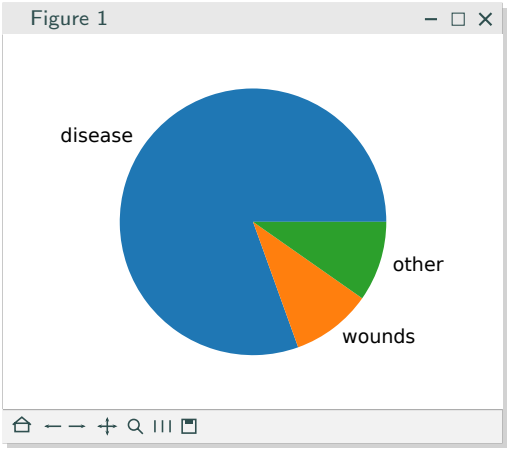


What Killed the Soldiers?

deaths in 2 years	
disease	**14,476**
wounds	1,758
other	1,748

```
counts = [14476, 1758, 1748]
names = ['disease', 'wounds', 'other']
plt.pie(counts, labels=names)
plt.show()
```

Read it: 8 of every 10 soldiers died of **disease**, not of wounds.

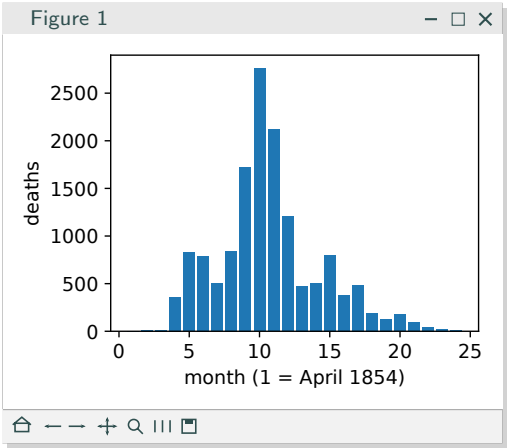


Month by Month

```
months  = [1,  2,  3,   4, ...]
disease = [1, 12, 11, 359, ...]
plt.bar(months, disease)
plt.xlabel('month (1 = April 1854)')
plt.ylabel('deaths')
plt.show()
```

bar: one bar for each month.

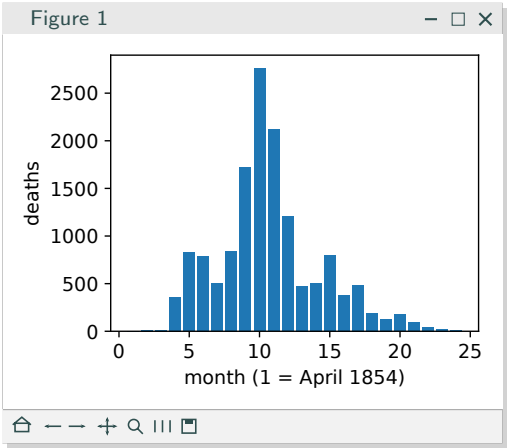
The first list says **where** each bar stands, the second how **tall** it is.



Month by Month

```
months  = [1,  2,  3,   4, ...]
disease = [1, 12, 11, 359, ...]
plt.bar(months, disease)
plt.xlabel('month (1 = April 1854)')
plt.ylabel('deaths')
plt.show()
```

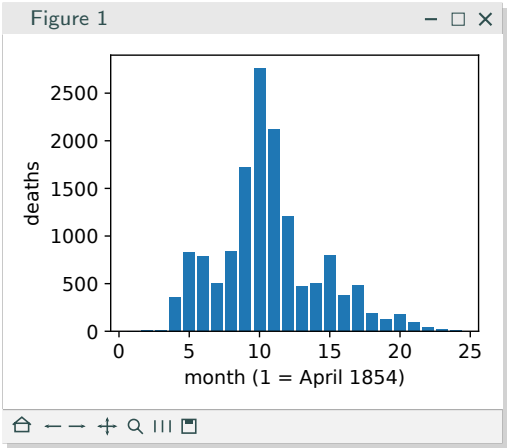
The bars stand in list order: month 1, then 2, then 3, ...



Month by Month

```
months  = [1,  2,  3,   4, ...]
disease = [1, 12, 11, 359, ...]
plt.bar(months, disease)
plt.xlabel('month (1 = April 1854)')
plt.ylabel('deaths')
plt.show()
```

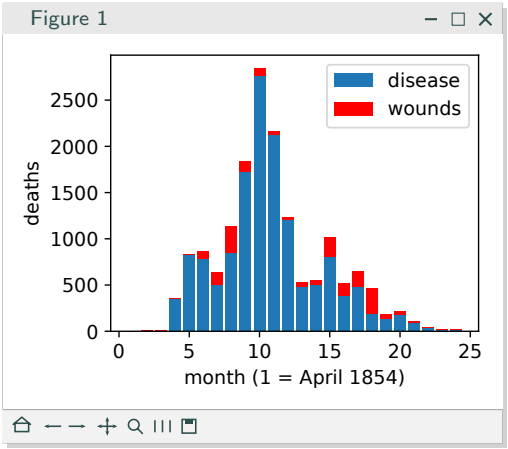
Read it: the worst month is month 10, January 1855: **2,761** dead of disease.



Bars on Bars

```
plt.bar(months, disease,
        label='disease')
plt.bar(months, wounds,
        bottom=disease,
        color='red', label='wounds')
plt.legend()
```

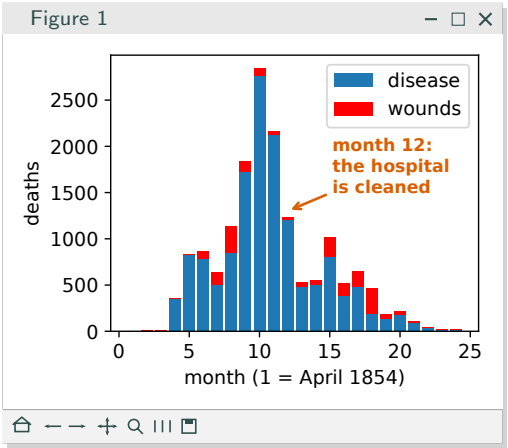
`bottom=disease`: each red bar **starts on top** of the blue bar of its month.



Bars on Bars

```
plt.bar(months, disease,
        label='disease')
plt.bar(months, wounds,
        bottom=disease,
        color='red', label='wounds')
plt.legend()
```

Read it: in March 1855 (month 12) the hospital is cleaned. A year later, almost nobody dies.

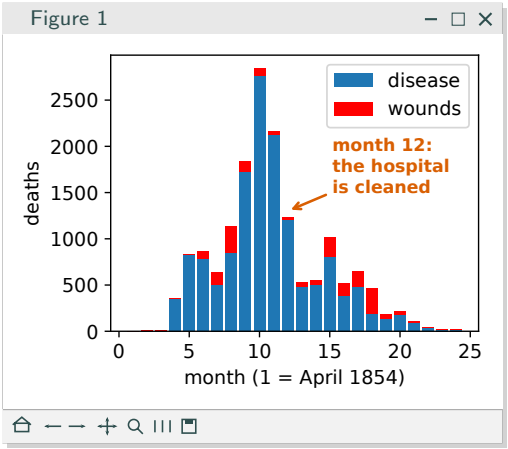


Bars on Bars

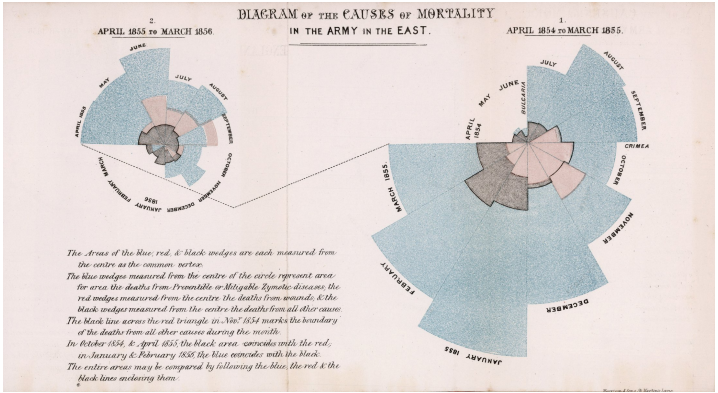
```
plt.bar(months, disease,
        label='disease')
plt.bar(months, wounds,
        bottom=disease,
        color='red', label='wounds')
plt.legend()
```

Read it: in March 1855 (month 12) the hospital is cleaned. A year later, almost nobody dies.

The red part is small in **every** month.



Her Own Picture, 1858



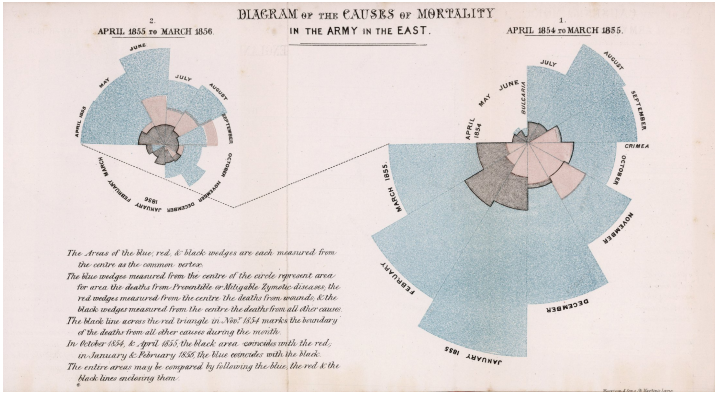
Nightingale, 1858. Right: the first year. Left: the second year.

One wedge = one **month**.

The **area** of a wedge = the deaths.

Blue: disease.
Red: wounds.
Black: other.

Her Own Picture, 1858



Nightingale, 1858. Right: the first year. Left: the second year.

One wedge = one **month**.

The **area** of a wedge = the deaths.

Blue: disease.
Red: wounds.
Black: other.

She sends it to the Queen and to the government.

Her Own Picture, 1858

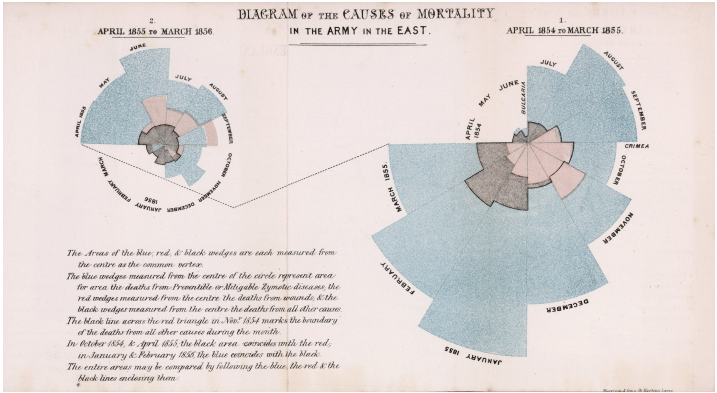
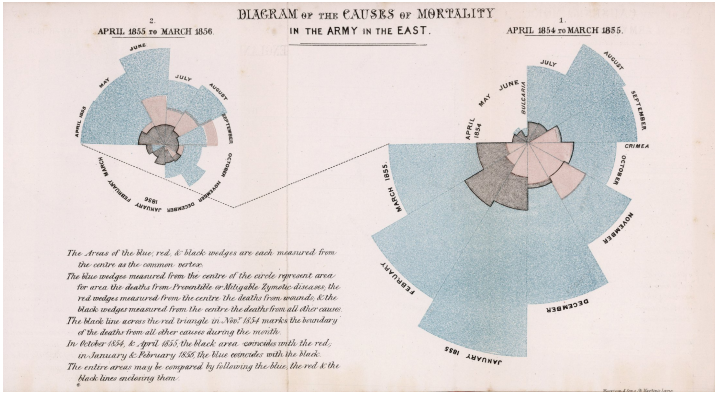


DIAGRAM OF THE CAUSES OF MORTALITY
IN THE ARMY IN THE EAST.
2.
APRIL 1855 TO MARCH 1856
1.
APRIL 1854 TO MARCH 1855.
JUNE
JULY
AUGUST
SEPTEMBER
OCTOBER
NOVEMBER
DECEMBER
JANUARY 1856
FEBRUARY
MARCH
APRIL
MAY
JUNE
JULY
AUGUST
SEPTEMBER
CRIMEA
OCTOBER
NOVEMBER
DECEMBER
JANUARY 1855
FEBRUARY
MARCH 1855
APRIL 1854
MAY
JUNE
JULY
AUGUST
SEPTEMBER
OCTOBER
NOVEMBER
DECEMBER
JANUARY 1855
FEBRUARY
MARCH
APRIL
MAY
JUNE
JULY
AUGUST
SEPTEMBER
OCTOBER
NOVEMBER
DECEMBER
JANUARY 1856
FEBRUARY
MARCH
APRIL
MAY
JUNE
JULY
AUGUST
SEPTEMBER
OCTOBER
NOVEMBER
DECEMBER
JANUARY 1856
FEBRUARY
MARCH
APRIL
MAY
JUNE
JULY
AUGUST
SEPTEMBER
OCTOBER
NOVEMBER
DECEMBER
JANUARY 1855
FEBRUARY
MARCH
APRIL

Her Own Picture, 1858



Nightingale, 1858. Right: the first year. Left: the second year.

One wedge = one **month**.

The **area** of a wedge = the deaths.

Blue: disease.
Red: wounds.
Black: other.

She sends it to the Queen and to the government.

Army hospitals are made clean.

A table is easy to ignore. **A picture is not.**

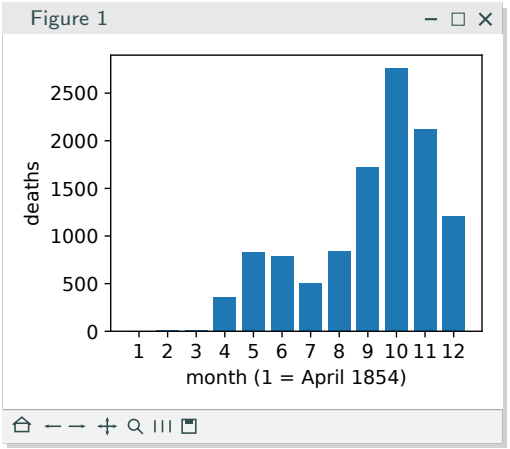
A Rose Diagram in Python

A rose diagram is a bar chart **bent around a circle**.

Each bar becomes a **wedge**.

Use it for data that **goes around**: months of a year, hours of a day.

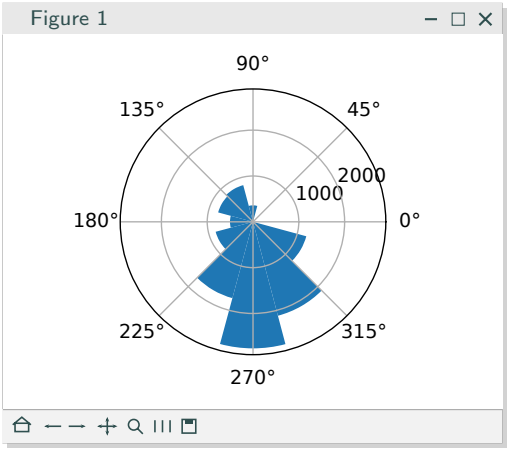
Here: the first year. 12 months, 12 bars.



A Rose Diagram in Python

```
step = 6.28 / 12
angles = []
for k in range(12):
    angles.append(k * step)
plt.axes(projection='polar')
plt.bar(angles, disease[:12],
        width=step)
plt.xticks(angles, months[:12])
plt.show()
```

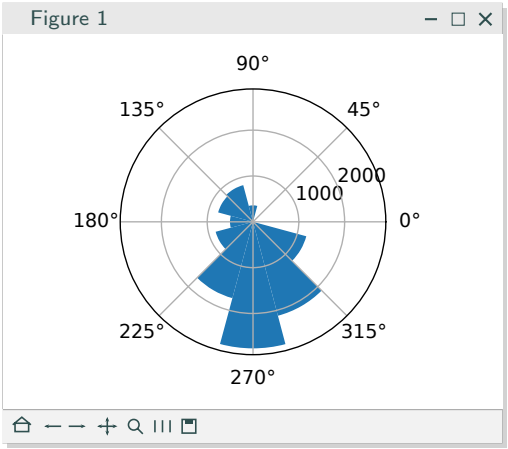
`projection='polar'`: a **round** plot. Now `bar` draws wedges.



A Rose Diagram in Python

```
step = 6.28 / 12
angles = []
for k in range(12):
    angles.append(k * step)
plt.axes(projection='polar')
plt.bar(angles, disease[:12],
        width=step)
plt.xticks(angles, months[:12])
plt.show()
```

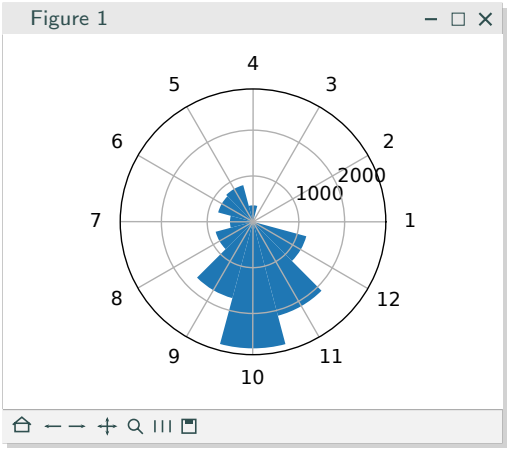
A full turn is **6.28**. The 12 wedges share it.
`angles`: where each wedge points.
`[:12]`: the first 12 months.



A Rose Diagram in Python

```
step = 6.28 / 12
angles = []
for k in range(12):
    angles.append(k * step)
plt.axes(projection='polar')
plt.bar(angles, disease[:12],
        width=step)
plt.xticks(angles, months[:12])
plt.show()
```

`xticks`: write the month number at each wedge.

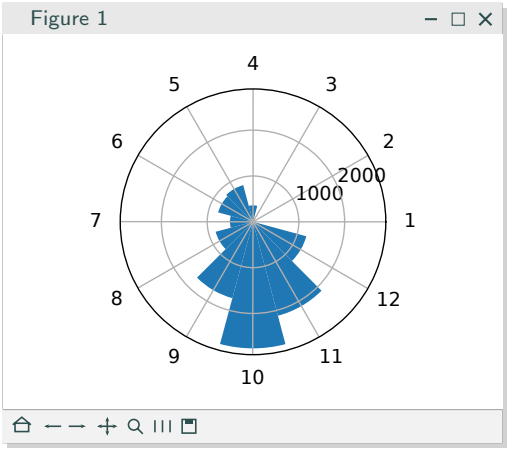


A Rose Diagram in Python

```
step = 6.28 / 12
angles = []
for k in range(12):
    angles.append(k * step)
plt.axes(projection='polar')
plt.bar(angles, disease[:12],
        width=step)
plt.xticks(angles, months[:12])
plt.show()
```

Read it: the same numbers as the bars. Month 10 is the longest.

Here: wedge **length** = deaths. Nightingale: **area**.



Story 5 of 5: Thirteen Tables

1 ✓ 
Water Pump
London 1854

2 ✓ 
Cold Launch
Florida 1986

3 ✓ 
24 Galaxies
California 1929

4 ✓ 
Army Hospital
Istanbul 1855

5 
13 Tables
2017

New tools: `hist` `subplot` `axis('off')`

Thirteen Tables

table 1		table 2		table 13	
x	y	x	y	x	y
32.33	61.41	53.37	90.21	55.38	97.18
53.42	26.19	52.80	90.09	51.54	96.03
63.92	30.83	47.05	90.46	46.15	94.49
70.29	82.53	42.45	89.51	42.82	91.41
34.12	45.73	42.70	90.44	40.77	88.33
...		...		...	

13 tables. Each has **142** points (x, y).

Thirteen Tables

table 1		table 2		table 13	
x	y	x	y	x	y
32.33	61.41	53.37	90.21	55.38	97.18
53.42	26.19	52.80	90.09	51.54	96.03
63.92	30.83	47.05	90.46	46.15	94.49
70.29	82.53	42.45	89.51	42.82	91.41
34.12	45.73	42.70	90.44	40.77	88.33
...		...		...	

13 tables. Each has **142** points (x, y).

	average x	average y
table 1	54.3	47.8
table 2	54.3	47.8
table 3	54.3	47.8
...	...	...
table 13	54.3	47.8

`sum(xs) / len(xs)`, for each table

Thirteen Tables

table 1		table 2		table 13	
x	y	x	y	x	y
32.33	61.41	53.37	90.21	55.38	97.18
53.42	26.19	52.80	90.09	51.54	96.03
63.92	30.83	47.05	90.46	46.15	94.49
70.29	82.53	42.45	89.51	42.82	91.41
34.12	45.73	42.70	90.44	40.77	88.33
...		...		...	

13 tables. Each has **142** points (x, y).

	average x	average y
table 1	54.3	47.8
table 2	54.3	47.8
table 3	54.3	47.8
...	...	...
table 13	54.3	47.8

`sum(xs) / len(xs)`, for each table

Same averages. So: the **same data**?

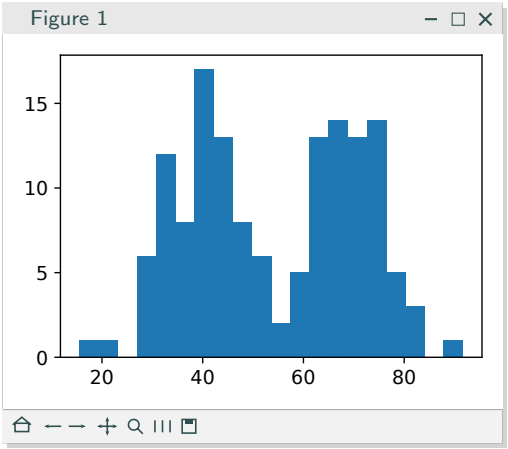
One List at a Time

`xs`: the 142 x values of **table 1**.

```
plt.hist(xs, bins=20)
plt.show()
```

`hist` takes **one** list. It makes piles and **counts** how many values fall in each.

`bins=20`: twenty piles.

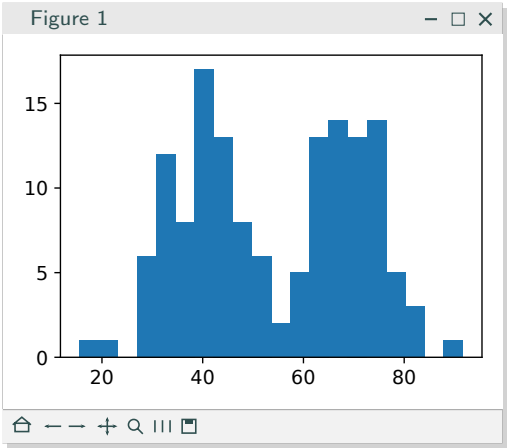


One List at a Time

`xs`: the 142 x values of **table 1**.

```
plt.hist(xs, bins=20)
plt.show()
```

Read it: table 1 has x values all the way from 15 to 92, in many piles.

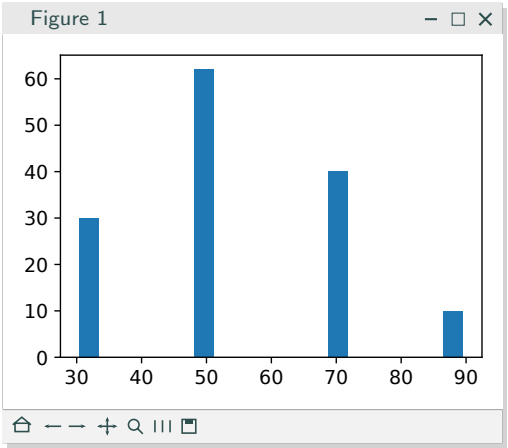


One List at a Time

`xs`: the 142 x values of **table 3**.

```
plt.hist(xs, bins=20)
plt.show()
```

Read it: table 3 has the **same** average x, 54.3. But its values sit in only four piles.



One List at a Time

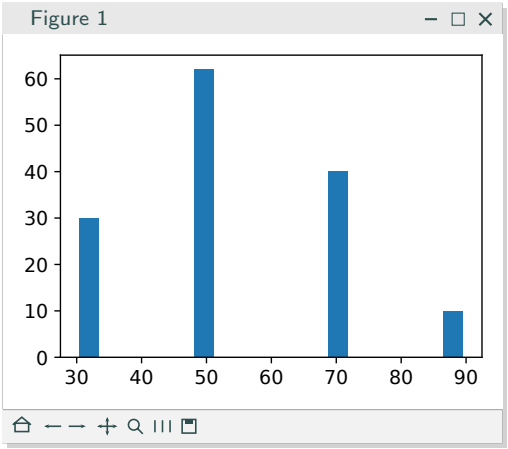
`xs`: the 142 x values of **table 3**.

```
plt.hist(xs, bins=20)
plt.show()
```

Read it: table 3 has the **same** average x, 54.3. But its values sit in only four piles.

Same average. Not the same list.

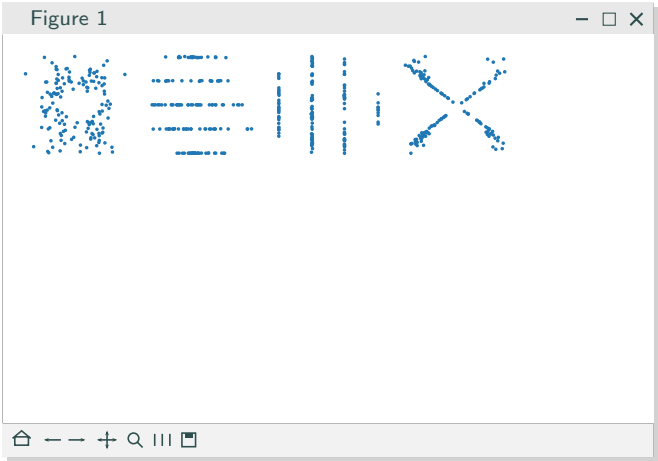
So let us look at all thirteen tables.



Thirteen Small Plots

```
k = 1
for name in tables:
    xs, ys = tables[name]
    plt.subplot(3, 5, k)
    plt.scatter(xs, ys, s=2)
    plt.axis('off')
    k += 1
plt.show()
```

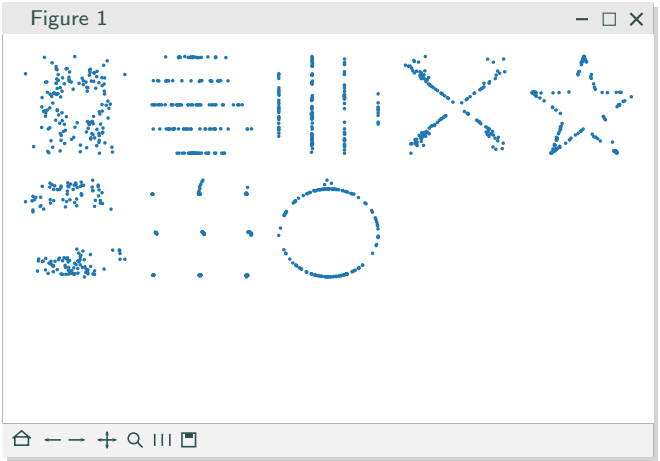
`subplot(3, 5, k)`: cut the window into 3 rows and 5 columns. Draw in place **k**.



Thirteen Small Plots

```
k = 1
for name in tables:
    xs, ys = tables[name]
    plt.subplot(3, 5, k)
    plt.scatter(xs, ys, s=2)
    plt.axis('off')
    k += 1
plt.show()
```

`axis('off')`: hide the axes. The plots are small.

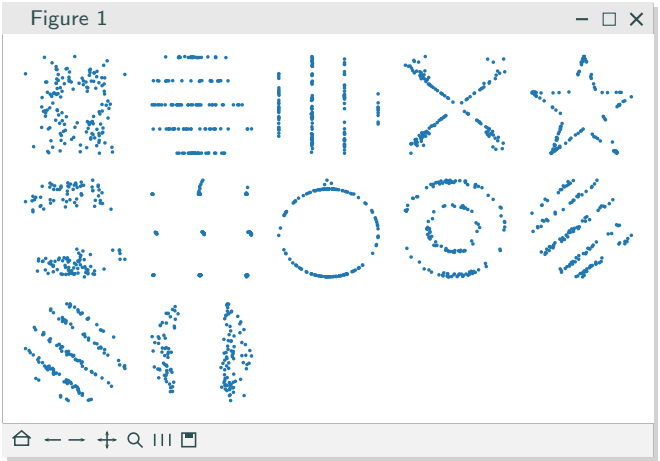


Thirteen Small Plots

```
k = 1
for name in tables:
    xs, ys = tables[name]
    plt.subplot(3, 5, k)
    plt.scatter(xs, ys, s=2)
    plt.axis('off')
    k += 1
plt.show()
```

Lines. A star. A circle.

All with the same two averages!



Thirteen Small Plots

```
k = 1
for name in tables:
    xs, ys = tables[name]
    plt.subplot(3, 5, k)
    plt.scatter(xs, ys, s=2)
    plt.axis('off')
    k += 1
plt.show()
```

And table 13?

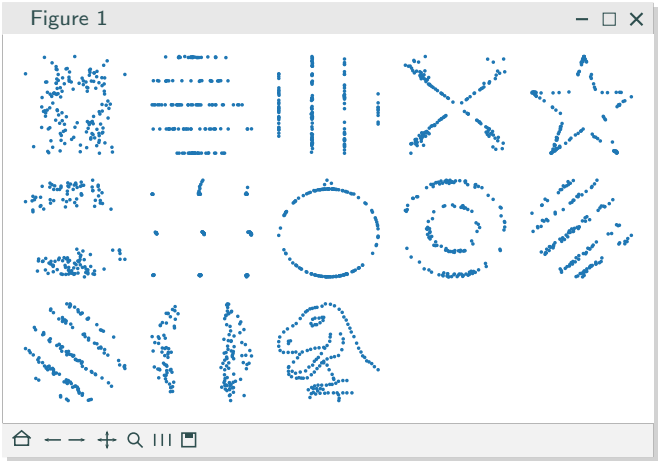
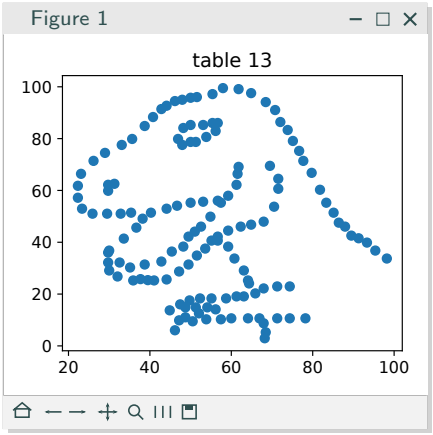


Table 13

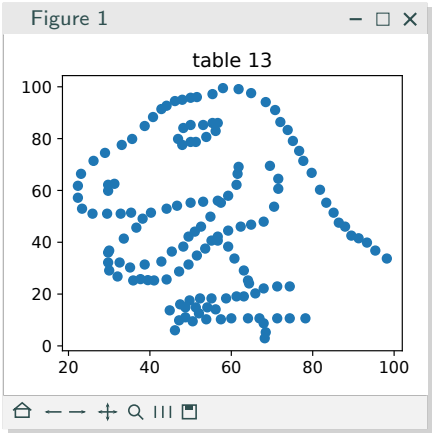


Average x: **54.3**

Average y: **47.8**

The same as the star, the circle and the lines.

Table 13



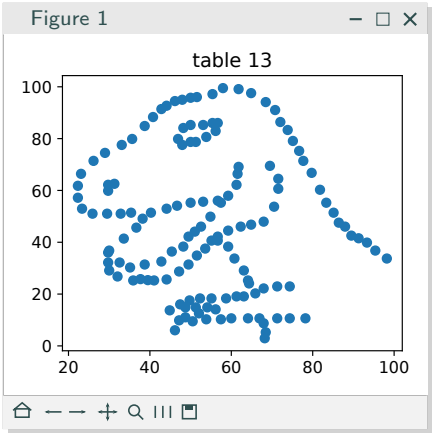
Average x: **54.3**

Average y: **47.8**

The same as the star, the circle and the lines.

An average hides the shape.

Table 13



Average x: **54.3**

Average y: **47.8**

The same as the star, the circle and the lines.

An average hides the shape.

Always **plot** your data.

Data: Cairo, 2016; Matejka and Fitzmaurice, 2017.

You Try! *(write the tool for each, 2 min)*


scatter


plot


hist


bar


pie

	the data	tool?
(a)	Lahore's temperature on each day of June	
(b)	The marks of 500 students: how do they spread?	
(c)	Height and weight of 50 people: do they go together?	
(d)	Your month's money: food, travel, phone, other	
(e)	Runs made by each of 11 players	

You Try! *(write the tool for each, 2 min)*


scatter


plot


hist


bar


pie

	the data	tool?
(a)	Lahore's temperature on each day of June	`plot`
(b)	The marks of 500 students: how do they spread?	
(c)	Height and weight of 50 people: do they go together?	
(d)	Your month's money: food, travel, phone, other	
(e)	Runs made by each of 11 players	

You Try! *(write the tool for each, 2 min)*


scatter


plot


hist


bar


pie

	the data	tool?
(a)	Lahore's temperature on each day of June	`plot`
(b)	The marks of 500 students: how do they spread?	`hist`
(c)	Height and weight of 50 people: do they go together?	
(d)	Your month's money: food, travel, phone, other	
(e)	Runs made by each of 11 players	

You Try! *(write the tool for each, 2 min)*


scatter


plot


hist


bar


pie

	the data	tool?
(a)	Lahore's temperature on each day of June	`plot`
(b)	The marks of 500 students: how do they spread?	`hist`
(c)	Height and weight of 50 people: do they go together?	`scatter`
(d)	Your month's money: food, travel, phone, other	
(e)	Runs made by each of 11 players	

You Try! *(write the tool for each, 2 min)*


scatter


plot


hist


bar


pie

	the data	tool?
(a)	Lahore's temperature on each day of June	`plot`
(b)	The marks of 500 students: how do they spread?	`hist`
(c)	Height and weight of 50 people: do they go together?	`scatter`
(d)	Your month's money: food, travel, phone, other	`pie`
(e)	Runs made by each of 11 players	

You Try! *(write the tool for each, 2 min)*


scatter


plot


hist


bar


pie

	the data	tool?
(a)	Lahore's temperature on each day of June	`plot`
(b)	The marks of 500 students: how do they spread?	`hist`
(c)	Height and weight of 50 people: do they go together?	`scatter`
(d)	Your month's money: food, travel, phone, other	`pie`
(e)	Runs made by each of 11 players	`bar`

Summary

Summary: Which Picture?

	`plt.scatter(xs, ys)`	**where**; do two lists go together?	Snow, Challenger
	`plt.plot(xs, ys)`	how it **changes**; a line	Snow, Hubble
	`plt.hist(values)`	how one list **spreads**	13 tables
	`plt.bar(xs, heights)`	**how many** in each group	Nightingale
	`plt.pie(counts)`	**parts** of one whole	Nightingale
	`plt.subplot(3, 5, k)`	**many** plots in one window	13 tables

Words: `title` `xlabel` `ylabel` `text` `label=` with `legend()`

Look: `s=` `marker=` `color=` `alpha=` `bins=` `bottom=` `xlim` `axvline` `polar`

Last line: `plt.show()` opens the window.

Big Idea

Snow:	a list of 578 addresses hid the pump.
Challenger:	7 launches hid what 23 showed.
Hubble:	24 rows hid one line.
Nightingale:	a table was ignored, a picture was not.
Table 13:	two averages hid a dinosaur.

Numbers hide. Plots show.

Next Time: Testing and Debugging

- 1947: a big computer stops working.

Next Time: Testing and Debugging

- 1947: a big computer stops working.
- Inside, they find a **moth**.



Next Time: Testing and Debugging

- 1947: a big computer stops working.
- Inside, they find a **moth**.
- They tape it in the notebook: *"first actual case of bug being found"*.



Questions?

Exit Ticket

Today's question *(2 minutes)*

(a) `xs` and `ys` are two lists of numbers. Write **two** lines that show them as dots, with no line.

(b) Sketch what this draws:

```
plt.plot([1, 2, 3], [3, 1, 3])
```

COMP 102 — Exit Ticket #24

Name: _____

Roll #: _____ Date: _____

Answer:

Hand it to me on your way out.
No name, no attendance.