**Population vs Sample, Bias**

As we get into the meaty part of the mathematical underpinnings of the course, here are a few takeways to keep in mind as you keep learning.

**Population vs Sample**

The core idea is the following: in practice, you almost never have access to the entire **population**, either because the data is unavailable, or because there are simply too many elements for you to study. Hence, you only take a portion, a **sample**, that you can study, and from which you can **estimate** what the total population looks like.

Remember the difference between the two situations we encounter:

* when we study the whole **population** (rarely in practice), the Greek letter �*μ* is used to denote the (population) mean, and �*N* (capital n) is used in the division.
* however, when we study only a **sample**, in order to make estimates about the population, �ˉ*x*ˉ is the name used for the (sample) mean, we divide by �*n*.

**What is a bias (and an unbiased estimate)?**

The sample mean is said to be an unbiased estimate of the population mean. What this means is that it is a good estimate of the population mean and can be proven mathematically. As Betty said, going deeper into the notion of a bias is beyond the scope of this MOOC. However, that does not mean you should not go and learn about it elsewhere (for instance, on [Wikipedia](https://en.wikipedia.org/wiki/Bias_of_an_estimator))!

The main idea here is the following: when taking a sample from a population, and then trying to figure out what the whole population looks like, it is more than likely that your sample actually does not reflect the entire population well! Take an oversimplified example: you have a population of three balls, and you sample only one from the three. Your three balls have all different colours, red, black and blue. If you only know the colour of your sample ball, let's say the blue one, it will obviously be biased in favour of that colour! In fact, it is so biased that you might think that all balls in your population are blue. You can now think of more general examples: a population of 100 people, the data in question being the size of each person. You can easily see that if your random sample happens to contain only tall, or only short people, your calculation could easily lead to a *biased* estimate (too tall, or too short) of the average person in your set.

Mathematicians have studied these notions rigorously, especially in the case where 1) your sampling technique is strictly random, and 2) you can theoretically apply that sampling method an infinite amount of times. Some of the results found, the mathematics of which again go somewhat too far for this class, will be useful to us later in this course, as they allow us to compensate for the potential bias arising from sampling. In preparation, you can already have a look at the one that will pop up in our calculations when looking at **variance** and **standard deviation** for samples, which is called [Bessel's correction](https://en.wikipedia.org/wiki/Bessel%27s_correction).

### Variability

There is more variability if your data is distributed far from the mean, less if the data points are close to it!

You can indeed have the same mean but high, or low, variability! As you can see when you run the code below, the values for x and y are not the same, yet the red line representing their respective means, is the same!

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

# two different arrays

x = np.array([2,3,5,6])

y = np.array([-8,-7,15,16])

# needed for the graph below

my\_range = np.arange(4)

# their respective means

mean\_x = np.mean(x)

mean\_y = np.mean(y)

fig, axs = plt.subplots(1,2, sharey=True)

axs[0].stem(my\_range, x, label='x values', bottom=mean\_x)

axs[1].stem(my\_range, y, label='y values', bottom=mean\_y)

legend\_x = axs[0].legend(loc='upper right')

legend\_y = axs[1].legend(loc='lower right')

plt.show()





RunReset

Remember this important idea: "the average of deviations is always zero as the sum of the deviations is always zero". And think about why that is. If you look at the above two charts, that may help you figure it out. Intuitively, if you are trying to measure the variability of your data around the mean, then by adding all these numbers you will necessarily find zero! (Simply because the mean is, you could say, the middle of all these deviations, where they cancel each other out).

We then use the square of the values to obtain more information: when you square a number, it always becomes positive! The cancellation of all deviations has been avoided. Now our deviations can tell us more about our data, and we can go on calculating the variance:

�2=∑(��−�ˉ)2�−1*s*2=*n*−1∑(*xi*​−*x*ˉ)2​.

When we are done with our calculation, and have computed the standard deviation, it is as if we had reached full circle, as we are now taking the square root in our equation, after having squared everything in the first place:

�=∑(��−�ˉ)2�−1*s*=*n*−1∑(*xi*​−*x*ˉ)2​​.

### Don't mix up the letters!

As before:

* when we talk about the **population**, the variance will be denoted by the Greek letter �2*σ*2, and we divide by �*N*;
* however, when we are dealing with a **sample**, then we use the letter �2*s*2 to denote the sample variance and we divide by �−1*n*−1.
* there are other versions where the sample variance is divided by �*n* but we do not use this version.

### The bias is back!

Some of you noticed that interesting �−1*n*−1 appearing in the division. As mentioned in the previous reading, this is called [Bessel's correction](https://en.wikipedia.org/wiki/Bessel's_correction). The mathematics for this is not included in this MOOC, but feel free to dig deeper into this, or post on the forum, if you feel like knowing more about it! The main idea to take away is that when you try to estimate the **variance** and/or the **standard deviation** of a population using a **random** **sample**, the result will be **biased**. This bias, which, by the way, carries no value judgement at all, but only indicates that if you repeat this technique an infinite number of time with different random samples, you will not, as you would hope, approximate the population values, but end up with something else, that can be calculated precisely. The division by �−1*n*−1 arises directly from these calculation (if you are hungry for the full details, [you can read them here](https://en.wikipedia.org/wiki/Bias_of_an_estimator#Sample_variance)). Mathematicians, then, having found out what exactly that bias is, deduced what method should be used to correct this, which is precisely to use �−1*n*−1 instead of �*n* in the division!

### Attention!

This will pop up again in this course, but it is good to mention it straight away: the need for this division by �−1*n*−1 only arises when calculating the ***variance*** and ***standard deviation***! If you only estimate the **mean** of a population using a sample, the division by �*n* is the only valid one.

The Jupyter environment is full of shortcuts and great things that make your life easier.

A quick list:

* press Enter to start typing into a cell, Escape to navigate your notebook (you can then press j or k, or the arrow keys, to go up and down;
* when navigating (not typing in a cell), you can press a to create a new cell above, b for below;
* if a cell is selected (but you are not typing in it), you can press m to turn it into a markdown cell, and y to make it a code cell;
* if you press Ctrl-Enter, you will run the cell and stay on it, whereas if you press Shift+Enter, you go one cell below (and if it is the last one, a new one will be created);
* if a cell is selected, you can press x to cut it, c to copy it, and then v to paste after the current cell, and V to paste before it;
* **Many** other shortcuts, which you can also customise, check in the menu above "Help" > "Keyboard Shortcuts".

## Markdown

Markdown is an easy way to write text in a web context without going for the full html syntax.

You can check out [this cheatsheet](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet).

Basic things:

* Separate paragraphs with an empty line, or two spaces at the end of the line;
* italic with \*asterisks\* or \_underscores\_, bold with \*\*two\*\* or \_\_two\_\_, (the two can be combined);
* lists, numbered or unnumbered, like so:

unnumbered list with a line starting with -

numbered list with 1., 2., etc.

* code snippets using this the grave accent `, multiline with three of them ```;
* titles (h1-6) using # starting the line, from just one to six;
* three stars \*\*\* or dashes --- give you a horizontal line like tihs one (it needs to be at the start of the line):
* links are generated like so: [text](link.com);
* images similarly: ![image text](image.jpg);

# Markdown Cheatsheet

Adam Pritchard edited this page on Mar 27, 2022 · [97 revisions](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet/_history)

### Pages 10



* [Home](https://github.com/adam-p/markdown-here/wiki)
* [Compatibility](https://github.com/adam-p/markdown-here/wiki/Compatibility)
* [Development Notes](https://github.com/adam-p/markdown-here/wiki/Development-Notes)
* [Markdown Cheatsheet](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet)
  + [Table of Contents](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#table-of-contents)
  + [Headers](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#headers)
  + [H1](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#h1)
  + [H2](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#h2)
  + [H3](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#h3)
  + [H4](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#h4)
  + [H5](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#h5)
  + [H6](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#h6)
  + [Alt-H1](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#alt-h1)
  + [Alt-H2](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#alt-h2)
  + [Emphasis](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#emphasis)
  + [Lists](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#lists)
  + [Links](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#links)
  + [Images](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#images)
  + [Code and Syntax Highlighting](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#code-and-syntax-highlighting)
  + [Footnotes](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#footnotes)
  + [Tables](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#tables)
  + [Blockquotes](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#blockquotes)
  + [Inline HTML](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#inline-html)
  + [Horizontal Rule](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#horizontal-rule)
  + [Line Breaks](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#line-breaks)
  + [YouTube Videos](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#youtube-videos)
* [Markdown Here Cheatsheet](https://github.com/adam-p/markdown-here/wiki/Markdown-Here-Cheatsheet)
* [Other Markdown Tools](https://github.com/adam-p/markdown-here/wiki/Other-Markdown-Tools)
* [Press, Posts, Reviews, Etc.](https://github.com/adam-p/markdown-here/wiki/Press,-Posts,-Reviews,-Etc.)
* [Reviews](https://github.com/adam-p/markdown-here/wiki/Reviews)
* [Tips and Tricks](https://github.com/adam-p/markdown-here/wiki/Tips-and-Tricks)
* [Troubleshooting](https://github.com/adam-p/markdown-here/wiki/Troubleshooting)

##### Clone this wiki locally



This is intended as a quick reference and showcase. For more complete info, see [John Gruber's original spec](http://daringfireball.net/projects/markdown/) and the [Github-flavored Markdown info page](http://github.github.com/github-flavored-markdown/).

Note that there is also a [Cheatsheet specific to Markdown Here](https://github.com/adam-p/markdown-here/wiki/Markdown-Here-Cheatsheet) if that's what you're looking for. You can also check out [more Markdown tools](https://github.com/adam-p/markdown-here/wiki/Other-Markdown-Tools).

##### Table of Contents

[Headers](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#headers)  
[Emphasis](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#emphasis)  
[Lists](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#lists)  
[Links](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#links)  
[Images](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#images)  
[Code and Syntax Highlighting](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#code)  
[Footnotes](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#footnotes)  
[Tables](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#tables)  
[Blockquotes](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#blockquotes)  
[Inline HTML](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#html)  
[Horizontal Rule](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#hr)  
[Line Breaks](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#lines)  
[YouTube Videos](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet#videos)

## Headers

# H1

## H2

### H3

#### H4

##### H5

###### H6

Alternatively, for H1 and H2, an underline-ish style:

Alt-H1

======

Alt-H2

------

# H1

## H2

### H3

#### H4

##### H5

###### H6

Alternatively, for H1 and H2, an underline-ish style:

# Alt-H1

## Alt-H2

## Emphasis

Emphasis, aka italics, with \*asterisks\* or \_underscores\_.

Strong emphasis, aka bold, with \*\*asterisks\*\* or \_\_underscores\_\_.

Combined emphasis with \*\*asterisks and \_underscores\_\*\*.

Strikethrough uses two tildes. ~~Scratch this.~~

Emphasis, aka italics, with asterisks or underscores.

Strong emphasis, aka bold, with **asterisks** or **underscores**.

Combined emphasis with **asterisks and underscores**.

Strikethrough uses two tildes.

## Lists

(In this example, leading and trailing spaces are shown with with dots: ⋅)

1. First ordered list item

2. Another item

⋅⋅\* Unordered sub-list.

1. Actual numbers don't matter, just that it's a number

⋅⋅1. Ordered sub-list

4. And another item.

⋅⋅⋅You can have properly indented paragraphs within list items. Notice the blank line above, and the leading spaces (at least one, but we'll use three here to also align the raw Markdown).

⋅⋅⋅To have a line break without a paragraph, you will need to use two trailing spaces.⋅⋅

⋅⋅⋅Note that this line is separate, but within the same paragraph.⋅⋅

⋅⋅⋅(This is contrary to the typical GFM line break behaviour, where trailing spaces are not required.)

\* Unordered list can use asterisks

- Or minuses

+ Or pluses

1. First ordered list item
2. Another item

* Unordered sub-list.

1. Actual numbers don't matter, just that it's a number
2. Ordered sub-list
3. And another item.

You can have properly indented paragraphs within list items. Notice the blank line above, and the leading spaces (at least one, but we'll use three here to also align the raw Markdown).

To have a line break without a paragraph, you will need to use two trailing spaces.  
Note that this line is separate, but within the same paragraph.  
(This is contrary to the typical GFM line break behaviour, where trailing spaces are not required.)

* Unordered list can use asterisks
* Or minuses
* Or pluses

## Links

There are two ways to create links.

[I'm an inline-style link](https://www.google.com)

[I'm an inline-style link with title](https://www.google.com "Google's Homepage")

[I'm a reference-style link][Arbitrary case-insensitive reference text]

[I'm a relative reference to a repository file](../blob/master/LICENSE)

[You can use numbers for reference-style link definitions][1]

Or leave it empty and use the [link text itself].

URLs and URLs in angle brackets will automatically get turned into links.

http://www.example.com or <http://www.example.com> and sometimes

example.com (but not on Github, for example).

Some text to show that the reference links can follow later.

[arbitrary case-insensitive reference text]: https://www.mozilla.org

[1]: http://slashdot.org

[link text itself]: http://www.reddit.com

[I'm an inline-style link](https://www.google.com/)

[I'm an inline-style link with title](https://www.google.com/)

[I'm a reference-style link](https://www.mozilla.org/)

[I'm a relative reference to a repository file](https://github.com/adam-p/markdown-here/blob/master/LICENSE)

[You can use numbers for reference-style link definitions](http://slashdot.org/)

Or leave it empty and use the [link text itself](http://www.reddit.com/).

URLs and URLs in angle brackets will automatically get turned into links. [http://www.example.com](http://www.example.com/) or [http://www.example.com](http://www.example.com/) and sometimes example.com (but not on Github, for example).

Some text to show that the reference links can follow later.

## Images

Here's our logo (hover to see the title text):

Inline-style:

![alt text](https://github.com/adam-p/markdown-here/raw/master/src/common/images/icon48.png "Logo Title Text 1")

Reference-style:

![alt text][logo]

[logo]: https://github.com/adam-p/markdown-here/raw/master/src/common/images/icon48.png "Logo Title Text 2"

Here's our logo (hover to see the title text):

Inline-style: alt text

Reference-style: alt text

## Code and Syntax Highlighting

Code blocks are part of the Markdown spec, but syntax highlighting isn't. However, many renderers -- like Github's and Markdown Here -- support syntax highlighting. Which languages are supported and how those language names should be written will vary from renderer to renderer. Markdown Here supports highlighting for dozens of languages (and not-really-languages, like diffs and HTTP headers); to see the complete list, and how to write the language names, see the [highlight.js demo page](http://softwaremaniacs.org/media/soft/highlight/test.html).

Inline `code` has `back-ticks around` it.

Inline code has back-ticks around it.

Blocks of code are either fenced by lines with three back-ticks ```, or are indented with four spaces. I recommend only using the fenced code blocks -- they're easier and only they support syntax highlighting.

```javascript

var s = "JavaScript syntax highlighting";

alert(s);

```

```python

s = "Python syntax highlighting"

print s

```

```

No language indicated, so no syntax highlighting.

But let's throw in a <b>tag</b>.

```

var s = "JavaScript syntax highlighting";

alert(s);

s = "Python syntax highlighting"

print s

No language indicated, so no syntax highlighting in Markdown Here (varies on Github).

But let's throw in a <b>tag</b>.

## Footnotes

Footnotes aren't part of the core Markdown spec, but they [supported by GFM](https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax#footnotes).

Here is a simple footnote[^1].

A footnote can also have multiple lines[^2].

You can also use words, to fit your writing style more closely[^note].

[^1]: My reference.

[^2]: Every new line should be prefixed with 2 spaces.

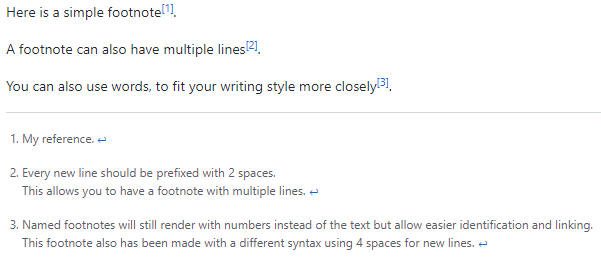
This allows you to have a footnote with multiple lines.

[^note]:

Named footnotes will still render with numbers instead of the text but allow easier identification and linking.

This footnote also has been made with a different syntax using 4 spaces for new lines.

Renders to:



## Tables

Tables aren't part of the core Markdown spec, but they are part of GFM and Markdown Here supports them. They are an easy way of adding tables to your email -- a task that would otherwise require copy-pasting from another application.

Colons can be used to align columns.

| Tables | Are | Cool |

| ------------- |:-------------:| -----:|

| col 3 is | right-aligned | $1600 |

| col 2 is | centered | $12 |

| zebra stripes | are neat | $1 |

There must be at least 3 dashes separating each header cell.

The outer pipes (|) are optional, and you don't need to make the

raw Markdown line up prettily. You can also use inline Markdown.

Markdown | Less | Pretty

--- | --- | ---

\*Still\* | `renders` | \*\*nicely\*\*

1 | 2 | 3

Colons can be used to align columns.

| **Tables** | **Are** | **Cool** |
| --- | --- | --- |
| col 3 is | right-aligned | $1600 |
| col 2 is | centered | $12 |
| zebra stripes | are neat | $1 |

There must be at least 3 dashes separating each header cell. The outer pipes (|) are optional, and you don't need to make the raw Markdown line up prettily. You can also use inline Markdown.

| **Markdown** | **Less** | **Pretty** |
| --- | --- | --- |
| Still | renders | **nicely** |
| 1 | 2 | 3 |

## Blockquotes

> Blockquotes are very handy in email to emulate reply text.

> This line is part of the same quote.

Quote break.

> This is a very long line that will still be quoted properly when it wraps. Oh boy let's keep writing to make sure this is long enough to actually wrap for everyone. Oh, you can \*put\* \*\*Markdown\*\* into a blockquote.

Blockquotes are very handy in email to emulate reply text. This line is part of the same quote.

Quote break.

This is a very long line that will still be quoted properly when it wraps. Oh boy let's keep writing to make sure this is long enough to actually wrap for everyone. Oh, you can put **Markdown** into a blockquote.

## Inline HTML

You can also use raw HTML in your Markdown, and it'll mostly work pretty well.

<dl>

<dt>Definition list</dt>

<dd>Is something people use sometimes.</dd>

<dt>Markdown in HTML</dt>

<dd>Does \*not\* work \*\*very\*\* well. Use HTML <em>tags</em>.</dd>

</dl>

*Definition list*

Is something people use sometimes.

*Markdown in HTML*

Does \*not\* work \*\*very\*\* well. Use HTML tags.

## Horizontal Rule

Three or more...

---

Hyphens

\*\*\*

Asterisks

\_\_\_

Underscores

Three or more...

Hyphens

Asterisks

Underscores

## Line Breaks

My basic recommendation for learning how line breaks work is to experiment and discover -- hit <Enter> once (i.e., insert one newline), then hit it twice (i.e., insert two newlines), see what happens. You'll soon learn to get what you want. "Markdown Toggle" is your friend.

Here are some things to try out:

Here's a line for us to start with.

This line is separated from the one above by two newlines, so it will be a \*separate paragraph\*.

This line is also a separate paragraph, but...

This line is only separated by a single newline, so it's a separate line in the \*same paragraph\*.

Here's a line for us to start with.

This line is separated from the one above by two newlines, so it will be a separate paragraph.

This line is also begins a separate paragraph, but...  
This line is only separated by a single newline, so it's a separate line in the same paragraph.

(Technical note: Markdown Here uses GFM line breaks, so there's no need to use MD's two-space line breaks.)

## YouTube Videos

They can't be added directly but you can add an image with a link to the video like this:

<a href="http://www.youtube.com/watch?feature=player\_embedded&v=YOUTUBE\_VIDEO\_ID\_HERE

" target="\_blank"><img src="http://img.youtube.com/vi/YOUTUBE\_VIDEO\_ID\_HERE/0.jpg"

alt="IMAGE ALT TEXT HERE" width="240" height="180" border="10" /></a>

Or, in pure Markdown, but losing the image sizing and border:

[![IMAGE ALT TEXT HERE](http://img.youtube.com/vi/YOUTUBE\_VIDEO\_ID\_HERE/0.jpg)](http://www.youtube.com/watch?v=YOUTUBE\_

**Multidimensional Data Points and Features Recap**

A few quick takeaways:

A **matrix** is a rectangular array of numbers, a way of packing numbers, or arrays of numbers (vectors) together, like so:

�=(12345678)**X**=⎝⎜⎜⎜⎛​1357​2468​⎠⎟⎟⎟⎞​

A **vector** can be seen as a matrix with either one row, or one column:

�=(46242)**Y**=⎝⎜⎜⎜⎜⎜⎛​46242​⎠⎟⎟⎟⎟⎟⎞​,

�=(46242)**Z**=(4​6​2​4​2​).

In the example presented at the end of the previous video, we had the marks of our students packed in one matrix, the first column being the English score, the second the Maths score:

�=(�1,�2)=(65557052454261343731)**X**=(*X*1​,*X*2​)=⎝⎜⎜⎜⎜⎜⎛​6570456137​5552423431​⎠⎟⎟⎟⎟⎟⎞​

We can now read the matrix either horizontally or line by line, that is, student by student, or we can do so vertically, looking at the English (the vector �1*X*1​) or Maths scores (the vector �2*X*2​).

The general matrix is: �=(�11⋯�1�⋮⋮⋮��1⋯���)=(�1,�2,...,��)*X*=⎝⎜⎜⎛​*x*11​⋮*xn*1​​⋯⋮⋯​*x*1*p*​⋮*xnp*​​⎠⎟⎟⎞​=(*X*1​,*X*2​,...,*Xp*​), where we would have �*p* different scores per student, so �*p* score vectors. In an abstract case mathematicians prefer to call �*p* using the letter �*j*, so we will switch to that now. Since we have �*n* students (observations) per subject (variable), our vector corresponding to each variable is:

��=(�1�⋮���)*Xj*​=⎝⎜⎜⎛​*x*1*j*​⋮*xnj*​​⎠⎟⎟⎞​

# Multidimensional Mean Recap

### Don't mix up the letters (once more)!

Remember, when we talk about the **population** (all data points), the mean is denoted by the Greek letter �*μ*, and we divide by the total number of elements �*N*, and the standard deviation is denoted by the Greek letter �*σ*. However, when we work with a **sample**, we use �ˉ*x*ˉ as a name for the mean, we divide by �*n*, and our standard deviation will be called �*s* (and in this case, don't forget, we divide by (�−1)(*n*−1) in order to calculate the estimate).

In vector form, when we pack everything into arrays for more efficient calculation, we obtain the following:

�=(�1�2�3)*μ*​*μ*=⎝⎜⎛​*μ*1​*μ*2​*μ*3​​⎠⎟⎞​ (the **population** mean vector), and

�ˉ=(�ˉ1�ˉ2�ˉ3)*x*ˉ*x*ˉ=⎝⎜⎛​*x*ˉ1​*x*ˉ2​*x*ˉ3​​⎠⎟⎞​ (the **sample** mean vector).

### Attention!

In the previous video, at 0.56, since we estimate the population **mean** through a sample, and not the **variance** or **standard deviation**, we divide by �*n* ((�−1)(*n*−1) is necessary only for the variance or standard deviation).

As earlier in the course, the notion of **bias** refers to the fact that if you try and estimate information about the **population** by doing calculation only on a **sample**, your calculation may not lead to an approximation of the desired value. Hence this distinction that you need to remember:

* for the **mean**, as seen in the previous video, the estimate is **unbiased**, which means dividing by �*n* will give you the right result;
* but, as mentioned already, if you are trying to estimate the **variance** or **standard deviation**, you have to use [Bessel's correction](https://en.wikipedia.org/wiki/Bessel%27s_correction), that is, you have to divide by (�−1)(*n*−1) instead of only by �*n*, which you could do if you had all the elements of your population available.

# Multidimensional Variables Recap

### Variance and Standard Deviation

Here are our two formulas:

��2=∑�=1�(���−��)2�*σj*2​=*Ni*=1∑*N*​(*xij*​−*μj*​)2​, for the **population**, and

��2=∑�=1�(���−�ˉ�)2(�−1)=1(�−1)∑�=1�(���−�ˉ�)2*sj*2​=(*n*−1)*i*=1∑*n*​(*xij*​−*x*ˉ*j*​)2​=(*n*−1)1​*i*=1∑*n*​(*xij*​−*x*ˉ*j*​)2 for a **sample**.

Let me unpack this a bit:

You remember the concept of **deviation**, which tells you how distant each variable is from the mean (remember that you can have two sets of numbers with the same mean, one with numbers very spread out, the other with all of them very close to the mean), and calculated like so: (��−�)(*xi*​−*μ*) for the population, (��−�ˉ)(*xi*​−*x*ˉ) for a sample.

When we wanted to calculate the average of all our deviations, which will give us the **variance** and then the **standard deviation**, we saw that they all summed up to zero, and so we squared them to cancel that, then divided by the infamous (�−1)(*n*−1) (I'll focus on the sample case from now on), using the following equations:

�2=∑(��−�ˉ)2(�−1)*s*2*s*2=(*n*−1)∑(*xi*​−*x*ˉ)2​, the **variance**, and

�=�2=∑(��−�ˉ)2(�−1)*ss*=*s*2​=(*n*−1)∑(*xi*​−*x*ˉ)2​​, the **standard deviation**.

As you can see, these are the equations above, only without the �=1*i*=1 and the �*n* (like so: ∑�=1�*i*=1∑*n*​) specifying that we are working on �*n* observations!

### Covariance

The concept of [covariance](https://en.wikipedia.org/wiki/Covariance) expresses how much variables are linked, or vary, together. Intuitively, you can imagine yourself changing one individual data point: is that change going to affect another one? If you increase this value, will that other one also increase, remain unchanged, or decrease? That is what we are trying to assess in this calculation. The equation is the following:

���=∑�=1�(���−�ˉ�)(���−�ˉ�)(�−1)=1(�−1)∑�=1�(���−�ˉ�)(���−�ˉ�)*sjk*​=(*n*−1)*i*=1∑*n*​(*xij*​−*x*ˉ*j*​)(*xik*​−*x*ˉ*k*​)​=(*n*−1)1​*i*=1∑*n*​(*xij*​−*x*ˉ*j*​)(*xik*​−*x*ˉ*k*​)

We take the difference of each of the two variables with the mean, multiply them (which will give us an idea of how one reacts if the other is changed), and then take the unbiased average of all that. The result is, as mentioned in the video:

* if ���=0*sjk*​=0 the two variables are **uncorrelated**;
* if ���>0*sjk*​>0 the two variables are **positively correlated** (they "move in the same direction", that is, if one increases the other one as well);
* if ���<0*sjk*​<0 the two variables are **negatively correlated** (they "move in contrary motion", if one increases the other one shrinks, and vice versa).

### The Variance-Covariance Matrix

When we work with two vectors containing each �*p* observations, we can pack all the values into the following matrix, called the **variance-covariance matrix**:

Σ=(�12�12⋯�1��21�22⋯�2�⋮⋮⋱⋮��1��2⋯��2)Σ=⎝⎜⎜⎜⎜⎛​*σ*12​​*σ*12​*σ*21​⋮*σp*1​​*σ*12​*σ*22​​*σ*22​⋮*σp*2​​⋯⋯⋱⋯​*σ*1*p*​*σ*2*p*​⋮*σp*2​​*σp*2​​⎠⎟⎟⎟⎟⎞​ for the **population**, and

�=(�12�12⋯�1��21�22⋯�2�⋮⋮⋱⋮��1��2⋯��2)*s*=⎝⎜⎜⎜⎜⎛​*s*12​​*s*12​*s*21​⋮*sp*1​​*s*12​*s*22​​*s*22​⋮*sp*2​​⋯⋯⋱⋯​*s*1*p*​*s*2*p*​⋮*sp*2​​*sp*2​​⎠⎟⎟⎟⎟⎞​ for a **sample**.

As seen in the previous video, you find the **variance** in the diagonal (indicated by the indices), whereas all the other slots are filled with the covariances between the different values. Why is that? Well, if you plug in twice the same value for the two different variables in the covariance equation above, you can see that this actually turns to be the equation of the variance, even further up on the page! It's as if the variance was a measure of the variability of one element with itself (as you can see [here](https://en.wikipedia.org/wiki/Covariance#Covariance_with_itself)). Food for thought!

### Distance metrics

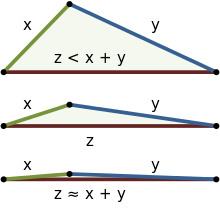
A **distance metric** is what we use to calculate nearness. It is an abstract generalisation of the concept of distance we use every day. The three properties required for a function on a set �*X* defined as �:�×�⟶�*d*:*X*×*X*⟶R to be called a **distance function**, are the following, for all �,�∈�*x*,*y*∈*X*:

* �(�,�)≥0*d*(*x*,*y*)≥0, or **non-negativity property**;
* �(�,�)=0*d*(*x*,*y*)=0 if and only if �=�*x*=*y*;
* �(�,�)=�(�,�)*d*(*x*,*y*)=*d*(*y*,*x*), or **symmetry.**

A distance function is a **metric** if, in addition to the above, it satisfies the following formula:

* �(�,�)+�(�,�)≥�(�,�)*d*(*x*,*y*)+*d*(*y*,*z*)≥*d*(*x*,*z*), called the **triangle inequality.**

Intuitively, you can see this as saying that the distance function is the shortest path between two points. In this case, if you take the distance between �*x* and �*z*, it will be the shortest path, namely: if to go from �*x* to �*z* you pass through another point, �*y*, and you add the two distances from �*x* to �*y* and from �*y* to �*z*, then it can either be the same distance, or greater. You can read more about it [here](https://en.wikipedia.org/wiki/Triangle_inequality). Here is a visual representation of this (from that same Wikipedia page):



Finally, when you take your set �*X* and consider it together with the distance function �*d*, that pair (�,�)(*X*,*d*) is called a **metric space**.

### Types of Distances

The big equation we encountered was this one (I omit the ∑�=1�*i*=1∑*m*​ for clarity):

��(�,�)=(∑∣��−��∣�)1/�=∑∣��−��∣�)�*Lp*​(*x*,*y*)=(∑∣*xi*​−*yi*​∣*p*)1/*p*=*p*∑∣*xi*​−*yi*​∣*p*)​.

If instead of �*p* you plug in a number, you can get different types that are frequently used in calculations:

* �=1*p*=1 gives you �1*L*1​, the [Manhattan distance](https://en.wikipedia.org/wiki/Taxicab_geometry), (∑∣��−��∣1)1/1=∑∣��−��∣(∑∣*xi*​−*yi*​∣1)1/1=∑∣*xi*​−*yi*​∣;
* �=2*p*=2 gives you �2*L*2​, the [Euclidean distance](https://en.wikipedia.org/wiki/Euclidean_distance#Definition), (∑∣��−��∣2)1/2=∑∣��−��∣2(∑∣*xi*​−*yi*​∣2)1/2=∑∣*xi*​−*yi*​∣2​;
* �=∞*p*=∞ gives you �∞*L*∞​, the [Infinity distance](https://en.wikipedia.org/wiki/Distance#Distance_in_Euclidean_space) in this case is max⁡�≥1∣��−��∣max*i*≥1​∣*xi*​−*yi*​∣.

We are mostly interested in the **Euclidean distance**!

Finally, notice that unpacking the �2*L*2​ distance gives you the following:

�2=(∑�=1�∣��−��∣2)1/2=(�1−�1)2+(�2−�2)2+⋯+(��−��)2*L*2​=(*i*=1∑*m*​∣*xi*​−*yi*​∣2)1/2=(*x*1​−*y*1​)2+(*x*2​−*y*2​)2+⋯+(*xm*​−*ym*​)2​.

That will be how we will calculate i

**Normalisation**, also known as **standardisation**, is a way of applying the same scale of measurement to your data: imagine you have measured two features for each data point, but one varies between 0 to 1000, while the other is only from 0 to 2. If you start adding these two together and making calculations, you can see that any change in the first ones will weight far more than in the second ones. This is very obvious when we think about proportions: your first data point x could have 900 for its first feature, and 1 for the second one. If suddenly the first one were to be halved, you would have 450 there, whereas the other one would still be 1. Conversely, if the second variable is halved, it is only going to become 0.5 (whereas the first one would remain 900). The second one is so small that its changes won't really be visible when you compute using these two together. To correct that, you transform them so that they are measured on the same scale, meaning, for instance, that all values should fall between 0 and 1, shrinking or increasing your variables, but respecting all proportions (in our case, the first variable would become 0.9, and the second 0.5). In this case, halving one variable (to 0.45) or the other (to 0.25) will have a similar effect on the overall result!

The method is the following: you **subtract the mean**, and **divide by the standard deviation**. Here is the equation:

����=���−�ˉ���*XijZ*​=*sj*​*xij*​−*x*ˉ*j*​​.

This will transform your data so that it now has a **mean of 0** and **a variance of 1.** These values are called z-scores (you can read more about them [here](https://en.wikipedia.org/wiki/Standard_score)), and make sure that the impact is the same for each feature on your calculations (even if you measure very large things, and then very small things, and want to work with all of them at the same time).

If on the contrary you would rather have them distributed between 0 and 1, in the domain [0,1][0,1], the following equation should be used:

����=���−min⁡����max⁡����−min⁡����*XijU*​=*i*max​*xij*​−*i*min​*xij*​*xij*​−*i*min​*xij*​​.

Very often in Data Science, and especially using the K-means algorithm, we have different features that are very different in scale (for instance, one feature might be the age of a person, and another the average number of coffee cups drunk each day). When trying to make calculations on such datasets, it is important to think about scale (in our example, any variation in age will weigh a lot, as ages will be greater numbers than the number of daily cups), and transform our data into a unified scale by **applying normalisation** before using other algorithms (after which, for instance, both our measurements would go from 0 to 1, preserving proportions, and allowing easier calculation between the two sets of features).

### Examples

Normalisation using one method or another is widely used in Data Science and machine learning. Here is a little list of examples where min max scaling is used, taken from [here](http://rasbt.github.io/mlxtend/user_guide/preprocessing/minmax_scaling/):

* "k-nearest neighbors with an Euclidean distance measure if want all features to contribute equally
* k-means (see k-nearest neighbors)
* logistic regression, SVMs, perceptrons, neural networks etc. if you are using gradient descent/ascent-based optimization, otherwise some weights will update much faster than others
* linear discriminant analysis, principal component analysis, kernel principal component analysis since you want to find directions of maximizing the variance (under the constraints that those directions/eigenvectors/principal components are orthogonal); you want to have features on the same scale since you'd emphasize variables on "larger measurement scales" more."

### Try it in Python!

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

my\_data = np.array([55, 67, 28, 235, 114])

x\_axis = np.arange(len(my\_data))

my\_mean = my\_data.mean()

my\_stdev = my\_data.std()

# normalisation: the data will now have

# a mean of 0 and a standard deviation of 1

my\_data\_normed = (my\_data - my\_mean) / my\_data.std()

# domain standardisation: the data points will all lie

# between 0 (smallest one) and 1 (largest one)

my\_data\_domain\_standardised = (my\_data - my\_data.min()) / (my\_data.max() - my\_data.min())

# Have a look at the y axis for each figure!

fig, (ax1, ax2, ax3) = plt.subplots(1,3, figsize=(10,5))

ax1.plot(x\_axis,my\_data)

ax1.set\_title('Original')

ax2.plot(x\_axis,my\_data\_normed)

ax2.set\_title('Normalised \n(mean = 0, stddev = 1)')

ax3.plot(x\_axis,my\_data\_domain\_standardised)

ax3.set\_title('Domain standardised \n(data points between 0 and 1)')

plt.show()





RunReset

# 3.12 Errata

At around 4'10 in the previous video, you can see Matthew being fully immersed into the coding and defining variables called 'min' and 'max'. This is in fact considered bad practice, and he would certainly correct it once rereading his code (don't worry if that happens to you, it is something any coder, even seasoned ones, knows well, hence the importance of writing clear and clean code and proofreading yourself!). The problem happening here is that the variable names are the same as the two Python built-in functions [min()](https://docs.python.org/3.5/library/functions.html#min) and [max()](https://docs.python.org/3.5/library/functions.html#max), that we have used in the course. Once you define variables with these names, the built-in fuctions will no longer be accessible!

Try that in your code, it is a good experiment (you can run all three examples below).

1. min() can be used as a function:

1

2

x = [1,2,3]

print(min(x))





RunReset

2. Min is redefined as a variable, it can be printed out:

1

2

3

x = [1,2,3]

min = 5

print(min) # min now is a variable holding the value 5!





RunReset

3. Min is now a variable, it cannot be used as a function any more!

1

2

3

4

x = [1,2,3]

min = 5

print(min(x)) # since min is a variable with value 5, the min() function can no longer be accessed!





RunReset

As you can see, it is better not to use the same names for your va