The efficient toolbox of the Computational Scientist

Dr. Gábor Závodszky - _Computational Science Lab_ https://github.com/gzavo/CS_Assignment

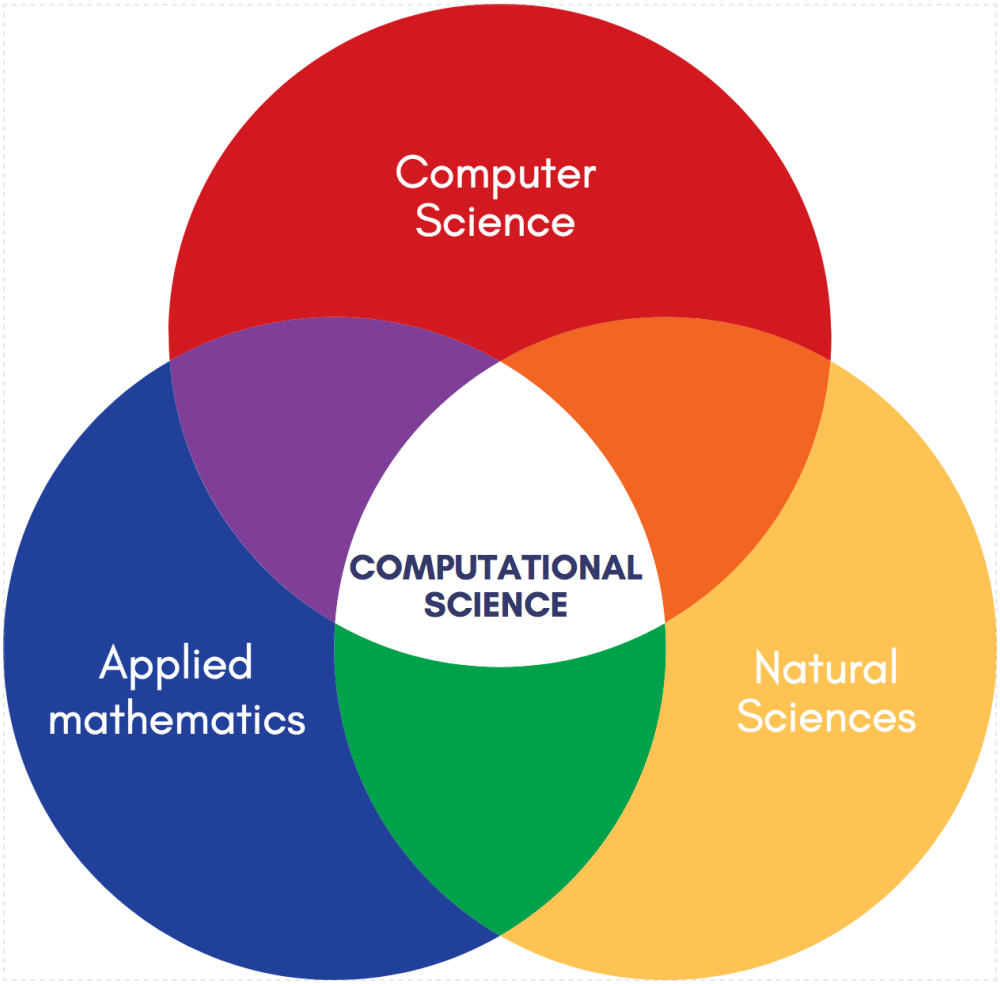
> "Never do a live demo" -- Every presenter ever

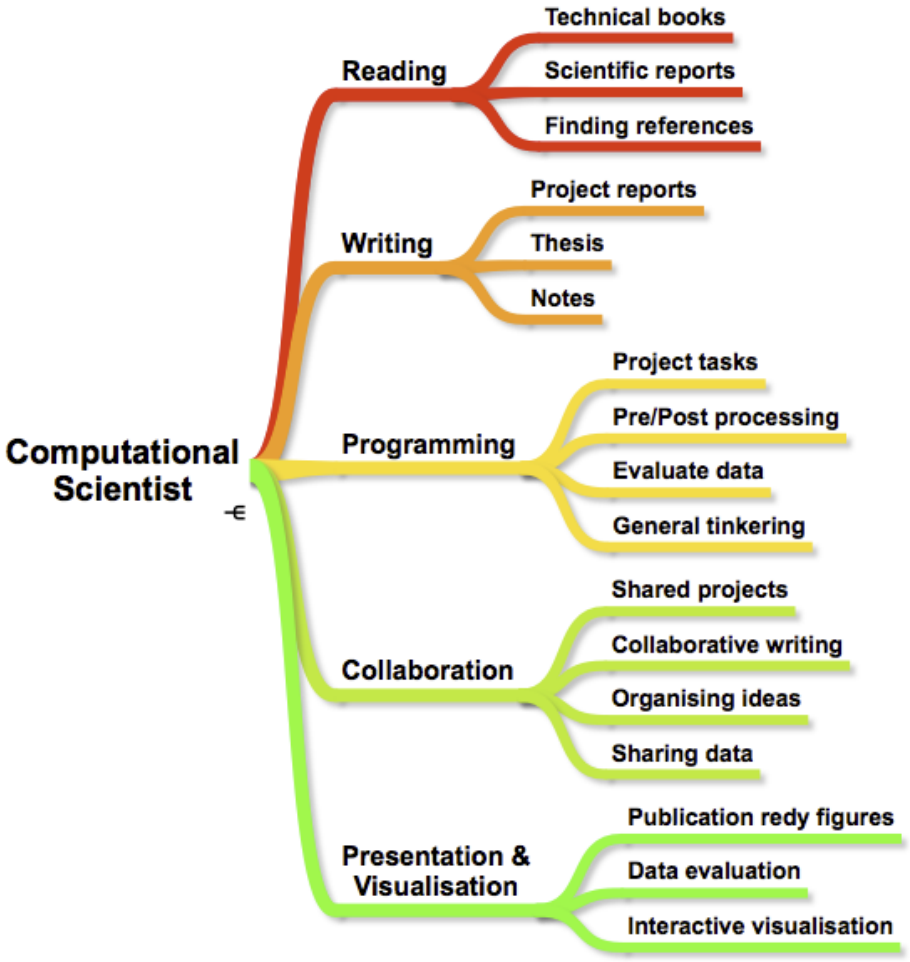
Structure of the talk

- Who is "The Computational Scientist" and what does he eat?
- Scientific writing and reading
- Programming languages
- Visualization 101
- Software engineering principles
- Homework assingment

- Can be an information overload!
- Some things are a must: (!)
- Some things are there to hear at least once.

Who is the Computational Scientist?


Computer
Science
COMPUTATIONAL
SCIENCE
Applied
mathematics
Natural
Sciences



Reading - Literature review

- Library (ofc.) (http://uba.uva.nl/en (http://uba.uva.nl/en))

- Google Scholar (!) - https://scholar.google.com (https://scholar.google.com)
  - Search for papers
  - export references
  - look at researcher profiles
  - E.g.: "Chickens prefer beautiful humans"
- Web of Science, Scopus, ...

- Managing references: Mendeley, Zotero, Papers, etc.
  - Sync. paper database
  - Annotate pdfs, add notes and sync them as well

Writing

- LaTeX
  - Overleaf (!) - https://www.overleaf.com/ (https://www.overleaf.com/)
  - TexMaker
  - TeXmacs
  - latex2png - http://latex2png.com/ (http://latex2png.com/)

-> Why is LaTeX useful?

- Not a WYSIWYG solution
- But often WYGIWYN
- Like a programming language for word processing


```
from IPython.display import IFrame

IFrame("https://www.nature.com/news/scientific-writing-the-online-cooperative-1.16039 (https://www.nature.com/news/scientific-writing-the-online-cooperative-1.16039)", "100%", 600)
```

- Markdown (!)
  - Use Pandoc (!) to turn it to .doc, .pdf, .html, you name it (https://pandoc.org/ (https://pandoc.org/))
  - Typora (https://typora.io/ (https://typora.io/))
  - Basically everywhere (websites, forums, editors...really, everywhere)
  - Supersets / alternatives (ASCIIDOC, ...)

Markdown

Write equations: $e^{i\pi} + 1 = 0$

Embed code:

```
def f(x):
    """docstring of this very useful function"""
    return x**2
```

Add tables:

This	is
a	table

Create lists:

- list item 1
- list item 2
- list item 3

Markdown is a simple way to format text that looks great on any device. It doesn't do anything fancy like change the font size, color, or type — just the essentials, using keyboard symbols you already know.

TRY OUR 10 MINUTE MARKDOWN TUTORIAL

Type	Or	... to Get
`*Italic*`	`_Italic_`	*Italic*
`**Bold**`	`__Bold__`	**Bold**
`# Heading 1`	`Heading 1` `=========`	Heading 1
`## Heading 2`	`Heading 2` `---------`	Heading 2

`[Link](http://a.com)`	`[Link][1]`
	⋮
	`[1]: http://b.org`
		Link

`![Image](http://url/a.png)`	`![Image][1]`
	⋮
	`[1]: http://url/b.jpg`
		

`> Blockquote`		Blockquote
`* List` `* List` `* List`	`- List` `- List` `- List`	• List • List • List
`1. One` `2. Two`	`1) One` `2) Two`	1. One 2. Two

from IPython.display import IFrame

IFrame("http://commonmark.org/help/ (http://commonmark.org/help/)", "100%", 600)

Writing - cont.

Pandoc a universal document converter

Donate Flattr

- About
- Installing
- Getting started
- Demos
- Documentation
- Help
- Extras
- Releases

About pandoc

If you need to convert files from one markup format into another, pandoc is your swiss-army knife. Pandoc can convert documents in (several dialects of) Markdown, reStructuredText, textile, HTML, DocBook, LaTeX, MediaWiki markup, TWiki markup, TikiWiki markup, Creole 1.0, Vimwiki markup, OPML, Emacs Org-Mode, Emacs Muse, txt2tags, Microsoft Word docx, LibreOffice ODT, EPUB, or Haddock markup to

HTML formats

XHTML, HTML5, and HTML slide shows using Slidy, reveal.js, Slideous, S5, or DZSlides

Word processor formats

Microsoft Word docx, OpenOffice/LibreOffice ODT, OpenDocument XML, Microsoft PowerPoint.

Ebooks

EPUB version 2 or 3, FictionBook2

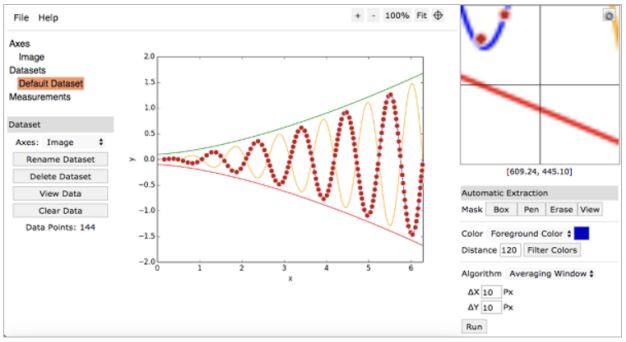
https://automeris.io/WebPlotDigitizer/ (https://automeris.io/WebPlotDigitizer/)

WebPlotDigitizer

Web based tool to extract data from plots, images, and maps

English Launch Download

Home Blog Tutorials Citation Privacy



Web Application

English

Launch Now!

Desktop Version

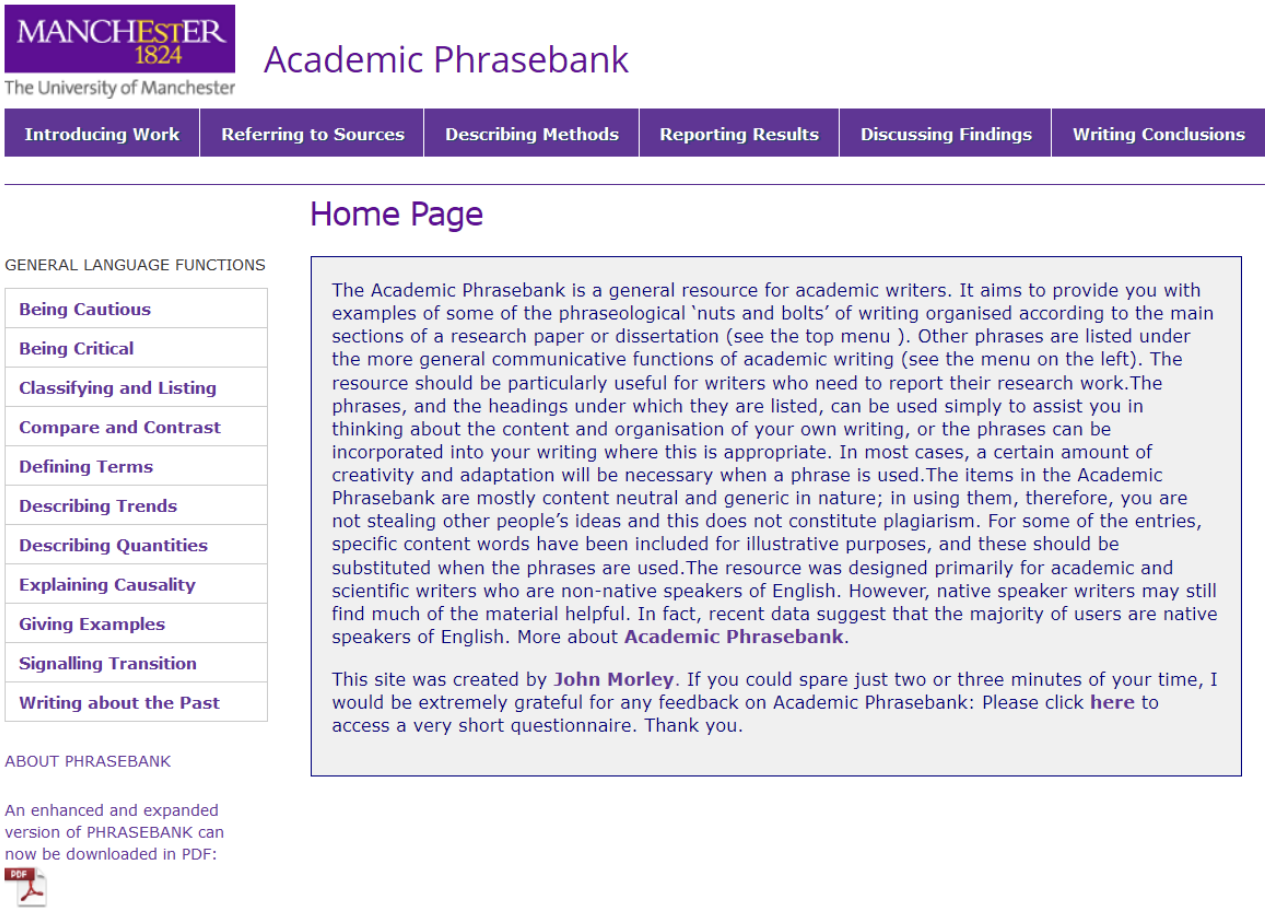
View Source

GitHub

It is often necessary to reverse engineer images of data visualizations to extract the underlying numerical data. WebPlotDigitizer is a semi-automated tool that makes this process extremely easy:

- Works with a wide variety of charts (XY, bar, polar, ternary, maps etc.)
- Automatic extraction algorithms make it easy to extract a large number of data points
- Free to use, opensource and cross-platform (web and desktop)
- Used in hundreds of published works by thousands of users
- Also useful for measuring distances or angles between various features
- More to come soon...

http://www.phrasebank.manchester.ac.uk/ (http://www.phrasebank.manchester.ac.uk/)



Programming

- programming languages
- editors
- examples

Programming languages

- Python (!)
- C (!) (performance, hardware access)
- C++ (If you need performance + OO)
- Julia (C + Python + Lisp structures)
- Kotlin (the new popular kid, mobile development)
- R (statistics)
- Scala (data science)
- Racket (for the gourmet)
- +1 Lobster lang.

> Application domains can overlap, choose 2-3 and learn them well!
>
> For instance, C/C++ and Python is a quite versatile combination.

In [ ]:

General editors

- Vim (!)
- Emacs
- Sublime Text (!)
- micro
- Atom
- Visual Studio Code
- ... wide palett of other editors ...

Specific editors

- PyCharm
- Spyder
- Juno
- Lazarus
- Code::Blocks
- **Jupyter** (!)

Why is Jupyter important?

- HTML5 platform (kernel as backend, can run remotely on a different machine)
- Compatible with several languages (C++, Python, Julia, Haskell, ...)
- Important step in hosting and sharing codes
- Towards reproducable science! (also see Jupyter Lab)

> Careful: non-linear state, see next example!

- Additional benefits of the separate backend: parallel execution, cluster management

In [3]:
```
b=2
print(b)
```

```
2
```

In [4]:
```
print(b)
b=3
```

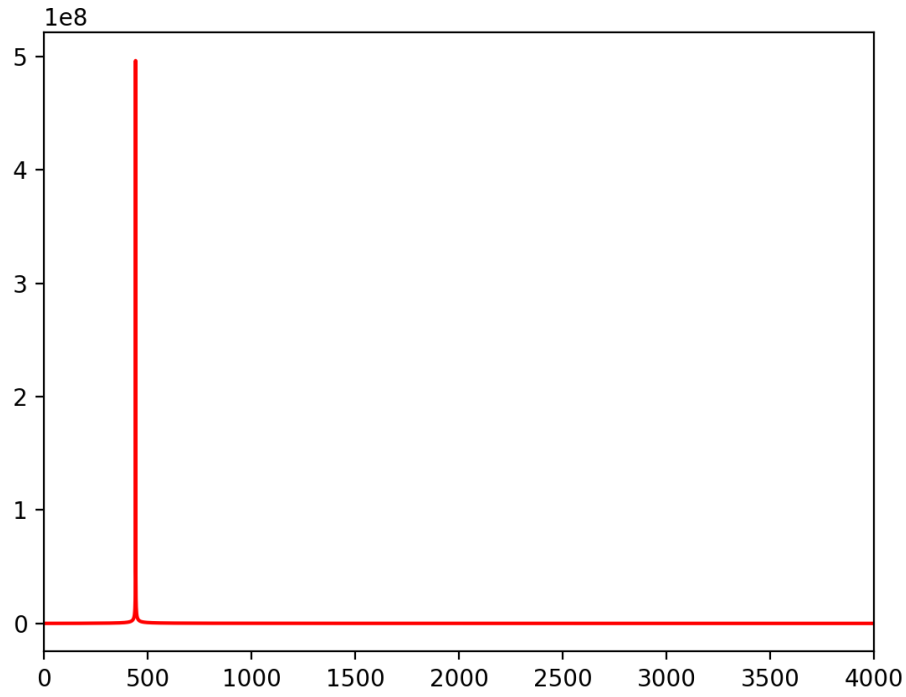
```
2
```

In [5]:
```
import sympy as sp
sp.init_printing(use_latex='mathjax')
x,y,z = sp.symbols('x,y,z')
f = sp.sin(x*y)+sp.cos(y*z)
sp.integrate(f,x)
```

Out[5]:

$$x\cos(yz) + \begin{cases} -\frac{\cos(xy)}{y} & \text{for } y \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

In [15]:
```
import matplotlib.pyplot as plt
%matplotlib notebook
from scipy.fftpack import fft; from scipy.io import wavfile
fs, data = wavfile.read('sound/sample440.wav') # load the data, 16 bit, 44.1 kH
z
c = fft(data.T) # calculate fourier transform (complex numbers list)
d = len(c)//2  # you only need half of the fft list (real signal symmetry)
plt.plot(abs(c[:(d-1)]),'r'); plt.xlim((0,4000))
```



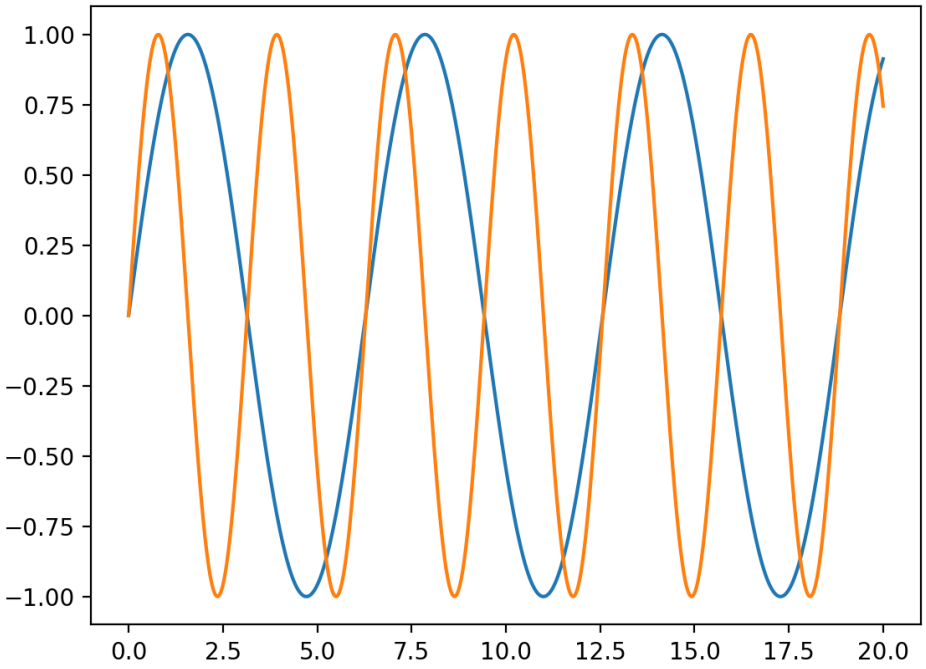
Out[15]: (0, 4000)

In [16]:
```
import IPython
IPython.display.Audio('sound/sample440.wav')
# IPython.display.Audio('sound/sample.wav')
```

Out[16]:

0:00 / 0:01

In [23]:
```
import matplotlib.pyplot as plt
%matplotlib notebook
import numpy as np
x = np.linspace(0,20,500)
plt.plot(x, np.sin(x))
```



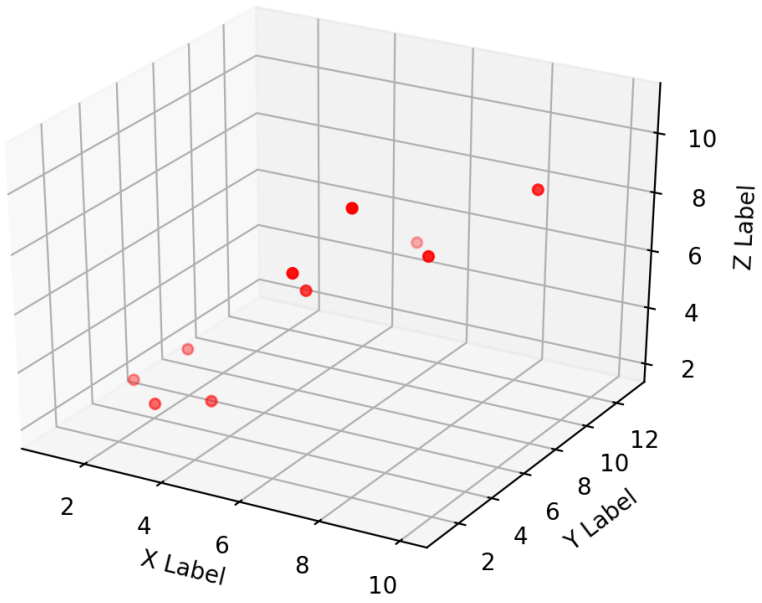
Out[23]:
```
[<matplotlib.lines.Line2D at 0x1c1fb19518>]
```

In [24]:
```
import matplotlib.pyplot as plt
%matplotlib notebook
from ipywidgets import interact, IntSlider
from IPython.display import display,clear_output

def f(freq):
    x = np.linspace(0,20,500)
    plt.plot(x, np.sin(x*freq))
    display(plt.figure)

interact(f, freq=IntSlider(min=1,max=5,step=1,value=1));
```

In [8]:
```
from IPython.display import display, Javascript
import ipywidgets as widgets

L = widgets.Label("Hello World")
display(L)
```

In [9]:
```
L.value = "Howdy World"
```

In [21]:
```
from mpl_toolkits.mplot3d import Axes3D; import matplotlib.pyplot as plt;
%matplotlib notebook

fig = plt.figure(); ax = fig.add_subplot(111, projection='3d')
x =[1,2,3,4,5,6,7,8,9,10]; y =[5,6,2,3,13,4,1,2,4,8]; z =[2,3,3,3,5,7,9,11,9,1
0]
ax.scatter(x, y, z, c='r', marker='o'); ax.set_xlabel('X Label'); ax.set_ylabel
('Y Label'); ax.set_zlabel('Z Label')
```



Out[21]: Text(0.5,0,'Z Label')

In [11]:
```
import ipyvolume as ipv
import numpy as np
import ipyvolume.datasets
stream = ipyvolume.datasets.animated_stream.fetch()
fig = ipv.figure()
# instead of doing x=stream.data[0], y=stream.data[1], ... vz=stream.data[5], u
se *stream.data
# limit to 50 timesteps to avoid having a huge notebook
q = ipv.quiver(*stream.data[:,0:50,:200], color="red", size=7)
ipv.style.use("dark") # looks better
ipv.animation_control(q, interval=200)
ipv.show()
```

In [12]:
```
import matplotlib.pyplot as plt
%matplotlib notebook
import pandas as pd
import seaborn as sns
df = sns.load_dataset("anscombe") #Anscombe's quarter, there are others (Titani
c)
df.head()
```

Out[12]:

	dataset	x	y
0	I	10.0	8.04
1	I	8.0	6.95
2	I	13.0	7.58
3	I	9.0	8.81
4	I	11.0	8.33

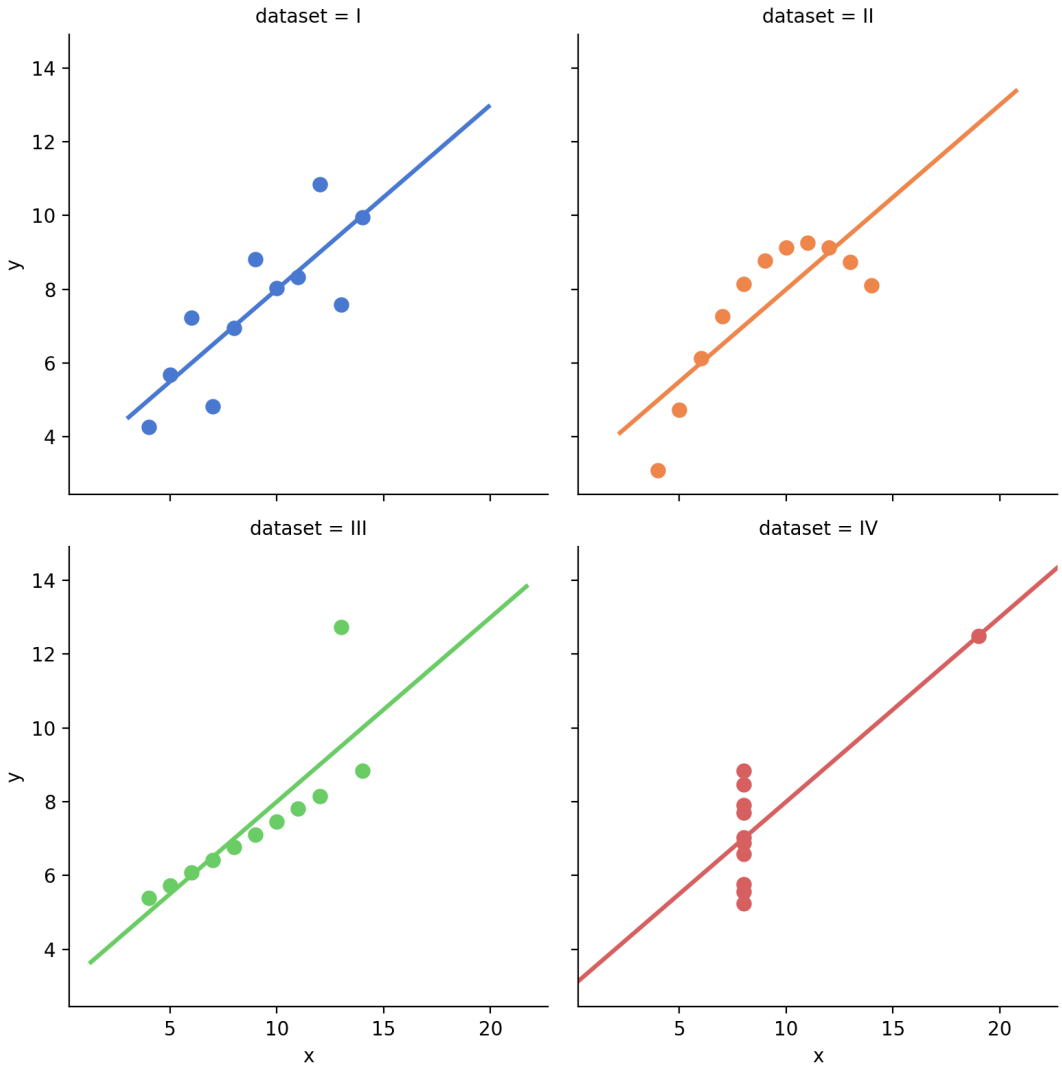
In [13]:
```
df.groupby(df.dataset).describe()
```

Out[13]:

	x								y					
	count	mean	std	min	25%	50%	75%	max	count	mean	std	min	25%	50%
dataset														
I	11.0	9.0	3.316625	4.0	6.5	9.0	11.5	14.0	11.0	7.500909	2.031568	4.26	6.315	7.58
II	11.0	9.0	3.316625	4.0	6.5	9.0	11.5	14.0	11.0	7.500909	2.031657	3.10	6.695	8.14
III	11.0	9.0	3.316625	4.0	6.5	9.0	11.5	14.0	11.0	7.500000	2.030424	5.39	6.250	7.11
IV	11.0	9.0	3.316625	8.0	8.0	8.0	8.0	19.0	11.0	7.500909	2.030579	5.25	6.170	7.04

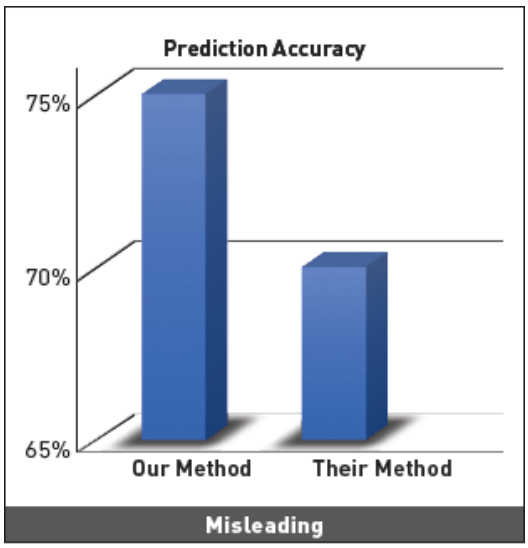
In [22]:
```
sns.lmplot(x="x", y="y", col="dataset", hue="dataset", data=df, col_wrap=2, ci=
None, palette="muted", size=4,
           scatter_kws={"s": 50, "alpha": 1}); plt.show()
```

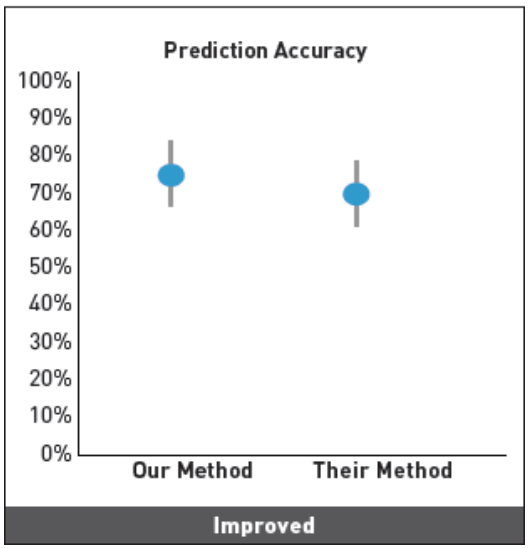
```
/anaconda3/lib/python3.7/site-packages/seaborn/regression.py:546: UserWarning:
The `size` paramter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

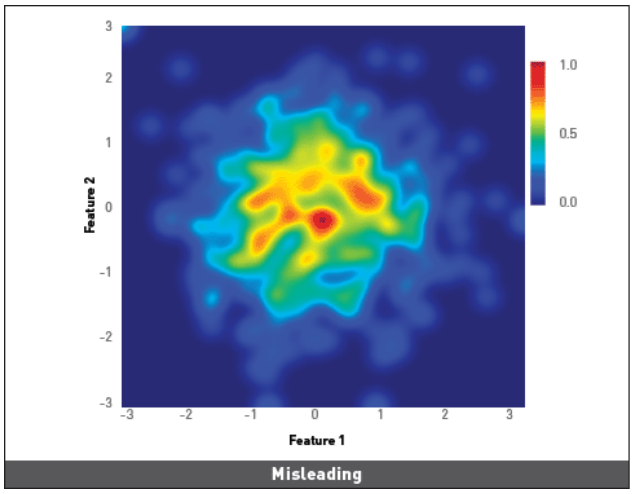


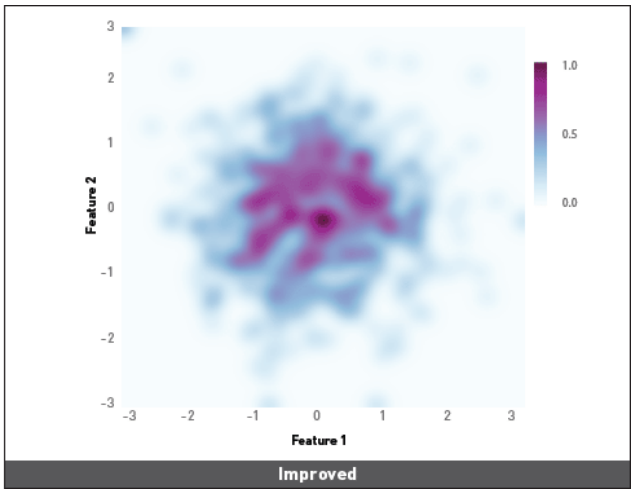
Visualization practices

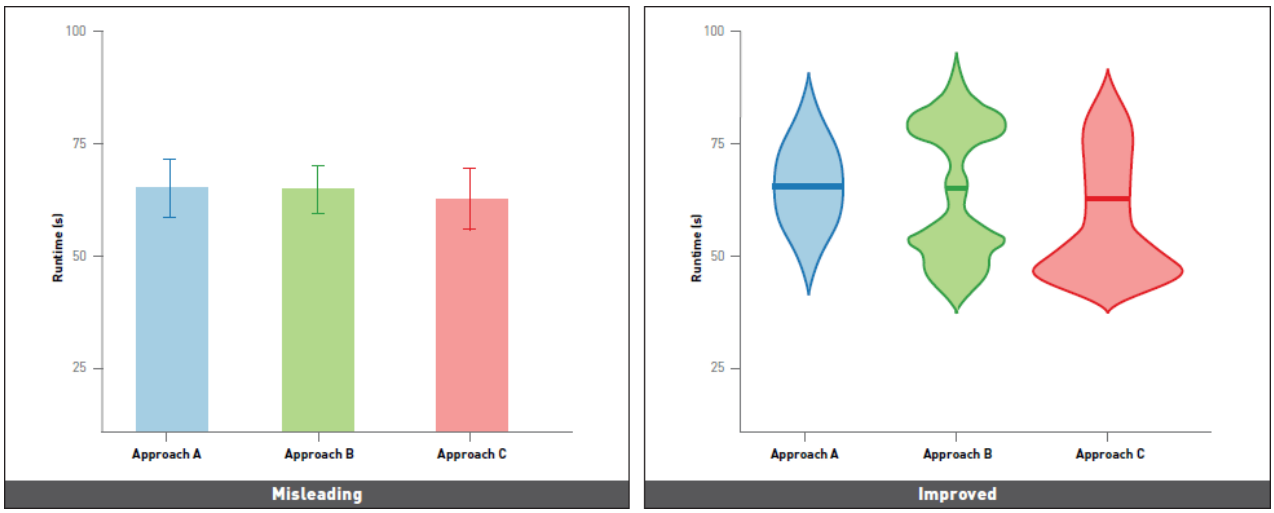
https://interactions.acm.org/archive/view/july-august-2018/the-good-the-bad-and-the-biased (https://interactions.acm.org/archive/view/july-august-2018/the-good-the-bad-and-the-biased)


Prediction Accuracy
75%
70%
65%
Our Method
Their Method
Misleading


Prediction Accuracy
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Our Method
Their Method
Improved


Feature 2
Feature 1
1.0
0.5
0.0
Misleading


Feature 2
Feature 1
1.0
0.5
0.0
Improved



Further visualization tools

Because our simulations produce a lot of data (big data?!), but the purpose of the computation is not numbers, but insight.

- python matplotlib (!) - (https://matplotlib.org/ (https://matplotlib.org/))
- ParaView (!) - VTK based parallel 3D visualization tool (https://www.paraview.org/ (https://www.paraview.org/))
- gnuplot (!) - Swiss army knif of plotters (http://www.gnuplot.info/ (http://www.gnuplot.info/))
- Processing - Programming language designed for 3D visualizations (https://processing.org/ (https://processing.org/))
- MayaVi - ParaView alternative, written in python (embeddable!) (https://docs.enthought.com/mayavi/mayavi/ (https://docs.enthought.com/mayavi/mayavi/))
- Inkscape - For that nice poster. (https://inkscape.org/ (https://inkscape.org/))

Further things to look at in the Python world

- https://github.com/barbagroup/CFDPython (https://github.com/barbagroup/CFDPython)
- http://mbakker7.github.io/exploratory_computing_with_python (http://mbakker7.github.io/exploratory_computing_with_python)
- https://github.com/vinta/awesome-python (https://github.com/vinta/awesome-python)

Other tools to mention

- Desktop jupyter: nteract (https://nteract.io/ (https://nteract.io/))
- Calculator: SpeedCrunch (https://speedcrunch.org/ (https://speedcrunch.org/))
- CAS: Maxima (http://maxima.sourceforge.net/ (http://maxima.sourceforge.net/))
- Worksheet: SMathStudio (https://en.smath.com/view/SMathStudio/summary (https://en.smath.com/view/SMathStudio/summary))
- Recording terminal session: ASCIInema (https://asciinema.org/ (https://asciinema.org/))

Software engineering principles

- Student-ware ('agile' method?)
- Guidelines PEP8
- Version control
- Tests

Software development as a student

- most often no preliminary design
- "let's see what happens" first version
- then the code is "grown" in incremental steps
- "backups" is some old version on a pendrive
- this is a natural process given the boundary conditions

Pro.:

- nothing really
- maybe time efficiency?

Con.:

- difficult collaboration
- multiple copies scattered on multiple machines

What can be improved?

Coding style guides:

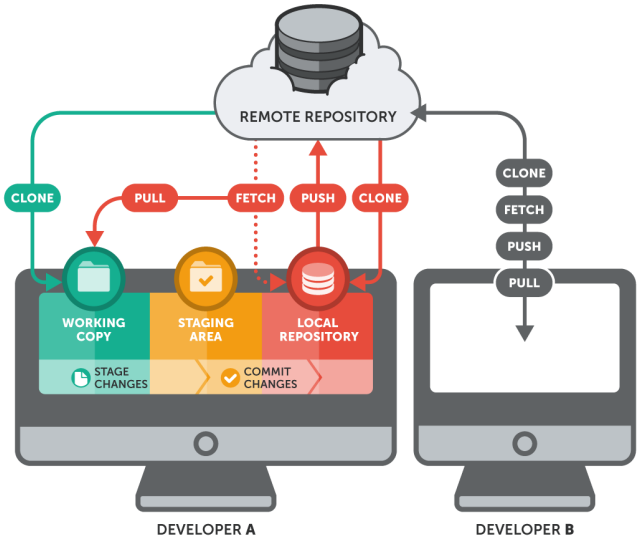
- Python PEP8 (https://www.python.org/dev/peps/ (https://www.python.org/dev/peps/))
- C++ Google Style guide (https://google.github.io/styleguide/cppguide.html (https://google.github.io/styleguide/cppguide.html))
- C - Many, e.g. Rob Pike's (https://www.maultech.com/chrislott/resources/cstyle/pikestyle.html (https://www.maultech.com/chrislott/resources/cstyle/pikestyle.html))

See the code on the next slide.

In [17]:
```
print("".join(map(lambda x: chr((lambda p, x: int(sum(map(lambda i: p[i]*x**(11
-i), range(12))) + 0.001))([__import__("struct").unpack('>d', bytes([int(s[8*i:
8*i+8], 2) for i in range(8)]))[0] for s in ["101111110101000011101010011001100
000110100110000011001000101111", "001111111011000000010111010000101110000000010
101110010010111001 1", "101111111111110101010011110000100011110000000111111001101
001000100", "010000000011100100110111000011010100011000000101010000110001 1000",
"110000000110111000001000001010011010100111101000000011101110 1101", "0100000010
010111011001010111000101110010000010000101010111011100", "110000001011011111111
011110000110101110001101110100111111111101100", "010000001100111110101101001001
110110110010110010000011100000011", "110000001101100110000101100100101001110011
001010011010100111001", "010000001101011001010101011001000011000111101111111001
1001000111", "110000001011111110000001100001100010011001011010000100011011011
0", "010000000101000111111111111111111111111101001001001010111111111001"]], x)),
range(12))))
```

```
Hello world!
```

Version control - git (!)



In [18]:
```
from IPython.display import IFrame
IFrame("doc/git_cheat_sheet.pdf", width=800, height=600) # from http://rogerdud
ler.github.io/git-guide/
```

Out[18]:

git cheat sheet

learn more about git the simple way at *rogerdudler.github.com/git-guide/*
cheat sheet created by Nina Jaeschke of *ninagrafik.com*

create & clone

create new repository	*git init*
clone local repository	*git clone /path/to/repository*
clone remote repository	*git clone username@host:/path/to/repository*

add & remove

add changes to INDEX	*git add <filename>*
add all changes to INDEX	*git add **
remove/delete	*git rm <filename>*

commit & synchronize

commit changes	*git commit -m "Commit message"*
push changes to remote repository	*git push origin master*
connect local repository to remote repository	*git remote add origin <server>*
update local repository with remote changes	*git pull*

Tools for git

- tig (https://github.com/jonas/tig (https://github.com/jonas/tig))
- sourcetree (https://www.sourcetreeapp.com/ (https://www.sourcetreeapp.com/))
- gitKraken (https://www.gitkraken.com/ (https://www.gitkraken.com/))
- most IDEs have built in support for git
- Free git service: github, bitbucket, gitlab

Testing

- unit tests
- doc tests
- CI (continuous integration)

In [19]:
```
import unittest

def fun(x):
    return x + 1

class MyTest(unittest.TestCase):
    def test(self):
        self.assertEqual(fun(3), 4)

# Testing
# Normally: unittest.main()
unittest.main(argv=['first-arg-is-ignored'], exit=False)
```

```
.
----------------------------------------------------------------------
Ran 1 test in 0.009s

OK
```

Out[19]:
```
<unittest.main.TestProgram at 0x1c1d94b278>
```

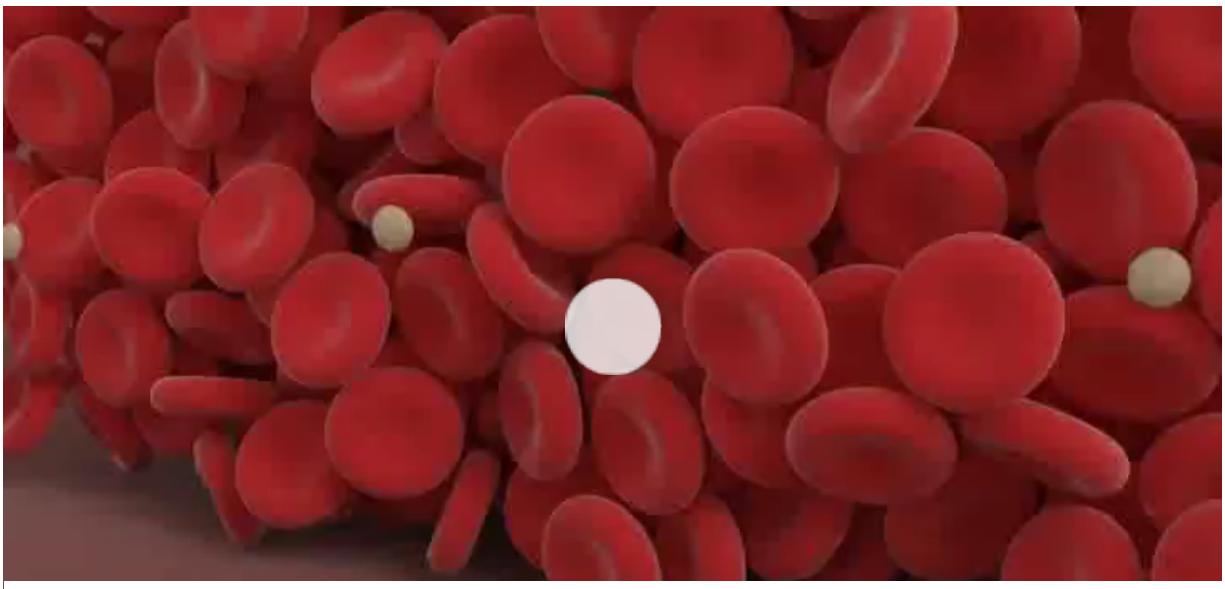
In [20]:
```
def square(x):
    """Return the square of x.

    >>> square(2)
    4
    >>> square(-2)
    4
    """

    return x * x

# Testing
import doctest
doctest.testmod()
```

Out[20]:
```
(0,  2)
```

Presentation tools

- MS PowerPoint (!)
- Reveal.js (https://github.com/hakimel/reveal.js/ (https://github.com/hakimel/reveal.js/))
- LaTex (e.g. beamer)
- Jupyter Notebook (weeell....)

Use case - HemoCell (www.hemocell.eu)



Points of interest:

- Developped in C++ with processing scripts in Python and Bash
- The source code is version controlled under git (GitLab)
- Uses 2 CI servers
- Edited mostly in Visual Studio Code, Sublime Text, and NetBeans
- The documentation is written in Markdown
- The papers are written in Overleaf
- Data visualization through ParaView
- Data evaluation through Python (+ HDF5, VTK, ...)
- Presentations, posters done in PowerPoint
- Illustrative graphics in Inkscape

Take home message

1. Write tidy, well commented code. I.e.: "Will I understand this 2 years from now?"
2. Write documentation! You'll be glad you did few years down the line.
3. Use version control, even if you are the only developer.
4. Try to include tests as much as possible.
5. Visualize whenever possible.
6. Use multiple visualization, observe it from different angles before you decide how will you present it.
7. For every storng statement you build on aim to include references (scientific ones, not wikipedia).

Homework assignement

https://github.com/gzavo/CS_Assignment (https://github.com/gzavo/CS_Assignment)

1. Create a free github account if you don't already have one.
2. Fork this repository.
3. Create a markdown (.md) named "solution_.md" file that will contain the following:
4. The title of the following papers pivotal to our knowledge:
    - MCC Van Dyke et al., 2019
    - JT Harvey, Applied Ergonomics, 2002
    - DW Ziegler et al., 2005
5. Create 1 plot from the dataset "istherecorrelation.csv", with DPI=300. The objective is to visualize the data as you see fit. Include the resulting image in the markdown file (and you can also write a few lines of interpretation if you like).
6. Commit and push these two files to your fork.
7. Create a pull request for me to this (original) repo. (Hint: use "compare across forks").

Thank you for your attention!

Questions?