Literate coding

Sampath Singamsetty

July 30, 2024

Contents

Abstract

The document provides the features of org-mode on Emacs and specifically serves as a reference on how to program w ith multiple langugaes like, emacs-lisp, python, golang, C C++ etc on org-babel. Org mode also provides first ha nd support for Bibliography and Citations through various addon packages and examples for the same are included.

1 Introduction

This guide features the way `org-babel` interacts with various languages by harnessing the rich type setting features of LaTeX and Emacs. It was originally written on 07-30-2024 using Emacs 29.4 and Org-mode 9.6.15 running on Mac OS X.

- Reproducible Research (RR)

  > An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and complete set of instructions which generated the figures.

- Literate Programming (LP)

  > Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20

1.1 elisp code

```
(defun fibonacci (n)
  (if (or (= n 0) (= n 1))
      n
    (+ (fibonacci (- n 1)) (fibonacci (- n 2)))))

(mapcar (lambda (row)
          (mapcar #'fibonacci row)) fib-inputs)
```

1.2 Python code

Here, we cover some code blocks in `python` covering basic to the advanced ones involving graphics.

1.2.1 Simple Python

```
import sys

print('Python Version: {}'.format(sys.version))
```

Pretty printing in `python`

```
from pprint import pprint as pp

data = {'powers': [x**10 for x in range(10)]}
pp(data)
```

```
import sys
import os

print('Hello ', os.getlogin(), ' system path is:')
[print(p) for p in sys.path]
```

```
print('node', 'child', 'child', 'node', sep=' -> ')
```

```
def hello(str):
    return "Hello, " + str + "!"

print(hello("Dude"))
```

```
import time

for i in range(5):
    print(i)
    time.sleep(1)
```

my-list is a variable in this org document and can be passed in as data to a source code block.

A	1
B	2
C	3

```python
print(lst)
```

The data structure that will be passed to the source block is a vector or vectors. And just like with a normal Python application you can manipulate the data.

```python
print([[chr((ord(x)+1)), y+1] for x,y in lst])
```

```python
import pandas as pd

df = pd.DataFrame({"a": [1, 2], "b": [3, 4]})
print(df)
```

```python
import random
random.seed(1)
print("Hello World! Here's a random number: %f" % random.random())
```

a	1
b	2
c	3

```python
# Return row specified by val.
# In non-session mode, use return to return results.
print(data[val])
```

The volume of an n-sphere of radius r is

$$\frac{\pi^{\frac{n}{2}}}{\Gamma\left(\frac{n}{2}+1\right)}r^{n}.$$

```python
from scipy import constants
from scipy.special import gamma

def vol(r, n):
    return constants.pi**(n/2)*r**n/gamma(n/2 + 1)

print(vol(1, 5))
```

```python
def stringcomposition(k, string):
    composition = []
    for i in range(len(string) - k + 1):
        pattern = string[i : i + k]
        composition.append(pattern)
    return composition

# Test
```

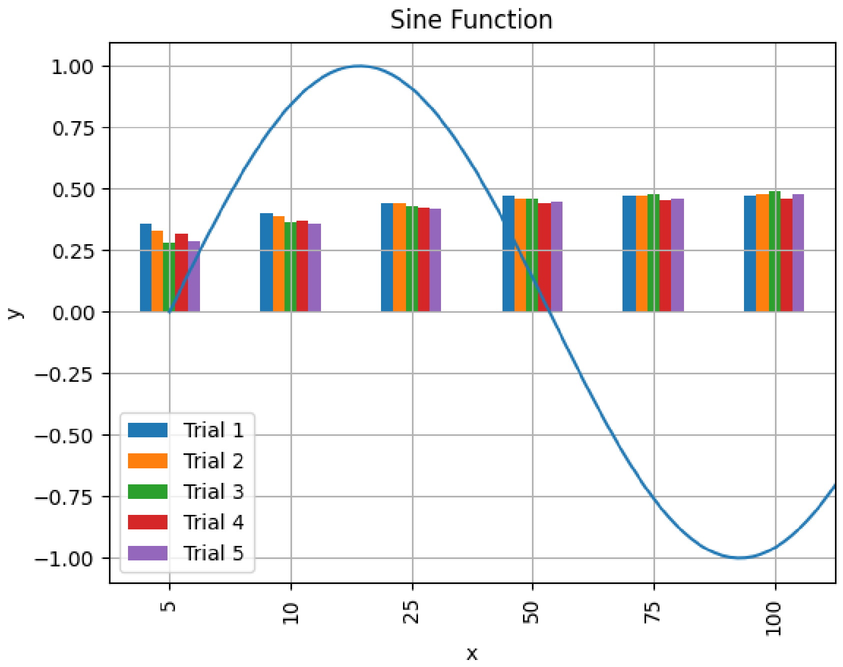
```python
k = 5
string = "CAATCCAAC"

result = stringcomposition(k, string)

# Sort results
result = sorted(result)
print(result)
```

```python
def stringcomposition(k, string):
    composition = []
    for i in range(len(string) - k + 1):
        pattern = string[i : i + k]
        composition.append(pattern)
    return composition

# Test
k = 5
string = "CAATCCAAC"

result = stringcomposition(k, string)

# Sort results
result = sorted(result)
print(result)
```

1.2.2 Python data analysis

Here some `python` code blocks predominantly involving the analysis of data and using packages like `matplotlib`, `pandas`, `numpy` etc., are covered.

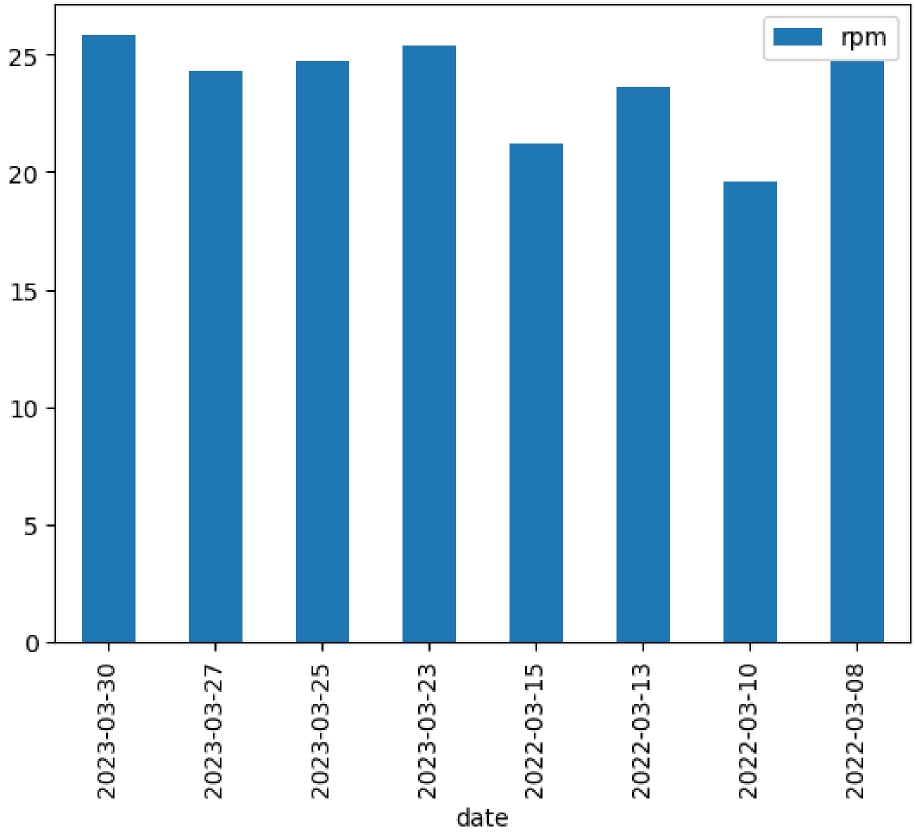
```python
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 2*np.pi, 100)
y = np.sin(x)

plt.plot(x, y)
plt.xlabel('x')
plt.ylabel('y')
plt.title('Sine Function')
plt.grid(True)

plt.savefig(image)
print(image)
```



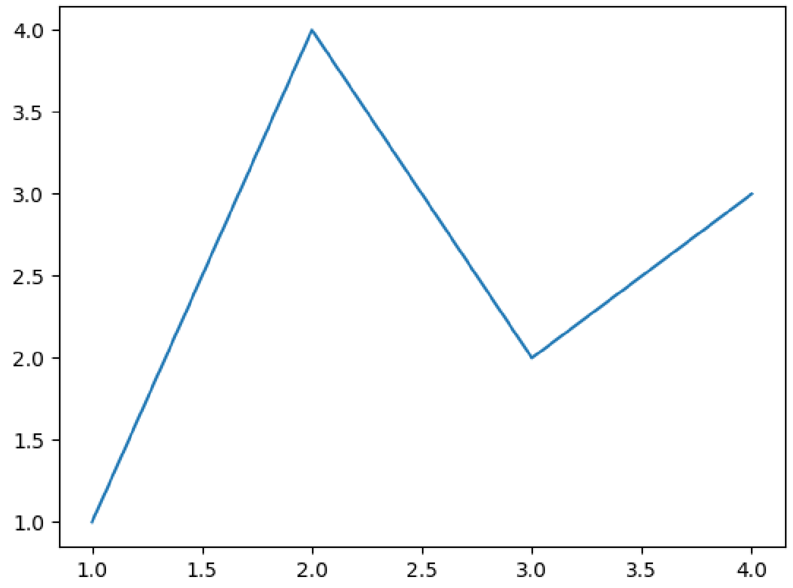
id	moving_time	date	rows	rpm
8804906424	901	2023-03-30	388	25.84
8786341302	902	2023-03-27	365	24.28
8775651293	902	2023-03-25	372	24.75
8765797455	903	2023-03-23	382	25.38
6830032281	902	2022-03-15	319	21.22
6819994746	903	2022-03-13	356	23.65
6804568223	902	2022-03-10	294	19.56
6794097174	902	2022-03-08	372	24.75

```
import pandas as pd

df = pd.DataFrame(tbl[1:], columns=tbl[0])
print(df.iloc[:2])
```

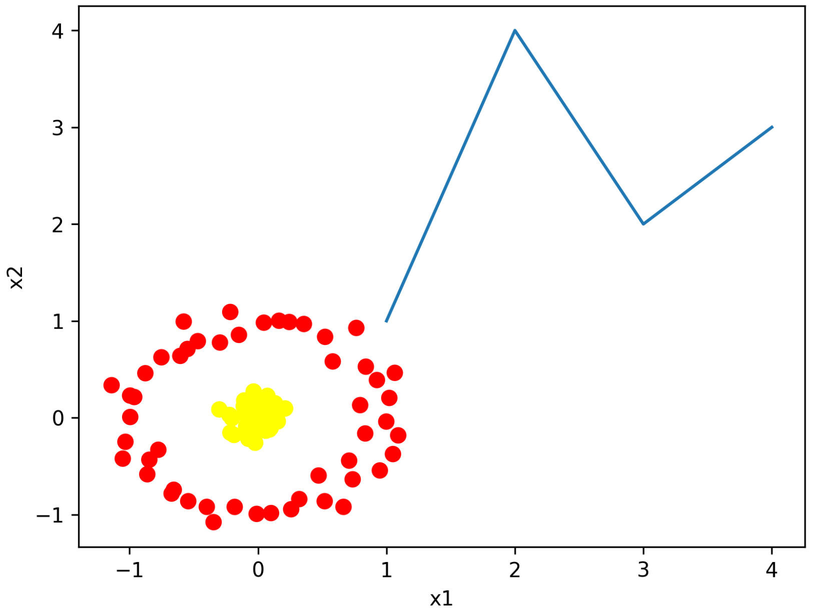
```
import pandas as pd

df = pd.DataFrame(tbl[1:], columns=tbl[0])
p = df.set_index("date").plot(y="rpm", kind="bar")
# bbox_inches="tight" cuts the image to the correct size

p.get_figure().savefig(fname, bbox_inches="tight")

print(fname)
# return filename
```



```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
plt.plot([1, 2, 3, 4], [1, 4, 2, 3])
plt.savefig('images/hello.png')
# print('[[file:images/hello.png]]')
print('images/hello.png')
pass
```



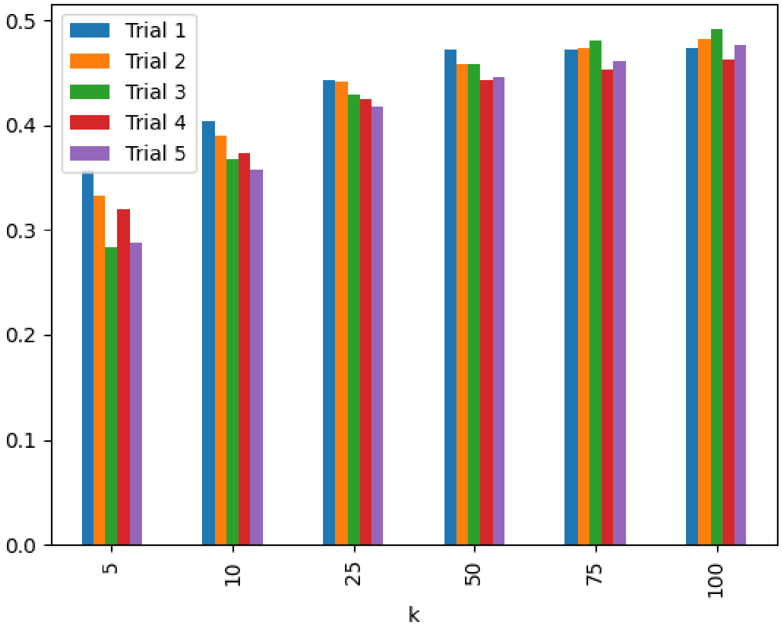
```python
import matplotlib.pyplot as plt
from sklearn.datasets import make_circles

X, y = make_circles(100, factor=.1, noise=.1)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn')
plt.xlabel('x1')
plt.ylabel('x2')
plt.savefig('images/plotCircles.png', dpi = 300)
print('images/plotCircles.png') # return filename to org-mode
```



```python
return x*x
```

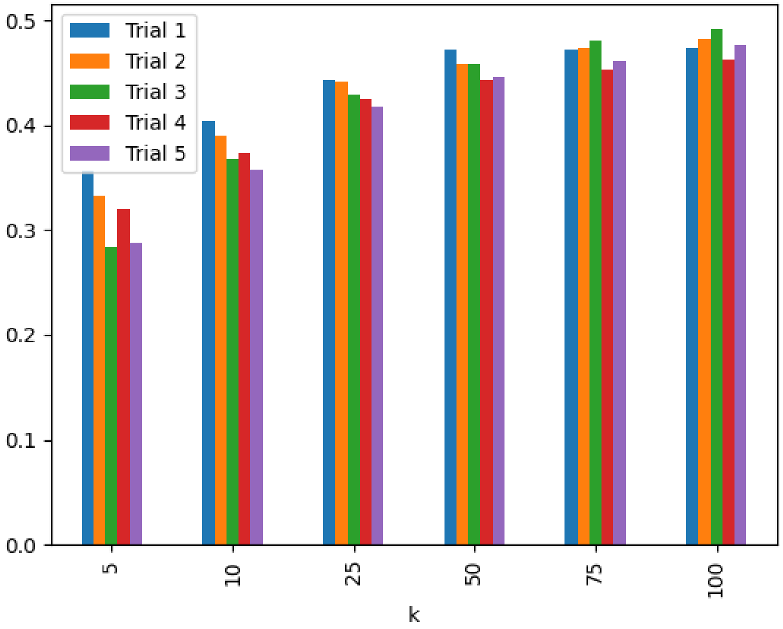
One	Two	Three	Four
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
17	18	19	20
21	22	23	24
25	26	27	28
29	30	31	32

```python
import matplotlib.pyplot as plt

'''If you have formatting lines on your table
(http://orgmode.org/manual/Column-groups.html) you need to remove them
"by hand" with a line like:

data = data[2:]
'''

'''Turn the table data into x and y data'''
x = [a[0] for a in data]
y1 = [a[1] for a in data]
y2 = [a[2] for a in data]
y3 = [a[3] for a in data]

''' Plot the x and y data'''
a, = plt.plot(x, y1, label="y1", marker='v')
b, = plt.plot(x, y2, label="y2", marker='o')
c, = plt.plot(x, y3, label="y3", marker='x')

''' Set the x and y labels on the graph '''
plt.xlabel("x axis label")
plt.ylabel("y axis label")

''' Create the legend '''
plt.legend(handles=[a,b,c],loc="upper left")

''' Save the PNG file '''
filename = "images/mySweetPlot.png"
plt.savefig(filename)

''' Return the PNG file path to OrgMode '''
print(filename)
# return(filename)
```



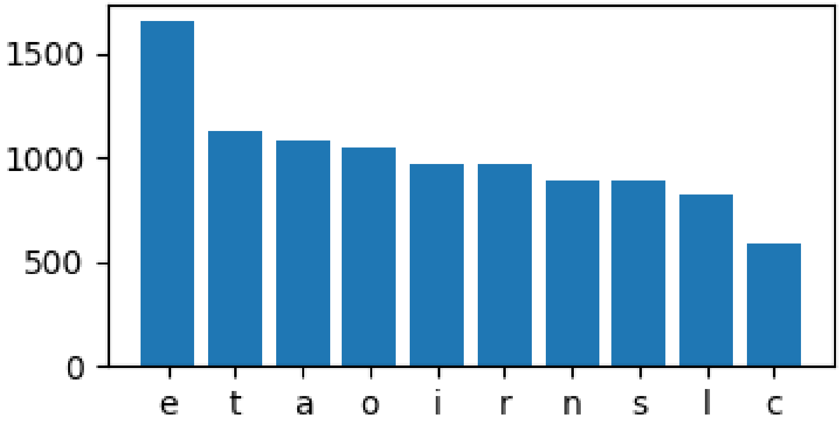
k	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
5	0.357094	0.332661	0.28434	0.320276	0.288069
10	0.403938	0.389808	0.36694	0.372952	0.357887
25	0.443313	0.441736	0.42937	0.425222	0.418354
50	0.471826	0.458904	0.45862	0.443338	0.445892
75	0.472505	0.473701	0.48072	0.452730	0.461352
100	0.473184	0.481455	0.49159	0.462386	0.476871

```python
import matplotlib.pyplot as plt
import pandas as pd

data = pd.DataFrame(data[1:], columns=data[0]).set_index('k')
data.plot(kind='bar', legend=True)

filename = "images/mySweetPlot.png"
plt.savefig(filename)

print(filename)
# return(filename)
```

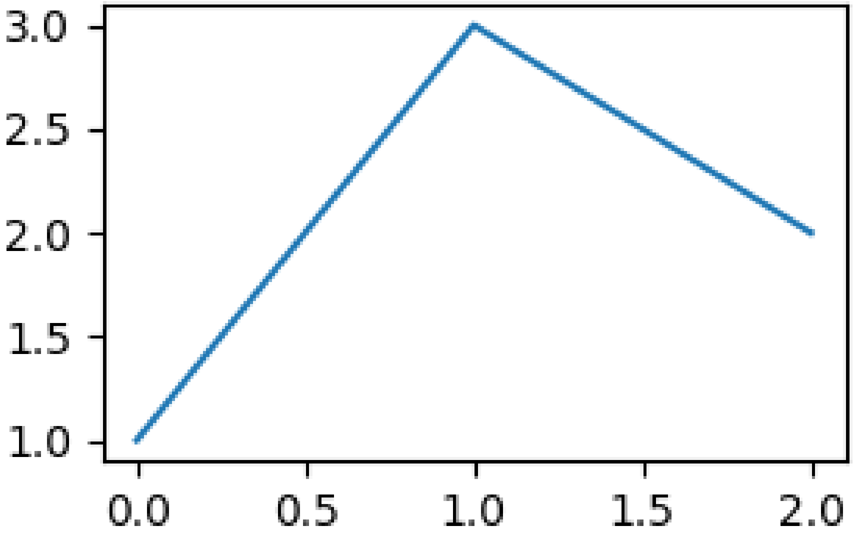


e	1648
t	1127
a	1082
o	1043
i	968
r	967
n	885
s	884
l	815
c	580

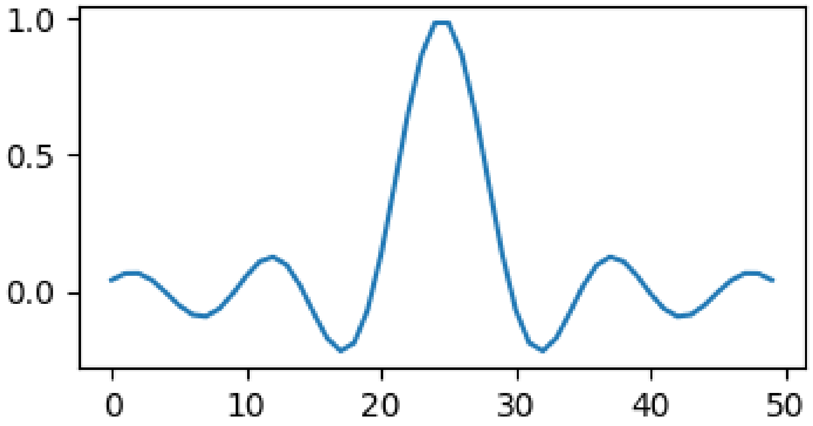
```python
import matplotlib.pyplot as plt
import pandas as pd

data = pd.DataFrame(tbl)
fig=plt.figure(figsize=(4,2))
fig.tight_layout()
plt.bar(data[0], data[1])
fgname = 'images/python-pyplot.png'
plt.savefig(fgname)
print(fgname)
```



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

fig=plt.figure(figsize=(3,2))
plt.plot([1,3,2])
fig.tight_layout()

plt.savefig(myfile)
print(myfile) # return this to org-mode
```

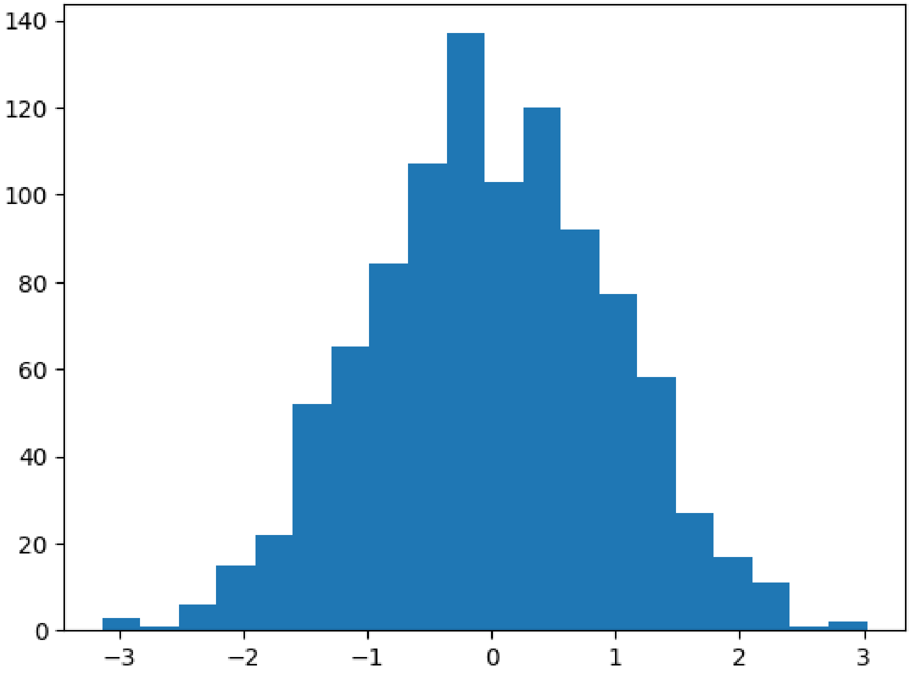


```python
import matplotlib, numpy
import matplotlib.pyplot as plt

fig=plt.figure(figsize=(4,2))
fig.tight_layout()
x=numpy.linspace(-15,15)
plt.plot(numpy.sin(x)/x)

plt.savefig(mfile)
print(mfile)
# return fname # return filename to org-mode
```



HEADER: :output-dir images

```
%matplotlib inline
%config InlineBackend.figure_format = 'png'
import matplotlib.pyplot as plt
import numpy as np

# fig, ax = plt.subplots()
# ax.plot([1, 2, 3, 4], [1, 4, 2, 3])
p = plt.hist(np.random.randn(1000), bins=20)
```

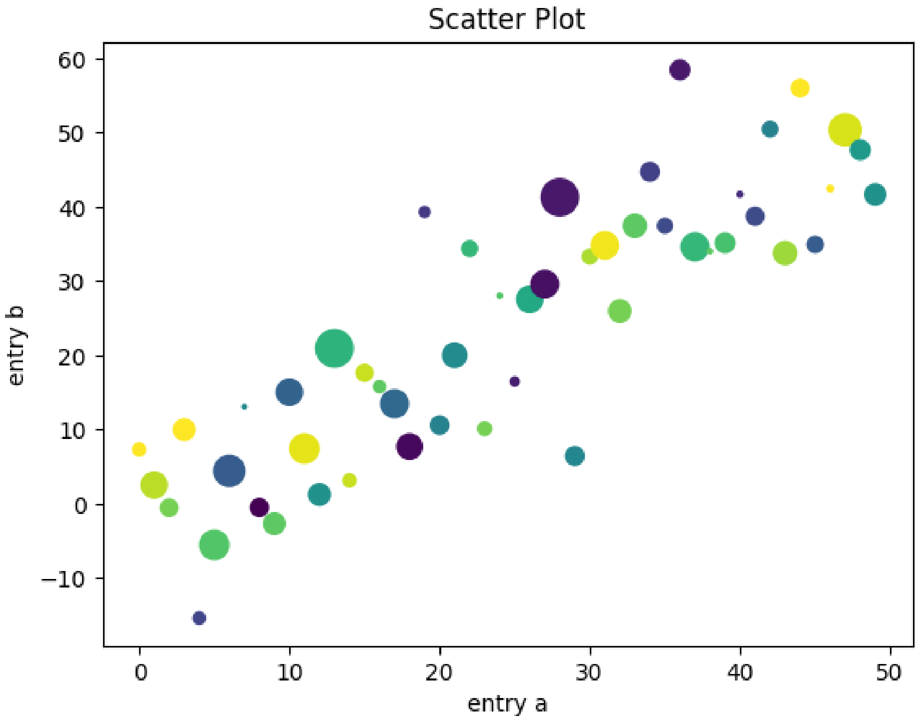


HEADER: :output-dir images

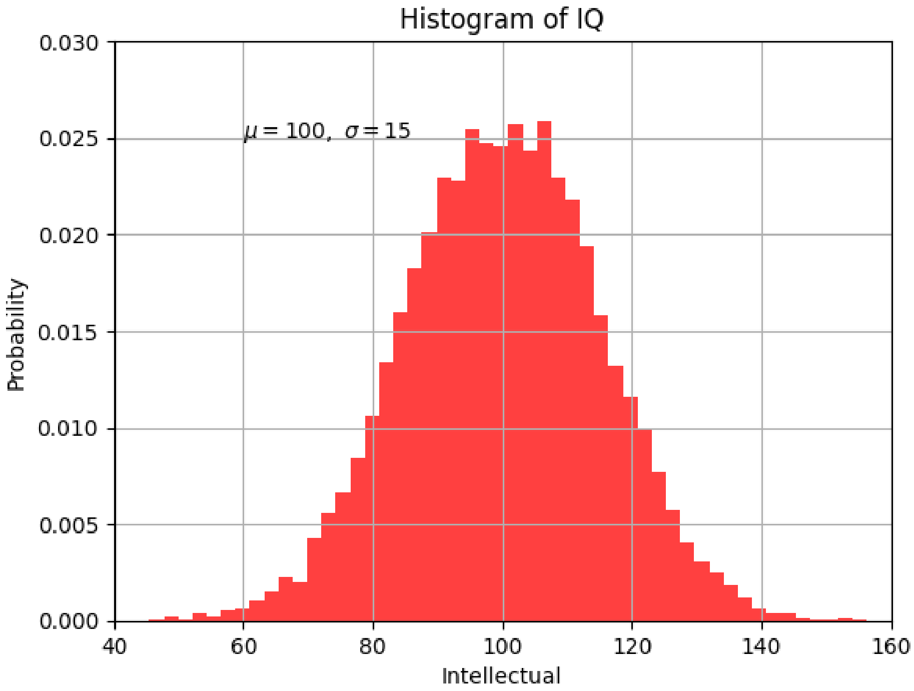
```
%matplotlib inline
%config InlineBackend.figure_format = 'png'
import matplotlib.pyplot as plt
```

```python
import numpy as np

data = {
    'a': np.arange(50),
    'c': np.random.randint(0, 50, 50),
    'd': np.random.randn(50)
}
data['b'] = data['a'] + 10*np.random.randn(50)
data['d'] = np.abs(data['d'])*100

plt.scatter('a', 'b', c='c', s='d', data=data)
plt.title('Scatter Plot')
plt.xlabel('entry a')
plt.ylabel('entry b')
```

```
Text(0, 0.5, 'entry b')
```

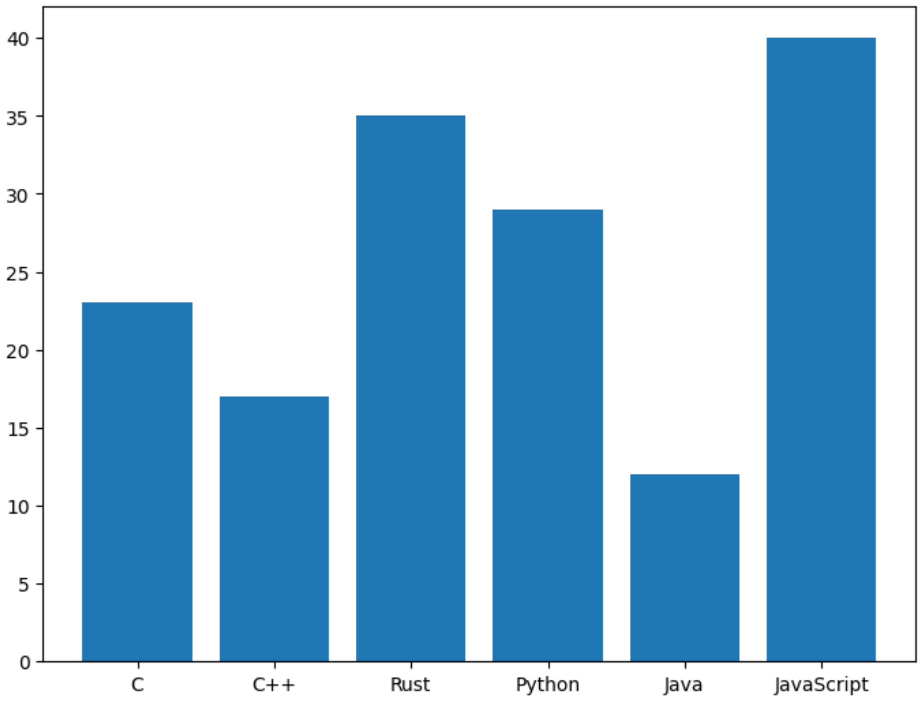


```python
%matplotlib inline
%config InlineBackend.figure_format = 'png'
import matplotlib.pyplot as plt
import numpy as np

mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)

n, bins, patches = plt.hist(x, 50, density=1, facecolor='r', alpha=0.75)

plt.xlabel('Intellectual')
plt.ylabel('Probability')
plt.title('Histogram of IQ')
plt.text(60, 0.025, r'$\mu=100,\ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
```



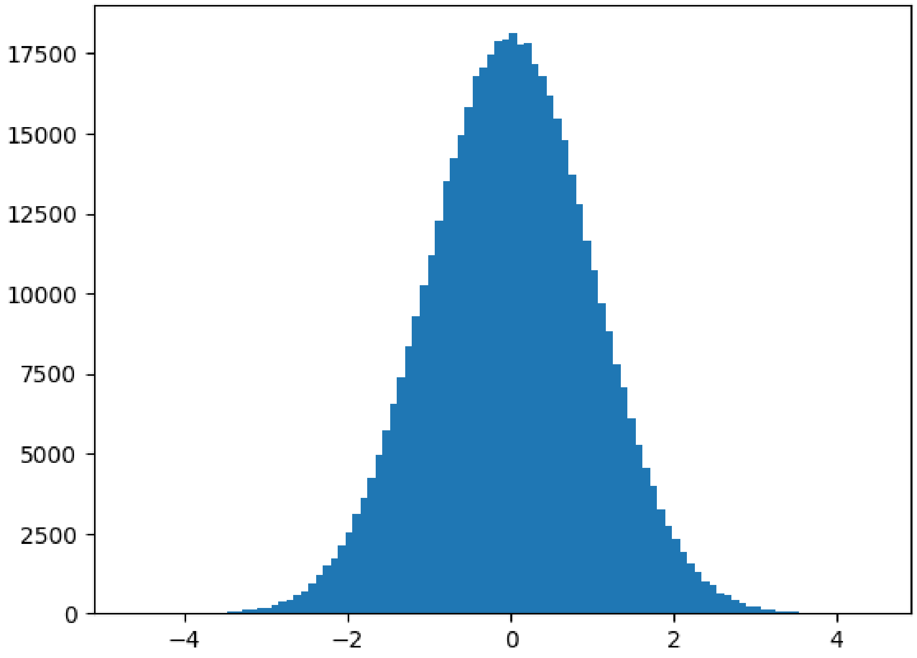
```
%matplotlib inline
%config InlineBackend.figure_format = 'png'
import matplotlib.pyplot as plt

fig = plt.figure()
ax = fig.add_axes([0, 0, 1, 1])
langs = ['C', 'C++', 'Rust', 'Python', 'Java', 'JavaScript']
students = [23, 17, 35, 29, 12, 40]

ax.bar(langs, students)
```

```
<BarContainer object of 6 artists>
```



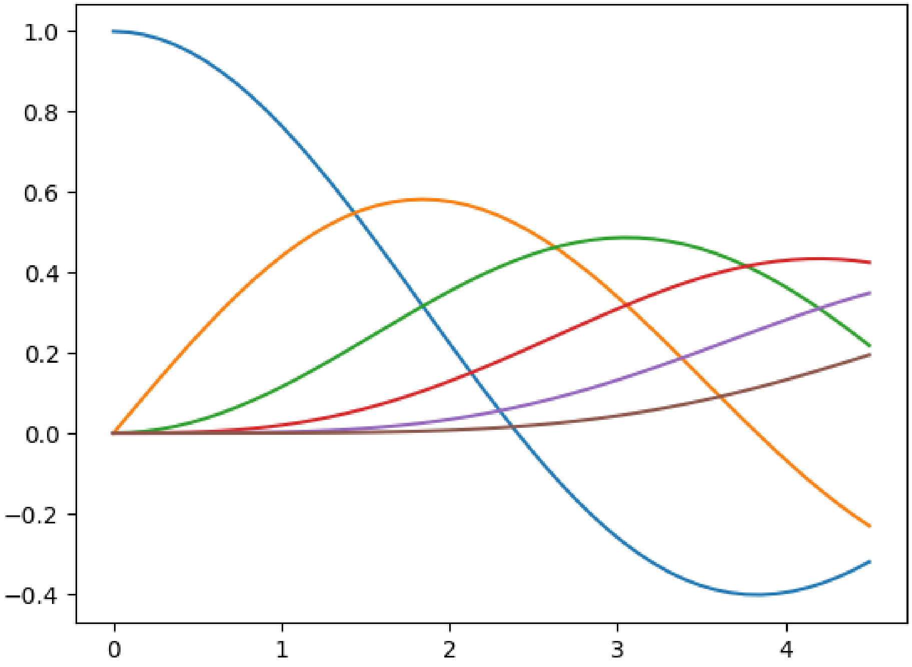
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

```
fig=plt.figure(facecolor='white')
plt.hist(np.random.randn(500000), bins=100);
```



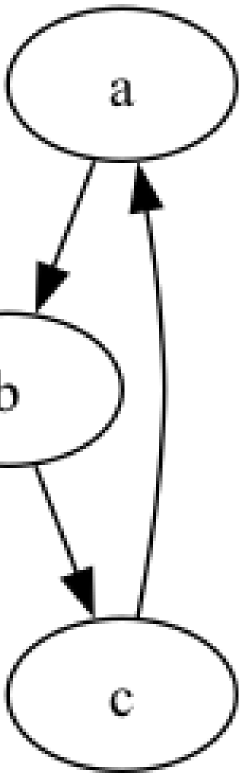
HEADER: :output-dir images

```
%matplotlib inline
%config InlineBackend.figure_format = 'png'
import matplotlib.pyplot as plt
import numpy as np
from scipy.special import jn

x = np.linspace(0, 4.5)
for i in range(6):
    plt.plot(x, jn(i, x))
```



2 Post processing

```
echo "#+ATTR_LATEX: :width $width"
echo "$data"
```


a
b
c