python org-babel exporting

Derek Feichtinger

<2013-07-10 Wed>

Contents

1 Version information **1**

2 Links and Documentation **2**

3 Generating tables as output **2**

4 Calling a python function in an org table **3**

5 Matplotlib **3**
5.1 plotting of a simple graph 3
5.2 Plotting from an Org table 4

6 Pandas **5**
6.1 printing a data frame as a table 5
6.1.1 an older and simpler dataFrame printing alternative: . 7
6.2 plotting a data frame (and placing a code reference) 8
6.3 time series resampling 9

7 Unicode related problems in Org Babel **10**

1 Version information

```
(princ (concat
        (format "Emacs version: %s\n"
                (emacs-version))
        (format "org version: %s\n"
                (org-version))))
```

```
Emacs version: GNU Emacs 24.3.1 (i686-pc-linux-gnu, GTK+ Version 3.4.2)
 of 2013-10-03 on hamsa, modified by Debian
org version: 8.2.3b
```

2 Links and Documentation

- `http://orgmode.org/worg/org-contrib/babel/languages/ob-doc-python.html`

3 Generating tables as output

Example 1:

```
x = range(1,10)
y = [xe*3 for xe in x]
return [x,y]
```

1	2	3	4	5	6	7	8	9
3	6	9	12	15	18	21	24	27

Example 2:

```
import numpy as np

x = range(1,10)
y = [xe*3 for xe in x]
return np.array([x,y]).transpose()
```

1	3
2	6
3	9
4	12
5	15
6	18
7	21
8	24
9	27

4 Calling a python function in an org table

Here I define the function. It takes `epoch` as the variable, which is a unix time stamp. I want to have it converted to an Org type time format.

```
time = epoch
import datetime
strtime = str(time)
datetimestamp = datetime.datetime.utcfromtimestamp(int(strtime[:10]))
print datetimestamp.strftime('[%Y-%m-%d %a %H:%M:%S]')
```

```
[2010-01-05 Tue 07:11:05]
```

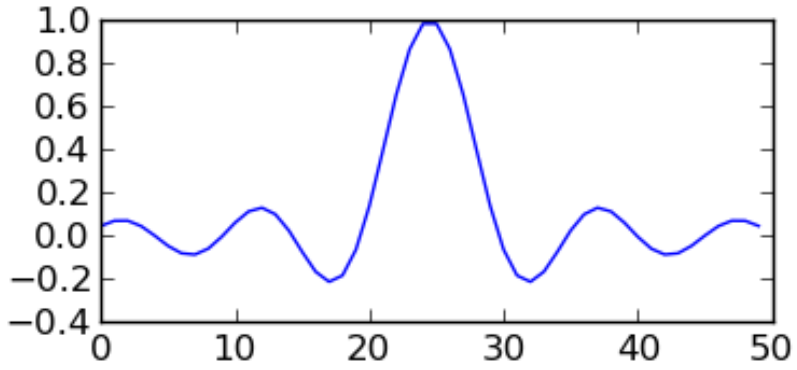
In the table we need to refer to the named source block by using the a short lisp form involving `org-sbe`. If the table value that is referred to in the function is to be interpreted as a number, the reference uses a single dollar sign, etc $1 (as here). If it should be interpreted as a string, one puts an additional dollar sign in front, e.g. $$1.

epoch	day
1262675465119	*[2010-01-05 Tue 07:11:05]*
123456	*[1970-01-02 Fri 10:17:36]*
99998754	*[1973-03-03 Sat 09:25:54]*

5 Matplotlib

5.1 plotting of a simple graph

```
import matplotlib, numpy
matplotlib.use('Agg')
import matplotlib.pyplot as plt
fig=plt.figure(figsize=(4,2))
x=numpy.linspace(-15,15)
plt.plot(numpy.sin(x)/x)
fig.tight_layout()
plt.savefig('python-matplot-fig.png')
return 'python-matplot-fig.png' # return filename to org-mode
```



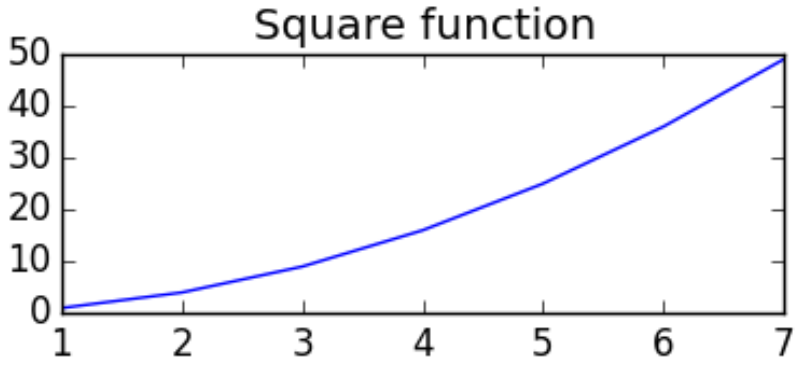
5.2 Plotting from an Org table

The table is passed to python as a list

x	y
1	1
2	4
3	9
4	16
5	25
6	36
7	49

```
import matplotlib
import numpy as np
matplotlib.use('Agg')
import matplotlib.pyplot as plt

fname='python-matplot-fig2.png'
ar = np.array(data).transpose()
fig=plt.figure(figsize=(4,2))
plt.plot(ar[0],ar[1])
plt.title('Square function')
fig.tight_layout()
plt.savefig(fname)
return fname # return filename to org-mode
```



6 Pandas

6.1 printing a data frame as a table

I define a function in a named src block with name `dframeToOrg`. This will print out a nice table format that org will recognize. The function currently assumes that the first line is the title line, and will put a horizontal line below it.

```
def dataFrameToOrgTbl(dframe, name=None, caption=None, attr=None, index=True):
    if name:
        print "#+NAME: %s" % name

    if caption:
        print "#+CAPTION: %s" % caption

    if attr:
        print "#+ATTR_LATEX: %s" % attr


    lines = '|' + dframe.to_csv(None, sep='|', line_terminator='|\n|', encoding='utf-8

    for i,l in enumerate(lines.split('\n')):
        if i == 1:
            print "|-----"
        print l
```

In the following source block, I demonstrate how to use the `noweb` syntax of including a named block within another, by referring to our DataFrame printing block by `«dframeToOrg»`

```
import pandas as pd
import numpy as np

# Here the block will be inserted
def dataFrameToOrgTbl(dframe, name=None, caption=None, attr=None, index=True):
    if name:
        print "#+NAME: %s" % name

    if caption:
        print "#+CAPTION: %s" % caption

    if attr:
        print "#+ATTR_LATEX: %s" % attr


    lines = '|' + dframe.to_csv(None, sep='|', line_terminator='|\n|', encoding='utf-8

    for i,l in enumerate(lines.split('\n')):
        if i == 1:
            print "|-----"
        print l
df = pd.DataFrame({'A' : ['one', 'one', 'two', 'three'] * 3,
                   'B' : ['A', 'B', 'C'] * 4,
                   'C' : ['foo', 'foo', 'foo', 'bar', 'bar', 'bar'] * 2,
                   'D' : np.random.randn(12),
                   'E' : np.random.randn(12)})

dataFrameToOrgTbl(df)
```

	A	B	C	D	E
0	one	A	foo	-0.236396241307	-0.276568067645
1	one	B	foo	-1.01951010991	1.7453786746
2	two	C	foo	-1.29371941308	1.09264039165
3	three	A	bar	-1.04427788416	0.513056847005
4	one	B	bar	0.572505261205	-0.115515573013
5	one	C	bar	-0.198276698791	0.303982746716
6	two	A	foo	-2.50621425568	0.332395607933
7	three	B	foo	0.201709578183	-0.856279014171
8	one	C	foo	-2.03374293998	-0.622682697996
9	one	A	bar	-0.268297986255	0.00646257542859
10	two	B	bar	1.19869717505	0.48755007568
11	three	C	bar	-0.0958952713139	1.04738590497

The noweb syntax is mostly used in literate programing, where we produce code files from the org file (the process is called *tangling*).

6.1.1 an older and simpler dataFrame printing alternative:

In order to get a nice org table, it is necessary to pass the frame's contents back as a list. The column names end up as the first row in the table. I cut this row away by using the [1:] slice.

```
import pandas as pd
import numpy as np
import sys

df = pd.DataFrame({'A' : ['one', 'one', 'two', 'three'] * 3,
                   'B' : ['A', 'B', 'C'] * 4,
                   'C' : ['foo', 'foo', 'foo', 'bar', 'bar', 'bar'] * 2,
                   'D' : np.random.randn(12),
                   'E' : np.random.randn(12)})

return(np.array(list(df.T.itertuples())).transpose()[1:])
#df.to_csv(sys.stdout, sep='|',line_terminator='|\n')
#return (df.to_string(col_space=5, justify='right',index=False))

# this is a good one
#print '|',(df.to_csv(None, sep='|', line_terminator='|\n|', encoding='utf-8'))
```

6.2 plotting a data frame (and placing a code reference)

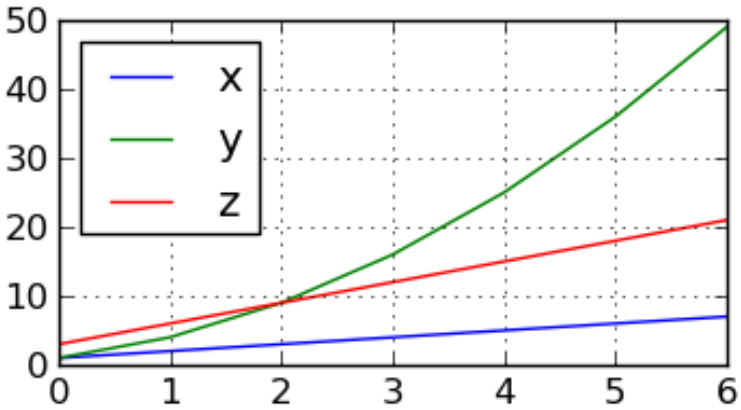
x	y
1	1
2	4
3	9
4	16
5	25
6	36
7	49

Here we also show how a code reference works. It can be inserted using the **org-store-link** command while editing the src code in the dedicated buffer:

In line 10 we define a new column (in this sentence you should see the number of the respective line in the exported file)

The **-r** flag in the `BEGIN_SRC` line removes the reference string from the source code listing in the output (else the string would have ended up in the exported version's source code). Regrettably the reference is not removed when the code gets executed, so I need to insert language specific commenting to keep the code functional.

```
1  import matplotlib
2  import matplotlib.pyplot as plt
3  import pandas as pd
4  import numpy as np
5  matplotlib.use('Agg')
6
7  fname='python-matplot-fig3.png'
8  df = pd.DataFrame(data)
9  df.columns = ['x','y']
10 df['z'] = df['x'] * 3                                          #
11
12 df.plot(figsize=(4,2))
13 plt.savefig(fname)
14 return fname
```



6.3 time series resampling

Let's say we are taking measurements twice a day, every 12h.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

ts = pd.date_range('2013-07-01 06:00:00', periods=20, freq='12h')
val = [x * 10.0 for x in range(len(ts))]

tdf = pd.DataFrame({'value': val}, index=ts)
# Now we put one observation as invalid
tdf.value[14] = np.NaN
# and we delete another one
#tdf = tdf.drop(tdf.index[2])
tdf = tdf.drop(tdf.index[6:8])

newdf = tdf.resample('1D', loffset='6h',how='min').rename(columns={'value': '1D_resamp
newdf['diff'] = newdf.diff()

return pd.concat([tdf,newdf], join='inner',axis=1)
```

```
value  1D_resample  diff
2013-07-01 06:00:00      0            0   NaN
2013-07-02 06:00:00     20           20    20
2013-07-03 06:00:00     40           40    20
2013-07-05 06:00:00     80           80   NaN
```

```
2013-07-06 06:00:00    100          100    20
2013-07-07 06:00:00    120          120    20
2013-07-08 06:00:00    NaN          150    30
2013-07-09 06:00:00    160          160    10
2013-07-10 06:00:00    180          180    20
```

7 Unicode related problems in Org Babel

The output terminal to which org babel writes output seems to be a dumb ASCII type of terminal. If one wants to print non-ASCII characters, the characteristics of the output device must be defined using the `codecs` module.

```
# -*- coding: iso-8859-15 -*-

# the above line is needed, so that python accepts the Umlauts
# in the following line
strg = u'Can we see Umlauts? . And accents? '

import sys

try:
    print strg
except:
    print "Expected error:", sys.exc_info()[0]

import codecs
sys.stdout = codecs.getwriter('utf8')(sys.stdout)

print "\nNow it works:\n", strg
```

```
Expected error: <type 'exceptions.UnicodeEncodeError'>

Now it works:
Can we see Umlauts? äöü. And accents? éè.
```