text

Steven Armour

July 30, 2015

Abstract

will do later

Note this is still a work in progress. As such I have not done any bibtex yet and therefore I use random key banging (Ex: alkdjfldjfk) as a place holder of information I need to come back and add to.

1 Introduction

Despite our societies hopeful move to a paperless society we are still bound to the need for formal reports. These reports are often time consuming, tedious and therefore subject to questions of accuracy of information presented therein; however, a formal report is often a legal binding document. The inaccuracy of reports can be further compounded when required to cite information from previously inaccurate reports or a person having to having to fill out the same report with only slight difference in say a few items in the report. These discrepancies can then be attributed to human error. Presented here is a guide to means of eliminating human error in only the creation of report by combining the power of LaTeX to create perfectly typeset documents via text markup code and integration of *Python* to handle retrieval and or processing of information for creating reports by means of *Pweave* .

The scope of coverage of this guide to *Pweave* it to that *Pweave* . While there will be use of various aspects of LaTeX and usage of various *Python* libraries packages the author assumes that the reader is already familiar with the basics of both. If the reader is not familiar with LaTeX or *Python* the following references should provide suplimentry materail learning both: adlakldsjfdjfkj

Discussion

Pweave was developed by adlkakfjkdj as a response from the *Python* community to the *S/R* community's *Sweave* integration of programing code and LaTeX markdown code in a single document to create reproducible literature. For a history and guide to *Sweave* please see the following references

lksdjfkajdrfj. While *Pweave* is a milestone in the *Python* community for "porting" *Sweave* to *Python* ; it however is not alone the integration of *Python* and LaTeX . For instants the *Ipython Notebook* (now under project *Jupyter*) can by way of adjkajdfkj create LaTeX from a *Ipython Notebook*. Further, there are also ways of producing *PDF* reports from languages other than *Python* . The interested reader can look at the following resources for exsamples: ajdfakdjfkajdkfjakd.

The niche that *Pweave* plays in the ecosystem of *Python* and LaTeX integration is in allowing the writer\programmer separation and recombination the LaTeX & *Python* sections of a document. This is the classical "Divide and Conquer" approach that is already well practiced in programming via th separations of backend: database, calculations, *ex* ;& Front End: Gr ph ical User Interface, graphs, Web page views, *ex*. For use of *Pwe* this means the task of drafting the content and creation of the docum nt la t with LaTeX is one action; while developing the *Python* code t pe rm the tobe automated action is another. These two semi-separate ac ar en "weaved" together with *Pweave* to create the final pro Wh thi is *Pweav*'s strength it is also a weakness due to having syn he elop-ment of the LaTeX and *Python* much like any fro nd Bac end develop-ment; therefore, best software engineering mana ent practices should be adopted by any team when creating *Pw ave* cum ts

3 *Pweave* Setup

To use *Pweave* the following mu installed:

- *Python 2* or *Pyth 3*
- LaTeX

The followi a highl recommended in the use of *Python* :

- *pip*
- An *Integr d Development Environment (IDE)* for *Python*

An the owing are highly recommended in the use of LaTeX :

- An *Integrated Writing Environment* for LaTeX

Once the requirements listed in table dsfajdfkj are satisfied *Pweave* can be tup on your machine as follows:

1. Open a termanal (*CMD* in Microsoft Winows) window
2. Enter the following into the windows followed by hitting enter

```
$ sudo pip install --upgrade Pweave
```

If you are using Microft windows or a non Debian system ignore *sudo*

3. After the instulation has completed varfy it was successfull by entering in the same window:

```
Pweave --help
```

where if the instalantion was succesful a listing of *Pweave* command will be shown. From here the window can be closed.

4 *Pweave* How To

4.1 Workflow

4.2 From *Pweave* to *PDF*

4.3 Code Chunk Controls

4.4 Common Examples

4.4.1 Tables

4.4.1.1 *Pandas*

The *Pandas* library extends th uctured arrays found in *NumPy* into more *SQL* database tabl lik rrays. These special arrays in *Pandas* are called "*dataframes*" a st lik *NumPy* arrays or *SQL* tables "Quarries" can be called against them extract or perform calculation on subsets of the original dataframe based certain criteria. Here is shown a primitive example of in *Pand* with *Pweave* .

For thi exs le a *CSV* file acured from alkdfjkajf listing some basic information out p USA superbowls.In order to use *Pandas* in LaTeX the LaTeX pac e "booktabs" will need to be added to the preample of the do um t St rt by importing *Pandas* into the *Python* environment as follows

```
import pandas as pd
```

Next the path to the .csv containing the Superbowl information is spec- ied followed by loading the data into a dataframe named `df`.

```
#path to .csv file
path=r'/home/iridium/string12oil@gmail.com/PWeaveEx/ExDoc/Untitled
Folder/SuperbowlFacts.csv'
```

```
#load .csv to dataframe df
df=pd.read_csv(path)
print(r'\vspace{.5cm}')
```

\vspace{.5cm}

The first three rows in `df` are specified by using the `.head(3)` and are printed to a LaTeX table by means of adding `.to_latex()` after the head method. The complete code is as shown

```
#<<echo=False, wrap=True, results='tex'>>=
print(df.head(3).to_latex())
print(r'\vspace{.5cm}')
#@
```

	Date	SB	Winner	Pts	Loser	Pts.1	MV
0	Feb 1 2015	XLIX (49)	New England Patriots	28	Seattle Seahawks	24	Tor
1	Feb 2 2014	XLVIII (48)	Seattle Seahawks	43	Denver Broncos	8	Mal
2	Feb 3 2013	XLVII (47)	Baltimore Ravens	34	San Francisco 49ers	31	Joe

Notice in the code chunk we have set `wrap=True` and `results='tex'` so that print out is in LaTeX format.

To quickly show how the data analysis abilities of *Pandas* we will use `.describe()` method to call find some simple statics on the *Pts.* and *Pts.1* columns

```
#<<echo=False, wrap=True, results='tex'>>=
print()
print(df.describe().to_latex())
print(r'\vspace{.5cm}')
#@
```

	Pts	Pts.1
count	49.000000	49.000000
mean	30.265306	15.938776
std	9.778325	6.862969
min	14.000000	3.000000
25%	23.000000	10.000000
50%	30.000000	17.000000
75%	35.000000	21.000000
max	55.000000	31.000000

As can be seen that *Pandas* dataframe `.to_latex()` method can be fully leveraged by means *Pweave* . For the interested reader the full `.to_latex()` method document can be found at ldkfjakdjfkj and more inforamtion about *Pandas* can be found at alsdkfjkadsjfk.

4.4.2 Plotting -*Matplotlib*

4.4.3 Mathematics -*Sympy*

SymPy is *Python*'s Computer Algebra System library. It provides a means of defining symbolic symbols to build mathematical functions in the traditional means of "paper and pencil" via the computer. This aids in the derivation process of formulas by leveraging the power of the computer to keep track of very large expression without miss transcribing terms between steps in the derivation. It has also been shown that with use of symbolic algebra systems to reduce the governing equations to their most reduce form before proceeding with numerical solving increase the efficiency of the numerical solution as opposed to using a non reduced formula. Further *SymPy* has built in plotting models to produce *matplotlib* plots from the expression and the `lambdify` function to convert the symbolic expression into a *NumPy* function. As of the time of writing this guide *SymPy* is still under much development; as such, some auxiliary functions are used in the following example and defined in the appendix. For more information on *SymPy* please consult asldfkjkadjfkj

The following example on how to use *SymPy* with *Pweave* is based on Example 9 from chapter 5 of Introduction to Mathematical Economics by Dr. Edward Dowling. The example show how to use the well known Lagrange Multiplier in order to optimize a function subject to constraint equation. In the example it will be shown how to

We begin by importing sympy into our environment

```
from sympy import *
from sympy.plotting import *
from pylab import *
```

We then need to define two functions the first taken from adkfjaj gives us a gradient function. The second enhances the *SymPy* `latex()` function.

```
#auxiliary function for gradient
grad = lambda func, vars :[diff(func,var) for var in vars]

#auxiliary function for printing sympy into latex
def PrintLaTex(func, inline=False):
    Statement=latex(func)
```

```
if inline:
    print('$'+Statement+'$')
else:
    print('$$'+Statement+'$$')
```

We are now ready to use *SymPy* . We start by creating our functions in the next code chunk. Where first we create our english alphabet symbols. In order to use non-english symbols in *SymPy* we have to directly import them from the `abs` module. Further to print out to latex we set in the `results` option in the chevrons to `'tex'`

```
#define are english alphabet symbols
x, y, z, Z=symbols('x, y, z, Z')

#load in the greek letter lamba
from sympy.abc import lamda

#print out the defined symbols to latex
PrintLaTex([x, y, z, Z, lamda])
```

$$[x, \quad y, \quad z, \quad Z, \quad \lambda]$$

The formula we wish to minimize is $z = 4x^2 + 3xy + 6y^2$. Subject to the constraint $56 = x + y$.

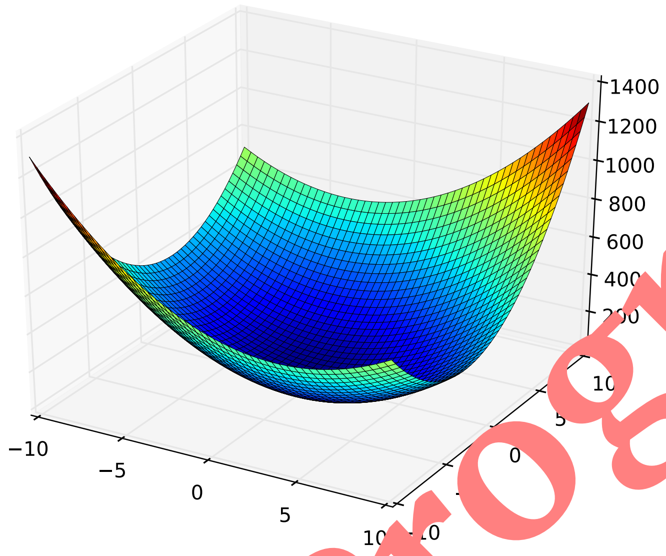
Two things need to be noted in the last paragraph in order to fully use *SymPy* with *Pweave* . The first is that in order to show equalities in *SymPy* we must use the `Eq()` function. It will be shown momentarily how to use this with differentiation, however the results are buggy. Second to print the formula inline we have have used the `<% ...%>` *Pweave* code chunk environment, the `;` to perform multiple operations in *Python* on the same line, and finally we have set the **PrintLaTex()** inline parameter to `True`. The code to show make the last paragraph is shown in the following verbatim section.

```
The formula we wish to minimize is #$z = 4 x^{2} + 3 x y + 6 y^{2}$.
Subject to the constraint $56 = x + y$.
```

SymPy does not normally use formula with equalities instead it uses expressions and by default sets them to 0. Next is shown a 3d plot called from *SymPy* of the z function along with the constraint equation.

```
x, y, z, Z=symbols('x, y, z, Z')
PrintLaTex(zform)
plot3d(zform.args[1], (x, -10, 10), (y, -10, 10))
```

$$z = 4x^2 + 3xy + 6y^2$$



Note that if you use both `print` and plotting in the same *Pweave* code chunk you can not use the `caption` or `name` options

```
print('our constraint equation set to zero')
constraint=Eq(0,cform.args[0]-cform.args[1])
PrintLaTex(constraint)
```

our constraint equation set to zero

$$0 = -x - y + 56$$

```
plot3d(constraint.args[1], (x, -10, 10), (y, -10, 10))
```

To then solve the lagrange multiplier method is employed to create a new function

```
#this is the variable we will calculate with
Zeq=zform.args[1]-lamda*constraint.args[1]

#this is the variable we will use to show the derivation
Zform=Eq(Z, Zeq)
PrintLaTex(Zform)
```

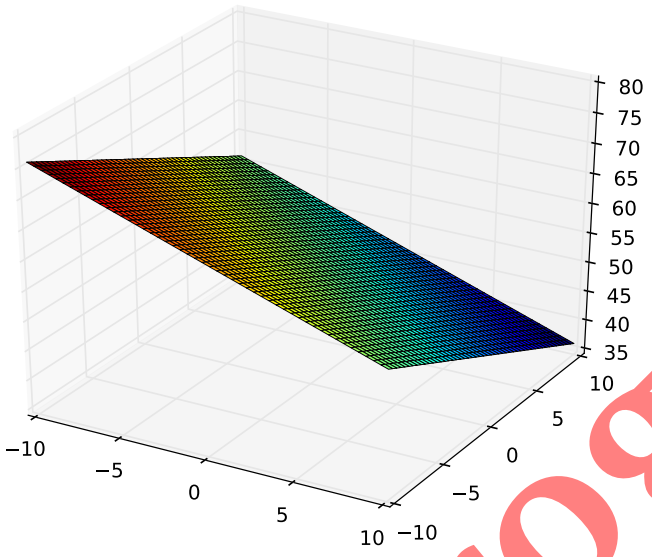


Figure 1: Cons[illegible] Eq[illegible] plot

$$Z = -\lambda\left(-x - y + 56\right) + 4x^2 + 3xy + 6y^2$$

To solve we find the gradie[illegible] of Z and set it to zero; $\nabla Z = 0$. This currently has propom i[illegible] try[illegible] to p[illegible]t out differentiation part in both raw useage and in imple[illegible]nting [illegible] differentiation throughout the whole equality expressions. Till t[illegible]is resolved it will not be shown; however the final solution to the mini[illegible]izat[illegible]n is. GZ=grad(Zeq, [x, y, lamda]) PrintLaTex(GZ) ©

5 Conclusion

6 Further Resources