Data Science With Python

L A Liggett

2019-05-10

Contents

Chapter 1

Prerequisites

This is a *sample* book written in **Markdown**. You can use anything that Pandoc's Markdown supports, e.g., a math equation $a^2 + b^2 = c^2$.

The **bookdown** package can be installed from CRAN or Github:

```
install.packages("bookdown")
# or the development version
# devtools::install_github("rstudio/bookdown")
```

Remember each Rmd file contains one and only one chapter, and a chapter is defined by the first-level heading `#`.

To compile this example to PDF, you need XeLaTeX. You are recommended to install TinyTeX (which includes XeLaTeX): https://yihui.name/tinytex/.

Chapter 2

Introduction

You can label chapter and section titles using `{#label}` after them, e.g., we can reference Chapter 2. If you do not manually label them, there will be automatic labels anyway, e.g., Chapter 4.

Figures and tables with captions will be placed in `figure` and `table` environments, respectively.

And this is some other random stuff.

```
par(mar = c(4, 4, .1, .1))
plot(pressure, type = 'b', pch = 19)
```

Reference a figure by its code chunk label with the `fig:` prefix, e.g., see Figure 2.1. Similarly, you can reference tables generated from `knitr::kable()`, e.g., see Table 2.1.

```
knitr::kable(
  head(iris, 20), caption = 'Here is a nice table!',
  booktabs = TRUE
)
```

```
knitr::include_graphics(rep("knit-logo.png", 3))
```

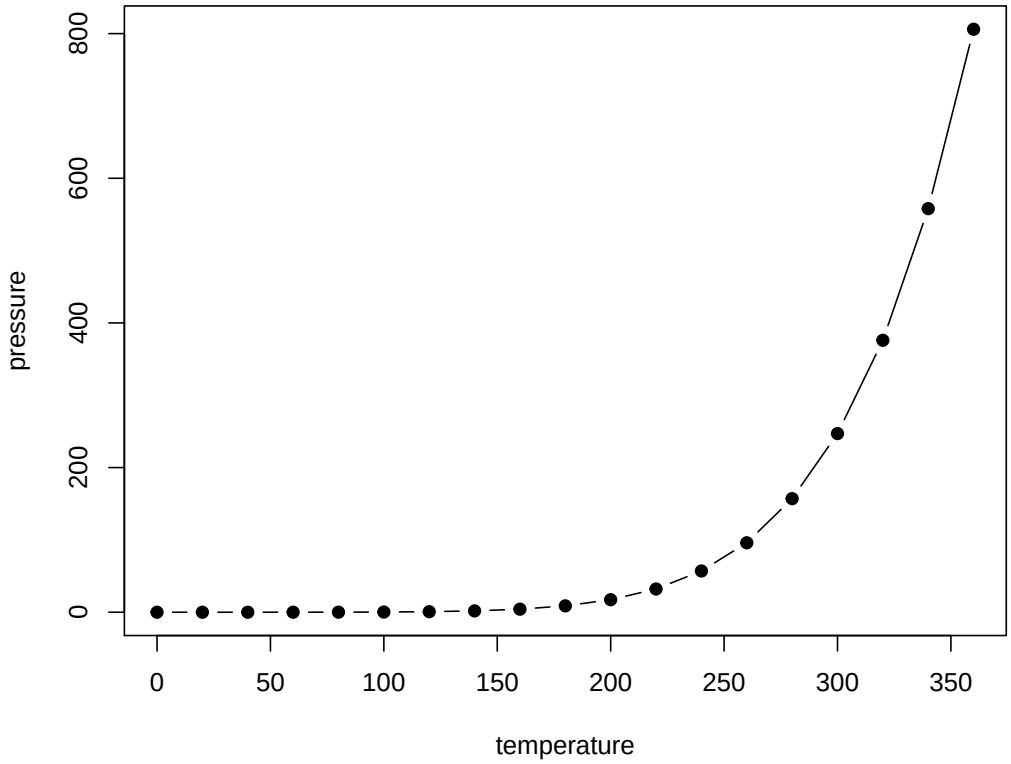

Figure 2.1: Here is a nice figure!

```
knitr::include_app("https://yihui.shinyapps.io/miniUI/",
                   height = "600px")
```

```
import pandas as pd
x = 'hello, python world!'
print(x.split(' '))
```

```
## ['hello,', 'python', 'world!']
```

You can write citations, too. For example, we are using the **bookdown** package (Xie, 2018) in this sample book, which was built on top of R Markdown and **knitr** (**?**).

Table 2.1: Here is a nice table!

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3.0	1.4	0.1	setosa
4.3	3.0	1.1	0.1	setosa
5.8	4.0	1.2	0.2	setosa
5.7	4.4	1.5	0.4	setosa
5.4	3.9	1.3	0.4	setosa
5.1	3.5	1.4	0.3	setosa
5.7	3.8	1.7	0.3	setosa
5.1	3.8	1.5	0.3	setosa

Chapter 3

JupyterLab

Here is a simple template that I use that controls a couple useful things when starting a new notebook.

```
import sys
sys.path.append('../util')

%reload_ext autoreload
%autoreload 2

from util import *
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns

sns.set_palette('pastel')
sns.set_style('ticks')
sns.set_context('paper', font_scale=1)
```

It is often convenient to have a notebook automatically refresh the imported libraries so that they can be modified while working on a JupyterLab notebook.

```
%reload_ext autoreload
%autoreload 2
```

To allow directory organization, dependcies can be separated into different directories and imported into a jupyter notebook using the following import statement.

```
import sys
sys.path.append('../util')
```

Chapter 4

Visualization

4.1 Color

4.1.1 Colorschemes

Seaborn Themes

```
Pastel: {'Blue':'#a3c6ff', 'Orange':'#f7ab60', 'Green':'#60f7a9', 'Red':'#fc9d94', 'Purple':'#bea3ff',
```

```
Deep: {'Green':'#5baf68'}
```

4.1.2 Controlling Coloration

Not all plots automatically plot with a white background, and when using something dark like jupyterlab or a presentation this can be frustrating. The background color can be set in pyplot like this.

```
fig.patch.set_facecolor('xkcd:mint green')
```

When plotting, samples will not always be colored with the same color, especially when different subsets of samples are included in different plots. Here is a manual workaround to specify the coloration of displayed data. This is a bit cumbersome so there might be a more elegant way of achieving the same outcome.

```
# here is an example where sample order is controlled from a pandas DataFrame
sample_order = all_vars.sort_values(['ID']).drop_duplicates(['Sample']).Sample

# the color order is specified here
# colors should be in the same order as the above sample_order Series, excluding samples with no data
colors = [pastel['Brown'], pastel['Blue'],
          pastel['Orange'], pastel['Purple'],
          pastel['Green'], pastel['Red'],
          ]

plt.figure()
# this is an example of plotting that uses the sample_order and palette to control coloration order
sns.catplot(x='Sample', y='VAF', hue='Gene', jitter=True,
            data=oncogenic[oncogenic.Location == 'Peripheral'],
            legend=False, order=sample_order, palette=sns.color_palette(colors))
```

```
# a colorscheme can be specified if desired
pastel = {'Blue':'#a3c6ff', 'Orange':'#f7ab60',
          'Green':'#60f7a9', 'Red':'#fc9d94',
          'Purple':'#bea3ff', 'Brown':'#d1b485',
          'Pink':'#f7afdf', 'Gray':'#c4c4c4',
          'Yellow':'#ffffaa', 'LBlue':'#baf6ff'}

# this controls the coloration in the legend
import matplotlib.patches as mpatches
egfr = mpatches.Patch(color=pastel['Blue'], label='EGFR')
pik3ca = mpatches.Patch(color=pastel['Orange'], label='PIK3CA')
myc = mpatches.Patch(color=pastel['Green'], label='MYC')

plt.legend(handles=[egfr,pik3ca,myc],
           loc='upper right', bbox_to_anchor=(1.5, 1),
           ncol=1) # no legend overlap
```

4.2 Matplotlib

Plotting a heatmap.

```
import matplotlib.pyplot as plt
import numpy as np
a = np.random.random((16, 16))
plt.imshow(a, cmap='RdBu'', interpolation='nearest')
plt.show()
```

Possible heatmap colors are:

```
Accent, Accent_r, Blues, Blues_r, BrBG, BrBG_r, BuGn, BuGn_r, BuPu, BuPu_r, CMRmap, CMRmap_r, Dark2, Da
Set1_r, Set2, Set2_r, Set3, Set3_r, Spectral, Spectral_r, Wistia, Wistia_r, YlGn, YlGnBu, YlGnBu_r, YlG
gist_stern, gist_stern_r, gist_yarg, gist_yarg_r, gnuplot, gnuplot2, gnuplot2_r, gnuplot_r, gray, gray_
twilight, twilight_r, twilight_shifted, twilight_shifted_r, viridis, viridis_r, vlag, vlag_r, winter, w
```

A simple venn diagram.

```
from matplotlib_venn import venn2
venn2(subsets = (3, 2, 1))
```

A more complicated venn diagram.

```
from matplotlib import pyplot as plt
import numpy as np
from matplotlib_venn import venn3, venn3_circles
plt.figure(figsize=(4,4))
v = venn3(subsets=(1, 1, 1, 1, 1, 1, 1), set_labels = ('A', 'B', 'C'))
v.get_patch_by_id('100').set_alpha(1.0)
v.get_patch_by_id('100').set_color('white')
v.get_label_by_id('100').set_text('Unknown')
v.get_label_by_id('A').set_text('Set "A"')
c = venn3_circles(subsets=(1, 1, 1, 1, 1, 1, 1), linestyle='dotted')
c[0].set_lw(1.0)
c[0].set_ls('dotted')
plt.title("Sample Venn diagram")
plt.annotate('Unknown set', xy=v.get_label_by_id('100').get_position() - np.array([0, 0.05]), xytext=(-
```

```
            ha='center', textcoords='offset points', bbox=dict(boxstyle='round,pad=0.5', fc='gray', alp
                         arrowprops=dict(arrowstyle='->', connectionstyle='arc3,rad=0.5',color='gray'))
                         plt.show()
```

4.3 Seaborn

Here is a general bar plot that includes some commonly used parameters.

```
# fits my 22 inch monitor
plt.figure(figsize=(19.17,11.98))
# order controls the display order of the samples
sns.catplot(x="Sample", y="Somatic", kind="bar", data=var_counts, order=labels);
# keeps x-axis labels, but eliminates the tick mark
plt.tick_params(labelbottom=True, bottom=False)
# trim off the x-axis
sns.despine(offset=10, trim=True, bottom=True)
# labels
plt.title('')
plt.ylabel('', fontsize=8)
plt.xlabel('', fontsize=8)
# manual control of xlabels
labels = ['Indiv_1-a','Indiv_2','Indiv_3','Indiv_1-b']
# control xtick order
plt.xticks(range(len(labels)), labels, rotation=45)
# control the number of x-ticks
plt.locator_params(axis='x', nbins=10)
# legend positioning
plt.legend(loc='upper right')
# log scale
plt.gca().set_yscale('log')
# this is better if neg values are needed
plt.gca().set_yscale('symlog')
# fit plot to display
plt.tight_layout()
plt.show()
# save figure with tight_layout
plt.savefig("test.svg", format="svg", bbox_inches="tight", dpi=1000)
```

Signifance information can be added by including p-values and label bars using the following code.

```
x1, x2 = 0, 1 # columns to annotate on the plot
y2, y1 = 20, 15 # placement of the line and how for down the vertical legs go
plt.plot([x1,x1, x2, x2], [y1, y2, y2, y1], linewidth=1, color='k') # stats line
plt.text((x1+x2)*.5, y2+2, "p=0.09", ha='center', va='bottom', fontsize=8) # p-value or sig
```

4.4 Statistics

This is a two-sided T-test for the null hypothesis that two populations have the same means. It is important to note that it assumes the population variances are the same, so this must be changed if the assumption is incorrect.

```
# ttest_ind(a, b, axis=0, equal_var=True, nan_policy='propagate')
from scipy.stats import ttest_ind
ttest_ind(df[df['sample'] == 'one']['means'], df[df['sample'] == 'two']['means'])
```

4.5 Various Plot Styles

This displays each individual datapoint overlayed on a boxplot

```
ax = sns.boxplot(x='day', y='total_bill', data=tips)
ax = sns.swarmplot(x='day', y='total_bill', data=tips, color='.25')
```

Chapter 5

Biology

5.1 General

Some helpful commands for genetic sequence.

```
from string import ascii_uppercase # python 3
from string import upper, lower # python 2
upper('tcga')
lower('TCGA')
title('tcga') # capitalize the first letter
```

5.2 Biopython

Reverse complement of sequence

```
from Bio.Seq import Seq
str(Seq(i).reverse_complement())
```

5.3 UCSC Genome Browser

Get sequence from UCSC genome browser

```
from subprocess import check_output, STDOUT
temp = check_output('wget -qO- http://genome.ucsc.edu/cgi-bin/das/hg19/dna?segment=%s:%s,%s' % (vcfObj.
```

5.4 Ref Genome

Get sequence from reference genome

```
from subprocess import check_output, STDOUT
temp = check_output('samtools faidx %s %s:%s-%s' % (ref, vcfObj.chrom, low, high), stderr=STDOUT, shell=

finalSeq = ''
for line in temp.decode('UTF-8').split('\n'):
```

```
for line in temp.decode('UTF-8').split('\n'): # this is only necessary in python 3 to convert binary to
    if '>' not in line:
        finalSeq += line

finalSeq = finalSeq.upper()
```

5.5 Personal Information

```
# parse vcf file with parseline
if '#' not in line and 'chr' in line: # skip the info
# vcf handling
from parseline import VCFObj
# or
from util import VCFObj
vcfObj = VCFObj(vcfLine)
# available attributes: ao, dp, af, wt, var, chrom, location
```

Chapter 6

Data I/O

6.1 Reading Data Files

Opening .gz files

```
import gzip
for line in gzip.open('myFile.gz'):
    print line
```

6.2 Pickles

Writing data in pickle format

```
import pickle
p = open('principle.pkl', 'wb')
pickle.dump(principleData, p)
p.close()
```

Reading data in pickle format

```
import pickle
p = open('principle.pkl', 'rb')
principleData = pickle.load(p)
p.close()
```

Chapter 7

Pandas

7.1 File I/O

Read a csv file into a DataFrame.

```
pd.read_csv(filepath)
```

7.2 Relabeling

Rename a column or group of columns can be done by passing a dictionary of the changes.

```
    df = df.rename(columns={'a':'b','c':'d'})
```

7.3 Editing Data

Drop columns from a DataFrame.

```
import numpy as np
df = pd.DataFrame(np.arange(12).reshape(3,4),
                  columns=['A', 'B', 'C', 'D'])
print(df)

df = df.drop(columns=['B', 'C']) # may not work in python 2
df = df.drop(['B', 'C'], axis=1) # this works in python 2
print(df)
```

Changing the datatype of a column of data can be done by just changing column type.

```
df.Age = df.Age.astype(str)
```

7.3.1 Replace values

New data can be set within a DataFrame one subset at a time in a way that will avoid the SwttingWithCopyWarning.

```
import pandas as pd
df = pd.DataFrame({'Trait':['Seed_Shape','Seed_Shape','Flower_Color','Flower_Color'],
                   'Phenotype':['Round','Wrinkled','Purple','White']})
df.loc[df.Trait == 'Seed_Shape', 'Affected_Part'] = 'Seed'
df.loc[df.Trait == 'Flower_Color', 'Affected_Part'] = 'Flower'
print(df)
```

```
##           Trait Phenotype Affected_Part
## 0    Seed_Shape     Round          Seed
## 1    Seed_Shape  Wrinkled          Seed
## 2  Flower_Color    Purple        Flower
## 3  Flower_Color     White        Flower
```

7.4 Combining Data Structures

The following merges df and df2 using inner to get the intersection on the Sample column, where indexes are ignored if the merging is performed on a column as in the following example. The other possible merging strategies are: left: use only keys from left frame, similar to a SQL left outer join; preserve key order. right: use only keys from right frame, similar to a SQL right outer join; preserve key order. outer: use union of keys from both frames, similar to a SQL full outer join; sort keys lexicographically. inner: use intersection of keys from both frames, similar to a SQL inner join; preserve the order of the left keys.

```
df = pd.merge(df, df2, how='inner', on=['Sample'])
```

Appending to a Dataframe attaches a DataFrame after another one.

```
df = pd.DataFrame([[1, 2], [3, 4]], columns=list('AB'))
df2 = pd.DataFrame([[5, 6], [7, 8]], columns=list('AB'))
df.append(df2)
```

7.5 Splitting

Remove duplicates

```
x = x[~x.index.duplicated(keep='first')] # most ideal method

data = pd.DataFrame({'k1':['one','two']*3+['two'],'k2':[1,1,2,3,3,4,4]})
data.duplicated() # identify duplicate data
data[‘k1’].duplicated()
data['k1'].drop_duplicates()
data.drop_duplicates['k1'] # this does the same thing as the previous line
data.drop_duplicates(['k1','k2'], keep='last') # drops unique found in k1 and k2 and keeps the last ind
```

7.6 Summarizing

The mean of column values can be calculated where each of the columns is grouped by the data in a specified column.

```
temp[['Sample','VAF','Var_Count']].groupby('Sample').mean()
```

Chapter 8

Git

8.1 Setup

8.1.1 Git Setup

The username and email needs to be added after git is installed.

```
git config --global user.name "me"
git config --global user.email "me@gmail.com"
```

After this information has been set, it can be checked.

```
git config --list
```

8.1.2 Repository Initiation

To setup a repository, create a folder with an initial file like a README and then initiate it.

```
git init
git status
```

8.1.3 Mirror on Online Repository

Create a repository on a repository like github, gitlab, bitbucket, or sourceforge. Then the local git repository can be synched with the online repository.

```
git remote add origin url-of-online-repository-here
git push -u origin master
```

Of course the repository could just be setup first and then cloned.

```
git clone url-of-online-repository-here
```

8.2 Manipulating Commits

8.2.1 Repository Status

The commit history of a repository can be displayed in verbose form and in summarized form.

```
git log
git log --oneline
```

8.2.2 File Checkout

To restore a previous version of a file it can be checked out by first identifying the version to be used using the log history and then restoring the desired file.

```
git log --oneline
git checkout <commit number> file.txt
```

8.2.3 Resetting a Repository

To discard the effect of the previous operation on a file.

```
git reset HEAD file.txt
```

The previous version of the a file can then be restored.

```
git checkout -- file.txt
```

Bibliography

Xie, Y. (2018). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.9.