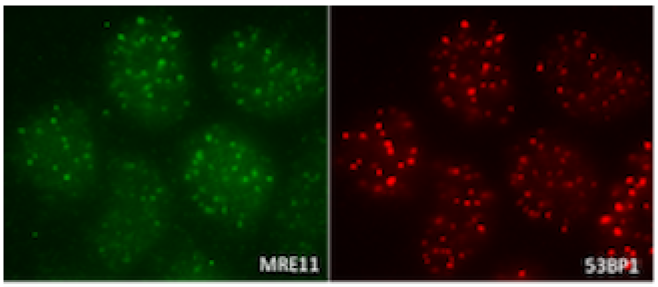
Making boxplot using Python

[illegible]

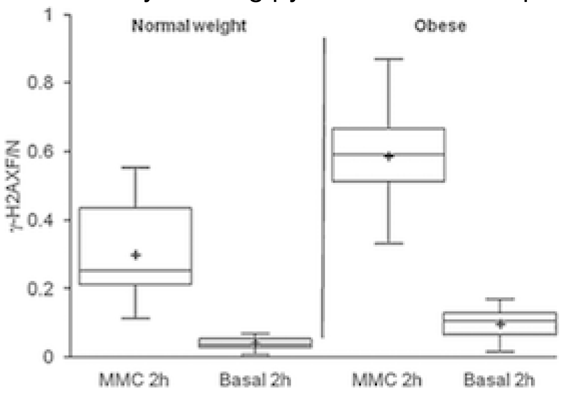
In this small article, I would like to summarize what I have learned about **making boxplot using Python** during the course of "Genomics and Omics Statistical Analysis"

1.The start of my learning Python

First, I would like to briefly explain why I am interested in this topic. I am from Graduate School of Medicine, Department of Radiation Genetics. Our lab is interested in the mechanisms of DNA repair. One of commonly performed experiments in our lab is to compare the accumulation of a target protein in different cell lines. This experiment requires analyzing three hundred cells for each cell line with the number of cell lines ranges from two to eight. Below is a representative image of protein accumulation in nuclear. Raw data is required to be presented as number of bright dots in each cell with each dot means one site of protein accummulation.



Bar graph has been the most common way to present this kind of data. However, dot plot and box plot gradually gain more popularity. My supervisor wants the data to be presented by box plot. With quite a lot of data per experiment and quite a lot of experiments of the same kind, it is more efficient to use statistical softwares for analysis. This is the start of my learning python to make boxplot for my data.

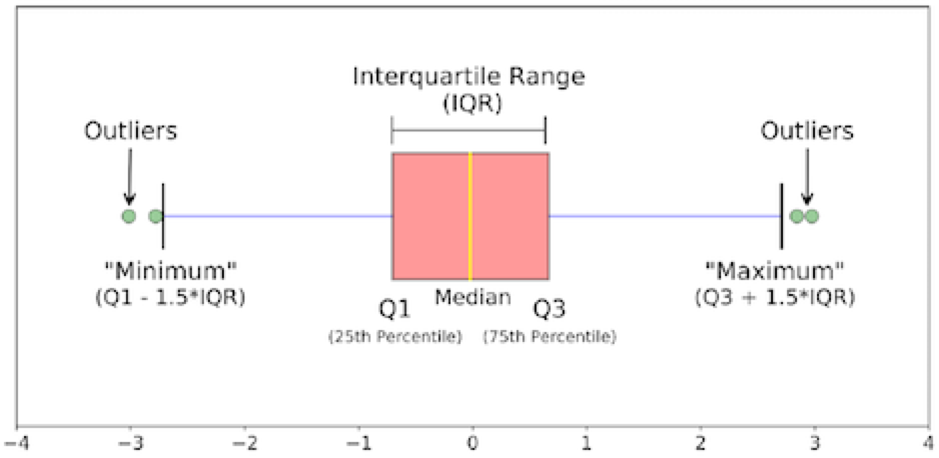


A representative image of boxplot from Alessia Azzarà et al. (Mutation Research 2016) (https://www.researchgate.net/publication/301829523_Different_repair_kinetic_of_DSBs_induced_by_mitomycin_C_in_peripheral_lymphocytes_of_

2.Basic understanding about box plot

I have seen many boxplots in many academic papers in my field; however, I did not fully understand the meaning of each component of a box plot until I came across two nice articles while searching for sample Python code to make box plot. Belows are the links of the two articles.
http://www.physics.csbsju.edu/stats/box2.html (http://www.physics.csbsju.edu/stats/box2.html) (This article describes box plot thoroughly in terms of statistical meaning with three illustrative examples.)
https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51 (https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51) (This article explains carefully about box plot with instructions on how to make and interpret boxplots using Python.)

According to those articles, boxplots are a standardized way of displaying the distribution of data based on a five number summary ("minimum", first quartile (Q1), median, third quartile (Q3), and "maximum").



- median (Q2/50th Percentile): the middle value of the dataset.
- first quartile (Q1/25th Percentile): the middle number between the smallest number (not the "minimum") and the median of the dataset.
- third quartile (Q3/75th Percentile): the middle value between the median and the highest value (not the "maximum") of the dataset.
- interquartile range (IQR): 25th to the 75th percentile.
- "maximum": Q3 + 1.5*IQR
- "minimum": Q1 -1.5*IQR

3.How I started with Python Seaborn package

In the beginning, I googled sample python codes for making boxplot and I got many results, most of which I could not understand much about arguments.
In the course "Genomics and Omics Statistical Analysis" I was taking, I could freely ask any question related to Python and R. When I asked Ms. Mio, a very kind and enthusiastic teaching assistant of the course, she recommended me trying **Seaborn** package for data visualization. I also learned from her about **stacking()** function in **Pandas** package to reshape my data.
To be honest, at that time, I did not know anything about these two packages and I wished there were an illustrative summary of Python packages with their functions so that I could have a general idea about Python packages. And my wish came true when I came across Seaborn **cheat sheet**. At that time, I realized that what exactly I wanted is expressed in the term **cheat sheet**. Later I knew that depending on the sources or websites, cheat sheet about the same package may have different content and visualization.

4.Python Seaborn package

After many searches, I realized that the first website I should have visited to learn about Seaborn package was the official website of this package: https://seaborn.pydata.org/installing.html (https://seaborn.pydata.org/installing.html)
Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. It is necessary to import other libraries in order to use Seaborn. Mandatory dependencies are:

- numpy (>= 1.9.3)
- scipy (>= 0.14.0)
- matplotlib (>= 1.4.3)
- pandas (>= 0.15.2)

Another website I really like is **datacamp** (https://www.datacamp.com/). I have come across this website many times when I try to learn more about Seaborn package. **Python Seaborn Tutorial** (https://www.datacamp.com/community/tutorials/seaborn-python-tutorial) on this page provides clear explanations about basics of Seaborn with an interactive environment of embedded executable codes on the page.

I also found a wonderful **Python Seaborn Cheatsheet** (https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python_Seaborn_Cheat_Sheet.pdf) which explains main functions of Seaborn package illustratively with codes and comments. That cheat sheet gave me an idea how I could use the functions Seaborn package to my own data. Here is the link for the cheat sheet. https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python_Seaborn_Cheat_Sheet.pdf (https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python_Seaborn_Cheat_Sheet.pdf)

5.Creating plots with Seaborn package of Python

The basic steps to creating plots with Seaborn are:

- 1.Import the necessary libraries
- 2.Prepare your data
- 3.Control figure aesthetics
- 4.Plot with Seaborn
- 5.Further customize your plot
- 6.Show the plot

5.1. Import the necessary libraries ¶

```
# import library using alias昭和
import昭和matplotlib.pyplot昭和as昭和plt昭和昭和
import昭和numpy昭和as昭和np昭和
import昭和seaborn昭和as昭和sns昭和
import昭和pandas昭和as昭和pd昭和
```

Other related commands which might be used

```
# import specific class of specific package昭和
from昭和pandas昭和import昭和(EXE*VEQ 昭和
```

5.2. Prepare your data, reshape your own data with DataFrame in Pandas package: stack() and unstack()

I have come across many example codes to make box plot. However, I noticed that most of the examples use either built- in dataset or random dataset generated using DataFrame. I struggled to apply example codes I found online to my own data.
Ms. Mio, as mentioned above, recommended me **Pandas** package and its **stacking()** function. After discussion with Ms. Mio, I realized the importance of **understanding data structure used in example codes** in order to apply those codes to your own data smoothly.
The following is a brief summary of good sources I have found about Pandas package and stacking() function.

- Official page of pandas package (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html)
- "Pandas Tutorial: DataFrames in Python" from datacamp (https://www.datacamp.com/community/tutorials/pandas-tutorial-dataframe-python)
  Pandas package offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things. The **DataFrame** is one of these structures.

- **"Reshaping in pandas"** from https://nikgrozev.com/2015/07/01/reshaping-in-pandas-pivot-pivot-table-stack-and-unstack-explained-with-pictures/ (https://nikgrozev.com/2015/07/01/reshaping-in-pandas-pivot-pivot-table-stack-and-unstack-explained-with-pictures/)
  Among the most common ways of reshaping data are: **pivoting, stacking, unstacking and melting**.
  - **Stacking** a DataFrame means moving the innermost column index to become the innermost row index.
  - The inverse operation is called **unstacking**. It means moving the innermost row index to become the innermost column index.

  stack-unstack1.png

Load the data

- To start working with a **built-in Seaborn data set**, you can make use of the **load_dataset()** function.

```
MMM昭和昭和RW←PSEHCHEXEWIX⌫⇱MMM⇱ 昭和
```

- To load in data your own data, you can use function **read_csv()** in **pandas** package-which is denoted as pd in the command below.

```
import昭和pandas昭和as昭和pd昭和昭和昭和昭和昭和昭和昭和昭和昭和昭和昭和昭和昭和昭和# Load the Pandas libraries with alias 'pd'昭和
HEXE昭和昭和TH←VIEHCGWZ⌫⇱JMPIREQI←GWZ⇱ 昭和昭和# Read data from file 'filename.csv' 昭和
HEXE←LIEH⌫ 昭和昭和昭和昭和昭和昭和昭和昭和昭和昭和昭和昭和昭和昭和昭和昭和昭和# Preview the first 5 lines of the loaded data昭和
```

I also came across a page which explains in detail about **how data is represented inside CSV files.** https://www.shanelynn.ie/python-pandas-read_csv-load-data-from-csv-files/ (https://www.shanelynn.ie/python-pandas-read_csv-load-data-from-csv-files/)

Below is an example of my data file.

In [illegible]:

```
import pandas as pd
data = pd.read_csv("/Users/tsudamasataka/Downloads/dotplot_gH2AX[illegible].csv")
data.head()
```

Out[illegible]:

	Cell	Time	FocigH2AX
0	WT	0.0	1
1	WT	0.0	0
2	WT	0.0	0
3	WT	0.0	1
4	WT	0.0	3

5.3. Control figure aesthetics

The following is a set of commands to control figure arguments which I collected from Python Seaborn Cheatsheet I mentioned above.

```
VRW-W X⌫ #(Re)set the seaborn default
VRW-W XCVX]PI ⌫[LMXI KVMH #Set the matplotlib parameters
VRW-W XCVX]PI ⌫XMGOW ⌫XMGO-QENSV-WM^I ”⌫]XMGO-QENSV-WM^I ”a #Set the matplotlib p
arameters
VRW-E¥I WCVX]PI ⌫[LMXI KVMH #Return a dict of params or use with with to temporarily set the
 style
```

I further googled sns.set() and got to know that **"Controlling figure aesthetics" section on Seaborn official page** (https://seaborn.pydata.org/tutorial/aesthetics.html) provides a very easy-to-understand and illustrative explanations about this topic. There are five preset seaborn themes: darkgrid, whitegrid, dark, white, and ticks. The default theme is darkgrid.
Googling sns.set_style also brought me to **Seaborn official page**. Parameters in this function are explained, followed by short example code. https://seaborn.pydata.org/generated/seaborn.set_style.html (https://seaborn.pydata.org/generated/seaborn.set_style.html)
Parameters: style : dict, None, or one of {darkgrid, whitegrid, dark, white, ticks}

- Next are commands from the page https://www.datacamp.com/community/tutorials/seaborn-python-tutorial (https://www.datacamp.com/community/tutorials/seaborn-python-tutorial)
  - **Set context, scale font elements** and override param mapping. The four predefined contexts are "paper", "notebook", "talk" and "poster".

    ```
    VRW-W XCGSRXI ¥X⌫RSXI FSSO JSRXCWGEPI ! ⌘ VG _PMRI W-PMRI [MHXL ”✓a
    VRW-W XCGSRXI ¥X⌫TETI V SRXCWGEPI ! ◆ VG _JSRX-WM^I ”⌫E¥I W-PEFI PWM^I ”a
    ```

  - **Color Palette**

    ```
    VRW-W XCTEPI XXI ⌫LYWP ◆ #Define the color palette
    VRW-GSPSVCTEPI XXI ⌫LYWP #Use with with to temporarily set palette
    JPEXYM “F“F “ F “EE “GG “ GG ⌘ A
    VRW-W XCTEPI XXI ⌫JPEXYM #Set your own color palette
    ```

  For this part, it is also a good idea to start from **Seaborn official page** (https://seaborn.pydata.org/index.html) for the relevant function you want to know more.

5.4. Plot your data with Seaborn

- Plot your data in boxplot form with **sns.boxplot()** function in Seaborn. Here are more detail example commands.

```
VRW←FS¥TPSX⌫¥! ⩓EPMZI ⩓⎘]! ⩓EKI ⩓⎘LYI ! ⩓EHYPXCQEPI ⩓⎘HEXE! XMXERMG 昭和
VRW←FS¥TPSX⌫HEXE! MVMW⎘SVMI RX! ⩓L⩓ 昭和 #Boxplot with wide-form data昭和
```

- Depend on the number of numerical variables in your data, **the command using sns.boxplot() might be slightly different.** (In the section "Basic Boxplot with Seaborn" from [The Python Graph Gallery (https://python-graph-gallery.com/30-basic-boxplot-with-seaborn/)](https://python-graph-gallery.com/30-basic-boxplot-with-seaborn/))
  The example below using built-in data set 'iris' to illustrate the differences.

```
HJ昭和昭和RW←PSEHCHEXEWIX⌫⎣MVMW⎦ 昭和#load in built-in data ser 'iris'昭和
```

  - If you have only one numerical variable, you can use this code to get a boxplot with only one group.

```
VRW←FS¥TPSX⌫昭和! HJ?⩓WITEPCPI RKXL⩓A昭和昭和
```

  - Let’s say we want to study the distribution of a numerical variable, but for each group separately. Here we study the sepal length of 3 species of flower.

```
VRW←FS¥TPSX⌫昭和¥! HJ?⩓WTI GMI W⩓A⎘昭和! HJ?⩓WITEPCPI RKXL⩓A昭和昭和
```

  - Finally we can study the distribution of several numerical variables, let’s say sepal length and width:

```
VRW←FS¥TPSX⌫HEXE! HJ←M¥?”⎘”✓A 昭和
```

5.5. Further customize your plot

Here are the codes from [**Seaborn Cheat Sheet**](https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python_Seaborn_Cheat_Sheet.pdf) (https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python_Seaborn_Cheat_Sheet.pdf) which is a good quick reference to me how to customize my plot, regardless of box plot or not.
Let's see the example with **lmplot**. Note that **g** is the plot made by function **sns.lmplot()** to the dataset **tips**

```
import matplotlib.pyplot as plt
import seaborn as sns
tips = sns.load_dataset("tips")
g = sns.lmplot(x="tip", y="total_bill", data=tips, aspect=2)
```

To customize Axisgrid Objects

```
g.despine(left=True)                    #Remove left spine
g.set_ylabels("Survived")               #Set the labels of the y-axis
g.set_xticklabels(rotation=45)          #Set the tick labels for x
g.set_axis_labels("Survived", "Sex")    #Set the axis labels
h.set(xlim=(0,5), ylim=(0,5), xticks=[0,2.5,5], yticks=[0,2.5,5])    #Set the limit and ticks
 of the x-and y-axis
```

To customize title, axis labels and axit limits of your plot

```
plt.title("A Title")                    #Add plot title
plt.ylabel("Survived")                  #Adjust the label of the y-axis
plt.xlabel("Sex")                       #Adjust the label of the x-axis
plt.ylim(0,100)                         #Adjust the limits of the y-axis
plt.xlim(0,10)                          #Adjust the limits of the x-axis
plt.setp(ax,yticks=[0,5])               #Adjust a plot property
plt.tight_layout()                      #Adjust subplot params
```

Please note that **plt** is the alias for **matplotlib.pyplot**. As Seaborn is a Python data visualization library based on matplotlib, you can adjust your plot made with seaborn package using function from matplotlib package as shown above.

I also found on Python Graph Gallery (https://python-graph-gallery.com/) and datacamp (https://www.datacamp.com/community/tutorials/seaborn-python-tutorial) several other factors you can customize to your plot.

- Order group in a specific order https://python-graph-gallery.com/35-control-order-of-boxplot/ (https://python-graph-gallery.com/35-control-order-of-boxplot/)
- Add jitter over boxplot https://python-graph-gallery.com/36-add-jitter-over-boxplot-seaborn/ (https://python-graph-gallery.com/36-add-jitter-over-boxplot-seaborn/)
- Rescale axis with logarithmic https://www.datacamp.com/community/tutorials/seaborn-python-tutorial (https://www.datacamp.com/community/tutorials/seaborn-python-tutorial)
- Calculate number of observation per group and median to position labels https://python-graph-gallery.com/38-show-number-of-observation-on-boxplot/ (https://python-graph-gallery.com/38-show-number-of-observation-on-boxplot/)

5.6. Show the plot

- Show the plot image using **sns.plt.show()**:

```
WRW↵TPX↵WLS[⌫昭和
```

- Other related functions

```
TPX↵WEZI JMK⌫⇗JMPI REQI ↵TRK⇗ 昭和昭和昭和昭和昭和昭和昭和昭和昭和昭和昭和昭和昭和#Save the plot as a figure昭和
TPX↵WEZI JMK⌫⇗JMPI REQI ↵TRK⇗⍗昭和XVERWTEVI RX! 8VYI  昭和#Save transparent figure昭和
```

6.Boxplot with Python Seaborn package

Now you have an overview about making plot using Seaborn package. Let's go into detail about box plot. Again, on the Seaborn official page, I found a full explanation about [seaborn.boxplot() function (https://seaborn.pydata.org/generated/seaborn.boxplot.html)](https://seaborn.pydata.org/generated/seaborn.boxplot.html)
Full set of **parameters** with their default values in **seaborn.boxplot()** function is shown as below.

```
WIEFSVR↵FS¥TPSX⌫¥! 2SRI ⍗昭和! 2SRI ⍗昭和YI ! 2SRI ⍗昭和EXE! 2SRI ⍗昭和SVHI V! 2SRI ⍗昭和YI CSVHI V! 2SRI ⍗昭和SVMI RX! 2
SRI ⍗昭和GSPSV! 2SRI ⍗昭和TEPI XXI ! 2SRI ⍗昭和WEXYVEXMSR! ⌘↵⌧⍗昭和[MHXL! ⌘↵⍗昭和SHKI ! 8VYI ⍗昭和PMI WMN ! ⌧⍗昭和MRI [
MHXL! 2SRI ⍗昭和[LMW⌘↵⌧⍗昭和RSXGL! *EPWI ⍗昭和¥! 2SRI ⍗昭和↦O[ EVKW昭和
```

- x, y, hue : names of variables in data or vector data, optional
  Inputs for plotting long-form data.
- data : DataFrame, array, or list of arrays, optional
  Dataset for plotting. If x and y are absent, this is interpreted as wide-form. Otherwise it is expected to be long-form.
- order, hue_order : lists of strings, optional
  Order to plot the categorical levels in, otherwise the levels are inferred from the data objects.

7.List of sources I found useful to me and Take home message

- **Official tutorial** for each Python package: [Official seaborn tutorial (https://seaborn.pydata.org/tutorial.html)](https://seaborn.pydata.org/tutorial.html), [Pandas tutorials (http://pandas.pydata.org/pandas-docs/version/0.15/tutorials.html)](http://pandas.pydata.org/pandas-docs/version/0.15/tutorials.html), [Matplotlib tutorials (https://matplotlib.org/tutorials/index.html)](https://matplotlib.org/tutorials/index.html)
  It might be a good idea to first search from the official tutorial for the relevant function you want to know. **Just google the function you want to know and choose the official package page to read first.**
- **Datacamp, towardsdatascience, Python graph gallery** are also good websites to learn as they provides clear and illustrative explanations with examples.
  When you read example codes online and do not understand some commands, it might be useful to **figure out which name of parameter, function and also name of package used in the command** to further google it and read official tutorial (if available) for better understanding.
- **[Seaborn Cheat Sheet](https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python_Seaborn_Cheat_Sheet.pdf)** [(https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python_Seaborn_Cheat_Sheet.pdf)](https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python_Seaborn_Cheat_Sheet.pdf) as a quick vizual reference for Seaborn package. ### ! Remember the term Cheat Sheet when you want to get an overview in an illustrative way about anything.

Sources about writing and coding in Jupyter notebook and GitHub

Below is the list of sources I referenced a lot to make in article look nicely organized.

- **Keyboard shortcuts in Jupyter notebook**
  List of keyboard shortcuts in Jupyter can be found under the menu at the top: **Help > Keyboard Shortcuts.**
  Many other Jupyter notebook tips or tricks can be found in the following page:
  https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts/
  (https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts/)
- **Basic writing and formatting syntax on GitHub**
  The page below is very useful for preparing file like this using Jupyter notebook in markdown mode.
  https://help.github.com/articles/basic-writing-and-formatting-syntax/
  (https://help.github.com/articles/basic-writing-and-formatting-syntax/)
  https://daringfireball.net/projects/markdown/syntax#backslash
  (https://daringfireball.net/projects/markdown/syntax#backslash) (Daring Fireball's "Markdown Syntax.")
  https://guides.github.com/features/mastering-markdown/
  (https://guides.github.com/features/mastering-markdown/)