jupyter_lab_coding_club

April 18, 2023

Index

0.1 Unix commands

To run Unix commands in a Jupyter Notebook, prepend `!` to the command. In the code block below, the `ls -1` command is used to list contents of my current working directory one item per line. The results of `ls -1` are sent to `head -5` using the pipe (|) symbol to print only the first five results.

[1]:
```
# Running Unix commands
!ls -1 | head -5
```

```
VLookup_in_R
VLookup_with_R.html
VLookup_with_R.md
VLookup_with_R.qmd
VLookup_with_R_files
```

0.2 Python code

We can work with Python code in Jupyter Notebook. Python has many external packages that can enhance its functions. These can be found at pypi.org.

Python packages used in this coding club session We use the the `import` command to activate Python packages. Below are what we used in the coding club session.

- pandas: to work with tabular data
- numpy: to perform mathematical computation
- plotting
  - seaborn
  - matplotlib
- ipywidgets: to create interactive Jupyter Notebook elements

[2]:
```
# Activate necessary packages
import pandas
import numpy
```

```python
import seaborn
import matplotlib.pyplot as plt
import session_info
from numpy import random
import ipywidgets as widgets
```

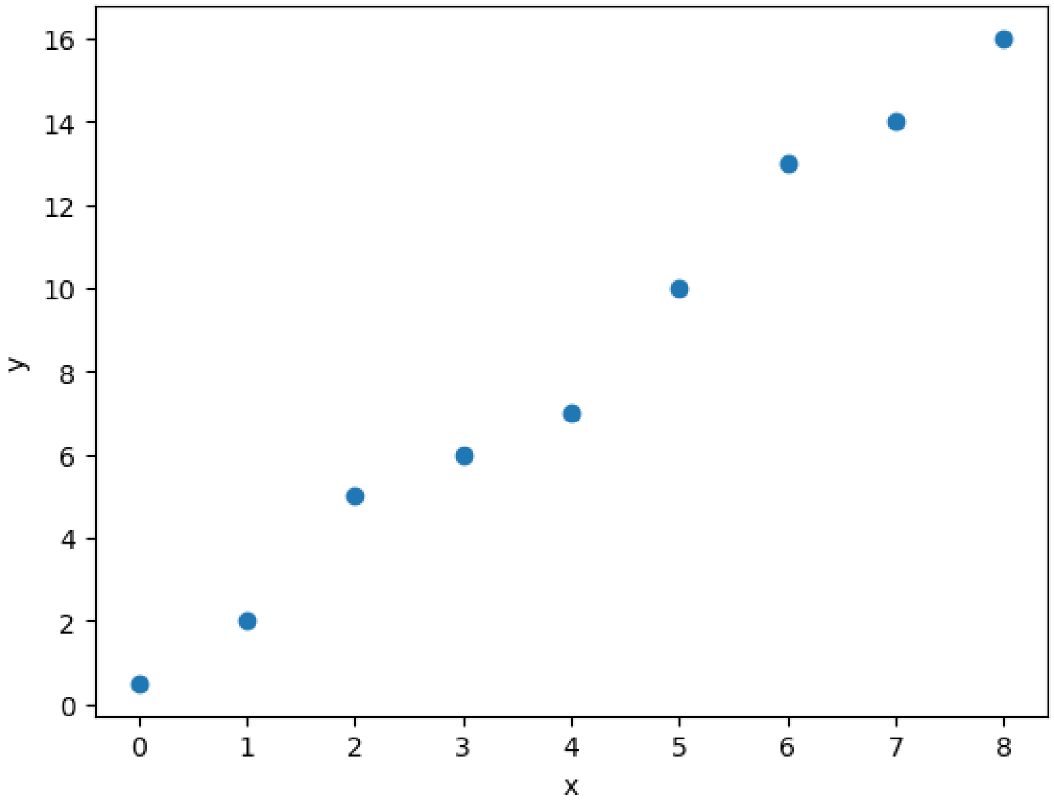
Data visualization with Python In this sectionn we will use Python to explore data.

1. Create two numeric vectors, x and y using `numpy.array` and then use matplotlib to creat the following visualizations
    - Scatter plot
    - Scatter plot with regression line and equation
    - Line plot
2. Generate a list of random numers using `random.randint`, which is a numpy module and then create
    - A histogram using seaborn
    - An interactive histogram where users can specify binwidth
3. Create a bar plot of a made up dataset
4. Import a gene expression dataset using `pandas.read_csv` and create
    - Expression heatmap
    - Expression density plot
    - Interactive volcano plot
5. Print out a list of Python packages and versions used

[3]:
```python
# Create numeric vectors, x and y
x=numpy.array([0,1,2,3,4,5,6,7,8])
y=numpy.array([0.5,2,5,6,7,10,13,14,16])
```

[4]:
```python
plt.scatter(x,y) # creates x,y scatter plot
plt.xlabel('x') # labels x axis
plt.ylabel('y') # labels y axis
```

[4]:
```
Text(0, 0.5, 'y')
```

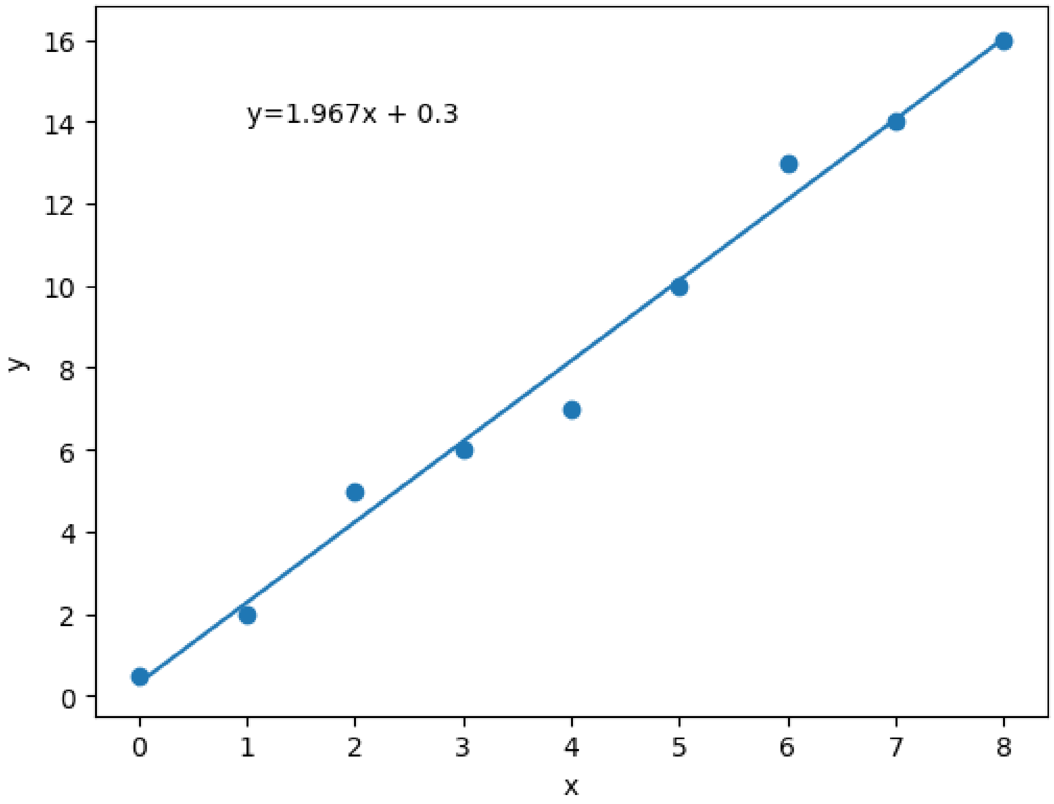


[5]:

```
plt.scatter(x,y) # creates scatter plot
slope, intercept=numpy.polyfit(x,y,1) # calculate slope and intercept
plt.plot(x,slope*x+intercept) # plot regression line
plt.text(1,14,'y='+str(round(slope,3))+'x' ' + ' + str(round(intercept,3))) #␣
 ↪plot regression equation
plt.xlabel('x') # labels x axis
plt.ylabel('y') # labels y axis
```

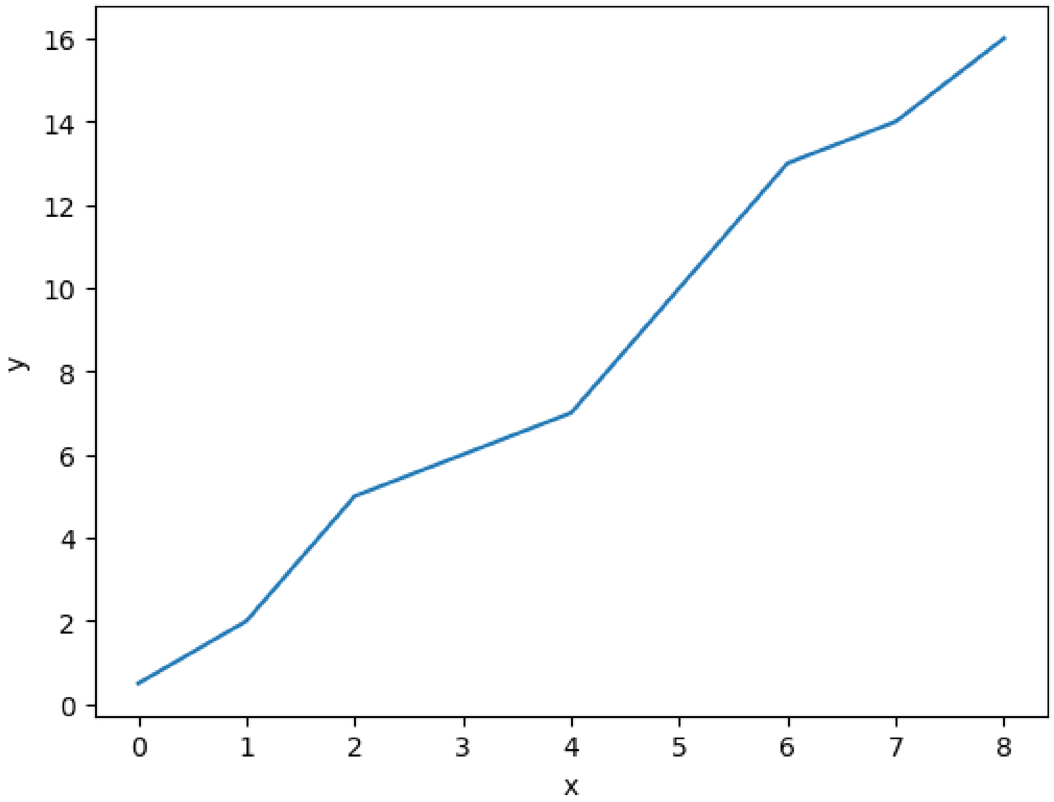
[5]:

```
Text(0, 0.5, 'y')
```



[6]:
```
plt.plot(x,y) # creates line plot
plt.xlabel('x') # labels x axis
plt.ylabel('y') # lables y axis
```

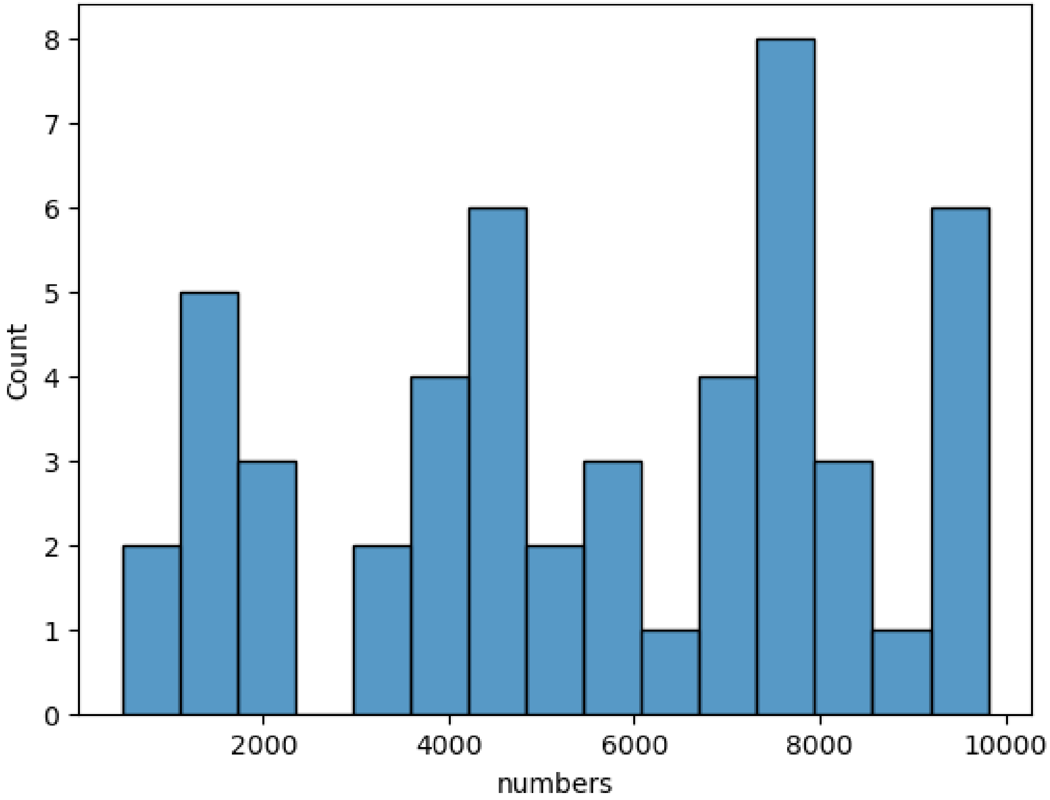
[6]:
```
Text(0, 0.5, 'y')
```



[7]:
```
# Generate random number list
my_random_list = random.randint(0, 10000, size=(50))
```

[8]:
```
# Create histogram of random number list
seaborn.histplot(my_random_list,bins=15)
plt.xlabel('numbers')
```

[8]:
```
Text(0.5, 0, 'numbers')
```



[9]:
```
# Create interactive histogram
def my_binwidth(bw=10):
    seaborn.histplot(my_random_list,bins=bw)
    plt.xlabel('numbers')

widgets.interact(my_binwidth,bw=10)
```

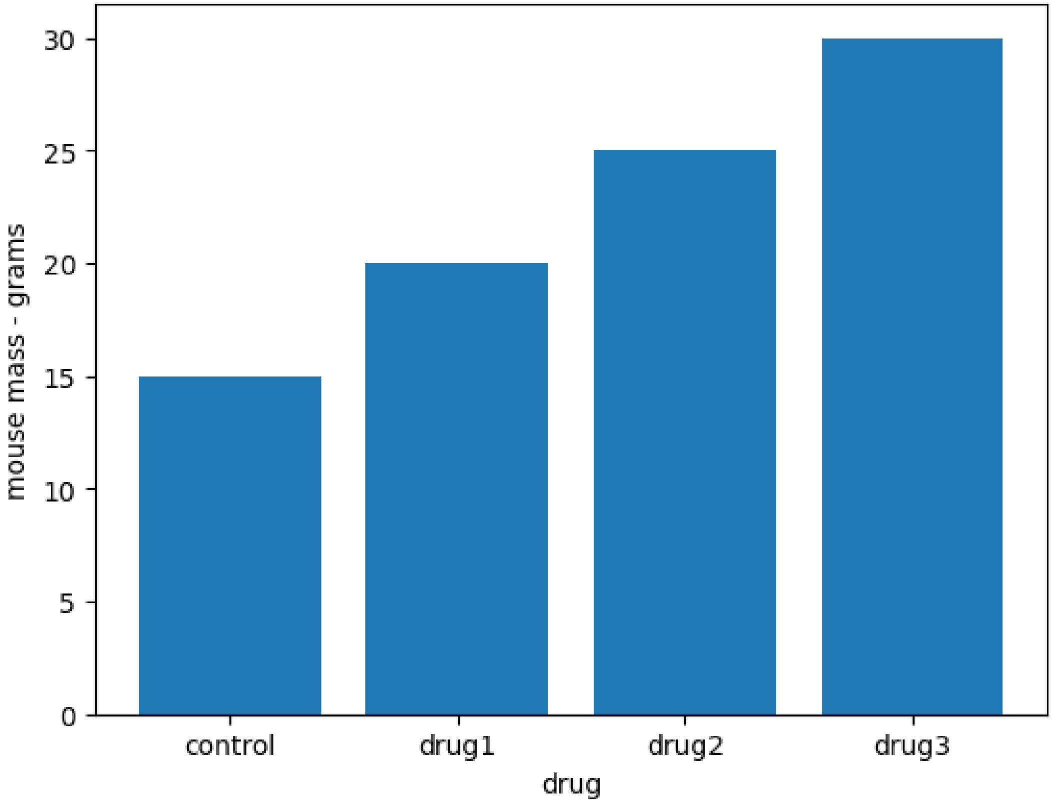
```
interactive(children=(IntSlider(value=10, description='bw', max=30, min=-10),␣
↪Output()), _dom_classes=('widget…
```

[9]:
```
<function __main__.my_binwidth(bw=10)>
```

[10]:
```
# Generate a character vector with mouse treatment group
mouse=['control','drug1','drug2','drug3']
# Generate numeric vector with mouse weight
weight=[15,20,25,30]
```

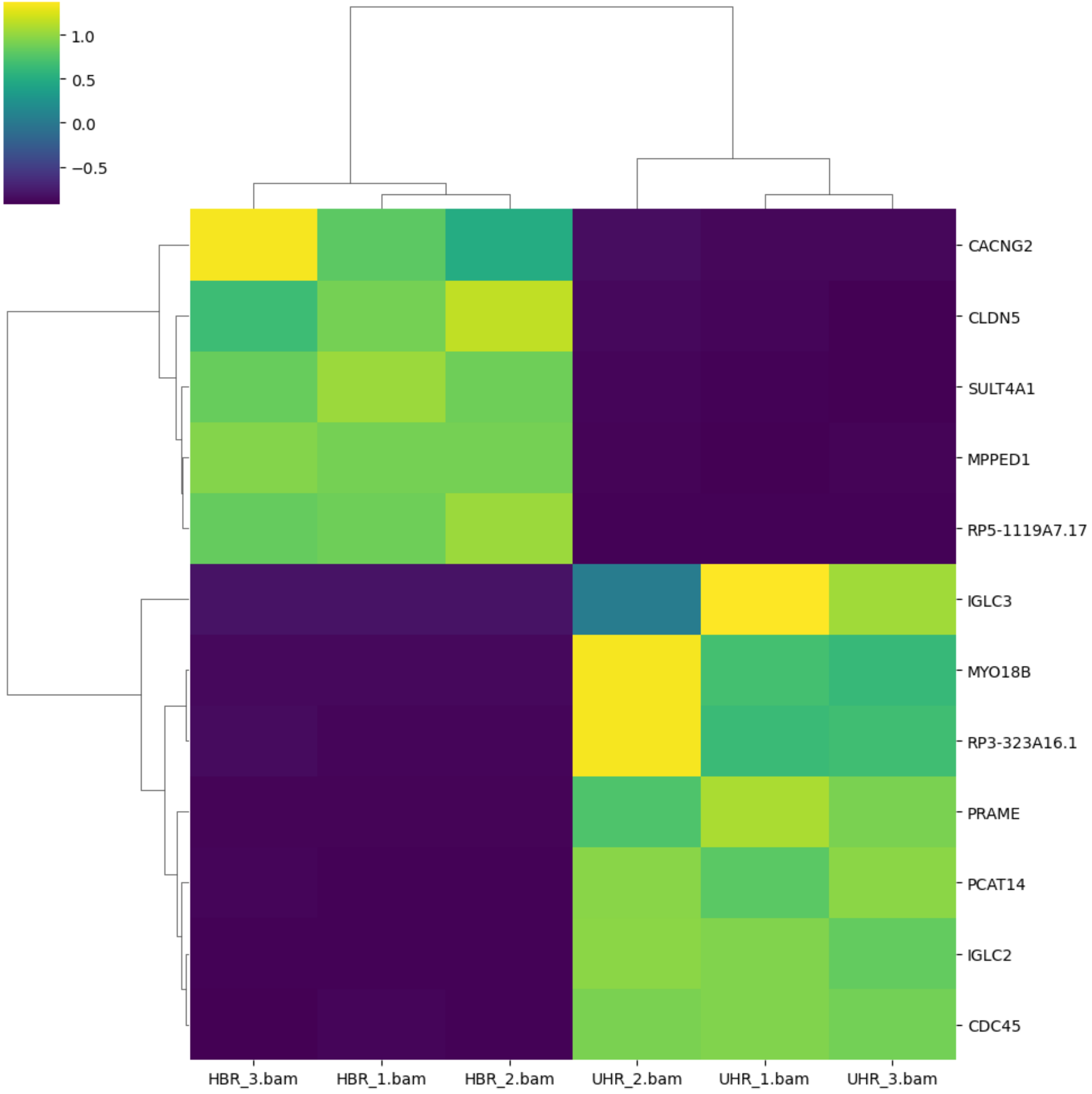
[11]:
```
plt.bar(mouse,weight) # create bar plot
plt.xlabel('drug')
plt.ylabel('mouse mass - grams')
```

[11]:
```
Text(0, 0.5, 'mouse mass - grams')
```



```
[12]: # Import gene expression dataset
      counts1=pandas.read_csv("/Users/wuz8/onedrive/GAU/BTEP/BTEP_Coding_Club/data/
        ↪S2_JUPYTER_NOTEBOOK/hbr_uhr_normalized_counts.csv", index_col=[0])

      # Create heatmap
      seaborn.clustermap(counts1,z_score=0,cmap="viridis")
```

```
[12]: <seaborn.matrix.ClusterGrid at 0x197f2c880>
```

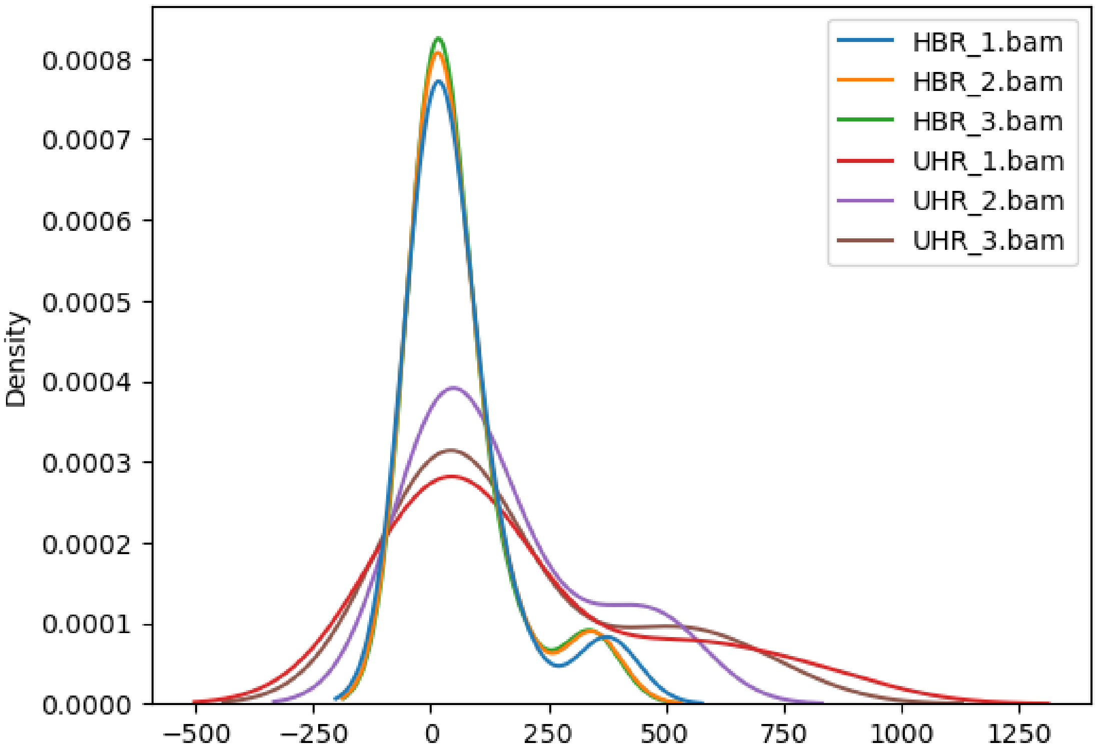


[13]:

```
# Density plot of gene expression
seaborn.kdeplot(data=counts1)
```

[13]:

```
<Axes: ylabel='Density'>
```



[14]:
```
# Activate more Python packages
from IPython.display import display
import plotly
import dash_bio
```

[15]:
```
# Import dataset
hbr_uhr_deg=pandas.read_csv("/Users/wuz8/onedrive/GAU/BTEP/BTEP_Coding_Club/
 ↪data/S2_JUPYTER_NOTEBOOK/hbr_uhr_deg_chr22_results.csv")
```

[16]:
```
padj_cutoff=widgets.FloatText(
    value=0.05,
    description='Adjusted p value cutoff:',
    continuous_update=True,
    disabled=False
)

hbr_uhr_sig_deg=None

def hbr_uhr_deg_filtering(padj):
    hbr_uhr_deg_filtered=hbr_uhr_deg[hbr_uhr_deg.PAdj<=padj]
    global hbr_uhr_sig_deg
    hbr_uhr_sig_deg=hbr_uhr_deg_filtered
```

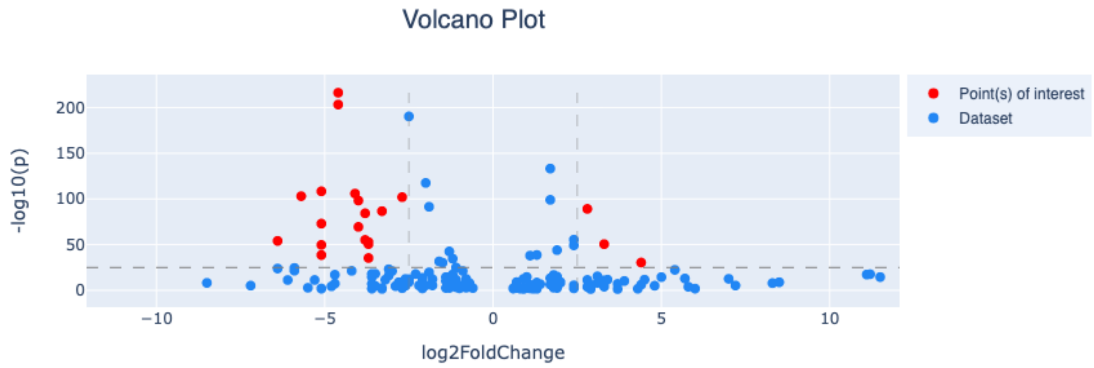
```
    display(hbr_uhr_deg_filtered)

widgets.interact(hbr_uhr_deg_filtering, padj=padj_cutoff)
```

```
interactive(children=(FloatText(value=0.05, continuous_update=True,␣
 ↪description='Adjusted p value cutoff:'), O…
```

```
[16]: <function __main__.hbr_uhr_deg_filtering(padj)>
```

[17]:

```
# Create interactive volcano plot
dash_bio.VolcanoPlot(dataframe=hbr_uhr_sig_deg,gene='name',
                     effect_size='log2FoldChange',p='PAdj',logp=True,
                     snp=None,xlabel="log2FoldChange",
                     genomewideline_value=25,effect_size_line=[-2.5,2.5],
                     point_size=8)
```



[18]:

```
session_info.show()
```

```
[18]: <IPython.core.display.HTML object>
```

0.3 R code

To run R code in a Jupyter Notebook dedicated to Python, we could install and load the package `rpy2.ipython`. Each line of R code should begin with `%%R`.

There are also many external packages that enhance the function of R. These can be found at CRAN and Bioconductor.

0.4 R packages used in this session

We use the `library` command to activate R packages. Here, we use

- pheatmap

- ggplot2
- ggfortify

0.5 Data visualization with R

We will create the following data visualization with R

- Gene expression heatmap using pheatmap
- PCA plot using ggfortify

We will also print out the packages and versions used in this analysis with R.

[19]:

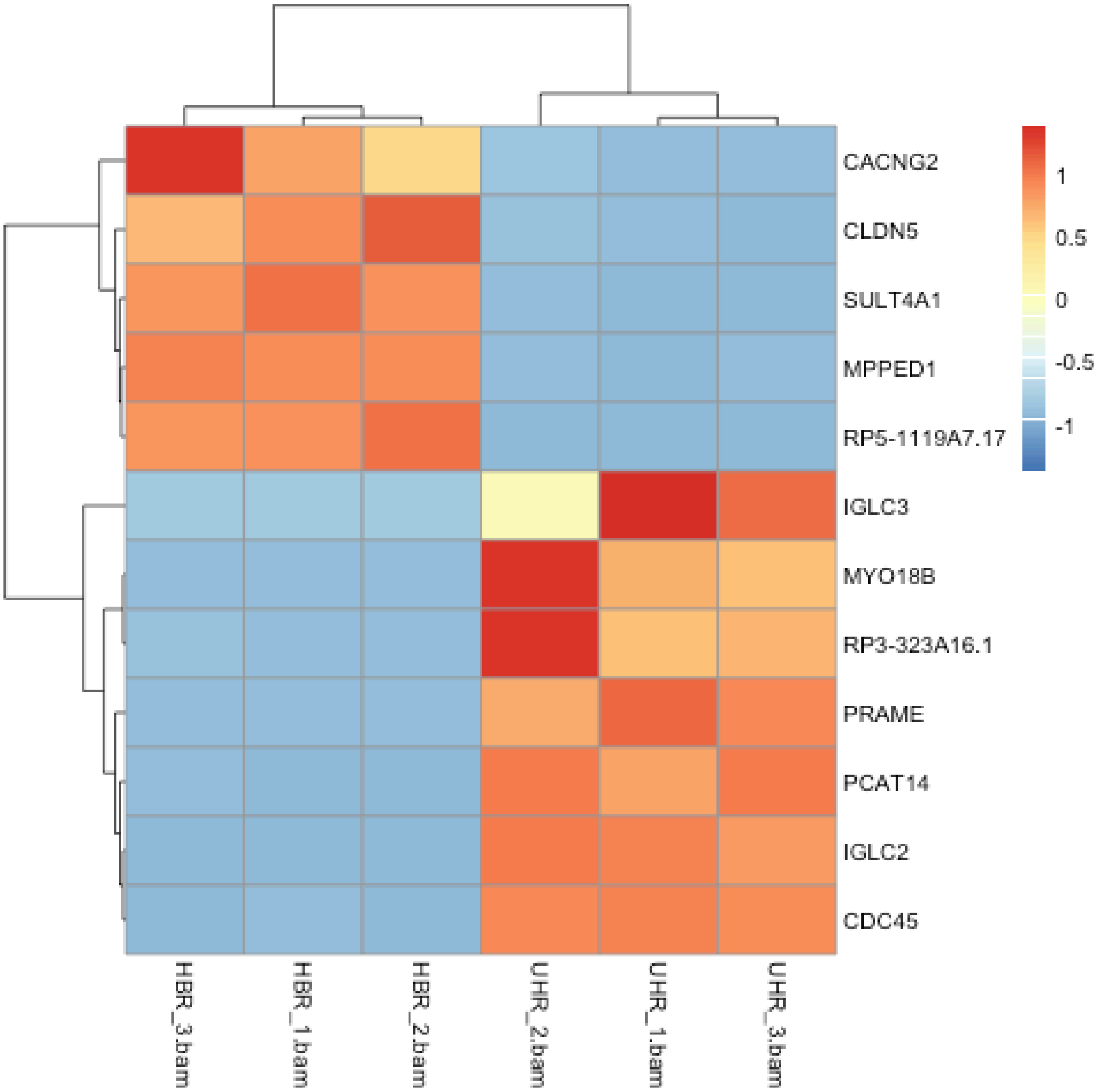
```
# Load rpy2.ipython
%load_ext rpy2.ipython
```

[20]:

```
%%R
# Activate necessary packages
library(pheatmap)
```

[21]:

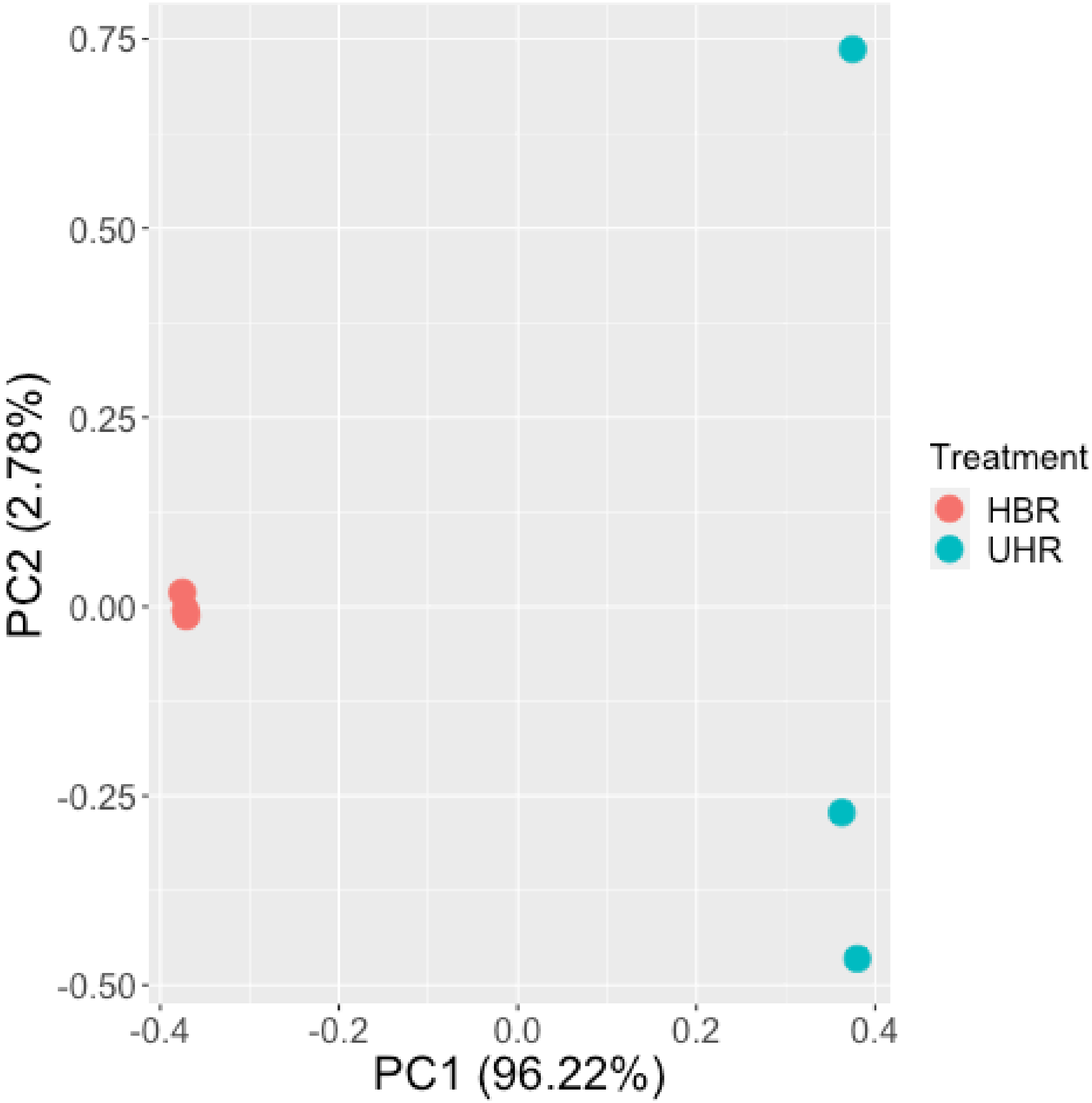
```
%%R
# Import gene expression data table
counts2 <- read.csv("/Users/wuz8/onedrive/GAU/BTEP/BTEP_Coding_Club/data/
  ↪S2_JUPYTER_NOTEBOOK/hbr_uhr_normalized_counts.csv", row.names=1)

# Construct the heatmap
pheatmap(counts2, scale='row')
```



[22]:
```
%%R
# Activate more packages
library(ggplot2)
library(ggfortify)
```

[23]:
```
%%R
# Import gene expression data
counts3 <- read.csv("/Users/wuz8/onedrive/GAU/BTEP/BTEP_Coding_Club/data/
↪S2_JUPYTER_NOTEBOOK/hbr_uhr_normalized_counts_pca.csv")

# Run PCA
hbr_uhr_pca <- prcomp(counts3[3:14],scale.=TRUE,center=TRUE)
```

```
# Plot PCA
autoplot(hbr_uhr_pca,data=counts3,colour="Treatment",size=5)+
    theme(axis.title=element_text(size=20),
          axis.text=element_text(size=15),
          legend.title=element_text(size=15),
          legend.text=element_text(size=15))
```



[24]:

```
%%R
sessionInfo()
```

```
R version 4.2.3 (2023-03-15)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: macOS Ventura 13.3.1
```

```
Matrix products: default
LAPACK: /Users/wuz8/miniconda3/lib/libopenblasp-r0.3.21.dylib

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] tools     stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
[1] ggfortify_0.4.16 ggplot2_3.4.1    pheatmap_1.0.12

loaded via a namespace (and not attached):
 [1] magrittr_2.0.3     tidyselect_1.2.0   munsell_0.5.0      colorspace_2.1-0
 [5] R6_2.5.1           rlang_1.1.0        fansi_1.0.4        stringr_1.5.0
 [9] dplyr_1.1.0        grid_4.2.3         gtable_0.3.1       utf8_1.2.3
[13] cli_3.6.0          withr_2.5.0        tibble_3.2.0       lifecycle_1.0.3
[17] gridExtra_2.3      farver_2.1.1       purrr_1.0.1        RColorBrewer_1.1-3
[21] tidyr_1.3.0        vctrs_0.6.0        glue_1.6.2         labeling_0.4.2
[25] stringi_1.7.12     compiler_4.2.3     pillar_1.8.1       generics_0.1.3
[29] scales_1.2.1       pkgconfig_2.0.3
```

[ ]: