Data Visualization

Lesson 8 - 8/13/19

(Please sign-in on the counter near the back door)

Penn
University of Pennsylvania

Sponsored by:

Institute for Biomedical Informatics

BGSA

GCB

Award Ceremony

Exercise 7 Review

How to measure run time?

- See “Extra Problem: `time` practice” in lab6
- At the top of your jupyter cell you can type **%%timeit** which runs the functions in the cell multiple times, and print the average and standard deviation of the runtimes it obtained

```
%%timeit

list_x = list(range(0,50000)) #50000 numbers
series_x = pd.Series(list_x)

series_squared = series_x.apply(lambda x: x**2)
```

```
26.8 ms ± 394 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

```
%%timeit

list_x = list(range(0,50000)) #50000 numbers
series_x = pd.Series(list_x)

list_squared=[]
for val in series_x:
    list_squared.append(val**2)
```

```
22 ms ± 299 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

Using the pandas series apply function is marginally faster than looping right the series.

Review of Python Bootcamp

- In this class, you may have learned about:
  - Print statements
  - Data Types: Strings, Ints, Floats, Booleans, etc.
  - Conditionals (If/Else if/Else)
  - Loops (For/While)
  - Custom Functions
  - Lists, Dictionaries, Tuples
  - Reading and Writing Files lines by line
  - Writing scripts and importing libraries
  - Importing, exporting, slicing, joining, and grouping pandas series and dataframes

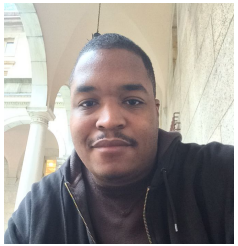
Review of Python Bootcamp

- What we hope you take away from this class is
  - Confidence and encouragement
  - A desire to write “clean” code that is well-commented and easy-to-understand variable names
  - An interest to learn more and make your code more efficient
  - **An ability to solve some more complex problems**

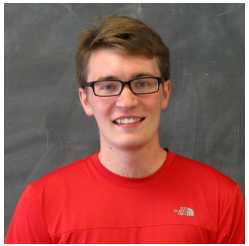
Today's TAs

Ben

Sammy

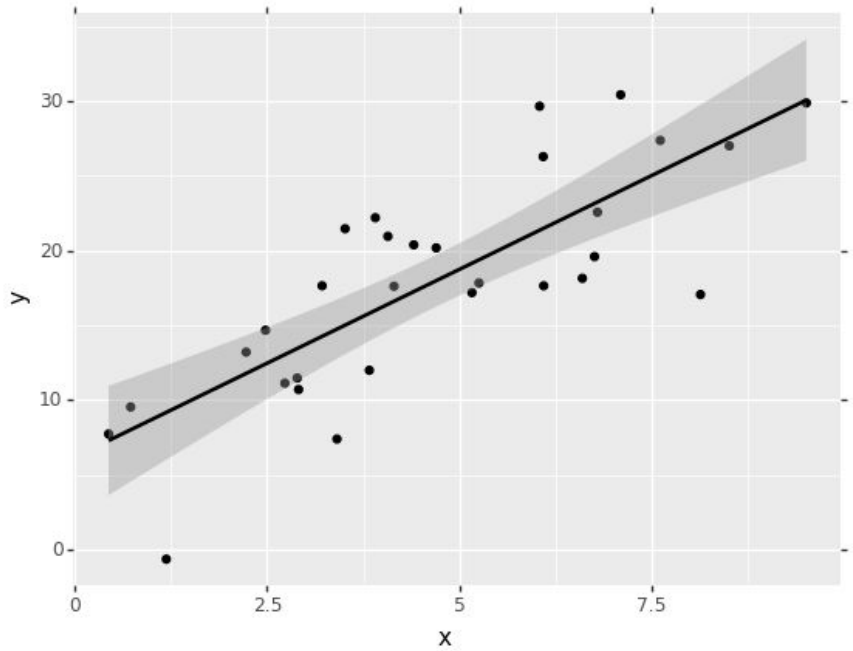


David



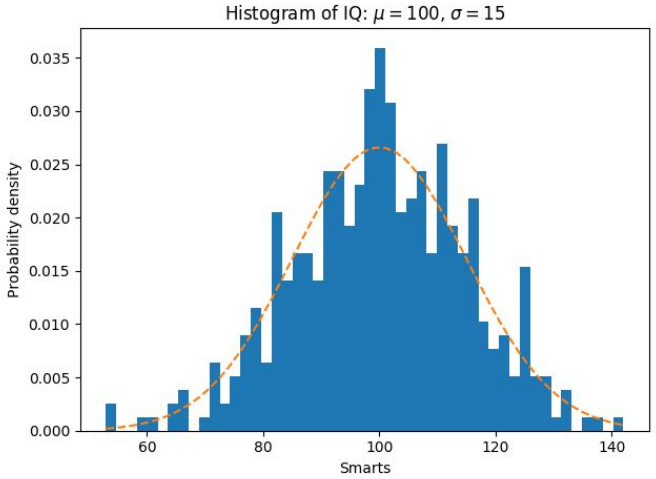
Louis

Why Data Visualization?

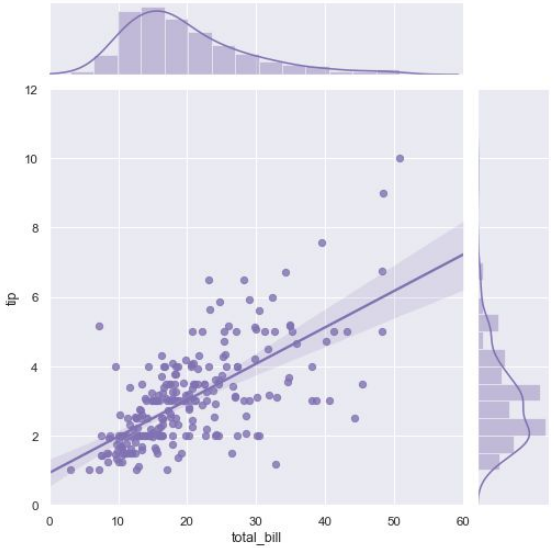


Visualization Options

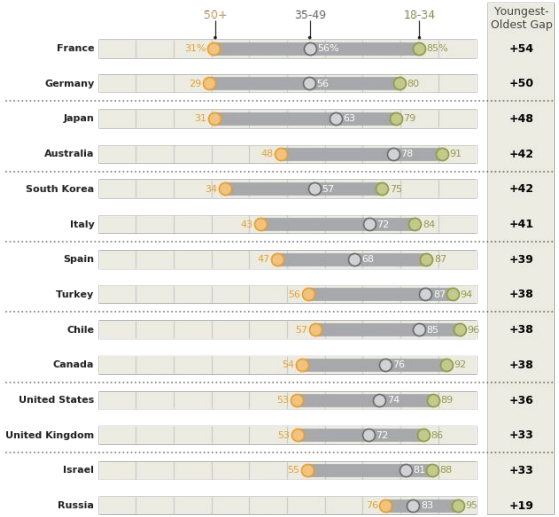
Matplotlib



Seaborn

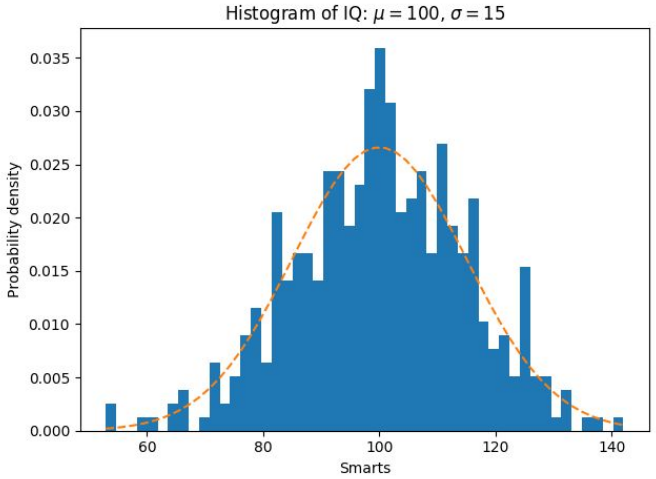


Plotnine

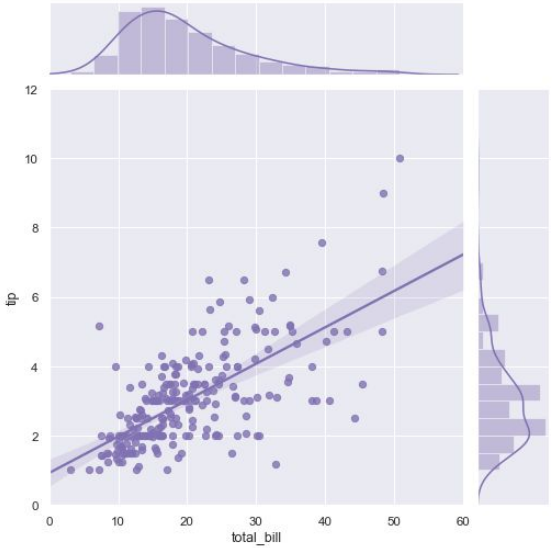


Visualization Options

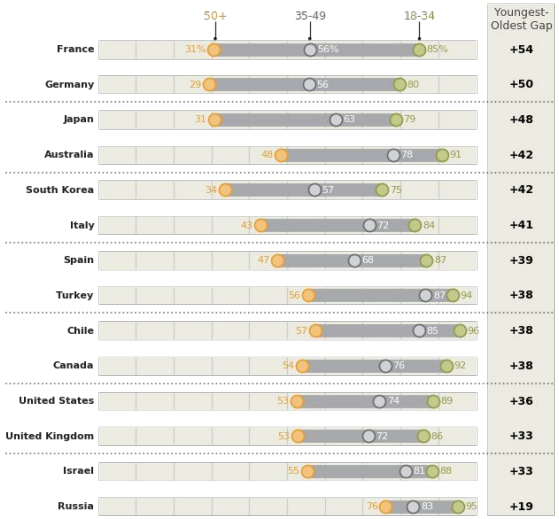
Matplotlib



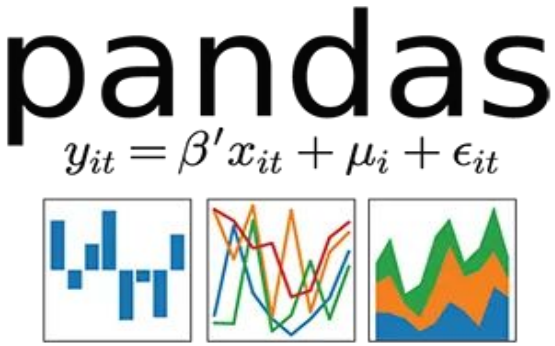
Seaborn



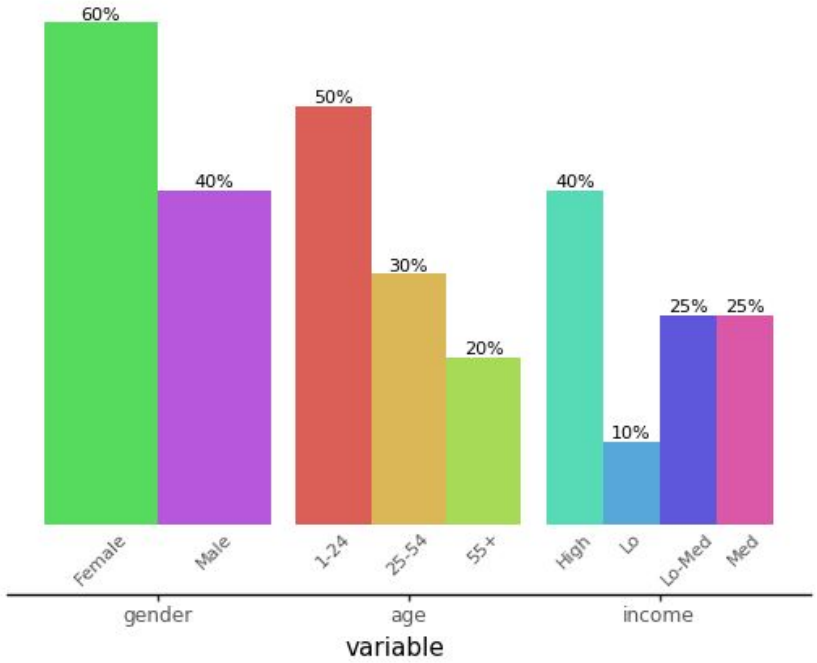
Plotnine



Connection between Pandas, R, and Plotnine



What do you need to make a plot?



1. Data

1. Data
2. Geometric Objects

1. Data
2. Geometric Objects
3. Color

1. Data
2. Geometric Objects
3. Color
4. Annotations

1. Data
2. Geometric Objects
3. Color
4. Annotations

Setup

In [34]:

```
import math
from plotnine import *

import pandas
import numpy as np

import warnings
warnings.filterwarnings('ignore')
```

Setup

In [34]:
```
import math
from plotnine import *

import pandas
import numpy as np

import warnings
warnings.filterwarnings('ignore')
```

Read in data from Zenodo

In [35]:
```
dataframe = pandas.read_csv('volcano_data.tsv', sep='\t')
dataframe['neg_log_p_val'] = -np.log(dataframe['adj.P.Val'])
```

Setup

In [34]:
```
import math
from plotnine import *

import pandas
import numpy as np

import warnings
warnings.filterwarnings('ignore')
```

Read in data from Zenodo

In [35]:
```
dataframe = pandas.read_csv('volcano_data.tsv', sep='\t')
dataframe['neg_log_p_val'] = -np.log(dataframe['adj.P.Val'])
```

In [36]:
```
dataframe.head()
```

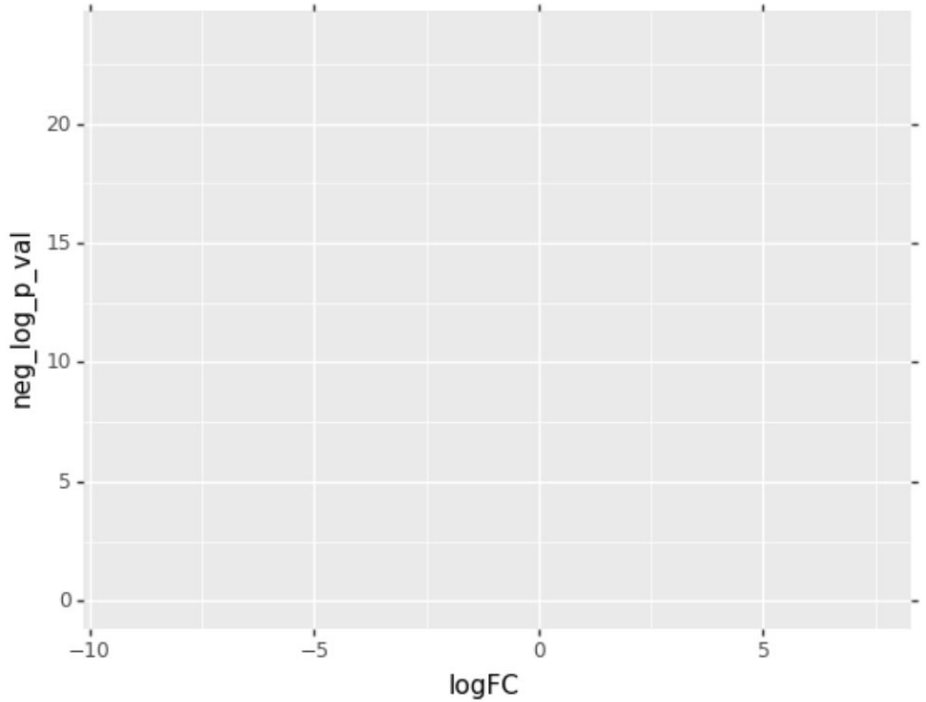
Out[36]:

	ENTREZID	SYMBOL	GENENAME	logFC	AveExpr	t	P.Value	adj.P.Val	neg_log_p_val
0	12992	Csn1s2b	casein alpha s2-like B	-8.603611	3.562950	-43.796498	3.830650e-15	6.053959e-11	23.527724
1	13358	Slc25a1	solute carrier family 25 (mitochondrial carrie...	-4.124175	5.779699	-29.907849	1.758595e-13	1.389642e-09	20.394220
2	11941	Atp2b2	ATPase, Ca++ transporting, plasma membrane 2	-7.386986	1.282143	-27.819499	4.836363e-13	2.432800e-09	19.834223
3	20531	Slc34a2	solute carrier family 34 (sodium phosphate), m...	-4.177812	4.278629	-27.072723	6.157428e-13	2.432800e-09	19.834223
4	100705	Acacb	acetyl-Coenzyme A carboxylase beta	-4.314320	4.440914	-25.223566	1.499977e-12	4.741129e-09	19.166991

Data from zenodo.org

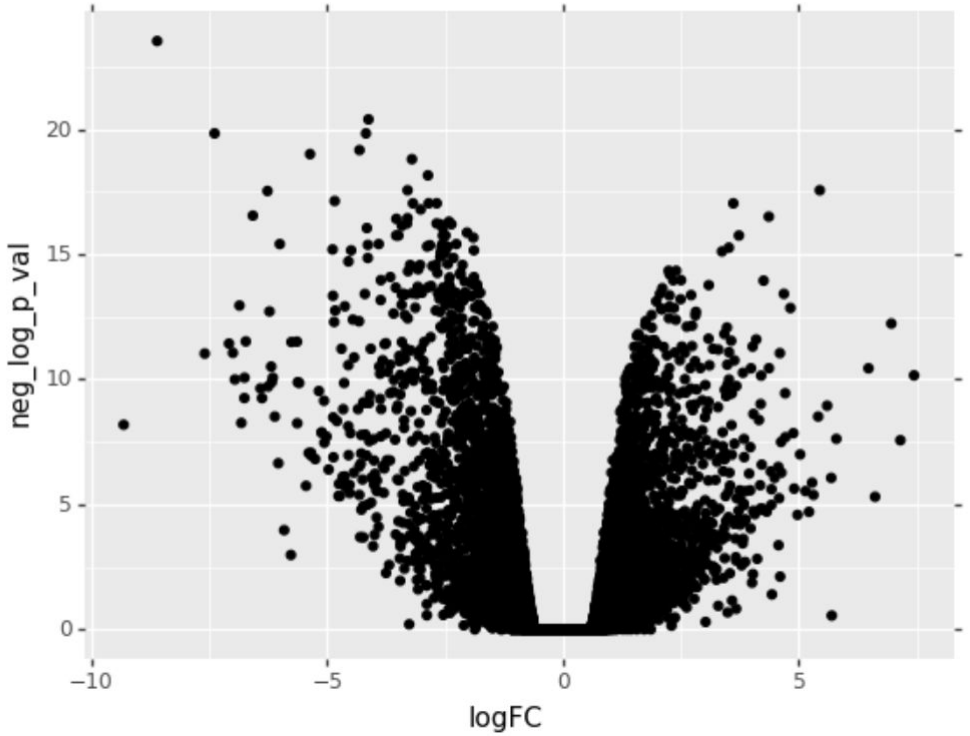
Data

```
ggplot(dataframe, aes(x='logFC', y='neg_log_p_val'))
```

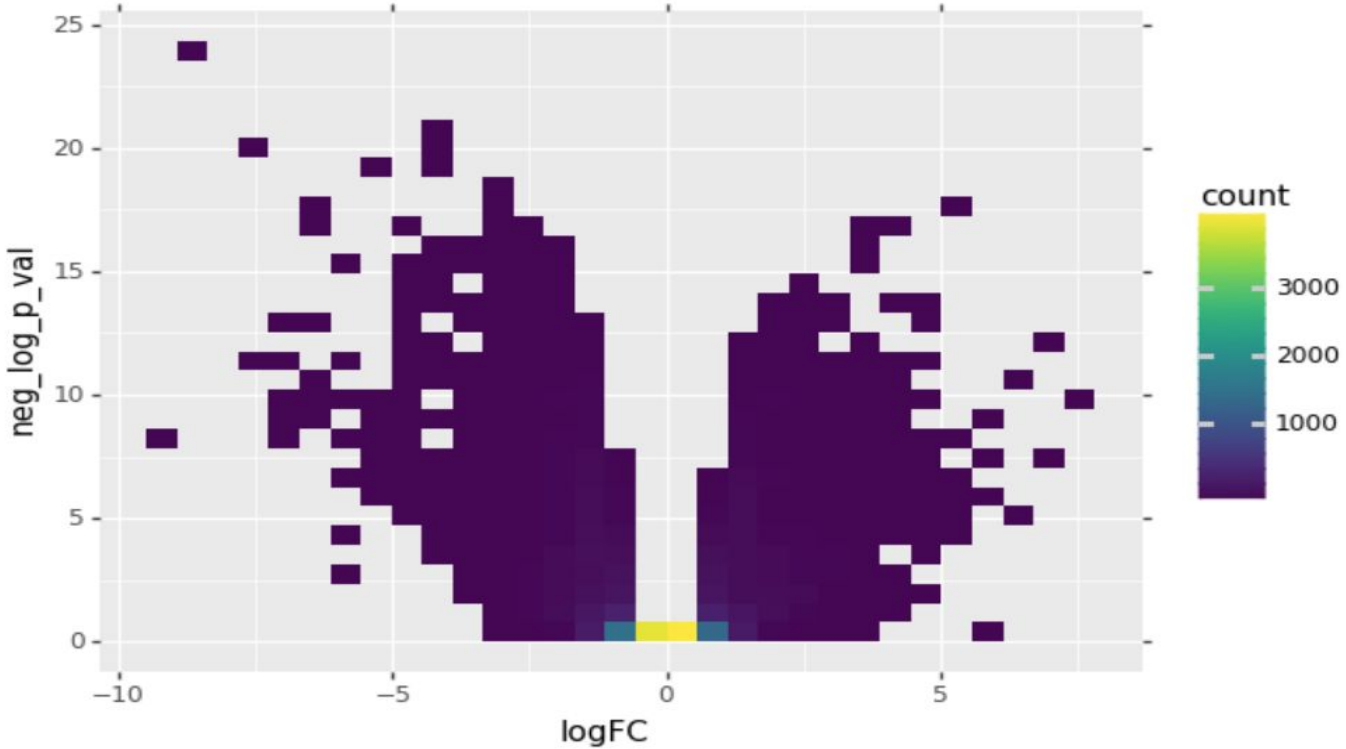


1. Data
2. Geometric Objects
3. Color
4. Annotations

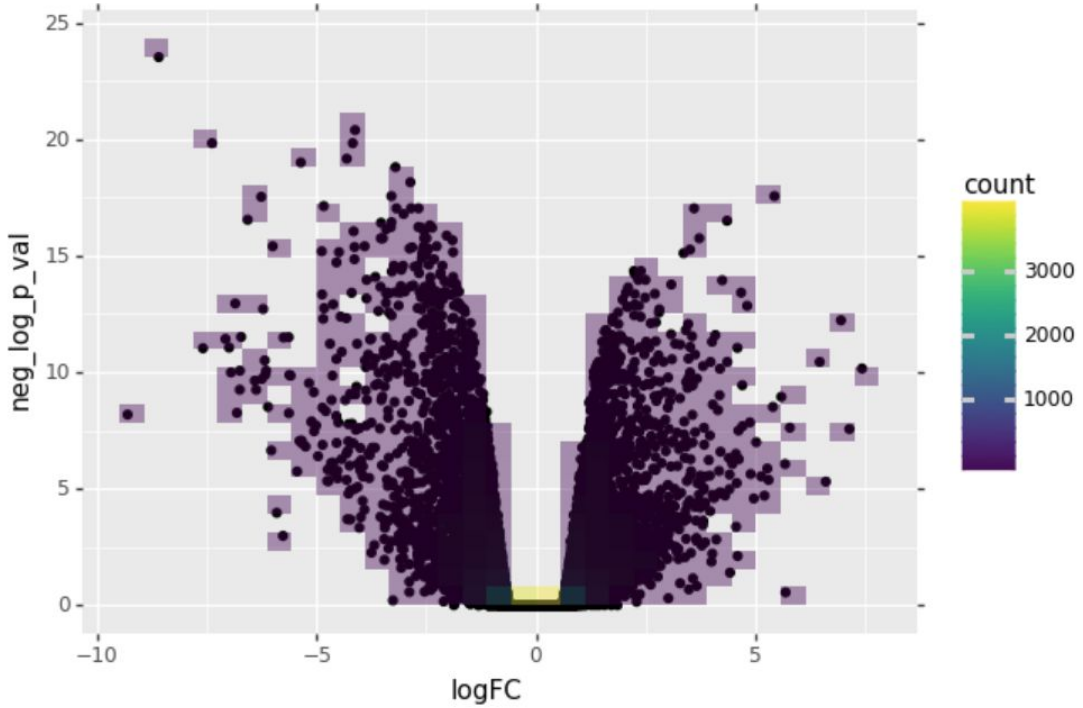
```
ggplot(dataframe, aes(x='logFC', y='neg_log_p_val')) + geom_point()
```



```
ggplot(dataframe, aes(x='logFC', y='neg_log_p_val')) + geom_bin2d()
```

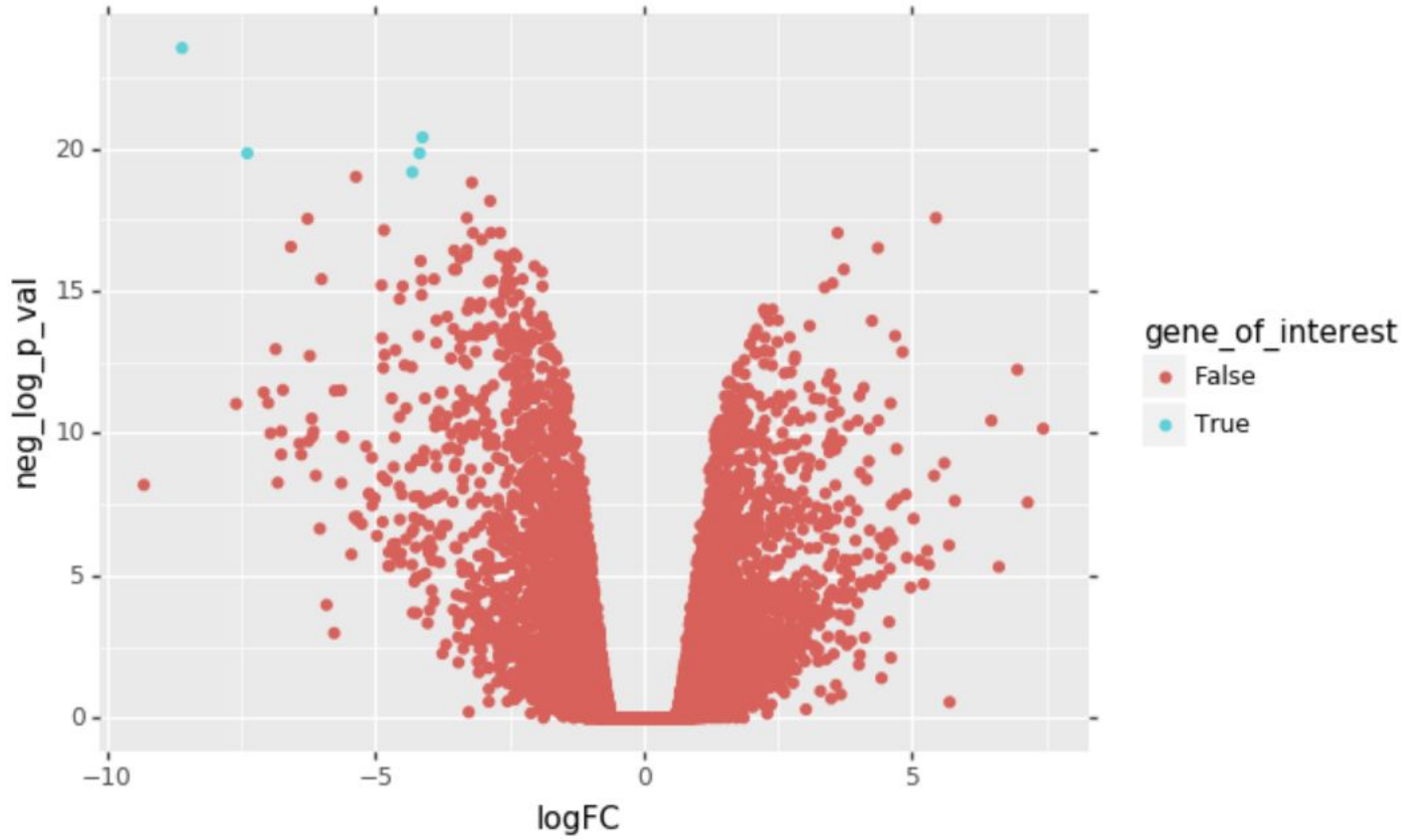


```
ggplot(dataframe, aes(x='logFC', y='neg_log_p_val'))
+ geom_point()
+ geom_bin2d(alpha=.4)
```

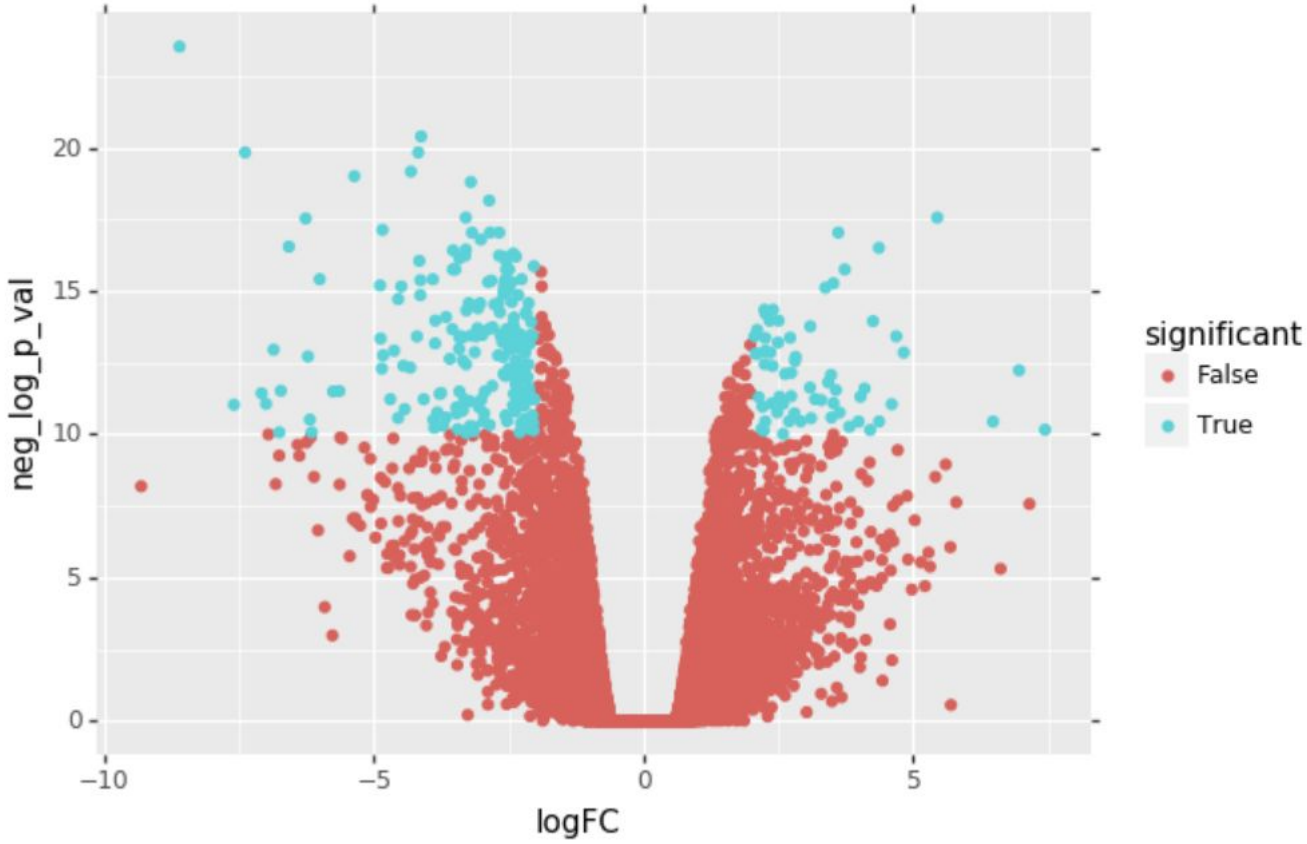


1. Data
2. Geometric Objects
3. Color
4. Annotations

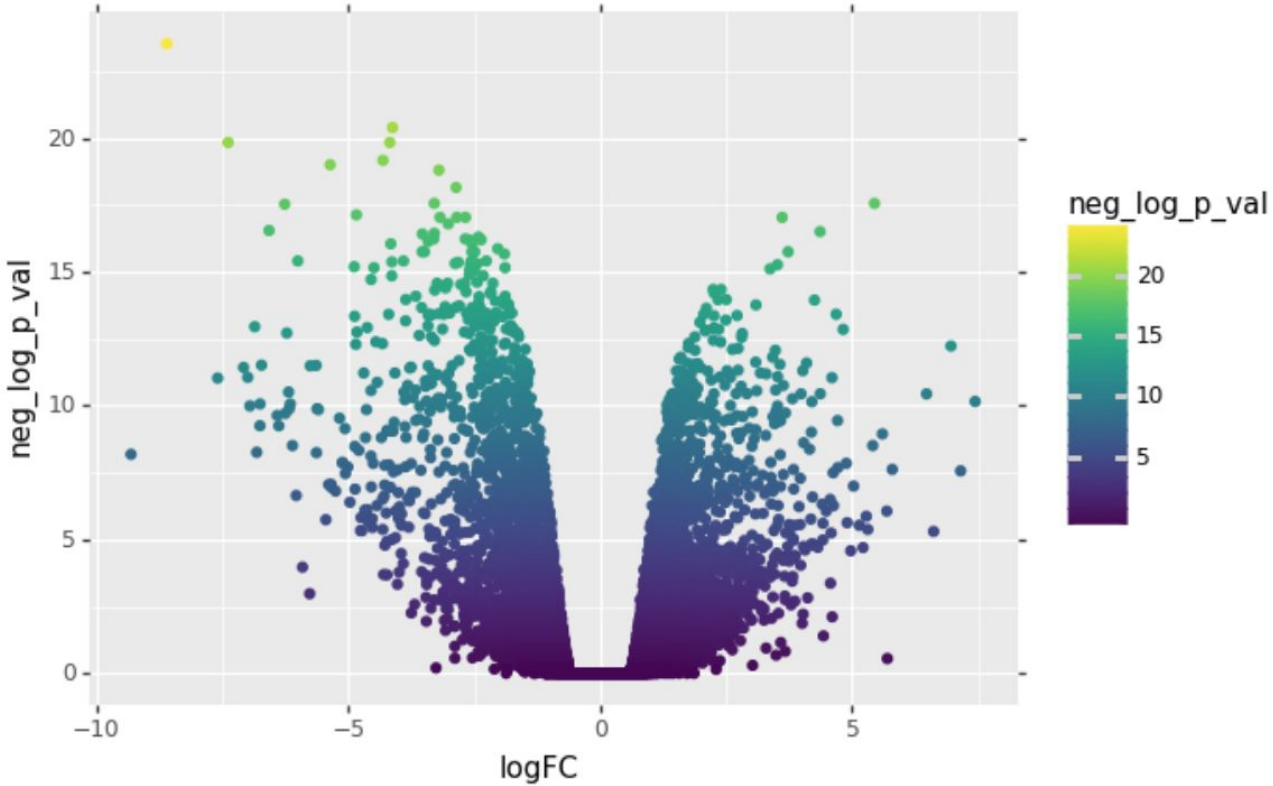
```
genes_of_interest = ['Csn1s2b', 'Slc25a1', 'Atp2b2', 'Slc34a2', 'Acacb']

dataframe['gene_of_interest'] = dataframe['SYMBOL'].isin(genes_of_interest)

ggplot(dataframe, aes(x='logFC', y='neg_log_p_val')) + geom_point(aes(color='gene_of_interest'))
```



In [47]:
```
condition1 = dataframe['neg_log_p_val'] >= 10
condition2 = abs(dataframe['logFC']) > 2
dataframe['significant'] = condition1 & condition2

ggplot(dataframe, aes(x='logFC', y='neg_log_p_val')) + geom_point(aes(color='significant'))
```

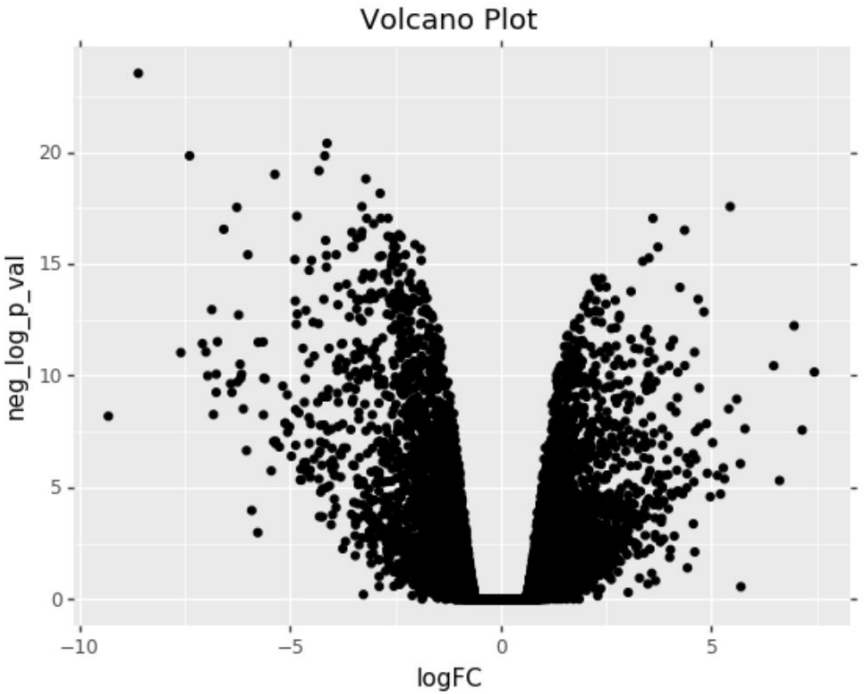


```
ggplot(dataframe, aes(x='logFC', y='neg_log_p_val'))
+ geom_point(aes(color='neg_log_p_val'))
```

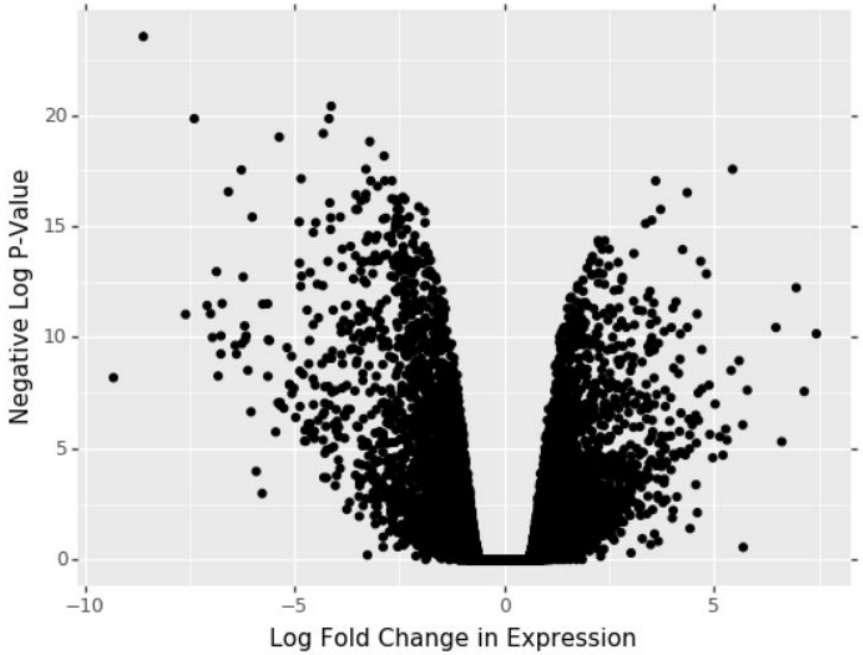


1. Data
2. Geometric Objects
3. Color
4. Annotations

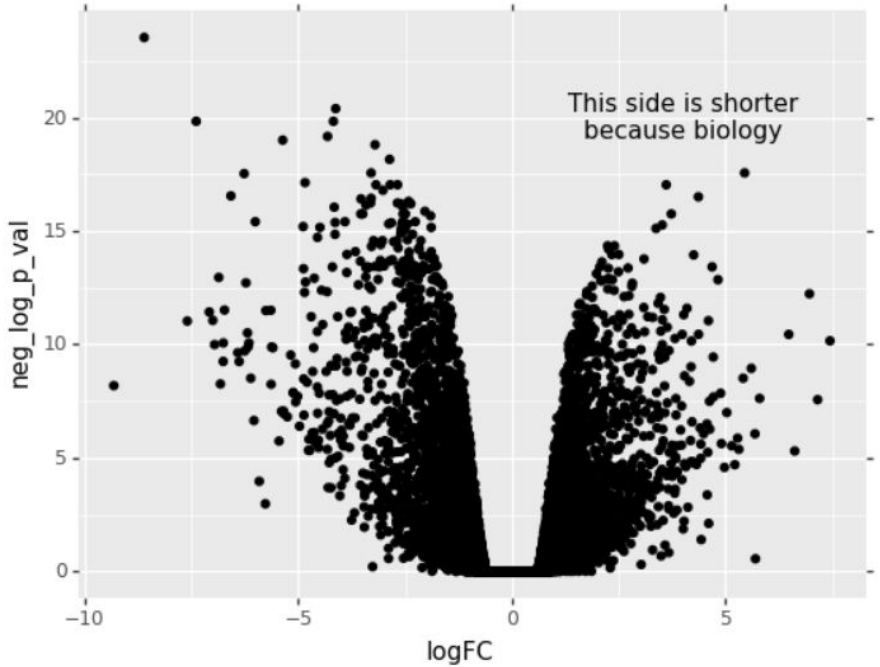
```
ggplot(dataframe, aes(x='logFC', y='neg_log_p_val')) \
+ geom_point()\
+ ggtitle('Volcano Plot')
```



```
ggplot(dataframe, aes(x='logFC', y='neg_log_p_val')) \
+ geom_point()\
+ xlab('Log Fold Change in Expression')\
+ ylab('Negative Log P-Value')
```



```
ggplot(dataframe, aes(x='logFC', y='neg_log_p_val', label='SYMBOL')) \
+ geom_point()\
+ annotate("text", x = 4, y = 20, label = "This side is shorter\nbecause biology")
```



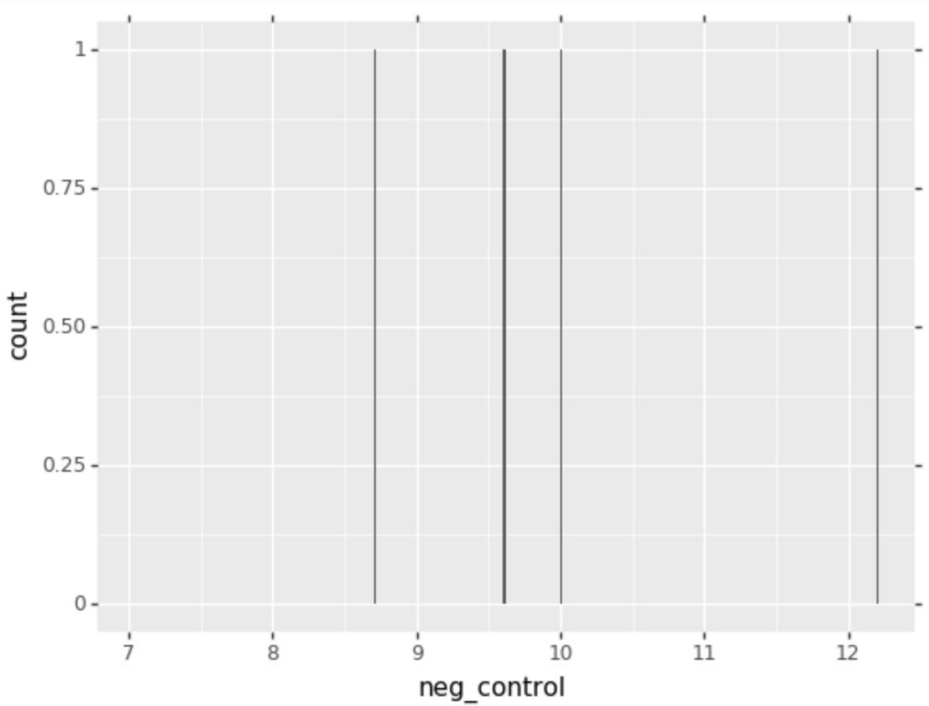
Common Use Cases: Bar Plot (Setup)

```
neg_control = np.random.normal(size=40) + 10
pos_control = np.random.normal(size=40) + 15
experiment = np.random.normal(size=40) + 14


data = {'neg_control': neg_control, 'pos_control': pos_control, 'experiment': experiment}
bar_df = pandas.DataFrame(data)
bar_df
```

	neg_control	pos_control	experiment
0	10.573405	15.795453	12.646862
1	10.266933	14.978045	12.372353
2	8.398704	14.365767	12.897853
3	9.794511	15.033012	14.688481

Bar Plot Attempt One

```
ggplot(bar_df, aes(x='neg_control')) + geom_bar()
```



Reshape Data

```
bar_df = pandas.melt(bar_df)
bar_df.head()
```

	neg_control	pos_control
0	10.573405	15.795453
1	10.266933	14.978045
2	8.398704	14.365767
3	9.794511	15.033012
4	8.888123	15.898007

Reshape Data

Convert data from columns with values to variable:value pairs

```
bar_df = pandas.melt(bar_df)
bar_df.head()
```

	neg_control	pos_control
0	10.573405	15.795453
1	10.266933	14.978045
2	8.398704	14.365767
3	9.794511	15.033012
4	8.888123	15.898007

Reshape Data

Convert data from columns with values to variable:value pairs

```
bar_df = pandas.melt(bar_df)
bar_df.head()
```

	neg_control	pos_control
0	10.573405	15.795453
1	10.266933	14.978045
2	8.398704	14.365767
3	9.794511	15.033012
4	8.888123	15.898007

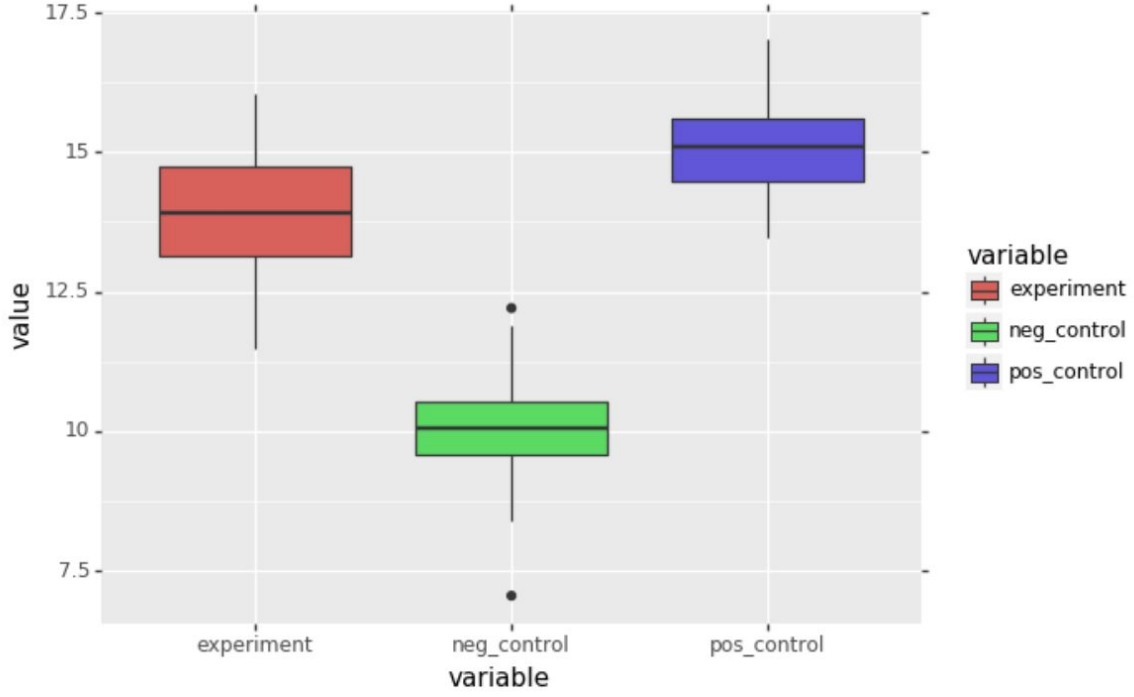
	variable	value
0	neg_control	10.573405
1	neg_control	10.266933
2	neg_control	8.398704
3	neg_control	9.794511
4	neg_control	8.888123

Box Plot Attempt 2

```
(ggplot(bar_df, aes(x='variable', y='value', fill='variable'))
+ geom_boxplot())
```

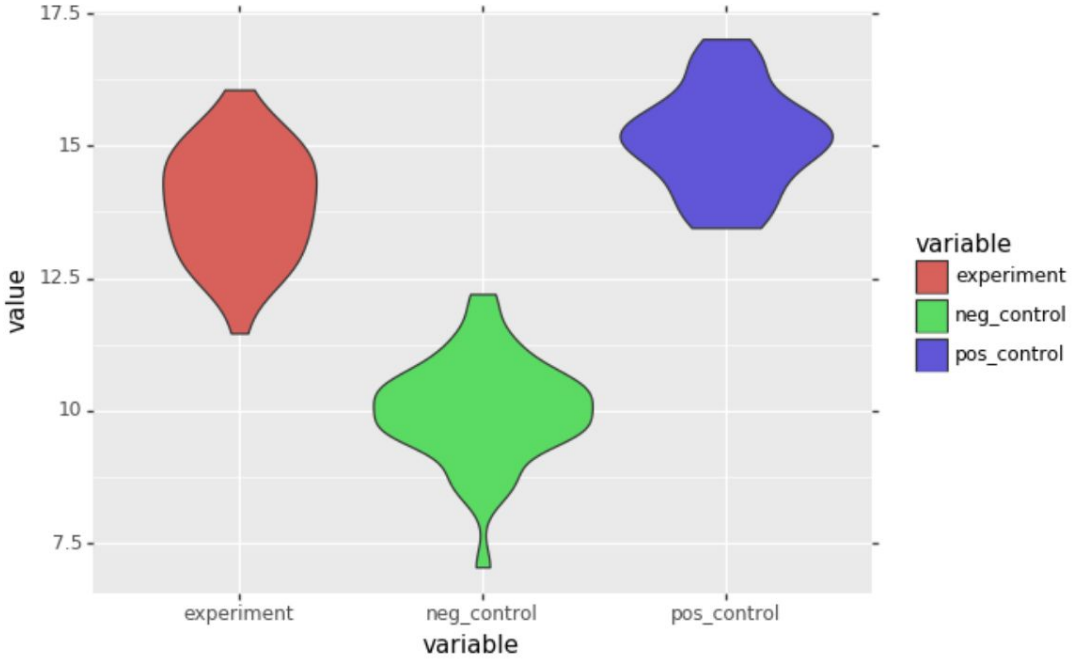
Box Plot Attempt 2

```
(ggplot(bar_df, aes(x='variable', y='value', fill='variable'))
+ geom_boxplot())
```



Bonus Box Plot

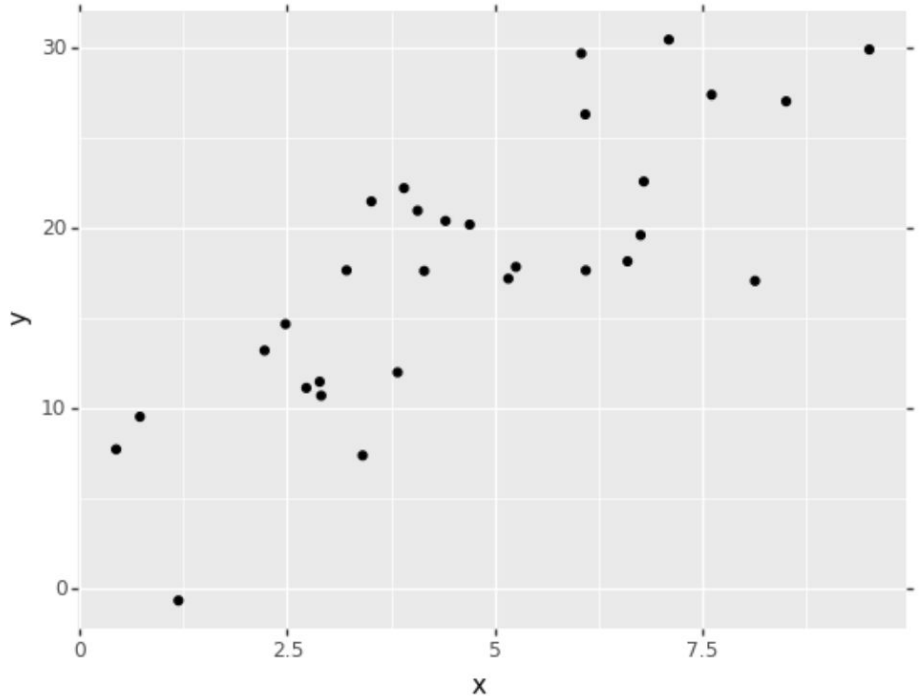
```
ggplot(bar_df, aes(x='variable', y='value', fill='variable'))\
+ geom_violin()
```



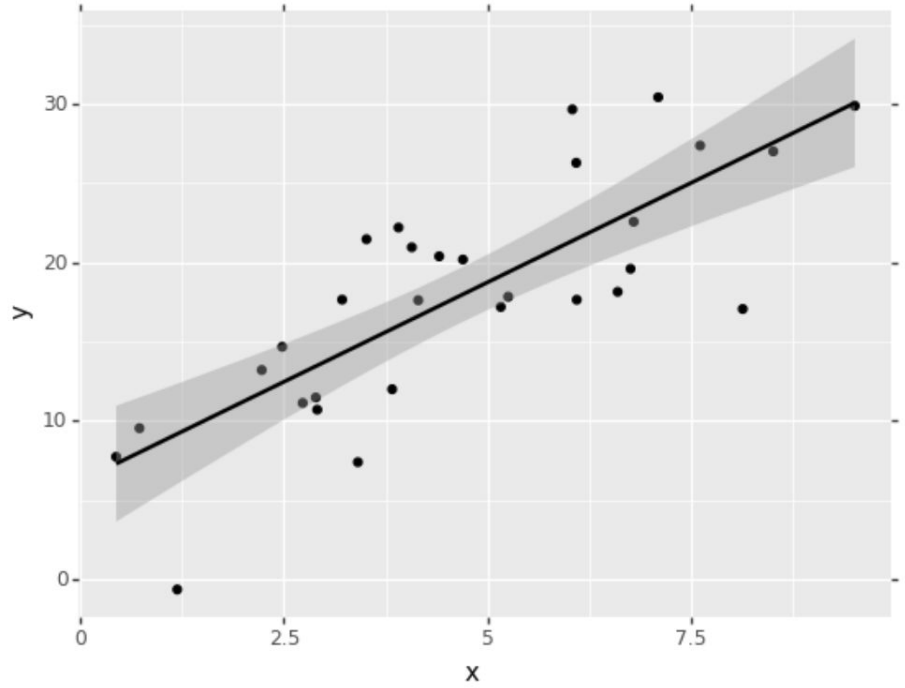
Common Use Cases: Scatter Plot

```
# Set the x range to be [0, 10)
scatter_x = np.random.random(30) * 10
# Give the line a slope of 3 and a y-intercept of 4
scatter_y = scatter_x * 3 + 4
# Add noise
scatter_y += np.random.normal(size=30) * 4

scatter_df = pandas.DataFrame({'x': scatter_x, 'y': scatter_y})
scatter_df.head()
```

```
ggplot(scatter_df, aes(x='x', y='y')) \
+ geom_point()
```



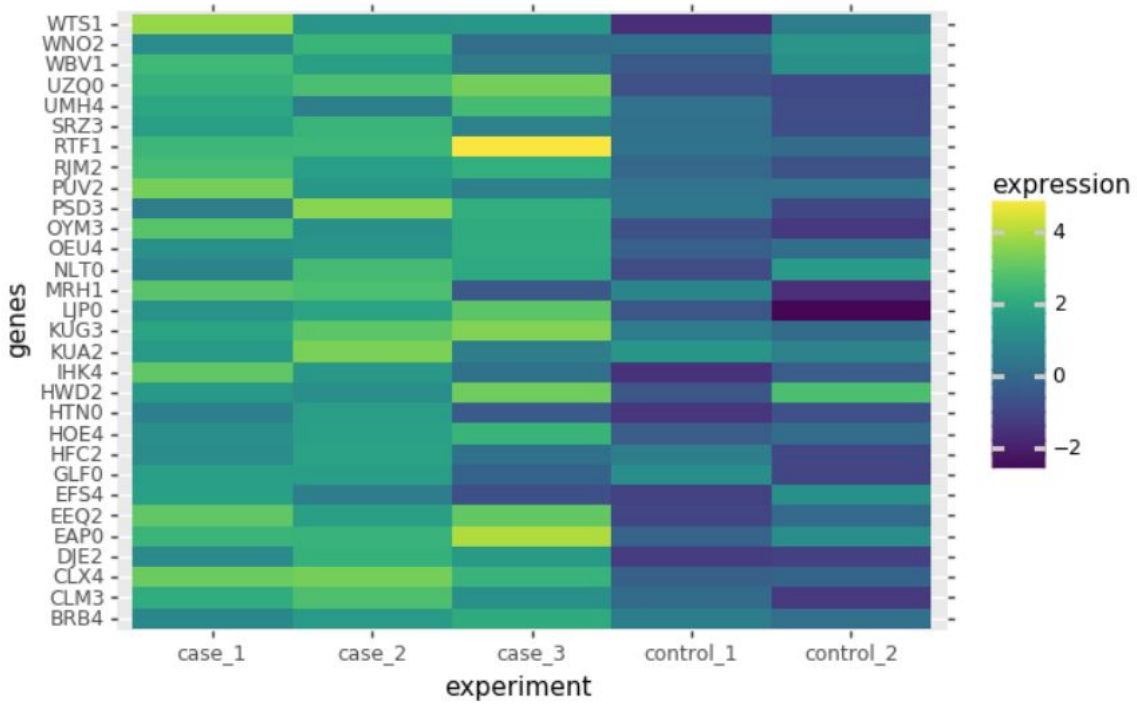
```
ggplot(scatter_df, aes(x='x', y='y'))\
+ geom_point()\
+ geom_smooth(method='lm')
```



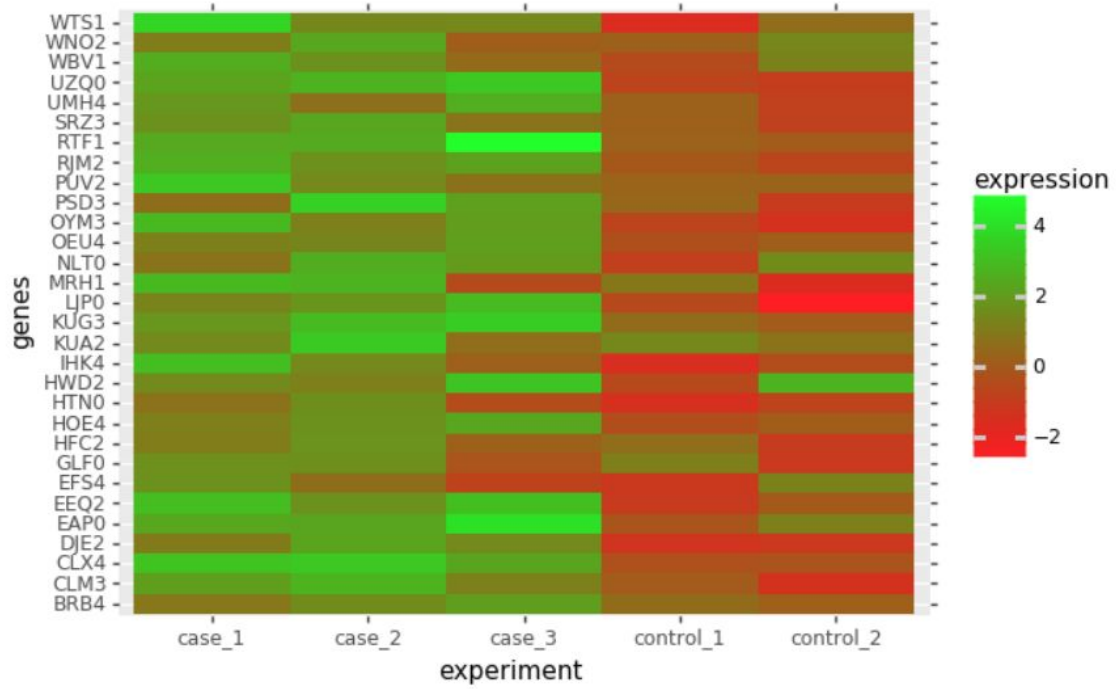
Common Use Cases: Heatmap

	genes	experiment	expression
0	WNO2	case_1	1.050926
1	NLT0	case_1	0.826808
2	RJM2	case_1	2.507592
3	EEQ2	case_1	2.911344
4	GLF0	case_1	1.611909

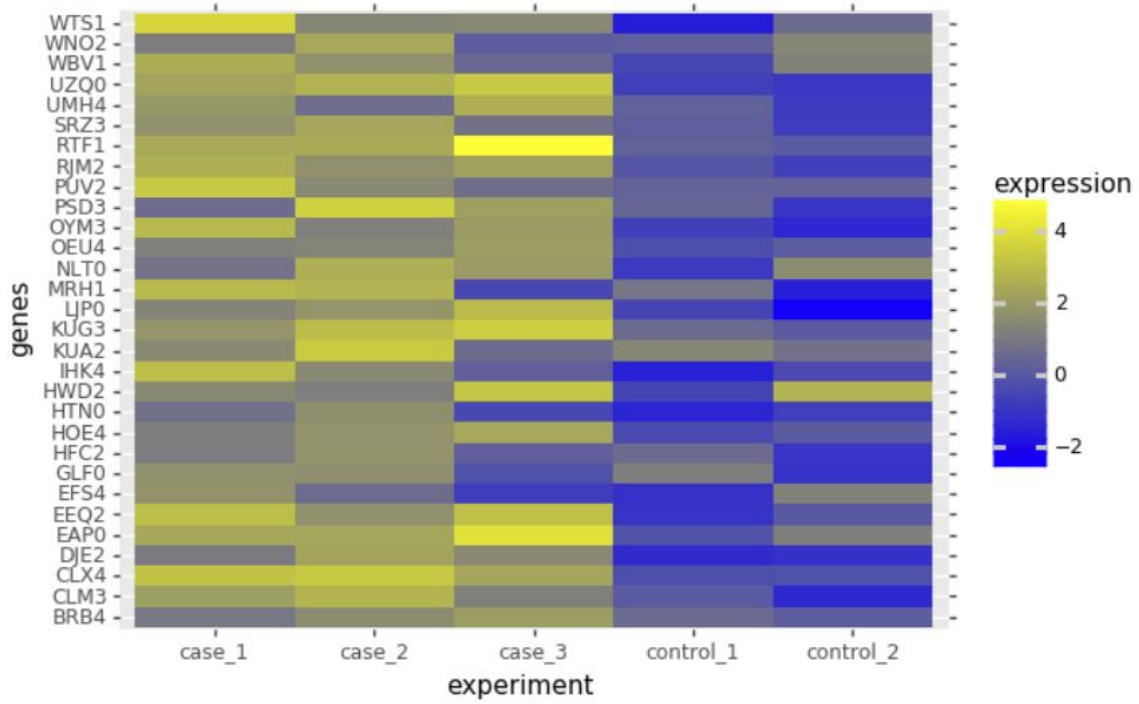
```
ggplot(heatmap_df, aes(x='experiment', y='genes'))\
+ geom_tile(aes(fill='expression'))\
```



```
ggplot(heatmap_df, aes(x='experiment', y='genes'))\
+ geom_tile(aes(fill='expression'))\
+ scale_fill_gradient(low='red', high='lime')
```



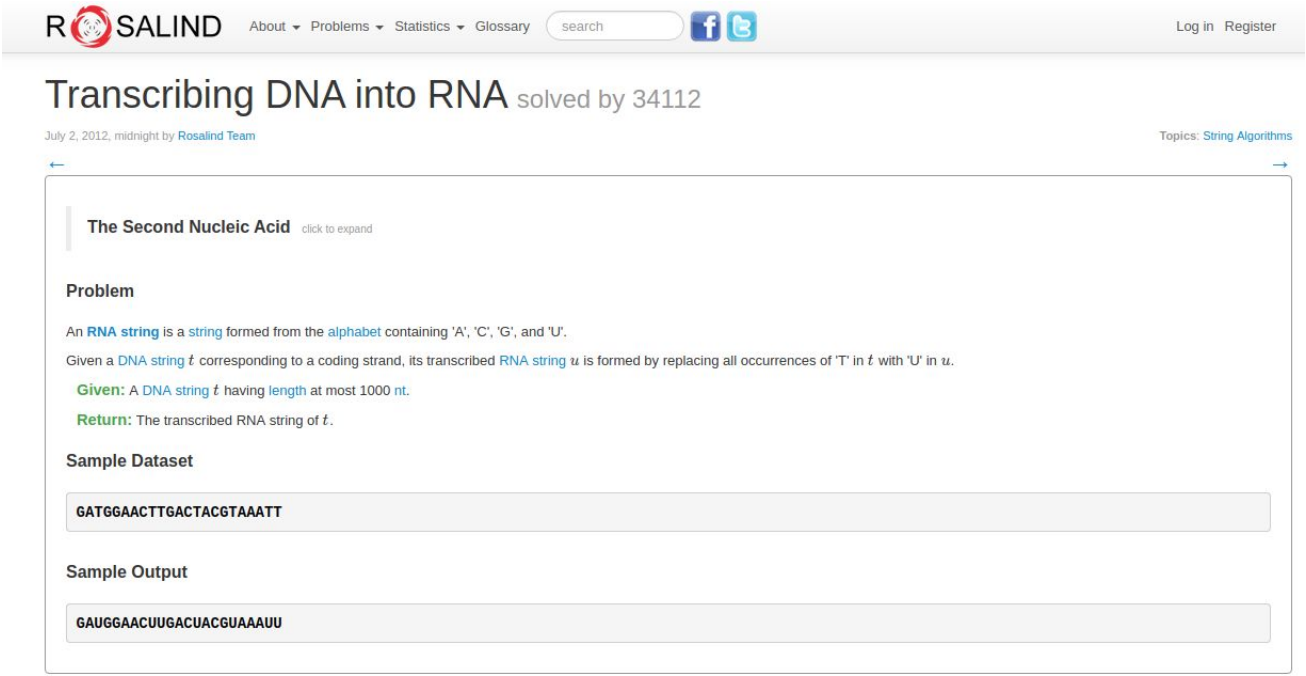
```
ggplot(heatmap_df, aes(x='experiment', y='genes'))\
+ geom_tile(aes(fill='expression'))\
+ scale_fill_gradient(low='blue', high='yellow')
```



Next Steps/Further Training

Rosalind

Codecademy



Next Steps/Further Training

Rosalind

Codecademy

Don’t get discouraged!

Further Reading

http://vita.had.co.nz/papers/layered-grammar.pdf