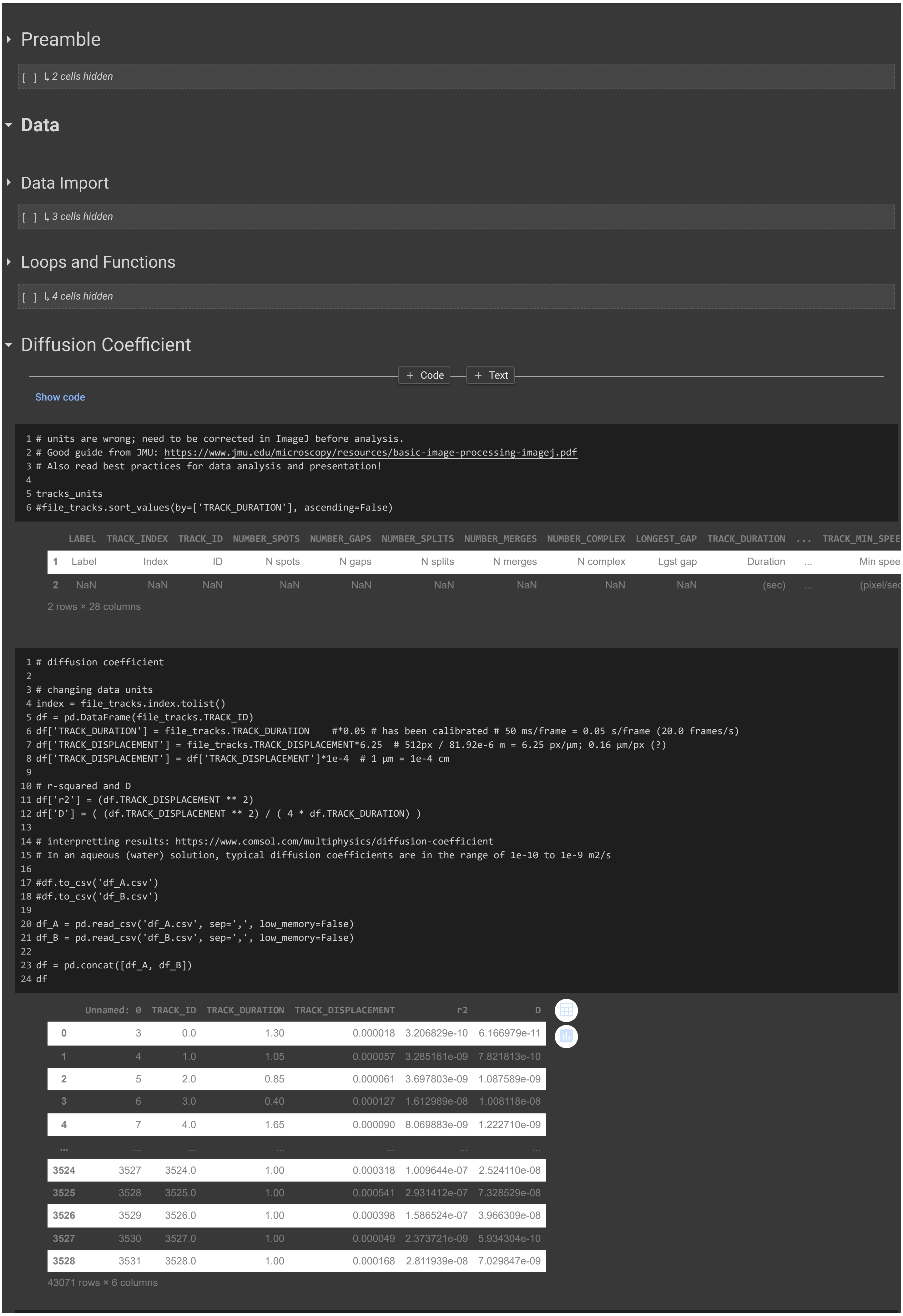


Preamble

[ ] ↳ 2 cells hidden

Data

Data Import

[ ] ↳ 3 cells hidden

Loops and Functions

[ ] ↳ 4 cells hidden

Diffusion Coefficient

```
# units are wrong; need to be corrected in ImageJ before analysis.
# Good guide from JMU: https://www.jmu.edu/microscopy/resources/basic-image-processing-imagej.pdf
# Also read best practices for data analysis and presentation!

tracks_units
#file_tracks.sort_values(by=['TRACK_DURATION'], ascending=False)
```

	LABEL	TRACK_INDEX	TRACK_ID	NUMBER_SPOTS	NUMBER_GAPS	NUMBER_SPLITS	NUMBER_MERGES	NUMBER_COMPLEX	LONGEST_GAP	TRACK_DURATION	...	TRACK_MIN_SPEE
1	Label	Index	ID	N spots	N gaps	N splits	N merges	N complex	Lgst gap	Duration	...	Min spee
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	(sec)	...	(pixel/sec

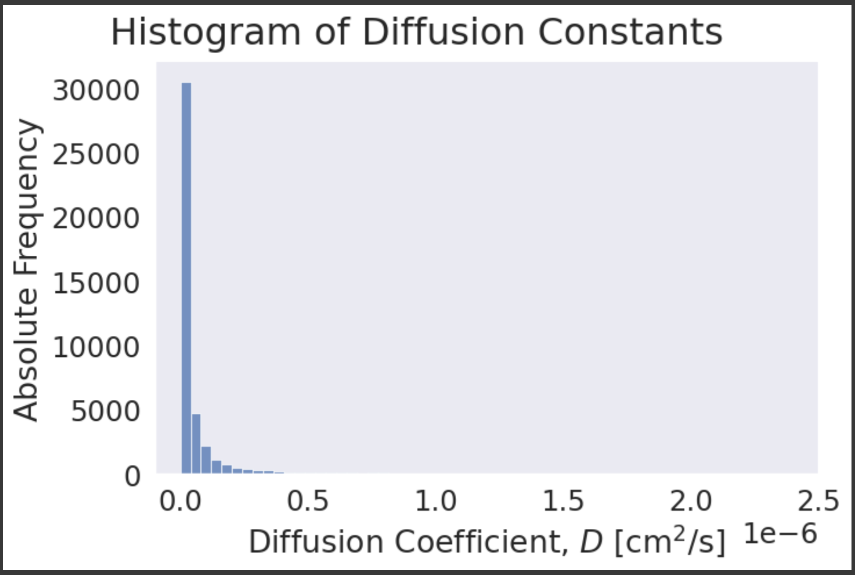
2 rows × 28 columns

```
# diffusion coefficient

# changing data units
index = file_tracks.index.tolist()
df = pd.DataFrame(file_tracks.TRACK_ID)
df['TRACK_DURATION'] = file_tracks.TRACK_DURATION    #*0.05 # has been calibrated # 50 ms/frame = 0.05 s/frame (20.0 frames/s)
df['TRACK_DISPLACEMENT'] = file_tracks.TRACK_DISPLACEMENT*6.25  # 512px / 81.92e-6 m = 6.25 px/µm; 0.16 µm/px (?)
df['TRACK_DISPLACEMENT'] = df['TRACK_DISPLACEMENT']*1e-4  # 1 µm = 1e-4 cm

# r-squared and D
df['r2'] = (df.TRACK_DISPLACEMENT ** 2)
df['D'] = ( (df.TRACK_DISPLACEMENT ** 2) / ( 4 * df.TRACK_DURATION) )

# interpretting results: https://www.comsol.com/multiphysics/diffusion-coefficient
# In an aqueous (water) solution, typical diffusion coefficients are in the range of 1e-10 to 1e-9 m2/s

#df.to_csv('df_A.csv')
#df.to_csv('df_B.csv')

df_A = pd.read_csv('df_A.csv', sep=',', low_memory=False)
df_B = pd.read_csv('df_B.csv', sep=',', low_memory=False)

df = pd.concat([df_A, df_B])
df
```

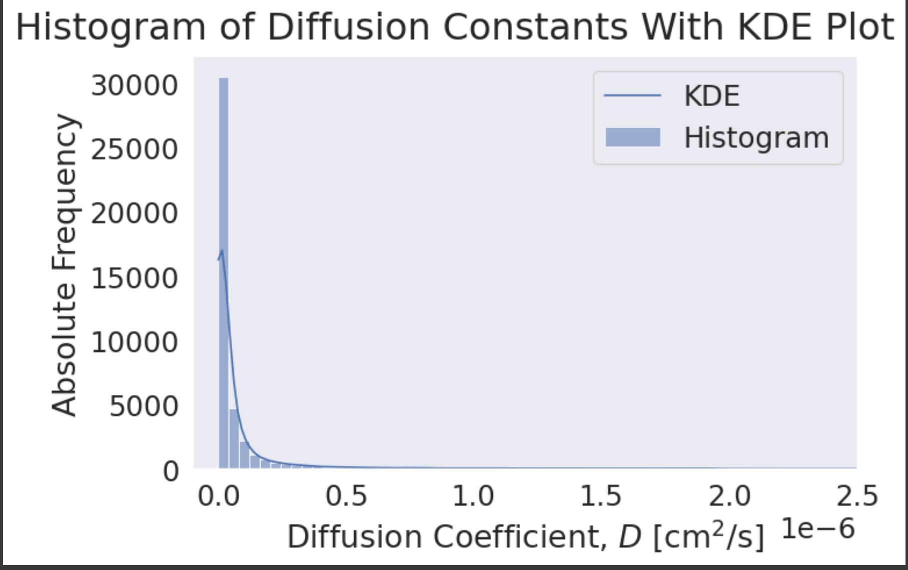
	Unnamed: 0	TRACK_ID	TRACK_DURATION	TRACK_DISPLACEMENT	r2	D
0	3	0.0	1.30	0.000018	3.206829e-10	6.166979e-11
1	4	1.0	1.05	0.000057	3.285161e-09	7.821813e-10
2	5	2.0	0.85	0.000061	3.697803e-09	1.087589e-09
3	6	3.0	0.40	0.000127	1.612989e-08	1.008118e-08
4	7	4.0	1.65	0.000090	8.069883e-09	1.222710e-09
...	...	...	...	...	...	...
3524	3527	3524.0	1.00	0.000318	1.009644e-07	2.524110e-08
3525	3528	3525.0	1.00	0.000541	2.931412e-07	7.328529e-08
3526	3529	3526.0	1.00	0.000398	1.586524e-07	3.966309e-08
3527	3530	3527.0	1.00	0.000049	2.373721e-09	5.934304e-10
3528	3531	3528.0	1.00	0.000168	2.811939e-08	7.029847e-09

43071 rows × 6 columns

```python
# histogram using Seaborn + matplotlib

plot = sns.displot(data=df, x="D", kind="hist", kde=False, bins = 75, aspect = 1.5, legend=True)
plot.figure.subplots_adjust(top=0.9);
plt.xlim(-0.1e-6, None)
plot.figure.suptitle("Histogram of Diffusion Constants");
plot.set(xlabel=r'Diffusion Coefficient, $D$ $\left[ \mathrm{cm}{^2}/\mathrm{s}{} \right]$', ylabel='Absolute Frequency', xlim=(None, 2.5e-6));
```



```python
# histogram + kernel density estimate (KDE) plot

plot = sns.displot(data=df, x="D", kind="hist", kde=True, bins = 75, aspect = 1.5)
plot.figure.subplots_adjust(top=0.9);
plot.figure.suptitle("Histogram of Diffusion Constants With KDE Plot");
plt.xlim(-0.1e-6, None)
plot.set(xlabel=r'Diffusion Coefficient, $D$ $\left[ \mathrm{cm}{^2}/\mathrm{s}{} \right]$', ylabel='Absolute Frequency', xlim=(None, 2.5e-6));
plt.legend(labels=["KDE","Histogram"]); # kernel density estimate (KDE) plot
```



```python
# adding label to the df
df['Data_Series'] = 'pH 9 #04'

D_statistics = df.groupby(['Data_Series'])['D'].describe() #[['mean','std']]
D_statistics
```

	count	mean	std	min	25%	50%	75%	max
Data_Series								
pH 9 #04	22640.0	3.803836e-08	1.765151e-07	0.0	1.043450e-09	3.749344e-09	1.251464e-08	0.000003

Michaelis-Menten Kinetics

pH 9.0

[ ] ↳ *16 cells hidden*

Michaelis-Menten Kinetics

pH 7.8

[ ] ↳ *5 cells hidden*

Code Snippets From Workshop

[ ] ↳ *6 cells hidden*

Out of Scope

[ ] ↳ *21 cells hidden*