Gene expression analysis of Camelina transgenics expressing *LIP36* gene

Hesham Abdullah

Python for Data Science
Final project

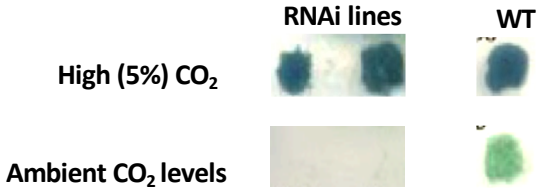
Abstract

A member of the mitochondrial carrier protein family and a component of algal carbon concentrating mechanisms (CCMs) in *Chlamydomonas reinhardtii* (Chlamy), designated, low-CO2 inducible protein (LIP36) was characterized in *planta* previously in our laboratory. LIP36 is localized to mitochondria in Camelina, and its suppression by RNA interference (RNAi) suggested its essential role in growth of Chlamy under low CO2 conditions3. To gain insight into the role of LIP36 proteins, we constitutively expressed LIP36 into the oilseed crop *Camelina sativa*. LIP36 expression has resulted in higher photosynthetic CO2 assimilation (30-46%), under limiting conditions, relative to the wildtype (WT) controls. LIP36 lines showed a more favorable redox status, higher water and nitrogen use efficiency, and increased seed and oil yields. In the current study, we aim to confirm the impact of LIP36 on the flux through central carbon metabolism in both source and sink (seed) tissues through expressing it under seed-specific oleosin promoter, and to emphasize whether its increased activity in seeds could lead to enhanced carbon assimilation, and therefore further increased seed and oil yields. The LIP36 expressing lines were subjected to RNA-Seq analysis to investigate the global changes in Camelina transcriptome in response to LIP36 expression in seeds, and to identify candidate genes/enzymes for further improving seed biomass and yield traits in Camelina.

Background

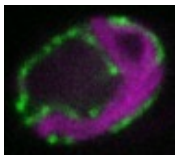
LIP36 is an uncharacterized mitochondrial carrier protein essential for growth under ambient conditions

LIP36 is essential for growth under low CO_2



Pollock et al., Plant Molecular Biology v 56, 2004.

LIP36 is localized to mitochondria in Chlamydomonas



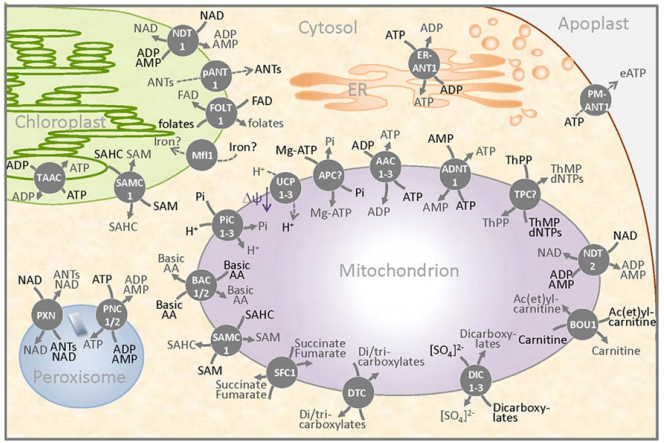
Atkinson et al., Plant Biotechnol J. 2016, 14(5): 1302–1315

LIP36 has an unknown function

Mitochondrial carrier protein family members

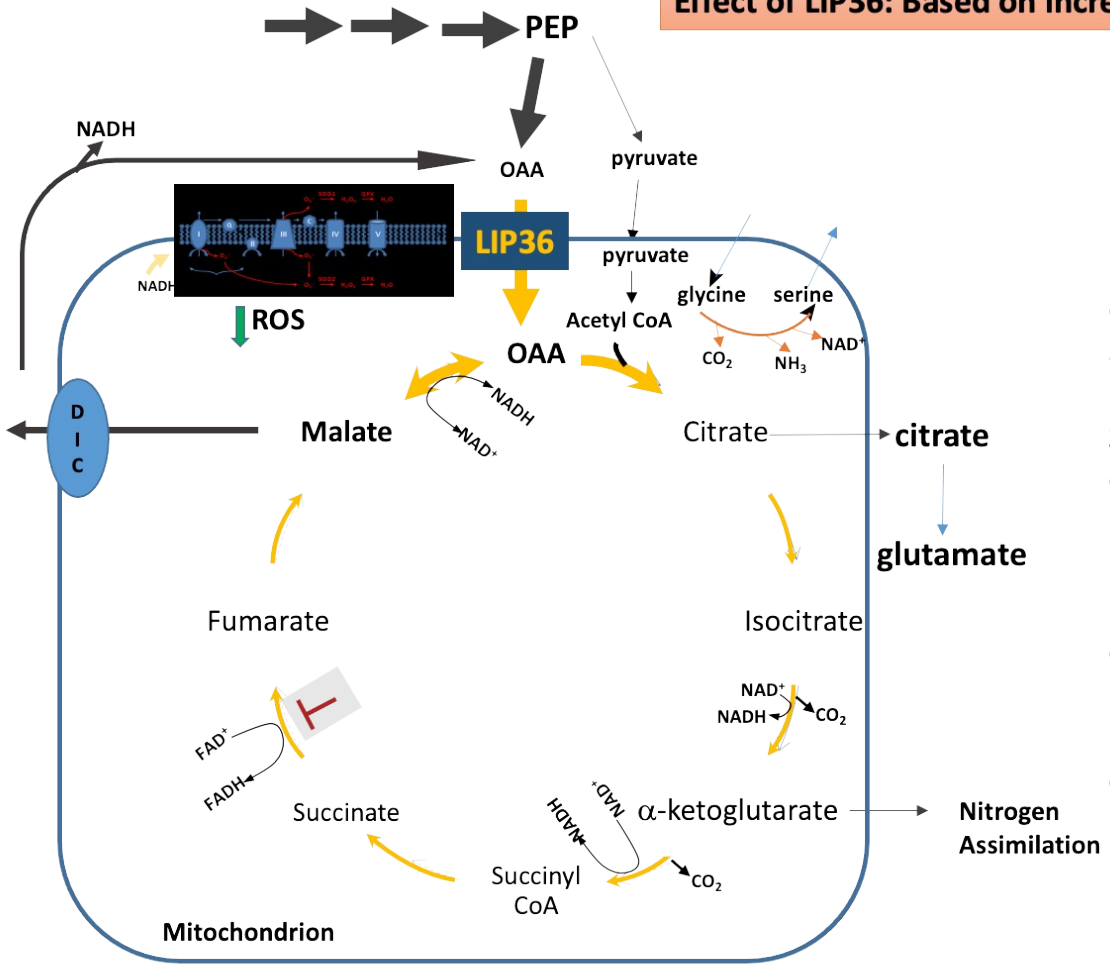
Transport a variety of substrates like inorganic ions, nucleotides, amino acids, keto acids and cofactors, and found in mitochondria and other organelles.

Some putative/characterized mitochondrial carriers in plants



Haferkamp and Schmitz-Esser, Front. Plant Sci., 2012

Effect of LIP36: Based on increased Partitioning of OAA to mitochondrial matrix



Proposed to work as a dicarboxylate transporter into the mitochondria thereby promoting the cellular redox balance, photorespiratory flux and anapleurotic metabolism

Shuttle reducing equivalents from mitochondria to cytoplasm thereby decreasing ROS levels in mitochondria leading to low ROS signaling

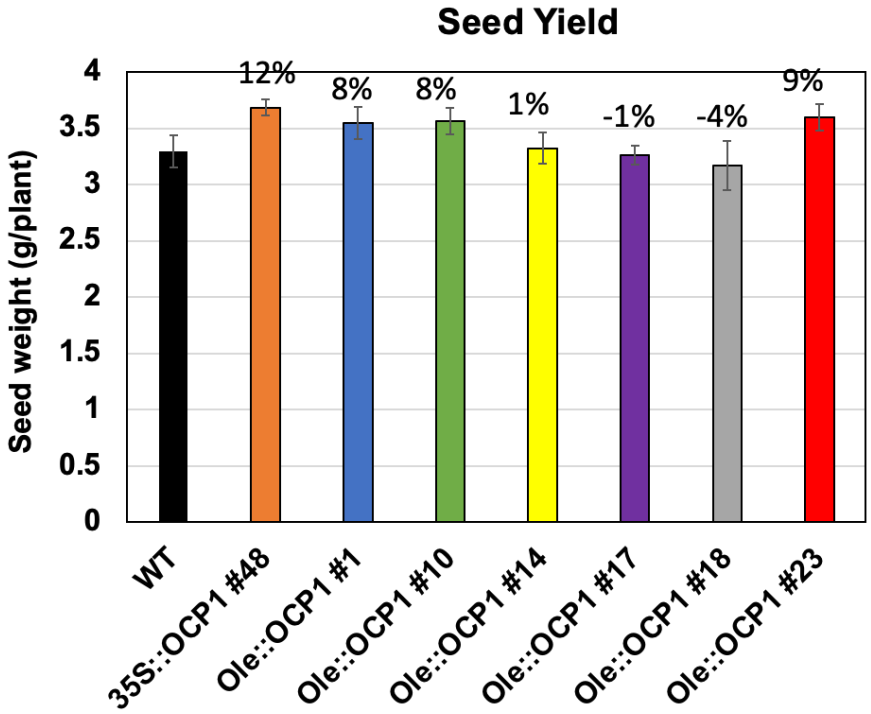
Photorespiratory flux balance: facilitates Gly to Ser conversion

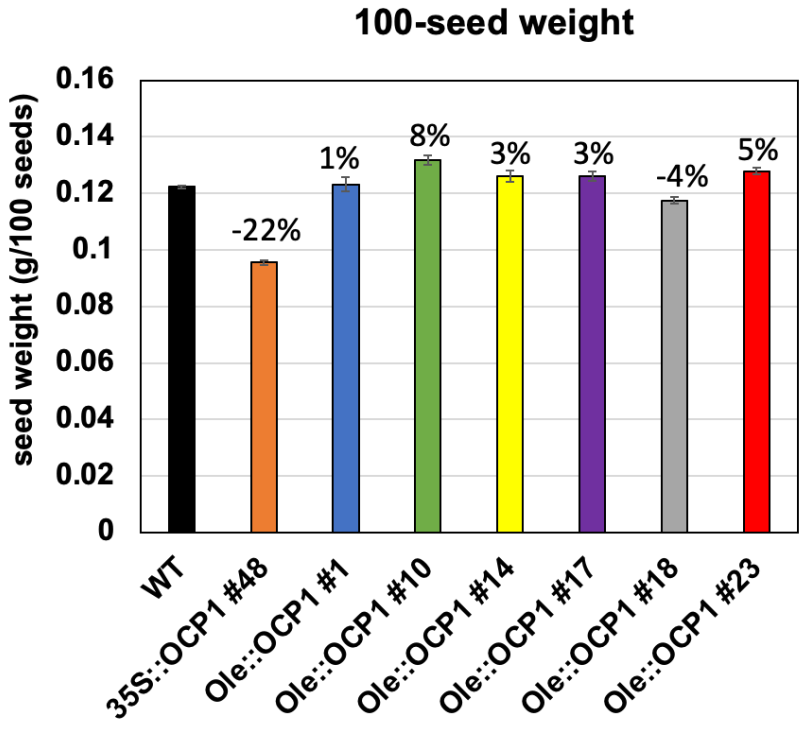
Promotes non-cyclic TCA pathway thus proving carbon skeleton for N-fixation.

Paulose et al., 2022. Plant Journal. Under revision

Phenotypes associated with LIP36 expression

- 35S-LIP36; small seed size, higher seed yield
- Ole::LIP36; slight bigger (or WT-like) seed size;





Motivation

Expressing LIP36 from the green algae Chlamydomonas (Chlamy) into Camelina plants caused significant improvements in several physiological, morphological, metabolic, and yield traits aspects. Therefore, investigating Camelina LIP36 expressing seeds at the molecular level (utilizing RNA-Seq datasets) would help interpreting these various changes and will lead to precisely select candidate genes (through differential gene expression analysis) for further improving seed biomass and yield traits in Camelina.

Through conducing this comprehensive analysis, utilizing Python in jupyter notebook, we aim to confirm the impact of LIP36 on the flux through central carbon metabolism in both source and sink (seed) tissues through expressing it under seed-specific oleosin promoter, and to emphasize whether its increased activity in seeds could lead to enhanced carbon assimilation, and therefore further increased seed and oil yields.

The LIP36 transgenic lines and their relative WT plants were subjected to RNA-Seq analysis to investigate the global changes in Camelina transcriptome in response to LIP36 expression in seeds, and to identify candidate genes/enzymes for further improving seed biomass and yield traits in Camelina.

Dataset and analysis focus

Gene expression dataset (GEdata) was utilized in the current study

- The dataset includes gene expression data (in TPMs) of Camelina seeds expressing *LIP36* gene (mutant) and WT seeds (control).
- The expression of a total of 178836 genes in the two genotypes (WT and LIP36) was analyzed.
- The dataset columns are gene IDs, gene expression data from three biological replicates of each genotype (WT-14 B1-B3, and LIP36-14 B1-B3).

The analysis using jupyter notebook focuses on:

1) Data wrangling
2) Data filtering
3) Exploratory data analysis
4) Differential expression analysis
5) Correlation analysis

Data Preparation and Cleaning

Our workflow reported in the current analysis includes:

1) Data wrangling
2) Data filtering
3) Exploratory data analysis
4) Differential expression analysis
5) Correlation analysis

- The original dataset (GEdata) contains gene expression data (in TPM) were cleaned to remove genome duplicates (gene IDs ended with '_2'); were transformed to log2 for statistical analysis.

- The original dataframe (contains both WT and LIP36 expression data) was splitted into two dataframes (one for WT and one for LIP36) to facilitate comparative analyses. dropped.filter function and RegExp were used to split the dataframe. Also, rows with 0 values were dropped.

- Data filtering was applied to report the up-regulated genes (fold change, log2-FC > 2) and down-regulated genes (log2-FC < -2) from the list of DEGs

Research Question(s)

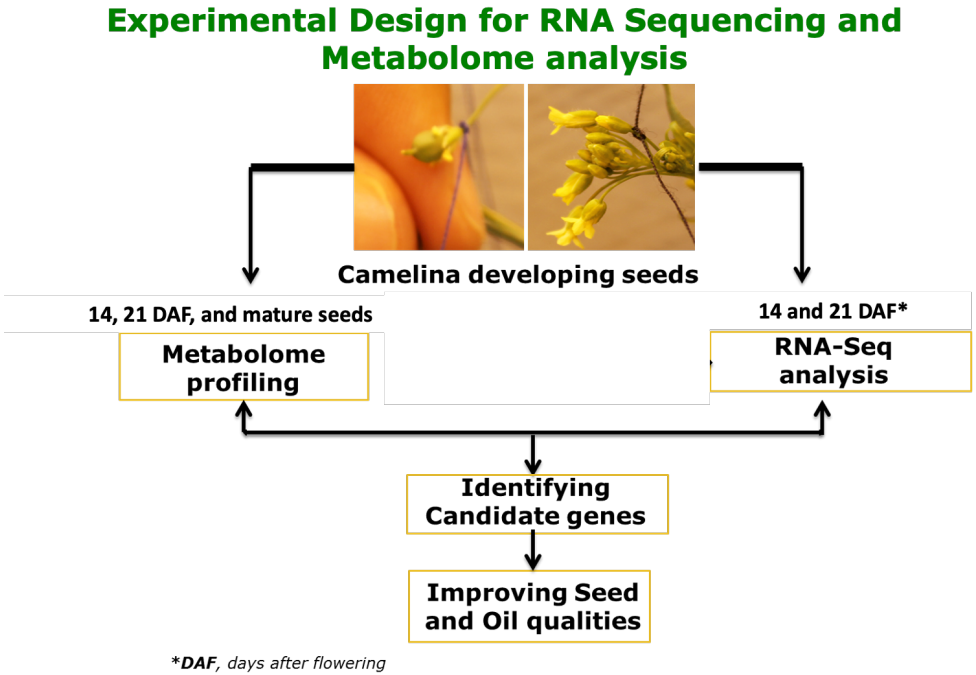
This research was designed to investigate the changes in gene expression in Camelina mutant (LIP36) to interpret the associated changes in metabolic profiling and seed yield attributed phenotypes.
In the current final project, I did utilize Data Science tools and resources to specifically answer these two questions

1) **How Camelina transcriptome (RNA-seq) has changed in response to *LIP36* gene expression in the LIP36 mutant relative to its WT control?**

2) **What are the top differentially regulated genes (DEGs, up or down) which could be good candidates for further investigations?**

3) **How the DEGs are correlated in terms of patterns of gene expression?**

Methods



- RNA-seq dataset used in this project obtained from Camelina developing seeds (14 DAF) of transgenic line (treated, LIP36) and control (WT).
- The Transcript per million (TPM) values were used to represent gene expression
- Differential expression analysis (DEA) was performed to report differential expressed genes (DEGs) between LIP36 and WT, and the up-regulated and down-regulated genes were reported (fold change).
- The top 25 Up and Down DEGs were reported in tables and were visualized in scatter plots and heatmaps.
- Correlation and co-expression analyses were performed to report the genes regulated in both genotypes (LIP36 and its control WT).

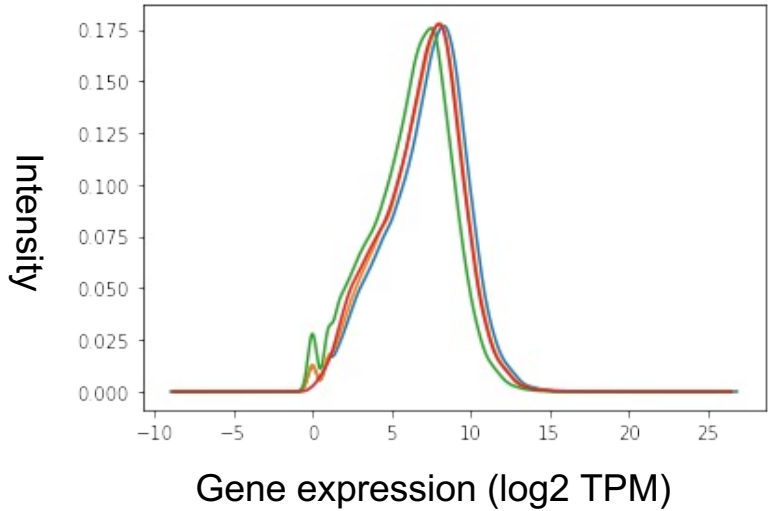
Findings

A snapshot for the normalized and log2 transformed dataset (WT) used in gene expression analysis

The density plots confirmed that the original dataset has been normalized and Both gene expression dataframes (WT and LIP36) appear normally distributed

Sample	WT-14-B1	WT-14-B2	WT-14-B3	mean1	variance1
Csa00382s010_1	10.556506	10.315150	9.076816	10.112005	17.371149
Csa00382s020_1	10.576484	10.288866	9.575539	10.204979	16.606457
Csa00382s040_1	2.807355	1.000000	2.321928	2.222392	2.078003
Csa00382s070_1	7.734710	7.546894	6.870365	7.429058	10.682605
Csa00382s080_1	5.930737	5.247928	4.392317	5.321928	8.069674

Density plots to confirm the shape of the distribution



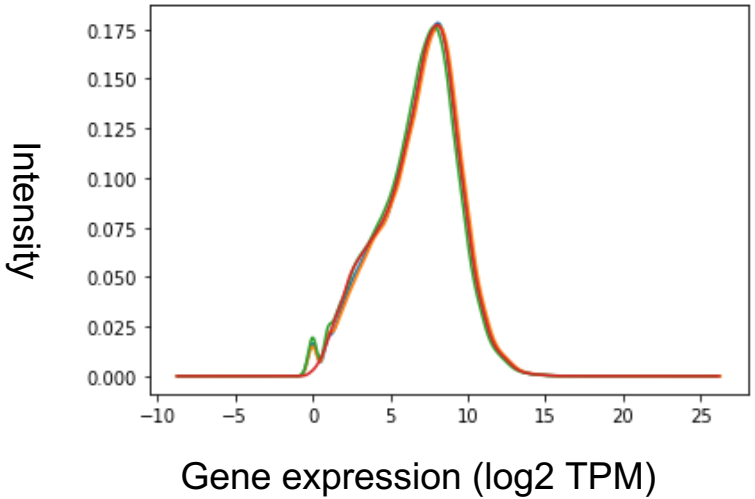
Findings

A snapshot for the normalized and log2 transformed dataset (LIP36) used in gene expression analysis

The density plots confirmed that the original dataset has been normalized and Both gene expression dataframes (WT and LIP36) appear normally distributed

Sample	WT-14-B1	WT-14-B2	WT-14-B3	mean1	variance1
Csa00382s010_1	10.556506	10.315150	9.076816	10.112005	17.371149
Csa00382s020_1	10.576484	10.288866	9.575539	10.204979	16.606457
Csa00382s040_1	2.807355	1.000000	2.321928	2.222392	2.078003
Csa00382s070_1	7.734710	7.546894	6.870365	7.429058	10.682605
Csa00382s080_1	5.930737	5.247928	4.392317	5.321928	8.069674

Density plots to confirm the shape of the distribution

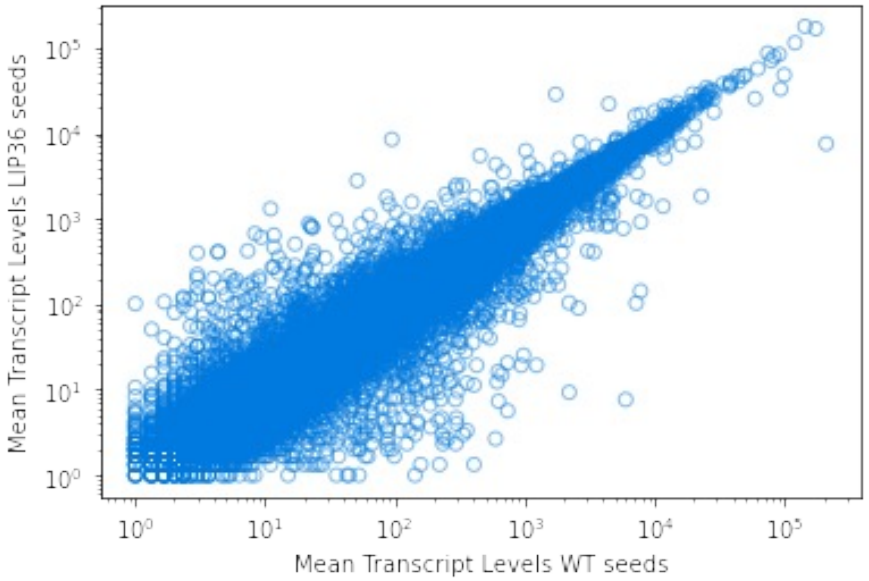


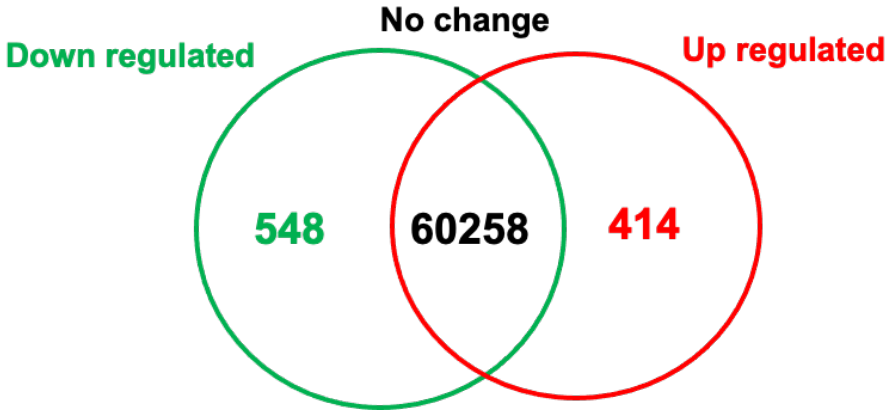
Differential gene expression analysis

The Scatter plot confirmed that there are biological differences exist between the two gene expression dataframes (WT and LIP36 gene expression data).

In the scatter plot, I plot the mean of the WT data against the mean of the LIP36 data

The genes with significant differences in mean expression levels are highlighted





Venn diagram shows the # of up-regulated and down-regulated genes in LIP36 seeds relative to WT seeds. Red, indicates UP; green, indicates Down; black, indicates no changes in expression between WT and LIP36

A table shows the top 10 Up-regulated genes in LIP36 seeds relative to WT seeds. Log2 fold change (Log2 FC) was reported

	mean1	variance1	mean2	variance2	FC	log2-FC
Sample	WT		LIP36			
Csa00855s010_1	3.000000	0.666667	399.000000	2648.000000	133.000000	7.055282
Csa09g066290_1	11.000000	0.666667	1309.666667	9709.555556	119.060606	6.895552
Csa02g005800_1	1.000000	0.000000	101.333333	189.555556	101.333333	6.662965
Csa20g006840_1	4.333333	6.888889	411.333333	1550.888889	94.923077	6.568687
Csa20g052950_1	4.333333	2.888889	405.666667	353.555556	93.615385	6.548674
Csa02g012920_1	94.333333	830.888889	8506.666667	212474.888889	90.176678	6.494682
Csa15g003390_1	8.333333	10.888889	673.333333	3360.888889	80.800000	6.336283
Csa09g005380_1	3.000000	0.666667	223.666667	104.222222	74.555556	6.220244
Csa09g034110_1	3.000000	2.000000	202.333333	124.222222	67.444444	6.075628
Csa08g047140_1	9.333333	22.222222	606.000000	3386.000000	64.928571	6.020782

A table shows the top 10 down-regulated genes in LIP36 seeds relative to WT seeds. Log2 fold change (Log2 FC) was reported

	mean1	variance1	mean2	variance2	FC	log2-FC
Sample	WT		LIP36			
Csa10g035950_1	5953.333333	1.509090e+06	7.666667	24.888889	0.001288	-9.600883
Csa03g012470_1	405.000000	6.734000e+03	1.333333	0.222222	0.003292	-8.246741
Csa05g002410_1	2188.000000	1.692687e+05	9.333333	3.555556	0.004266	-7.873005
Csa01g041890_1	592.666667	2.460289e+04	2.666667	2.888889	0.004499	-7.796040
Csa09g004680_1	220.000000	4.468667e+03	1.333333	0.222222	0.006061	-7.366322
Csa09g076020_1	142.333333	6.935556e+02	1.000000	0.000000	0.007026	-7.153130
Csa02g001800_1	734.333333	3.951756e+04	5.666667	1.555556	0.007717	-7.017791
Csa19g014920_1	293.666667	3.036222e+03	2.333333	0.222222	0.007946	-6.975643
Csa02g030650_1	157.333333	4.264222e+03	1.333333	0.222222	0.008475	-6.882643
Csa13g048260_1	334.666667	3.200889e+03	3.000000	0.666667	0.008964	-6.801619

Transcriptional changes during seed development (p-value < 0.05; FC>1.5)
Mapping DEGs to pathways analysis

The functional annotation of the DEGs resulted from the previous analyses.

The DEGs were mapped to pathways to identify their potential functions.

The annotation results confirmed the involvement of many DEGs in functions related to seed attributes

35S::LIP36 (@ 14 and 21 DAF)

Up-regulated				Down-regulated		
14 DAF		21 DAF		14 DAF		21 DAF
2		4	TCA cycle	3		4
66		94	AA metabolism	79		154
1		0	CBC cycle	0		19
3		2	Glycolysis	7		20
39		60	Lipid metabolism	26		79
0		0	nutrient uptake	0		2
21		27	Mitochondrial-localized	22		27
4		5	Photorespiration	2		27
15		8	Photosyntheis	9		127
10		7	Redox homeostasis	10		25
6		5	ROS production	18		52
75		99	Seed development	80		174
8		7	Seed maturation	11		21
8		16	Seed germination	9		20
59		62	Seedling growth	52		137
17		14	Ethylene-related	24		22
11		21	GA-related	11		39
16		10	Transcription factors	21		36
34		22	ABA-related	62		34

Limitation(s):

The datasets analyzed represent gene expression data obtained from developing Camelina seeds harvested at 14 days after flowering. However, to better profile gene expression, we need datasets cover different stages of seed development, therefore candidate genes associated with metabolic pathways of interest could be precisely selected and studied.

The research funding is always the top limitation to conduct high through-put and large-scale analyses, such as RNA-Seq, the technology utilized herein.

Conclusions

LIP36 expressing in Camelina seeds exhibited global changes in transcriptome and metabolite profiling which would lead to select candidate genes/enzymes for engineering strategies to improve Camelina productivity and qualities.

Acknowledgements

The datasets utilized in the current final project was an output of the research project funded by the Department of Energy - Biosystems Design to Enable Next-Generation Biofuels and Bioproducts (Award #DE-SC0018269).

The jupyter notebook was modified/forked from the following workflows:

- https://github.com/francismjenkins/Capstone_project
- https://github.com/LivLilli/Differentially-Expressed-Genes-Analysis

I also acknowledge the available online resources for problem solving with python coding and analysis workflows, and anyone I obtained help from throught this research.

References

- United States Department of Agriculture, F.A.S. https://apps.fas.usda.gov/psdonline/app/index.html#/app/downloads (2017).
- Rosillo-Calle, F., Pelkmans, L. & Walter, A. A global overview of vegetable oils, with reference to biodiesel. A Report for the IEA Bioenergy Task 40 (2009).
- Pollock SV, Prout DL, Godfrey AC, Lemaire SD, Moroney JV. The Chlamydomonas reinhardtii proteins Ccp1 and Ccp2 are required for long-term growth, but are not necessary for efficient photosynthesis, in a low-CO2 environment. Plant Mol Biol. 2004 Sep;56(1):125-32. doi: 10.1007/s11103-004-2650-4. PMID: 15604732.

In this notebook:

- Analysis of the dataset of Camelina WT and transgenic lines expressing LIP36 gene.
- The dataset is gene expression values from developing seeds (14 days after flowering). The expression levels are in "Transcript per milion TPM".
- Data wrangling
- Data filtering
- Exploratory data analysis
- Differential expression analysis
- Correlation analysis

Data wrangling

In [38]:
```
import pandas as pd
import os
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline

print("Current working directory" , os.getcwd()) # print out current working di
```

```
Current working directory /Users/hesham_biotech/Desktop/Research_work/Courses/
Python_for_Data_Science/Final_project
```

In [39]:
```
cd /Users/hesham_biotech/Desktop/Research_work/Courses/Python_for_Data_Science/
```

```
/Users/hesham_biotech/Desktop/Research_work/Courses/Python_for_Data_Science/Fi
nal_project
```

In [40]:
```
pwd
```

Out[40]:
```
'/Users/hesham_biotech/Desktop/Research_work/Courses/Python_for_Data_Science/F
inal_project'
```

Read in data and print header

In [54]:
```
filename = 'GEdata.csv' # assign dataset to filename object
CsDEG = pd.read_csv(filename, sep = ',') # read in csv file as pandas datafram
```

In [55]:
```
CsDEG = CsDEG.set_index('Sample')
```

In [56]:
```
CsDEG.head(100)
```

Out[56]:

Sample	WT-14-B1	WT-14-B2	WT-14-B3	OCP48-14-B1	OCP48-14-B2	OCP48-14-B3
Csa00382s010_1	1506	1274	540	838	883	892
Csa00382s010_2	29	60	29	27	44	41
Csa00382s020_1	1527	1251	763	1280	1476	1149
Csa00382s020_2	0	0	1	0	0	0
Csa00382s030_1	0	0	0	0	0	0
...	...	...	...	...	...	...
Csa00427s100_2	1	2	1	0	0	2
Csa00427s110_1	13	28	1	13	12	13
Csa00427s110_2	1	1	0	3	1	0
Csa00427s120_1	5	2	2	6	10	3
Csa00427s120_2	0	2	1	2	3	2

100 rows × 6 columns

In [57]:
```
CsDEG.shape
```

Out[57]:
```
(178836, 6)
```

In [58]:
```
CsDEG = CsDEG.filter(like ='_1', axis =0)
```

In [59]:
```
CsDEG.head()
```

Out[59]:

Sample	WT-14-B1	WT-14-B2	WT-14-B3	OCP48-14-B1	OCP48-14-B2	OCP48-14-B3
Csa00382s010_1	1506	1274	540	838	883	892
Csa00382s020_1	1527	1251	763	1280	1476	1149
Csa00382s030_1	0	0	0	0	0	0
Csa00382s040_1	7	2	5	9	5	8
Csa00382s050_1	0	0	0	1	0	0

In [28]:
```
# select rows containing '_1'
#CsDEG = CsDEG.filter(like='_1', axis=0)
```

In [60]:
```
CsDEG.shape
```

Out[60]:
```
(89418, 6)
```

Drop rows with zero values

In [61]:
```
df_replace = CsDEG.replace(0.0000, np.nan)
df_dropped = df_replace.dropna(axis=0, how='any')
df_dropped.shape
```

Out[61]:
```
(61220, 6)
```

In [62]:
```
df_dropped.head()
```

Out[62]:

Sample	WT-14-B1	WT-14-B2	WT-14-B3	OCP48-14-B1	OCP48-14-B2	OCP48-14-B3
Csa00382s010_1	1506.0	1274.0	540.0	838.0	883.0	892.0
Csa00382s020_1	1527.0	1251.0	763.0	1280.0	1476.0	1149.0
Csa00382s040_1	7.0	2.0	5.0	9.0	5.0	8.0
Csa00382s070_1	213.0	187.0	117.0	183.0	199.0	177.0
Csa00382s080_1	61.0	38.0	21.0	32.0	40.0	26.0

In [220…
```
#data = df_dropped.filter(like = '_1', axis =0)
# Creating new dataframe with only columns containing expression values for WT
```

Split data into two dataframes:

- The first dataframe (WT_df) contains expression values for WT seeds
- The second dataframe (LIP36_df) contains expression values for LIP36 seeds

In [73]:
```
WT_df = df_dropped.filter(like = 'WT', axis =1)
# Creating new dataframe with only columns containing expression values for WT
```

In [74]:
```
WT_df
```

Out[74]:

Sample	WT-14-B1	WT-14-B2	WT-14-B3
Csa00382s010_1	1506.0	1274.0	540.0
Csa00382s020_1	1527.0	1251.0	763.0
Csa00382s040_1	7.0	2.0	5.0
Csa00382s070_1	213.0	187.0	117.0
Csa00382s080_1	61.0	38.0	21.0
...	...	...	...
Csa34454s010_1	7.0	11.0	7.0
Csa34686s010_1	2.0	3.0	1.0
Csa34903s010_1	1.0	7.0	1.0
Csa36836s010_1	13.0	21.0	8.0
Csa37955s010_1	28.0	45.0	13.0

61220 rows × 3 columns

In [ ]:
```
# transform the WT_df dataframe to Log2
```

In [165…
```
WT_df_log2 = np.log2(WT_df)
```

In [166…
```
WT_df_log2.head()
```

Out[166]:

Sample	WT-14-B1	WT-14-B2	WT-14-B3	mean1	variance1
Csa00382s010_1	10.556506	10.315150	9.076816	10.112005	17.371149
Csa00382s020_1	10.576484	10.288866	9.575539	10.204979	16.606457
Csa00382s040_1	2.807355	1.000000	2.321928	2.222392	2.078003
Csa00382s070_1	7.734710	7.546894	6.870365	7.429058	10.682605
Csa00382s080_1	5.930737	5.247928	4.392317	5.321928	8.069674

In [167…
```
WT_df_log2.plot(kind='kde', legend = False) # WT seeds
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [167], in <cell line: 1>()
----> 1 WT_df_log2.plot(kind='kde', legend = False)

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/pandas/plotting/_core.py:972, in PlotAccessor.__call__(self, *args, **kwargs)
    969             label_name = label_kw or data.columns
    970             data.columns = label_name
--> 972 return plot_backend.plot(data, kind=kind, **kwargs)

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/pandas/plotting/_matplotlib/__init__.py:71, in plot(data, kind, **kwargs)
     69         kwargs["ax"] = getattr(ax, "left_ax", ax)
     70 plot_obj = PLOT_CLASSES[kind](data, **kwargs)
---> 71 plot_obj.generate()
     72 plot_obj.draw()
     73 return plot_obj.result

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/pandas/plotting/_matplotlib/core.py:329, in MPLPlot.generate(self)
    327 self._compute_plot_data()
    328 self._setup_subplots()
--> 329 self._make_plot()
    330 self._add_table()
    331 self._make_legend()

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/pandas/plotting/_matplotlib/hist.py:140, in HistPlot._make_plot(self)
    137 if weights is not None and np.ndim(weights) != 1:
    138     kwds["weights"] = weights[:, i]
--> 140 artists = self._plot(ax, y, column_num=i, stacking_id=stacking_id, **kwds)
    142 # when by is applied, show title for subplots to know which group it is
    143 if self.by is not None:

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/pandas/plotting/_matplotlib/hist.py:216, in KdePlot._plot(cls, ax, y, style, bw_method, ind, column_num, stacking_id, **kwds)
    213 from scipy.stats import gaussian_kde
    215 y = remove_na_arraylike(y)
--> 216 gkde = gaussian_kde(y, bw_method=bw_method)
    218 y = gkde.evaluate(ind)
    219 lines = MPLPlot._plot(ax, ind, y, style=style, **kwds)

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/scipy/stats/_kde.py:207, in gaussian_kde.__init__(self, dataset, bw_method, weights)
    204             raise ValueError("`weights` input should be of length n")
    205         self._neff = 1/sum(self._weights**2)
--> 207 self.set_bandwidth(bw_method=bw_method)

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/scipy/stats/_kde.py:555, in gaussian_kde.set_bandwidth(self, bw_method)
    551     msg = "`bw_method` should be 'scott', 'silverman', a scalar " \
    552           "or a callable."
    553     raise ValueError(msg)
--> 555 self._compute_covariance()
```

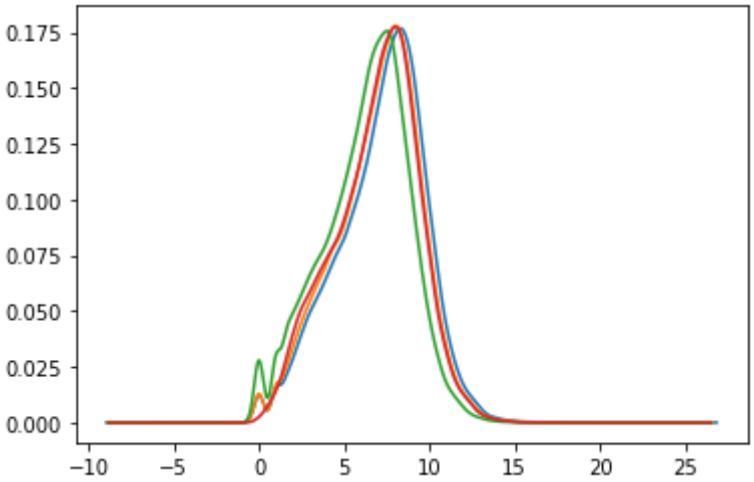
```
File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-pack
ages/scipy/stats/_kde.py:567, in gaussian_kde._compute_covariance(self)
    563 if not hasattr(self, '_data_inv_cov'):
    564     self._data_covariance = atleast_2d(cov(self.dataset, rowvar=1,
    565                                            bias=False,
    566                                            aweights=self.weights))
--> 567     self._data_inv_cov = linalg.inv(self._data_covariance)
    569 self.covariance = self._data_covariance * self.factor**2
    570 self.inv_cov = self._data_inv_cov / self.factor**2

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-pack
ages/scipy/linalg/_basic.py:939, in inv(a, overwrite_a, check_finite)
    900 def inv(a, overwrite_a=False, check_finite=True):
    901     """
    902     Compute the inverse of a matrix.
    903
   (...)
    937
    938     """
--> 939     a1 = _asarray_validated(a, check_finite=check_finite)
    940     if len(a1.shape) != 2 or a1.shape[0] != a1.shape[1]:
    941         raise ValueError('expected square matrix')

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-pack
ages/scipy/_lib/_util.py:287, in _asarray_validated(a, check_finite, sparse_o
k, objects_ok, mask_ok, as_inexact)
    285         raise ValueError('masked arrays are not supported')
    286 toarray = np.asarray_chkfinite if check_finite else np.asarray
--> 287 a = toarray(a)
    288 if not objects_ok:
    289     if a.dtype is np.dtype('O'):

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-pack
ages/numpy/lib/function_base.py:603, in asarray_chkfinite(a, dtype, order)
    601 a = asarray(a, dtype=dtype, order=order)
    602 if a.dtype.char in typecodes['AllFloat'] and not np.isfinite(a).all():
--> 603     raise ValueError(
    604         "array must not contain infs or NaNs")
    605 return a

ValueError: array must not contain infs or NaNs
```



- Both gene exp dataframes appear normally distributed
- This confirms that the original dataset has been normalized

In [75]:
```
LIP36_df = df_dropped.filter(like = 'OCP', axis =1)
# Creating new dataframe with only columns containing expression values for WT
```

In [76]:
```
LIP36_df
```

Out[76]:

Sample	OCP48-14-B1	OCP48-14-B2	OCP48-14-B3
Csa00382s010_1	838.0	883.0	892.0
Csa00382s020_1	1280.0	1476.0	1149.0
Csa00382s040_1	9.0	5.0	8.0
Csa00382s070_1	183.0	199.0	177.0
Csa00382s080_1	32.0	40.0	26.0
...	...	...	...
Csa34454s010_1	16.0	14.0	7.0
Csa34686s010_1	9.0	2.0	4.0
Csa34903s010_1	2.0	2.0	1.0
Csa36836s010_1	7.0	8.0	3.0
Csa37955s010_1	22.0	37.0	28.0

61220 rows × 3 columns

transform the LIP36_df dataframe to Log2

In [168…
```
LIP36_df_log2 = np.log2(LIP36_df)
```

In [169…
```
LIP36_df_log2.head()
```

Out[169]:

Sample	OCP48-14-B1	OCP48-14-B2	OCP48-14-B3	mean2	variance2
Csa00382s010_1	9.710806	9.786270	9.800900	9.766529	9.124121
Csa00382s020_1	10.321928	10.527477	10.166163	10.346144	14.140208
Csa00382s040_1	3.169925	2.321928	3.000000	2.874469	1.530515
Csa00382s070_1	7.515700	7.636625	7.467606	7.541742	6.429988
Csa00382s080_1	5.000000	5.321928	4.700440	5.029747	5.039528

In [170…
```
LIP36_df_log2.plot(kind='kde', legend = False) # LIP36 seeds
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [170], in <cell line: 1>()
----> 1 LIP36_df_log2.plot(kind='kde', legend = False)

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/pandas/plotting/_core.py:972, in PlotAccessor.__call__(self, *args, **kwargs)
    969             label_name = label_kw or data.columns
    970             data.columns = label_name
--> 972 return plot_backend.plot(data, kind=kind, **kwargs)

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/pandas/plotting/_matplotlib/__init__.py:71, in plot(data, kind, **kwargs)
     69         kwargs["ax"] = getattr(ax, "left_ax", ax)
     70 plot_obj = PLOT_CLASSES[kind](data, **kwargs)
---> 71 plot_obj.generate()
     72 plot_obj.draw()
     73 return plot_obj.result

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/pandas/plotting/_matplotlib/core.py:329, in MPLPlot.generate(self)
    327 self._compute_plot_data()
    328 self._setup_subplots()
--> 329 self._make_plot()
    330 self._add_table()
    331 self._make_legend()

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/pandas/plotting/_matplotlib/hist.py:140, in HistPlot._make_plot(self)
    137 if weights is not None and np.ndim(weights) != 1:
    138     kwds["weights"] = weights[:, i]
--> 140 artists = self._plot(ax, y, column_num=i, stacking_id=stacking_id, **kwds)
    142 # when by is applied, show title for subplots to know which group it is
    143 if self.by is not None:

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/pandas/plotting/_matplotlib/hist.py:216, in KdePlot._plot(cls, ax, y, style, bw_method, ind, column_num, stacking_id, **kwds)
    213 from scipy.stats import gaussian_kde
    215 y = remove_na_arraylike(y)
--> 216 gkde = gaussian_kde(y, bw_method=bw_method)
    218 y = gkde.evaluate(ind)
    219 lines = MPLPlot._plot(ax, ind, y, style=style, **kwds)

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/scipy/stats/_kde.py:207, in gaussian_kde.__init__(self, dataset, bw_method, weights)
    204         raise ValueError("`weights` input should be of length n")
    205     self._neff = 1/sum(self._weights**2)
--> 207 self.set_bandwidth(bw_method=bw_method)

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/scipy/stats/_kde.py:555, in gaussian_kde.set_bandwidth(self, bw_method)
    551     msg = "`bw_method` should be 'scott', 'silverman', a scalar " \
    552           "or a callable."
    553     raise ValueError(msg)
--> 555 self._compute_covariance()
```

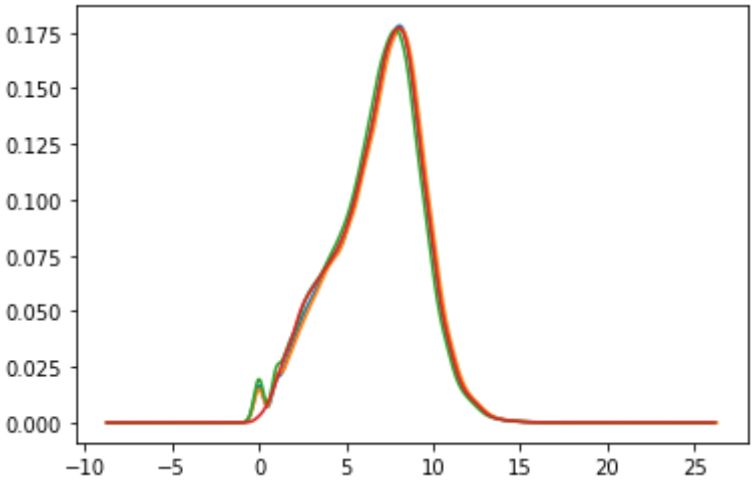
```
File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/scipy/stats/_kde.py:567, in gaussian_kde._compute_covariance(self)
    563 if not hasattr(self, '_data_inv_cov'):
    564     self._data_covariance = atleast_2d(cov(self.dataset, rowvar=1,
    565                                            bias=False,
    566                                            aweights=self.weights))
--> 567     self._data_inv_cov = linalg.inv(self._data_covariance)
    569 self.covariance = self._data_covariance * self.factor**2
    570 self.inv_cov = self._data_inv_cov / self.factor**2

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/scipy/linalg/_basic.py:939, in inv(a, overwrite_a, check_finite)
    900 def inv(a, overwrite_a=False, check_finite=True):
    901     """
    902     Compute the inverse of a matrix.
    903
   (...)
    937
    938     """
--> 939     a1 = _asarray_validated(a, check_finite=check_finite)
    940     if len(a1.shape) != 2 or a1.shape[0] != a1.shape[1]:
    941         raise ValueError('expected square matrix')

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/scipy/_lib/_util.py:287, in _asarray_validated(a, check_finite, sparse_ok, objects_ok, mask_ok, as_inexact)
    285         raise ValueError('masked arrays are not supported')
    286 toarray = np.asarray_chkfinite if check_finite else np.asarray
--> 287 a = toarray(a)
    288 if not objects_ok:
    289     if a.dtype is np.dtype('O'):

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/numpy/lib/function_base.py:603, in asarray_chkfinite(a, dtype, order)
    601 a = asarray(a, dtype=dtype, order=order)
    602 if a.dtype.char in typecodes['AllFloat'] and not np.isfinite(a).all():
--> 603     raise ValueError(
    604         "array must not contain infs or NaNs")
    605 return a

ValueError: array must not contain infs or NaNs
```

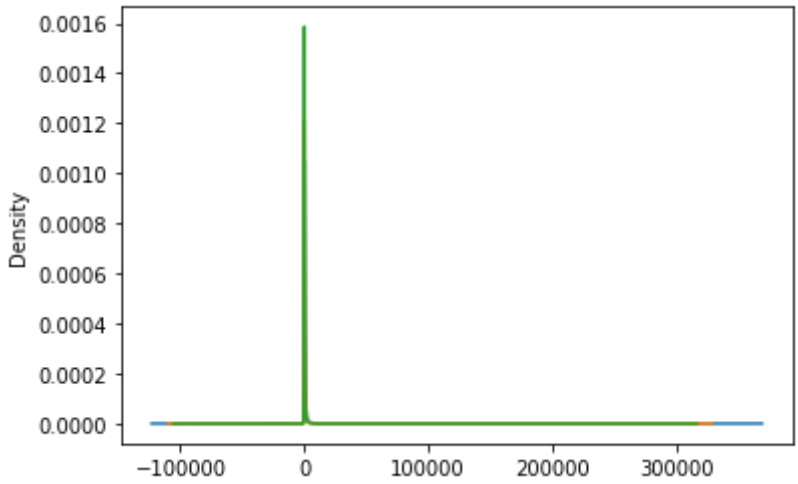


Generate density plots to confirm the shape of the distribution

KDE plots of processed data separated by sample type (transgenic/WT)

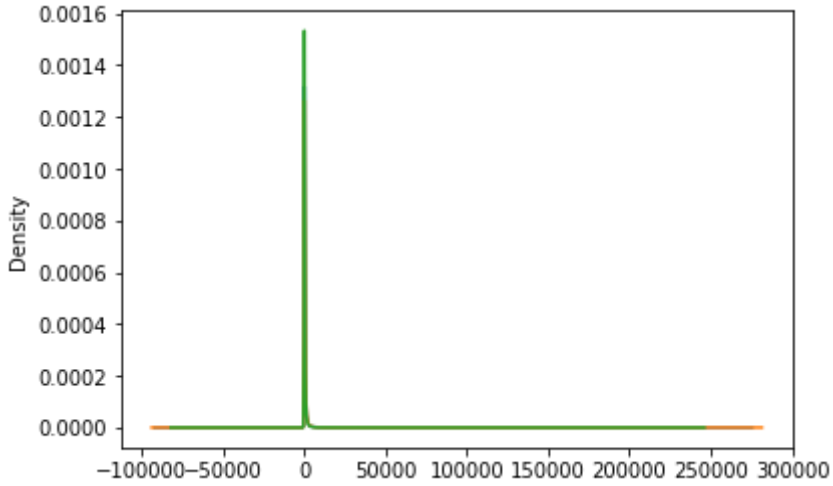
In [77]:
```
WT_df.plot(kind='kde', legend = False) # WT seeds
```

Out[77]:
```
<AxesSubplot:ylabel='Density'>
```



In [78]:
```
LIP36_df.plot(kind='kde', legend = False) # LIP36 seeds
```

Out[78]:
```
<AxesSubplot:ylabel='Density'>
```



Add columns with row-wise mean & variance to gene exp dataframes

In [79]:
```
import warnings
warnings.simplefilter("ignore")
from scipy.stats import ttest_ind, ttest_ind_from_stats


WT_df['mean1'] = WT_df[WT_df.columns].mean(axis=1)
WT_df['variance1'] = WT_df[WT_df.columns].var(axis=1)
```

```
LIP36_df['mean2'] = LIP36_df[LIP36_df.columns].mean(axis=1)
LIP36_df['variance2'] = LIP36_df[LIP36_df.columns].var(axis=1)
data['mean'] = data[data.columns].mean(axis=1)
data['variance'] = data[data.columns].var(axis=1)
```

In [80]:
```
WT_df_stats = WT_df[['mean1', 'variance1']]
WT_df_stats.head()
```

Out[80]:

Sample	mean1	variance1
Csa00382s010_1	1106.666667	169526.222222
Csa00382s020_1	1180.333333	99779.555556
Csa00382s040_1	4.666667	4.222222
Csa00382s070_1	172.333333	1643.555556
Csa00382s080_1	40.000000	268.666667

In [81]:
```
LIP36_df_stats = LIP36_df[['mean2', 'variance2']]
LIP36_df_stats.head()
```

Out[81]:

Sample	mean2	variance2
Csa00382s010_1	871.000000	558.000000
Csa00382s020_1	1301.666667	18056.222222
Csa00382s040_1	7.333333	2.888889
Csa00382s070_1	186.333333	86.222222
Csa00382s080_1	32.666667	32.888889

Concatenate mean/variance columns of gene exp dataframes

In [82]:
```
df = pd.concat([WT_df_stats, LIP36_df_stats], axis=1)
```

Visualize the new dataframe to confirm proper formatting

In [83]:
```
df.head()
```

Out[83]:

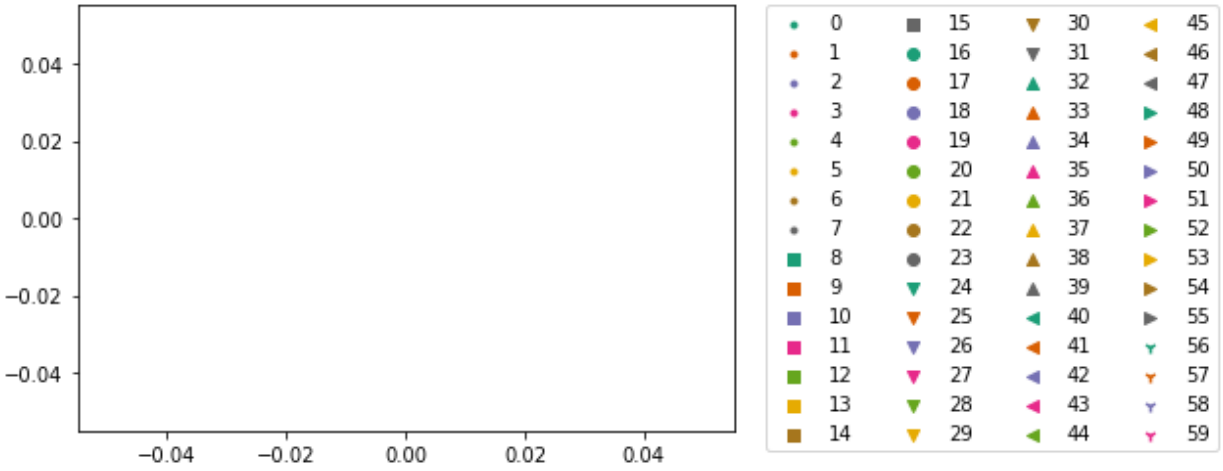
Sample	mean1	variance1	mean2	variance2
Csa00382s010_1	1106.666667	169526.222222	871.000000	558.000000
Csa00382s020_1	1180.333333	99779.555556	1301.666667	18056.222222
Csa00382s040_1	4.666667	4.222222	7.333333	2.888889
Csa00382s070_1	172.333333	1643.555556	186.333333	86.222222
Csa00382s080_1	40.000000	268.666667	32.666667	32.888889

Generate scatter plots:

- Confirm that biological differences exist between the two gene expression dataframes
- In the first scatter plot, we plot the mean vs the variance of the entire dataset
- In the second scatter plot, we plot the mean of the WT data against the mean of the LIP36 data
- In both cases,

In [89]:
```
import itertools
import numpy as np
from matplotlib import markers
import matplotlib.pyplot as plt

m_styles = markers.MarkerStyle.markers
N = 60
colormap = plt.cm.Dark2.colors   # Qualitative colormap
for i,(marker,color) in zip(range(N),itertools.product(m_styles, colormap)):
    plt.scatter(data["mean"], data["variance"], color=color,marker=marker,label
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.,ncol=4);
```

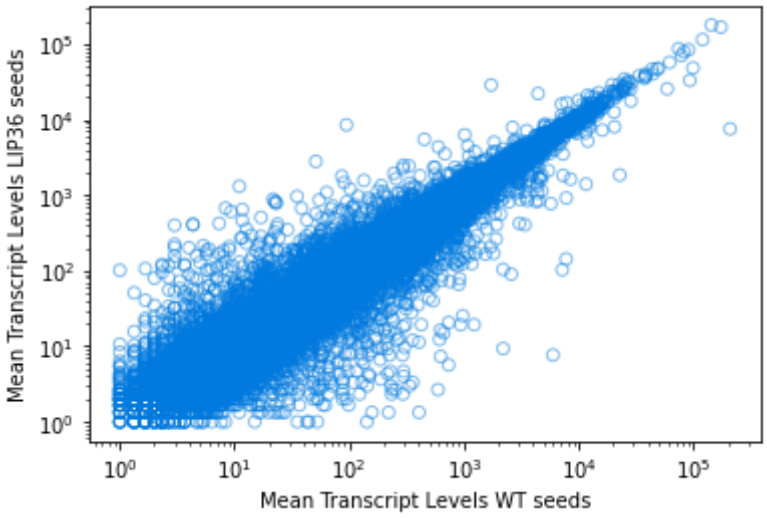


In [90]:
```
import copy
import matplotlib.pyplot as plt
my_figure = plt.figure()
plt.loglog() # log-scaling makes it easier


plot_df = copy.deepcopy(df)


# now, make a scatter plot
plt.scatter(x=plot_df["mean1"],
            y=plot_df["mean2"],
            marker=None,alpha=0.5,facecolor="none",edgecolor="#007ADF")
plt.xlabel("Mean Transcript Levels WT seeds")
plt.ylabel("Mean Transcript Levels LIP36 seeds")

plt.show()
```



- We can see genes with significant differences in mean expression levels
- This indicates that true biological differences exist between the LIP36 and WT gene exp dataframes

Differential expression analysis

- Run t-test and load results in new column
- Sort results in descending order
- Print top 25 differentially expressed genes

In [91]:
```
df['ttest'] = abs((df['mean1'] - df['mean2'])/(np.sqrt(df['variance1']/3 + df['
```

In [92]:
```
diffexp_results = df['ttest']
diffexp = diffexp_results.sort_values(ascending=False)
diffexp.shape
```

Out[92]:
```
(61220,)
```

The top 25 differentially expressed genes:

In [93]:
```
results = diffexp.iloc[1:25]
```

In [94]:
```
results.head(25)
```

Out[94]:
```
Sample
Csa02g002370_1          inf
Csa04g057840_1    94.208280
Csa03g058410_1    62.105074
Csa09g042880_1    54.008928
Csa20g079450_1    46.243754
Csa08g062550_1    42.668541
Csa04g043970_1    38.406966
Csa09g005380_1    37.319225
Csa20g052950_1    36.818826
Csa07g030300_1    36.233737
Csa19g040760_1    35.196143
Csa17g050280_1    34.639097
Csa11g015040_1    33.824945
Csa13g025940_1    33.678257
Csa03g019280_1    33.411076
Csa11g073680_1    31.983932
Csa07g059760_1    31.625662
Csa02g012920_1    31.548288
Csa09g034110_1    30.730714
Csa11g074500_1    30.618622
Csa03g001020_1    30.424497
Csa15g019640_1    28.762633
Csa15g020930_1    28.354625
Csa06g009080_1    28.117133
Name: ttest, dtype: float64
```

Correlation analysis

- Transpose dataframes so gene names are column names
- Perform correlation analysis using corr()

In [95]:
```
WT_df_t = WT_df.T
correlations1 = WT_df_t.corr()
correlations1.head()
```

Out[95]:

Sample	Csa00382s010_1	Csa00382s020_1	Csa00382s040_1	Csa00382s070_1	Csa
Sample					
Csa00382s010_1	1.000000	0.999999	-0.110453	0.998820	
Csa00382s020_1	0.999999	1.000000	-0.109478	0.998889	
Csa00382s040_1	-0.110453	-0.109478	1.000000	-0.102935	
Csa00382s070_1	0.998820	0.998889	-0.102935	1.000000	
Csa00382s080_1	0.991097	0.991311	-0.045038	0.995990	

5 rows × 61220 columns

In [96]:
```
LIP36_df_t = LIP36_df.T
correlations2 = LIP36_df_t.corr()
correlations2.head()
```

Out[96]:

Sample	Csa00382s010_1	Csa00382s020_1	Csa00382s040_1	Csa00382s070_1	Csa
Sample					
Csa00382s010_1	1.000000	-0.989463	0.745876	0.976468	
Csa00382s020_1	-0.989463	1.000000	-0.810874	-0.981413	
Csa00382s040_1	0.745876	-0.810874	1.000000	0.699119	
Csa00382s070_1	0.976468	-0.981413	0.699119	1.000000	
Csa00382s080_1	-0.030468	0.035522	-0.477295	0.154380	

5 rows × 61220 columns

Create dataframes with correlated values for differentially expressed genes

In [97]:
```
diff_exp1 = correlations1 [['Csa02g002370_1', 'Csa04g057840_1', 'Csa03g058410_1
```

In [98]:
```
diff_exp2 = correlations2 [['Csa02g002370_1', 'Csa04g057840_1', 'Csa03g058410_1
```

Create object with only highly correlated genes

In [99]:
```
top_corr1 = diff_exp1[(diff_exp1 > abs(0.90)).any(1)]
top_corr1 = diff_exp1[(diff_exp1 < abs(1.0)).any(1)]
```

In [101…
```
top_corr1.shape
```

Out[101]:
```
(61220, 22)
```

In [102…
```
top_corr2 = diff_exp2[(diff_exp2 > abs(0.90)).any(1)]
top_corr2 = diff_exp2[(diff_exp2 < abs(1.0)).any(1)]
```

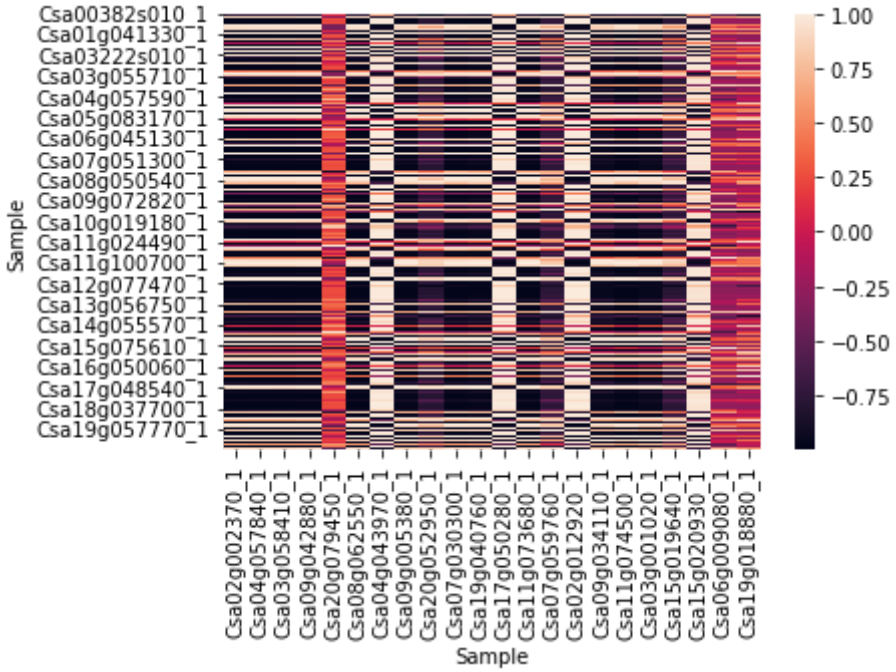
In [103…
```
top_corr2.shape
```

Out[103]:
```
(61220, 22)
```

Let's look at heatmaps of the top correlated cancer and normal genes

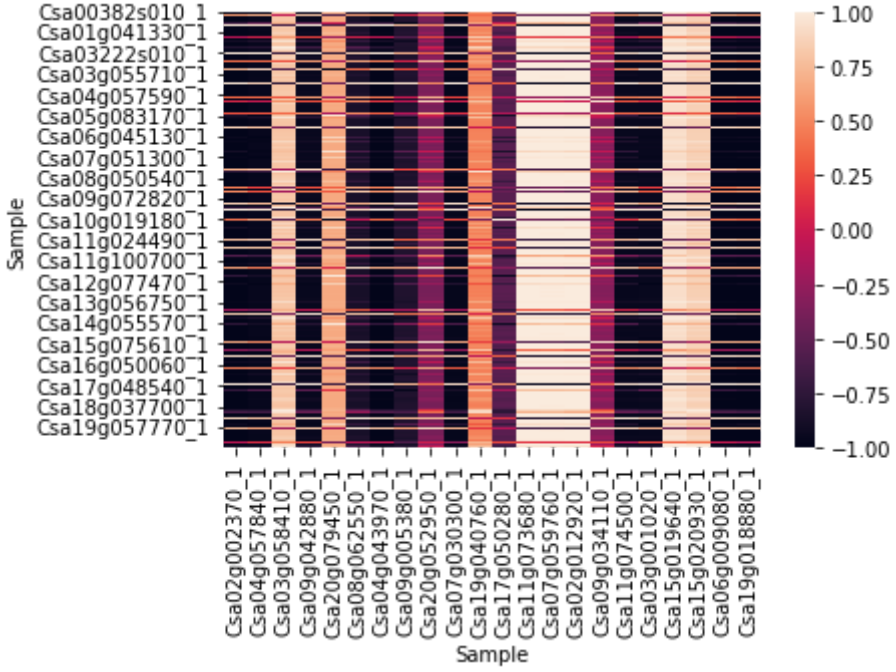
In [105…
```
import seaborn as sns
sns.heatmap(top_corr2)
```

Out[105]:
```
<AxesSubplot:xlabel='Sample', ylabel='Sample'>
```



In [125…
```
sns.heatmap(top_corr1)
```

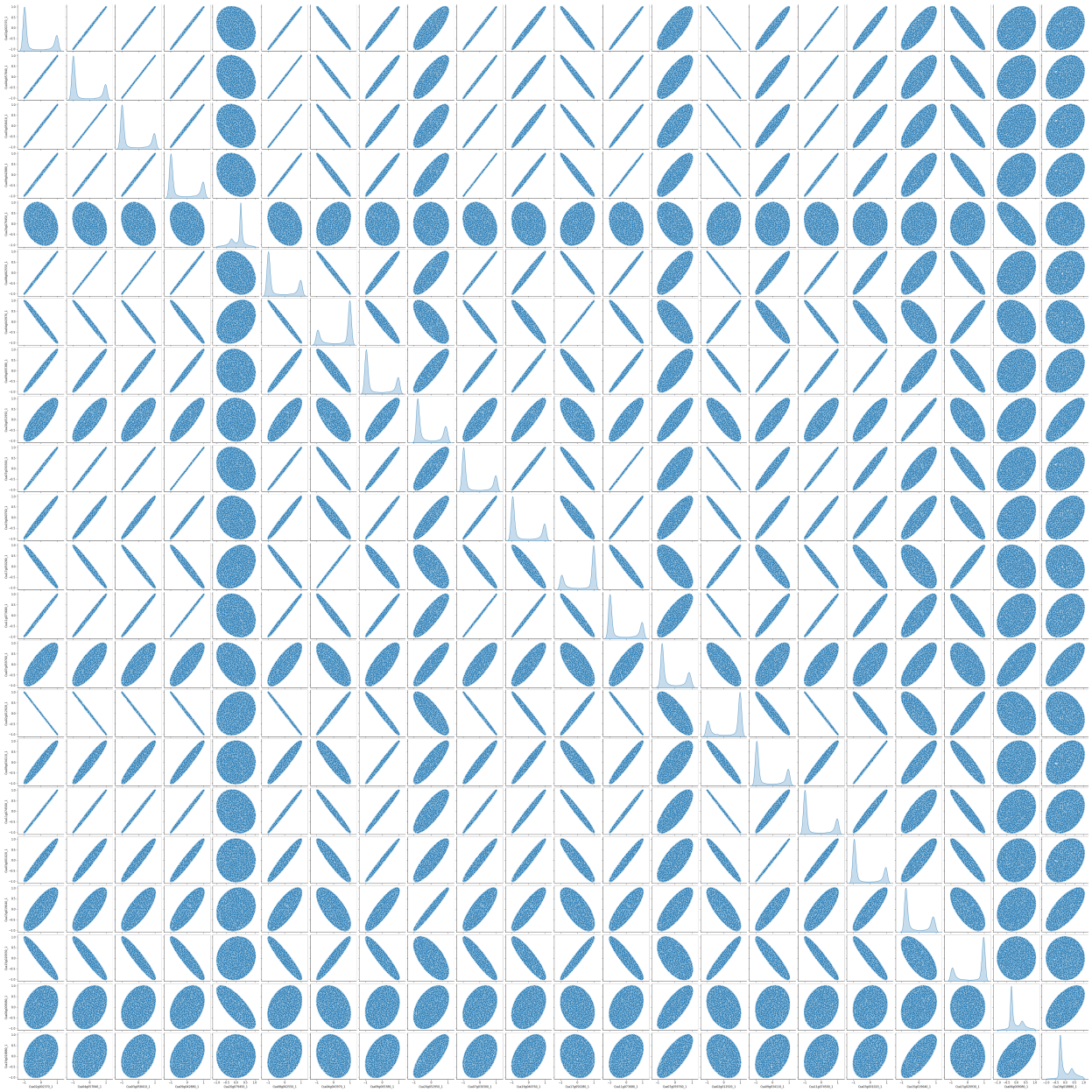
Out[125]:
```
<AxesSubplot:xlabel='Sample', ylabel='Sample'>
```



Let's generate pairs plots for the correlation data to further demonstrate the trends shown by the heatmaps

In [126…
```
sns.pairplot(top_corr2, diag_kind='kde')
```

Out[126]:
```
<seaborn.axisgrid.PairGrid at 0x7f738ea86640>
```



In [ ]:

Co-Expression network using Fold-Change values

Fold Change is useful to identify genes whose expression in the two groups of considered samples varies by a certain proportion (doubles, halves, ...).

We define FC as follows:

$$FC = log_2(\frac{TumoralExpr}{NormalExpr})$$

Choosing a certain FC threshold, all the genes with absolute value of FC higher than the threshold will be differentially expressed, then they will change significantly passing from a normal condition to a tumoral one.

After this, based on the FC matrix, we compute the correlation matrix and, choosing a certain threshold, we will build an adjacency matrix: higher values will correspond to ones in our adjacency matrix, lower ones will be zeros.

So, we can obtain our directed graph.

Concatenate mean for the two dataframes (WT and LIP36) to calculate the fold change

In [107…
```
WT_df_stats = WT_df[['mean1', 'variance1']]
WT_df_stats.head()
```

Out[107]:

Sample	mean1	variance1
Csa00382s010_1	1106.666667	169526.222222
Csa00382s020_1	1180.333333	99779.555556
Csa00382s040_1	4.666667	4.222222
Csa00382s070_1	172.333333	1643.555556
Csa00382s080_1	40.000000	268.666667

In [108…
```
LIP36_df_stats = LIP36_df[['mean2', 'variance2']]
LIP36_df_stats.head()
```

Out[108]:

Sample	mean2	variance2
Csa00382s010_1	871.000000	558.000000
Csa00382s020_1	1301.666667	18056.222222
Csa00382s040_1	7.333333	2.888889
Csa00382s070_1	186.333333	86.222222
Csa00382s080_1	32.666667	32.888889

In [109…
```
Mean_for_FC = pd.concat([WT_df_stats, LIP36_df_stats], axis=1)
```

In [110…
```
Mean_for_FC.head()
```

Out[110]:

Sample	mean1	variance1	mean2	variance2
Csa00382s010_1	1106.666667	169526.222222	871.000000	558.000000
Csa00382s020_1	1180.333333	99779.555556	1301.666667	18056.222222
Csa00382s040_1	4.666667	4.222222	7.333333	2.888889
Csa00382s070_1	172.333333	1643.555556	186.333333	86.222222
Csa00382s080_1	40.000000	268.666667	32.666667	32.888889

In [113…
```
Mean_for_FC1 = Mean_for_FC.copy()
```

In [114…
```
Mean_for_FC1
```

Out[114]:

Sample	mean1	variance1	mean2	variance2
Csa00382s010_1	1106.666667	169526.222222	871.000000	558.000000
Csa00382s020_1	1180.333333	99779.555556	1301.666667	18056.222222
Csa00382s040_1	4.666667	4.222222	7.333333	2.888889
Csa00382s070_1	172.333333	1643.555556	186.333333	86.222222
Csa00382s080_1	40.000000	268.666667	32.666667	32.888889
...	...	...	...	...
Csa34454s010_1	8.333333	3.555556	12.333333	14.888889
Csa34686s010_1	2.000000	0.666667	5.000000	8.666667
Csa34903s010_1	3.000000	8.000000	1.666667	0.222222
Csa36836s010_1	14.000000	28.666667	6.000000	4.666667
Csa37955s010_1	28.666667	170.888889	29.000000	38.000000

61220 rows × 4 columns

In [120…
```
Mean_for_FC1['FC'] = Mean_for_FC1['mean2'] / Mean_for_FC1['mean1']
```

In [122…
```
Mean_for_FC1.head()
```

Out[122]:

Sample	mean1	variance1	mean2	variance2	FC
Csa00382s010_1	1106.666667	169526.222222	871.000000	558.000000	0.787048
Csa00382s020_1	1180.333333	99779.555556	1301.666667	18056.222222	1.102796
Csa00382s040_1	4.666667	4.222222	7.333333	2.888889	1.571429
Csa00382s070_1	172.333333	1643.555556	186.333333	86.222222	1.081238
Csa00382s080_1	40.000000	268.666667	32.666667	32.888889	0.816667

In [123…
```
# calculate log2FC
Mean_for_FC1['log2-FC'] = Mean_for_FC1['FC'].apply(np.log2)
```

In [124…
```
Mean_for_FC1
```

Out[124]:

Sample	mean1	variance1	mean2	variance2	FC	log2-F
Csa00382s010_1	1106.666667	169526.222222	871.000000	558.000000	0.787048	-0.3454
Csa00382s020_1	1180.333333	99779.555556	1301.666667	18056.222222	1.102796	0.1411
Csa00382s040_1	4.666667	4.222222	7.333333	2.888889	1.571429	0.6520
Csa00382s070_1	172.333333	1643.555556	186.333333	86.222222	1.081238	0.1126
Csa00382s080_1	40.000000	268.666667	32.666667	32.888889	0.816667	-0.2921
…	…	…	…	…	…	
Csa34454s010_1	8.333333	3.555556	12.333333	14.888889	1.480000	0.5655
Csa34686s010_1	2.000000	0.666667	5.000000	8.666667	2.500000	1.3219
Csa34903s010_1	3.000000	8.000000	1.666667	0.222222	0.555556	-0.8479
Csa36836s010_1	14.000000	28.666667	6.000000	4.666667	0.428571	-1.2223
Csa37955s010_1	28.666667	170.888889	29.000000	38.000000	1.011628	0.0166

61220 rows × 6 columns

select genes with log2FC > 2 (up regulated) and log2FC <= -2 (down regulated)

In [144…
```
Up_regulated_genes = Mean_for_FC1[Mean_for_FC1['log2-FC'] > 2]
```

In [146…
```
Up_regulated_genes.head()
```

Out[146]:

Sample	mean1	variance1	mean2	variance2	FC	log2-F
Csa00430s140_1	331.333333	8322.888889	2433.333333	22966.222222	7.344064	2.87657
Csa00595s010_1	2.666667	2.888889	12.000000	32.666667	4.500000	2.16992
Csa00603s020_1	4.000000	0.666667	201.000000	552.666667	50.250000	5.65105
Csa00855s010_1	3.000000	0.666667	399.000000	2648.000000	133.000000	7.05528
Csa01299s010_1	2.666667	2.888889	11.000000	34.666667	4.125000	2.04439

In [147…
```
Up_regulated_genes.shape
```

Out[147]:
```
(414, 6)
```

ascending sort of log2FC column and highlight the top 10 up regulated genes

In [162…
```
Up_regulated_genes.sort_values(by='log2-FC', ascending=False).head(10)
```

Out[162]:

Sample	mean1	variance1	mean2	variance2	FC	log2-FC
Csa00855s010_1	3.000000	0.666667	399.000000	2648.000000	133.000000	7.055282
Csa09g066290_1	11.000000	0.666667	1309.666667	9709.555556	119.060606	6.895552
Csa02g005800_1	1.000000	0.000000	101.333333	189.555556	101.333333	6.662965
Csa20g006840_1	4.333333	6.888889	411.333333	1550.888889	94.923077	6.568687
Csa20g052950_1	4.333333	2.888889	405.666667	353.555556	93.615385	6.548674
Csa02g012920_1	94.333333	830.888889	8506.666667	212474.888889	90.176678	6.494682
Csa15g003390_1	8.333333	10.888889	673.333333	3360.888889	80.800000	6.336283
Csa09g005380_1	3.000000	0.666667	223.666667	104.222222	74.555556	6.220244
Csa09g034110_1	3.000000	2.000000	202.333333	124.222222	67.444444	6.075628
Csa08g047140_1	9.333333	22.222222	606.000000	3386.000000	64.928571	6.020782

decending sort of log2FC column and highlight the top 10 down regulated genes

In [156…
```
Down_regulated_genes = Mean_for_FC1[Mean_for_FC1['log2-FC'] <= -2]
```

In [157…
```
Down_regulated_genes.head()
```

Out[157]:

Sample	mean1	variance1	mean2	variance2	FC	log2-FC
Csa00511s010_1	21.000000	34.666667	2.000000	0.666667	0.095238	-3.392317
Csa00511s020_1	310.333333	4648.222222	36.333333	66.888889	0.117078	-3.094453
Csa00532s070_1	6.333333	22.222222	1.333333	0.222222	0.210526	-2.247928
Csa00607s020_1	229.666667	2410.888889	30.333333	4.222222	0.132075	-2.920566
Csa00711s010_1	24.333333	121.555556	2.666667	2.888889	0.109589	-3.189825

In [161…
```
Down_regulated_genes.sort_values(by=['log2-FC']).head(10)
```

Out[161]:

Sample	mean1	variance1	mean2	variance2	FC	log2-FC
Csa10g035950_1	5953.333333	1.509090e+06	7.666667	24.888889	0.001288	-9.600883
Csa03g012470_1	405.000000	6.734000e+03	1.333333	0.222222	0.003292	-8.246741
Csa05g002410_1	2188.000000	1.692687e+05	9.333333	3.555556	0.004266	-7.873005
Csa01g041890_1	592.666667	2.460289e+04	2.666667	2.888889	0.004499	-7.796040
Csa09g004680_1	220.000000	4.468667e+03	1.333333	0.222222	0.006061	-7.366322
Csa09g076020_1	142.333333	6.935556e+02	1.000000	0.000000	0.007026	-7.153130
Csa02g001800_1	734.333333	3.951756e+04	5.666667	1.555556	0.007717	-7.017791
Csa19g014920_1	293.666667	3.036222e+03	2.333333	0.222222	0.007946	-6.975643
Csa02g030650_1	157.333333	4.264222e+03	1.333333	0.222222	0.008475	-6.882643
Csa13g048260_1	334.666667	3.200889e+03	3.000000	0.666667	0.008964	-6.801619

In [160…
```
Down_regulated_genes.shape
```

Out[160]:
```
(548, 6)
```

In [138…

Out[138]:

Sample
Csa00382s010_1
Csa00382s020_1
Csa00382s040_1
Csa00382s070_1
Csa00382s080_1
...
Csa34454s010_1
Csa34686s010_1
Csa34903s010_1
Csa36836s010_1
Csa37955s010_1

61220 rows × 0 columns

In [141…
```
# saving labels genes in a list
labels = list([Mean_for_FC1.iloc[:,:-6]])
```

In [142…
```
labels
```

Out[142]:
```
[Empty DataFrame
 Columns: []
 Index: [Csa00382s010_1, Csa00382s020_1, Csa00382s040_1, Csa00382s070_1, Csa0
0382s080_1, Csa00382s090_1, Csa00382s100_1, Csa00382s110_1, Csa00382s120_1, C
sa00382s140_1, Csa00382s210_1, Csa00382s220_1, Csa00382s230_1, Csa00382s240_
1, Csa00382s250_1, Csa00382s270_1, Csa00382s280_1, Csa00382s300_1, Csa00395s0
10_1, Csa00395s020_1, Csa00395s050_1, Csa00395s060_1, Csa00395s070_1, Csa0042
7s020_1, Csa00427s040_1, Csa00427s050_1, Csa00427s060_1, Csa00427s070_1, Csa0
0427s080_1, Csa00427s100_1, Csa00427s110_1, Csa00427s120_1, Csa00427s130_1, C
sa00430s010_1, Csa00430s020_1, Csa00430s030_1, Csa00430s040_1, Csa00430s060_
1, Csa00430s090_1, Csa00430s100_1, Csa00430s110_1, Csa00430s130_1, Csa00430s1
40_1, Csa00441s010_1, Csa00441s020_1, Csa00441s030_1, Csa00441s040_1, Csa0044
1s050_1, Csa00441s060_1, Csa00441s070_1, Csa00441s080_1, Csa00441s090_1, Csa0
0441s100_1, Csa00441s110_1, Csa00441s120_1, Csa00441s130_1, Csa00441s160_1, C
sa00441s170_1, Csa00441s180_1, Csa00441s200_1, Csa00441s210_1, Csa00441s230_
1, Csa00441s240_1, Csa00441s250_1, Csa00441s260_1, Csa00441s270_1, Csa00441s2
80_1, Csa00441s290_1, Csa00441s300_1, Csa00441s380_1, Csa00441s390_1, Csa0044
1s400_1, Csa00441s410_1, Csa00456s010_1, Csa00456s020_1, Csa00456s070_1, Csa0
0456s080_1, Csa00456s090_1, Csa00456s100_1, Csa00456s130_1, Csa00456s140_1, C
sa00462s010_1, Csa00462s020_1, Csa00462s030_1, Csa00462s040_1, Csa00462s050_
1, Csa00462s060_1, Csa00462s070_1, Csa00462s080_1, Csa00462s100_1, Csa00462s1
10_1, Csa00470s020_1, Csa00470s030_1, Csa00470s040_1, Csa00474s010_1, Csa0047
4s020_1, Csa00474s030_1, Csa00474s060_1, Csa00474s070_1, Csa00474s080_1, ...]

 [61220 rows x 0 columns]]
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: