# Experiments1_Nico

October 11, 2023

This document does in essence not differ a lot form Experiments1_Gernot.pdf

It merely contains the same plots but serves as a demostration for the data_lib library added with this document. Moreover there is one new example of plotted labels. See below the `pairwise_plots_labels` function.

[ ]:
```
import numpy as np
import pandas as pd
import sklearn.cluster as cl
import matplotlib.pyplot as plt
import itertools
from typing import Dict
import data_lib
```

```
--------------------------------------------------------------------------------
--------------
-- The following 0 groups were found
-- They contain 0 datasets
-- The first printed entity is the key to the returned dictionary
-----------------------------------
```

[ ]:
```
# print available data summary
_ = data_lib.explore_datasets(datafolder="../Data",verbose=True)
```

```
--------------------------------------------------------------------------------
--------------
-- The following 4 groups were found
-- They contain 40 datasets
-- The first printed entity is the key to the returned dictionary
-----------------------------------
Group: ../Data/6P-positive-dilution-series-2-labelled/droplet-level-data/RawData
po-di-se-2-A4, files: 13                po-di-se-2-C4, files: 13
po-di-se-2-A1, files: 13
po-di-se-2-B1, files: 13                po-di-se-2-D1, files: 13
po-di-se-2-B4, files: 13
-----------------------------------
Group: ../Data/6P-positive-dilution-series-1-labelled/droplet-level-data/RawData
po-di-se-1-D4, files: 13                po-di-se-1-A4, files: 13
po-di-se-1-A1, files: 13
po-di-se-1-D1, files: 13                po-di-se-1-B1, files: 13
```

```
po-di-se-1-C1, files: 13
-----------------------------------
Group: ../Data/6P-positive-dilution-series-labelled/droplet-level-data/RawData
po-di-se-B8, files: 13                  po-di-se-A8, files: 13
po-di-se-C8, files: 13
-----------------------------------
Group: ../Data/6P-wastewater-samples-labelled/droplet-level-data/RawData
wa-sa-A2, files: 13                     wa-sa-B4, files: 13
wa-sa-C5, files: 13
wa-sa-C4, files: 13                     wa-sa-B3, files: 13
wa-sa-B2, files: 13
wa-sa-A5, files: 13                     wa-sa-A3, files: 13
wa-sa-C2, files: 13
wa-sa-C3, files: 13                     wa-sa-D3, files: 13
wa-sa-D4, files: 13
wa-sa-B1, files: 13                     wa-sa-A4, files: 13
wa-sa-A1, files: 13
wa-sa-D2, files: 13                     wa-sa-D5, files: 13
wa-sa-C1, files: 13
-----------------------------------
```

[ ]:
```python
def pairwise_plots_labels(df, num_clusters: int):
    np_features = df.to_numpy()
    preds = np_features[:, 6]

    combinations = itertools.combinations(df.columns[:6], 2)
    fig, ax = plt.subplots(5, 3, sharex=False, sharey=False)
    fig.set_figheight(15)
    fig.set_figwidth(15)
    for i, combination in enumerate(combinations):
        np_features = df.loc[:, combination]
        np_features = np_features.to_numpy()

        ax[i //3, i %3].set_xlabel(combination[0])
        ax[i //3, i %3].set_ylabel(combination[1])
        ax[i //3, i %3].scatter(np_features[:, 0], np_features[:, 1], c = preds)
    fig.tight_layout()
```

[ ]:
```python
def pairwise_plots_knn(df, num_clusters: int):
    np_features = df.to_numpy()
    classifier = cl.KMeans(num_clusters)
    preds = classifier.fit_predict(np_features)

    combinations = itertools.combinations(df.columns, 2)
    fig, ax = plt.subplots(5, 3, sharex=False, sharey=False)
    fig.set_figheight(15)
    fig.set_figwidth(15)
```

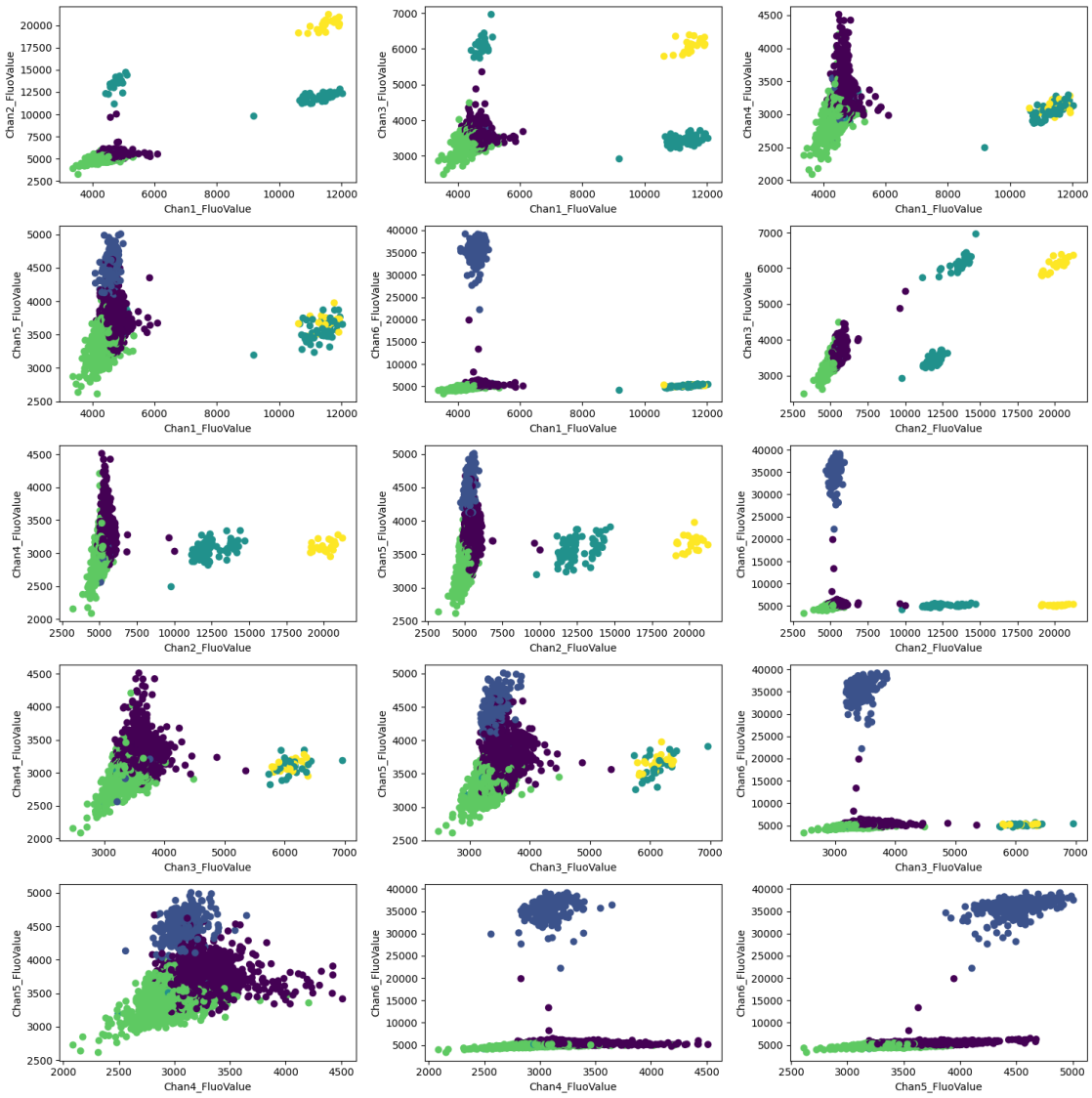
```
    for i, combination in enumerate(combinations):
        np_features = df.loc[:, combination]
        np_features = np_features.to_numpy()

        ax[i //3, i %3].set_xlabel(combination[0])
        ax[i //3, i %3].set_ylabel(combination[1])
        ax[i //3, i %3].scatter(np_features[:, 0], np_features[:, 1], c = preds)
    fig.tight_layout()
```

# 1 KNN On A3 Dataset

[ ]:
```
# num_clusters = 5
df = data_lib.load_dataset([], ["wa-sa-A3"],"../Data")
pairwise_plots_knn(df, num_clusters = 5)
```

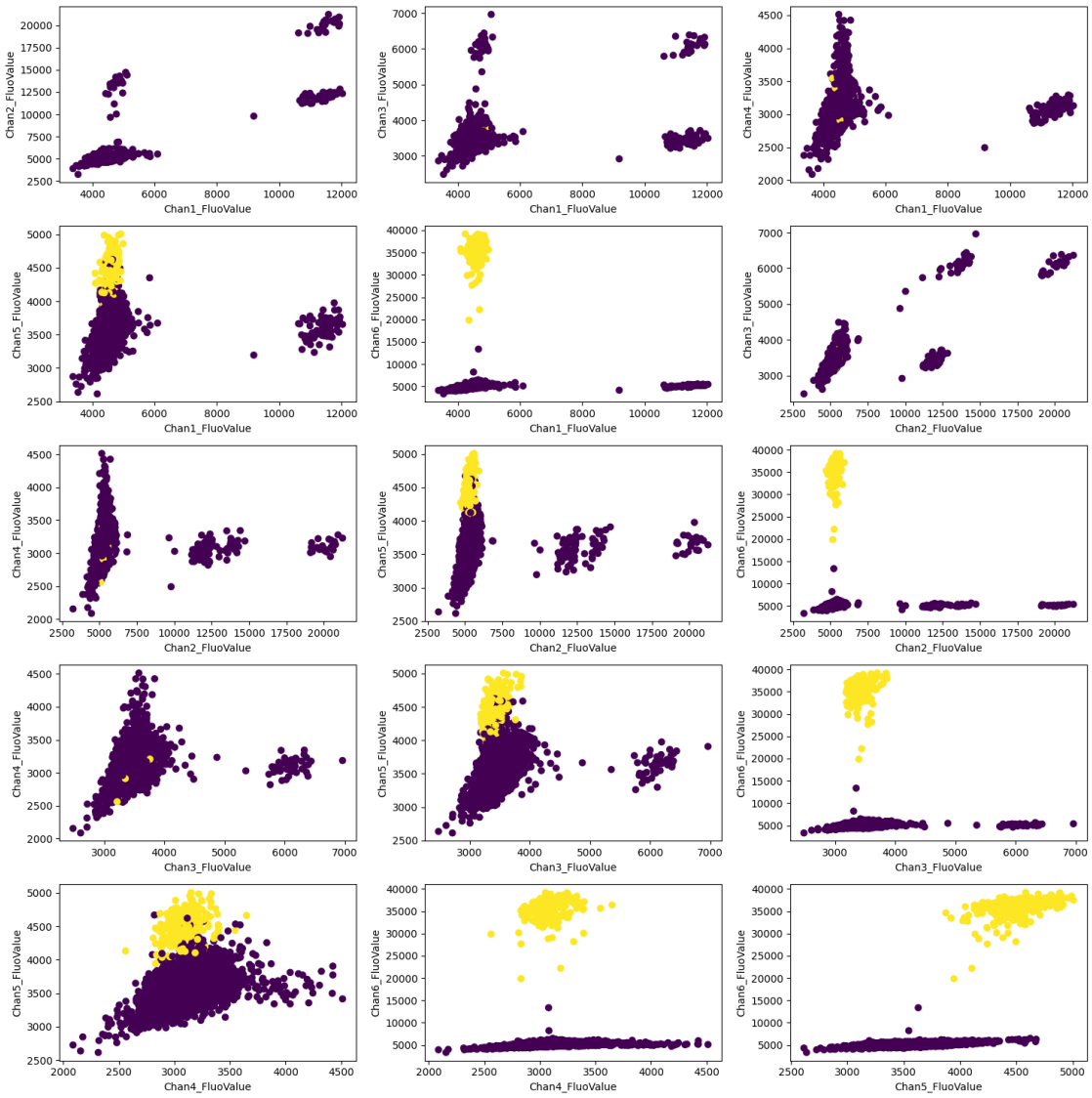
```
(18950, 6)
```

```
/home/nico/.cache/pypoetry/virtualenvs/ds-lab-4Qf2VVQw-
py3.11/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1416:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```




[ ]:
```
df = data_lib.load_dataset([data_lib.LABELS_LIST[4]], ["wa-sa-A3"],"../Data")
pairwise_plots_labels(df, num_clusters = 5)
```

```
(18950, 7)
```



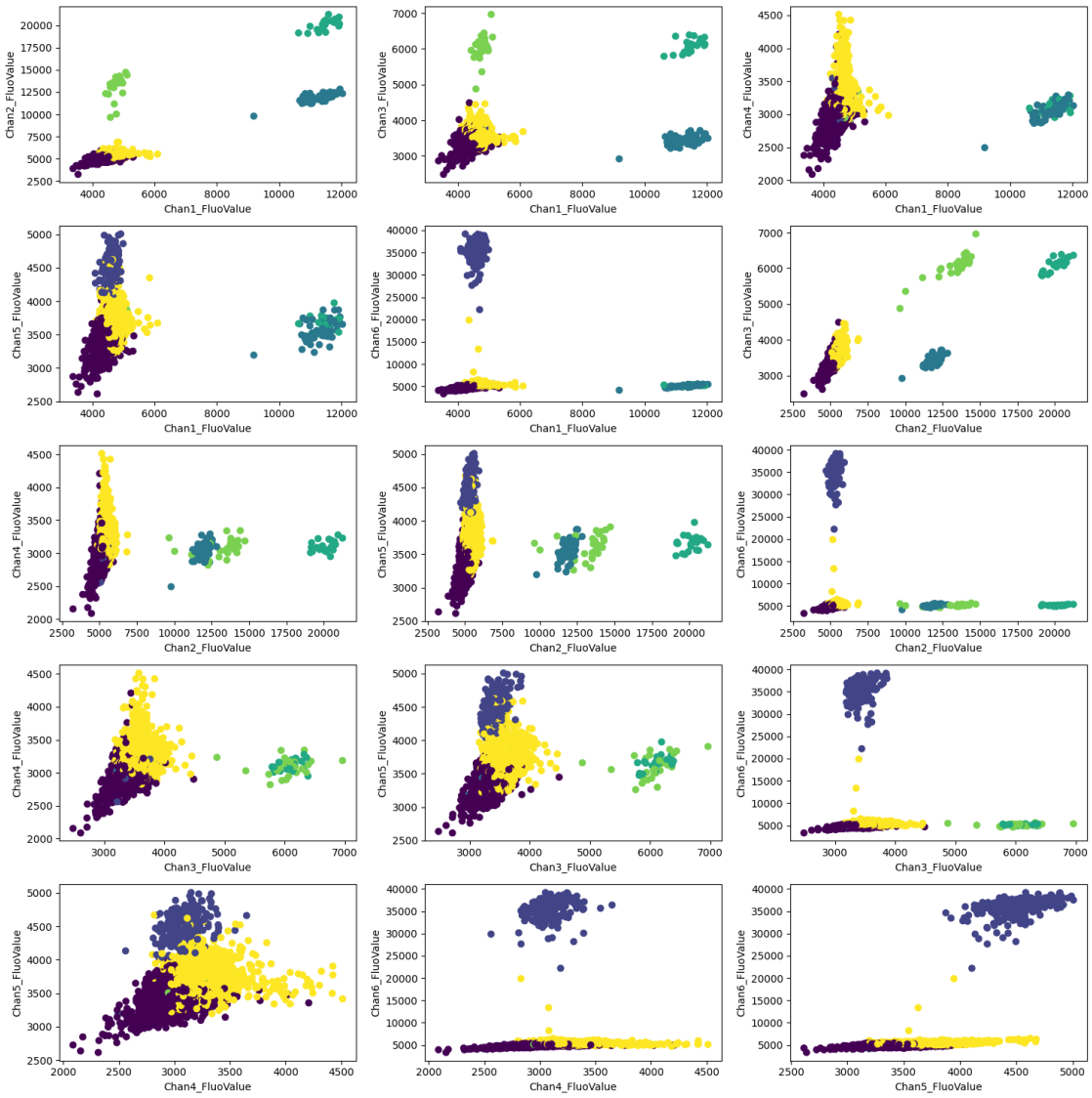

We can see by looking at the data that there are 5 clusters. However, when setting 5 clusters, instead of finding the fifth cluster, it instead makes the negatives into two clusters. Why is that? I looked and the data and it is because somehow if there is fluctation in one dimension in the negative data, it is in every dimension. Hence the difference in 6 dimensions is much bigger than can be seen in the 2 dimensional plots. But probably they should still not be considered as 2 clusters.

[ ]:
```
# num_clusters = 6
pairwise_plots_knn(df, num_clusters = 6)
```

```
(18950, 6)
```

```
/home/nico/.cache/pypoetry/virtualenvs/ds-lab-4Qf2VVQw-
py3.11/lib/python3.11/site-packages/sklearn/cluster/_kmeans.py:1416:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
```

```
1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```




By setting num_clusters = 6 we also find the last cluster!

## 2 DB Scan

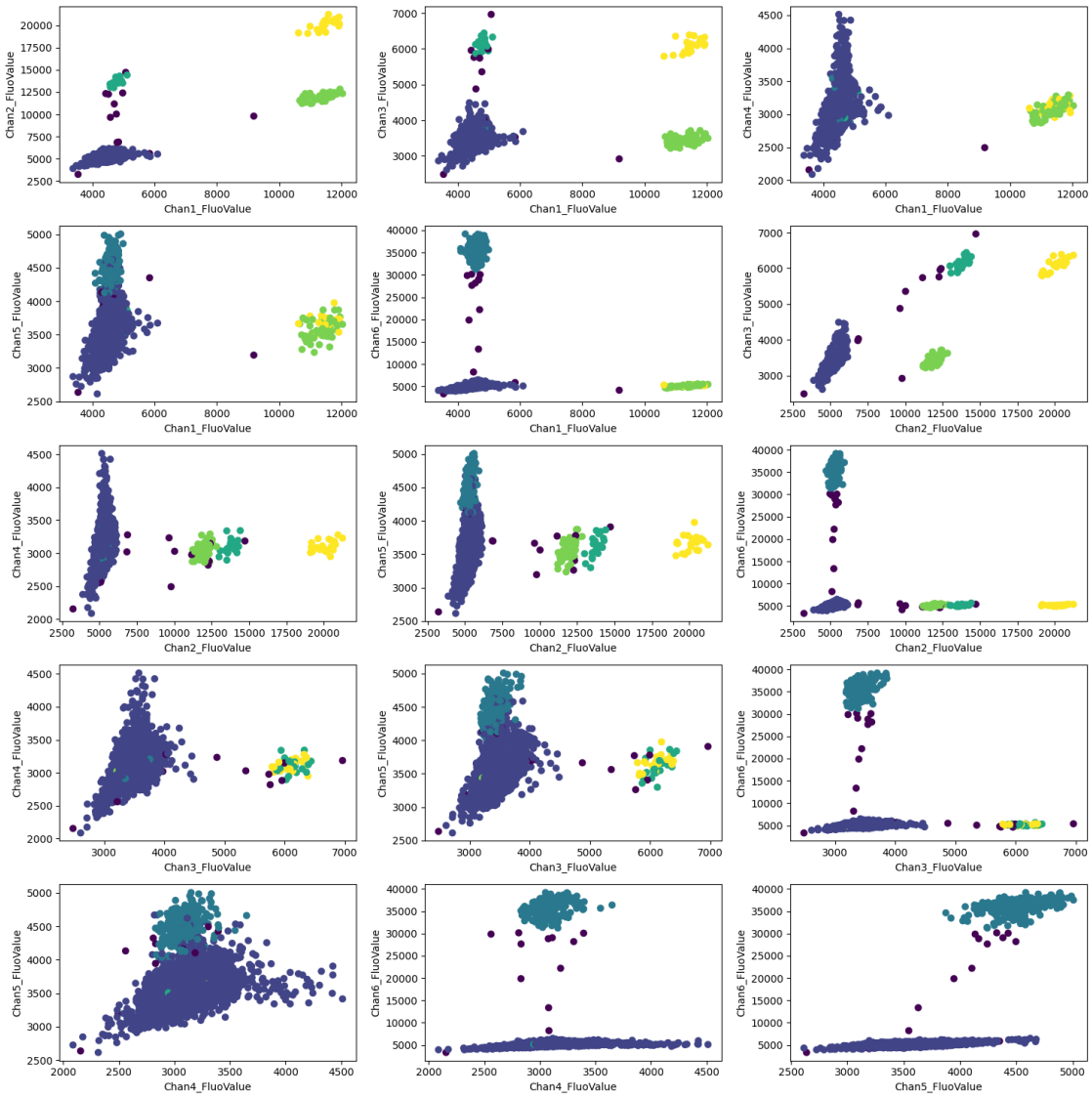
[ ]:
```
def pairwise_plots_dbscan(df, eps, min_samples = 5):
    np_features = df.to_numpy()
    classifier = cl.DBSCAN(eps = eps, min_samples = min_samples)
    preds = classifier.fit_predict(np_features)
    print(preds.shape)
```

```
    print(f"Number of outliers: {len([x for x in preds if x == -1])}") # print number of outliers
    print(f"Number of clusters: {max(preds)+1}") # print number of clusters


    combinations = itertools.combinations(df.columns, 2)
    fig, ax = plt.subplots(5, 3, sharex=False, sharey=False)
    fig.set_figheight(15)
    fig.set_figwidth(15)
    for i, combination in enumerate(combinations):
        np_features = df.loc[:, combination]
        np_features = np_features.to_numpy()

        ax[i //3, i %3].set_xlabel(combination[0])
        ax[i //3, i %3].set_ylabel(combination[1])
        ax[i //3, i %3].scatter(np_features[:, 0], np_features[:, 1], c = preds)
    fig.tight_layout()
```

### 2.0.1 DB Scan on A3 Dataset

[ ]:
```
df = data_lib.load_dataset([], ["wa-sa-A3"], "../Data")
pairwise_plots_dbscan(df, eps = 700, min_samples = 5)
```

```
Number of outliers: 23
Number of clusters: 5
```

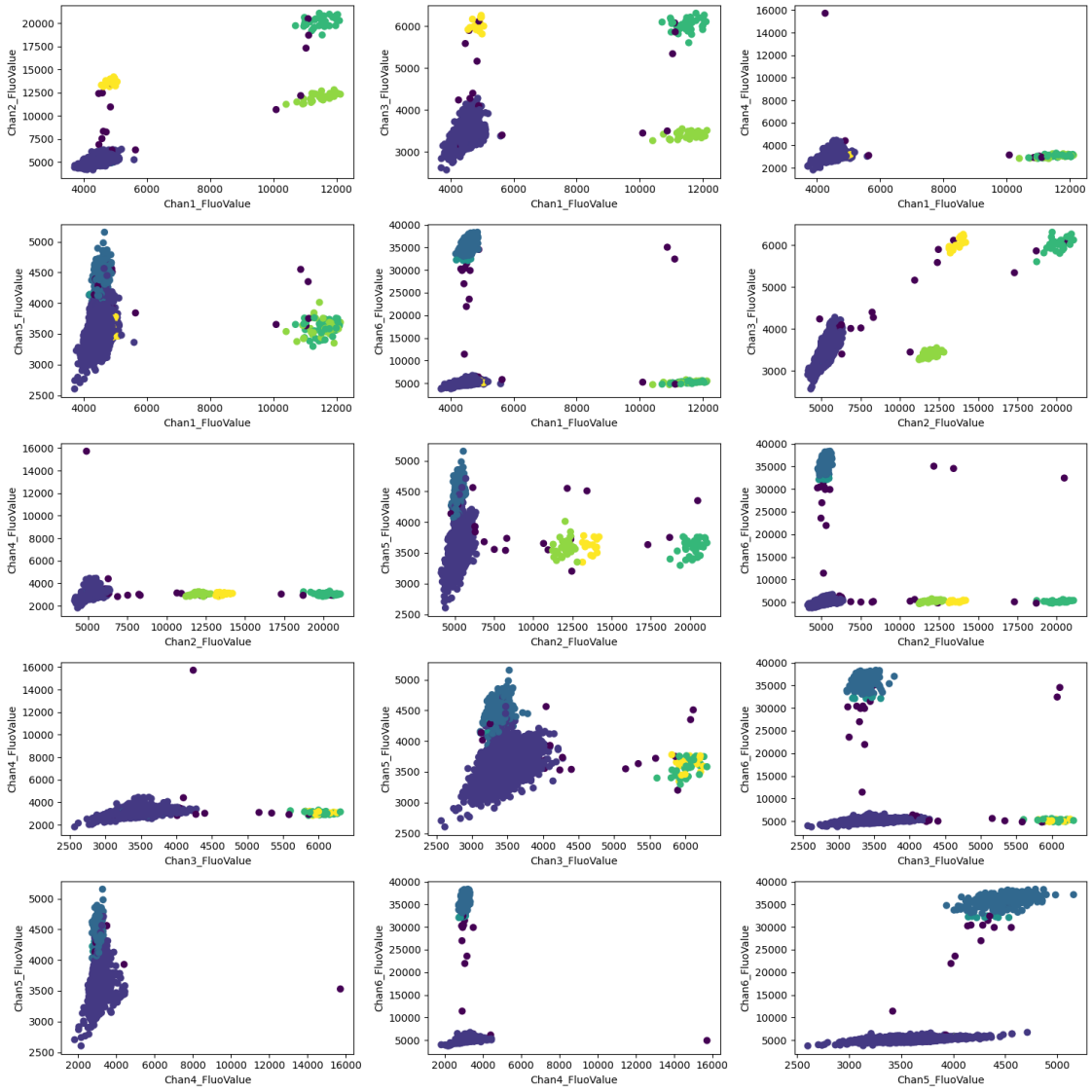



We see here the classification is completely correct. We even classified the outliers correctly. The only thing one might say is that a few too many points got classified as outliers, but that is debatable.

### 2.0.2 DB Scan on B3 Dataset

[ ]:
```
df = data_lib.load_dataset([], ["wa-sa-B3"], "../Data")
pairwise_plots_dbscan(df, eps = 700, min_samples = 5)
```

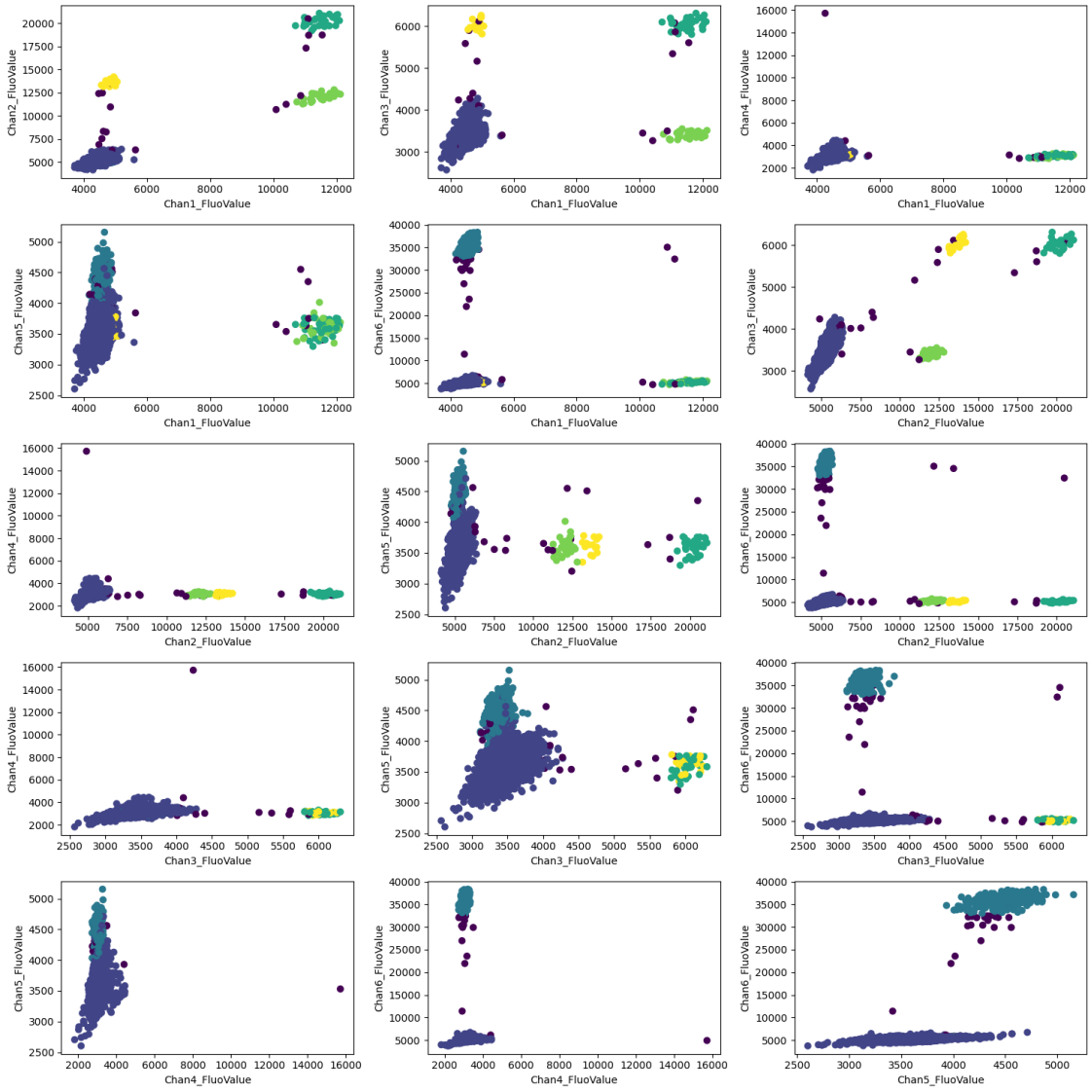
```
Number of outliers: 27
Number of clusters: 6
```




Here we have a very similar situation to the previous dataset, but somehow an extra cluster emerges, but this cluster has only very few points.

[ ]:
```
df = data_lib.load_dataset([], ["wa-sa-B3"], "../Data")
pairwise_plots_dbscan(df, eps = 700, min_samples = 10)
```

```
Number of outliers: 38
Number of clusters: 5
```

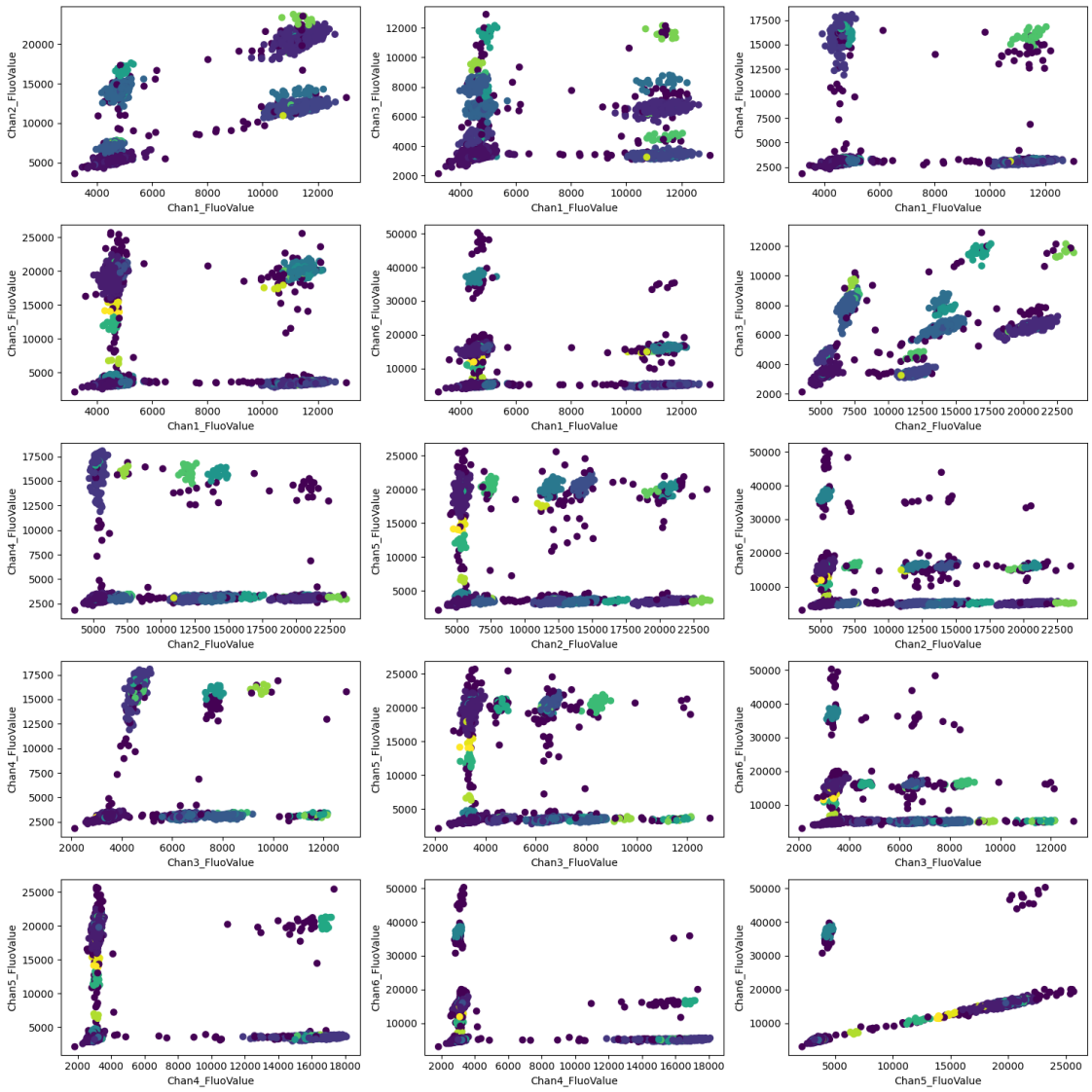



At the cost of increasing the number of outliers, we can also get the correct number of clusters again (this is managed by the min_samples parameter, which controls how many points a cluster needs to contain at least.)

### 2.0.3 DB Scan on C3 Dataset

[ ]:
```
df = data_lib.load_dataset([], ["wa-sa-C3"], "../Data")
pairwise_plots_dbscan(df, eps = 700, min_samples = 5)
```

```
Number of outliers: 229
Number of clusters: 25
```



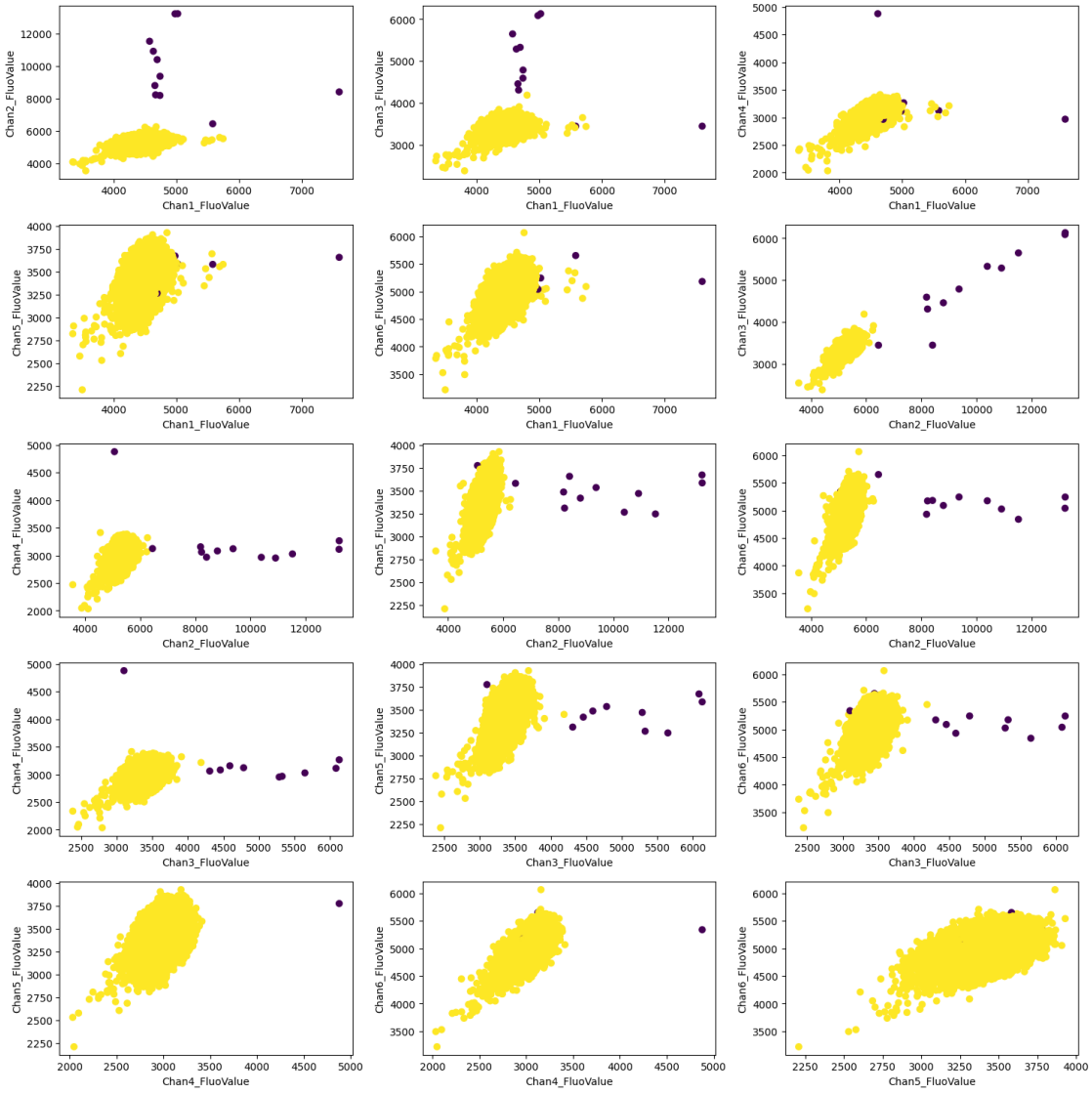

OOoops! This data is much more messy! We have 229 outliers and 24 clusters. The data is so complicated it is not obvious how many clusters this data actually contains. I think it would be really helpful if we had some sense of how many clusters there are (it is possible to get an estimate by doing the manual procedure in 2 dimensions just as they have done it).

I did some basic quality control checks to see if the clustering makes sense and at least it does not make obvious errors: For example, no point that has a value of more than 8000 in dimension 1 has the same label as a point that has a value of less than 8000 in dimension 1 (see the pictures to see why this makes sense and is desired). It is similar for all the other obvious separations I have checked.

### 2.0.4 DB Scan on D3 Dataset

```
[ ]: df = data_lib.load_dataset([], ["wa-sa-D3"], "../Data")
     pairwise_plots_dbscan(df, eps = 700, min_samples = 5)
```

```
Number of outliers: 12
Number of clusters: 1
```

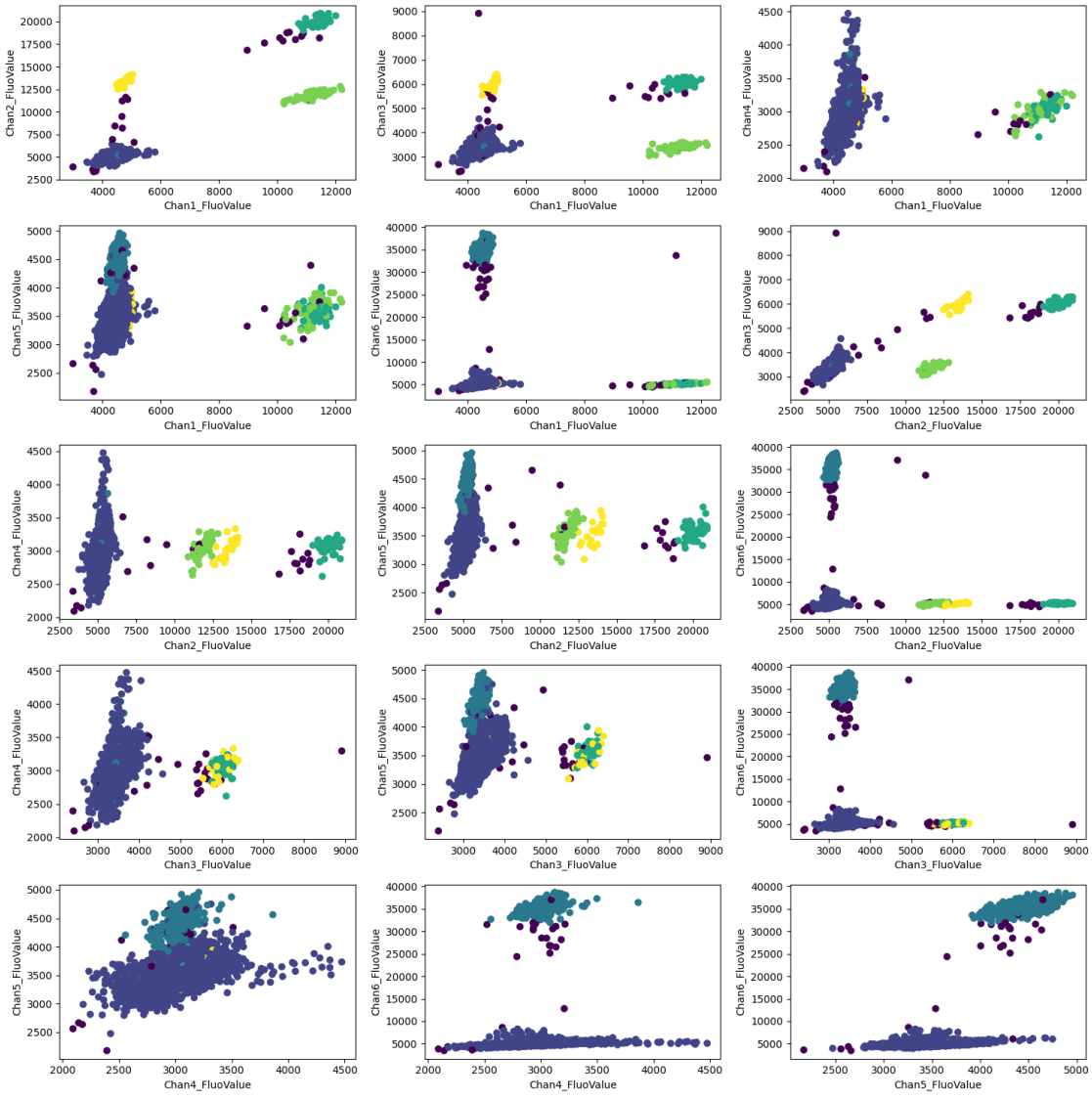



This data seems to be just negatives, and they are classified as so, which is desirable.

### 2.0.5 DB Scan on A2 Dataset

[ ]:
```
df = data_lib.load_dataset([], ["wa-sa-A2"], "../Data")
pairwise_plots_dbscan(df, eps = 700, min_samples = 10)
```

```
Number of outliers: 43
Number of clusters: 5
```

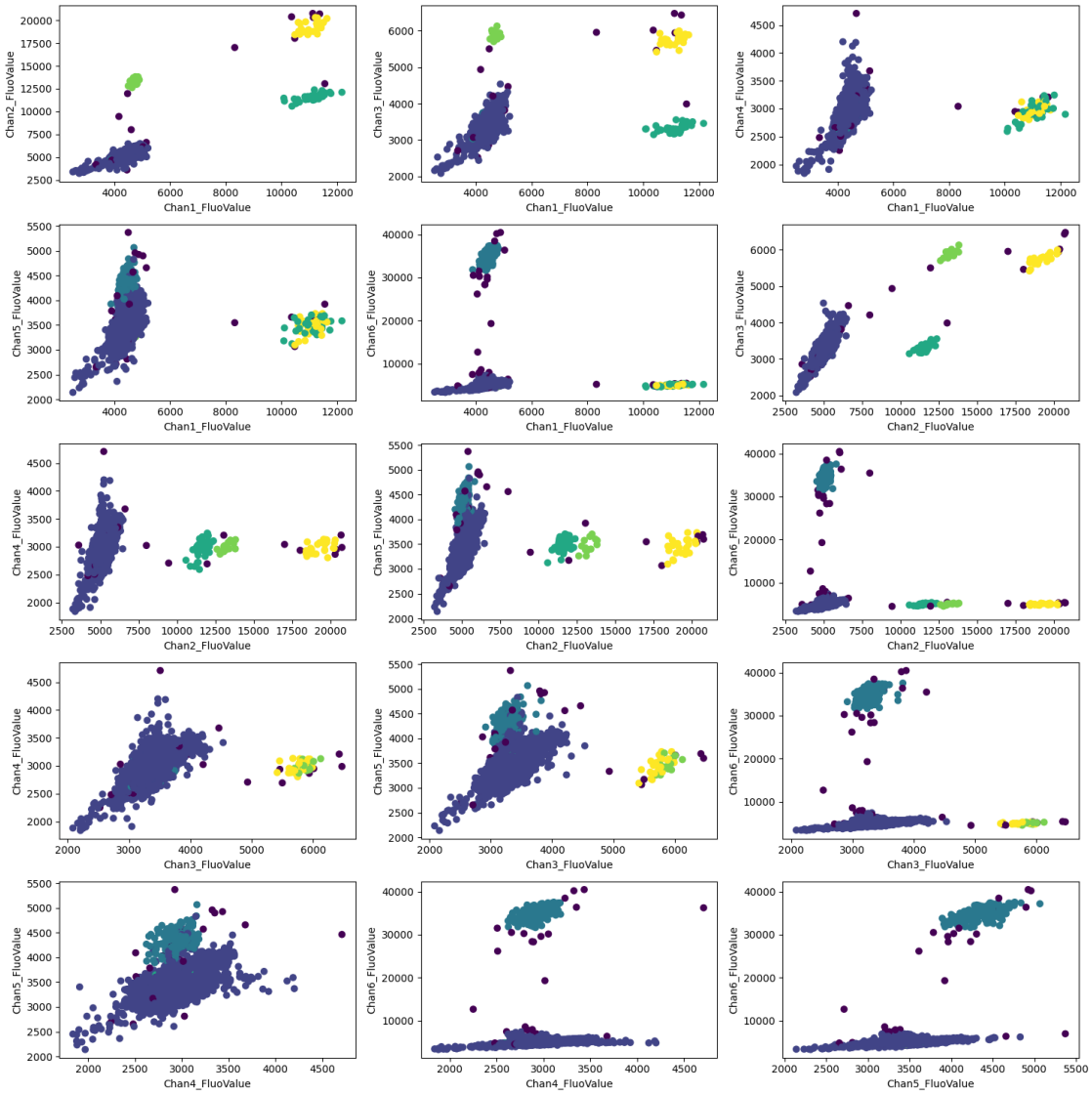



Also very OK clustering, though kind of a lot of outliers.

### 2.0.6 DB Scan on B2 Dataset

[ ]:
```
df = data_lib.load_dataset([], ["wa-sa-B2"], "../Data")
pairwise_plots_dbscan(df, eps = 700, min_samples = 10)
```

```
Number of outliers: 34
Number of clusters: 5
```

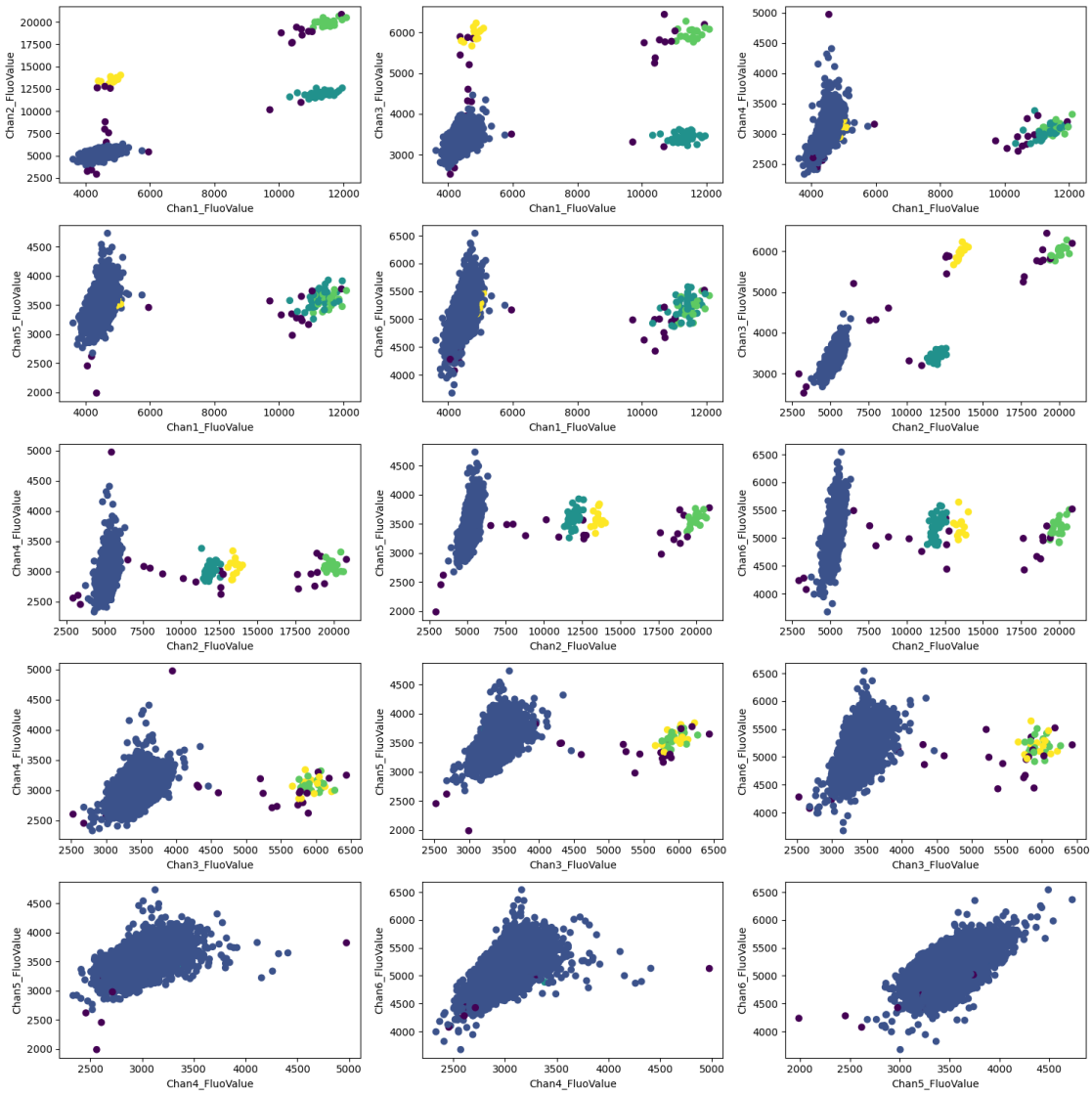



### 2.0.7 DB Scan on C2 Dataset

[ ]:
```
df = data_lib.load_dataset([], ["wa-sa-C2"], "../Data")
pairwise_plots_dbscan(df, eps = 700, min_samples = 10)
```

```
Number of outliers: 24
```

```
Number of clusters: 4
```

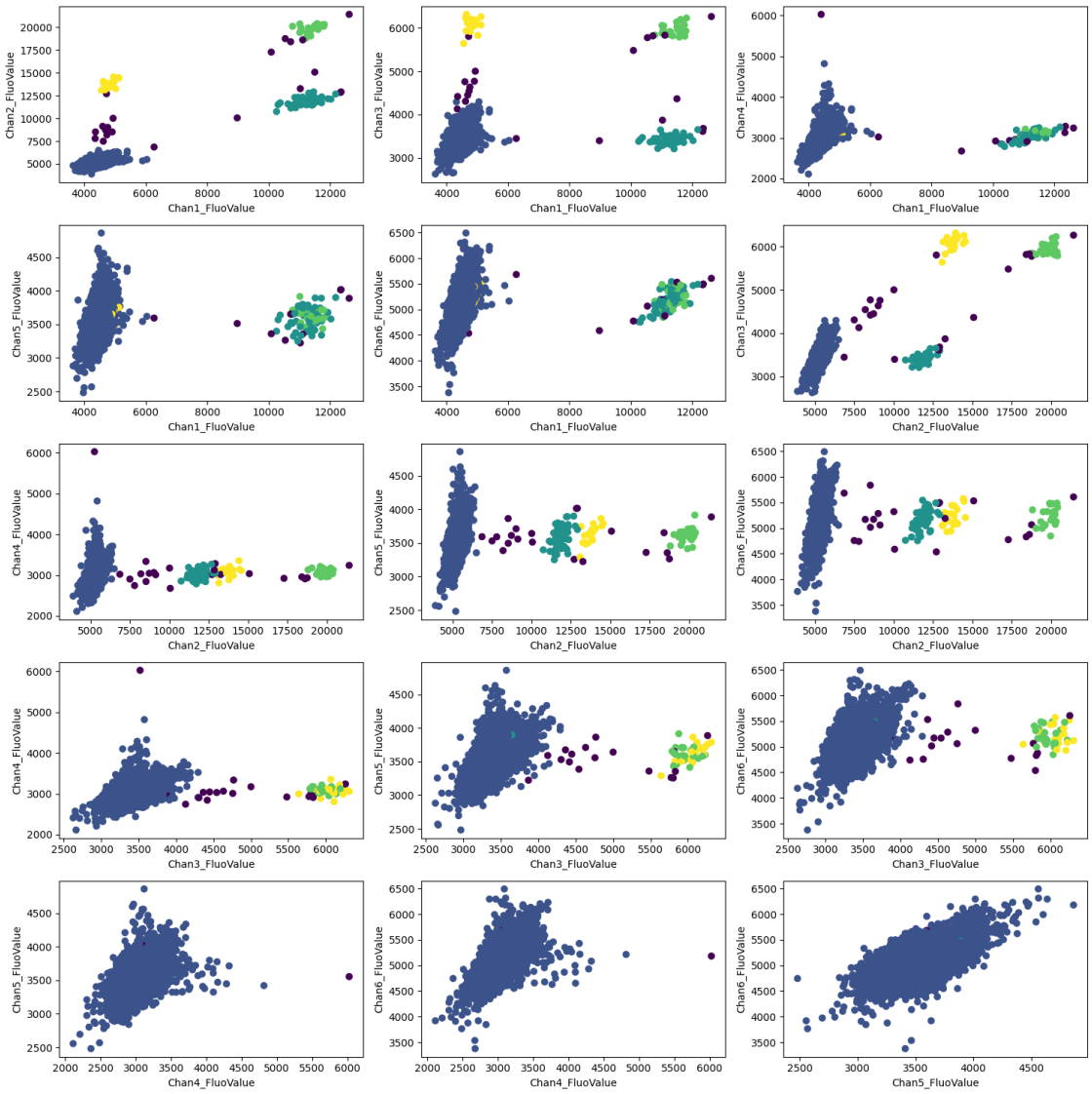



Interesting: First time we have only 4 clusters, but the number is again correct.

### 2.0.8 DB Scan on D2 Dataset

[ ]:
```
df = data_lib.load_dataset([], ["wa-sa-D2"], "../Data")
pairwise_plots_dbscan(df, eps = 700, min_samples = 10)
```

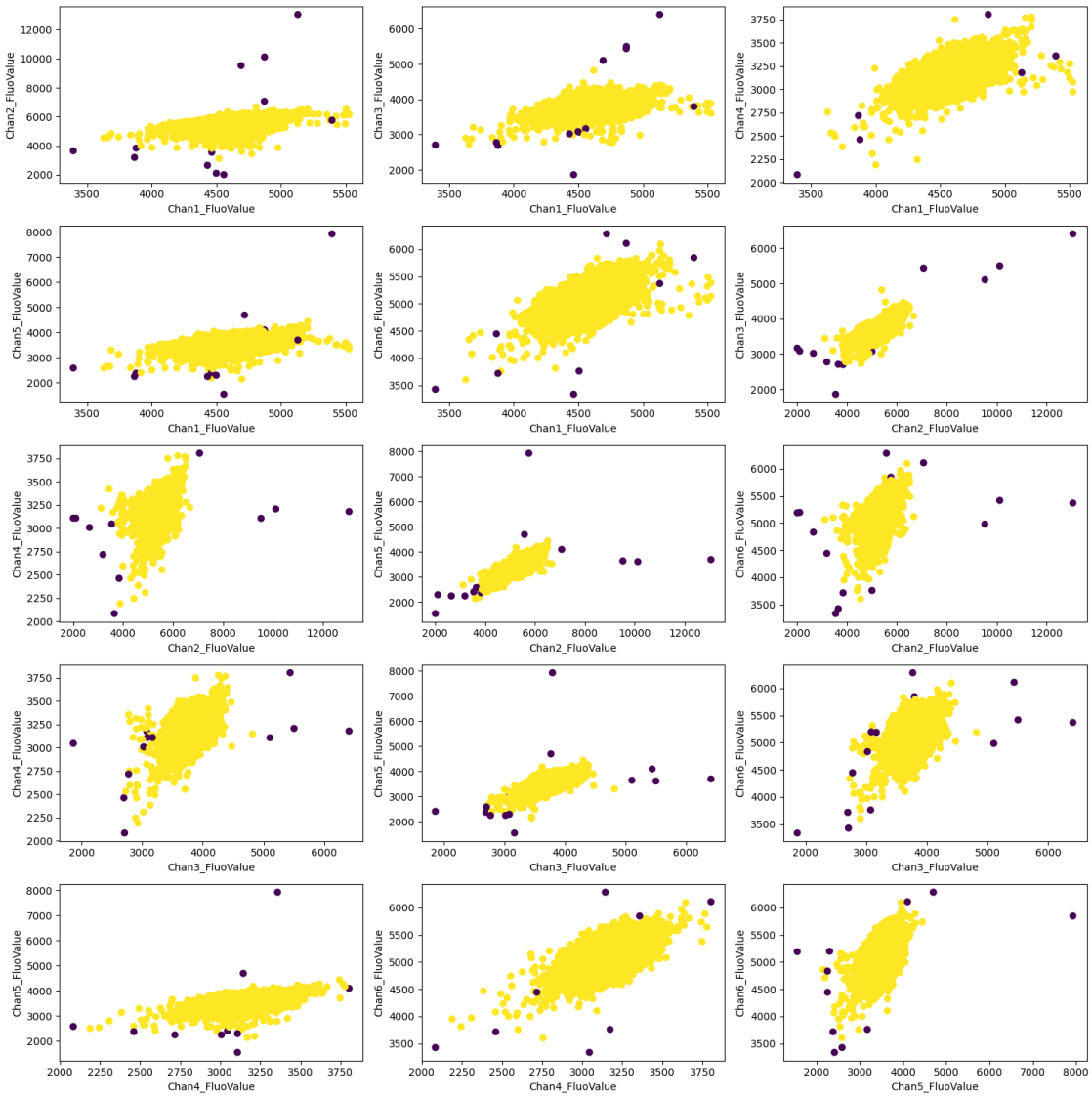
```
Number of outliers: 22
Number of clusters: 4
```




## 2.1 Dillution Series Data

[ ]:
```
df = data_lib.load_dataset([], ["wa-sa-A4"], "../Data")
pairwise_plots_dbscan(df, eps = 700, min_samples = 10)
```

```
Number of outliers: 14
Number of clusters: 1
```




WTF is going on here...??

# 3 Remarks and Ideas to Try Out

First thing is to consider the different data we have and look whats going on there. For example with the Dillution Series Data above - what is going on?

One thing I think we need to do is to ask McLeod what the clusters will exactly be used for. As we can see from previous examples, some data may have way too many clusters, meaning that it wont be easy for us to understand what the clusters do. If we want to do classification afterwards, then why clusters at all and not just do supervised training? I guess if we just need to find out whether or not a dimension is activated or not, we can try some tricks with the clusters.

One thing I want to try out is to decorrelate the data, and then cluster the de-correlated data. We

know there is quite simple linear correlation going on, so maybe decorrelation should help. In fact, I think I will try it right now and see if it takes a lot of time or not.

And then using different clustering algorithms and doing more data exploration might help a lot too!

### 3.0.1 Questions Nico

The labelled data only contains exactly 6 classes for each sample, should the algorithm be able to only distinguish for those 6 classes whether a point belongs to the class or not? (Note that I only count the POS files as classes, because NEG contains RawData - POS)

For later, what API do they have in mind (Not important in the begining now, but as I worked with finding the data I was wondering how their data will look like when they want to use the algorithm later)

[ ]:
```
df = data_lib.load_dataset([], ["wa-sa-A4"], "../Data")
```

[ ]:
```
from sklearn.decomposition import PCA
pca = PCA(whiten=True)
```

[ ]:
```
eps = 0.05
min_samples = 10

np_features = df.to_numpy()

np_features = pca.fit_transform(np_features)

classifier = cl.DBSCAN(eps = eps, min_samples = min_samples)
preds = classifier.fit_predict(np_features)

print(f"Number of outliers: {len([x for x in preds if x == -1])}") # print␣
 ↪number of outliers
print(f"Number of clusters: {max(preds)+1}") # print number of clusters


"""combinations = itertools.combinations(df.columns[2:-1], 2)
fig, ax = plt.subplots(5, 3, sharex=False, sharey=False)
fig.set_figheight(15)
fig.set_figwidth(15)
for i, combination in enumerate(combinations):
    np_features = df.loc[:, combination]
    np_features = np_features.to_numpy()

    ax[i //3, i %3].set_xlabel(combination[0])
    ax[i //3, i %3].set_ylabel(combination[1])
    ax[i //3, i %3].scatter(np_features[:, 0], np_features[:, 1], c = preds)
fig.tight_layout()"""
```

```
Number of outliers: 25432
Number of clusters: 0
```

```
[ ]: 'combinations = itertools.combinations(df.columns[2:-1], 2)\nfig, ax =
plt.subplots(5, 3, sharex=False,
sharey=False)\nfig.set_figheight(15)\nfig.set_figwidth(15)\nfor i, combination
in enumerate(combinations): \n    np_features = df.loc[:, combination]\n
np_features = np_features.to_numpy()\n\n    ax[i //3, i
%3].set_xlabel(combination[0])\n    ax[i //3, i %3].set_ylabel(combination[1])\n
ax[i //3, i %3].scatter(np_features[:, 0], np_features[:, 1], c =
preds)\nfig.tight_layout()'
```

need to look further into how to decorrelate properly.