Single-cell Morphology Quality Control (coSMicQC)

Dave Bunten[1]*, Jenna Tomkinson[1]*, Gregory Way[1]

[1]*Department of Biomedical Informatics, University of Colorado Anschutz Medical Campus*
**These authors contributed equally to this work.*

Scan for coSMicQC!

Department of Biomedical Informatics
SCHOOL OF MEDICINE
UNIVERSITY OF COLORADO **ANSCHUTZ MEDICAL CAMPUS**

I. Erroneous outliers and analysis

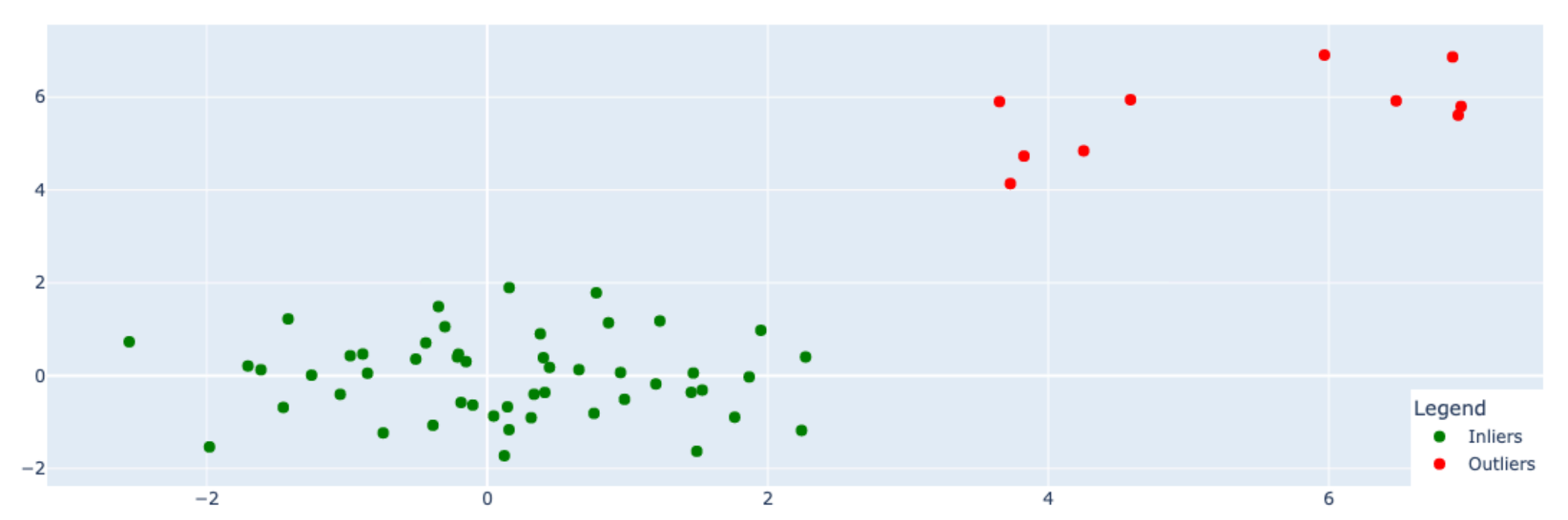


Figure 1: *Erroneous outlier anomalies can measure outside our expectations and impact analysis.*

Single-cell morphology data from high-throughput microscopy provide critical insights into disease mechanisms and therapeutic efficacy. However, ***segmentation errors*** during image analysis such as misidentifying cell compartments or artifacts as cells can lead to inaccurate single-cell measurements and ***erroneous anomalies*** within the data.

Researchers often resort to ***error-prone, bespoke filtering methods*** or aggregate data into bulk profiles to avoid discrepancies caused by anomaly outliers. These techniques fail to perform ***quality control*** on the data, often compromising the quality of single-cell profiles and impeding the potential for meaningful discoveries.

II. Single-cell quality control package

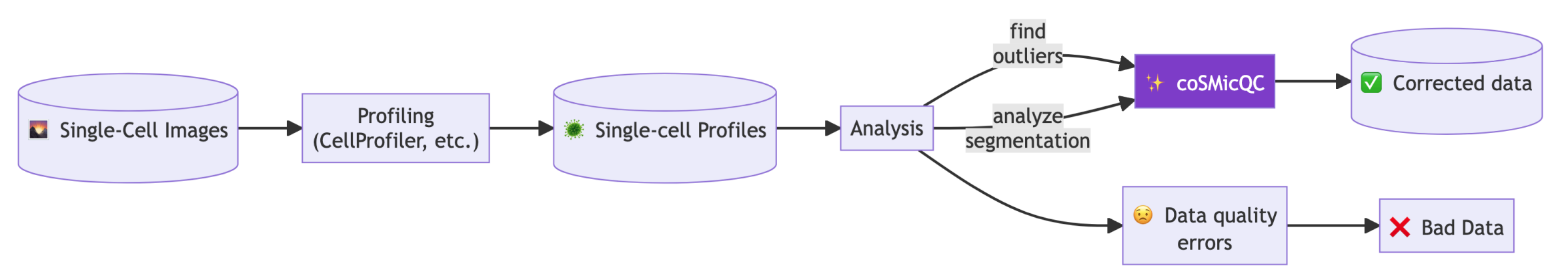


Figure 2: *coSMicQC enables high-quality data outcomes by checking for outliers.*

To address these challenges, we introduce ***`coSMicQC` (Single-cell Morphology Quality Control)***, an open source Python package designed to enhance the accuracy of single-cell morphology analysis. **`coSMicQC`** offers default and customizable thresholds for quality control, integrating seamlessly into both command line and Python API workflows.

`coSMicQC` features interactive visualizations that help users identify outlier distributions, and it introduces the ***CytoDataFrame*** — a novel data format that links single-cell measurements with their corresponding images and segmentation masks in real-time, enriching data analysis and interpretation.

III. Getting started with coSMicQC

☆ 1) *Installation*

```
# pip install from pypi
pip install coSMicQC

# install directly from source
pip install git+https://github.com/
WayScience/coSMicQC.git
```

`coSMicQC` may be installed from PyPI or source.

☆ 2) *Finding outliers*

```
import cosmicqc

# load a parquet file with single-cell
profile data and find outliers
scdf = cosmicqc.analyze.find_outliers(
    df="single-cell-profiles.parquet",
    metadata_columns=[
      "Metadata_ImageNumber",
      "Image_Metadata_Plate_x"
    ],
    feature_thresholds={
      "Nuclei_AreaShape_Area": -1
    },
)
```

```
Number of outliers: 328
Outliers Range:
Nuclei_AreaShape_Area Min: 734.0
Nuclei_AreaShape_Area Max: 1904.0
      Nuclei_AreaShape_Area  Metadata_ImageNumber Image_Metadata_Plate_x
23                    921.0                     2                Plate_2
28                    845.0                     2                Plate_2
29                   1024.0                     2                Plate_2
32                    787.0                     2                Plate_2
37                   1347.0                     2                Plate_2
...                     ...                   ...                    ...
```

Figure 3: *Reports in **coSMicQC** provide quality control feedback for analysis.*

The `find_outliers` function in **coSMicQC** uses single-cell feature thresholds to provide a report on how many outliers were detected. We use ***z-scores*** to help define thresholds used throughout coSMicQC. `find_outliers` may be used through Python or CLI (see below).

```
# CLI interface for coSMicQC find_outliers
$ cosmicqc find_outliers \
  --df single-cell-profiles.parquet \
  --metadata_columns \[Metadata_ImageNumber\]
\
  --feature_thresholds
'{"Nuclei_AreaShape_Area": -1}'

Number of outliers: 328
Outliers Range:
Nuclei_AreaShape_Area Min: 734.0
Nuclei_AreaShape_Area Max: 1904.0
...
```

☆ 3) *Visualizing outlier distributions*

```
import cosmicqc

# label outliers within the profiles
scdf = cosmicqc.analyze.label_outliers(
    df="single-cell-profiles.parquet",
    include_threshold_scores=True,
# show outlier histogram plots
).show_report()
```

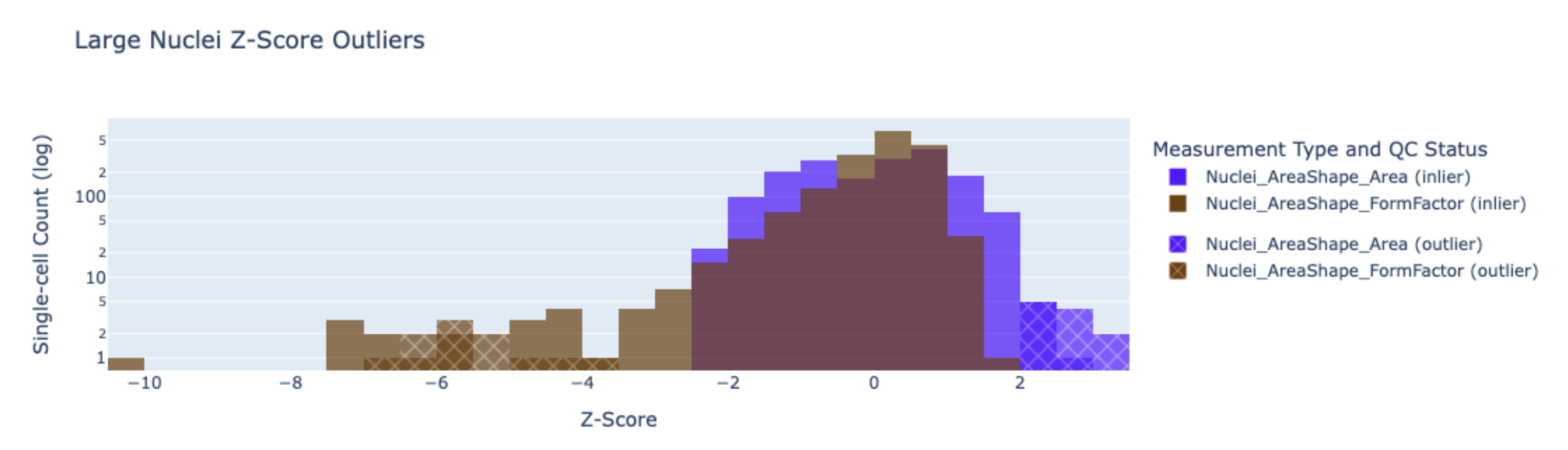


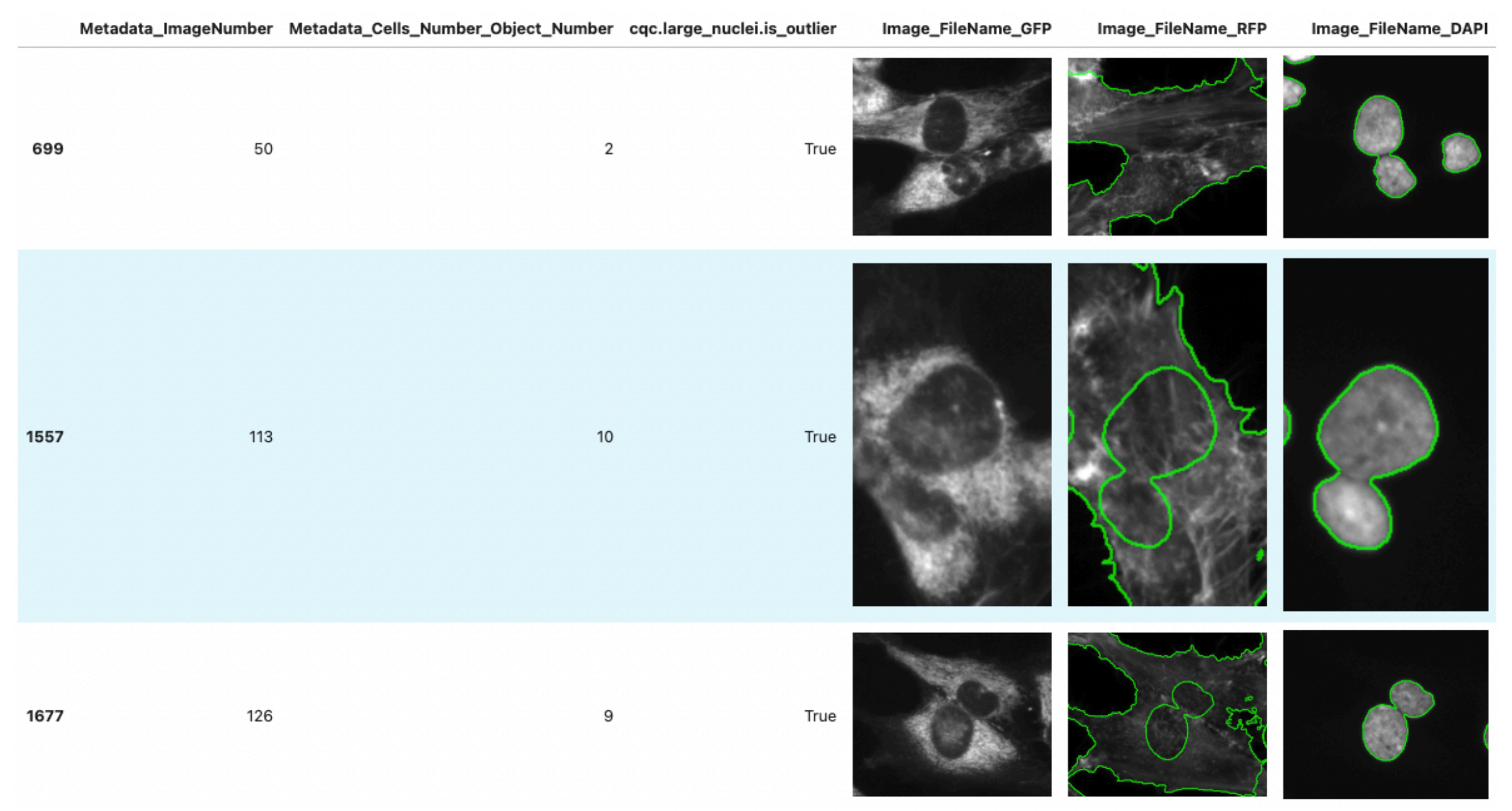
Figure 4: *Histograms from **coSMicQC** help scientists identify where outliers occur in the distribution.*

Deep erroneous anomaly analysis is enabled within **coSMicQC** through the ***`label_outliers`*** function, which appends z-score data for features, and the ***`CytoDataFrame.show_report`*** method to visualize where outliers are detected within the dataset.

☆ 4) *Understanding outlier segmentation issues*

```
import cosmicqc

# passing image and mask dirs to display
images
cosmicqc.CytoDataFrame(
    data="single-cell-profiles.parquet",
    data_context_dir="./image_directory/",
    data_mask_context_dir="./
mask_directory/",
)
```



	Metadata_ImageNumber	Metadata_Cells_Number_Object_Number	cqc.large_nuclei.is_outlier	Image_FileName_GFP	Image_FileName_RFP	Image_FileName_DAPI
699	50	2	True			
1557	113	10	True			
1677	126	9	True			

Figure 5: *Displaying outlier and image data increases analysis iteration within familiar interfaces.*

CytoDataFrames returned by all **coSMicQC** operations enable researchers to analyze outlier status alongside single-cell images directly in a Jupyter environment.

IV. Real-world applications

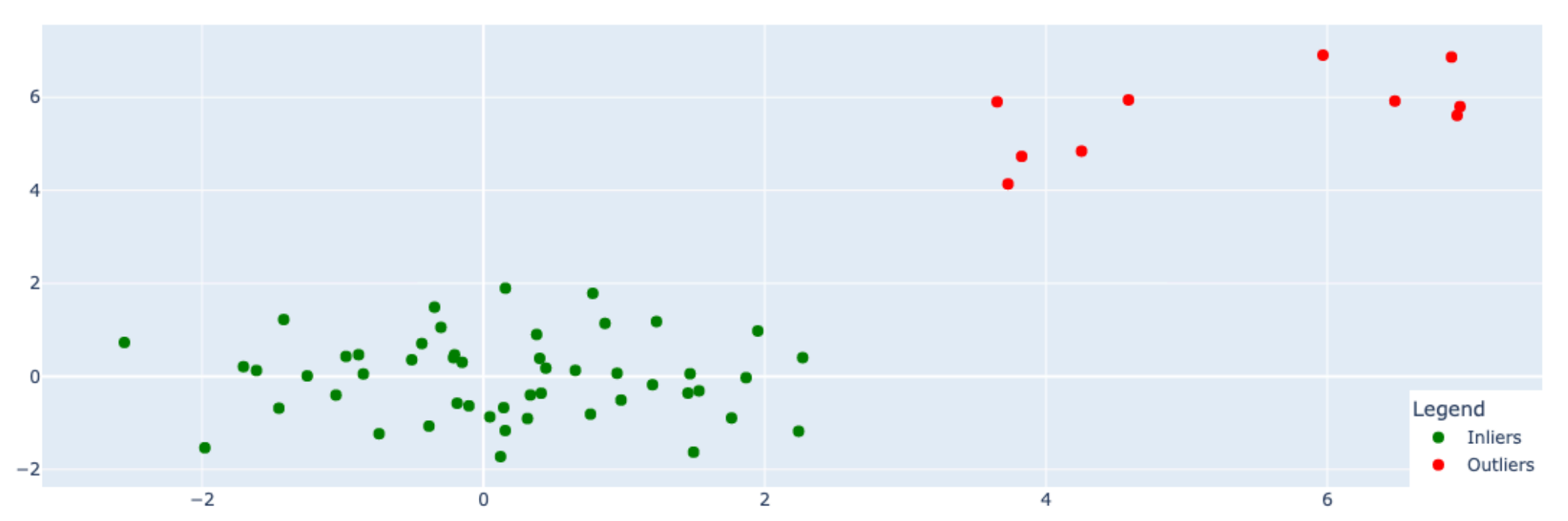


Figure 6: *Etiam viverra sollicitudin velit.*

Lorem ipsum dolor sit amet, consectetur adipiscing elit. In vel augue ante. Duis placerat ex id turpis consequat, at aliquet arcu vehicula.

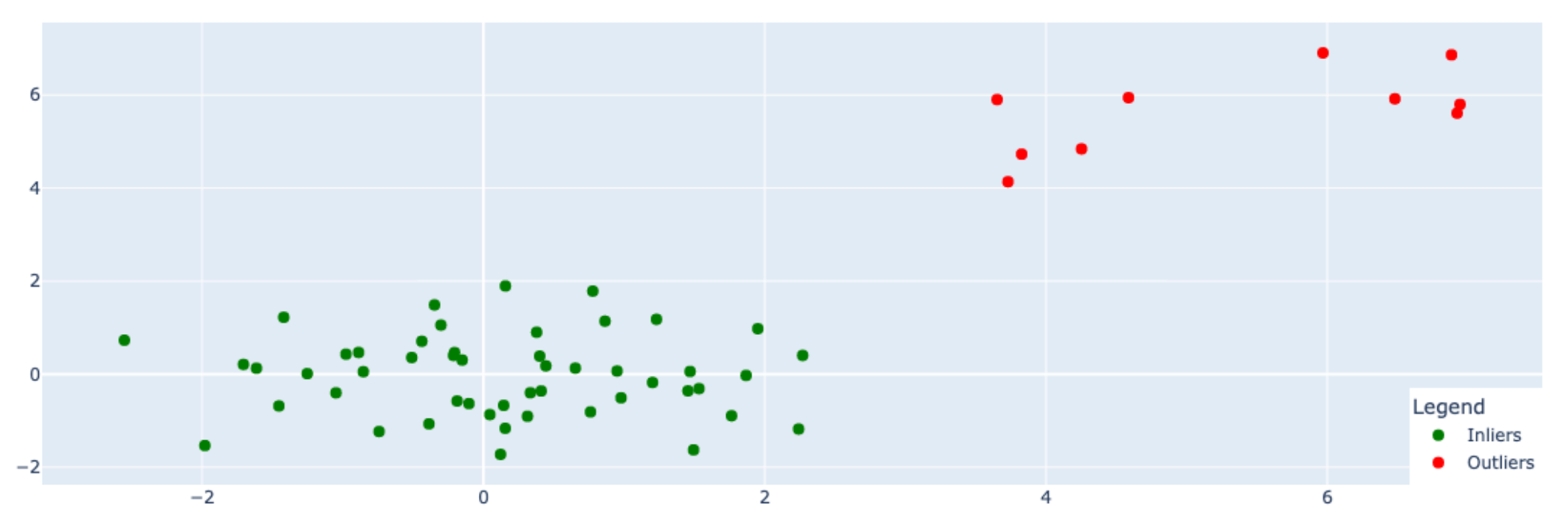


Figure 7: *Ut ut metus at diam hendrerit dictum.*

Donec bibendum ex vitae egestas ornare. Cras ut vulputate diam. Nullam feugiat feugiat purus, eu various metus maximus sit amet.

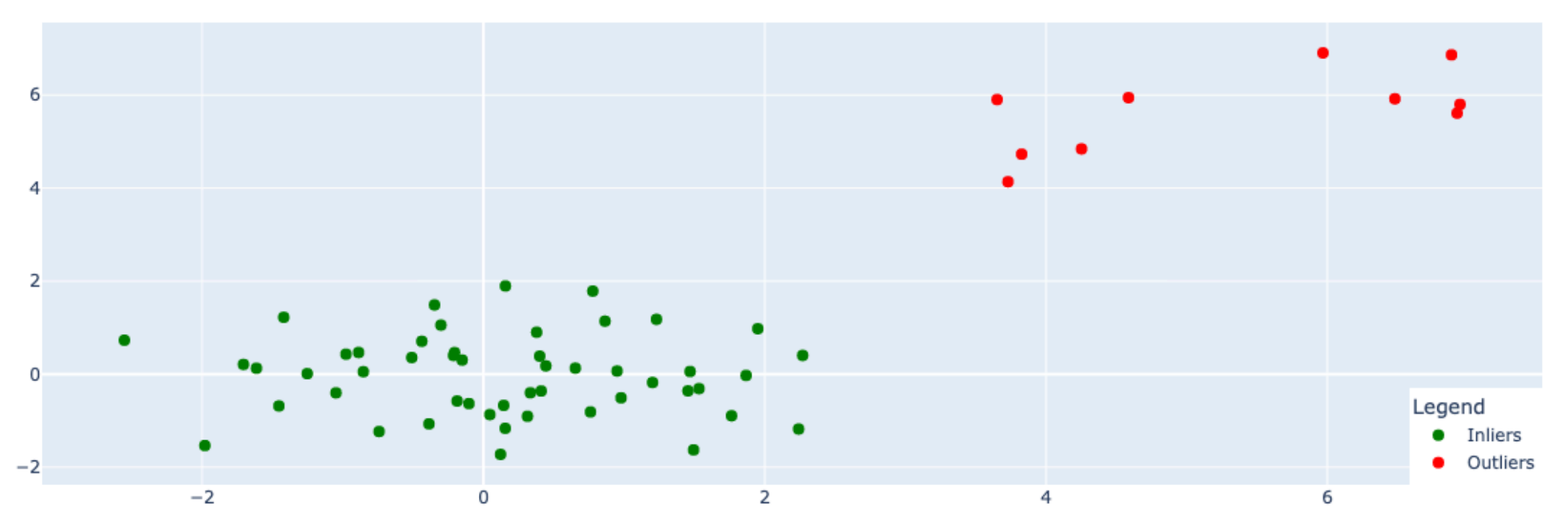


Figure 8: *Praesent eu sem id nibh viverra finibus.*

Fusce ac mi eu augue suscipit sodales. Morbi quis gravida dui. Name tellus elit, ullamcorper ut ipsum ut, eleifend facilisis dui.

V. Future steps

We plan to strengthen **coSMicQC** through further erroneous anomaly detection techniques, integrate with existing single-cell pipelines technologies, and expand image format compatibility.