



Nitrogenous Fate

Susan Garcia, Aaron Lecciones, Brian Roman, Carlyn Schmidgall

CSE583 Software Development for Data Scientists
University of Washington, Seattle

Autumn 2023
13 December 2023



CSE 583 UW Fall AY 2023-24

Presentation Outline

1. Problem Statement
2. Proposed Solution
3. Software Design
 - Target Users
 - Functional Design
 - Technology Review
4. Project Design
 - Github Repository
 - Project Directory Structure
 - Components
 - Scripts
 - Testing
5. Results

Problem Statement

Biological oceanographers and marine biologists who study environmental samples in their microbial system experiments to characterize cellular phenotype and biochemical pathways are faced with the effects of obscuring variation when normalization is not conducted during measuring the rate of intracellular and dissolved metabolite production.

One normalization method employed is the best-matched internal standard (B-MIS) normalization which is a step-heavy process (Boysen *et al* 2018). Data visualization is also another step conducted as part of the process (Sacks *et al* 2022).

Proposed Solution

The team shall write code to handle data normalization and data visualization:

- Python Script for normalization of data and calculation of peak areas.
 - Panda
- Jupyter Notebook for data visualization
 - Seaborn for exploratory data analysis
 - Altair to develop interactive plots

Target Users

User Characterization for NF Software

The Nitrogenous Fate Software uses metabolite data of isotopically-labeled molecules (Nitrite, Ammonia, and Urea) through incubations from samples in the Equatorial Pacific to show pathways of nitrogen within communities through simple data visualization.

=====

User	Needs and Desires	Skill Level	Use Cases
Research Scientist (Marine Biologist)	Track isotopically-labeled Nitrogen within marine communities. Desires a simple yet parameter-configurable interface with multiple options for visualization.	Highly skilled scientist who is familiar with basic data science tools	Run program as is
Research Scientist (Generic)	Use software to track defined elements in a known system. Desires a simple yet highly parameter-configurable interface with multiple options for visualization.	Highly skilled scientist who is familiar with basic data science tools and Github coding	Run program with customization
Research Scientist (administrator)	Maintain, improve, and update program as needed. Desires a simple system reporting tool for user feedback.	Highly skilled scientist who is familiar with the coding used for the program	Run and add code or debug program
Researcher	Use software but without much knowledge of underlying mechanics. Desires a way to use the software without much research or professional background in the domain	Adequately skilled researcher who might not be familiar with data science tools	Run program with assistance

Document Information: CSE583 NF Project Documentation for User Characterization

Functional Design

Version 0.1 of this software implements only the following functional design for nitrogenousfate.py:

Functional Design	Use Case	Description	Prompt
Run Program	Input Data Set	Loads data file based on run command checks for data format	May raise exceptions when encountered
	Data Set Processing	scripts runs using either customization or without customizations.	
		<i>Section I: Data Cleaning and Organization</i>	
		<i>Section II: Best Matched Internal Standard Normalization</i>	
		Data analysis encounters an error	raise Exception or display Error, exit program
		Data analysis is successful	Display: 'Analysis completed' save output files.
End Program	Terminate Program	Safely terminate program and show credits	Display thank you prompt with credits, then exit app.

Section III is handled by a Jupyter Notebook.

Document Information: CSE583 NF Project Documentation for Functional Design

Technology Review



PANDAS

pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.



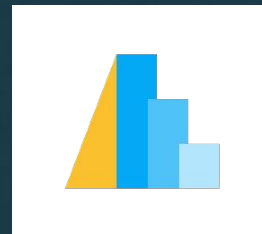
SEABORN

Positives:

- Standard plots out of the box
- Perfect for statistical analysis
- Fast to use for standard plots

Negatives:

- Built on top of matplotlib
- Less ability to customize



ALTAIR

Built on top of Vega and Vega-Lite grammars

Positives:

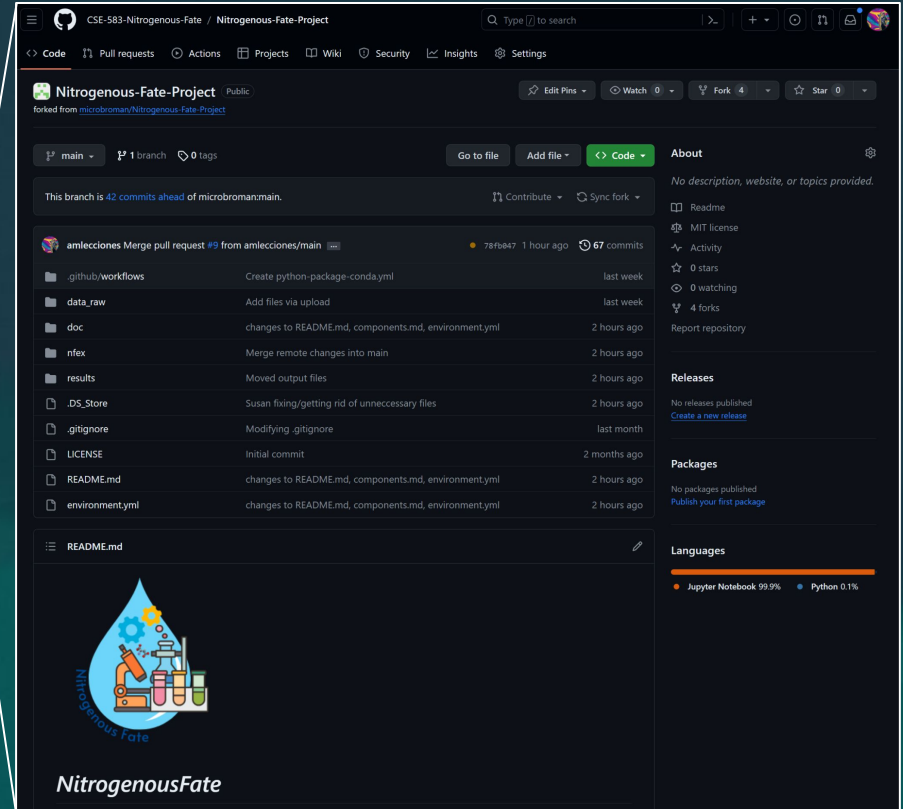
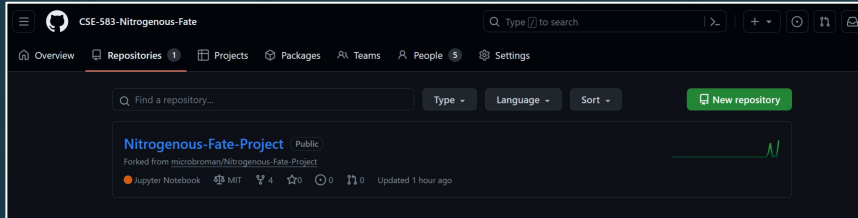
- Intuitive and structured approach to plotting
- Altair is interactive (zoom in, pan and grab, tooltips, etc)
- Flexible

Negatives:

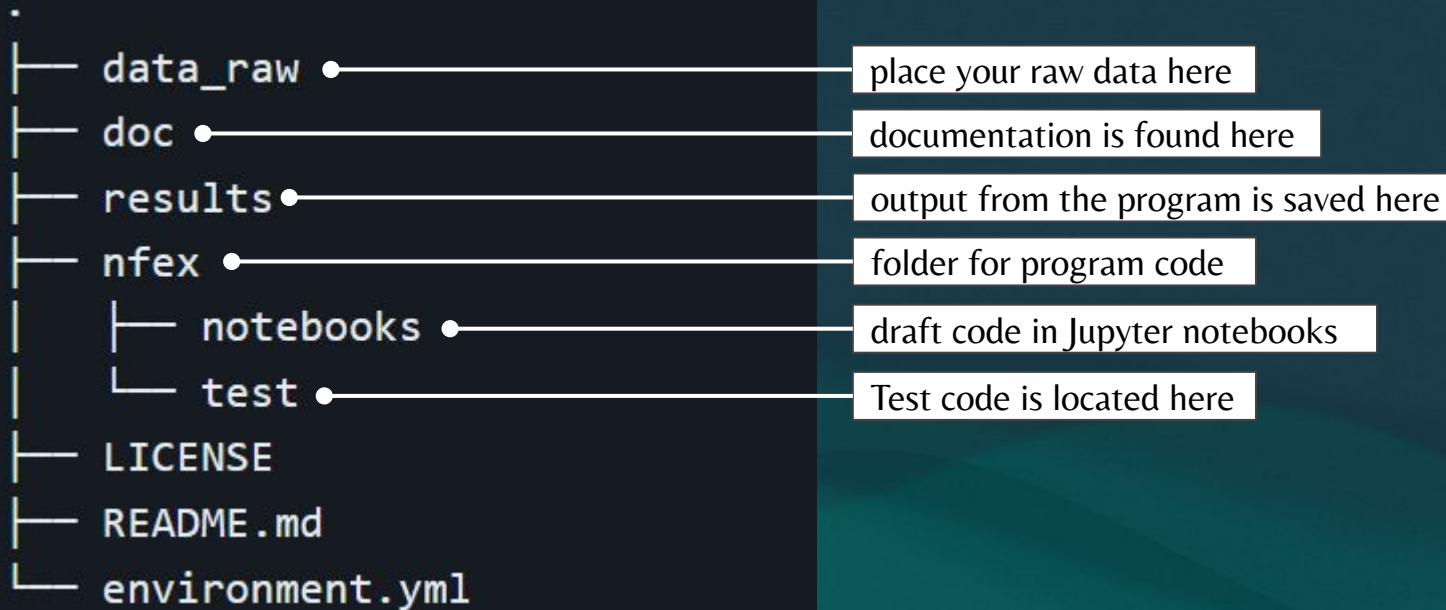
- No 3D plotting
- Not as customizable

Project Design - Github Repository

<https://github.com/orgs/CSE-583-Nitrogenous-Fate/repositories>

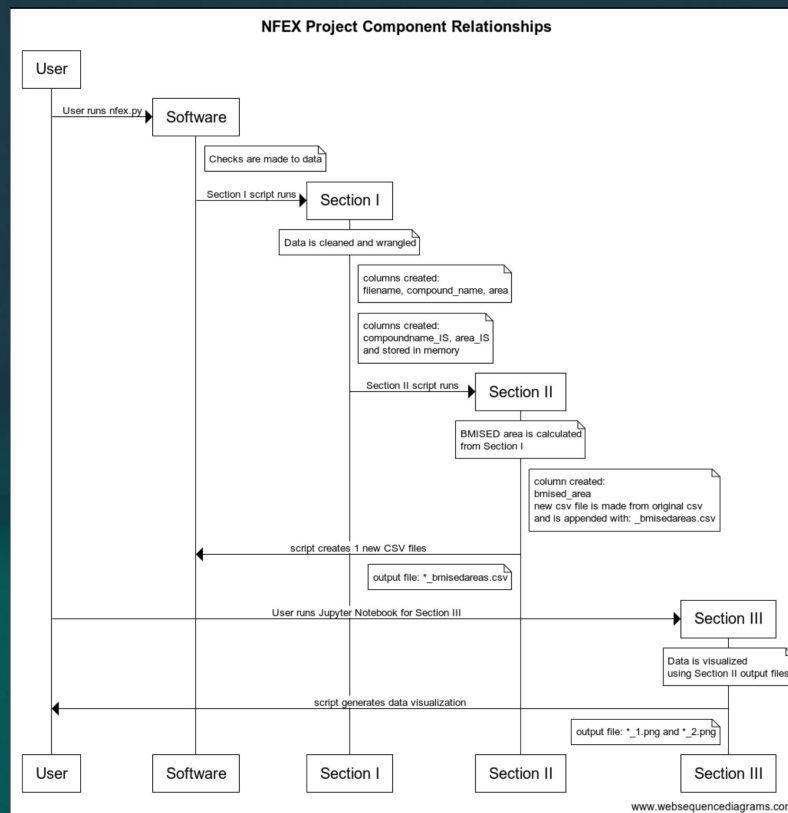


Project Design - Project Directory Structure



Project Design - Components

Name	Function	Inputs	Output	Interactions
Data Cleaning and Wrangling (Section I)	A <code>dataset</code> is loaded (either dissolved, exo or particulate, endo) and this component looks for relevant data (by column) from the data set. The script then calculates values for the following new columns which are appended to the <code>dataset</code> under a new file named <code>*_clean_areas.csv : filename, compound_name, and area</code> . It then computes an internal standard which is used to normalize data for use in section II, it creates two new columns for these values: <code>compoundname_IS</code> and <code>area_IS</code> , which are appended to the <code>dataset</code> under a new file names <code>*_IS_list.csv</code> .	run <code>nfex</code> (nitrogenousfate.py) <code>Section I script</code> with data in <code>data_raw</code> folder.	component will output values for: <code>filename, compound_name, area, compoundname_IS</code> and <code>area_IS</code> .	User monitors for any errors.
BMISED Computation (Section II)	The new data in memory which now contains values in the new columns: <code>filename, compound_name, area, compoundname_IS</code> and <code>area_IS</code> is used to calculate BMISED values. These values are placed in a new column: <code>bmised_area</code> and is saved in a dataset named <code>*_bmisedareas.csv</code> .	<code>nfex Section II script</code> runs using new data stored in memory.	component will output the new dataset: <code>*_bmisedareas.csv</code> for values in the new column: <code>bmised_area</code> in <code>results</code> folder.	User monitors for any errors expects a CSV file in <code>results</code> folder.
Data Visualization (Section III)	The <code>new dataset</code> which now contains values in the new column <code>bmised_area</code> is used to generate data visualization. The Jupyter Notebook generates multiple data visualizations to choose from. The user may save them into files.	run Jupyter Notebook (XXXX.ipynb) <code>Section III script</code> with correct data in <code>results</code> folder.	component will output image files from user input in Jupyter Notebook script.	User monitors for any errors, uses Jupyter Notebook to generate and save visualization files.



Project Design - Scripts

Version 0.1

R Script

Section I & II

Computation for BMIS

```

2 library(tidyverse)
3 library(jstatmod)
4 library(dplyr)
5 library(tibble)
6 library(rlog)
7
8 # a section ----
9
10 # cleaning and wrangling
11 transition_list <- read_csv("data/raw/WILDCO_POSITIVE_WGS_disinfect_NFLC1")
12
13 clear_names <- transition_list %>%
14   select(filenam:Replicate name, compound_name:Precursor ion name, "non-
15     right_join(right_join(select(compound_name, "A", "I5", "non-ri
16     mutate(arname=number(arname)) %>%
17     select(filenam:Replicate name, compound_name:Precursor ion name, "non-
18     mutate(dose=extract(filenam, "fileid")) %>%
19     select(filenam:Replicate name, compound_name:Precursor ion name, "non-
20     mutate(filenam=number(filenam), dose=extract(filenam, "fileid")) %>%
21     arrange(compound_name, filenam)
22
23 IS_list <- clear_names %>%
24   filter(comp_type == "Internal Standard")
25
26 standards_list_all_pos <- read_csv("data/raw/EXTINGUISH_lab_standards")
27 clear_names %>%
28   filter(comp_type == "Internal Standard")
29   filter(column=="WILDCO")
30
31 standards_list_pos <- standards_list_all_pos %>%
32   filter(comp_type == "Internal Standard")
33   filter(column=="WILDCO")
34
35 # a section ----
36
37 # cleaning and wrangling
38 transition_list <- read_csv("data/raw/WILDCO_POSITIVE_WGS_disinfect_NFLC1")
39
40 clear_names <- transition_list %>%
41   select(filenam:Replicate name, compound_name:Precursor ion name, "non-
42     right_join(right_join(select(compound_name, "A", "I5", "non-ri
43     mutate(arname=number(arname)) %>%
44     select(filenam:Replicate name, compound_name:Precursor ion name, "non-
45     mutate(dose=extract(filenam, "fileid")) %>%
46     select(filenam:Replicate name, compound_name:Precursor ion name, "non-
47     mutate(filenam=number(filenam), dose=extract(filenam, "fileid")) %>%
48     arrange(compound_name, filenam)
49
50 IS_list <- clear_names %>%
51   filter(comp_type == "Internal Standard")
52
53 standards_list_all_pos <- read_csv("data/raw/EXTINGUISH_lab_standards")
54 clear_names %>%
55   filter(comp_type == "Internal Standard")
56   filter(column=="WILDCO")
57
58 standards_list_pos <- standards_list_all_pos %>%
59   filter(comp_type == "Internal Standard")
60   filter(column=="WILDCO")
61
62 # a section ----
63
64 # cleaning and wrangling
65 transition_list <- read_csv("data/raw/WILDCO_POSITIVE_WGS_disinfect_NFLC1")
66
67 clear_names <- transition_list %>%
68   select(filenam:Replicate name, compound_name:Precursor ion name, "non-
69     right_join(right_join(select(compound_name, "A", "I5", "non-ri
70     mutate(arname=number(arname)) %>%
71     select(filenam:Replicate name, compound_name:Precursor ion name, "non-
72     mutate(dose=extract(filenam, "fileid")) %>%
73     select(filenam:Replicate name, compound_name:Precursor ion name, "non-
74     mutate(filenam=number(filenam), dose=extract(filenam, "fileid")) %>%
75     arrange(compound_name, filenam)
76
77 IS_list <- clear_names %>%
78   filter(comp_type == "Internal Standard")
79
80 standards_list_all_pos <- read_csv("data/raw/EXTINGUISH_lab_standards")
81 clear_names %>%
82   filter(comp_type == "Internal Standard")
83   filter(column=="WILDCO")
84
85 standards_list_pos <- standards_list_all_pos %>%
86   filter(comp_type == "Internal Standard")
87   filter(column=="WILDCO")
88
89 # a section ----
90
91 # cleaning and wrangling
92 transition_list <- read_csv("data/raw/WILDCO_POSITIVE_WGS_disinfect_NFLC1")
93
94 clear_names <- transition_list %>%
95   select(filenam:Replicate name, compound_name:Precursor ion name, "non-
96     right_join(right_join(select(compound_name, "A", "I5", "non-ri
97     mutate(arname=number(arname)) %>%
98     select(filenam:Replicate name, compound_name:Precursor ion name, "non-
99     mutate(dose=extract(filenam, "fileid")) %>%
100    select(filenam:Replicate name, compound_name:Precursor ion name, "non-
101    mutate(filenam=number(filenam), dose=extract(filenam, "fileid")) %>%
102    arrange(compound_name, filenam)
103
104 IS_list <- clear_names %>%
105   filter(comp_type == "Internal Standard")
106
107 standards_list_all_pos <- read_csv("data/raw/EXTINGUISH_lab_standards")
108 clear_names %>%
109   filter(comp_type == "Internal Standard")
110   filter(column=="WILDCO")
111
112 standards_list_pos <- standards_list_all_pos %>%
113   filter(comp_type == "Internal Standard")
114   filter(column=="WILDCO")
115
116 # a section ----
117
118 # cleaning and wrangling
119 transition_list <- read_csv("data/raw/WILDCO_POSITIVE_WGS_disinfect_NFLC1")
120
121 clear_names <- transition_list %>%
122   select(filenam:Replicate name, compound_name:Precursor ion name, "non-
123     right_join(right_join(select(compound_name, "A", "I5", "non-ri
124     mutate(arname=number(arname)) %>%
125     select(filenam:Replicate name, compound_name:Precursor ion name, "non-
126     mutate(dose=extract(filenam, "fileid")) %>%
127     select(filenam:Replicate name, compound_name:Precursor ion name, "non-
128     mutate(filenam=number(filenam), dose=extract(filenam, "fileid")) %>%
129     arrange(compound_name, filenam)
130
131 IS_list <- clear_names %>%
132   filter(comp_type == "Internal Standard")
133
134 standards_list_all_pos <- read_csv("data/raw/EXTINGUISH_lab_standards")
135 clear_names %>%
136   filter(comp_type == "Internal Standard")
137   filter(column=="WILDCO")
138
139 standards_list_pos <- standards_list_all_pos %>%
140   filter(comp_type == "Internal Standard")
141   filter(column=="WILDCO")
142
143 # a section ----
144
145 # cleaning and wrangling
146 transition_list <- read_csv("data/raw/WILDCO_POSITIVE_WGS_disinfect_NFLC1")
147
148 clear_names <- transition_list %>%
149   select(filenam:Replicate name, compound_name:Precursor ion name, "non-
150     right_join(right_join(select(compound_name, "A", "I5", "non-ri
151     mutate(arname=number(arname)) %>%
152     select(filenam:Replicate name, compound_name:Precursor ion name, "non-
153     mutate(dose=extract(filenam, "fileid")) %>%
154     select(filenam:Replicate name, compound_name:Precursor ion name, "non-
155     mutate(filenam=number(filenam), dose=extract(filenam, "fileid")) %>%
156     arrange(compound_name, filenam)
157
158 IS_list <- clear_names %>%
159   filter(comp_type == "Internal Standard")
160
161 standards_list_all_pos <- read_csv("data/raw/EXTINGUISH_lab_standards")
162 clear_names %>%
163   filter(comp_type == "Internal Standard")
164   filter(column=="WILDCO")
165
166 standards_list_pos <- standards_list_all_pos %>%
167   filter(comp_type == "Internal Standard")
168   filter(column=="WILDCO")
169
170 # a section ----
171
172 # cleaning and wrangling
173 transition_list <- read_csv("data/raw/WILDCO_POSITIVE_WGS_disinfect_NFLC1")
174
175 clear_names <- transition_list %>%
176   select(filenam:Replicate name, compound_name:Precursor ion name, "non-
177     right_join(right_join(select(compound_name, "A", "I5", "non-ri
178     mutate(arname=number(arname)) %>%
179     select(filenam:Replicate name, compound_name:Precursor ion name, "non-
180     mutate(dose=extract(filenam, "fileid")) %>%
181     select(filenam:Replicate name, compound_name:Precursor ion name, "non-
182     mutate(filenam=number(filenam), dose=extract(filenam, "fileid")) %>%
183     arrange(compound_name, filenam)
184
185 IS_list <- clear_names %>%
186   filter(comp_type == "Internal Standard")
187
188 standards_list_all_pos <- read_csv("data/raw/EXTINGUISH_lab_standards")
189 clear_names %>%
190   filter(comp_type == "Internal Standard")
191   filter(column=="WILDCO")
192
193 standards_list_pos <- standards_list_all_pos %>%
194   filter(comp_type == "Internal Standard")
195   filter(column=="WILDCO")
196
197 # a section ----
198
199 # cleaning and wrangling
200 transition_list <- read_csv("data/raw/WILDCO_POSITIVE_WGS_disinfect_NFLC1")
201
202 clear_names <- transition_list %>%
203   select(filenam:Replicate name, compound_name:Precursor ion name, "non-
204     right_join(right_join(select(compound_name, "A", "I5", "non-ri
205     mutate(arname=number(arname)) %>%
206     select(filenam:Replicate name, compound_name:Precursor ion name, "non-
207     mutate(dose=extract(filenam, "fileid")) %>%
208     select(filenam:Replicate name, compound_name:Precursor ion name, "non-
209     mutate(filenam=number(filenam), dose=extract(filenam, "fileid")) %>%
210     arrange(compound_name, filenam)
211
212 IS_list <- clear_names %>%
213   filter(comp_type == "Internal Standard")
214
215 standards_list_all_pos <- read_csv("data/raw/EXTINGUISH_lab_standards")
216 clear_names %>%
217   filter(comp_type == "Internal Standard")
218   filter(column=="WILDCO")
219
220 standards_list_pos <- standards_list_all_pos %>%
221   filter(comp_type == "Internal Standard")
222   filter(column=="WILDCO")
223
224 # a section ----
225
226 # cleaning and wrangling
227 transition_list <- read_csv("data/raw/WILDCO_POSITIVE_WGS_disinfect_NFLC1")
228
229 clear_names <- transition_list %>%
230   select(filenam:Replicate name, compound_name:Precursor ion name, "non-
231     right_join(right_join(select(compound_name, "A", "I5", "non-ri
232     mutate(arname=number(arname)) %>%
233     select(filenam:Replicate name, compound_name:Precursor ion name, "non-
234     mutate(dose=extract(filenam, "fileid")) %>%
235     select(filenam:Replicate name, compound_name:Precursor ion name, "non-
236     mutate(filenam=number(filenam), dose=extract(filenam, "fileid")) %>%
237     arrange(compound_name, filenam)
238
239 IS_list <- clear_names %>%
240   filter(comp_type == "Internal Standard")
241
242 standards_list_all_pos <- read_csv("data/raw/EXTINGUISH_lab_standards")
243 clear_names %>%
244   filter(comp_type == "Internal Standard")
245   filter(column=="WILDCO")
246
247 standards_list_pos <- standards_list_all_pos %>%
248   filter(comp_type == "Internal Standard")
249   filter(column=="WILDCO")
250
251 # a section ----
252
253 # cleaning and wrangling
254 transition_list <- read_csv("data/raw/WILDCO_POSITIVE_WGS_disinfect_NFLC1")
255
256 clear_names <- transition_list %>%
257   select(filenam:Replicate name, compound_name:Precursor ion name, "non-
258     right_join(right_join(select(compound_name, "A", "I5", "non-ri
259     mutate(arname=number(arname)) %>%
260     select(filenam:Replicate name, compound_name:Precursor ion name, "non-
261     mutate(dose=extract(filenam, "fileid")) %>%
262     select(filenam:Replicate name, compound_name:Precursor ion name, "non-
263     mutate(filenam=number(filenam), dose=extract(filenam, "fileid")) %>%
264     arrange(compound_name, filenam)
265
266 IS_list <- clear_names %>%
267   filter(comp_type == "Internal Standard")
268
269 standards_list_all_pos <- read_csv("data/raw/EXTINGUISH_lab_standards")
270 clear_names %>%
271   filter(comp_type == "Internal Standard")
272   filter(column=="WILDCO")
273
274 standards_list_pos <- standards_list_all_pos %>%
275   filter(comp_type == "Internal Standard")
276   filter(column=="WILDCO")
277
278 # a section ----
279
280 # cleaning and wrangling
281 transition_list <- read_csv("data/raw/WILDCO_POSITIVE_WGS_disinfect_NFLC1")
282
283 clear_names <- transition_list %>%
284   select(filenam:Replicate name, compound_name:Precursor ion name, "non-
285     right_join(right_join(select(compound_name, "A", "I5", "non-ri
286     mutate(arname=number(arname)) %>%
287     select
```

Jupyter Notebook

[illegible]

Python Script

[illegible]

Data Visualization

[illegible]

Future Versions

Project Design - Testing

Unit Tests performed to ensure correct data formatting, types in CSV file passed into data processing scripts.

Tests ensure:

- Viable, non-empty CSV is supplied
- Data types within CSV is correct (numeric)
- CSV contains internal standards to allow for quality control of data



```
"""
This module contains the unit test functions for the nitrogenous fate (nfex)
project

Classes:
- TestKFEX(unittest.TestCase)
"""

import unittest
from nitrogenousfate import process_csv

class TestNFEX(unittest.TestCase):
    """
    TestNFEX(unittest.TestCase): tests the code for data cleaning and wrangling for NFEX.
    """

    def test_smoke_pass(self):
        """
        test_smoke_pass(self): Smoke test (1) verifying function should run without
        crashing or throwing errors if given an appropriate CSV file.
        """
```

Results

Altair visualization:

- All data points in one graph
- Interactive graph - can select any compound from the legend
- Tooltip - can see information for any datapoint





Nitrogenous Fate

Susan Garcia, Aaron Lecciones, Brian Roman, Carlyn Schmidgall

CSE583 Software Development for Data Scientists
University of Washington, Seattle

Autumn 2023

Thank you For listening!

