FINAL REPORT

Life Cycle Assessment of Nano-enabled Products and Devices for their Safe and Rapid Development

Cooperative Agreement Number: W912HZ-19-SOI-0028

Thomas P Seager, Paul Westerhoff, co-PIs
School of Sustainable Engineering & the Built Environment
Arizona State University, Tempe AZ

Summary

Assement of life-cycle environmental impacts during the development of novel technoligies may help reorient research along trajectories that mitigate or minimize unintended deleterious consequences. Earlier assessment affords greater developmental flexibility, but also presents increasingly difficult challenges with regard to data availability, reliablity, and management of uncertainty. While traditional methods of environmental life cycle assessment (LCA) are of limited applicability, *anticipatory* LCA is a relatively new analytical method for exploring and prioritizing uncertainties with regard to technology research and development.

Existing approaches to anticipatory LCA rely on computationally intense Monte Carlo simulation of data uncertainties to identify those that are either: 1) most decision-relevant, and/or 2) high priorities for research-based uncertainty reductions.

There are no commercial software packages that execute anticipatory LCA computations. Rather, existing studies have been supported by custom software solutions, typically developed within academic or government research groups. Among the most sophisticated of these is called *FELCI*, developed by the US Army Corp of Engineers Engineer Research & Development Center (ERDC). Programmed in the language R, *FELCI* calls on a popular open-source *Excel*-based software model called *USEtox* made available thru usetox.org by the Technical University of Denmark (Fantke et al. 2017).

USEtox represents the state-of-the-art model for estimating characterization factors of unknown compounds, such as nano-structured materials. However, USEtox relies on point estimates, rather than probability distributions. *FELCI* operates by populating USEtox with information about uncertainty, simulating hundreds or thousands of Monte Carlo trials, collecting the output data from *USEtox*, and integrating it with research cost information supplied by expert users to graphically explore efficient frontiers between budget and hypothetical uncertainty reductions. Thus, *FELCI* may be a powerful tool for evaluating competing research agendas.

The current challenge facing *FELCI* users is the long computational time required to run multiple Monte Carlo trials, which presents an obstacle to testing different research scenarios. This project investigated the computational bottlenecks associated with *FELCI* and reprogrammed a new version in Python called *PyFELCI* that is computationally more efficient. Despite faster speeds resulting from multi-core processing, *PyFELCI* operates in the same way as *FELCI* – as an external shell for controlling *USETox* and collecting data output. Therefore, it is subject to the same computational bottlenecks – namely, read/write latency and execution of the extensive *Excel* macros on which *USETox* relies.

There are *Excel* plug-ins that speed Monte Carlo and uncertainty analysis. For example, Oracle provides a plug-in called *Crystal Ball*, that carries out rapid exploration of probability distribution functions. However, implementing *Excel* plug-ins to execute *USEtox* is problematic for two reasons: 1) the extensive macros on which *USEtox* depends make plug-ins like *Crystal Ball* impractical, and 2) security concerns related to plug-ins prohibit use in enterprise computing environments like the Department of Defense and public universities.

This project has developed a Python-based code called *PyFELCI.* To foster adoption and further development of new tools for advancing and automating methods of anticipatory LCA, *PyFELCI* can be made freely downloadable and open source in two versions: 1) a US Army Corp ERDC version available on a secure file-sharing server approved for DoD use, and 2) a public version hosted on a box.com server open to anyone with the link.

Installation instructions for the public version are listed in Appendix A.

Introduction

The effective deployment of nanomaterial-enabled technologies could enable breakthroughs in energy storage, communications, environmental treatment, and myriad other technologies. Nonetheless, the potential application of nanomaterials must be weighed against the potential implications, because the environmental uncertainties associated with nanomaterials are extraordinarily high. In particular, the characterization factors required by LCA that translate environmental releases to impacts are largely unknown for novel nanomaterials, and these unknowns can create considerable uncertainty that is expensive to resolve using traditional laboratory toxicological assays.

Recent advances in anticipatory life cycle assessment (LCA) have enabled researchers to identify opportunities to steer the developmental trajectory of nanomaterial technologies towards environmentally preferable pathways by exploring and prioritizing uncertainties in environmental assessment (e.g., van der Giesen, Coen, et al. 2020). For example, *USEtox* is an Excel-based program that estimates characterization factors from physio-chemical quantities -- e.g., octanol-water partition coefficients. However, *USEtox* works exclusively with point-estimates, thereby complicating exploration of uncertainty and frustrating global sensitivity analysis.

To overcome limitations of *USEtox* for identifying critical uncertainties and sensitivities, researchers at ERDC created a computer program in R called *FELCI* to automate Monte Carlo analysis by populating multiple instances of *USEtox* with samplings from user-specified probability distributions of uncertain physio-chemical parameters. Ideally, this exploration allows identification of those uncertainties that are most relevant to a broader LCA, so that they might be prioritized for experimental investigation.

The problem is that *FELCI* is computationally inefficient. It requires long run times to create the minimum number of simulations required for informative exploration of typical probability distributions. Although *Excel*-add-ons like *Crystal Ball* can speed Monte Carlo simulation in simpler spreadsheets, *USEtox* makes extensive use of macros that interfere with simulations, and complicate export of data for production of custom figures utilized in *FELCI*.

This document describes the *PyFELCI* software tools designed to speed computations essential for anticipatory LCA. These tools are made available in two locations: 1) A Python application called *PyFELCI* suitable for installation on USACE ERDC computers that applies multi-core processing to speed execution of Monte Carlo trials in *USEtox* and made available on DoD approved file sharing servers, and 2) a second copy posted on publicly accessible servers hosted by box.com.

***PyFELCI* Testing**

To reduce the time taken for simulations, a multiprocessing Python application that reconstructs the ERDC software called *FELCI* (originally written in R) has been developed under a cooperative agreement between ASU and ERDC. This new application, called *PyFELCI,* uses multiple instances of *USEtox* running in parallel to deploy unused computational resources, and then merges the results of all the parallel simulations. The advantages of this multiprocessing approach depend upon the specific of each machine, including number of cores. For example, experimenting on a machine with 8 cores reveals that dedicating 6 cores to *USEtox* captures all available time savings.

Objective

The objective of this report is to show that *PyFELCI* application delivers same results as of *FELCI* application in less time.

Procedure

The Python application (*PyFELCI*) is tested against the original *FELCI* using identical inputs, in two scenarios: 1) a default input vector, and 2) randomly selected input values. The computation times and graphical results are then compared to assess consistency. For this purpose, several experiments were executed and typical results reported here.

PyFELCI Time Performance

Time utilization for **100** simulations:

# *USEtox* Instance	Total Time (min)	Time per sim (sec)	Note
4	26.65	15.99	Slow
6	24.45	14.67	Decent time: balance of time and resource utilization
8	24.43	14.65	Better time, but heavy resource utilization

Note: Actual *FELCI* takes 30.18 min for 100 simulations
Machine specs: intel - i7 core, 1.80GHz, 16 GB of RAM, 500GB SSD.

Results

To compare the results, the graphs produced in the results folder by both applications for each experiment are displayed below. The pattern observed was similar in all the graphs, even though a difference in scale could be found between them. Small differences between the results generated by *FELCI* and *PyFELCI* result from differences in stochastic sampling of identical probability distributions. Reliable results of 100 simulations show comparable overall patterns, without replicating exact datasets.

Direct comparison using identical default inputs

Value of Information Analysis. *FELCI* adds capabilities for prioritizing research projects that are not embedded in *USEtox.* The first of these is value-of-informaton (VOI) analysis that seeks to narrow uncertainty of results through experimental research that is hypothesized to decrease the range of probability distributons. Research experiments that are highest priority are those that are hypothesized to reduce uncertainties the most (e.g., Wender et al. 2018). Because the propogation of data uncertainties through life cycle assessment models like *USEtox* is complex, Monte Carlo simulation is necessary to inform experts regarding which experiments are most efficient for increasing confidence in the overall environmental portential of new technologies. In the figures below, *FELCI* results are on the left and *PyFELCI* results are on the right. Using identical inputs, the graphs show the same overall pattern of result, and identify comparable efficiency frontiers.

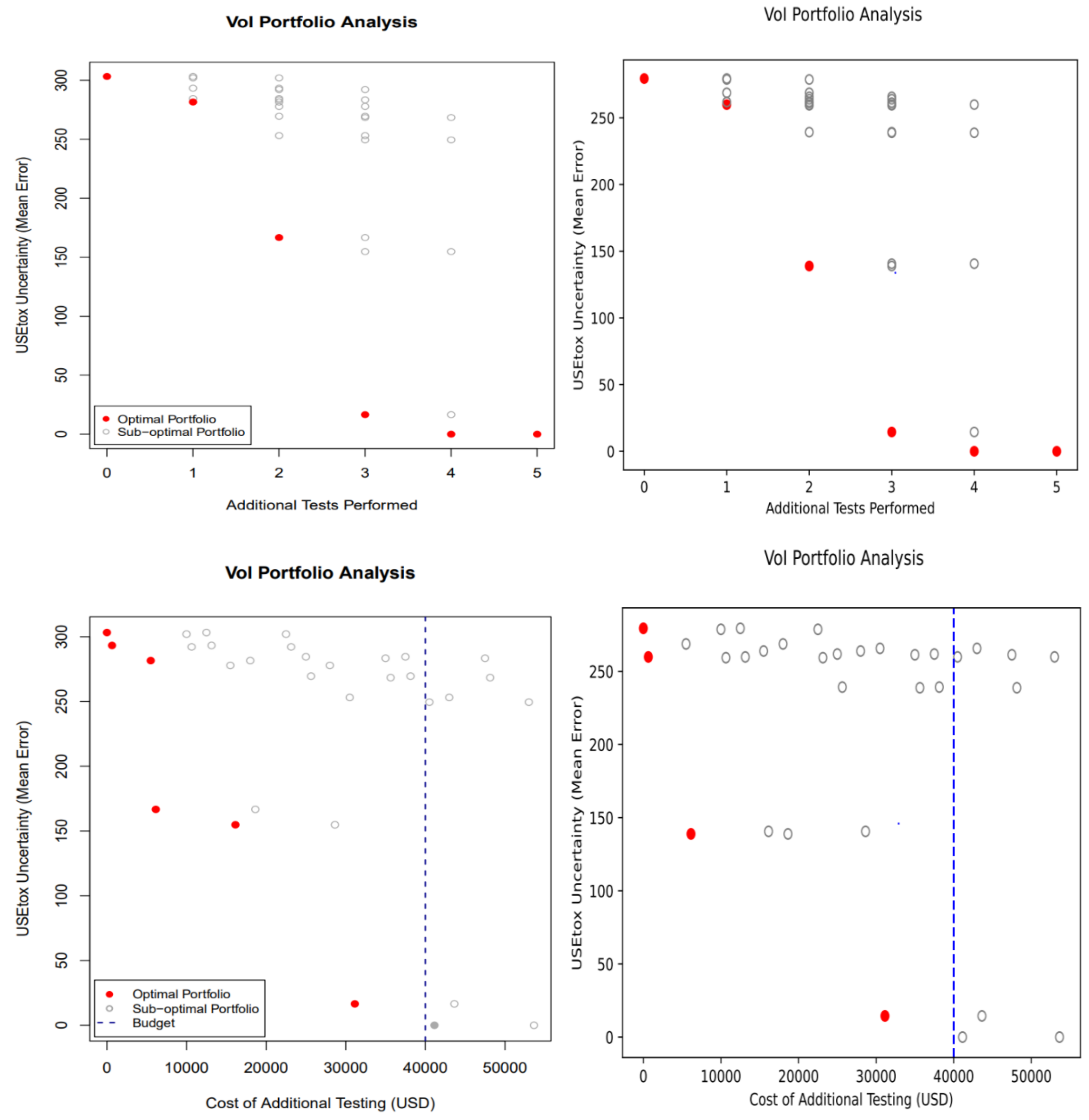


Figure 1. *FELCI* results (left) and *PyFELCI* (right) display comparable value-of-information results.

Research Portfolio Rankings. Because research is an inherently uncertain process, prioritizing among different projects requires direct comparison of the value of proposed projects. Highest ranked projects are those likely to generate efficient increases in information. *FELCI* (left) displays a stack-ranking of research project preferences, and their stability as the number of Monte Carlo trials increases. *PyFELCI* results in comparable rank-orderings, with improving comparison at larger numbers of simulations.

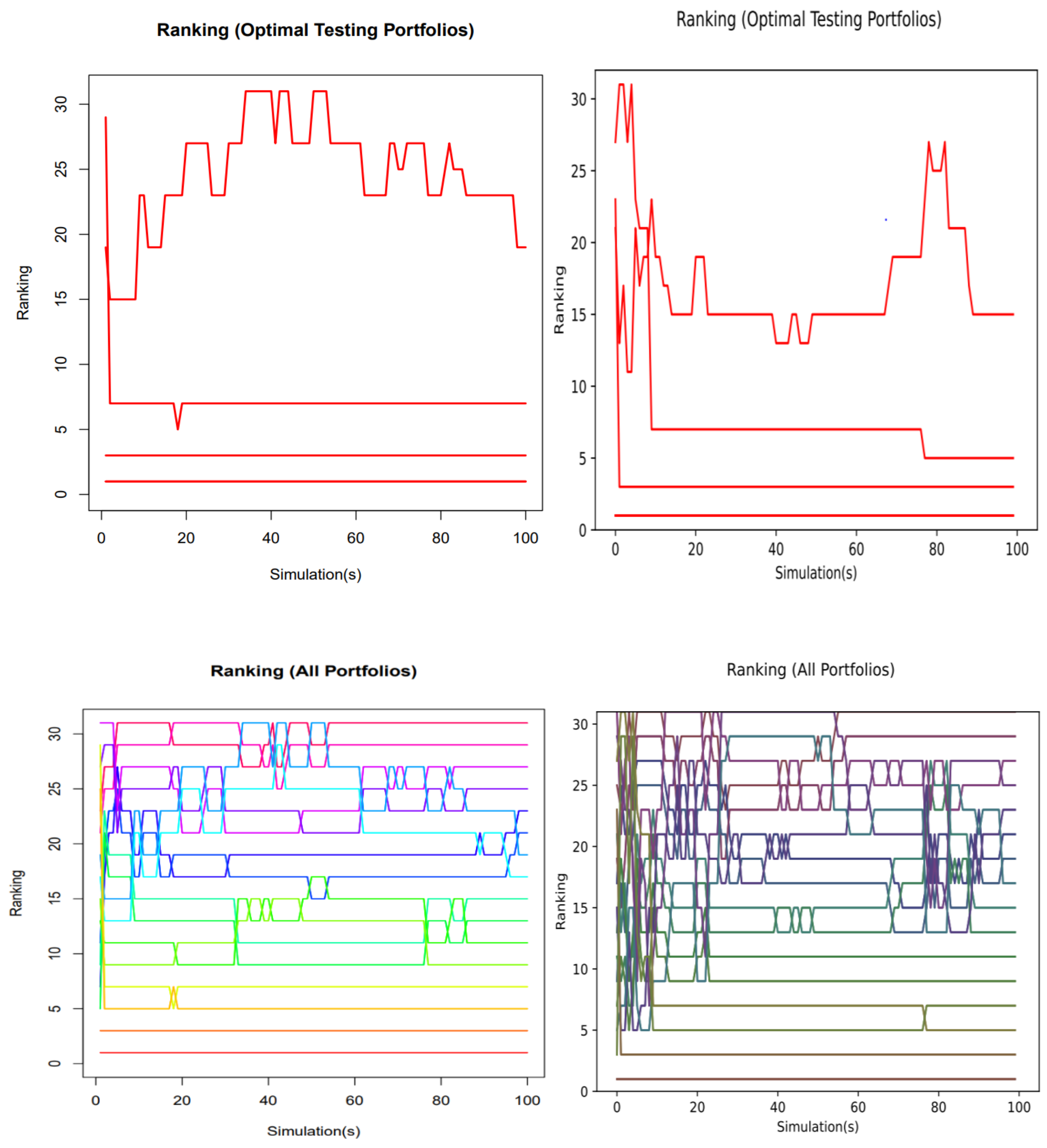


Figure 2. FELCI results (left) and comparable to PyFELCI results (right) in stack-ranking hypothetical research portfolios.

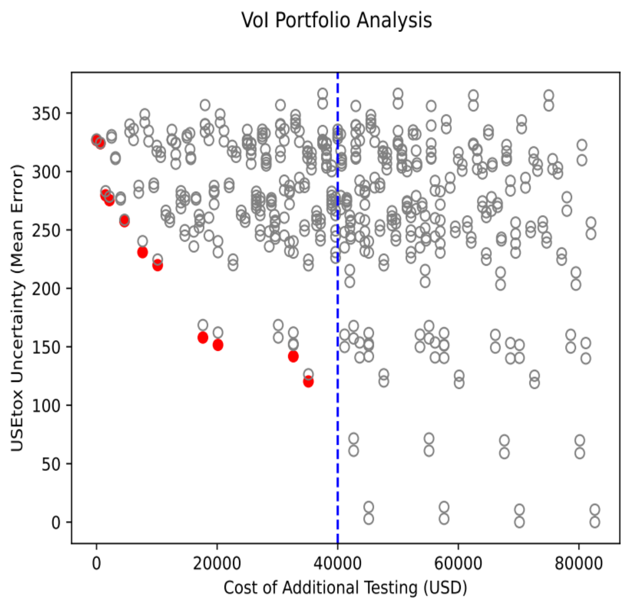
Figure 2 establishes the stability of the optimal research porfolio rankings at different numbers of Monte Carlo simulations. The portfolios represent different combinations of experimental test results that narrow uncertainty in the physical parameters that constitute USETox inputs. Optimal portfolios (top row) are those represented by red dots in Figure 1. These are the experimental tests that lie along frontier of least mean error and least additional testing costs. The bottom row of Figure 2 depicts all dots – white and red – represented in Figure 1.

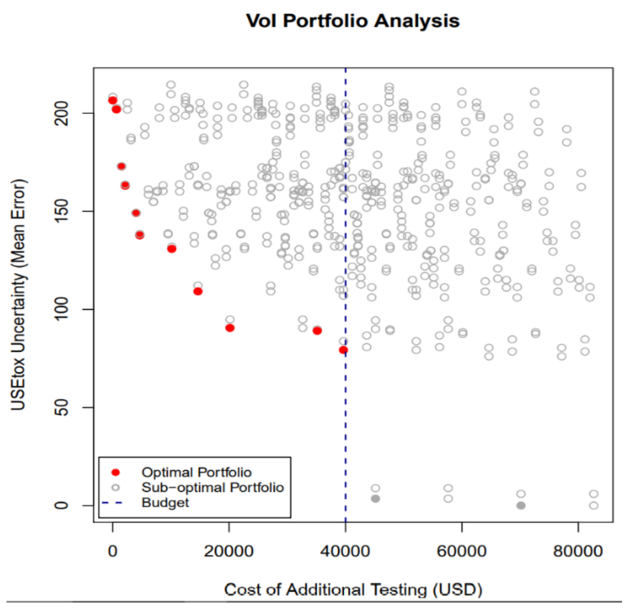
Evaluating the stability of the research portfolio rankings allows assessment of the minimum number of simulations necessary to reach reliable results. Because of the nature of Monte Carlo sampling, random variations will fail to produce reliable results when the number of samplings is too low. In this case, the critical output is the *relative ranking* of the research portfolios, rather than the mean error estimated for each. As a consequence, stable results in relative ranking can be obtained with fewer simulations than might otherwise be necessary for results to converge on absolute estimates of mean error. In the example given in Figure 2, the top three research portfolios are identifiable in fewer than 5 simulations, obviating the need to expend additional time and computational resources to compute additional simulations. Thus, much faster results can be obtained with confidence when the minimum number of simulations is established by the results depicted in Figure 2.

Direct comparison using identical randomly selected inputs

While comparison of *FELCI* and *PyFELCI* yields good results using a vector of default inputs (previous section), to investigate whether good comparison is stable over a wider range of input parameters, we compared results using randomly selected inputs.

Value of information. In Figure 3 below, *FELCI* results are on the left, while *PyFELCI* are reported on the right. Differences in scale distort visual interpretations of the results. Nonetheless, the same efficient frontier emerges in both instances, showing that optimal experiments exhibit decreasing returns in uncertainty reductions, with respec to increasing costs.





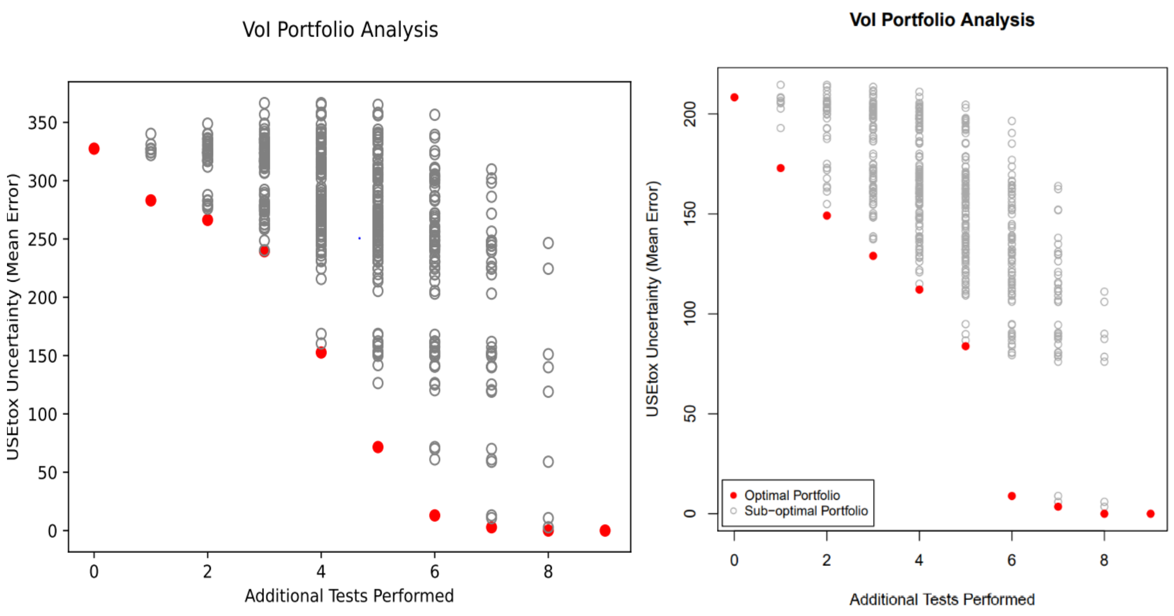
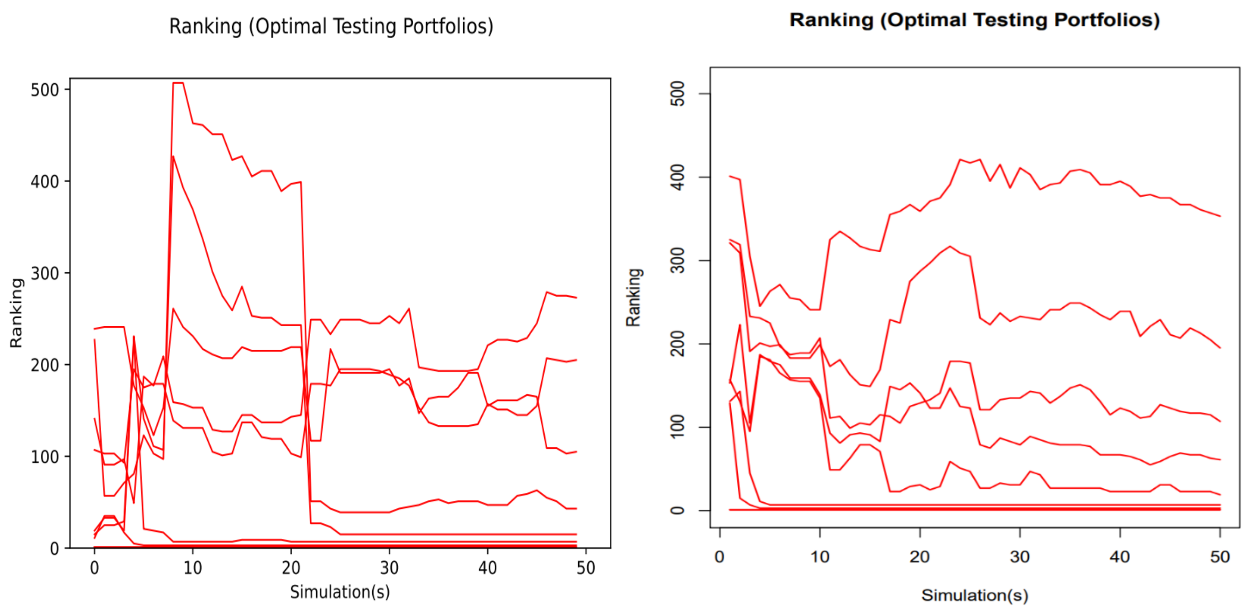


Figure 3. Randomly selected input vectors reveal that comparable results are robust across a wide range of parameters. *FELCI* results on the left, *PyFELCI* on the right.

Research Portfolio Rankings. Stable research portfolio rankings emerge between 20-30 simulations in both *FELCI* (left) and *PyFELCI* (right).



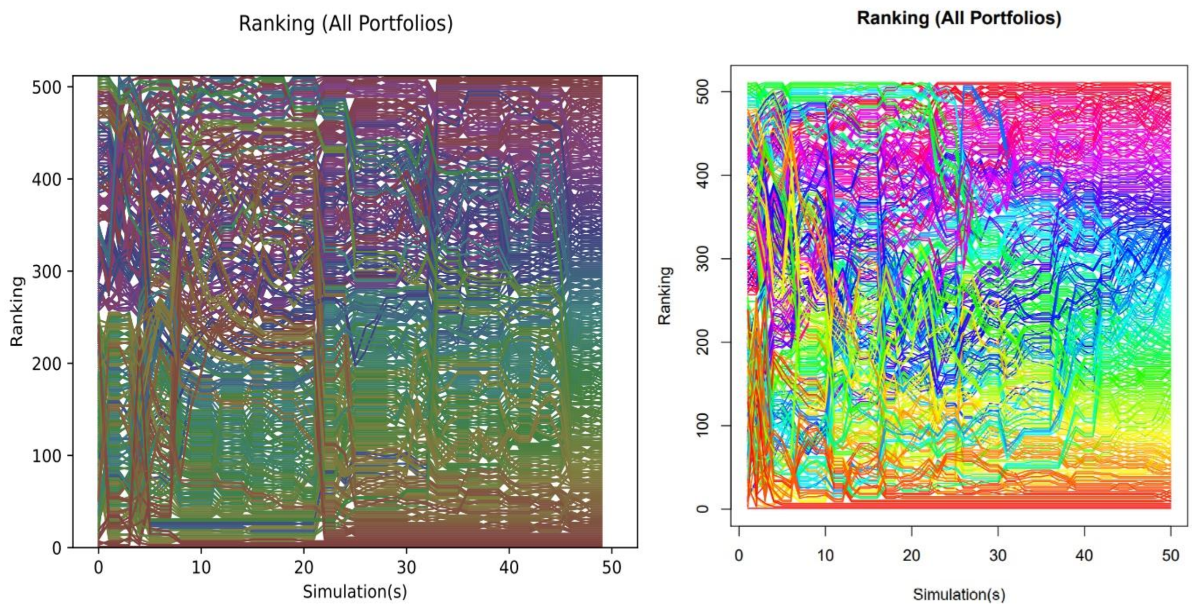


Figure 4. Using randomly selected input vectors, direct comparison of FELCI (left) and PyFELCI (right) results in comparable and stable research portfolio rankings after approximately 30 simulations.

PyFELCI Conclusions

PyFELCI delivers results comparable to FELCI in 30% less computational time.

References

Eckelman MJ, Mauter MS, Isaacs JA, Elimelech M. "New perspectives on nanomaterial aquatic ecotoxicity: Production impacts exceed direct exposure impacts for carbon nanotubes" *Environmental Science & Technology*. 46 (2012):2902-2910.

Fantke, P, et al. "USEtox® 2.0 Documentation (Version 1.00)." (2017). https://orbit.dtu.dk/en/publications/usetox-20-documentation-version-100

van der Giesen, Coen, et al. "A critical view on the current application of LCA for new technologies and recommendations for improved practice." *Journal of Cleaner Production* 259 (2020): 120904.

Wender, BA., et al. "Sensitivity-based research prioritization through stochastic characterization modeling." *The International Journal of Life Cycle Assessment* 23.2 (2018): 324-332.

Case Study 1: Niacinamide

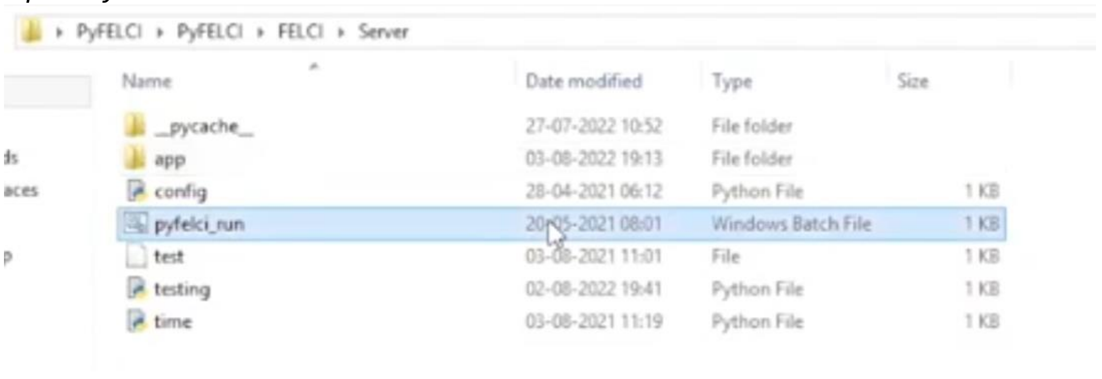
Among the first studies to employ Monte Carlo exploration of data uncertainties in USETox for the estimation of characterization factors in environmental life cycle assessment (LCA) of novel materials is Wender et al. (2018). Their approach identified critical contributors to uncertainty in the aquatic ecotoxicity characterization factors for niacinamide, a topical antioxidant, by automating Monte Carlo sampling of uncertain chemical parameters from user-defined probability distribution functions, and aggregating the USETox results. Comparative examination of Spearman rank correlations between sampled input parameters and variation in the resulting characterization factors allowed identification of those input uncertainties that make the greatest contribution to eventual uncertainty in the USETox results. Thus, the Wender et al. study established a *scientific* basis for prioritizing new research for reducing uncertainties in LCA of novel materials.

Although the Wender et al. study was an important intellectual antecedent to FELCI, it did not consider experimental costs. The principal contribution of the FELCI software was to add two additional dimensions to evaluation of research portfolios (i.e., combinations of experiment): 1) cost, and 2) BA level. With FELCI, research managers can explore the frontier of optimal trade-offs between scientific merit (measured by reduction of uncertainty) and experimental cost. Finally, the principal contribution of PyFELCI is to speed the same computations made by FELCI by enabling multi-core processing capabilities in Python. The additional speed, along with progress bars that estimate total time to completion, allow research managers to iterate on different scenarios or ideas, using the results of one run to inform the direction of new explorations.

To demonstrate the utility of PyFELCI and facilitate its use in new applications, this case study provides step-by-step instructions that recreate the niacinamide example, so that users can test their understanding of PyFELCI input and output screens prior to designing their own investigations.

Operating PyFELCI for Niacinamide

Open PyFELCI



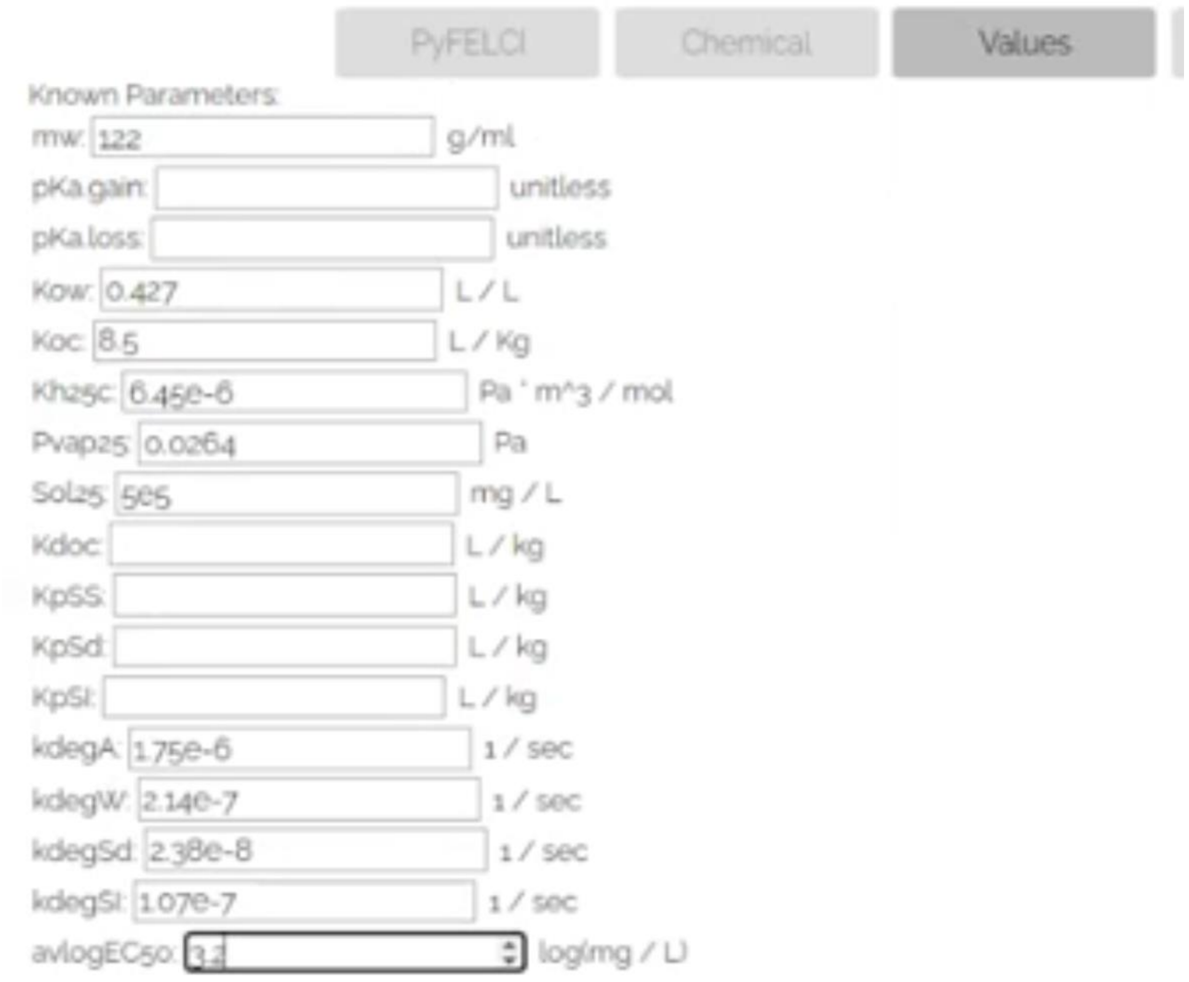
PyFELCI ▸ PyFELCI ▸ FELCI ▸ Server

Name	Date modified	Type	Size
__pycache__	27-07-2022 10:52	File folder	
app	03-08-2022 19:13	File folder	
config	28-04-2021 06:12	Python File	1 KB
pyfelci_run	20-05-2021 08:01	Windows Batch File	1 KB
test	03-08-2021 11:01	File	1 KB
testing	02-08-2022 19:41	Python File	1 KB
time	03-08-2021 11:19	Python File	1 KB

Figure 5. To run PyFELCI, double click the pyfelci_run batch file.

After installing PyFELCI (see Appendix A), open the pyfelci_run batch file found in the '>Server' subdirectory (Figure 5).

Values



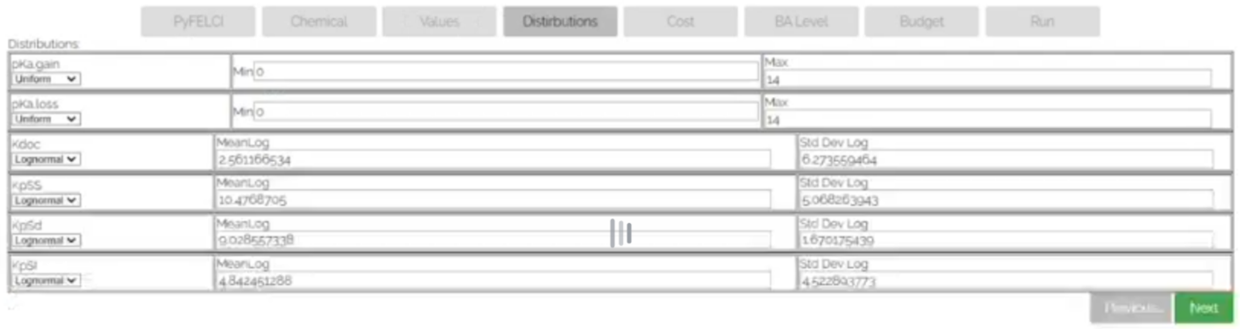
Known Parameters:

mw:	122	g/ml
pKa.gain:		unitless
pKa.loss:		unitless
Kow:	0.427	L / L
Koc:	8.5	L / Kg
Kh25c:	6.45e-6	Pa * m^3 / mol
Pvap25:	0.0264	Pa
Sol25:	5e5	mg / L
Kdoc:		L / kg
KpSS:		L / kg
KpSd:		L / kg
KpSl:		L / kg
kdegA:	1.75e-6	1 / sec
kdegW:	2.14e-7	1 / sec
kdegSd:	2.38e-8	1 / sec
kdegSl:	1.07e-7	1 / sec
avlogEC50:	3.2	log(mg / L)

Figure 6. Known chemical characteristics for niacinamide are entered as point estimates in the PyFELCI Chemical tab, leaving other values blank.

There are several chemical characteristics of niacinamide that are known, such as molecular weight. Enter the known chemical characteristics in the PyFELCI ‘Values’ tab, as show in Figure 6. Leave unknown values blank. These will be specified on the ‘Distributions’ tab.

Distributions



Distributions:

pKa.gain Uniform	Min 0	Max 14
pKa.loss Uniform	Min 0	Max 14
Kdoc Lognormal	MeanLog 2.561166534	Std Dev Log 6.273559464
KpSS Lognormal	MeanLog 10.4768705	Std Dev Log 5.068263943
KpSd Lognormal	MeanLog 9.028557338	Std Dev Log 1.670175439
KpSl Lognormal	MeanLog 4.842461288	Std Dev Log 4.522893773

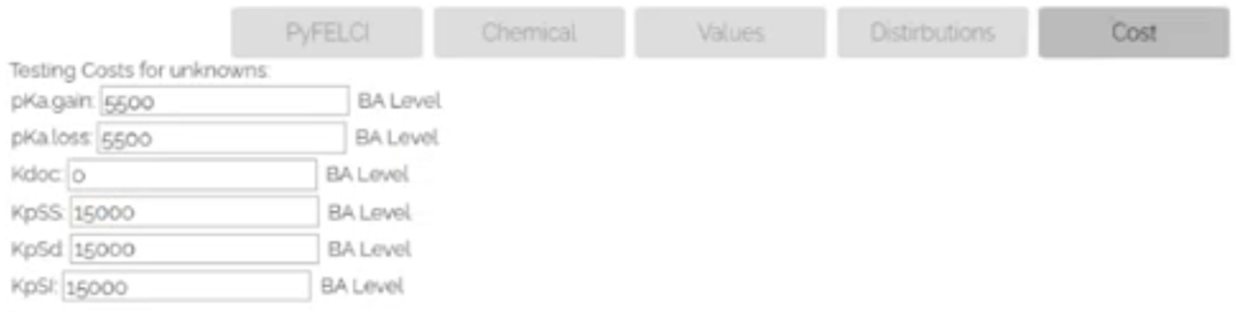
Figure 7. Specify unknown chemical characteristics as probability density functions using the 'Distributions' tab.

Unknown chemical characteristics must be entered as probability density functions (PDF). PyFELCI allows a variety of different PDF shapes, including uniform, triangular, normal, and log-normal.

For each unknown parameter, specify the shape of the PDF and the parameters that characterize that shape. For example, in the screen below, nothing is known about the pKa characteristics of niacinamide. Therefore, the shape of the PDF selected is uniform. Minimum and maximum values are specified as 0 and 14, respectively, because these represent the extreme limits of possible pKa values. In this case, the uncertainty in pKa is extreme.

By contrast, other PDFs are specified as lognormal, and parameterized with the mean and standard deviation of the log of the PDF.

Cost



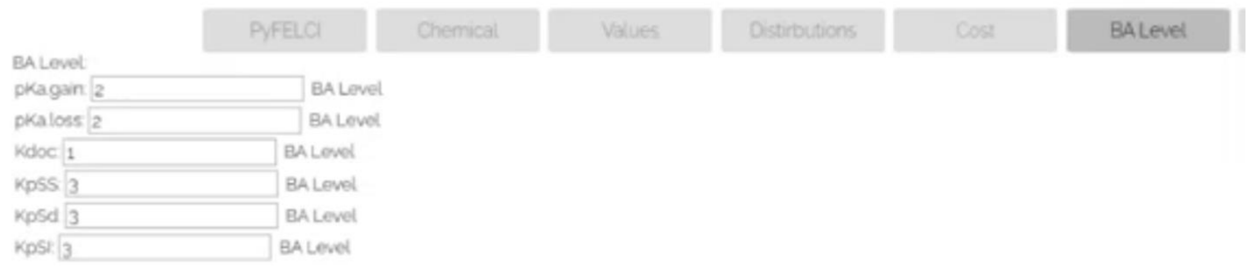
Testing Costs for unknowns:

pKa.gain:	5500	BA Level
pKa.loss:	5500	BA Level
Kdoc:	0	BA Level
KpSS:	15000	BA Level
KpSd:	15000	BA Level
KpSl:	15000	BA Level

Figure 8. Enter the cost to experimentally determine more precise estimates of the uncertain chemical characteristics.

In this example, we have no special insight into the experimental cost of determining more precise values of the unknown values. Although the unique contribution of PyFELCI is its ability to compare the value of experimental information with its experimental cost, the arbitrary values we input into PyFELCI in this example serve only to illustrate this capacity. Users may input their own values, and interpret the results accordingly. Note that the only prompts on the 'Cost' tab (Figure 8) are those that were specified earlier as probability distributions, as known point values specified on the 'Values' tab require no further experimental investment.

BA Level

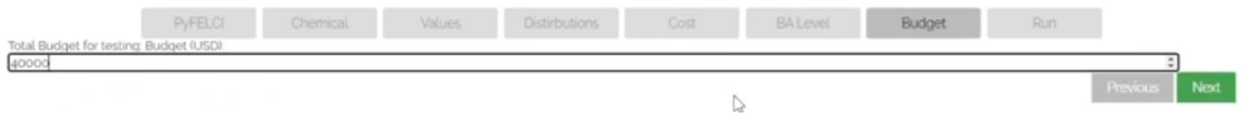


BA Level:
pKa gain: 2 BA Level
pKa loss: 2 BA Level
Kdoc: 1 BA Level
KpSS: 3 BA Level
KpSd: 3 BA Level
KpSl: 3 BA Level

Figure 9. Enter any other numerical prioritization levels on the 'BA Level' tab.

Because cost may not be the only system of determining experimental priorities, PyFELCI allows a second approach to investigating value of information tradeoffs on the 'BA Level' tab (Figure 9). The units allowable on this screen accommodate any numerical scale the user applies, although in this example we will not revisit the BA Level during consideration of the results.

Budget

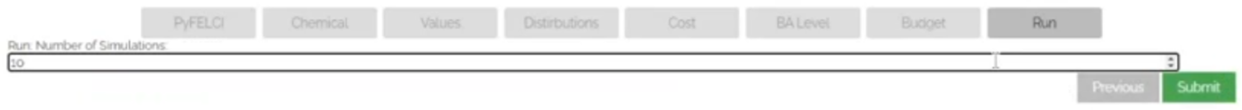


Total Budget for testing: Budget (USD)
40000

Figure 10. Enter a total experimental budget on the 'Budget' tab.

One of the advantages of PyFELCI is its capacity to compare accumulated costs of various experimental combinations (called 'research portfolios') to an overall budget number (Figure 10), facilitating optimal investment of available funds. Additionally, PyFELCI will show any gains in value of information that are available at expenditure levels *above* the specified figure, which may fortify internal arguments for upward budget adjustments where larges gains in information can be had.

Run (Simulations)

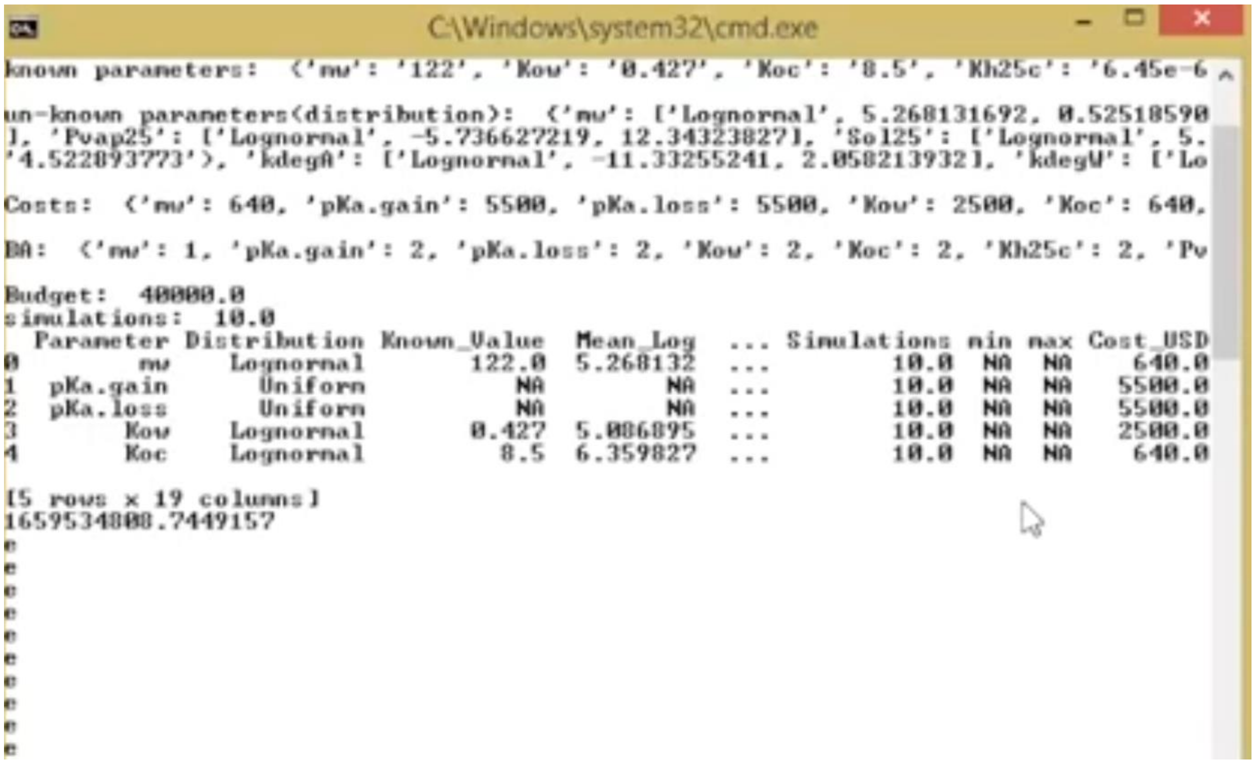


Run: Number of Simulations:
10

Figure 11. Enter an integer corresponding to the number of Monte Carlo simulations to run in USETox.

PyFELCI allows users to specify the number of simulations to explore within USTETox. A larger number of simulations will take longer to run, but result in more stable results (with fewer random variations). In this example (Figure 11), we run only 10 simulations to minimize computation time. Later, we can check for stability of the results in the ranking of the research portfolios after only 10 simulations and decide to rerun results at fewer or more simulations.

Run-time feedback



```
known parameters:  {'mw': '122', 'Kow': '0.427', 'Koc': '8.5', 'Kh25c': '6.45e-6

un-known parameters(distribution):  {'mw': ['Lognormal', 5.268131692, 0.52518590
], 'Pvap25': ['Lognormal', -5.736627219, 12.34323827], 'Sol25': ['Lognormal', 5.
'4.522893773'}, 'kdegA': ['Lognormal', -11.33255241, 2.058213932], 'kdegW': ['Lo

Costs:  {'mw': 640, 'pKa.gain': 5500, 'pKa.loss': 5500, 'Kow': 2500, 'Koc': 640,

BA:  {'mw': 1, 'pKa.gain': 2, 'pKa.loss': 2, 'Kow': 2, 'Koc': 2, 'Kh25c': 2, 'Pv

Budget:  40000.0
simulations:  10.0
  Parameter Distribution Known_Value  Mean_Log  ... Simulations min max Cost_USD
0        mw    Lognormal       122.0  5.268132  ...        10.0  NA  NA    640.0
1  pKa.gain      Uniform          NA        NA  ...        10.0  NA  NA   5500.0
2  pKa.loss      Uniform          NA        NA  ...        10.0  NA  NA   5500.0
3       Kow    Lognormal       0.427  5.086895  ...        10.0  NA  NA   2500.0
4       Koc    Lognormal         8.5  6.359827  ...        10.0  NA  NA    640.0

[5 rows x 19 columns]
1659534808.7449157
e
e
e
e
e
e
e
e
e
e
```

Figure 12. The operations of the pyfelci_run batch files are printed in a command dialog box to facilitate troubleshooting. Under ordinary circumstances, these dialog boxes can be ignored.

PyFELCI adds several indicators of progress that were absent in FELCI, including a command line dialog bopx (Figure 12) and different representations of progress bars. These assist the user in planning their time during longer runs requiring additional simulations.

Value of Information Outputs

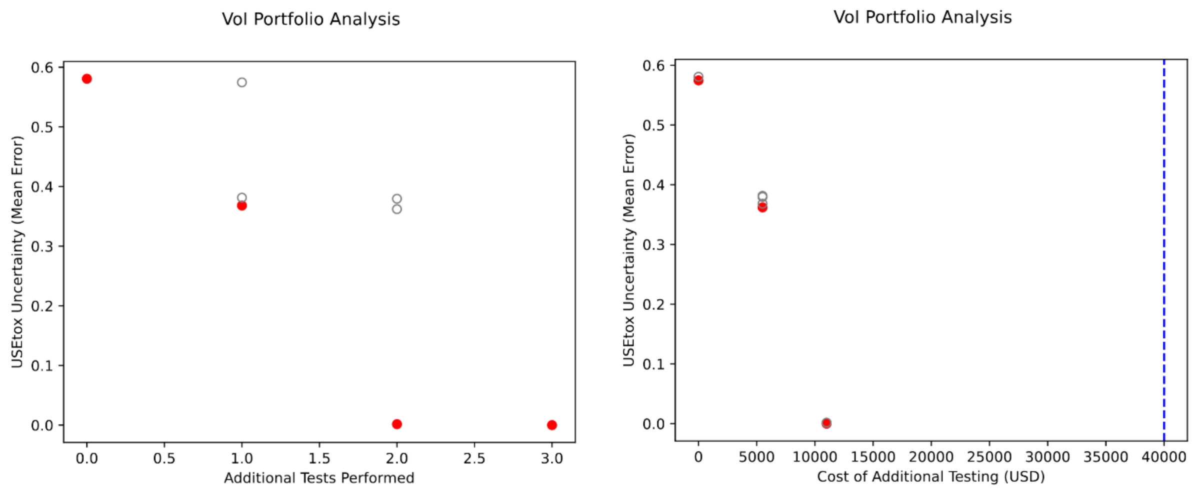


Figure 13. Value of information results are presented in two ways. The first (left) compares reduction in uncertainties relative to the total *number* of experiments required, while the second (right) compares to the *cost* of experiments.

The value of information results for niacinamide are presented relative to the number of experiments required to reduce uncertainty (Figure 13, left) and the cost of those experiments (Figure 13, right). The red dots represent the frontier of optimal (i.e., “non-dominated”) choices.

In this case, just *one* experiment reduces uncertainty, as measured by the mean error of the resulting distribution of characterization factors produced by USETox, by more than a third, and two experiments result in near certainty. No further benefit is derived from three experiments, as error has already been so greatly reduced. Moreover, the recommended experiments can be accomplished for far less than the budget available.

Stability

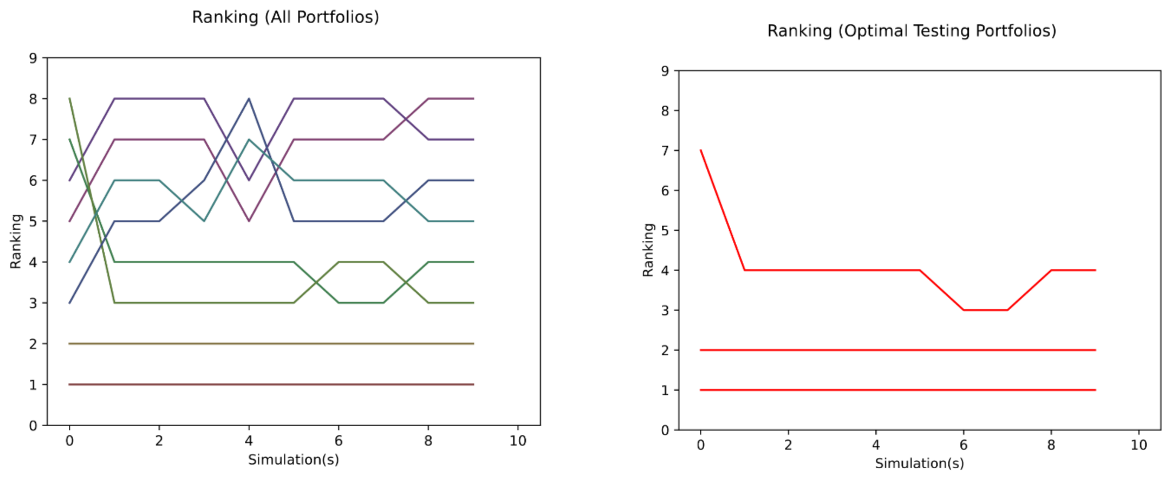
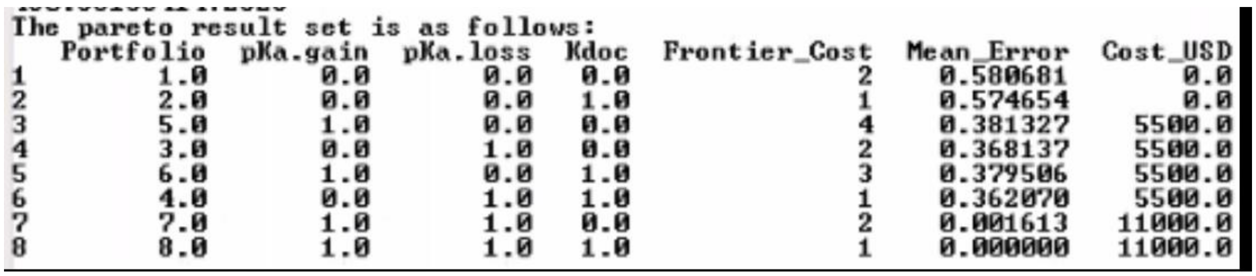


Figure 14. PyFELCI represents the stability of rankings in two ways. On the left are the rankings of all possible combinations of experiments (i.e., research portfolios), including sub-optimal (or dominated) choices. On the right, the sub-optimal portfolios are removed to show the stability of just the non-dominated options represented by red dots in the Value of Information graphs.

The nature of Monte Carlo sampling introduces the possibility that results are an artifact of random sampling, rather than a robust and reliable representation of uncertainty. As the number of simulations increases, both computational requirements and reliability of results increase – to a point. After reliability is already established, additional simulations only lengthens run times. Therefore, it is advantageous to run the fewest number of simulations that yield reliable results. Fortunately, the absolute estimates of mean error are unimportant compared to the relative ranking of the research portfolios. Because relative ranking can be established in fewer simulations than reliable estimates of mean error, PyFELCI can operate well with fewer simulations than is typical of most Monte Carlo simulations – speeding the time to obtain results. In this case, the top two research portfolios are reliably established in the first simulation, and remain constant even as the simulations increase. The correct inference is that the results in Figure 13 are reliable and repeatable, and no further simulations are required to minimize random samplings errors.

Recommendations



```
The pareto result set is as follows:
   Portfolio  pKa.gain  pKa.loss  Kdoc  Frontier_Cost  Mean_Error  Cost_USD
1        1.0       0.0       0.0   0.0              2    0.580681       0.0
2        2.0       0.0       0.0   1.0              1    0.574654       0.0
3        5.0       1.0       0.0   0.0              4    0.381327    5500.0
4        3.0       0.0       1.0   0.0              2    0.368137    5500.0
5        6.0       1.0       0.0   1.0              3    0.379506    5500.0
6        4.0       0.0       1.0   1.0              1    0.362070    5500.0
7        7.0       1.0       1.0   0.0              2    0.001613   11000.0
8        8.0       1.0       1.0   1.0              1    0.000000   11000.0
```

Figure 14. The recommended experiments are tabulated in combinations called research portfolios. The columns correspond to the experimental investigations, wherein a value of "1" indicates that the experiment in that column belongs to the research portfolio in that row. In this case, the optimal combination of two experiments is research portfolio #7, which combines testing for pKa.gain and pKa.loss. Adding Kdoc to the portfolio (#8) results in a miniscule improvement in mean error.

While the red dots represent the optimal frontier identifying the experimental combinations that most reduce uncertainty for least cost, the figures alone do not name the experiments. To extract recommendations from the results, it is necessary to interpret the data tabulated in Figure 14.

Each column represents an experimental investigation. PyFELCI constructs portfolios by testing the experiments individually, and in combination. For example, portfolio #1 shows the uncertainty without any new experimental investigations, and #2 presents the Mean Error results when only Kdoc is established experimentally. In this case, the reduction to Mean Error is miniscule when Kdoc is investigated alone.

A value of "1.0" under the column header corresponding to each experiment indicates that experiment is included in the research portfolio corresponding to that row, whereas a value of "0.0" indicates that it is excluded.

Hypothetically, the massive uncertainty in pKa.gain and pKa.loss, originally entered in the 'Distribution' tab as a uniform distribution between 0 and 14 could be determined experimentally in the lab. These results indicate that these pKa parameters are the principal contributor to uncertainty in the characterization factor for niacinamide estimated by USETox, and that research portfolio #7 that combines testing of both pKa.gain and pKa.loss results in a massive reduction of Mean Error and a cost of $11,000.

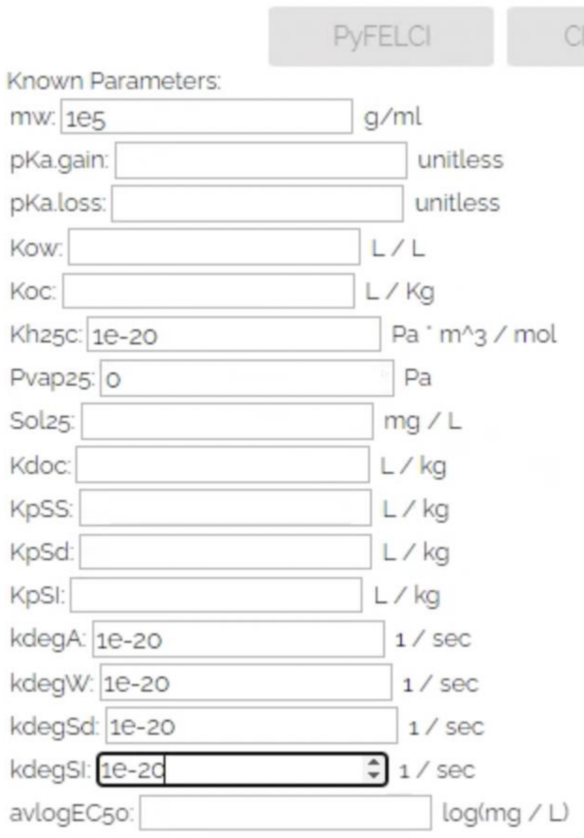
Case Study 2: Carbon Nanotubes

Nanomaterials are among the most vexing to assess from an environmental life-cycle perspective. The wide variety of chemical and structural alternatives available makes possible a myriad of unique materials that may perform in surprising ways. Therefore, it may be advantageous to employ USETox via PyFELCI to prioritize new experimental investigations when developing novel nanomaterials.

The second case study investigates carbon nanotube characterization factors, following the examples given by Eckelman et al. (2012).

Open PyFELCI in the same way as the previous case study.

Values

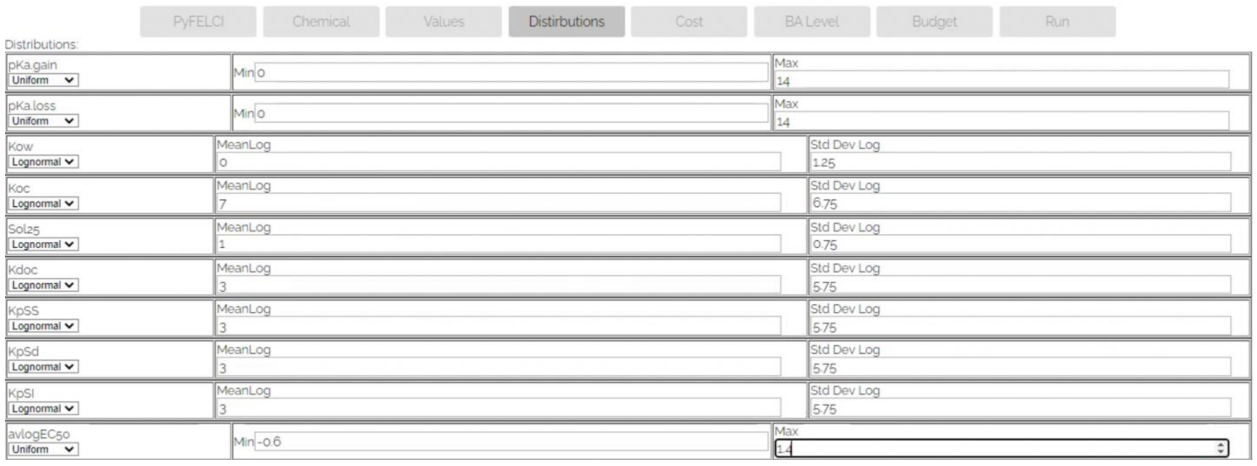


Known Parameters:

mw:	1e5	g/ml
pKa.gain:		unitless
pKa.loss:		unitless
Kow:		L / L
Koc:		L / Kg
Kh25c:	1e-20	Pa * m^3 / mol
Pvap25:	0	Pa
Sol25:		mg / L
Kdoc:		L / kg
KpSS:		L / kg
KpSd:		L / kg
KpSl:		L / kg
kdegA:	1e-20	1 / sec
kdegW:	1e-20	1 / sec
kdegSd:	1e-20	1 / sec
kdegSl:	1e-20	1 / sec
avlogEC50:		log(mg / L)

Figure 15. Enter known information in the ‘Values’ tab. The carbon nanotube case study involves less known information than niacinamide, which means that there are additional blanks to fill in as distributions on the next screen.

Distributions

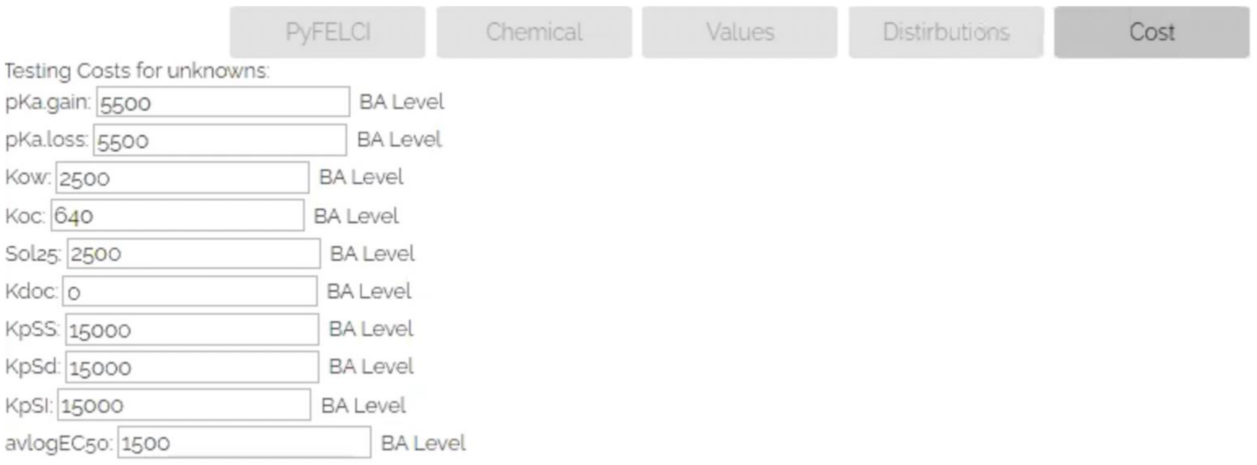


Distributions:

pKa.gain Uniform	Min 0	Max 14
pKa.loss Uniform	Min 0	Max 14
Kow Lognormal	MeanLog 0	Std Dev Log 1.25
Koc Lognormal	MeanLog 7	Std Dev Log 6.75
Sol25 Lognormal	MeanLog 1	Std Dev Log 0.75
Kdoc Lognormal	MeanLog 3	Std Dev Log 5.75
KpSS Lognormal	MeanLog 3	Std Dev Log 5.75
KpSd Lognormal	MeanLog 3	Std Dev Log 5.75
KpSl Lognormal	MeanLog 3	Std Dev Log 5.75
avlogEC50 Uniform	Min -0.6	Max 1.4

Figure 16. Unknown chemical characteristics are specified as probability density functions on the 'Distributions' tab. The pKa.gain and pKa.loss are given as uniform distributions, while all others are specified as log normal, from values given in Eckelmen et al. (2012).

Cost

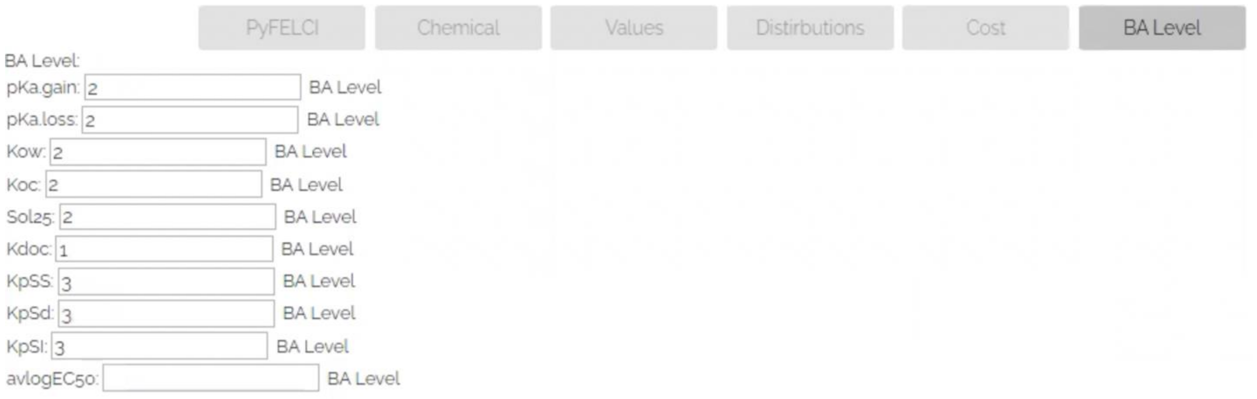


Testing Costs for unknowns:

pKa.gain:	5500	BA Level
pKa.loss:	5500	BA Level
Kow:	2500	BA Level
Koc:	640	BA Level
Sol25:	2500	BA Level
Kdoc:	0	BA Level
KpSS:	15000	BA Level
KpSd:	15000	BA Level
KpSl:	15000	BA Level
avlogEC50:	1500	BA Level

Figure 17. Enter the estimated cost of laboratory determination of unknown chemical characteristics.

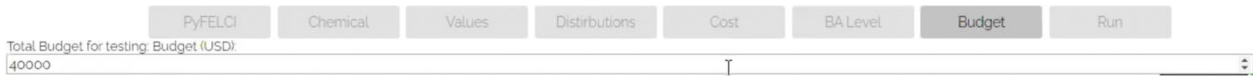
BA Level



BA Level:		
pKa.gain:	2	BA Level
pKa.loss:	2	BA Level
Kow:	2	BA Level
Koc:	2	BA Level
Sol25:	2	BA Level
Kdoc:	1	BA Level
KpSS:	3	BA Level
KpSd:	3	BA Level
KpSl:	3	BA Level
avlogEC50:		BA Level

Figure 18. As in the niacinamide case study, BA Levels refer to internal, non-budgetary priorities. The values listed here are arbitrary.

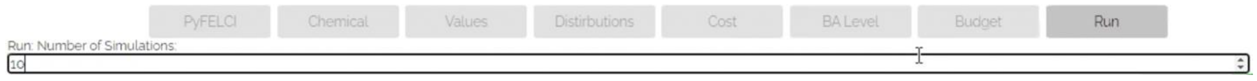
Budget



Total Budget for testing: Budget (USD):
40000

Figure 19. Enter a total laboratory experimental budget, to facilitate planning.

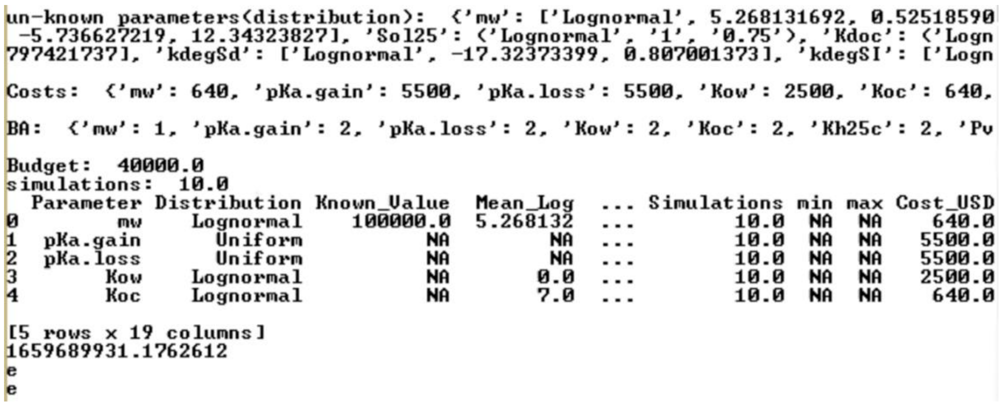
Run (Simulations)



Run: Number of Simulations:
10

Figure 20. Enter 10 simulations to speed computations, then check the results for stability when PyFELCI is complete.

Run-time feedback



```
un-known parameters(distribution):  {'mw': ['Lognormal', 5.268131692, 0.52518590
 -5.736627219, 12.34323827], 'Sol25': ('Lognormal', '1', '0.75'), 'Kdoc': ('Logn
797421737], 'kdegSd': ['Lognormal', -17.32373399, 0.807001373], 'kdegSI': ['Logn

Costs:  {'mw': 640, 'pKa.gain': 5500, 'pKa.loss': 5500, 'Kow': 2500, 'Koc': 640,

BA:  {'mw': 1, 'pKa.gain': 2, 'pKa.loss': 2, 'Kow': 2, 'Koc': 2, 'Kh25c': 2, 'Pv

Budget:  40000.0
simulations:  10.0
  Parameter Distribution Known_Value  Mean_Log  ... Simulations min max Cost_USD
0        mw    Lognormal    100000.0  5.268132  ...        10.0  NA  NA    640.0
1  pKa.gain      Uniform          NA        NA  ...        10.0  NA  NA   5500.0
2  pKa.loss      Uniform          NA        NA  ...        10.0  NA  NA   5500.0
3       Kow    Lognormal          NA       0.0  ...        10.0  NA  NA   2500.0
4       Koc    Lognormal          NA       7.0  ...        10.0  NA  NA    640.0

[5 rows x 19 columns]
1659689931.1762612
e
e
```

Figure 21. PyFELCI provides runtime information in an open dialog box, as well as two types of progress bars.

Value of Information Outputs

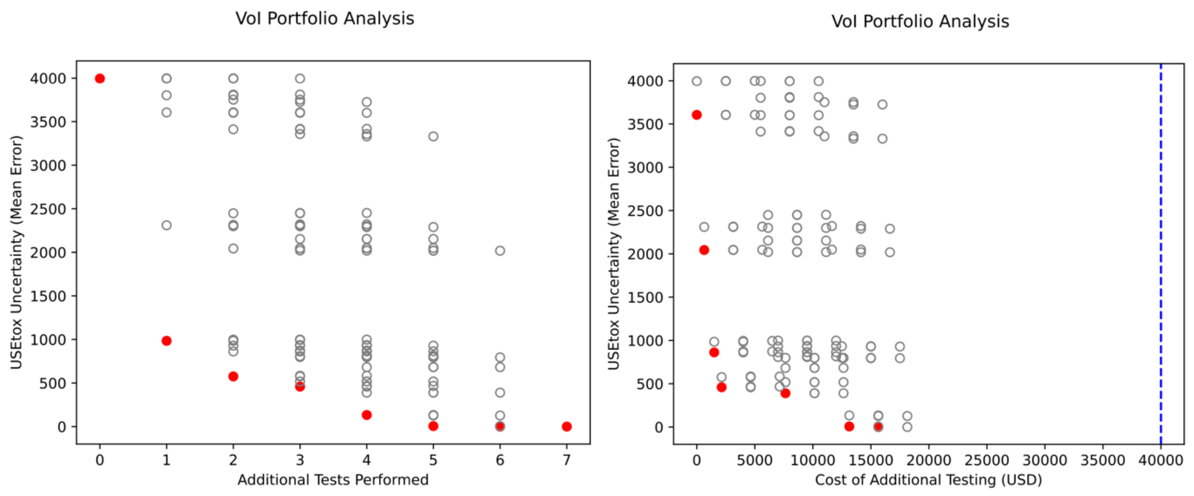


Figure 22. Compared to the niacinamide case study, the CNT case study results are more complicated, due to the greater number of unknowns.

Results in Figure 22 (left) indicate that proper selection of just *one* experiment results in a 75% reduction in mean error, as evidenced by the red dot corresponding to the point (1, 1000). Several inferior options are also available, some of which fail to reduce error at all. Both value of information representations show that optimal selection of experimental effort will conserve resources, and that diminishing returns exist after about 3 or 4 experiments.

Stability

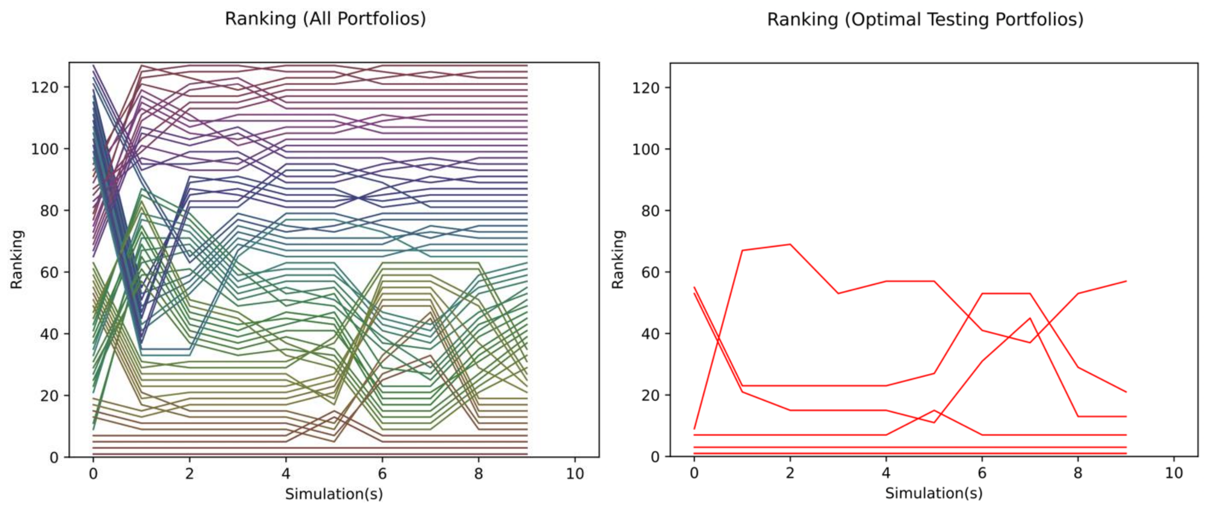
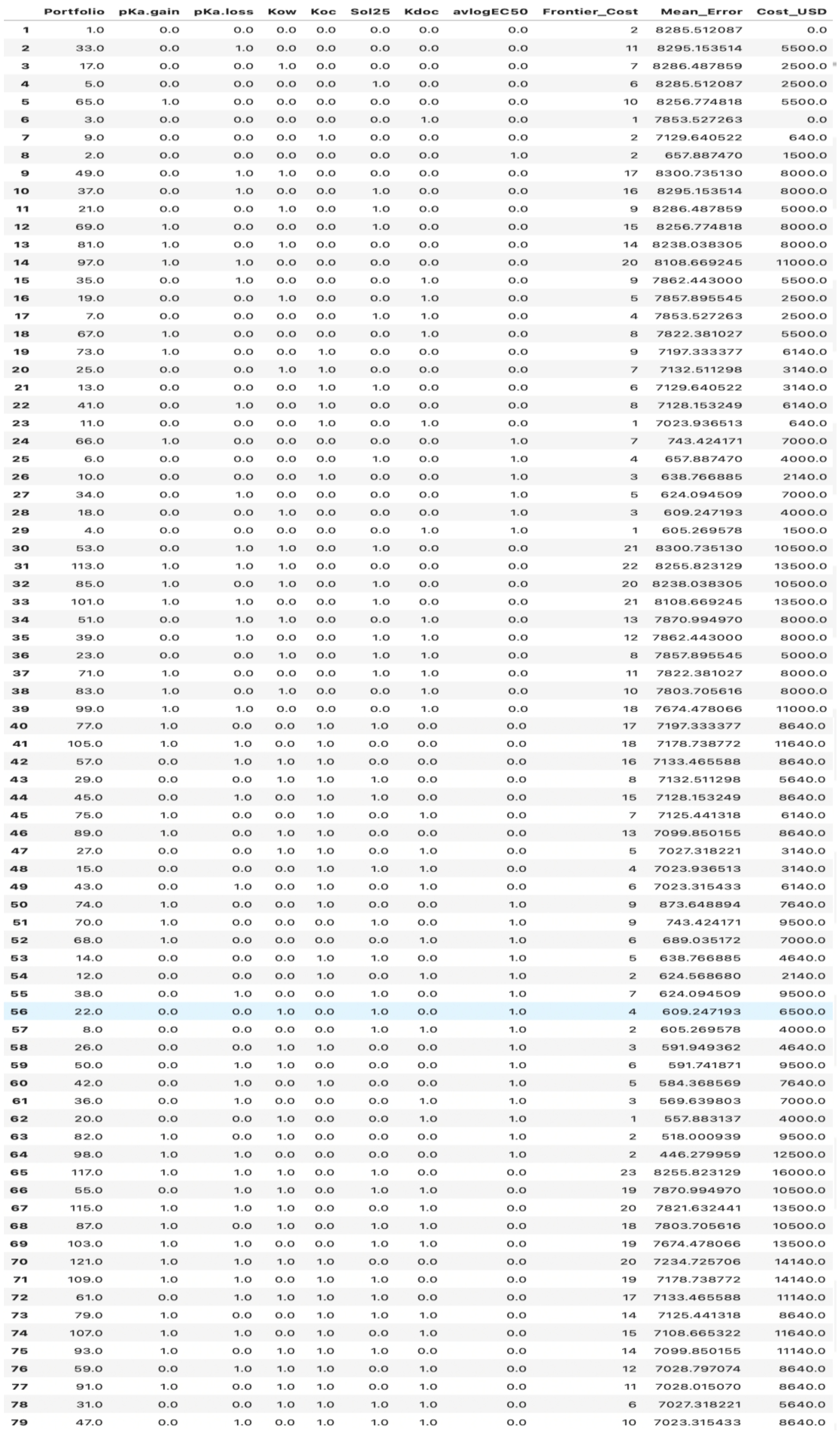
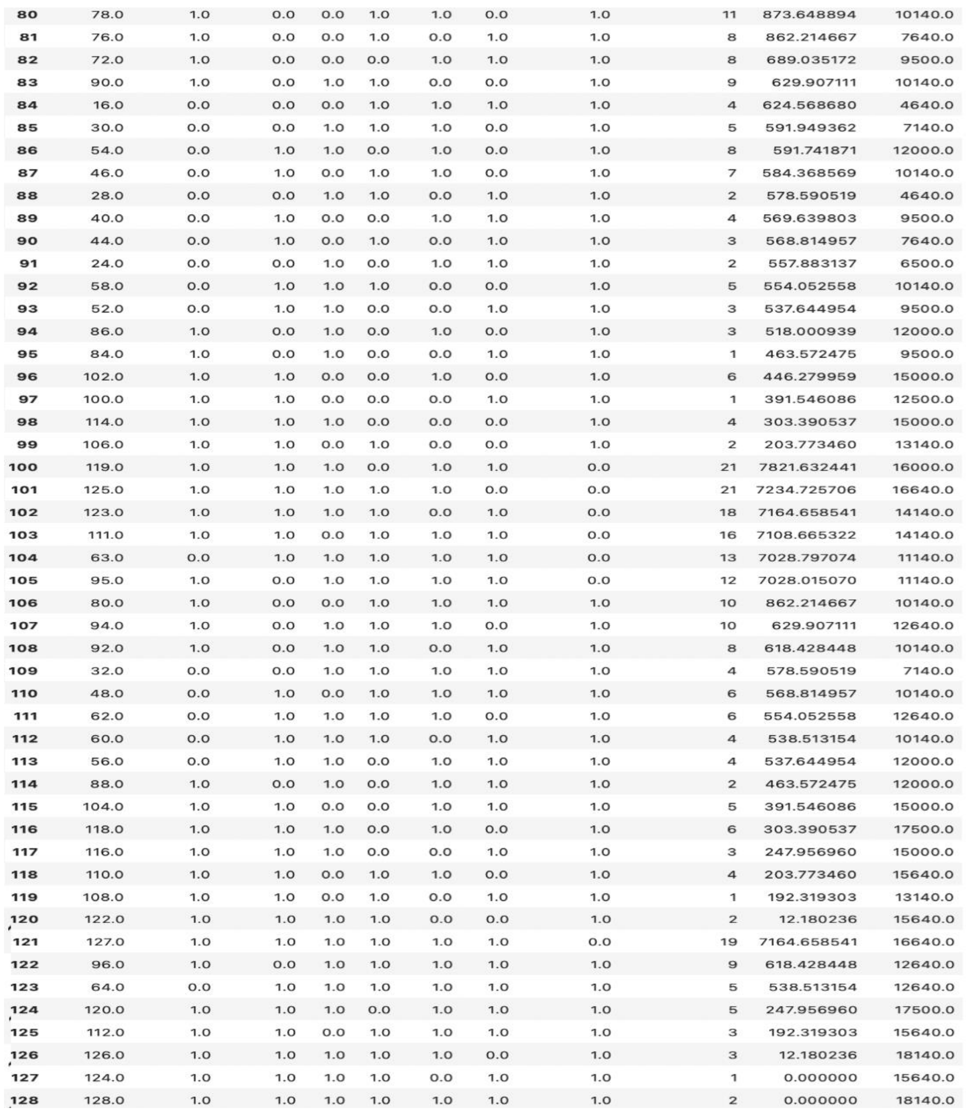


Figure 23. The first two research portfolio rankings are stable for all simulations, 1-10. However, some rank reversal is evident in the third or fourth best research portfolios (right) represented by red dots in Figure 22.

Recommendations



	Portfolio	pKa.gain	pKa.loss	Kow	Koc	Sol25	Kdoc	avlogEC50	Frontier_Cost	Mean_Error	Cost_USD
1	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2	8285.512087	0.0
2	33.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	11	8295.153514	5500.0
3	17.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	7	8286.487859	2500.0
4	5.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	6	8285.512087	2500.0
5	65.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	10	8256.774818	5500.0
6	3.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1	7853.527263	0.0
7	9.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	2	7129.640522	640.0
8	2.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	2	657.887470	1500.0
9	49.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	17	8300.735130	8000.0
10	37.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	16	8295.153514	8000.0
11	21.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	9	8286.487859	5000.0
12	69.0	1.0	0.0	0.0	0.0	1.0	0.0	0.0	15	8256.774818	8000.0
13	81.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	14	8238.038305	8000.0
14	97.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	20	8108.669245	11000.0
15	35.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	9	7862.443000	5500.0
16	19.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	5	7857.895545	2500.0
17	7.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	4	7853.527263	2500.0
18	67.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	8	7822.381027	5500.0
19	73.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	9	7197.333377	6140.0
20	25.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	7	7132.511298	3140.0
21	13.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	6	7129.640522	3140.0
22	41.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	8	7128.153249	6140.0
23	11.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1	7023.936513	640.0
24	66.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	7	743.424171	7000.0
25	6.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	4	657.887470	4000.0
26	10.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	3	638.766885	2140.0
27	34.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	5	624.094509	7000.0
28	18.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	3	609.247193	4000.0
29	4.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1	605.269578	1500.0
30	53.0	0.0	1.0	1.0	0.0	1.0	0.0	0.0	21	8300.735130	10500.0
31	113.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	22	8255.823129	13500.0
32	85.0	1.0	0.0	1.0	0.0	1.0	0.0	0.0	20	8238.038305	10500.0
33	101.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0	21	8108.669245	13500.0
34	51.0	0.0	1.0	1.0	0.0	0.0	1.0	0.0	13	7870.994970	8000.0
35	39.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	12	7862.443000	8000.0
36	23.0	0.0	0.0	1.0	0.0	1.0	1.0	0.0	8	7857.895545	5000.0
37	71.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	11	7822.381027	8000.0
38	83.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	10	7803.705616	8000.0
39	99.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	18	7674.478066	11000.0
40	77.0	1.0	0.0	0.0	1.0	1.0	0.0	0.0	17	7197.333377	8640.0
41	105.0	1.0	1.0	0.0	1.0	0.0	0.0	0.0	18	7178.738772	11640.0
42	57.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	16	7133.465588	8640.0
43	29.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	8	7132.511298	5640.0
44	45.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	15	7128.153249	8640.0
45	75.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	7	7125.441318	6140.0
46	89.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	13	7099.850155	8640.0
47	27.0	0.0	0.0	1.0	1.0	0.0	1.0	0.0	5	7027.318221	3140.0
48	15.0	0.0	0.0	0.0	1.0	1.0	1.0	0.0	4	7023.936513	3140.0
49	43.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0	6	7023.315433	6140.0
50	74.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	9	873.648894	7640.0
51	70.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	9	743.424171	9500.0
52	68.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	6	689.035172	7000.0
53	14.0	0.0	0.0	0.0	1.0	1.0	0.0	1.0	5	638.766885	4640.0
54	12.0	0.0	0.0	0.0	1.0	0.0	1.0	1.0	2	624.568680	2140.0
55	38.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	7	624.094509	9500.0
56	22.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	4	609.247193	6500.0
57	8.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	2	605.269578	4000.0
58	26.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0	3	591.949362	4640.0
59	50.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	6	591.741871	9500.0
60	42.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	5	584.368569	7640.0
61	36.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	3	569.639803	7000.0
62	20.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	1	557.883137	4000.0
63	82.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	2	518.000939	9500.0
64	98.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	2	446.279959	12500.0
65	117.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0	23	8255.823129	16000.0
66	55.0	0.0	1.0	1.0	0.0	1.0	1.0	0.0	19	7870.994970	10500.0
67	115.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0	20	7821.632441	13500.0
68	87.0	1.0	0.0	1.0	0.0	1.0	1.0	0.0	18	7803.705616	10500.0
69	103.0	1.0	1.0	0.0	0.0	1.0	1.0	0.0	19	7674.478066	13500.0
70	121.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	20	7234.725706	14140.0
71	109.0	1.0	1.0	0.0	1.0	1.0	0.0	0.0	19	7178.738772	14140.0
72	61.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0	17	7133.465588	11140.0
73	79.0	1.0	0.0	0.0	1.0	1.0	1.0	0.0	14	7125.441318	8640.0
74	107.0	1.0	1.0	0.0	1.0	0.0	1.0	0.0	15	7108.665322	11640.0
75	93.0	1.0	0.0	1.0	1.0	1.0	0.0	0.0	14	7099.850155	11140.0
76	59.0	0.0	1.0	1.0	1.0	0.0	1.0	0.0	12	7028.797074	8640.0
77	91.0	1.0	0.0	1.0	1.0	0.0	1.0	0.0	11	7028.015070	8640.0
78	31.0	0.0	0.0	1.0	1.0	1.0	1.0	0.0	6	7027.318221	5640.0
79	47.0	0.0	1.0	0.0	1.0	1.0	1.0	0.0	10	7023.315433	8640.0



80	78.0	1.0	0.0	0.0	1.0	1.0	0.0	1.0	11	873.648894	10140.0
81	76.0	1.0	0.0	0.0	1.0	0.0	1.0	1.0	8	862.214667	7640.0
82	72.0	1.0	0.0	0.0	0.0	1.0	1.0	1.0	8	689.035172	9500.0
83	90.0	1.0	0.0	1.0	1.0	0.0	0.0	1.0	9	629.907111	10140.0
84	16.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	4	624.568680	4640.0
85	30.0	0.0	0.0	1.0	1.0	1.0	0.0	1.0	5	591.949362	7140.0
86	54.0	0.0	1.0	1.0	0.0	1.0	0.0	1.0	8	591.741871	12000.0
87	46.0	0.0	1.0	0.0	1.0	1.0	0.0	1.0	7	584.368569	10140.0
88	28.0	0.0	0.0	1.0	1.0	0.0	1.0	1.0	2	578.590519	4640.0
89	40.0	0.0	1.0	0.0	0.0	1.0	1.0	1.0	4	569.639803	9500.0
90	44.0	0.0	1.0	0.0	1.0	0.0	1.0	1.0	3	568.814957	7640.0
91	24.0	0.0	0.0	1.0	0.0	1.0	1.0	1.0	2	557.883137	6500.0
92	58.0	0.0	1.0	1.0	1.0	0.0	0.0	1.0	5	554.052558	10140.0
93	52.0	0.0	1.0	1.0	0.0	0.0	1.0	1.0	3	537.644954	9500.0
94	86.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	3	518.000939	12000.0
95	84.0	1.0	0.0	1.0	0.0	0.0	1.0	1.0	1	463.572475	9500.0
96	102.0	1.0	1.0	0.0	0.0	1.0	0.0	1.0	6	446.279959	15000.0
97	100.0	1.0	1.0	0.0	0.0	0.0	1.0	1.0	1	391.546086	12500.0
98	114.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0	4	303.390537	15000.0
99	106.0	1.0	1.0	0.0	1.0	0.0	0.0	1.0	2	203.773460	13140.0
100	119.0	1.0	1.0	1.0	0.0	1.0	1.0	0.0	21	7821.632441	16000.0
101	125.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	21	7234.725706	16640.0
102	123.0	1.0	1.0	1.0	1.0	0.0	1.0	0.0	18	7164.658541	14140.0
103	111.0	1.0	1.0	0.0	1.0	1.0	1.0	0.0	16	7108.665322	14140.0
104	63.0	0.0	1.0	1.0	1.0	1.0	1.0	0.0	13	7028.797074	11140.0
105	95.0	1.0	0.0	1.0	1.0	1.0	1.0	0.0	12	7028.015070	11140.0
106	80.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0	10	862.214667	10140.0
107	94.0	1.0	0.0	1.0	1.0	1.0	0.0	1.0	10	629.907111	12640.0
108	92.0	1.0	0.0	1.0	1.0	0.0	1.0	1.0	8	618.428448	10140.0
109	32.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	4	578.590519	7140.0
110	48.0	0.0	1.0	0.0	1.0	1.0	1.0	1.0	6	568.814957	10140.0
111	62.0	0.0	1.0	1.0	1.0	1.0	0.0	1.0	6	554.052558	12640.0
112	60.0	0.0	1.0	1.0	1.0	0.0	1.0	1.0	4	538.513154	10140.0
113	56.0	0.0	1.0	1.0	0.0	1.0	1.0	1.0	4	537.644954	12000.0
114	88.0	1.0	0.0	1.0	0.0	1.0	1.0	1.0	2	463.572475	12000.0
115	104.0	1.0	1.0	0.0	0.0	1.0	1.0	1.0	5	391.546086	15000.0
116	118.0	1.0	1.0	1.0	0.0	1.0	0.0	1.0	6	303.390537	17500.0
117	116.0	1.0	1.0	1.0	0.0	0.0	1.0	1.0	3	247.956960	15000.0
118	110.0	1.0	1.0	0.0	1.0	1.0	0.0	1.0	4	203.773460	15640.0
119	108.0	1.0	1.0	0.0	1.0	0.0	1.0	1.0	1	192.319303	13140.0
120	122.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	2	12.180236	15640.0
121	127.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	19	7164.658541	16640.0
122	96.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	9	618.428448	12640.0
123	64.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	5	538.513154	12640.0
124	120.0	1.0	1.0	1.0	0.0	1.0	1.0	1.0	5	247.956960	17500.0
125	112.0	1.0	1.0	0.0	1.0	1.0	1.0	1.0	3	192.319303	15640.0
126	126.0	1.0	1.0	1.0	1.0	1.0	0.0	1.0	3	12.180236	18140.0
127	124.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	1	0.000000	15640.0
128	128.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2	0.000000	18140.0

Figure 24. Over one hundred different research portfolios are tested in the CNT case study.

In the CNT case, large reductions in uncertainty can be obtained from four or fewer experimental results, after which diminishing returns set in. Inspection of the 'Frontier Cost' column guides the reader towards those portfolios that lie along the optimum information/cost frontier, which are indicated with a "1". For example, research portfolio #4 (line 29, Figure 24) tests only Kdoc and avglogEC50, and obtains over 80% of available error reduction. Adding pKa.gain and pKa.loss (research portfolio #100, line 97 in Figure 24) further reduces errors to the point at which they can hardly be reduced further.

APPENDIX A

PyFELCI Installation & Testing Instructions

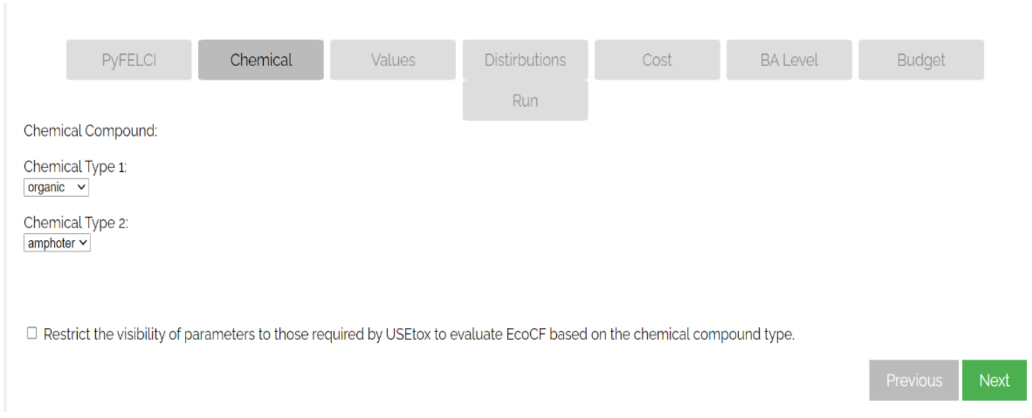
Introduction

The *PyFELCI* tool is developed in Python and uses an open-source framework WinPython to make it a portable application.

To prepare your machine to run *PyFELCI*, first close all instances of Excel. Other open workbooks may interfere with the running of *USEtox*.

Step-by-Step Instructions:

1) Download the PyFELCI.rar file from box.com at: https://app.box.com/file/808401523528?s=2sl7iqq7f8udt02agi5os1e1prq1krjc
2) Extract the folder file *PyFELCI* (suggested software: WinRARhttps://www.win-rar.com/download.html?&L=0)
3) Go to folder FELCI -> Server
4) Double click on the pyfelci_run.bat file.
5) Then open http://127.0.0.1:5000/ either on Google Chrome or Microsoft Edge. (This was successfully tested on both).
6) Fill each tab with the appropriate values and then start the simulations. In the below steps we see an example run for 100 simulations. Fill the below values in each tab.
7) **Chemical Tab:** select "Organic" for Chem Type 1 and "Amphoter" for Chem Type 2. Then press Next.



Chemical Compound:

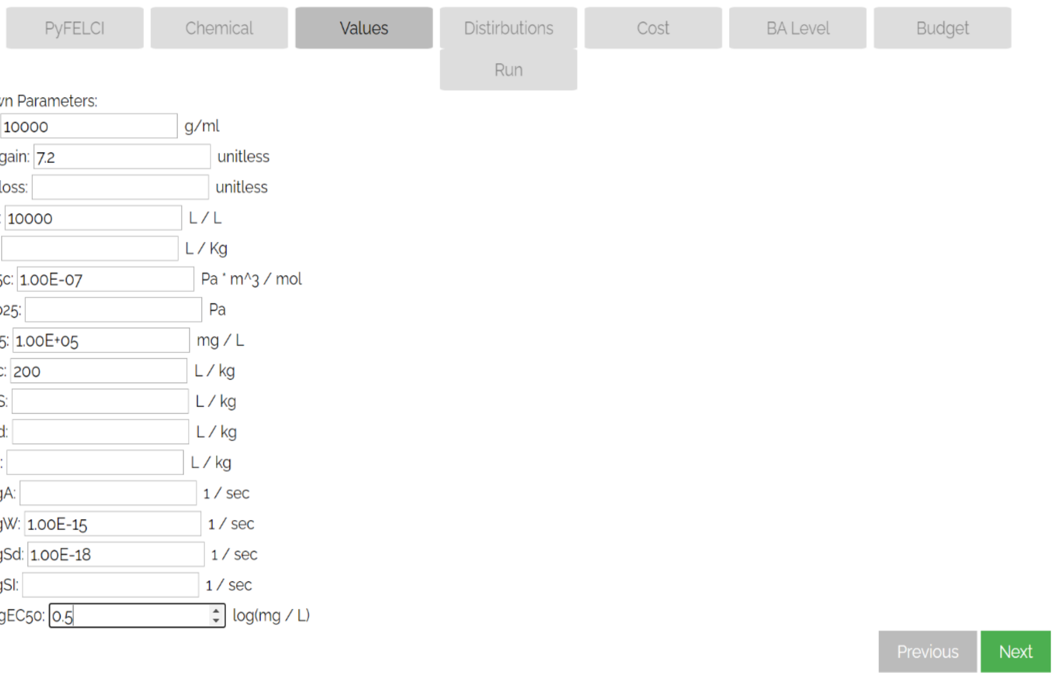
Chemical Type 1: organic

Chemical Type 2: amphoter

☐ Restrict the visibility of parameters to those required by USEtox to evaluate EcoCF based on the chemical compound type.

8) **Values Tab:** *PyFELCI* does not have the ability to deliver error messages when values are entered incorrectly. For this reason, it is important to input values in a format that can be parsed by *Excel*. For instance, "E" notation will work. That is, if the user wishes to efficiently input 1.25x1019, the user could enter 1.25E19. Invalid syntax in data entry will cause the *USEtox Excel* spreadsheet to stop working with no error function passed to the user and the computer will most likely need to be restarted for the app to function properly. For instance, the following values will successfully be interpreted by the app:

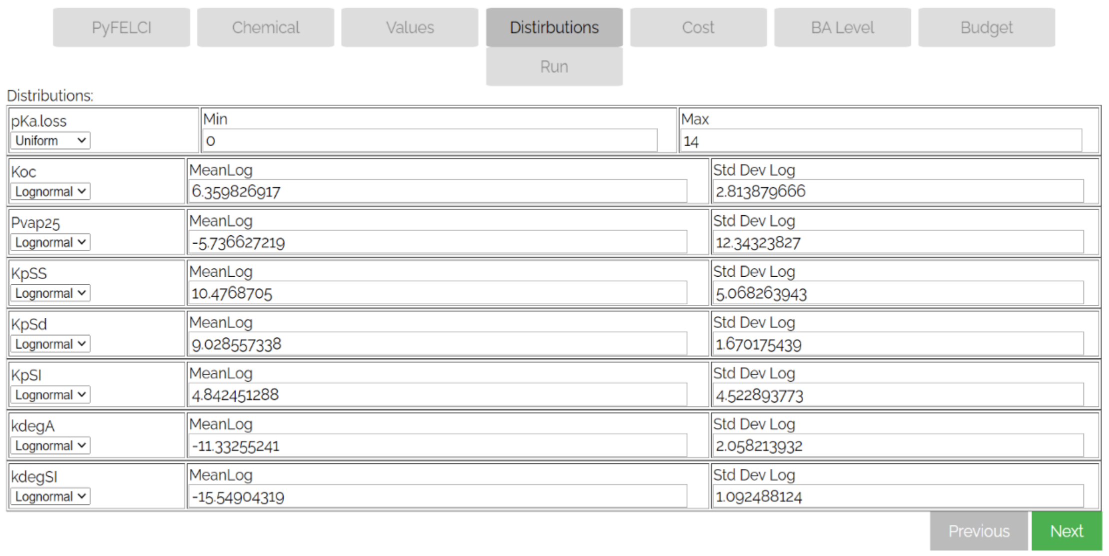
Parameter	Input Value
mw	10000
pKa.gain	7.2
pKa.loss	*No input value*
Kow	10000
Koc	*No input value*
Kh25c	1.00E-07
Pvap25	*No input value*
Sol25	1.00E+05
Kdoc	200
kdegA	*No input value*
kdegW	1.00E-15
kdegSd	1.00E-18
kdegSI	*No input value*
avlogEC50	0.5



Known Parameters:

mw:	10000	g/ml
pKa.gain:	7.2	unitless
pKa.loss:		unitless
Kow:	10000	L / L
Koc:		L / Kg
Kh25c:	1.00E-07	Pa * m^3 / mol
Pvap25:		Pa
Sol25:	1.00E+05	mg / L
Kdoc:	200	L / kg
KpSS:		L / kg
KpSd:		L / kg
KpSl:		L / kg
kdegA:		1 / sec
kdegW:	1.00E-15	1 / sec
kdegSd:	1.00E-18	1 / sec
kdegSl:		1 / sec
avlogEC50:	0.5	log(mg / L)

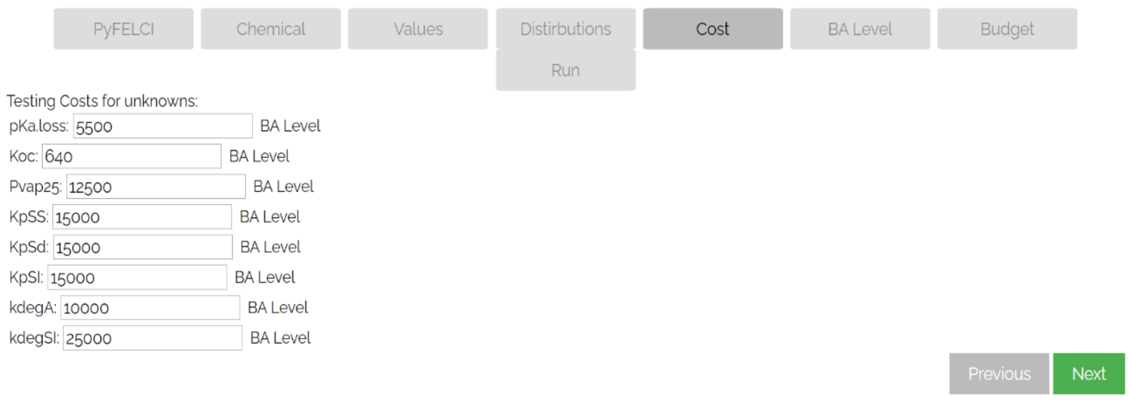
9) **Distributions Tab:** Accept the 5 defaults without making any changes.



Distributions:

pKa.loss Uniform	Min 0	Max 14
Koc Lognormal	MeanLog 6.359826917	Std Dev Log 2.813879666
Pvap25 Lognormal	MeanLog -5.736627219	Std Dev Log 12.34323827
KpSS Lognormal	MeanLog 10.4768705	Std Dev Log 5.068263943
KpSd Lognormal	MeanLog 9.028557338	Std Dev Log 1.670175439
KpSI Lognormal	MeanLog 4.842451288	Std Dev Log 4.522893773
kdegA Lognormal	MeanLog -11.33255241	Std Dev Log 2.058213932
kdegSI Lognormal	MeanLog -15.54904319	Std Dev Log 1.092488124

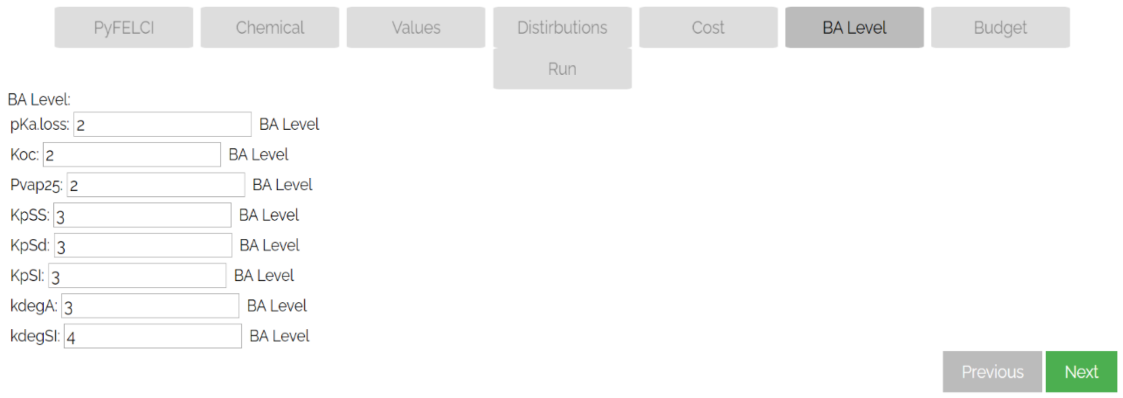
10) **Costs Tab:** Accept the 5 defaults without making any changes.



Testing Costs for unknowns:

pKa.loss:	5500	BA Level
Koc:	640	BA Level
Pvap25:	12500	BA Level
KpSS:	15000	BA Level
KpSd:	15000	BA Level
KpSI:	15000	BA Level
kdegA:	10000	BA Level
kdegSI:	25000	BA Level

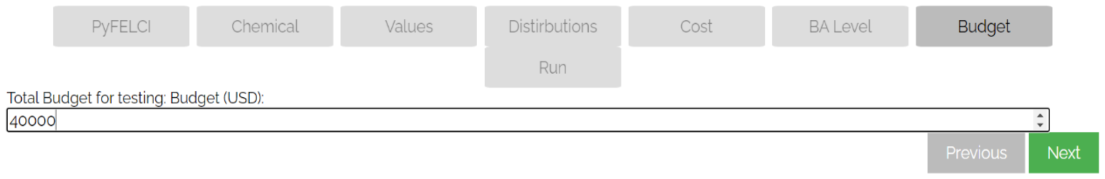
11) **BA Level Tab:** Accept the 5 defaults without making any changes.



BA Level:

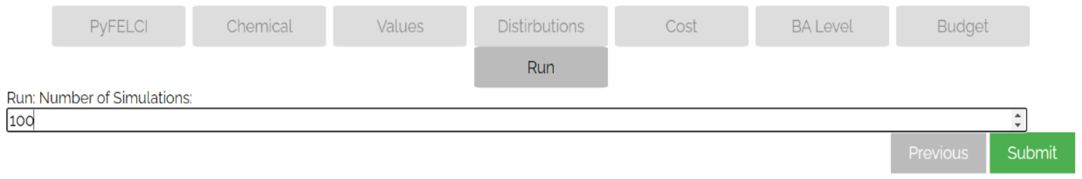
pKa.loss:	2	BA Level
Koc:	2	BA Level
Pvap25:	2	BA Level
KpSS:	3	BA Level
KpSd:	3	BA Level
KpSI:	3	BA Level
kdegA:	3	BA Level
kdegSI:	4	BA Level

12) **Budget Tab:** Input 40000, which ends up being about 75% of the total cost of testing all unknowns.



Total Budget for testing: Budget (USD):
40000

13) **Run Tab:** To finish this example, the user then clicks to the final <Run> tab, leaves it on 100 simulations, and presses "Run VOI Analysis".



Run: Number of Simulations:
100

Saving Results

Completing all simulations may require 20 seconds per simulation or more. For example, 1000 simulations may require 20,000 seconds, or between five and six hours. After completing all the simulations, the results will pop up on your screen and be saved to a folder: PyFELCI->FELCI->Server->app->results.

On computers that do not allow creation of new folders, the plots that appear at the end of the simulations will show up as pop-up windows so that users may save them in the location of choice. These plots would include:

- VoI Portfolio Analysis (*USEtox* Uncertainty vs Additional Tests Performed)
- VoI Portfolio Analysis (*USEtox* Uncertainty vs Cost of Additional Testing)
- Ranking – All Portfolios (Ranking vs Simulations)
- Ranking – Optimal Testing Portfolios (Ranking vs Simulations)