ISYE 6420 – Project Report
Differences in Cycle Life for Fast Charging Methods

Brian Keith
bkeith9@gatech.edu

Abstract—This project was inspired by my interest in batteries as an ever-growing part of the technology we use every day. The goal of this project is to utilize Bayesian techniques to determine if there is a statistical difference in the cycle life[i] between batteries charged under different fast charging methods. The data for this projected was sourced from Severson et. al., 2019[1] consisting of "a comprehensive dataset consisting of 124 commercial lithium iron phosphate/graphite cells cycled under fast-charging conditions, with widely varying cycle lives ranging from 150 to 2,300 cycles".

1 INTRODUCTION

Batteries are an ever-growing part of our everyday lives and are an essential aspect of humanity's transition into a greener future. After all, many of the current solutions rely on more cyclic power generation, e.g., solar and wind, compared to technologies like natural gas. The cyclic nature of the power generation means the energy generated by these sources will need to be stored in batteries to deal with the cyclic power demand from the energy grid. Therefore, the ability for the United States and other countries to move toward more green energy sources is intertwined with battery technology.

One very critical aspect of battery technology development and usage in the field is the number of charging cycles for a battery before it begins to lose capacity[2]. While this is a multivariate problem with many factors that determine the rate at which a battery degrades[3], one aspect that can have a significant impact is the charging method[4]. This is a critical piece of the battery development process, especially when it comes to consumer-facing products (e.g., phones, cars, laptops) which is evident in phone manufacturers such as Xiaomi using it as a key component of their marketing[5]. This paper will explore the effect of fast charging

[i] "…with cycle life (or equivalently, end of life) defined as the number of cycles until 80% of nominal capacity" (Severson et al., 2019)

methods on the cycle life[i] of batteries to explore whether there is a statistical difference in the cycle life based on the charging method utilized.

2 METHODOLOGY

2.1 Data Preprocessing

In this section, I will address the methods utilized to get the final structured data used in my analysis. The data for this project was taken from the article, *Data-driven prediction of battery cycle life before capacity degradation*[1]. The data was provided by the researchers via three sets of `.mat` files[6,7,8] ("batches") which consisted of various data sets gathered by the researchers during their analysis. The methods utilized for each data set can be found in the source data downloads page[6,7,8]. Before I could begin my own analysis, I needed to extract the necessary data from these files and transform them into structured data that could be handled with Python. Luckily, the researchers provided code via GitHub[9] which could be used to transform the `mat` data into a `dict` object.

The code provided by the researchers followed two primary steps. The first step utilized the `h5py` library to iterate through each batch of data, translate them into `dict` objects, and save them to `pickle` files for later use[ii]. After the individual batch data was stored, the second step of the process was to consolidate all three of the batches into a single data structure[iii,iv]. During this process, some data cleaning was performed by the researchers which included removing some batteries with issues during the data collection process and moving some batteries between batches which were continued between batches 1 & 2.

While working though the data consolidation, it became evident that some of the data provided by the researchers was not necessary for the purposes of my analysis including some of the detailed cycle information which made up most of the data's size. What I needed for my analysis was the summary information for each cycle, so I separated out and built a `DataFrame` and saved it separately. The

[ii] The details for this portion of the code can be seen in the *00 Build Pickle Files for Data.ipynb* file provided with my submission.

[iii] The details for this portion of the code can be seen in the *01 Clean and Consolidate Battery Data.ipynb* file provided with my submission.

[iv] Please note that in the event you wish to run the data consolidation script (01), it required ~50 GB of available memory on my machine, mostly related to consolidating the detailed cycle data which was not used in later analysis. I would recommend not trying to run notebooks 00 and 01 and skipping straight to the last file (02) provided with my submission as it only utilizes the summary data which is much smaller.

information in this summary table was what I utilized in the detailed analysis. The final data structure that was used for the remainder of this paper can be seen in **Table 1**.

Table 1 — Starting data structure used in the analysis[v].

	BatteryID	ChargingMethod	CycleLife	QD	CycleNumber
0	b1c0	3.6C(80%)-3.6C	1852	0.000000	1
1	b1c0	3.6C(80%)-3.6C	1852	1.070689	2
...	...	...	...	...	...
100500	b3c45	4.8C(80%)-4.8C	1801	0.880473	1800

2.2 Analysis and Modeling

The remaining code for my project can be found in the *02 Analyze Battery Data.ipynb* file that was provided with my submission[vi]. However, while performing some early analysis, I determined that the best course of action was to focus specifically on batch 3. The primary reason for this was that in batches 1 & 2 the researchers did not utilize the same charging method more than two times[vii]. This meant that I would not have enough data for each charging method to make a meaningful comparison.

The original inspiration to focus on batch 3 came from Zhou et al., 2022[10] which utilized the same dataset. I noted the researchers had a.) only utilized batch 3 and b.) had excluded some batteries from their analysis but did not note the reason why. To investigate, I plotted the discharge capacity as a % of the nominal capacity[viii] as a function of the cycle number which can be seen in **Figure 1**[ix].

[v] Some variable names are changed relative to the source code for consistency with later tables.

[vi] This file contains a significant amount of additional analysis performed while developing this project as well as notes on my thoughts while constructing the analysis. I recommend reviewing the source code for supplementary information and details on the exact methods utilized.

[vii] For batch 1, most methods were utilized two times and for batch 2, no method was utilized more than once.

[viii] Nominal discharge capacity is based on the 1.1 Ah rating of the batteries. So, the discharge capacity as a % of the nominal capacity is $QD/1.1$.

[ix] The battery in the plot is offset by +1 to relative to the second number in the BatteryID from Table 1 to be consistent with the labeling used by the researchers in by Zhou et al., 2022[10]. E.g., b3c45 would be labeled as "46:" in Figure 1.

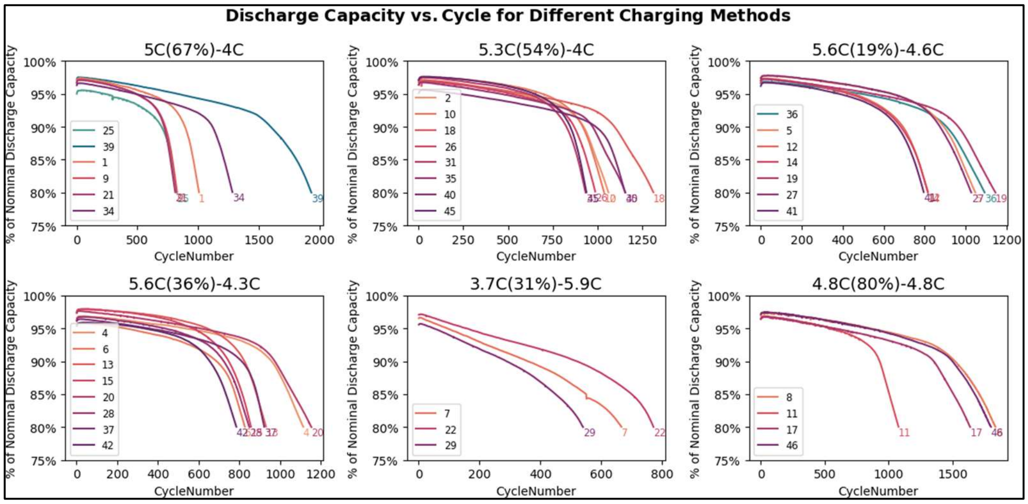

Figure 1— % of Nominal Discharge Capacity vs. Cycle Number for different charging methods in batch 3. The blue colored lines are batteries that were excluded by the researchers in Zhou et al., 2022[10].

Based on the visualization in **Figure 1**, I decided to exclude batteries 39 (batch b3c38, upper left of **Figure 1**) & 11 (b3c10, lower right of **Figure 1**). In addition, charging methods 5.9C(60%)-3.1C and 5.9C(15%)-4.6C (not shown in **Figure 1**) were excluded entirely due to having two and one batteries cycled, respectively.

With the charging methods determined, the next step was to extract the data that would be used for the model. To utilize Bayesian techniques to determine if there is a statistical difference in the cycle life between batteries charged under different fast charging methods, the only data really required was the unique combinations of the charging method and cycle life. This data structure can be seen in **Table 2**. The full data of the 34 batteries used for modeling can be found in **Appendix 1**.

Table 2 — Data structure used for modeling.

	Battery	ChargingMethod	CycleLife	ChargingMethod_Num
0	29	3.6C(80%)-3.6C	541	0
1	22	3.6C(80%)-3.6C	772	0
...	...	...	...	...
33	9	5C(67%)-4C	828	5

With the final data set for modeling in hand, it was time to approach the core issue at hand of determine if there is a statistical difference in the cycle life between batteries charged under different fast charging methods. To approach this problem, I chose to utilize `PyMC` as my probabilistic programming language of choice.

The approach to this problem was at its core very simple, the idea was to utilize `PyMC` to estimate the posterior distribution of the mean cycle lives for each charging method. From there, deterministic variables could be added into the model which would calculate the difference of the means for each charging method. The code for the final implementation of the model can be found in **Appendix 2**.

While the model is very simple, I did perform some experiments during development. One of the most critical aspects of model development in the case of Bayesian models is the prior selection as the prior selection can have a significant impact on the results. For this analysis, I required two different priors, one to estimate the mean and one to estimate the standard deviation. This could then be used to predict the likelihood using a normal distribution. **Figure 2** shows a KDE of the observed input data.

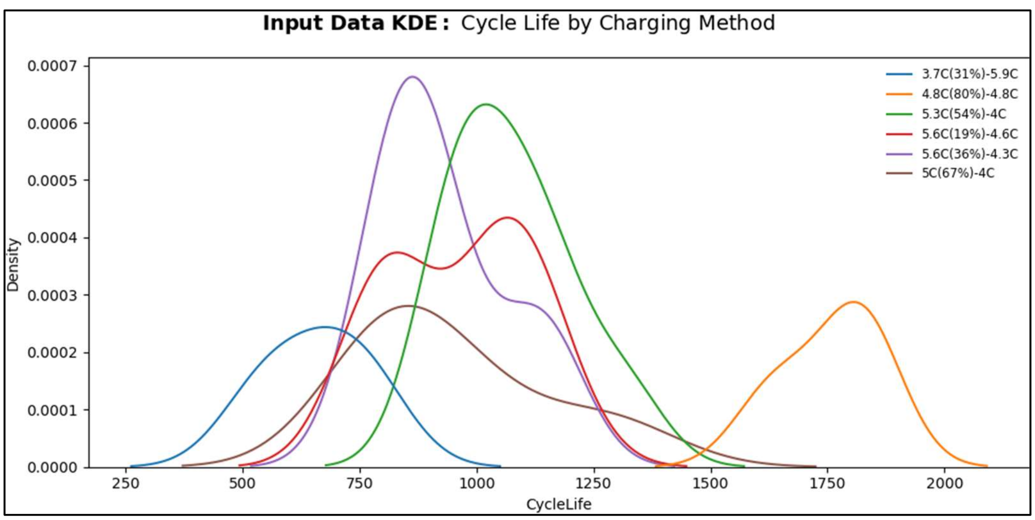

Figure 2— KDE of Cycle Life by Charging method for the input data for the model.

Based on the distributions of the cycle lives across the different charging methods, they are not perfectly normal. However, a normal prior should be a good starting point for the model in `PyMC`. Initially, I was planning to just utilize a normal prior to estimate the standard deviation as well. However, after reading an article on `PyMC` on Bayesian difference in differences[x], where they used a half-normal to estimate the standard deviation for a similar analysis, I also decided to investigate other priors for my standard deviation. In total, I built three different model which utilizing `pm.Normal`, `pm.HalfNormal`, and `pm.HalfCauchy` as the variance estimate. For all three of the models, I utilized the same slightly informed prior for mean of `pm.Normal` with a `mu` of the mean of the cycle life. The reason, `mu` was not left as completely non-informative is that it caused the results of the model to be illogical as the estimates for the mean of all charging methods would be the same, which we can see from **Figure 2** should not be the case.

After building the models, the results in terms of general model performance were similar, but the statistically significant differences for each charging method varied slightly. Based on just the summary statistics of the model, it was not easy to tell which model should be utilized. Therefore, to determine which model to use, I performed a test using `pm.loo` to perform leave-one-out cross-validation[xi]. The results of this analysis can be seen in **Figure 3**[xii].

	rank	elpd_loo	p_loo	elpd_diff	weight	se	dse	warning	scale
HalfCauchy	0	-232.0127	4.7126	0.0	0.6413	4.6693	0.0	False	log
HalfNormal	1	-243.332	18.0994	11.3193	0.3587	11.5009	9.664	True	log
Normal	2	-243.7511	18.5439	11.7383	0.0	11.5923	9.7599	True	log

Figure 3— Comparison of results for leave-one-out cross validation of the three models[xiii].

Based on the results of this test, I decided to utilize the model where the prior variance was estimated using `pm.HalfCauchy` as the final model since it had the best estimate of out-of-sample predictive fit. The coming section will discuss the detailed results for this model.

3 RESULTS

Now the methodology for the model has been discussed and the final model has been determined, the results of the analysis can be discussed. The first important

---

[x] "Difference in Differences." PyMC, www.pymc.io/projects/examples/en/latest/causal_inference/difference_in_differences.html#bayesian-difference-in-differences.

[xi] The possibility of utilizing posterior predictive checks was also considered. However, due to the model structure, the prediction with `pm.sample_posterior_predictive` would be based on all the different methods leading to the information related to the observed data and posterior predictive to be messy since it wasn't considering the different methods separately.

[xii] Additional details for each model can be found in the supporting code files.

[xiii] Model comparison — PyMC 5.10.0 documentation. (n.d.). https://www.pymc.io/projects/docs/en/stable/learn/core_notebooks/model_comparison.html#model-log-likelihood

check when modeling in `PyMC` is to look at the trace plot to ensure that the model doesn't have any abnormal behavior in the chain as well as to get a glimpse of the estimated posterior distribution. This visualization can be seen in **Figure 4**.

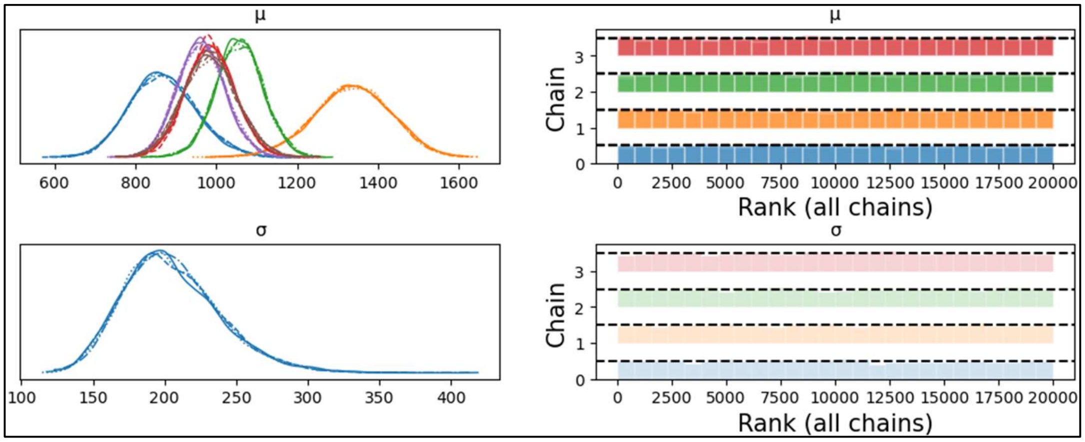

Figure 4— Trace plot of means of each charging policy.

Based on the results of the trace plot, we can see that it appears the model is stable across the different chains with no major cause for concern in any of the rank plots. This provides some assurance that there were no major issues under the hood while generating the posterior distribution.

With that check out of the way, the detailed statistics summarizing the model can be seen in **Figure 5**. The key things here are the HDI sets (none of which include zero; indicating we can be certain the means are significant) and the `r_hat` (all of which are 1.0; indicating the model converged).

	mean	sd	hdi_2.5%	hdi_97.5%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat	HDI_Inc_0
μ[0]	868.66	81.884	708.574	1028.89	0.553	0.393	22049.0	15386.0	1.0	False
μ[1]	1336.368	95.296	1147.717	1520.738	0.691	0.489	19021.0	16784.0	1.0	False
μ[2]	1057.453	58.888	943.009	1173.158	0.368	0.261	25627.0	15633.0	1.0	False
μ[3]	984.971	61.116	864.804	1104.304	0.395	0.28	23923.0	14747.0	1.0	False
μ[4]	962.927	59.441	848.267	1081.256	0.369	0.262	25959.0	14743.0	1.0	False
μ[5]	983.802	67.167	850.679	1113.905	0.424	0.301	25055.0	15292.0	1.0	False
σ	205.192	33.999	144.278	273.63	0.27	0.191	15717.0	14581.0	1.0	False

Figure 5— Summary of the mean estimates of each charging policy modeled by PyMC[xiv].

With the general model properties discussed, we can move on to discussing the core issue at hand of determining whether there are statistical differences in the cycle life based on the charging methods analyzed. To determine this, a

[xiv] Different charging policies are represented by values 0-5 as shown in Appendix 1.

deterministic variable was added to the model which calculated the difference of the means of the modeled data. **Figure 6** shows the results for comparison of the differences in posterior distribution of mean cycle lives for each charging method.

	mean	sd	hdi_2.5%	hdi_97.5%	HDI_Inc_0
Δμ 3.7C(31%)-5.9C vs. 4.8C(80%)-4.8C	-464.595	137.644	-724.747	-189.42	False
Δμ 3.7C(31%)-5.9C vs. 5.3C(54%)-4C	-186.308	100.275	-380.191	13.823	True
Δμ 3.7C(31%)-5.9C vs. 5.6C(19%)-4.6C	-114.636	100.454	-310.413	82.642	True
Δμ 3.7C(31%)-5.9C vs. 5.6C(36%)-4.3C	-92.241	99.045	-287.403	98.735	True
Δμ 3.7C(31%)-5.9C vs. 5C(67%)-4C	-113.19	105.557	-319.106	93.701	True
Δμ 4.8C(80%)-4.8C vs. 5.3C(54%)-4C	278.287	110.485	60.803	489.932	False
Δμ 4.8C(80%)-4.8C vs. 5.6C(19%)-4.6C	349.958	116.965	111.316	572.118	False
Δμ 4.8C(80%)-4.8C vs. 5.6C(36%)-4.3C	372.354	115.678	152.861	603.742	False
Δμ 4.8C(80%)-4.8C vs. 5C(67%)-4C	351.405	120.17	112.156	581.723	False
Δμ 5.3C(54%)-4C vs. 5.6C(19%)-4.6C	71.671	85.122	-91.814	245.436	True
Δμ 5.3C(54%)-4C vs. 5.6C(36%)-4.3C	94.067	83.938	-66.855	261.726	True
Δμ 5.3C(54%)-4C vs. 5C(67%)-4C	73.118	90.275	-106.595	246.727	True
Δμ 5.6C(19%)-4.6C vs. 5.6C(36%)-4.3C	22.395	84.571	-141.695	189.771	True
Δμ 5.6C(19%)-4.6C vs. 5C(67%)-4C	1.446	91.875	-178.81	182.325	True
Δμ 5.6C(36%)-4.3C vs. 5C(67%)-4C	-20.949	90.462	-198.074	156.765	True

Figure 6— Results for comparison of the differences in posterior distribution of mean cycle lives for each charging method[xv].

The key take-away from Figure 6 is that based on the model, the only charging method with a significant difference in the mean cycle life is 4.8C(80%)-4.8C (relative all other charging methods). This is indicated by the fact that the 95% HDI set does not contains zero. Since the 95% HDI set does not contains zero, we can have a high confidence that there is a statistically significant difference in the mean of cycle life using charging method 4.8C(80%)-4.8C. For the other charging methods, since the 95% HDI set includes zero, we cannot be confident that there is a significant different in the mean cycle life relative to any method aside from 4.8C(80%)-4.8C.

We can validate that this result makes sense by expanding on **Figure 2** to include the modeled distribution of cycle life for each charging method. This can be seen in **Figure 7**.

[xv] Differences are calculated relative to the first term. E.g., Δμ 3.7C(31%)-5.9C vs. 4.8C(80%)-4.8C is equivalent to μ[0] - μ[1] from Figure 5.

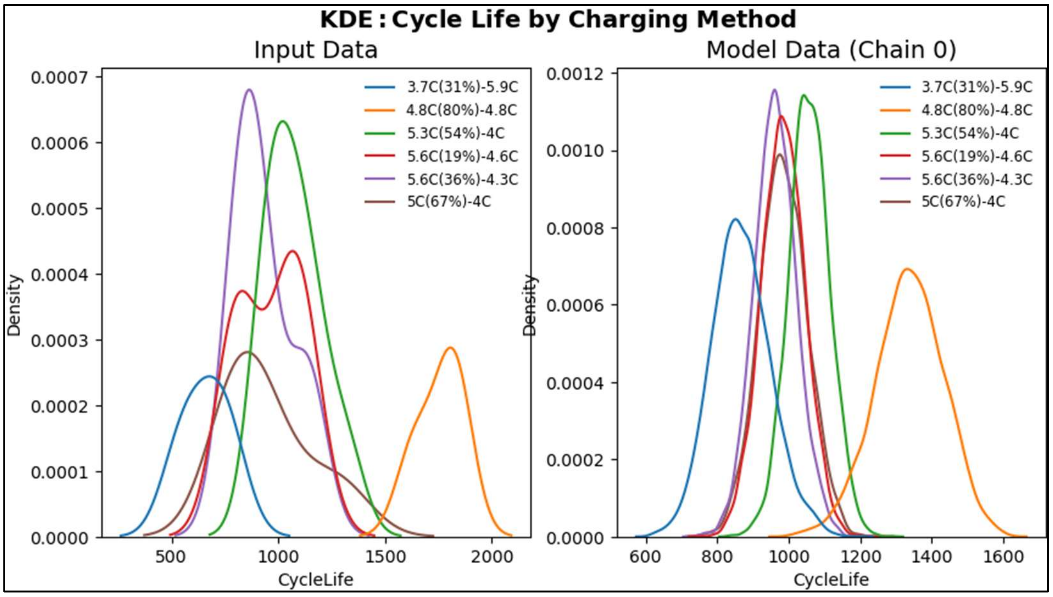

Figure 7— KDE of Cycle Life by Charging method for the input data for the model (left) and the first chain of the model (right).

From **Figure 7**, by looking at the distributions, we can see the distribution 4.8C(80%)-4.8C is shifted towards much higher cycle lives than the other charging methods. While for example, 3.7C(31%)-5.9C and 5.3C(54%)-4C do seem like they could be statistically similar, based on the model, there is not enough of a difference for it to be statistically significant at a 95% level.

4 CONCLUSIONS

The goal of this project was to analyze the distribution of cycle lives for batteries cycled using different charging methods to determine if there is a significant difference in cycle life based on the charging method. By utilizing a dataset of using a dataset of 34 batteries cycled under various fast-charging methods and a model for the distribution of cycle life built using `PyMC`, the only charging method that resulted in significant differences in cycle life was method 4.8C(80%)-4.8C. For this charging method, there was a significant increase in the cycle life for the batteries relative to the other charging methods analyzed.

5 REFERENCES

[1] Severson, Kristen A., et al. "Data-driven prediction of battery cycle life before capacity degradation." *Nature Energy* Volume 4 (2019): 383–391

[2] Han, Xiao, et al. “A Review on the Key Issues of the Lithium Ion Battery Degradation Among the Whole Life Cycle.” eTransportation, vol. 1, Aug. 2019, p. 100005, doi:10.1016/j.etran.2019.100005.

[3] Battery Degradation: Maximizing Battery Life and Performance. www.exro.com/industry-insights/battery-degradation-explained#:~:text=Nevertheless%2C%20battery%20degradation%20sets%20in,charged%2C%20and%20its%20maintenance%20routine.

[4] Mathieu, Romain, et al. “Comparison of the Impact of Fast Charging on the Cycle Life of Three Lithium-ion Cells Under Several Parameters of Charge Protocol and Temperatures.” Applied Energy, vol. 283, Feb. 2021, p. 116344, doi:10.1016/j.apenergy.2020.116344.

[5] Roth, Emma. “Redmi’s Latest 300W Charging Feat Powers Your Phone in Under Five Minutes.” The Verge, 28 Feb. 2023, www.theverge.com/2023/2/28/23618321/redmi-300w-charging-phone-under-five-minutes-xiaomi.

[6] Data for Batch 1 (2017-05-12): `2017-05-12_batchdata_updated_struct_errorcorrect.mat`
https://data.matr.io/1/projects/5c48dd2bc625d700019f3204/batches/5c86c0b5fa2ede00015ddf67

[7] Data for Batch 2 (2017-06-30): `2017-06-30_batchdata_updated_struct_errorcorrect.mat`
https://data.matr.io/1/projects/5c48dd2bc625d700019f3204/batches/5c86bf14fa2ede00015ddd83

[8] Data for Batch 3 (2018-04-12): `2018-04-12_batchdata_updated_struct_errorcorrect.mat`
https://data.matr.io/1/projects/5c48dd2bc625d700019f3204/batches/5c86bd64fa2ede00015ddbb3

[9] “GitHub - Rdbraatz/Data-driven-prediction-of-battery-cycle-life-before-capacity-degradation: Code for Nature Energy Manuscript.” GitHub, https://github.com/rdbraatz/data-driven-prediction-of-battery-cycle-life-before-capacity-degradation/tree/master.

[10] Zhou, Ruomei, et al. “State of Health Estimation for Fast-charging Lithium-ion Battery Based on Incremental Capacity Analysis.” Journal of Energy Storage, vol. 51, July 2022, p. 104560, doi:10.1016/j.est.2022.104560.

Appendix 1: Model Data Table

Battery	ChargingMethod	CycleLife	ChargingMethod_Num
29	3.7C(31%)-5.9C	541	0
22	3.7C(31%)-5.9C	772	0
7	3.7C(31%)-5.9C	667	0
46	4.8C(80%)-4.8C	1801	1
8	4.8C(80%)-4.8C	1836	1
17	4.8C(80%)-4.8C	1638	1
40	5.3C(54%)-4C	1156	2
35	5.3C(54%)-4C	1158	2
31	5.3C(54%)-4C	935	2
45	5.3C(54%)-4C	940	2
26	5.3C(54%)-4C	989	2
2	5.3C(54%)-4C	1063	2
10	5.3C(54%)-4C	1039	2
18	5.3C(54%)-4C	1315	2
41	5.6C(19%)-4.6C	796	3
36	5.6C(19%)-4.6C	1093	3
12	5.6C(19%)-4.6C	817	3
5	5.6C(19%)-4.6C	1048	3
27	5.6C(19%)-4.6C	1028	3
19	5.6C(19%)-4.6C	1146	3
14	5.6C(19%)-4.6C	816	3
13	5.6C(36%)-4.3C	932	4
20	5.6C(36%)-4.3C	1155	4
15	5.6C(36%)-4.3C	858	4
37	5.6C(36%)-4.3C	923	4
4	5.6C(36%)-4.3C	1115	4
6	5.6C(36%)-4.3C	828	4
42	5.6C(36%)-4.3C	786	4
28	5.6C(36%)-4.3C	850	4
1	5C(67%)-4C	1009	5
25	5C(67%)-4C	825	5
21	5C(67%)-4C	813	5
34	5C(67%)-4C	1284	5
9	5C(67%)-4C	828	5

Appendix 2: Final Model Code

```python
rng = np.random.default_rng(903027850)

################################################
#! MODEL
################################################
printmd('## Generating Model of Cycle Life based on Charging Method')
with pm.Model() as model:
    #! Prior estimates of the mean and standard deviation of the cycle life
    #? use of HalfCauchy is inspired by PyMC's bayesian difference in
    #? differences article where a half normal is used for the stdev
    sd = pm.HalfCauchy('σ', beta=10)
    means = pm.Normal(
        'μ', mu=b3_final['CycleLife'].mean(), tau = 0.0001,
        shape=len(b3_final['ChargingMethod'].unique())
        )

    #! Likelihood observed cycle life
    likelihood = pm.Normal(
        'likelihood',
        mu=means[b3_final['ChargingMethod_Num']],
        sigma=sd,
        observed=b3_final['CycleLife']
        )

    #! add deterministic variables to track
    #! difference in means of charging methods
    for i in range(len(b3_final['ChargingMethod'].unique())):
        for j in range(i+1, len(b3_final['ChargingMethod'].unique())):
            pm.Deterministic(f'Δμ_{i}_{j}', means[i] - means[j])

    printmd('### Creating Trace')
    trace = pm.sample(
        5000,
        return_inferencedata=True,
        random_seed=rng,
    )
```