Sampling Interval Analysis

Kyle Nessen

2025-03-21

Table of contents

Background

This report was generated to answer the question of what is an appropriate sampling interval for the VSFB camera study. The goal here is to empirically interrogate if subsampling the photo data is functionally equivalent to the full resolution of images. My motivation for sub-sampling is that labeling these photos is labor-intensive, and I suspect that such a high frequency of 10-minute intervals is introducing unneeded noise. If we can reduce the number of images that we label, we can process these images much sooner. Even switching from 30-minute intervals to 10-minute intervals will reduce the total number of images we have to process by three times. However, we should be careful that we're not throwing away necessary information to answer the question at hand, which this report is intended to shed some light on.

Data Preparation

I won't go into the full detail of how I prepared and cleaned the data, but instead I'll just give a quick summary to provide additional context for the results that will follow. With that said, I have all the code and the full analysis script that I'm happy to share if someone reading this would like to see the specifics.

Monarch Count Data Preparation

- Loaded SC1 and SC2 datasets with 5-minute intervals; reduced to 10-minute intervals by keeping every other row.
    - These two deployments are the only in the full dataset with this interval, which is why I chose to reduce to 10 minutes.
- Parsed timestamps and sorted data by deployment and time.
- Appended SC4 data and re-sorted the full 10 min dataset.

Nighttime Data Removal

- Defined specific night periods for each deployment where labelers copy and pasted last valid count value
- Removed observations during these periods to avoid duplicated or unreliable counts.
- Generated an "observation period" unique identifier, which is the combination of deployment and date. This gives a way to group by complete observation periods.

Subsampling

- Created datasets at 10, 20, 30, 60, 90, and 120-minute intervals by retaining every nth row within each deployment group.

Monarch Change Calculation

- Calculated changes in butterfly counts between time points.
- Replaced zero changes with 0.1 and applied a signed log transformation to prepare for modeling.
    - This was done based on initial modeling and poor performance. Log transforming helped.

Wind Data Integration

- Loaded wind data and aligned it with butterfly observation intervals based on deployment and timestamps.
- Calculated average wind speed, average and max gusts, and number of wind observations per interval.

Modeling

Based on my conversation with Francis, we decided to try an information theoretic approach: essentially build the sub-sampled datasets, run the same stastics model on each, and compare AIC scores. Below is some information on how I did that.

Model Preparation

- Defined `monarch_change` as the response variable (log transformed).
- Selected wind metrics (`avg_wind_speed_mph`, `avg_wind_gust_mph`, `max_wind_gust_mph`) as fixed effects.
- Included `deployment_id` as a random effect to account for variation between deployments.
- Scaled all wind predictors for comparability.
- Converted timestamps to numeric values to model temporal autocorrelation.

Model Fitting

- Used `lme()` from the `nlme` package to fit a linear mixed-effects model.
- Included a first-order autoregressive structure (`corAR1`) to account for temporal correlation within deployments.
- Applied the model to each subsampled dataset (10 to 120 minutes).
- Captured model convergence status, AIC, fixed and random effects, and autocorrelation parameter.

AIC Comparison

Below are the AIC scores for each sub-sample dataset along with a chart. It's clear from these results that the longest interval is the best performing model. However, I'm cautious about these interpretations because I think we are misusing AIC as a tool of comparison since we are effectively changing the dataset for each model. I believe the AIC score is very sensitive to the number of data points, which may explain why we see such a predictable decrease as we increase the interval between samples. I take an alternative approach below.

Table 1: Model comparison across different sampling intervals

Interval (minutes)	AIC Score	Number of Observations	Model Converged
10	1343.45	298	TRUE
20	682.85	149	TRUE
30	468.60	99	TRUE
60	234.91	50	TRUE
90	156.23	33	TRUE
120	116.14	25	TRUE

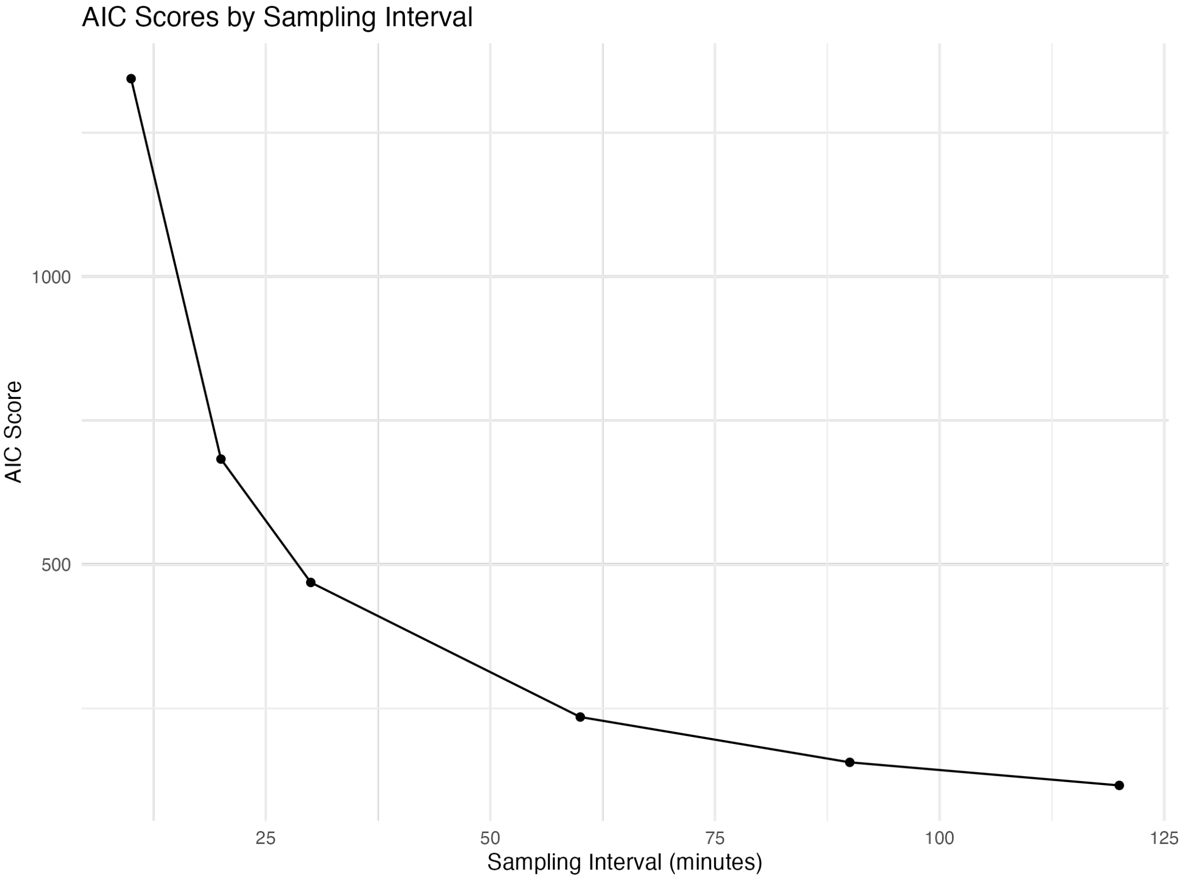

Figure 1: AIC scores for all sub-sampled datasets.

RMSE Comparison

After searching around for another approach to this question, I came across the idea of comparing root mean square error. This has the advantage of giving a direct estimate of how much information is lost at each subsampling level. The graph below shows the relative loss of information compared to the baseline for each interval. Happily, it appears that 30 minutes results in the lowest change, less than 5% of the original information lost. The RMSE increases rapidly after that. So from my view, 30 minutes seems like the sweet spot, and certainly defensible. Looking forward to your thoughts.

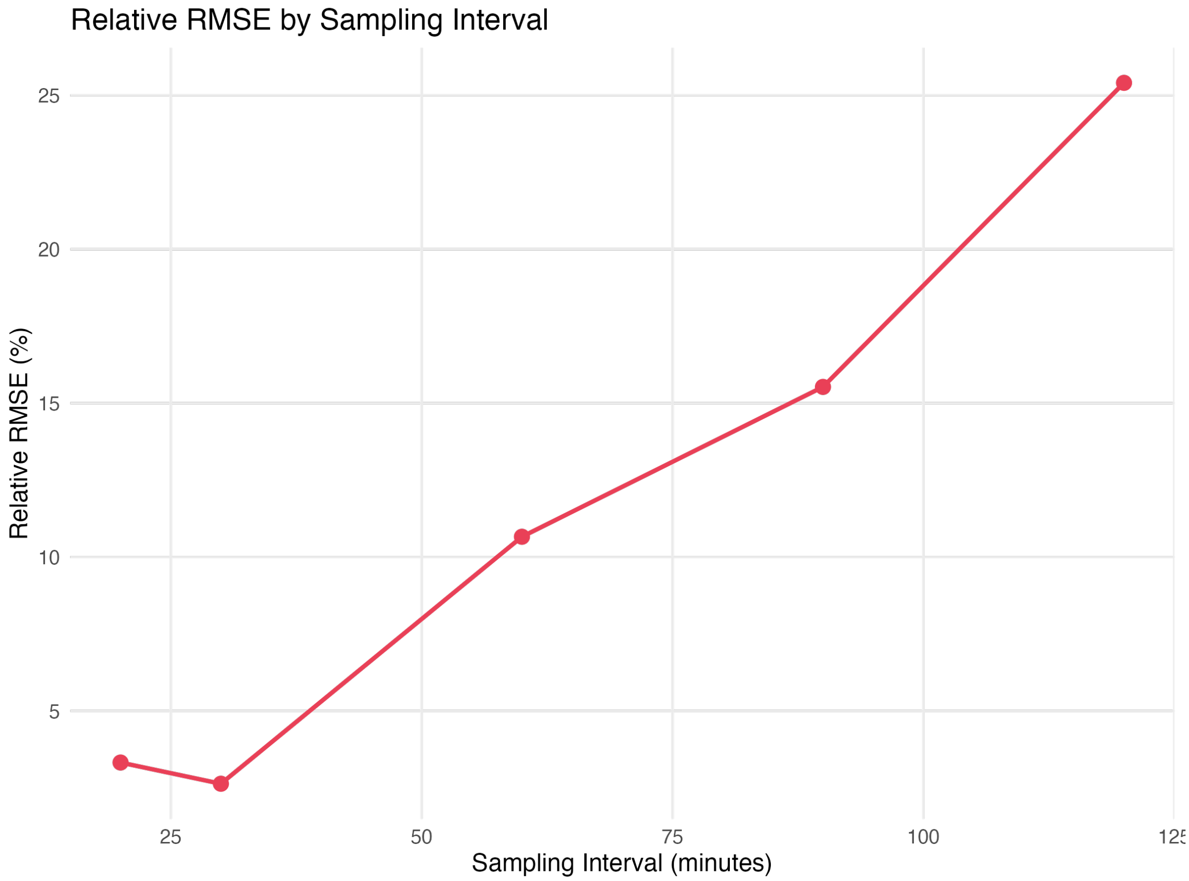

Figure 2: Relative root mean square error (RMSE) for each subsampling interval compared to the full dataset.

30 Min Model Results

Out of curiosity, I wanted to see how some of these models were performing and if we can glean any insight from them. Based on the results of above, and from my initial exploratory analysis, the 30-minute model seemed to have the best performance, but by no means was it perfect.

Below is the performance model output showing that there's some issues, but we're getting close to a defensible model. And below that is the raw output of the results from that model.

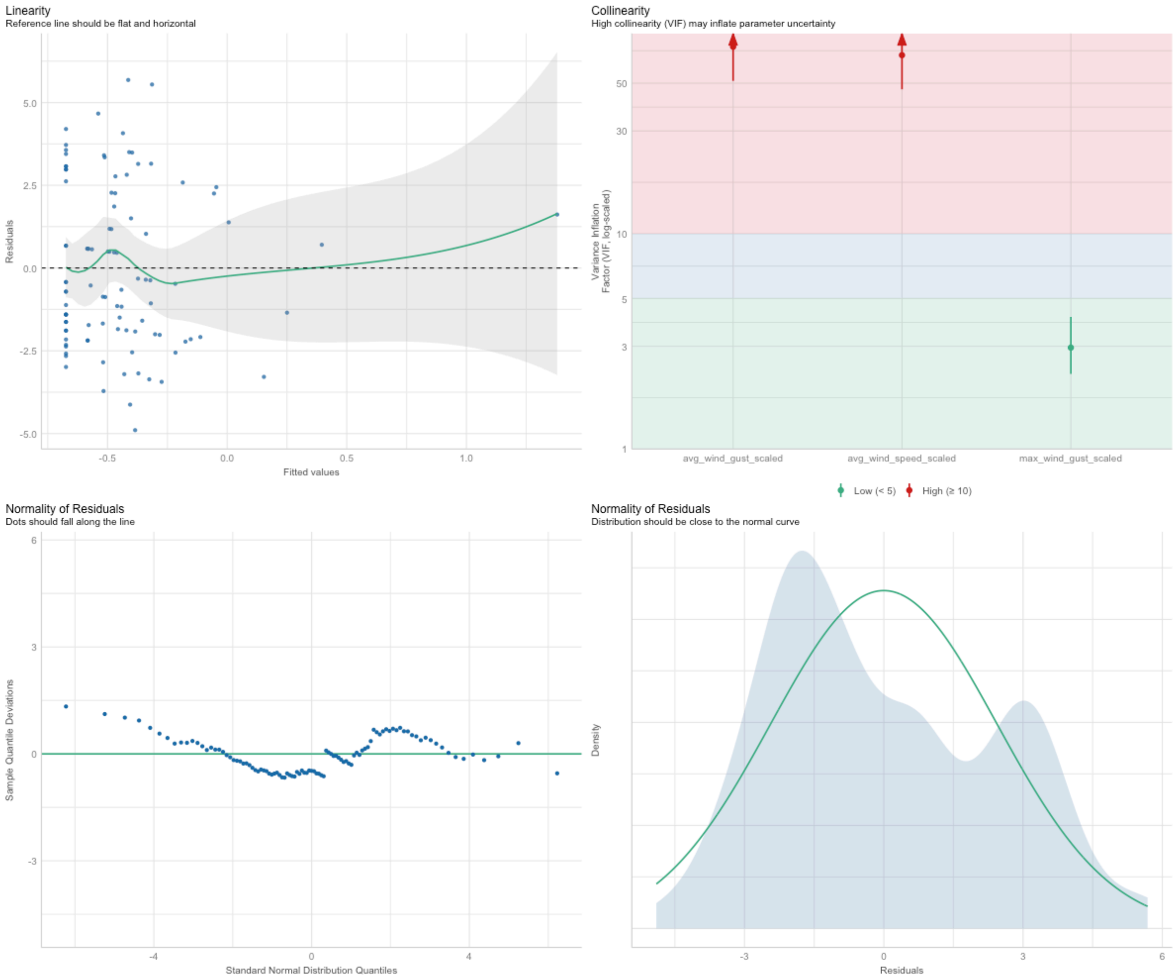

Figure 3: Model Performance Diagnostics

```
Linear mixed-effects model fit by REML
  Data: model_data
       AIC      BIC    logLik
  468.6036 486.4807 -227.3018

Random effects:
 Formula: ~1 | deployment_id
```

```
         (Intercept) Residual
StdDev: 9.468284e-05 2.471424

Correlation Structure: ARMA(1,0)
 Formula: ~time_numeric | deployment_id
 Parameter estimate(s):
Phi1
   0
Fixed effects:  monarch_change ~ avg_wind_speed_scaled + avg_wind_gust_scaled +      max_win
                           Value Std.Error DF    t-value p-value
(Intercept)           -0.4428602 0.2483875 93 -1.7829406  0.0779
avg_wind_speed_scaled -0.5588815 2.0504707 93 -0.2725625  0.7858
avg_wind_gust_scaled   0.3342085 2.1447222 93  0.1558283  0.8765
max_wind_gust_scaled   0.4028745 0.4293107 93  0.9384216  0.3505
 Correlation:
                      (Intr) avg_wnd_s_ avg_wnd_g_
avg_wind_speed_scaled  0.000
avg_wind_gust_scaled   0.000 -0.981
max_wind_gust_scaled   0.000  0.198     -0.349

Standardized Within-Group Residuals:
       Min         Q1        Med         Q3        Max
-1.9820287 -0.7706652 -0.1926982  0.9128569  2.2988928

Number of Observations: 99
Number of Groups: 3
```

My key takeaway here is that wind does not appear to predict changes in monarch abundance from moment to moment (in this case, every 30 minutes). The same was true for all other models, including the 120 minute, where the p-values were unambigously large. Still, this is only a fraction of the dataset and the model performance still has many issues. These preliminary results agree, however, with what I have been seeing in the video, where clusters don't obviously respond to wind patterns. Exposure to direct sunlight is another story, which I can tackle at another time.

Appendix

I'm including here a lot of time series plots to give you a better sense of the raw data. Each observation period (e.g. SC1 on November 19th) has it's own figure, with all subsampling intervals plotted. Mostly as reference.

SC1 Deployment

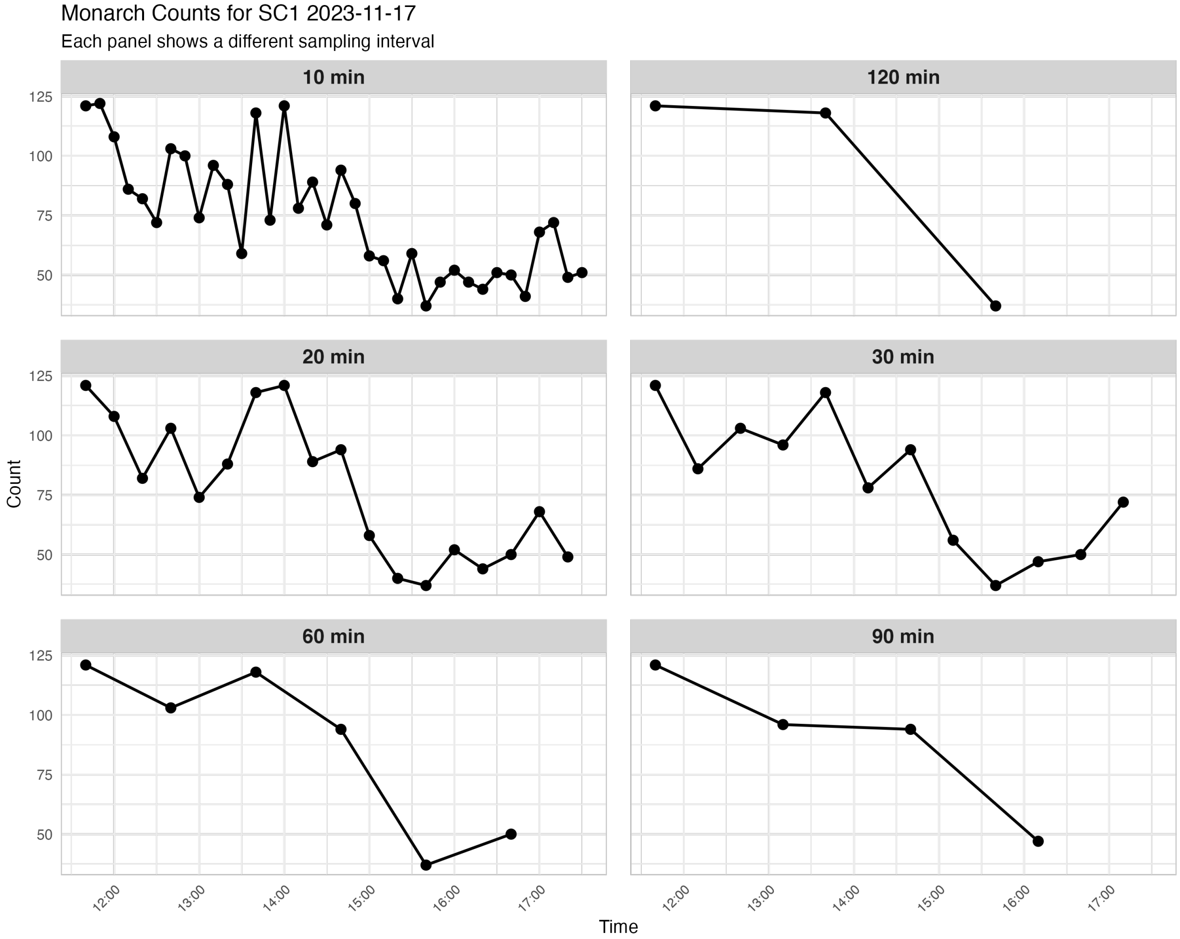

Figure 4: SC1 November 17th, 2023

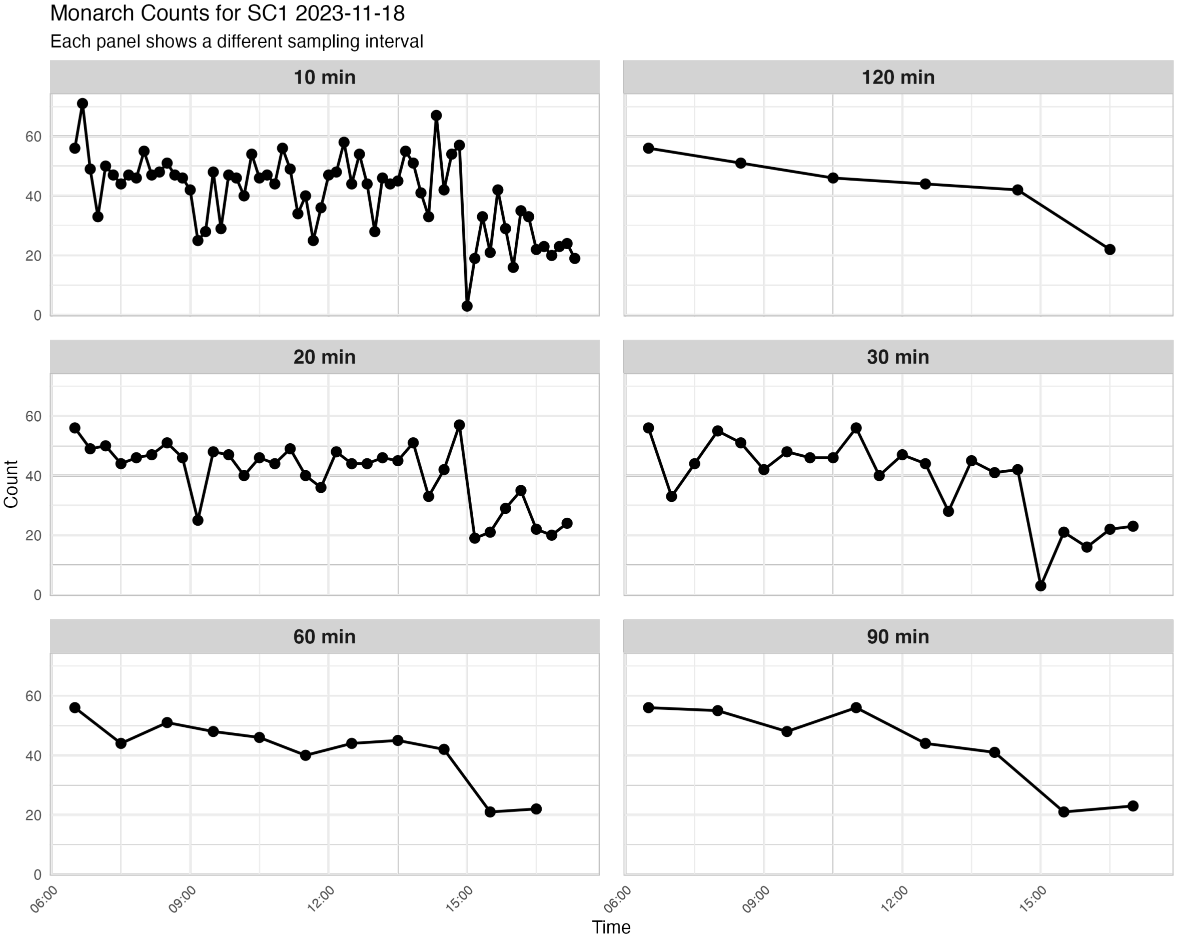

Figure 5: SC1 November 18th, 2023

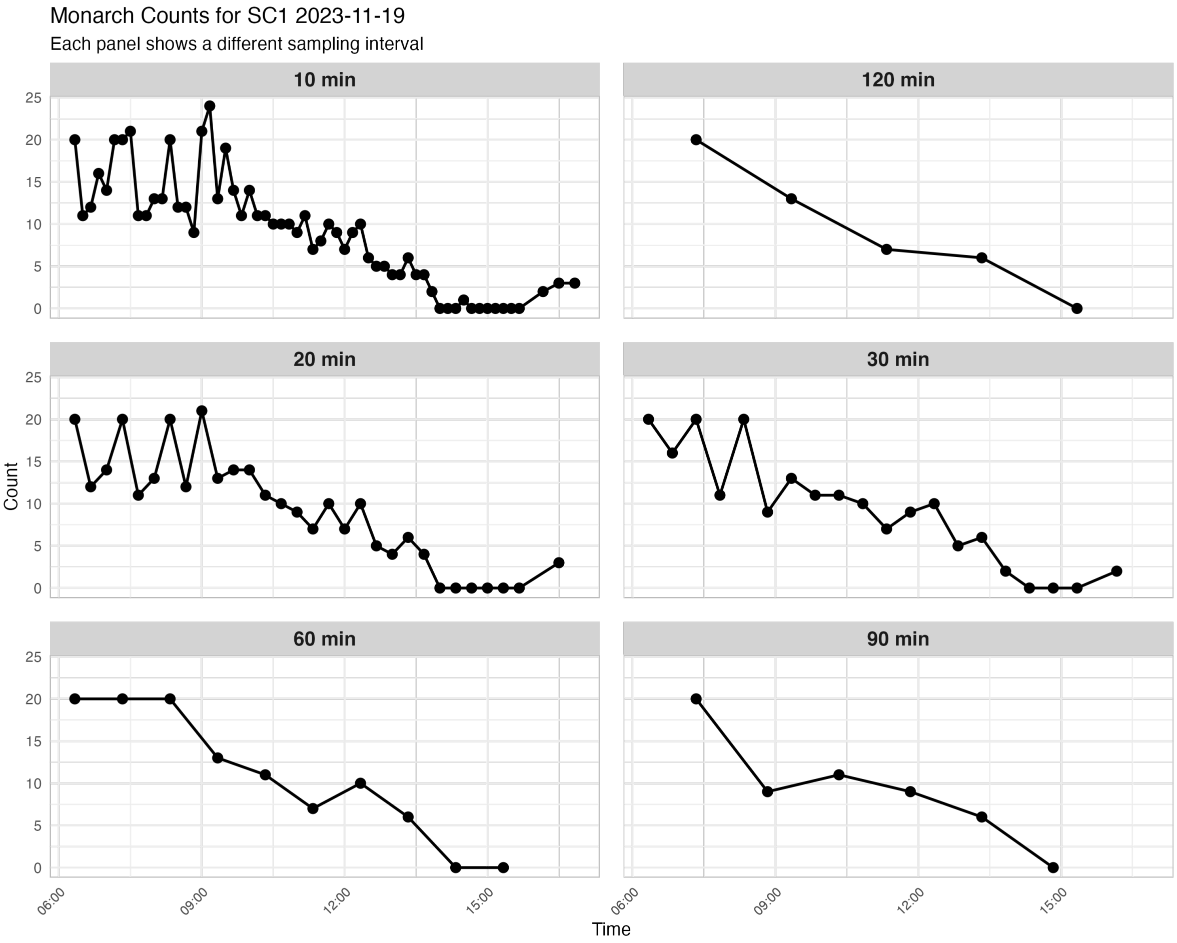

Figure 6: SC1 November 19th, 2023

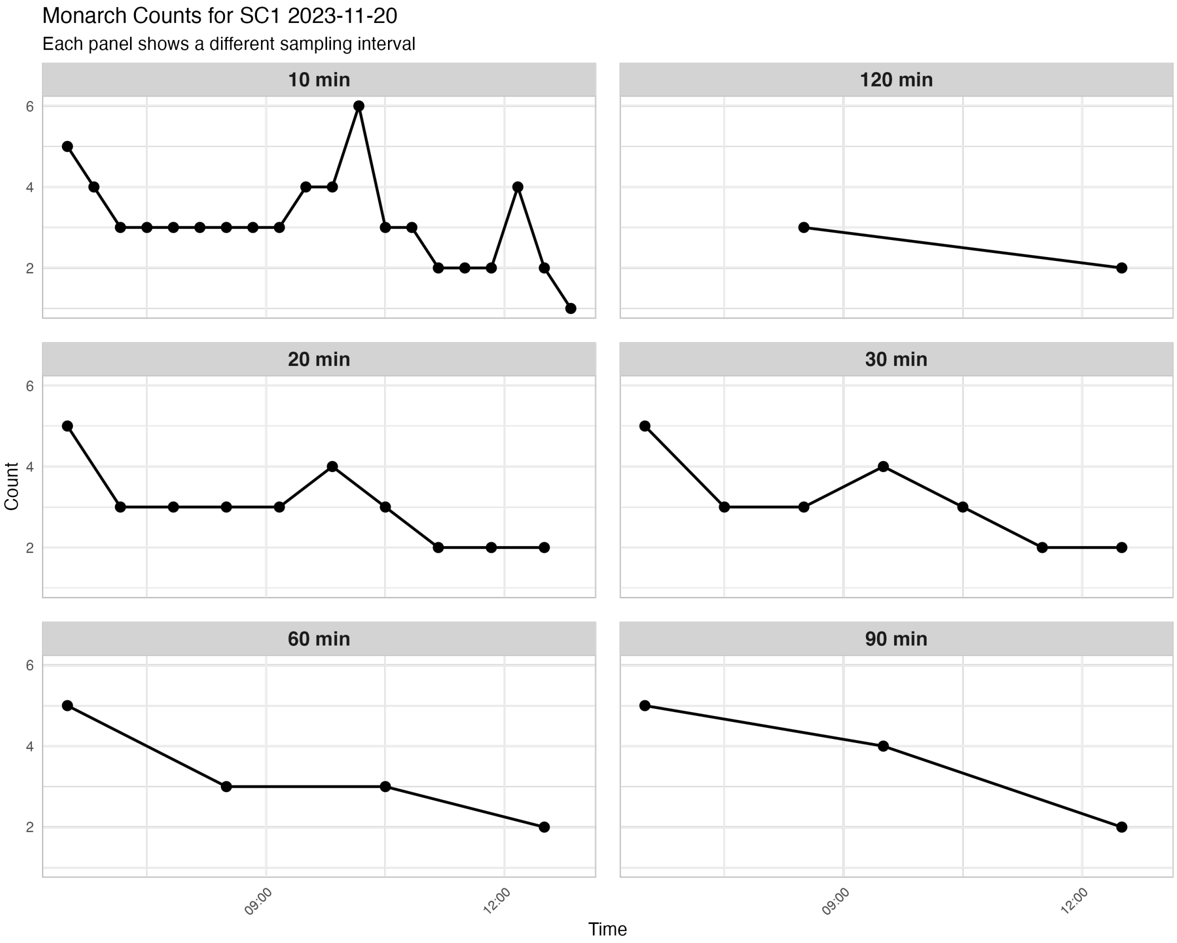

Figure 7: SC1 November 20th, 2023

SC2 Deployment

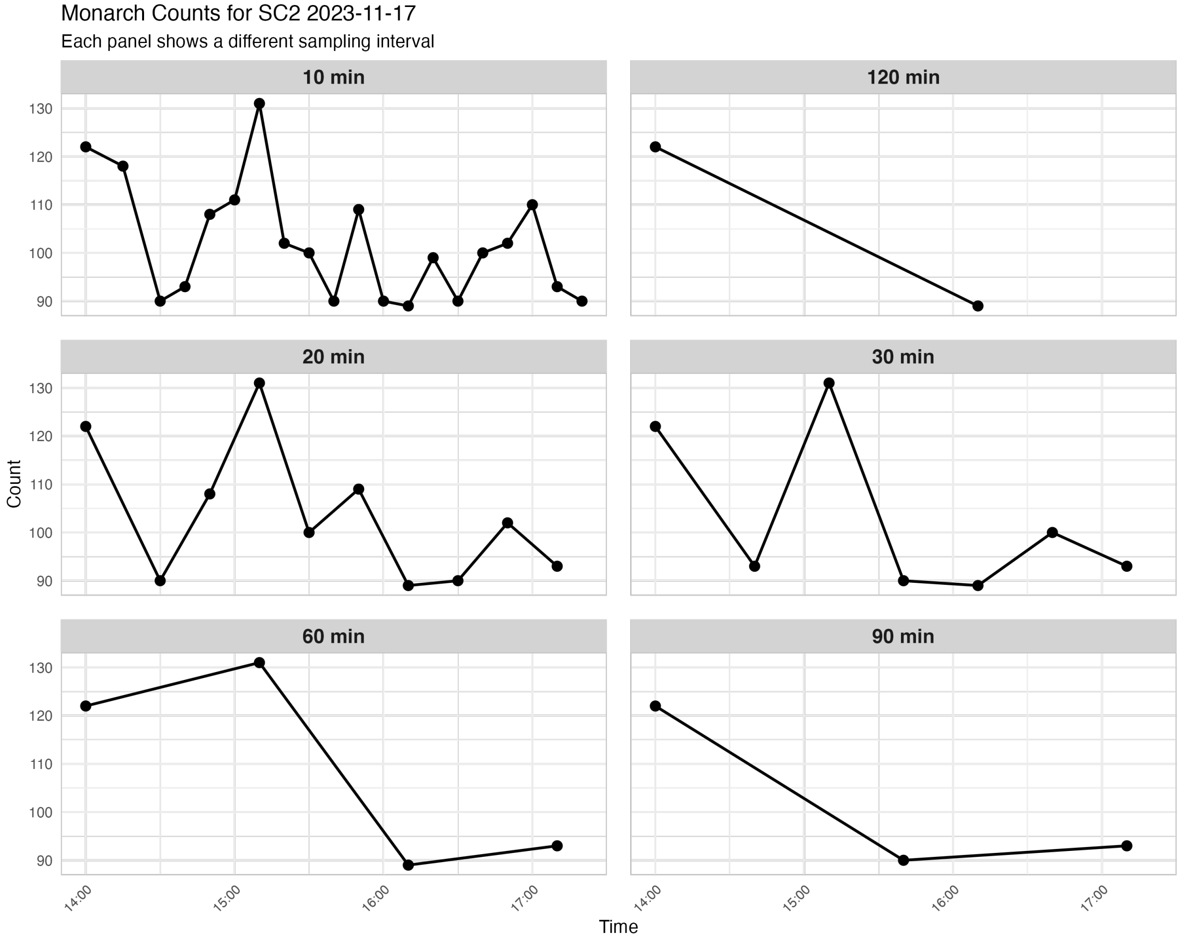

Figure 8: SC2 November 17th, 2023

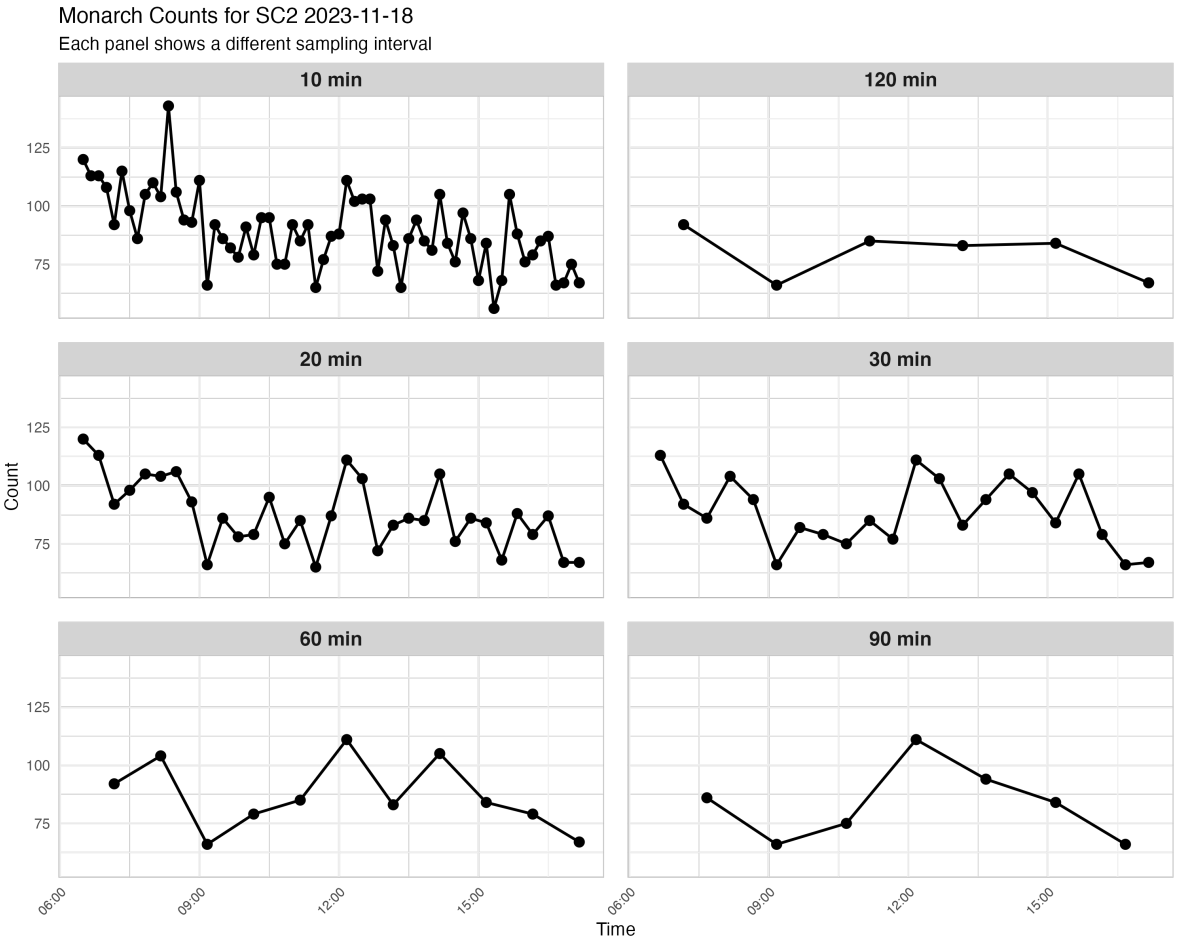

Figure 9: SC2 November 18th, 2023

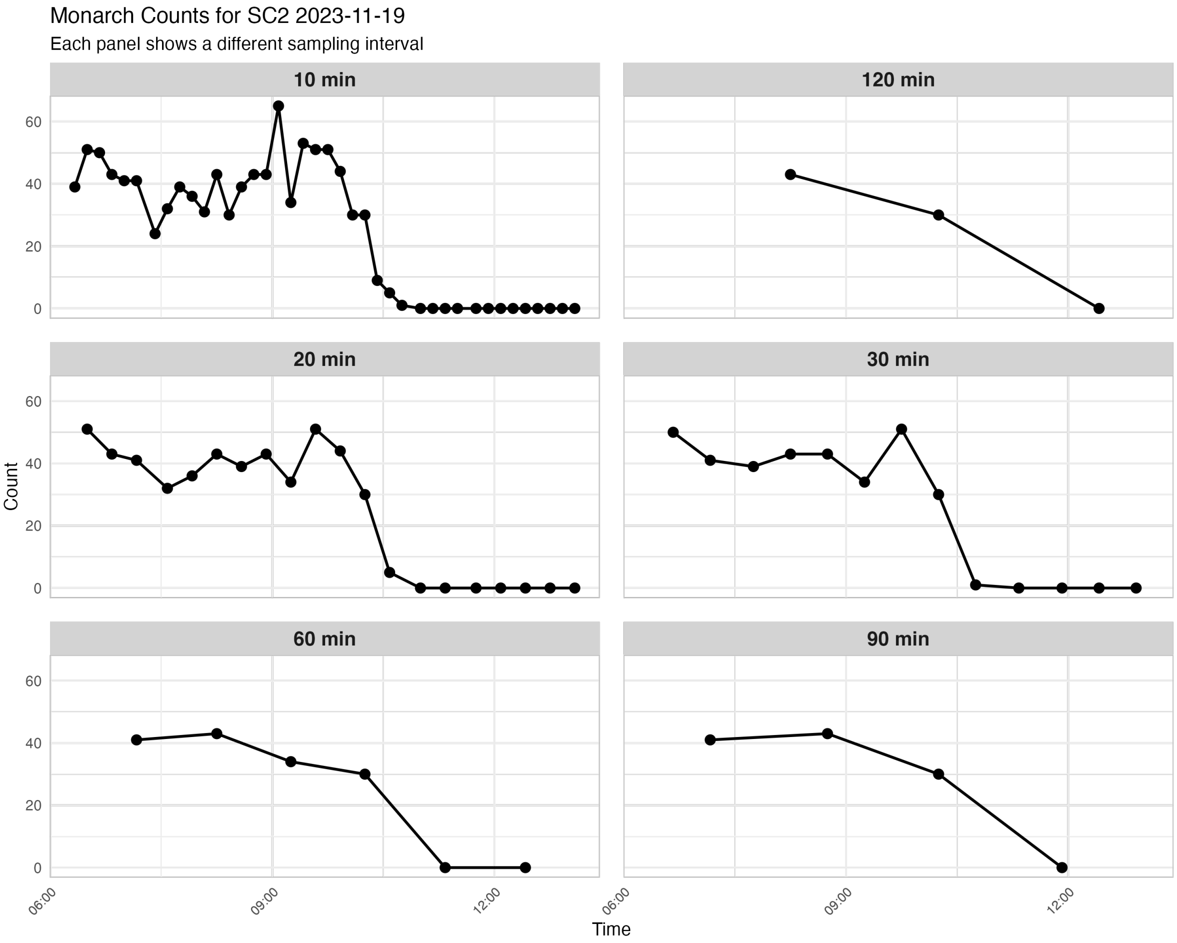

Figure 10: SC2 November 19th, 2023

SC4 Deployment

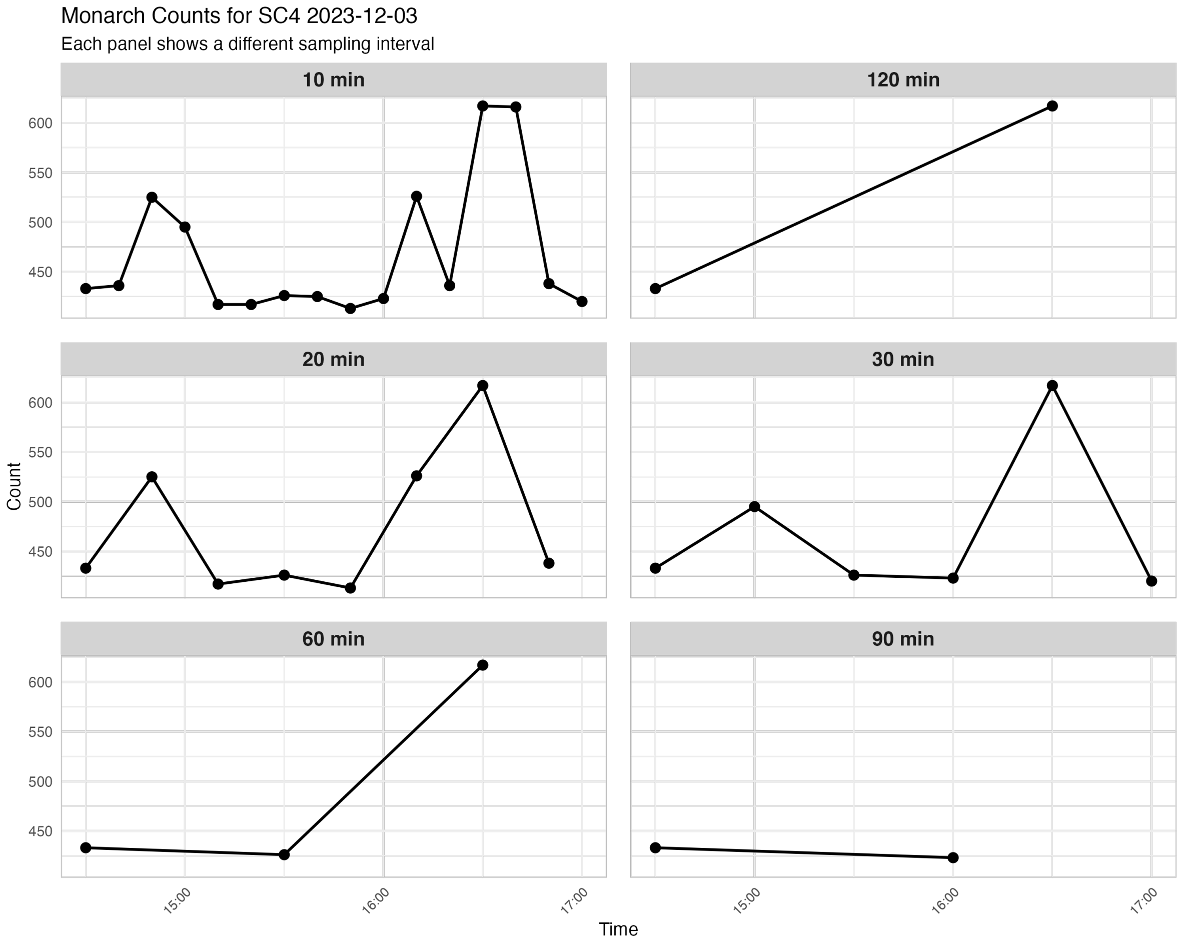

Figure 11: SC4 December 3rd, 2023

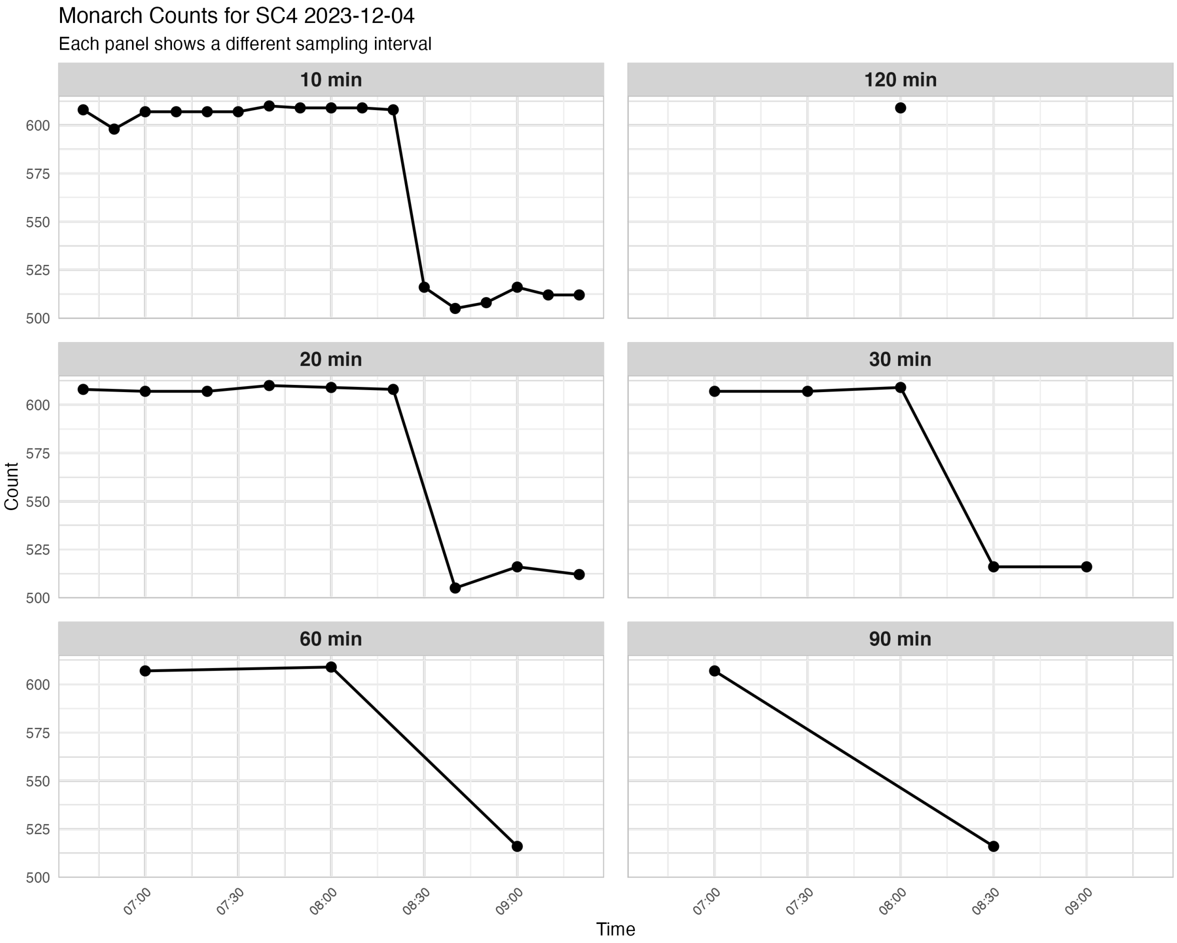

Figure 12: SC4 December 4th, 2023