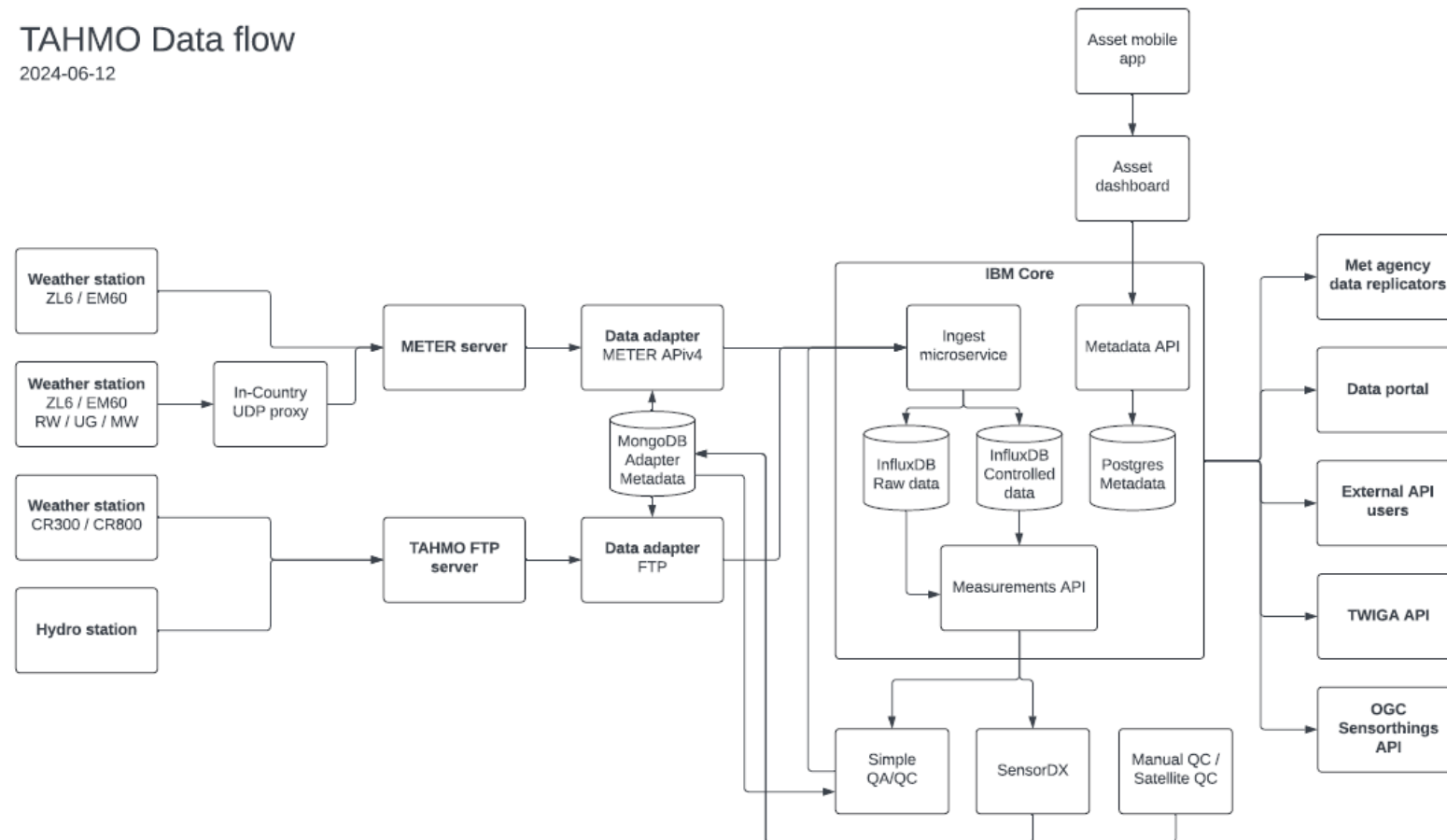


Understanding TAHMO Data Flow for QC

Tom Dietterich

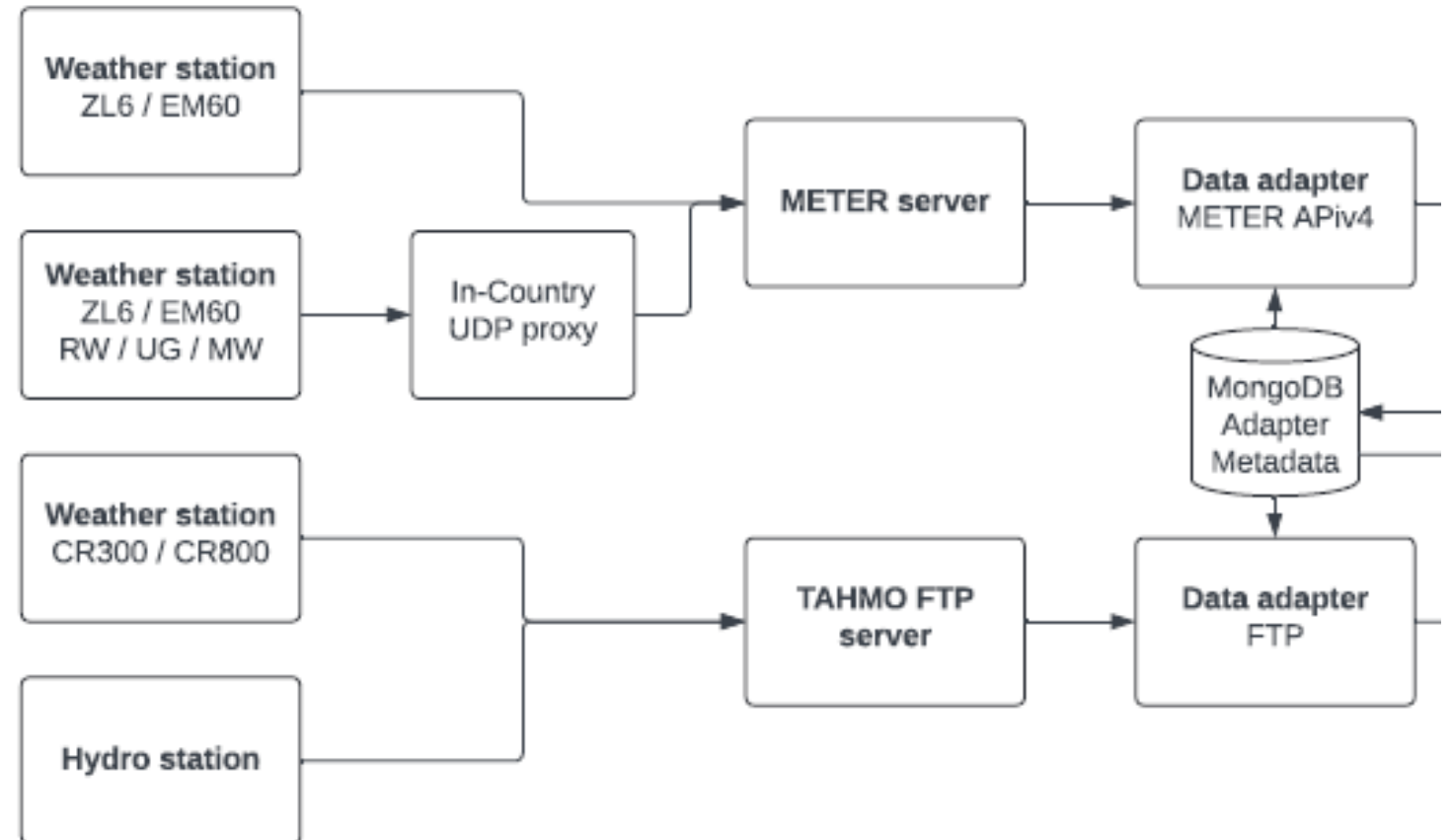
Goal: Understand where QC happens

TAHMO Data flow
2024-06-12



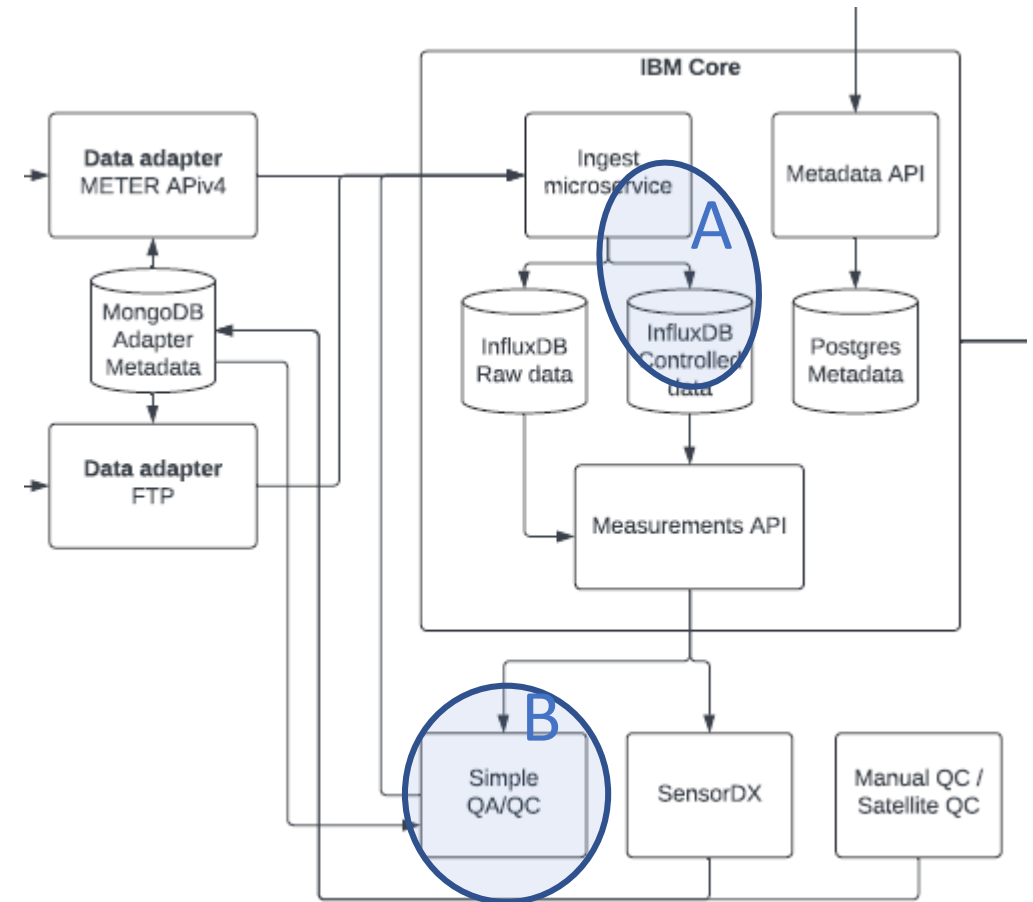
Data Acquisition

- Questions for Rick:
 - What are these different weather station types?
 - What is in the MongoDB?
- Note:
 - Output from the two Data adapters goes to the Ingest Microservice



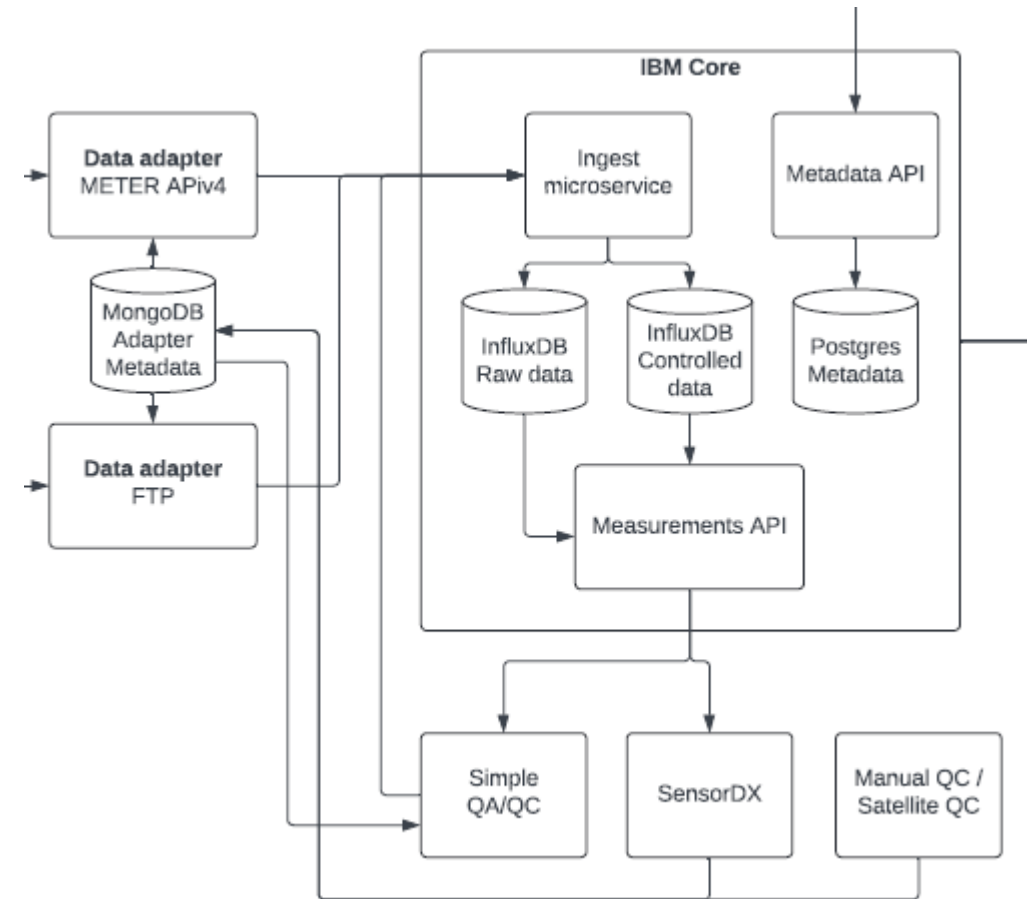
Limit Checks and Related Rules

- Where: A vs B?
- Range checks
 - Temperature
 - Relative Humidity
 - Atmospheric Pressure
 - Precipitation
 - Solar Radiation (?)
- Other rules?
 - Minimum variance (to catch stuck sensors)?



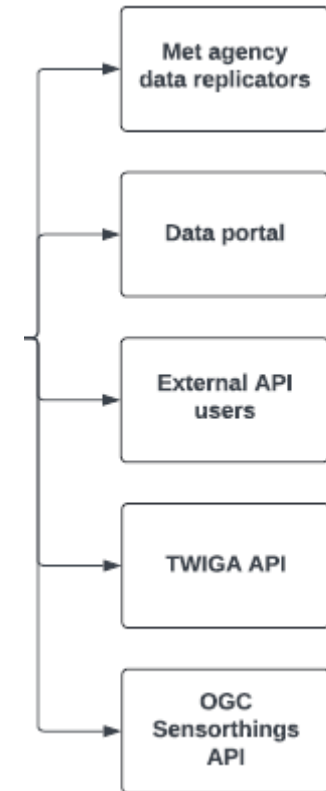
Order of Processing: Is this correct?

- Raw data → Ingest → InfluxDB Raw data
- Simple QA/QC requests raw data, applies range checks and other rules → Ingest → InfluxDB Controlled data
- SensorDX requests Controlled data and applies neighbor regression → writes output to MongoDB?
- Manual/Satellite. Rick prepares spreadsheet, Gilbert & Victor manually compare and create QC Objects in MongoDB



Customer Data Access

- How do the QC flags in the MongoDB become visible to the customers?
- What API do they access?

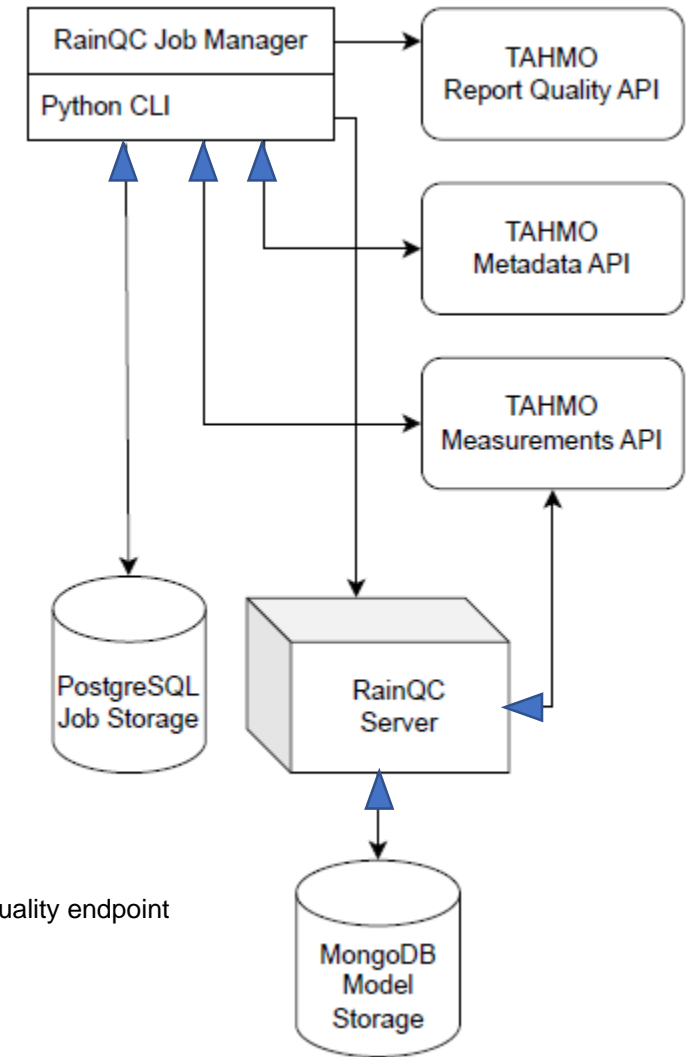


SensorDX Quality Control

[Not to be confused with the ticketing system]

- JobManager queries ?? to find the list of stations having models (“target stations”)
- JobManager creates a new job for each target station for the current day
- JobManager queries PostgreSQL to find list of incomplete jobs from previous run and merge them into the list of jobs
- JobManager queries Measurements API to determine which jobs are “data complete” and can be run
 - Target station and its neighbors must all have sufficient data for the specified date
- JobManager invokes RainQC server on all runnable jobs
- RainQC server retrieves data for the target station and its neighbors from the Measurements API
- RainQC server computes the data quality score (1 or 2) and writes it to the Measurements API ?? [missing on Rick’s diagram] The job manager writes the flags to the /report quality endpoint
- RainQC server returns a result code to the Job Manager
- JobManager updates the PostgreSQL Job table to indicate which jobs succeeded and which failed. The failed jobs will be re-tried the next day

Diagram has been updated. See related email.



Explanation of the Daily JobManager report

- Indicates which date is being scored
- Note: the Job Manager is stateful, rerunning it will create new jobs. There is a command line flag to prevent this

```
Current UTC date: 2024-07-02 -> scoring models for previous day: 2024-07-01
```

```
-----  
Daily Model Data Completeness Check:
```

```
data completeness 50% | complete models: 115 of 273 (42.12%)  
data completeness 60% | complete models: 114 of 273 (41.76%)  
data completeness 70% | complete models: 109 of 273 (39.93%)  
data completeness 75% | complete models: 108 of 273 (39.56%)  
data completeness 80% | complete models: 100 of 273 (36.63%)  
data completeness 85% | complete models: 96 of 273 (35.16%)  
data completeness 90% | complete models: 96 of 273 (35.16%)  
data completeness 95% | complete models: 96 of 273 (35.16%)  
data completeness 100% | complete models: 93 of 273 (34.07%)
```

```
-----  
station status | total: 313, delayed: 144, offline 24h: 82, offline week: 73  
| battery, min: 0, max: 100, mean: 58.53, std dev: 28.79  
| battery, common values: [(100, 158), (0, 83), (74, 5), (60, 4), (69, 4)]  
| battery <= mean, common countries: [('KE', 31), ('GH', 21), ('UG', 9), ('TG', 7), ('ML', 7)]
```

This is correct. The --process command line flag is used when the day's jobs have already been added to the job table.

Explanation of the Daily JobManager report

- What fraction of models (target stations) are data complete?
- I watch the 100% completeness number as an overall indication of network health
- Question: What is the definition of data completeness?

```
Current UTC date: 2024-07-02 -> scoring models for previous day: 20
```

```
-----  
Daily Model Data Completeness Check:
```

data completeness	50%		complete models:	115 of 273	(42.12%)
data completeness	60%		complete models:	114 of 273	(41.76%)
data completeness	70%		complete models:	109 of 273	(39.93%)
data completeness	75%		complete models:	108 of 273	(39.56%)
data completeness	80%		complete models:	100 of 273	(36.63%)
data completeness	85%		complete models:	96 of 273	(35.16%)
data completeness	90%		complete models:	96 of 273	(35.16%)
data completeness	95%		complete models:	96 of 273	(35.16%)
data completeness	100%		complete models:	93 of 273	(34.07%)

```
-----  
station status | total: 313, delayed: 144, offline 24h: 82, offline  
| battery, min: 0, max: 100, mean: 58.53, std dev: 28.79  
| battery, common values: [(100, 158), (0, 83), (74, 5), (60, 4),  
| battery <= mean, common countries: [('KE', 31), ('GH', 21), ('UG
```

Data completeness is measured at different percentage levels. For a model to be 100% data complete, this means that the target station for the model and its neighbors all have 100% of the expected measurement data present. If we assume for the sake of discussion that every station reads in 5 minute intervals, this means the target and its neighbors all have 288 non-null/non-empty readings. Similarly, for a data completeness level of 80%, the target and all its neighbors would need to have at least 230 readings (80% of 288). Models that are 100% data complete of course are also data complete for lower percentages of data completeness.

Explanation of the Daily JobManager report

- General status information (not required by JobManager, but it was easy to show)
- Question: What is “total”? Are these all of the target stations and neighbors that we use?

```
Current UTC date: 2024-07-02 -> scoring models for previous day: 2024-07-01
```

```
-----  
Daily Model Data Completeness Check:
```

```
data completeness 50% | complete models: 115 of 273 (42.12%)  
data completeness 60% | complete models: 114 of 273 (41.76%)  
data completeness 70% | complete models: 109 of 273 (39.93%)  
data completeness 75% | complete models: 108 of 273 (39.56%)  
data completeness 80% | complete models: 100 of 273 (36.63%)  
data completeness 85% | complete models: 96 of 273 (35.16%)  
data completeness 90% | complete models: 96 of 273 (35.16%)  
data completeness 95% | complete models: 96 of 273 (35.16%)  
data completeness 100% | complete models: 93 of 273 (34.07%)
```

```
station status | total: 313, delayed: 144, offline 24h: 82, offline week: 73  
| battery, min: 0, max: 100, mean: 58.53, std dev: 28.79  
| battery, common values: [(100, 158), (0, 83), (74, 5), (60, 4), (69, 4)]  
| battery <= mean, common countries: [('KE', 31), ('GH', 21), ('UG', 9), ('TG', 7), ('ML', 7)]
```

Yes, the total here refers to all the stations that are a target and/or a neighbor for the current set of models. I had originally planned to write out a simple 'text' histogram for the battery health, but had some formatting issues, so this was never completed. As a somewhat poor substitute for a histogram, we instead see a few common values.

Number of targets affected by low-data stations

```
108 LOW DATA (< 0.9) and 89 NO DATA weather stations impacted 177 RainQC models
LOW/NO data station impact on models: [('TA00057', 11), ('TA00127', 8), ('TA00715', 8), ('TA00182', 8), ('TA00568', 8), ('TA00185', 7),
('TA00199', 7), ('TA00016', 6), ('TA00414', 6), ('TA00129', 6), ('TA00327', 6), ('TA00621', 5), ('TA00320', 5), ('TA00045', 5),
('TA00587', 5), ('TA00537', 5), ('TA00067', 4), ('TA00231', 4), ('TA00301', 4), ('TA00530', 4), ('TA00565', 4), ('TA00543', 4),
('TA00700', 4), ('TA00636', 4), ('TA00035', 3), ('TA00222', 3), ('TA00041', 3), ('TA00267', 3), ('TA00116', 3), ('TA00126', 3),
('TA00274', 3), ('TA00174', 3), ('TA00217', 3), ('TA00482', 3), ('TA00385', 3), ('TA00289', 3), ('TA00436', 3), ('TA00430', 3),
('TA00542', 3), ('TA00020', 2), ('TA00308', 2), ('TA00050', 2), ('TA00072', 2), ('TA00101', 2), ('TA00256', 2), ('TA00118', 2),
('TA00133', 2), ('TA00136', 2), ('TA00165', 2), ('TA00148', 2), ('TA00164', 2), ('TA00487', 2), ('TA00210', 2), ('TA00223', 2),
('TA00232', 2), ('TA00271', 2), ('TA00399', 2), ('TA00691', 2), ('TA00335', 2), ('TA00339', 2), ('TA00364', 2), ('TA00373', 2),
('TA00397', 2), ('TA00451', 2), ('TA00462', 2), ('TA00471', 2), ('TA00592', 2), ('TA00001', 1), ('TA00014', 1), ('TA00031', 1),
('TA00044', 1), ('TA00062', 1), ('TA00070', 1), ('TA00091', 1), ('TA00095', 1), ('TA00123', 1), ('TA00157', 1), ('TA00212', 1),
('TA00219', 1), ('TA00237', 1), ('TA00251', 1), ('TA00268', 1), ('TA00392', 1), ('TA00269', 1), ('TA00286', 1), ('TA00290', 1),
('TA00336', 1), ('TA00343', 1), ('TA00344', 1), ('TA00362', 1), ('TA00369', 1), ('TA00382', 1), ('TA00389', 1), ('TA00396', 1),
('TA00416', 1), ('TA00422', 1), ('TA00432', 1), ('TA00433', 1), ('TA00493', 1), ('TA00524', 1), ('TA00528', 1), ('TA00529', 1),
('TA00533', 1), ('TA00535', 1), ('TA00652', 1), ('TA00655', 1), ('TA00677', 1), ('TA00702', 1)]
```

- Example: TA00199 is used as a neighbor or target for 7 models, so it prevented 7 target stations from being scored
- This is for general information only, but it suggests that TA00199 should be a high priority to fix, if possible

Note that as the job manager computes data completeness using 'raw' data, TA00057 may already have a quality control flag indicating some problem with it. Ideally, this report (or a dedicated data quality report) would display such relevant station metadata for clarity.

Session Summary

- Total time: 57 minutes + 31 seconds

IIRC, this time is for processing the jobs only and doesn't include time necessary for computing data completeness.

```
-----  
Processed daily jobs for UTC date: 2024-07-01  
Start time: 2024-07-02T05:38:09+00:00  
End time   : 2024-07-02T06:35:41+00:00  
Elapsed time HH:MM:SS: 0:57:31  
-----
```

```
-----  
Before job processing job table stats:  
Total 'success' count: 71  
Total 'failure' count: 199  
Total record count: 1487  
Job history table record count: 175501  
Scoring job record table record count: 662  
-----
```

Session Summary

- I don't remember what the success and failure counts mean "Before" job processing
 - Michael??
- Job history table record count is the total number of jobs that have been created since the database was initialized. This will just keep growing

```
-----  
Processed daily jobs for UTC date: 2024-07-01  
Start time: 2024-07-02T05:38:09+00:00  
End time   : 2024-07-02T06:35:41+00:00  
Elapsed time HH:MM:SS: 0:57:31
```

```
-----  
Before job processing job table stats:  
Total 'success' count: 71  
Total 'failure' count: 199  
Total record count: 1487  
Job history table record count: 175501  
Scoring job record table record count: 662  
-----
```

'before' job processing means what's in the active job table before we remove the jobs from the day before that were either successfully scored or have hit their retry count limit.

'after' job processing is what the active job table looks like after adding the day's jobs and attempting to score these new jobs as well as the previous jobs that we couldn't score (usually due to data unavailability/insufficiency)

Job Results Table

After job processing job table stats:

Total 'success' count:	68	(flag=2 count: 1)	(flag 2->1 downgrades: 4)
'success' count for 2024-07-01:	67	(flag=2 count: 1)	
'success' count for 2024-06-30:	1	(flag=2 count: 0)	
'success' count for 2024-06-29:	0	(flag=2 count: 0)	
'success' count for 2024-06-28:	0	(flag=2 count: 0)	
'success' count for 2024-06-27:	0	(flag=2 count: 0)	
'success' count for 2024-06-26:	0	(flag=2 count: 0)	
'success' count for 2024-06-25:	0	(flag=2 count: 0)	

Anomalies (flag=2):

TA00409 2024-07-01 | score: 168.289 (thresh: 79.492) -- 'pr' t: 0.000 mm n: (6.329 mm, 98 km)

- 68 jobs were successfully run
 - 67 for today
 - 1 left over from yesterday
- Two flag = 2 (“inconsistent”) QC flags were reported
- Four stations scored as “anomalous” (flag 2) by the neighbor regression model were “downgraded” (flag 1) by a special rule that detects and removes false alarms involving low, but non-zero, precipitation values
 - Anyone remember the exact details?
There are two rules, only one of which is used at any given time by RainQC. Rule 1 is that all the stations ≤ 1.0 mm rain, Rule 2 is that all the stations have ≤ 5.0 mm of rain.

Most important result: List of flagged stations

```
After job processing job table stats:
Total 'success' count:          68 (flag=2 count:  1) (flag 2->1 downgrades:  4)
| 'success' count for 2024-07-01: 67 (flag=2 count:  1)
| 'success' count for 2024-06-30:  1 (flag=2 count:  0)
| 'success' count for 2024-06-29:  0 (flag=2 count:  0)
| 'success' count for 2024-06-28:  0 (flag=2 count:  0)
| 'success' count for 2024-06-27:  0 (flag=2 count:  0)
| 'success' count for 2024-06-26:  0 (flag=2 count:  0)
| 'success' count for 2024-06-25:  0 (flag=2 count:  0)
Anomalies (flag=2):
TA00409 2024-07-01 | score:  168.289 (thresh:  79.492) -- 'pr' t:  0.000 mm n: (6.329 mm, 98 km)
```

- TA00409 was flagged as 2.
 - Score: 168.289 is an anomaly score assigned by the model
 - Thresh: is the anomaly threshold (also computed by the model)
 - Because $168.289 > 79.492$, this is flagged as 2
 - Measured precipitation ('pr') was 0.000 mm
 - There is one neighboring station 98km away, and it reported 6.329mm

Asset Dashboard / Sensordx

SensorDX quality reports

Station	Sensor	Date	Target precipitation	Neighbours precipitation	Neighbours
TA00409	S000417	2024-07-01	0.0	6.3	TA00408

- The results also appear here
- However, here the station ids are listed, but not the distances
- In the jobmanager report, the distances are listed, but not the station ideas

station IDs.

Total flags are also summarized in assetdashboard/qc

Quality control report (2024-06-26 - 2024-07-03)

Station	Atmospheric pressure	Precipitation	Radiation	Relative humidity	Temperature	Wind direction	Wind gusts	Wind speed	Soil moisture	Water level	Tilt NS	Tilt EW
TA00409		287										

- I don't see how TA00409 could have been flagged 287 times in just one week. Rick?
- Neither dashboard is sortable or searchable
- No linkage to a time series of the 'pr' readings plotted along with the neighbors (e.g., as a double mass plot or parallel time series plot)

Job table statistics after scoring

- Michael, can you explain this?
- Which of these numbers is the count of jobs that still need to be run?
- How many jobs “timed out” after waiting a week?

```
-----  
Total 'failure' count: 205  
Total record count: 1490  
Job history table record count: 175771  
Scoring job record table record count: 663  
-----
```

IIRC, the 'failure' account is the number of jobs that failed out (hit their retry limit) after the current day's job processing. The record count is the total number of job records in the active jobs table. Which would mean there are $1490 - 205 = 1285$ jobs that were either successful or need to be retried (have not hit their retry limit).

Monthly Summary

```
TA00025 2024-04-18 | score: 1181.327 (thresh: 221.074) -- 'pr' t: 0.697 mm n: (63.908 mm, 5 km), (0.289 mm, 12 km), (0.051 mm, 17 km)
TA00025 2024-04-20 | score: 314.735 (thresh: 221.074) -- 'pr' t: 0.170 mm n: (29.775 mm, 5 km), (0.255 mm, 12 km), (0.748 mm, 17 km)
TA00025 2024-04-22 | score: 560.082 (thresh: 221.074) -- 'pr' t: 59.274 mm n: (51.495 mm, 5 km), (0.357 mm, 12 km), (1.735 mm, 17 km)
TA00025 2024-04-23 | score: 405.521 (thresh: 221.074) -- 'pr' t: 24.526 mm n: (43.587 mm, 5 km), (0.391 mm, 12 km), (0.68 mm, 17 km)
TA00025 2024-04-28 | score: 1102.611 (thresh: 221.074) -- 'pr' t: 33.469 mm n: (68.917 mm, 5 km), (0.272 mm, 12 km), (0.272 mm, 17 km)
```

- Michael produces a monthly summary report. I assume this is a separate script? It's a command line option that can be specified when running the job manager.
- For each station, it prints one row for each day that station was flagged
- In this example, TA00025 was flagged 5 times in April
 - The first two times, TA00025 reported low precipitation when one of its neighbors was reporting high values
 - The final three times, TA00025 reported large values when two of its neighbors were reporting small values
 - These look like false alarms to me