# Deployment Data Cleanup

D Goldsmith, R Wilkins

December 17, 2012

## 1 Introduction

Below is a description of the data summary process, how the information is processed an yields calculated.

It takes the form of a walk though of the yield calculation process, and includes code snippets from the R script used to generate the data.

The document was generated in R and Latex, using the Sweave plugin.

## 2 Summary Table

A new table added to the database, the summary table is intended to hold summary statistics on deployments. This means that future work can avoid having to process entire data sets when dealing with yield, or other summarised functions.

The summary table takes the same form as the reading table, with an additional *summary type* column, these summary types are taken from a lookup table in the database.

Database Rows, and expected inputs are given below

| Row | Type | Description |
|---|---|---|
| Time | PK,Required | Timestamp of summary, In general I would expect this to use midnight to summarise a complete day. However, if more detailed summaries (such as hourly) are needed, this should not be a problem. |
| nodeId | PK,Required | Id of node that this summary is from |
| sensorTypeId | PK | Id of sensor that this summary is from, this can be left NULL to indicate whole node summary samples (for example yield) |
| summaryTypeId | FK | Id of summary type. |
| locationId | FK | Id of location this node is from, to keep parity with the reading table |
| value | float | Value of the summary |
| textValue | string(30) | Optional text description of the summary, for example “Hot” if we are dealing with exposure graphs. |

Table 1: Summary Table Description

## 3 Scripts

This section has a description of the scripts used process the data, and combine all samples into one database.

These scripts are designed to work with the new format (location aware) database format.

They can found in the *dataclense* directory of the *cogent-house/djgoldsmith-devel* repository.

**processCC.py** Transfers current cost data from the old style sqlite database, into the new format database.

**processAr.py** Transfers data from an Archrock postgresql database into the new format database.

**getStats.R** R script that calculates yields for each deployment in a given database

**calcKwh.R** R script to caluclate KwH usage from current cost readings.

Further details of these scripts are given below

# 4 getStats.R

This script calculates summary statistics for all houses in a given database. The statistics are output in two formats.

- *.csv* file with summary output for this database
- update rows in the *summary* table given these statistics

To run the script modify the source file with the relevant database access name. Then run the script through R.

## 4.1 Script initialisation

- Load the relevant R librarys
- Connects to the database
- Loads the Relevant Lookup tables into memory.
  - Houses Table
  - Sensor Table (For Calibration)
  - Sensor Type Table
  - Summary Type Table

## 4.2 Loading the Inital Dataset

This section of code fetches the relevant data for each property.

First the relevant house is fetched, and the start and end dates processed to convert to a format R is comfortable with.

Next the Relevant locations are fetched using the following query This query also fetches the corresponding room names for each location for use in a lookup table if required.

The Start and End dates of the deployment are calculated, and converted into a format that R is happy with.

Finally, the relevant dataset is fetched.

Rather than fetch the entire dataset, and process it on the local machine, the query used makes use of SQL summarising functions to group the data by day. The query fetches the following

**nodeId** Used to Identify the Node

**locationId** Used to Identify the Location

**date** Timestamp converted to a date (i.e. with the time removed)

**minTime** Time of first sample on this date

**maxTime** Time of last sample on this date

**minVal** Minimum reported Value on this date

**maxVal** Maximum reported value on this date

**meanVal** Average value reported on this date

```
[1] "Processing House  131Jodrell"
```

```
[1] "Expected Start  2011-12-12 00:00:00 Expected End 2012-02-17 00:00:00"
```

## 4.3 Calibration

For the purpouses of error detection, it should be sufficent to examine the minimum and maximum values.

The dataset is calibrated by merging with the calibration table, to append the calibration slope and offset (where known), where there is no calibration data, a default value of 1 for slope, and 0 for offset are used. From this a calibrated version of the Min,Max and Mean values is calculated.

Figure 1 shows the calibrated data, before any error handling takes place.

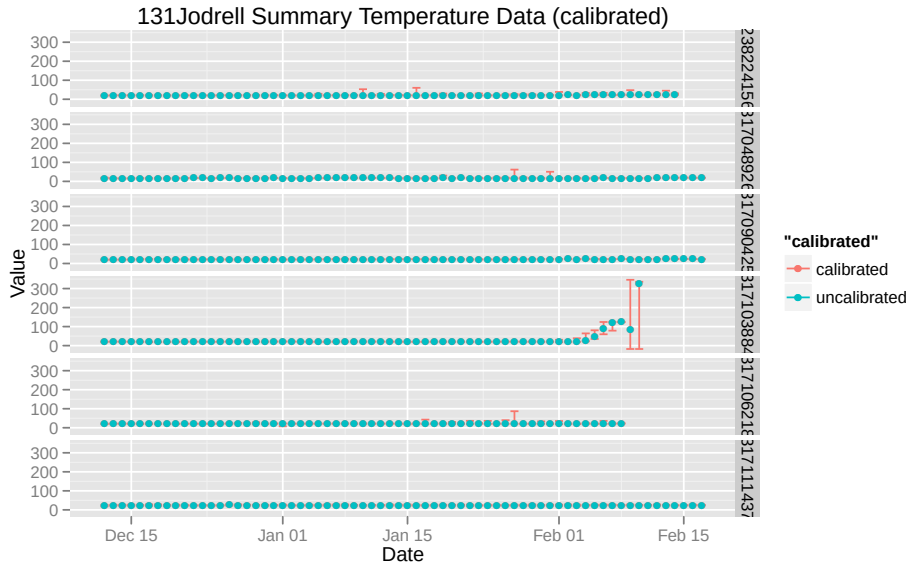


Figure 1: Calibrated Data

## 4.4 Error Detection

Errors in the data are detected using a basic threshold based approach: Dates where the minimum or maximum values fall out of the ranges in Table 2 are marked as invalid.

*NOTE:* Given that with the Telos the data for temperature and humidity sensors tends to go bad when the battery voltage falls below 2.3V, it may be appropriate to filter any temperature values when the battery drops below this level.

| Parameter | Minimum Threshold | Maximum Threshold |
|---|---|---|
| Temperature | -10 | 50 |
| Humidity | 0 | 100 |
| Co2 | 0 | 6000 |

Table 2: Accepted Ranges

Where data is marked as invalid, the full data stream for that day and node is collected.

This full data stream is then calibrated and the bad readings removed using the method above.

Each day is then summarised, and the original entry in the *houseSummary* data frame replaced with the new summary values. Additionally, a new column *badCount* is added to the data table to represent the total number of samples before any data is removed. This may allow some insight into the number of bad samples per deployment.

```
> plt <- ggplot(subset(fixCalib,type==0))
> plt <- plt+geom_point(aes(dt,calibValue,color=badValue))
> plt <- plt + ylab("Value") + xlab("Date")
> plt <- plt+opts(title="Calibrating 'Bad' data")
> plt <- plt+facet_grid(nodeId~.)
> print(plt)
```