AMR Database Report

A report on the schema and construction of a database to collect data for cows on farms

Dhruv Chauhan
overseen by Prof. Seth Bullock
03.08.16

Introduction

This report outlines a prototype of a possible design for the creation of a database for the use of collection of information about cows on farms (currently including information about 7 farms in Northern Somerset). This includes the metrics that various sources have stated it would be useful to incorporate, and other data that it would be ideal to import from other databases.

In this context, what is meant by a database is set of data held in a specific structure, which would be accessed through a webpage. This would then make it easier for people to access the database, to contribute and pull data from it. This would also hopefully encourage a more systematic way of entering data.

The specific technical schema that is outlined is simply a possible implementation of the many that could exist, and since it is still in its early stages may be lacking in certain features such as distributability, security and reliability. However, for the current scale of the system, it should fit well.

Metrics

Having consulted various sources, the inclusion of the following metrics in the database is important. A way of including this usefully in the database can be seen in the Document / Model Schema which is used as part of the technical prototype in the relevant section. Derivative data such as percentages is excluded.

On the Farm

- farm ID / number (*unique identifier*)
- number of cows (milking, adult, young, beefing, calfs)
- their average approximate weights
- number of samples taken
- number of staff
- number of isolate samples taken
- amount of ESBL isolate samples
- number of multidrug resistant isolates
- the amount of resistance to the following antibiotics (may need expanding)[1]
  - SXT - *trimethoprim / sulfamethoxazole*
  - CAZ - *ceftazidime, 3rd gen cef*
  - TE - *tetracycline*
  - EFT - *ceftiofur, 3rd gen cef*
  - CIP - *ciprofloxacin*
  - CTX - *ceftriaxone, 3rd gen cef*
  - FOX - *cefoxitin, 2nd gen cef*
  - AML - *amoxicillin*
  - NA - *fluroquinolone*
  - IPM - *imipenem*
  - C - *chloramphenicol*
  - P - *penicillin*
  - CL - *cephalexin, 1st gen cef*
  - AMC - *amoxicillin clavulanate*

[1] the data to import would relate to these antimicrobials

- assorted questionaire data, such as:
  - are there footpaths on the farm?
  - do milking cattle have access to a water course?
  - do you footbath the milking herd?

The final questionnaire data is often stored using a 'key' based system, which is perhaps a good indicator that it can be stored as a separate entity, and decode the number 'key' into an appropriate response.

Audit Data

The audit data is complex and varies from source to source, therefore the following is only a selection of the data that could be audited.

- animal daily dose (mg / kg / day),
- cost,
- use per litre,
- batch number,
- and tx rate of:
  - systemic antibiotics
  - anti-pneumonials (first line and protected)
  - topical antibiotics (first line and protected)
  - intrauterine antibiotics (first line and protected)
  - teat sealent
  - anti-microbials

Misc.

- geographic location
- footpath data
- high-level ordering by year
- farmer's self-reported AM use
- bin audit data

To Import

Some of the people interviewed stated that in order to easily do their calculations / to reduce unnecessary manual updating it would be useful to include drug information automatically (preferably with live updating). This could be extracted from a system such as NOAH[2], which publishes updated drug information every year.

In addition to this data, basic common calculations would ideally also be pre-computed for ease.

[2] http://www.noah.co.uk

High-Level Overview

Introduction

According to the prototype, this database would ideally function in the form of a webapp. This would mean that any farmer, lab worker, researcher etc. would be able to visit the webpage and login with their credentials to access the database.

From the database, it would be easy to submit any of the metrics outlined above, preferably in the form of simple text input or drop-down menus, in order to reduce any unnecessary clutter on the page.

In addition to this, access to (anonymised) data would be readily available in a variety of formats in order to do further processing on. Ideally there would be an ability to produce and view simple graphs / charts with the data inside the webapp.
Moving forward from the basic webapp, there would be a mobile version that allowed easy access / entry on the farms.

Prototype

The prototype built, dubbed 'Cowllection' meets some of the criteria outlined above, as it is a work in progress. It contains some of the resistance data that has been provided so far, along with other general farm information. The prototype itself is supposed to demonstrate that it is possible to move some of calculations towards the back, leaving the user with a simple way to access their data. Also, it encourages a more systematic way of contributing data, with a more high-level, user-friendly approach for less technical users.

Access to the prototype can be done through github, and by following the instructions on the README page: https://github.com/dhruuuuuv/amrapp

The entry page can be seen below, and other screenshots are collected in the appendix.

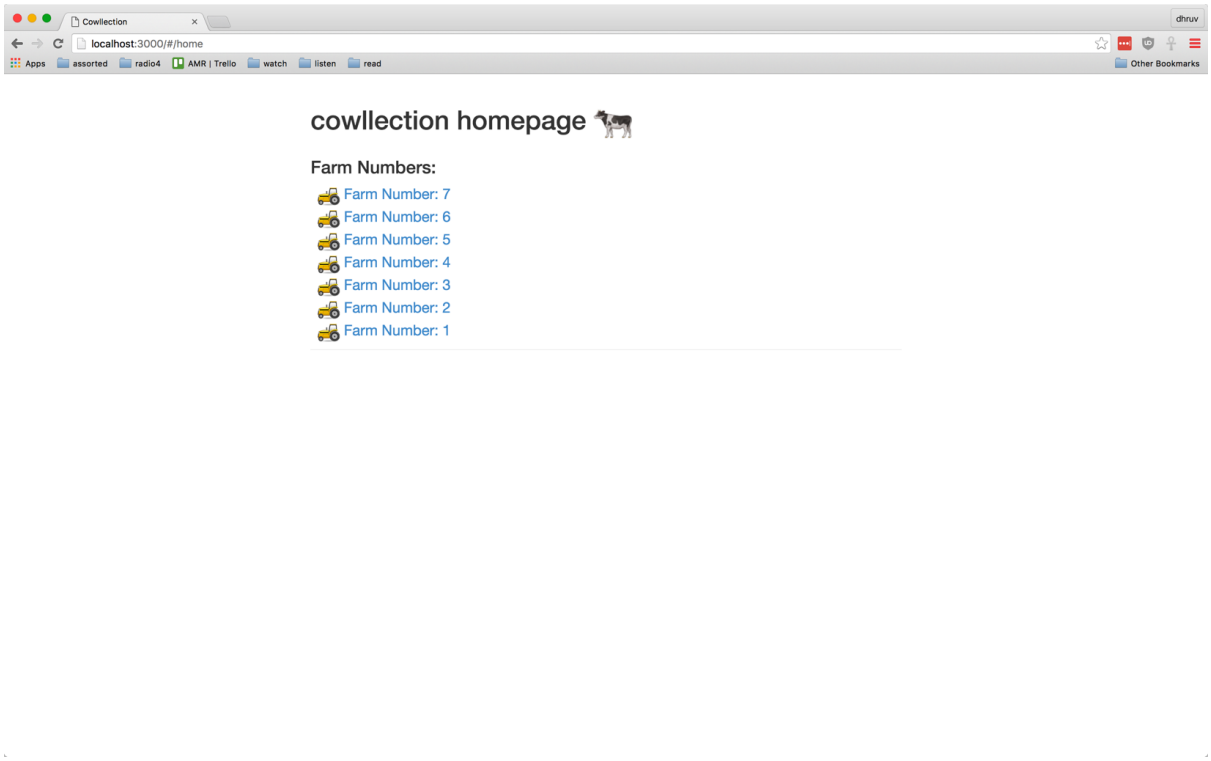

Figure 1: Entry Page

The README file is inserted on the next page:

README.md

Synopsis

This project is a prototype for an online database to collect information about cows and farms. It relies on the MEAN stack.

This means that it requires Node.js, which can be downloaded from:
https://nodejs.org/en/download/

Installation

To use:

- open a terminal / command prompt, and `cd` to a directory.
- type in the command `git clone https://github.com/dhruuuuuv/amrapp.git`
- change into the new directory with `cd amrapp`
- ensure node is installed by typing `node -v`. If it is a version number should be displayed.
- enter the command `npm start`, and you should be greeted with: `> node ./bin/www`
- in your web-browser, type in the address bar `http://localhost:3000`, after a while you should be greeted with the cowllection homepage!

Usage

Navigate through the app by clicking on the relevant **farm number**. This should then reveal a **farm overview** with multiple **animal id**s.

If you click on an animal id, the information about that ID is revealed below, so **scroll down** to reveal the new information about the specfic cow.

Each cow has associated isolates, which also by clicking on the relevant **isolate number** will reveal the specific data about that isolate.

At any point, click on another **animal id** to view information about that, or another **isolate number** for that. Click on the HOME button to return to the farm overview.

Technical Overview

Technologies Used

This prototype was constructed using the MEAN stack. This means it used: [**4**]

- Node.JS, a server side javascript environment,
- Express, a framework used for routing the urls and connecting the back-end to the front-end,
- A MongoDB database, to act as the actual database for the data to reside inside. It is a NoSQL system,
- Mongoose, a mongoDB object modeller for node.JS
- Angular.JS, a front-end framework with two-way data binding, and
- Bootstrap, a HTML and CSS framework.

All of these frameworks are javascript based.

Reasons for selection

mongoDB

MongoDB is a noSQL database with no schema. This means that it is best suited to data without specific relations - in which case a SQL database would fit better. The lack of certainty about the fields to be included led me to choose a noSQL database. The system also allows an inherent hierarchy through embedded documents, which I deemed modelled the Farm hierarchy well. It then readily converts this to JSON to be passed forward.

Angular.JS

This front-end solution to MVC (model-view-controller) easily allowed access to the data brought forward by the node / express / mongoose framework. Angular splits MVC up into different components (services, controllers), and then pipes these together. The front-end can be written in HTML and using pre-existing technologies such at Bootstrap and angular-chart.js. Overall Angular was chosen for its ability to seamlessly integrate with other technologies - perfect for a stack.

The Mongoose 'Superfarm' Model / Schema

Mongoose provides a Schema to access the data from the mongoDB. The Schema demonstrates the imagined hierarchy on a farm, which is then used to model the hierarchy through the webapp. By following the Schema, one can see how the data is laid out in the database, in the backend and in the frontend.

```
1   var Superfarms  = new mongoose.Schema({
2       farm_number             : {type: Number},
3       cows                    : [ {
4           animal_id               : {type: String},
5           parity                  : {type: Number, default: 0},
6           latest_calving          : {type: Number, default: 0},
7           latest_calving_days     : {type: Number, default: 0},
8           isolates            : [ {
9               isolate_number      : {type: Number, default: 0},
10              sample_number       : {type: Number, default: 0},
11              date_sampled        : {type: Number, default: 0},
12              date_sampled_days   : {type: Number, default: 0},
13              DIM_at_sampling     : {type: Number, default: 0},
14              mean_305d_milk      : {type: Number, default: 0},
15              antimicrobials          : [
16                  { name: String }, {value : Number, default: 0 }
17              ]
18          }]
19      }],
20  });
```

which can be simplified down to:

```
superfarm
├── farm number
└── cows
    ├── animal id
    ├── parity
    ├── latest calving
    ├── latest calving days
    └── isolates
        ├── isolate number
        ├── date sampled
        ├── date sampled days
        ├── DIM at sampling
        ├── mean 305d milk
        └── antimicrobials
            ├── name (example SXT)
            └── value
```

Farmvet Systems Case Study

FarmVet Systems produce VetIMPRESS Software which allows vets and farmers to integrate and monitor data on farms. In addition they produce the mobile application myVI. I spoke with the CEO of the company, George, who answered some general questions about both the technical and user side of the company.

Technical

The desktop application uses a SQL database, and is powered by .NET applications. The mobile application is using a Cordova frontend, with a C# and Java backend, and a SQLite database. These were chosen as they were the available skills to the team, and there is a relatively small user base. The mobile application is intended to be the main entry point to the overall system.

The app uses SSL connections for security.

Users

The CEOs are vets, which aided in the construction of the apps, in order to make useful, convenient features. In order to make the system as user friendly as possible, the development was heavily customer led.

The users need about a day of training before they can use it. Most of the learning is done by people that use the app and then ask for support.

General user feedback has been positive, they have a constant update cycle which allows them to adapt to the user's requirements.

Data

The data is fairly open and easy to export. The company functions also functions as a data engineering company, as the first 2 years of development were spent forming integrations and links to get animal data, government data and farms data, which would then be standardised.

Their back end is highly focused around cleaning the data, using techniques such as fuzzy matching, which then feeds live into the systems.

Evaluation

Need

The perceived need for this arose out of the lack of consistency occurring through the use of different methods of collecting data, such as via pen and paper, excel spreadsheets and CSV files. This discrepancy makes it difficult for others to do further processing on the data, or to fairly easily extract certain bits. The prototype was designed in mind of eliminating these problems, along with migrating the information over to a more rigorous structure. This aims to show that some of the non-essential data can be moved behind the scenes.

Criticisms

Since the prototype is still in its infancy, the following are deemed 'key features' that would need to be added on before it could be rolled out:

Security Currently anyone can access the information, and meddle the data that exists / launch false requests. In order to deal with this, implementation of OAuth, perhaps using a system like Passport (with Google Sign-in) is essential.

Exportability The data is only stored in the MongoDB database, and the backend produces JSON from this for the frontend to use. There would be a frontend method of downloading this JSON file, or flattening it into a CSV file.

Deployment The mongoDB is currently configured to use a database hosted by the service MLab. For this to be officially deployed, the database would need to be hosted on a server, such as AWS. In addition to this, the rest of the 'MEAN' stack would need to be implemented on a sever also, with a domain name to allow global access. All this server would then require is Node.js and MongoDB installed.

Imports By importing the data that has already been published, this allows specific, live and accurate calculations to be done. An example of this is from NOAH, outlined above.

I believe the prototype's ultimate use comes from the ability to extend out the basic structure. With this comes the flexibility to adapt to the requirements of the users - ranging from implementing a full usable stack to having an option to refine down the required data and then export it to a csv file, to be used by for further processing. In short, the prototype demonstrates that the move away from disparate excel spreadsheet files isn't infeasible.

References

[1] Antimicrobial Resistance (AMR) - GOV.UK
https://www.gov.uk/government/collections/antimicrobial-resistance-amr-information-and-resources, 2014.

[2] NOAH
http://www.noah.co.uk/

[3] Pig Hub
http://www.pighub.org.uk/iip.home.eb

[4] Introduction to the MEAN Stack
https://www.sitepoint.com/introduction-mean-stack/

[5] Farmvet Systems Ltd
http://www.farmvetsystems.com/

Appendix

People Consulted

Harriet PhD researcher planning to further propagate the database.

Andrea PhD researcher collecting data at Langford, analysing isolate samples for AMR.

Gwen PhD researcher also collecting farmer self-reported AM use data with bin tallies.

George CEO of FarmVet Systems, producer of VetImpress Software which allows vets and farmers to integrate and monitor data on farms.

Additional Prototype Screenshots

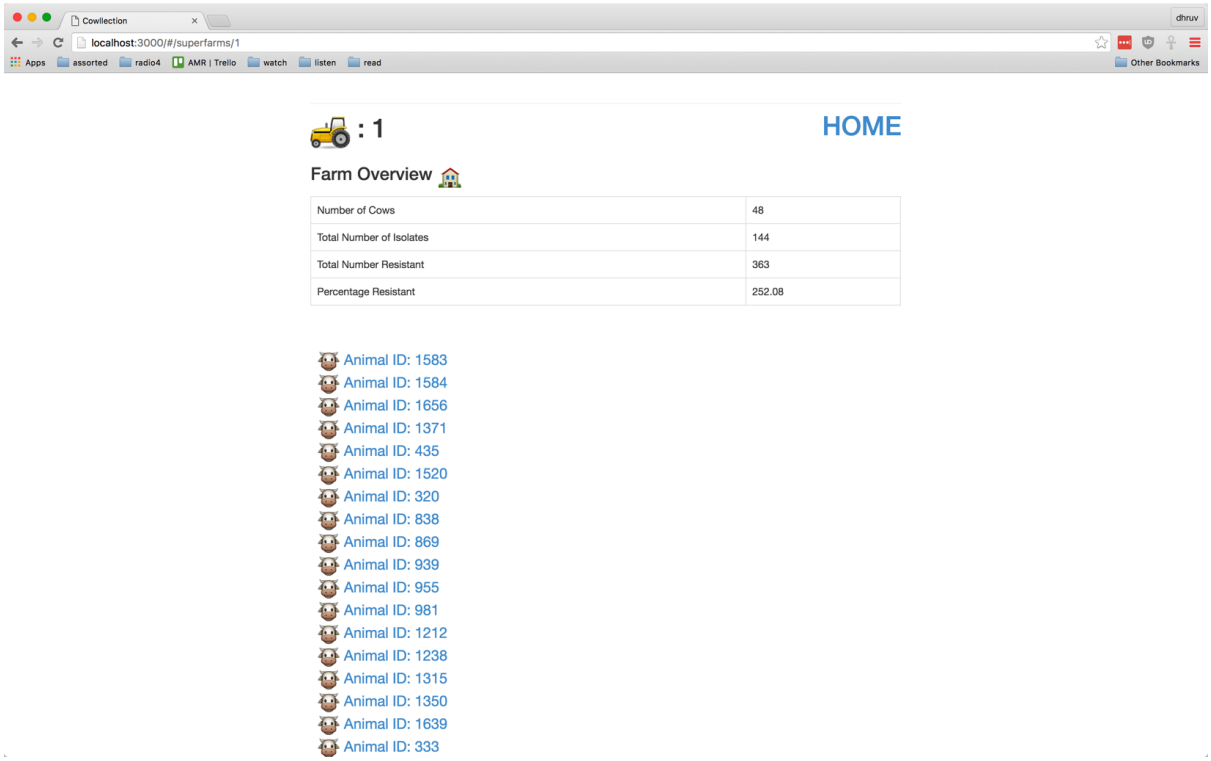

Figure 2: Farm Overview

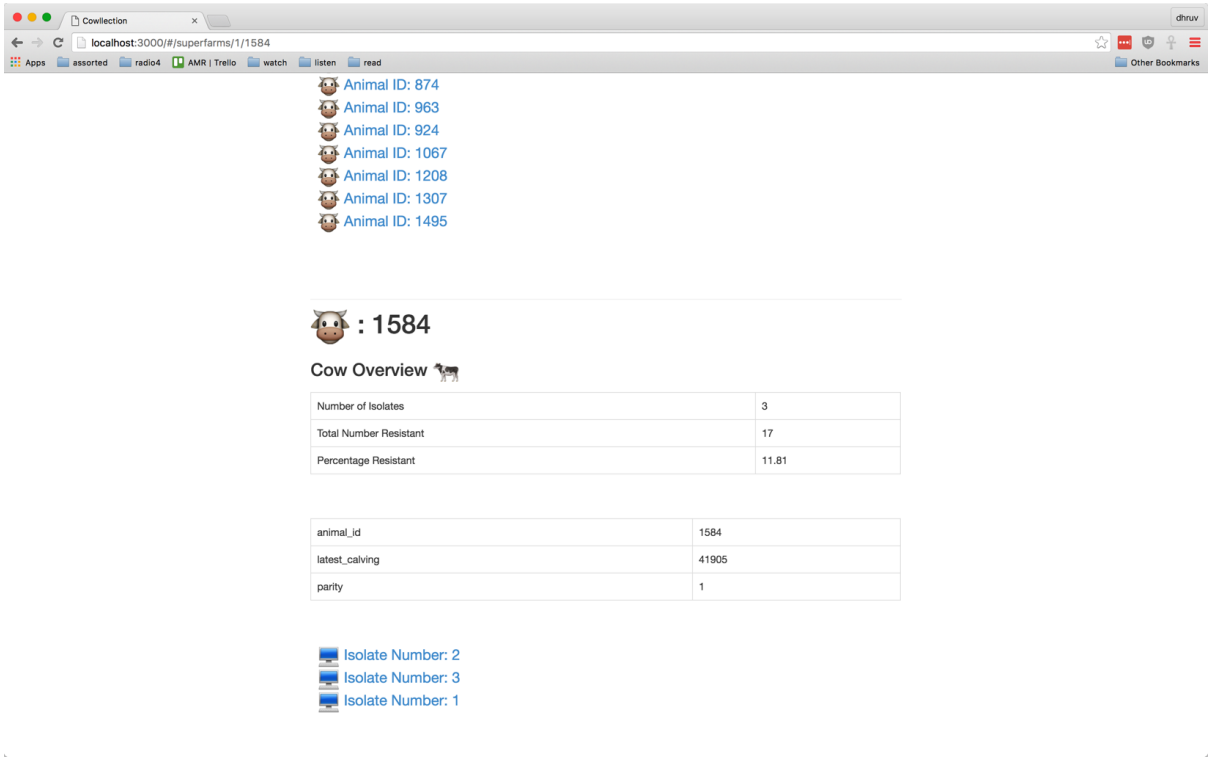

Figure 3: Cow Overview

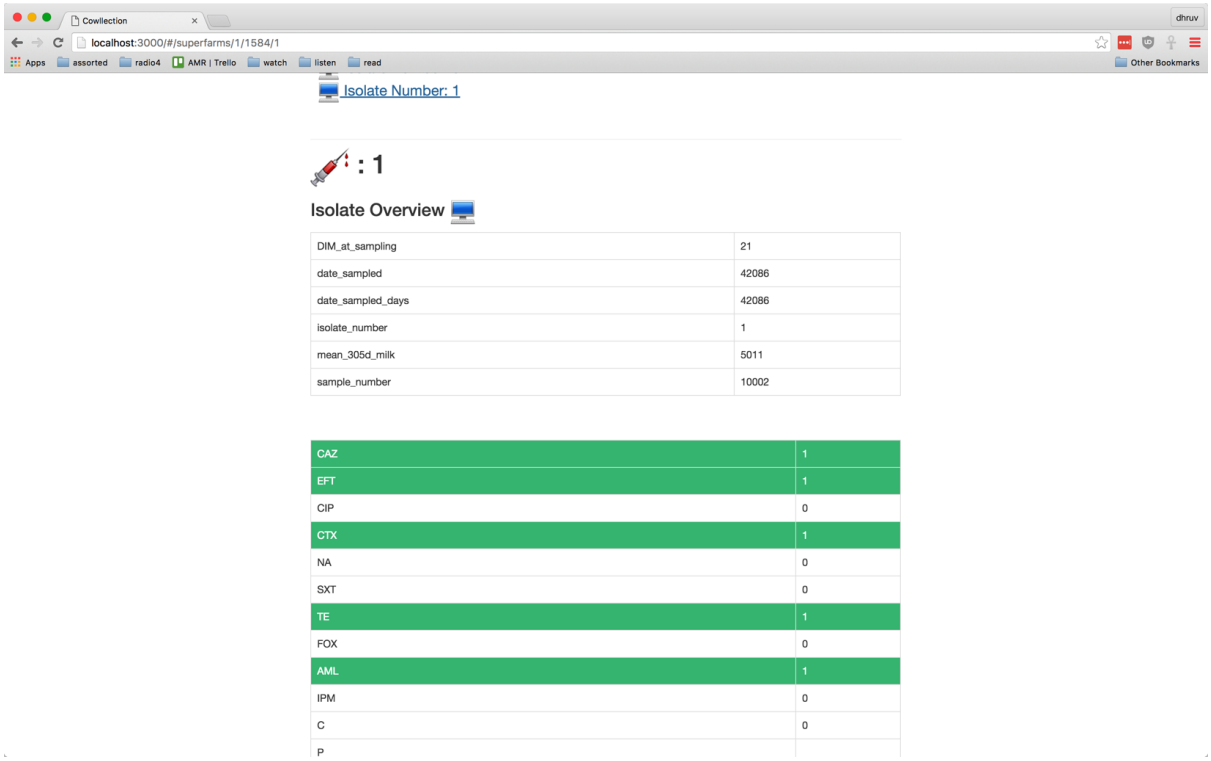

Figure 4: Isolate Overview

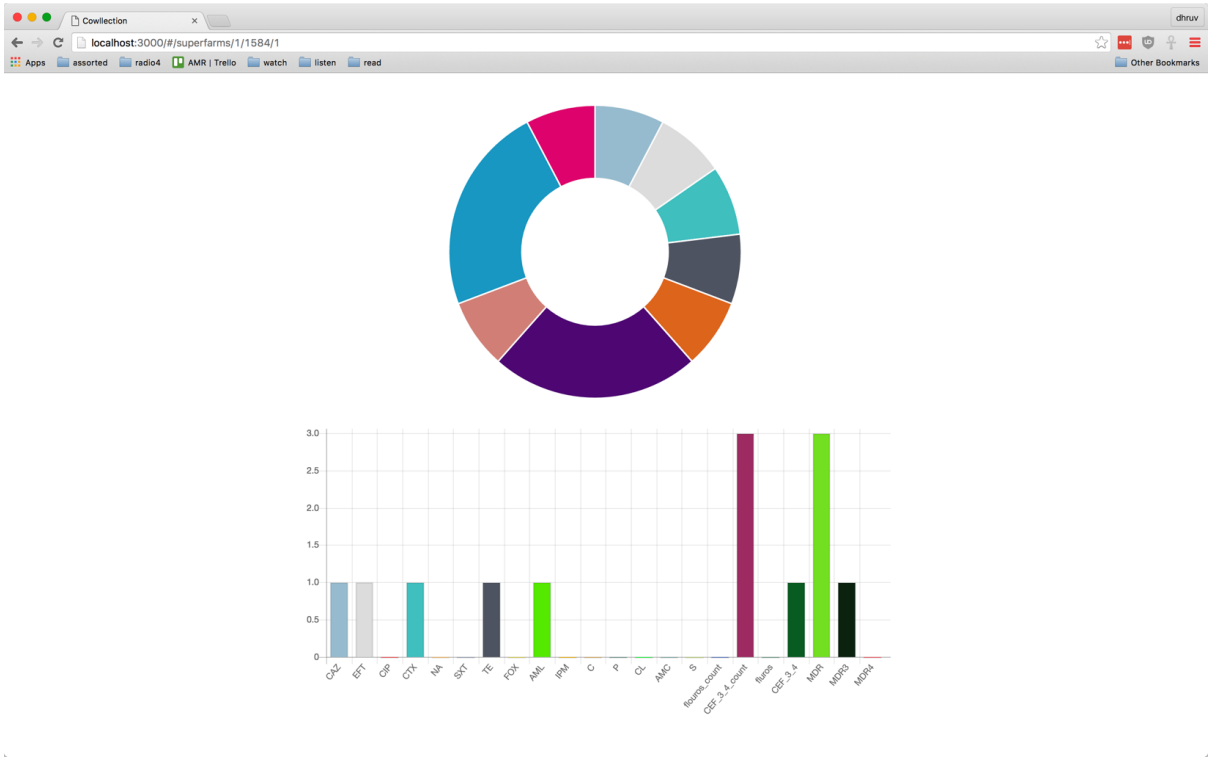

Figure 5: AMR Overview