Report On Version 1 Of PoCRA Field Dashboard

Rahul Gokhale,

Consultant to IIT team

Table of Contents

Introduction

The Maharashtra Project On Climate Resilient Agriculture (PoCRA) is a World Bank aided project being implemented by Government of Maharashtra in 15 drought prone districts of Maharashtra. The project development objectives are to enhance climate-resilience and profitability of small-holder farming systems in project districts of Maharashtra. The IITB-PoCRA collaboration has been done to undertake the development of water balance based scientific planning methodology for surface water and soil moisture management, in order to develop resilience of farmers against climate vulnerability and change.

Memorandum of Understanding (MoU) 2 [see *MoU-2,* 2] was signed between IITB-PoCRA as a follow-up to MoU-1 [see *Closure document*, 1] that had involved among other things, the development of a soil-moisture oriented water-balance model and a consequent village-level water budgeting framework[3]. MoU-2 consists, among other important things, the development of a GIS based dashboard to monitor and analyze key agricultural field-level water parameters, many of which are estimated by the model developed in MoU-1, and thus enable advisories to the farmers for resilient agricultural decisions in the project area. This report documents version 1 of this Dashboard as part of the deliverables of Phase-2 in MoU-2.

The remaining introductory section briefly explains the motivation for the dashboard. The next two sections describe the functional and architectural aspects of version 1 the dashboard. The final section summarizes the report by putting version 1 and the upcoming version 2 of phase-3 into an overall dashboard development perspective.

Motivation

MoU-1 between IITB and PoCRA produced a location-based soil-moisture water-budgeting model. Being location-based, it can produce estimates of water-balance components (like infiltration, actual evapotranspiration(AET), etc.) only for the location(latitude-longitude) for which it is run. To use it in practice for a region, a QGIS-based desktop plugin was developed that would run and produce estimates for all points of a uniform grid overlaying a specified cluster. However, to expand the utility of the model, it or at least its results had to be publicly available for assessment and monitoring. This prompts the idea of a web-based dashboard application.

This idea quickly extends to a web-based dashboard application that can provide many such related functionalities as follows.

- The primary idea behind the dashboard is to be able to monitor, track and assess the latest situation in the field in a geo-referenced manner.
    - Geo-referencing the field-level information is most convenient and safe with modern digital GIS technologies. Thus, the dashboard can provide a reference point for all GIS data within the project. More importantly, the dashboard being

    publicly accessible, all the data can be automatically attested by all stakeholders and corrections, if any, may be reported immediately.
    - Tracking the field situation can be daunting when done at large regional scales. The model developed during MoU-1 can allow tracking the estimated field situation based on daily rainfall and other information that is relatively easily available. Thus, the dashboard can provide a monitoring and tracking facility for the estimated field-status.
    - Various aggregate assessment tools may be provided through the dashboard based on the estimates generated from the model and also for related available data.
- Other important use-case is to track and assess the current status of project activities. Although this use-case is not the direct concern of the deliverable proposed in MoU-2, the field dashboard gains practical importance in a project like PoCRA, only when it is sufficiently linked to the project's activities. In other words, it would be of practical value to geo-reference the data related to project activities and make it (or links to it) available on the field-dashboard.
- Action-research has to be an important component of any project like PoCRA that aims to develop sustainable solutions by the end of its implementation for climate-resilient agriculture. With the GIS and related database technology already playing an integral role in the design of the dashboard, it would be a significant value addition to have support for basic analysis of the information that is made available on the dashboard. Going further, the dashboard can act as a significant component of a technical decision support system for choosing and implementing solutions proposed in the project.
- The features that the users of such an action-research project may desire are bound to evolve over the project duration. In order to accommodate such evolution, the development of the dashboard has to be done in stages that essentially amount to versions of the dashboard when viewed as an IT product.

While the overall motivation for the dashboard gives an impression of a very broad scope web-application, the deliverables of MoU-2 are focussed on implementing the estimation model and making its results publicly available for tracking. The first version will set the basic platform that implements the basic model with fixed et0 values[3] and allows the user to view maps of the estimated parameters. The second version, as planned within the scope of MoU-2, has to implement a refined model wherein the et0 values are computed on a daily basis using the weather parameters. It also needs to incorporate more features for the users to access and query geo-referenced information for assessment purposes.

Functional aspects

Three functional aspects of the dashboard are envisioned based on the various use-cases elaborated earlier.

1. Estimation model
2. Website/webpage
3. Database

Described below are the salient features of each these functional roles of the dashboard.

Estimation model

Monitoring and estimation of the amounts of various soil-water components in the field has been the primary motivation behind the field-dashboard. To this end, we use the soil-water balance model provided as one of the deliverables from the MoU-1 between PoCRA and IITB. [3]

This model is a point-based model that estimates the daily changes in the various soil-water balance components like infiltration, soil-moisture, actual evapotranspiration as well as the pre- and post-soil-moisture phase components like runoff and groundwater recharge. To do this estimation, it requires as input the data about the soil-texture and depth, the crop and the amount of daily rain at that point.

In addition to the above data requirement, the model also requires reference evapotranspiration values - termed as ET_0 - at the point for each day of the crop-duration. These values are generally derived from the values of various weather parameters on the day on interest. WALMI[3] has derived a set of ET_0 values per month per district of Maharashtra. For convenience of implementation, we consider these suggested static(fixed) set of values for the model as a first approximation.

Since the model is a point-based model, in order to get a picture at a regional scale we need to select points of interest within the region and run the model simulation for each of these points. The conventional method to choose the points of interest is to generate and maintain a uniform grid of points covering the region. Following this paradigm we use a uniform grid of points covering the entire region of 15 districts of PoCRA and run the model simulation for each of these points.

The functionality of model-based soil-water balance estimation relies on the database functionality of the dashboard as follows. All the input data required by the model will be made available from the database. Most of this data is static and will be pre-loaded into the database. The remaining data that is dynamic on a daily scale, like daily rain (and other weather parameters for later use in deriving refined estimates of ET_0), will be fetched on a daily basis by this model-estimation software module from the weather monitoring source and will be uploaded to the database before being used for estimation.

The model will run once every day. It will use the previous day’s field status and the static data from the database and the latest available daily dynamic data fetched from the weather monitoring source, in order to generate estimates for the field status on the current date. These estimates will again be stored back into the database for use in estimation for the next day.

Website/webpage

As the name “Dashboard” suggests, the primary functionality of the dashboard will be to provide a user-interface that allows the tracking and monitoring of various technical aspects within the project. Since it is required to be publicly accessible, the most suitable form to provide such a functionality would be to create a website or a webpage embedded within any existing or planned master website for the project.

The webpage should provide access to every geo-referenced technical data that is to be made publicly available. Since almost every dataset to be published through the dashboard will be geo-referenced, web-mapping technology, i.e. providing map-based visualization on the webpage(s), will be an integral part of this functionality.

All geo-referenced data is either in vector format (like ESRI shapefiles) or in raster format (like GeoTiff images). No vector datasets are to be made publicly available through the project. It is thus imperative to provide all geo-referenced data only as raster images on the webpage. This in turn implies that a map-rendering functionality be used in the background to convert vector datasets to images for display on the webpage.

The model-estimation functionality generates estimates for each point of the grid covering the region. To represent the values of any estimated parameter(water-balance component from the model) at all points of the grid, the parameter values at all points are geo-referenced and converted to an image, each of whose pixels are coloured based on the parameter value at that point. This image is then displayed on the webpage.

A facility to request such a map has been provided on the webpage. The user has to select the crop, the date and the parameter of interest and then submit the request. The map image then gets overlayed over the existing layers in the map-section on the webpage.

In addition to raster maps for parameters, there already exists a corpus of data that is geo-referenced into vector format and which needs to be made publicly available on the dashboard. Such tabular, yet geo-referenced data will also be made available in a separate data-section on the webpage. This data-section is nevertheless linked to map-section for visualization of the vector data whenever necessary; that being one of the motivations for a dashboard.

Finally, when presented with any dataset, whether raster(image) or vector(tabular), any curious user would want to explore it and find characteristics of the dataset in order to develop an understanding about it. To this end, it is useful to have at least a basic facility on the webpage to query the presented datasets, for example to shade pixels of a raster image by user-chosen values or filter rows of a vector table by values of some specified column.

Database

PoCRA is being implemented in selected clusters of 15 districts of Maharashtra. Monitoring and analysis of various hydrological data relevant to agriculture, throughout these clusters and over the duration of the project, will be an important part of the project. Given the significance of the geographic distribution of both surface and ground water resources, it is necessary to have all this data geo-referenced. Furthermore, it would be most desirable to have a single source of reference for all geo-spatial information in the project. The dashboard should thus provide geo-spatial database functionality that can act as the reference for all primary geo-spatial information related to the project.

Most of the geo-spatial data that is already available for the project is static vector data. This includes:

- a shapefile of the administrative boundaries for all the villages in the 15 PoCRA districts
- a shapefile for the cadastral plots in the villages that have been selected for implementing the project
- a shapefile demarcating the land-use throughout the 15 districts
- a shapefile demarcating the soil characteristics throughout the 15 districts

In addition to this, a raster of digital elevation map (DEM) covering the 15 districts is also already available. The source for all these datasets can be found in [3]

While the geo-spatial data that is already available is primarily vector data, the model-estimated soil-moisture related data and for most use-cases even the monitored weather data are meant to be visualized essentially in raster(image) form. Thus, raster storage would be an essential functionality expected from the database.

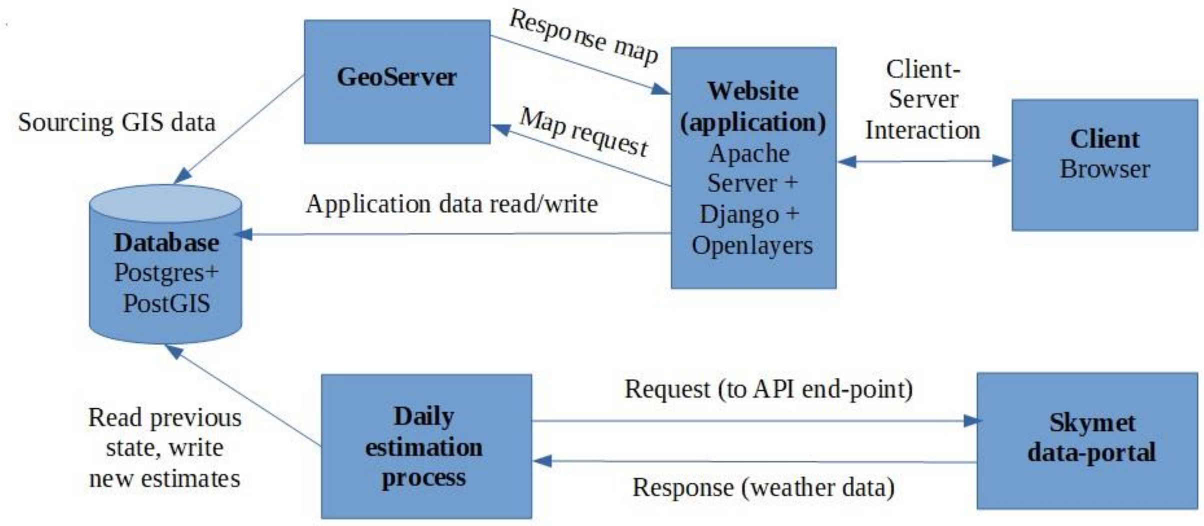
Apart from the geo-spatial data, the estimation model also requires auxiliary look-up data that provides numeric constants for various categorical inputs. Such values are required in the various empirical estimation formulae employed in the model. For example, the empirical formula for runoff estimation requires a constant termed as the curve number that is determined based on the soil-texture category and the land-use category at that point. Such data will invariably be best to be stored in the database.[3]

A significant part of the project's implementation is expected to respect the various crop-and-water realities in the field, almost all of which have been and will be geo-referenced. Furthermore, project implementation will be aligned to the administrative units (village, cluster, taluka, district), all of which are essentially geo-spatial units. As such, tracking and visualizing the status of project's implementation has a significant geo-spatial aspect to it. With the dashboard intending to serve as the single source of reference for geo-spatial information in the project, it is necessary that the database directly incorporates or indirectly links to such non-technical, yet geo-referenced (or geo-reference-able), data in the project.

Architectural Design and Implementation

The implementation of the dashboard is made up of several components, most of which are common to websites that provides web-mapping facilities. The functionality of model-based estimation, however, is a distinct one and is incorporated as will be described below. We first describe the overall architecture of the implementation that explains how every component of the implementation fits in. Next, we provide details for each of the major components. The reader is assumed to have basic familiarity with web technology.

Software Architecture


GeoServer
Response map
Map request
Website (application) Apache Server + Django + Openlayers
Client-Server Interaction
Client Browser
Sourcing GIS data
Application data read/write
Database Postgres+ PostGIS
Read previous state, write new estimates
Daily estimation process
Request (to API end-point)
Response (weather data)
Skymet data-portal

Components of the Dashboard and their linkages

Depicted above is the broad software architecture for the dashboard. Client browser and the API for weather data provided by Skymet is are the external entities with which the dashboard will interact over the internet. The dashboard itself has four major components as follows.

1. **Database:** This precisely addresses the primary database functionality as envisioned and described earlier. We have used the free and open-source Postgres relational database system to build the actual database. It provides a powerful extension called PostGIS that enables storage, manipulation and access of geo-spatial datasets. The Postgres+PostGIS combination has a large usage world-wide and robust open-source community support.
2. **(Daily) Estimation Process:** This precisely addresses the model-based estimation functionality of the dashboard described above. This software module is developed using the Python language along with its official as well as community contributed library of useful packages.
3. **Website/webpage:** This addresses the main part of the web-based access functionality of the dashboard envisioned above. The free and open-source httpd server developed by the Apache Foundation, has been used as the web-server. We have used the python-

based Django web development framework for the backend and the React web-application development framework for the frontend. The OpenLayers javascript library has been used to provide web-mapping features on the browser.

4. **GeoServer:** This is a free and open-source GIS map server that implements the Web Map Service protocol to serve map images from the geo-spatial data source. This is the only component that will be used off-the-shelf, after basic configuration to link it to the PostGIS geo-spatial data source.

The overall working of the dashboard can be understood by understanding the interaction between these four components. This interaction as depicted in the figure above, is as follows.

The database stores all the data. The estimation process fetches weather data from the API provided by Skymet's weather data-portal. Then, from the database, it loads the status of field conditions (i.e., the values of soil-water balance parameters) for the latest date that has already been estimated earlier. Next, using this status and the newly fetched weather parameter values, it runs the model and estimates the field-status for the next date. This new status is stored into the database tagged by the date.

Whenever the user requests some data using the browser, the web server receives the request. All requests for maps(like the map of, say, soil-moisture status for some given date) are routed to GeoServer. GeoServer has been configured to use the Postgres+PostGIS database instance as the source of GIS data. As such, it internally fetches GIS data from the database and creates the required map from it. This map is sent back to the webserver, which then sends it as its response to the browser's map request. Those requests which do not need the mapping functionality, are tackled directly by Django-based application's request handlers specifically written for such requests. As and when necessary, the request handlers also fetch data from the database to generate the response that is sent to the client-browser.

In the following sections, we provide the details of the implementation of the three components, i.e. other than GeoServer which merely required configuration.

Database design and implementation

- We have used the Postgres object-relational database system, version 11.1, extended with PostGIS version 2.5 for geo-spatial data.
- Vector data available in ESRI shapefile format for cadastral boundaries, village boundaries, soil-characteristics demarcation and land-use demarcation has been imported into the database system using the 'shp2pgsql.exe' utility program provided by the PostGIS's product distribution. Similarly, raster data for DEM has been imported into the database system using the 'raster2pgsql.exe' utility program provided by PostGIS's product distribution.
- Vector tables for various administrative boundaries have been derived from respective attributes attached to the village boundary vector data. Specifically, the village boundary vector table contains village polygons along with the names of cluster, revenue-circle, taluka and district to which each village belongs. By grouping villages according to their

district, for example, and then dissolving the boundaries between adjacent villages in each group, we get the district boundaries. Vector tables for clusters, revenue-circles, talukas and districts have been derived in such manner.

- The following auxiliary look-up datasets, as required by the model-estimation process, have been imported into the database
  - Constants related to each crop's characteristics
  - Constants related to each soil-texture category
  - The so-called curve-number constants (used in runoff calculation) as have been empirically determined for each soil-texture category paired with land-use category
  - Fixed ET_0 values per district for each month.
- Also, daily rainfall data for the past 5 years, from 2013 to 2017, for the revenue-circles in the 15 districts as available on Maharain website has also been loaded into Postgres.
- Since the estimation process requires the database for all the data for model estimation and storage of estimation results, special tables have been created for this purpose. In particular, a raster table, named 'pd_field_static_params_100', has been created to store the static field parameters required for estimation at each point of a 100 metres resolution grid covering the entire region of 15 PoCRA districts.
- The daily soil-moisture status as estimated by the model is stored for all points of the grid as a raster in a table named 'field_daily_state'. Similarly, the estimates of each of the soil-water balance components are stored in rasters in a table named 'field_daily_water_params'.
- Currently, no provision has been made to incorporate other project-related data into the database. However, when required later, such data may be linked to from the dashboard's other components as a separate database instance or may be synchronized with the dashboard's database instance.

Implementation of the estimation model

- The estimation process is written as a Python 3.7 program which relies on the Postgres+PostGIS database for storage as well as computation. PostGIS provides a large library of raster processing functions which can be used to process all the points of the grid when represented as a raster; each pixel of which represents a point. Using this approach all the necessary input parameters at all point are encoded into rasters, one per input parameter(like one raster for daily rainfall values at the grid-points, etc.). Then using the raster processing functions, in particular the so-termed map-algebra functions, the model estimation formulae are applied to each pixel of the input raster to generate the output rasters, one for each desired output parameter (like one raster for the infiltration values at the grid-points, etc.). The 'psycopg2' module provides necessary bindings between Python and Postgres+PostGIS.
- Currently, this estimation process has been provided as an offline utility program with a command-line interface(CLI). The concerned personnel from the PoCRA office can use it to generate estimates for any desired input scenarios. In particular, there are options to set the crop, the year and the mechanism to decide crop-sowing dates. There are two

mechanisms to decide the sowing date at each point; either mention the sowing-date directly of specify a threshold value for the total-(monsoon)-rainfall-till-date, which when exceeded as days pass by, determines the sowing-date. This utility program has been used to generate estimates for the previous years for a specific set of crops.

- The provided offline utility program is a python file named 'estimate.py' that itself acts as the command to the CLI. All the necessary options to run the estimation model are to be given as command-line arguments following the 'estimate.py' command. The most helpful of these is the '-h' (help) argument; so that typing 'estimate.py -h' will display all available arguments and options to run the model. Given below are two examples of issuing commands to run the model.
    - `>estimate.py cotton 2017 --sowing_date 2017-06-25`
      As it suggests, this will run the model for cotton for year 2017 with sowing-date as 25th June for all the points of the grid (having default resolution as 100m) in the 15 districts
    - `>estimate.py tur 2018 --sowing_threshold 30 --resolution 500 --till_date 2018-11-30`
      This will run the model for tur crop for year 2018 with sowing-threshold as 30mm, so that the sowing date at every point of the specified 500m resolution grid depends on when the total monsoon rainfall at that point exceeds 30mm. The model is run only upto 30th Nov as the till-date option specifies.
- This process is meant to be run as a daily scheduled task that fetches weather data from the Skymet's API and runs the model to generate next set of estimates. This feature will be enabled after getting satisfactory feedback from PoCRA's users about the estimates of previous 5 years.
- The 100m grid overlying the 15 districts has about 12.5 million points and as such the model takes significant time to run for each day (any crop). The offline utility program for estimation has been provided with a 5-sec timer-based user-prompt after every date's run for aborting the estimation process, if the user so desires. Nevertheless, for user-convenience, provided below are approximate average running times of the model for each resolution.

Resolution	Approximate average running times for each date's estimation (any crop)
100m	11 minutes
200m	3 minutes
400m	1 minute 10 seconds
500m	55 seconds
1000m	40 seconds

Website design and implementation

- The website has been designed with a complete separation of concerns between the backend and the frontend. The backend is developed using the python-based Django framework, while the frontend is a React application.
- We have used Django version 2.1 to provide all the API required for the website. There are just two main endpoints in this version of the dashboard.
    - Maps: All map requests have a ‘map’ endpoint.
        - For a raster map request, it is responsible for exporting the PostGIS raster of the relevant raster map to the GeoServer’s data-directory.
        - For a vector map request, it is responsible for setting the values of the relevant attribute of the vector table, as may be used to colour the vector features according to those values.

      The appropriate URL corresponding to the requested map-layer that is to be added to the map is then generated and sent to the browser as a response.
    - Point-data: The other requests supported currently are for the data at any given point on the map. In particular, the request is received with the latitude and longitude of the desired point as the parameters. All the static information relevant to the point that is to be made public is retrieved from the database and sent as a response in JSON format. This includes the administrative location of the point, the soil-characteristics and land-use properties at the point.
- We have used React version 16.8 as the framework for the frontend. The ‘Ant Design’ React UI framework, version 3.15, has been used for React-based UI components for interaction with the user. Openlayers version 5.3 is used for providing web-mapping facilities. In addition to these javascript libraries, other purpose-tailored javascript packages like konva, moment, etc. have been used for their niche roles. All these free and open-source frontend development components have reached considerable maturity and have witnessed extensive popular usage and support from the open-source javascript development community. Node and its package manager have been used as the basic tool to manage and build the frontend web-app files.
- Shown below are examples of the webpage usage:
    - As Figure A shows, since the dashboard application is focussed on web-mapping, the webpage has been designed to display a map in full-screen. It shows various layers in a separate ‘Layers’ panel which is expanded for display on clicking the ‘L’ button on the top-right. The ordering of the layers is as per the z-indexing of the layers on the map.
    - Figure B shows the panel for making map-requests. It has two tabs: Raster and Vector. Appropriate choices for crop, parameter and date can be made in the provided forms and then submitted. This will load the corresponding map-layer onto the existing layers on the map. One such example is shown in Figure C.
    - Figure D shows how clicking at a point on the map displays its static data in an information panel on the right while also pinning the location with a popup.

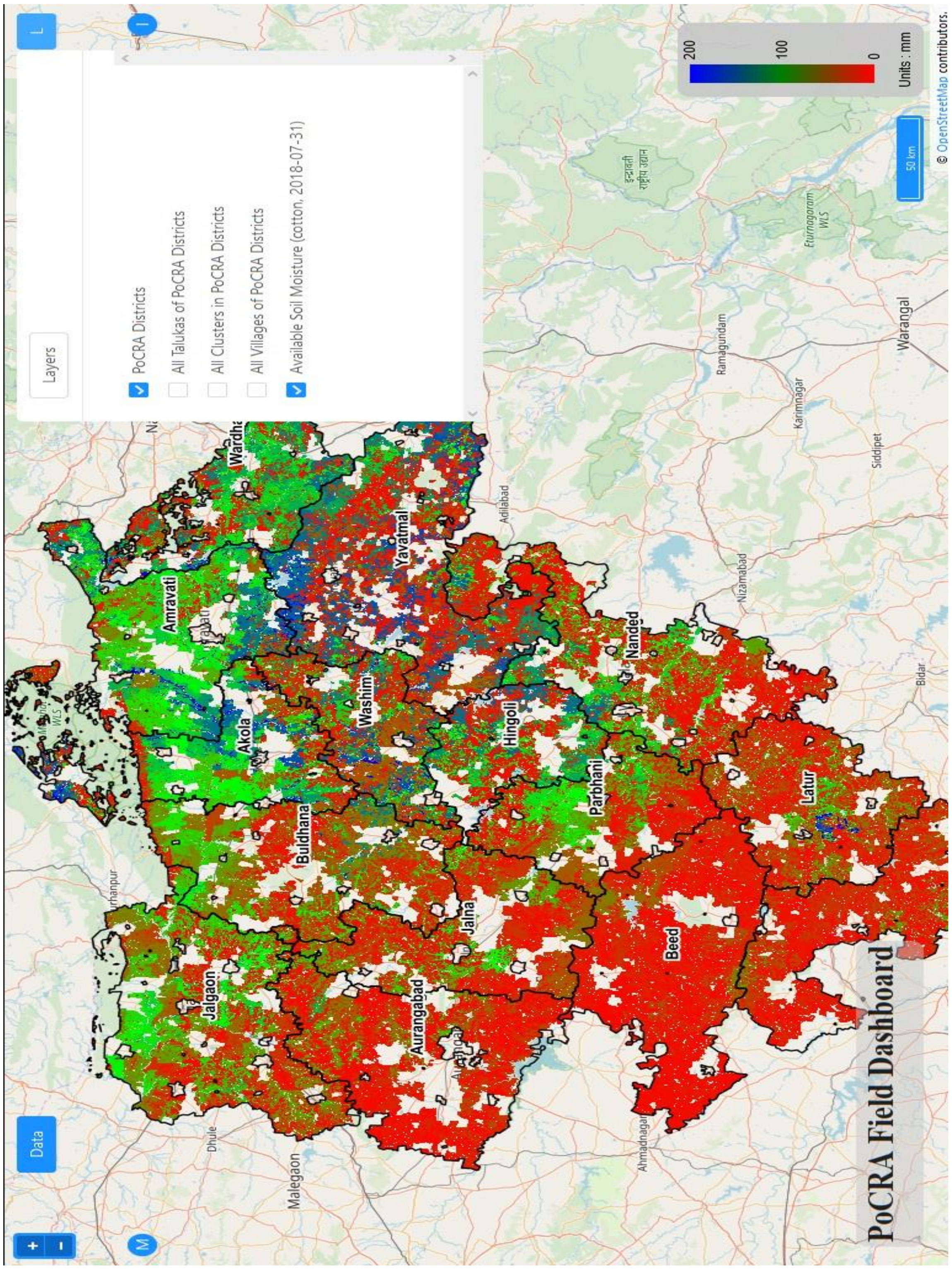
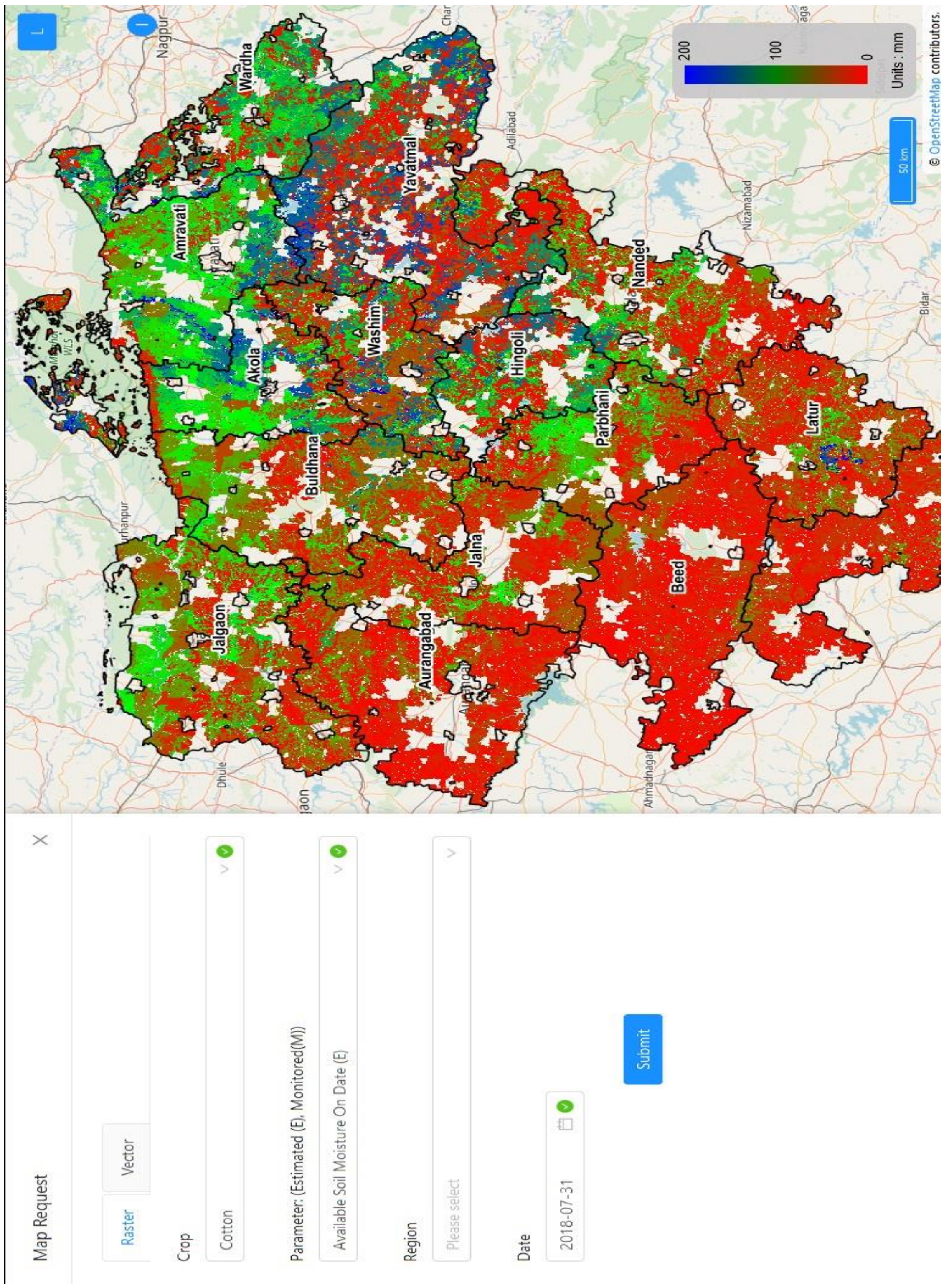

PoCRA Field Dashboard
Layers
PoCRA Districts
All Talukas of PoCRA Districts
All Clusters in PoCRA Districts
All Villages of PoCRA Districts
Available Soil Moisture (cotton, 2018-07-31)
200
100
0
Units : mm
50 km
Data
Jalgaon
Aurangabad
Buldhana
Jalna
Beed
Akola
Washim
Hingoli
Parbhani
Latur
Nanded
Amravati
Yavatmal
Wardha
© OpenStreetMap contributors.

Figure A: Dashboard showing various layers


Map Request

Raster | Vector

Crop: Cotton

Parameter: (Estimated (E), Monitored(M)): Available Soil Moisture On Date (E)

Region: Please select

Date: 2018-07-31

Figure B: Dashboard showing the Map Request panel

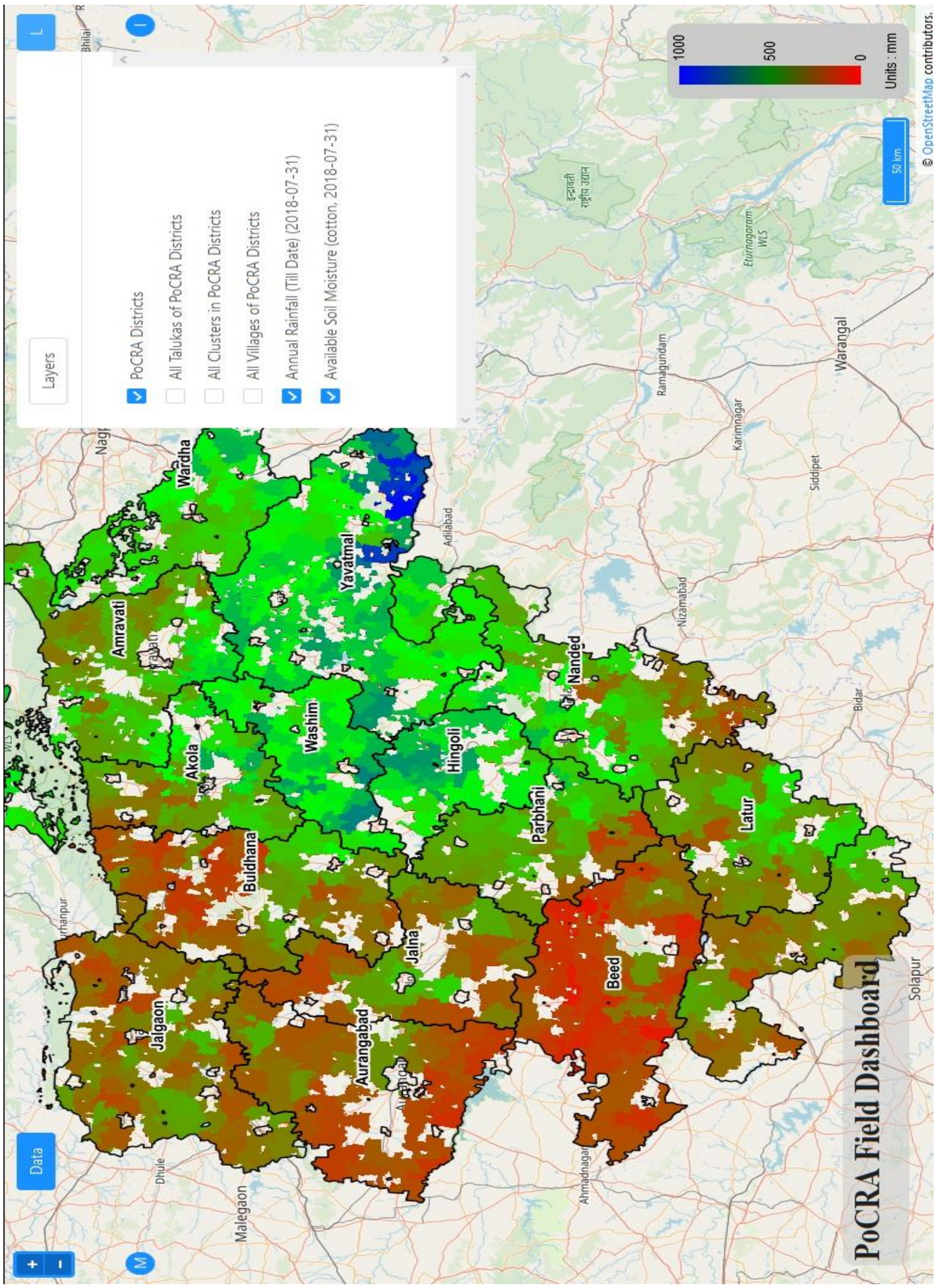
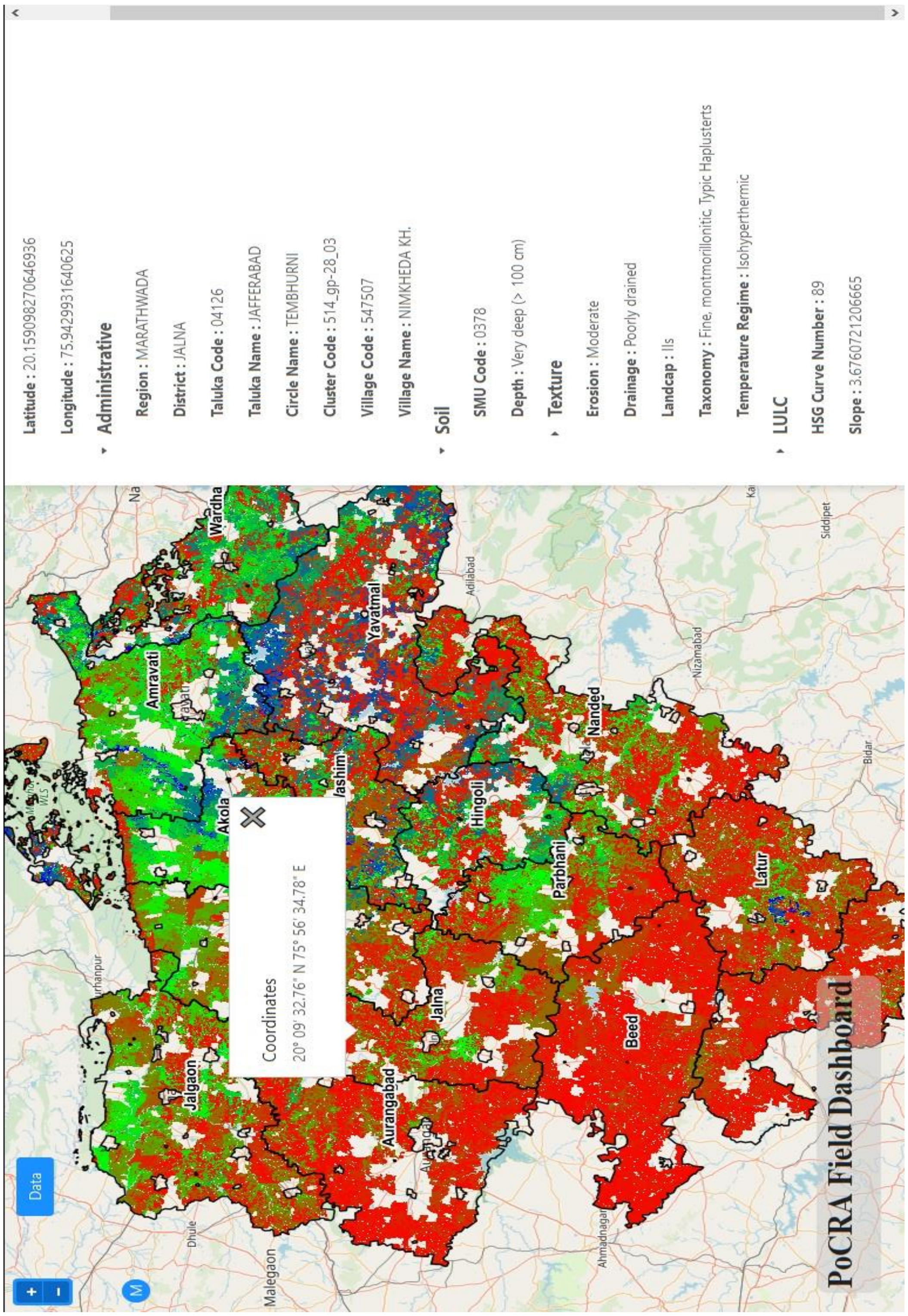


Figure C: Dashboard showing the total rainfall map loaded after a request



Latitude : 20.159098270646936
Longitude : 75.9429931640625
▾ Administrative
Region : MARATHWADA
District : JALNA
Taluka Code : 04126
Taluka Name : JAFFERABAD
Circle Name : TEMBHURNI
Cluster Code : 514_gp-28_03
Village Code : 547507
Village Name : NIMKHEDA KH.
▾ Soil
SMU Code : 0378
Depth : Very deep (> 100 cm)
▸ Texture
Erosion : Moderate
Drainage : Poorly drained
Landcap : IIs
Taxonomy : Fine, montmorillonitic, Typic Haplusterts
Temperature Regime : Isohyperthermic
▸ LULC
HSG Curve Number : 89
Slope : 3.6760721206665

Figure D: Dashboard showing a clicked point on the map along with its static data in the Info panel

Summary

Following is an overall software development perspective that points out the capabilities and limitations of version 1 of the dashboard and describes the salient improvements to be achieved in version 2.

What version 1 enables

- The model has been implemented to carry out soil-water-balance estimation over the entire region of the 15 PoCRA districts for any given crop and any given year. This enables generation of estimates for both those villages which are included in the project as well as those which are not included in the project but lie in the 15 districts and hence potentially have similar weather conditions. Such complete-region estimation enables a comparative assessment between project and non-project villages.
- The dashboard webpage provides web-mapping facility for rasters of water-balance components estimated from the model. These raster layers span the entire extent of the 15 districts of PoCRA. These rasters are overlain by maps of administrative boundaries of districts, talukas, clusters and villages. Progressively finer-level administrative boundaries become visible as the user progressively zooms in the map. This enables visual assessment of field situation at increasingly finer resolution.
- Estimates have been generated both for daily values of water-balance components as well as for their accumulated (from June 1st of the monsoon-year) values. This is especially useful for chronological assessment of the overall magnitude of any particular water-component at, say, the monsoon-season end or the crop-end date.
- Access and basic querying of rainfall vector data (circle-wise) is also provided; but only at geo-unit level, i.e. cluster-wise in a given taluka or taluka-wise in a given district, etc.

Limitations of version 1

- Version 1 of the dashboard is being delivered as a snapshot of the product in development phase. Although it can potentially be, but is not meant to be delivered as a publicly accessible web application at this stage of MoU-2. It is primarily intended to set the stage for version 2 through an introduction of the application to the concerned PoCRA personnel. Suitable feedback received from such introduction is planned to be accommodated in version 2. Version 2 should thus become a suitable candidate for public release.
- Version 1 does not include access to other project-related geo-referenced data. Particularly to be mentioned is village-wise and cluster-wise data that can provide better decision support and (possibly real-time) evaluation of project’s technical activities.

- In version 1, the estimation model has been implemented directly using PostGIS's map-algebra functions for efficiency. However, it has been noted after considerable exploration that these map-algebra functions are not conveniently expressive for spatial aggregation tasks. In particular, there appears to be no convenient, yet efficient, method to achieve geo-unit-wise spatial aggregate values (like village-wise average soil-moisture) from the estimated parameter rasters (soil-moisture raster in this case). All this means that such geo-unit-wise spatial aggregate values are not available as maps (like cluster-wise average soil-moisture map for some crop, etc. are not available).

Features to be incorporated in version 2

- As mentioned above, the current implementation of the model does not provide geo-unit-wise spatial aggregates. To remedy this, and to incorporate quite a few other programming related improvements, the model estimation component will be re-implemented in Python, while reusing any code from version 1 that serves its purpose well. Owing to the flexibility of Python, geo-unit-wise aggregates could then be efficiently computed during the estimation process. This primarily implies that maps of geo-unit-wise aggregates will be available in version 2.
- A couple of suggestions received during the mid-phase demonstrations of the dashboard application at PoCRA office, have been incorporated into the features for version 2.
    - Calibrable colour-ramp for pseudo-colouring any map-layer by pixel/attribute value. Essentially, this will allow the user to choose the colouring of pixels in raster layers and of features in vectors (based on attribute values).
    - Animating a sequence of chronologically ordered layers. This is meant to specifically provide animations of rasters for parameters (like daily rain) between user-specified dates. Such visualization can potentially provide better insight into the field status.
- Originally, the dashboard has been envisioned for the long-run, as a technical decision support GIS system for the project and possibly even beyond. As per the MoU-2, however, version 2 will be delivered only within a couple of months after the delivery of version 1. That leaves little time to develop such a broad purpose web-application as a decision support information system. More crucially, not everything is clear as to what kind of technical-decision situations would be critical during the project implementation. So it is difficult to pin down the exact long-term expectations from such decision support system.
  Nevertheless, there is one possible approach that can make provisions for extending the dashboard to such a system easier. The basic idea is to make the application pluggable, wherein any later-developed external (to dashboard) application could be plugged into the dashboard website interface (not requiring it to be embedded into the Django backend or React frontend code directly) by requiring it to expose a particular API designed for such purpose. In order to test and demonstrate this concept, version 2 will incorporate a 'pluggable data' interface. Using this, any geo-units related data (not necessarily geo-referenced) residing in other trusted databases may be used to produce

map-layers on the dashboard. Such data access will also be powered by table-rows filtering facilities to enable basic data exploration.

- One last, but not the least important, feature that is feasible within the version 2 delivery schedule is that of allowing selective content to be downloadable. This would be implemented, in particular, for map-layers to be downloaded as images. Download handles may also be provided for data tables selectively.

References

1. *MoU-1 Closure Report*, Sohoni Milind, Gupta Parth, Sali Shubhada https://www.cse.iitb.ac.in/~pocra/MoU%20I%20Closure%20report.pdf
2. *MoU-2* https://www.cse.iitb.ac.in/~pocra/MoU-II.pdf
3. *Adaptation refinement, and transfer of full Package to PMU*; a report that documents the estimation model used in the dashboard, https://www.cse.iitb.ac.in/~pocra/Phase%20III%20Plugin%20description%20document.pdf