Building a Conservation Irrigation Water Management Application

Presented by Paul Senne

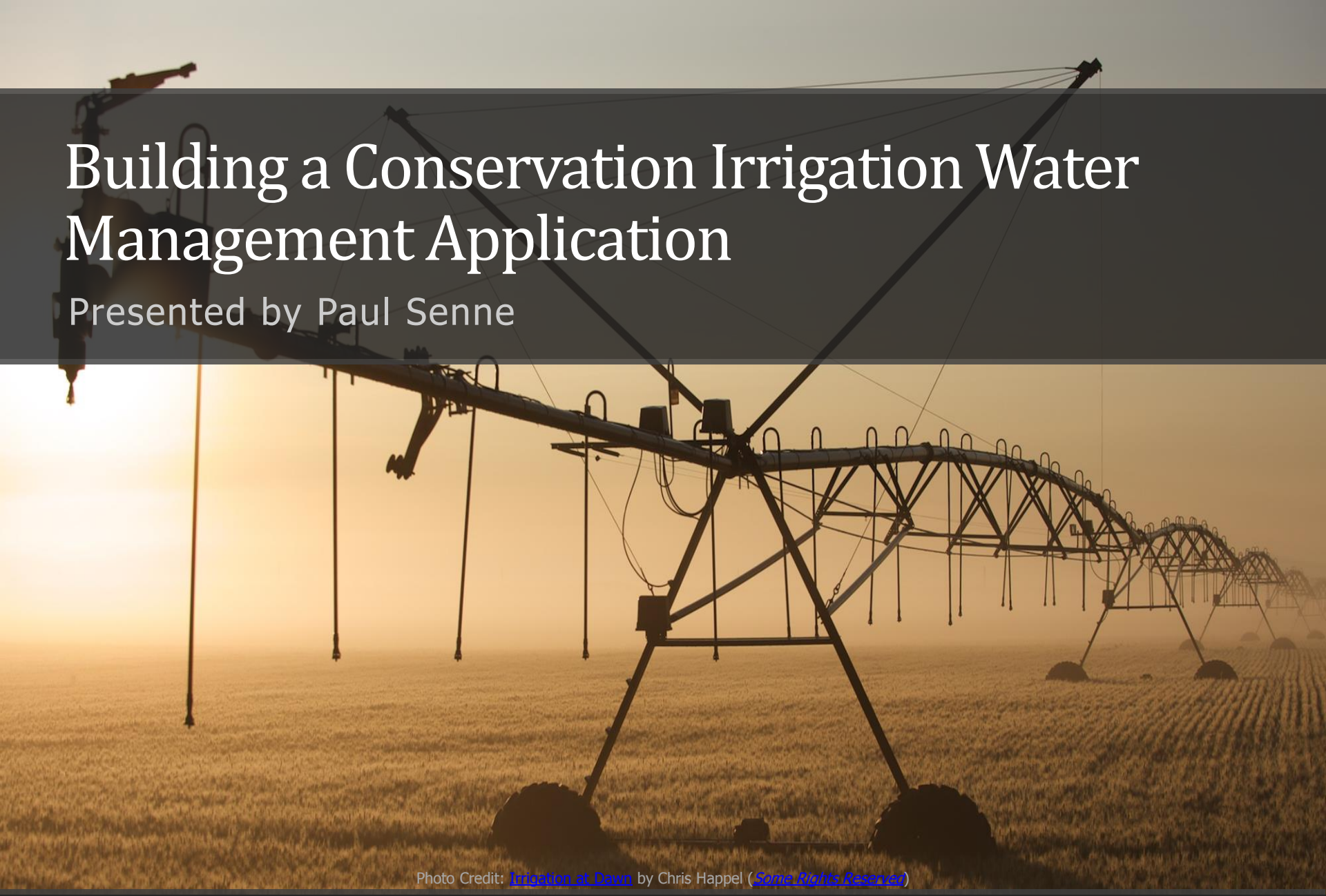


Photo Credit: Irrigation at Dawn by Chris Happel (*Some Rights Reserved*)

RESPEC

Building a Conservation Irrigation Water Management Application

Presented by Paul Senne
paul.senne@respec.com

Overview

- Background
- Irrigation & Conservation
- Irrigation scheduling
- Monitoring Conservation Indicators

- Software Stack
- Inputs to the Irrigation Management Assistant (IMA)
  - Farm Field / Crop Planting Setup
  - Daily Scheduling
  - Resource Monitoring
- Data Model

- Demo

Background

- MN Soil and Water Conservation District (SWCD)
  - Benton County SWCD
- Grant from Minnesota Environmental Trust Fund
- In response to watershed with reduced stream flows
- Open to any irrigator

ENVIRONMENT AND NATURAL RESOURCES TRUST FUND

Irrigation & Conservation

- Groundwater is main source of irrigation water
- Groundwater also feeds surface water streams
- Reduced stream flows can lead to impairments
- Crops and yields depend on irrigation on some soils
- So…. when should we irrigate…?

Irrigation Scheduling

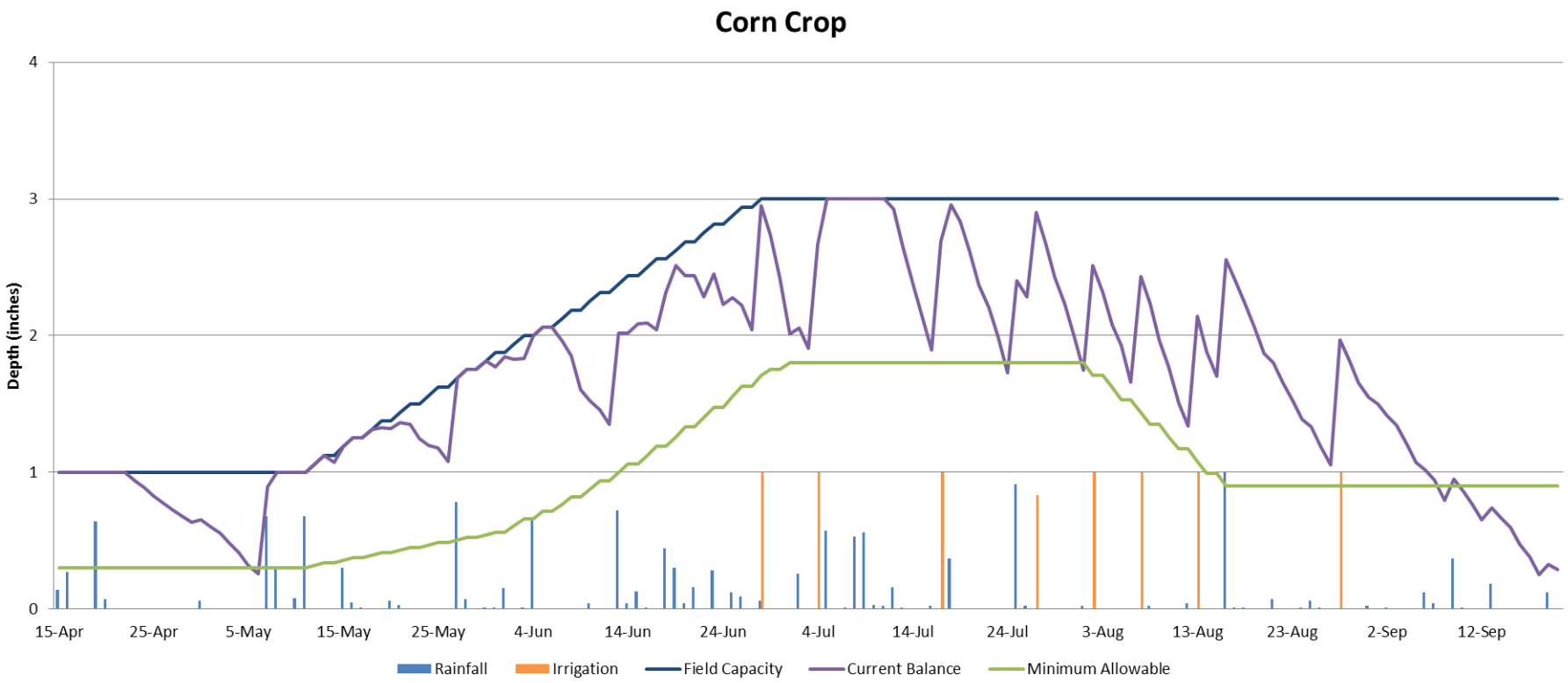
- Scheduling Methods

Irrigation Scheduling

- Applying science to verify/ensure best practice

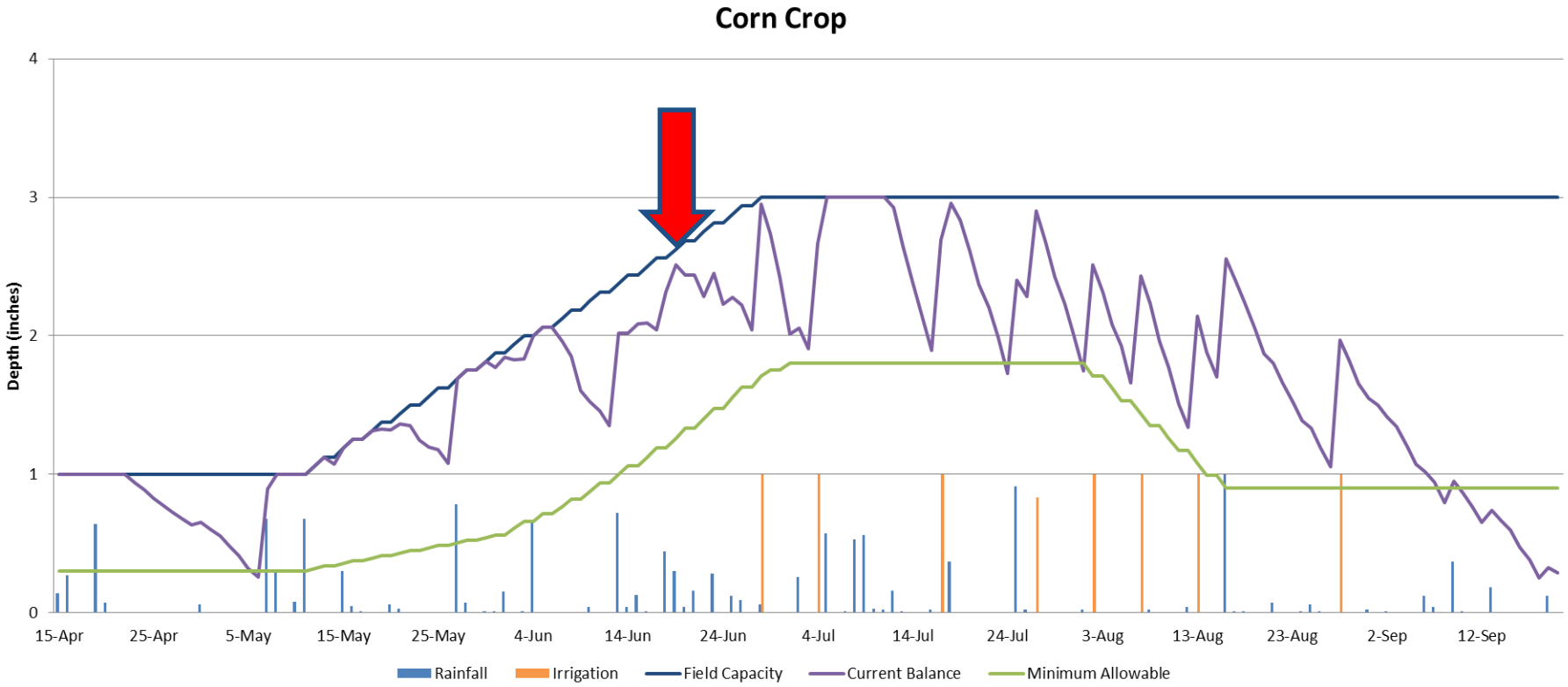
| | | |

Irrigation Scheduling



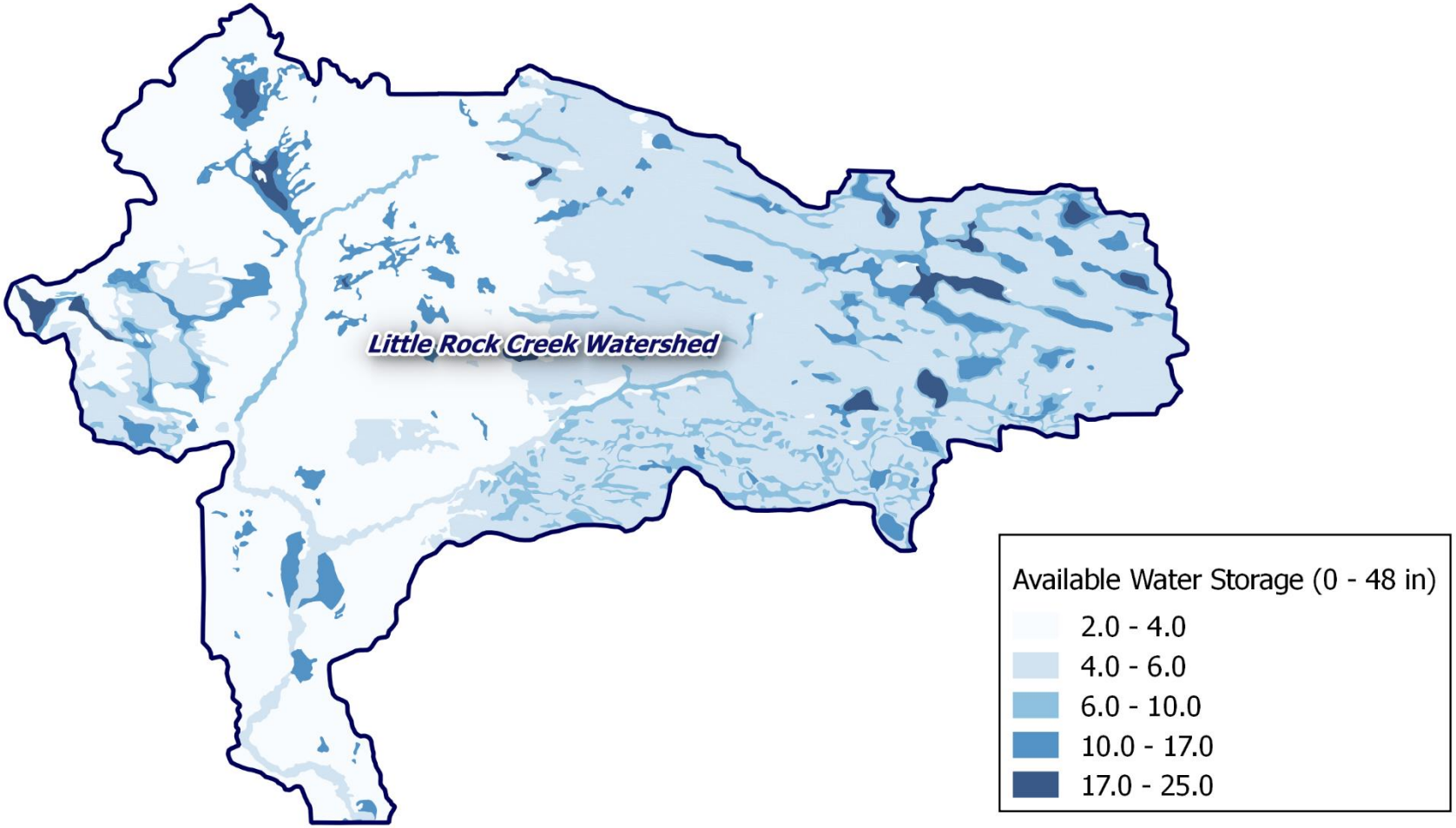
Irrigation Scheduling

Field Capacity



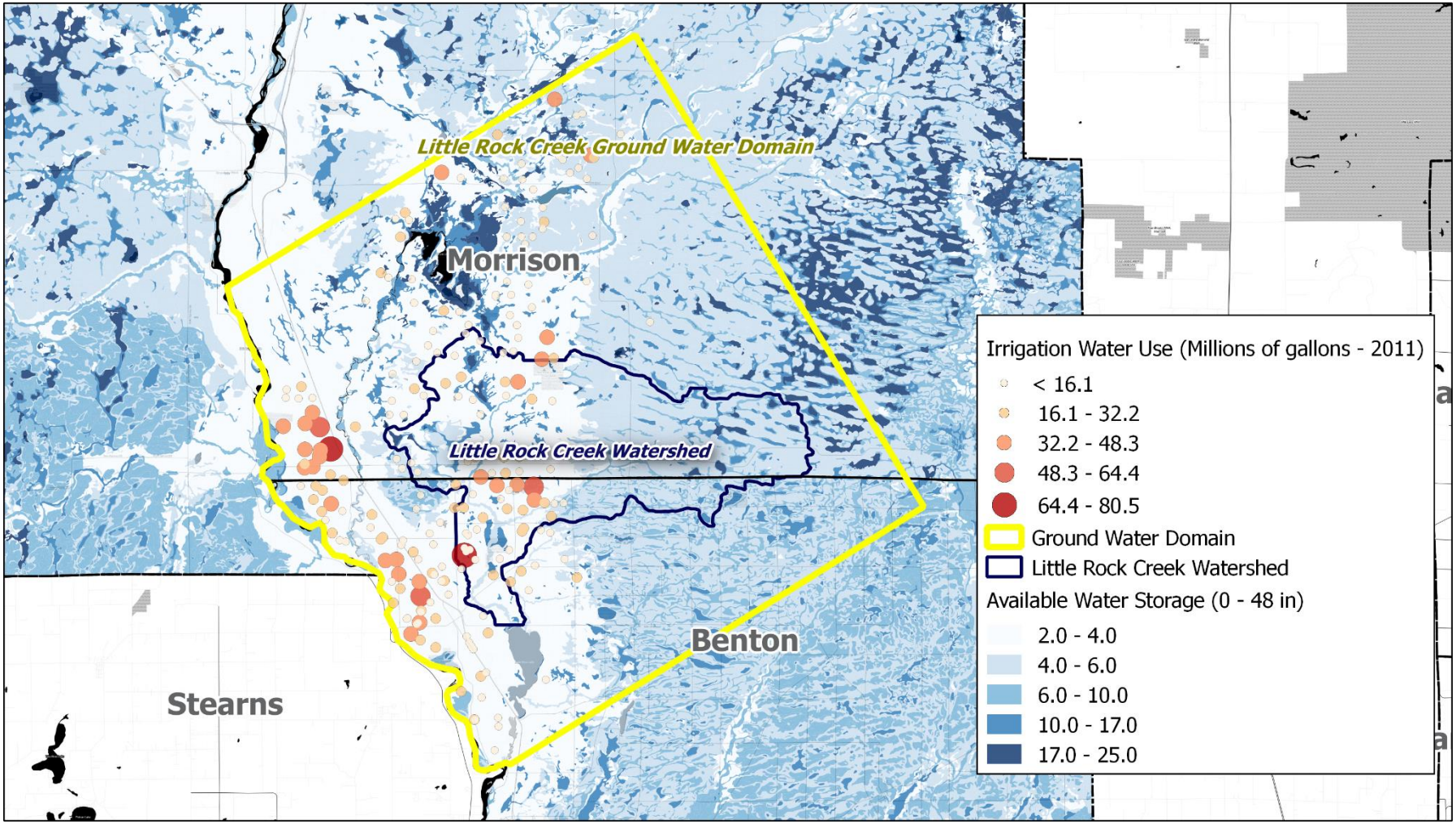
Irrigation Scheduling

Field Water Holding Capacity



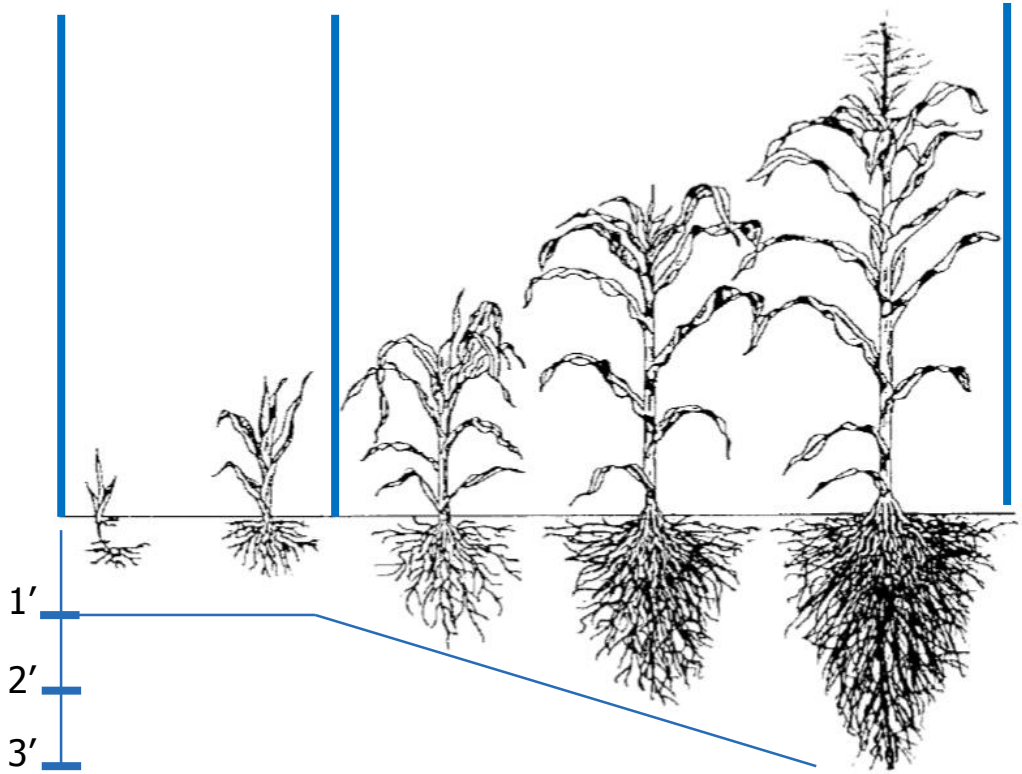
Irrigation Scheduling

Field Water Holding Capacity



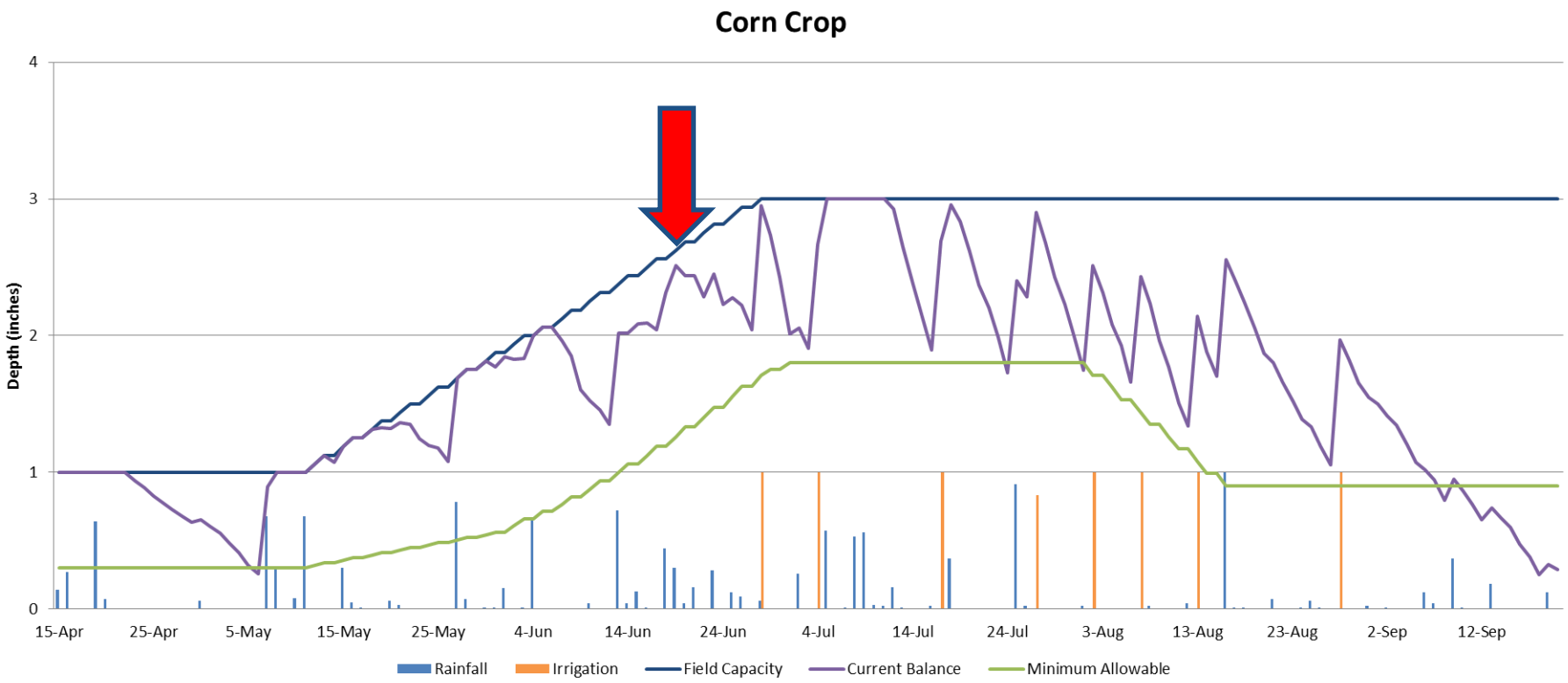
Irrigation Scheduling

Field Water Holding Capacity & Rooting Depth



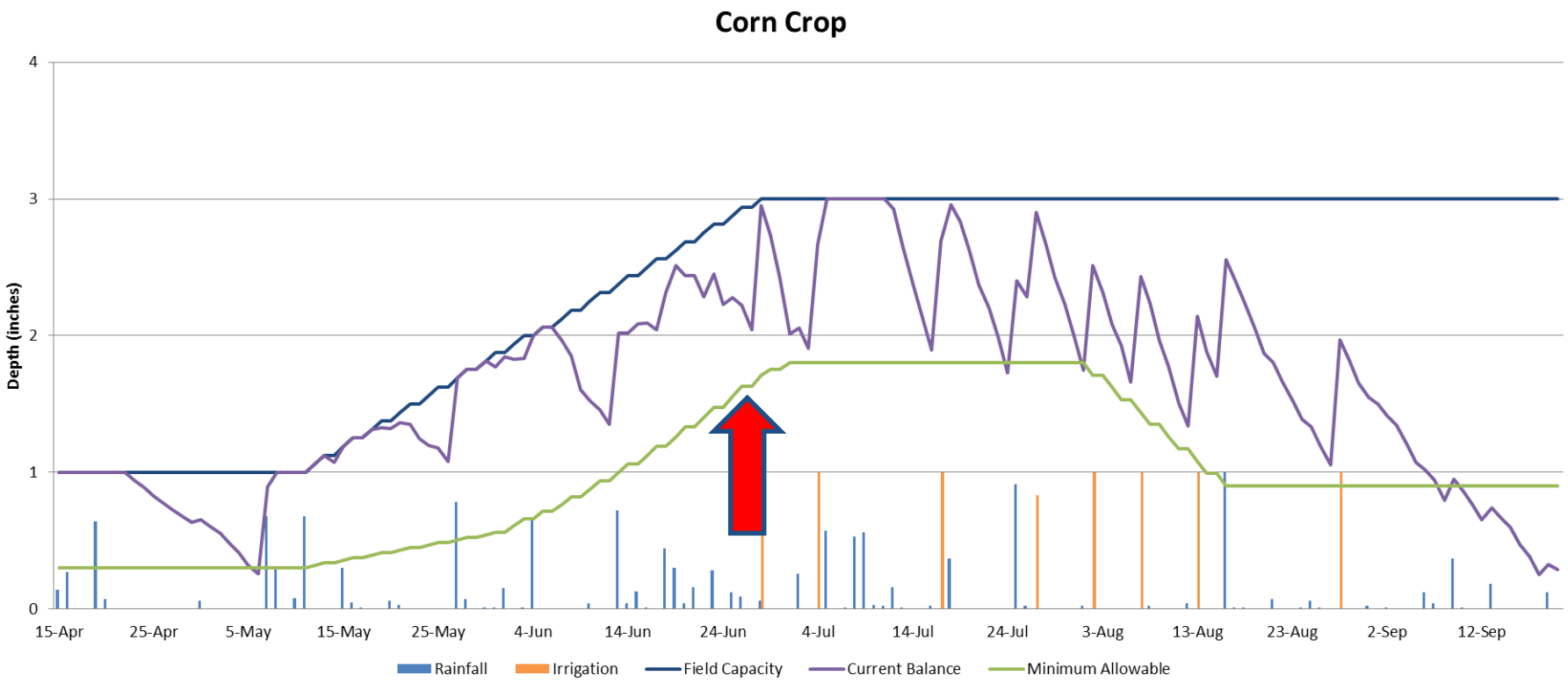
Irrigation Scheduling

Field Water Holding Capacity



Irrigation Scheduling

Minimum Allowable



Irrigation Scheduling

Calculation of Field Water Balance

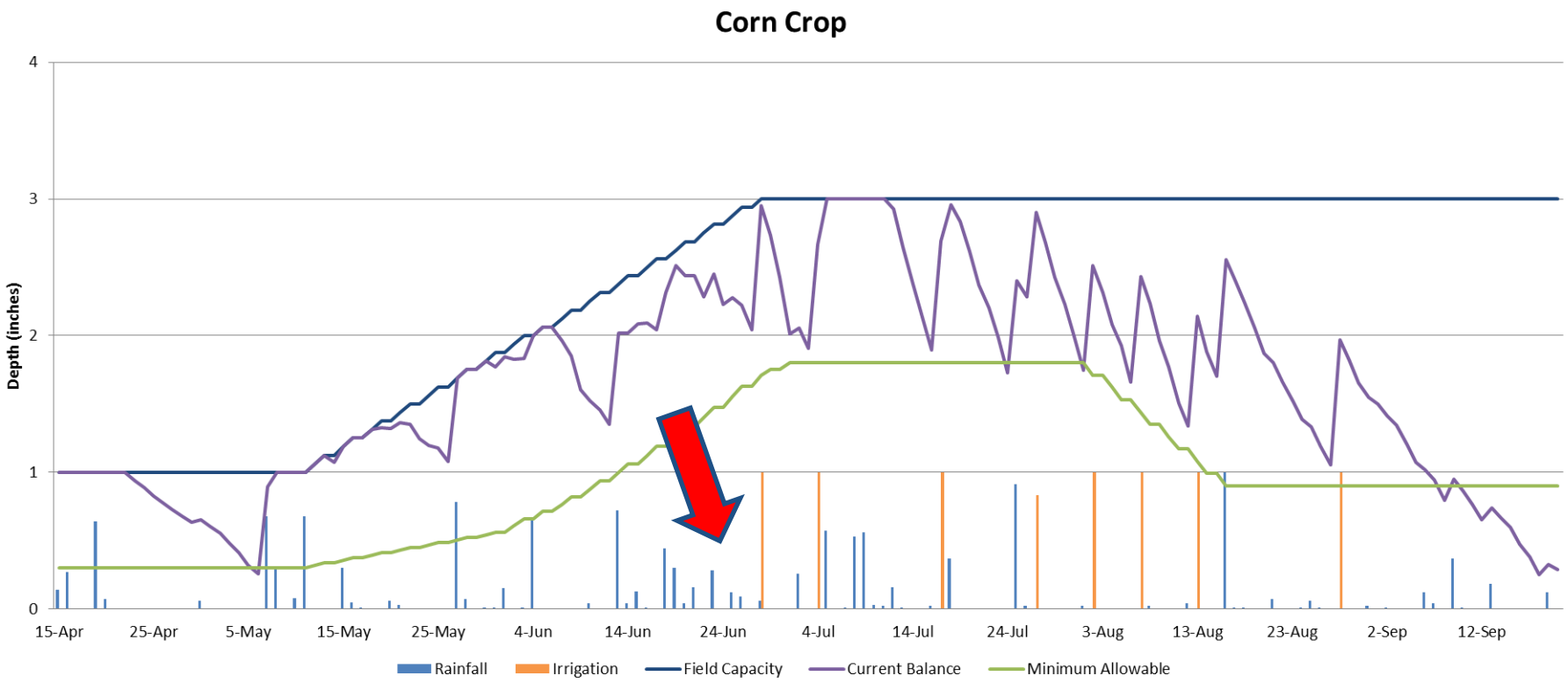
- Calculate Available Water

Available H_20 = Previous Day Available H_2O – ET + Rainfall + Irrigation

- Evapotranspiration (ET) Estimates:
  - weather stations ET0 * Kc Coefficient for crop stage * Ka Coefficient
- Rainfall: Radar or local rain gauges
- Irrigation: Entered by the farmer

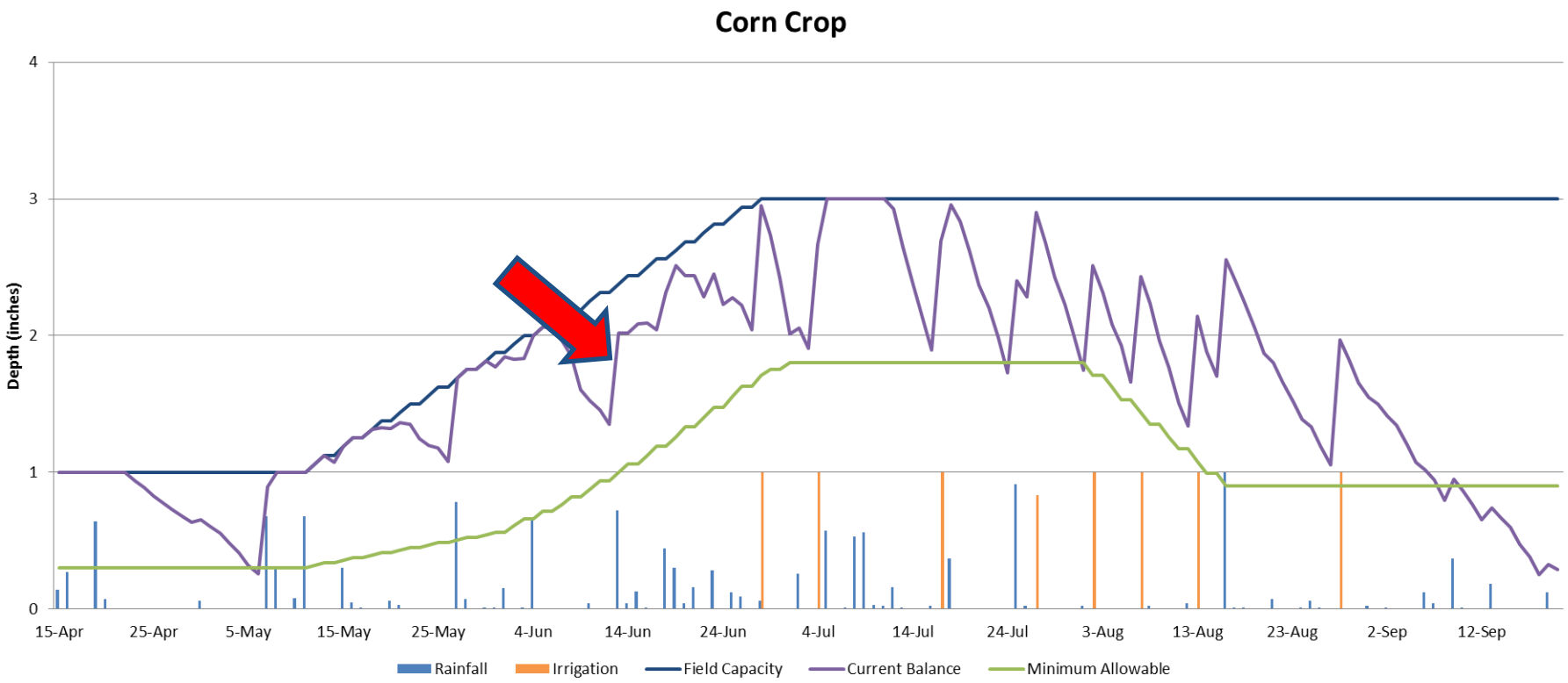
Irrigation Scheduling

Rainfall & Irrigation



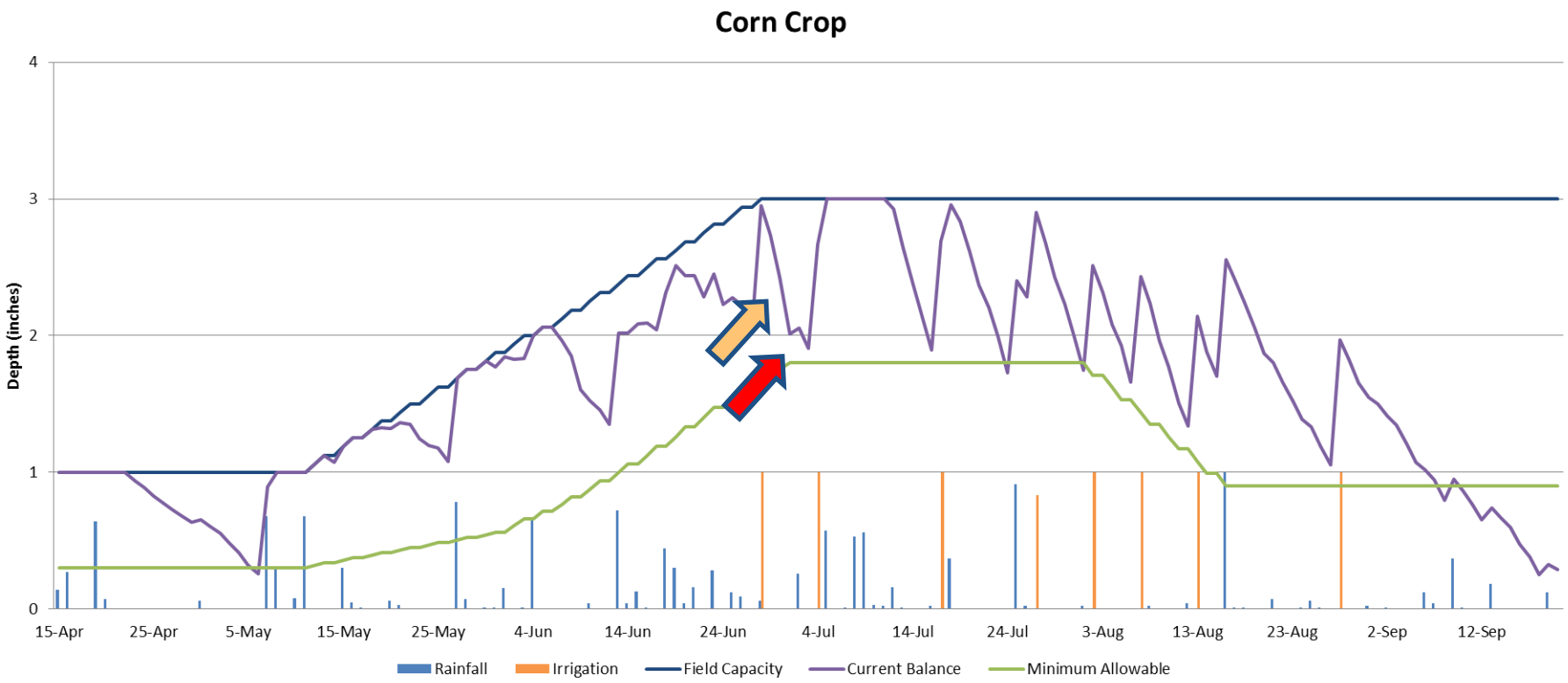
Irrigation Scheduling

Current Field Water Balance



Irrigation Scheduling

Recommending Irrigation



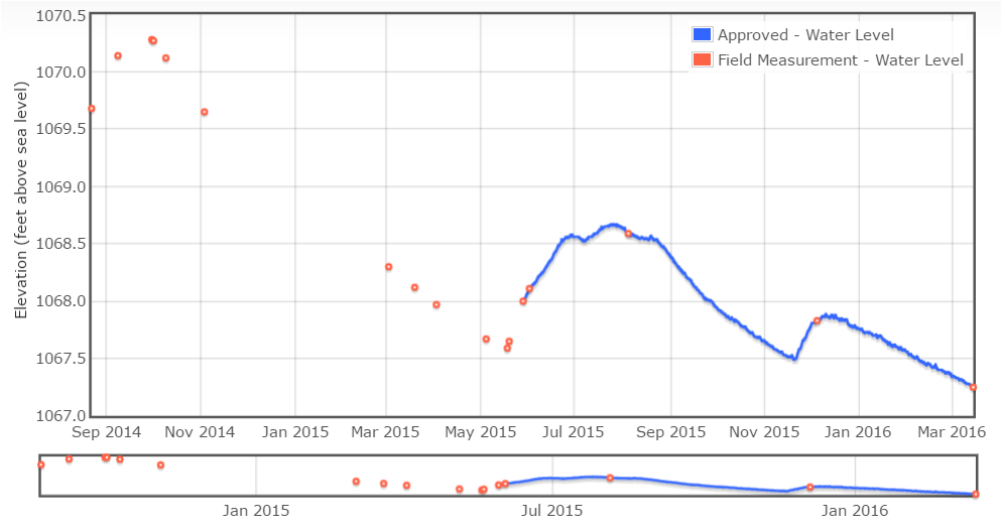
Monitoring Conservation Indicators

- Indicators of stream resource health

Photo Credit: Algae by David Strom; Dead Fish by Ben Brophy (*Some Rights Reserved*) | Dry Creek by usgs.gov

Monitoring Conservation Indicators

- Field data



Putting it together

- Understand precise crop needs
- Easy access to resource data

Software Stack

Leaflet

Apache

Flask

Bootstrap

PostGIS

Geospatial Objects for PostgreSQL

jQuery

C3.js D3-based reusable chart library

SQLAlchemy

Inputs to the Irrigation Management Assistant (IMA)

- User inputs
- Pre-processed
- Timeseries

Inputs to IMA

User Inputs

- Draw field boundary
- Field irrigation delivery capacity
- Crop type
- Planting Date
- Maturity Date *(optional override)*

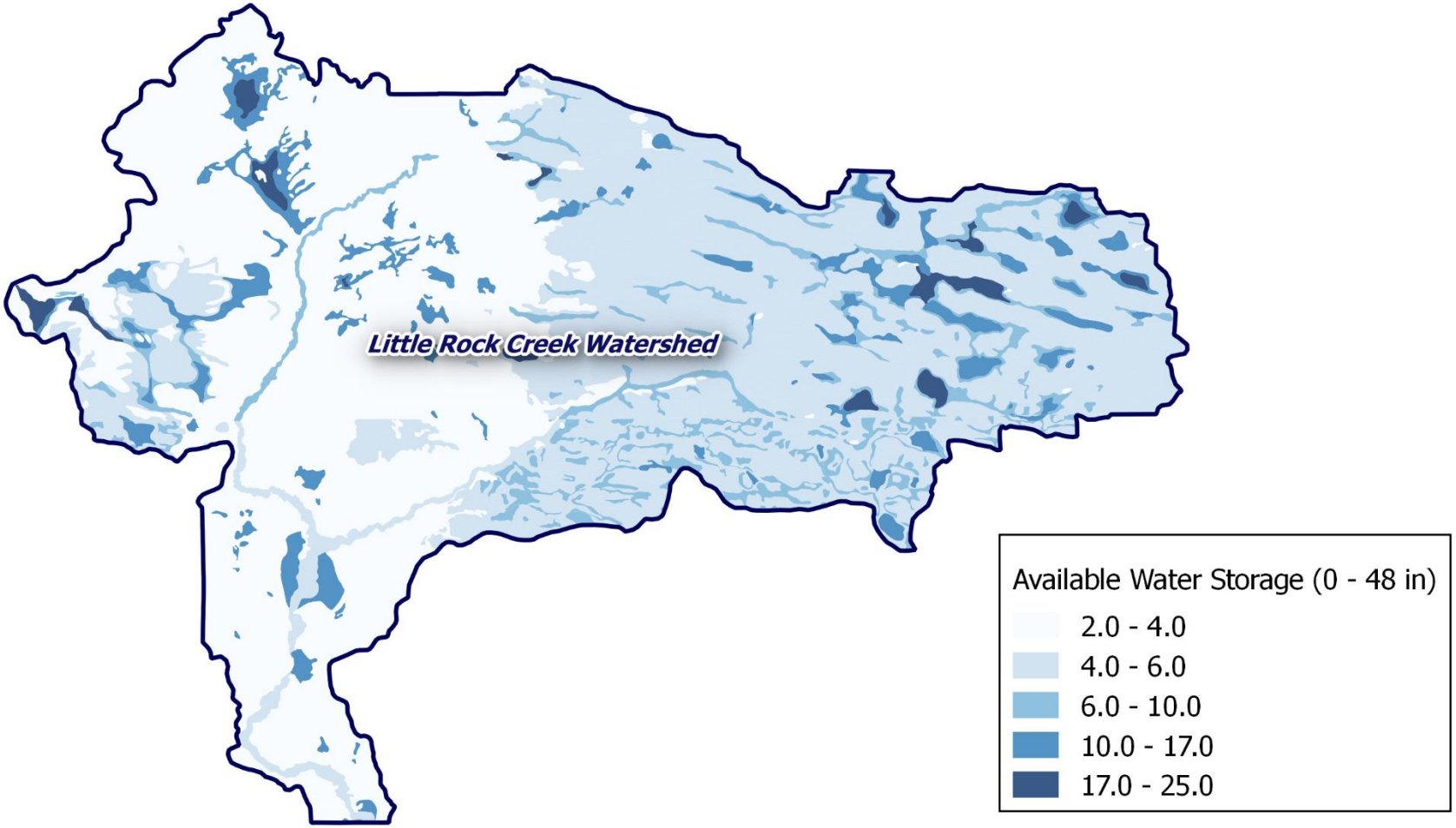
Inputs to IMA

Pre-processed: Available Water Storage (AWS)

- County based SSURGO
- SQL calculation of AWS
  - Map Unit -> Component -> Horizon
- Holes in data (past wetlands, data gaps)
- Filled via adjoin boundary length to distance weight

Inputs to IMA

AWS



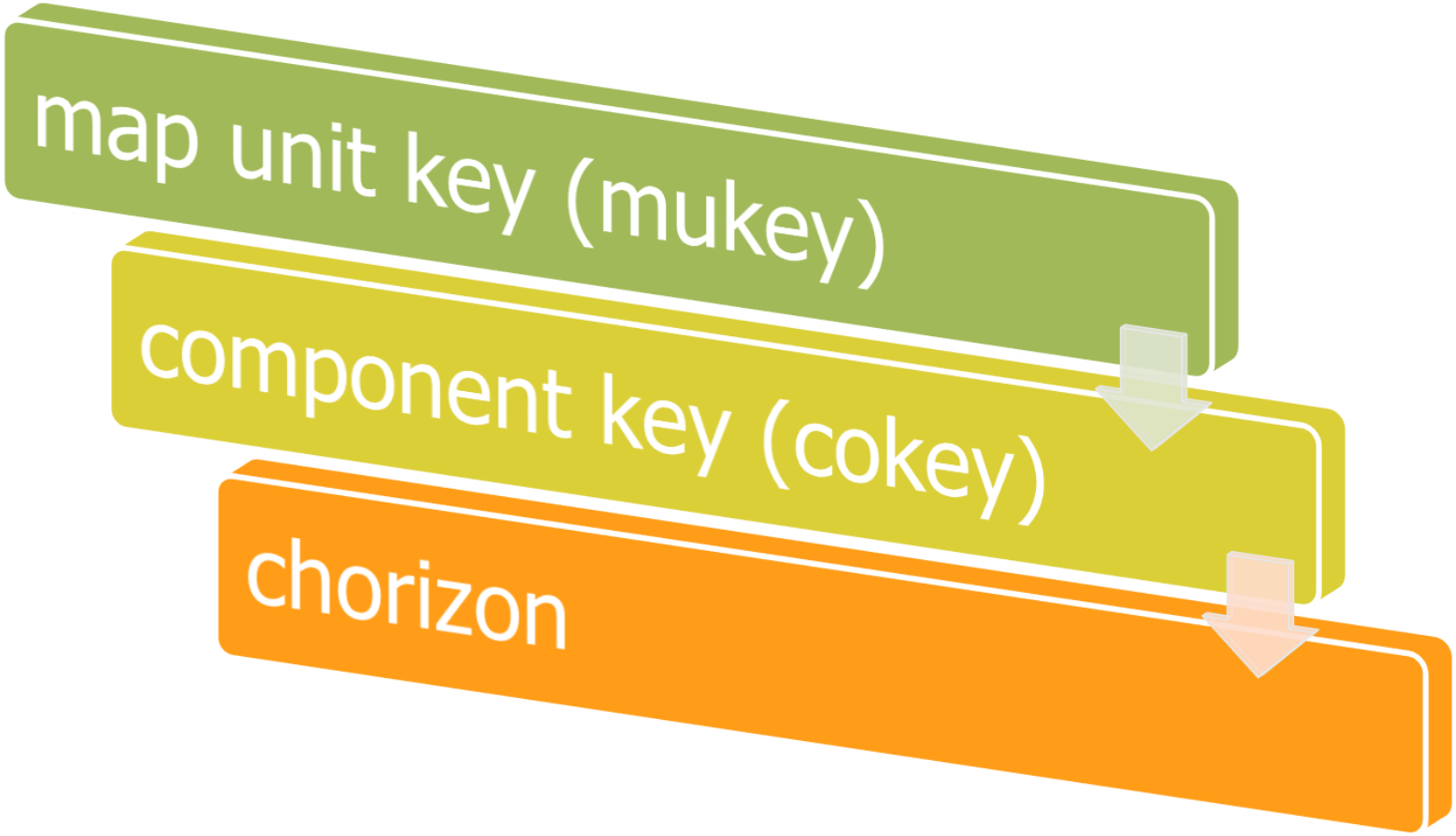
Inputs to IMA
AWS

- Shout out to California Soil Resource Lab for providing my starting point

```
-- create the new table with both geometry and attributes
CREATE TABLE yolo_whc AS
SELECT  ogc_fid, wkb_geometry AS wkb_geometry, b.mukey, b.comppct_weighted_whc_cm
FROM mapunit_poly
-- use LEFT JOIN to include non-soil polygons in the result set
-- alternatively use JOIN to ignore non-soil polygons
LEFT JOIN
        (
        SELECT mukey,
        -- note that weights from non-soil components must be removed
        -- otherwise, weighted mean values will be too low
        SUM(comppct_r * component_whc) / SUM(comppct_r) AS comppct_weighted_whc_cm
        FROM component
        JOIN
          (
          SELECT cokey, sum( (hzdepb_r - hzdept_r) * awc_r) AS component_whc,
          sum((hzdepb_r - hzdept_r)) AS depth
          FROM chorizon
          WHERE areasymbol = 'ca113'
          GROUP BY cokey
          ) AS a
        USING (cokey)
        WHERE component.areasymbol = 'ca113'
        -- filter out components that are missing soils data
        AND a.component_whc IS NOT NULL
```

Inputs to IMA

AWS



```sql
--1a)Get chorizon data by component splitting into zones 12,24,36,48,>48
select mukey,cokey,comppct_r,chkey,hzdept_r,hzdepb_r,top,bottom,awc_r,aws,
case
    when bottom <= 12 then 1
    when top <12 and bottom >12 then (12-top)/(bottom-top)
    else 0
end as z12p,
case
    when (top <= 12) and (bottom <= 24 and bottom >12) then ((bottom-top)-(12-top))/(bottom-top)
    when (top <= 12) and (bottom >24) then (24-12)/(bottom-top)
    when (top > 12 and top <=24) and (bottom > 24) then (24-top)/(bottom-top)
    when top > 12 and bottom <= 24 then 1
    else 0
end as z24p,
case
    when (top <= 24) and (bottom <= 36 and bottom >24) then ((bottom-top)-(24-top))/(bottom-top)
    when (top <= 24) and (bottom >36) then (36-24)/(bottom-top)
    when (top > 24 and top <=36) and (bottom > 36) then (36-top)/(bottom-top)
    when top > 24 and bottom <= 36 then 1
    else 0
end as z36p,
case
    when (top <= 36) and (bottom <= 48 and bottom >36) then ((bottom-top)-(36-top))/(bottom-top)
    when (top <= 36) and (bottom >48) then (48-36)/(bottom-top)
    when (top > 36 and top <=48) and (bottom > 48) then (48-top)/(bottom-top)
    when top > 36 and bottom <= 48 then 1
    else 0
end as z48p,
case
    when (top <= 48) and (bottom <= 1000 and bottom >48) then ((bottom-top)-(48-top))/(bottom-top)
    when (top <= 48) and (bottom >1000) then (1000-48)/(bottom-top)
    when (top > 48 and top <=1000) and (bottom > 1000) then (1000-top)/(bottom-top)
    when top > 48 and bottom <= 1000 then 1
    else 0
end as z1000p
from(
select mukey,cokey,comppct_r,chkey,hzdept_r,hzdepb_r,cm2in(hzdept_r) as top,cm2in(hzdepb_r) as bottom,awc_r, (hzdepb_r-hzdept_r)*awc_r as aws
from chorizon
JOIN component USING(cokey)
--where mukey='753680'
ORDER BY cokey,hzdept_r
) foo;
```

RESPEC

```sql
--2a)Group by component key (cokey)
select mukey,cokey,avg(comppct_r) comppct_r,
sum(aws*z12p) as z12_aws_co,
sum(aws*z24p) as z24_aws_co,
sum(aws*z36p) as z36_aws_co,
sum(aws*z48p) as z48_aws_co,
sum(aws*z1000p) as z1000_aws_co
from(
select mukey,cokey,comppct_r,chkey,hzdept_r,hzdepb_r,top,bottom,awc_r,aws,
case
    when bottom <= 12 then 1
    when top <12 and bottom >12 then (12-top)/(bottom-top)
    else 0
end as z12p,
case
    when (top <= 12) and (bottom <= 24 and bottom >12) then ((bottom-top)-(12-top))/(bottom-top)
    when (top <= 12) and (bottom >24) then (24-12)/(bottom-top)
    when (top > 12 and top <=24) and (bottom > 24) then (24-top)/(bottom-top)
    when top > 12 and bottom <= 24 then 1
    else 0
end as z24p,
case
    when (top <= 24) and (bottom <= 36 and bottom >24) then ((bottom-top)-(24-top))/(bottom-top)
    when (top <= 24) and (bottom >36) then (36-24)/(bottom-top)
    when (top > 24 and top <=36) and (bottom > 36) then (36-top)/(bottom-top)
    when top > 24 and bottom <= 36 then 1
    else 0
end as z36p,
case
    when (top <= 36) and (bottom <= 48 and bottom >36) then ((bottom-top)-(36-top))/(bottom-top)
    when (top <= 36) and (bottom >48) then (48-36)/(bottom-top)
    when (top > 36 and top <=48) and (bottom > 48) then (48-top)/(bottom-top)
    when top > 36 and bottom <= 48 then 1
    else 0
end as z48p,
case
    when (top <= 48) and (bottom <= 1000 and bottom >48) then ((bottom-top)-(48-top))/(bottom-top)
    when (top <= 48) and (bottom >1000) then (1000-48)/(bottom-top)
    when (top > 48 and top <=1000) and (bottom > 1000) then (1000-top)/(bottom-top)
    when top > 48 and bottom <= 1000 then 1
    else 0
end as z1000p
from(
select mukey,cokey,comppct_r,chkey,hzdept_r,hzdepb_r,cm2in(hzdept_r) as top,cm2in(hzdepb_r) as bottom,awc_r, (hzdepb_r-hzdept_r)*awc_r as aws
from chorizon
JOIN component USING(cokey)
where mukey='753680'
ORDER BY cokey,hzdept_r
) foo
) foo2
group by mukey,cokey;
```

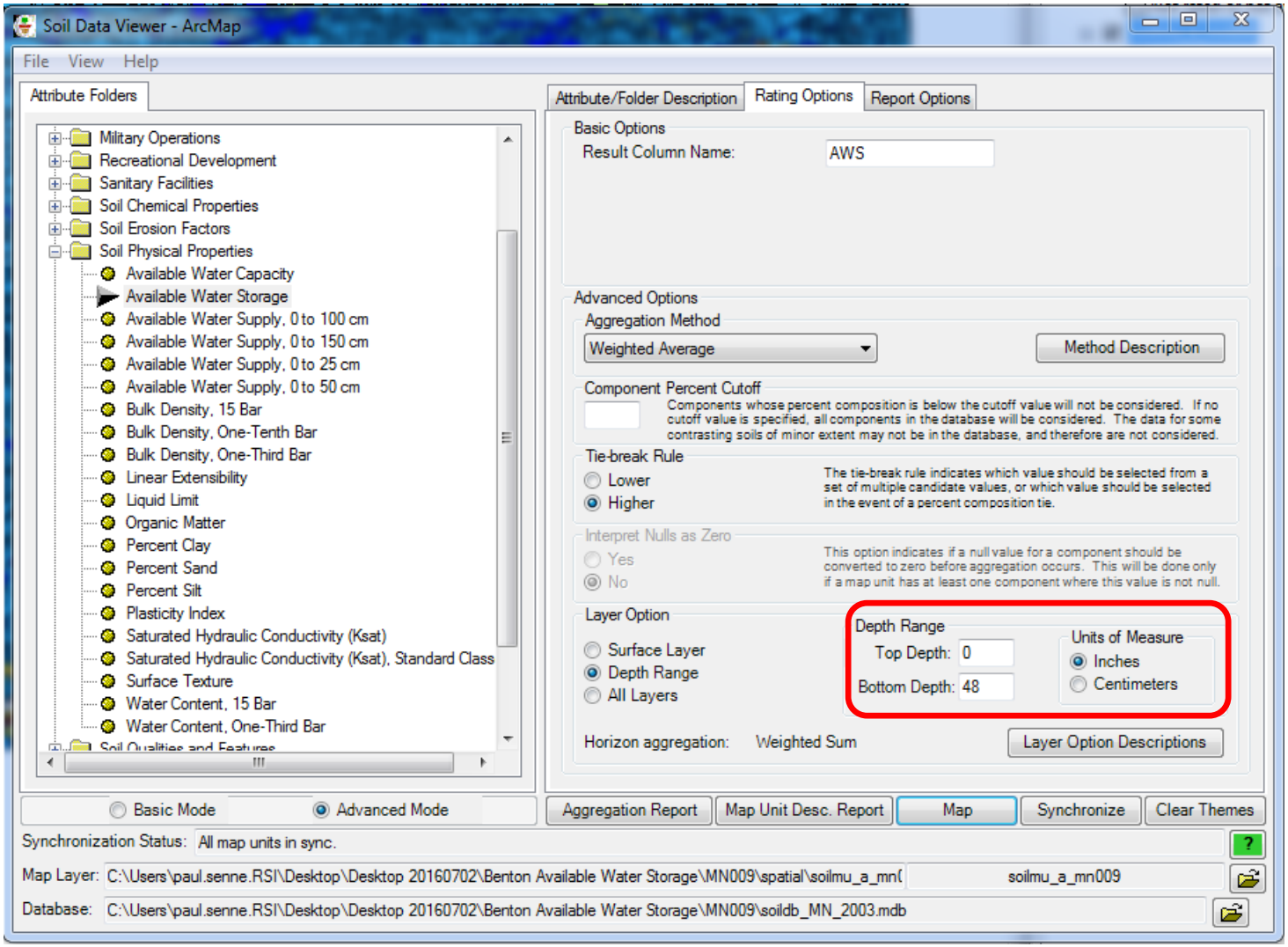
```
from(
select mukey,cokey,avg(comppct_r) comppct_r,
sum(aws*z12p) as z12_aws_co,
sum(aws*z24p) as z24_aws_co,
sum(aws*z36p) as z36_aws_co,
sum(aws*z48p) as z48_aws_co,
sum(aws*z1000p) as z1000_aws_co
from(
select mukey,cokey,comppct_r,chkey,hzdept_r,hzdepb_r,top,bottom,awc_r,cm2in(aws::numeric) as aws,
case
    when bottom <= 12 then 1
    when top <12 and bottom >12 then (12-top)/(bottom-top)
    else 0
end as z12p,
case
    when (top <= 1
    when (top <= 1
    when (top > 12
    when top > 12
    else 0
end as z24p,
case
    when (top <= 24) and (bottom <= 36 and bottom >24) then ((bottom-top)-(24-top))/(bottom-top)
    when (top <= 24) and (bottom >36) then (36-24)/(bottom-top)
    when (top > 24 and top <=36) and (bottom > 36) then (36-top)/(bottom-top)
    when top > 24 and bottom <= 36 then 1
    else 0
end as z36p,
case
    when (top <= 36) and (bottom <= 48 and bottom >36) then ((bottom-top)-(36-top))/(bottom-top)
    when (top <= 36) and (bottom >48) then (48-36)/(bottom-top)
    when (top > 3
    when top > 36
    else 0
end as z48p,
case
    when (top <= 48) and (bottom <= 1000 and bottom >48) then ((bottom-top)-(48-top))/(bottom-top)
    when (top <= 48) and (bottom >1000) then (1000-48)/(bottom-top)
    when (top > 48 and top <=1000) and (bottom > 1000) then (1000-top)/(bottom-top)
    when top > 48 and bottom <= 1000 then 1
    else 0
end as z1000p
from(
select mukey,cokey,comppct_r,chkey,hzdept_r,hzdepb_r,cm2in(hzdept_r) as top,cm2in(hzdepb_r) as bottom,awc_r, (hzdepb_r-hzdept_r)*awc_r as aws
from chorizon
JOIN component USING(cokey)
--where mukey='753680'
ORDER BY cokey,hzdept_r
) foo
) foo2
group by mukey,cokey
)foo3
```

Join back to map units

Interpolate Missing Data From Adjacent

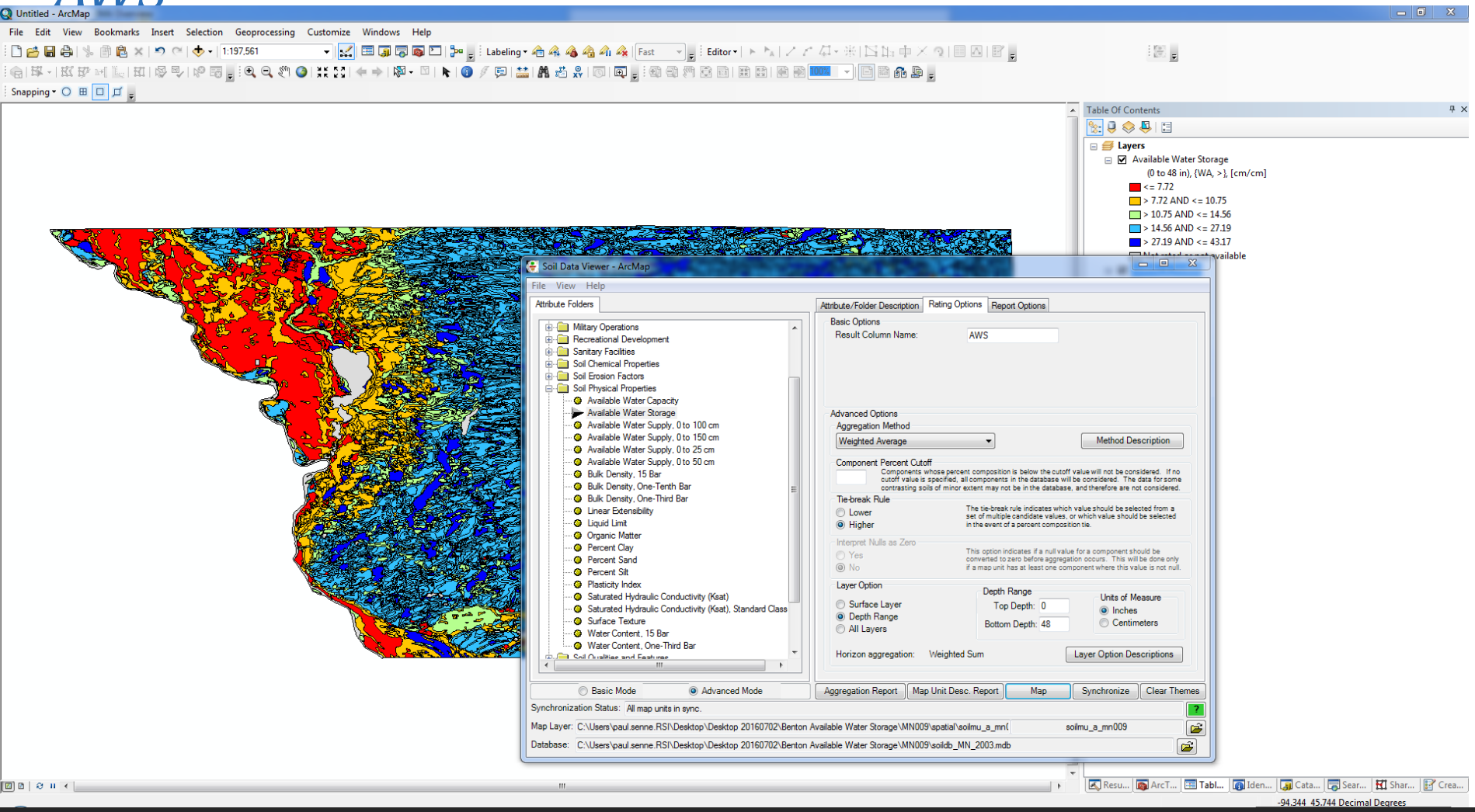
Inputs to IMA

AWS



Inputs to IMA

AWS



Inputs to IMA

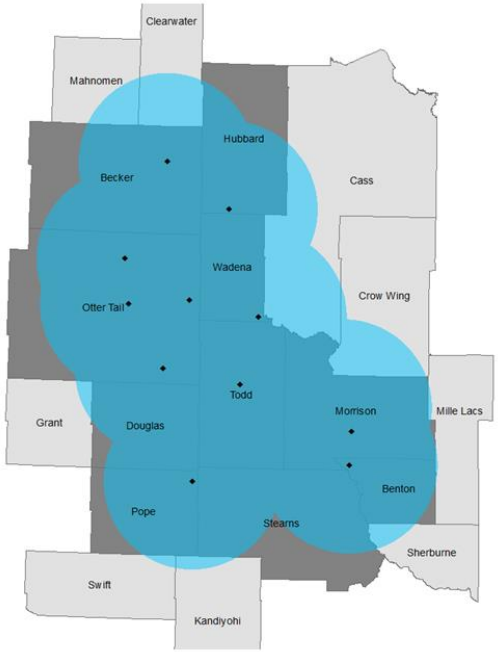
Time Series Datasets

- *Weather*
- *Rainfall*
- *Irrigation*

Inputs to IMA

Weather Stations and Zones

- MN Agricultural Weather Network
- Delaunay triangles to determine zones for each station
- Data logger access
- File Foreign Data Wrappers
- Data import from data source



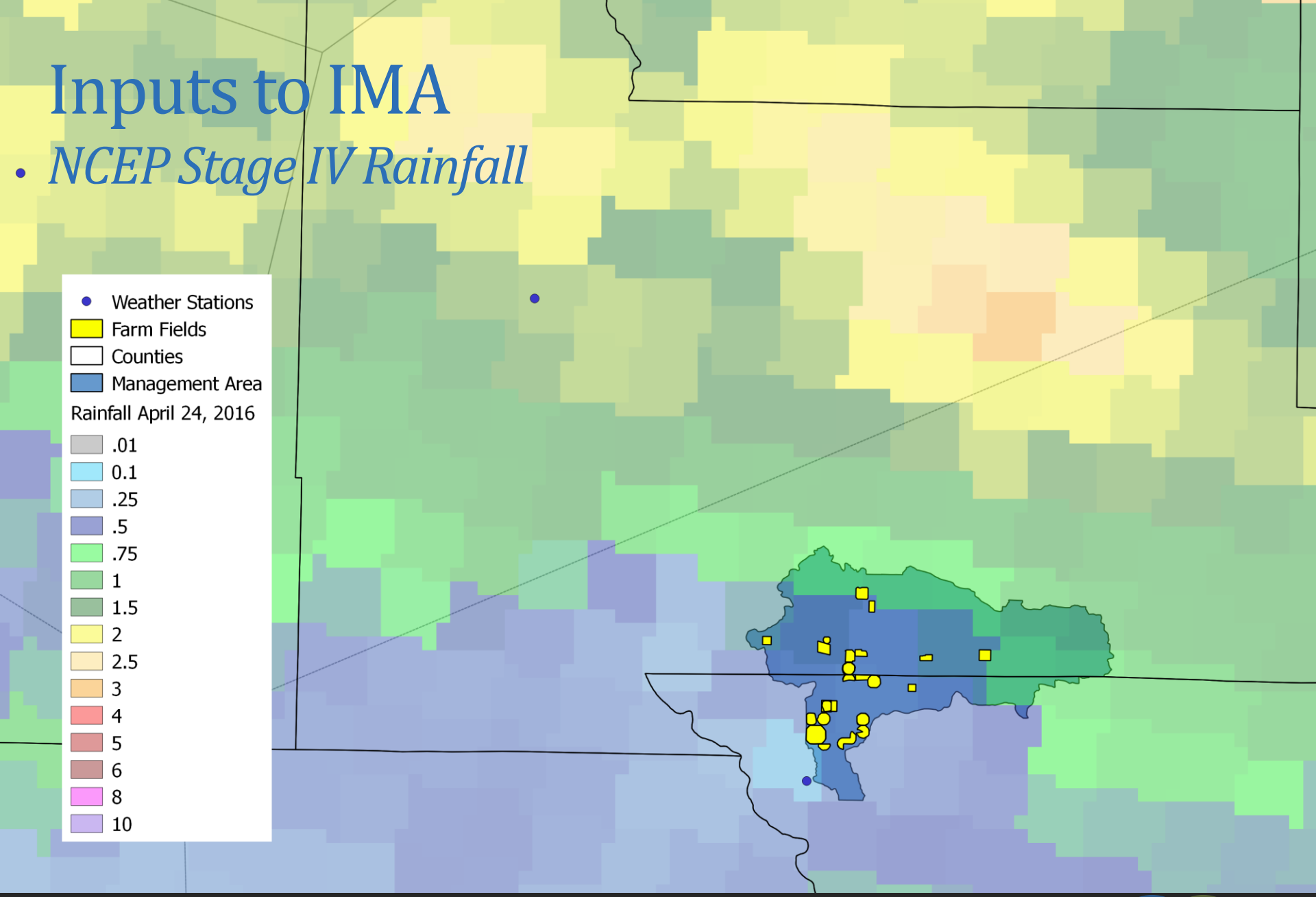
Inputs to IMA

NCEP Stage IV Rainfall

- Python script to pull .gz GRIB file by the hour
- Weather and Climate Toolkit (WCT) conversion to tiff
- GDAL & Python gdal_calc for transforming & aggregating to daily
- Raster2pgsql push into PostGIS
- Triggers calculations

Inputs to IMA

- *NCEP Stage IV Rainfall*



-- Calculate field rainfall each day – code from plpython function in postgresql

```
WITH
field AS (
        SELECT gfb."id", planting."id" as planting_id, geom
        FROM "%(schema)s".geom_field_boundaries gfb
        LEFT JOIN "%(schema)s".planting ON gfb."id" = planting.geometry_field_boundaries_id
        WHERE gfb."id" IN(planting.geometry_field_boundaries_id) AND planting."id" IN(%(planting_ids)s)
),

 aws_stats AS
(
        SELECT  id, planting_id, (stats).*, alternate
        FROM (
                SELECT b.id, b.planting_id, ST_SummaryStats(ST_Clip(a.rast,1,b.geom,1)) As stats, st_value(a.rast,
st_centroid(b.geom)) as alternate
                FROM (SELECT * FROM statewide.rast_ncep_stage_iv_daily_inches WHERE datestamp =
'%(cdate)s') as a
                INNER JOIN field b
                ON ST_Intersects(b.geom,a.rast)
        ) As foo
)

UPDATE "%(schema)s".field_irrigation_schedule fis
SET rainfall_ncep_in = COALESCE(avg_pval,0),
    et_total_in = (fis.eto_in * fis.kc * fis.ka)
FROM (
        SELECT planting_id, COALESCE((SUM(mean*count)/SUM(count)),avg(alternate),0) As avg_pval
        FROM aws_stats
        GROUP BY id,planting_id
        ORDER BY id
)foo
WHERE fis.planting_id=foo.planting_id AND fis.record_date = '%(cdate)s' AND fis.planting_id IN(%(planting_ids)s);
```

Inputs to IMA

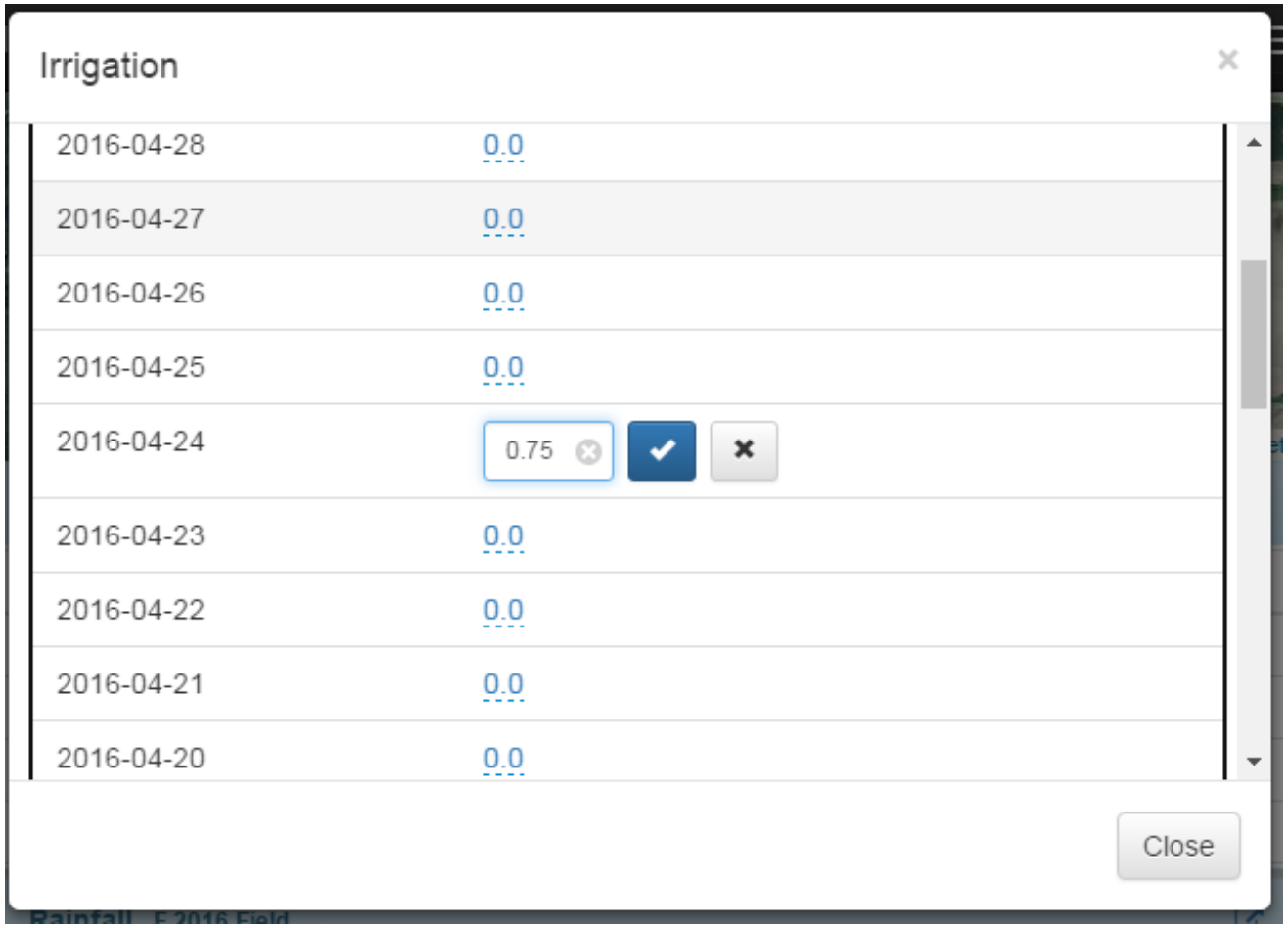
Rainfall (Override)

Rainfall

Date	NOAA Rainfall (Inches)	Rainfall Override (Inches)
2016-07-12	0.0	None
2016-07-11	4.56	None
2016-07-10	0.78	0.5
2016-07-09	0.0	None
2016-07-08	0.16	None
2016-07-07	0.21	None
2016-07-06	0.0	None
2016-07-05	0.22	None

Inputs to IMA

Irrigation



Inputs to IMA

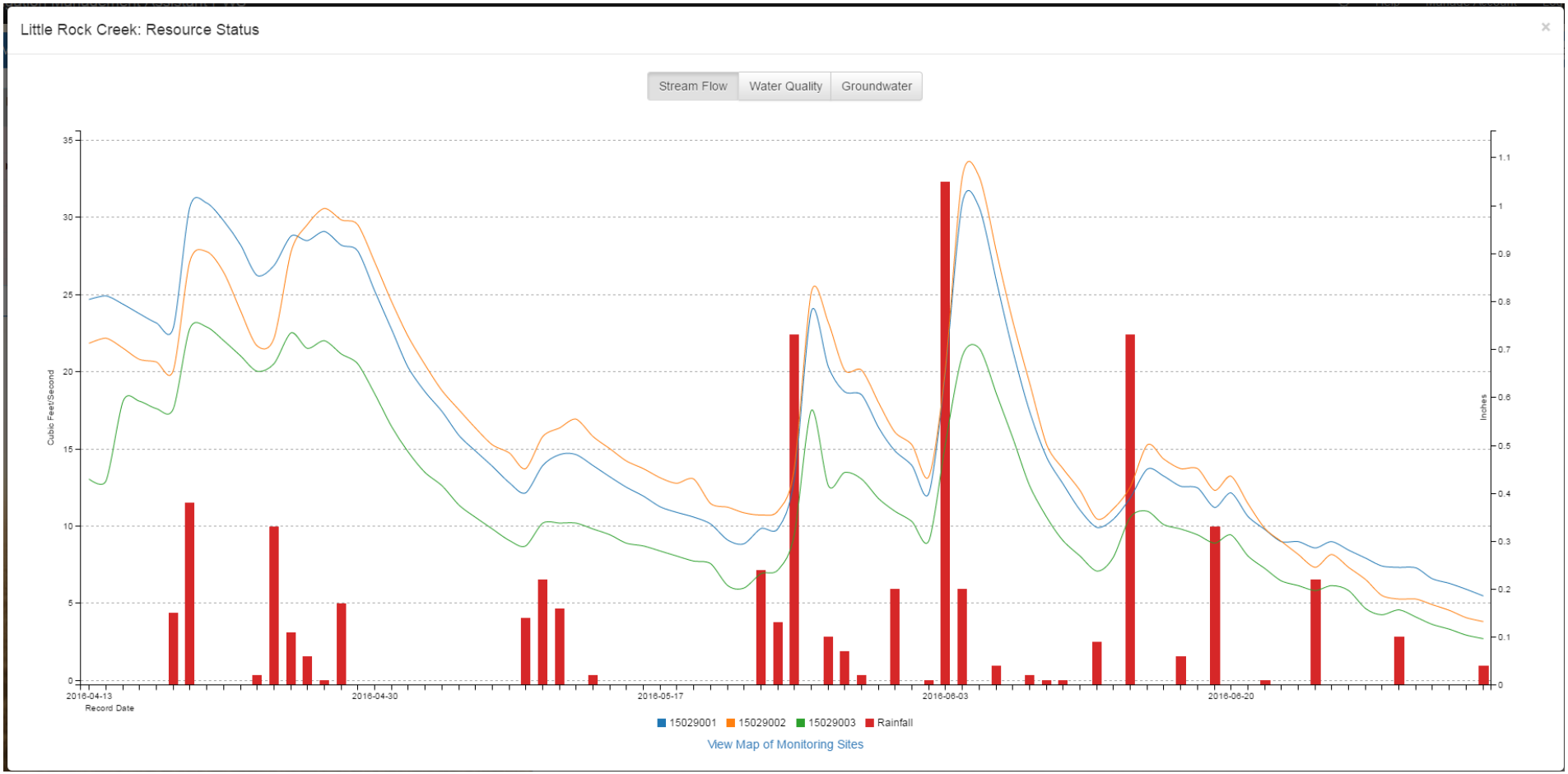
Soil Moisture Re-Balance

Soil Moisture

Date	Calc. Inches	User Re-balance
2016-07-12	2.28	None
2016-07-11	2.28	None
2016-07-10	1.53	None
2016-07-09	0.87	None
2016-07-08	1.03	None
2016-07-07	1.01	None
2016-07-06	0.95	None
2016-07-05	1.12	None
2016-07-04	0.83	1.0
2016-07-03	0.96	None

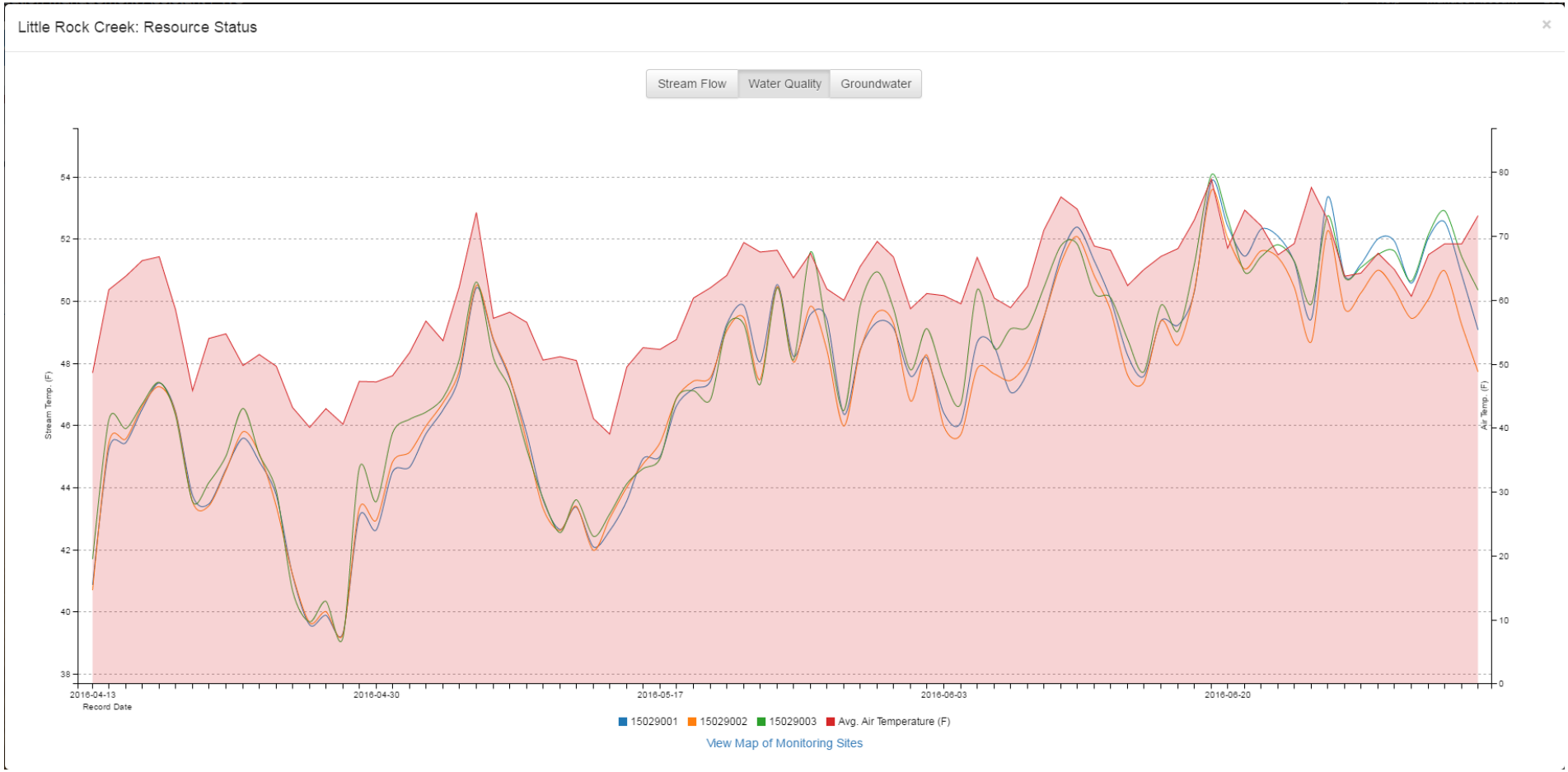
Resource Monitoring

Stream Flow



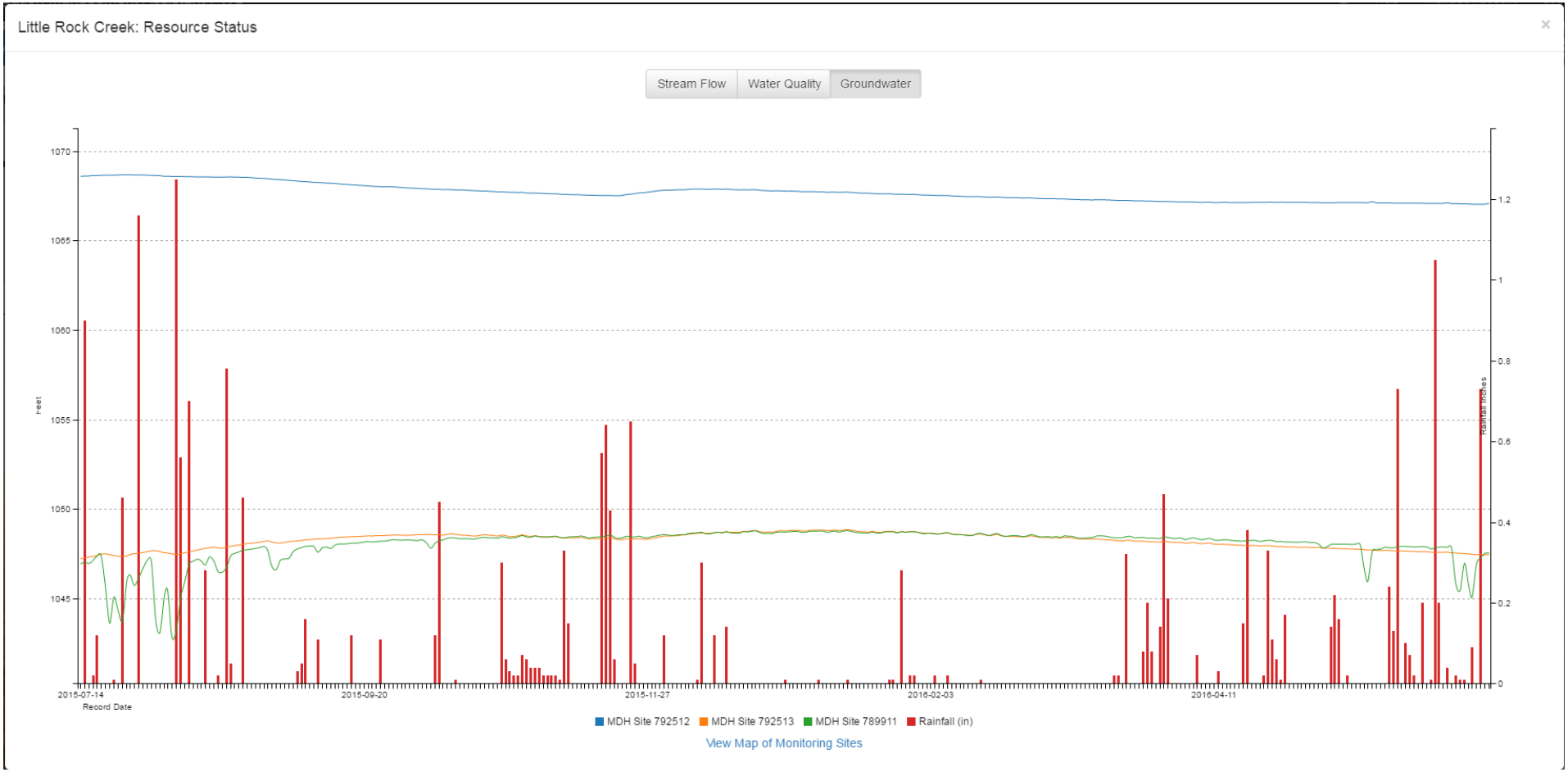
Resource Monitoring

Stream Temperature



Resource Monitoring

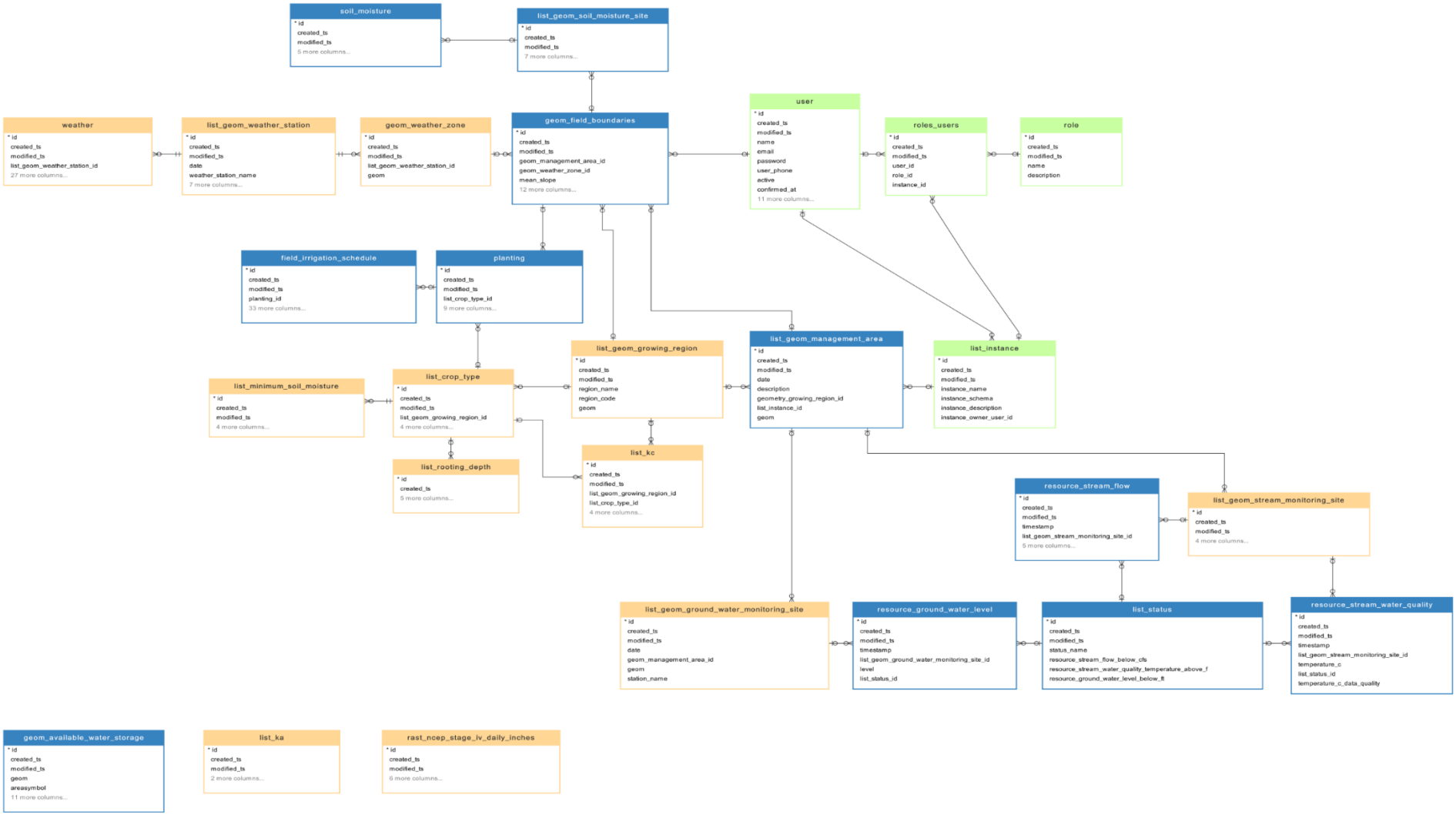
Groundwater Level



Data model

- Scalability
  - Multi-schema approach
  - Common tables and datasets for entire system are kept once
  - Data for an “instance” is kept separate
  - Allow multiple management areas (watersheds) in an instance
- Some scalability is not fully built, but the data model is built around this
  - Re-usable SQLAlchemy model for “instances”
  - Ideal future would generate new instances on demand and take inputs for setup

Data model


soil_moisture
* id
created_ts
modified_ts
5 more columns...
list_geom_soil_moisture_site
* id
created_ts
modified_ts
7 more columns...
weather
* id
created_ts
modified_ts
list_geom_weather_station_id
27 more columns...
list_geom_weather_station
* id
created_ts
modified_ts
date
weather_station_name
7 more columns...
geom_weather_zone
* id
created_ts
modified_ts
list_geom_weather_station_id
geom
geom_field_boundaries
* id
created_ts
modified_ts
geom_management_area_id
geom_weather_zone_id
mean_slope
12 more columns...
user
* id
created_ts
modified_ts
name
email
password
user_phone
active
confirmed_at
11 more columns...
roles_users
* id
created_ts
modified_ts
user_id
role_id
instance_id
role
* id
created_ts
modified_ts
name
description
field_irrigation_schedule
* id
created_ts
modified_ts
planting_id
33 more columns...
planting
* id
created_ts
modified_ts
list_crop_type_id
9 more columns...
list_geom_growing_region
* id
created_ts
modified_ts
region_name
region_code
geom
list_geom_management_area
* id
created_ts
modified_ts
date
description
geometry_growing_region_id
list_instance_id
geom
list_instance
* id
created_ts
modified_ts
instance_name
instance_schema
instance_description
instance_owner_user_id
list_minimum_soil_moisture
* id
created_ts
modified_ts
4 more columns...
list_crop_type
* id
created_ts
modified_ts
list_geom_growing_region_id
4 more columns...
list_rooting_depth
* id
created_ts
5 more columns...
list_kc
* id
created_ts
modified_ts
list_geom_growing_region_id
list_crop_type_id
4 more columns...
resource_stream_flow
* id
created_ts
modified_ts
timestamp
list_geom_stream_monitoring_site_id
5 more columns...
list_geom_stream_monitoring_site
* id
created_ts
modified_ts
4 more columns...
list_geom_ground_water_monitoring_site
* id
created_ts
modified_ts
date
geom_management_area_id
geom
station_name
resource_ground_water_level
* id
created_ts
modified_ts
timestamp
list_geom_ground_water_monitoring_site_id
level
list_status_id
list_status
* id
created_ts
modified_ts
status_name
resource_stream_flow_below_cfs
resource_stream_water_quality_temperature_above_f
resource_ground_water_level_below_ft
resource_stream_water_quality
* id
created_ts
modified_ts
timestamp
list_geom_stream_monitoring_site_id
temperature_c
list_status_id
temperature_c_data_quality
geom_available_water_storage
* id
created_ts
modified_ts
geom
areasymbol
11 more columns...
list_ka
* id
created_ts
modified_ts
2 more columns...
rast_ncep_stage_iv_daily_inches
* id
created_ts
modified_ts
6 more columns...

Data Model

Key Triggers

- Create/Update Field
  - Calculate AWS, Area, Management Zone, Weather zone, etc.
- Create Planting
  - Lookup growing season length
  - Insert rows for each day of growing season length, pre-populating what is known.

- Calculate Daily irrigation Recommendation
  - For all fields, run each day when new rainfall is pushed
  - 22 fields – processed in specific sequence
  - Includes referencing past days' values as well as projected
- Update Field Irrigation Recommendation
  - Loop through daily calculations starting on date of a user change (rainfall, irrigation)

DEMO

Photo Credit: Irrigation at Dawn by Chris Happel (Some Rights Reserved)

Thank you!

paul.senne@respec.com

Photo Credit: Irrigation at Dawn by Chris Happel (*Some Rights Reserved*)

RESPEC