Phase 1: Proof of Concept toward the development of software tools described in EWD's SDLF Grant Request

Report generated at 2021-11-02 16:00:28.

Contents

1 Introduction

Phase 1 deliverables are separated into two parts (1A and 1B) and support the development of longer-term, Phase 2 deliverables that have yet to be fully scoped. Hence, the Phase 1 deliverables described in this terse report serve as a proof-of-concept for Phase 2 work. Phase 1A concerns data extraction, transformation, and loading of data from online platforms including data.ca.gov and the DIGI Connect API. Phase 1B concerns the representation of timeseries data in an interactive, web-dashboard framework. Phase 1A and 1B can be thought of as covering essential "backend" (calculations made by a computer) and "frontend" (what the user sees) components of a website.

Phase 1A deliverables include software that:

- extracts 15-minute interval recharge well data from data.ca.gov
- extracts data from DIGI Connect Sensors (such as those used by EWD)
- transforms these extracted data to informative values including:
  - injected volume
  - extracted volume
  - operating well pressure and/or hydraulic head

  - injection system water pressure

- loads these data to an SQLite database

Phase 1B deliverables include a non-password-protected (public) data dashboard with:

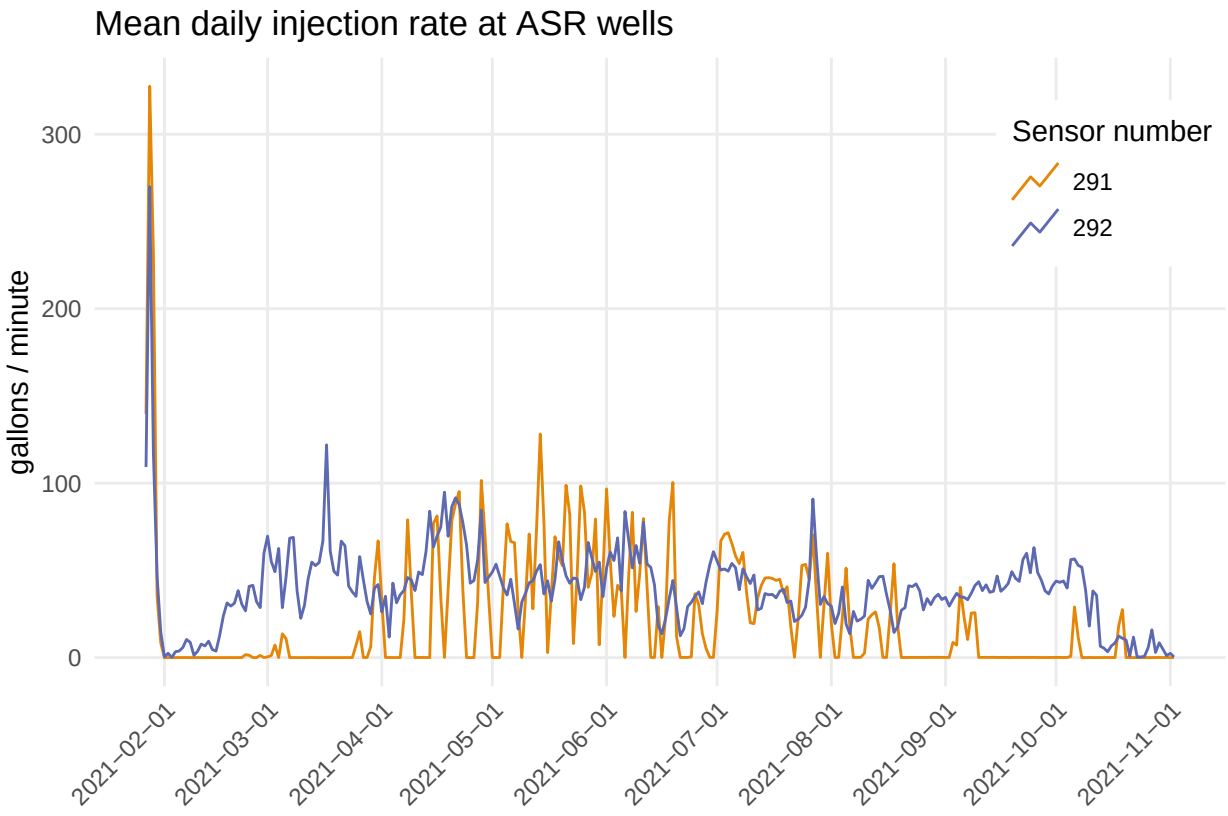
- interactive timeseries data visualization using "dummy data"
- example interactive time history "summaries" (e.g., cumulative volume injected extracted in the past month) that inform ASR management

2 Phase 1A deliverables

2.1 Extract recharge data from data.ca.gov

Here we demonstrate the extraction of 15-minute interval recharge well data from data.ca.gov. Data is extracted for sensors 291 and 292.

```
library(tidyverse)
library(here)
library(fs)

# source CDEC API extract module
source(here("src/01_download.R"))

# print the resulting plot
p_flow
```




```
# view some of the most recent data
flow %>%
  select(station_id, sensor_number, date_time,
         value, units, everything()) %>%
  tail(10)
```

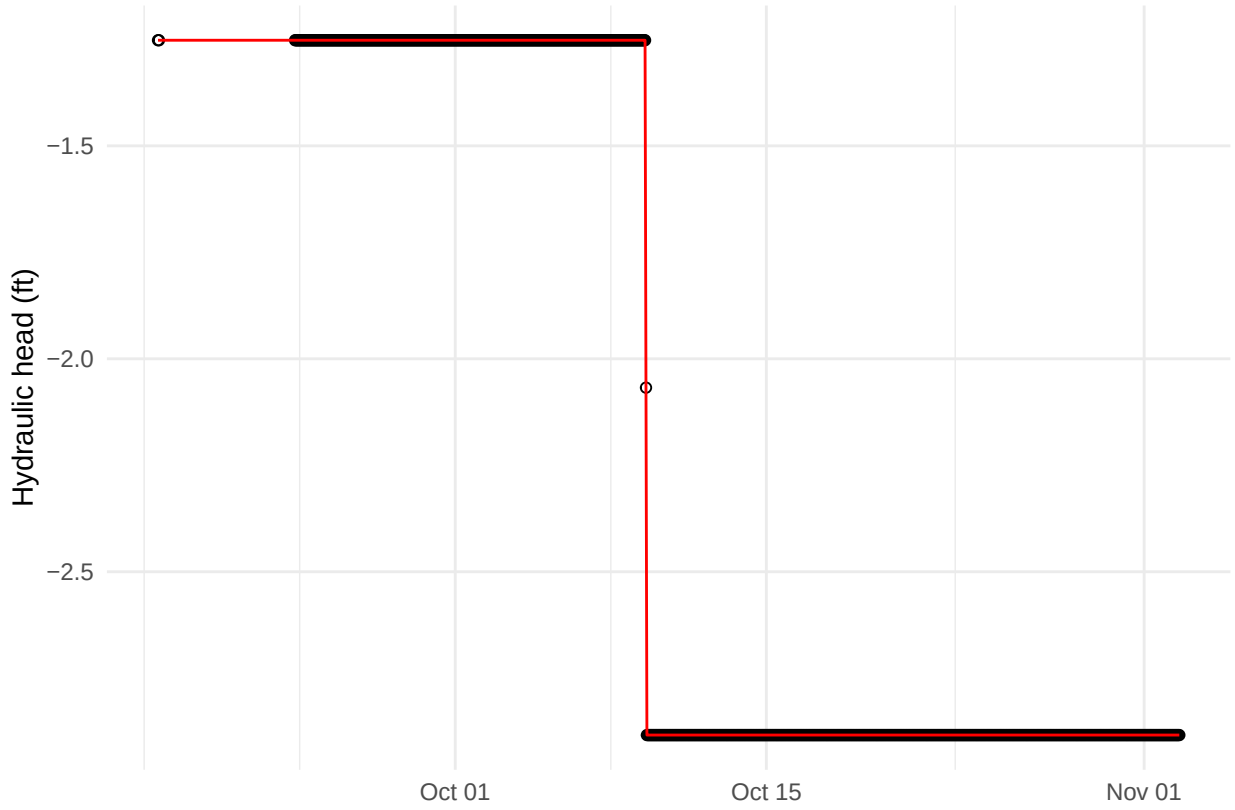
```
## # A tibble: 10 x 10
##    station_id sensor_number date_time           value units duration sensor_type
##    <chr>      <fct>         <dttm>              <dbl> <chr> <chr>    <chr>
##  1 MDB        292           2021-11-01 21:45:00  0.38 GAL/M E        FLWGALB
##  2 MDB        292           2021-11-01 22:00:00  0.37 GAL/M E        FLWGALB
##  3 MDB        292           2021-11-01 22:15:00  0.37 GAL/M E        FLWGALB
##  4 MDB        292           2021-11-01 22:30:00  0.39 GAL/M E        FLWGALB
##  5 MDB        292           2021-11-01 22:45:00  0.39 GAL/M E        FLWGALB
##  6 MDB        292           2021-11-01 23:00:00  0.37 GAL/M E        FLWGALB
##  7 MDB        292           2021-11-01 23:15:00  0.37 GAL/M E        FLWGALB
##  8 MDB        292           2021-11-01 23:30:00  0.37 GAL/M E        FLWGALB
##  9 MDB        292           2021-11-01 23:45:00  0.37 GAL/M E        FLWGALB
## 10 MDB        292           2021-11-02 00:00:00  0.37 GAL/M E        FLWGALB
## # ... with 3 more variables: obs_date <dttm>, data_flag <lgl>, date <date>
```

2.2 Extract data from DIGI Connect

A DIGI Connect sensor was used for experimentation purposes. Around October 9, it was moved from an open-air position to the bottom of a 5 gallon bucket. A corresponding step change in the hydraulic head at

this time is apparent in the data extracted from the DIGI API.

```
# source DIGI Connect API extract module
source(here("src/03_digi_api.R"))

# view the resulting plot
p_digi
```




```
# view some of the most recent data
digi %>%
  tail(10)
```

```
## # A tibble: 10 x 2
##    value dt
##    <dbl> <dttm>
##  1 -2.88 2021-11-02 05:00:00
##  2 -2.88 2021-11-02 06:00:00
##  3 -2.88 2021-11-02 07:00:00
##  4 -2.88 2021-11-02 08:00:00
##  5 -2.88 2021-11-02 09:00:00
##  6 -2.88 2021-11-02 10:00:00
##  7 -2.88 2021-11-02 11:00:00
##  8 -2.88 2021-11-02 12:00:00
##  9 -2.88 2021-11-02 13:00:00
## 10 -2.88 2021-11-02 14:00:00
```

2.3 Transform data

There are many ways to roll up and summarize data. The data available from data.ca.gov and DIGI only permit summary operations on injected volume. However, these approaches scale well to extracted volume, operating well pressure, and injection system water pressure.

First, we demonstrate summaries of injection well data at data.ca.gov sensors 291 and 292 for the month of March in 2021.

```
# source transform module
source(here("src/functions/f_transform_flow_volume.R"))

# by default, reported values are in gallons
f_transform_flow_volume(flow, "2021-03-01", "2021-03-31")
```

```
## # A tibble: 2 x 2
##   sensor_number volume_gal_sum
##   <fct>                  <dbl>
## 1 291                  194700.
## 2 292                 2170971.
```

```
# but we can report these in acre-feet with the "af = TRUE" argument
f_transform_flow_volume(flow, "2021-03-01", "2021-03-31", af = TRUE)
```

```
## # A tibble: 2 x 2
##   sensor_number volume_af_sum
##   <fct>                 <dbl>
## 1 291                   0.598
## 2 292                   6.66
```

```
# for a monthly rollup of values, use the "monthly_rollup = TRUE" argument
# here, we demonstrate monthy sums for February - March
f_transform_flow_volume(flow, "2021-02-01", "2021-03-31",
                        af = TRUE, monthly_rollup = TRUE)
```

```
## # A tibble: 4 x 3
##   year_month sensor_number volume_af_sum
##   <chr>      <fct>                 <dbl>
## 1 Feb 2021   291                  0.0215
## 2 Feb 2021   292                  2.21
## 3 Mar 2021   291                  0.598
## 4 Mar 2021   292                  6.84
```

Finally, we load data to a staging SQLite database, `staging.sqlite`, and provide data in two tables: `data_ca_gov_recharge` and `digi_injection`. This table is included for review. Schema, data loads, smart appending, and so on can be addressed in Phase 2.

```
library(RSQLite)
library(DBI)

# connect to staging.sqlite
con <- dbConnect(RSQLite::SQLite(), here("data_output/staging.sqlite"))
```

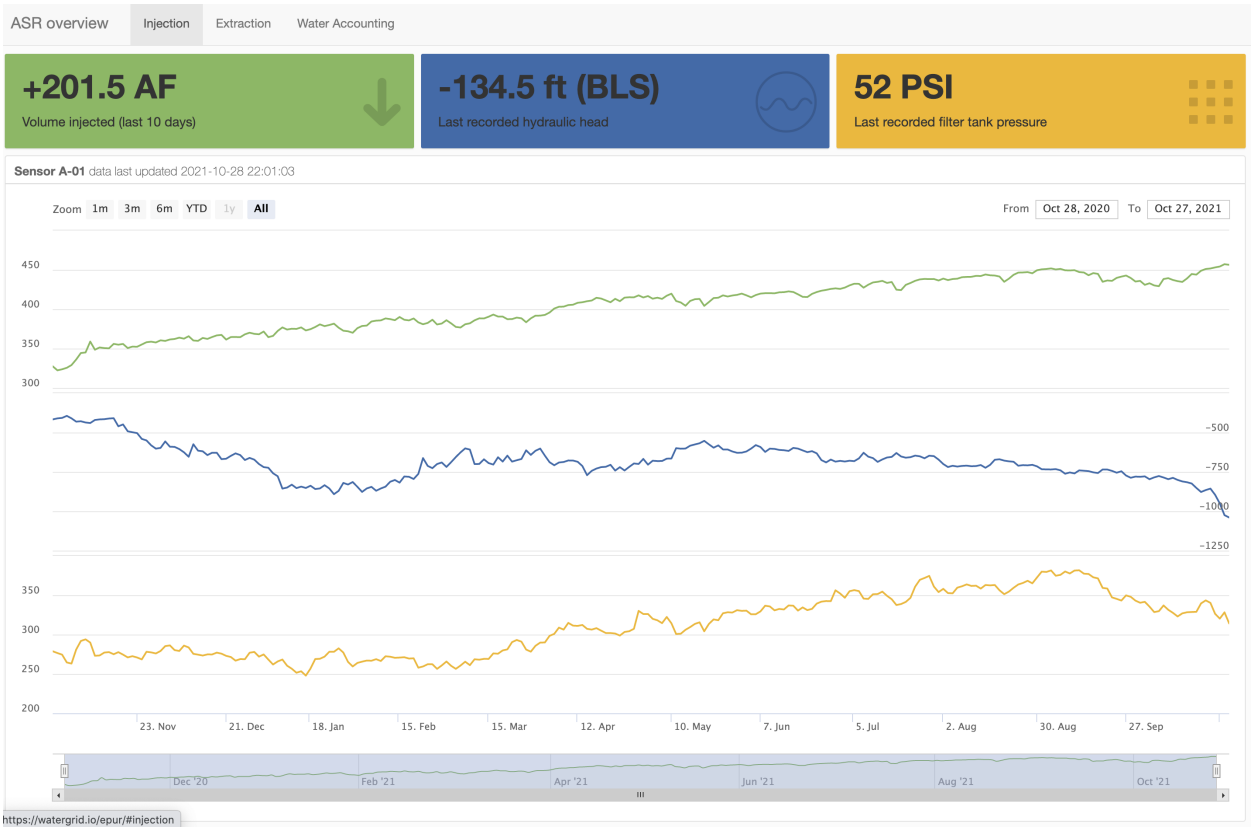
```
# write tables
dbWriteTable(con, "data_ca_gov_recharge", flow)
dbWriteTable(con, "digi_injection", digi)

# disconnect from databse
dbDisconnect(con)
```

3 Phase 1B deliverables

Whereas Phase 1A deliverables concern typical "backend" operations (e.g., extract, transform, load), Phase 1B deliverables concern "frontend" operations (e.g., user-facing data visualization and reports that depend on the backend). The process of "gluing together" the back and front end components of website or dashboard requires fairly complex and time consuming "plumbing", or code that links these extract, transform, load operations to the visualizations that a user sees. Because the "plumbing" that allows databases to speak with user interfaces is involved and routine, in this proof of concept scope of work, effort was directed to the design and layout of the user interface. "Dummy data" is used in all dashboard components. In a Phase 2 scope of work, this user interface can be operationalized to render data from any number of sources, including data.ca.gov, DIGI Connect, or otherwise.

The dashboard is visible at watergrid.io/epur. It is designed around an ASR operation use case. It has three tabs for Injection, Extraction, and Water Accounting. Each tab shows summary values for the timeseries data at the top of the page with linked colors to the respective timeseries below. The "Water Accounting" page shows a high level summary of the volume injected and extracted (on the previous two pages) and the year-to-date volume difference. Data can be downloaded by clicking on the "CSV" and "Excel" buttons in the interactive data table, and brought into Excel for further analysis. The data table is searchable, and sortable.

```
knitr::include_graphics(here("docs/dash_example.png"))
```



4 Conclusion

This Phase 1 scope of work delivers proof of concept backend (i.e., extract, transform, load operations) and frontend (i.e., user interface) components of a web dashboard to monitor and manage ASR well operations. In a future scope of work, both components can be modified to deliver other sources of information. Different URLs can be configured, and multiple dashboards can be rendered for many sites. Dashboards can be configured as public-facing and open, or sit behind password-protected walls for designated user groups.