Water Tracker

An Automated Surface Water Tracking System for California’s Central Valley

System and Operation Manual

Version 1.1, September 2018

Point Blue Conservation Science

3820 Cypress Drive #11, Petaluma, CA 95954

Phone: (707) 781-2555

www.pointblue.org

www.pointblue.org/watertracker
Contributing authors for v1.0-1.1: Nathan Elliott (nelliott@pointblue.org), Doug Moody (dmoody@pointblue.org), Blake Barbaree (bbarabee@pointblue.org), and Matthew Reiter (mreiter@pointblue.org)

Contact:

Matthew Reiter, PhD
Principal Scientist - Quantitative Ecologist
Point Blue Conservation Science
11912 Pine Forest Road, Truckee, CA 96161
760-417-9997
# CONTENTS

BACKGROUND ........................................................................................................................................ 1

OVERVIEW ........................................................................................................................................... 1

Satellite Imagery ................................................................................................................................. 1

Process .................................................................................................................................................. 2

Architecture .......................................................................................................................................... 3

Code ..................................................................................................................................................... 4

PROCESSING STEPS .......................................................................................................................... 4

Collect and Store Images .................................................................................................................... 6

Download Imagery ............................................................................................................................. 6

Process Imagery ................................................................................................................................... 7

Unzip Imagery ....................................................................................................................................... 7

Parse USGS Metadata ........................................................................................................................ 8

Standardize Imagery ........................................................................................................................... 8

Calculate Normalized Difference Indices .......................................................................................... 9

Create Raster Stack ............................................................................................................................. 10

Create Cloud Mask ............................................................................................................................. 10

Buffer Cloud Mask ............................................................................................................................ 10

Calculate Cloud Statistics .................................................................................................................. 11

Apply Cloud Mask ............................................................................................................................... 11

Predict Water ....................................................................................................................................... 11

Apply Threshold .................................................................................................................................... 12

Validate and Publish Grid .................................................................................................................. 12

Create Mosaic ....................................................................................................................................... 14

Validate and Publish Mosaic .............................................................................................................. 15

Generate Cloud-Filled Predictions ....................................................................................................... 15

Create Cloud-filled Mosaic ................................................................................................................ 16

Produce Zonal Statistics ..................................................................................................................... 16

Update Website ..................................................................................................................................... 17

Add to Map ........................................................................................................................................... 18

Create Custom Data Summaries ......................................................................................................... 18

FILE NAMING CONVENTIONS .......................................................................................................... 19

REFERENCES ....................................................................................................................................... 21
BACKGROUND

California’s Central Valley is a highly modified landscape where water and wetland management decisions determine the spatial distribution of surface water via an extensive system of levees, canals, and pumps (Hanak and Lund, 2012). Highly variable annual precipitation and snowpack, can limit the availability of freshwater creating a need for data on surface water distributions. Up-to-date data on surface water can be especially useful for water and wetland managers during periods of drought. Point Blue Conservation Science, with support from NASA, the Nature Conservancy, U.S. Fish and Wildlife Service, and the California Landscape Conservation Cooperative, developed an automated water tracking system for California’s Central Valley (Water Tracker; www.pointblue.org/watertracker) that provides data on surface water distributions at a relatively fine spatial (30m pixels) and temporal resolution (16-days), across broad spatial extent, and that are available as soon as possible after satellite imagery is acquired.

Water Tracker was launched in December 2016 and uses an innovative informatics process to convert regularly acquired, open source satellite imagery into spatial data on surface water distributions that are available for download online. By using data from Water Tracker, water and wetland managers as well as decision makers can implement informed and coordinated decisions on water use. Water Tracker also can guide decisions on when and where to put water on the landscape to meet wildlife habitat targets, enhance habitat connectivity, and promote ecosystem services such as groundwater recharge. Water data are available online approximately 15 – 20 days after an image is acquired. The Water Tracker website provides access to the classified spatial imagery and also provides graphical time-series summaries of the data and map-based visualizations that delineate between different types of wetlands and flooded agricultural lands. In August 2018, new capacity allowed for download and summary of the water data layers from user-selected spatial extents as well as the acquisition of cloud-filled imagery.

OVERVIEW

Satellite Imagery

NASA’s Landsat 8 satellite collects high-resolution (30 m) daytime imagery over nearly the entire surface of the Earth every 16 days (NASA, 2017). There are two primary sensors on Landsat 8: the Operational Land Imager (OLI) and the Thermal Infrared Sensor (TIRS). OLI imagery consists of nine spectral bands in the visible and near-infrared range; TIRS imagery consists of two bands in the thermal infrared (Barsi, 2014).

Landsat imagery is cataloged by scene using a unique combination of path and row numbers that identify the geographic location of an image and date the image was acquired. Four scenes cover the majority of the Central Valley, which was defined as the area within the boundaries of the Central Valley Joint Venture (CJVJ) region: p44r33 covers most of Sacramento Valley, p44r34 covers Suisun and part of the Delta, p43r34 covers the San Joaquin Basin, and p42r35 covers most of Tulare basin (Fig. 1).
Because the required scenes are on different paths, they are acquired on three different dates over the 16-day Landsat cycle:

A. Day 1 – first day of Landsat cycle.
B. Day 5 – scenes p44r33 and p44r34 acquired.
C. Day 7 – scene 3 (p42r35) acquired; mosaic created from most recent set of all four scenes.
D. Day 14 – scene 4 (p43r44) acquired; mosaic created from most recent set of all four scenes.
E. Day 16 – last day of Landsat cycle.

**Process**

As of January 2018, Landsat images are available for download on the U.S. Geological Survey (USGS) EarthExplorer website (https://earthexplorer.usgs.gov/) approximately 14 days after the image is initially taken. Water Tracker then downloads each image within 24 hours of it becoming available. After unzipping the scene into individual raster bands, each band is standardized and stacked as a multi-band raster. We also calculate several indices for use during the classification process, including the Normalized Difference Vegetation Index (NDVI; Rouse et al., 1974), the Normalized Difference Water Index (NDWI; McFeeters, 1996), and the Modified Normalized Difference Water Index (MNDWI; Xu, 2006).

Clouds are an inherent challenge to predicting land cover using satellite imagery. To account for clouded pixels within an image, Water Tracker creates a cloud mask for each scene with >5% cloud cover by parsing the Quality Assessment (QA) band, removing known false-cloud pixels, and applying a 2 km

---

**Fig. 1.** Map of California’s Central Valley showing the Central Valley Joint Venture planning region and the extent of Landsat 8 scenes.
buffer around remaining clouded pixels to exclude cloud edges and shadows. The cloud mask is applied to an image if cloud cover over the portion of the image within the Central Valley (as defined by the CVJV boundary; Fig. 1) is >5%. A cloud mask is not applied to images with <5% clouds because preliminary runs showed that the average rate of false-cloud pixels was 2-3% after known false-cloud pixels were removed. Images with >95% cloud cover over the portion of the image within the Central Valley are not processed.

A water classification model developed using methods of Reiter et al. (2015) is applied to the image to predict the probability of surface water in each pixel. A threshold is then used to create a binary raster depicting water/no water pixels. Water Tracker validates the binary raster using a set of test points and then masks the raster to exclude areas beyond the CVJV boundary (Fig. 1). The classified imagery along with relevant metadata is uploaded to the website after a total processing time of around 24 – 48 hours depending on complexity of the cloud mask.

After all four scenes have been acquired and processed for each Landsat cycle, Water Tracker creates a single map for the entire Central Valley by mosaicking the four scenes together. If an image is not processed because of excessive cloud cover, Water Tracker substitutes a blank (no data) scene into the mosaicked image. After mosaicking, summary statistics on water coverage by basin and land cover type are calculated and uploaded to the website for visualization. Each processed individual scene and valley-wide mosaic is uploaded to the website in geotiff format (.tif). Water Tracker also creates a Google Earth version (.kml) of each scene and mosaic that is available for download as well as used in the map viewer application. Mosaicking and upload requires an additional 12 – 36 hours of processing time before the mosaic is available in the map viewer.

See Figure 2 for a graphical overview of the processing steps and find details in the next section.

**Architecture**

Water Tracker uses the Google Cloud Platform to process Landsat 8 imagery: [https://console.cloud.google.com/storage/browser?project=automated-water](https://console.cloud.google.com/storage/browser?project=automated-water)

Martin Magana is the project administrator and billing manager for Point Blue. Nathan Elliott and Doug Moody lead development and operation. A user must have a Point Blue single sign-on account (Point Blue email) to be granted access.

Water Tracker uses several compute nodes, which are virtual machines set up in the Google Cloud Platform:

- **task-scheduler** (type: f1-micro with 1 vCPU, 0.6 GB memory) is a micro-instance that is always on and continuously runs code checking for available imagery. When new imagery is detected, it downloads the imagery and starts up another compute node to process it.

- **autowater6** (type: n1-standard-1 with 1 vCPU, 3.75 GB memory) is a standard instance that is usually off but spun up by **task-scheduler** when new imagery is ready for processing. **autowater6** runs the code processing the water classification maps.
- *autowater8* (type: n1-standard-4 with 4 vCPUs, 15 GB memory) is a heftier instance that is usually off. It is not part of the automated processing and is used for testing and manually-started bulk processing of past imagery (e.g., to correct errors in processed imagery or to calculate new statistics).

**Code**

Water Tracker is coded primarily in R (R Core Team, 2017) and utilizes the packages *raster* (Hijmans, 2016) and *rgdal* (Bivand *et al.*, 2017) for processing layers. Additional code is written in Python (3.x) and Bash (Unix Shell).

The majority Water Tracker’s code is maintained in the Point Blue Github repository *sparklemotion* (folder *autowater*): [github.com/pointblue/sparklemotion/tree/master/autowater](https://github.com/pointblue/sparklemotion/tree/master/autowater). This folder contains all R code as well as the master python script. The remainder of Water Tracker’s code is maintained in the Github repository *autowater*: [github.com/dm00dy/autowater](https://github.com/dm00dy/autowater); this folder contains the downloading and mapping scripts. Currently access to the Water Tracker code repository requires a Github account and permission granted by the Point Blue Github admin (Rob Serafini; rserafini@pointblue.org) or the code owners (Nathan Elliott [nelliott@pointblue.org] or Doug Moody [dmoody@pointblue.org]). Next steps include making the code open source.

In *autowater*, we split repositories into a branch for testing (nellyott) and one for production (master). Code for Water Tracker is tested on the branch *nellyott* before being pulled into the *master* branch once changes are confirmed to work. The repositories are cloned on local machines for development as well as Google compute nodes for testing and production (image processing). Code is deployed from GitHub to the compute nodes using the git command line (git pull origin *nellyott*). All code is must be manually pulled to nodes *autowater6* (for automated processing) and *autowater8* (for test and manual runs) after changes are committed to the master repository.

**PROCESSING STEPS**

Processing steps are described below by order of occurrence within each processing cycle. Section names correspond to processing steps in Figure 2. For each step, a contact person is listed along with the locations of code files (if applicable). Unless otherwise noted, R code (and all other code by Nathan Elliott) is stored in the repository *PointBlue/sparklemotion/autowater* while Bash scripts (and all other code by Doug Moody) are stored in the Github repository *dm00dy/autowater*. 
Fig. 2. Water Tracker data processing steps from top to bottom and left to right. Columns separate the process by entity and location. Blue boxes show code scripts, gray boxes represent data. Solid lines indicate control processes and dashed lines data inputs and outputs.
Collect and Store Images

Lead: NASA/USGS

The Landsat 8 satellite collects an image for each scene during daytime every 16 days. Each scene is available for download on EarthExplorer as three types of images: Real Time, Level 1, and Level 2. Real Time images are rapidly orthorectified and made available 1-2 days after collection. Level 1 products are radiometrically calibrated and orthorectified by NASA and USGS, which currently requires 8 – 12 days. Level 2 images are available 10-20 days after collection and have been corrected for atmospheric effects including standardized surface reflectance. For more details on image types see the Landsat 8 Surface Reflectance Product Guide (USGS, 2017).

Water Tracker requires Level 2 images with standardized surface reflectance. Currently tests are underway to reduce total processing time by using Real Time images and calculating surface reflectance on Point Blue servers. If successful, surface reflectance results from Real Time images will be used to create provisional water classification rasters that would be replaced once Level 2 imagery becomes available and is processed by Water Tracker.

Download Imagery

Lead: Doug Moody

Code: espa_order.sh, download_espa_order.py

In January 2017, ESPA released version 1.0.0 of their REST API, which allows orders to be submitted directly as a well-formed URL request. Each day at 1200 AM UTC an automated script logs into the EROS Science Processing Architecture (ESPA) server, initiates an authenticated session, and submits an order for each of the 4 scenes. Each scene can be ordered by submitting a simple encoded javascript string, such as:

```json
{
    "note": "",
    "olitirs8_collection": {
        "inputs": [
            "LC08_L1TP_043034_20180202_20180220_01_T1"
        ],
        "products": [
            "sr",
            "pixel_qa",
            "l1"
        ],
        "resampling_method": "nn",
        "format": "gtiff"
    }
}
```
If no scene was acquired that day, no further action is taken. If a scene was acquired, then the scene is added to a download queue, and is generally made available when NASA completes the scene post-processing. ESPA provides their own automated download script written in Python, which runs continuously checking the queue for new scenes to download. Because login attempts frequently fail due to network connectivity or server maintenance at ESPA, Water Tracker submits a direct order to ESPA for every scene until either the REST API returns an HTTP status code of “200 OK”, which indicates that a scene exists and is ready to be downloaded. Upon receipt of a “200” code, a semaphore file is written to the local filesystem which signals the automated ordering script to ignore that scene on subsequent dates.

Water Tracker verifies that new images are complete and not corrupted using a script comparing the “checksum” of each downloaded image with the target. A checksum is a relatively small sequence of digital values (e.g., a 10-digit number) that is derived from a larger sequence of digital values (e.g., a file). The function used to calculate the checksum is designed produce different results with even small differences in input, making it useful to verify data integrity. It’s sort of like a bar code that changes depending on the contents of the package it is on. The script then spins up a two-core compute node (autowater6, as of Jan 2018) and initiates the water classification process by calling the python script run_autowater_all.py; this script runs on the micro instance task-scheduler. Downloaded images are stored in the bucket pointblue-autowater-cdr.

**Process Imagery**

*Lead: Nathan Elliott*

*Code: run_autowater_all.py, run_autowater_function.r*

The classification process is controlled by a master Python script run_autowater_all.py. This script is run by the downloading script after the downloaded imagery has been validated. The master script initiates individual processing steps, handles and records errors, and notifies staff (currently Nathan Elliott) about processing progress and errors. The master script also assigns a date range and scene identifier (single path/row) before fully processing individual scenes. The master script calls the R functions via a series of system calls to run_autowater_function.r. Scenes are processed individually because an error during any step will stop all subsequent automated processing.

**Unzip Imagery**

*Lead: Nathan Elliott*

*Code: standardize_landsat_rasters.r*

In this step the Landsat image is extracted from a compressed file after download. Each scene is composed of 15 files: nine rasters from the Operational Land Imager (.tif), three from the Thermal Infrared Sensor (.tif), one quality assessment (QA) band (bitwise .tif), one cloud band (.tif), and one metadata file (.txt). These files are double compressed and bundled as a gzipped tar archive (.tar.gz; a tar file, or “tarball”, is a type of file compression). Files are unzipped in R by calling the Unix utility tar via a system call.
Extracted files are written to the Point Blue compute node storage for processing and not saved in google cloud storage.

**Parse USGS Metadata**

*Lead: Nathan Elliott*

*Code: compile_metadata.r*

USGS provides extensive metadata for each image contained in a plain text file. Each line uses an identifier and a value in a standardized format of `TAG = VALUE` (e.g., `LANDSAT_SCENE_ID = "LC80420352017007LGN01"`). These identifier/values are subset into groups using the tag/value pairs of `GROUP = GROUP_NAME` and `END_GROUP = GROUP_NAME`. Example below:

```r
GROUP = L1_METADATA_FILE
GROUP = METADATA_FILE_INFO
  ORIGIN = "Image courtesy of the U.S. Geological Survey"
  REQUEST_ID = "0501702164534_00019"
  LANDSAT_SCENE_ID = "LC80420352017007LGN01"
  LANDSAT_PRODUCT_ID = "LC08_L1GT_042035_20170107_20170218_01_T2"
  COLLECTION_NUMBER = 01
  FILE_DATE = 2017-02-18T14:06:44Z
  STATION_ID = "LGN"
  PROCESSING_SOFTWARE_VERSION = "LPGS_2.7.0"
END_GROUP = METADATA_FILE_INFO
GROUP = PRODUCT_METADATA
  <more metadata>
END_GROUP = PRODUCT_METADATA
GROUP = PRODUCT_METADATA
  <more groups and metadata>
END_GROUP = L1_METADATA_FILE
END
```

Water Tracker then uses R to convert each metadata text file into a data frame using the `read.table` command. The data frame is flattened (restructured to a format with all data for each image on the same row). Dataframes are then written to a csv file in `pvt/metadata/flat`. The final step of parsing is to combine all flattened metadata files. This step can be troublesome because tag names vary between metadata files when NASA/USGS adds new metadata tags or occasionally changes or removes existing tags. When combining the dataframes, the code checks the column names (tags) for each file, adding new fields with NAs for prior scenes where applicable.

The final combined metadata file is published as `pointblue-autowater-pub/metadata/landsat_metadata.csv` and available for download: [https://data.pointblue.org/apps/autowater/?page_id=40](https://data.pointblue.org/apps/autowater/?page_id=40)

**Standardize Imagery**

*Lead: Nathan Elliott*

*Code: standardize_landsat_rasters.r*

Further processing requires that each image have the same extent and origin (location of top left cell). However, images of the same scene on different dates have slightly different extents and
origins. This occurs because of slight differences in time and position of the satellite when an image is acquired. To account for image size variation, each image is cropped (reduced in size) to include only pixels that are sampled in every image, which reduces image size by around two percent. The four rasters used for standardization were created by calculating the intersection of all available Landsat 5 and Landsat 8 imagery from 2000 – 2016. After cropping, each raster image is resampled to verify a matching origin. Resampling is time intensive but allows direct comparison between dates. Resampling of each band requires around one hour.

Cropping and resampling are completed in R using the `raster` and `rgdal` packages. Numeric bands are resampled using bilinear interpolation; QA and cloud bands are resampled using nearest neighbor. Reference grids are stored in: `pointblue-autowater-pvt/reference/scene_extents/`. Water Tracker writes the standardized imagery to the Google cloud bucket `pointblue-autowater-pvt/snapped/`. The raw USGS metadata file is copied to the Google cloud (in `pointblue-autowater-pvt/metadata/raw/`). All extracted files are deleted after all processing is complete to conserve disk space.

**Calculate Normalized Difference Indices**

*Lead: Nathan Elliott*

*Code: calculate_normalized_difference_index.r*

Normalized difference indices are used to assist with classification of remote sensing imagery (e.g., Rouse *et al.*, 1974). Water Tracker calculates three normalized difference indices for use in classification of surface water:

- The Normalized Difference Vegetation Index (NDVI) is used to identify areas of live green vegetation. NDVI is calculated by dividing the difference of the near infra-red band (B5) and red bands (B4) by their sum (Rouse *et al.*, 1974). The formula for Landsat 8 is:
  \[ NDVI = \frac{(B5-B4)}{(B5+B4)}. \]

- The Normalized Difference Water Index (NDWI) is used to delineate bodies of surface water. NDWI is calculated by dividing the difference of the green (B3) and near infra-red bands (B5) by their sum (McFeeters, 1996). The formula for Landsat 8 is:
  \[ NDWI = \frac{(B3-B5)}{(B3+B5)}. \]

- The Modified Normalized Difference Water Index (MNDWI) is also used to delineate bodies of surface water. MNDWI is calculated by dividing the difference of the green (B3) and middle infra-red bands (B6) by their sum (Xu, 2006). The formula for Landsat 8 is:
  \[ MNDWI = \frac{(B3-B6)}{(B3+B6)}. \]

Calculations are performed using the `rgdal` and `raster` packages in R. Index files are saved as .tif raster files in `pointblue-autowater-pvt/ndi`. 
Create Raster Stack
Lead: Nathan Elliott
Code: stack_landsat_rasters.r

A stack of raster layers is created containing each of the thirteen bands of interest (1 – 7 from OLI; 9 – 11 from TIRS, NDVI, NDWI, and MNDWI). This step uses the raster and rgdal packages in R. The resulting multiband .tif files are written to pointblue-autowater-pvt/stacked.

Create Cloud Mask
Lead: Nathan Elliott
Code: create_cloud_mask.r

USGS provides pixel-by-pixel cloud information with each scene in the Quality Assessment (QA) raster. Cloud presence and probability is calculated using CFMask and stored as bitwise values in the QA raster (Fig. 3; USGS 2017b). Normal rasters have a single value per band stored as the binary value of all their bits. Bitwise rasters store multiple values per band using subsets of their bits to refer to different attributes. QA raster attributes consist of either one or two bits. Single-bit (True/False) attributes have individual bits with values of either 1 (condition exists) or 0 (condition does not exist). Two-bit (four value) attributes are used to record the confidence USGS has in the accuracy of certain estimates. For example, USGS classifies its confidence in the cloud probability estimates as 00 (not determined), 01 (low confidence, 0 – 33%), 10 (medium confidence, 34 – 66%), and 11 (high confidence, 67 – 100%). Water Tracker only uses high confidence band values because initial testing suggested using medium- and low-confidence intervals results in excessive masking of unclouded areas. The QA band is then parsed by applying the bitwise-and function bitwAnd into a raster with a value of 1 indicating clouded and No Data elsewhere.

Cloud mask calculations are performed using the rgdal and raster packages in R. The resulting .tif files are written to pointblue-autowater-pvt/cloud_mask/unbuffered.

Buffer Cloud Mask
Lead: Nathan Elliott
Code: buffer_cloud_mask.r

Fig. 3. Table of Landsat 8 Quality Band attributes and values. Bits must be read from right to left starting with bit 0. From USGS 2017b.
The Landsat 8 cloud QA band often incorrectly classifies pixels along the edge of a cloud or within a cloud shadow as non-cloud. Water Tracker accounts for these errors by increasing the size of the cloud mask using a 2-km buffer around an identified cloud pixel. To reduce processing time, a square buffer is calculated rather than a circular one. A square buffer uses a simple pixel count and chaining, whereas the circular buffer calculates distances for each pixel. Before buffering, Water Tracker also removes pixels that are repeatedly classified as clouded. These ‘false cloud’ pixels are removed by stacking all previous cloud masks from clear scenes (<5% cloudy) for Landsat 8 prior to 2017 and identifying pixels that are consistently flagged as cloudy in clear scenes (e.g., some buildings).

Buffers are calculated in R using the `raster` and `rgdal` packages. The final cloud mask raster is written to `pointblue-autowater-pvt/cloud_mask/buffered/`.

### Calculate Cloud Statistics

**Lead:** Nathan Elliott  
**Code:** `summarize_cloud_masks.r`

After creating a buffered cloud mask for a scene, Water Tracker calculates its coverage area and proportion by scene and CVJV basin. Coverage areas are also calculated for the raw, unbuffered cloud estimates from USGS. Cloud coverage area values are used later during the validation stage when image quality is assessed.

Coverage statistics are calculated in R using the `raster` and `rgdal` packages. Individual cloud statistic files are written as .csv files in `pointblue-autowater-pvt/cloud_mask/summary/`; a combined statistic file is written as `pointblue-autowater-pvt/cloud_mask/cloud_cover.csv`.

### Apply Cloud Mask

**Lead:** Nathan Elliott  
**Code:** `apply_cloud_mask.r`

Water Tracker does not apply a cloud mask to every scene if total cloud cover is <5%. If cloud cover is >5% and <95%, the buffered cloud mask is applied converting clouded pixels into No Data pixels. If cloud cover is >95% for any scene, processing is aborted.

The cloud mask is applied in R using the `raster` and `rgdal` packages. The resulting masked stack is written to `pointblue-autowater-pvt/stacked/masked/`.

### Predict Water

**Lead:** Nathan Elliott  
**Code:** `predict_water_rasters.r`
After the raster stack is masked, each image is classified on a pixel-by-pixel basis as either open water, not open water, or no data (usually because of cloud cover). Classification uses a Boosted Regression Tree model developed for the surface reflectance layers from ground truth data (detailed below), which is similar to techniques used by Reiter et al. (2015). The model includes all 11 bands from Landsat 8’s OLI and TIRS (corrected to surface reflectance) and three normalized difference indices (NDVI, NDWI, and MDWI; see above). The result is that each pixel receives a continuous value from 0 – 1 (inclusive) that represents the probability of the pixel containing open surface water. The classification model was trained on >10,000 water ground truth points from aerial surveys and ground surveys from July 2013 to April 2015 in addition to 5,000 points that were added manually using classified Landsat images. Test runs found that the model has high prediction accuracy (Area-Under-the-Curve = 0.96; 1 = perfect prediction, 0.5 = poor prediction [random]).

The model object is loaded from pointblue-autowater-pvt/model/landsat8_brt.RData. Processing is done in R with the gbm (Ridgeway et al., 2017), dismo (Hijmans et al., 2017), raster, and rgdal packages. Output rasters are written to pointblue-autowater-pvt/predicted/continuous/.

### Apply Threshold

**Lead:** Nathan Elliott  
**Code:** threshold_water_rasters.r

Water Tracker converts the continuous predicted water value produced for each pixel into a binary value (water/no water) using a threshold value of 0.07897. Therefore, if a predicted value is ≥0.07897 the pixel is classified as water (1) and values <0.07897 are classified as no water (0). This threshold was selected using the sensitivity-specificity sum maximization approach to maximize accuracy of the prediction (more details in Liu et al., 2005).

Threshold processing is completed in R with the raster and rgdal packages and the resulting rasters are written to pointblue-autowater-pvt/predicted/thresholded/.

### Validate and Publish Grid

**Lead:** Nathan Elliott  
**Code:** validate_and_publish_water_rasters.r

After an image has been classified into binary water/no water values, Water Tracker validates accuracy of the predictions using a set of validation points (1000 – 1800 depending on the scene). These points were selected manually in locations that are consistently water or no water. For example, ‘water’ validation points were in the centers of lakes, canals, rivers, sewage treatment ponds, and large permanent wetlands. ‘No water’ points were on hillsides, human structures (roads and buildings), pastures, and non-irrigated fields.
The validation process begins when Water Tracker extracts the binary pixels values from a classified raster and compares it to the value of the validation point. A mean accuracy rate is then calculated for all pixels with validation points within a scene as well as for all points within the Central Valley mosaic rasters. A matrix of misclassified pixels is computed that summarizes the number of true positives, false positives, true negatives, and false negatives.

Images are then ranked as Excellent, Very Good, Good, Fair, Poor, Bad, and Unusable. Categorization is based on the lowest score from two summary statistics at both the scene and valley-wide levels (validation accuracy rate and the amount of cloud cover) plus several USGS image quality metrics (Table 1). For example, if an image has high coverage (few clouds) but a low validation rate, the image would be ranked as ‘Poor’.

<table>
<thead>
<tr>
<th>Ranking</th>
<th>Result</th>
<th>Cloud Cover Rate</th>
<th>Validation Rate</th>
<th>USGS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Scene</td>
<td>Valley</td>
<td>Scene</td>
<td>Valley</td>
</tr>
<tr>
<td>Excellent</td>
<td>PASS</td>
<td>&lt;0.10</td>
<td>&lt;0.5</td>
<td>0.97 - 0.99</td>
</tr>
<tr>
<td>Very Good</td>
<td>PASS</td>
<td>0.10 - 0.25</td>
<td>0.50 - 0.10</td>
<td>0.95 - 0.97</td>
</tr>
<tr>
<td>Good</td>
<td>PASS</td>
<td>0.25 - 0.50</td>
<td>0.10 - 0.25</td>
<td>0.90 - 0.95</td>
</tr>
<tr>
<td>Fair</td>
<td>PASS</td>
<td>0.50 - 0.70</td>
<td>0.25 - 0.50</td>
<td>0.80 - 0.90</td>
</tr>
<tr>
<td>Poor</td>
<td>PASS</td>
<td>0.70 - 0.90</td>
<td>0.50 - 0.75</td>
<td>0.70 - 0.80</td>
</tr>
<tr>
<td>Bad</td>
<td>FAIL</td>
<td>0.90 - 0.99</td>
<td>0.75 - 0.95</td>
<td>0.60 - 0.70</td>
</tr>
<tr>
<td>Unusable</td>
<td>FAIL</td>
<td>&gt;0.99</td>
<td>&gt;0.95</td>
<td>&lt;0.60</td>
</tr>
</tbody>
</table>

**Table 1. Values used to determine image quality rankings.**

If a classified raster passes validation (see Table 1), it is published to the `pointblue-autowater-public/single_scenes/` (see next step for details). Grids that fail validation are not published on the website. A staff member (currently Nathan Elliott) receives an email notification upon the completion of the validation process for each image and can manually change its classification if warranted (for example, if thin cirrus clouds are present over much of the image but not masked out). Validation processing is completed in R using the `raster` and `rgdal` packages. The image quality ranking is written to `pointblue-autowater-public/metadata/image_quality.csv` and available for download on the Water Tracker webpage as “Image Quality Metadata”; https://data.pointblue.org/apps/autowater/?page_id=40.

Classified images that pass validation are compressed and published on the Water Tracker website as a gzipped tar ball (.tar.gz). Each published tar ball contains a geotiff water grid,.csv metadata file for the scene (extracted from `landsat_metadata.csv`), and spatial metadata .xml file for the validated water grid. The spatial metadata file uses the standard Federal Geographic Data Committee (FGDC) format and contains detailed information about the projection, extent, data values, provenance, and methodology of the water grid. FGDC format is compatible with ArcGIS programs (Environmental Systems Research Institute, Inc.) and most text editor or xml viewer programs.

Spatial metadata is compiled in R and compressed via a system call to tar. The spatial metadata template is located at `pointblue-autowater-pvt/metadata`
(central_valley_water_dataset_landsat8_metadata.xml). The final compressed files for each individual scene are written to pointblue-autowater-pub/single_scenes/ and linked to the website.

Create Mosaic
Lead: Nathan Elliott
Code: mosaic_water_rasters.r

Water Tracker creates two mosaics using images from two consecutive 16-day Landsat cycles. The first (Valley Mosaic #1) is composed of images for all four scenes and created after the last image is collected during each cycle (day 14; A, B, C, or D, E, F in Fig. 4). The second mosaic (Valley Mosaic #2) is composed of the last image of each cycle and the first three images of the subsequent cycle (C, D, E in Fig. 4).

<table>
<thead>
<tr>
<th>Cycle 1</th>
<th>Cycle 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A. Day 5 – scenes 1 and 2 (p44r33 and p44r34) acquired.</td>
<td>Mosaic 1 (1st cycle)</td>
</tr>
<tr>
<td>B. Day 7 – scene 3 (p42r35) acquired.</td>
<td>Mosaic 2 (1st cycle)</td>
</tr>
<tr>
<td>C. Day 14 – scene 4 (p43r44) acquired.</td>
<td>Mosaic 1 (2nd cycle)</td>
</tr>
<tr>
<td>D. Day 5 – scenes 1 and 2 (p44r33 and p44r34) acquired.</td>
<td></td>
</tr>
<tr>
<td>E. Day 7 – scene 3 (p42r35) acquired.</td>
<td></td>
</tr>
<tr>
<td>F. Day 14 – scene 4 (p43r44) acquired.</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4. Diagram showing the timing and composition of mosaics for the Central Valley.

The naming convention for mosaics is L8_valley_DATESTARTtoDATEEND_water.tif; DATESTART and DATEEND are the first and last dates respectively, of the four images used in each mosaic. Dates use the YYYYMMDD format. Classified water / no water values for areas of overlap between scenes are averaged in the mosaicking process (e.g., so a pixel that was classified as water in one overpass and not water in another overpass is averaged to 0.5).

Valley Mosaics #1 and Valley #2 are processed after all four scenes have been classified and validated. Upon the completion of validation for each scene, Water Tracker determines whether to begin the mosaicking process by assessing the status of the other three scenes for each mosaic. Water Tracker initiates mosaicking after processing for every scene because Level 2 images are sometimes available in non-sequential order after USGS calibration. If a validated scene is not available after three weeks (usually because of excessive cloud cover), a blank (all No Data pixels) version of that scene is loaded for mosaicking from pointblue-autowater-pvt/reference/blanks/.

Mosaicking is completed in R using the raster and rgdal packages in R. The resulting mosaics are written to pointblue-autowater-pvt/mosaics/.
**Validate and Publish Mosaic**
*Lead: Nathan Elliott*
*Code: validate_and_publish_water_mosaics.r*

After creating each mosaic, Water Tracker excludes areas outside the CVJV boundary using a clipping process. Prior to clipping, Water Tracker tests the mosaic for accuracy by validating pixel values and image spatial extent. The clipped mosaic is then published with metadata using methods described in the “Publish Grid” step.

Clipping, validation, and metadata compilation are completed in R using the `raster` and `rgdal` packages. The compressed mosaics are written to `pointblue-autowater-pub/mosaics/` and linked to the website.

**Generate Cloud-Filled Predictions**
*Lead: Nathan Elliott*
*Code: calculate_average_rasters.r, cloud_fill_rasters.r*

Because Landsat imagery can be substantially limited during cloudy periods, we developed a model to help us better predict what is under the clouds. Note again that images with >90% cloud cover are not even processed by Water Tracker. The cloud-fill model was developed by simulating clouds in cloud-free regions and then using a set of covariate data to generate a model to predict under the clouds. Covariates such as the previous 10-year average of open water, proportion of water in observed areas of same scene, water-year type, precipitation, and basin all helped to predict under clouds and are included as covariates in our model. The current cloud-fill model that is implemented in Water Tracker uses a 10-year rolling average, the proportion of each basin observed to be flooded, and the basin being filled. It has an R-squared value of 0.8 and an average standard error of approximately 14,000 acres. (For comparison, our previous extrapolation method had an average standard error of over 30,000 acres.). For additional details of the cloud-filling model see Reiter et al. 2018b.

After publishing the water classification raster for each scene, `calculate_average_rasters.r` creates a 10-year rolling average for that date if the image is more than 5% clouded (if less than 5% clouded, no mask is applied). The 10-year rolling average uses Landsat 8 imagery for 2013 – present and Landsat 5 imagery for 2003 – 2011 that was produced as part of prior work (Reiter et al., 2015) and uploaded to Water Tracker for this use. For each average, we include historical scenes that are within 16 days of the day of year of the image to fill, inversely weighting the values by the difference in days so that images closer by day of year are considered more accurate.

This 10-year rolling average is combined with the zonal statistics calculated above to produce a raster of predicted water under cloudy areas in the original image by `cloud_fill_rasters.r` (please see Reiter et al., 2018b for more details on the cloud filling model). These predicted values are used to fill in the NA values in the final water classification raster that are due to clouds to produce a seamless map of the valley. In cases when no original image exists (e.g., 90% or more clouds or problem with satellite), the 10-year average is used in place of the model.
Historical averages are written to `pointblue-autowater-pvt/historical_averages/` with the final section of the filename indicating the years included in the average (e.g., 10-year). Cloud-filled rasters are written to `pointblue-autowater-pvt/predicted/cloud-filled/`.

**Create Cloud-filled Mosaic**

*Lead: Nathan Elliott  
*Code: `mosaic_cloud_filled_rasters.r`

After creation of the single-scene cloud-filled rasters, Water Tracker mosaics them together with `mosaic_cloud_filled_rasters.r`. This follows the same process as described above in the ‘Create Mosaic’ section but substitutes cloud-filled rasters for scenes that are more than 5% cloudy (or scenes that are otherwise missing).

Mosaics are then zipped, packaged, and published with metadata as gzipped tarballs to `pointblue-auwotwater-pub/mosaics/cloud_filled/`. The entire cloud-filling process takes 2 – 4 hours.

**Produce Zonal Statistics**

*Lead: Nathan Elliott  
*Code: `calculate_basin_stats.r`, `calculate_wetland_stats.r`, `calculate_ag_stats.r`, `calculate_basin_wetland_stats.r`, `calculate_basin_ag_stats.r`, `combine_stats.r`, `create_plots_by_basin.r`

Eleven time series graphs on the website use zonal statistics on water data that are calculated during this step ([https://data.pointblue.org/apps/autowater/?page_id=201](https://data.pointblue.org/apps/autowater/?page_id=201)). One graph shows the total area (in hectares) of water by CVJV basin, and a second graph shows total area of water categorized by four land cover types of interest (corn, rice, seasonal wetlands, and semi-permanent wetlands). The final nine show water by cover type for each of the CVJV basins. Crop cover data are currently from The Nature Conservancy (unpublished data; derived from the USDA’s National Agriculture Statistic Service’s Cropscape) and wetland data is from Duck’s Unlimited (Petrik et al. 2014).

Water Tracker begins calculating zonal statistics by creating a coverage raster (observed area) for each mosaic that quantifies the total number of clouded and non-clouded pixels; value = 1 = not clouded (i.e., classified pixel) or value = 0 = clouded (i.e., excluded from classification). Water Tracker then calculates the total area covered by water in hectares and total area of classified pixels (value = 1 in coverage raster) for each land cover category using the `zonal()` function in R.

As of version 1.1, these summary scripts use the initial version of our cloud-filled mosaics. This has resulted in significant boosts to accuracy compared to a simple extrapolation. Total area of water calculations require now extrapolating the proportional area of water in the classified area to the entire scene only in the rare case when there is no available cloud-filled data. In that case, the following formula is used:

\[
\text{Total Area of Water (ha) for Scene or Land Cover Type} = \frac{\text{Observed Area of Water}}{\text{Classified Area}} \times \text{Total Area.}
\]
Coverage rasters are written to `pointblue-autowater-pvt/mosaics/coverage/`. Rasters created for categorical zonal statistics are written to `pointblue-autowater-pvt/reference/areas_of_interest/`. Intermediate statistical outputs are saved as `.csv` files in `pointblue-autowater-pvt/stats/`. Final statistical outputs are saved by CVJV basin as `pointblue-autowater-pub/stats/basin_stats.csv` and final cover type statistics are saved as `pointblue-autowater-pub/stats/combined_stats.csv`. By-basin cover type data are saved in `pointblue-autowater-pvt/stats/basin_ag.csv` and `pointblue-autowater-pvt/stats/basin_wetlands.csv`. Time series graphs on the website are updated automatically when new the `.csv` zonal statistics files are saved. All zonal statistics are calculated in R using the `raster` and `rgdal` packages.

We now are generating plots of the average probability of water by cover type and Central Valley Joint Venture basin in R using the highcharts package. Code for these graphs are located in `create_plots_by_basin.r` and outputs are written to `pointblue-autowater-pvt/stats/plots/by_basin/`. These plots are pulled over to the website after being written by the script in the next section.

Land cover data layers should be updated in future versions of Water Tracker if new layers are available. Once updated, related code will also require edits to account for new files names using read table process to identify date ranges for all cover layers.

**Update Website**

*Lead: Doug Moody*

*Code: epsa_catalog_embed.php*

At this point, a separate cron job (automatically-scheduled process) simply loops through all of the possible dates for each Landsat cycle. Using the PHP programming language, we try to open a file handle to read the data raster. If it is found, we write out an HTML link to the data. If it is not found, we write out a simple “No Data” message with no link.

Example code:

```php
$handle = fopen($the_link, "r");
if ($handle) {
    $page_content .= "<td >$mosaic_date_label<BR>"
    $page_content .= "<a href="$the_link">GeoTIFF</a>"
    $page_content .= " | <a href="$the_kml">KML</a>"
    fclose($handle);
} else {
    $page_content .= "<td >[No Data Available]"
}
```
Add to Map

Lead: Doug Moody
Code: espa_tile.sh, espa_mosaic.sh, espa_gdal.sh, gdal2tiles.py

After the final compressed files for each scene are written to pointblue-awater-pub/single_scenes/, a bash script pulls them down to an Amazon-based webservice to be hosted on the Water Tracker web site. The `gdal` software stack is used to compress the raw data rasters and to generate tiled RGB images for inclusion in the online map tool. The individual steps are as follows:

1.) Download the scene from the Google Cloud bucket, using Google’s proprietary `gsutil` toolchain to securely log in to the Google Cloud and copy the scene.
2.) Unzip the compressed scene (which has been packaged into a compressed tarball archive) and then use `gdalwarp` to utilize GeoTIFF native compression to shrink the file size.
3.) Use `gdaldem` to generate a simple two-color PNG file from the compressed data raster.
4.) Use `gdaldem` to generate a simple two-color GeoTiff file, which will be wrapped up in a KML file for use in Google Earth.
   Build a static set of image tiles from the PNG file which will then be added to the map in the web mapping format used by Google Maps, Bing Maps, and other online mapping tools.
5.) Add the new scenes to a MySQL database containing all the completed scenes and mosaics. The MySQL database is used to generate pulldown menus and static HTML links to the completed scenes and mosaics.
6.) Upload the KML file to pointblue-awater-pub/single_scenes/

Example code, where `${SCENE_ID}` is a variable representing the current Landsat scene identifier:

```
gdalwarp -co COMPRESS=LZW ${SCENE_ID}.tif ${SCENE_ID}.3310.tif

gdal_translate -of KMLSUPEROVERLAY ${SCENE_ID}.rgb.tif ${SCENE_ID}.kmz

gdal2tiles.py -z 1-12 ${SCENE_ID}.3310.png tiles/${SCENE_ID}
```

Create Custom Data Summaries

Lead: Doug Moody and Nathan Elliott
Code: summarize_water_rasters.r

Users can receive custom data summaries by drawing a polygon in the mapping interface or by uploading a zipped polygon shapefile. Pressing the ‘download scenes’ or ‘create report’ button in the mapping interface will produce an html report of the specified region. This report includes summary information, a dynamic plot of water across time (created using the highcharter package), and a table of
the water across time. The summary script also saves the data table as a csv file for easy viewing and manipulation.

The code is an R script that is executed via a command line call from the map website’s PHP script. It takes one required and two optional arguments: the directory containing the raster files to summarize (required), the directory to write the summary report in (if different than the input directory; optional), and whether to run silently or verbosely (default: verbosely). Example call:

```
path/to/Rscript.exe path/to/summarize_water_rasters.r folder/to/summarize optional/path/to/output.
```

Requires R version 3.4 or later (note: requires Ubuntu 18.04 or later). Requires R packages docopt, rgdal, raster, highcharter, htmlwidets, and their dependencies. Requires linux packages libcurl4-openssl-dev, libxml2-dev, and pandoc (if saving as self-contained file).

Setup code:

```
echo "deb [arch=amd64,i386] https://cloud.r-project.org/bin/linux/ubuntu artful/" | sudo tee -a /etc/apt/sources.list #add to source
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys E084DAB9 #add key of repo
sudo apt-get update
sudo apt-get install r-base
sudo apt-get install <package> #to install linux packages
```

NOTE: currently debugging problems with this process in which the final formatted report is not being exported, only the dynamic plot.

**FILE NAMING CONVENTIONS**

Landsat images follow a USGS naming convention until they are converted to Water Tracker’s naming convention during the ‘Standardize Imagery’ and ‘Parse USGS Metadata’ steps.

Water Tracker naming convention for processed images: L8_p##r##_YYYYMMDD_TYPE.tif

- **L8**: image is from the satellite Landsat 8
- **p##**: the path of the image (## is the two-digit path number)
- **r##**: the row of the image (## is the two-digit path number)
- **p##r##**: the unique scene identifier
- **YYYYMMDD**: date of the image, including the four digit year, two digit month, and two digit day with no spaces
- **TYPE**: unique alphabetic descriptor for the current raster (e.g., ‘snapped’ for just standardized or ‘water’ for the final classified water raster).
- **.tif**: all rasters are saved as geotiffs. Processed image metadata and statistics the same naming convention and are saved in .csv format.
USGS naming convention for Landsat 8 imagery (as of Feb 2018):

LXSS_LLLL_PPPRRR_YYYYMMDDD_yyyymmdd_CC_TX_B##

- L = Landsat (constant)
- X = Sensor ("C" = OLI/TIRS Combined, "O" = OLI-only, "T" = TIRS-only (Landsat 8), "E" = ETM+, "T" = TM (Landsat 1-7), "M" = MSS)
- SS = Satellite (e.g. “07” = Landsat 7, “08” = Landsat 8)
- LLLL = Processing correction level, L1TP (Precision Terrain), L1GT (Systematic Terrain), L1GS (Systematic)
- PPP = WRS path
- RRR = WRS row
- YYYYMMDD = Acquisition Date, Year (YYYY) Month (MM) Day (DD)
- yyyymmdd = Processing Date, Year (yyyy) Month (mm) Day (dd)
- CC = Collection number ("01", "02")
- TX = Collection category, RT = Real Time, T1 = Tier 1, T2 = Tier 2
- B = Band (constant; only appended to single band files)
- ## = Band number (e.g., “01”, “QA”)

Prior to April 2017, USGS used the following naming convention: LC80440332015224LGN00_B1

- SatPthRowYearDoyStnVn_Band
- Satellite, 3-digit path id, 3-digit row id, year, day of year, 3-letter station id, version, band designation
REFERENCES


