WGFP_Enc_Summaries_RShinyApp How-To

Location: U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\RFID\Detections\CodingDetections\WGFP_Enc_Summaries_RShinyApp

Most current R version used: 4.4.2

How-Last Updated: May 9, 2025

Table of Contents

Uses

- Cleans, combines, and displays data from stationary OregonRFID readers, Biomark antennas, mobile runs, recaptures, release data, pressure transducers, site visits, and USGS in one location.
- Displays fish movement by day in map, table, plot, and custom animation
- Generates dataset of MARK "states"
- QAQC of marker tags, lengths and weights, unknown tags, crosstalk, ghost tag movements, avian predation

General Notes

- All plots are interactive, used with plotly package. Hover over plot for more info and toggle what is displayed by clicking and double clicking items in the legend. Zoom in by clicking and dragging within the plot. Plots can also be saved as .jpgs.
- Datatables are also interactive through the DT package. Click the column titles of any column to sort in ascending/descending order. You can use the Search bar on the right or for each column to search/filter for any specific values. Most of the individual filters work, some don't, depending on column type I believe.
- It's important when adding a new updated file to the app that it had the exact same column names and order as the older file
- Data displayed in tables is often downloadable
    - When the table is accompanied by this download button, all data within the table will be downloaded

    - When the table is accompanied by this download button, only the data that is displayed on the screen will be downloaded. Display all rows of the table to download all data in this table.

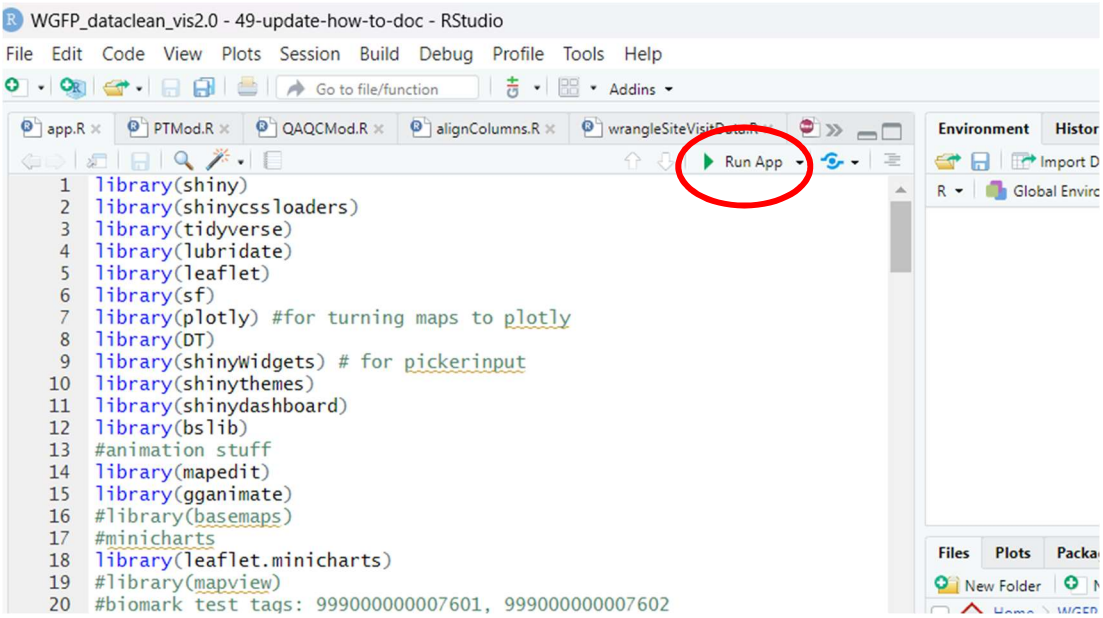
      

- UTM's for stationary and biomark stations are assigned in the metadata file
- DTY field for Stationary Antenna data refers to Detection Type in the raw .txt files but refers to date in the app

How to Open

OPENING FROM U: DRIVE

1. Navigate to U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\RFID\Detections\CodingDetections\WGFP_Enc_Summaries_RShinyApp and open the WGFP_dataclean_vis "R Project" file. Rstudio will open
2. If the "app.R" file isn't already open in the top left panel: in the bottom right panel, navigate to the "files" tab and open the app.R file.
3. Click "Run App" in the top right of the screen. You may be required to download some packages if this is the first time running the app.



Note: It typically takes 20-40 seconds to read in all files from the app. If you are frequently opening and closing the app in one session, you can speed this process up by reading in the files to your local environment. These files are located in the "data read ins" section of the app.r file. To read them in, select all the code in this section, then use cntrl+enter to run this code. It takes 20-40 seconds to read in the files (sometimes 4+ minutes when opeining from the U drive) but once you do this, when you run the app each subsequent time, it will boot right up in a couple seconds.

```
# Data Read Ins ------------------------------------------------------------

start_time <- Sys.time()
print("Reading in Static Files for app.....")
if(!exists("combinedData_df_list")){
  combinedData_df_list <- readRDS("data/flatFilesforApp/combinedData_df_list.rds")
}
if(!exists("indiv_datasets_list")){
  indiv_datasets_list <- readRDS("data/flatFilesforApp/indiv_datasets_list.rds")
}
if(!exists("Enc_release_data")){
  Enc_release_data <- readRDS("data/flatFilesforApp/Enc_release_data.rds")
}
if(!exists("states_data_list")){
  states_data_list <- readRDS("data/flatFilesforApp/states_data_list.rds")
}
if(!exists("movements_list")){
  movements_list <- readRDS("data/flatFilesforApp/movements_list.rds")
}
if(!exists("Marker_tags")){
  Marker_tags <- readRDS("data/flatFilesforApp/Marker_tags.rds")
}
if(!exists("unknown_tags")){
  unknown_tags <- readRDS("data/flatFilesforApp/unknown_tags.rds")
```

Navigating the App

About/How to Use Tab

This tab is a concise version of this document. It doesn't have much there now but can be modified to include the most relevant pieces of the how-to doc.

Individual Datasets Tab

This tab shows data incorporated into the app from different sources.

- Stationary Clean: A clean dataset of all Stationary antenna data with added UTMs, no marker tags, or duplicate rows. All timestamps and dates are in the same format. 900_ is taken off of the tags. See cleanStationary function in update data app for more info on cleaning.
- Biomark: All Biomark combined detections including Marker Tags
- Mobile: All detections from all Mobile Runs
- Recaptures: All recaptured fish
- Release: All release data along with frequency plots for length and weight. Note: entries with no L/W listed won't show up in the plot
- Ghost Data: a list of ghost tags that also have a "ghost date" associated with them, or the date they turned into a ghost tag
- Aviation predation: list of tags with a "predation date" associated with them when they turned into a predated tag
- Pressure Transducers: a combined dataframe of all pressure transducer data in the app
- USGS 15 Minute Data: USGS Data from their website from the stream gauge at Hitching Post. As of 5/22/2024, only water temp and discharge are read in
- Site Visit Data: Combined site visit data from all stationary and biomark sites.

Encounter Histories Tab

Encounter Release History Summary Wide

Shows ENC_Release_wide_summary dataframe from enc_hist_summary_wide_function.

- This tab is meant to be a summary file for all fish in "wide" data format. Each tag will have just one entry, along with release info and information about what sort of Events it has experienced over its life (ie Release, Red Barn Detection, Recapture, Mobile Detection).
- Filters are pretty self-explanatory. Click Render Table/Data to display new tables with filters.
- Should be same amount of entries as release file
- Columns and filters to note:
    - SiteCode_n is the raw number of detections at one antenna, or recaptured, etc.
    - SiteCode/Recapture binary columns are just whether or not a fish was detected or captured by that method
    - TotalEncounters adds the number of unique Events (a recapture, stationary detection, mobile detection, etc). There is a filter for this column in the sidebar.
    - Above/Below/Through the Dam tells whether a fish has gone through the dam or through the connectivity channel in its history, based on release info, recaps, and detections. It does NOT tell if a fish went upstream or downstream through the dam. There is also a filter for this in the sidebar
    - State Summary Filter (condensedAllStates column) is a quick history of the "states" dataframe (see MARK States Tab) for each fish. Consecutive, identical letters are removed, leaving only the concise history.
    - ChannelSummary: tells if a fish has had ANY detection on any CRCC channel
    - Went_above_dam_no_channel and subsequent columns are derived from the createMARKStatesfunction, which tells if a fish went below/above the dam using the channel or not from

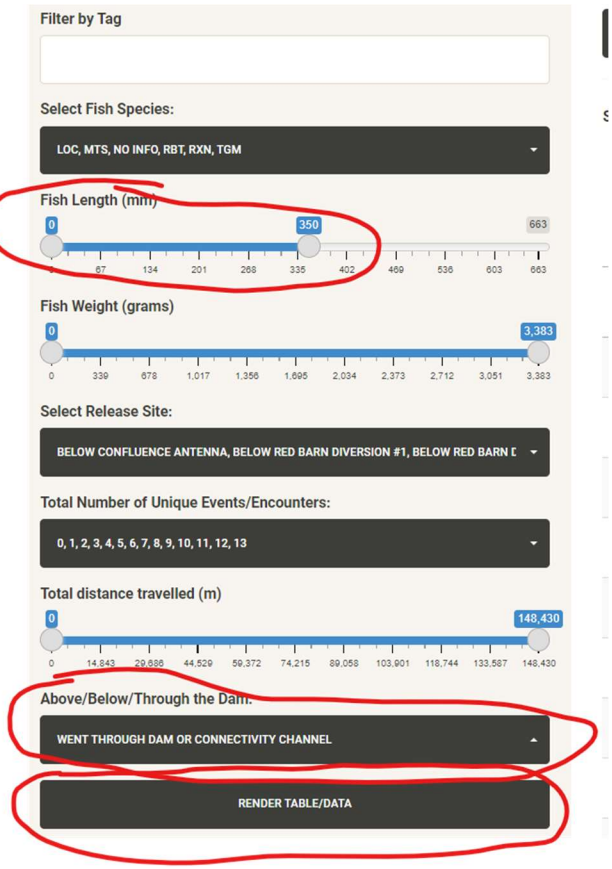
upstream or downstream, compared to through_dam which simply says if a fish as gone through the connectivity channel or dam or not. These columns give more specific info than through_dam, telling *how* the fish got above/below the dam: whether through the connectivity channel or not, and whether they started upstream or downstream. For example, a "TRUE" value in went_above_dam_no_channel would mean that the fish started below the dam, made it above the dam, but didn't use the connectivity channel. createMARKStatesfunction section for more details

- Sum_dist is the total number of meters travelled by a fish in its history. There's a filter for this in the sidebar.

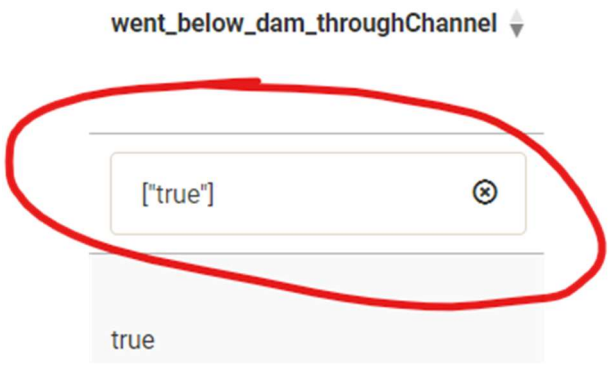
- This is a good tab to see how many fish have swam through CRCC, fish that have swam a long way, or find fish with extensive encounter histories while being able to filter on tag number, species, length/weight, Release Site.

Examples

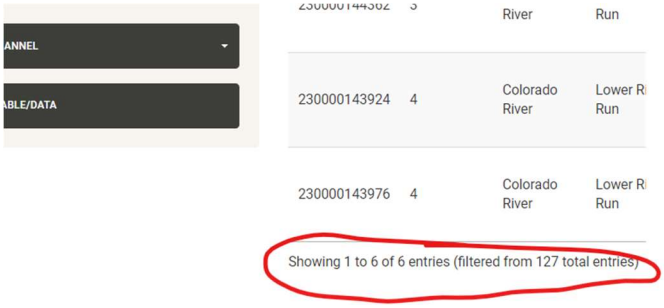
- Finding fish that have swam through the CRCC, specifically down through, that were less than 350mm.
  Strategy 1: Using Above/Below/Through_dam
  a. Adjust the filters pictured to desired categories and render table



  b. Scroll to the column went_below_dam_throughChannel and select "true" from the filters here



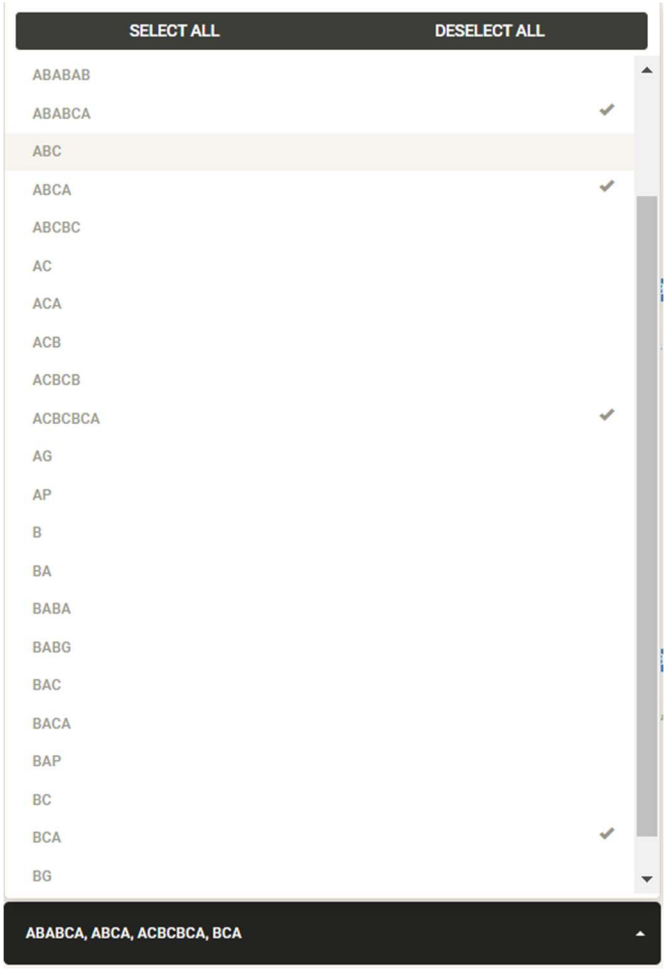
c. Scroll back to the beginning of the table and view the row numbers. There have been 6 fish of length 350 or below that have swam through the connectivity channel from upstream to downstream (as of 5/14/2024).



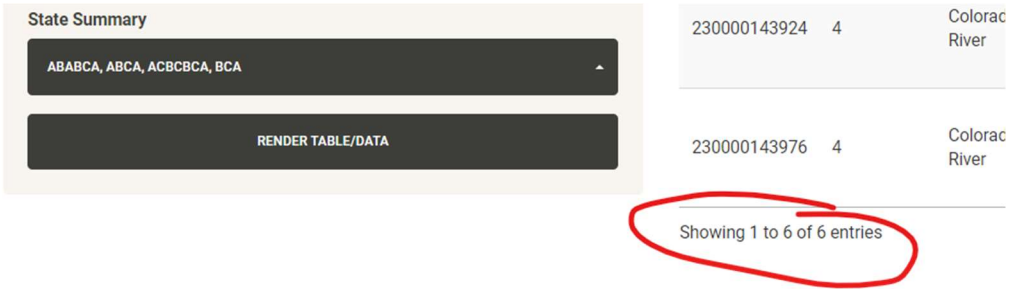
i. Note that a fish registers TRUE in went_below_dam_throughChannel not solely if it is detected in the CRCC (see All Events and plot for that example), but only if the fish was actually detected in CRCC then subsequently detected (whether at RB, HP, Biomark, recapture, or mobile run) downstream of CD or CS.

Strategy 2: Using Condensed States

- A fish will have the history "BCA" at some point (See "MARK States tab") if they have been recorded above the CRCC (whether by mobile detection, recapture, biomark or stationary antenna), then detected by any of the CRCC antennas, then was recorded downstream of the channel (whether by mobile detection, recapture, biomark or stationary antenna). In the sidebar filter, find and select those instances, in addition to changing the "Fish length" filter



- Render this table, and here you can see again the 6 fish we found in strategy 1

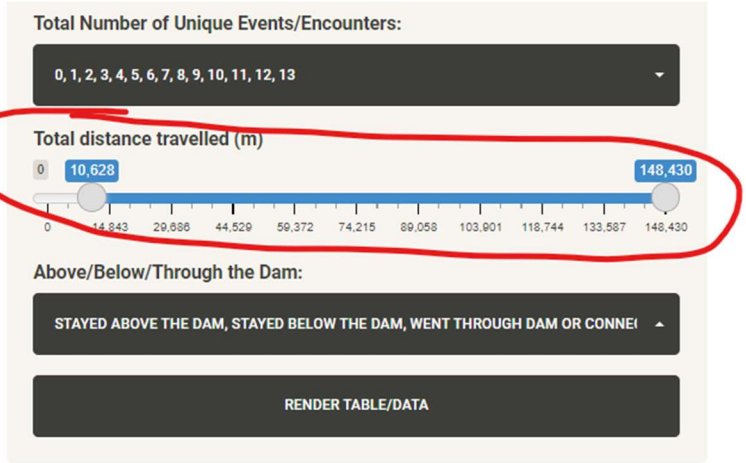


- This differs from finding "Sequences" in the CRCC (see "Sequences") because the states include recaptures, release, and mobile detections. However, you can't see timeframe for each fish with this data; use the Sequences tab or plug the tag into "ALL Encounter Histories" for that

- Identify Fish to explore as potentially avian predated

  Note: since new avian predation QAQC tab added, these examples are less relevant but still useful to understand how potential avian predated tags are found

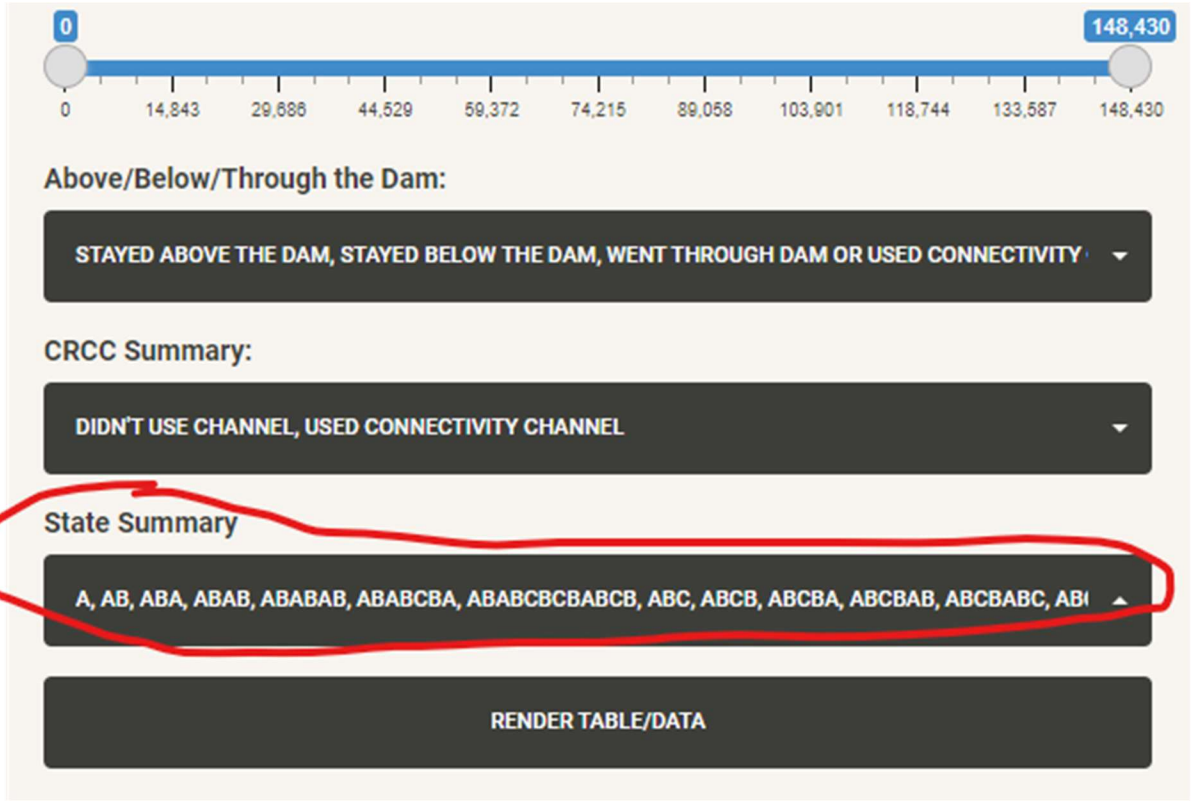
  - Total Distance Travelled
    - Fish that have swam abnormally long distances may have been avian predated. Adjust the distance moved filter and render table



    - Scroll to the "sum_dist" column and click it to sort by ascending

    - Scroll back to the beginning of the table to view the tag names, then manually view the encounter history of each tag in the movements tab or "All Encounter Histories" to see if their encounter history makes spatial and chronological sense. Many tags may already be in the avian predation file, so check that file in the "individual datasets" tab to see if that tag has already been accounted for.
  - Using State Summary

- You can also use the "States" filter to view all the different states that fish have been in in a week. Select the most dubious like ABABA, BAB, etc. This data is subset and used in "QAQC" -> "Avian Predation" -> "States"



- Copy Tag number and investigate in the "All Encounters" or "Movements" tab

All Encounter Histories

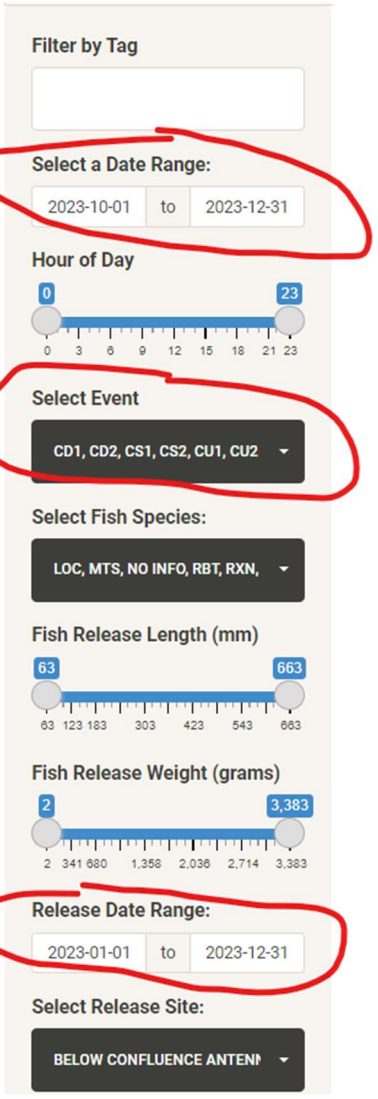
Shows and plots All_Events Dataframe from All_combined_events_function. These are all detections/events combined, including Stationary, Biomark, and mobile antennas, recaptures and releases. This shows raw detections so it's the largest dataframe shown in the app. 230000142723 has over 120k detections on its own at Hitching Post.

- Most filters are self-explanatory, but the checkboxes are important to note:
  - Display USGS Discharge Filter:
    - Allows you to filter data based off USGS discharge reading at Hitching Post within 13 hours of the detection (in winter, they record the data every 8-12 hours, 15 minutes in the summer). However, not every detection has an associated discharge reading because of this, so detections without a discharge reading get filtered out of the dataset immediately if you use this filter
  - Display Environmental Data Filter:
    - Allows you to filter data based off Pressure Transducer readings at specific sites within one hour of the detection. However, only sites that have a transducer at them will get this data associated with their detections and only within one hour, so using this filter will automatically filter out detection data that do not have any of these readings associated with the detections.
  - Remove Duplicate Days, TAGs, Events and UTMs:
    - Condenses detections into the first and last detections of the day as well as any other antenna detections with unique UTMs. In this way, it filters out most of the redundant detections across days. For example, using this filter condenses TAG 230000142723 down to 996 entries. This equates to about 259 days of detections from May 6, 2021 to Feb 3, 2022. All at Hitching Post.

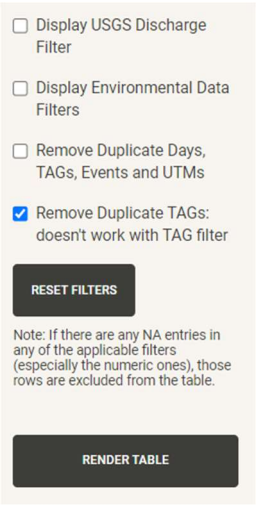
        - The df that results of this filter used on all detections is downloaded, brought into GIS, and used to make stations with, which is then subsequently used to make the "movements" df.
        - This is one of the most "useful" files you can get from this app because of how it condenses the dataset to the most relevant detections.
    - Remove Duplicate Tags Filter:
        - Will remove instances where there are multiple detections of the same tag. This is helpful in answering questions involving "How many unique fish?". Example: with this filter you could answer the question "How many unique rainbow trout over 250 mm hit Windy Gap Biomark and Hitching Post antennas during the day in spring 2021?"
- The plot tab will display anything made with the filtered data and the frequency table will display the raw number of detections from that filtered data for each antenna
    - Raw Detections Time Series
        - Total Detections by day on each antenna for the selected data
    - Raw Detection Frequencies by Event
        - Total Detections on each antenna for the selected data
- Tags with no lengths/weights listed are automatically changed to 0 for length and weight in order to avoid them getting automatically filtered out by the length/weight filters; ie all TGMs

Examples

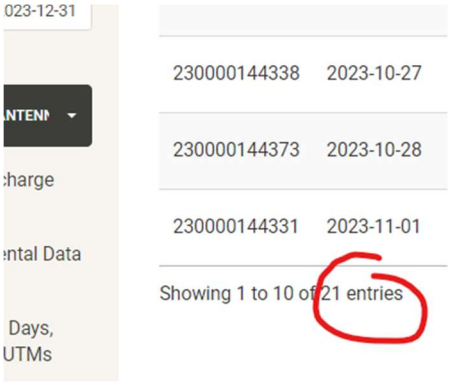
- How many fish released in 2023 were detected in the CRCC in Fall 2023?
    - In the filters tab, change the following filters:



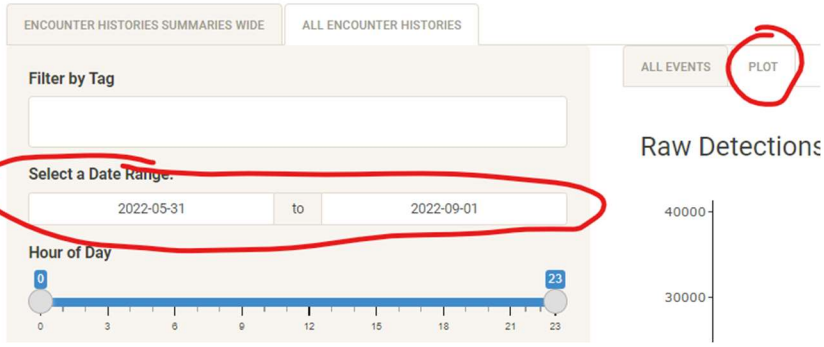
And make sure the "Remove Duplicate Tags" filter is checked before clicking "render table"



- The code filters the data based on the date range and events selected, then removes instances where the same tag shows up multiple times under these filters. The result is a df of unique tags that qualify for those filters. Find the number of fish by viewing the row number: 21



- Which antenna had the most raw detections in summer 2022?
  - Select a date range, render the table, and navigate to the "Plot and Table" tab



  - Scroll to the "Raw Detection Frequencies by Event" table to see the distribution of events during that time frame: Looks like HP3 had the most raw detections.

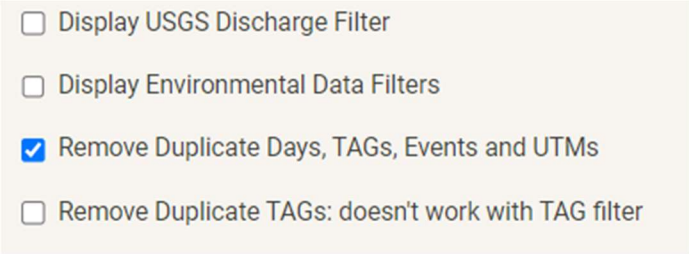
Raw Detection Frequencies by Event

Copy CSV Print Show 10 entries Search:

Event	Raw Detections
All	All
HP3	840487
WG1	606174
CF6	399632
HP4	300309
GD1	168594
WG2	98010
RB1	97001
RB2	72061
CF5	19243
Release	4837

Showing 1 to 10 of 22 entries Previous 1 2 3 Next

- These numbers can shoot up if a fish is sitting on an antenna for a long amount of time. If we want to get a better picture of how much activity actually occurred on a daily level between antennas, we can use the "Remove duplicate days, tags, events and UTM's" filter described above.

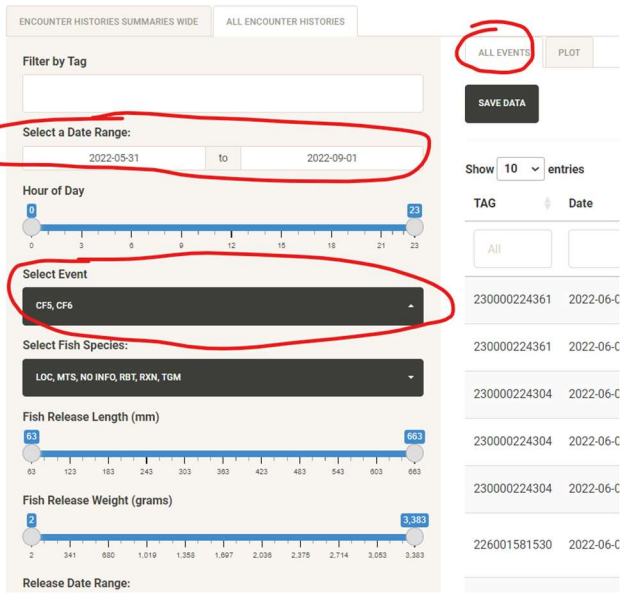


- The result shows a much better representation of what is going on at the sites across this timeframe, showing instances where fish had their first and last detections of the day and all detections in between. Hitching Post saw a lot of action.

Event	Raw Detections
All	All
HP3	1398
HP4	1139
RB2	402
RB1	337
CF6	246
Release	218
GD1	132
Recapture	65
WG1	52
CF5	49
Recapture and Release	10

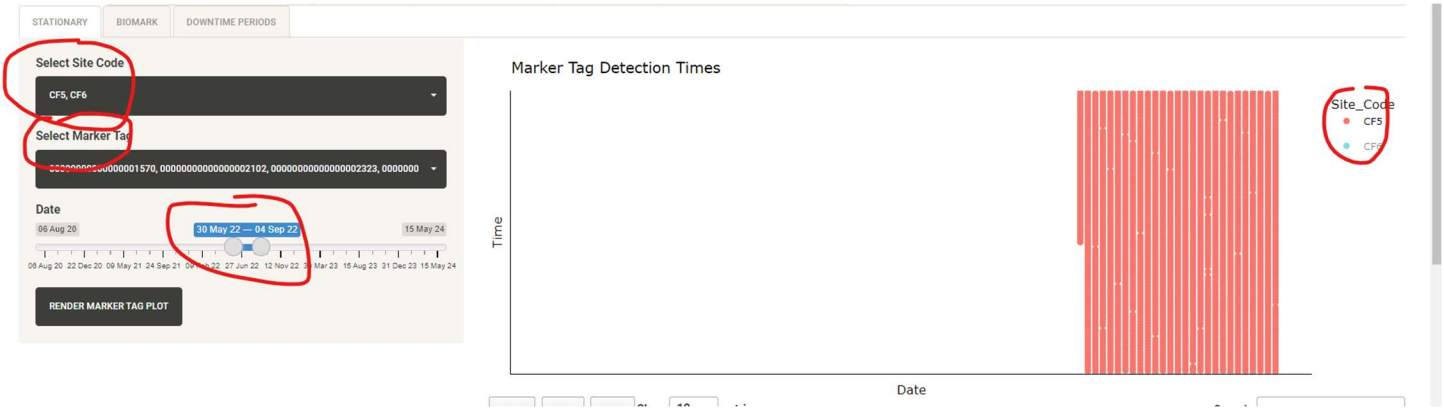
Showing 1 to 11 of 11 entries Previous 1 Next

- Why might CF5 have so much less action than CF6? To answer this, there's a few things to check. My first thought would be that there's a sculpin hanging out on just CF. We can go back to the "All Events" tab, select just those antennas in the "Event" filter, and scroll through the data so see if this is true



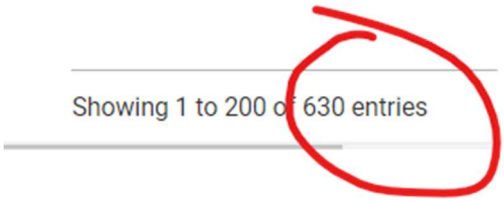
- We see that there are a lot of different tags on CF6 during this timeframe, so that theory doesn't work. Next thought is if the detection distance was very low, or the antenna was just off during this time. To check that data, we go to the QAQC tab.

- Under "Marker Tags", we can check if the antenna was running. Select CF5 and CF6 from the options, select all marker tags, change the timeframe to what we were looking at before, and render the data. Then toggle the plot displays by clicking in the legend.

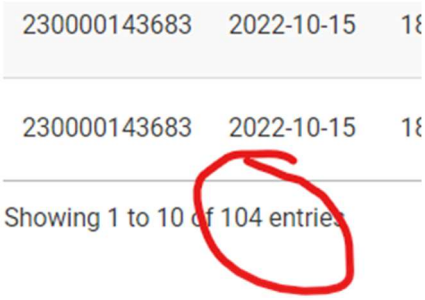


- What we learn is that CF5 actually didn't seem to be working for much of this time period. If you were to turn on CF6 on the map, you'd see marker tag detections pretty consistently across the same timeframe. This explains the discrepancy between CF5 and CF6 during this timeframe.

- Seeing a Fish's Full Encounter History/Investigate Individual Tag for Avian Predation
  - Using one of the potential avian predated tags found (most typically from one of the tables in the "QAQC" -> "Avian Predation" tab), we copy and paste that tag into the "All Encounter Histories" Tab. For example, I found 230000143683 (a big rainbow trout) by filtering for fish on Encounter Histories Summaries Wide tab that had 11 or more "events", and this fish shows up quite a bit in the Avian Predation tab as well.
  - Once the table is rendered, you can scroll to look at the fish's history (it default sorts by datetime). This fish has 630 encounters.

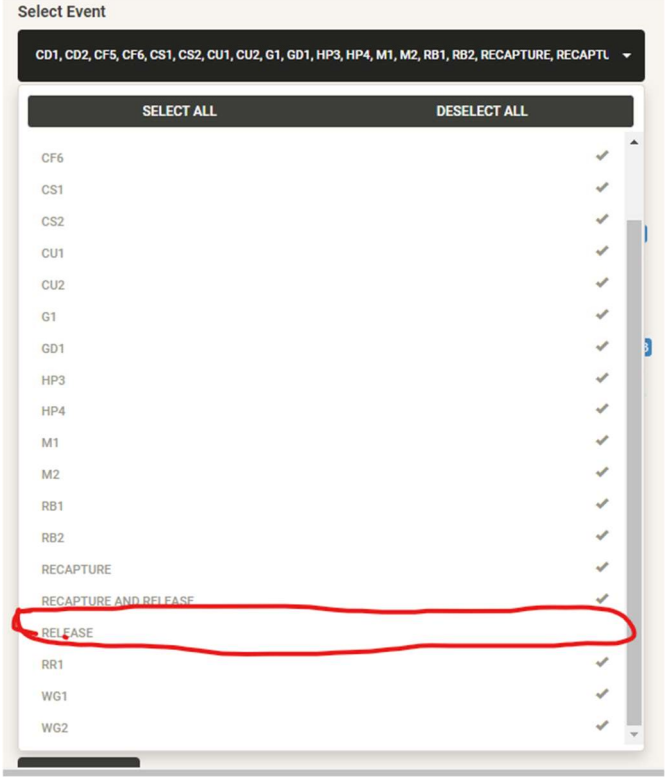


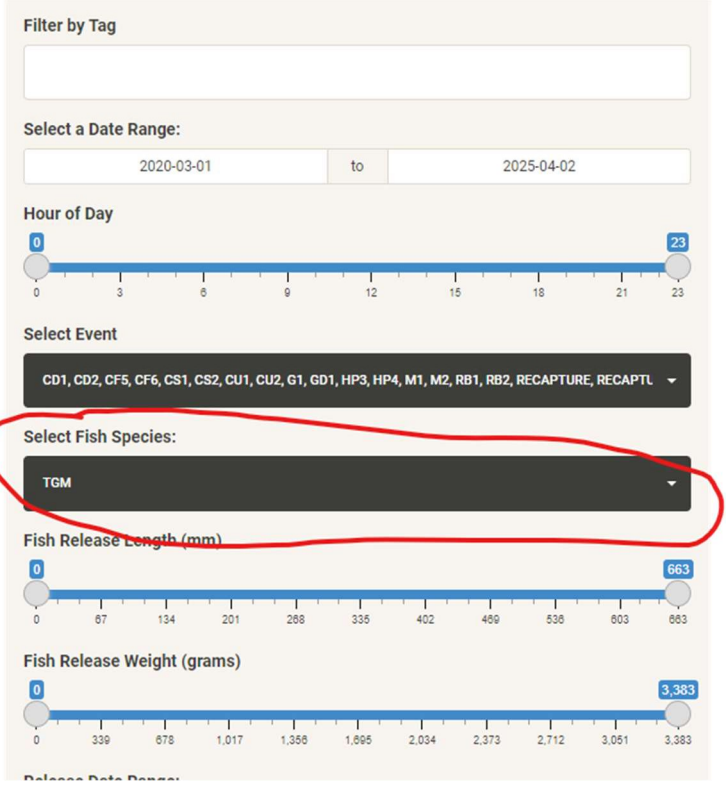
Scrolling through this data, you find yourself seeing a lot of detections on the same antenna across a bunch of days, which doesn't tell you much. However, using the "Remove duplicate days, tags, events and UTM's" filter here is helpful again, preserving useful detections and removing ones that are unnecessary. In this case, it takes the total encounters down to 104.



- Now this is easier to view. The Encounter history of the fish started with its release 2022-10-5, hit WG1 until the antenna was taken out of the river 2022-11-02. It hit hitching post and red barn next spring intermittently, including missing hp3 in late May, a time when the marker tag on HP3 wasn't being registered consistently (seen from the Marker tag QAQC tab). It was recaptured in June at lower Red Barn Fry Site and was registered as having grown 9mm in length. In December 2023, it made its way up the CRCC, hitting CD in December then resuming its journey in March, hitting CU March 12, then hit CF a week later where its more or less stayed since then (as of 5/14/2024). Based on this encounter history, I'd say the fish is not a tag succumbed to avian predation, just well-travelled.

- Have any Tiger Muskies hit our antennas?
  - Filter for species and deselect "Release" from the Events list



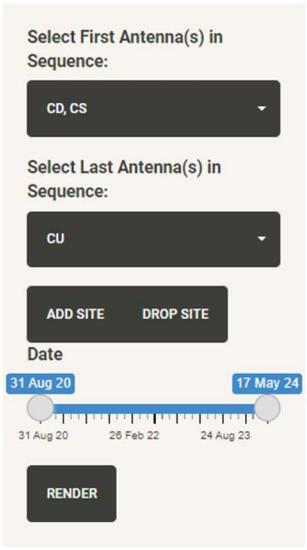


- Render Table. An empty table means no activity on any antennas nor recaptures.

Sequences

This is where you can pick a first antenna (or multiple), a last antenna (or multiple), and antennas in between then find instances where a fish have hit that sequence, and how long it has taken. This is an attempt to be able to easily answer questions like "How many times have fish used the connectivity channel?" or "are there any fish that have gone from Red Barn to the Granby Diversion Antenna?"

To use, select a downstream site (or more) and then select an upstream site (or more). Select Date range if desired. The table will then show instances where a fish has started at the first site(s) then was detected at the last site(s) with instances of middle detections in between. In the above example, this is helpful in showing instances where fish have moved through the CRCC (with CD, CS as first sites and CU as the last site).



SAVE DATA

Instances of Detections Between CD, CS and CU with middle antennas

Show 10 entries Search:

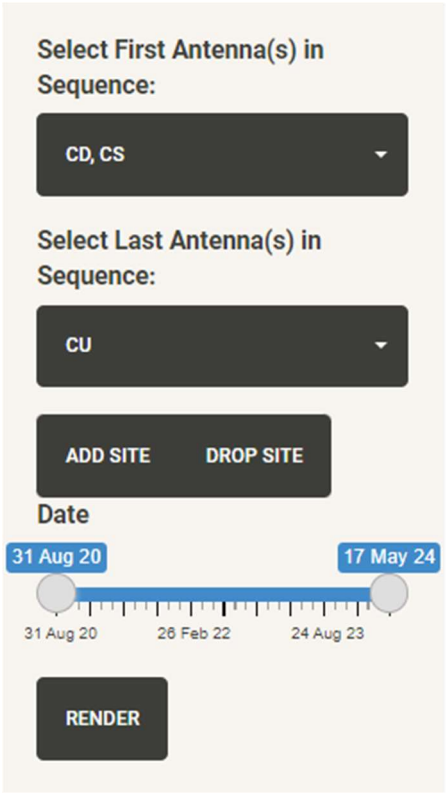
TAG	DatetimeDetectedAtFirstAntennas	DatetimeDetectedAtLastAntennas	Time Between First/Last Detections (User Friendly)	Time Between First/Last Detections (For Sorting)
All	All	All	/	/
230000143234	2024-01-09T12:50:26Z	2024-05-05T15:04:08Z	117.09 days	-10116822
230000143620	2023-10-26T16:47:39Z	2024-04-12T02:40:36Z	168.41 days	-14550777
230000143637	2024-04-21T20:25:27Z	2024-04-22T02:15:51Z	5.84 hours	-21024

- Fish that have traveled multiple times in the sequence will have multiple rows
- Ghost tags recorded in the ghost tags csv are included
- I added a user-optional filter to include avian predation tags in results. Not all screenshots have this included because I added it later.
- Mobile detections, recaptures, and release data are NOT included
- A fish starts the sequence when it last hits an antenna at the first selected site and ends it when it first hits an antenna from the last selected site
- For example, say the selected sequence was “CD, CS” to “CU” like the screenshot above. A fish could hit CD1 at 2023-11-03 12:04:15, hit CD2 on 2023-11-04 19:03:54, hit CS2 on 2023-11-06 03:34:41, hit CU1 on 2023-12-01 11:04:05, and CU2 on 2023-12-02 4:55:02. The Date/time sequence for this instance would be "2023-11-03 12:04:15" for the "DatetimeFirstDetectedatFirstAntennas" column, and 2023-12-01 11:04:05 for the "DatetimeFirstDetectedatLastAntennas" column. The time between US/DS detections would register as 28 days.
- When multiple antennas are selected for first and/or last, the sequence begins on the last instance of one of the first antennas. For example, if you have "CD, CS" selected for the Downstream antennas and a fish has hit CD2 on 2023-11-04 19:03:54, hit CU1 on 2023-12-01 11:04:05, and CU2 on 2023-12-02 4:55:02, the sequence will show up with 2023-11-04 19:03:54 in the "DatetimeFirstDetectedatFirstAntennas" column and 2023-12-01 11:04:05 in the DatetimeFirstDetectedatLastAntennas column.
- If there are other antennas that were hit in between the selected upstream sites and selected downstream sites, it breaks the sequence. So if you selected "HP" for first and "CF" for last, the results would not include fish that used the CRCC or hit WG1/2 along the way to get there. If you wanted to find these results, click “add site” and select all those antennas as the middle antennas.
- You cannot use a first/last antenna as one of the middle antennas, so sequences like “RB -> HP ->RB -> HP” aren’t possible
- However, you CAN use the same antenna for first/last, so sequences like “RB -> HP -> RB” are possible to find.
- Use “All Encounter Histories” tab to plug in individual tags and investigate results you feel don’t line up with what you’d expect to see in the results
- There’s currently an infrequent bug that I’m not really able to replicate where sometimes you click “add site” and multiple inputs pop up off of that one click, instead of just one. If this happens, restart the app.

Examples

- How many time have fish have swam up the connectivity channel?

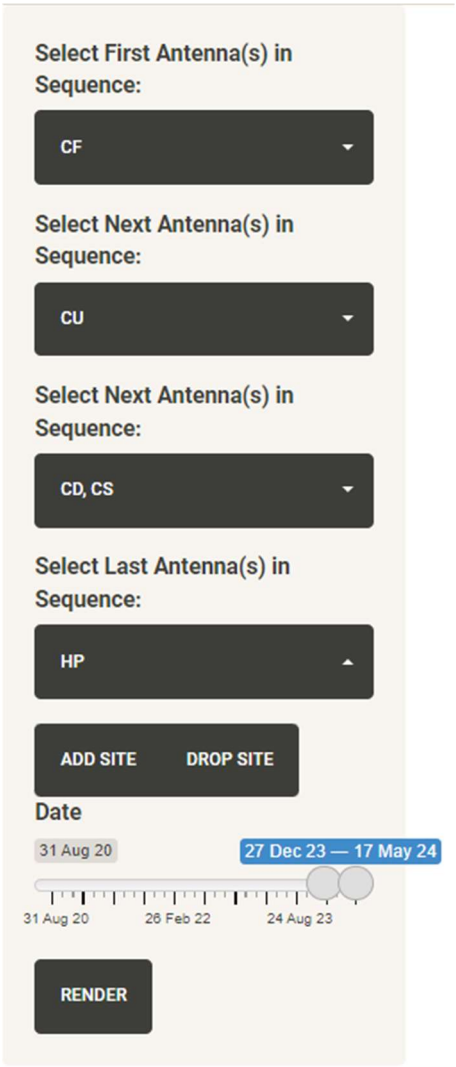
- Selecting CD, CS as downstream antennas and CU as upstream antennas can tell you how many times fish have used the channel.



- Check the number of rows in the dataframe to get the answer: 20 times (as of May 27, 2024). A couple fish have made this journey multiple times.

Showing 1 to 10 of 20 entries

- How many times have fish swam from Confluence to Hitching Post through the channel in 2024?
  - Select the appropriate filters, using "Add Site" to include CRCC antennas in the desired sequence.
    - Note: There's a difference in selecting this sequence vs just selecting "CU, CD, CS" in just one middle antenna input. Selecting all the CRCC antennas in the same "middle antennas" input would allow for the fish to have a history like "2024-01-02 13:00:00 CF, 2024-01-03 15:00:00 CU, 2024-01-04 13:00:00 HP" and get included in the results dataframe when it misses CD/CS detections. The current inputs state that the fish have to first hit CF, then CU, then CD or CS before hitting Hitching Post. Another example: in a hypothetical scenario a fish could have this history: "2024-01-02 13:00:00 CF, 2024-01-03 15:00:00 CU, 2024-01-04 13:00:00 CS, 2024-01-08 13:00:00 CU, 2024-01-08 13:00:00 HP". With the current inputs, this instance would not be included in the results data, but if the inputs were instead changed to "CF", then "CS/CD/CU", then "HP", then this instance would be included since the fish just needs to have hit *any* of the selected middle antenna, not in a specific order.



- In this case, there has been one fish (as of May 27 2024).

SAVE DATA

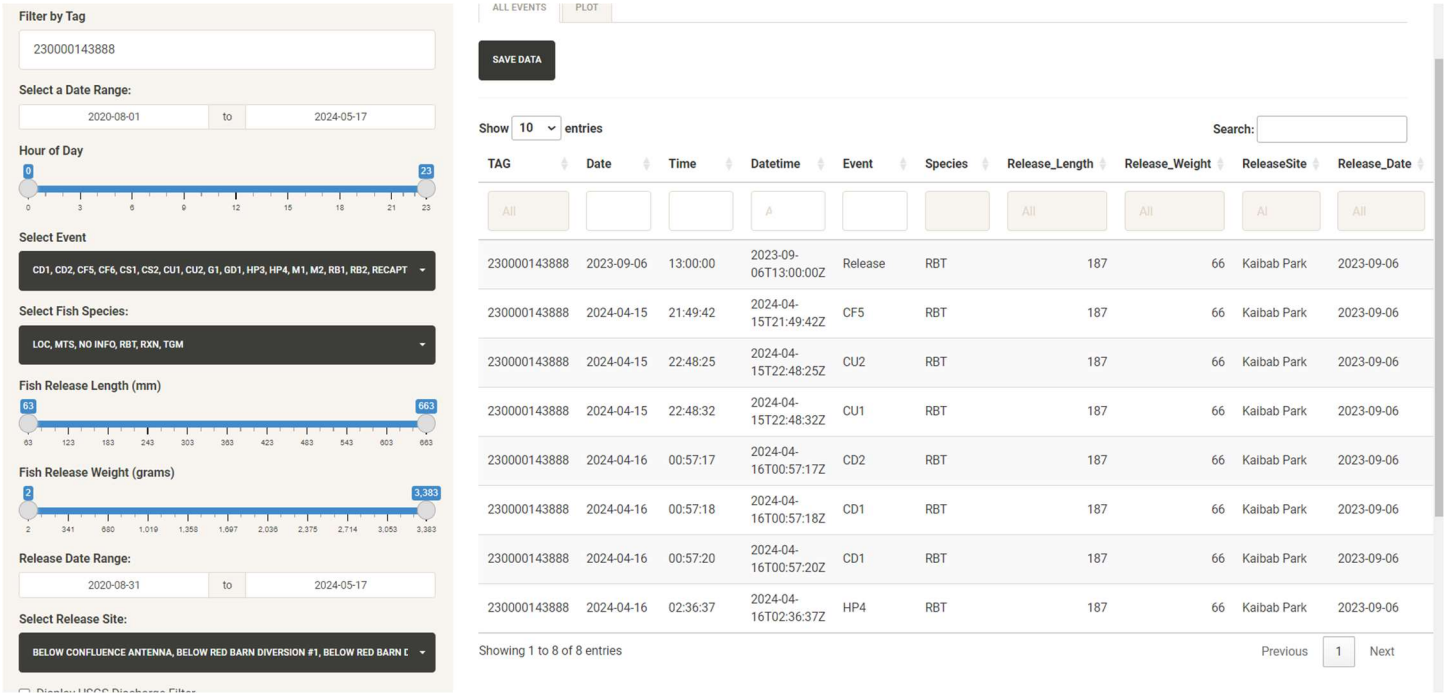
Instances of Detections Between CF and HP with middle antennas CU, c("CD", "CS")

Show 10 entries Search:

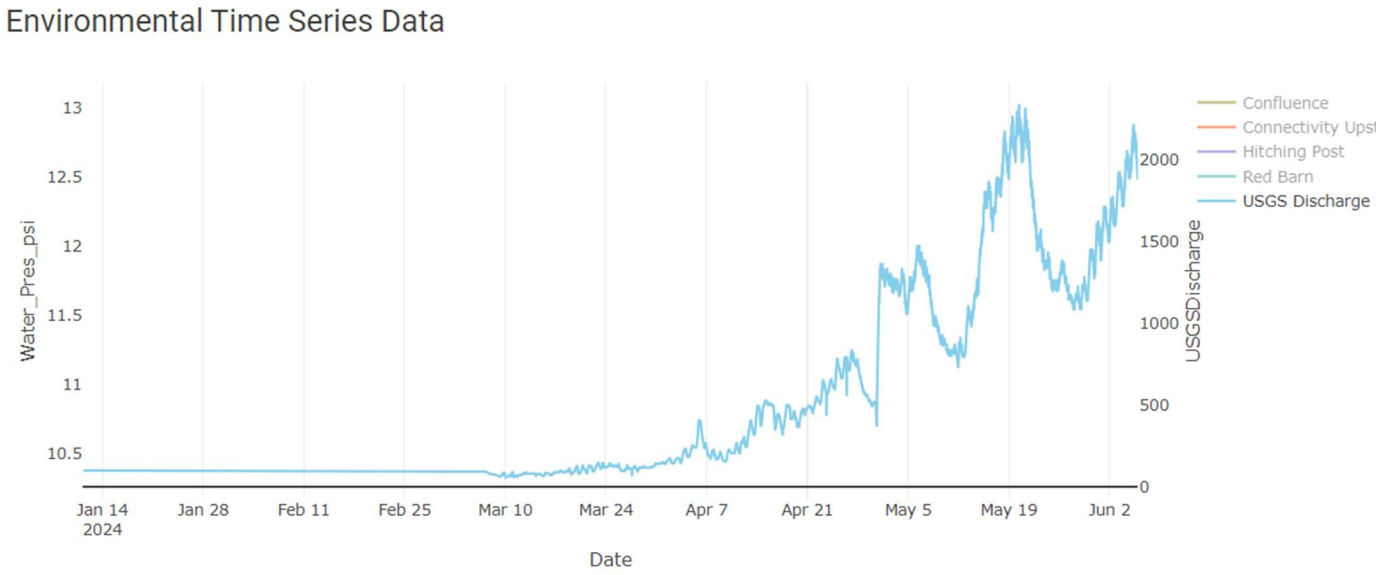
TAG	DatetimeDetectedAtFirstAntennas	DatetimeDetectedAtLastAntennas	Time Between US/DS Detections (User Friendly)	Time Between US/DS Detections (For Sorting)
All	All	All	/	/
230000143888	2024-04-15T21:49:42Z	2024-04-16T02:36:37Z	4.78 hours	-17215

Showing 1 to 1 of 1 entries Previous 1 Next

- We can plug this tag number into the "All Encounter Histories" tab or Movements tab to get more details about that instance.



- One interesting takeaway is that the fish missed detection on CF6 and HP3 in its journey. A quick look at the marker tags data (see QAQC-Marker tags) shows us that the antennas were functioning well during that time. If you look at the time between the detections though, you'll see it took only 4.78 hours late at night for the fish to travel that far; it was moving super fast. Looking at the movements data –map and table tab in the "MPerSecond" column, we can see specifically that between CF and CU, the fish moved an average of 0.01 m/s, between CU and CD it moved .18 m/s, and between CD and HP it moved .34 m/s.
- A quick look at the discharge data (see "Pressure Transducer, USGS and Detection distance" tab) shows that flows measured at HP really started to rise around then, so it had some good momentum.



- What is the quickest time it's taken a fish to swim from RB to Hitching Post?
  - Select "RB" as the first antenna and "HP" as the last one. Render the data then click "Time Between First/Last Detections (for sorting)" to sort the results in either ascending or descending order.

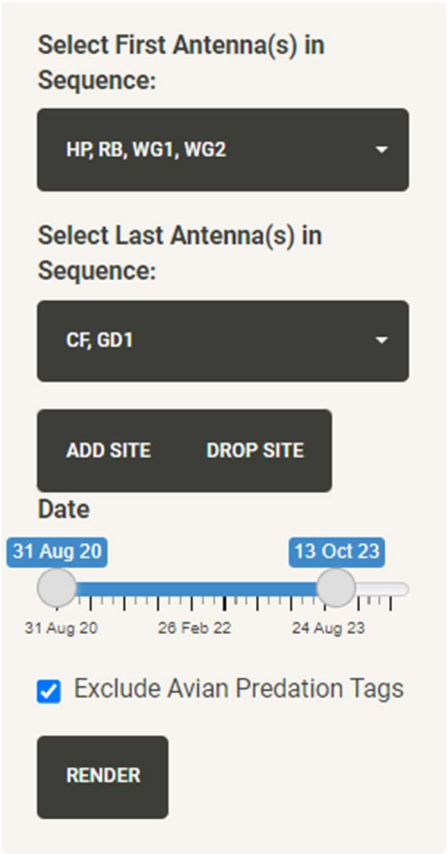
Instances of Detections Between RB and HP with middle antennas

Show 10 entries Search:

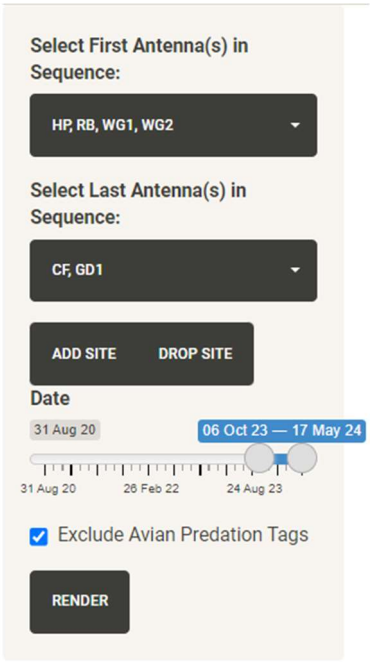
TAG	DatetimeDetectedAtFirstAntennas	DatetimeDetectedAtLastAntennas	Time Between First/Last Detections (User Friendly)	Time Between First/Last Detections (For Sorting)
All	All	All	/	/
230000228275	2021-04-29T10:46:53Z	2021-04-29T11:25:03Z	38.17 minutes	-2290
230000228275	2021-04-27T07:29:57Z	2021-04-27T08:10:32Z	40.58 minutes	-2435
230000228275	2021-04-21T17:52:03Z	2021-04-21T18:35:20Z	43.28 minutes	-2597
230000228314	2021-05-03T11:19:50Z	2021-05-03T12:21:01Z	1.02 hours	-3671

There was a fish that has gone multiple times in under an hour upstream from RB to HP. This is a bit of a red flag! Input the tag number into the "All Encounter Histories" tab to investigate further and record this in the avianpredation.csv if judged to be predation.

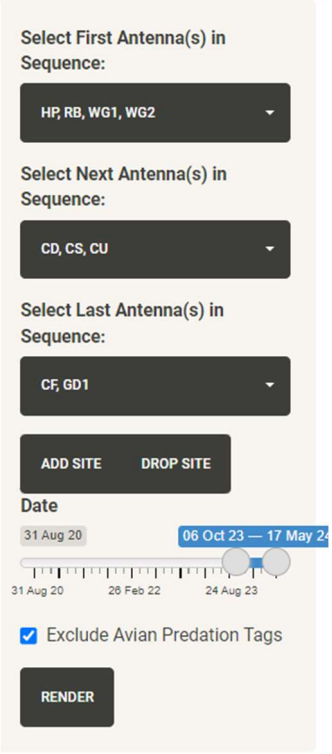
- How many fish have gone through the dam to go upstream since the connectivity channel was built vs before? (As of May 17, 2024)
    - Set filters to below dam antennas for the first antennas, then the upstream of dam filters for the last antennas. Set the date range to "pre CRCC antenna construction" and filter out instances of avian predation



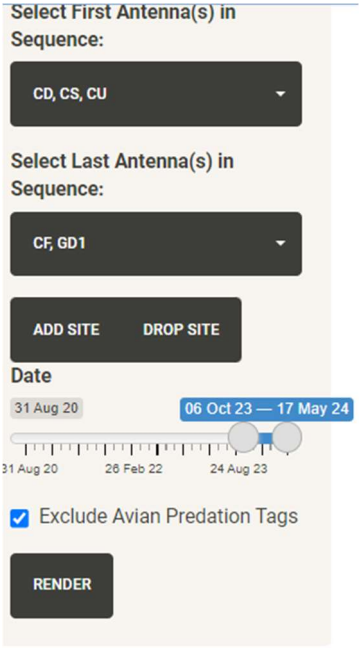
    - Here we get 44 instances where non-avian predated fish* have gone upstream through the dam
        - *based on what is in the avian predation file. There are selected sequences used in the "QAQC" -> "Avian Predation" tab used to help flag avian predation. See this section for more info.
    - Change the filters to post-CRCC construction



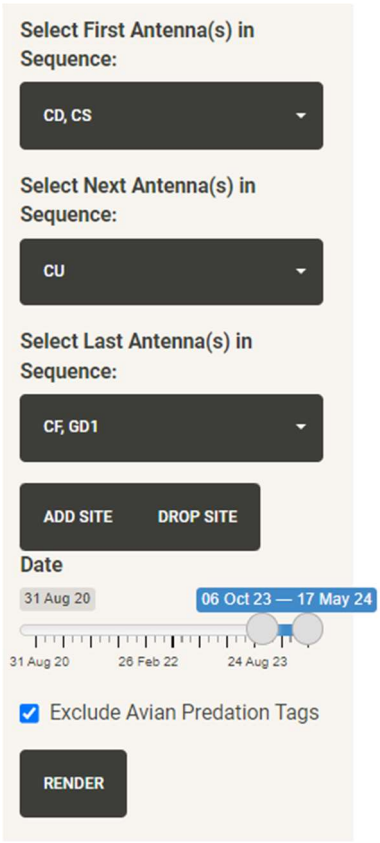
- Here we see only 3 instances of that sequence since CRCC was built
- To compare, add a Site and adjust the filters as you wish



- In this instance, there's one instance of a tag recording this sequence. Since the CRCC hasn't been built long, a better measure might just be to exclude the below dam antennas.



- These filters yield 24 instances. However, it should be noted that this would include fish that might have swam down from CF, hit CU, then headed back to confluence. To get the more-desired sequence, the following filters might be most helpful.



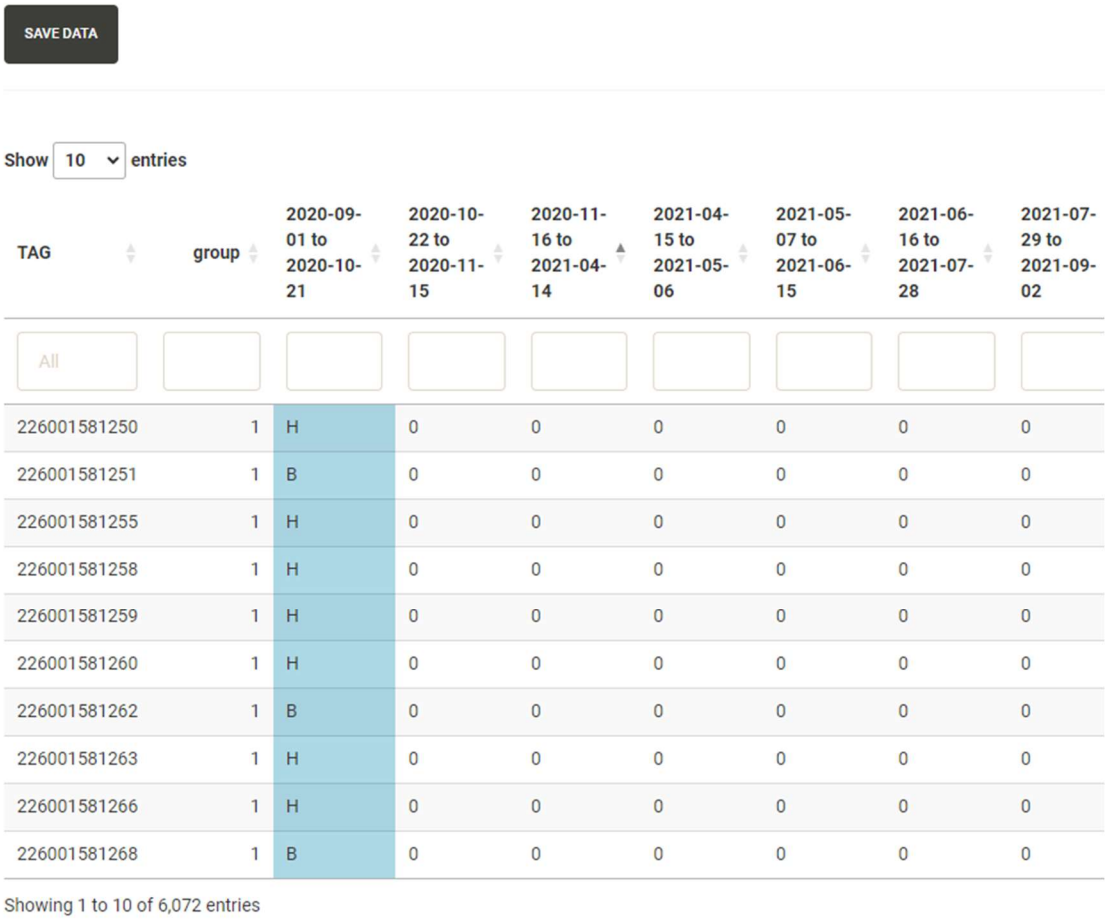
- This yields 17 instances of fish who have used this sequence.
- Interpret the data how you feel especially given how you can change the filters, but it's important to know what they are showing about the data. This would definitely suggest that fish are travelling upstream more now that the CRCC is built (17 instances vs 3 instances) but it's not exactly a straightforward comparison. When the Windy Gap Biomark antenna is consistently deployed, that may be a good way to level the comparison because then maybe we can see more clearly when fish come up to the dam, then decide which way of passage to use.

MARK States Tab

Displays data output from createMARKEncounterHistoriesFunction()

This is where MARK States are displayed, intended for download and input into Program MARK. This data is a static file generated in CreateFlatFilesRunscript.rmd. It is re-created every time the runscript is re-ran, typically when there is new data downloads.

- It is ran on all tags in the release file that aren't TGMs.
- Currently, there are no filters in the app or runscript to change the inputs: ie, you can't filter for "rainbow trout released at Dark Timber and Hitching Post in the Fall over 300mm" inputs and get that dataset. Since this code takes about 1.5 minutes to run and I feel like that functionality isn't super needed in the app, I've omitted it. But if we wanted/needed to filter the data and re-run with specific user-desired subsets, we could definitely do that.
- You can use the search boxes below each column to filter for specific tags, groups, states, etc



How to read

The data shows the last state that a fish was detected/released/recaptured in the specified time period (defined in WGFP Metadata.xlsx) across a Tag's history. If a fish is recaptured, that line ends as a -1 in the "Count" column, and a new line begins with that same tag number, beginning in the same state it ended in. Ghost tags or avian predation tags end in -1. A fish's history ends with 1 in the "Count" column. The “group” column denotes times a fish has multiple lines/has been recaptured. If a tag has not been recaptured, it will have just 1 "group" and 1 line in the data. If a tag/line is group 5, that means it has been recaptured 4 times. States other than 0 are shaded for easier viewing within the app. There are length, weight, and Species columns as well. The "Save data" icon in the top left allows you to save the data as a csv or rds file.

Requirements

The function that creates this data, createMARKEncounterHistories(), relies on having the following up to date files:

- A combined file of all types of detections, release, and recapture events on a daily level. This is created automatically in the CreateFlatFilesRunscript.rmd using function All_combined_events_function(), generating new data whenever the new downloads are incorporated
- The ghost tag file, manually created and living here: U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\RFID\Detections\GhostTags\
- Avian predation file, manually created (see QAQC -> Avian Predation section) and living here: U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\RFID\Detections\AvianPredation
- Time Periods file, created in the WGFP metadata file that lives here: "U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\RFID\Detections\WGFP Metadata.xlsx". This time periods file defines the start and end dates for each period and keeps track of the detection data used each period.
    - Needs to periodically be updated with correct dates/desired time periods. If the runscript is ran and there is antenna data that doesn't fall under a listed time period, there will be an explicit warning in the html that is rendered by the Runscript.

Movements Function took 0.37 minutes

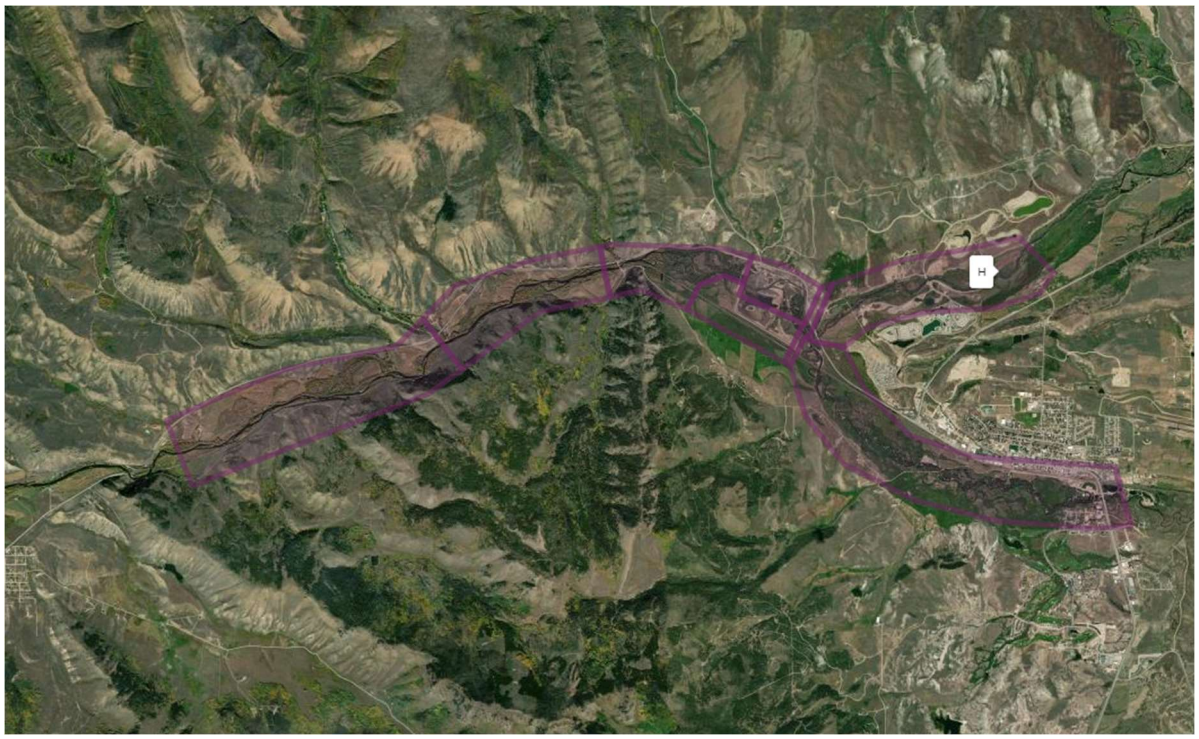
When assigning states for program MARK, some data don't have a time period assigned to them, date ranges 2025-04-16 to 2025-04-21. Add or adjust time periods in the csv.

Saving flat files for the app took 1.05 minutes

    - The time periods are also adjustable. You can use this sheet to modify the start and end dates of a period and this will be reflected in the app the next time the Runscript is compiled.

	periods	start date	end date	type of period	notes	Data available
2	1	9/1/2020	10/21/2020	fall	first release to first mobile runs	Electrofishing, mobile and stationary
3	2	10/22/2020	11/15/2020	late fall	after brown trout spawn	Stationary only
4	3	11/16/2020	4/14/2021	winter	mobile spring runs	Stationary (over winter) and mobile (at end)
5	4	4/15/2021	5/6/2021	spring	spring release events	Stationary and electrofishing (at end)
6	5	5/7/2021	6/15/2021	late spring	Back end of spawn	Stationary only
7	6	6/16/2021	7/28/2021	summer	3 summer days of mobile runs	Stationary and mobile (at end)
8	7	7/29/2021	9/2/2021	late summer	Fraser river and Kaibab Park	Stationary and electrofishing (Fraser only)
9	8	9/3/2021	10/20/2021	fall	second year release and mobile runs	Stationary, electrofishing (River Run), mobile (at end)
10	9	10/21/2021	11/15/2021	late fall	after brown trout spawn	Stationary only
11	10	11/16/2021	4/20/2022	winter	mobile spring runs	Stationary (over winter) and mobile (at end)
12	11	4/21/2022	5/5/2022	spring	spring release events	Stationary, electrofishing (3 sites on lower CO)
13	12	5/6/2022	6/15/2022	late spring	Ghostbuster at windy gap dam 5/18	Stationary Only
14	13	6/16/2022	7/31/2022	summer	stationary only	Stationary only
15	14	8/1/2022	8/31/2022	late summer	Fraser river and Kaibab Park	Stationary and electrofishing (Fraser only)
16	15	9/1/2022	10/19/2022	fall	third year release and mobile runs	Stationary, Electrofishing (upper C and WG dam), and mobile (at end)
17	16	10/20/2022	11/15/2022	late fall	after brown trout spawn	Stationary Only
18	17	11/16/2022	4/15/2023	winter	no mobile runs this period	Stationary Only
19	18	4/16/2023	5/6/2023	spring	no releases this period, high river	Stationary Only
20	19	5/7/2023	6/7/2023	late spring	lots of electrofishing at end on lower C	Stationary, Electrofishing (at end)
21	20	6/8/2023	7/26/2023	summer	mobile runs at end	Stationary, mobile (at end)
22	21	7/27/2023	9/6/2023	late summer	Fraser river and Kaibab Park	Stationary and electrofishing (Fraser only)
23	22	9/7/2023	10/4/2023	fall	fourth year release, no mobile runs. TGM release 9/14	Stationary and electrofishing (upper C and WG dam) at end
24	23	10/5/2023	11/15/2023	late fall	after brown trout spawn	Stationary Only
25	24	11/16/2023	4/15/2024	winter	no mobile runs this period	Stationay Only
26	25	4/16/2024	5/16/2024	spring	release and recap events on the lower C	Stationary and Electrofishing (lower C)
27	26	5/17/2024	6/15/2024	late spring	back end of spawn	Stationary Only
28	27	6/16/2024	7/26/2024	summer	no mobile runs this period	Stationary Only
29	28	7/27/2024	9/4/2024	late summer	Fraser river and Kaibab Park. Merganser released 8/8	Stationary and electrofishing (Fraser only)
30	29	9/5/2024	10/15/2024	fall	upper C and CRCC electrofishing, mobile runs	Stationay, electrofishing (upper C + CRCC), mobile runs (at the end)
31	30	10/16/2024	11/15/2024	late fall	after brown trout spawn	Stationary Only
32	31	11/16/2024	4/15/2025	winter	hasn't occurred yet ; not currently used	
33						

- States Polygon (gis/WGFP_States_2024.shp): defines the states by spatially joining with the detection data based on where a detection spatially intersects with the state (done automatically in CreateFlatFilesRunscript.rmd using function spatial_join_stations_detections()). Can be changed as needed. Viewable in the app in the "Daily Movements Map" tab as a layer in the layer control.



QAQC and Edge Cases Columns

There's a few columns in the data that aren't imperative to the analysis but help for QAQC purposes.

There are 4 columns at the end of the data: NeedsLogWeight, releaseAndGhostSamePeriod, releasedAndRecappedInSamePeriod, and multipleRecapsInPeriod. These help keep track of some edge cases that the code accounts for.

2024-11-16 to 2025-04-15	Count	Length	Weight	RBT	LOC	MTS	NeedsLogWeight	releaseAndGhostSamePeriod	releasedAndRecappedInSamePeriod	multipleRecapsInPeriod
							All	All	All	All
0	1	124	28	0	0	1	false	false	false	false
0	1	130	30	0	0	1	false	false	false	false
0	1	117	24	0	0	1	false	false	false	false
0	1	122	28	0	0	1	false	false	false	false
0	1	79	4	0	1	0	false	false	false	false
0	1	112	21	0	0	1	false	false	false	false
0	1	121	26	0	0	1	false	false	false	false
0	1	113	18	0	0	1	false	false	false	false
0	1	108	22	0	0	1	false	false	false	false
0	1	121	26	0	0	1	false	false	false	false

- NeedsLogWeight
    - If a fish doesn't have a weight listed, weights are listed as NA. This most commonly occurs when a fish is recaptured. Example: 230000142594. When this occurs, the tag will get flagged in “NeedsLogWeight” as a reminder to estimate this fish’s weight before analysis.
    - If the Release/Recap instance were in the same time period, check first to see if there was an original Release weight assigned to the fish if the release/recap were in the same time period, and use that.

- multipleRecapsInPeriod
    - If a tag has multiple recaps in the same time period, only the more recent recap instance is used, along with its length and weight. The recap before that (within the same time period) is removed in the data. Example: 230000143396.
- releasedAndRecappedInSamePeriod
    - If a fish is released then recapped in the same time period; the "recap" is used as the new "release", and its length/weight is used. If there is no weight in these instances, the original release weight is used. Example: 230000272087.

Note: Instances of multipleRecapsInPeriod and releasedAndRecappedInSamePeriod explain why the total rows in this data are not quite the same as the total rows in a combined "release/recapture" file, because it means some recaptures/releases have been removed to account for these cases.

- releaseAndGhostSamePeriod
    - If a ghost state originally occurs in the same time period as its release state, the ghost state is moved to the next time period, even if it wasn't explicitly detected in that time period. Example: 230000142981.
- To manually QAQC, I downloaded the csv from the app, saved as a .xslx, checked every 10th row of data in the resultant file and kept track here: U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\RFID\States\MARK States from app\MARKEncounterQAQC_20241216.xlsx

Daily Movements Map, Plots, and Data

Displays data of "movements" (dailyMovementsTable1 from Get_movements_function()), where a fish has changed position along the river.

"Movements" Defined

Movements are inferred by encounters of the same tag on different antennas (including mobile runs) or recaptures throughout the river. Each detection along the river is assigned a "station", breaking up the centerline of the stream into 10 m increments. The stations start at 0 m, starting about 150 m below the Sherriff Ranch Fry Site. They extend to 13480 m on the Upper Colorado River, where the mobile run begins, and to 22850 m (10120 m at confluence + 12730 m from confluence to Fraser River canyon) on the Fraser River (5/23/2024).

These stations, which are made in GIS from stream centerlines, are joined by coordinates to a dataset of all relevant detections (first and last detections of the day and unique detections in between for each unique UTMs and TAG), using the Spatial_Join_function. The resulting dataset is passed back to a larger dataset of all detections to assign stations to all UTMs. From there, the data is condensed down to a dataset of daily fish detections, keeping daily movements. See Get_Movements_Function() for more details.

The distance moved between days is calculated for each fish when a new detection is recorded, calculated by taking the difference between stations.

Types of Movement

- Upstream movement: positive difference between stations (IE new detection station = 8000, previous detection station 6000, so 8000-6000 > 0; upstream movement).
- Downstream movement: negative difference between current station and previous station (new station = 6000 m, old station is 8000 m, -2000 <0; negative movement).
- No movement: the previous station and current station are the same (IE new station = 8000, previous station = 8000. Difference = 0, no movement). Consecutive detections at the same stationary antennas spanning more than one day are registered as no movement.
- Changed Rivers: Where a fish has moved from the Fraser River to the Upper Colorado (above the confluence) or vice versa. (IE a fish was detected on a mobile run on the upper Colorado but its release site was Fraser River Ranch. Although this is no upstream/downstream movement, the total distance moved in one example is

1910 meters, since that is the summed distances of Fraser River Ranch to the confluence + confluence to gps coordinates where the fish was detected on the Upper Colorado mobile run).

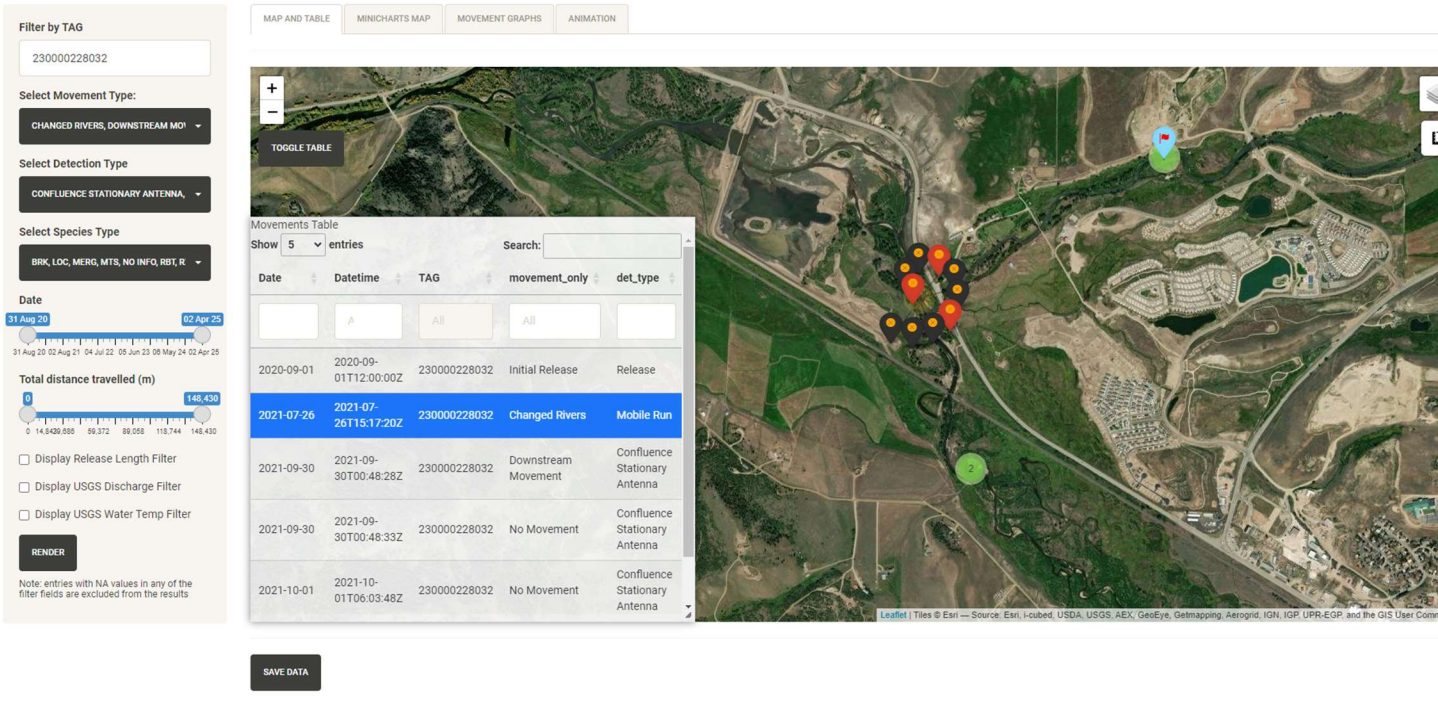
- Initial Release: Event when the fish was released. Used to establish the first station on the river that a fish was seen.

The absolute value of total distance moved is summed for each fish and displayed in the movements tab under "sum_dist". It also seen in Encounter Release Histories Wide created in the Ind_tag_enc_hist_summary_wide function.

- Filters in the sidebar control data displayed in the map or plots
    - USGS Data is combined with movements on a *daily* level, contrasted with events in All Encounter Histories where PTdata and USGS data is associated with an event within 13 hours for USGS data and 1 hour for PTdata
    - Like the Environmental Filters and USGS data filters in Encounter Histories Tab, using the filters for Release length, USGS WaterTemp, and USGS Discharge will automatically filter out rows that don't have that data associated with the movements. All the Tiger Musky in the release file have been changed to 0 lengths and weights so that they show up on the map/table/plots if the release length filter is on

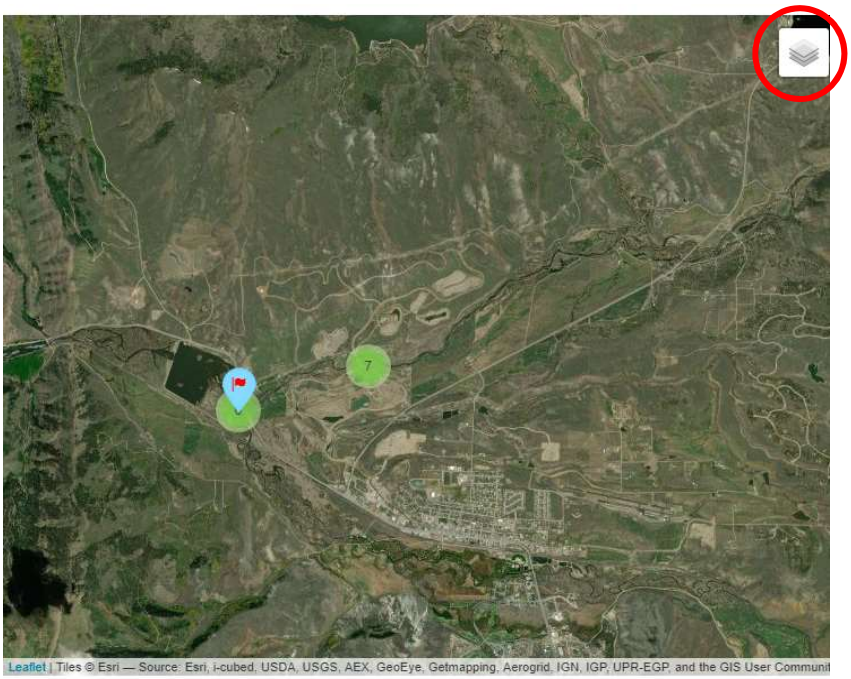
Map and Table

- This tab can be used to answer questions like "what time of year are fish moving?" and is the easiest/funnest way to examine individual tag histories. A method I frequently use is to find fish that have moved a long distance or have weird daily states recorded, copy and paste the tag, and plug it into this tab. Or find an event on the map, click on it, copy the tag number, and plug that in.
- Filters "Release Length", "USGS Discharge", "USGS Temperature" are optional to apply because using them can result in unintentional omitting of data. Many rows have NA values for these fields and when these filters are applied, NA rows automatically get filtered out. For example, some fish don't have Lengths/Weights, and in winter time, there isn't always a reading for Discharge or Temperature.
- Map is interactive, where you can click and hover to get more info about a movement. Clicking on an event in the map will navigate to and highlight the selected event in the table, and vice versa where a blue icon with a red flag will appear in the map when an event in the table is selected

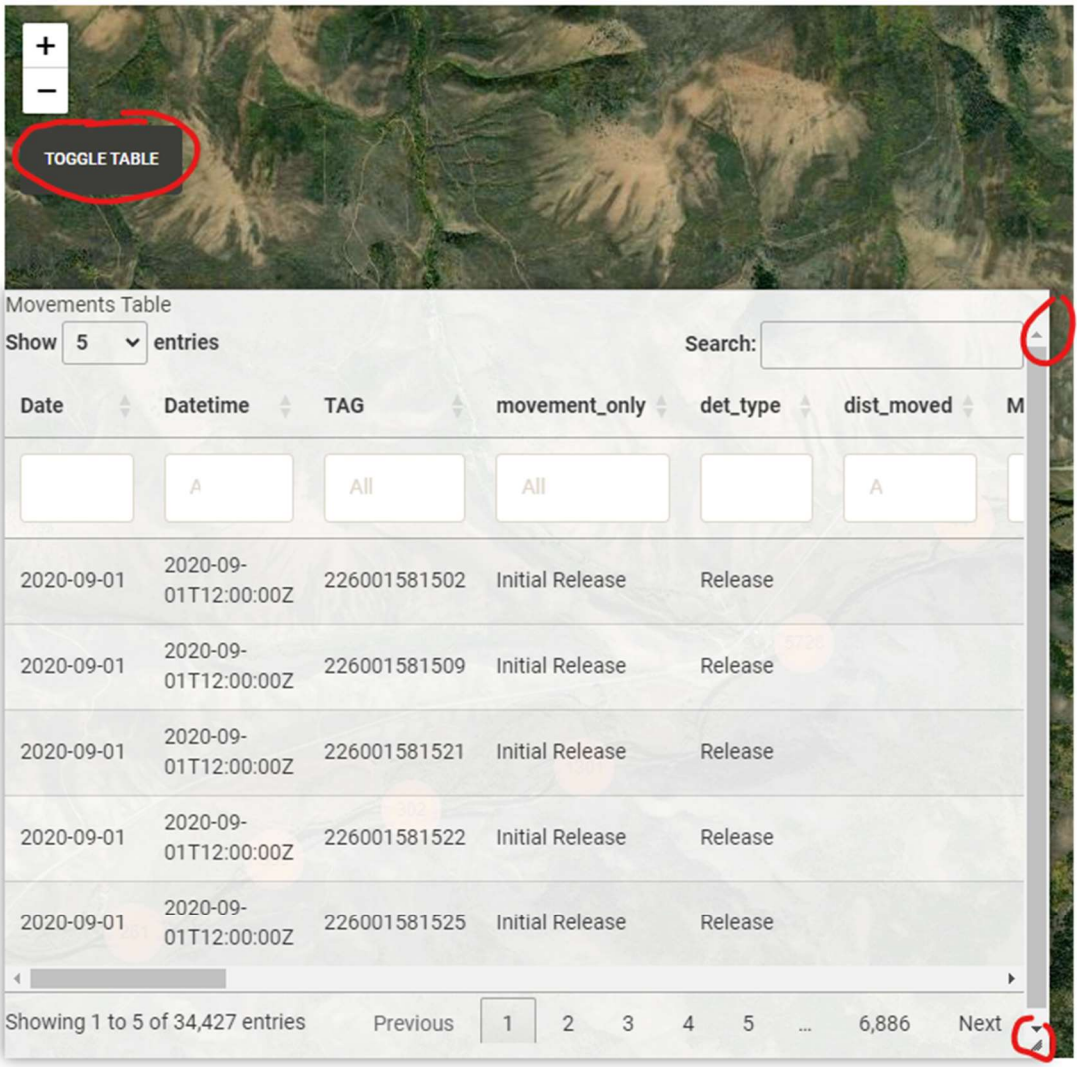


- Layer control is available in top right, controlling detections, antennas, Release Sites, Stream centerlines, Stations that were used to make these movements, Mobile Reaches, and States (used to define States used in MARK

States tab). Note that loading the Stations takes about 10 seconds and can also slow map performance.

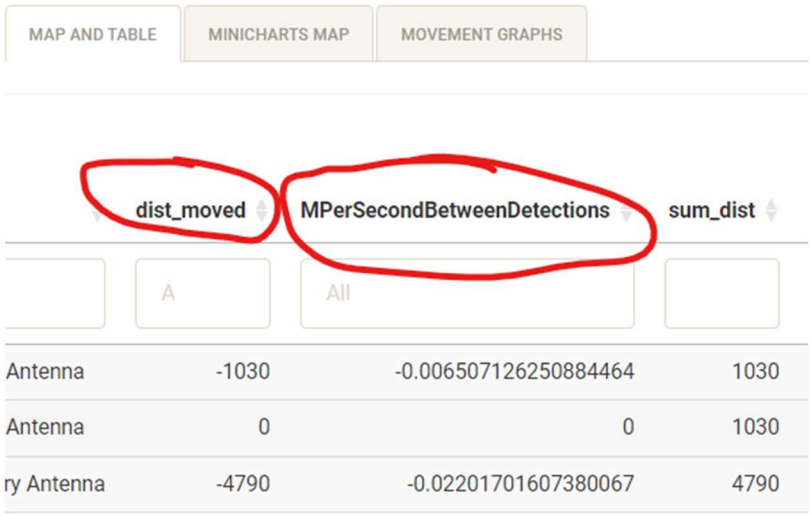


- You can toggle the table to get it to disappear, as well as resize it by dragging the sides. Displayed data is downloadable with "Save Data"



- Measurement tool exists below layer control and can be used to discern approximate distances

- Table can be used to help find suspect avian predation tags by clicking/sorting on the dist_moved and MetersPerSecondBetweenDetections columns. Outliers in these fields are subset and used in "QAQC" -> "Avian Predation" -> "Movements" table

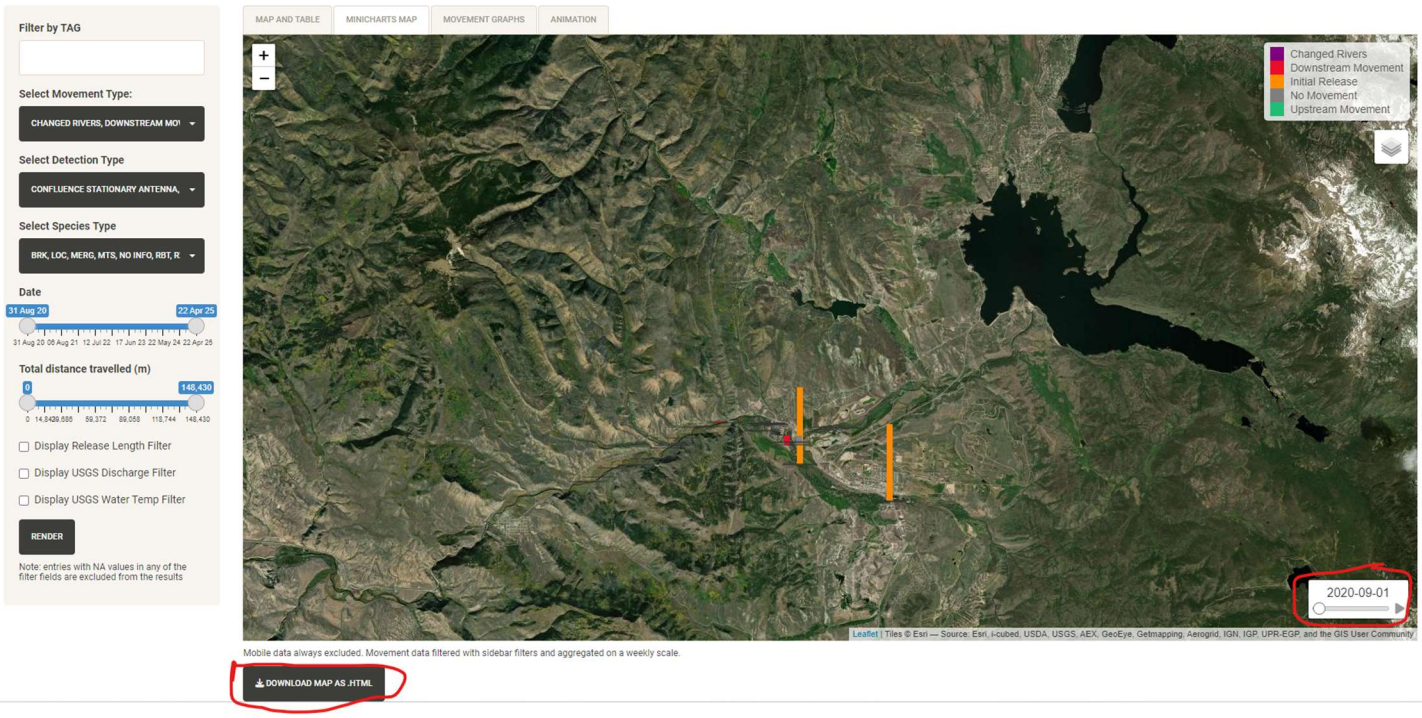


Note: left sidebar filters automatically filter out data with NAs in fields displayed. As of May 2, 2025, there are 5 entries that are automatically filtered out because they are not assigned movements, all fish that were released/recapped 2 days from each other in early May 2021.

Minicharts Map

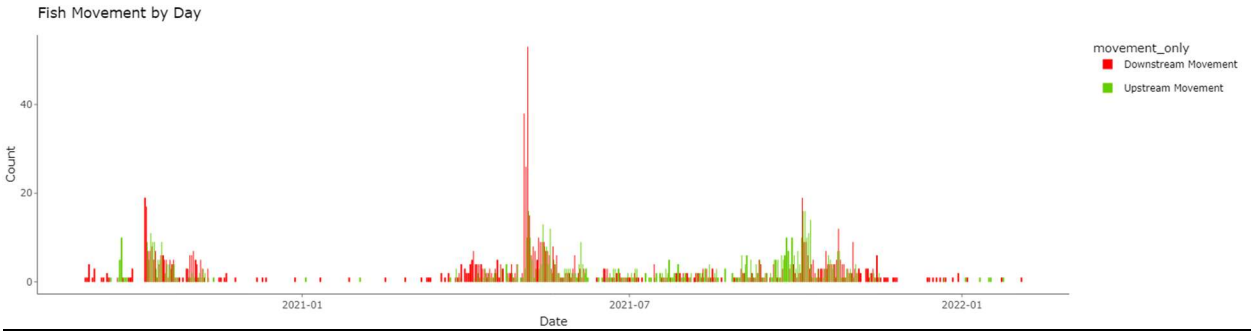
Minicharts are an animation option to spatially display histograms at each antenna by movement type.

- Use the left sidebar filters to select desired data.
- Data are automatically grouped by week and mobile data is always excluded
- Click the play button in the bottom right to go through the animation.
- Save animation as an interactive, shareable .html web page using the "Save Map as HTML" button.

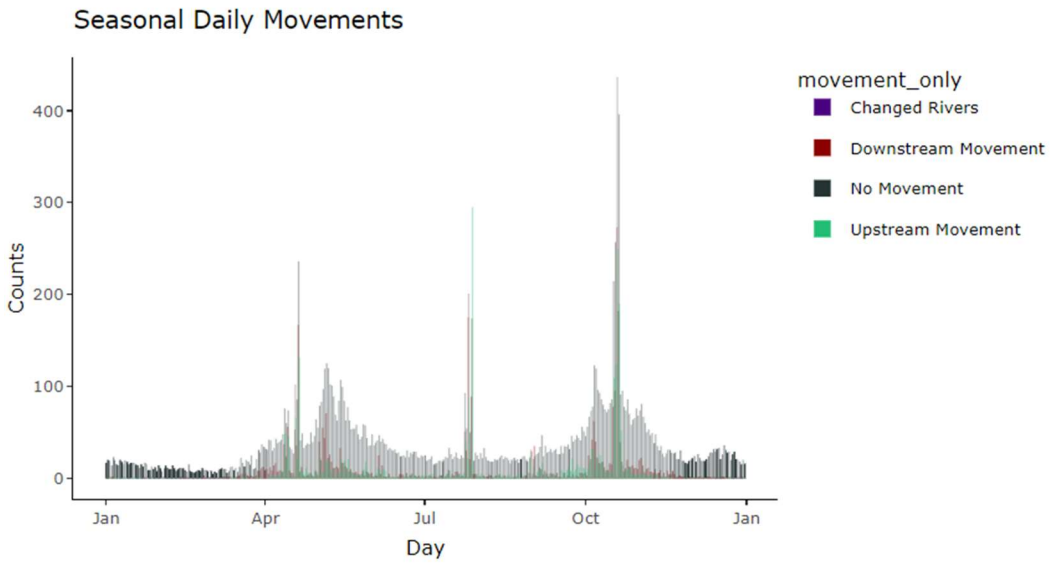


Movement Graphs

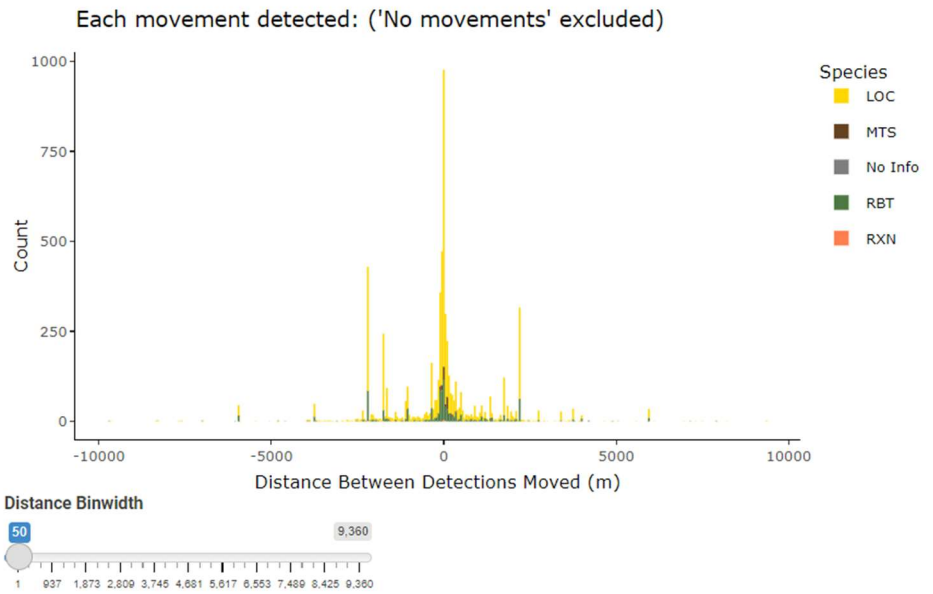
- The plots are also controlled by the sidebar filters
- See "types of movements" above for movement definitions
- Fish Movement by Day: simple histogram representation of movements across time, colored by movement type
    - Note: data can be misleading if mobile detections aren't filtered out, as these cause big spikes in movement activity the day of the mobile run. Activity is also generally higher after release events as well.



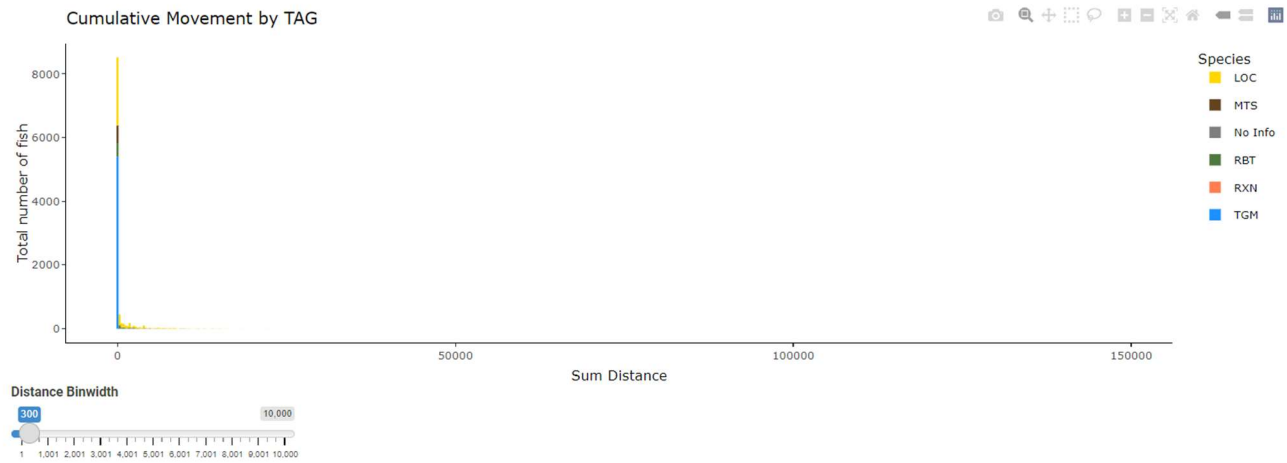
- Seasonal Daily Movements: Same plot as above but grouped by day across the years, which is helpful in seeing seasonal movement trends (big spikes here are probably mobile runs)



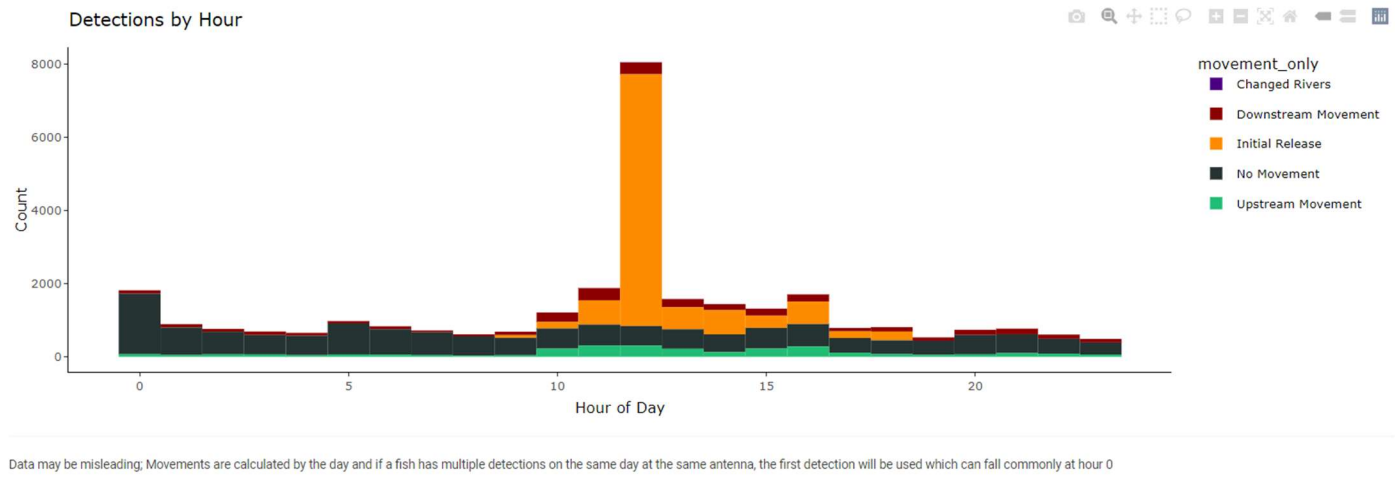
- Individual Movements: showing all the times a fish has moved (upstream, downstream or changed rivers) upstream and changed rivers are positive integers, downstream is negative. Note the spikes around 2200m due to fish travelling between red barn and hitching post, a distance of 2190m according to the stations.



- Cumulative Movement: the amount each fish has moved in absolute distance, in meters, in total over the course of its life.

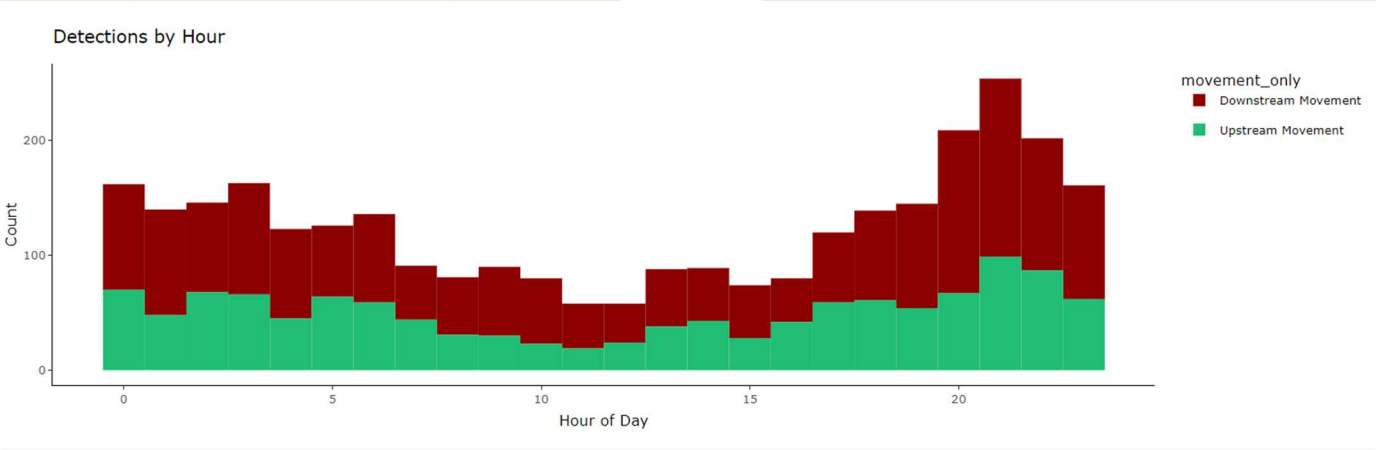


- Movement by hour: displays how many movements have been recorded during each hour of the day. Can be skewed towards hour 0 (12:00 AM) especially if a fish is hanging out on an antenna, since the movements df take the first and last activity of a fish during the day in addition to unique events in between.



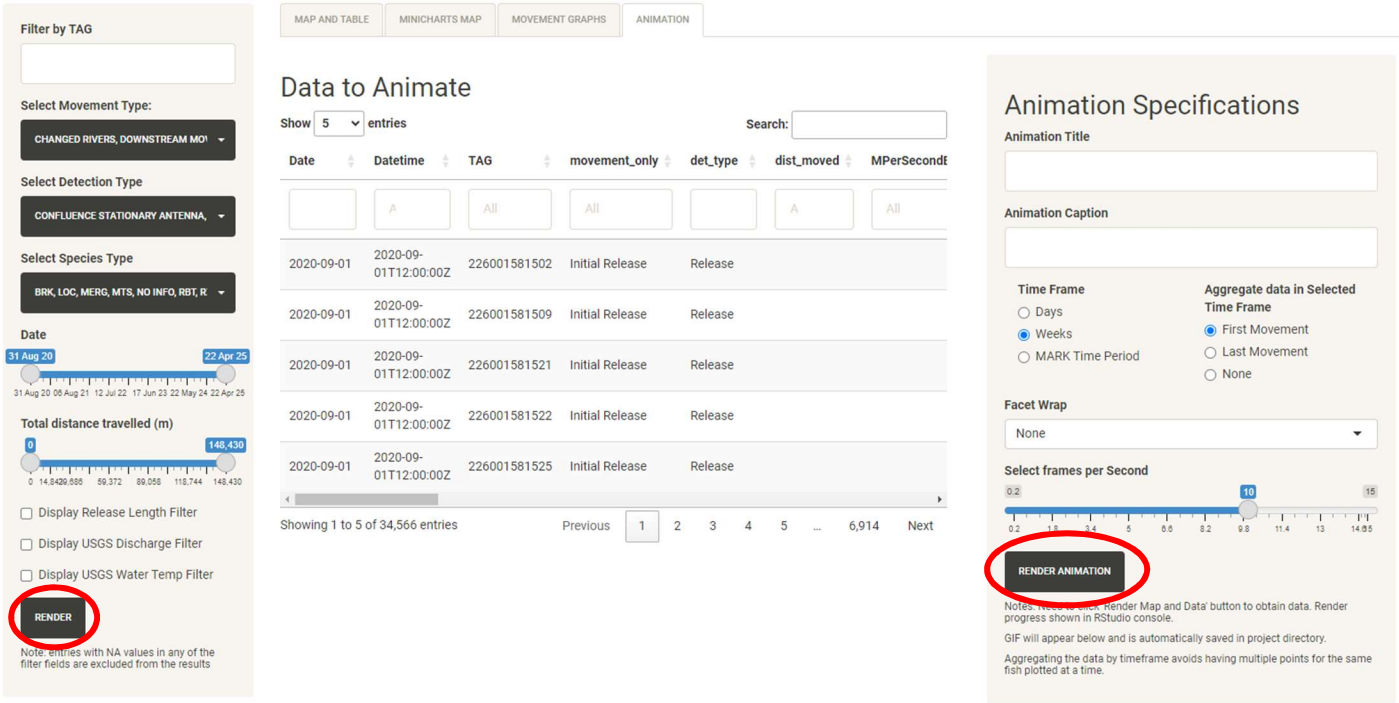
  - If "No Movements" and "initial release" are filtered out from movement type, and "mobile detections", "release" and "recaptures" are filtered from movement types, you get movements registered by

antennas in the river from fish that are not hanging out on antennas for weeks at a time. This potentially helps to see "true" movement activity of fish throughout the day.



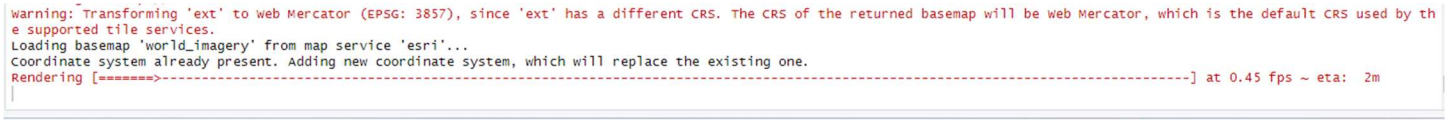
Animation

This tab renders a .gif file from filtered movement data.



- First filter desired data from the left sidebar and click "Render". "Data to Animate" shows filtered movement data to be animated. This is the same table/data as displayed under the "Map and Table" tab.
- Enter desired Animation Specifications, then click "Render Animation"
    - Time frame will group data based on days, weeks, or MARK time period. The shorter the time period, the longer the animation will take to render. For example, if "days" are selected, each day will be 1 frame
    - Aggregate data is necessary because the same fish can have a different movement in the specified time frame. If the timeframe is "weeks", and a fish had an upstream and downstream movement in that time, it will appear as two dots in the same week frame (which may be desirable). "First movement" will use the first movement within that week, "Last movement" uses the last movement within that time frame.
    - Facet wrap splits the data by the selected category (species or release site are current options).

    - Less selected frames per second will render a slower gif, and vice versa for more frames per second
- There's no in-app loading progress, but within the RStudio console you will see some text denoting progress. Depending on selected data, it may take minutes to hours to render the animation but eventually a progress bar will show up in the console.



- Animations will show up below the data table in the app and are automatically saved in the project working directory as "WindyGapFishMovements.gif". If you make a new animation, this will overwrite the one in the current working directory unless you copy it to another directory and/or change the filename before rendering another animation. Sometimes the .gif appears low in the window and you need to scroll.

- Some errors may occur. If the animation isn't rendering, check the "known bugs" animation section of the how-to and see if troubleshooting there helps.

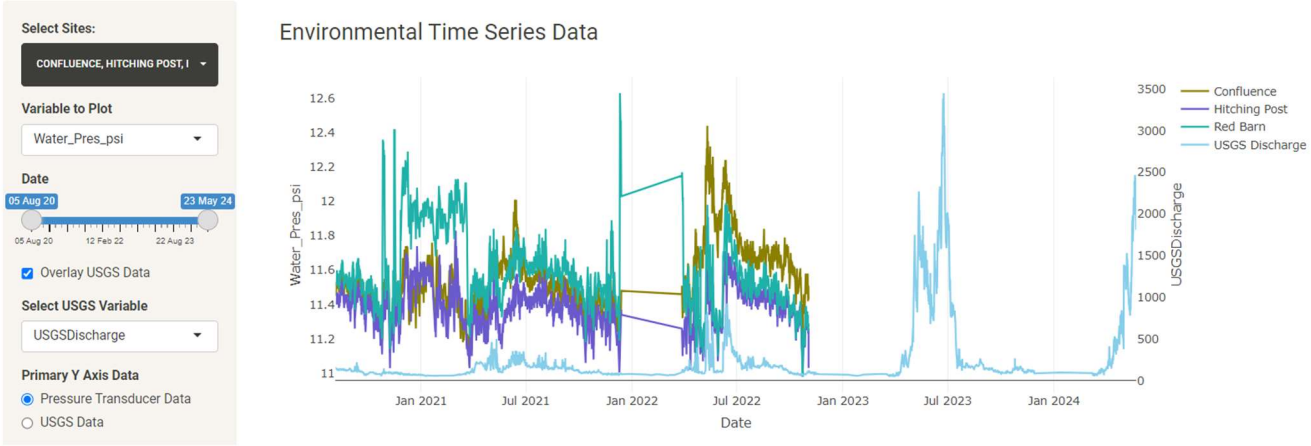
Pressure Transducer, USGS and Detection Distance

This tab shows graphical representation and overlays of data from Pressure Transducers, USGS and Detection Distance from Site Visits.

- Water_level_NoIce is the best variable for water level. Ice readings/observations are displayed as 0.
- USGS data is all from the gauge at Hitching Post. It can be misleading if you select a site and see discharge, but it's all from the same gauge.

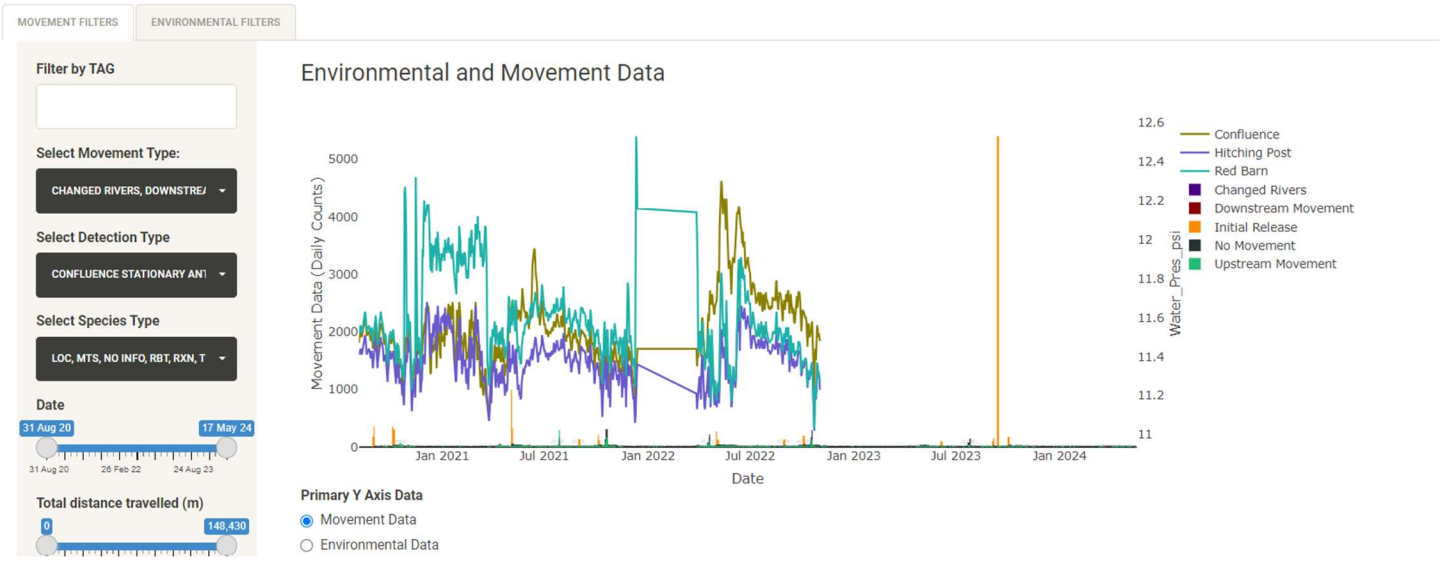
Time Series

PT data with the option to overlay USGS data. You can select variables to plot and which Y axis to display it on. If you get an "Error: Object" warning, it's because the date range does not encapsulate any of the Pressure Transducer data



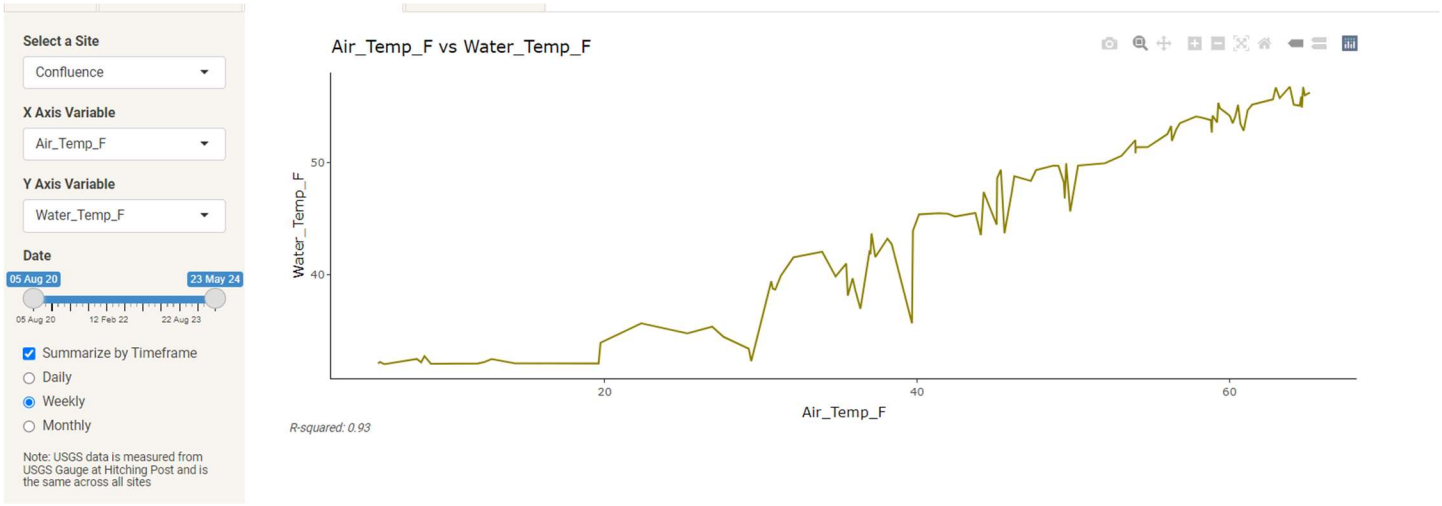
Movements Overlay

PT/USGS data overlaid with the same movements displayed and used in the movements tab. All the same filters are available for movements, and the graph used is the Fish Movement by Day histogram of movements across time. Environmental data filters are accessed in the top left tab. To get the dates to line up, you'll have to change the date ranges on both the environmental data filters and movement filters.



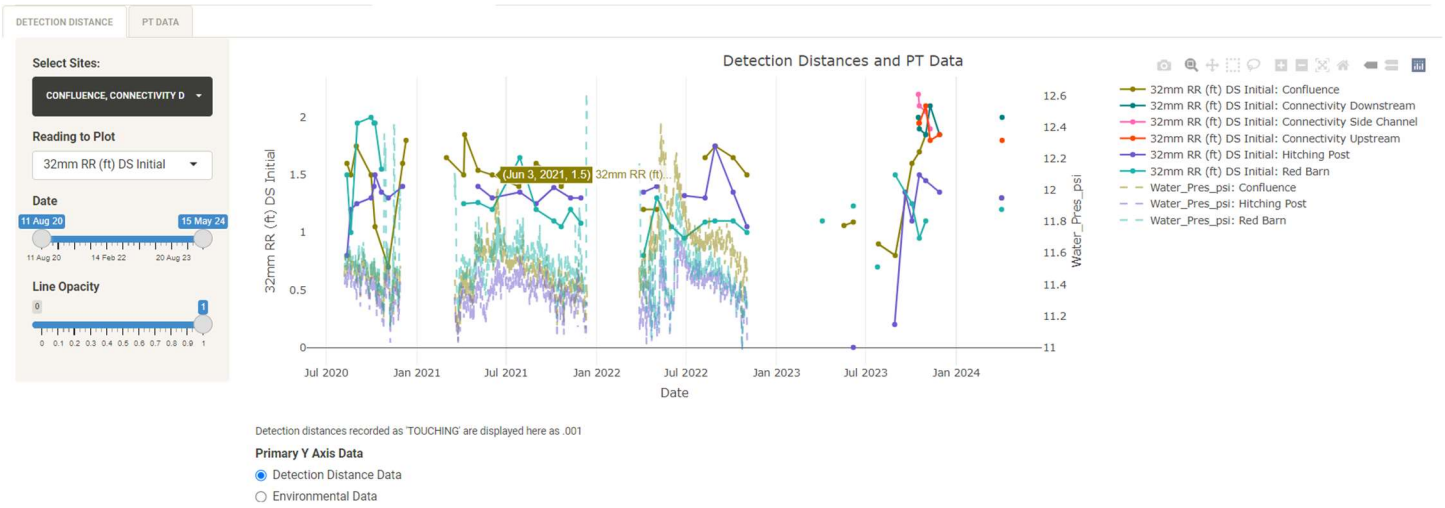
Variable Correlations

Plot PT/USGS variables against each other. "Summarize by timeframe" is an option used to group data by specified timeframe and can get rid of daily fluctuations. R-squared value in the bottom left graph corner is calculated and found with this function in R: rquared = summary(lm(variableX ~ variableY))$r.squared



Detection Distance

Detection distance from site visits of Biomark and Stationary Sites with PT data overlaid. You can select which reading you'd like to plot from the sidebar. You can change the opacity of the lines for preferred representation as well.



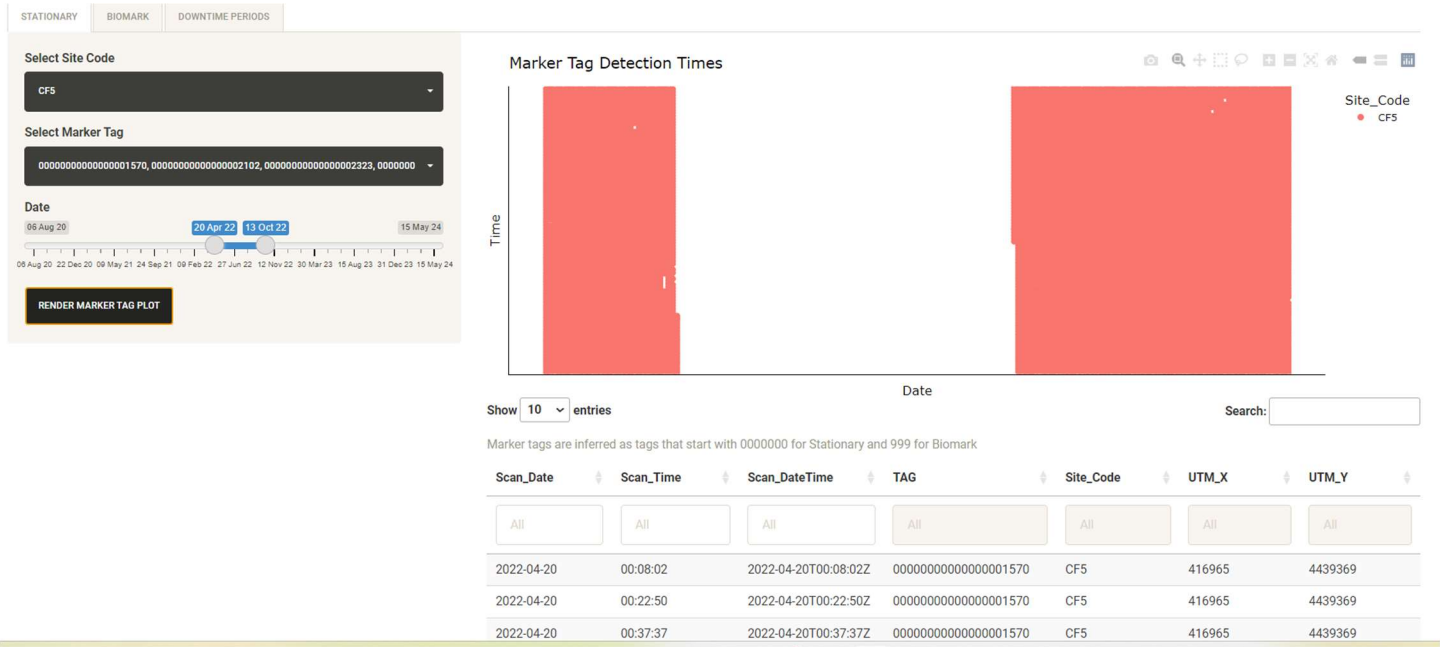
QAQC Tab

Used to help verify data.

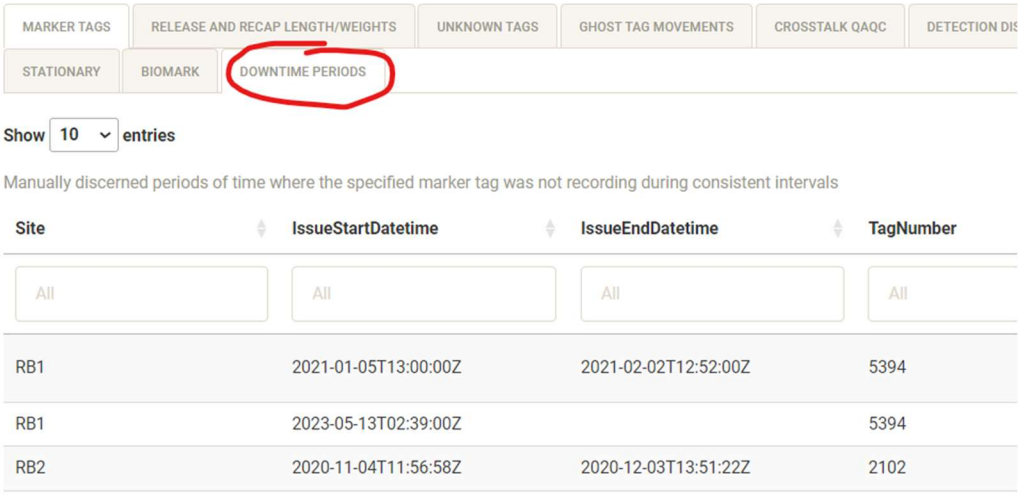
Marker Tags

Stationary and Biomark Marker Tag Data set from All_Combined_events_function

- Helpful for finding periods of inconsistent marker tag detections, indicating antenna was not functioning properly during these time.
- In this case, there was a gap: closer examination shows there were not detections between May 21, 2022 and August 9, 2022 at CF5



- These found gaps are manually recorded in the "Marker Tag Issues" tab in "WGFP Metadata.xlsx" and are displayed in the "Downtime Periods" tab here



- This is a large dataset to plot with a scatterplot, so it will run quickest when plotting a smaller date range with only one marker tag or site selected.
    - A smaller date range will get better resolution in the plot, making it easier to see downtime periods with the tag.
    - What I do to find these periods is examine a small date range of about 2-8 months, select just one site, then select all marker tags and render the plot.
- If the ARR column is brought in incorrectly from stationary data (which can sometimes happen if the stationary ARR column gets corrupted, rarely happens) then there will be an abundance of detections in the 12:00-13:00 range.

Summarized Marker Tag data

- Raw counts of marker tag detections across selected sites for the duration of the study.
- Helpful for seeing which tags have been deployed at which sites and which sites may have had more downtime periods than others

Summarized Marker tag Data for Selected Tags and Sites

Show 10 entries

Date filter does not apply to this table

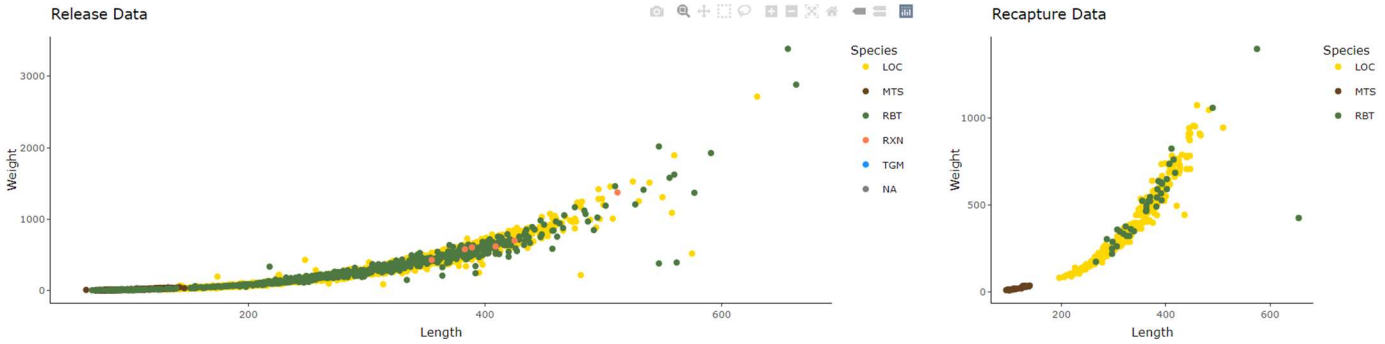
Site_Code	TAG	totalDetectionsSinceProjectInception
All	All	All
CF5	00000000000000001570	100927
CF5	00000000000000004948	81603

Showing 1 to 2 of 2 entries Previous 1 Next

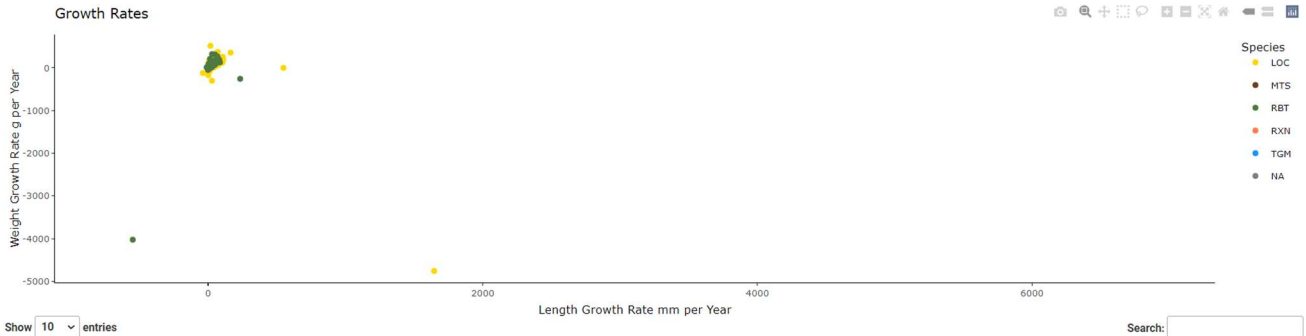
SAVE DATA

Release and Recap Lengths/Weights, Growth Rates

- The first plots help ensure that there were no length/weight typos in the release and recapture files.



- The Growth Rates plot shows the rate of growth per year between each release/recapture instance of a fish. If a fish has been recaptured multiple times, it will have multiple points on this graph.



Show 10 entries Search:

Data from each time a fish was recaptured

Species	Median Length Growth Rate (g per year)	Median Weight Growth Rate (mm per year)	Mean Length Growth Rate (g per year)	Mean Weight Growth Rate (mm per year)
All	All	All	All	All
LOC	43.38	95.72	66.8	91.98
MTS	7.53	4.01	7.84	3.21
RBT	40.89	124.37	29.68	-0.36
RXN				
TGM				

Showing 1 to 6 of 6 entries Previous 1 Next

Unknown Tags

- List of Tags without release info but started with 900_ initially and have detections on some sort of antenna
- Display of Enc_hist_wide_summary_function output “Unknown Tags”

Show 200 entries

Tags that initially started with 900_ but are not marker tags, test tags, or part of the release file.

TAG	Length	Weight	RB1_n	RB2_n	HI
All					
00000495240232787684					
10240123810058196921					
450115000072106					
00000247620116478391			1		

Ghost Tag Movements

A df made from the "Ghost Tags" csv input. This shows registered ghost tags that have moved >0m after their assigned ghost date. This is helpful in revising the ghost tags file if one erroneously turns out to be a fish.

- Table is automatically sorted by "total_distmovedAfterGhostDate" so the most problematic-looking fish are first displayed
- antennasDetectedAfterGhostDate is a vector of all antennas that have detected the tag after the ghost date. If the tag was identified as a ghost tag somewhere near upstream of a stationary antenna, it isn't uncommon to see the tag be detected on that antenna during a high flow event
    - To get more context, you can check USGS discharge in the "Pressure Transducer, USGS, and Detection Distance" tab, find the discharge associated with that specific detection within 13 hours in "Encounter Histories – All Encounter Histories, or find the daily discharge associated with each daily movement in "Daily Movements" tab – Map and Table – "USGSDischargeDaily" column
- If a fish is moving upstream significantly after its ghost date, it might be worth investigating. This is seen by sorting/clicking the "maxUpstreamDistMovedAfterGhost" column.
- Movements of 10-50m ish between mobile runs can typically be attributed to GPS variation
- Runoff events can significantly move tags downstream, sometimes across antennas

Ghost Tags To Double Check: each of these tags has moved > 0m since its assigned Ghost Date.

TagID	GhostDate	antennasDetectedAfterGhostDate	total_distmovedAfterGhostDate	maxUpstreamDistMovedAfterGhost	Notes
All	All	All	All	All	All
230000272949	2021-07-28	Mobile Run	230	0	Likely Ghost; Big fish location and probably date of 7-28-21); ghos 26-23
230000142602	2021-07-28	Mobile Run	220	20	confirmed using encc

Avian Predation

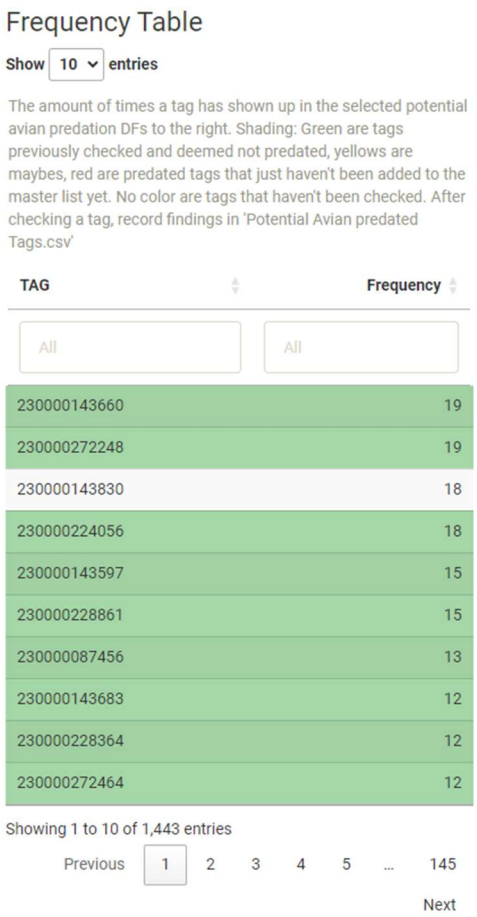
- It's important to keep a running list of tags that have been predated by birds in order to be able to filter them from the data later as needed. This tab is meant to organize methods used to flag potential avian-predated tags and keep track of tags that have already been checked.
- Workflow:
    - copy potential predated tag from one of the tables in the tab (each dataframe explained below)
    - examine that tag's encounter history in the "Encounter Histories" and/or Movements tab
        - See "Encounter Histories" -> Examples -> "Seeing a Fish's Full Encounter History/Investigate Individual Tag for Avian Predation"

    - Record the tag in "U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\RFID\Detections\AvianPredation\Potential Avian Predated tags.xlsx" and also in "U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\RFID\Detections\AvianPredation\WGFP_AvianPredation.xlsx" if confirmed a predated tag
    - Repeat for all tags in question
    - Save above .xlsx files as .csvs and copy .csvs over to data directory in the app
    - Re-run CreateFlatFilesRunscript to get most up-to-date shading and data
- Tags confirmed by predation through encounter histories are listed in "WGFP_AvianPredation.csv". Tags checked already are listed in "Potential Avian Predated tags.csv".
- Shading for each dataframe: Green signifies tags previously checked and deemed not predated with a "No" in the "Opinion" column in "Potential Avian Predated tags.csv", yellows are maybes that don't have yes or no in the "Opinion" column, red are predated tags marked with "Yes" in the "Opinion" column in "Potential Avian Predated tags.csv" that *haven't* been added to "WGFP_AvianPredation.csv" yet. No shading signifies tags that haven't been checked and recorded in "Potential Avian Predated tags.csv".

Tables

Frequency table

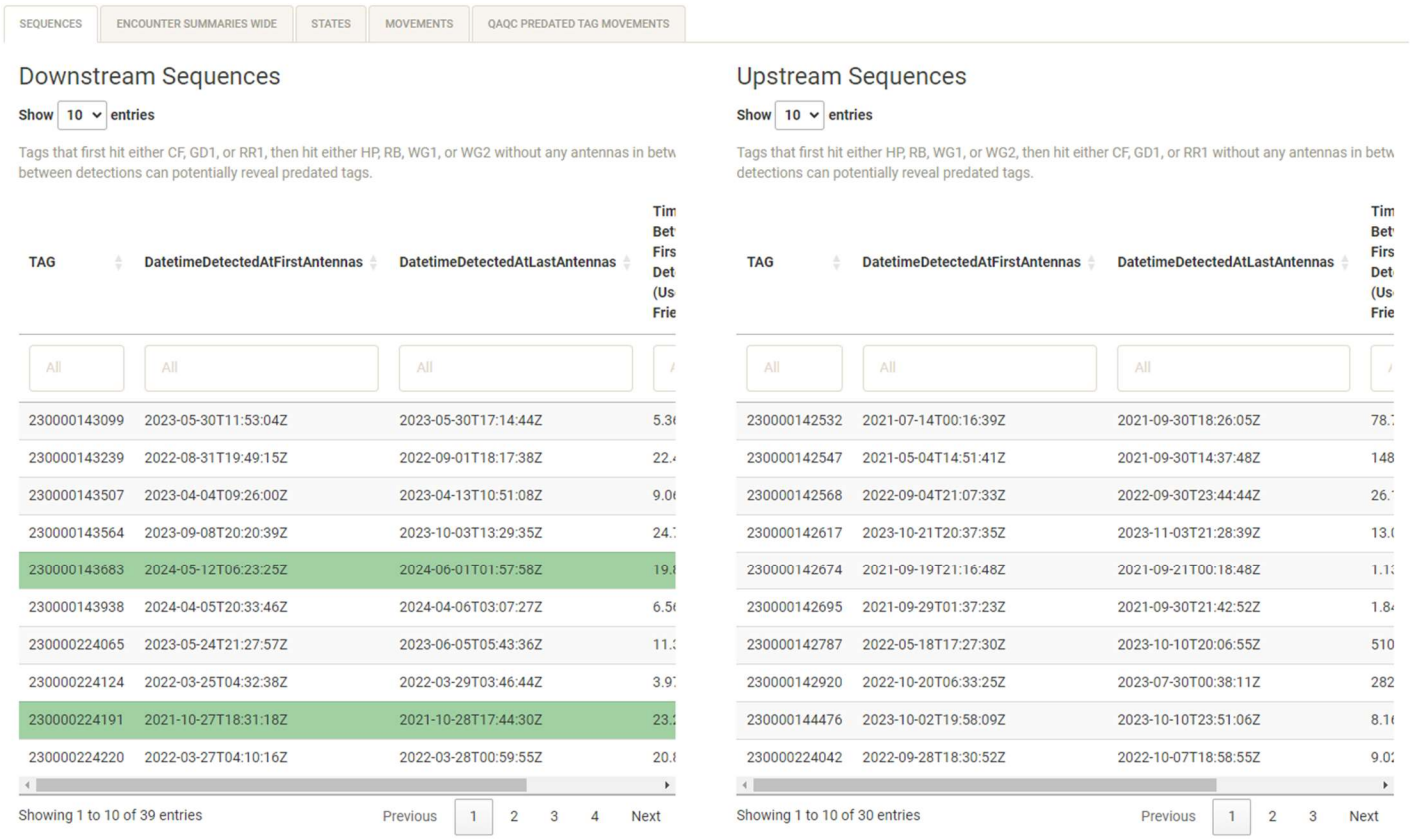
Will always appear on the left of the screen on the avian predation tab and keeps track of how often a tag has appeared, in total, in all of the flagging methods in the tables to the right. The best place to start looking for potential predated tags.



Sequences

Created using functions from the "Encounter Histories" -> "Sequences" tab, these are specific sequences that it wouldn't always necessarily make sense for a fish to hit. See "Sequences" section for more info on how these dataframes are made.

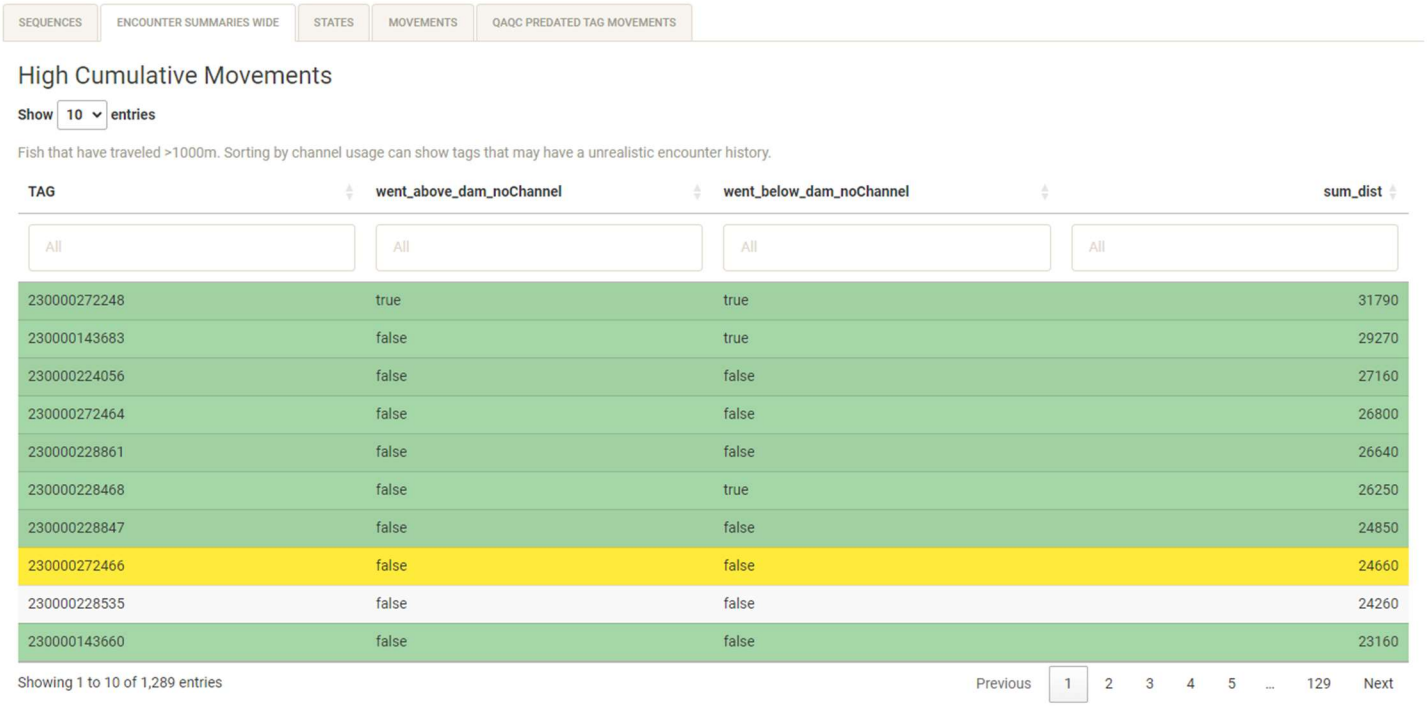
- Downstream Sequences table lists tags that first hit either CF, GD1, or RR1, then hit either HP, RB, WG1, or WG2 without any antennas in between, indicating they did not use the connectivity channel to move past the dam.
- Upstream Sequences are tags that first hit either HP, RB, WG1, or WG2, then hit either CF, GD1, or RR1 without any antennas in between, again indicating that they did not use the connectivity channel to move past the dam.
- Sometimes these sequences happen organically, which is why it's important to manually check the tag's encounter history before adding it to the avian predation list. Sorting by the time between detections can be helpful here to start the search, because while a fish may organically move past the dam upstream or downstream (especially downstream), if a fish flies past the dam in a non-realistic timeframe, it's a good indication that predation has occurred, rather than a natural movement.



Encounter Summaries Wide

This is a subset from "Encounter Histories" -> "Encounter Histories Summaries Wide" tab for fish that have travelled > 1000m in their lifetime, as represented by the column sum_dist.

- The binary columns "went_above_damNoChannel" and "went_below_dam_noChannel" are included to help sort, telling if a fish bypassed the dam without hitting connectivity channel antennas.
- Tags that move high distance and make multiple trips through the reservoir without hitting connectivity channel antennas are worth investigating to see if the tag has been predated.



SEQUENCES | ENCOUNTER SUMMARIES WIDE | STATES | MOVEMENTS | QAQC PREDATED TAG MOVEMENTS

High Cumulative Movements

Show 10 entries

Fish that have traveled >1000m. Sorting by channel usage can show tags that may have a unrealistic encounter history.

TAG	went_above_dam_noChannel	went_below_dam_noChannel	sum_dist
All	All	All	All
230000272248	true	true	31790
230000143683	false	true	29270
230000224056	false	false	27160
230000272464	false	false	26800
230000228861	false	false	26640
230000228468	false	true	26250
230000228847	false	false	24850
230000272466	false	false	24660
230000228535	false	false	24260
230000143660	false	false	23160

Showing 1 to 10 of 1,289 entries

Previous 1 2 3 4 5 ... 129 Next

States

A similar table as the distance table above, this time defined by MARK states across a week (High Weekly Activity) or an entire tag's history (High Cumulative Activity). See the MARK States section for more info how MARK States are defined and how this table is made.

- High Weekly Activity shows tags with > 6 state changes in a week or > 4 weekly unique events.
  - Weeks_since is the number of weeks past the beginning of the study
  - allWeeklyStates shows a fish's whole state history for that week
  - weekly_unique_events is how many different antennas a fish hit during that week.
- High Cumulative Activity shows tags NOT in High Weekly Activity with >6 total state changes across its entire encounter history
  - condensedAllStates is a tag's overall state history
  - ChannelSummary and subsequent columns are summary columns taken from "encounter histories summaries wide"

SEQUENCES | ENCOUNTER SUMMARIES WIDE | STATES | MOVEMENTS | QAQC PREDATED TAG MOVEMENTS

High Weekly Activity

Show 10 entries

Tags with > 2 state changes in a week or > 4 weekly unique events. Very active fish on a weekly basis.

TAG	Date	weeks_since	State	allWeeklyStates	weekly_uniqu
All		All		All	All
230000228313	2020-09-12	1	ECB	EEEEECCCBB	
230000228706	2020-10-06	5	CB	CCCBB	
230000228708	2020-10-06	5	CB	CCCBB	
230000228782	2020-10-06	5	CB	CCCCCBB	
230000228738	2020-10-07	5	ABC	ABBCCCC	
230000228923	2020-10-07	5	ABC	ABBCCCC	
230000228929	2020-10-07	5	ABC	ABBCCCC	
230000228972	2020-10-07	5	ABC	ABBBCCCC	
230000228837	2020-10-07	5	ABC	ABBBCCC	
230000229053	2020-10-07	5	ABC	ABBCC	

Showing 1 to 10 of 152 entries Previous 1 2 3 4 5 ... 16 Next

High Cumulative Activity

Show 10 entries

Tags >2 total state changes across its entire encounter history. Very active fish on an overall basis.

TAG	condensedAllStates	channelSummary	went_above_dam_noChannel
All	All	All	All
230000228032	FHEFEHE	Didn't Use Channel	false
230000228419	HEFECBC	Didn't Use Channel	false
230000228468	HECBCBCDCBC	Used Connectivity Channel	false
230000228896	BCBCBCB	Didn't Use Channel	false
230000228927	ABCBCBAB	Didn't Use Channel	false
230000229027	ABCBABC	Didn't Use Channel	false
230000142592	BCBCBCB	Didn't Use Channel	false
230000272194	BCBCBCB	Didn't Use Channel	false
230000272332	ABABCBA	Didn't Use Channel	false
230000272651	ABCBABC	Didn't Use Channel	false

Showing 1 to 10 of 21 entries Previous 1 2 3 Next

Movements

Displays tags with large movements and fast movements. These are subsets from the "Movements" tab, see that section for how these tables are made and how movements are defined.

- Large Movements shows tags and instances that have moved > 3700m on one movement. For reference, HP to CF is 3760m. A movement larger than this might imply a skipped antenna, potentially a bird flying over the river upstream or downstream
- Fast Movements show the top 5% of individual tag movements by speed (meters per second) between detections) upstream or downstream. Mobile run detections often deviate by a few meters, resulting in "fast" movements which can typically be ignored.

Large Movements

Show 10 entries

Tags that have moved > 3700m (fyi HP to CF is 3760m) on one movement.

Date	Datetime	TAG	dist_moved	det_type	Event	ET
	A	All	A			
2024-10-10	2024-10-10T16:31:18Z	230000087647	-9720	Hitching Post Stationary Antenna	HP3	
2023-04-22	2023-04-22T06:42:22Z	230000228178	-9660	Hitching Post Stationary Antenna	HP4	
2024-10-12	2024-10-12T07:13:34Z	230000087597	8740	River Run Biomark Antenna	RR1	
2025-03-29	2025-03-29T10:49:43Z	230000087858	-8300	Red Barn Stationary Antenna	RB1	
2025-03-31	2025-03-31T10:05:17Z	230000087862	-8300	Red Barn Stationary Antenna	RB2	
2022-08-31	2022-08-31T12:00:00Z	230000224258	8210	Recapture	Recapture	
2022-10-04	2022-10-04T12:00:00Z	230000228246	7490	Recapture	Recapture	
2023-07-25	2023-07-25T11:16:23Z	230000228968	6970	Mobile Run	M2	
2023-07-24	2023-07-24T12:36:56Z	230000228295	6100	Mobile Run	M1	

Fast Movements

Show 10 entries

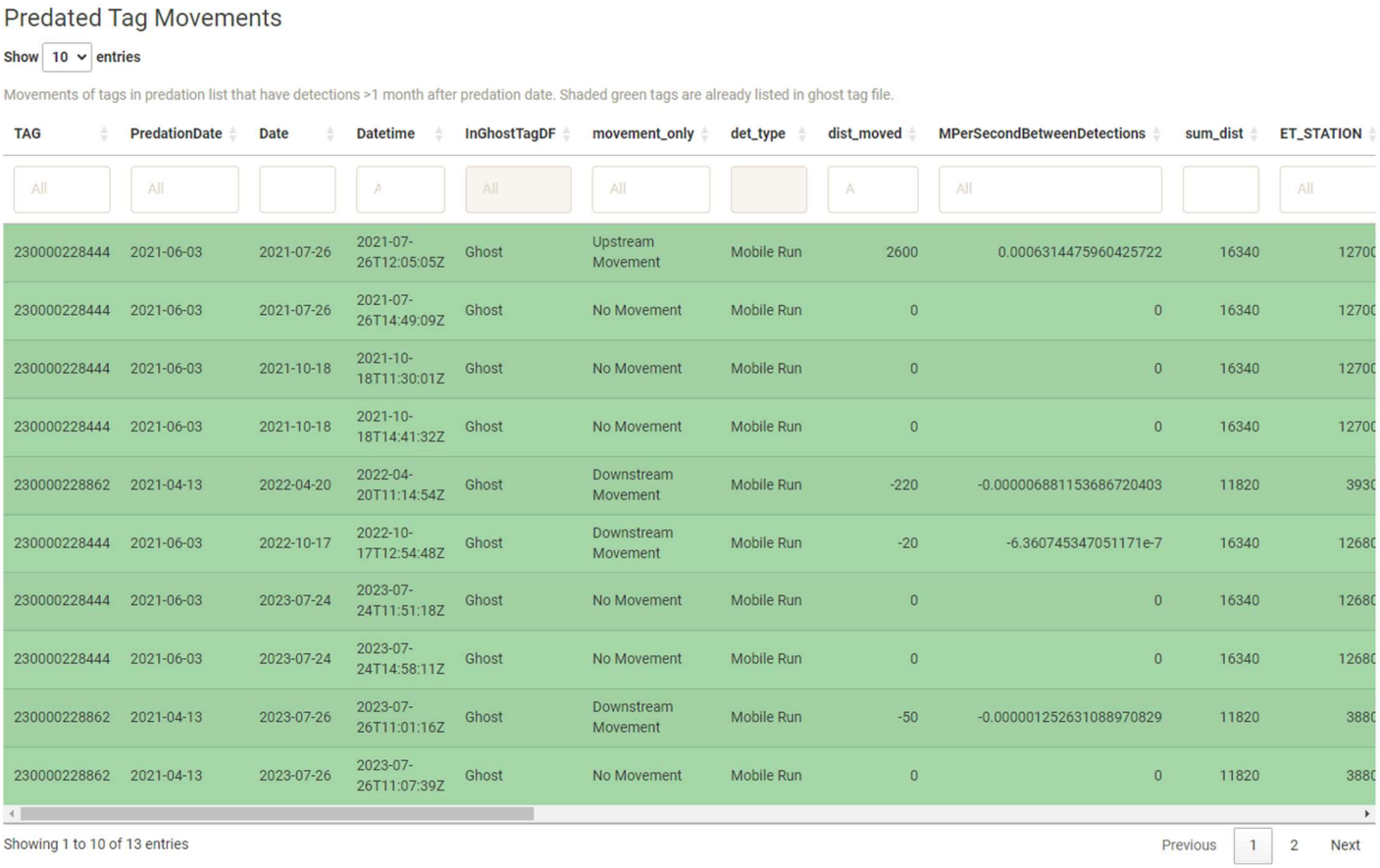
Top 5% of individual movements by fish by speed (meters per second between detections) upstream or

Date	Datetime	TAG	MPerSecondBetweenDetections	det_type
	A	All	All	
2024-10-19	2024-10-19T23:59:44Z	230000143830	350	Confluence Stationary Antenna
2021-07-26	2021-07-26T15:14:03Z	230000228446	-10	Mobile Run
2022-04-20	2022-04-20T09:58:19Z	230000228669	10	Mobile Run
2023-07-24	2023-07-24T15:18:59Z	230000228378	-10	Mobile Run
2024-10-14	2024-10-14T11:25:51Z	230000087756	-10	Mobile Run
2021-07-26	2021-07-26T15:03:27Z	230000228615	-5	Mobile Run
2021-10-18	2021-10-18T11:45:29Z	230000224236	-5	Mobile Run
2023-07-24	2023-07-24T15:18:59Z	230000142998	-5	Mobile Run
2024-10-14	2024-10-14T11:18:30Z	230000228378	-5	Mobile Run
2024-10-27	2024-10-27T21:36:29Z	230000272794	-3.728448275862069	Connectivity Channel Downstream Stationary Antenna

QAQC Predated Tag Movements

Most predated tags are dispelled after about 3 weeks of high activity. This tab shows detections of tags listed as Predated that have also been detected >1 month after their assigned predation date. This is meant to help find if we have erroneously entered any tags in the predation file.

- Green shading here are tags listed as Ghost tags, which happens when a bird eats a fish, poops out the tag somewhere in the river, then we keep detecting it at the same spot or downstream on mobile runs.
- If a tag shows up here not listed as a ghost tag, especially a with an upstream movement or substantial downstream movement not during runoff, then it's a signal to reexamine that tag's history and perhaps remove it from the Predation file
- Recaptures will also show up here



Predated Tag Movements

Show 10 entries

Movements of tags in predation list that have detections >1 month after predation date. Shaded green tags are already listed in ghost tag file.

TAG	PredationDate	Date	Datetime	InGhostTagDF	movement_only	det_type	dist_moved	MPerSecondBetweenDetections	sum_dist	ET_STATION
All	All		A	All	All		A	All		All
230000228444	2021-06-03	2021-07-26	2021-07-26T12:05:05Z	Ghost	Upstream Movement	Mobile Run	2600	0.0006314475960425722	16340	1270
230000228444	2021-06-03	2021-07-26	2021-07-26T14:49:09Z	Ghost	No Movement	Mobile Run	0	0	16340	1270
230000228444	2021-06-03	2021-10-18	2021-10-18T11:30:01Z	Ghost	No Movement	Mobile Run	0	0	16340	1270
230000228444	2021-06-03	2021-10-18	2021-10-18T14:41:32Z	Ghost	No Movement	Mobile Run	0	0	16340	1270
230000228862	2021-04-13	2022-04-20	2022-04-20T11:14:54Z	Ghost	Downstream Movement	Mobile Run	-220	-0.00000688115368672040 3	11820	393
230000228444	2021-06-03	2022-10-17	2022-10-17T12:54:48Z	Ghost	Downstream Movement	Mobile Run	-20	-6.360745347051171e-7	16340	1268
230000228444	2021-06-03	2023-07-24	2023-07-24T11:51:18Z	Ghost	No Movement	Mobile Run	0	0	16340	1268
230000228444	2021-06-03	2023-07-24	2023-07-24T14:58:11Z	Ghost	No Movement	Mobile Run	0	0	16340	1268
230000228862	2021-04-13	2023-07-26	2023-07-26T11:01:16Z	Ghost	Downstream Movement	Mobile Run	-50	-0.00000125263108897 0829	11820	388
230000228862	2021-04-13	2023-07-26	2023-07-26T11:07:39Z	Ghost	No Movement	Mobile Run	0	0	11820	388

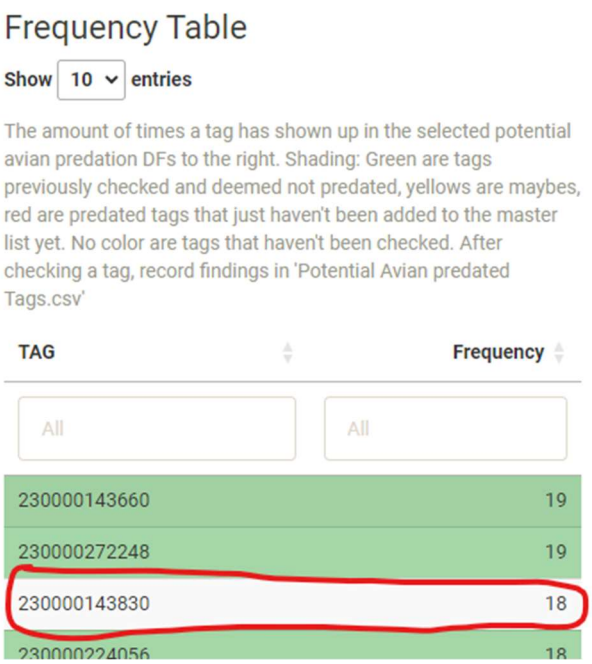
Showing 1 to 10 of 13 entries

Previous 1 2 Next

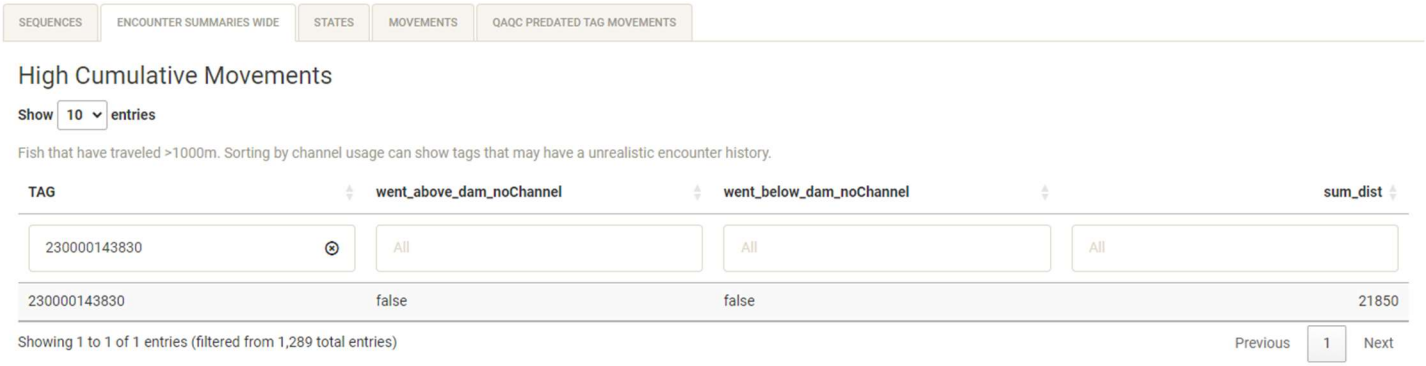
Example

Step 1: Select Tag to examine

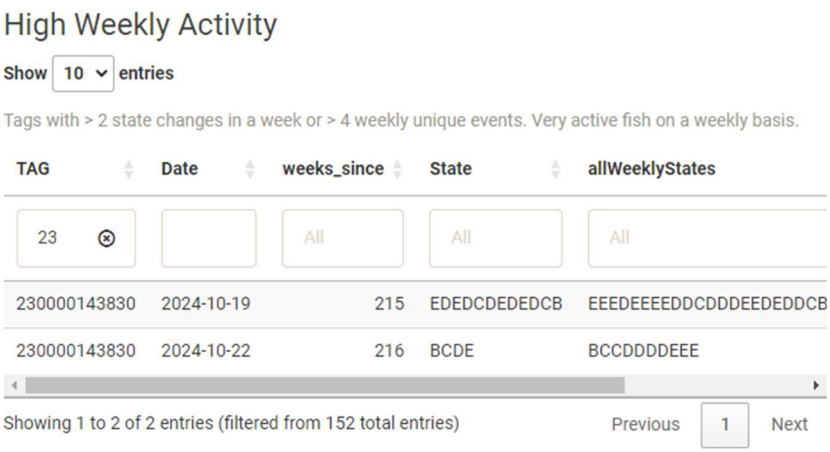
From the frequency table, I notice that tag 230000143830 has not been checked as a predated tag.



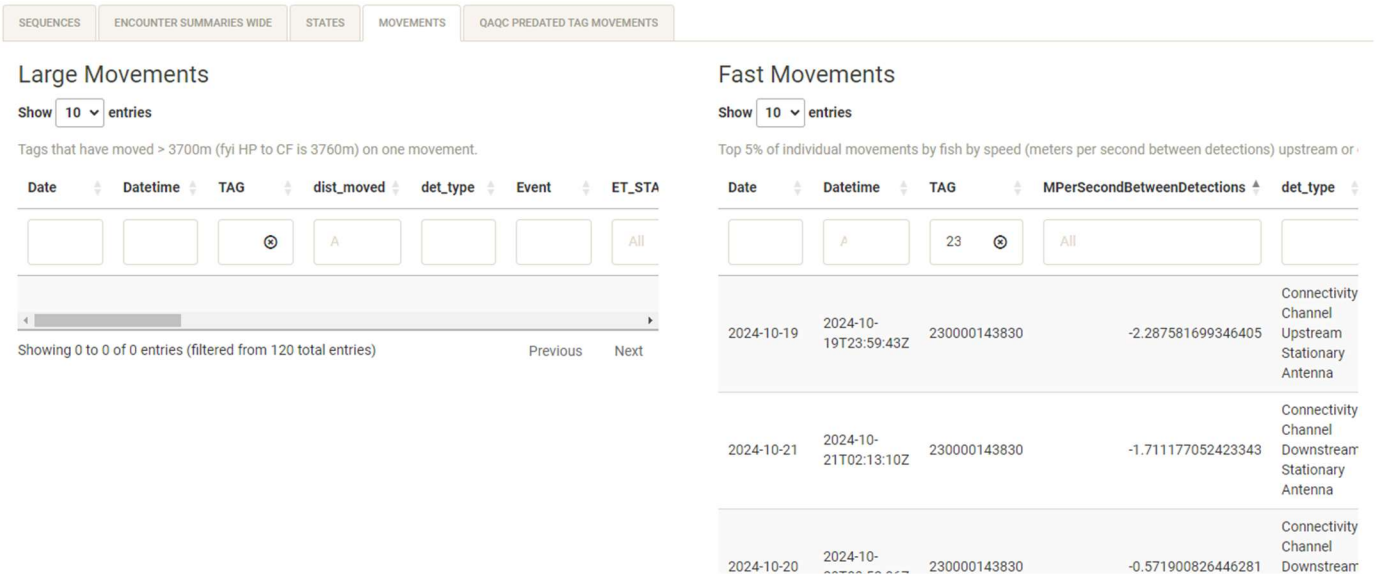
By copying and pasting that tag into the "TAG" search box in each table, I see the tag has moved 21850m in total.



It appears multiple times in the States table for high weekly activity across subsequent weeks.



The tag doesn't appear to have any large movements, but appears in the "Fast Movement" table 15 times, which accounts for most of the tallies in the frequency table.



Step 2: Copy and Paste Tag into "Encounter Histories" -> "All Encounter Histories"

We see immediately that the fish pretty much stayed put in the Fraser River Ranch for a year before a Recapture. Then that fall 2024, it went downstream to CF.

Show 200 entries Search:

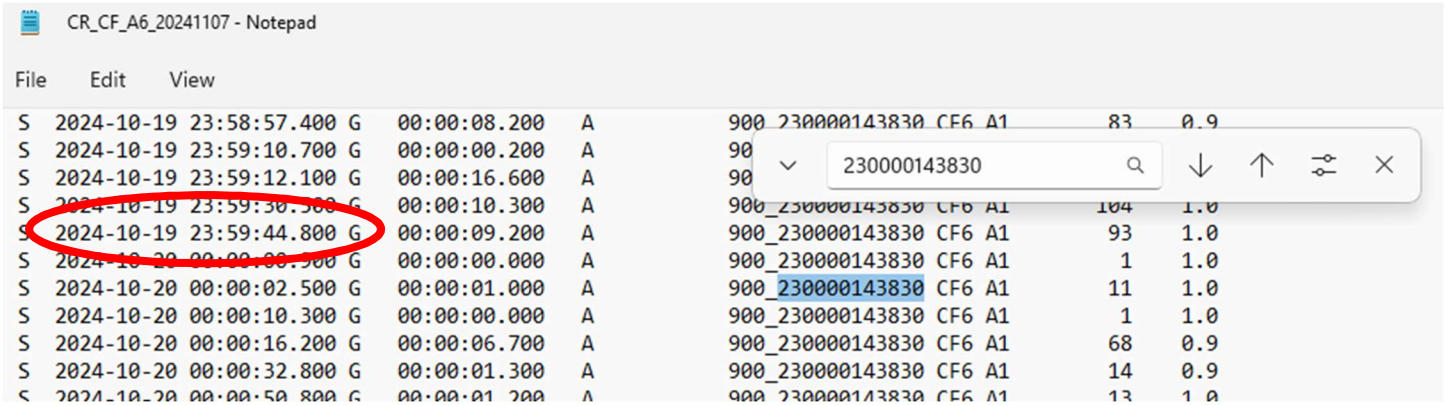
TAG	Date	Time	Datetime	Event	Species	Release_Length	Release_Weight	ReleaseSite	Release_Date	RecaptureSite	R
230000143830	2023-09-05	13:00:00	2023-09-05T13:00:00Z	Release	LOC	321	329	Fraser River Ranch	2023-09-05		
230000143830	2024-09-03	12:30:00	2024-09-03T12:30:00Z	Recapture	LOC	321	329	Fraser River Ranch	2023-09-05	Fraser River Ranch	
230000143830	2024-10-08	21:01:57	2024-10-08T21:01:57Z	CF6	LOC	321	329	Fraser River Ranch	2023-09-05		
230000143830	2024-10-08	21:02:00	2024-10-08T21:02:00Z	CF5	LOC	321	329	Fraser River Ranch	2023-09-05		
230000143830	2024-10-08	21:17:45	2024-10-08T21:17:45Z	CF5	LOC	321	329	Fraser River Ranch	2023-09-05		
230000143830	2024-10-08	21:18:05	2024-10-	CF5	LOC	321	329	Fraser River	2023-09-05		

The detections get confusing starting on the 19th of October. First there is this detection between CF6 and CU1 (missing CF5 and CU2 in between) with only 1second in between.

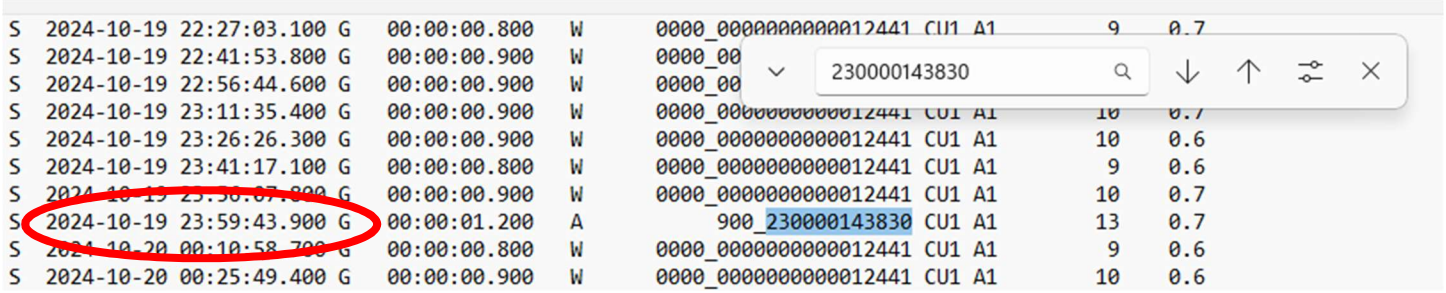
230000143830	2024-10-19	23:59:10	2024-10-19T23:59:10Z	CF6	LOC	321
230000143830	2024-10-19	23:59:12	2024-10-19T23:59:12Z	CF6	LOC	321
230000143830	2024-10-19	23:59:30	2024-10-19T23:59:30Z	CF6	LOC	321
230000143830	2024-10-19	23:59:43	2024-10-19T23:59:43Z	CU1	LOC	321
230000143830	2024-10-19	23:59:44	2024-10-19T23:59:44Z	CF6	LOC	321
230000143830	2024-10-20	00:00:00	2024-10-20T00:00:00Z	CF6	LOC	321
230000143830	2024-10-20	00:00:02	2024-10-20T00:00:02Z	CF6	LOC	321

This is actually so bizarre even for Predated tags that I went and checked the raw files for CU1 and CF6 from that time period just to make sure something didn't get messed up.

Here is a screenshot from "U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\RFID\Detections\CF\CR_CF_A6_20241107.TXT"

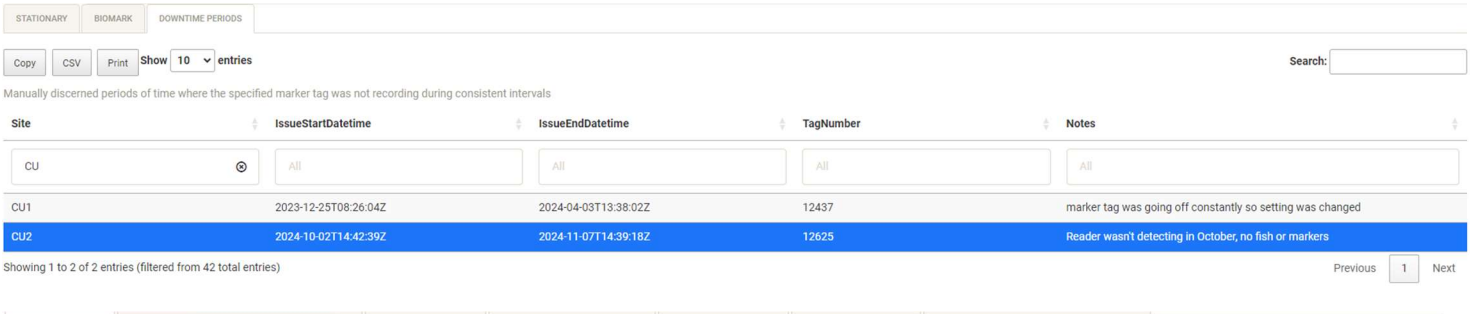


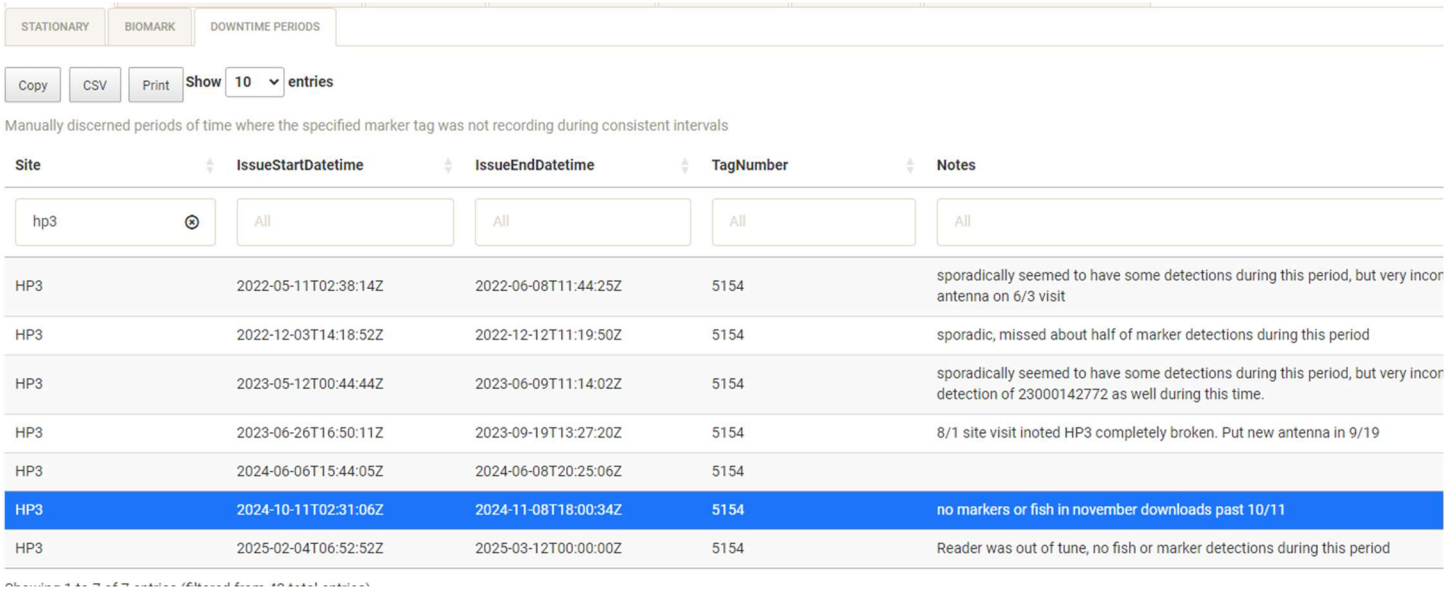
And here is "U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\RFID\Detections\CU\CRCC_CU_A11_20241107.TXT"



There was definitely a detection from that tag on both antennas within 1 second of each other, which makes no sense, but at least the data in the app is accurate from the Raw files. Moving on….

Under the "QAQC" -> "Marker Tags" tab, you can see if any antennas were off/not working to explain any absences. Turns out that both HP3 and CU2 were down during October 2024 while these fish detections were occuring, so it makes sense these antennas wouldn't show up in the tag's history during this time. See the "Marker Tags" section on how to find downtime periods on the antennas.





Sometimes the tag executed the expected sequences, like here, hitting each working antenna on the way from Hitching Post to Confluence over about 21 hours.

Other times, the tag bounced between antennas, missing some in between but almost always hitting the anticipated sites.

230000143830	2024-10-20	03:20:13	2024-10-20T03:20:13Z	HP4	LOC
230000143830	2024-10-20	18:26:12	2024-10-20T18:26:12Z	HP4	LOC
230000143830	2024-10-20	21:11:38	2024-10-20T21:11:38Z	CS1	LOC
230000143830	2024-10-20	21:11:46	2024-10-20T21:11:46Z	CS1	LOC
230000143830	2024-10-20	21:12:15	2024-10-20T21:12:15Z	CS2	LOC
230000143830	2024-10-20	21:12:23	2024-10-20T21:12:23Z	CS2	LOC
230000143830	2024-10-20	23:13:47	2024-10-20T23:13:47Z	CU1	LOC
230000143830	2024-10-21	00:37:16	2024-10-21T00:37:16Z	CF5	LOC
230000143830	2024-10-21	00:38:11	2024-10-21T00:38:11Z	CF6	LOC
230000143830	2024-10-21	00:38:17	2024-10-21T00:38:17Z	CF6	LOC

In the below screenshot, you see a sequence where the fish moved upstream from CD1 to CF6. On Upstream journeys especially, you would expect both antennas at a site to get hit so it's a red flag that the fish missed CD2, CU1, and CF5 on the first journey upstream, which took the tag 18 minutes. Then it's right back downstream to Red Barn over the course of 6 hours.

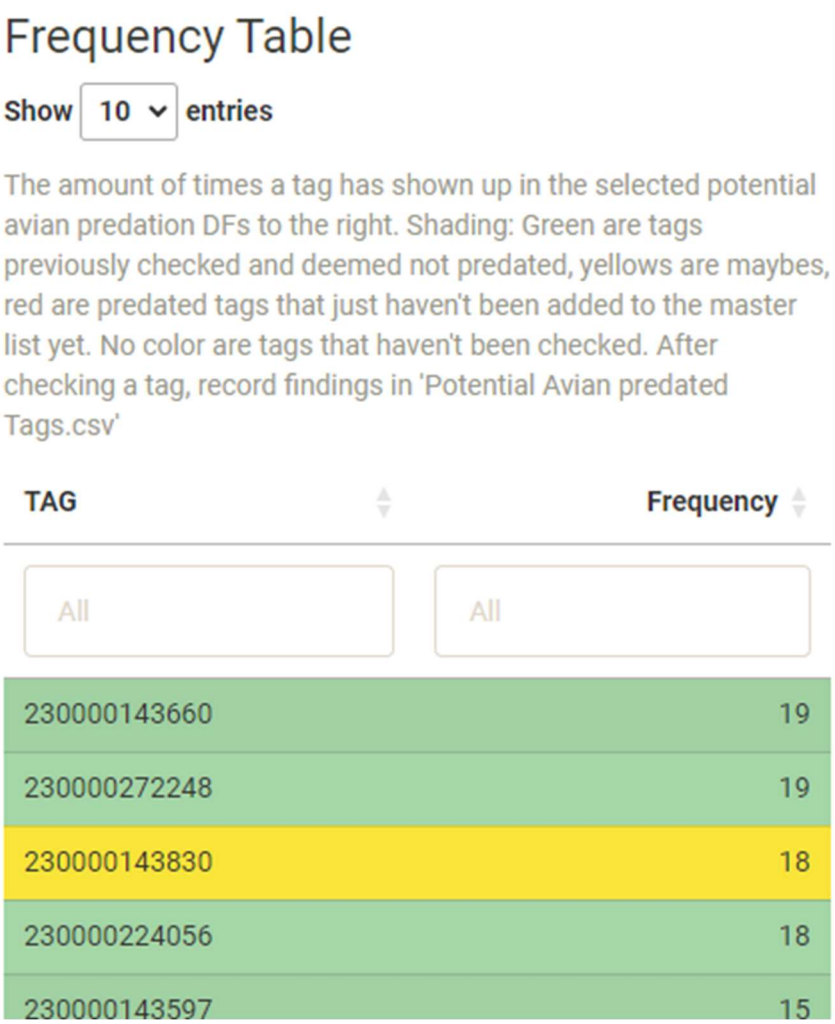
230000143830	2024-10-21	18:18:36	2024-10-21T18:18:36Z	CD1
230000143830	2024-10-21	18:37:29	2024-10-21T18:37:29Z	CF6
230000143830	2024-10-21	18:37:48	2024-10-21T18:37:48Z	CF5
230000143830	2024-10-21	19:18:35	2024-10-21T19:18:35Z	CD2
230000143830	2024-10-21	23:17:52	2024-10-21T23:17:52Z	HP4
230000143830	2024-10-22	00:38:08	2024-10-22T00:38:08Z	RB2
230000143830	2024-10-22	00:38:18	2024-10-22T00:38:18Z	RB1
230000143830	2024-10-22	16:19:18	2024-10-22T16:19:18Z	RB1
230000143830	2024-10-22	16:20:21	2024-10-22T16:20:21Z	RB2
230000143830	2024-10-22	21:03:33	2024-10-22T21:03:33Z	HP4
230000143830	2024-10-22	21:03:39	2024-10-22T21:03:39Z	HP4
230000143830	2024-10-23	12:29:44	2024-10-23T12:29:44Z	CD1
230000143830	2024-10-23	12:29:52	2024-10-23T12:29:52Z	CD1
230000143830	2024-10-23	12:30:20	2024-10-23T12:30:20Z	CD1

Overall, the tag exhibits some strange patterns but often hits antennas in sequence. For now we will mark this tag as a "Maybe" for avian predation and ask Eric Richer/Fetherman.

Step 3: List Tag and Re-run Data

- List tag in "Potential Avian Predated tags", located in "U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\RFID\Detections\AvianPredation\Potential Avian Predated tags.xlsx".

- If tag/s are deemed predated, list them in "U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\RFID\Detections\AvianPredation\WGFP_AvianPredation.xlsx".
- Once you have listed all desired tags, save "Potential Avian Predated Tags.xslx" and "WGFP_AvianPredation.xlsx" files as .csvs and copy them over to the "data" directory within the WGFP app directory. Re-run "CreateFlatFiles_Runscript.rmd".
- Clear your global environment and re-run app.
- Tag 230000143830 is now highlighted yellow, indicating it has been checked and there is uncertainty on its status.

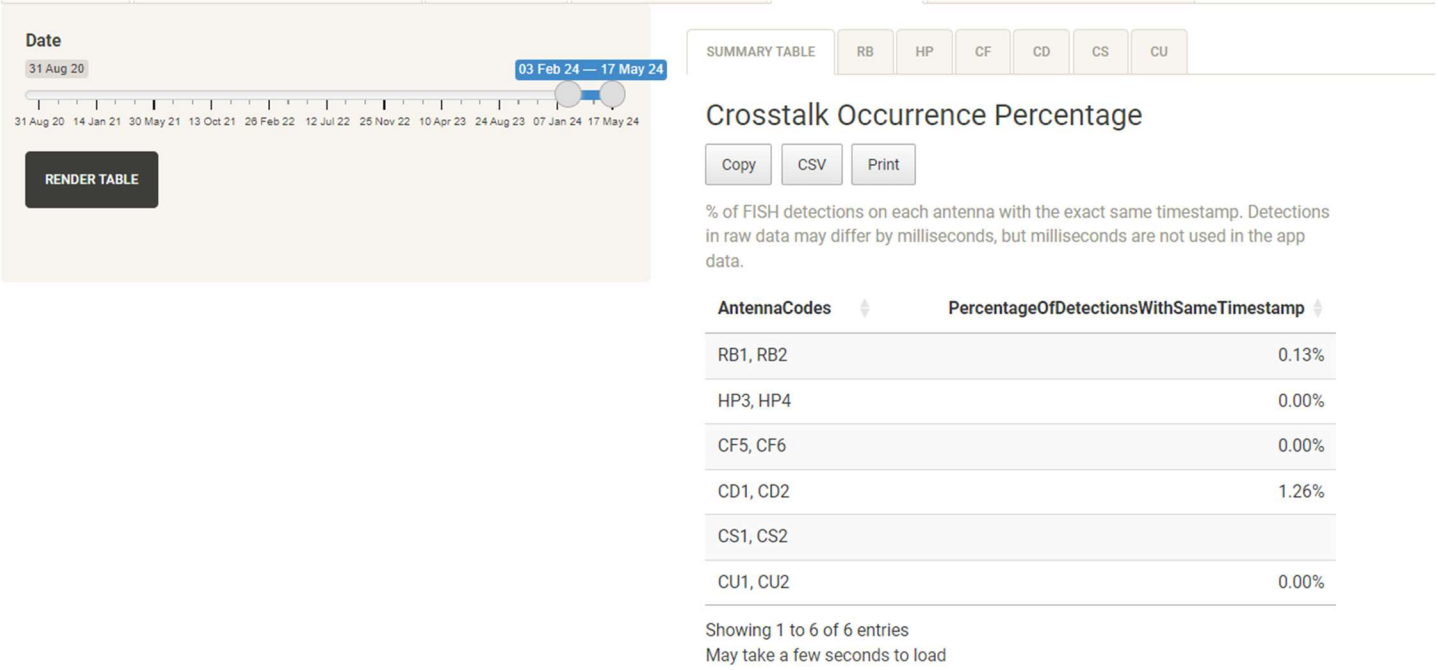


Crosstalk QAQC

Marker Tag crosstalk, where the tag goes off on one antenna but also registers on the other antenna as well, has been occurring frequently on CD since its installation. This tab is meant to monitor crosstalk instances of *Fish*, where a non-marker tag is registered at both antennas of a site at the exact same time.

Is created from combinedData_df_list$All_Events, originally stemming from All_combined_events_function().

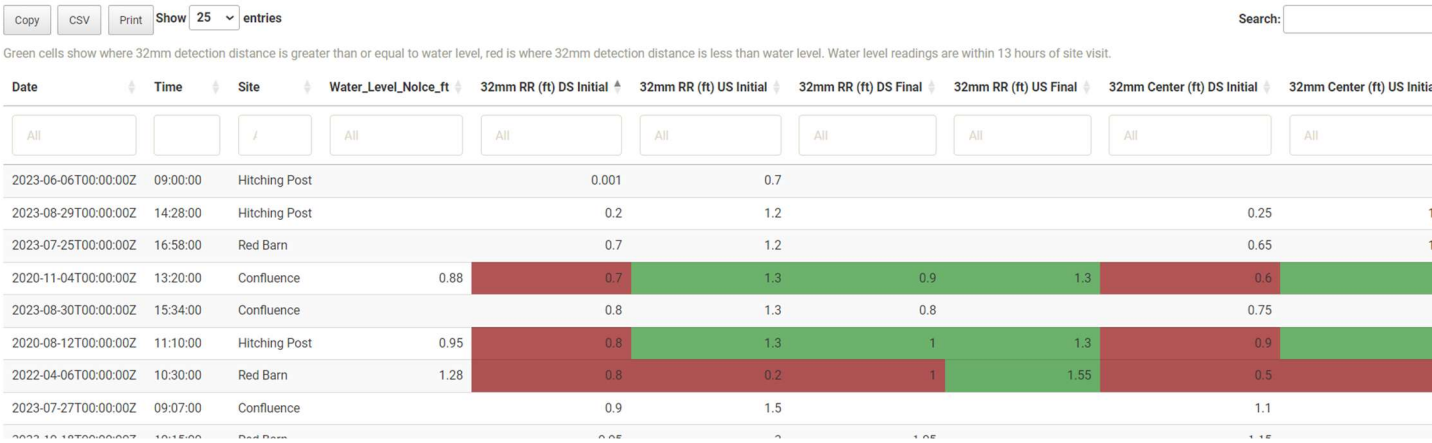
- Summary Table shows the percentage of fish detections across the selected timeframe have been recorded at the exact same time
- Individual site tabs show when those simultaneous detections occurred
- There are instances where the simultaneous detections occurs naturally. In the app, milliseconds are not used, but the data coming off the readers do use milliseconds. Sometimes there are instances where a fish is recorded milliseconds apart on different antennas, but these will show up as the exact same timestamp in the app. However, when CD is doing it far more frequently than the other sites, that probably means something more is going on.



Detection Distance/Water Level

Shows a color coded table of SiteVisitAndPTData, a joined df of PTdata and Site Visit Data by time to the nearest 13 hours.

- Using the variable "Water_Level_NoIce_ft", this table colors red the 32mm readings that are *below* the observed water level and green *above or at* the observed water level
- Helpful for identifying periods where detections may have been missed because of high flows.
- Readings that were entered as "TOUCHING" are changed here to .001



- Notes: water level data is calibrated to staff gage at antenna site, but actual depths across antennas will vary

Adding/Removing Dummy rows in the data

Dummy rows were added to the stationary, biomark, and release data using the dummy_rows.R script to ensure the framework for new antennas was in place. This code is retained for when/if new antennas are to be added. See "Updating the app: -> "Modifying, Updating, Adding New Antenna/Stations" for more info.

```
55  #### This part was for checking if new antennas to be put in will work
56  source("functions/dummy_rows.R")
57  dummy_rows_list <- add_dummy_rows(stationary = Stationary, biomark = Bic
58  Stationary <- dummy_rows_list$Stationary
59  Biomark <- dummy_rows_list$Biomark
60  Release <- dummy_rows_list$Release
61  #ghost_tag_df <- dummy_rows_list$Ghost_tags #date column is named "Ghost
62
```

The rows remain while the function is ran to set the framework/structure of the data. Then the rows are removed in the data so this dummy data isn't used.

In the code, the rows are removed….

- In the get_movements_function

```
 8    movement_table_notrans <- combined_events_stations %>%
 9      #### removing dummy tag
10      filter(!TAG %in% c("230000999999")) %>%
11      select(Date, Datetime, TAG, det type, Event, FT STATION, SI
```

- In All_Combined_Events_function:

```
condensedAllEventsWithReleaseandEnvironmentalInfo <- condensedAllEventsWithReleaseandEnvironmentalInfo %>%
  replace_na(list(Species = "No Info", ReleaseSite = "No Info",
                  #changing release wight and length to 0 keeps the fish in the df when weight/lentgh filters are active
                  Release_Weight = 0, Release_Length = 0,
                  Site = "No Site Associated")) %>%
  dplyr::filter(!TAG %in% c("230000999999"))
```

- After the functions are ran in the following individual datasets in CreatFlatFiles_Runscript.rmd:

```
### taking dummy tag out

Biomark <- Biomark %>%
  filter(!`DEC Tag ID` %in% c("900.230000999999"))
cleanedRelease <- cleanedRelease %>%
  filter(!TagID %in% c("230000999999"))
Stationary <- Cleaned_Stationary_Fishdetectionsonly %>%
  filter(!TAG %in% c("900230000999999"))
```

- In Ind_tag_enc_hist_wide_summary function:

```
170
171    #### dummy rows removal: 1/14/23
172    ENC_Release6 <- ENC_Release6 %>%
173      filter(!TAG %in% c("230000999999"))
174
```

- In the reactive and frequencies table in allEncountersMod.r

```
#### filter dummy row
all_events_filtered <- all_events_filtered %>%
  filter(!TAG %in% c("230000999999"))

output$alleventsfrequencies1 <- renderDataTable({
  frequenciesSummarized <- all_events_data() %>%
    filter(!TAG %in% c(230000999999)) %>%
    count(Event, name = "Raw Detections")
```

To put the dummy rows back in, comment out the code that filters out the dummy rows above. This is useful for when a new antenna is to be added.

Data Used in the App

The following files are used in the app and are manually derived.

Tabular data: WGFP_dataclean_vis2.0\data

Stationary detections: WGFP_Raw_yyyymmdd.rds

- Combined and cleaned file of all stationary detections. Obtained from combining data using the "Combine data RShiny app" found at U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\RFID\Detections\CodingDetections\WGFP_CombiningData_RShinyApp.
- Most recent file is found at U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\RFID\Detections\All_Stationary

Mobile: WGFP_Mobile_Detect_AllData.csv

- A combined file of all mobile detections found at U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\RFID\Detections\MobileRaftAntenna\Mobile_Detections
- Usually Eric R prepares this
- Column names and order: "Num", "River" , "MobileSite", "Date", "Time" , "T109_C", "UTM_X", "UTM_Y" ,"TagType" , "TagID", "Event", "Ant", "Pass", "Species" , "Length" , "Weight" , "TagSize" , "RS_Num", "ReleaseSite", "Survey", "Notes"
- Of these, TagID, Date, Time, UTM_X, UTM_Y, and Ant are the most important columns

Biomark: Biomark_Raw_yyyymmdd.rds

- Combined file of all Biomark detections, found at U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\RFID\Detections\Biomark
- Made from the "Combine New Data RShiny App"
- Column names and order: Scan Date, Scan Time, Download Date, Download Time, Reader ID, Antenna ID, HEX Tag ID, DEC Tag ID, Temperature,C, Signal,mV, Is Duplicate, Latitude, Longitude, File Name
- Most important columns are Reader ID, Scan Date, Scan Time, DEC Tag ID

Release: WGFP_ReleaseData_Master_ yyyymmdd.csv

- Master Release file of all tagged fish. Found at U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\RFID\Tagging
- Column names and order: "RS_Num", "River", "ReleaseSite", "Date" , "Time" , "Year", "UTM_X", "UTM_Y", "Species", "Length" , "Weight", "TagType", "TagID", "QAQC", "TagSize", "Ant" , "Event", "FinClip", "Mortality", "Comments"

Recaptures: WGFP_RecaptureData_Master_ yyyymmdd.csv

- File of all fish that were recaptured found at U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\RFID\Recaptures
- Column names and order: "RS_Num", "River" , "RecaptureSite" ,"Date", "Time", "UTM_X" , "UTM_Y", "Species" , "Length", "Weight", "TagType", "TagID", "QAQC" , "TagSize" , "Ant", "Event", "FinClip", "Mortality" , "Comments"

Avian Predation:

- WGFP_AvianPredation.csv
    - csv of tags succumbed to avian predation with a date of predation found at "U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\RFID\Detections\AvianPredation\WGFP_AvianPredation.csv"
    - used in the "States" function to assign a predated state
    - Avian Predation QAQC tab is useful in finding new tags
- Potential Avian Predated tags.csv
    - Made from "U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\RFID\Detections\AvianPredation\Potential Avian Predated tags.xlsx"
    - List used to keep track of tags already checked to see if they are predated or not
    - List tags in .xlsx file, then save as .csv, then transfer to app
    - Used to color cells in the Avian Predation QAQC tab

Ghost Tags: WGFP_GhostTags.csv

- "U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\RFID\Detections\GhostTags\WGFP_GhostTags.xlsx" – Need to export/save the "Ghost tags" tab to csv
- csv of ghost tags with date of ghost tag
- used in the "States" function to assign a ghost state
- Column names: RS_Num, River, ReleaseSite, ReleaseDate, Species, Length, Weight, TagID, TagSize, Event, GhostDate, UTM_X, UTM_Y, Comments

Metadata: WGFP Metadata.xlsx

- Found in U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\RFID\Detections
- Used to keep track of antenna names, UTMs and what they're called in the data, as well as test tags, marker tag/antenna downtimes, and important stationing variables.
- Includes tabs:
    - Antenna Metadata: Used to help assign names and UTMs to the antennas
        - AntennaSite: Name of Antenna (ie Red Barn (Upstream))
        - SiteName: Name of Site (ie Red Barn Stationary Antenna). **DO NOT CHANGE THIS FIELD** for an antenna
        - FrontendSiteCode: How we want the shorthand code to display in the app. Needs to be just one entry but can be changed as you wish.
        - BackendSiteCode: code(s) that have been used in the data to denote this antenna. Make sure this code is the same ones that come off the readers: SCD field for Stationary, Reader ID for Biomark. It's ok to have multiple names for the Biomark Antennas separated by comma and a space, but not for Stationary ones
        - AntennaSiteShortHandShorthand: Short abbreviation for site
        - PressureTransducerSiteName: Name of the Site as it appears in the Pressure Transducer Data. Must be updated if new sites are added
        - DetectionDistanceSiteName: Name of the site as it appears in Site Visit/Detection Distance Data
        - UTM_X and UTM_Y: UTMs of the site
        - River: River it's deployed on. **DO NOT CHANGE THIS FIELD** for an antenna
        - Deployment Duration: Dates and times where they have been deployed
        - Notes: any notes on the antenna
    - MarkerTagIssues: used to keep track of downtime periods on the antennas. Displayed in the QAQC tab of Marker Tags under "Downtimes"

- ImportantStationingVariables: Used to keep track of important sites, used for movement calculations within the app
- TestTags: Keeps track of all test tags and removes these tag detections from the data
- Notes: Keeps track of general metadata in the worksheet. Current notes: (5/24/2024)
    - Assumes that for AntennaMetadata, SiteName and River will not be changing. If they do change, you'll have to go into the runscript and a couple functions including "PrepareforMovementsStatesand Summaries" to change how those variables are located
    - For the biomark antennas, I don't think you can actually code the readers to detect as the frontend codes; only codes like A4,B2 etc…hence the need for frontend/backend codes
    - For antenna metadatea codes, the frontend code should only have 1 entry. It's ok for the biomark backend codes to have multiple entries, but not the stationary ones
    - For marker color to be assigned correctly, need to have "Biomark Antenna", "Stationary Antenna", and "Mobile Run" in SiteName
    - First data date: 2020-08-06. detections before this date are removed
    - PressureTransducerSiteName needs to line up with the sites that are in the Pressuretransducer data

Pressure Transducer (in PressureTransducerFolder): WGFP_PressureTransducer_sitename.csv

- U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\Pressure_Transducer
- Cleaned Pressure Transducer files from each site in csv form are all put into \WGFP_dataclean_vis2.0\data\PressureTransducer
- **ALL FILES MUST HAVE SAME COLUMN NAMES/ORDER**

Site Visits: WGFP_SiteVisits_FieldData.xlsx

- U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\RFID\SiteVisits
- 2 relevant sheets, Stationary and Biomark used for detection distances.

Spatial Data: WGFP_dataclean_vis2.0\gis

These data are obtained by exporting from .dbfs in ArcMap or ArcCatalog.

Read into the app with miscR/map_polygon_readins.R

newAntennaSites.shp: point file of antenna sites locations. Up to date as of May 9, 2025

Stream_Centerline_Post.shp: centerline of the Fraser, Upper Colorado, and Colorado River below windy gap

Release_Sites2021.shp: point file of release site locations

mobile_reaches.shp: mobile reaches extents

WGFP_States_2024.shp: States defined for assignment to detections in Spatial_join_function.R

simpleStations.rds: stream centerlines broken into 10 meter sections. Originally from Stations_10m_Post.shp, but simplified to 10% of original resolution to improve map loading efficiency (see Adding or updating layers on the map)

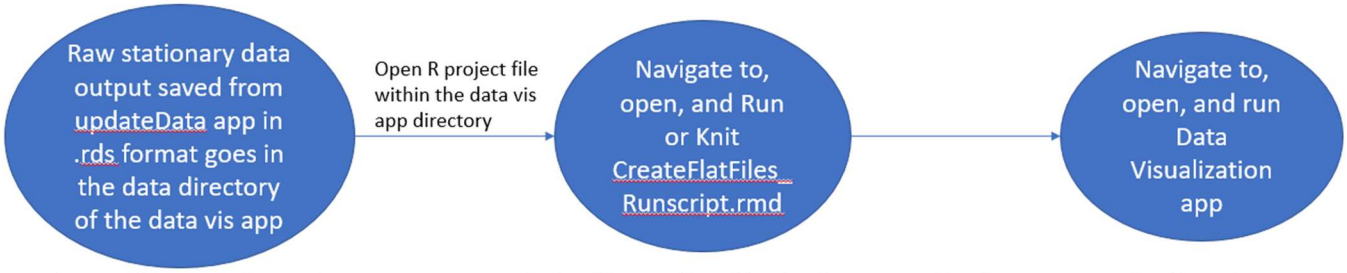
Static Files for app: WGFP_dataclean_vis2.0\data/FlatFilesForApp

These files are created in CreateFlatFiles_Runscript.rmd (see Updating the app) and saved to this directory to reduce calculations and processing within the app as much as possible.

Updating the App

The app data is updated by code ran within CreateFlatFiles_Runscript, an RMarkdown (.rmd) file that takes files (see Data Used in the App), wrangles the data with a variety of functions, and saves them as .rds files in \WGFP_dataclean_vis2.0\data\flatFilesforApp. This wrangling is done outside the app to reduce load time and computations within the actual app as much as possible. Anytime there is new data for any of these files, the runscript must be ran in order to update the data in the app.

The typical workflow for updating the app for new data is below, following the use of UpdateDataApp to combine new detections with the previous detection files.



- The resulting dataframe is a cleaned file of stationary data
- Biomark process Also uses .rds
- 2023 example Stationary file size of .rds file vs .csv: ~13 mb vs ~157 mb

- Uses tabular files outlined in the "Data used in the app" section (File Readins)
- Creates static files used in the data visualization app by cleaning and combining with other files in "data" directory (Creating New Flat Files/Running the Script)
- Outputs .rds files into the flatFilesforApp folder
- Needs to be run anytime there is a change to a data file to get the most up to date data in the app; this includes files in the "data" directory and updated spatial .shp files in the "gis" directory

- App loading time cut down by at least 3 minutes
- Output should be consistent without the need to clean the files within the app
- Check to make sure the expected up to date data is included

File Readins

Anytime there is new data of any of the files ending in yyyymmdd listed above, the name of the file must be updated in the Runscript. Make sure the file is saved in the correct directory, update the file name in the runscript, and knit the runscript.

Column names and order matter. When updating the csv or excel files, make sure the new file has the same column name/order as the old one.*

Column names outlined below are how they appear when brought into R at the beginning of the app. In Excel, names will appear slightly differently.

**If you do need to change the column names or order, you'll have to go into the app and follow where that column is used in the app and change the name*

Make sure tag is read in correctly. The way to do this is in excel. Save the Tag column in csvs as numeric type with 0 decimal places, instead of general (default). If not, the runscript won't run all the way through (see Common Errors)

Updating file names

To update the file name, change these dates/names for the appropriate file.

```
Stationary <- readRDS("./data/WGFP_Raw_20240514.rds")

Mobile <- read.csv("./data/WGFP_Mobile_Detect_AllData.csv" , colClasses= c(rep
rep("character", 3)))

Biomark <- read.csv("./data/Biomark_Raw_20221102.csv", dec = ",")

# need to have tagID as a numeric field in the .csv file in order to be read i

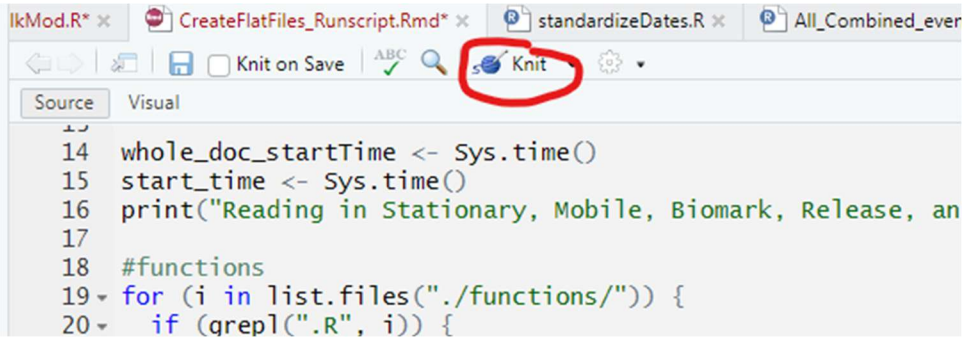
Release <- read.csv("./data/WGFP_ReleaseData_Master_20240520.csv", na.strings
colClasses=c(rep("character",8), "numeric", "numeric",rep("character",8)))
```

Other Data Updates

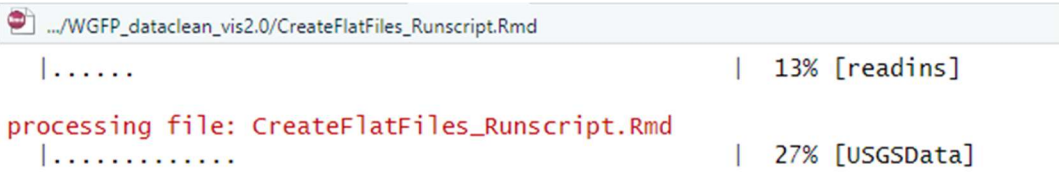
- Pressure Transducer: overwrite the current file with the new one. If a new site with new pressure transducer data is added, add that file to the "data/PressureTransducer" directory. Open "WGPF Metdata.xlsx" (located in the data folder of the app, and U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\RFID\Detections) and enter the correct "PressureTransducerSiteName" as it corresponds to the name used in the PT file. Names in the PT file need to match those in the metadata file. No need to change anything in the app, it will automatically get read in and combined with the other site data **as long as the new site and data has the same columns/order as the other data**
- USGS data: scraped from the USGS server and is updated when the runscript runs.
- All other files: overwrite the previous file, making sure column names/order were the same as the previous one

Creating New Flat Files/Running the Script

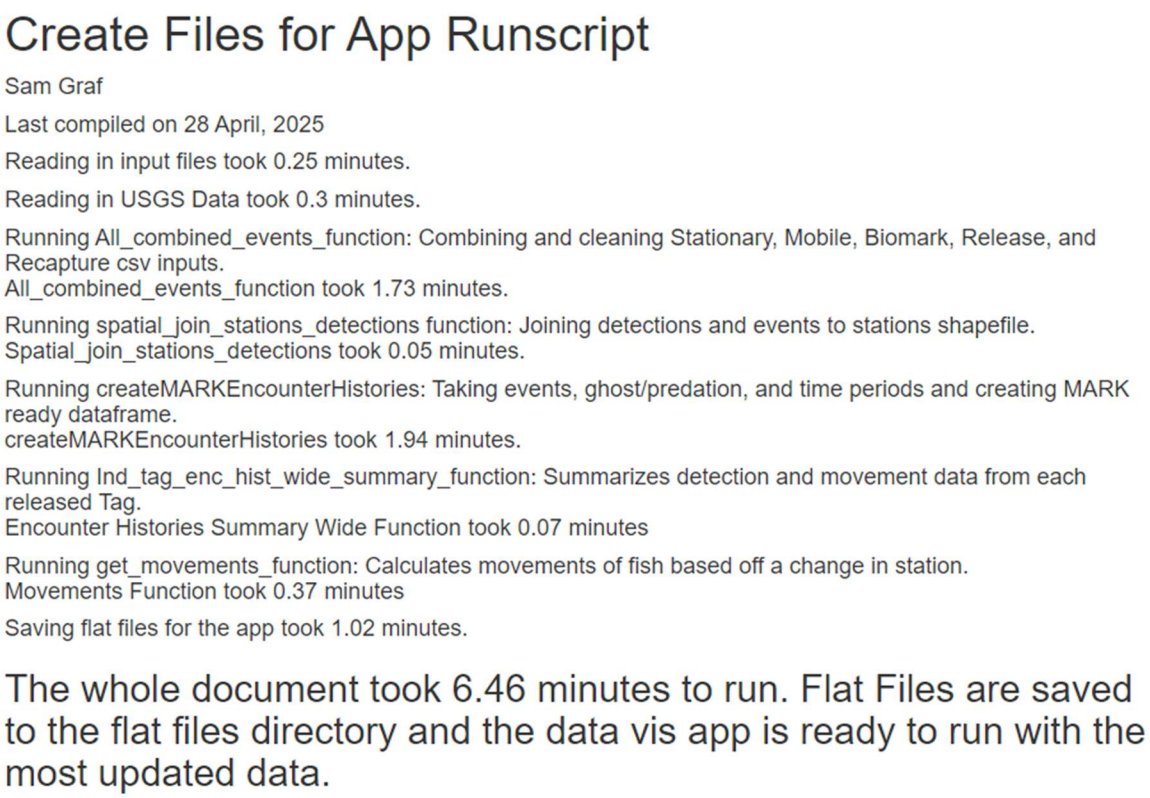
To run the script and update the data in the app, click "Knit"



Progress will show in the "Render" window based off chunks defined in the runscript



At the end, the document will output a HTML document just to confirm that the data was saved. On a local machine this usually takes 5-6 minutes, sometimes up to 10 or more when using a vpn or directly on the U drive. Saving the files takes particularly long on the U drive.



There is some QAQC that happens in this runscript as well, including checking if Tags have multiple entries in the Avian Predation, ghost tag, and release files. If they do, it will show up in the final HTML doc. This also catches if tags were saved incorrectly (see Common Errors).

The resulting flat/static files are saved to data/flatFilesforApp and are automatically used in the app, it's ready to use (see How to Open)!

Common errors:

Warning: cannot open file './data/Biomar_Raw_20221102.csv': No such file or directoryError in file(file, "rt") : cannot open the connection.

- This usually means the filename is spelled wrong or is in the wrong directory

"Error in scan(file = file, what = what, sep = sep, quote = quote, dec = dec, : scan() expected 'a real', got a ___"

- This typically means that there is an incorrect number of columns in the data compared to what the code was expecting. Check the code where the problem file is being read in to see which column types it expects to find in each place, and/or modify the excel file.

Runscript is not able to run all the way through and the console spits out that there are multiple tags of the same name (2.3E11, 2.26E11)

- Annoyingly, the tags in one of the csvs didn't save correctly after being opened and closed in excel. In order to get them to save correctly, the best way is to open the problem file in excel, select the TAG column, change the type from General to Numeric, move the decimal place over 2 places so there isn't a decimal at all, and resave it. You may have to go grab the file from the original location on the U drive again is the tag numbers have already been saved as 230000000000 instead of 230000678123
    - Note: this error should appear less the more that we use .rds files

USGS not being read in

- I can't remember exactly what error this throws but it's something related to server connection. This happens sometimes when the USGS server is not working well and so we're unable to scrape the most recent metrics from their site. Sometimes it helps to not use a VPN, but mainly it's not something we can fix on our end and typically this resolves within a few hours. Otherwise, try to run the script tomorrow.

This section of the app adds dummy rows with fake data from CD, CU, B5, and B6. It was added to make sure the functionality for these new antennas was working, and is not currently in use.

```
#### This part was for checking if new antennas to be put in
source("functions/dummy_rows.R")
dummy_rows_list <- add_dummy_rows(stationary = Stationary, bi
Stationary <- dummy_rows_list$Stationary
Biomark <- dummy_rows_list$Biomark
Release <- dummy_rows_list$Release
ghost_tag_df <- dummy_rows_list$Ghost_tags #date column is na
```

Modifying, Updating, Adding New Antenna/Stations

There are 2 parts to adding a new antenna to the app: updating the spatial files and updating the code that wrangles the tabular data. You can just update the tabular data code without updating the spatial file if you want.

Updating the code and data that wrangles the tabular data

Metadata

- Add new antenna and its data in with WGFP Antenna Metadata (See Data in app)

AntennaSite	SiteName	FrontendSiteCode	BackendSiteCode	PressureTransducerSiteName	DetectionDistance	UTM_X	UTM_Y	River	DeploymentDur
Biomark test Antenna in random spot	Biomark test Antenna	T1	A5			420727	4437229	Fraser River	

CreateFlatFilesRunscript

Metadata Variable names

- In CreateFlatFilesRunscript, add the variable names to the list in the "metadatavariables" code chunk
    - Do this for a backend variable name and a frontend one. "AntennaSite" column in the metadata is used to identify these codes, so change the code to match what is in AntennaSite

```
```{r metadatavariables, include=FALSE, echo = FALSE}
###Variables that are used throughout the funcitons
###Frontend (display) codes
TestBiomarkAntennaFrontendSiteCode <-
as.character(wgfpMetadata$AntennaMetadata[!is.na(wgfpMetadata$AntennaMetadata$AntennaSite) &
wgfpMetadata$AntennaMetadata$AntennaSite == "Biomark test Antenna in random spot", "FrontendSiteCode"])

WindyGapBypassAntennaFrontendSiteCode <-
as.character(wgfpMetadata$AntennaMetadata[!is.na(wgfpMetadata$AntennaMetadata$AntennaSite) &
wgfpMetadata$AntennaMetadata$AntennaSite == "Windy Gap Bypass Channel", "FrontendSiteCode"])
```

- Add these variables to metaDataVariableNames

```
metaDataVariableNames <- list(
  "TestBiomarkAntennaBackendSiteCode" = TestBiomarkAntennaBackendSiteCode,
  "TestBiomarkAntennaFrontendSiteCode" = TestBiomarkAntennaFrontendSiteCode,
  "WindyGapBypassAntennaFrontendSiteCode" = WindyGapBypassAntennaFrontendSiteCode,
  "WindyGapAuxiliaryAntennaFrontendSiteCode" = WindyGapAuxiliaryAntennaFrontendSiteCod
  "GranbyDiversionAntennaFrontendSiteCode" = GranbyDiversionAntennaFrontendSiteCode,
  "RiverRunAntennaFrontendSiteCode" = RiverRunAntennaFrontendSiteCode,
  "FraserRiverCanyonAntennaFrontendSiteCode" = FraserRiverCanyonAntennaFrontendSiteCod
```

Dummy Rows

If the antenna doesn't have any new data yet, you will need to set up the functions to account for that antenna. This is done by using the function dummy_rows.r. This is also described in "Adding/removing dummy rows"
```

- Navigate to dummy_rows.r (in the functions directory) and add a row to the biomark or stationary data depending on which antenna type we want to add. The only columns that are important are adding the dumy tag number (230000999999) and making sure we've got the right reader ID that lines up with what we put in the metadata

```
Biomark <- biomark1 %>%
  add_row(Scan.Date = "2022-12-07",Scan.Time = "12:52:10.810", Download.Date = "9/16/2022",Download.Time = "11:30:41",
          Reader.ID = "A3",Antenna.ID = 1,HEX.Tag.ID = "384.358D14F739",
          DEC.Tag.ID = "900.230000999999",Temperature.C = NA,Signal.mV= NA,Is.Duplicate= "Yes",Latitude = NA,Longitude= NA,
  add_row(Scan.Date = "2022-12-08",Scan.Time = "12:52:10.810", Download.Date = "9/16/2022",Download.Time = "11:30:41",
          Reader.ID = "A4",Antenna.ID = 1,HEX.Tag.ID = "384.358D14F739",
          DEC.Tag.ID = "900.230000999999",Temperature.C = NA,Signal.mV= NA,Is.Duplicate= "Yes",Latitude = NA,Longitude= NA,
  add_row(Scan.Date = "2022-12-09",Scan.Time = "12:52:10.810", Download.Date = "9/16/2022",Download.Time = "11:30:41",
          Reader.ID = "A5",Antenna.ID = 1,HEX.Tag.ID = "384.358D14F739",
          DEC.Tag.ID = "900.230000999999",Temperature.C = NA,Signal.mV= NA,Is.Duplicate= "Yes",Latitude = NA,Longitude= NA,
```

- Comment out the lines of code that remove the dummy tag from the data (add them back in when there is actual data for the antenna).
  - In Runscript:

```
355
356  ### taking dummy tag out
357
358  Biomark <- Biomark #%>%
359    #filter(!DEC.Tag.ID %in% c("900.230000999999"))
360  cleanedRelease <- cleanedRelease #%>%
361    #filter(!TagID %in% c("230000999999"))
362  Stationary <- Cleaned_Stationary_FishdetectionsOnly #%>%
363    #filter(!TAG %in% c("900230000999999"))
364
```

  - In "get_movements_function"

```
6    dailyMovementsTable <- combined_events_stations %>%
7      #### removing dummy tag
8      #filter(!TAG %in% c("230000999999")) %>%
9      select(Date, Datetime, TAG, det_type, Event, ET_STATION
```

  - In Ind_tag_enc_hist_wide_summary_function.r:

```
149
150   #### dummy rows removal:
151   encountersAndRelease6 <- encountersAndRelease6# %>%
152     #filter(!TAG %in% c("230000999999"))
153
```

  - In All_combined_events.R

```
147  condensedAllEventsWithReleaseandEnvironmentalInfo <- condensedAllEventsWithReleaseandEnvironmentalInfo
148    replace_na(list(Species = "No Info", ReleaseSite = "No Info",
149                    Site = "No Site Associated")) %>%
150    dplyr::filter(!TAG %in% c("230000999999"))
```

  - In AllEncountersMod, twice

```
421            #### filter dummy row
422            all_events_filtered <- all_events_filtered #%>%
423              #filter(!TAG %in% c("230000999999"))
424

464          output$alleventsfrequencies1 <- renderDataTable({
465            frequenciesSummarized <- all_events_data() %>%
466              #filter(!TAG %in% c(230000999999)) %>%
```

All_combined_events function

- In this function, add the following line to the case_when() function based off the Frontend and Backend variable codes

```
biomarkCleaned <- Biomark %>%
  mutate(TAG = str_replace(DEC.Tag.ID, "\\.", ""),
         # i wish this could be a join, but when there are 2 dif codes (A1, B1, etc) used for backend na
         Reader.ID = case_when(
           Reader.ID %in% TestBiomarkAntennaBackendSiteCode ~ TestBiomarkAntennaFrontendSiteCode,
           Reader.ID %in% WindyGapBypassAntennaBackendSiteCode ~ WindyGapBypassAntennaFrontendSiteCode,
```

Ind_tag_hist_summary_wide_function:

- Add in your antennaFrontend Variable in column order

```
#column order is just nice to have for the user
columnOrder <- c(RedBarnFrontendCodes, HitchingPostFront
                 MobileRunFrontendCodes, WindyGapBypassA
                 GranbyDiversionAntennaFrontendSiteCode,
                 TestBiomarkAntennaFrontendSiteCode)
```

- Add the frontend site code to these lines of code. If it's a biomark antenna, add it to "TotalBiomark", and if its a Stationary antenna, add to TotalStationary

```
    mutate(
      TotalEncounters = rowSums(select(., all_of(c(RedBarnFrontendCodes, HitchingPostFronte
                                                   ConnectivityChannelDownstreamFrontendCod
                                                   ConnectivityChannelUpstreamFrontendCodes
                                                   WindyGapBypassAntennaFrontendSiteCode, W
                                                   RiverRunAntennaFrontendSiteCode, FraserR
                                                   TestBiomarkAntennaFrontendSiteCode))) ==
      TotalAntennas = rowSums(select(., all_of(c(RedBarnFrontendCodes, HitchingPostFrontend
                                                 ConnectivityChannelDownstreamFrontendCodes
                                                 ConnectivityChannelUpstreamFrontendCodes,
                                                 WindyGapBypassAntennaFrontendSiteCode, Win
                                                 GranbyDiversionAntennaFrontendSiteCode,
                                                 RiverRunAntennaFrontendSiteCode, FraserRiv
                                                 TestBiomarkAntennaFrontendSiteCode))) == T
      TotalStationary = rowSums(select(., all_of(c(RedBarnFrontendCodes, HitchingPostFronte
                                                   ConnectivityChannelDownstreamFrontendCod
                                                   ConnectivityChannelUpstreamFrontendCodes

      TotalMobile = rowSums(select(.,  all_of(MobileRunFrontendCodes)) == TRUE),
      TotalBiomark = rowSums(select(., all_of(c(WindyGapBypassAntennaFrontendSiteCode, Wind
                                                GranbyDiversionAntennaFrontendSiteCode,
                                                RiverRunAntennaFrontendSiteCode, FraserRive
                                                TestBiomarkAntennaFrontendSiteCode))) == TR
      TotalRedBarn =rowSums(select(.,  all_of(RedBarnFrontendCodes)) == TRUE),
      TotalHitchingPost = rowSums(select(.,  all_of(HitchingPostFrontendCodes)) == TRUE),
```

With that, you are ready to Run/Knit the CreateFlatFilesRunscript. Once the data is updated, run the app. Check to make sure your antenna has been integrated correctly in the data by filtering for the dummy tag in the Encounters dataframes, movements, and States tabs where it will also have dummy Release Data.

Show 10 entries Search:

TAG	Date	Time	Datetime	Event	Species	Release_Length	Release_Weight	ReleaseSite	Release_Date
230000999999	2020-09-01	12:00:00	2020-09-01T12:00:00Z	Release	LOC	566	600	Fraser River Ranch	2020-09-01
230000999999	2022-12-07	12:52:10.810	2022-12-07T12:52:10Z	RR1	LOC	566	600	Fraser River Ranch	2020-09-01
230000999999	2022-12-08	12:52:10.810	2022-12-08T12:52:10Z	FC1	LOC	566	600	Fraser River Ranch	2020-09-01
230000999999	2024-05-24	12:52:10.810	2024-05-24T12:52:10Z	T1	LOC	566	600	Fraser River Ranch	2020-09-01

Showing 1 to 4 of 4 entries Previous 1 Next

Figure 1all Encounters History with Dummy Data

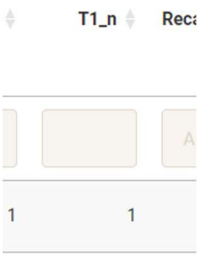


Figure 2 colunm was automatically created in All EncountersHistory SummaryWide for that new antenna

Show 10 entries

Date	weeks_since	TAG	State	det_type	ReleaseSite	Species
2020-09-01	0	230000999999	B	Release	Fraser River Ranch	LOC
2022-12-07	118	230000999999	B	River Run Biomark Antenna	Fraser River Ranch	LOC
2024-05-24	194	230000999999	B	Biomark test Antenna	Fraser River Ranch	LOC

Showing 1 to 3 of 3 entries

Figure 3States df with dummy data

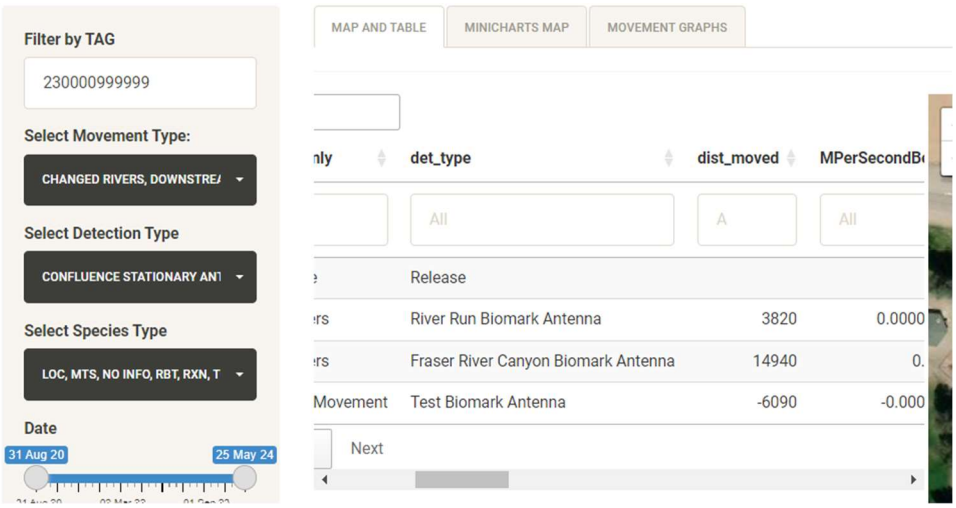


Figure 4 Movements table with dummy data

Adding or updating layers on the map

- In GIS, go to the GIS database and export desired layer as shapefile, saving it in "gis" folder within the app directory
    - Right click on layer in left column, scroll down to data -> export
- Using the sf package, read in the .shp file with the name of your layer

```
layerLocation <- file.path("./gis/")
latLongCRS <- st_crs("+proj=longlat +datum=WGS84 +no_defs") #should be same as +init=epsg:4326

antenna_sites <- st_transform(read_sf(file.path(layerLocation, "antenna_sites1.shp")), latLongCRS)
```

- The espg:4326 converts the projection to lat/longs able to be plotted with the leaflet package
- If the layer is very large, it might be good to convert the .shp file to a .rds file, which decreases resolution. For example, the stations file is a .rds file and was converted below. It keeps 10% of the original resolution

```
stations_10m <- readOGR(dsn = layer_location, layer = "stations_10m")
stations_10m <- sp::spTransform(stations_10m, CRS("+init=epsg:4326"))
simple_stations1 <- ms_simplify(stations_10m, keep = .1)
write_rds(simple_stations1, file = file.path(paste0(layer_location,"/"), paste0("simple_stations.rds")))
```

- It is read in like so and is much faster than when read in as a shapefile

```
simple_stations2 <- read_rds(file.path("./gis/simple_stations.rds"))
```

- If you make a new layer (not just updating an existing antenna, stations or stream centerline layer that is already included on the map) that you want to display, you'll have to add it to the code that makes the map, found in movementsMod.R.
- Add_polylines is used to bring in lines like stream_centerline, addMarkers is used for SpatialPoints, addPolygons is used for polygons.
- Add a Group argument to the data and add it in the addLayersControl function to make sure it shows up

```
addAwesomeMarkers(data = releasesites@coords,
                  icon = release_icons,
                  clusterOptions = markerClusterOptions(),
                  label = releasesites@data$ReleaseSit,
                  popup = paste("Release Date1:", releasesites@data$ReleaseDat, "<br>","Release Da
                  group = "Release Sites") %>%
addPolylines(data = simple_stations2,
             label = simple_stations2@data$ET_STATION,
             labelOptions = labelOptions(noHide = T, textOnly = TRUE, style = label_style),
             group = "Stations (m)") %>%
addLayersControl(overlayGroups = c("Antennas", "Detections", "Release Sites", "Stream Centerlines"
hideGroup(c("Stream Centerlines", "Stations (m)", "Antennas", "Release Sites", "Mobile Reaches"))
```

Adding New Antenna to map

The data will still be displayed spatially on the movements map even without an updated antenna file, but if you want to display the antenna on the map (and you should), then here a couple options to do this. The first and BEST option is to add it in GIS, then export the layer (see Spatial Data) and read in that later.

The second option is to modify the already existing Antenna layer within R. This can be done with already-written code in map_polygon_readins.R starting on line 57.

- Modify the UTMs and other relevant fields to fit the antenna.

```
57  #####adding new antenna to data
58
59  utm_x <- 420727
60  utm_y <- 4437229
61  new_point <- st_sfc(st_point(c(utm_x, utm_y)), crs = 32613) # EPSG:32613 is UTM zone 13N
62
63  # Transform the new point to the desired CRS (latLongCRS)
64  new_point_transformed <- st_transform(new_point, st_crs(latLongCRS))
65
66  new_row <- tibble(
67    OBJECTID = NA,
68    SiteName = NA,
69    Objective = NA,
70    StudyPerio = NA,
71    SiteLabel = NA,
72    AntennaNam = NA,
73    WaterName = NA,
74    ChannelWid = NA,
75    UTM_X = utm_x,
76    UTM_Y = utm_y,
77    Notes = NA,
78    geometry = new_point_transformed
79  )
```

- Add this row to the data and save it in the correct location with st_write. This WILL overwrite the previous file if saved with the same name as the previous file.

```
1  antenna_sites1 <- antenna_sites %>%
2    add_row(new_row)
3
4  output_file <- file.path(layerLocation, "antenna_sites1.shp")
5
6  # Write the updated sf object to a new shapefile
7  st_write(antenna_sites, output_file)
```

- Once the new layer is saved, comment out those lines of code from lines 57 to 87 to make sure these lines aren't ran again.

Relevant Runscript and App Functions

Here are some of the main functions of the app.

Note: some variables used in the functions are not included as arguments, but are read in earlier in the Runscript so should always exist before running.

All_Combined_events_function.R

All_combined_events_function()

- Main function that combines and cleans detections and release files

- Inputs
    - Stationary: combined Stationary detections
        - WGFP_Raw_yyyymmdd.rds
        - UTMs are added with addUTMsAndReformatStationary(), which relies on WGFP Metadata.xlsx for the correct UTM assignments
    - Mobile: all Mobile data
        - WGFP_Mobile_Detect_AllData.csv
    - Biomark: combined Biomark detections
        - Biomark_Raw_yyyymmdd.rds
    - Release: all release info
        - WGFP_ReleaseData_Master_yyyymmdd.csv
    - Recapture: all recapture info
        - WGFP_RecaptureData_Master_ yyyymmdd.csv
- Returns:
    - All_Events: cleaned dataset of all 5 input datasets, containing every detection/event from each tag that hit an antenna, complete with release info.
    - WGFP Clean: a clean dataset of all stationary detections, no weird tags or marker tags, or duplicate rows. All timestamps and dates are in the same format. 900_ is taken off of the tags
    - Marker_Tag_data: file of just marker tags, to be used in QAQC
    - Recaps_detections: a file of all antenna detections and recaptures, but no release data. This is just to be used in enc_hist_wide_summary_function

About the Code

Stationary data has already been semi "cleaned" in the WGFPCombineFiles app. The function addUTMsAndReformatStationary() in this app includes changing antenna names based off metadata, formatting dates, assigning UTMs for each antenna, and removing duplicate rows if there are any.

Biomark dataset is then cleaned, including changing the reader ID since we want it in the format specified in the metadata file, changing date format, filtering out test tags and marker tags, assigning UTMs, and removing duplicate rows.

```
# biomark cleaning, getting dates into uniform format,
biomarkCleaned <- Biomark %>%
  mutate(TAG = str_replace(DEC.Tag.ID, "\\.", ""),
         # i wish this could be a join, but when there are 2 dif codes (A1, B1, etc) used for backend name, this is a bit simpler
         Reader.ID = case_when(
           Reader.ID %in% WindyGapBypassAntennaBackendSiteCode ~ WindyGapBypassAntennaFrontendSiteCode,
           Reader.ID %in% WindyGapAuxiliaryAntennaBackendSiteCode ~ WindyGapAuxiliaryAntennaFrontendSiteCode,
           Reader.ID %in% GranbyDiversionAntennaBackendSiteCode ~ GranbyDiversionAntennaFrontendSiteCode,
           Reader.ID %in% RiverRunAntennaBackendSiteCode ~ RiverRunAntennaFrontendSiteCode,
           Reader.ID %in% FraserRiverCanyonAntennaBackendSiteCode ~ FraserRiverCanyonAntennaFrontendSiteCode,
           TRUE ~ Reader.ID),
         #make a column for Scan>Date if parentheses are detected in the string, that means the format is in mdy
         # and we want to convert it to YYYYMMDD format. elsewise, leave it as is
         Scan.Date = ifelse(str_detect(Scan.Date, "/"),
                            as.character(mdy(Scan.Date)),
                            Scan.Date)
  ) %>%
  #we want to filter out test tags here, but not marker tags
  filter(!TAG %in% test_tags) %>%
  #get UTMs based off what is in metaDataTable
  #left_join() is faster than merge()
  left_join(wgfpMetadata$AntennaMetadata, by = c("Reader.ID" = "FrontendSiteCode")) %>%
  distinct()
```

Then Mobile, Stationary, and Biomark columns are all renamed and then they are joined together. Mobile data requires a little Tag cleaning but not much.

```
###Create one big clean dataset
WGFP_condensed <- WGFP_Clean %>%
  select(DTY, ARR, TAG, SCD, UTM_X, UTM_Y) %>%
  rename(Scan_Date = DTY, Scan_Time = ARR, Site_Code = SCD, UTM_X = UTM_X, UTM_Y = UTM_Y)


Biomark_condensed <- biomark2 %>%
  mutate(TAG = ifelse(str_detect(TAG, "^900"), str_sub(TAG, 4,-1), TAG)) %>%
  select(Scan.Date, Scan.Time, TAG, Reader.ID, UTM_X, UTM_Y) %>%
  rename(Scan_Date = Scan.Date, Scan_Time = Scan.Time, Site_Code = Reader.ID, UTM_X = UTM_X, UTM_Y = UTM_Y)

Mobile_condensed <- Mobile %>% #gonna have to just change to mobile eventually
  rename(TAG = TagID) %>%
  mutate(TAG = ifelse(str_detect(TAG, "^900"), str_sub(TAG, 4,-1), TAG),
         Date = ifelse(str_detect(Date, "/"),
                       as.character(mdy(Date)),
                       Date)) %>% #end of mutate
  select(Date, Time, TAG, Ant, UTM_X, UTM_Y) %>%
  rename(Scan_Date = Date, Scan_Time = Time, Site_Code = Ant)


WG_bio <- bind_rows(WGFP_condensed,Biomark_condensed)
All_detections <- bind_rows(WG_bio, Mobile_condensed)
```

Data is filtered to only include detections past the study start date, and datetime is put in a good format

```
cleanedAllDetections <- allDetections %>%
  filter(Scan_Date >= as.Date("2020-08-06")) %>% #right before the first date of marker tag detections on stationary antennas
  mutate(
    Scan_DateTime = ymd_hms(paste(Scan_Date, Scan_Time))) %>%
  select(Scan_Date, Scan_Time, Scan_DateTime, TAG, Site_Code, UTM_X, UTM_Y)
```

Release and recapture files are brought in and timestamps are cleaned. There is need for this because sometimes the way that times are entered in the spreadsheet are not uniform. Columns are renamed for joining.

```
#getting timestamps in order and getting relevant columns
cleanedRelease <- Release %>%
  rename(TAG = TagID) %>%
  mutate(TAG = str_trim(TAG),
         Date = mdy(Date),
         DateTime = lubridate::ymd_hms(paste(Date, Time))) %>%
  select(RS_Num, River, ReleaseSite, Date, Time, DateTime, UTM_X, UTM_Y, Species, Length, Weight, TAG, TagSize, Ant, Event)

#getting timestamps in order and getting relevant columns

cleanedRecaptures <- Recaptures %>%
  rename(TAG = TagID) %>%
  filter(!Date %in% c("", " ", NA)) %>%
  mutate(TAG = str_trim(TAG),
         Date = mdy(Date),
         DateTime = ymd_hms(paste(Date, Time))) %>%
  select(RS_Num, River, RecaptureSite, DateTime, Date, Time, UTM_X, UTM_Y, Species, Length, Weight, TAG, TagSize, Ant, Event) %>%
  rename(
    Recap_Length = Length,
    Recap_Weight = Weight
  )
```

Reformatting the all detections file to also be ready to join, then binding release and recap data to the df. Binding the release df to the main df makes it so there will be an event for “release”, and then left joining will then give release info for each fish for each detection.

```
#getting all detections file ready to merge with encounters
allDetectionsForBinding <- cleanedAllDetections %>%
  mutate(Date = ymd(Scan_Date)) %>%
  rename(
    Time = Scan_Time,
    DateTime = Scan_DateTime,
    Event = Site_Code
  )

# this file is used in enc_hist_summary_wide
recapturesAndDetections <- bind_rows(allDetectionsForBinding, cleanedRecaptures)

allEvents <- bind_rows(recapturesAndDetections, cleanedRelease)

# bind rows vs left join; bind rows will make it so there is a "release" or "recapture" event and also make columns with relevant

#fills in release info so it is known at any row of detection
allEventsWithReleaseInfo <- left_join(allEvents, cleanedRelease, by = c("TAG"))
```

Then there's some minor formatting/renaming for display purposes in the app

```
condensedAllEventsWithReleaseInfo <- allEventsWithReleaseInfo %>%
  select(Date.x, Time.x, DateTime.x, TAG, Event.x, Species.y, Length.y, Weight.y, ReleaseSite.y, Date.y,
  rename(Release_Date = Date.y,
         Date = Date.x,
         Time = Time.x,
         Datetime = DateTime.x,
         Event = Event.x,
         Species = Species.y,
         Release_Length = Length.y,
         Release_Weight = Weight.y,
         ReleaseSite = ReleaseSite.y,
         UTM_X = UTM_X.x,
         UTM_Y = UTM_Y.x) %>%
  #gets rid of all duplicate rows but keeps all info
  distinct(Datetime, TAG, Event, .keep_all = TRUE)

Tags_only <- cleanedRelease %>%
  select(TAG)
```

This line makes sure that the tags displayed are from the release file. There is a tab in the QAQC tab to see unknown tags histories.

```
#makes sure all events are from tags ONLY in the release file
filled_in_release_rows_condensed <- left_join(Tags_only, filled_in_release_rows_condensed, by = "TAG")
```

PT/environmental data is added to the data with combineEnvironmentalandDetectionsData() function. USGS data are attached to each detection within 13 hour of the detection, and PT data is associated to that detection within 1 hours.

```
###add temp/environmental data to that
#these arguments come from the createFlatFilesRunscript
condensedAllEventsWithReleaseandEnvironmentalInfo <- combineEnvironmentalandDetectionsData(Detections = condensedAllEvents
                                                    allPressureTransducerDataWithDischarge = allPressureTransducerDataWithDischarge
                                                    DischargeData = windyGap
                                                    )
```

This data frame condenses detections down to their daily summary; so it will take the first detection a fish had that day, and the last detection a fish had that day, and all unique detections in between. This reduces a fish to the "most relevant" detections. This df is ultimately used in the movements df, but first in joining with the Stations Data, which is the next function. WARNING: It's important to note that if a fish has a day where the movement sequence is like "RB1, RB2, HP4, HP3, HP4, RB2, HP3, HP4", then for that day, the sequence will register as "RB1, RB2, HP4, HP3, HP4".

```
### This is getting the events dataframe to only the data relevant for joining with stations

all_events_relevant_to_stations <- filled_in_release_rows_condensed %>%
  #this part is for making sure the sequence of events will make sense
  # if there's no tag input then have to group_by TAG as well
      group_by(Date, TAG) %>%
        mutate(first_last = case_when(Datetime == min(Datetime) ~ "First_of_day",
                                      Datetime == max(Datetime) ~ "Last_of_day",
                                      Datetime != min(Datetime) & Datetime != max(Datetime) ~ "0")
        ) %>%
        ungroup() %>%
        distinct(TAG, Event, Date, first_last,  UTM_X, UTM_Y, .keep_all = TRUE) %>%
        arrange(Datetime) %>%
        select(-first_last)
```

Little bit more reformatting to replace NA to work with filters within the app. This is the final df that is shown in Encounters History tab

```
  #this is the final df

  #Change na to "No info" in select columns so that it will register with the Picker input in the app
  #pretty sure that's just a bug on the DT or shinyWidgets end that it can't select by NA
  # 87 rows were not even showing up on the all_events app because the Species was NA -12/14/21 SG
  condensedAllEventsWithReleaseandEnvironmentalInfo <- condensedAllEventsWithReleaseandEnvironmentalInfo %>%
    replace_na(list(Species = "No Info", ReleaseSite = "No Info",
                    Site = "No Site Associated")) %>%
    dplyr::filter(!TAG %in% c("230000999999"))
```

Growth rates per year are also calculated with the custom function "getGrowthRates()" using Release and Recaptures. To account for leap years, 1 year is defined as 52.25 weeks. This data is displayed in the QAQC tab -> Release/Recap Lengths/Weights

```
###get growth rates for QAQC tab
growthRates <- getGrowthRates(Release = Release, Recaptures = Recaptures)
```

combineEnvironmentalandDetectionsData.R

combineEnvironmentalandDetectionsData()

- Combines detections, release etc with environmental data
- Inputs:
  - Detections: dataframe with combined Release, Mobile, Stationary, and Biomark information with just info from Tags in release file. Created in All_Combined_events_function()
  - allPressureTransducerDataWithDischarge: Combined Pressure transducer and USGS Discharge data, joined in CreateFlatFiles_Runscript.rmd
  - DischargeData: dataframe of USGS discharge only
- Outputs:
  - AllData: Combined detections, Pressure Transducer and USGS data with discharge readings within 15 min detections (up to 13 hours), within 1 hour for Pressure Transducer data.

About the Code

Since the timestamps for detection data and environmental data do not often line up exactly, we need to break up the detection data and join to environmental data in different ways. This is also complicated by the fact that we need to join Site-Specific Pressure Transducer data to their correct sites. We can do rolling joins from the data.table package to get match the Environmental data to the nearest detection timestamps, then make sure those times are within the buffer we specify (13 hours for USGS data with no Pressure Transducer data associated, 1 hour for Pressure Transducer data). We can use dplyr::inner_join() for the exact timestamp matches.

First we attach the data.table package to be used for rolling joins later in the code. Since this package has a lot of overlap in functions naming with other packages used in the app, we'll detach it at the end of this function.

We also get the Pressure transducer site name for all associated detections, defined in WGFP Metadata.xlsx. This is used to join PT and Discharge data to Detections.

```
#using this package for the main nearest naighbor timestamp join,
#but it has a lot of cross functionality with other packages especiallly lubridate
#so we're going to detach it at the end
library(data.table, quietly = TRUE,warn.conflicts = FALSE)
#########joining to get PT siteName from metadata
DetectionswithPTSiteName <- Detections %>%
  #add pressure transducer site name to have a key to join with PT data
  #this part works as long as the site name in the metadata is what is used in the PT files as well
  #as more sites are added, we'll have to make sure the names in the pressure transducer files
  #are the same as in the metadata "PressureTransducerSiteName" field
  left_join(wgfpMetadata$AntennaMetadata[,c("FrontendSiteCode", "PressureTransducerSiteName")], by = c("Event" = "FrontendSiteCode")) %>%
  rename(SiteName = PressureTransducerSiteName)
```

Discharge data is checked to make sure there's not any NA rows in the DateTime field

```
###remove rows here that have NA's associated with every field except datetime
keyColumns <- c("dateTime")

#filters rows from the data frame allPressureTransducerDataWithDischarge based on whether they have missing values only in specified columns (keyColumns),
#filtered_PTData is the dates where there all all NA values in rest of the columns
filtered_PTData <- allPressureTransducerDataWithDischarge[rowSums(is.na(allPressureTransducerDataWithDischarge[, setdiff(names(allPressureTransducerDataWithDisc
#then we anti_join that with the original PT data with discharge to get df of data with datetimes that have some sort of relevant data
allPressureTransducerDataWithDischargeNoNA <- allPressureTransducerDataWithDischarge %>%
  anti_join(filtered_PTData) %>%
  suppressMessages()
```

dateTime formatted into correct format and timezone is checked to make sure our data lines up with what is coming from USGS. This is necessary in order to use rolling join data based on times. We then subset to get data that do not directly match a Pressure Transducer or USGS Discharge timestamp. We're going to start with this data first, referred to as "NotExactTimeStampMatches"

```
PTData <- allPressureTransducerDataWithDischargeNoNA %>%
  mutate(Datetime = as.POSIXct(dateTime)) %>%
  rename(SiteName = Site)

#check timezone
if(attr(PTData$Datetime, "tzone") != attr(DetectionswithPTSiteName$Datetime, "tzone")){
  #force detection df to correct zone
  PTData$Datetime <- lubridate::force_tz(PTData$Datetime, tzone = "UTC")

}
#these are all the detections from the original Detection file that do not exactly match a PT or discharge recording timestamp
notExactTimestampMatchesDetections <- DetectionswithPTSiteName %>%
  anti_join(PTData, by = c("Datetime")) %>%
  suppressMessages()
```

Join 1: Rolling (NotExactTimeStampMatches PT Associated Detections)

Joining Pressure Transducer/Discharge data to stationary antennas associated with a Pressure Transducer where detection and PT/Discharge timestamps do not line up

We subset NotExactTimeStampMatches even further to get just Stationary Antenna detections that don't have exact timestamp matches with PT/Discharge Readings. Then we convert this data and Pressure transducer/Discharge data into data.table objects in preparation for a rolling join. We set key columns "Datetime" and "SiteName" to use for our join. Order matters. Then we do a rolling join, where detection data is joined to PT/Discharge data based on the SiteName and the nearest Datetime stamp.

```
# rolling join on just specific site stuff --------------------------------
  #the rolling join with 2 can only take those with the right site name
  notExactTimestampMatchesDetectionsStationaryOnly <- notExactTimestampMatchesDetections %>%
    filter(!is.na(SiteName))

  #make PT data and timestamps into data.table objects so that we can perform rolling join
  notExactTimestampMatchesDetectionsStationaryOnly <- data.table(notExactTimestampMatchesDetectionsStationaryOnly)
  PTData <- data.table(PTData)

  #set keycols
  keycols <- c("Datetime", "SiteName")
  setkeyv(notExactTimestampMatchesDetectionsStationaryOnly, keycols)
  setkeyv(PTData, keycols)

  #perform a rolling join from data.table
  #first joins on sitename then datetime
  #for data where we have a stationary site attached, discharge data is from the closest hour. Otherwise, dishcarge data is within 15 minutes
  #gives df of envrionmental data and detections with closest timestamps
  notExactTimestampMatchesDetectionsWithClosestEnvironmentalReading <- PTData[notExactTimestampMatchesDetectionsStationaryOnly,
                                                                              roll = "nearest", on = .(SiteName,Datetime), nomatch = NULL]
```

If any Pressure transducer or Discharge readings are outside 1 hour difference of the time of detection, we've decided that's too much variation for a reliable reading and we change those environmental readings to NA

```
notExactTimestampMatchesDetectionsWithClosestEnvironmentalReading <- notExactTimestampMatchesDetectionsWithClosestEnvironmentalReading %>%
  relocate(dateTime, Datetime) %>%
  rename(environmentalDataMeasurementTime = dateTime)

#change environmental data columns to NA that are outside 1 hour time frame
#cols to change to NA: all that include a "_" and USGSDischarge
columnstoChange <- c(colnames(PTData)[grepl("_", colnames(PTData))], "USGSDischarge")

notExactTimestampMatchesDetectionsWithClosestEnvironmentalReadingWithin1Hour <- notExactTimestampMatchesDetectionsWithClosestEnvironmentalReading %>%
  mutate(timeDifference = ifelse(abs(difftime(environmentalDataMeasurementTime, Datetime, units = c("hours"))) >= 1, 1, 0),
         across(all_of(columnstoChange), ~ ifelse(timeDifference == 1, NA_real_, .))
  )
```

We then subset to get all the detection data from stationary Antenna sites with Pressure transducer readings within an hour of detection. We are now done with this data, which will get joined at the end to the rest of the data we need to assign environmental data to.

```
#this is one of the df's to join
#this is stationary PT readings for all detections within 1 hour, including discharge
#if there is 1 valid reading in any of the speificed columns, the row is kept
#but rows are removed if all the entires in the specified columns are NA
notExactTimestampMatchesDetectionsWithClosestEnvironmentalReadingWithin1HourValidDataOnly <- as.data.frame(notExactT
  filter(rowSums(is.na(.[columnstoChange])) < length(columnstoChange))
```

Join 2: Rolling (NotExactTimeStampMatches Non-PT Associated Detections)

Joining detections without a Pressure Transducer associated with that site to just Discharge data, where detection timestamp does not line up exactly with Discharge reading timestamp

All "NotExactTimeStampMatches" Data is anti_joined with previous finished "Not Exact timestamp PT-associated sites" data from above to get remainder of the "Not Exact timestamp" data that doesn't line up with discharge reading or have a Pressure Transducer associated with the site.

Data.table objects are created and keyCol "Datetime" is specified. Since we're just joining these data with nearest Discharge reading and they don't have Pressure Transducers associated, we don't need to specify the site like in rolling join 1 above.

Join is executed to nearest Datetime.

```
#want to buffer by 13 hours for just discharge


restOftheNotExactTimestampMatches <- notExactTimestampMatchesDetections %>%
  anti_join(notExactTimestampMatchesDetectionsWithClosestEnvironmentalReadingWithin1HourValidDataOnly,
            by = c("Datetime", "Event", "TAG", "UTM_X", "UTM_Y")) %>%
  arrange(SiteName) %>%
  suppressMessages()

restOftheNotExactTimestampMatches <- data.table(restOftheNotExactTimestampMatches)
DischargeDataforJoin <- data.table(DischargeData) %>%
  mutate(Datetime = dateTime)

#set keycols
keycols <- c("Datetime")
setkeyv(restOftheNotExactTimestampMatches, keycols)
setkeyv(DischargeDataforJoin, keycols)

restOftheNotExactTimestampMatchesWithDischargeReading <- DischargeDataforJoin[restOftheNotExactTimestampMatches, roll = "nearest", on = .(Datetime),
                                                                               nomatch = NULL]
```

Here we buffer Discharge reading by 13 hours around each detection, instead of just the 1 hour that we used for Pressure Transducers. If it's outside 13 hours it get changed to NA.

```
restOftheNotExactTimestampMatchesWithDischargeReading <- restOftheNotExactTimestampMatchesWithDischargeReading %>%
  relocate(dateTime, Datetime) %>%
  rename(environmentalDataMeasurementTime = dateTime)

#change environmental data columns to NA that are outside 13 hour time frame
#cols to change to NA: includes USGSDischarge
columnstoChange <- c("USGSDischarge")


##this is the rest of the not exact time stamp entries to fill in the gaps
restOftheNotExactTimestampMatchesWithDischargeReading <- restOftheNotExactTimestampMatchesWithDischargeReading %>%
  mutate(timeDifference = ifelse(abs(difftime(environmentalDataMeasurementTime, Datetime, units = c("hours"))) >= 13, 1, 0),
         across(all_of(columnstoChange), ~ ifelse(timeDifference == 1, NA_real_, .))
  )
```

Join 3: Inner (Exact TimeStamp Matches, no PT Data Associated)

For detections without a Pressure Transducer Associated with that site, we can user dplyr::inner_join() to associate discharge data with those detections with the exact timestamp

```
# getting environmental data for non-stationary events with exact timestamp matches----------------------

  #gives exact matches of detections NOT at the stationary sites: release, mobile rus, biomark
  #can only join with USGS data here
  #currently joins discharge data to this no matter if the event is above or below the dam
  exactMatchesNotPTdata <- DetectionswithPTSiteName %>%
    filter(!SiteName %in% na.omit(unique(wgfpMetadata$AntennaMetadata$PressureTransducerSiteName))) %>%
    inner_join(DischargeData, by = c("Datetime" = "dateTime")) %>%
    mutate(environmentalDataMeasurementTime = Datetime)
```

Join 4: Inner (Exact TimeStamp Matches, PT Associated but Timestamp outside desired PT range)

For the rest of the detection data that do have a pressure transducer associated with that site but there are no PT readings from these exact timestamps (just Discharge), we join with the Discharge data for the exact timestamp matches.

```
 # getting environmental data for all other detections that do not  have a stationary antenna site attached --------

 #this should be redundant when we actually have PT data from connectivity channel
 #should be discharge only right now unless we add some stuff
 ptData_noSite <- PTData %>%
   filter(is.na(SiteName))

 exactMatchesatSiteNoPTSite <- DetectionswithPTSiteName %>%
   filter(SiteName %in% na.omit(unique(wgfpMetadata$AntennaMetadata$PressureTransducerSiteName))) %>%
   inner_join(ptData_noSite, by = c("Datetime" = "dateTime")) %>%
   suppressMessages() %>%
   mutate(environmentalDataMeasurementTime = Datetime)
```

Join 5: Inner (Exact Timestamp Matches, PT Associated and Timestamp exactly lines up)

For the rest of the detection data that do have a pressure transducer associated with that site, we join with the PT/Discharge data for the exact timestamp matches by Site and Time

```
# gets environmental data associated with stationry antenna sites attached to stationary sites that have an exact timestamp match


exactMatchesWithPTdata <- DetectionswithPTSiteName %>%
  filter(SiteName %in% na.omit(unique(wgfpMetadata$AntennaMetadata$PressureTransducerSiteName))) %>%
  inner_join(PTData, by = c("Datetime" = "dateTime", "SiteName")) %>%
  mutate(environmentalDataMeasurementTime = Datetime)
```

Binding all Data together

All data from above is reformatted to desired columns before binding together. If done correctly, allData should have the same number of rows as the original Detections input, but with desired PT/Discharge data

```
###then bind together notExactTimeMatches at statoinary sites, exact time matches not at stationary sites, and exact time matches at Stationary sit
#this should give df the same size as the original condensedAllEventswithReleaseInfo (or Detections) with environmental variables attached
desiredColumns <- c(colnames(Detections), colnames(allPressureTransducerDataWithDischarge), "environmentalDataMeasurementTime", "SiteName")

#align columns is another function that gets columns to the right names
df_list <- list(
  "exactTimestampMatchesWithPTdata1" = alignColumns(exactMatchesWithPTdata, desiredColumns, Detections),
  "exactTimestampMatchesatSiteNoPTSite1" = alignColumns(exactMatchesatSiteNoPTSite, desiredColumns, Detections),
  "exactTimestampMatchesNotPTdata1" = alignColumns(exactMatchesNotPTdata, desiredColumns, Detections),
  "notExactTimestampMatchesDetectionsWithClosestEnvironmentalReadingWithin1Hour1" = alignColumns(notExactTimestampMatchesDetectionsWithClosestEnvir
  "notExactTimestampMatchesDetectionsStationaryValidDataWithClosestEnvironmentalReadingWithin1Hour1" = alignColumns(restOftheNotExactTimestampMatch
  )

## some rows have an entry for ptData at an exact timestamp, but the variables are all NA so it gets put under "notExactTimestampMatchesDetectionsW
#however since there's an exact match it also gets put under "exactMatchesNotPTdata"
#this leads to duplicate entries in the final df, just with different environmental timestamp fields
#so that's why here we filter out those rows (out of 2.6 million, it was just 3 rows) based on having a different envrionmnetal timestamp
#bunc hof different ways to bind dfs in a list together
allData <- do.call(rbind, df_list)
```

Data.table package is detached, data slightly reformatted

```
#removing row names, though I can see how they'd be useful
row.names(allData) <- NULL
allData <- allData %>%
  select(-Site, -dateTime) %>%
  rename(Site = SiteName) %>%
  dplyr::distinct(across(-environmentalDataMeasurementTime), .keep_all = TRUE)

detach("package:data.table", unload=TRUE)
```

combineEnvironmentalAndSiteVisitData.R

combineEnvironmentalAndSiteVisitData()

- Combined Site Visit and Environmental Data
- Inputs
  - WGFPSiteVisitsFieldData: All Site visit data, reformmated/cleaned with wrangleSiteVisitData() in the Runscript
  - EnvironmentalData: allPressureTransducerDataWithDischarge, including USGS Data.
- Returns
  - allRowsPTDataSiteVisits: Combined site visit and Environmental data, where PT readings correspond the site within 13 hours. Sites without Pressure Transducer associated are left blank.
    - Used in "Pressure Transducer, USGS, and Detection Distance" tab as well as QAQC tab

About the Code

Similar to the combineEnvironmentalandDetectionsData function above, Environmental data timestamps do not often line up exactly with our site Visit times, so we have to break up the data in order to associate desired data.

First we reformat the field data date/time to one suitable for joining and working with.

New Environmental data columns are created to keep track of time recorded and Notes after joining.

Site Visit data with exact timestamp matches are joined using dplyr::inner_join(), and the rest of the data is subset to be rolling joined to the nearest dateTime

```
#expect NAs here so suprressing these warnings
suppressWarnings({
  WGFPSiteVisitsFieldData1 <- WGFPSiteVisitsFieldData %>%
    mutate(dateTime = lubridate::ymd_hms(paste(Date, Time)),
           fieldDataNotes = Notes)
})

#prep for rolling join with PT data
EnvironmentalData_prepped <- EnvironmentalData %>%
  dplyr::filter(!is.na(Site)) %>%
  mutate(ptTimeRecorded = dateTime,
         ptDataNotes = Notes)

#exact time matches: 12 in total
ExactTimeMatches <- WGFPSiteVisitsFieldData1 %>%
  inner_join(EnvironmentalData_prepped, by = c("Site", "dateTime")) %>%
  rename(Notes = Notes.x)

#rest of them:
WGFPSiteVisitsFieldData2 <- WGFPSiteVisitsFieldData1 %>%
  anti_join(ExactTimeMatches, by = colnames(WGFPSiteVisitsFieldData1)) %>%
  dplyr::filter(!is.na(dateTime))
```

Data.table package is attached, will be detached at the end of the function due to lots of function naming conflicts with other packages. Keycols are set and rolling join is performed on the remainder of the data. Data is converted back to dataframe object.

```
library(data.table, quietly = TRUE,warn.conflicts = FALSE)

#make PT data and timestamps into data.table objects so that we can perform rolling join
WGFPSiteVisitsFieldData2 <- data.table(WGFPSiteVisitsFieldData2)
EnvironmentalData_prepped <- data.table(EnvironmentalData_prepped)


#set keycols
#the order these are in matter
keycols <- c("Site","dateTime")
setkeyv(WGFPSiteVisitsFieldData2, keycols)
setkeyv(EnvironmentalData_prepped, keycols)


WGFPSiteVisitsFieldData3 <- EnvironmentalData_prepped[WGFPSiteVisitsFieldData2, roll = "nearest", on = .(Site, dateTime), nomatch = NULL]

WGFPSiteVisitsFieldData3 <- as.data.frame(WGFPSiteVisitsFieldData3) #%>%
```

Data is subset for nearest 13 hours on readings. NOTE: This is different than Detections above where Pressure Transducer Data are given 1 hour buffer and Discharge 13 hours

ExactTimestampMatches and NotExactTimestampMatches are bound together and the rest of the rows (where there is no PT data to associate with the site) are aligned and bound back together to get the same amount of data as the original.

```
#get only rows with timestaps within 13 hours
WGFPSiteVisitsFieldData4 <- WGFPSiteVisitsFieldData3 %>%
  mutate(timeDifference = ifelse(abs(difftime(ptTimeRecorded, dateTime, units = c("hours"))) > 13, 1, 0)
  ) %>%
  dplyr::filter(timeDifference == 0)

ExactTimeMatchesOrganized <- alignColumns(ExactTimeMatches, colnames(WGFPSiteVisitsFieldData4), WGFPSiteVisitsFieldData4)
###joining back up
siteVisitDataWithPTData <- bind_rows(ExactTimeMatchesOrganized, WGFPSiteVisitsFieldData4)
###need to get rest of rows to join for final df
restOfRows <- WGFPSiteVisitsFieldData %>%
  anti_join(siteVisitDataWithPTData, by = c("Site", "Date"))

restOfRowsAligned <- alignColumns(restOfRows, names(siteVisitDataWithPTData), siteVisitDataWithPTData)
allRowsPTDataSiteVisits <- bind_rows(restOfRowsAligned, siteVisitDataWithPTData)
```

Date is reformatted, columns are reorganized to be a bit more readable and situated where we like.

```
#expect NAs here so suprressing these warnings
suppressWarnings({
  allRowsPTDataSiteVisits <- allRowsPTDataSiteVisits %>%
    mutate(dateTime = lubridate::ymd_hms(paste(Date, Time)))
})

mm_columns <- grep("[0-9]+mm", names(WGFP_SiteVisits_FieldData), value = TRUE)

#reorder columns so detectoin distance is first
allRowsPTDataSiteVisits <- allRowsPTDataSiteVisits %>%
  select(Date, Time, Site, Water_Level_NoIce_ft, all_of(mm_columns), dplyr::everything())
```

Spatial_join_function.R

spatial_join_stations_detections()

- Joins detections and events to the shapefile of stations and states in order to later help calculate states and distance moved for each fish.
- Inputs
  - condensed events (arg 1), made from all_combined_events_function (all_events_most_relevant).
  - simple_stations (arg2), which is usually read in as simple_stations2 in the map_polygon_read_ins.r file
  - statesPolygon (arg3) created by Eric R in GIS and is read in as WGFP_States_2024 in map_polygon_readins.R file
- Returns
  - stationsStatesandDetections: a dataframe of condensed_events with information on all station data and State data.
    - Used in createMARKEncounterHistories() and PrepareforMovementsandSummary()
  - noUTMs: df of data with no UTMs assigned, used for QAQC in the CreateFlatFilesRunscript
    - will return a message in the runscript if this dataframe isn't empty
  - noState: dataframe with no state assigned because UTMs were spatially outside the states Polygon. Used for QAQC in the CreateFlatFilesRunscript
    - will also return a message in the runscript if this dataframe isn't empty

About the Code

This function first checks if there are any NA entries in UTM_X or UTM_Y, because those rows will be excluded when joining with the spatial files (simpleStations and statesPolygon). We keep track of these

problem rows and if any exist, you will get a message in the CreateFlatFiles_Runscript.rmd output that stations weren't able to be assigned to those rows due to NA entries.

```
#the utms are grs80 and utm zone 13, which corresponds to espg  32613
#can't do it if there's any NA values in the utm fields
problemRows <- condensedEvents %>%
  filter(is.na(UTM_X))
condensedEventsFiltered <- condensedEvents %>%
  filter(!is.na(UTM_X))
```

The function then uses the package sf to convert the data to a spatial object using the UTMs as coordinates, before transforming it to lat/long projection that is needed for joining states and stations to the data.

```
condensedEventsSF <- sf::st_as_sf(condensedEventsFiltered, coords = c("UTM_X", "UTM_Y"), crs = 32613)
#convert to lat/long
condensedEventsSFLatLong <- sf::st_transform(condensedEventsSF, latLongCRS)
```

The spatial joining for stations is done by the nearest feature, so if a detection is not quite on top of the station, it will get assigned the closest one. Conversely, states are joined by intersection, so if a detection falls outside the spatial StatesPolygon.shp file, it will not be assigned a state. These rows are also tracked and a message will display in the CreateFlatFiles_Runscript.rmd output that states weren't able to be assigned to those rows. TGMs are excepted from this since they're outside our states.

```
#stattions needed to calculate movements and distance moved
stationsAndDetections <- sf::st_join(condensedEventsSFLatLong, simpleStations, st_nearest_feature)
#joins states based off states polygon
stationsStatesandDetections <- sf::st_join(stationsAndDetections, statesPolygon, st_intersects)

#TGM excepted, fish that weren;t assigned a state
noState <- as.data.frame(stationsStatesandDetections) %>%
  filter(!Species %in% c("TGM"),
         is.na(State))
```

PrepareforMovementsandSummary_function.R

PrepareforMovementsandSummary()

- Puts stations onto a condensed All_events dataframe
- Takes stationStateData returned from spatial_join function.
- Returns:
    - DailyMovements_withStationsAndSummaryInfo: a condensed dataframe of all pertinent daily detection info for each tag with stations and states attached.
        - Row entries for tags with multiple detections on the same UTM_X, UTM_Y, at the same antenna, on the same day are filtered out. Leaving the first and last detections of the day, and all detections on unique antennas and UTM's in between.

About the Code

River is assigned for specific antennas **based off the WGFP Metadata.xlsx.**

Station data is corrected for Fraser River, because those stations start at 0. This adds the station where the Colorado/Fraser confluence (**based on fraserColoradoRiverConfluence, a variable defined in WGFP Metadata.xlsx)** to the Fraser stationing. Then duplicate rows are taken out (shouldn't be any) and relevant columns are selected

```
DailyMovements_withStationsFraserColoradoCorrected <- DailyMovements_withStations %>%
  left_join(wgfpMetadata$AntennaMetadata[,c("FrontendSiteCode", "River")], by = c("Event" = "FrontendSiteCode")) %>%
  #selects first non-NA value from set of columns; by having River.Y first it prioritizes that column
  mutate(River = coalesce(River.y, River.x),
         # this part is needed because stations are assigned from 0 up the fraser river starting at the confluence
         #new antennas weren't showing up because I didn't include connectivity channel to to river
         # this assigns a station, then in the get_movements function the distance moved is calculated
         ET_STATION = case_when(River %in% "Fraser River" ~ ET_STATION + fraserColoradoRiverConfluence, #10120 is above Fraser River Confluence; pre-construciton was 9566
                                River %in% c("Colorado River", "Connectivity Channel") ~ ET_STATION,
                                TRUE ~ ET_STATION)
         ) %>%
  select(-River.x, -River.y)
```

This part gets the number of daily unique events and detections for a fish, lets you know if a detection was above and below the dam. This is based on DamLocation, another variable **defined in WGFP Metadata.xlsx.** The type of movement is condensed to a "detection type" field, assigning general site names to antennas based on info in **WGFP Metadata.xlsx.** This is helpful in the movements map where it doesn't necessarily matter which antenna specifically is hit, only if the stationing is different between events.

```
#getting number of daily events
DailyMovements_withStationsNumberofDailyEvents <- DailyMovements_withStationsFraserColoradoCorrected %>%
  group_by(Date, TAG) %>%
  mutate(c_number_of_detections = n(),
         daily_unique_events = length(unique(Event))
  ) %>%
  ungroup()
#generating generic event title for movements map

####getting site name by joining with metadata
DailyMovements_withStationsAndDetectionType <- DailyMovements_withStationsNumberofDailyEvents %>%
  left_join(wgfpMetadata$AntennaMetadata[,c("FrontendSiteCode", "SiteName")], by = c("Event" = "FrontendSiteCode")) %>%
  mutate(det_type = coalesce(SiteName, Event),
         #in this instance, "above the dam" would be mean the CRCC too; any detection in it
         above_below = case_when(
           ET_STATION >= DamLocation ~ "Above the Dam",
           ET_STATION < DamLocation ~ "Below the Dam"
         )
  )
```

The data is then transformed from a spatial object to a regular dataframe for faster processing later, and UTMs are restored to preserve the original data format. Relevant columns are selected

```
# Transform the coordinates back to UTM
sf_object_utm <- st_transform(DailyMovements_withStationsAndDetectionType, crs = 32613)  # Assuming UTM zone 13 with GRS80


coordinates <- st_coordinates(sf_object_utm)

# Convert the coordinates to a data frame
coordinatesDf <- as.data.frame(coordinates)

# Rename the columns
colnames(coordinatesDf) <- c("UTM_X", "UTM_Y")
# # Extract the UTM_X and UTM_Y coordinates
DailyMovements_withStationsAndDetectionType$UTM_X <- round(coordinatesDf$UTM_X, 0)
DailyMovements_withStationsAndDetectionType$UTM_Y <- round(coordinatesDf$UTM_Y, 0)

#need to convert class sf object back to dataframe so that it processes faster in combine_events_stations_function
DailyMovements_withStationsAndDetectionType <- as.data.frame(DailyMovements_withStationsAndDetectionType)
DailyMovements_withStationsAndSummaryInfo <- DailyMovements_withStationsAndDetectionType %>%
  select(Date, Datetime, TAG, Event, det_type, ReleaseSite,Species, Release_Length, Release_Weight, Release_Date, RecaptureSit

return(DailyMovements_withStationsAndSummaryInfo)
```

createMARKEncounterHistories.R

createMARKEncounterHistories()

- Creates and formats data for input into Program MARK
- Inputs

    - DailyDetectionsStationsStates$spatialList$stationStateData: Daily relevant detections with station and state info. Condensed states created from All_Combined_events_function() and states joined in Spatial_join_function()
    - GhostTags, from WGFP_GhostTags.csv
    - AvianPredation, from /data/WGFP_AvianPredation.csv
    - wgfpMetadata$TimePeriods, defined in WGFP Metadata.xlsx
- Outputs
    - MARKEncounterHistories: MARK ready dataframe
    - eventsWithPeriodsSelectForMovementJoining: Daily relevant detections with station and state info (same as stationStateData) with MARK periods. Used for animations
    - States_summarized: summary of states for use in Ind_tag_enc_hist_wide_summary_function
    - possibleAvianPredation: list of dataframes of flagged active fish on a weekly and cumulative basis based on states. Used in QAQC tab

About the Code

Periods are first reformatted from excel numeric to date.

```
#getting start/end dates to correct format
timePeriodsClean <- TimePeriods %>%
  mutate(`start date` = janitor::excel_numeric_to_date(as.numeric(`start date`)),
         `end date` = janitor::excel_numeric_to_date(as.numeric(`end date`))
  )
#need to add the datetime to make sure the <= and >= filtering later is interpreted correctly
timePeriodsCorrect <- timePeriodsClean %>%
  mutate(`end date` = ymd_hms(paste(`end date`, "23:59:59")))
```

Daily detection data with periods is created, based on date of detection and defined start/end dates from periods defined in WGFP Metadata.xlsx.

```
# Gets DF with row of time periods based on when the detection was
# getting periods now so they are used in other DFs like movement animations
DailyDetectionsPeriods <- DailyDetectionsStationsStates1  %>%
  rowwise() %>%
  mutate(
    TimePeriod = timePeriodsCorrect %>%
      filter(Datetime >= `start date` & Datetime <= `end date`) %>%
      pull(periods) %>%
      first()
  ) %>%
  ungroup() #stops rowwise()
```

Some QAQC for checking if any data fall outside the defined periods. If they do, a QAQC message will appear in the CreateFlatFiles_Runcript.rmd output.

```
#for qaqc: this should be a blank df,
#otherwise it means some data don't fall within the time periods sepecified in the csv
dataWithoutPeriods <- DailyDetectionsPeriods %>%
  filter(is.na(TimePeriod))
```

TGMs excluded from the data. Ghost tags and Avian Predation are reformatted to get relevant columns only to join with daily detection data with Periods. Data are joined.

```
#don't need TGM in analysis
detectionsWithStatesPeriods <- as.data.frame(DailyDetectionsPeriods) %>%
  filter(Species != "TGM")
#getting most pertinent info
GhostTagsForJoining <- GhostTags %>%
  rename(TAG = TagID) %>%
  select(TAG, GhostDate)

AvianPredationForJoining <- AvianPredation %>%
  rename(TAG = TagID) %>%
  select(TAG, PredationDate)

#joining with ghost tag and predation dfs
eventsWithGhostDates <- left_join(detectionsWithStatesPeriods, GhostTagsForJoining, by = c("TAG"))
eventsWithGhostDatesAndAvianPredation <- left_join(eventsWithGhostDates, AvianPredationForJoining, by = c("TAG"))
```

Assigning ghost/predation to state "G"

```
#combining ghost and predation just to 1 state "G"
#if it's predated and ghost, predated will always come first
eventsWithGhostDatesAndAvianPredation <- eventsWithGhostDatesAndAvianPredation %>%
  mutate(GhostOrPredationDate = coalesce(PredationDate, GhostDate),
         State = case_when(Date >= GhostOrPredationDate ~ "G",
                           TRUE ~ State)
  ) %>%
  #getting rid of data for tags after their ghost/predation date
  #makes sure to keep non-predated tags if they don'thave a ghost/predation date
  filter(is.na(GhostOrPredationDate) |
           Date <= GhostOrPredationDate)
```

Take just the first instance of ghost/predation data since everything after that is ghost anyway

```
# we don't need duplicated rows of states and tags that fall on the same day.
#getting rid of same-day ghost data, keeping just the first detection that day that qualifies as ghost/predation
eventsWithOneGhostEvent <- eventsWithGhostDatesAndAvianPredation %>%
  arrange(TAG, Datetime) %>%
  group_by(TAG, State) %>%
  mutate(firstDatetime = dplyr::if_else(!is.na(GhostOrPredationDate) & State == "G", first(Datetime), NA)) %>% |
  relocate(TAG, Datetime, GhostOrPredationDate, firstDatetime) %>%
  filter(is.na(GhostOrPredationDate) | is.na(firstDatetime) |
           Datetime == firstDatetime)
```

Organizing releases and recaptures and creating the QAQC column "NeedsLogWeight" if the entry doesn't have a weight

```
#release and recaps only
recapsAndRelease <- detectionsWithStatesPeriods %>%
  filter(Event %in% c("Recapture", "Recapture and Release", "Release", "Recapture ")) %>%
  mutate(NeedsLogWeight = ifelse((Release_Length > 0 & Release_Weight == 0) | (Event %in% c("Recapture") & is.na(Recap_Weight)), TRUE, FALSE))

#select pertinent columns
eventsWithPeriodsSelect <- eventsWithOneGhostEvent %>%
  select(TAG, Datetime, Event, TimePeriod, State)
```

For fish that were recaptured in the same time period that they were released, the second event (the recapture) is used for lengths and weights. A column is created "releaseAndRecappedInSamePeriod" to help keep track of these fish.

```
####There are some fish that were recaptured in the same time period they were released.
#for these fish, we just want their second event (the recapture) as their "Release" event.
#this code finds fish where this is the case and removes rows previous to the "Recapture" event

eventsReleaseRecapSameTimePeriodCorrection <- eventsWithPeriodsSelect %>%
  group_by(TAG, TimePeriod) %>%
  arrange(Datetime) %>%
  mutate(releasedAndRecappedInSamePeriod = any(Event %in% c("Release", "Recapture and Release")) & any(Event %in% c("Recapture"))) %>%
  filter(!(releasedAndRecappedInSamePeriod & row_number() < max(which(Event == "Recapture"), default = 0))) %>%
  #this changes recap Event to Release when there is a release/recap in the same time period
  mutate(Event = ifelse(Event == "Recapture" & releasedAndRecappedInSamePeriod, "Release", Event))
```

For fish with multiple recaptures in the same period, the most recent row is used for lengths/weights. The older row(s) is removed. multipleRecapsInPeriod is created as a QAQC column to keep track of these fish.

```
# Now getting times where a fish was recapped twice in the same period
multipleRecapInstancesSamePeriod <- eventsReleaseRecapSameTimePeriodCorrection %>%
  filter(Event == "Recapture") %>%
  group_by(TAG, TimePeriod) %>%
  count(name = "recapsInSinglePeriod") %>%
  filter(recapsInSinglePeriod > 1) %>%
  mutate(multipleRecapsInPeriod = TRUE) %>%
  select(-recapsInSinglePeriod)

#joins to get instances of multiple "Recaptures" in the same period
eventsWithRecapReleaseSameTimePeriodCorrection <- eventsReleaseRecapSameTimePeriodCorrection %>%
  left_join(multipleRecapInstancesSamePeriod, by = c("TAG", "TimePeriod")) %>%
  group_by(TAG, TimePeriod) %>%
  arrange(Datetime) %>%
  #removes rows where there are 2 instances of recaps in the same period
  #it's already arranged by datetime, so it finds where the max (most recent) recap is within that grouping of TAG and timePeriod,
  # and removes instances of Recap that have row numbers beofre the max recap instance within that time period
  #using default = 0 makes sure there's a value when there's no recap events. Speeds up the code, ensures there's no warnings.
  filter(!(Event == "Recapture" & row_number() < max(which(Event == "Recapture"), default = 0))) %>%
  ungroup()
```

Column “group” is created to help identify how many times a fish has been recaptured and ensure the tag has the appropriate number of lines in our data.

```
### makes Identifier (group) for TAGs based on number of times they've been recapped
groupedRecapEventsByTag <- eventsWithRecapReleaseSameTimePeriodCorrection %>%
  group_by(TAG) %>%
  mutate(
    isRecapture = ifelse(str_trim(Event) == "Recapture", 1, 0),
    group = cumsum(lag(isRecapture, default = 0)) + 1
  )
```

Adding recaptures with a new “group” added +1 so that when a fish is recaptured it will start another line of data

```
#adds another row for recapture with a new group so that later the new recap will start the new line of data
additionalRecapInstance <- groupedRecapEventsByTag %>%
  bind_rows(groupedRecapEventsByTag %>%
              filter(str_trim(Event) == "Recapture") %>%
              mutate(group = group + 1)
  ) %>%
  arrange(TAG, Datetime)
```

We decided that for Ghost states that occur in the same time Period as their release period, we want to put a Ghost state in the subsequent period, even if there wasn’t necessarily a detection during that period. This code accounts for that.

```
# ghost state cannot occur the same state as a fish is released. So this code adds 1 period where the ghost state occurs as the same time period as the release.

releaseGhostCorrection <- additionalRecapInstance %>%
  mutate(TimePeriod = as.numeric(TimePeriod)) %>%
  group_by(TAG, TimePeriod) %>%
  arrange(Datetime) %>%
  mutate(releaseAndGhostSamePeriod = State == "G" & any(Event %in% c("Release", "Recapture and Release")),
    TimePeriod = ifelse(releaseAndGhostSamePeriod, TimePeriod + 1, TimePeriod))
```

Takes the last state in each period because that’s the state we want to use for each period in the final df.

```
#grabs the last state the tag appeared in for that time period
lastStateInTimePeriod <- releaseGhostCorrection %>%
  group_by(TAG, TimePeriod, group) %>%
  summarize(condensedStates = gsub('([[:alpha:]])\\1+', '\\1', paste(State, collapse = ""))) %>%
  mutate(newState = str_sub(condensedStates,-1,-1)) %>%
  select(-condensedStates)
```

Finally, pivoting the data to wide format and giving 1 or -1 values to the columns based on whether it's the last line of a Tag's history or not.

```
#puts the data to wide format and assigns the number based on if the Tag's history ended or not (group identifier)
tagsEventsWideFormatWithCount <- lastStateInTimePeriod %>%
  group_by(TAG, TimePeriod, group) %>%
  arrange(as.numeric(TimePeriod)) %>%
  pivot_wider(names_from = TimePeriod, values_from = newState) %>%
  group_by(TAG) %>%
  mutate(Count = ifelse(group == max(group), 1, -1))
```

Ghost states are automatically assigned a -1

```
#modifies the data checking for state "G", if the fish has a G in the history its automatically a -1
tagsEventsWideWithGhost <- tagsEventsWideFormatWithCount %>%
  rowwise() %>%
  mutate(Count = case_when(any(c_across(matches("^[0-9]+$")) == "G") ~ -1,
                           TRUE ~ Count)
  ) %>%
  ungroup() #stops rowwise()
```

QAQC columns are summarized for each tag to help keep track of the tags that have needed some special attention. This will be joined to the final data later.

```
######
QAQCColumnsDF <- releaseGhostCorrection %>%
  group_by(TAG) %>%
  summarize(releaseAndGhostSamePeriod = any(releaseAndGhostSamePeriod),
            releasedAndRecappedInSamePeriod = any(releasedAndRecappedInSamePeriod),
            multipleRecapsInPeriod = any(multipleRecapsInPeriod)) %>%

   mutate(multipleRecapsInPeriod = replace_na(multipleRecapsInPeriod, FALSE) )
```

Attribute info is then added. We need to join by both Tag number and group in order to get the correct data, but since we modifed the group field earlier in the code for recaps, we need to make some slight corrections here before joining.

```
#getting recap and release info ready to join with MARK data;
# using the same identifier (group) to help join
#can't just create new group() because group number has already been corrected for in groupedRecapEventsByTag (release/recaps in same time period)
#but there hasn't been additoinal recap instance added yet in groupedRecapEventsByTag so we should get that we need

recapsAndReleaseWithGroup <- recapsAndRelease %>%

  mutate(
         RBT = ifelse(Species == "RBT", 1, 0),
         LOC = ifelse(Species == "LOC", 1, 0),
         MTS = ifelse(Species == "MTS", 1, 0)) %>%
  #first, filter out which recaps/release we don't want based off group assinings from earlier
  left_join(groupedRecapEventsByTag[,c("TAG", "Datetime", "group")], by = c("TAG", "Datetime")) %>%
  filter(!is.na(group)) %>%
  #then make new group numbers
  group_by(TAG) %>%
  arrange(Datetime) %>%
  mutate(group = row_number())
```

Now we're able to join length weight info. Recap lengths and weight take preference if there are entries for both. NAs are also replaced with 0s because that's how we want the non-state entries formatted.

```r
#joining by TAG and group, merging all LEgnth/weight columns and deselecting redundant ones
##can't join by event because 2nd recap events have already been changed to "Release" now
tagsEventsWideLW <- tagsEventsWideWithGhost %>%
  #adding length and weight columns
  #coalescing recap dat first so that will take priority
  left_join(recapsAndReleaseWithGroup[,c("TAG", "group", "Release_Length", "Release_Weight", "Recap_Length", "Recap_Weight", "RBT", "LOC", "MTS", "NeedsLogWeight")] %>%
              mutate(Length = coalesce(Recap_Length, Release_Length),
                     Weight = ifelse(NeedsLogWeight, NA, coalesce(Recap_Weight, Release_Weight))),
            by = c("TAG", "group")) %>%
  select(-c(Release_Length, Release_Weight, Recap_Length, Recap_Weight))

## replace NA with 0
tagsEventsWide0s <- tagsEventsWideLW %>%
  rowwise() %>%
  mutate(across(matches("^[0-9]+$"), ~ replace_na(.x, "0"))) %>%
  ungroup() #stops rowwise()
```

In order to make the column names correspond to the correct time periods, we pivot the data back to long format, add those dates and put them back to wide format.

```r
### getting the column names to the time periods described
# Reshape the wide df to long format
tagsEventsLong <- tagsEventsWide0s %>%
  pivot_longer(matches("^[0-9]+$"), names_to = "periods", values_to = "State") %>%
  mutate(periods = as.character(periods))  # Convert Number to integer for matching

# Join with the time periods dataframe to get the date range for each Number
tagsEventsLongwithTpDates <- tagsEventsLong %>%
  left_join(timePeriodsClean[,c("periods", "start date", "end date")], by = "periods") %>%
  mutate(NewColumnName = paste(`start date`, "to", `end date`))

#getting columns in desired order
colsToMov <- c("Count", "Length", "Weight", "RBT", "LOC", "MTS", "NeedsLogWeight")
# Reshape back to wide format with new column names
# this df will not be the same number of rows as recaps and releases because some fish were released/recapped in the same time periods,
# so for those fish, the most recent instance is taken as that recap/release
tagsEventsWideCorrectTpLabels <- tagsEventsLongwithTpDates %>%
  select(-`start date`, -`end date`, -periods) %>%
  pivot_wider(names_from = NewColumnName, values_from = "State") %>%
  select(-all_of(colsToMov), all_of(colsToMov))

#adding on QAQC columns
```

Our desired QAQC columns are added from earlier, and this is our final dataframe.

```r
#adding on QAQC columns
# this is the final DF
tagsEventsWideCorrectTpLabelswithQAQC <- tagsEventsWideCorrectTpLabels %>%
  left_join(QAQCColumnsDF, by = "TAG")
```

Other DFs are created based off these new states to help flag avian predation. These will go in the QAQC tab.

```r
##### Potential Avian Predation
DailyMovements_withStationsWeeksSince <- as.data.frame(DailyDetectionssStationsStates1) %>%
  ungroup() %>%
  mutate(
    #makes sense to use floor not ceiling with weeks because then there are are more fish in week 0
    # if you want to start at week 1 instead of week 0, add +1 to the end of expression
    # when you change this too, it changes the number of entries in the states dataframe
    weeks_since = as.numeric(floor(difftime(Datetime, min(Datetime), units = "weeks")))
  )
weeklyStates <- DailyMovements_withStationsWeeksSince %>%
  group_by(weeks_since, TAG) %>%
  arrange(Datetime) %>%
  mutate(
    allWeeklyStates = paste(State, collapse = ""),
    condensedWeeklyStates = gsub('([[:alpha:]])\\1+', '\\1', allWeeklyStates), #removes consecutive letters
    weekly_unique_events = length(unique(Event))
  )
```

Fish with more than 4 weekly events or more than 6 States in a week are flagged. This constitutes our dataframe in the Avian Predation tab as “Overactive on a weekly basis”

```
#this is now a weekly chart
cleanedWeeklyStates <- weeklyStates %>%
  distinct(weeks_since, TAG, condensedWeeklyStates, .keep_all = TRUE) %>%
  select(-State) %>%
  rename(State = condensedWeeklyStates)

weeklyActiveFish <- cleanedWeeklyStates %>%
  filter(weekly_unique_events > 4 | str_length(State) > 6) %>%
  select(TAG, Date, weeks_since, State, allWeeklyStates, weekly_unique_events)
```

We also create a summarized state dataframe for joining later in Ind_tag_enc_hist_wide_summary_function(), whose data is displayed in the Encounter Release History Summary Wide tab.

```
#used for avian predation and also in encounter histories summary wide
#we want to keep TGM here so using original DF
summarizedStates <- as.data.frame(DailyDetectionsStationsStates1) %>%
  group_by(TAG) %>%
  arrange(Datetime) %>%
  mutate(allStates = paste(State, collapse = ""),
         condensedAllStates = gsub('([[:alpha:]])\\1+', '\\1', allStates), #removes consecutive letters
         channelSummary = case_when(str_detect(condensedAllStates, "D") ~ "Used Connectivity Channel",
                                    TRUE ~ "Didn't Use Channel"),

         #new columns to say if fish stayed above or below?
         went_above_dam_noChannel = str_detect(condensedAllStates, "CE|BE|AE|CF|BF|AF|CH|BH|AH"),
         went_below_dam_noChannel = str_detect(condensedAllStates, "EC|EB|EA|FC|FB|FA|HC|HB|HA"),
         went_below_dam_throughChannel = str_detect(condensedAllStates, "EDC|EDB|EDA|FDC|FDB|FDA|HDC|HDB|HDA"),
         went_above_dam_throughChannel = str_detect(condensedAllStates, "CDE|BDE|ADE|CDF|BDF|ADF|CDH|BDH|DAH"),
         entered_channel_from_DS = str_detect(condensedAllStates, "AD|BD|CD"),
         entered_channel_from_US = str_detect(condensedAllStates, "ED|FD|HD"),

  ) %>%
  select(TAG, condensedAllStates, channelSummary, went_above_dam_noChannel, went_below_dam_noChannel,went_below_dam_throughChannel,
         went_above_dam_throughChannel,entered_channel_from_DS,entered_channel_from_US) %>%
  distinct(TAG, .keep_all = TRUE)
```

Another dataframe for avian predation is created based the summarized states. If a fish had more than 6 states throughout its life and was NOT also listed in the "weeklyActiveFish" dataframe, it gets flagged.

```
overAllActiveFish <- summarizedStates %>%
  filter(str_length(condensedAllStates) >6)

overAllActiveFishNotinWeeklyDF <- anti_join(overAllActiveFish, weeklyActiveFish, by = "TAG")
possibleAvianPredation <- list(
  "weeklyActiveFish" = weeklyActiveFish,
  "overAllActiveFishNotinWeeklyDF" = overAllActiveFishNotinWeeklyDF
)
```

Ind_tag_enc_hist_wide_summary_function.R

Ind_tag_enc_hist_wide_summary_function()

- Makes a summary dataframe of all released/tagged fish with over 60 columns of summary info, like "total number of antenna encounters", "total distance travelled", "moved through dam or not"
- Inputs
  - allDetectionsAndRecaptures from All_Combined_events_function(). A file of all antenna detections and recaptures, but no release data.
  - Release data from the .csv
  - combined_events_stations from PrepareforMovementsandSummary_function(). A condensed dataframe of all pertinent daily detection info for each tag with stations and states attached.
  - States_summarized from the createMARKEncounterHistories function. Summary of states across all a Tag's history.
  - markerTag:, dataframe of all marker tags used

    - It doesn't take All Events from All_combined_Events_function because we want to do some operations on the events that aren't Releases, before bringing the release data back in for joining.
- Returns:
    - ENC_Release_wide_summary : Summary "wide" dataframe of each tagged fish and summary info. Used in Encounter Release History Summary Wide Tab
    - Unknown_Tags: dataframe of non-marker tags that have encounters, but have no release info. Used in QAQC tab.
    - possibleAvianPredation: flagged fish moving over 1000m in their lifetime, used in "QAQC"-> "Avian Predation" tab

About the Code

This part counts the number of Events for each fish. It then pivots the data to wide format, so each fish/tag has a row. Columns are then renamed. It's important that detections from the readers are the same as the BackendSiteCodes described in WGFP Metadata.xlsx. These codes are converted to FrontEndSiteCodes described WGFP Metadata.xlsx for what we want to display. This is the start of the summary file.

```
allEncountersWide <- allDetectionsAndRecaptures %>%
  count(TAG, Event, name = "Encounters") %>%
  pivot_wider(id_cols = TAG, names_from = Event, values_from = Encounters) %>%
  rename_with(~ paste0(., "_n"), -TAG)

#column order is just nice to have for the user
columnOrder <- c(RedBarnFrontendCodes, HitchingPostFrontendCodes, ConfluenceFrontendCodes, ConnectivityChannelDownstreamFrontendCodes, ConnectivityChannelSideChannelFrontendCodes,
                 ConnectivityChannelUpstreamFrontendCodes, MobileRunFrontendCodes, WindyGapBypassAntennaFrontendSiteCode, WindyGapAuxiliaryAntennaFrontendSiteCode,
                 GranbyDiversionAntennaFrontendSiteCode, RiverRunAntennaFrontendSiteCode, FraserRiverCanyonAntennaFrontendSiteCode
                 )
allEncountersWideOrdered <- allEncountersWide %>%
  select(TAG, one_of(paste0(columnOrder, "_n")), Recapture_n)
```

The release data is then joined to the summary file using dplyr::full_join() to get all columns from each dataframe. If there wasn't a count for a fish, that NA gets changed to 0. This is also when the unknown tags df is made (displayed later in qaqc). These tags all start with 900. They are probably cormorants or mergansers that have swallowed other PIT tagged trout and chubs from glenwood ;)

```
#### Combine Release data ###
releasePreparedForJoin <- Release %>%
  mutate(TAG = str_trim(TagID)) %>%
  replace_na(list(Species = "No Info", ReleaseSite = "No Info")) #replaced species and releasesite to follow the same convention as AllEvents

# was geting a massive dataframe because the Release df is called TAGid not TAG.
#need to actually join on full join not merge
encountersAndRelease <- full_join(releasePreparedForJoin, allEncountersWideOrdered,  by = "TAG")

#gets rest of the number count columns to 0 from NA
encountersAndRelease[is.na(encountersAndRelease)] = 0

#gets tag list that wasn't in release file or markerTags
unknown_tags <- encountersAndRelease %>%
  filter(is.na(ReleaseSite),
         !TAG %in% markerTags) %>%
  select(TAG,where(is.numeric))
```

True/false columns are made on whether a fish hit a specific antenna or was recaptured.

```
#### Make 1 or 0 for encounter history rather than counts ###
#gets df with TF of whether a fish was detected at a antenna
encountersAndReleasePreparedForRowSums <- encountersAndRelease %>%
  #applies the mutation logic to all columns that end with "_n".
  #It checks if each value is greater than 0
  #creates a new column with the name obtained by removing "_n" from the original column name.
  mutate(across(ends_with("_n"), ~ (. > 0), .names = "{sub('_n', '', .col)}"))
```

Here some summary statistics based on the previous columns are calculated. It tells the number of detections a fish may have at a paired antenna. Also a T/F column to see if a fish was detected at a paired antenna. The fish with unknown release data are also filtered out here. Again, this is where if the BackendSiteCode in WGFP Metadata.xslx didn't match up with what was coming off the readers, there will be issues.

```
encountersAndReleaseRowsums <- encountersAndReleasePreparedForRowSums %>%
  #counts number of TRUE across rows specified by antenna codes. -SG
  mutate(
    TotalEncounters = rowSums(select(., all_of(c(RedBarnFrontendCodes, HitchingPostFrontendCodes, ConfluenceFrontendCodes,
                                                 ConnectivityChannelDownstreamFrontendCodes, ConnectivityChannelSideChannelFrontendCodes,
                                                 ConnectivityChannelUpstreamFrontendCodes, MobileRunFrontendCodes,
                                                 WindyGapBypassAntennaFrontendSiteCode, WindyGapAuxiliaryAntennaFrontendSiteCode, GranbyDiversionAntennaFrontendSiteCode,
                                                 RiverRunAntennaFrontendSiteCode, FraserRiverCanyonAntennaFrontendSiteCode, "Recapture"))) == TRUE),
    TotalAntennas = rowSums(select(., all_of(c(RedBarnFrontendCodes, HitchingPostFrontendCodes, ConfluenceFrontendCodes,
                                               ConnectivityChannelDownstreamFrontendCodes, ConnectivityChannelSideChannelFrontendCodes,
                                               ConnectivityChannelUpstreamFrontendCodes, MobileRunFrontendCodes,
                                               WindyGapBypassAntennaFrontendSiteCode, WindyGapAuxiliaryAntennaFrontendSiteCode,
                                               GranbyDiversionAntennaFrontendSiteCode,
                                               RiverRunAntennaFrontendSiteCode, FraserRiverCanyonAntennaFrontendSiteCode
                                               ))) == TRUE),
    TotalStationary = rowSums(select(., all_of(c(RedBarnFrontendCodes, HitchingPostFrontendCodes, ConfluenceFrontendCodes,
                                                 ConnectivityChannelDownstreamFrontendCodes, ConnectivityChannelSideChannelFrontendCodes,
                                                 ConnectivityChannelUpstreamFrontendCodes))) == TRUE),

    TotalMobile = rowSums(select(.,  all_of(MobileRunFrontendCodes)) == TRUE),
    TotalBiomark = rowSums(select(., all_of(c(WindyGapBypassAntennaFrontendSiteCode, WindyGapAuxiliaryAntennaFrontendSiteCode,
                                              GranbyDiversionAntennaFrontendSiteCode,
                                              RiverRunAntennaFrontendSiteCode, FraserRiverCanyonAntennaFrontendSiteCode
                                              ))) == TRUE),
    TotalRedBarn =rowSums(select(.,  all_of(RedBarnFrontendCodes)) == TRUE),
    TotalHitchingPost = rowSums(select(.,  all_of(HitchingPostFrontendCodes)) == TRUE),
    TotalConfluence = rowSums(select(.,  all_of(ConfluenceFrontendCodes)) == TRUE),
    TotalConnectivityDownstream = rowSums(select(., all_of(ConnectivityChannelDownstreamFrontendCodes)) == TRUE),
    TotalConnectivitySideChannel = rowSums(select(., all_of(ConnectivityChannelSideChannelFrontendCodes)) == TRUE),
    TotalConnectivityUpstream = rowSums(select(., all_of(ConnectivityChannelUpstreamFrontendCodes)) == TRUE)
  ) %>%
  # just says if the fish was ever detected at these sites
  mutate(RedBarn = TotalRedBarn > 0,
         HitchingPost = TotalHitchingPost > 0,
         Confluence = TotalConfluence > 0,
         ConnectivityDownstream = TotalConnectivityDownstream > 0,
         ConnectivitySideChannel = TotalConnectivitySideChannel > 0,
         ConnectivityUpstream = TotalConnectivityUpstream > 0,
         Biomark = TotalBiomark > 0,
         Mobile = TotalMobile > 0
  ) %>%
  filter(TAG %in% unique(releasePreparedForJoin$TAG))
```

This part brings the condensed df (combined_events_stations) with stations back in for summaries. Summarizes if a fish has been above/below the dam. Then counts the number of encounters each fish had above and below the dam. Then describes that event (eg "Mobile Detection above the dam"). Then it pivots the data wider to get each fish its own row and get in a format for joining back to the original release summary file. Selects desired columns. And also changes the NA's to FALSE, because when you use count(), you only sum fish that have had encounters.

```
###Bringing in Station data with info about ABOVE/BELOW dam for joining
### was running into issue where release data isn't being brought in well; The tags aren't being brought in as full numbers, so when the release data is
# it can't match up 23000088888 to 2.3E+11; so release site gets put in as "no info", and
#therefore when the columns join, it doesn't make a column called "release above dam"
#SOLVED: needed to change the format in release csv file

# trying to go based on movements
### thinking of disbanding this and doing the same process but with the states in order to say if fish went above/below

throughDamInfo <- combined_events_stations %>%
  group_by(TAG) %>%
  summarize(through_dam = if_else(all(above_below == "Above the Dam"), "Stayed Above the Dam",
                                  if_else(all(above_below == "Below the Dam"), "Stayed Below the Dam",
                                          "Went through dam or Used Connectivity Channel")))

aboveAndBelowInfo <- combined_events_stations %>%
  count(TAG, det_type, above_below, name = "Encounters") %>%
  mutate(combined_event = paste(det_type, above_below),
         EncountersTF = ifelse(Encounters > 0,
                               TRUE,
                               FALSE))

aboveAndBelowInfo1 <- pivot_wider(data = aboveAndBelowInfo, id_cols = TAG, names_from = combined_event, values_from = EncountersTF)
aboveAndBelowInfoRecapsAndMobile <- aboveAndBelowInfo1 %>%
  select(TAG, `Release Above the Dam`, `Release Below the Dam`,`Recapture Above the Dam`,`Recapture Below the Dam`,`Recapture and Release Above the Dam`,
         `Recapture and Release Below the Dam`, `Mobile Run Above the Dam`, `Mobile Run Below the Dam`)
#turns all the NA's made to FALSE
aboveAndBelowInfoRecapsAndMobile[is.na(aboveAndBelowInfoRecapsAndMobile)] = FALSE
```

A series of dataframes are joined with the original summary file to get all desired summary columns. Sum_distance moved by each fish is calculated from combined_events_stations (see the All Events Combined description to see how this is formed) and used as a column. Here, some data might be lost since it is using the condensed "most relevant" daily rather than every single detection, but it's pretty rare that that will matter anyway, because most fish that this warning/edge case applies to are likely predated.

```
encountersAndRelease3 <- encountersAndReleaseRowsums %>%
  left_join(aboveAndBelowInfoRecapsAndMobile, by = "TAG")
### need to figure out how connectivity channel fits into this part?
#currently counts the connectivity channel as going "through the dam"
encountersAndRelease4 <- encountersAndRelease3 %>%
  left_join(throughDamInfo, by = "TAG")
# left joining states summary to encountersAndRelease
encountersAndRelease5 <- left_join(encountersAndRelease4, States_summarized, by = "TAG")
#rearranging so that Tag is first column shown
encountersAndRelease5 <- encountersAndRelease5 %>%
  select(TAG, 1:ncol(encountersAndRelease5))
###joining on column with sum data
#same code appears in movements function
sum_dist1 <- combined_events_stations %>%
  group_by(TAG) %>%
  arrange(Datetime) %>%
  mutate(dist_moved = ET_STATION - lag(ET_STATION, order_by = Datetime),
         sum_dist = (sum(abs(diff(ET_STATION, na.rm = TRUE))))
  ) %>% #end of mutate
  distinct(TAG, .keep_all = TRUE) %>%
  select(TAG, sum_dist)

encountersAndRelease6 <- encountersAndRelease5 %>%
  left_join(sum_dist1, by = "TAG") %>%
  mutate(Date = ifelse(str_detect(Date, "/"),
                       as.character(mdy(Date)),
                       Date))
```

Dummy rows are also taken out and avian predation dataframe is created for fish travelling >1000m in their lifetime. Used for QAQC.

```
#### dummy rows removal:
encountersAndRelease6 <- encountersAndRelease6 %>%
  filter(!TAG %in% c("230000999999"))


####avian predation filtering
possibleAvianPredation <- encountersAndRelease6 %>%
  filter(sum_dist > 1000) %>%
  select(TAG, went_above_dam_noChannel, went_below_dam_noChannel, sum_dist) %>%
  arrange(desc(sum_dist))
```

Get_movements_function.R

get_movements_function()

- Makes a condensed daily dataframe of fish "movements"; see "movements defined" for more info
    - A movement is defined as a change in stationing (assigned to detections in the spatial_join function). A fish that continuously hits RB1 and RB2 for example would register as a "No Movement", but a fish that goes from RB2 to HP3 would register as an "Upstream Movement" on HP3. All data is incorporated in this calculation, so if a fish was released below red barn, hit the red barn stationary antenna, then was detected by a mobile run upstream of red barn, then was recaptured downstream of that mobile detection, it will register as a "Initial release", "Upstream Movement", "Upstream Movement", "Downstream Movement". The absolute total of this distance is also summed and displayed as a column in the Encounter Histories Summaries Wide tab.
- Inputs
    - combined_events_stations from PrepareforMovementsandSummary_function
        - A condensed dataframe of all pertinent daily detection info for each tag with stations and states attached.
    - dailyUSGSData: USGS Discharge and temperature data read in from the dataRetreival Package each time the runscript is compiled
    - eventsWithPeriodsSelect: condensed data with Time Period from createMARKEncounterHistories
    - TimePeriods, from WGFP Metadata.xlsx
- Returns:
    - dailyMovementsTable1: dataframe where only daily movements are included per tagged fish on unique antennas and UTM's. See "Movements defined" for about them
    - avianPredationDFS: flagged movement dataframes for use in "QAQC" -> "Avian Predation" tab
        - longMovements: a tag travelled a suspiciously long distance on a single movement
        - fastMovements: a tag travelled suspiciously fast on a single movement

<u>About the Code</u>

The code first groups by each tag and arranges each tag in order by their first, then subsequent events. It then calculates the distance moved between each event for each fish, based on the stationing. The code accounts for Fraser/Colorado River transitions by first calculating the distance to the confluence from the last event, then finding the distance from the confluence to the present event, then adding those together. Otherwise, it's just the difference between the last event and the current one.

The sum distance is then calculated as an absolute value of each movement that was calculated for a fish.

Marker Colors and icon colors are then assigned for later mapping.

```r
dailyMovementsTableAll <- combined_events_stations %>%
  #### removing dummy tag
  filter(!TAG %in% c("230000999999")) %>%
  select(Date, Datetime, TAG, det_type, Event, ET_STATION, Species, Release_Length, Release_Weight, ReleaseSite, Release_Date, RecaptureSite, River, UTM_X, UTM_Y, first_las
  #grouping by TAG and arranging by datetime makes sure that total distance moved is totalled and summed in order
  group_by(TAG) %>%
  arrange(Datetime) %>%
  #dist_moved would be the place to fenagle new movements based on previous event...ie hitting wg biomark followed by connectivity channel = add 300 m
  #trickier but doable for mobile runs
  ### accounting for FRASER/UPPER COLROADO MOVEMENTS
  #if previous station is above the confluence and current station is above the confluence and you changed rivers,
  #then take the previous station and subtract the confluence station to get distance travelled to the confluence (A), then subtract the new station minus confluence static
  # otherwise, just subtract current station from previous
  mutate(dist_moved = case_when(lag(ET_STATION, order_by = Datetime) > fraserColoradoRiverConfluence & ET_STATION > fraserColoradoRiverConfluence & River != lag(River, orde
                                TRUE ~ ET_STATION - lag(ET_STATION, order_by = Datetime)
  ),
  sum_dist = (sum(abs(dist_moved), na.rm = TRUE)),
  #learned that you need to specify units = "" not just provide the arguments
  MPerSecondBetweenDetections = ifelse(dist_moved == 0, 0,
                               dist_moved/(as.numeric(difftime(Datetime, lag(Datetime), units = "secs")))
  ),

  movement_only = case_when(lag(ET_STATION, order_by = Datetime) > fraserColoradoRiverConfluence & ET_STATION > fraserColoradoRiverConfluence & River != lag(River, order_by
                            Event %in% c("Release", "Recapture and Release")  ~ "Initial Release",
                            dist_moved == 0 ~ "No Movement",
                            dist_moved > 0 ~ "Upstream Movement",
                            dist_moved < 0 ~ "Downstream Movement"),
  #this is for mapping later on
  #MARKERCOLOR options: limited because markers rely on static image
  #red", "darkred", "lightred", "orange", "beige", "green", "darkgreen", "lightgreen", "blue", "darkblue", "lightblue", "purple", "darkpurple", "pink", "cadetblue", "white"
  # these also correspond to the movements maps, so if you add or change a color you should change it on the movements map as well.
  marker_color = case_when(movement_only == "No Movement" ~ "black",
                           movement_only == "Upstream Movement" ~ "green",
                           movement_only == "Downstream Movement" ~ "red",
                           movement_only == "Initial Release" ~ "orange",
                           movement_only == "Changed Rivers" ~ "purple",
                           #str_detect(movement_only, "Initial Release (or recapture and release)") ~ "yellow"
  ),

  icon_color = case_when(str_detect(det_type, "Stationary Antenna") ~ "orange",
                         str_detect(det_type, "Biomark Antenna") ~ "yellow",
                         str_detect(det_type, "Mobile Run") ~ "purple",
                         det_type %in% c("Release", "Recapture and Release") ~ "blue",
                         det_type == "Recapture" ~ "brown",
  )
  )
```

Data is parred down with the distinct() function, using “movement” as a main field to instead of “first/last”, as was done to get the original dataframe “combined_events_stations”. This removes a lot of unnecessary data and gets to the most relevant movements.

When we do this, most of the data that we remove are “No Movements” from fish staying on one antenna during the course of the day. However, there are some edge cases where filtering in this way omits a few relevant movements. For example, fish 230000144498 started one sequence at CF. Then on November 4 2024, it went to CU, then CF, then CU all in one day. This results in there being 2 “Downstream” Movements for that tag from CF to CU on that same day, and using distinct() on Date, Tag, det_type, movement_only, UTM_X and UTM_Y results in the second Downstream Movement being omitted. However, we can find those rows if we use distinct() on Date, Tag, det_type, first_last, movement_only, UTM_X and UTM_Y then filter the movements we want from the larger data. Those data are then joined to the most relevant movements data so that we have all the desired movements.

For reference: as of May 2, 2025 there were only 23 cases like above that were added out of 34571 relevant movements. So using distinct() on Date, Tag, det_type, movement_only, UTM_X and UTM_Y was still capturing 99.93% of desired relevant data.

```r
dailyMovementsTableMost <- dailyMovementsTableAll %>%
  #takes “movement” into account instead of “first_last”, this is why this df winds up with less rows than combined_events_stations
  distinct(Date, TAG, det_type, movement_only, UTM_X, UTM_Y, .keep_all = TRUE)
###upstream and downstream movements that are not captured in the above code:
#typically predated fish that have multipl US/DS movements in the same day to an antenna, but we have them just to be safe.
#For example: 230000144498 went to CU to CF to CU on the same day and this code captures that last movement to end at CU
dailyMovementsEdgeCases <- dailyMovementsTableAll %>%
  distinct(Date, TAG, det_type, first_last, movement_only, UTM_X, UTM_Y, .keep_all = TRUE) %>%
  anti_join(dailyMovementsTableMost) %>%
  filter(movement_only %in% c("Upstream Movement", "Downstream Movement", "Changed Rivers"))

dailyMovementsTable <- bind_rows(dailyMovementsTableMost, dailyMovementsEdgeCases)
```

Environmental data from USGS is added

```r
##add on environmental Data
dailyMovementsTable <- dailyMovementsTable %>%
  left_join(dailyUSGSData[,c("Date", "WtempF", "Flow")], by = "Date") %>%
  rename(USGSDischargeDaily = Flow)
```

Avian Predation data is flagged. longMovements are subjectively anything over 3700m at a time (HP to CF is 3760m). fastMovements are the top 5% of fastest movements, upstream or downstream.

```r
#avian Predation
#red batr to confluence: 5950/60
#rb to HP: 2190
#hp to CF: 3760
longMovements <- dailyMovementsTable %>%
  filter(abs(dist_moved) > 3700) %>%
  arrange(desc(abs(dist_moved))) %>%
  relocate(dist_moved, .after = TAG)
#gets top 5% of speedy movements, US or DS
fastMovements <- head(dailyMovementsTable[order(abs(dailyMovementsTable$MPerSecondBetweenDetections),decreasing=T),], .05*nrow(dailyMovementsTable))
fastMovements <- fastMovements %>%
  relocate(MPerSecondBetweenDetections, .after = TAG)

avianPredationDFs <- list(
  "longMovements" = longMovements,
  "fastMovements" = fastMovements
)
```

The UTM's are fully converted to Lat/longs for later mapping with leaflet.

```r
#######get lat/longs for plotting with leaflet
#convert events to sf object
#the utms are grs80 and utm zone 13, which corresponds to crs  32613
dailyMovementsTableSF <- sf::st_as_sf(dailyMovementsTable, coords = c("UTM_X", "UTM_Y"), crs = 32613, remove = FALSE)
#convert to lat/long
dailyMovementsTableSFLatLong <- sf::st_transform(dailyMovementsTableSF, latLongCRS)

coordinates <- st_coordinates(dailyMovementsTableSFLatLong)

# Convert the coordinates to a data frame
coordinatesDf <- as.data.frame(coordinates)
# # Extract the lat/long coordinates
dailyMovementsTableSFLatLong$X <- coordinatesDf$X
dailyMovementsTableSFLatLong$Y <- coordinatesDf$Y
```

Periods are added. These are used as a time frame option in the Animations tab. Relevant columns are then selected for the final dataframe.

```r
###get Time periods with movements: used to group together in animation
#time periods comees from WGFP metadata;
timePeriodsClean <- TimePeriods %>%
  mutate(`start date` = janitor::excel_numeric_to_date(as.numeric(`start date`)),
         `end date` = janitor::excel_numeric_to_date(as.numeric(`end date`))
  )
tagsEventsLongwithTpDates <- eventsWithPeriodsSelect %>%
  left_join(timePeriodsClean[,c("periods", "start date", "end date")], by = c("TimePeriod" = "periods")) %>%
  mutate(TimePeriodDates = paste(`start date`, "to", `end date`))

#as of now, movement table still has rows reminiscent from first_last etc which are helpful when you want to know where it ended the day and stuff.
#but if you want to know concise movments, then this will eliminate uneeded rows
#example: 230000142723
dailyMovementsTable1 <- as.data.frame(dailyMovementsTableSFLatLong) %>%
  left_join(tagsEventsLongwithTpDates, by = c("TAG", "Datetime", "Event")) %>%
  select(Date, Datetime, TAG, movement_only, det_type, dist_moved, MPerSecondBetweenDetections, sum_dist,
         ET_STATION, Species, Release_Length, Release_Weight, ReleaseSite, Release_Date, RecaptureSite, River,
         USGSDischargeDaily, WtempF, UTM_X, UTM_Y, X, Y, marker_color, icon_color, TimePeriod, TimePeriodDates, State)
```

Shapefile/Polygon Readins

The file "map_polygon_readins.R" reads in all shapefiles for the movements map. Also contains some graphics options for release sites, stationary antenna sites, and labels for stations.

```
#mapping
source("map_polygon_readins.R")
```

.Rds files

.rds files in R are a binary format used to save R objects, such as data frames, lists, or models, preserving their internal structure and attributes. Unlike CSV files, which store data in a plain text format, .rds files maintain the integrity of R-specific data types and structures, allowing for more complex objects to be saved and loaded efficiently. For example, if you have a data frame df that you want to save as an .rds file, you can use the saveRDS(df, "data.rds") function. To load this data back into R, you would use df <- readRDS("data.rds"). Editing an .rds file involves loading the object, making the desired changes, and then saving it again.

Advantages of .rds files in this app

- Easy to save data and reload it again across different R projects without losing any metadata or attributes
    - Helpful when saving Stationary data from CombineDataApp and moving over to this app
- Preserves data types and structures
    - Ensures that when the data is read into R, it preserves the same format, vs a csv which saves data in plaintext, sometimes necessitating type conversion to change the data back to its original structure when opened in Excel. This is currently seen when we open and save data in Excel that have tag numbers, and every time we save/modify it in Excel, we have to make sure the tag number is saved as a numeric type rather than general (see Creating New Flat Files/Running the Script: Common errors). This was also causing issues with the timestamps in the stationary data, leading to the decision to keep the Stationary Data in .rds form.
- Ability to save objects other than DFs
    - This enables us to save lists and spatial files as flat files created in the runscript (in data/FlatFilesForApp), reducing computational load within the app for faster speed
- Efficiency
    - .rds files are smaller than csvs of the same data. This leads to quicker read-in times and easier file transfers across projects.

As of late 2023, WGFP_Raw (containing all Stationary Antenna data) has been saved as a .rds file in from the update Data app. The new resultant workflow described in "Updating the App".

Shiny Modules

Each tab is split up into its own module. This makes the code reusable (can plug the ui and server into any app easily with the right arguments) and organized. This is where the data processing occurs from the sidebar filters and how graphs and tables are rendered. Read more about Shiny Modules here.

Site, Movement and Species Colors

The colors of the Sites, movements and species used in the app are determined by this chunk of code in the beginning of the app.r file

```
#colors
#rainbow trout color pallete used to assign to Sites:
# "Confluence, Connectivity Downstream, Connectivity Side Channel, Connectivity Upstream, Hitching Post,
# Kaibab Park, Red Barn, Windy Gap Auxiliary, Windy Gap Bypass Channel"
rainbow_trout_colors <- c("#8B8000", "#008080", "#FF69B4", "#FF4500", "#6A5ACD","#32CD32", "#20B2AA", "#FF1C55", "#4682B4
#currently "Changed Rivers", "Downstream Movement" "Initial Release", "No Movement", "Upstream Movement" (5/17/24)
#note: these are a little different than what we see in the map because the map/marker color optoins are very limited.
movementColors <- c("#4B0082", "#8B0000", "#FF8C00", "#253333", "#22bd74") #, "#66FF00"
# currently "LOC" "MTS" "RBT" "RXN" "TGM" (5/17/24)
speciesColors <- c("#FFD700", "#654321", "#4F7942", "#FF7F50", "#1E90FF", "#008080", "#DAA520", "#D2691E", "#9A5ECD")
```

Rainbow_trout_colors is used to assign colors to site in succession of the colors: so Confluence gets assigned #8b8000, Connectivity Downstream is assigned #008080, etc. When a new site is added to the data, it will automatically get assigned a color based off the rainbow_trout_colors vector, provided that the number of sites is less than the length of rainbow_trout_colors. As of 5/24/2024, there are 9 sites and 15 colors in the vector, so we can add 6 more antenna sites that won't require adjustment to this code.

MovementColors and SpeciesColors are similar to rainbow_trout_colors. If we want to change the colors for a given species/movement etc, then change the colors in this code. Note: changing movementColors does not affect the color shown in the map, just the graphs. Go to get_movements_function.R to edit the movement colors on the map.

Known Bugs

There are a few bugs in the app that occasionally show up. Typically, the best remedy is to restart the app. Clear your R environment and close and restart R studio if none of that works.

Sequences

Sometimes when you click "add site" or "drop site", multiple boxes appear/disappear when there should just be one. When displaying the data, there might be error messages in the console. I believe it's related to assigning Ids and maybe not subsequently destroying them effectively. This bug rarely happens and is fixed with a restart on the app.

Movements

- Once the "Toggle Table" button on the movements map didn't turn the table back on after it was closed. Should be a rare bug, restart the app when this occurs.

Animations

Error in if (times[1] <= 1) { : missing value where TRUE/FALSE needed, I'm pretty sure this means you just need to include more data: there's only 1 frame of the selected time period used.

Sometimes when attempting to use facet wrap, there are a few errors that occur that I suspect are memory-related. These include:

```
Error in colour_state_interpolator(data, states) :
  the dims contain missing values
```

```
Error in numeric_state_interpolator(lapply(data, as.numeric), states) :
  negative length vectors are not allowed
```

These errors can be prone to happen in smaller subsetted data, particularly with data subsetted past 2023.

Detection Distance

```
Caused by error in `seq.int()`:
! 'to' must be a finite number
```

Caused when one of the variables doesn't have data for the date range selected. Doesn't cause the app to crash.

Crosstalk QAQC

Sometimes when you click this tab, the tabs for each individual site are rendered many times. I believe it's related to assigning Ids and maybe not subsequently destroying them effectively. Restarting the app and navigating right to the tab works to fix this.

Let me know if there are other questions and concerns, I'm happy to add to this if needed. Let me know
sfigraf@gmail.com