# WGFP_Enc_Summaries_RShinyApp How-To

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

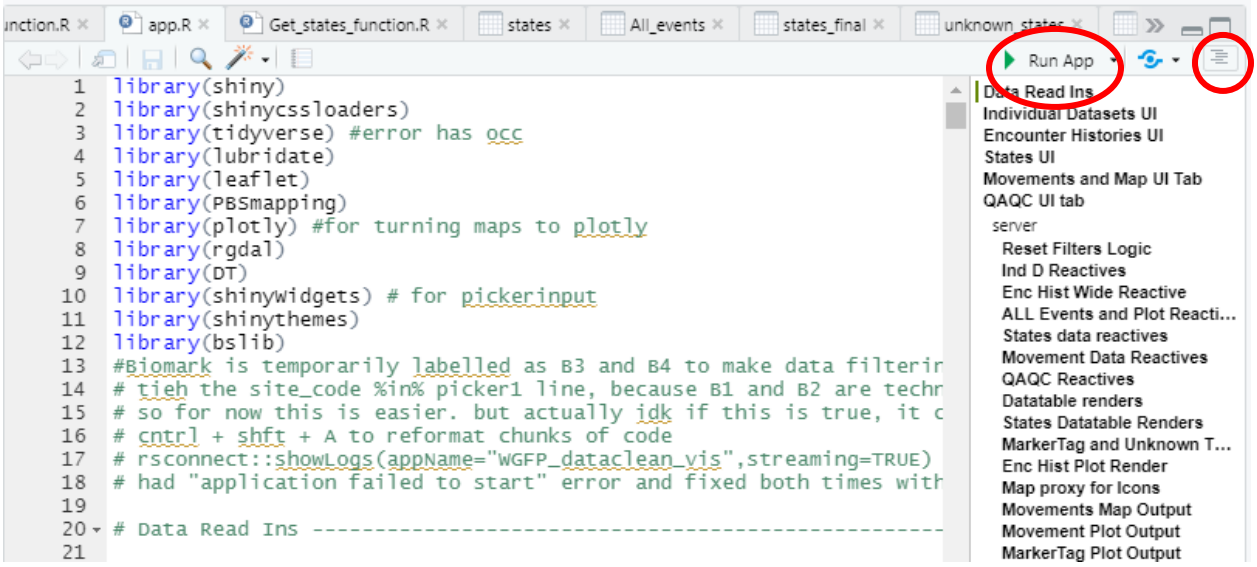
## Uses

- Cleans, combines, and displays data from OregonRFID readers, Biomark antennas, mobile runs, Recaptures, and release data into one file without marker tags or the one weird ghost tag from winter 2021.
- Shows fish movement by day in map, table, plot, and custom animation
- Shows dataset of physical "states"
- QAQC of marker tags, lengths and weights, unknown tags

## How to Open

**OPENING FROM U: DRIVE**

1. Navigate to App location 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. To display the outline of different sections, click the panel on the right.




## CSV ReadIns

The app takes the following .csv files to run. If the column names of any important columns change in the csv's, it will require a bit of easy tweaking to the code.

```
Stationary <- read.csv(paste0("WGFP_Raw_20221227.csv")) #WGFP_Raw_20211130.csv WGFP_Raw_2022
Mobile <- read.csv("WGFP_Mobile_Detect_AllData.csv" , colClasses= c(rep("character",14), rep
Biomark <- read.csv("Biomark_Raw_20221102.csv", dec = ",")
# need to have tagID as a numeric field in the .csv file in order to be read in correctly as
Release <- read.csv("WGFP_ReleaseData_Master.csv", na.strings = c("","  ","NA"), colClasses=c
Recaptures <- read.csv("WGFP_RecaptureData_Master.csv", na.strings = c("","  ","NA"), colClass
#ghost tag df
#avitaion predation
AvianPredation <- read_csv("WGFP_AvianPredation.csv", col_types = cols(TagID = col_character
GhostTags <- read_csv("WGFP_GhostTags.csv",
                      col_types = cols(TagID = col_character()))
```

When a new version of these files comes out, simply change the first argument of the read_ins to the new file name. For example, WGFP_Raw_20220203.csv would become WGFP_Raw_20220303.csv.Put the old file in an archive folder.

**Column names and order matter**. The read-ins and code could be tweaked so that at least column order won't matter (use read_csv and specify column types instead of read.csv), but names will still matter.

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

- Stationary
  - Combined file of all stationary detections. Obtained from 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
  - This is a sample of what the data should look like. The most important columns are DTY, ARR, TAG, and SCD. The ARR column can sometimes get corrupted and brought in incorrectly for whatever reason. This is easiest seen when using the CombiningData_RShinyApp and outlined in that How-To

Filter

| | Code | DTY | ARR | TRF | DUR | TTY | TAG | SCD | ANT | NCD | EFA |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | S | 2021-03-02 | 1:01:11 PM | G | 00:00.9 | W | 0000_0000000000005394 | RB1 | A1 | 10 | 0.8 |
| 2 | S | 2021-03-02 | 1:16:06 PM | G | 00:00.9 | W | 0000_0000000000005394 | RB1 | A1 | 10 | 0.8 |
| 3 | S | 2021-03-02 | 1:22:41 PM | G | 00:00.0 | R | 0000_0495240232956782 | RB1 | A1 | 1 | 0.8 |
| 4 | S | 2021-03-02 | 1:23:55 PM | G | 00:00.0 | R | 0000_0495240232956782 | RB1 | A1 | 1 | 0.8 |
| 5 | S | 2021-03-02 | 1:24:48 PM | G | 00:00.0 | R | 0000_0495240232956782 | RB1 | A1 | 1 | 0.8 |
| 6 | S | 2021-03-02 | 1:31:00 PM | G | 00:00.8 | W | 0000_0000000000005394 | RB1 | A1 | 9 | 0.8 |
| 7 | S | 2021-03-02 | 1:39:49 PM | G | 00:00.0 | R | 0000_0495240232956782 | RB1 | A1 | 1 | 0.8 |
| 8 | S | 2021-03-02 | 1:45:55 PM | G | 00:00.9 | W | 0000_0000000000005394 | RB1 | A1 | 10 | 0.8 |
| 9 | S | 2021-03-02 | 1:52:30 PM | G | 00:00.0 | R | 0000_0495240232956782 | RB1 | A1 | 1 | 0.8 |
| 10 | S | 2021-03-02 | 1:56:50 PM | G | 00:00.0 | R | 0000_0495240232956782 | RB1 | A1 | 1 | 0.8 |

- Mobile

    - 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
    - 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
    - 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" , "UTM_X", "UTM_Y", "Species", "Length" , "Weight", "TagType", "TagID", "QAQC", "TagSize", "Ant" , "Event", "FinClip", "Mortality", "Comments"
- Recaptures
    - File of all fish that were recaptured found at U:\Projects\Colorado_River\Windy_Gap_FishMovementStudy\Data\RFID\Recaptures
    - Column names and order: "Num" , "RS_Num", "River" , "RecaptureSite" ,"Date", "Time", "UTM_X" , "UTM_Y", "Species" , "Length", "Weight", "TagType", "TagID", "QAQC" , "TagSize" , "Ant", "Event", "FinClip", "Mortality" , "Comments"
- Avian Predation
    - csv of tags succumbed to avian predation with a date of predation
    - used in the "States" function to assign a ghost or predated state
- Ghost tags
    - csv of ghost tags with date of ghost tag
    - used in the "States" function to assign a ghost or predated state
    - Column names: RS_Num, River, ReleaseSite, ReleaseDate, Species, Length, Weight, TagID, TagSize, Event, GhostDate, UTM_X, UTM_Y, Comments

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
```

# Functions

There are currently 8 functions used in the app.

## All_Combined_events_function

- Main function that combines and cleans detections and release files
- Takes Stationary, Mobile, Biomark, Release, and Recapture files as arguments
- Returns:
  - 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
  - All_Events: cleaned dataset of all 5 input datasets, containing every detection/event from each tag that hit an antenna, complete with release info.
  - 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

### Under the Hood

The app first splits Stationary dataset into a dataset with no marker tags, and one with only marker tags. It also removes known test tags

```
  WGFP_NoMarkers <- Stationary %>%
    mutate(TAG = str_replace(str_trim(TAG), "\\_", "")) %>%
    filter(str_detect(TAG, "^900"),
           !TAG %in% c("900230000102751","900226001581072","900230000004000"))

#marker tag only file
  Markers_only <- Stationary %>%
    mutate(TAG = str_replace(str_trim(TAG), "\\_", "")) %>%
    filter(
      str_detect(TAG, "^0000000")
    )
```

The Marker tags are cleaned, first putting them into a mdy date format.

Then it goes through a timestamp cleaning process that is necessary because there are timestamps that have AM/PM, and others that do not. So we need to get them to military time.

```
Markers_only1 <- Markers_only %>%
  mutate(
    DTY =
      ifelse(str_detect(DTY, "/"),
                 as.character(mdy(DTY)),
                 DTY),
    DTY2 = as.Date(DTY))


Markers_only2 <- Markers_only1 %>%
  #this is the same process that all_detections goes through
  mutate(Scan_Time1 = case_when(str_detect(ARR, "AM") & str_detect(ARR, "^12:") ~ hms(ARR) - hours(12),
                                str_detect(ARR, "PM") & str_detect(ARR, "^12:") ~ hms(ARR),

                                str_detect(ARR, "AM") & str_detect(ARR, "^12:", negate = TRUE) ~ hms(ARR),
                                str_detect(ARR, "PM") & str_detect(ARR, "^12:", negate = TRUE) ~ hms(ARR) + hours(12),
                                #if it doesn't detect PM or AM just do hms(ARR)
                                str_detect(ARR, "PM|AM") == FALSE ~ hms(ARR)),
         Scan_Time2 = as.character(as_datetime(Scan_Time1)),
         CleanARR = str_trim(str_sub(Scan_Time2, start = 11, end = -1))
  ) %>%

  select(Code, DTY2, ARR, CleanARR, TRF, DUR, TTY, TAG, SCD, ANT, NCD, EFA) %>%
  rename(DTY = DTY2)
```

Then the rest of the stationary data without marker tags or test tags is cleaned, including changing antenna names for the upcoming multiplexor, formatting dates, assigning UTMs for each antenna, and removing duplicate rows if there are any

```
## rest of getting "clean" windy gap stationary data
### Subset Detection Type "Codes" to only include Summary (S) and Individual (I) ###
WGFP_Clean= data.frame(WGFP_NoMarkers[which(WGFP_NoMarkers$Code == "I" | WGFP_NoMarkers$Code == "S"),])

#### Add Lat Longs to detections ###

# takes out 900 from TAG in WGFP Clean
# also takes out duplicate rows
WGFP_Clean <- WGFP_Clean %>%
  mutate(TAG = ifelse(str_detect(TAG, "^900"), str_sub(TAG, 4,-1), TAG),
         SCD = case_when(SCD == "CD7" & ANT == "A1" ~ "CD7",
                         SCD == "CD7" & ANT == "A2" ~ "CD8",
                         SCD == "CD7" & ANT == "A3" ~ "CD9",
                         SCD == "CD7" & ANT == "A4" ~ "CD10",
                         TRUE ~ SCD),
         DTY = ifelse(str_detect(DTY, "/"),
                      as.character(mdy(DTY)),
                      DTY)) %>%

  # mutate(TAG = case_when(str_detect(TAG, "^900") ~ str_sub(TAG, 4,-1),
  #                        str_detect(TAG, "!^900") ~ TAG)) %>%
  # assigning UTM's are important because they are plotted later when getting stations file in GIS
  mutate(UTM_X =case_when(SCD == "RB1" | SCD == "RB2" ~ "412489",
                          SCD == "HP3" | SCD == "HP4" ~ "414375",
                          SCD == "CF5" | SCD == "CF6" ~ "416965",
                          SCD == "CD7" | SCD == "CD8" | SCD == "CD9" | SCD == "CD10" ~ "415801",
                          SCD == "CU11" | SCD == "CU12" ~ "416802"),
         UTM_Y = case_when(SCD == "RB1" | SCD == "RB2" ~ "4439413",
                           SCD == "HP3" | SCD == "HP4" ~ "4440241",
                           SCD == "CF5" | SCD == "CF6" ~ "4439369",
                           SCD == "CD7" | SCD == "CD8" | SCD == "CD9" | SCD == "CD10" ~ "4439899",
                           SCD == "CU11" | SCD == "CU12" ~ "4439507")) %>%
  distinct()
```

Biomark dataset is then cleaned, including changing the reader ID since we want it in B3-B6 format, changing date format, filtering out test tags and marker tags, assigning UTMs, and removing duplicate rows

```
# biomark cleaning, getting dates into uniform format,
biomark2 <- Biomark %>%
  mutate(TAG = str_replace(DEC.Tag.ID, "\\.", ""),
         Reader.ID = case_when(Reader.ID == "A1" | Reader.ID == "B1" ~ "B3",
                               Reader.ID == "A2" | Reader.ID == "B2" ~ "B4",
                               Reader.ID == "A3" ~ "B5",
                               Reader.ID == "A4" ~ "B6",
                               TRUE ~ Reader.ID),
         #make a column for Scan>Date if parentheses are detected in the string,
         # and we want to convert it to YYYYMMDD format. elsewise, leave it as i
         Scan.Date = ifelse(str_detect(Scan.Date, "/"),
                            as.character(mdy(Scan.Date)),
                            Scan.Date)
           ) %>%
  filter(!TAG %in% c("900230000102751", "900226001581072", "999000000007586", "9

  # from gis: B1 416026, 4440196
  #B2: 420727.9, 4437221
  # b3 is wg, b4 is kaibab
  # b5 is river run
  # b6 is fraser river canyon
  mutate(UTM_X =case_when(Reader.ID == "B3" ~ "416026",
                          Reader.ID == "B4" ~ "420728",
                          Reader.ID == "B5" ~ "419210",
                          Reader.ID == "B6" ~ "424543"),
         UTM_Y = case_when(Reader.ID == "B3" ~ "4440196",
                           Reader.ID == "B4" ~ "4437221",
                           Reader.ID == "B5" ~ "4439961",
                           Reader.ID == "B6" ~ "4435559")) %>%
  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)
```

Scan time is now cleaned like it was for marker tags, but with all the data

```
#cleaning timestamps for mobile and old stationary detections mainly
if (    length(unique( str_detect(All_detections$Scan_Time, "PM|AM"))) > 1) {
  All_detections1 <- All_detections %>%
    mutate(Scan_Time1 = case_when(str_detect(Scan_Time, "AM") & str_detect(Scan_Time, "^12:") ~ hms(Scan_Time) - hours(12),
                                  str_detect(Scan_Time, "PM") & str_detect(Scan_Time, "^12:") ~ hms(Scan_Time),

                                  str_detect(Scan_Time, "AM") & str_detect(Scan_Time, "^12:", negate = TRUE) ~ hms(Scan_Time),
                                  str_detect(Scan_Time, "PM") & str_detect(Scan_Time, "^12:", negate = TRUE) ~ hms(Scan_Time) + hours(12),
                                  #if it doesn't detect PM or AM just do hms(Scan_Time)
                                  str_detect(Scan_Time, "PM|AM") == FALSE ~ hms(Scan_Time)),
    ) %>%
    mutate(Scan_Time2 = as.character(as_datetime(Scan_Time1)),
           clean_time = str_trim(str_sub(Scan_Time2, start = 11, end = -1))) %>%

    select(Scan_Date, clean_time, TAG, Site_Code, UTM_X, UTM_Y ) #%>%
    #rename(Scan_Time = (clean_time))
}
```

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

```
All_detections2 <- All_detections1 %>%
  filter(Scan_Date >= as.Date("2020-08-06")) %>% #right before the first date of
  mutate(
    #datetime1 = as.POSIXct(paste(Scan_Date, Scan_Time),format="%Y-%m-%d %H:%M:%S
    Scan_DateTime = ymd_hms(paste(Scan_Date, clean_time))) %>%
  #rename(Scan_DateTime = datetime2) %>%
  select(Scan_Date, clean_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.

```
### all detections and recaps and release "EVENTS" DF

  #getting timestamps in order and getting relevant columns
  Release1 <- Release %>%
    rename(TAG = TagID) %>%
    mutate(TAG = str_trim(TAG),
           Date = mdy(Date),
           Time1 = case_when(str_length(Time) > 5 ~ as_datetime(hms(Time)),
                             str_length(Time) <= 5 ~ as_datetime(hm(Time))), #warning message: problem with mutate(time1,
           Time2 = str_sub(Time1, start = 11, end = -1),
           DateTime = ymd_hms(paste(Date, Time2))) %>%
    select(RS_Num,River,ReleaseSite,Date, Time2, DateTime,UTM_X,UTM_Y,Species,Length,Weight,TAG,TagSize,Ant,Event) %>%
    rename(Time = Time2)

  #getting timestamps in order and getting relevant columns

  recaps1 <- Recaptures %>%
    rename(TAG = TagID) %>%
    filter(!Date %in% c("", " ", NA)) %>%
    mutate(TAG = str_trim(TAG),
           Date = mdy(Date),
           # added in like I did the release file, functionality for when the time contains just HH:mm and hh:mm:ss
           Time1 = case_when(str_length(Time) > 5 ~ as_datetime(hms(Time)),
                             str_length(Time) <= 5 ~ as_datetime(hm(Time))),
           Time2 = str_sub(Time1, start = 11, end = -1),
           DateTime = ymd_hms(paste(Date, Time2))) %>%
    select(RS_Num,River,RecaptureSite,DateTime,Date,Time2,UTM_X,UTM_Y,Species,Length,Weight,TAG,TagSize,Ant,Event) %>%
    rename(Time = Time2,
           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
All_Detections_1_merge <- All_detections2 %>%
  mutate(Date = as.Date(Scan_Date)) %>%
  rename(
    Time = clean_time,
    DateTime = Scan_DateTime,
    Event = Site_Code)

##


# this file is used in enc_hist_summary_wide
recaps_detections <- bind_rows(All_Detections_1_merge, recaps1)

detections_release_recaps <- bind_rows(recaps_detections, Release1)


# bind rows vs left join; bind rows will make it so there is a "release" or "recapture"


#fills in release info so it is known at any row of detection
filled_in_release_rows <- left_join(detections_release_recaps, Release1, by = c("TAG"))
```

Then there's some minor formatting/renaming for display purposes in the app, and this is the final df that is used in the All Encounters Tab!

```
#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

filled_in_release_rows_condensed <- filled_in_release_rows %>%
  select(Date.x, Time.x, DateTime.x, TAG, Event.x, Species.y, Length.y, Weight.y, ReleaseSite.y, Date
  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) %>%
  replace_na(list(Species = "No Info", ReleaseSite = "No Info"))
```

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")
```

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)
```

## Spatial_join_function

- Joins detections and events to the shapefile of stations in order to later help calculate states and distance moved for each fish.
- Takes condensed events (arg 1), made from all_combined_events_function (all_events_most_relevant). Also takes simple_stations (arg2), which is usually read in as simple_stations2 in the polygon_read_ins.r file
- Returns a dataframe of condensed_events with all station data attached.

### Under the Hood

This function first converts the utm's to numeric, assigns a projection/zone/datum, converts to lat/long, then to a spatial points object, then to sf object, then joins with the stations shapefile, also converted to an sf object. The joining is by the nearest feature, so if a detection is not quite on top of the station, it will get assigned the closest one.

```
condensed_events <- condensed_events %>%

  mutate(
    X = as.numeric(UTM_X),
    Y = as.numeric(UTM_Y)
  ) #end of mutate

# assigning projection to ready df lat/longs for plotting
attr(condensed_events, "zone") = "13"
attr(condensed_events, "projection") = "UTM"
attr(condensed_events, "datum") = "GRS80"

# need a column that has x and Y for this
# converts lutms to lat/long
condensed_events <- convUL(condensed_events, km=FALSE, southern=NULL)

#converting lat/long entries to spatial points dataframe
# needs to have a df of just coordinates (xy)
xy <- condensed_events %>%
  select(X, Y)

spdf <- SpatialPointsDataFrame(coords = xy, data = condensed_events,
                               proj4string = CRS("+init=epsg:4326"))

## making sf objects
detections_sf <- st_as_sf(spdf)
stations_sf <- st_as_sf(simple_stations)

joined <- st_join(detections_sf, stations_sf, st_nearest_feature)
#need to convert class sf object back to dataframe so that it goes faster in combine_events_stations_function
station_data <- as.data.frame(joined)
```

## PrepareforStatesMovementsandSummary_function.R

- Puts stations onto a condensed All_events dataframe

- Takes StationData returned from spatial_join function.
- Returns:
  - All_events_days1: a condensed dataframe of all pertinent daily detection info for each tag with stations 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.

Under the Hood

The columns are renamed and the River is assigned for specific antennas

```
mutate(
  #River also needs to be assigned for new detections
  River = case_when(
    (Event %in% c("RB1", "RB2")) ~ "Colorado River", # there is no is.na here because RB UTM
    (Event %in% c("HP3", "HP4")) ~ "Colorado River",
    (Event %in% c("CF5", "CF6")) ~ "Colorado River",
    (Event %in% c("CD7", "CD8", "CD9", "CD10", "CU11", "CU12")) ~ "Connectivity Channel",
    (Event %in% c("B3", "B5")) ~ "Colorado River",
    (Event %in% c("B4", "B6")) ~ "Fraser River",
    TRUE ~ River
  ),
```

This part accounts for stations up the Fraser River, because those stations start at 0. This adds the station where the Colorado/Fraser confluence to the Fraser stationing. Then duplicate rows are taken out (shouldn't be any) and relevant columns are selected

```
  # 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 + 10120, #10120 is above Fraser River Confluence; pre-construciton was 9566
                         River %in% c("Colorado River", "Connectivity Channel") ~ ET_STATION,
                         TRUE ~ ET_STATION)
%>%
# this line just makes the df smaller if htere are duplicates; usually doesn't change anything since All_events has a line that does this also i
distinct(Datetime, Event, TAG, .keep_all = TRUE) %>%
select(Date, Time, Datetime, TAG, Event, Species, Release_Length, Release_Weight, ReleaseSite, Release_Date, RecaptureSite, River, Recap_Length,
```

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. The type of movement is condensed to a "detection type" field. This is helpful in the movements map where it doesn't necessarily matter which antenna specifically is hit, only if the stationing is different.

```r
#getting number of daily events
DailyMovements_withStations <- DailyMovements_withStations %>%
  group_by(Date, TAG) %>%
  mutate(c_number_of_detections = n(),
         daily_unique_events = length(unique(Event))
         ) %>%
  ungroup()
#generating generic event title for movements map
DailyMovements_withStations <- DailyMovements_withStations %>%
  mutate(
    #this part is used in movemnets map
    det_type = case_when(str_detect(Event, "RB1|RB2") ~ "Red Barn Stationary Antenna",
                         str_detect(Event, "HP3|HP4") ~ "Hitching Post Stationary Antenna",
                         str_detect(Event, "CF5|CF6") ~ "Confluence Stationary Antenna",
                         str_detect(Event, "CD7|CD8|CD9|CD10") ~ "Connectivity Channel Downstream Stationary Antenna",
                         str_detect(Event, "CU11|CU12") ~ "Connectivity Channel Upstream Stationary Antenna",
                         str_detect(Event, "B3") ~ "Windy Gap Dam Biomark Antenna",
                         str_detect(Event, "B4") ~ "Kaibab Park Biomark Antenna",
                         str_detect(Event, "B5") ~ "River Run Biomark Antenna",
                         str_detect(Event, "B6") ~ "Fraser River Canyon Biomark Antenna",
                         str_detect(Event, "M1|M2") ~ "Mobile Run",
                         Event == "Recapture" ~ "Recapture",
                         TRUE ~ Event),
    #need to check this function out given new stationing and connectivity channel
    above_below = case_when(
      ET_STATION >= 8330 ~ "Above the Dam",
      ET_STATION < 8330 ~ "Below the Dam"
    )
```

## 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”
- Takes recaps_and_all_detections from All_Combined_events_function, Release data, combined_events_stationsfrom PrepareforStatesMovementsandSummary_function, and States_summarized from the Get_states_function
- It tdoesn’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
  - Unknown_Tags: dataframe of tags that have encounters, but have no release info

Under the Hood

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. This is the start of the summary file.

```
all_enc12 <- recaps_and_all_detections %>%
  count(TAG, Event, name = "Encounters")

all_enc12 <- pivot_wider(data = all_enc12, id_cols = TAG, names_from = Event, values_from = Encounters)

x <- all_enc12 %>%
  replace_na(list(Species = "No Info", ReleaseSite = "No Info"))

#all_enc12[is.na(all_enc12)]=0
#### NEED To make this compatible with new antennas!!

ENC_ALL <- all_enc12 %>%
  rename(RB1_n = RB1,
         RB2_n = RB2,
         HP3_n = HP3,
         HP4_n = HP4,
         CF5_n = CF5,
         CF6_n = CF6,
         CD7_n = CD7,
         CD8_n = CD8,
         CD9_n = CD9,
         CD10_n = CD10,
         CU11_n = CU11,
         CU12_n = CU12,
         M1_n = M1,
         M2_n = M2,
         B3_n = B3,
         B4_n = B4,
         B5_n = B5,
         B6_n = B6,
         Recap_n = Recapture
  ) %>%
  select(TAG, RB1_n,RB2_n,HP3_n, HP4_n, CF5_n, CF6_n, CD7_n, CD8_n, CD9_n, CD10_n, CU11_n, CU12_n, M1_n,
```

The release data is then joined to the summary file. This is also when the unkown 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 ###
Release1 <- release_data %>%
  rename(TAG = TagID) %>%
  mutate(TAG = str_trim(TAG)) %>%
  replace_na(list(Species = "No Info", ReleaseSite = "No Info

# was geting a massive dataframe because the Release df is ca
#need to actually join on full join not merge
ENC_Release <- full_join(Release1, ENC_ALL,  by = "TAG")

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

If there wasn't a count for a fish, that NA gets changed to 0. True/falsecolumns are made on whether a fish hit a specific antenna or was recaptured.

```
#ENC_Release11$TAG[3433:nrow(ENC_Release11)]
ENC_Release[is.na(ENC_Release)]=0 #gets rest of the number count columns to 0 from NA

#### Make 1 or 0 for encounter history rather than counts ###
#gets df with TF of whether a fish was detected at a antenna
ENC_Release1 <- ENC_Release %>%
  mutate(RB1 = (RB1_n >0),
         RB2 = (RB2_n >0),
         HP3 = (HP3_n >0),
         HP4 = (HP4_n >0),
         CF5 = (CF5_n >0),
         CF6 = (CF6_n >0),
         CD7 = (CD7_n >0),
         CD8 = (CD8_n >0),
         CD9 = (CD9_n >0),
         CD10 = (CD10_n >0),
         CU11 = (CU11_n >0),
         CU12 = (CU12_n >0),
         M1 = (M1_n >0),
         M2 = (M2_n >0),
         B3 = (B3_n >0),
         B4 = (B4_n>0),
         B5 = (B5_n >0),
         B6 = (B6_n>0),
         Recapture = (Recap_n > 0))
```

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.

```
totalcols <- ncol(ENC_Release1)

ENC_Release2 <- ENC_Release1 %>%
  #counts number of TRUE across specified rows. -SG
  # need to have parentheses (totalcols-1) that's why i was getting bad numbers on
  #added 8 new columns for new antennas
  mutate(
    TotalEncounters = rowSums(ENC_Release1[(totalcols-18):totalcols] == TRUE),

    TotalAntennas1 = rowSums(ENC_Release1[(totalcols-18):(totalcols-1)] == TRUE),
    TotalStationary = rowSums(ENC_Release1[(totalcols-18):(totalcols-7)] == TRUE),
    TotalMobile = rowSums(ENC_Release1[(totalcols-6):(totalcols-5)] == TRUE),
    TotalBiomark = rowSums(ENC_Release1[(totalcols-4):(totalcols-1)] == TRUE),
    TotalRB = rowSums(ENC_Release1[(totalcols-18):(totalcols-17)] == TRUE),
    TotalHP = rowSums(ENC_Release1[(totalcols-16):(totalcols-15)] == TRUE),
    TotalCF = rowSums(ENC_Release1[(totalcols-14):(totalcols-13)] == TRUE),
    TotalCD = rowSums(ENC_Release1[(totalcols-12):(totalcols-9)] == TRUE),
    TotalCU = rowSums(ENC_Release1[(totalcols-8):(totalcols-7)] == TRUE),
  ) %>%
  # just says if the fish was ever detected at these sites
  mutate(RB = (RB1_n > 0 | RB2_n >0),
         HP = (HP3_n > 0 | HP4_n >0),
         CF = (CF5_n > 0 | CF6_n >0),
         CD = (CD7_n > 0 | CD8_n >0 | CD9_n > 0 | CD10_n >0),
         CU = (CU11_n > 0 | CU12_n >0),
         Biomark = (B3_n > 0 | B4_n >0 | B5_n > 0 | B6_n >0),
         Mobile = (M1_n > 0 | M2_n >0)) %>%
  filter(!UTM_X %in% c(0, NA)) # one way to filter out tags that don't have any sor
```

This part brings the condensed df with stations back in for summaries. It first couts the number of encounters each fish had above and below the dam. Then describes that event (eg "Mobile Detection above the dam"). Then if 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. It also changes the NA's to FALSE, because when you use count(), you only sum fish that have had encounters.

```
above_below_counts <- 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))

above_below_counts1 <- pivot_wider(data = above_below_counts, id_cols = TA
above_below_counts2 <- above_below_counts1 %>%
  select(TAG, `Release Above the Dam`, `Release Below the Dam`,`Recapture
#turns all the NA's made to FALSE
above_below_counts2[is.na(above_below_counts2)] = FALSE
```

The data is joined with the original summary file. A column is made based on those new joined columns describing if a fish went through the dam or not.

```
ENC_Release3 <- left_join(ENC_Release2, above_below_counts2, by = "TAG")
### need to figure out how connectivity channel fits into this part?
ENC_Release4 <- ENC_Release3 %>%
  mutate(through_dam = case_when(
    (RB1|RB2|HP3|HP4|B3|`Release Below the Dam`|`Recapture Below the Dam`
    (RB1|RB2|HP3|HP4|B3|`Release Below the Dam`|`Recapture Below the Dam`
    (RB1&RB2&HP3&HP4&B3&`Release Below the Dam`&`Recapture Below the Dam`

  ))
```

The states_summarized dataframe is then brought in and added. This has data about how a fish got through the dam, if it used the connectivity channel or not.

```
# left joining states summary to enc_release
ENC_Release5 <- left_join(ENC_Release4, States_summarized, by = "TAG")
#rearranging so that Tag is first column shown
ENC_Release5<- ENC_Release5 %>%
  select(TAG, 1:ncol(ENC_Release5))
```

The distance a fished moved is then calculated from the condensed df detections (see the All Events Combined description to see how this is formed). 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 that warning/edge case would apply to have actually been predated.

The total movement is calculated for each fish in the sequence of its encounter history. This is then joined to the summary dataframe as well.

Dummy rows are also taken out

```
###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)

ENC_Release6 <- ENC_Release5 %>%
  left_join(sum_dist1, by = "TAG")

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

## Get_movements_function

- Makes a condensed 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.
- Takes all_events_days1 from PrepareforStatesMovementsandSummary_function
- Returns:
  - Movement_table_no_trans: dataframe where only daily movements are included per tagged fish on unique antennas and UTM's. See "Movements defined" for about them

Under the Hood

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.

Lastly in this code chunk, X and Y are defined as columns in preparation for changing the utms to lat/longs. Note: there are a couple places in the app where UTMs are converted to lat/longs, and the code would be a little more elegant if these were just all made to sf objects, but this stuff also all works fine.

```
movement_table_notrans <- 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, Recaptur
  #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 chan
  #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 th
  # otherwise, just subtract current station from previous
  mutate(dist_moved = case_when(lag(ET_STATION, order_by = Datetime) > 10120 & ET_STATION >10120 & River != lag(River, order_by = Datet
                                TRUE ~ ET_STATION - lag(ET_STATION, order_by = Datetime)
                                ),
         sum_dist = (sum(abs(dist_moved), na.rm = TRUE)),


         movement_only = case_when(lag(ET_STATION, order_by = Datetime) > 10120 & ET_STATION >10120 & River != lag(River, order_by = Da
                                   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"
          # these also correspond to the movements maps, so if you add or change a color you should change it on the movements map as w
         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",
         ),
         X = as.numeric(UTM_X),
         Y = as.numeric(UTM_Y)
  ) #end of mutate
```

The UTM's are fully converted to Lat/longs, then the df is parred down with the distinct() function like it has before, but this time, it takes "movement" into account instead of "first/last". Movement_table_notransitions1 is what is mapped and filtered in the app.

```
# assigning projection to ready df lat/longs for plotting
attr(movement_table_notrans, "zone") = "13"
attr(movement_table_notrans, "projection") = "UTM"
attr(movement_table_notrans, "datum") = "GRS80"

# need a column that has x and Y for this
# converts lutms to lat/long
movement_table_notrans <- convUL(movement_table_notrans, km=FALSE, southern=NULL)
#as of now, movement table still has rows reminiscent from first_last etc which are helpful when you want to know wher
#but if you want to know concise movments, then this will eliminate uneeded rows
#example: 230000142723
movement_table_notrans1 <- movement_table_notrans %>%
  distinct(Date, TAG, det_type, movement_only, UTM_X, UTM_Y, .keep_all = TRUE) %>%
  select(Date, Datetime, TAG, movement_only, det_type, dist_moved, sum_dist, ET_STATION, Species, Release_Length, Rele

#giving id column to make map proxy easier
# actually the id column needs to be remade every time a filter is applied. See the movements_df_reactives
#movement_table_notrans1$id <- seq.int(nrow(movement_table_notrans1))
```

## Get_States_function

- Makes dataframe of weekly "states" either above or below the dam. Below the dam is state A and above is B. State C is a detection in the connectivity channel. Ghost state is G, and Predated state is P
  - Any detection, recapture, or release is registered as a state

        - A fish is inferred to remain in that state until it becomes a ghost tag or is detected/recaptured again.
    - Takes combined_events_stations from PrepareforStatesMovementsandSummary_function, the Ghost tags df, and the aviation predation df.
    - Returns:
        - All_States: Dataframe of daily States of fish. Certain assumptions are made to infer transitions, such as if a fish hits 1/2 antennas at a site then hits an upstream antenna, it is said that an upstream transition originally occurred and it just missed 1 antenna. For more assumptions see thee Daily States Tab section
        - Days_and_states_wide: same data as all_states but in wide format where days are the columns, TAG is the first column, and values are states
        - Flagged Movements: States that couldn't be accounted for in the code

Under the Hood

The 3 dataframes are joined then the df is cut down to the most succinct: rows with a distinct combo of Tag, Event, Datetime, UTM's, and first and last event of day.

```
# these dates are cleaned before they go into this function
ghost_tag_df <- GhostTags %>%
  rename(TAG = TagID) %>%
  select(TAG, GhostDate)

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

#joining with ghost tag df
wghost_date <- left_join(combined_events_stations, ghost_tag_df, by = c("TAG"))
# joining with avian predation
#Shouldn't matter really, but this just cuts down unnecesary rows. left_joining creates more rows than df1 if there are duplicate v
#some new rows are created then when joining to the ghost tag df because many of the ghost tags have multiple detections after the
# Im'm just cutting out these excess rows but I think they would get cut down anyway later in this function. As long as each ghost
wghost_av <- left_join(wghost_date, av_pred_df, by = c("TAG")) %>%
  distinct(TAG, Event, Datetime, UTM_X, UTM_Y, first_last, .keep_all = TRUE)
```

The states are then assigned based on above/below the dam, ghost tag, and predation date

```
states1 <- wghost_av %>%
  filter(!TAG %in% c('230000999999')) %>%
  mutate(
    #the case_whens also are a priority list, so important not to rearange these
    #might have to readjust 8330 stationing
    state1 = case_when(Date >= GhostDate ~ "G",
                       Date >= PredationDate ~ "P",
                       str_detect(Event, "CD7|CD8|CD9|CD10|CU11|CU12") ~ "C",
                       ET_STATION <= 8330 ~ "A",
                       ET_STATION > 8330 ~ "B")
  )
```

The states are then kinda "squished" together into one row on a weekly basis, and duplicated letters in a row are removed, because a fish doesn't need a weekly states history that says "AAAA", it just needs "A". A column is also calculated to get the amount of weekly unique events.

The df then gets rid of duplicated rows, since each fish has multiple rows of states in the same week at this point. Relevant columns are selected and renamed

Some summary information for the fish are then calculated based off these states. In the states_summarized df, the states are "squished" together for all states the fish has. From there, you can see if a fish went above/below the dam, and used the channel, which is what the new columns check. These columns are eventually joined back into the encounter histories summaries wide dataframe.

```
states2 <- states1 %>%
  group_by(weeks_since, TAG) %>%
  #arranging my datetime ensures that all states will be recorded in the correct order
  arrange(Datetime) %>%
  mutate(
    teststate_2 = paste(state1, collapse = ""),
    teststate_3 = gsub('([[:alpha:]])\\1+', '\\1', teststate_2), #removes consecutive letters
    weekly_unique_events = length(unique(Event))
  )


#this is now a weekly chart
states_final <- states2 %>%
  distinct(weeks_since, TAG, teststate_3, .keep_all = TRUE) %>%
  select(Date, weeks_since, TAG, teststate_3, det_type, ReleaseSite, Species, Release_Length, Release_Weight, c_number_of_detection
  rename(State = teststate_3)

# this makes some columns from all states of fish
states_summarized <- states1 %>%
  group_by(TAG) %>%
  arrange(Datetime) %>%
  mutate(teststate_2 = paste(state1, collapse = ""),
         teststate_3 = gsub('([[:alpha:]])\\1+', '\\1', teststate_2), #removes consecutive letters
         #new columns to sa yif fish stayed above or below?
         went_above_dam_noChannel = str_detect(teststate_3, "AB"),
         went_below_dam_noChannel = str_detect(teststate_3, "BA"),
         went_below_dam_throughChannel = str_detect(teststate_3, "BCA"),
         went_above_dam_throughChannel = str_detect(teststate_3, "ACB"),
         entered_channel_from_DS = str_detect(teststate_3, "AC"),
         entered_channel_from_US = str_detect(teststate_3, "BC"),

  ) %>%
  select(TAG, went_above_dam_noChannel, went_below_dam_noChannel,went_below_dam_throughChannel,went_above_dam_throughChannel,entere
  distinct(TAG, .keep_all = TRUE)
```

This is an idea to put all the wonky states in place so it would be easy to see, but it isn't fully compatible with the app yet and therefore isn't used. My idea is that instead they would be in one tab on the states page so less one manual parsing is needed.

```
checking <- states_final %>%
  group_by(TAG) %>%
  arrange(Date) %>%
  mutate(through_dam1 = case_when(det_type == "Release" ~ "Initial Release",
                                  str_sub(State,-1,-1) == "A" & lag(str_sub(State,-1,-1) %in% c("B", "C"), order_by = Date) ~ "Went Below Dam",
                                  str_sub(State,-1,-1) == "B" & lag(str_sub(State,-1,-1) %in% c("A", "C"), order_by = Date) ~ "Went Above Dam",
                                  State %in% c("BA", "CA", "BCA") ~ "Went Below Dam",
                                  State %in% c("AB", "CB", "ACB") ~ "Went Above Dam",
                                  State == lag(str_sub(State,-1,-1), order_by = Date) ~ "No state change",
                                  # TRUE ~ NA

  )
  )

unknown_states <- checking %>%
  filter(is.na(through_dam1) & !det_type %in% c("Release", "Recapture and Release", "Recapture"))
```

## 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")
```

## Modifying, Updating, Adding New Antenna/Stations

Need to update this to new function: prepare for states, movements, summaries

### Adding to the map

- Head 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 rgdal package, read in the .shp file,
  - If you are adding an antenna or making new stations, you can change the "layer name" argument to the name of the new .shp or .rds file file, and it will automatically get added to the map for that layer.

```
layer_location <- file.path("./gis/")


stationary_antennas <- readOGR(dsn = layer_location, layer = "stationary_points")
stationary_antennas <- sp::spTransform(stationary_antennas, CRS("+init=epsg:4326"))
```

- The espg:4326 converts the projection to something usable
- 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 that you want to put on the map, you'll have to add it to the map.

```
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"))
```

- 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

## Incorporating the new antenna with the data

First, go to the dummy_rows.r function and add rows with the desired antenna (see adding/removing dummy rows). Then you need to modify the following functions.

### In the All_Combined_Events _function

Antennas are assigned specific UTM's during this section. These are specifically used later when each detection is assigned to a station in the "spatial join" function.

For stationary antennas, you need to assign UTM's based off the site code field

```
# takes out 900 from TAG in WGFP Clean
# also takes out duplicate rows
WGFP_Clean <- WGFP_Clean %>%
  mutate(TAG = ifelse(str_detect(TAG, "^900"), str_sub(TAG, 4,-1), TAG),
         DTY = ifelse(str_detect(DTY, "/"),
                      as.character(mdy(DTY)),
                      DTY)) %>%

  # mutate(TAG = case_when(str_detect(TAG, "^900") ~ str_sub(TAG, 4,-1),
  #                        str_detect(TAG, "!^900") ~ TAG)) %>%
  # assigning UTM's are important because they are plotted later when getting
  mutate(UTM_X =case_when(SCD == "RB1" | SCD == "RB2" ~ "412489",
                          SCD == "HP3" | SCD == "HP4" ~ "414375",
                          SCD == "CF5" | SCD == "CF6" ~ "416965",
                          SCD == "CD7" | SCD == "CD8" | SCD == "CD9" | SCD ==
                          SCD == "CU11" | SCD == "CU12" ~ "416806"),
         UTM_Y = case_when(SCD == "RB1" | SCD == "RB2" ~ "4439413",
                           SCD == "HP3" | SCD == "HP4" ~ "4440241",
                           SCD == "CF5" | SCD == "CF6" ~ "4439369",
                           SCD == "CD7" | SCD == "CD8" | SCD == "CD9" | SCD ==
                           SCD == "CU11" | SCD == "CU12" ~ "4439489")) %>%
  distinct()

# biomark cleaning, getting dates into uniform format,
```

For biomark antennas, you need to add in the Reader.ID field, based off what the antenna is called in the field. Since Windy gap biomark has been called A1 or B1 at times in the past, so this changes all those entries to B3.

That last line also removes marker tags. There's another line later that also removes all biomark tags that start with anything other than 900, which would include marker tags.

```
# biomark cleaning, getting dates into uniform format,
biomark2 <- Biomark %>%
  mutate(TAG = str_replace(DEC.Tag.ID, "\\.", ""),
         Reader.ID = case_when(Reader.ID == "A1" | Reader.ID == "B1" ~ "B3",
                               Reader.ID == "A2" | Reader.ID == "B2" ~ "B4",
                               Reader.ID == "A3" ~ "B5",
                               Reader.ID == "A4" ~ "B6",
                               TRUE ~ Reader.ID),
         #make a column for Scan>Date if parentheses are detected in the strin
         # and we want to convert it to YYYYMMDD format. elsewise, leave it as
         Scan.Date = ifelse(str_detect(Scan.Date, "/"),
                            as.character(mdy(Scan.Date)),
                            Scan.Date)
           ) %>%
  filter(!TAG %in% c("900230000102751", "900226001581072", "999000000007586",
```

This is the section that assigns UTM's to biomark antennas, after making the correct reader.id

```
# b5 is river run
# b6 is fraser river canyon
mutate(UTM_X =case_when(Reader.ID == "B3" ~ "416026",
                        Reader.ID == "B4" ~ "420728",
                        Reader.ID == "B5" ~ "419210",
                        Reader.ID == "B6" ~ "424543"),
       UTM_Y = case_when(Reader.ID == "B3" ~ "4440196",
                         Reader.ID == "B4" ~ "4437221",
                         Reader.ID == "B5" ~ "4439961",
                         Reader.ID == "B6" ~ "4435559")) %>%
```

PrepareforStatesMovementsandSummary_function

The River is assigned to the new antennas in this part.

```
mutate(
  #River also needs to be assigned for new detections
  River = case_when(
    (Event %in% c("RB1", "RB2")) ~ "Colorado River", # there is no is.na her
    (Event %in% c("HP3", "HP4")) ~ "Colorado River",
    (Event %in% c("CF5", "CF6")) ~ "Colorado River",
    (Event %in% c("CD7", "CD8", "CD9", "CD10", "CU11", "CU12")) ~ "Connectiv
    (Event %in% c("B3", "B5")) ~ "Colorado River",
    (Event %in% c("B4", "B6")) ~ "Fraser River",
    TRUE ~ River
  ),
```

The type of movement is condensed to a "detection type" field. This is helpful in the movements map where it doesn't necessarily matter which antenna specifically is hit.

```
group_by(TAG) %>%
mutate(
  #this is used in getting states; not sure if needed anymore
  previous_event = lag(Event, order_by = Datetime),
  #this part is used in movemnets map
  det_type = case_when(str_detect(Event, "RB1|RB2") ~ "Red Barn Stationary A
                       str_detect(Event, "HP3|HP4") ~ "Hitching Post Statior
                       str_detect(Event, "CF5|CF6") ~ "Confluence Stationary
                       str_detect(Event, "CD7|CD8|CD9|CD10") ~ "Connectivity
                       str_detect(Event, "CU11|CU12") ~ "Connectivity Channe
                       str_detect(Event, "B3") ~ "Windy Gap Dam Biomark Ante
                       str_detect(Event, "B4") ~ "Kaibab Park Biomark Antenr
                       str_detect(Event, "B5") ~ "River Run Biomark Antenna'
                       str_detect(Event, "B6") ~ "Fraser River Canyon Biomar
                       str_detect(Event, "M1|M2") ~ "Mobile Run",
                       Event == "Recapture" ~ "Recapture",
                       TRUE ~ Event),
  above_below = case_when(
    ET_STATION >= 8330 ~ "Above the Dam",
    ET_STATION < 8330 ~ "Below the Dam"
  )

) %>%
```

### In the Ind_tag_hist_summary_wide_function:

You need to assign column names for the new antennas quite a bit in this function, starting with this bit of code. It should be pretty intuitive what to add/where to add if you examine the code.

```
ENC_ALL <- all_enc12 %>%
  rename(RB1_n = RB1,
         RB2_n = RB2,
         HP3_n = HP3,
         HP4_n = HP4,
         CF5_n = CF5,
         CF6_n = CF6,
         CD7_n = CD7,
         CD8_n = CD8,
         CD9_n = CD9,
         CD10_n = CD10,
         CU11_n = CU11,
         CU12_n = CU12,
         M1_n = M1,
         M2_n = M2,
         B3_n = B3,
         B4_n = B4,
         B5_n = B5,
         B6_n = B6,
         Recap_n = Recapture
```

This bit is the tricky part: Where sums of rows are totaled and it's based off the total number of columns in the dataframe. Add the desired columns here, but be sure to check your work to make sure you're counting the columns right and getting the results you expect.

```
ENC_Release2 <- ENC_Release1 %>%
  #counts number of TRUE across specified rows. -SG
  # need to have parentheses (totalcols-1) that's why i was getting bad numbers on biomark T/F initially
  #added 8 new columns for new antennas
  mutate(
    TotalEncounters = rowSums(ENC_Release1[(totalcols-18):totalcols] == TRUE),

    TotalAntennas1 = rowSums(ENC_Release1[(totalcols-18):(totalcols-1)] == TRUE),
    TotalStationary = rowSums(ENC_Release1[(totalcols-18):(totalcols-7)] == TRUE),
    TotalMobile = rowSums(ENC_Release1[(totalcols-6):(totalcols-5)] == TRUE),
    TotalBiomark = rowSums(ENC_Release1[(totalcols-4):(totalcols-1)] == TRUE),
    TotalRB = rowSums(ENC_Release1[(totalcols-18):(totalcols-17)] == TRUE),
    TotalHP = rowSums(ENC_Release1[(totalcols-16):(totalcols-15)] == TRUE),
    TotalCF = rowSums(ENC_Release1[(totalcols-14):(totalcols-13)] == TRUE),
    TotalCD = rowSums(ENC_Release1[(totalcols-12):(totalcols-9)] == TRUE),
    TotalCU = rowSums(ENC_Release1[(totalcols-8):(totalcols-7)] == TRUE),
```

# Navigating the App

## About/How to Use Tab

This tab is a concise version of this document. NEED TO EDIT

## Individual Datasets Tab

This Tab shows the following tables and are all controlled by the same date filter in the sidebar. Click Render table to display all tables.

- Stationary Clean: A clean dataset of all Stationary Data (WGFP_Clean) with 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
- Biomark: All Biomark combined detections from Kaibab Park and Windy Gap including Marker Tags. In the future, it might be a good idea to separate out Marker Tags from this file as it grows.
- Mobile: All Detections from all Mobile Runs
- Recaptures: All Recaptured fish

- Release: All release data
- 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

## Encounter Histories Tab

- Encounter Release History Summary Wide: shows ENC_Release_wide_summary Dataframe from enc_hist_summary_wide_function.
    - 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 weather 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 and the highest number it could be right now is 11.
        - Through_dam tells weather a fish has gone through the dam 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
        - Went_above_dam_no_channel and subsequent columns are derived from the States function, which tells if a fish went below/above the dam using the channel or not. See the Get-states_function 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.
- All Events and Plot: Shows and plots All_Events Dataframe from All_combined_events_function. These are raw detections. 230000142723 has over 120k detections on its own at Hitching Post.
    - Most filters are self-explanatory.
    - 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 a way, it filters out most of the redundant detections. 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 result of this filter is downloaded, brought into GIS, and used to make stations with.
        - This is one of the most "useful" files you can get from this app because of how it shaves off the dataset to the most relevant detections
    - Remove Duplicate Tags Filter is helpful in answering questions involving "How many unique fish?". Example: on this tab 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 will plot anything made with the filtered data

## Weekly States Tab

This is where "States" are displayed. The states are defined on a weekly basis as A (below the dam), B (above the Dam), C (in the Connectivity channel), G (ghost tag detection), or P (predated by bird). A weekly "Unique event" is a filter in the sidebar, and is defined as a detection on a new antenna for that week, recapture, or release. A fish could have 6 different events in a week, but if they are all below the dam, it will just get assigned state "A" for that week.

### States Dataframe

Here is part of the history of fish 230000228709, a sizable brown trout.

Show 10 entries Search:

| Date | Datetime | TAG | State | det_type | ReleaseSite | Species | Release_Length | Release_Weight | c |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | All | All | |
| 2020-10-06 | 2020-10-06T13:58:00Z | 230000228709 | C | Release | Hitching Post | LOC | 360 | 505 | |
| 2020-10-07 | 2020-10-07T18:00:28Z | 230000228709 | J | Hitching Post Stationary Antenna | Hitching Post | LOC | 360 | 505 | |
| 2020-10-08 | 2020-10-08T06:51:01Z | 230000228709 | I | Hitching Post Stationary Antenna | Hitching Post | LOC | 360 | 505 | |
| 2020-10-10 | 2020-10-10T23:02:31Z | 230000228709 | J | Hitching Post Stationary Antenna | Hitching Post | LOC | 360 | 505 | |
| 2020-10-11 | 2020-10-11T02:47:46Z | 230000228709 | I | Hitching Post Stationary Antenna | Hitching Post | LOC | 360 | 505 | |
| 2020-10-13 | 2020-10-13T18:37:49Z | 230000228709 | J | Hitching Post Stationary Antenna | Hitching Post | LOC | 360 | 505 | |
| 2020-10-14 | 2020-10-14T02:23:45Z | 230000228709 | I | Hitching Post Stationary Antenna | Hitching Post | LOC | 360 | 505 | |
| 2020-10-15 | 2020-10-15T00:58:31Z | 230000228709 | JI | Hitching Post Stationary Antenna | Hitching Post | LOC | 360 | 505 | |

- The States filter in the sidebar can help determine "magic" fish that may have succumbed to avian predation, since they hit multiple weird states in a day. For example, 230000228444 has one day where it transitions at confluence, then red barn, then confluence again. All in 1 day.
- Det_type column is useful when there is only 1 state of the day, but doesn't accurately capture everything if there are multiple states on a day
- Datetime column doesn't really mean anything in this Dataframe either
- C_number_of_detections is the total number of raw detections for the day. Might be 22k, might be 1.
- Daily_unique_events is the total number of events that happened in a day. This can also be helpful in finding "magic" fish.

### States and Days Wide

This is the same data displayed in States_dataframe, but in wide format where each unique tag gets a row and each column is a day since the beginning of the study. This may have a bit more rows than the release file because there are some fish in there without release info.

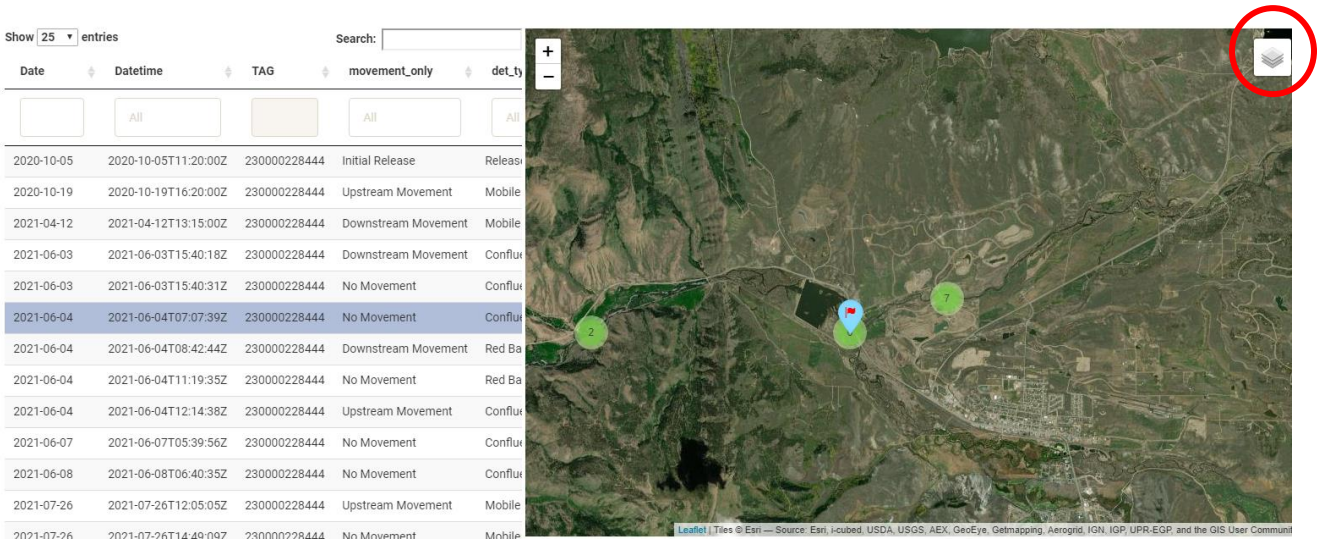
### Unknown States

This is where events that were unable to be captured by states are displayed. This hopefully should be pretty small...it's filled with tags with detections before official 'Release' such as in in May 2021 and tags without release info. Mainly tags without an idea where they came from. But if a Tag shows up where release info is known, might have to go into the get_states_function.R code to make another case_when entry to account for the new state. All in all, this is a check to see how well the get_states_function is working, and also to see if fish without release info have gotten by the other filters.

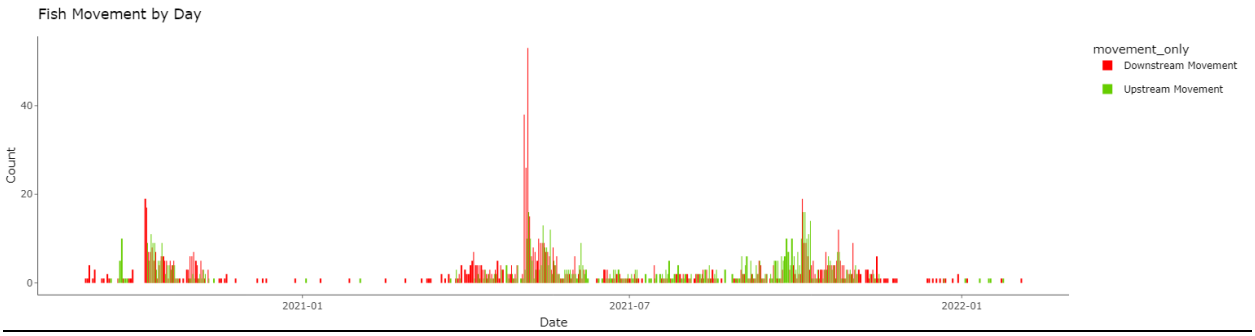
## Daily Movements Map, Plot, and Data

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

- 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 the event column in the table, and vice versa.
- Filters in the sidebar control what is displayed on the map
- Layer control is available in top right




- The plot is also controlled by the sidebar filters.




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 a event on the map, click on it, copy the tag number, and plug that in.

- The animation tab lets you render an amination of filtered data from the left sidebar. Click "render" in the left sidebar before clicking the "render animation" button. There is a progress bar in the Rstudio terminal that shows if you minimize the map. It automatically saves a .gif in the parent directory called WindyGapFishMovements.gif

## "Movements" Defined

Movements are inferred by detections of the same tag on different antennas 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 12930 m on the Upper Colorado River, where the mobile run begins, and to 16170 m (10120 m at confluence + 6050 m from confluence to Kaibab Park) on the Fraser River.

These stations are joined by coordinates to a dataset of all relevant detections for each unique UTM, 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 the first and last detections of the day and one detection in between. Wacko fish that hit multiple different antenna in between the first and last

detection of the day won't display all of these detections. These CAN be seen in the states and All Events dataframes though.

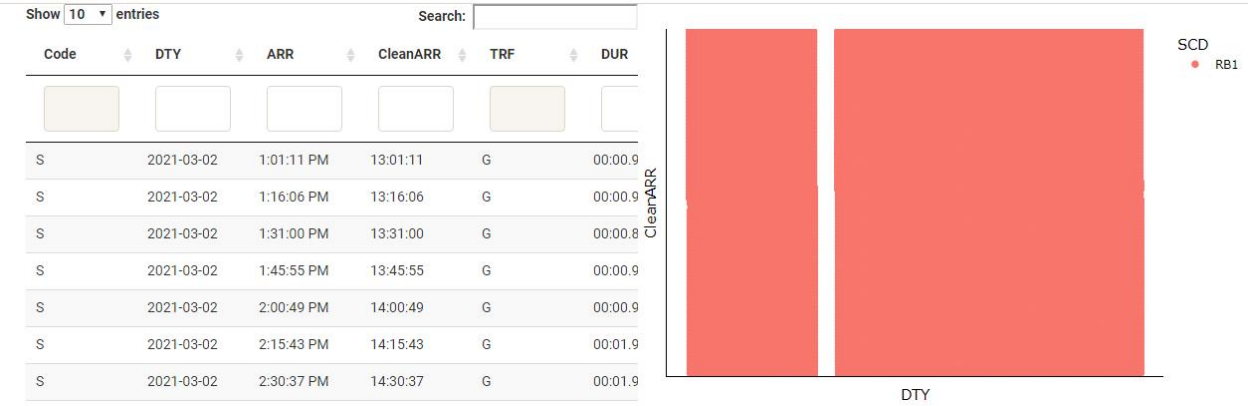
Once there is a dataset of daily movements, the distance moved between days is calculated for each fish when a new detection is recorded, calculated by taking the difference between stations. An upstream movement is a positive difference (IE new detection station = 8000, previous detection station 6000, so 8000-6000 > 0; upstream movement). A downstream movement is a negative difference between current station and previous station (new station = 6000 m, old station is 8000 m, -2000 <0; negative movement). No movement is if there is no difference.

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 the Encounter Histories created in the Ind_tag_enc_hist_summary_wide function.

## QAQC Tab

Used to help verify data.

- Marker Tags:
    - Marker Tag Data set from All_Combined_events_function
    - Shows if there was a gap in the marker tags
    - In this case, there was a gap: closer examination shows there were not detections for the month of January 2021




    - 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.
    - There is currently no functionality for Biomark marker tags but wouldn't be too hard to get it in the future
    - If the ARR column is brought in incorrectly (which can sometimes happen if the stationary ARR column gets corrupted) then there will be an abundance of detections in the 12:00-13:00 range.
- Release and Recap Lengths/Weights
    - Plot to help ensure that there were no length/weight typos in the release and recapture files.
- 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"

## 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.

```
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 then the rows are removed in the data so this dummy data isn't used. 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, ET STATION, SI
```

- In the reactive for all events in app.r:

```
919
920       }
921     #### filter dummy row
922     all_events_filtered <- all_events_filtered %>%
923       filter(!TAG %in% c("230000999999"))
924
925
926       return(all_events_filtered)
927   }) #end of ENC data list eventReactive
928
```

- In get_states_function:

```
30    #daily_unique_events = length(unique(Event))
31    states1 <- wghost_av %>%
32      filter(!TAG %in% c('230000999999')) %>%
33      mutate(
```

- After the functions are ran in the following individual datasets in app.r:

```
151 ### taking dummy tag out
152
153 Biomark <- Biomark %>%
154   filter(!DEC.Tag.ID %in% c("900.230000999999"))
155 Release_05 <- Release_05 %>%
156   filter(!TagID %in% c("230000999999"))
157 Stationary <- Stationary %>%
158   filter(!TAG %in% c("900_230000999999"))
159
160 # Define UI for application that draws a histogram
```

- 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
```

To put the dummy rows back in, comment out the code that filters out the dummy rows above. Once there is actual data using cd7-cu12 antennas, you can delete these lines.

## Variables.R

There is a variables.R folder that is meant to put important info in one place.

## General Notes

- All plots are interactive. 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.
- Datatables are also interactive. Click the column titles of any column to sort in ascending/descending order. You can use the Search bar on the right to search/filter for any specific values. Some 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
- UTM's for stationary and biomark stations are assigned within the app
- Keep a folder of CSV backups in case the time column goes wonky
- Currently there is a Release site called "Shefiff Ranch Middle Field"; if this typo is corrected in Release File, it will have to be changed in the get_states_function code as well. Otherwise all fish will show up in the "unknown states" tab.
- This information is in the Combining Data How To as well, but when entering Biomark data, ensure the Reader ID column is correct in the original excel files. While the data will read in as B1 for Windy Gap and B2 for Kaibab Park, they need to change to A1 and A2 respectively (just change B to A) in order to read in properly in the Get_Movements_Function. If they are not adjusted, you will get this error message: *Error in convUL(movement_table_notrans, km = FALSE, southern = NULL) : Invalid X/Y data 'xydata'. One or more columns (where NAs are not allowed) contains NAs. Columns that cannot contain NAs: X, Y.*
- The above error message has also shown up for me when loading in data where the full TAG ID has been deleted/cut off. Double check that the "DEC TAG ID" column has actual full values, if not, they were lost somewhere along the way and you'll likely have to re-combine files. Still unsure why it sometimes cuts them off, but I've found that changing the column format to "Number" with 12 decimal places before making edits fixes the problem.

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

## Very Doable tasks for the future

- Expand on the "variables.R" folder. My idea is to put important data in this about gps coordinates of antennas, antenna names, and other things that come up. Then we don't have to go on a goose chase hunting down the variables in the code when we just change it in the variables
- Use the package "minicharts" to make fun bar graph animations on the movements map. This would be a good way to visualize the quantity of detections over time.
- Streamline the data cleaning process so that the data, especially Stationary data, is clean when it is read in to the app. This would help with the time it takes to load the app, since it would make most of the All combined events function obsolete. My idea is to take out the pieces of that function and clean each individual dataset, save as a csv's, then modify it so the AllCombinedEvents function will only basically combine all the DF's.
- Have the data be read in from a database instead of CSV's. Maybe a good step would actually be to use .rds files instead of csv's at the least. Sometimes the time columns of biomark and stationary get corrupted and I have no idea why, and I can't help but think this wouldn't happen if they were being read in from a DB instead of CSV's.
- Convert and use spatial data as sf objects, not shapefiles. Data wrangling is a bit more intuitive and the spatial join function could be cleaned up because you can more easily convert from utm's with st_transform
- Change all the plots to the rainbow trout color pallete
- Make all "upstream/downstream movements" colors on the animation map the same as the ones on the movements map