library(tidyverse)  
library(lubridate)  
library(here)  
library(geosphere)  
library(data.table)  
  
  
options(scipen = 999)

# First load data and do some minor management.

## Drone data

The drone telemetry data has 1-11 locations per second. We only need 1 location per second, so then calculate the mean position for each second.

drone\_tele\_tags <- readRDS(here("data/combined\_drone\_telemetry")) %>%   
 select(ts = date.time, drone, latitude, longitude, heading, mfg, file, flight.num, flight.height)  
  
drone\_tele\_tags1sec<- drone\_tele\_tags %>%   
 mutate(ts = as.character(ts)) %>%   
 group\_by(drone, ts, mfg) %>%   
 summarise(latitude = mean(latitude),  
 longitude = mean(longitude),  
 heading = mean(heading)) %>%   
 ungroup() %>%   
 mutate(date = as.Date(ts),  
 flight.loc = case\_when(ts < "2021-09-14 12:00:00" ~ "Cypress Grove",  
 ts > "2021-09-14 12:00:00" & date == "2021-09-14" ~ "Walker Creek",  
 date == "2021-09-15" ~ "Toms Point"))  
head(drone\_tele\_tags1sec)

## # A tibble: 6 x 8  
## drone ts mfg latitude longitude heading date flight.loc   
## <chr> <chr> <chr> <dbl> <dbl> <dbl> <date> <chr>   
## 1 green 2021-09-14 09:~ CTT 38.2 -123. -101. 2021-09-14 Cypress Gro~  
## 2 green 2021-09-14 09:~ CTT 38.2 -123. -101. 2021-09-14 Cypress Gro~  
## 3 green 2021-09-14 09:~ CTT 38.2 -123. -101. 2021-09-14 Cypress Gro~  
## 4 green 2021-09-14 09:~ CTT 38.2 -123. -100. 2021-09-14 Cypress Gro~  
## 5 green 2021-09-14 09:~ CTT 38.2 -123. -100. 2021-09-14 Cypress Gro~  
## 6 green 2021-09-14 09:~ CTT 38.2 -123. -100. 2021-09-14 Cypress Gro~

# Motus data

all\_tags\_tower <- readRDS(here("data/combined\_by\_tower")) %>%   
 select(ts, sig, motusTagID, recv, antBearing, date, mfg)

Interpolate missed pings.

For each flight X antenna combo, one or more pings might have been missed: 1) after the flight began but before the first detected ping, 2) anywhere between the first and last detected ping, 3) after the last detected ping but before the flight ended, or 4) an entire flight X antenna combo might have no detected pings There is slightly different logic for interpolating these different potential missed pings; they must be interpolated separately then joined

First generate df representing start and end times for each drone flight duplicated for each antenna.  
ctt: 8 antennae \* 12 flights = total 96 expected flight X antenna combinations.

drone\_flight\_times\_antennae <- drone\_tele\_tags %>%   
 mutate(date = as.Date(ts)) %>%   
 group\_by(drone, date, flight.num, mfg, flight.height) %>%   
 summarise(start.time = min(ts),  
 end.time = max(ts)) %>%   
 ungroup() %>%   
 full\_join(distinct(all\_tags\_tower, antBearing, mfg)) %>%   
 filter(!is.na(antBearing)) %>%   
 mutate(flight.length = difftime(end.time, start.time, units = "secs"),  
 flight.length = as.numeric(flight.length),  
 expected.pings = ifelse(mfg == "CTT", flight.length/14.8, flight.length/10.7))  
  
# save this for generating summary table for report  
saveRDS(drone\_flight\_times\_antennae, here("data/drone\_flight\_times\_antennae"))

Combine flight X antenna info with detections for each antenna

flights\_antennae\_detections <- all\_tags\_tower %>%   
 full\_join(., drone\_flight\_times\_antennae) %>%   
 filter(data.table::between(ts, start.time, end.time)) %>% # just tower detections during drone flights  
 group\_by(date, mfg, antBearing, drone, flight.num) %>%   
 summarise(detected.pings = n())%>%   
 ungroup() %>%   
 full\_join(., drone\_flight\_times\_antennae) %>%   
 mutate(detected.pings = ifelse(is.na(detected.pings), 0, detected.pings)) %>%   
 full\_join(distinct(all\_tags\_tower, recv, antBearing)) %>%   
 arrange(antBearing, date, drone, flight.num)

Identify the first and last detected ping for each flight X antenna combo.

Note: using data.table::between is critical here for filtering to just the signals that were detected during one of the drone flights. dplyr::between is apparently not vectorized on the left and right arguments, so it doesn’t work correctly across the entire data frame.

flight\_times\_antennae\_first\_last\_ping <- all\_tags\_tower %>%   
 full\_join(., drone\_flight\_times\_antennae) %>%  
 filter(data.table::between(ts, start.time, end.time)) %>%   
 mutate(ts = ifelse(is.na(ts), start.time, ts),  
 ts = as.POSIXct(ts, origin = "1970-01-01")) %>%   
 group\_by(drone, date, flight.num, mfg, antBearing) %>%   
 summarise(first.ping.ts = min(ts),  
 last.ping.ts = max(ts)) %>%   
 ungroup()

# Now on to actual interpolation

First the missed leading and trailing pings

missed\_lead\_trail <- full\_join(flights\_antennae\_detections, flight\_times\_antennae\_first\_last\_ping) %>%   
 mutate(exp.ping.int = ifelse(mfg == "CTT", 14.8, 10.7),  
 missed.lead = as.numeric(difftime(first.ping.ts, start.time, units = "secs"))/exp.ping.int,  
 missed.lead = floor(missed.lead),  
 missed.trail = as.numeric(difftime(end.time, last.ping.ts, units = "secs"))/exp.ping.int,  
 missed.trail = floor(missed.trail))

For interpolated lead pings, want to count backward from the first detected ping but don’t want to duplicate the first detected ping.

interp\_lead <- missed\_lead\_trail %>%   
 filter(detected.pings > 0, missed.lead > 0) %>%   
 select(date, mfg, antBearing, drone, flight.num, start.time, first.ping.ts, missed.lead, exp.ping.int) %>%  
 group\_by(date, mfg, antBearing, drone, flight.num, start.time, first.ping.ts) %>%  
 complete(missed.lead = full\_seq(1:missed.lead, 1)) %>%   
 mutate(total.missed.lead = max(missed.lead)) %>%   
 ungroup() %>%   
 mutate(exp.ping.int = ifelse(mfg == "CTT", 14.8, 10.7),  
 ts = first.ping.ts - ((total.missed.lead - (missed.lead - 1)) \* exp.ping.int))  
  
interp\_lead <- full\_join(flights\_antennae\_detections, flight\_times\_antennae\_first\_last\_ping) %>%   
 mutate(exp.ping.int = ifelse(mfg == "CTT", 14.8, 10.7),  
 missed.lead = as.numeric(difftime(first.ping.ts, start.time, units = "secs"))/exp.ping.int,  
 missed.lead = floor(missed.lead)) %>%   
 filter(detected.pings > 0, missed.lead > 0) %>%   
 select(date, mfg, antBearing, drone, flight.num, start.time, first.ping.ts, missed.lead, exp.ping.int) %>%  
 group\_by(date, mfg, antBearing, drone, flight.num, start.time, first.ping.ts) %>%  
 complete(missed.lead = full\_seq(1:missed.lead, 1)) %>%   
 mutate(total.missed.lead = max(missed.lead)) %>%   
 ungroup() %>%   
 mutate(exp.ping.int = ifelse(mfg == "CTT", 14.8, 10.7),  
 ts = first.ping.ts - ((total.missed.lead - (missed.lead - 1)) \* exp.ping.int))

For interpolated trailing pings, want to count forward from the last detected ping but don’t want to duplicate the last detected ping.

interp\_trail <- full\_join(flights\_antennae\_detections, flight\_times\_antennae\_first\_last\_ping) %>%   
 mutate(exp.ping.int = ifelse(mfg == "CTT", 14.8, 10.7),  
 missed.trail = as.numeric(difftime(end.time, last.ping.ts, units = "secs"))/exp.ping.int,  
 missed.trail = floor(missed.trail)) %>%   
 filter(detected.pings > 0, missed.trail > 0) %>%   
 select(date, mfg, antBearing, drone, flight.num, end.time, last.ping.ts, missed.trail) %>%  
 group\_by(date, mfg, antBearing, drone, flight.num, end.time, last.ping.ts) %>%  
 complete(missed.trail = full\_seq(1:missed.trail, 1)) %>%   
 mutate(total.missed.trail = max(missed.trail)) %>%   
 ungroup() %>%   
 mutate(exp.ping.int = ifelse(mfg == "CTT", 14.8, 10.7),  
 ts = last.ping.ts + (missed.trail \* exp.ping.int))

Next, missed pings between first and last detected ping

missed\_middle <- all\_tags\_tower %>%   
 select(mfg, antBearing, date, ts) %>%   
 full\_join(flights\_antennae\_detections) %>%   
 filter(data.table::between(ts, start.time, end.time)) %>%   
 arrange(mfg, antBearing, ts) %>%   
 group\_by(date, mfg, antBearing, drone, flight.num) %>%   
 mutate(next.ts = lead(ts)) %>%  
 ungroup() %>%   
 mutate(exp.ping.int = ifelse(mfg == "CTT", 14.8, 10.7),  
 ping.int = as.numeric(difftime(next.ts, ts, units = "secs")),  
 ping.int = round(ping.int, 1),  
 missed.middle = ping.int/exp.ping.int,  
 missed.middle = floor(missed.middle))  
   
interp\_middle <- missed\_middle %>%   
 filter(!is.na(missed.middle)) %>%   
 select(date, ts, next.ts, mfg, antBearing, drone, flight.num, missed.middle) %>%  
 group\_by(date, ts, next.ts, mfg, antBearing, drone, flight.num) %>%  
 complete(missed.middle = full\_seq(1:missed.middle, 1)) %>%   
 mutate(total.missed.middle = max(missed.middle)) %>%   
 ungroup() %>%   
 mutate(exp.ping.int = ifelse(mfg == "CTT", 14.8, 10.7),  
 interp.ts = ts + (exp.ping.int \* missed.middle),  
 check.diff = difftime(next.ts, interp.ts, units = "secs"),  
 check.diff = as.numeric(check.diff),  
 drop = ifelse(missed.middle == total.missed.middle & check.diff < (0.7 \* exp.ping.int), TRUE, FALSE)) %>%   
 select(contains("ts"), everything()) %>%   
 filter(drop == FALSE) %>%   
 arrange(date, drone, flight.num, mfg, antBearing, ts)

Finally, flight X antenna combos with no detections.

interp\_flight\_ant <- flights\_antennae\_detections %>%   
 filter(detected.pings == 0) %>%   
 select(date, start.time, end.time, mfg, antBearing, drone, flight.num, expected.pings) %>%  
 mutate(expected.pings = round(expected.pings, 0)) %>%   
 group\_by(date, start.time, end.time, mfg, antBearing, drone, flight.num) %>%  
 complete(expected.pings = full\_seq(0:expected.pings, 1)) %>%   
 mutate(total.expected = max(expected.pings)) %>%   
 ungroup() %>%   
 mutate(exp.ping.int = ifelse(mfg == "CTT", 14.8, 10.7),  
 ts = start.time + (exp.ping.int \* expected.pings),  
 keep = ts <= end.time)

Join all those with the records of actually detected pings.

all\_tags\_tower\_interp <- all\_tags\_tower %>%   
 full\_join(flights\_antennae\_detections) %>%   
 filter(data.table::between(ts, start.time, end.time)) %>%   
 select(date, ts, drone, flight.num, mfg, antBearing, sig) %>%   
 full\_join(select(interp\_lead, date, ts, drone, flight.num, mfg, antBearing)) %>%   
 full\_join(select(interp\_trail, date, ts, drone, flight.num, mfg, antBearing)) %>%   
 full\_join(select(interp\_middle, date, ts = interp.ts, drone, flight.num, mfg, antBearing)) %>%   
 full\_join(select(interp\_flight\_ant, date, ts, drone, flight.num, mfg, antBearing)) %>%   
 full\_join(select(flights\_antennae\_detections, date, drone, flight.num, mfg, recv, antBearing, start.time, end.time)) %>%   
 distinct() %>%   
 arrange(date, drone, flight.num, mfg, antBearing, ts) %>%   
 group\_by(date, drone, flight.num, mfg, antBearing) %>%   
 mutate(ts.diff = difftime(ts, lag(ts), units = "secs"),  
 ts.diff = round(ts.diff),  
 exp.ping.int = ifelse(mfg == "CTT", 14.8, 10.7)) %>%   
 ungroup()

We should have close to the same number of pings for each tower within each flight; but since we are interpolating theoretical pings, it is expected that we won’t get exactly the same number for each tower.

all\_tags\_tower\_interp %>%   
 filter(mfg == "CTT") %>%   
 count(date, drone, flight.num, mfg, antBearing) %>%   
 pivot\_wider(id\_cols = c(date, drone, mfg, flight.num), names\_from = "antBearing", values\_from = n) %>% view()  
  
# all\_tags\_tower\_interp %>%   
# group\_by(date, drone, flight.num, mfg, antBearing) %>%   
# summarise(mean.ts.diff = mean(as.numeric(ts.diff)),  
# num.ping = n()) %>%   
# view()

Join tag detections with drone telemetry a second time to attribute the interpolated tag pings with a drone location.

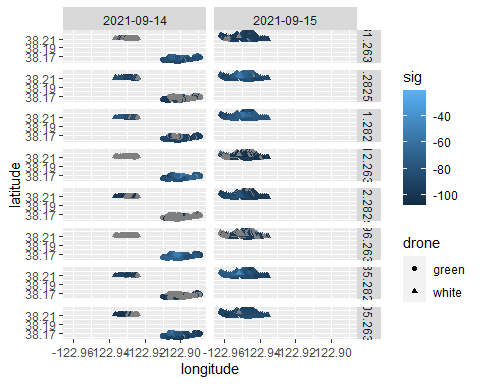
df <- drone\_tele\_tags1sec %>%   
 inner\_join(., all\_tags\_tower\_interp %>%   
 mutate(ts = as.character(ts))) %>%   
 full\_join(., readRDS(here("data/tower\_coords"))) %>%   
 filter(!is.na(latitude)) %>%   
 arrange(drone, antBearing, ts)

One more check to make sure we have similar numbers of signals for each tower X flight:

df %>%   
 filter(mfg == "CTT") %>%   
 count(date, drone, flight.num, mfg, antBearing) %>%   
 pivot\_wider(id\_cols = c(date, drone, mfg, flight.num), names\_from = "antBearing", values\_from = n) %>%   
 view()

And a quick spatial plot to make sure the paths look reasonable:

df %>%   
 mutate(ts = as.POSIXct(ts)) %>%  
 filter(mfg == "CTT") %>%   
 ggplot() +  
 geom\_point(aes(x = longitude, y = latitude, color = sig, shape = drone)) +  
 facet\_grid(antBearing~as.Date(ts))



# Calculate distances and angles between towers/antennae and drones.

First need matrices of UAV locations and tower locations.

tag.loc <- df %>%   
 select(longitude, latitude) %>%   
 data.matrix()  
  
tower.loc <- df %>%   
 select(tower.longitude, tower.latitude) %>%   
 data.matrix()

Calculate distances:

ant.tag.dist <- distHaversine(tower.loc, tag.loc) %>%   
 data.frame() %>%   
 rename(ant.tag.distance = 1)

Calculate bearings:

df\_bearings <- bearing(p1 = tower.loc, p2 = tag.loc) %>%   
 data.frame() %>%   
 rename(tower.tag.bearing = 1) %>%  
 # change scale from -180 to 180 to 0 to 360  
 mutate(tower.tag.bearing = ifelse(tower.tag.bearing < 0, (180 + tower.tag.bearing) + 180, tower.tag.bearing))

Combine distances and bearings.

df\_bearings\_dist <- df\_bearings %>%   
 cbind(df) %>%   
 cbind(ant.tag.dist)

Calculate how far off the antenna axis the tag was. Need to convert bearing off antenna axis to 0-180 scale; assuming effect of bearing is equal left and right.

df\_bearings\_dist <- df\_bearings\_dist %>%   
 filter(!is.na(antBearing), !is.na(name)) %>%   
 mutate(tag.ant.bearing.dif = antBearing - tower.tag.bearing,  
 tag.ant.bearing.dif = case\_when(tag.ant.bearing.dif < -180 ~ 180 + (tag.ant.bearing.dif + 180),  
 tag.ant.bearing.dif > 180 ~ -180 + (tag.ant.bearing.dif - 180),  
 TRUE ~ tag.ant.bearing.dif)) %>%   
 select(-contains("bearing"), everything(), contains("bearing")) %>%   
 mutate(tag.ant.bearing.dif = abs(tag.ant.bearing.dif),  
 wc = ifelse(flight.loc == "Walker Creek", "Walker Creek flights", "Other flight locations"))  
  
  
saveRDS(df\_bearings\_dist, here("data/all\_pings\_distance\_bearing"))