What is an elevational range?

The elevational distribution of species have long fascinated scientists by isolating specific niche axes from geography, shaping global biogeographic patterns, and highlighting the link between natural selection, trait values, and species ranges. In the era of climate change, this interest has burgeoned, as studies of global warming mediated range-shifts have focused on the predicted upslope movement of species and populations. However, this body of work has yielded equivocal empirical results.

In this study we use citizen science data (observations of Yellow-eyed Juncos—*Junco phaeonotus*, hereafter YEJU, in the Madrean Sky Islands of Arizona, USA) to highlight four important considerations:

1) Elevational ranges are estimated, not observed directly (i.e. we treat them as statistical objects);
2) Elevational ranges include variation in abundance; i.e., they are multidimensional;
3) Elevational ranges are spatially and temporally dynamic, and that variation is meaningful;
4) Range shift inferences are sensitive to elevational range concepts and definitions;
5) Elevational ranges are scale dependent phenomena shaped by scale dependent phenomena.

The following notebook will prepare our data for downstream analyses and figures. It requires the following libraries:

```
library(auk)
library(lubridate)
library(sf)
library(stars)
library(terra)
library(tidyterra)
library(tidyverse)
library(ggplot2)
library(maps)
library(dggridR)
library(scam)
library(PresenceAbsence)
library(stars)
library(elevatr)
library(raster)
library(cowplot)
library(ggdist)
library(ranger)
library(fGarch)
library(metR)
library(flextable)
```

Preparation

We'll start by setting our working directory and loading eBird data: all YEJU observations from Arizona, and the associated eBird sampling effort data released in April 2023.

```
# set wd
setwd("~/Dropbox/what_is_elev_range/")

# resolve namespace conflicts
select <- dplyr::select

# set data directory
data_dir <- "data"
if (!dir.exists(data_dir)) {
  dir.create(data_dir)
}

# load ebd and file sampling
ebd <- auk_ebd("data/ebd_US-AZ_yeejun_relApr-2023.txt",
               file_sampling = "data/ebd_sampling_relApr-2023.txt")
```

Next, we'll apply a first round of filters to the full dataset, selecting the months of June / July / August and making sure we restrict the sampling dataset to the state of Arizona as well. We'll then create a zero-filled dataset, i.e. a dataset of all checklists with inferred absences as well:

```
# define filters
ebd_filters <- ebd %>%
  # sierra madre bird conservation region
  auk_bcr(bcr = 34) %>%
  # restrict to the standard traveling and stationary count protocols
  auk_protocol(protocol = c("Stationary", "Traveling")) %>%
  # may / june / july, * gets data from any year
  auk_date(date = c("*-05-01", "*-07-31")) %>%
  auk_state("US-AZ") %>%
  auk_complete()

# view filters
ebd_filters
```

```
## Input
##   EBD: /Users/k14m234/Dropbox/what_is_elev_range/data/ebd_US-AZ_yeejun_relApr-2023.txt
##   Sampling events: /Users/k14m234/Dropbox/what_is_elev_range/data/ebd_sampling_relApr-2023.txt
##
## Output
##   Filters not executed
##
## Filters
##   Species: all
##   Countries: all
##   States: US-AZ
##   Counties: all
##   BCRs: 34
##   Bounding box: full extent
```

```
##   Years: all
##   Date: *-05-01 - *-07-31
##   Start time: all
##   Last edited date: all
##   Protocol: Stationary, Traveling
##   Project code: all
##   Duration: all
##   Distance travelled: all
##   Records with breeding codes only: no
##   Exotic Codes: all
##   Complete checklists only: yes
```

```r
# assign filenames
f_ebd <- file.path(data_dir, "ebd_yeju.txt")
f_sampling <- file.path(data_dir, "ebd_checklists.txt")

# run filters if the files don't already exist
if (!file.exists(f_ebd)) {
  auk_filter(ebd_filters, file = f_ebd, file_sampling = f_sampling)
}

# generate "zero-filled" dataset
ebd_zf <- auk_zerofill(f_ebd, f_sampling, collapse = TRUE)
```

Next we'll tidy up dates and times, and perform additional filtering:

```r
# function to convert time observation to hours since midnight
time_to_decimal <- function(x) {
  x <- hms(x, quiet = TRUE)
  hour(x) + minute(x) / 60 + second(x) / 3600
}

# clean up variables
ebd_zf <- ebd_zf %>%
  mutate(
    # convert X to NA
    observation_count = if_else(observation_count == "X",
                               NA_character_, observation_count),
    observation_count = as.integer(observation_count),
    # effort_distance_km to 0 for non-travelling counts
    effort_distance_km = if_else(protocol_type != "Traveling",
                                 0, effort_distance_km),
    # convert time to decimal hours since midnight
    time_observations_started = time_to_decimal(time_observations_started),
    # split date into year and day of year
    year = year(observation_date),
    day_of_year = yday(observation_date)
  )

# additional filtering for short distance and duration, <=5 birders, only the last 15 years of data
ebd_zf_filtered <- ebd_zf %>%
  filter(
    # effort filters
    duration_minutes <= 5 * 60,
```

```
    effort_distance_km <= 2,
    # last 15 years of data
    year >= 2007,
    # 10 or fewer observers
    number_observers <= 5)

# select only useful columns
ebird <- ebd_zf_filtered %>%
  select(checklist_id, observer_id, sampling_event_identifier,
         scientific_name,
         observation_count, species_observed,
         state_code, locality_id, latitude, longitude,
         protocol_type, all_species_reported,
         observation_date, year, day_of_year,
         time_observations_started,
         duration_minutes, effort_distance_km,
         number_observers)
write_csv(ebird, "data/ebd_yeju_zf.csv", na = "")
```

Next, we'll subsample checklists to include 1 for each square half-kilometer per week:

```
# generate hexagonal grid with ~ 1 km between cells
dggs <- dgconstruct(spacing = 1)
```

```
## Resolution: 16, Area (km^2): 1.18491167242236, Spacing (km): 1.07508800966481, CLS (km): 1.228281889
```

```
# get hexagonal cell id and week number for each checklist
checklist_cell <- ebird %>%
  mutate(cell = dgGEO_to_SEQNUM(dggs, longitude, latitude)$seqnum,
         year = year(observation_date),
         week = week(observation_date))

# sample one checklist per grid cell per week
# sample detection/non-detection independently
ebird_ss <- checklist_cell %>%
  group_by(species_observed, year, week, cell) %>%
  sample_n(size = 1) %>%
  ungroup()
```

Here are some stats describing the original and the subsampled dataset:

```
# original data
nrow(ebird)
```

```
## [1] 106364
```

```
count(ebird, species_observed) %>%
  mutate(percent = n / sum(n))
```

```
## # A tibble: 2 x 3
##   species_observed     n percent
```

```
##   <lgl>            <int>   <dbl>
## 1 FALSE            96963  0.912
## 2 TRUE              9401  0.0884
```

```
nrow(ebird_ss)
```

```
## [1] 44207
```

```
count(ebird_ss, species_observed) %>%
  mutate(percent = n / sum(n))
```

```
## # A tibble: 2 x 3
##   species_observed     n percent
##   <lgl>            <int>   <dbl>
## 1 FALSE            40379  0.913
## 2 TRUE              3828  0.0866
```

Now, let's transform our eBird data into an sf file, obtain a shape file of Arizona for plotting, and download a DEM as a raster:

```
# set map projections
prj_dd <- "EPSG:4326"
map_proj <- st_crs("ESRI:102003")

# transform ebird data to sf
ebird_sf <- ebird_ss %>%
  # convert to spatial points
  st_as_sf(coords = c("longitude", "latitude"), crs = 4326) %>%
  st_transform(crs = map_proj)

# get state and county shpfiles
states <- st_as_sf(map("state", plot = FALSE, fill = TRUE), crs = map_proj)
arizona <- states[states$ID=="arizona",]
counties <- st_as_sf(map("county", plot = FALSE, fill = TRUE), crs = map_proj)
counties <- subset(counties, grepl("arizona", counties$ID))

# download state elev raster or load locally
f_elev_state = file.path(data_dir, "az_elevs.tif")
if (!file.exists(f_elev_state)) {
  elevation_data <- get_elev_raster(locations=arizona, z = 7, clip = "locations")
      terra::writeRaster(elevation_data, f_elev_state, filetype = "GTiff")
} else {
    elevation_data <- rast(f_elev_state)
}

# calculate ground resolution for z=7:
ground_resolution <- (cos(32.25 * pi/180) * 2 * pi * 6378137) / (256 * 2^7)
ground_resolution
```

```
## [1] 1034.319
```

Let's crop the raster down to scale and drop NAs:

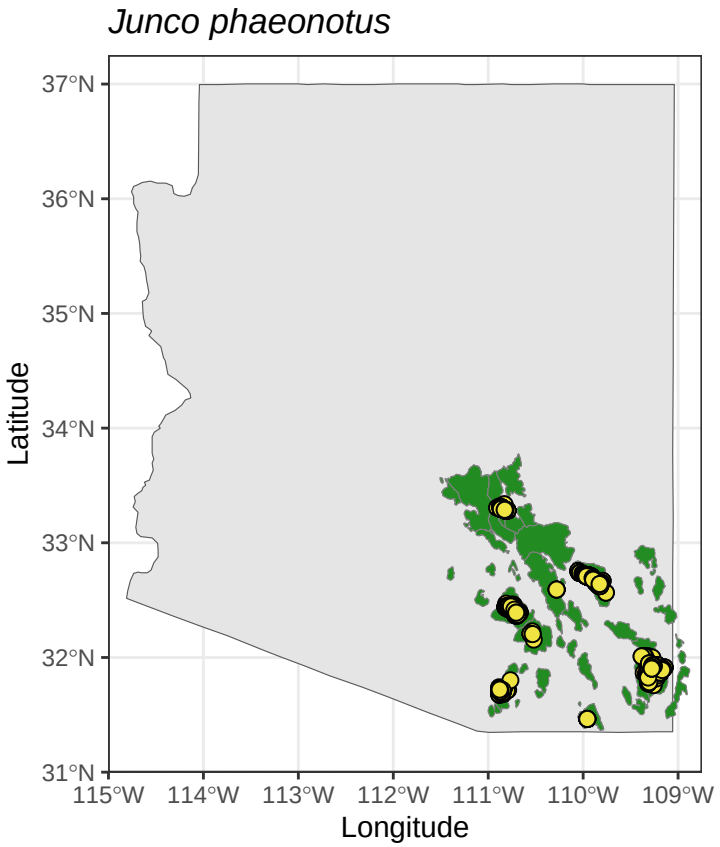
```r
# crop raster down to scale
cropped_elev <- crop(elevation_data, counties)
aspect <- terra::terrain(cropped_elev, v = "aspect", unit = "degrees")
elev_aspect <- c(cropped_elev, aspect)
elev_aspect_df <- as.data.frame(elev_aspect, xy = TRUE)

# remove rows of data frame with one or more NA's,using complete.cases
elev_aspect_df <- elev_aspect_df[complete.cases(elev_aspect_df), ]
colnames(elev_aspect_df)[3] <- "elevation"
```

It will be useful to break out observations by which mountain range they occur in. To do this, we'll want to use the Global Mountain Biodiversity Assessment's Mountain Inventory v2 mountain deliniation shapefiles. Let's load those, filter points that do not fall within our geographic area of focus, and make a second plot highlighting each mountain range:

```r
data_dir_gmba <- "data/GMBA_Inventory_v2.0_standard/"
f_mountains <- file.path(data_dir_gmba, "GMBA_Inventory_v2.0_standard.shp")
mountains <- st_read(f_mountains)
```

```
## Reading layer 'GMBA_Inventory_v2.0_standard' from data source
##   '/Users/k14m234/Dropbox/what_is_elev_range/data/GMBA_Inventory_v2.0_standard/GMBA_Inventory_v2.0_s
##   using driver 'ESRI Shapefile'
## Simple feature collection with 8327 features and 40 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -180 ymin: -55.91472 xmax: 180 ymax: 83.63908
## Geodetic CRS:  WGS 84
```

```r
mountains <- st_transform(mountains, crs = map_proj)
us_mountains <- mountains[mountains$Countries=="United States",]
sw_mountains <- us_mountains[us_mountains$Level_05=="Southeast Arizona Ranges",]
junco_points <- ebird_sf[ebird_sf$species_observed==TRUE,]
junco_points_id <- st_join(junco_points, sw_mountains, join = st_within)
junco_points_within <- junco_points_id[!is.na(junco_points_id$MapName), ]
```



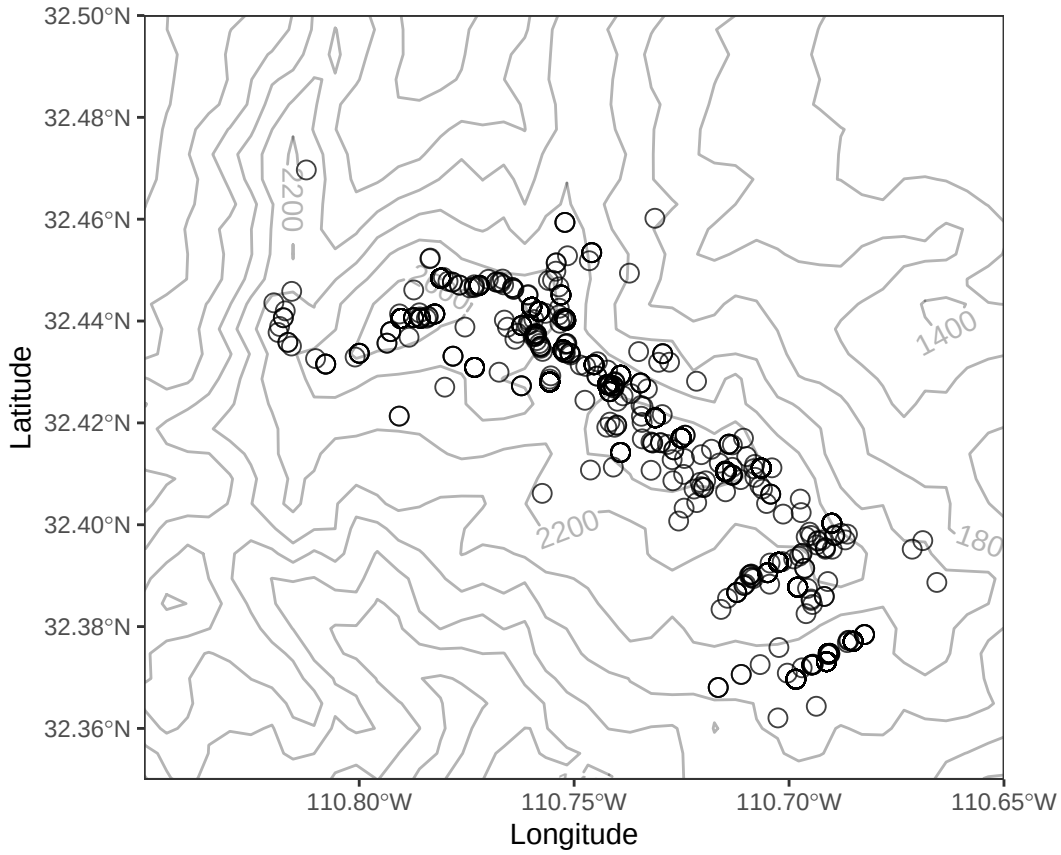
Passes the gut-check. Let's assign our points to each of these ranges, and then convert our data into a tibble for plotting:

```
# assign points to mountain ranges
ebird_range_id <- st_join(ebird_sf, sw_mountains, join = st_within)

# extract elevation and aspect data
elev_aspect_df <- raster::extract(elev_aspect, ebird_range_id)

# turn elevations and original data back to tible
ebird_range_id_tib <- as_tibble(ebird_range_id)
ebird_range_id_tib$elevation <- elev_aspect_df$az_elevs
ebird_range_id_tib$aspect <- elev_aspect_df$aspect

# select only the columns to be used in the model, drop NAs
ebdf <- ebird_range_id_tib %>%
  select(species_observed,
         year, day_of_year,
         time_observations_started, duration_minutes,
         effort_distance_km, number_observers, elevation,
         aspect, MapName,
         starts_with("pland_"),
         starts_with("elevation_")) %>%
  drop_na()
```

Elevational ranges include variation in local density and abundance

Our first argument is that elevational ranges are not simple intervals, but include important variation in the denisty of individuals (which can also be considered differences in relative abundance across elevational bands). To illustrate this, we'll select checklists from a single mountain range, plot the location of YEJUs on a topographic map, plot their location across elevation, and plot a histogram of encounter rate across elevation, overlaid with common elevational range summary statistics.

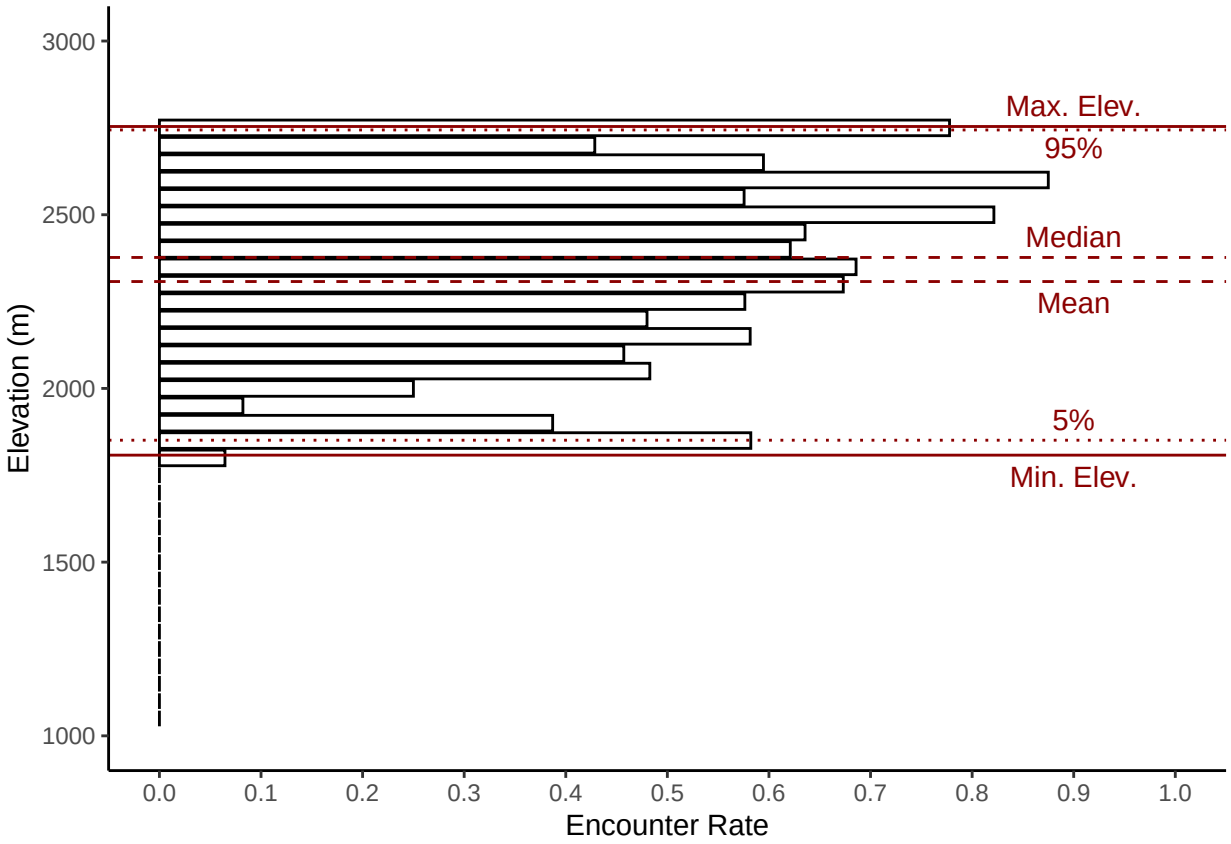
```
a <- ggplot(data=arizona) +
  theme_bw() +
  theme(panel.grid = element_blank()) +
  geom_sf(data = junco_points_within, pch=21, size=3, alpha=0.75) +
  geom_contour2(data=elev_aspect, aes(x=x, y=y, z=az_elevs, label = stat(level)),
                binwidth = 200, alpha=0.3) +
  coord_sf(xlim = c(-110.85, -110.65), ylim = c(32.35, 32.5), expand = FALSE) +
  ylab("Latitude") +
  xlab("Longitude")
a
```



Let's visualize the distribution of records across the gradient:

```
# subset santa catalinas
catalinas <- ebdf[ebdf$MapName=="Santa Catalina Mountains",]

# assign each observation using a 100-m bin
df <- catalinas %>% mutate(bin = elevation %/% 50) %>%
  group_by(bin) %>%
```

```
  summarise(observed = mean(species_observed)) %>%
  #mutate(observed = observed/max(observed)) %>%
  mutate(bin = bin*50)

# calculate range descriptors / summary statistics
df$min <- catalinas[catalinas$species_observed==TRUE,]$elevation %>% min()
df$max <- catalinas[catalinas$species_observed==TRUE,]$elevation %>% max()
df$five <- catalinas[catalinas$species_observed==TRUE,]$elevation %>% quantile(probs=c(0.05))
df$ninetyfive <- catalinas[catalinas$species_observed==TRUE,]$elevation %>% quantile(probs=c(0.95))
df$median <- catalinas[catalinas$species_observed==TRUE,]$elevation %>% median()
df$mean <- catalinas[catalinas$species_observed==TRUE,]$elevation %>% mean()


b <- ggplot(df, aes(x = bin, y=observed)) +
  theme_classic() +
  geom_bar(stat = "identity", color = "black", fill="white") +
  coord_flip() +
  geom_vline(aes(xintercept = min), color = 'darkred',
             linetype="solid") +
  annotate("text", x= unique(df$min)-60, y=0.9, label="Min. Elev.", color = "darkred") +
  geom_vline(aes(xintercept = max), color = 'darkred',
             linetype="solid") +
  annotate("text", x= unique(df$max)+60, y=0.9, label="Max. Elev.", color = "darkred") +
  geom_vline(aes(xintercept = median), color = 'darkred',
             linetype="dashed") +
  annotate("text", x= unique(df$median)+60, y=0.9, label="Median", color = "darkred") +
  geom_vline(aes(xintercept = mean), color = 'darkred',
             linetype="dashed") +
  annotate("text", x= unique(df$mean)-60, y=0.9, label="Mean", color = "darkred") +
  #geom_vline(aes(xintercept = abund), color = 'darkred',
  #           linetype="dashed") +
  #annotate("text", x= unique(df$abund)-60, y=0.75, label="Abundance Peak", color = "darkred") +
  geom_vline(aes(xintercept = five), color = 'darkred',
             linetype="dotted") +
  annotate("text", x= unique(df$five)+60, y=0.9, label="5%", color = "darkred") +
  geom_vline(aes(xintercept = ninetyfive-10), color = 'darkred',
             linetype="dotted") +
  annotate("text", x= unique(df$ninetyfive)-60, y=0.9, label="95%", color = "darkred") +
  xlim(1000,3000) +
  scale_y_continuous(breaks = seq(0, 1, by=0.1), limits=c(0,1)) +
  ylab("Encounter Rate") +
  xlab("Elevation (m)") +
  labs(linetype=NULL, fill=NULL) +
  theme(legend.position = "bottom")
b
```



Let's join these plots to create our first figure:

```
fig1 <- plot_grid(a, b, labels = c('A', 'B'), rel_widths=c(0.5,0.5))
pdf("figures/fig1.pdf",width=11, height=5)
fig1
dev.off()
```

```
## pdf
##   2
```

Elevational ranges are estimated, not observed directly

Our second argument is that elevational ranges are estimated, not directly observed. To do this, we will show the impact of subsampling our full YEJU dataset on an estimate of the impact of elevation on encounter rate from a random forest model built in `ranger`.

```
# set seed for reproducibility
set.seed(9927)

#create subsampled datasets
ebdf_10 <- ebdf %>% slice_sample(n=10)
df_10 <- ebdf_10 %>% mutate(bin = elevation %/% 50) %>%
  group_by(bin) %>%
  summarise(observed = mean(species_observed)) %>%
  #mutate(observed = observed/max(observed)) %>%
  mutate(bin = bin*50)
```

```
ebdf_100 <- ebdf %>% slice_sample(n=100)
df_100 <- ebdf_100 %>% mutate(bin = elevation %/% 50) %>%
  group_by(bin) %>%
  summarise(observed = mean(species_observed)) %>%
  #mutate(observed = observed/max(observed)) %>%
  mutate(bin = bin*50)

ebdf_1000 <- ebdf %>% slice_sample(n=1000)
df_1000 <- ebdf_1000 %>% mutate(bin = elevation %/% 50) %>%
  group_by(bin) %>%
  summarise(observed = mean(species_observed)) %>%
  #mutate(observed = observed/max(observed)) %>%
  mutate(bin = bin*50)

# split each dataset 80/20
ebird_split_10 <- ebdf_10 %>% split(if_else(runif(nrow(.)) <= 0.8, "train", "test"))
ebird_split_100 <- ebdf_100 %>% split(if_else(runif(nrow(.)) <= 0.8, "train", "test"))
ebird_split_1000 <- ebdf_1000 %>% split(if_else(runif(nrow(.)) <= 0.8, "train", "test"))

# assign detection frequency
detection_freq_10 <- mean(ebird_split_10$train$species_observed)
detection_freq_100 <- mean(ebird_split_100$train$species_observed)
detection_freq_1000 <- mean(ebird_split_1000$train$species_observed)

# factor response so ranger can do do classification
ebird_split_10$train$species_observed <- factor(ebird_split_10$train$species_observed)
ebird_split_100$train$species_observed <- factor(ebird_split_100$train$species_observed)
ebird_split_1000$train$species_observed <- factor(ebird_split_1000$train$species_observed)

# grow random forests
rf_10 <- ranger(formula =  species_observed ~ .,
             data = ebird_split_10$train,
             importance = "impurity",
             probability = TRUE,
             replace = TRUE,
             sample.fraction = c(detection_freq_10, detection_freq_10))

rf_100 <- ranger(formula =  species_observed ~ .,
             data = ebird_split_100$train,
             importance = "impurity",
             probability = TRUE,
             replace = TRUE,
             sample.fraction = c(detection_freq_100, detection_freq_100))

rf_1000 <- ranger(formula =  species_observed ~ .,
             data = ebird_split_1000$train,
             importance = "impurity",
             probability = TRUE,
             replace = TRUE,
             sample.fraction = c(detection_freq_1000, detection_freq_1000))
```

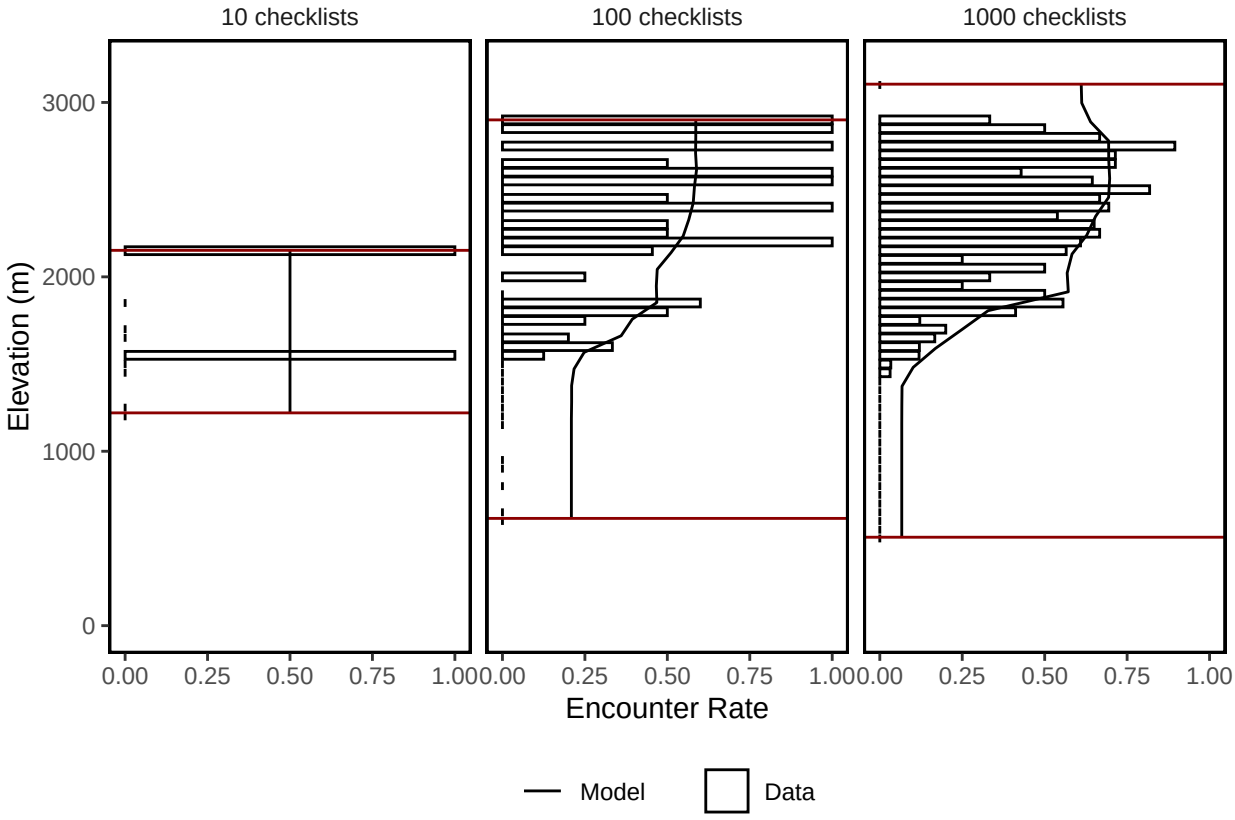
Next, we'll calculate "partial dependence" (or marginal effect) of each predictor on encounter rate:

```r
# set new seed
set.seed(1927)

# function to calculate partial dependence for a single predictor
calculate_pd <- function(predictor, model, data,
                         x_res = 25, n = 1000) {
  # create prediction grid
  rng <- range(data[[predictor]], na.rm = TRUE)
  x_grid <- seq(rng[1], rng[2], length.out = x_res)
  grid <- data.frame(covariate = predictor, x = x_grid,
                     stringsAsFactors = FALSE)
  names(grid) <- c("covariate", predictor)

  # subsample training data
  n <- min(n, nrow(data))
  s <- sample(seq.int(nrow(data)), size = n, replace = FALSE)
  data <- data[s, ]

  # drop focal predictor from data
  data <- data[names(data) != predictor]
  grid <- merge(grid, data, all = TRUE)

  # predict
  p <- predict(model, data = grid)

  # summarize
  pd <- grid[, c("covariate", predictor)]
  names(pd) <- c("covariate", "x")
  pd$pred <- p$predictions[, 2]
  pd <- dplyr::group_by(pd, covariate, x) %>%
    dplyr::summarise(pred = mean(pred, na.rm = TRUE)) %>%
    dplyr::ungroup()

  return(pd)
}

# extract values for elevation alone:
pd_10 <- calculate_pd(model = rf_10, data = ebdf_10, predictor = "elevation")
pd_100 <- calculate_pd(model = rf_100, data = ebdf_100, predictor = "elevation")
pd_1000 <- calculate_pd(model = rf_1000, data = ebdf_1000, predictor = "elevation")
```

We'll combine subsampled datasets to plot a comparison of the impact of data quantity on encounter rate:

```r
pd_10$id <- "10 checklists"
pd_100$id <- "100 checklists"
pd_1000$id <- "1000 checklists"
pd_df <- bind_rows(pd_10, pd_100, pd_1000)
df_10$id <- "10 checklists"
df_100$id <- "100 checklists"
df_1000$id <- "1000 checklists"
df_compare <- bind_rows(df_10, df_100, df_1000)
stats_df <- tibble(
  id = c("10 checklists", "100 checklists", "1000 checklists"),
```

```
  max = c(pd_10[pd_10$pred>0.05,]$x %>% max(), pd_100[pd_100$pred>0.05,]$x %>% max(), pd_1000[pd_1000$p
  min = c(pd_10[pd_10$pred>0.05,]$x %>% min(), pd_100[pd_100$pred>0.05,]$x %>% min(), pd_1000[pd_1000$p
)
```



An alternate approach to demonstrate the same point would be to plot the raw data from each subsample with estimated limits:

```
set.seed(1927)

# separate subsamples
ebdf_10 <- ebdf %>% slice_sample(n=10)
ebdf_100 <- ebdf %>% slice_sample(n=100)
ebdf_1000 <- ebdf %>% slice_sample(n=1000)

# merge datasets
ebdf_10$id <- "10 checklists"
ebdf_100$id <- "100 checklists"
ebdf_1000$id <- "1000 checklists"
ebdf_merged <- rbind(ebdf_10, ebdf_100, ebdf_1000)
ebdf_merged$id <- factor(ebdf_merged$id, levels=c("10 checklists", "100 checklists", "1000 checklists"))

# stat summary function
f <- function(x) {
  r <- quantile(x, probs = c(0.05, 0.25, 0.5, 0.75, 0.95))
  names(r) <- c("ymin", "lower", "middle", "upper", "ymax")
  r
}
```

```
f <- ggplot(ebdf_merged, aes(id, elevation)) +
  theme_classic() +
  geom_jitter(width = 0.3, aes(shape=species_observed, size=species_observed)) +
  geom_boxplot(data=ebdf_merged[ebdf_merged$species_observed==TRUE,]) +
  scale_shape_manual(values=c(3,21), labels=c("Unobserved","Observed")) +
  scale_size_manual(values=c(1,2), labels=c("Unobserved","Observed"))+
  xlab("Sample Size") +
  ylab("Elevation (m)") +
  ylim(0,3200) +
  theme(legend.title = element_blank()) +
  theme(legend.position = "bottom")
```

Let's combine these for Figure 2:

```
pdf("figures/fig2.pdf",width=12, height=5)
plot_grid(f, e, labels = c('A', 'B'), rel_widths=c(0.35,0.65))
dev.off()
```

```
## pdf
##   2
```

Elevational ranges are spatially and temporally dynamic, and that variation is meaningful

Next, we want to demonstrate variation in elevational ranges across time and aspect. Let's focus on the Pinaleños, a large mountain range with good sampling.

```
# subset mountain ranges
pinalenos <- ebdf[ebdf$MapName=="Pinaleño Mountains",]
pinalenos$cardinal <- cut(pinalenos$aspect, breaks = c(0,seq(45,315,45),360),
              labels = c("N","NE","E","SE","S","SW","W","NW"),
              include.lowest = TRUE)
ritas <- ebdf[ebdf$MapName=="Santa Rita Mountains",]
ritas$cardinal <- cut(ritas$aspect, breaks = c(0,seq(45,315,45),360),
              labels = c("N","NE","E","SE","S","SW","W","NW"),
              include.lowest = TRUE)
aspect_plot <- rbind(pinalenos, ritas)
juncos <- aspect_plot[aspect_plot$species_observed==TRUE,]

# function to assign observations to quantile
func <- function(x) {
  r <- quantile(x, probs = c(0.05, 0.25, 0.5, 0.75, 0.95))
  names(r) <- c("ymin", "lower", "middle", "upper", "ymax")
  r
}

# plot and assign to pane
g <- ggplot(aspect_plot, aes(x = cardinal, y = elevation)) +
  geom_jitter(aes(shape=species_observed, size=species_observed), color="black", fill="white") +
  stat_summary(data=aspect_plot[aspect_plot$species_observed==TRUE,], fun.data = func, geom="boxplot") +
  scale_shape_manual(values=c(3,21), labels=c("Unobserved","Observed")) +
```

```
  scale_size_manual(values=c(1,2), labels=c("Unobserved","Observed"))+
  guides(fill = guide_legend(reverse = TRUE)) +
  #geom_jitter(pch=21) +
  coord_polar() +
  #ylim(0,3300) +
  scale_y_continuous(breaks = seq(0, 3200, by = 250), guide = FALSE) +
  theme_bw() +
  theme(strip.background = element_blank(),
        strip.text = element_text(size=12),
        axis.title.x = element_blank()) +
  theme(legend.position = "none") +
  ylab("Elevation (m)") +
  facet_wrap(~MapName)
```

We'll look at year to year variation in elevational range quantiles for the Santa Catalinas:

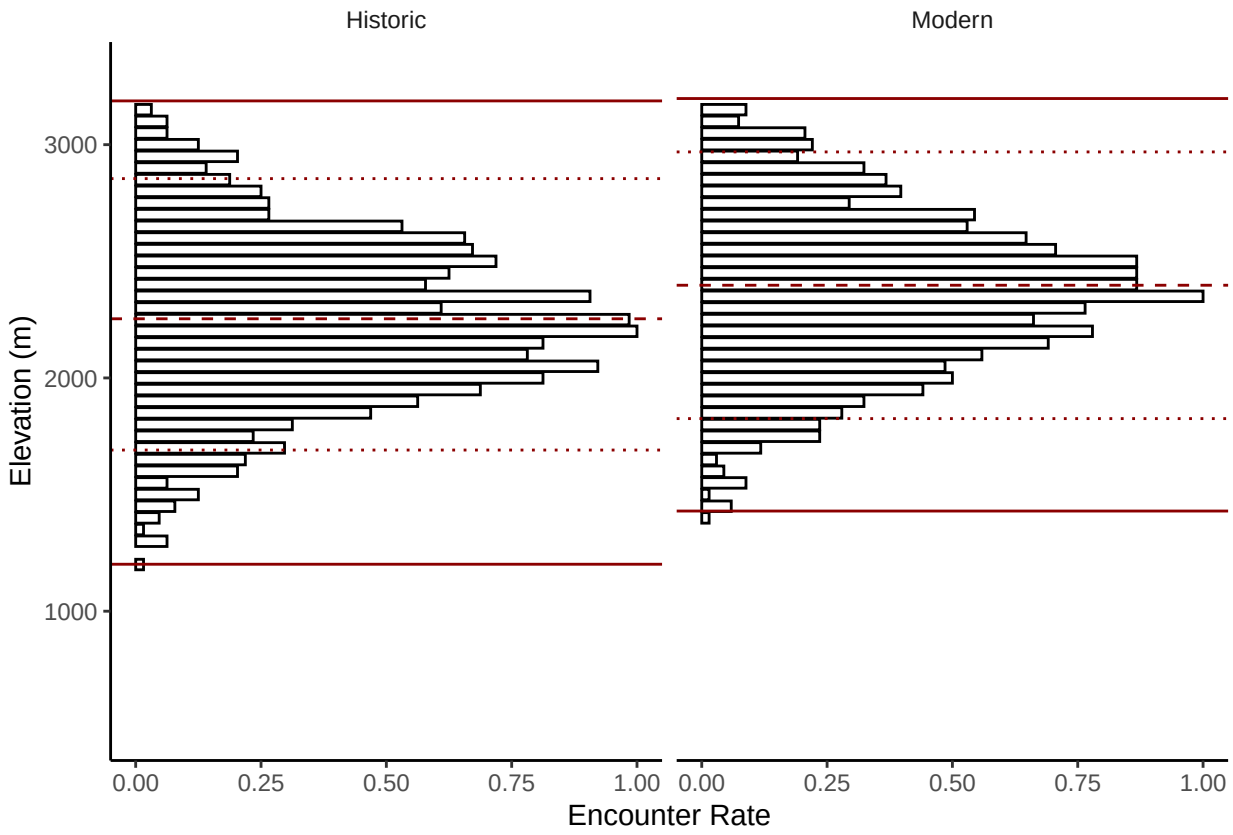
```
## pdf
##   2
```

Range Shift Inferences Are Sensitive to Elevational Range Concepts and Definitions

In this section we will simulate encounter rate data across elevation for a population that has undergone a range shift:

```
set.seed(8676)
mean <- ebdf[ebdf$species_observed==TRUE,]$elevation %>% mean()
sd <- ebdf[ebdf$species_observed==TRUE,]$elevation %>% sd()
overall <- count(ebird_ss, species_observed) %>%
  mutate(percent = n / sum(n))
percent_yes <- overall[2,3] %>% as.numeric

# simulate pre-range-shift encounters
rand_norms_old <- tibble(
  elev = c(rnorm(1000, mean=mean, sd=sd))
)
rand_norms_old <- rand_norms_old %>% filter(elev < 3200)

# simulate post-range-shift encounters
rand_norms_new <- tibble(
  elev = c(rnorm(1000, mean=(mean+150), sd=sd))
)
rand_norms_new <- rand_norms_new %>% filter(elev < 3200)

# create tibble for pre-range shift elevational distribution
df_sim_old <- rand_norms_old %>%
  filter(elev < 3200) %>%
  mutate(bin = elev %/% 50) %>%
  group_by(bin) %>%
  summarise(n()) %>%
  rename("observed"='n()') %>%
  mutate(bin = bin*50) %>%
```

```
  mutate(observed=observed/max(observed))
df_sim_old$id <- "Historic"

# create tibble for post-range shift elevational distribution
df_sim_new <- rand_norms_new %>%
  mutate(bin = elev %/% 50) %>%
  group_by(bin) %>%
  summarise(n()) %>%
  rename("observed"='n()') %>%
  mutate(bin = bin*50)  %>%
  mutate(observed=observed/max(observed))
df_sim_new$id <- "Modern"

df_sim <- bind_rows(df_sim_old, df_sim_new)

# assembly summary stats for plotting
sim_stats <- tibble(
  id = c("Historic", "Modern"),
  max = c(max(rand_norms_old), max(rand_norms_new)),
  min = c(min(rand_norms_old), min(rand_norms_new)),
  quant_05 = c(quantile(rand_norms_old$elev, probs=c(0.05)), quantile(rand_norms_new$elev, probs=c(0.05)
  quant_95 = c(quantile(rand_norms_old$elev, probs=c(0.95)), quantile(rand_norms_new$elev, probs=c(0.95)
  median = c(median(rand_norms_old$elev), median(rand_norms_new$elev)),
  mean = c(mean(rand_norms_old$elev), mean(rand_norms_new$elev))
)
```



Let's also look at range limits to understand how inferences of shifts may differ:

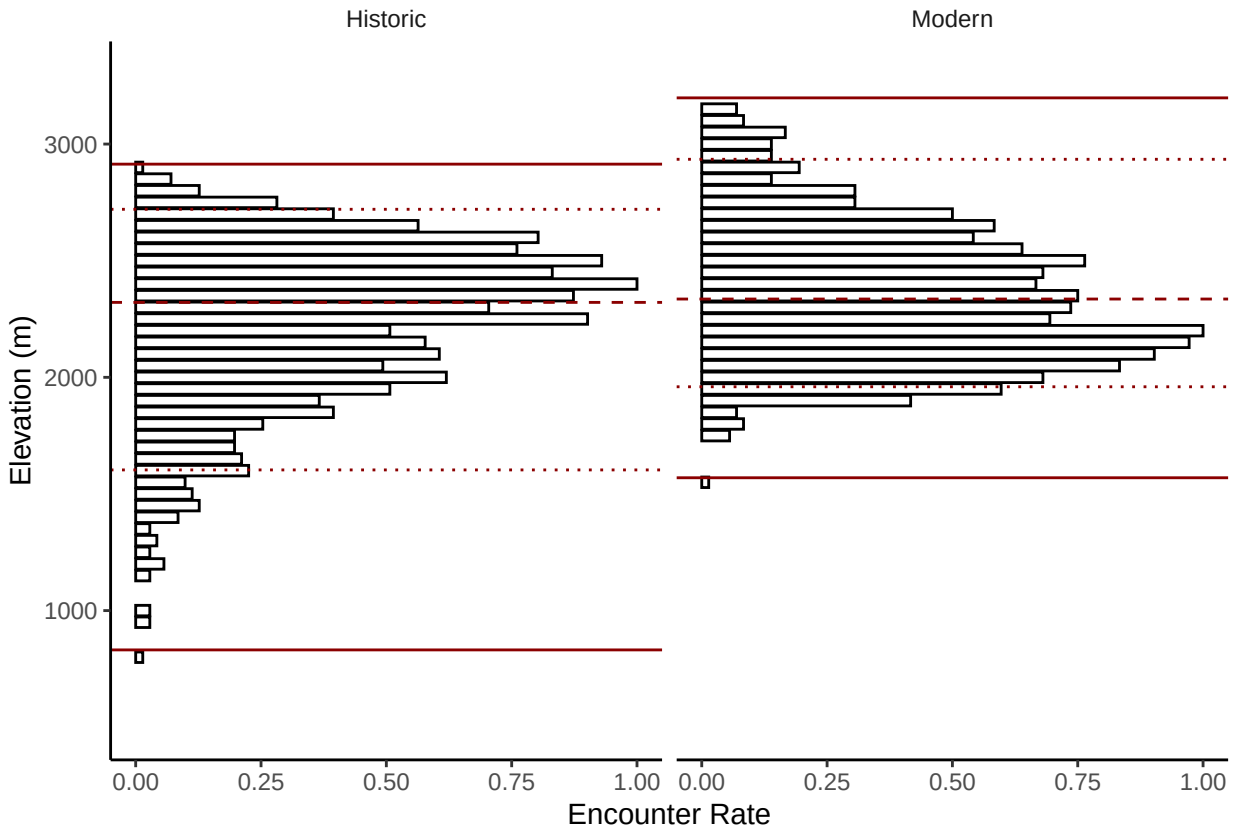
```
sim_stats
```

```
## # A tibble: 2 x 7
##   id         max   min quant_05 quant_95 median  mean
##   <chr>    <dbl> <dbl>    <dbl>    <dbl>  <dbl> <dbl>
## 1 Historic 3188. 1201.    1691.    2855.  2254. 2267.
## 2 Modern   3198. 1429.    1826.    2969.  2397. 2394.
```

Next, let's simulate a case where our species' elevational distribution is skewed to its upper range limit, but then shifts its skew during a range shift:

```
set.seed(2361)

# simulate pre-range-shift encounters
rand_norms_old_skew <- tibble(
  elev = c(rsnorm(1000, mean=mean, sd=sd, xi=-2))
)
rand_norms_old_skew <- rand_norms_old_skew %>% filter(elev < 3200)

# simulate post-range-shift encounters
rand_norms_new_skew <- tibble(
  elev = c(rsnorm(1000, mean=(mean+150), sd=sd, xi=2))
)
rand_norms_new_skew <- rand_norms_new_skew %>% filter(elev < 3200)

# create tibble for pre-range shift elevational distribution
df_sim_old_skew <- rand_norms_old_skew %>%
  filter(elev < 3200) %>%
  mutate(bin = elev %/% 50) %>%
  group_by(bin) %>%
  summarise(n()) %>%
  rename("observed"='n()') %>%
  mutate(bin = bin*50) %>%
  mutate(observed=observed/max(observed))
df_sim_old_skew$id <- "Historic"

# create tibble for post-range shift elevational distribution
df_sim_new_skew <- rand_norms_new_skew %>%
  mutate(bin = elev %/% 50) %>%
  group_by(bin) %>%
  summarise(n()) %>%
  rename("observed"='n()') %>%
  mutate(bin = bin*50)  %>%
  mutate(observed=observed/max(observed))
df_sim_new_skew$id <- "Modern"

df_sim_skew <- bind_rows(df_sim_old_skew, df_sim_new_skew)

# assembly summary stats for plotting
sim_stats_skew <- tibble(
  id = c("Historic", "Modern"),
  max = c(max(rand_norms_old_skew), max(rand_norms_new_skew)),
  min = c(min(rand_norms_old_skew), min(rand_norms_new_skew)),
```

```
  quant_05 = c(quantile(rand_norms_old_skew$elev, probs=c(0.05)), quantile(rand_norms_new_skew$elev, pr
  quant_95 = c(quantile(rand_norms_old_skew$elev, probs=c(0.95)), quantile(rand_norms_new_skew$elev, pr
  median = c(median(rand_norms_old_skew$elev), median(rand_norms_new_skew$elev)),
  mean = c(mean(rand_norms_old_skew$elev), mean(rand_norms_new_skew$elev))
)
```




```
sim_stats_skew
```

```
## # A tibble: 2 x 7
##   id          max   min quant_05 quant_95 median  mean
##   <chr>     <dbl> <dbl>    <dbl>    <dbl>  <dbl> <dbl>
## 1 Historic 2914.  832.     1603.    2720.  2321. 2259.
## 2 Modern   3198. 1569.     1960.    2935.  2336. 2373.
```

We'll combine these for our fourth figure:

```
fig4 <- plot_grid(i, j, labels=c("A","B"), nrow=2)
pdf("figures/fig4.pdf",width=6, height=7)
fig4
dev.off()
```

```
## pdf
##   2
```

Elevation Ranges are scale-dependent phenomena

Lastly, we want to show how elevational ranges and the mechanisms responsible for them vary across spatial scales. To do this, we will create a figure showing the distribution of YEJU at different scales, starting with a single slope:

```
k <- ggplot(data=arizona) +
  theme_bw() +
  theme(panel.grid = element_blank()) +
  scale_x_continuous(breaks = c(-110.82, -110.80)) +
  scale_y_continuous(breaks = c(32.425, 32.445)) +
  geom_sf(data = junco_points_within, pch=21, size=3, alpha=0.75) +
  geom_contour2(data=elev_aspect, aes(x=x, y=y, z=az_elevs, label = stat(level)),
                binwidth = 100, alpha=0.2) +
  coord_sf(xlim = c(-110.825, -110.795), ylim = c(32.45, 32.42), expand = FALSE) +
  ylab("Latitude") +
  xlab("Longitude") +
  ggtitle("Local")
```

A single mountain range:

```
l <- ggplot(data=arizona) +
  theme_bw() +
  theme(panel.grid = element_blank()) +
  scale_x_continuous(breaks = c(-110.80, -110.70)) +
  scale_y_continuous(breaks = c(32.38, 32.48)) +
  geom_sf(data = junco_points_within, pch=21, size=3, alpha=0.75) +
  geom_contour2(data=elev_aspect, aes(x=x, y=y, z=az_elevs, label = stat(level)),
                binwidth = 200, alpha=0.2) +
  coord_sf(xlim = c(-110.82, -110.67), ylim = c(32.35, 32.5), expand = FALSE) +
  ylab("Latitude") +
  xlab("Longitude") +
  ggtitle("Regional")
```

And lastly, the entire region:

```
m <- ggplot(data=arizona) +
  theme_bw() +
  theme(panel.grid = element_blank()) +
    scale_x_continuous(breaks = c(-110.5, -109.5)) +
  scale_y_continuous(breaks = c(32, 33)) +
  geom_sf(data = junco_points_within, pch=1, size=3) +
  geom_contour2(data=elev_aspect, aes(x=x, y=y, z=az_elevs),
                binwidth = 200, alpha=0.2) +
  coord_sf(xlim = c(-111, -109), ylim = c(31.5, 33.5), expand = FALSE) +
  ylab("Latitude") +
  xlab("Longitude") +
  ggtitle("Global")
```

We'll pair these plots with quantifications of the elevational range of sampled points. To do so, we need to subsample the original data:

```
zoom <- ebird_sf %>% st_transform(4326)
zoom_elev <- raster::extract(elev_aspect, zoom)
zoom$elevation <- zoom_elev$az_elevs

local <- st_crop(zoom, xmin=-110.82, xmax=-110.80, ymin=32.425, ymax=32.445)
local$longitude <- st_coordinates(local)[,1]
local$id <- "local"
regional <- st_crop(zoom, xmin=-110.85, xmax=-110.65, ymin=32.35, ymax=32.55)
regional$id <- "regional"
global <- st_crop(zoom, xmin=-111, xmax=-109, ymin=31, ymax=33)
global$id <- "global"
```

Lastly, we'll prepare simple tables laying out our hypotheses for range limiting mechanisms:

```
Scale <- c("Local","Regional","Global")
Field <- c("Ecophysiology, experimental ecology, community ecology, resurveys", "Ecological biogeography
Hypothesis <- c("Range limits are determined by indvidual biotic and abiotic tolerance and natal dispers
                "Range limits are determined by metapopulation dynamics",
                "Range limits determined by adaptive evolution and historical contengency")
Prediction <- c("Low organismal performance beyond range limits",
                "Deaths exceed births and immigration past range limits",
                "Phylogenetic signal in range elevation")
Data <- c("Raw elevations of observations", "Binned relative abundances", "Summary statistics across po
table1 <- data.frame(Scale, Field, Hypothesis, Prediction, Data) %>% flextable() %>% gen_grob()
plot_a <- plot_grid(top, middle, nrow=2, rel_heights = c(1,0.9))
plot_b <- plot_grid(table1, nrow=1, scale=0.8)
```

```
## pdf
##   2
```