Geographic overlap from SDM and Ace2 results

Norah Saarman

2025-04-18

R build: Geospatial 4.4.0

1. Load and Process Species Distribution Models (SDMs)

```
# Load MaxEnt SDMs
sdm_pipiens <- raster("../gis/culex_pipiens_meansuitability.nc")
sdm_quinque <- raster("../gis/culex_quinquefasciatus_meansuitability.nc")

# Threshold to binary
threshold <- 0.5
sdm_pipiens_bin <- sdm_pipiens >= threshold
sdm_quinque_bin <- sdm_quinque >= threshold
```

2. Convert SDMs to Polygons

```
# Raster to terra
sdm_pipiens_v <- terra::rast(sdm_pipiens_bin)
sdm_quinque_v <- terra::rast(sdm_quinque_bin)

# Raster to polygons
poly_pipiens <- terra::as.polygons(sdm_pipiens_v, dissolve = TRUE)
poly_quinque <- terra::as.polygons(sdm_quinque_v, dissolve = TRUE)

# Terra to sf
poly_pipiens_sf <- st_as_sf(poly_pipiens)
poly_quinque_sf <- st_as_sf(poly_quinque)

# Filter to presence only
names(poly_pipiens_sf)[1] <- "presence"
names(poly_quinque_sf)[1] <- "presence"
poly_pipiens_sf <- poly_pipiens_sf %>% filter(presence == 1)
poly_quinque_sf <- poly_quinque_sf %>% filter(presence == 1)
```

3. Project to Albers Equal Area and Calculate Overlap

```
# Define CRS
aea_crs <- st_crs("+proj=aea +lat_1=29.5 +lat_2=45.5 +lat_0=23 +lon_0=-96")

# Project
```

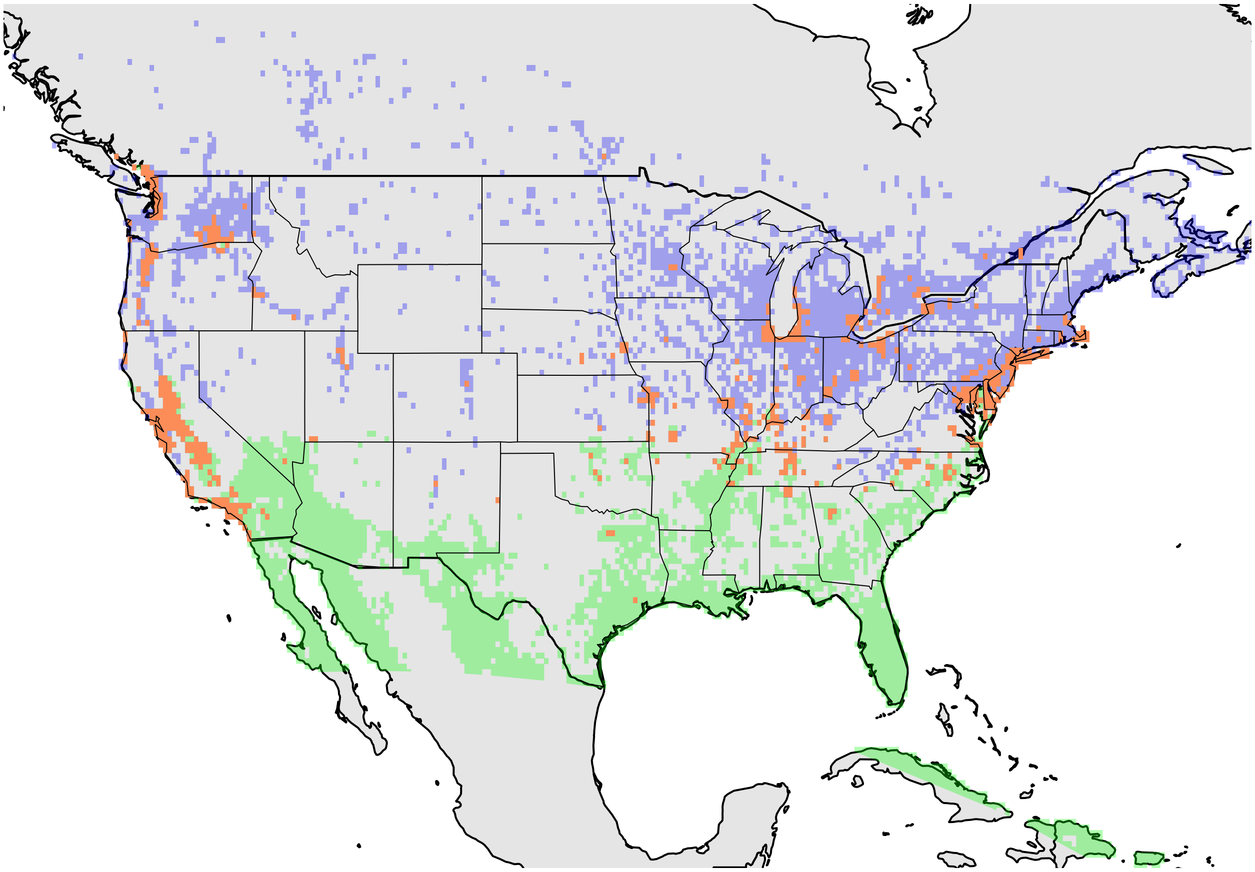
```r
poly_pipiens_sf <- st_transform(poly_pipiens_sf, aea_crs)
poly_quinque_sf <- st_transform(poly_quinque_sf, aea_crs)

# Overlap
overlap_sf <- st_intersection(poly_pipiens_sf, poly_quinque_sf)

# Calculate areas (km²)
area_pipiens_km2 <- sum(st_area(poly_pipiens_sf)) / 1e6
area_quinque_km2 <- sum(st_area(poly_quinque_sf)) / 1e6
area_overlap_km2 <- sum(st_area(overlap_sf)) / 1e6
```

4. Clip to North America and Reproject to WGS84

```r
# US base map
us_states <- st_as_sf(map("usa", plot = FALSE, fill = TRUE))
us_states <- st_transform(us_states, crs = aea_crs)

# Clip extent
bbox_na <- st_as_sfc(st_bbox(c(xmin = -170, xmax = -50, ymin = 5, ymax = 85), crs = st_crs(4326)))
bbox_na_sf <- st_transform(bbox_na, crs = aea_crs)

# Clip
poly_pipiens_sf <- st_intersection(st_make_valid(poly_pipiens_sf), bbox_na_sf)
poly_quinque_sf <- st_intersection(st_make_valid(poly_quinque_sf), bbox_na_sf)
overlap_sf <- st_intersection(st_make_valid(overlap_sf), bbox_na_sf)

# Reproject to WGS84
poly_pipiens_ll <- st_transform(poly_pipiens_sf, 4326)
poly_quinque_ll <- st_transform(poly_quinque_sf, 4326)
overlap_ll <- st_transform(overlap_sf, 4326)
```

5. Visualize SDMs and Overlap

```r
par(xpd = NA)
map("usa")
map(add = TRUE, col = "grey90", fill = TRUE)

plot(st_geometry(poly_pipiens_ll), col = rgb(0, 0, 1, 0.3), border = NA, add = TRUE)
plot(st_geometry(poly_quinque_ll), col = rgb(0, 1, 0, 0.3), border = NA, add = TRUE)
plot(st_geometry(overlap_ll), col = adjustcolor("#fc8d59", alpha.f = 1), border = NA, add = TRUE)

map("state", add = TRUE, col = "black", lwd = 0.5)
```



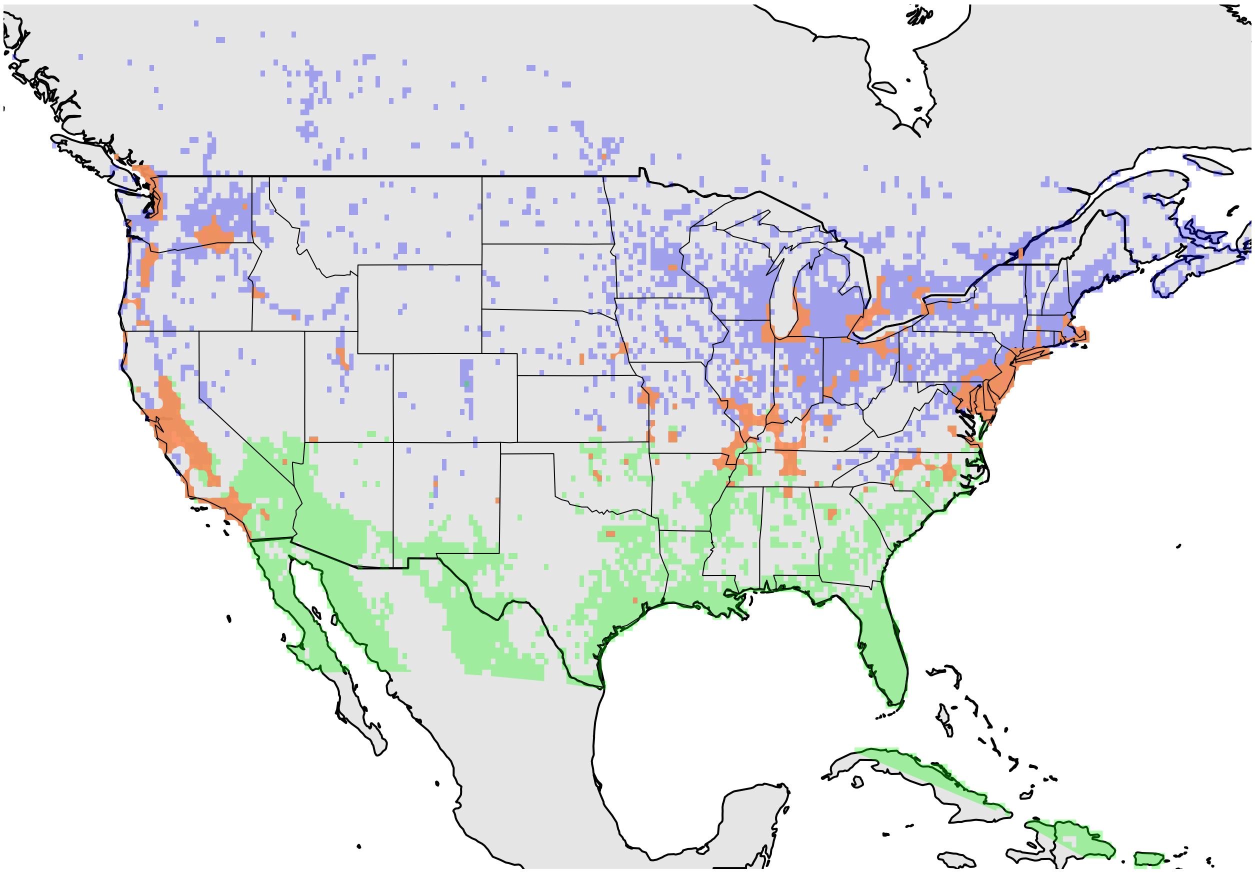
6. Merge Nearby Overlapping zones

```
# Reproject overlap
overlap_ll_proj <- st_transform(overlap_ll, aea_crs)

# Make valid and extract polygons
overlap_valid <- st_make_valid(overlap_ll_proj)
overlap_polygons <- st_collection_extract(overlap_valid, "POLYGON")
overlap_parts <- st_cast(overlap_polygons, "POLYGON")

# Buffer outward
buffer_dist_meters <- 25000
overlap_buffered <- st_buffer(overlap_parts, dist = buffer_dist_meters)
overlap_buffered <- st_make_valid(overlap_buffered)

# Merge touching patches
overlap_combined <- st_union(overlap_buffered)
overlap_combined <- st_make_valid(overlap_combined)

# Buffer inward
overlap_combined <- st_buffer(overlap_combined, dist = -buffer_dist_meters)
overlap_combined <- st_make_valid(overlap_combined)

# Finalize
overlap_combined <- st_cast(overlap_combined, "MULTIPOLYGON")
```

```r
overlap_combined <- st_transform(overlap_combined, 4326)
```

7. Visualize Combined Overlap zones

```r
#pdf("../figs/overlap_SDM.pdf", width = 8, height = 6)
par(xpd = NA)
map("usa")
map(add = TRUE, col = "grey90", fill = TRUE)

plot(st_geometry(poly_pipiens_ll), col = rgb(0, 0, 1, 0.3), border = NA, add = TRUE)
plot(st_geometry(poly_quinque_ll), col = rgb(0, 1, 0, 0.3), border = NA, add = TRUE)
plot(st_geometry(overlap_combined), col = adjustcolor("#fc8d59", alpha.f = 0.9), border = NA, add = TRUE

map("state", add = TRUE, col = "black", lwd = 0.5)
```

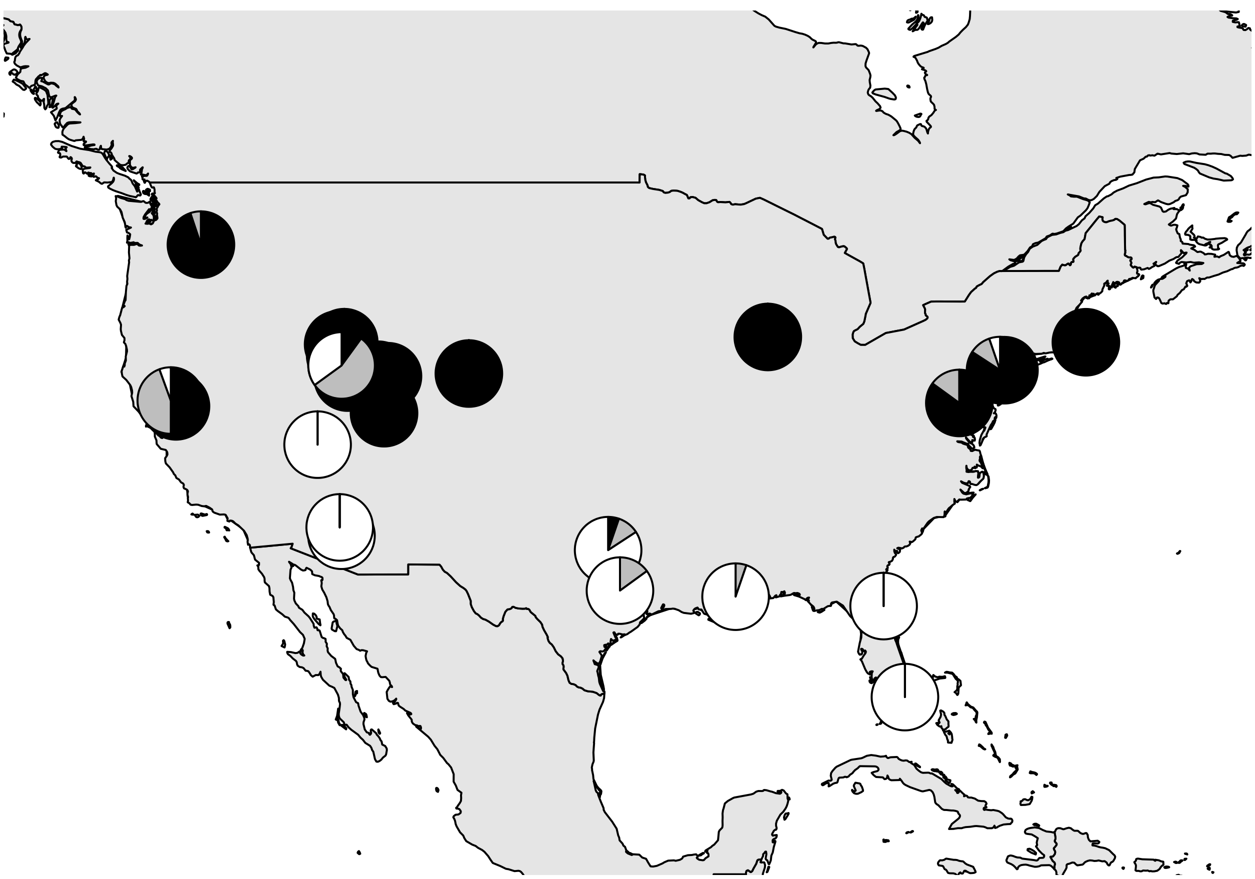


```r
#dev.off()
```

8. Add Sampling Points and Population Structure (Pie Charts)

```r
# Load counts
counts <- read.csv("../data/ThisStudy_v9.txt", sep = "\t")
coord <- as.data.frame(counts[,c("long","lat")])
```

```r
# Prepare
countsDf <- as.data.frame(counts[,c(2,1,4,5,6,9,10,7,8)])
names(countsDf) <- c("locality","site","pp","pq","qq","latitude","longitude","year","h_index")
rownames(countsDf) <- countsDf$site

# Frequencies
freqsDf <- t(apply(countsDf[,c("pp","pq","qq")], 1, function(row) row / sum(row)))
freqsDf <- as.matrix(freqsDf)

# Plot pies
par(xpd = NA)
plot_order <- rev(order(countsDf$h_index))

map("usa")
map(add = TRUE, col = "grey90", fill = TRUE)

for (i in plot_order) {
  add.pie(z = freqsDf[i,],
          x = coord[i,1],
          y = coord[i,2],
          clockwise = TRUE,
          labels = "",
          col = c("black", "grey", "white"),
          cex = 1.5, radius = 1.5)
}
```



Some jitter, added by hand:

Create a jitter data frame by hand:

```
# Create a jitter df
jitterDf <- data.frame(x_jitter = rep(0,26), y_jitter = rep(0,26))
rownames(jitterDf) <- rownames(countsDf)
jitterDf[3,] <- c(0,1)
jitterDf[4,] <- c(5,-1)
jitterDf[5,] <- c(1,-1)
jitterDf[6,] <- c(-2,0)
jitterDf[7,] <- c(-4,-1)
jitterDf[8,] <- c(-4,1)
jitterDf[9,] <- c(2,-3)
jitterDf[10,] <- c(5,-1)
jitterDf[15,] <- c(-7,3.5)
jitterDf[17,] <- c(-6,-1.5)
jitterDf[20,] <- c(1,1)
jitterDf[21,] <- c(1,-1)
```

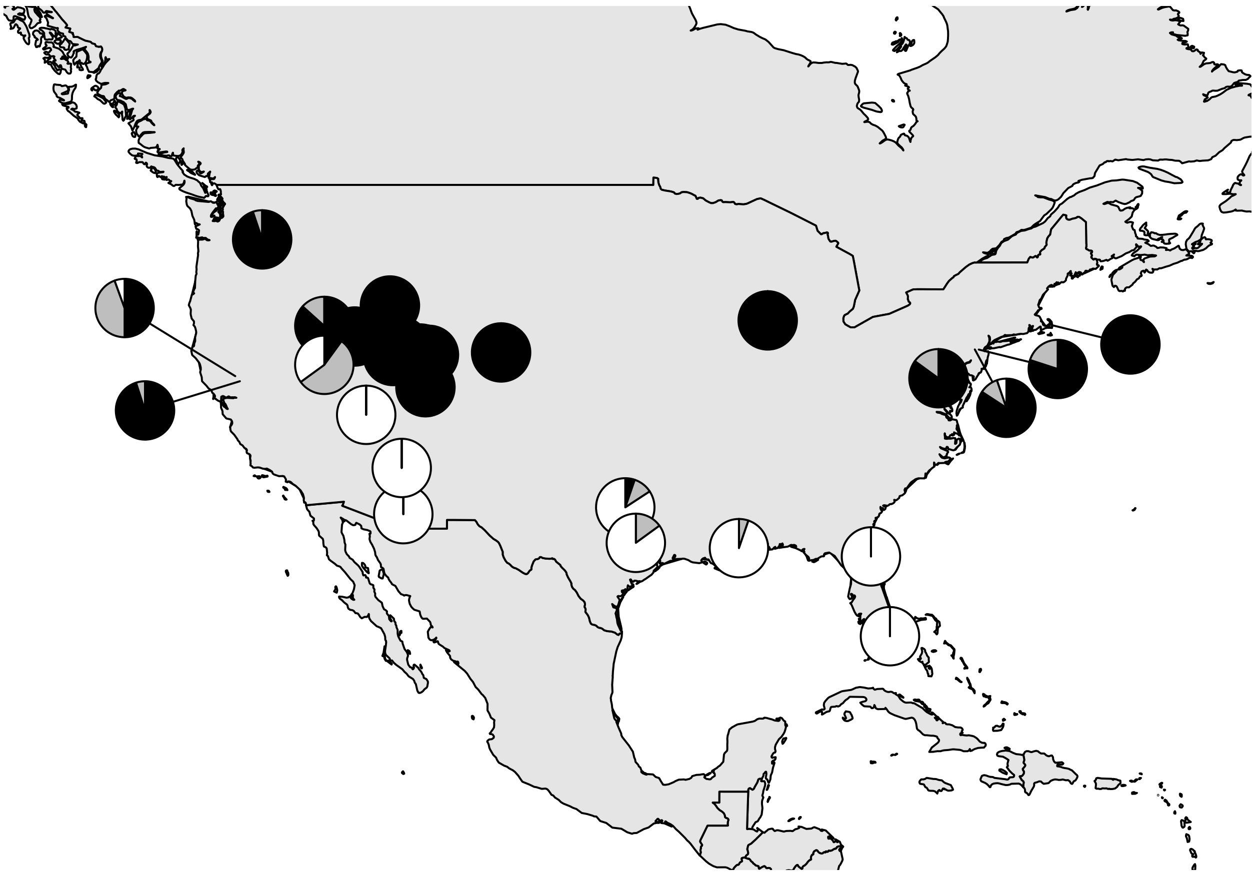
Calculate bounds of map based on coordinates + jitter:

```
# Expand x/y limits by a small margin (e.g., 2 degrees)
x_range <- range(coord[, 1] + jitterDf[, 1], na.rm = TRUE)
y_range <- range(coord[, 2] + jitterDf[, 2], na.rm = TRUE)

x_margin <- 2
y_margin <- 2

x_lim <- c(x_range[1] - x_margin, x_range[2] + x_margin)
y_lim <- c(y_range[1] - y_margin, y_range[2] + y_margin)
```

Add jitter with a line from true origin:

```
# Open PDF device
#pdf("../figs/ace2_pies_ThisStudy.pdf", width = 8, height = 6)

# Set xpd to NA to allow for plotting in the margins
par(xpd = NA)

# Plot map with expanded margins
map("usa", xlim = x_lim, ylim = y_lim)
map(add = TRUE, col = "grey90", fill = TRUE)

for (i in plot_order) {
  # Jittered coordinates
  jittered_x <- coord[i, 1] + (jitterDf[i, 1])
  jittered_y <- coord[i, 2] + (jitterDf[i, 2])

  # Draw line from true location to jittered pie
  segments(x0 = coord[i, 1],
           y0 = coord[i, 2],
           x1 = jittered_x,
           y1 = jittered_y,
           col = "black", lty = 1, lwd = 1)

  # Plot pie chart at jittered location
  add.pie(z = freqsDf[i,],
```

```
        x = jittered_x,
        y = jittered_y,
        clockwise = TRUE,
        labels = "",
        col = c("black", "grey", "white"),
        cex = 1.5, radius = 1.5)
}
```

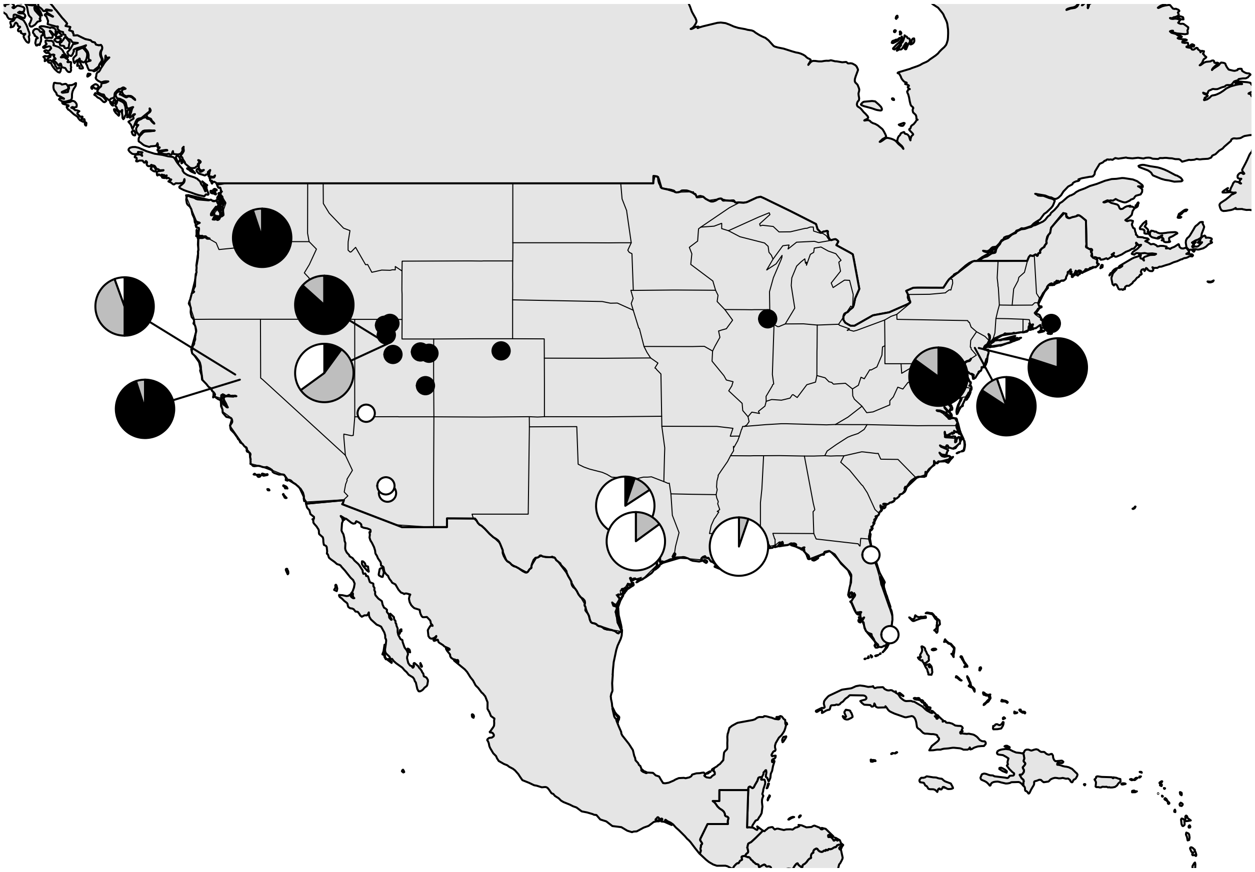


```
#dev.off()
```

Add jitter with a line from true origin AND fixed sites as small dots

```
# Open PDF device
#pdf("../figs/ace2_pies_ThisStudy.pdf", width = 8, height = 6)

#ADJUST the jitter df
jitterDf <- data.frame(x_jitter = rep(0,26), y_jitter = rep(0,26))
rownames(jitterDf) <- rownames(countsDf)
#jitterDf[3,] <- c(0,1)
#jitterDf[4,] <- c(5,-1)
#jitterDf[5,] <- c(1,-1)
#jitterDf[6,] <- c(-2,0)
jitterDf[7,] <- c(-4,-1.5)
jitterDf[8,] <- c(-4,2)
jitterDf[9,] <- c(2,-3)
jitterDf[10,] <- c(5,-1)
jitterDf[15,] <- c(-7,3.5)
```

```r
jitterDf[17,] <- c(-6,-1.5)
#jitterDf[20,] <- c(1,1)
#jitterDf[21,] <- c(1,-1)

# Set xpd to NA to allow for plotting in the margins
par(xpd = NA)

# Add flag for fixed sites and what type of fixed (pp or qq)
freqs_fixed <- freqsDf[, 1] == 1 | freqsDf[, 3] == 1
freqs_fixed_type <- ifelse(freqsDf[, 1] == 1, "pp",
                           ifelse(freqsDf[, 3] == 1, "qq", NA))

# Plot map with expanded margins
map("usa", xlim = x_lim, ylim = y_lim)
map(add = TRUE, col = "grey90", fill = TRUE)


map("state", add = TRUE, col = "black", lwd = 0.5)

for (i in plot_order) {
  jittered_x <- coord[i, 1] + (jitterDf[i, 1])
  jittered_y <- coord[i, 2] + (jitterDf[i, 2])

  segments(x0 = coord[i, 1],
           y0 = coord[i, 2],
           x1 = jittered_x,
           y1 = jittered_y,
           col = "black", lty = 1, lwd = 1)

  if (freqs_fixed[i]) {
    if (freqs_fixed_type[i] == "pp") {
      # Black dot
      points(jittered_x, jittered_y, pch = 21, bg = "black", col = "black", cex = 1.2)
    } else if (freqs_fixed_type[i] == "qq") {
      # White dot with black border
      points(jittered_x, jittered_y, pch = 21, bg = "white", col = "black", cex = 1.2)
    }
  } else {
    add.pie(z = freqsDf[i,],
            x = jittered_x,
            y = jittered_y,
            clockwise = TRUE,
            labels = "",
            col = c("black", "grey", "white"),
            cex = 1.5, radius = 1.5)
  }
}
```



```r
#dev.off()
```

Predicted level of overlapping suitable habitat?

```r
# -------------------------------------------------
# Quantify predicted overlap around each sample point
# -------------------------------------------------

# Step 1: Make sample points an sf object
samples_sf <- st_as_sf(countsDf, coords = c("longitude", "latitude"), crs = 4326)

# Step 2: Reproject sample points to Albers Equal Area CRS
samples_sf <- st_transform(samples_sf, crs = aea_crs)

# Step 3: Buffer each point by 30 km (30,000 meters)
sample_buffers <- st_buffer(samples_sf, dist = 30000)

# Step 4: Prepare overlap_combined layer in same CRS
overlap_combined_proj <- st_transform(overlap_combined, crs = aea_crs)

# Step 5: Calculate proportion of buffer area that overlaps
predicted_overlap_prop <- sapply(1:nrow(sample_buffers), function(i) {
  buf <- sample_buffers[i,]
  intersection <- st_intersection(buf, overlap_combined_proj)
```

```r
  if (nrow(intersection) == 0) {
    # No overlap
    overlap_area <- 0
  } else {
    # Sum all overlapping parts
    overlap_area <- sum(st_area(intersection))
  }

  buffer_area <- st_area(buf)

  # Return proportion
  as.numeric(overlap_area) / as.numeric(buffer_area)
})

# Step 6: Add predicted overlap to countsDf
countsDf$predicted_overlap <- predicted_overlap_prop

# Step 7: Quick check
head(countsDf[, c("site", "h_index", "predicted_overlap")])
```

```
##                             site h_index predicted_overlap
## 001.ByroWA.2023 001.ByroWA.2023    0.97         0.8411216
## 007.CookIL.2023 007.CookIL.2023    1.00         0.9993222
## 008.CachUT.2023 008.CachUT.2023    1.00         0.0000000
## 009.BarnMA.2023 009.BarnMA.2023    1.00         0.8248844
## 011.BoxEUT.2024 011.BoxEUT.2024    1.00         0.0000000
## 012.OgdeUT.2024 012.OgdeUT.2024    1.00         0.4750927
```

Now, add directionality with 0 = quinq, 1 = pip, to estimate "predicted_h_index"

```r
# ---------------------------------------------
# New: Predict directional hybrid index from habitat around each sampling site
# ---------------------------------------------

# Step 1: Prepare the pipiens-only and quinque-only polygons
# (We already have poly_pipiens_ll and poly_quinque_ll, but need to transform)
poly_pipiens_proj <- st_transform(poly_pipiens_ll, crs = aea_crs)
poly_quinque_proj <- st_transform(poly_quinque_ll, crs = aea_crs)
overlap_combined_proj <- st_transform(overlap_combined, crs = aea_crs)  # already done above

# Step 2: Create pip-only and quinque-only polygons (remove overlap area)
pip_only <- st_difference(poly_pipiens_proj, overlap_combined_proj)
quinque_only <- st_difference(poly_quinque_proj, overlap_combined_proj)

# Step 3: Calculate areas for each sample buffer
library(units)  # make sure units package is loaded

predicted_h_index <- sapply(1:nrow(sample_buffers), function(i) {
  buf <- sample_buffers[i, ]

  pip_intersect <- st_intersection(buf, pip_only)
  quinque_intersect <- st_intersection(buf, quinque_only)
  overlap_intersect <- st_intersection(buf, overlap_combined_proj)

  pip_area <- if (nrow(pip_intersect) == 0) units::set_units(0, "m^2") else sum(st_area(pip_intersect))
```

```r
  quinque_area <- if (nrow(quinque_intersect) == 0) units::set_units(0, "m^2") else sum(st_area(quinque_
  overlap_area <- if (nrow(overlap_intersect) == 0) units::set_units(0, "m^2") else sum(st_area(overlap_

  total_area <- pip_area + quinque_area + overlap_area

  if (as.numeric(total_area) == 0) {
    return(NA)  # no habitat found
  } else {
    pred_h <- (pip_area + 0.5 * overlap_area) / total_area
    return(as.numeric(pred_h))  # strip units at the end
  }
})
# Step 4: Add predicted_h_index to countsDf
countsDf$predicted_h_index <- predicted_h_index

# Step 5: Quick check and save
(summary_table <- countsDf[, c("site", "latitude", "longitude", "h_index", "predicted_overlap", "predic
```

```
##                              site latitude  longitude h_index predicted_overlap
## 001.ByroWA.2023 001.ByroWA.2023 46.19300 -119.89900    0.97         0.8411216
## 007.CookIL.2023 007.CookIL.2023 42.03176  -87.93087    1.00         0.9993222
## 008.CachUT.2023 008.CachUT.2023 41.79696 -111.82005    1.00         0.0000000
## 009.BarnMA.2023 009.BarnMA.2023 41.79362  -69.99427    1.00         0.8248844
## 011.BoxEUT.2024 011.BoxEUT.2024 41.69945 -112.16386    1.00         0.0000000
## 012.OgdeUT.2024 012.OgdeUT.2024 41.20359 -112.04803    1.00         0.4750927
## 015.SaltUT.2023 015.SaltUT.2023 40.74805 -111.96788    0.38         0.4850847
## 015.SaltUT.2018 015.SaltUT.2018 40.74805 -111.96788    0.38         0.4850847
## 022.HuntNJ.2023 022.HuntNJ.2023 40.53959  -74.83462    0.89         0.4277893
## 023.SomeNJ.2023 023.SomeNJ.2023 40.53358  -74.58610    0.90         0.6953889
## 024.FortCO.2023 024.FortCO.2023 40.38428 -104.78940    1.00         0.0000000
## 025.UteTUT.2024 025.UteTUT.2024 40.32890 -109.89110    1.00         0.0000000
## 026.VernUT.2024 026.VernUT.2024 40.26180 -109.35110    1.00         0.0000000
## 028.ProvUT.2024 028.ProvUT.2024 40.20065 -111.62731    1.00         0.1866844
## 035.SuttCA.2023 035.SuttCA.2023 39.16647 -121.59845    0.72         0.9144982
## 038.RockMD.2023 038.RockMD.2023 39.05775  -77.13055    0.93         0.8748454
## 041.LincCA.2023 041.LincCA.2023 38.90447 -121.30633    0.98         0.8557477
## 054.MoabUT.2024 054.MoabUT.2024 38.59597 -109.57376    1.00         0.0000000
## 072.StGeUT.2023 072.StGeUT.2023 37.17900 -113.32000    0.00         0.3040211
## 119.PhoeAZ.2024 119.PhoeAZ.2024 33.44844 -112.07414    0.00         0.0000000
## 123.MariAZ.2023 123.MariAZ.2023 33.07362 -111.97377    0.00         0.0000000
## 124.DalFTX.2024 124.DalFTX.2024 32.42235  -96.93698    0.11         0.0000000
## 130.CollTX.2023 130.CollTX.2023 30.60044  -96.26893    0.08         0.0000000
## 131.SlidLA.2024 131.SlidLA.2024 30.32700  -89.74900    0.03         0.0000000
## 133.AnasFL.2023 133.AnasFL.2023 29.90311  -81.40074    0.00         0.0000000
## 137.MiDaFL.2023 137.MiDaFL.2023 25.80141  -80.19909    0.00         0.0000000
##                 predicted_h_index
## 001.ByroWA.2023         0.5208546
## 007.CookIL.2023         0.5003389
## 008.CachUT.2023         1.0000000
## 009.BarnMA.2023         0.5000000
## 011.BoxEUT.2024         1.0000000
## 012.OgdeUT.2024         0.7051935
## 015.SaltUT.2023         0.7010426
## 015.SaltUT.2018         0.7010426
```

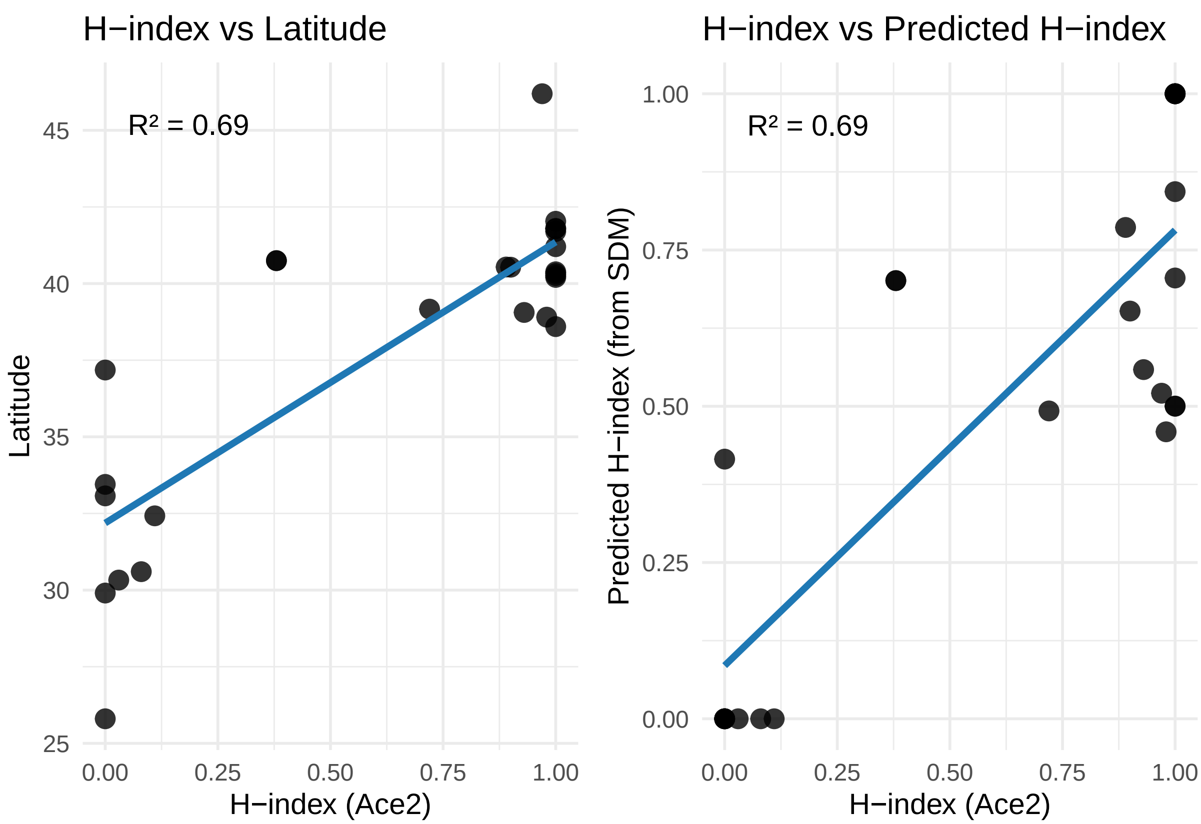
```
## 022.HuntNJ.2023        0.7861053
## 023.SomeNJ.2023        0.6523055
## 024.FortCO.2023        1.0000000
## 025.UteTUT.2024        1.0000000
## 026.VernUT.2024        1.0000000
## 028.ProvUT.2024        0.8433459
## 035.SuttCA.2023        0.4925275
## 038.RockMD.2023        0.5586669
## 041.LincCA.2023        0.4591174
## 054.MoabUT.2024               NA
## 072.StGeUT.2023        0.4154164
## 119.PhoeAZ.2024        0.0000000
## 123.MariAZ.2023        0.0000000
## 124.DalFTX.2024        0.0000000
## 130.CollTX.2023        0.0000000
## 131.SlidLA.2024        0.0000000
## 133.AnasFL.2023        0.0000000
## 137.MiDaFL.2023        0.0000000
```

```
# Save as CSV
#write.csv(summary_table, file = "../data/summary_hindex_prediction.csv", row.names = FALSE)
```

Plot h_index versus latitude, and h_index versus predicted h_index based on SDM habitat MaxEnt models:

```
library(ggplot2)
library(patchwork)

# --------------------------------
# First plot: H-index vs Latitude
# --------------------------------

# Fit linear model for H-index vs Latitude
lm_lat <- lm(latitude ~ h_index, data = countsDf)
r_squared_lat <- summary(lm_lat)$r.squared
r_squared_lat_text <- paste0("R² = ", round(r_squared_lat, 2))

p1 <- ggplot(countsDf, aes(x = h_index, y = latitude)) +
  geom_point(size = 3, alpha = 0.8, color = "black") +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE, color = "#1f78b4", linewidth = 1.2) +
  annotate("text", x = 0.05, y = max(countsDf$latitude, na.rm = TRUE) - 1, label = r_squared_lat_text, h
  theme_minimal() +
  labs(x = "H-index (Ace2)", y = "Latitude", title = "H-index vs Latitude") +
  xlim(0, 1)

# --------------------------------
# Second plot: H-index vs Predicted H-index
# --------------------------------

# Fit linear model for H-index vs Predicted H-index
lm_pred <- lm(predicted_h_index ~ h_index, data = countsDf)
r_squared_pred <- summary(lm_pred)$r.squared
r_squared_pred_text <- paste0("R² = ", round(r_squared_pred, 2))

p2 <- ggplot(countsDf, aes(x = h_index, y = predicted_h_index)) +
  #geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "grey40") +
```

```r
  geom_point(size = 3, alpha = 0.8, color = "black") +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE, color = "#1f78b4", linewidth = 1.2) +
  annotate("text", x = 0.05, y = 0.95, label = r_squared_pred_text, hjust = 0, size = 4) +
  theme_minimal() +
  labs(x = "H-index (Ace2)", y = "Predicted H-index (from SDM)",
       title = "H-index vs Predicted H-index") +
  xlim(0, 1) +
  ylim(0, 1)

# --------------------------------
# Combine side-by-side
# --------------------------------

(p1 | p2) +
  plot_layout(guides = 'collect')
```



HWE Fisher's Exact Tests

```r
library(dplyr)

# Create an empty list to store results
hwe_results <- list()

# Loop through each site
for (i in 1:nrow(countsDf)) {
  # Extract observed counts
  obs_pp <- countsDf$pp[i]
  obs_pq <- countsDf$pq[i]
```

```r
  obs_qq <- countsDf$qq[i]

  total_genotypes <- obs_pp + obs_pq + obs_qq

  # Skip if total is 0
  if (total_genotypes == 0) {
    hwe_results[[countsDf$site[i]]] <- NA
    next
  }

  # Check if there are at least two non-zero genotype classes
  non_zero_classes <- sum(c(obs_pp, obs_pq, obs_qq) > 0)

  if (non_zero_classes < 2) {
    hwe_results[[countsDf$site[i]]] <- NA
    next
  }

  # Calculate allele frequencies
  p <- (2 * obs_pp + obs_pq) / (2 * total_genotypes)
  q <- 1 - p

  # Expected counts under HWE
  exp_pp <- p^2 * total_genotypes
  exp_pq <- 2 * p * q * total_genotypes
  exp_qq <- q^2 * total_genotypes

  # Round expected counts
  exp_pp <- round(exp_pp)
  exp_pq <- round(exp_pq)
  exp_qq <- round(exp_qq)

  # Create contingency table
  table_hwe <- matrix(
    c(obs_pp, obs_pq, obs_qq,
      exp_pp, exp_pq, exp_qq),
    nrow = 2, byrow = TRUE
  )
  colnames(table_hwe) <- c("pp", "pq", "qq")
  rownames(table_hwe) <- c("Observed", "Expected")

  # Perform Fisher's exact test
  fisher_test <- fisher.test(table_hwe, simulate.p.value = TRUE, B = 1e5)

  # Save the result
  hwe_results[[countsDf$site[i]]] <- list(
    p_value = fisher_test$p.value,
    observed = c(pp = obs_pp, pq = obs_pq, qq = obs_qq),
    expected = c(pp = exp_pp, pq = exp_pq, qq = exp_qq)
  )
}

# Convert to a dataframe
```

```r
hwe_summary <- do.call(rbind, lapply(names(hwe_results), function(site) {
  res <- hwe_results[[site]]
  if (is.null(res) || all(is.na(res))) {
    return(data.frame(site = site, p_value = NA))
  } else {
    return(data.frame(site = site, p_value = res$p_value))
  }
}))

# View the result
hwe_summary
```

```
##               site   p_value
## 1  001.ByroWA.2023 1.0000000
## 2  007.CookIL.2023        NA
## 3  008.CachUT.2023        NA
## 4  009.BarnMA.2023        NA
## 5  011.BoxEUT.2024        NA
## 6  012.OgdeUT.2024        NA
## 7  015.SaltUT.2023 0.8258217
## 8  015.SaltUT.2018 1.0000000
## 9  022.HuntNJ.2023 0.6609634
## 10 023.SomeNJ.2023 1.0000000
## 11 024.FortCO.2023        NA
## 12 025.UteTUT.2024        NA
## 13 026.VernUT.2024        NA
## 14 028.ProvUT.2024        NA
## 15 035.SuttCA.2023 1.0000000
## 16 038.RockMD.2023 1.0000000
## 17 041.LincCA.2023 1.0000000
## 18 054.MoabUT.2024        NA
## 19 072.StGeUT.2023        NA
## 20 119.PhoeAZ.2024        NA
## 21 123.MariAZ.2023        NA
## 22 124.DalFTX.2024 0.6581134
## 23 130.CollTX.2023 1.0000000
## 24 131.SlidLA.2024 1.0000000
## 25 133.AnasFL.2023        NA
## 26 137.MiDaFL.2023        NA
```

Create a summary table

```r
# Merge HWE p-values into countsDf
countsDf$HWE_p_value <- hwe_summary$p_value[match(countsDf$site, hwe_summary$site)]

# Add total N per site
countsDf$N <- countsDf$pp + countsDf$pq + countsDf$qq

# Select and reorder columns
summary_sites <- countsDf %>%
  select(site, pp, pq, qq, N, latitude, longitude, h_index, predicted_overlap, predicted_h_index, HWE_p

# Select and reorder columns
```

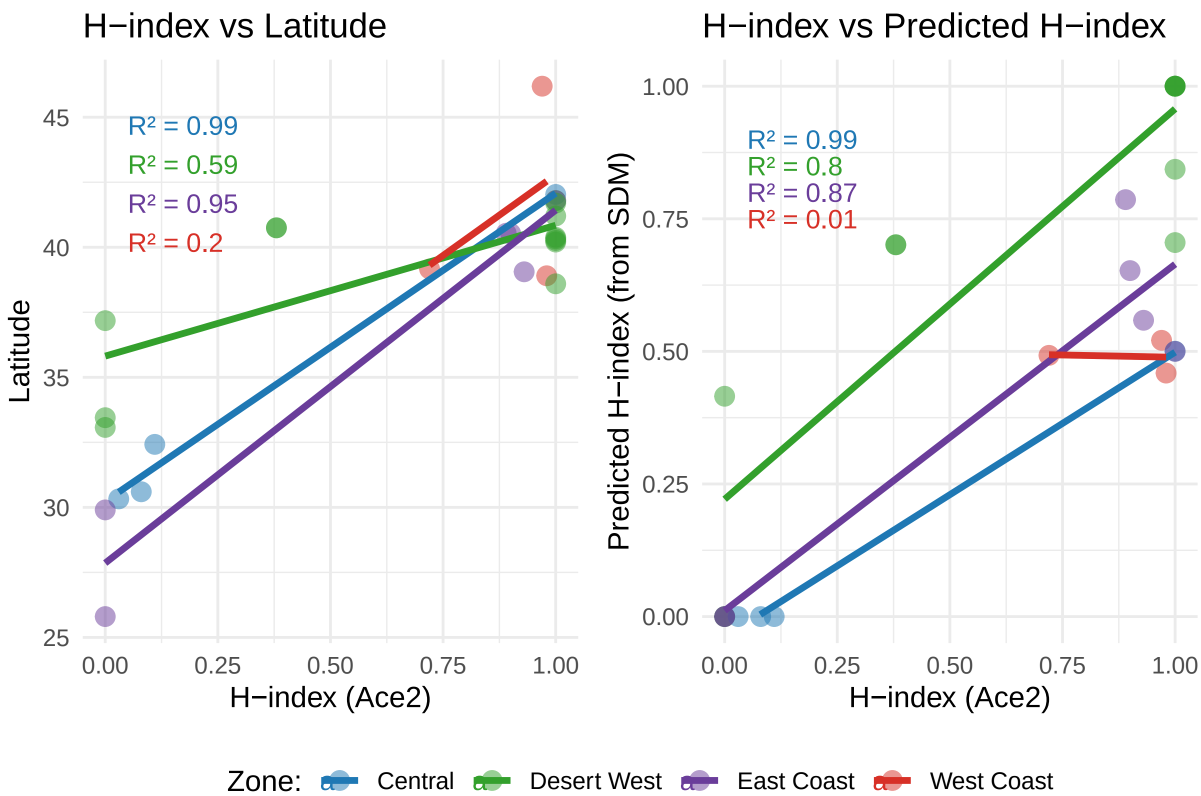
```
summary_sites <- countsDf %>%
  select(site, pp, pq, qq, N, latitude, longitude, h_index, predicted_overlap, predicted_h_index, HWE_p

# Save to CSV
#write.csv(summary_sites, file = "../data/summary_sites_hindex_overlap_HWE.csv", row.names = FALSE)

# Function to pool and run HWE
pool_hwe_test <- function(df_zone) {
  total_pp <- sum(df_zone$pp, na.rm = TRUE)
  total_pq <- sum(df_zone$pq, na.rm = TRUE)
  total_qq <- sum(df_zone$qq, na.rm = TRUE)

  total_N <- total_pp + total_pq + total_qq

  if (total_N == 0) {
    return(data.frame(total_pp, total_pq, total_qq, total_N, pooled_HWE_p_value = NA))
  }

  p <- (2 * total_pp + total_pq) / (2 * total_N)
  q <- 1 - p

  exp_pp <- round(p^2 * total_N)
  exp_pq <- round(2 * p * q * total_N)
  exp_qq <- round(q^2 * total_N)

  table_hwe <- matrix(c(total_pp, total_pq, total_qq,
                        exp_pp, exp_pq, exp_qq),
                      nrow = 2, byrow = TRUE)
  colnames(table_hwe) <- c("pp", "pq", "qq")
  rownames(table_hwe) <- c("Observed", "Expected")

  fisher_test <- fisher.test(table_hwe, simulate.p.value = TRUE, B = 1e5)

  return(data.frame(total_pp, total_pq, total_qq, total_N, pooled_HWE_p_value = fisher_test$p.value))
}

countsDf$zone <- NULL
countsDf$zone <- case_when(
  grepl("CA|OR|WA", countsDf$site) ~ "West Coast",
  grepl("NJ|MD|MA|DE|CT|NY|VA|FL", countsDf$site) ~ "East Coast",
  grepl("TX|LA|IL", countsDf$site) ~ "Central",
  grepl("UT|CO|AZ", countsDf$site) ~ "Desert West",
  #grepl("UT|CO", countsDf$site) ~ "Mountain",
  TRUE ~ "Other"
)

# Apply by zone
summary_zones <- countsDf %>%
  group_by(zone) %>%
  group_modify(~ pool_hwe_test(.x)) %>%
  ungroup()

# Save to CSV
```

```
#write.csv(summary_zones, file = "../data/summary_zones_hindex_overlap_HWE.csv", row.names = FALSE)

summary_zones
```

```
## # A tibble: 4 x 6
##   zone         total_pp total_pq total_qq total_N pooled_HWE_p_value
##   <chr>           <int>    <int>    <int>   <int>              <dbl>
## 1 Central            18        6       51      75          0.0000100
## 2 Desert West       144       13       81     238          0.0000100
## 3 East Coast         57        8       39     104          0.0000100
## 4 West Coast         84       12        1      97          1
```

```
library(ggplot2)
library(patchwork)
library(RColorBrewer)

# Plot to PDF
#pdf("../figs/hindex_vs_lat-n-pred.pdf", width = 8, height = 5)

# Define your own fixed palette
custom_colors <- c(
  "#1f78b4",  # Anna - dark blue
  "#33a02c",  # Ricei - forest green
  "#6a3d9a",  # Idas - dark purple
  "#d73027",  # Melissa - red
  "#ffb400",  # Warner - mustard
  "#1dd5c8",  # Alpine - cyan
  "#fb9a99"   # Jackson - pink
)

# Assign unique colors to each zone
countsDf$zone <- as.factor(countsDf$zone)  # or use another unique ID
n_colors <- length(unique(countsDf$zone))
color_palette <- setNames(custom_colors[1:n_colors], levels(countsDf$zone))

# ---------------------------------
# First plot: H-index vs Latitude
# ---------------------------------
library(dplyr)

# For p1 (Latitude vs H-index)
r2_lat <- countsDf %>%
  group_by(zone) %>%
  summarise(r2 = summary(lm(latitude ~ h_index))$r.squared)

# p1 with per-zone R2
p1 <- ggplot(countsDf, aes(x = h_index, y = latitude, color = zone)) +
  geom_point(size = 3, alpha = 0.5) +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE, linewidth = 1.2) +
  geom_text(data = r2_lat, aes(x = 0.05, y = max(countsDf$latitude, na.rm = TRUE) - as.numeric(factor(z
                              label = paste0("R² = ", round(r2, 2)), color = zone),
            hjust = 0, size = 3.5, inherit.aes = FALSE) +
  theme_minimal() +
  labs(x = "H-index (Ace2)", y = "Latitude", title = "H-index vs Latitude", color = "Zone:") +
```

```
  xlim(0, 1) +
  scale_color_manual(values = color_palette)

# ---------------------------------
# Second plot: H-index vs Predicted H-index
# ---------------------------------

# For p2 (Predicted H-index vs H-index)
r2_pred <- countsDf %>%
  group_by(zone) %>%
  summarise(r2 = summary(lm(predicted_h_index ~ h_index))$r.squared)


# p2 with per-zone R2
p2 <- ggplot(countsDf, aes(x = h_index, y = predicted_h_index, color = zone)) +
  geom_point(size = 3, alpha = 0.5) +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE, linewidth = 1.2) +
  geom_text(data = r2_pred, aes(x = 0.05, y = 0.95 - as.numeric(factor(zone))*0.05,
                                label = paste0("R² = ", round(r2, 2)), color = zone),
            hjust = 0, size = 3.5, inherit.aes = FALSE) +
  theme_minimal() +
  labs(x = "H-index (Ace2)", y = "Predicted H-index (from SDM)",
       title = "H-index vs Predicted H-index", color = "Zone:") +
  xlim(0, 1) +
  ylim(0, 1) +
  scale_color_manual(values = color_palette)
# ---------------------------------
# Combine side-by-side
# ---------------------------------

(p1 | p2) + plot_layout(guides = 'collect') & theme(legend.position = "bottom")
```




```
countsDf[,c("site","zone","h_index","predicted_h_index")]
```

```
##                            site        zone h_index predicted_h_index
## 001.ByroWA.2023 001.ByroWA.2023  West Coast    0.97         0.5208546
## 007.CookIL.2023 007.CookIL.2023     Central    1.00         0.5003389
## 008.CachUT.2023 008.CachUT.2023 Desert West    1.00         1.0000000
## 009.BarnMA.2023 009.BarnMA.2023  East Coast    1.00         0.5000000
## 011.BoxEUT.2024 011.BoxEUT.2024 Desert West    1.00         1.0000000
## 012.OgdeUT.2024 012.OgdeUT.2024 Desert West    1.00         0.7051935
## 015.SaltUT.2023 015.SaltUT.2023 Desert West    0.38         0.7010426
## 015.SaltUT.2018 015.SaltUT.2018 Desert West    0.38         0.7010426
## 022.HuntNJ.2023 022.HuntNJ.2023  East Coast    0.89         0.7861053
## 023.SomeNJ.2023 023.SomeNJ.2023  East Coast    0.90         0.6523055
## 024.FortCO.2023 024.FortCO.2023 Desert West    1.00         1.0000000
## 025.UteTUT.2024 025.UteTUT.2024 Desert West    1.00         1.0000000
## 026.VernUT.2024 026.VernUT.2024 Desert West    1.00         1.0000000
## 028.ProvUT.2024 028.ProvUT.2024 Desert West    1.00         0.8433459
## 035.SuttCA.2023 035.SuttCA.2023  West Coast    0.72         0.4925275
## 038.RockMD.2023 038.RockMD.2023  East Coast    0.93         0.5586669
## 041.LincCA.2023 041.LincCA.2023  West Coast    0.98         0.4591174
## 054.MoabUT.2024 054.MoabUT.2024 Desert West    1.00                NA
## 072.StGeUT.2023 072.StGeUT.2023 Desert West    0.00         0.4154164
## 119.PhoeAZ.2024 119.PhoeAZ.2024 Desert West    0.00         0.0000000
## 123.MariAZ.2023 123.MariAZ.2023 Desert West    0.00         0.0000000
## 124.DalFTX.2024 124.DalFTX.2024     Central    0.11         0.0000000
## 130.CollTX.2023 130.CollTX.2023     Central    0.08         0.0000000
## 131.SlidLA.2024 131.SlidLA.2024     Central    0.03         0.0000000
## 133.AnasFL.2023 133.AnasFL.2023  East Coast    0.00         0.0000000
## 137.MiDaFL.2023 137.MiDaFL.2023  East Coast    0.00         0.0000000
```

```r
#dev.off()
```

Let's repeat some of this with the full dataset?