Alabama Scouting Report

2022-10-21

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.1.2
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.1.2
```

```
library(gamezoneR)
library(LearnGeom)
library(hexbin)

# set a ggplot2 theme
# theme_set(
#   theme_bw() +
#     theme(plot.title = element_text(face = "bold", size = 32/.pt, hjust = 0),
#           plot.subtitle = element_text(face = "italic", size = 26/.pt),
#           plot.caption = element_text(face = "italic", size = 20/.pt),
#           strip.background = element_rect(color = "black", size = 3, linetype = "blank"),
#           strip.text = element_text(face = "bold", size = 24/.pt, hjust = 0),
#           panel.grid.minor = element_blank(),
#           panel.border = element_blank(),
#           axis.ticks = element_blank(),
#           axis.text = element_text(size = 24/.pt),
#           axis.title = element_text(face = "bold", size = 26/.pt),
#           legend.title = element_text(face = "bold", size = 26/.pt),
#           legend.text = element_text(size = 24/.pt)))
```

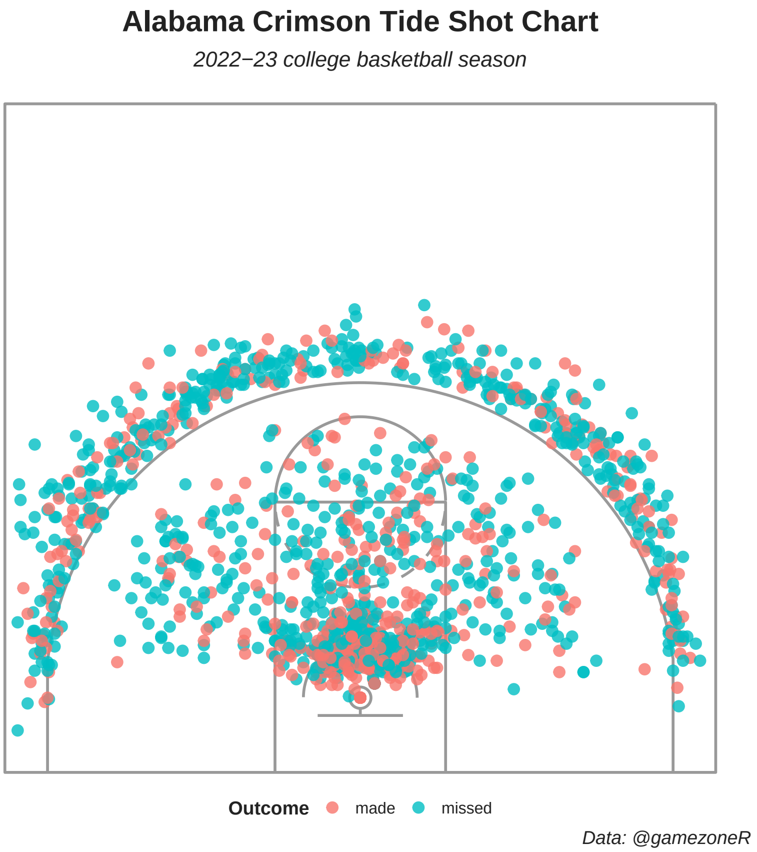
```
ala_schedule <- gamezoneR::gamezone_mbb_team_schedule(team = "Alabama", season = "2022-23")
```

```
## i Scraping 2022-23 season schedule for: Alabama
```

```
ala_pbp <- purrr::map_df(ala_schedule$game_id,
                          gamezoneR::gamezone_mbb_pbp, sub_parse = F)
```

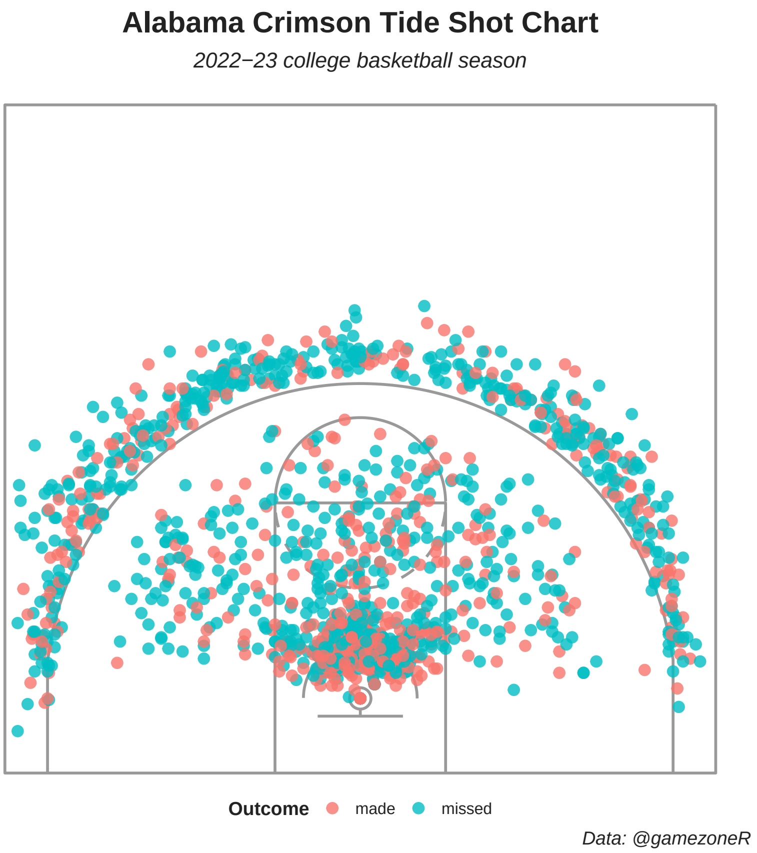
```
ala_shots <- ala_pbp %>%
  dplyr::filter(!is.na(loc_x))

gamezoneR::base_court +
  geom_point(data = ala_shots,
             aes(loc_x, loc_y, color = shot_outcome),
             alpha = 0.8) +
  theme(axis.line = element_blank(),
        axis.text= element_blank(),
        axis.ticks = element_blank(),
        axis.title = element_blank(),
        panel.background = element_blank(),
        panel.grid = element_blank(),
        plot.title = element_text(face = "bold", hjust = 0.5, size = 30/.pt, margin = margin(0, 0, 5, 0
        plot.subtitle = element_text(face = "italic", hjust = 0.5, size = 24/.pt),
        plot.caption = element_text(face = "italic", hjust = 1, size = 20/.pt, margin = margin(0, 0, 0,
        legend.spacing.x = grid::unit(0, 'cm'),
        legend.title = element_text(size = 20/.pt, face = "bold"),
        legend.text = element_text(size = 16/.pt),
        legend.margin = margin(0, 0, 0, 0),
        legend.position = 'bottom',
        legend.box.margin = margin(-35, 0, 0, 0),
        plot.margin = margin(5, 0, 5, 0)) +
  labs(title = "Alabama Crimson Tide Shot Chart",
       subtitle = "2022-23 college basketball season",
       color = "Outcome",
       caption = "Data: @gamezoneR")
```



```
ala_shots <- ala_pbp %>%
  dplyr::filter(!is.na(loc_x))
```

```
gamezoneR::base_court +
  geom_point(data = ala_shots,
             aes(loc_x, loc_y, color = shot_outcome),
             alpha = 0.8) +
  theme(axis.line = element_blank(),
        axis.text= element_blank(),
        axis.ticks = element_blank(),
        axis.title = element_blank(),
        panel.background = element_blank(),
        panel.grid = element_blank(),
        plot.title = element_text(face = "bold", hjust = 0.5, size = 30/.pt, margin = margin(0, 0, 5, 0
        plot.subtitle = element_text(face = "italic", hjust = 0.5, size = 24/.pt),
        plot.caption = element_text(face = "italic", hjust = 1, size = 20/.pt, margin = margin(0, 0, 0,
        legend.spacing.x = grid::unit(0, 'cm'),
        legend.title = element_text(size = 20/.pt, face = "bold"),
        legend.text = element_text(size = 16/.pt),
        legend.margin = margin(0, 0, 0, 0),
        legend.position = 'bottom',
        legend.box.margin = margin(-35, 0, 0, 0),
        plot.margin = margin(5, 0, 5, 0)) +
  labs(title = "Alabama Crimson Tide Shot Chart",
       subtitle = "2022-23 college basketball season",
       color = "Outcome",
       caption = "Data: @gamezoneR")
```



```
progressr::with_progress({
  pbp <- gamezoneR::load_gamezone_pbp("2021-22")
})
```

```
get_shot_angle <- function(x) {
  if(is.na(Angle(c(0, 5.25), c(25, 5.25), x))) {
    return(0)
  }else{
    return(Angle(c(0, 5.25), c(25, 5.25), x))
  }
}

euc.dist <- function(x1, x2) sqrt(sum((x1 - x2) ^ 2))

get_shot_distance <- function(x) {
  return(euc.dist(x, c(25, 5.35)))
}
```

```
shots <- pbp %>%
  dplyr::distinct() %>%
  dplyr::filter(!is.na(loc_x)) %>%
  dplyr::mutate(shot_made_numeric = dplyr::case_when(
                  is.na(shot_outcome) ~ NA_real_,
                  shot_outcome == "made" ~ 1,
                  shot_outcome == "missed" ~ 0),
                shot_value = dplyr::case_when(
                  is.na(shot_outcome) ~ NA_real_,
                  three_pt == 1 ~ 3,
                  T ~ 2),
                points = dplyr::case_when(
                  shot_made_numeric == 0 ~ 0,
                  shot_made_numeric == 1 & three_pt == 1 ~ 3,
                  shot_made_numeric == 1 & three_pt == 0 ~ 2),
                shot_angle = mapply(function(x, y) get_shot_angle(c(x, y)),
                                  loc_x, loc_y),
                shot_distance = mapply(function(x, y) get_shot_distance(c(x, y)),
                                  loc_x, loc_y),
                shot_zone = dplyr::case_when(
                  shot_distance < 6 ~ 0,
                  shot_distance >= 6 & shot_distance < 15 & shot_angle >= 0 & shot_angle <= 36 ~ 1,
                  shot_distance >= 6 & shot_distance < 15 & shot_angle > 36 & shot_angle <= 72 ~ 2,
                  shot_distance >= 6 & shot_distance < 15 & shot_angle > 72 & shot_angle <= 108 ~ 3,
                  shot_distance >= 6 & shot_distance < 15 & shot_angle > 108 & shot_angle <= 144 ~ 4,
                  shot_distance >= 6 & shot_distance < 15 & shot_angle > 144 & shot_angle <= 180 ~ 5,
                  shot_distance >= 15 & three_pt == 0 & shot_angle >= 0 & shot_angle <= 36 ~ 6,
                  shot_distance >= 15 & three_pt == 0 & shot_angle > 36 & shot_angle <= 72 ~ 7,
                  shot_distance >= 15 & three_pt == 0 & shot_angle > 72 & shot_angle <= 108 ~ 8,
                  shot_distance >= 15 & three_pt == 0 & shot_angle > 108 & shot_angle <= 144 ~ 9,
                  shot_distance >= 15 & three_pt == 0 & shot_angle > 144 & shot_angle <= 180 ~ 10,
                  three_pt == 1 & shot_angle >= 0 & shot_angle <= 36 ~ 11,
                  three_pt == 1 & shot_angle > 36 & shot_angle <= 72 ~ 12,
                  three_pt == 1 & shot_angle > 72 & shot_angle <= 108 ~ 13,
                  three_pt == 1 & shot_angle > 108 & shot_angle <= 144 ~ 14,
                  three_pt == 1 & shot_angle > 144 & shot_angle <= 180 ~ 15))
```

```
hex_bounds <- function(x, binwidth) {
  c(
    plyr::round_any(min(x), binwidth, floor) - 1e-6,
    plyr::round_any(max(x), binwidth, ceiling) + 1e-6
  )
}

calculate_hex_coords = function(shots, binwidths) {
  xbnds = hex_bounds(shots$loc_x, binwidths[1])
  xbins = diff(xbnds) / binwidths[1]
  ybnds = hex_bounds(shots$loc_y, binwidths[2])
  ybins = diff(ybnds) / binwidths[2]

  hb = hexbin(
    x = shots$loc_x,
    y = shots$loc_y,
    xbins = xbins,
    xbnds = xbnds,
    ybnds = ybnds,
    shape = ybins / xbins,
    IDs = TRUE
  )

  shots = mutate(shots, hexbin_id = hb@cID)

  hexbin_stats = shots %>%
    group_by(hexbin_id) %>%
    summarize(
      hex_attempts = n(),
      hex_pct = mean(shot_made_numeric),
      hex_points_scored = sum(shot_made_numeric * shot_value),
      hex_points_per_shot = mean(shot_made_numeric * shot_value)
    )

    hexbin_ids_to_zones = shots %>%
      group_by(hexbin_id, shot_zone) %>%
      summarize(attempts = n()) %>%
      ungroup() %>%
      arrange(hexbin_id, desc(attempts)) %>%
      group_by(hexbin_id) %>%
      filter(row_number() == 1) %>%
      select(hexbin_id, shot_zone)

  hexbin_stats = inner_join(hexbin_stats, hexbin_ids_to_zones, by = "hexbin_id")

  sx = hb@xbins / diff(hb@xbnds)
  sy = (hb@xbins * hb@shape) / diff(hb@ybnds)
  dx = 1 / (2 * sx)
  dy = 1 / (2 * sqrt(3) * sy)
  origin_coords = hexcoords(dx, dy)
```

```r
  hex_centers = hcell2xy(hb)

  hexbin_coords = bind_rows(lapply(1:hb@ncells, function(i) {
    data.frame(
      x = origin_coords$x + hex_centers$x[i],
      y = origin_coords$y + hex_centers$y[i],
      center_x = hex_centers$x[i],
      center_y = hex_centers$y[i],
      hexbin_id = hb@cell[i]
    )
  }))

  inner_join(hexbin_coords, hexbin_stats, by = "hexbin_id")
}

calculate_hexbins_from_shots = function(shots, binwidths, min_radius_factor, pps_limits) {
  if (nrow(shots) == 0) {
    return(list())
  }

  grouped_shots = group_by(shots, shot_zone)

  zone_stats = grouped_shots %>%
    summarize(
      zone_attempts = n(),
      zone_pct = mean(shot_made_numeric),
      zone_points_scored = sum(shot_made_numeric * shot_value),
      zone_points_per_shot = mean(shot_made_numeric * shot_value)
    )

  hex_data = calculate_hex_coords(shots, binwidths = binwidths)

  hex_data = hex_data %>%
    inner_join(zone_stats, by = "shot_zone")

  max_hex_attempts = max(hex_data$hex_attempts)

  hex_data = mutate(hex_data,
                    radius_factor = min_radius_factor + (1 - min_radius_factor) * log(hex_attempts + 1)
                    adj_x = center_x + radius_factor * (x - center_x),
                    adj_y = center_y + radius_factor * (y - center_y),
                    bounded_points_per_shot = pmin(pmax(zone_points_per_shot, pps_limits[1]), pps_limit

  list(hex_data = hex_data, pps_limits = pps_limits)
}
```

```r
shots <- shots %>%
  dplyr::filter(!is.na(loc_x),
                event_team == "Alabama")

hex_data <- calculate_hexbins_from_shots(shots, binwidths = c(1.5, 1.5), min_radius_factor = .25, pps_li
```

```
## ‘summarise()‘ has grouped output by ’hexbin_id’. You can override using the
## ‘.groups‘ argument.
```

```
df <- hex_data

df <- as.data.frame(df[1])
```

```
library(prismatic)
```

```
## Warning: package ’prismatic’ was built under R version 4.1.2
```

```
library(extrafont)
```

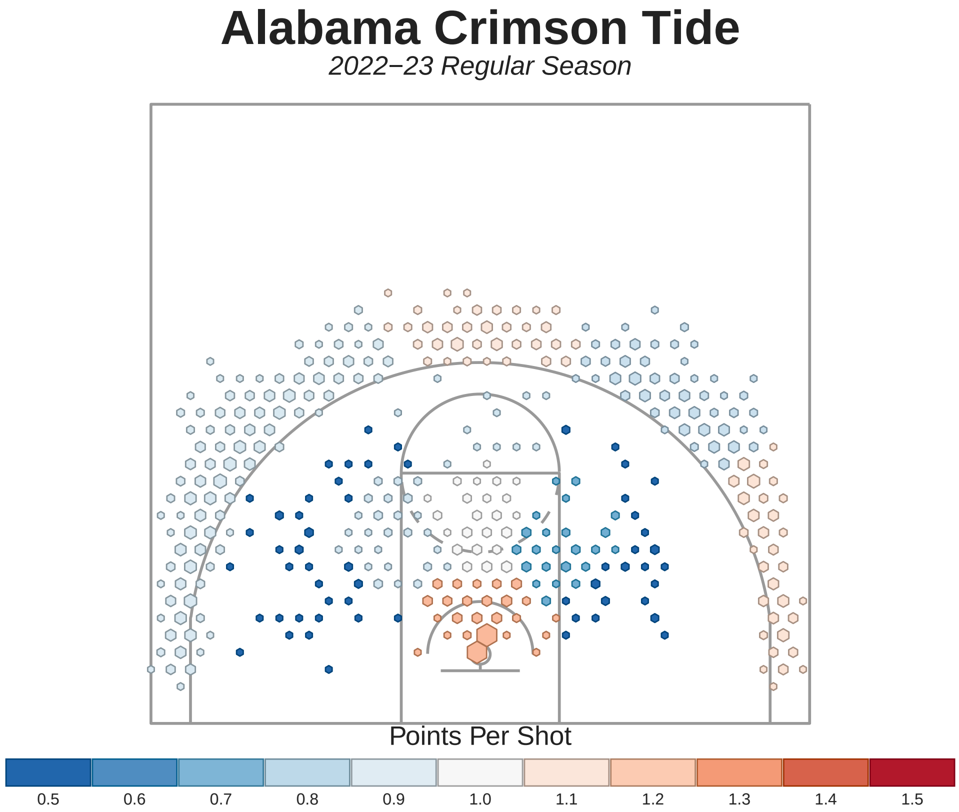
```
## Warning: package ’extrafont’ was built under R version 4.1.2

## Registering fonts with R
```

```
library(cowplot)

p <- gamezoneR::base_court +
  geom_polygon(
    data = df,
    aes(
      x = hex_data.adj_x,
      y = hex_data.adj_y,
      group = hex_data.hexbin_id,
      fill = hex_data.bounded_points_per_shot,
      color = after_scale(clr_darken(fill, .333))),
    size = .25) +
  scale_fill_distiller(direction = -1,
                       palette = "RdBu",
                       limits = c(0.5, 1.5),
                       breaks = seq(0.5, 1.5, .1),
                       labels = c("0.5", "0.6", "0.7", "0.8", "0.9", "1.0", "1.1", "1.2", "1.3", "1.4",
                       "Points Per Shot") +
  guides(fill=guide_legend(
    label.position = 'bottom',
    title.position = 'top',
    keywidth=.45,
    keyheight=.15,
    default.unit="inch",
    title.hjust = .5,
    title.vjust = 0,
    label.vjust = 3,
    nrow = 1)) +
  theme(text=element_text(size=10),
        legend.spacing.x = unit(0, 'cm'),
        legend.title=element_text(size=10),
        legend.text = element_text(size = rel(0.6)),
        legend.margin=margin(-10,0,-1,0),
        legend.position = 'bottom',
        legend.box.margin=margin(-30,0,15,0),
        plot.title = element_text(hjust = 0.5, vjust = -1, size = 18, face = "bold"),
        plot.subtitle = element_text(hjust = 0.5, size = 10, vjust = -.5),
        plot.caption = element_text(face = "italic", size = 8),
        plot.margin = margin(0, -2, 0, -2, "cm"))  +
  labs(title = "Alabama Crimson Tide",
       subtitle = "2022-23 Regular Season")

ggdraw(p) +
  theme(plot.background = element_rect(fill="white", color = NA))
```



```
player_shots <- shots %>%
  filter(!is.na(loc_x),
                shooter == "Noah Gurley")

hex_data <- calculate_hexbins_from_shots(player_shots, binwidths = c(1.5, 1.5), min_radius_factor = .25
```

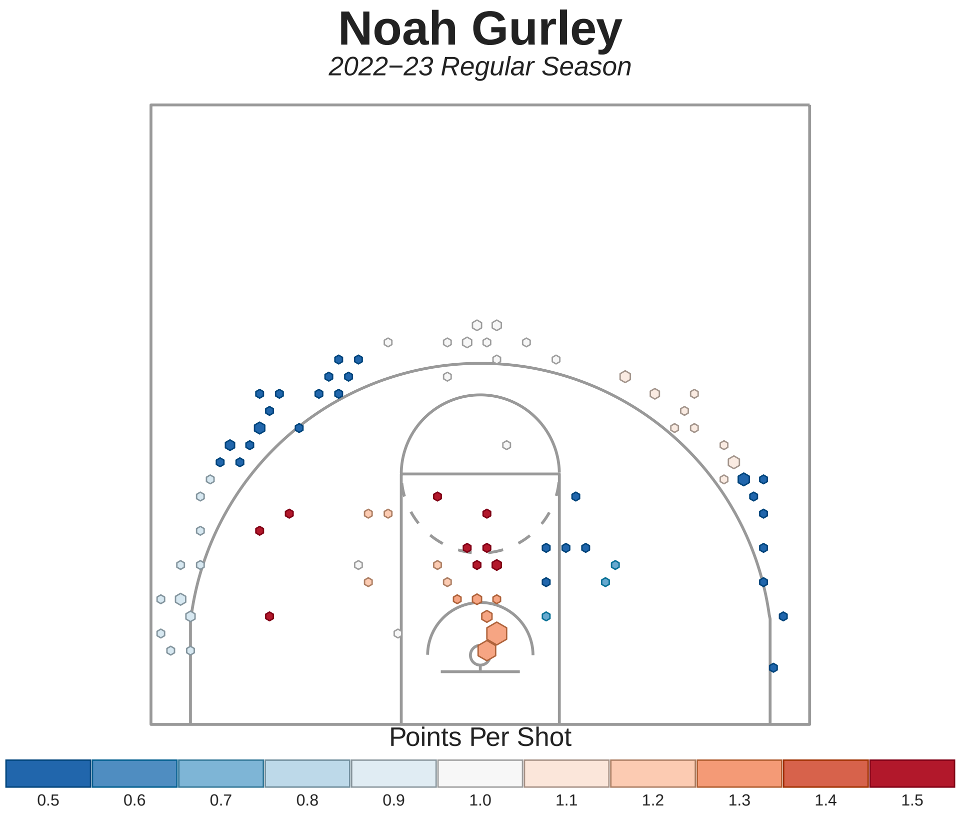
```
## `summarise()` has grouped output by 'hexbin_id'. You can override using the
## `.groups` argument.
```

```
df <- hex_data

df <- as.data.frame(df[1])
```

```r
p <- gamezoneR::base_court +
  geom_polygon(
    data = df,
    aes(
      x = hex_data.adj_x,
      y = hex_data.adj_y,
      group = hex_data.hexbin_id,
      fill = hex_data.bounded_points_per_shot,
      color = after_scale(clr_darken(fill, .333))),
    size = .25) +
  scale_fill_distiller(direction = -1,
                       palette = "RdBu",
                       limits = c(0.5, 1.5),
                       breaks = seq(0.5, 1.5, .1),
                       labels = c("0.5", "0.6", "0.7", "0.8", "0.9", "1.0", "1.1", "1.2", "1.3", "1.4",
                       "Points Per Shot") +
  guides(fill=guide_legend(
    label.position = 'bottom',
    title.position = 'top',
    keywidth=.45,
    keyheight=.15,
    default.unit="inch",
    title.hjust = .5,
    title.vjust = 0,
    label.vjust = 3,
    nrow = 1)) +
  theme(text=element_text(size=10),
        legend.spacing.x = unit(0, 'cm'),
        legend.title=element_text(size=10),
        legend.text = element_text(size = rel(0.6)),
        legend.margin=margin(-10,0,-1,0),
        legend.position = 'bottom',
        legend.box.margin=margin(-30,0,15,0),
        plot.title = element_text(hjust = 0.5, vjust = -1, size = 18, face = "bold"),
        plot.subtitle = element_text(hjust = 0.5, size = 10, vjust = -.5),
        plot.caption = element_text(face = "italic", size = 8),
        plot.margin = margin(0, -2, 0, -2, "cm"))  +
  labs(title = "Noah Gurley",
       subtitle = "2022-23 Regular Season")

ggdraw(p) +
  theme(plot.background = element_rect(fill="white", color = NA))
```



```
#ohio_schedule <- gamezoneR::gamezone_mbb_team_schedule(team = "Ohio", season = "2021-22")
```

```
#ohio_pbp <- purrr::map_df(ohio_schedule$game_id,
                            #gamezoneR::gamezone_mbb_pbp, sub_parse = F)
```