Kings Project R Code

Ian Turner

2025-10-16

Setup: Libraries & Global Options

```r
library(jsonlite)
library(dplyr)
library(lubridate)
library(MASS)
library(pROC)
library(caret)
library(ggplot2)
library(scatterplot3d)
library(tidyr)
library(shiny)
library(DT)
library(plotly)
```

Parse JSON CSV & Load Data

```r
convert_json_to_csv <- function(in_file, out_file) {
  txt <- readLines(in_file, warn = FALSE, encoding = "UTF-8")
  txt <- paste(txt, collapse = "\n")
  objs <- unlist(regmatches(txt, gregexpr("\\{[\\s\\S]*?\\}", txt, perl = TRUE)))
  if (length(objs) == 0) stop("No JSON objects found.")
  json_arr <- paste0("[", paste(objs, collapse = ","), "]")
  json_arr <- gsub('(:)\\s*-\\s*(?=[,}])', '\\1 null', json_arr, perl = TRUE)
  if (!jsonlite::validate(json_arr)) stop("JSON invalid after cleanup.")
  dat <- fromJSON(json_arr, flatten = TRUE)
  write.csv(dat, out_file, row.names = FALSE)
  cat(sprintf("%s → %s (%d rows, %d cols)\n", in_file, out_file, nrow(dat), ncol(dat)))
}

convert_json_to_csv("nba_box_player_season.json", "nba_data.csv")
```

```
## nba_box_player_season.json → nba_data.csv (1685 rows, 55 cols)
```

```r
nba_data <- read.csv("nba_data.csv", header = TRUE)

convert_json_to_csv("international_box_player_season.json",
                    "international_data.csv")
```

```
## international_box_player_season.json → international_data.csv (3370 rows, 52 cols)
```

```
international_data <- read.csv("international_data.csv")

convert_json_to_csv("player.json", "player_data.csv")
```

```
## player.json → player_data.csv (1663 rows, 3 cols)
```

```
player_data <- read.csv("player_data.csv")
```

Normalize Player Names (Title Case)

```
# Capitalize each name properly
international_data$first_name <- tools::toTitleCase(tolower(international_data$first_name))
international_data$last_name  <- tools::toTitleCase(tolower(international_data$last_name))

player_data$first_name <- tools::toTitleCase(tolower(player_data$first_name))
player_data$last_name  <- tools::toTitleCase(tolower(player_data$last_name))
```

Aggregate Seasons and Player Totals & Weighted Averages

```
wmean_safe <- function(x, w) {
  ok <- is.finite(x) & is.finite(w) & w > 0
  if (!any(ok)) return(NA_real_)
  weighted.mean(x[ok], w[ok])
}

# totals to SUM
totals_cols <- c(
  "games","starts","minutes","points",
  "two_points_made","two_points_attempted",
  "three_points_made","three_points_attempted",
  "free_throws_made","free_throws_attempted",
  "blocked_shot_attempts","offensive_rebounds","defensive_rebounds",
  "assists","screen_assists","turnovers","steals","deflections",
  "loose_balls_recovered","blocked_shots","personal_fouls","personal_fouls_drawn",
  "offensive_fouls","charges_drawn","technical_fouls","flagrant_fouls","ejections",
  "points_off_turnovers","points_in_paint","second_chance_points","fast_break_points",
  "possessions", "estimated_possessions", "team_possessions"
)

wa_cols <- names(international_data)[41:52]

player_intl_totals <- international_data %>%
  group_by(first_name, last_name) %>%
  mutate(w_games = games) %>%                       # <- preserve per-season weights
  summarise(
    across(all_of(totals_cols), ~ sum(.x, na.rm = TRUE)),
    across(all_of(wa_cols), ~ wmean_safe(.x, w_games), .names = "{.col}_wa"),
    .groups = "drop"
  ) %>%
  mutate(across(where(is.numeric), ~ round(.x, 2))) %>%
  arrange(last_name, first_name)
```

Add Age & NBA Experience (as of 2021-12-31)

```r
# helper so joins actually match
fmt <- function(x) tools::toTitleCase(tolower(x))

# --- standardize names in all three sources ---
player_intl_totals_std <- player_intl_totals %>%
  mutate(first_name = fmt(first_name),
         last_name  = fmt(last_name))

player_data_std <- player_data %>%
  mutate(first_name = fmt(first_name),
         last_name  = fmt(last_name),
         birth_date = ymd(birth_date))

nba_players_lookup <- nba_data %>%
  mutate(first_name = fmt(first_name),
         last_name  = fmt(last_name)) %>%
  distinct(first_name, last_name) %>%
  mutate(nba_experience = TRUE)

# Freeze time at end of 2021
ref_date <- ymd("2021-12-31")

player_intl_totals_final <- player_intl_totals_std %>%
  left_join(
    player_data_std %>% dplyr::select(first_name, last_name, birth_date),
    by = c("first_name", "last_name")
  ) %>%
  mutate(
    birth_date = as_date(birth_date),
    age = if_else(
      !is.na(birth_date),
      floor(time_length(interval(birth_date, ref_date), "years")),
      NA_integer_
    )
  ) %>%
  relocate(age, .after = last_name) %>%
  left_join(nba_players_lookup, by = c("first_name", "last_name")) %>%
  mutate(
    nba_experience = coalesce(nba_experience, FALSE) %>% as.logical()
  ) %>%
  relocate(nba_experience, .after = age)
```

Convert Totals to Per-Game Rates

```r
intl_per_game <- player_intl_totals_final %>%
  mutate(across(
    c(minutes, points, two_points_made, two_points_attempted,
      three_points_made, three_points_attempted, free_throws_made,
      free_throws_attempted, blocked_shot_attempts, offensive_rebounds,
      defensive_rebounds, assists, screen_assists, turnovers, steals,
      deflections, loose_balls_recovered, blocked_shots, personal_fouls,
      personal_fouls_drawn, offensive_fouls, charges_drawn, technical_fouls,
      flagrant_fouls, ejections, points_off_turnovers,
      second_chance_points, fast_break_points, possessions, estimated_possessions,
      team_possessions),
    ~ if_else(games > 0, .x / games, NA_real_)
  )) %>%
  mutate(across(where(is.numeric), ~ round(.x, 2)))
```

Attach Most-Recent Season (Recency Key)

```
recent_season <- international_data %>%
  group_by(first_name, last_name) %>%
  summarise(most_recent_year = max(season, na.rm = TRUE), .groups = "drop")

intl_per_game <- intl_per_game %>%
  left_join(recent_season, by = c("first_name", "last_name")) %>%
  relocate(most_recent_year, .after = nba_experience)
```

Export Master Table (intl_per_game.csv)

```
intl_per_game
```

```
## # A tibble: 1,473 x 52
##    first_name last_name     age nba_experience most_recent_year games starts
##    <chr>      <chr>       <dbl> <lgl>                     <int> <dbl>  <dbl>
##  1 Zanotti    Abalde         27 FALSE                      2021    12      5
##  2 Vezenkov   Abdul-Wahad    21 FALSE                      2021    17      4
##  3 Sead       Abdur-Rahim    24 FALSE                      2021    18      3
##  4 Markel     Abrines        23 FALSE                      2021     8      3
##  5 Cissoko    Abromaitis     29 FALSE                      2019    65     10
##  6 Morse      Acie           20 TRUE                       2020    42     24
##  7 Patricio   Adam           29 FALSE                      2017    10      4
##  8 Devecchi   Adams          20 FALSE                      2021    12      0
##  9 Daryl      Adel           31 TRUE                       2016    61     50
## 10 Djurisic   Adiguzel       33 TRUE                       2018    69     45
## # i 1,463 more rows
## # i 45 more variables: minutes <dbl>, points <dbl>, two_points_made <dbl>,
## #   two_points_attempted <dbl>, three_points_made <dbl>,
## #   three_points_attempted <dbl>, free_throws_made <dbl>,
## #   free_throws_attempted <dbl>, blocked_shot_attempts <dbl>,
## #   offensive_rebounds <dbl>, defensive_rebounds <dbl>, assists <dbl>,
## #   screen_assists <dbl>, turnovers <dbl>, steals <dbl>, deflections <dbl>, ...
```

```
write.csv(intl_per_game, "full_international_data.csv", row.names = FALSE)
```

Logistic Regression: Train/Test, Stepwise AIC, Evaluate, Score Full Dataset

```r
set.seed(42)

# ---- 1) Data prep ----

df <- intl_per_game %>%
  mutate(nba_experience = as.numeric(nba_experience == TRUE)) %>%
  dplyr::select(nba_experience, 3:22, 37:51, -20, -5)


# Replace NAs with column means
df <- df %>%
  mutate(across(where(is.numeric), ~ ifelse(is.na(.x), mean(.x, na.rm = TRUE), .x)))

# ---- 2) Train/test split ----
set.seed(42)
train_index <- createDataPartition(df$nba_experience, p = 0.8, list = FALSE)
train <- df[train_index, ]
test  <- df[-train_index, ]

# ---- 3) Logistic regression + stepwise variable selection ----
base_model <- glm(nba_experience ~ ., data = train, family = binomial)

# AIC-based stepwise selection
step_model <- stepAIC(base_model, direction = "both", trace = FALSE)
summary(step_model)
```

```
##
## Call:
## glm(formula = nba_experience ~ age + games + starts + minutes +
##     two_points_made + two_points_attempted + three_points_made +
##     three_points_attempted + free_throws_made + free_throws_attempted +
##     assists + steals + possessions + team_possessions + usage_percentage_wa +
##     offensive_rebounding_percentage_wa + total_rebounding_percentage_wa +
##     block_percentage_wa + internal_box_plus_minus_wa, family = binomial,
##     data = train)
##
## Coefficients:
##                                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)                        -5.0218565  0.7304054  -6.875 6.18e-12 ***
## age                                 0.0281903  0.0163407   1.725 0.084500 .
## games                               0.0218101  0.0042419   5.142 2.72e-07 ***
## starts                             -0.0155268  0.0061935  -2.507 0.012178 *
## minutes                             0.3255053  0.1315630   2.474 0.013356 *
## two_points_made                    -0.6663397  0.2487127  -2.679 0.007381 **
## two_points_attempted                0.5757929  0.1453315   3.962 7.43e-05 ***
## three_points_made                  -1.0507758  0.3883245  -2.706 0.006812 **
## three_points_attempted              0.5420426  0.1621109   3.344 0.000827 ***
## free_throws_made                   -0.6259636  0.2997058  -2.089 0.036744 *
## free_throws_attempted               0.6562010  0.2443809   2.685 0.007250 **
## assists                            -0.1603808  0.0839710  -1.910 0.056139 .
## steals                             -0.6625899  0.2697931  -2.456 0.014052 *
## possessions                        -0.1795976  0.0735322  -2.442 0.014588 *
## team_possessions                    0.0007029  0.0003913   1.796 0.072437 .
## usage_percentage_wa                  0.0480293  0.0231366   2.076 0.037903 *
## offensive_rebounding_percentage_wa -0.0836314  0.0300049  -2.787 0.005316 **
## total_rebounding_percentage_wa      0.0806175  0.0263396   3.061 0.002208 **
## block_percentage_wa                 0.0845886  0.0417944   2.024 0.042978 *
## internal_box_plus_minus_wa          0.0449792  0.0232534   1.934 0.053075 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1420.0  on 1178  degrees of freedom
## Residual deviance: 1073.1  on 1159  degrees of freedom
## AIC: 1113.1
##
## Number of Fisher Scoring iterations: 5
```

```
# ---- 4) Predictions and NBA probability scores ----
test$nba_prob <- predict(step_model, newdata = test, type = "response")

# ---- 5) Model evaluation ----
roc_obj <- roc(test$nba_experience, test$nba_prob)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
auc_val <- auc(roc_obj)
cat("AUC =", auc_val, "\n")
```

```
## AUC = 0.7384291
```

```
# Classification at 0.5 threshold
test$predicted_class <- ifelse(test$nba_prob >= 0.5, 1, 0)

confusion <- confusionMatrix(
  factor(test$predicted_class),
  factor(test$nba_experience),
  positive = "1"
)
print(confusion)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 185  62
##          1  18  29
##
##                Accuracy : 0.7279
##                  95% CI : (0.6732, 0.7779)
##     No Information Rate : 0.6905
##     P-Value [Acc > NIR] : 0.09155
##
##                   Kappa : 0.2654
##
##  Mcnemar's Test P-Value : 1.528e-06
##
```

```
##             Sensitivity : 0.31868
##             Specificity : 0.91133
##          Pos Pred Value : 0.61702
##          Neg Pred Value : 0.74899
##              Prevalence : 0.30952
##          Detection Rate : 0.09864
##    Detection Prevalence : 0.15986
##       Balanced Accuracy : 0.61501
##
##        'Positive' Class : 1
##
```

```
# ---- 7) Add NBA probability score to full dataset ----
intl_per_game <- intl_per_game %>%
  mutate(
    nba_probability = predict(step_model, newdata = df, type = "response")
  )
```

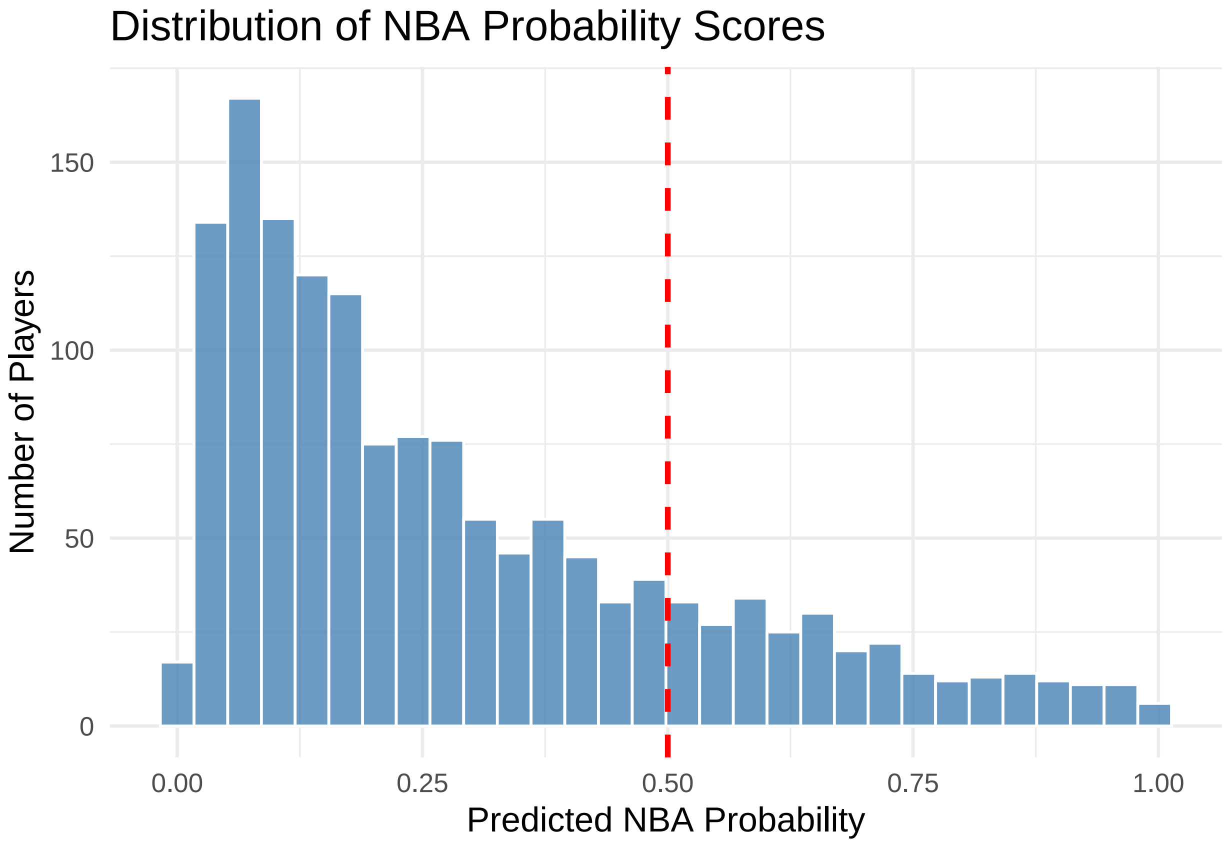
High-Prob Non-NBA Candidates (False Positives)

```
# Top non-NBA players by predicted probability
false_positives <- intl_per_game %>%
  filter(nba_experience == 0, most_recent_year >= 2019) %>%                 # not actually in NBA
  arrange(desc(nba_probability)) %>%              # sort by model score
  dplyr::select(first_name, last_name, age, nba_probability, everything())

false_positives
```

```
## # A tibble: 903 x 53
##    first_name last_name      age nba_probability nba_experience most_recent_year
##    <chr>      <chr>        <dbl>           <dbl> <lgl>                     <int>
##  1 Reynolds   Smith-Rivera    33           0.967 FALSE                      2021
##  2 Wolkowyski Carlisle        38           0.933 FALSE                      2019
##  3 Cotton     Clyburn         39           0.906 FALSE                      2021
##  4 Emmett     Kaman           39           0.857 FALSE                      2019
##  5 Omic       Tomas           30           0.809 FALSE                      2020
##  6 Parrillo   Marinkovic      29           0.803 FALSE                      2021
##  7 Tanoulis   Graham          32           0.802 FALSE                      2021
##  8 Burks      Eddie           31           0.802 FALSE                      2020
##  9 Bouquet    Konchar         31           0.769 FALSE                      2021
## 10 Tom        Cacok           33           0.754 FALSE                      2021
## # i 893 more rows
## # i 47 more variables: games <dbl>, starts <dbl>, minutes <dbl>, points <dbl>,
## #   two_points_made <dbl>, two_points_attempted <dbl>, three_points_made <dbl>,
## #   three_points_attempted <dbl>, free_throws_made <dbl>,
## #   free_throws_attempted <dbl>, blocked_shot_attempts <dbl>,
## #   offensive_rebounds <dbl>, defensive_rebounds <dbl>, assists <dbl>,
## #   screen_assists <dbl>, turnovers <dbl>, steals <dbl>, deflections <dbl>, ...
```

Plot: NBA Probability Distribution (Threshold = 0.5)

```r
ggplot(intl_per_game, aes(x = nba_probability)) +
  geom_histogram(
    bins = 30,
    fill = "steelblue",
    color = "white",
    alpha = 0.8
  ) +
  geom_vline(xintercept = 0.5, linetype = "dashed", color = "red", linewidth = 1) +
  labs(
    title = "Distribution of NBA Probability Scores",
    x = "Predicted NBA Probability",
    y = "Number of Players"
  ) +
  annotate("text", x = 0.52, y = Inf, label = "0.5 Threshold", vjust = -1, hjust = 0, color = "red", si
  theme_minimal(base_size = 13)
```



Standardize Features (z-Scores)

```r
intl_scaled <- intl_per_game %>%
  mutate(across(
    c(points, usage_percentage_wa, true_shooting_percentage_wa,
      three_points_made, three_point_attempt_rate_wa,
      assists, assist_percentage_wa, turnovers,
      offensive_rebounding_percentage_wa, defensive_rebounding_percentage_wa, total_rebounding_percentag
      blocked_shots, block_percentage_wa, defensive_rebounds),
    ~ scale(.) %>% as.numeric()
  ))
```

Archetype Coefficients: Scorer (Points + TS%)

```r
model_interact <- lm(
  internal_box_plus_minus_wa ~ scale(points) +scale(true_shooting_percentage_wa),
  data = intl_per_game
)

summary(model_interact)
```

```
##
## Call:
## lm(formula = internal_box_plus_minus_wa ~ scale(points) + scale(true_shooting_percentage_wa),
##     data = intl_per_game)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.894  -1.676   0.188   2.098  41.726
##
## Coefficients:
##                                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)                        -1.8228     0.1106  -16.48   <2e-16 ***
## scale(points)                       1.5129     0.1250   12.11   <2e-16 ***
## scale(true_shooting_percentage_wa)  3.5084     0.1232   28.47   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.178 on 1428 degrees of freedom
##   (42 observations deleted due to missingness)
## Multiple R-squared:  0.5123, Adjusted R-squared:  0.5116
## F-statistic:   750 on 2 and 1428 DF,  p-value: < 2.2e-16
```

Compute Scorer Score & Percentile

```r
intl_per_game <- intl_per_game %>%
  mutate(
    scorer_score = 1.51 * as.numeric(scale(points)) +
                   3.51 * as.numeric(scale(true_shooting_percentage_wa)),

    scorer_percentile = 100 * round(percent_rank(scorer_score), 3)
  )
```

Archetype Coefficients: Shot Blocker (Blocks - Fouls)

```r
model_interact <- lm(
  internal_box_plus_minus_wa ~ scale(blocked_shots) + scale(-personal_fouls),
  data = intl_per_game
)
```

```
summary(model_interact)
```

```
##
## Call:
## lm(formula = internal_box_plus_minus_wa ~ scale(blocked_shots) +
##     scale(-personal_fouls), data = intl_per_game)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -60.383  -2.071   0.472   2.983  34.597
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)             -1.8658     0.1446 -12.903  < 2e-16 ***
## scale(blocked_shots)     1.2021     0.1517   7.922 4.66e-15 ***
## scale(-personal_fouls)  -1.9236     0.1617 -11.896  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.461 on 1428 degrees of freedom
##   (42 observations deleted due to missingness)
## Multiple R-squared:  0.1667, Adjusted R-squared:  0.1655
## F-statistic: 142.8 on 2 and 1428 DF,  p-value: < 2.2e-16
```

Archetype Coefficients: Shot Blocker (Blocks - Fouls)

```
intl_per_game <- intl_per_game %>%
  mutate(
    shot_blocker_score =
      as.numeric(scale(blocked_shots))       * coef(model_interact)[["scale(blocked_shots)"]] +
      as.numeric(scale(-personal_fouls))      * coef(model_interact)[["scale(-personal_fouls)"]],
    shot_blocker_percentile = 100 * round(percent_rank(shot_blocker_score), 3)
  )
```

Archetype Coefficients: Rebounder (OREB + DREB)

```
model_rebounder_pct <- lm(
  internal_box_plus_minus_wa ~
    scale(offensive_rebounds) +
    scale(defensive_rebounds),
  data = intl_per_game
)
summary(model_rebounder_pct)
```

```
##
## Call:
## lm(formula = internal_box_plus_minus_wa ~ scale(offensive_rebounds) +
##     scale(defensive_rebounds), data = intl_per_game)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -62.388  -1.948   0.612   2.904  33.421
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 -1.8625     0.1413 -13.179  < 2e-16 ***
## scale(offensive_rebounds)    0.7941     0.1876   4.233 2.45e-05 ***
## scale(defensive_rebounds)    2.1561     0.1911  11.280  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.34 on 1428 degrees of freedom
##   (42 observations deleted due to missingness)
## Multiple R-squared:  0.203,  Adjusted R-squared:  0.2019
## F-statistic: 181.9 on 2 and 1428 DF,  p-value: < 2.2e-16
```

```
intl_per_game <- intl_per_game %>%
  mutate(
    rebounder_score = 0.79 * scale(offensive_rebounds) +
                      2.16 * scale(defensive_rebounds),
    rebounder_percentile = 100 * round(percent_rank(rebounder_score), 3)
  )
```

Archetype Coefficients: Facilitator (AST - TOV%) & Score

```
intl_per_game <- intl_per_game %>%
  mutate(
    assist_to_turnover_ratio = ifelse(turnovers > 0, assists / turnovers, NA)
  )

model_facilitator <- lm(
  internal_box_plus_minus_wa ~
    scale(assists) +
    scale(turnover_percentage_wa),
  data = intl_per_game
)
summary(model_facilitator)
```

```
##
## Call:
## lm(formula = internal_box_plus_minus_wa ~ scale(assists) + scale(turnover_percentage_wa),
##     data = intl_per_game)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -56.242  -2.151   0.592   2.987  27.043
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    -1.8933     0.1456  -13.00   <2e-16 ***
## scale(assists)                  1.8382     0.1472   12.49   <2e-16 ***
## scale(turnover_percentage_wa)  -2.2669     0.1871  -12.12   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 5.496 on 1428 degrees of freedom
##   (42 observations deleted due to missingness)
## Multiple R-squared:  0.1559, Adjusted R-squared:  0.1547
## F-statistic: 131.8 on 2 and 1428 DF,  p-value: < 2.2e-16
```

```
intl_per_game <- intl_per_game %>%
  mutate(
    facilitator_score =
      1.84 * as.numeric(scale(assists)) -      # from model coefficient
      2.27 * as.numeric(scale(turnover_percentage_wa)),  # inverse relationship
    facilitator_percentile = 100 * round(percent_rank(facilitator_score), 3)
  )
```

Archetype Coefficients: Floor Spacer (3PM + TS%) & Score

```
intl_per_game <- intl_per_game %>%
  mutate(
    three_point_percentage = ifelse(three_points_attempted > 0,
                                    three_points_made / three_points_attempted, NA),

    floor_spacer_score =
      1.25* as.numeric(scale(three_points_made)) +
      3 * as.numeric(scale(true_shooting_percentage_wa)),

    floor_spacer_percentile = 100 * round(percent_rank(floor_spacer_score), 3)
  )
model_3p <- lm(
  internal_box_plus_minus_wa ~
    scale(three_points_made) +  scale(true_shooting_percentage_wa),
  data = intl_per_game
)
summary(model_3p)
```

```
##
## Call:
## lm(formula = internal_box_plus_minus_wa ~ scale(three_points_made) +
##     scale(true_shooting_percentage_wa), data = intl_per_game)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -50.047  -1.708   0.324   2.199  41.573
##
## Coefficients:
##                                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)                         -1.7731     0.1143 -15.510  < 2e-16 ***
## scale(three_points_made)             0.7742     0.1181   6.557 7.66e-11 ***
## scale(true_shooting_percentage_wa)   3.9373     0.1196  32.924  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.322 on 1428 degrees of freedom
##   (42 observations deleted due to missingness)
## Multiple R-squared:  0.478,  Adjusted R-squared:  0.4772
## F-statistic: 653.7 on 2 and 1428 DF,  p-value: < 2.2e-16
```

Archetype Coefficients: Post Scorer (2P%, 2PA, FT%, FTA, PF Drawn)

```
# Create efficiency variables
intl_per_game <- intl_per_game %>%
  mutate(
    two_point_percentage = ifelse(two_points_attempted > 0,
                                  two_points_made / two_points_attempted, NA),
    free_throw_percentage = ifelse(free_throws_attempted > 0,
                                   free_throws_made / free_throws_attempted, NA)
  )

# Build the linear model
model_post_scorer_final <- lm(
  internal_box_plus_minus_wa ~
    scale(two_point_percentage) +
    scale(two_points_attempted) +
    scale(free_throws_attempted) +
    scale(free_throw_percentage) +
    scale(personal_fouls_drawn),
  data = intl_per_game
)

summary(model_post_scorer_final)
```

```
##
## Call:
## lm(formula = internal_box_plus_minus_wa ~ scale(two_point_percentage) +
##     scale(two_points_attempted) + scale(free_throws_attempted) +
##     scale(free_throw_percentage) + scale(personal_fouls_drawn),
##     data = intl_per_game)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -23.7629  -1.8493   0.1438   2.2070  11.7037
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  -1.26907    0.09832 -12.907  < 2e-16 ***
## scale(two_point_percentage)   2.16661    0.12198  17.762  < 2e-16 ***
## scale(two_points_attempted)   0.24150    0.15079   1.602 0.109489
## scale(free_throws_attempted)  0.05778    0.25900   0.223 0.823513
## scale(free_throw_percentage)  0.63197    0.09996   6.322 3.52e-10 ***
## scale(personal_fouls_drawn)   0.99115    0.26790   3.700 0.000225 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.533 on 1322 degrees of freedom
##   (145 observations deleted due to missingness)
## Multiple R-squared:  0.2888, Adjusted R-squared:  0.2861
## F-statistic: 107.4 on 5 and 1322 DF,  p-value: < 2.2e-16
```

Compute Post Scorer Score & Percentile

```r
intl_per_game <- intl_per_game %>%
  mutate(
    post_scorer_score =
      2.17 * as.numeric(scale(two_point_percentage)) +
      0.24 * as.numeric(scale(two_points_attempted)) +
      0.63 * as.numeric(scale(free_throw_percentage)) +
      0.99 * as.numeric(scale(personal_fouls_drawn)),

    post_scorer_percentile = 100 * round(percent_rank(post_scorer_score), 3)
  )
```

Tiering & Top-3 Archetype Labels per Player

```r
get_tier <- function(x) {
  case_when(
    x >= 95 ~ "Elite",
    x >= 85 ~ "Great",
    x >= 70 ~ "Good",
    x >= 50 ~ "Above Average",
    x >= 25 ~ "Below Average",
    TRUE ~ "Poor"
  )
}

# Apply tiers for each archetype
intl_per_game <- intl_per_game %>%
  mutate(
    scorer_tier = get_tier(scorer_percentile),
    post_scorer_tier = get_tier(post_scorer_percentile),
    rebounder_tier = get_tier(rebounder_percentile),
    facilitator_tier = get_tier(facilitator_percentile),
    shot_blocker_tier = get_tier(shot_blocker_percentile),
    floor_spacer_tier = get_tier(floor_spacer_percentile)
  )

# Create a summary dataframe with core player info + percentiles
player_archetypes <- intl_per_game %>%
  dplyr::select(first_name, last_name, age, nba_experience,
         scorer_percentile, post_scorer_percentile, rebounder_percentile,
         facilitator_percentile, shot_blocker_percentile, floor_spacer_percentile,
         scorer_tier, post_scorer_tier, rebounder_tier,
         facilitator_tier, shot_blocker_tier, floor_spacer_tier, games)

# Compute each player's top 3 archetypes by percentile
player_archetypes <- player_archetypes %>%
  rowwise() %>%
  mutate(
    # Get the archetype names and scores
    archetypes = list(c("Scorer", "Post Scorer", "Rebounder",
                        "Facilitator", "Shot Blocker", "Floor Spacer")),
    scores = list(c(scorer_percentile, post_scorer_percentile, rebounder_percentile,
                    facilitator_percentile, shot_blocker_percentile, floor_spacer_percentile)),

    # Order top 3 by percentile
    top3_indices = list(order(unlist(scores), decreasing = TRUE)[1:3]),
    top3_archetypes = list(unlist(archetypes)[unlist(top3_indices)]),
    top3_scores = list(unlist(scores)[unlist(top3_indices)]),

    # Extract top 3 + labels
    archetype_1 = top3_archetypes[[1]],
    archetype_2 = top3_archetypes[[2]],
    archetype_3 = top3_archetypes[[3]],
    archetype_1_label = paste(get_tier(top3_scores[[1]]), archetype_1),
    archetype_2_label = paste(get_tier(top3_scores[[2]]), archetype_2),
    archetype_3_label = paste(get_tier(top3_scores[[3]]), archetype_3)
  ) %>%
  ungroup() %>%
  dplyr::select(first_name, last_name, age, nba_experience,
         archetype_1_label, archetype_2_label, archetype_3_label,
         scorer_percentile:floor_spacer_percentile)
player_archetypes
```

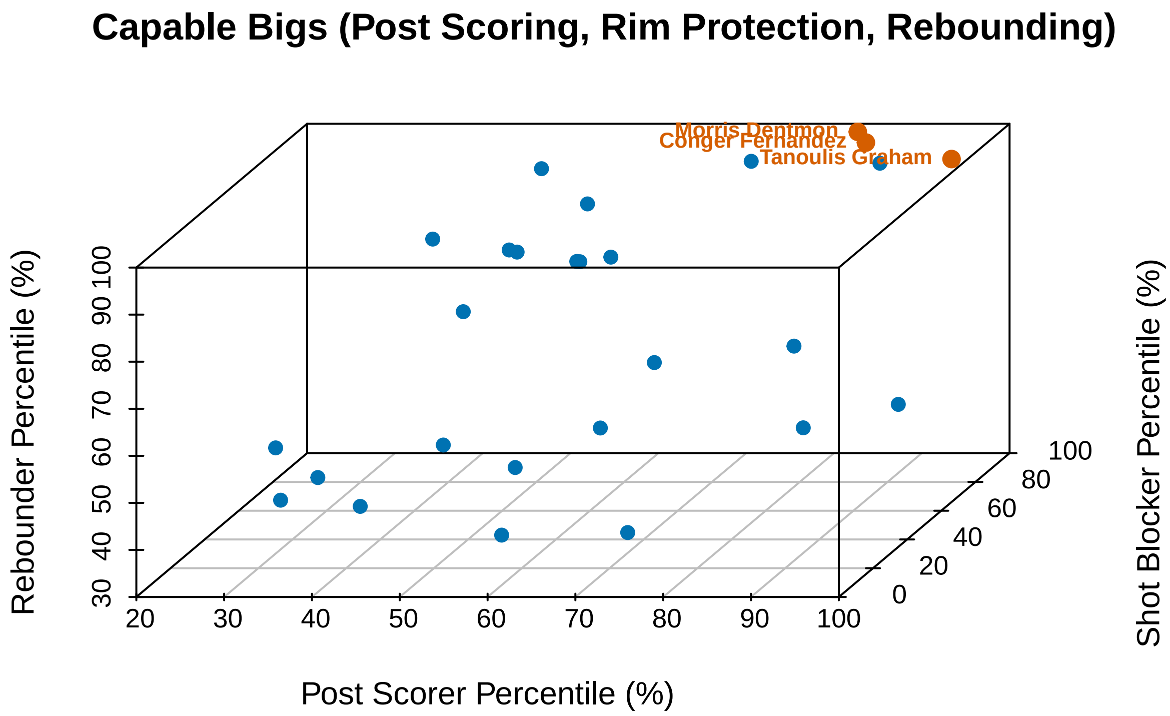
```
## # A tibble: 1,473 x 13
##    first_name last_name   age nba_experience archetype_1_label archetype_2_label
##    <chr>      <chr>     <dbl> <lgl>          <chr>             <chr>
##  1 Zanotti    Abalde       27 FALSE          Above Average Po~ Above Average Fa~
##  2 Vezenkov   Abdul-Wa~    21 FALSE          Poor Facilitator  Poor Rebounder
##  3 Sead       Abdur-Ra~    24 FALSE          Good Shot Blocker Below Average Re~
##  4 Markel     Abrines      23 FALSE          Below Average Fl~ Below Average Sh~
##  5 Cissoko    Abromait~    29 FALSE          Great Shot Block~ Good Post Scorer
##  6 Morse      Acie         20 TRUE           Above Average Sh~ Above Average Fa~
##  7 Patricio   Adam         29 FALSE          Great Shot Block~ Great Floor Spac~
##  8 Devecchi   Adams        20 FALSE          Below Average Fa~ Poor Shot Blocker
##  9 Daryl      Adel         31 TRUE           Great Rebounder   Good Scorer
## 10 Djurisic   Adiguzel     33 TRUE           Above Average Re~ Above Average Sh~
## # i 1,463 more rows
## # i 7 more variables: archetype_3_label <chr>, scorer_percentile <dbl>,
## #   post_scorer_percentile <dbl>, rebounder_percentile <dbl>,
## #   facilitator_percentile <dbl>, shot_blocker_percentile <dbl>,
## #   floor_spacer_percentile <dbl>
```

Capable Bigs: Filter, 3D Plot, Top-3 Table

```
# ---- 1) Filter & prep ----
capable_bigs <- intl_per_game %>%
  filter(
    games >= 10,
    minutes >=15,
    age <= 33,
    most_recent_year >= 2019,
    !is.na(post_scorer_percentile),
```

```r
    !is.na(shot_blocker_percentile),
    !is.na(rebounder_percentile),
    !is.na(nba_probability),
    nba_probability > 0.5,
    nba_experience == FALSE
  ) %>%
  mutate(
    balance_score = (post_scorer_percentile + shot_blocker_percentile + rebounder_percentile) / 3
  )

# ---- 2) Identify top 3 by overall big score ----
top_n <- min(3, nrow(capable_bigs))
top3_idx <- order(capable_bigs$balance_score, decreasing = TRUE)[seq_len(top_n)]

# Colors: top 3 = orange, others = blue
cols <- rep("#0072B2", nrow(capable_bigs))
if (top_n > 0) cols[top3_idx] <- "#D55E00"

# ---- 3) 3D plot ----
s3d <- scatterplot3d(
  x = capable_bigs$post_scorer_percentile,
  y = capable_bigs$shot_blocker_percentile,
  z = capable_bigs$rebounder_percentile,
  xlab = "Post Scorer Percentile (%)",
  ylab = "Shot Blocker Percentile (%)",
  zlab = "Rebounder Percentile (%)",
  color = cols,
  pch = 19,
  cex.symbols = 0.9,
  main = "Capable Bigs (Post Scoring, Rim Protection, Rebounding)",
  angle = 55
)

# Re-plot and label top 3
if (top_n > 0) {
  s3d$points3d(
    capable_bigs$post_scorer_percentile[top3_idx],
    capable_bigs$shot_blocker_percentile[top3_idx],
    capable_bigs$rebounder_percentile[top3_idx],
    col = "#D55E00", pch = 19, cex = 1.15
  )

  lab <- s3d$xyz.convert(
    capable_bigs$post_scorer_percentile[top3_idx],
    capable_bigs$shot_blocker_percentile[top3_idx],
    capable_bigs$rebounder_percentile[top3_idx]
  )

  text(
    lab$x, lab$y,
    labels = paste(capable_bigs$first_name[top3_idx], capable_bigs$last_name[top3_idx]),
    pos = 2, cex = 0.7, font = 2, col = "#D55E00"
  )
```

```
}
```

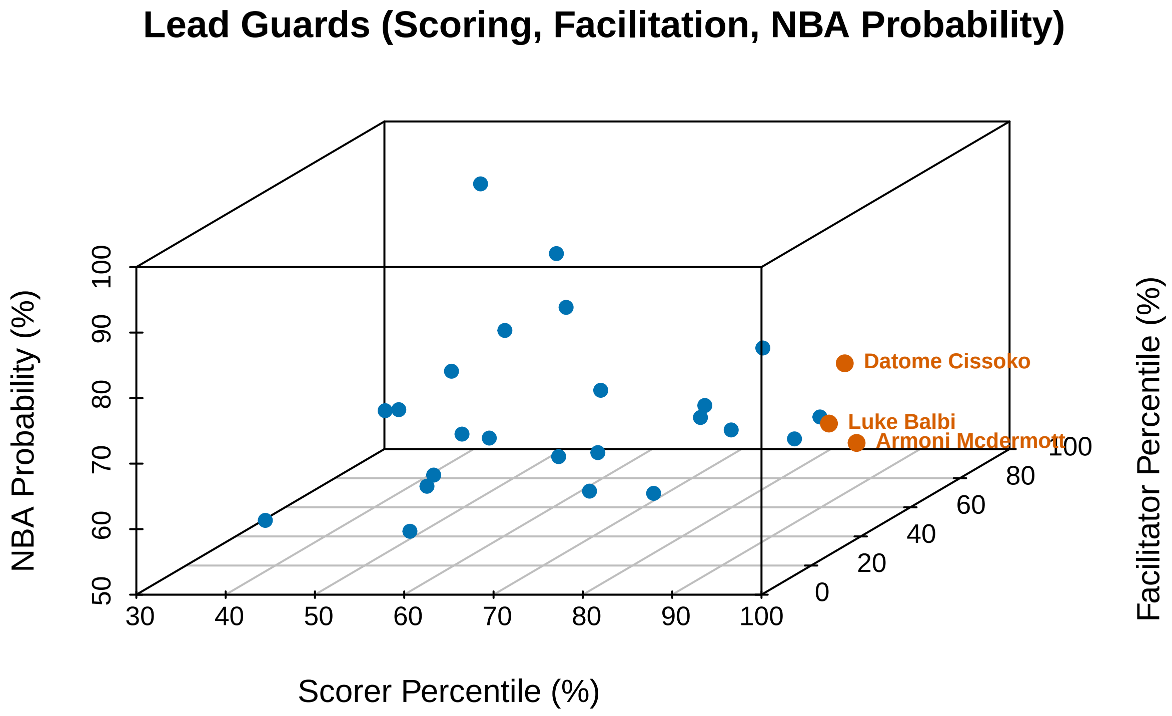


```
capable_bigs_top3 <- capable_bigs %>%
  arrange(desc(balance_score)) %>%
  slice_head(n = 3) %>%                         # top 3 only
  mutate(
    rebounds = coalesce(
      offensive_rebounds + defensive_rebounds
    )
  ) %>%
  dplyr::select(
    first_name, last_name,
    points, rebounds, assists,                 # box score
    games, age, minutes,                               # context
    scorer_tier, post_scorer_tier, rebounder_tier,
    facilitator_tier, shot_blocker_tier, floor_spacer_tier
  )

print(capable_bigs_top3, n = 3)
```

```
## # A tibble: 3 x 14
##   first_name last_name points rebounds assists games   age minutes scorer_tier
##   <chr>      <chr>      <dbl>    <dbl>   <dbl> <dbl> <dbl>   <dbl> <chr>
## 1 Tanoulis   Graham      12.9     5.83    1.1    186    32    23.7 Great
## 2 Morris     Dentmon     11.2     8.66    0.67    12    25    23.5 Above Average
## 3 Conger     Fernandez   15.1     6.8     0.8     10    25    29   Good
## # i 5 more variables: post_scorer_tier <chr>, rebounder_tier <chr>,
## #   facilitator_tier <chr>, shot_blocker_tier <chr>, floor_spacer_tier <chr>
```

Lead Guards: Filter, 3D Plot

```r
# ---- 1) Filter & prep data ----
guards_3d <- intl_per_game %>%
  filter(
    games >= 10,
    age <= 33,
    minutes >= 15,
    nba_probability > 0.5,
    most_recent_year >= 2019,
    nba_experience %in% c(FALSE, 0),       # exclude NBA players
    !is.na(scorer_percentile),
    !is.na(facilitator_percentile),
    !is.na(nba_probability)
  ) %>%
  mutate(
    guard_score = (scorer_percentile + facilitator_percentile) / 2,
    nba_prob_pct = 100 * nba_probability
  )

# ---- 2) Identify top 5 ----
top5_idx <- order(guards_3d$guard_score, decreasing = TRUE)[1:3]

# ---- 3) Assign colors ----
cols <- rep("#0072B2", nrow(guards_3d))
cols[top5_idx] <- "#D55E00"

# ---- 4) 3D Plot ----
s3d <- scatterplot3d(
  x = guards_3d$scorer_percentile,
  y = guards_3d$facilitator_percentile,
  z = guards_3d$nba_prob_pct,
  xlab = "Scorer Percentile (%)",
  ylab = "Facilitator Percentile (%)",
  zlab = "NBA Probability (%)",
  color = cols,
  pch = 19,
  cex.symbols = 0.9,
  main = "Lead Guards (Scoring, Facilitation, NBA Probability)"
)

# ---- 5) Highlight + label top 5 ----
if (length(top5_idx) > 0) {
  s3d$points3d(
    guards_3d$scorer_percentile[top5_idx],
    guards_3d$facilitator_percentile[top5_idx],
    guards_3d$nba_prob_pct[top5_idx],
    col = "#D55E00", pch = 19, cex = 1.1
  )

  lab <- s3d$xyz.convert(
    guards_3d$scorer_percentile[top5_idx],
    guards_3d$facilitator_percentile[top5_idx],
    guards_3d$nba_prob_pct[top5_idx]
  )

  text(
    lab$x, lab$y,
    labels = paste(guards_3d$first_name[top5_idx], guards_3d$last_name[top5_idx]),
    pos = 4,
    cex = 0.7,
    font = 2,
    col = "#D55E00"
  )
}
```

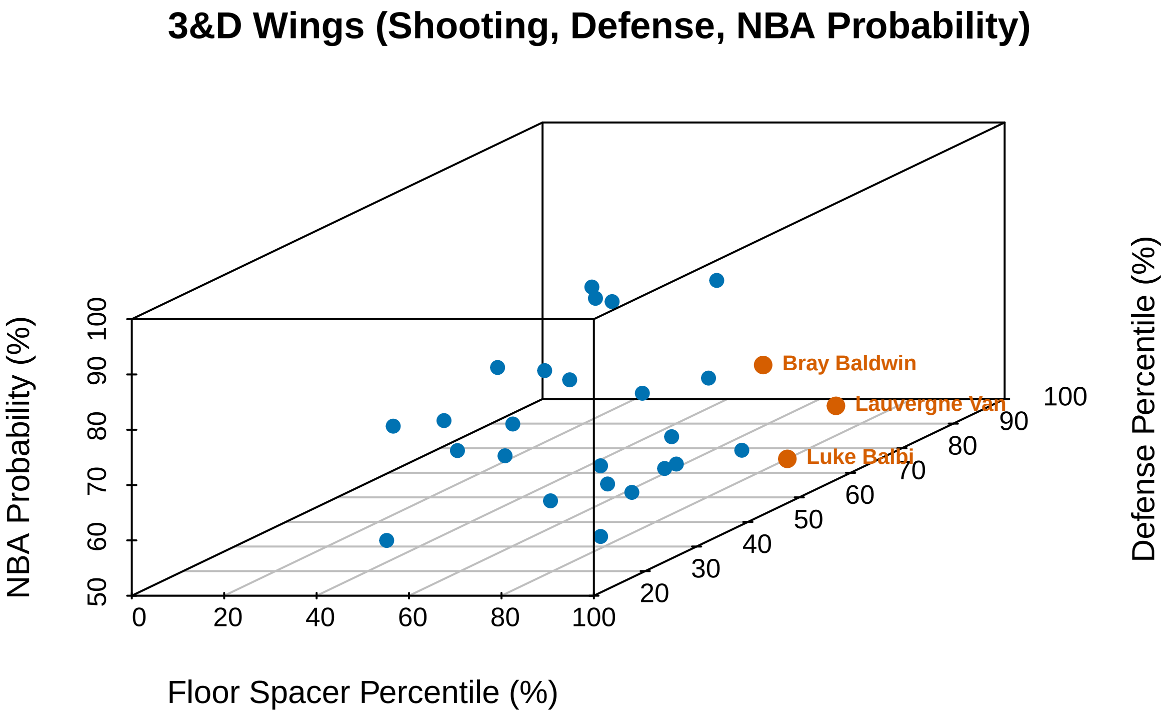


```
guards_top3 <- guards_3d %>%
  mutate(rank_guard = rank(-guard_score, ties.method = "first")) %>%
  filter(rank_guard <= 3) %>%
  arrange(rank_guard) %>%
  mutate(
    Name = paste(first_name, last_name),
    # If you have total_rebounds, use that; otherwise sum O/D boards
    REB = dplyr::coalesce(offensive_rebounds + defensive_rebounds),
    PTS = points,
    AST = assists,
    G   = games
  ) %>%
  dplyr::select(Name, age, PTS, REB, AST, G,
         scorer_tier, facilitator_tier, floor_spacer_tier)

print(guards_top3, n = 3)
```

```
## # A tibble: 3 x 9
##   Name               age   PTS   REB   AST     G scorer_tier facilitator_tier
##   <chr>            <dbl> <dbl> <dbl> <dbl> <dbl> <chr>       <chr>
## 1 Armoni Mcdermott    27  15.5  2.79  3.74    19 Good        Elite
## 2 Datome Cissoko      32  13.1  3.01  4.29   163 Good        Great
## 3 Luke Balbi          28  14.1  1.98  3.46   147 Good        Great
## # i 1 more variable: floor_spacer_tier <chr>
```

3&D Wings: Build Defense Metric, Filter, 3D Plot

```
# --- 1) Build Defense Percentile (Steal% + Block%) ---
intl_per_game <- intl_per_game %>%
  mutate(
    steal_pctile = percent_rank(steal_percentage_wa),
    block_pctile = percent_rank(block_percentage_wa),
    defense_score = (steal_pctile + block_pctile) / 2,
    defense_percentile = 100 * round(percent_rank(defense_score), 3)
  )

# --- 2) Filter data for 3&D view ---
three_d_wings <- intl_per_game %>%
  filter(
    games >= 10,
    minutes >= 15,
    age <= 33,
    nba_probability >= 0.5,
    nba_experience == FALSE,
    most_recent_year >= 2019,
    !is.na(floor_spacer_percentile),
    !is.na(defense_percentile),
    !is.na(nba_probability)
  ) %>%
  mutate(
    nba_prob_pct = 100 * nba_probability,
    three_d_combo = (floor_spacer_percentile + defense_percentile) / 2
  )

# --- 3) Identify top 3 players by 3&D combo score ---
top3_idx <- order(three_d_wings$three_d_combo, decreasing = TRUE)[1:3]

# Assign colors: top 3 orange, rest blue
cols <- rep("#0072B2", nrow(three_d_wings))
cols[top3_idx] <- "#D55E00"

# --- 4) 3D Plot ---
s3d <- scatterplot3d(
  x = three_d_wings$floor_spacer_percentile,
  y = three_d_wings$defense_percentile,
  z = three_d_wings$nba_prob_pct,
  xlab = "Floor Spacer Percentile (%)",
  ylab = "Defense Percentile (%)",
  zlab = "NBA Probability (%)",
  color = cols,
  pch = 19,
  cex.symbols = 0.9,
  main = "3&D Wings (Shooting, Defense, NBA Probability)"
)

# --- 5) Highlight and label top 3 ---
if (length(top3_idx) > 0) {
  s3d$points3d(
    three_d_wings$floor_spacer_percentile[top3_idx],
    three_d_wings$defense_percentile[top3_idx],
    three_d_wings$nba_prob_pct[top3_idx],
    col = "#D55E00", pch = 19, cex = 1.15
  )

  lab <- s3d$xyz.convert(
    three_d_wings$floor_spacer_percentile[top3_idx],
    three_d_wings$defense_percentile[top3_idx],
    three_d_wings$nba_prob_pct[top3_idx]
  )
  text(
    lab$x, lab$y,
    labels = paste(three_d_wings$first_name[top3_idx], three_d_wings$last_name[top3_idx]),
    pos = 4, cex = 0.7, font = 2, col = "#D55E00"
  )
}
```

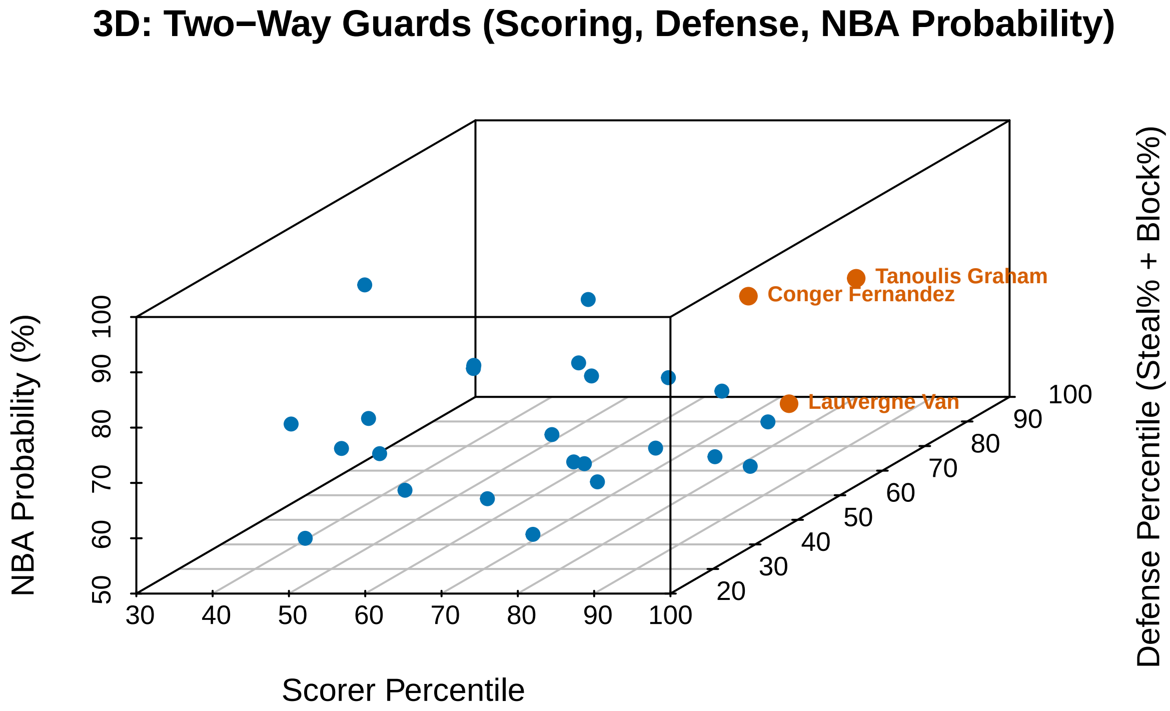


```
three_d_top3 <- three_d_wings %>%
  arrange(desc(three_d_combo)) %>%            # rank by 3&D composite
  slice_head(n = 3) %>%                       # top 3 only
  mutate(
    Name = paste(first_name, last_name),
    PTS  = round(points, 1),
    REB  = round(coalesce(offensive_rebounds + defensive_rebounds), 1),
    AST  = round(assists, 1),
    G    = games,
    defense_tier = if ("defense_tier" %in% names(.)) defense_tier else get_tier(defense_percentile)
  ) %>%
  dplyr::select(Name, age, PTS,  AST, G,
         floor_spacer_tier, defense_tier, three_point_percentage)

print(three_d_top3, n = 3)
```

```
## # A tibble: 3 x 8
##   Name             age   PTS   AST     G floor_spacer_tier defense_tier
##   <chr>          <dbl> <dbl> <dbl> <dbl> <chr>             <chr>
## 1 Lauvergne Van    28  13.7   2.5    61 Great             Good
## 2 Luke Balbi       28  14.1   3.5   147 Great             Above Average
## 3 Bray Baldwin     31   7.4   1.5   262 Above Average     Good
## # i 1 more variable: three_point_percentage <dbl>
```

Two-Way Guards: Filter, 3D Plot

```
# --- 1) Filter + two-way guard score ---
two_way_guards <- intl_per_game %>%
  filter(
    games >= 10,
    age <= 33,
    nba_experience == FALSE,
    nba_probability >= 0.5,
    most_recent_year >=2019,
    minutes >= 15,
    !is.na(scorer_percentile),
    !is.na(defense_percentile),
    !is.na(nba_probability)
  ) %>%
  mutate(
    nba_prob_pct = 100 * nba_probability,
    two_way_guard_score = (scorer_percentile + defense_percentile) / 2
  )

# --- 2) Top 3 by two-way score ---
top3_idx <- order(two_way_guards$two_way_guard_score, decreasing = TRUE)[1:3]
cols <- rep("#0072B2", nrow(two_way_guards))
cols[top3_idx] <- "#D55E00"

# --- 3) 3D plot: Scoring vs Defense vs NBA Prob ---
s3d <- scatterplot3d(
  x = two_way_guards$scorer_percentile,
  y = two_way_guards$defense_percentile,
  z = two_way_guards$nba_prob_pct,
  xlab = "Scorer Percentile",
  ylab = "Defense Percentile (Steal% + Block%)",
  zlab = "NBA Probability (%)",
  color = cols,
  pch = 19,
  cex.symbols = 0.9,
  main = "3D: Two-Way Guards (Scoring, Defense, NBA Probability)"
)

# highlight + label top 3
s3d$points3d(
  two_way_guards$scorer_percentile[top3_idx],
  two_way_guards$defense_percentile[top3_idx],
  two_way_guards$nba_prob_pct[top3_idx],
  col = "#D55E00", pch = 19, cex = 1.15
)
lab <- s3d$xyz.convert(
  two_way_guards$scorer_percentile[top3_idx],
  two_way_guards$defense_percentile[top3_idx],
  two_way_guards$nba_prob_pct[top3_idx]
)
text(
  lab$x, lab$y,
  labels = paste(two_way_guards$first_name[top3_idx], two_way_guards$last_name[top3_idx]),
  pos = 4, cex = 0.7, font = 2, col = "#D55E00"
)
```



Two-Way Guards: Ranked Table

```
# --- 4) Table: top 3 first, then everyone else ---
two_way_guards %>%
  mutate(is_top3 = row_number() %in% top3_idx) %>%
  arrange(desc(is_top3), desc(two_way_guard_score)) %>%
  dplyr::select(first_name, last_name, age, minutes, games,
                scorer_percentile, defense_percentile, nba_probability,
         two_way_guard_score, is_top3) %>%
  print(n = 20)
```

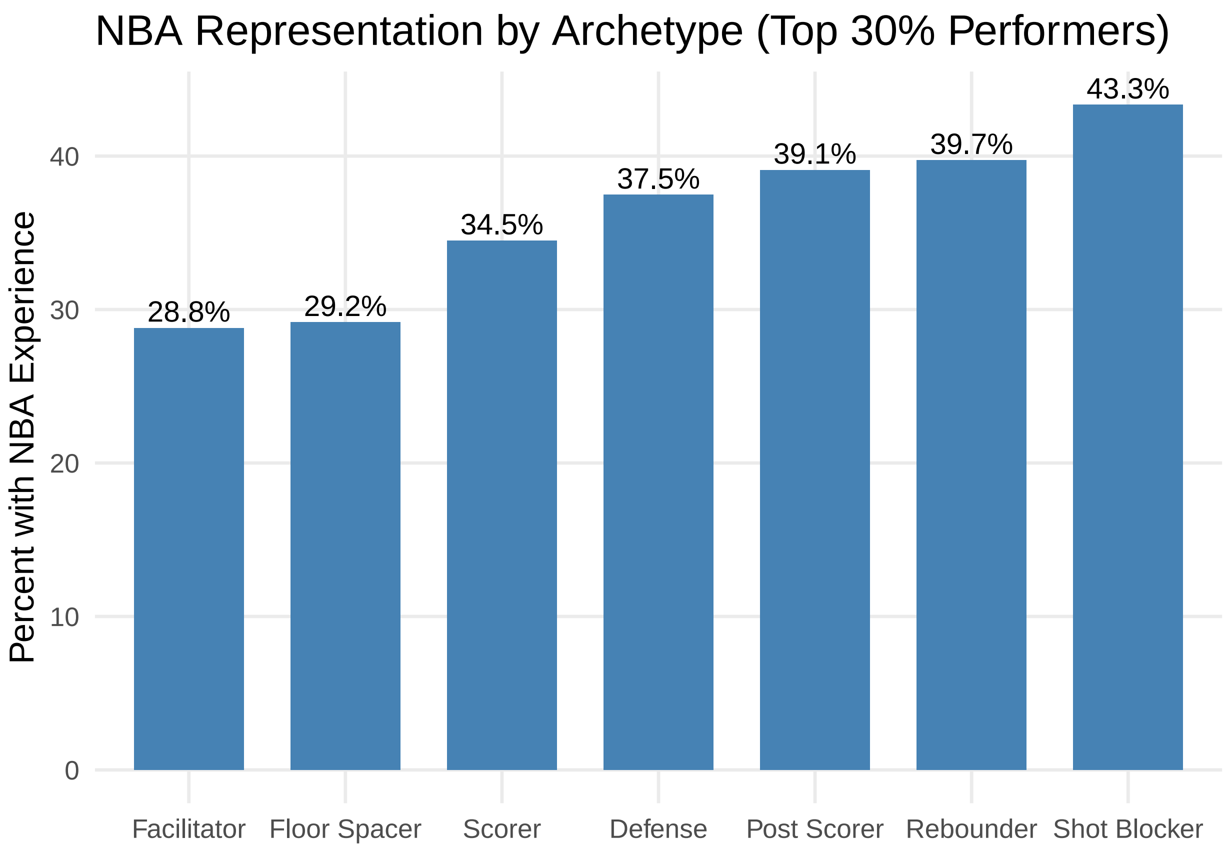
```
## # A tibble: 27 x 10
##    first_name last_name   age minutes games scorer_percentile defense_percentile
##    <chr>      <chr>     <dbl>   <dbl> <dbl>             <dbl>              <dbl>
##  1 Tanoulis   Graham       32    23.7   186              90.9               80.2
##  2 Lauvergne  Van          28    24.1    61              82.6               79.3
##  3 Conger     Fernandez    25    29      10              71                 90.6
##  4 Thon       Knight       31    22.8    74              71.3               83.8
##  5 Armoni     Mcdermott    27    27.6    19              83.9               67.8
##  6 Xabi       Nwaba        27    27.3    29              90.5               60.1
##  7 Morris     Dentmon      25    23.5    12              66.4               80
##  8 Luke       Balbi        28    26.0   147              81.5               63.8
##  9 Holmes     Mcdyess      33    29.0    20              72                 66.9
## 10 Tom        Cacok        33    17.0   169              54.5               82.5
## 11 Killian    Bohannon     30    23.0   182              56.1               80.4
## 12 Bray       Baldwin      31    23.3   262              53.3               82.4
## 13 Strautins  Matthews     33    21.8    83              51.8               78.8
## 14 Padgett    Sykes        29    30.1    65              69.6               51.9
## 15 Vinicius   Lipkevic~    31    25.8    17              76                 42.9
## 16 Semi       Mujakovic    29    26.2    14              43.4               75.5
## 17 Datome     Cissoko      32    27.7   163              83.1               33.2
## 18 Niccolo    Fortas       33    25.1   115              57.4               53.5
## 19 Roko       Blackmon     24    25.5    27              71.2               39.4
## 20 Billy      Faye         25    21      14              32                 73.8
## # i 7 more rows
## # i 3 more variables: nba_probability <dbl>, two_way_guard_score <dbl>,
## #   is_top3 <lgl>
```

Archetype Translation: % with NBA Experience (Top 30%)

```
# --- 1) Gather all archetype percentiles ---
arche_cols <- c(
  "scorer_percentile",
  "facilitator_percentile",
  "floor_spacer_percentile",
  "rebounder_percentile",
  "shot_blocker_percentile",
  "post_scorer_percentile",
  "defense_percentile"
)

arche_long <- intl_per_game %>%
  filter(!is.na(nba_experience)) %>%
  dplyr::select(first_name, last_name, nba_experience, all_of(arche_cols)) %>%
  pivot_longer(
    cols = all_of(arche_cols),
    names_to = "archetype",
    values_to = "percentile"
  ) %>%
  mutate(
    archetype = recode(
      archetype,
      scorer_percentile = "Scorer",
      facilitator_percentile = "Facilitator",
      floor_spacer_percentile = "Floor Spacer",
      rebounder_percentile = "Rebounder",
      shot_blocker_percentile = "Shot Blocker",
      post_scorer_percentile = "Post Scorer",
      defense_percentile = "Defense"
    )
  )

# --- 2) For each archetype, count how many players with NBA experience are Good-or-better ---
arche_value <- arche_long %>%
  filter(percentile >= 70) %>%
  group_by(archetype) %>%
  summarise(
    total_players = n(),
    nba_players = sum(nba_experience == TRUE, na.rm = TRUE),
    pct_nba = nba_players / total_players * 100
  ) %>%
  arrange(desc(pct_nba))

print(arche_value)
```

```
## # A tibble: 7 x 4
##   archetype    total_players nba_players pct_nba
##   <chr>                <int>       <int>   <dbl>
## 1 Shot Blocker           443         192    43.3
## 2 Rebounder              443         176    39.7
## 3 Post Scorer            399         156    39.1
## 4 Defense                443         166    37.5
## 5 Scorer                 432         149    34.5
## 6 Floor Spacer           432         126    29.2
## 7 Facilitator            434         125    28.8
```

```
# --- 3) Plot NBA representation by archetype ---
ggplot(arche_value, aes(x = reorder(archetype, pct_nba), y = pct_nba)) +
  geom_col(fill = "steelblue", width = 0.7) +
  geom_text(aes(label = sprintf("%.1f%%", pct_nba)), vjust = -0.3, size = 3.8) +
  labs(
    title = "NBA Representation by Archetype (Top 30% Performers)",
    x = NULL,
    y = "Percent with NBA Experience"
  ) +
  theme_minimal(base_size = 13) +
  theme(panel.grid.minor = element_blank())
```



```r
nba_exp_summary <- intl_per_game %>%
  summarise(
    total_players = n(),
    nba_players = sum(nba_experience == TRUE, na.rm = TRUE),
    pct_nba = round(100 * nba_players / total_players, 2)
  )
```

Shiny App Source (Not Evaluated)

```r
# --- assume intl_per_game already exists with percentiles, *_tier, nba_probability ---
# Build/ensure a single consistent defense metric: defense_percentile
players <- intl_per_game %>%
  mutate(
    defense_percentile = if ("defense_percentile" %in% names(intl_per_game)) {
      defense_percentile
    } else {
      # Fallback: average of block and (optional) steal percentiles if present
      rowMeans(cbind(
        shot_blocker_percentile,
        if ("steal_percentile" %in% names(intl_per_game)) steal_percentile else NA_real_
      ), na.rm = TRUE)
    }
  ) %>%
  dplyr::select(
    first_name, last_name, age, nba_experience, most_recent_year,
    games, minutes, nba_probability,
    scorer_percentile, post_scorer_percentile, rebounder_percentile,
    facilitator_percentile, shot_blocker_percentile, floor_spacer_percentile,
    defense_percentile,
    scorer_tier, post_scorer_tier, rebounder_tier,
    facilitator_tier, shot_blocker_tier, floor_spacer_tier,
    dplyr::any_of(c("currently_in_nba"))  # optional
  )

# Dropdown mapping for 3D axis choices
percentile_cols <- c(
  "Scorer"        = "scorer_percentile",
  "Post Scorer"   = "post_scorer_percentile",
  "Rebounder"     = "rebounder_percentile",
  "Facilitator"   = "facilitator_percentile",
  "Rim Protector" = "shot_blocker_percentile",
  "Floor Spacer"  = "floor_spacer_percentile",
  "Defensive"     = "defense_percentile"
)

ui <- fluidPage(
  titlePanel("Kings Player Finder"),
  sidebarLayout(
    sidebarPanel(
      textInput("q", "Search name", ""),
      sliderInput("age_max","Max age", min = 16, max = 45, value = 30, step = 1),
      sliderInput("min_games","Min games", min = 0, max = 82, value = 10, step = 1),
      sliderInput("min_minutes","Min minutes", min = 0, max = 40, value = 15, step = 1),

      radioButtons(
        "nba_status", "NBA status",
        choices = c("Any" = "any", "NBA only" = "nba_only", "Not currently in NBA" = "not_nba"),
        selected = "any", inline = TRUE
      ),
      checkboxInput("nba_ready","NBA ready (nba_probability > 0.5)", FALSE),

      tags$hr(),
      h4("Archetype filters (percentile mins)"),
      sliderInput("min_scorer", "Scorer",         0, 100, 0, step = 5),
      sliderInput("min_post",   "Post Scorer",    0, 100, 0, step = 5),
      sliderInput("min_reb",    "Rebounder",      0, 100, 0, step = 5),
      sliderInput("min_fac",    "Facilitator",    0, 100, 0, step = 5),
      sliderInput("min_block",  "Rim Protector",  0, 100, 0, step = 5),
      sliderInput("min_space",  "Floor Spacer",   0, 100, 0, step = 5),
      sliderInput("min_def",    "Defense",        0, 100, 0, step = 5),

      tags$hr(),
      h4("Quick presets"),
      actionButton("preset_big","Two-way Big (post+block+reb 70+)"),
      actionButton("preset_guard","Lead Guard (facil+scorer 85+)"),
      actionButton("preset_3nd","3-and-D Wing (space 80+, block 60+)"),

      tags$hr(),
      downloadButton("dl", "Download CSV")
    ),
    mainPanel(
      tabsetPanel(
        tabPanel("Finder (Table)", DTOutput("tbl")),
        tabPanel("3D Graph",
          fluidRow(
            column(4, selectInput("x_var", "X axis", choices = percentile_cols, selected = "post_scorer_
            column(4, selectInput("y_var", "Y axis", choices = percentile_cols, selected = "shot_blocker
            column(4, selectInput("z_var", "Z axis", choices = percentile_cols, selected = "rebounder_pe
          ),
          plotlyOutput("p3d", height = "620px"),
          helpText("Top 3 by the mean of selected axes are highlighted in orange and labeled. Only playe
        )
      )
    )
  )
)

server <- function(input, output, session){

  # Presets
  observeEvent(input$preset_big, {
    updateSliderInput(session,"min_post",  value = 70)
    updateSliderInput(session,"min_block", value = 70)
    updateSliderInput(session,"min_reb",   value = 70)
  })
  observeEvent(input$preset_guard, {
    updateSliderInput(session,"min_fac",   value = 85)
    updateSliderInput(session,"min_scorer",value = 85)
  })
  observeEvent(input$preset_3nd, {
    updateSliderInput(session,"min_space", value = 80)
    updateSliderInput(session,"min_block", value = 60)
  })

  # Core filtered data
  filtered <- reactive({
    df <- players %>%
      filter(!is.na(most_recent_year) & most_recent_year >= 2019) %>%
      filter(
        is.na(age) | age <= input$age_max,
        games >= input$min_games,
        minutes >= input$min_minutes,
        scorer_percentile       >= input$min_scorer,
        post_scorer_percentile  >= input$min_post,
        rebounder_percentile    >= input$min_reb,
        facilitator_percentile  >= input$min_fac,
        shot_blocker_percentile >= input$min_block,
        floor_spacer_percentile >= input$min_space,
        defense_percentile      >= input$min_def
      )

    # NBA status
    if (input$nba_status == "nba_only") {
      df <- df %>% filter(if ("currently_in_nba" %in% names(.)) currently_in_nba else nba_experience)
    } else if (input$nba_status == "not_nba") {
      df <- df %>% filter(if ("currently_in_nba" %in% names(.)) !currently_in_nba else !nba_experience)
    }

    # NBA ready threshold
    if (input$nba_ready) df <- df %>% filter(!is.na(nba_probability) & nba_probability > 0.5)

    # Name search
    if (nzchar(input$q)) {
      pat <- tolower(input$q)
      df <- df %>%
        mutate(full_name = paste(first_name, last_name)) %>%
        filter(grepl(pat, tolower(full_name), fixed = TRUE)) %>%
        select(-full_name)
    }

    df %>%
      mutate(
        big_balance = rowMeans(cbind(post_scorer_percentile,
                                     shot_blocker_percentile,
                                     rebounder_percentile), na.rm = TRUE),
        guard_score = rowMeans(cbind(facilitator_percentile,
                                     scorer_percentile), na.rm = TRUE)
      ) %>%
      arrange(desc(pmax(big_balance, guard_score, scorer_percentile)))
  })

  output$tbl <- renderDT({
    datatable(
      filtered(),
      options = list(pageLength = 25, scrollX = TRUE),
      rownames = FALSE
    )
  })

  # 3D plot with selectable axes & top-3 highlight
  output$p3d <- renderPlotly({
    df <- filtered()
    req(nrow(df) > 0)

    xcol <- input$x_var; ycol <- input$y_var; zcol <- input$z_var

    df <- df %>%
      mutate(
        axes_mean = rowMeans(cbind(.data[[xcol]], .data[[ycol]], .data[[zcol]]), na.rm = TRUE)
      ) %>%
      arrange(desc(axes_mean)) %>%
      mutate(
        rank_axes = row_number(),
        is_top3   = rank_axes <= 3
      )
```

```r
has_prob <- "nba_probability" %in% names(df)

df <- df %>%
  mutate(
    label = paste0(
      first_name, " ", last_name,
      "<br>Age: ", age,
      " | G: ", games, " | Min: ", minutes,
      "<br>", names(percentile_cols)[percentile_cols == xcol], ": ", round(.data[[xcol]]),
      "<br>", names(percentile_cols)[percentile_cols == ycol], ": ", round(.data[[ycol]]),
      "<br>", names(percentile_cols)[percentile_cols == zcol], ": ", round(.data[[zcol]]),
      if (has_prob) paste0("<br>NBA Prob: ", sprintf("%.2f", nba_probability)) else ""
    )
  )

df_top  <- df %>% filter(is_top3)
df_rest <- df %>% filter(!is_top3)

p <- plot_ly(type = "scatter3d", mode = "markers")

if (nrow(df_rest) > 0) {
  p <- add_trace(
    p, data = df_rest,
    x = ~.data[[xcol]], y = ~.data[[ycol]], z = ~.data[[zcol]],
    text = ~label, hoverinfo = "text",
    marker = list(size = 4),
    name = "Others", showlegend = TRUE
  )
}

if (nrow(df_top) > 0) {
  p <- add_trace(
    p, data = df_top,
    x = ~.data[[xcol]], y = ~.data[[ycol]], z = ~.data[[zcol]],
    text = ~label, hoverinfo = "text",
    marker = list(size = 7, color = "orange", line = list(width = 1)),
    name = "Top 3 (by selected axes)", showlegend = TRUE
  ) %>%
  add_text(
    data = df_top,
    x = ~.data[[xcol]], y = ~.data[[ycol]], z = ~.data[[zcol]],
    text = ~paste(first_name, last_name),
    textposition = "top center",
    showlegend = FALSE
  )
}

p %>% layout(
  scene = list(
    xaxis = list(title = names(percentile_cols)[percentile_cols == xcol]),
    yaxis = list(title = names(percentile_cols)[percentile_cols == ycol]),
    zaxis = list(title = names(percentile_cols)[percentile_cols == zcol])
  )
```

```r
    )
  })

  output$dl <- downloadHandler(
    filename = function() paste0("kings_player_finder_", Sys.Date(), ".csv"),
    content = function(file) write.csv(filtered(), file, row.names = FALSE)
  )
}

shinyApp(ui, server)
```