nflWAR: A Reproducible Method for Offensive Player Evaluation in Football

Sophy Huang, Nancy (Sijia) Li, Siyao Li

July 21, 2024

1 Introduction

nflWAR [6] is an innovative statistical framework designed to revolutionize the way we evaluate individual player contributions in the National Football League (NFL). The incentive of nflWAR is rooted in addressing the need for a reproducible and robust evaluation method within the realm of professional football. Building upon the foundation of Wins Above Replacement (WAR) [4], a metric traditionally used in baseball to assess a player's overall impact on team success, nflWAR adapts and refines these concepts specifically for football. Emerging as a solution to often-limited statistical frameworks available for a complex team sport like football, this advanced metric allows analysts, coaches, and fans to quantify how many additional wins a player provides over a replacement-level counterpart. By taking into account the unique roles and responsibilities in football, nflWAR provides a more standardized way of evaluating players across different leagues and seasons, offering a comprehensive tool that enhances player evaluation and strategic decision-making in NFL. Moreover, it brings a level of precision and insight to player evaluation that goes beyond traditional statistics, enabling a deeper understanding of a player's contribution to a team's success.

2 Related Work

2.1 Wins Above Replacement (WAR)

The concept of WAR serves as a foundational concept in nflWAR. Originated in baseball, it has become a staple metric for assessing player value in team sports. Specifically, WAR evaluates the overall impact of a player by estimating how many more wins they are worth compared to a replacement-level player at the same position. This metric integrates a player's offensive and defensive contributions into a single statistic. The computation of WAR varies slightly between different analytical platforms like Fangraphs and Baseball-Reference, with formulas that consider batting runs, fielding runs, positional adjustment, and replacement runs. For example, Fangraphs version of WAR for position players [3] can be represented as

$$WAR = \frac{PlayerRunsAboveAverage + ReplacementLevelRuns}{RunsPerWin}$$

where *PlayerRunsAboveAverage* represents the total contribution of the player above an average player, which is the sum of batting runs, baserunning runs, fielding runs, positional adjustment (more runs above average may associate with harder positions), and league Adjustment (more prestigious league may associate with higher number of runs as adjustment). *ReplacementLevelRuns* is the number of runs a replacement-level player is expected to contribute, which is typically set below the league average. And finally, *RunsPerWin* is the number of additional runs that contribute to an extra win in the standings, varying slightly each season based on scoring environment.

Because of WAR's multifaceted capability, it has transformed player valuation across various sports. Beyond baseball, its application extends to team management decisions, such as trades and contract negotiations, as it provides a quantifiable measure of a player's value. Furthermore, WAR has been tailored to other sports like hockey, with metrics like Goals Above Replacement (GAR) [2] and Wins Above Replacement Player (WARP) [5] in basketball, reflecting sport-specific performance factors. These variants adapt the core concept of WAR to accommodate the distinctive strategies and player roles found in each sport, allowing for nuanced analysis and strategic insights.

2.2 Other Player Evaluation Methods

Besides WAR, there are various methods used to evaluate players in sports. These include traditional statistics like touchdowns, passing yards, and receptions in football, which provide direct performance measures. Advanced analytics have also evolved, incorporating metrics such as Player Efficiency Rating (PER) [1] and Value Over Replacement Player (VORP) in basketball and Expected Goals (xG) in soccer. Each sport tailors its evaluation tools to reflect the specific skills and game dynamics, emphasizing the need for comprehensive and adaptable evaluation metrics like nflWAR that consider the unique aspects of each role and play situation. For example, in basketball, PER offers insights into a player's contributions per minute, and VORP represents their overall impact relative to a replacement-level player, considering the team's total minutes played.

3 nflWAR

The nflWAR framework adapts the concept of WAR from baseball to the context of NFL, addressing specific demands of football strategy, player roles, and varied game outcomes. Leveraging statistical innovations, this framework offers a sophisticated approach to quantify individual player contributions within dynamic NFL games.

3.1 Play Value Estimation

The initial step in constructing the nflWAR metric involves evaluating each play's intrinsic value, a necessary process given the nature of the available data. The publicly accessible NFL play-by-play dataset, summarized in Table 1, provides a detailed breakdown of each play's game context but does not include specific player identifiers. This necessitates a method to quantify play value independently from individual player contributions initially.

Table 1: Description of the play-by-play dataset.

Variable	**Description**
Possession Team	Team with the ball on offense (opposing team is on defense)
Down	Four downs to advance the ball ten (or more) yards
Yards to go	Distance in yards to advance and convert first down
Yard line	Distance in yards away from opponent's endzone (100 to zero)
Time Remaining	Seconds remaining in game, each game is 3600 seconds long (four quarters, halftime, and a potential overtime)
Score differential	Difference in score between the possession team and opposition

3.1.1 Expected Points (EP)

To quantify the potential scoring outcome of a play, the nflWAR framework adopts a model for estimating expected points using a multinomial logistic regression approach. This model employs logit transformations relative to a baseline 'No Score' event, providing a statistical comparison of various scoring outcomes (touchdowns, field goals, or safeties) to the outcome of no score. Specifically, the model is specified with the following form:

$$\log\left(\frac{P(Y=\text{Touchdown}|X)}{P(Y=\text{No Score}|X)}\right) = X\cdot\beta_{\text{Touchdown}},$$
$$\log\left(\frac{P(Y=\text{Field Goal}|X)}{P(Y=\text{No Score}|X)}\right) = X\cdot\beta_{\text{Field Goal}},$$
$$\vdots$$
$$\log\left(\frac{P(Y=\text{-Touchdown}|X)}{P(Y=\text{No Score}|X)}\right) = X\cdot\beta_{-\text{Touchdown}}.$$

X denotes the covariates describing the game situation for each play, and β is the corresponding coefficient vector for the type of scoring event. The expected points for a play, EP, are calculated by aggregating the probabilities of all scoring outcomes, each weighted by their respective point values:

$$EP = E[Y|X] = \sum_y y\cdot P(Y=y|X). \tag{1}$$

3.1.2 Win Probability (WP)

In addition to scoring potential, understanding a team's likelihood of winning at any point in the game is crucial. The nflWAR utilizes a generalized additive model (GAM) to estimate win probability based on current game context. This model is particularly suited for handling the complexities of football data, as it can accommodate non-linear relationships through smooth functions and incorporate linear and categorical variables. This holistic approach ensures that each play's valuation is reflective of both its scoring impact and its contribution to the overall game outcome.

3.1.3 Expected Points Added (EPA) and Win Probability Added (WPA)

The nflWAR framework refines player evaluation with Expected Points Added (EPA) and Win Probability Added (WPA), collectively referred to as $\delta_{f,i}$. EPA quantifies the change in expected points from the start to the end of a play, while WPA measures the shift in win probability over the course of a play. These metrics are derived for individual plays and attribute specific player contributions to overall team performance and game outcomes, offering a detailed view on how plays influence team success.

3.2 Player Effect Estimation on Estimated Play Value

Following the estimation of each play's value, the next step in the nflWAR framework is to assess the specific contributions of individual players.

3.2.1 Multilevel Modeling Framework

nflWAR utilizes a multilevel modeling approach to address two fundamental characteristics of NFL games: the distinct roles of different positional groups and the interdependence of player involvement across plays. This approach does not treat players as isolated contributors but rather as members of positional groups that significantly influence their performance probabilities. For instance, quarterbacks and receivers each belong to unique positional groups that exhibit specific behavioral patterns and play outcomes. To accommodate these distinctions, the model incorporates varying intercepts for these player groups, effectively capturing the inherent differences in performance baselines. We can illustrate this with an example where the performance metric $\delta_{f,i}$, representing the effect of player i on play f, is modeled with unique intercepts for quarterbacks and receivers. This is specified as follows:

$$\delta_{f,i} \sim \text{Normal}(Q_{q[i]} + C_{c[i]} + X_i'\beta, \sigma^2), \tag{2}$$

where $Q_{q[i]}$ and $C_{c[i]}$ are the varying intercepts for quarterbacks and receivers respectively, modeled as:

$$Q_q \sim \text{Normal}(\mu_Q, \sigma_Q^2), \quad \text{for all quarterbacks } q, \tag{3}$$

$$C_c \sim \text{Normal}(\mu_C, \sigma_C^2), \quad \text{for all receivers } c. \tag{4}$$

Here, X_i represents other covariates affecting the play's outcome. The model accounts for the different impacts that quarterbacks and receivers have on play outcomes by allowing their performance evaluations to vary around group-specific means. This setup not only reflects the typical contributions and specific traits of these positions but also enhances the precision of performance evaluations by recognizing the unique contributions of different player types within their respective groups.

3.2.2 Passing & Rushing Models

Based on the foundational multilevel modeling, nflWAR differentiates its approach to estimating player contributions for passing and rushing plays:

- **Air Yards Model for Passing Plays:** This model specifically quantifies a quarterback's skill in delivering the ball through the air. It focuses on the precision and effectiveness of the throw, attributing the air yards gained directly to the quarterback's performance.

- **Yards After Catch (YAC) Model for Passing Plays:** Complementing the Air Yards Model, the YAC Model assesses the receiver's ability to advance the ball after making the catch. This includes evaluating the receiver's agility, speed, and situational awareness to maximize the play's outcome post-catch.

- **Unified Model for Rushing Plays:** For rushing attempts, a single comprehensive model evaluates the running back's ability to navigate and exploit openings, considering factors like initial burst, tackle-breaking, and field vision. This model integrates various aspects of a rushing play into a holistic assessment of the running back's contribution.

These models are critical for parsing out the specific contributions of players in different play types, allowing coaches, analysts, and fans to appreciate and quantify how individual skills and decisions impact the game's progress and outcome.

3.3 Player Performance Evaluation

In the nflWAR framework, player performance evaluation measures individual contributions based on roles and compares them to replacement-level benchmarks for each position. Players at each position are evaluated according to the key responsibilities and metrics that are most relevant to their roles on the field. Additionally, for positions that entail versatile roles, such as running backs and wide receivers, nflWAR further distinguishes performance metrics between rushing and receiving. This structured evaluation process ensures that player performance is not only recognized within the context of their general position but also in relation to the specific actions they perform, such as receiving and rushing.

3.4 Conversion of Player Performance to WAR

The final step in the nflWAR methodology involves converting the refined performance metrics into WAR value. For the conversion based on Expected Points Added (EPA), a linear regression model is employed to relate the EPA, calculated as points added above replacement, directly to wins. The model is defined as:

$$\text{Wins}_s = \beta_0 + \beta_S S_t + \epsilon_t, \tag{5}$$

where S_t represents the EPA for the team accumulated over the season. The key parameter β_S translates the increase in expected points into an increase in expected wins, capturing a player's impact over the season. The factor $1/\beta_S$ determines how many points contribute to a single win, refining the model's accuracy in win prediction.

The Win Probability Added (WPA) model takes a more direct approach. Here, the final WPA values, which represent the cumulative change in a team's win probability attributable to a player, are translated directly into WAR values. This conversion of EPA and WPA into WAR allows teams to directly assess a player's impact on wins, influencing strategic decisions in player management and team strategy.

4 Data Analysis and Results

The authors of nflWAR developed an R package (*nflscrapR*) for analyzing data from the National Football League (NFL) API [6]. The data used for this analysis is the 2017 NFL season play-by-play data, scraped using *nflscrapR* by the authors of nflWAR. The data contains 45,241 plays and 103 variables. Some examples of the variables included are date, play_id, AirYards, YardsAfterCatch, Passer, Rusher, Receiver, and etc. The data contains plays involving 71 quarterbacks (QBs), 148 rushing backs (RBs), 201 wide receivers (WR), and 109 tight ends (TEs). Four separate models are built for each player position. Two approaches are used to evaluate NFL plays, namely expected points added (EPA) and win probability added (WPA). nflWARs are computed to evaluate the players, and each nflWAR is comprised of three components, which are air_WAR, yac_WAR, and rush_WAR. The findings

from our analysis below are consistent with the results in the nflWAR paper [6]. Detailed data analysis and full results can be found in Appendix A.

4.1 Interpreting EPA-Based and WPA-Based WAR

Figure 1 shows examples of (a) EPA-based WAR and (b) WPA-based WAR for QBs. Player A.Dalton has a EPA-based total_WAR of around 3.55 (or 0.95 for WPA-based), meaning that he contributes around 3.55 (or 0.95 for WPA-based) additional wins compared to a replacement level player at the QB position. Player A.McCarron has a total_WAR of 0, meaning that he is a replacement level player at the QB position.

Player_ID_Name	A.Dalton-00-0027973	A.McCarron-00-0031288
Pass_Attempts	474	14
Rush_Attempts	20	0
Sacks	39	1
Position	QB	QB
Total_Plays	533	15
Replacement_Level	0	1
Player_Model_ID	A.Dalton-00-0027973	Replacement_QB
air_iPA	-0.0409008	-0.1489969
yac_iPA	0.02280526	-0.13068000
rush_iPA	-0.2975750	-0.2436734
air_iPAA	-19.386979	-2.085957
yac_iPAA	10.80969	-1.82952
rush_iPAA	-17.5569275	-0.2436734
air_iPAR	51.23755	0.00000
yac_iPAR	72.75201	0.00000
rush_iPAR	-3.180195	0.000000
total_iPAR	120.8094	0.0000
air_WAR	1.504021	0.000000
yac_WAR	2.135553	0.000000
rush_WAR	-0.09335105	0.00000000
total_WAR	3.546223	0.000000

(a) EPA-Based WAR

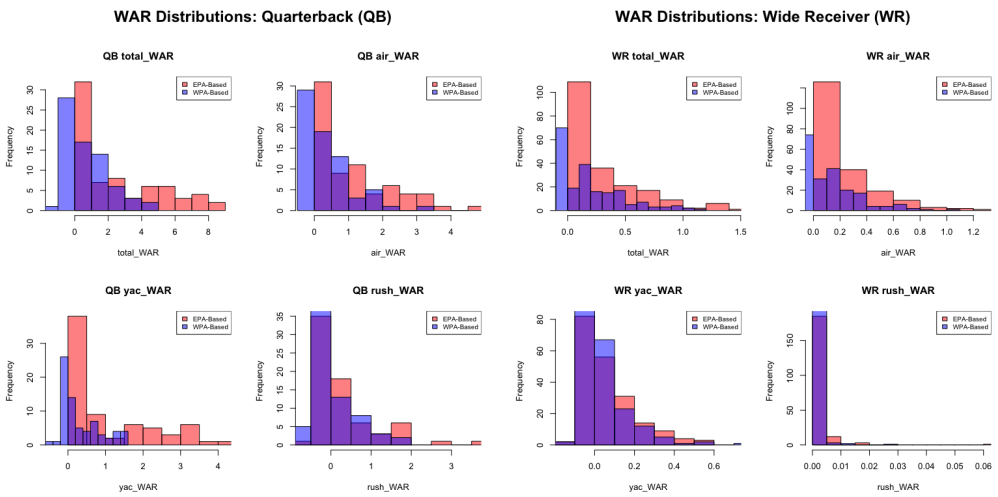
Player_ID_Name	A.Dalton-00-0027973	A.McCarron-00-0031288
Pass_Attempts	474	14
Rush_Attempts	20	0
Sacks	39	1
Position	QB	QB
Total_Plays	533	15
Replacement_Level	0	1
Player_Model_ID	A.Dalton-00-0027973	Replacement_QB
air_iPA	-0.0007491534	-0.0018589553
yac_iPA	0.0006482179	-0.0013491094
rush_iPA	-8.888879e-03	-6.695802e-06
air_iPAA	-0.35509869	-0.02602537
yac_iPAA	0.30725527	-0.01888753
rush_iPAA	-5.244439e-01	-6.695802e-06
air_iPAR	0.5260461	0.0000000
yac_iPAR	0.9467331	0.0000000
rush_iPAR	-0.5240488	0.0000000
total_iPAR	0.9487304	0.0000000
air_WAR	0.5260461	0.0000000
yac_WAR	0.9467331	0.0000000
rush_WAR	-0.5240488	0.0000000
total_WAR	0.9487304	0.0000000

(b) WPA-Based WAR

Figure 1: Examples of EPA- and WPA-Based WAR For QBs

4.2 Comparing EPA-Based and WPA-Based WAR Distributions

For all the player positions, it is found that EPA-based WAR (in red) tend to be higher than WPA-based WAR (in blue), as shown in Figure 2. This may be because that players are performing well in meaningless situations, for example, plays with a large score differential (either leading or trailing by many points). In addition, QBs have higher WAR values as compared to other player positions. This may be because QBs have more pass attempts than non-QBs. Having more opportunities for passing may result in more passing success and contribute to more wins.

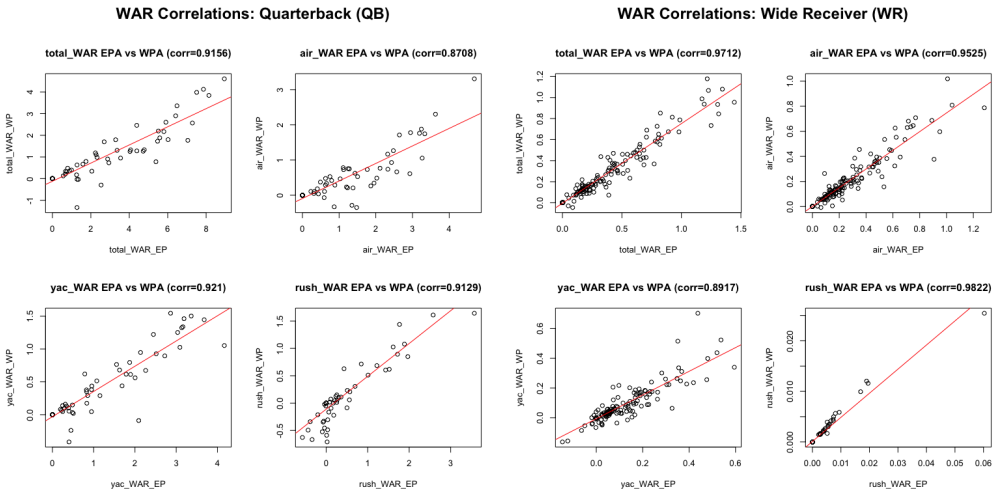


(a) Quarterback (QB) (b) Wide Receiver (WR)

Figure 2: WAR Distributions of QBs and WRs

4.3 Correlations Between EPA-Based and WPA-Based WAR

Figure 3 shows examples of scatter plots of EPA-based WAR against WPA-based WAR for QBs and WRs. It is observed that there are high correlations between the two, showing that the two player evaluation approaches are consistent.

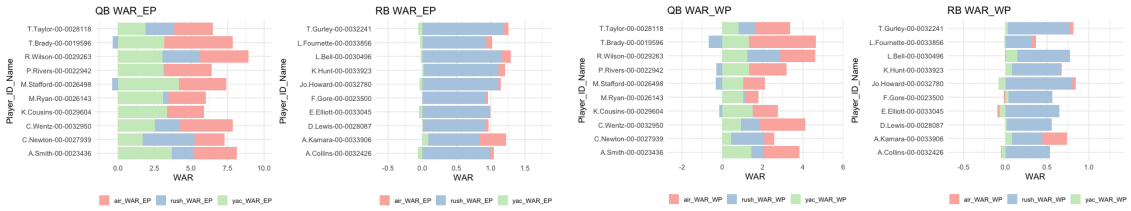


(a) Quarterback (QB) (b) Wide Receiver (WR)

Figure 3: Correlations Between EPA- and WPA-Based WAR For QBs and WRs

4.4 Top 10 Players By Total WAR

Players are ranked by EPA-based and WPA-based total_WAR from the highest to the lowest. As an example, Figure 4 shows the top 10 players by total_WAR for QBs and RBs, stacked by the 3 WAR components (i.e., WAR_{air}, WAR_{yac}, WAR_{rush}). For both EPA-based and WPA-based player evaluation approaches, passing success drives top QB performances while rushing success drives top RB performances.



(a) EPA-Based WAR (QB and RB) (b) WPA-Based WAR (QB and RB)

Figure 4: Top 10 Players by Total WAR For QBs and RBs

5 Conclusion

To summarize, there are three main contributions of nflWAR. First, nflWAR provides an R package *nflscrapR* that enables easy access to NFL play-by-play data. Second, nflWAR develops novel approaches to evaluate plays: estimate expected points (EP) using multinomial logistic regression and estimate win probabilities (WP) using generalized additive model. Third, nflWAR develops novel approach to evaluate NFL players, by not only computing $\text{WAR}_{\text{total}}$, but also separating $\text{WAR}_{\text{total}}$ into 3 components (i.e., WAR_{air}, WAR_{yac}, WAR_{rush}).

One current limitation is that there is no player participation data from NFL. Therefore, there is no information about which players are on the field for each play, limiting the analysis to players directly involved in the play. If player participation data is made available, then nflWAR can be extended to estimate WAR for players of any position in the future. This can be particularly useful for making both on-field and player personnel decisions, influencing team performances in games. NFL teams may also consider leveraging nflWAR to assess draft pick values and make player trade decisions to maximize team performances.

References

[1] Basketball Reference. *About Player Efficiency Rating (PER)*. 2024. URL: https://www.basketball-reference.com/about/per.html (visited on 04/29/2024).

[2] Evolving Hockey. *Goals Above Replacement*. 2024. URL: https://evolving-hockey.com/glossary/goals-above-replacement/ (visited on 04/29/2024).

[3] Fangraphs. *Wins Above Replacement (WAR)*. 2024. URL: https://library.fangraphs.com/misc/war/ (visited on 04/29/2024).

[4] MLB Advanced Media. *Wins Above Replacement (WAR)*. 2024. URL: https://www.mlb.com/glossary/advanced-stats/wins-above-replacement (visited on 04/29/2024).

[5] NBA Stuffer. *Wins Above Replacement Player (WARP)*. 2024. URL: https://www.nbastuffer.com/analytics101/wins-above-replacement-player-warp/ (visited on 04/29/2024).

[6] Ronald Yurko, Samuel Ventura, and Maksim Horowitz. "nflWAR: a reproducible method for offensive player evaluation in football". In: *Journal of Quantitative Analysis in Sports* 15.3 (2019), pp. 163–183. DOI: 10.1515/jqas-2018-0010. URL: https://doi.org/10.1515/jqas-2018-0010 (visited on 04/29/2024).

Appendix A Full Data Analysis

The appendix contains the full analysis conducted in R. The play-by-play data from the 2017 NFL season is used. The analysis starts from data retrieval, to data processing, application of nflWAR, and analysis and visualization of the results.

STAT143_Project

Nancy Li

2024-04-27

Building nlfWAR

Step 1: Get NFL data (2017)

```
library(nflWAR)
```

```
## Loading required package: magrittr
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::extract()   masks magrittr::extract()
## x dplyr::filter()    masks stats::filter()
## x dplyr::lag()       masks stats::lag()
## x purrr::set_names() masks magrittr::set_names()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
##### Step 1: Get NFL Data #####
nfl2017 = get_pbp_data(2017)
print(dim(nfl2017))
```

```
## [1] 45241   103
```

```
print(colnames(nfl2017))
```

```
##   [1] "Date"                      "GameID"
##   [3] "play_id"                   "Drive"
##   [5] "qtr"                       "down"
##   [7] "time"                      "TimeUnder"
##   [9] "TimeSecs"                  "PlayTimeDiff"
##  [11] "SideofField"               "yrdln"
##  [13] "yrdline100"                "ydstogo"
##  [15] "ydsnet"                    "GoalToGo"
##  [17] "FirstDown"                 "posteam"
##  [19] "DefensiveTeam"             "desc"
##  [21] "PlayAttempted"             "Yards.Gained"
##  [23] "sp"                        "Touchdown"
##  [25] "ExPointResult"             "TwoPointConv"
```

```
##  [27] "DefTwoPoint"                "Safety"
##  [29] "Onsidekick"                 "PuntResult"
##  [31] "PlayType"                   "Passer"
##  [33] "Passer_ID"                  "PassAttempt"
##  [35] "PassOutcome"                "PassLength"
##  [37] "AirYards"                   "YardsAfterCatch"
##  [39] "QBHit"                      "PassLocation"
##  [41] "InterceptionThrown"         "Interceptor"
##  [43] "Rusher"                     "Rusher_ID"
##  [45] "RushAttempt"                "RunLocation"
##  [47] "RunGap"                     "Receiver"
##  [49] "Receiver_ID"                "Reception"
##  [51] "ReturnResult"               "Returner"
##  [53] "BlockingPlayer"             "Tackler1"
##  [55] "Tackler2"                   "FieldGoalResult"
##  [57] "FieldGoalDistance"          "Fumble"
##  [59] "RecFumbTeam"                "RecFumbPlayer"
##  [61] "Sack"                       "Challenge.Replay"
##  [63] "ChalReplayResult"           "Accepted.Penalty"
##  [65] "PenalizedTeam"              "PenaltyType"
##  [67] "PenalizedPlayer"            "Penalty.Yards"
##  [69] "PosTeamScore"               "DefTeamScore"
##  [71] "ScoreDiff"                  "AbsScoreDiff"
##  [73] "HomeTeam"                   "AwayTeam"
##  [75] "Timeout_Indicator"          "Timeout_Team"
##  [77] "posteam_timeouts_pre"       "HomeTimeouts_Remaining_Pre"
##  [79] "AwayTimeouts_Remaining_Pre" "HomeTimeouts_Remaining_Post"
##  [81] "AwayTimeouts_Remaining_Post" "No_Score_Prob"
##  [83] "Opp_Field_Goal_Prob"        "Opp_Safety_Prob"
##  [85] "Opp_Touchdown_Prob"         "Field_Goal_Prob"
##  [87] "Safety_Prob"                "Touchdown_Prob"
##  [89] "ExPoint_Prob"               "TwoPoint_Prob"
##  [91] "ExpPts"                     "EPA"
##  [93] "airEPA"                     "yacEPA"
##  [95] "Home_WP_pre"                "Away_WP_pre"
##  [97] "Home_WP_post"               "Away_WP_post"
##  [99] "Win_Prob"                   "WPA"
## [101] "airWPA"                     "yacWPA"
## [103] "Season"
```

```
print(head(nfl2017))
```

```
## # A tibble: 6 x 103
##   Date           GameID play_id Drive   qtr  down time   TimeUnder TimeSecs
##   <date>          <dbl>   <dbl> <dbl> <dbl> <dbl> <time>     <dbl>    <dbl>
## 1 2017-09-07 2017090700      44     1     1    NA 15:00         15     3600
## 2 2017-09-07 2017090700      68     1     1     1 14:55         15     3595
## 3 2017-09-07 2017090700      94     1     1     2 14:49         15     3589
## 4 2017-09-07 2017090700     118     1     1     3 14:14         15     3554
## 5 2017-09-07 2017090700     139     1     1     1 13:52         14     3532
## 6 2017-09-07 2017090700     160     1     1     2 13:26         14     3506
## # i 94 more variables: PlayTimeDiff <dbl>, SideofField <chr>, yrdln <dbl>,
## #   yrdline100 <dbl>, ydstogo <dbl>, ydsnet <dbl>, GoalToGo <dbl>,
## #   FirstDown <dbl>, posteam <chr>, DefensiveTeam <chr>, desc <chr>,
## #   PlayAttempted <dbl>, Yards.Gained <dbl>, sp <dbl>, Touchdown <dbl>,
```

```
## #   ExPointResult <chr>, TwoPointConv <chr>, DefTwoPoint <lgl>, Safety <dbl>,
## #   Onsidekick <dbl>, PuntResult <chr>, PlayType <chr>, Passer <chr>,
## #   Passer_ID <chr>, PassAttempt <dbl>, PassOutcome <chr>, ...
```

There are 103 variables in the file. The description of the variables can be found in this link: https://github.com/ryurko/nflscrapR-data/tree/master/legacy_data.

Step 2: Add player positions to play-by-play data

```
##### Step 2: Add player positions to play-by-play data #####
nfl2017 = add_positions(nfl2017, 2017)
print(dim(nfl2017))
```

```
## [1] 45241   109
```

```
nfl2017_last6 = nfl2017[, (ncol(nfl2017)-5):ncol(nfl2017)]
print(colnames(nfl2017_last6))
```

```
## [1] "Passer_ID_Name"    "Receiver_ID_Name"  "Rusher_ID_Name"
## [4] "Passer_Position"   "Receiver_Position" "Rusher_Position"
```

```
print(head(nfl2017_last6))
```

```
## # A tibble: 6 x 6
##   Passer_ID_Name     Receiver_ID_Name      Rusher_ID_Name     Passer_Position
##   <chr>              <chr>                 <chr>              <chr>
## 1 T.Brady-None       D.Hopkins-None        J.Hekker-None      <NA>
## 2 T.Brady-00-0019596 D.Allen-00-0029689    J.Hekker-None      QB
## 3 T.Brady-00-0019596 R.Burkhead-00-0030288 J.Hekker-None      QB
## 4 T.Brady-None       D.Hopkins-None        J.White-00-0031062 <NA>
## 5 T.Brady-None       D.Hopkins-None        J.White-00-0031062 <NA>
## 6 T.Brady-00-0019596 B.Cooks-00-0031236    J.Hekker-None      QB
## # i 2 more variables: Receiver_Position <chr>, Rusher_Position <chr>
```

There are now a total of 109 variables in the data, with 6 variables added:

1. `Passer_ID_Name`: e.g., T.Brady-00-0019596
2. `Receiver_ID_Name`: e.g., D.Allen-00-0029689
3. `Rusher_ID_Name`: e.g., J.Hekker-None
4. `Passer_Position`: e.g., QB
5. `Receiver_Position`: e.g., TE
6. `Rusher_Position`: e.g., NA

Player positions: running backs (RB), wide receiver (WR), tight end (TE), and quarterback (QB).

Step 3: Add additional model variables to play-by-play data

```
##### Step 3: Add additional model variables to play-by-play data #####
nfl2017 = add_model_variables(nfl2017)
print(dim(nfl2017))
```

```
## [1] 45241   117
```

```
nfl2017_last8 = nfl2017[, (ncol(nfl2017)-7):ncol(nfl2017)]
print(colnames(nfl2017_last8))
```

```
## [1] "Shotgun_Ind"   "No_Huddle_Ind" "Home_Ind"      "airEPA_Result"
## [5] "airWPA_Result" "yacEPA_Result" "yacWPA_Result" "Team_Side_Gap"
```

```
print(head(nfl2017_last8))
```

```
## # A tibble: 6 x 8
##   Shotgun_Ind No_Huddle_Ind Home_Ind airEPA_Result airWPA_Result yacEPA_Result
##         <dbl>         <dbl>    <dbl>         <dbl>         <dbl>         <dbl>
## 1           0             0        1         0.194       0.00601         0.194
## 2           0             0        1        -0.764      -0.0220         -0.764
## 3           0             0        1        -0.849      -0.0243          1.67
## 4           1             0        1         1.37        0.0430          1.37
## 5           1             1        1        -0.238      -0.00601        -0.238
## 6           1             1        1         1.57        0.0516          0.0509
## # i 2 more variables: yacWPA_Result <dbl>, Team_Side_Gap <chr>
```

There are now a total of 117 variables in the data, with 8 variables added:

1. `Shotgun_Ind`: Indicator whether or not the play was in shotgun.
2. `No_Huddle_Ind`: Indicator whether or not the play was no huddle.
3. `Home_Ind`: Indicator whether or not the possession team was home.
4. `airEPA_Result`: airEPA for complete passes and EPA for incomplete.
5. `airWPA_Result`: airWPA for complete passes and WPA for incomplete.
6. `yacEPA_Result`: yacEPA for complete passes and EPA for incomplete.
7. 'yacWPA_Result": yacWPA for complete passes and WPA for incomplete.
8. `Team_Side_Gap`: Combine the team, side, and run gap for O-line proxy

Step 4: Filter the play-by-play data to data for modeling

```
##### Step 4: Filter the play-by-play data to data for modeling #####
nfl2017_model_dfs = prepare_model_data(nfl2017)
pass_model_df2017 = nfl2017_model_dfs$pass_model_df
rush_model_df2017 = nfl2017_model_dfs$rush_model_df

cat("Passing play-by-play data:", dim(pass_model_df2017))
```

```
## Passing play-by-play data: 16980 127
```

```
pass_model_df2017_last10 = pass_model_df2017[
  , (ncol(pass_model_df2017)-9):ncol(pass_model_df2017)]
print(colnames(pass_model_df2017_last10))
```

```
##  [1] "Pass_EPA"      "Pass_WPA"      "Pass_Attempts" "Pass_EPA_Att"
##  [5] "Pass_WPA_Att"  "Rush_EPA"      "Rush_WPA"      "Rush_Attempts"
##  [9] "Rush_EPA_Att"  "Rush_WPA_Att"
```

```
print(head(pass_model_df2017))
```

```
## # A tibble: 6 x 127
##   Date           GameID play_id Drive   qtr  down time   TimeUnder TimeSecs
##   <date>          <dbl>   <dbl> <dbl> <dbl> <dbl> <time>     <dbl>    <dbl>
## 1 2017-09-07 2017090700      68     1     1     1 14:55         15     3595
## 2 2017-09-07 2017090700      94     1     1     2 14:49         15     3589
## 3 2017-09-07 2017090700     160     1     1     2 13:26         14     3506
## 4 2017-09-07 2017090700     210     1     1     2 12:35         13     3455
## 5 2017-09-07 2017090700     309     1     1     1 12:17         13     3437
## 6 2017-09-07 2017090700     427     3     1     1 12:00         12     3420
## # i 118 more variables: PlayTimeDiff <dbl>, SideofField <chr>, yrdln <dbl>,
## #   yrdline100 <dbl>, ydstogo <dbl>, ydsnet <dbl>, GoalToGo <dbl>,
```

```
## #   FirstDown <dbl>, posteam <chr>, DefensiveTeam <chr>, desc <chr>,
## #   PlayAttempted <dbl>, Yards.Gained <dbl>, sp <dbl>, Touchdown <dbl>,
## #   ExPointResult <chr>, TwoPointConv <chr>, DefTwoPoint <lgl>, Safety <dbl>,
## #   Onsidekick <dbl>, PuntResult <chr>, PlayType <chr>, Passer <chr>,
## #   Passer_ID <chr>, PassAttempt <dbl>, PassOutcome <chr>, ...
```

```
cat("Rushing play-by-play data:", dim(rush_model_df2017))
```

```
## Rushing play-by-play data: 14444 127
```

```
rush_model_df2017_last10 = rush_model_df2017[
  , (ncol(rush_model_df2017)-9):ncol(rush_model_df2017)]
print(colnames(rush_model_df2017))
```

```
##   [1] "Date"                          "GameID"
##   [3] "play_id"                       "Drive"
##   [5] "qtr"                           "down"
##   [7] "time"                          "TimeUnder"
##   [9] "TimeSecs"                      "PlayTimeDiff"
##  [11] "SideofField"                   "yrdln"
##  [13] "yrdline100"                    "ydstogo"
##  [15] "ydsnet"                        "GoalToGo"
##  [17] "FirstDown"                     "posteam"
##  [19] "DefensiveTeam"                 "desc"
##  [21] "PlayAttempted"                 "Yards.Gained"
##  [23] "sp"                            "Touchdown"
##  [25] "ExPointResult"                 "TwoPointConv"
##  [27] "DefTwoPoint"                   "Safety"
##  [29] "Onsidekick"                    "PuntResult"
##  [31] "PlayType"                      "Passer"
##  [33] "Passer_ID"                     "PassAttempt"
##  [35] "PassOutcome"                   "PassLength"
##  [37] "AirYards"                      "YardsAfterCatch"
##  [39] "QBHit"                         "PassLocation"
##  [41] "InterceptionThrown"            "Interceptor"
##  [43] "Rusher"                        "Rusher_ID"
##  [45] "RushAttempt"                   "RunLocation"
##  [47] "RunGap"                        "Receiver"
##  [49] "Receiver_ID"                   "Reception"
##  [51] "ReturnResult"                  "Returner"
##  [53] "BlockingPlayer"                "Tackler1"
##  [55] "Tackler2"                      "FieldGoalResult"
##  [57] "FieldGoalDistance"             "Fumble"
##  [59] "RecFumbTeam"                   "RecFumbPlayer"
##  [61] "Sack"                          "Challenge.Replay"
##  [63] "ChalReplayResult"              "Accepted.Penalty"
##  [65] "PenalizedTeam"                 "PenaltyType"
##  [67] "PenalizedPlayer"               "Penalty.Yards"
##  [69] "PosTeamScore"                  "DefTeamScore"
##  [71] "ScoreDiff"                     "AbsScoreDiff"
##  [73] "HomeTeam"                      "AwayTeam"
##  [75] "Timeout_Indicator"             "Timeout_Team"
##  [77] "posteam_timeouts_pre"          "HomeTimeouts_Remaining_Pre"
##  [79] "AwayTimeouts_Remaining_Pre"    "HomeTimeouts_Remaining_Post"
##  [81] "AwayTimeouts_Remaining_Post"   "No_Score_Prob"
```

```
##  [83] "Opp_Field_Goal_Prob"         "Opp_Safety_Prob"
##  [85] "Opp_Touchdown_Prob"          "Field_Goal_Prob"
##  [87] "Safety_Prob"                 "Touchdown_Prob"
##  [89] "ExPoint_Prob"                "TwoPoint_Prob"
##  [91] "ExpPts"                      "EPA"
##  [93] "airEPA"                      "yacEPA"
##  [95] "Home_WP_pre"                 "Away_WP_pre"
##  [97] "Home_WP_post"                "Away_WP_post"
##  [99] "Win_Prob"                    "WPA"
## [101] "airWPA"                      "yacWPA"
## [103] "Season"                      "Passer_ID_Name"
## [105] "Receiver_ID_Name"            "Rusher_ID_Name"
## [107] "Passer_Position"             "Receiver_Position"
## [109] "Rusher_Position"             "Shotgun_Ind"
## [111] "No_Huddle_Ind"               "Home_Ind"
## [113] "airEPA_Result"               "airWPA_Result"
## [115] "yacEPA_Result"               "yacWPA_Result"
## [117] "Team_Side_Gap"               "Pass_EPA"
## [119] "Pass_WPA"                    "Pass_Attempts"
## [121] "Pass_EPA_Att"                "Pass_WPA_Att"
## [123] "Rush_EPA"                    "Rush_WPA"
## [125] "Rush_Attempts"               "Rush_EPA_Att"
## [127] "Rush_WPA_Att"
```

```
print(head(rush_model_df2017))
```

```
## # A tibble: 6 x 127
##   Date           GameID play_id Drive   qtr  down time   TimeUnder TimeSecs
##   <date>          <dbl>   <dbl> <dbl> <dbl> <dbl> <time>     <dbl>    <dbl>
## 1 2017-09-07 2017090700     118     1     1     3 14:14         15     3554
## 2 2017-09-07 2017090700     139     1     1     1 13:52         14     3532
## 3 2017-09-07 2017090700     189     1     1     1 13:02         14     3482
## 4 2017-09-07 2017090700     345     1     1     2 12:12         13     3432
## 5 2017-09-07 2017090700     395     2     1     1 12:08         13     3428
## 6 2017-09-07 2017090700     473     3     1     3 11:21         12     3381
## # i 118 more variables: PlayTimeDiff <dbl>, SideofField <chr>, yrdln <dbl>,
## #   yrdline100 <dbl>, ydstogo <dbl>, ydsnet <dbl>, GoalToGo <dbl>,
## #   FirstDown <dbl>, posteam <chr>, DefensiveTeam <chr>, desc <chr>,
## #   PlayAttempted <dbl>, Yards.Gained <dbl>, sp <dbl>, Touchdown <dbl>,
## #   ExPointResult <chr>, TwoPointConv <chr>, DefTwoPoint <lgl>, Safety <dbl>,
## #   Onsidekick <dbl>, PuntResult <chr>, PlayType <chr>, Passer <chr>,
## #   Passer_ID <chr>, PassAttempt <dbl>, PassOutcome <chr>, ...
```

There are 2 dataframes (passing model and rushing model), each with a total of 127 variables.

Step 5: Create player tables for each position

```
##### Step 5: Create player tables for each position #####
# add_position_tables(model_data_list)
# model_data_list: List of two dataframes (1) pass_model_df, (2) rush_model_df
# return List of dataframes with the given model_data_list,
# as well as the tables for each position containing player names/IDs,
# respective number of attempts for both passing/receiving and rushing/sacks.

nfl2017_player_tables = add_position_tables(nfl2017_model_dfs)
```

Step 6: Find replacement level for each position

```
##### Step 6: Find replacement level for each position #####

# create_percentage_replacement_fn(
# 1. replacement_depth: Number indicating how many players of that position
# and performance type each team should have, then every player have that will
# be considered replacement level
# 2. positions: String indicating which position(s) to find the replacement
# level for, can only be (1) "QB", (2) "RB" (or "FB"), (3) "WR", or (4) "TE"
# 4. attempt_type: String indicating which type of attempts to use from position
# tables to sort the players by for finding the league replacement level
# 5. combine_wrte = 0: Indicator for whether or not to combine the TE and WR
# position tables, which will be primarily used for rushing attempts by
# WRs and TEs (default is 0).
# )
# return: function to find percentage based replacement level QBs
# examples
# Create replacement function for RBs by stating every team has 3 RBs on their
# team for rushing, and every RB with less rushing attempts than the top 3*32 is
# considered replacement level:
# find_replacement_RB_rush=create_league_replacement_fn(3,"RB","Rush_Attempts")
# replacement_RBs = find_replacement_RB_rush(model_data_list)
# Here, our model_data_list is nfl2017_player_tables

league_replacement_functions =
  list("find_replacement_QB" =
         create_percentage_replacement_fn("Perc_Total_Plays", .1),
       "find_replacement_RB_rec" =
         create_league_replacement_fn(3, "RB", "Targets"),
       "find_replacement_WR_rec" =
         create_league_replacement_fn(4, "WR", "Targets"),
       "find_replacement_TE_rec" =
         create_league_replacement_fn(2, "TE", "Targets"),
       "find_replacement_RB_rush" =
         create_league_replacement_fn(3, "RB", "Rush_Attempts"),
       "find_replacement_WR_TE_rush" =
         create_league_replacement_fn(1, "WR", "Rush_Attempts",
                                      combine_wrte = 1))

# find_positional_replacement_level(model_data_list, replacement_fns)
# model_data_list is nfl2017_model_dfs
# replacement_fns: A list of functions to find replacement levels of
# 1. QBs, 2. RBs for receiving plays, 3. WRs for receiving plays,
# 4. TEs for receiving plays, 5. RBs for rushing plays,
# 6. WRs and TEs for rushing plays
# return: the input model_data_list but with replacement-level
# players' names in the play-by-play data all set to be
# Replacement_POSITION and a new column for each of the
# positional tables indicating which players are replacement
# level.

nfl2017_player_tables = find_positional_replacement_level(
  nfl2017_player_tables, league_replacement_functions)
```

```
# qb_table = nfl2017_player_tables$QB_table
# rb_table = nfl2017_player_tables$RB_table
# wr_table = nfl2017_player_tables$WR_table
# te_table = nfl2017_player_tables$TE_table
# cat("\nqb_table:", dim(qb_table))
# print(head(qb_table))
# cat("\nrb_table:", dim(rb_table))
# print(head(rb_table))
# cat("\nwr_table:", dim(wr_table))
# print(head(wr_table))
# cat("\nte_table:", dim(te_table))
# print(head(te_table))
```

Step 7: Calculate the 3 types iPA and iPAA values for each player

```
##### Step 7: Calculate the 3 types iPA and iPAA values for each player #####

### EPA-based WAR ###
## 1. Create the expected points based model formulas: ##
# e.g., (1 | Team_Side_Gap) specifies that a random intercept should be
# included for each level of Team_Side_Gap.
ep_model_formula_list = list(
  "air_formula" = as.formula(
    airEPA_Result ~ Home_Ind + Shotgun_Ind + No_Huddle_Ind + QBHit +
      Receiver_Position + PassLocation + Rush_EPA_Att +
      (1|Passer_ID_Name) + (1|Receiver_ID_Name) + (1|DefensiveTeam)),
  "yac_formula" = as.formula(
    yacEPA_Result ~ Home_Ind + Shotgun_Ind + No_Huddle_Ind + QBHit +
      AirYards*Receiver_Position + PassLocation + Rush_EPA_Att +
      (1|Passer_ID_Name) + (1|Receiver_ID_Name) + (1|DefensiveTeam)),
  "qb_rush_formula" = as.formula(
    EPA ~ Home_Ind + Shotgun_Ind + No_Huddle_Ind + Pass_EPA_Att +
      (1|Rusher_ID_Name) + (1|DefensiveTeam)),
  "main_rush_formula" = as.formula(
    EPA ~ Home_Ind + Shotgun_Ind + No_Huddle_Ind + Rusher_Position +
      Pass_EPA_Att + (1|Team_Side_Gap) + (1|Rusher_ID_Name) + (1|DefensiveTeam)))

## 2. Calculate iPA and iPAA values ##
# estimate_player_value_added(
# 1. model_data_list: nfl2017_player_tables
# 2. model_formula_list: ep_model_formula_list
# 3. return_models = 1)
# return: the original position tables from the input but add these columns:
# air_iPA: Individual player points/probability added in the air.
# yac_iPA: Individual player points/probability added after the catch.
# rush_iPA: Individual player points/probability added from rushing.
# air_iPAA: Individual player points/probability above average in the air.
# yac_iPAA: Individual player points/probability above average after the catch.
# rush_iPAA: Individual player points/probability above average from rushing.
# and the resulting models:
# air_model: Model fit for air component
# yac_model: Model fit for yac component
# qb_rush_model: Model fit for QB rushing/sacks
```

```
# main_rush_model: Model for for all non-QB rushing attempts

nfl2017_player_tables_ep = estimate_player_value_added(
  nfl2017_player_tables, ep_model_formula_list, return_models = 1)
# print("\nEP Example: QB_table and air_model")
# qb_table_ep = nfl2017_player_tables_ep$QB_table
# cat("\nQB_table:", dim(qb_table_ep))
# print(head(qb_table_ep))
# print("\nEP Air Model:")
# print(summary(nfl2017_player_tables_ep$air_model))


### WPA-based WAR ###
## 1. Create the win probability based model formulas: ##
wp_model_formula_list = list(
  "air_formula" = as.formula(
    airWPA_Result ~ Home_Ind + Shotgun_Ind + No_Huddle_Ind + QBHit +
      Receiver_Position + PassLocation + Rush_EPA_Att +
      (1|Passer_ID_Name) + (1|Receiver_ID_Name) + (1|DefensiveTeam)),
  "yac_formula" = as.formula(
    yacWPA_Result ~ Home_Ind + Shotgun_Ind + No_Huddle_Ind + QBHit +
      AirYards*Receiver_Position + PassLocation + Rush_EPA_Att +
      (1|Passer_ID_Name) + (1|Receiver_ID_Name) + (1|DefensiveTeam)),
  "qb_rush_formula" = as.formula(
    WPA ~ Home_Ind + Shotgun_Ind + No_Huddle_Ind + Pass_EPA_Att +
      (1|Rusher_ID_Name) + (1|DefensiveTeam)),
  "main_rush_formula" = as.formula(
    WPA ~ Home_Ind + Shotgun_Ind + No_Huddle_Ind +
      Rusher_Position + Pass_EPA_Att +
      (1|Team_Side_Gap) + (1|Rusher_ID_Name) + (1|DefensiveTeam)))

## 2. Calculate iPA and iPAA values ##
nfl2017_player_tables_wp = estimate_player_value_added(
  nfl2017_player_tables, wp_model_formula_list, return_models = 1)
```

```
## boundary (singular) fit: see help('isSingular')
```

```
# print("\nWP Example: QB_table and air_model")
# qb_table_wp = nfl2017_player_tables_wp$QB_table
# cat("\nQB_table:", dim(qb_table_wp))
# print(head(qb_table_wp))
# print("\nWP Air Model:")
# print(summary(nfl2017_player_tables_wp$air_model))
```

Step 8: Calculate above replacement level

```
##### Step 8: Calculate above replacement level #####
# calculate_above_replacement(model_data_list)
# return: the input position tables with individual points/probability above
# replacement (iPAR) columns: air_iPAR, yac_iPAR, rush_iPAR, and total_iPAR

# model_data_list: nfl2017_player_tables_ep, nfl2017_player_tables_wp

### EPA-based WAR ###
```

```
nfl2017_player_tables_ep = calculate_above_replacement(nfl2017_player_tables_ep)
# print(head(nfl2017_player_tables_ep$QB_table))

### WPA-based WAR ###
nfl2017_player_tables_wp = calculate_above_replacement(nfl2017_player_tables_wp)
```

Step 9: Convert points / probabilities to wins

```
##### Step 9: Convert points / probabilities to wins #####
library(knitr)

### EPA-based WAR: Convert Points to Wins ###
# convert_points_to_wins(model_data_list, points_per_win)
# model_data_list: nfl2017_player_tables_ep
# points_per_win: value indicating the number of points per win to use
# (i.e., result of function calculate_points_per_win(years=2017))
points_per_win = calculate_points_per_win(years=2017)
print(paste("points_per_win:", points_per_win))
```

```
## [1] "points_per_win: 34.0670540902995"
```

```
nfl2017_ep = convert_points_to_wins(nfl2017_player_tables_ep, points_per_win)
print("\n##### EPA-based WAR #####")
```

```
## [1] "\n##### EPA-based WAR #####"
```

```
print("\nPlayer Tables By Position:")
```

```
## [1] "\nPlayer Tables By Position:"
```

```
qb_table_ep = nfl2017_ep$QB_table; rb_table_ep = nfl2017_ep$RB_table
wr_table_ep = nfl2017_ep$WR_table; te_table_ep = nfl2017_ep$TE_table
cat("\nQB_table:", dim(qb_table_ep))
```

```
##
## QB_table: 71 22
```

```
kable(t(head(qb_table_ep, 2)))
```

Player_ID_Name	A.Dalton-00-0027973	A.McCarron-00-0031288
Pass_Attempts	474	14
Rush_Attempts	20	0
Sacks	39	1
Position	QB	QB
Total_Plays	533	15
Replacement_Level	0	1
Player_Model_ID	A.Dalton-00-0027973	Replacement_QB
air_iPA	-0.0409008	-0.1489969
yac_iPA	0.02280526	-0.13068000
rush_iPA	-0.2975750	-0.2436734
air_iPAA	-19.386979	-2.085957
yac_iPAA	10.80969	-1.82952
rush_iPAA	-17.5569275	-0.2436734
air_iPAR	51.23755	0.00000
yac_iPAR	72.75201	0.00000

rush_iPAR	-3.180195	0.000000
total_iPAR	120.8094	0.0000
air_WAR	1.504021	0.000000
yac_WAR	2.135553	0.000000
rush_WAR	-0.09335105	0.00000000
total_WAR	3.546223	0.000000

```
cat("\nRB_table:", dim(rb_table_ep))
```

```
##
## RB_table: 148 24
```

```
kable(t(head(rb_table_ep, 2)))
```

Player_ID_Name	A.Abdullah-00-0032104	A.Blue-00-0031075
Receptions	25	7
Targets	35	9
Rush_Attempts	165	71
Position	RB	RB
Total_Plays	225	87
Replacement_Level_Rusher	0	0
Replacement_Level_Receiver	0	0
Player_Model_ID_Rush	A.Abdullah-00-0032104	A.Blue-00-0031075
Player_Model_ID_Rec	A.Abdullah-00-0032104	A.Blue-00-0031075
air_iPA	0.026385821	-0.001897444
yac_iPA	-0.003720839	0.009735117
rush_iPA	-0.038184456	0.001590085
air_iPAA	0.9235037	-0.0170770
yac_iPAA	-0.13022936	0.08761605
rush_iPAA	-6.300435	0.112896
air_iPAR	2.9432776	0.5022934
yac_iPAR	-1.1708242	-0.1799655
rush_iPAR	10.105343	7.172352
total_iPAR	11.87780	7.49468
air_WAR	0.08639660	0.01474426
yac_WAR	-0.034368225	-0.005282684
rush_WAR	0.2966309	0.2105363
total_WAR	0.3486593	0.2199979

```
cat("\nWR_table:", dim(wr_table_ep))
```

```
##
## WR_table: 201 24
```

```
kable(t(head(wr_table_ep, 2)))
```

Player_ID_Name	A.Benn-00-0027654	A.Brown-00-0027793
Receptions	1	100
Targets	2	162
Rush_Attempts	0	0
Position	WR	WR
Total_Plays	3	262

Replacement_Level_Receiver	1	0
Replacement_Level_Rusher	1	1
Player_Model_ID_Rec	Replacement_WR_rec	A.Brown-00-0027793
Player_Model_ID_Rush	Replacement_WR_TE_rush	Replacement_WR_TE_rush
air_iPA	-0.1391017	0.0727865
yac_iPA	-0.06047183	-0.01640342
rush_iPA	-0.03227709	-0.03227709
air_iPAA	-0.2782034	11.7914126
yac_iPAA	-0.1209437	-2.6573541
rush_iPAA	0	0
air_iPAR	0.00000	34.32589
yac_iPAR	0.000000	7.139082
rush_iPAR	0	0
total_iPAR	0.00000	41.46497
air_WAR	0.000000	1.007598
yac_WAR	0.0000000	0.2095597
rush_WAR	0	0
total_WAR	0.000000	1.217157

```
cat("\nTE_table:", dim(te_table_ep))
```

```
##
## TE_table: 109 24
```

```
kable(t(head(te_table_ep, 2)))
```

Player_ID_Name	A.Auclair-00-0033421	A.Cross-00-0032745
Receptions	2	5
Targets	2	6
Rush_Attempts	0	0
Position	TE	TE
Total_Plays	4	11
Replacement_Level_Receiver	1	1
Replacement_Level_Rusher	1	1
Player_Model_ID_Rec	Replacement_TE_rec	Replacement_TE_rec
Player_Model_ID_Rush	Replacement_WR_TE_rush	Replacement_WR_TE_rush
air_iPA	-0.02075941	-0.02075941
yac_iPA	0.03867179	0.03867179
rush_iPA	-0.03227709	-0.03227709
air_iPAA	-0.04151882	-0.12455645
yac_iPAA	0.07734357	0.23203071
rush_iPAA	0	0
air_iPAR	0	0
yac_iPAR	0	0
rush_iPAR	0	0
total_iPAR	0	0
air_WAR	0	0
yac_WAR	0	0
rush_WAR	0	0
total_WAR	0	0

```r
print("\nEP Air Model:")
```

```
## [1] "\nEP Air Model:"
```

```r
print(summary(nfl2017_ep$air_model))
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: airEPA_Result ~ Home_Ind + Shotgun_Ind + No_Huddle_Ind + QBHit +
##     Receiver_Position + PassLocation + Rush_EPA_Att + (1 | Passer_ID_Name) +
##     (1 | Receiver_ID_Name) + (1 | DefensiveTeam)
##
## REML criterion at convergence: 57762.7
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -8.5709 -0.4832 -0.1098  0.4307  4.6164
##
## Random effects:
##  Groups           Name        Variance Std.Dev.
##  Receiver_ID_Name (Intercept) 0.009681 0.09839
##  Passer_ID_Name   (Intercept) 0.006861 0.08283
##  DefensiveTeam    (Intercept) 0.002655 0.05152
##  Residual                     1.740015 1.31910
## Number of obs: 16980, groups:
## Receiver_ID_Name, 291; Passer_ID_Name, 47; DefensiveTeam, 32
##
## Fixed effects:
##                     Estimate Std. Error t value
## (Intercept)         -0.07645    0.05892  -1.298
## Home_Ind             0.02326    0.02042   1.139
## Shotgun_Ind         -0.11582    0.02514  -4.606
## No_Huddle_Ind        0.12751    0.03369   3.785
## QBHit               -0.23095    0.03606  -6.404
## Receiver_PositionFB -0.50305    0.12387  -4.061
## Receiver_PositionRB -0.59481    0.03029 -19.634
## Receiver_PositionTE -0.09938    0.03133  -3.172
## PassLocationmiddle   0.20542    0.02693   7.629
## PassLocationright    0.01540    0.02337   0.659
## Rush_EPA_Att         0.09585    0.21896   0.438
##
## Correlation of Fixed Effects:
##             (Intr) Hm_Ind Shtg_I N_Hd_I QBHit  Rc_PFB Rc_PRB Rc_PTE PssLctnm
## Home_Ind    -0.188
## Shotgun_Ind -0.338  0.024
## No_Hddl_Ind  0.007 -0.035 -0.083
## QBHit       -0.056  0.009  0.000  0.015
## Rcvr_PstnFB -0.082  0.000  0.096  0.005  0.005
## Rcvr_PstnRB -0.153  0.004 -0.018  0.011 -0.001  0.091
## Rcvr_PstnTE -0.150 -0.003  0.043  0.005 -0.002  0.074  0.291
## PssLctnmddl -0.151 -0.006 -0.036 -0.010 -0.003 -0.001 -0.005 -0.063
## PssLctnrght -0.211  0.011  0.029  0.007  0.006  0.004  0.000 -0.035  0.445
## Rsh_EPA_Att  0.809 -0.011 -0.003  0.045  0.003 -0.011 -0.005  0.002  0.006
##             PssLctnr
## Home_Ind
```

```
## Shotgun_Ind
## No_Hddl_Ind
## QBHit
## Rcvr_PstnFB
## Rcvr_PstnRB
## Rcvr_PstnTE
## PssLctnmddl
## PssLctnrght
## Rsh_EPA_Att  0.000
```

```
print("\nEP YAC Model:")
```

```
## [1] "\nEP YAC Model:"
```

```
print(summary(nfl2017_ep$yac_model))
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: yacEPA_Result ~ Home_Ind + Shotgun_Ind + No_Huddle_Ind + QBHit +
##     AirYards * Receiver_Position + PassLocation + Rush_EPA_Att +
##     (1 | Passer_ID_Name) + (1 | Receiver_ID_Name) + (1 | DefensiveTeam)
##
## REML criterion at convergence: 57650.4
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -8.7901 -0.4123  0.0184  0.3530  6.8385
##
## Random effects:
##  Groups           Name        Variance Std.Dev.
##  Receiver_ID_Name (Intercept) 0.007262 0.08522
##  Passer_ID_Name   (Intercept) 0.007113 0.08434
##  DefensiveTeam    (Intercept) 0.002731 0.05226
##  Residual                     1.726646 1.31402
## Number of obs: 16980, groups:
## Receiver_ID_Name, 291; Passer_ID_Name, 47; DefensiveTeam, 32
##
## Fixed effects:
##                               Estimate Std. Error t value
## (Intercept)                   0.333677   0.060317   5.532
## Home_Ind                      0.035183   0.020337   1.730
## Shotgun_Ind                  -0.133151   0.025098  -5.305
## No_Huddle_Ind                 0.047358   0.033547   1.412
## QBHit                        -0.384893   0.035978 -10.698
## AirYards                     -0.023707   0.001241 -19.110
## Receiver_PositionFB           0.044993   0.139817   0.322
## Receiver_PositionRB           0.454263   0.032638  13.918
## Receiver_PositionTE           0.091241   0.039925   2.285
## PassLocationmiddle           -0.045823   0.026882  -1.705
## PassLocationright            -0.038477   0.023291  -1.652
## Rush_EPA_Att                  0.390669   0.217665   1.795
## AirYards:Receiver_PositionFB  0.006166   0.025442   0.242
## AirYards:Receiver_PositionRB -0.032166   0.004122  -7.803
## AirYards:Receiver_PositionTE -0.007618   0.003088  -2.467
##
## Correlation matrix not shown by default, as p = 15 > 12.
```

```
## Use print(summary(nfl2017_ep$yac_model), correlation=TRUE)  or
##     vcov(summary(nfl2017_ep$yac_model))        if you need it
```

```
print("\nEP Rush Model:")
```

```
## [1] "\nEP Rush Model:"
```

```
print(summary(nfl2017_ep$qb_rush_model))
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: EPA ~ Home_Ind + Shotgun_Ind + No_Huddle_Ind + Pass_EPA_Att +
##     (1 | Rusher_ID_Name) + (1 | DefensiveTeam)
##
## REML criterion at convergence: 9571.6
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -6.8230 -0.4624 -0.0035  0.5295  3.5939
##
## Random effects:
##  Groups         Name        Variance Std.Dev.
##  Rusher_ID_Name (Intercept) 0.18948  0.4353
##  DefensiveTeam  (Intercept) 0.02596  0.1611
##  Residual                   2.93752  1.7139
## Number of obs: 2423, groups:  Rusher_ID_Name, 47; DefensiveTeam, 32
##
## Fixed effects:
##               Estimate Std. Error t value
## (Intercept)   -0.78490    0.13199  -5.947
## Home_Ind       0.12533    0.07086   1.769
## Shotgun_Ind   -0.11857    0.08243  -1.438
## No_Huddle_Ind -0.03434    0.13315  -0.258
## Pass_EPA_Att  -0.22238    0.59172  -0.376
##
## Correlation of Fixed Effects:
##             (Intr) Hm_Ind Shtg_I N_Hd_I
## Home_Ind    -0.271
## Shotgun_Ind -0.469  0.017
## No_Hddl_Ind -0.044 -0.056 -0.049
## Pss_EPA_Att -0.577 -0.002  0.008 -0.006
```

```
### WPA-based WAR: Convert Probabilities to Wins  ###
# # convert_prob_to_wins(model_data_list)
# model_data_list: nfl2017_player_tables_wp
nfl2017_wp = convert_prob_to_wins(nfl2017_player_tables_wp)

print("\n##### WPA-based WAR #####")
```

```
## [1] "\n##### WPA-based WAR #####"
```

```
print("\nPlayer Tables By Position:")
```

```
## [1] "\nPlayer Tables By Position:"
```

```
qb_table_wp = nfl2017_wp$QB_table; rb_table_wp = nfl2017_wp$RB_table
wr_table_wp = nfl2017_wp$WR_table; te_table_wp = nfl2017_wp$TE_table
cat("\nQB_table:", dim(qb_table_wp))
```

```
##
## QB_table: 71 22
```

```
kable(t(head(qb_table_wp, 2)))
```

Player_ID_Name	A.Dalton-00-0027973	A.McCarron-00-0031288
Pass_Attempts	474	14
Rush_Attempts	20	0
Sacks	39	1
Position	QB	QB
Total_Plays	533	15
Replacement_Level	0	1
Player_Model_ID	A.Dalton-00-0027973	Replacement_QB
air_iPA	-0.0007491534	-0.0018589553
yac_iPA	0.0006482179	-0.0013491094
rush_iPA	-8.888879e-03	-6.695802e-06
air_iPAA	-0.35509869	-0.02602537
yac_iPAA	0.30725527	-0.01888753
rush_iPAA	-5.244439e-01	-6.695802e-06
air_iPAR	0.5260461	0.0000000
yac_iPAR	0.9467331	0.0000000
rush_iPAR	-0.5240488	0.0000000
total_iPAR	0.9487304	0.0000000
air_WAR	0.5260461	0.0000000
yac_WAR	0.9467331	0.0000000
rush_WAR	-0.5240488	0.0000000
total_WAR	0.9487304	0.0000000

```
cat("\nRB_table:", dim(rb_table_wp))
```

```
##
## RB_table: 148 24
```

```
kable(t(head(rb_table_wp, 2)))
```

Player_ID_Name	A.Abdullah-00-0032104	A.Blue-00-0031075
Receptions	25	7
Targets	35	9
Rush_Attempts	165	71
Position	RB	RB
Total_Plays	225	87
Replacement_Level_Rusher	0	0
Replacement_Level_Receiver	0	0
Player_Model_ID_Rush	A.Abdullah-00-0032104	A.Blue-00-0031075
Player_Model_ID_Rec	A.Abdullah-00-0032104	A.Blue-00-0031075
air_iPA	5.000704e-04	-2.005677e-06
yac_iPA	0.0002424435	0.0003490375
rush_iPA	-0.0008859304	-0.0001833949
air_iPAA	1.750246e-02	-1.805109e-05
yac_iPAA	0.008485523	0.003141337
rush_iPAA	-0.14617851	-0.01302104
air_iPAR	0.033978264	0.004218583
yac_iPAR	-0.007885249	-0.001068290
rush_iPAR	0.11208644	0.09811115

total_iPAR	0.1381795	0.1012614
air_WAR	0.033978264	0.004218583
yac_WAR	-0.007885249	-0.001068290
rush_WAR	0.11208644	0.09811115
total_WAR	0.1381795	0.1012614

```
cat("\nWR_table:", dim(wr_table_wp))
```

```
##
## WR_table: 201 24
```

```
kable(t(head(wr_table_wp, 2)))
```

Player_ID_Name	A.Benn-00-0027654	A.Brown-00-0027793
Receptions	1	100
Targets	2	162
Rush_Attempts	0	0
Position	WR	WR
Total_Plays	3	262
Replacement_Level_Receiver	1	0
Replacement_Level_Rusher	1	1
Player_Model_ID_Rec	Replacement_WR_rec	A.Brown-00-0027793
Player_Model_ID_Rush	Replacement_WR_TE_rush	Replacement_WR_TE_rush
air_iPA	-0.003048071	0.003237740
yac_iPA	-0.0012540402	-0.0002717185
rush_iPA	-0.0005573602	-0.0005573602
air_iPAA	-0.006096141	0.524513807
yac_iPAA	-0.00250808	-0.04401840
rush_iPAA	0	0
air_iPAR	0.000000	1.018301
yac_iPAR	0.0000000	0.1591361
rush_iPAR	0	0
total_iPAR	0.000000	1.177437
air_WAR	0.000000	1.018301
yac_WAR	0.0000000	0.1591361
rush_WAR	0	0
total_WAR	0.000000	1.177437

```
cat("\nTE_table:", dim(te_table_wp))
```

```
##
## TE_table: 109 24
```

```
kable(t(head(te_table_wp, 2)))
```

Player_ID_Name	A.Auclair-00-0033421	A.Cross-00-0032745
Receptions	2	5
Targets	2	6
Rush_Attempts	0	0
Position	TE	TE
Total_Plays	4	11
Replacement_Level_Receiver	1	1

Replacement_Level_Rusher	1	1
Player_Model_ID_Rec	Replacement_TE_rec	Replacement_TE_rec
Player_Model_ID_Rush	Replacement_WR_TE_rush	Replacement_WR_TE_rush
air_iPA	0.00104609	0.00104609
yac_iPA	0.0007885088	0.0007885088
rush_iPA	-0.0005573602	-0.0005573602
air_iPAA	0.002092180	0.006276541
yac_iPAA	0.001577018	0.004731053
rush_iPAA	0	0
air_iPAR	0	0
yac_iPAR	0	0
rush_iPAR	0	0
total_iPAR	0	0
air_WAR	0	0
yac_WAR	0	0
rush_WAR	0	0
total_WAR	0	0

```
print("\nEP Air Model:")
```

```
## [1] "\nEP Air Model:"
```

```
print(summary(nfl2017_wp$air_model))
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: airWPA_Result ~ Home_Ind + Shotgun_Ind + No_Huddle_Ind + QBHit +
##     Receiver_Position + PassLocation + Rush_EPA_Att + (1 | Passer_ID_Name) +
##     (1 | Receiver_ID_Name) + (1 | DefensiveTeam)
##
## REML criterion at convergence: -57586.9
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -21.9450  -0.4332  -0.0584   0.3247  17.4663
##
## Random effects:
##  Groups           Name        Variance  Std.Dev.
##  Receiver_ID_Name (Intercept) 7.931e-06 0.002816
##  Passer_ID_Name   (Intercept) 5.472e-06 0.002339
##  DefensiveTeam    (Intercept) 0.000e+00 0.000000
##  Residual                     1.947e-03 0.044128
## Number of obs: 16980, groups:
## Receiver_ID_Name, 291; Passer_ID_Name, 47; DefensiveTeam, 32
##
## Fixed effects:
##                       Estimate Std. Error t value
## (Intercept)         -1.014e-03  1.795e-03  -0.565
## Home_Ind            -4.768e-04  6.818e-04  -0.699
## Shotgun_Ind         -2.898e-03  8.380e-04  -3.458
## No_Huddle_Ind        3.364e-03  1.121e-03   3.001
## QBHit               -5.562e-03  1.205e-03  -4.616
## Receiver_PositionFB -1.746e-02  4.078e-03  -4.282
## Receiver_PositionRB -1.680e-02  9.787e-04 -17.161
```

```
## Receiver_PositionTE -3.643e-03  1.007e-03  -3.618
## PassLocationmiddle   5.609e-03  8.982e-04   6.245
## PassLocationright    4.035e-05  7.806e-04   0.052
## Rush_EPA_Att         8.770e-04  6.603e-03   0.133
##
## Correlation of Fixed Effects:
##             (Intr) Hm_Ind Shtg_I N_Hd_I QBHit  Rc_PFB Rc_PRB Rc_PTE PssLctnm
## Home_Ind    -0.205
## Shotgun_Ind -0.369  0.024
## No_Hddl_Ind  0.011 -0.036 -0.083
## QBHit       -0.061  0.009  0.000  0.015
## Rcvr_PstnFB -0.086  0.000  0.097  0.005  0.005
## Rcvr_PstnRB -0.159  0.003 -0.020  0.011 -0.001  0.085
## Rcvr_PstnTE -0.157 -0.003  0.044  0.006 -0.002  0.072  0.286
## PssLctnmddl -0.165 -0.006 -0.036 -0.010 -0.002 -0.001 -0.005 -0.066
## PssLctnrght -0.231  0.011  0.029  0.007  0.006  0.004 -0.001 -0.037  0.445
## Rsh_EPA_Att  0.798 -0.011 -0.004  0.054  0.004 -0.011 -0.008  0.001  0.006
##             PssLctnr
## Home_Ind
## Shotgun_Ind
## No_Hddl_Ind
## QBHit
## Rcvr_PstnFB
## Rcvr_PstnRB
## Rcvr_PstnTE
## PssLctnmddl
## PssLctnrght
## Rsh_EPA_Att  0.000
## optimizer (nloptwrap) convergence code: 0 (OK)
## boundary (singular) fit: see help('isSingular')
```

```
print("\nEP YAC Model:")
```

```
## [1] "\nEP YAC Model:"
```

```
print(summary(nfl2017_wp$yac_model))
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: yacWPA_Result ~ Home_Ind + Shotgun_Ind + No_Huddle_Ind + QBHit +
##     AirYards * Receiver_Position + PassLocation + Rush_EPA_Att +
##     (1 | Passer_ID_Name) + (1 | Receiver_ID_Name) + (1 | DefensiveTeam)
##
## REML criterion at convergence: -57671.7
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -22.2918  -0.3458  -0.0106   0.2653  21.9102
##
## Random effects:
##  Groups           Name        Variance  Std.Dev.
##  Receiver_ID_Name (Intercept) 7.257e-06 0.0026940
##  Passer_ID_Name   (Intercept) 2.982e-06 0.0017268
##  DefensiveTeam    (Intercept) 9.300e-07 0.0009644
##  Residual                     1.932e-03 0.0439495
## Number of obs: 16980, groups:
```

```
## Receiver_ID_Name, 291; Passer_ID_Name, 47; DefensiveTeam, 32
##
## Fixed effects:
##                                 Estimate Std. Error t value
## (Intercept)                    8.666e-03  1.710e-03   5.067
## Home_Ind                       7.602e-04  6.786e-04   1.120
## Shotgun_Ind                   -2.887e-03  8.355e-04  -3.456
## No_Huddle_Ind                  7.060e-04  1.114e-03   0.634
## QBHit                         -9.936e-03  1.202e-03  -8.268
## AirYards                      -6.118e-04  4.143e-05 -14.768
## Receiver_PositionFB            5.855e-03  4.648e-03   1.260
## Receiver_PositionRB            1.173e-02  1.081e-03  10.853
## Receiver_PositionTE            3.174e-03  1.324e-03   2.397
## PassLocationmiddle            -1.138e-04  8.968e-04  -0.127
## PassLocationright             -1.356e-03  7.779e-04  -1.744
## Rush_EPA_Att                   7.827e-03  5.801e-03   1.349
## AirYards:Receiver_PositionFB -7.888e-04  8.503e-04  -0.928
## AirYards:Receiver_PositionRB -7.687e-04  1.377e-04  -5.584
## AirYards:Receiver_PositionTE -3.466e-04  1.032e-04  -3.361
##
## Correlation matrix not shown by default, as p = 15 > 12.
## Use print(summary(nfl2017_wp$yac_model), correlation=TRUE)  or
##     vcov(summary(nfl2017_wp$yac_model))        if you need it
```

```
print("\nEP Rush Model:")
```

```
## [1] "\nEP Rush Model:"
```

```
print(summary(nfl2017_wp$qb_rush_model))
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: WPA ~ Home_Ind + Shotgun_Ind + No_Huddle_Ind + Pass_EPA_Att +
##     (1 | Rusher_ID_Name) + (1 | DefensiveTeam)
##
## REML criterion at convergence: -7136.9
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -12.3020  -0.4529   0.0230   0.4039   7.2106
##
## Random effects:
##  Groups         Name        Variance  Std.Dev.
##  Rusher_ID_Name (Intercept) 1.584e-04 0.012586
##  DefensiveTeam  (Intercept) 1.827e-05 0.004275
##  Residual                   2.942e-03 0.054241
## Number of obs: 2423, groups:  Rusher_ID_Name, 47; DefensiveTeam, 32
##
## Fixed effects:
##                 Estimate Std. Error t value
## (Intercept)   -0.0171379  0.0039912  -4.294
## Home_Ind       0.0024912  0.0022398   1.112
## Shotgun_Ind   -0.0042987  0.0026054  -1.650
## No_Huddle_Ind  0.0009219  0.0042054   0.219
## Pass_EPA_Att  -0.0079531  0.0177247  -0.449
##
```

```
## Correlation of Fixed Effects:
##             (Intr) Hm_Ind Shtg_I N_Hd_I
## Home_Ind    -0.285
## Shotgun_Ind -0.491  0.018
## No_Hddl_Ind -0.045 -0.056 -0.049
## Pss_EPA_Att -0.573 -0.001  0.009 -0.007
```

Analyzing EPA-Based and WPA-Based Results

```
# Combine EPA-Based WAR and WPA-BASED WAR results together #
combine_epa_wpa = function(data_ep, data_wp) {
  subset_cols = c("Player_ID_Name","air_WAR","yac_WAR","rush_WAR","total_WAR")
  WAR_ep = data_ep[, subset_cols]
  WAR_wp = data_wp[, subset_cols]

  WAR_columns = grepl("_WAR$", names(WAR_ep))
  names(WAR_ep)[WAR_columns] = paste0(names(WAR_ep)[WAR_columns], "_EP")
  names(WAR_wp)[WAR_columns] = paste0(names(WAR_wp[WAR_columns]), "_WP")

  WAR_all = left_join(WAR_ep, WAR_wp, by = "Player_ID_Name")
  return (WAR_all)
}

qb_WAR_all = combine_epa_wpa(qb_table_ep, qb_table_wp)
rb_WAR_all = combine_epa_wpa(rb_table_ep, rb_table_wp)
wr_WAR_all = combine_epa_wpa(wr_table_ep, wr_table_wp)
te_WAR_all = combine_epa_wpa(te_table_ep, te_table_wp)

positions = c("QB", "RB", "WR", "TE")
data_list = list(qb_WAR_all=qb_WAR_all, rb_WAR_all=rb_WAR_all,
                 wr_WAR_all=wr_WAR_all, te_WAR_all=te_WAR_all)

for (i in 1:length(positions)) {
  df = data_list[[i]]
  pos = positions[i]
  print(paste0("There are ", dim(df)[1], " ", pos, "s in the 2017 season."))
}
```

```
## [1] "There are 71 QBs in the 2017 season."
## [1] "There are 148 RBs in the 2017 season."
## [1] "There are 201 WRs in the 2017 season."
## [1] "There are 109 TEs in the 2017 season."
```

```
# knable(qb_WAR_all)
```

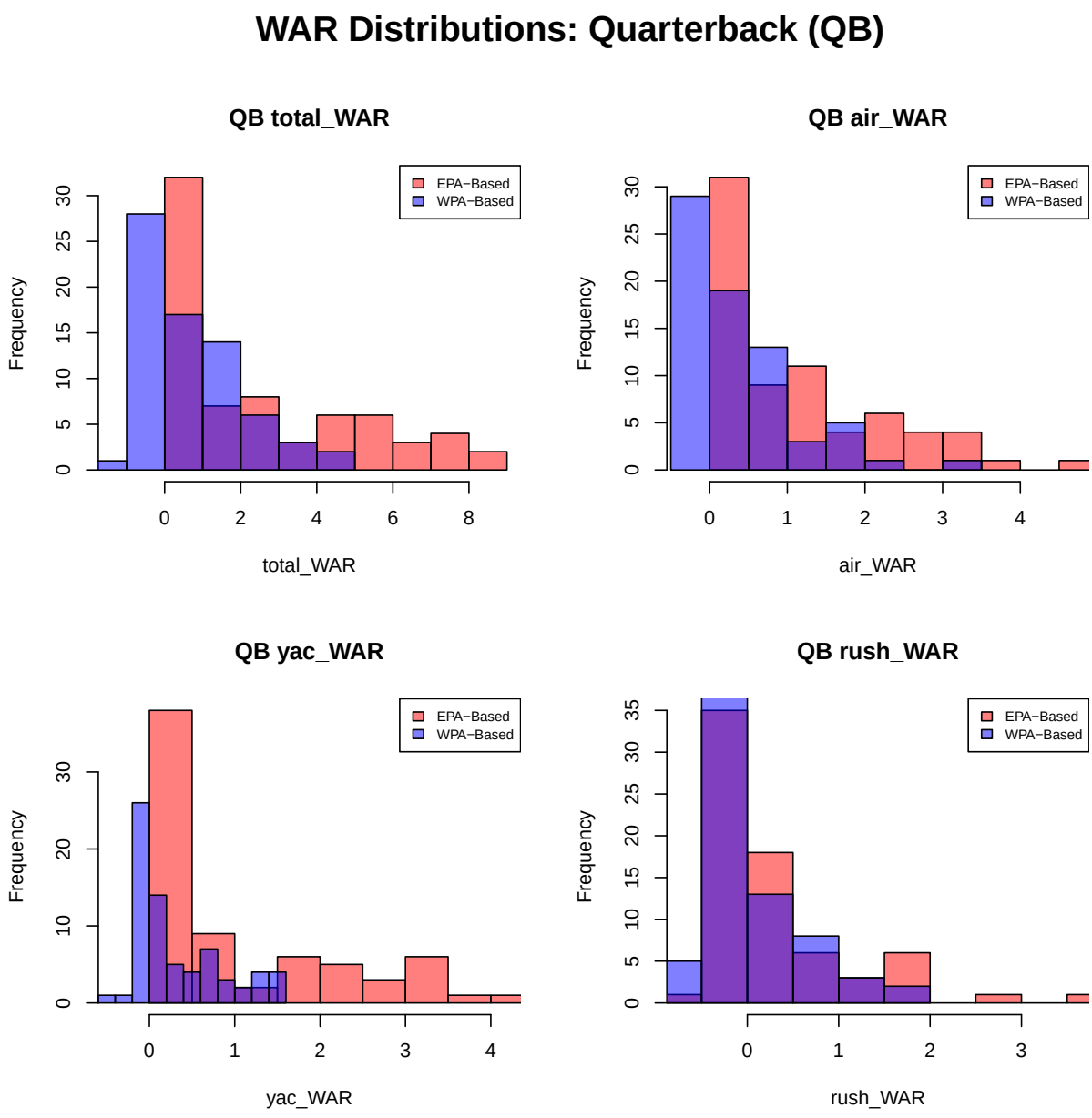
1) Distributions of EPA-Based and WPA-Based WAR By Player Position

```
plot_WAR_distributions = function(data_WAR, position, plot_title) {
  print(summary(qb_WAR_all))

  par(mfrow = c(2, 2), oma = c(0, 0, 3, 0))
  # total
  x_range = range(c(data_WAR["total_WAR_EP"], data_WAR["total_WAR_WP"]))
```

```r
  hist(data_WAR[["total_WAR_EP"]], col = rgb(1, 0, 0, 0.5),
       xlim = c(x_range[1], x_range[2]),
       main = paste(position, "total_WAR"), xlab = "total_WAR")
  hist(data_WAR[["total_WAR_WP"]], col = rgb(0, 0, 1, 0.5), add = TRUE)
  legend("topright", legend = c("EPA-Based", "WPA-Based"),
         fill = c(rgb(1, 0, 0, 0.5), rgb(0, 0, 1, 0.5)), cex = 0.75)

  # air
  x_range = range(c(data_WAR["air_WAR_EP"], data_WAR["air_WAR_WP"]))
  hist(data_WAR[["air_WAR_EP"]], col = rgb(1, 0, 0, 0.5),
       xlim = c(x_range[1], x_range[2]),
       main = paste(position, "air_WAR"), xlab = "air_WAR")
  hist(data_WAR[["air_WAR_WP"]], col = rgb(0, 0, 1, 0.5), add = TRUE)
  legend("topright", legend = c("EPA-Based", "WPA-Based"),
         fill = c(rgb(1, 0, 0, 0.5), rgb(0, 0, 1, 0.5)), cex = 0.75)

  # yac
  x_range = range(c(data_WAR["yac_WAR_EP"], data_WAR["yac_WAR_WP"]))
  hist(data_WAR[["yac_WAR_EP"]], col = rgb(1, 0, 0, 0.5),
       xlim = c(x_range[1], x_range[2]),
       main = paste(position, "yac_WAR"), xlab = "yac_WAR")
  hist(data_WAR[["yac_WAR_WP"]], col = rgb(0, 0, 1, 0.5), add = TRUE)
  legend("topright", legend = c("EPA-Based", "WPA-Based"),
         fill = c(rgb(1, 0, 0, 0.5), rgb(0, 0, 1, 0.5)), cex = 0.75)

  # rush
  x_range = range(c(data_WAR["rush_WAR_EP"], data_WAR["rush_WAR_WP"]))
  hist(data_WAR[["rush_WAR_EP"]], col = rgb(1, 0, 0, 0.5),
       xlim = c(x_range[1], x_range[2]),
       main = paste(position, "rush_WAR"), xlab = "rush_WAR")
  hist(data_WAR[["rush_WAR_WP"]], col = rgb(0, 0, 1, 0.5), add = TRUE)
  legend("topright", legend = c("EPA-Based", "WPA-Based"),
         fill = c(rgb(1, 0, 0, 0.5), rgb(0, 0, 1, 0.5)), cex = 0.75)

  mtext(plot_title, side = 3, outer = TRUE, cex = 1.5, line = 1, font = 2)

  par(mfrow = c(1, 1))
}

plot_WAR_distributions(qb_WAR_all,"QB","WAR Distributions: Quarterback (QB)")
```

```
##  Player_ID_Name       air_WAR_EP       yac_WAR_EP      rush_WAR_EP
##  Length:71          Min.   :0.0000   Min.   :0.0000   Min.   :-0.5720
##  Class :character   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 0.0000
##  Mode  :character   Median :0.6578   Median :0.4494   Median : 0.0000
##                     Mean   :1.0659   Mean   :0.9923   Mean   : 0.3505
##                     3rd Qu.:1.8154   3rd Qu.:1.8290   3rd Qu.: 0.3944
##                     Max.   :4.6853   Max.   :4.1656   Max.   : 3.5695
##   total_WAR_EP     air_WAR_WP        yac_WAR_WP        rush_WAR_WP
##  Min.   :0.000   Min.   :-0.3496   Min.   :-0.41518   Min.   :-0.70930
##  1st Qu.:0.000   1st Qu.: 0.0000   1st Qu.: 0.00000   1st Qu.:-0.03624
##  Median :1.283   Median : 0.2064   Median : 0.09803   Median : 0.00000
##  Mean   :2.409   Mean   : 0.4371   Mean   : 0.35045   Mean   : 0.09968
##  3rd Qu.:4.391   3rd Qu.: 0.6953   3rd Qu.: 0.61769   3rd Qu.: 0.11754
```

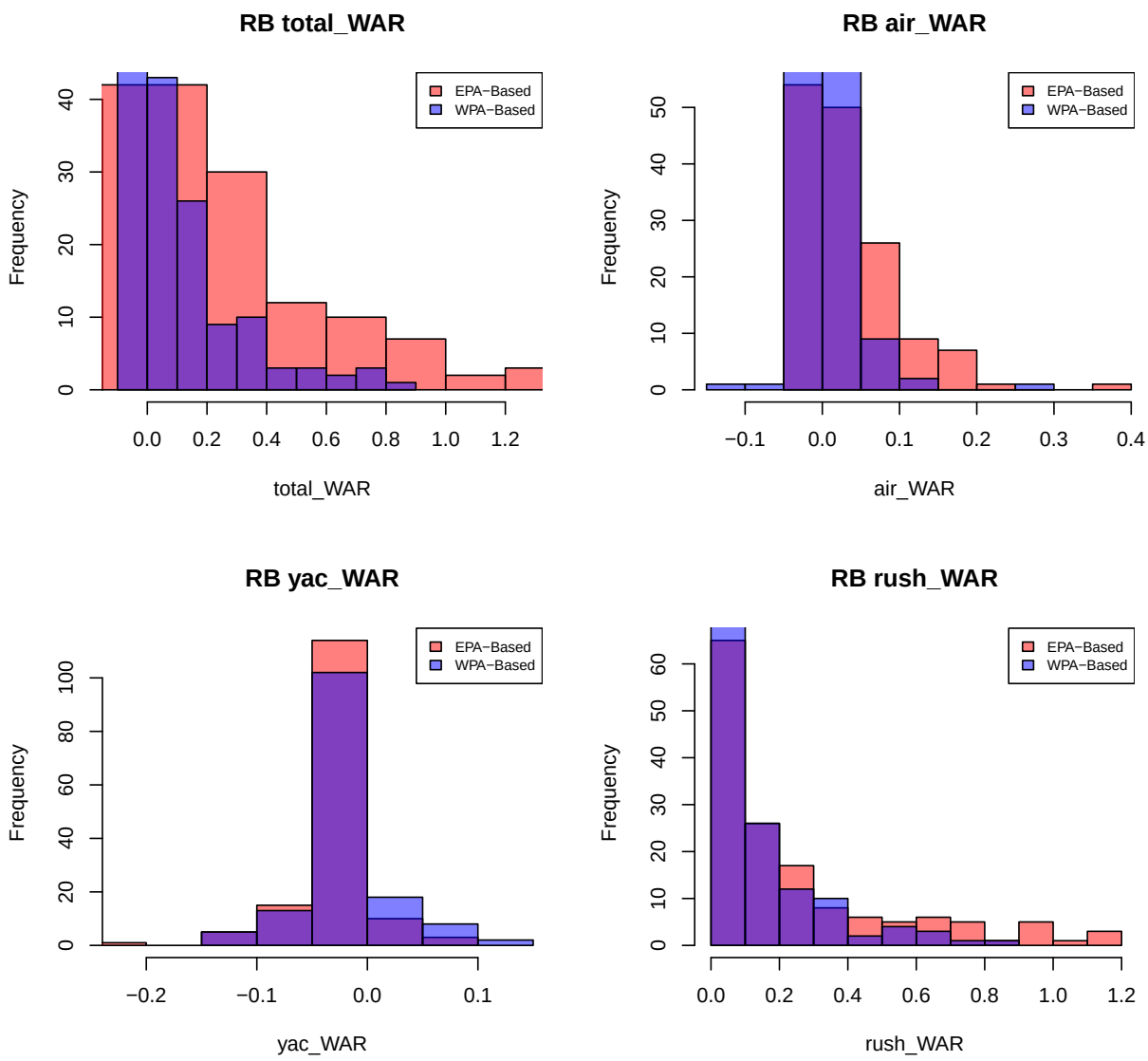
```
##  Max.   :8.946   Max.   : 3.3090   Max.   : 1.54413   Max.   : 1.64211
##   total_WAR_WP
##  Min.   :-1.3334
##  1st Qu.: 0.0000
##  Median : 0.3561
##  Mean   : 0.8872
##  3rd Qu.: 1.3298
##  Max.   : 4.6064
```




```
plot_WAR_distributions(rb_WAR_all,"RB","WAR Distributions: Running back (RB)")
```

```
##  Player_ID_Name       air_WAR_EP       yac_WAR_EP      rush_WAR_EP
##  Length:71          Min.   :0.0000   Min.   :0.0000   Min.   :-0.5720
##  Class :character   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 0.0000
##  Mode  :character   Median :0.6578   Median :0.4494   Median : 0.0000
```

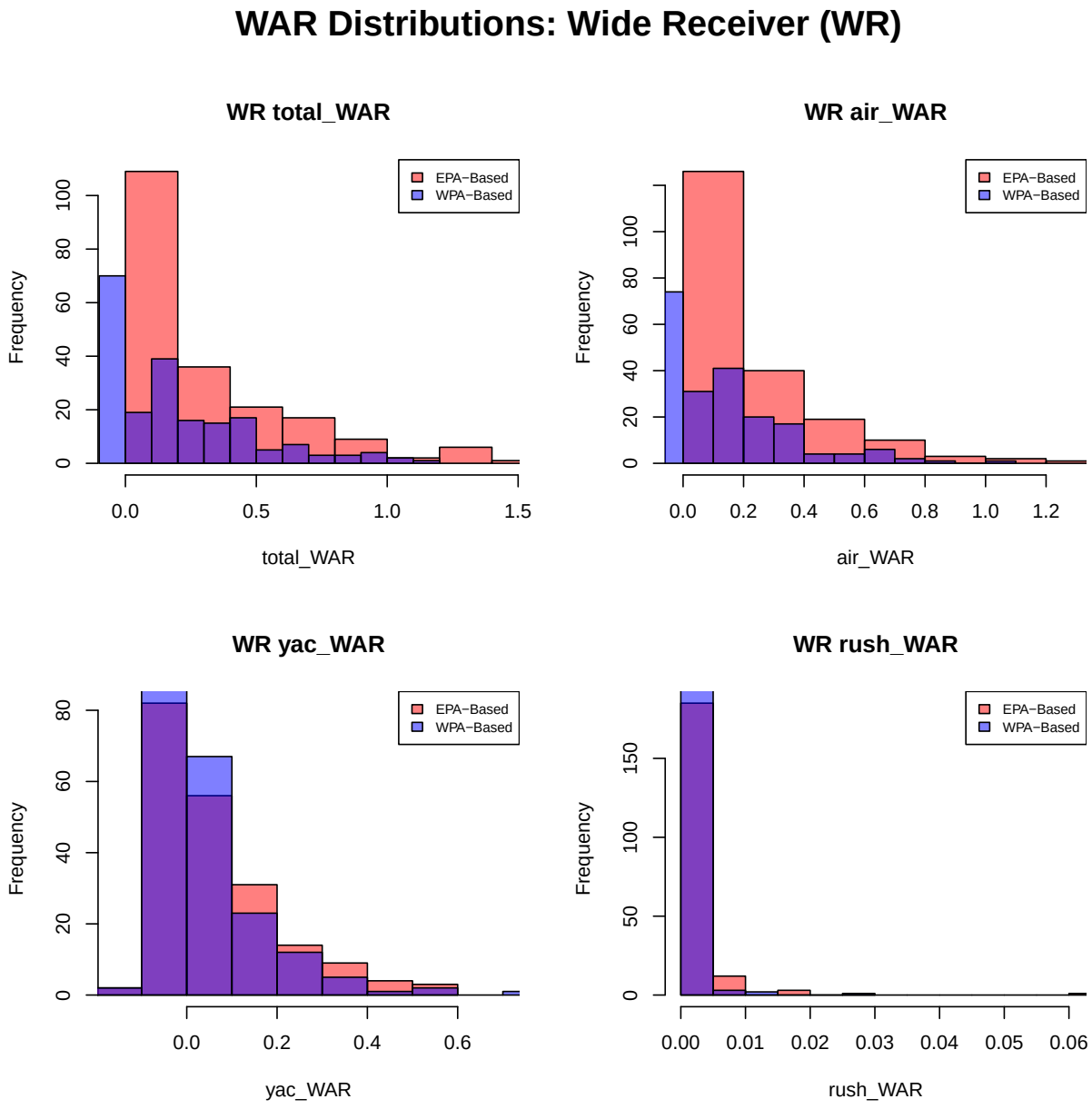
```
##                    Mean   :1.0659   Mean   :0.9923   Mean   : 0.3505
##                    3rd Qu.:1.8154   3rd Qu.:1.8290   3rd Qu.: 0.3944
##                    Max.   :4.6853   Max.   :4.1656   Max.   : 3.5695
##   total_WAR_EP     air_WAR_WP        yac_WAR_WP        rush_WAR_WP
##  Min.   :0.000   Min.   :-0.3496   Min.   :-0.41518   Min.   :-0.70930
##  1st Qu.:0.000   1st Qu.: 0.0000   1st Qu.: 0.00000   1st Qu.:-0.03624
##  Median :1.283   Median : 0.2064   Median : 0.09803   Median : 0.00000
##  Mean   :2.409   Mean   : 0.4371   Mean   : 0.35045   Mean   : 0.09968
##  3rd Qu.:4.391   3rd Qu.: 0.6953   3rd Qu.: 0.61769   3rd Qu.: 0.11754
##  Max.   :8.946   Max.   : 3.3090   Max.   : 1.54413   Max.   : 1.64211
##   total_WAR_WP
##  Min.   :-1.3334
##  1st Qu.: 0.0000
##  Median : 0.3561
##  Mean   : 0.8872
##  3rd Qu.: 1.3298
##  Max.   : 4.6064
```




```
plot_WAR_distributions(wr_WAR_all,"WR","WAR Distributions: Wide Receiver (WR)")
```

```
##  Player_ID_Name       air_WAR_EP       yac_WAR_EP      rush_WAR_EP
##  Length:71          Min.   :0.0000   Min.   :0.0000   Min.   :-0.5720
##  Class :character   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 0.0000
##  Mode  :character   Median :0.6578   Median :0.4494   Median : 0.0000
##                     Mean   :1.0659   Mean   :0.9923   Mean   : 0.3505
##                     3rd Qu.:1.8154   3rd Qu.:1.8290   3rd Qu.: 0.3944
##                     Max.   :4.6853   Max.   :4.1656   Max.   : 3.5695
##   total_WAR_EP     air_WAR_WP        yac_WAR_WP        rush_WAR_WP
##  Min.   :0.000   Min.   :-0.3496   Min.   :-0.41518   Min.   :-0.70930
##  1st Qu.:0.000   1st Qu.: 0.0000   1st Qu.: 0.00000   1st Qu.:-0.03624
##  Median :1.283   Median : 0.2064   Median : 0.09803   Median : 0.00000
##  Mean   :2.409   Mean   : 0.4371   Mean   : 0.35045   Mean   : 0.09968
```

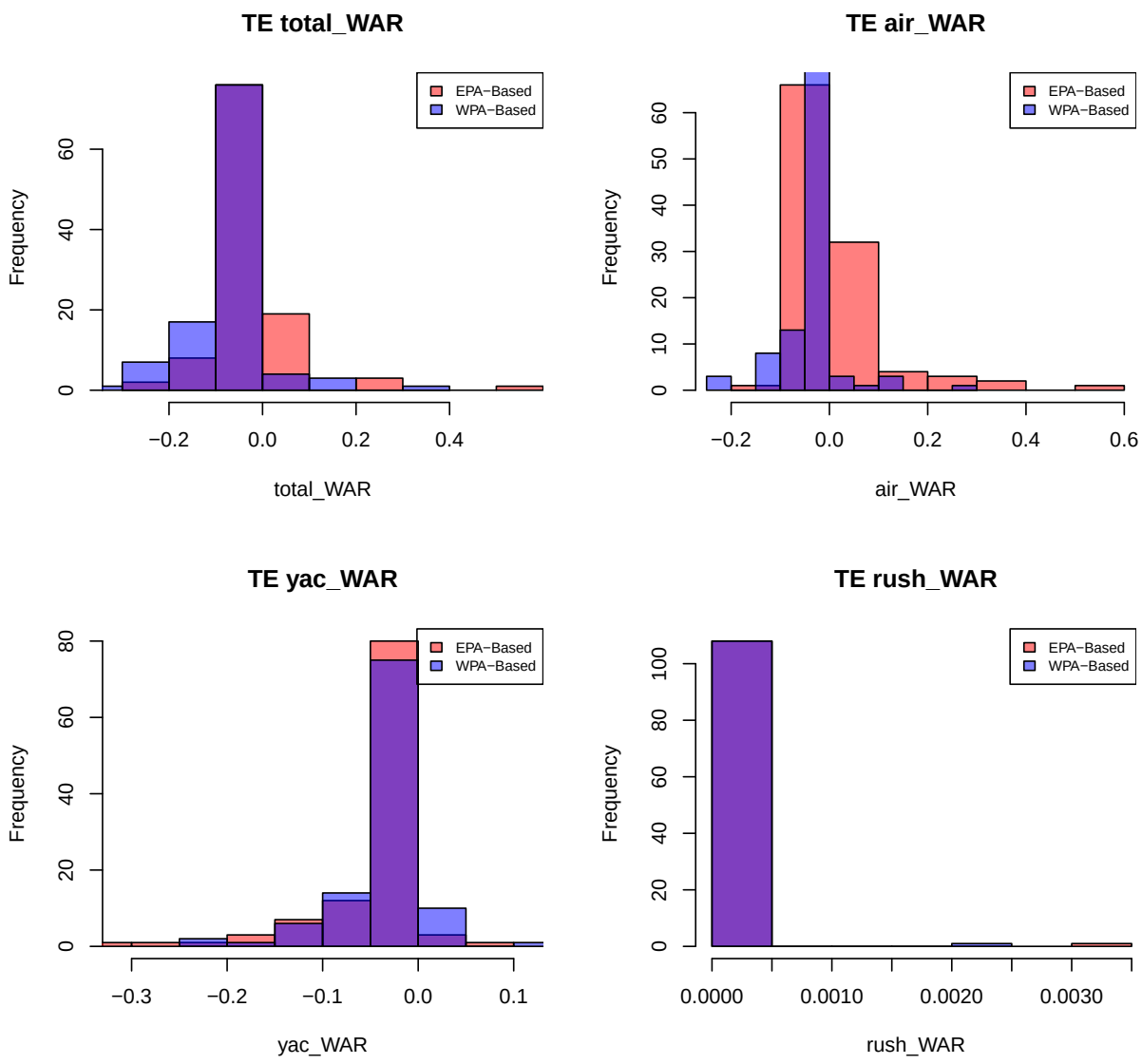
```
##  3rd Qu.:4.391   3rd Qu.: 0.6953   3rd Qu.: 0.61769   3rd Qu.: 0.11754
##  Max.   :8.946   Max.   : 3.3090   Max.   : 1.54413   Max.   : 1.64211
##   total_WAR_WP
##  Min.   :-1.3334
##  1st Qu.: 0.0000
##  Median : 0.3561
##  Mean   : 0.8872
##  3rd Qu.: 1.3298
##  Max.   : 4.6064
```




```
plot_WAR_distributions(te_WAR_all,"TE","WAR Distributions: Tight End (TE)")
```

```
##  Player_ID_Name       air_WAR_EP       yac_WAR_EP      rush_WAR_EP
##  Length:71          Min.   :0.0000   Min.   :0.0000   Min.   :-0.5720
##  Class :character   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.: 0.0000
```

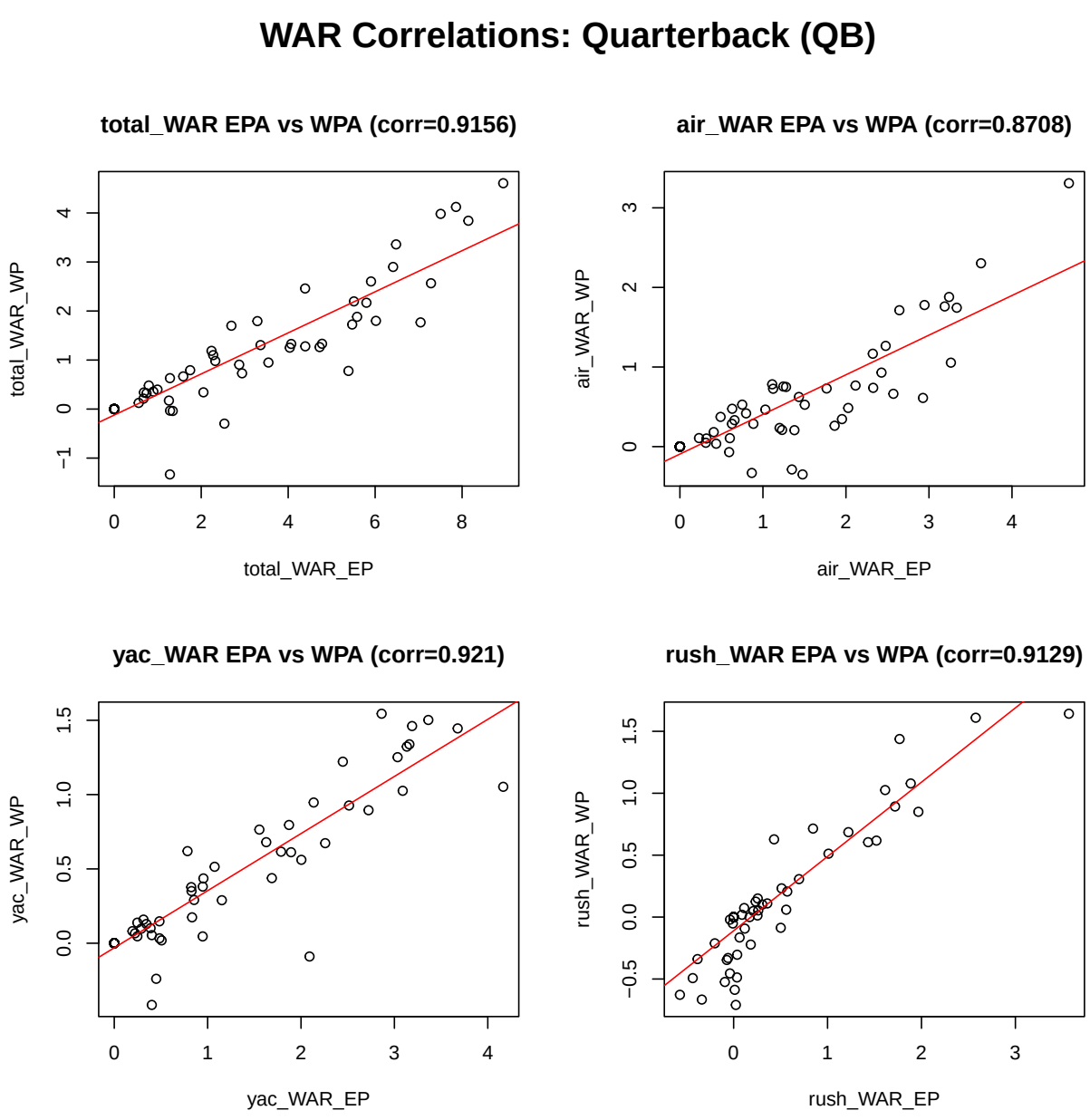
```
##  Mode  :character   Median :0.6578   Median :0.4494   Median : 0.0000
##                     Mean   :1.0659   Mean   :0.9923   Mean   : 0.3505
##                     3rd Qu.:1.8154   3rd Qu.:1.8290   3rd Qu.: 0.3944
##                     Max.   :4.6853   Max.   :4.1656   Max.   : 3.5695
##   total_WAR_EP     air_WAR_WP        yac_WAR_WP        rush_WAR_WP
##  Min.   :0.000   Min.   :-0.3496   Min.   :-0.41518   Min.   :-0.70930
##  1st Qu.:0.000   1st Qu.: 0.0000   1st Qu.: 0.00000   1st Qu.:-0.03624
##  Median :1.283   Median : 0.2064   Median : 0.09803   Median : 0.00000
##  Mean   :2.409   Mean   : 0.4371   Mean   : 0.35045   Mean   : 0.09968
##  3rd Qu.:4.391   3rd Qu.: 0.6953   3rd Qu.: 0.61769   3rd Qu.: 0.11754
##  Max.   :8.946   Max.   : 3.3090   Max.   : 1.54413   Max.   : 1.64211
##   total_WAR_WP
##  Min.   :-1.3334
##  1st Qu.: 0.0000
##  Median : 0.3561
##  Mean   : 0.8872
##  3rd Qu.: 1.3298
##  Max.   : 4.6064
```



2) Correlations of EPA-Based and WPA-Based WAR By Player Position

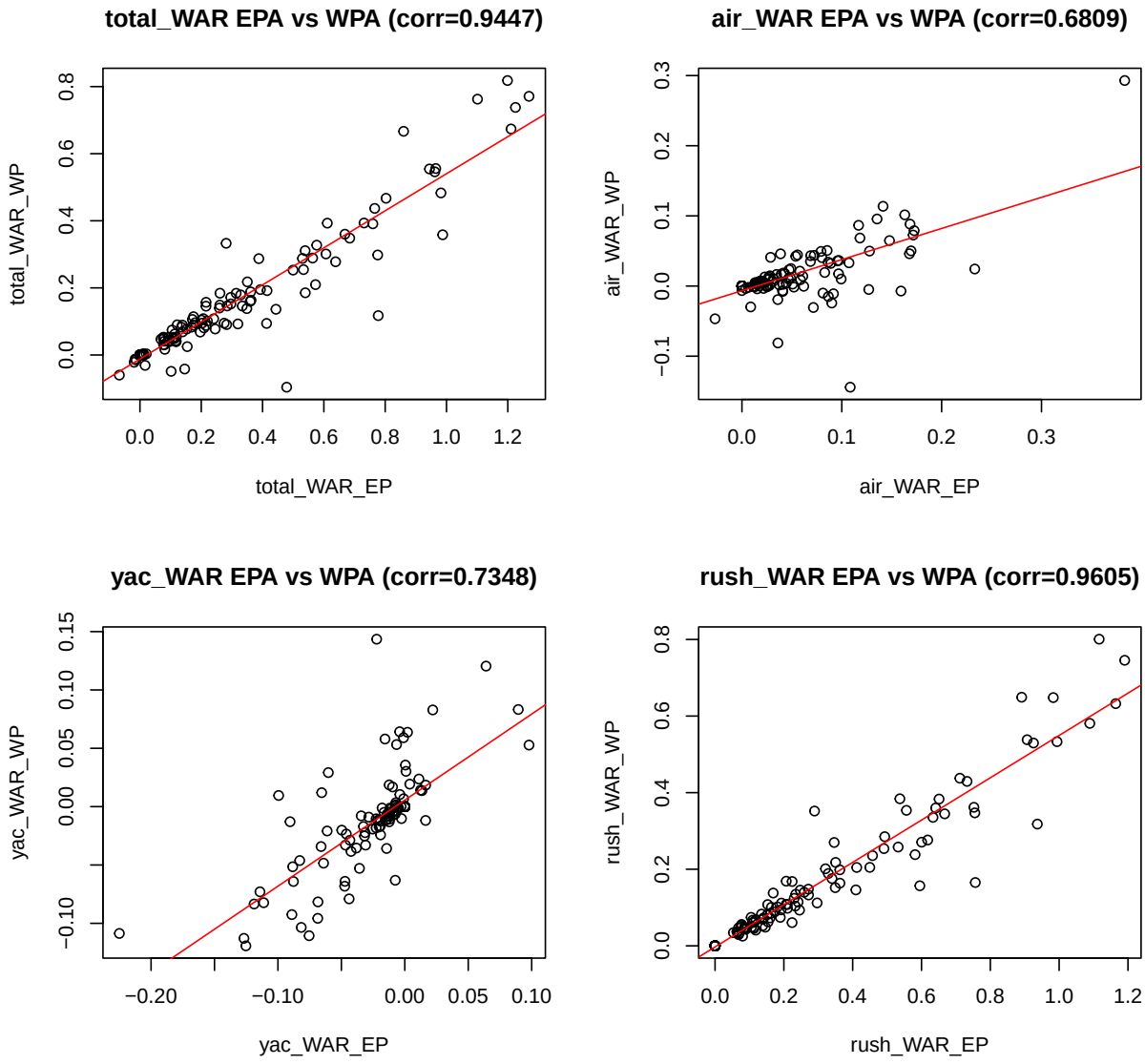
```
plot_WAR_corr = function(data_WAR, plot_title) {
  par(mfrow = c(2, 2), oma = c(0, 0, 3, 0))
  WAR_list = c("total_WAR", "air_WAR", "yac_WAR", "rush_WAR")
  for (i in 1:length(WAR_list)) {
    col_ep = paste0(WAR_list[i], "_EP")
    col_wp = paste0(WAR_list[i], "_WP")
    corr = round(cor(data_WAR[[col_ep]], data_WAR[[col_wp]],
                     method = "pearson"), 4)
    plot(data_WAR[[col_ep]], data_WAR[[col_wp]],
         main=paste0(WAR_list[i], " EPA vs WPA (corr=", corr, ")"),
         xlab=col_ep, ylab=col_wp)
```

```
    model = lm(as.formula(paste(col_wp, "~", col_ep)), data=data_WAR)
    abline(model, col = "red", lwd = 1)
  }
  mtext(plot_title, side = 3, outer = TRUE, cex = 1.5, line = 1, font = 2)
  par(mfrow = c(1, 1))
}

plot_WAR_corr(qb_WAR_all, "WAR Correlations: Quarterback (QB)")
```

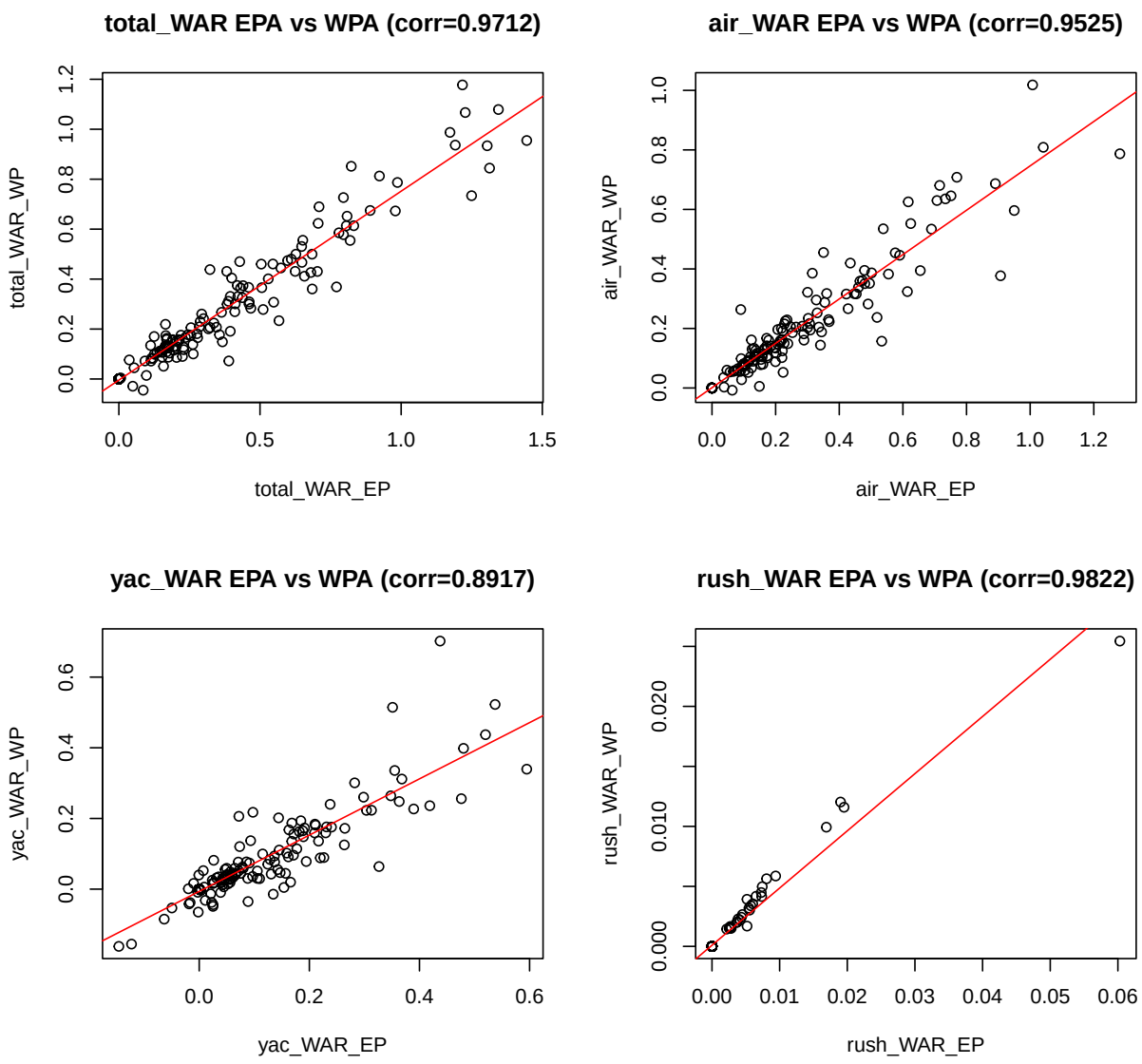



```
plot_WAR_corr(rb_WAR_all, "WAR Correlations: Running back (RB)")
```

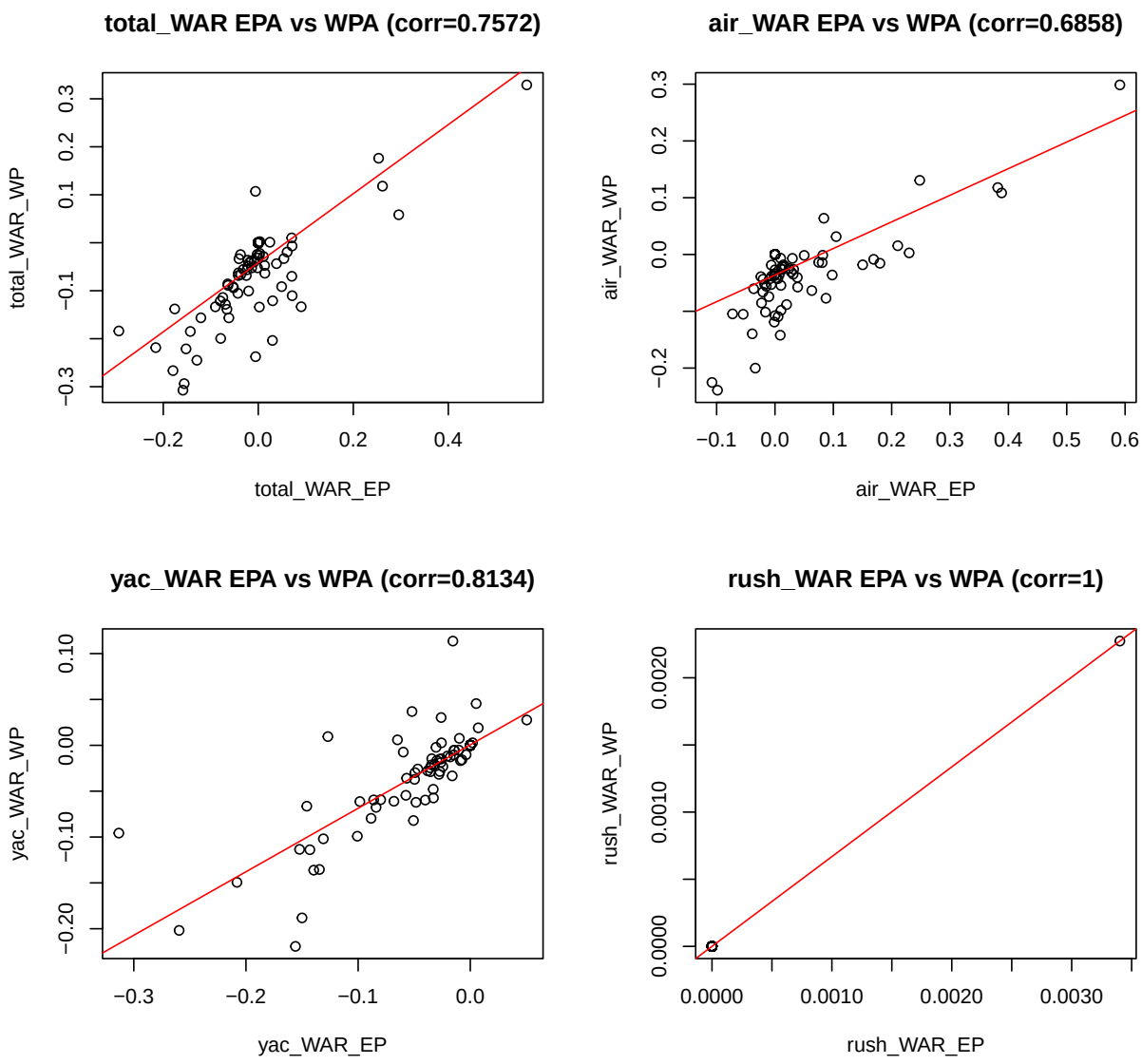
WAR Correlations: Running back (RB)




```
plot_WAR_corr(wr_WAR_all, "WAR Correlations: Wide Receiver (WR)")
```




```
plot_WAR_corr(te_WAR_all, "WAR Correlations: Tight End (TE)")
```

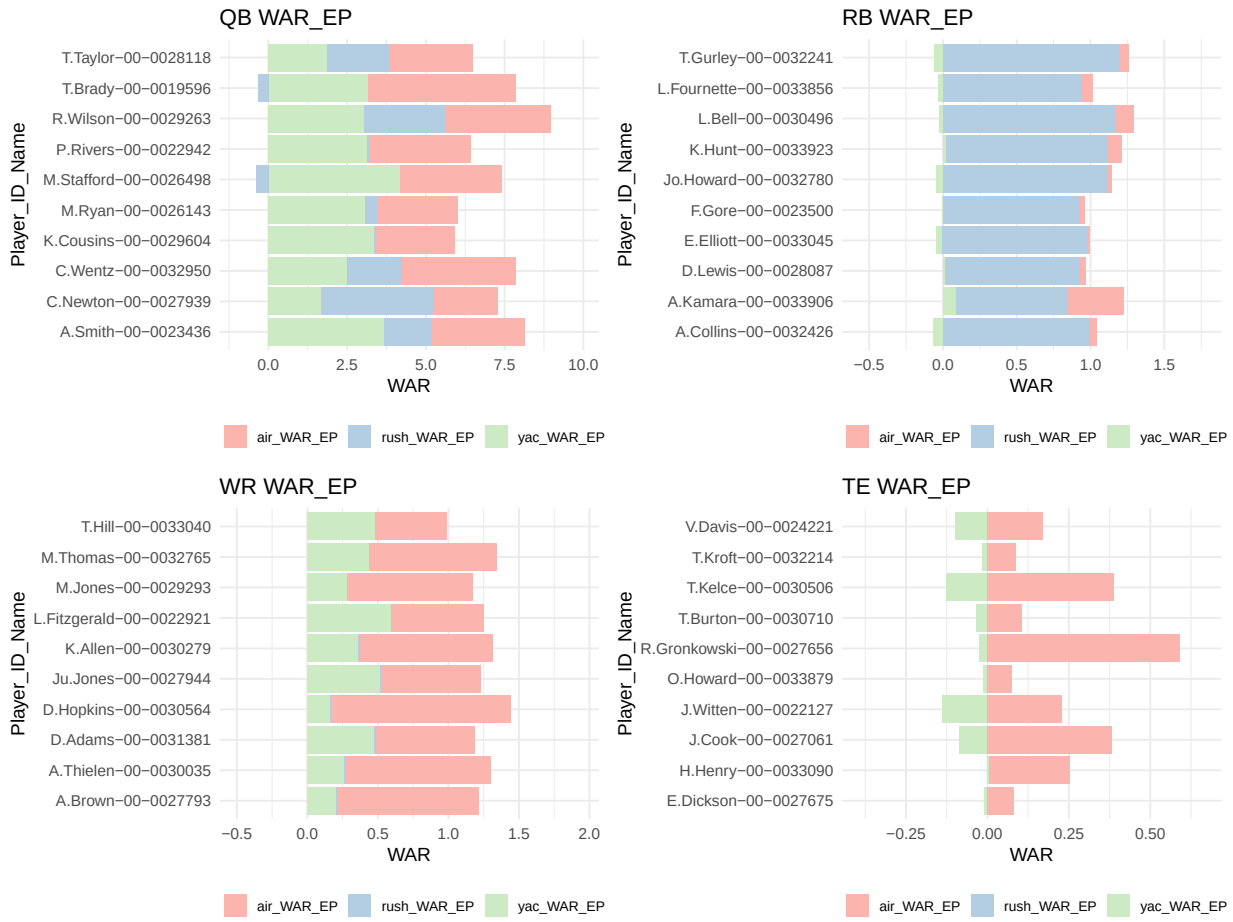


3) Top 10 Players Based on total_WAR By Player Position

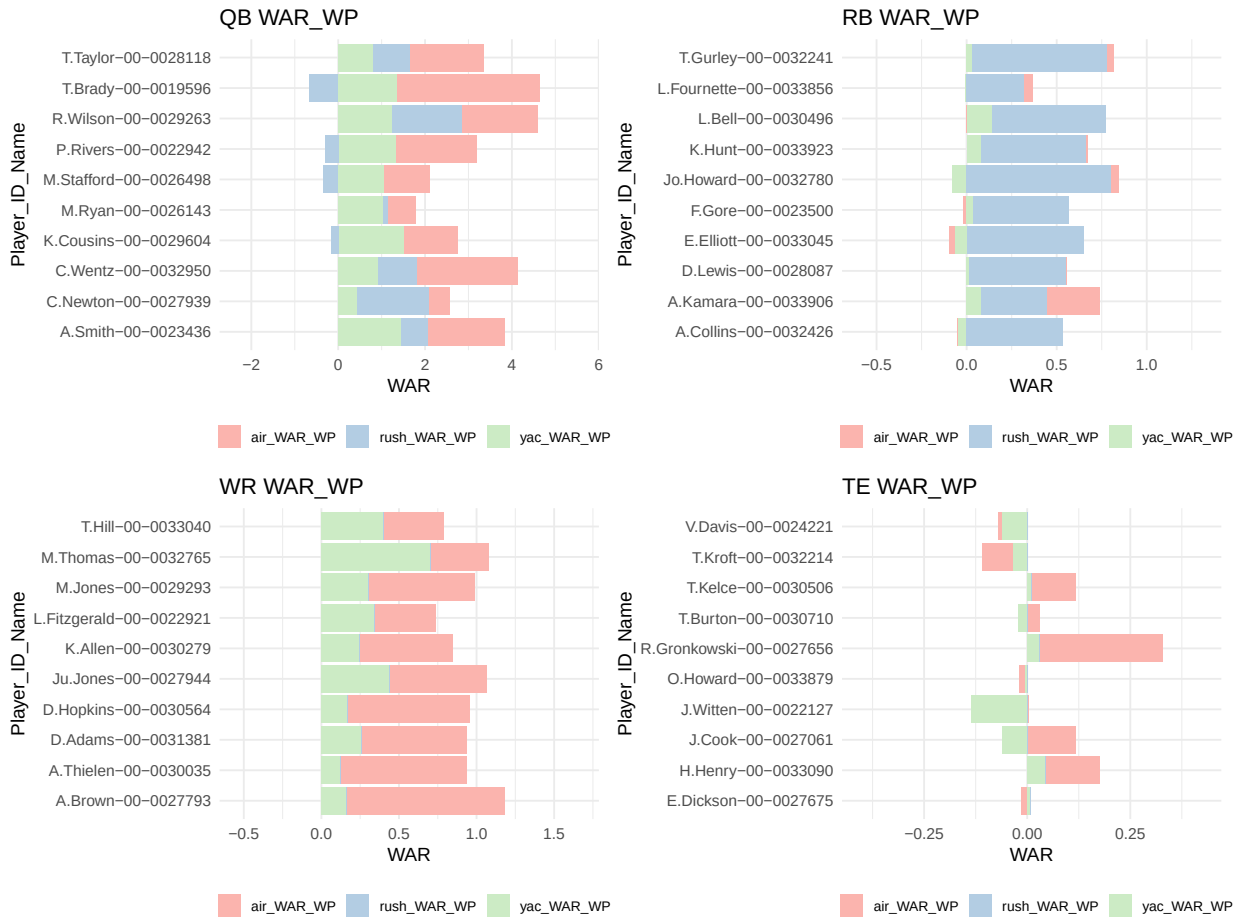
```
# Top 10 players based on total WAR

# Order data by descending order to match the plot's bottom-to-top fill
library(ggplot2)
library(patchwork)

plot_top10_players = function(positions, margins_rt, data_list,
                              plot_title, ep=TRUE) {
  plot_list = list()
  for (i in 1:length(positions)) {
    data_all = data_list[[i]]
```

```
  data_top10 = head(data_all[order(-data_all[["total_WAR_EP"]]),], 10)
  if (ep==TRUE) {
    data_top10 = data_top10[c("Player_ID_Name", "air_WAR_EP",
                              "yac_WAR_EP","rush_WAR_EP")]
    data_long = pivot_longer(data_top10, cols = -Player_ID_Name,
                             names_to = "Component",
                             values_to = "WAR")
    p = ggplot(data_long, aes(x = Player_ID_Name, y = WAR, fill = Component)) +
      geom_bar(stat = "identity", position = "stack") +
      labs(title = paste(positions[i], "WAR_EP"),
           x = "Player_ID_Name", y = "WAR") +
      ylim(min(data_all[["total_WAR_EP"]]) - margins_rt[i],
           max(data_all[["total_WAR_EP"]]) + margins_rt[i]) +
      coord_flip() + # Flip coordinates for horizontal bars +
      theme_minimal() +
      scale_fill_brewer(palette = "Pastel1") +
      theme(legend.position = "bottom", legend.key.size = unit(0.5, "cm"),
            legend.text = element_text(size = 8),
            legend.title = element_text(size = 0, face = "bold"))
    # p = ggplot(data_top10, aes(x = reorder(Player_ID_Name, total_WAR_EP),
    #                    y = total_WAR_EP)) +
    #   geom_bar(stat = "identity", position = "dodge", fill = "lightblue") +
    #   ylim(min(data_all[["total_WAR_EP"]]),
    #        max(data_all[["total_WAR_EP"]]) + margins_rt[i]) +
    #   coord_flip() + # Flip coordinates for horizontal bars +
    #   geom_text(aes(label = round(total_WAR_EP, 4), hjust = -0.1),
    #             color = "black", size = 3) +
    #   labs(title = paste(positions[i], "total_WAR_EP"),
    #        x = "Player_ID_Name", y = "total_WAR_EP") +
    #   theme_minimal()
  } else {
    data_top10 = data_top10[c("Player_ID_Name", "air_WAR_WP",
                              "yac_WAR_WP","rush_WAR_WP")]
    data_long = pivot_longer(data_top10, cols = -Player_ID_Name,
                             names_to = "Component",
                             values_to = "WAR")
    p = ggplot(data_long, aes(x = Player_ID_Name, y = WAR, fill = Component)) +
      geom_bar(stat = "identity", position = "stack") +
      labs(title = paste(positions[i], "WAR_WP"),
           x = "Player_ID_Name", y = "WAR") +
      ylim(min(data_all[["total_WAR_WP"]]) - margins_rt[i],
           max(data_all[["total_WAR_WP"]]) + margins_rt[i]) +
      coord_flip() + # Flip coordinates for horizontal bars +
      theme_minimal() +
      scale_fill_brewer(palette = "Pastel1") +
      theme(legend.position = "bottom", legend.key.size = unit(0.5, "cm"),
            legend.text = element_text(size = 8),
            legend.title = element_text(size = 0, face = "bold"))
    # p = ggplot(data_top10, aes(x = reorder(Player_ID_Name, total_WAR_WP),
    #                    y = total_WAR_WP)) +
    #   geom_bar(stat = "identity", position = "dodge", fill = "lightpink") +
    #   ylim(min(data_all[["total_WAR_WP"]]),
    #        max(data_all[["total_WAR_WP"]]) + margins_rt[i]) +
```

```
      #   coord_flip() + # Flip coordinates for horizontal bars +
      #   geom_text(aes(label = round(total_WAR_WP, 4), hjust = -0.1),
      #             color = "black", size = 3) +
      #   labs(title = paste(positions[i], "total_WAR_WP"),
      #        x = "Player_ID_Name", y = "total_WAR_WP") +
      #   theme_minimal()
    }

    plot_list[[i]] = p
  }

  # Combine the plots
  combined_plot = plot_list[[1]]+plot_list[[2]]+plot_list[[3]]+plot_list[[4]] +
    plot_layout(ncol = 2) +
    plot_annotation(
      title = plot_title,
      theme = theme(
        plot.title = element_text(size = 20, face = "bold", hjust = 0.5)
      )
    )
  print(combined_plot)
}

positions = c("QB", "RB", "WR", "TE")
margins_rt = c(1, 0.5, 0.5, 0.1)

top10_players_ep = plot_top10_players(positions, margins_rt, data_list,
                              "Top 10 Players: Total EPA-Based WAR", ep=TRUE)
```




```
top10_players_wp = plot_top10_players(positions, margins_rt, data_list,
                          "Top 10 Players: Total WPA-Based WAR", ep=FALSE)
```


Top 10 Players: Total WPA−Based WAR
QB WAR_WP
T.Taylor−00−0028118
T.Brady−00−0019596
R.Wilson−00−0029263
P.Rivers−00−0022942
M.Stafford−00−0026498
M.Ryan−00−0026143
K.Cousins−00−0029604
C.Wentz−00−0032950
C.Newton−00−0027939
A.Smith−00−0023436
Player_ID_Name
−2
0
2
4
6
WAR
RB WAR_WP
T.Gurley−00−0032241
L.Fournette−00−0033856
L.Bell−00−0030496
K.Hunt−00−0033923
Jo.Howard−00−0032780
F.Gore−00−0023500
E.Elliott−00−0033045
D.Lewis−00−0028087
A.Kamara−00−0033906
A.Collins−00−0032426
Player_ID_Name
−0.5
0.0
0.5
1.0
WAR
WR WAR_WP
T.Hill−00−0033040
M.Thomas−00−0032765
M.Jones−00−0029293
L.Fitzgerald−00−0022921
K.Allen−00−0030279
Ju.Jones−00−0027944
D.Hopkins−00−0030564
D.Adams−00−0031381
A.Thielen−00−0030035
A.Brown−00−0027793
Player_ID_Name
−0.5
0.0
0.5
1.0
1.5
WAR
TE WAR_WP
V.Davis−00−0024221
T.Kroft−00−0032214
T.Kelce−00−0030506
T.Burton−00−0030710
R.Gronkowski−00−0027656
O.Howard−00−0033879
J.Witten−00−0022127
J.Cook−00−0027061
H.Henry−00−0033090
E.Dickson−00−0027675
Player_ID_Name
−0.25
0.00
0.25
WAR
air_WAR_WP
rush_WAR_WP
yac_WAR_WP