Finalfourr Vignette

```
library(finalfourr)
```

Package can be found at https://github.com/ZayneHMaughan/finalfourr

Introduction

This vignette describes the main functionality of this package which is to predict what team make it to the Sweet Sixteen for a given year of data. The original intent was to predict the Final Four. However, after trail and error no model was close in it's predictive power. This same iteration was done for the Elite Eight. After moving on, the Sweet 16 proved to be more feasible and have a strong predictive capability. Though the predictions are good, this is not the only functionality of this model.

IMPORTANT:

Installing and using the package requires running the below command to fully use the dataset:

```
devtools::install_github("andreweatherman/cbbdata")
```

Predicting Sweet 16

`predict_s16_teams()`

This returns a table with the teams that are predicted to make it to the sweet 16. For this example, I will be using the 2020 season to make a prediction as to what teams would have made it if their season was not ended before the tournament due to Covid.

```
predict_s16_teams(2020)
#> # A tibble: 16 x 2
#> # Groups:   team [16]
#>    team          probs
#>    <fct>         <dbl>
#>  1 Duke          1.00
#>  2 Dayton        1.00
#>  3 Kansas        0.999
#>  4 Gonzaga       0.996
#>  5 Michigan St.  0.995
#>  6 Florida St.   0.994
#>  7 Baylor        0.991
#>  8 Louisville    0.988
```

```
#>  9 Creighton     0.962
#> 10 Ohio St.      0.962
#> 11 San Diego St. 0.923
#> 12 Villanova     0.920
#> 13 Michigan      0.914
#> 14 Oregon        0.912
#> 15 West Virginia 0.852
#> 16 Seton Hall    0.847
```

This vignette also demonstrates how to use several functions from the package to explore NCAA basketball team data, including:

- Comparing two teams (`compare_teams()` and `plot_compare_teams()`)
- Listing bracket teams for a given year (`get_bracket_teams()`)
- Getting team lists by conference (`get_teams_by_conf()`)
- Searching for teams by keyword (`search_team()`)
- Simulating a matchup between two teams (`simulate_matchup()`)
- Flipping a coin between two teams (`flip_a_coin()`)

All functions assume the presence of a dataset like `cbb_data`, containing columns such as `year`, `team`, `CONF`, and `POSTSEASON`.

Comparing Two Teams

`compare_teams()`

Returns offensive and defensive metrics for two teams in a given year:

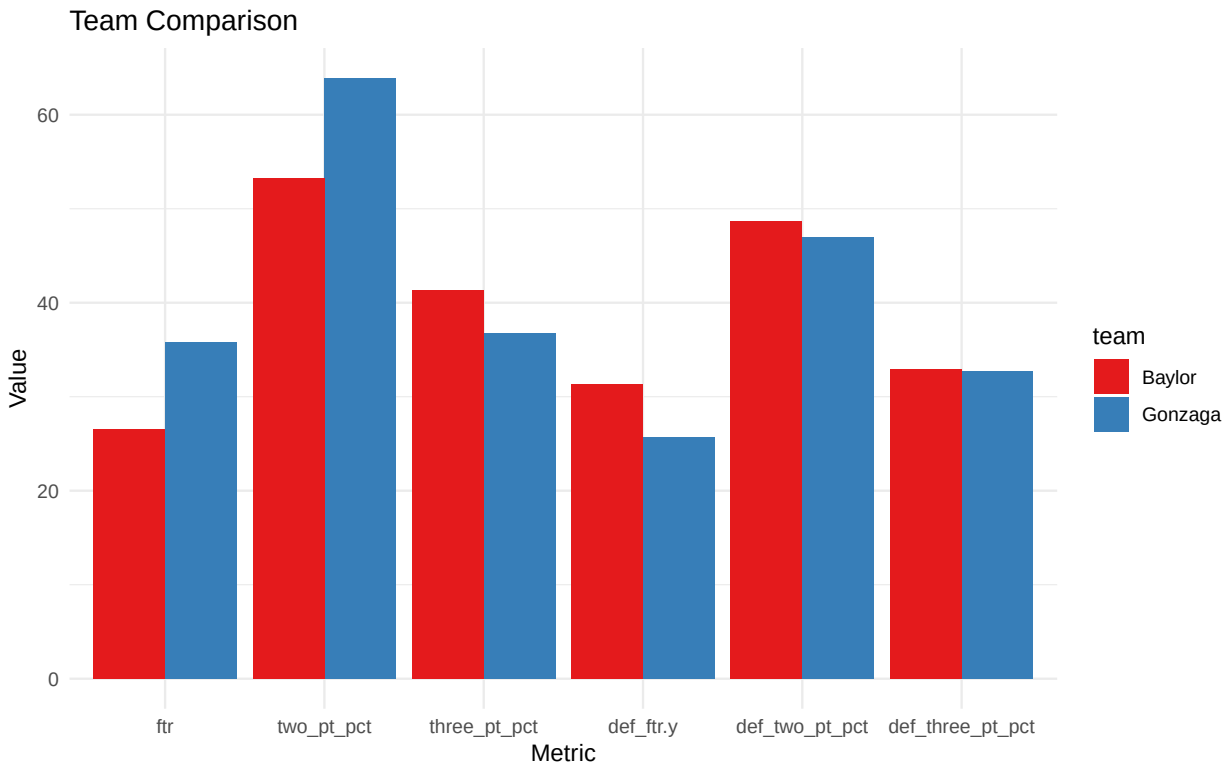
- `ftr`: Free throw rate
- `two_pt_pct`: 2-point percentage
- `three_pt_pct`: 3-point percentage
- `def_ftr.y`: Opponent free throw rate
- `def_two_pt_pct`: Opponent 2-point percentage
- `def_three_pt_pct`: Opponent 3-point percentage

```
comparison <- compare_teams(2021, "Gonzaga", "Baylor")
comparison
#> # A tibble: 2 x 7
#> # Groups:   team [2]
#>   team      ftr two_pt_pct three_pt_pct def_ftr.y def_two_pt_pct def_three_pt_pct
#>   <chr>   <dbl>      <dbl>        <dbl>     <dbl>          <dbl>            <dbl>
#> 1 Gonzaga  35.8       63.9         36.8      25.7           47               32.7
#> 2 Baylor   26.5       53.2         41.3      31.3           48.7             32.9
```

`plot_compare_teams()`

Visualizes the comparison in a bar chart.

```
plot_compare_teams(comparison)
```



Getting Bracket Teams

`get_bracket_teams()`

Returns all teams that made the NCAA tournament in a given year. Teams are identified via the `POSTSEASON` column.

```
bracket_teams <- get_bracket_teams(2023)
head(bracket_teams)
#> [1] "Alabama"     "Arizona"     "Arizona St." "Arkansas"    "Auburn"      "Baylor"
```

Note: Any `POSTSEASON` value not equal to `"0"`, `"N/A"`, or `NA` is considered bracket-eligible.

Getting Teams by Conference

`get_teams_by_conf()`

Returns a named list of conferences and the teams that belonged to each in a given year.

```
conf_teams <- get_teams_by_conf(2018)
names(conf_teams) # List of conference names
#>  [1] "A10"  "ACC"  "AE"   "Amer" "ASun" "B10"  "B12"  "BE"   "BSky" "BSth" "BW"   "CAA"
#> [13] "CUSA" "Horz" "Ivy"  "MAAC" "MAC"  "MEAC" "MVC"  "MWC"  "NEC"  "OVC"  "P12"  "Pat"
#> [25] "SB"   "SC"   "SEC"  "Slnd" "Sum"  "SWAC" "WAC"  "WCC"
head(conf_teams$`Big Ten`) # Teams in the Big Ten
#> NULL
```

Searching for Teams by Keyword using `search_team()`

`search_team()` allows you to quickly find teams whose names contain a specific keyword. This is useful when you're working with this large dataset and want to filter down to teams of interest and available years of data.

For example, we can see all teams with the keyword `utah`:

```
search_team("utah", include_years = TRUE)
#> # A tibble: 5 x 2
#>   team          years
#>   <chr>         <chr>
#> 1 Southern Utah 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024
#> 2 Utah          2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024
#> 3 Utah St.      2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024
#> 4 Utah Tech     2021, 2022, 2023, 2024
#> 5 Utah Valley   2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024
```

Simulating a Matchup Between Two Teams using `simulate_matchup()`

`simulate_matchup()` allows you to simulate head-to-head matchups between two teams in a given season. This can help estimate win probabilities based on adjusted offensive, defensive, and tempo metrics.

For example, here we simulate 1000 games between Utah and Baylor in the 2015 season:

```
set.seed(387)
simulate_matchup("Utah", "Baylor", 2015)
#>
#> Year: 2015
#> Utah vs Baylor
#>
#> Baylor Wins: 547 (54.7%)
#> Utah Wins: 453 (45.3%)
#>
#> Winner: Baylor
```

Flipping a Coin Between Two Teams using `flip_a_coin()`

`flip_a_coin()` is a whimsical tool that stochastically selects a winner between two teams—essentially a virtual coin flip. It provides no analytical value but can be a fun way to break ties or help decide between picking two teams.

For example, here we flip a coin between BYU and Utah:

```
set.seed(348)
flip_a_coin("BYU", "Utah")
#> [=--------------------]   5%[==------------------]  10%[===-----------------]  15%[====----------------
#> The Winner is BYU
```

Notes

- These functions rely on consistent naming in the dataset: columns like `year`, `team`, `CONF`, and `POSTSEASON` must be present.
- Use validation helpers (e.g., `validate_inputs()`, `validate_bracket_inputs()`) to ensure robust function calls.

Session Info

```
sessionInfo()
#> R version 4.5.0 (2025-04-11 ucrt)
#> Platform: x86_64-w64-mingw32/x64
#> Running under: Windows 11 x64 (build 26100)
#>
#> Matrix products: default
#>   LAPACK version 3.12.1
#>
#> locale:
#> [1] LC_COLLATE=English_United States.utf8  LC_CTYPE=English_United States.utf8
#> [3] LC_MONETARY=English_United States.utf8 LC_NUMERIC=C
#> [5] LC_TIME=English_United States.utf8
#>
#> time zone: America/Denver
#> tzcode source: internal
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#> [1] finalfourr_0.0.0.9000 testthat_3.2.3        devtools_2.4.5        usethis_3.1.0
#>
#> loaded via a namespace (and not attached):
#>   [1] RColorBrewer_1.1-3   rstudioapi_0.17.1    jsonlite_2.0.0       magrittr_2.0.3
#>   [5] farver_2.1.2         rmarkdown_2.29       fs_1.6.6             vctrs_0.6.5
#>   [9] memoise_2.0.1        janitor_2.2.1        htmltools_0.5.8.1    curl_6.2.2
#>  [13] xgboost_1.7.9.1      pROC_1.18.5          caret_7.0-1          parallelly_1.43.0
```

```
#>  [17] htmlwidgets_1.6.4     desc_1.4.3           plyr_1.8.9           httr2_1.1.2
#>  [21] lubridate_1.9.4       cachem_1.1.0         mime_0.13            lifecycle_1.0.4
#>  [25] themis_1.0.3          iterators_1.0.14     pkgconfig_2.0.3      Matrix_1.7-3
#>  [29] R6_2.6.1              fastmap_1.2.0        fasttime_1.1-0       future_1.40.0
#>  [33] shiny_1.10.0          snakecase_0.11.1     digest_0.6.37        colorspace_2.1-1
#>  [37] rprojroot_2.0.4       pkgload_1.4.0        labeling_0.4.3       timechange_0.3.0
#>  [41] httr_1.4.7            compiler_4.5.0       remotes_2.5.0        bit64_4.6.0-1
#>  [45] withr_3.0.2           backports_1.5.0      pkgbuild_1.4.7       R.utils_2.13.0
#>  [49] MASS_7.3-65           lava_1.8.1           rappdirs_0.3.3       sessioninfo_1.2.3
#>  [53] ModelMetrics_1.2.2.2  tools_4.5.0          httpuv_1.6.15        future.apply_1.11.3
#>  [57] lintr_3.2.0           nnet_7.3-20          R.oo_1.27.0          glue_1.8.0
#>  [61] nlme_3.1-168          R.cache_0.16.0       cbbdata_0.3.0.9000   promises_1.3.2
#>  [65] grid_4.5.0            rsconnect_1.3.4      reshape2_1.4.4       generics_0.1.3
#>  [69] recipes_1.3.0         gtable_0.3.6         tzdb_0.5.0           R.methodsS3_1.8.2
#>  [73] class_7.3-23          tidyr_1.3.1          data.table_1.17.0    hms_1.1.3
#>  [77] xml2_1.3.8            utf8_1.2.4           foreach_1.5.2        pillar_1.10.2
#>  [81] stringr_1.5.1         later_1.4.2          splines_4.5.0        dplyr_1.1.4
#>  [85] lattice_0.22-6        survival_3.8-3       bit_4.6.0            tidyselect_1.2.1
#>  [89] miniUI_0.1.1.1        knitr_1.50           stats4_4.5.0         xfun_0.52
#>  [93] hardhat_1.4.1         timeDate_4041.110    brio_1.1.5           rex_1.2.1
#>  [97] stringi_1.8.7         lazyeval_0.2.2       yaml_2.3.10          evaluate_1.0.3
#> [101] codetools_0.2-20      tibble_3.2.1         cli_3.6.4            rpart_4.1.24
#> [105] arrow_19.0.1.1        xtable_1.8-4         munsell_0.5.1        styler_1.10.3
#> [109] roxygen2_7.3.2        xmlparsedata_1.0.5   Rcpp_1.0.14          ROSE_0.0-4
#> [113] globals_0.17.0        parallel_4.5.0       ellipsis_0.3.2       gower_1.0.2
#> [117] ggplot2_3.5.2         readr_2.1.5          assertthat_0.2.1     profvis_0.4.0
#> [121] urlchecker_1.0.1      listenv_0.9.1        ipred_0.9-15         scales_1.3.0
#> [125] prodlim_2024.06.25    purrr_1.0.4          rlang_1.1.6          rvest_1.0.4
```

Get All of the Team Stats Using `filter_team()`

`filter_team()` allows you to pull of the available stats for a team during a a given season.

```
filter_team("N.C. State", season = 2016)
#> # A tibble: 1 x 84
#> # Groups:   team [1]
#>   team    year   min   pos   fgm   fga fg_pct   tpm   tpa tp_pct   ftm   fta  oreb  dreb   reb
#>   <chr>  <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#> 1 N.C. ~  2016  201.  68.6  27.5  61.2  0.449  7.15  18.9  0.373  15.2  21.9  13.3  25.7  39.0
#> # i 69 more variables: ast <dbl>, stl <dbl>, blk <dbl>, to <dbl>, pf <dbl>, pts <dbl>,
#> #   opp_fgm <dbl>, opp_fga <dbl>, opp_fg_pct <dbl>, opp_tpm <dbl>, opp_tpa <dbl>,
#> #   opp_tp_pct <dbl>, opp_ftm <dbl>, opp_fta <dbl>, opp_ft_pct <dbl>, opp_oreb <dbl>,
#> #   opp_dreb <dbl>, opp_reb <dbl>, opp_ast <dbl>, opp_stl <dbl>, opp_blk <dbl>, opp_to <dbl>,
#> #   opp_pf <dbl>, opp_pts <dbl>, pts_scored <dbl>, pts_allowed <dbl>, adj_o.x <dbl>,
#> #   adj_d.x <dbl>, off_ppp <dbl>, off_efg <dbl>, off_to <dbl>, off_or <dbl>, off_ftr <dbl>,
#> #   def_ppp <dbl>, def_efg.x <dbl>, def_to <dbl>, def_or <dbl>, def_ftr.x <dbl>, ...
```

Notes

- If season is Null filter team will return all of the teams stats for all available seasons.

Plot the Power Ranking Using `power_plot()`

`power_plot()` allows you to visualize how a specific college basketball team's power ranking—measured by the barthag metric—has evolved over time. This plot is especially useful for identifying trends, such as periods of sustained success or decline, and comparing team trajectories across seasons. The only input to the function is the team name (as a string), and it returns a base R plot showing the team's power rating from 2013 to the most recent season available in the dataset.

```
plot_power("Utah St.")
```