College Basketball: A Sport Starving for Data

Carnegie Mellon Sports Analytics Conference, 2022

Abstract

Backed by a Fast API framework, `toRvik` provides best-in-class open-source men's college basketball data to a starving community at lightning speeds. Breaking beyond the box score, `toRvik` offers granular game-by-game player statistics, unprecedented access to detailed transfer and recruiting histories, and industry-leading prediction models – updated quarter-hourly and backlogged to the 2007-08 season.

1 A Data Ghost Town:

Accessible open-source sports data is a primary catalyst to innovation; analytics has reshaped consensus wisdom across major sports around the world. No longer does success start on the field or court: For many professional and amateur teams, winning begins with a spreadsheet.

American college basketball is no different. On average, major Division 1 athletic departments spend upwards of $2.5M yearly on the sport – and that is *before* any coaching salaries.[1] A portion of expenses typically go towards licensing popular analytics solutions.

College fans, however, are largely left in the dark. Consumer-focused analytics solutions are pricey and sparse: A monthly subscription to Synergy runs $30; CBBAnalytics nears $500 per year; and an access key to Genius Sports, the NCAA's official stat partner, pushes five figures. Sports Radar, another leader in college APIs, does not even offer a fan package. While consumer-targeted analytics sites do exist, many, including the ever-popular KenPom, have a pay-wall. Accessible open-source data is a luxury.

Despite attracting over 10,000,000 viewers per window in the national tournament last March, the open-source community remains starving for college basketball data.[2] Beyond television viewership, an estimated 15% of all U.S. adults filled out a March Madness bracket in 2022.[3] With viewership trending upwards and interest in sports analytics showing no signs of slowing, a massive data gap is forming.

2 toRvik:

Enter, `toRvik`: The only CRAN-hosted college basketball package. `toRvik` offers comprehensive statistics on every Division 1 player and game back to 2008, including:

- Granular game-by-game data with over 40 observations per player per game, splitable by numerous variables
- Adjusted and raw four factor data by game for every team, also splittable
- Extensive transfer and recruiting histories
- Detailed coaching and program trends
- Industry-leading prediction models for individual games and tournaments

`toRvik` ships with more than 20 functions, spanning player and team statistics to detailed program and coaching histories to NCAA tournament performance. Aggregating data from its namesake barttorvik.com, ESPN, 247Sports, and Verbal Commits, `toRvik` sources data from trusted sites to ensure accuracy. Functions are extensively documented at www.torvik.dev/.

[1] Caron, Emily. *College hoops spending gap prevails as women trail in travel and meals.*
[2] SportsMediaWatch. *Kansas-UNC big for cable, modest for National Championship.*
[3] Morning Consult. *Estimated Number of adults that intend to fill out a bracket during March Madness.*

2.0.1 Package Installation

At the time of writing, all code examples utilize the GitHub development version. A CRAN update is expected to be pushed by mid-September. The GitHub version, along with the CRAN, can be downloaded using:

```r
# the current CRAN version can be downloaded with:
install.packages("toRvik")

# the GitHub development version can be downloaded with:
if (!requireNamespace('devtools', quietly = TRUE)){
  install.packages('devtools')
}
devtools::install_github("andreweatherman/toRvik")
```

3 Fast Data Delivery:

Constructing models, digging for insights, and designing visualizations are time-intensive, and often time-sensitive, tasks. Waiting for functions to scrape data is a luxury that not many have; immediate analysis drives engagement.

`toRvik` is built atop a dedicated Fast API framework, aptly named `cbbstat`, written and deployed specifically for the package, providing dependable up-time, safety against harmful data restructuring, and lightning-quick data delivery speeds for end users. Anchored by on-instance data storage and backed by a high-performance MySQL server, `cbbstat` data is updated every fifteen minutes during peak game times (12 p.m. EST to 2 a.m. EST), ensuring that `toRvik` provides the most up-to-date basketball data possible.

3.0.1 All about the speed:

Benchmarks of select functions over 25 trials can be found below.

```r
library(microbenchmark)
library(knitr)

# benchmarking several toRvik functions
results <- summary(microbenchmark(
  Ratings = toRvik::bart_ratings(),
  Archive = toRvik::bart_archive(team='Duke'),
  Transfers = toRvik::transfer_portal(year=2022),
  Recruits = toRvik::player_recruiting_rankings(),
  `Player Splits` = toRvik::bart_player_splits(year=2022, split='result'),
  `Player Season` = toRvik::bart_player_season(year=2022, stat='box'),
  `Player Game` = toRvik::bart_player_game(year=2022, stat='box'),
  times = 1L,
  unit = 'seconds'
))

kable(results[, c('expr', 'median', 'min', 'max')], format='simple',
     caption='Median benchmark in seconds over 25 trials')
```

Table 1: Median benchmark in seconds over 25 trials

expr	median	min	max
Ratings	0.2114988	0.2114988	0.2114988
Archive	0.3051951	0.3051951	0.3051951
Transfers	0.2967748	0.2967748	0.2967748
Recruits	1.1226768	1.1226768	1.1226768
Player Splits	1.2809041	1.2809041	1.2809041
Player Season	1.0755454	1.0755454	1.0755454
Player Game	7.3690005	7.3690005	7.3690005

3.0.2 Cross-language compatibility:

While `toRvik` is exclusively available in R, leveraging a custom-built API liberates data availability. Using `api.cbbstat.com`, users can access large swaths of college basketball data in any programming language. Partial documentation, the API will soon expand, is available at the `/docs` endpoint.

4 Working in Harmony:

Despite aggregating data from several public sources, `toRvik` makes every effort to cross-link observations. This alleviates the frustration of fuzzy matching and joins on the end user; every player is associated with a unique player identification number, and every team is referred to by a single name – no more `gsub` calls to truncate *State* to *St.*. The cross-package identifiers let the data shine: Whether you are working with stats sourced from ESPN, the NCAA, or Verbal Commits, every function works in harmony and can be immediately connected to extensive player, game, and team statistics.

5 Beyond the Box Score:

Analytic fans understand that per-game box scores have a limited usage; advanced player and team statistics contextualize performance far more accurately. With `toRvik`, these advanced stats are provided at the individual level for every Division 1 game. These include usage rate, offensive and defensive ratings, box-stat rates, shot location splits, and dozens more. In total, over *70 million* combined player data points are available at a per-game level. Most functions shipped with `toRvik` do not pass a year parameter by default to the API, meaning that looping through individual years to collect all available data is not needed! Just a single line of code can access every data point supplied by the API.

These hyper-specific data points allow us to answer some ridiculous questions like: **Which senior had the most dunks over a three-game stretch in 2022?** Previously, approaching such a problem required access to play-by-play data, a luxury rarely afforded in college basketball, but with `toRvik`, you have immediate access to shooting splits for every Division 1 game back to 2008.

5.0.1 It's a dunk fest:

```r
library(data.table)
library(lubridate)
library(knitr)

# pull all shooting stats for seniors last season
data <- data.table(toRvik::bart_player_game(year=2022, stat='shooting', exp='Sr'))

#  create a column for tracking three-game windows by player
data <- data[, .(date=as_date(date), id, player, team, dunk_m, game_bin=ceiling(seq_len(.N) /3)), id]

# calculate rolling sums over each three-game window
data <- data[, dunk_sum := Reduce(`+`, dunk_m), by=.(id, game_bin)][, .(player, team, dunk_sum, date, game_bin)]

# Collapse the first and last date in each three-game window as a separate column to create a
# 'date range' for plotting purposes
data <- unique(data[, .(team, dunk_sum, date,
        date_parsed = glue::glue("{format(min(date), format='%b. %e')} - {format(max(date), format='%b. %e')}")),
        .(player, game_bin)][,-c('date', 'game_bin')][order(-dunk_sum)])

# set clear column names
setnames(data, c('Player', 'Team', 'Total Dunks', 'Game Range'))

kable(data[(1:5)],
      format = 'simple',
      caption='Most dunks by seniors over a three-game stretch in 2022')
```

Table 2: Most dunks by seniors over a three-game stretch in 2022

Player	Team	Total Dunks	Game Range
Kevin Samuel	Florida Gulf Coast	13	Dec. 11 - Dec. 22
Osun Osunniyi	St. Bonaventure	13	Feb. 14 - Feb. 19
Kris Bankston	Norfolk St.	11	Feb. 26 - Mar. 3
Kris Bankston	Norfolk St.	11	Mar. 9 - Mar. 12
Osun Osunniyi	St. Bonaventure	11	Jan. 14 - Jan. 21

6 Splits, Splits, and More Splits:

6.0.1 Player splits:

Not only does `toRvik` offer extensive statistical coverage, users can query season-long player data by detailed splits. As of this writing, all statistics can be split by result (wins vs. losses), game month, and game type (non-conference, in-conference, post-season). This functionality can answer detailed questions like: **Which player had the largest average scoring difference between wins and losses last season?**

```r
library(data.table)
library(knitr)

# grab per-game box averages by result split ('W' vs. 'L')
data <- data.table(toRvik::bart_player_splits(split='result', year=2022, type='box'))

# filter for players who average at least 20 mpg
data <- data[ min >= 20, .(player, id, team, result, min, pts)]

# exclude players who only average 20+ minutes in either wins or losses but not in both splits
data <- data[data[, .I[.N==2], by=.(id)]$V1][order(id, result)]

# grouping by player, take the lead of the pts column -- which will always be points scored in wins b/c the data is
data <- data[, in_wins := shift(pts, 1, type='lead'), .(player, id, team)]

# calculate the difference in points scored in wins and losses
data <- data[, difference := (in_wins - pts), .(player, id, team)][order(-difference)][(1:5),-c('result', 'min', 'i

# set clear column names
setnames(data, c('Player', 'Team', 'Pts. in Losses', 'Pts. in Wins', 'Difference'))

kable(data, format = 'simple',
      caption='Which players show out in wins?')
```

Table 3: Which players show out in wins?

Player	Team	Pts. in Losses	Pts. in Wins	Difference
Jose Placer	North Florida	12.44	26.50	14.06
Ithiel Horton	Pittsburgh	6.60	20.33	13.73
Mitchell Sueker	North Dakota	8.08	21.50	13.42
Darius Lee	Houston Baptist	14.82	27.29	12.47
Joe Quintana	Loyola Marymount	8.17	20.60	12.43

6.0.2 Team splits:

Splittable stats is not a feature exclusive to player data. `toRvik` offers detailed team splits along several variables, including game result, game type, date range (start and/or end), location, and last `n` games. These splits summarize two-sided four factor data, adjusted efficiency, and tempo while aggregating total wins and losses – allowing us to answer questions like: **Which ACC team gets to the line at the highest rate during home conference games?**

```r
library(data.table)
library(glue)
library(knitr)

# grab data
data <- data.table(toRvik::bart_factors(location='H', type='conf'))

# filter for ACC teams and create record column
data <- data[conf=='ACC']
data[,`:=` (record = paste(wins, losses, sep='-'))]

# select columns to plot
data <- data[, .(team, record, off_ftr)][order(-off_ftr)]

# set clear column names
setnames(data, c('Team', 'Record', 'Free Throw Rate'))

# plot
kable(data[(1:5)], format='simple',
      caption = 'Which ACC team gets to the line most in home conference
      games?')
```

Table 4: Which ACC team gets to the line most in home conference games?

Team	Record	Free Throw Rate
Wake Forest	8-2	39.59
Pittsburgh	4-6	38.64
Syracuse	6-4	34.74
Notre Dame	9-1	33.40
Florida St.	6-4	32.86

7 Transfers and Recruits:

`toRvik` offers unprecedented access to transfer and recruiting data for men's college basketball. Previously, no service has made it possible to quickly and efficiently match player statistics to transfer decisions or compare college performance to high school rankings. Leveraging Verbal Commits' vast transfer database, `toRvik` offers to-the-minute transfer portal updates, including for players coming from or going to a non-Division 1 school. Utilizing fuzzy match algorithms, `toRvik` has associated over 5,600 transfer decisions to unique player IDs back to 2012 – allowing users to pull granular player statistics to better contextualize why players might decide to leave a school.

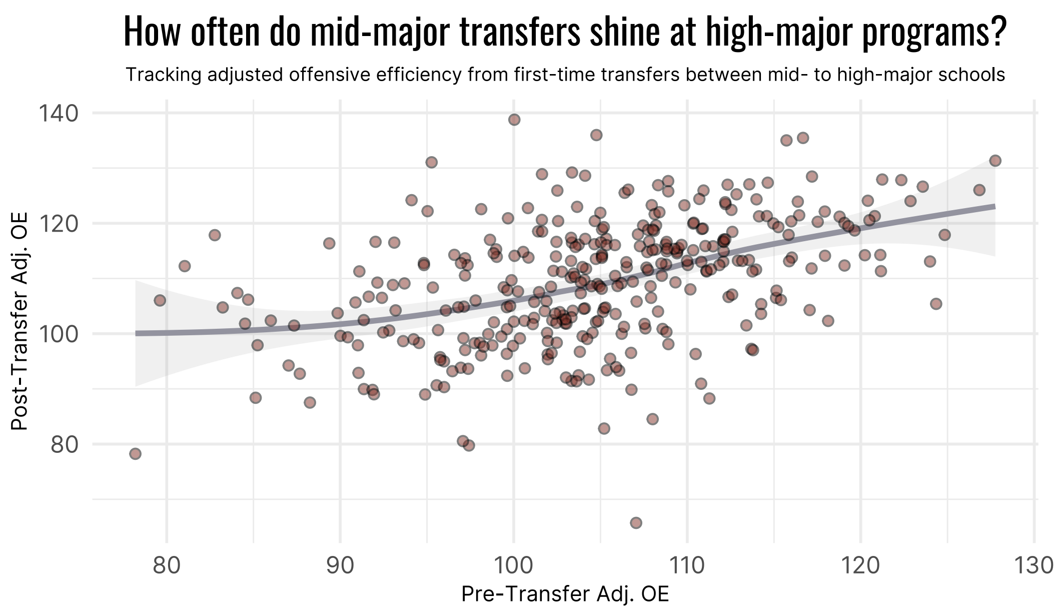
On the recruiting side, `toRvik` pulls from 247Sports' API to source recruiting data for over 6,300 high school prospects back to 2008. This data set aggregates several leading recruiting services, including 247Sports, ESPN, and Rivals, and returns player meta data, prospect locations, positional, state, and national rankings, and a unique player ID – allowing users to compare college performance to various recruiting rankings. This data not only allows users to track in-college performance but also opens the door for exciting network visualizations by leveraging player high school and hometown meta data.

The existence of this data presents exciting research and analytic opportunities that were previously shuttered behind tedious web scraping projects and complex matching algorithms. `toRvik` makes this possible by aggregating public data while matching observations to a package-wide player identification system, coalescing extensive statistical histories with to-the-minute transfer and recruiting updates. With just a few lines of code, users can associate advanced tempo-free player statistics to recruits or transfers.

7.0.1 Example of plotting transfer data:

Sourcing transfer data is a challenging ask in men's basketball. Often, announcements are buried in player tweets or program releases. Luckily, Verbal Commits, a leader in tracking player movement, has a public-facing collection of transfer decisions back to 2012. Coupled with `toRvik`'s extensive player statistic database, we can answer questions like: **Do mid-major transfers see a dip in efficiency after joining a high-major program?**

```r
library(data.table)
library(showtext)
library(ggplot2)
# qdapRegex is only needed to render code chunk wrapping, not to run the script
library(qdapRegex)

# get all transfer data for mid-to-high major transfers
hm <- c('ACC', 'B10', 'B12', 'P10', 'P12', 'SEC')
high_majors <- data.table(toRvik::teams_by_year())[conf %in% hm, team]
transfers <- data.table(toRvik::transfer_portal())[to %in% high_majors & !from %in% high_majors]

# filter for players who transfered only once -- they will appear just once in the data set
transfers <- transfers[transfers[, .I[.N==1], by=.(id)]$V1]

# pull advanced statistics for each player
stats <- data.table(toRvik::bart_player_season(stat='advanced'))

# filter for players who appeared in at least 15 games and played in at least 30% of team minutes
stats <- stats[id %in% transfers[, id] & g >= 15 & min >= 30, .(id, year, team, g, adj_oe, min)]

# calculate pre- and post-transfer averages
averages <- stats[, .(avg_oe = mean(adj_oe)), .(id, team)]

# prepare data to be transformed to wide format by creating a 'pre' and 'post' column that will be split on
averages <- averages[, .(avg_oe, name = fifelse(seq_len(.N) == 1, 'pre', 'post'), num = seq_len(.N)), .(id)]

# cast the data to wide format to plot values
averages <- dcast(averages[num<=2], id ~ name, value.var = 'avg_oe')

# plot the data
font_add_google(name = 'Oswald', family='oswald')
font_add_google(name = 'Inter', family='inter')
showtext_auto()
averages[complete.cases(averages)] |>
  ggplot(aes(pre, post)) +
    geom_smooth(alpha=0.15, color='#93939F') +
    geom_point(shape=21, fill='#823329', alpha=0.5) +
    theme_minimal() +
    theme(plot.title = element_text(family='oswald', hjust=0.5, size=14),
        plot.subtitle = element_text(family='inter', hjust = 0.5, size=7),
        plot.caption = element_text(family='inter', hjust=0, size=6, lineheight = 1.2),
        plot.caption.position = 'plot',
        axis.text = element_text(family='inter'),
        axis.title = element_text(family='inter', size=8)) +
    labs(title = 'How often do mid-major transfers shine at high-major programs?',
       subtitle = rm_white('Tracking adjusted offensive efficiency from first-time transfers between
                              mid- to high-major schools'),
       caption = rm_white('Adjusted offensive efficiency represents estimated points produced per 100
                          possessions scaled to opponent strength. Data points are \nrestricted to players
                          with at least 15 games and 30% of team minutes played in a season.'),
        x = 'Pre-Transfer Adj. OE',
        y = 'Post-Transfer Adj. OE')
```

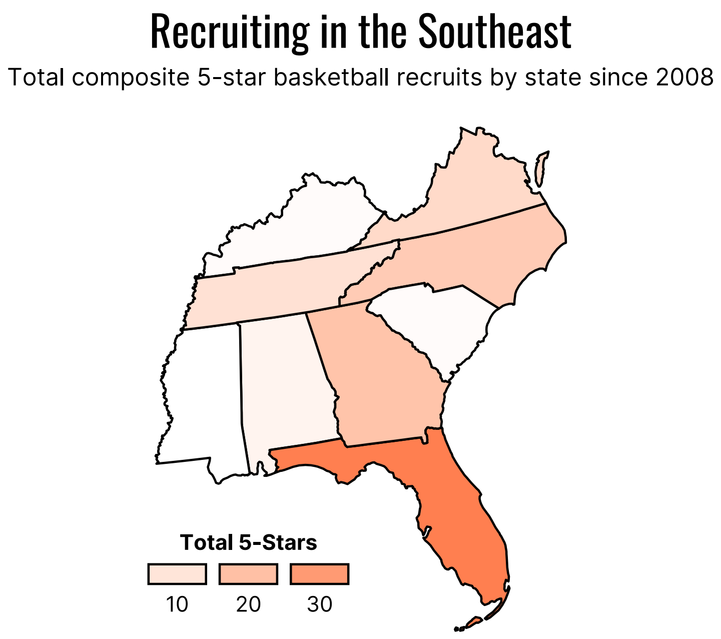


Adjusted offensive efficiency represents estimated points produced per 100 possessions scaled to opponent strength. Data points are restricted to players with at least 15 games and 30% of team minutes played in a season.

7.0.2 Example of mapping recruiting data:

In addition to state, positional, and national rankings across three major services, the recruiting data set also returns prospect high school and hometown. With this information, we can answer questions like: **Which southern state produces the most basketball talent?**

```r
library(data.table)
library(usmap)
library(ggplot2)
library(showtext)

# grab all 5-star recruits
players <- data.table(toRvik::player_recruiting_rankings(stars=5))

# aggregate player count by state
count <- players[,.(total=.N), state]

southeast <- c('VA', 'NC', 'SC', 'GA', 'FL', 'TN', 'AL', 'MS', 'KY')
font_add_google(name = 'Oswald', family='oswald')
font_add_google(name = 'Inter', family='inter')
showtext_auto()
plot_usmap(include=southeast,
           data = count,
           values='total') +
  scale_fill_continuous(
    low = 'white',
    high = 'coral',
    name = 'Total 5-Stars',
    guide = guide_legend(
      keyheight = unit(3, units='mm'),
      keywidth = unit(8, units='mm'),
      label.position = 'bottom',
      title.position = 'top',
      title.hjust = 0.5
    )
  ) +
  theme(
    plot.title = element_text(family='oswald', size = 16, hjust=0.5),
    plot.subtitle = element_text(family='inter', hjust = 0.5, size=9),
    legend.position = c(0, 0.05),
    legend.direction = 'horizontal',
    legend.title = element_text(family='inter', face='bold', size=8),
    legend.text = element_text(family='inter', size=8)
  ) +
  labs(
    title = 'Recruiting in the Southeast',
    subtitle = 'Total composite 5-star basketball recruits by state since 2008'
  )
```


Recruiting in the Southeast
Total composite 5-star basketball recruits by state since 2008
Total 5-Stars
10
20
30

8 Taking on Vegas and Your Bracket Pool

The popular college basketball analytics site barttorvik.com, the namesake of `toRvik`, offers an industry-leading predictive game model named *T-Rank*. It frequently finishes atop consensus analytic rankings for predictive accuracy. Last season, Barttorvik's *T-Rank* model finished eighth in a sample of 68 total models.[4] Working with the founder of barttorvik.com and using the same data fed to his models, `toRvik` brings a powerful predictive engine to the open-source community.

8.0.1 Game Predictions

`bart_game_prediction` returns the expected score and win percentage, projected points per possession, and pace of play for two teams on any given day at any given venue (home, away, or neutral). While barttorvik.com returns expected score and win percentage for set games, there is no way to customize dates and locations or test unscheduled games. Want to know how your favorite team would stack up against Duke in Cameron Indoor Stadium in mid-January? Previously, you would have to go scrape the data and run the calculations yourself. But with `toRvik`, game predictions are just a line of code away.

```
library(knitr)
```

[4] Wobus Sports. http://sports.vaporia.com/bb-six.html

```r
data <- toRvik::bart_game_prediction(team='Charlotte',
                                     opp='Duke',
                                     date='20220113',
                                     location='A')
kable(data, format = 'simple')
```

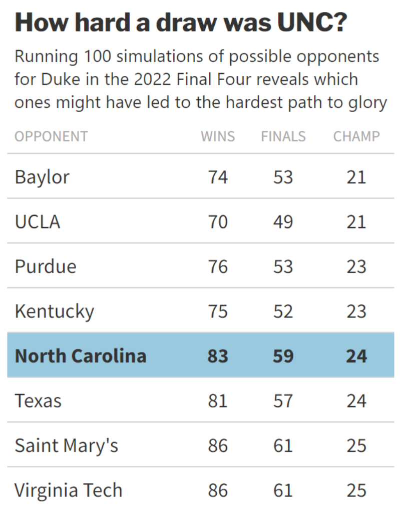
team	date	location	tempo	ppp	pts	win_per	did_win
Duke	Jan. 13, 2022	Home	69.29584	1.2512874	86.7	93.157420	TRUE
Charlotte	Jan. 13, 2022	Away	69.29584	0.9648186	66.9	6.842581	FALSE

8.0.2 Tournament Predictions

Extending the core functionality of the game prediction model leads to `bart_tournament_prediction`. Taking in a list of teams as an argument, the function simulates a single-elimination tournament `n` times, logging the number of wins, finals appearances, and championships won by each team. `toRvik` provides a free, results-driven model to be leveraged in competitive bracket pools. Using the tournament predictor function, **we can assess how Duke might have fared in the Final Four given every possible East region opponent.**

As the season approaches, `bart_tournament_prediction` will be updated to include support for MTEs (multiple-team events; in-season, non-conference tournaments) and overhauled table returns for full NCAA and conference tournament simulations.

```r
library(data.table)
library(purrr)
library(gt)
library(gtExtras)

# list all possible East region opponents
possible_teams <- c('Baylor', 'Norfolk St.', 'North Carolina', 'Marquette', "Saint Mary's", 'Indiana', 'UCLA',
                    'Akron', 'Texas', 'Virginia Tech', 'Purdue', 'Yale', 'Murray St.', 'San Francisco', 'Kentucky',

# loop through each opponent
results <- data.table(map_dfr(.x = possible_teams,
            .f = function(opp) {
            teams <- c('Duke', opp, 'Kansas', 'Villanova')
            results <- toRvik::bart_tournament_predicition(
              teams = teams,
              date = '20220402',
              sims = 100,
              seed = 20
            ) |>
              dplyr::mutate(opponent=opp) |>
              dplyr::filter(team=='Duke')
        }))

# set names
results <- setnames(results[,c(5, 2:4)], c('Opponent', 'Wins', 'Finals', 'Champ'))

# plot results
results[order(Champ)][(1:8)] |>
  gt() |>
  gt_theme_nytimes() |>
  cols_align(align='center', columns = c('Wins', 'Finals', 'Champ')) |>
  gt_highlight_rows(rows=5) |>
  tab_header(title="How hard a draw was UNC?",
             subtitle=html('Running 100 simulations of possible
                           opponents<br>for Duke in the 2022 Final Four reveals which<br>ones might have
                           led to the hardest path to glory'))
```



How hard a draw was UNC?

Running 100 simulations of possible opponents for Duke in the 2022 Final Four reveals which ones might have led to the hardest path to glory

OPPONENT	WINS	FINALS	CHAMP
Baylor	74	53	21
UCLA	70	49	21
Purdue	76	53	23
Kentucky	75	52	23
North Carolina	**83**	**59**	**24**
Texas	81	57	24
Saint Mary's	86	61	25
Virginia Tech	86	61	25

9 Conclusion and the Future

`toRvik` offers seamless access to the most extensive repository of men's college basketball data publicly available at rapid speeds. Previously, sourcing such data would have required long scraping scripts, complex matching algorithms, and a load of patience. `toRvik` seeks to establish parity and equity in the data community, extending much-needed open-source access to one of America's most-followed sports.

As the college basketball season approaches, development will focus on further building out the API that grounds much of `toRvik`'s functionality. Users can expect more data coverage and subsequently less retrieval times for many of the non-API functions in the package. Calculating more detailed splits by player and team statistics will also be a focus of improvement.