**Prologue**

Package [comperank](https://github.com/echasnovski/comperank) is [on CRAN](https://cran.r-project.org/package=comperank) now. It offers consistent implementations of several ranking and rating methods. Originally, it was intended to be my first CRAN package when I started to build it 13 months ago. Back then I was very curious to learn about different ranking and rating methods that are used in sport. This led me to two conclusions:

* There is an amazing book [“Who’s #1”](https://www.amazon.com/Whos-1-Science-Rating-Ranking/dp/069116231X) by Langville and Meyer which describes several ideas in great detail.
* Although there are some CRAN packages dedicated specifically to ranking methods (for example, [elo](https://cran.r-project.org/package=elo), [mvglmmRank](https://cran.r-project.org/package=mvglmmRank)), I didn’t find them to be [tidy](https://cran.r-project.org/web/packages/tidyverse/vignettes/manifesto.html) enough.

These discoveries motivated me to write my first ever CRAN package. Things didn’t turn out the way I was planning, and now comperank is actually my fourth. After spending some time writing it I realized that most of the package will be about storing and manipulating competition results in consistent ways. That is how [comperes](https://echasnovski.github.io/comperes/) was born.

After diverging into creating this site and writing [ruler](https://echasnovski.github.io/ruler/) in pair with [keyholder](https://echasnovski.github.io/keyholder/), a few months ago I returned to competition results and rankings. Gained experience helped me to improve functional API of both packages which eventually resulted into submitting them to CRAN.

**Overview**

This post, as [one of the previous ones](http://www.questionflow.org/2018/05/09/harry-potter-and-competition-results-with-comperes/), has two goals:

* Explore different types of rankings on [Harry Potter Books Survey](http://www.questionflow.org/2018/04/09/struggle-with-harry-potter-data/) results (data provided by comperes).
* Demonstrate basic functionality of comperank package. To learn more go to its [README](https://echasnovski.github.io/comperank/index.html), [vignettes](https://echasnovski.github.io/comperank/articles/) and [manual pages](https://echasnovski.github.io/comperank/reference/index.html).

We will cover the following topics:

* Short notes about **functionality of comperank**.
* **Exploration ranking** with ranking based on mean book score. No comperank package functionality is required.
* **Rankings with fixed Head-to-Head structure**. This will cover Massey and Colley ranking methods.
* **Rankings with variable Head-to-Head structure**. This will cover Keener, Markov and Offense-Defense ranking methods.
* **Combined rankings** in which average ranks will be computed using all described comperank methods.

Another very interesting set of ranking methods implemented in comperank are methods with iterative nature. However, their usage with mentioned [Harry Potter Books Survey dataset](https://echasnovski.github.io/comperes/reference/hp_survey.html) is meaningless as temporal ordering of games (acts of book scoring by one person) should make sense, which it doesn’t.

The idea behind converting survey results into competition results is described in aforementioned post. We will need the following setup:

library(dplyr)

library(purrr)

library(rlang)

# This will automatically load {comperes}

library(comperank)

# Create competition results from hp\_survey

hp\_cr <- hp\_survey %>%

transmute(

game = person, player = book,

score = as.integer(gsub("[^0-9].\*$", "", score))

) %>%

as\_longcr()

**Functionality of comperank**

**Rating** is considered to be a list (in the ordinary sense) of numerical values, one for each player, or the numerical value itself. Its interpretation depends on rating method: either bigger value indicates better player performance or otherwise.

**Ranking** is considered to be a rank-ordered list (in the ordinary sense) of players: rank 1 indicates player with best performance.

comperank leverages the [tidyverse](https://www.tidyverse.org/) ecosystem of R packages. Among other things, it means that the main output format is [tibble](http://tibble.tidyverse.org/).

There are three sets of functions:

* rate\_\*() (\* stands for ranking method short name). Its output is a tibble with columns player (player identifier) and at least one rating\_\* (rating value). Names of rating columns depend on rating method.
* rank\_\*(). Its default output is similar to previous one, but with ranking\_\* instead of rating columns. It runs rate\_\*() and does ranking with correct direction. One can use option keep\_rating = TRUE to keep rating columns in the output.
* add\_\*\_ratings(). These functions are present only for algorithms with iterative nature and competition results with games only between two players. They return tibble with row corresponding to a game and extra columns indicating ratings of players before and after the game.

**Exploration ranking**

Previously we established that “Harry Potter and the Prisoner of Azkaban” seems to be “the best” book and “Harry Potter and the Chamber of Secrets” comes last. This was evaluated by mean score:

hp\_rank\_explore <- hp\_cr %>%

summarise\_player(rating\_explore = mean(score)) %>%

# round\_rank() is a function from {comperank} package for doing ranking

mutate(ranking\_explore = round\_rank(rating\_explore))

hp\_rank\_explore

## # A tibble: 7 x 3

## player rating\_explore ranking\_explore

##

## 1 HP\_1 3.91 5

## 2 HP\_2 3.55 7

## 3 HP\_3 4.19 1

## 4 HP\_4 4 3

## 5 HP\_5 3.90 6

## 6 HP\_6 4.13 2

## 7 HP\_7 3.96 4

As simple as it is, this approach might leave some available information unused. Survey originally was designed to obtain information not only about books performance as separate objects, but also to learn about possible pair relationships between them. Maybe some book is considered generally “not the best” but it “outperforms” some other “better” book. This was partially studied in “Harry Potter and competition results with comperes” by computing different Head-to-Head values and manually studying them.

Here we will attempt to summarise books performance based on their Head-to-Head relationships.

**Rankings with fixed H2H structure**

In comperank there are two methods which operate on fixed Head-to-Head structure: **Massey** and **Colley**. Both of them are designed for competitions where:

* Games are held only between two players.
* It is assumed that score is numeric and higher values indicate better player performance in a game.

Being very upset for moment, we realize that in dataset under study there are games with different number of players. Fortunately, comperes package comes to rescue: it has function [to\_pairgames()](https://echasnovski.github.io/comperes/reference/pairgames.html) just for this situation. It takes competition results as input and returns completely another (strictly speaking) competition results where “crowded” games are split into small ones. More strictly, games with one player are removed and games with three and more players are converted to multiple games between all unordered pairs of players. The result is in [wide](https://echasnovski.github.io/comperes/reference/widecr.html) format (as opposed to [long](https://echasnovski.github.io/comperes/reference/longcr.html) one of hp\_cr):

hp\_cr\_paired <- to\_pairgames(hp\_cr)

# For example, second game was converted to a set of 10 games

hp\_cr %>% filter(game == 2)

## # A longcr object:

## # A tibble: 5 x 3

## game player score

##

## 1 2 HP\_1 3

## 2 2 HP\_4 5

## 3 2 HP\_5 2

## 4 2 HP\_6 4

## 5 2 HP\_7 5

hp\_cr\_paired %>% slice(2:11)

## # A widecr object:

## # A tibble: 10 x 5

## game player1 score1 player2 score2

##

## 1 2 HP\_1 3 HP\_4 5

## 2 3 HP\_1 3 HP\_5 2

## 3 4 HP\_1 3 HP\_6 4

## 4 5 HP\_1 3 HP\_7 5

## 5 6 HP\_4 5 HP\_5 2

## 6 7 HP\_4 5 HP\_6 4

## 7 8 HP\_4 5 HP\_7 5

## 8 9 HP\_5 2 HP\_6 4

## 9 10 HP\_5 2 HP\_7 5

## 10 11 HP\_6 4 HP\_7 5

**Massey method**

Idea of Massey method is that difference in ratings should be proportional to score difference in direct confrontations. Bigger value indicates better player competition performance.

hp\_cr\_massey <- hp\_cr\_paired %>% rank\_massey(keep\_rating = TRUE)

hp\_cr\_massey

## # A tibble: 7 x 3

## player rating\_massey ranking\_massey

##

## 1 HP\_1 -0.00870 5

## 2 HP\_2 -0.514 7

## 3 HP\_3 0.293 1

## 4 HP\_4 0.114 3

## 5 HP\_5 0.00195 4

## 6 HP\_6 0.124 2

## 7 HP\_7 -0.00948 6

**Colley method**

Idea of Colley method is that ratings should be proportional to share of player’s won games. Bigger value indicates better player performance.

hp\_cr\_colley <- hp\_cr\_paired %>% rank\_colley(keep\_rating = TRUE)

hp\_cr\_colley

## # A tibble: 7 x 3

## player rating\_colley ranking\_colley

##

## 1 HP\_1 0.497 5

## 2 HP\_2 0.326 7

## 3 HP\_3 0.599 1

## 4 HP\_4 0.534 3

## 5 HP\_5 0.505 4

## 6 HP\_6 0.542 2

## 7 HP\_7 0.497 6

Both Massey and Colley give the same result differing from Exploration ranking in treating “HP\_5” (“Order of the Phoenix”) and “HP\_7” (“Deathly Hallows”) differently: “HP\_5” moved up from 6-th to 4-th place.

**Rankings with variable H2H structure**

All algorithms with variable Head-to-Head structure depend on user supplying custom Head-to-Head expression for computing quality of direct confrontations between all pairs of players of interest.

There is much freedom in choosing Head-to-Head structure appropriate for ranking. For example, it can be “number of wins plus half the number of ties” (implemented in h2h\_funs[["num\_wins2"]] from comperes) or “mean score difference from direct matchups” (h2h\_funs[["mean\_score\_diff"]]). In this post we will use the latter one. Corresponding Head-to-Head matrix looks like this:

hp\_h2h <- hp\_cr %>%

h2h\_mat(!!! h2h\_funs[["mean\_score\_diff"]]) %>%

round(digits = 2)

# Value indicates mean score difference between "row-player" and

# "column-player". Positive - "row-player" is better.

hp\_h2h

## # A matrix format of Head-to-Head values:

## HP\_1 HP\_2 HP\_3 HP\_4 HP\_5 HP\_6 HP\_7

## HP\_1 0.00 0.50 -0.39 0.04 0.00 -0.14 -0.06

## HP\_2 -0.50 0.00 -0.77 -0.58 -0.72 -0.62 -0.45

## HP\_3 0.39 0.77 0.00 0.05 0.51 0.11 0.25

## HP\_4 -0.04 0.58 -0.05 0.00 -0.04 0.09 0.20

## HP\_5 0.00 0.72 -0.51 0.04 0.00 -0.17 -0.04

## HP\_6 0.14 0.62 -0.11 -0.09 0.17 0.00 0.15

## HP\_7 0.06 0.45 -0.25 -0.20 0.04 -0.15 0.00

**Keener method**

Keener method is based on the idea of “relative strength” – the strength of the player relative to the strength of the players he/she has played against. This is computed based on provided Head-to-Head values and some flexible algorithmic adjustments to make method more robust. Bigger value indicates better player performance.

hp\_cr\_keener <- hp\_cr %>%

rank\_keener(!!! h2h\_funs["mean\_score\_diff"], keep\_rating = TRUE)

hp\_cr\_keener

## # A tibble: 7 x 3

## player rating\_keener ranking\_keener

##

## 1 HP\_1 0.147 5

## 2 HP\_2 0.0816 7

## 3 HP\_3 0.191 1

## 4 HP\_4 0.150 4

## 5 HP\_5 0.153 3

## 6 HP\_6 0.155 2

## 7 HP\_7 0.122 6

Results for Keener method again raised “HP\_5” one step up to third place.

**Markov method**

The main idea of Markov method is that players “vote” for other players’ performance. Voting is done with Head-to-Head values and the more value the more “votes” gives player2 (“column-player”) to player1 (“row-player”). For example, if Head-to-Head value is “number of wins” then player2 “votes” for player1 proportionally to number of times player1 won in a matchup with player2.

Actual “voting” is done in [Markov chain](https://en.wikipedia.org/wiki/Markov_chain) fashion: Head-to-Head values are organized in stochastic matrix which vector of stationary probabilities is declared to be output ratings. Bigger value indicates better player performance.

hp\_cr\_markov <- hp\_cr %>%

rank\_markov(!!! h2h\_funs["mean\_score\_diff"], keep\_rating = TRUE)

hp\_cr\_markov

## # A tibble: 7 x 3

## player rating\_markov ranking\_markov

##

## 1 HP\_1 0.140 5

## 2 HP\_2 0.0500 7

## 3 HP\_3 0.196 1

## 4 HP\_4 0.168 2

## 5 HP\_5 0.135 6

## 6 HP\_6 0.167 3

## 7 HP\_7 0.143 4

We can see that Markov method put “HP\_4” (“Goblet of Fire”) on second place. This is due to its reasonably good performance against the leader “HP\_3” (“Prisoner of Azkaban”): mean score difference is only 0.05 in “HP\_3” favour. Doing well against the leader in Markov method has a great impact on output ranking, which somewhat resonates with common sense.

**Offense-Defense method**

The idea of Offense-Defense (OD) method is to account for different abilities of players by combining different ratings:

* For player which can achieve *high* Head-to-Head value (even against the player with strong defense) it is said that he/she has **strong offense** which results into *high* offensive rating.
* For player which can force their opponents into achieving *low* Head-to-Head value (even if they have strong offense) it is said that he/she has **strong defense** which results into *low* defensive rating.

Offensive and defensive ratings describe different skills of players. In order to fully rate players, OD ratings are computed: offensive ratings divided by defensive. The more OD rating the better player performance.

hp\_cr\_od <- hp\_cr %>%

rank\_od(!!! h2h\_funs["mean\_score\_diff"], keep\_rating = TRUE)

print(hp\_cr\_od, width = Inf)

## # A tibble: 7 x 7

## player rating\_off rating\_def rating\_od ranking\_off ranking\_def

##

## 1 HP\_1 5.42 1.03 5.29 5 5

## 2 HP\_2 1.45 1.88 0.771 7 7

## 3 HP\_3 7.91 0.522 15.1 1 1

## 4 HP\_4 6.51 0.869 7.49 3 3

## 5 HP\_5 5.30 0.888 5.97 6 4

## 6 HP\_6 6.59 0.809 8.14 2 2

## 7 HP\_7 5.54 1.05 5.29 4 6

## ranking\_od

##

## 1 5

## 2 7

## 3 1

## 4 3

## 5 4

## 6 2

## 7 6

All methods give almost equal results again differing only in ranks of “HP\_5” and “HP\_7”.

**Combined rankings**

To obtain averaged, and hopefully less “noisy”, rankings we will combine rankings produced with comperank by computing their mean.

list(hp\_cr\_massey, hp\_cr\_colley, hp\_cr\_keener, hp\_cr\_markov, hp\_cr\_od) %>%

# Extract ranking column

map(. %>% select(., player, starts\_with("ranking"))) %>%

# Join all ranking data in one tibble

reduce(left\_join, by = "player") %>%

# Compute mean ranking

transmute(player, ranking\_combined = rowMeans(select(., -player))) %>%

# Join exploration rankings for easy comparison

left\_join(y = hp\_rank\_explore %>% select(-rating\_explore), by = "player")

## # A tibble: 7 x 3

## player ranking\_combined ranking\_explore

##

## 1 HP\_1 5 5

## 2 HP\_2 7 7

## 3 HP\_3 1 1

## 4 HP\_4 3 3

## 5 HP\_5 4.43 6

## 6 HP\_6 2.14 2

## 7 HP\_7 5.43 4

As we can see, although different ranking methods handle results differently for books with “middle performance”, combined rankings are only slightly different from exploration ones. Only notable difference is in switched rankings of “Order of the Phoenix” and “Deathly Hallows”.

**Conclusion**

* “Harry Potter and the Prisoner of Azkaban” still seems to be considered “best” among R users. And yet “Harry Potter and the Chamber of Secrets” still suffers the opposite fate.
* Using different ranking methods is a powerful tool in analyzing Head-to-Head performance. This can be done in very straightforward manner with new addition to CRAN – [comperank](https://github.com/echasnovski/comperank) package.

[sessionInfo()](http://www.questionflow.org/2018/05/31/harry-potter-and-rankings-with-comperank/#sessionInfo)

sessionInfo()

## R version 3.4.4 (2018-03-15)

## Platform: x86\_64-pc-linux-gnu (64-bit)

## Running under: Ubuntu 16.04.4 LTS

##

## Matrix products: default

## BLAS: /usr/lib/openblas-base/libblas.so.3

## LAPACK: /usr/lib/libopenblasp-r0.2.18.so

##

## locale:

## [1] LC\_CTYPE=ru\_UA.UTF-8 LC\_NUMERIC=C

## [3] LC\_TIME=ru\_UA.UTF-8 LC\_COLLATE=ru\_UA.UTF-8

## [5] LC\_MONETARY=ru\_UA.UTF-8 LC\_MESSAGES=ru\_UA.UTF-8

## [7] LC\_PAPER=ru\_UA.UTF-8 LC\_NAME=C

## [9] LC\_ADDRESS=C LC\_TELEPHONE=C

## [11] LC\_MEASUREMENT=ru\_UA.UTF-8 LC\_IDENTIFICATION=C

##

## attached base packages:

## [1] methods stats graphics grDevices utils datasets base

##

## other attached packages:

## [1] bindrcpp\_0.2.2 comperank\_0.1.0 comperes\_0.2.0 rlang\_0.2.0

## [5] purrr\_0.2.4 dplyr\_0.7.5

##

## loaded via a namespace (and not attached):

## [1] Rcpp\_0.12.17 knitr\_1.20 bindr\_0.1.1 magrittr\_1.5

## [5] tidyselect\_0.2.4 R6\_2.2.2 stringr\_1.3.1 tools\_3.4.4

## [9] xfun\_0.1 utf8\_1.1.3 cli\_1.0.0 htmltools\_0.3.6

## [13] yaml\_2.1.19 rprojroot\_1.3-2 digest\_0.6.15 assertthat\_0.2.0

## [17] tibble\_1.4.2 crayon\_1.3.4 bookdown\_0.7 tidyr\_0.8.1

## [21] glue\_1.2.0 evaluate\_0.10.1 rmarkdown\_1.9 blogdown\_0.6

## [25] stringi\_1.2.2 compiler\_3.4.4 pillar\_1.2.2 backports\_1.1.2

## [29] pkgconfig\_2.0.1