**Prologue**

About a month ago I decided to add interesting data set to my almost finished (hopefully, soon to be on CRAN) [comperes](https://github.com/echasnovski/comperes) package. Data should represent results of some not ordinary competition. After some thought I picked a “competition” between Harry Potter books with a goal eventually to rate them from worst to best. After a series of events I ended up creating data myself. You can read more about that in my [previous post](http://www.questionflow.org/2018/04/09/struggle-with-harry-potter-data/).

Post and survey in general were popularized mostly among R users with [R-bloggers](https://www.r-bloggers.com/) (which gave me ~53 respondents), [Twitter](https://twitter.com/) (which added the rest) and [Reddit](https://www.reddit.com/r/rstats/) (which added ~0 people as post was deleted soon after publication). Survey managed to attract 182 respondents. I want to greatly thank all people who took their time to take part in and spread a word about my survey. Special thanks goes to [Mara Averick](https://twitter.com/dataandme) who started a Twitter wave.

This post has two goals:

* Present and explore results of the survey.
* Demonstrate basic functionality of comperes package. To learn more go to its [README](https://github.com/echasnovski/comperes#comperes) and vignettes.

**Overview**

Survey results can be obtained by installing development version of comperes package from GitHub. They are present as package data named hp\_survey.

This post will cover the following topics:

* **Exploration** of survey results (most important being **Book scores** section).
* Description of comperes **competition results** formats with conversion hp\_survey to one of them.
* **Head-to-Head** “performance” of books against each other.

We will need the following setup:

library(dplyr)

library(tidyr)

library(rlang)

library(stringr)

library(ggplot2)

library(comperes)

set.seed(201805)

theme\_set(theme\_bw())

# Authenticity palette

hp\_pal <- c(Gryff = "#D02037", Huffl = "#F0C346",

Raven = "#2450A8", Raven\_light = "#0088FF",

Slyth = "#09774A")

# For less noisy bar charts

theme\_bar <- function() {

list(theme(panel.grid.major.x = element\_blank(),

panel.grid.minor.x = element\_blank()))

}

**Exploration**

**Data preparation**

hp\_suvery is a [tibble](http://tibble.tidyverse.org/) (enhanced data frame) and has the following columns:

* **person** : Identifier of a person.
* **book** : Identifier of a Harry Potter book. Its values are of the form “HP\_x” where “x” represents book’s number in the series (from 1 to 7).
* **score** : Book’s score. Can be one of “1 – Poor”, “2 – Fair”, “3 – Good”, “4 – Very Good”, “5 – Excellent”.

*For exploration*, let’s transform hp\_survey for more expressive code and results:

* Convert scores to numerical.
* Add book names.

book\_names <- c(

"Philosopher's (Sorcerer's) Stone (#1)",

"Chamber of Secrets (#2)",

"Prisoner of Azkaban (#3)",

"Goblet of Fire (#4)",

"Order of the Phoenix (#5)",

"Half-Blood Prince (#6)",

"Deathly Hallows (#7)"

)

book\_name\_tbl <- tibble(

book = paste0("HP\_", 1:7),

book\_name = factor(book\_names, levels = book\_names)

)

hp <- hp\_survey %>%

# Extract numerical score

rename(score\_chr = score) %>%

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

# Add book names

left\_join(y = book\_name\_tbl, by = "book")

hp

## # A tibble: 657 x 5

## person book score\_chr score book\_name

##

## 1 1 HP\_6 5 - Excellent 5 Half-Blood Prince (#6)

## 2 1 HP\_7 5 - Excellent 5 Deathly Hallows (#7)

## 3 2 HP\_1 3 - Good 3 Philosopher's (Sorcerer's) Stone (#1)

## 4 2 HP\_4 5 - Excellent 5 Goblet of Fire (#4)

## 5 2 HP\_5 2 - Fair 2 Order of the Phoenix (#5)

## # ... with 652 more rows

**Subset uniformity**

The first step in the survey was to choose the first element in the randomly shuffled list to simulate generation of random subset from all books. Each of 127 list element was connected to one subset. Lets visualize subset frequency to ensure a good faith of respondents:

# Compute subset representations

hp\_subsets <- hp %>%

arrange(person, book) %>%

group\_by(person) %>%

summarise(subset = paste0(book, collapse = "-"))

# Compute the number of actually picked subsets

n\_distinct(hp\_subsets$subset)

## [1] 95

# Visualize

hp\_subsets %>%

ggplot(aes(subset)) +

geom\_bar(fill = hp\_pal["Gryff"]) +

labs(

x = "Subset", y = "Number of times subset was picked",

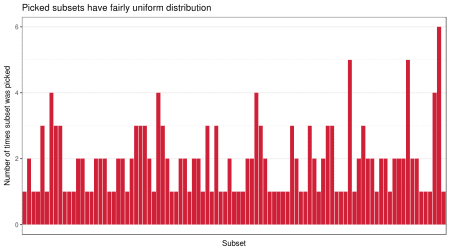
title = "Picked subsets have fairly uniform distribution"

) +

scale\_x\_discrete(labels = NULL) +

theme\_bar() +

theme(axis.ticks.x = element\_blank())



So there are 95 subsets actually picked and their distribution seems reasonably uniform. This is enough for me to confirm that randomization for subsets was successful.

**Book presence**

Other important thing to explore is number of times book was actually rated:

hp %>%

ggplot(aes(book\_name)) +

geom\_bar(fill = hp\_pal["Huffl"]) +

# Cool way to wrap labels for a given width

scale\_x\_discrete(labels = function(x) str\_wrap(x, width = 15)) +

labs(

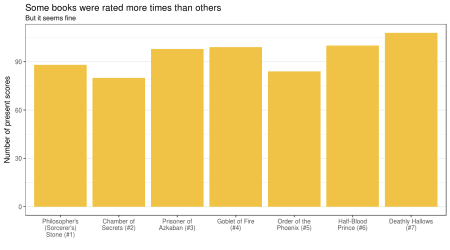
x = "", y = "Number of present scores",

title = "Some books were rated more times than others",

subtitle = "But it seems fine"

) +

theme\_bar()



**Book scores**

The most obvious way to summarise book “performance” is its mean score of numerical representation of scale. Using mean is not harmful in this study as no outlier can be present.

hp\_book\_score <- hp %>%

group\_by(book\_name) %>%

summarise(mean\_score = round(mean(score), digits = 2)) %>%

arrange(desc(mean\_score))

hp\_book\_score

## # A tibble: 7 x 2

## book\_name mean\_score

##

## 1 Prisoner of Azkaban (#3) 4.19

## 2 Half-Blood Prince (#6) 4.13

## 3 Goblet of Fire (#4) 4.00

## 4 Deathly Hallows (#7) 3.96

## 5 Philosopher's (Sorcerer's) Stone (#1) 3.91

## 6 Order of the Phoenix (#5) 3.90

## 7 Chamber of Secrets (#2) 3.55

**So, “the best” book seems to be “Harry Potter and the Prisoner of Azkaban (#3)”**.

For more understanding of results, lets also visualize score distribution.

hp %>%

# Compute share of score per book

count(book\_name, score) %>%

group\_by(book\_name) %>%

mutate(share = n / sum(n)) %>%

ungroup() %>%

# Visualize

ggplot() +

geom\_col(

aes(score, share, colour = score, fill = score),

show.legend = FALSE

) +

geom\_text(

data = hp\_book\_score,

mapping = aes(label = paste0("Mean = ", mean\_score)),

x = -Inf, y = Inf,

hjust = -0.05, vjust = 1.3

) +

facet\_wrap(~ book\_name) +

scale\_x\_continuous(

breaks = 1:5,

labels = c("1\nPoor", "2\nFair", "3\nGood",

"4\nVery\nGood", "5\nExcellent")

) +

scale\_fill\_gradient(low = hp\_pal["Raven"], high = hp\_pal["Raven\_light"]) +

scale\_colour\_gradient(low = hp\_pal["Raven"], high = hp\_pal["Raven\_light"]) +

labs(

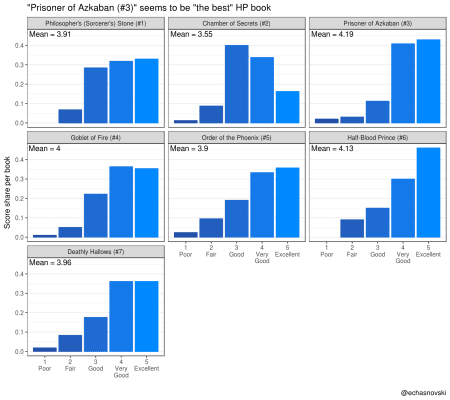
x = "", y = "Score share per book",

title = '"Prisoner of Azkaban (#3)" seems to be "the best" HP book',

caption = "@echasnovski"

) +

theme\_bar()



**Competition results**

**Formats of comperes**

Understanding of **competition** is quite general: it is a set of **games** (abstract event) in which **players** (abstract entity) gain some abstract **scores** (typically numeric). Inside games all players are treated equally. The most natural example is sport results, however not the only one. For example, product rating can be considered as a competition between products as “players”. Here a “game” is a customer that reviews a set of products by rating them with numerical “score” (stars, points, etc.).

In case of Harry Potter Books Survey results “game” is an act of respondent taking part in survey, “player” – Harry Potter book, “score” – discrete scale values converted to numerical score from 1 to 5.

In comperes there are two supported formats of competition results:

* **Long format**. It is the most abstract way of presenting competition results. Basically, it is a data frame (or tibble) with columns game (game identifier), player (player identifier) and score where *each row represents the score of particular player in particular game*. One game can consist from **variable** number of players which makes this format more usable. Extra columns are allowed.
* **Wide format** is a more convenient way to store results with **fixed** number of players in a game. *Each row represents scores of all players in particular game*. Data should be organized in pairs of columns “player”-“score”. Identifier of a pair should go after respective keyword and consist only from digits. For example: player1, score1, player2, score2. Order doesn’t matter. Column game is optional. Extra columns are also allowed.

Programmatically these formats are implemented as S3 classes longcr and widecr respectively. Essentially, they are tibbles with fixed structure. Objects of these classes should be created using functions as\_longcr() and as\_widecr() which also do conversions to other format.

**Conversion**

hp\_survey presents results in **long format**.

hp\_cr <- hp\_survey %>%

transmute(

game = person, player = book,

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

) %>%

as\_longcr()

hp\_cr

## # A longcr object:

## # A tibble: 657 x 3

## game player score

##

## 1 1 HP\_6 5

## 2 1 HP\_7 5

## 3 2 HP\_1 3

## 4 2 HP\_4 5

## 5 2 HP\_5 2

## # ... with 652 more rows

Here is the demonstration of conversion to **wide format**. It detects the maximum number of players in a game, which is 7, and assumes that data is missing in games with less number of players.

as\_widecr(hp\_cr)

## # A widecr object:

## # A tibble: 182 x 15

## game player1 score1 player2 score2 player3 score3 player4 score4

##

## 1 1 HP\_6 5 HP\_7 5 NA NA

## 2 2 HP\_1 3 HP\_4 5 HP\_5 2 HP\_6 4

## 3 3 HP\_1 3 HP\_3 4 HP\_5 1 NA

## 4 4 HP\_6 5 HP\_7 5 NA NA

## 5 5 HP\_4 4 HP\_5 3 NA NA

## # ... with 177 more rows, and 6 more variables: player5 ,

## # score5 , player6 , score6 , player7 , score7

**Head-to-Head**

**Functionality of comperes**

Head-to-Head value is a **summary statistic of direct confrontation between two players**. It is assumed that this value can be computed based only on the players’ matchups (results for ordered pairs of players from one game). In other words, every game is converted into series of “subgames” between ordered pairs of players (including selfplay) which is stored as widecr object. After that, summary of item, defined by columns player1 and player2, is computed.

comperes has function get\_matchups() for computing matchups:

get\_matchups(hp\_cr)

## # A widecr object:

## # A tibble: 2,697 x 5

## game player1 score1 player2 score2

##

## 1 1 HP\_6 5 HP\_6 5

## 2 1 HP\_6 5 HP\_7 5

## 3 1 HP\_7 5 HP\_6 5

## 4 1 HP\_7 5 HP\_7 5

## 5 2 HP\_1 3 HP\_1 3

## # ... with 2,692 more rows

To compute multiple Head-to-Head values, use h2h\_long() supplying competition results and summarizing expressions in [dplyr::summarise()](https://dplyr.tidyverse.org/reference/summarise.html) fashion. They will be applied to a data frame of matchups.

hp\_cr\_h2h <- hp\_cr %>% h2h\_long(

# Number of macthups

n = n(),

# Number of wins plus half the number of ties

# num\_wins() is a function from comperes to compute number of times

# first score is bigger than second one

num\_wins = num\_wins(score1, score2, half\_for\_draw = TRUE),

# Mean rating of a book scored in matchups with other books

mean\_score = mean(score1),

# Mean rating difference of books scored in direct matchups

mean\_score\_diff = mean(score1 - score2)

) %>%

mutate\_if(is.numeric, funs(round(., 2)))

hp\_cr\_h2h

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

## # A tibble: 49 x 6

## player1 player2 n num\_wins mean\_score mean\_score\_diff

##

## 1 HP\_1 HP\_1 88. 44.0 3.91 0.

## 2 HP\_1 HP\_2 42. 29.5 3.88 0.500

## 3 HP\_1 HP\_3 51. 19.5 3.92 -0.390

## 4 HP\_1 HP\_4 48. 24.0 3.79 0.0400

## 5 HP\_1 HP\_5 42. 21.5 3.79 0.

## # ... with 44 more rows

So here we see, for example, that HP\_1 and HP\_2 had 42 matchups, i.e. they were rated by the same person 42 times. HP\_1 “won” 29.5 (respecting ties) times, gained mean score of 3.88 in those matchups and had, on average, 0.5 points more.

There is also an h2h\_mat() function which computes a matrix of Head-to-Head values for one expression.

hp\_cr %>% h2h\_mat(num\_wins(score1, score2, half\_for\_draw = TRUE))

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

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

## HP\_1 44.0 29.5 19.5 24.0 21.5 17.0 24.0

## HP\_2 12.5 40.0 12.0 11.5 10.5 12.0 19.0

## HP\_3 31.5 32.0 49.0 31.5 28.0 25.0 33.5

## HP\_4 24.0 33.5 26.5 49.5 23.5 30.5 31.5

## HP\_5 20.5 25.5 15.0 24.5 42.0 23.0 24.5

## HP\_6 25.0 30.0 20.0 27.5 24.0 50.0 34.0

## HP\_7 26.0 34.0 21.5 29.5 25.5 26.0 54.0

For more convenient usage, comperes has a list h2h\_funs of some common Head-to-Head functions stored as [expressions](http://rlang.r-lib.org/reference/quotation.html). To use them you need a little bit of [rlang](http://rlang.r-lib.org/index.html)’s [unquoting magic](http://rlang.r-lib.org/reference/quasiquotation.html).

h2h\_funs[1:3]

## $mean\_score\_diff

## mean(score1 - score2)

##

## $mean\_score\_diff\_pos

## max(mean(score1 - score2), 0)

##

## $mean\_score

## mean(score1)

hp\_cr %>% h2h\_long(!!! h2h\_funs)

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

## # A tibble: 49 x 11

## player1 player2 mean\_score\_diff mean\_score\_diff\_pos mean\_score

##

## 1 HP\_1 HP\_1 0. 0. 3.91

## 2 HP\_1 HP\_2 0.500 0.500 3.88

## 3 HP\_1 HP\_3 -0.392 0. 3.92

## 4 HP\_1 HP\_4 0.0417 0.0417 3.79

## 5 HP\_1 HP\_5 0. 0. 3.79

## # ... with 44 more rows, and 6 more variables: sum\_score\_diff ,

## # sum\_score\_diff\_pos , sum\_score , num\_wins ,

## # num\_wins2 , num

**Harry Potter books**

Head-to-Head “performance” of Harry Potter books is summarised in the following plot:

hp\_cr\_h2h %>%

gather(h2h\_fun, value, -player1, -player2) %>%

# Manually produce a dummy colour variable to use in facets

group\_by(h2h\_fun) %>%

mutate(col = (value - min(value)) / (max(value) - min(value))) %>%

ungroup() %>%

# Make factors for correct orders

mutate(

player1 = factor(player1, levels = rev(sort(unique(player1)))),

player2 = factor(player2, levels = sort(unique(player2))),

h2h\_fun = factor(h2h\_fun,

levels = c("n", "num\_wins",

"mean\_score", "mean\_score\_diff")),

h2h\_fun = recode(

h2h\_fun,

n = "Number of matchups (ratings by common person)",

num\_wins = 'Number of "wins" in matchups (half for ties)',

mean\_score = "Mean score in matchups",

mean\_score\_diff = "Mean score difference in matchups"

)

) %>%

# Visualize

ggplot(aes(player1, player2)) +

geom\_text(

aes(label = value, colour = col),

size = 5, fontface = "bold", show.legend = FALSE

) +

facet\_wrap(~ h2h\_fun, scales = "free") +

# To coordinate well with matrix form of Head-to-Head results

coord\_flip() +

scale\_colour\_gradient(low = hp\_pal["Slyth"], high = hp\_pal["Gryff"]) +

labs(

x = "", y = "",

title = "Head-to-Head performance of Harry Potter books",

subtitle = paste0(

'"HP\_x" means Harry Potter book number "x" in series\n',

"Numbers are Head-to-Head values of book in row against book in column"

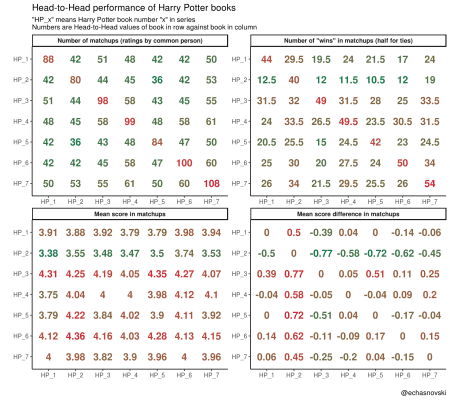
),

caption = "@echasnovski"

) +

theme\_classic() +

theme(strip.text = element\_text(face = "bold"))



There is a lot of information hidden in this plot. The most obvious discoveries:

* It happened that book *“HP\_7”* (“Deathly Hallows”) was rated with *“HP\_4”* (“Goblet of Fire”) by one person the most: 61 times.
* *“HP\_7”* scored over *“HP\_2”* (“Chamber of Secrets”) the most wins (34, half for ties) as did *“HP\_6”* (“Half-Blood Prince”) over “HP\_7”.
* Book *“HP\_6”* made the highest mean score of 4.36 in matchups with *“HP\_2”*, which is bigger by 0.23 from its overall mean score.
* In terms of score differences, *“HP\_3”* (“Prisoner of Azkaban”) did best in matchups with *“HP\_2”*, scoring on average 0.77 points more. This pair also represents “the best” and “the worst” books in terms of mean score.

**Conclusion**

* A public call for help in creating data set for R package shouldn’t be made on Reddit but rather on R-bloggers or Twitter.
* Among all original Harry Potter books, “Harry Potter and the Prisoner of Azkaban” seems to be considered “best” among R users. “Harry Potter and the Chamber of Secrets” suffers the opposite fate.
* Package comperes is useful for storing, manipulating and summarising abstract competition results.
* However informative, manually inspecting competition results with direct summaries and Head-to-Head tables is hard. They can display complex nature of performance relations between players. Next analysis of Harry Potter Books Survey data will be using my package [comperank](https://github.com/echasnovski/comperank) which implements different ranking methods for automatic discovery of player’s performance.

[sessionInfo()](http://www.questionflow.org/2018/05/09/harry-potter-and-competition-results-with-comperes/#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 comperes\_0.2.0 ggplot2\_2.2.1 stringr\_1.3.0

## [5] rlang\_0.2.0 tidyr\_0.8.0.9000 dplyr\_0.7.5.9000

##

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

## [1] Rcpp\_0.12.16 pillar\_1.2.1 compiler\_3.4.4 plyr\_1.8.4

## [5] bindr\_0.1.1 tools\_3.4.4 digest\_0.6.15 evaluate\_0.10.1

## [9] tibble\_1.4.2 gtable\_0.2.0 pkgconfig\_2.0.1 cli\_1.0.0

## [13] yaml\_2.1.17 blogdown\_0.5 xfun\_0.1 knitr\_1.20

## [17] rprojroot\_1.3-2 grid\_3.4.4 tidyselect\_0.2.4 glue\_1.2.0

## [21] R6\_2.2.2 rmarkdown\_1.9 bookdown\_0.7 purrr\_0.2.4

## [25] magrittr\_1.5 backports\_1.1.2 scales\_0.5.0 htmltools\_0.3.6

## [29] assertthat\_0.2.0 colorspace\_1.3-2 labeling\_0.3 utf8\_1.1.3

## [33] stringi\_1.1.6 lazyeval\_0.2.1 munsell\_0.4.3 crayon\_1.3.4