Package 'dndR'

March 30, 2023

Type Package

Title Dungeons & Dragons Functions for Players and Dungeon Masters

Version 1.1.0

Maintainer Nicholas Lyon <njlyon@alumni.iastate.edu>

Description The goal of 'dndR' is to provide a suite of Dungeons & Dragons related functions. This package is meant to be useful both to players and Dungeon Masters (DMs). All functions currently focus on Fifth Edition (a.k.a. ``5e'') but once the next edition is published functions will likely be expanded to include any rule changes.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Language en-US

RoxygenNote 7.2.3

Depends R (>= 3.5)

Imports dplyr, ggplot2, magrittr, purrr, stringr, tidyr

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Nicholas Lyon [aut, cre] (<https://orcid.org/0000-0003-3905-1078>, https://njlyon0.github.io/), Tim Schatto-Eckrodt [ctb] (https://kudusch.de/), Humberto Nappo [ctb] (<https://orcid.org/0000-0001-7810-1635>)

Repository CRAN

Date/Publication 2023-03-30 09:00:02 UTC

R topics documented:

`ability_scores` *Roll for All Ability Scores*

Description

Rolls for six ability scores using the desired method of rolling (4d6 drop lowest, 3d6, or 1d20). Doesn't assign abilities to facilitate player selection of which score should be each ability for a given character. Prints a warning if the total of all abilities is less than 70 or if any one ability is less than 8.

Usage

```
ability_scores(method = "4d6", quiet = FALSE)
```

Arguments

`method`	(character) string of "4d6", "3d6", or "1d20" ("d20" also accepted). Enter your preferred method of rolling for each ability score ("4d6" drops lowest before summing)
`quiet`	(logical) whether to print warnings if the total score is very low or one ability score is very low

Value

(dataframe) two columns and six rows for ability score for each ability

Examples

```
# Roll ability scores using four d6 and dropping the lowest
ability_scores(method = "4d6")

# Roll using 3d6 and dropping nothing
ability_scores("3d6")

# Or if you're truly wild, just roll a d20 for each ability
ability_scores('d20')
```

---

ability_singular *Rolls for a Single Ability Score*

Description

Rolls for a single ability score using the specified method of dice rolling.

Usage

```
ability_singular(method = "4d6")
```

Arguments

method	(character) string of "4d6", "3d6", or "1d20" ("d20" also accepted). Enter your preferred method of rolling for each ability score ("4d6" drops lowest before summing)

Value

(numeric) vector of roll outcomes (not summed)

---

class_block *Assign Ability Scores Based on Class*

Description

Assign rolled ability scores based on the recommendations for quick class building given in the Player's Handbook (PHB).

Usage

```
class_block(
  class = NULL,
  score_method = "4d6",
  scores_rolled = FALSE,
  scores_df = NULL,
  quiet = FALSE
)
```

Arguments

class	(character) name of character class (supported classes returned by ‘dnd_classes()‘). Also supports "random" and will randomly select a supported class
score_method	(character) preferred method of rolling for ability scores "4d6", "3d6", or "1d20" ("d20" also accepted synonym of "1d20"). Only values accepted by ‘ability_scores()‘ are accepted here
scores_rolled	(logical) whether ability scores have previously been rolled (via ‘ability_scores()‘). Defaults to FALSE
scores_df	(dataframe) if ’scores_rolled’ is TRUE, the name of the dataframe object returned by ‘ability_scores()‘
quiet	(logical) whether to print warnings if the total score is very low or one ability score is very low

Value

(dataframe) two columns and six rows

Examples

```
# Can roll up a new character of the desired class from scratch
class_block(class = "wizard", score_method = "4d6")

# Or you can roll separately and then create a character with that dataframe
my_scores <- ability_scores(method = "4d6")
class_block(class = "fighter", scores_rolled = TRUE, scores_df = my_scores)
```

---

coin *Flip a Coin*

Description

Picks a random number from 1-2. Essentially a "d2".

Usage

```
coin()
```

Value

(numeric) result of coin flip (either 1 or 2)

`cr_convert` *Convert Challenge Rating to Experience Points*

Description

Converts challenge rating (CR) into experience points (XP) using two formulas for a parabola (one for CR less than/equal to 20 and one for greater than 20). The relationship between CR and XP in the Dungeon Master's Guide (DMG) is disjointed in this way so this is a reasonable move. Accepts '1/8', '1/4, and '1/2' in addition to numbers between 1 and 30.

Usage

```
cr_convert(cr = NULL)
```

Arguments

`cr`	(numeric) Challenge rating for which you want to calculate experience points

Value

(numeric) value of XP equivalent to the user-supplied challenge rating

---

`d10` *Roll a Ten-Sided Dice ("d10")*

Description

Picks a random number from 1-10

Usage

```
d10()
```

Value

(numeric) result of "roll" of specified dice type

d100 *Roll a One Hundred-Sided Dice ("d100")*

Description

Picks a random number from 1-100

Usage

```
d100()
```

Value

(numeric) result of "roll" of specified dice type

d12 *Roll a Twelve-Sided Dice ("d12")*

Description

Picks a random number from 1-12

Usage

```
d12()
```

Value

(numeric) result of "roll" of specified dice type

d2 *Roll a Two-Sided Dice*

Description

Picks a random number from 1-2. Essentially flips a coin.

Usage

```
d2()
```

Value

(numeric) result of "roll" of specified dice type

d20 — *Roll a Twenty-Sided Dice ("d20")*

Description

Picks a random number from 1-20

Usage

```
d20()
```

Value

(numeric) result of "roll" of specified dice type

d3 — *Roll a Three-Sided Dice*

Description

Picks a random number from 1-3

Usage

```
d3()
```

Value

(numeric) result of "roll" of specified dice type

d4 — *Roll a Four-Sided Dice ("d4")*

Description

Picks a random number from 1-4

Usage

```
d4()
```

Value

(numeric) result of "roll" of specified dice type

d6 *Roll a Six-Sided Dice ("d6")*

Description

Picks a random number from 1-6

Usage

```
d6()
```

Value

(numeric) result of "roll" of specified dice type

d8 *Roll an Eight-Sided Dice ("d8")*

Description

Picks a random number from 1-8

Usage

```
d8()
```

Value

(numeric) result of "roll" of specified dice type

dnd_classes *Return Vector of Accepted Classes*

Description

Simply returns a vector of classes that ‘class_block()‘ accepts currently. Submit an issue on the GitHub repository if you want a class added.

Usage

```
dnd_classes()
```

Value

(character) vector of accepted class names

Examples

```
# Want to check which classes this package supports?
dnd_classes()
```

`dnd_damage_types` *Return Vector of Supported DnD Damage Types*

Description

Simply returns a vector of damage types in DnD

Usage

```
dnd_damage_types()
```

Value

character vector of damage types

Examples

```
# Full set of damage types included in DnD Fifth Edition (5e)
dnd_damage_types()
```

`dnd_races` *Return Vector of Supported DnD Races*

Description

Simply returns a vector of races that ‘race_mods()‘ accepts currently. Submit an issue on the GitHub repository if you want a race added.

Usage

```
dnd_races()
```

Value

(character) vector of supported race designations

Examples

```
# Want to check which races this package supports?
dnd_races()
```

monsters *Dungeons and Dragons Quick Table for Creature Statistics*

Description

On pages 274 and 275 in the Dungeon Master's Guide (Fifth Edition) there are two tables that relate creature Challenge Rating (CR) to various vital statistics (armor, hit points, etc.) and to Experience Points (XP). These tables have been transcribed into this data object for ease of reference.

Usage

```
monsters
```

Format

Dataframe with 7 columns and 34 rows

Challenge Challenge Rating (CR) expressed as a number

DMG_XP Experience Points (XP) for that CR as dictated by the DMG

Prof_Bonus Modifier to add to rolls where the creature is proficient

Armor_Class Armor class of the creature

HP_Range Range of hit points (HP) for the creature

HP_Average Average of minimum and maximum HP of range for the creature

Attack_Bonus Modifier to add to the creature's attack rolls

Save_DC Save Difficulty Class (DC) for rolls against the creature's spells / certain abilities

Source

Mearls, M., Crawford, J., Perkins, C., Wyatt, J. et al. Dungeon Master's Guide (Fifth Edition). 2014.

monster_creator *Creates a Monster for Given Party Level and Size*

Description

Returns the vital statistics of a randomized monster based on a the average player level and number of players in the party. This function follows the advice of [Zee Bashew](https://twitter.com/Zeebashew) on how to build interesting, challenging monsters for your party. These monsters are built somewhat according to the Dungeon Master's Guide for creating monsters, partly Zee's [YouTube video on homebrewing monsters based on The Witcher videogame](https://www.youtube.com/watch?v=GhjkPv4qo5w), and partly on my own sensibilities about scaling the difficulty of a creature. Creatures are spawned randomly so you may need to re-run the function several times (or mentally modify one or more parts of the output) to get a monster that fits your campaign and players, but the vulnerabilities and resistances should allow for cool quest building around what this function provides. Happy DMing!

Usage

```
monster_creator(party_level = NULL, party_size = NULL)
```

Arguments

party_level	(numeric) indicating the average party level. If all players are the same level, that level is the average party level
party_size	(numeric) indicating how many player characters (PCs) are in the party

Value

(dataframe) two columns and 15 rows

Examples

```
# Creates a monster from the specified average party level
monster_creator(party_level = 4, party_size = 3)
```

---

monster_stats *Quickly Identify Monster Statistics*

Description

Quickly identify the vital statistics of a single creature worth the provided experience points (XP) or Challenge Rating (CR). Uses the table provided in p. 274-275 of the Dungeon Master's Guide. Accepts Challenge Ratings of 0, '1/8', '1/4, and '1/2' in addition to numbers between 1 and 30. CR is *not necessary* to provide **if** XP is provided.

Usage

```
monster_stats(xp = NULL, cr = NULL)
```

Arguments

xp	(numeric) experience point (XP) value of the monster
cr	(numeric) challenge rating (CR) of the monster. Note that this is NOT necessary if XP is provided

Value

(dataframe) two columns and eight rows

npc_creator *Create a Non-Player Character (NPC)*

Description

Randomly selects a race and job for a user-specified number of NPCs

Usage

```
npc_creator(npc_count = 1)
```

Arguments

npc_count	(numeric) number of NPCs for which to choose race/positions

Value

(dataframe) dataframe with two columns (one for race and one for job) and a number of rows equal to ‘npc_count‘

Examples

```
# Create some information for an NPC
npc_creator(npc_count = 1)
```

party_diagram *Generate a Diagram of a Party's Ability Scores*

Description

Input a party's ability scores and visualize either by ability or player character. Includes dashed line for average of ability scores within chosen ‘by‘ parameter.

Usage

```
party_diagram(by = "player", pc_stats = NULL, quiet = FALSE)
```

Arguments

by	(character) either "player" (default) or "ability". Defines the facets of the party diagram
pc_stats	(null / list) either ‘NULL‘ (default) or named list of ability scores for each character. If ‘NULL‘, player names and scores are requested interactively in the console
quiet	(logical) if FALSE (default), prints interactively assembled PC list for ease of subsequent use

Value

(ggplot object) party diagram as a ggplot object

Examples

```
# Create named list of PCs and their scores
party_list <- list(
Vax = list(
STR = "10", DEX = "13", CON = "14", INT = "15", WIS = "16", CHA = "12"),
Beldra = list(
STR = "20", DEX = "15", CON = "10", INT = "10", WIS = "11", CHA = "12"),
Rook = list(
STR = "10", DEX = "10", CON = "18", INT = "9", WIS = "11", CHA = "16"))

# Create a party diagram using that list (by player)
party_diagram(by = "player", pc_stats = party_list, quiet = TRUE)

# Can easily group by ability with the same list!
party_diagram(by = "ability", pc_stats = party_list, quiet = FALSE)
```

---

pc_creator *Create a Player Character (PC)*

Description

Stat out a player character (PC) of specified race and class using your preferred method for rolling ability scores.

Usage

```
pc_creator(
  class = NULL,
  race = NULL,
  score_method = "4d6",
  scores_rolled = FALSE,
  scores_df = NULL,
  quiet = FALSE
)
```

Arguments

class	(character) name of character class (supported classes returned by 'dnd_classes()'). Also supports "random" and will randomly select a supported class. Random class returned as message

race	(character) name of character race (supported classes returned by ‘dnd_races()‘). Also supports "random" and will randomly select a supported race. Random race returned as message
score_method	(character) preferred method of rolling for ability scores "4d6", "3d6", or "1d20" ("d20" also accepted synonym of "1d20"). Only values accepted by ‘ability_scores()‘ are accepted here
scores_rolled	(logical) whether ability scores have previously been rolled (via ‘ability_scores()‘). Defaults to FALSE
scores_df	(dataframe) if ’scores_rolled’ is TRUE, the name of the dataframe object returned by ‘ability_scores()‘
quiet	(logical) whether to print warnings if the total score is very low or one ability score is very low

Value

(dataframe) raw ability score, race modifier, total ability score, and the roll modifier for each of the six abilities

Examples

```
# Create a PC's base statistics from scratch
pc_creator(class = 'barbarian', race = 'half orc', score_method = "4d6", quiet = TRUE)

# Or you can roll separately and then create a character with that dataframe
my_scores <- ability_scores(method = "4d6", quiet = TRUE)
pc_creator(class = 'sorcerer', race = 'dragonborn', scores_rolled = TRUE, scores_df = my_scores)
```

---

pc_level_calc *Calculate Player Character (PC) Level from Current Experience Points (XP)*

Description

Uses total player experience points (XP) to identify player character (PC) level and proficiency modifier. Only works for a single PC at a time (though this is unlikely to be an issue if all party members have the same amount of XP)

Usage

```
pc_level_calc(player_xp = NULL)
```

Arguments

player_xp	(numeric) total value of experience points earned by one player

Value

(dataframe) current player level, XP threshold for that level, and the proficiency modifier used at that level

Examples

```
# Calculate player level from XP earned
pc_level_calc(player_xp = 950)
```

`race_mods` *Identify Race-Based Ability Modifiers*

Description

Identify the race-based ability modifiers identified in the Player's Handbook (PHB).

Usage

```
race_mods(race = NULL)
```

Arguments

`race`	(character) string of race (supported classes returned by 'dnd_races()'). Also supports "random" and will randomly select a supported race

Value

(dataframe) two columns and as many rows as there are abilities modified by the race

Examples

```
# Identifies race modifiers of provided race
race_mods(race = "mountain dwarf")
```

roll *Roll Any Number of Dice*

Description

Rolls the Specified Number and Type of Dice

Usage

```
roll(dice = "d20", show_dice = FALSE)
```

Arguments

dice	(character) specifying the number of dice and which type (e.g., "2d4" for two, four-sided dice). Defaults to a single twenty-sided die
show_dice	(logical) whether to print the values of each individual die included in the total. Defaults to FALSE

Value

(numeric) sum of specified dice outcomes

Examples

```
# Roll your desired dice (i.e., randomly sample the specified die)
roll(dice = "4d6")

# Returned as a number so you can add rolls together or integers
roll('1d20') + 5
roll('2d8') + roll('1d4')
```

xp_cost *Adjust the XP Total by Number of Monsters and Party Size*

Description

Encounters are more difficult than the total of the monsters' experience points (XP). Both the number of monsters making attacks and the number of players attacking those creatures can affect the difficulty of an encounter. The Dungeon Master's Guide (DMG) accounts for this by providing an XP multiplier for given party sizes and numbers of monsters. This function accepts the unmodified total of the monsters' XP and adjusts this as specified in the DMG without the pain of the tables in that book.

Usage

```
xp_cost(monster_xp = NULL, monster_count = NULL, party_size = NULL)
```

Arguments

monster_xp	(numeric) XP total across all monsters
monster_count	(numeric) count for the number of monsters in the encounter
party_size	(numeric) value for the number of PCs in the party

Value

(numeric) value for "realized" XP

Examples

```
# Calculate the realized XP from the raw XP, number of monsters, and number of PCs
xp_cost(monster_xp = 100, monster_count = 3, party_size = 2)
```

---

xp_pool *Calculate Total XP of Monsters for Given Party Level and Difficulty*

Description

Returns the total XP (experience points) of all creatures that would make an encounter the specified level of difficulty for a party of the supplied level. This 'pool' can be used by a GM (game master) to "purchase" monsters to identify how many a party is likely to be able to handle given their average level. NOTE: this does not take into account creature-specific abilities or traits so care should be taken if a monster has many such traits that modify its difficulty beyond its experience point value.

Usage

```
xp_pool(party_level = NULL, party_size = NULL, difficulty = NULL)
```

Arguments

party_level	(numeric) integer indicating the average party level. If all players are the same level, that level is the average party level
party_size	(numeric) integer indicating how many player characters (PCs) are in the party
difficulty	(character) one of "easy", "medium", "hard", or "deadly" for the desired difficulty of the encounter.

Value

(numeric) total encounter XP as an integer

Examples

```
# Supply a party level and difficulty and get the total XP of such an encounter
xp_pool(party_level = 3, party_size = 2, difficulty = 'medium')
```

Index