Package: dndR (via r-universe)

November 2, 2024

Type Package

Title Dungeons & Dragons Functions for Players and Dungeon Masters

Version 2.0.0.900

Date 2024-04-26

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 7.3.1

URL https://njlyon0.github.io/dndR/

BugReports https://github.com/njlyon0/dndR/issues

Depends R (>= 3.5)

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

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Repository https://njlyon0.r-universe.dev

RemoteUrl https://github.com/njlyon0/dndr

RemoteRef HEAD

RemoteSha d2ca14221fbbf7edd562d6b7d906805082a8e7f9

Contents

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)

---

creatures *Dungeons and Dragons Creature Information*

Description

Creatures in Dungeons and Dragons all fall into certain, well-documented categories. This table summarizes all of that information into a long format dataframe for easy navigation. Unless otherwise noted, all creature querying functions in ‘dndR‘ use this table as their starting point.

Usage

```
creatures
```

Format

Dataframe with 26 columns and 1721 rows

creature_name Name of the creature

creature_source Source book for the creature

STR Strength ability score of creature with roll modifier indicated parenthetically

DEX Dexterity ability score of creature with roll modifier indicated parenthetically

CON Constitution ability score of creature with roll modifier indicated parenthetically

INT Intelligence ability score of creature with roll modifier indicated parenthetically

WIS Wisdom ability score of creature with roll modifier indicated parenthetically

CHA Charisma ability score of creature with roll modifier indicated parenthetically

creature_size Size category of the creature (one of 'tiny', 'small', 'medium', 'large', 'huge', or 'gargantuan')

creature_type Type of the creature (e.g., undead, elemental, etc.)

creature_alignment The creature's alignment (e.g., chaotic evil, etc.)

creature_xp Experience point (XP) value of the creature

creature_cr Challenge rating (CR) of the creature

languages Any languages understood or spoken by the creature

skills Any skills in which the creature is proficient and the roll modifiers for each

speed Movement speed of the creature

hit_points Number of hit points (HP) of the creature (and the dice to roll if rolling for HP is desired)

armor_class Armor class of the creature

senses Any special senses of the creature

saving_throws Any saving throws in which the creature is proficient and the roll modifiers for each

damage_immunities Damage types to which the creature is immune (i.e., no damage)

damage_resistances Damage types to which the creature is resistant (i.e., half damage)

damage_vulnerabilities Damage types to which the creature is vulnerable (i.e., double damage)

condition_immunities Conditions to which the creature is immune

abilities Description of all abilities the creature has as well as any bonus actions or reactions it can take. Each item name is surrounded by triple asterisks

actions Description of all actions the creature can take. Each item name is surrounded by triple asterisks

Source

Crawford, J., Hickman, L., Hickman, T., Lee, A., Perkins, C., Whitters, R. Curse of Strahd. Wizards of the Coast 2015.

Waterdeep: Dungeon of the Mad Mage. Wizards of the Coast 2018.

Elemental Evil. Wizards of the Coast 2015.

Explorer's Guide to Wildemount. Wizards of the Coast 2020.

Guildmasters' Guide to Ravnica. Wizards of the Coast 2018.

Lost Mine of Phandelver. Wizards of the Coast 2014.

Mearls, M. and Crawford, J. Dungeons & Dragons Monster Manual (Fifth Edition). Wizards of the Coast 2014

Morenkainen's Tome of Foes. Wizards of the Coast 2018.

Out of the Abyss. Wizards of the Coast 2015.

Storm King's Thunder. Wizards of the Coast 2016.

Tales from the Yawning Portal. Wizards of the Coast 2017.

Tomb of Annihilation. Wizards of the Coast 2017.

Baur, W. Tome of Beasts. Paizo Inc. 2016.

Tome of Horrors. Frog God Games 2019.

Tyranny of Dragons. Wizards of the Coast 2015.

Volo's Guide to Monsters. Wizards of the Coast 2016.

`creature_list` *List Creatures Based on Criteria*

Description

Query list of Dungeons & Dragons creatures based on partial string matches between user inputs and the relevant column of the creature information data table. Currently supports users querying the creature list by creature name, size, type, source document, experience point (XP), and challenge rating (CR). All characters arguments are case-insensitive. XP and CR may be specified as either characters or numbers but match to creature must be exact in either case (rather than partial). Any argument set to 'NULL' (the default) will not be used to include/exclude creatures from the returned set of creatures

Usage

```
creature_list(
  name = NULL,
  size = NULL,
  type = NULL,
  source = NULL,
  xp = NULL,
  cr = NULL
)
```

Arguments

`name`	(character) text to look for in creature names
`size`	(character) size(s) of creature
`type`	(character) creature 'type' (e.g., "undead", "elemental", etc.)

source	(character) source book/document of creature
xp	(character/numeric) experience point (XP) value of creature (note this must be an exact match as opposed to partial matches tolerated by other arguments)
cr	(character/numeric) challenge rating (CR) value of creature (note this must be an exact match as opposed to partial matches tolerated by other arguments)

Value

(dataframe) Up to 23 columns of information with one row per creature(s) that fit(s) the user-specified criteria. Fewer columns are returned when no creatures that fit the criteria have information for a particular category (e.g., if no queried creatures have damage vulnerabilities, that column will be excluded from the results). If no creatures fit the criteria, returns a message to that effect instead of a data object

Examples

```
# Identify medium undead creatures from the Monster Manual worth 450 XP
creature_list(type = "undead", size = "medium", source = "monster manual", xp = 450)
```

---

creature_text *Retrieve Full Creature Description Text by Creature Name*

Description

Accepts user-provided Dungeons & Dragons creature name(s) and returns the full set of creature information and the complete description text. Unlike 'dndR::creature_list', this function requires an exact match between the user-provided creature name(s) and how they appear in the main creature data object. The argument in this function is not case-sensitive.

Usage

```
creature_text(name = NULL)
```

Arguments

name	(character) exact creature name(s) for which to gather description information

Value

(dataframe) one column per creature specified by the user. Creature name is stored as the column name for that creature's information. Returns all fields for which there are data for at least one of the specified creatures so row number will vary with query (maximum 26 rows if all fields have information).

Examples

```
creature_text(name = c("hill giant", "goblin"))
```

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()
```

`encounter_creator` *Balance a Combat Encounter for Given Party Composition and Difficulty*

Description

Identifies set of creature XP values that constitute a balanced encounter of specified difficulty for given party composition information (i.e., average player character level and number of party members). Creature selection is semi-random so re-running this function will return similar but not necessarily identical results. It is not always possible to exactly spend all available XP so the true maximum XP and the realized XP (see ‘?dndR::xp_pool‘ and ‘?dndR::xp_cost‘) are both returned in the output for context. This function _will not_ exceed the allowed XP so you may need to alter the party information and/or difficulty arguments in order to return an encounter that meets your needs.

Usage

```
encounter_creator(
  party_level = NULL,
  party_size = NULL,
  difficulty = NULL,
  try = 5
)
```

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
`try`	(numeric) integer indicating the number of attempts to make to maximize encounter XP while remaining beneath the threshold defined by the other parameters

Value

(dataframe) creature experience point (XP) values as well as the maximum XP for an encounter of the specified difficulty and the realized XP cost of the returned creatures

Examples

```
# Create a hard encounter for a 2-person, 9th level party
encounter_creator(party_level = 9, party_size = 2, difficulty = "hard")
```

mod_calc *Calculate Modifier for Specified Ability Score*

Description

Ability scores (typically 0-20 for most creatures) relate to roll modifiers. These values are what a player or DM actually adds to a given skill or ability check. This function performs the simple calculation to identify the roll modifier that relates to the supplied ability score.

Usage

```
mod_calc(score = 10)
```

Arguments

score	(numeric) ability score value for which to identify the roll modifier

Value

(character) roll modifier for a given ability score. If positive, includes a plus sign to make the addition explicit. Negative values are also returned as characters for consistency with positive modifiers

Examples

```
# Calculate roll modifier for an ability score of 17
mod_calc(score = 17)
```

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

monster_table *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

```
monster_table
```

Format

Dataframe with 8 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). Wizards of the Coast 2014

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. Huge shout out to Tim Schatto-Eckrodt for contributing this function!

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). Big thanks to Humberto Nappo for contributing this function!

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)
```

probability_plot *Generate a Plot of the Frequency of Roll Outcomes*

Description

Input the number and type of dice to roll and the number of times to roll the dice. This is used to generate a plot of the real distribution of dice outcomes and create a ggplot2 plot of that result. A vertical dashed line is included at the median roll result. Note that low numbers of rolls may not generate realistic frequencies of outcomes

Usage

```
probability_plot(dice = "2d20", roll_num = 999)
```

Arguments

dice	(character) specifying the number of dice and which type (e.g., "2d4" for two, four-sided dice). Defaults to two, six-sided dice
roll_num	(integer) number of times to roll the specified dice to generate the data fro the probability plot. Defaults to 999

Value

(ggplot object) roll outcome frequency as a ggplot2 object

Examples

```
# Generate a probability plot of 3d8
probability_plot(dice = "3d8", roll_num = 99)
```

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")
```

`reroll` *Re-Roll 1s from a Prior Dice Roll*

Description

Re-rolls only the dice that "landed on" 1 from a prior use of ‘roll‘. Retains other dice results from the first roll but replaces the ones.

Usage

```
reroll(dice_faces, first_result = NULL)
```

Arguments

`dice_faces`	(numeric) number of sides on the die to be rerolled (i.e., type of dice without the "d" found in the ‘roll‘ function)
`first_result`	(numeric) vector of original dice results (including 1s to reroll)

Value

(numeric) vector of non-1 original dice results with re-rolled dice results appended

Examples

```
# Re-roll ones from a prior result
reroll(dice_faces = 8, first_result = c(1, 3, 1))
```

`roll` *Roll Any Number of Dice*

Description

Rolls the specified number and type of dice. Dice are specified in the shorthand common to Dungeons & Dragons (i.e., number of dice, "d", number of faces of those dice). Includes an argument for whether each die's value should be returned as a message (rather than just the total of all dice in the roll). Rolling two twenty-sided dice (i.e., "2d20") is assumed to be rolling with advantage/disadvantage so both numbers are returned.

Usage

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

Arguments

`dice`	(character) number and type of dice to roll specified in Dungeons & Dragons shorthand (e.g., "2d4" to roll two four-sided dice). Defaults to a single twenty-sided die (i.e., "1d20")
`show_dice`	(logical) whether to print the values of each individual die included in the total. Defaults to FALSE
`re_roll`	(logical) whether to re-roll 1s from the initial roll result. Defaults to FALSE

Value

(numeric) sum of specified dice outcomes

Examples

```
# Roll your desired dice
roll(dice = "4d6", show_dice = TRUE)

# Returned as a number so you can add rolls together or integers
roll(dice = '1d20') + 5

# Can also re-roll ones if desired
roll(dice = '4d4', re_roll = TRUE)
```

`spells` *Dungeons and Dragons Spell Information*

Description

Spells in Dungeons and Dragons fit within several categories and their effects are well-documented. This table summarizes all of that information into a long format dataframe for easy navigation. Unless otherwise noted, all spell querying functions in ‘dndR‘ use this table as their starting point.

Usage

```
spells
```

Format

Dataframe with 12 columns and 513 rows

spell_name Name of the spell

spell_source Source book(s) for that spell with page numbers

pc_class Player Character (PC) class(es) that have access to this spell. If multiple classes, each is separated by commas. If a class has a colon and another word next to it (e.g., "cleric: grave") that indicates that only a specific sub-class of that class has access to the spell

spell_level Either "cantrip" or the level of spell slot required to cast the spell

spell_school School of the spell (e.g., "necromancy", "divination", etc.)

ritual_cast Whether the spell can be cast as a ritual expressed as a logical

casting_time Time required to cast the spell. Expressed as either the phase of a turn in which the spell can be cast (e.g., "1 action", "bonus action", etc.) or the actual in-game time required

range Range at which the spell can be cast

components Whether the spell has verbal ("V"), somatic ("S"), and/or material ("M") components required for casting. If material components are required they are described parenthetically

duration How long the spell lasts once cast

description Full description of the spell. Spells that require the player or Dungeon Master (DM) to roll on a table for an effect have those tables excluded for brevity. Similarly, spells that summon creatures have those creatures’ statistics excluded.

higher_levels Some spells can be cast using a higher level spell slot for an increased effect. Similarly, damage-dealing cantrips tend to deal more damage as PCs gain levels. This text describes how a spell’s effects change with higher spell slot levels or PC levels or is NA for spells that remain constant

Source

Mearls, M. and Crawford, J. Dungeons & Dragons Player's Handbook (Fifth Edition). Wizards of the Coast 2014

System Reference Document (Fifth Edition). Wizards of the Coast 2014

Elemental Evil. Wizards of the Coast 2015

Sword Coast Adventurer's Guide. Wizards of the Coast 2015

Xanathar's Guide to Everything. Wizards of the Coast 2017

Guildmasters' Guide to Ravnica. Wizards of the Coast 2018

Lost Laboratory of Kwalish. Wizards of the Coast 2018

Explorer's Guide to Wildemount. Wizards of the Coast 2020

Icewind Dale: Rime of the Frostmaiden. Wizards of the Coast 2020

Tasha's Cauldron of Everything. Wizards of the Coast 2020

Fizban's Treasury of Dragons. Wizards of the Coast 2021

From Cyan Depths. Wizards of the Coast 2021

Strixhaven: A Curriculum of Chaos. Wizards of the Coast 2021

A Verdant Tomb. Wizards of the Coast 2021

Astral Adventurer's Guide. Wizards of the Coast 2022

`spell_list` *List Spells Based on Criteria*

Description

Query list of all Dungeons & Dragons spells based on partial string matches between user inputs and the relevant column of the spell information data table. Currently supports users querying the spell list by spell name, which class lists allow the spell, spell's level, the school of magic the spell belongs in, whether or not the spell can be cast as a ritual, and the time it takes to cast the spell. All character arguments are case-insensitive (note that the ritual argument expects a logical). Any argument set to 'NULL' (the default) will not be used to include/exclude spells from the returned set of spells

Usage

```
spell_list(
  name = NULL,
  class = NULL,
  level = NULL,
  school = NULL,
  ritual = NULL,
  cast_time = NULL
)
```

Arguments

name	(character) text to look for in spell names
class	(character) character class(es) with the spell(s) on their list
level	(character) "cantrip" and/or the minimum required spell slot level
school	(character) school(s) of magic within which the spell belongs (e.g., 'evocation', 'necromancy', etc.)
ritual	(logical) whether the spell can be cast as a ritual
cast_time	(character) either the phase of a turn needed to cast the spell or the in-game time required (e.g., "reaction", "1 minute", etc.)

Value

(dataframe) 10 columns of information with one row per spell(s) that fit(s) the user-specified criteria. If no spells fit the criteria, returns a message to that effect instead of a data object

Examples

```
# Search for evocation spells with 'fire' in the name that a wizard can cast
spell_list(name = "fire", class = "wizard", school = "evocation")
```

---

spell_text *Retrieve Full Spell Description Text by Spell Name*

Description

Accepts user-provided Dungeons & Dragons spell name(s) and returns the full set of spell information and the complete description text. Unlike 'dndR::spell_list', this function requires an exact match between the user-provided spell name(s) and how they appear in the main spell data object. The argument in this function is not sensitive. This function's output differs from 'dndR::spell_list' only in that it returns the additional spell description text.

Usage

```
spell_text(name = NULL)
```

Arguments

name	(character) exact spell name(s) for which to gather description information

Value

(dataframe) 11 columns of spell information with one row per spell specified by the user. Returns 12 columns if the spell is a damage-dealing cantrip that deals increased damage as player level increases or if spell can be cast with a higher level spell slot (i.e., "upcast") for an increased effect.

Examples

```
spell_text(name = "chill touch")
```

---

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. Non-integer values are supported but results will be slightly affected
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