

# Package ‘capl’

October 12, 2022

**Title** Compute and Visualize CAPL-2 Scores and Interpretations

**Version** 1.42

**Maintainer** Joel Barnes <j@barnzilla.ca>

**Description** A toolkit for computing and visualizing CAPL-2  
(Canadian Assessment of Physical Literacy, Second Edition;  
<<https://www.capl-eclp.ca>>) scores and interpretations from raw data.

**License** GPL (>= 3)

**URL** <https://github.com/barnzilla/capl>

**BugReports** <https://github.com/barnzilla/capl/issues>

**Depends** R (>= 2.10)

**VignetteBuilder** knitr

**Encoding** UTF-8

**Language** en

**LazyData** true

**RoxygenNote** 7.1.1

**Imports** dplyr, ggplot2, lubridate, magrittr, readxl, stats, stringr,  
writexl

**Suggests** knitr, rmarkdown

**NeedsCompilation** no

**Author** Joel Barnes [aut, cre] (<<https://orcid.org/0000-0002-7621-9020>>),  
Michelle Guerrero [aut] (<<https://orcid.org/0000-0001-8169-5040>>)

**Repository** CRAN

**Date/Publication** 2022-04-08 02:52:39 UTC

## R topics documented:

capitalize_character . . . . .	3
capl_demo_data . . . . .	3
export_capl_data . . . . .	4

get_24_hour_clock . . . . .	4
get_adequacy_score . . . . .	5
get_binary_score . . . . .	6
get_camsa_score . . . . .	7
get_camsa_skill_time_score . . . . .	8
get_camsa_time_score . . . . .	9
get_capl . . . . .	10
get_capl_bar_plot . . . . .	11
get_capl_demo_data . . . . .	12
get_capl_domain_status . . . . .	13
get_capl_interpretation . . . . .	15
get_capl_score . . . . .	16
get_db_score . . . . .	17
get_fill_in_the_blanks_score . . . . .	18
get_intrinsic_motivation_score . . . . .	20
get_ku_score . . . . .	21
get_mc_score . . . . .	22
get_missing_capl_variables . . . . .	23
get_pacer_20m_laps . . . . .	26
get_pacer_score . . . . .	27
get_pa_competence_score . . . . .	27
get_pc_score . . . . .	29
get_pedometer_wear_time . . . . .	30
get_plank_score . . . . .	31
get_predilection_score . . . . .	31
get_self_report_pa_score . . . . .	32
get_step_average . . . . .	33
get_step_score . . . . .	35
import_capl_data . . . . .	35
rename_variable . . . . .	37
validate_age . . . . .	38
validate_character . . . . .	39
validate_domain_score . . . . .	40
validate_gender . . . . .	40
validate_integer . . . . .	41
validate_number . . . . .	42
validate_scale . . . . .	42
validate_steps . . . . .	43
<b>Index</b>	<b>44</b>

---

capitalize\_character *Capitalize a character vector.*

---

**Description**

This function capitalizes a character vector.

**Usage**

```
capitalize_character(x = NA)
```

**Arguments**

x                    A character vector.

**Details**

Other capl functions called by this function include: [validate\\_character\(\)](#).

**Value**

Returns a character vector (if valid) or NA (if not valid).

**Examples**

```
capitalize_character(c("beginning", "progressing", "achieving", "excelling"))  
# [1] "Beginning" "Progressing" "Achieving" "Excelling"
```

---

capl\_demo\_data            *CAPL demo raw data.*

---

**Description**

A dataset containing CAPL-2 demo raw data.

**Usage**

```
capl_demo_data
```

**Format**

A data frame with 500 rows of data on 60 variables that are required to compute CAPL-2 scores and interpretations:

...

**Source**

<https://github.com/barnzilla/capl>

---

export_capl_data	<i>Export CAPL-2 data to an Excel workbook.</i>
------------------	---

---

**Description**

This function exports CAPL-2 data to an Excel workbook on a local computer.

**Usage**

```
export_capl_data(x = NULL, file_path = NA)
```

**Arguments**

x	A data frame.
file_path	A character vector representing the file path to a location on the user's local computer (e.g., "c:/users/user_name/desktop/file.xlsx") where x will be saved as an Excel workbook on the user's computer. The file path is not case-sensitive.

**Details**

Other capl functions called by this function include: [validate\\_character\(\)](#).

**Value**

No return value.

---

get_24_hour_clock	<i>Convert 12-hour clock values to 24-hour clock values.</i>
-------------------	--

---

**Description**

This function converts 12-hour clock values to 24-hour clock values.

**Usage**

```
get_24_hour_clock(x = NA)
```

**Arguments**

x	A character vector representing values in 12-hour clock format.
---	---

**Details**

Other cap1 functions called by this function include: [validate\\_character\(\)](#) and [validate\\_integer\(\)](#).

**Value**

Returns a 24-hour clock vector (if valid) or NA (if not valid).

**Examples**

```
get_24_hour_clock(c("5:00 am", "7:10PM", "9:37", NA, "21:13", "", 9, "6:17"))
```

```
# [1] "05:00" "19:10" "09:37" NA      "21:13" NA      NA      "06:17"
```

---

get\_adequacy\_score      *Compute an adequacy score.*

---

**Description**

This function computes an adequacy score (`adequacy_score`) for responses to items 2, 4 and 6 of the CSAPPA (Children's Self-Perceptions of Adequacy in and Predilection for Physical Activity; Hay, 1992) Questionnaire as they appear in [the CAPL-2 Questionnaire](#). This score is used to compute the motivation and confidence domain score (`mc_score`).

**Usage**

```
get_adequacy_score(csappa2 = NA, csappa4 = NA, csappa6 = NA)
```

**Arguments**

<code>csappa2</code>	A numeric (integer) vector representing a response to CSAPPA item 2 (valid values are integers between 1 and 4).
<code>csappa4</code>	A numeric (integer) vector representing a response to CSAPPA item 4 (valid values are integers between 1 and 4).
<code>csappa6</code>	A numeric (integer) vector representing a response to CSAPPA item 6 (valid values are integers between 1 and 4).

**Details**

Valid values (integers between 1 and 4) represent the following responses:

- 1 = REALLY TRUE for me for "some kids" statements
- 2 = SORT OF TRUE for me for "some kids" statements
- 3 = REALLY TRUE for me for "other kids" statements
- 4 = SORT OF TRUE for me for "other kids" statements

Other cap1 functions called by this function include: [validate\\_scale\(\)](#).

**Value**

Returns a numeric vector with values between 1.8 and 7.5 (if valid) or NA (if not valid).

**Examples**

```
get_adequacy_score(
  csappa2 = c(1:3, 0),
  csappa4 = c(4, 2, 1, "3"),
  csappa6 = c(4, 4, 2, 2)
)

# [1] 4.9 4.8 4.9 NA
```

---

get_binary_score	<i>Compute a binary score.</i>
------------------	--------------------------------

---

**Description**

This function computes a binary score (0 = incorrect answer, 1 = correct answer) for a response to a questionnaire item based on the value(s) set as answer(s) to the item.

**Usage**

```
get_binary_score(x, answer)
```

**Arguments**

x	A character or numeric vector representing a response to a questionnaire item.
answer	A character or numeric vector representing the correct answer(s) to the questionnaire item. The answer argument does not have to match x in case for a correct answer to be computed.

**Details**

This function is called by [get\\_fill\\_in\\_the\\_blanks\\_score\(\)](#).

**Value**

Returns 1 (if correct), 0 (if incorrect) or NA (if not valid).

**Examples**

```

get_binary_score(
  x = c(1:4, NA, ""),
  answer = 3
)

# [1] 0 0 1 0 NA 0

get_binary_score(
  x = c("20 minutes", "30 minutes", "60 minutes or 1 hour", "120 minutes or 2 hours"),
  answer = "60 minutes or 1 hour"
)

# [1] 0 0 1 0

get_binary_score(
  x = c(1:5, "Heart", "hello, world", NA),
  answer = c(3, "heart")
)

# [1] 0 0 1 0 0 1 0 NA

```

---

<code>get_camsa_score</code>	<i>Select the maximum CAMSA skill + time score.</i>
------------------------------	---

---

**Description**

This function selects the maximum CAMSA (Canadian Agility and Movement Skill Assessment) skill + time score for two trials (`camsa_score`) and then divides by 2.8 so that the score is out of 10. This score is used to compute the physical literacy score (`pc_score`).

**Usage**

```
get_camsa_score(camsa_skill_time_score1 = NA, camsa_skill_time_score2 = NA)
```

**Arguments**

`camsa_skill_time_score1`  
A numeric (integer) vector representing the skill + time score from trial 1 (valid values are between 1 and 28).

`camsa_skill_time_score2`  
A numeric (integer) vector representing the skill + time score from trial 2 (valid values are between 1 and 28).

**Details**

Other `cap1` functions called by this function include: [validate\\_scale\(\)](#).

**Value**

Returns a numeric vector with values between 0 and 10 (if valid) or NA (if not valid).

**Examples**

```
get_camsa_score(  
  camsa_skill_time_score1 = c(1, 5, 10, 28, 29),  
  camsa_skill_time_score2 = c(5, 7, 12, NA, 27)  
)  
  
# [1] 5 7 12 NA NA
```

---

get\_camsa\_skill\_time\_score

*Compute the CAMSA skill + time score.*

---

**Description**

This function computes the CAMSA (Canadian Agility and Movement Skill Assessment) skill + time score (e.g., `camsa_skill_time_score1`) for a given trial. This score is used to compute the CAMSA score (`camsa_score`).

**Usage**

```
get_camsa_skill_time_score(camsa_skill_score = NA, camsa_time_score = NA)
```

**Arguments**

`camsa_skill_score`

A numeric (integer) vector representing the CAMSA skill score (valid values are between 0 and 14).

`camsa_time_score`

A numeric vector representing the CAMSA time score (valid values are between 1 and 14).

**Details**

Other cap1 functions called by this function include: [validate\\_scale\(\)](#).

**Value**

Returns a numeric (integer) vector with values between 1 and 28 (if valid) or NA (if not valid).

**Examples**

```
get_camsa_skill_time_score(  
  camsa_skill_score = c(0, 5, 10, 14, 15),  
  camsa_time_score = c(1, 10, 12, 15, 30)  
)  
  
# [1] 1 15 22 NA NA
```

---

```
get_camsa_time_score  Compute the CAMSA time score.
```

---

**Description**

This function computes the CAMSA (Canadian Agility and Movement Skill Assessment) time score based on the time taken (in seconds) to complete a trial.

**Usage**

```
get_camsa_time_score(camsa_time = NA)
```

**Arguments**

`camsa_time` A numeric vector representing the time taken (in seconds) to complete a CAMSA trial (valid values are > 0).

**Details**

Other cap1 functions called by this function include: [validate\\_number\(\)](#).

**Value**

Returns a numeric vector with values between 1 and 14 (if valid) or NA (if not valid).

**Examples**

```
get_camsa_time_score(c(14, 12, 30, 25, 0))  
  
# [1] 13 14 1 4 NA
```

---

`get_capl`*Compute all CAPL-2 scores and interpretations at once.*

---

### Description

This function is the main function in the `capl` package. It is a wrapper function that calls all other `capl` functions to compute all CAPL-2 scores and interpretations from raw data at once. If required CAPL-2 variables are missing, the function will create the variables and set values for these variables to NA so the function can proceed.

### Usage

```
get_capl(raw_data = NULL, sort = "asis", version = 2)
```

### Arguments

<code>raw_data</code>	A data frame of raw CAPL-2 data.
<code>sort</code>	An optional character vector representing how the variables in the returned data frame are to be sorted (valid values are "asis", "abc" and "zyx"; valid values are not case-sensitive). This argument is set to "asis" by default.
<code>version</code>	An optional numeric (integer) vector representing the version of CAPL. This argument is set to 2 by default. If set to 1, <a href="#">get_fill_in_the_blanks_score()</a> will ignore the <code>when_cooling_down</code> parameter and re-weight the score so that it's out of six.

### Details

Other `capl` functions called by this function include: [get\\_missing\\_capl\\_variables\(\)](#), [get\\_pacer\\_20m\\_laps\(\)](#), [get\\_pacer\\_score\(\)](#), [get\\_capl\\_interpretation\(\)](#), [get\\_plank\\_score\(\)](#), [get\\_camsa\\_time\\_score\(\)](#), [get\\_camsa\\_skill\\_time\\_score\(\)](#), [get\\_camsa\\_score\(\)](#), [get\\_pc\\_score\(\)](#), [get\\_capl\\_domain\\_status\(\)](#), [get\\_pedometer\\_wear\\_time\(\)](#), [validate\\_steps\(\)](#), [get\\_step\\_average\(\)](#), [get\\_step\\_score\(\)](#), [get\\_self\\_report\\_pa\\_score\(\)](#), [get\\_db\\_score\(\)](#), [get\\_predilection\\_score\(\)](#), [get\\_adequacy\\_score\(\)](#), [get\\_intrinsic\\_motivation\\_score\(\)](#), [get\\_pa\\_competence\\_score\(\)](#), [get\\_mc\\_score\(\)](#), [get\\_binary\\_score\(\)](#), [get\\_fill\\_in\\_the\\_blanks\\_score\(\)](#), [get\\_ku\\_score\(\)](#) and [get\\_capl\\_score\(\)](#)

### Value

Returns a merged data frame of raw data and CAPL-2 scores and interpretations.

### Examples

```
get_capl(raw_data)
```

---

get_capl_bar_plot	<i>Render a bar plot for a given CAPL-2 domain score, grouped by CAPL-2 interpretative categories.</i>
-------------------	--

---

### Description

This function renders a bar plot for a given CAPL-2 domain score, grouped by CAPL-2 interpretative categories.

### Usage

```
get_capl_bar_plot(  
  score = NA,  
  interpretation = NA,  
  x_label = "Interpretation",  
  y_label = "Score",  
  colors = c("#333376", "#00a79d", "#f26522", "#a6ce39")  
)
```

### Arguments

score	A numeric vector.
interpretation	A character vector representing CAPL-2 interpretative categories ("beginning", "progressing", "achieving", "excelling").
x_label	An optional character vector representing the x-axis label. This argument is set to "Interpretation" by default.
y_label	An optional character vector representing the y-axis label. This argument is set to "Score" by default.
colors	An optional character vector representing the color palette for the bars. This argument is set to CAPL-2 branding colors by default (c("#333376", "#00a79d", "#f26522", "#a6ce39", "#747474")).

### Details

Other capl functions called by this function include: [validate\\_character\(\)](#), [validate\\_number\(\)](#) and [capitalize\\_character\(\)](#).

### Value

Renders a ggplot2 bar plot (if valid).

### Examples

```
capl_results <- get_capl_demo_data(n = 25)  
  
get_capl_bar_plot(  
  score = capl_results$capl_score,
```

```

interpretation = capl_results$capl_interpretation,
x_label = "Overall physical literacy interpretation",
y_label = "Overall physical literacy score",
)

```

---

```

get_capl_demo_data      Generate CAPL-2 demo (fake) raw data.

```

---

## Description

This function generates a data frame of CAPL-2 demo (fake) raw data containing the 60 required variables that the capl package needs to compute scores and interpretations.

## Usage

```
get_capl_demo_data(n = 500)
```

## Arguments

**n** A numeric (integer) vector representing the number of rows of data to generate. By default, n is set to 500.

## Value

Returns a data frame containing the 60 required variables that the capl package needs to compute scores and interpretations.

## Examples

```

capl_demo_data <- get_capl_demo_data(10000)

str(capl_demo_data)

# 'data.frame': 10000 obs. of 60 variables:
# $ age                : int  9 10 8 8 11 9 12 NA 10 7 ...
# $ gender              : chr  "Girl" "Boy" "Boy" "Girl" ...
# $ pacer_lap_distance  : num  20 15 20 20 15 15 15 20 15 20 ...
# $ pacer_laps          : int  5 112 150 46 51 82 43 189 55 91 ...
# $ plank_time          : int  238 66 95 173 299 172 169 33 277 152 ...
# $ camsa_skill_score1  : int  9 3 7 NA 8 14 13 14 11 11 ...
# $ camsa_time1         : int  17 33 26 22 31 28 NA 24 12 11 ...
# $ camsa_skill_score2  : int  12 11 12 9 NA 9 7 10 14 11 ...
# $ camsa_time2         : int  15 13 15 20 12 15 29 12 12 18 ...
# $ steps1              : int  29663 30231 3157 5751 23362 28283 ...
# $ time_on1            : chr  "05:00" "5:13am" "07:00" "8:00am" ...
# $ time_off1           : chr  "11:57pm" "10:57 pm" "10:57 pm" "11:57pm" ...
# $ non_wear_time1     : int  38 47 38 40 36 32 36 82 25 51 ...
# $ steps2              : int  29703 9142 5424 23763 3645 28625 3019 ...
# $ time_on2            : chr  "07:00" "07:48am" "6:07" "06:00" ...

```

```

# $ time_off2           : chr "22:00" "21:00" "8:17pm" "10:57 pm" ...
# $ non_wear_time2     : int 5 34 41 60 84 18 19 47 66 55 ...
# $ steps3             : int 20380 10987 5885 13518 14385 30680 14120 ...
# $ time_on3           : chr "07:00" "06:00" "6:07" "8:00am" ...
# $ time_off3          : chr "11:13pm" "11:57pm" "21:00" "08:30pm" ...
# $ non_wear_time3     : int 54 70 16 36 72 16 89 86 26 81 ...
# $ steps4             : int 13224 20817 19640 2326 16605 25783 23078 ...
# $ time_on4           : chr "07:48am" "5:13am" "5:13am" "6:07" ...
# $ time_off4          : chr "11:13pm" NA "22:00" "23:00" ...
# $ non_wear_time4     : int 2 48 61 NA 81 81 2 30 35 14 ...
# $ steps5             : int 28408 8845 5802 6966 24499 18561 13771 ...
# $ time_on5           : chr "5:13am" NA "06:00" "6:07" ...
# $ time_off5          : chr "11:13pm" NA "11:57pm" "11:13pm" ...
# $ non_wear_time5     : int 75 10 70 45 77 75 90 61 17 72 ...
# $ steps6             : int 9581 18237 6377 3282 16898 15649 19890 ...
# $ time_on6           : chr "6:13" "6:07" "07:00" "8:00am" ...
# $ time_off6          : chr "11:57pm" "21:00" "10:57 pm" "8:17pm" ...
# $ non_wear_time6     : int 13 14 37 28 14 86 89 19 78 40 ...
# $ steps7             : int 8205 15351 16948 19442 4026 10830 4644 ...
# $ time_on7           : chr "05:00" NA "07:48am" "6:07" ...
# $ time_off7          : chr NA "22:00" "08:30pm" "08:30pm" ...
# $ non_wear_time7     : int 84 40 42 34 13 58 67 86 64 46 ...
# $ self_report_pa     : int 4 NA NA 7 1 1 6 7 6 6 ...
# $ csappa1            : int 2 1 1 1 2 1 4 3 3 3 ...
# $ csappa2            : int 3 3 1 4 4 2 3 1 4 4 ...
# $ csappa3            : int 1 2 4 1 2 4 1 4 4 1 ...
# $ csappa4            : int 4 1 3 4 2 3 1 2 2 4 ...
# $ csappa5            : int 2 4 2 2 4 1 1 1 3 1 ...
# $ csappa6            : int 2 2 2 3 4 3 2 3 1 1 ...
# $ why_active1        : int 5 2 5 5 2 5 1 1 5 1 ...
# $ why_active2        : int 4 5 2 4 3 1 5 1 4 1 ...
# $ why_active3        : int 2 1 4 3 1 2 1 5 3 3 ...
# $ feelings_about_pa1 : int 4 1 5 3 4 4 4 5 4 5 ...
# $ feelings_about_pa2 : int 5 3 4 4 1 2 5 2 1 3 ...
# $ feelings_about_pa3 : int 3 4 3 5 1 1 4 2 1 4 ...
# $ pa_guideline       : int 1 3 3 1 4 1 1 4 4 2 ...
# $ crf_means: int 2 3 2 3 4 1 3 4 1 3 ...
# $ ms_means          : int 1 1 4 2 4 4 2 1 1 3 ...
# $ sports_skill       : int 3 1 1 4 1 3 1 1 3 2 ...
# $ pa_is              : int 10 1 9 5 7 7 8 3 7 10 ...
# $ pa_is_also         : int 7 1 7 9 1 6 3 4 3 7 ...
# $ improve            : int 3 3 3 3 3 3 10 3 3 3 ...
# $ increase           : int 8 8 10 4 8 8 8 9 8 8 ...
# $ when_cooling_down  : int 5 2 2 2 2 2 4 2 3 7 ...
# $ heart_rate         : int 4 9 7 4 4 4 4 4 5 7 ...

```

---

```
get_capl_domain_status
```

*Compute the status of a CAPL domain.*

---

**Description**

This function computes the status ("complete", "missing interpretation", "missing protocol" or "incomplete") of a CAPL domain (e.g., `pc_status`, `db_status`, `mc_status`, `ku_status`, `capl_status`).

**Usage**

```
get_capl_domain_status(x = NULL, domain = NA)
```

**Arguments**

<code>x</code>	A data frame that includes the required variables for a given domain (see Details).
<code>domain</code>	A character vector representing one of the CAPL-2 domains (valid values are "pc", "db", "mc", "ku" and "capl")

**Details**

If the domain argument is set to "pc", the following variables must be included in the `x` argument:

- `pc_score`
- `pc_interpretation`
- `pacer_score`
- `plank_score`
- `camsa_score`

If the domain argument is set to "db", the following variables must be included the `x` argument:

- `db_score`
- `db_interpretation`
- `step_score`
- `self_report_pa_score`

If the domain argument is set to "mc", the following variables must be included the `x` argument:

- `mc_score`
- `mc_interpretation`
- `predilection_score`
- `adequacy_score`
- `intrinsic_motivation_score`
- `pa_competence_score`

If the domain argument is set to "ku", the following variables must be included the `x` argument:

- `ku_score`
- `ku_interpretation`
- `pa_guideline_score`

- crf\_means\_score
- ms\_means\_score
- sports\_skill\_score
- fill\_in\_the\_blanks\_score

If the domain argument is set to "capl", the following variables must be included the x argument:

- capl\_score
- capl\_interpretation
- pc\_score
- db\_score
- mc\_score
- ku\_score
- capl\_score

Other capl functions called by this function include: [validate\\_character\(\)](#) and [validate\\_number\(\)](#).

### Value

Returns a character vector with a value of "complete", "missing interpretation", "missing protocol" or "incomplete".

### Examples

```
capl_demo_data <- get_capl_demo_data(3)

capl_results <- get_capl(capl_demo_data)

get_capl_domain_status(capl_results, "pc")

# [1] "complete"           "incomplete"         "missing interpretation"
```

---

get\_capl\_interpretation

*Compute a CAPL-2 interpretation for a given CAPL-2 protocol or domain score.*

---

### Description

This function computes an age- and gender-specific CAPL-2 interpretation for a given CAPL-2 protocol or domain score (e.g., pc\_interpretation).

### Usage

```
get_capl_interpretation(age = NA, gender = NA, score = NA, protocol = NA)
```

**Arguments**

age	A numeric vector (valid values are between 8 and 12).
gender	A character vector (valid values currently include "girl", "g", "female", "f", "boy", "b", "male", "m").
score	A numeric vector. If the protocol argument is set to "pacer" or "steps", this argument must contain integers.
protocol	A character vector representing a CAPL protocol (valid values include "pacer", "plank", "camsa", "pc", "steps", "self_report_pa", "db", "mc", "ku", "capl"; valid values are not case-sensitive).

**Details**

Other capl functions called by this function include: [validate\\_age\(\)](#), [validate\\_gender\(\)](#), [validate\\_character\(\)](#), [validate\\_number\(\)](#) and [validate\\_scale\(\)](#). This function will check whether a score for a given protocol is within a valid range; if not, NA will be returned.

**Value**

Returns a character vector with values of "beginning", "progressing", "achieving" or "excelling" (if valid) or NA (if not valid).

**Examples**

```
get_capl_interpretation(
  age = 7:13,
  gender = c("g", "g", "b", "Boy", "m", "f", "Female"),
  score = c(50, 25, 100, 5, 150, 23, 78),
  protocol = "pacer"
)

# [1] NA          "achieving"  "excelling"  "beginning"  "excelling"  "progressing"
# [7] NA
```

---

get\_capl\_score      *Compute an overall physical literacy score.*

---

**Description**

This function computes an overall physical literacy score (capl\_score) based on the physical competence (pc\_score), daily behaviour (db\_score), motivation and confidence (mc\_score), and knowledge and understanding (ku\_score) domain scores. If one of the scores is missing or invalid, a weighted score will be computed from the other three scores.

**Usage**

```
get_capl_score(pc_score = NA, db_score = NA, mc_score = NA, ku_score = NA)
```

**Arguments**

pc_score	A numeric vector (valid values are between 0 and 30).
db_score	A numeric (integer) vector (valid values are between 0 and 30).
mc_score	A numeric vector (valid values are between 0 and 30).
ku_score	A numeric vector (valid values are between 0 and 10).

**Details**

Other cap1 functions called by this function include: [validate\\_number\(\)](#), [validate\\_integer\(\)](#) and [validate\\_domain\\_score\(\)](#).

**Value**

Returns a numeric vector with values between 0 and 100 (if valid) or NA (if not valid).

**Examples**

```
get_cap1_score(
  pc_score = c(20, 15, 12, 5, 31),
  db_score = c(20, 15, 6, 4.1, 25),
  mc_score = c(20, 20, 19, 15.4, 25),
  ku_score = c(11, 4, 5, 7.8, 10)
)

# [1] 66.66667 54.00000 42.00000 40.28571 85.71429
```

---

get_db_score	<i>Compute a daily behaviour domain score.</i>
--------------	--

---

**Description**

This function computes a daily behaviour domain score (db\_score) based on the step and self-reported physical activity scores. This score is used to compute the overall physical literacy score (cap1\_score).

**Usage**

```
get_db_score(step_score = NA, self_report_pa_score = NA)
```

**Arguments**

step_score	A numeric (integer) vector representing the pedometer steps score (valid values are integers between 0 and 25).
self_report_pa_score	A numeric (integer) vector representing the self-reported physical activity score (valid values are integers between 0 and 5).

**Details**

Other capl functions called by this function include: [validate\\_scale\(\)](#).

**Value**

Returns a numeric (integer) vector with values between 0 and 30 (if valid) or NA (if not valid).

**Examples**

```
get_db_score(
  step_score = c(20, 6, 13, 5, NA, 4.5),
  self_report_pa_score = c(3, 2, 1, 4, 7, 3)
)

# [1] 23  8 14  9 NA NA
```

---

```
get_fill_in_the_blanks_score
```

*Compute a fill in the blanks score.*

---

**Description**

This function computes a score (`fill_in_the_blanks_score`) for responses to the fill in the blanks items (story about Sally) in [the CAPL-2 Questionnaire](#). This score is used to compute the knowledge and understanding domain score (`ku_score`).

**Usage**

```
get_fill_in_the_blanks_score(
  pa_is = NA,
  pa_is_also = NA,
  improve = NA,
  increase = NA,
  when_cooling_down = NA,
  heart_rate = NA,
  version = 2
)
```

**Arguments**

<code>pa_is</code>	A vector representing a response to the first fill in the blank item (correct answers are 1, 7, "Fun" or "Good").
<code>pa_is_also</code>	A vector representing a response to the second fill in the blank item (correct answers are 1, 7, "Fun" or "Good").
<code>improve</code>	A vector representing a response to the third fill in the blank item (correct answers are 3 or "Endurance").

increase	A vector representing a response to the fourth fill in the blank item (correct answers are 8 or "Strength").
when_cooling_down	A vector representing a response to the fifth fill in the blank item (correct answers are 2 or "Stretches").
heart_rate	A vector representing a response to the sixth fill in the blank item (correct answers are 4 or "Pulse").
version	An optional numeric (integer) vector representing the version of CAPL. This argument is set to 2 by default. If set to 1, the when_cooling_down parameter will be ignored and the score re-weighted so that it's out of six.

### Details

The following integers represent the responses for the items/arguments in this function:

- 1 = Fun
- 2 = Stretches
- 3 = Endurance
- 4 = Pulse
- 5 = Breathing
- 6 = Flexibility
- 7 = Good
- 8 = Strength
- 9 = Bad
- 10 = Sport

Other capl functions called by this function include: [get\\_binary\\_score\(\)](#).

### Value

Returns a numeric (integer) vector with values between 0 and 5 (if valid) or NA (if not valid).

### Examples

```
get_fill_in_the_blanks_score(
  pa_is = c(2, 3, "fun", 9),
  pa_is_also = c(2, 5, "Fun", 9),
  improve = c(1, 3, 10, "Endurance"),
  increase = c(2, 3.5, "strength", "strength"),
  when_cooling_down = c("stretches", 9, 2, ""),
  heart_rate = c(3, 9, 4, "pulse")
)

# [1] 0 1 3 1
```

---

`get_intrinsic_motivation_score`*Compute an intrinsic motivation score.*

---

## Description

This function computes an intrinsic motivation score (`intrinsic_motivation_score`) for responses to items 1-3 of the Behavioral Regulation in Exercise Questionnaire (BREQ) as they appear in [the CAPL-2 Questionnaire](#). This score is used to compute the motivation and confidence domain score (`mc_score`).

## Usage

```
get_intrinsic_motivation_score(  
  why_active1 = NA,  
  why_active2 = NA,  
  why_active3 = NA  
)
```

## Arguments

<code>why_active1</code>	A numeric (integer) vector representing a response to BREQ item 1 (valid values are integers between 1 and 5).
<code>why_active2</code>	a numeric (integer) vector representing a response to BREQ item 2 (valid values are integers between 1 and 5).
<code>why_active3</code>	a numeric (integer) vector representing a response to BREQ item 3 (valid values are integers between 1 and 5).

## Details

Other cap1 functions called by this function include: [validate\\_scale\(\)](#).

Valid values (integers between 1 and 5) represent the following responses:

- 1 = Not true for me
- 2 = Not really true for me
- 3 = Sometimes true for me
- 4 = Often true for me
- 5 = Very true for me

## Value

Returns a numeric vector with values between 1.5 and 7.5 (if valid) or NA (if not valid).

### Examples

```
get_intrinsic_motivation_score(  
  why_active1 = c(4, 3, 6, 5, "2"),  
  why_active2 = c(1:5),  
  why_active3 = c(1, 5, 4, 3, 3)  
)  
  
# [1] 3 5 NA 6 5
```

---

get_ku_score	<i>Compute a knowledge and understanding domain score.</i>
--------------	--

---

### Description

This function computes a knowledge and understanding domain score (`ku_score`) based on the physical activity guideline (`pa_guideline_score`), cardiorespiratory fitness means (`crf_means_score`), muscular strength and endurance means (`ms_score`), sports skill (`sports_skill_score`) and fill in the blanks (`fill_in_the_blanks_score`) scores. If one of the scores is missing or invalid, a weighted domain score will be computed from the other four scores. This score is used to compute the overall physical literacy score (`capl_score`).

### Usage

```
get_ku_score(  
  pa_guideline_score = NA,  
  crf_means_score = NA,  
  ms_means_score = NA,  
  sports_skill_score = NA,  
  fill_in_the_blanks_score = NA  
)
```

### Arguments

`pa_guideline_score` A numeric (integer) vector (valid values are between 0 and 1).  
`crf_means_score` A numeric (integer) vector (valid values are between 0 and 1).  
`ms_means_score` A numeric (integer) vector (valid values are between 0 and 1).  
`sports_skill_score` A numeric (integer) vector (valid values are between 0 and 1).  
`fill_in_the_blanks_score` A numeric (integer) vector (valid values are between 0 and 6).

### Details

Other `capl` functions called by this function include: [validate\\_scale\(\)](#).

**Value**

Returns a numeric vector with values between 0 and 10 (if valid) or NA (if not valid).

**Examples**

```
get_ku_score(
  pa_guideline_score = c(1, 0, 1, 1, NA),
  crf_means_score = c(0, 1, "", 2, 1),
  ms_means_score = c(1, 1, 1, 0, 0),
  sports_skill_score = c(0, 0, 1, 0, 1),
  fill_in_the_blanks_score = c(5, 6, 3, 1, 2)
)

# [1] 7.000000 8.000000 6.666667 2.222222 4.444444
```

---

get\_mc\_score

*Compute a motivation and confidence domain score.*

---

**Description**

This function computes a motivation and confidence domain score (`mc_score`) based on the predilection (`predilection_score`), adequacy (`adequacy_score`), intrinsic motivation (`intrinsic_motivation_score`) and physical activity competence (`pa_competence_score`) scores. If one of the scores is missing or invalid, a weighted domain score will be computed from the other three scores. This score is used to compute the overall physical literacy score (`capl_score`).

**Usage**

```
get_mc_score(
  predilection_score = NA,
  adequacy_score = NA,
  intrinsic_motivation_score = NA,
  pa_competence_score = NA
)
```

**Arguments**

`predilection_score`

A numeric vector (valid values are between 1.8 and 7.5).

`adequacy_score` A numeric vector (valid values are between 1.8 and 7.5).

`intrinsic_motivation_score`

A numericvector (valid values are between 1.5 and 7.5).

`pa_competence_score`

A numeric vector (valid values are between 1.5 and 7.5).

**Details**

Other capl functions called by this function include: [validate\\_number\(\)](#).

**Value**

Returns a numeric vector with values between 0 and 30 (if valid) or NA (if not valid).

**Examples**

```
get_mc_score(  
  predilection_score = c(7, 7.5, 5, 8, 4),  
  adequacy_score = c(NA, 5, 3, 1, 4),  
  intrinsic_motivation_score = c(5, 7.5, 4, 2, 3.5),  
  pa_competence_score = c(6, 3, 6, 7, 2)  
)  
  
# [1] 24.0 23.0 18.0 NA 13.5
```

---

get\_missing\_capl\_variables

*Add required CAPL-2 variables to a data frame of raw data if they are missing.*

---

**Description**

This function adds required CAPL-2 variables (see Details for a full list) to a data frame of raw data if they are missing. When missing variables are added, the values for a given missing variable are set to NA. This function is called within [get\\_capl\(\)](#) so that CAPL-2 score and interpretation computations will run without errors in the presence of missing variables.

**Usage**

```
get_missing_capl_variables(raw_data = NULL)
```

**Arguments**

raw\_data            a data frame of raw CAPL-2 data.

**Details**

The required CAPL-2 variables include:

- age
- gender
- pacer\_lap\_distance
- pacer\_laps

- plank\_time
- camsa\_skill\_score1
- camsa\_time1
- camsa\_skill\_score2
- camsa\_time2
- steps1
- time\_on1
- time\_off1
- non\_wear\_time1
- steps2
- time\_on2
- time\_off2
- non\_wear\_time2
- steps3
- time\_on3
- time\_off3
- non\_wear\_time3
- steps4
- time\_on4
- time\_off4
- non\_wear\_time4
- steps5
- time\_on5
- time\_off5
- non\_wear\_time5
- steps6
- time\_on6
- time\_off6
- non\_wear\_time6
- steps7
- time\_on7
- time\_off7
- non\_wear\_time7
- self\_report\_pa
- csappa1
- csappa2
- csappa3

- csappa4
- csappa5
- csappa6
- why\_active1
- why\_active2
- why\_active3
- feelings\_about\_pa1
- feelings\_about\_pa2
- feelings\_about\_pa3
- pa\_guideline
- crf\_means
- ms\_means
- sports\_skill
- pa\_is
- pa\_is\_also
- improve
- increase
- when\_cooling\_down
- heart\_rate

Examining the structure (see [str\(\)](#)) of some CAPL-2 demo data (see [get\\_capl\\_demo\\_data\(\)](#)) provides additional information about these variables.

**Value**

returns a merged data frame of raw data and missing required CAPL-2 variables (values are set to NA).

**Examples**

```
raw_data <- get_missing_capl_variables(raw_data)
```

---

get\_pacer\_20m\_laps      *Convert PACER shuttle run laps to their equivalent in 20-metre laps.*

---

### Description

This function converts PACER (Progressive Aerobic Cardiovascular Endurance Run) shuttle run laps to their equivalent in 20-metre laps (pacer\_laps\_20m). If laps are already 20-metre laps, they are returned unless outside the valid range (1-229). This variable is used to compute the PACER score (pacer\_score).

### Usage

```
get_pacer_20m_laps(lap_distance = NA, laps_run = NA)
```

### Arguments

lap\_distance      A numeric (integer) vector (valid values are 15 or 20).  
laps\_run            A numeric (integer) vector (if lap\_distance = 15, valid values are integers between 1 and 298; if lap\_distance = 20, valid values are integers between 1 and 229).

### Details

Other cap1 functions called by this function include: [validate\\_integer\(\)](#) and [validate\\_scale\(\)](#).

### Value

Returns a numeric (integer) vector with values between 1 and 229 (if valid) or NA (if not valid).

### Examples

```
get_pacer_20m_laps(  
  lap_distance = c(15, 20, NA, "15", 20.5),  
  laps_run = rep(100, 5)  
)  
  
# [1] 77 100 NA 77 NA
```

---

get\_pacer\_score      *Compute a PACER score.*

---

### Description

This function computes a PACER (Progressive Aerobic Cardiovascular Endurance Run) score (pacer\_score) based on the number of PACER laps run at a 20-metre distance. This score is used to compute the physical competence domain score variable (pc\_score).

### Usage

```
get_pacer_score(pacer_laps_20m = NA)
```

### Arguments

pacer\_laps\_20m A numeric (integer) vector (valid values between 1 and 229).

### Details

Other cap1 functions called by this function include: [validate\\_scale\(\)](#) and [validate\\_integer\(\)](#).

### Value

Returns a numeric (integer) vector with values between 0 and 10 (if valid) or NA (if not valid).

### Examples

```
get_pacer_score(c(1, 6, 12, 18, NA, 46, 31, 45.1))  
  
# [1] 0 1 2 3 NA 9 6 NA
```

---

get\_pa\_competence\_score  
*Compute a physical activity competence score.*

---

### Description

This function computes a physical activity competence score (pa\_competence\_score) for responses to items 4-6 of the the Behavioral Regulation in Exercise Questionnaire (BREQ) as they appear in [the CAPL-2 Questionnaire](#). This score is used to compute the motivation and confidence domain score (mc\_score).

**Usage**

```
get_pa_competence_score(  
  feelings_about_pa1 = NA,  
  feelings_about_pa2 = NA,  
  feelings_about_pa3 = NA  
)
```

**Arguments**

`feelings_about_pa1`  
A numeric (integer) vector representing a response to BREQ item 4 (valid values are integers between 1 and 5).

`feelings_about_pa2`  
A numeric (integer) vector representing a response to BREQ item 5 (valid values are integers between 1 and 5).

`feelings_about_pa3`  
A numeric (integer) vector representing a response to BREQ item 6 (valid values are integers between 1 and 5).

**Details**

Other cap1 functions called by this function include: [validate\\_scale\(\)](#).

Valid elements (integers between 1 and 5) represent the following responses:

- 1 = Not true for me
- 2 = Not really true for me
- 3 = Sometimes true for me
- 4 = Often true for me
- 5 = Very true for me

**Value**

Returns a numeric vector with values between 1.5 and 7.5 (if valid) or NA (if not valid).

**Examples**

```
get_pa_competence_score(  
  feelings_about_pa1 = c(4, 3, 6, 5, "2"),  
  feelings_about_pa2 = c(1:5),  
  feelings_about_pa3 = c(1, 5, 4, 3, 3)  
)  
  
# [1] 3 5 NA 6 5
```

---

get\_pc\_score                      *Compute a physical competence domain score.*

---

### Description

This function computes a physical competence domain score (`pc_score`) based on the PACER (Progressive Aerobic Cardiovascular Endurance Run), plank and CAMSA (Canadian Agility and Movement Skill Assessment) scores. If one protocol score is missing or invalid, a weighted domain score will be computed from the other two protocol scores. This score is used to compute the physical competence domain score (`pc_score`).

### Usage

```
get_pc_score(pacer_score = NA, plank_score = NA, camsa_score = NA)
```

### Arguments

<code>pacer_score</code>	A numeric (integer) vector representing the PACER score (valid values are integers between 0 and 10).
<code>plank_score</code>	a numeric (integer) vector representing the plank score (valid values are integers between 0 and 10).
<code>camsa_score</code>	A numeric vector representing the best CAMSA skill + skill score divided by 2.8 (valid values are between 0 and 10).

### Details

Other cap1 functions called by this function include: [validate\\_scale\(\)](#).

### Value

Returns a numeric vector with values between 0 and 30 (if valid) or NA (if not valid).

### Examples

```
get_pc_score(  
  pacer_score = c(1, 5, 8, 10, NA),  
  plank_score = c(4, 5, 5, 6, 9),  
  camsa_score = c(-1, 0, 6, 4, 3)  
)  
  
# [1] 7.5 10.0 19.0 20.0 18.0
```

---

`get_pedometer_wear_time`*Compute pedometer wear time in decimal hours for a given day.*

---

### Description

This function computes pedometer wear time in decimal hours for a given day (e.g., `wear_time1`). This variable is used to compute the `step_average` variable and the step score (`step_score`).

### Usage

```
get_pedometer_wear_time(time_on = NA, time_off = NA, non_wear_time = NA)
```

### Arguments

<code>time_on</code>	A character vector representing the time (in 12- or 24-hour clock format) when the pedometer was first worn on a given day.
<code>time_off</code>	A character vector representing the time (in 12- or 24-hour clock format) when the pedometer was removed at the end of a given day.
<code>non_wear_time</code>	A numeric vector representing the total time (in minutes) the pedometer was not worn during waking hours on a given day.

### Details

Other cap1 functions called by this function include: [get\\_24\\_hour\\_clock\(\)](#) and [validate\\_number\(\)](#).

### Value

Returns a numeric vector with values  $\geq 0$  (if valid) or NA (if not valid).

### Examples

```
get_pedometer_wear_time(  
  time_on = c("6:23", "5:50 am", NA),  
  time_off = c("21:37", "9:17pm", ""),  
  c(60, 90, 0)  
)  
  
# [1] 14.23 13.95 NA
```

---

get\_plank\_score      *Compute a plank score.*

---

### Description

This function computes a plank score (plank\_score) based on the duration of time (in seconds) for which a plank is held. This score is used to compute the physical competence domain score (pc\_score).

### Usage

```
get_plank_score(plank_time = NA)
```

### Arguments

plank\_time      A numeric vector representing time (in seconds).

### Details

Other cap1 functions called by this function include: [validate\\_number\(\)](#).

### Value

Returns a numeric vector with values between 0 and 10 (if valid) or NA (if not valid).

### Examples

```
get_plank_score(c(120.5, 75.6, 40, 10.99, 90))  
# [1] 10 6 3 0 8
```

---

get\_predilection\_score  
*Compute a predilection score.*

---

### Description

This function computes a predilection score (predilection\_score) for responses to items 1, 3 and 5 of the CSAPPA (Children's Self-Perceptions of Adequacy in and Predilection for Physical Activity; Hay, 1992) Questionnaire as they appear in [the CAPL-2 Questionnaire](#). This score is used to compute the motivation and confidence domain score (mc\_score).

### Usage

```
get_predilection_score(csappa1 = NA, csappa3 = NA, csappa5 = NA)
```

**Arguments**

csappa1	A numeric (integer) vector representing a response to CSAPPA item 1 (valid values are integers between 1 and 4).
csappa3	A numeric (integer) vector representing a response to CSAPPA item 3 (valid values are integers between 1 and 4).
csappa5	A numeric (integer) vector representing a response to CSAPPA item 5 (valid values are integers between 1 and 4).

**Details**

Valid values (integers between 1 and 4) represent the following responses:

- 1 = REALLY TRUE for me for "some kids" statements
- 2 = SORT OF TRUE for me for "some kids" statements
- 3 = REALLY TRUE for me for "other kids" statements
- 4 = SORT OF TRUE for me for "other kids" statements

Other cap1 functions called by this function include: [validate\\_scale\(\)](#).

**Value**

Returns a numeric vector with values between 1.8 and 7.5 (if valid) or NA (if not valid).

**Examples**

```
get_predilection_score(
  csappa1 = c(1:3, 0),
  csappa3 = c(4, 2, 1, "3"),
  csappa5 = c(4, 4, 2, 2)
)

# [1] 4.2 4.2 4.3 NA
```

---

```
get_self_report_pa_score
```

*Compute a score for a response to the self-reported physical activity question.*

---

**Description**

This function computes a score (`self_report_pa_score`) for a response to "During the past week (7 days), on how many days were you physically active for a total of at least 60 minutes per day? (all the time you spent in activities that increased your heart rate and made you breathe hard)?" in [the CAPL-2 Questionnaire](#). This score is used to compute the daily behaviour domain score (`db_score`).

**Usage**

```
get_self_report_pa_score(x = NA)
```

**Arguments**

`x` A numeric (integer) vector representing the self-reported physical activity question (valid values are integers between 0 and 7).

**Details**

Other cap1 functions called by this function include: [validate\\_scale\(\)](#).

**Value**

Returns a numeric (integer) vector with values between 0 and 5 (if valid) or NA (if not valid).

**Examples**

```
get_self_report_pa_score(c(1, 8, 3, 4, 5, 2, 7))
# [1] 0 NA 2 3 4 1 5
```

---

get_step_average	<i>Compute average daily steps taken.</i>
------------------	---

---

**Description**

This function computes the daily arithmetic mean of a week of steps taken as measured by a pedometer (`step_average`). This variable is used to compute the step score (`step_score`).

**Usage**

```
get_step_average(raw_data = NULL)
```

**Arguments**

`raw_data` A data frame that includes seven days of pedometer steps and their corresponding on and off times. See Details for how these variables must be named.

**Details**

This function will throw an error unless the following variables are found in the `raw_data` argument:

- `steps1`
- `steps2`
- `steps3`

- steps4
- steps5
- steps6
- steps7
- time\_on1
- time\_on2
- time\_on3
- time\_on4
- time\_on5
- time\_on6
- time\_on7
- time\_off1
- time\_off2
- time\_off3
- time\_off4
- time\_off5
- time\_off6
- time\_off7

There must be at least three valid days for an arithmetic mean to be computed. If only three valid days, one of the step values from a valid day will be randomly sampled and used for the fourth valid day before computing the mean.

Other cap1 functions called by this function include: [validate\\_steps\(\)](#) and [get\\_pedometer\\_wear\\_time\(\)](#).

### **Value**

Returns a data frame with nine columns: steps1 (validated), steps2 (validated), steps3 (validated), steps4 (validated), steps5 (validated), steps6 (validated), steps7 (validated), valid\_days and step\_average. The steps are validated with the [validate\\_steps\(\)](#) function.

### **Examples**

```
cap1_demo_data <- get_cap1_demo_data(10)

get_step_average(cap1_demo_data)$step_average

# [1] 18365 12655 15493 12966 11396 13954 18456 13589 17543 11276
```

---

get_step_score	<i>Compute a step score.</i>
----------------	------------------------------

---

**Description**

This function computes a step score (`step_score`) based on the average daily steps taken as measured by a pedometer. This score is used to compute the daily behaviour domain score (`db_score`).

**Usage**

```
get_step_score(step_average = NA)
```

**Arguments**

`step_average` A numeric vector representing average daily steps taken. See [get\\_step\\_average\(\)](#).

**Details**

Other capl functions called by this function include: [validate\\_number\(\)](#).

**Value**

Returns a numeric (integer) vector with values between 0 and 25 (if valid) or NA (if not valid).

**Examples**

```
capl_demo_data <- get_capl_demo_data(10)
step_average <- get_step_average(capl_demo_data)$step_average
get_step_score(step_average)
# [1] 25 18 22 18 15 20 25 20 24 15
```

---

import_capl_data	<i>Import CAPL-2 data from an Excel workbook.</i>
------------------	---

---

**Description**

This function imports CAPL-2 data from an Excel workbook on a local computer.

**Usage**

```
import_capl_data(file_path = NA, sheet_name = NA)
```

**Arguments**

file_path	A character vector representing the file path to an Excel workbook on the user's local computer (e.g., "c:/users/user_name/desktop/file.xlsx"). The file path is not case-sensitive.
sheet_name	An optional character vector representing the sheet to import from the Excel workbook. If this argument is not set, the first sheet in the workbook will be imported.

**Details**

Other capl functions called by this function include: [validate\\_character\(\)](#).

**Value**

Returns a data frame if the Excel workbook sheet is successfully imported.

**Examples**

```
capl_demo_data <- import_capl_data(
  file_path = "c:/users/joel/desktop/capl_demo_data.xlsx",
  sheet_name = "Sheet1"
)

str(capl_demo_data)

# tibble [500 x 60] (S3: tbl_df/tbl/data.frame)
# $ age                : num [1:500] 8 9 9 8 12 10 12 10 12 9 ...
# $ gender              : chr [1:500] "Male" "Female" "Male" "f" ...
# $ pacer_lap_distance  : num [1:500] 15 20 20 15 20 15 15 15 15 NA ...
# $ pacer_laps          : num [1:500] 23 31 169 50 63 15 32 143 43 182 ...
# $ plank_time         : num [1:500] 274 282 9 228 252 110 21 185 6 41 ...
# $ camsa_skill_score1  : num [1:500] 14 5 6 13 2 9 4 11 5 11 ...
# $ camsa_time1         : num [1:500] 34 27 13 35 21 NA NA 16 20 14 ...
# $ camsa_skill_score2  : num [1:500] 14 5 13 11 14 14 0 4 0 4 ...
# $ camsa_time2        : num [1:500] 35 23 14 35 23 23 33 30 29 18 ...
# $ steps1             : num [1:500] 30627 27788 8457 8769 14169 ...
# $ time_on1           : chr [1:500] "5:13am" "6:13" "6:07" "6:13" ...
# $ time_off1          : chr [1:500] "22:00" NA "21:00" "22:00" ...
# $ non_wear_time1     : num [1:500] 25 31 33 25 83 67 20 10 49 64 ...
# $ steps2             : num [1:500] 14905 24750 30111 21077 15786 ...
# $ time_on2           : chr [1:500] "06:00" "5:13am" "6:13" "6:13" ...
# $ time_off2          : chr [1:500] "21:00" "23:00" "11:13pm" "23:00" ...
# $ non_wear_time2     : num [1:500] 20 82 4 55 1 53 65 47 82 79 ...
# $ steps3             : num [1:500] 21972 15827 14130 13132 18022 ...
# $ time_on3           : chr [1:500] "07:00" "05:00" "07:48am" NA ...
# $ time_off3          : chr [1:500] "11:57pm" NA "08:30pm" NA ...
# $ non_wear_time3     : num [1:500] 6 79 23 65 34 15 72 76 60 40 ...
# $ steps4             : num [1:500] 28084 27369 14315 9963 6993 ...
# $ time_on4           : chr [1:500] "05:00" "6:13" "6:07" NA ...
# $ time_off4          : chr [1:500] "08:30pm" "10:57 pm" "22:00" "11:13pm" ...
# $ non_wear_time4     : num [1:500] 32 38 74 20 75 22 84 59 42 22 ...
```

```

# $ steps5 : num [1:500] 14858 21112 16880 11707 20917 ...
# $ time_on5 : chr [1:500] "6:07" "6:13" "06:00" "05:00" ...
# $ time_off5 : chr [1:500] "11:57pm" "23:00" "8:17pm" "8:17pm" ...
# $ non_wear_time5 : num [1:500] 61 64 73 23 82 42 66 38 55 18 ...
# $ steps6 : num [1:500] 17705 5564 16459 12235 27766 ...
# $ time_on6 : chr [1:500] "06:00" "06:00" NA "6:07" ...
# $ time_off6 : chr [1:500] "21:00" NA "10:57 pm" "08:30pm" ...
# $ non_wear_time6 : num [1:500] 33 24 89 8 27 56 66 21 14 7 ...
# $ steps7 : num [1:500] 11067 13540 12106 18795 15039 ...
# $ time_on7 : chr [1:500] "6:07" "6:07" "8:00am" "06:00" ...
# $ time_off7 : chr [1:500] "08:30pm" "11:13pm" "8:17pm" "10:57 pm" ...
# $ non_wear_time7 : num [1:500] 8 72 4 38 9 32 49 36 34 43 ...
# $ self_report_pa : num [1:500] NA 2 2 4 3 5 NA 7 6 7 ...
# $ csappa1 : num [1:500] 1 2 4 2 2 2 3 2 2 3 ...
# $ csappa2 : num [1:500] 3 2 1 1 1 1 4 1 4 3 ...
# $ csappa3 : num [1:500] 2 3 2 1 NA 1 3 3 4 4 ...
# $ csappa4 : num [1:500] 4 1 1 3 4 4 4 4 4 1 ...
# $ csappa5 : num [1:500] 4 2 3 2 1 2 2 2 4 1 ...
# $ csappa6 : num [1:500] 3 4 1 4 2 2 2 3 4 4 ...
# $ why_active1 : num [1:500] 4 3 5 3 1 5 4 1 1 2 ...
# $ why_active2 : num [1:500] 5 3 4 2 5 3 5 NA 5 NA ...
# $ why_active3 : num [1:500] 3 3 1 4 2 3 4 4 5 3 ...
# $ feelings_about_pa1 : num [1:500] 4 3 2 2 1 1 3 4 4 2 ...
# $ feelings_about_pa2 : num [1:500] 5 2 2 3 4 2 4 4 2 5 ...
# $ feelings_about_pa3 : num [1:500] 2 5 2 5 3 2 2 1 3 5 ...
# $ pa_guideline : num [1:500] 2 3 4 1 2 4 3 2 2 2 ...
# $ crf_means : num [1:500] 1 4 4 2 2 1 2 1 4 1 ...
# $ ms_means : num [1:500] 3 2 1 2 3 1 1 2 4 2 ...
# $ sports_skill : num [1:500] 2 4 4 1 3 1 3 1 4 3 ...
# $ pa_is : num [1:500] 10 1 1 1 1 1 2 1 3 1 ...
# $ pa_is_also : num [1:500] 5 1 4 4 1 7 2 7 2 8 ...
# $ improve : num [1:500] 3 3 9 3 9 9 3 3 3 6 ...
# $ increase : num [1:500] 2 8 3 8 8 1 3 3 8 8 ...
# $ when_cooling_down : num [1:500] 4 2 4 2 2 2 2 5 2 2 ...
# $ heart_rate : num [1:500] 5 6 4 4 4 9 4 8 7 4 ...

```

---

rename\_variable

*Rename variables in a data frame.*

---

## Description

This function renames variables in a data frame.

## Usage

```
rename_variable(x = NULL, search = NA, replace = NA)
```

**Arguments**

x	A data frame.
search	A character vector representing the variable names to be renamed.
replace	A character vector representing the new names for those variables identified in the search argument.

**Details**

Other capl functions called by this function include: [validate\\_character\(\)](#).

**Value**

Returns a data frame with the renamed variables (if variables in the search argument are successfully found and renamed).

**Examples**

```
capl_demo_data <- get_capl_demo_data(n = 25)

str(capl_demo_data[, 1:2])

# 'data.frame': 25 obs. of  2 variables:
# $ age   : int  11 9 10 11 9 8 11 9 10 12 ...
# $ gender: chr  "Female" "Girl" "Girl" "f" ...

capl_demo_data <- rename_variable(
  x = capl_demo_data,
  search = c("age", "gender"),
  replace = c("hello", "world")
)

str(capl_demo_data[, 1:2])

# 'data.frame': 25 obs. of  2 variables:
# $ hello: int  11 9 10 11 9 8 11 9 10 12 ...
# $ world: chr  "Female" "Girl" "Girl" "f" ...
```

---

validate\_age

*Check whether an age is valid for CAPL-2.*

---

**Description**

This function checks whether an age is valid (numeric and between 8 and 12). CAPL-2 scores and interpretations are valid for children between the ages of 8 and 12 years.

**Usage**

```
validate_age(x)
```

**Arguments**

x                    A numeric vector.

**Details**

If x contains a decimal value that is otherwise valid (e.g., 8.5, 10.1), this function will return the `floor()` of the value.

Other cap1 functions called by this function include: `validate_number()`.

**Value**

Returns a numeric (integer) vector with a value between 8 and 12 (if valid) or NA (if not valid).

**Examples**

```
validate_age(c(7:13, "", NA, "12", 8.5))
# [1] NA  8  9 10 11 12 NA NA NA 12  8
```

---

validate\_character      *Check whether a vector is a character and not of length zero or "".*

---

**Description**

This function checks whether a vector is a character and not of length zero or "".

**Usage**

```
validate_character(x)
```

**Arguments**

x                    A vector.

**Value**

Returns a character vector (if valid) or NA (if not valid).

**Examples**

```
validate_character(c("beginning", "progressing", "achieving", "excelling", "", NA, 7))
# [1] "beginning" "progressing" "achieving" "excelling" NA NA
# [7] "7"
```

---

validate\_domain\_score *Check whether a CAPL-2 domain score is valid.*

---

**Description**

This function checks whether a CAPL-2 domain score is numeric and within a valid range.

**Usage**

```
validate_domain_score(x = NA, domain = NA)
```

**Arguments**

x	A vector representing a CAPL domain score.
domain	A character vector representing domains within CAPL (valid values are "pc", "db", "mc", "ku"; valid values are not case-sensitive).

**Details**

Other capl functions called by this function include: [validate\\_number\(\)](#) and [validate\\_integer\(\)](#).

**Value**

Returns a numeric vector (if valid) or NA (if not valid).

**Examples**

```
validate_domain_score(  
  x = c(34, 15, 10, 12.5, 25),  
  domain = "pc"  
)  
  
# [1] NA 15.0 10.0 12.5 25.0
```

---

validate\_gender *Check whether a vector can be classified as "girl" or "boy".*

---

**Description**

This function checks whether a vector can be classified as "girl" or "boy".

**Usage**

```
validate_gender(x)
```

**Arguments**

x                    A vector (see Examples for valid values).

**Value**

Returns a character vector with values of "girl" or "boy" (if valid) or NA (if not valid).

**Examples**

```
validate_gender(c("Girl", "GIRL", "g", "G", "Female", "f", "F", "", NA, 1))
# [1] "girl" "girl" "girl" "girl" "girl" "girl" "girl" NA    NA    "girl"
validate_gender(c("Boy", "BOY", "b", "B", "Male", "m", "M", "", NA, 0))
# [1] "boy" "boy" "boy" "boy" "boy" "boy" "boy" NA    NA    "boy"
```

---

validate\_integer            *Check whether a vector is an integer.*

---

**Description**

This function checks whether a vector is an integer.

**Usage**

```
validate_integer(x)
```

**Arguments**

x                    A vector.

**Value**

Returns a numeric (integer) vector (if valid) or NA (if not valid).

**Examples**

```
validate_integer(c(2, 6, 3.3, "", NA, "6", "hello, world"))
# [1] 2 6 NA NA NA 6 NA
```

---

validate_number	<i>Check whether a vector is numeric.</i>
-----------------	---

---

**Description**

This function checks whether a vector is numeric.

**Usage**

```
validate_number(x)
```

**Arguments**

x                    A vector.

**Value**

Returns a numeric vector (if valid) or NA (if not valid).

**Examples**

```
validate_number(c(1:5, "5", "", NA, "hello, world!"))
# [1] 1 2 3 4 5 5 NA NA NA
```

---

validate_scale	<i>Check whether a response to a given questionnaire item or scale is valid.</i>
----------------	--

---

**Description**

This function checks whether a vector for a given questionnaire item or scale is valid.

**Usage**

```
validate_scale(x, lower_bound = NA, upper_bound = NA)
```

**Arguments**

x                    A numeric (integer) vector representing the response to a questionnaire item (valid values are between the values set by the lower\_bound and upper\_bound arguments).

lower\_bound        A numeric (integer) vector representing the value below which x is invalid.

upper\_bound        A numeric (integer) vector representing the value above which x is invalid.

**Value**

Returns a numeric (integer) vector (if valid) or NA (if not valid).

**Examples**

```
validate_scale(
  x = c(0:10, NA, "7"),
  lower_bound = 1,
  upper_bound = 7
)

# [1] NA  1  2  3  4  5  6  7 NA NA NA NA  7
```

---

validate_steps	<i>Check whether daily steps as measured by a pedometer are valid.</i>
----------------	--

---

**Description**

This function checks whether daily steps as measured by a pedometer are valid. The variables from this function are used to compute `step_average` and the step score (`step_score`).

**Usage**

```
validate_steps(steps = NA, wear_time = NA)
```

**Arguments**

<code>steps</code>	A numeric (integer) vector representing the steps taken on a given day (valid values are between 1000 and 30000).
<code>wear_time</code>	A numeric vector representing the duration of time (in decimal hours) that a pedometer was worn on a given day (valid values are $\geq 10.0$ hours).

**Details**

Other `cap1` functions called by this function include: [validate\\_scale\(\)](#) and [validate\\_number\(\)](#).

**Value**

Returns the `steps` argument (if valid) or NA (if not valid).

**Examples**

```
validate_steps(
  steps = c(5400, 11001, 999, 31000, 8796),
  wear_time = c(10.1, 12.6, 10.2, 10.9, 9.5)
)

# [1] 5400 11001  NA  NA  NA
```

# Index

## \* datasets

- capl\_demo\_data, 3
- capitalize\_character, 3
- capitalize\_character(), 11
- capl\_demo\_data, 3
- export\_capl\_data, 4
- floor(), 39
- get\_24\_hour\_clock, 4
- get\_24\_hour\_clock(), 30
- get\_adequacy\_score, 5
- get\_adequacy\_score(), 10
- get\_binary\_score, 6
- get\_binary\_score(), 10, 19
- get\_camsa\_score, 7
- get\_camsa\_score(), 10
- get\_camsa\_skill\_time\_score, 8
- get\_camsa\_skill\_time\_score(), 10
- get\_camsa\_time\_score, 9
- get\_camsa\_time\_score(), 10
- get\_capl, 10
- get\_capl(), 23
- get\_capl\_bar\_plot, 11
- get\_capl\_demo\_data, 12
- get\_capl\_demo\_data(), 25
- get\_capl\_domain\_status, 13
- get\_capl\_domain\_status(), 10
- get\_capl\_interpretation, 15
- get\_capl\_interpretation(), 10
- get\_capl\_score, 16
- get\_capl\_score(), 10
- get\_db\_score, 17
- get\_db\_score(), 10
- get\_fill\_in\_the\_blanks\_score, 18
- get\_fill\_in\_the\_blanks\_score(), 6, 10
- get\_intrinsic\_motivation\_score, 20
- get\_intrinsic\_motivation\_score(), 10
- get\_ku\_score, 21
- get\_ku\_score(), 10
- get\_mc\_score, 22
- get\_mc\_score(), 10
- get\_missing\_capl\_variables, 23
- get\_missing\_capl\_variables(), 10
- get\_pa\_competence\_score, 27
- get\_pa\_competence\_score(), 10
- get\_pacer\_20m\_laps, 26
- get\_pacer\_20m\_laps(), 10
- get\_pacer\_score, 27
- get\_pacer\_score(), 10
- get\_pc\_score, 29
- get\_pc\_score(), 10
- get\_pedometer\_wear\_time, 30
- get\_pedometer\_wear\_time(), 10, 34
- get\_plank\_score, 31
- get\_plank\_score(), 10
- get\_predilection\_score, 31
- get\_predilection\_score(), 10
- get\_self\_report\_pa\_score, 32
- get\_self\_report\_pa\_score(), 10
- get\_step\_average, 33
- get\_step\_average(), 10, 35
- get\_step\_score, 35
- get\_step\_score(), 10
- import\_capl\_data, 35
- rename\_variable, 37
- str(), 25
- validate\_age, 38
- validate\_age(), 16
- validate\_character, 39
- validate\_character(), 3–5, 11, 15, 16, 36, 38
- validate\_domain\_score, 40
- validate\_domain\_score(), 17

validate\_gender, 40  
validate\_gender(), 16  
validate\_integer, 41  
validate\_integer(), 5, 17, 26, 27, 40  
validate\_number, 42  
validate\_number(), 9, 11, 15–17, 23, 30, 31,  
35, 39, 40, 43  
validate\_scale, 42  
validate\_scale(), 5, 7, 8, 16, 18, 20, 21,  
26–29, 32, 33, 43  
validate\_steps, 43  
validate\_steps(), 10, 34