Package ‘mnreadR’

January 18, 2024

Type Package

Title MNREAD Parameters Estimation and Curve Plotting

Version 2.1.7

Description Allows to analyze the reading data obtained with the MNREAD Acuity Chart, a continuous-text reading acuity chart for normal and low vision. Provides the necessary functions to plot the MNREAD curve and estimate automatically the four MNREAD parameters: Maximum Reading Speed, Critical Print Size, Reading Acuity and Reading Accessibility Index. Parameters can be estimated either with the standard method or with a nonlinear mixed-effects (NLME) modeling. See Calabrese et al. 2018 for more details <doi:10.1167/18.1.8>.

Depends R (>= 2.10), dplyr, tidyr, ggplot2

Imports stats, nlme, tibble

URL https://legge.psych.umn.edu/mnread-acuity-charts

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 7.2.1

NeedsCompilation no

Author Aurélie Calabrèse [aut, cre], J. Steve Mansfield [aut], Gordon E. Legge [aut]

Maintainer Aurélie Calabrèse <aurelie.calabrese@univ-amu.fr>

Repository CRAN

Date/Publication 2024-01-18 20:30:02 UTC

R topics documented:

   accIndex .......................................................... 2
curveParam_RS ..................................................... 4
Reading Accessibility Index (ACC) calculation

Description

This function calculates the Reading Accessibility Index, while applying suited rules for missing data.

Usage

accIndex(data, print_size, reading_time, errors, ... = NULL)

Arguments

data The name of your dataframe
print_size The variable that contains print size values for each sentence (print size uncorrected for viewing distance)
reading_time The variable that contains the reading time for each sentence
errors The variable that contains the number of errors for each sentence
... Optional grouping arguments

Value

The function returns a new dataframe with a variable called "ACC" that contains the Reading Accessibility Index estimate.
Notes

The Reading ACCessibility Index (ACC) is a new measure representing an individual’s access to text over the range of print sizes found in everyday life. Its calculation does not rely on curve fitting and gives a direct comparison with the performance of normally sighted individuals. The ACC calculation uses the print size values non corrected for non-standard viewing distance. For more details on the Reading Accessibility Index, see http://doi.org/10.1001/jamaophthalmol.2015.6097

Warning

To ensure that missing data are handled properly and that ACC calculation is correct, data need to be entered along certain rules:

- For the smallest print size that is presented but not read, right before the test is stopped: reading_time = NA, errors = 10
- For all the small sentences that are not presented because the test was stopped before them: reading_time = NA, errors = NA
- If a sentence is presented, and read, but the time was not recorded by the experimenter: reading_time = NA, errors = actual number of errors (cf. s5-regular in low vision data sample)
- If a large sentence was skipped to save time but would have been read well: reading_time = NA, errors = NA (cf. s1-regular in normal vision data sample)
- If a large sentence was skipped to save time because the subject cannot read large print: reading_time = NA, errors = 10 (cf. s7 in low vision data sample)

See Also

mnreadParam for all MNREAD parameters estimation
curveParam_RT for MRS and CPS estimation using values of reading time (instead of reading speed)
curveParam_RS for MRS and CPS estimation using values of reading speed (instead of reading time)
readingAcuity for Reading Acuity calculation

Examples

# inspect the structure of the dataframe
head(data_low_vision, 10)

#------

# restrict dataset to one MNREAD test only (subject s1, regular polarity)
data_s1 <- data_low_vision %>%
  filter (subject == "s1", polarity == "regular")

# run the reading accessibility index calculation
data_low_vision_ACC <- accIndex(data_s1, ps, rt, err)

# inspect the newly created dataframe
data_low_vision_ACC
# run the reading accessibility index calculation
# on the whole dataset grouped by subject and polarity
data_low_vision_ACC <- accIndex(data_low_vision, ps, rt, err,
subject, polarity)

# inspect the structure of the newly created dataframe
head(data_low_vision_ACC, 10)

---

**curveParam_RS**

*Standard estimation of Maximum Reading Speed (MRS) and Critical Print Size (CPS) using reading speed values.*

**Description**

This function estimates simultaneously:

- Maximum Reading Speed (MRS)
- Critical Print Size (CPS)

while performing print size correction for non-standard testing viewing distance.

**Usage**

`curveParam_RS(data, print_size, viewing_distance, reading_speed, ... = NULL)`

**Arguments**

- **data**
  - The name of your dataframe

- **print_size**
  - The variable that contains print size values for each sentence (print size uncorrected for viewing distance)

- **viewing_distance**
  - The variable that contains the viewing distance value used for testing

- **reading_speed**
  - The variable that contains the reading speed for each sentence

- **...**
  - Optional grouping arguments

**Value**

The function returns a new dataframe with two variables:

- "CPS" -> contains the Critical Print Size estimate (in logMAR)
- "MRS" -> contains the Maximum Reading Speed estimate (in words/min)
Notes

This function uses the original algorithm described in Legge (2007) to estimate Maximum Reading Speed (MRS) and Critical Print Size (CPS). This algorithm searches for a reading speed plateau in the data. A plateau is defined as a range of print sizes that supports reading speed at a significantly faster rate than the print sizes smaller or larger than the plateau range. Concretely, the plateau is determined as print sizes which reading speed is at least 1.96 SD faster than the other print sizes. The Maximum Reading Speed is estimated as the mean reading speed for print sizes included in the plateau. The Critical Print Size is defined as the smallest print size on the plateau.


To ensure proper estimation of the MRS and CPS, individual MNREAD curves should be plotted using mnreadCurve and inspected visually.

Warning

To run the function properly, one needs to make sure that the variables are of the class:

- `print_size` -> numeric
- `viewing_distance` -> integer
- `reading_speed` -> numeric

In cases where only 3 or less sentences were read during a test, the function won’t be able to estimate the MRS and CPS and will return NA values instead. The ACC should be used to estimate the MNREAD score in such cases where there are not enough data points to fit the MNREAD curve.

See Also

curveParam_RT for standard MRS and CPS estimation using values of reading time (instead of reading speed)
nlmeParam for MRS and CPS estimation using a nonlinear mixed-effect model (NLME)
mnreadParam for all MNREAD parameters estimation
readingAcuity for Reading Acuity calculation
accIndex for Reading Accessibility Index calculation

Examples

# inspect the structure of the dataframe
head(data_low_vision, 10)

# create the reading speed variable
data_low_vision <- data_low_vision %>%
  mutate(rs = (10 - replace(err, err > 10, 10)) / rt * 60)

#------

# restrict dataset to one MNREAD test only (subject s1, regular polarity)
```r
data_s1 <- data_low_vision %>%
  filter(subject == "s1", polarity == "regular")

# run the parameters estimation
data_low_vision_MRS_CPS <- curveParam_RS(data_s1, ps, vd, rs)

# inspect the newly created dataframe
data_low_vision_MRS_CPS

#------

# run the parameters estimation on the whole dataset grouped by subject and polarity
data_low_vision_MRS_CPS <- curveParam_RS(data_low_vision, ps, vd, rs, subject, polarity)

# inspect the structure of the newly created dataframe
head(data_low_vision_MRS_CPS, 10)
```

---

**curveParam_RT**

Standard estimation of Maximum Reading Speed (MRS) and Critical Print Size (CPS) using individual data of reading time and number of errors.

**Description**

This function estimates simultaneously:

- Maximum Reading Speed (MRS)
- Critical Print Size (CPS)

while performing print size correction for non-standard testing viewing distance.

**Usage**

```r
curveParam_RT(
  data, print_size, viewing_distance, reading_time, errors,
  ... = NULL
)
```

**Arguments**

- `data`:
  
  The name of your dataframe

- `print_size`:
  
  The variable that contains print size values for each sentence (print size uncorrected for viewing distance)
viewing_distance  The variable that contains the viewing distance value used for testing
reading_time    The variable that contains the reading time for each sentence
errors          The variable that contains the number of errors for each sentence
...             Optional grouping arguments

Value
The function returns a new dataframe with two variables:

- "CPS" -> contains the Critical Print Size estimate (in logMAR)
- "MRS" -> contains the Maximum Reading Speed estimate (in words/min)

Notes
This function uses the original algorithm described in Legge (2007) to estimate Maximum Reading Speed (MRS) and Critical Print Size (CPS). This algorithm searches for a reading speed plateau in the data. A plateau is defined as a range of print sizes that supports reading speed at a significantly faster rate than the print sizes smaller or larger than the plateau range. Concretely, the plateau is determined as print sizes which reading speed is at least 1.96 SD faster than the other print sizes. The Maximum Reading Speed is estimated as the mean reading speed for print sizes included in the plateau. The Critical Print Size is defined as the smallest print size on the plateau.


https://books.google.fr/books/about/Psychophysics_of_Reading_in_Normal_and_L.html?id=BGTHS8zANiUC&redir_esc=y

To ensure proper estimation of the MRS and CPS, individual MNREAD curves should be plotted using mnreadCurve and inspected visually.

Warning
For the function to run properly, one needs to make sure that the variables are of the class:

- print_size  -> numeric
- viewing_distance  -> integer
- reading_time  -> numeric
- errors  -> integer

In cases where only 3 or less sentences were read during a test, the function won’t be able to estimate the MRS and CPS and will return NA values instead. The ACC should be used to estimate the MNREAD score in such cases where there are not enough data points to fit the MNREAD curve.

To ensure proper parameters estimation, the data should be entered along certain rules:

- For the smallest print size that is presented but not read, right before the test is stopped:
  reading_time = NA, errors = 10
- For all the small sentences that are not presented because the test was stopped before them:
  reading_time = NA, errors = NA
• If a sentence is presented, and read, but the time was not recorded by the experimenter: **reading_time = NA, errors = actual number of errors** (cf. s5-regular in low vision data sample)

• If a large sentence was skipped to save time but would have been read well: **reading_time = NA, errors = NA** (cf. s1-regular in normal vision data sample)

• If a large sentence was skipped to save time because the subject cannot read large print: **reading_time = NA, errors = 10** (cf. s7 in low vision data sample)

**See Also**

- `curveParam_RS` for standard MRS and CPS estimation using values of reading speed (instead of reading time)
- `nlmeParam` for MRS and CPS estimation using nonlinear mixed-effect (NLME) modeling
- `mnreadParam` for all MNREAD parameters estimation (using standard calculation)
- `readingAcuity` for Reading Acuity calculation
- `accIndex` for Reading Accessibility Index calculation

**Examples**

```r
# inspect the structure of the dataframe
head(data_low_vision, 10)

#------

# restrict dataset to one MNREAD test only (subject s1, regular polarity)
data_s1 <- data_low_vision %>%
  filter (subject == "s1", polarity == "regular")

# run the parameters estimation
data_low_vision_MRS_CPS <- curveParam_RT(data_s1, ps, vd, rt, err)

# inspect the newly created dataframe
data_low_vision_MRS_CPS

#------

# run the parameters estimation on the whole dataset grouped by subject and polarity
data_low_vision_MRS_CPS <- curveParam_RT(data_low_vision, ps, vd, rt, err, subject, polarity)

# inspect the structure of the newly created dataframe
head(data_low_vision_MRS_CPS, 10)
```
**data_low_vision**  
MNREAD data collected in subjects with low vision.

### Description
A dataset containing raw MNREAD data for 12 subjects with low vision. 6 subjects were treated with treatment A while the other 6 were given treatment B. Each subject was tested twice on the MNREAD:

- once on the regular polarity of the test (black print on white background)
- once on the reverse polarity of the test (white print on black background)

### Usage
data_low_vision

### Format
A data frame with 437 rows and 7 variables, where each line stores data for one sentence:

- **subject**  subject ID code
- **polarity**  test polarity used (regular or reverse)
- **treatment**  treatment given to the subject (A or B)
- **vd**  viewing distance in cm
- **ps**  print size in logMAR, as written on the chart (print size uncorrected for viewing distance)
- **rt**  reading time in seconds
- **err**  number of errors ...

### Source
Data collected at the Minnesota Laboratory for Low-Vision Research (UMN)

---

**data_normal_vision**  
MNREAD data collected in subjects with normal vision.

### Description
A dataset containing raw MNREAD data for 18 young adults with normal vision. Each subject was tested twice:

- once on the regular polarity of the test (black print on white background)
- once on the reverse polarity of the test (white print on black background)
Usage

logMARcorrect(data, print_size, viewing_distance)

Arguments

data        The name of your dataframe
print_size  The variable that contains print size values (print size uncorrected for viewing distance)
viewing_distance  The variable that contains the viewing distance value used for testing

Value

The function returns the original dataframe with an added variable called "correct_ps" that contains corrected print size values (in logMAR).
Examples

# inspect the structure of the dataframe
head(data_low_vision, 10)

# run the correction
data_low_vision_new <- logMARcorrect(data_low_vision, ps, vd)

# inspect the structure of the newly created dataframe
head(data_low_vision_new, 10)


---

mnreadCurve  MNREAD curve plotting.

Description

This function plots individual MNREAD curves, while showing the estimated MNREAD parameters:

- Maximum Reading Speed (MRS)
- Critical Print Size (CPS)
- Reading Acuity (RA)

Usage

mnreadCurve(
  data,  The name of your dataframe
  print_size, The variable that contains print size values for each sentence (print size uncorrected for viewing distance)
  viewing_distance, The variable that contains the viewing distance value used for testing
  reading_time, The variable that contains the reading time for each sentence
  errors, The variable that contains the number of errors for each sentence
  ... = NULL
)

Arguments

data  The name of your dataframe
print_size  The variable that contains print size values for each sentence (print size uncorrected for viewing distance)
viewing_distance  The variable that contains the viewing distance value used for testing
reading_time  The variable that contains the reading time for each sentence
errors  The variable that contains the number of errors for each sentence
...  Optional grouping arguments
mnreadCurve

Value

The function returns a plot of reading speed (in words/min) as a function of print size (in logMAR). Reading Acuity is marked as a triangle, Maximum Reading Speed and Critical Print Size are shown with dashed lines. When using two grouping arguments, a colored diamond is added for clarification. Highlighted data points represent the range of print sizes included in the Reading Accessibility Index calculation.

Notes

This function can’t take more that two grouping arguments. The first grouping argument is used to draw sub-plots (using facet_wrap from ggplot2). The second grouping argument is color-coded.

This function performs print size correction for non-standard testing viewing distance before plotting the curve.

This function uses the original algorithm described in Legge (2007) to estimate Maximum Reading Speed (MRS) and Critical Print Size (CPS). For more details on the parameters estimation, see curveParam_RT.

Warning

For the function to run properly, one needs to make sure that the variables are of the class:

- print_size -> numeric
- viewing_distance -> integer
- reading_time -> numeric
- errors -> integer

In cases where only 3 or less sentences were read during a test, MRS and CPS cannot be estimated and won’t be displayed on the plot. In such cases, the Reading Accessibility Index (ACC) can be used to estimate the MNREAD score instead (cf. accIndex).

To ensure proper plotting, the data should be entered along certain rules:

- For the smallest print size that is presented but not read, right before the test is stopped: reading_time = NA, errors = 10
- For all the small sentences that are not presented because the test was stopped before them: reading_time = NA, errors = NA
- If a sentence is presented, and read, but the time was not recorded by the experimenter: reading_time = NA, errors = actual number of errors (cf. s5-regular in low vision data sample)
- If a large sentence was skipped to save time but would have been read well: reading_time = NA, errors = NA (cf. s1-regular in normal vision data sample)
- If a large sentence was skipped to save time because the subject cannot read large print: reading_time = NA, errors = 10 (cf. s7 in low vision data sample)

See Also

- curveParam_RT for standard estimation of MRS and CPS using values of reading time (instead of reading speed)
- readingAcuity for Reading Acuity calculation
Examples

```r
# inspect the structure of the dataframe
head(data_low_vision, 10)

#------

# restrict dataset to one MNREAD test only (subject s1, regular polarity)
data_s1_reg <- data_low_vision %>%
  filter (subject == "s1", polarity == "regular")

# plot the MNREAD curve
mnreadCurve(data_s1_reg, ps, vd, rt, err)

#------

# restrict dataset to one subject (s1) and plot the MNREAD curves using ONE GROUPING ARGUMENT (ie. polarity)
data_s1 <- data_low_vision %>%
  filter (subject == "s1")

# plot the MNREAD curve using ONE GROUPING ARGUMENT (ie. polarity)
mnreadCurve(data_s1, ps, vd, rt, err, polarity)

#------

# restrict dataset to two subject (s1 & s2) and plot the MNREAD curves using TWO GROUPING ARGUMENTS (ie. subject and polarity)
data_s2 <- data_low_vision %>%
  filter (subject == "s1" | subject == "s2")

  mnreadCurve(data_s2, ps, vd, rt, err, subject, polarity)

#------

# Once created, the MNREAD curve can be customized as needed using ggplot2,
# for ex., by adding the number of errors for each sentence on top of the curve

# plot the MNREAD curve
my.plot <- mnreadCurve(data_s1, ps, vd, rt, err, polarity)

# display my.plot
print(my.plot)

# calculate reading speed and perform print size correction
data_s1_new <- as.data.frame(
data_s1 %>%
  filter (err != "NA" & rt > 0) %>%
  mutate (errors10 = replace (err, err > 10, 10) ) %>%
  mutate (rs = 60 * (10 - errors10) / rt ) %>%
  mutate (correct_ps = ps + round(log10(40/(vd)), 2) )
)

# add the number of errors for each sentence
```
my.new.plot <- my.plot + geom_text(aes(x = correct_ps, y = rs + 5, label = errors10),
    alpha = 0.5,
    data = data_s1_new %>% filter (errors10 != 0) )

# display my.new.plot
print(my.new.plot)

# MNREAD curves can also be saved in a pdf file, with each page showing a different subject
# count the number of subjects to define the number of pages
num_pages = length(unique(data_s2$subject))

# create a pdf file
pdf ("MNREAD_curves.pdf", width = 10.5, height = 8, paper = "special", useDingbats = TRUE)

# wrap the plots over several pages
for (i in seq(num_pages)){
  p <- mnreadCurve(data_s2 %>% filter (subject == sort(unique(data_s2$subject)))[i]),
                ps, vd, rt, err, subject, polarity)
  print(p)
}

dev.off()

mnreadParam

Standard MNREAD parameters' estimation

Description
This function calculates simultaneously all four MNREAD parameters:

- Maximum Reading Speed (MRS)
- Critical Print Size (CPS)
- Reading Acuity (RA)
- Reading ACCessibility Index (ACC)

while performing print size correction for non-standard testing viewing distance.

Usage

mnreadParam(
  data,
  print_size,
Arguments

data The name of your dataframe
print_size The variable that contains print size values for each sentence (print size uncorrected for viewing distance)
viewing_distance The variable that contains the viewing distance value used for testing
reading_time The variable that contains the reading time for each sentence
errors The variable that contains the number of errors for each sentence
...
Optional grouping arguments

Value

The function returns a new dataframe with four variables:

- "RA" -> contains the Reading Acuity estimate (in logMAR)
- "CPS" -> contains the Critical Print Size estimate (in logMAR)
- "MRS" -> contains the Maximum Reading Speed estimate (in words/min)
- "ACC" -> contains the Reading Accessibility Index estimate

Notes

This function uses the original algorithm described in Legge (2007) to estimate Maximum Reading Speed (MRS) and Critical Print Size (CPS). This algorithm searches for a reading speed plateau in the data. A plateau is defined as a range of print sizes that supports reading speed at a significantly faster rate than the print sizes smaller or larger than the plateau range. Concretely, the plateau is determined as print sizes which reading speed is at least 1.96 SD faster than the other print sizes. The Maximum Reading Speed is estimated as the mean reading speed for print sizes included in the plateau. The Critical Print Size is defined as the smallest print size on the plateau.

For more details on the parameters estimation, see https://legge.psych.umn.edu/mnread-acuity-charts


To ensure proper estimation of the MRS and CPS, individual MNREAD curves should be plotted using mnreadCurve and inspected visually.
Warning

For the function to run properly, one needs to make sure that the variables are of the class:

- **print_size** -> numeric
- **viewing_distance** -> integer
- **reading_time** -> numeric
- **errors** -> integer

In cases where only 3 or less sentences were read during a test, the function won’t be able to estimate the MRS and CPS and will return NA values instead. The ACC should be used to estimate the MNREAD score in such cases where there are not enough data points to fit the MNREAD curve.

To ensure proper ACC calculation, the data should be entered along certain rules:

- For the smallest print size that is presented but not read, right before the test is stopped: **reading_time** = **NA**, **errors** = **10**
- For all the small sentences that are not presented because the test was stopped before them: **reading_time** = **NA**, **errors** = **NA**
- If a sentence is presented, and read, but the time was not recorded by the experimenter: **reading_time** = **NA**, **errors** = **actual number of errors** (cf. s5-regular in low vision data sample)
- If a large sentence was skipped to save time but would have been read well: **reading_time** = **NA**, **errors** = **NA** (cf. s1-regular in normal vision data sample)
- If a large sentence was skipped to save time because the subject cannot read large print: **reading_time** = **NA**, **errors** = **10** (cf. s7 in low vision data sample)

See Also

curveParam_RT for standard MRS and CPS estimation using values of reading time (instead of reading speed)
curveParam_RS for standard MRS and CPS estimation using values of reading speed (instead of reading time)
nlmeParam for MRS and CPS estimation using a nonlinear mixed-effect model (NLME)
readingAcuity for Reading Acuity calculation
accIndex for Reading Accessibility Index calculation

Examples

```r
# inspect the structure of the dataframe
head(data_low_vision, 10)

#------

# restrict dataset to one MNREAD test only (subject s1, regular polarity)
data_s1 <- data_low_vision %>%
  filter(subject == "s1", polarity == "regular")

# run the parameters estimation
data_low_vision_param <- mnreadParam(data_s1, ps, vd, rt, err)
```
# inspect the newly created dataframe
data_low_vision_param

#------

# run the parameters estimation on the whole dataset grouped by subject and polarity
data_low_vision_param <- mnreadParam(data_low_vision, ps, vd, rt, err, subject, polarity)

# inspect the structure of the newly created dataframe
head(data_low_vision_param, 10)

---

**mnreadR**

**mnreadR: An R package for analyzing MNREAD data**

---

**Description**

mnreadR provides simple functions to estimate the four MNREAD parameters:

- **Maximum Reading Speed** (MRS) -> can be estimated with the standard method: alone with `curveParam_RT` and `curveParam_RS` or simultaneously with the other MNREAD parameters with `mnreadParam`. -> Alternatively, it can be estimated with NLME modeling using `nlmeParam`.

- **Critical Print Size** (CPS) -> can be estimated with the standard method: alone with `curveParam_RT` and `curveParam_RS` or simultaneously with the other MNREAD parameters with `mnreadParam`. -> Alternatively, it can be estimated with NLME modeling using `nlmeParam`.

- **Reading Acuity** (RA) -> can be estimated alone with `readingAcuity` or simultaneously with the other MNREAD parameters with `mnreadParam`.

- **Reading ACCessibility Index** (ACC) -> can be estimated alone with `accIndex` or simultaneously with the other MNREAD parameters with `mnreadParam`.

mnreadR also provides functions for graphical display:

- Raw data can be plotted with `mnreadCurve`

- Estimates from the NLME fit can be plotted with `nlmeCurve`
nlmeCurve

Plot individual MNREAD fitted curves as estimated by a nonlinear mixed-effect (NLME) modeling.

Description

This function uses the NLME model created from nlmeModel to plot individual curves and Critical Print Size (CPS).

Usage

nlmeCurve(nlme.model, displayCPS = TRUE, CPScriterion = NULL)

Arguments

nlme.model The object returned by nlmeModel
displayCPS Optional argument to display the CPS on the individuals curves. Default is TRUE. If set to FALSE, the CPS won’t be plotted.
CPScriterion Optional argument to specify a criterion for CPS estimation. The default criterion value is ‘90 of MRS’. This criterion can vary from 75 to 95 of MRS and should only be modified for specific purposes, as discussed in Cheung et al. 2008

Value

The function returns a plot of reading speed (in log words/min) as a function of print size (in logMAR). If displayCPS is not specified, the Critical Print Size will be marked as an inverted triangle.

Notes

Print size shown on the plot(s) have been corrected for non-standard testing viewing distance.


See Also

nlmeModel to fit MNREAD data using a nonlinear mixed-effect (NLME) modeling
nlmeParam to estimate Maximum Reading Speed (MRS) and Critical Print Size (CPS) from the NLME model
mnreadCurve for standard MNREAD curve
Examples

# inspect the structure of the dataframe
head(data_low_vision, 10)

#------

# restrict dataset to one MNREAD test per subject (regular polarity only)
data_regular <- data_low_vision %>%
  filter (polarity == "regular")

# run the NLME model for data grouped by subject
nlme_model <- nlmeModel(data_regular, ps, vd, rt, err, subject)

#------

# plot MNREAD curves and CPS with a default CPS criterion of '90 of MRS'
nlmeCurve(nlme_model)

# plot MNREAD curves without the CPS for a default CPS criterion of '90 of MRS'
nlmeCurve(nlme_model, FALSE)

# plot MNREAD curves and CPS with a specific CPS criterion of '80 of MRS'
nlmeCurve(nlme_model, TRUE, 0.8)

#------

# Once created, the NLME curve can be further customized using ggplot2

# plot the NLME curve
my_plot <- nlmeCurve(nlme_model)

# display my.plot
print(my_plot)

# modify my.plot
my_new_plot <- my_plot +
  # overwrites the raw data points
  geom_point(data = nlme_model[[1]], aes(x=correct_ps, y = rs), size = 4) +
  # changes the colors of the curve and raw data (effective only for nested designs)
  scale_color_brewer(palette="Set1") +
  # changes the colors of the CPS diamond (effective only for nested designs)
  scale_fill_brewer(palette="Set1") +
  # modifies the aspect of the x-axis
  scale_x_continuous(breaks = seq (-0.5,2.5,0.4))

# display my.new.plot
print(my_new_plot)

#------

# For very large datasets, it can be usefull to plot only selected facets to inspect individual fit
# To do so, one needs to restrict the dataframe called in each of the three layers of the plot

# list of subject names to keep
subjects_to_keep <- paste("s", 1:4, sep = "")

# first filter the original data points (data called in the first layer)
my_plot$data <- my_plot$data %>%
  filter(subject %in% subjects_to_keep) %>%
  droplevels()

# then filter the fitted data points (data called in the second layer)
my_plot$layers[[2]]$data <- my_plot$layers[[2]]$data %>%
  filter(subject %in% subjects_to_keep) %>%
  droplevels()

# and finally, if 'displayCPS' was set to TRUE, filter the data used to display the CPS
my_plot$layers[[4]]$data <- my_plot$layers[[4]]$data %>%
  filter(subject %in% subjects_to_keep) %>%
  droplevels()

# plot the restricted my.plot
my_plot

#------

# It is also possible to export the curves in a pdf file running over several pages
# and select the desired number of curves per page

# set the desired number of subjects by page
facet_nb = 4

# count the resulting number of pages
num_pages = ceiling(length(unique(data_low_vision$subject))/facet_nb)

# identify the list of subject names
subjects_to_plot <- unique(as.character(data_low_vision$subject))

# split the list into chunks the same size as the number of subjects per page
subjects_to_plot_split <- split(subjects_to_plot, ceiling(seq_along(subjects_to_plot)/facet_nb))

# create a pdf and wrap plots over several pages
pdf("nlme-MNREAD-curves.pdf",
    width = 10.5, height = 8,
    paper="USr", useDingbats=T)

for (i in seq(num_pages))
{
  my.plot <- nlmeCurve(nlme_model, displayCPS = F)

  # filter the original data points for the selected chunk of subjects
  my.plot$data <- my.plot$data %>%
    filter(subject %in% subjects_to_plot_split[[i]]) %>%
    droplevels()
# filter the fitted data points for the selected chunk of subjects
my.plot$layers[[2]]$data <- my.plot$layers[[2]]$data %>%
filter(subject %in% subjects_to_plot_splitted[[i]]) %>%
droplevels()

print (my.plot + geom_line(colour = "red"))
}

dev.off()
subjectID  The variable that contains the subject identifiers

nested  Optional argument to build a model with a nested structure. ‘nested’ specifies which variable should be nested within subject. Default is NULL.

group  Optional argument to build a model with a grouped structure. ‘group’ specifies which variable should be used as grouping argument. Default is NULL.

Value

The function returns a list of two objects:

• an object of class dataframe which is a cleaned version of the dataset called by the function to fit the model
• an object of class nlme returned by the function nlme

Notes

For subjects with incomplete data, warning messages might appear in the console. However, the NLME model will run, using supporting data from the rest of the population.

This functions supports nested, grouped and nested + grouped structures.

If needed, the nlme object returned can be further explored using generic functions from the nlme package.

This function implements several functions from the nlme package to build the NLME model:

• it first calls groupedData() to format the dataset in order to match the desired structure
• it then uses nlsList() to generate starting values
• it finally calls nlme() to build the model


Warning

For the function to run properly, please make sure that variables are of the following classes:

• print_size  -> numeric
• viewing_distance  -> integer
• reading_time  -> numeric
• errors  -> integer

The optional arguments "nested" and "group" should only be specified when they are needed. In case they are called and set to NULL, the function will not run and will return an error.

See Also

nlmeParam to estimate Maximum Reading Speed (MRS) and Critical Print Size (CPS) from the NLME model

nlmeCurve to plot the individual MNREAD curves estimated from the NLME model
Examples

# inspect the structure of the dataframe
head(data_low_vision, 10)

#------

# restrict dataset to one MNREAD test per subject (regular polarity only)
data_regular <- data_low_vision %>%
  filter (polarity == "regular")

# run the NLME model for data grouped by subject
model_simple <- nlmeModel(data_regular, ps, vd, rt, err, subject)

# to print the model summary
summary(model_simple[[2]])

# to print the first 3 rows of the cleaned dataset containing the raw data and used to run the model
head(model_simple[[1]], 3)

#------

# run the NLME model on the whole dataset with polarity nested within subject
model_nested <- lmeModel(data_low_vision, ps, vd, rt, err, subject,
                          nested = polarity)

#------

# run the NLME model on the whole dataset with polarity nested within subject
# and grouped based on treatment
model_nested_grouped <- nlmeModel(data_low_vision, ps, vd, rt, err, subject,
                                   nested = polarity, group = treatment)

---

nlmeParam

Maximum Reading Speed (MRS) and Critical Print Size (CPS) estimation using a nonlinear mixed-effect (NLME) modeling.

Description

This function uses the NLME model created from nlmeModel to extract the following MNREAD parameters:

- Maximum Reading Speed (MRS)
- Critical Print Size (CPS)

Usage

nlmeParam(nlme.model, CPScriterion = NULL)
nlmeParam

Arguments

nlme.model The object returned by nlmeModel
CPScriterion Optional argument to specify a criterion for CPS estimation. The default criterion value is '90 of MRS'. This criterion can vary from 75 to 95 of MRS and should only be modified for specific purposes, as discussed in Cheung et al. 2008

Value

The function returns a new dataframe with two variables:

- "CPS" -> contains the Critical Print Size estimate (in logMAR)
- "MRS" -> contains the Maximum Reading Speed estimate (in words/min)

Notes

To ensure proper estimation of the MRS and CPS, individual MNREAD fit should be plotted using nlmeCurve and inspected visually. If some of the estimated values of MRS and CPS seem off given the actual data, we advise you to run mnreadCurve and overwrite these estimates with values estimated visually from the actual MNREAD curve.


See Also

nlmeModel to fit MNREAD data using a nonlinear mixed-effect (NLME) modeling
nlmeCurve to plot the individual MNREAD curves estimated from the NLME model
curveParam_RT for standard estimation of MRS and CPS
mnreadParam for all MNREAD parameters estimation

Examples

# inspect the structure of the dataframe
head(data_low_vision, 10)

#-----

# restrict dataset to one MNREAD test per subject (regular polarity only)
data_regular <- data_low_vision %>%
  filter (polarity == "regular")

# run the NLME model for data grouped by subject
nlme_model <- nlmeModel(data_regular, ps, vd, rt, err, subject)

#-----

# run the parameters' estimation for a default CPS criterion of '90 of MRS'
nlmeParam(nlme_model)
# run the parameters' estimation for a specific CPS criterion of '80 of MRS'
nlmeParam(nlme_model, 0.8)

nlmePredict_PS | Estimation of the print size value necessary to achieve a given reading speed.

Description
This function uses results from the NLME model created with nlmeModel to estimate the print size value required to achieve a specific reading speed.

Usage

nlmePredict_PS(nlme.model, reading.speed)

Arguments

- nlme.model: The object returned by nlmeModel
- reading.speed: A specific value of reading speed in words/minute

Value
The function returns a dataframe with a with two variables:
- "reading_speed" -> the reading speed value passed to the function (in words/min)
- "required_print_size" -> the print size required to achieve the reading speed value passed to the function (in logMAR)

Notes
The values of print size returned have been corrected for non-standard testing viewing distance.

See Also

- nlmeModel to fit MNREAD data using a nonlinear mixed-effect (NLME) modeling
- nlmeParam to estimate Maximum Reading Speed (MRS) and Critical Print Size (CPS) from the NLME model
- nlmeCurve to plot the individual MNREAD curves estimated from the NLME model
Examples

```r
# inspect the structure of the dataframe
head(data_low_vision, 10)

#------

# restrict dataset to one MNREAD test per subject (regular polarity only)
data_regular <- data_low_vision %>%
  filter (polarity == "regular")

# run the NLME model for data grouped by subject
nlme_model <- nlmeModel(data_regular, ps, vd, rt, err, subject)

#------

# extract the critical print size required
# to achieve 40 words/min (ie. spot reading) according to the NLME fit
nlmePredict_PS(nlme_model, 40)

#------

# extract the critical print size required
# to achieve 80 words/min (ie. fluent reading) according to the NLME fit
nlmePredict_PS(nlme_model, 80)
```

---

**nlmePredict_RS**

*Estimation of the reading speed achieved for a given print size.*

**Description**

This function uses results from the NLME model created with `nlmeModel` to estimate the reading speed achieved for a specific print size.

**Usage**

```r
nlmePredict_RS(nlme.model, print.size)
```

**Arguments**

- `nlme.model`: The object returned by `nlmeModel`
- `print.size`: A specific value of print size in logMAR
Value

The function returns a dataframe with with two variables:

- "print_size" -> the print size value passed to the function (in logMAR)
- "estimated_reading_speed" -> the reading speed achieved at the specified print size as estimated by the NLME model (in words/min)

Notes

The values of print size returned have been corrected for non-standard testing viewing distance.


See Also

- `nlmeModel` to fit MNREAD data using a nonlinear mixed-effect (NLME) modeling
- `nlmeParam` to estimate Maximum Reading Speed (MRS) and Critical Print Size (CPS) from the NLME model
- `nlmeCurve` to plot the individual MNREAD curves estimated from the NLME model

Examples

```r
# inspect the structure of the dataframe
head(data_low_vision, 10)

#-----

# restrict dataset to one MNREAD test per subject (regular polarity only)
data_regular <- data_low_vision %>%
  filter (polarity == "regular")

# run the NLME model for data grouped by subject
nlme_model <- nlmeModel(data_regular, ps, vd, rt, err, subject)

#-----

# extract reading speed achieved at 1.6 logMAR according to the NLME fit
nlmePredict_RS(nlme_model, 1.6)
```
Reading Acuity (RA) calculation

**Description**

Reading Acuity (RA) is defined as the smallest print size at which one can read without making significant errors. This function measures Reading Acuity to the nearest 0.1 logMAR, while performing print size correction for non-standard testing viewing distance.

**Usage**

```r
readingAcuity(
  data,
  print_size,
  viewing_distance,
  reading_time,
  errors,
  ... = NULL
)
```

**Arguments**

- `data` The name of your dataframe
- `print_size` The variable that contains print size values for each sentence (print size uncorrected for viewing distance)
- `viewing_distance` The variable that contains the viewing distance value used for testing
- `reading_time` The variable that contains the reading time for each sentence
- `errors` The variable that contains the number of errors for each sentence
- `...` Optional grouping arguments

**Value**

The function returns a new dataframe with a variable called "RA" that contains the Reading Acuity estimate (in logMAR).

**See Also**

- `mnreadParam` for all MNREAD parameters estimation
- `curveParam_RT` for MRS and CPS estimation using values of reading time (instead of reading speed)
- `curveParam_RS` for MRS and CPS estimation using values of reading speed (instead of reading time)
- `accIndex` for Reading Accessibility Index calculation
`readingSpeed`  

**Examples**  

```r  
# inspect the structure of the dataframe  
head(data_low_vision, 10)  

#------  

# restrict dataset to one MNREAD test only (subject s1, regular polarity)  
data_s1 <- data_low_vision %>% filter(subject == "s1" & polarity == "regular")  

# run the reading acuity calculation  
data_low_vision_RA <- readingAcuity(data_s1, ps, vd, rt, err)  

# inspect the newly created dataframe  
data_low_vision_RA  

#------  

# run the reading acuity calculation on the whole dataset grouped by subject and polarity  
data_low_vision_RA <- readingAcuity(data_low_vision, ps, vd, rt, err,  
subject, polarity)  

# inspect the structure of the newly created dataframe  
head(data_low_vision_RA, 10)  
```

---

`readingSpeed`  

*Reading speed calculation corrected for the number of errors*

**Description**

This function calculates reading speed (in words per minute) for each sentence tested. This calculation takes into account the number of misread words and gives a more precise reading speed measurement than `readingSpeed_nonCorrected`.

**Usage**

`readingSpeed(data, reading_time, errors)`

**Arguments**

- `data`  
  The name of your dataframe
- `reading_time`  
  The variable that contains the reading time for each sentence
- `errors`  
  The variable that contains the number of errors for each sentence

**Value**

The function returns the original dataframe with an added variable called "reading_speed" that contains reading speed (in words/min) for each sentence tested.
Notes

For general purposes, this method of reading speed calculation should be used preferentially over the less precise `readingSpeed_nonCorrected`.

See Also

`readingSpeed_nonCorrected` for reading speed non corrected for errors

Examples

```r
# inspect the strucutre of the dataframe
head(data_low_vision, 10)

# run the reading speed calculation
data_low_vision_new <- readingSpeed(data_low_vision, rt, err)

# inspect the structure of the newly created dataframe
head(data_low_vision_new, 10)
```

---

`readingSpeed_nonCorrected`

*Reading speed calculation not corrected for the number of errors*

Description

This function calculates reading speed (in words per minute) using reading time (in seconds) only. This calculation provides a simplified value of reading speed, that does not take into account the number of misread words.

Usage

`readingSpeed_nonCorrected(data, reading_time)`

Arguments

- `data` The name of your dataframe
- `reading_time` The variable that contains the reading time for each sentence

Value

The function returns the original dataframe with an added variable called "reading_speed_nonCorrected" that contains reading speed (in words/min) for each sentence tested.

Notes

This function gives a less precise reading speed measurement than `readingSpeed`. Unless you know what you are doing, consider using `readingSpeed` instead of this function.
See Also

readingSpeed for reading speed corrected for errors

Examples

# inspect the structure of the dataframe
head(data_low_vision, 10)

# run the reading speed calculation
data_low_vision_new <- readingSpeed_nonCorrected(data_low_vision, rt)

# inspect the structure of the newly created dataframe
head(data_low_vision_new, 10)
Index

* datasets
  - data_low_vision, 9
  - data_normal_vision, 9

accIndex, 2, 5, 8, 12, 16, 17, 28

curveParam_RS, 3, 4, 8, 16, 17, 28
curveParam_RT, 3, 5, 6, 12, 16, 17, 24, 28

data_low_vision, 9
data_normal_vision, 9

logMARcorrect, 10

mnreadCurve, 5, 7, 11, 15, 17, 18, 24
mnreadParam, 3, 5, 8, 14, 17, 24, 28
mnreadR, 17

nlme, 22

nlmeCurve, 17, 18, 22, 24, 25, 27
nlmeModel, 18, 21, 23–27

nlmeParam, 5, 8, 16–18, 22, 23, 25, 27

nlmePredict_PS, 25
nlmePredict_RS, 26

readingAcuity, 3, 5, 8, 12, 16, 17, 28
readingSpeed, 29, 30, 31
readingSpeed_nonCorrected, 29, 30, 30