Package ‘xgxr’

April 14, 2020

Title Exploratory Graphics for Pharmacometrics

Version 1.0.9

Description Supports a structured approach for exploring PKPD data <https://opensource.nibr.com/xgx>. It also contains helper functions for enabling the modeler to follow best R practices (by appending the program name, figure name location, and draft status to each plot). In addition, it enables the modeler to follow best graphical practices (by providing a theme that reduces chart ink, and by providing time-scale, log-scale, and reverse-log-transform-scale functions for more readable axes). Finally, it provides some data checking and summarizing functions for rapidly exploring pharmacokinetics and pharmacodynamics (PKPD) datasets.

License MIT + file LICENSE

URL https://opensource.nibr.com/xgx

Depends R (>= 3.5.0)

Imports assertthat, binom, dplyr, ggplot2, graphics, grDevices, labeling, magrittr, pander, png, scales, stats, tibble, utils

Suggests caTools, gridExtra, knitr, rmarkdown, RxODE, stringr, testthat, tidyr

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

NeedsCompilation no

Author Andrew Stein [aut, cre], Alison Margolskee [aut], Fariba Khanshan [aut], Konstantin Krismer [aut] (<https://orcid.org/0000-0001-8994-3416>), Matthew Fidler [ctb] (<https://orcid.org/0000-0001-8538-6691>), Novartis Pharma AG [cph, fnd]

Maintainer Andrew Stein <andy.stein@gmail.com>
Repository CRAN
Date/Publication 2020-04-14 12:20:03 UTC

R topics documented:

- case1_pkpd .................................................. 3
- mad ......................................................... 4
- mad_missing_duplicates .................................. 4
- mad_nca ..................................................... 5
- nlmixr_theo_sd ............................................. 6
- sad ......................................................... 6
- theme_xgx .................................................. 7
- xgx_anotate_filenames ................................... 8
- xgx_anotate_status ....................................... 9
- xgx_anotate_status_png ................................ 10
- xgx_breaks_log10 ......................................... 12
- xgx_breaks_time .......................................... 13
- xgx_check_data ........................................... 14
- xgx_dirs2char ............................................. 15
- xgx_geom_ci ............................................... 16
- xgx_geom_pi ............................................... 17
- xgx_labels_log10 ......................................... 19
- xgx_minor_breaks_log10 ................................ 19
- xgx_plot ................................................... 20
- xgx_save ................................................... 21
- xgx_save_table ........................................... 22
- xgx_scale_x_log10 ........................................ 23
- xgx_scale_x_reverselog10 ............................... 24
- xgx_scale_x_time_units ................................ 25
- xgx_scale_y_log10 ........................................ 26
- xgx_scale_y_percentchangelog10 ........................ 27
- xgx_scale_y_reverselog10 ............................... 28
- xgx_stat_ci ............................................... 29
- xgx_stat_pi ............................................... 31
- xgx_summarize_covariates ................................ 33
- xgx_summarize_data ...................................... 33
- xgx_theme .................................................. 34
- xgx_theme_set ............................................ 35

Index 36
### Case 1 PKPD Data Set

**Description**

Case 1 PKPD Data Set

**Usage**

case1_pkpd

**Format**

A data frame with the following 21 columns:

<table>
<thead>
<tr>
<th>Column</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ID</td>
<td>integer; unique subject ID</td>
</tr>
<tr>
<td>2</td>
<td>TIME</td>
<td>numeric; time relative to first drug administration</td>
</tr>
<tr>
<td>3</td>
<td>NOMTIME</td>
<td>numeric; nominal time</td>
</tr>
<tr>
<td>4</td>
<td>TIMEUNIT</td>
<td>factor; unit of TIME</td>
</tr>
<tr>
<td>5</td>
<td>AMT</td>
<td>integer; dosing amount (for dosing events) in mg</td>
</tr>
<tr>
<td>6</td>
<td>LIDV</td>
<td>numeric; observation on a linear scale (observation type determined by CMT), units determined by EVENTU column</td>
</tr>
<tr>
<td>7</td>
<td>CMT</td>
<td>integer; compartment number (determines observation type):</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CMT 1 = Dosing event</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CMT 2 = PK concentration</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CMT 3 = Continuous response data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CMT 4 = Count response data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CMT 5 = Ordinal response data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CMT 6 = Binary response data</td>
</tr>
<tr>
<td>8</td>
<td>NAME</td>
<td>factor; description of event</td>
</tr>
<tr>
<td>9</td>
<td>EVENTU</td>
<td>factor; unit for observation</td>
</tr>
<tr>
<td>10</td>
<td>CENS</td>
<td>integer; censored values (0 = not censored, 1 = censored)</td>
</tr>
<tr>
<td>11</td>
<td>EVID</td>
<td>integer; event ID (0 = observation, 1 = dosing event)</td>
</tr>
<tr>
<td>12</td>
<td>WEIGHTB</td>
<td>numeric; baseline body weight (kg)</td>
</tr>
<tr>
<td>13</td>
<td>eff0</td>
<td>numeric; efficacy</td>
</tr>
<tr>
<td>14</td>
<td>TRTACT</td>
<td>factor; treatment group label</td>
</tr>
<tr>
<td>15</td>
<td>DOSE</td>
<td>integer; Dose in mg</td>
</tr>
<tr>
<td>16</td>
<td>PROFDAY</td>
<td>integer; day of profile</td>
</tr>
<tr>
<td>17</td>
<td>PROFTIME</td>
<td>numeric; time within PROFDAY</td>
</tr>
<tr>
<td>18</td>
<td>CYCLE</td>
<td>integer; count of drug administrations received</td>
</tr>
<tr>
<td>19</td>
<td>PART</td>
<td>integer; part of study</td>
</tr>
<tr>
<td>20</td>
<td>STUDY</td>
<td>integer; study</td>
</tr>
<tr>
<td>21</td>
<td>IPRED</td>
<td>numeric; individual prediction</td>
</tr>
</tbody>
</table>
**mad**  
*Multiple Ascending Dose Data Set*

**Description**

Model generated PK and PD data to mimic an orally administered small molecule with various endpoints from continuous to ordinal response and count data. Simulated multiple dose administration ranging from 100 mg to 1600 mg, once per day.

**Usage**

mad

**Format**

A data frame with the following 19 columns:

- **column 1**: ID  numeric; unique subject ID
- **column 2**: TIME  numeric; time relative to first drug administration
- **column 3**: NOMTIME  numeric; nominal time
- **column 4**: TIMEUNIT  character; unit of TIME
- **column 5**: AMT  numeric; dosing amount (for dosing events) in mg
- **column 6**: LIDV  numeric; observation on a linear scale (observation type determined by CMT), units determined by EVENTU column
- **column 7**: MDV  numeric; missing dependent variable
- **column 8**: CMT  integer; compartment number (determines observation type):
  - CMT 1 = Dosing event
  - CMT 2 = PK concentration
  - CMT 3 = Continuous response data
  - CMT 4 = Count response data
  - CMT 5 = Ordinal response data
  - CMT 6 = Binary response data
- **column 9**: NAME  character; description of event
- **column 10**: EVENTU  character; unit for observation
- **column 11**: CENS  integer; censored values (0 = not censored, 1 = censored)
- **column 12**: EVID  integer; event ID (0 = observation, 1 = dosing event)
- **column 13**: WEIGHTB  numeric; baseline body weight (kg)
- **column 14**: SEX  character; sex
- **column 15**: TRTACT  factor; treatment group label
- **column 16**: DOSE  numeric; randomized dose in mg
- **column 17**: PROFDAY  numeric; day of profile
- **column 18**: PROFTIME  numeric; time within PROFDAY
- **column 19**: CYCLE  numeric; count of drug administrations received
**mad_missing_duplicates**

*Multiple Ascending Dose Data Set (Duplicates Removed)*

---

**Description**

Model generated PK and PD data to mimic an orally administered small molecule with various endpoints from continuous to ordinal response and count data. Simulated multiple dose administration ranging from 100 mg to 1600 mg, once per day.

**Usage**

mad_missing_duplicates

**Format**

A data frame with the following 19 columns:

- **column 1:** ID  
  numeric; unique subject ID
- **column 2:** TIME  
  numeric; time relative to first drug administration
- **column 3:** NOMTIME  
  numeric; nominal time
- **column 4:** TIMEUNIT  
  character; unit of TIME
- **column 5:** AMT  
  numeric; dosing amount (for dosing events) in mg
- **column 6:** LIDV  
  numeric; observation on a linear scale (observation type determined by CMT), units determined by EVENTU
- **column 7:** MDV  
  numeric; missing dependent variable
- **column 8:** CMT  
  integer; compartment number (determines observation type):
  
  - CMT 1 = Dosing event
  - CMT 2 = PK concentration
  - CMT 3 = Continuous response data
  - CMT 4 = Count response data
  - CMT 5 = Ordinal response data
  - CMT 6 = Binary response data
- **column 9:** NAME  
  character; description of event
- **column 10:** EVENTU  
  character; unit for observation
- **column 11:** CENS  
  integer; censored values (0 = not censored, 1 = censored)
- **column 12:** EVID  
  integer; event ID (0 = observation, 1 = dosing event)
- **column 13:** WEIGHTB  
  numeric; baseline body weight (kg)
- **column 14:** SEX  
  character; sex
- **column 15:** TRTACT  
  factor; treatment group label
- **column 16:** DOSE  
  numeric; randomized dose in mg
- **column 17:** PROFDAY  
  numeric; day of profile
- **column 18:** PROFTIME  
  numeric; time within PROFDAY
- **column 19:** CYCLE  
  numeric; count of drug administrations received

---

**mad_nca**

*Multiple Ascending Dose Noncompartmental Analysis (NCA) dataset*
Description

Multiple Ascending Dose Noncompartmental Analysis (NCA) dataset

Usage

mad_nca

Format

A data frame with the following 7 columns:

- column 1: ID numeric; unique subject ID
- column 2: PARAM character; NCA parameter
- column 3: VALUE numeric; Value of the NCA parameter
- column 4: DOSE numeric; randomized dose in mg
- column 15: TRTACT factor; treatment group label
- column 14: SEX character; sex
- column 13: WEIGHTB numeric; baseline body weight (kg)

nlmixr_theo_sd

Description

Theophylline dataset, from the nlmixr R package

Usage

nlmixr_theo_sd

Format

A data frame with the following 7 columns:

- column 1: ID integer; unique patient identifier
- column 2: TIME numeric; time relative to first drug administration
- column 3: DV numeric; dependent variable (drug concentration)
- column 4: AMT numeric; dose of drug
- column 5: EVID integer; event ID, 1 if dose, 0 otherwise
- column 6: CMT integer; compartment number
- column 7: WT numeric; weight

sad

Description

Single Ascending Dose Data Set

Usage

sad
### Description

Model generated PK data to mimic an orally administered small molecule. Simulated single dose administration ranging from 100 mg to 1600 mg.

### Usage

sad

### Format

A data frame with the following 16 columns:

<table>
<thead>
<tr>
<th>Column</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ID</td>
<td>numeric; unique subject ID</td>
</tr>
<tr>
<td>2</td>
<td>TIME</td>
<td>numeric; time relative to first drug administration</td>
</tr>
<tr>
<td>3</td>
<td>NOMTIME</td>
<td>numeric; nominal time</td>
</tr>
<tr>
<td>4</td>
<td>TIMEUNIT</td>
<td>character; unit of TIME</td>
</tr>
<tr>
<td>5</td>
<td>AMT</td>
<td>numeric; dosing amount (for dosing events) in mg</td>
</tr>
<tr>
<td>6</td>
<td>LIDV</td>
<td>numeric; observation on a linear scale (observation type determined by CMT), units</td>
</tr>
<tr>
<td>7</td>
<td>MDV</td>
<td>numeric; missing dependent variable (1 if missing, 0 otherwise)</td>
</tr>
<tr>
<td>8</td>
<td>CMT</td>
<td>integer; compartment number (determines observation type):</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CMT 1 = Dosing event</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CMT 2 = PK concentration</td>
</tr>
<tr>
<td>9</td>
<td>NAME</td>
<td>character; description of event</td>
</tr>
<tr>
<td>10</td>
<td>EVENTU</td>
<td>character; unit for observation</td>
</tr>
<tr>
<td>11</td>
<td>CENS</td>
<td>integer; censored values (0 = not censored, 1 = censored)</td>
</tr>
<tr>
<td>12</td>
<td>EVID</td>
<td>integer; event ID (0 = observation, 1 = dosing event)</td>
</tr>
<tr>
<td>13</td>
<td>WEIGHTB</td>
<td>numeric; baseline body weight (kg)</td>
</tr>
<tr>
<td>14</td>
<td>SEX</td>
<td>character; sex</td>
</tr>
<tr>
<td>15</td>
<td>TRTACT</td>
<td>factor; treatment group label</td>
</tr>
<tr>
<td>16</td>
<td>DOSE</td>
<td>numeric; randomized dose in mg received</td>
</tr>
</tbody>
</table>

---

**theme_xgx**

Calls the standard theme for xGx graphics

### Description

Calls the standard theme for xGx graphics

### Usage

```
theme_xgx()
```

### Value

xgx ggplot2 compatible theme
Examples

conc <- 10^(seq(-3, 3, by = 0.1))
ec50 <- 1
data <- data.frame(concentration = conc,
    bound_receptor = 1 * conc / (conc + ec50))
ggplot2::ggplot(data, ggplot2::aes(y = concentration, x = bound_receptor)) +
ggplot2::geom_point() +
ggplot2::geom_line() +
xgx_scale_y_log10() +
xgx_scale_x_reverselog10() +
theme_xgx()

xgx_annotate_filenames

Append filenames to bottom of the plot

Description

xgx_annotate_filenames appends file details to the bottom of a plot using the plot caption option. File details to append include the parent directory, the path of the R script which generated the plot, and the path of the plot.

Usage

xgx_annotate_filenames(dirs, hjust = 0.5)

Arguments

dirs list containing directories and filenames. It must contain five fields
   1. parent_dir = Parent directory containing the Rscript and the Results folder
   2. rscript_dir = Subdirectory of parent_dir that contains the Rscript used to
      generate the figure
   3. rscript_name = Name of the Rscript used to generate the figure
   4. results_dir = Subdirectory of parent_dir where the figure is stored
   5. filename = Filename

hjust horizontal justification of the caption

Value

None
Examples

dirs <- list(parent_dir = "~/your/parent/path/",
             rscript_dir = "./Rscripts/",
             rscript_name = "Example.R",
             results_dir = "./Results/",
             filename = "your_file_name.png")
data <- data.frame(x = 1:1000, y = rnorm(1000))
ggplot2::ggplot(data = data, ggplot2::aes(x = x, y = y)) +
ggplot2::geom_point() +
xgx_annotate_filenames(dirs)

Description

xgx_annotate_status adds a status (e.g. DRAFT) annotation layer to a plot. The text of the annotation can be customized, the default is "DRAFT". The color, location, size, fontface, transparency of the annotation can also be customized.

Usage

xgx_annotate_status(
  status = "DRAFT",
  x = Inf,
  y = Inf,
  color = "grey",
  hjust = 1.2,
  vjust = 1.2,
  fontsize = 7,
  fontface = "bold",
  alpha = 0.5,
  ...
)

Arguments

status the text to
x x location, default Inf (right most point)
y y location, default Inf (up most point)
color default "grey"
hjust horizontal justification, default 1.2
vjust vertical justification, default 1.2
fontsize font size to use, default 7
fontface font style to use, default "bold"
alpha transparency, default is 0.5
... other arguments passed on to layer

Value

ggplot layer

Examples

data <- data.frame(x = 1:1000, y = rnorm(1000))
ggplot2::ggplot(data = data, ggplot2::aes(x = x, y = y)) +
ggplot2::geom_point() +
xgx_annotate_status("DRAFT")

xgx_annotate_status_png

Annotate a png file or directory of png files

Description

These function annotates a single png file or all files within a directory.

Usage

xgx_annotate_status_png(
  file_or_dir,
  script = "",
  status = "DRAFT",
  date_format = "%a %b %d %X %Y",
  col = grDevices::grey(0.8, alpha = 0.7),
  font = 2,
  cex_status_mult = 7,
  cex_footnote_mult = 0.8,
  status_angle = 45,
  x11 = FALSE
)

Arguments

file_or_dir The png file to annotate or directory location for annotating png files. Note this will annotate just once, so if you generate multiple png files and then annotate at the end of your script it will have the correct script name on it. Then if you create new images in a different script in the same directory and then annotate with the script name the second script, the PNG files will show the correct script location for each file.
xgx_annotate_status_png

script Script name to add as a footnote; By default this is empty, though it could name the script that

status Draft or other status; If `status="Final"` or `status=""` the status overlay will be removed. By default the status is DRAFT.

date_format Date format for adding the time the png was annotated.

col Color for annotating the draft status

font Font to use for the annotation function

cex_status_mult Multiplication factor for the status annotation. By default 7

cex_footnote_mult Multiplication factor for the footnote annotation. By default 0.8

status_angle Angle to rotate status

x11 Display on the X11/Windows device

Details

If a png file has been annotated once, this function will not annotate it again. Therefore, you can run this function on directories with different input script names and it will label each file based on when each file was run.

Based on code from MrFlick on Stack Overflow.

Value

nothing

Author(s)

Matthew Fidler, Alison M, ....

Examples

```r
# using the examples from plot()
file.name <- tempfile()
grDevices::png(file.name)
grDevices::plot(cars)
grDevices::lines(stats::lowess(cars))
grDevices::dev.off()
# annotate one file
xgx_annotate_status_png(file.name, "/tmp/script1.R")
```
xgx_breaks_log10

Sets the default breaks for log10

Description

xgx_breaks_log10 sets nice breaks for log10 scale. It’s better than the default function because it ensures there is at least 2 breaks and also, it will try to go by 3s (i.e. 1,3,10,30,100) if it makes sense.

Usage

xgx_breaks_log10(data_range)

Arguments

data_range range of the data

Details

For the extended breaks function, weights is a set of 4 weights for:

1. simplicity - how early in the Q order are you
2. coverage - labelings that don’t extend outside the data: range(data) / range(labels)
3. density (previously granularity) - how close to the number of ticks do you get (default is 5)
4. legibility - has to do with fontsize and formatting to prevent label overlap

Value

Numeric vector of breaks

References


Examples

xgx_breaks_log10(c(1, 1000))
xgx_breaks_log10(c(0.001, 100))
xgx_breaks_log10(c(1e-4, 1e4))
xgx_breaks_log10(c(1e-9, 1e9))
xgx_breaks_log10(c(1, 2))
xgx_breaks_log10(c(1, 5))
xgx_breaks_log10(c(1, 10))
xgx_breaks_log10(c(1, 100))
xgx_breaks_log10(c(1, 1000))
xgx_breaks_log10(c(1, 1.000001))
print(xgx_breaks_log10(c(1, 1.000001)), digits = 10)
xgx_breaks_time

Sets the default breaks for a time axis

Description

\texttt{xgx_breaks_time} sets the default breaks for a time axis, given the units of the data and the units of the plot. It is inspired by scales::extended_breaks

Usage

\begin{verbatim}
xgx_breaks_time(data_range, units_plot, number_breaks = 5)
\end{verbatim}

Arguments

\begin{description}
\item[data_range] range of the data
\item[units_plot] units to use in the plot
\item[number_breaks] number of breaks to aim for (default is 5)
\end{description}

Details

for the extended breaks function, weights is a set of 4 weights for

1. simplicity - how early in the Q order are you
2. coverage - labelings that don’t extend outside the data: range(data) / range(labels)
3. density (previously granularity) - how close to the number of ticks do you get (default is 5)
4. legibility - has to do with fontsize and formatting to prevent label overlap

Value

numeric vector of breaks

References


Examples

\begin{verbatim}
xgx_breaks_time(c(0, 5), "h")
xgx_breaks_time(c(0, 6), "h")
xgx_breaks_time(c(-3, 5), "h")
xgx_breaks_time(c(0, 24), "h")
xgx_breaks_time(c(0, 12), "h")
xgx_breaks_time(c(1, 4), "d")
xgx_breaks_time(c(1, 12), "d")
xgx_breaks_time(c(1, 14), "d")
\end{verbatim}
Description

`xgx_check_data` performs a series of checks on a PK or PKPD dataset. It was inspired by the dataset preparation table from IntiQuan.

Usage

```
xgx_check_data(data, covariates = NULL)
```

Arguments

- `data`, the dataset to check. Must contain the above columns
- `covariates`, the column names of covariates, to explore

Details

The dataset must have the following columns:

- **ID** = unique subject identifier. USUBJID is another option if ID is not there
- **EVID** = event ID: 1 for dose, 0 otherwise
- **AMT** = value of the dose
- **TIME** = time of the measurement
- **DV** = dependent value (linear scale). Will check if LIDV or LNDV are also there if DV is not
- **YTYPE** = data measurement for LIDV. Will check if CMT is there, if YTYPE is not

The dataset may also have additional columns:

- **CENS** = flag for censoring of the data because it’s below the limit of quantification (BLOQ)
- **MDV** = missing dependent variable - will be counted and then filtered out from the data check

Value

data.frame

Examples

```r
covariates <- c("WEIGHTB", "SEX")
check <- xgx_check_data(mad_missing_duplicates, covariates)
```
Description

*xgx_dirs2char* returns a character variable based on the dirs list. The caption gives the filename.

Usage

```r
xgx_dirs2char(dirs, include_time = TRUE)
```

Arguments

- **dirs**: list containing directories and filenames. It must contain five fields:
  1. `parent_dir`: Parent directory containing the Rscript and the Results folder.
  2. `rscript_dir`: Subdirectory of `parent_dir` that contains the Rscript used to generate the figure.
  3. `rscript_name`: Name of the Rscript used to generate the figure.
  4. `results_dir`: Subdirectory of `parent_dir` where the figure is stored.
  5. `filename`: Filename.
- **include_time**: is logical with default `TRUE`. If `TRUE`, it includes date/time in the output character.

Value

- character

Examples

```r
dirs <- list(parent_dir = "/your/parent/path/",
             rscript_dir = "/./Rscripts/",
             rscript_name = "Example.R",
             results_dir = "/./Results/",
             filename = "your_file_name.png")
caption <- xgx_dirs2char(dirs)
```
xgx_geom_ci

Plot data with mean and confidence intervals

Description

Plot data with mean and confidence intervals

Usage

xgx_geom_ci(
mapping = NULL,
data = NULL,
conf_level = 0.95,
distribution = "normal",
geom = list("point", "line", "errorbar"),
position = "identity",
fun.args = list(),
na.rm = FALSE,
show.legend = NA,
inherit.aes = TRUE,
...)

Arguments

mapping Set of aesthetic mappings created by ‘aes’ or ‘aes_’. If specified and ‘inherit.aes = TRUE’ (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot.
A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data.
conf_level The percentile for the confidence interval (should fall between 0 and 1). The default is 0.95, which corresponds to a 95 percent confidence interval.
distribution The distribution which the data follow, used for calculating confidence intervals. The options are "normal", "lognormal", and "binomial". The "normal" option will use the Student t Distribution to calculate confidence intervals, the "lognormal" option will transform data to the log space first. The "binomial" option will use the `binom.exact` function to calculate the confidence intervals. Note: binomial data must be numeric and contain only 1's and 0's.
geom Use to override the default geom. Can be a list of multiple geoms, e.g. list("point","line","errorbar"), which is the default.
**xgx_geom_pi**

position  
Position adjustment, either as a string, or the result of a call to a position adjustment function.

fun.args  
Optional additional arguments passed on to the functions.

na.rm  
If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

show.legend  
logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.

inherit.aes  
If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. borders.

...  
other arguments passed on to layer. These are often aesthetics, used to set an aesthetic to a fixed value, like color = "red" or size = 3. They may also be parameters to the paired geom/stat.

**Value**

ggplot2 plot layer

**Examples**

data <- data.frame(x = rep(c(1, 2, 3), each = 20),  
y = rep(c(1, 2, 3), each = 20) + stats::rnorm(60))  
ggplot2::ggplot(data, ggplot2::aes(x = x, y = y)) +  
xgx_geom_ci(conf_level = 0.95)

---

**Description**

Plot data with median and percent intervals

**Usage**

```r
  xgx_geom_pi(  
    mapping = NULL,  
    data = NULL,  
    percent_level = 0.95,  
    geom = list("line", "ribbon"),  
    position = "identity",  
    fun.args = list(),  
    na.rm = FALSE,  
    show.legend = NA,  
    inherit.aes = TRUE,  
    ...  
  )
```

Arguments

mapping Set of aesthetic mappings created by ‘aes’ or ‘aes_’. If specified and ‘inherit.aes = TRUE’ (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.

data The data to be displayed in this layer. There are three options:
If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot.
A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a data.frame., and will be used as the layer data.

percent_level The upper or lower percentile for the percent interval (should fall between 0 and 1). The default is 0.95, which corresponds to (0.05, 0.95) interval. Supplying 0.05 would give the same result

geom Use to override the default geom. Can be a list of multiple geoms, e.g. list("line","ribbon"), which is the default.

position Position adjustment, either as a string, or the result of a call to a position adjustment function.

fun.args Optional additional arguments passed on to the functions.

na.rm If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.

inherit.aes If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. borders.

... other arguments passed on to layer. These are often aesthetics, used to set an aesthetic to a fixed value, like color = "red" or size = 3. They may also be parameters to the paired geom/stat.

Value
ggplot2 plot layer

Examples

```r
data <- data.frame(x = rep(c(1, 2, 3), each = 20),
                   y = rep(c(1, 2, 3), each = 20) + stats::rnorm(60))
ggplot2::ggplot(data, ggplot2::aes(x = x, y = y)) +
               xgx_geom_pi(percent_level = 0.95)
```
xgx_labels_log10  

Nice labels for log10.

Description

Returns a set of labels for ggplot

Usage

xgx_labels_log10(breaks)

Arguments

breaks, breaks for the function

Value

either character or expression

Examples

print(xgx_labels_log10(c(1e-5, 1, 1e5)))

———

xgx_minor_breaks_log10

Sets the default minor_breaks for log10 scales

Description

xgx_minor_breaks_log10 sets nice minor_breaks for log10 scale.

Usage

xgx_minor_breaks_log10(data_range)

Arguments

data_range range of the data

Value

numeric vector of breaks
xgx_plot

Create a new xgx plot

Description

Create a new xgx plot

Usage

xgx_plot(
  data = NULL,
  mapping = ggplot2::aes(),
  ..., 
  environment = parent.frame()
)

Arguments

data Default dataset to use for plot. If not already a data.frame, will be converted to

mapping As in ggplot2; Default list of aesthetic mappings to use for plot. Must define x,

... Other arguments passed on to methods. Not currently used.

environment If an variable defined in the aesthetic mapping is not found in the data, ggplot

Value

ggplot2 object
Examples

time <- rep(seq(1, 10), 5)
id <- sort(rep(seq(1, 5), 10))
conc <- exp(-time) * sort(rep(stats::rlnorm(5), 10))
data <- data.frame(time = time, concentration = conc, id = id)
xgx_plot(data = data,
        mapping = ggplot2::aes(x = time, y = concentration, group = id)) +
        ggplot2::geom_line() +
        ggplot2::geom_point()

Description

Saving plot, automatically annotating the status and denoting the filenames

Usage

xgx_save(
    width,
    height,
    dirs = NULL,
    filename_main = NULL,
    status = "DRAFT",
    g = ggplot2::last_plot(),
    filetype = "png",
    status_x = Inf,
    status_y = Inf,
    status_fontsize = 7
)

Arguments

width        width of plot
height       height of plot
dirs         list of directories. If NULL or if directories missing, there is default behavior below

1. parent_dir = Parent directory containing the Rscript and the Results folder, default getwd()
2. rscript_dir = Subdirectory of parent_dir that contains the Rscript used to generate the figure, default "./"
3. rscript_name= Name of the Rscript used to generate the figure, default "Name_Of_Script_Here.R"
4. results_dir = Subdirectory of parent_dir where the figure is stored, default "/"
5. filename_prefix = prefix of filename to be appended to filename_main

filename_main  main part of the filename, excluding prefix and suffix. no default
status  status to be annotated
g  ggplot plot object, default is ggplot::last_plot()
filetype  file extension (e.g. "pdf","csv" etc.)
status_x  x location of the status in plot
status_y  y location of the status in plot
status_fontsize  font size for status in plot

Value

ggplot2 plot object

Examples

directory = tempdir()
dirs <- list(parent_dir = directory,
rscript_dir = directory,
rscript_name = "example.R",
results_dir = directory,
filename_prefix = "example_")
data <- data.frame(x = 1:1000, y = stats::rnorm(1000))
ggplot2::ggplot(data = data, ggplot2::aes(x = x, y = y)) +
ggplot2::geom_point()
xgx_save(4, 4, dirs, "Example", "DRAFT")

xgx_save_table  Saving table as an image, also labeling the program that created the table and where the table is stored

Description

Saving table as an image, also labeling the program that created the table and where the table is stored

Usage

xgx_save_table(data, dirs = NULL, filename_main = NULL)
xgx_scale_x_log10

Arguments

- `data` : data.frame or table of results
- `dirs` : list of directories. If NULL or if directories missing, there is default behavior below
  1. `parent_dir` = Parent directory containing the Rscript and the Results folder, default getwd()
  2. `rscript_dir` = Subdirectory of parent_dir that contains the Rscript used to generate the figure, default "./"
  3. `rscript_name` = Name of the Rscript used to generate the figure, default "Name_Of_Script_Here.R"
  4. `results_dir` = Subdirectory of parent_dir where the figure is stored, default "/"
  5. `filename_prefix` = prefix of filename to be appended to filename_main
- `filename_main` : main part of the filename, excluding prefix and extension. no default

Value

- ggplot2 plot object

Examples

```r
directory = tempdir()
dirs <- list(parent_dir = directory,
             rscript_dir = directory,
             rscript_name = "example.R",
             results_dir = directory,
             filename_prefix = "example_")
data <- data.frame(x = c(1, 2), y = c(1, 2))
xgx_save_table(data, dirs = dirs, filename_main = "test")
```

---

**xgx_scale_x_log10**

$log_{10}$ scales the x axis with a "pretty" set of breaks

Description

`xgx_scale_x_log10` is similar to `scale_x_log10`. But it uses what we believe to be a nicer spacing and set of tick marks it can be used the same as `scale_x_log10`

Usage

```r
xgx_scale_x_log10(
  breaks = xgx_breaks_log10,
  minor_breaks = NULL,
  labels = xgx_labels_log10,
  ...
)
```
Arguments

- `breaks` major breaks, default is a function defined here
- `minor_breaks` minor breaks, default is a function defined here
- `labels` function for setting the labels, defined here
- `...` other arguments passed to `scale_x_log10`

Value

ggplot2 compatible scale object

Examples

```r
conc <- 10^seq(-3, 3, by = 0.1)
ec50 <- 1
data <- data.frame(concentration = conc,
                     bound_receptor = 1 * conc / (conc + ec50))
ggplot2::ggplot(data, ggplot2::aes(x = concentration, y = bound_receptor)) +
ggplot2::geom_point() +
ggplot2::geom_line() +
xgx_scale_x_log10() +
xgx_scale_y_reverse_log10()
```

---

**xgx_scale_x_reverse_log10**

Reverse-log transform for the x scale.

Description

`xgx_scale_x_reverse_log10` is designed to be used with data that approaches 100. A common example is receptor occupancy in drug development. It is used when you want even spacing between 90, 99, 99.9, etc.

Usage

```r
xgx_scale_x_reverse_log10(labels = NULL, accuracy = NULL, ...)
```

Arguments

- `labels` if NULL, then the default is to use `scales::percent()`
- `accuracy` if NULL, then use the the default as specified by `scales::percent()` to round to the hundredths place, set accuracy 0.01
- `...` other parameters passed to `scale_x_continuous`
Value

- ggplot2 compatible scale object

Examples

```r
conc <- 10^seq(-3, 3, by = 0.1)
ec50 <- 1
data <- data.frame(concentration = conc,
                    bound_receptor = 1 * conc / (conc + ec50))
ggplot2::ggplot(data, ggplot2::aes(y = concentration, x = bound_receptor)) +
ggplot2::geom_point() +
ggplot2::geom_line() +
xgx_scale_y_log10() +
xgx_scale_x_reverselog10()
```

---

**xgx_scale_x_time_units**

*Convert time units for plotting*

Description

*xgx_scale_x_time_units* converts x axis scale from one time unit to another. Supported units include hours, days, weeks, months, and years, which can also be called using just the first letter (h, d, w, m, y).

Usage

```r
xgx_scale_x_time_units(
  units_dataset,
  units_plot = NULL,
  breaks = NULL,
  labels = NULL,
  ...
)
```

Arguments

- **units_dataset** units of the input dataset, must be specified by user as "h", "d", "w", "m", or "y"
- **units_plot** units of the plot, will be units of the dataset if empty
- **breaks** One of:
  - NULL for no breaks
  - waiver() for the default breaks computed by the transformation object
  - A numeric vector of positions
  - A function that takes the limits as input and returns breaks as output (e.g., a function returned by *scales::extended_breaks()*
- **labels**
labels

One of:
- `NULL` for no labels
- `waiver()` for the default labels computed by the transformation object
- A character vector giving labels (must be same length as breaks)
- A function that takes the breaks as input and returns labels as output

... other parameters for `scale_x_continuous`

Details

Note: `xgx_scale_x_time_units` only scales the plot axis, all other specifications must be on the original scale of the dataset (e.g. breaks, position, width)

Value

ggplot2 compatible scale object

Examples

data <- data.frame(x = 1:1000, y = rnorm(1000))
ggplot2::ggplot(data = data, ggplot2::aes(x = x, y = y)) +
ggplot2::geom_point() +
xgx_scale_x_time_units(units_dataset = "hours", units_plot = "weeks")

| xgx_scale_y_log10 | log10 scales the y axis with a "pretty" set of breaks |

Description

`xgx_scale_y_log10` is similar to `scale_y_log10`. But it uses what we believe to be a nicer spacing and set of tick marks it can be used the same as `scale_y_log10`

Usage

```r
xgx_scale_y_log10(
  breaks = xgx_breaks_log10,
  minor_breaks = NULL,
  labels = xgx_labels_log10,
  ...
)
```

Arguments

- `breaks`: major breaks, default is a function defined here
- `minor_breaks`: minor breaks, default is a function defined here
- `labels`: function for setting the labels, defined here
- `...`: other arguments passed to `scale_y_log10`
Value

ggplot2 compatible scale object

Examples

```r
cconc <- 10^(seq(-3, 3, by = 0.1))
cec50 <- 1
data <- data.frame(concentration = conc,
                   bound_receptor = 1 * conc / (conc + ec50))
ggplot2::ggplot(data, ggplot2::aes(y = concentration, x = bound_receptor)) +
ggplot2::geom_point() +
ggplot2::geom_line() +
xgx_scale_y_log10() +
xgx_scale_x_reverselog10()
```

Description

xgx_scale_y_percentchangelog10 is designed to be used with percent change (PCHG) from baseline data (on a scale of -1 to +Inf). Common examples include. It is used when you have a wide range of data on a percent change scale, especially data close to -100.

Usage

```r
xgx_scale_y_percentchangelog10(
  breaks = NULL,
  minor_breaks = NULL,
  labels = NULL,
  accuracy = 1,
  n_breaks = 7,
  ...
)
```

Arguments

- **breaks**: if NULL, then default is to use a variant of \(2^{(\text{labeling::extended(log2(PCHG + 1))))} - 1\), where PCHG represents the range of the data
- **minor_breaks**: if NULL, then default is to use nicely spaced \(\log 10(PCHG + 1)\) minor breaks
- **labels**: if NULL, then the default is to use scales::percent_format()
- **accuracy**: accuracy to use with scales::percent_format(), if NULL, then the default is set to 1
- **n_breaks**: number of desired breaks, if NULL, then the default is set to 7
- **...**: other parameters passed to `scale_y_continuous`
Value

ggplot2 compatible scale object

Examples

dat1 <- data.frame(x = rnorm(100), PCHG = exp(rnorm(100)) - 1)
ggplot2::ggplot(dat1, ggplot2::aes(x = x, y = PCHG)) +
ggplot2::geom_point() +
xgx_theme() +
xgx_scale_y_percentchangelog10()

xgx_scale_y_reverselog10

Reverselog transform for the y scale.

Description

xgx_scale_y_reverselog10 is designed to be used with data that approaches 100 A common example is receptor occupancy in drug development. It is used when you want even spacing between 90, 99, 99.9, etc.

Usage

xgx_scale_y_reverselog10(labels = NULL, accuracy = NULL, ...)

Arguments

labels if NULL, then the default is to use scales::percent()
accuracy if NULL, then use the the default as specified by scales::percent() to round to the hundredths place, set accuracy 0.01
...
other parameters passed to scale_y_continuous

Value

ggplot2 compatible scale object

Examples

conc <- 10^seq(-3, 3, by = 0.1))
ec50 <- 1
data <- data.frame(concentration = conc,
                   bound_receptor = 1 * conc / (conc + ec50))
ggplot2::ggplot(data, ggplot2::aes(x = concentration, y = bound_receptor)) +
ggplot2::geom_point() +
ggplot2::geom_line() +
xgx_stat_ci

xgx_scale_x_log10() +
xgx_scale_y_reverselog10()

---

xgx_stat_ci  Plot data with mean and confidence intervals

Description

xgx_stat_ci returns a ggplot layer plotting mean +/- confidence intervals

Usage

xgx_stat_ci(
  mapping = NULL,
  data = NULL,
  conf_level = 0.95,
  distribution = "normal",
  geom = list("point", "line", "errorbar"),
  position = "identity",
  fun.args = list(),
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)

Arguments

mapping  Set of aesthetic mappings created by ‘aes’ or ‘aes_’. If specified and ‘inherit.aes = TRUE’ (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.

data    The data to be displayed in this layer. There are three options:
            If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot.
            A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify for which variables will be created.
            A function will be called with a single argument, the plot data. The return value must be a data.frame., and will be used as the layer data.

conf_level  The percentile for the confidence interval (should fall between 0 and 1). The default is 0.95, which corresponds to a 95 percent confidence interval.

distribution  The distribution which the data follow, used for calculating confidence intervals. The options are "normal", "lognormal", and "binomial". The "normal" option will use the Student t Distribution to calculate confidence intervals, the "lognormal" option will transform data to the log space first. The "binomial" option will use the binom.exact function to calculate the confidence intervals. Note: binomial data must be numeric and contain only 1’s and 0’s.
geom	Use to override the default geom. Can be a list of multiple geoms, e.g. list("point","line","errorbar"), which is the default.

group	Position adjustment, either as a string, or the result of a call to a position adjustment function.

fun.args	Optional additional arguments passed on to the functions.

na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.

inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. borders.

... 

other arguments passed on to layer. These are often aesthetics, used to set an aesthetic to a fixed value, like color = "red" or size = 3. They may also be parameters to the paired geom/stat.

Details

This function can be used to generate mean +/- confidence interval plots for different distributions, and multiple geoms with a single function call.

Value

ggplot2 plot layer

Examples

# default settings for normally distributed data, 95% confidence interval,
data <- data.frame(x = rep(c(1, 2, 3), each = 20),
y = rep(c(1, 2, 3), each = 20) + stats::rnorm(60),
  group = rep(1:3, 20))
xgx_plot(data, ggplot2::aes(x = x, y = y)) +
xgx_stat_ci(conf_level = 0.95)

# try different geom
xgx_plot(data, ggplot2::aes(x = x, y = y)) +
xgx_stat_ci(conf_level = 0.95, geom = list("ribbon", "point", "line"))

# plotting lognormally distributed data
data <- data.frame(x = rep(c(1, 2, 3), each = 20),
y = 10^((rep(c(1, 2, 3), each = 20) + stats::rnorm(60)),
  group = rep(1:3, 20))
xgx_plot(data, ggplot2::aes(x = x, y = y)) +
xgx_stat_ci(conf_level = 0.95, distribution = "lognormal")

# note: you DO NOT need to use both distribution = "lognormal"
# and scale_y_log10()
xgx_plot(data, ggplot2::aes(x = x, y = y)) +
xgx_stat_ci(conf_level = 0.95) + xgx_scale_y_log10()
# plotting binomial data

data <- data.frame(x = rep(c(1, 2, 3), each = 20),
                   y = stats::rbinom(60, 1, rep(c(0.2, 0.6, 0.8), each = 20)),
                   group = rep(1:3, 20))

xgx_plot(data, ggplot2::aes(x = x, y = y)) +
          xgx_stat_ci(conf_level = 0.95, distribution = "binomial")

# including multiple groups in same plot

xgx_plot(data, ggplot2::aes(x = x, y = y)) +
          xgx_stat_ci(conf_level = 0.95, distribution = "binomial",
                      ggplot2::aes(color = factor(group)),
                      position = ggplot2::position_dodge(width = 0.5))

---

**xgx_stat_pi**

Plot data with median and percent intervals

**Description**

xgx_stat_pi returns a ggplot layer plotting median +/- percent intervals

**Usage**

```r
xgx_stat_pi(
    mapping = NULL,
    data = NULL,
    percent_level = 0.95,
    geom = list("line", "ribbon"),
    position = "identity",
    fun.args = list(),
    na.rm = FALSE,
    show.legend = NA,
    inherit.aes = TRUE,
    ...
)
```

**Arguments**

- **mapping**: Set of aesthetic mappings created by ‘aes’ or ‘aes_’. If specified and ‘inherit.aes = TRUE’ (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.

- **data**: The data to be displayed in this layer. There are three options:
  - If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot.
  - A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data.

percent_level  The upper or lower percentile for the percent interval (should fall between 0 and 1). The default is 0.95, which corresponds to (0.05, 0.95) interval. Supplying 0.05 would give the same result.

geom  Use to override the default geom. Can be a list of multiple geoms, e.g. list("line", "ribbon"), which is the default.

position  Position adjustment, either as a string, or the result of a call to a position adjustment function.

fun.args  Optional additional arguments passed on to the functions.

na.rm  If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

show.legend  logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.

inherit.aes  If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. borders.

...  other arguments passed on to layer. These are often aesthetics, used to set an aesthetic to a fixed value, like color = "red" or size = 3. They may also be parameters to the paired geom/stat.

Value

ggplot2 plot layer

Examples

# default settings for normally distributed data, (5%, 95%) interval,
data <- data.frame(x = rep(c(1, 2, 3), each = 20),
y = rep(c(1, 2, 3), each = 20) + stats::rnorm(60),
group = rep(1:3, 20))
xgx_plot(data, ggplot2::aes(x = x, y = y)) +
xgx_stat_pi(percent_level = 0.95)

# try different geom
xgx_plot(data, ggplot2::aes(x = x, y = y)) +
xgx_stat_pi(percent_level = 0.95, geom = list("errorbar", "point", "line"))

# including multiple groups in same plot
xgx_plot(data, ggplot2::aes(x = x, y = y)) +
xgx_stat_pi(percent_level = 0.95,
            ggplot2::aes(color = factor(group), fill = factor(group)),
            position = ggplot2::position_dodge(width = 0.5))

# including multiple percent intervals in same plot
xgx_plot(data, ggplot2::aes(x = x, y = y)) +
xgx_stat_pi(percent_level = 0.90) +
xgx_stat_pi(percent_level = 0.80) +
### xgx_summarize_covariates

Summarize Covariate information in a dataset

**Description**

xgx_summarize_covariates

**Usage**

```r
xgx_summarize_covariates(data, covariates = NULL, n_cts = 8)
```

**Arguments**

- `data`, the dataset to check. must contain a USUBJID or ID column for subject id
- `covariates`, the column names of covariates, to explore
- `n_cts`, the number of unique values for a covariate to be treated as continuous, default is 8

**Value**

list

**Examples**

```r
data <- data.frame(ID = 1:10, WT0 = rnorm(10, 70, 10), SEX = round(runif(10)))
x <- xgx_summarize_covariates(data, c("WT0", "SEX"))
```

---

### xgx_summarize_data

Check data for various issues

**Description**

Calls `xgx_check_data`

**Usage**

```r
xgx_summarize_data(data, covariates = NULL)
```
Arguments

data the dataset to check. Must contain the above columns
covariates the column names of covariates, to explore

Value
data.frame

Examples

covariates <- c("WEIGHTB", "SEX")
check <- xgx_summarize_data(mad_missing_duplicates, covariates)

xgx_theme

Calls the standard theme for xGx graphics

Description

Calls the standard theme for xGx graphics

Usage

xgx_theme()

Value

xgx ggplot2 compatible theme

Examples

conc <- 10^seq(-3, 3, by = 0.1))
ec50 <- 1
data <- data.frame(concentration = conc,
  bound_receptor = 1 * conc / (conc + ec50))
ggplot2::ggplot(data, ggplot2::aes(y = concentration, x = bound_receptor)) +
ggplot2::geom_point() +
ggplot2::geom_line() +
xgx_scale_y_log10() +
xgx_scale_x_reverselog10() +
xgx_theme()
Description

\texttt{xgx_theme_set}

Usage

\texttt{xgx_theme_set()}

Value

\texttt{xgx ggplot2 compatible theme}

Examples

\begin{verbatim}
conc <- 10^(seq(-3, 3, by = 0.1))
ce50 <- 1
data <- data.frame(concentration = conc,
                   bound_receptor = 1 * conc / (conc + ec50))
xgx_theme_set()
\texttt{ggplot2::ggplot(data, ggplot2::aes(y = concentration, x = bound_receptor)) +}
\texttt{ggplot2::geom_point() +}
\texttt{ggplot2::geom_line() +}
\texttt{xgx_scale_y_log10() +}
\texttt{xgx_scale_x_reverselog10()}
\end{verbatim}
Index

*Topic datasets
  case1_pkpd, 3
  mad, 4
  mad_missing_duplicates, 5
  mad_nca, 5
  nlmixr_theo_sd, 6
  sad, 6

binom.exact, 16, 29

case1_pkpd, 3

ggplot, 20

layer, 10

mad, 4
mad_missing_duplicates, 4
mad_nca, 5

nlmixr_theo_sd, 6

sad, 6

scale_x_continuous, 24, 26
scale_x_log10, 23, 24
scale_y_continuous, 27, 28
scale_y_log10, 26
scales::extended_breaks(), 25

theme_xgx, 7
transformation object, 25

xgx_annotate_filenames, 8
xgx_annotate_status, 9
xgx_annotate_status_png, 10
xgx_breaks_log10, 12
xgx_breaks_time, 13
xgx_check_data, 14, 33
xgx_dirs2char, 15
xgx_geom_ci, 16
xgx_geom_pi, 17

xgx_labels_log10, 19
xgx_minor_breaks_log10, 19
xgx_plot, 20
xgx_save, 21
xgx_save_table, 22
xgx_scale_x_log10, 23
xgx_scale_x_reverselog10, 24
xgx_scale_x_time_units, 25
xgx_scale_y_log10, 26
xgx_scale_y_percentchangelog10, 27
xgx_scale_y_reverselog10, 28
xgx_stat_ci, 29
xgx_stat_pi, 31
xgx_summarize_covariates, 33
xgx_summarize_data, 33
xgx_theme, 34
xgx_theme_set, 35