Title Exploratory Graphics for Pharmacometrics
Version 1.1.1
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, Deriv, DescTools, dplyr, ggplot2, glue, graphics, grDevices, gtable, Hmisc, labeling, magrittr, minpack.lm, pander, png, RCurl, readr, scales, stats, stringr, tibble, utils
Suggests caTools, gridExtra, knitr, pkgdown, rmarkdown, testthat, tidyverse
VignetteBuilder knitr
Encoding UTF-8
LazyData true
RoxygenNote 7.1.1
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  2021-04-22 19:20:02 UTC

R topics documented:

case1_pkpd ................................................................. 3
edit_rmd_template_str .................................................. 4
get_rmd_name ............................................................... 5
get_rmd_str ................................................................. 6
mad ................................................................. 7
mad_missing_duplicates .................................................. 7
mad_nca ................................................................. 8
nlmixr_theo_sd .............................................................. 9
predict.nls ................................................................. 9
predictdf ................................................................. 11
predictdf.nls ............................................................... 12
predictdf.polr .............................................................. 13
sad ................................................................. 14
StatSmoothOrdinal ........................................................ 14
StatSummaryBinQuant ...................................................... 15
StatSummaryOrdinal ....................................................... 15
theme_xgx ................................................................. 16
xgx_annotate_filenames .................................................. 16
xgx_annotate_status ...................................................... 17
xgx_annotate_status.png .................................................. 18
xgx_auto_explore .......................................................... 20
xgx_breaks_log10 .......................................................... 23
xgx_breaks_time ........................................................... 24
xgx_check_data ............................................................. 25
xgx_conf_int ................................................................. 26
xgx_dirs2char ............................................................... 27
xgx_geom_ci ................................................................. 28
xgx_geom_pi ................................................................. 29
xgx_labels_log10 ............................................................. 31
xgx_minor_breaks_log10 .................................................... 31
xgx_plot ................................................................. 32
xgx_save ................................................................. 33
xgx_save_table ............................................................. 35
xgx_scale_x_log10 .......................................................... 36
xgx_scale_x_reverselog10 ................................................... 37
xgx_scale_x_time_units ..................................................... 38
xgx_scale_y_log10 .......................................................... 39
xgx_scale_y_percentchangelog10 ........................................... 40
xgx_scale_y_reverselog10 ................................................... 41
xgx_stat_ci ................................................................. 42
xgx_stat_pi ................................................................. 45
**Description**

Case 1 PKPD Data Set

**Usage**

```r
case1_pkpd
```

**Format**

A data frame with the following 21 columns:

- **column 1**: `ID` integer; unique subject ID
- **column 2**: `TIME` numeric; time relative to first drug administration
- **column 3**: `NOMTIME` numeric; nominal time
- **column 4**: `TIMEUNIT` factor; unit of TIME
- **column 5**: `AMT` integer; 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**: `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 8**: `NAME` factor; description of event
- **column 9**: `EVENTU` factor; unit for observation
- **column 10**: `CENS` integer; censored values (0 = not censored, 1 = censored)
- **column 11**: `EVID` integer; event ID (0 = observation, 1 = dosing event)
- **column 12**: `WEIGHTB` numeric; baseline body weight (kg)
- **column 13**: `EFF0` numeric; efficacy
- **column 14**: `TRTACT` factor; treatment group label
- **column 15**: `DOSE` integer; Dose in mg
- **column 16**: `PROFDAY` integer; day of profile
- **column 17**: `PROFTIME` numeric; time within PROFDAY
- **column 18**: `CYCLE` integer; count of drug administrations received
- **column 19**: `PART` integer; part of study
Description

edit_rmd_template_str returns a path to the altered Rmd template

Usage

edit_rmd_template_str(
  rmd_str = NULL,
  mapping = NULL,
  rmd_output_path = NULL,
  data_path = NULL,
  multiple_dosing = FALSE,
  pk_cmt = NULL,
  pd_cmt = NULL,
  dose_cmt = NULL,
  steady_state_day = NULL,
  time_between_doses = NULL,
  author_name = NULL,
  add_datetime = TRUE,
  show_explanation = TRUE
)

Arguments

rmd_str A character string containing the Rmd template raw characters
mapping A list of column name mappings from the original (template) dataset column names to the corresponding columns in the new dataset
rmd_output_path A custom output path for the generated Rmd file (This is typically left as ‘NULL’ in order to maintain the hierarchical directory structure of ‘xgx_autoexplore_output’)
data_path Path (as a string) to the dataset that is to be analyzed
multiple_dosing if FALSE use single ascending dose template, if TRUE use multiple
pk_cmt An integer denoting the "compartment" containing the PK data. The "CMT" column will typically have these integers, where each row may contain either PK or PD data, potentially of different types (continuous, ordinal, etc.)
pd_cmt An integer denoting the "compartment" containing the PD data, of the desired type (continuous, ordinal, etc.). The "CMT" column will typically have these integers, where each row may contain either PK or PD data
get_rmd_name

- `dose_cmt`: CMT associated with dosing event
- `steady_state_day`: For multiple ascending dose, what day is steady state rich profile?
- `time_between_doses`: time interval between doses
- `author_name`: The name of the author to be displayed on the template
- `add_datetime`: Boolean indicating addition of a date stamp to the beginning of the Rmd file
- `show_explanation`: Boolean indicating if the additional explanations (text in between figures) are needed for the user.

**Value**
A string of the new R markdown template

---

**get_rmd_name**

*Determine the name of a Rmd template*

**Description**
get_rmd_name returns a name for an Rmd template, based on the desired PKPD parameters

**Usage**

```
get_rmd_name(
  rmd_template_name = NULL,
  multiple_dosing = FALSE,
  pk_cmt = NULL,
  pd_cmt = NULL,
  pd_data_type = NULL
)
```

**Arguments**

- `rmd_template_name`: A custom output name for the generated Rmd file
- `multiple_dosing`: if FALSE use single ascending dose template, if TRUE use multiple
- `pk_cmt`: An integer denoting the "compartment" containing the PK data. The "CMT" column will typically have these integers, where each row may contain either PK or PD data, potentially of different types (continuous, ordinal, etc.)
- `pd_cmt`: An integer denoting the "compartment" containing the PD data, of the desired type (continuous, ordinal, etc.). The "CMT" column will typically have these integers, where each row may contain either PK or PD data
- `pd_data_type`: The type of PD data - acceptable values exist in the following list: ["binary","continuous","count","ordinal",...]

---
get_rmd_str

get_rmd_str returns a Rmd template string, based on the desired PKPD parameters

Description

get_rmd_str returns a Rmd template string, based on the desired PKPD parameters

Usage

get_rmd_str(
  rmd_template_name = NULL,
  multiple_dosing = FALSE,
  pk_cmt = NULL,
  pd_cmt = NULL,
  pd_data_type = NULL
)

Arguments

rmd_template_name
  A custom output name for the generated Rmd file

multiple_dosing
  if FALSE use single ascending dose template, if TRUE use multiple

pk_cmt
  An integer denoting the "compartment" containing the PK data. The "CMT"
column will typically have these integers, where each row may contain either
PK or PD data, potentially of different types (continuous, ordinal, etc.)

pd_cmt
  An integer denoting the "compartment" containing the PD data, of the desired
type (continuous, ordinal, etc.). The "CMT" column will typically have these
integers, where each row may contain either PK or PD data

pd_data_type
  The type of PD data - acceptable values exist in the following list: ["binary","continuous","count","ordinal"

Value

a string for the Rmd template name
**mad_missing_duplicates**

---

**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**: WEIGHT 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
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
**predict.nls**

**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**  *nlmixr Theophylline SD Data Set*

**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
**Description**

predict.nls

**Usage**

```r
## S3 method for class 'nls'
predict(
  object,
  newdata = NULL,
  se.fit = FALSE,
  interval = "none",
  level = 0.95,
  ...
)
```

**Arguments**

- `object`: Object of class inheriting from "nls"
- `newdata`: An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
- `se.fit`: A switch indicating if standard errors are required.
- `interval`: Type of interval calculation, "none" or "confidence"
- `level`: Level of confidence interval to use
- `...`: additional arguments affecting the predictions produced.

**Value**

`predict.nls` produces a vector of predictions or a matrix of predictions and bounds with column names `fit`, `lwr`, and `upr` if interval is set.

If `se.fit` is TRUE, a list with the following components is returned:

- `fit`: vector or matrix as above
- `se.fit`: standard error of predicted means
- `residual.scale`: residual standard deviations
- `df`: degrees of freedom for residual

**Examples**

```r
set.seed(12345)
data_to_plot <- data.frame(x1 = rep(c(0, 25, 50, 100, 200, 400, 600), 10)) %>%
dplyr::mutate(AUC = x1*rlnorm(length(x1), 0, 0.3),
  x2 = x1*stats::rlnorm(length(x1), 0, 0.3),
  Response = (15 + 50*x2/(20+x2))*stats::rlnorm(length(x2), 0, 0.3))

gg <- ggplot2::ggplot(data = data_to_plot, ggplot2::aes(x = AUC, y = Response)) +
```
predictdf

Prediction data frame from ggplot2 Get predictions with standard errors into data frame

Description

Prediction data frame from ggplot2 Get predictions with standard errors into data frame

Usage

predictdf(model, xseq, se, level)

Arguments

model  model object
xseq   newdata
se     Display confidence interval around smooth?
level  Level of confidence interval to use
predictdf.nls  

**Prediction data frame for nls**

**Description**

Get predictions with standard errors into data frame for use with `geom_smooth`

**Usage**

```r
## S3 method for class 'nls'
predictdf(model, xseq, se, level)
```

**Arguments**

- `model`  
  nls object
- `xseq`  
  newdata
- `se`  
  Display confidence interval around smooth?
- `level`  
  Level of confidence interval to use

**Details**

`ggplot2::geom_smooth` produces confidence intervals by silently calling functions of the form `predictdf.method`, where method is "loess", "lm", "glm" etc. depending on what method is specified in the call to `geom_smooth`. Currently `ggplot2` does not define a `predictdf.nls` function for method of type "nls", and thus confidence intervals cannot be automatically generated by `geom_smooth` for method = "nls". Here we define `predictdf.nls` for calculating the confidence intervals of an object of type nls. `geom_smooth` will silently call this function whenever method = "nls", and produce the appropriate confidence intervals.

`predictdf.nls` calculates CI for a model fit of class `nls` based on the "delta-method" http://sia.webpopix.org/nonlinearRegression.html#confidence-intervals-and-prediction-intervals

\[
CI = [ f(x_0, \beta) + qt(\alpha / 2, n - d) \times se(f(x_0, \beta)), f(x_0, \beta) + qt(1 - \alpha / 2, n - d) \times se(f(x_0, \beta)) ]
\]

where: \( \beta \) = vector of parameter estimates \( x \) = independent variable \( se(f(x_0, \beta)) = \sqrt{\text{delta}(f)(x_0, \beta) \times \text{Var}(\beta) \times (\text{delta}(f)(x_0, \beta))'} \) \( \text{delta}(f) \) is the gradient of \( f \)

**Value**

dataframe with x and y values, if `se` is TRUE dataframe also includes `ymin` and `ymax`
**predictdf.polr**

**Prediction data frame for polr**

---

**Description**

Get predictions with standard errors into data frame for use with geom_smooth

**Usage**

```r
## S3 method for class 'polr'
predictdf(  
  model,  
  xseq,  
  se,  
  level,  
  data,  
  method,  
  formula,  
  method.args,  
  weight,  
  n_boot = 200  
)
```

**Arguments**

- `model`: object returned from polr
- `xseq`: sequence of x values for which to compute the smooth
- `se`: if TRUE then confidence intervals are returned
- `level`: confidence level for confidence intervals
- `data`: data to fit
- `method`: only works for method MASS::polr
- `formula`: formula to fit
- `method.args`: arguments to pass to method
- `weight`: weights to use for method
- `n_boot`: number of bootstraps to perform for confidence interval calculation, default is 200

**Details**

`predictdf.polr` is used by `xgx_geom_smooth` when method = "polr" to calculate confidence intervals via bootstraps.
sad

Single Ascending Dose Data Set

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:

- **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
  
  (1 if missing, 0 otherwise)
- **column 8:** CMT integer; compartment number (determines observation type):
  CMT 1 = Dosing event
  CMT 2 = PK concentration
- **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 received

StatSmoothOrdinal

Stat object for producing smooths through ordinal data

Description

Stat object for producing smooths through ordinal data

Usage

StatSmoothOrdinal
StatSummaryBinQuant

Format

An object of class StatSmoothOrdinal (inherits from Stat, ggproto, gg) of length 8.

StatSummaryBinQuant  Stat ggproto object for binning by quantile for xgx_stat_ci

Description


Usage

StatSummaryBinQuant

Format

An object of class StatSummaryBinQuant (inherits from Stat, ggproto, gg) of length 5.

Details

StatSummaryBinQuant returns a ggproto object for plotting mean +/- confidence bins

Value

ggplot2 ggproto object

StatSummaryOrdinal  Stat ggproto object for creating ggplot layers of binned confidence intervals for probabilities of classes in ordinal data

Description

StatSummaryOrdinal returns a ggproto object for plotting mean +/- confidence intervals for ordinal data. It also allows for binning values on the independent axis.

Usage

StatSummaryOrdinal

Format

An object of class StatSummaryOrdinal (inherits from Stat, ggproto, gg) of length 8.

Value

ggplot2 ggproto object
theme_xgx  

**Calls the standard theme for xGx graphics**

**Description**

Calls the standard theme for xGx graphics

**Usage**

```r
theme_xgx()
```

**Value**

xgx ggplot2 compatible theme

**Examples**

```r
cconc <- 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_reverse_log10() +
theme_xgx()
```

---

xgx_anonotate_filenames  

**Append filenames to bottom of the plot**

**Description**

xgx_anonotate_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**

```r
xgx_anonotate_filenames(dirs, hjust = 0.5, color = "black", size = 11)
```
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
color font color for caption, default black
size font size for caption, default 11

Value
None

Examples

defs <- 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)


xgx_annotate_status Create a status (e.g. DRAFT) annotation layer

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 font 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_anntate_status("DRAFT")

---

xgx_anntate_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

```r
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.
- `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
# 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")

Description

xgx_auto_explore returns an HTML and PDF document with plots describing the provided dataset

Usage

xgx_auto_explore(
  data_path = NULL,
  mapping = list(),
  author_name = NULL,
  multiple_dosing = FALSE,
  pk_cmt = NULL,
  pd_cmt = NULL,
  pd_data_type = NULL,
  dose_cmt = NULL,
  steady_state_day = NULL,
  time_between_doses = NULL,
  rmd_template_name = NULL,
  rmd_template_path = NULL,
  rmd_output_path = NULL,
  pdf_output_path = NULL,
  html_output_path = NULL,
  add_datetime = TRUE,
  show_explanation = TRUE
)
Arguments

data_path Path (as a string) to the dataset that is to be analyzed

mapping A list of column name mappings from the original (template) dataset column names to the corresponding columns in the new dataset.

author_name The name of the author to be displayed on the template

multiple_dosing Whether or not to use a "Multiple" or "Single" Ascending dose template

pk_cmt An integer denoting the "compartment" containing the PK data. The "CMT" column will typically have these integers, where each row may contain PK, PD, dosing or other events/observations data

pd_cmt An integer denoting the "compartment" containing the PD data, of the desired type (continuous, ordinal, etc.). The "CMT" column will typically have these integers, where each row may contain PK, PD, dosing or other events/observations data

pd_data_type The type of PD data - acceptable values exist in the following list: ["binary","continuous","count","ordinal","real_example","receptor_occupancy","time_to_event"]

dose_cmt Integer denoting the compartment for dosing records

steady_state_day used to denote the day of rich sampling of PK at steady state

time_between_doses dosing interval, has units to match the time variable of the dataset

rmd_template_name A custom output name for the generated Rmd file

rmd_template_path A user provided custom template (as a string)

rmd_output_path A custom output path for the generated Rmd file (This is typically left as 'NULL' in order to maintain the hierarchical directory structure of 'xgx_autoexplore_output')

pdf_output_path A custom output path for the generated PDF file (This is typically left as 'NULL' in order to maintain the hierarchical directory structure of 'xgx_autoexplore_output')

html_output_path A custom output path for the generated HTML file (This is typically left as 'NULL' in order to maintain the hierarchical directory structure of 'xgx_autoexplore_output')

add_datetime Boolean indicating addition of a date stamp to the beginning of the Rmd file

show_explanation Boolean indicating if the additional explanations (text in between figures) are needed for the user.

Details

This function can be used quickly to explore your data by generating overview plots before constructing non-linear mixed effects models.
Examples

```r
author_name = "Your Name Here"
show_explanation = FALSE

## Not run:
# Try out the nonlinear_pkpd dataset with the
# Multiple Ascending Dose PK Rmd template
data_path <- "~/nonlinear_pkpd.csv"

# Specify the mapping of column names
mapping <- list(
  "TIME" = "TIM2",
  "NOMTIME" = "NT",
  "EVID" = 0,
  "CENS" = 0,
  "DOSE" = "MGKG",
  "TRTACT" = "TRT",
  "LIDV_NORM" = "LIDV/MGKG",
  "LIDV_UNIT" = "UNIT",
  "PROFDAY" = 1,
  "SEX" = 0,
  "WEIGHTB" = 0)

# 5 contains the PK Concentration in this dataset
pk_cmt = 5
# We don't need PD right now
pd_cmt = NULL
pd_data_type = NULL

dose_cmt = 1
steady_state_day = c(0, 6)
time_between_doses = 24
multiple_dosing = TRUE

output_directory = tempdir()
xgx_auto_explore(data_path = data_path,
  mapping = mapping,
  author_name = author_name,
  pk_cmt = pk_cmt,
  pd_cmt = pd_cmt,
  dose_cmt = dose_cmt,
  steady_state_day = steady_state_day,
  time_between_doses = time_between_doses,
  multiple_dosing = multiple_dosing,
  pd_data_type = pd_data_type,
  rmd_output_path = output_directory,
  show_explanation = show_explanation)
```
### 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

```r
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

```r
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, 1.01))
xgx_breaks_log10(c(1, 1.0001))
print(xgx_breaks_log10(c(1, 1.000001)), digits = 10)

---

**xgx_breaks_time**  
*Sets the default breaks for a time axis*

**Description**

*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**

xgx_breaks_time(data_range, units_plot, number_breaks = 5)

**Arguments**

- `data_range` range of the data
- `units_plot` units to use in the plot
- `number_breaks` number of breaks to aim for (default is 5)

**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

```r
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")
xgx_breaks_time(c(1, 50), "d")
xgx_breaks_time(c(1000, 3000), "d")
xgx_breaks_time(c(-21, 100), "d")
xgx_breaks_time(c(-1, 10), "w")
```

---

**xgx_check_data**  
**Check data for various issues**

**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**

```r
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

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

xgx_conf_int returns a dataframe with mean +/- confidence intervals

Description

xgx_conf_int returns a dataframe with mean +/- confidence intervals

Usage

xgx_conf_int(y, conf_level = 0.95, distribution = "normal")

Arguments

y data to compute confidence interval of
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.

Value

data.frame

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_conf_int(data$y)
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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>dirs</code></td>
<td>list containing directories and filenames. It must contain five fields</td>
</tr>
<tr>
<td></td>
<td>1. <code>parent_dir</code> = Parent directory containing the Rscript and the Results folder</td>
</tr>
<tr>
<td></td>
<td>2. <code>rscript_dir</code> = Subdirectory of <code>parent_dir</code> that contains the Rscript used to generate the figure</td>
</tr>
<tr>
<td></td>
<td>3. <code>rscript_name</code> = Name of the Rscript used to generate the figure</td>
</tr>
<tr>
<td></td>
<td>4. <code>results_dir</code> = Subdirectory of <code>parent_dir</code> where the figure is stored</td>
</tr>
<tr>
<td></td>
<td>5. <code>filename</code> = Filename</td>
</tr>
<tr>
<td><code>include_time</code></td>
<td>is logical with default TRUE. If TRUE, it includes date / time in the output character</td>
</tr>
</tbody>
</table>

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

```r
xgx_geom_ci(
  mapping = NULL,
  data = NULL,
  conf_level = 0.95,
  distribution = "normal",
  bins = NULL,
  breaks = NULL,
  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.
xgx_geom_pi

bins
number of bins to cut up the x data, cuts data into quantiles.

breaks
breaks to cut up the x data, if this option is used, bins is ignored.

gem
Use to override the default geom. Can be a list of multiple geoms, e.g. list("point","line","errorbar"), 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

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)

xgx_geom_pi

Plot data with median and percent intervals

Description
Plot data with median and percent intervals

Usage

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

data <- data.frame(x = rep(c(1, 2, 3), each = 20),
y = rep(c(1, 2, 3), each = 20) + stats::rnorm(60))
**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**

```r
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
Examples

```r
xgx_minor_breaks_log10(c(1, 1000))
xgx_minor_breaks_log10(c(0.001, 100))
xgx_minor_breaks_log10(c(1e-4, 1e4))
xgx_minor_breaks_log10(c(1e-9, 1e9))
xgx_minor_breaks_log10(c(1, 2))
xgx_minor_breaks_log10(c(1, 5))
xgx_minor_breaks_log10(c(1, 10))
xgx_minor_breaks_log10(c(1, 100))
xgx_minor_breaks_log10(c(1, 1.01))
xgx_minor_breaks_log10(c(1, 1.0001))
print(xgx_minor_breaks_log10(c(1, 1.000001)), digits = 10)
```

---

**xgx_plot**

Create a new xgx plot

**Description**

Create a new xgx plot

**Usage**

```r
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 one by fortify.
- `mapping` : As in ggplot2: Default list of aesthetic mappings to use for plot. Must define x, y, and group for xgx_spaghetti.
- `...` : 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 will look for it in this environment. It defaults to using the environment in which ggplot is called.

**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,
  status_fontcolor = "grey",
  filenames_fontsize = 11,
  filenames_fontcolor = "black"
)

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()
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
status_fontcolor: font color for status in plot
filenames_fontsize: font size for filenames info in plot
filenames_fontcolor: font color for filenames info 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, ggplo2::aes(x = x, y = y)) +
ggplot2::geom_point()
xgx_save(4, 4, dirs, "Example", "DRAFT")
xgx_save_table

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)

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

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")
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

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

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()
Reverse-log transform for the x scale.

Description

\texttt{xgx\_scale\_x\_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

\texttt{xgx\_scale\_x\_reverselog10(labels = NULL, accuracy = NULL, \ldots )}

Arguments

- \textit{labels} if NULL, then the default is to use \texttt{scales::percent()}
- \textit{accuracy} if NULL, then use the the default as specified by \texttt{scales::percent()} to round to the hundredths place, set accuracy 0.01
- \ldots other parameters passed to \texttt{scale\_x\_continuous}

Value

\texttt{ggplot2} compatible scale object

Examples

\begin{verbatim}
conc <- 10^(seq(-3, 3, by = 0.1))
ce50 <- 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()
\end{verbatim}
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

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

xgx_scale_y_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 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

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
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_y_percentchangelog10**

percentchangelog10 transform for the y scale.

---

Description

`xgx_scale_y_percentchangelog10` and `xgx_scale_x_percentchangelog10` are 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,
  ...
)
```

```r
xgx_scale_x_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 \(2^{\text{labeling::extended}(\log_{2}(\text{PCHG} + 1))} - 1\), where PCHG represents the range of the data.
xgx_scale_y_reverselog10

- **minor_breaks**: if NULL, then default is to use nicely spaced log10(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**

```r
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()
```

**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

```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_reverselog10()
```

---

**xgx_stat_ci**

*Plot data with mean and confidence intervals*

Description

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

Usage

```r
xgx_stat_ci(
    mapping = NULL,
    data = NULL,
    conf_level = 0.95,
    distribution = "normal",
    bins = NULL,
    breaks = NULL,
    geom = list("point", "line", "errorbar"),
    position = "identity",
    fun.args = list(),
    fun.data = NULL,
    na.rm = FALSE,
    orientation = "x",
    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.
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.

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.

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.

number of bins to cut up the x data, cuts data into quantiles.
breaks to cut up the x data, if this option is used, bins is ignored
Use to override the default geom. Can be a list of multiple geoms, e.g. list("point","line","errorbar"),
which is the default.
Position adjustment, either as a string, or the result of a call to a position adjust-
ment function.
Optional additional arguments passed on to the functions.
A function that is given the complete data and should return a data frame with
variables ymin, y, and ymax.
If FALSE, the default, missing values are removed with a warning. If TRUE,
missing values are silently removed.
The orientation of the layer, passed on to ggplot2::stat_summary. Only imple-
mented for ggplot2 v.3.3.0 and later. The default ("x") summarizes y values over
x values (same behavior as ggplot2 v.3.2.1 or earlier). Setting orientation =
"y" will summarize x values over y values, which may be useful in some sit-
uations where you want to flip the axes, e.g. to create forest plots. Setting
orientation = NA will try to automatically determine the orientation from the
aesthetic mapping (this is more stable for ggplot2 v.3.3.2 compared to v.3.3.0).
See stat_summary (v.3.3.0 or greater) for more information.
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.
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))

# plotting ordinal or multinomial data
set.seed(12345)
data = data.frame(x = 120*exp(stats::rnorm(100,0,1)),
                  response = sample(c("Mild","Moderate","Severe"), 100, replace = TRUE),
                  covariate = sample(c("Male","Female"), 100, replace = TRUE))
xgx_plot(data = data) +
xgx_stat_ci(mapping = ggplot2::aes(x = x, response = response, colour = covariate),
distribution = "ordinal", bins = 4) +
ggplot2::scale_y_continuous(labels = scales::percent_format()) + ggplot2::facet_wrap(~response)

xgx_plot(data = data) +
xgx_stat_ci(mapping = ggplot2::aes(x = x, response = response, colour = response),
distribution = "ordinal", bins = 4) +
ggplot2::scale_y_continuous(labels = scales::percent_format()) + ggplot2::facet_wrap(~covariate)

# Example plotting categorical vs categorical data
set.seed(12345)
data = data.frame(x = 120*exp(stats::rnorm(100,0,1)),
 response = sample(c("Trt1", "Trt2", "Trt3"), 100, replace = TRUE),
 covariate = factor(sample(c("White","Black","Asian","Other"), 100, replace = TRUE),
 levels = c("White", "Black", "Asian", "Other")))

xgx_plot(data = data) +
xgx_stat_ci(mapping = ggplot2::aes(x = response, response = covariate),
distribution = "ordinal") +
xgx_stat_ci(mapping = ggplot2::aes(x = 1, response = covariate), geom = "hline",
distribution = "ordinal") +
ggplot2::scale_y_continuous(labels = scales::percent_format()) +
ggplot2::facet_wrap(~covariate) +
ggplot2::xlab("Treatment group") +
ggplot2::ylab("Percent of subjects by category")

# Same example with orientation flipped (only works for ggplot2 v.3.3.0 or later)
# only run if ggplot2 v.3.3.0 or later
if(ggplot2_geq_v3.3.0){
    xgx_plot(data = data) +
xgx_stat_ci(mapping = ggplot2::aes(y = response, response = covariate), orientation = "y",
distribution = "ordinal") +
xgx_stat_ci(mapping = ggplot2::aes(y = 1, response = covariate), orientation = "y",
geom = "vline", distribution = "ordinal") +
ggplot2::scale_x_continuous(labels = scales::percent_format()) +
ggplot2::facet_wrap(~covariate) +
ggplot2::ylab("Percent of subjects by category")
}

---

**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",
  bins = NULL,
  breaks = NULL,
  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.
- **bins**: Number of bins to cut up the x data, cuts data into quantiles.
- **breaks**: Breaks to cut up the x data, if this option is used, bins is ignored.
- **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_stat_pi(percent_level = 0.70) +
         xgx_stat_pi(percent_level = 0.60)

xgx_stat_smooth Wrapper for stat_smooth

Description

xgx_stat_smooth and xgx_geom_smooth produce smooth fits through continuous or categorical data. For categorical, ordinal, or multinomial data use method = polr. This wrapper also works with nonlinear methods like nls and nlsLM for continuous data.

xgx_geom_smooth_emax uses minpack.lm::nlsLM, predictdf.nls, and stat_smooth to display Emax model fit to data
Usage

```r
xgx_stat_smooth(
  mapping = NULL,
  data = NULL,
  geom = "smooth",
  position = "identity",
  ..., 
  method = NULL,
  formula = NULL,
  se = TRUE,
  n = 80,
  span = 0.75,
  n_boot = 200,
  fullrange = FALSE,
  level = 0.95,
  method.args = list(),
  na.rm = FALSE,
  orientation = "x",
  show.legend = NA,
  inherit.aes = TRUE
)
```

```r
xgx_geom_smooth(
  mapping = NULL,
  data = NULL,
  geom = "smooth",
  position = "identity",
  ..., 
  method = NULL,
  formula = NULL,
  se = TRUE,
  n = 80,
  span = 0.75,
  fullrange = FALSE,
  level = 0.95,
  method.args = list(),
  na.rm = FALSE,
  orientation = "x",
  show.legend = NA,
  inherit.aes = TRUE
)
```

```r
xgx_geom_smooth_emax(
  mapping = NULL,
  data = NULL,
  geom = "smooth",
  position = "identity",
  ..., 
)```
method = "nlsLM",
formula,
se = TRUE,
n = 80,
span = 0.75,
fullrange = FALSE,
level = 0.95,
method.args = list(),
na.rm = FALSE,
orientation = "x",
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. Warning: for ‘method = polr’, do not define ‘y’ aesthetic, use ‘response’ instead.
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.geom
Use to override the default geom. Can be a list of multiple geoms, e.g. list("point","line","errorbar"), which is the default.
position
Position adjustment, either as a string, or the result of a call to a position adjustment function.

... 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.
method
method (function) to use, eg. lm, glm, gam, loess, rlm. Example: "polr" for ordinal data. "nlsLM" for nonlinear least squares. If method is left as ‘NULL’, then a typical ‘StatSmooth’ is applied, with the corresponding defaults, i.e. For datasets with n < 1000 default is loess. For datasets with 1000 or more observations defaults to gam.

formula
formula to use in smoothing function, eg. y ~ x, y ~ poly(x, 2), y ~ log(x)
se
display confidence interval around smooth? (TRUE by default, see level to control)
n
number of points to evaluate smoother at
span
Controls the amount of smoothing for the default loess smoother. Smaller numbers produce wigglier lines, larger numbers produce smoother lines.
n_boot
number of bootstraps to perform to compute confidence interval, currently only used for method = "polr", default is 200

fullrange
should the fit span the full range of the plot, or just the data

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.

method.args
Optional additional arguments passed on to the method.

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

orientation
The orientation of the layer, passed on to ggplot2::stat_summary. Only implemented for ggplot2 v.3.3.0 and later. The default ("x") summarizes y values over x values (same behavior as ggplot2 v.3.2.1 or earlier). Setting orientation = "y" will summarize x values over y values, which may be useful in some situations where you want to flip the axes, e.g. to create forest plots. Setting orientation = NA will try to automatically determine the orientation from the aesthetic mapping (this is more stable for ggplot2 v.3.3.2 compared to v.3.3.0).

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.

Value

ggplot2 plot layer

Warning

nlsLM uses nls.lm which implements the Levenberg-Marquardt algorithm for fitting a nonlinear model, and may fail to converge for a number of reasons. See ?nls.lm for more information.

nls uses Gauss-Newton method for estimating parameters, and could fail if the parameters are not identifiable. If this happens you will see the following warning message: Warning message: Computation failed in 'stat_smooth()': singular gradient

nls will also fail if used on artificial "zero-residual" data, use nlsLM instead.

See Also

predictdf.nls for information on how nls confidence intervals are calculated.

Examples

# Example with nonlinear least squares (method = "nlsLM")
Nsubj <- 10
Doses <- c(0, 25, 50, 100, 200)
Ntot <- Nsubj*length(Doses)
times <- c(0,14,30,60,90)
xgx_stat_smooth

```r
# Load required libraries
library(dplyr)
library(ggplot2)
library(stats)

# Generate data
Dat1 <- data.frame(ID = 1:(Ntot),
                   DOSE = rep(Doses, Nsubj),
                   PD0 = stats::rlnorm(Ntot, log(100), 1),
                   Kout = exp(stats::rnorm(Ntot,-2, 0.3)),
                   Imax = 1,
                   ED50 = 25) %>%
  dplyr::mutate(PDSS = PD0*(1 - Imax*DOSE/(DOSE + ED50))*exp(stats::rnorm(Ntot, 0.05, 0.3))) %>%
  merge(data.frame(ID = rep(1:(Ntot), each = length(times)), Time = times), by = "ID") %>%
  dplyr::mutate(PD = ((PD0 - PDSS)*(exp(-Kout*Time)) + PDSS),
                PCHG = (PD - PD0)/PD0)

# Generate plots
gg <- ggplot2::ggplot(Dat1 %>% subset(Time == 90),
                       ggplot2::aes(x = DOSE, y = PCHG)) +
    ggplot2::geom_boxplot(ggplot2::aes(group = DOSE)) +
    xgx_theme() +
    xgx_scale_y_percentchangelog10() +
    ggplot2::ylab("Percent Change from Baseline") +
    ggplot2::xlab("Dose (mg)"

gg +
  xgx_stat_smooth(method = "nlsLM", formula = y ~ E0 + Emax*x/(ED50 + x),
                   method.args = list(
                       start = list(Emax = -0.50, ED50 = 25, E0 = 0),
                       lower = c(-Inf, 0, -Inf))
                   se = TRUE)
```

## Not run:

# Example with ordinal data (method = "polr")
set.seed(12345)
data = data.frame(x = 120*exp(stats::rnorm(100,0,1)),
                   response = sample(c("Mild","Moderate","Severe"), 100, replace = TRUE),
                   covariate = sample(c("Male","Female"), 100, replace = TRUE)) %>%
  dplyr::mutate(y = (50 + 20*x/(200 + x))*exp(stats::rnorm(100, 0, 0.3)))

# Example coloring by the response categories
xgx_plot(data = data) +
  xgx_stat_smooth(mapping = ggplot2::aes(x = x, response = response,
                                         colour = response, fill = response),
                 method = "polr") +
  ggplot2::scale_y_continuous(labels = scales::percent_format())

# Example faceting by the response categories, coloring by a different covariate
xgx_plot(data = data) +
  xgx_stat_smooth(mapping = ggplot2::aes(x = x, response = response,
                                          colour = covariate, fill = covariate),
                  method = "polr", level = 0.80) +
  ggplot2::facet_wrap(~response) +
  ggplot2::scale_y_continuous(labels = scales::percent_format())
```
xgx_summarize_covariates

Summarize Covariate information in a dataset

Description

xgx_summarize_covariates

Usage

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

Value

list

Examples

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

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**

concentration <- 10^(seq(-3, 3, by = 0.1))
ec50 <- 1
data <- data.frame(concentration = concentration,
                    bound_receptor = 1 * concentration / (concentration + 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()
xgx_theme_set

Sets the standard theme for xGx graphics

Description

xgx_theme_set

Usage

xgx_theme_set()

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))
xgx_theme_set()
ggplot2::ggplot(data, ggplot2::aes(y = concentration, x = bound_receptor)) +
ggplot2::geom_point() +
ggplot2::geom_line() +
xgx_scale_y_log10() +
xgx_scale_x_reverse_log10()
Index

* datasets
  - case1_pkpd, 3
  - mad, 7
  - mad_missing_duplicates, 8
  - mad_nca, 8
  - nlmixr_theo_sd, 9
  - sad, 14
  - StatSmoothOrdinal, 14
  - StatSummaryBinQuant, 15
  - StatSummaryOrdinal, 15
  - stats::extended_breaks(), 38
  - stat_summary, 43

  - binom.exact, 26, 28, 43
  - case1_pkpd, 3
  - edit_rmd_template_str, 4
  - get_rmd_name, 5
  - get_rmd_str, 6
  - ggplot, 32
  - layer, 18
  - mad, 7
  - mad_missing_duplicates, 7
  - mad_nca, 8
  - nlmixr_theo_sd, 9
  - predict.nls, 9
  - predictdf, 11
  - predictdf.nls, 12, 50
  - predictdf.polr, 13
  - sad, 14
  - scale_x_continuous, 37, 38
  - scale_x_log10, 36
  - scale_y_continuous, 41
  - scale_y_log10, 39
  - scales::extended_breaks(), 38
  - stat_summary, 43

  - StatSmoothOrdinal, 14
  - StatSummaryBinQuant, 15
  - StatSummaryOrdinal, 15
  - theme_xgx, 16
  - transformation object, 38
  - xgx_annotate_filenames, 16
  - xgx_annotate_status, 17
  - xgx_annotate_status_png, 18
  - xgx_auto_explore, 20
  - xgx_breaks_log10, 23
  - xgx_breaks_time, 24
  - xgx_check_data, 25, 52
  - xgx_conf_int, 26
  - xgx_dirs2char, 27
  - xgx_geom.ci, 28
  - xgx_geom.pi, 29
  - xgx_geom_smooth(xgx_stat_smooth), 47
  - xgx_geom_smooth_emax(xgx_stat_smooth), 47
  - xgx_labels_log10, 31
  - xgx_minor_breaks_log10, 31
  - xgx_plot, 32
  - xgx_save, 33
  - xgx_save_table, 35
  - xgx_scale_x_log10, 36
  - xgx_scale_x_percentchangelog10
    (xgx_scale_y_percentchangelog10), 40
  - xgx_scale_x_reverselog10, 37
  - xgx_scale_x_time_units, 38
  - xgx_scale_y_log10, 39
  - xgx_scale_y_percentchangelog10, 40
  - xgx_scale_y_reverselog10, 41
  - xgx_scale_y_time_units
    (xgx_scale_x_time_units), 38
xgx_summarize_covariates, 52
xgx_summarize_data, 52
xgx_theme, 53
xgx_theme_set, 54