

Package ‘coursekata’

September 22, 2026

Title Packages and Functions for 'CourseKata' Courses

Version 0.21.0

Date 2026-09-21

Description Easily install and load all packages and functions used in 'CourseKata' courses. Aid teaching with helper functions and augment generic functions to provide cohesion between the network of packages. Learn more about 'CourseKata' at <<https://www.coursekata.org>>.

License GPL (>= 3)

URL <https://coursekata.github.io/coursekata-r/>,
<https://github.com/coursekata/coursekata-r>

BugReports <https://github.com/coursekata/coursekata-r/issues>

Depends R (>= 4.1)

Imports cli (>= 3.2.0), dslabs (>= 0.7.4), ggformula (>= 1.0.0),
ggplot2 (>= 4.0.2), glue (>= 1.6.2), grid, lifecycle (>= 1.0.3),
lsr (>= 0.5.2), Metrics, mosaic (>= 1.10.2), palmerpenguins,
purrr (>= 0.3.4), remotes, rlang (>= 1.0.2), supernova (>= 2.5.1),
vctrs (>= 0.4.1), viridisLite

Suggests fivethirtyeight (>= 0.6.2), knitr (>= 1.40), lubridate (>= 1.8.0),
MASS, mockery (>= 0.4.3), mockr (>= 0.1), readr (>= 2.1.2),
readxl (>= 1.4.0), rmarkdown (>= 2.17), scales, usethis (>= 2.1.6),
simstudy (>= 0.5.0), testthat (>= 3.1.5), tibble (>= 3.1.7),
tidyr (>= 1.2.0), vdiff (>= 1.0.2), withr (>= 2.5.0)

Config/testthat/edition 3

Config/testthat/parallel true

Language en-US

Encoding UTF-8

LazyData true

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Adam Blake [cre, aut] (ORCID: <<https://orcid.org/0000-0001-7881-8652>>),
 Ji Son [aut] (ORCID: <<https://orcid.org/0000-0002-4258-4791>>),
 Jim Stigler [aut] (ORCID: <<https://orcid.org/0000-0001-6107-7827>>),
 CourseKata [cph]

Maintainer Adam Blake <adam@coursekata.org>

Repository CRAN

Date/Publication 2026-09-21 22:50:02 UTC

Contents

Ames	3
class_data	5
coursekata_attach	6
coursekata_install	6
coursekata_load_theme	7
coursekata_packages	7
coursekata_palette	8
coursekata_palette_provider	8
coursekata_repos	9
coursekata_unload_theme	9
er	10
estimate_extraction	12
fevdata	14
Fingers	15
FingersMessy	16
fit_stats	17
game_data	17
GeomCutoff	18
GeomModel	19
GeomResid	19
GeomSquareplot	20
GeomSquareResid	21
geom_b	21
geom_model	23
geom_resid	25
geom_squareplot	28
gf_b	30
gf_model	35
gf_reduce	38
gf_resid	40
gf_resid_fun	42
gf_sd_ruler	44
gf_squareplot	46
gf_square_reduce	50
gf_square_resid	53
gf_square_resid_fun	55
guide_cutoff	57

guide_dgp	58
middle	60
penguins	61
scale_discrete_coursekata	62
show_cutoffs	63
show_dgp	64
show_mean	65
Smallville	66
split_data	67
StatCutoff	68
StatDistMean	69
StatModel	70
StatReduce	71
StatResid	71
StatSdRuler	72
StatSquareplot	73
Survey	74
Tables	74
theme_coursekata	75
TipExperiment	75
tip_exp	76
World	76
Index	78

Ames

Ames, Iowa housing data

Description

Data describing all residential home sales in Ames, Iowa from the years 2006–2010 as reported by the Ames City Assessor’s Office and compiled by De Cock (2011). Ames is located about 30 miles north of Des Moines (the state capitol) and is home to Iowa State University (the largest university in the state). Each row represents the latest sale of a home (one row per home in the dataset). Columns represent home features and sale prices (outcome). The original dataset includes a uniquely detailed (81 features per home) and comprehensive look at the housing market. The data included here are only a subset used for examples in CourseKata course material. See the references and data source for the full dataset.

Pedagogical Modifications:

To simplify the dataset for instructional purposes, the data were filtered to include only single family homes, residential zoning, 1-2 story homes, homes with brick, cinder block, or concrete foundations, and average to excellent kitchen qualities. Further, the descriptive variables were reduced to the subset described in the format section.

Usage

Ames

Format

A data frame with 2930 observations on the following 80 variables:

YearBuilt Year home was built (YYYY).

YearSold Year of home sale (YYYY). Note: all home sales in this dataset occurred between 2006 - 2010. If a home was sold more than once between 2006 - 2010, only its latest sale is included in dataset.

Neighborhood One of two neighborhoods in Ames county:

- College Creek (CollegeCreek), a neighborhood located adjacent to Iowa State University (the largest University in the state).
- Old Town (OldTown), a nationally designated historic district in Ames. The old neighborhood is located just north of the central business district.

HomeSizeR Raw above-ground area of home, measured in square feet.

HomeSizeK Above-ground area of home, measured in thousands of square feet.

LotSizeR Raw total property lot size, measured in square feet.

LotSizeK Total property lot size, in thousands of square feet.

Floors Number of above-ground floors (1 story or 2 story).

BuildQuality Assessor's rating of overall material and finish of the house.

- 10: Very Excellent
- 9: Excellent
- 8: Very Good
- 7: Good
- 6: Above Average
- 5: Average
- 4: Below Average
- 3: Fair
- 2: Poor
- 1: Very Poor

Foundation Type of foundation (ground material underneath the house).

- Brick&Tile: Brick and Tile
- CinderBlock: Cinder Blocks
- PouredConcrete: Poured Concrete

HasCentralAir Indicator if home contains central air conditioning (0 = No, 1 = Yes).

Bathrooms Number of full above-ground bathrooms.

Bedrooms Number of full above-ground bedrooms.

TotalRooms Number of above-ground rooms in home, excluding bathrooms.

KitchenQuality Assessor's rating of kitchen material quality.

- Excellent
- Good
- Average

HasFireplace Indicator if home contains at least one fireplace (0 = No, 1 = Yes).

GarageType Type of garage.

- Attached: includes attached, built-in, basement, and dual-type garages
- Detached: includes detached and carport garages
- None: home does not have a garage or carport

GarageCars Number of cars that can fit in garage.

PriceR Sale price of home, in raw USD (\$)

PriceK Sale price of home, in thousands of USD (\$)

TinySet (Ignore) Whether or not this row is in ames_tiny.csv

Source

<https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/data>

References

De Cock, Dean, (2011). Ames, Iowa: Alternative to the Boston Housing Data as an end of semester regression project, *Journal of Statistics Education*, 19(3). doi:10.1080/10691898.2011.11889627

class_data

Generated "class data" for exploring pairwise tests

Description

These data were generated as outcomes for "students" for three different "instructors" named A, B, and C. The outcome have means such that $C > B > A$, but the difference is only clearly significant for $C > A$, and borderline for the others.

Usage

```
class_data
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 105 rows and 2 columns.

Details

outcome A hypothetical, numerical outcome of an intervention.

teacher Either "A", "B", or "C", associating the outcome to a teacher.

coursekata_attach	<i>Attach the CourseKata course packages</i>
-------------------	--

Description

Attach the CourseKata course packages

Usage

```
coursekata_attach(do_not_ask = FALSE, quietly = FALSE)
```

Arguments

do_not_ask	Prevent asking the user to install missing packages (they are skipped).
quietly	Whether to suppress messages.

Value

A named logical vector indicating which packages were attached.

Examples

```
coursekata_attach()
```

coursekata_install	<i>Install or update all CourseKata packages.</i>
--------------------	---

Description

Install or update all CourseKata packages.

Usage

```
coursekata_install(...)
```

```
coursekata_update(...)
```

Arguments

...	Arguments passed on to <code>remotes::install_cran</code> or <code>remotes::install_github</code> depending on whether the package appears to be from CRAN or GitHub.
-----	---

Value

The state of all the packages after any updates have been performed.

coursekata_load_theme *Utility function for loading all themes.*

Description

This function is called at package start-up and should rarely be needed by the user. The exception is when the user has called `coursekata_unload_theme()` and wants to go back to the CourseKata look and feel. When run, this function sets the CourseKata color palettes `coursekata_palette()`, sets the default theme to `theme_coursekata()`, and tweaks some default settings for specific plots. To restore the plotting settings that were active before this function first changed them, run `coursekata_unload_theme()`.

Usage

```
coursekata_load_theme()
```

Value

No return value, called to adjust the global state of ggplot2.

See Also

`coursekata_palette` `theme_coursekata` `scale_discrete_coursekata` `coursekata_unload_theme`

coursekata_packages *List all CourseKata course packages*

Description

List all CourseKata course packages

Usage

```
coursekata_packages(check_remote_version = FALSE)
```

Arguments

`check_remote_version`

Should the remote version number be checked? Requires internet, and will take longer.

Value

A data frame with three variables: the name of the package `package`, the version, and whether it is currently attached.

Examples

```
coursekata_packages()
```

coursekata_palette	<i>The color palettes used in our theme system</i>
--------------------	--

Description

The color palettes used in our theme system

Usage

```
coursekata_palette(indices = integer(0))
```

Arguments

indices The indices of the colors to pull (or all colors if no indices are given).

Value

A named list of the requested colors in the palette.

Examples

```
coursekata_palette()
coursekata_palette(c(1, 3, 5))
```

coursekata_palette_provider	<i>Create a function that provides a colorblind palette.</i>
-----------------------------	--

Description

Create a function that provides a colorblind palette.

Usage

```
coursekata_palette_provider()
```

Value

A function that accepts one argument `n`, which is the number of colors you want to use in the plot. This function is used by scales like `scale_color_discrete` to provide colorblind- safe palettes. Where possible, the function will use the hand-picked colors from `coursekata_palette()`, and when more colors are needed than are available, it will use the `viridisLite::viridis()` palette.

See Also

scale_discrete_coursekata

Examples

```
palette <- coursekata_palette_provider()
palette(3)
```

coursekata_repos	<i>Get repositories for the packages.</i>
------------------	---

Description

Ensures a default CRAN is set if one is not already set.

Usage

```
coursekata_repos(repos = getOption("repos"))
```

Arguments

repos Optionally set a repository character vector to augment.

Value

A set of repositories that can be used to install or update the CourseKata packages.

Examples

```
coursekata_repos()
```

coursekata_unload_theme	<i>Restore the caller's plotting settings</i>
-------------------------	---

Description

This function restores the theme, options, and geom defaults that were active before [coursekata_load_theme\(\)](#) first changed them. Calling it again before another load has no effect. To go back to the CourseKata look and feel, run [coursekata_load_theme\(\)](#).

Usage

```
coursekata_unload_theme()
```

Value

No return value, called to restore the global state of ggplot2.

See Also

coursekata_load_theme

er

Emergency room canine therapy

Description

Data from: Controlled clinical trial of canine therapy versus usual care to reduce patient anxiety in the emergency department.

Abstract:*Objective:*

Test if therapy dogs can reduce anxiety in emergency department (ED) patients.

Methods:

In this controlled clinical trial (NCT03471429), medically stable, adult patients were approached if the physician believed that the patient had “moderate or greater anxiety.” Patients were allocated on a 1:1 ratio to either 15 min exposure to a certified therapy dog and handler (dog), or usual care (control). Patient reported anxiety, pain and depression were assessed using a 0-10 scale (10=worst). Primary outcome was change in anxiety from baseline (T0) to 30 min and 90 min after exposure to dog or control (T1 and T2 respectively); secondary outcomes were pain, depression and frequency of pain medication.

Results:

Among 98 patients willing to participate in research, 7 had aversions to dogs, leaving 91 (93%) were willing to see a dog; 40 patients were allocated to each group (dog or control). No data were normally distributed. Median baseline anxiety, pain and depression were similar between groups. With dog exposure, anxiety decreased significantly from T0 to T1: 6 (IQR 4-9.75) to T1: 2 (0-6) compared with 6 (4-8) to 6 (2.5-8) in controls ($P < 0.001$, for T1, Mann-Whitney U). Dog exposure was associated with significantly lower anxiety at T2 and a significant overall treatment effect on two-way repeated measures ANOVA for anxiety, pain and depression. After exposure, 1/40 in the dog group needed pain medication, versus 7/40 in controls ($P = 0.056$, Fisher’s).

Conclusions:

Exposure to therapy dogs plus handlers significantly reduced anxiety in ED patients.

Usage

er

Format

A data frame with 84 observations on the following 53 variables:

id Subject ID

condition Whether the subject saw a Dog or was in the Control group

age Subject's age in years

gender Subject's self-identified gender

race Subject's self-identified race

veteran Is the subject a veteran?

disabled Is the subject disabled?

dog_name The name of the therapy dog

base_pain Subject's self reported pain before the intervention (T0)

base_depression Subject's self reported depression before the intervention (T0)

base_anxiety Subject's self reported anxiety before the intervention (T0)

base_total The sum of the subject's base_* scores

later_pain Subject's self reported pain after the intervention (T1)

later_depression Subject's self reported depression after the intervention (T1)

later_anxiety Subject's self reported anxiety after the intervention (T1)

later_total The sum of the subject's later_* scores

last_pain Subject's self reported pain after the intervention (T2)

last_depression Subject's self reported depression after the intervention (T2)

last_anxiety Subject's self reported anxiety after the intervention (T2)

last_total The sum of the subject's last_* scores

change_pain The change in subject's pain from before the intervention to after

change_depression The change in subject's depression from before the intervention to after

change_anxiety The change in subject's anxiety from before the intervention to after

change_total The sum of the subject's change_* scores

provider_male Was the health care provider male?

provider The health care provider's status: either an Advanced Practitioner, Resident physician, or Attending physician

heart_rate The subject's heart rate at baseline (T0)

resp_rate The subject's respiratory rate at baseline (T0)

sp_o2 The subject's SpO2 at baseline (T0)

bp_syst The subject's systolic blood pressure at baseline (T0)

bp_diast The subject's diastolic blood pressure at baseline (T0)

med_given Was the subject given medication prior to the study? (T0)

mh_none None of the other medical history items were indicated

mh_asthma Medical history: asthma

mh_smoker Medical history: smoker
 mh_cad Medical history: coronary artery disease
 mh_diabetes Medical history: diabetes mellitus
 mh_hypertension Medical history: hypertension
 mh_stroke Medical history: prior stroke
 mh_chronic_kidney Medical history: chronic kidney disease
 mh_copd Medical history: chronic obstructive pulmonary disease
 mh_hyperlipidemia Medical history: hyperlipidemia
 mh_hiv Medical history: HIV
 mh_other Medical history: other (write-in)
 ph_adhd Psychiatric history: attention-deficit/hyperactivity disorder
 ph_anxiety Psychiatric history: anxiety
 ph_bipolar Psychiatric history: bipolar
 ph_borderline Psychiatric history: borderline personality disorder
 ph_depression Psychiatric history: depression
 ph_schizophrenia Psychiatric history: schizophrenia
 ph_ptsd Psychiatric history: PTSD
 ph_none None of the other psychiatric history items were indicated
 ph_other Psychiatric history: other (write-in)

References

Kline, J. A., Fisher, M. A., Pettit, K. L., Linville, C. T., & Beck, A. M. (2019). Controlled clinical trial of canine therapy versus usual care to reduce patient anxiety in the emergency department. *PloS One*, 14(1), e0209232. doi:10.1371/journal.pone.0209232

estimate_extraction	<i>Extract estimates/statistics from a model</i>
---------------------	--

Description

This collection of functions is useful for extracting estimates and statistics from a fitted model. They are particularly useful when estimating many models, like when bootstrapping confidence intervals. Each function can be used with an already fitted model as an [lm](#) object, or a formula and associated data can be passed to it. **All of these assume the comparison is the empty model.**

Usage

```

b0(object, data = NULL)

b1(object, data = NULL)

b(object, data = NULL, all = FALSE, predictor = character())

f(object, data = NULL, all = FALSE, predictor = character(), type = 3)

pre(object, data = NULL, all = FALSE, predictor = character(), type = 3)

p(object, data = NULL, all = FALSE, predictor = character(), type = 3)

```

Arguments

object	A lm object, or formula .
data	If object is a formula, the data to fit the formula to as a data.frame .
all	If TRUE, return a named list of all related terms (e.g. all <i>F</i> -values). The name for the full model value is the name of the function (e.g. "f"), and the names for the constituent terms are the term names prefixed by the function name (e.g. "f_a:b" for the <i>F</i> -value of the a:b interaction term).
predictor	Filter the output down to just the statistics for these terms (e.g. "hp" to just get the statistics for that term in the model). This argument is flexible: you can pass a character vector of terms (c("hp", "hp:cyl")), a one-sided formula (~hp), or a list of formulae (c(~hp, ~hp:cyl)).
type	The type of sums of squares to calculate (see generate_models()). Defaults to the widely used Type III SS.

Details

- b0: The intercept from the full model.
- b1: The slope b1 from the full model.
- b: The coefficients from the full model.
- f: The F value from the full model.
- pre: The Proportional Reduction in Error for the full model.
- p: The *p*-value from the full model.
- sse: The SS Error (SS Residual) from the model.
- ssm: The SS Model (SS Regression) for the full model.
- ssr: Alias for SSM.

fVal() and PRE() are older names for [f\(\)](#) and [pre\(\)](#). They are kept for backward compatibility and behave identically to the newer functions.

Value

The value of the estimate as a single number.

References

Judd, C. M., McClelland, G. H., & Ryan, C. S. (2017). *Data Analysis: A Model Comparison Approach to Regression, ANOVA, and Beyond* (3rd ed.). New York: Routledge. ISBN:879-1138819832

Examples

```
supernova(lm(mpg ~ disp, data = mtcars))

change_p_decimals <- supernova(lm(mpg ~ disp, data = mtcars))
print(change_p_decimals, pcut = 8)
```

fevdata

Forced Expiratory Volume (FEV) Data

Description

Data from: Fundamentals of Biostatistics Notes from: Kahn, M.

Abstract:

Sample of 654 youths, aged 3 to 19, in the area of East Boston during middle to late 1970's. Interest concerns the relationship between smoking and FEV. Since the study is necessarily observational, statistical adjustment via regression models clarifies the relationship.

Pedagogical Notes::

This is a versatile dataset that can be used throughout an introductory statistics course as well as an introductory modeling course. It includes many issues from statistical adjustment in observational studies, to subgroup analysis, quadratic regression and analysis of covariance.

Usage

```
fevdata
```

Format

A data frame with 654 observations on the following 5 variables:

AGE Age, in years

FEV Forced expiratory volume, in liters

HEIGHT Height, in inches

SEX 0 = Female, 1 = Male

SMOKE 0 = Non-smoker, 1 = Smoker

References

Kahn, M. (2003). Data Sleuth, STATS, 37, 24. <https://jse.amstat.org/datasets/fev.txt>
 Rosner, B. (1999). Fundamentals of Biostatistics, Pacific Grove, CA: Duxbury

Fingers

*Data from introductory statistics students at a university.***Description**

Students at a university taking an introductory statistics course were asked to complete this survey as part of their homework.

Usage

Fingers

Format

A data frame with 157 observations on the following 16 variables:

Gender Gender of participant.

RaceEthnic Racial or ethnic background.

FamilyMembers Members of immediate family (excluding self).

SSLast Last digit of social security number (NA if no SSN).

Year Year in school: 1=First, 2=Second, 3=Third, 4=Fourth, 5=Other

Job Current employment status: 1=Not Working, 2=Part-time Job, 3=Full-time Job

MathAnxious Agreement with the statement "In general I tend to feel very anxious about mathematics": 1=Strongly Disagree, 2=Disagree, 3=Neither Agree nor Disagree, 4=Agree, 5=Strongly Agree

Interest Interest in statistics and the course: 1=No Interest, 2=Somewhat Interested, 3=Very Interested

GradePredict Numeric prediction for final grade in the course. The value is converted from the student's letter grade prediction. 4.0=A, 3.7=A-, 3.3=B+, 3.0=B, 2.7=B-, 2.3=C+, 2.0=C, 1.7=C-, 1.3=Below C-

Thumb Length in mm from tip of thumb to the crease between the thumb and palm.

Index Length in mm from tip of index finger to the crease between the index finger and palm.

Middle Length in mm from tip of middle finger to the crease between the middle finger and palm.

Ring Length in mm from tip of ring finger to the crease between the middle finger and palm.

Pinkie Length in mm from tip of pinkie finger to the crease between the pinkie finger and palm.

Height Height in inches.

Weight Weight in pounds.

Sex Sex of participant.

FingersMessy

Raw data from introductory statistics students at a university.

Description

This is the Fingers dataset before it was cleaned. In the cleaning process, we converted the values from numbers to appropriate types (where applicable), removed outliers that suggested data was input incorrectly, and we removed incomplete cases. The description for the dataset is: Students at a university taking an introductory statistics course were asked to complete this survey as part of their homework. (This is the same data set as the Fingers data)

Usage

FingersMessy

Format

A data frame with 157 observations on the following 16 variables:

Gender Gender of participant.

RaceEthnic Racial or ethnic background.

FamilyMembers Members of immediate family (excluding self).

SSLast Last digit of social security number (NA if no SSN).

Year Year in school: 1=First, 2=Second, 3=Third, 4=Fourth, 5=Other

Job Current employment status: 1=Not Working, 2=Part-time Job, 3=Full-time Job

MathAnxious Agreement with the statement "In general I tend to feel very anxious about mathematics": 1=Strongly Disagree, 2=Disagree, 3=Neither Agree nor Disagree, 4=Agree, 5=Strongly Agree

Interest Interest in statistics and the course: 1=No Interest, 2=Somewhat Interested, 3=Very Interested

GradePredict Numeric prediction for final grade in the course. The value is converted from the student's letter grade prediction. 4.0=A, 3.7=A-, 3.3=B+, 3.0=B, 2.7=B-, 2.3=C+, 2.0=C, 1.7=C-, 1.3=Below C-

Thumb Length in mm from tip of thumb to the crease between the thumb and palm.

Index Length in mm from tip of index finger to the crease between the index finger and palm.

Middle Length in mm from tip of middle finger to the crease between the middle finger and palm.

Ring Length in mm from tip of ring finger to the crease between the middle finger and palm.

Pinkie Length in mm from tip of pinkie finger to the crease between the pinkie finger and palm.

Height Height in inches.

Weight Weight in pounds.

Sex Sex of participant.

fit_stats	<i>Test the fit of a model on a train and test set.</i>
-----------	---

Description

Test the fit of a model on a train and test set.

Usage

```
fit_stats(model, df_train, df_test)
```

```
fitstats(model, df_train, df_test)
```

Arguments

model	An <code>lm</code> model.
df_train	A data frame with the training data.
df_test	A data frame with the test data.

Value

A data frame with the fit statistics.

Examples

```
set.seed(123)
parts <- split_data(Fingers)
model <- lm(Thumb ~ Height, data = parts$train)
fit_stats(model, parts$train, parts$test)
fitstats(model, parts$train, parts$test)
```

game_data	<i>Simulated math game data.</i>
-----------	----------------------------------

Description

The simulated results of a small study comparing the effectiveness of three different computer-based math games in a sample of 105 fifth-grade students. All three games focused on the same topic and had identical learning goals, and none of the students had any prior knowledge of the topic.

Usage

```
game_data
```

Format

A data frame with 105 observations on the following 2 variables:

game The game the student was randomly assigned to, coded as "A", "B", or "C".

outcome Each student's score on the outcome test.

GeomCutoff

Draw short cutoff stems inside a panel

Description

GeomCutoff draws a cutoff at each supplied xintercept. The statistical position is transformed by the active coordinate system, while height is measured as a fraction of the panel. The result is a short stem that starts at the distribution baseline and follows `ggplot2::coord_flip()` without changing the orthogonal scale. Optional label, side, and call_id aesthetics let a teaching helper add measured callouts without training the count axis. When the x scale is positioned at the top, the stem starts at that panel edge and points inward instead.

Usage

GeomCutoff

```
geom_cutoff(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  height = 0.2,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by <code>ggplot2::aes()</code> .
data	The data to be displayed in this layer.
stat	The statistical transformation to use. Defaults to "identity".
position	A position adjustment. Defaults to "identity".
...	Other arguments passed to <code>ggplot2::layer()</code> .
height	Stem height as a fraction of the panel, from 0 to 1.
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 legends?
inherit.aes	If FALSE, override the default aesthetics rather than combining with them.

Format

GeomCutoff is a [ggplot2::Geom](#) object.

Details

`geom_cutoff()` is the conventional layer constructor. It does not compute cutoffs. [show_cutoffs\(\)](#) supplies the optional marker and callout metadata after it computes one whole-distribution plan.

Value

`geom_cutoff()` returns a ggplot2 layer.

GeomModel

Draw the representation produced by StatModel

Description

GeomModel is a narrow dispatcher. The statistical representation chooses a line, group mark, or intercept; the corresponding ggplot2 geom does the drawing.

Usage

GeomModel

Format

GeomModel is a [ggplot2::Geom](#) object.

GeomResid

Draw a residual as a segment from a prediction to an observation

Description

The segment is drawn from what the model predicts to what was observed, so `x/y` is the observation and `xend/yend` the prediction until the moment of drawing. Which of the two ends arrives says which axis the residual is measured on. [geom_resid\(\)](#) pairs it with [StatResid](#) and computes those endpoints from a fitted model.

Usage

GeomResid

Format

A [ggplot2::Geom](#) object.

See Also

[gf_resid\(\)](#), the ggformula function for the same layer.

GeomSquareplot

Draw one countable rectangle per observation

Description

Re-expands each bin from [StatSquareplot](#) into one unit rectangle per observation, stacked. The rectangles are one unit tall in data units, so the y axis is a real count. Pair it with [StatSquareplot](#) in a `ggplot2::layer()` to draw the squares yourself, where a mapped fill stacks its groups within each bin.

Usage

GeomSquareplot

Format

A `ggplot2::Geom` object.

Details

The geom takes either a bin's edges (`xmin/xmax`, from `ggplot2::stat_bin()`) or a level's position plus a column width (`x/width`, from `ggplot2::stat_count()`), deriving the edges it was not given. Paired with `stat = "count"`, it draws one countable column per category the way `geom_bar()` draws one bar.

The `bars` parameter chooses what a bin is drawn as, on one grid and in one layer: "none" draws the squares, "outline" frames them with the bar they add up to, and "solid" draws that bar with the squares covered over. The bar is derived from the squares rather than re-binned, so a bin holding no observations has no bar, and a solid bar stacks a mapped fill in the same spans its squares did. `bar_color` and `bar_linewidth` are the bar's own color and width; `color` and `linewidth` are the separators between the squares, and either may be mapped.

See Also

[gf_squareplot\(\)](#), which pairs this stat and geom for you.

GeomSquareResid	<i>Draw a residual as the square it would make</i>
-----------------	--

Description

The square is expanded here rather than in [StatResid](#) because a stat runs before the position: four corners jittered one row at a time tear apart. It is also the only place the panel's final ranges are known, and the square is a square on the page rather than in data units.

Usage

```
GeomSquareResid
```

Format

A [ggplot2::Geom](#) object.

See Also

[geom_square_resid\(\)](#) and [gf_square_resid\(\)](#).

geom_b	<i>Annotate model coefficients with ggplot2</i>
--------	---

Description

Add an intercept and a rise-over-run triangle, or arrows for group differences. Marks use tagged segment, point, and text layers. See [gf_b\(\)](#) for the mark geometry and appearance controls.

Usage

```
geom_b(
  mapping = NULL,
  data = NULL,
  ...,
  model = NULL,
  color = "#b599ed",
  label_color = "black",
  label_size = 3.5,
  arrow_linewidth = 0.5,
  show_b0 = TRUE,
  run = NULL,
  run_x = NULL,
  b0_alpha = 0.3,
  b0_linewidth = 0.8,
  b0_size = 4,
```

```

    arrow_nudge = 0.18,
    label_nudge = 0.08,
    inherit.aes = TRUE
  )

```

Arguments

mapping, data	Optional observation mapping and data. These select the source for this annotation without changing the other layers' source. With inherited data, the annotation reuses the visible source's resolved rows and matching positional mappings. Explicit data is independent and can also be a function of the plot data.
...	Unsupported appearance arguments produce a warning. British spellings colour and label_colour are accepted.
model	A model fit by <code>stats::lm()</code> or <code>stats::aov()</code> , or NULL to use the plot's model.
color, label_color, label_size, arrow_linewidth, show_b0, run, run_x	See <code>gf_b()</code> .
b0_alpha, b0_linewidth, b0_size, arrow_nudge, label_nudge	See <code>gf_b()</code> .
inherit.aes	Inherit the observation source's mappings.

Details

This constructor returns a composite annotation. On addition it selects the observation source and model layer; its component stats calculate the marks when the plot builds. The model stat supplies its prediction grid once per build. Adding the annotation does not evaluate data callbacks or fit a model. There is no separate `stat_b()` entry point because an ordinary stat cannot select a sibling model layer or compose the different drawing geoms.

A supplied model keeps its fitted coefficients. With no model or local source, the annotation uses the plotted model, or computes an implied model through `StatModel` when no model layer is present. An explicit mapping or data selects the annotation's own source instead of a sibling model. Scale transformations and hard limits then affect the inferred fit just as they affect `geom_model()`. `run` and `run_x` use those transformed units for an inferred model. Scales and limits can be added before or after the annotation. Coordinate zooming does not change the fitted rows. Replacing plot data updates inferred coefficients and annotation placement; a supplied fit keeps its coefficients.

Inference requires one model in one panel. Pass a fitted model to annotate a faceted plot. A non-linear curve has no single slope to annotate. A plotted model must use the identity position: a positional adjustment moves its marks away from the fitted claim.

Value

A composite annotation to add to a ggplot.

Examples

```

fit <- lm(Thumb ~ Height, data = Fingers)
ggplot2::ggplot(Fingers, ggplot2::aes(Height, Thumb)) +
  ggplot2::geom_point() + geom_model(model = fit) + geom_b(model = fit)

```

geom_model

*Draw fitted and implied models with ggplot2***Description**

`geom_model()` adds a model to an ordinary `ggplot2::ggplot()`. A supplied model may be a fit from `stats::lm()` or `stats::aov()`, or a two-sided formula that is fitted once against the layer data. With no model, the layer draws the model implied by its mapped positions, separately in each panel and group.

Usage

```
geom_model(
  mapping = NULL,
  data = NULL,
  stat = "model",
  position = "identity",
  ...,
  model = NULL,
  orientation = NA,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_model(
  mapping = NULL,
  data = NULL,
  geom = "model",
  position = "identity",
  ...,
  model = NULL,
  orientation = NA,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

`mapping`, `data`, `position`, `show.legend`, `inherit.aes`

See `ggplot2::geom_smooth()`. `data` may be a data frame, a function, or a one-sided formula.

`stat` The statistical transformation. Defaults to "model".

... Fixed aesthetics and other layer parameters. For an inferred continuous model these include `formula`, `se`, `n`, `fullrange`, `level`, and `method.args`, with the

	meanings used by <code>ggplot2::stat_smooth()</code> . <code>width</code> controls categorical model marks. With a supplied model, <code>n</code> controls the prediction grid. <code>se = TRUE</code> requires an inferred continuous model; it is refused when <code>model</code> is supplied.
<code>model</code>	A fitted <code>lm</code> or <code>aov</code> , a two-sided model formula, or <code>NULL</code> to draw the model implied by the mapped positions.
<code>orientation</code>	Layer orientation. <code>"x"</code> puts the outcome on y; <code>"y"</code> puts it on x. <code>NA</code> uses the mapped model outcome for a supplied model, and <code>ggplot2</code> 's orientation rules for an inferred model.
<code>na.rm</code>	If <code>FALSE</code> , missing values are removed with a warning. If <code>TRUE</code> , they are removed silently.
<code>geom</code>	The geometric object. <code>stat_model()</code> defaults to <code>"model"</code> .

Details

Empty models draw an intercept, numeric predictors draw fitted lines, and categorical predictors draw one short mark per group. A model with one numeric and one categorical predictor draws one line per category.

A supplied model is one fixed claim evaluated on a prediction grid built from the layer data.

The displayed predictor is read from the plot mapping or this layer's local mapping. One additional predictor may be present in the data: categories produce separate traces, and numeric values use the mean and mean plus or minus one standard deviation. Outcome-axis expressions such as `log(y)` are applied to the predictions. A supplied model is prepared once when added to the plot, including when data is a function or formula.

Override unrelated inherited aesthetics locally, or set `inherit.aes = FALSE` and supply a predictor mapping such as `aes(x = Height)`. Without explicit data or positions, the layer follows the first observation layer, preferring points, so a model describes the rows and axes the plot actually shows.

Value

A `ggplot2` layer.

Examples

```
fit <- lm(Thumb ~ Height, data = Fingers)
ggplot2::ggplot(Fingers, ggplot2::aes(Height, Thumb)) +
  ggplot2::geom_point() +
  geom_model(model = fit)

group_fit <- lm(Thumb ~ Sex, data = Fingers)
ggplot2::ggplot(Fingers, ggplot2::aes(Sex, Thumb)) +
  ggplot2::geom_jitter(width = 0.1) +
  geom_model(model = group_fit)
```

geom_resid*Residual and reduction layers for ggplot2*

Description

These layers measure a fitted model directly from an ordinary `ggplot2::ggplot()`. A residual segment runs from the model's prediction to the observed value; an arrow at its last end points to the observation. A reduction segment runs from the prediction to the model's grand mean. The square variants draw areas proportional to those squared distances at a shared aspect ratio.

Usage

```
geom_resid(  
  mapping = NULL,  
  data = NULL,  
  stat = "resid",  
  position = "identity",  
  ...,  
  model = NULL,  
  fun = NULL,  
  orientation = NA,  
  linewidth = 0.2,  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE  
)
```

```
geom_square_resid(  
  mapping = NULL,  
  data = NULL,  
  stat = "resid",  
  position = "identity",  
  ...,  
  model = NULL,  
  fun = NULL,  
  orientation = NA,  
  aspect = 4/6,  
  alpha = 0.1,  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE  
)
```

```
geom_reduce(  
  mapping = NULL,  
  data = NULL,  
  stat = "reduce",
```

```
position = "identity",
...,
model = NULL,
fun = NULL,
orientation = NA,
linewidth = 0.2,
na.rm = FALSE,
show.legend = NA,
inherit.aes = TRUE
)
```

```
geom_square_reduce(
  mapping = NULL,
  data = NULL,
  stat = "reduce",
  position = "identity",
  ...,
  model = NULL,
  fun = NULL,
  orientation = NA,
  aspect = 4/6,
  alpha = 0.1,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

```
stat_resid(
  mapping = NULL,
  data = NULL,
  geom = "resid",
  position = "identity",
  ...,
  model = NULL,
  fun = NULL,
  orientation = NA,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

```
stat_reduce(
  mapping = NULL,
  data = NULL,
  geom = "resid",
  position = "identity",
  ...,
  model = NULL,
```

```

    fun = NULL,
    orientation = NA,
    na.rm = FALSE,
    show.legend = NA,
    inherit.aes = TRUE
  )

```

Arguments

mapping, data, position, show.legend, inherit.aes	See <code>ggplot2::geom_segment()</code> . data may also be a function or one-sided formula; predictions are added after that function has selected its rows.
stat	The statistical transformation to use. The geom constructors use "resid" or "reduce" by default.
...	Other arguments passed to <code>ggplot2::layer()</code> . These are usually fixed aesthetics such as colour, fill, alpha, or linetype.
model	A model already fit by <code>stats::lm()</code> or <code>stats::aov()</code> . Name this argument in a ggplot2 call, as in <code>geom_resid(model = fit)</code> .
fun	For residual layers, a function instead of model. It receives the mapped predictor values and returns predicted outcomes. The default orientation predicts y from x; <code>orientation = "y"</code> predicts x from y. Supply only one of model and fun. Reductions require a fitted model.
orientation	Layer orientation. NA infers a model's outcome axis; "x" puts it on y and "y" puts it on x. For fun, NA means "x".
linewidth	The line width. The default is 0.2.
na.rm	If FALSE, the default, missing observations are removed with a warning. If TRUE, they are removed silently.
aspect	The square's aspect ratio. The default is 4 / 6.
alpha	The square's transparency. The default is 0.1.
geom	The geometric object to use. The stat constructors use "resid" by default; use "square_resid" to draw areas.

Details

When the observations have their own data or mappings, these layers follow the first point layer (or the first non-annotation layer when there are no points). Explicit layer data and mapping arguments take precedence.

With `orientation = NA`, a model's outcome determines the direction. The outcome expression must match an axis exactly: `log(y)` is refused for a model of y, because that distance is not the model's residual. An explicit `orientation = "x"` puts the outcome on y; "y" puts it on x and must agree with that mapping. Coordinate systems such as `ggplot2::coord_flip()` are applied later and do not change this argument.

A jittered point layer and its model layer must use the same `ggplot2::position_jitter()` object with a numeric seed. The model layer keeps the fitted endpoint fixed while moving the observed endpoint by the same amount as its point.

Positional expressions are evaluated reproducibly on the current data, so random mappings give the observations and residuals the same coordinates. Native layers still respond to later data and mapping changes. A new unseeded random mapping must be added before the residual layer, or carry its own fixed seed. Checks that need the rows or mappings run when the plot is built: axes, prediction, outcome axis, then reduction eligibility.

Native square layers inherit mapped aesthetics by default, including colour. `gf_square_resid()` and `gf_square_reduce()` default to `inherit = FALSE` for neutral square outlines. Set their `inherit = TRUE` to match this API.

Reduction layers express the ordinary least-squares sum-of-squares identity. They require an un-weighted model with an intercept, no offset, and its stored model frame. On the fitted observations, the identity holds across the sums of the square areas, not separately for each observation or for new prediction data.

Value

A ggplot2 layer.

Examples

```
model <- lm(Thumb ~ Height, data = Fingers)
ggplot2::ggplot(Fingers, ggplot2::aes(Height, Thumb)) +
  ggplot2::geom_point() +
  geom_resid(model = model, colour = "firebrick")

ggplot2::ggplot(Fingers, ggplot2::aes(Height, Thumb)) +
  ggplot2::geom_point() +
  geom_square_reduce(model = model, fill = "forestgreen")

jitter <- ggplot2::position_jitter(width = 0.1, seed = 42)
group_model <- lm(Thumb ~ Sex, data = Fingers)
ggplot2::ggplot(Fingers, ggplot2::aes(Sex, Thumb)) +
  ggplot2::geom_point(position = jitter) +
  geom_resid(model = group_model, position = jitter)
```

geom_squareplot

Draw a countable squareplot layer

Description

`geom_squareplot()` and `stat_squareplot()` expose `gf_squareplot()` through ggplot2 syntax. With the default `bars = "none"`, each observation is drawn as one square. Continuous x values use `ggplot2::stat_bin()`'s binning rules; factors, characters, and logicals use `ggplot2::stat_count()`'s columns. Both constructors retain `gf_squareplot()`'s defaults, warnings, factor-level handling, and count scale. Counts must be finite, nonnegative whole numbers. Integer frequency weights are supported; use a histogram or bar plot for fractional weighted counts. Fill and separator colour inherit mappings, with teal and white defaults when no mapping or fixed value is supplied. Put a staged `linewidth = after_scale(...)` mapping on this layer to keep its width fixed. Plot-level

after_scale() linewidth mappings use automatic border fitting: ggplot2 evaluates them after the geom's setup.

Usage

```
geom_squareplot(  
  mapping = NULL,  
  data = NULL,  
  stat = "squareplot",  
  position = "identity",  
  ...,  
  binwidth = NULL,  
  bins = NULL,  
  center = NULL,  
  boundary = NULL,  
  closed = NULL,  
  breaks = NULL,  
  bars = "none",  
  na.rm = TRUE,  
  show.legend = NA,  
  inherit.aes = TRUE  
)
```

```
stat_squareplot(  
  mapping = NULL,  
  data = NULL,  
  geom = "squareplot",  
  position = "identity",  
  ...,  
  binwidth = NULL,  
  bins = NULL,  
  center = NULL,  
  boundary = NULL,  
  closed = NULL,  
  breaks = NULL,  
  bars = "none",  
  na.rm = TRUE,  
  show.legend = NA,  
  inherit.aes = TRUE  
)
```

Arguments

mapping	Aesthetic mappings created by <code>ggplot2::aes()</code> .
data	A data frame for this layer, or NULL to inherit the plot's data. If x is mapped only on the plot's first layer, its data and mapping are used together. Function-valued and formula-valued layer data are not supported.
stat	The statistical transformation used by <code>geom_squareplot()</code> .

position, show.legend, inherit.aes
 See `ggplot2::geom_histogram()`.

... Fixed aesthetics and other layer parameters. These include pad from `ggplot2::stat_bin()`.
 bar_color and bar_linewidth control the bar drawn by bars = "outline" or
 bars = "solid".

binwidth, bins, center, boundary, closed, breaks
 Binning parameters with the meanings used by `ggplot2::stat_bin()`. They
 have no effect on a discrete x, which is counted instead.

bars Display style: "none" draws squares, "outline" frames them with their bars,
 and "solid" draws the bars alone.

na.rm Must be TRUE. A missing value has no square to draw.

geom The geometric object used by `stat_squareplot()`.

Details

A squareplot is x-only, as it is through `gf_squareplot()`. Map the distribution to x. If the plot
 already maps y for another layer, set `inherit.aes = FALSE` and supply this layer's x mapping di-
 rectly.

Value

A `ggplot2` layer.

See Also

`gf_squareplot()`

Examples

```
ggplot2::ggplot(Fingers, ggplot2::aes(Thumb)) +  
  geom_squareplot()  
  
ratings <- data.frame(rating = factor(c(1, 1, 3), levels = 1:5))  
ggplot2::ggplot(ratings, ggplot2::aes(rating)) +  
  stat_squareplot()
```

gf_b

Annotate a model's coefficients on a plot

Description

Draws the intercept and slope (or group differences) of a fitted model as arrows and labels directly
 on the plot they describe: a rise-over-run triangle for a continuous predictor, an arrow to each non-
 reference group for a categorical one. Where `gf_model()` draws the fit itself, `gf_b()` draws the
 numbers that describe it.

`gf_coef()` is a fully supported alias of `gf_b()`. The package already exports `b()`, `b0()`, `b1()` as
 its vocabulary for coefficients, and a reader who knows `stats::coef()` will look for a plot-side
 counterpart under that name.

Usage

```
gf_b(  
  object = NULL,  
  gformula = NULL,  
  data = NULL,  
  ...,  
  model,  
  color = "#b599ed",  
  label_color = "black",  
  label_size = 3.5,  
  arrow_linewidth = 0.5,  
  show_b0 = TRUE,  
  run = NULL,  
  run_x = NULL,  
  b0_alpha = 0.3,  
  b0_linewidth = 0.8,  
  b0_size = 4,  
  arrow_nudge = 0.18,  
  label_nudge = 0.08,  
  xlab,  
  ylab,  
  title,  
  subtitle,  
  caption,  
  geom = ggplot2::GeomSegment,  
  stat = "identity",  
  position = "identity",  
  show.legend = NA,  
  show.help = NULL,  
  inherit = TRUE,  
  environment = parent.frame()  
)
```

```
gf_coef(  
  object = NULL,  
  gformula = NULL,  
  data = NULL,  
  ...,  
  model,  
  color = "#b599ed",  
  label_color = "black",  
  label_size = 3.5,  
  arrow_linewidth = 0.5,  
  show_b0 = TRUE,  
  run = NULL,  
  run_x = NULL,  
  b0_alpha = 0.3,  
  b0_linewidth = 0.8,
```

```

b0_size = 4,
arrow_nudge = 0.18,
label_nudge = 0.08,
xlab,
ylab,
title,
subtitle,
caption,
geom = ggplot2::GeomSegment,
stat = "identity",
position = "identity",
show.legend = NA,
show.help = NULL,
inherit = TRUE,
environment = parent.frame()
)

```

Arguments

object	A plot created with the ggformula package.
gformula	Not used. <code>gf_b()</code> annotates a model, not an aesthetic formula; a model given positionally lands here and is moved to <code>model</code> .
data	Not used. The marks are placed from the model's own coefficients and data.
...	Not used. Every mark states its own geom and params; set appearance with <code>color</code> , <code>label_color</code> , <code>label_size</code> , <code>arrow_linewidth</code> , <code>b0_linewidth</code> , <code>b0_size</code> , <code>b0_alpha</code> . Anything else here (<code>alpha</code> , <code>linetype</code> , ...) is warned about and dropped, because the marks are heterogeneous geoms with no single params bag to receive it.
model	The model to annotate: a fit from <code>lm()</code> or <code>aov()</code>, with one predictor at most, and that predictor spelled the way the plot's own axis spells it – <code>log(Height)</code> and <code>Height</code> are the same column but not the same axis, and <code>b1</code> is a rise per unit of whichever one the model was fit on. It needs an intercept, because every mark here is measured from <code>b0</code>; a categorical predictor needs treatment coding, because every arrow is drawn as one group's difference from the reference group and no other coding's coefficients are that. Each of those is refused rather than drawn, because each would otherwise produce a picture that looks right. A formula is refused too – <code>gf_b()</code>'s whole output is a set of labeled numbers, and there is no fit to read them from. May be given positionally or as <code>model =</code>. Omitted, the model the plot implies is fit and annotated instead. Coefficient marks require Cartesian coordinates. Reversed position scales, <code>coord_cartesian(reverse =)</code>, and <code>coord_flip()</code> are supported; polar and other non-Cartesian coordinates are refused because rise, run, and label sides lack a linear coefficient interpretation there.
color, label_color	The arrows/lines and the label text. <code>colour</code> and <code>label_colour</code> are accepted too. Each is a single value, not a mapping – every mark is one row computed from the coefficients, so there are no rows of data to map an aesthetic over; <code>color = ~variable</code> is refused.

label_size, arrow_linewidth, b0_linewidth, b0_size	Sizes for the labels, the arrows, the b0 line and the b0 dot.
show_b0	Draw the b0 line/dot and its label, and expand the PREDICTOR's axis to include 0 on a continuous model – x on most plots, y on one that puts the outcome on x. TRUE by default; a later <code>gf_lims()</code> on that axis overrides the expansion and can push the b0 dot off the page.
run, run_x	The run a continuous model's rise is measured over, and the x position the triangle starts at. Both chosen from the data when left NULL. Naming run on a categorical model is warned about and ignored – its coefficients are group differences, not a rate.
b0_alpha	The transparency of the categorical b0 reference line.
arrow_nudge	A categorical arrow's x position in level units (1 = one group apart).
label_nudge	The predictor-axis clearance for coefficient labels. It uses level units for a categorical predictor and adjusts the physical gap beside the b0 and rise marks for a continuous predictor. Not used on the empty model, whose one axis is a count rather than a predictor and whose b0 label is placed at the panel's edge instead.
xlab, ylab, title, subtitle, caption	Labels for the plot.
geom, stat, position	Not set by the caller. Every mark states its own geom.
show.legend	Not used. The marks are annotations and never contribute to a legend; a non-default value is warned about and dropped.
show.help	Print the function's own help instead of drawing.
inherit	Not set by the caller. Every mark states its own aesthetics.
environment	The environment mappings are resolved in.

Value

A ggplot object with the model's coefficients annotated on it.

What is drawn

A continuous predictor: a vertical rise arrow from `fit(run_x)` to `fit(run_x + run)`, a horizontal run segment at its tip, a rise label (`plotmath b1` when `run` is 1, otherwise `run times b1`), a run-distance label on the triangle's interior side of the run segment, and a hollow dot at $(0, b0)$ with a `b0` label.

A categorical predictor: one horizontal reference line at `b0` (the reference level's mean), and for each level after it a segment from `b0` to `b0` plus that level's coefficient, with an arrow head, labeled (`plotmath`) `b1`, `b2`, ... Level order is read off `coef(model)`, so a relevelled factor still labels the arrow that matches its coefficient.

No predictor (the empty model): the `b0` line and its label, nothing else.

Layers have stable role tags such as `"b0"`, `"b1"`, `"run"`, and their `"_label"` counterparts. Categorical annotations use numbered roles plus shared `"bk_more"` and `"bk_more_label"` layers for additional groups. Replacement data can add groups without changing the layer list. A role can be empty when the model does not need it.

No model

With no model, `gf_b()` reads the model the plot implies – the same decision `gf_model()` uses. The marks read that model's prediction grid, after position scales transform values and remove rows outside hard limits. If no model layer exists, the same model stat supplies the predictions. Inference is refused on a faceted plot because its panels have different fits. Position scales and `gf_lims()` can precede or follow the annotation. For an inferred model on transformed scales, `run` and `run_x` use transformed units. `geom_b()` provides the same annotation with `ggplot2` syntax.

Placement

Every mark is placed from the model's coefficients and from level indices, never from a drawn point's position, so jitter never moves an arrow.

`show_b0 = TRUE` (the default) expands the PREDICTOR's axis to include 0 on a continuous model, because `b0` is the prediction where the predictor is 0 and a picture of it that does not show that point is not a picture of `b0`. Usually that is `x`; on a plot that puts the outcome on `x` it is `y`, and the expansion follows the predictor rather than the letter. Calling `gf_lims()` on that axis afterward overrides the expansion and can push the `b0` dot off the page.

See Also

`gf_model()` draws the fit itself.

Examples

```
# continuous: b1 as a rise-over-run triangle, b0 where the line meets x = 0
height_model <- lm(Thumb ~ Height, data = Fingers)
gf_point(Thumb ~ Height, data = Fingers, alpha = .3) %>% gf_b(height_model)

# the slope per one unit
gf_point(Thumb ~ Height, data = Fingers) %>% gf_b(height_model, run = 1)

# an explicit run labels the rise "10 x b1"
gf_point(Thumb ~ Height, data = Fingers) %>% gf_b(height_model, run = 10)

# categorical: b0 is the reference group's mean, each b_k is an arrow to group k
tip_model <- lm(Tip ~ Condition, data = TipExperiment)
gf_jitter(Tip ~ Condition, data = TipExperiment, width = .1) %>% gf_b(tip_model)

# no model: the model the plot implies, on the values the plot drew
set.seed(1)
gf_jitter(shuffle(Height) ~ Sex, data = Fingers, width = .1) %>%
  gf_model() %>%
  gf_b()

# gf_coef() is the same function under the name coef() readers look for
flipper_model <- lm(body_mass_kg ~ flipper_length_m, data = penguins)
gf_point(body_mass_kg ~ flipper_length_m, data = penguins) %>%
  gf_coef(flipper_model)
```

gf_model

*Add a model to a plot***Description**

When teaching about regression it can be useful to visualize the data as a point plot with the outcome on the y-axis and the explanatory variable on the x-axis. For regression models, this is most easily achieved by calling `ggformula::gf_lm()`, with empty models `ggformula::gf_hline()` using the mean, and a more complicated call to `ggformula::gf_segment()` for group models. This function simplifies this by making a guess about what kind of model you are plotting (empty/null, regression, group) and then making the appropriate plot layer for it.

Usage

```
gf_model(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  model,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "model",
  stat = "model",
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)
```

Arguments

<code>object</code>	A plot created with the ggformula package.
<code>gformula</code>	Not used. <code>gf_model()</code> draws a model, not an aesthetic formula; a model given positionally lands here and is moved to <code>model</code> .
<code>data</code>	Not used. The layer draws the model's predictions, which are computed from the data the plot was built from.
<code>...</code>	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~expression</code> , or (c) attributes of the layer as a whole, which are set with <code>attribute = value</code> .

With no model, and a plot whose implied shape is a regression line, ... also reaches the fitting vocabulary `ggformula::gf_lm()` uses: `se = TRUE` draws the confidence band `gf_lm()` calls `interval = "confidence"`, `n` = sets the prediction grid's length, and `method.args` = is `gf_lm()`'s `lm.args` =. `formula = y ~ poly(x, 2)` fits a curve, written with the literal tokens `x` and `y` rather than the plot's own variable names – a one-sided `formula = ~x` would instead be read as a mapping, so it is not this.

With a supplied model, `n` controls the prediction grid and `se = TRUE` is refused. One additional predictor may be present in the data without being mapped: its categories, or its mean and mean plus or minus one standard deviation, define separate model traces.

`ggplot2::stat_smooth()`'s `fullrange = TRUE` reaches it too, and draws the line past the data. Whether that is honest is yours to decide and not something this function will decide for you: inside the range the model was fit on, every point the line interpolates has observations on both sides of it, and outside there are none. Extending it says the pattern keeps holding where nothing was measured, which needs a theory or a physical constraint behind it. Nothing about the plot's own axis is such a reason, which is why a wider axis – from `gf_lims()`, or from the `b0` dot `gf_b()` puts at zero – does not lengthen the line on its own.

model The model to draw. Either a model already fit by `lm()` or `aov()`, or the formula for one – such as `body_mass_kg ~ species` – which is fit against the data the plot was built from. The empty model is written `body_mass_kg ~ NULL`. The outcome must be named: a one-sided formula such as `~species` names predictors and no claim, so there would be nothing to draw that you had not described. May be given positionally or as `model =`. Omitted, the model the plot implies is drawn instead: a regression line for a numeric predictor, one group mean per level for a categorical one, and the grand mean when the plot draws only an outcome. `gf_model(model)` draws the ONE model you named in every panel of a faceted plot; `gf_model()` draws EACH panel's own implied model – a faceted `gf_point(body_mass_kg ~ flipper_length_m | species)` `%>% gf_model()` fits a different line per species, where naming `flipper_model` above would repeat one whole-data fit in every panel.

xlab, ylab, title, subtitle, caption

Labels for the plot.

geom Not set by the caller. The geometry is derived from the model.

stat, position With a named model, reach the layer as given, but `gf_model()` computes the model's predictions before the layer is built. Changing the `stat` recomputes something else on that prediction grid rather than changing the model's claim, so leave these at their defaults. With no model, the shared `stat_model()` is used and these are not read.

show.legend Whether this layer contributes to the legend.

show.help Print the layer's own help instead of drawing.

inherit Not set by the caller. A named intercept stands alone; other named models inherit compatible plot aesthetics while stating their own predicted outcome. With no model, the layer states its own positions and does not inherit a color grouping the model never named.

environment The environment mappings are resolved in.

Details

This function only works with models that have a continuous outcome measure.

Value

A ggplot object with the model added. With no model, and a plot whose positional mapping is an expression rather than a bare variable (`shuffle(body_mass_kg)`, `log(flipper_length_m)`), the RETURNED plot is pinned to the values it drew when `gf_model()` was called – its data gains a fixed column and its mapping names it, while its axis titles and everything else about how it reads keep your own words. The plot passed IN is untouched.

Supported plots

`gf_model()` is built for and tested against plots made with `ggformula::gf_point()`, `ggformula::gf_jitter()`, `ggformula::gf_boxplot()`, `ggformula::gf_violin()` and `ggformula::gf_histogram()`. Other plots may work if they map their variables the same way, but they are not tested.

Examples

```
# the empty model predicts the same value (the mean) for every observation
empty_model <- lm(body_mass_kg ~ NULL, data = penguins)
gf_histogram(~body_mass_kg, data = penguins, binwidth = 0.25) %>%
  gf_model(empty_model)

# a two-group model (categorical explanatory variable) on a jitter plot
gentoo_model <- lm(body_mass_kg ~ gentoo, data = penguins)
gf_jitter(body_mass_kg ~ gentoo, data = penguins, width = .1) %>%
  gf_model(gentoo_model)

# a three-group model works the same way
species_model <- lm(body_mass_kg ~ species, data = penguins)
gf_jitter(body_mass_kg ~ species, data = penguins, width = .1) %>%
  gf_model(species_model)

# group models can also be layered onto faceted histograms
gf_histogram(~body_mass_kg, data = penguins, binwidth = 0.25) %>%
  gf_facet_grid(species ~ .) %>%
  gf_model(species_model)

# a regression model (quantitative explanatory variable) on a scatter plot
flipper_model <- lm(body_mass_kg ~ flipper_length_m, data = penguins)
gf_point(body_mass_kg ~ flipper_length_m, data = penguins) %>%
  gf_model(flipper_model)

# layer the empty model and the regression model in different colors to
# compare the two models on the same plot
gf_point(body_mass_kg ~ flipper_length_m, data = penguins) %>%
  gf_model(empty_model, color = "dodgerblue") %>%
  gf_model(flipper_model, color = "firebrick")

# with a categorical and a quantitative predictor, the model is drawn
```

```

# as one line for each group
ancova_model <- lm(body_mass_kg ~ species + flipper_length_m, data = penguins)
gf_point(body_mass_kg ~ flipper_length_m, color = ~species, data = penguins) %>%
  gf_model(ancova_model)

# a model that has not been fit yet can be written as a formula, and is fit
# against the data the plot was built from
gf_point(body_mass_kg ~ flipper_length_m, data = penguins) %>%
  gf_model(body_mass_kg ~ flipper_length_m)

# the empty model, written as a formula
gf_histogram(~body_mass_kg, data = penguins, binwidth = 0.25) %>%
  gf_model(body_mass_kg ~ NULL)

# with no model, gf_model() draws the model the plot implies: a numeric
# predictor draws the regression line gf_lm() would fit
gf_point(body_mass_kg ~ flipper_length_m, data = penguins) %>%
  gf_model()

# a categorical predictor draws one mark at each group's mean
gf_jitter(body_mass_kg ~ species, data = penguins, width = .1) %>%
  gf_model()

# a plot that draws only its outcome implies the grand mean
gf_histogram(~body_mass_kg, data = penguins, binwidth = 0.25) %>%
  gf_model()

```

gf_reduce

Add Reduction Lines to a Plot

Description

Draws reduction lines from the value a fitted model predicts for each observation to the grand mean. Squaring and summing those lengths across observations gives the model sum of squares; [gf_resid\(\)](#) supplies the error term in the same decomposition. Each line runs along whichever axis the plot puts the model's outcome on, so a model of the variable drawn on x is measured across x rather than down y.

Usage

```

gf_reduce(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  model,
  linewidth = 0.2,
  xlab,
  ylab,

```

```

    title,
    subtitle,
    caption,
    geom = coursekata::GeomResid,
    stat = coursekata::StatReduce,
    position = "identity",
    show.legend = NA,
    show.help = NULL,
    inherit = TRUE,
    environment = parent.frame()
  )

```

Arguments

object	A ggformula plot object, typically created with <code>gf_point()</code> .
gformula	Not used. <code>gf_reduce()</code> measures a model, not an aesthetic formula; a model given positionally lands here and is moved to <code>model</code> .
data	Not used. The reductions are measured over the data the plot was built from. Anything supplied here is left for ggformula and ggplot2 to answer, exactly as it is for any other <code>gf_</code> layer.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , such as <code>color</code> , <code>alpha</code> or <code>linetype</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~ expression</code> , or (c) attributes of the layer as a whole.
model	A model already fit by <code>lm()</code> or <code>aov()</code> . The plot supplies the observations' position on the other axis; the model supplies what it predicted for each of them. May be given positionally or as <code>model =</code> . A fit without an intercept, or one fit with weights or an offset, is refused. For an unweighted least-squares fit with an intercept, the sum of squared total deviations equals the sum of squared residuals plus the sum of squared reductions. That identity need not hold without an intercept. Weighted least squares instead guarantees a weighted identity based on a weighted mean, which the plot's plain areas do not represent. <code>gf_resid()</code> can measure either fit because it does not rely on the decomposition.
linewidth	The width of the reduction lines. Default is 0.2. Must be named.
xlab, ylab, title, subtitle, caption	Labels for the plot.
geom, stat, position	Not set by the caller. A reduction is drawn by its own geom and stat, with a jitter that holds the outcome axis still so its segments start at the grand mean without floating off it, while jittering the other axis exactly the points layer's own jitter did.
show.legend	Whether this layer contributes to the legend.
show.help	Print the layer's own help instead of drawing.
inherit	Whether the layer inherits the plot's aesthetics. The axes and the prediction are stated outright; everything else – a mapped color, for instance – is inherited from the plot.
environment	The environment mappings are resolved in.

Details

The grand mean is the model's own, `mean()` of the outcome column the model was fit on, not anything read off the plot's data. On a faceted plot that is one number for every panel: every panel is measured against the same line, which is what makes the picture in each panel a piece of one decomposition rather than a decomposition of its own.

Value

A ggplot object with reduction lines added.

Examples

```
set.seed(1)
penguins_20 <- sample(penguins, 20)

# the reduction: how far a model's fit moves the prediction from the grand
# mean, for a regression model
flipper_model <- lm(body_mass_kg ~ flipper_length_m, data = penguins_20)
gf_point(body_mass_kg ~ flipper_length_m, data = penguins_20) %>%
  gf_model(flipper_model) %>%
  gf_reduce(flipper_model, color = "blue")

# and for a two-group model on a jitter plot
gentoo_model <- lm(body_mass_kg ~ gentoo, data = penguins_20)
gf_jitter(body_mass_kg ~ gentoo, data = penguins_20, width = .1) %>%
  gf_model(gentoo_model) %>%
  gf_reduce(gentoo_model, color = "blue")

# each observation's signed deviation from the grand mean is its residual
# (firebrick) plus its reduction (blue)
gf_point(body_mass_kg ~ flipper_length_m, data = penguins_20) %>%
  gf_model(flipper_model) %>%
  gf_resid(flipper_model, color = "firebrick") %>%
  gf_reduce(flipper_model, color = "blue")
```

gf_resid

Add Residual Lines to a Plot

Description

Draws residual lines from the values a fitted model predicts to the observed points. Each residual runs along whichever axis the plot puts the model's outcome on, so a model of the variable drawn on x is measured across x rather than down y.

Usage

```
gf_resid(
  object = NULL,
  gformula = NULL,
```

```

data = NULL,
...,
model,
linewidth = 0.2,
xlab,
ylab,
title,
subtitle,
caption,
geom = coursekata::GeomResid,
stat = coursekata::StatResid,
position = "identity",
show.legend = NA,
show.help = NULL,
inherit = TRUE,
environment = parent.frame()
)

```

Arguments

object	A ggformula plot object, typically created with <code>gf_point()</code> .
gformula	Not used. <code>gf_resid()</code> measures a model, not an aesthetic formula; a model given positionally lands here and is moved to <code>model</code> .
data	Not used. The residuals are measured over the data the plot was built from. Anything supplied here is left for <code>ggformula</code> and <code>ggplot2</code> to answer, exactly as it is for any other <code>gf_</code> layer.
...	Additional arguments. Typically these are (a) <code>ggplot2</code> aesthetics to be set with <code>attribute = value</code> , such as <code>color</code> , <code>alpha</code> or <code>linetype</code> , (b) <code>ggplot2</code> aesthetics to be mapped with <code>attribute = ~ expression</code> , or (c) attributes of the layer as a whole.
model	A model already fit by <code>lm()</code> or <code>aov()</code> . The plot supplies the observations; the model supplies what it predicted for each of them. May be given positionally or as <code>model =</code> .
linewidth	The width of the residual lines. Default is 0.2. Must be named.
xlab, ylab, title, subtitle, caption	Labels for the plot.
geom, stat, position	Not set by the caller. A residual is drawn by its own geom and stat, and moved by the position the observations are already drawn with, so that a segment stays on the point it belongs to.
show.legend	Whether this layer contributes to the legend.
show.help	Print the layer's own help instead of drawing.
inherit	Whether the layer inherits the plot's aesthetics. The axes and the prediction are stated outright; everything else – a mapped color, for instance – is inherited from the plot.
environment	The environment mappings are resolved in.

Value

A ggplot object with residual lines added.

Examples

```
# residuals can be drawn on a full data set, but with hundreds of points
# the plot gets hard to read
flipper_model <- lm(body_mass_kg ~ flipper_length_m, data = penguins)
gf_point(body_mass_kg ~ flipper_length_m, data = penguins) %>%
  gf_model(flipper_model) %>%
  gf_resid(flipper_model)

# a small sample makes the residuals much easier to see
set.seed(1)
penguins_20 <- sample(penguins, 20)

# residuals from the empty model (in blue)
empty_model <- lm(body_mass_kg ~ NULL, data = penguins_20)
gf_point(body_mass_kg ~ flipper_length_m, data = penguins_20) %>%
  gf_model(empty_model) %>%
  gf_resid(empty_model, color = "blue")

# residuals from a two-group model on a jitter plot (in firebrick)
gentoo_model <- lm(body_mass_kg ~ gentoo, data = penguins_20)
gf_jitter(body_mass_kg ~ gentoo, data = penguins_20, width = .1) %>%
  gf_model(gentoo_model) %>%
  gf_resid(gentoo_model, color = "firebrick")

# residuals from a regression model (in firebrick)
sample_flipper_model <- lm(body_mass_kg ~ flipper_length_m, data = penguins_20)
gf_point(body_mass_kg ~ flipper_length_m, data = penguins_20) %>%
  gf_model(sample_flipper_model) %>%
  gf_resid(sample_flipper_model, color = "firebrick")
```

gf_resid_fun

Add Residual Lines from a Function to a Plot

Description

[Experimental]

Usage

```
gf_resid_fun(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  fun,
```

```

    linewidth = 0.2,
    xlab,
    ylab,
    title,
    subtitle,
    caption,
    geom = coursekata::GeomResid,
    stat = coursekata::StatResid,
    position = "identity",
    show.legend = NA,
    show.help = NULL,
    inherit = TRUE,
    environment = parent.frame()
  )

```

Arguments

object	A ggformula plot object, typically created with <code>gf_point()</code> .
gformula	Not used. <code>gf_resid_fun()</code> measures a function, not an aesthetic formula; a function given positionally lands here and is moved to <code>fun</code> .
data	Not used. The residuals are measured over the data the plot was built from. Anything supplied here is left for <code>ggformula</code> and <code>ggplot2</code> to answer, exactly as it is for any other <code>gf_</code> layer.
...	Additional arguments. Typically these are (a) <code>ggplot2</code> aesthetics to be set with <code>attribute = value</code> , such as <code>color</code> , <code>alpha</code> or <code>linetype</code> , (b) <code>ggplot2</code> aesthetics to be mapped with <code>attribute = ~ expression</code> , or (c) attributes of the layer as a whole.
fun	A function of one argument. It is called on the <code>x</code> values the plot draws and must return one predicted <code>y</code> for each of them. May be given positionally or as <code>fun =</code> .
linewidth	The width of the residual lines. Default is 0.2. Must be named.
xlab, ylab, title, subtitle, caption	Labels for the plot.
geom, stat, position	Not set by the caller. A residual is drawn by its own <code>geom</code> and <code>stat</code> , and moved by the position the observations are already drawn with, so that a segment stays on the point it belongs to.
show.legend	Whether this layer contributes to the legend.
show.help	Print the layer's own help instead of drawing.
inherit	Whether the layer inherits the plot's aesthetics. The axes and the prediction are stated outright; everything else – a mapped color, for instance – is inherited from the plot.
environment	The environment mappings are resolved in.

Details

Draws residual lines from predicted values to the observed points, using a user-supplied function of `x` (e.g., the function plotted with `gf_function()`). Where `gf_resid()` measures a fitted model,

this measures a function you wrote: it is called on the x values the plot draws, and each residual runs from the prediction to the observation.

Value

A ggplot object with residual lines added.

Examples

```
set.seed(1)
df <- data.frame(X = 1:10, Y = 2 + 3 * (1:10) + rnorm(10))
my_fun <- function(x) 2 + 3 * x

gf_point(Y ~ X, data = df) %>%
  gf_function(my_fun) %>%
  gf_resid_fun(my_fun, color = "red", alpha = 0.5)
```

gf_sd_ruler

Add a Standard Deviation Ruler to a Plot

Description

[Experimental]

Usage

```
gf_sd_ruler(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  where = "middle",
  na.rm = FALSE,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = "segment",
  stat = coursekata::StatSdRuler,
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = TRUE,
  environment = parent.frame()
)
```

Arguments

object	The plot or data to add the ruler to; typically a plot piped in from <code>gf_point()</code> , <code>gf_jitter()</code> , or <code>gf_histogram()</code> .
ggformula	A formula naming the outcome and, optionally, the x variable: <code>y ~ x</code> . Defaults to the plot's own mapping when the plot already names one.
data	Dataset. Defaults to the plot's data.
...	Additional arguments accepted by <code>ggplot2::geom_segment()</code> , including fixed color and linewidth values. Defaults come from the segment geom, as they do for <code>stat_sd_ruler()</code> .
where	For a vertical ruler, where on the x-axis to place it: "middle" (midpoint of x range), "mean", or "median". Ignored for a horizontal ruler, which always starts at the mean.
na.rm	Should missing values be silently removed?
xlab, ylab, title, subtitle, caption	Axis and plot labels; see <code>ggformula::gf_point()</code> .
geom, stat, position	Layer components; see <code>ggformula::gf_point()</code> .
show.legend	Should this layer be included in the legends?
show.help	If TRUE, display some minimal help.
inherit	A logical indicating whether default attributes are inherited from a parent plot.
environment	An environment in which to evaluate the formula.

Details

Adds a segment showing one standard deviation of the outcome, anchored at its mean. The orientation depends on where the outcome variable lives: on a scatter or jitter plot (outcome on the y-axis) the ruler is a vertical segment placed at a chosen x position; on a histogram (outcome on the x-axis, no y aesthetic) it is a horizontal segment running from the mean to mean + SD along the baseline. The orientation is detected automatically from the plot's axis mappings.

Both the outcome and, where relevant, the placement are measured in the space the panel is drawn in: a faceted plot measures each panel's own subset, and a transformed axis or a computed mapping such as `~log(Thumb)` is measured in the transformed or computed values, not the raw column.

`gf_sd_ruler()` draws one ruler per panel, so an aesthetic mapped on the call – `gf_sd_ruler(color = ~Sex)` – is refused; split the plot instead with `y ~ x | group` to get one ruler per group.

Value

A ggplot object with the SD ruler segment added.

See Also

`stat_sd_ruler()` provides the ggplot2 interface. The model visualization guide shows the ruler alongside residuals and compares groups with different spread: <https://coursekata.github.io/coursekata-r/articles/model-visualization.html>

Examples

```
# the ruler runs from the mean (the empty model) up by one standard
# deviation -- it looks like a residual because SD is a typical residual
gf_point(Thumb ~ Height, data = Fingers, alpha = .4) %>%
  gf_model(lm(Thumb ~ NULL, data = Fingers)) %>%
  gf_sd_ruler()

# `where` controls placement along the x-axis
gf_point(Thumb ~ Height, data = Fingers, alpha = .4) %>%
  gf_sd_ruler(where = "mean")

# categorical x works the same way
gf_jitter(Thumb ~ Sex, data = Fingers, width = .1, alpha = .4) %>%
  gf_sd_ruler(where = "median")

# on a histogram the outcome is on the x-axis, so the ruler is horizontal
# and runs along the baseline from the mean to one SD above it
gf_histogram(~Thumb, data = Fingers, binwidth = 5) %>%
  gf_sd_ruler(linewidth = 2)

# name the variable explicitly when the plot does not make it obvious
gf_point(Thumb ~ Height, data = Fingers, alpha = .4) %>%
  gf_sd_ruler(Thumb ~ Height)

# one ruler per panel
gf_sd_ruler(Thumb ~ Height | Sex, data = Fingers)
```

gf_squareplot

Countable-Rectangle Histogram

Description**[Experimental]****Usage**

```
gf_squareplot(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  binwidth = NULL,
  bins = NULL,
  center = NULL,
  boundary = NULL,
  closed = NULL,
  breaks = NULL,
  bars = "none",
```

```

na.rm = TRUE,
xlab,
ylab,
title,
subtitle,
caption,
geom = coursekata::GeomSquareplot,
stat = coursekata::StatSquareplot,
position = "identity",
show.legend = NA,
show.help = NULL,
inherit = TRUE,
environment = parent.frame()
)

```

Arguments

object	A ggplot object, a data frame, or a formula. When a plot, the squares are added to it.
gformula	A formula with shape $\sim x$, optionally faceted as $\sim x \mid z$.
data	A data frame holding the variable in gformula.
...	Aesthetics such as fill or alpha, either set to a value or mapped with a one-sided formula (fill = $\sim group$). color sets the color of the separators between squares and may be mapped; bar_color sets the bar's own color, and bar_linewidth its width. Also takes ggplot2::stat_bin()'s pad, which adds an empty bin on either end of the range.
binwidth	Width of the bins, for a continuous x. Chosen from the data when unset: 1 for whole-number data spanning 50 or less, so every value gets a column of its own, and a thirtieth of the range otherwise. Has no effect on a discrete x, which is counted instead.
bins	How many bins to divide the range into, used when binwidth is unset.
center, boundary	The center of one bin, or an edge of one. Either places the whole grid; give one or the other, not both.
closed	Which end of a bin holds a value that lands exactly on it, "right" or "left". For whole-number data, boundary = 0.5 puts every value in the column it is labeled with, whichever end is closed.
breaks	The bin edges themselves, which need not be evenly spaced.
bars	Display style: "none" (squares only), "outline" (squares inside the bar they add up to) or "solid" (that bar alone).
na.rm	Must be TRUE. A missing value has no square to draw.
xlab, ylab, title, subtitle, caption	Labels.
geom, stat, position	The layer's geom, stat and position.

<code>show.legend</code>	Whether to show a legend, or NA to decide per aesthetic.
<code>show.help</code>	Print the function's own help instead of drawing.
<code>inherit</code>	Whether to inherit the plot's aesthetics.
<code>environment</code>	Where to evaluate the formula.

Details

Creates histograms where each observation is drawn as its own square, stacked into columns, so a bin's height can be counted as well as read off the axis: $n = 47$ is 47 squares. Designed for teaching statistical concepts like sampling distributions and hypothesis testing.

Sensible defaults are chosen based on the data:

- For integer-valued data with a small range, the `binwidth` defaults to 1 so that each integer gets its own column. Everything else about the bins is `stat_bin()`'s: `bins`, `center`, `boundary`, `closed`, `breaks` and `pad` put a squareplot's columns exactly where a `gf_histogram()`'s bars would be.
- A factor keeps its levels, so a level nobody landed in still holds its place on the axis. Knowing a value never occurred is the point.
- A discrete `x` – a factor, character or logical vector – is counted, one column per level, positioned the way `gf_bar()` positions its bars. The binning arguments (`bins`, `binwidth`, `center`, `boundary`, `closed`, `breaks`, `pad`) belong to a continuous `x`; supplying one alongside a discrete `x` warns rather than changing the plot.
- The white separator between two squares is capped at a quarter of a square's smaller side, so as a bin fills and its squares shrink the separator thins with them and the squares stay countable.
- The `y` axis is a count, so its breaks are whole numbers.
- Counts must be finite, nonnegative whole numbers. Integer frequency weights are supported; fractional weighted counts need a histogram or bar plot instead.

Fill and separator colour inherit the plot's mappings. With no mapping or fixed value, squares use teal fill and white separators. Set `fill` or `color` explicitly to override an inherited mapping. Plot-level `after_scale()` linewidth mappings use automatic border fitting. Supply the mapping on the squareplot layer itself to keep its width fixed.

The bins are a histogram's bins: `binwidth`, `bins`, `center`, `boundary`, `closed` and `breaks` mean what they mean on [ggformula:gf_histogram\(\)](#), and the same arguments give the same bin edges and the same counts, so squares laid over bars land inside them.

Everything that is not the squares is a layer or a scale: `%>% show_mean()`, `%>% show_dgp()`, `%>% gf_lims(x = )`, `%>% gf_refine(ggplot2::expand_limits(y = ))`. Each of these was an argument here once, and passing the old name is refused with the replacement named, so a call written against the old signature says what to write rather than drawing a plot with the mark missing.

Value

A ggplot object.

See Also

`show_mean()` and `show_dgp()` annotate a distribution. The sampling distributions guide shows this plot in the context of a full shuffle-and-estimate workflow: <https://coursekata.github.io/coursekata-r/articles/sampling-distributions.html>

Examples

```
# each observation is a countable square
gf_squareplot(~Thumb, data = Fingers)

# `bars` controls the display: "none" (default), "outline", or "solid"
gf_squareplot(~Thumb, data = Fingers, bars = "outline")

# the bins are a histogram's bins, so squares laid over bars land inside them --
# name the grid on both layers, because a layer never reads its neighbor's
gf_histogram(~Thumb, data = Fingers, bins = 8) %>% gf_squareplot(bins = 8)

# customize fill color, binwidth, and axis limits
gf_squareplot(~Thumb, data = Fingers, fill = "coral", binwidth = 5) %>%
  gf_lims(x = c(30, 90))

# integer data with a small range gets one column per integer
int_data <- data.frame(rolls = sample(1:6, 30, replace = TRUE))
gf_squareplot(~rolls, data = int_data)

# the plot is a real ggformula layer, so it facets and takes mapped aesthetics
gf_squareplot(~ Thumb | Sex, data = Fingers)
gf_squareplot(~Thumb, data = Fingers, fill = ~Sex)

# with 2000 observations the squares shrink, and their separators thin to fit
set.seed(24)
large_data <- data.frame(x = rnorm(2000, mean = 50, sd = 10))
gf_squareplot(~x, data = large_data)

# show a dashed line at the sample mean
gf_squareplot(~Thumb, data = Fingers) %>% show_mean()

# frame a sampling distribution with its data generating process: with only
# 10 shuffles, the mean of the distribution can land far from the null.
# Position-scale limits and show_dgp() compose in either order; neither
# changes the count range.
shuffled_b1 <- function(n) {
  data.frame(b1 = replicate(n, {
    shuffled_tip <- base::sample(TipExperiment$Tip)
    b1(lm(shuffled_tip ~ Condition, data = TipExperiment))
  }))
}

set.seed(42)
gf_squareplot(~b1, data = shuffled_b1(10), binwidth = 2) %>%
  gf_lims(x = c(-30, 30)) %>%
  gf_refine(ggplot2::expand_limits(y = 10)) %>%
```

```

show_mean() %>%
show_dgp()

# a factor keeps every level, including the ones nothing landed in
ratings <- data.frame(rating = factor(
  base::sample(1:5, 20, replace = TRUE, prob = c(1, 2, 4, 2, 1)),
  levels = 1:5
))
gf_squareplot(~rating, data = ratings)

```

gf_square_reduce

Add Squared Reduction Visualization to a Plot

Description

[Experimental]

gf_squareduce() is a fully supported alias of gf_square_reduce(), named the way the classroom that asked for it says it.

Usage

```

gf_square_reduce(
  object = NULL,
  gformula = NULL,
  data = NULL,
  ...,
  model,
  aspect = 4/6,
  alpha = 0.1,
  xlab,
  ylab,
  title,
  subtitle,
  caption,
  geom = coursekata::GeomSquareResid,
  stat = coursekata::StatReduce,
  position = "identity",
  show.legend = NA,
  show.help = NULL,
  inherit = FALSE,
  environment = parent.frame()
)

gf_squareduce(
  object = NULL,
  gformula = NULL,
  data = NULL,

```

```

...,
model,
aspect = 4/6,
alpha = 0.1,
xlab,
ylab,
title,
subtitle,
caption,
geom = coursekata::GeomSquareResid,
stat = coursekata::StatReduce,
position = "identity",
show.legend = NA,
show.help = NULL,
inherit = FALSE,
environment = parent.frame()
)

```

Arguments

object	A ggformula plot object, typically created with <code>gf_point()</code> .
gformula	Not used. <code>gf_square_reduce()</code> measures a model, not an aesthetic formula; a model given positionally lands here and is moved to <code>model</code> .
data	Not used. The reductions are measured over the data the plot was built from. Anything supplied here is left for ggformula and ggplot2 to answer, exactly as it is for any other <code>gf_</code> layer.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , such as <code>color</code> or <code>fill</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~ expression</code> , or (c) attributes of the layer as a whole.
model	A model already fit by <code>lm()</code> or <code>aov()</code> . The plot supplies the observations' position on the other axis; the model supplies what it predicted for each of them. May be given positionally or as <code>model =</code> . A fit without an intercept, or one fit with weights or an offset, is refused. For an unweighted least-squares fit with an intercept, the sum of squared total deviations equals the sum of squared residuals plus the sum of squared reductions. That identity need not hold without an intercept. Weighted least squares instead guarantees a weighted identity based on a weighted mean, which the plot's plain areas do not represent. <code>gf_resid()</code> can measure either fit because it does not rely on the decomposition.
aspect	The square's aspect ratio. Default is 4/6. Must be named.
alpha	The transparency of the square's fill. Default is 0.1. Must be named.
xlab, ylab, title, subtitle, caption	Labels for the plot.
geom, stat, position	Not set by the caller. A squared reduction is drawn by its own geom and stat, with a jitter that holds the outcome axis still so its squares start at the grand mean without floating off it, while jittering the other axis exactly the points layer's own jitter did.

<code>show.legend</code>	Whether this layer contributes to the legend.
<code>show.help</code>	Print the layer's own help instead of drawing.
<code>inherit</code>	Whether the layer inherits the plot's aesthetics. FALSE, where <code>gf_reduce()</code> is TRUE – see <code>gf_square_resid()</code> for why: a square is a filled region drawn in the geom's own colors, and inheriting a plot's mapped color would outline every square in the color of the group it measures instead of leaving one neutral area per observation. Set it to TRUE to take the outline anyway.
<code>environment</code>	The environment mappings are resolved in.

Details

Draws squared reduction polygons between the grand mean and the values a fitted model predicts. Each polygon shows one observation's squared reduction; together, their areas represent the model sum of squares. The square is built on the reduction itself and turns with it: a model of the variable the plot puts on x squares the horizontal distance. Its side is scaled to stay square on the page rather than in data units.

Use the same aspect for all three square layers. Across observations, the reduction areas and residual areas sum to the total areas. The equality is between those sums, not between the three squares for any one observation. `gf_square_resid()`, `gf_square_reduce()`, and any squared total drawn beside them must use the same aspect for their areas to share a scale.

Value

A ggplot object with squared reduction polygons added.

Examples

```
set.seed(1)
penguins_20 <- sample(penguins, 20)

# two collections of squares in one sample-level decomposition: squared
# residuals (firebrick) and squared reductions (blue), drawn at one aspect
flipper_model <- lm(body_mass_kg ~ flipper_length_m, data = penguins_20)
gf_point(body_mass_kg ~ flipper_length_m, data = penguins_20) %>%
  gf_model(flipper_model) %>%
  gf_square_resid(flipper_model, color = "firebrick") %>%
  gf_square_reduce(flipper_model, color = "blue")

# and for a two-group model on a jitter plot
gentoo_model <- lm(body_mass_kg ~ gentoo, data = penguins_20)
gf_jitter(body_mass_kg ~ gentoo, data = penguins_20, width = .1) %>%
  gf_model(gentoo_model) %>%
  gf_square_reduce(gentoo_model, color = "blue")
```

gf_square_resid*Add Squared Residual Visualization to a Plot*

Description

[Experimental]

gf_squaresid() is a fully supported alias of gf_square_resid(). The name honors [Tyler Haslam](#), the Utah high school teacher whose efforts shaped the residual and squared-residual visualizations and who requested this function by that name.

Usage

```
gf_square_resid(  
  object = NULL,  
  gformula = NULL,  
  data = NULL,  
  ...,  
  model,  
  aspect = 4/6,  
  alpha = 0.1,  
  xlab,  
  ylab,  
  title,  
  subtitle,  
  caption,  
  geom = coursekata::GeomSquareResid,  
  stat = coursekata::StatResid,  
  position = "identity",  
  show.legend = NA,  
  show.help = NULL,  
  inherit = FALSE,  
  environment = parent.frame()  
)
```

```
gf_squaresid(  
  object = NULL,  
  gformula = NULL,  
  data = NULL,  
  ...,  
  model,  
  aspect = 4/6,  
  alpha = 0.1,  
  xlab,  
  ylab,  
  title,  
  subtitle,
```

```

caption,
geom = coursekata::GeomSquareResid,
stat = coursekata::StatResid,
position = "identity",
show.legend = NA,
show.help = NULL,
inherit = FALSE,
environment = parent.frame()
)

```

Arguments

object	A ggformula plot object, typically created with <code>gf_point()</code> .
gformula	Not used. <code>gf_square_resid()</code> measures a model, not an aesthetic formula; a model given positionally lands here and is moved to <code>model</code> .
data	Not used. The residuals are measured over the data the plot was built from. Anything supplied here is left for ggformula and ggplot2 to answer, exactly as it is for any other <code>gf_</code> layer.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , such as <code>color</code> or <code>fill</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~ expression</code> , or (c) attributes of the layer as a whole.
model	A model already fit by <code>lm()</code> or <code>aov()</code> . The plot supplies the observations; the model supplies what it predicted for each of them. May be given positionally or as <code>model =</code> .
aspect	The square's aspect ratio. Default is 4/6. Must be named.
alpha	The transparency of the square's fill. Default is 0.1. Must be named.
xlab, ylab, title, subtitle, caption	Labels for the plot.
geom, stat, position	Not set by the caller. A squared residual is drawn by its own geom and stat, and moved by the position the observations are already drawn with, so that a square stays on the point it belongs to.
show.legend	Whether this layer contributes to the legend.
show.help	Print the layer's own help instead of drawing.
inherit	Whether the layer inherits the plot's aesthetics. FALSE, where <code>gf_resid()</code> is TRUE: a square is a filled region drawn in the geom's own colors, so inheriting a plot's mapped color outlines every square in the color of the group it measures instead of leaving one neutral area per observation. The axes and the prediction are stated outright, so nothing the square needs is lost by not inheriting. Set it to TRUE to take the outline anyway.
environment	The environment mappings are resolved in.

Details

Draws squared residual polygons between observed points and the values a fitted model predicts for them, so squared error is an area you can see. The square is built on the residual itself and turns

with it: a model of the variable the plot puts on x squares the horizontal distance. Its side is scaled to stay square on the page rather than in data units.

Value

A ggplot object with squared residual polygons added.

Examples

```
# squared residuals can be drawn on a full data set, but with hundreds of
# points the plot gets hard to read
flipper_model <- lm(body_mass_kg ~ flipper_length_m, data = penguins)
gf_point(body_mass_kg ~ flipper_length_m, data = penguins) %>%
  gf_model(flipper_model) %>%
  gf_square_resid(flipper_model)

# a small sample makes the squared residuals much easier to see
set.seed(1)
penguins_20 <- sample(penguins, 20)

# squared residuals from the empty model (in blue)
empty_model <- lm(body_mass_kg ~ NULL, data = penguins_20)
gf_point(body_mass_kg ~ flipper_length_m, data = penguins_20) %>%
  gf_model(empty_model) %>%
  gf_square_resid(empty_model, color = "blue")

# squared residuals from a two-group model on a jitter plot (in firebrick)
gentoo_model <- lm(body_mass_kg ~ gentoo, data = penguins_20)
gf_jitter(body_mass_kg ~ gentoo, data = penguins_20, width = .1) %>%
  gf_model(gentoo_model) %>%
  gf_square_resid(gentoo_model, color = "firebrick")

# squared residuals from a regression model (in firebrick)
sample_flipper_model <- lm(body_mass_kg ~ flipper_length_m, data = penguins_20)
gf_point(body_mass_kg ~ flipper_length_m, data = penguins_20) %>%
  gf_model(sample_flipper_model) %>%
  gf_square_resid(sample_flipper_model, color = "firebrick")
```

gf_square_resid_fun *Add Squared Residual Visualization from a Function to a Plot*

Description

[Experimental]

Usage

```
gf_square_resid_fun(
  object = NULL,
  gformula = NULL,
```

```

data = NULL,
...,
fun,
aspect = 4/6,
alpha = 0.1,
xlab,
ylab,
title,
subtitle,
caption,
geom = coursekata::GeomSquareResid,
stat = coursekata::StatResid,
position = "identity",
show.legend = NA,
show.help = NULL,
inherit = FALSE,
environment = parent.frame()
)

```

Arguments

object	A ggformula plot object, typically created with <code>gf_point()</code> .
gformula	Not used. <code>gf_square_resid_fun()</code> measures a function, not an aesthetic formula; a function given positionally lands here and is moved to <code>fun</code> .
data	Not used. The residuals are measured over the data the plot was built from. Anything supplied here is left for ggformula and ggplot2 to answer, exactly as it is for any other <code>gf_</code> layer.
...	Additional arguments. Typically these are (a) ggplot2 aesthetics to be set with <code>attribute = value</code> , such as <code>color</code> or <code>fill</code> , (b) ggplot2 aesthetics to be mapped with <code>attribute = ~ expression</code> , or (c) attributes of the layer as a whole.
fun	A function of one argument. It is called on the x values the plot draws and must return one predicted y for each of them. May be given positionally or as <code>fun =</code> .
aspect	The square's aspect ratio. Default is 4/6. Must be named.
alpha	The transparency of the square's fill. Default is 0.1. Must be named.
xlab, ylab, title, subtitle, caption	Labels for the plot.
geom, stat, position	Not set by the caller. A squared residual is drawn by its own geom and stat, and moved by the position the observations are already drawn with, so that a square stays on the point it belongs to.
show.legend	Whether this layer contributes to the legend.
show.help	Print the layer's own help instead of drawing.
inherit	Whether the layer inherits the plot's aesthetics. FALSE, where <code>gf_resid_fun()</code> is TRUE: a square is a filled region drawn in the geom's own colors, so inheriting a plot's mapped color outlines every square in the color of the group it measures instead of leaving one neutral area per observation. The axes and the prediction

are stated outright, so nothing the square needs is lost by not inheriting. Set it to TRUE to take the outline anyway.

environment The environment mappings are resolved in.

Details

Draws squared residual polygons between observed points and the values predicted by a user-supplied function of x . Where `gf_square_resid()` measures a fitted model, this measures a function you wrote: it is called on the x values the plot draws, and each square is built on the residual from an observation to what the function predicts for it.

Value

A ggplot object with squared residual polygons added.

Examples

```
set.seed(1)
df <- data.frame(X = 1:10, Y = 2 + 3 * (1:10) + rnorm(10))
my_fun <- function(x) 2 + 3 * x

gf_point(Y ~ X, data = df) %>%
  gf_function(my_fun) %>%
  gf_square_resid_fun(my_fun, color = "red", alpha = 0.3)
```

guide_cutoff

Draw cutoff anchors as a measured position guide

Description

`guide_cutoff()` draws one truthful marker rail and, when requested, one measured label lane per helper call. Two-sided labels dodge along that lane while their leaders remain tied to the truthful cutoff anchors. The guide receives already-computed cutoff values; it never inspects the distribution or recomputes a statistic.

Usage

```
guide_cutoff(
  value,
  label = NULL,
  side = NULL,
  call_id = 1L,
  colour = ggplot2::waiver(),
  shape = 24,
  size = 4,
  linewidth = 0.5,
  title = NULL,
  theme = NULL,
```

```
    order = 0,  
    position = ggplot2::waiver()  
  )  
  
  GuideCutoff
```

Arguments

value	One or two finite numeric cutoff values in the position scale’s data space.
label	Optional labels parallel to value.
side	Optional side names parallel to value, each either "lower" or "upper". Supply label and side together.
call_id	One positive integer identifying this guide instance.
colour	Colour of the markers, leaders, and labels.
shape	Marker shape. The default triangle points toward the panel.
size	Marker size.
linewidth	Width of marker outlines and leaders.
title	Optional guide title.
theme	A theme for this guide, or NULL.
order	Guide order.
position	Guide position. The default is derived from the scale.

Format

GuideCutoff is a [ggplot2::Guide](#) object.

Value

A ggplot2 guide.

guide_dgp	<i>Draw a data-generating-process frame as a position guide</i>
-----------	---

Description

guide_dgp() places a truthful null anchor in measured guide space. The population role supplies the population heading, equation, axis line, null label, and triangle. The estimate role supplies the estimate heading, equation, and null label. It never computes a mean or trains a scale.

Usage

```
guide_dgp(
  value = 0,
  role = c("population", "estimate"),
  label = ggplot2::waiver(),
  equation = ggplot2::waiver(),
  colour = ggplot2::waiver(),
  shape = ggplot2::waiver(),
  size = ggplot2::waiver(),
  linewidth = ggplot2::waiver(),
  title = ggplot2::waiver(),
  theme = NULL,
  order = 0,
  position = ggplot2::waiver()
)
```

GuideDgp

Arguments

value	One finite numeric null value in the position scale's data space.
role	Either "population" or "estimate".
label	The null label. A role-specific plotmath label is used by default.
equation	The model equation. A role-specific plotmath expression is used by default.
colour	Colour of the keyed null content.
shape	Point shape for the population marker. The default points toward the panel.
size	Size of the population marker.
linewidth	Width of the population axis line.
title	The role heading. A role-specific heading is used by default.
theme	A theme for this guide, or NULL.
order	Guide order.
position	Guide position. The default is derived from the scale.

Format

GuideDgp is a [ggplot2::Guide](#) object.

Value

A ggplot2 guide.

middle

Find a percentage of a distribution

Description

Given a distribution, find which values lie in the upper, lower, or middle proportion of the distribution. Useful when you want to do something like shade in the middle 95% of a plot. This is a greedy operation, meaning that if the cutoff point is between two whole numbers the specified region will suck up the extra space. For example, the requesting the upper 30% of the [1 2 3 4] will return [FALSE FALSE TRUE TRUE] because the 30% was greedy.

[Experimental]

`outer()` marks values in both outer tails of a distribution. It is the complement of `middle()`: `outer(x, prop)` is equivalent to `tails(x, 1 - prop)`.

Usage

```
middle(x, prop = 0.95, greedy = TRUE)
```

```
outer(x, prop)
```

```
tails(x, prop = 0.95, greedy = TRUE)
```

```
lower(x, prop = 0.025, greedy = TRUE)
```

```
upper(x, prop = 0.025, greedy = TRUE)
```

Arguments

<code>x</code>	The distribution of values to check.
<code>prop</code>	The total proportion in both tails combined, must be in (0, 1).
<code>greedy</code>	Whether the function should be greedy, as per the description above.

Details

Note that NA values are ignored when sizing the region, and come back as NA.

Value

A logical vector indicating which values are in the specified region.

See Also

The sampling distributions guide walks through building these distributions with `do()` and `shuffle()`, and shows the bootstrap variant: <https://coursekata.github.io/coursekata-r/articles/sampling-distributions.html>

Examples

```
# each function returns a logical vector marking the values in its region
upper(1:10, .1)
lower(1:10, .2)
middle(1:10, .5)
tails(1:10, .5)

# they are most often used as the fill aesthetic of a histogram of a
# sampling distribution -- here, b1s estimated from shuffled (null) data
set.seed(42)
shuffled <- data.frame(b1 = replicate(200, {
  shuffled_tip <- base::sample(TipExperiment$Tip)
  b1(lm(shuffled_tip ~ Condition, data = TipExperiment))
}))

# color the middle 95%: the b1 values we would expect to see often
# if the empty model were true
gf_histogram(~b1, data = shuffled, binwidth = 1, fill = ~ middle(b1, .95))

# tails() marks the same cutoffs with the opposite coloring: the values
# outside the middle 95% are the 5% most extreme
gf_histogram(~b1, data = shuffled, binwidth = 1, fill = ~ tails(b1, .95))

# outer() marks the same region as tails() but takes the tail proportion
# directly: the outer 5%
gf_histogram(~b1, data = shuffled, binwidth = 1, fill = ~ outer(b1, .05))

# upper() and lower() are for directional hypotheses: all 5% goes in one tail
gf_histogram(~b1, data = shuffled, binwidth = 1, fill = ~ upper(b1, .05))
gf_histogram(~b1, data = shuffled, binwidth = 1, fill = ~ lower(b1, .05))
```

penguins

A modified form of the `palmerpenguins::penguins` data set.

Description

The modifications are to select only a subset of the variables, and convert some of the units.

Usage

```
penguins
```

Format

A data frame with 333 observations on the following 7 variables:

`species` The species of penguin, coded as "Adelie", "Chinstrap", or "Gentoo".

`gentoo` Whether the penguin is a Gentoo penguin (1) or not (0).

`body_mass_kg` The mass of the penguin's body, in kilograms.

flipper_length_m The length of the penguin's flipper, in m.
bill_length_cm The length of the penguin's bill, in cm.
female Whether the penguin is female (1) or not (0).
island The island where the penguin was observed, coded as "Biscoe", "Dream", or "Torgersen".

scale_discrete_coursekata

A discrete color scale constructor with colorblind-safe palettes.

Description

See [coursekata_palette\(\)](#) for more information.

Usage

```
scale_discrete_coursekata(...)
```

Arguments

... Additional parameters passed on to the scale type.

Value

A discrete color scale.

See Also

[coursekata_palette](#)

Examples

```
gf_point(Thumb ~ Height, data = Fingers, color = ~RaceEthnic) +  
  scale_discrete_coursekata()
```

show_cutoffs

*Add Cutoff Markers to a Distribution***Description****[Experimental]****Usage**

```
show_cutoffs(
  object = NULL,
  part,
  color = "#1e3a8a",
  size = 4,
  show_labels = FALSE
)
```

Arguments

object	A ggplot of one distribution – a histogram, bar chart, density, dotplot, or gf_squareplot() .
part	A distribution part, e.g. <code>middle(Thumb, .95)</code> . Optional: without it, the part is read off the plot's fill aesthetic.
color	Marker/line color. Default <code>"#1e3a8a"</code> .
size	Marker size. Default 4.
show_labels	Whether to annotate the cutoffs. Default FALSE.

Details

Adds empirical quantile cutoffs for a distribution part – `middle()`, `tails()`, `upper()`, `lower()`, or `outer()`. Short dashed stems end in downward triangles that touch the numeric axis. Optional callouts are drawn in the data panel. Repeated labelled calls are measured and placed together. Their boxes stay separate when the panel has room, and their routed leaders remain tied to the exact cutoffs.

In very small panels, leaders use a simple elbow connection if there is no room for an outward route. Labels keep their text and size, so they may overlap or be clipped. Enlarge the figure, use fewer facet columns, or set `show_labels = FALSE` when labels do not fit.

By default the part is read off the plot's fill aesthetic, e.g. `fill = ~middle(Thumb, .95)`. Passing `part` overrides that reading: it marks whatever part is named there instead, and the fill (if any) is ignored – marking the 99% cutoffs on a plot shaded for the 95% is a deliberate, lossless override, not a mismatch.

`show_cutoffs()` refuses a plot whose first layer does not draw a distribution (a scatterplot, for instance) and, when `part` is given explicitly, a `part` that names a variable other than the one the plot puts on x.

Value

A ggplot object with cutoff markers and optional labels.

See Also

`stat_cutoff()` computes the same rule per panel; `geom_cutoff()` is the lower-level drawing component. `guide_cutoff()` can place cutoff anchors in a position guide instead.

Examples

```
gf_histogram(~Thumb, data = Fingers, binwidth = 5, fill = ~middle(Thumb, .95)) %>%
  show_cutoffs(show_labels = TRUE)

# an explicit part overrides the fill instead of requiring it to match
gf_histogram(~Thumb, data = Fingers, binwidth = 5, fill = ~middle(Thumb, .95)) %>%
  show_cutoffs(middle(Thumb, .99))
```

show_dgp

*Frame a Sampling Distribution With Its Data Generating Process***Description**

[Experimental]

Usage

```
show_dgp(object = NULL, color = "#003d70", null_color = "#E60000", size = 4)
```

Arguments

<code>object</code>	A plot of one distribution of estimates.
<code>color</code>	Color of the axes, equations and titles. Default "#003d70".
<code>null_color</code>	Color of the null hypothesis marker. Default "#E60000".
<code>size</code>	Size of the null hypothesis marker. Default 4.

Details

Frames a distribution of estimates with the process that generated them: the population model on a top axis labeled "Population Parameter (DGP)", the sample estimate below the plot, and a marker at the null hypothesis ($\beta_1 = 0$) on both – drawn only when zero is on the axis.

The population and estimate frames are position guides, outside the data panel. They therefore do not change the count range and remain compatible with fixed, zoomed, transformed, and free count axes. The ordinary numeric x guide stays in place as the estimate scale. Zero is omitted when it is not a finite visible value on that scale; it is never moved to a boundary. An x position scale can be added before or after `show_dgp()`; the final scale supplies the limits and breaks for both guides. A later `ggplot2::guides()` call can replace either guide in the usual ggplot2 way.

The plot's x label is retained when a later guide replaces the estimate frame. Existing plot theme settings, including inherited `axis.text`, `axis.title`, and `axis.line` settings, take precedence over the frame's default axis styling. Later themes follow `ggplot2`'s usual inheritance: change `axis.text.x`, `axis.title.x`, or `axis.line.x/axis.line.y` to override the frame's axis-specific settings.

Value

The plot, with population and estimate guides added.

See Also

The sampling distributions guide draws this figure inside a full shuffle-and-estimate workflow: <https://coursekata.github.io/coursekata-r/articles/sampling-distributions.html>

Examples

```
# with only ten shuffles the mean of the distribution can land well away
# from the null hypothesis marked on the top axis
set.seed(42)
shuffled <- data.frame(b1 = replicate(10, {
  b1(lm(base::sample(TipExperiment$Tip) ~ Condition, data = TipExperiment))
}))

# expand_limits() sets the count axis so two runs can be compared side by
# side; show_dgp() does not alter that range
gf_histogram(~b1, data = shuffled, binwidth = 2) %>%
  gf_refine(ggplot2::expand_limits(y = 10)) %>%
  show_mean() %>%
  show_dgp()
```

show_mean

Mark a Distribution's Mean

Description

[Experimental]

Usage

```
show_mean(
  object = NULL,
  color = "#E60000",
  linetype = "longdash",
  linewidth = 0.7
)
```

Arguments

object	A plot of one distribution.
color	Line color. Default "#E60000".
linetype	Line type. Default "longdash".
linewidth	Line width. Default 0.7.

Details

Draws a vertical line at the mean of the variable a distribution is built from, spanning the count axis. Works on any plot of one distribution – `gf_histogram()`, `gf_dotplot()`, `gf_squareplot()`. A faceted plot gets one line per panel, at that panel’s own mean, because a facet is a region with its own subset of the data. Each line spans the count axis of the panel it is drawn in, so `scales = "free_y"` is supported and no panel is stretched to hold another panel’s line.

A plot with a variable on each axis is refused. The middle of a two-variable plot is a model, and `gf_model()` draws models – an hline at the mean of the outcome for the empty model. Averaging the x variable of a scatterplot would draw a line nobody asked for.

Value

The plot, with a tagged mean line added.

See Also

`show_dgp()` frames a sampling distribution with the process that generated it; `gf_model()` draws the mean of a two-variable plot’s outcome.

Examples

```
gf_histogram(~Thumb, data = Fingers, binwidth = 5) %>% show_mean()

# a facet is a region with its own subset, so each panel gets its own mean
gf_histogram(~Thumb | Sex, data = Fingers, binwidth = 5) %>% show_mean()
```

Smallville	<i>Simulated housing data</i>
------------	-------------------------------

Description

These data are simulated to be similar to the Ames housing data, but with far fewer variables and much smaller effect sizes.

Usage

Smallville

Format

A data frame with 32 observations on the following 4 variables:

PriceK Price the home sold for (in thousands of dollars)

Neighborhood The neighborhood the home is in (Eastside, Downtown)

HomeSizeK The size of the home (in thousands of square feet)

HasFireplace Whether the home has a fireplace (0 = no, 1 = yes)

split_data	<i>Split data into train and test sets.</i>
------------	---

Description

Split data into train and test sets.

Usage

```
split_data(data, prop = 0.7)
```

Arguments

data	A data frame.
prop	The proportion of rows to assign to the training set.

Value

A list with two data frames, train and test.

Examples

```
set.seed(123)
parts <- split_data(Fingers, prop = 0.8)
nrow(parts$train)
nrow(parts$test)
```

StatCutoff

Compute a distribution part's empirical cutoffs, per panel

Description

StatCutoff calls the same `cutoff_plan()` used by `show_cutoffs()`, but the two interfaces deliberately operate at different scopes. `ggplot2` supplies a stat with one panel's rows at a time, so this stat computes panel-local cutoffs. `show_cutoffs()` plans once from the whole resolved distribution and repeats those global marks across facets.

Usage

StatCutoff

```
stat_cutoff(
  mapping = NULL,
  data = NULL,
  geom = "cutoff",
  position = "identity",
  ...,
  part = "middle",
  prop = 0.95,
  greedy = TRUE,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

<code>mapping</code>	Set of aesthetic mappings created by <code>ggplot2::aes()</code> .
<code>data</code>	The data to be displayed in this layer.
<code>geom</code>	The geometric object used to display the data. Defaults to "cutoff".
<code>position</code>	A position adjustment. Defaults to "identity".
<code>...</code>	Other arguments passed to <code>ggplot2::layer()</code> .
<code>part</code>	The distribution part to mark: "middle", "tails", "upper", "lower", or "outer".
<code>prop</code>	The requested proportion. The endpoints 0 and 1 are valid for this lower-level stat.
<code>greedy</code>	Whether a fractional observation is included in the selected region.
<code>na.rm</code>	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
<code>show.legend</code>	Logical. Should this layer be included in the legends?
<code>inherit.aes</code>	If FALSE, override the default aesthetics rather than combining with them.

Format

StatCutoff is a [ggplot2::Stat](#) object.

Details

The stat accepts "middle", "tails", "upper", "lower", and "outer". It selects cutoffs in the data's original space: transformed position values are inverted before `cutoff_plan()` runs, and the selected cutoffs are transformed forward again for ggplot2. This preserves the requested distribution part on increasing, decreasing, and transformed scales.

`stat_cutoff()` is the conventional layer constructor. Its default geom is "cutoff"; use `geom = "vline"` when a full-height reference line is wanted instead. Since the stat computes once per panel, styling aesthetics from the source data cannot be mapped; set them to one value, or facet the plot to compute one set of cutoffs per group.

Value

`stat_cutoff()` returns a ggplot2 layer.

See Also

[show_cutoffs\(\)](#), which marks whole-distribution cutoffs.

StatDistMean

Compute a distribution's mean, per panel

Description

StatDistMean receives the rows that ggplot2 has assigned to each panel and emits their mean as an xintercept. Because ggplot2 transforms position scales and applies hard-limit out-of-bounds handling before a stat runs, the result follows that lifecycle: a transformed scale changes the values being averaged, hard scale limits can remove values, and coordinate zoom does not change the result.

Usage

StatDistMean

```
stat_dist_mean(
  mapping = NULL,
  data = NULL,
  geom = "vline",
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by <code>ggplot2::aes()</code> .
data	The data to be displayed in this layer.
geom	The geometric object used to display the data. Defaults to "vline".
position	A position adjustment. Defaults to "identity".
...	Other arguments passed to <code>ggplot2::layer()</code> .
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?
inherit.aes	If FALSE, override the default aesthetics rather than combining with them.

Format

StatDistMean is a `ggplot2::Stat` object.

Details

`stat_dist_mean()` is the conventional layer constructor. It computes from `x` and defaults to a vertical line; `show_mean()` is the teaching-oriented helper that resolves a distribution plot's mapping and styles this layer. Since the stat computes once per panel, styling aesthetics from the source data cannot be mapped; set them to one value, or facet the plot to compute one mean per group.

Value

`stat_dist_mean()` returns a `ggplot2` layer.

See Also

`show_mean()` and `gf_model()`.

StatModel	<i>Compute the model represented by a layer</i>
-----------	---

Description

`StatModel` passes through rows prepared from a supplied model. With no supplied model, it fits the model implied by the mapped positions: a linear model for a continuous predictor, group means for a categorical predictor, or an intercept when only the outcome is mapped.

Usage

`StatModel`

Format

`StatModel` is a `ggplot2::Stat` object.

StatReduce*Carry reduction endpoints alongside their observations*

Description

A reduction uses the same endpoint representation as a residual, but its starting point is the grand mean rather than the observation. The separate stat class keeps that statistical meaning visible to ggplot2 users and to code inspecting the layer. [GeomResid](#) and [GeomSquareResid](#) can draw either stat.

Usage

StatReduce

Format

A [ggplot2::Stat](#) object.

See Also

[stat_reduce\(\)](#), [geom_reduce\(\)](#), and [geom_square_reduce\(\)](#).

StatResid*Carry a prediction alongside the observation it belongs to*

Description

xend/yend is a positional aesthetic on purpose: that is what gets the prediction transformed by its scale and trained into the panel's range. Exactly one of the two arrives, on the axis the plot put the model's outcome on.

Usage

StatResid

Format

A [ggplot2::Stat](#) object.

Details

compute_layer checks the required aesthetics, then returns every row unchanged. An observation with an NA on it (a predictor the model dropped) survives the stat instead of being removed before the position runs. The residual's jitter is a function of the seed and the rows it is handed, exactly as the point layer's is, so losing a row here would hand it a different sequence of draws and the segment would land away from its point.

See Also

[stat_resid\(\)](#), [geom_resid\(\)](#), and [geom_square_resid\(\)](#).

StatSdRuler

Compute a panel's standard deviation ruler

Description

StatSdRuler reduces each panel to the segment for one standard deviation, anchored at the outcome's mean. With x and y aesthetics, the outcome is y; the ruler is vertical and where places it along x. With x alone, the outcome is x and the ruler runs along the baseline from the mean to one standard deviation above it.

Usage

StatSdRuler

```
stat_sd_ruler(
  mapping = NULL,
  data = NULL,
  geom = "segment",
  position = "identity",
  ...,
  where = "middle",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by ggplot2::aes() .
data	The data to be displayed in this layer.
geom	The geometric object used to display the data. Defaults to "segment".
position	A position adjustment. Defaults to "identity".
...	Other arguments passed to ggplot2::layer() .
where	With both x and y mapped, where to place the ruler along x: "middle", "mean", or "median". Ignored when only x is mapped.
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?
inherit.aes	If FALSE, override the default aesthetics rather than combining with them.

Format

A `ggplot2::Stat` object.

Details

`stat_sd_ruler()` is the conventional layer constructor. It uses the same stat as `gf_sd_ruler()`, so both interfaces measure transformed values and compute one ruler from each panel's rows. Groups within a panel do not get separate rulers. Styling aesthetics from the source data cannot be mapped; set them to one value, or facet the plot to draw one ruler per group.

Value

`stat_sd_ruler()` returns a `ggplot2` layer.

See Also

`gf_sd_ruler()` provides the `ggformula` interface.

Examples

```
ggplot2::ggplot(Fingers, ggplot2::aes(Height, Thumb)) +
  ggplot2::geom_point() +
  stat_sd_ruler(where = "mean", colour = "red", linewidth = 1)

ggplot2::ggplot(Fingers, ggplot2::aes(Thumb)) +
  ggplot2::geom_histogram(bins = 30) +
  stat_sd_ruler(colour = "red", linewidth = 1)
```

StatSquareplot

Bin observations the way `ggplot2::stat_bin()` does

Description

`StatSquareplot` is `ggplot2::StatBin`: `binwidth`, `bins`, `center`, `boundary`, `closed`, `breaks` and `pad` all mean exactly what they mean on a histogram, because they are the histogram's own code. `GeomSquareplot` re-expands each bin into one rectangle per observation. Pair the two in a `ggplot2::layer()` to put countable squares in a plot you are assembling yourself.

Usage

```
StatSquareplot
```

Format

A `ggplot2::Stat` object.

Details

The one difference from a plain histogram is the binwidth chosen when the call names no grid at all: integer-valued data over a small range gets one bin per integer, rather than `stat_bin`'s `bins = 30` default, because thirty slivers is the wrong advice for a plot whose point is countability.

See Also

[gf_squareplot\(\)](#), which pairs this stat and geom for you.

Survey	<i>Students at a university were asked to enter a random number between 1-20 into a survey.</i>
--------	---

Description

Students at a university taking an introductory statistics course were asked to complete this survey as part of their homework.

Usage

Survey

Format

A data frame with 211 observations on the following 1 variable:

Any1_20 The random number between 1 and 20 that a student thought of.

Tables	<i>Tables data</i>
--------	--------------------

Description

Data about tips collected from an experiment with 44 tables at a restaurant.

Usage

Tables

Format

A data frame with 44 observations on the following 2 variables.

TableID A number assigned to each table.

Tip How much the tip was.

theme_coursekata	<i>A simple theme built on top of <code>ggplot2::theme_bw</code></i>
------------------	--

Description

The coursekata package automatically loads this theme when the package is loaded. This is in addition to a number of other plot tweaks and option settings. To just restore the theme to the default, you can run `set_theme(theme_grey)`. If you want to restore all plot related settings and/or prevent them when loading the package, see [coursekata_unload_theme](#).

Usage

```
theme_coursekata()
```

Value

A gg theme object

Examples

```
gf_boxplot(Thumb ~ RaceEthnic, data = Fingers, fill = ~RaceEthnic)
```

TipExperiment	<i>Data from an experiment about smiley faces and tips</i>
---------------	--

Description

Tables were randomly assigned to receive checks that either included or did not include a drawing of a smiley face. Data was collected from 44 tables in an effort to examine whether the added smiley face would cause more generous tipping.

Usage

```
TipExperiment
```

Format

A data frame with 44 observations on the following 3 variables.

TableID A number assigned to each table.

Tip How much the tip was.

Condition Which experimental condition the table was randomly assigned to.

Check (Simulated) The amount of money the table paid for their meal.

FoodQuality (Simulated) The perceived quality of the food.

 tip_exp

Simulated data for an experiment about smiley faces and tips

Description

These are simulated data that are similar to the TipExperiment data. Hypothetical tables were randomly assigned to receive checks that either included or did not include a drawing of a smiley face, either from a male or a female server.

Usage

tip_exp

Format

A data frame with 44 observations on the following 3 variables.

gender Whether the server was female or male

condition Whether the check had a smiley face or not (control)

tip_percent The size of the tip as a percentage of the price of the meal

 World

Data on countries from the Happy Planet Index project.

Description

These data have been updated with some historical height data (from [Our World in Data](#)), drinking data (collected by the World Health Organization and published with the [FiveThirtyEight alcohol-consumption data](#)), population and land characteristics, and vaccination data (from March 2023).

Usage

World

Format

A data frame with 130 observations on the following 14 variables:

Country Name of country

Region One of 5 UN defined regions: Africa, Americas, Asia, Europe, Oceania

Code Three-letter country codes defined by the International Organization for Standardization ([ISO](#)) to represent countries in a way that avoids errors since a country's name changes depending on the language being used.

LifeExpectancy Average life expectancy (in years)

GirlsH1900 The average of 18-year-old girls heights in 1900 (in cm)
GirlsH1980 The average of 18-year-old girls heights in 1980 (in cm)
Happiness Score on a 0-10 scale for average level of happiness (10 being happiest)
GDPperCapita Gross Domestic Product (per capita)
FertRate The average number of children that will be born to a woman over her lifetime
PeopleVacc Total number of people vaccinated in the country
PeopleVacc_per100 Total number of people vaccinated in the country (in percent)
Population2010 Population (in millions) in 2010
Population2020 Population (in millions) in 2020
WineServ Average wine consumption per capita for those age 15 and over per week (collected by WHO)

Index

* datasets

- Ames, 3
- class_data, 5
- er, 10
- fevdata, 14
- Fingers, 15
- FingersMessy, 16
- game_data, 17
- penguins, 61
- Smallville, 66
- Survey, 74
- Tables, 74
- tip_exp, 76
- TipExperiment, 75
- World, 76

Ames, 3

aov(), 32, 36, 39, 41, 51, 54

b (estimate_extraction), 12

b0 (estimate_extraction), 12

b1 (estimate_extraction), 12

class_data, 5

coursekata_attach, 6

coursekata_install, 6

coursekata_load_theme, 7

coursekata_load_theme(), 9

coursekata_packages, 7

coursekata_palette, 8

coursekata_palette(), 7, 8, 62

coursekata_palette_provider, 8

coursekata_repos, 9

coursekata_unload_theme, 9, 75

coursekata_unload_theme(), 7

coursekata_update (coursekata_install), 6

data.frame, 13

er, 10

estimate_extraction, 12

f (estimate_extraction), 12

f(), 13

fevdata, 14

Fingers, 15

FingersMessy, 16

fit_stats, 17

fitstats (fit_stats), 17

formula, 13

fVal (estimate_extraction), 12

game_data, 17

generate_models(), 13

geom_b, 21

geom_b(), 34

geom_cutoff (GeomCutoff), 18

geom_cutoff(), 64

geom_model, 23

geom_model(), 22

geom_reduce (geom_resid), 25

geom_reduce(), 71

geom_resid, 25

geom_resid(), 19, 72

geom_square_reduce (geom_resid), 25

geom_square_reduce(), 71

geom_square_resid (geom_resid), 25

geom_square_resid(), 21, 72

geom_squareplot, 28

GeomCutoff, 18

GeomModel, 19

GeomResid, 19, 71

GeomSquareplot, 20, 73

GeomSquareResid, 21, 71

gf_b, 30

gf_b(), 21, 22

gf_coef (gf_b), 30

gf_model, 35

gf_model(), 30, 34, 66, 70

gf_reduce, 38

gf_reduce(), 52
 gf_resid, 40
 gf_resid(), 20, 38, 39, 43, 51, 54
 gf_resid_fun, 42
 gf_resid_fun(), 56
 gf_sd_ruler, 44
 gf_sd_ruler(), 73
 gf_square_reduce, 50
 gf_square_reduce(), 28, 52
 gf_square_resid, 53
 gf_square_resid(), 21, 28, 52, 57
 gf_square_resid_fun, 55
 gf_squarereduce (gf_square_reduce), 50
 gf_squareplot, 46
 gf_squareplot(), 20, 28, 30, 63, 66, 74
 gf_squaresid (gf_square_resid), 53
 ggformula::gf_boxplot(), 37
 ggformula::gf_histogram(), 37, 48
 ggformula::gf_hline(), 35
 ggformula::gf_jitter(), 37
 ggformula::gf_lm(), 35, 36
 ggformula::gf_point(), 37, 45
 ggformula::gf_segment(), 35
 ggformula::gf_violin(), 37
 ggplot2::aes(), 18, 29, 68, 70, 72
 ggplot2::coord_flip(), 18, 27
 ggplot2::Geom, 19–21
 ggplot2::geom_histogram(), 30
 ggplot2::geom_segment(), 27, 45
 ggplot2::geom_smooth(), 23
 ggplot2::ggplot(), 23, 25
 ggplot2::Guide, 58, 59
 ggplot2::guides(), 64
 ggplot2::layer(), 18, 27, 68, 70, 72
 ggplot2::position_jitter(), 27
 ggplot2::Stat, 69–71, 73
 ggplot2::stat_bin(), 30
 ggplot2::stat_smooth(), 24, 36
 ggplot2::theme_bw, 75
 guide_cutoff, 57
 guide_cutoff(), 64
 guide_dgp, 58
 GuideCutoff (guide_cutoff), 57
 GuideDgp (guide_dgp), 58

 lm, 12, 13, 17
 lm(), 32, 36, 39, 41, 51, 54
 lower (middle), 60

 middle, 60
 middle(), 60

 outer (middle), 60

 p (estimate_extraction), 12
 palmerpenguins::penguins, 61
 penguins, 61
 PRE (estimate_extraction), 12
 pre (estimate_extraction), 12
 pre(), 13

 remotes::install_cran, 6
 remotes::install_github, 6

 scale_discrete_coursekata, 62
 show_cutoffs, 63
 show_cutoffs(), 19, 68, 69
 show_dgp, 64
 show_dgp(), 49, 66
 show_mean, 65
 show_mean(), 49, 70
 Smallville, 66
 split_data, 67
 stat_cutoff (StatCutoff), 68
 stat_cutoff(), 64
 stat_dist_mean (StatDistMean), 69
 stat_model (geom_model), 23
 stat_model(), 36
 stat_reduce (geom_resid), 25
 stat_reduce(), 71
 stat_resid (geom_resid), 25
 stat_resid(), 72
 stat_sd_ruler (StatSdRuler), 72
 stat_sd_ruler(), 45
 stat_squareplot (geom_squareplot), 28
 StatCutoff, 68
 StatDistMean, 69
 StatModel, 22, 70
 StatReduce, 71
 StatResid, 19, 21, 71
 stats::aov(), 22, 23, 27
 stats::coef(), 30
 stats::lm(), 22, 23, 27
 StatSdRuler, 72
 StatSquareplot, 20, 73
 Survey, 74

 Tables, 74

`tails (middle)`, [60](#)
`theme_coursekata`, [75](#)
`theme_coursekata()`, [7](#)
`tip_exp`, [76](#)
`TipExperiment`, [75](#)

`upper (middle)`, [60](#)

`viridisLite::viridis()`, [8](#)

`World`, [76](#)