Package 'mosaicresid'

November 20, 2026

Type Package

Title Residual-Shaded Mosaic Plots for 'ggplot2'

Version 0.1.1

Description A 'ggplot2'-native alternative to the shaded mosaic plots drawn by mosaicplot() in base 'R'. geom_mosaic_resid() maps two categorical variables with ordinary 'ggplot2' aesthetics, recursively partitions the unit square into tiles sized by their joint proportions, computes Pearson, deviance, or Freeman-Tukey residuals against a two-way independence model, and maps the residuals to fill color. It works on raw data, already aggregated counts, or contingency tables, and the tile geometry and residuals are also returned as a data frame by compute_mosaic_residuals().

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

Imports ggplot2 (>= 3.4.0), dplyr, tibble, purrr, rlang

Suggests testthat (>= 3.0.0), knitr, rmarkdown, S7

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/sbutler22/geom_mosaic

BugReports https://github.com/sbutler22/geom_mosaic/issues

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Sam Butler [aut, cre]

Maintainer Sam Butler <swmbutler@gmail.com>

Repository CRAN

Date/Publication 2026-11-20 13:00:08 UTC

Contents

compute_mosaic_residuals 2
fortify.table 3
geom_mosaic_resid 4
ggplot_add.mosaicresid_layers 7
scale_fill_mosaic_residual 8
StatMosaicLabel 9
StatMosaicResidual 9
stat_mosaic_label 10
stat_mosaic_residual 12
to_long_counts 15

Index **16**

`compute_mosaic_residuals` *Compute mosaic tile geometry and independence residuals*

Description

The plain-data-frame engine behind `geom_mosaic_resid()`, with no ggplot2 involved: converts `data` to long-format counts, recursively partitions the unit square into tiles sized by joint proportion, and attaches independence-model residuals to each tile. Returned as an ordinary tibble, so it's directly inspectable – useful for checking the underlying numbers, or for driving a completely custom plot.

Usage

```
compute_mosaic_residuals(
  data,
  x,
  y,
  weight = NULL,
  spacing = 0.01,
  dir = NULL,
  type = c("pearson", "deviance", "FT")
)
```

Arguments

`data`	A data frame – either one row per observation, or already aggregated with an existing count/weight column.
`x, y`	Strings naming the two grouping variables.
`weight`	Optional string naming an existing count column.
`spacing`	Gap size (fraction of each split's dimension) between sibling rectangles at every level, passed through to the internal partitioning step. `0` = no gap.

dir	Optional character vector of "v"/"h" (one per variable, i.e. length 2 here) forcing the split direction at each level, matching base R's mosaicplot(dir =). NULL (default) alternates v/h starting with v, same as base R's default.
type	Residual formula: "pearson" (default), "deviance", or "FT" (Freeman-Tukey), matching base R's mosaicplot(type =).

Value

A tibble with one row per (x, y) cell: xmin/xmax/ymin/ ymax (tile bounds in [0,1] plot units), n (cell count), level, the x/y category columns, row_total/col_total/expected, and residual.

Examples

```
df <- data.frame(
  genre = sample(c("Action", "Sports", "Puzzle"), 200, replace = TRUE),
  platform = sample(c("PS", "Switch"), 200, replace = TRUE)
)
compute_mosaic_residuals(df, "genre", "platform")
```

---

fortify.table *Fortify a table object for ggplot2*

Description

Registers a method so ggplot(some_table) works directly on a base R table (e.g. Titanic, HairEyeColor, UCBAdmissions) – matching how base R's own mosaicplot() accepts a table as x. Without this method, ggplot() rejects a table object outright, since ggplot2 only fortifies objects it has a registered method for.

Usage

```
## S3 method for class 'table'
fortify(model, data, ...)
```

Arguments

model	A table object.
data	Unused; present for compatibility with the fortify() generic.
...	Unused.

Details

This is intentionally just as.data.frame(): a table's cell values already *are* the counts, so converting always produces one column per table dimension plus a literal Freq count column. See geom_mosaic_resid() for how that Freq column is then used automatically as an implicit weight.

Value

A data frame with one row per table cell.

`geom_mosaic_resid` *Residual-shaded mosaic plot*

Description

The main entry point for this package. A ggplot2-native equivalent of base R's `mosaicplot(shade = TRUE)`: give it two categorical variables the normal ggplot2 way, `aes(x =, y =)`, and it draws a true mosaic plot (tiles sized by joint proportion, not a grid approximation), shaded by independence-model residuals, with in-tile category labels – no manual stat work required.

Usage

```
geom_mosaic_resid(
  mapping = NULL,
  data = NULL,
  stat = "mosaic_residual",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  spacing = 0.01,
  label = TRUE,
  min_label_width = 0.02,
  min_label_height = 0.03,
  label.size = 3,
  label.color = "black",
  default_scale = TRUE,
  scale_low = "firebrick",
  scale_mid = "white",
  scale_high = "steelblue",
  scale_midpoint = 0,
  dir = NULL,
  type = c("pearson", "deviance", "FT"),
  ...
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.

A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).

stat — The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following:

- A Stat ggproto subclass, for example StatCount.
- A string naming the stat. To give the stat as a string, strip the function name of the stat_ prefix. For example, to use stat_count(), give the stat as "count".
- For more information and other ways to specify the stat, see the layer stat documentation.

position — A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:

- The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter".
- For more information and other ways to specify the position, see the layer position documentation.

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

show.legend — logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

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

spacing — Gap between sibling tiles, as a fraction of their parent's dimension.

label — Set to FALSE to get only the rectangles, no text labels.

min_label_width, min_label_height — Minimum cell width/height (0-1 fractional units) required for a label to be drawn; smaller cells are colored but left unlabeled rather than overflowing.

label.size, label.color — Passed through to the label's geom_text() layer.

default_scale — Set FALSE to skip the built-in color scale entirely (e.g. if you always add your own).

scale_low, scale_mid, scale_high, scale_midpoint — Passed through to the default scale (scale_fill_mosaic_residual()).

`dir` Optional character vector of `"v"`/`"h"` (one per variable, i.e. `c(x_direction, y_direction)`) forcing the split direction at each level – matches base R's `mosaicplot(dir = )`. `NULL` (default) alternates v/h starting with v, same as base R's default.

`type` Residual formula: `"pearson"` (default), `"deviance"`, or `"FT"` (Freeman-Tukey) – matches base R's `mosaicplot(type = )`.

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `....` Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `....` This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

Details

Returns a `list` of ggplot2 layers/scales – the rectangles (`GeomRect`), in-tile category labels (`GeomText`), and a default diverging fill scale – so a single call gives you all three. ggplot2 natively supports `+`-ing a list of layers/scales onto a plot, so this composes just like a normal single-layer geom would.

Value

A `list` of ggplot2 layers (and, by default, a scale), suitable for adding to a `ggplot()` call with `+`.

Examples

```
library(ggplot2)
df <- data.frame(
  genre = sample(c("Action", "Sports", "Puzzle"), 200, replace = TRUE),
  platform = sample(c("PS", "Switch"), 200, replace = TRUE)
)

# colors, spacing, and labels all come for free -- no manual
```

```
# scale_fill_gradient2() call needed, it's the default
ggplot(df) +
  geom_mosaic_resid(aes(x = genre, y = platform)) +
  theme_minimal()

# the default scale is fully overridable like any ggplot2 layer
ggplot(df) +
  geom_mosaic_resid(aes(x = genre, y = platform)) +
  scale_fill_gradient2(low = "purple", mid = "white", high = "darkgreen") +
  theme_minimal()
```

`ggplot_add.mosaicresid_layers` *Add mosaicresid layers to a plot (ggplot2 < 4.0 entry point)*

Description

The classic S3 `ggplot_add()` method that handles `+`-ing a `mosaicresid_layers` list onto a plot on `ggplot2` versions before the 4.0 S7 rewrite. See `add_mosaicresid_layers()` for what this actually does; on `ggplot2` >= 4.0 the equivalent work happens via the S7 method registered in `zzz.R` instead, since this S3 method is never reached there.

Usage

```
## S3 method for class 'mosaicresid_layers'
ggplot_add(object, plot, object_name, ...)
```

Arguments

`object`	A `mosaicresid_layers` list, as returned by `geom_mosaic_resid()`.
`plot`	The `ggplot` object being added to.
`object_name`	The name of `object`, for error messages (required by the `ggplot_add()` generic's signature; unused here).
`...`	Unused; present for compatibility with the `ggplot_add()` generic, whose signature differs between ggplot2 versions.

Value

The updated `ggplot` object.

`scale_fill_mosaic_residual` *Default diverging fill scale for mosaic residuals*

Description

A continuous red-white-blue gradient (red = negative residual / under-represented, blue = positive / over-represented), rather than base R's discrete HSV-stepped bins. Chosen as the default because it reads better alongside in-tile labels – there's no hard color-bin boundary cutting across a cell's text the way base R's stepped bins can. It's still an ordinary ggplot2 scale object, so it composes and overrides the usual way.

Usage

```
scale_fill_mosaic_residual(
  low = "firebrick",
  mid = "white",
  high = "steelblue",
  midpoint = 0,
  name = "Residual",
  ...
)
```

Arguments

`low`, `mid`, `high`	Colors for the most negative, zero, and most positive residual.
`midpoint`	Residual value treated as the color midpoint (`0` = independence).
`name`	Legend title.
`...`	Passed on to `ggplot2::scale_fill_gradient2()`.

Value

A ggplot2 `ScaleContinuous` object.

Examples

```
library(ggplot2)
df <- data.frame(
  genre = sample(c("Action", "Sports", "Puzzle"), 200, replace = TRUE),
  platform = sample(c("PS", "Switch"), 200, replace = TRUE)
)
ggplot(df) +
  geom_mosaic_resid(aes(x = genre, y = platform), default_scale = FALSE) +
  scale_fill_mosaic_residual()
```

`StatMosaicLabel` *StatMosaicLabel*

Description

Computes label text + centroid position for each mosaic cell, so category names can be drawn directly inside their rectangles. This sidesteps a real constraint of mosaic plots: the inner variable's split positions differ from column to column (they're conditional on the outer variable), so there's no single shared axis tick that's correct for the whole plot. In-tile labels are the structurally correct fix, not a workaround.

Details

Cells smaller than `min_label_width`/`min_label_height` (in the same 0-1 fractional units as `xmin`/`xmax`/`ymin`/`ymax`) are dropped from the label set entirely – they're still colored by the rect layer, just with no text crammed into a sliver. These are exposed as tunable parameters because the "right" threshold depends on plot size, font size, and label text length, not on the data alone.

Value

A `ggproto` object of class `StatMosaicLabel` (inheriting from ggplot2::Stat). It is not called directly; it is the `stat` used by `stat_mosaic_label()` and by the label layer of `geom_mosaic_resid()`. During plot building it returns one row per labeled tile, with the tile's center (`x`, `y`) and its `label` text (the `x` and `y` categories on two lines). Tiles smaller than `min_label_width` or `min_label_height` are left out.

`StatMosaicResidual` *StatMosaicResidual*

Description

Takes discrete `catx`/`caty` aesthetics (and optional `weight`), computes mosaic rectangles + independence residuals, and hands `xmin`/`xmax`/`ymin`/`ymax` + `residual` off to a geom (`GeomRect` by default).

Value

A `ggproto` object of class `StatMosaicResidual` (inheriting from ggplot2::Stat). It is not called directly; it is the `stat` used by `stat_mosaic_residual()` and `geom_mosaic_resid()`. During plot building it returns one row per combination of the `x` and `y` categories, with tile bounds `xmin`, `xmax`, `ymin`, `ymax` (in 0-1 plot units), the cell count `n`, the `expected` count under independence, and the `residual` used for the fill color.

`stat_mosaic_label` *Compute in-tile mosaic labels as a ggplot2 stat*

Description

The stat underlying `geom_mosaic_resid()`'s in-tile category labels. Use this directly if you want the labels without the default rectangles and fill scale.

Usage

```
stat_mosaic_label(
  mapping = NULL,
  data = NULL,
  geom = "text",
  position = "identity",
  na.rm = FALSE,
  spacing = 0.01,
  min_label_width = 0.02,
  min_label_height = 0.03,
  dir = NULL,
  show.legend = FALSE,
  inherit.aes = TRUE,
  ...
)
```

Arguments

`mapping`	Set of aesthetic mappings created by `aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.
`data`	The data to be displayed in this layer. There are three options: If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`. A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created. A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).
`geom`	The geometric object to use to display the data for this layer. When using a `stat_*()` function to construct a layer, the `geom` argument can be used to override the default coupling between stats and geoms. The `geom` argument accepts the following: • A `Geom` ggproto subclass, for example `GeomPoint`.

- A string naming the geom. To give the geom as a string, strip the function name of the `geom_` prefix. For example, to use `geom_point()`, give the geom as `"point"`.
- For more information and other ways to specify the geom, see the layer geom documentation.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`na.rm` If `FALSE` (the default), missing values are removed with a warning; if `TRUE`, missing values are silently removed.

`spacing` Gap between sibling tiles, as a fraction of their parent's dimension – must match whatever `spacing` was passed to the paired `stat_mosaic_residual()`/`geom_mosaic_resid()` call, so labels land on the same rectangles.

`min_label_width`, `min_label_height`
Minimum cell width/height (in 0-1 fractional plot units) required for a label to be drawn at all. Tune these per-plot: smaller plots or longer category names typically need larger thresholds to avoid overflow.

`dir` Optional character vector of `"v"`/`"h"` forcing split direction per level – must match whatever `dir` was passed to the paired `stat_mosaic_residual()`/`geom_mosaic_resid()` call, so labels land on the same rectangles. `NULL` (default) alternates v/h.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

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

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.

- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the geom part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `....` This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

Value

A `ggplot2` layer object (class `LayerInstance`) that can be added to a plot with `+`. When the plot is built, the layer writes each tile's `x` and `y` category names at the center of that tile, skipping tiles too small to hold the text.

Examples

```
library(ggplot2)
df <- data.frame(
  genre = sample(c("Action", "Sports", "Puzzle"), 200, replace = TRUE),
  platform = sample(c("PS", "Switch"), 200, replace = TRUE)
)
ggplot(df) +
  stat_mosaic_residual(aes(x = genre, y = platform)) +
  stat_mosaic_label(aes(x = genre, y = platform))
```

---

stat_mosaic_residual *Compute mosaic tiles and independence residuals as a ggplot2 stat*

Description

The stat underlying `geom_mosaic_resid()`'s rectangles. Use this directly if you want to pair the mosaic-residual computation with a geom other than `GeomRect`.

Usage

```
stat_mosaic_residual(
  mapping = NULL,
  data = NULL,
  geom = "rect",
  position = "identity",
  na.rm = FALSE,
  spacing = 0.01,
  dir = NULL,
  type = c("pearson", "deviance", "FT"),
```

```
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
geom	The geometric object to use to display the data for this layer. When using a stat_*() function to construct a layer, the geom argument can be used to override the default coupling between stats and geoms. The geom argument accepts the following: • A Geom ggproto subclass, for example GeomPoint. • A string naming the geom. To give the geom as a string, strip the function name of the geom_ prefix. For example, to use geom_point(), give the geom as "point". • For more information and other ways to specify the geom, see the layer geom documentation.
position	A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following: • The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position. • A string naming the position adjustment. To give the position as a string, strip the function name of the position_ prefix. For example, to use position_jitter(), give the position as "jitter". • For more information and other ways to specify the position, see the layer position documentation.
na.rm	If FALSE (the default), missing values are removed with a warning; if TRUE, missing values are silently removed.
spacing	Gap between sibling tiles, as a fraction of their parent's dimension.
dir	Optional character vector of "v"/"h" (one per variable: here, c(x_direction, y_direction)) forcing the split direction at each level, matching base R's mosaicplot(dir =). NULL (default) alternates v/h starting with v, same as base R's default.

`type` Residual formula: `"pearson"` (default), `"deviance"`, or `"FT"` (Freeman-Tukey), matching base R's `mosaicplot(type = )`.

`show.legend` logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

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

`...` Other arguments passed on to `layer()`'s `params` argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the `position` argument, or aesthetics that are required can *not* be passed through `...`. Unknown arguments that are not part of the 4 categories below are ignored.

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `...`. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

Value

A `ggplot2` layer object (class `LayerInstance`) that can be added to a plot with `+`. When the plot is built, the layer draws one tile per combination of the `x` and `y` categories, sized by that combination's share of the total count and filled by its residual against a two-way independence model (positive = more observations than independence predicts, negative = fewer).

Examples

```
library(ggplot2)
df <- data.frame(
  genre = sample(c("Action", "Sports", "Puzzle"), 200, replace = TRUE),
  platform = sample(c("PS", "Switch"), 200, replace = TRUE)
)
ggplot(df) + stat_mosaic_residual(aes(x = genre, y = platform))
```

`to_long_counts` *Convert raw or pre-aggregated data into long-format counts*

Description

Accepts either one-row-per-observation data or data already aggregated with an existing count column (via `weight`), and returns a single tidy long-format count table either way – the same code path is used downstream regardless of which kind of input was supplied.

Usage

```
to_long_counts(data, x, y, weight = NULL)
```

Arguments

`data`	A data frame – either one row per observation, or already aggregated with an existing count/weight column.
`x, y`	Strings naming the two grouping variables.
`weight`	Optional string naming an existing count column. Leave `NULL` if `data` is one-row-per-observation.

Details

NA handling matches base R's `table()`/`mosaicplot()` default (`useNA = "no"`): rows with `NA` in either grouping variable are silently excluded, not counted as their own category. Unused factor levels are dropped so phantom zero-total categories never enter the partitioning step.

Value

A tibble with columns `x`, `y`, and `n` (named after the values of the `x`/`y` arguments, plus a literal `n` count column).

Examples

```
df <- data.frame(
  genre = sample(c("Action", "Sports", "Puzzle"), 200, replace = TRUE),
  platform = sample(c("PS", "Switch"), 200, replace = TRUE)
)
to_long_counts(df, "genre", "platform")
```

Index

aes(), *4*, *10*, *13*
annotation_borders(), *5*, *11*, *14*

compute_mosaic_residuals, 2

fortify(), *4*, *10*, *13*
fortify.table, 3

geom_mosaic_resid, 4
geom_mosaic_resid(), *2, 3*, *7*, *9, 10*, *12*
ggplot(), *4*, *10*, *13*
ggplot2::scale_fill_gradient2(), *8*
ggplot2::Stat, *9*
ggplot_add.mosaicresid_layers, 7

key glyphs, *6*, *12*, *14*

layer geom, *11*, *13*
layer position, *5*, *11*, *13*
layer stat, *5*
layer(), *6*, *11, 12*, *14*

scale_fill_mosaic_residual, 8
scale_fill_mosaic_residual(), *5*
stat_mosaic_label, 10
stat_mosaic_label(), *9*
stat_mosaic_residual, 12
stat_mosaic_residual(), *9*
StatMosaicLabel, 9
StatMosaicResidual, 9

to_long_counts, 15