Package 'ggpsychro'

July 30, 2026

Title Create Psychrometric Charts

Version 0.1.0

Description Provides 'ggplot2' coordinates, layers, scales, themes, and presets for creating psychrometric charts. The package supports metric and inch-pound unit systems, psychrometric grids, state points, process lines, zones, and thermal comfort overlays for heating, ventilation, air conditioning, and building performance workflows. Psychrometric property calculations are based on 'PsychroLib' (Meyer and Thevenard, 2019) <doi:10.21105/joss.01137> where appropriate.

Depends R (>= 4.1), ggplot2 (>= 4.0.0)

Imports gridGeometry, polyclip, isoband, checkmate, psychrolib, S7, scales (>= 1.1.0)

Suggests testthat (>= 3.0.0), vdiffr

License MIT + file LICENSE

URL https://hongyuanjia.github.io/ggpsychro/, https://github.com/hongyuanjia/ggpsychro

BugReports https://github.com/hongyuanjia/ggpsychro/issues

Encoding UTF-8

RoxygenNote 7.3.3

Collate 'aaa-.R' 'stat.R' 'utils.R' 'stat-psychro-bin.R' 'psychro-extra.R' 'psychro-state.R' 'comfort-core.R' 'comfort-adaptive.R' 'comfort-grid.R' 'comfort-band.R' 'comfort-heat-index.R' 'comfort-pmv-label.R' 'comfort-native.R' 'comfort-model.R' 'comfort-dispatch.R' 'comfort-pmv-curve.R' 'comfort-pmv-band.R' 'comfort-pmv.R' 'comfort-calc.R' 'comfort-contour.R' 'comfort-givoni.R' 'comfort-zone.R' 'comfort-stat.R' 'comfort-stat-pmv.R' 'comfort-layer-field.R' 'scale-comfort.R' 'comfort-stat-givoni.R' 'comfort-layer-givoni.R' 'comfort-stat-heat-index.R' 'comfort-layer-heat-index.R' 'comfort-layer-pmv.R' 'coord-psychro.R' 'coord-clip.R' 'coord-foreground.R' 'psychro-tile-grid.R' 'psychro-zone.R' 'psychro-tile-clip.R'

'geom-psychro-tile.R' 'ggpsychro-package.R' 'ggpsychro.R'
'grid-label-renderer.R' 'grid-layer.R' 'guide-label.R'
'guide-panel.R' 'guide-protractor.R' 'guides-grid.R' 'labels.R'
'layer.R' 'layout.R' 'plot-build.R' 'plot-construction.R'
'trans.R' 'scale.R' 'theme.R' 'zzz.R'

Config/testthat/edition 3

NeedsCompilation yes

Author Hongyuan Jia [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0002-0075-8183>)

Maintainer Hongyuan Jia <hongyuanjia@cqust.edu.cn>

Repository CRAN

Date/Publication 2026-07-30 12:40:13 UTC

Contents

ggpsychro-package *ggpsychro: psychrometric charts with ggplot2*

Description

ggpsychro extends ggplot2 with psychrometric coordinates, grids, data layers, zones, process lines, and thermal-comfort overlays.

Author(s)

Maintainer: Hongyuan Jia <hongyuanjia@cqust.edu.cn> (ORCID) [copyright holder]

See Also

Useful links:

- https://hongyuanjia.github.io/ggpsychro/
- https://github.com/hongyuanjia/ggpsychro
- Report bugs at https://github.com/hongyuanjia/ggpsychro/issues

comfort_model_pmv *Comfort model objects*

Description

Model objects capture the fixed inputs used by comfort layers. They can be reused across overlays, contours, zones, and point states.

Usage

```
comfort_model_pmv(
  tr = NULL,
  vr = 0.1,
  met = 1.2,
  clo = 0.5,
  wme = 0,
  model = "7730-2005",
  limit_inputs = FALSE,
  round_output = FALSE
)

comfort_model_set(
  tr = NULL,
  v = 0.1,
  met = 1.2,
```

```
  clo = 0.5,
  wme = 0,
  limit_inputs = FALSE,
  body_surface_area = 1.8258,
  p_atm = NULL,
  position = c("standing", "sitting"),
  round_output = FALSE
)

comfort_model_adaptive(
  t_running,
  tr = NULL,
  v = 0.1,
  standard = c("ashrae55", "en16798"),
  category = NULL,
  limit_inputs = TRUE,
  round_output = FALSE
)

comfort_model_heat_index(
  solar_exposure = 0,
  limit_inputs = TRUE,
  round_output = FALSE
)
```

Arguments

tr	Mean radiant temperature. Defaults to tdb.
vr	Relative air speed for PMV.
met	Metabolic rate in met.
clo	Clothing insulation in clo.
wme	External work in met.
model	PMV model/version label. Currently "7730-2005" is implemented.
limit_inputs	If TRUE, values outside the model applicability range are returned as NA.
round_output	If TRUE, round model outputs. Comfort plot layers use unrounded values by default so contours and zones remain smooth.
v	Air speed for SET and adaptive comfort.
body_surface_area	Body surface area in square meters for SET.
p_atm	Atmospheric pressure in Pa.
position	Body position, "standing" or "sitting".
t_running	Running mean outdoor temperature.
standard	Adaptive comfort standard.
category	Adaptive comfort category.
solar_exposure	Relative solar exposure for heat index, from 0 to 1.

Details

`comfort_model_*()` helpers are for chart layers and stats. For direct vectorized calculations, use `comfort_pmv()`, `comfort_set()`, `comfort_adaptive()`, or `comfort_heat_index()`. For drawing on a psychrometric chart, pass a model object to `geom_comfort_*()` layers.

Value

A comfort model object.

Examples

```
# Create a PMV model object.
comfort_model_pmv(met = 1.4, clo = 0.5)

# Create a SET model object.
comfort_model_set(v = 0.2)

# Create an adaptive comfort model object.
comfort_model_adaptive(t_running = 22)

# Create a heat-index model object.
comfort_model_heat_index(solar_exposure = 0.5)

# Draw a PMV overlay using custom activity and clothing assumptions.
ggpsychro(tdb_lim = c(15, 35), hum_lim = c(0, 24)) +
    geom_comfort_pmv(
        model = comfort_model_pmv(met = 1.4, clo = 0.5),
        n = c(45, 30)
    )

# Draw SET as the filled comfort metric.
ggpsychro(tdb_lim = c(15, 35), hum_lim = c(0, 24)) +
    geom_comfort_set(
        model = comfort_model_set(v = 0.2),
        n = c(45, 30)
    )

# Draw the adaptive acceptability region.
ggpsychro(tdb_lim = c(15, 35), hum_lim = c(0, 24)) +
    geom_comfort_adaptive(
        t_running = 22,
        n = c(45, 30),
        alpha = 0.3
    )

# Draw heat-index categories with a solar exposure adjustment.
ggpsychro(tdb_lim = c(25, 45), hum_lim = c(0, 32)) +
    geom_comfort_heat_index(
        model = comfort_model_heat_index(solar_exposure = 0.5),
        n = c(55, 35)
    )
```

comfort_pmv *Thermal comfort calculations*

Description

These functions evaluate common comfort models without requiring Python at runtime. Relative humidity is always supplied in percent. Temperature and air-speed inputs follow `units`: SI uses degree C and m/s; IP uses degree F and ft/s.

Usage

```
comfort_pmv(
  tdb,
  tr = tdb,
  vr = 0.1,
  rh,
  met = 1.2,
  clo = 0.5,
  wme = 0,
  units = c("SI", "IP"),
  limit_inputs = TRUE,
  round_output = TRUE
)

comfort_set(
  tdb,
  tr = tdb,
  v = 0.1,
  rh,
  met = 1.2,
  clo = 0.5,
  wme = 0,
  units = c("SI", "IP"),
  limit_inputs = TRUE,
  round_output = TRUE,
  body_surface_area = 1.8258,
  p_atm = 101325,
  position = c("standing", "sitting")
)

comfort_adaptive(
  tdb,
  tr = tdb,
  t_running,
  v = 0.1,
  standard = c("ashrae55", "en16798"),
  category = NULL,
```

```
  units = c("SI", "IP"),
  limit_inputs = TRUE,
  round_output = TRUE
)

comfort_heat_index(
  tdb,
  rh,
  solar_exposure = 0,
  units = c("SI", "IP"),
  limit_inputs = TRUE,
  round_output = TRUE
)
```

Arguments

tdb	Dry-bulb air temperature.
tr	Mean radiant temperature. Defaults to tdb.
vr	Relative air speed for PMV.
rh	Relative humidity in percent.
met	Metabolic rate in met.
clo	Clothing insulation in clo.
wme	External work in met.
units	Unit system, "SI" or "IP".
limit_inputs	If TRUE, values outside the model applicability range are returned as NA.
round_output	If TRUE, round outputs like pythermalcomfort.
v	Air speed for SET and adaptive comfort.
body_surface_area	Body surface area in square meters for SET.
p_atm	Atmospheric pressure in Pa.
position	Body position, "standing" or "sitting".
t_running	Running mean outdoor temperature for adaptive comfort.
standard	Adaptive comfort standard, either "ashrae55" or "en16798".
category	Comfort category. For ASHRAE 55 use "80" or "90"; for EN 16798 use "I", "II", or "III".
solar_exposure	Relative solar exposure for heat index, from 0 to 1.

Details

Use these comfort_*() functions for vectorized data-frame calculations. Use comfort_model_pmv() and the other comfort_model_*() helpers to store fixed model assumptions for chart layers, and use geom_comfort_*() layers to draw comfort fields or zones on a psychrometric chart.

Value

A data frame with model outputs.

Output columns

- `comfort_pmv()` returns `pmv`, `ppd`, and `tsv` (thermal sensation vote).
- `comfort_set()` returns `set`.
- `comfort_adaptive()` returns the selected standard, acceptability flags, comfort temperature, and upper/lower comfort-temperature limits. ASHRAE 55 includes 80% and 90% acceptability columns; EN 16798 includes category I, II, and III acceptability columns.
- `comfort_heat_index()` returns `heat_index`, `category`, and `category_id`.

Examples

```
comfort_pmv(25, rh = 50, met = 1.4, clo = 0.5)
comfort_set(25, rh = 50)
comfort_adaptive(25, t_running = 20)
comfort_heat_index(32, rh = 70)
```

`comfort_pmv_ashrae55` *PMV-based comfort standards*

Description

These helpers describe static PMV-based comfort zones. They are distinct from adaptive comfort models such as `comfort_model_adaptive()`, which use running mean outdoor temperature and produce operative-temperature bands.

Usage

```
comfort_pmv_ashrae55(edition = "2017", range = c(-0.5, 0.5))

comfort_pmv_en15251(edition = "2007", breaks = c(-0.7, -0.2, 0.2, 0.7))
```

Arguments

`edition`	Standard edition. Currently `"2017"` for ASHRAE 55 and `"2007"` for EN 15251.
`range`	PMV comfort interval for ASHRAE 55.
`breaks`	PMV boundaries for EN 15251 comfort bands.

Value

A comfort standard object.

Examples

```
# Create the ASHRAE 55 PMV comfort interval.
comfort_pmv_ashrae55()

# Create the EN 15251 PMV comfort bands.
comfort_pmv_en15251()

# Draw the ASHRAE 55 comfort zone.
ggpsychro(tdb_lim = c(15, 35), hum_lim = c(0, 24)) +
    geom_comfort_pmv(
        standard = comfort_pmv_ashrae55(),
        bands = FALSE,
        contours = FALSE,
        n = 80
    )

# Draw the EN 15251 comfort bands.
ggpsychro(tdb_lim = c(15, 35), hum_lim = c(0, 24)) +
    geom_comfort_pmv(
        standard = comfort_pmv_en15251(),
        bands = FALSE,
        contours = FALSE,
        n = 80
    )
```

---

comfort_strategy_givoni

Givoni-Milne strategy overlay

Description

`comfort_strategy_givoni()` stores the fixed inputs used by `geom_comfort_givoni()`. By default it anchors the comfort zone to the Givoni/Milne 1979 bounds: 20 to 25.5 degrees C dry-bulb temperature and 20% to 80% relative humidity with the hot-humid corner clipped. The adaptive variant shifts the base comfort zone from a mean outdoor temperature. Strategy zones are drawn in dry-bulb/relative-humidity space before conversion to humidity ratio.

Usage

```
comfort_strategy_givoni(
  mean_outdoor = NULL,
  units = c("SI", "IP"),
  variant = c("fixed", "adaptive"),
  tdb_range = NULL,
  relhum_range = c(20, 80)
)
```

Arguments

`mean_outdoor`	Mean or running-mean outdoor temperature used when `variant = "adaptive"`. It is ignored when `variant = "fixed"` and can be `NULL` for that variant.
`units`	Unit system for `mean_outdoor`, `"SI"` or `"IP"`.
`variant`	Givoni-Milne strategy variant. `"fixed"` uses the fixed 1979 comfort anchor; `"adaptive"` shifts the comfort anchor from `mean_outdoor`.
`tdb_range`	Optional dry-bulb comfort-anchor range used when `variant = "fixed"`. Values use `units`. When `NULL`, the fixed variant uses 20 to 25.5 degrees C.
`relhum_range`	Relative-humidity comfort-anchor range in percent.

Details

This overlay is a climate-screening and design-strategy aid. It should not be interpreted as a comfort-standard compliance method or as a substitute for building energy/thermal simulation. In particular, the high-mass and night ventilation regions indicate potential strategy ranges; using them to count comfort hours requires daily temperature profiles, nighttime conditions, and building assumptions that are outside this layer.

Value

A Givoni-Milne comfort strategy object.

References

Milne M, Givoni B. Architectural design based on climate. In: Watson D, ed. Energy Conservation Through Building Design. McGraw-Hill; 1979:96-113.

Givoni B. Comfort, climate analysis and building design guidelines. Energy and Buildings. 1992;18(1):11-23. doi:10.1016/03787788(92)90047K

Herb S, Wolk S, Reinhart C. Beyond the bioclimatic chart: An automated simulation-based method for the assessment of natural ventilation and passive design potential. Building and Environment, 269, 112362. doi:10.1016/j.buildenv.2024.112362

Examples

```
# Create the fixed Givoni/Milne 1979 strategy.
comfort_strategy_givoni()

# Create an adaptive Givoni-Milne strategy for a warm outdoor mean.
comfort_strategy_givoni(variant = "adaptive", mean_outdoor = 22)

# Or provide project-specific comfort anchor ranges.
comfort_strategy_givoni(
    tdb_range = c(22, 27),
    relhum_range = c(30, 70)
)

# Draw the Givoni strategy overlay for that outdoor mean.
ggpsychro(tdb_lim = c(5, 45), hum_lim = c(0, 30)) +
```

```
    geom_comfort_givoni(
        strategy = comfort_strategy_givoni(
            variant = "adaptive",
            mean_outdoor = 22
        ),
        labels = FALSE
    )
```

---

coord_psychro *Advanced psychrometric coordinates*

Description

Advanced psychrometric coordinates

Usage

```
coord_psychro(
  tdb_lim = NULL,
  hum_lim = NULL,
  altitude = NULL,
  units = NULL,
  mollier = NULL,
  expand = FALSE,
  default = TRUE,
  clip = "on"
)
```

Arguments

tdb_lim	A numeric vector of length-2 indicating the dry-bulb temperature limits. Should be in range `[-50, 100]` degrees C [SI] or `[-58, 212]` degrees F [IP]. If `NULL`, trained data ranges will be used when available, otherwise a default display range will be used. Default: `NULL`.
hum_lim	A numeric vector of length-2 indicating the humidity ratio limits. Should be in range `[0, 60]` g H2O per kg dry air [SI] or `[0, 420]` grains H2O per lb dry air [IP]. If `NULL`, trained data ranges will be used when available, otherwise a default display range will be used. Default: `NULL`.
altitude	A single number of altitude in m [SI] or ft [IP]. If `NULL`, inherits the altitude from the parent `ggpsychro()` plot.
units	Unit system, either `"SI"` or `"IP"`. If `NULL`, inherits the unit system from the parent `ggpsychro()` plot.
mollier	If `TRUE`, use Mollier chart coordinates. If `NULL`, inherits the chart type from the parent `ggpsychro()` plot.
expand	If `TRUE`, add a small expansion factor to the limits. Defaults to `FALSE` for psychrometric charts.

default	Is this the default coordinate system? Defaults to TRUE so replacing the coordinate system created by ggpsychro() does not emit a ggplot2 replacement message.
clip	Should drawing be clipped to the extent of the plot panel? A setting of "on" (the default) means yes, and a setting of "off" means no. In most cases, the default of "on" should not be changed, as setting clip = "off" can cause unexpected results. It allows drawing of data points anywhere on the plot, including in the plot margins. If limits are set via xlim and ylim and some data points fall outside those limits, then those data points may show up in places such as the axes, the legend, the plot title, or the plot margins.

Details

Most plots should start with ggpsychro(), which installs this coordinate system and the matching psychrometric metadata for you. Call coord_psychro() directly only when you are replacing or configuring the coordinate system on an existing ggpsychro plot. When altitude, units, or mollier is NULL, the value is inherited from the parent plot. Supply these arguments explicitly when using the coordinate system outside that path.

Value

A ggplot2 coordinate system object for psychrometric charts.

Examples

```
ggpsychro() +
    coord_psychro(tdb_lim = c(10, 35), hum_lim = c(0, 25))

ggpsychro(units = "IP", altitude = 1000) +
    coord_psychro(
        tdb_lim = c(50, 100),
        hum_lim = c(0, 140),
        units = "IP",
        altitude = 1000
    )

ggpsychro(mollier = TRUE) +
    coord_psychro(
        tdb_lim = c(0, 50),
        hum_lim = c(0, 30),
        mollier = TRUE
    )
```

---

demo_scale *Demonstrate psychrometric label and scale functions*

Description

This helper builds a compact ggplot2 scale preview for label and scale functions.

Usage

```
demo_scale(x, ...)
```

Arguments

x	A vector of data
...	Other arguments pass to scale functions

Value

A ggplot object demonstrating the supplied scale settings.

Examples

```
demo_scale(0:10, labels = scales::label_number())
```

---

drybulb_trans *Create transformation objects for psychrometric chart*

Description

Create transformation objects for psychrometric chart

Usage

```
drybulb_trans(units = "SI")

humratio_trans(units = "SI")

relhum_trans(units = "SI")

wetbulb_trans(units = "SI")

vappres_trans(units = "SI")

specvol_trans(units = "SI")

enthalpy_trans(units = "SI")
```

Arguments

units	A string indicating the system of units chosen. Should be either "SI" or "IP".

Value

A scales::trans_new() transformation object.

Examples

```
plot(drybulb_trans("SI"), xlim = c(0, 5))
plot(humratio_trans("SI"), xlim = c(0, 1000))
plot(relhum_trans("SI"), xlim = c(0, 100))
plot(wetbulb_trans("SI"), xlim = c(-50, 40))
plot(vappres_trans("SI"), xlim = c(1000, 4000))
plot(specvol_trans("SI"), xlim = c(0.8, 1))
plot(enthalpy_trans("SI"), xlim = c(1000, 2000))
```

---

element_givoni_zone *Comfort zone style element*

Description

`element_givoni_zone()` creates a small style object for comfort strategy zones. It is used by `geom_comfort_givoni()` through the `zone_style` argument to override the default Givoni-Milne zone styles.

Usage

```
element_givoni_zone(
  fill = ggplot2::waiver(),
  colour = ggplot2::waiver(),
  linewidth = ggplot2::waiver(),
  linetype = ggplot2::waiver(),
  alpha = ggplot2::waiver(),
  linejoin = ggplot2::waiver(),
  color = NULL
)
```

Arguments

`fill`, `colour`, `color`, `linewidth`, `linetype`, `alpha`, `linejoin`
: Zone drawing properties. Values left as `ggplot2::waiver()` inherit the layer default.

Value

A comfort zone style element.

Examples

```
# Fill and outline the comfort zone with custom colours.
ggpsychro(tdb_lim = c(5, 45), hum_lim = c(0, 30)) +
    geom_comfort_givoni(
        labels = FALSE,
        zone_style = list(
            comfort = element_givoni_zone(
```

```
                fill = "#6FCF97",
                colour = "#1B7F4A",
                alpha = 0.35
            )
        )
    )

# Emphasize the air-conditioning region with a light fill.
ggpsychro(tdb_lim = c(5, 45), hum_lim = c(0, 30)) +
    geom_comfort_givoni(
        labels = FALSE,
        zone_style = list(
            air_conditioning = element_givoni_zone(
                fill = "#7BC8F6",
                colour = "#1B5E8C",
                alpha = 0.18,
                linetype = "solid"
            )
        )
    )

# Restyle a line-only region without filling it.
ggpsychro(tdb_lim = c(5, 45), hum_lim = c(0, 30)) +
    geom_comfort_givoni(
        labels = FALSE,
        zone_style = list(
            winter = element_givoni_zone(
                colour = "#C44536",
                linewidth = 1.2,
                linetype = "dashed"
            )
        )
    )
```

`geom_comfort_adaptive` *Draw adaptive comfort zones*

Description

`geom_comfort_adaptive()` draws the adaptive comfort acceptability region for ASHRAE 55 or EN 16798.

Usage

```
geom_comfort_adaptive(
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
```

```
  model = NULL,
  t_running = NULL,
  tr = NULL,
  v = 0.1,
  standard = c("ashrae55", "en16798"),
  category = NULL,
  n = NULL,
  gap = 0,
  alpha = 0.3,
  na.rm = FALSE,
  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)).
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.
...	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.

`model` An adaptive comfort model object. If `NULL`, one is created from `t_running`, `tr`, `v`, `standard`, and `category`.

`t_running` Running mean outdoor temperature when `model` is `NULL`.

`tr` Mean radiant temperature. If `NULL`, the model uses dry-bulb temperature.

`v` Air speed when `model` is `NULL`.

`standard` Adaptive comfort standard used when `model` is `NULL`.

`category` Adaptive comfort category.

`n` Grid resolution in dry-bulb and humidity-ratio directions. If `NULL`, adaptive comfort zones use `c(240, 160)`.

`gap` Relative gap between generated tiles.

`alpha` Layer transparency.

`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()`.

Details

Adaptive comfort zones are sampled on a dry-bulb and humidity-ratio grid. Increase `n` for smoother zone boundaries and decrease it for faster exploratory builds.

Value

A ggplot layer.

Examples

```
ggpsychro(tdb_lim = c(15, 35), hum_lim = c(0, 24)) +
    geom_comfort_adaptive(t_running = 22, alpha = 0.3)
```

`geom_comfort_givoni` *Draw Givoni-Milne strategy zones*

Description

`geom_comfort_givoni()` draws a Givoni-Milne strategy overlay, the optional mean outdoor-temperature marker for adaptive strategies, and optional zone labels.

Usage

```
geom_comfort_givoni(
  strategy = comfort_strategy_givoni(),
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
  alpha = 0.55,
  labels = TRUE,
  show_pmv = FALSE,
  pmv_model = comfort_model_pmv(),
  zone_alpha = 0.2,
  zone_style = NULL,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

`strategy`	A Givoni-Milne strategy object.
`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)`).

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.

... — 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.

alpha — Layer transparency for the optional PMV background.

labels — If TRUE, draw strategy labels.

show_pmv — If TRUE, draw the PMV comfort background under the Givoni strategy outlines.

pmv_model — PMV model used when show_pmv = TRUE.

zone_alpha — Alpha for the filled Givoni comfort zone. Other Givoni strategy regions are drawn as outlines.

zone_style — Optional named list of per-zone style overrides. Names must match Givoni zone ids such as "comfort", "winter", or "air_conditioning". Values can be created with element_givoni_zone(), ggplot2::element_polygon(), or ordinary named lists with fields fill, colour/color, linewidth, linetype, alpha, and linejoin.

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().

Value

A list of ggplot additions.

Examples

```
ggpsychro(tdb_lim = c(5, 45), hum_lim = c(0, 30)) +
    geom_comfort_givoni(labels = TRUE)
```

---

geom_comfort_heat_index

Draw heat-index comfort categories

Description

geom_comfort_heat_index() draws Outdoor Work Heat Index categories, boundary lines, and optional category labels.

Usage

```
geom_comfort_heat_index(
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
  model = comfort_model_heat_index(),
  n = c(160, 100),
  alpha = 0.55,
  labels = TRUE,
  na.rm = FALSE,
  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)).
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.
...	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.

model	A heat-index comfort model object.
n	Grid resolution in dry-bulb and humidity-ratio directions. Defaults to c(160, 100).
alpha	Layer transparency.
labels	If TRUE, draw category labels.
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().

Details

Heat-index categories are sampled on a dry-bulb and humidity-ratio grid. Increase n for smoother category boundaries and decrease it for faster exploratory builds.

Value

A list of ggplot additions.

Examples

```
ggpsychro(tdb_lim = c(25, 45), hum_lim = c(0, 32)) +
    geom_comfort_heat_index(n = c(55, 35), labels = FALSE)
```

---

geom_comfort_pmv *Draw PMV comfort layers*

Description

geom_comfort_pmv() draws filled PMV bands, PMV contour lines and labels, plus optional PMV-based standard zones.

Usage

```
geom_comfort_pmv(
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
```

```
  model = comfort_model_pmv(),
  standard = NULL,
  bands = TRUE,
  contours = TRUE,
  labels = TRUE,
  contour_levels = seq(-3, 3, by = 0.5),
  band_levels = NULL,
  n = NULL,
  band_render = c("band", "tile"),
  band_method = c("auto", "root", "isoband"),
  alpha = NULL,
  na.rm = FALSE,
  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)).
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.
...	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.

model — A comfort model object.

standard — PMV-based standard object.

bands, contours, labels — Single logical values controlling whether geom_comfort_pmv() draws filled PMV bands, PMV contour lines, and text labels. PMV standard zones still draw when standard is supplied.

contour_levels — PMV contour levels for geom_comfort_pmv().

band_levels — Number of PMV filled bands, or a numeric vector of PMV band breaks for geom_comfort_pmv().

n — Grid resolution in dry-bulb and humidity-ratio directions. If NULL, PMV bands use c(360, 220) and PMV curves use 360.

band_render — Band rendering mode. "band" draws filled polygon regions from continuous band boundaries; "tile" draws sampled grid cells directly.

band_method — Advanced PMV boundary construction method used only when band_render = "band". "auto" and "root" use root-traced PMV boundaries; "isoband" uses gridded isobands.

alpha — Layer transparency. PMV standards keep their own defaults unless alpha is supplied.

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().

Details

`n` trades drawing smoothness for build time. For PMV bands, supply one value to use the same dry-bulb and humidity-ratio resolution, or two values for separate directions. PMV contour curves and standard-zone boundaries use the first value because they trace roots along one sampling direction. Smaller values such as `n = c(45, 30)` are useful for exploratory work; larger values produce smoother publication graphics.

`band_render = "band"` draws filled polygon bands from continuous boundaries. `band_render = "tile"` draws sampled grid cells directly. `band_method` is a PMV-specific advanced option for polygon bands: `"root"` traces PMV band boundaries directly, while `"isoband"` builds them from the sampled grid.

Value

A list of ggplot additions.

Examples

```
# Draw PMV comfort bands, contours, and labels.
ggpsychro(tdb_lim = c(15, 35), hum_lim = c(0, 24)) +
    geom_comfort_pmv(n = c(45, 30)) +
    scale_fill_comfort_pmv(name = "PMV")

# Draw labelled PMV contour lines.
ggpsychro(tdb_lim = c(15, 35), hum_lim = c(0, 24)) +
    geom_comfort_pmv(
        bands = FALSE,
        contours = TRUE,
        labels = TRUE,
        contour_levels = c(-1, 0, 1),
        n = 80
    )

# Draw sampled PMV values as grid tiles.
ggpsychro(tdb_lim = c(15, 35), hum_lim = c(0, 24)) +
    geom_comfort_pmv(band_render = "tile", contours = FALSE, n = c(45, 30))

# Draw a PMV-based comfort standard zone.
ggpsychro(tdb_lim = c(15, 35), hum_lim = c(0, 24)) +
    geom_comfort_pmv(
        standard = comfort_pmv_ashrae55(),
        bands = FALSE,
        contours = FALSE,
        n = 80
    )
```

geom_comfort_set *Draw SET comfort layers*

Description

geom_comfort_set() draws Standard Effective Temperature bands and optional contour lines on a psychrometric chart.

Usage

```
geom_comfort_set(
  mapping = NULL,
  data = NULL,
  position = "identity",
  ...,
  model = comfort_model_set(),
  bands = TRUE,
  contours = FALSE,
  labels = FALSE,
  band_levels = NULL,
  contour_levels = NULL,
  n = NULL,
  band_render = c("band", "tile"),
  alpha = 0.55,
  na.rm = FALSE,
  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)).
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.

`...` 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.

`model` A SET comfort model object.

`bands, contours, labels`
Single logical values controlling whether to draw filled SET bands, SET contour lines, and contour text labels.

`band_levels` Number of filled SET bands, or a numeric vector of SET band breaks. Used only for `band_render = "band"`.

`contour_levels` SET contour break values.

`n` Grid resolution in dry-bulb and humidity-ratio directions. If `NULL`, SET bands use `c(80, 50)`.

`band_render` Band rendering mode. `"band"` draws filled polygon regions from continuous band boundaries; `"tile"` draws sampled grid cells directly.

`alpha` Layer transparency.

`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().

Details

n trades drawing smoothness for build time. band_render = "band" draws filled SET regions from gridded isobands, while band_render = "tile" draws sampled grid cells directly.

Value

A list of ggplot additions.

Examples

```
ggpsychro(tdb_lim = c(15, 35), hum_lim = c(0, 24)) +
    geom_comfort_set(n = c(45, 30))

ggpsychro(tdb_lim = c(15, 35), hum_lim = c(0, 24)) +
    geom_comfort_set(
        contours = TRUE,
        labels = TRUE,
        contour_levels = seq(20, 35, 5)
    )
```

---

geom_psychro_grid_relhum

Add psychrometric grid lines

Description

These helpers provide ggplot-style controls for psychrometric reference grids. They do not add data layers; they mark a grid as visible and let coord_psychro() render it in the panel background.

Usage

```
geom_psychro_grid_relhum(
  ...,
  show = TRUE,
  label = TRUE,
  label_loc = 0.95,
  label_parse = FALSE
)
```

```
geom_psychro_grid_wetbulb(
  ...,
  show = TRUE,
  label = TRUE,
  label_loc = 0.1,
  label_parse = TRUE
)

geom_psychro_grid_vappres(
  ...,
  show = TRUE,
  label = TRUE,
  label_loc = 0.5,
  label_parse = FALSE
)

geom_psychro_grid_specvol(
  ...,
  show = TRUE,
  label = TRUE,
  label_loc = 0.95,
  label_parse = TRUE
)

geom_psychro_grid_enthalpy(
  ...,
  show = TRUE,
  label = TRUE,
  label_loc = 0.95,
  label_parse = TRUE
)
```

Arguments

`...`	Line style settings passed to `ggplot2::element_line()`, such as `colour`, `color`, `linewidth`, `size`, and `linetype`. Label style settings can be supplied with `label.*` names to keep them separate from line style settings, such as `label.colour`, `label.color`, `label.size`, `label.alpha`, `label.family`, `label.fontface`, `label.lineheight`, and `label.vjust`.
`show`	A single logical value. If `FALSE`, hide the corresponding grid.
`label`	A single logical value. If `FALSE`, hide labels for the corresponding major grid lines.
`label_loc`	A single number in range `[0, 1]` indicating the label position along each grid line. `NA` hides labels.
`label_parse`	If `TRUE`, labels are parsed as plotmath expressions.

Value

A ggplot addition that controls rendering of a psychrometric grid.

Examples

```
ggpsychro(tdb_lim = c(0, 50), hum_lim = c(0, 30)) +
    geom_psychro_grid_relhum() +
    geom_psychro_grid_wetbulb() +
    geom_psychro_grid_enthalpy()

ggpsychro(tdb_lim = c(0, 50), hum_lim = c(0, 30)) +
    geom_psychro_grid_relhum(color = "black", linewidth = 0.6, label.size = 4) +
    scale_relhum_continuous(
        breaks = seq(25, 75, by = 25),
        minor_breaks = NULL
    ) +
    geom_psychro_grid_wetbulb(color = "black", label = FALSE) +
    scale_wetbulb_continuous(
        breaks = seq(10, 30, by = 10),
        minor_breaks = NULL
    ) +
    geom_psychro_grid_vappres(show = FALSE) +
    geom_psychro_grid_specvol(label_loc = 0.90) +
    geom_psychro_grid_enthalpy(label.size = 4)
```

---

geom_psychro_process *Draw psychrometric state points or process lines*

Description

`stat_psychro_state()` converts a dry-bulb temperature plus exactly one psychrometric state property to chart coordinates. `geom_psychro_process()` uses the same conversion and draws the states as a path.

Usage

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

stat_psychro_state(
```

```
  mapping = NULL,
  data = NULL,
  geom = "point",
  position = "identity",
  ...,
  na.rm = FALSE,
  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)).
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.
...	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.

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().

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.

Details

The tdb aesthetic is required. Exactly one of humratio, relhum, wetbulb, vappres, specvol, or enthalpy must also be mapped.

relhum is supplied in percent. humratio is supplied in chart display units: g/kg in SI and gr/lb in IP.

Value

A ggplot layer.

Examples

```
process <- data.frame(
    state = c("outdoor", "mixed", "supply", "room"),
    tdb = c(18, 23, 28, 31),
    relhum = c(70, 55, 45, 55)
)

# Draw state points. Map exactly one psychrometric property with tdb.
ggpsychro(tdb_lim = c(0, 40), hum_lim = c(0, 25)) +
    stat_psychro_state(
        aes(tdb = tdb, relhum = relhum, colour = state),
        data = process
    )

# Draw the same states as a process line.
ggpsychro(tdb_lim = c(0, 40), hum_lim = c(0, 25)) +
    geom_psychro_process(
        aes(tdb = tdb, relhum = relhum),
        data = process,
        linewidth = 1,
        arrow = grid::arrow(length = grid::unit(0.08, "inches"))
    ) +
    stat_psychro_state(
        aes(tdb = tdb, relhum = relhum),
        data = process,
        size = 2
    )
```

`geom_psychro_protractor` *Add a psychrometric protractor*

Description

`geom_psychro_protractor()` adds a heat-moisture ratio and sensible heat ratio reference protractor to the mask area of a `ggpsychro()` plot. The orientation is determined by the parent chart: regular psychrometric charts place the protractor in the upper-left mask area, while Mollier charts rotate the same protractor geometry into the lower-right mask area. `guide_psychro_protractor()` configures the protractor ticks and labels.

Usage

```
geom_psychro_protractor(
  ...,
  show = TRUE,
```

```
  label = TRUE,
  annotation = TRUE,
  scale = 1,
  radius = 0.24,
  margin = 0.08,
  guide = guide_psychro_protractor()
)

guide_psychro_protractor(
  shr_breaks = waiver(),
  shr_minor_breaks = waiver(),
  shr_labels = waiver(),
  ratio_breaks = waiver(),
  ratio_minor_breaks = waiver(),
  ratio_labels = waiver(),
  check_overlap = TRUE
)
```

Arguments

`...`	Line style settings passed to `ggplot2::element_line()`, such as `colour`, `color`, `linewidth`, `size`, and `linetype`. Label style settings can be supplied with `label.*` names, such as `label.colour`, `label.color`, `label.size`, `label.alpha`, `label.family`, and `label.fontface`.
`show`	A single logical value. If `FALSE`, hide the protractor.
`label`	A single logical value. If `FALSE`, hide protractor labels.
`annotation`	A single logical value, character vector, or expression vector. If `FALSE`, hide the helper formula text. Character and expression vectors must have length 2 and are used as the helper formula labels.
`scale`	A single positive number for overall protractor scaling. It multiplies `radius`, protractor line width, and label text size; `margin` is not scaled.
`radius`	A single number in panel coordinates controlling the protractor radius.
`margin`	A single number or length-2 numeric vector in panel coordinates controlling the distance from the mask-area edge. When length 2, the values are horizontal and vertical margins, respectively.
`guide`	A protractor guide created by `guide_psychro_protractor()`.
`shr_breaks`, `shr_minor_breaks`	Numeric vectors controlling the labelled and unlabelled ticks for the sensible heat ratio axis. Use `NULL` to hide that tick tier.
`shr_labels`, `ratio_labels`	A character vector, expression vector, function, `NULL`, or `waiver()` controlling labels for the corresponding major breaks. `waiver()` uses the default numeric labels, and `NULL` hides the labels.
`ratio_breaks`, `ratio_minor_breaks`	Numeric vectors controlling the labelled and unlabelled ticks for the heat-moisture-ratio axis (`dh/dW`). Use `waiver()` for the ASHRAE-style defaults, or `NULL` to hide that tick tier.

check_overlap A single logical value. If TRUE, overlapping protractor labels are dropped by the grid text grob.

Value

A ggplot addition.

Examples

```
# Draw the default protractor on a psychrometric chart.
ggpsychro(tdb_lim = c(0, 50), hum_lim = c(0, 30)) +
    geom_psychro_protractor()

# Draw the protractor in Mollier orientation.
ggpsychro(tdb_lim = c(0, 50), hum_lim = c(0, 30), mollier = TRUE) +
    geom_psychro_protractor()

# Customize major tick breaks with a guide.
ggpsychro(tdb_lim = c(0, 50), hum_lim = c(0, 30)) +
    geom_psychro_protractor(
        guide = guide_psychro_protractor(
            shr_breaks = seq(0, 1, by = 0.25),
            ratio_breaks = c(0, 2000, 4000)
        )
    )

# Hide the helper formula annotations.
ggpsychro(tdb_lim = c(0, 50), hum_lim = c(0, 30)) +
    geom_psychro_protractor(annotation = FALSE)

# Scale and move the protractor inside the mask area.
ggpsychro(tdb_lim = c(0, 50), hum_lim = c(0, 30)) +
    geom_psychro_protractor(scale = 1.2, margin = c(0.03, 0.10))

# Supply custom labels for selected sensible heat ratio ticks.
ggpsychro(tdb_lim = c(0, 50), hum_lim = c(0, 30)) +
    geom_psychro_protractor(
        guide = guide_psychro_protractor(
            shr_breaks = c(0, 0.5, 1),
            shr_labels = c("0", "half", "1")
        )
    )
```

---

geom_psychro_zone *Draw psychrometric zones*

Description

`stat_psychro_zone()` samples psychrometric boundary curves and returns polygon coordinates. `geom_psychro_zone()` draws those polygons.

Usage

```
geom_psychro_zone(
  mapping = NULL,
  data = NULL,
  stat = "psychro_zone",
  position = "identity",
  ...,
  type = "dbt-rh",
  n = 100L,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

stat_psychro_zone(
  mapping = NULL,
  data = NULL,
  geom = "polygon",
  position = "identity",
  ...,
  type = "dbt-rh",
  n = 100L,
  na.rm = FALSE,
  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)).
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.

`...` 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.

`type` A zone type.

`n` Number of samples used for computed zone boundaries.

`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()`.

`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.

Details

Supported zone types are:

- `"dbt-rh"`: `tdb_min`, `tdb_max`, `relhum_min`, and `relhum_max`
- `"enthalpy-rh"`: `enthalpy_min`, `enthalpy_max`, `relhum_min`, and `relhum_max`
- `"specvol-rh"` or `"volume-rh"`: `specvol_min`, `specvol_max`, `relhum_min`, and `relhum_max`
- `"dbt-wmax"`: `tdb_min`, `tdb_max`, `humratio_max`, and optional `humratio_min`
- `"xy-points"`: `tdb`, `humratio`, and optional `group`

`relhum` values are supplied in percent. `humratio` values are supplied in chart display units: g/kg in SI and gr/lb in IP.

Value

A ggplot layer.

Examples

```
comfort <- data.frame(
    name = c("cool", "warm"),
    tdb_min = c(20, 24),
    tdb_max = c(26, 32),
    relhum_min = c(35, 45),
    relhum_max = c(60, 75)
)

ggpsychro(tdb_lim = c(0, 40), hum_lim = c(0, 25)) +
    geom_psychro_zone(
        aes(tdb_min = tdb_min, tdb_max = tdb_max,
            relhum_min = relhum_min, relhum_max = relhum_max,
            fill = name),
        data = comfort,
        type = "dbt-rh",
        alpha = 0.28,
```

```
        colour = NA
    )

# Dry-bulb range with humidity-ratio limits.
humidity_cap <- data.frame(
    tdb_min = 10,
    tdb_max = 34,
    humratio_min = 4,
    humratio_max = 12
)
ggpsychro(tdb_lim = c(0, 40), hum_lim = c(0, 25)) +
    geom_psychro_zone(
        aes(tdb_min = tdb_min, tdb_max = tdb_max,
            humratio_min = humratio_min, humratio_max = humratio_max),
        data = humidity_cap,
        type = "dbt-wmax",
        alpha = 0.25
    )

# Property-bounded zones.
enthalpy_zone <- data.frame(
    enthalpy_min = 40000,
    enthalpy_max = 65000,
    relhum_min = 30,
    relhum_max = 80
)
ggpsychro(tdb_lim = c(0, 40), hum_lim = c(0, 25)) +
    geom_psychro_zone(
        aes(enthalpy_min = enthalpy_min, enthalpy_max = enthalpy_max,
            relhum_min = relhum_min, relhum_max = relhum_max),
        data = enthalpy_zone,
        type = "enthalpy-rh",
        alpha = 0.25
    )

# Draw an already-specified polygon in chart display units.
polygon_zone <- data.frame(
    tdb = c(20, 26, 28),
    humratio = c(6, 8, 6)
)
ggpsychro(tdb_lim = c(0, 40), hum_lim = c(0, 25)) +
    geom_psychro_zone(
        aes(tdb = tdb, humratio = humratio),
        data = polygon_zone,
        type = "xy-points",
        alpha = 0.25
    )

ggpsychro(tdb_lim = c(0, 40), hum_lim = c(0, 25)) +
    stat_psychro_zone(
        aes(tdb = tdb, humratio = humratio),
        data = polygon_zone,
        type = "xy-points",
```

```
    geom = "polygon",
    alpha = 0.25
)
```

---

ggpsychro *Create a ggpsychro plot*

Description

`ggpsychro()` creates a ggplot object configured for psychrometric charts. It stores chart metadata, installs `coord_psychro()`, applies default axis labels, and adds `theme_psychro()` so psychrometric grids, states, zones, and comfort overlays can be added with the usual ggplot2 + workflow.

Usage

```
ggpsychro(
  data = NULL,
  mapping = aes(),
  tdb_lim = NULL,
  hum_lim = NULL,
  altitude = 0L,
  units = "SI",
  mollier = FALSE
)
```

Arguments

`data`	Default dataset to use for plot. If not already a data.frame, will be converted to one by `ggplot2::fortify()`. If not specified, must be supplied in each layer added to the plot.
`mapping`	Default list of aesthetic mappings to use for plot. If not specified, must be supplied in each layer added to the plot.
`tdb_lim`	A numeric vector of length-2 indicating the dry-bulb temperature limits. Should be in range `[-50, 100]` degrees C [SI] or `[-58, 212]` degrees F [IP]. If `NULL`, trained data ranges will be used when available, otherwise a default display range will be used. Default: `NULL`.
`hum_lim`	A numeric vector of length-2 indicating the humidity ratio limits. Should be in range `[0, 60]` g H2O per kg dry air [SI] or `[0, 420]` grains H2O per lb dry air [IP]. If `NULL`, trained data ranges will be used when available, otherwise a default display range will be used. Default: `NULL`.
`altitude`	A single number of altitude in m [SI] or ft [IP]. Default: `0`.
`units`	A string indicating the system of units chosen. Should be either `"SI"` or `"IP"`.
`mollier`	If `TRUE`, a Mollier chart will be created instead of a psychrometric chart. Default: `FALSE`.

Value

An object of class gg onto which layers, scales, etc. can be added.

Author(s)

Hongyuan Jia

Examples

```
ggpsychro(tdb_lim = c(0, 50), hum_lim = c(0, 50))

ggpsychro(tdb_lim = c(0, 50), hum_lim = c(0, 50), mollier = TRUE)

ggpsychro(tdb_lim = c(0, 50), hum_lim = c(0, 50), units = "IP", altitude = 100)
```

---

is_ggpsychro *Test for ggpsychro plots*

Description

Test for ggpsychro plots

Usage

```
is_ggpsychro(x)
```

Arguments

x	An object to test

Value

A single logical value.

Examples

```
is_ggpsychro(ggpsychro())
is_ggpsychro(ggplot2::ggplot())
```

label_drybulb *Label psychrometric scale breaks*

Description

Format numbers as main variables on the psychrometric chart.

Usage

```
label_drybulb(
  accuracy = NULL,
  scale = 1,
  units,
  big.mark = ",",
  decimal.mark = ".",
  trim = TRUE,
  parse = FALSE,
  ...
)

label_humratio(
  accuracy = NULL,
  scale = 1,
  units,
  big.mark = ",",
  decimal.mark = ".",
  trim = TRUE,
  parse = FALSE,
  ...
)

label_relhum(
  accuracy = NULL,
  scale = 1,
  units,
  big.mark = ",",
  decimal.mark = ".",
  trim = TRUE,
  parse = FALSE,
  ...
)

label_wetbulb(
  accuracy = NULL,
  scale = 1,
  units,
  big.mark = ",",
```

```
  decimal.mark = ".",
  trim = TRUE,
  parse = FALSE,
  ...
)

label_vappres(
  accuracy = NULL,
  scale = 1,
  units,
  big.mark = ",",
  decimal.mark = ".",
  trim = TRUE,
  parse = FALSE,
  ...
)

label_specvol(
  accuracy = NULL,
  scale = 1,
  units,
  big.mark = ",",
  decimal.mark = ".",
  trim = TRUE,
  parse = FALSE,
  ...
)

label_enthalpy(
  accuracy = NULL,
  scale = 1,
  units,
  big.mark = ",",
  decimal.mark = ".",
  trim = TRUE,
  parse = FALSE,
  ...
)
```

Arguments

accuracy	A number to round to. Use (e.g.) `0.01` to show 2 decimal places of precision. If `NULL`, the default, uses a heuristic that should ensure breaks have the minimum number of digits needed to show the difference between adjacent values. Applied to rescaled data.
scale	A scaling factor: `x` will be multiplied by `scale` before formatting. This is useful if the underlying data is very small or very large.
units	A single string indicating the unit system to use. Should be either `"SI"` or `"IP"`

big.mark	Character used between every 3 digits to separate thousands. The default (NULL) retrieves the setting from the number options.
decimal.mark	The character to be used to indicate the numeric decimal point. The default (NULL) retrieves the setting from the number options.
trim	Logical, if FALSE, values are right-justified to a common width (see base::format()).
parse	If TRUE, the labels will be parsed into expressions and displayed as described in ?plotmath. Default: FALSE.
...	Other arguments passed on to base::format().

Value

A labelling function that formats numeric breaks.

Examples

```
demo_scale(10:50, labels = label_drybulb(units = "SI", parse = TRUE))
demo_scale(10:50, labels = label_drybulb(units = "IP", parse = TRUE))

demo_scale(10:20, labels = label_humratio(units = "SI", parse = TRUE))
demo_scale(70:140, labels = label_humratio(units = "IP", parse = TRUE))

demo_scale(seq(0.1, 0.5, by = 0.1), labels = label_relhum(units = "SI"))
demo_scale(seq(0.1, 0.5, by = 0.1), labels = label_relhum(units = "IP"))

demo_scale(10:50, labels = label_wetbulb(units = "SI", parse = TRUE))
demo_scale(10:50, labels = label_wetbulb(units = "IP", parse = TRUE))

demo_scale(10:50, labels = label_specvol(units = "SI", parse = TRUE))
demo_scale(10:50, labels = label_specvol(units = "IP", parse = TRUE))

demo_scale(10:50, labels = label_vappres(units = "SI"))
demo_scale(10:50, labels = label_vappres(units = "IP"))

demo_scale(seq(1000, 2000), labels = label_enthalpy(units = "SI", parse = TRUE))
demo_scale(seq(1000, 2000), labels = label_enthalpy(units = "IP", parse = TRUE))
```

---

psychro_preset *Apply a psychrometric chart preset*

Description

psychro_preset() returns a list of ggplot additions that can be added to a ggpsychro() plot. Presets make selected reference grids explicit, configure grid labels, and apply a matching psychrometric chart theme. The "ashrae" and "minimal" presets are inspired by psychrochart's chart styles.

Usage

```
psychro_preset(name = c("ashrae", "minimal"), labels = TRUE)
```

Arguments

name	A preset name. One of "ashrae" or "minimal".
labels	A single logical value. If FALSE, preset grid labels are hidden while grid visibility and theme styling are kept.

Value

A list of ggplot additions.

Examples

```
ggpsychro(tdb_lim = c(0, 50), hum_lim = c(0, 30)) +
    psychro_preset("ashrae")

ggpsychro(tdb_lim = c(0, 50), hum_lim = c(0, 50)) +
    psychro_preset("minimal", labels = FALSE)
```

---

scale_drybulb_continuous

Psychrometric continuous scales

Description

Psychrometric continuous scales

Usage

```
scale_drybulb_continuous(
  name = waiver(),
  breaks = waiver(),
  minor_breaks = waiver(),
  n.breaks = NULL,
  labels = waiver(),
  limits = NULL,
  expand = waiver(),
  trans = waiver(),
  transform = waiver(),
  guide = waiver(),
  ...
)

scale_humratio_continuous(
```

```
  name = waiver(),
  breaks = waiver(),
  minor_breaks = waiver(),
  n.breaks = NULL,
  labels = waiver(),
  limits = NULL,
  expand = waiver(),
  trans = waiver(),
  transform = waiver(),
  guide = waiver(),
  ...
)

scale_relhum_continuous(
  name = waiver(),
  breaks = waiver(),
  minor_breaks = waiver(),
  n.breaks = NULL,
  labels = waiver(),
  limits = NULL,
  expand = waiver(),
  trans = waiver(),
  transform = waiver(),
  guide = waiver(),
  ...
)

scale_wetbulb_continuous(
  name = waiver(),
  breaks = waiver(),
  minor_breaks = waiver(),
  n.breaks = NULL,
  labels = waiver(),
  limits = NULL,
  expand = waiver(),
  trans = waiver(),
  transform = waiver(),
  guide = waiver(),
  ...
)

scale_vappres_continuous(
  name = waiver(),
  breaks = waiver(),
  minor_breaks = waiver(),
  n.breaks = NULL,
  labels = waiver(),
  limits = NULL,
```

```
  expand = waiver(),
  trans = waiver(),
  transform = waiver(),
  guide = waiver(),
  ...
)

scale_specvol_continuous(
  name = waiver(),
  breaks = waiver(),
  minor_breaks = waiver(),
  n.breaks = NULL,
  labels = waiver(),
  limits = NULL,
  expand = waiver(),
  trans = waiver(),
  transform = waiver(),
  guide = waiver(),
  ...
)

scale_enthalpy_continuous(
  name = waiver(),
  breaks = waiver(),
  minor_breaks = waiver(),
  n.breaks = NULL,
  labels = waiver(),
  limits = NULL,
  expand = waiver(),
  trans = waiver(),
  transform = waiver(),
  guide = waiver(),
  ...
)
```

Arguments

name	The name of the scale. Used as the axis or legend title. If `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.
breaks	One of: • `NULL` for no breaks • `waiver()` for the default breaks computed by the transformation object • A numeric vector of positions • A function that takes the limits as input and returns breaks as output (e.g., a function returned by `scales::extended_breaks()`). Note that for position scales, limits are provided after scale expansion. Also accepts rlang lambda function notation.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`n.breaks` An integer guiding the number of major breaks. The algorithm may choose a slightly different number to ensure nice break labels. Will only have an effect if `breaks = waiver()`. Use `NULL` to use the default number of breaks given by the transformation.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- `NULL` for no labels
- `waiver()` for the default labels computed by the transformation object
- A character vector giving labels (must be same length as `breaks`)
- An expression vector (must be the same length as breaks). See ?plotmath for details.
- A function that takes the breaks as input and returns labels as output. Also accepts rlang lambda function notation.

`limits` One of:

- `NULL` to use the default scale range
- A numeric vector of length two providing limits of the scale. Use `NA` to refer to the existing minimum or maximum
- A function that accepts the existing (automatic) limits and returns new limits. Also accepts rlang lambda function notation. Note that setting limits on positional scales will **remove** data outside of the limits. If the purpose is to zoom, use the limit argument in the coordinate system (see `coord_cartesian()`).

`expand` For position scales, a vector of range expansion constants used to add some padding around the data to ensure that they are placed some distance away from the axes. Use the convenience function `expansion()` to generate the values for the expand argument. The defaults are to expand the scale by 5% on each side for continuous variables, and by 0.6 units on each side for discrete variables.

`trans, transform`

A transformation object or transformer name passed to the underlying ggplot2 continuous scale. The default `ggplot2::waiver()` keeps the psychrometric scale in chart display units.

`guide` A function used to create a guide or its name. See `guides()` for more information.

`...` Other arguments passed on to `scale_(x|y)_continuous()`

Value

A ggplot2 continuous scale object.

Examples

```
# Configure dry-bulb and humidity-ratio axes.
ggpsychro(tdb_lim = c(0, 35), hum_lim = c(0, 25)) +
    scale_drybulb_continuous(breaks = seq(0, 35, by = 5)) +
    scale_humratio_continuous(breaks = seq(0, 25, by = 5))

# Configure relative-humidity and wet-bulb grid breaks.
ggpsychro(tdb_lim = c(0, 35), hum_lim = c(0, 25)) +
    scale_relhum_continuous(n.breaks = 8) +
    scale_wetbulb_continuous(
        breaks = seq(10, 30, by = 5),
        minor_breaks = NULL
    )

# Configure vapor-pressure, specific-volume, and enthalpy grids.
ggpsychro(tdb_lim = c(0, 35), hum_lim = c(0, 25)) +
    scale_vappres_continuous(
        breaks = seq(1000, 4000, by = 1000),
        limits = c(1000, 4500)
    ) +
    scale_specvol_continuous(
        breaks = seq(0.80, 0.95, by = 0.05),
        limits = c(0.80, 0.95)
    ) +
    scale_enthalpy_continuous(
        breaks = seq(20000, 80000, by = 20000),
        limits = c(20000, 80000)
    )
```

`scale_fill_comfort_pmv` *Comfort PMV fill scale*

Description

A diverging blue-white-red fill scale centered on PMV 0.

Usage

```
scale_fill_comfort_pmv(
  ...,
  limits = c(-3, 3),
  low = "#3B5FFF",
  mid = "#F7F7F7",
```

```
  high = "#FF3B30",
  midpoint = 0,
  oob = scales::squish
)
```

Arguments

`...`	Passed to `ggplot2::scale_fill_gradient2()`.
`limits`	Scale limits.
`low`, `mid`, `high`	Endpoint and midpoint colours.
`midpoint`	Scale midpoint.
`oob`	Out-of-bounds handler.

Value

A ggplot2 fill scale.

Examples

```
# Use the default PMV colour scale.
ggpsychro(tdb_lim = c(15, 35), hum_lim = c(0, 24)) +
    geom_comfort_pmv(contours = FALSE, labels = FALSE, n = c(45, 30)) +
    scale_fill_comfort_pmv(name = "PMV")

# Focus the legend on the usual comfort range.
ggpsychro(tdb_lim = c(15, 35), hum_lim = c(0, 24)) +
    geom_comfort_pmv(contours = FALSE, labels = FALSE, n = c(45, 30)) +
    scale_fill_comfort_pmv(limits = c(-1.5, 1.5), name = "PMV")

# Use a custom diverging palette.
ggpsychro(tdb_lim = c(15, 35), hum_lim = c(0, 24)) +
    geom_comfort_pmv(contours = FALSE, labels = FALSE, n = c(45, 30)) +
    scale_fill_comfort_pmv(
        low = "#2166AC",
        mid = "white",
        high = "#B2182B",
        name = "PMV"
    )
```

`stat_comfort_state` *Evaluate comfort metrics at state points*

Description

`stat_comfort_state()` evaluates a comfort model at supplied psychrometric state points and exposes the model outputs through `after_stat()`.

Usage

```
stat_comfort_state(
  mapping = NULL,
  data = NULL,
  geom = "point",
  position = "identity",
  ...,
  model = comfort_model_pmv(),
  na.rm = FALSE,
  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	Geom used to draw evaluated state points.
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.
...	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.

`model` A comfort model object.

`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()`.

Value

A ggplot layer.

Examples

```
states <- data.frame(
    tdb = c(24, 28, 31),
    relhum = c(45, 55, 65)
)

ggpsychro(states, tdb_lim = c(15, 35), hum_lim = c(0, 24)) +
    stat_comfort_state(
        aes(tdb = tdb, relhum = relhum, colour = after_stat(pmv)),
        size = 3
    )
```

`stat_psychro_bin` *Bin data on psychrometric chart coordinates*

Description

`stat_psychro_bin()` bins observations on dry-bulb temperature and humidity ratio coordinates. `geom_psychro_tile()` draws the result as tiles, which is useful for weather-hour distributions and gridded simulation summaries.

Usage

```
stat_psychro_bin(
  mapping = NULL,
  data = NULL,
  geom = "tile",
  position = "identity",
  ...,
  bins = 30,
  binwidth = NULL,
  boundary = c(0, 0),
  drop = TRUE,
  fun = "sum",
  gap = 0.08,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)

geom_psychro_tile(
  mapping = NULL,
  data = NULL,
  stat = "psychro_bin",
  position = "identity",
  ...,
  gap = 0.08,
  boundary = c(0, 0),
  cell.grid = TRUE,
  cell.grid.colour = ggplot2::waiver(),
  cell.grid.linewidth = ggplot2::waiver(),
  cell.grid.linetype = ggplot2::waiver(),
  cell.grid.alpha = ggplot2::waiver(),
  na.rm = FALSE,
  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.
...	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.

`bins` Number of bins in the dry-bulb and humidity-ratio directions. A single number is recycled to both directions. Ignored when `binwidth` is supplied.

`binwidth` Width of bins in chart display units. The first value is in dry-bulb temperature units, and the second value is in humidity-ratio units (`g/kg` for SI and `gr/lb` for IP). A single number is recycled to both directions.

`boundary` Bin boundary in chart display units. The first value is a dry-bulb temperature boundary, and the second value is a humidity-ratio boundary (`g/kg` for SI and `gr/lb` for IP). A single number is recycled to both directions. Only used when `binwidth` is supplied.

`drop` If `TRUE`, the default, omit empty bins.

`fun` Summary function used when the `value` aesthetic is supplied. One of `"sum"`, `"mean"`, `"median"`, `"min"`, or `"max"`.

`gap` Relative gap between adjacent tiles. The default, `0.08`, draws tiles at 92% of the bin width and height. Use `gap = 0` for full-size tiles. Must be a single finite number greater than or equal to 0 and less than 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 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()`.

`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.

`cell.grid` If `TRUE`, the default, draw a tile-local grid across the chart area. The grid uses the current x/y scale major and minor breaks by default. If `binwidth` is finer than, and aligned with, those scale breaks, the cell grid uses the finer bin spacing while preserving the existing x/y breaks as grid lines. If scale breaks are unavailable, the grid falls back to the computed bin spacing.

`cell.grid.colour`, `cell.grid.linewidth`, `cell.grid.linetype`, `cell.grid.alpha`
Appearance of the tile-local cell grid. The default, `ggplot2::waiver()`, inherits from the current `panel.grid.*.x` and `panel.grid.*.y` theme elements. Explicit values override the inherited theme style.

Details

The stat accepts either `x` and `y` aesthetics, where `y` is humidity ratio in chart display units, or `x` and `relhum`, where `relhum` is relative humidity in percent. Relative humidity inputs inherit the plot unit system and pressure from `ggpsychro()`. Tiles default to a small gap and `alpha = 0.85` so psychrometric grid lines remain visible. Tile bodies are clipped to the saturation line in psychrometric coordinates. When `binwidth` is used, each tile represents one dry-bulb and humidity-ratio cell aligned to `boundary`. The optional cell grid follows the chart's x/y breaks so it stays aligned with the visible dry-bulb and humidity-ratio grid. Choose a `binwidth` that evenly subdivides those breaks when a denser cell grid should still coincide with the existing x/y grid.

Value

A ggplot layer.

Computed variables

- `count`: number of observations in each tile.
- `hours`: same as `count`, named for hourly weather data.
- `value`: aggregated `value` aesthetic when supplied.
- `width`, `height`: tile dimensions after applying `gap`.
- `cell_xmin`, `cell_xmax`, `cell_ymin`, `cell_ymax`: full bin boundaries.

Examples

```
d <- data.frame(
    dry_bulb = c(20.1, 20.4, 22.2, 22.5),
    relative_humidity = c(50, 52, 60, 62),
    cooling_load = c(1.2, 1.6, 3.4, 4.2)
)

ggpsychro(d, tdb_lim = c(15, 30), hum_lim = c(0, 20)) +
    geom_psychro_tile(
        aes(dry_bulb, relhum = relative_humidity, fill = after_stat(hours)),
        binwidth = c(2, 2)
    )
```

```
ggpsychro(d, tdb_lim = c(15, 30), hum_lim = c(0, 20)) +
    stat_psychro_bin(
        aes(dry_bulb, relhum = relative_humidity, fill = after_stat(hours)),
        binwidth = c(2, 2)
    )

ggpsychro(d, tdb_lim = c(15, 30), hum_lim = c(0, 20)) +
    geom_psychro_tile(
        aes(dry_bulb, relhum = relative_humidity, value = cooling_load,
            fill = after_stat(value)),
        binwidth = c(2, 2),
        fun = "mean"
    )
```

`stat_relhum` *Draw constant-property psychrometric data*

Description

Draw constant-property psychrometric data

Usage

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

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

stat_vappres(
```

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

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

stat_enthalpy(
  mapping = NULL,
  data = NULL,
  geom = "point",
  position = "identity",
  ...,
  na.rm = FALSE,
  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 over-

ride 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.

... — 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.

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().

Details

These stats convert common psychrometric properties to the humidity-ratio coordinate used by the chart. Use them when the data records dry-bulb temperature plus another property instead of humidity ratio.

Required input aesthetics:

- stat_relhum() uses x and relhum; relhum is relative humidity in percent from 0 to 100.
- stat_wetbulb() uses x and wetbulb; wet-bulb temperature follows the parent chart units.
- stat_vappres() uses x and vappres; vapor pressure is Pa in SI and Psi in IP.
- stat_specvol() uses x and specvol; specific volume is m^3/kg dry air in SI and ft^3/lb dry air in IP.
- stat_enthalpy() uses x and enthalpy; enthalpy is J/kg dry air in SI and Btu/lb dry air in IP.

When used with ggpsychro(), units and pressure are inherited from the plot. If these stats are used directly inside ordinary ggplot2 geoms via stat = "relhum" or similar, supply both units and pres explicitly.

Internally, each stat computes humidity ratio and replaces the layer y aesthetic before the psychrometric coordinate transform is applied.

Value

A ggplot layer.

Examples

```
states <- data.frame(
    tdb = c(18, 22, 26, 30),
    relhum = c(70, 55, 45, 35)
)

ggpsychro(tdb_lim = c(10, 35), hum_lim = c(0, 25)) +
    stat_relhum(aes(x = tdb, relhum = relhum), data = states)

wetbulb_line <- data.frame(tdb = 18:30, wetbulb = 16)
ggpsychro(tdb_lim = c(10, 35), hum_lim = c(0, 25)) +
    geom_psychro_grid_wetbulb() +
    stat_wetbulb(aes(x = tdb, wetbulb = wetbulb),
        data = wetbulb_line, geom = "line")

# The stats can also be used from ordinary ggplot2 geoms.
ggpsychro(tdb_lim = c(10, 35), hum_lim = c(0, 25)) +
    geom_psychro_grid_wetbulb() +
```

```
    geom_line(aes(x = tdb, wetbulb = wetbulb),
        data = wetbulb_line, stat = "wetbulb")

vapour_pressure <- data.frame(
    tdb = c(12, 18, 24, 30),
    vappres = c(900, 1200, 1800, 2400)
)
ggpsychro(tdb_lim = c(10, 35), hum_lim = c(0, 25)) +
    stat_vappres(aes(x = tdb, vappres = vappres),
        data = vapour_pressure)

specific_volume <- data.frame(
    tdb = c(20, 25, 30),
    specvol = c(0.84, 0.86, 0.88)
)
ggpsychro(tdb_lim = c(10, 35), hum_lim = c(0, 25)) +
    stat_specvol(aes(x = tdb, specvol = specvol),
        data = specific_volume)

enthalpy <- data.frame(
    tdb = c(18, 24, 30),
    enthalpy = c(35000, 50000, 65000)
)
ggpsychro(tdb_lim = c(10, 35), hum_lim = c(0, 25)) +
    stat_enthalpy(aes(x = tdb, enthalpy = enthalpy), data = enthalpy)
```

---

theme_psychro *Custom theme for psychrometric chart.*

Description

Custom theme for psychrometric chart.

Usage

```
theme_grey_psychro(
  base_size = 11,
  base_family = "",
  header_family = NULL,
  base_line_size = base_size/22,
  base_rect_size = base_size/22,
  ink = "black",
  paper = "white",
  accent = "#3366FF"
)

theme_gray_psychro(
  base_size = 11,
```

```
  base_family = "",
  header_family = NULL,
  base_line_size = base_size/22,
  base_rect_size = base_size/22,
  ink = "black",
  paper = "white",
  accent = "#3366FF"
)

theme_psychro(
  base_size = 11,
  base_family = "",
  base_line_size = base_size/22,
  base_rect_size = base_size/22
)

theme_psychro_ashrae(
  base_size = 11,
  base_family = "",
  base_line_size = base_size/22,
  base_rect_size = base_size/22
)

theme_psychro_minimal(
  base_size = 11,
  base_family = "",
  base_line_size = base_size/22,
  base_rect_size = base_size/22
)
```

Arguments

base_size	base font size, given in pts.
base_family	base font family
header_family	font family for titles and headers. The default, NULL, uses theme inheritance to set the font. This setting affects axis titles, legend titles, the plot title and tag text.
base_line_size	base size for line elements
base_rect_size	base size for rect elements
ink, paper, accent	colour for foreground, background, and accented elements respectively.

Value

A ggplot2 theme.

theme_psychro_ashrae() and theme_psychro_minimal() are inspired by psychrochart's ASHRAE and minimal chart styles.

Author(s)

Hongyuan Jia

See Also

- psychrochart ASHRAE style
- psychrochart minimal style

Examples

```
ggpsychro(tdb_lim = c(0, 50), hum_lim = c(0, 30)) +
    theme_grey_psychro()

ggpsychro(tdb_lim = c(0, 50), hum_lim = c(0, 30)) +
    theme_gray_psychro()

ggpsychro(tdb_lim = c(0, 50), hum_lim = c(0, 30)) +
    theme_psychro()

ggpsychro(tdb_lim = c(0, 50), hum_lim = c(0, 30)) +
    theme_psychro_ashrae()

ggpsychro(tdb_lim = c(0, 50), hum_lim = c(0, 30)) +
    theme_psychro_minimal()
```

Index