Package ‘ggstudent’

May 11, 2020

Type Package
Date 2020-05-05
Title Continuous Confidence Interval Plots using t-Distribution
Version 0.1.1-1
License GPL (>= 2)
Description Provides an extension to 'ggplot2' (Wickham, 2016, <doi:10.1007/978-3-319-24277-4>) for creating two
types of continuous confidence interval plots (Violin CI and Gradient CI plots), typi-
cally for the sample mean.
These plots contain multiple user-defined confidence areas with varying colours,
defined by the underlying t-distribution used to compute standard confidence intervals for
the mean of the normal distribution when the variance is unknown.
Two types of plots are available, a gradient plot with rectangular areas, and a violin plot where the
shape (horizontal width) is defined by the probability density function of the t-distribution.

Encoding UTF-8
Depends R (>= 3.1.0)
Imports dplyr, ggplot2, stats
Suggests scales
RoxygenNote 6.1.1
NeedsCompilation no
Author Jouni Helske [aut, cre] (<https://orcid.org/0000-0001-7130-793X>)
Maintainer Jouni Helske <jouni.helske@iki.fi>
Repository CRAN
Date/Publication 2020-05-11 11:50:02 UTC

R topics documented:

geom_student ......................................................... 2

Index 4
Description

A Student CI plot (or Violin CI plot) is a mirrored density plot similar to violin plot but instead of kernel density estimate it is based on the density of the t-distribution. It can be though of as a continuous "confidence interval density" (hence the name), which could reduce the dichotomous interpretations due to a fixed confidence level. geom_student can also be used to draw Gradient CI plots (using argument type), which replaces the violin shaped density with a rectangle.

Usage

geom_student(mapping = NULL, data = NULL, position = "identity", width = 0.25, type = "density", scale = TRUE, draw_lines = NULL, draw_mean = TRUE, show.legend = NA, inherit.aes = TRUE, ...)

Arguments

mapping Set of aesthetic mappings created by aes() or aes_.() If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).
position Position adjustment, either as a string, or the result of a call to a position adjustment function.
width Scaling parameter for the width of the violin/rectangle.
type Type of the plot. The default is density which draws violin style density plot, whereas "box" draws a rectangle shaped gradient plot.
scale If "TRUE" (default), violins/rectangles are scaled according to the maximum width of the groups (max(dt(0, df) / se)).
draw_lines If not NULL (default), draw horizontal lines at the given quantiles of the density estimate.
draw_mean If TRUE (default), draw horizontal line at mean.
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.
geom_student  

**Parameters**

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

... Other arguments passed to `layer()`, such as fixed aesthetics.

**Value**

A ggplot object.

**References**


**Examples**

```r
library("dplyr")
library("ggplot2")
library("scales")

ci_levels <- c(0.999, 0.95, 0.9, 0.8, 0.5)
n <- length(ci_levels)

PlantGrowth %>% factor(ci_levels, levels = ci_levels) %>%
  dplyr::group_by(group) %>%
  dplyr::summarise(
    mean = mean(weight),
    df = dplyr::n() - 1,
    se = sd(weight)/sqrt(df + 1)
  ) %>%
  dplyr::full_join(
    data.frame(group = rep(levels(PlantGrowth$group), each = n),
               level = ci_levels), by = "group") -> d

p <- ggplot(data = d, aes(group)) +
  geom_student(aes(mean = mean, se = se, df = df,
                   level = level, fill = level), draw_lines = c(0.95, 0.5))
p
g <- scales::seq_gradient_pal("#e5f5f9", "#2ca25f")
p + scale_fill_manual(values=g(seq(0,1,length = n))) + theme_bw()
p2 <- ggplot(data = d, aes(group)) +
  geom_student(aes(mean = mean, se = se, df = df,
                   level = level, fill = level), type = "box", draw_lines = c(0.95, 0.5))
p2
```
Index

aes(), 2
aes_(), 2
borders(), 3
fortify(), 2
geom_student, 2
ggplot(), 2