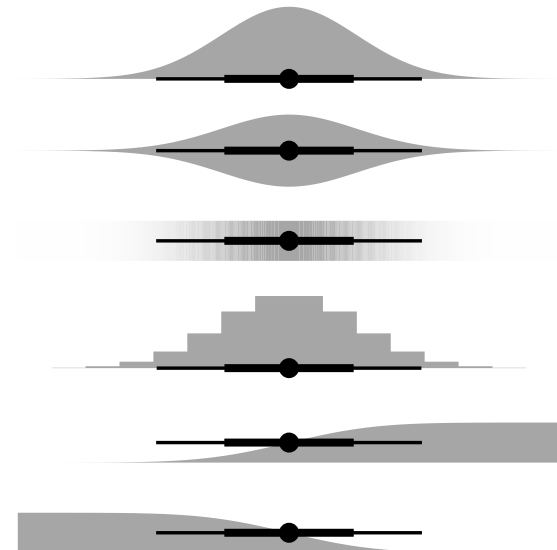


Shortcut slab+interval geometries

The `stat_slabinterval()` stat is a flexible meta-geometry for visualizing **sample data** and **analytical distributions**. With that flexibility comes a cost in remembering particular combinations of parameters that yield specific visualization types. Thus, `ggdist` also provides several **shortcut stats** with sensible default parameters:



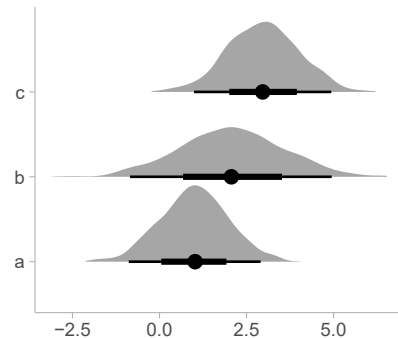
This geometry uses these defaults:				
	mapping = <i>aesthetic mapping</i>	slab_type = <i>function assigned to the computed aesthetic f</i>	side = <i>side to draw the slab on</i>	justification = <i>position of interval relative to slab</i>	normalize = <i>What groups to normalize max height of slab thickness within</i>
<code>stat_slabinterval()</code>	<code>aes(thickness = f)</code>	"pdf"	"topright"	0	"all"
<code>stat_halfeye()</code>	<code>aes(thickness = f)</code>	"pdf"	"topright"	0	"all"
<code>stat_eye()</code>	<code>aes(thickness = f)</code>	"pdf"	"both"	0	"all"
<code>stat_gradientinterval()</code>	<code>aes(slab_alpha = f)</code>	"pdf"	"topright"	0.5	"all"
<code>stat_histinterval()</code>	<code>aes(thickness = f)</code>	"histogram"	"topright"	0	"all"
<code>stat_cdfinterval()</code>	<code>aes(thickness = f)</code>	"cdf"	"topleft"	0.5	"none"
<code>stat_ccdfinterval()</code>	<code>aes(thickness = f)</code>	"ccdf"	"topleft"	0.5	"none"



Example on **sample data**

```
df = data.frame(
  group = c("a", "b", "c"),
  value = rnorm(
    3000,
    mean = c(1, 2, 3),
    sd = c(1, 1.5, 1)
  )
)

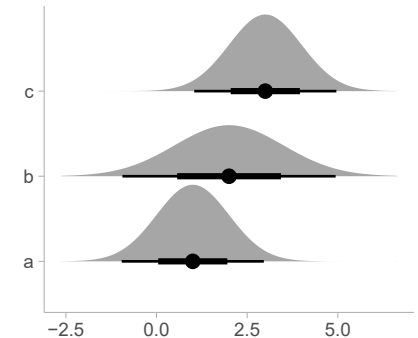
ggplot(df) +
  aes(y = group, x = value) +
  stat_halfeye()
```



Example on **distributional vectors**

```
df = data.frame(
  group = c("a", "b", "c"),
  mean = c(1, 2, 3),
  sd = c(1, 1.5, 1)
)

ggplot(df) +
  aes(
    y = group,
    xdist = dist_normal(mean, sd)
  ) +
  stat_halfeye()
```



Custom slab+interval geometries using computed variables: x/y, CDF, and PDF mappings



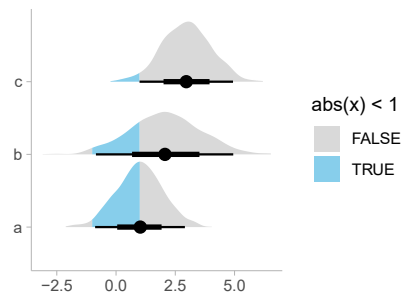
The `stat_slabinterval()` family computes `cdf` and `pdf` variables representing the cumulative distribution function and the probability density function of the underlying data. Along with x/y position, after the stats are computed these can be mapped onto aesthetics like `fill`, `alpha`, or `color`, or combined with functions like `cut_cdf_qi()` to create more esoteric visualization types.

This geometry combined with this mapping does this:	
<code>stat_gradientinterval()</code>	<code>aes(slab_alpha = stat(cdf))</code>	Encodes the CDF using opacity. Can be thought of as many transparent bars overlapping each other to build up a “fuzzy” bar chart. Similar to a <i>fuzzygram</i> , per Wilkinson (Graphical displays, <i>Stat Meth in Med Res</i> , 1992)	
<code>stat_halfeye()</code>	<code>aes(fill = stat(abs(x) < 1.5))</code>	Uses a logical condition to select a fill region of the slab to color differently. Useful for highlighting a <i>region of practical equivalence</i> , or ROPE, per Kruschke (Bayesian estimation supersedes the t test, <i>JEP</i> , 2013)	
<code>stat_gradientinterval()</code>	<code>aes(slab_alpha = stat(-pmax(abs(1 - 2*cdf), .95)))</code>	Fades the tails of the slab outside a desired interval (here 95%) in proportion to $ 1 - 2F(x) $ where $F(x)$ is the CDF. Correll & Gleicher (Error bars considered harmful, <i>TVCG</i> , 2014) argue that this might reduce dichotomous interpretations, though evidence is unclear.	
<code>stat_eye()</code>	<code>aes(slab_alpha = stat(-pmax(abs(1 - 2*cdf), .95)))</code>	Fades the tails of the slab as in the previous example, but combines this with an eye plot, per Helske <i>et al.</i> (Are You Sure You're Sure?, <i>arXiv:2002.07671</i>)	
<code>stat_halfeye()</code>	<code>aes(fill = stat(cut_cdf_qi(cdf, .width = c(.66, .95, 1))))</code>	Bins the CDF into an arbitrary number of intervals (here 66% and 95%) and highlights the intervals using the fill color of the slab. Similar in spirit to <code>bayesplot::mcmc_areas()</code> .	

Example on **sample data**

```
df = data.frame(
  group = c("a", "b", "c"),
  value = rnorm(
    3000,
    mean = c(1, 2, 3),
    sd = c(1, 1.5, 1)
  )
)

ggplot(df) +
  aes(
    y = group,
    x = value,
    fill = stat(abs(x) < 1)
  ) +
  stat_halfeye() +
  scale_fill_manual(values = c("gray85", "skyblue"))
```



Example on **distributional vectors**

```
df = data.frame(
  group = c("a", "b", "c"),
  mean = c(1, 2, 3),
  sd = c(1, 1.5, 1)
)

ggplot(df) +
  aes(
    y = group,
    xdist = dist_normal(mean, sd),
    fill = stat(abs(x) < 1)
  ) +
  stat_halfeye() +
  scale_fill_manual(values = c("gray85", "skyblue"))
```

