seaborn.factorplot

`seaborn.` **`factorplot`** (*x=None, y=None, hue=None, data=None, row=None, col=None, col_wrap=None, estimator=<function mean>, ci=95, n_boot=1000, units=None, order=None, hue_order=None, row_order=None, col_order=None, kind='point', size=4, aspect=1, orient=None, color=None, palette=None, legend=True, legend_out=True, sharex=True, sharey=True, margin_titles=False, facet_kws=None, **kwargs*)

Draw a categorical plot onto a FacetGrid.

The default plot that is shown is a point plot, but other seaborn categorical plots can be chosen with the `kind` parameter, including box plots, violin plots, bar plots, or strip plots.

It is important to choose how variables get mapped to the plot structure such that the most important comparisons are easiest to make. As a general rule, it is easier to compare positions that are closer together, so the `hue` variable should be used for the most important comparisons. For secondary comparisons, try to share the quantitative axis (so, use `col` for vertical plots and `row` for horizontal plots). Note that, although it is possible to make rather complex plots using this function, in many cases you may be better served by created several smaller and more focused plots than by trying to stuff many comparisons into one figure.

After plotting, the **`FacetGrid`** (seaborn.FacetGrid.html#seaborn.FacetGrid) with the plot is returned and can be used directly to tweak supporting plot details or add other layers.

Note that, unlike when using the underlying plotting functions directly, data must be passed in a long-form DataFrame with variables specified by passing strings to `x` , `y` , `hue` , and other parameters.

As in the case with the underlying plot functions, if variables have a `categorical` data type, the correct orientation of the plot elements, the levels of the categorical variables, and their order will be inferred from the objects. Otherwise you may have to use the function parameters (`orient` , `order` , `hue_order` , etc.) to set up the plot correctly.

Parameters:

x, y, hue : names of variables in `data`

> Inputs for plotting long-form data. See examples for interpretation.

data : DataFrame

> Long-form (tidy) dataset for plotting. Each column should correspond to a variable, and each row should correspond to an observation.

row, col : names of variables in `data` , optional

> Categorical variables that will determine the faceting of the grid.

col_wrap : int, optional

> “Wrap” the column variable at this width, so that the column facets span multiple rows. Incompatible with a `row` facet.

estimator : callable that maps vector -> scalar, optional

Statistical function to estimate within each categorical bin.

ci : float or None, optional

Size of confidence intervals to draw around estimated values. If `None` , no bootstrapping will be performed, and error bars will not be drawn.

n_boot : int, optional

Number of bootstrap iterations to use when computing confidence intervals.

units : name of variable in `data` or vector data, optional

Identifier of sampling units, which will be used to perform a multilevel bootstrap and account for repeated measures design.

order, hue_order : lists of strings, optional

Order to plot the categorical levels in, otherwise the levels are inferred from the data objects.

row_order, col_order : lists of strings, optional

Order to organize the rows and/or columns of the grid in, otherwise the orders are inferred from the data objects.

kind : { `point` , `bar` , `count` , `box` , `violin` , `strip` }

The kind of plot to draw.

size : scalar, optional

Height (in inches) of each facet. See also: `aspect` .

aspect : scalar, optional

Aspect ratio of each facet, so that `aspect * size` gives the width of each facet in inches.

orient : “v” | “h”, optional

Orientation of the plot (vertical or horizontal). This is usually inferred from the dtype of the input variables, but can be used to specify when the “categorical” variable is a numeric or when plotting wide-form data.

color : matplotlib color, optional

Color for all of the elements, or seed for **`light_palette()`** (seaborn.light_palette.html#seaborn.light_palette) when using hue nesting.

palette : seaborn color palette or dict, optional

Colors to use for the different levels of the `hue` variable. Should be something that can be interpreted by **`color_palette()`** (seaborn.color_palette.html#seaborn.color_palette), or a dictionary mapping hue levels to matplotlib colors.

legend : bool, optional

If `True` and there is a `hue` variable, draw a legend on the plot.

legend_out : bool, optional

If `True`, the figure size will be extended, and the legend will be drawn outside the plot on the center right.

share{x,y} : bool, optional

If true, the facets will share y axes across columns and/or x axes across rows.

margin_titles : bool, optional

If `True`, the titles for the row variable are drawn to the right of the last column. This option is experimental and may not work in all cases.

facet_kws : dict, optional

Dictionary of other keyword arguments to pass to `FacetGrid` (seaborn.FacetGrid.html#seaborn.FacetGrid).

kwargs : key, value pairings

Other keyword arguments are passed through to the underlying plotting function.

Returns:

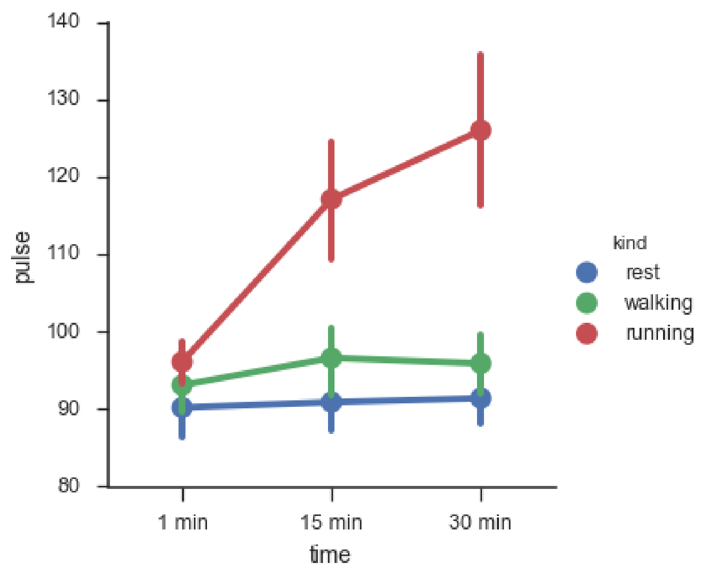
g : `FacetGrid` (seaborn.FacetGrid.html#seaborn.FacetGrid)

Returns the `FacetGrid` (seaborn.FacetGrid.html#seaborn.FacetGrid) object with the plot on it for further tweaking.

Examples

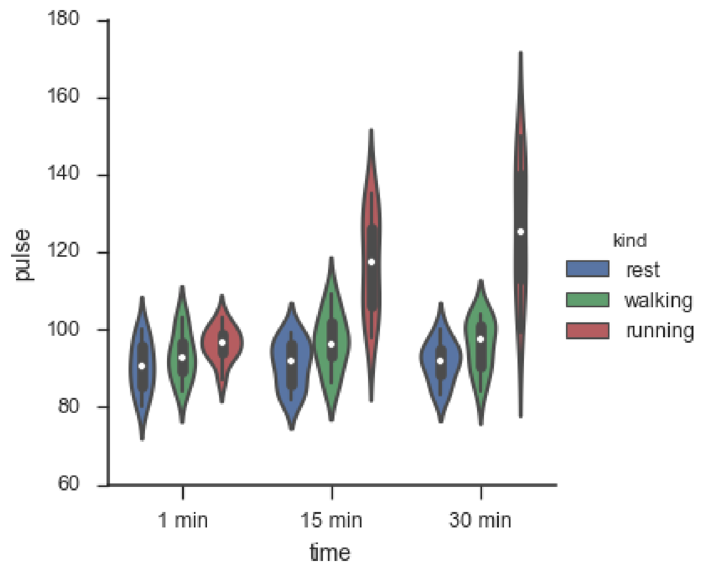
Draw a single facet to use the `FacetGrid` (seaborn.FacetGrid.html#seaborn.FacetGrid) legend placement:

```
>>> import seaborn as sns
>>> sns.set(style="ticks")
>>> exercise = sns.load_dataset("exercise")
>>> g = sns.factorplot(x="time", y="pulse", hue="kind", data=exercise)
```



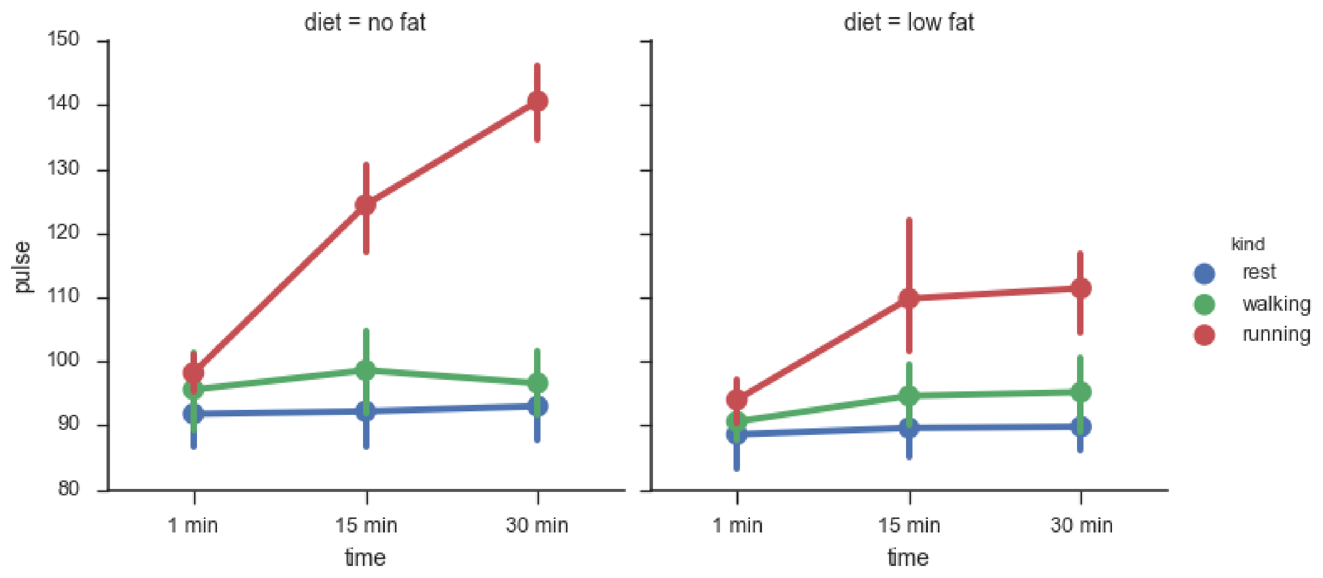
Use a different plot kind to visualize the same data:

```
>>> g = sns.factorplot(x="time", y="pulse", hue="kind",
...                    data=exercise, kind="violin")
```



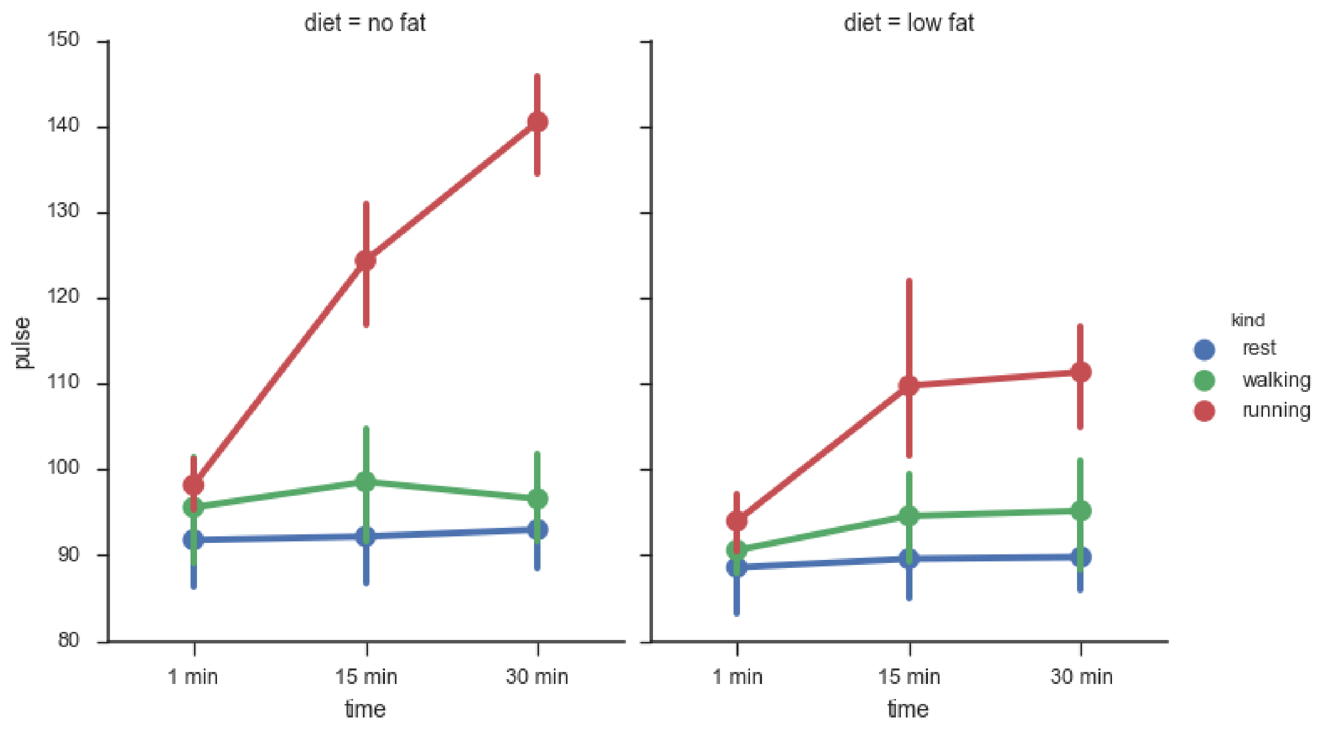
Facet along the columns to show a third categorical variable:

```
>>> g = sns.factorplot(x="time", y="pulse", hue="kind",
...                    col="diet", data=exercise)
```



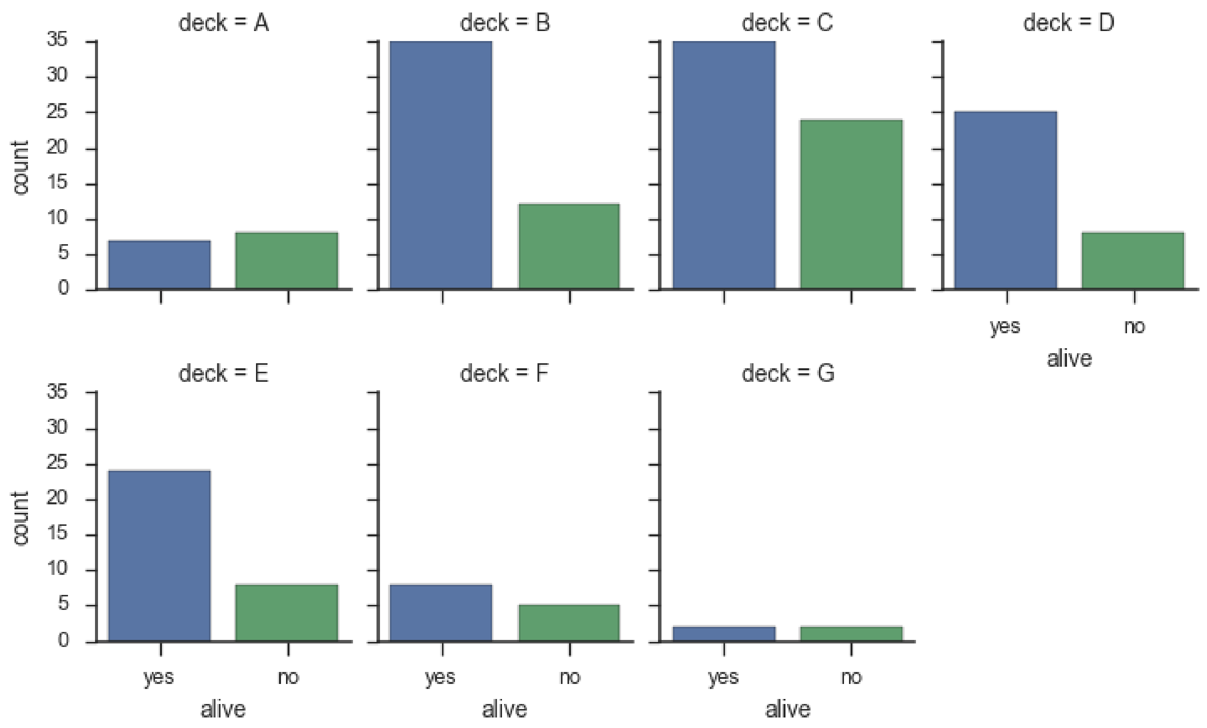
Use a different size and aspect ratio for the facets:

```
>>> g = sns.factorplot(x="time", y="pulse", hue="kind",
...                    col="diet", data=exercise,
...                    size=5, aspect=.8)
```



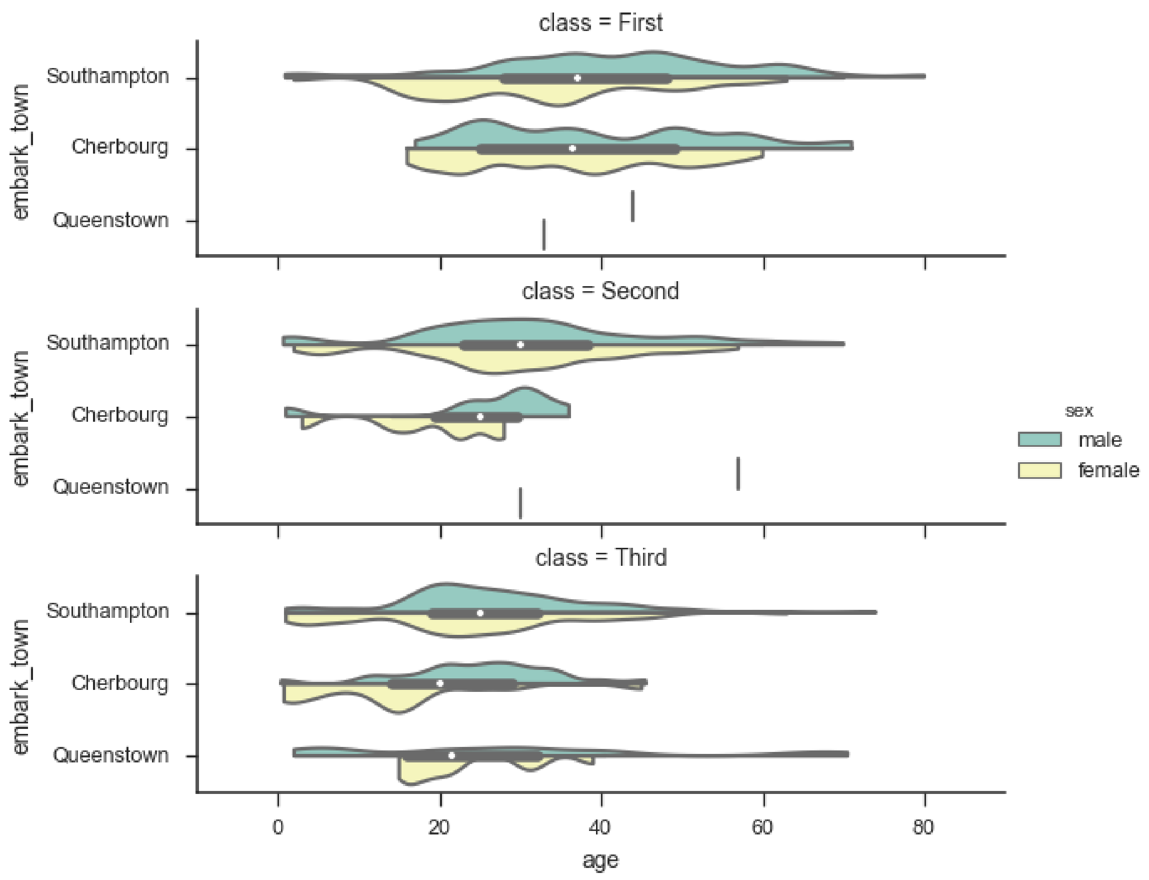
Make many column facets and wrap them into the rows of the grid:

```
>>> titanic = sns.load_dataset("titanic")
>>> g = sns.factorplot("alive", col="deck", col_wrap=4,
...                    data=titanic[titanic.deck.notnull()],
...                    kind="count", size=2.5, aspect=.8)
```



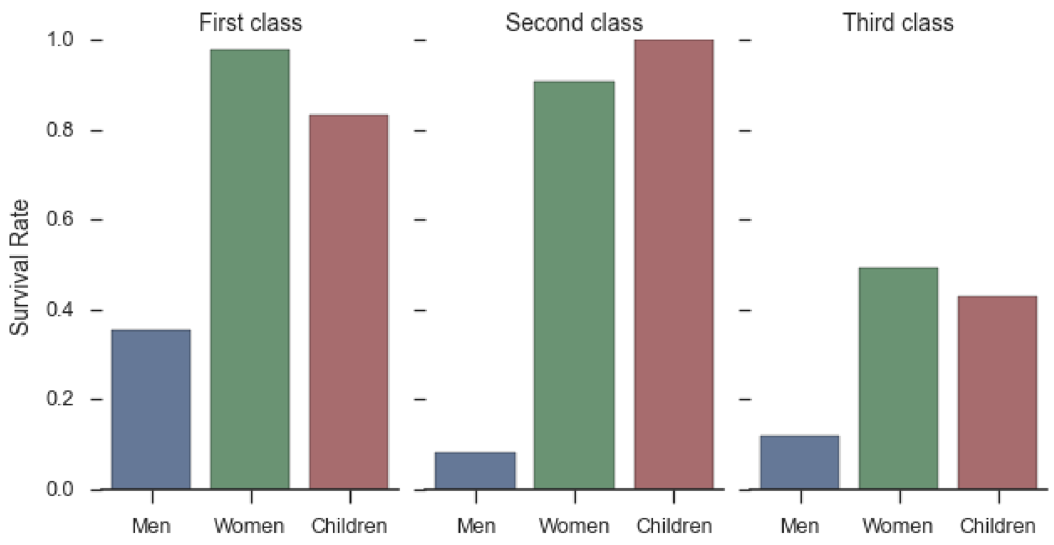
Plot horizontally and pass other keyword arguments to the plot function:

```
>>> g = sns.factorplot(x="age", y="embark_town",
...                    hue="sex", row="class",
...                    data=titanic[titanic.embark_town.notnull()],
...                    orient="h", size=2, aspect=3.5, palette="Set3",
...                    kind="violin", split=True, cut=0, bw=.2)
```



Use methods on the returned **`FacetGrid`** (seaborn.FacetGrid.html#seaborn.FacetGrid) to tweak the presentation:

```
>>> g = sns.factorplot(x="who", y="survived", col="class",
...                    data=titanic, saturation=.5,
...                    kind="bar", ci=None, aspect=.6)
>>> (g.set_axis_labels("", "Survival Rate")
...   .set_xticklabels(["Men", "Women", "Children"])
...   .set_titles("{col_name} {col_var}")
...   .set(ylim=(0, 1))
...   .despine(left=True))
<seaborn.axisgrid.FacetGrid object at 0x...>
```


First class
Second class
Third class
Survival Rate
1.0
0.8
0.6
0.4
0.2
0.0
Men
Women
Children