
statannotations

Release 1.0.0a0

Florian Charlier

Feb 28, 2022

CONTENTS:

- 1 statannotations** **1**
- 1.1 statannotations package 1
- 2 Indices and tables** **9**
- Python Module Index** **11**
- Index** **13**

STATANNOTATIONS

1.1 statannotations package

1.1.1 Subpackages

statannotations.stats package

Submodules

statannotations.stats.ComparisonsCorrection module

```
class statannotations.stats.ComparisonsCorrection.ComparisonsCorrection(method: Union[str, callable], alpha: float = 0.05, name: Optional[str] = None, method_type: Optional[int] = None, statsmodels_api: bool = True, corr_kwargs: Optional[dict] = None)
```

Bases: object

apply(*test_result_list*)

document(*func*)

statannotations.stats.ComparisonsCorrection.**check_valid_correction_name**(*name*)

statannotations.stats.ComparisonsCorrection.**get_correction_parameters**(*name*)

statannotations.stats.ComparisonsCorrection.**get_validated_comparisons_correction**(*comparisons_correction*)

statannotations.stats.StatResult module

```
class statannotations.stats.StatResult.StatResult(test_description, test_short_name, stat_str, stat,  
pval, alpha=0.05)
```

Bases: object

adjust(*stat_summary*)

property **corrected_significance**

property **correction_method**

property **formatted_output**

property **significance_suffix**

statannotations.stats.StatTest module

```
class statannotations.stats.StatTest.StatTest(func: Callable, test_long_name: str, test_short_name:  
str, stat_name: str = 'Stat', alpha: float = 0.05, *args,  
**kwargs)
```

Bases: object

static from_library(*test_name: str*) → *statannotations.stats.StatTest.StatTest*

property **short_name**

```
statannotations.stats.StatTest.wilcoxon(group_data1, group_data2, verbose=1, **stats_params)
```

This function provides the equivalent behavior from earlier versions of statannot/statannotations.

statannotations.stats.test module

```
statannotations.stats.test.apply_test(group_data1, group_data2, test:  
Optional[Union[statannotations.stats.StatTest.StatTest, str]] =  
None, comparisons_correction: Op-  
tional[Union[statannotations.stats.ComparisonsCorrection.ComparisonsCorrection,  
str]] = None, num_comparisons: int = 1, alpha: float = 0.05,  
**stats_params)
```

Get formatted result of two-sample statistical test.

Parameters

- **group_data1** – data
- **group_data2** – data
- **test** – Union[StatTest, str]: Statistical test to run. Either a *StatTest* instance or one of: - *Levene* - *Mann-Whitney* - *Mann-Whitney-gt* - *Mann-Whitney-ls* - *t-test_ind* - *t-test_welch* - *t-test_paired* - *Wilcoxon* - *Kruskal*
- **comparisons_correction** – Union[ComparisonsCorrection, str]: (Default value = None) Method to use for multiple comparisons correction. Either a *ComparisonsCorrection* instance or one of (interfacing statsmodels): - *Bonferroni* - *Holm-Bonferroni* - *Benjamini-Hochberg* - *Benjamini-Yekutieli*
- **num_comparisons** – int: (Default value = 1) Number of comparisons to use for multiple comparisons correction.

- **alpha** – float: (Default value = 0.05) Used for pvalue interpretation in case of comparisons_correction.
- **stats_params** – Additional keyword arguments to pass to the test function

statannotations.stats.utils module

```
statannotations.stats.utils.check_alpha(alpha)
statannotations.stats.utils.check_num_comparisons(num_comparisons)
statannotations.stats.utils.check_pvalues(p_values)
statannotations.stats.utils.get_num_comparisons(p_values, num_comparisons)
statannotations.stats.utils.return_results(results_array)
```

Module contents

1.1.2 Submodules

1.1.3 statannotations.Annotation module

```
class statannotations.Annotation.Annotation(structs, data: Union[str,
statannotations.stats.StatResult.StatResult], formatter:
Optional[statannotations.PValueFormat.Formatter] =
None)
```

Bases: object

Holds data, linked structs and an optional Formatter.

property formatted_output

print_labels_and_content(sep=' vs. ')

property text

1.1.4 statannotations.Annotator module

```
class statannotations.Annotator.Annotator(ax, pairs, plot='boxplot', data=None, x=None, y=None,
hue=None, order=None, hue_order=None, engine='seaborn',
verbose=True, **plot_params)
```

Bases: object

Optionally computes statistical test between pairs of data series, and add statistical annotation on top of the groups (boxes, bars...). The same exact arguments provided to the seaborn plotting function must be passed to the constructor.

This Annotator works in one of the three following modes:

- Add custom text annotations (*set_custom_annotations*)
- Format pvalues and add them to the plot (*set_pvalues*)
- **Perform a statistical test and then add the results to the plot** (*apply_test*)

property alpha

`annotate(line_offset=None, line_offset_to_group=None)`

Add configured annotations to the plot.

`annotate_custom_annotations(text_annot_custom)`

Parameters `text_annot_custom` – List of strings to annotate for each *pair*

`apply_and_annotate()`

Applies a configured statistical test and annotates the plot

`apply_test(num_comparisons='auto', **stats_params)`

Parameters

- `stats_params` – Parameters for statistical test functions.
- `num_comparisons` – Override number of comparisons otherwise calculated with number of pairs

property `comparisons_correction`

`configure(**parameters)`

- *alpha*: Acceptable type 1 error for statistical tests, default 0.05
- *color*
- ***comparisons_correction*: Method for multiple comparisons correction.** One of *statsmodels* *multipletests* methods (w/ default FWER), or a *ComparisonsCorrection* instance.
- ***correction_format*: How to format the star notation on the plot when the multiple comparisons correction method removes the significance** * *default*: a ‘(ns)’ suffix is added, such as in printed output,
corresponds to “{star} ({suffix})”
 - ***replace*: the original star value is replaced with ‘ns’** corresponds to “{suffix}”
 - **a custom formatting string using “{star}” for the original pvalue and ‘{suffix}’ for ‘ns’**
- *line_height*: in axes fraction coordinates
- *line_offset*
- *line_offset_to_group*
- *line_width*
- *loc*
- ***pvalue_format*: list of lists, or tuples. Default values are:**
 - For “star” text_format: `[[1e-4, “***”], [1e-3, “**”], [1e-2, “*”], [0.05, “.”], [1, “ns”]]`.
 - For “simple” text_format `[[1e-5, “1e-5”], [1e-4, “1e-4”], [1e-3, “0.001”], [1e-2, “0.01”], [5e-2, “0.05”]]`
- ***show_test_name*: Set to False to not show the (short) name of test**
- *test*
- *text_offset*: in points
- *test_short_name*

- *use_fixed_offset*
- *verbose*

property `fig`

`get_annotations_text()`

`get_configuration()`

static `get_empty_annotator()`

This instance will have to be initialized with `new_plot()` before being used. This behavior can be useful to create an `Annotator` before using it in a `FacetGrid` mapping.

classmethod `get_offset_func(position)`

`has_type0_comparisons_correction()`

property `loc`

`new_plot(ax, pairs=None, plot='boxplot', data=None, x=None, y=None, hue=None, order=None, hue_order=None, engine: str = 'seaborn', **plot_params)`

property `orient`

static `plot_and_annotate(plot: str, pairs: list, plot_params: dict, configuration: dict, annotation_func: str, annotation_params: Optional[dict] = None, ax_op_before: Optional[List[Union[str, list, None, dict]]] = None, ax_op_after: Optional[List[Union[str, list, None, dict]]] = None, annotate_params: Optional[dict] = None)`

Plots using seaborn and annotates in a single call.

Parameters

- **plot** – seaborn plotting function to call
- **pairs** – pairs to compare (see `Annotator`)
- **plot_params** – parameters for plotting function call
- **configuration** – parameters for `Annotator.configure`
- **annotation_func** – name of annotation function to be called, from: *
'set_custom_annotations' * 'set_pvalues' * 'apply_test'
- **annotation_params** – parameters for the annotation function
- **ax_op_before** – list of [func_name, args, kwargs] to apply on `ax` before annotating
- **ax_op_after** – list of [func_name, args, kwargs] to apply on `ax` after annotating
- **annotate_params** – parameters for `Annotator.annotate`

`plot_and_annotate_facets(plot: str, plot_params: dict, configuration: dict, annotation_func: str, *args, annotation_params: Optional[dict] = None, ax_op_before: Optional[List[Union[str, list, None, dict]]] = None, ax_op_after: Optional[List[Union[str, list, None, dict]]] = None, annotate_params: Optional[dict] = None, **kwargs)`

Plots using seaborn and annotates in a single call, to be used within a `FacetGrid`. First, initialize the `Annotator` with `Annotator(None, pairs)` to define the pairs, then map this function onto the `FacetGrid`.

Parameters

- **plot** – seaborn plotting function to call
- **plot_params** – parameters for plotting function call

- **configuration** – parameters for `Annotator.configure`
- **annotation_func** – name of annotation function to be called, from: *
'set_custom_annotations' * 'set_pvalues' * 'apply_test'
- **annotation_params** – parameters for the annotation function
- **ax_op_before** – list of [func_name, args, kwargs] to apply on *ax* before annotating
- **ax_op_after** – list of [func_name, args, kwargs] to apply on *ax* after annotating
- **annotate_params** – parameters for `Annotator.annotate`
- **args** – additional parameters for the seaborn function
- **kwargs** – additional parameters for the seaborn function

`print_pvalue_legend()`

property `pvalue_format`

`reset_configuration()`

`set_custom_annotations(text_annot_custom)`

Parameters **text_annot_custom** – List of strings to annotate for each *pair*

`set_pvalues(pvalues, num_comparisons='auto')`

Parameters

- **pvalues** – list or array of p-values for each pair comparison.
- **num_comparisons** – Override number of comparisons otherwise calculated with number of pairs

`set_pvalues_and_annotate(pvalues, num_comparisons='auto')`

property `test`

`validate_test_short_name()`

property `verbose`

1.1.5 statannotations.PValueFormat module

class `statannotations.PValueFormat.Formatter`

Bases: `object`

`config(*args, **kwargs)`

`format_data(data)`

class `statannotations.PValueFormat.PValueFormat`

Bases: `statannotations.PValueFormat.Formatter`

`config(**parameters)`

property `correction_format`

`format_data(result)`

`get_configuration()`

`print_legend_if_used()`

statannotations.utils.**check_pairs_in_data**(*pairs: Union[list, tuple], data: Optional[Union[List[list], pandas.core.frame.DataFrame]] = None, coord: Optional[Union[str, list]] = None, hue: Optional[str] = None, hue_order: Optional[List[str]] = None*)

Checks that values referred to in *order* and *pairs* exist in data.

statannotations.utils.**check_valid_text_format**(*text_format*)

statannotations.utils.**empty_dict_if_none**(*data*)

statannotations.utils.**get_closest**(*a_list, value*)

Assumes *myList* is sorted. Returns closest value to *myNumber*. If two numbers are equally close, return the smallest number. from <https://stackoverflow.com/a/12141511/9981846>

statannotations.utils.**get_x_values**(*data, x*) → set

statannotations.utils.**raise_expected_got**(*expected, for_, got, error_type=<class 'ValueError'>*)

Raise a standardized error message.

Raise an *error_type* error with the message Expected *expected* for *for_*; got *got* instead.

Or, if *for_* is *None*, Expected *expected*; got *got* instead.

statannotations.utils.**remove_null**(*series*)

statannotations.utils.**render_collection**(*collection*)

1.1.8 Module contents

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

- statannotations, 8
- statannotations.Annotation, 3
- statannotations.Annotator, 3
- statannotations.format_annotations, 7
- statannotations.PValueFormat, 6
- statannotations.stats, 3
- statannotations.stats.ComparisonsCorrection,
1
- statannotations.stats.StatResult, 2
- statannotations.stats.StatTest, 2
- statannotations.stats.test, 2
- statannotations.stats.utils, 3
- statannotations.utils, 7

A

adjust() (*statannotations.stats.StatResult.StatResult*
method), 2
 alpha (*statannotations.Annotator.Annotator* property), 3
 annotate() (*statannotations.Annotator.Annotator*
method), 3
 annotate_custom_annotations() (*statannota-*
tions.Annotator.Annotator method), 4
 Annotation (class in *statannotations.Annotation*), 3
 Annotator (class in *statannotations.Annotator*), 3
 apply() (*statannotations.stats.ComparisonsCorrection.ComparisonsCorrection*
method), 1
 apply_and_annotate() (*statannota-*
tions.Annotator.Annotator method), 4
 apply_test() (in module *statannotations.stats.test*), 2
 apply_test() (*statannotations.Annotator.Annotator*
method), 4

C

check_alpha() (in module *statannotations.stats.utils*), 3
 check_is_in() (in module *statannotations.utils*), 7
 check_not_none() (in module *statannotations.utils*), 7
 check_num_comparisons() (in module *statannota-*
tions.stats.utils), 3
 check_order_in_data() (in module *statannota-*
tions.utils), 7
 check_pairs_in_data() (in module *statannota-*
tions.utils), 7
 check_pvalues() (in module *statannota-*
tions.stats.utils), 3
 check_valid_correction_name() (in module *stata-*
notations.stats.ComparisonsCorrection), 1
 check_valid_text_format() (in module *statannota-*
tions.utils), 8
 comparisons_correction (*statannota-*
tions.Annotator.Annotator property), 4
 ComparisonsCorrection (class in *statannota-*
tions.stats.ComparisonsCorrection), 1
 config() (*statannotations.PValueFormat.Formatter*
method), 6
 config() (*statannotations.PValueFormat.PValueFormat*
method), 6

configure() (*statannotations.Annotator.Annotator*
method), 4
 corrected_significance (*statannota-*
tions.stats.StatResult.StatResult
 property), 2
 correction_format (*statannota-*
tions.PValueFormat.PValueFormat
 property), 6
 correction_method (*statannota-*
tions.stats.StatResult.StatResult
 property), 2

D

document() (*statannota-*
tions.stats.ComparisonsCorrection.ComparisonsCorrection
method), 1

E

empty_dict_if_none() (in module *statannota-*
tions.utils), 8

F

fig (*statannotations.Annotator.Annotator* property), 5
 format_data() (*statannota-*
tions.PValueFormat.Formatter
 method), 6
 format_data() (*statannota-*
tions.PValueFormat.PValueFormat
 method), 6
 formatted_output (*statannota-*
tions.Annotation.Annotation
 property), 3
 formatted_output (*statannota-*
tions.stats.StatResult.StatResult
 property), 2
 Formatter (class in *statannotations.PValueFormat*), 6
 from_library() (*statannotations.stats.StatTest.StatTest*
 static method), 2

G

get_annotations_text() (*statannota-*
tions.Annotator.Annotator
 method), 5
 get_closest() (in module *statannotations.utils*), 8

get_configuration() (*statannotations.Annotator.Annotator method*), 5
 get_configuration() (*statannotations.PValueFormat.PValueFormat method*), 6
 get_corrected_star() (*in module statannotations.PValueFormat*), 7
 get_correction_parameters() (*in module statannotations.stats.ComparisonsCorrection*), 1
 get_empty_annotator() (*statannotations.Annotator.Annotator static method*), 5
 get_num_comparisons() (*in module statannotations.stats.utils*), 3
 get_offset_func() (*statannotations.Annotator.Annotator class method*), 5
 get_validated_comparisons_correction() (*in module statannotations.stats.ComparisonsCorrection*), 1
 get_x_values() (*in module statannotations.utils*), 8

H

has_type0_comparisons_correction() (*statannotations.Annotator.Annotator method*), 5

I

InvalidParametersError, 7

L

loc (*statannotations.Annotator.Annotator property*), 5

M

module

- statannotations, 8
- statannotations.Annotation, 3
- statannotations.Annotator, 3
- statannotations.format_annotations, 7
- statannotations.PValueFormat, 6
- statannotations.stats, 3
- statannotations.stats.ComparisonsCorrection, 1
- statannotations.stats.StatResult, 2
- statannotations.stats.StatTest, 2
- statannotations.stats.test, 2
- statannotations.stats.utils, 3
- statannotations.utils, 7

N

new_plot() (*statannotations.Annotator.Annotator method*), 5

O

orient (*statannotations.Annotator.Annotator property*), 5

P

plot_and_annotate() (*statannotations.Annotator.Annotator static method*), 5
 plot_and_annotate_facets() (*statannotations.Annotator.Annotator method*), 5
 print_labels_and_content() (*statannotations.Annotation.Annotation method*), 3
 print_legend_if_used() (*statannotations.PValueFormat.PValueFormat method*), 6
 print_pvalue_legend() (*statannotations.Annotator.Annotator method*), 6
 pval_annotation_text() (*in module statannotations.format_annotations*), 7
 pvalue_format (*statannotations.Annotator.Annotator property*), 6
 pvalue_format_string (*statannotations.PValueFormat.PValueFormat property*), 6
 pvalue_thresholds (*statannotations.PValueFormat.PValueFormat property*), 7
 PValueFormat (*class in statannotations.PValueFormat*), 6

R

raise_expected_got() (*in module statannotations.utils*), 8
 remove_null() (*in module statannotations.utils*), 8
 render_collection() (*in module statannotations.utils*), 8
 reset_configuration() (*statannotations.Annotator.Annotator method*), 6
 return_results() (*in module statannotations.stats.utils*), 3

S

set_custom_annotations() (*statannotations.Annotator.Annotator method*), 6
 set_pvalues() (*statannotations.Annotator.Annotator method*), 6
 set_pvalues_and_annotate() (*statannotations.Annotator.Annotator method*), 6
 short_name (*statannotations.stats.StatTest.StatTest property*), 2
 significance_suffix (*statannotations.stats.StatResult.StatResult property*), 2

simple_format_string (*statannotations.PValueFormat.PValueFormat* property),
 7
 simple_text() (*in module statannotations.format_annotations*), 7
 sort_pvalue_thresholds() (*in module statannotations.PValueFormat*), 7
 statannotations
 module, 8
 statannotations.Annotation
 module, 3
 statannotations.Annotator
 module, 3
 statannotations.format_annotations
 module, 7
 statannotations.PValueFormat
 module, 6
 statannotations.stats
 module, 3
 statannotations.stats.ComparisonsCorrection
 module, 1
 statannotations.stats.StatResult
 module, 2
 statannotations.stats.StatTest
 module, 2
 statannotations.stats.test
 module, 2
 statannotations.stats.utils
 module, 3
 statannotations.utils
 module, 7
 StatResult (*class in statannotations.stats.StatResult*), 2
 StatTest (*class in statannotations.stats.StatTest*), 2

T

test (*statannotations.Annotator.Annotator* property), 6
 text (*statannotations.Annotation.Annotation* property), 3
 text_format (*statannotations.PValueFormat.PValueFormat* property),
 7

V

validate_test_short_name() (*statannotations.Annotator.Annotator* method), 6
 verbose (*statannotations.Annotator.Annotator* property), 6

W

wilcoxon() (*in module statannotations.stats.StatTest*), 2