Report for Assignment 1

Project chosen

Name: plotly.py

URL: https://github.com/emelcin/plotly.py

Number of lines of code and the tool used to count it: 384,239 lines (measured using `cloc`)

Programming language: Python

Coverage measurement - Existing tool

Execution: To measure code coverage, the `Coverage.py` tool was used. The tool was executed using the following commands: ```coverage run --branch -m pytest coverage report -m coverage html open htmlcov/index.html

```
plotly/validators/table/cells/fill/__init__.py                    7      2      2      1    67%   5-6
plotly/validators/table/cells/fill/_color.py                      4      0      0      0   100%
plotly/validators/table/cells/line/__init__.py                    9      4      2      1    55%   5-8
plotly/validators/table/cells/line/_color.py                      4      0      0      0   100%
plotly/validators/table/header/__init__.py                       19     14      2      1    29%   5-18
plotly/validators/table/header/_fill.py                           4      0      0      0   100%
plotly/validators/table/header/_line.py                           4      0      0      0   100%
plotly/validators/table/header/fill/__init__.py                   7      2      2      1    67%   5-6
plotly/validators/table/header/fill/_color.py                     4      0      0      0   100%
plotly/validators/table/header/line/__init__.py                   9      4      2      1    55%   5-8
plotly/validators/table/header/line/_color.py                     4      0      0      0   100%
plotly/validators/treemap/__init__.py                            53     48      2      1    11%   5-52
plotly/validators/treemap/_branchvalues.py                        4      0      0      0   100%
plotly/validators/treemap/_domain.py                              4      0      0      0   100%
plotly/validators/treemap/_hovertemplate.py                       4      0      0      0   100%
plotly/validators/treemap/_labels.py                              4      0      0      0   100%
plotly/validators/treemap/_marker.py                              4      0      0      0   100%
plotly/validators/treemap/_maxdepth.py                            4      0      0      0   100%
plotly/validators/treemap/_name.py                                4      0      0      0   100%
plotly/validators/treemap/_parents.py                             4      0      0      0   100%
plotly/validators/treemap/_values.py                              4      0      0      0   100%
plotly/validators/treemap/domain/__init__.py                      9      4      2      1    55%   5-8
plotly/validators/treemap/domain/_x.py                            4      0      0      0   100%
plotly/validators/treemap/domain/_y.py                            4      0      0      0   100%
plotly/validators/treemap/marker/__init__.py                     22     17      2      1    25%   5-21
plotly/validators/treemap/marker/_coloraxis.py                    4      0      0      0   100%
plotly/validators/treemap/marker/_colors.py                       4      0      0      0   100%
plotly/validators/violin/__init__.py                             66     61      2      1     9%   5-65
plotly/validators/violin/_alignmentgroup.py                       4      0      0      0   100%
plotly/validators/violin/_box.py                                  4      0      0      0   100%
plotly/validators/violin/_hovertemplate.py                        4      0      0      0   100%
plotly/validators/violin/_legendgroup.py                          4      0      0      0   100%
plotly/validators/violin/_line.py                                 4      0      0      0   100%
plotly/validators/violin/_marker.py                               4      0      0      0   100%
plotly/validators/violin/_name.py                                 4      0      0      0   100%
plotly/validators/violin/_offsetgroup.py                          4      0      0      0   100%
plotly/validators/violin/_orientation.py                          4      0      0      0   100%
plotly/validators/violin/_points.py                               4      0      0      0   100%
plotly/validators/violin/_scalegroup.py                           4      0      0      0   100%
plotly/validators/violin/_showlegend.py                           4      0      0      0   100%
plotly/validators/violin/_x0.py                                   4      0      0      0   100%
plotly/validators/violin/_x.py                                    4      0      0      0   100%
plotly/validators/violin/_xaxis.py                                4      0      0      0   100%
plotly/validators/violin/_y0.py                                   4      0      0      0   100%
plotly/validators/violin/_y.py                                    4      0      0      0   100%
plotly/validators/violin/_yaxis.py                                4      0      0      0   100%
plotly/validators/violin/box/__init__.py                          9      4      2      1    55%   5-8
plotly/validators/violin/box/_visible.py                          4      0      0      0   100%
plotly/validators/violin/line/__init__.py                         7      2      2      1    67%   5-6
plotly/validators/violin/line/_color.py                           4      0      0      0   100%
plotly/validators/violin/marker/__init__.py                      12      7      2      1    43%   5-11
plotly/validators/violin/marker/_color.py                         4      0      0      0   100%
plotly/validators/violin/marker/_symbol.py                        4      0      0      0   100%
plotly/validators/waterfall/__init__.py                          78     73      2      1     8%   5-77
plotly/validators/waterfall/_y.py                                 4      0      0      0   100%
test_init/__init__.py                                             3      0      0      0   100%
test_init/test_dependencies_not_imported.py                      22     17      2      0    29%   11-36
test_init/test_lazy_imports.py                                   22     17      8      0    23%   12-39
-----------------------------------------------------------------------------------------------------
TOTAL                                                        172042  50098  86071  13487    73%
Wrote HTML report to htmlcov/index.html
(plotly_env_new) emelcinoglu@Emels-MacBook-Air plotly %
```

p

Coverage tool

Emel Dzhinoglu:

Function1: _vectorize_zvalue

Commit made in forked repository that shows the instrumented code:
https://github.com/emelcin/plotly.py/commit/bf954cb231c7fd6065f63d4bdb5340c0411badae

Screenshot of the coverage results output by the instrumentation:

Coverage for **/Users/emelcinoglu/Documents/GitHub/SEP/plotly.py/packages/python/plotly/plotly/express/_imshow.py**: 89%

264 statements | 245 run | 19 missing | 0 excluded | 25 partial

« prev ^ index » next coverage.py v7.5.3, created at 2024-06-18 20:12 +0200

```
1  import plotly.graph_objs as go
2  from _plotly_utils.basevalidators import ColorscaleValidator
3  from ._core import apply_default_cascade, init_figure, configure_animation_controls
4  from .imshow_utils import rescale_intensity, _integer_ranges, _integer_types
5  import pandas as pd
6  import numpy as np
7  import itertools
8  from plotly.utils import image_array_to_data_uri
9
10 try:
11     import xarray
12
13     xarray_imported = True
14 except ImportError:
15     xarray_imported = False
16
17 _float_types = []
18
19 branch_coverage = {
20     "_vectorize_zvalue_none": False,
21     "_vectorize_zvalue_scalar": False,
22     "_vectorize_zvalue_len_1": False,
23     "_vectorize_zvalue_len_3": False,
24     "_vectorize_zvalue_len_4": False,
25     "_vectorize_zvalue_else": False
26 }
27
28 def _vectorize_zvalue(z, mode="max"):
29     alpha = 255 if mode == "max" else 0
30     if z is None:                                                    30 → 31
31         branch_coverage["_vectorize_zvalue_none"] = True
32         return z
33     elif np.isscalar(z):
34         branch_coverage["_vectorize_zvalue_scalar"] = True
35         return [z] * 3 + [alpha]
36     elif len(z) == 1:
37         branch_coverage["_vectorize_zvalue_len_1"] = True
38         return list(z) * 3 + [alpha]
39     elif len(z) == 3:
40         branch_coverage["_vectorize_zvalue_len_3"] = True
41         return list(z) + [alpha]
42     elif len(z) == 4:                                                42 → 46
43         branch_coverage["_vectorize_zvalue_len_4"] = True
44         return z
45     else:
46         branch_coverage["_vectorize_zvalue_else"] = True
47         raise ValueError(
```

Function2: annotation_dict_for_label

Commit made in forked repository that shows the instrumented code:
https://github.com/emelcin/plotly.py/compare/bf954cb231c7fd6065f63d4bdb5340c0411badae...59d83806134e48a387631dd21f0786bfca5c91c7

Screenshot of the coverage results output by the instrumentation:

Coverage for **figure_factory/utils.py**: 62%

102 statements | 62 run | 40 missing | 0 excluded | 6 partial

« prev ^ index » next coverage.py v7.5.3, created at 2024-06-18 21:07 +0200

figure_factory/utils.py: 62% 62 40 0 6

```
192 def annotation_dict_for_label(
193     text,
194     lane,
195     num_of_lanes,
196     subplot_spacing,
197     row_col="col",
198     flipped=True,
199     right_side=True,
200     text_color="#0f0f0f",
201 ):
202     """
203     Returns annotation dict for label of n labels of a 1xn or nx1 subplot.
204 
205     :param (str) text: the text for a label.
206     :param (int) lane: the label number for text. From 1 to n inclusive.
207     :param (int) num_of_lanes: the number 'n' of rows or columns in subplot.
208     :param (float) subplot_spacing: the value for the horizontal_spacing and
209         vertical_spacing params in your plotly.tools.make_subplots() call.
210     :param (str) row_col: choose whether labels are placed along rows or
211         columns.
212     :param (bool) flipped: flips text by 90 degrees. Text is printed
213         horizontally if set to True and row_col='row', or if False and
214         row_col='col'.
215     :param (bool) right_side: only applicable if row_col is set to 'row'.
216     :param (str) text_color: color of the text.
217     """
218     l = (1 - (num_of_lanes - 1) * subplot_spacing) / (num_of_lanes)
219     if not flipped:                                                    219 → 233
220         xanchor = "center"
221         yanchor = "middle"
222         if row_col == "col":                                           222 → 227
223             branch_coverage_annotation["not_flipped_col"] = True
224             x = (lane - 1) * (l + subplot_spacing) + 0.5 * l
225             y = 1.03
226             textangle = 0
227         elif row_col == "row":
228             branch_coverage_annotation["not_flipped_row"] = True
229             y = (lane - 1) * (l + subplot_spacing) + 0.5 * l
230             x = 1.03
231             textangle = 90
232     else:
233         if row_col == "col":
234             branch_coverage_annotation["flipped_col"] = True
235             xanchor = "center"
236             yanchor = "bottom"
237             x = (lane - 1) * (l + subplot_spacing) + 0.5 * l
238             y = 1.0
239             textangle = 270
240         elif row_col == "row":
241             yanchor = "middle"
242             y = (lane - 1) * (l + subplot_spacing) + 0.5 * l
243             if right_side:
244                 branch_coverage_annotation["flipped_row_right_side"] = True
245                 x = 1.0
246                 xanchor = "left"
247             else:
248                 branch_coverage_annotation["flipped_row_left_side"] = True
249                 x = -0.01
250                 xanchor = "right"
251             textangle = 0
252 
253     annotation_dict = dict(
254         textangle=textangle,
255         xanchor=xanchor,
256         yanchor=yanchor,
257         x=x,
258         y=y,
259         showarrow=False,
```

Poyraz Temiz

Function1: is_source_key(key) from *chart_studio\utils.py*

Commit made in forked repository that shows the instrumented code:
https://github.com/plotly/plotly.py/commit/399c3d4f620d0267b7a69385124edd443f169528

Screenshot of the coverage results output by the instrumentation:

chart_studio\utils.py: 77% 65 14 0 5

```
### source key
def is_source_key(key):
    src_regex = re.compile(r".+src$")
    if src_regex.match(key) is not None:
        return True
    else:
        return False
```

chart_studio\utils.py: 83% 76 10 0 5

```
### source key
def is_source_key(key):
    src_regex = re.compile(r".+src$")
    if src_regex.match(key) is not None:
        branch_coverage["is_source_key_match"] = True
        return True
    else:
        branch_coverage["is_source_key_no_match"] = True
        return False

def print_coverage_to_file_source_key(filepath='coverage1.txt'):
    with open(filepath, 'w') as file:
        for branch, hit in branch_coverage.items():
            file.write(f"{branch}: {'hit' if hit else 'not hit'}\n")
```

Function2: update_session_plot_options(**kwargs) from *chart_studio\session.py*

Commit made in forked repository that shows the instrumented code:
https://github.com/plotly/plotly.py/commit/d8bd4b6b581b478881e0151d7209c77f29a861ff

Screenshot of the coverage results output by the instrumentation:

chart_studio\session.py: 84% 37 6 0 6

```
def update_session_plot_options(**kwargs):
    """
    Update the _session plot_options

    :param (str|optional) filename: What the file will be named in Plotly
    :param (str|optional) fileopt: 'overwrite', 'append', 'new', or 'extend'
    :param (bool|optional) world_readable: Make public or private.
    :param (dict|optional) sharing: 'public', 'private', 'secret'
    :param (bool|optional) auto_open: For `plot`, open in new browser tab?
    :param (bool|optional) validate: Error locally if data doesn't pass?

    """
    # raise exception if key is invalid or value is the wrong type
    for key in kwargs:
        if key not in PLOT_OPTIONS:
            raise _plotly_utils.exceptions.PlotlyError(
                "{} is not a valid config or plot option key".format(key)
            )
        if not isinstance(kwargs[key], PLOT_OPTIONS[key]):
            raise _plotly_utils.exceptions.PlotlyError(
                "{} must be of type '{}'".format(key, PLOT_OPTIONS[key])
            )

        # raise exception if sharing is invalid
        if key == "sharing" and not (kwargs[key] in SHARING_OPTIONS):
            raise _plotly_utils.exceptions.PlotlyError(
                "'{0}' must be of either '{1}', '{2}'"
                " or '{3}'".format(key, *SHARING_OPTIONS)
            )

    # update local _session dict with new plot options
    _session["plot_options"].update(kwargs)
```

chart_studio\session.py: 91% 50 4 0 4

```python
def update_session_plot_options(**kwargs):
    """
    Update the _session plot_options

    :param (str|optional) filename: What the file will be named in Plotly
    :param (str|optional) fileopt: 'overwrite', 'append', 'new', or 'extend'
    :param (bool|optional) world_readable: Make public or private.
    :param (dict|optional) sharing: 'public', 'private', 'secret'
    :param (bool|optional) auto_open: For `plot`, open in new browser tab?
    :param (bool|optional) validate: Error locally if data doesn't pass?

    """
    # raise exception if key is invalid or value is the wrong type
    for key in kwargs:
        if key not in PLOT_OPTIONS:
            branch_coverage["update_session_plot_options_invalid_key"] = True
            raise _plotly_utils.exceptions.PlotlyError(
                "{} is not a valid config or plot option key".format(key)
            )
        else:
            branch_coverage["update_session_plot_options_invalid_key_else"] = True

        if not isinstance(kwargs[key], PLOT_OPTIONS[key]):
            branch_coverage["update_session_plot_options_wrong_type"] = True
            raise _plotly_utils.exceptions.PlotlyError(
                "{} must be of type '{}'".format(key, PLOT_OPTIONS[key])
            )
        else:
            branch_coverage["update_session_plot_options_wrong_type_else"] = True

        # raise exception if sharing is invalid
        if key == "sharing" and not (kwargs[key] in SHARING_OPTIONS):
            branch_coverage["update_session_plot_options_invalid_sharing"] = True
            raise _plotly_utils.exceptions.PlotlyError(
                "'{0}' must be of either '{1}', '{2}'"
                " or '{3}'".format(key, *SHARING_OPTIONS)
            )
        else:
            branch_coverage["update_session_plot_options_invalid_sharing_else"] = True


    # update local _session dict with new plot options
     session["plot_options"].update(kwargs)
```

```python
def print_coverage_to_file_session(filepath='coverage2.txt'):
    with open(filepath, 'w') as file:
        for branch, hit in branch_coverage.items():
            file.write(f"{branch}: {'hit' if hit else 'not hit'}\n")
```

Georgi Ivanov

Function1: should_retry

Commit made in forked repository that shows the instrumented code:
https://github.com/plotly/plotly.py/compare/master...emelcin:plotly.py:georgi

Screenshot of the coverage results output by the tool:

chart_studio\api\v2\utils.py: 92% 80 3 0 6

```
7
8
9  def should_retry(exception):
0      if isinstance(exception, exceptions.PlotlyRequestError):
1          if isinstance(exception.status_code, int) and (
2              500 <= exception.status_code < 600 or exception.status_code == 429
3          ):
4              # Retry on 5XX and 429 (image export throttling) errors.
5              return True
6          elif "Uh oh, an error occurred" in exception.message:
7              return True
8
9      return False
0
```

Function2: sign_in()

Commit made in forked repository that shows the instrumented code:
https://github.com/plotly/plotly.py/compare/master...emelcin:plotly.py:georgi

Screenshot of the coverage results output by the tool:

chart_studio\plotly\chunked_requests__init__.py	1	0	0	0	100%
chart_studio\plotly\chunked_requests\chunked_request.py	163	143	50	0	9%
chart_studio\plotly\plotly.py	673	269	320	42	57%
chart_studio\presentation_objs__init__.py	1	0	0	0	100%
chart_studio\presentation_objs\presentation_objs.py	481	191	213	25	59%
chart_studio\session.py	43	6	30	6	84%

Selin Lal Saracoglu
Function1: _list_repr_elided

Commit made in forked repository that shows the instrumented code:
https://github.com/plotly/plotly.py/commit/a4fc88fd3929e66da8687bfbb760129c8f57be2e

Screenshot of the coverage results output by the instrumentation:

plotly/utils.py: 56% 62 40 0 6

```
8  branch_coverage = {
9      "is_list": False,
10     "is_tuple": False,
11     "invalid_type": False,
12     "len_lower_than_threshold": False,
13     "len_greater_than_threshold": False
14 }
15
16
17 # Pretty printing
18 def _list_repr_elided(v, threshold=200, edgeitems=3, indent=0, width=80):
19     """
20     Return a string representation for of a list where list is elided if
21     it has more than n elements
22
23     Parameters
24     ----------
25     v : list
26         Input list
27     threshold :
28         Maximum number of elements to display
29
30     Returns
31     -------
32     str
33     """
34     if isinstance(v, list):                                                    34 → 37
35         branch_coverage["is_list"] = True
36         open_char, close_char = "[", "]"
37     elif isinstance(v, tuple):
38         branch_coverage["is_tuple"] = True
39         open_char, close_char = "(", ")"
40     else:
41         branch_coverage["invalid_type"] = True
42         raise ValueError("Invalid value of type: %s" % type(v))
43
44     if len(v) <= threshold:
45         branch_coverage["len_lower_than_threshold"] = True
46         disp_v = v
47     else:
48         branch_coverage["len_greater_than_threshold"] = True
49         disp_v = list(v[:edgeitems]) + ["..."] + list(v[-edgeitems:])
50
51     v_str = open_char + ", ".join([str(e) for e in disp_v]) + close_char
52
53     v_wrapped = "\n".join(
54         textwrap.wrap(
55             v_str,
56             width=width,
57             initial_indent=" " * (indent + 1),
58             subsequent_indent=" " * (indent + 1),
59         )
60     ).strip()
61     return v_wrapped
```

Function2: _replace_newline

Commit made in forked repository that shows the instrumented code:
https://github.com/plotly/plotly.py/commit/a4fc88fd3929e66da8687bfbb760129c8f57be2e

Screenshot of the coverage results output by the instrumentation:

plotly/tools.py: 35% 90 140 0 8

```
488     try:
489         cls = getattr(graph_objs, obj_type)
490     except (AttributeError, KeyError):
491         raise exceptions.PlotlyError(
492             "'{}' is not a recognized graph_obj.".format(obj_type)
493         )
494     return cls(obj)
495
496
497 branch_coverage = {
498     "replace_newline_dict": False,
499     "replace_newline_list": False,
500     "replace_newline_str": False,
501     "replace_newline_else": False
502 }
503
504
505 def _replace_newline(obj):
506     """Replaces '\n' with '<br>' for all strings in a collection."""
507     if isinstance(obj, dict):
508         branch_coverage["replace_newline_dict"] = True
509         d = dict()
510         for key, val in list(obj.items()):
511             d[key] = _replace_newline(val)
512         return d
513     elif isinstance(obj, list):
514         branch_coverage["replace_newline_list"] = True
515         l = list()
516         for index, entry in enumerate(obj):
517             l += [_replace_newline(entry)]
518         return l
519     elif isinstance(obj, str):
520         branch_coverage["replace_newline_str"] = True
521         s = obj.replace("\n", "<br>")
522         if s != obj:
523             warnings.warn(
524                 "Looks like you used a newline character: '\\n'.\n\n"
525                 "Plotly uses a subset of HTML escape characters\n"
526                 "to do things like newline (<br>), bold (<b></b>),\n"
527                 "italics (<i></i>), etc. Your newline characters \n"
528                 "have been converted to '<br>' so they will show \n"
529                 "up right on your Plotly figure!"
530             )
531         return s
532     else:
533         branch_coverage["replace_newline_else"] = True
534         return obj  # we return the actual reference... but DON'T mutate.
535
```

Coverage improvement - Individual tests

Emel Dzhinoglu:

Test1: test_vectorize_zvalue.py:

Commit made in forked repository that shows the new test:
https://github.com/emelcin/plotly.py/commit/bf954cb231c7fd6065f63d4bdb5340c0411badae

Old coverage result:

Coverage for **/Users/emelcinoglu/Documents/GitHub/SEP/plotly.py/packages/python/plotly/plotly/express/_imshow.py**: 89%

264 statements | 245 run | 19 missing | 0 excluded | 25 partial

« prev ^ index » next coverage.py v7.5.3, created at 2024-06-18 20:12 +0200

```
1  import plotly.graph_objs as go
2  from _plotly_utils.basevalidators import ColorscaleValidator
3  from .._core import apply_default_cascade, init_figure, configure_animation_controls
4  from .imshow_utils import rescale_intensity, _integer_ranges, _integer_types
5  import pandas as pd
6  import numpy as np
7  import itertools
8  from plotly.utils import image_array_to_data_uri
9
10 try:
11     import xarray
12
13     xarray_imported = True
14 except ImportError:
15     xarray_imported = False
16
17 _float_types = []
18
19 branch_coverage = {
20     "_vectorize_zvalue_none": False,
21     "_vectorize_zvalue_scalar": False,
22     "_vectorize_zvalue_len_1": False,
23     "_vectorize_zvalue_len_3": False,
24     "_vectorize_zvalue_len_4": False,
25     "_vectorize_zvalue_else": False
26 }
27
28 def _vectorize_zvalue(z, mode="max"):
29     alpha = 255 if mode == "max" else 0
30     if z is None:                                                    30->31
31         branch_coverage["_vectorize_zvalue_none"] = True
32         return z
33     elif np.isscalar(z):
34         branch_coverage["_vectorize_zvalue_scalar"] = True
35         return [z] * 3 + [alpha]
36     elif len(z) == 1:
37         branch_coverage["_vectorize_zvalue_len_1"] = True
38         return list(z) * 3 + [alpha]
39     elif len(z) == 3:
40         branch_coverage["_vectorize_zvalue_len_3"] = True
41         return list(z) + [alpha]
42     elif len(z) == 4:                                                42->46
43         branch_coverage["_vectorize_zvalue_len_4"] = True
44         return z
45     else:
46         branch_coverage["_vectorize_zvalue_else"] = True
47         raise ValueError(
```

New coverage result:

Coverage for **/Users/emelcinoglu/Documents/GitHub/SEP/plotly.py/packages/python/plotly/plotly/express/_imshow.py**: 92%

264 statements | 252 run | 12 missing | 0 excluded | 23 partial

« prev ^ index » next coverage.py v7.5.3, created at 2024-06-18 20:19 +0200

```
1  import plotly.graph_objs as go
2  from _plotly_utils.basevalidators import ColorscaleValidator
3  from .._core import apply_default_cascade, init_figure, configure_animation_controls
4  from .imshow_utils import rescale_intensity, _integer_ranges, _integer_types
5  import pandas as pd
6  import numpy as np
7  import itertools
8  from plotly.utils import image_array_to_data_uri
9
10 try:
11     import xarray
12
13     xarray_imported = True
14 except ImportError:
15     xarray_imported = False
16
17 _float_types = []
18
19 branch_coverage = {
20     "_vectorize_zvalue_none": False,
21     "_vectorize_zvalue_scalar": False,
22     "_vectorize_zvalue_len_1": False,
23     "_vectorize_zvalue_len_3": False,
24     "_vectorize_zvalue_len_4": False,
25     "_vectorize_zvalue_else": False
26 }
27
28 def _vectorize_zvalue(z, mode="max"):
29     alpha = 255 if mode == "max" else 0
30     if z is None:
31         branch_coverage["_vectorize_zvalue_none"] = True
32         return z
33     elif np.isscalar(z):
34         branch_coverage["_vectorize_zvalue_scalar"] = True
35         return [z] * 3 + [alpha]
36     elif len(z) == 1:
37         branch_coverage["_vectorize_zvalue_len_1"] = True
38         return list(z) * 3 + [alpha]
39     elif len(z) == 3:
40         branch_coverage["_vectorize_zvalue_len_3"] = True
41         return list(z) + [alpha]
42     elif len(z) == 4:
43         branch_coverage["_vectorize_zvalue_len_4"] = True
44         return z
45     else:
46         branch_coverage["_vectorize_zvalue_else"] = True
47         raise ValueError(
48             "zmax can be a scalar, or an iterable of length 1, 3 or 4. "
49             "A value of %s was passed for zmax." % str(z)
50         )
```

The code coverage improved by 3%, from 89% to 92%, due to the addition of new tests in the commit. These tests cover previously untested code paths, increasing the overall test coverage.

Test2: test_annotation_dic_for_label.py:

Commit made in forked repository that shows the new test:
https://github.com/emelcin/plotly.py/compare/bf954cb231c7fd6065f63d4bdb5340c0411badae...59d83806134e48a387631dd21f0786bfca5c91c7

Old coverage result:

Coverage for **figure_factory/utils.py**: 62%

102 statements | 62 run | 40 missing | 0 excluded | 6 partial

« prev ^ index » next coverage.py v7.5.3, created at 2024-06-18 21:07 +0200

figure_factory/utils.py: 62% 62 40 0 6

```
def annotation_dict_for_label(
    text,
    lane,
    num_of_lanes,
    subplot_spacing,
    row_col="col",
    flipped=True,
    right_side=True,
    text_color="#0f0f0f",
);
    """
    Returns annotation dict for label of n labels of a 1xn or nx1 subplot.

    :param (str) text: the text for a label.
    :param (int) lane: the label number for text. From 1 to n inclusive.
    :param (int) num_of_lanes: the number 'n' of rows or columns in subplot.
    :param (float) subplot_spacing: the value for the horizontal_spacing and
        vertical_spacing params in your plotly.tools.make_subplots() call.
    :param (str) row_col: choose whether labels are placed along rows or
        columns.
    :param (bool) flipped: flips text by 90 degrees. Text is printed
        horizontally if set to True and row_col='row', or if False and
        row_col='col'.
    :param (bool) right_side: only applicable if row_col is set to 'row'.
    :param (str) text_color: color of the text.
    """
    l = (1 - (num_of_lanes - 1) * subplot_spacing) / (num_of_lanes)
    if not flipped:                                                          219 → 233
        xanchor = "center"
        yanchor = "middle"
        if row_col == "col":                                                 222 → 227
            branch_coverage_annotation["not_flipped_col"] = True
            x = (lane - 1) * (l + subplot_spacing) + 0.5 * l
            y = 1.03
            textangle = 0
        elif row_col == "row":
            branch_coverage_annotation["not_flipped_row"] = True
            y = (lane - 1) * (l + subplot_spacing) + 0.5 * l
            x = 1.03
            textangle = 90
    else:
        if row_col == "col":
            branch_coverage_annotation["flipped_col"] = True
            xanchor = "center"
            yanchor = "bottom"
            x = (lane - 1) * (l + subplot_spacing) + 0.5 * l
            y = 1.0
            textangle = 270
        elif row_col == "row":
            yanchor = "middle"
            y = (lane - 1) * (l + subplot_spacing) + 0.5 * l
            if right_side:
                branch_coverage_annotation["flipped_row_right_side"] = True
                x = 1.0
                xanchor = "left"
            else:
                branch_coverage_annotation["flipped_row_left_side"] = True
                x = -0.01
                xanchor = "right"
            textangle = 0

    annotation_dict = dict(
        textangle=textangle,
        xanchor=xanchor,
        yanchor=yanchor,
        x=x,
        y=y,
        showarrow=False,
```

New coverage result:

Coverage for **figure_factory/utils.py**: 85%

102 statements | 88 run | 14 missing | 0 excluded | 6 partial

« prev ^ index » next coverage.py v7.5.3, created at 2024-06-18 21:18 +0200

figure_factory/utils.py: 85% 88 14 0 6

```
def annotation_dict_for_label(
    text,
    lane,
    num_of_lanes,
    subplot_spacing,
    row_col="col",
    flipped=True,
    right_side=True,
    text_color="#0f0f0f",
);
    """
    Returns annotation dict for label of n labels of a 1xn or nx1 subplot.

    :param (str) text: the text for a label.
    :param (int) lane: the label number for text. From 1 to n inclusive.
    :param (int) num_of_lanes: the number 'n' of rows or columns in subplot.
    :param (float) subplot_spacing: the value for the horizontal_spacing and
        vertical_spacing params in your plotly.tools.make_subplots() call.
    :param (str) row_col: choose whether labels are placed along rows or
        columns.
    :param (bool) flipped: flips text by 90 degrees. Text is printed
        horizontally if set to True and row_col='row', or if False and
        row_col='col'.
    :param (bool) right_side: only applicable if row_col is set to 'row'.
    :param (str) text_color: color of the text.
    """
    l = (1 - (num_of_lanes - 1) * subplot_spacing) / (num_of_lanes)
    if not flipped:
        xanchor = "center"
        yanchor = "middle"
        if row_col == "col":
            branch_coverage_annotation["not_flipped_col"] = True
            x = (lane - 1) * (l + subplot_spacing) + 0.5 * l
            y = 1.03
            textangle = 0
        elif row_col == "row":                                               227 → 253
            branch_coverage_annotation["not_flipped_row"] = True
            y = (lane - 1) * (l + subplot_spacing) + 0.5 * l
            x = 1.03
            textangle = 90
    else:
        if row_col == "col":
            branch_coverage_annotation["flipped_col"] = True
            xanchor = "center"
            yanchor = "bottom"
            x = (lane - 1) * (l + subplot_spacing) + 0.5 * l
            y = 1.0
            textangle = 270
        elif row_col == "row":                                               240 → 253
            yanchor = "middle"
            y = (lane - 1) * (l + subplot_spacing) + 0.5 * l
            if right_side:
                branch_coverage_annotation["flipped_row_right_side"] = True
                x = 1.0
                xanchor = "left"
            else:
                branch_coverage_annotation["flipped_row_left_side"] = True
                x = -0.01
                xanchor = "right"
            textangle = 0

    annotation_dict = dict(
        textangle=textangle,
        xanchor=xanchor,
        yanchor=yanchor,
        x=x,
        y=y,
```

The code coverage improved by 23%, from 62% to 85%, due to the addition of new tests in the commit. These tests cover previously untested code paths, increasing the overall test coverage.

Poyraz Temiz
Test1: is_source_key(key) from *chart_studio\utils.py*

Commit made in forked repository that shows the new test:
https://github.com/emelcin/plotly.py/commit/1b7215cee9643c37827703bc68d6b27cbb45862d

Old coverage result:

chart_studio\utils.py: 77% | 65 | 14 | 0 | 5

```
### source key
def is_source_key(key):
    src_regex = re.compile(r".+src$")
    if src_regex.match(key) is not None:
        return True
    else:
        return False
```

New coverage result:

chart_studio\utils.py: 82% | 69 | 10 | 0 | 5

```
### source key
def is_source_key(key):
    src_regex = re.compile(r".+src$")
    if src_regex.match(key) is not None:
        return True
    else:
        return False
```

Branch coverage was 0% as this function was not tested before. However, now, it has been increased to 100%.

Test2: update_session_plot_options(**kwargs)from chart_studio\session.py

Commit made in forked repository that shows the new test:
https://github.com/emelcin/plotly.py/commit/1b7215cee9643c37827703bc68d6b27cbb45862d

Old coverage result:

chart_studio\session.py: 84% 37 6 0 6

```
def update_session_plot_options(**kwargs):
    """
    Update the _session plot_options

    :param (str|optional) filename: What the file will be named in Plotly
    :param (str|optional) fileopt: 'overwrite', 'append', 'new', or 'extend'
    :param (bool|optional) world_readable: Make public or private.
    :param (dict|optional) sharing: 'public', 'private', 'secret'
    :param (bool|optional) auto_open: For `plot`, open in new browser tab?
    :param (bool|optional) validate: Error locally if data doesn't pass?

    """
    # raise exception if key is invalid or value is the wrong type
    for key in kwargs:
        if key not in PLOT_OPTIONS:
            raise _plotly_utils.exceptions.PlotlyError(
                "{} is not a valid config or plot option key".format(key)
            )
        if not isinstance(kwargs[key], PLOT_OPTIONS[key]):
            raise _plotly_utils.exceptions.PlotlyError(
                "{} must be of type '{}'".format(key, PLOT_OPTIONS[key])
            )

        # raise exception if sharing is invalid
        if key == "sharing" and not (kwargs[key] in SHARING_OPTIONS):
            raise _plotly_utils.exceptions.PlotlyError(
                "'{0}' must be of either '{1}', '{2}'"
                " or '{3}'".format(key, *SHARING_OPTIONS)
            )

    # update local _session dict with new plot options
    _session["plot_options"].update(kwargs)
```

New coverage result:

chart_studio\session.py: 91% 50 4 0 4

```python
def update_session_plot_options(**kwargs):
    """
    Update the _session plot_options

    :param (str|optional) filename: What the file will be named in Plotly
    :param (str|optional) fileopt: 'overwrite', 'append', 'new', or 'extend'
    :param (bool|optional) world_readable: Make public or private.
    :param (dict|optional) sharing: 'public', 'private', 'secret'
    :param (bool|optional) auto_open: For `plot`, open in new browser tab?
    :param (bool|optional) validate: Error locally if data doesn't pass?

    """
    # raise exception if key is invalid or value is the wrong type
    for key in kwargs:
        if key not in PLOT_OPTIONS:
            branch_coverage["update_session_plot_options_invalid_key"] = True
            raise _plotly_utils.exceptions.PlotlyError(
                "{} is not a valid config or plot option key".format(key)
            )
        else:
            branch_coverage["update_session_plot_options_invalid_key_else"] = True

        if not isinstance(kwargs[key], PLOT_OPTIONS[key]):
            branch_coverage["update_session_plot_options_wrong_type"] = True
            raise _plotly_utils.exceptions.PlotlyError(
                "{} must be of type '{}'".format(key, PLOT_OPTIONS[key])
            )
        else:
            branch_coverage["update_session_plot_options_wrong_type_else"] = True

        # raise exception if sharing is invalid
        if key == "sharing" and not (kwargs[key] in SHARING_OPTIONS):
            branch_coverage["update_session_plot_options_invalid_sharing"] = True
            raise _plotly_utils.exceptions.PlotlyError(
                "'{0}' must be of either '{1}', '{2}'"
                " or '{3}'".format(key, *SHARING_OPTIONS)
            )
        else:
            branch_coverage["update_session_plot_options_invalid_sharing_else"] = True


    # update local _session dict with new plot options
    _session["plot_options"].update(kwargs)
```

Before, as there were 2 if statements that were untested, the branch coverage was around 66% (by counting the invisible else cases as well). However, now, the branch coverage is 100% as all branches, including the invisible else cases, are tested.

Georgi Ivanov:
Test1 should_retry():

Commit made in forked repository that shows the new test:
https://github.com/plotly/plotly.py/compare/master...emelcin:plotly.py:georgi

Old coverage result:

chart_studio\api\v2\utils.py: 92% 80 3 0 6

```
def should_retry(exception):
    if isinstance(exception, exceptions.PlotlyRequestError):
        if isinstance(exception.status_code, int) and (
            500 <= exception.status_code < 600 or exception.status_code == 429
        ):
            # Retry on 5XX and 429 (image export throttling) errors.
            return True
        elif "Uh oh, an error occurred" in exception.message:
            return True

    return False
```

New coverage result:

chart_studio\api\v2\utils.py: 97% 89 1 0 3

```
    return headers

branch_coverage = {
    "f1": False,
    "f2": False,
    "f3": False,
    "f4": False,
    "f5": False,
    "f6": False
}

def should_retry(exception):
    if isinstance(exception, exceptions.PlotlyRequestError):
        branch_coverage["f1"] = True
        if isinstance(exception.status_code, int) and (
            500 <= exception.status_code < 600 or exception.status_code == 429
        ):
            # Retry on 5XX and 429 (image export throttling) errors.
            branch_coverage["f2"] = True
            return True
        elif "Uh oh, an error occurred" in exception.message:
            branch_coverage["f3"] = True
            return True
        else:
            branch_coverage["f4"] = True

    else:
        branch_coverage["f5"] = True

    branch_coverage["f6"] = True

    return False
```

Test2:

Commit made in forked repository that shows the new test:
https://github.com/plotly/plotly.py/compare/master...emelcin:plotly.py:georgi

Old coverage result:

chart_studio\plotly\chunked_requests__init__.py	1	0	0	0	100%
chart_studio\plotly\chunked_requests\chunked_request.py	163	143	50	0	9%
chart_studio\plotly\plotly.py	673	269	320	42	57%
chart_studio\presentation_objs__init__.py	1	0	0	0	100%
chart_studio\presentation_objs\presentation_objs.py	481	191	213	25	59%
chart_studio\session.py	43	6	30	6	84%

New coverage result:

chart_studio\session.py: 92% 51 4 0 3

```
61  def sign_in(username, api_key, **kwargs):
62      """
63      Set set session credentials and config (not saved to file).
64
65      If unspecified, credentials and config are searched for in `.plotly` dir.
66
67      :param (str) username: The username you'd use to sign in to Plotly
68      :param (str) api_key: The api key associated with above username
69      :param (list|optional) stream_ids: Stream tokens for above credentials
70      :param (str|optional) proxy_username: The un associated with with your Proxy
71      :param (str|optional) proxy_password: The pw associated with your Proxy un
72
73      :param (str|optional) plotly_domain:
74      :param (str|optional) plotly_streaming_domain:
75      :param (str|optional) plotly_api_domain:
76      :param (bool|optional) plotly_ssl_verification:
77      :param (bool|optional) plotly_proxy_authorization:
78      :param (bool|optional) world_readable:
79
80      """
81      # TODO: verify these _credentials with plotly
82
83      # kwargs will contain all our info
84      kwargs.update(username=username, api_key=api_key)
85
86      # raise error if key isn't valid anywhere
87      for key in kwargs:
88          if key not in CREDENTIALS_KEYS and key not in CONFIG_KEYS:
89              branch_coverage["f1"] = True
90              raise _plotly_utils.exceptions.PlotlyError(
91                  "{} is not a valid config or credentials key".format(key)
92              )
93
94          else:
95              branch_coverage["f2"] = True
96
97      # add credentials, raise error if type is wrong.
98      for key in CREDENTIALS_KEYS:
99          if key in kwargs:
100             if not isinstance(kwargs[key], CREDENTIALS_KEYS[key]):
101                 branch_coverage["f3"] = True
102                 raise _plotly_utils.exceptions.PlotlyError(
103                     "{} must be of type '{}'".format(key, CREDENTIALS_KEYS[key])
104                 )
105             else:
106                 branch_coverage["f4"] = True
107             _session["credentials"][key] = kwargs[key]
108
109         else:
110             branch_coverage["f5"] = True
```

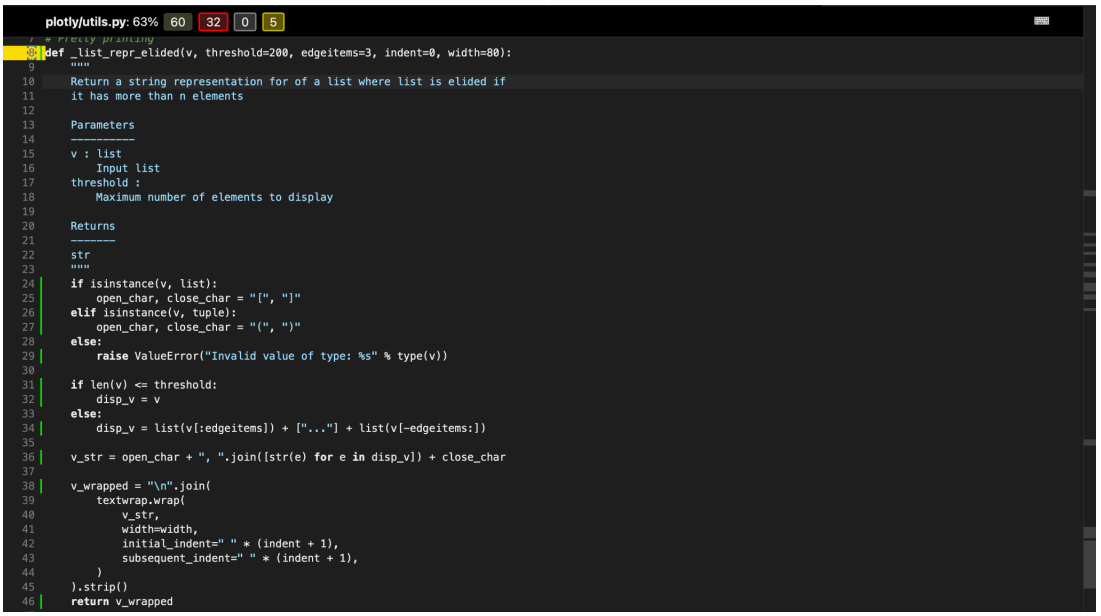
Selin Saracoglu:

Test1: _list_repr_elided

Commit made in forked repository that shows the new test:
https://github.com/plotly/plotly.py/commit/66f7203bbee0b23179cde6c7bf554e4d97f60065

Old coverage result:

New coverage result:



As seen above, the previous branch coverage was 33%, as only the first if statement was covered. After the addition of tests, the branch coverage increased to 100%.

Test2: _replace_new_line

Commit made in forked repository that shows the new test:
https://github.com/plotly/plotly.py/commit/66f7203bbee0b23179cde6c7bf554e4d97f60065

Old coverage result:

plotly/tools.py: 35% 90 140 0 8

```
488     try:
489         cls = getattr(graph_objs, obj_type)
490     except (AttributeError, KeyError):
491         raise exceptions.PlotlyError(
492             "'{}' is not a recognized graph_obj.".format(obj_type)
493         )
494     return cls(obj)
495
496
497 branch_coverage = {
498     "replace_newline_dict": False,
499     "replace_newline_list": False,
500     "replace_newline_str": False,
501     "replace_newline_else": False
502 }
503
504
505 def _replace_newline(obj):
506     """Replaces '\n' with '<br>' for all strings in a collection."""
507     if isinstance(obj, dict):
508         branch_coverage["replace_newline_dict"] = True
509         d = dict()
510         for key, val in list(obj.items()):
511             d[key] = _replace_newline(val)
512         return d
513     elif isinstance(obj, list):
514         branch_coverage["replace_newline_list"] = True
515         l = list()
516         for index, entry in enumerate(obj):
517             l += [_replace_newline(entry)]
518         return l
519     elif isinstance(obj, str):
520         branch_coverage["replace_newline_str"] = True
521         s = obj.replace("\n", "<br>")
522         if s != obj:
523             warnings.warn(
524                 "Looks like you used a newline character: '\\n'.\n\n"
525                 "Plotly uses a subset of HTML escape characters\n"
526                 "to do things like newline (<br>), bold (<b></b>),\n"
527                 "italics (<i></i>), etc. Your newline characters \n"
528                 "have been converted to '<br>' so they will show \n"
529                 "up right on your Plotly figure!"
530             )
531         return s
532     else:
533         branch_coverage["replace_newline_else"] = True
534         return obj  # we return the actual reference... but DON'T mutate.
535
```

New coverage result:

plotly/tools.py: 46% 104 117 0 8

```
490     except (AttributeError, KeyError):
491         raise exceptions.PlotlyError(
492             "'{}' is not a recognized graph_obj.".format(obj_type)
493         )
494     return cls(obj)
495
496
497 def _replace_newline(obj):
498     """Replaces '\n' with '<br>' for all strings in a collection."""
499     if isinstance(obj, dict):
500         d = dict()
501         for key, val in list(obj.items()):
502             d[key] = _replace_newline(val)
503         return d
504     elif isinstance(obj, list):
505         l = list()
506         for index, entry in enumerate(obj):
507             l += [_replace_newline(entry)]
508         return l
509     elif isinstance(obj, str):
510         s = obj.replace("\n", "<br>")
511         if s != obj:
512             warnings.warn(
513                 "Looks like you used a newline character: '\\n'.\n\n"
514                 "Plotly uses a subset of HTML escape characters\n"
515                 "to do things like newline (<br>), bold (<b></b>),\n"
516                 "italics (<i></i>), etc. Your newline characters \n"
517                 "have been converted to '<br>' so they will show \n"
518                 "up right on your Plotly figure!"
519             )
520         return s
521     else:
522         return obj  # we return the actual reference... but DON'T mutate.
523
524
525 def return_figure_from_figure_or_data(figure_or_data, validate_figure):
526     from plotly.graph_objs import Figure
527     from plotly.basedatatypes import BaseFigure
```

As seen above, the old branch coverage was 0% as none of the branches are tested. After the addition of the tests, now the branch coverage is 100%.

Overall

Old coverage result:

```
plotly/validators/table/cells/fill/__init__.py                    7      2      2      1    67%   5-6
plotly/validators/table/cells/fill/_color.py                      4      0      0      0   100%
plotly/validators/table/cells/line/__init__.py                    9      4      2      1    55%   5-8
plotly/validators/table/cells/line/_color.py                      4      0      0      0   100%
plotly/validators/table/header/__init__.py                       19     14      2      1    29%   5-18
plotly/validators/table/header/_fill.py                           4      0      0      0   100%
plotly/validators/table/header/_line.py                           4      0      0      0   100%
plotly/validators/table/header/fill/__init__.py                   7      2      2      1    67%   5-6
plotly/validators/table/header/fill/_color.py                     4      0      0      0   100%
plotly/validators/table/header/line/__init__.py                   9      4      2      1    55%   5-8
plotly/validators/table/header/line/_color.py                     4      0      0      0   100%
plotly/validators/treemap/__init__.py                            53     48      2      1    11%   5-52
plotly/validators/treemap/_branchvalues.py                        4      0      0      0   100%
plotly/validators/treemap/_domain.py                              4      0      0      0   100%
plotly/validators/treemap/_hovertemplate.py                       4      0      0      0   100%
plotly/validators/treemap/_labels.py                              4      0      0      0   100%
plotly/validators/treemap/_marker.py                              4      0      0      0   100%
plotly/validators/treemap/_maxdepth.py                            4      0      0      0   100%
plotly/validators/treemap/_name.py                                4      0      0      0   100%
plotly/validators/treemap/_parents.py                             4      0      0      0   100%
plotly/validators/treemap/_values.py                              4      0      0      0   100%
plotly/validators/treemap/domain/__init__.py                      9      4      2      1    55%   5-8
plotly/validators/treemap/domain/_x.py                            4      0      0      0   100%
plotly/validators/treemap/domain/_y.py                            4      0      0      0   100%
plotly/validators/treemap/marker/__init__.py                     22     17      2      1    25%   5-21
plotly/validators/treemap/marker/_coloraxis.py                    4      0      0      0   100%
plotly/validators/treemap/marker/_colors.py                       4      0      0      0   100%
plotly/validators/violin/__init__.py                             66     61      2      1     9%   5-65
plotly/validators/violin/_alignmentgroup.py                       4      0      0      0   100%
plotly/validators/violin/_box.py                                  4      0      0      0   100%
plotly/validators/violin/_hovertemplate.py                        4      0      0      0   100%
plotly/validators/violin/_legendgroup.py                          4      0      0      0   100%
plotly/validators/violin/_line.py                                 4      0      0      0   100%
plotly/validators/violin/_marker.py                               4      0      0      0   100%
plotly/validators/violin/_name.py                                 4      0      0      0   100%
plotly/validators/violin/_offsetgroup.py                          4      0      0      0   100%
plotly/validators/violin/_orientation.py                          4      0      0      0   100%
plotly/validators/violin/_points.py                               4      0      0      0   100%
plotly/validators/violin/_scalegroup.py                           4      0      0      0   100%
plotly/validators/violin/_showlegend.py                           4      0      0      0   100%
plotly/validators/violin/_x0.py                                   4      0      0      0   100%
plotly/validators/violin/_x.py                                    4      0      0      0   100%
plotly/validators/violin/_xaxis.py                                4      0      0      0   100%
plotly/validators/violin/_y0.py                                   4      0      0      0   100%
plotly/validators/violin/_y.py                                    4      0      0      0   100%
plotly/validators/violin/_yaxis.py                                4      0      0      0   100%
plotly/validators/violin/box/__init__.py                          9      4      2      1    55%   5-8
plotly/validators/violin/box/_visible.py                          4      0      0      0   100%
plotly/validators/violin/line/__init__.py                         7      2      2      1    67%   5-6
plotly/validators/violin/line/_color.py                           4      0      0      0   100%
plotly/validators/violin/marker/__init__.py                      12      7      2      1    43%   5-11
plotly/validators/violin/marker/_color.py                         4      0      0      0   100%
plotly/validators/violin/marker/_symbol.py                        4      0      0      0   100%
plotly/validators/waterfall/__init__.py                          78     73      2      1     8%   5-77
plotly/validators/waterfall/_y.py                                 4      0      0      0   100%
test_init/__init__.py                                             3      0      0      0   100%
test_init/test_dependencies_not_imported.py                      22     17      2      0    29%   11-36
test_init/test_lazy_imports.py                                   22     17      8      0    23%   12-39
------------------------------------------------------------------------------------------------------
TOTAL                                                        172042  50098  86071  13487    73%
Wrote HTML report to htmlcov/index.html
(plotly_env_new) emelcinoglu@Emels-MacBook-Air plotly %
```

New coverage result:

```
plotly/validators/treemap/_parents.py                             4      0      0      0   100%
plotly/validators/treemap/_values.py                              4      0      0      0   100%
plotly/validators/treemap/domain/__init__.py                      9      4      2      1    55%   5-8
plotly/validators/treemap/domain/_x.py                            4      0      0      0   100%
plotly/validators/treemap/domain/_y.py                            4      0      0      0   100%
plotly/validators/treemap/marker/__init__.py                     22     17      2      1    25%   5-21
plotly/validators/treemap/marker/_coloraxis.py                    4      0      0      0   100%
plotly/validators/treemap/marker/_colors.py                       4      0      0      0   100%
plotly/validators/violin/__init__.py                             66     61      2      1     9%   5-65
plotly/validators/violin/_alignmentgroup.py                       4      0      0      0   100%
plotly/validators/violin/_box.py                                  4      0      0      0   100%
plotly/validators/violin/_hovertemplate.py                        4      0      0      0   100%
plotly/validators/violin/_legendgroup.py                          4      0      0      0   100%
plotly/validators/violin/_line.py                                 4      0      0      0   100%
plotly/validators/violin/_marker.py                               4      0      0      0   100%
plotly/validators/violin/_name.py                                 4      0      0      0   100%
plotly/validators/violin/_offsetgroup.py                          4      0      0      0   100%
plotly/validators/violin/_orientation.py                          4      0      0      0   100%
plotly/validators/violin/_points.py                               4      0      0      0   100%
plotly/validators/violin/_scalegroup.py                           4      0      0      0   100%
plotly/validators/violin/_showlegend.py                           4      0      0      0   100%
plotly/validators/violin/_x0.py                                   4      0      0      0   100%
plotly/validators/violin/_x.py                                    4      0      0      0   100%
plotly/validators/violin/_xaxis.py                                4      0      0      0   100%
plotly/validators/violin/_y0.py                                   4      0      0      0   100%
plotly/validators/violin/_y.py                                    4      0      0      0   100%
plotly/validators/violin/_yaxis.py                                4      0      0      0   100%
plotly/validators/violin/box/__init__.py                          9      4      2      1    55%   5-8
plotly/validators/violin/box/_visible.py                          4      0      0      0   100%
plotly/validators/violin/line/__init__.py                         7      2      2      1    67%   5-6
plotly/validators/violin/line/_color.py                           4      0      0      0   100%
plotly/validators/violin/marker/__init__.py                      12      7      2      1    43%   5-11
plotly/validators/violin/marker/_color.py                         4      0      0      0   100%
plotly/validators/violin/marker/_symbol.py                        4      0      0      0   100%
plotly/validators/waterfall/__init__.py                          78     73      2      1     8%   5-77
plotly/validators/waterfall/_y.py                                 4      0      0      0   100%
test_init/__init__.py                                             3      0      0      0   100%
test_init/test_dependencies_not_imported.py                      22     17      2      0    29%   11-36
test_init/test_lazy_imports.py                                   22     17      8      0    23%   12-39
------------------------------------------------------------------------------------------------------
TOTAL                                                        172176  50042  86091  13488    73%
Wrote HTML report to htmlcov/index.html
```

Statement of individual contributions

Emel Dzhinoglu

- Instrumented functions _vectorize_zvalue and annotation_dict_for_label
- Added tests for test_vectorize_zvalue.py and test_annotation_dic_for_label.py

Poyraz Temiz

- Instrumented functions is_source_key(key) and update_session_plot_options(**kwargs)
- Added tests for is_source_key(key) and update_session_plot_options(**kwargs)

Georgi Ivanov

- Instrumented functions should_retry() and sign_in()
- Updated the according test files by adding extra tests that cover the above-mentioned functions

Selin Saracoglu

- Instrumented functions _replace_new_line and _list_repr_elided
- Added tests for test_replace_new_line and test_list_repr_elided