Timer unit: 1e-09 s

Total time: 0.0127658 s
File: /tmp/ipykernel_80/672168055.py
Function: check_fixes_and_bounds at line 1

Old function

Line #	Hits	Time	Per Hit	% Time	Line Contents
1					def check_fixes_and_bounds(constr_info, transformations, parnames):
2					"""Check fixes.
3					
4					Warn the user if he fixes a parameter to a value even though that parameter has
5					a different non-nan value in params
6					
7					check that fixes are compatible with other constraints.
8					
9					Args:
10					constr_info (dict): Dict of 1d numpy arrays with info about constraints.
11					transformations (list): Processed transforming constraints.
12					parnames (list): List of parameter names.
13					
14					"""
15	1	3164389.0	3164389.0	24.8	df = pd.DataFrame(constr_info, index=parnames)
16					
17					# Check fixes and bounds are compatible with other constraints
18	1	180.0	180.0	0.0	prob_msg = (
19	1	340.0	340.0	0.0	"{} constraints are incompatible with fixes or bounds. "
20					"This is violated for:\n{}"
21					)
22					
23	1	180.0	180.0	0.0	cov_msg = (
24	1	310.0	310.0	0.0	"{} constraints are incompatible with fixes or bounds except for the first "
25					"parameter. This is violated for:\n{}"
26					)
27					
28	1	1493.0	1493.0	0.0	for constr in transformations:
29	1	1263.0	1263.0	0.0	if constr["type"] in ["covariance", "sdcorr"]:
30	1	784793.0	784793.0	6.1	subset = df.iloc[constr["index"][1:]]
31	1	430494.0	430494.0	3.4	if subset["is_fixed_to_value"].any():
32					problematic = subset[subset["is_fixed_to_value"]].index
33					raise InvalidConstraintError(
34					cov_msg.format(constr["type"], problematic)
35					)
36	1	2628840.0	2628840.0	20.6	if np.isfinite(subset[["lower_bounds", "upper_bounds"]]).any(axis=None):
37					problematic = (
38					subset.replace([-np.inf, np.inf], np.nan).dropna(how="all").index
39					)
40					raise InvalidConstraintError(
41					cov_msg.format(constr["type"], problematic)
42					)
43					elif constr["type"] == "probability":
44					subset = df.iloc[constr["index"]]
45					if subset["is_fixed_to_value"].any():
46					problematic = subset[subset["is_fixed_to_value"]].index
47					raise InvalidConstraintError(
48					prob_msg.format(constr["type"], problematic)
49					)
50					if np.isfinite(subset[["lower_bounds", "upper_bounds"]]).any(axis=None):
51					problematic = (
52					subset.replace([-np.inf, np.inf], np.nan).dropna(how="all").index
53					)
54					raise InvalidConstraintError(
55					prob_msg.format(constr["type"], problematic)
56					)
57					
58	1	4555192.0	4555192.0	35.7	invalid = df.query("lower_bounds >= upper_bounds")[["lower_bounds", "upper_bounds"]]
59	1	330.0	330.0	0.0	msg = (
60	1	1194458.0	1194458.0	9.4	"lower_bound must be strictly smaller than upper_bound. "
61	1	250.0	250.0	0.0	f"This is violated for:\n{invalid}"
62					)
63	1	3277.0	3277.0	0.0	if len(invalid) > 0:
64					raise InvalidConstraintError(msg)

```
Total time: 0.00025644 s
File: /tmp/ipykernel_80/2913211772.py
Function: check_fixes_and_bounds_1 at line 1
```

New function

Line #	Hits	Time	Per Hit	% Time	Line Contents
1					def check_fixes_and_bounds_1(constr_info, transformations, parnames):
2					"""Check fixes.
3					
4					Warn the user if he fixes a parameter to a value even though that parameter has
5					a different non-nan value in params
6					
7					check that fixes are compatible with other constraints.
8					
9					Args:
10					constr_info (dict): Dict of 1d numpy arrays with info about constraints.
11					transformations (list): Processed transforming constraints.
12					parnames (list): List of parameter names.
13					
14					"""
15	1	1543.0	1543.0	0.6	df = constr_info
16	1	691.0	691.0	0.3	df["index"] = parnames
17					
18	1	280.0	280.0	0.1	prob_msg = (
19	1	291.0	291.0	0.1	"{} constraints are incompatible with fixes or bounds. "
20					"This is violated for:\n{}"
21					)
22					
23	1	280.0	280.0	0.1	cov_msg = (
24	1	170.0	170.0	0.1	"{} constraints are incompatible with fixes or bounds except for the first "
25					"parameter. This is violated for:\n{}"
26					)
27					
28	1	1082.0	1082.0	0.4	for constr in transformations:
29	1	1593.0	1593.0	0.6	if constr["type"] in ["covariance", "sdcorr"]:
30	1	85950.0	85950.0	33.5	subset = _iloc(df, constr["index"], True)
31	1	6342.0	6342.0	2.5	if any(subset["is_fixed_to_value"]):
32					problematic = np.where(subset["is_fixed_to_value"])[0]
33					raise InvalidConstraintError(
34					cov_msg.format(constr["type"], problematic)
35					)
36	1	96430.0	96430.0	37.6	if np.isfinite([subset["lower_bounds"], subset["upper_bounds"]]).any():
37					problematic = [k for k, v in subset.items() if np.any(~np.isfinite(v))]
38					raise InvalidConstraintError(
39					prob_msg.format(constr["type"], problematic)
40					)
41					elif constr["type"] == "probability":
42					subset = _iloc(df, constr["index"], False)
43					if any(subset["is_fixed_to_value"]):
44					problematic = np.where(subset["is_fixed_to_value"])[0].tolist()
45					raise InvalidConstraintError(
46					prob_msg.format(constr["type"], problematic)
47					)
48					
49					if np.isfinite([subset["lower_bounds"], subset["upper_bounds"]]).any():
50					problematic = [k for k, v in subset.items() if np.any(~np.isfinite(v))]
51					raise InvalidConstraintError(
52					prob_msg.format(constr["type"], problematic)
53					)
54					
55	1	331.0	331.0	0.1	invalid = {}
56	1	652.0	652.0	0.3	lower_bounds = df["lower_bounds"]
57	1	381.0	381.0	0.1	upper_bounds = df["upper_bounds"]
58					
59	1	19647.0	19647.0	7.7	invalid = np.where(lower_bounds >= upper_bounds)[0]
60					
61	1	271.0	271.0	0.1	msg = (
62	1	39303.0	39303.0	15.3	"lower_bound must be strictly smaller than upper_bound. "
63	1	251.0	251.0	0.1	f"This is violated for:\n{invalid}"
64					)
65	1	952.0	952.0	0.4	if len(invalid) > 0:
66					raise InvalidConstraintError(msg)