Supplementary Material Thesis

The purpose of this folder is to make the experiments presented in my thesis reproducible.

Head over to the main README for more information and instructions of how to install the package.

Visualize effect of β-adaptivity

```
import rareeventestimation as ree
import numpy as np
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import scipy as sp
from rareeventestimation.evaluation.constants import *
from IPython.display import display, Markdown
# recommended: use autoreload for development: https://ipython.readthedocs.io/en/stable/config/extensions
%load_ext autoreload
%autoreload 2
```

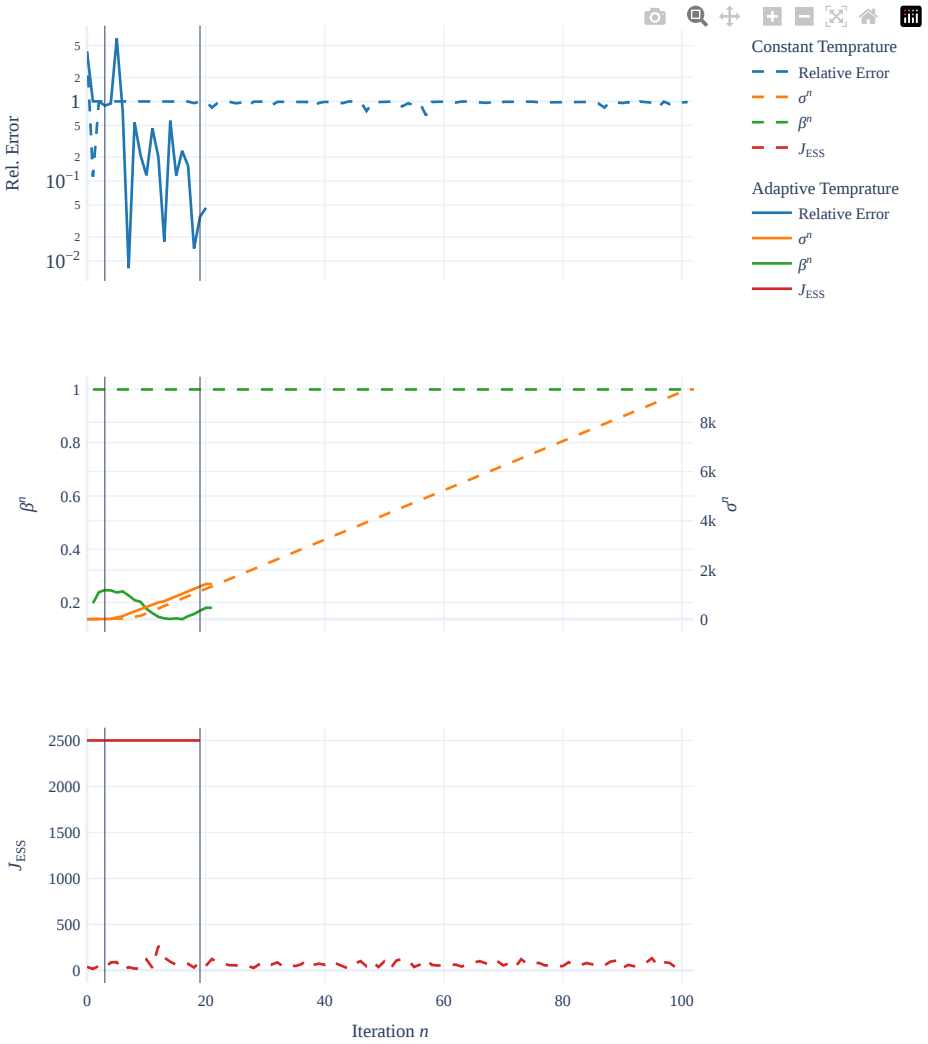
Plot error and parameters during iterations

```
# solvers
N = 5000
d=50
seed=1
cvar_tgt=5
cbree_const = ree.CBREE(beta_adaptivity=1.0,
                        save_history=True,
                        stepsize_tolerance=1,
                        seed=seed,
                        return_other = True,
                        divergence_check=False,
                        cvar_tgt=cvar_tgt,
                        return_caches = True,
                        name = "Constant Temprature",
                        )
cbree = ree.CBREE(save_history=True,
                  seed=seed,
                  stepsize_tolerance=1,
                  return_other=True,
                  divergence_check=False,
                  return_caches = True,
                  name = "Adaptive Temprature",
                  cvar_tgt=cvar_tgt)
solver_list = [cbree_const, cbree]

# prepare figure
solver_dash = dict(zip([str(s) for s in solver_list], ["dash", "solid"]))
solver_marker = dict(zip([str(s) for s in solver_list], [1,3]))
fig = make_subplots(rows=3,
                    cols=1,
                    shared_xaxes=True,
                    specs=[[{"secondary_y": False}],
                           [{"secondary_y": True}],
                           [{"secondary_y": False}]])
fig_name = "constant_beta_iterations"

# populate figure
solution_list = []
for solver in solver_list:
    # solve
    prob = ree.make_linear_problem(d)
    prob.set_sample(N, seed = seed)
    sol = solver.solve(prob)
    sol.other["beta"][0] = np.nan
    solution_list.append(sol)
    # plot error
    fig.add_trace(
        go.Scatter(
            y=sol.get_rel_err(prob),
            name = f"Relative Error",
            line_dash=solver_dash[str(solver)],
            marker_color=CMAP[0],
            marker_symbol=solver_marker[str(solver)],
            legendgroup=str(solver),
            legendgrouptitle_text=str(solver)
        ),
        row=1,
        col=1)
    # plot parameter
    params = {
        "sigma": STR_SIGMA_N,
        "beta" : STR_BETA_N,
    }
    for i,(param,name) in enumerate(params.items()):
        fig.add_trace(
        go.Scatter(
            y=sol.other[param],
            name = name,
            line_dash=solver_dash[str(solver)],
            marker_color=CMAP[i+1],
            marker_symbol=solver_marker[str(solver)],
            legendgroup=str(solver),
        ),
        row=2
```

Skip to main content

```
    # plot effective sample size
    fig.add_trace(
        go.Scatter(
            y=sol.other["ess"],
            name = STR_J_ESS,
            line_dash=solver_dash[str(solver)],
            marker_color=CMAP[3],
            marker_symbol=solver_marker[str(solver)],
            legendgroup=str(solver),
        ),
        row=3,
        col=1)
# Add vertical lines
iters = [3,19]
for i in iters:
    fig.add_vline(i,line_width=0.5)

# style and save fig
fig.update_yaxes(title_text="Rel. Error", type="log", row=1, col=1)
fig.update_yaxes(title_text=STR_J_ESS, row=3, col=1)
fig.update_yaxes(title_text=STR_SIGMA_N, title_standoff=0, row=2, col=1, secondary_y=True)
fig.update_yaxes(title_text=STR_BETA_N, row=2, col=1, secondary_y=False)
fig.update_xaxes(title_text="Iteration <i>n<i>", row=3, col=1)
fig.update_layout(**MY_LAYOUT)
fig.update_layout(height=800)
fig.write_image(fig_name + ".png",scale=7)
fig.show()
# Make and save caption
fig_description = f"Solving the {prob.name} with the CBREE method using  \
$J = {N}$ particles, \
stopping criterion $\\Delta_{{\\text{{Target}}}} = {cvar_tgt}$, \
stepsize tolerance $\\epsilon_{{\\text{{Target}}}} = {cbree.stepsize_tolerance}$, \
controlling the increase of $\\sigma$ with $\\text{{Lip}}(\\sigma) = {cbree.lip_sigma}$ \
and approximating the indicator function with {INDICATOR_APPROX_LATEX_NAME[cbree.tgt_fun]}. \
No divergence check has been performed and the solver was able to run for at most {cbree.num_steps} itera
The vertical lines mark iterations whose weights we investigate."
with open("desc_constant_beta.tex", "w") as file:
    file.write(fig_description)
display(Markdown(fig_description))
```

```
adaptive: True
adaptive: True
```



Solving the Linear Problem (d=50) with the CBREE method using $J = 5000$ particles, stopping criterion $\Delta_{\text{Target}} = 5$, stepsize tolerance $\epsilon_{\text{Target}} = 1$, controlling the increase of σ with $\text{Lip}(\sigma) = 1$ and approximating the indicator function with $I_\text{{alg}}$. No divergence check has been performed and the solver was able to run for at most 100 iterations. The vertical lines mark iterations whose weights we investigate.

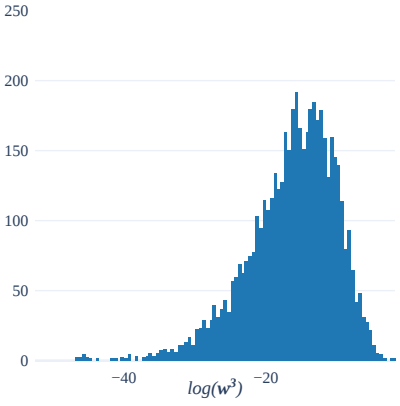
Skip to main content

Make histogram of weights for selected iterations

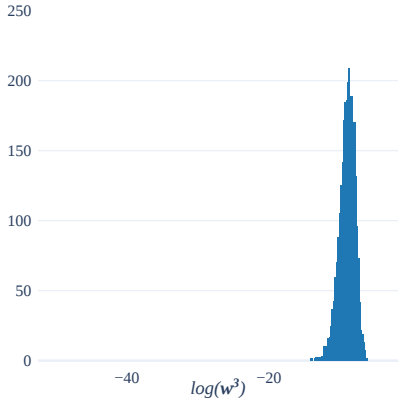
```
# set up figure
hist_fig_name="constant_beta_histograms"
hist_fig = make_subplots(cols=2, rows = 2,subplot_titles=("Plot 1", "Plot 2", "plot3", "plot4"))
x_lims = None
# populate
plot_counter=0
for i_n, i in enumerate(iters):
    for soln, sol in enumerate(solution_list):
        w = solver_list[soln]._CBREE__compute_weights(sol.other["cache_list"][i], return_weights=True)
        hist_name = f"<i>log</i> of weights <i><b>w<sup>{i}</sup></b></i>" +(  f" ({STR_BETA_N} constant
            if not solver_list[soln].beta_adaptivity else "") + \
            f": Skewness {sp.stats.skew(np.exp(w)):.1f}"
        hist = go.Histogram(
            x=np.log(w),
            showlegend=False,
            marker_color = CMAP[0])
        hist_fig.add_trace(hist, row=i_n+1, col=soln+1)
        # get x limits
        hist_fig.layout.annotations[plot_counter].update(text= hist_name)
        this_x_lims = np.array([np.min(np.log(w)), np.max(np.log(w))])
        if x_lims is None:
            x_lims=this_x_lims
        else:
            x_lims[0] = min(x_lims[0], this_x_lims[0])
            x_lims[1] = max(x_lims[1], this_x_lims[1])
        plot_counter += 1
# set x limits
hist_fig.update_xaxes(range=x_lims*1.01)

# add subplot title
for soln, sol in enumerate(solution_list):
    for i_n, i in enumerate(iters):
        hist_fig.update_xaxes(title=f"<i>log(<b>w<sup>{i}</sup></b>)<i>", row=i_n+1, col = soln+1,title_s
# style and save
hist_fig.update_yaxes(range=[0,250])
hist_fig.update_layout(**MY_LAYOUT)
hist_fig.update_layout(height=800)
hist_fig.update_layout(margin_t=50)
hist_fig.for_each_annotation(lambda a: a.update(y=a.y+0.01))
hist_fig.write_image(hist_fig_name + ".png",scale=WRITE_SCALE)
hist_fig.show()
```

Skip to main content

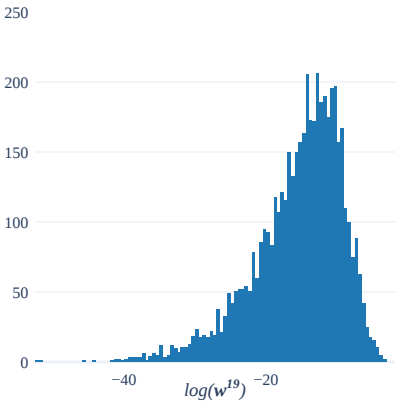
log of weights w^3 (β^n constant): Skewness 44.9



log of weights w^3: Skewness 2.3



log of weights w^{19} (β^n constant): Skewness 15.8



log of weights w^{19}: Skewness 2.4