in predicted cal case:

Z

has_ordinal_parents

no → has_latent_range

no → normal sampling as before

yes →

df = df_create_from_sampled *

counterfactual_freq = [ ]

for i, zi in enumerate latentdf[z]:

range_samples = student_logistic_truncated(n, lower, upper) ***

range_samples save f"z/sampling/range_samples_{i}_{idx}.csv"

df = df.row[i], copy n times

samples = sample_node_ordinal(df, ...)

samples save f"z/sampling/sampling_{i}_{idx}.csv"

counterfactual_fry.append[freq(samples)] # tuples here (0.1, 0.2, 0.7)

counterfactual_freq . save ("z/sampling/sampled.pt")

Yes → has_latent_range

yes →

df = create_df_from_sampled *

parents = parents of (z)

counterfactual_freq = [ ]

for i, zi in enumerate (latentdf[z]):

range_samples = student_lg(n, lower, upper) **

save

df = load_pa_with_range(i, df, parents) ***

samples "

"

"

"

counterfactual.save("z/sampling/sampled.pt")

no →

df = create_df_from_sampled

parents = parents of (z)

for i, zi in enumerate(latent_df[z]):

df = load_pa_with_range(i, df, parents)

is continuous:

sampled =

is ordinal:

sampled

counterfactual.save("z/sampling/sampled.pt")

* add fct such that tuples are not loaded but skipped!

** student logistic truncated implement!

*** load df with range

*** load df with range(i, df, parents):

return df