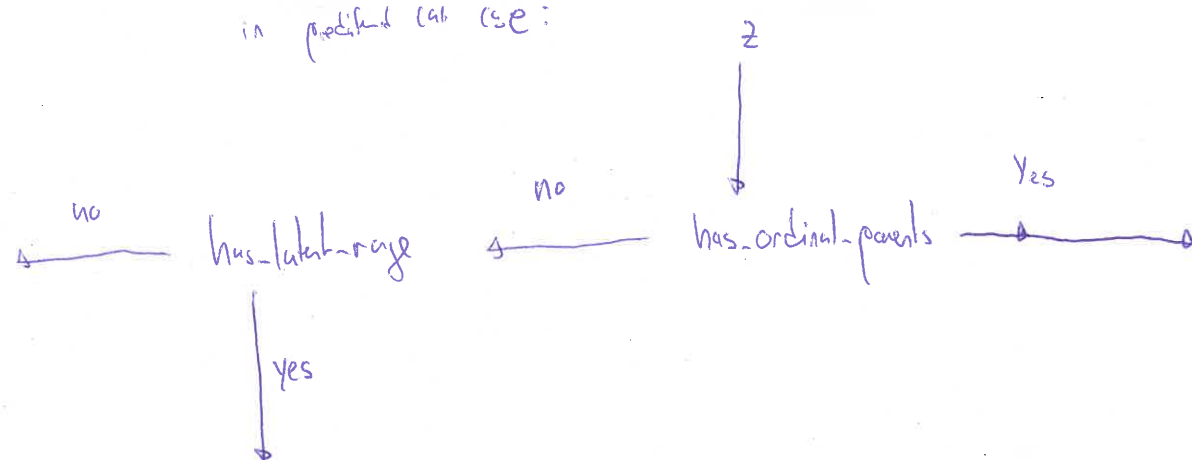




normal sample
as before

in modified case:



df = df_create_from_sampled
counterfactual_freq = []

~~XXXXXXXXXX~~

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

range_samples = statlib LogisticTruncated(n, lower, upper)

range_samples.save(f"z/sample/range_samples_{i3}.csv")

df = df_row[i], copy n times

samples = sample_node_ordinal(df, ...)

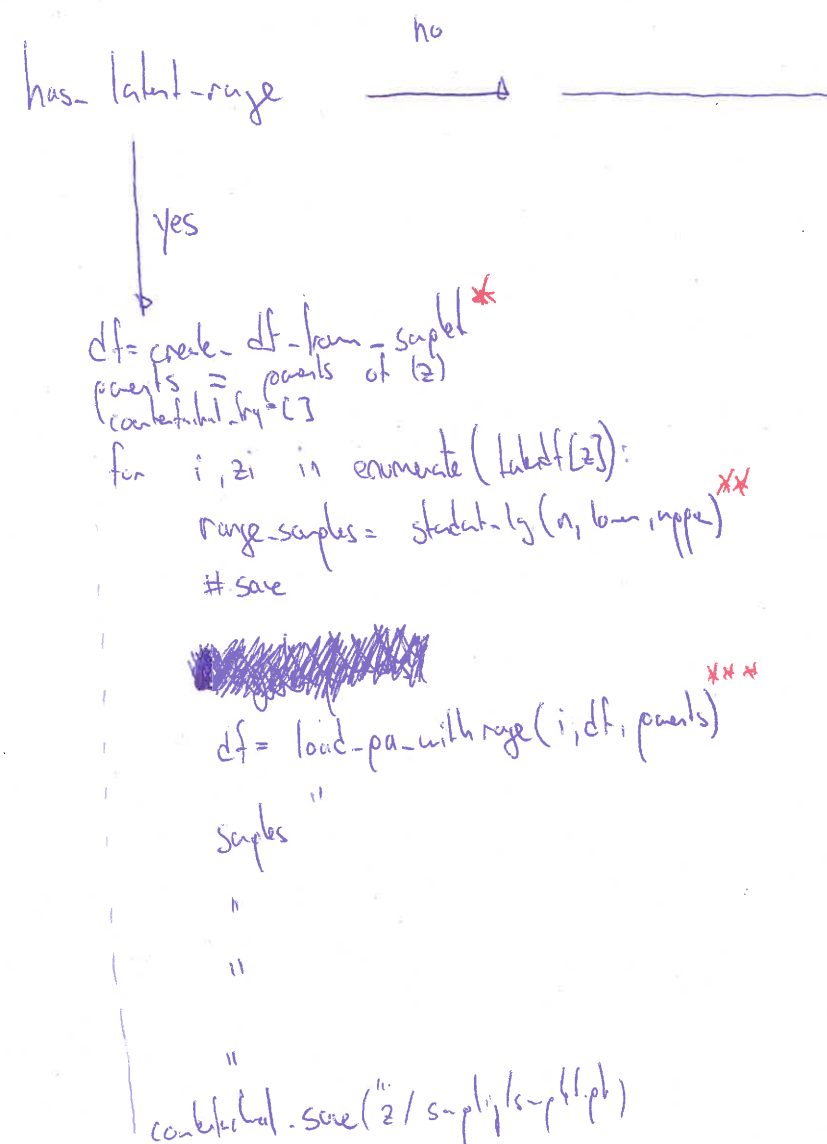
samples.save(f"z/sample/sample_{i3}_{id3}.csv")

counterfactual_freq.append(freq(samples)) # types here (0,1,0,2,0,2)

counterfactual_freq.save(f"z/sample/sample.pt")

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

return df



df = create_df_from_sampled
parents = parents of (z)
counterfactual_freq = []

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

range_samples = statlib LogisticTruncated(n, lower, upper)

save

~~XXXXXXXXXX~~

df = load_df_with_range(i, df, parents)

samples

"

"

"

counterfactual.save(f"z/sample/sample.pt")

* add for sure that types are not loaded but skipped!

*** Student logistic truncated implement!

*** load df with range

df = create_df_from_sampled
parents = parents of (z)

~~XXXXXXXXXX~~

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

df = load_df_with_range(i, df, parents)

is continuous:

sampled =

is ordinal:

sampled

counterfactual.save(f"z/sample/sample.pt")