In [1]:

```
%cd ../../
```

```
/home/jan/FMF/masters
```

Train sig generator

In [2]:

```
from ml_hep_sim.pipeline.prebuilt.flow_pipeline import FlowPipeline
```

In [3]:

```
sig_override = {
    "datasets": {
        "data_name": "higgs_sig",
        "data_params": {
            "subset_n": [10 ** 6, 10 ** 5, 10 ** 5],
            "rescale": "logit_normal",
            "to_gpu": True,
        },
    },
    "logger_config": {"run_name": "Higgs_glow", "experiment_name": "analysis"},
    "trainer_config": {"gpus": 1, "max_epochs": 51},
    "model_config": {"num_flows": 10},
}

FP_sig = FlowPipeline(
    run_name="Higgs_Glow_sig",
    model_name="Glow",
    override=sig_override,
    pipeline_path="ml_pipeline/analysis/Higgs_glow/",
)

FP_sig.build_train_pipeline()
FP_sig.fit()
```

```
WARNING:root:Built flow training pipeline...
WARNING:root:Loadindg fitted flow...
```

Out[3]:

```
<ml_hep_sim.pipeline.pipes.Pipeline at 0x7fa40a70a4c0>
```

Train bkg generator

In [4]:

```
import copy
```

In [5]:

```
bkg_override = copy.deepcopy(sig_override)
bkg_override["datasets"]["data_name"] = "higgs_bkg"

FP_bkg = FlowPipeline(
    run_name="Higgs_Glow_bkg",
    model_name="Glow",
    override=bkg_override,
    pipeline_path="ml_pipeline/analysis/Higgs_glow/",
)

FP_bkg.build_train_pipeline()
FP_bkg.fit()
```

```
WARNING:root:Built flow training pipeline...
WARNING:root:Loadindg fitted flow...
```

Out[5]:

```
<ml_hep_sim.pipeline.pipes.Pipeline at 0x7fa42c56e5b0>
```

Build sig and bkg generators using inference pipelines

In [6]:

```
N_gen = 10 ** 6

FP_sig.build_inference_pipeline(N_gen, rescale_data=False, device="cuda")
FP_bkg.build_inference_pipeline(N_gen, rescale_data=False, device="cuda")

sig_infer_pipeline = FP_sig.pipeline["inference_pipeline"]
bkg_infer_pipeline = FP_bkg.pipeline["inference_pipeline"]
```

```
WARNING:root:Built flow inference pipeline...
WARNING:root:Built flow inference pipeline...
```

Get MC sig data block

In [7]:

```
from ml_hep_sim.pipeline.blocks import DatasetBuilderBlock, ReferenceDataLoaderBlock
```

In [8]:

```
mc_sig_config = copy.deepcopy(FP_sig.pipeline["train_pipeline"].pipes[0]) # same config as before
mc_sig_config.config["datasets"]["data_params"]["subset_n"] = [0, 0, N_gen]
mc_sig_config.config["datasets"]["data_params"]["rescale"] = "none"

b_mc_sig_dataset = DatasetBuilderBlock()(mc_sig_config)
b_mc_sig_data = ReferenceDataLoaderBlock(rescale_reference="logit_normal", device="cpu")(b_mc_sig_dataset)
```

Get MC bkg data block

In [9]:

```
mc_bkg_config = copy.deepcopy(FP_bkg.pipeline["train_pipeline"].pipes[0])
mc_bkg_config.config["datasets"]["data_params"]["subset_n"] = [0, 0, N_gen]
mc_bkg_config.config["datasets"]["data_params"]["rescale"] = "none"

b_mc_bkg_dataset = DatasetBuilderBlock()(mc_bkg_config)
b_mc_bkg_data = ReferenceDataLoaderBlock(rescale_reference="logit_normal", device="cpu")(b_mc_bkg_dataset)
```

Train binary classifier

In [10]:

```
from ml_hep_sim.pipeline.prebuilt.classifier_pipeline import ClassifierPipeline
```

In [11]:

```
override = {
    "datasets": {
        "data_name": "higgs",
        "data_params": {
            "subset_n": [10 ** 6, 10 ** 5, 10 ** 5],
            "rescale": "logit_normal",
            "to_gpu": True,
        },
    },
    "logger_config": {"run_name": "Higgs_classifier", "experiment_name": "analysis"},
    "trainer_config": {"gpus": 1, "max_epochs": 101},
    "model_config": {
        "resnet": False,
        "hidden_layers": [256, 128, 64, 1],
    },
}

CP = ClassifierPipeline(
    "Higgs_classifier", override=override, pipeline_path="ml_pipeline/analysis/classifiers/"
)

CP.build_train_pipeline()
CP.fit(force=False)
```

```
WARNING:root:Built classification training pipeline...
WARNING:root:Loadindg fitted classifier...
```

Out[11]:

```
<ml_hep_sim.pipeline.pipes.Pipeline at 0x7fa40a71cd00>
```

Load trained classifier

In [12]:

```
from ml_hep_sim.pipeline.blocks import ModelLoaderBlock
```

In [13]:

```
class_train_pipeline = CP.pipeline["train_pipeline"]

config = class_train_pipeline.pipes[0] # classifier config block
model = class_train_pipeline.pipes[3] # classifier model trainer block

b_classifier_model = ModelLoaderBlock(device="cuda")(config, model)
```

Use classifier

In [14]:

```
from ml_hep_sim.pipeline.blocks import ClassifierRunnerBlock
```

In [15]:

```
b_flow_sig_generated = sig_infer_pipeline.pipes[-1]
b_flow_bkg_generated = bkg_infer_pipeline.pipes[-1]

b_sig_gen_class = ClassifierRunnerBlock(save_data=False, device="cuda")(b_flow_sig_generated, b_classifier_mode
b_bkg_gen_class = ClassifierRunnerBlock(save_data=False, device="cuda")(b_flow_bkg_generated, b_classifier_mode

b_sig_mc_class = ClassifierRunnerBlock(save_data=False, device="cuda")(b_mc_sig_data, b_classifier_model)  # MC
b_bkg_mc_class = ClassifierRunnerBlock(save_data=False, device="cuda")(b_mc_bkg_data, b_classifier_model)  # MC
```

Use variable

In [16]:

```
from ml_hep_sim.pipeline.blocks import VariableExtractBlock
from ml_hep_sim.analysis.utils import get_colnames_dict
```

In [17]:

```
var = "m bb"

dct = get_colnames_dict()
idx = dct[var]
```

```
WARNING:root:available variables: {'lepton pT': 0, 'lepton eta': 1, 'missing energy': 2, 'jet1 pT': 3, 'jet1 eta
': 4, 'jet2 pT': 5, 'jet2 eta': 6, 'jet3 pT': 7, 'jet3 eta': 8, 'jet4 pT': 9, 'jet4 eta': 10, 'm jj': 11, 'm jjj
': 12, 'm lv': 13, 'm jlv': 14, 'm bb': 15, 'm wbb': 16, 'm wwbb': 17}
```

In [18]:

```
b_sig_gen_var = VariableExtractBlock(idx, save_data=False, device="cuda")(b_flow_sig_generated) # sig gen var
b_bkg_gen_var = VariableExtractBlock(idx, save_data=False, device="cuda")(b_flow_bkg_generated) # bkg gen var

b_sig_mc_var = VariableExtractBlock(idx, save_data=False, device="cuda")(b_mc_sig_data) # MC sig var
b_bkg_mc_var = VariableExtractBlock(idx, save_data=False, device="cuda")(b_mc_bkg_data) # MC bkg var
```

Build and fit

In [19]:

```
from ml_hep_sim.pipeline.pipes import Pipeline
```

In [20]:

```
pipe = Pipeline()
pipe.compose(
    b_mc_sig_dataset,
    b_mc_sig_data,
    b_mc_bkg_dataset,
    b_mc_bkg_data,
    b_classifier_model,
    sig_infer_pipeline,
    bkg_infer_pipeline,
    b_sig_gen_var,
    b_bkg_gen_var,
    b_sig_mc_var,
    b_bkg_mc_var,
    b_sig_gen_class,
    b_bkg_gen_class,
    b_sig_mc_class,
    b_bkg_mc_class,
)
pipe.fit()
```

```
WARNING:root:fitting #0: <ml_hep_sim.pipeline.blocks.DatasetBuilderBlock object at 0x7fa40a70a970>!
WARNING:root:fitting #1: <ml_hep_sim.pipeline.blocks.ReferenceDataLoaderBlock object at 0x7fa40a71c0a0>!
WARNING:root:fitting #2: <ml_hep_sim.pipeline.blocks.DatasetBuilderBlock object at 0x7fa42c56e7c0>!
WARNING:root:fitting #3: <ml_hep_sim.pipeline.blocks.ReferenceDataLoaderBlock object at 0x7fa40a70aa90>!
WARNING:root:fitting #4: <ml_hep_sim.pipeline.blocks.ModelLoaderBlock object at 0x7fa40a71c640>!
WARNING:root:fitting #5: <ml_hep_sim.pipeline.blocks.ModelLoaderBlock object at 0x7fa40a70a3d0>!
WARNING:root:fitting #6: <ml_hep_sim.pipeline.blocks.DataGeneratorBlock object at 0x7fa40a70af40>!
WARNING:root:Generating 1000000 examples in 10 chunks of 100000 examples each using GlowFlowModel model.
100%|██████████| 10/10 [00:00<00:00, 12.95it/s]
WARNING:root:fitting #7: <ml_hep_sim.pipeline.blocks.GeneratedDataVerifierBlock object at 0x7fa40a70aca0>!
WARNING:root:Generated data check...
WARNING:root:nan OK
WARNING:root:pos-inf OK
WARNING:root:neg-inf OK
WARNING:root:pos-inf or neg-inf OK
WARNING:root:pos-inf or neg-inf or nan OK
WARNING:root:fitting #8: <ml_hep_sim.pipeline.blocks.ModelLoaderBlock object at 0x7fa4305bf850>!
WARNING:root:fitting #9: <ml_hep_sim.pipeline.blocks.DataGeneratorBlock object at 0x7fa40a70a190>!
WARNING:root:Generating 1000000 examples in 10 chunks of 100000 examples each using GlowFlowModel model.
100%|██████████| 10/10 [00:00<00:00, 13.03it/s]
WARNING:root:fitting #10: <ml_hep_sim.pipeline.blocks.GeneratedDataVerifierBlock object at 0x7fa40a70a520>!
WARNING:root:Generated data check...
WARNING:root:nan OK
WARNING:root:pos-inf OK
WARNING:root:neg-inf OK
WARNING:root:pos-inf or neg-inf OK
WARNING:root:pos-inf or neg-inf or nan OK
WARNING:root:fitting #11: <ml_hep_sim.pipeline.blocks.VariableExtractBlock object at 0x7fa40a76e3d0>!
WARNING:root:fitting #12: <ml_hep_sim.pipeline.blocks.VariableExtractBlock object at 0x7fa40a76ec40>!
WARNING:root:fitting #13: <ml_hep_sim.pipeline.blocks.VariableExtractBlock object at 0x7fa40a76e460>!
WARNING:root:fitting #14: <ml_hep_sim.pipeline.blocks.VariableExtractBlock object at 0x7fa40a76e910>!
WARNING:root:fitting #15: <ml_hep_sim.pipeline.blocks.ClassifierRunnerBlock object at 0x7fa40a76ee50>!
100%|██████████| 10/10 [00:00<00:00, 3802.29it/s]
WARNING:root:fitting #16: <ml_hep_sim.pipeline.blocks.ClassifierRunnerBlock object at 0x7fa40a76ef40>!
100%|██████████| 10/10 [00:00<00:00, 1696.93it/s]
WARNING:root:fitting #17: <ml_hep_sim.pipeline.blocks.ClassifierRunnerBlock object at 0x7fa40a76ecd0>!
100%|██████████| 10/10 [00:00<00:00, 147.13it/s]
WARNING:root:fitting #18: <ml_hep_sim.pipeline.blocks.ClassifierRunnerBlock object at 0x7fa40a76eb20>!
100%|██████████| 10/10 [00:00<00:00, 147.74it/s]
```

```
Out[20]: <ml_hep_sim.pipeline.pipes.Pipeline at 0x7fa40a76eee0>
```

```
In [21]: pipe.pipes[6]#.trained_model.__class__.__name__
```

```
Out[21]: <ml_hep_sim.pipeline.blocks.DataGeneratorBlock at 0x7fa40a70af40>
```

```
In [22]: pipe.draw_pipeline_tree(to_graphviz_file="pipeline_mc", block_idx=-1)
         pipe.draw_pipeline_tree(to_graphviz_file="pipeline_gen", block_idx=-3)

         pipe.draw_pipeline_tree(to_graphviz_file="pipeline_gen_cut", block_idx=-7)
```

```
Out[22]: <treelib.tree.Tree at 0x7fa409d1a5e0>
```

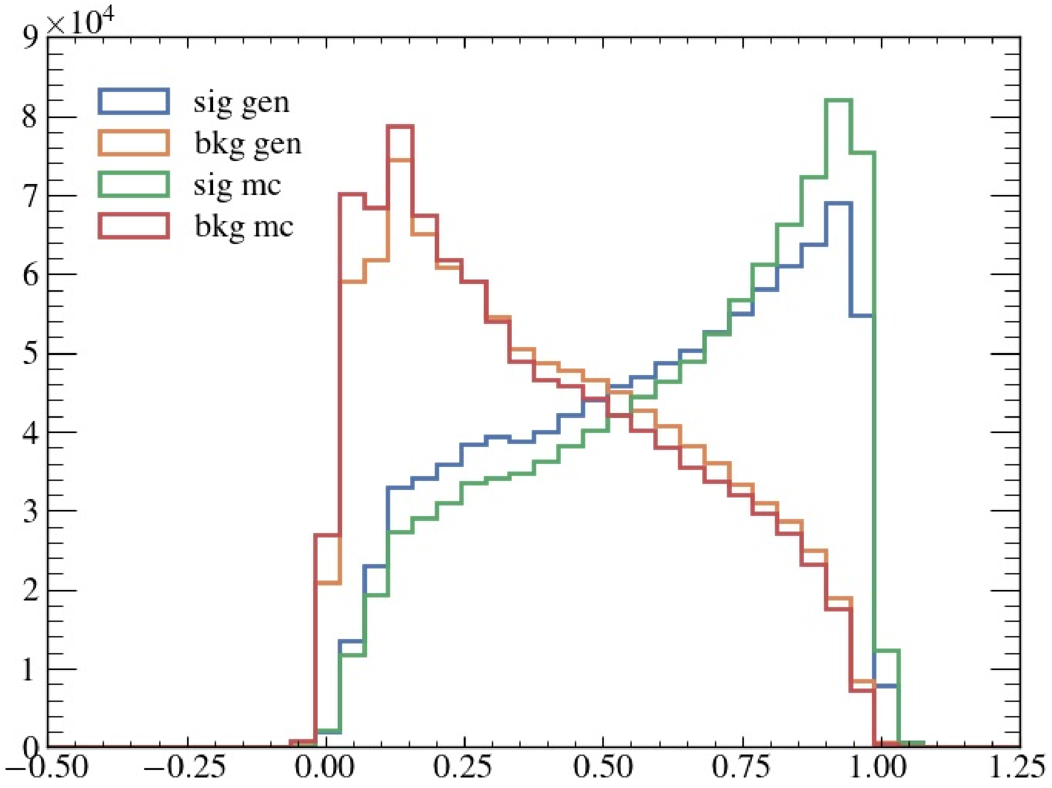
Plot histograms - classifier

```
In [23]: import matplotlib.pyplot as plt
         from ml_hep_sim.plotting.style import style_setup, set_size

         set_size()
         style_setup(seaborn_pallete=True)
```

```
In [24]: sig_gen = pipe.pipes[-4].results
         bkg_gen = pipe.pipes[-3].results
         sig_mc = pipe.pipes[-2].results[: len(sig_gen)]
         bkg_mc = pipe.pipes[-1].results[: len(sig_gen)]
```

```
In [25]: plt.hist(sig_gen, histtype="step", range=(-0.5, 1.25), bins=40, lw=2)
         plt.hist(bkg_gen, histtype="step", range=(-0.5, 1.25), bins=40, lw=2)
         plt.hist(sig_mc, histtype="step", range=(-0.5, 1.25), bins=40, lw=2)
         plt.hist(bkg_mc, histtype="step", range=(-0.5, 1.25), bins=40, lw=2)
         plt.legend(["sig gen", "bkg gen", "sig mc", "bkg mc"], loc="upper left")
         plt.show()
```



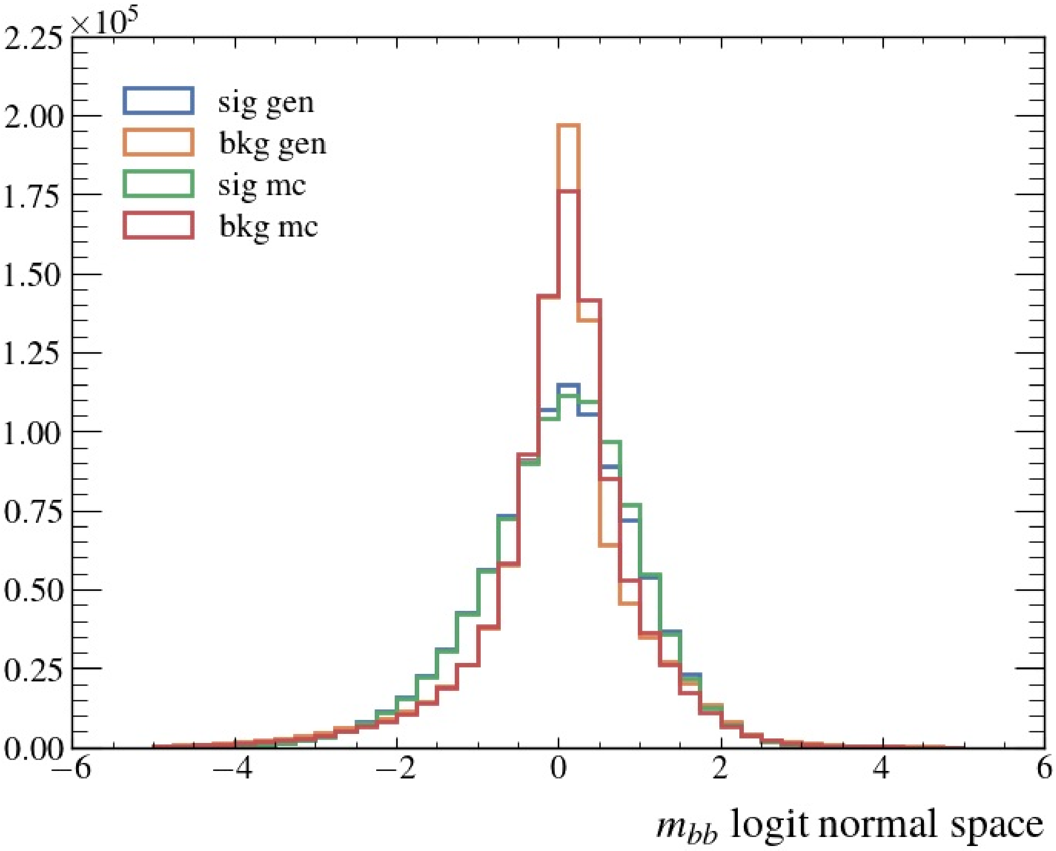
Plot histograms - variable

In [26]:

```
sig_gen = pipe.pipes[-1-4].results
bkg_gen = pipe.pipes[-2-4].results
sig_mc = pipe.pipes[-3-4].results[: len(sig_gen)]
bkg_mc = pipe.pipes[-4-4].results[: len(sig_gen)]
```

In [27]:

```
plt.hist(sig_gen, histtype="step", range=(-5, 5), bins=40, lw=2)
plt.hist(bkg_gen, histtype="step", range=(-5, 5), bins=40, lw=2)
plt.hist(sig_mc, histtype="step", range=(-5, 5), bins=40, lw=2)
plt.hist(bkg_mc, histtype="step", range=(-5, 5), bins=40, lw=2)
plt.legend(["sig gen", "bkg gen", "sig mc", "bkg mc"], loc="upper left")
plt.xlabel("$m_{bb}$ logit normal space")
plt.show()
```



```
break
```

In [28]:

```
break
```

```
  File "/tmp/ipykernel_166809/668683560.py", line 1
    break
    ^
SyntaxError: 'break' outside loop
```

Plot all distributions

change rescale_data to False and use None rescaling instead of logit_normal for this

In [ ]:

```
from ml_hep_sim.stats.stat_plots import N_sample_plot
from ml_hep_sim.data_utils.higgs.process_higgs_dataset import LATEX_COLNAMES
```

In [ ]:

```
sig_gen = pipe.pipes[1].reference_data.cpu().numpy()
bkg_gen = pipe.pipes[3].reference_data.cpu().numpy()
sig_mc = pipe.pipes[7].generated_data[: len(sig_gen)].cpu().numpy()
bkg_mc = pipe.pipes[10].generated_data[: len(sig_gen)].cpu().numpy()
```

In [ ]:

```
BIN_RANGES = [
    [0, 4],
    [-3, 3],
    [-0.1, 4],
    [0, 5],
    [-4, 4],
    [0, 4],
    [-5, 5],
    [0, 5],
    [-4, 4],
    [0, 5],
    [-3, 3],
    [0, 3],
    [0, 3],
    [0.75, 1.5],
    [0, 3],
    [0, 3],
    [0, 3],
    [0, 3],
]
```

In [ ]:

```
fig, axs = plt.subplots(6, 3, figsize=(13, 19))
axs = axs.flatten()

res = [sig_gen, bkg_gen, sig_mc, bkg_mc]

N_sample_plot(res, axs, n_bins=40, log_scale=False,
              labels=LATEX_COLNAMES, lw=2, alpha=1,
              label=["sig gen", "bkg gen", "sig mc", "bkg mc"],
              xlim=BIN_RANGES, bin_range=BIN_RANGES)
plt.tight_layout()
```

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js