In [1]:

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

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

Get cut pipeline

In [2]:

```
from ml_hep_sim.analysis.cut_pipeline import get_cut_pipeline
```

In [3]:

```
cut_pipeline = get_cut_pipeline(cut_value=0.5, use_classifier=True) # this is different classifier...
```

Extract variable

In [4]:

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

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle

from ml_hep_sim.plotting.style import style_setup, set_size

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

In [5]:

```
saved = "ml_hep_sim/analysis/results/hists/"
```

In [6]:

```
var = "m bb"

dct = get_colnames_dict()
idx = dct[var]

b_sig_gen_data_cut, b_bkg_gen_data_cut, b_sig_mc_data_cut, b_bkg_mc_data_cut = cut_pipeline.pipes[-4:]

b_sig_gen_var = VariableExtractBlock(idx, save_data=False)(b_sig_gen_data_cut)
b_bkg_gen_var = VariableExtractBlock(idx, save_data=False)(b_bkg_gen_data_cut)
b_sig_mc_var = VariableExtractBlock(idx, save_data=False)(b_sig_mc_data_cut)
b_bkg_mc_var = VariableExtractBlock(idx, save_data=False)(b_bkg_mc_data_cut)
```

```
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}
```

Make histograms

In [7]:

```
from ml_hep_sim.analysis.utils import SigBkgBlock
from ml_hep_sim.analysis.hists_pipeline import  MakeHistsFromSamples
```

In [8]:

```
b_sig_bkg_gen_mc = SigBkgBlock(b_sig_gen_var, b_bkg_gen_var, b_sig_mc_var, b_bkg_mc_var)(b_sig_gen_var, b_bkg_g
```

In [9]:

```
bins = 40
N_sig = 100
N_bkg = 1000

N_gen = 10**6

use_class = False
```

In [10]:

```
saved = "ml_hep_sim/analysis/results/hists/"
```

In [11]:

```
if use_class:
    saved += "class_"
else:
    saved += "mbb_"
```

In [12]:

```
saved
```

Out[12]:

```
'ml_hep_sim/analysis/results/hists/mbb_'
```

Build and fit pipeline

In [13]:

```
from ml_hep_sim.analysis.hists_pipeline import get_hists_pipeline # TODO: write same for var
from ml_hep_sim.pipeline.pipeline_loggers import setup_logger

logger = setup_logger(log_name="hists", log_path="ml_pipeline/")
```

In [14]:

```
if use_class:
    bin_range = (0.5, 1.1)

    pipe = get_hists_pipeline(var="",
                  bin_range=bin_range,
                  bins=bins,
                  N_sig=N_sig,
                  N_bkg=N_bkg,
                  N_gen=N_gen,
                  logger=logger,
                  use_classifier=True)
    pipe.fit()

else:
    bin_range = (0.01, 3.0)

    b_hists = MakeHistsFromSamples(bin_range=bin_range, bins=bins, N_sig=N_sig, N_bkg=N_bkg, N_gen=N_gen)(b_sig_

    pipe = Pipeline()
    pipe.compose(
        cut_pipeline,
        b_sig_gen_var,
        b_bkg_gen_var,
        b_sig_mc_var,
        b_bkg_mc_var,
        b_sig_bkg_gen_mc,
        b_hists,
    )
    pipe.fit()
```

```
100%|███████████████████████████████████████████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████████████████████████████████████████
███████████████████████████████████████████████████████████████████████████████| 10/10 [00:05<00:00,  1.87it/s]
100%|███████████████████████████████████████████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████████████████████████████████████████
███████████████████████████████████████████████████████████████████████████████| 10/10 [00:05<00:00,  1.87it/s]
100%|███████████████████████████████████████████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████████████████████████████████████████
█████████████████████████████████████████████████████████████████████████████| 10/10 [00:00<00:00, 5878.49it/s]
100%|███████████████████████████████████████████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████████████████████████████████████████
█████████████████████████████████████████████████████████████████████████████| 10/10 [00:00<00:00, 5683.34it/s]
100%|███████████████████████████████████████████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████████████████████████████████████████
██████████████████████████████████████████████████████████████████████████████| 10/10 [00:00<00:00, 354.05it/s]
100%|███████████████████████████████████████████████████████████████████████████████████████████████████████████
████████████████████████████████████████████████████████████████████████████████████████████████████████████████
██████████████████████████████████████████████████████████████████████████████| 10/10 [00:00<00:00, 359.30it/s]
```

In [15]:

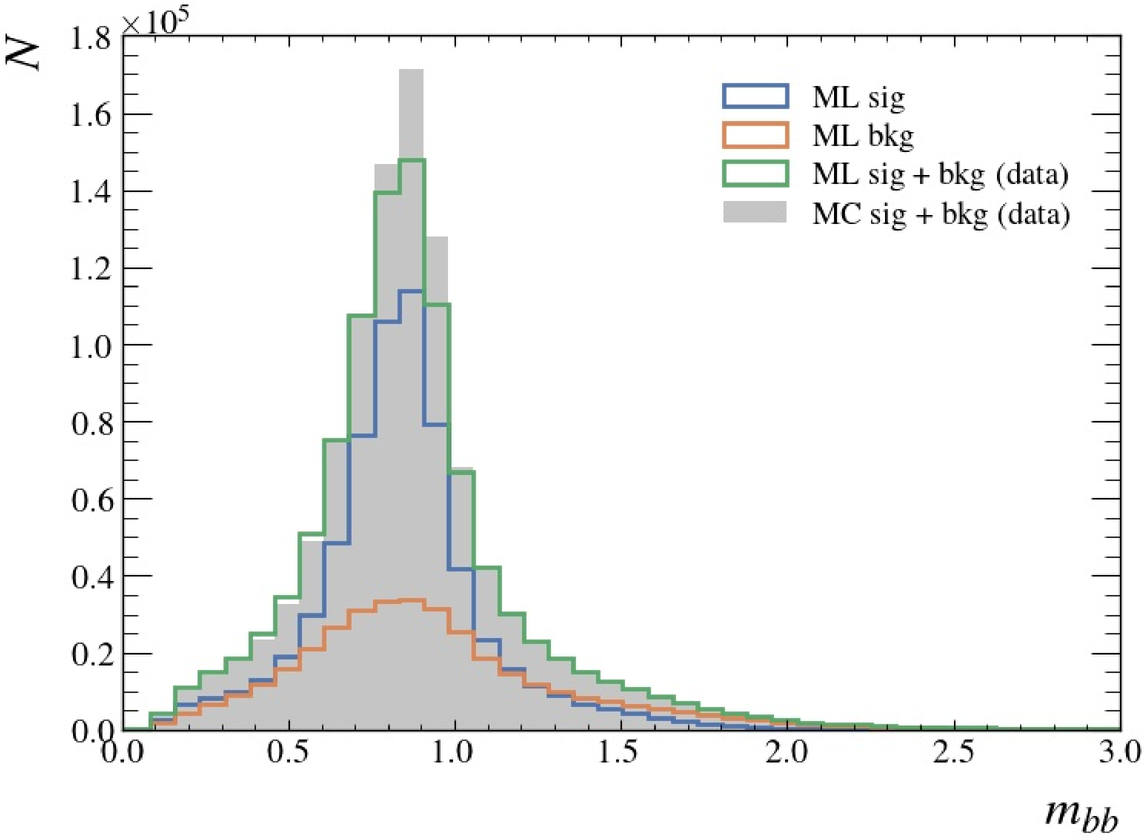
```
# pipe.draw_pipeline_tree(to_graphviz_file="pipeline_gen_cut", block_idx=-1)
```

In [16]:

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

In [17]:

```
plt.hist(sig_gen, range=bin_range, bins=bins, histtype="step", lw=2)
plt.hist(bkg_gen, range=bin_range, bins=bins, histtype="step", lw=2)
plt.hist(np.concatenate([sig_gen, bkg_gen]), range=bin_range, bins=bins, histtype="step", lw=2)
plt.hist(np.concatenate([sig_mc, bkg_mc]), range=bin_range, bins=bins, histtype="stepfilled", color="C7", alpha=
plt.legend(["ML sig", "ML bkg", "ML sig + bkg (data)", "MC sig + bkg (data)"], fontsize=15)
plt.ylabel("$N$", fontsize=24)
plt.xlabel("$m_{bb}$" if not use_class else "class. output", fontsize=24)

plt.tight_layout()
plt.savefig(saved + "post_cut_dists.pdf")
```



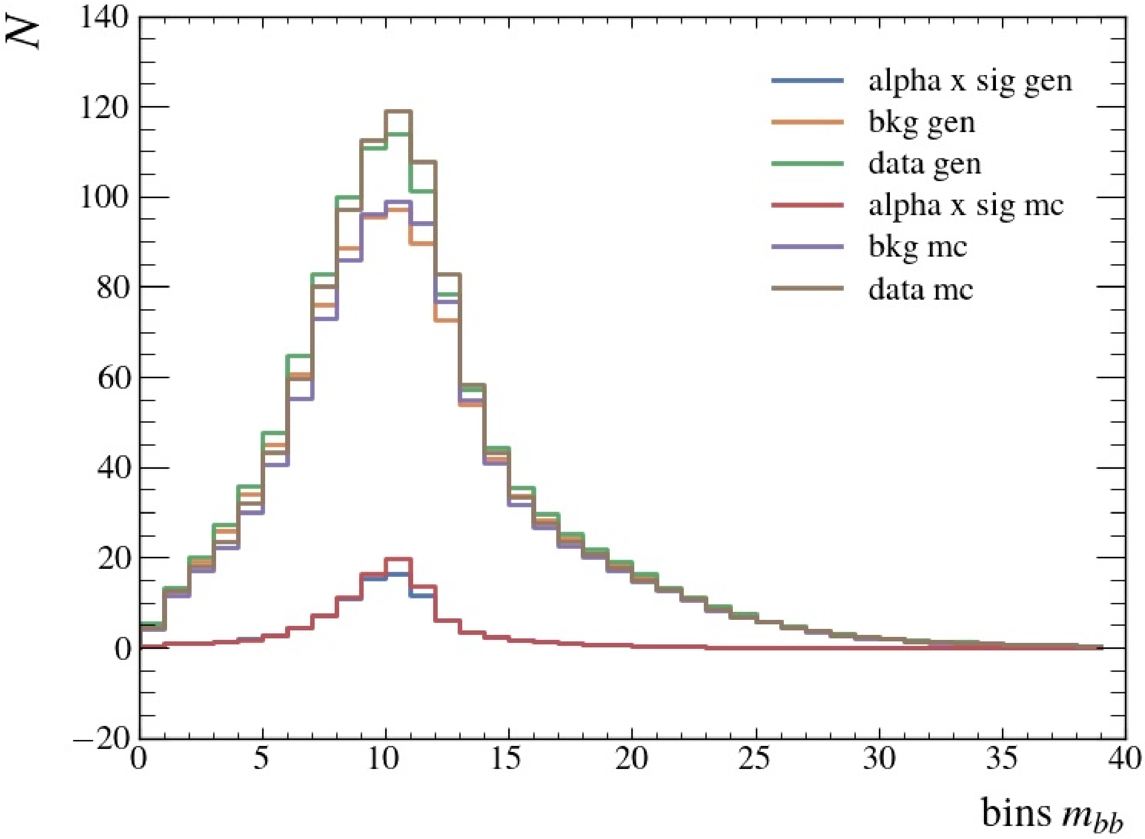
In [18]:

```
p = pipe.pipes[-1]
hists = p.histograms
errors = p.errors
alpha = p.alpha
N = p.N_gen

r = range(bins)
```

Steps

In [19]:

```
plt.step(r, alpha * hists["sig_gen"], label="alpha x sig gen")
plt.step(r, hists["bkg_gen"], label="bkg gen")
plt.step(r, hists["data_gen"], label="data gen")

plt.step(r, alpha * hists["sig_mc"], label="alpha x sig mc")
plt.step(r, hists["bkg_mc"], label="bkg mc")
plt.step(r, hists["data_mc"], label="data mc")

plt.xlabel("bins $m_{bb}$")

plt.legend()

plt.ylabel("$N$", fontsize=24)

plt.tight_layout()
```

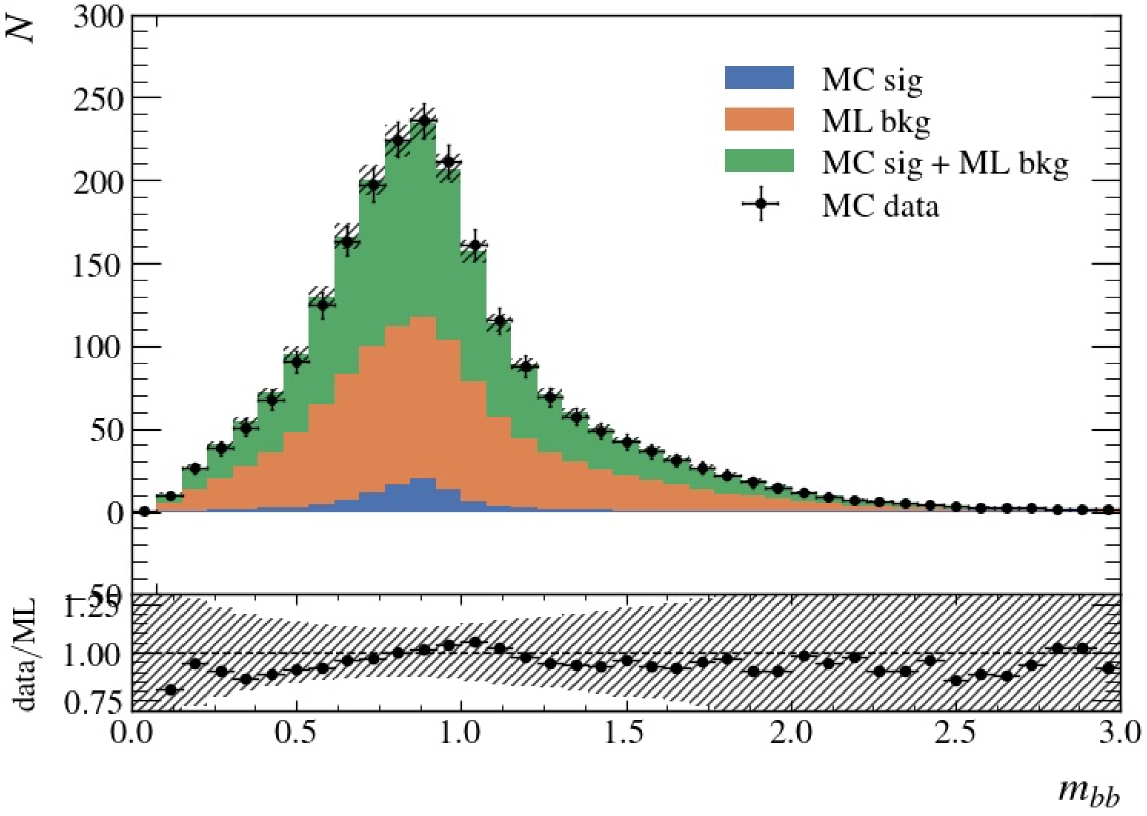


In [20]:

```
sys_err = 0.1

data_err = errors["data_mc"]
bkg_err = np.sqrt(errors["nu_b_ml"] ** 2 + (hists["bkg_gen"] * sys_err)**2)
```

In [21]:

```
from ml_hep_sim.plotting.hep_plots import StackPlot


x = np.arange(0, bins, 1)

sp = StackPlot(
    x,
    hists_lst=[alpha * hists["sig_mc"], hists["bkg_gen"], alpha * hists["sig_mc"] + hists["bkg_gen"]],
    data_hist=alpha * hists["sig_mc"] + hists["bkg_gen"] + hists["data_mc"],
)

sp.plot_stack(labels=["MC sig", "ML bkg", "MC sig + ML bkg"])

# data_err = np.sqrt(hists["sig_mc"] + hists["bkg_mc"])
sp.plot_data(label="MC data", err=data_err, fmt='.', capsize=1, lw=1)

# eff = len(bkg_gen) / N
# N_hist, _ = np.histogram(bkg_gen, bins=bins, range=bin_range)
# bkg_err = np.sqrt(N_hist * eff * (1 - eff))

sp.plot_mc_errors(bkg_err)

counts_num, counts_den = hists["data_mc"], alpha * hists["sig_mc"] + hists["bkg_gen"]
counts_num_err = data_err
counts_den_err = bkg_err

sp.plot_lower_panel(counts_num, counts_den, counts_num_err, counts_den_err, ylabel="data$/$ML",
                    label_x_start=bin_range[0],
                    label_x_end=bin_range[1])

ax = sp.ax
ax_lower = sp.ax_lower

if use_class:
    ax_lower.set_xlabel("class. output")
else:
    ax_lower.set_xlabel("$m_{bb}$")

ax.set_ylabel('$N$')

ax_lower.set_ylim(0.7, 1.3)

plt.legend(loc='upper right')
plt.tight_layout()
plt.savefig(saved + "stacked.pdf")
```



```
In [ ]:
```

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