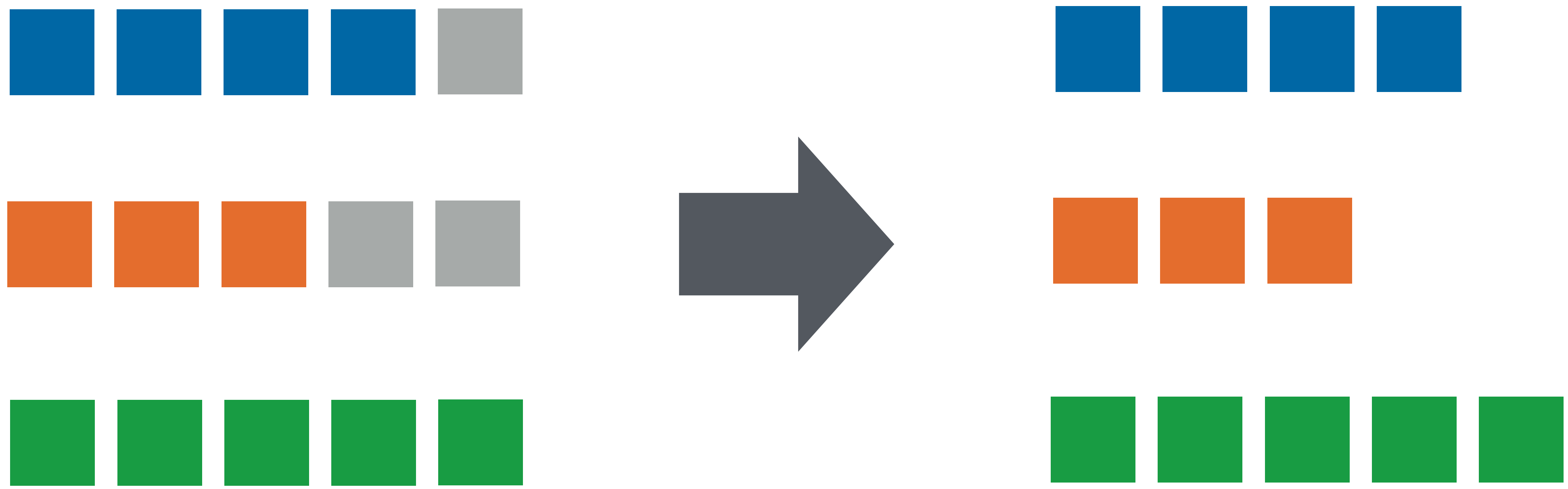


super-histograms: practical use-cases

-

Peter Fackeldey

IRIS-HEP

15.11.23

RWTH AACHEN UNIVERSITY

Histograms in a HEP analysis

- Dimensions of a typical histogram in a HEP analysis (1 quantity "m_{ll}"):
  - [**hist.axis.StrCat**] *Dataset*: O(1000) entries
  - [**hist.axis.StrCat**] *Category*: O(100) entries
  - [**hist.axis.StrCat**] *Systematic*: O(100) entries
  - [**hist.axis.Variable**] *Variable*: O(100) bins

- Total size is defined by the cartesian product:
  Dataset x *Category* x *Systematic* x *Variable* $\approx 10^9$ values → 8 GB (double prec.)

- Histograms become/are a bottleneck in terms of memory consumption for full-fledged HEP analyses!

- From my experience:
  - We had to split histograms up (e.g.: not every variable needs every systematic)
    → tree-like structure of individual histograms with *non*-contiguous data
  - We had to re-run the full analysis just to fill a different quantity of interest...
    ...re-running was fast, but it's far from ideal!

Sparsity

- Dimensionality reminder for 1 quantity:
  - [**hist.axis.StrCat**] *Dataset*: O(1000) entries
  - [**hist.axis.StrCat**] *Category*: O(100) entries
  - [**hist.axis.StrCat**] *Systematic*: O(100) entries
  - [**hist.axis.Variable**] *Variable*: O(100) bins

- Is this histogram sparse? Yes, just some examples:
  - No *Systematics* for data *Datasets* (extreme case: >100 PDF variations)
  - Only some *Variables* in some *Categories used* for e.g. ABCD estimations
  - Only some *Datasets* have dedicated *Systematics* (e.g.: t-quark p_T corr.)
  - Some *Variables* are only interesting for control plots in certain *Categories*
  - ...

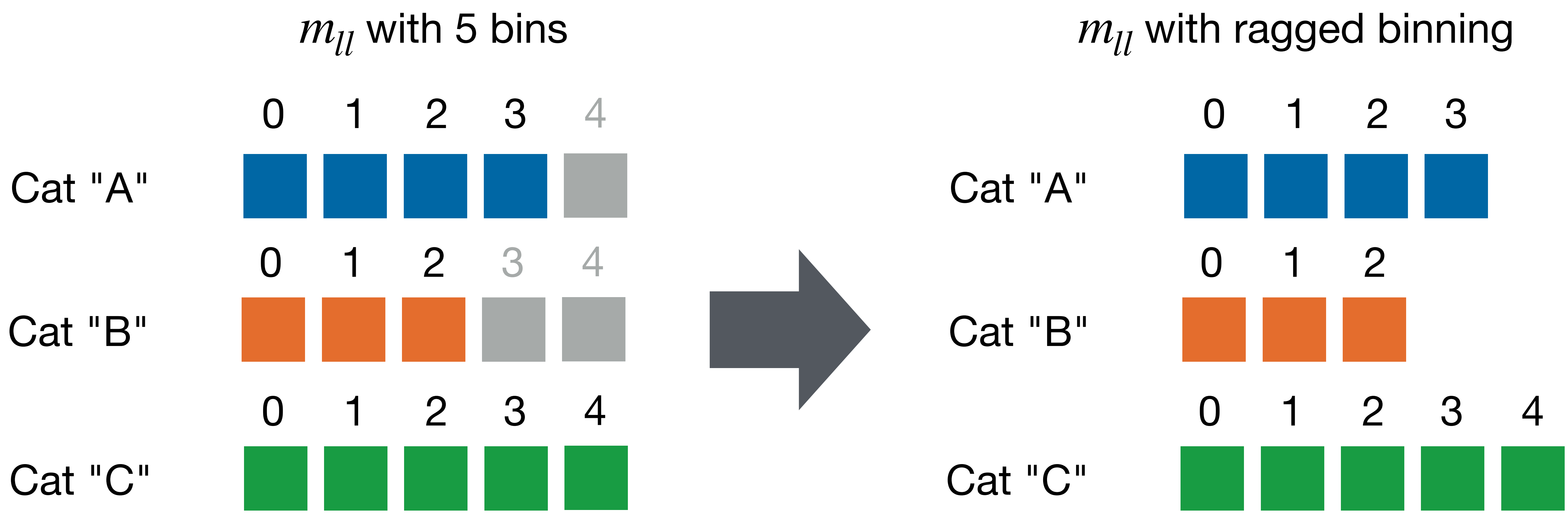
→ **Idea: "ragged/sparse" histograms (*super-histogram*)**

Ragged Histograms: Examples (1/3)

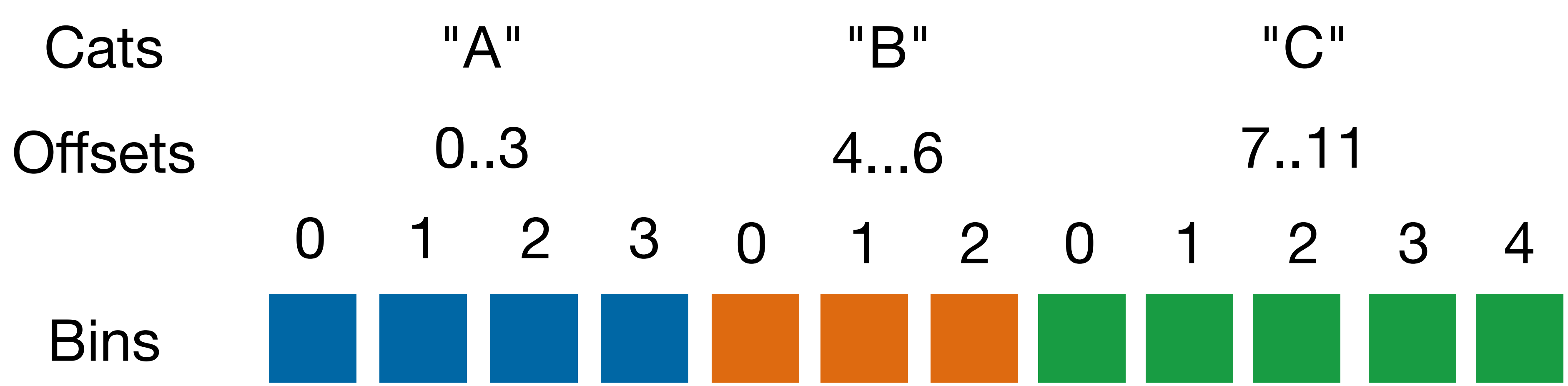
RWTHAACHEN UNIVERSITY
Peter F. - 15.11.23

Regular histogram

super-histogram

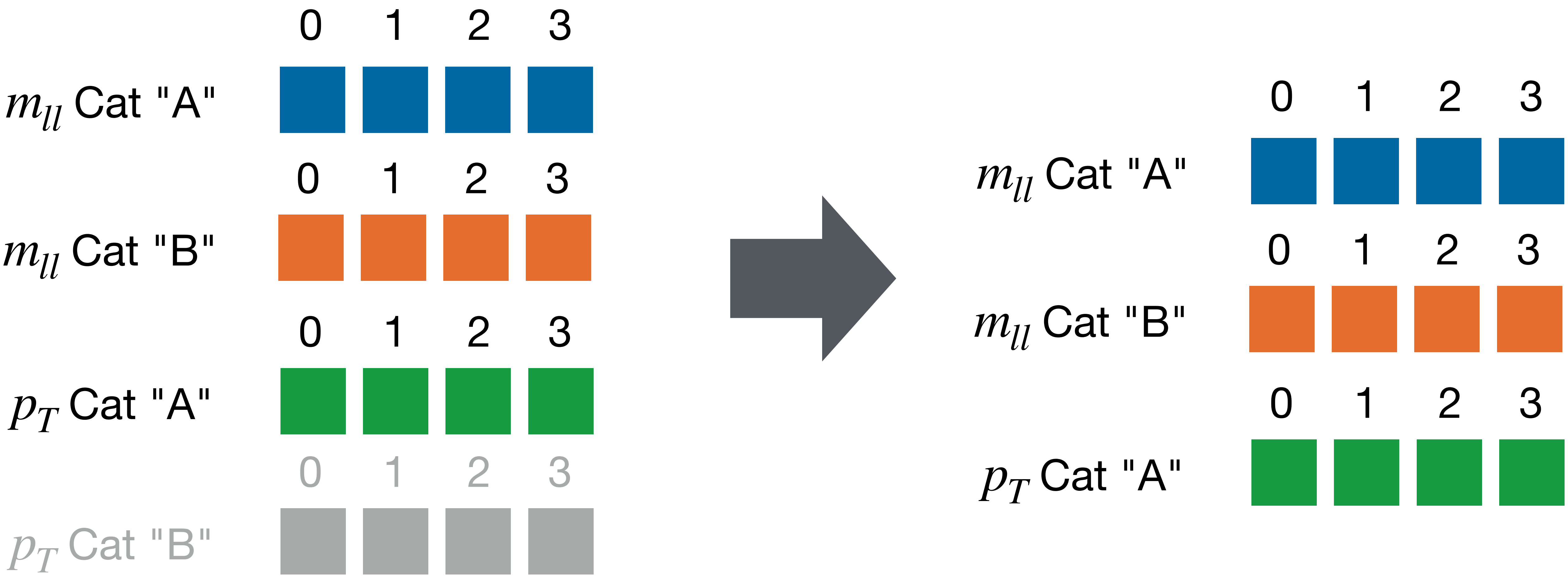


Internal layout of *super-histogram*

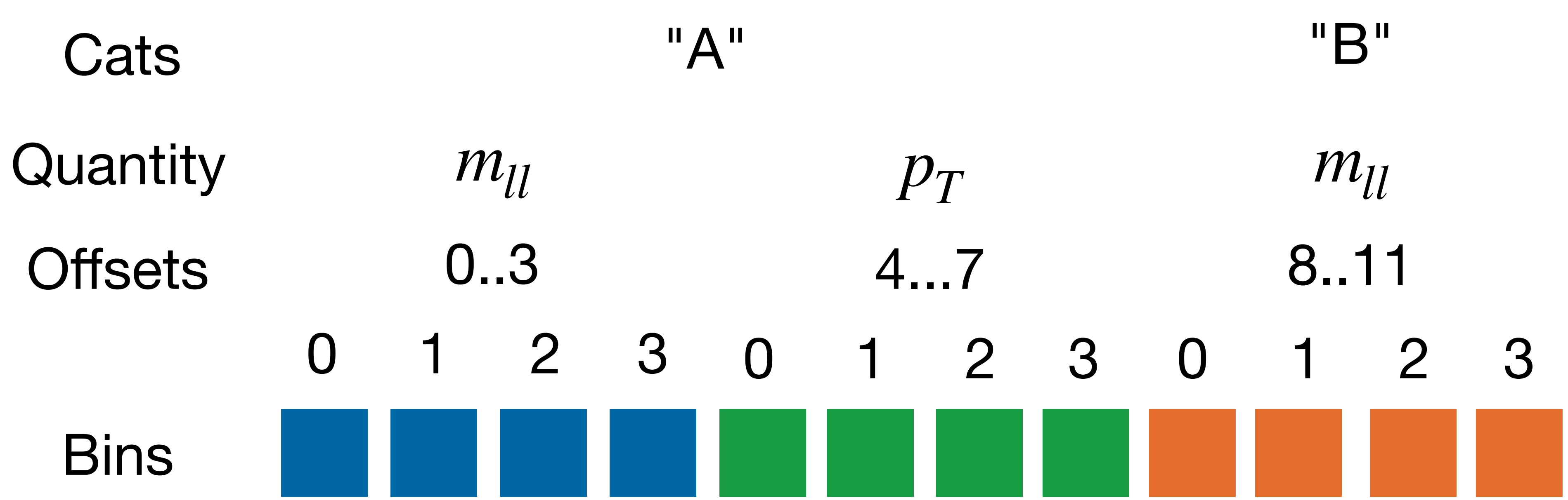


Ragged Histograms: Examples (2/3)

Regular histogram | *super-histogram*

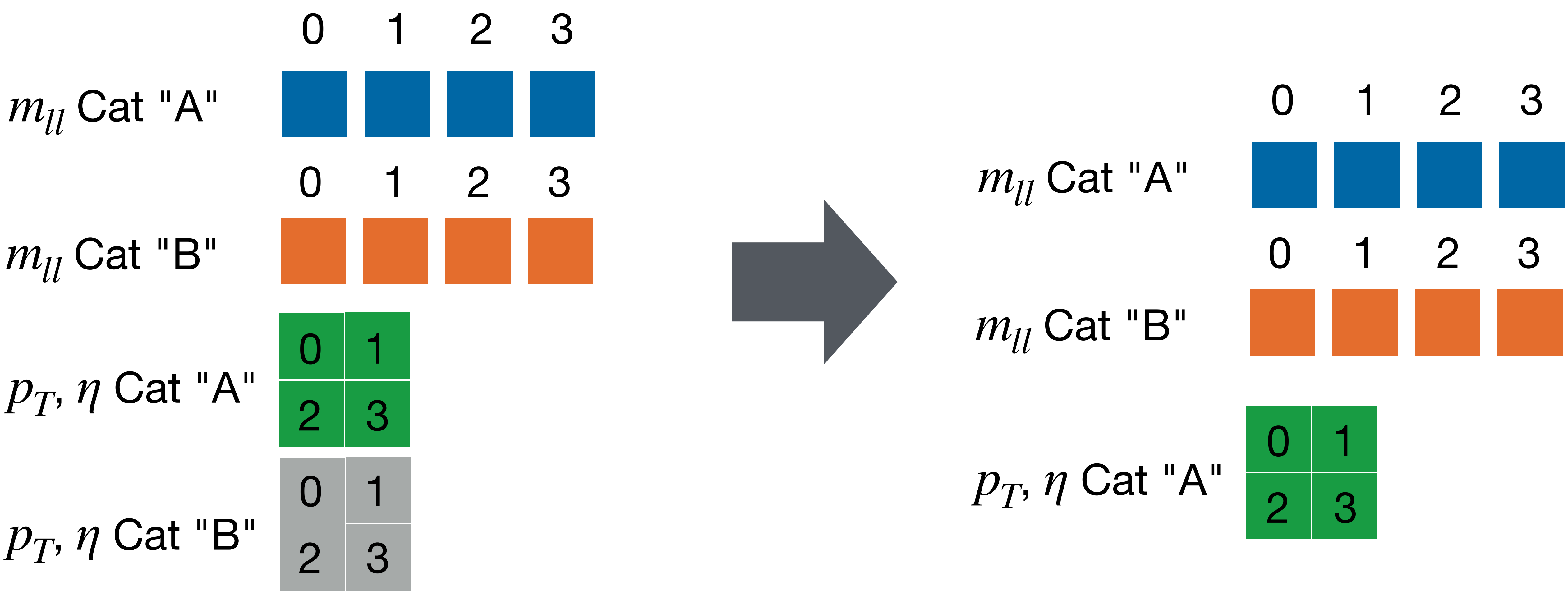


Internal layout of *super-histogram*



RWTH AACHEN UNIVERSITY
Peter F. - 15.11.23

Regular histogram — *super-histogram*



Internal layout of *super-histogram*

Cats	"A"		"B"
Quantity	m_{ll}	p_T, η	m_{ll}
Offsets	0..3	4...7	8..11
Bins	0 1 2 3	0 1 2 3	0 1 2 3

Implementation thoughts

- *super-histogram* should follow UHI

- Integrate with scikit-hep/hist
  → benefit from existing well-thought implementations & community-use

- Potential (lightweight) backend: only NumPy

- Potential backend: JAX (*super-histograms* as PyTrees?)
  - Usually: dynamic slices ↔ JAX ϟ ..but we know all offsets at compile-time!
  - (Multi-)GPU accelerated filling?
  - JAX array-sharding for sharded *super-histograms*?
  - JAX-compatible histograms are automatically compatible with JAX-based fitting libraries

Conclusion

- Memory consumption of histograms become/are a bottleneck for HEP analyses
- Usually these histograms are sparse
- *super-histogram* implements ragged histograms to reduce the memory footprint (similar to ak.Array)
- *super-histogram* exploits semiring properties for improved serialization (see Jim's Talk)
- 3 Implementation ideas:
  - scikit-hep/hist
  - NumPy (optional), benefit: lightweight
  - JAX (optional), benefit: GPU-filling, data sharding, compatibility with JAX fitting libraries