Skills for physics analysis in DrSai v3

Tong Liu for the DrSai group

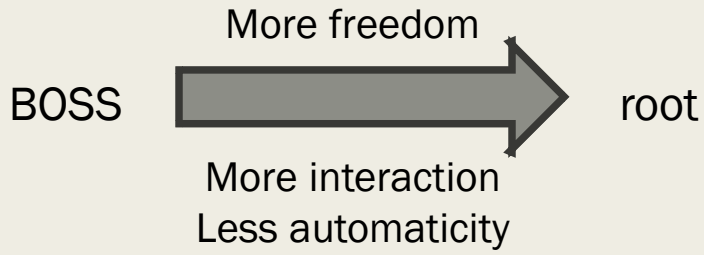
https://pan.quark.cn/s/a61916086d83

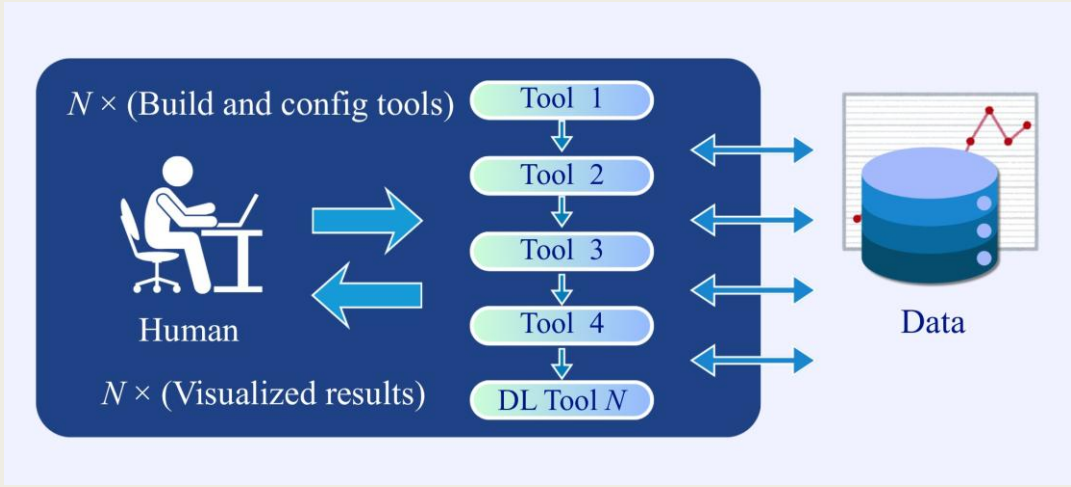
Outline

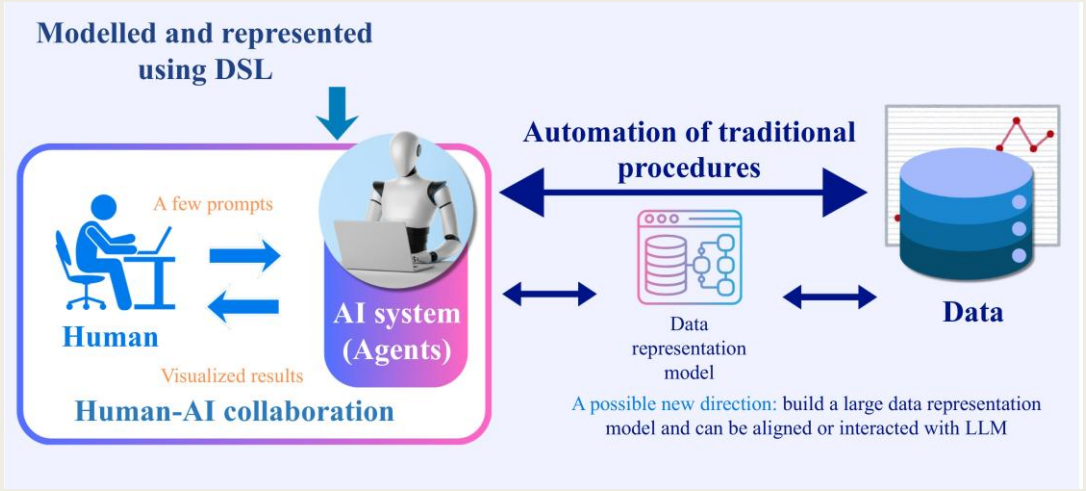
- Motivation
- Workflow Based on Skills
- How to write a skill

Motivation

- DrSai v3 is released: General-purpose Agent + DSL + Skills + RAG
  - Smart General-purpose Agent
  - Use a DSL to represent the analysis pipeline
  - RAG to suggest strategy
  - GraphRAG to improve generation
- Everything is generated/read/used with **skills**
- Only human-AI interaction, no manual coding, we should know what is happening

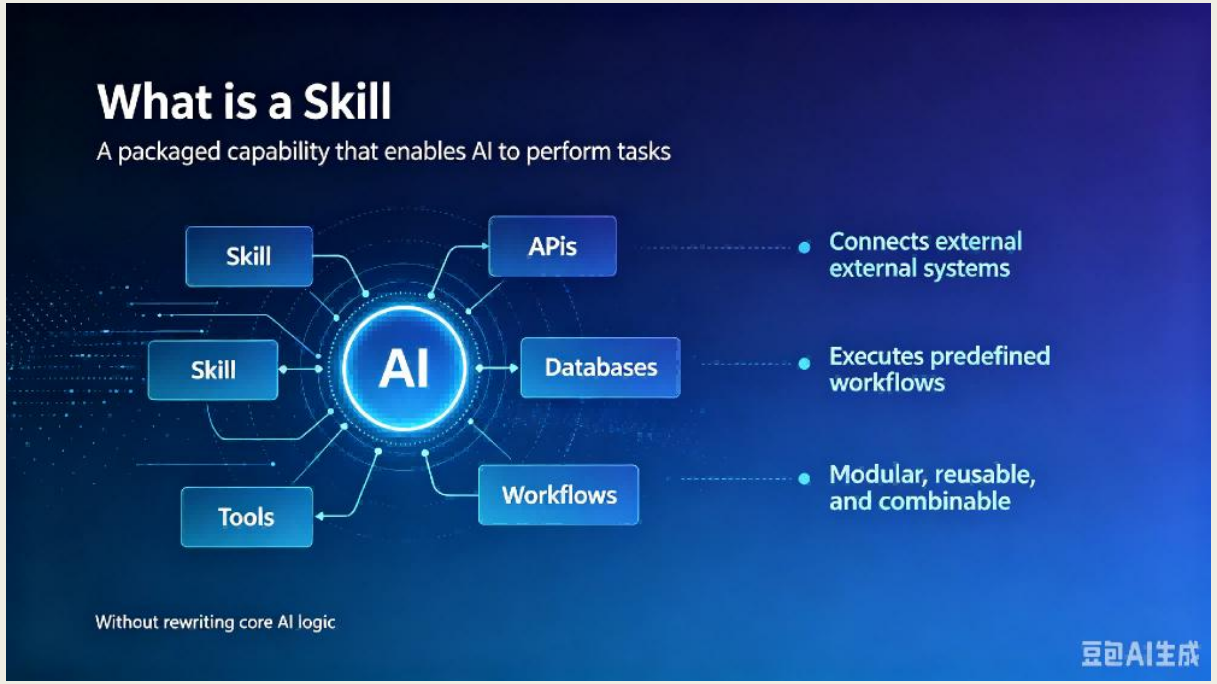






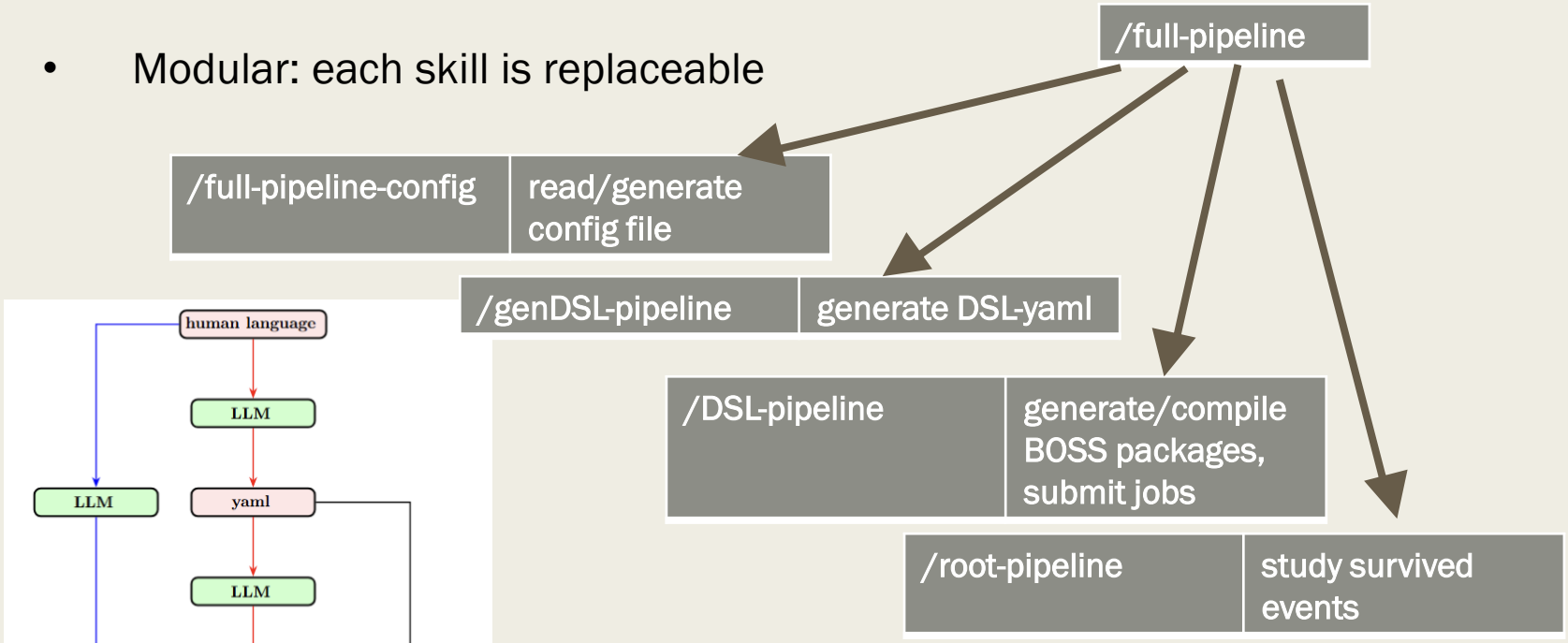
Motivation

- A Skill is a packaged capability that gives AI the power to perform tasks
- It connects AI to external systems (APIs, databases, tools) and executes predefined workflows
- Skills are **modular and reusable**, you can add, update, or combine them without rewriting core AI logic
- Skills employ **progressive disclosure**: they load only brief descriptions upfront, then pull full information only when a skill is triggered, therefore dramatically cutting context overhead.

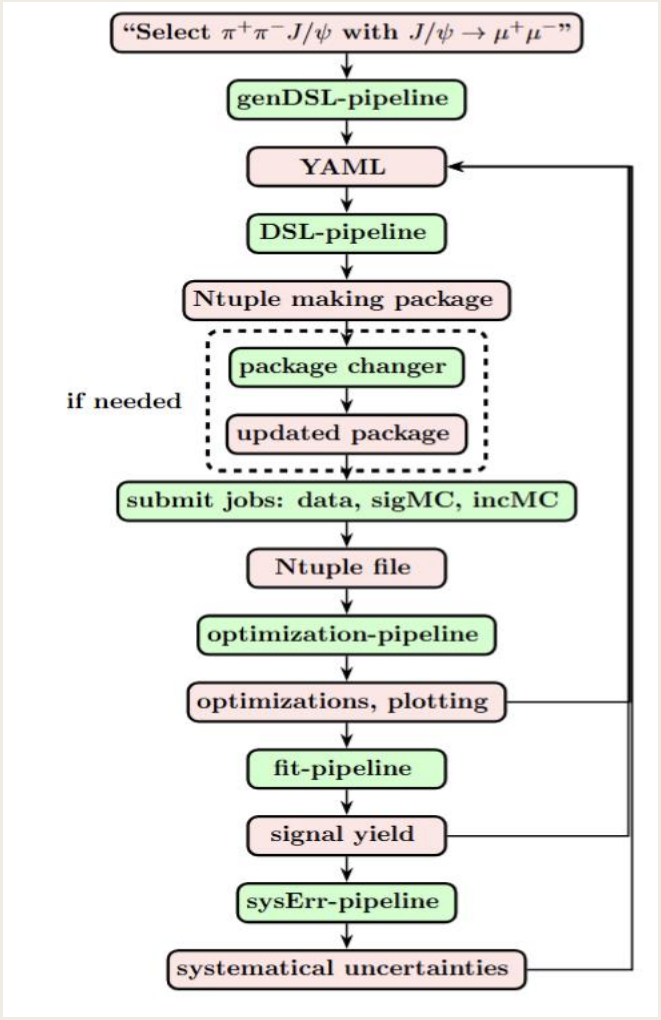


Workflow Based on Skills

- General-purpose Agent + DSL + Skills + RAG
- Three tires of skills
- Modular: each skill is replaceable



different BOSS package generating method



Skills

Input/Output

Workflow Based on Skills

流程总览

full-pipeline-config (阶段0)

▼

阶段1: genDSL-pipeline → DSL_<ww>.yaml

▼

阶段2: dsl-pipeline → BOSS 算法包 (basic → decay-card → [ruby|direct] → check-boss → confirm-selection → compile)

▼

阶段2a: BOSS 作业提交 (data → incMC → MC)

▼

阶段3: root-pipeline → ROOT 分析 (B1a→B1b→B1c→B2a→ROOT检查点
→B2b→B2c→B2d→B2e→B3a→B3b→B3c→B4a→B4b→B4c)

▼

阶段3.5: DSL-add-result → 物理结果计算

▼

阶段4: 系统误差分析 (rag-query / sysErr-rubyBased)

“analysis with DrSai, see config.yaml”

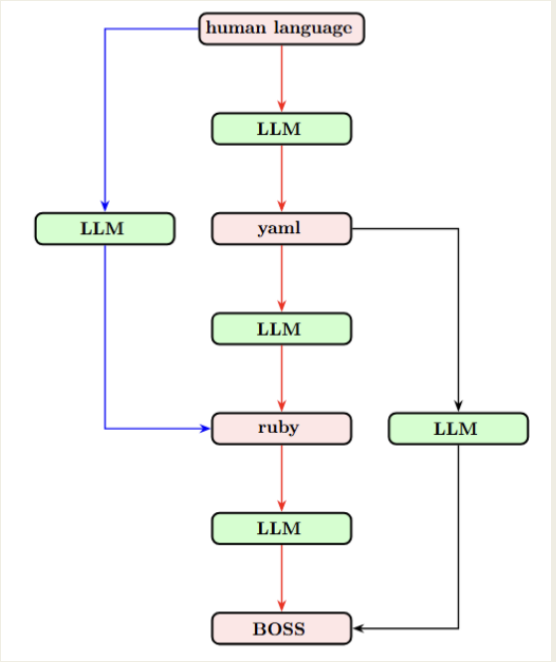
```
# ========== 全局字段 ==========
outdir: ./logR          # 必填。分析输出根目录
runBOSS_outdir: ./logR/proj # 阶段2a BOSS 作业项目根目录。缺省 → 交互询问
workarea: /workfs2/bes/liutong2016/boss713/workarea/Analysis/R # BOSS 编译目标 workarea。缺省 → 运行时询>问

confirm_channel: terminal       # terminal(终端弹窗) | wechat(微信远程确认)。步骤8等人工确认环节的交互渠道。缺省 → terminal
confirm_mode: minimal       # detailed(每步都问) | minimal(仅必须项交互)
parallel: yes               # yes | no。是否启用 Agent 并行执行
sysErr: yes                 # yes | no。是否RAG生成系统误差代码（阶段4）。缺省 → 运行时询问
sysErr-ruby: yes            # yes | no。使用针对 HepScript 开发的系统误差分析benchmark 分析系统误差。缺省 → no
genBOSSCode: direct     # ruby | direct。BOSS 代码生成方式。
                #   ruby   — Ruby/HepScript 生成 BOSS 代码
                #   direct — 直接按 BOSS-event-selection / BOSS-Dtag-selection 模板生成

# ========== entries 模式与条目 ==========
mode: single                # single | multi。必填（entries 仅 1 行时可省略，自动 = single）
                    #   single — 一个分析：所有 entries 共享 ww
                    #   multi — 多个独立分析：每个 entry 的 process 自动为独立 ww
ww: R               # 可选。仅 mode=single 且 entries≥2 时有效。
entries:
  # 每行格式：  process   threshold_cut=<GeV>   描述
  - "dimu     threshold_cut=0.2113     select mu+mu- at 2.0 GeV"
  - "KSKppim        threshold_cut=1.130850  select K_S0 Kp pim, K_S0->pip pim at 2.0 GeV"

# ========== 数据和 incMC 信息（可选）==========
# 取消注释并填入实际值，跳过交互询问。填"跳过" 跳过该类样本。
# 不填（保持注释）→ 运行时交互询问。

data_dstlist: /path/to/data/dstlist
incmc_dstlist: /path/to/incMC/dstlist
dst_keyword: "2000"           # 文件名筛选关键字，空格分隔多个，"all" 不过滤
data_list: /path/to/data_list.txt # 数据点信息文件，含能量、能散等，LLM 自行读取解析
```

1: generate DSL-yaml

- DSL as work"note" :
  - metadata: brief description of the work, decay process
  - analysis_i: each individual process
    - base_selection: BOSS selection
    - further_selection: further selection with ROOT
    - fit: (optional) single-channel fit
  - result: multi-channel combination (calculation, combined fit, etc.)
  - systematic_uncertainties: systematic uncertainties
- Interactive workflow, revisable at any time manually or via LLM
- Finally obtain "notes" that contain all the information
  - Write MEMO automatically

2: generate BOSS packages

- Different BOSS package generating method are provided:
  - Similar performence and speed
  - If (parallel: yes), use subagents to speed up
  - data, incMC, MC jobs are submitted then
    - cutflow in BOSS-selection is stored

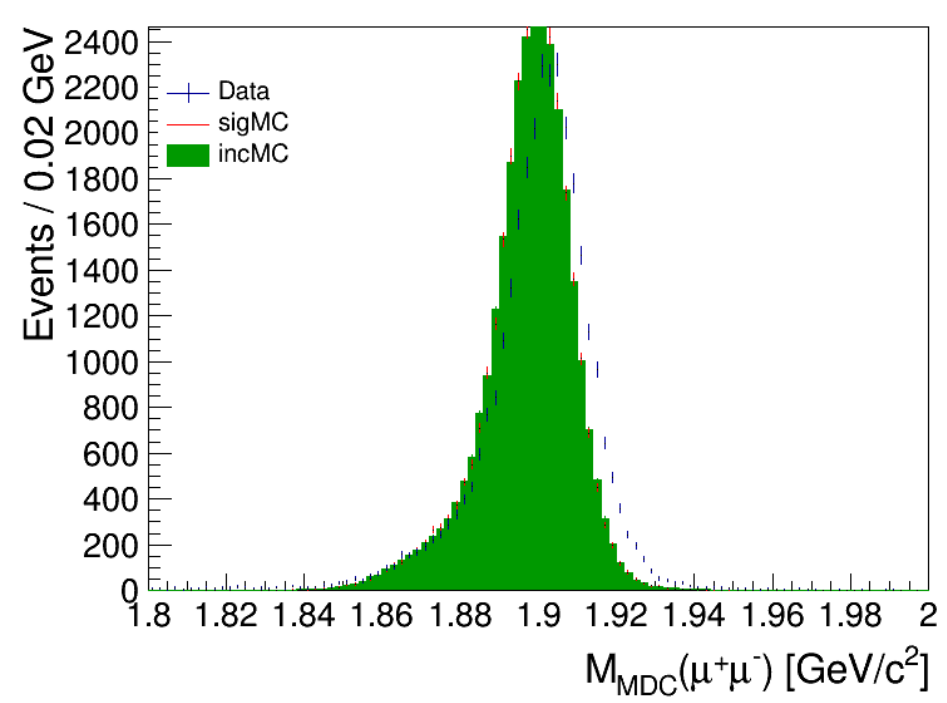



```
› ▌

▸▸ auto mode on · 25 local agents · ↓ to manage

● main                                                    ↑/↓ to select · Enter to view
  general-purpose  BOSS code gen: bhabha (j2=1)                  6m 52s · ↓ 69.0k tokens
  general-purpose  BOSS code gen: digam (j2=2)                    6m 3s · ↓ 73.4k tokens
  general-purpose  BOSS code gen: dimu (j2=3)                     6m 3s · ↓ 75.8k tokens
  general-purpose  BOSS code gen: eeee (j2=4)                     6m 3s · ↓ 61.9k tokens
  general-purpose  BOSS code gen: eeeta (j2=5)                    6m 3s · ↓ 53.4k tokens
  general-purpose  BOSS code gen: eeetap (j2=6)                   6m 3s · ↓ 50.9k tokens
  general-purpose  BOSS code gen: eekk (j2=7)                    5m 17s · ↓ 31.8k tokens
  general-purpose  BOSS code gen: eepipi (j2=8)                  5m 17s · ↓ 51.8k tokens
  general-purpose  BOSS code gen: eemumu (j2=9)                  5m 17s · ↓ 54.7k tokens
  general-purpose  BOSS code gen: pi0 gamma (j2=10)              5m 17s · ↓ 56.5k tokens
  general-purpose  BOSS code gen: eta gamma (j2=11)              5m 17s · ↓ 55.8k tokens
  general-purpose  BOSS code gen: pi+pi- (j2=12)                 5m 17s · ↓ 47.3k tokens
  general-purpose  BOSS code gen: pi+pi-pi0 (j2=13)              4m 22s · ↓ 48.4k tokens
  general-purpose  BOSS code gen: pi+pi-pi+pi- (j2=14)           4m 22s · ↓ 49.9k tokens
  general-purpose  BOSS code gen: pi+pi-pi0pi0 (j2=15)           4m 22s · ↓ 38.5k tokens
  general-purpose  BOSS code gen: pi+pi-pi+pi-pi0 (j2=16)        4m 22s · ↓ 39.7k tokens
  general-purpose  BOSS code gen: pi+pi-3pi0 (j2=17)             4m 22s · ↓ 35.7k tokens
  general-purpose  BOSS code gen: 6pi (j2=18)                    4m 22s · ↓ 50.6k tokens
  general-purpose  BOSS code gen: 4pi2pi0 (j2=19)                3m 18s · ↓ 11.5k tokens
  general-purpose  BOSS code gen: pipi4pi0 (j2=20)               3m 18s · ↓ 30.3k tokens
  general-purpose  BOSS code gen: omega eta pi0 (j2=21)          3m 18s · ↓ 38.3k tokens
  general-purpose  BOSS code gen: K+K- (j2=22)                   3m 18s · ↓ 29.2k tokens
  general-purpose  BOSS code gen: K_S0 K_L0 (j2=23)              3m 18s · ↓ 46.8k tokens
  general-purpose  BOSS code gen: KKpipi (j2=24)                 3m 18s · ↓ 31.5k tokens
  general-purpose  BOSS code gen: KKpi (j2=25)                   3m 18s · ↓ 49.4k tokens
```

process	total	nCharge	PID	kmfit	final	survived
dimu (signal)	450,000	90.8%	91.7%	77.1%	64.3%	289,270
bhabha	6,603,119	40.1%	0.7%	71.4%	0.20%	13,477
Hadron_Hybrid	460,000	40.5%	7.7%	12.5%	0.39%	1,793
eeuu	500,000	23.2%	2.5%	7.0%	0.04%	203
digam	1,181,820	1.8%	0.3%	0%	0%	0
eeeta	50,000	9.6%	0.8%	0%	0%	0
eeetap	50,000	35.6%	2.0%	0%	0%	0
eekk	50,000	29.6%	1.9%	0%	0%	0

3: plotting / optimizing / topology

- Topology automically, BG suppression suggestions are given
  - By default incMC is scaled according to maximum
  - Signal is not removed



排名	事例数	占比	末态
1	7,842	50.7%	e^+e^-
2	4,288	27.7%	$e^+e^-\gamma$
3	1,653	10.7%	$\pi^+\pi^-\gamma$
4	1,141	7.4%	$e^+e^-\gamma\gamma$
5	178	1.2%	$e^+e^-\gamma\gamma\gamma$
6	139	0.9%	$\pi^+\pi^-\gamma\gamma$
7-10	~223	~1.4%	ψ(4260)→μe/cc̄...

Key Findings

- Bhabha ~86% (e^+e^- + $e^+e^-\gamma$ + $e^+e^-\gamma\gamma$…) – electrons are misidentified as muons because their E/p is mistakenly taken as that of a muon (muons have E/p < 0.4), and electron energy loss sometimes overlaps, especially at low energies.

- $\pi^+\pi^-\gamma$ ~10.7% – two charged pions plus a photon are misidentified as $\mu^+\mu^-$ (PID confusion).

Suppression Suggestions

- Stricter E/p cut – the current eop < 0.4 may be too loose for 2.0 GeV electrons; consider tightening to eop < 0.3 or combining with MUC depth for joint identification.

- MUC information – use mucDepth and mucNumHits to further distinguish muons from pions.

- cosOpenAngle – a back-to-back requirement can help suppress backgrounds with photons.

How to write a skill

- Manually is not suggested, ask AI to do that
  - Please check carefully the descriptions! For trigger…
- Eg:
  - “Create a Skill named test_skill that replicates the functionality of run.sh.”
  - “pack1 and pack2 are functionally identical but use different framework versions. Write a Skill that converts packages from one version to the other.”
  - “Read BOSS packages in ./packages, write a skill that generates new packages”
    - Iteration: (1) read packages from ./more_packages and store their targets in ./target; (2) use /BOSS_gen to generate packages; (3) compare old vs new packages, if they differ, adjust the skill to reproduce the original behavior.”

BACKUP

阶段 0：配置生成

- full-pipeline-config — 入口。对话收集分析需求 → 生成 YAML 配置文件；或解析已有 YAML。输出 ---CONFIG_BEGIN---/---END--- 块供下游

阶段 1：genDSL-pipeline — 生成完整 DSL YAML

- genDSL-pipeline — 总入口。从分析描述生成完整 DSL YAML（A→B→C→D 全流程）
- genDSL-selection — 步骤 A。从分析描述直接生成 DSL selection YAML（挑选策略）
- genDSL-fit — 步骤 B。从 selection 输出生成拟合策略 YAML（fit 节）
- genDSL-full-yaml — 步骤 C。合并 selection + fit + NumCount 为完整 DSL YAML
- genDSL-check-yaml — 步骤 D。校验完整 DSL YAML 格式和内容一致性

跨阶段/通用工具

- isr-correction — ISR 修正因子 (1+δ) 计算（Kuraev-Fadin 结构函数）
- use_wechat_oneTime — 一次性微信问答（远程人工确认）
- use_wechat-warmup — 微信通道预热/验证
- dsl-pause — 保存 DSL 工作流暂停状态
- rag-query — 通用 RAG 查询（BESIII 物理分析问题）
- full-pipeline-withYAML — 使用已有 DSL YAML 完全复现分析流程，自动检测跳过已完成阶段
- full-pipeline-addMCProcess — 在已有分析中追加新信号过程
- _shared — 内部共享片段（文件列表更新、包名解析等，不对外暴露）

阶段 2：dsl-pipeline — BOSS 算法包生成

2a. 通用前置步骤（始终串行）
- dsl-pipeline — 总入口（步骤 1-9）。从 DSL YAML 生成可编译的 BOSS 算法包
- dsl-gen-basic — 步骤 1。生成 basic thinking（分析思路文档）
- dsl-gen-decay-card — 步骤 2。生成衰变卡（decay card）

2b. genBOSSCode=ruby 路线（步骤 3~7）
- dsl-pipeline-ruby — 子 skill 总入口。Ruby 路线 3~7 步的循环调度（串行/并行 Agent）
- dsl-gen-ruby — 步骤 3。生成 Ruby 选择脚本（三步流水线：生成→PID→校验合并）
- 步骤 4 (check-ruby-alg) — 内联，检查 ruby 算法包名
- 步骤 5 (run-ruby) — 内联，运行 ruby 生成 BOSS 代码
- 步骤 6 (cp-result) — 内联，复制 ruby 产出
- dsl-check-boss — 步骤 7。AI 代码检查+修复（⛔ 永远执行）
- description_2_Ruby — 独立工具。将物理分析描述翻译为 Ruby/HepScript 选择脚本

2c. genBOSSCode=direct 路线（步骤 3D~7D）
- dsl-pipeline-direct — 子 skill 总入口。Direct 路线 3D~7D 步的循环调度（串行/并行 Agent）
- BOSS-event-selection — 模板。通用 BOSS 事例挑选 6 步标准流程模板（非 D-tag）
- BOSS-Dtag-selection — 模板。D-tag 特有挑选流程模板（束流信息→DDmcmode→DTagTool→Match→NTuple）
- 步骤 3D — 内联，读模板 + DSL YAML 提取参数
- 步骤 4D — 内联，生成 .h 文件
- 步骤 5D — 内联，生成 .cxx 文件
- 步骤 6D — 内联，生成 requirements 文件
- 步骤 7D — 内联，策略输出 + check-boss

2d. 编译 & 确认（步骤 8~9）
- dsl-pipeline-userConfirmSelection — 步骤 8（⛔ 不可跳过）。子智能体并行读源码提取参数→ 父级串行逐 j2 用户确认 → 回填 YAML
- dsl-compile-boss — 步骤 9（始终串行）。复制到 workarea + cmt config + make 编译

2e. changeBOSS 系列（ruby 路线可选修改）
- changeBOSS_BOSS710 — BOSS 709 → 712 格式升级
- changeBOSS_toBOSS709 — BOSS 712 → 709 格式降级
- changeBOSS_BOSS8 — BOSS 709/713 → BOSS800 环境迁移
- changeBOSS_heff — 添加 heff cut-flow 直方图
- changeBOSS_LepUseEoP — 改为 E/p 轻子鉴别 + π/K/p 并行
- changeBOSS_SaveEmcInfo — 添加 RecEmcShower 信息保存
- changeBOSS_SaveMucInfo — 添加 RecMucTrack 信息保存
- changeBOSS_SaveMCTruth_WLJStyle — 添加 MCStore MC 真理信息保存

阶段 2a：BOSS 作业提交
- runBOSS_frontend — 前台运行 BOSS 作业脚本（测试用，EvtMax=100）
- runBOSS_backend_MC — 提交信号 MC 任务（sim → rec → ana 三阶段）
- runBOSS_backend_data-incMC — 批量生成 data/incMC job options 并提交批处理
- runBOSS_genSimJob — 选择产生子模式并生成 sim job options（BabayagaNLO/KKMC/ConExc/BesEvtGen/Ekhara）
- runBOSS_genConExcSimJob — 提交 ConExc ISR 产生子作业
- checkBOSS_jobs — 检查 BOSS 批处理作业完成状态并合并 ROOT 文件
- genBOSS-from-decay — 从衰变道描述直接生成 BOSS 算法包 + 最小 DSL YAML（可独立使用）

阶段 3：root-pipeline — ROOT 分析

3a. 生成期（Phase 1，不需要 ROOT 文件）

- root-pipeline — 总入口（B1~B4）。从 BOSS 输出 root 文件生成 ROOT 分析代码并运行
- 步骤 B1a — 内联，生成 branch list（从 .cxx/.h 提取变量类型）
- root-review-selection — 步骤 B1b。RAG 审查 further_selection 策略 + 用户确认（⛔ 不可跳过）
- 步骤 B1c — 内联，多重组合分析类型判定
- root-gen-opti — 步骤 B2a（串行版）。生成优化画图代码，逐文件串行运行，适合逐图把关
- root-gen-opti-parallel — 步骤 B2a（并行版）。N 个 (key,j) Agent 并行生成，适合大批量场景

3b. ROOT 就绪检查点（B2a 后强制，⛔ 不可跳过）

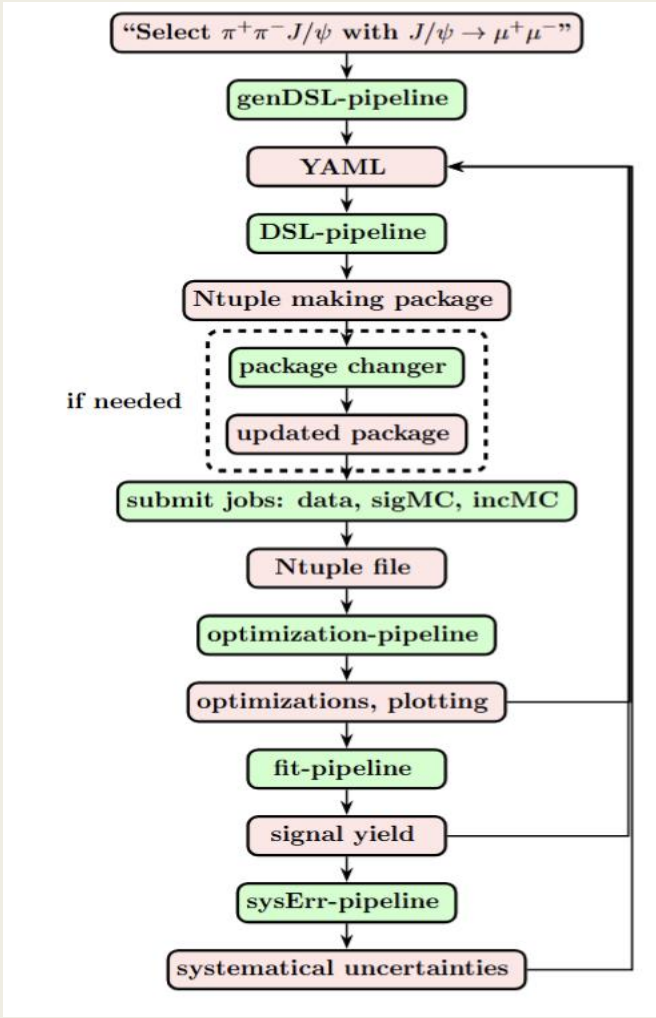
- checkBOSS_jobs — 合并 ROOT 文件 + 回填 YAML __TOBEFILLED__ → 真实路径

3c. 运行期（Phase 2，需要 ROOT 文件）

- 步骤 B2b — 内联，多轮迭代优化（copy→sed cut→root运行→确认）
- root-topo-review — 步骤 B2c（⛔ 不可跳过）。topoana 本底分析 + RAG 检索 → 用户确认是否新增 further_selection
- 步骤 B2d — 内联，最终 cut 确认（⛔ 不可跳过）
- root-cutflow — 步骤 B2e（⛔ 不可跳过）。完整 cut-flow 效率表（BOSS heff + ROOT further_selection）
- root-review-fit-result — 步骤 B3a（⛔ 不可跳过）。拟合策略审查 + RAG 建议 result 键值
- root-gen-fit — 步骤 B3b。生成 RooFit 拟合代码
- root-run-cxx — 步骤 B3c。运行 Fit_*.cxx（仅 draw_*，不跑 Fit_* / NumCount_*）
- 步骤 B4a — 内联，看图确定 NumCount 区间 → 回填 YAML
- root-gen-fit-numcount — 步骤 B4b。生成 NumCount 侧带计数代码
- root-run-cxx — 步骤 B4c。运行 NumCount_*.cxx

3d. ROOT 辅助工具

- root-topoana — inclusive MC 拓扑分析（topo 包，独立使用）
- root-reweight-xgb — XGBoost 相空间 MC 重加权
- rootcode-rag — 生成/运行/自动修复 ROOT C++ 代码（RooFit 等）
- rag-query — BESIII 物理分析 RAG 查询（Milvus 检索）



Skills

Input/Output

阶段 3.5：结果分析（result）

- DSL-add-result — 从 DSL YAML 直接生成物理结果（截面/分支比/形状因子等）计算代码

阶段 4：系统误差分析

- rag-query — RAG 路线（sysErr=yes）。RAG 检索相似 BAM 的系统误差信息作为参考注释
- sysErr-rubyBased — Ruby 路线（sysErr-ruby=yes）。YAML 驱动公式优先推理：四类分类 + 逐项赋值 + 合成误差表
- systematic-uncertainties — 独立使用。系统误差研究（解析规格→公式→RAG→四类分类→子代理评估→ROOT宏→组合误差表）