



INCT
CERN
BRASIL



Introdução à Análise de Dados em FAE

Escola INCT CERN Brasil de Análise de Dados 2024 - SPRACE, NCC-UNESP

Eliza



Introdução

- Aspectos sobre a análise de dados em Física de Altas Energias (FAE)

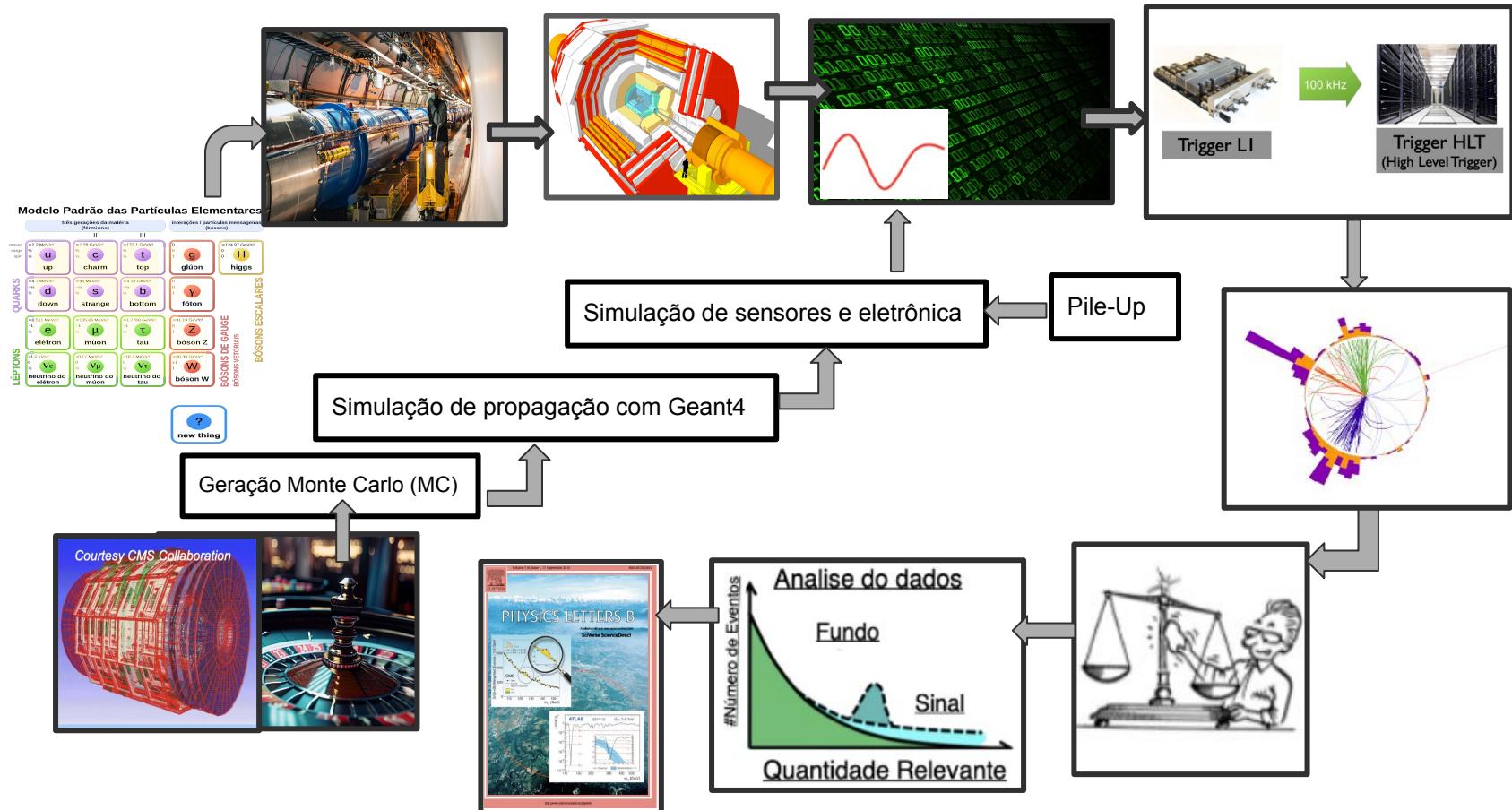
- A física de interesse
- Acelerador - LHC
- Detector - CMS
 - Subsistemas de detecção de partículas
 - Observáveis do detector
 - Seção de choque
 - Etapas da análise de dados

Parte I

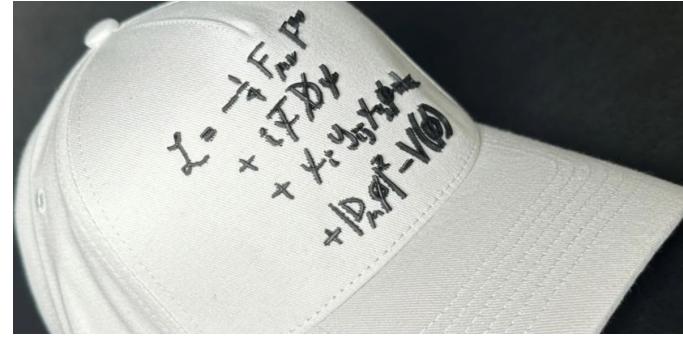
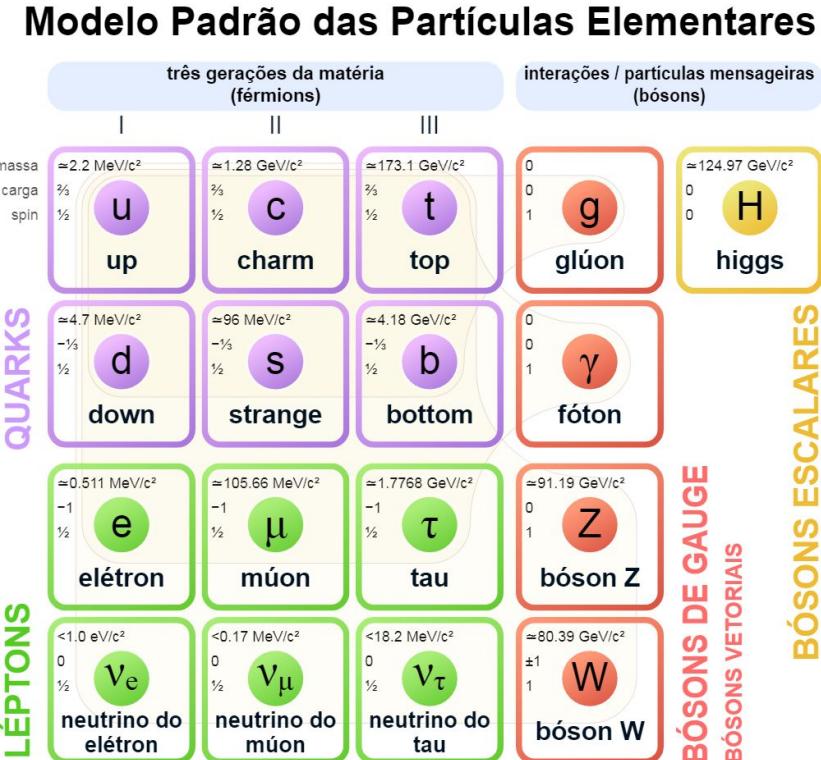
- Ferramentas de análise: ROOT e RooFit
- Formato e organização dos dados do CMS
- Técnicas de análises em FAE - colunar
 - Ferramentas para uma análise colunar

Parte II

A jornada das medições e buscas em física experimental de altas energias



O Modelo Padrão das Partículas Elementares



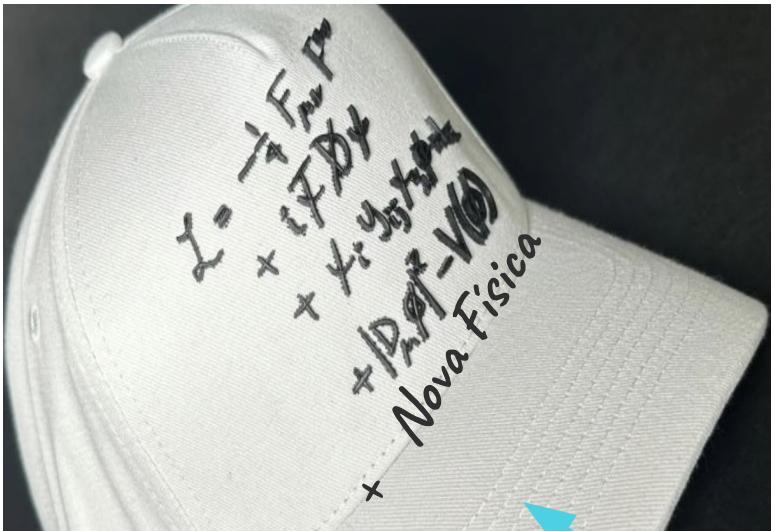
Descreve as partículas fundamentais e suas interações.

As partículas são divididas em três famílias: **quarks, léptons e bóson**.

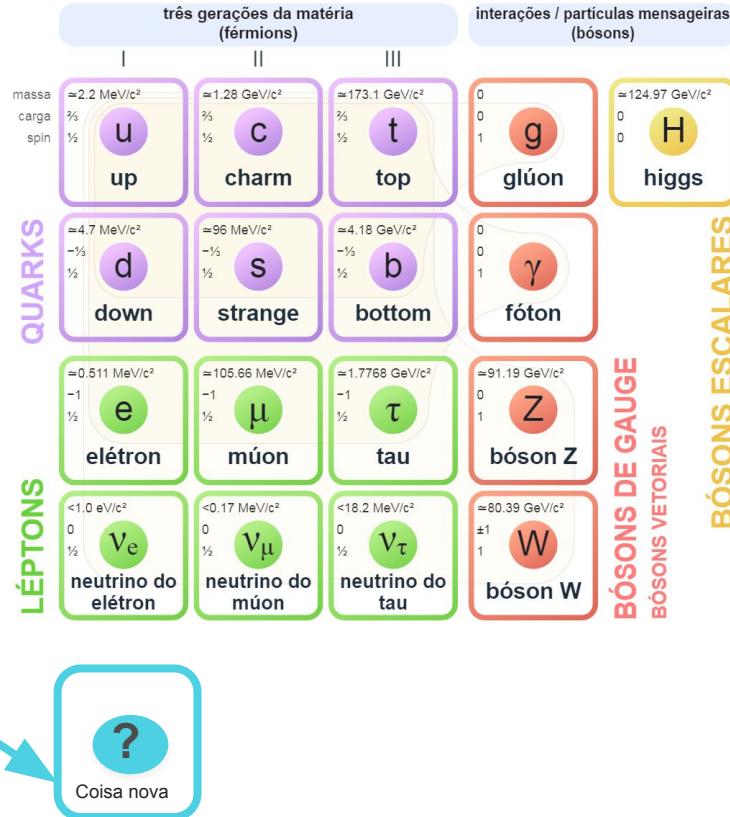
Descreve as forças forte (g), fraca (Z , W^\pm) e eletromagnética (γ).

As partículas do modelo padrão adquirem massa ao interagirem com o campo de Higgs.

Ir além do Modelo Padrão (BSM)



Modelo Padrão das Partículas Elementares



Por que a necessidade de ir além do Modelo Padrão?

(Alguns) dados que não conseguimos explicar:

- Assimetria entre matéria e antimateria (violação de CP?)
- Matéria escura (candidatos como WIMPs, ALPs?)
- Inflação cósmica

Hierarquia eletrofraca:

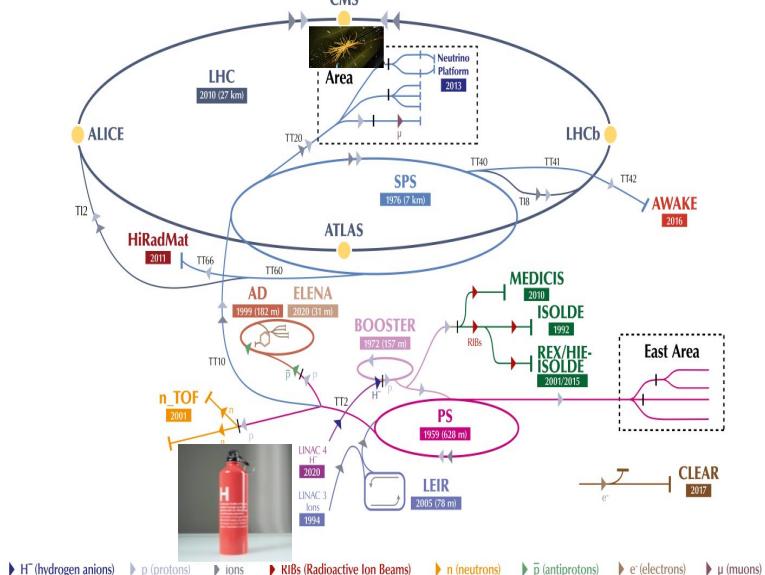
- A fraqueza da gravidade em comparação com a força eletrofraca ($\sim 10^{24}$)
- Diferença entre a escala eletrofraca (EWK) e a escala de Planck (possível "Deserto" entre elas?)
- ...

Hierarquia de sabor:

- Por que tantos parâmetros no Modelo Padrão (mais de 19)?
- Por que existem três famílias de partículas?
- Por que enormes hierarquias nas massas e acoplamentos dos férmons?
- ...

Física de altas energias?

- ver com partículas



LHC - Large Hadron Collider // SPS - Super Proton Synchrotron // PS - Proton Synchrotron // AD - Antiproton Decelerator // CLEAR - CERN Linear Electron Accelerator for Research // AWAKE - Advanced WAKefield Experiment // ISOLDE - Isotope Separator OnLine // REX/HIE-ISOLDE - Radioactive Experiment/High Intensity and Energy ISOLDE // MEDICIS // LEIR - Low Energy Ion Ring // LINAC - LINear ACcelerator // n_TOF - Neutrons Time Of Flight // HiRadMat - High-Radiation to Materials // Neutrino Platform

Aceleradores de partículas

- Assumindo a colisão de dois feixes de partículas 1 e 2, a **energia total** (relativística) E_T :

$$E_1 = E_A \text{ e } E_2 = E_A$$

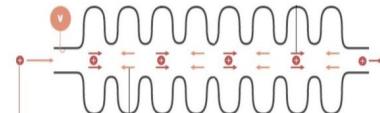
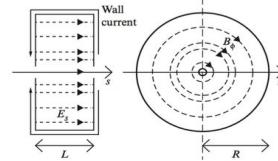
$$|\vec{p}_1| = |-\vec{p}_2| \approx E_A/c$$

$$E_T = 2E_A$$

- Aceleração** por radiofrequência

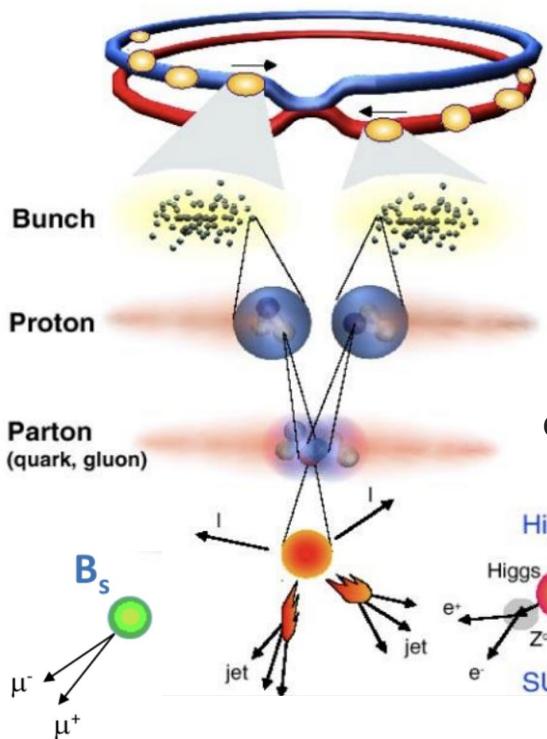
$$\mathbf{F} = q(\mathcal{E} + \mathbf{v} \times \mathbf{B})$$

$$\mathbf{E} = -\nabla\varphi - \frac{\partial \mathbf{A}}{\partial t}$$



- Desvio** de trajetórias
- Focalização** do feixe via quadrupolos
- Partículas aceleradas**
 - radiação síncrontron

LHC - especificações técnicas



Proton-Proton 2808 bunch/beam

Protons/bunch 10^{11}

27 km de circunferência

Crossing rate 40 MHz

Collisions $\approx 10^7 - 10^9$ Hz

Collisions $4000 W^\pm s / sec$

$1200 Z^0 s / sec$

$17 t\bar{t} s / sec$

$1 h^0 s / sec$

Produção de novas partículas 10^{-5} Hz



Estamos interessados em eventos raros

Filtragem em tempo real - Trigger

LHC - Luminosidade

Número de eventos = Luminosidade [pb⁻¹] x Seção de choque [pb]

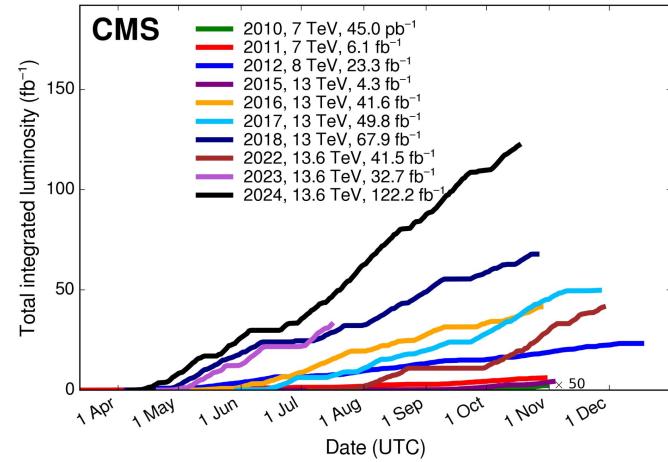
Depende de parâmetros dos feixes e da forma como eles se cruzam no ponto de interação.

Taxa de eventos: $\frac{dN}{dt} = \mathcal{L} \times \sigma_p$

$$\mathcal{L}_{\text{int}} = \int_0^t \mathcal{L}(t) dt$$

$$L = \frac{kN^2 f \gamma}{4\pi \beta^* \varepsilon} \cdot F$$

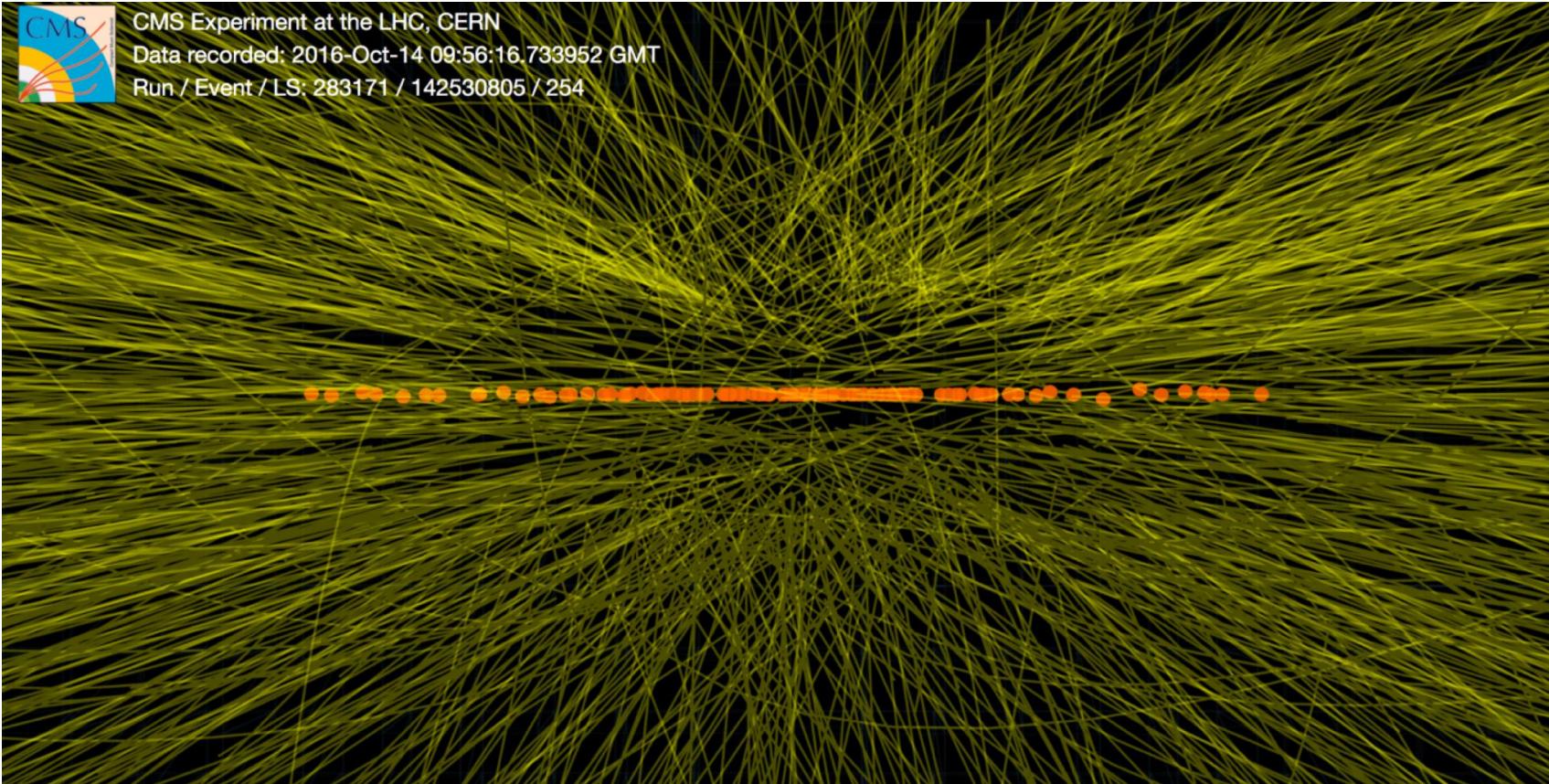
N..... No. particles per bunch
 k..... No. bunches
 f..... revolution freq.
 g..... rel. gamma
 β^* beta-function at IPs
 ε norm. trans. emit



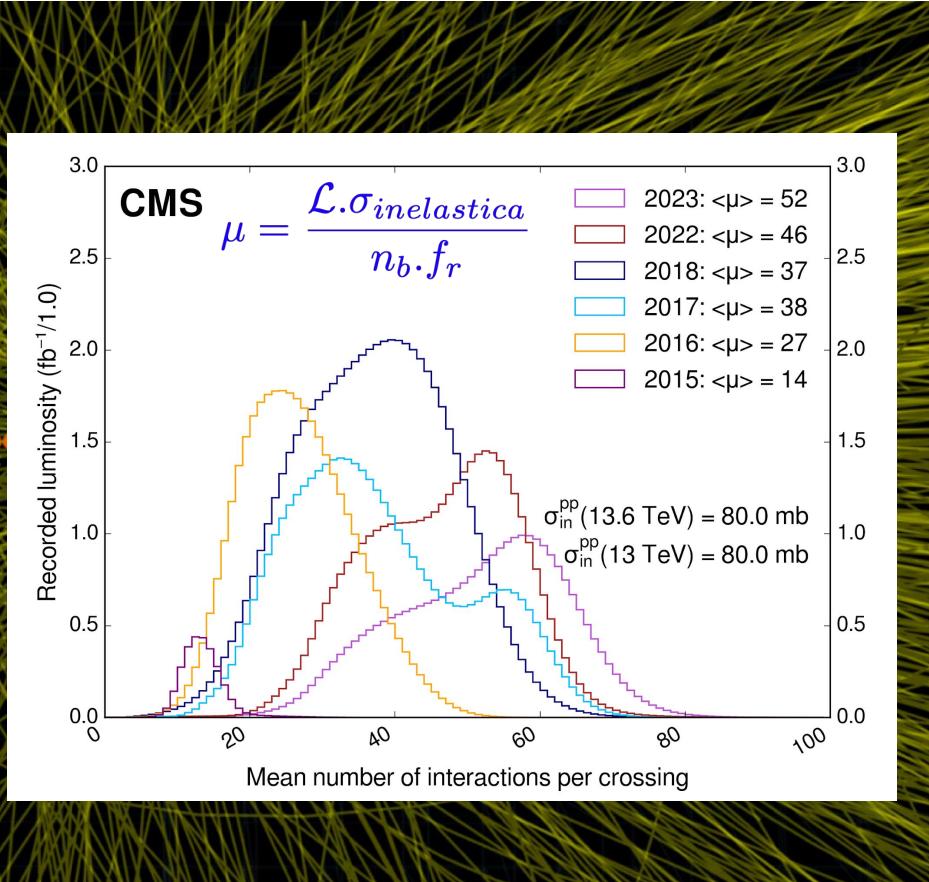
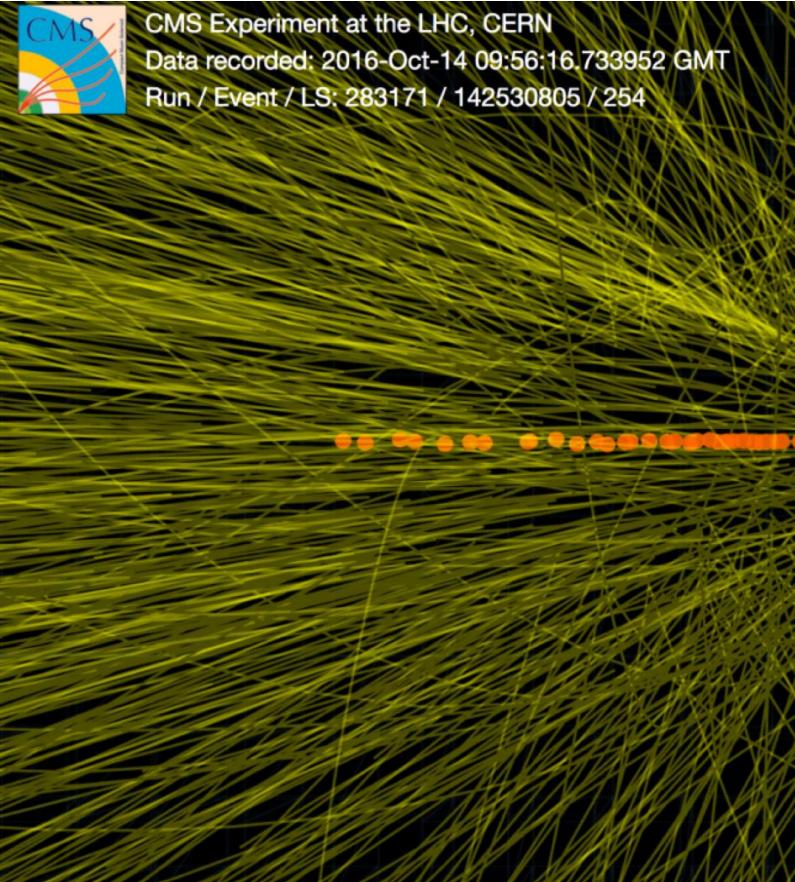
O fator de proporcionalidade entre a taxa de interações (ou eventos) e a seção de choque de um processo em um acelerador de partículas como o LHC.

Ex.: $dN_{\text{Higgs}}/dt \geq \mathcal{L} \cdot \sigma_{\text{Higgs}} \approx (\underline{0.02 \text{ pb/s}})(\underline{50 \text{ pb}}) = 1 / \text{s}$

LHC - Pile-up



LHC - Pile-up



LHC - Planos



- O maior nível de energia já registrado de $\sqrt{s}=13,6 \text{ TeV}$. E continuará até pelo menos 2025;
- Até agora, o LHC entregou cerca de < 10% do total da luminosidade integrada planejada;
 - a luminosidade quantifica a taxa de colisões;
- O LHC iniciará sua fase de alta luminosidade (HL-LHC) em 2026-2040.

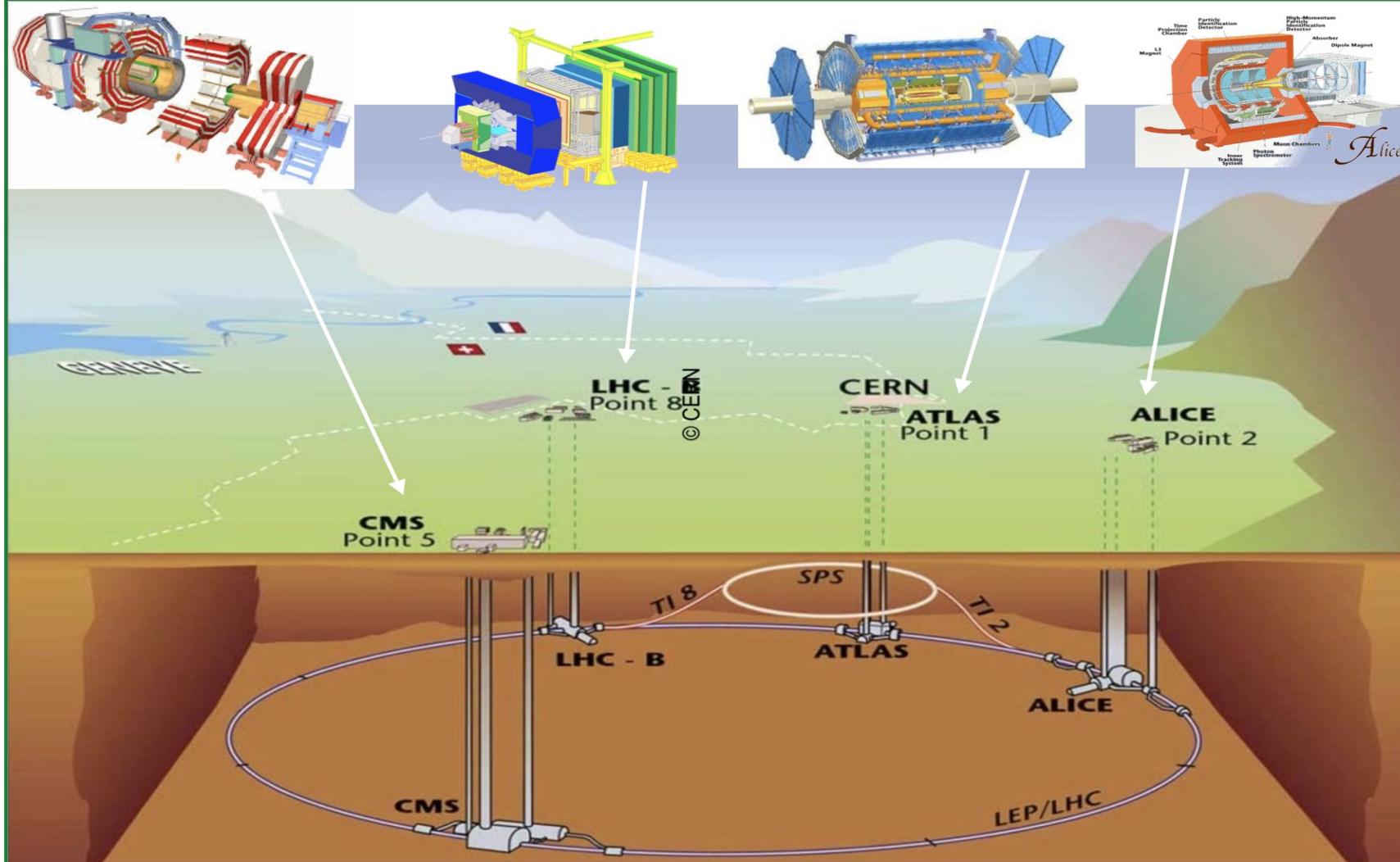
Os objetivos da física no LHC

Testar o Modelo Padrão (MP):

- Realizar medições de precisão e estudar processos raros.

Buscar física além do Modelo Padrão (BSM):

- Realizar buscas diretas e indiretas por novas partículas.
 - **Buscas diretas:** produzir partículas BSM nas colisões e detectar seus produtos de decaimento, **como aconteceu com o bóson de Higgs.**
 - **Buscas indiretas:** inferir a presença de partículas BSM através dos efeitos que elas causam nas propriedades das partículas do Modelo Padrão.





detector de partículas

Detector - CMS

CMS DETECTOR

Total weight : 14,000 tonnes
Overall diameter : 15.0 m
Overall length : 28.7 m
Magnetic field : 3.8 T

STEEL RETURN YOKE
12,500 tonnes

SILICON TRACKERS
Pixel ($100 \times 150 \mu\text{m}$) $\sim 16\text{m}^2$ $\sim 66\text{M}$ channels
Microstrips ($80 \times 180 \mu\text{m}$) $\sim 200\text{m}^2$ $\sim 9.6\text{M}$ channels

SUPERCONDUCTING SOLENOID
Niobium titanium coil carrying $\sim 18,000\text{A}$

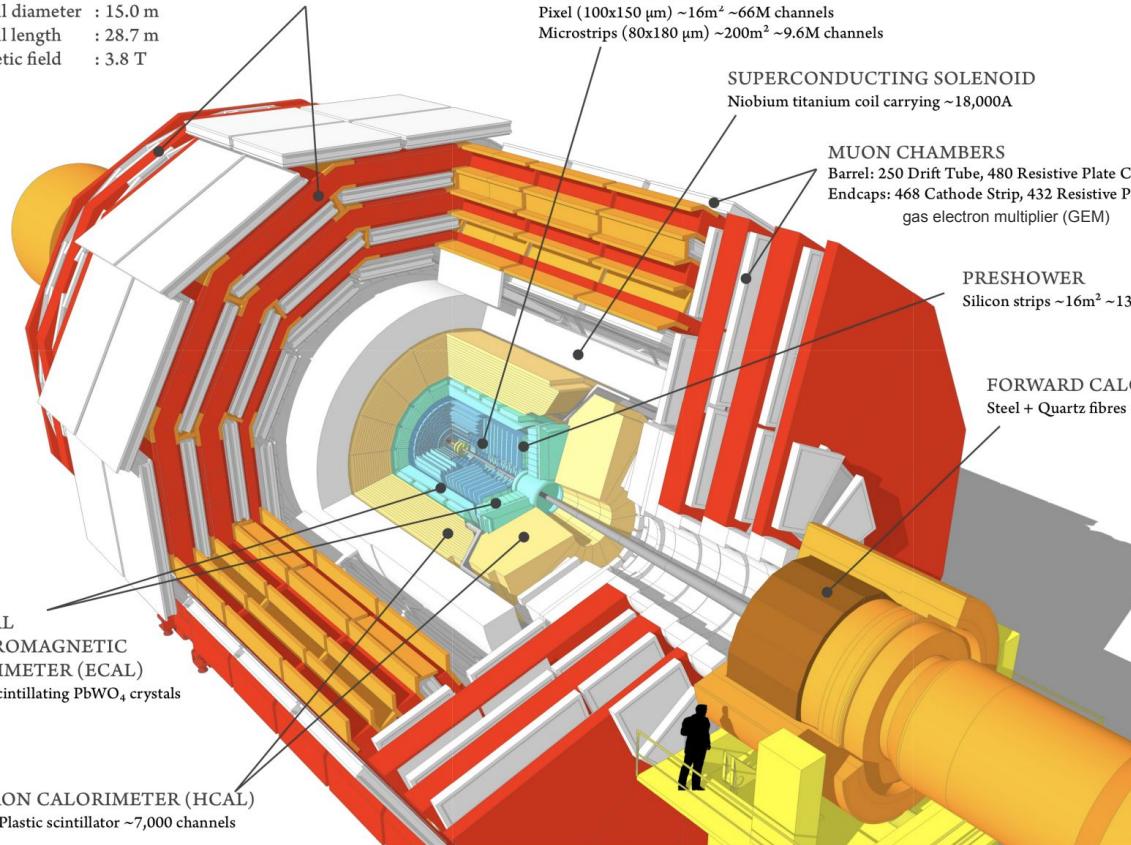
MUON CHAMBERS
Barrel: 250 Drift Tube, 480 Resistive Plate Chambers
Endcaps: 468 Cathode Strip, 432 Resistive Plate Chambers
gas electron multiplier (GEM)

PRESHOWER
Silicon strips $\sim 16\text{m}^2$ $\sim 137,000$ channels

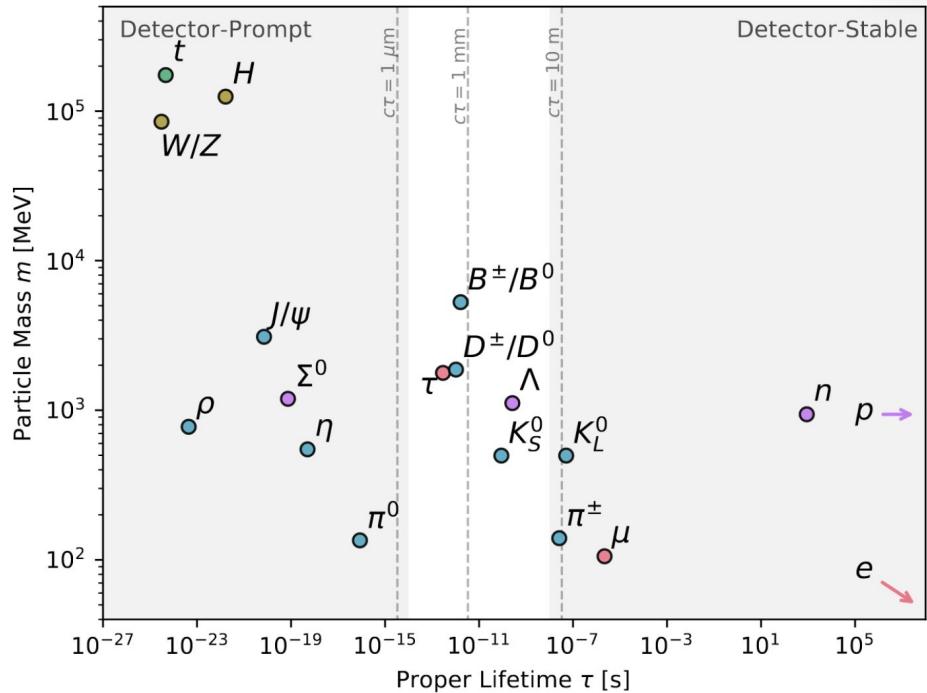
FORWARD CALORIMETER
Steel + Quartz fibres $\sim 2,000$ Channels

CRYSTAL
ELECTROMAGNETIC
CALORIMETER (ECAL)
 $\sim 76,000$ scintillating PbWO_4 crystals

HADRON CALORIMETER (HCAL)
Brass + Plastic scintillator $\sim 7,000$ channels



Detector - Identificação de partículas



- Apenas partículas quase estáveis são detectáveis diretamente.
- Todas as outras partículas, instáveis, decaem em outras partículas mais estáveis pouco após serem produzidas.

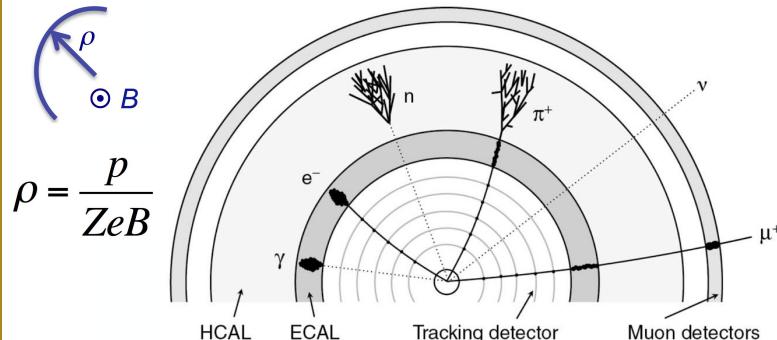
$\gamma, e, \mu, \pi, k, p, n$

ex.: A vida média do múon é $\tau \sim 2.2 \times 10^{-6} \text{ s} \Rightarrow c\tau \sim 660 \text{ m!}$

CMS - reconstrução e identificação dos objetos físicos

Os objetos são reconstruídos usando as informações dos diferentes subsistemas do detector, combinado com o algoritmo “Particle Flow” .

- Elétrons irradiam via freamento (Bremsstrahlung) ;
- Fótons convertem-se em pares e^+e^- no tracker;
- A energia dos jatos de partículas é formada de hadrons carregados/neutros (65%/10%) e fótons (25%):
 - a informação é explorada nos calorímetros e no tracker;
- A energia faltante ($\text{missing } E_T$) requer a reconstrução total do evento.

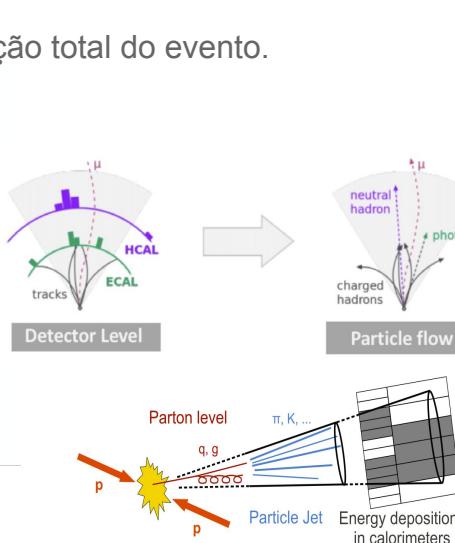


$$\rho = \frac{p}{ZeB}$$

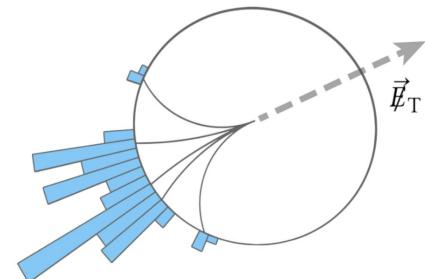
calorímetros:
mede a energia das partículas absorvendo-as

trackers:
detecta a trajetória das partículas carregadas

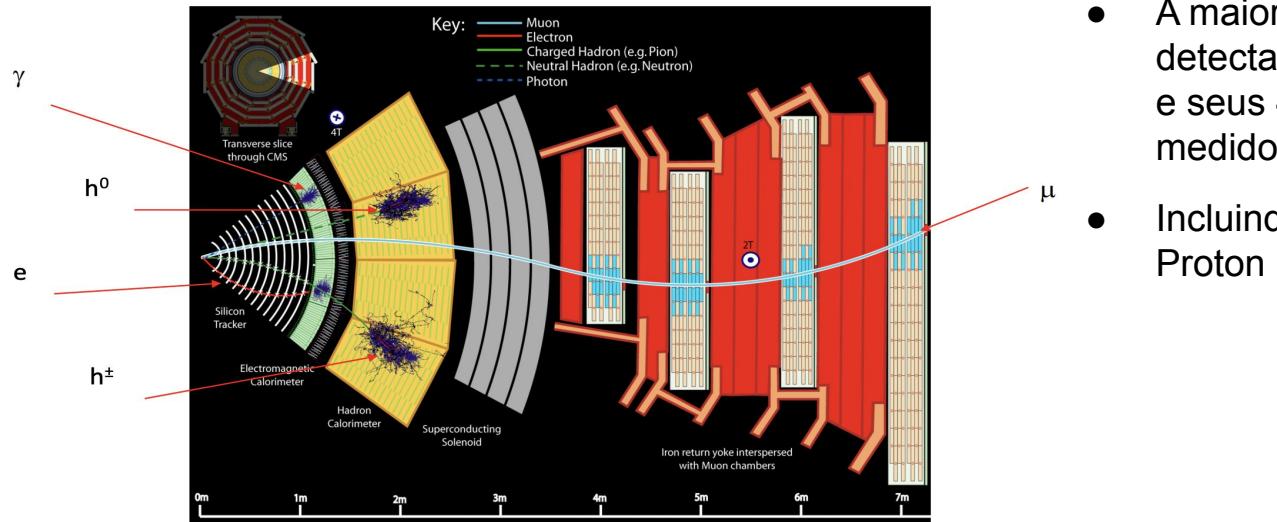
múons:
detectados em camadas mais externas do detector



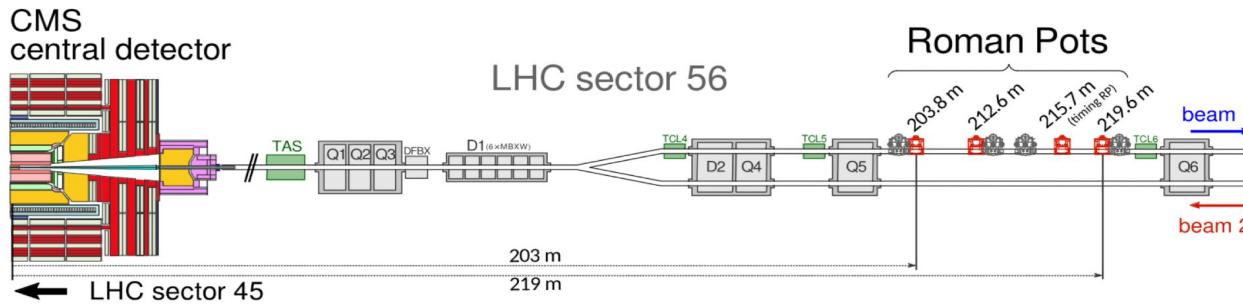
$$\vec{p}_T^{\text{miss}} = - \sum_i \vec{p}_{T,i}$$



CMS - reconstrução e identificação dos objetos físicos



- A maioria das partículas estáveis são detectadas diretamente, identificadas, e seus 4-vetores (E, p_x, p_y, p_z) são medidos.
- Incluindo prótons frontais no Precision Proton Spectrometer (PPS).



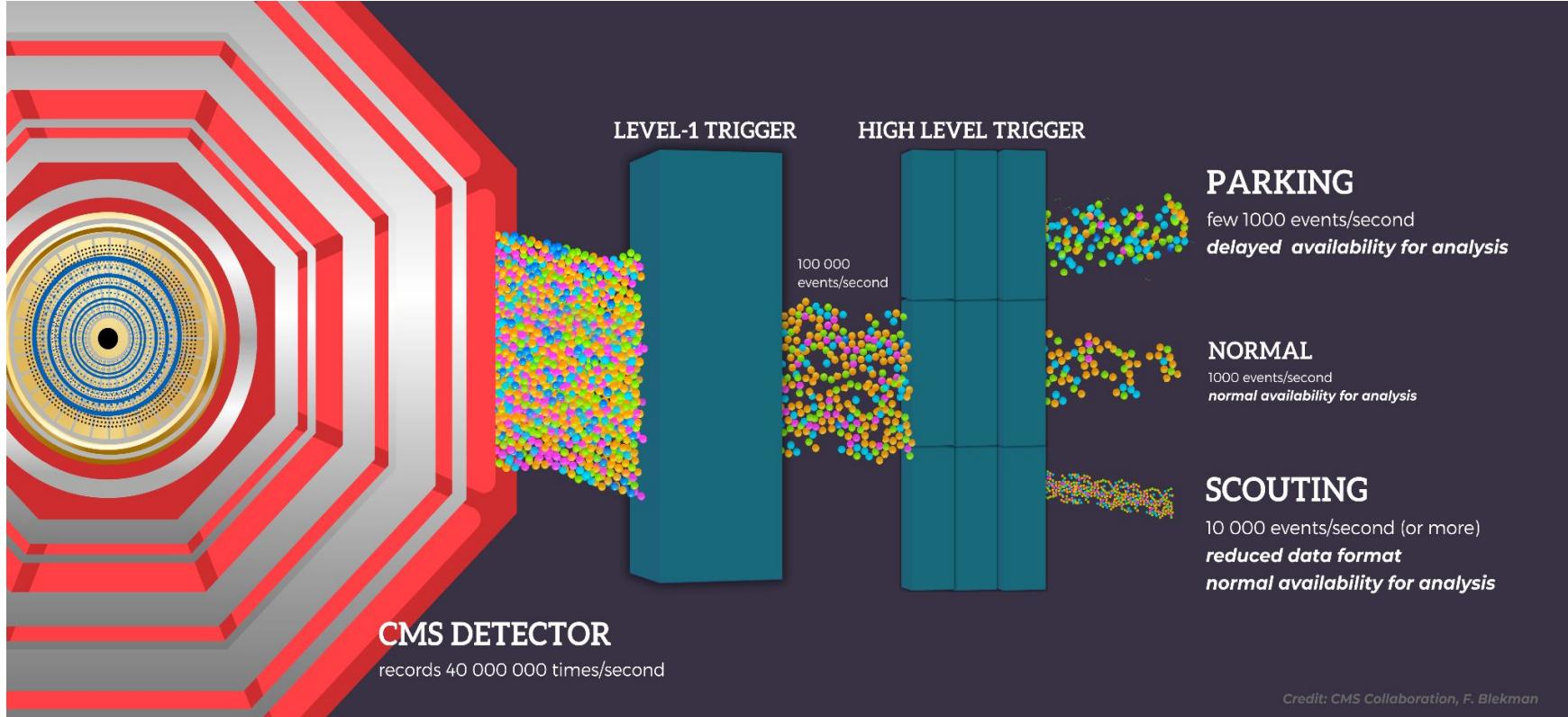
Sistema de trigger



- O papel do sistema de Trigger é reduzir a taxa de eventos produzidos pelo LHC para um nível que possa ser armazenado e analisado posteriormente, sem comprometer o alcance físico do experimento. Isso é feito levando em consideração as limitações da eletrônica dos detectores, do sistema de aquisição de dados e do próprio sistema de trigger.
- No CMS, o sistema de trigger foi projetado com dois níveis :



Filtrar eventos



Novos paradigmas de aquisição de dados estão sendo implementados no **Run 3** e serão ainda mais explorados na fase do **HL-LHC** (Alta Luminosidade do LHC).

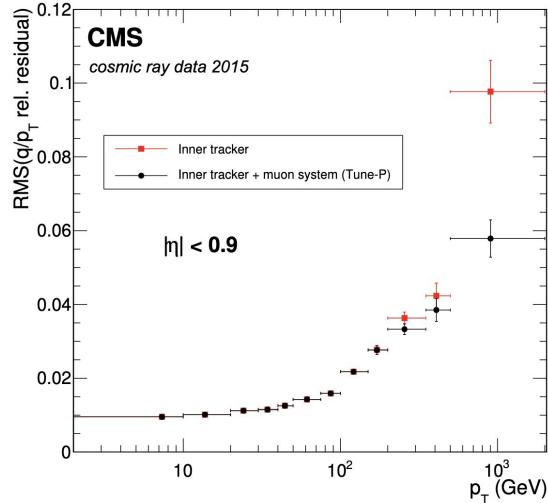
Calibração, alinhamento,...

Alinhamentos e calibrações são necessários:

- no nível do detector (alinhamento do tracker);
- no nível dos objetos físicos (por exemplo, escala de energia de jatos, escala de momento dos múons).

Fatores de escala (SF) e/ou eficiências são calculados para corrigir os dados simulados em relação aos dados reais:

- eficiência de identificação/tag, eficiência de trigger, resoluções,...



A resolução do momento dos múons é frequentemente apresentada como q/p_T porque isso é proporcional à curvatura da trajetória.

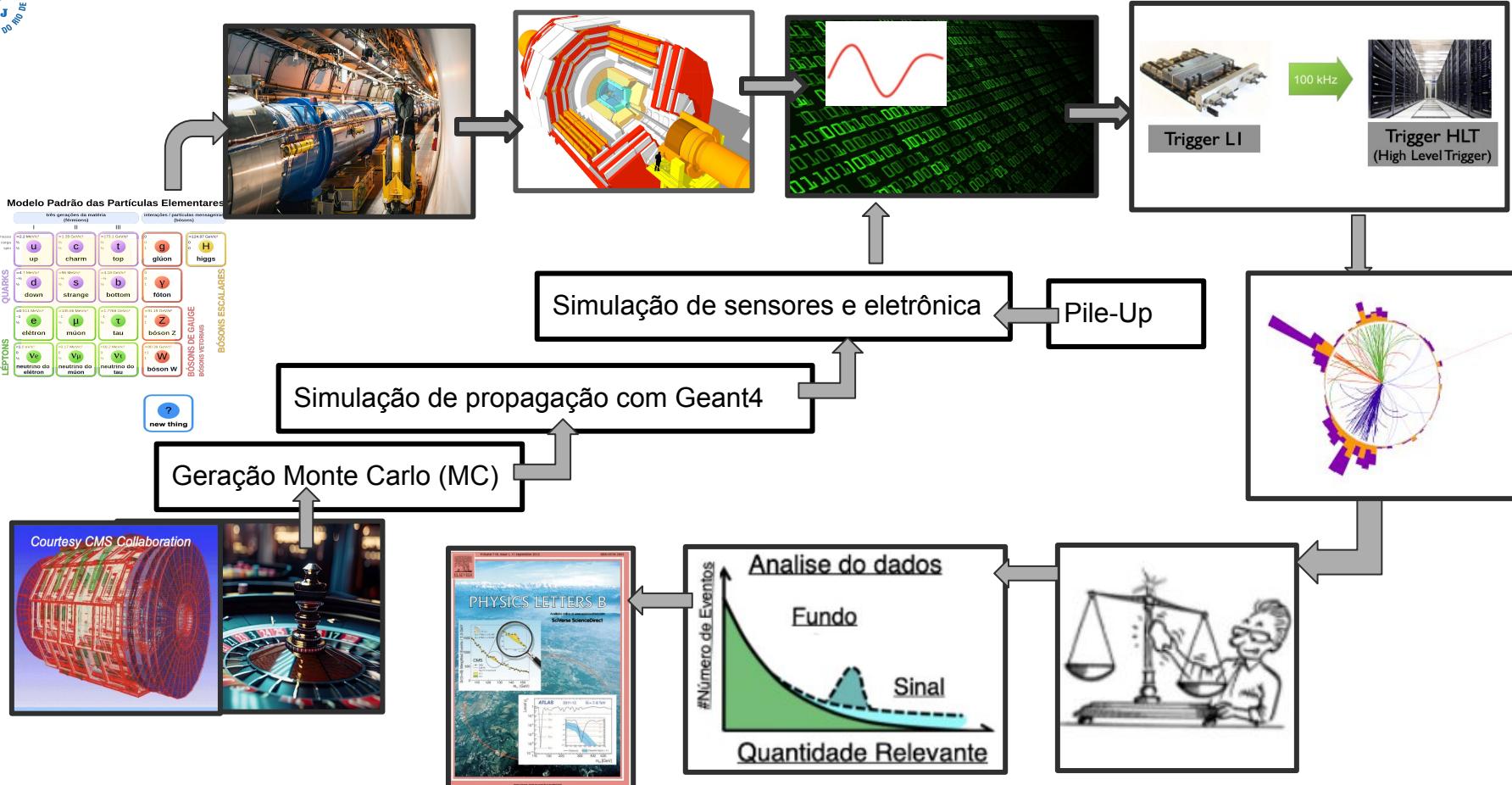
O modelo de computação do CMS

É baseado em uma estrutura hierarquizada de camadas (Tiers):

- **Tier 0 (CERN):**
 - Recebe os dados diretamente do ponto de coleta (P5) e realiza o processamento rápido (prompt reconstruction).
- **Tiers 1 (vários laboratórios, um por região geográfica):**
 - Responsáveis principalmente pela re-reconstrução de dados e simulação Monte Carlo (MC).
 - Armazenam cópias custodiadas dos dados.
- **Tiers 2 (vários, nacionais):**
 - Focados na produção de Monte Carlo (MC) e algumas análises de usuários finais.
 - Aqui estão as amostras usadas para análise de dados.
 - Uso do CRAB (ferramenta de análise).
- **Tiers 3:**
 - São normalmente clusters institucionais para análises locais.

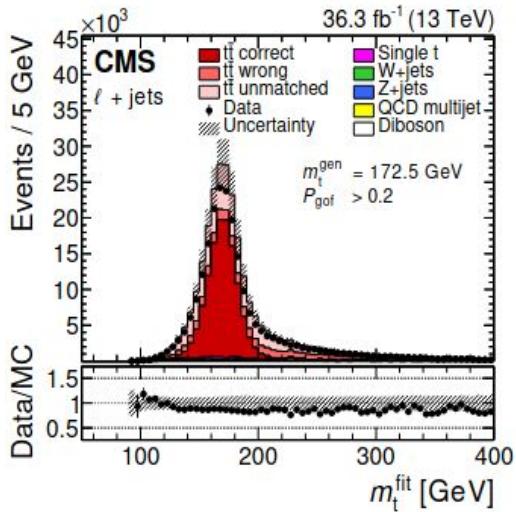
A estrutura é parte do **Worldwide LHC Computing Grid (WLCG)**, que utiliza recursos computacionais distribuídos por todo o mundo. Cada nível (Tier) possui uma função específica para garantir a eficiência na coleta, processamento, armazenamento e análise dos dados do CMS.



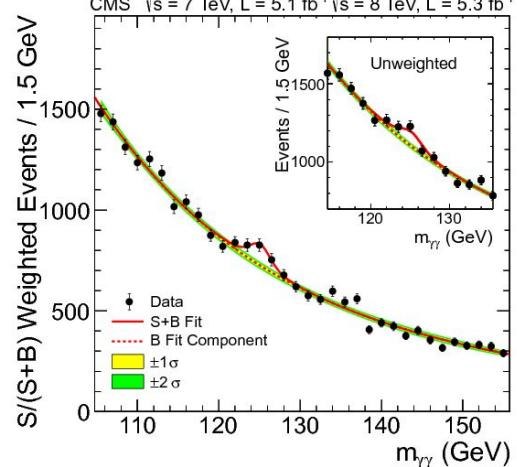
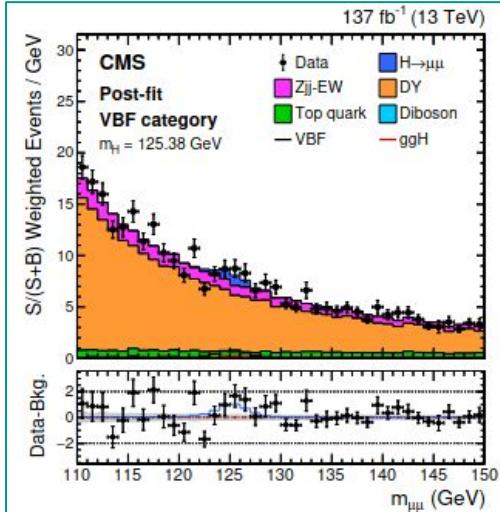


Medições e buscas

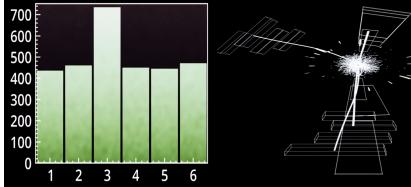
Medições



Buscas



Medições e buscas



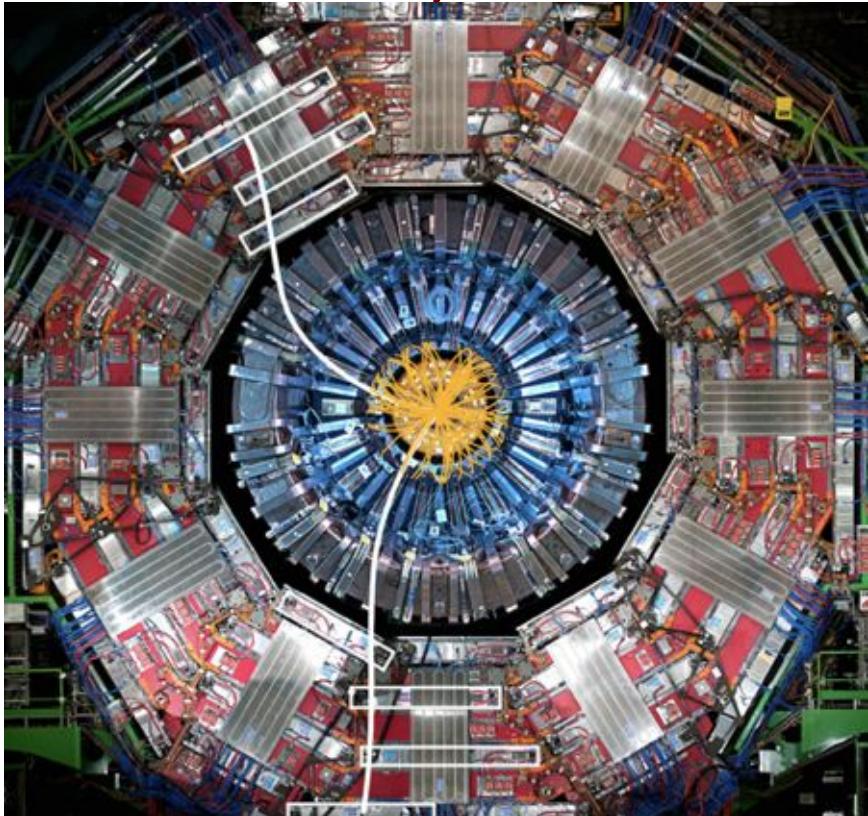
Os dados dos eventos são todos diferentes uns dos outros por várias razões:

- **Aleatoriedade intrínseca dos processos físicos** (Mecânica Quântica: $P \propto |A|^2$).
- **Resposta do detector**, que apresenta certa aleatoriedade (flutuações, resoluções, eficiências, etc.).

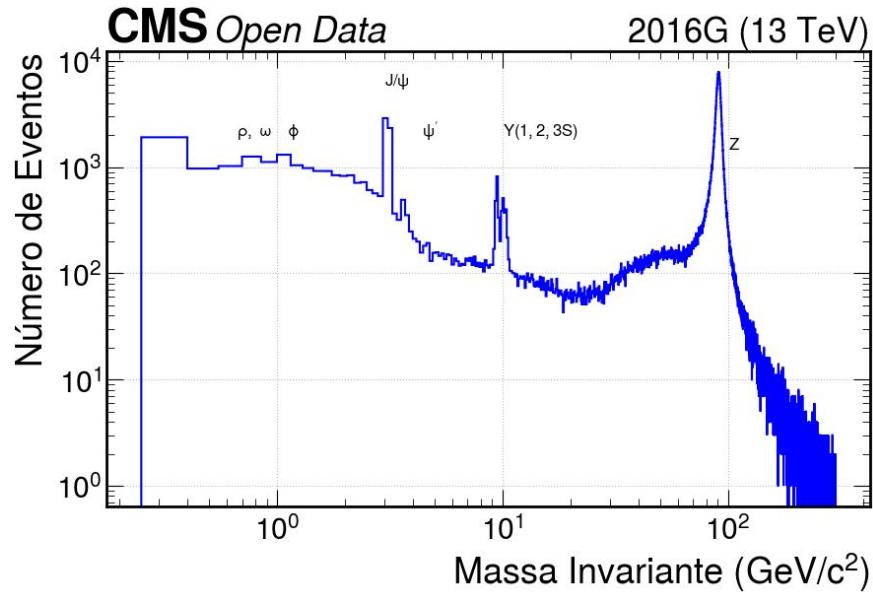
Um **banco de dados de condições** (Condition Database) registra as condições experimentais (alinhamento, resfriamento, canais inativos, informações de monitoramento da qualidade dos dados, etc.), que são levadas em consideração ao reconstruir os dados dos eventos.

Normalmente, um experimento coleta um grande número de eventos, e cada evento contém grandes quantidades de dados. O que se estuda, então, são **distribuições de grandezas físicas** (por exemplo, a massa de uma partícula, o tempo de vida, etc.).

Do detector à física

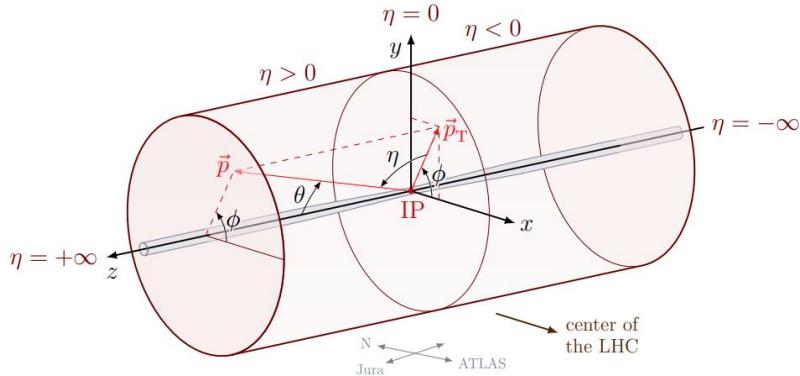
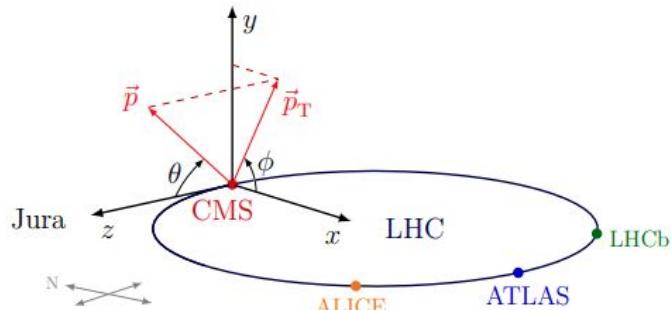


?



Transformar pulsos de sinais brutos do detector em algo comprehensível.

(Alguns) Observáveis do CMS

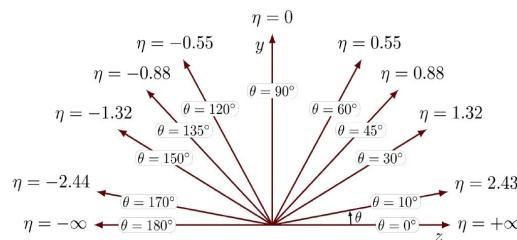


$$p_T = \sqrt{p_x^2 + p_y^2}$$

Momentum transversal (p_T)

$$\phi = \arcsin \left(\frac{p_y}{p_T} \right)$$

Ângulo azimuthal (ϕ)



$$\eta = -\ln \left[\tan \left(\frac{\theta}{2} \right) \right]$$

Pseudorapidez (η)

$$y = \frac{1}{2} \ln \left(\frac{E + p_z}{E - p_z} \right)$$

rapidez (y)

$$y|_{m=0} = \eta = \frac{1}{2} \ln \left(\frac{1 + \cos \theta}{1 - \cos \theta} \right) = -\ln \left[\tan \left(\frac{\theta}{2} \right) \right]$$

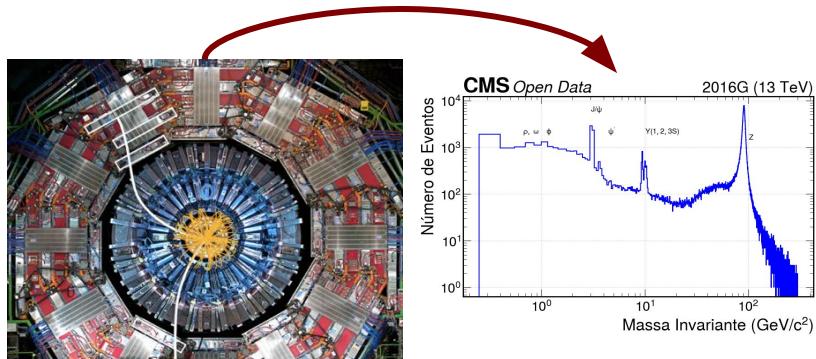
Do detector à física

Algoritmos de reconstrução convertem os sinais eletrônicos em informações como:

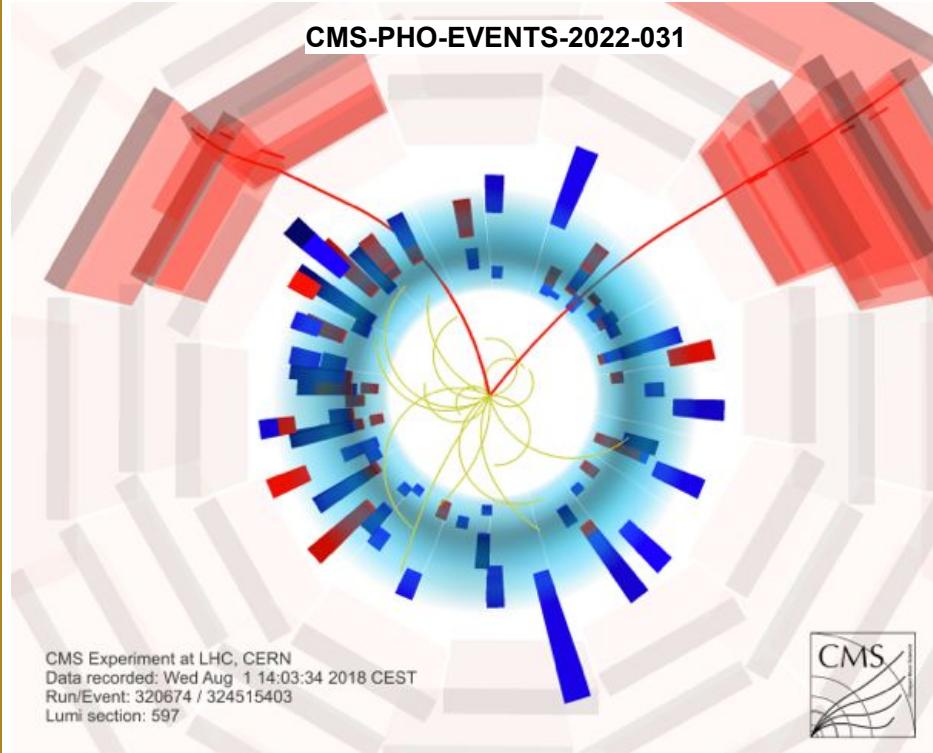
- Momenta das partículas
- Posições
- Tipos de partículas
- Ângulos polar e azimutal
- Tempo de vôo

Um evento contém toda a informação de uma colisão em particular.

Se uma partícula decai em duas partículas, podemos construir os 4-momenta delas ($P^\mu = (E, p_x, p_y, p_z)$ ou em termos de (E, p_T, η, ϕ)) para obter o 4-momentum da partícula original.



Do detector à física



1. **Identificação de partículas**
sinal nas câmaras de múons
→ é um múon!
 $m = m(\mu) \sim 106 \text{ MeV}/c^2$
2. **Trajetória da partícula**
câmaras de múons, mas especialmente o silicon tracker
momentum linear , $\mathbf{p} = (p_x, p_y, p_z)$
 - 4-momenta de cada múon: $\mathbf{P}_\mu = (E, p_x, p_y, p_z)$
 - para o par de dimuon:
$$\mathbf{P}_{\mu\mu} = \mathbf{P}_{\mu 1} + \mathbf{P}_{\mu 2}$$
 - massa invariante: $M_{\mu\mu} = (\mathbf{P}_{\mu 1} + \mathbf{P}_{\mu 2})^2$

Número de eventos observados

O número de eventos observados N_{obs} para um dado processo físico pode ser expresso como:

$$N_{obs} = \mathcal{L} \cdot \sigma \cdot \mathcal{A} \cdot \epsilon$$

onde:

- σ : é a **seção de choque** de produção, que é uma quantidade calculada teoricamente e indica a probabilidade de um determinado processo ocorrer. (normalmente medida em **barns**, onde $1 \text{ barn} = 10^{-28} \text{ m}^2$).
- \mathcal{L} : é a **luminosidade** do acelerador, que mede quantas partículas colidem em uma determinada área por unidade de tempo.
- \mathcal{A} : é a **aceitação**, que indica a fração de partículas geradas pelo processo que realmente entram no volume do detector e podem ser detectadas.
- ϵ : é a **eficiência de seleção**, que pode ser subdividida em duas partes:
 - $\epsilon = \epsilon_1 \times \epsilon_2$
 - ϵ_1 : é a **eficiência do objeto**, ou seja, a eficiência de reconstruir as partículas detectadas como objetos físicos no detector.
 - ϵ_2 : é a **eficiência de análise**, ou seja, a eficiência de aplicar corretamente os critérios de seleção para identificar os eventos de interesse.

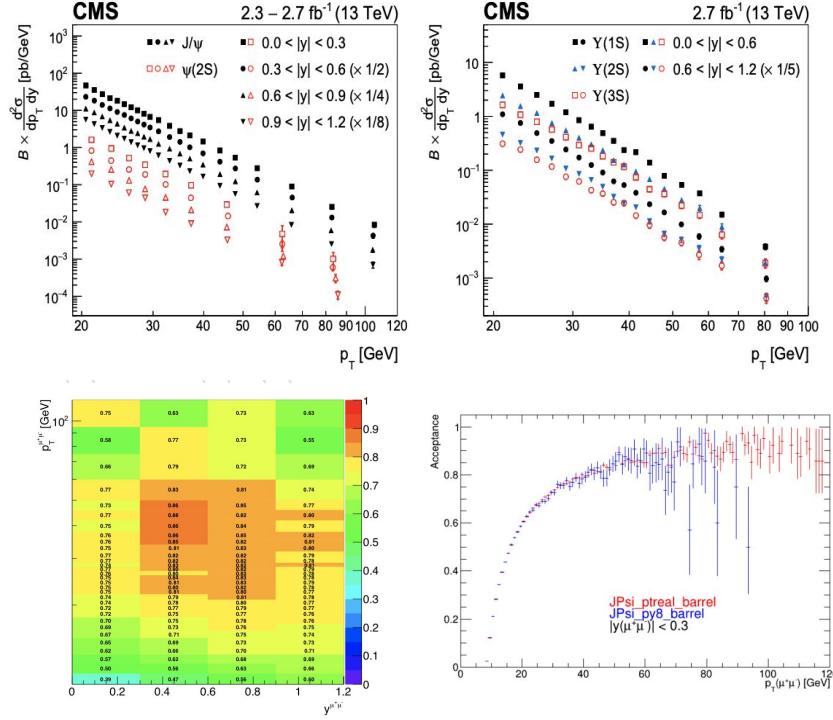
A seção de choque

Ex.: Measurement of Quarkonium production cross sections in pp collisions at $\sqrt{s}=13$ TeV
(arXiv:1710.11002)

$$\mathcal{B}(\mathcal{Q} \rightarrow \mu^-\mu^+) \frac{d^2\sigma}{dp_T dy} = \frac{N(p_T, y)}{\mathcal{L}\Delta y\Delta p_T} \left\langle \frac{1}{\epsilon(p_T, y)\mathcal{A}(p_T, y)} \right\rangle$$

- $\Delta p_T \Delta y$: largura dos bins de p_T e rapidez
- N : yield do sinal ajustado (J/Ψ , $\Psi(2S)$ e $Y(nS)$)
- \mathcal{A} : aceitação do detector obtida por simulação
- ϵ : eficiências de reconstrução e trigger do detector (simulação ou baseada em dados)
- \mathcal{L} : luminosidade integrada da amostra
- $\langle \rangle$: a média sobre bin de rapidez , p_T

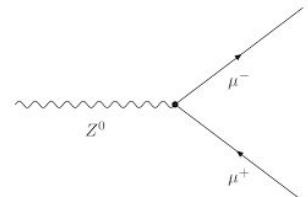
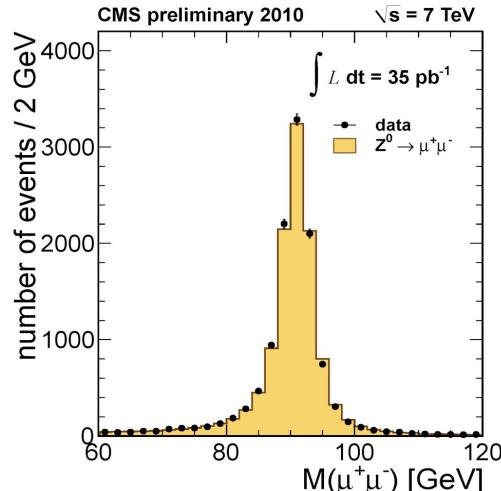
$\mathcal{N} = \mathcal{L} \cdot \sigma$
A área efetiva de interação
unidade: barn, $1b = 10^{-28} m^2 = 100 fm^2$



(Algumas) Etapas de uma análise de dados

Veremos, as etapas práticas de uma análise de dados em física de partículas, que incluem a **seleção de eventos e objetos**, a **otimização de cortes** e a separação entre **sinal e fundo**.

- **Seleção de Eventos:** determinar a configuração final dos eventos que se quer medir ou das novas físicas que se quer procurar. Isso envolve tanto os objetos físicos (jatos, léptons, etc.) quanto a **topologia e as propriedades cinemáticas** desses objetos.
- **Aceitação e Eficiência:** a aceitação e as limitações do detector forçam a aplicação de limites no p_T e η dos objetos.
- **Seleção de Objeto:** requisitos de qualidade são aplicados a objetos reconstruídos (como jatos, léptons, etc.) para garantir que eles sejam adequadamente isolados e minimamente afetados por ruído ou **pile-up** (acúmulo de interações de baixa energia no detector). Estes cortes garantem que apenas os eventos que contêm os objetos e topologias de interesse sejam mantidos para análise.



Etapas da Análise - otimização dos cortes

A otimização de cortes é útil para maximizar a quantidade de eventos de sinal e minimizar o fundo (background). O objetivo é selecionar a maior quantidade possível de eventos de sinal, com a menor contaminação de eventos de fundo, levando em consideração a pureza e a eficiência da seleção.

- **Significância:** para medir a eficácia dos cortes, utiliza-se a significância, que é uma medida de quão bem o sinal pode ser distinguido do fundo. A significância:

$$\frac{S}{\sqrt{N}} = \frac{S}{\sqrt{S+B}} \approx \frac{S}{\sqrt{B}}$$

$N = S+B$

Seleções mais rígidas aumentam a pureza, mas diminuem a eficiência (menos eventos de sinal sobrevivem aos cortes), enquanto seleções mais frouxas aumentam a eficiência, mas resultam em maior contaminação por eventos de fundo.

$$\epsilon = \frac{N_{sig,sel}}{N_{sig,total}} \qquad \qquad P = \frac{N_{sig}}{N_{sig} + N_{bkg}}$$

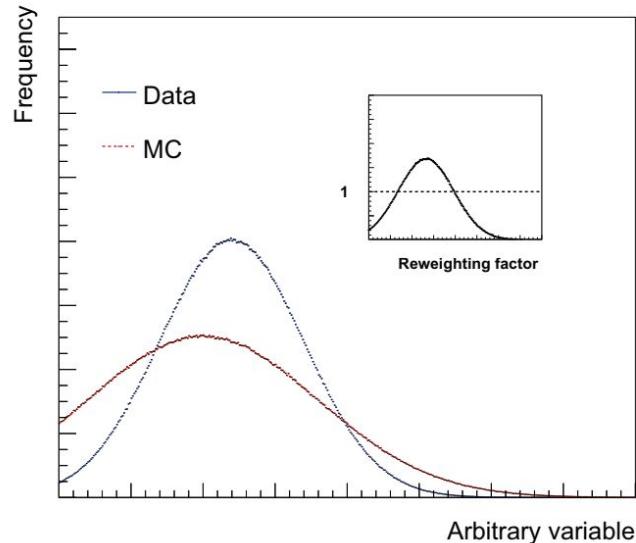
- **Tabela do fluxo de seleção:** uma tabela do fluxo de seleção é utilizada para rastrear quantos eventos de sinal e de fundo sobrevivem após cada corte aplicado. Isso é feito usando simulações para otimizar as escolhas de corte.

Etapas da Análise

Reweighting: utilizada para ajustar as distribuições geradas por simulações de Monte Carlo (MC) para que correspondam melhor aos dados experimentais. Algumas razões:

- Modelagem incorreta dos efeitos do detector;
- Falhas no detector ou efeitos experimentais;
- Inadequação no modelo do processo físico;
- Modelagem inadequada do pile-up em MC.

O reweighting permite corrigir discrepâncias de maneira flexível, sem a necessidade de rodar novamente simulações inteiras, o que muitas vezes é impraticável.



Esse peso é calculado com base na diferença entre as distribuições de MC e de dados em cada bin (ou faixa de variáveis cinemáticas).

Etapas da Análise - estimar o fundo

- O fundo (background) representa eventos que não correspondem ao sinal que você está buscando, mas que podem ser confundidos com ele. Tipos(alguns) de background :
 - **Background Irreduzível**: processos do MP que produzem o mesmo estado final que o sinal procurado. Esses eventos são indistinguíveis do sinal em termos de partículas finais produzidas.
 - Para estimar a contribuição desses fundos irreduzíveis com alta precisão, é essencial ter medições precisas desses processos.
 - **Background Redutível**: processos que produzem um estado final semelhante, mas que pode ser identificado erroneamente como sinal devido a objetos mal identificados ou mal reconstruídos no detector.
 - Impor cortes de seleção rigorosos sobre os objetos e a topologia do evento.
- Ele pode ser estimado a partir de simulações de Monte Carlo, mas para processos não bem modelados por simulações, são usados métodos baseados em dados reais.
- A melhor metodologia de estimativa de background para uma análise específica é aquela que **melhor desempenho** apresenta nessa análise particular.

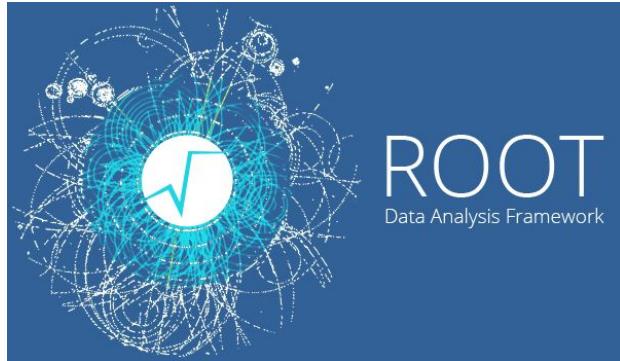
Etapas da Análise

- **Métodos estatísticos:** conceitos estatísticos são essenciais para quantificar a significância dos resultados, calcular incertezas e realizar testes de hipóteses. A estatística é usada para comparar dados reais com previsões do MC, ajudando a determinar se os dados estão de acordo com o Modelo Padrão ou se apontam para novas físicas.
 - **Ferramentas computacionais:** um grande exemplo é o **ROOT** (RooFit e RooStats) , ele é o software mais utilizado para realizar análises em física de altas energias, fornecendo uma variedade de ferramentas para análise estatística, ajuste de funções, visualização de dados, e mais.
 - Além disso, o ecossistema **Scikit-HEP** oferece várias ferramentas voltadas para a análise de dados em HEP, incluindo o hepstats para testes estatísticos, estimativa de incertezas, intervalos de confiança e outras operações estatísticas comuns em análises de física de partículas.

Parte II

ROOT

- É a principal ferramenta adotada em **Física de Altas Energias** e outras ciências (assim como na indústria).
- Desenvolvido pelo CERN ([René Brun, Fons Rademakers](#)) é um programa de computador e biblioteca orientado a objetos.
 - Última versão: **6.32.06 - 2024-09-22**;
 - Disponibilidade de instalação em diferentes sistemas:
<https://root.cern.ch/build-prerequisites> ;
- O ROOT é escrito principalmente em **C++** e possui integrações poderosas com **Python**.
- Principais funcionalidades:
 - Histogramas e gráficos (2D, 3D e visualização de eventos), funções, ajustes, ferramentas estatísticas (RooFit/RooStats), ferramentas matemáticas em geral, Manipulação de dados, geração de números aleatórios, ...



Página Principal: <https://root.cern.ch/>

Maneiras de usar o ROOT

- Interpretador de linha de comando C++ do ROOT
- Macros C++ do ROOT
- C++ compilado usando ROOT como biblioteca
- Como uma biblioteca em código Python (PyROOT)

```
Last login: Thu Oct 31 07:53:30 on ttys000
eliza@MacBook-Air-de-Eliza Escola2024 % root
```

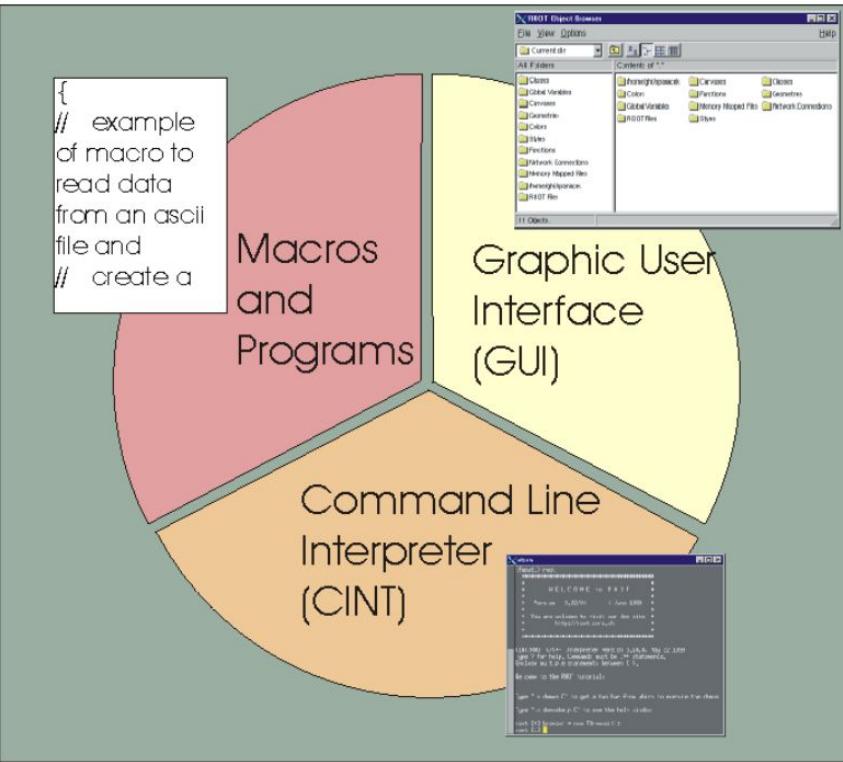
```
| Welcome to ROOT 6.30/04          https://root.cern |
| (c) 1995-2024, The ROOT Team; conception: R. Brun, F. Rademakers |
| Built for macosxarm64 on Jan 31 2024, 08:17:06 |
| From heads/master@tags/v6-30-04 |
| With Apple clang version 15.0.0 (clang-1500.1.0.2.5) |
| Try '.help'/'?'. 'demo', '.license', '.credits', '.quit'/'q' |
```

```
root [0] ■
```

\$ root (para inicializar o prompt do ROOT)

\$ root -l (para inicializar o ROOT sem mensagens de inicialização)

\$ root -b (para não permitir que o ROOT abra janelas)



Maneiras de usar o ROOT



O **Jupyter Notebook** é um ambiente de computação interativa que permite aos usuários criar documentos (notebooks) que incluem:

- **Código ao vivo**: Blocos de código que podem ser executados e modificados em tempo real.
- **Widgets interativos**: Ferramentas que facilitam a interação com dados e visualizações.
- **Gráficos**: Visualizações geradas diretamente dos dados analisados.
- **Equações**: Suporte para escrever fórmulas matemáticas usando LaTeX.
- **Imagens e vídeos**: Mídia visual embutida para complementar as análises.

```
import ROOT
```

SWAN (Service for Web-based ANalysis) é um serviço do CERN que permite aos usuários realizar análises de dados interativas na nuvem, seguindo o modelo de "software como serviço" (SaaS). Ele é baseado nos populares **notebooks Jupyter**, permitindo que os usuários escrevam e executem suas análises de dados usando apenas um navegador web.



o Colab é uma lista de células que podem conter textos explicativos ou códigos executáveis e suas respectivas saídas.

Para usar o root no colab

```
[ ] 1 !pip install -q condacolab
```

```
▶ 1 import condacolab
```

```
[ ] 1 condacolab.install()
```

```
[ ] 1 !conda --version
```

```
[ ] 1 !conda install ROOT
```

```
[ ] 1 import ROOT as rt
```



Root - no prompt

.q	Quit
.L macro.C	carrega um arquivo macro
.x macro.C	Carrega e executa a macro
.x macro.C++	Compila e executa
.! ...	Para permitir um comando Shell
.help	Nos dá a lista de comandos

O prompt do ROOT é um interpretador de comandos em C++ já compilado

```
> root
root [0] int x = 4
(int) 4
root [1] int y = 5
(int) 5
root [2] x + y
(int) 9
root [3]
```

Root - sintaxe

A estrutura geral da criação de objetos em C++ no ROOT segue esta forma:

TSomething* m = new **TSomething(stuff);**

Aqui, TSomething representa a classe do objeto que você quer criar (por exemplo, TH1F para histogramas unidimensionais de floats).

m é o nome do ponteiro, ou seja, a variável que armazena o endereço de memória do objeto criado.

Obs.: Use, "delete mything;" para liberar essa memória.

o símbolo * indica que m será um **ponteiro** para um objeto do tipo TSomething.

new é um operador em C++ que **aloca memória** para um novo objeto e retorna o endereço dessa memória, que é armazenado no ponteiro.

TSomething(stuff) é o **construtor** é um método especial da classe que inicializa o objeto com os parâmetros específicos (neste caso, stuff).



Root - syntaxe

Python

```
m = TSomething(stuff)
```

TSomething: classe do objeto que você deseja criar.

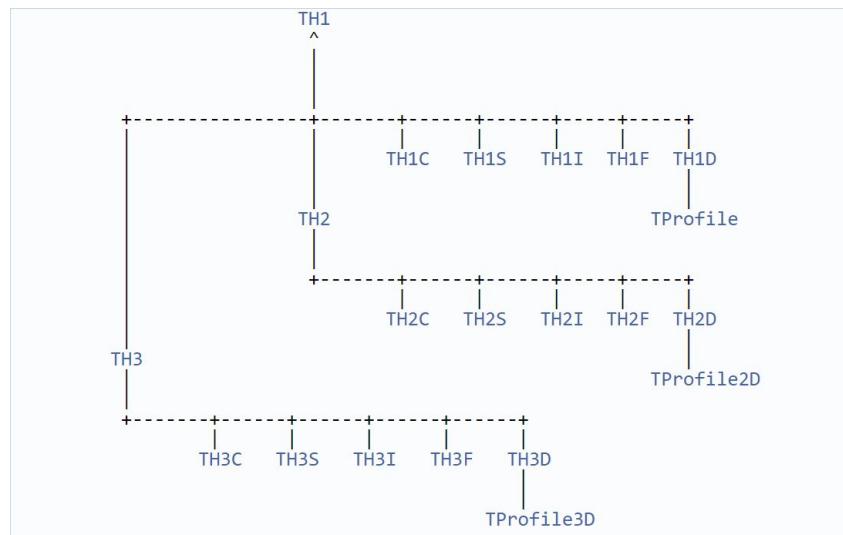
m: nome da variável que armazena a referência ao objeto **TSomething**.

TSomething(stuff): construtor da classe **TSomething**, que inicializa o objeto com o parâmetro **stuff**.

Root - Histogramas

Oferece uma variedade de histogramas com funcionalidades diversas:

- Existem classes de histogramas para diferentes dimensões: TH{1,2,3} classes + THN
- Os histogramas podem armazenar dados em diferentes tipos de precisão, dependendo das necessidades da análise:
 - TH1D: Um histograma unidimensional que armazena valores como números de precisão dupla (double).



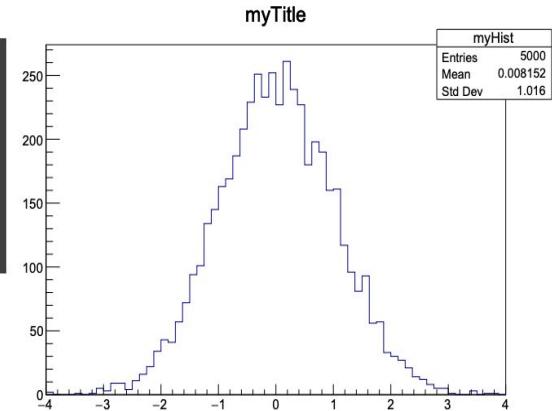
Outras variações incluem:

- **TH1F, TH2F, TH3F**: Armazenam valores de ponto flutuante (float).
- **TH1I, TH2I, TH3I**: Usam inteiros de 32 bits.
- **TH1S, TH2S, TH3S**: Usam inteiros de 16 bits (int16).
- **TH1C, TH2C, TH3C**: Usam inteiros de 8 bits (int8).

Root - Histogramas - TH1F

C++

```
> root
root [0] TH1F h("myHist", "myTitle", 64, -4, 4)
root [1] h.FillRandom("gaus")
root [2] h.Draw()
```



Python

```
> python
>>> import ROOT
>>> h = ROOT.TH1F("myHist", "myTitle", 64, -4, 4)
>>> h.FillRandom("gaus")
>>> h.Draw()
```

No ROOT, você também pode importar classes específicas individualmente, em vez de carregar todo o módulo.

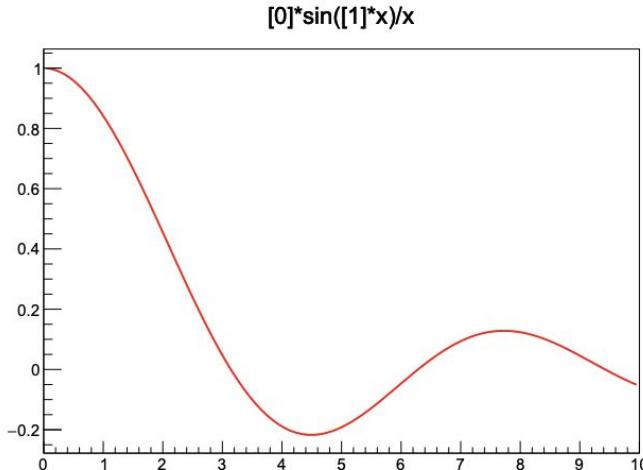
```
>>> from ROOT import TH1F
```

ROOT - funções -TF1

```
root [0] TF1 f1("f1","sin(x)/x",0.,10.); // name, formula, min, max
root [1] f1.Draw();
```

Versão com a definição dos parâmetros:

```
root [2] TF1 f2("f2","[0]*sin([1]*x)/x",0.,10.);
root [3] f2.SetParameters(1,1);
root [4] f2.Draw();
```





ROOT - TFile

```
TFile *file = new TFile("myFile.root", "RECREATE");
```

c++

```
import ROOT  
file = ROOT.TFile("myFile.root", "RECREATE")
```

python

Você pode usar outras opções:

- "[READ](#)": abre um arquivo existente para leitura.
- "[UPDATE](#)": abre um arquivo existente para leitura e escrita, criando-o se não existir.
- "[NEW](#)" ou "[CREATE](#)": tenta criar um novo arquivo e falha se o arquivo já existe.
- "[RECREATE](#)": cria um novo arquivo, substituindo um arquivo existente com o mesmo nome.

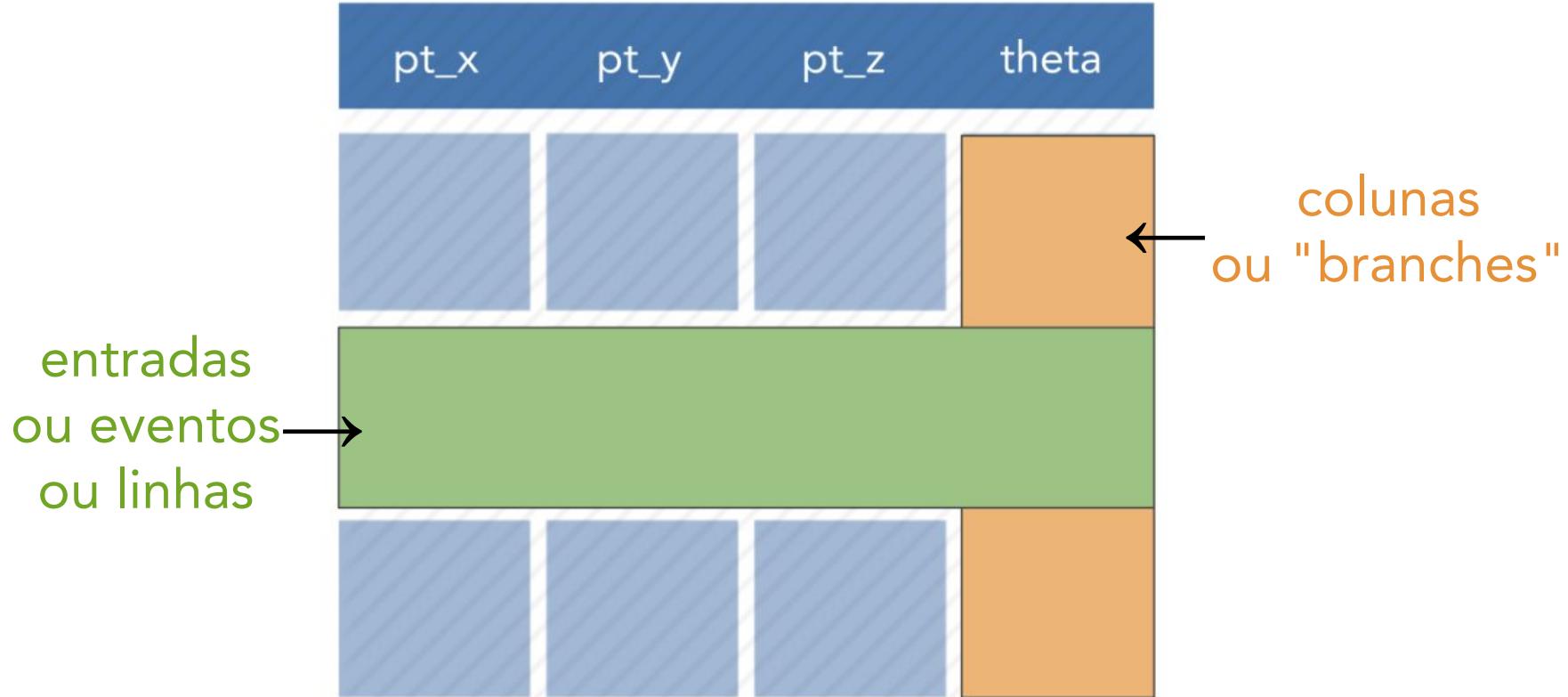
ROOT - TTree

Uma **TTree** é uma estrutura de dados no ROOT usada para representar conjuntos de dados em formato **colunar**. Com ela podemos organizar e manipular várias variáveis ao mesmo tempo.

- Os dados em um TTree são organizados em colunas, também chamadas de branches .
- Cada branch representa uma variável ou um conjunto de variáveis relacionadas, e as colunas podem conter diferentes tipos de dados (como inteiros, floats, vetores, etc.).
- Um TTree pode armazenar qualquer tipo de objeto, permitindo organizar dados complexos (como vetores e classes personalizadas).
- Cada linha no TTree representa uma **entrada** (ou **evento**), com valores correspondentes em cada coluna.
- Uma maneira moderna de interagir com TTree é a interface **RDataFrame**.
 - Ela fornece uma maneira mais intuitiva de manipular e processar dados armazenados em TTrees, permitindo aplicar filtros, transformações e realizar operações de análise de forma mais eficiente.

Obs.: Tutoriais do **RDataFrame**

ROOT - TTree



ROOT - TTree

Uma TTree pode ser criada e preenchida com dados da seguinte maneira:

C++

```
TFile *file = new TFile("myFile.root", "RECREATE");
TTree *tree = new TTree("tree", "Example TTree");
// Variáveis que serão armazenadas
float x, y;
tree->Branch("x", &x, "x/F");
tree->Branch("y", &y, "y/F");
// Preencher o TTree com dados
for (int i = 0; i < 100; i++) {
    x = i * 0.1;
    y = x * x;
    tree->Fill(); }
// Salvar e fechar o arquivo
file->Write();
file->Close();
```



ROOT - TTree

Uma TTree pode ser criada e preenchida com dados da seguinte maneira:

python

```
1 # Criar um novo arquivo ROOT com a opção RECREATE
2 file = ROOT.TFile("myFile.root", "RECREATE")
3
4 # Criar uma TTree
5 tree = ROOT.TTree("tree", "Example TTree")
6
7 # Variáveis que serão armazenadas
8 x = np.zeros(1, dtype=float)
9 y = np.zeros(1, dtype=float)
10
11 # Definir as branches da árvore
12 tree.Branch("x", x, "x/D") # /D para float em double precisão
13 tree.Branch("y", y, "y/D")
14
15 # Preencher o TTree com dados
16 for i in range(100):
17     x[0] = i * 0.1
18     y[0] = x[0] ** 2
19     tree.Fill()
20
21 # Salvar e fechar o arquivo
22 file.Write()
23 file.Close()
24
```

ROOT - TTree - RDataFrame

Importar o ROOT e Criar um RDataFrame

1. Criar o RDataFrame a partir de uma TTree em um arquivo ROOT
2. Aplicar transformações aos dados: filtros, definir novas colunas,...
3. Criar e preencher histogramas

1. Criar RDataFrame

```
import ROOT
```

```
df = ROOT.RDataFrame("t", "f.root")  
h = df.Filter("theta > 5").Histo1D("pt")  
h.Draw()
```

2. Cortes em
theta

3. preencher o
histograma com o
pt



ROOT

Documentação e Links Úteis

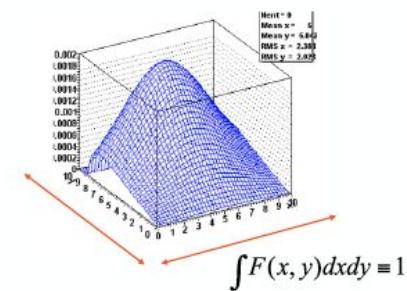
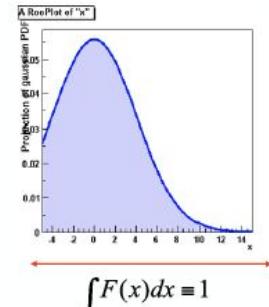
- Página Principal: <https://root.cern.ch/>
- [ROOT - Documentação](#)
- [ROOT - Primer](#)
- [ROOT - Class Index](#)
- [ROOT - Versões](#)
- [ROOT - Pre-Requisitos](#)
- [Como compilar e juntar bibliotecas](#)

RooFit

É um pacote de ferramentas distribuído com o ROOT para modelagem de dados.

- Uma coleção de classes que ampliam o ambiente ROOT para modelar dados
 - É usado para modelar distribuições que são usadas para ajustes e análise estatística de dados.
 - **distribuição do modelo da observável x em termos do parâmetro p**
 - Função Densidade de Probabilidade (P.D.F.): $P(x;p)$
 - A PDF é normalizada sobre o intervalo permitido das observáveis x em relação ao parâmetro p (sobre o qual a PDF depende).

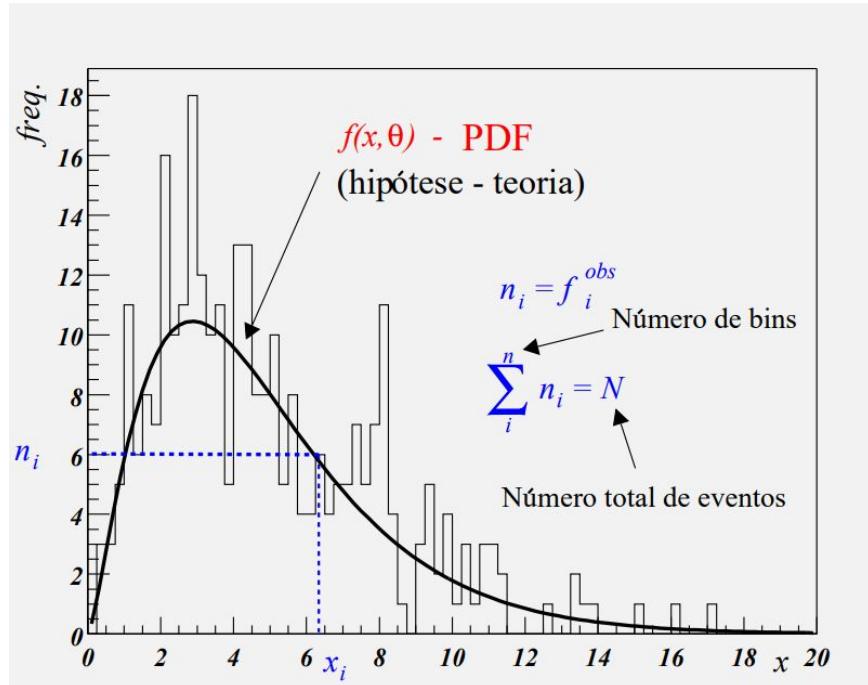
$$\int_{\Omega} P(\vec{x}; \vec{p}) d\vec{x} = 1$$



RooFit - Funções Densidade de Probabilidade

Função Densidade de Probabilidade (pdf) ou Distribuição de Probabilidade: é a função que fornece a probabilidade de se observar um valor x de uma determinada variável contínua dentro de um intervalo infinitesimal $|x, x+dx|$.

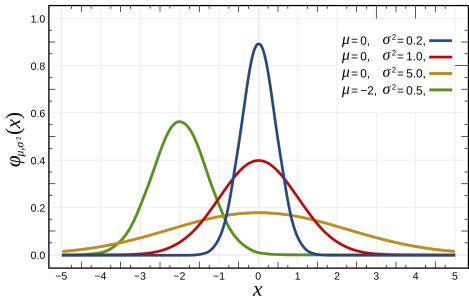
- Em física de altas energias, as medições experimentais frequentemente seguem distribuições contínuas. A pdf descreve como os dados se distribuem ao longo dos possíveis valores.
- Permite ajustar modelos teóricos aos dados experimentais, ajudando a testar a validade das teorias e a entender a física subjacente.



RooFit - Algumas PDFs

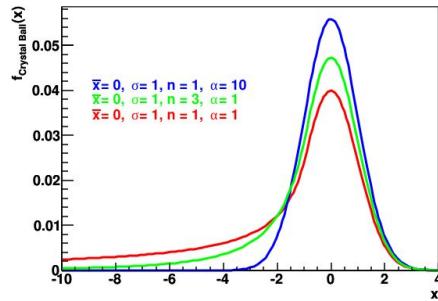
Funções de densidade de probabilidade (PDFs):

Gaussiana



$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\bar{x})^2}{2\sigma^2}\right)$$

Crystal Ball

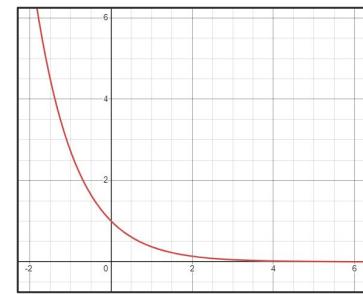


$$f(x) = N \cdot \begin{cases} \exp\left(\frac{-(x-\bar{x})^2}{2\sigma^2}\right), & \text{para } \frac{x-\bar{x}}{\sigma} > -\alpha \\ A \cdot \left(B - \frac{x-\bar{x}}{\sigma}\right)^{-n}, & \text{para } \frac{x-\bar{x}}{\sigma} \leq -\alpha \end{cases}$$

$$A = \left(\frac{n}{|\alpha|}\right)^n \cdot \exp\left(-\frac{|\alpha|^2}{2}\right)$$

$$B = \frac{n}{|\alpha|} - |\alpha|$$

Exponencial



$$f(x) = Ae^{-\lambda x}$$

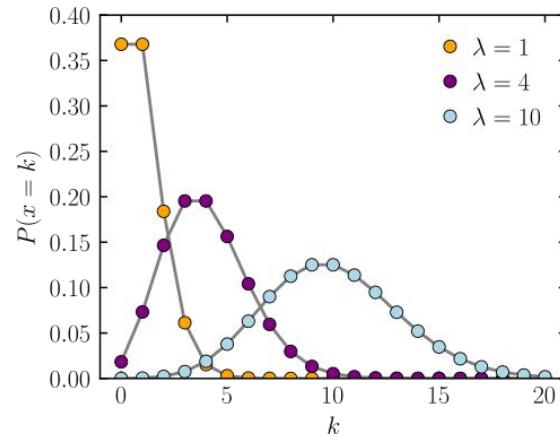
RooFit - Algumas PDFs

Distribuições de Poisson

- é uma distribuição discreta de probabilidade que descreve o número de eventos que ocorrem em um intervalo fixo de tempo ou espaço, dado que esses eventos ocorrem com uma taxa média constante e independentemente uns dos outros.

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

- onde λ é a taxa média de eventos (ou a média da distribuição) e k é o número de eventos observados.
- É ideal para modelar a contagem de eventos raros ou pouco frequentes em física de altas energias, como o número de partículas detectadas em um detector.



RooFit - estimativa de parâmetros

- O modelo dos dados observados é expresso usando a Função Densidade de Probabilidade(PDF)

$$p(x|\theta)$$

ex.: a probabilidade de observar um evento em um bin de um histograma $P_{bin} = \int_{bin} p(x|\theta)dx$

-A PDF é normalizada a 1 quando integrada em todo o espaço da amostra Ω $\int_{\Omega} p(x|\theta)dx = 1$

- Para estimar os parâmetros usamos **M. Likelihood Function (Função de Verossimilhança)**

$$\mathcal{L}(x_1, x_2, \dots x_N; \theta) = \prod_{i=1}^N p(x_i|\theta)$$

- Aplica-se o log(ln) na likelihood-function. Por conveniência usa-se o negativo log-likelihood function e encontramos o minimum global

$$-\ln \mathcal{L}(x_1, x_2, \dots x_N; \theta) = -\sum_{i=1}^N \ln p(x_i|\theta) \quad \frac{\partial(-\ln \mathcal{L})}{\partial \theta} = 0$$

ML estendida:

$$\mathcal{L}_{ext}(N_{exp}; \theta) = \frac{e^{-N_{exp}} \cdot N_{exp}^{N_{obs}}}{N_{obs}!} \cdot \prod_{i=1}^{N_{obs}} p(x_i; \theta)$$

RooFit

- RooFit fornece funcionalidades para construir as PDFs
 - construção de modelo complexo
 - composição com produto, adição, ...
- Todos os modelos fornecem a funcionalidade para:
 - ajuste ***maximum(minimum) likelihood***
 - gerador de amostras a partir de uma PDF
 - visualização



RooFit

Conceitos matemáticos são representados como objetos em C++

Conceitos matemáticos	Classe no RooFit
Varável	RooRealVar
Função	RooAbsReal
PDF	RooAbsPdf
Ponto espacial	RooArgSet
Integral	RooRealIntegral
Listas de pontos espaciais	RooAbsData

RooFit - exemplo simples

```
[1]: # Importar ROOT e configurações
import ROOT
%jsroot on

[2]: #Definir a variável
x = ROOT.RooRealVar("x", "x", 0, -10, 10)

[3]: #definir a média e a largura
mean = ROOT.RooRealVar("mean", "Mean of Gaussian", 0, -10, 10)
width = ROOT.RooRealVar("width", "Width of Gaussian", 3, 0.1, 10)

[4]: #criar a função
g = ROOT.RooGaussian("g", "Gaussian", x, mean, width)

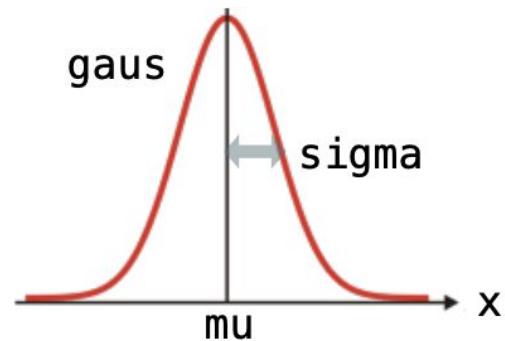
[5]: #criar o objeto RooPlot para a variável
frame = x.frame()

[6]: # Plotar a função gaussiana no frame
g.plotOn(frame)

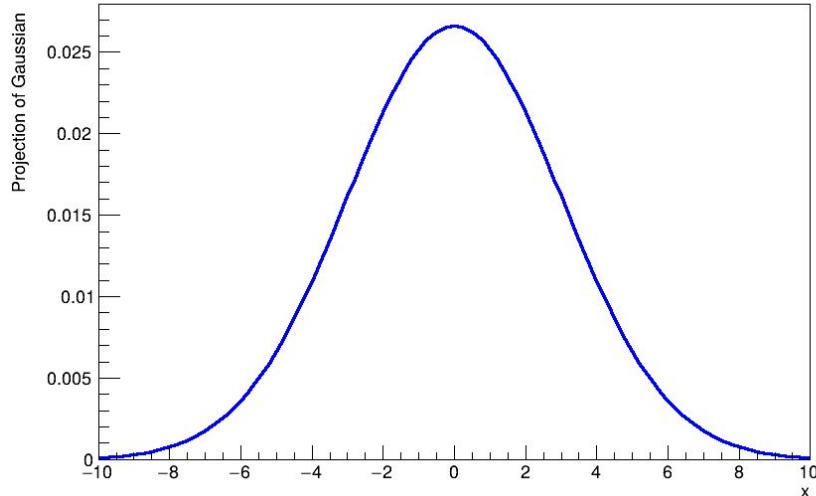
[6]: <cppyy.gbl.RooPlot object at 0x55eb6edcabe0>

*[8]: # Criar o canvas para visualizar o gráfico e draw
c1 = ROOT.TCanvas("c1", "Gaussian Plot", 900, 600)
frame.Draw()
c1.Draw()
c1.SaveAs("gauss2.png")
```

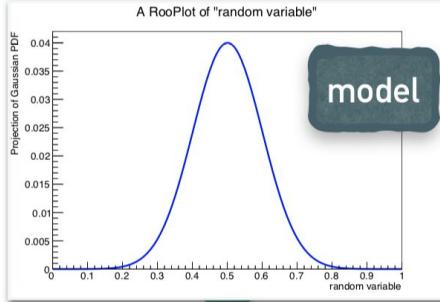
Info in <TCanvas::Print>: png file gauss2.png has been created



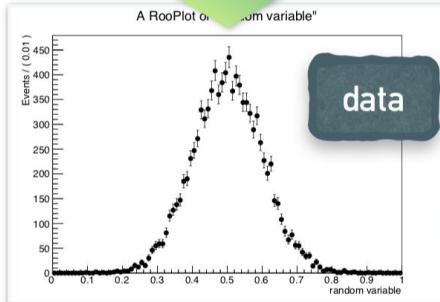
A RooPlot of "x"



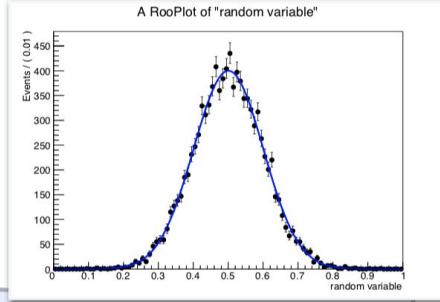
RooFit - Geração e Ajuste



```
model.generate(x, 10000);
```



```
RooPlot* fm = x.frame();
data->plotOn(fm);
model.plotOn(fm);
fm->Draw();
```



COVARIANCE MATRIX CALCULATED SUCCESSFULLY
 FCN=-8863.01 FROM HESSE STATUS=OK 10 CALLS 34 TOTAL
 EDM=1.57332e-06 STRATEGY= 1 ERROR MATRIX ACCURATE
 EXT PARAMETER

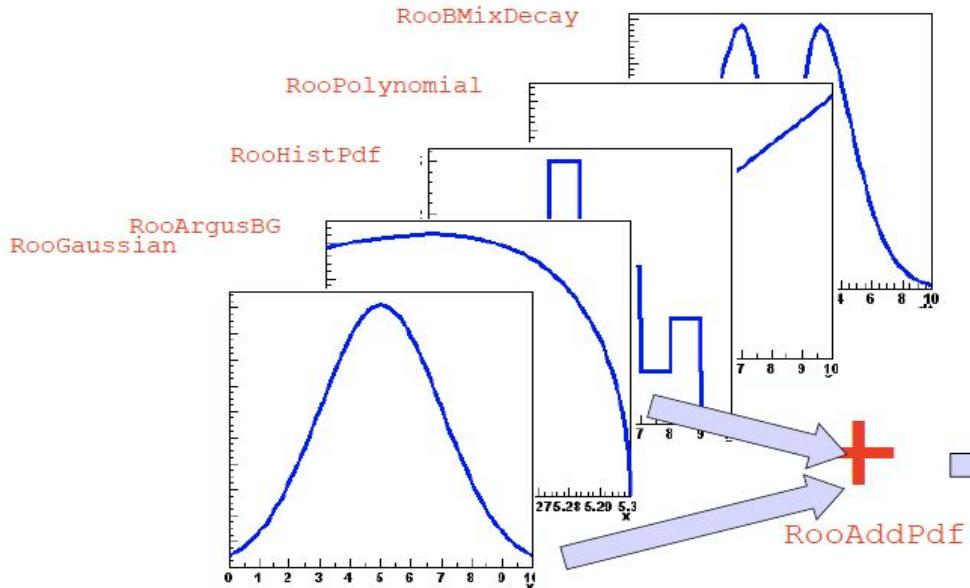
NO.	NAME	VALUE	ERROR
1	mu	5.00665e-01	9.97372e-04
2	sigma	9.97366e-02	7.05314e-04

```
RooFitResult *res = model.fitTo(*data, ...);
```

Essa classe armazena os resultados de um ajuste de modelo, incluindo os parâmetros ajustados, suas incertezas, a matriz de covariância, entre outros.

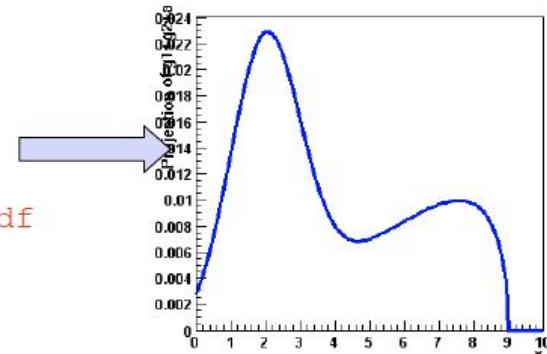
RooFit - construindo modelos

- Os modelos mais realistas são construídos como a soma de uma ou mais p.d.f.s (ex.: sinal e fundo (*background*))
- Facilitado por meio de classes operador p.d.f `RooAddPdf`



Constrói a soma de N PDFs com coeficientes N-1:

$$S = c_0 P_0 + c_1 P_1 + c_2 P_2 + \dots + c_{n-1} P_{n-1} + \left(1 - \sum_{i=0, n-1} c_i\right) P_n$$

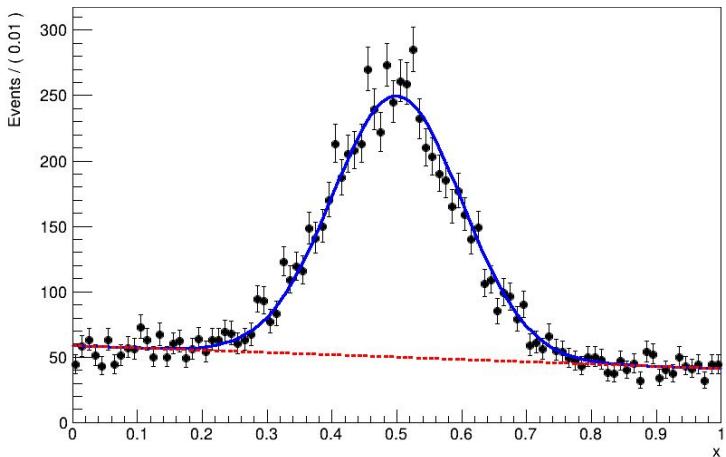


RooFit - PDF com Componentes de Sinal e Fundo

Definir uma função de densidade de probabilidade que represente o modelo dos dados:

- Definir uma PDF com componentes de sinal e fundo;
- Construir o modelo, gerar dados simulados (toys);
- ajustar e plotar os resultados.

$$\ln \mathcal{L} = \sum_{i=1}^N \ln [f \cdot \text{Gauss}(x_i|\mu,\sigma) + (1-f) \cdot \text{Linear}(x_i|\text{slope})]$$



```
[1]: import ROOT
%jsroot on
from ROOT import RooFit
from ROOT import RooArgSet
from ROOT import RooArgList

[2]: x = ROOT.RooRealVar("x","x",0.0,1.)
mu = ROOT.RooRealVar("mean","mean",0.5,0,1.0)
sigma = ROOT.RooRealVar("sigma","sigma",0.1, 0.0,0.3)

[3]: gaus = ROOT.RooGaussian("gaus","gauss",x,mu,sigma) #pdf 1

[4]: slope = ROOT.RooRealVar("slope","slope", -0.3,-10.,10.)

[5]: linear = ROOT.RooPolynomial("linear","linear", x, ROOT.RooArgSet(slope)) #pdf 2

[6]: #Adicionando as pds: gaus + linear
fracao = ROOT.RooRealVar("fracao","fração da Gaussiana",0.5,0.,1.)
modelo = ROOT.RooAddPdf("modelo", "Soma das PDFs",ROOT.RooArgList(gaus,linear), ROOT.RooArgList(fracao))

[7]: dataset = modelo.generate(ROOT.RooArgSet(x), 10000)

[8]: c = ROOT.TCanvas("c","c",900,600)

frame = x.frame()
dataset.plotOn(frame)
modelo.plotOn(frame)
modelo.plotOn(frame,RooFit.Components(linear),RooFit.LineStyle(7),RooFit.LineColor(ROOT.kRed))
result = modelo.fitTo(dataset, RooFit.Save())

result.Print()

[9]: frame.Draw()
c.Draw()
c.SaveAs("exemploCombinandoPDF.png")
```

Teste do χ^2

Os dados observados são consistentes com o modelo proposto?

- O teste do χ^2 é uma ferramenta estatística usada para avaliar a adequação de um modelo aos dados observados.
- Ele mede a discrepância entre os dados observados e os valores esperados de um modelo, ajudando a verificar a qualidade do ajuste.
- É calculado por:
$$\chi^2 = \sum_{i=1}^n \frac{(O_i - E_i)^2}{E_i}$$
- onde O_i são os valores observados e E_i são os valores esperados.

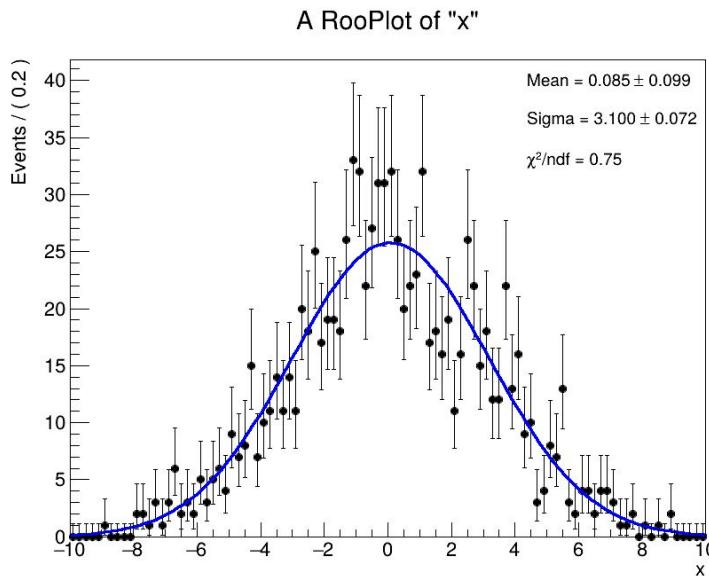
Teste do χ^2

- O melhor ajuste é obtido quando o valor de χ^2 dividido pelo número de graus de liberdade (ndf) se aproxima de 1.
- Graus de liberdade referem-se ao número de valores independentes que podem variar em uma análise.
- Em um ajuste de curva, os graus de liberdade são geralmente dados pelo número de pontos de dados menos o número de parâmetros ajustados.

RooFit - validação do ajuste

Algumas ferramentas implementadas no [RooPlot](#) são capazes de calcular o χ^2/ndf de uma curva com relação aos dados

```
double chi2 = frame->chisquare(nFloatParam);
```



```
# Realizar ajuste e salvar o resultado
fit_result = gauss.fitTo(data,
ROOT.RooFit.Save())
# Obter número de parâmetros flutuantes
nFloatParam =
fit_result.floatParsFinal().getSize()
# Calcula o chi2 para o frame com o número de parâmetros
# flutuantes
chi2 = frame.chiSquare(nFloatParam)
```

RooFit - validação do ajuste - distribuição Pull

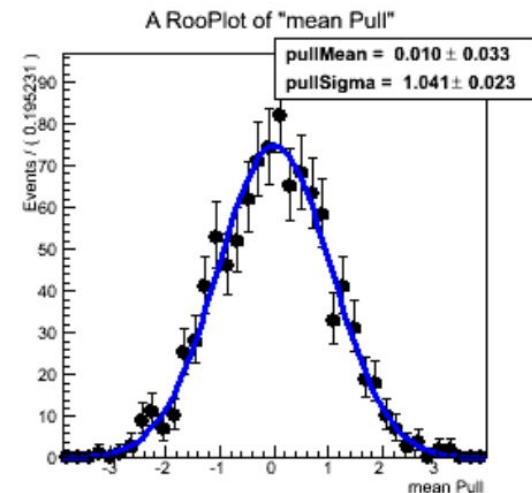
- **Pull distribution**

Definição:

$$pull(N_{sig}) = \frac{N_{sig}^{fit} - N_{sig}^{true}}{\sigma_N^{fit}}$$

```
pull_hist = frame.pullHist()  
# Equivalente ao frame->pullHist() em C++
```

- Propriedades da pull:
 - Mean é 0 se não há bias
 - Width é 1 se o erro está correto
 - Nesse exemplo: sem bias, erro correto dentro da precisão estatística de estudo

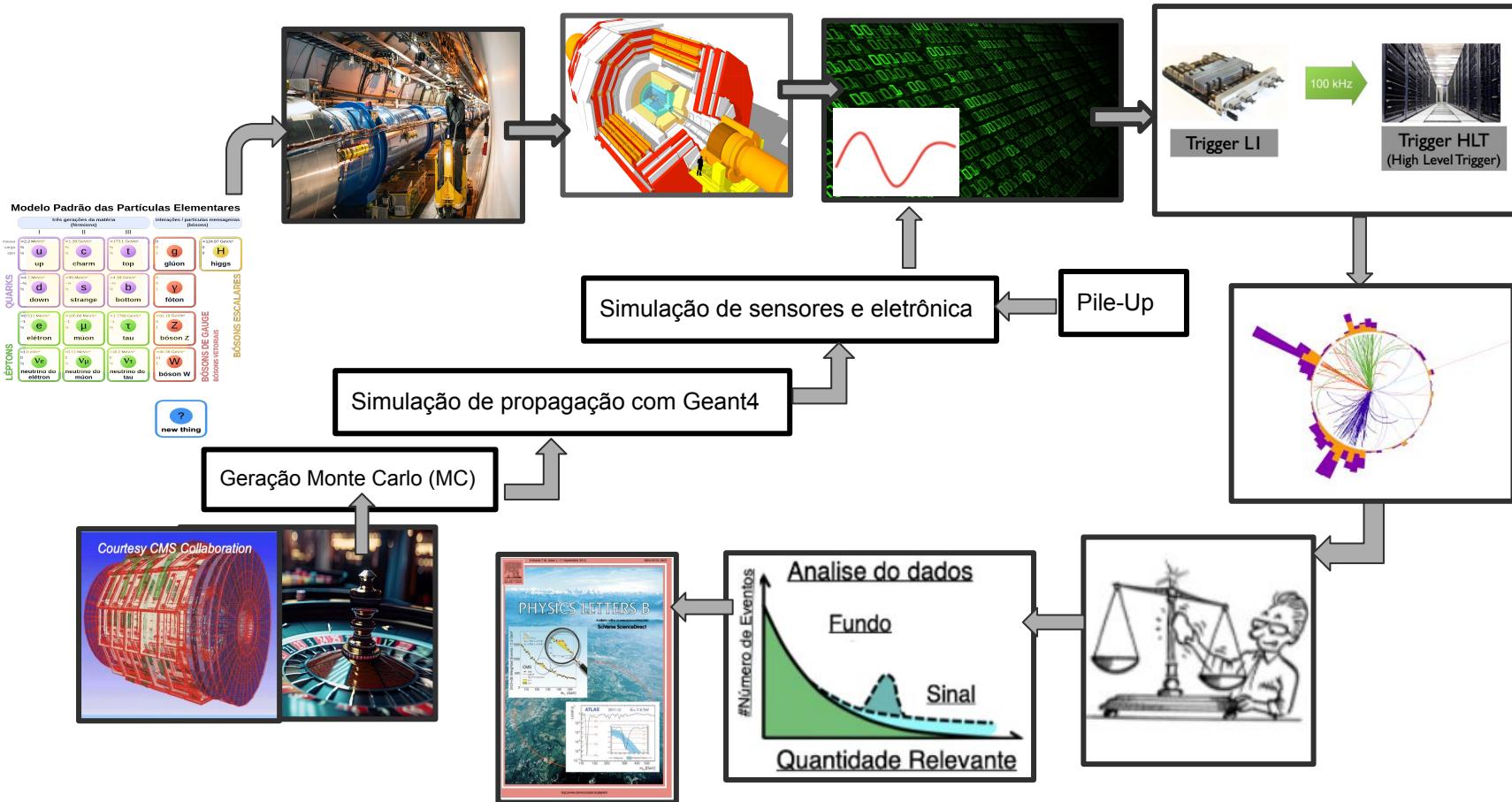


RooFit - Documentação e Links

Ponto inicial: <https://root.cern/manual/roofit/>

- Manual: https://root.cern/download/doc/RooFit_Users_Manual_2.91-33.pdf
- [Quick Start Guide](#) (20 pages, recent):
- Tutorial-macros (also in \$ROOTSYS/tutorials/roofit):
https://root.cern/doc/master/group__tutorial__roofit.html
- Mais de 200 slides do **Wouter Verkerke** documentando todos os recursos do RooFit, eles estão disponíveis na “French School of Statistics 2008”:
<http://indico.in2p3.fr/getFile.py/access?contribId=15&resId=0&materia lId=slides&confId=750>

Como você obtém os dados para fazer a análise?



Processamento e formato dos dados no CMS

O processamento dos dados coletados envolve várias etapas, cada uma correspondendo a uma "camada de dados" ou data tier, com tempos de processamento diferentes:

RAW → RECO/AOD:

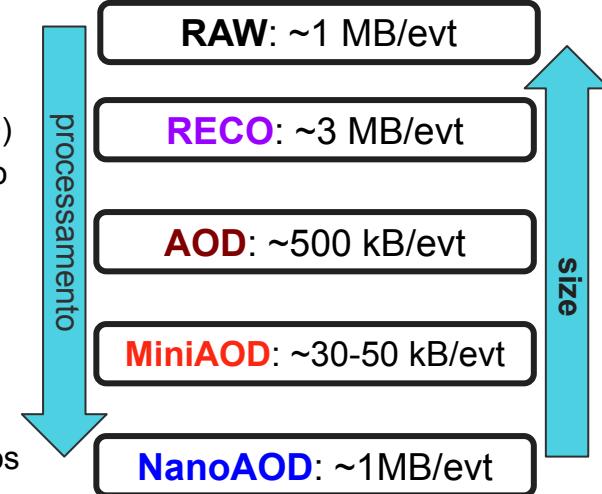
- Os dados brutos (RAW) são processados em dados reconstruídos (RECO) e armazenados no formato **AOD** (Analysis Object Data). O processamento deste passo leva cerca de 30 segundos por evento.

AOD → MiniAOD:

- O formato **AOD** é transformado em **MiniAOD**, um formato mais leve que mantém as informações mais relevantes para a análise física, com menor uso de armazenamento. Este processamento leva cerca de 1 a 2 segundos por evento.

MiniAOD → NanoAOD:

- A última etapa do processamento é a conversão de **MiniAOD** para **NanoAOD**, um formato extremamente compacto, ideal para análises finais e estudos de grande escala. Esse processamento é mais rápido, levando cerca de 0,1 segundos por evento.



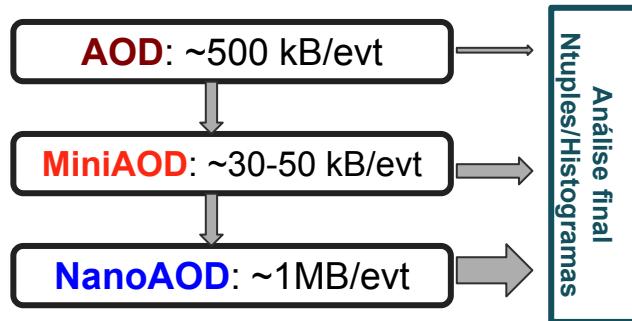
Qual é o formato dos dados utilizados em análises?

Existem várias camadas/níveis de dados que podem ser utilizadas para análise, cada uma servindo a diferentes propósitos, são elas:

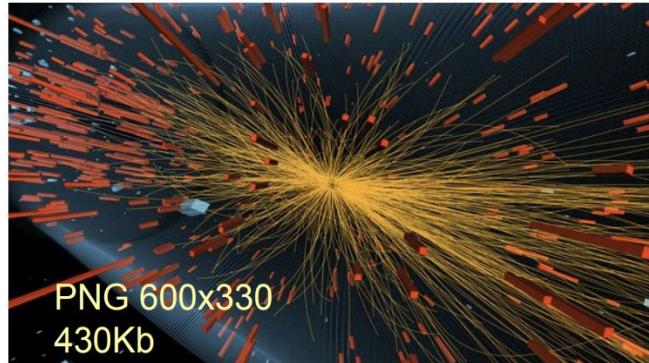
AOD: contém informações detalhadas de objetos físicos reconstruídos. É usado para análises que requerem todos os detalhes dos eventos.

MiniAOD: um formato mais leve que mantém as informações essenciais, bastante utilizado para a maioria das análises de física.

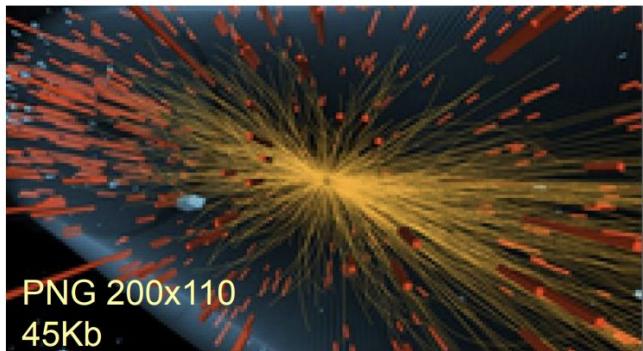
NanoAOD: um formato ainda mais compacto, utilizado para análises mais rápidas e eficientes, contendo apenas as informações absolutamente necessárias.



AOD vs MINIAOD vs NANOAOD em imagens



AOD



miniAOD



nanoAOD



MiniAOD vs NanoAOD

MiniAOD

É um formato **EDM** (Event Data Model) **object type**.

Ex.: `pat::Electron, pat::Muon, ...`
(Objetos do tipo C++)

Oferece informações mais completa para permitir desenvolvimentos futuros

NanoAOD

É um formato flat NTuple (colunar), que pode ser lido diretamente com ROOT/pyROOT, com ROOT usando RDataFrame, uproot/AwkwardArrays/Coffea, uproot/pandas/...

Ex.: `Int_t nMuons`
`Float_t Muon_pt[nMuons]`
`Float_t Muon_eta[nMuons]`

NanoAOD - como os eventos são organizados?

Um Ntuple NANOAOD é um formato de arquivo que contém dados de eventos físicos. Ele é organizado usando um formato chamado TTree do ROOT, que é uma estrutura de dados colunar, ou seja, os dados são armazenados em colunas (ramos), onde cada ramo contém um tipo de informação sobre um evento físico.

Branch name	Type	Data type	Function	Example
nObject	Scalar	Unsigned integer	Number of objects in collection Object	nMuon, nJet, nGenPart
Object_var[i]	Array	Any	Attribute var of the i -th object in collection Object	Muon_pt[i], Jet_mass[i], GenPart_pdgId[i]
Object_otherIdx[i]	Array	Signed integer	Index of the object from collection Other if it is linked to the i -th object from collection Object, and -1 otherwise	Muon_jetIdx[i], Muon_genPartIdx[i]

[Karl Ehatäht](#)

NanoAOD - como os eventos são organizados?

O arquivo contém uma TTree principal chamada “Events” e outras TTrees auxiliares, como as: LuminosityBlocks, Runs, MetaData e ParameterSets

name	typename	interpretation
nMuon	uint32_t	AsDtype('>u4')
Muon_dxy	float[]	AsJagged(AsDtype('>f4'))
Muon_dxyErr	float[]	AsJagged(AsDtype('>f4'))
Muon_dxybs	float[]	AsJagged(AsDtype('>f4'))
Muon_dz	float[]	AsJagged(AsDtype('>f4'))
Muon_dzErr	float[]	AsJagged(AsDtype('>f4'))
Muon_eta	float[]	AsJagged(AsDtype('>f4'))

CMS Open Data

- O CMS Open Data oferece conjuntos de dados abertos do experimento CMS no LHC para a comunidade científica e o público.
- Esses dados incluem eventos de colisão de partículas reais e simulados, e estão disponíveis para pesquisa, ensino e aprendizado.
- Os dados estão armazenados em formatos como ROOT e podem ser acessados via Uproot e outras ferramentas no ecossistema Python.

/Tau/Run2016H-UL2016_MiniAODv2_NanoAODv9-v1/NANOAOAD

Tau primary dataset in NANOAOAD format from RunH of 2016. Run period from run number 281613 to 284044. The list of validated runs, which must be applied to all analyses, either with the full validation...

[Dataset](#) [Collision](#) [CMS](#)

/SinglePhoton/Run2016H-UL2016_MiniAODv2_NanoAODv9-v1/NANOAOAD

SinglePhoton primary dataset in NANOAOAD format from RunH of 2016. Run period from run number 281613 to 284044. The list of validated runs, which must be applied to all analyses, either with the full...

[Dataset](#) [Collision](#) [CMS](#)

/SingleMuon/Run2016H-UL2016_MiniAODv2_NanoAODv9-v1/NANOAOAD

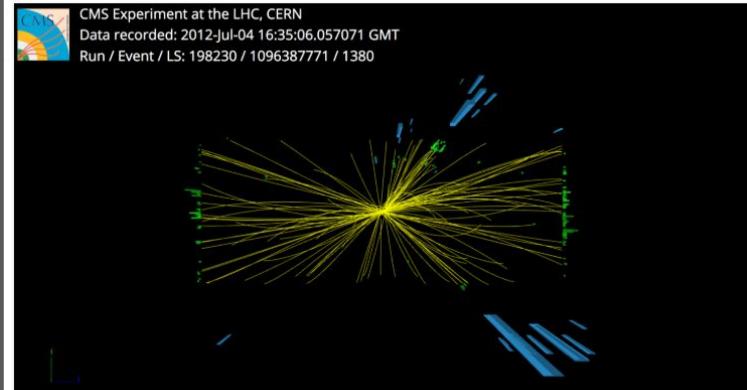
SingleMuon primary dataset in NANOAOAD format from RunH of 2016. Run period from run number 281613 to 284044. The list of validated runs, which must be applied to all analyses, either with the full v...

[Dataset](#) [Collision](#) [CMS](#)

CMS releases more than one petabyte of open data

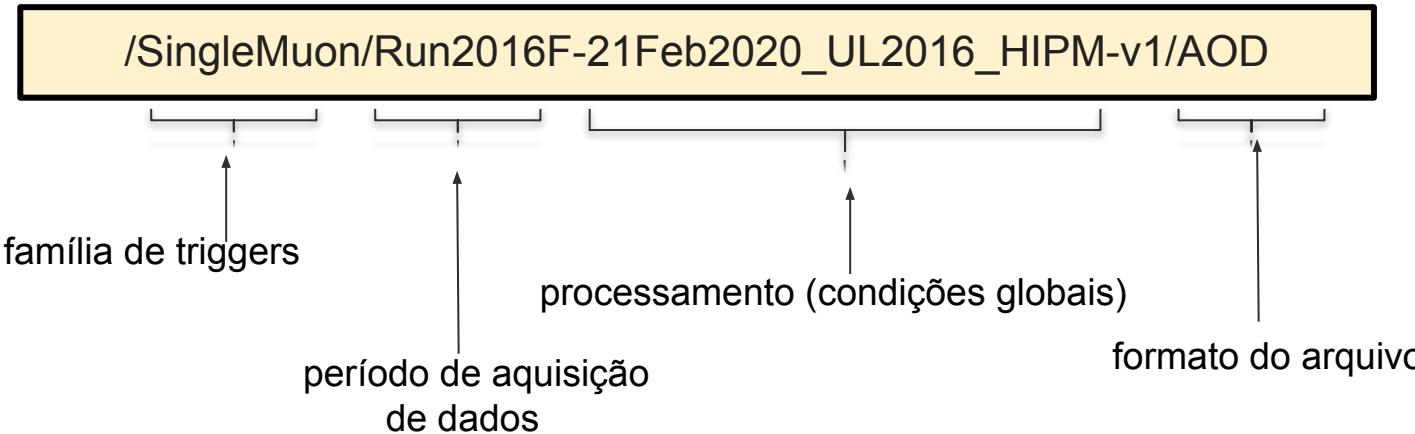
This release includes datasets that were used to discover the Higgs boson

20 DECEMBER, 2017 | By Achintya Rao



<https://opendata.cern.ch/docs/cms-getting-started-nanoaod>

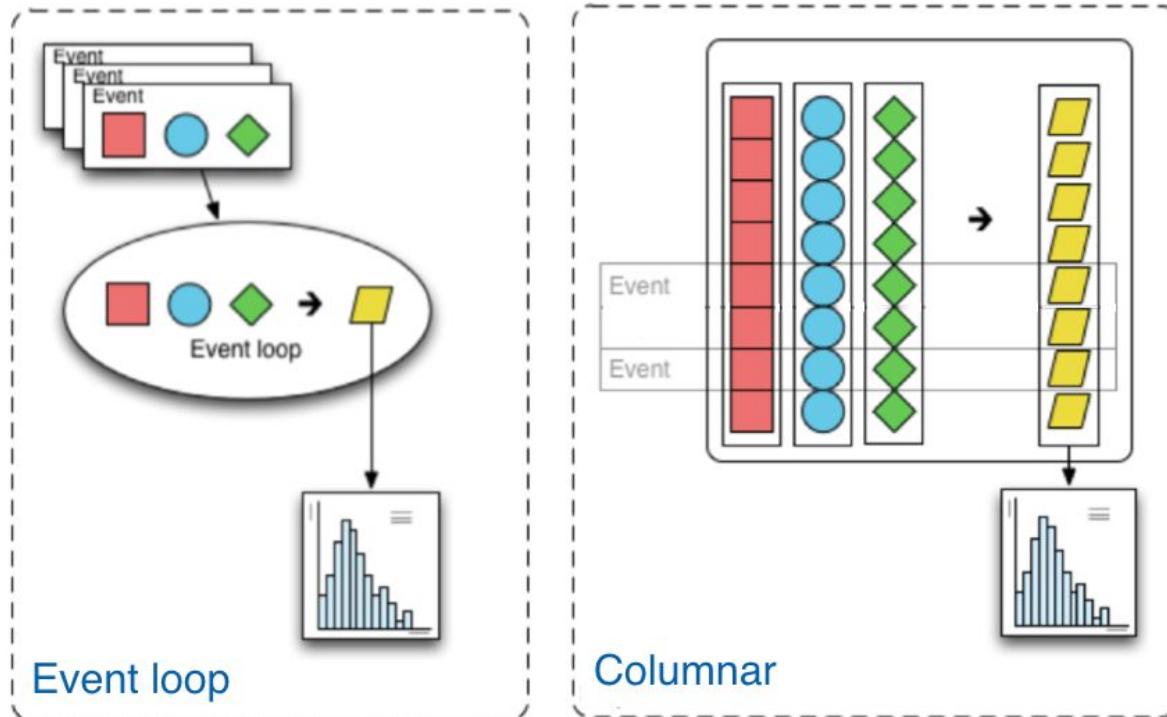
Nomenclatura dos dados reais



Certificação de Dados:

- **Certificação de dados ruins (bad runs):** quando estamos processando dados reais, é comum precisarmos filtrar os dados para excluir runs ou lumisections que foram marcadas como ruins durante a certificação de dados. Isso garante que você esteja trabalhando apenas com dados de boa qualidade.
- Um arquivo chamado "**Golden JSON**" é fornecido para essa finalidade. Este arquivo contém uma lista de runs e lumisections certificadas como boas, e pode ser usado para filtrar automaticamente os dados ruins.

Técnicas de análise em FAE

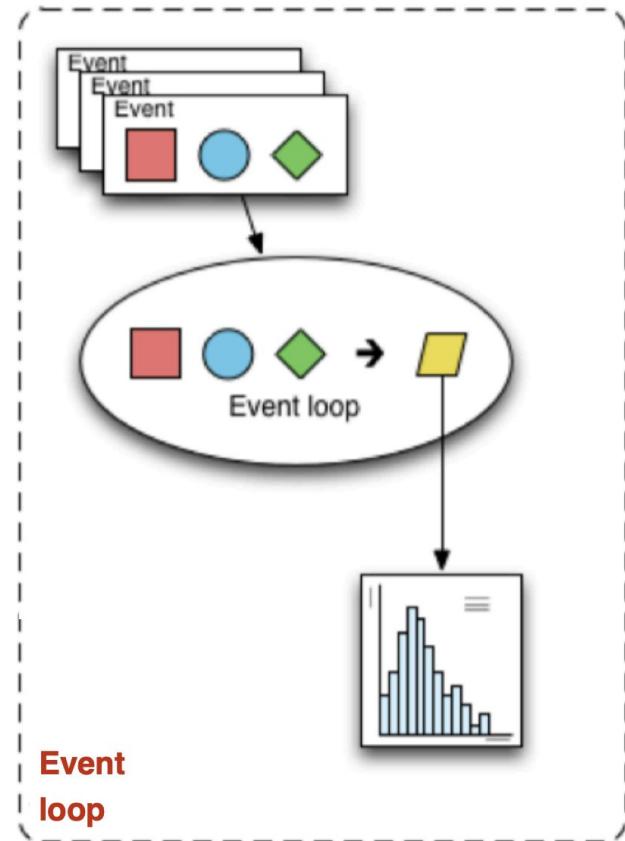


Análise em loop de eventos - Análise orientada a registros

- Carregar valores relevantes
- Avaliar expressões
- Armazenar os valores derivado
- Repetir o processo

Os dados são processados evento a evento, realizando cálculos e armazenando resultados para análise posterior. Cada evento é processado individualmente.

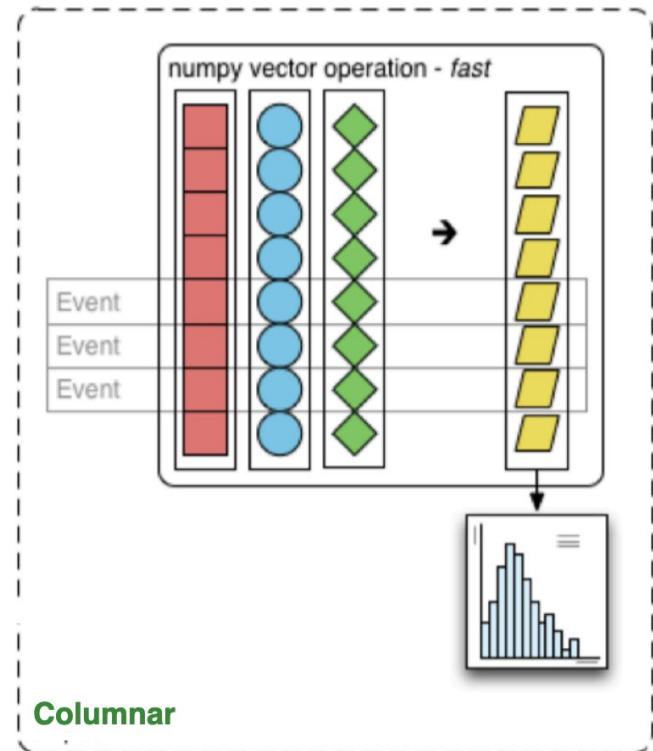
A análise é estruturada como um loop que percorre os eventos um por um. Cada "registro" ou evento é carregado na memória individualmente, e o código de análise opera sobre os campos ou propriedades desse evento, como os momentos das partículas reconstruídas.



Análise de forma Colunar

- Representação de dados em formato colunar:
 - Os dados são carregados em colunas contíguas, onde cada coluna corresponde a uma variável ou observável e cada linha corresponde a um evento.
 - Você trabalha diretamente com colunas de dados, representando múltiplos eventos de uma vez.
- Análise Colunar:
 - Expressões vetorizadas, as operações de vetor podem ser aplicadas diretamente nas colunas de dados usando bibliotecas como NumPy.
 - Sem loops explícitos, as operações são aplicadas diretamente em vetores (ou colunas).
 - Armazenar valores derivados em novas colunas.

Aproveitando o processamento em vetores, torna o processo mais rápido e escalável.



Exemplos

```
void MyClass::Loop() {
    size_t nEvents;
    // load...

    for (Long64_t iEvent=0; iEvent<nEvents; iEvent++) {
        double MET_pt;
        int nElectron;
        double * Electron_pt;
        double * Electron_eta;
        // load...

        if ( MET_pt > 100. ) continue;

        for(size_t iEl=0; iEl<nElectron; ++iEl) {
            if ( Electron_pt[iEl] > 30. ) {
                hist->Fill(Electron_eta[iEl]);
            }
        }
    }
}
```

Event loop

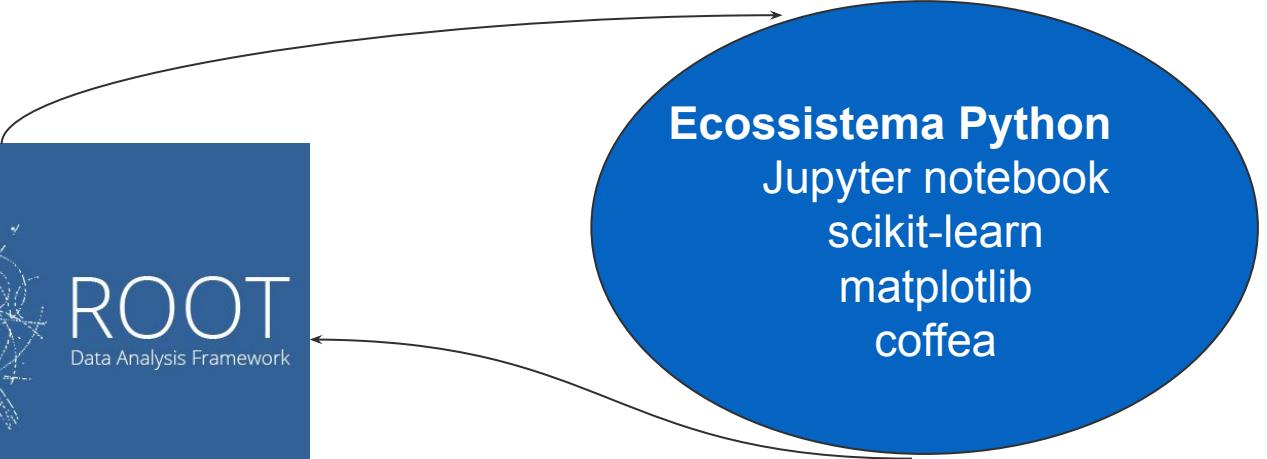
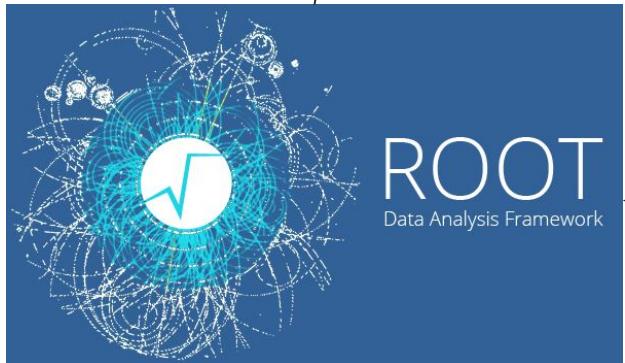
```
cut = (events.MET.pt < 100.) & (events.Electron.pt > 30.)
hist.fill(eta=events.Electron.eta[cut].flatten())
```

Columnar

from:Lindsey Gray

Análise de forma Colunar - Ferramentas

As ferramentas como **uproot** e **awkward array** são usadas para ler e processar os dados nesse formato. Essas bibliotecas permitem trabalhar com arquivos ROOT (TTree) como se fossem arrays nativos no Python, possibilitando operações colunares diretamente.



The diagram illustrates the integration between the ROOT Data Analysis Framework and the Python ecosystem. On the left, the ROOT logo is shown. A curved arrow originates from the top right of the ROOT logo and points towards a large blue circle on the right. Another curved arrow originates from the bottom right of the blue circle and points back towards the bottom left of the ROOT logo, indicating a bidirectional relationship or data flow between the two environments.

Ecossistema Python
Jupyter notebook
scikit-learn
matplotlib
coffea

Análise de forma Colunar - Ferramentas

- Visualização
- Algoritmos
- Array API
- Ingestão de Dados
- Agendador de tarefas
- Provisão de recursos



Análise de forma Colunar - uproot

- O uproot é um módulo Python que permite ler e escrever arquivos no formato ROOT.
- A dependência obrigatória, além do próprio Python, é o NumPy, que é a biblioteca mais popular para manipulação de arrays de dados em Python.
- Ele é capaz de processar grandes volumes de dados de forma rápida.

```
import uproot
file = uproot.open("data.root")
file
```



Saída: <ReadOnlyDirectory '/' at 0x(some hexadecimal number here)>

Análise de forma Colunar - uproot

Como navegar e acessar o conteúdo do arquivo ROOT usando o uproot?

Ex.:

Listar o conteúdo do arquivo:

file.keys()

```
# Saída: ['Events;1']
```

Verificar o tipo de objeto no arquivo:

fileclassnames()

```
# Saída: {'Events;1': 'TTree'}
```

Acessar o conteúdo do arquivo (file['key']):

file['Events']

```
# Saída: <TTree 'Events' (6 branches) at 0x(hexadecimal number)>
```



Como acessar e manipular a TTree do arquivo ROOT usando o uproot?

```
tree = file['Events']

tree.keys()

#Saída: ['nMuon', 'Muon_pt', 'Muon_eta', 'Muon_phi', 'Muon_mass',
'Muon_charge']

tree.array()

#Saída: <Array [{nMuon: 2, Muon_pt: [10.8, ... -1, 1]}] type='100000 * uint32,...'>

branches = tree.array()

branches['nMuon']

#Saída: <Array [2, 2, 1, 4, 4, 3, ... 0, 3, 2, 3, 2, 3] type='100000 * uint32'>
```

Quando usamos `tree.arrays()`, ele retorna um objeto Awkward Array.

Como acessar e manipular a TTree do arquivo ROOT usando o uproot?

```
branches['Muon_pt']
```

```
#Saída: <Array [[10.8, 15.7], ... 11.4, 3.08, 4.97]] type='100000 * var * float32'>
```

São arrays irregulares (jagged array), pois o número de entradas ('nMuon') varia por evento.

Para acessar os dados de um evento específico, você pode indexar o array como faria com qualquer array normal.

```
branches['Muon_pt'][0]
```

```
#Saída:<Array [10.8, 15.7] type='2 * float32'>
```

Análise de forma Colunar - Awkward Array

Os dados em altas energias são estruturados de forma irregulares:

- Não pode ser representado como uma tabela “retangular”.
 - Em diferentes eventos - números variáveis de objetos, como $\mu/e/jatos/...$

A representação de Arrays Irregulares (Jagged Arrays):

Muon pt: table			
Event 1	40.2	25.6	10.2
Event 2	71.1	35.7	
Event 3	52.3		
Event 4	34.5	15.7	

Awkward Array

Muon pt: jagged array							
	40.2	25.6	10.2		71.1	35.7	
	Event 1				Event 2		
	52.3				52.3		

Análise de forma Colunar - Awkward Array

Como aplicamos seleções em arrays irregulares (jagged arrays)?

- Usando máscaras.

$$\text{mu_pt} = [[\ 40.2 \quad 25.6 \quad 10.2 \] [\ 71.1 \quad 35.7 \] [\ 52.3 \] [\ 34.5 \quad 15.7 \]]$$

$$\text{mask} = (\text{mu_pt} > 30) = [[\ T \quad F \quad F \] [\ T \quad T \] [\ T \] [\ T \quad F \]]$$

$$\text{mu_pt[mask]} = [[\ 40.2 \] [\ 71.1 \quad 35.7 \] [\ 52.3 \] [\ 34.5 \]]$$

A máscara foi aplicada a cada mûon em cada evento de maneira vetorizada.

Análise de forma Colunar - Awkward Array

```
✓ 1 !pip install awkward
  2 import awkward as ak
  3 import numpy as np

  ↗ Show hidden output

✓ [9] 1 mu_pt = ak.Array([[40.2, 25.6, 10.2], # Evento 1: três múons
  2                 [71.1, 35.7], # Evento 2: dois múons
  3                 [52.3], # Evento 3: um múon
  4                 [34.5, 15.7]]) # Evento 4: dois múons

✓ [10] 1 mask_mupt = mu_pt > 30
  2 mu_pt_sel = mu_pt[mask_mupt]

✓ [11] 1 print("Array original:", mu_pt)
  2 print("\nMáscara aplicada:")
  3 print(mask_mupt)
  4 print("\nArray após a aplicação da máscara:")
  5 print(mu_pt_sel)

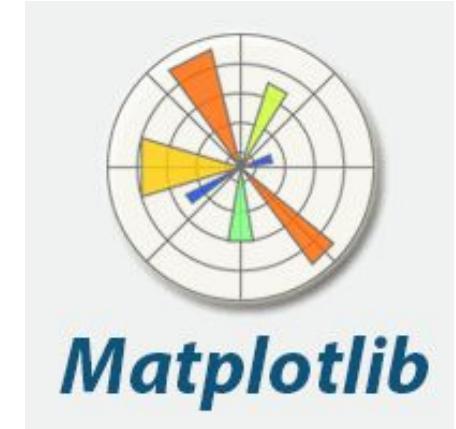
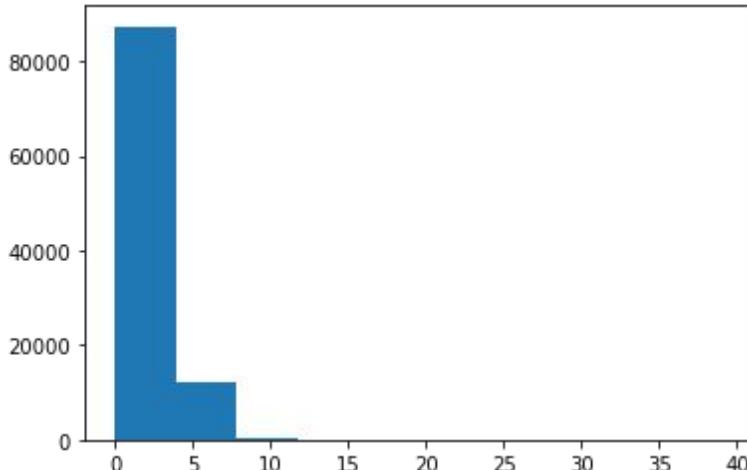
  ↗ Array original: [[40.2, 25.6, 10.2], [71.1, 35.7], [52.3], [34.5, 15.7]]

  Máscara aplicada:
  [[True, False, False], [True, True], [True], [True, False]]

  Array após a aplicação da máscara:
  [[40.2], [71.1, 35.7], [52.3], [34.5]]]
```

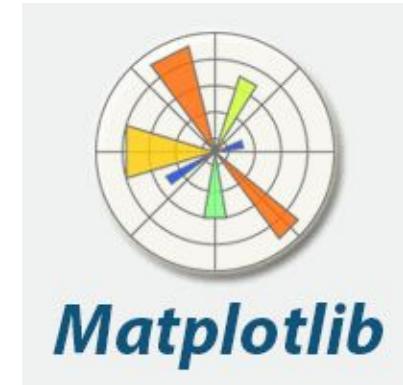
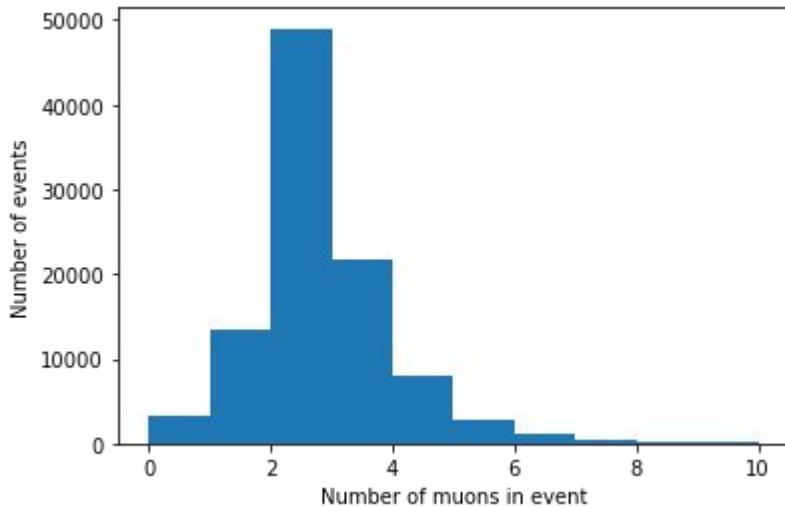
Análise de forma Colunar - Matplotlib

```
import matplotlib.pyplot as plt
plt.hist(branches['nMuon'])
#Saída: (array([8.7359e+04, 1.2253e+04, 3.5600e+02,
2.8000e+01, 2.0000e+00, 1.0000e+00, 0.0000e+00,
0.0000e+00, 0.0000e+00, 1.0000e+00]), array([ 0., 3.9,
7.8, 11.7, 15.6, 19.5, 23.4, 27.3, 31.2, 35.1, 39. ]),
<a list of 10 Patch objects>)
```



Análise de forma Colunar - Matplotlib

```
plt.hist(branches['nMuon'], bins=10, range=(0, 10))  
plt.xlabel('Number of muons in event')  
plt.ylabel('Number of events')  
plt.show()
```

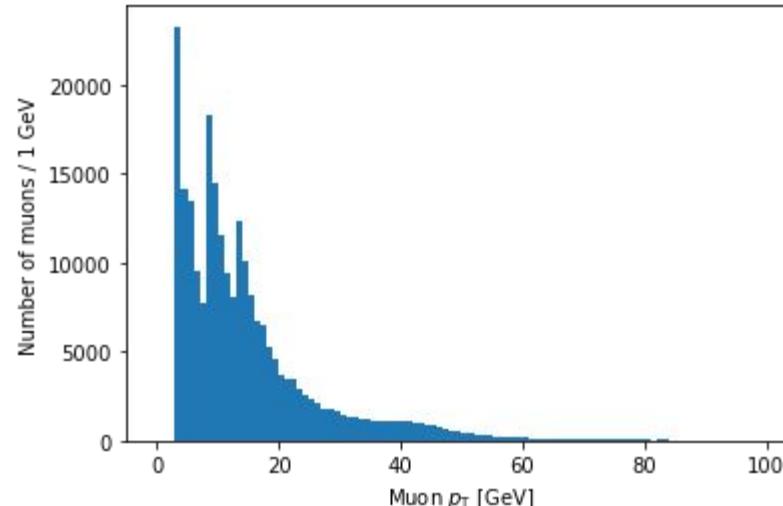


Análise de forma Colunar - Matplotlib

Para fazer um histograma de um array irregular (jagged array), precisamos converter o array 2D para 1D usando o Awkward Array com a função `ak.flatten()`. Ex.:

```
import awkward as ak

plt.hist(ak.flatten(branches['Muon_pt']), bins=100, range=(0, 100))
plt.xlabel('Muon  $p_T$  [GeV]')
plt.ylabel('Number of muons / 1 GeV')
plt.show()
```



Análise de forma Colunar - Contagem

```
len(branches['Muon_pt']) #número total de eventos
```

#Saída: 100000

```
len(ak.flatten(branches['Muon_pt'])) #número total de múons
```

#Saída: 235286

Análise de forma Colunar - Seleções

Para seleções:

```
branches['nMuon']== 1 # o resultado é array booleano  
#Saída:<Array [False, False, True, ... False, False] type='100000 * bool'>
```

Para contar quantos eventos atendem a essa condição:

```
single_muon_mask = branches['nMuon']== 1  
np.sum(single_muon_mask) #Saída: 13447
```

Para selecionar o p_T dos mísions em eventos com exatamente um mísion, usamos a máscara single_muon_mask como um índice:

```
branches['Muon_pt'][single_muon_mask]  
#Saída:<Array [[3.28], [3.84], ... [13.3], [9.48]] type='13447 * var * float32'>  
len(branches['Muon_pt'][single_muon_mask])  
#Saída: 13447
```

Análise de forma Colunar - Seleções

Construindo os 4-momenta dos mûons:

```
two_muons_mask = branches['nMuon']== 2
```



Com o pacote [Vector](#), podemos construir os quadrimomentos dos mûons a partir das variáveis p_T , η , ϕ , e massa:

```
import vector
```

```
muon_p4 = vector.zip({'pt': branches['Muon_pt'],
                      'eta': branches['Muon_eta'],
                      'phi': branches['Muon_phi'],
                      'mass': branches['Muon_mass']})
```

```
two_muons_p4 = muon_p4[two_muons_mask]
```

Agora podemos acessar propriedades como p_T , η , ϕ , E , e mass dos mûons filtrados.

Análise de forma Colunar - Seleções

Para calcular a massa invariante dos dois mûons em cada evento, somamos os quadrvetores. Primeiro, selecionamos o quadrimomento do primeiro mûon em cada evento usando o slicing [:, 0]:

```
first_muon_p4 = two_muons_p4[:, 0] #primeiro mûon de cada evento  
second_muon_p4 = two_muons_p4[:, 1] #segundo mûon de cada evento
```

Agora podemos somar os quadrvetores e calcular a massa invariante dos dois mûons para cada evento.

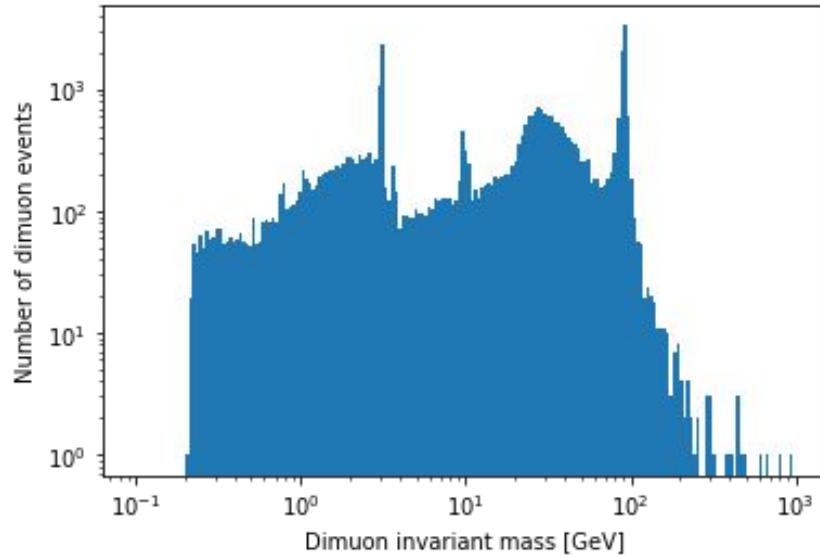
```
sum_p4 = first_muon_p4 + second_muon_p4 #o resultado é um array 1D  
two_muons_charges = branches['Muon_charge'][two_muons_mask]  
opposite_sign_muons_mask = two_muons_charges[:, 0] != two_muons_charges[:, 1]
```

Por fim, aplicamos essa máscara ao somatório dos quadrvetores para obter os eventos de mûons de sinais opostos:

```
dimuon_p4 = sum_p4[opposite_sign_muons_mask]
```

Massa invariante dos pares de mísseis de cargas opositas

```
plt.hist(dimuon_p4.mass, bins=np.logspace(np.log10(0.1), np.log10(1000), 200))  
plt.xlabel('Dimuon invariant mass [GeV]')  
plt.ylabel('Number of dimuon events')  
plt.xscale('log')  
plt.yscale('log')  
plt.show()
```



Análise de forma Colunar - Ferramentas

Existem várias bibliotecas Python poderosas para a análise de dados de física de altas energias (HEP). Aqui está um resumo das que vimos:

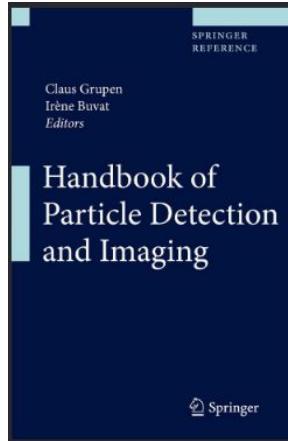
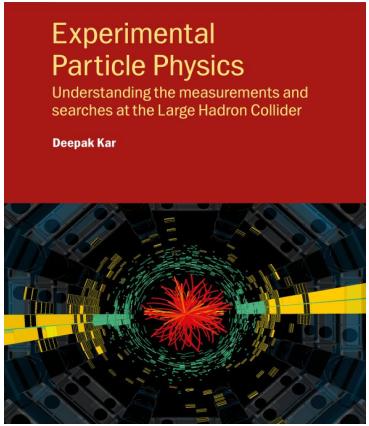
- **Uproot**: leitura e escrita de arquivos ROOT.
- **Awkward Array**: manipulação de arrays irregulares.
- **Vector**: operações com quadrvetores.
- **mplhep**: criação e estilização de histogramas.

Conclusão

Realizamos uma breve revisão dos principais passos e processos básicos envolvidos na análise de dados de um experimento no LHC.

Abordamos as etapas fundamentais para o desenvolvimento de uma análise, incluindo a estrutura dos dados, os diferentes formatos utilizados e as ferramentas essenciais empregadas ao longo do processo.

Referências





Fundação Carlos Chagas Filho de Amparo
à Pesquisa do Estado do Rio de Janeiro



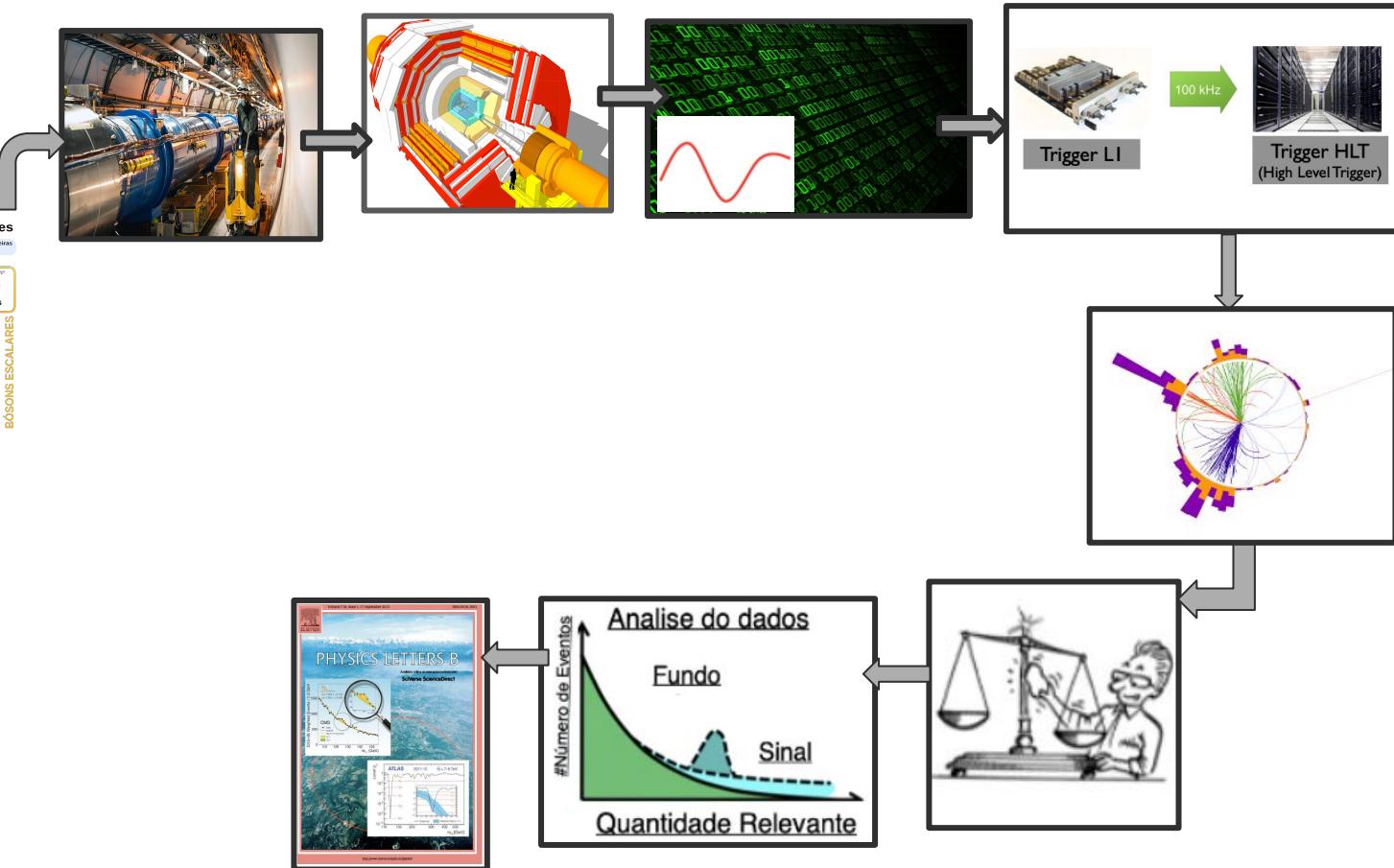
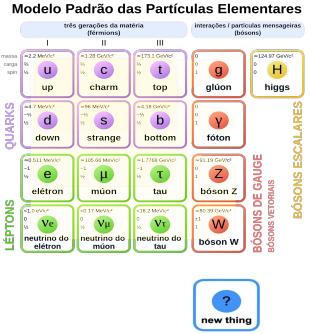
Backup

Multiple files and large files

```
uproot.concatenate({'*.root': 'Events'})
```

Another method is to iterate through the entries across all files:

```
for events in uproot.iterate({'*.root': 'Events'}, step_size=100):  
    # do something with events
```



Teoria

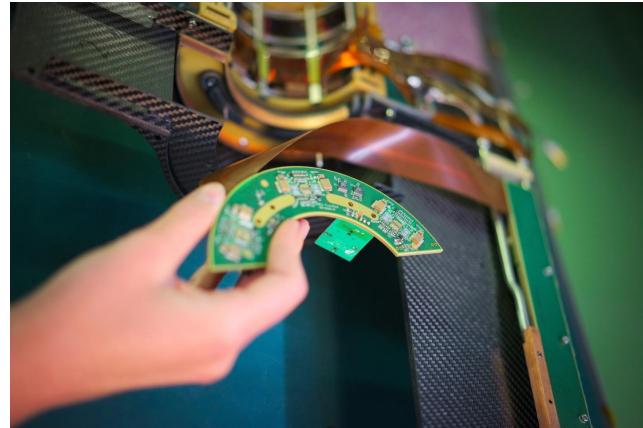
$$\begin{aligned}
 \mathcal{L}_{SM} = & \underbrace{\frac{1}{4}W_{\mu\nu}\cdot W^{\mu\nu} - \frac{1}{4}B_{\mu\nu}B^{\mu\nu} - \frac{1}{4}G_{\mu\nu}^\alpha G_\alpha^{\mu\nu}}_{\text{kinetic energies and self-interactions of the gauge bosons}} \\
 & + \underbrace{\bar{L}\gamma^\mu \left(i\partial_\mu - \frac{1}{2}g\tau\cdot W_\mu - \frac{1}{2}g'YB_\mu \right) L + \bar{R}\gamma^\mu \left(i\partial_\mu - \frac{1}{2}g'YB_\mu \right) R}_{\text{kinetic energies and electroweak interactions of fermions}} \\
 & + \underbrace{\frac{1}{2} \left| \left(i\partial_\mu - \frac{1}{2}g\tau\cdot W_\mu - \frac{1}{2}g'YB_\mu \right) \phi \right|^2 - V(\phi)}_{W^\pm, Z, \gamma \text{ and Higgs masses and couplings}} \\
 & + \underbrace{g'' \left(\bar{q}\gamma^\mu T_a q \right) G_\mu^\alpha}_{\text{interactions between quarks and gluons}} + \underbrace{\left(G_1 \bar{L}\phi R + G_2 \bar{L}\phi_c R + h.c. \right)}_{\text{fermion masses and couplings to Higgs}}
 \end{aligned}$$

LHC - Medir a Luminosidade

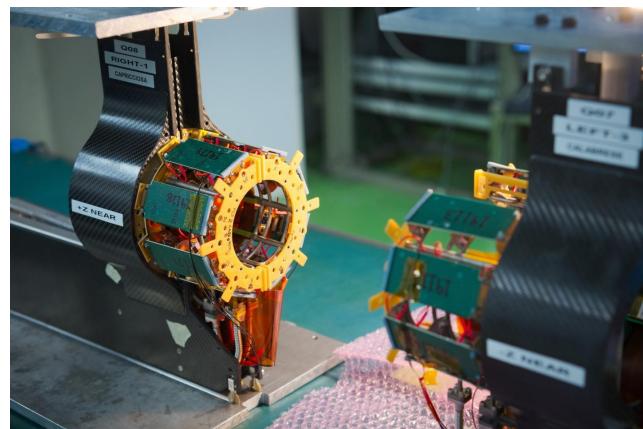
Dois sistemas de detecção são colocados a uma distância de aproximadamente **1,8 metros** do ponto de interação, com um raio de **6 cm**.

- **Fast Beam Condition Monitor (BCM1F):**
É composto por placas de circuito impresso em forma de "C" (PCBs), com dois anéis em cada lado do CMS. Possui uma alta resolução temporal, com **6,25 ns**, o que permite detectar as colisões de maneira muito rápida e em detalhes.
- **Pixel Luminosity Telescope (PLT):**
Consiste em **16 módulos** no total, com **8 em cada lado do CMS**. Esse sistema usa uma técnica de contagem rápida de clusters, operando a **40 MHz**.

Outros subdetectores usados: Pixel Cluster Counting (PCC) e Hadronic Forward (HF).



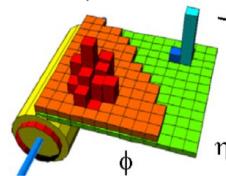
BCM1F



PLT

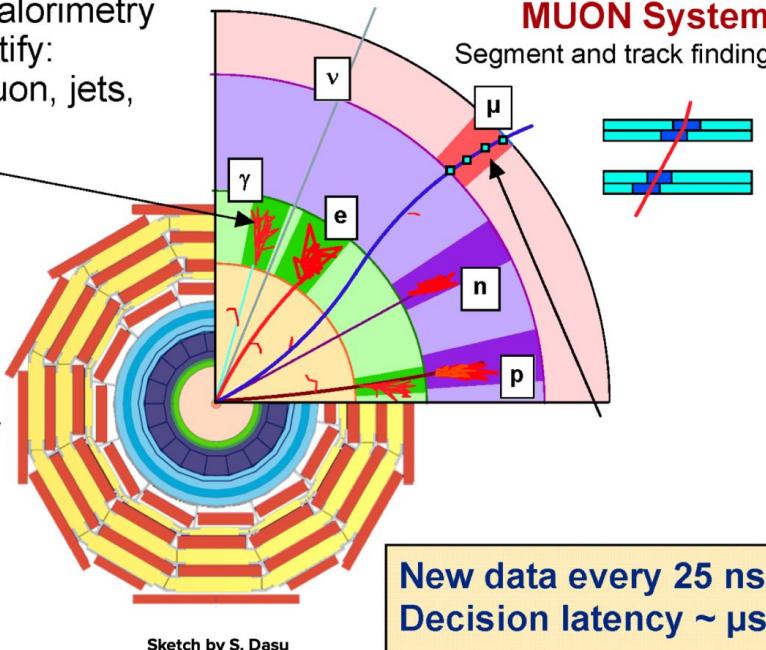
L1TRIGGER IN RUN1/RUN2/RUN3

Use prompt data (calorimetry and muons) to identify:
High p_t electron, muon, jets,
missing E_T

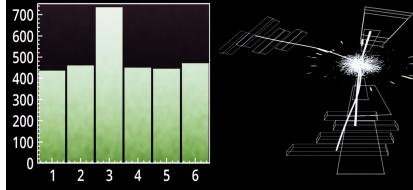


CALORIMETERS

Cluster finding and energy deposition evaluation



Medições e buscas



A distribuição de quantidades medidas nos dados:

- É prevista por um modelo teórico;
- Depende de alguns parâmetros teóricos, como por exemplo: massa de partículas, seção de choque, etc.

Dado o nosso conjunto de dados, queremos:

- Medir parâmetros teóricos, por exemplo: $m_H = 125,38 \text{ GeV}$
- Responder a perguntas sobre a natureza dos dados:
 - Existe um bóson de Higgs? Sim! (evidência forte? Quantifique!)
 - Existe Matéria Escura? Até agora, nenhuma evidência...
 - Se não houver evidência, qual é o intervalo de parâmetros teóricos compatível com os dados observados? Qual intervalo de parâmetros podemos excluir?

Devemos utilizar a teoria de probabilidade sobre nossos dados e o modelo teórico para extrair informações que respondam às nossas perguntas, ou seja, usamos a estatística para análise de dados.

ROOT - funções -TF1- ajuste

UW PICO 5.09

```
Void ex_gaussfit() {
    // Criar um histograma com 100 bins, entre -5 e 5
    TH1F *hist = new TH1F("hist", "Exemplo de Histograma com Ajuste Gaussiano", 100, -5, 5);

    // Preencher o histograma com dados aleatórios seguindo uma distribuição Gaussiana
    hist->FillRandom("gaus", 10000); // 10000 entradas aleatórias

    // Criar uma função Gaussiana para o ajuste
    TF1 *gausFit = new TF1("gausFit", "gaus", -5, 5);

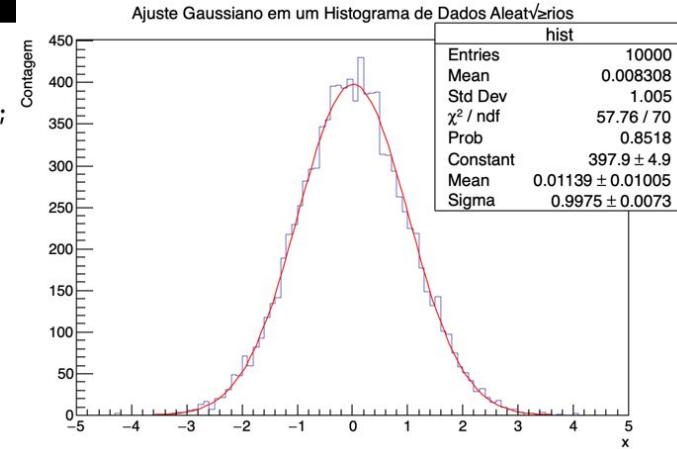
    // Realizar o ajuste
    hist->Fit("gausFit");

    // Definir títulos para o histograma
    hist->SetTitle("Ajuste Gaussiano em um Histograma de Dados Aleatórios");
    hist->GetXaxis()->SetTitle("x");
    hist->GetYaxis()->SetTitle("Contagem");

    // Desenhar o histograma e o ajuste
    hist->Draw();

    // Mostrar os parâmetros do ajuste
    double mean = gausFit->GetParameter(1); // Média (mu)
    double sigma = gausFit->GetParameter(2); // Desvio padrão (sigma)

    std::cout << "Parâmetros do Ajuste Gaussiano:" << std::endl;
    std::cout << "Média (mu) = " << mean << std::endl;
    std::cout << "Desvio padrão (sigma) = " << sigma << std::endl;
}
```



Para exibir os parâmetros do ajuste:
`gStyle->SetOptFit(1111);`

ROOT - funções -TF1- ajuste

```
[eliza@MacBook-Air-de-Eliza Escola2024 % root -l ex_gaussfit.C
root [0]
Processing ex_gaussfit.C...
Info in <TCanvas::MakeDefCanvas>: created default TCanvas with name c1
*****
Minimizer is Minuit2 / Migrad
Chi2 = 57.7629
Ndf = 70
Edm = 1.24167e-06
NCalls = 54
Constant = 397.889 +/- 4.93044
Mean = 0.0113938 +/- 0.0100493
Sigma = 0.997452 +/- 0.00730912 (limited)
Parâmetros do Ajuste Gaussiano:
Média (mu) = 0.0113938
Desvio padrão (sigma) = 0.997452
root [1] Info in <TCanvas::Print>: file ./c1.png has been created
```

O que é o múon?

- Um múon é um lípton. É o primo mais pesado do elétron.
 - Massa do múon $\sim 105,7$ MeV $\Rightarrow m_\mu/m_e \sim 200$.
 - O múon possui carga elétrica (+,-).
 - Participa das interações eletromagnéticas e fracas, mas não das interações fortes.
- No que os múons decaem?
 - $BR(\mu \rightarrow e\nu\bar{\nu}) \sim 1$. Como isso funciona?
 - E quanto ao decaimento $\mu \rightarrow e\gamma$?
- Se os múons decaem, como conseguimos observá-los no CMS?
 - A vida média do múon é $\tau \sim 2,2 \times 10^{-6}$ s $\Rightarrow c\tau \sim 660$ m!
 - Qual é a distância típica entre o ponto de colisão e o início das câmaras de mÚons?
- Como a vida média do múon se compara à de outras partículas?
 - Por que a vida média do múon é tão longa?
 - Qual é o impacto disso?

Por que nos importamos com os mûons?

- Mûons são produzidos em interações fracas
 - Se um mûon está presente no estado final, um W, Z ou H deve ter estado envolvido.
 - O estudo da física eletrofraca é uma das principais missões do LHC e do CMS.
 - Ou uma nova física está próxima. A busca por física além do Modelo Padrão é a outra grande missão.
- Mûons podem ser um indicador útil de física potencialmente interessante
 - Se conseguirmos diferenciar entre mûons de W, Z ou H (mûons pontuais) e aqueles de outras fontes.
 - Durante este exercício, discutiremos maneiras de fazer isso e como os mûons podem ser usados em uma medida ou busca.
- O detector CMS foi projetado para detectar e identificar mûons com alta eficiência e medir seu momento e carga em uma ampla faixa de parâmetros cinematáticos.

identificação de múons

Identificação de Múons para Diferenciar Fontes: A identificação de múons é essencial para distinguir entre diferentes origens de múons. Em um experimento como o CMS, múons podem ser produzidos por vários processos, como decaimentos de quarks pesados, decaimentos de partículas instáveis ou mesmo como parte do ruído de fundo. A escolha de critérios de identificação específicos ajuda a diferenciar entre esses casos.

IDs Baseados em Partícula Fluida (Particle Flow, PF):

- **Loose ID:** Focado em uma alta eficiência para todos os tipos de múons, sem restrições rigorosas. É útil para quando se quer captar o máximo de múons possíveis, inclusive aqueles de fontes secundárias.
- **Medium ID (CutBased e MVA-based):** Alta eficiência para múons prompt (múons originados diretamente da colisão, não de decaimentos secundários) e decaimentos de quarks pesados (como o quark charm e o bottom).
- **Tight ID (CutBased e MVA-based):** Alta eficiência especificamente para múons prompt, sendo mais seletivo e usado quando há necessidade de maior certeza de que o muon seja "prompt".

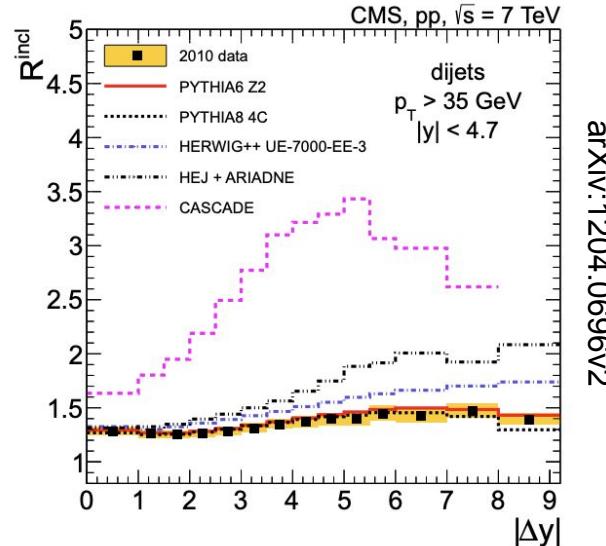
Outros Tipos de IDs Específicos:

- **High pT ID:** Projetado para selecionar múons de alto momento transversal ($pT > 200 \text{ GeV}$) com alta eficiência e qualidade de reconstrução. Esses múons não são mais minimamente ionizantes (MIPs) e perdem energia ao passar pelo sistema de detecção de múons, o que torna a identificação um pouco mais complexa.
- **SoftID/SoftMVA ID:** Específico para a física de b-quarks (b-física), onde é comum lidar com múons de baixa energia resultantes de decaimentos de quarks pesados.
- **MVA ID:** Baseado em uma técnica de Machine Learning (MVA - Multi-Variate Analysis) para fornecer alta eficiência em identificar múons prompt e decaimentos de quarks pesados.
- **Prompt MVA:** Um identificador que combina critérios de isolamento, parâmetros de impacto e outras variáveis para selecionar múons prompt com grande pureza, essencial quando é necessário reduzir a presença de ruído de fundo.

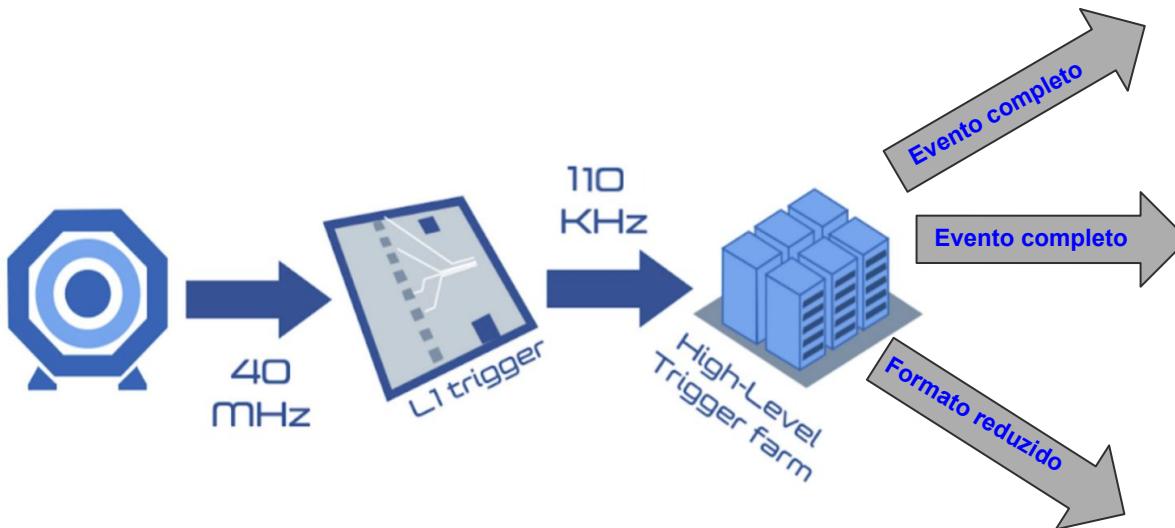
Etapas da Análise

Seleção Inclusiva e Exclusiva

- **Seleção Inclusiva:** Inclui uma coleção de processos que conterão o processo de interesse, mas também incluirão outros processos. Por exemplo, uma seleção dijet inclusiva manterá todos os eventos com pelo menos dois jatos.
- **Seleção Exclusiva:** Foca-se apenas no processo específico de interesse. Uma seleção dijet exclusiva manterá apenas eventos com exatamente dois jatos.



Data Streams



Parking Stream: 3 kHz e 1,5MB/evt, armazena eventos completos em fita para uso posterior. Esses dados não são processados em tempo real, mas sim "estacionados" até que haja capacidade suficiente para processá-los.

Standard Stream: 1kHz e 1,5 MB/evt, são processados em tempo real no Tier-0.

Scouting Stream: formato de dados reduzidos **em cerca de 100 vezes** em comparação com o **Standard Stream**. É **~ 30 vezes maior** do que a da stream padrão. Feito inteiramente com base na reconstrução realizada no **HLT**.

Thiel M.

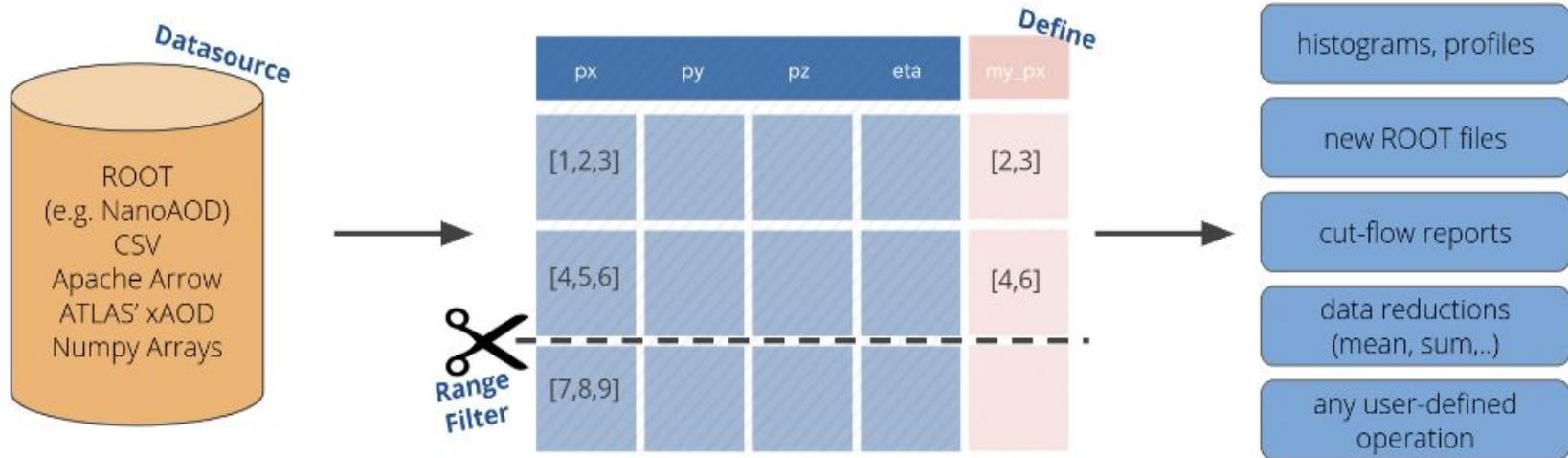
Classes do ROOT

Uma classe é uma estrutura que encapsula objetos e funções que operam sobre esses objetos

- Por convenção no ROOT, classes começam com T
- Com essas classes podemos criar janelas, histogramas, gráficos, legenda para os gráficos, manipular os elementos dos gráficos/histogramas, fazer fit, gerar números aleatórios e uma infinidade ações
- Lista de classes do ROOT:
 - <https://root.cern/doc/v612/annotated.html>

C TGViewPort
C TGVProgressBar
C TGVScrollBar
C TGVSlider
C TGVSplitter
C TGWidget
C TGWin32
C TGWin32GLManager
C TGWin32GLManagerProxy
C TGWin32InterpreterProxy
C TGWin32ProxyBase
C TGWin32VirtualXProxy
C TGWindow
C TGX11
C TGX11TTF
C TGXYLayout
C TGXYLayoutHints
C TH1
C TH1C
C TH1D
C TH1Editor
C TH1F
C TH1I
C TH1K
C TH1Merger
C TH1S
C TH2
C TH2C
C TH2D
C TH2Editor
C TH2F
C TH2GL
C TH2I
C TH2Poly

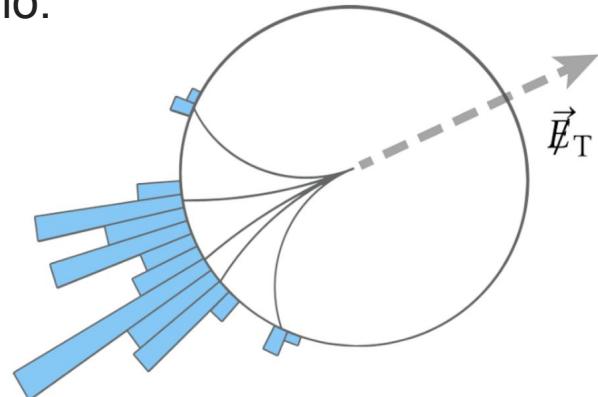
ROOT - TTree - RDataFrame



CMS - Energia transversa ausente (MET)

- um conceito em física de partículas usado para inferir a presença de partículas que interagem fracamente (como neutrinos, candidatos a matéria escura e algumas partículas de vida longa) e que não deixam sinal direto no detector.
- A existência dessas partículas é indicada por um desequilíbrio no momento no plano perpendicular ao feixe.
- O MET é definido como a soma negativa dos momentos transversais (MET, p_T) de todos os candidatos de partículas reconstruídas em um evento, representado matematicamente como:

$$\vec{p}_T^{\text{miss}} = - \sum_i \vec{p}_{T,i}$$





ROOT - TFile

Escrever em um arquivo:

```
eliza@MacBook-Air-de-Eliza EscolaINCT2024 % root -l
root [0] TFile f("file.root", "RECREATE");
root [1] TH1F h("h", "h", 64, 0.0, 8.0);
root [2] h.Write("h");
root [3] f.Close();
```

Feche o arquivo e certifique-se de que a operação foi bem-sucedida

```
eliza@MacBook-Air-de-Eliza EscolaINCT2024 % rootls -l file.root
TH1F Oct 31 12:20 2024 h;1 "h"
```

Leitura

```
TFile f("file.root");
TH1F* h2 = f.Get<TH1F>("h");
h2->Draw();
```

Estrutura dos nomes para MC e Dados reais

/ThePhysicProcess/TheHow(it was processed)/TheWhat(it contains)

MC

/ThePrimaryDataset/TheHow(it was processed)/TheWhat(it contains)

Dados

Diferente do MC, onde o nome do dataset é baseado no processo simulado, nos dados reais o nome do dataset primário reflete o trigger que capturou os eventos.

Certificação de Dados:

- **Certificação de dados ruins (bad runs):** quando estamos processando dados reais, é comum precisarmos filtrar os dados para excluir runs ou lumisections que foram marcadas como ruins durante a certificação de dados. Isso garante que você esteja trabalhando apenas com dados de boa qualidade.
- Um arquivo chamado "**Golden JSON**" é fornecido para essa finalidade. Este arquivo contém uma lista de runs e lumisections certificadas como boas, e pode ser usado para filtrar automaticamente os dados ruins.

Análise colunar

A análise colunar é uma abordagem moderna para manipulação e análise de grandes volumes de dados, especialmente em áreas como a Física de Altas Energias (HEP), onde o volume de dados é imenso. Diferente do método tradicional baseado em loops de eventos, no qual cada evento é processado individualmente, a análise colunar utiliza operações com arrays, tratando cada campo dos eventos como uma coluna. Isso permite aplicar operações a colunas inteiras de dados em paralelo, de forma similar ao que acontece em bibliotecas como o NumPy.

Principais características da Análise Colunar:

1. **Operações Vetorizadas:** Ao invés de iterar sobre os eventos um por um, as operações são realizadas em arrays inteiros ou colunas de dados de uma vez só. Isso torna a execução muito mais rápida, pois operações em arrays podem ser otimizadas e paralelizadas.
2. **Sem Loops Explícitos:** Não é necessário utilizar loops como `for` para processar os dados evento a evento. Em vez disso, é possível filtrar e manipular os dados com operações concisas em arrays, o que simplifica o código e reduz a chance de erros.

Eficiência: Processar dados diretamente como arrays reduz a sobrecarga de processamento, aumenta a eficiência de cache e aproveita melhor as arquiteturas modernas de CPU e GPU, que são otimizadas para operações vetorializadas.

Bibliotecas: Em HEP, ferramentas como `uproot` e `awkward-array` são usadas para ler e processar os dados nesse formato. Essas bibliotecas permitem trabalhar com arquivos ROOT e árvores (trees) como se fossem arrays nativos no Python, possibilitando operações colunares diretamente.

Aplicações: A análise colunar é amplamente utilizada para seleção de dados (como aplicar cortes), extração de características e outras manipulações de dados que precisam ser realizadas de forma eficiente em grandes volumes de dados.

Descoberta

- Em física de partículas, uma descoberta é geralmente reivindicada quando um resultado atinge um nível de significância estatística de cinco desvios-padrão (5σ).
- Esse limiar corresponde a uma probabilidade de cerca de 1 em 3,5 milhões de que o resultado seja devido a uma flutuação aleatória, garantindo assim um nível de confiança muito alto.
- O critério de 5σ é utilizado para reduzir a probabilidade de falsos positivos, dado o caráter complexo e altamente sensível dos experimentos em física de partículas.
- Esse padrão foi fundamental, por exemplo, para confirmar descobertas como a do bóson de Higgs no Grande Colisor de Hádrons (LHC) em 2012.

Medições

As medições tratam de quantificar aspectos específicos do Modelo Padrão, enquanto as buscas envolvem a investigação de fenômenos que poderiam indicar novas físicas além do Modelo Padrão.

As **medições** são um componente fundamental da física experimental e são projetadas para verificar previsões do Modelo Padrão (SM). A ideia central é realizar essas medições de forma que elas sejam úteis para comparar com modelos teóricos, sem depender de suposições excessivamente modeladas a priori.

Objetivo das buscas

As buscas visam encontrar **desvios das previsões do Modelo Padrão**, que possam sugerir a presença de nova física. Estas buscas geralmente focam em modelos específicos de física além do SM (como supersimetria, teorias de grande unificação, etc.), os quais preveem assinaturas específicas no estado final das colisões, como novas partículas ou ressonâncias.

Sinal e Fundo

- **Sinal:** O que se está tentando detectar, geralmente uma nova partícula ou interação prevista por um modelo além do SM.
- **Fundo:** Qualquer outro processo do SM que produza o mesmo estado final, mas sem nova física. O fundo é estimado com base em simulações de Monte Carlo (MC) ou medições do SM e representa o ruído no qual se tenta encontrar o sinal.

A ideia central nas análises de buscas é **realçar o sinal sobre o fundo**, otimizando as seleções de eventos e estados finais.

Medições - Região Fiducial

Um conceito-chave aqui é o de **região fiducial**, que refere-se a uma área do detector onde os objetos físicos, como partículas, podem ser reconstruídos de maneira confiável. Geralmente, essa região é definida por faixas de **momento transverso (pT)** e **pseudorapidez (η)**, que delimitam a capacidade de detecção e reconstrução de objetos em experimentos como o CMS ou ATLAS.

A importância de realizar medições dentro dessa região é para evitar extrapolações para áreas fora do alcance direto do detector. Extrapolações dependem fortemente de simulações de Monte Carlo (MC), que por sua vez são baseadas no Modelo Padrão. Fazer medições fora da região fiducial adiciona um viés modelado que pode mascarar ou distorcer os resultados.

Veremos os passos necessários para alcançar essa abstração e redução de dados no contexto de experimentos de Física de Partículas.

Serão dados exemplos de medições realizadas em colisores de hadrons. Os conceitos básicos por trás dos algoritmos de reconstrução, como a busca de trajetórias em detectores de rastreamento e medições de energia em calorímetros, serão discutidos, juntamente com algoritmos de nível superior, como a reconstrução de jatos de partículas.

Por fim, será descrito o ambiente típico de software e computação dos grandes experimentos de colisores, essencial para atingir os objetivos da análise de dados.

Medições e buscas

- **estimativa de fundo e desdobramento (*unfolding*)**: a estimativa de fundo é crucial para distinguir sinais de possíveis novas físicas a partir de fenômenos conhecidos. O *unfolding* é usado para corrigir os efeitos do detector e permitir que as medidas experimentais se aproximem das variáveis físicas verdadeiras.
- **Métodos estatísticos**: conceitos estatísticos são essenciais para quantificar a significância dos resultados, calcular incertezas e realizar testes de hipóteses. A estatística é usada para comparar dados reais com previsões do MC, ajudando a determinar se os dados estão de acordo com o Modelo Padrão ou se apontam para novas físicas.
- **Ferramentas computacionais**: um grande exemplo é o ROOT, ele é o software mais utilizado para realizar análises em física de altas energias, fornecendo uma variedade de ferramentas para análise estatística, ajuste de funções, visualização de dados, e mais.

Medições - Correções de Resultados

Nível de Detector vs. Nível de Gerador

As medições podem ser realizadas em diferentes níveis:

- **Nível de Detector (*Detector level*)**: refere-se aos dados coletados diretamente do detector, após reconstrução de objetos físicos (como múons, jatos, etc.). Esses resultados são específicos do experimento e dependem da performance do detector.
- **Nível de Gerador (*Generator Level*)**: para que as medições sejam úteis para teoristas e desenvolvedores de MC, os resultados precisam ser corrigidos para remover os efeitos específicos do detector. Isso é feito por um processo chamado *unfolding* (desdobramento), que reconstrói os resultados no nível das partículas geradas, o que torna as medições independentes do detector.

Essa distinção é importante porque, para medições úteis, é necessário que os resultados possam ser comparados com previsões teóricas e simulações MC em uma base independente do detector. No entanto, para buscas de novas físicas (BSM), o nível de detector pode ser suficiente, pois o objetivo é procurar por **desvios das previsões do SM**. Se houver um desvio, isso pode ser indicativo de nova física, mesmo que os efeitos específicos do detector estejam presentes.

Medições - Correções de Resultados

aspectos importantes sobre como objetos físicos (como léptons, jatos e energia transversa ausente) são reconstruídos a partir dos dados detectados e as complexidades envolvidas ao comparar esses dados com simulações e previsões teóricas. Aqui estão os principais conceitos discutidos:

1. Níveis de Detecção e Simulação

- **Nível de Detetor (Detector Level):** Os dados que vêm diretamente do detetor são chamados de "nível de detetor". Isso inclui partículas reconstruídas e outros objetos físicos que são observados após o processamento das colisões reais.
- **Nível de Partícula (Particle Level ou Truth Level):** Nas simulações de Monte Carlo (MC), os eventos começam no nível de partícula, antes que os efeitos do detetor sejam simulados. O nível de partícula reflete o que acontece fisicamente na colisão, mas não o que o detetor realmente observa. Os experimentos do LHC às vezes chamam esse nível de "truth level" (nível verdadeiro), porque reflete o que é gerado diretamente na colisão sem a influência do detetor.
- **Nível de Parton (Parton Level):** Para certas medições, como aquelas envolvendo quarks top, os resultados podem ser apresentados no nível de parton, que refere-se ao estado dos quarks e glúons antes de processos como hadronização. No entanto, o nível de parton não corresponde a algo que seja diretamente detectável, e essas medições frequentemente precisam ser corrigidas para o nível de partícula usando fatores de correção não-perturbativos derivados das simulações MC.