~~Creating HEP analyses with Agents~~
The AGC As AI Skills

Matthieu Foresi (IRIS-HEP Fellow @ UW/Seattle, Stony Brook Grad Student!)

Gordon Watts (UW/Seattle)

PyHEP.Dev 2026

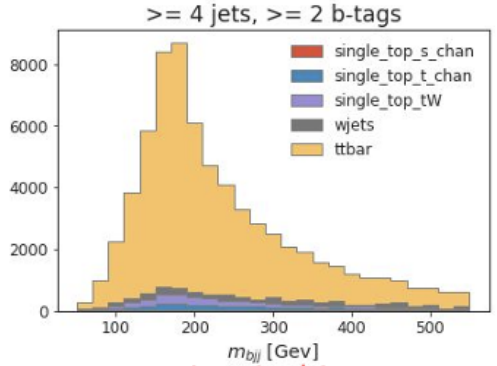
About the project

- Analysis Grand Challenge: researching ways to implement a standard analysis more efficiently with new software
- My goal: series of Claude Code skills, each teaches specific part of analysis:
  - Retrieve datasets (using ServiceX)
  - Process data and create histograms (using coffea)
  - Outputting and plotting histograms (using hist, mplhep, uproot)
  - Statistical analysis (using cabinetry)
- User specifies high-level information, such as what process to measure, what backgrounds to consider, what systematics to use, what discriminating variable to use, etc.
- Claude uses user's prompt and implements the code

From A Held & O. Shadura's 2022 ACAT talk

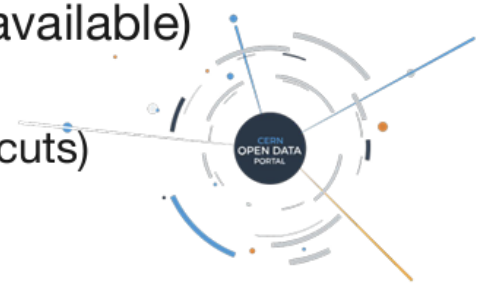
AGC: analysis task

Community benchmark



reconstructed top mass

- Analysis task: **ttbar cross-section measurement** in single lepton channel
  - Includes simple top reconstruction
  - Captures relevant workflow aspects and can easily be extended
    - E.g. conversion into a BSM search
  - Analysis task prominently features handling of systematic uncertainties
- Analysis is based on **Run-2 CMS Open Data** (~400 TB of MiniAOD available)
  - Open Data is crucial: everyone can participate
  - Currently using 4 TB of ntuple inputs (pre-converted, ~1B events before cuts)
- Goal of setup is showing **functionality**, not discovering new physics
  - Want to capture workflow; use made-up tools for calibrations & systematic uncertainties



One change: Use ATLAS open data and do $H \rightarrow ZZ \rightarrow 4\ell$

Prompt

- An extensive prompt that describes both physics goals and technology choices
- Names of steps are names of skills
- There is an `input_files` dictionary supplied for data sources
- Prompt specifies the needed variables
- Clearly lays out the steps that need to be executed
- Technology Choices
  - ServiceX data retrieval
  - coffea processing
  - histogram plotting
  - cabinetry fitting/statistics

Full Query

Implement a complete H->ZZ->4l analysis pipeline from scratch in a file called Analysis.py. Create a new code block (separated by an empty line) for each major step. The pipeline consists of four main stages, to be implemented in order:

1. Dataset retrieval - Retrieve the input columns from the input_files dictionary (Data, Background Z ttbar, Background ZZ*, Signal). Tree name: mini. Event selection: exactly four leptons (lep_n == 4). Columns needed downstream: lepton pt, eta, phi, energy, charge, type, mcWeight, and the scale factors scaleFactor_ELE, scaleFactor_MUON, scaleFactor_LepTRIGGER, scaleFactor_PILEUP. Compute a scale factor variation systematic at query stage (product of the four scale factors above, scaled up by 1.1 and down by 0.9). Support both a func_adl (FuncADL_Uproot) query and an UprootRaw alternative, selectable via a USE_SERVICEX_UPROOT_RAW flag, and respect an IGNORE_CACHE flag.
2. Event processing - Process the retrieved columns to build the discriminating variable m4l (invariant mass of the four-lepton system). Select events with exactly two same-flavor opposite-charge lepton pairs (electrons or muons). Produce histograms of m4l for one signal region, for each sample (Data, Signal MC, Background ZZ*, Background Z ttbar), including variations for the following systematics: scale factor variation, 4l mass variation, and ZZ normalization. Also account for a signal normalization factor.
3. Plotting - Plot the m4l histograms for the signal region, showing all samples and the systematic variations listed above. Create a figures directory as a sub-directory of the working sandbox directory if one does not already exist, and save each plot as both a .pdf and a .png file in that directory.
4. Statistical analysis - Using the histograms produced above, build a cabinetry configuration, collect and postprocess the templates, build a workspace, and perform the fit. The parameter of interest is the signal normalization. Produce and save (as .pdf and .png in the figures directory) a pull plot, a correlation matrix plot, and a post-fit data/model comparison plot for the signal region with a meaningful x-axis label. Finally, write a results.json file containing the POI value and error, formatted as: { “poi_val” : x, “poi_err” : y }

After each stage, run the Analysis.py file and capture any error output. If there is an error, examine the message and correct it to the best of your ability, then re-run until the file executes without errors and the output looks as expected.

Original sources can all be found in Matt’s github repo

Running...

Matt has a set of scripts:

- Runs the results repeatedly
- Builds comparisons
- Can run both CMS and ATLAS

I have copied his skills word-for-work into a .agents standard plug in: agc_marketplace.

I can now use his prompt to run a full test with the local harness I have:

```
uvx --from git+https://github.com/gordonwatts/test-wsl2-llm.git test-wsl2-llm
run --model gpt-5.6-luna:high --marketplace
https://github.com/gordonwatts/agc_marketplace --plugin agc-
skills@agc_marketplace --wsl-parent /home/gwatts/llm-testing --distro Ubuntu --
copy-file C:\Users\gordo\servicex.yaml --copy-file agc/analysis.py --prompt-
file .\agc\prompt-full.md
```

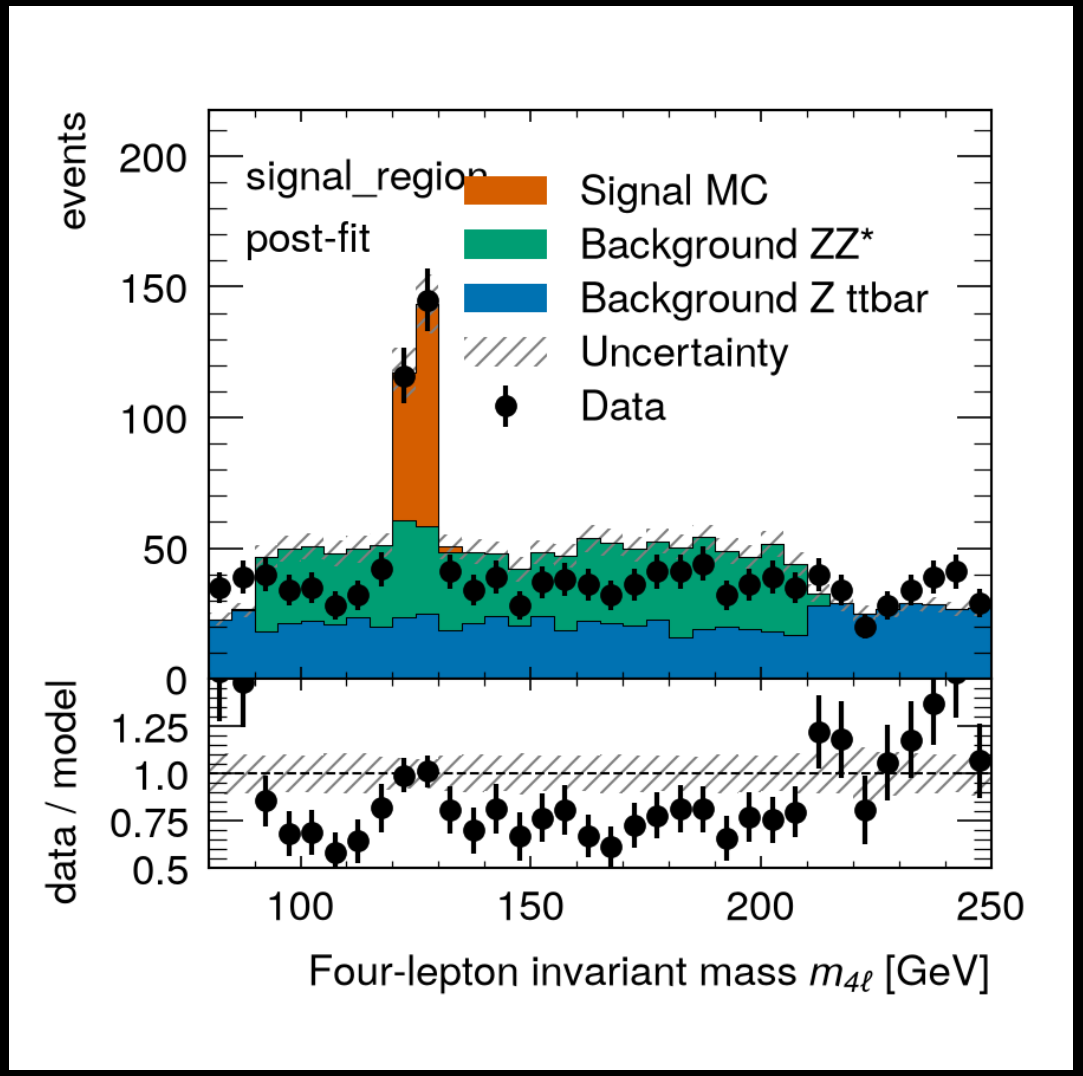
Log uploaded to agenda

Results... Ops...

Matt has made repeated runs and gotten the proper answer

Phase	Started	Finished	Duration
preflight	07:23:25	07:23:27	2s
workspace_creation	07:23:27	07:23:27	0s
input_transfer	07:23:27	07:23:29	2s
codex_home_setup	07:23:29	07:23:29	0s
plugin_installation	07:23:29	07:23:30	1s
codex_execution	07:23:30	07:34:16	10m 46s
workspace_inventory	07:34:16	07:34:16	0s
copy_back	07:34:16	07:34:19	3s
session_trace_collection	07:34:19	07:34:20	0s

11 minutes for complete implementation



Model	Attribution	Input	Cached input	Output	Reasoning output	USD total
gpt-5.6-luna	inferred - model not directly reported	2,118,857	2,021,888	31,153	10,369	$0.10

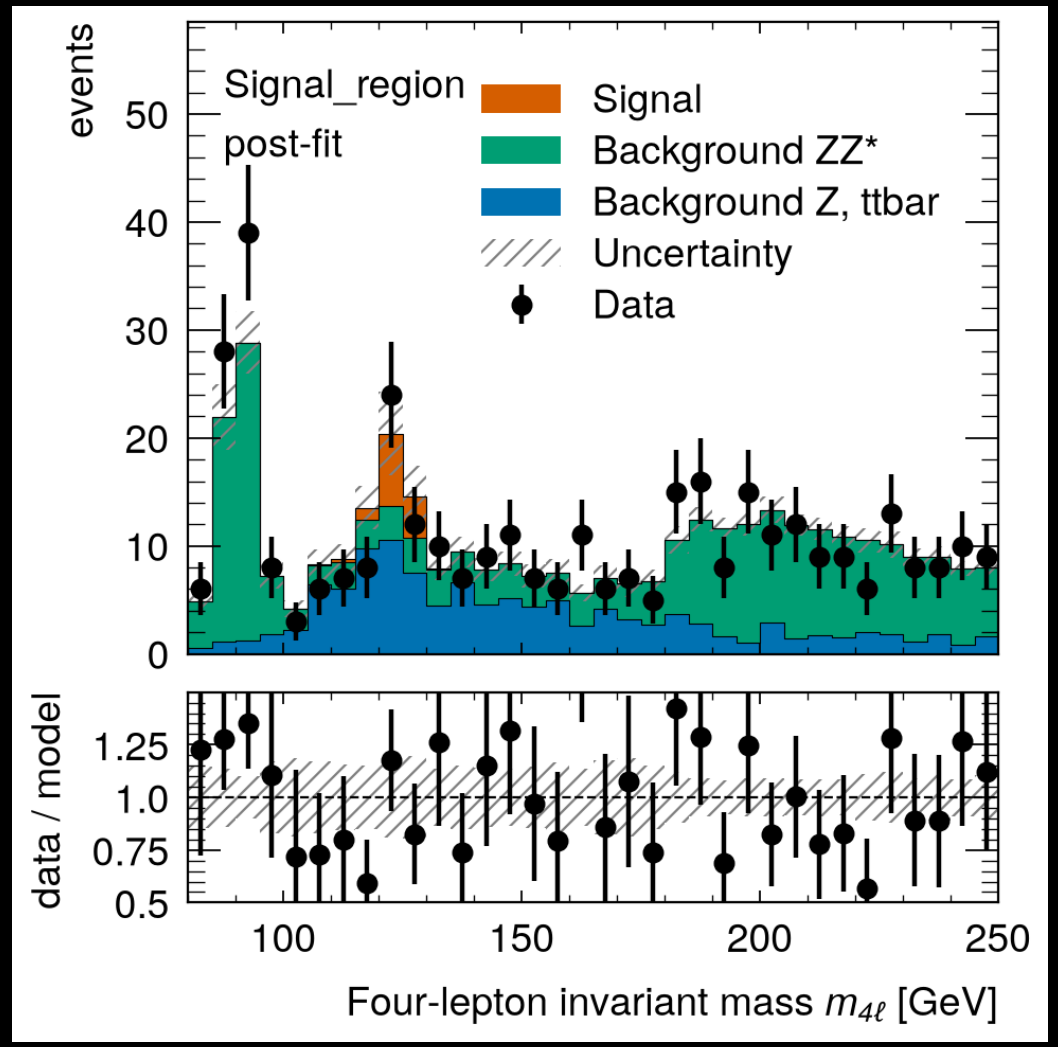
Log uploaded to agenda

Play it again Sam…

Phase	Started	Finished	Duration
preflight	07:56:54	07:56:58	4s
workspace_creation	07:56:58	07:56:58	0s
input_transfer	07:56:58	07:57:01	3s
codex_home_setup	07:57:01	07:57:01	0s
plugin_installation	07:57:01	07:57:03	1s
codex_execution	07:57:03	08:20:31	23m 29s
workspace_inventory	08:20:31	08:20:34	2s
copy_back	08:20:34	08:20:36	2s
session_trace_collection	08:20:36	08:20:36	0s

Note longer time!

Have seen it the other way…



And correct plot!!

Model	Attribution	Input	Cached input	Output	Reasoning output	USD total
gpt-5.6-sol	inferred - model not directly reported	6,183,522	6,052,608	24,645	6,579	$3.44

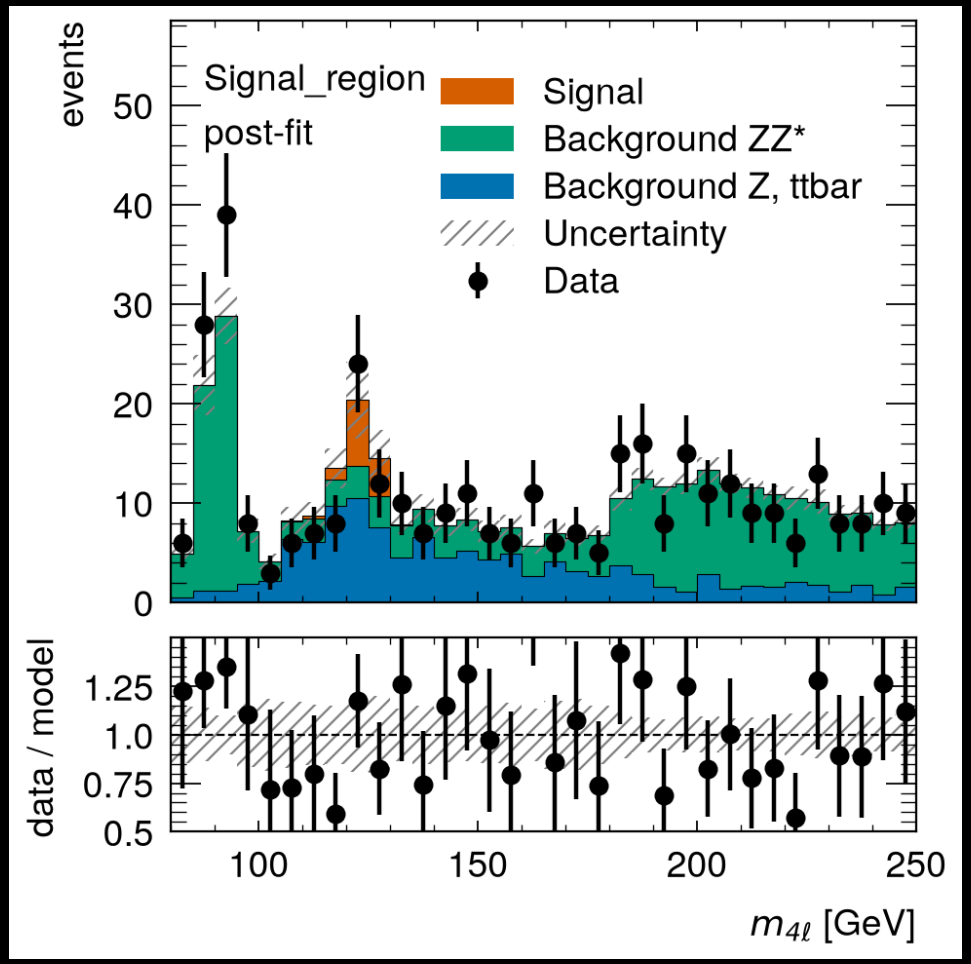
And less tokens!

Log uploaded to agenda

More, Sam...

Phase	Started	Finished	Duration
preflight	02:56:14	02:56:16	2s
workspace_creation	02:56:16	02:56:16	0s
input_transfer	02:56:16	02:56:19	3s
codex_home_setup	02:56:19	02:56:19	0s
plugin_installation	02:56:19	02:56:20	1s
codex_execution	02:56:20	03:05:17	8m 57s
workspace_inventory	03:05:17	03:05:18	0s
copy_back	03:05:18	03:05:21	3s
session_trace_collection	03:05:21	03:05:21	0s

Note shorter time!



And correct plot!!

Model	Attribution	Input	Cached input	Output	Reasoning output	USD total
gpt-6-astra	inferred - model not directly reported	1,424,424	1,358,720	13,205	1,260	$2.68

And less tokens!

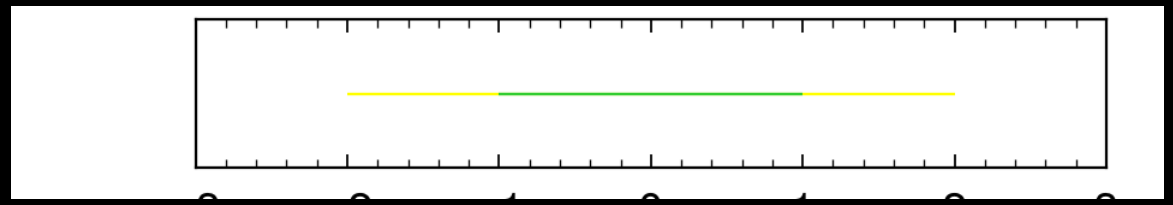
And cheaper!?!?!?!

Plots cause LLM's lots of problems

Haichen Wang's idea of finetuning a model on making plots is probably not a bad one...

```
# information for histograms
bin_edge_low = 80  # 80 GeV
bin_edge_high = 250  # 250 GeV
num_bins = 34
```

Hints for binning in the `analysis.py` file



Seems to be a mechanical issue sometimes...

Conclusions

- The analysis steps are very well specified
  - Imagine another skill on top what would generate that well specified set of steps to be really useful
- Models are capable of doing these well specified steps
- Interesting that cheapest model failed
  - Does quite well on most simpler plotting…
  - And why do models some times take x2 as long?
- What is next?
  - Create a more general skill so the prompt contains the analysis specification (H to 2 Z's).
  - Include the data specification
  - Move to running inside ATLAS
  - As always – when can we use a cheap model? And how to automate that?
  - Matt finished the CMS open data version, but I've not tried to run it yet
- Recall: AGC is old
  - It is possible that some of the AGC is now in modern model's training data
  - And these CLI's do have access to the web
  - But I have the full logs, so haven't carefully checked.