LSST DESC Analysis Planning Exercise

Anže Slosar, Analysis Coordinator, Dec 2020

The purpose of this document is to plan LSST DESC analysis from ~now to DR1. In some sense, this is a reboot of the franchize, I lay down plans how I would do it from scratch, without much regard to how it was in the past and how it relates to current and past planning documents. The purpose is to give me a clean slate and to punt the integration to a later iteration.

This text is divided into roughly two parts. In An attic of thoughts we discuss a set of pertinent points that need to be weaved into any plan. The impatient can jump straight to the Planning Proposal.

An attic of thoughts

Natural Tensions

There are inherent tensions within DESC that any planning document will have to deal with:

- Computing vs Analysis: where does one stop and the other begin even if you acknowledge that boundaries are fuzzy. Analysis people often take CI personnel as “people who make sure python runs”, while CI people often see themselves as providing service beyond execution frameworks.
- Early Science vs Pipeline Validation: Pipeline validation means that you can convince yourself that the analysis you want to do will work even when you do not have the sensitivity to perform competitive measurement. Early Science means that you punt pipeline validation and try to do something else that is scientifically interesting in some other context
- Infrastructure building vs Paper writing: everyone would like to write papers, but we need a working infrastructure. Killing two flies with the same swat is the game we are trying to play (analysis coordinators don’t injure birds).

  reprocessing / tools / resources

Anatomy of a pipeline

Pipelines are different, but they can be mapped to a similar set of operations:

- **Data Ingestion**. This is where we interface to the data management products. This process refers to the actual code that can read those products efficiently and at scale. Fundamentally this is a plumbing operation that needs to be done early enough to allow for validation, but probably after the first pass pipeline has shaken out.
- **Data curation.** This broadly refers to performing quality checks on the input data, selecting samples, throwing away bad exposures, etc. This involves some checking of the data quality.
- **Data processing.** Actual processing of the data. At the other end of this pipe come summary statistics and a set of systematics control quantities
- **Quality assurance.** This means ensuring the data pass a set of systematic and sanity checks. Null tests and the like. These need to be carefully designed. This is the step that drives the repeated processing.
- **Cosmological implications per probe**. Internal, per probe investigation of cosmological implications
- **Grand DESC probe combination.** Combination of all dark energy probes into DESC result

Probes that depend on the transient science (SN and perhaps SL) additionally need to issue follow-up request:

- **Daily analysis of previous night's events.** This involves looking at those events and deciding which events to follow-up, based on resources and current data
- **Data reduction of follow-ups.** This will likely be performed by those facilities, but we need to be part of the process and prepare the data for ingestion into our own pipelines.
- **Month-year scale quality assurance.** After a sufficient number of follow-up observations to enable a meaningful assessment has accumulated, it is necessary to check if data makes sense.

Finally, some probes will require external data, most notably PZ with training sets and clustering samples, also clusters with X-ray observations:

- **Search, curation and QA on external datasets needed before Y1 data.** This step also includes turning the data into the format that is appropriate for our pipelines.
- **Non-time sensitive follow up based on LSST data**, e.g. high-res SL observations.

Pipeline Validation

There is this formal distinction between validation and verification. I don't think it is useful in practice. All I care about is that the pipeline does what it is supposed to do. If it doesn't, I genuinely don't care if it is due to a bug in the code or a bug in the method.

Science Requirements

Science requirements help us both track whether the SNR we are aiming at is even achievable and also identify weak points in the analysis. But to me science requirements should also be a way to exercise the pipeline with synthetic data and thus act both to establish the requirements, help us understand the scientific reach and also validate the code

Processing and Reprocessing Needs

In the not too distant future we need to be able to estimate what are our requirements both for re-processing as well as actual non-DM processing. This is to make sure we have sufficient allocations ready.

Pipeline diagrams

The current pipeline diagrams in the SRM are less useful than one would anticipate. I think this is because they are made with a different level of granularity and often do not correspond to the current state of affairs.

Working Group Conveners participation

We want the working group conveners to spend most time doing useful things rather than planning. The planning so far was really heavy and the participation by the collaboration was not whole-hearted. It was focused on the SW that we want to build, but the actual pieces of software often had little to do with what was planned.

Planning Proposal

The main idea is to have a much nimbler and much shorter planning document. The document should be around 10 pages in whatever format is implemented. It would assume an educated reader and its main target audience would be management and WG conveners (although it would of course be open). The main proposal is to plan in terms of

- *Research Questions* that need to be answered. These questions are *research questions* that can lead to papers and whose answers we must know before we can write our DR1 papers. For example: “What is the highest kmax we can use in 3x2 point analysis given one loop modelling?” or “What algorithm are we going to use to fit the light-curves”? They can also be things like “What are CPU processing needs for our pipeline?” Note that questions like “Can we do X?” do not fall into this category -- instead they go with tests. In principle, questions should be “a paper worth of stuff”. Each question could correspond to either a pre-DR1 paper or a support paper with analysis release.
- *Capabilities* that analysis requires. These should be simple statements “We are capable of measuring power spectrum and calculate Gaussian Covariance errors.” Note that we do not split capability testing and validation into a separate task. The idea is that you either have a capability or you don’t. And if you claim you have it, it goes without saying that basic validation tasks have been performed.
- *Tests / Tasks* that need to be done. These are things that we think will be informative and exercise our pipelines in useful ways. They could be things like “Reprocess HSC from pixel-level data” or “Do SRD v2 using firecrown / augur”.

The planning then revolves around:

- Setting a list of questions, tasks and capabilities that you want to address by each time-chunk.
- Every 6 months you asses:
    - Which questions you answered, tasks completed and capabilities enabled?
    - Re-asses future based on the current state

The idea is to move from “This is what we will be doing next” into this is what we need to be capable of or know about by XYZ but don’t specify how exactly you plan to get there. This is closer to what is happening in practice, where people mostly play by the ear.

There are items that are common to all analysis pipelines and then there are items that are per analysis pipeline. We also need a set of real-time transient pipeline needs and a set of auxiliary data collection.

Comparison to SRM

In response to Katrin’s question, here I try to spell out how this is different from what was done in SRM. Here is an example from SRM:

Completed deliverable: *Power-spectrum estimation code (*TXTwoPoint*) (DC2 SW)*

Host: LSS *Started, originally due:* 10/01/17, 06/30/18 *Completed:* 01/16/19

URL : https://github.com/DESC/NaMaster

Objective: Measure power spectra for any number of spin-0 and spin-2 projected quantities.

The main Deliverable of this effort is a well-written and documented code (a Fourier-space version of TXTwoPoint, a module in TXPipe) that can compress positions of billions of objects, their shear estimates, their photo-z estimates and their window function into a 2-point function measurement in either Fourier or configuration space, using the information gathered in the Deliverable *"Two-point preliminary studies"*. This code can marginalize over systematic templates and provides uncertainty estimates based on several different methods. It should also be possible to use this code for null tests (e.g. correlation with known systematics), which will be necessary at any validation stage.

Prerequisites: Deliverable *"Two-point preliminary studies" "Software for characterizing mask as a function of pixelization"*

The SRM was built around deliverables. Each deliverable had a description that someone had to write. Each deliverable also had a set of dependencies that sounded plausible at the time of writing, but often don't work in that manner in practice. Each deliverable also had a name that allegedly corresponds to a software package or a github repo. In the example above, we have this code, but it is called NaMaster, which incidentally also does Gaussian covariance, but for example, doesn't really depend on the "Software for characterizing mask…", because that is still in development. In the new scheme, this is shortened to a single line in a tracking document of capabilities (note that I am not yet specifying whether this is a google sheet or what):

"Measuring power spectra of projected fields" [yes] [links to github repo] [links to paper].

Note:

- No dependency tracking. People will work out dependencies as they go along as they did so far.
- Very coarse grained. It is up to WG to assess whether something has been achieved or not
- Very succinct descriptions. They should be sufficiently accurate that WG conveners and managers would understand exactly what is meant, but not necessarily caters to newcomers.
- SRM had a ~constant granularity from immediate tasks to 3 years in the future. Because this is such a compressed format it naturally allows increase in granularity as time passses. I.e. something that is a single capability now for something that we have an idea that we will eventually need, can get resolved into multiple items as time progresses
- We do not make stuff up when the future is fuzzy. Instead we leave a placeholder item to be detailed as we go along.

- In SRM we had the same capability listed many times with an increasing feature list. I.e. DC1 measurement of something, DC2 measurement of something. Now this all belongs to the same line, but gets “upmarked” with every iteration (as needed, not as planned)

The Question of Newcomers

I think we have been quite successful with onboarding the newcomers to the collaboration, even without any of them reading our planning docs. There is this idea that a newcomer will first want a “big picture’. But you never just “get” the big picture, you start working on one little knob assigned to you and then slowly amoeba in the picture. No newcomer has started by reading all the pipeline descriptions from beginning to end.

The Beatification of Richard Dubois

One of the successes of SRM was that it allowed Richard to produce various visual representations of progress with time. Those plots lead to increases in serotonin production at various reviews. In this scheme each item could be assigned some numerical units of work that could be conserved if an item got split into multiple ones.

Worked Examples

Below we have three worked examples. Really more exploratory to see how this would work. These items would be in some sort of table, together with:

- Target completion dates
- Links to projects, papers and github pages
- Some gradation of current status:
    - For questions: None, Working, Partially answered, Fully Answered and written up
    - For capabilities: None, Placeholder, Basic Functionality, Incomplete Functionality (essentially works, but doesn’t scale, isn’t parallel, or non-production level code), Complete
    - For tasks: None, Working, Done and written up

In lists below I use underline to mark questions that would be required of all pipelines with consistent target dates. I use *italic* to mark items that could be associated with a paper of support paper.

At the end, we will need such lists for the following

Pipelines:

- 3x2pt
- Shear estimation
- PZ pipeline
- Clusters

- SN
- SL
- joint probes

RT pipelines:

- SN followup,
- SL followup(?)

External Datasets:

- PZ spectroscopic training sample
- PZ clustering redshifts sample
- CL external datasets
- SL external datasets

One item that repeats in every single set is “What is the estimate of the required computing resources?” Here I suggest that CI develops a set of guidelines on how to estimate this and what common assumptions to make.

3x2 pipeline

Research Questions:

- *What model are we going to use for galaxy clustering in auto- and cross-correlations?*
- *How are we going to model intrinsic-alignments?*
- *How are we going to split input catalogs into tomographic samples?*
- *Are we going to have the same or different samples for lenses and sources?*
- *How blending errors impact the 3x2pt analysis?*
- *How are we going to model the connected part of the covariance matrix?*
- *What is the complete list of survey properties that we want to track?*
- What is the estimate of the required computing resources?

Capabilities:

- Measure power spectra of projected fields
- Measure correlation function of projected fields
- Divide input catalogs into tomographic samples
- Estimate N(z) and its uncertainty for a tomographic sample
- *Predict full covariance matrix, including Gausiand and non-Gaussian parts*
- Perform a suite of standard PSF null-tests and QA plots
- Perform a suite of shear estimation null-tests and QA plots
- Perform a suite of cross-correlation statistics with survey property maps
- Perform likelihood evaluation
- Interface to DM products
- Ability to export summary statistics in sacc format
- *Complete Pipeline framework that start with DM and outputs sacc*

Tasks:

- Store predicted theory + noise covariance into sacc to exercise inference engine
- Update SRD predictions
- *Analyze DC2 data starting from CosmoDC2 catalogs*
- *Analyze DC2 data starting from Image Simulations / DM Catalogs*
- *Re-analyze DES, HSC, Kidds data starting from public catalogs*
- *Re-analyze DES, HSC, Kidds data starting from images where possible*
- *Analyze DR1% data making reasonable simplifications when possible*

SN real-time follow-up pipeline

Research Questions:

- *What algorithm are we going to use to trigger daily follow-ups?*
- What is the estimate of the required computing resources?

Capabilities:

- Create mock data stream of nightly alerts
- Run selections of potential targets for a given run
- Autonomous daily pipeline with automatic fall-back to redundant sites.
- Interface to DM products

Tasks:

- Design and test interfaces to external observatories
- A month-long dry-run campaign on mock data including simulated facility failure

PZ Spectroscopic Training Samples Curation

Research Questions:

- How are we going to assess spectroscopic completeness in each sample?
- What is the desired output in a library of heterogeneously selected spectral templates?
- Which surveys are we going to use?
- What is the estimate of the required computing resources?

Capabilities:

- *Automatically assemble, heterogenize and decorate with metadata libraries of spectra for PZ training*
- Interface to PZ infrastructure code

Tasks:

- Prepare internal data releases on prespecified cadences (replace with actual projected dates once they become known)