Thank you so much for your valuable comments and suggestions that led us making improvements. Below, we present each comment for the Reviewer #1 in blue and our respective answer in green.

Reviewer #1:

Major Concerns:

The package is presented as a scalable solution suitable for HPCs. However, the study is limited to a small region that fits comfortably within the memory of a conventional workstation (~15 GB), and no benchmarks are provided showing how the framework actually scales on HPC infrastructure. For example, how does processing time change with area size, number of scenes, or number of cores? Does the processing break at some point? How is load distributed across nodes in an HPC environment? Without this, the HPC contribution cannot be properly evaluated. The authors should either provide a systematic performance analysis or scale back the HPC claims to match what is actually demonstrated.

We thank the reviewer for this fair observation regarding HPC. We agree that the manuscript presented scalability as a headline feature without providing the systematic benchmarking that such a claim requires (scaling with area, scene count, and distribution across cores/nodes). Following the reviewer's second suggested option, we have scaled back the HPC and scalability claims in the revised manuscript so that they match what is actually demonstrated: that the pipeline is deployable on both HPC and standard environments and is feasible at multi-gigabyte data volumes, presented as a feasibility case rather than a benchmarked scaling study.

Concretely, we have: (i) revised the title to remove "Scalable" and "on HPCs and Beyond" and reframed it around the integration and pre-processing contribution; (ii) rephrased the abstract and introduction to describe the package as running on HPC and local environments, demonstrated on a representative region, rather than as a scalable HPC solution; (iii) stated explicitly in the methods and results that the ~15 GB single-region run is a feasibility demonstration, with a systematic multi-node performance evaluation identified as future work; and (iv) clarified the reported timing so that data transfer (I/O) is not conflated with processing time (see our response to the comment on Line 178).

The paper motivates the work partly on the premise that traditional data cube pipelines fail in dynamic environments, but does not provide an in-depth comparison showing how they fail or how this package specifically addresses those failures. It also does not mention or compare against other relevant Sentinel-2 preprocessing efforts such as xcube, cubo, sentle, or sen2r. The technical comparison to FORCE is limited. A discussion of how this package relates to and differs from these existing tools is necessary to properly contextualize the contribution.

We thank the reviewer for this important comment, and we agree that our original framing was misleading. We do not intend to claim that existing data cube pipelines fail or are deficient. These are mature, widely used, and valuable tools. Our intention is the opposite: to build upon them and to provide additional capabilities that are currently not readily available in STAC-based Python cube workflows and that are required for certain demanding Sentinel-2 applications, such as the analysis of dynamic braided river systems. In the revised manuscript we have removed the "failure/deficiency" framing throughout and replaced it with a positive statement of what stac2cube adds.

In particular, we have explicitly acknowledged and cited relevant prior work, including xcube, cubo, sentle, and sen2r, as valuable contributions on which the community (and our own work) builds. We then state clearly what stac2cube offers on top of these: an integrated pipeline combining three steps that are not jointly available in existing STAC-based Python packages: probabilistic cloud masking (s2cloudless), geometric co-registration (AROSICS), and super-resolution (SEN2SR). A specific example we highlight is probabilistic cloud masking: while this is available within Google Earth Engine as a precomputed collection, to our knowledge it is not readily accessible from a STAC-based Python package, which is one gap stac2cube addresses.

We have also clarified why stac2cube is implemented as a separate pipeline rather than as a direct extension of one of these packages. The three processing steps require handling that existing packages do not expose. For example, the s2cloudless workflow requires Sentinel-2 L1C input in addition to L2A, which in turn requires precise handling of STAC metadata (e.g. the processing baseline) and specific resampling choices that are not accessible through the existing packages. Implementing these steps therefore required a dedicated pipeline built on the cube foundation provided by cubo, rather than a lightweight extension of it.

Regarding FORCE specifically, we have expanded the technical comparison. We note that FORCE is a standalone system rather than a component that can be readily integrated into a Python environment, unlike AROSICS. In addition, FORCE's co-registration requires a Landsat reference pair, which increases processing time and complicates time-series production, whereas our AROSICS-based approach operates on Sentinel-2 alone, simplifying the workflow.

The individual methods implemented are not novel — each component (s2cloudless, AROSICS, SEN2SR) is an existing tool — and the primary contribution lies in their integration into a unified pipeline. This is a legitimate and valued form of contribution, but the paper should frame itself accordingly rather than implying broader methodological novelty. Additionally, the title proposes a "customizable" framework; however, the customization offered operates within each component's parameters rather than enabling modular exchange of components. The authors should clarify and reframe the contribution to accurately reflect what is genuinely offered.

We thank the reviewer, and we fully agree on the first point. The individual methods (s2cloudless, AROSICS, SEN2SR) are existing tools, and we do not claim methodological novelty for them. We agree that our contribution lies in their integration into a unified, cube-based pipeline, and in the revised manuscript we have framed the contribution explicitly in these terms, removing any wording that implies novelty of the underlying algorithms.

On the term "customizable," we appreciate the reviewer prompting us to clarify, as our original text did not make clear what the package actually offers. We would like to clarify that stac2cube does not enforce a fixed sequential order of the three steps. The steps are independent and composable, and the recommended workflow (cube construction -> s2cloudless -> co-registration -> super-resolution) is a recommendation for robustness, not a mandatory pipeline. In practice the user can, for example:

- use the SCL mask instead of s2cloudless (skipping s2cloudless entirely) and still apply co-registration;

- export only the binary cloud-mask files without masking the original cube, and apply co-registration using those masks;
- apply super-resolution directly, without either cloud masking or co-registration.

The appropriate combination and order of steps depend on the intended Sentinel-2 time-series application. For instance, a natural time series that retains clouds (e.g. for visual animation) can undergo co-registration and super-resolution while the clouds are preserved through the exported binary mask rather than removed from the cube; whereas a machine-learning application can apply probabilistic cloud detection together with cloud-shadow masking so that non-surface values are not passed to the model. The package is designed to support these different configurations rather than enforcing a single fixed pipeline.

We recognise that our original use of "customizable" did not make this modularity clear. In the revised manuscript we have adopted clearer wording that accurately reflects the independence of the processing steps and substantiated it with the concrete configuration options above.

We would also like to note, for transparency, that since submission we have added a cloud-shadow masking capability (derived from the s2cloudless result) and released stac2cube v1.5.0. These additions are described as post-submission updates in the revised manuscript.

Minor:

Table 1: The methodology used to obtain the "processing time" results needs to be more clearly described. If this measurement was taken within a short timeframe and from a single location, it is of limited value. Response times from STAC servers — even within a single provider such as Planetary Computer — vary with time and request location. Server load also fluctuates, affecting measured processing time. This measurement therefore needs to be either far more comprehensive or removed. A meaningful performance assessment of response time would require globally and temporally distributed queries, and even then the results would need careful interpretation, as server-side throttling mechanisms may temporarily affect repeated requests from the same IP. Furthermore, the term "processing time" is potentially misleading here, as it only encompasses the STAC query and not the actual data access, which is arguably the more operationally relevant metric.

We agree that a single-location, single-time measurement of STAC query response is of limited value and potentially misleading, as it reflects server load and network conditions rather than package performance. We have removed this column in the revised manuscript.

Table 1: It is unclear how the license column is relevant to the comparison being made, and what exactly it applies to. Please clarify or remove this column.

Thank you for the feedback. We have retained the license column and clarified its purpose: it indicates which data catalogues are openly accessible, which is decision-relevant for users on different platforms (e.g. terrabyte for researchers with LRZ access in Germany, versus Planetary Computer and Element84 as openly accessible alternatives). We have also revised the table to remove the potential confusion introduced by using a query location (Würzburg) unrelated to the

Kyrgyzstan study area; instead, we note that terrabyte returns comparatively limited data outside the EU, while being highly efficient for users working within Germany, as raw data (e.g. SAFE products) are served directly from LRZ. Finally, we have added a column indicating which Sentinel-2 collections each catalogue provides. This clarifies an important practical point: Element84 is, to our knowledge, the only publicly accessible (in cloud optimized format) source of L1C data, which is required for the s2cloudless workflow, whereas terrabyte's L1C coverage is very limited outside the EU and Planetary Computer does not offer L1C at all.

Line 111: STAC APIs do not store data. The servers providing a STAC API process spatio-temporal queries and return metadata items containing pointers to where the actual data — in this case Sentinel-2 tiles — are stored. The storage location can be entirely decoupled from where the STAC server is running. Please revise accordingly.

We thank the reviewer and have corrected this. STAC APIs process spatio-temporal queries and return metadata items pointing to the asset storage location, which is decoupled from the API server; we have revised the text accordingly.

Line 116: Please review the definition of an "API." The term appears to be used incorrectly in several places throughout the manuscript.

We have reviewed all uses of the term 'API' throughout the manuscript and corrected those that are imprecise (distinguishing API, catalogue, endpoint, and service).

Section 2.2: It is unclear why the detailed climate specifics of the Naryn River catchment are relevant to the HPC processing of Sentinel-2 data. If these details are not directly referenced in the methods or results, consider condensing this section.

We agree the section can be more focused. We have retained the climate description but tightened it and made its relevance explicit: the climatic regime strongly affects the flow and sediment dynamics of the braided Naryn system, which in turn create the rapidly changing surface conditions that motivate our pipeline. We have removed detail not connected to this motivation.

Line 169–170: UTM coordinates are expressed in metres (easting/northing), not latitude and longitude. Please correct this.

Thank you for noticing this simple mistake. UTM coordinates are indeed expressed in metres. We have fixed this and checked dimension naming for consistency.

Line 178: The description of the export step lacks sufficient methodological detail. What does "exported" mean in this context — is this a direct download followed by naive resampling? If so, the reported five minutes and 16 GB figure primarily reflects network speed rather than processing performance. If these figures are part of an evaluation, a methodology section should precede them.

We thank the reviewer, and we agree that our use of 'exported' was imprecise and obscured what the step involves. The reported figure does not correspond to a simple download. It reflects the full pipeline: querying the STAC API, accessing the data, computing the output arrays (which dominate the runtime) with selected resampling method, and writing the result to disk as a NetCDF file (with Zarr also supported as of v1.5.0). In the revised manuscript we have described this sequence explicitly and precede the reported figures with the corresponding methodology, so that the computational cost is not mistaken for network-transfer time.

Line 182: The choice of NetCDF and GeoTIFF as output formats should be briefly justified. While these remain widely used in operational workflows, newer formats such as Zarr V3 are increasingly adopted as a standard for spatio-temporal data. A sentence acknowledging this and explaining the format choice would strengthen the paper.

We have justified the output-format choice: NetCDF and GeoTIFF are retained for their broad operational familiarity and interoperability, and as of v1.5.0 the package additionally supports Zarr (V3), which we adopt in line with its emergence as a standard for cloud-native spatio-temporal data.

Figure 3 caption: Please expand the caption to walk the reader through the figure. The current description is too brief to be informative without reading the surrounding text.

We have expanded the Figure 3 caption, so the workflow is understandable without the surrounding text.

Figure 3: It would be helpful to clarify whether intermediate data cubes are duplicated and deleted during the processing workflow, and if so, how this is managed. What are the implications for storage consumption, and does this approach scale to larger datasets?

We have clarified the lifecycle of intermediate cubes (the cloud cube and cloud-free cube are stored as separate files), noted the associated storage implications, and explained how chunking manages this for larger datasets.

Section 3: Please specify the data type and compression method used for the output cube.

We have specified the output data types and compression. The Sentinel-2 L2A reflectance cube is stored as float32 (changed from float64 as of v1.5.0), while the cloud-probability and cloud-mask cubes are stored as integers to reduce file size, as probability values can be represented on a 0–100

scale. In addition, as of v1.5.0 the package applies lossless compression to the output cubes (zlib for NetCDF, and default compression for Zarr).

Section 3.2.2: If the co-registration approach relies on a moving window strategy, please clarify how alignment is ensured across scenes spanning multiple years. As currently described, scenes within a given window will be aligned to the master, but it is not clear whether a consistent alignment is maintained across longer time spans. A creeping spatial shift over time seems possible and should be discussed.

We thank the reviewer for this careful observation, which led us to clarify the co-registration strategy. Our aim is relative, intra-cube alignment (internal geometric consistency so that a fixed ground feature remains at the same pixel throughout the series), rather than absolute geolocation, and we have stated this explicitly.

We have also clarified the reference strategy, which the original text described insufficiently. A user-selected master scene, chosen from a favorable acquisition (cloud-, snow- and ice-free), anchors the series, and scenes are co-registered outward from this master in both temporal directions. Rather than referencing a single distant master, each scene is aligned to its temporal neighbour. This is a deliberate design choice: adjacent-date scenes exhibit minimal surface change, which makes the feature matching underlying co-registration more reliable than matching across large temporal (and phenological) gaps.

We acknowledge the reviewer's point that neighbour-based referencing could, in principle, allow small alignment errors to propagate along with the series. We mitigate this in two ways. First, the anchoring master can be placed by the user's choice in the time series, knowing that the first master (the most important one) is solid. Second, eligibility parameters determine whether a given scene may serve as a reference for the next: low-quality scenes (e.g. residual cloud, snow or ice cover) are excluded from acting as references, preventing them from degrading the alignment of subsequent scenes. Together these greatly minimize the magnitude and propagation of any residual drift. We have described this strategy explicitly in Section 3.2.2 and note openly that, while cumulative drift cannot be entirely excluded in principle, it is minimised by design and, in our application, the per-step corrections are small because they operate between temporally adjacent scenes.

Line 281: Please clarify how Dask is used in practice. For the 15 GB example provided, Dask is not strictly necessary, as a dataset of this size could fit into memory multiple times on a standard workstation. A discussion of when and how Dask provides meaningful benefit would be useful.

We have clarified how Dask is used: it enables chunked, out-of-core processing that becomes beneficial for larger areas and longer time series. For the 15 GB demonstration case the data fit in memory, but Dask supports scaling to larger workloads. In addition, Dask works naturally alongside the Xarray data structure of stac2cube, which makes out-of-core processing available for cubes that do not fit in memory.

Figure 4 / Section 4.1: Without ground truth data, the statistical basis for the cloud masking comparison is unclear. Please either provide ground truth validation or more explicitly acknowledge and discuss this limitation.

We agree and have more explicitly acknowledged this limitation. We do not claim absolute cloud-detection accuracy: constructing a reliable ground-truth mask for diffuse/hazy clouds is itself ill-posed, as their boundaries are inherently ambiguous. Our comparison instead demonstrates relative behaviour and threshold-tunability (e.g. s2cloudless avoiding false positives over bright gravel bars where SCL over-masks). We have reframed this analysis explicitly as a consistency and sensitivity assessment rather than an accuracy validation and state the limitation openly.

Figure 6: Without reading the surrounding text, it is not clear what effect the reader is expected to observe, or why a high or low difference value is considered good or bad. Please revise the caption to make the figure interpretable on its own.

We have revised the Figure 6 caption to explain what is shown (the co-registration difference metric), and how to interpret higher vs lower values, so the figure is self-contained.

Figure 7: The font size in the plots is too small to be legible. Please increase it.

We have regenerated Figure 7 with larger, legible fonts.

Code Repository for stac2cube v1.5.0:

Code availability: The stac2cube source code is openly available on GitHub (https://github.com/BaturalpArisoy/stac2cube) and archived on Zenodo. The version described in this revision is v1.5.0 (DOI: 10.5281/zenodo.20666787), released under the Apache License 2.0.