**Response to Reviewers**

We would like to thank the editor and reviewers for their comments.

**NGSANE: A Lightweight Production Informatics Framework for High Throughput Data Analysis**

Fabian A. Buske, Hugh J. French, Martin A. Smith, Susan J. Clark, and Denis C. Bauer

The reviewers’ comments and suggestions are boxed in gray and itemized below, followed by our responses and a description of the consequent revisions to the paper. Text from the manuscript or supplemental materials is “quoted in red”. Please note that in our responses we have used the abbreviations ‘pg’ and ‘para’ for page and paragraph, respectively.

**Reviewer: 1**

The authors use much of the space to elaborate on NGSANEs applicability in HPC and Cloud Computing environments. I understand that there is a page limit for application notes. Hence, I would highly recommend to shorten the sections that deal with HPC customization and control in favor for the actual analysis that are currently possible (out-of-the-box) with NGSANE. This would surely make the tool more attractive. Secondly, it would be very helpful to get an idea (only an idea!) on how to configure and customize the pipeline. A clear advantage of GALAXY is that it allows the less experienced user to do NGS analysis.  On the other hand, experienced bioinformaticians would be able to write their own shell script based pipeline (e.g. qc, trimming, mapping, vnc calling) in a matter of minutes. Thus the usability of NGSANE is an important issue.

We agree with the reviewer that the out-of-the-box analysis functionality is desirable for inexperienced users. We therefore extended the Github Wiki (<https://github.com/BauerLab/ngsane/wiki/How-to-use-ngsane>) to include a user guide on how to use NGSANE out-of-the-box and provide a virtual machine enabling easy testing of NGSANE’s functionality (see reviewer 3). Furthermore, we added a Wiki-page (<https://github.com/BauerLab/ngsane/wiki/HPC-Customization>) detailing how to customize NGSANE and the HPC environment (see reviewer 2, Major 2). Finally, we expanded the documentation to include instructions for creating new modules (see reviewer 3, Major 1).

NGSANE is focused on providing a framework that allows advanced utilization and fine control of HPC resources. This is one of the discriminating factors over other available tools (including Galaxy), Therefore we respectfully disagree with the recommendation to shorten this section as we see it is critical that we maintain emphasis on this point of difference. In addition, while we agree with the reviewer’s assessment that a nuclear analysis script can be written in minutes, the difficulty in real-world applications arises from making these scripts scalable and generic to maintain consistency across different projects. Notably NGSANE caters for these needs by managing the submission of large batches, and therefore governing the resubmission process of unsuccessful subsets from the point of failure, and providing generic and consistent documentation for all covered projects.

To highlight these aims better we adjusted the sentence (pg1, para1)

This is critical as further decreasing sequencing costs and expanding use of replicates to assess biological variability (Auer and Doerge, 2010) will substantially increase future study sizes therefore making the automated, documented and reproducible processing of large numbers of samples across diverse projects using High Performance Computing (HPC) clusters paramount.

**Reviewer: 2**

Major 1: The authors claim that modules for individual tasks can be easily exchanged. Unfortunately there is very little information available (either in the manuscript or as part of the online documentation) describing how one would go about adding a new module to the pipeline. Based on what I have seen in the sources this requires additions to the central configuration file as well as the main script (trigger.sh). This is not what I would call “hot swapping”.

The reviewer is correct in their assessment that new modules need the program’s path/module name listed in the header.sh as well as an entry in the trigger.sh file. To address the reviewers concern that this process is not sufficiently documented we have added a “New Module Tutorial” on our Github-Wiki (<https://github.com/BauerLab/ngsane/wiki/How-to-make-a-mod>). This tutorial details the steps required in creating a new module.

In the text “hot swapping” referred to existing modules only, where different analysis modules, parameter settings, or software versions can be executed by changes to the project-specific configuration text file rather than requiring changes in the software code itself.

To make this clear distinction we changed the paragraph (pg1, para6)

*Hot swapping and adaptability.* Individual task blocks (e.g. read mapping) are packaged in bash script modules, which can be executed locally or on subsets to test module code, submission parameters and compute node environment in stages. During production, NGSANE automatically submits separate module calls for each input file or set of files to the HPC queue. This allows different existing modules, parameter settings, or software versions to be executed by changes to the project-specific configuration file rather than requiring changes the software code itself (hot swapping).

Major 2: It is unclear to me how easy it is to adapt the HPC capabilities to the configuration of a given compute cluster. Again, manuscript and documentation are silent on this. I consider this to an important aspect. For example, it wasn’t obvious to me how one is supposed to communicate the name of the queue that jobs should be submitted to.

We agree with the reviewer that NGSANE’s distinctive feature, the HPC customization, needs to be demonstrated by concrete examples. To address this concern we added a section to the Github-Wiki (<https://github.com/BauerLab/ngsane/wiki/HPC-Customization>).

To answer the reviewer’s specific question about particular queue definitions: SGE as well as PBS users need to add the following line to the project-specific configuration text file to submit to the queue “tmppriority”.

QSUBEXTRA=" -q tmppriority”

This variable then gets handed over to qsub via core/prepareJobSubmission.sh and core/ jobSubmission.sh.

Minor 1: In their brief comparison with other tools the authors dismiss Galaxy as GUI-based. While it is certainly true that Galaxy provides a GUI (which, I agree, is not suitable for the sort of use case that NGSANE is designed for) it also provides an API for batch processing. This API is what NGSANE should be compared to rather than the GUI.

We would like to thank the reviewer for pointing the comparison with Galaxy. The Galaxy-API allows a script-based access to the tools and functionality enabled in the web service instance (<http://wiki.galaxyproject.org/Learn/API>). While it greatly improves running existing workflows on multiple files or handling loops and parallel tasks, it is still limited to the functionality provided by Galaxy, which may not always utilize the most recent versions of software, reference data or workflows.

In response to this comment we updated the sentence (Pg1, para2)

However, currently available tools are either web-based services (e.g. Galaxy (Goecks et al., 2010)), where even API-based access to the web service functionality is not readily amenable to production-scale analysis practices, or heavyweight frameworks written in user-friendly languages like Python (SNAKEMAKE – Köster and Rahmann (2012), NESTLY – McCoy et al. (2013)), Scala (GATK’s QUEUE – Broadgsa (2013)) or Groovy (BPIPE – Sadedin et al. (2012)), which encapsulate the actual program call in a wrapper-script specific syntax, hindering the development of pipeline extensions.

Minor 2: The manuscript contains a note (in the paragraph on reproducibility and checkpoint recovery) stating that NGSANE is not using GNU make. This seems out of place. There are hardly any other implementation details given in the manuscript. If any such details are to be discussed I would rather hear about what is used rather than what isn’t.

The intent was to justify why a custom implementation was necessary and to endorse snakemake’s assessment about GNU make. However, we agree with the reviewer that this sentence does not add much information and thus removed it.

Minor 3: From the manuscript it is unclear what format the summary reports are in. As far as I can tell these are in HTML. This should be stated clearly in the manuscript.

We would like to thank the reviewer for pointing this out.

We have added this information (pg2, para4)

This interactive HTML report provides an access point for new lab members or collaborators.

**Reviewer: 3**

While the code is well presented, the documentation is somewhat sparse and there are no tutorials and/or example workflows and data. Also, it became apparent that package/dependency management is not part of the system and it is therefore expected for the user to individually install all the external tools and their dependencies, and also maintain them as new versions appear. It would therefore take quite some investment to get to the point of a running pipeline, and in particular to reach all the functionality outlined in the manuscript - it is far from working 'out of the box' like e.g. Galaxy. I therefore believe some self-encapsulated tutorials are essential to the success of this system.

We agree with the reviewer for the need to expand on the documentation and the documentation has been significantly extended (please see our responses to reviewer 2).

To address reviewer 3’s other concern about having to manage external tools, we now provide a virtual machine image that contains all of the software tools necessary for tasks currently implemented within NGSANE allowing workstation-based testing (<https://github.com/BauerLab/ngsane/wiki/How-to-use-the-virtual-machine>). The image is hosted by Amazon Web Services and utilizes MIT’s StarCluster (<http://star.mit.edu/cluster/about.html>). This image can be configured to spin up a multi-node cluster on Amazon’s Elastic Compute Cloud (EC2), each containing the NGSANE image. This allows users to test NGSANE on the provided toy dataset or link in their own data to process large datasets with NGSANE without having to own a cluster or install/configure the required third party tools themselves. We therefore updated the last sentence to read:

NGSANE is also available as Amazon Machine Image (AMI) and can be deployed to the Amazon Elastic Compute Cloud (EC2) using StarCluster to allow on-demand processing of samples without requiring software installation or HPC maintenance.

However, for users wanting to install a local instance of NGSANE we now also provide an install script (<https://raw.github.com/BauerLab/ngsane/master/doc/installation/setup>) and slimmed-down version of the reference dataset (<https://github.com/BauerLab/ngsane/wiki/How-to-setup-ngsane>), which allows new users to easily get everything in place to test NGSANE’s functionality.

In addition, we clarified on the Wiki that not all programs need to be installed to test different analysis parts as all modules are self-encapsulated with clearly documented software and resource requirements (<https://github.com/BauerLab/ngsane/wiki/How-to-setup-ngsane>).

**Reviewer: 4**

For "the interjacent human quality control" for "Robust execution and full monitoring",  when will this control take effect and how this will coordinate with NGSANE's automatic workflow ?

We agree with the reviewer that the details of “interjacent human quality control” could be clarified and accordingly changed the sentence (pg2, para3)

In our experience, modular workflows are executed in stages with optional human quality control. NGSANE hence focuses on providing robust check pointing and intuitive report generation (Fig. 1b)

The following sentence was also added to the supplementary materials.

The Project Card can be created at the very end of the execution or at any intermittent step allowing human quality control throughout the stages of a project. The "Notes" and "Error" tabs specifically highlight, which (if any) subset of files contain error and the "Logfile" tab facilitates easy access to the file-specific log-files to identify the source of the problem. Once the faulty files are identified and the error is removed NGSANE allows the automated resubmission of the file-subset starting from the point of error.

The authors mentioned current software packages forces "a scientist to be two steps removed from the actual program call". What are the two steps? It does not seem clear.

The sentence was referring to the encapsulated program call in the wrapper-script (first-step) and the framework governing the execution of this wrapper script (second-step).

We have reworded the sentence to (pg1, para2)

However, currently available tools are either web-based services (e.g. Galaxy (Goecks et al., 2010)), where even API-based access to the web service functionality is not readily amenable to production-scale analysis practices, or heavyweight frameworks written in user-friendly languages like Python (SNAKEMAKE – Köster and Rahmann (2012), NESTLY – McCoy et al. (2013)), Scala (GATK’s QUEUE – Broadgsa (2013)) or Groovy (BPIPE – Sadedin et al. (2012)), which encapsulate the actual program call in a wrapper-script specific syntax, hindering the development of pipeline extensions.

Fig. 1 B) and C) are blurry and hard to read. Please improve.

We apologize for this and will follow up with the editorial team to ensure that the provided vector graphic is rendered appropriately during the remote LaTeX compilation to publication quality.