ALPACA Documentation

Release 0.3.1

Johannes Koester

June 11, 2015

Contents

ALPACA is a single nucleotide variant caller for next-generation sequencing data, providing intuitive control over the false discovery rate with generic sample filtering scenarios, leveraging OpenCL on CPU, GPU or any coprocessor to speed up calculations and an using HDF5 based persistent storage for iterative refinement of analyses within seconds.

Often, variant calling entails filtering different samples against each other, e.g. disease samples vs. healthy samples, tumor vs. normal or children vs. parents. The filtering can be seen as operations over the set algebra of variant loci. In general, the filtering is applied after calling. This results in the null hypothesis considered by the variant caller to not properly reflect the actual research question, which in fact entails the filtering. In consequence, controlling the false discovery rate becomes difficult.

Unlike other state of the art variant callers, **ALPACA integrates the filtering into the calling** by introducing a novel algebraic variant calling model. When calling, a filtering scenario can be specified with an algebraic expression like A - (B + C) with A, B and C being samples. Algebraic calling allows ALPACA to report **posterior probabilities** for a variant to occur in the **unknown set of true variant loci** in A that are not in B or C here. Since the probabilities reflect the filtering, they can be directly used to **intuitively control the false discovery rate**.

ALPACA splits variant calling into a preprocessing step and the actual calling. Preprocessed samples are stored in HDF5 index data structures. In a lightweight and massively parallel step, the sample indexes are merged into an optimized index. On the optimized index, **variant calling becomes a matter of seconds**. Upon the addition of a sample, merging and the calling have to be repeated. The sample indexes of the other samples remain untouched, **avoiding redundant computations**.

Algorithmic and mathematical details will be described in my thesis:

> Parallelization, Scalability and Reproducibility in Next-Generation Sequencing Analysis, Johannes Köster, 2015 (work in progress)

CHAPTER 1

Prerequisites

ALPACA needs

- Linux
- Python >= 3.3
- Numpy >= 1.7
- PyOpenCL >= 2013.1
- h5py >= 1.8.4
- samtools >= 1.0
- mawk
- a working OpenCL device (CPU, GPU, a coprocessor like Intel Xeon Phi or an FPGA)

Python 3 should be installed on most systems. You can make Debian and Ubuntu ready for installing ALPACA by issueing:

```
$ sudo apt-get install python3-setuptools python3-numpy python3-h5py samtools mawk
```

Without admin rights, we recommend to use a userspace Python 3 distribution like https://store.continuum.io/cshop/anaconda.

If you want to use ALPACA on the GPU, a decent NVIDIA or AMD GPU with the proprietary drivers installed should be enough. On Ubuntu and Debian, you can install them via:

```
$ sudo apt-get install nvidia-current
```

or:

```
$ sudo apt-get install fglrx
```

To use ALPACA with the CPU, you need an OpenCL runtime installed. You can e.g. install the AMD APP SDK (which will work on any x86 CPU) from here: http://developer.amd.com/tools-and-sdks/opencl-zone/amd-accelerated-parallel-processing-app-sdk

CHAPTER 2

Installation

Once the prerequisites are in place, ALPACA can be installed and updated with:

```
$ easy_install3 --user -U alpaca
```

CHAPTER 3

Usage

Usage of ALPACA consists of three major steps.

- sample indexing
- index merging
- calling

Given mapped reads for a sample *A* in BAM format and a reference genome in FASTA format, a sample index can be created with:

```
$ alpaca index reference.fasta A.bam A.hdf5
```

Here, various parameters like the expected ploidy of the sample can be adjusted. The resulting index *A.hdf5* will be much smaller than the BAM file. Merging indexes for samples *A* and *B* is achieved with:

```
$ alpaca merge A.hdf5 B.hdf5 all.hdf5
```

Finally, calling can be performed on the merged index. ALPACA allows to specify query expressions at the command line by representing the union operator with a plus sign and the difference operator with a minus sign. The variant calls are streamed out in VCF:

```
$ alpaca call --fdr 0.05 all.hdf5 A-B > calls.vcf
```

Here, we limit the desired rate of false discoveries to 5%. To assess the biological importance of a variant, it is useful to annotate it with additional information like the gene it may be contained in, its effect on a protein that is encoded by the gene or whether it is already known and maybe associated to some disease. ALPACA can annotate a VCF file with such information, using the Ensembl Variant Effect Predictor web service. Since the VCF format is rather technical, ALPACA can compose a human readable HTML file summarizing the calls. We can combine the two commands using Unix pipes:

```
$ alpaca annotate < calls.vcf | alpaca show > calls.html
```

For further information on various parameters of all steps (e.g. how to select the compute device) can be obtained with:

```
$ alpaca --help
```

CHAPTER 4

News

13 Jan 2014	Release 0.3.1 of ALPACA. Fixed imprecision in strand bias results. Further, this release introduces the k-relaxed intersection operator. A locus is contained in the k-relaxed intersection of a given set of samples if and only if it is variant in at least k samples.
2 Dez 2014	Release 0.2.4 of ALPACA. Further improved HTML output of alpaca show.
1 Dez 2014	Release 0.2.3 of ALPACA. Improved HTML output of alpaca show.
30 Nov 2014	Release 0.2.2 of ALPACA. This initial release provides all functionality descibed in my thesis "Parallelization, Scalability and Reproducibility in Next-Generation Sequencing Analysis".

4.1 License

ALPACA is available under the `MIT license`.

4.2 Analysis Pipeline

The pipeline used for the analysis done in the thesis can be obtained `here`.

4.3 HTML summary of variant calls

With the show subcommand, ALPACA allows to provide a human readable summary of variant calls in a single HTML5 file. An example can be seen `here`.

4.4 Author

Johannes Köster

Genome Informatics, Institute of Human Genetics, Faculty of Medicine, University of Duisburg-Essen

johannes.koester@gmail.com

http://johanneskoester.bitbucket.org