Description of expands

Noemi Andor

March 17, 2018

Contents

1 Introduction

This document contains examples to help a user understand the ExPANdS model. Users who are familiar with the model or who would like to try a quick test-run first should use function `runExPANdS` instead, which bundles the functionalities demonstrated here. Expanding Ploidy and Allele Frequency on Nested Subpopulations (ExPANdS) characterizes genetically diverse subpopulations (SPs) in a tumor using copy number and allele frequencies derived from exome- or whole genome sequencing input data [1]. Given a set of somatic point mutations detected in a tumor sample and the copy number of the mutated loci, ExPANdS identifies the number of clonal expansions within the tumor, the relative size of the resulting subpopulations in the tumor bulk and the genetic landscape unique to each subpopulation. Sequencing errors, mapping errors and germline

mutations have to be filtered first. The remaining set of somatic point mutations can be extended to contain loss of heterozygosity (LOH), that is loci with heterozygous germline polymorphisms where the mutated allele is overrepresented in the cancer cell. For tumor types with a low number of somatic point mutations, this approach can provide a sufficient number of somatic events for the subsequent procedure [1]. The model predicts subpopulations based on two assumptions:

- Two independent driver-events of the same type will not happen at the exact same genomic position in two different cells. Therefore, no more than two distinct cell populations co-exist with respect to a specific locus.
- Multiple passenger mutations accumulate in a cell before a driver mutation causes a clonal expansion. Thus, each clonal expansion is marked by multiple mutations.

These two assumptions are translated into the ExPANdS model in five main steps: cell frequency estimation, clustering, filtering, assignment of mutations to clusters and phylogenetic tree estimation. The following example demonstrates each of these steps separately. The main function `runExPANdS` performs all five steps. The robustness of the subpopulation predictions by ExPANdS increases with the number of mutations provided. It is recommended that at least 200 mutations are used as an input to obtain stable results.

2 Data

We illustrate the utility of ExPANdS on data derived from exome sequencing of a Glioblastoma tumor (TCGA-06-0152-01) from TCGA. Somatic mutations and LOH have been obtained by applying MuTutect [2] on the tumor derived BAM file and the patient-matched normal BAM file. Copy number segments have been obtained by a circular binary segmentation algorithm. We load the data into the workspace and assign each mutation the copy number of the segment in which the mutation is embedded:

```
> library(expands)
> ##load mutations:
> data(snv);
> ## use only a subset of mutations (to reduce time required to run example):
> set.seed(6); idx=sample(1:nrow(snv), 80, replace=FALSE); snv=snv[idx,];
> ##load copy number segments:
> data(cbs);
> ##assign copy numbers to point mutations:
> dm=assignQuantityToMutation(snv,cbs,"CN_Estimate");

[1] "Assigning copy number to mutations..."
[1] "Finding overlaps for CBS segment 100 out of  120 ..."
[1] "... Done."
```

Note that we limit the number of mutations used to 80 to accelerate the computation. In practice however, the inclusion of all available mutations is recommended, as the robustness and accuracy of the algorithm depends on the completeness of the input.

3 Parameter Settings

Next we set the parameters for the subsequent prediction. Type `help(runExPANdS)` for more information on these parameters.

```
> ##parameters
> max_PM=6; maxS=0.7; precision=0.018;
> plotF=1;
> ##the name of the sample
> snvF="TCGA-06-0152-01";
```

4 Predicting coexisting subpopulations with ExPANdS

Now we are ready to predict the number of clonal expansions in TCGA-06-0152-01, the size of the resulting subpopulations in the tumor bulk and which mutations accumulate in a cell prior to its clonal expansion.

4.1 Cell frequency estimation

First we calculates P - the probability density distribution of cellular frequencies for each single mutation separately. For each cellular frequency f, the value of $P(f)$ reflects the probability that the mutation is present in a fraction f of cells. For more information see `help(cellfrequency_pdf)`. This step may take several minutes to complete.

```
> ##calculate cell frequency probability distribution for each mutation
> cfd=computeCellFrequencyDistributions(dm, max_PM, p=precision)

[1] "Computing cell-frequency probability distributions..."
[1] "Processed 20 out of  80 SNVs --> success:  20 / 20"
[1] "Processed 40 out of  80 SNVs --> success:  40 / 40"
[1] "Processed 60 out of  80 SNVs --> success:  60 / 60"
[1] "Processed 80 out of  80 SNVs --> success:  80 / 80"
[1] "...Done."
```

In the subsequent step - `clusterCellFrequencies` - we will use only those mutations for which the cell frequency estimation was successful:

```
> ##cluster mutations with valid distributions
> toUseIdx=which(apply(is.finite(cfd$densities),1,all) )
```

In this case the cell-frequency probability distributions could be estimated for all mutations.

4.2 Clustering and Filtering

Next we find overrepresented cell frequencies using a two-step clustering procedure. Based on the assumption that passenger mutations occur within a cell prior to the driver event that initiates the expansion, each clonal expansion should be marked by multiple mutations. Thus SNVs and CNVs that took place in a cell prior to a clonal expansion should be present in a similar fraction of cells and leave a similar trace during their propagation. The aim is to find common peaks in the distribution of $P_l(f)$ for multiple mutated loci l. In the first step, mutations with similar $P_l(f)$ are grouped together by hierarchical cluster analysis of the probability distributions $P_l(f)$ using the Kullback-Leibler divergence as a distance measure. This step may take several minutes or hours to complete, depending on the number of mutations provided. In the second step, each cluster is extended by members with similar distributions in an interval around the cluster-maxima (core-region). Clusters are pruned based on statistics within and outside the core region [1]. All these steps are performed within the function `clusterCellFrequencies`:

```
> SPs=clusterCellFrequencies(cfd$densities[toUseIdx,], p=precision)

[1] "Clustering  80 probability distributions..."
[1] "Clustering agglomeration method: average"
[1] "0 SNVs excluded due to non-finite pdfs"
[1] "Done"
[1] "Filtering Clusters..."
[1] "0 % completed"
[1] "10 % completed"
[1] "20 % completed"
[1] "30 % completed"
[1] "40 % completed"
[1] "50 % completed"
[1] "60 % completed"
[1] "70 % completed"
[1] "80 % completed"
[1] "90 % completed"
[1] "Done."

> SPs=SPs[SPs[,"score"]<=maxS,]; ## exclude SPs detected at high noise levels
```

At this point we already know that four subpopulations have been predicted to coexist in this tumor:

```
> print(SPs)
```

```
     Mean Weighted     score precision nMutations
[1,]         0.154 0.6334136     0.018         13
[2,]         0.262 0.6059665     0.018          6
[3,]         0.388 0.6552837     0.018          5
[4,]         0.838 0.4211439     0.018         14
```

4.3 Assignment of SNVs to clusters

Now, all that remains to be done is to assign each point mutation to one of the predicted subpopulations. A point mutation is assigned to the subpopulation C, whose size is closest to the maximum likelyhood cellular frequency of the point mutation. Cell frequency probability distributions are calculated for four alternative evolutionary scenarios (for more information see details of function `assignMutations`). The mutated loci assigned to each subpopulation cluster represent the genetic profile of each predicted subpopulation.

```
> ##assign mutations to subpopulations:
> aM= assignMutations( dm, SPs, verbose = F)
```

```
[1] "Resolving potential phylogeny conflicts among 3 loci..."
```

`aM$dm` contains the input matrix `snv` with seven additional columns, including: `SP` - the size of the subpopulation to which the mutation has been assigned; and `%maxP` - confidence of the assignment. See `help(assignMutations)` for more information on the output values of this function.

4.4 Visualization of predicted subpopulations

Now we plot the coexistent subpopulations predicted in the previous steps.

```
> o=plotSPs(aM$dm, snvF,cex=1)
```

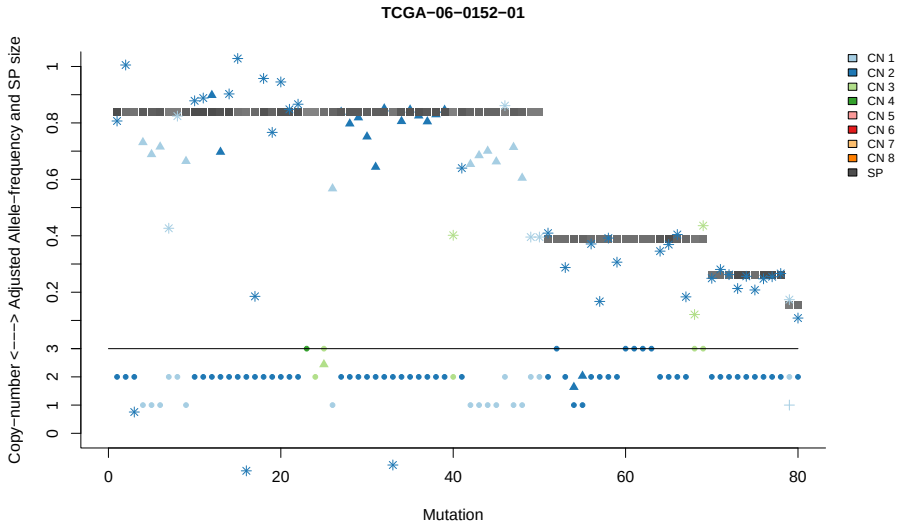

Figure 1: Coexistent subpopulations determined by ExPANdS in a Glioblastoma genome. Four subpopulations were identified based on the allele-frequency and copy number of 80 mutations detected within the cancer-genome. Subpopulations were present in 84%, 39%, 26% and 15% of the sample (y-axis). For each of the 80 exonic mutations (x-axis) we show: - the subpopulation to which the mutation has been assigned (squares), - the copy number of the locus in that subpopulation (dots) and - the adjusted allele frequency of the mutation (stars - somatic SNVs, triangles - LOH). Allele frequencies and subpopulation specific copy numbers are colored based on the average copy number measured for the genomic segment within which the mutation is located. Subpopulations are colored based on the confidence with which the mutation has been assigned to the subpopulation (black - highest, white - lowest).

4.5 Inferring phylogenetic relations between subpopulations

We model the tumor's phylogeny based on pairwise distances between SPs. Pairwise phylogenetic distances between SPs are calculated from SP specific copy number profiles. First we have to assign SP specific copy numbers for the input genome segments obtained by circular binary segmentation:

```
> ##assigning copy number to subpopulations
> aQ=assignQuantityToSP(cbs, aM$dm, v=F)

[1] "Assigning copy number to SPs..."
```

The subpopulation phylogeny is obtained by running a neighbor-joining tree estimation algorithm on pairwise phylogenetic distances between SPs:

```
> ##building phylogeny
> spPhylo=buildPhylo(aQ,snvF,add = NULL)

[1] "Building phylogeny using bionjs algorithm"
[1] "Pairwise SP distances calculated as: % segments with identical copy number"
[1] "Insufficient copy number segments for SP_0.262. SP excluded from phylogeny"
[1] "distance-matrix saved under TCGA-06-0152-01.dist"
[1] "tree saved under TCGA-06-0152-01.tree"
```

Finally we plot the phyloegentic tree.

```
> plot(spPhylo$tree,cex=3,type = "c")
```

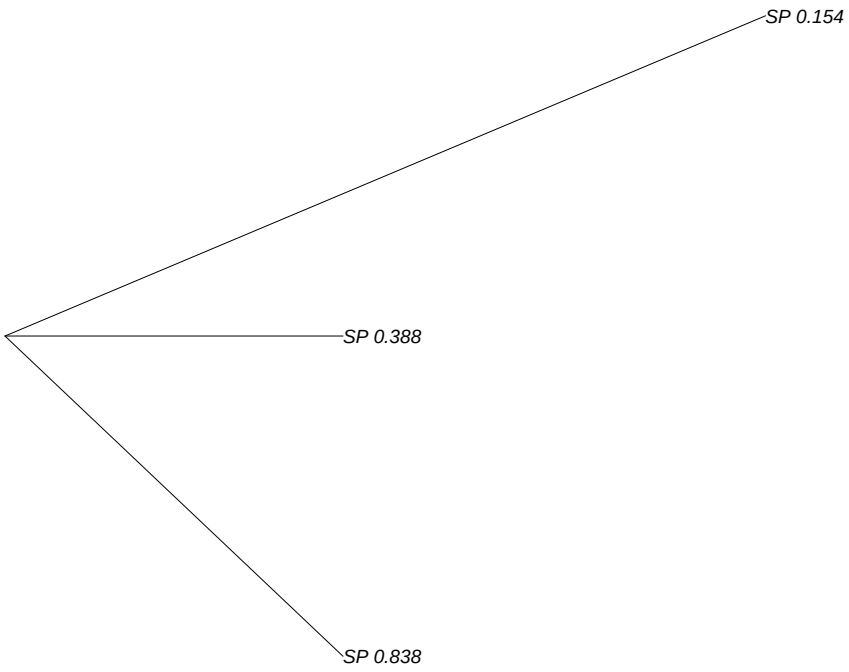

Figure 2: Phylogram representation of the inferred relations between three predicted SPs. Each branch spans proportional to the amount of copy number change between SPs.

5 Inferring phylogenetic relations between subpopulations from multiple geographical tumor samples

Next we integrate the subpoulations predicted in multiple, geographically distinct tumor-samples of a patient into one common phylogeny:

```
> #Patient and sample labels
> patient='ID_MRD_001';
> samples=c('_primPancreas','_metKidney','_metLung');
> output=patient;
> #The CBS files for each sample:
> cbs=as.list(paste(patient, samples,'.cbs',sep=""));
> #The SP files for each sample (previously calculated via runExPANdS-function):
> sps=as.list(paste(patient, samples,'.sps',sep=""));
```

We build a sample group for this patient to calculate the combined phylogeny:

```
> sampleGroup=list(cbs=cbs,sps=sps,labels=samples)
> tr=buildMultiSamplePhylo(sampleGroup,output,e = 0, plotF=0);
```

```
[1] "Processing sample 1 out of 3"
[1] "Assigning copy number to SPs..."
[1] "Assigning copy number to SPs..."
[1] "Processing sample 2 out of 3"
[1] "Assigning copy number to SPs..."
[1] "Assigning copy number to SPs..."
[1] "Processing sample 3 out of 3"
[1] "Assigning copy number to SPs..."
[1] "Assigning copy number to SPs..."
[1] "Building phylogeny using bionjs algorithm"
[1] "Pairwise SP distances calculated as: % segments with identical copy number"
[1] "distance-matrix saved under ID_MRD_001.dist"
[1] "tree saved under ID_MRD_001.tree"
```

```
> ##Tree tip color labels according to sample origin of SPs:
> jet <- colorRampPalette(c("#00007F", "blue", "#007FFF",
+             "cyan", "#7FFF7F", "yellow", "#FF7F00", "red", "#7F0000"))
> colmap = jet( length(sampleGroup$labels) )
> colors <- rep(colmap[1], each = length(tr$tip.label))
> for (i in 1: length(sampleGroup$labels) ) {
+      ii = grep(sampleGroup$labels[[i]], tr$tip.label)
+      colors[ii] = colmap[i]
+ }
```

Finally plot the inter-sample phylogeny:

```
> plot(tr, tip.col = colors, cex = 0.9, type = "u")
```

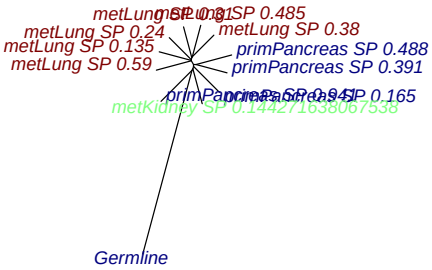

Figure 3: Phylogram representation of the inferred relations between SPs from three distinct geographical samples. Each branch spans proportional to the amount of copy number change between SPs. The germline copy number profile (assumed diploid throughout the genome) is included as control.

6 Acknowledgements

Special thanks to Dr. Ryan Morin for his contributions that have led to higher accuracy measures during simulations for mutation assignment to subpopulations, as well as advanced visualization features of assigned mutations. Thanks also to Dr. Ruchira S. Datta for her contributions to the structure and presentation of this manuscript.

References

[1] Noemi Andor, Julie Harness, Sabine Mueller, Hans Werner Mewes and Claudia Petritsch. *ExPANdS: Expanding Ploidy and Allele Frequency on Nested Subpopulations.* Bioinformatics (2013).

[2] Cibulskis K, Lawrence MS, Carter SL, Sivachenko A, Jaffe D, Sougnez C, Gabriel S, Meyerson M, Lander ES, Getz G. *Sensitive detection of somatic point mutations in impure and heterogeneous cancer samples.* Nat Biotech (2013).