Introduction to *DriverProbabilities*

Bastiaan Van der Roest [1]

[1]University Medical Center Utrecht, Utrecht, The Netherlands

October 28, 2019

Contents

1 Introduction

In vitro culturing stem cells for application in for instance regenerative medicine, takes potential risks on genetic diseases. Genetic changes, caused by mutation accumulation during culturing, can lead to many genetic disorders, including cancer. In cancer, these genetic changes are mostly on positions which benefit the affected cell, by activating oncogenic genes. A large number of those positions is already found and can be used to calculate a risk on getting the so-called driver mutations.

During *in vitro* culturing the mitosis rate and doubling time of a cell population can be monitored, along with the number of mutations per time. Not only physiological changes can be monitored, also the genetic changes. Mutation accumulation leaves a spectrum of different mutation types behind in the genome, giving potential risks on the mutation types. This spectrum can differ between cell types and even between cells of the same type. Mutations occur in the whole genome, however cells have developed methods to protect the important parts of the genome against mutation accumulation. Genomic regions are in general more depleted in mutations than whole genome, so in order to predict the risk on mutations in cancer, this depletion has to be taken into account.

The package *DriverProbabilities* is developed to use all of the above data, that is the mitosis rate, doubling time, mutational spectrum and depletion tests, to calculate the number of oncogenic driver mutations occuring during culturing of stem cells. Background information of the package is described in the article found on bioRxiv: `https://www.biorxiv.org/content/biorxiv/early/2018/09/29/430165.full.pdf`

2 Load package

If you are a first time user, read the README on how to install the package from Github. After installing load the package by using the path to the package:

```
> devtools::load_all("../../DriverProbs")
```

3 Data

To build a probability model for oncogenic driver mutations, data is needed about the cell dynamics and genetics. First store all input data into variables and then process the data to get it in the right format for modelling.

3.1 Import data into variables

Start with importing data about the cell dynamics, including cell type, doubling time, mitosis rate etc. To read the data as strings, first set `options(stringsAsFactors = F)`. Then set the basic parameters:

```
> # Number of cells at day 0
> init_numb_cells <- 1
> # The length of the whole genome
> wg_length <- 2881033286
> # The percentage of the Coding Sequence of the whole genome
> CDS_perc <- 0.015
```

The information about which sample is which cell type must be included:

```
> # Set the in vitro cell types
> cell_types <- c("iPS", "liver", "SI")
> # Give the file with sample name and cell type information
> ST <- "Sampletypes.txt"
> # Set the in vivo tissue
> invivo_tissue <- "colon"
```

Now set the parameters of the cell dynamics:

```
> # Doubling time for cell population in hours
> doubling_time <- list('iPS'=23.08, 'liver'=46 ,'SI'=44.38)
> # Mitosis rate for cell population
> mitosis_rate <- list('iPS'=18, 'liver'=26, 'SI'=26)
> # Mutation rate per genome per doubling
> mutation_rate <- 'mutation_rates.txt'
```

The time to culture the cells must be given in hours. In order to give the time in whole factors of generation cycle time, the mitosis rate, the least common multiplier between the cell types can be used:

```
> library(pracma)
> # Time to culture the cells in hours. Give time in whole factors of
> # generation cycle time
> t_culture <- Lcm(Lcm(mitosis_rate[['iPS']], mitosis_rate[['SI']]),
+                  mitosis_rate[['liver']])*10
```

Furthermore, give the data about the cell genetics, that is the mutational spectra of *in vitro* and *in vivo* grown cells and the results of the enrichment/depletion tests from the *MutationalPatterns* package.

```
> # File containing the mutational spectrum for all the samples together
> mut_spec_file <- 'Mutation_spectrum_tissue.txt'
> # File containing the mutational spectrum for the in vivo samples
> mut_spec_in_vivo_file <- 'Mutation_spectrum_in_vivo_tot.txt'
> # File containing the enrichment or depletion of the CDS in mutations
> CDS_dep_file_invitro <- 'CDS_gene_regions_distr.txt'
> CDS_dep_file_invivo <- 'CDS_gene_regions_distr_invivo.txt'
```

To see which mutations are ongenic drivers, import a catalog with oncogenic mutations, along with a file in which for each gene the mode of action is given.

```
> # File containing validated oncogenic mutations
> oncogenic_mutations <- 'catalog_of_validated_oncogenic_mutations.tsv'
> # File containing activity of driver genes while mutated
> gene_MoA <- 'gene_MoA.tsv'
```

As final input, state if you want to calculate the probabilities on oncogenic driver mutations per gene or over all genes. If per gene is wanted, specific genes can be selected:

```
> # Select if you want probabilities per gene of all genes at once
> prob_per_gene = F
> # Give a vector of the genes for which you want to visualize the probabilities
> wanted_genes = 'all'
```

3.2 Process data into the modelling format

To process the data to have it in the right format for using the model, the function `processData` can be runned. All needed variables for downstream functions will be stored in the global environment:

```
> processData(mutation_rate_file = mutation_rate,
+             oncogenic_mutations_file = oncogenic_mutations,
+             gene_MoA_file = gene_MoA,
+             CDS_dep_invitro_file = CDS_dep_file_invitro,
+             CDS_dep_invivo_file = CDS_dep_file_invivo,
+             sample_celltypes_file = ST,
+             invivo_tissue = invivo_tissue,
+             mut_spec_invitro_file = mut_spec_file,
+             mut_spec_invivo_file = mut_spec_in_vivo_file)
```

Per default, the newly made list `driver_counts` will contain counts of six single nucleotide variants ("C>A", "C>G", "C>T", "T>A", "T>C", "T>G"). It is also posible to give counts of the 96 mutation spectrum, taken the 5' and 3' prime nucleotide of a mutation basepair into account. Therefore run the `processData` function with the `trinucleotide` argument set to `True`:

```
> processData(mutation_rate_file = mutation_rate,
+             oncogenic_mutations_file = oncogenic_mutations,
+             gene_MoA_file = gene_MoA,
+             trinucleotide = T,
+             CDS_dep_invitro_file = CDS_dep_file_invitro,
+             CDS_dep_invivo_file = CDS_dep_file_invivo,
+             sample_celltypes_file = ST,
+             invivo_tissue = invivo_tissue,
+             mut_spec_invitro_file = mut_spec_file,
+             mut_spec_invivo_file = mut_spec_in_vivo_file)
```

This manual will continue with the six mutation types.

A check must be done to see if the cell types in the given files were spelled in the same way. If this is not the case, the model can take the wrong data for the cell types.

```
> cell_types

[1] "iPS"   "liver" "SI"

> names(probs_snv)

[1] "IPS"     "SI"      "liver"   "In_vivo"

> names(CDS_dep)

[1] "Liver"           "iPS"             "Small_Intestine"
```

Change the names if necessary:

```
> names(probs_snv)[names(probs_snv)=="IPS"] <- "iPS"
> names(CDS_dep)[names(CDS_dep) %in% c("Liver","Small_Intestine")] <- c("liver",
+                                                                      "SI")
```

The list `driver_counts` can be extended with genes which are not in the given catalog:

```
> driver_counts[['TP53']] <- table(c("C>T","C>T","C>T","C>G","C>T","C>T","T>C","C>T","C>A","C>T","C>A"))
> driver_counts[['BRAF.V600E']] <- table("T>A")
```

If only driver counts of genes not present in the catalog must be analyzed, it is optional to not fill the `driver_counts` list with `processData`:

```
> processData(mutation_rate_file = mutation_rate,
+             process_drivers = F,
+             CDS_dep_invitro_file = CDS_dep_file_invitro,
+             CDS_dep_invivo_file = CDS_dep_file_invivo,
+             sample_celltypes_file = ST,
+             invivo_tissue = invivo_tissue,
+             mut_spec_invitro_file = mut_spec_file,
+             mut_spec_invivo_file = mut_spec_in_vivo_file)
```

4 *In vitro* mutation accumulation and probabilities

4.1 Calculate the results

Once all the prerequired data is loaded and processed, the results for the *in vitro* mutation accumulation and oncogenic driver probabilities can be calculated with the `probDriverMut` function. Here, you can give the cell types and genes for which the results must be calculated.

```
> invitro_res <- probDriverMut(cell_types = cell_types,
+                              genes = names(driver_counts))
> head(invitro_res)
```

```
      cells         muts        probs cell_type gene stat
1  1.000000 0.000000e+00 0.000000e+00       iPS  all mean
2  1.717007 1.057971e-07 6.347887e-07       iPS  all mean
3  2.948113 2.874514e-07 1.089936e-06       iPS  all mean
4  5.061932 5.993532e-07 1.871428e-06       iPS  all mean
5  8.691373 1.134891e-06 3.213253e-06       iPS  all mean
6 14.923149 2.054413e-06 5.517172e-06       iPS  all mean
```

The statistics type (mean, mean minus standard deviation or mean plus standard deviations) can be also be given as a vector.

```
> invitro_res <- probDriverMut(cell_types = cell_types,
+                              genes = names(driver_counts),
+                              stat_type = c("mean",
+                                            "mean-std",
+                                            "mean+std"))
```

4.2 Plot the results

Before the results will be plotted, set the type of mutations and cell types in the order you want:

```
> invitro_res$cell_type <- factor(invitro_res$cell_type, levels = c("liver", "SI", "iPS"))
> invitro_res$gene <- factor(invitro_res$gene, levels = c("all", "TP53", "BRAF.V600E"))
```

Calculate the intersections of all plotted with a given horizontal line with the function `calcIntersects`:

```
> xintersects <- calcIntersects(r = invitro_res, hline = 1)
```

Then plot the results, along with the horizontal and vertical lines and save the figure as pdf with the filename given in `pdf_out`:

```
> plotMutationsPerCell(invitro_res,
+                      xmax = 1e13,
+                      ymax = 1e1,
+                      hline = 1,
+                      vline = xintersects,
```

```
        +                       ylab = 'Number of mutations',
        +                       pdf_out = pdf_out)
```