STstuff package vignette

David Venet

February 15, 2024

1 Introduction

This vignette quickly presents the features of the STstuff package:

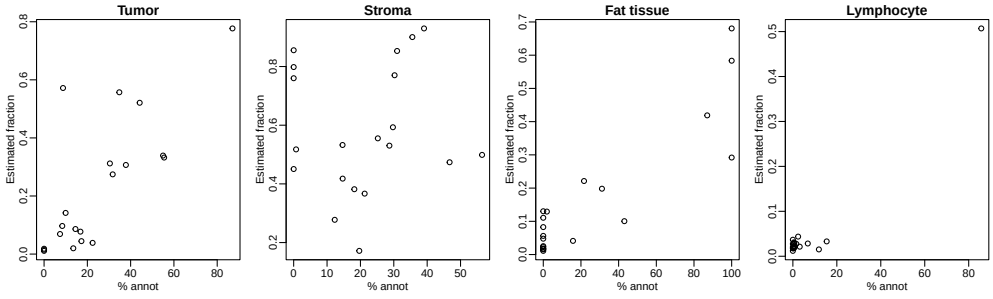
1. Estimation of the proportion of each annotation is each spot
2. Determination of expression in annotations
3. Deconvolution of megaclusters
4. Ecotypes
5. Deconvolution of k-means prototypes

2 Estimation of the proportion of each annotation is each spot

Detailed morphological annotations were done on 94 triple negative breast cancer (TNBC). Using those, the fraction of each annotation in each spot (e.g. the percentage of the spot annotated as tumor) was determined. Regressors were designed to estimate those fractions from the gene expression data of each spot. Those regressors are available via the command `classifySpots`.

For instance, those regressors can be applied on an example dataset. This correspond to 10 random spots from the TNBC1 sample from the paper. The normalized counts are in `cnts` while `annot` contains the fraction of each annotation in each spot.

```
> library(STstuff);
> library(parallel);
> data(TNBC1);
> reg = classifySpots(cnts)
> par(mfrow=c(1,4), mar=c(3,3,1.5,.5), mgp=c(1.5,.5,.0))
> for (i in c("Tumor", "Stroma", "Fat tissue", "Lymphocyte"))
+ { plot(annot[,i], reg[,i], xlab="% annot", ylab="Estimated fraction", main=i);
+ }
```



Of course, those regressors were made on TNBC on a specific platform and there is no guarantee whatsoever of their usefulness on another dataset.

3 Determination of expression in annotations

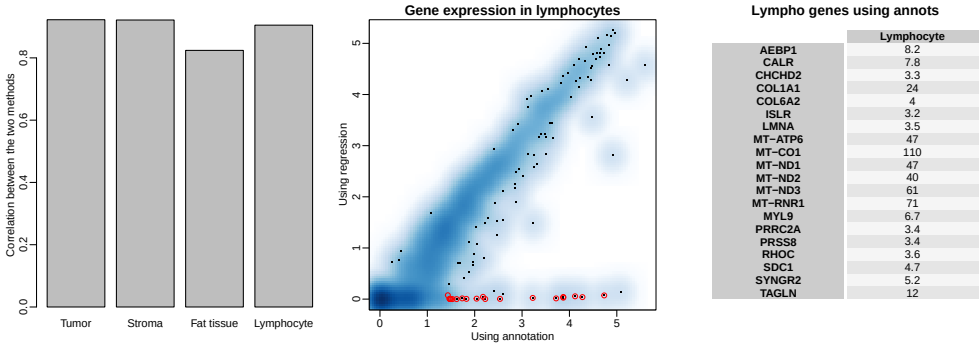
We assume that for each spot of an ST experiment there is an estimate of the presence of some annotations. A possibility is that a pathologist would have annotated the slide so that it is possible to estimate the fraction of tumor, stroma, lymphocytes and so on in each spot. Another possibility is to use the estimates obtained with `classifySpots`. Using those, by deconvolution it is possible to estimate the expression related to each annotation, i.e. the gene expressions that would be obtained if only the cells in those annotations were sequenced. In our experience, using the estimates obtained by regression leads to better results than using the original annotation counts.

As an example, the method is applied to the example dataset. Note that with only 10 spots great results are not to be expected. We will compare the deconvolution using the original annotations and the estimates computed supra.

```
> w = c("Tumor", "Stroma", "Fat tissue", "Lymphocyte")
> a = annot[,w]; a = a/rowSums(a);
> x = cnts[rowSums(cnts)>=50,];
> ex = expressionInAnnotations(t(x), a, maxIter=3, quiet=TRUE, mccores=FALSE)
> prot = ex$W; prot = 1e4*prot/rep(colSums(prot), each=nrow(prot));
> ex2 = expressionInAnnotations(t(x), reg[,w], maxIter=3, quiet=TRUE, mccores=FALSE)
> prot2 = ex2$W; prot2 = 1e4*prot2/rep(colSums(prot2), each=nrow(prot2));
```

We can see how well the two deconvolution fit (in `prot` and `prot2`). The correlations are typically high. However, focusing on genes expressed in lymphocytes using the annotations, we can see that those genes are more typical of stroma than lymphocytes.

```
> library(nonSTstuff)
> par(mfrow=c(1,3), mar=c(3,3,1.5,.5), mgp=c(1.5,.5,.0))
> barplot(diag(cor(log(prot+1), log(1+prot2))), ylab="Correlation between the two methods")
> smoothScatter(log(prot[,4]+1), log(prot2[,4]+1), xlab="Using annotation",
+   ylab="Using regression", main="Gene expression in lymphocytes")
> w = which(prot2[,4]<.1 & prot[,4]>3);
> points(log(prot[w,4]+1), log(prot2[w,4]+1), col='red')
> plotTable(signif(prot[w,4,drop=FALSE],2));
> title(main="Lympho genes using annots");
```



Lympho genes using annots

	Lymphocyte
AEBP1	8.2
CALR	7.8
CHCHD2	3.3
COL1A1	24
COL6A2	4
ISLR	3.2
LMNA	3.5
MT−ATP6	47
MT−CO1	110
MT−ND1	47
MT−ND2	40
MT−ND3	61
MT−RNR1	71
MYL9	6.7
PRRC2A	3.4
PRSS8	3.4
RHOC	3.6
SDC1	4.7
SYNGR2	5.2
TAGLN	12

4 Recovery of megaclusters from another dataset

Megaclusters (MC) are inter-patients clusters of intra-patient clusters. In the TNBC dataset, we determined that there were 14 MCs. The idea is to recover from gene expression on bulk samples the level of presence of those MC. The package proposes two functions to do this: a general function (`clustersInOtherDS`) that could be used for any MCs, and the `computeMC` function that uses the MCs that were found in TNBCs.

5 Determination of ecotypes in another dataset

Ecotypes (ET) are groups of tumors that were found by clustering the MC estimates. It is possible to recover them in other datasets. The packages proposes the function `computeET` to recover them, starting from the MC estimates obtained by `computeMC`.

6 Deconvolution of k-means prototypes

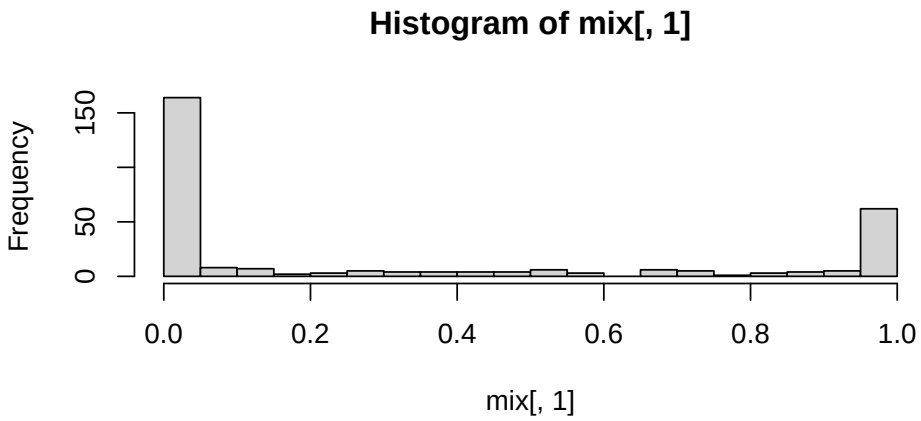
Using k-means on spots from ST, we found that because many spots were not pure (i.e. they were localised on the border between different regions) the prototypes of each k-means was a mixture of different signals. To address this issue, we designed a method to obtain k-means prototypes that were not as correlated.

We simulate data, where each spot is a mixture of 3 basic vectors. Those vectors are chosen to be correlated, as gene expression is. The parameters of the mixture are taken from a Dirichlet distribution, so that they are often close to 1 but not always. Then we take the mixture from each spot as the mean of a negative binomial and draw random values from there.

```
> library(STstuff);
> library(dirmult); # for the Dirichlet (in the simulation)
> Nprot = 3; Nspot = 100*Nprot;
> Ngene = 50;
> gb = rexp(Ngene)^2+1; # Gene specific values
> prot = matrix(rexp(Ngene*Nprot), ncol=Nprot) * gb; # prototypes
> protl = log(1+1e4*t(prot)/colSums(prot))
> mix = rdirichlet(Nspot, alpha=c(1,1,1)*.1) # Mixture from all 3 classes for each spot
> base = mix %*% t(prot)
> X = matrix(rnbinom(base, mu=base, size=1), ncol=ncol(base));
```

The distribution of the mix is often close to 0 or 1, as wanted.

```
> hist(mix[,1], breaks=30)
```

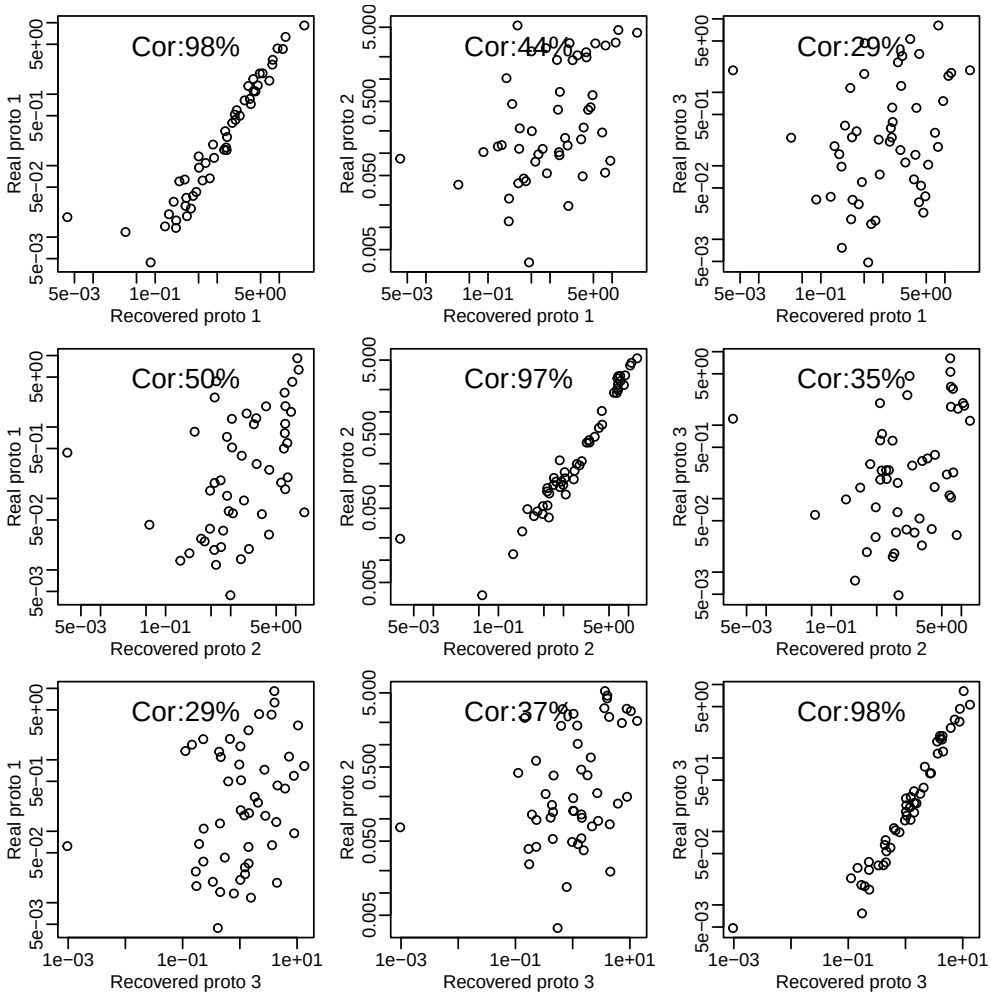


From here, we first do a k-means to get the clusters. We start from the real prototype to ensure that we get the right ordering.

```
> y = log(1+1e4*X/rowSums(X))
> km = kmeans(y, prot1);
```

We can see that the cluster centers, although correlated to the correct prototypes, are also correlated to each other.

```
> pr = t(km$centers)
> pr0 = (exp(pr)-1)*colSums(prot)/1e4;
> par(mfrow=c(3,3), mar=c(3,3,.5,.5), mgp=c(1.5,.5,0))
> for (i in 1:3)
+ { for (j in 1:3)
+   { plot(prot[,i], pr0[,j], log='xy', xlab=paste("Recovered proto", i),
+       ylab=paste("Real proto", j)); #abline(0,1)
+     mtext(paste0("Cor:", round(cor(prot[,i], pr[,j], method='s')*100), "%"), side=3, line=-2)
+   }
+ }
```

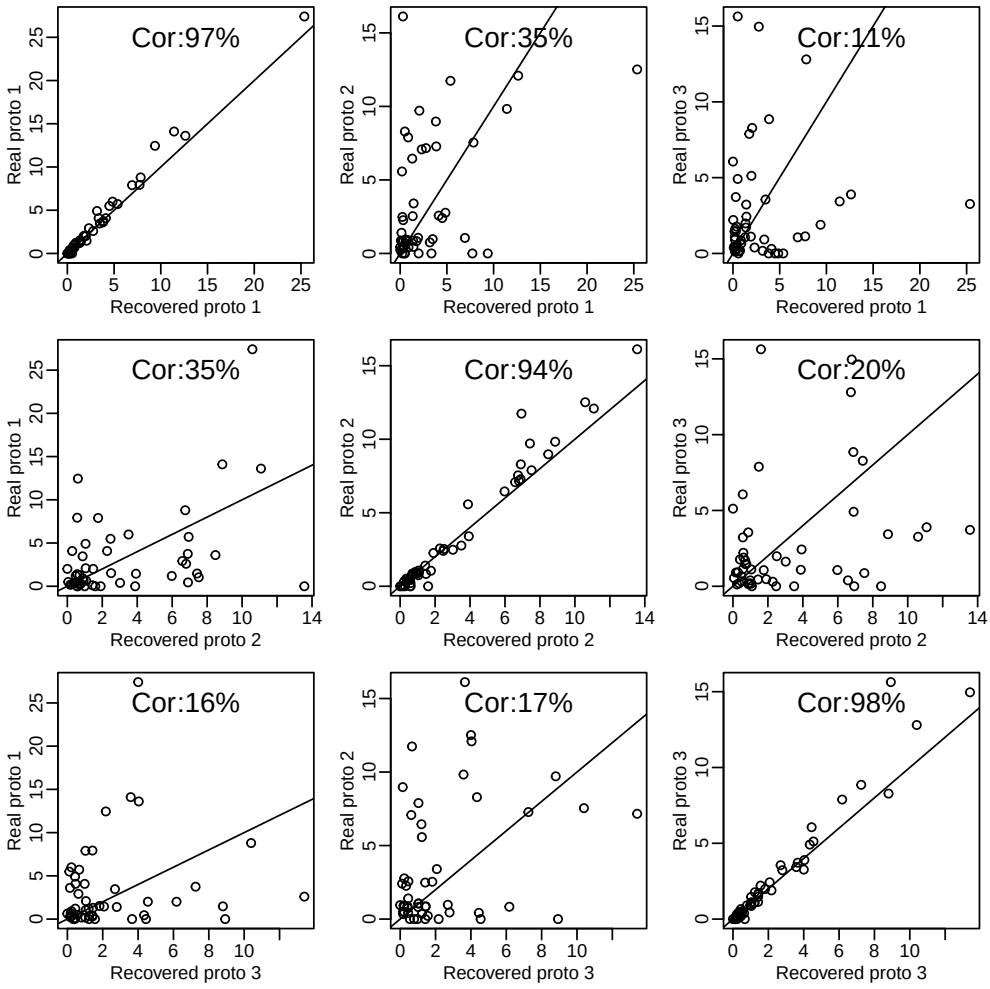


We can deconvolute the cluster.

```
> deconv = deconvoluteClusters(X, km$cluster, Niter=10, clean=FALSE)
> pr2 = deconv$proto$proto; # pr2 are the new prototypes
```

The new prototypes are still well correlated with the real prototypes, but they are also less correlated to each other. Note that some correlation is still expected because the original prototypes are also correlated.

```
> par(mfrow=c(3,3), mar=c(3,3,.5,.5), mgp=c(1.5,.5,0))
> for (i in 1:3)
+ { for (j in 1:3)
+   { plot(prot[,i], pr2[,j], xlab=paste("Recovered proto", i),
+       ylab=paste("Real proto", j)); abline(0,1)
+     mtext(paste0("Cor:", round(cor(prot[,i], pr2[,j], method='s')*100), "%"), side=3, line=-2)
+   }
+ }
```


Cor:97%
Real proto 1
Recovered proto 1
Cor:35%
Real proto 2
Recovered proto 1
Cor:11%
Real proto 3
Recovered proto 1
Cor:35%
Real proto 1
Recovered proto 2
Cor:94%
Real proto 2
Recovered proto 2
Cor:20%
Real proto 3
Recovered proto 2
Cor:16%
Real proto 1
Recovered proto 3
Cor:17%
Real proto 2
Recovered proto 3
Cor:98%
Real proto 3
Recovered proto 3