```
> library(CellCODE)
```

1 Kidney Transplant Dataset

Load the Data from GSE20300 (kidney transplant data) and pure cell datasets (DMAP and IRIS).

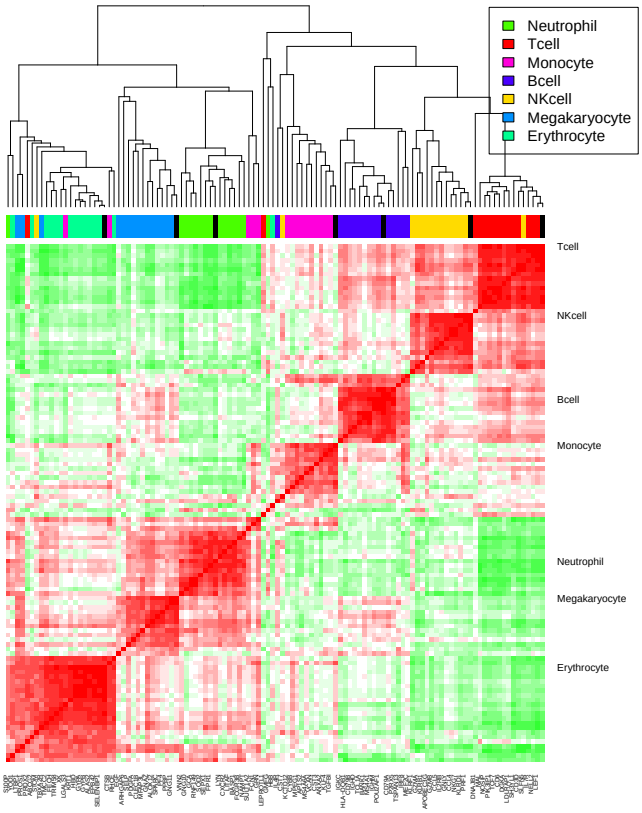
```
> data("GSE20300")
> data("GSE20300.grp")
> data("cellCounts")
> data("Garvan")
> data("IRIS")
> data("DMAP")
```

Make a cell type tag matrix using the IRIS dataset for lymphocytes and the DMAP dataset for megakaryocyte and erythrocyte.

```
> dmapTag=tagData(DMAP[, c("MEGA2", "ERY5", "MONO2", "GRAN3",
+                          "DENDA1", "TCELLA7",
+                          "NKA2",   "BCELLA2")], 0.7, max=15,
+                 ref=GSE20300, ref.mean=F)
> colnames(dmapTag)[1:2]=c("Megakaryocyte", "Erythrocyte")
> irisTag=tagData(IRIS[,c("Neutrophil-Resting","CD4Tcell-N0",
+                         "Monocyte-Day0", "Bcell-naïve",
+                         "NKcell-control" )],2, max=15,
+                 ref=GSE20300, ref.mean=F);
> colnames(irisTag)=c("Neutrophil","Tcell", "Monocyte",
+                     "Bcell", "NKcell" )
> tmpTag=combineTags(irisTag, dmapTag[,1:2])
```

Estimate proportions using the tag genes

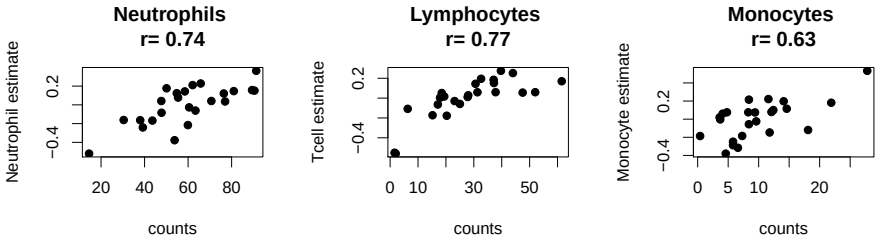
```
> SPVs=getAllSPVs(GSE20300, grp=GSE20300.grp, tmpTag, method="mixed",
+                 plot=T, mix.par=0.3)
```



The correlation structure looks block-like suggesting that we are accurately capturing separate variance components for these cell types.

Plot proportion variables against Coulter counter measurements

```
> par(mfrow=c(1,3), cex=1.1)
> for ( i in 1:3){
+   plot( cellCounts[,i], SPVs[,i], pch=19,
+         ylab=paste(colnames(SPVs)[i], "estimate"),
+         xlab="counts")
+   title(paste(colnames(cellCounts)[i],"\nr=",
+               format.pval(cor(SPVs[,i],
+                               cellCounts[,i], method="p"),
+                           digits=2)))
+ }
```



2 Simulated Data

Create a 5 cell type subset of pure cell expressions

```
> IRISsub=IRIS[,c("Neutrophil-Resting", "CD4Tcell-N0",
+                 "Monocyte-Day0", "DendriticCell-Control", "Bcell-naïve")]
```

Get rid of low expression genes

```
> mm=apply(IRISsub,1,max)
> IRISsub=IRISsub[mm>4,]
```

Simulate the data. Pure expression files are used in linear space.

```
> set.seed(123)
> simRes=simulateMixture(pureData=2**IRISsub, targetVals=t(cellCounts), cellpop=2)
```

Extract the simulated data. Add noise and log transform.

```
> simData=simRes$data
> simData=randomizeExp(simRes$data, sd=0.22)
> simData=log2(simData)
> print(mean(varExplained(simData, model.matrix(~1+cellCounts))))
```

```
[1] 0.3351131
```

Create the group variable for differential expression analysis. The simulatieMixture() function makes two equal groups

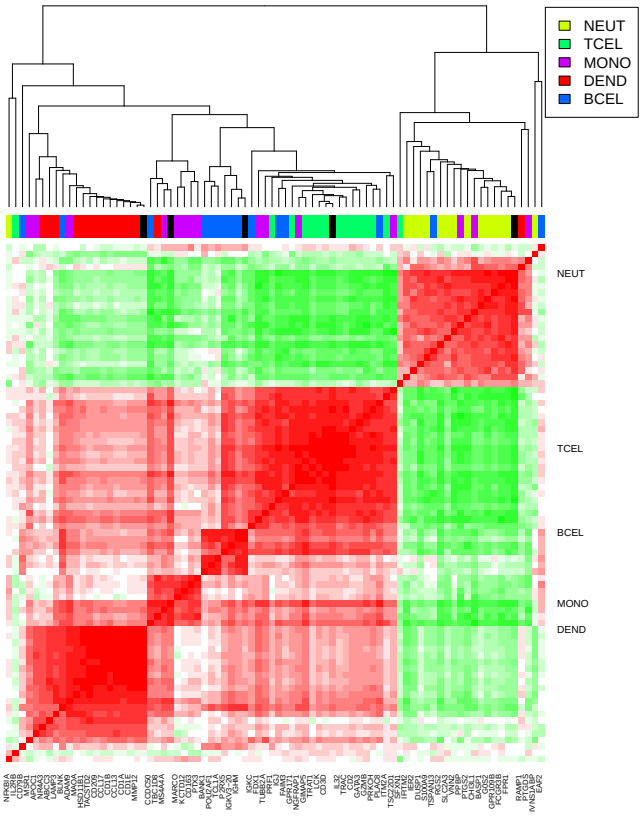
```
> grp=as.factor(rep(c(1,2), each=12))
```

Use the supplied subset of the Garvan dataset to generate tag Genes for the same cell types

```
> Gt=tagData(Garvan, cutoff=2.5, max=15, ref=simData)
```

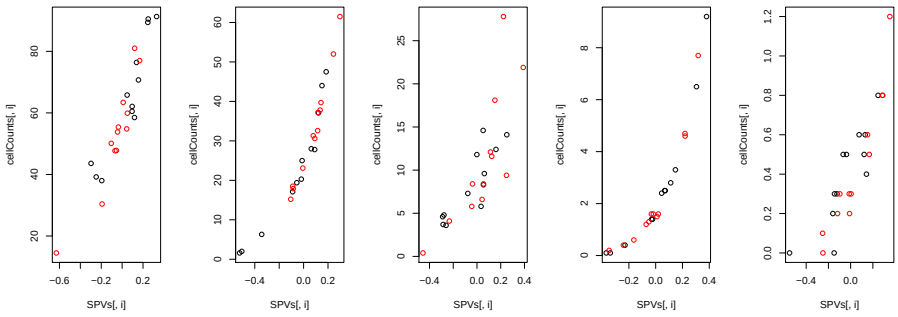
Estimate proportions.

```
> SPVs=getAllSPVs((simData),grp=grp, dataTag=Gt[,], plot=T, method="mixed")
```



The IRIS and Garvan dataset do not perfectly agree but the proportion estimation is still accurate

```
> par(mfrow=c(1,5));for (i in 1:5){plot(SPVs[,i], cellCounts[,i], col=grp)}
```



Test for differential expression with and without covariates

```
> ttraw=lm.coef(simData,model.matrix(~1+grp))
> ttcount=lm.coef(simData,model.matrix(~1+grp+cellCounts))
> ttcellcode=lm.coef(simData,model.matrix(~1+grp+SPVs))
> deRes=c(numAtFDR(simRes$vals!=0,abs(ttraw$tstat[,2]), at=0.1),
+ numAtFDR(simRes$vals!=0,abs(ttcount$tstat[,2]), at=0.1),
+ nn<-numAtFDR(simRes$vals!=0,abs(ttcellcode$tstat[,2]), at=0.1))
> deRes=cbind(deRes)
> rownames(deRes)=c("raw", "count", "CellCODE")
> message("number of genes at FDR=0.1")
> print(deRes)

         deRes
raw        436
count      446
CellCODE   474
```

Assign the *detectable* differentially expressed genes to a cell type

```
> cutoff=sort(abs(ttcellcode$tstat[,2]),T)[nn]
> iiDF=which(abs(ttcellcode$tstat[,2])>=cutoff&simRes$vals!=0)
> resf=cellPopF(simData[iiDF,], grp, SPVs)
> rest=cellPopT(simData[iiDF,], grp, SPVs)
> resc=cellPopC(simData[iiDF,], grp, SPVs)
> fracs=c(colSums(isRowmax(resc))[2], colSums(isRowmax(rest))[2],colSums(isRowmax(resf))[2])
> fracs=cbind(fracs)
> rownames(fracs)=c("cor Test", "T Test", "F Test")
> message("fraction correctly assigned")
> print(fracs)

            fracs
cor Test 0.4192037
T Test   0.7002342
F Test   0.7166276
```