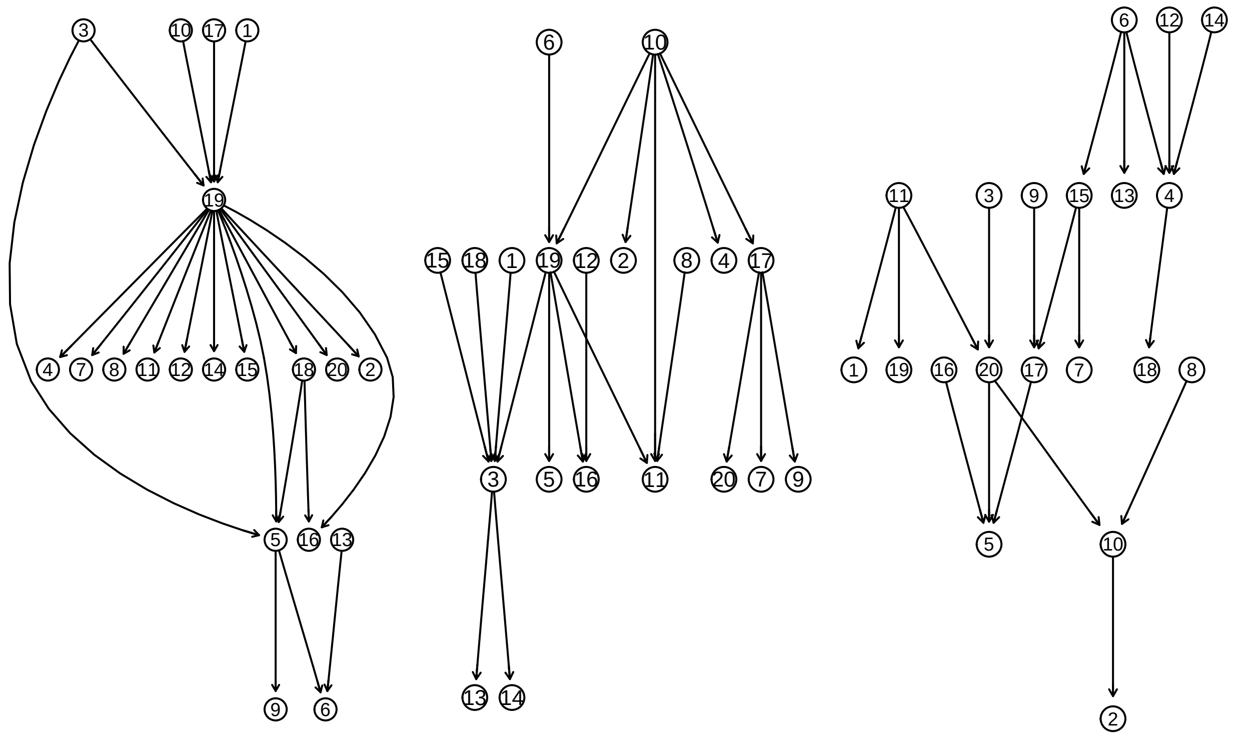
Bayesian network mixture model clustering

Generating networks and data

In this example we will consider 3 Bayesian networks with n=20 nodes each. We first generate 3 random power-law networks with an average neighbourhood of size $d = 2$ (average sum of number of parents and children of a node). We will also need to generate conditional probability tables for each node and data (using functions from generatebinaryBN.R).

```r
library(BiDAG)
library(pcalg)
library(RBGL)
library(clue)
source('../generatebinaryBN.R')

#number of nodes in BNs
n<-20

ss1<-500 #number of samples in the first cluster
ss2<-400 #number of samples in the second cluster
ss3<-300 #number of samples in the third cluster
ss<-ss1+ss2+ss3 #total number of samples

#set seed for reproducibility
sseed<-111

#generate 3 power-law networks each with 20 nodes
BNs<-list()
BNs[[1]]<-generatebinaryBN(n, ii=sseed,baseline=c(0.3,0.5))
BNs[[2]]<-generatebinaryBN(n, ii=sseed+1,baseline=c(0.3,0.5))
BNs[[3]]<-generatebinaryBN(n, ii=sseed+2,baseline=c(0.3,0.5))

#plot BNs
par(mfrow=c(1,3))
plot(BNs[[1]]$DAG, attrs=list(node=list(fontsize=10, fixedsize=TRUE,
                                        height=0.5,width=0.5)))
plot(BNs[[2]]$DAG, attrs=list(node=list(fontsize=10, fixedsize=TRUE,
                                        height=0.5,width=0.5)))
plot(BNs[[3]]$DAG, attrs=list(node=list(fontsize=8, fixedsize=TRUE,
                                        height=0.3,width=0.3)))
```



```
#generate data from 3 BNs, with sample sizes defined above
set.seed(sseed)
bindata1<-generatebinaryBN.data(n,BNs[[1]], ss1)
bindata2<-generatebinaryBN.data(n,BNs[[2]], ss2)
bindata3<-generatebinaryBN.data(n,BNs[[3]], ss3)
```

We join all data in a single table:

```
Datafull<-rbind(bindata1,bindata2,bindata3)
```

And store the true membership assignment:

```
#number of clusters
kclust<-3
#true number of clusters
kclusttrue<-3
#true membership
truememb<-c(rep(1,ss1),rep(2,ss2),rep(3,ss3))
#true cluster labels
truelabels<-list()
for (i in 1:kclusttrue) {
  truelabels[[i]]<-which(truememb==i)
}
```

Bayesian network mixture model clustering

Now we will apply the EM algorithm to cluster the mixture model. For this we need to initialize several values:

```
#prior pseudo counts
chixi<-0.5
#define when EM converges
err<-1e-6
#edge penalization factor
edgepfy<-2
#define different seeds to run EM
EMseeds<-c(101,102,103,104,105)
#number of EM attempts to ensure highest likelihood
EMn<-length(EMseeds)
#clustering functions
source('../clusterfns.R')
#number of iterations in the internal cycle
nit.internal<-10
```

We run the EM in 2 cycles: in the external cycle we estimate cluster centers, which is computationally more expensive than estimating parameters, which we do in the internal cycle for nit.internal=10 iterations without re-estimating the center. We also run the whole EM scheme with different seeds to possilby reach solutions with higher likelihood. For each observation we initially randomly assign probabilities of belonging to each cluster. We use the EM algorithm to iteratively recalculate these probabilities and reassign cluster memberships.

```
#to store accuracy of cluster centers
centerprogress<-list()
#accuracy of assignments for different EM runs
assignprogress<-list()
#to store scores against clusters for each EM run
scoresprogress<-list()
#to store cluster probabilities for all sample for each EM run
probs<-list()
#to store relabelling
relabs<-list()
#to store cluster centers
clustercenters<-list()
#to store scores against clusters
scoresagainstclusters<-matrix(ncol=kclust,nrow=ss)

for (s in 1:EMn) {
  diffy<-1
  cnt<-1
  assignprogress[[s]]<-list()
  assignprogress_local<-list()
  assignprogress_local$corrvec<-numeric()
  assignprogress_local$likel<-numeric()
  set.seed(EMseeds[s])
  print(paste("EM seed",EMseeds[s]))
  centers<-list()
  for (i in 1:kclusttrue) {
  centers[[i]]<-as.data.frame(matrix(ncol=3))
  clustercenters[[i]]<-matrix(rep(0,n^2),nrow=n)
```

```
}

#generate random assignment of belonging to each cluster for each sample
newallrelativeprobabs<-generatetriple(ss)
newclustermembership<-reassignsamples(newallrelativeprobabs,nrow(Datafull))

#learn how many samples were asssigned correctly
res<-checkmembership(kclust,kclusttrue,truelabels,newclustermembership)
print(paste("number of correctly assigned samples by random assignment:", res$ncorr,
            "of total", ss, "samples"))

#outer EM cycle, learning cluster centers
while (diffy>err) {
  allrelativeprobabs<-newallrelativeprobabs
  allprobprev<-newallrelativeprobabs
  coltots<-colSums(allrelativeprobabs) + chixi # add prior to clustersizes
  tauvec<-coltots/sum(coltots)

  #define cluster centers and assign probabilities
  for (k in 1:kclust) {

    #define score parameters
    scorepar<-scoreparameters(n,"bde",Datafull,
                              weightvector=allrelativeprobabs[,k],
                              bdepar=list(edgepf=edgepfy,chi=chixi))
    #find MAP DAG using iterative order MCMC
    maxfit<-iterativeMCMCsearch(n,scorepar,addspace=clustercenters[[k]],verbose=FALSE)
    #store it
    clustercenters[[k]]<-maxfit$max$DAG
  }

  #internal EM cycle, estimating parameters
  for (i in 1:nit.internal) {
    allrelativeprobabs<-newallrelativeprobabs
    coltots<-colSums(allrelativeprobabs) + chixi # add prior to clustersizes
    tauvec<-coltots/sum(coltots)
    for (k in 1:kclust) {
      scorepar<-scoreparameters(n,"bde",as.data.frame(Datafull),
              weightvector=allrelativeprobabs[,k], bdepar=list(edgepf=edgepfy,chi=chixi))
      scoresagainstclusters[,k]<-BiDAG::scoreagainstDAG(n,scorepar,clustercenters[[k]])
    }
    newallrelativeprobabsnotau<-allrelativeprobs(scoresagainstclusters,nrow(Datafull))
    newallrelativeprobabs<-relativeprobswithtau(newallrelativeprobabsnotau,tauvec)
  }

  diffy<-sum((allprobprev-newallrelativeprobabs)^2)
  newclustermembership<-reassignsamples(newallrelativeprobabs,nrow(Datafull))
  res<-checkmembership(kclust,kclusttrue=3,truelabels,newclustermembership)
  assignprogress_local$corrvec[cnt]<-res$ncorr
  assignprogress_local$likel[cnt]<-calcloglike(scoresagainstclusters,tauvec)
  for (k in 1:3) {
    centers[[k]][cnt,]<-compareDAGs(adjacency2dag(clustercenters[[k]]),
                                    BNs[[res$relabel[k]]]$DAG)
    }
    cnt<-cnt+1
  }
  print(paste("EM converged after",cnt-1,"iterations"))
  print(paste("number of correctly assigned samples:",
              res$ncorr, "of", ss))
  #store number of correct assignments and likelihood
  assignprogress[[s]]<-assignprogress_local
  #store center progress
  centerprogress[[s]]<-centers
  scoresprogress[[s]]<-scoresagainstclusters
  probs[[s]]<-newallrelativeprobabs
  relabs[[s]]<-res$relabel
}
```

```
## [1] "EM seed 101"
## [1] "number of correctly assigned samples by random assignment: 427 of total 1200 samples"
## [1] "EM converged after 16 iterations"
## [1] "number of correctly assigned samples: 1101 of 1200"
## [1] "EM seed 102"
## [1] "number of correctly assigned samples by random assignment: 410 of total 1200 samples"
## [1] "EM converged after 15 iterations"
## [1] "number of correctly assigned samples: 1101 of 1200"
## [1] "EM seed 103"
## [1] "number of correctly assigned samples by random assignment: 424 of total 1200 samples"
## [1] "EM converged after 7 iterations"
## [1] "number of correctly assigned samples: 1097 of 1200"
## [1] "EM seed 104"
## [1] "number of correctly assigned samples by random assignment: 417 of total 1200 samples"
## [1] "EM converged after 11 iterations"
## [1] "number of correctly assigned samples: 995 of 1200"
## [1] "EM seed 105"
## [1] "number of correctly assigned samples by random assignment: 435 of total 1200 samples"
## [1] "EM converged after 12 iterations"
## [1] "number of correctly assigned samples: 1101 of 1200"
```

Evaluation of results

We sort the results of different EM runs by their likelihood and choose the highest one.

```
#get the vector with loglikelihood reached at the EM convergence
likelihoodvector<-unlist(lapply(assignprogress,function(x)x$likel[length(x$likel)]))
assignvector<-unlist(lapply(assignprogress,function(x)x$corrvec[length(x$likel)]))

#order loglikelihood vector
so<-order(likelihoodvector,decreasing = TRUE)
summary.res<-cbind(likelihoodvector[so],round(assignvector[so]/ss,3),EMseeds[so])
colnames(summary.res)<-c("logL","acc","seed")
print(summary.res)
```

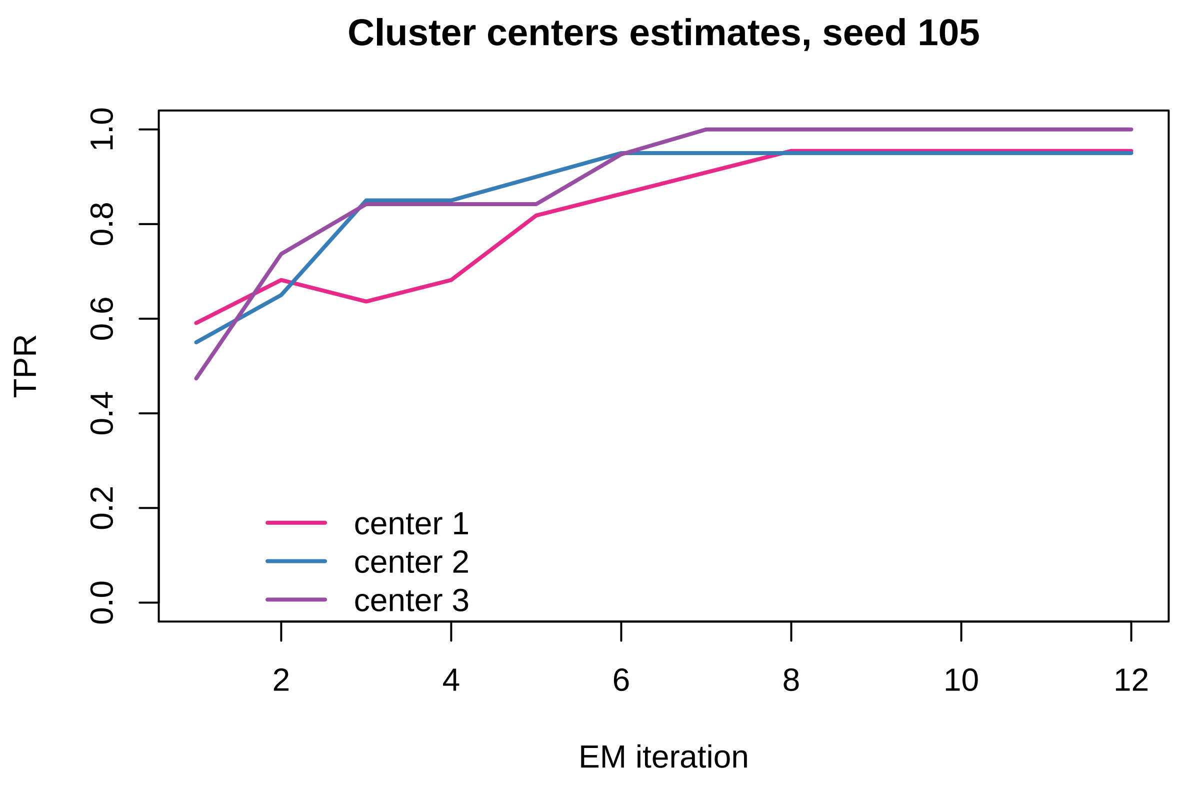
```
##            logL   acc seed
## [1,] -13395.42 0.918  105
## [2,] -13395.42 0.918  101
## [3,] -13395.42 0.918  102
## [4,] -13397.98 0.914  103
## [5,] -13522.52 0.829  104
```

```
#define best seed and corresponding accuracy
perccorr<-round(assignvector[so][1]/ss*100)
bestseed<-EMseeds[so][1]
percworst<-round(assignvector[so][EMn]/ss*100)
worstseed<-EMseeds[so][EMn]
```

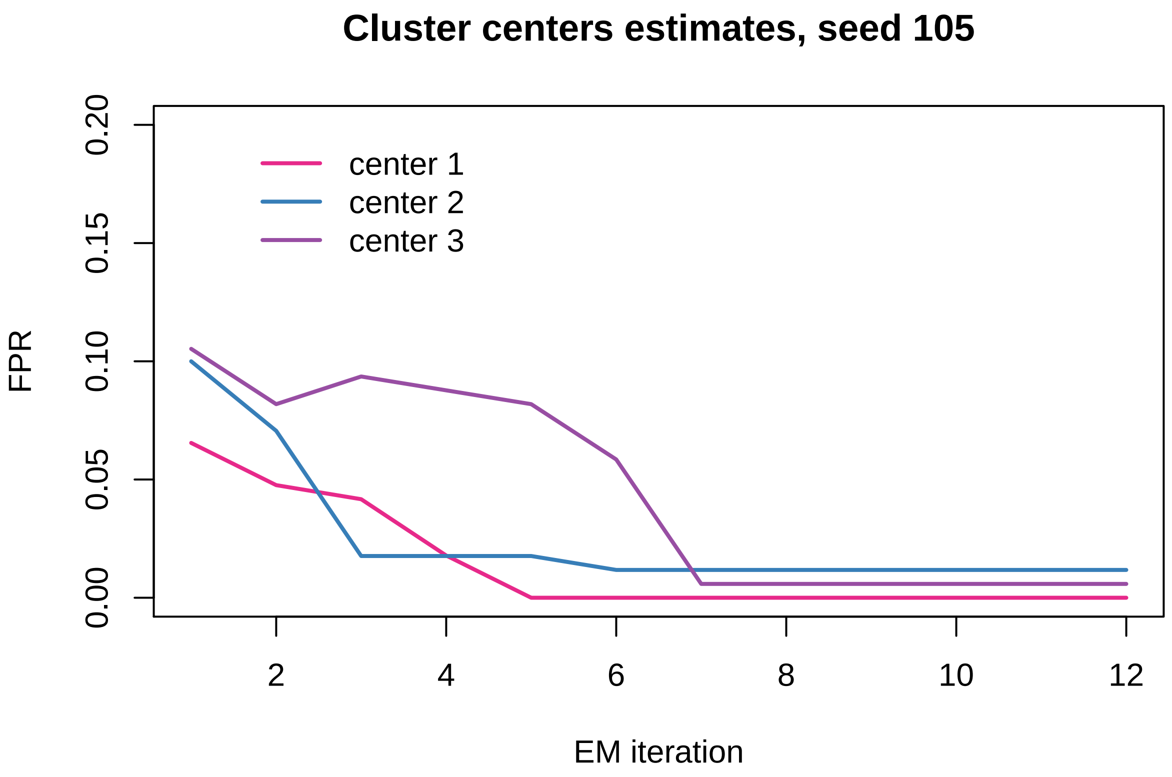
In our example 92% of observations were assigned to correct clusters for the EM run with highest likelihood which corresponds to seed 105. Other EM runs have similar or identical accuracy apart from the run with seed 104, which only reaches 83% accuracy.

We can also check how precise the estimates of the cluster centers are; we will plot the TPR (true positive rate) and the FPR (false positive rate) against the true cluster centers and over the outer EM iterations.

```
#we first calculate some summary figures which we will use for visualizing the results:
cols3<-c("#e7298a", "#377eb8", "#984ea3")
ne<-vector()
TPRb<-list()
FPRb<-list()
relab<-relabs[[which(EMseeds==bestseed)]]
for ( i in 1:kclusttrue) {
ne[i]<-sum(BNs[[relab[i]]]$adj)
TPRb[[i]]<-centerprogress[[which(EMseeds==bestseed)]][[i]][,2]/ne[i]
FPRb[[i]]<-centerprogress[[which(EMseeds==bestseed)]][[i]][,3]/((n*(n-1)/2)-ne[i])
}
plot(TPRb[[1]],type="l",col=cols3[1],ylim=c(0,1),ylab="TPR",xlab="EM iteration",
     lwd=2,main=paste("Cluster centers estimates, seed",bestseed))
lines(TPRb[[2]],type="l",col=cols3[2],lwd=2)
lines(TPRb[[3]],type="l",col=cols3[3],lwd=2)
legend(x=1.5,y=0.25,legend=c("center 1",
                             "center 2",
                             "center 3"),
       col=cols3,lty=c(1,1,1),lwd=c(2,2,2),bty="n")
```



```
plot(FPRb[[1]],type="l",col=cols3[1],ylim=c(0,0.2),ylab="FPR",xlab="EM iteration",lwd=2,
     main=paste("Cluster centers estimates, seed",bestseed))
lines(FPRb[[2]],type="l",col=cols3[2],lwd=2)
lines(FPRb[[3]],type="l",col=cols3[3],lwd=2)
legend(x=1.5,y=0.2,legend=c("center 1","center 2","center 3"),
       col=cols3,lty=c(1,1,1),lwd=c(2,2,2),bty="n")
```



We can see that center estimates are very precise for seed 105. TPR ranges between 95% and 100% and FPR falls to almost 0 for all three cluster centers.

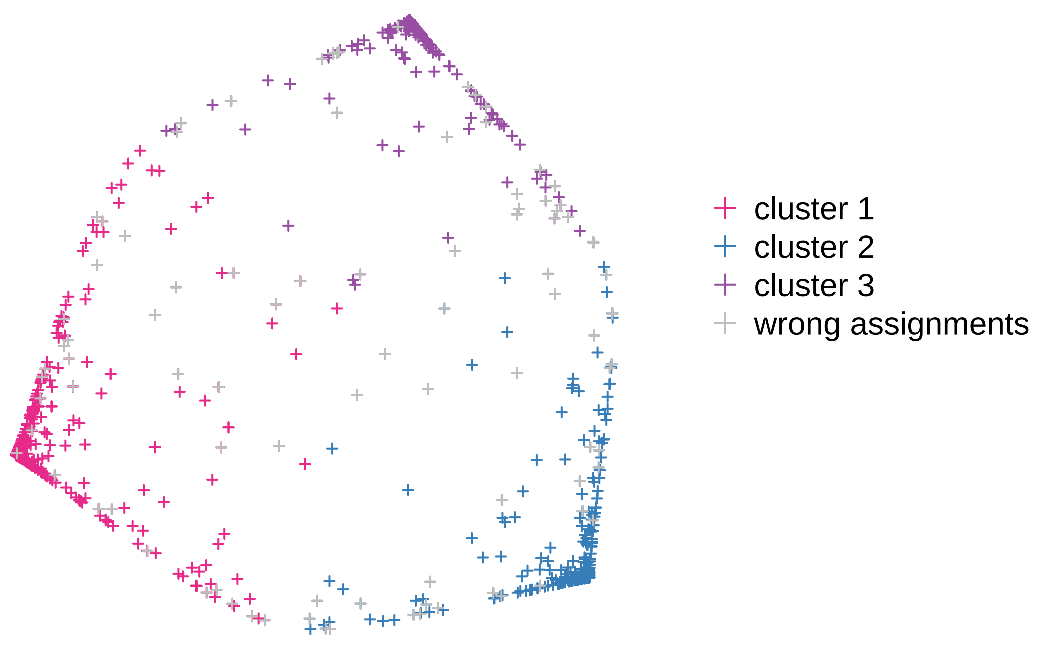
Cluster visualisation

We project the clusters on a 2D plane. In our example with just 3 clusters, this is straightforward because each sample is fully charactarized by the two probabilities of belonging to clusters 1 and 2 (the 3rd is just 1 minus two other probabilities). However for higher number of clusters we need to project higher dimension probability vectors on a 2D plane, which we perform with multidimensional scaling. For this reason we will show this approach here.

```
matsize<-nrow(Datafull)
divergy<-matrix(0,matsize,matsize)
for(k1 in 1:(matsize-1)){

  # take the log scores each sample against the cluster
  P<-scoresprogress[[which(EMseeds==bestseed)]][k1,]

  P<-exp(P-max(P)) # turn to scores
  P<-P/sum(P) # renormalise
  for(k2 in (k1+1):matsize){

    # take the log scores of graphs against another
    Q<-scoresprogress[[which(EMseeds==bestseed)]][k2,]

    Q<-exp(Q-max(Q)) # turn to scores
    Q<-Q/sum(Q) # renormalise
    M<-(P+Q)/2 # find the average

    # Jenson-Shannon divergence
    JS<- (P%*%log(P/M) + Q%*%log(Q/M))/2

    divergy[k1,k2]<-JS
    divergy[k2,k1]<-JS
  }
}
reducey<-cmdscale(divergy,k=2)
```

```
#plotting the result
bestmemb<-reassignsamples(probs[[which(EMseeds==bestseed)]],nrow(Datafull))
plot(reducey[which(bestmemb==relab[1]),1],reducey[which(bestmemb==relab[1]),2],
     col=cols3[1],xlim=c(-0.43,1),ylim=c(-0.45,0.20),pch=3,axes=F,bty="n",
     xlab="",ylab="",asp=1,main="Cluster assignments: 2D projection",cex=0.5)
lines(reducey[which(bestmemb==relab[2]),1],reducey[which(bestmemb==relab[2]),2],
      col=cols3[2],type="p",pch=3,cex=0.5)
lines(reducey[which(bestmemb==relab[3]),1],reducey[which(bestmemb==relab[3]),2],
      col=cols3[3],type="p",pch=3,cex=0.5)

#We can then plot points which were assigned erroneously:
errassign<-c(which(bestmemb==1)[!which(bestmemb==1)%in%which(truememb==relab[1])],
            which(bestmemb==2)[!which(bestmemb==2)%in%which(truememb==relab[2])],
            which(bestmemb==3)[!which(bestmemb==3)%in%which(truememb==relab[3])])
lines(reducey[errassign,1],reducey[errassign,2],col="grey",pch=3,axes=F,type="p",cex=0.5)
legend(0.45,0.25,legend=c("cluster 1","cluster 2","cluster 3","wrong assignments"),
       col=c(cols3,"grey"),pch=c(3,3,3,3),bty="n")
```

Cluster assignments: 2D projection


cluster 1
cluster 2
cluster 3
wrong assignments