

Exploratory analysis of one performance measure and more than one data set

R supplement of

“Exploratory and Inferential Analysis
of Benchmark Experiments”

Manuel J. A. Eugster

benchmark version 0.01

In Section 4.1 – Exploratory analysis – we introduce exploratory methods to analyse the result of benchmark experiments with more than one data set.

Requirements:

```
> Sweave('uci621.Rnw')
```

Extract misclassification results:

```
> miscl <- uci621[,,'Misclassification',]
```

Benchmark experiment

samples	algorithms	performances	data sets
250	6	1	21

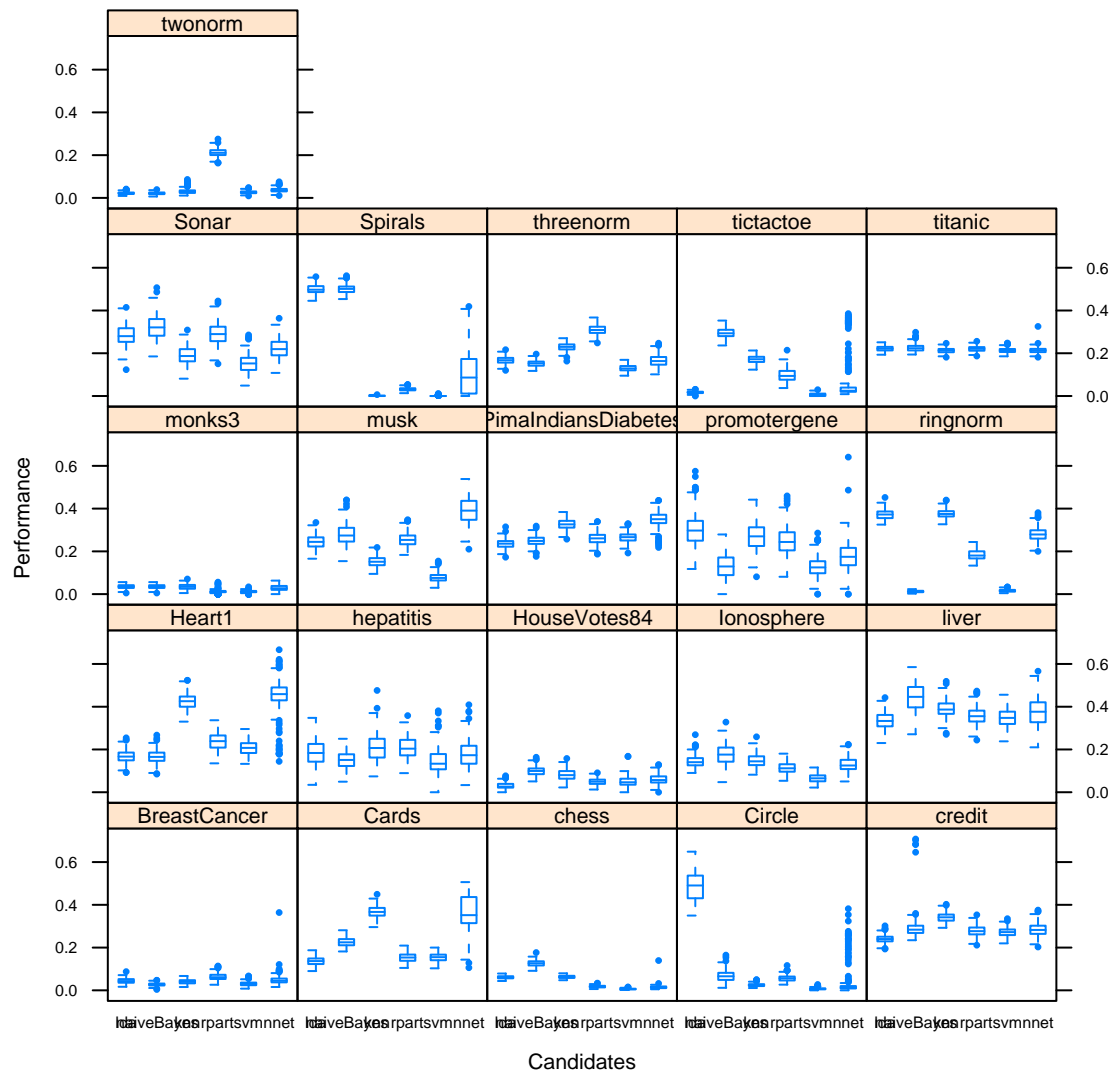
1 Raw results

Use Trellis displays to get a rough overview of the raw results, e.g., boxplots for each data set.

```

> print(bwplot(misc1,
+           par.settings=list(fontsize=list(text=8, points=8),
+                           box.dot=list(pch='|'),
+                           plot.symbol=list(pch=19, cex=0.3)))

```



2 Relations

Instead of analyzing the raw results we can infer an order on each data set and examine these compressed results.

2.1 The “lmer-path”

```

> ibea <- make.lmer.ibea()
> rels <- list()
> for ( i in 1:dim(misc1)[4] )

```

```

+ rels[[i]] <- ibea$relation(misc1[, , i], 0.05)
> names(rels) <- dimnames(misc1)$ds
> rels <- relation_ensemble(list=rels)

```

The order for each data set is:

```
> tsort(rels)
```

```
$BreastCancer
```

```
naiveBayes < svm < knn < lda < nnet < rpart
```

```
$Cards
```

```
lda < rpart - svm < naiveBayes < nnet < knn
```

```
$chess
```

```
svm < nnet - rpart < knn - lda < naiveBayes
```

```
$Circle
```

```
svm < knn < nnet < naiveBayes - rpart < lda
```

```
$credit
```

```
lda < rpart - svm < nnet < naiveBayes < knn
```

```
$Heart1
```

```
lda - naiveBayes < svm < rpart < knn < nnet
```

```
$hepatitis
```

```
naiveBayes - svm < lda - nnet < knn - rpart
```

```
$HouseVotes84
```

```
lda < rpart - svm < nnet < knn < naiveBayes
```

```
$Ionosphere
```

```
svm < rpart < nnet < knn - lda < naiveBayes
```

```
$liver
```

```
lda < rpart - svm < nnet < knn < naiveBayes
```

```
$monks3
```

```
rpart - svm < nnet < knn - lda - naiveBayes
```

```
$musk
```

```
svm < knn < lda < rpart < naiveBayes < nnet
```

```
$PimaIndiansDiabetes
```

```
lda < naiveBayes < rpart - svm < knn < nnet
```

```
$promotergene
```

```
naiveBayes - svm < nnet < rpart < knn < lda
```

```
$ringnorm
```

```
naiveBayes - svm < rpart < nnet < knn - lda
```

```
$Sonar
```

```
svm < knn < nnet < lda - rpart < naiveBayes
```

```
$Spirals
knn - svm < rpart < nnet < lda - naiveBayes
```

```
$threernorm
svm < naiveBayes < lda - nnet < knn < rpart
```

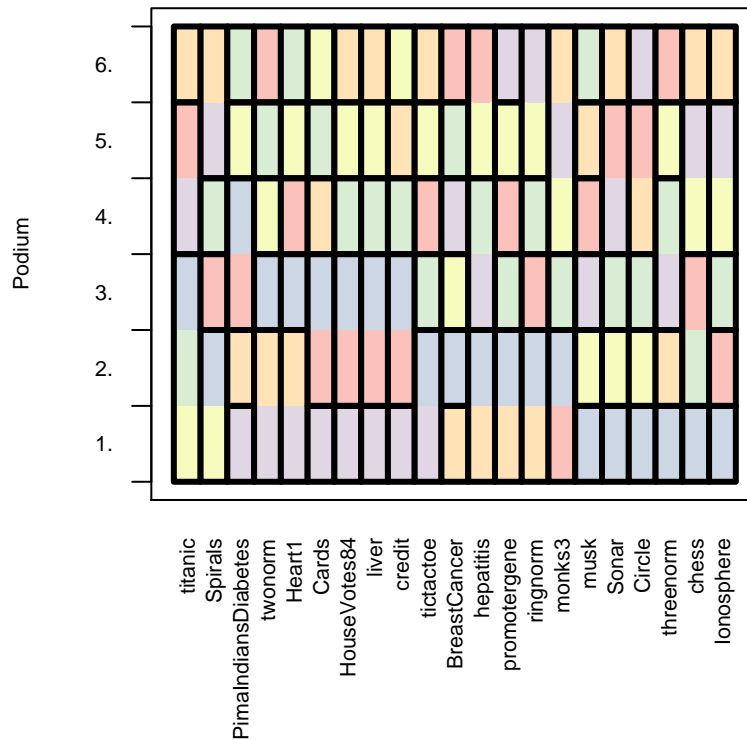
```
$tictactoe
lda - svm < nnet < rpart < knn < naiveBayes
```

```
$titanic
knn - nnet - svm < lda - rpart < naiveBayes
```

```
$twonorm
lda - naiveBayes < svm < knn < nnet < rpart
```

A better overview provides the benchmark experiment summary plot. The basic version hierarchically orders the data sets according to the algorithms on each podium place.

```
> par(mar=c(7, 4, 0, 0)+0.1, ps=8)
> bsplot(rels, col=sixcols.light)
```



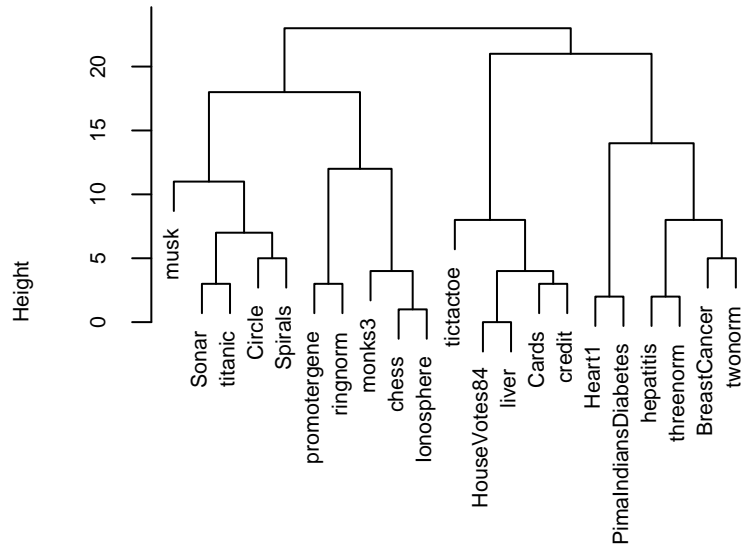
A more sophisticated order is based on a distance measure between the data sets by means of the order relations.

```
> dist <- relation_syndiff_matrix(rels)
```

We cluster the distance matrix using hierarchical clustering and use the thereby determined data set order as order for the summary plot.

```
> hc <- hclust(dist, method='complete')
```

```
> par(mar=c(0, 4, 0, 0)+0.1, ps=8)
> plot(hc, main="")
```



The data set order is

```
> hc$labels[hc$order]
```

```
[1] "musk"           "Sonar"          "titanic"
[4] "Circle"        "Spirals"        "promotergene"
[7] "ringnorm"      "monks3"         "chess"
[10] "Ionosphere"    "tictactoe"      "HouseVotes84"
[13] "liver"         "Cards"          "credit"
[16] "Heart1"        "PimaIndiansDiabetes" "hepatitis"
[19] "threenorm"    "BreastCancer"  "twonorm"
```

Additionally we can display summary performance measure for each algorithm on each data set.

```
> means <- apply(misc1, 'ds',
+               function(ds) {
+                 apply(ds, 'alg', mean, na.rm=TRUE)
+               })
```

	ds						
	BreastCancer	Cards	chess	Circle	credit	Heart1	
lda	0.04442405	0.1368634	0.060967903	0.490231592	0.2403389	0.1676137	
naiveBayes	0.02699879	0.2251562	0.126451762	0.066694509	0.2931273	0.1668039	
knn	0.04005126	0.3676005	0.062911243	0.024578024	0.3405488	0.4247871	
rpart	0.06404871	0.1527427	0.017341141	0.056260600	0.2780170	0.2367657	
svm	0.03101182	0.1533420	0.006210742	0.007442649	0.2723763	0.2072253	
nnet	0.04834667	0.3533370	0.015307582	0.038687253	0.2842046	0.4503889	
	ds						
	hepatitis	HouseVotes84	Ionosphere	liver	monks3	musk	
lda	0.1857398	0.02962477	0.14384933	0.3339383	0.03522340	0.24395942	
naiveBayes	0.1494783	0.09875895	0.17399269	0.4440583	0.03526528	0.27882026	
knn	0.2076030	0.08096881	0.14728306	0.3891794	0.03444507	0.15210080	
rpart	0.2091730	0.04957943	0.11372103	0.3572005	0.01163782	0.25542655	

```

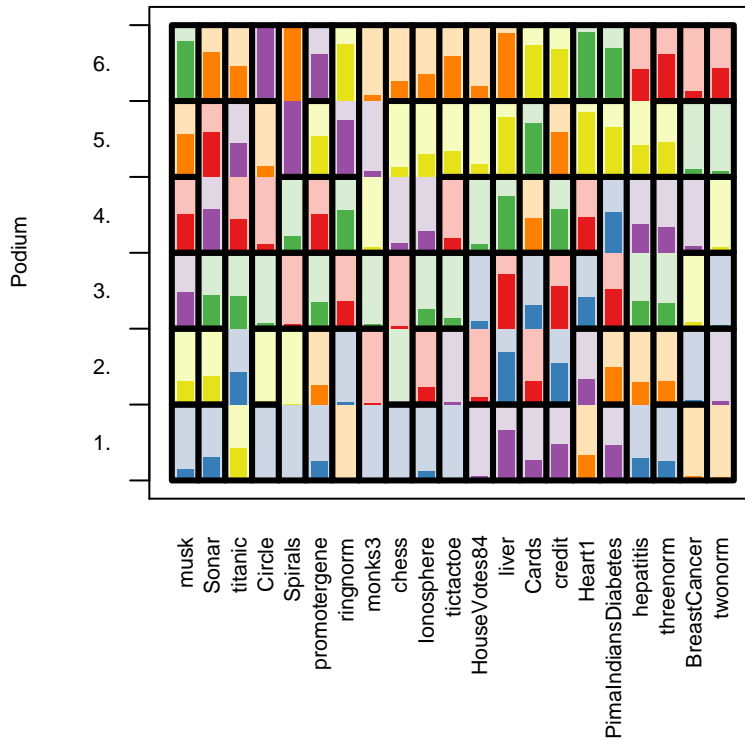
svm      0.1461527  0.04959985 0.06500161 0.34717111 0.01095148 0.07752166
nnet     0.1791356  0.05887722 0.12979724 0.3753380  0.02932404 0.39149328
ds
  PimaIndiansDiabetes promotergene  ringnorm  Sonar  Spirals
lda      0.2346950  0.3037110 0.37252600 0.2846688 0.4992614645
naiveBayes 0.2494870  0.1311887 0.01214795 0.3225559 0.5005232396
knn      0.3262285  0.2675246 0.37518120 0.1897354 0.0009292269
rpart    0.2616278  0.2509919 0.18277867 0.2950106 0.0319588022
svm      0.2658915  0.1263359 0.01586678 0.1531498 0.0007345569
nnet     0.3496113  0.1771749 0.27932426 0.2215610 0.1108001724
ds
  threernorm  tictactoe  titanic  twonorm
lda      0.1678388 0.016518527 0.2216311 0.02233067
naiveBayes 0.1523101 0.294405950 0.2258738 0.02145629
knn      0.2294445 0.171104030 0.2130349 0.03334928
rpart    0.3098066 0.097929701 0.2208273 0.21170860
svm      0.1298029 0.006493132 0.2134454 0.02617291
nnet     0.1654714 0.069185581 0.2135701 0.03644518

```

```

> par(mar=c(7, 4, 0, 0)+0.1, ps=8)
> bsplot(rels, col=sixcols.light,
+        perf=means, perfcoll=sixcols,
+        datasets.order=hc$order)

```



The distance matrix also can be represented as graph. Due to the readability we encode the data sets as letters.

```

> ldist <- dist
> attr(ldist, 'Labels') <- LETTERS[1:21]
> rbind(attr(dist, 'Labels'), attr(ldist, 'Labels'))

```

```

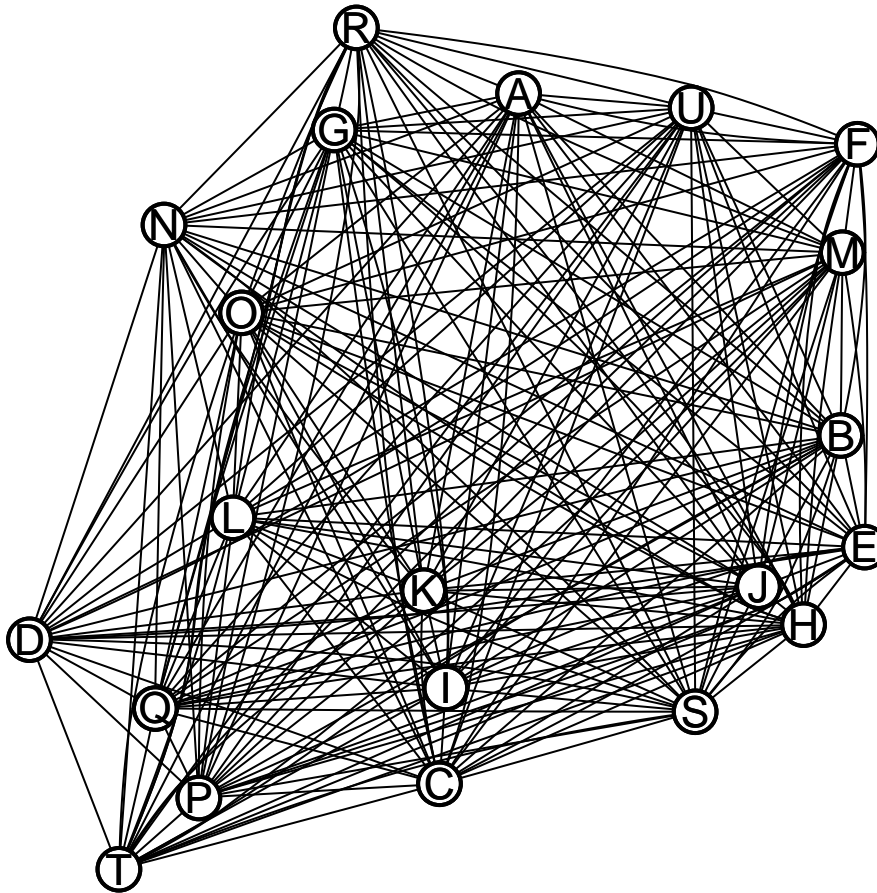
      [,1]      [,2]  [,3]  [,4]  [,5]  [,6]  [,7]
[1,] "BreastCancer" "Cards" "chess" "Circle" "credit" "Heart1" "hepatitis"
[2,] "A"          "B"    "C"   "D"   "E"   "F"   "G"
      [,8]      [,9]      [,10]  [,11]  [,12]  [,13]
[1,] "HouseVotes84" "Ionosphere" "liver" "monks3" "musk" "PimaIndiansDiabetes"
[2,] "H"          "I"          "J"    "K"    "L"    "M"
      [,14]      [,15]      [,16]  [,17]  [,18]      [,19]
[1,] "promotergene" "ringnorm" "Sonar" "Spirals" "threenorm" "tictactoe"
[2,] "N"          "O"          "P"    "Q"    "R"      "S"
      [,20]      [,21]
[1,] "titanic" "twonorm"
[2,] "T"          "U"

```

```

> par(lwd=2, ps=8)
> bsgraph(ldist)

```



We know that the graph is a complete graph, therefore we are not interested in all edges, which represent distance layers. In this case the distance layers are

```

> table(ldist)

```

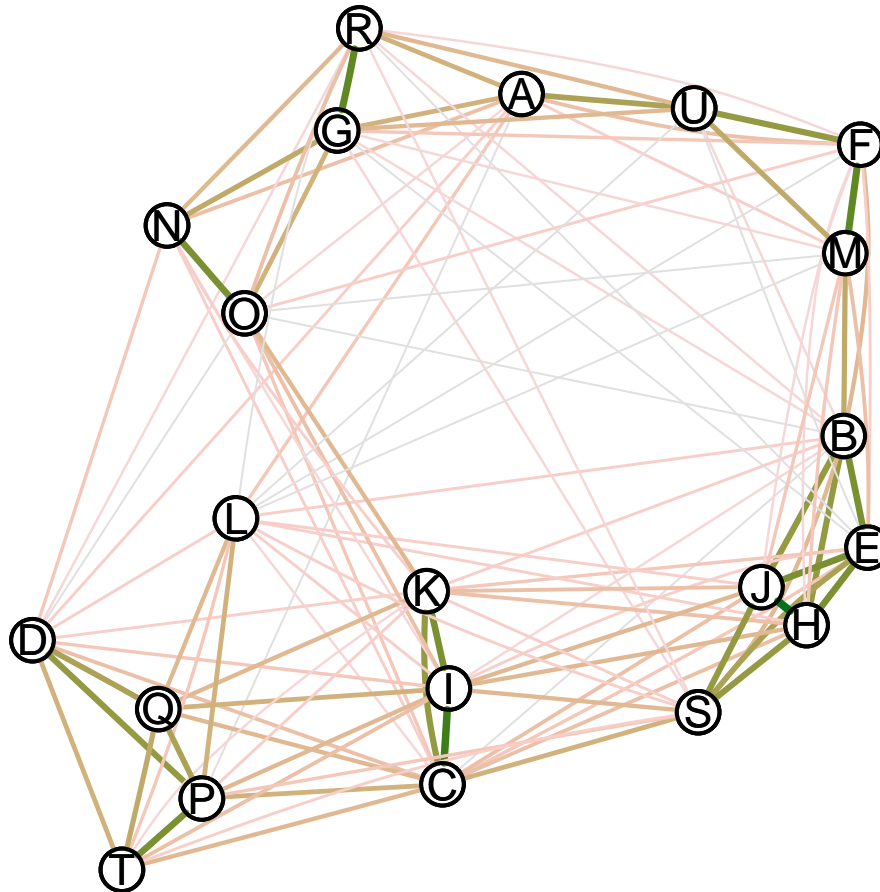
```

ldist
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 1  1  2  6  7  4  4  8 14 11 11 18 15 12 15 15 19 12  8  9  7  6  3  2

```

We are only interested in the first 14 layers, for example, and additionally set color and line width.

```
> layers <- 14
> require(vcd)
> edgecol <- terrain_hcl(layers,
+                       c=c(65,0), l=c(45,90), power=c(1/2, 1.5))
> edgelwd <- seq(4, 1, length.out=layers)
> par(lwd=2, ps=8)
> bsgraph(ldist,
+         edgecol=edgecol, edgelwd=edgelwd)
```

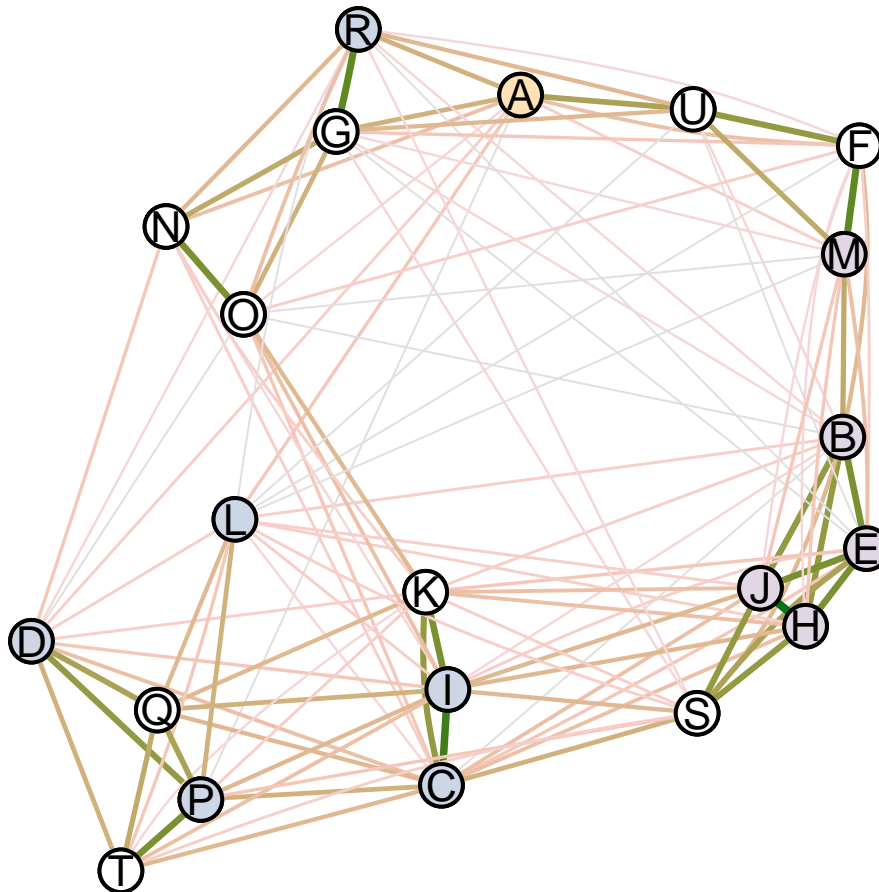


We can use the nodes to display information too, e.g., show the color of the winner algorithm if there is a unique winner on this specific data set.

```

> nodecol <- sapply(lapply(tsort(rels), as.ranking),
+                   function(w) {
+                     ifelse(sum(w==1) > 1, '#FFFFFF',
+                               sixcols.light[names(w)[w==1]])
+                   })
> names(nodecol) <- LETTERS[1:21]
> par(lwd=2, ps=8)
> bsgraph(ldist,
+         edgecol=edgecol, edgelwd=edgelwd,
+         nodefillcol=nodecol)

```



2.2 The “fr-path”

We can calculate the same things based on the Friedman-rank based analysis path.

```

> ibea <- make.fr.ibea()
> rels2 <- list()
> for ( i in 1:dim(misc1)[4] )
+   rels2[[i]] <- ibea$relation(misc1[, , i], 0.05)
> names(rels2) <- dimnames(misc1)$ds
> rels2 <- relation_ensemble(list=rels2)

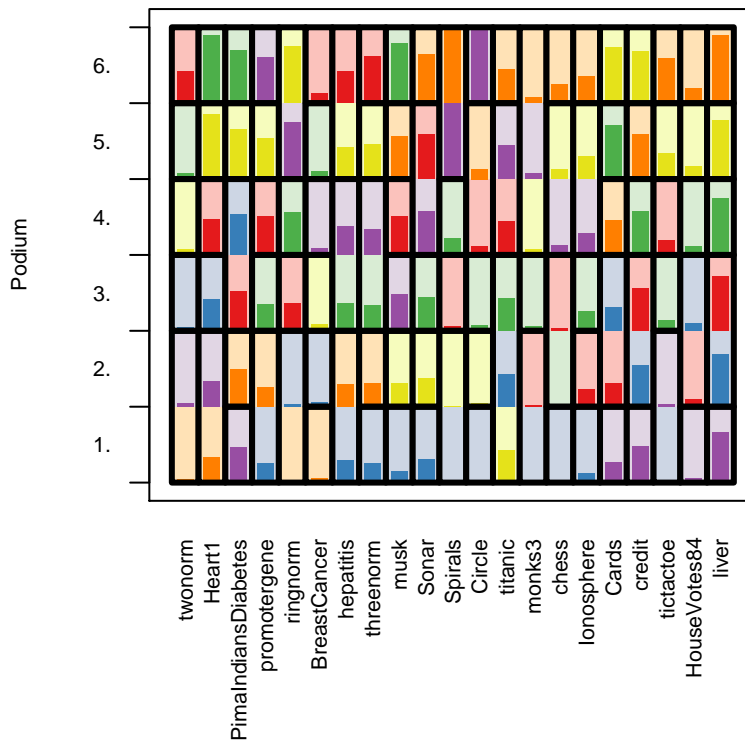
```

The result is similar, but not equally; see for example the summary plot.

```

> dist2 <- relation_symdiff_matrix(rels2)
> hc2 <- hclust(dist2, method='complete')
> par(mar=c(7, 4, 0, 0)+0.1, ps=8)
> bsplot(rels, col=sixcols.light,
+       perf=means, perfcoll=sixcols,
+       datasets.order=hc2$order)

```



References

M. J. A. Eugster, T. Hothorn, and F. Leisch. Exploratory and inferential analysis of benchmark experiments. Technical Report 30, Institut für Statistik, Ludwig-Maximilians-Universität München, Germany, 2008. URL <http://epub.ub.uni-muenchen.de/4134/>.