# Burrows Delta: Clustering techniques

*Stefan Evert*

*9 March 2016*

## Contents



# 1 Main evaluation results: clustering techniques and languages

## 1.1 Data sets & functions

Load relative frequencies and z-scores for the German, English and French data set. For technical reasons, the data structures store the transposed document-term matrices $\mathbf{F}^T$ and $\mathbf{Z}^T$

```
load("data/delta_corpus.rda")
## FreqDE, FreqEN, FreqFR ... text-word matrix with absolute and relative frequencies
## zDE, zEN, zFR          ... standardized (z-transformed) relative frequencies
## goldDE, goldEN, goldFR ... gold standard labels (= author names)
```

- $\mathbf{F}^T$ is available under the names `FreqDE$S`, `FreqEN$S` and `FreqFR$S`
- $\mathbf{Z}^T$ is available under the names `zDE`, `zEN` and `zFR`
- absolute frequencies $n_{D_j} \cdot f_i(D_j)$ can be found in `FreqDE$M`, `FreqEN$M`, `FreqFR$M`

In this section, evaluation results are usually visualized by plotting ARI against $n_w$ for 5 selected variants of $\Delta$ based on different metrics, produced by the function `ari.plot()`. Other parameters (feature scaling, normalization, clustering technique, etc.) are analyzed by comparing such plots, preferably in the form of animations (i.e. they are mapped to a time dimension). They can be passed as named parameters to `ari.plot()`.

```
## roughly logarithmic steps from 10 to 20000
n.vals <- c(10,15,20,25,32,40,50,65,80,100,125,150,200,250,320,400,500,650,800,1000,
            1250,1500,2000,2500,3200,4000,5000,6500,8000,10000,12500,15000,20000)
draw.grid <- function () { # corresponding grid for plot region
  abline(h=seq(0, 100, 10), col="grey60")
  abline(v=c(10,20,50,100,200,500,1000,2000,5000,10000,20000), col="grey60")
}
```

Overview of function parameters:

- `M`: row matrix of feature vectors

- `gold`: vector of gold standard categories (e.g. authors)
- `normalize`, `norm.p`: normalization applied to feature vectors (or `NA` for none; see `delta.dist()`)
- `clust.method`: clustering method (see `pam.cluster()`)
- `n.clusters`: desired number of clusters (specify range for automatic selection based on silhouette width)
- `transform`: transformation function applied to complete feature matrix
- `skip`: number of mfw to skip (i.e. drop first columns from `M`)

```
ari.plot <- function (M, gold, normalize=NA, norm.p=2, clust.method="pam",
                      transform=NULL, skip=0, n.clusters=NULL,
                      n=n.vals, grid=draw.grid, main="") {
  if (skip > 0) M <- M[, -(1:skip)]
  if (ncol(M) < max(n.vals)) stop("feature matrix M doesn't have enough columns")
  max.n <-  # don't reduce matrix by a small amount
  if (ncol(M) > max(n.vals) + 5000) M <- M[, 1:max(n.vals)]
  if (!is.null(transform)) M <- transform(M)
  plot(1, 1, type="n", log="x", xlim=range(n.vals), ylim=c(0,100),
       xlab="# features", ylab="adjusted Rand index (%)", main=main,
       xaxs="i", yaxs="i", las=3, xaxp=c(range(n.vals), 3))
  if (!is.null(draw.grid)) draw.grid()
  for (R in list(list(method="manhattan", p=1, col=1),
                 list(method="cosine", p=2, col=3),
                 list(method="euclidean", p=2, col=2),
                 list(method="minkowski", p=0.5, col=4),
                 list(method="minkowski", p=4, col=6))) {
    res <- evaluate(M, gold, n=n.vals, method=R$method, p=R$p, clusters=n.clusters,
                    clust.method=clust.method, normalize=normalize, norm.p=norm.p)
    lines(n.vals, res$adj.rand, lwd=3, col=R$col)
  }
  legend("bottomright", inset=.02, bg="white", lwd=3, col=c(3,4,1,2,6),
         legend=expression("Cosine Delta", L[1/2]*"-Delta", "Burrows "*(L[1])*" Delta",
                           "Quadratic "*(L[2])*" Delta", L[4]*"-Delta"))
}
```

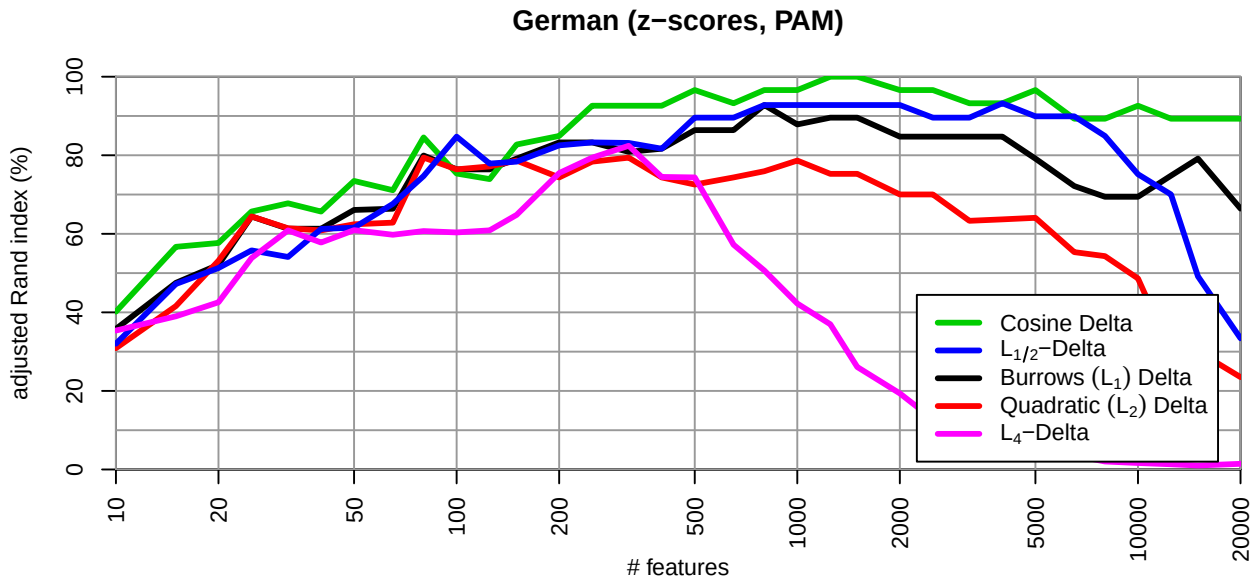
Callback functions for feature transformations with suitable defaults; `tern.binarize()` ternarizes the first $n$ mfw (default: `n=3000`) and binarizes the remaining features.

```
clamp <- function (x, min=-2, max=2) {
  pmax(pmin(x, max), min)
}
ternarize3 <- function (x, neutral.p=1/3, crossover=NULL) {
  ternarize(x, neutral.p=neutral.p, crossover=crossover)
}
tern.binarize <- function (x, n=3000) {
  if (ncol(x) > n) {
    cbind(ternarize3(x[, 1:n, drop=FALSE]), binarize(x[, -(1:n), drop=FALSE]))
  } else {
    ternarize3(x)
  }
}
```

## 1.2 German: PAM Clustering

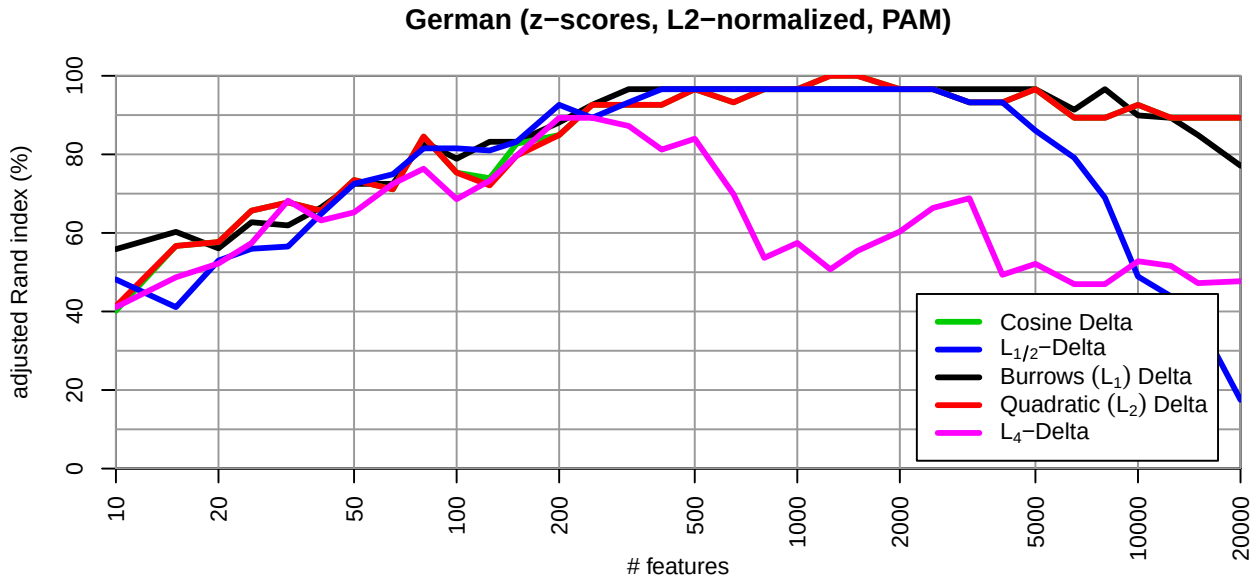
Standard approach using z-transformed relative frequencies (Burrows 2002).

```
ari.plot(zDE, goldDE, main="German (z-scores, PAM)")
```



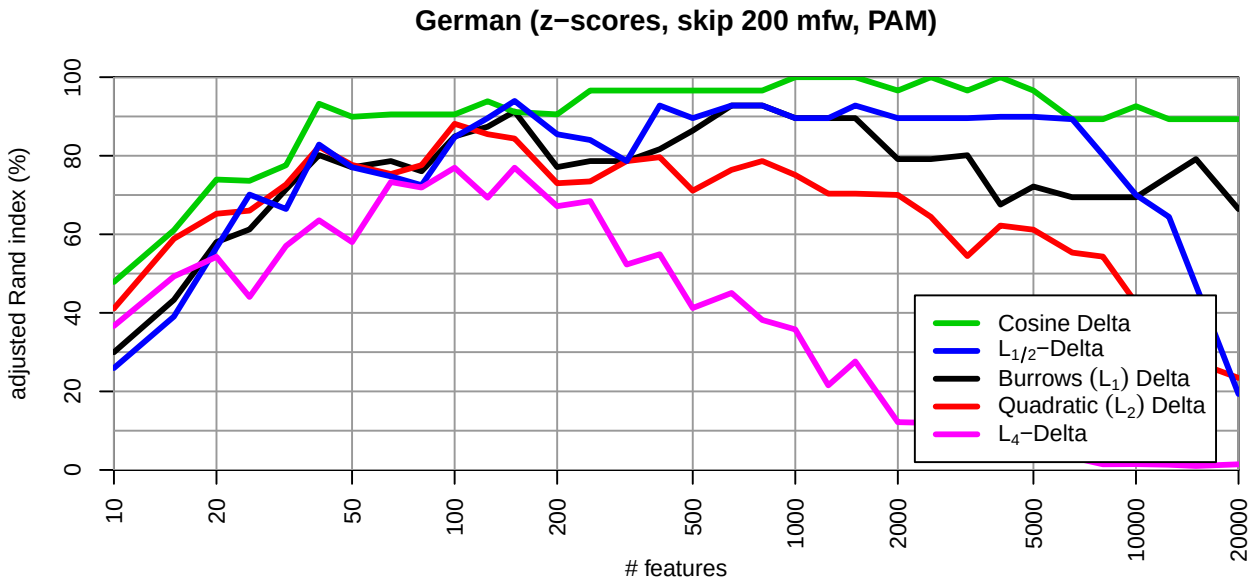

L2-normalization improves all Delta variants:

```
ari.plot(zDE, goldDE, normalize="euclidean", main="German (z-scores, L2-normalized, PAM)")
```



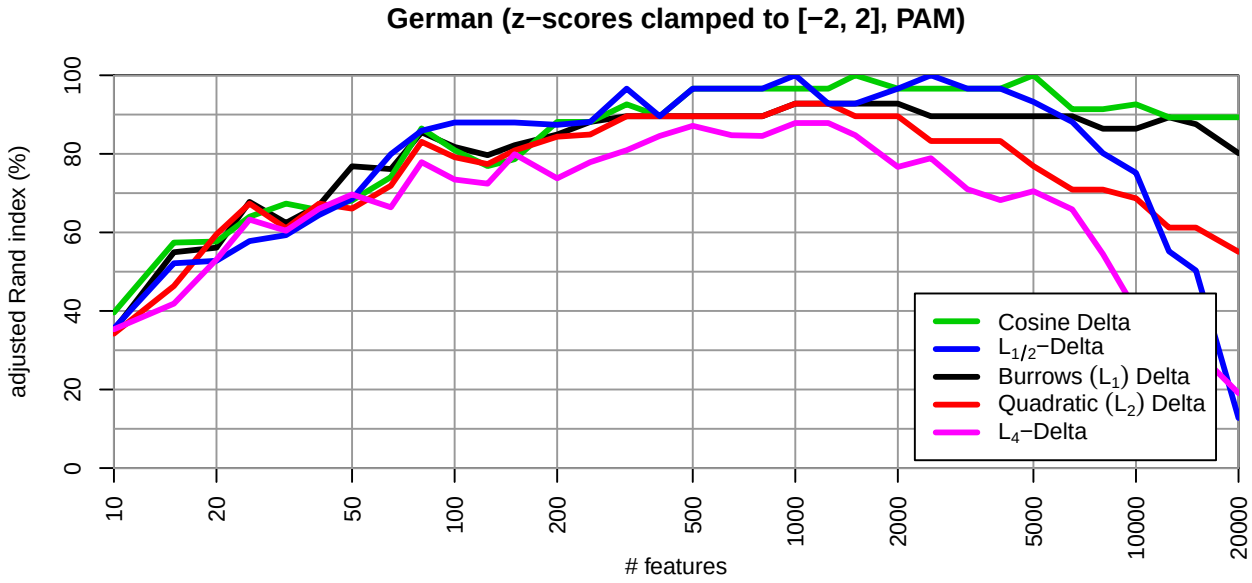

Skipping the top mfw is only beneficial when combined with normalization:

```
ari.plot(zDE, goldDE, skip=200, main="German (z-scores, skip 200 mfw, PAM)")
```



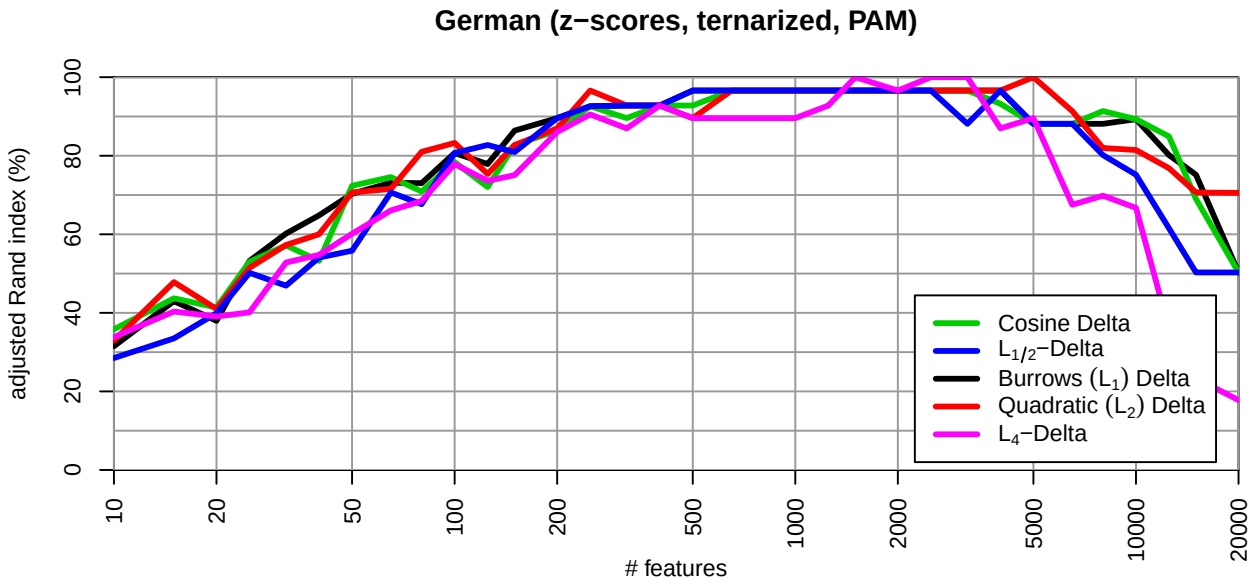

Clipping outliers improves performance and robustness, but somewhat less well than normalization:

```
ari.plot(zDE, goldDE, transform=clamp, main="German (z-scores clamped to [-2, 2], PAM)")
```



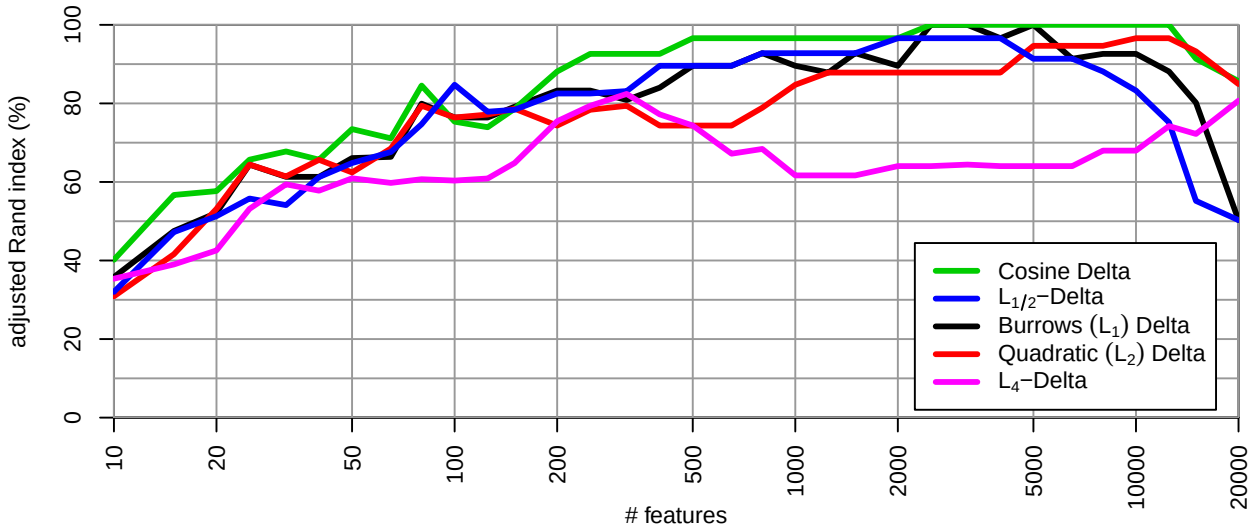

Ternarization is highly robust for $200 \leq n_w \leq 5000$, but falls off for longer feature vectors:

```
ari.plot(zDE, goldDE, transform=ternarize3, main="German (z-scores, ternarized, PAM)")
```



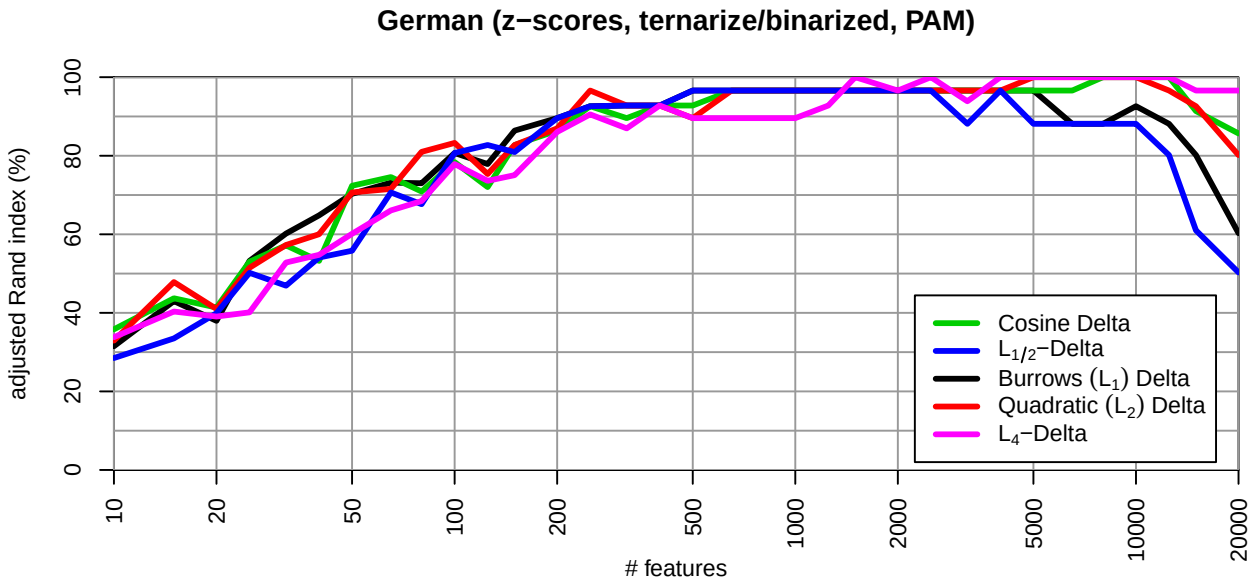

Binarization seems to work well for very long feature vectors. Full binarizaiton is highly robust at about 90% ARI, but a late crossover as shown here improves top results to perfect clustering (100%):

```
ari.plot(zDE, goldDE, transform=function (x) binarize(x, crossover=2000),
         main="German (z-scores, binarized after 2000 mfw, PAM)")
```



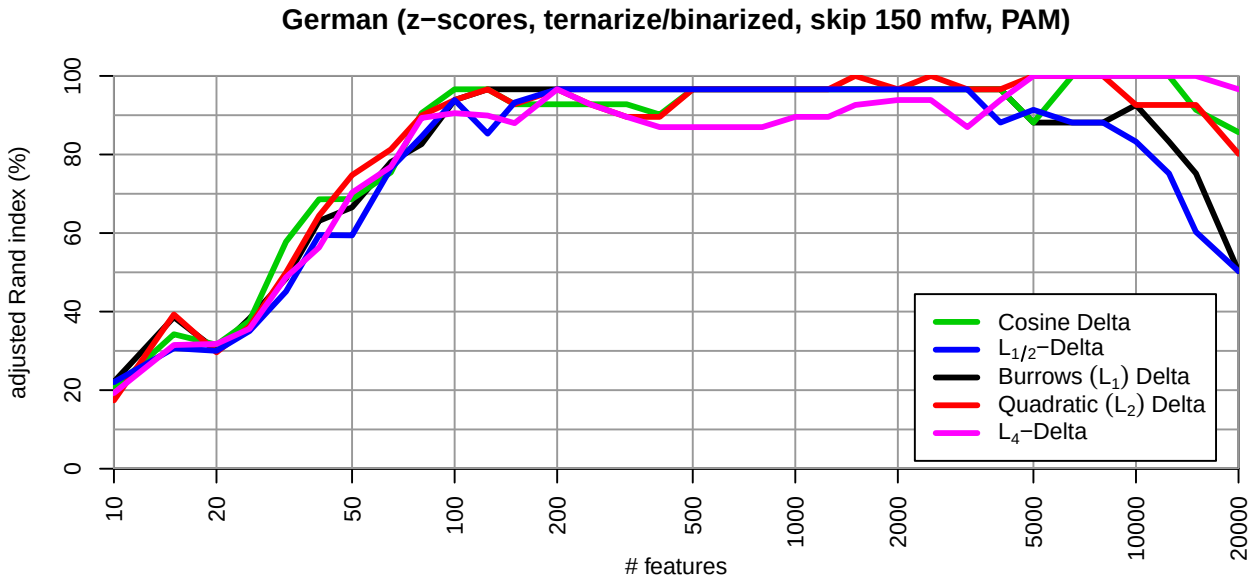

A combination of ternarization (up to $n_w = 3000$) with binarization of the remaining features gives both excellent clustering quality and robust performance. Part of the reason is certainly that the binarized features get lower relative weight especially for $\Delta_Q$ and $\Delta_4$. Hence this result should not be overinterpreted.

```
ari.plot(zDE, goldDE, transform=tern.binarize, main="German (z-scores, ternarize/binarized, PAM)")
```




Skipping a few mfw helps clustering reach good performance faster, even after as few as $n_w = 100$ features. It seems reasonable to skip between 100 and 200 mfw, but there is quite some fluctuation in the evaluation graphs, indicating a considerable amount of sampling variation.

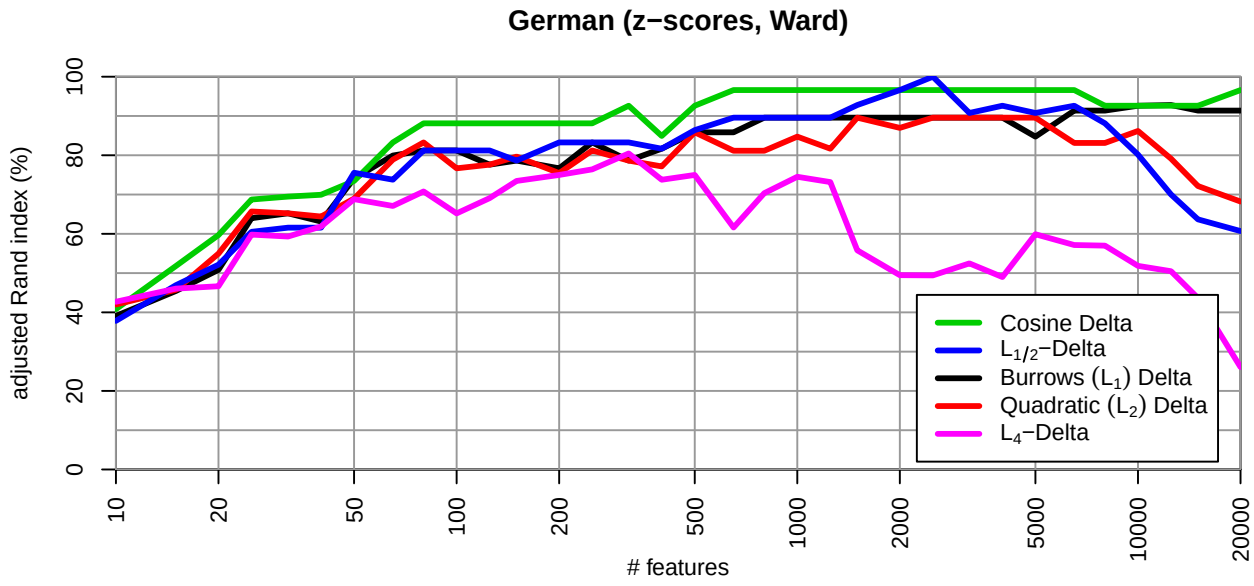
```
ari.plot(zDE, goldDE, skip=150, transform=tern.binarize,
         main="German (z-scores, ternarize/binarized, skip 150 mfw, PAM)")
```



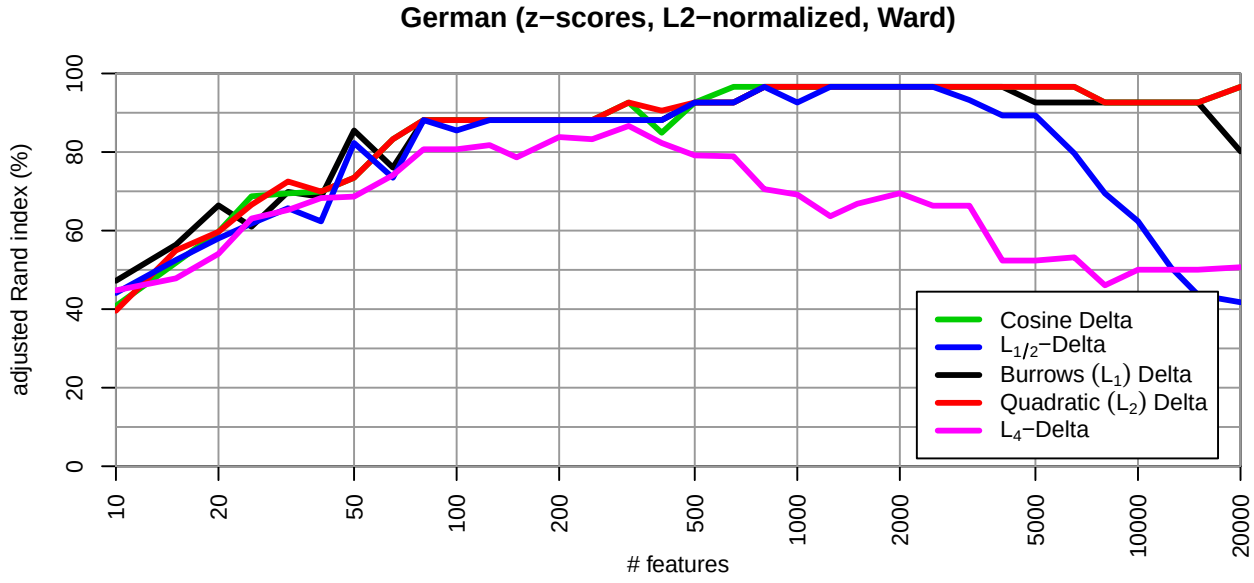

## 1.3 German: Ward Clustering

Replication of the German results using Ward clustering instead of PAM.

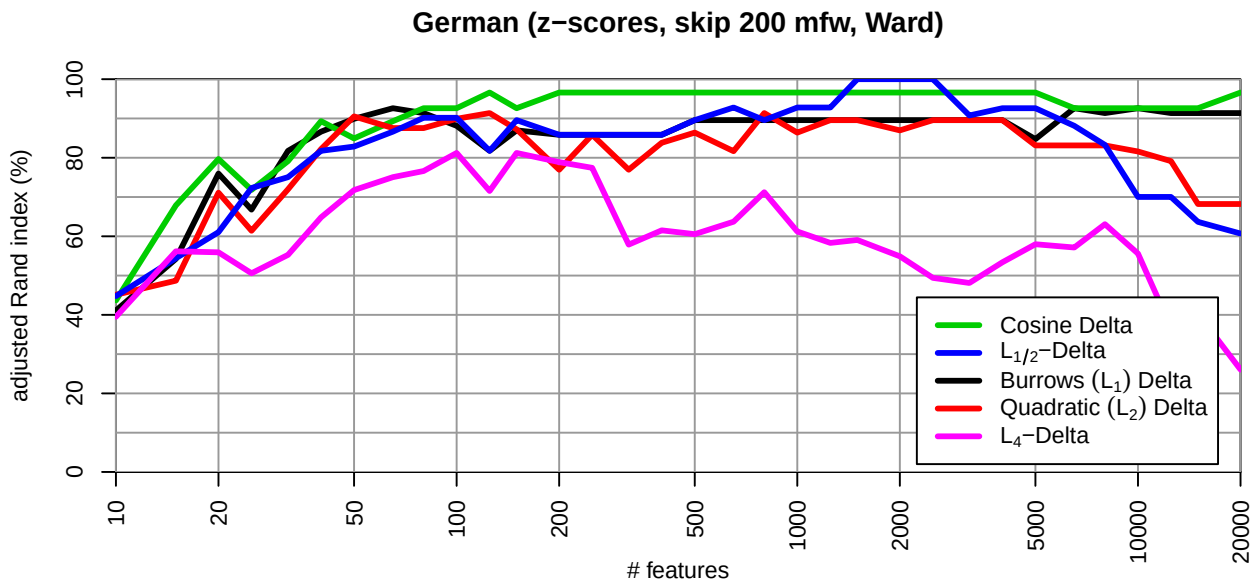
```
ari.plot(zDE, goldDE, clust.method="ward", main="German (z-scores, Ward)")
```



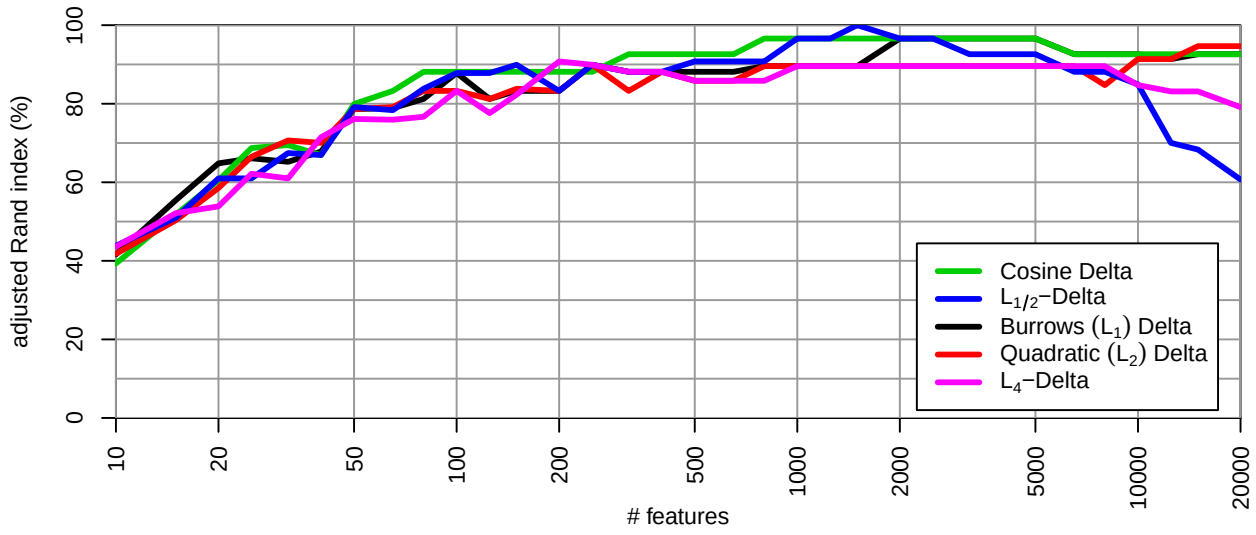

```
ari.plot(zDE, goldDE, clust.method="ward", normalize="euclidean",
         main="German (z-scores, L2-normalized, Ward)")
```



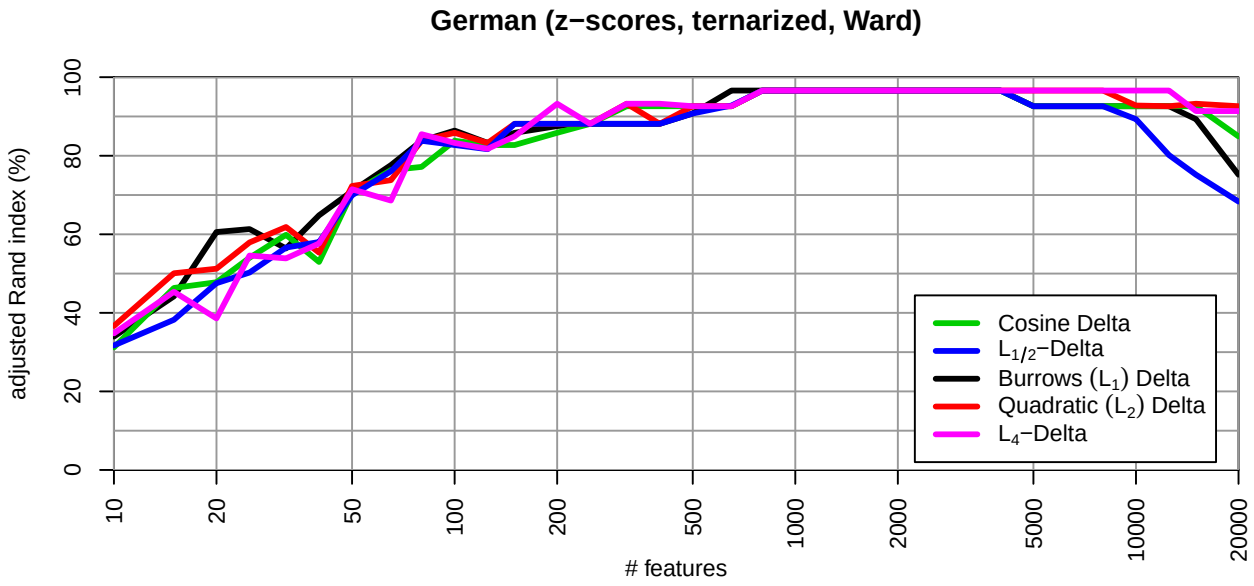

```
ari.plot(zDE, goldDE, clust.method="ward", skip=200,
         main="German (z-scores, skip 200 mfw, Ward)")
```



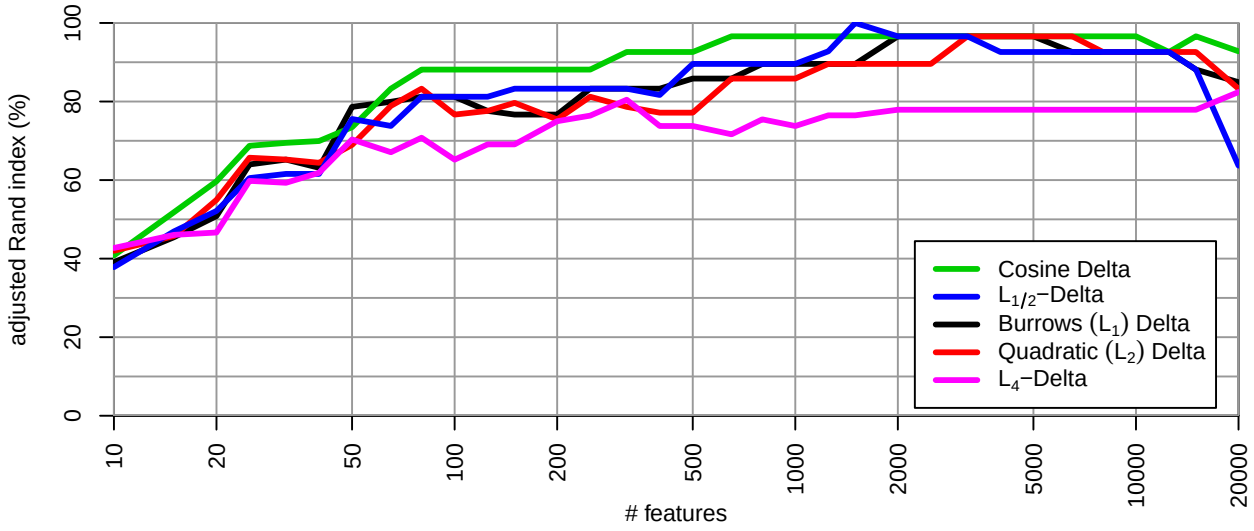

```
ari.plot(zDE, goldDE, clust.method="ward", transform=clamp,
         main="German (z-scores clamped to [-2, 2], Ward)")
```



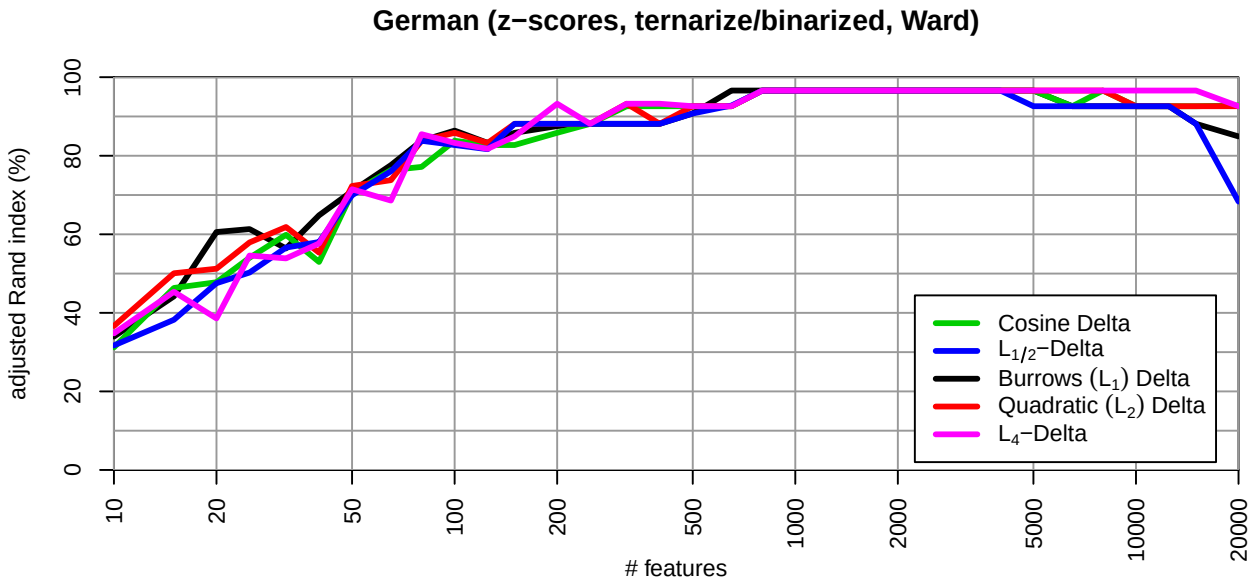

```
ari.plot(zDE, goldDE, clust.method="ward", transform=ternarize3,
         main="German (z-scores, ternarized, Ward)")
```



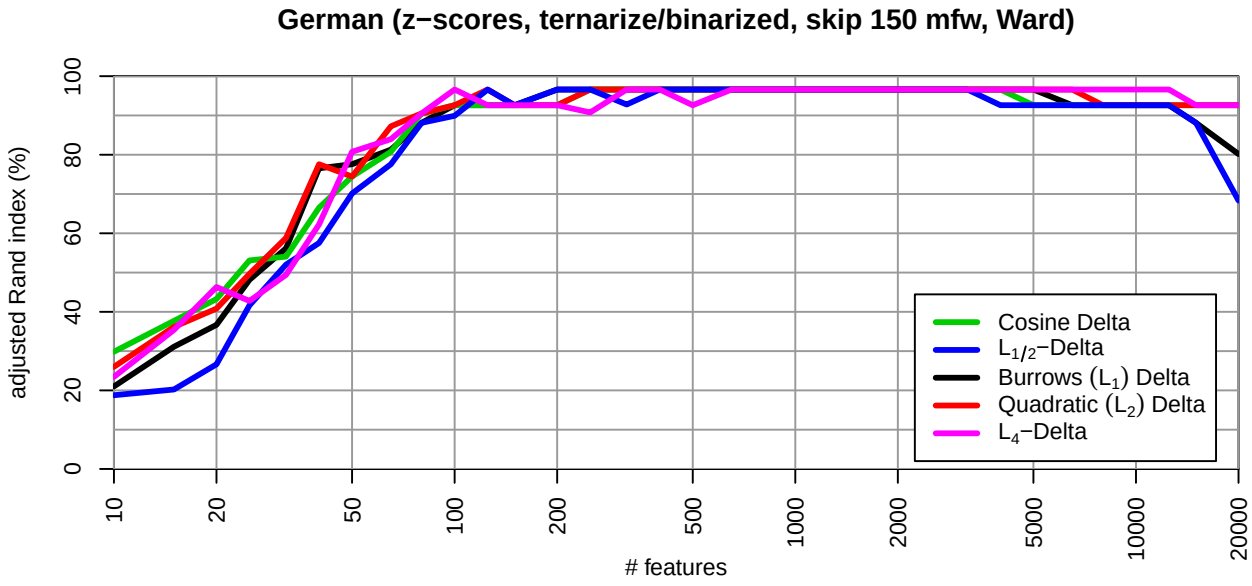

```
ari.plot(zDE, goldDE, clust.method="ward", transform=function (x) binarize(x, crossover=2000),
         main="German (z-scores, binarized after 2000 mfw, Ward)")
```




```
ari.plot(zDE, goldDE, clust.method="ward", transform=tern.binarize,
         main="German (z-scores, ternarize/binarized, Ward)")
```



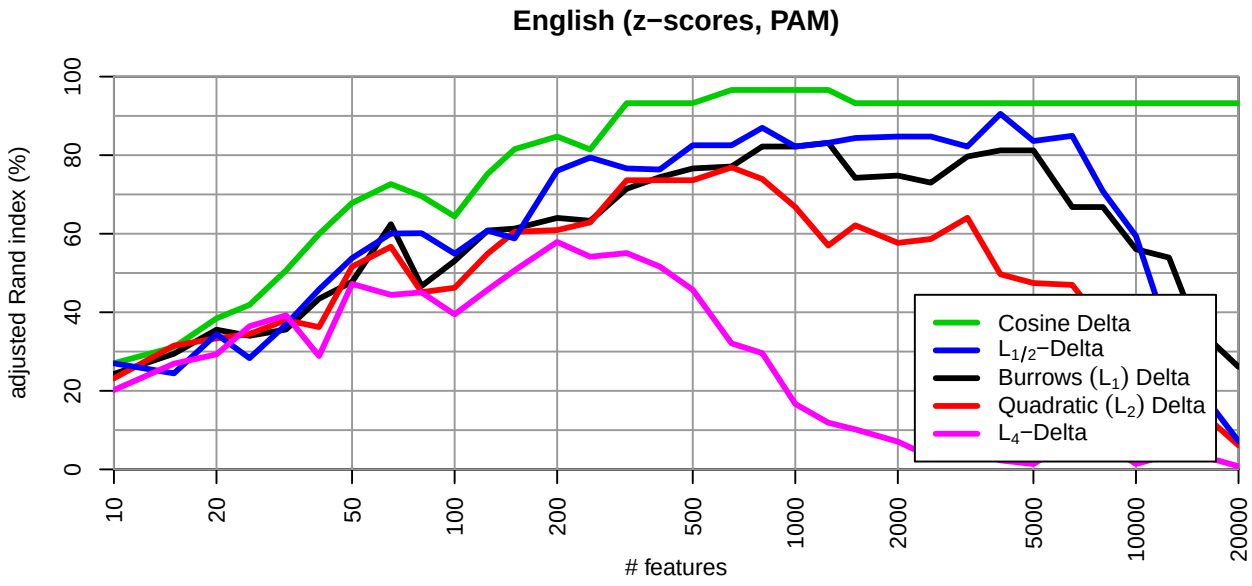

```
ari.plot(zDE, goldDE, clust.method="ward", skip=150, transform=tern.binarize,
         main="German (z-scores, ternarize/binarized, skip 150 mfw, Ward)")
```



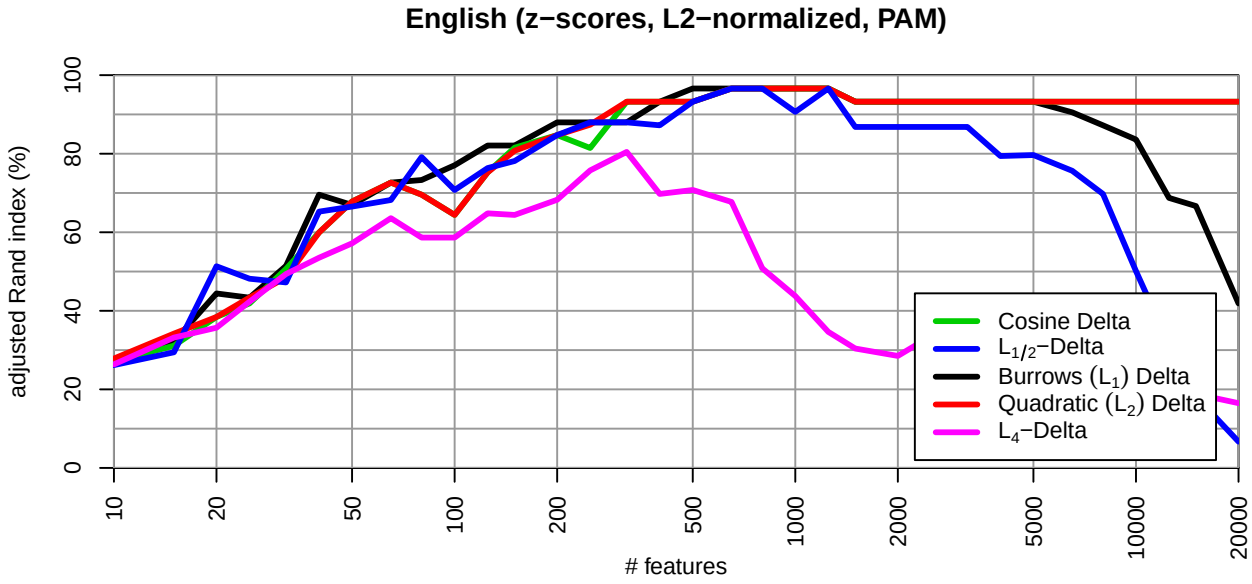

## 1.4 English: PAM Clustering

Replication of results on English corpus, using PAM clustering.

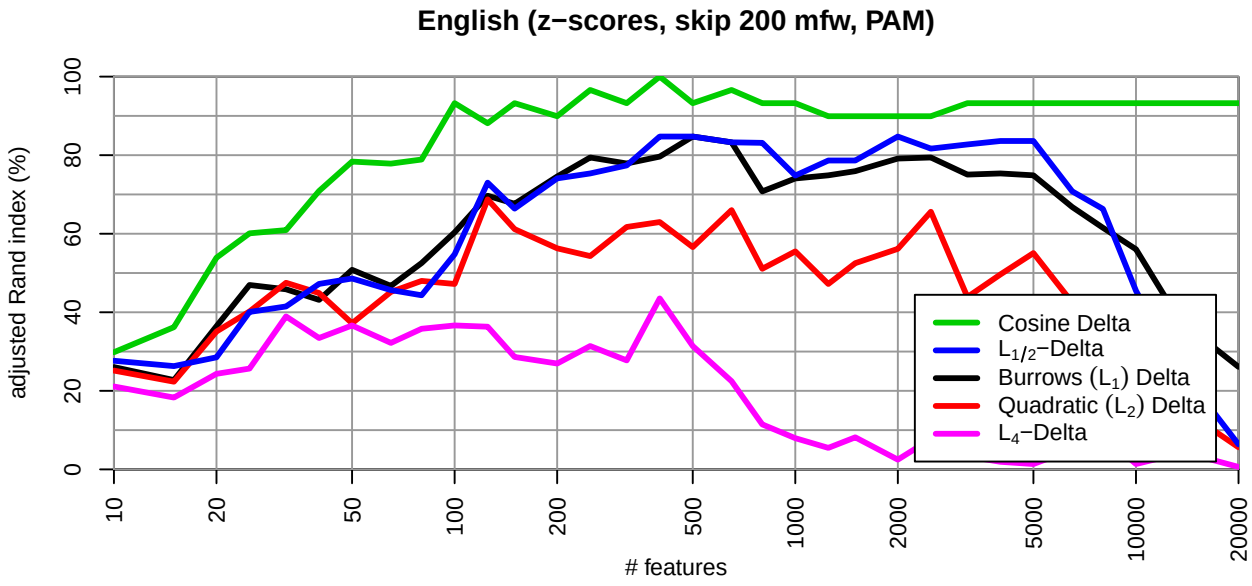
```
ari.plot(zEN, goldEN, main="English (z-scores, PAM)")
```



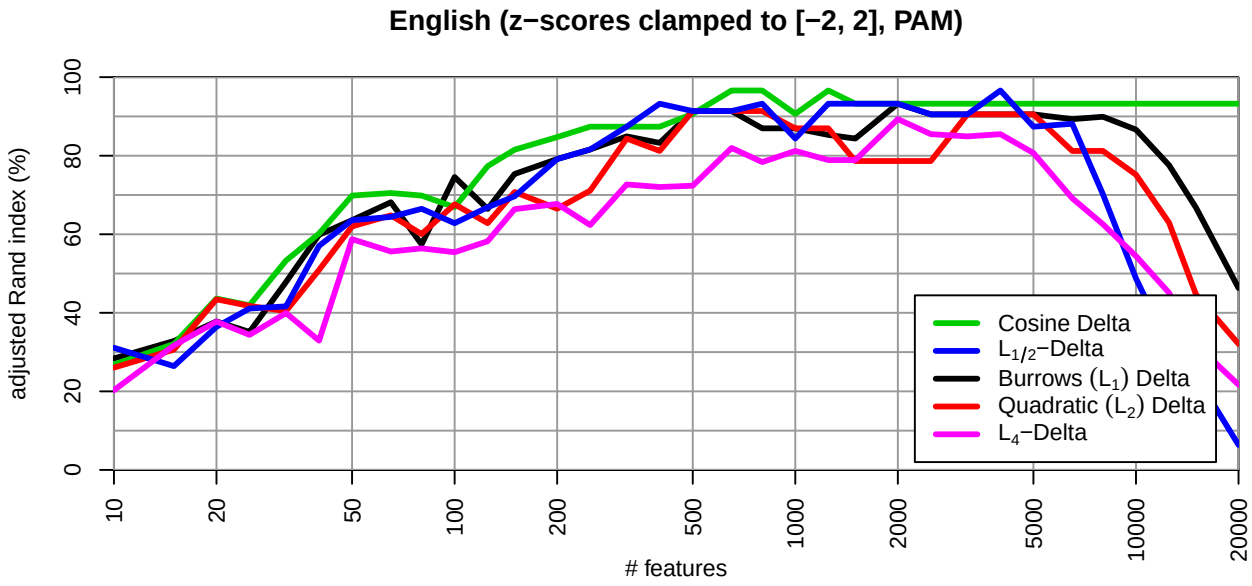

```
ari.plot(zEN, goldEN, normalize="euclidean", main="English (z-scores, L2-normalized, PAM)")
```



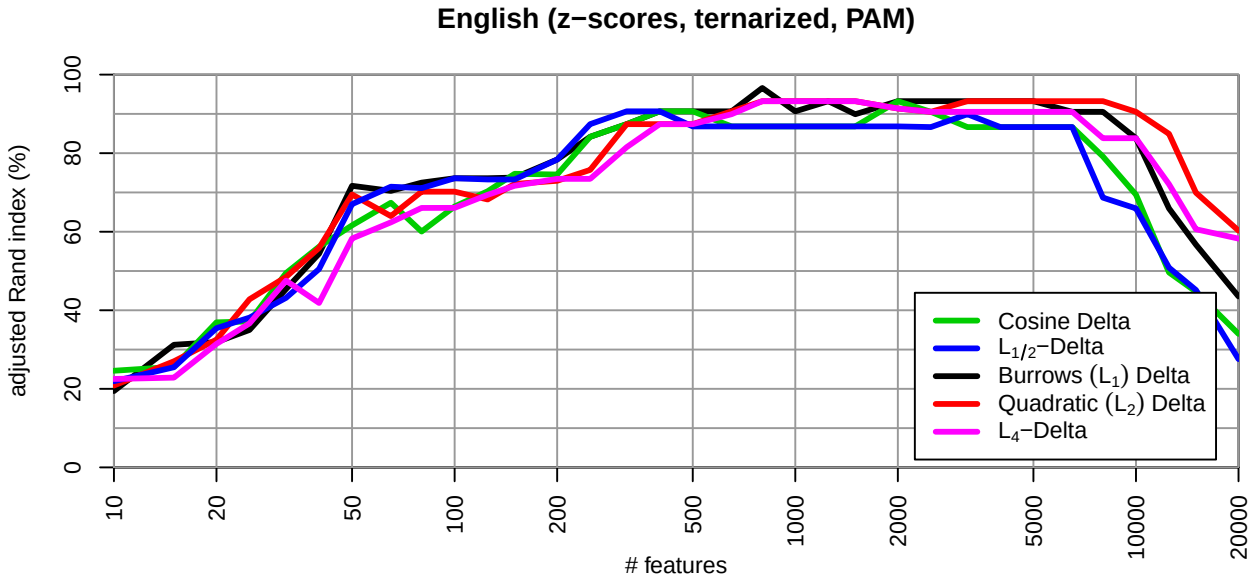

```
ari.plot(zEN, goldEN, skip=200, main="English (z-scores, skip 200 mfw, PAM)")
```



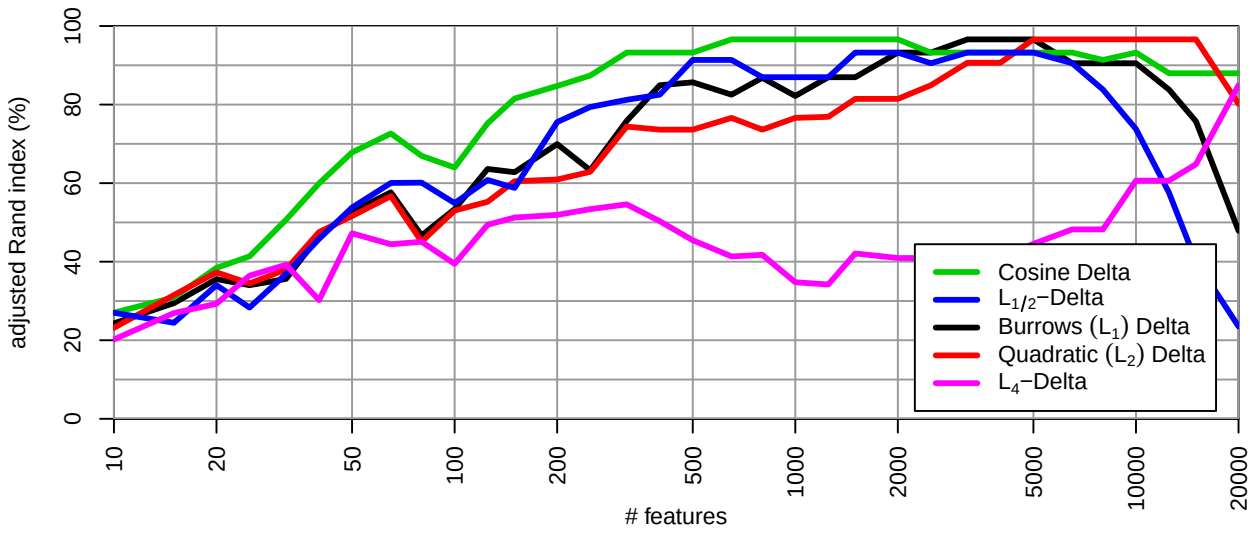

```
ari.plot(zEN, goldEN, transform=clamp, main="English (z-scores clamped to [-2, 2], PAM)")
```



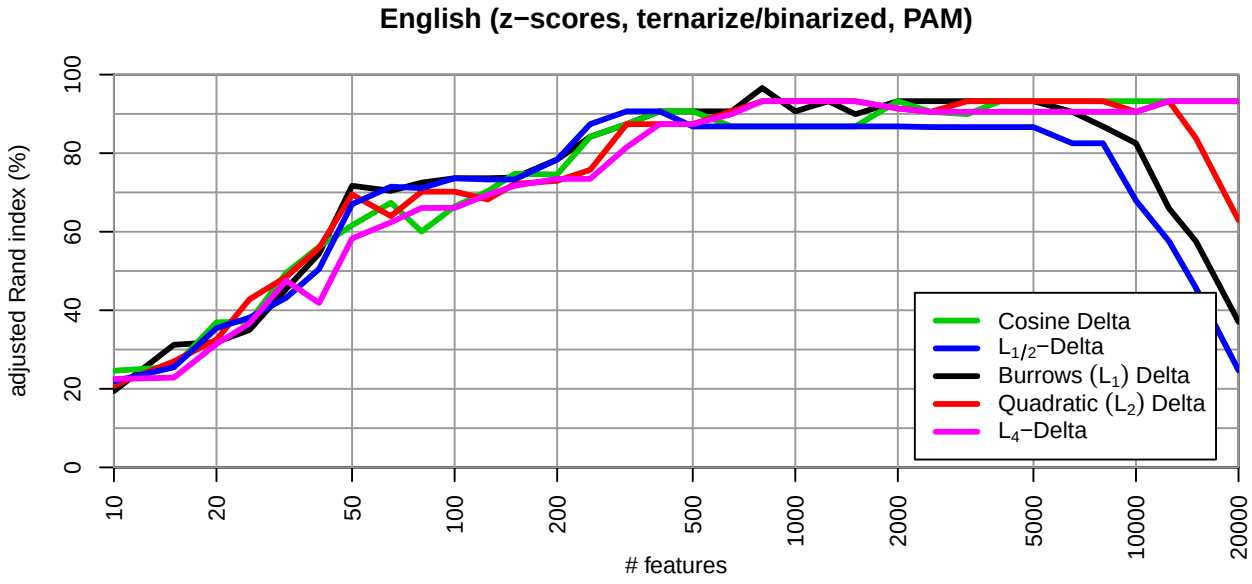

```
ari.plot(zEN, goldEN, transform=ternarize3, main="English (z-scores, ternarized, PAM)")
```



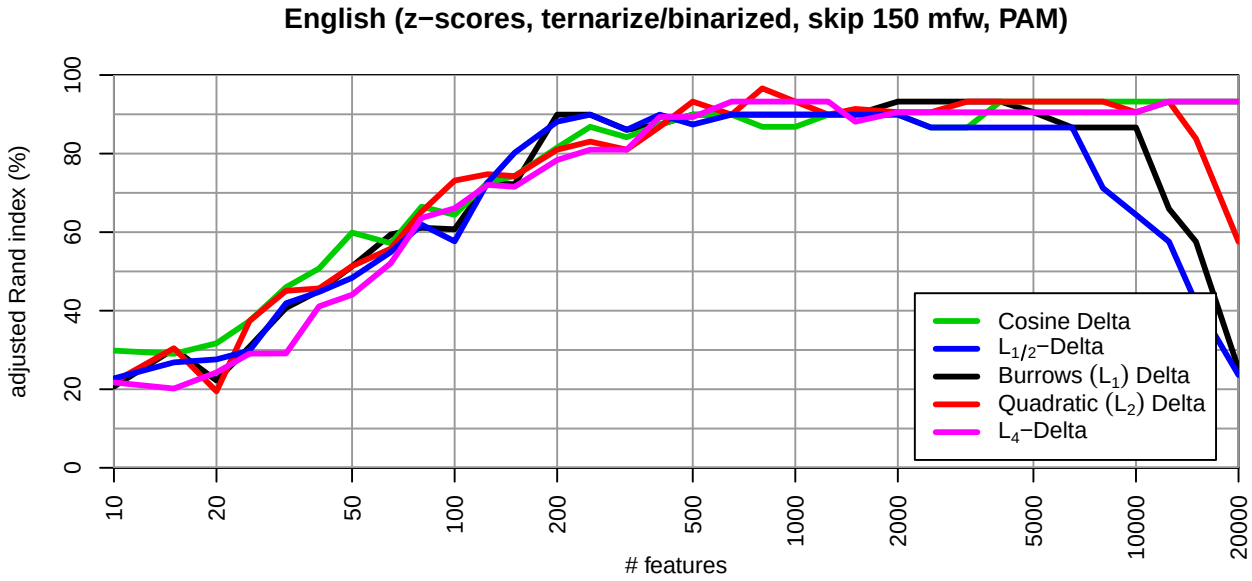

```
ari.plot(zEN, goldEN, transform=function (x) binarize(x, crossover=2000),
         main="English (z-scores, binarized after 2000 mfw, PAM)")
```




```
ari.plot(zEN, goldEN, transform=tern.binarize, main="English (z-scores, ternarize/binarized, PAM)")
```



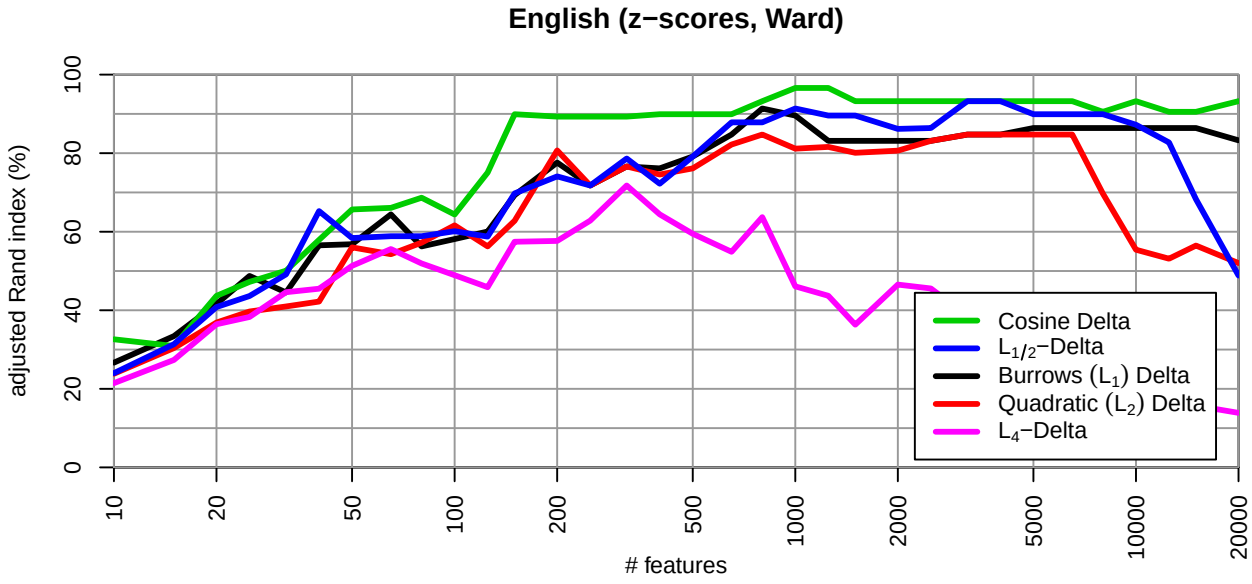

```
ari.plot(zEN, goldEN, skip=150, transform=tern.binarize,
         main="English (z-scores, ternarize/binarized, skip 150 mfw, PAM)")
```



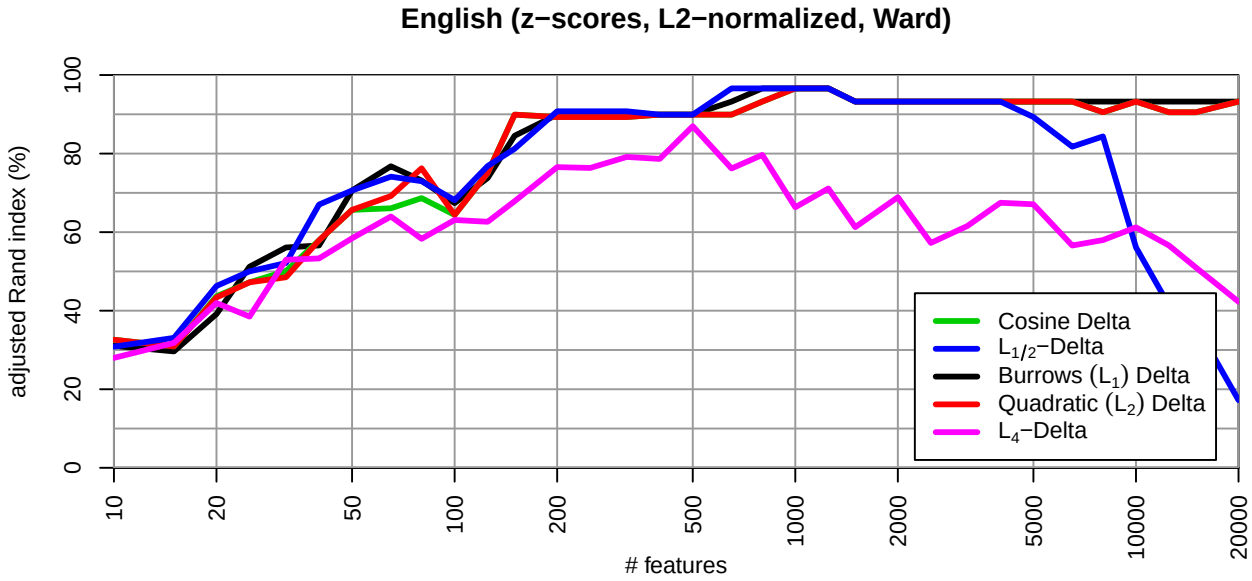

## 1.5 English: Ward Clustering

Replication of results on English corpus, using Ward clustering.

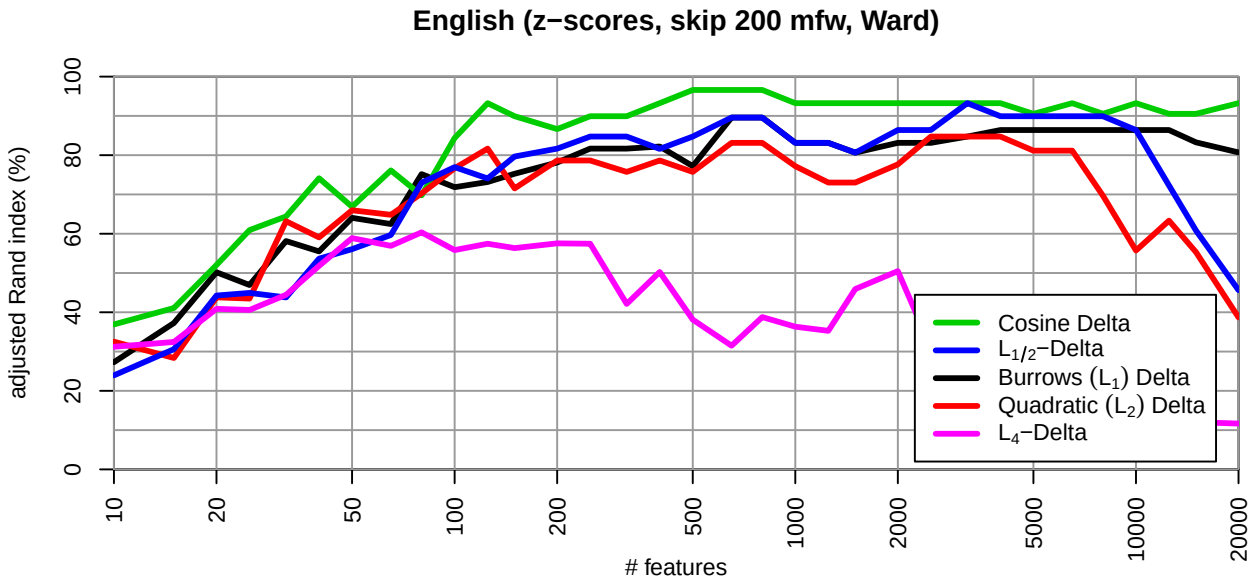
```
ari.plot(zEN, goldEN, clust.method="ward", main="English (z-scores, Ward)")
```



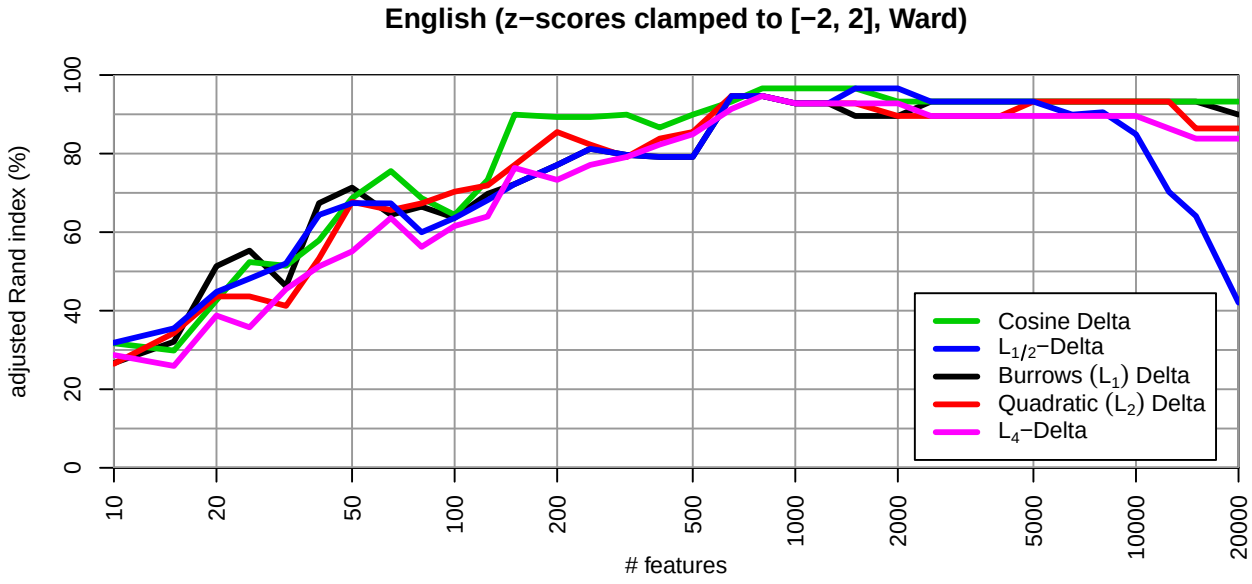

```
ari.plot(zEN, goldEN, clust.method="ward", normalize="euclidean",
         main="English (z-scores, L2-normalized, Ward)")
```



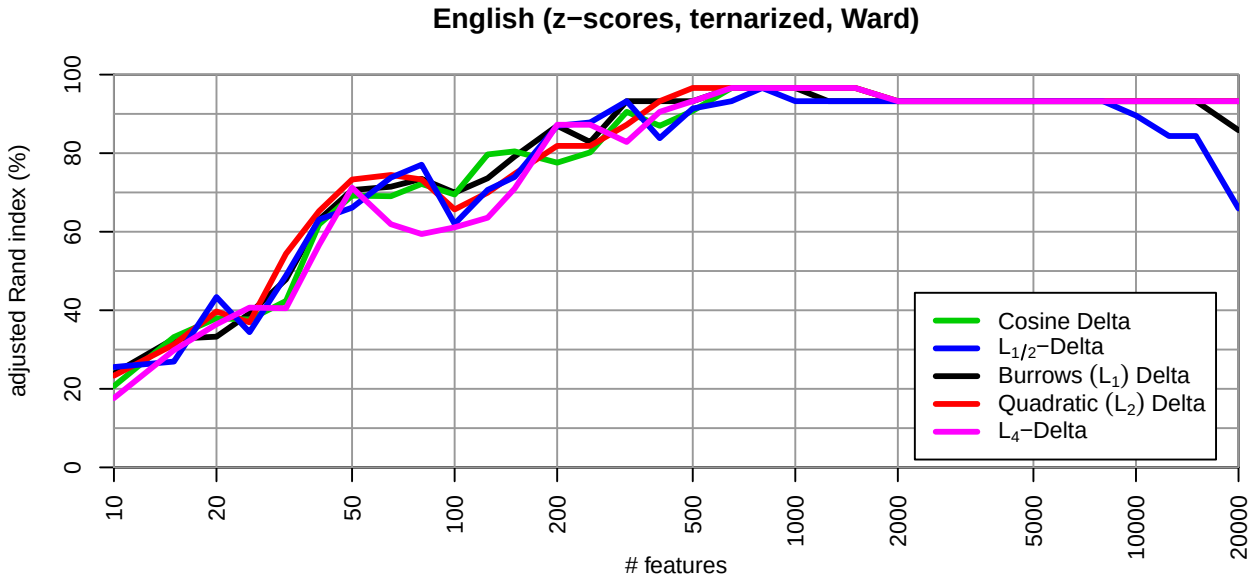

```
ari.plot(zEN, goldEN, clust.method="ward", skip=200,
         main="English (z-scores, skip 200 mfw, Ward)")
```



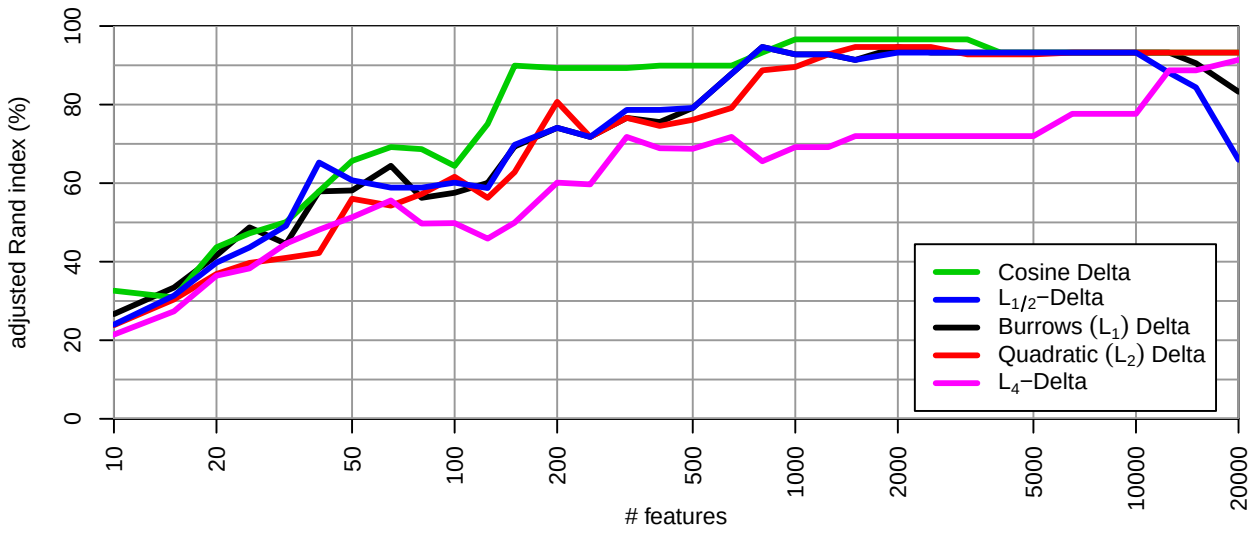

```
ari.plot(zEN, goldEN, clust.method="ward", transform=clamp,
         main="English (z-scores clamped to [-2, 2], Ward)")
```



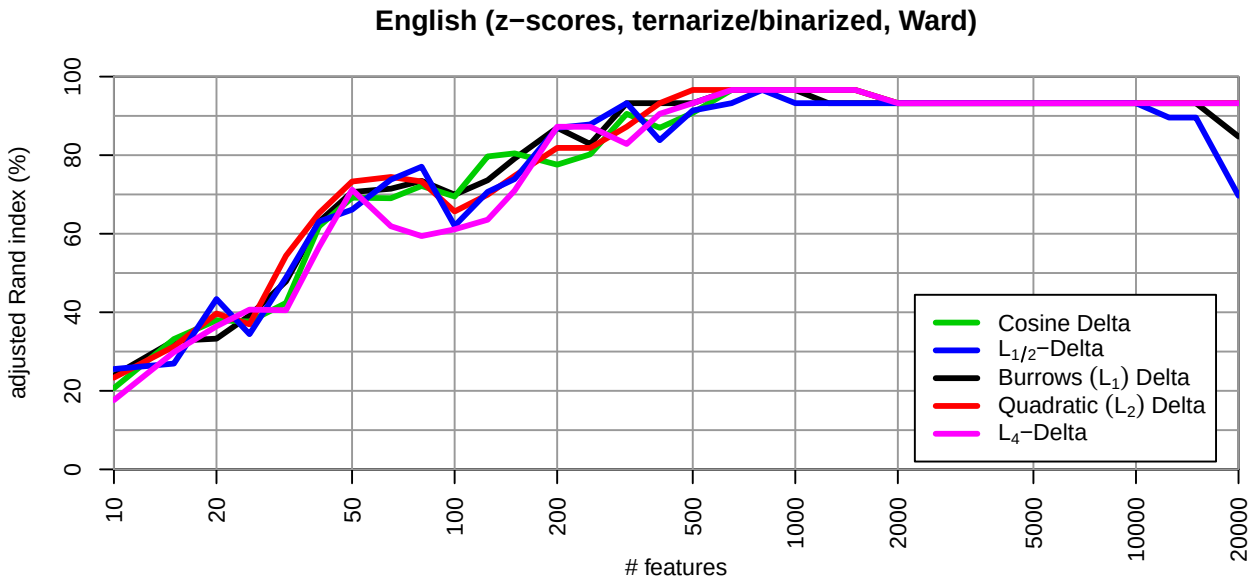

```
ari.plot(zEN, goldEN, clust.method="ward", transform=ternarize3,
         main="English (z-scores, ternarized, Ward)")
```



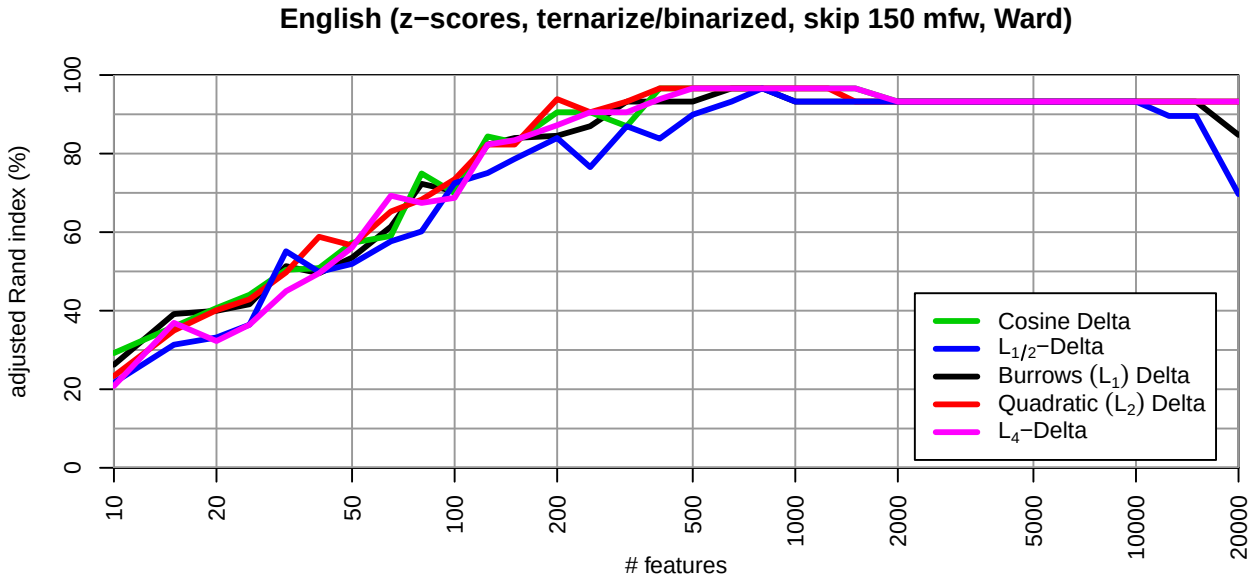

```
ari.plot(zEN, goldEN, clust.method="ward", transform=function (x) binarize(x, crossover=2000),
         main="English (z-scores, binarized after 2000 mfw, Ward)")
```




```
ari.plot(zEN, goldEN, clust.method="ward", transform=tern.binarize,
         main="English (z-scores, ternarize/binarized, Ward)")
```



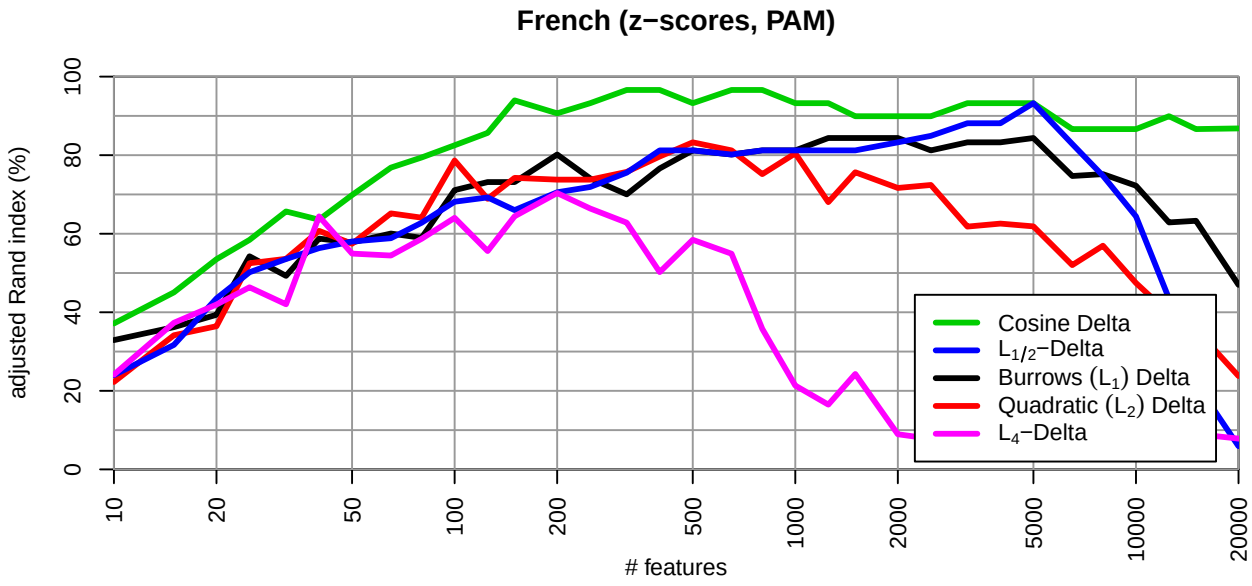

```
ari.plot(zEN, goldEN, clust.method="ward", skip=150, transform=tern.binarize,
         main="English (z-scores, ternarize/binarized, skip 150 mfw, Ward)")
```



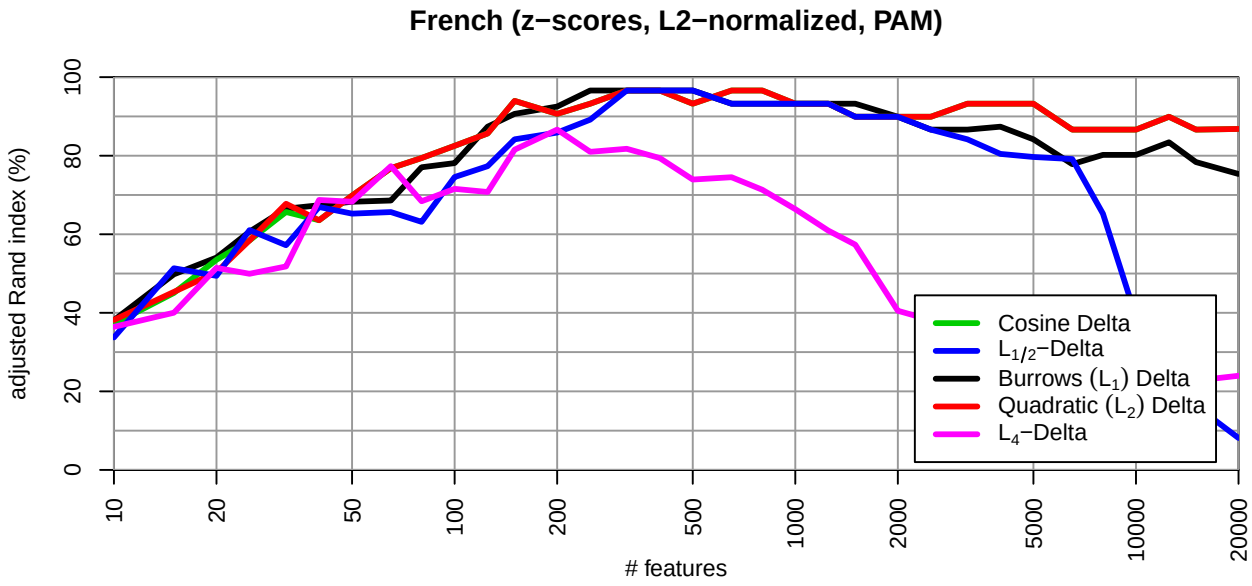

## 1.6 French: PAM Clustering

Replication of results on French corpus, using PAM clustering.

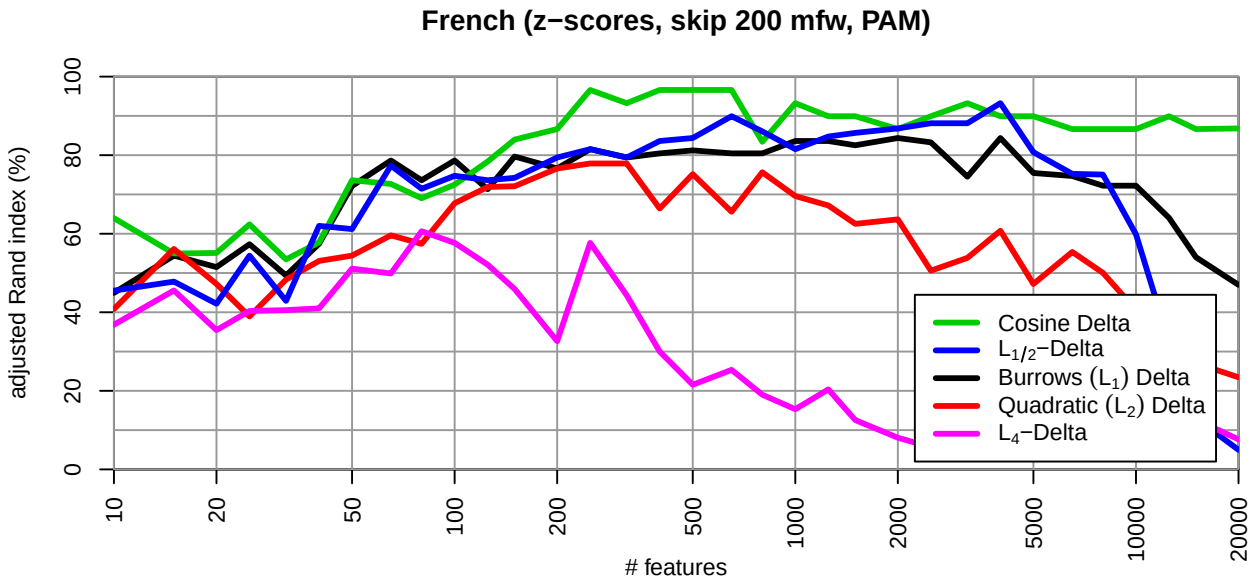
```
ari.plot(zFR, goldFR, main="French (z-scores, PAM)")
```



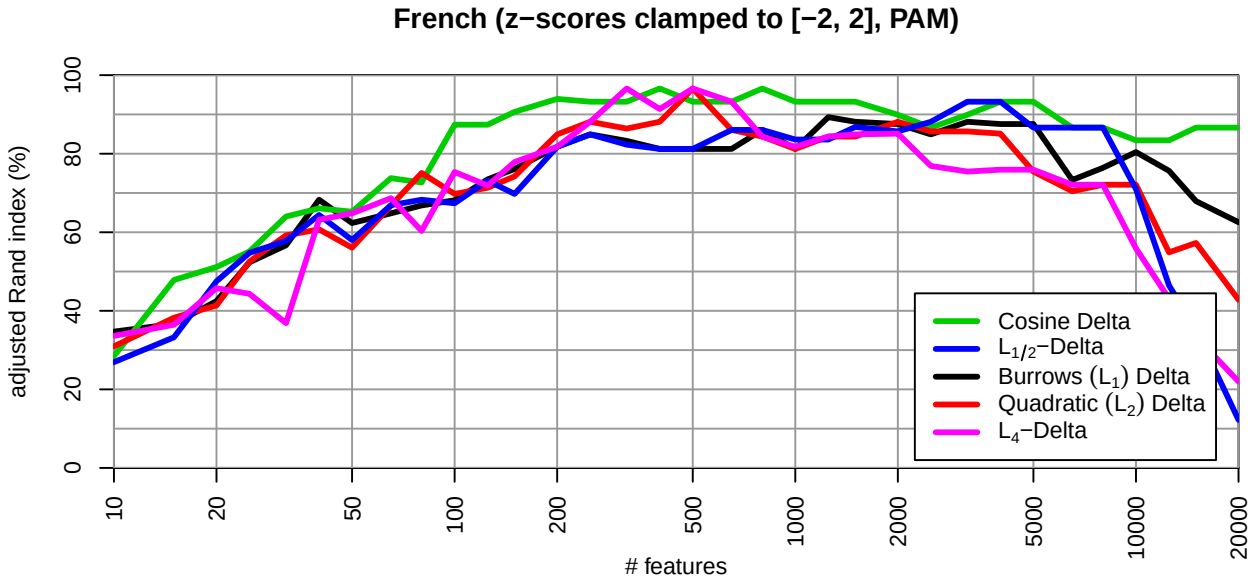

```
ari.plot(zFR, goldFR, normalize="euclidean", main="French (z-scores, L2-normalized, PAM)")
```



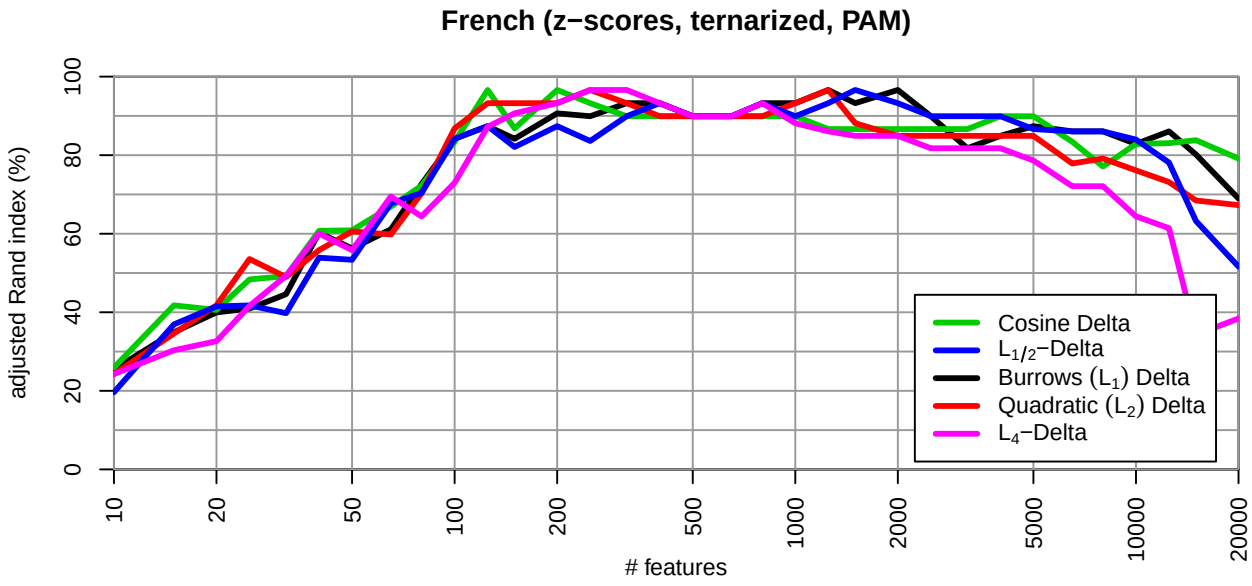

```
ari.plot(zFR, goldFR, skip=200, main="French (z-scores, skip 200 mfw, PAM)")
```



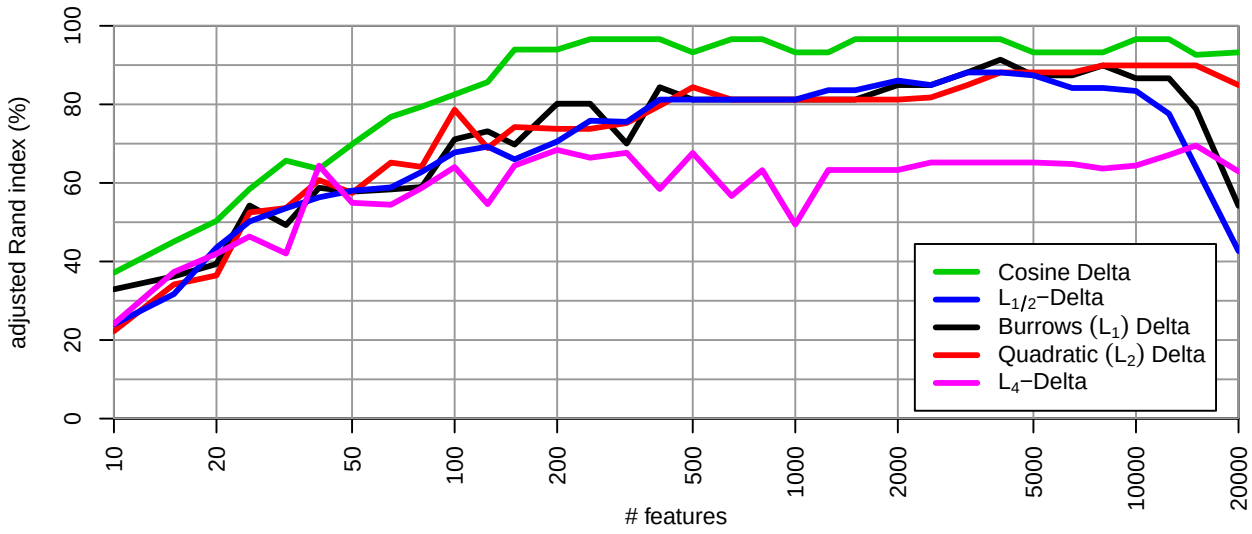

```
ari.plot(zFR, goldFR, transform=clamp, main="French (z-scores clamped to [-2, 2], PAM)")
```



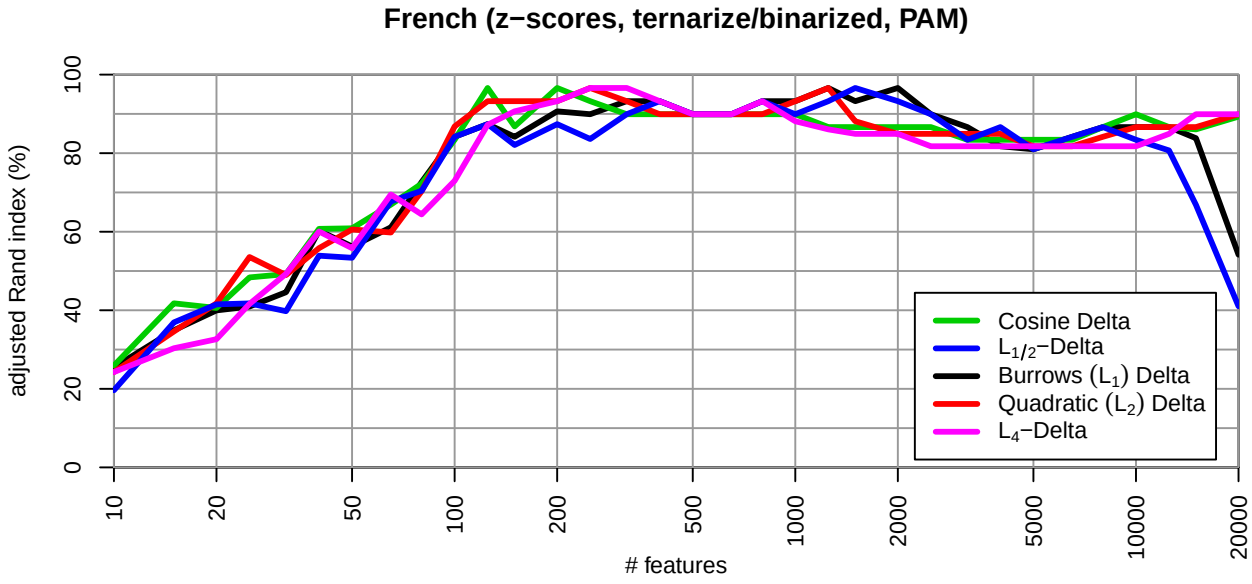

```
ari.plot(zFR, goldFR, transform=ternarize3, main="French (z-scores, ternarized, PAM)")
```



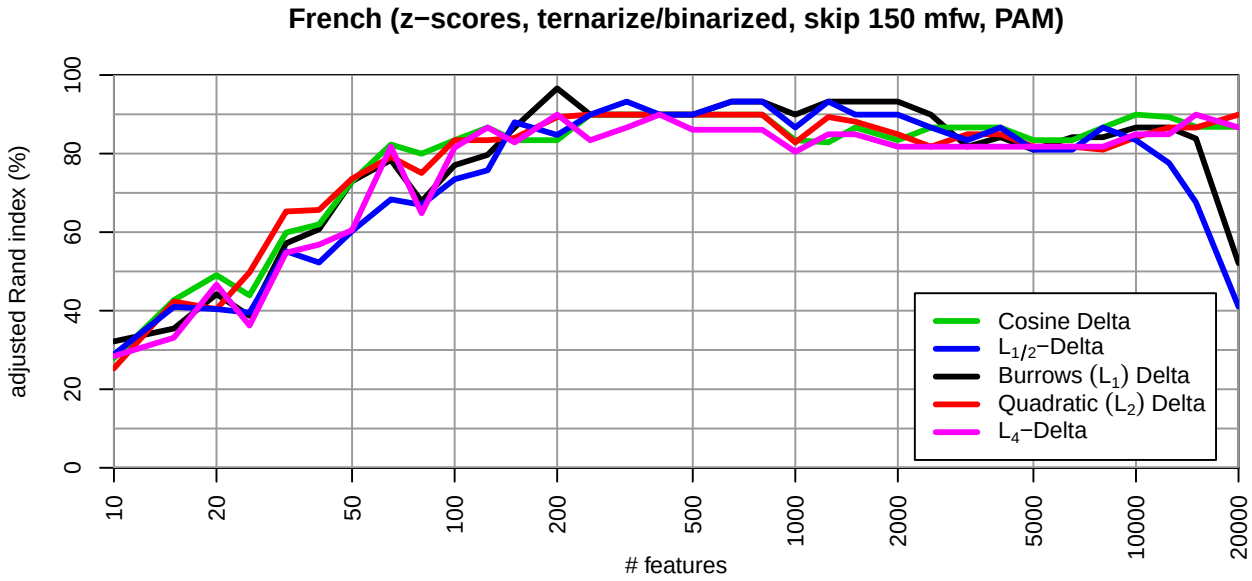

```
ari.plot(zFR, goldFR, transform=function (x) binarize(x, crossover=2000),
         main="French (z-scores, binarized after 2000 mfw, PAM)")
```




```
ari.plot(zFR, goldFR, transform=tern.binarize, main="French (z-scores, ternarize/binarized, PAM)")
```



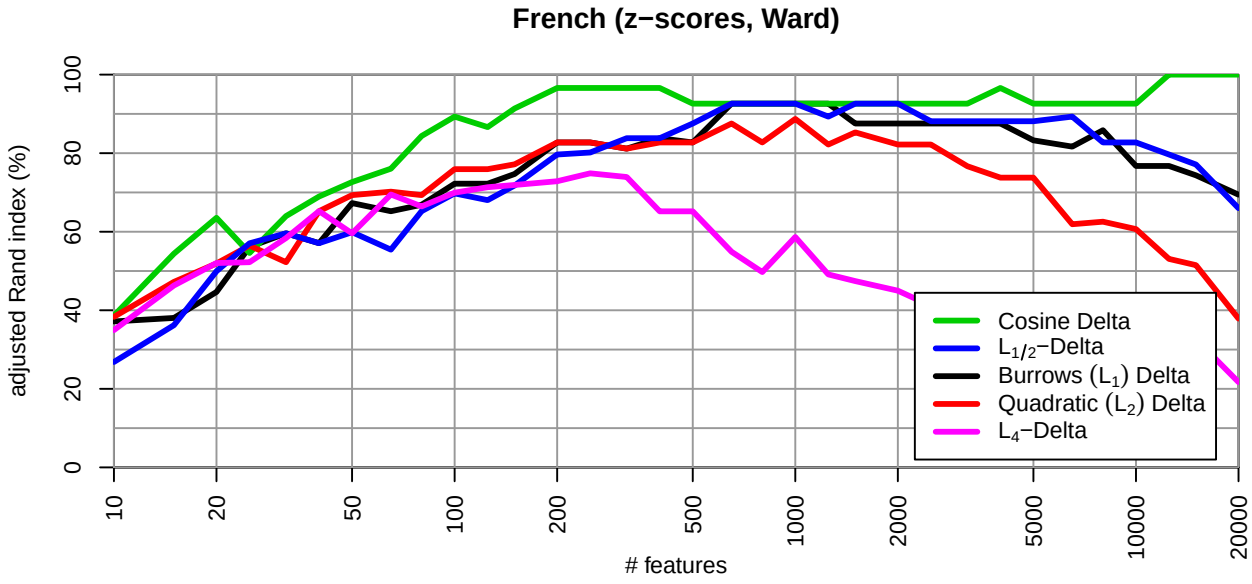

```
ari.plot(zFR, goldFR, skip=150, transform=tern.binarize,
         main="French (z-scores, ternarize/binarized, skip 150 mfw, PAM)")
```



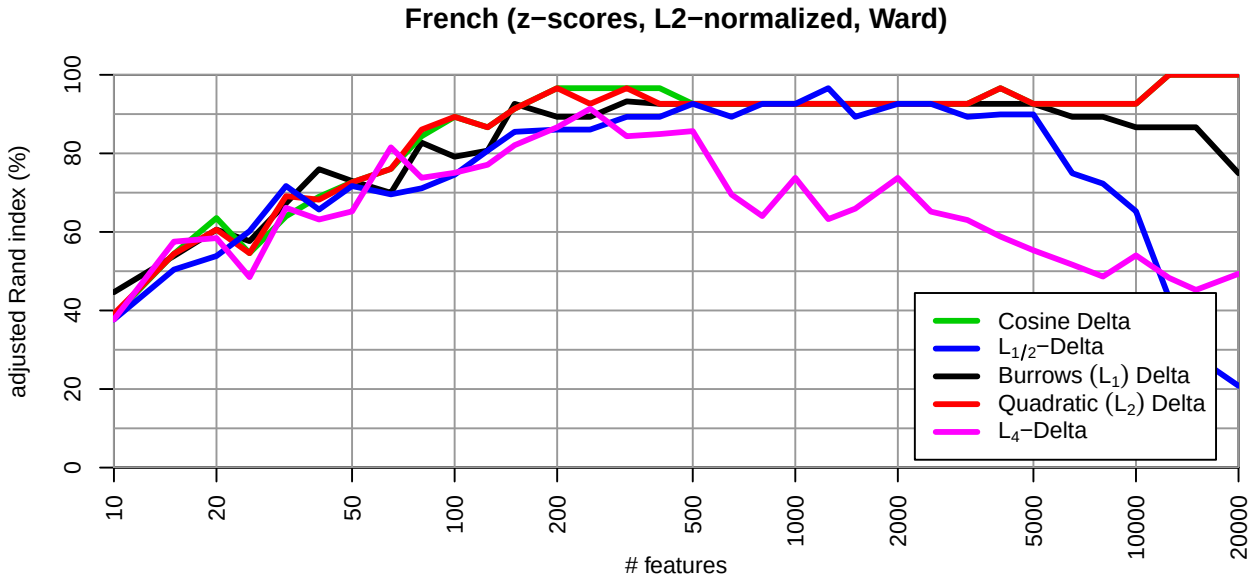

## 1.7 French: Ward Clustering

Replication of results on French corpus, using Ward clustering.

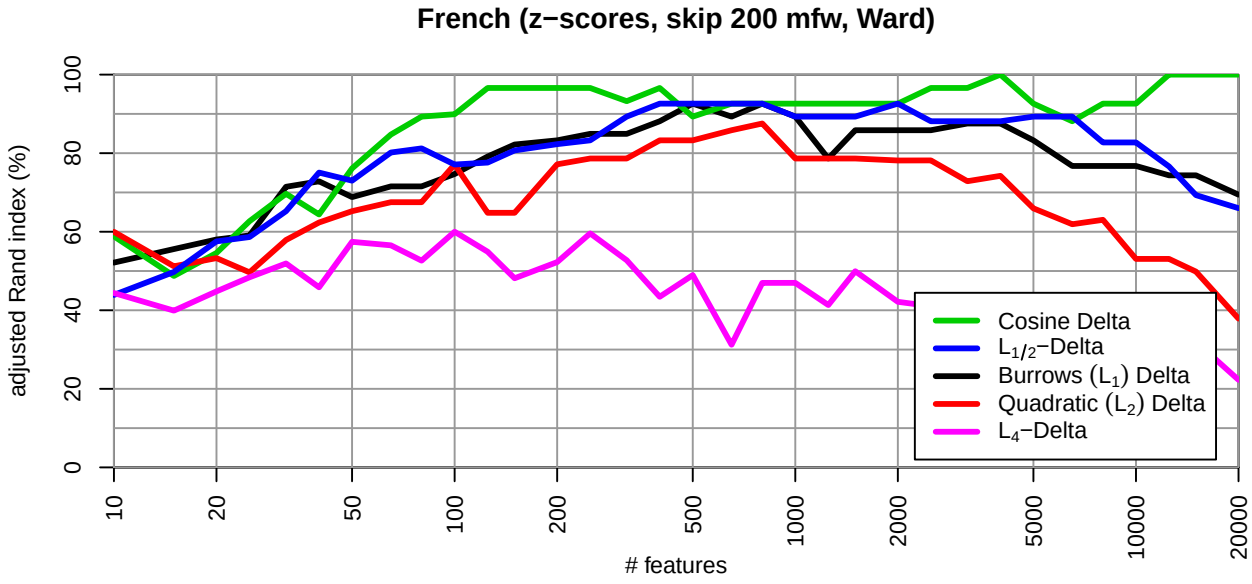
```
ari.plot(zFR, goldFR, clust.method="ward", main="French (z-scores, Ward)")
```



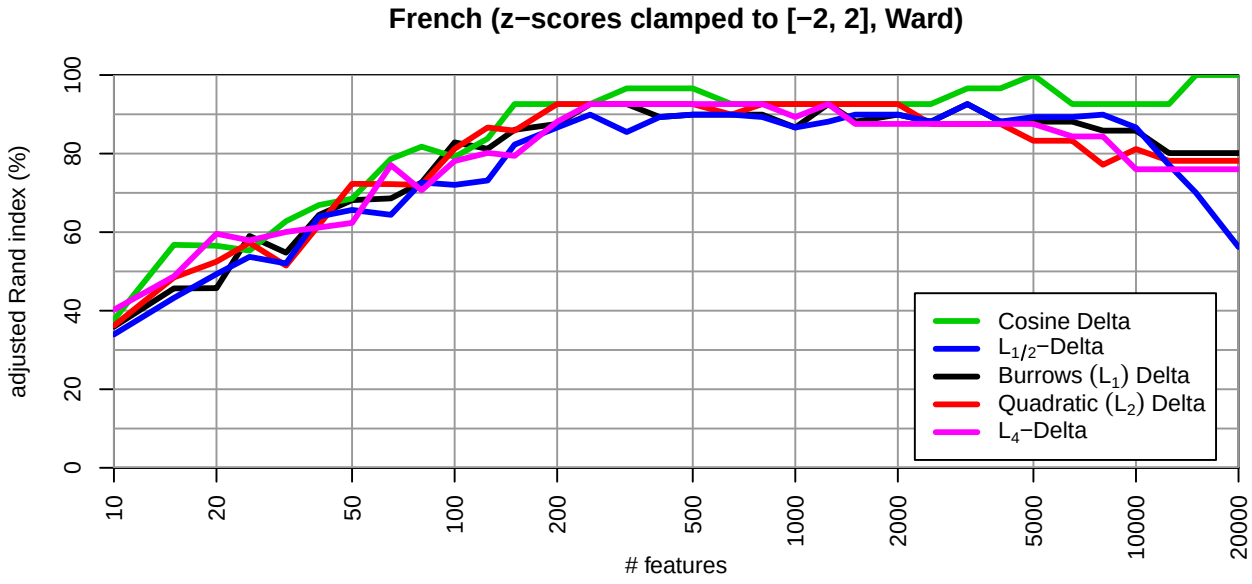

```
ari.plot(zFR, goldFR, clust.method="ward", normalize="euclidean",
         main="French (z-scores, L2-normalized, Ward)")
```



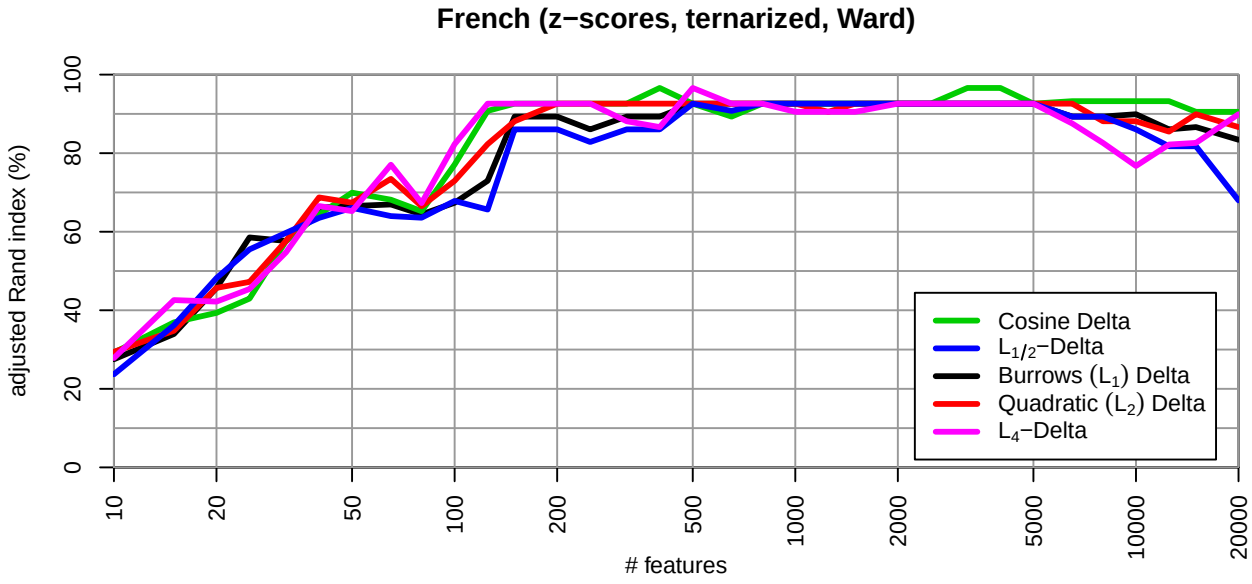

```
ari.plot(zFR, goldFR, clust.method="ward", skip=200,
         main="French (z-scores, skip 200 mfw, Ward)")
```



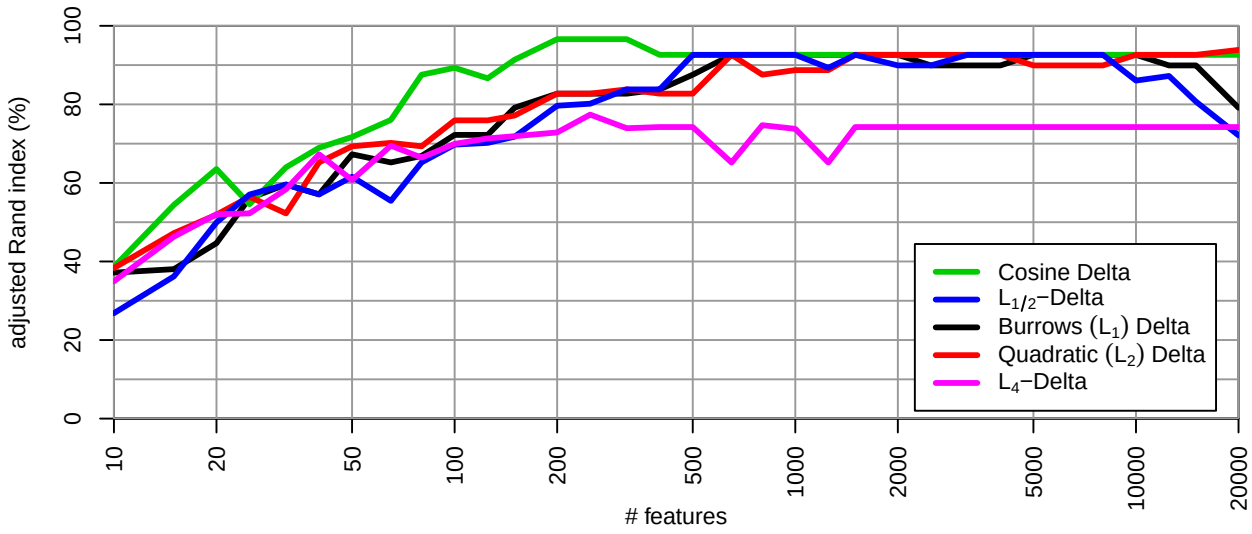

```
ari.plot(zFR, goldFR, clust.method="ward", transform=clamp,
         main="French (z-scores clamped to [-2, 2], Ward)")
```



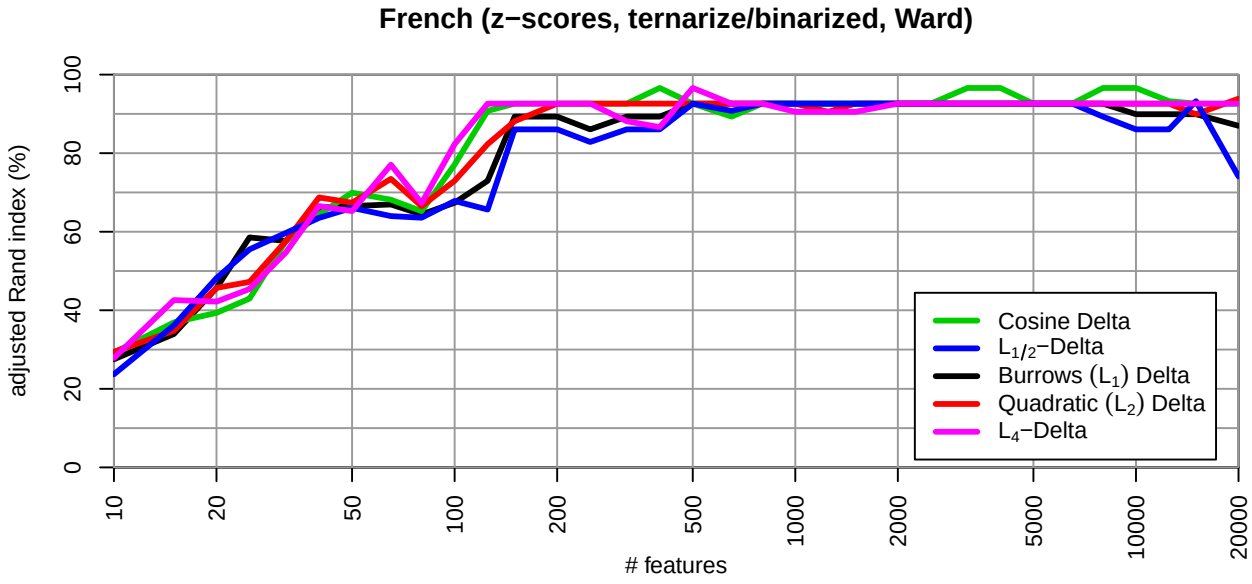

```
ari.plot(zFR, goldFR, clust.method="ward", transform=ternarize3,
         main="French (z-scores, ternarized, Ward)")
```



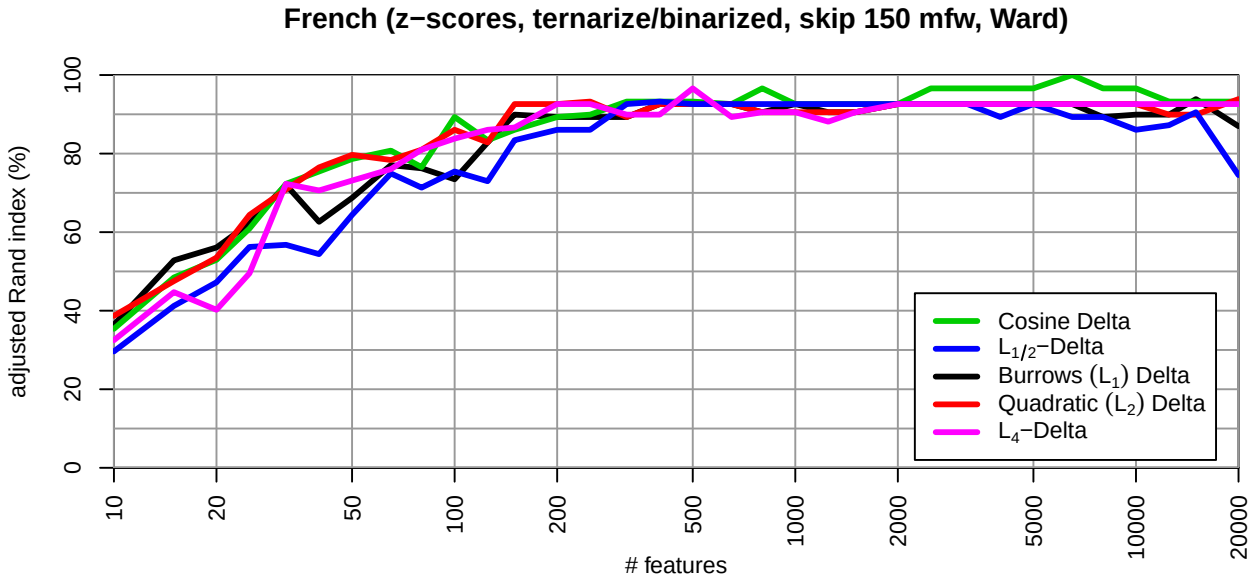

```
ari.plot(zFR, goldFR, clust.method="ward", transform=function (x) binarize(x, crossover=2000),
         main="French (z-scores, binarized after 2000 mfw, Ward)")
```




```
ari.plot(zFR, goldFR, clust.method="ward", transform=tern.binarize,
         main="French (z-scores, ternarize/binarized, Ward)")
```




```
ari.plot(zFR, goldFR, clust.method="ward", skip=150, transform=tern.binarize,
         main="French (z-scores, ternarize/binarized, skip 150 mfw, Ward)")
```




# 2 Clustering experiments

## 2.1 Dendrograms, silhouette width and the number of clusters

Select a reasonably good – but not perfect – clustering as case study: German, $\Delta_B$, $n_w = 1000$.

```
M <- zDE[, 1:1000]
DM <- delta.dist(M, method="manhattan") # Burrows Delta
```

For colour-coding authors in silhouette plots etc., we need a palette of 25 discernible shades. It is also easier

to pick out the correct author if the gold standard vectors are labelled with the document names.

```
authors.pal <- rainbow_hcl(25) # colour palette for 25 authors
names(goldDE) <- rownames(zDE)
colDE <- authors.pal[factor(goldDE)] # look up plot colours by text label
names(colDE) <- names(goldDE)
```

Support function for silhouette plot given flat clustering and distance matrix, as well as gold standard labels and colouring (both named with text labels). Prints ARI score for this clustering.

```
do.silhouette <- function (clusters, DM, gold=goldDE, col=colDE,
                           cex.names=.4, mar=c(4,10,2,1)+.1, ...) {
  ari <- 100 * adjustedRandIndex(clusters, gold)
  cat(sprintf("ARI = %5.1f%%\n", ari))
  sil <- silhouette(clusters, DM)
  rownames(sil) <- names(gold)
  sil <- sortSilhouette(sil)
  par.save <- par(mar=mar)
  plot(sil, col=col[rownames(sil)], nmax.lab=100, max.strlen=100,
       cex.names=cex.names, ...)
  par(par.save)
  invisible(ari)
}
```

Average silhouette width can be used to select a suitable number of clusters automatically. We visualized the procedure by plotting silhouette width and ARI for a range of different cluster counts.

```
silhouette.profile <- function (method, DM, gold=goldDE, k=10:40,
                                col1=seaborn.pal[2], col2="black", col3=seaborn.pal[3],
                                ...) {
  if (inherits(method, "hclust")) {
    ## extract clusters from pre-computed hierarchical clustering
    clusters.list <- lapply(k, function (i) cutree(method, i))
  } else if (is.function(method)) {
    ## method must be function object that returns cluster assignment
    clusters.list <- lapply(k, function (i) method(DM, i))
  } else if (method == "pam") {
    clusters.list <- lapply(k, function (i) pam(DM, i, diss=TRUE)$clustering)
  } else {
    hc <- agnes(DM, diss=TRUE, method=method)
    clusters.list <- lapply(k, function (i) cutree(hc, i))
  }
  ari.vec <- sapply(clusters.list, function (C) adjustedRandIndex(C, gold))
  sil.vec <- sapply(clusters.list, function (C) summary(silhouette(C, dmatrix=DM))$avg.width)
  plot(0, 0, xlim=range(k), ylim=c(0, 1), xlab="# clusters", ylab="", xaxs="i", yaxs="i", ...)
  abline(h=seq(0, 1, .1), lwd=1, col="grey40")
  rug(k) # tick marks for individual k values
  lines(k, ari.vec, type="o", col=col1, lwd=3, pch=20)
  lines(k, sil.vec, type="o", col=col2, lwd=2, pch=20)
  i.max <- which.max(sil.vec)
  abline(v=k[i.max], lwd=2, col=col3)
  text(k[i.max], .02, sprintf("ARI = %.1f%%", 100 * ari.vec[i.max]),
       adj=c(0, 1.4), col=col3, srt=90)
  legend("topleft", inset=.02, bg="white", lwd=3, col=c(col1, col2),
         legend=c("ARI", "silhouette"))
}
```

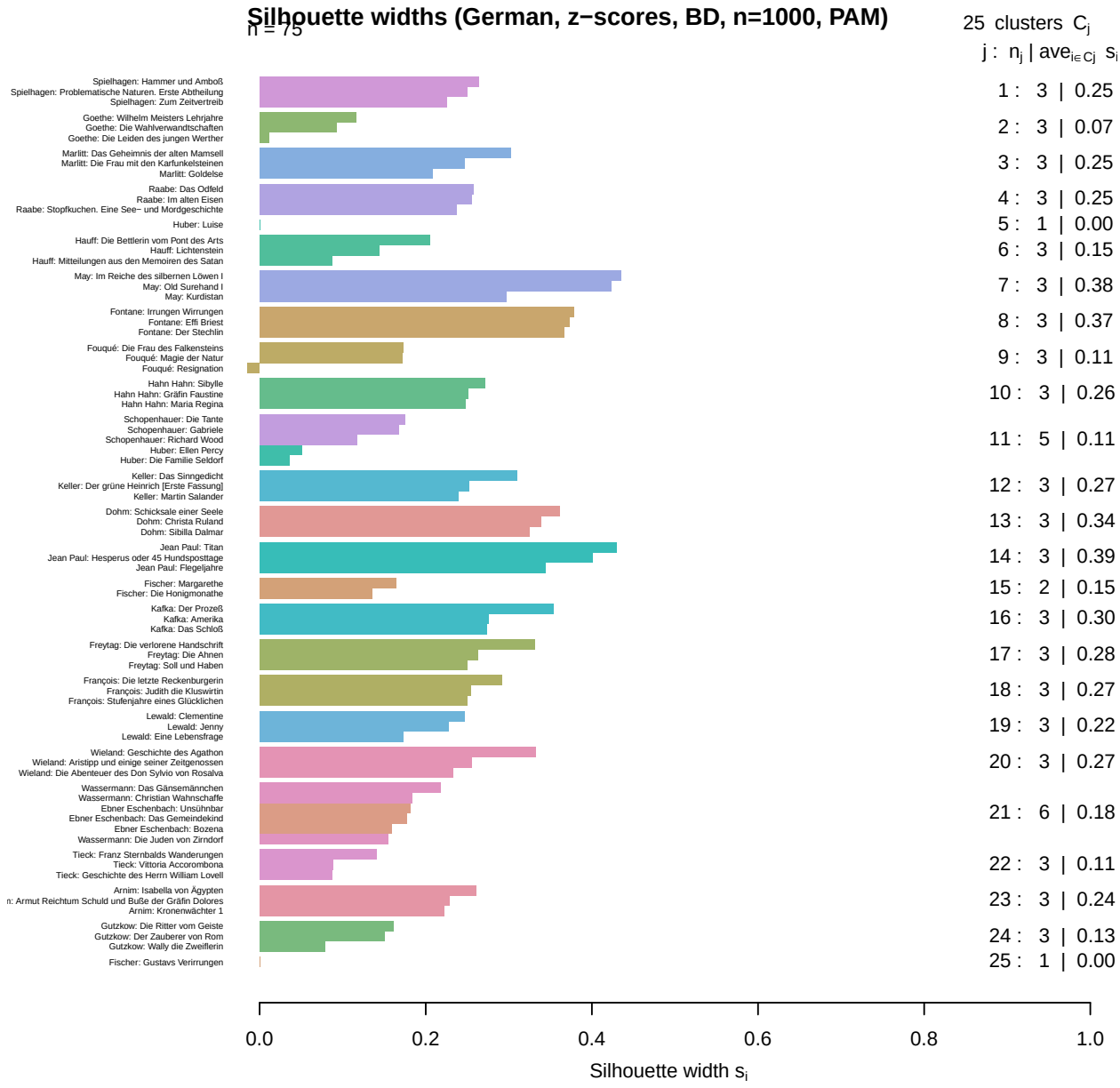
Support function for dendrogram given hierarchical clustering (must be coercible to `hclust` object), with colouring based on gold standard labels. The cut into a flat clustering can only be displayed if `horiz=FALSE`.

```
do.dendrogram <- function (hc, gold=goldDE, col=colDE, horiz=TRUE, cut=NULL,
                           lab.cex=.75, lab.font=2, dLeaf=(if (horiz) strwidth("x") else strheight("x"))
                           mar=if (horiz) c(2,2,2,8)+.1 else c(10,2,2,2)+.1, ...) {
  hc <- as.hclust(hc)
  stopifnot(all(hc$order.lab == names(gold)[hc$order])) # may need to ensure labels here
  if (horiz && !is.null(cut)) stop("cut rectangles can only be displayed if horiz=FALSE")
  dend <- as.dendrogram(hc, hang=0)
  dend <- dendrapply(dend, function (N) {
    if (is.leaf(N)) {
      lab <- attr(N, "label") # set display parameters for leaf labels
      nodePar <- attr(N, "nodePar")
      nodePar$lab.cex <- lab.cex
      nodePar$lab.col <- col[lab]
      nodePar$lab.font <- lab.font
      nodePar$pch <- NA
      attr(N, "nodePar") <- nodePar
    }
    N
  })
  par.save <- par(mar=mar)
  plot(dend, horiz=horiz, dLeaf=dLeaf, ...)
  if (!is.null(cut)) rect.hclust(hc, k=cut, border="#666666")
  par(par.save)
}
```

PAM clustering with silhouette:

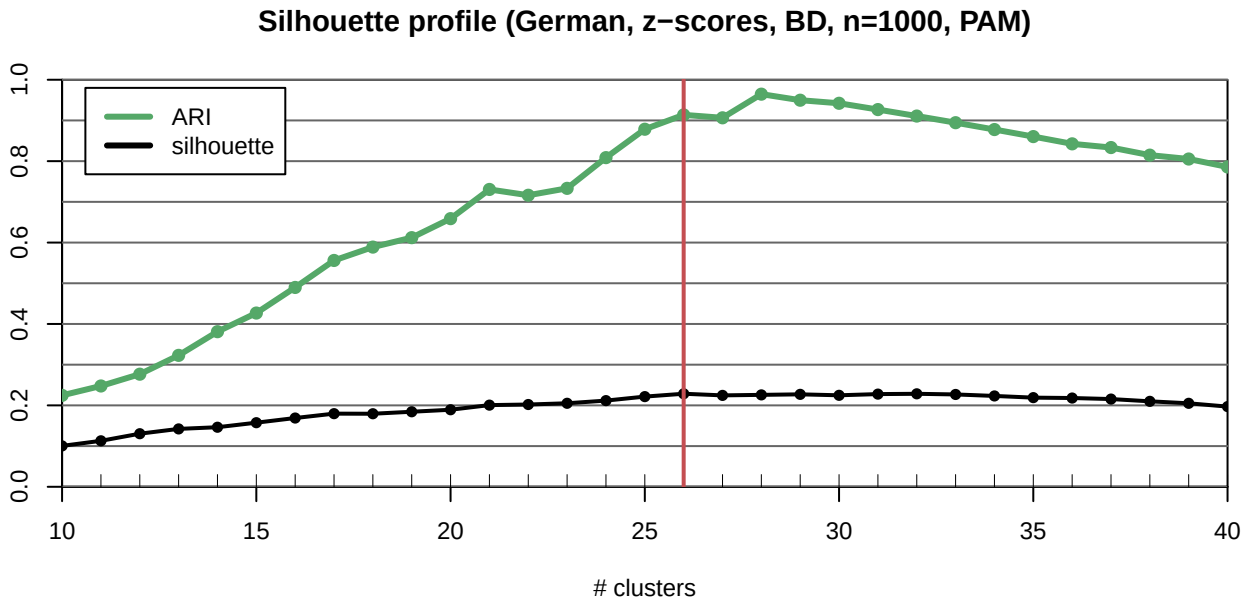
```
cl.pam <- pam(DM, diss=TRUE, k=25)
do.silhouette(cl.pam$clustering, DM, main="Silhouette widths (German, z-scores, BD, n=1000, PAM)")
```

```
## ARI =  87.8%
```




Using average silhouette width as a criterion, we can automatically determine almost exactly the "correct" number of clusters, reaching a maximum at $k = 26$. Note that ARI is even a little better (above 90%) than for fixed $k = 25$, but the profile of average silhouette width is rather flat and the automatic choice of $k$ may well be unstable. Optimal ARI would have been obtained for $28 \leq k \leq 30$, because the clustering algorithm would no longer have been forced to merge Wassermann and Ebner-Eschenbach into a single cluster.

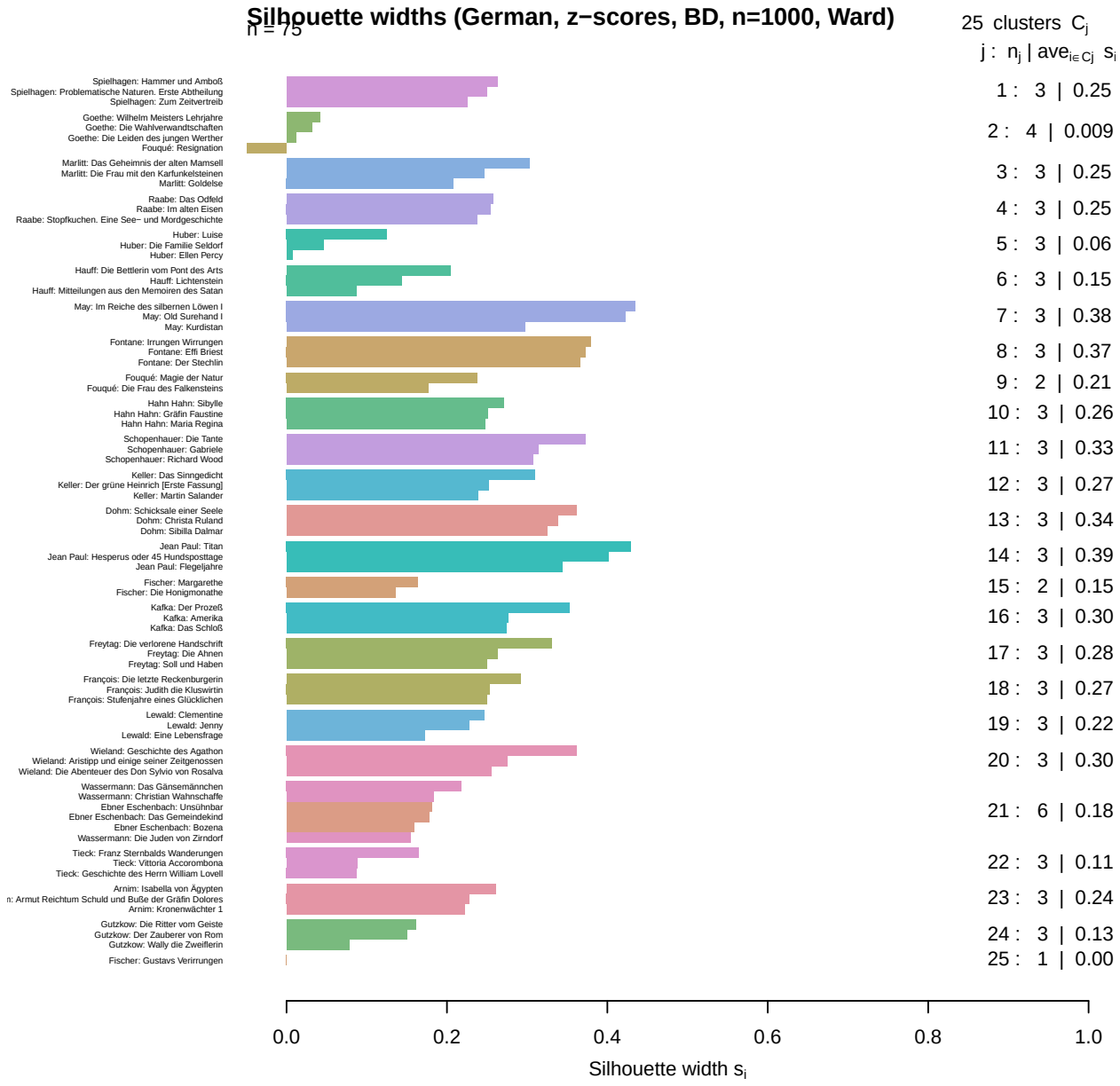
```
silhouette.profile("pam", DM, main="Silhouette profile (German, z-scores, BD, n=1000, PAM)")
```



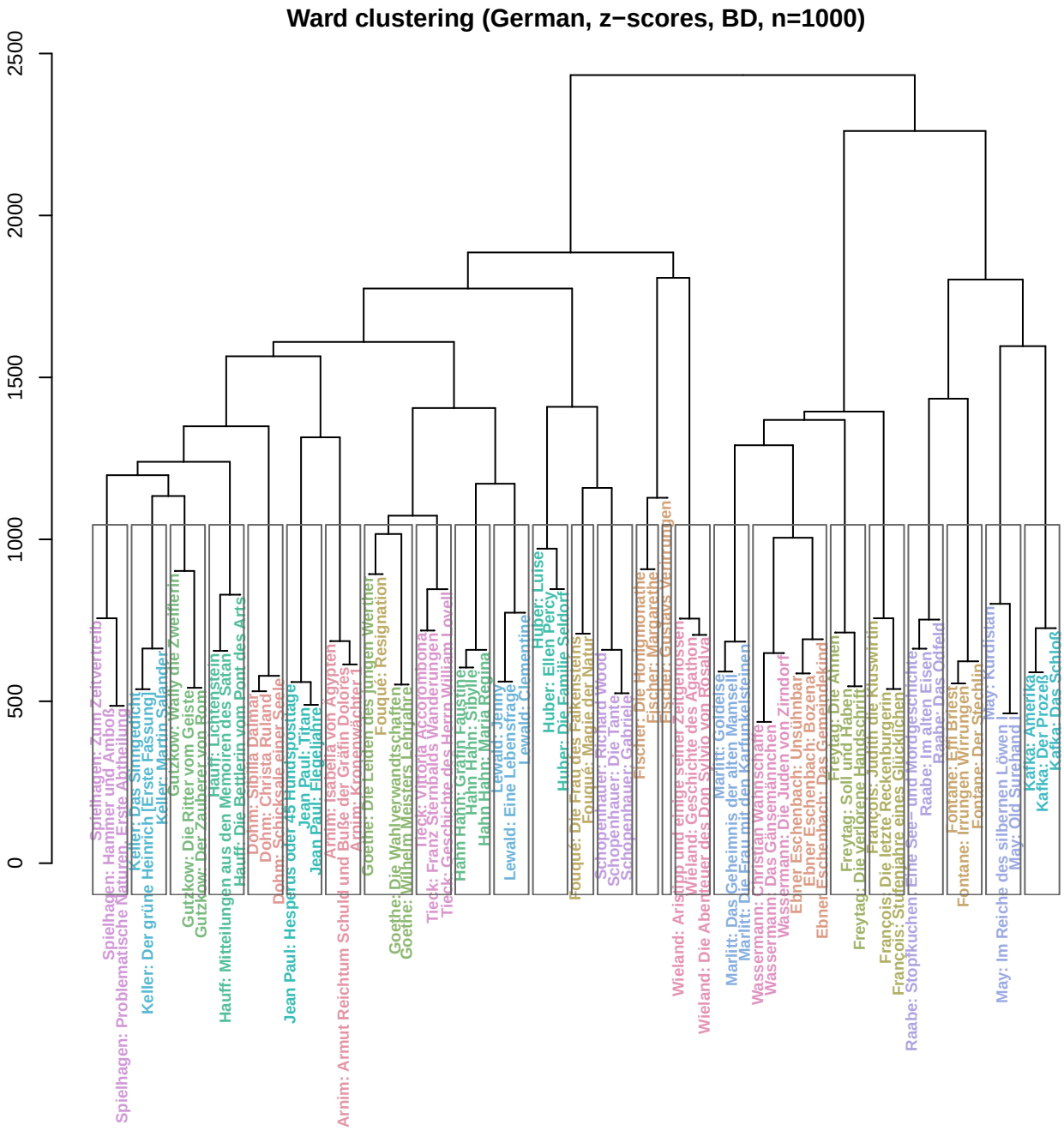

Ward clustering with silhouette (for split into $k = 25$ clusters) and dendrogram:

```
hcl.ward <- agnes(DM, diss=TRUE, method="ward")
cl.ward <- cutree(hcl.ward, 25)
do.silhouette(cl.ward, DM, main="Silhouette widths (German, z-scores, BD, n=1000, Ward)")
```

```
## ARI =  89.6%
```

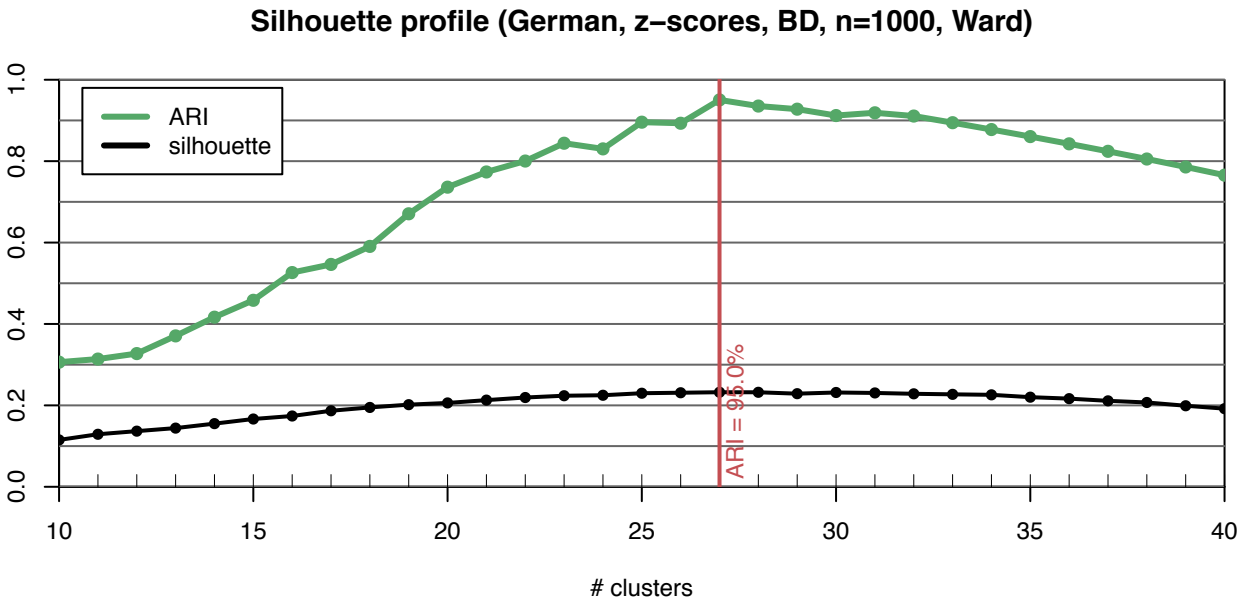



```
do.dendrogram(hcl.ward, horiz=FALSE, cut=25, main="Ward clustering (German, z-scores, BD, n=1000)")
```



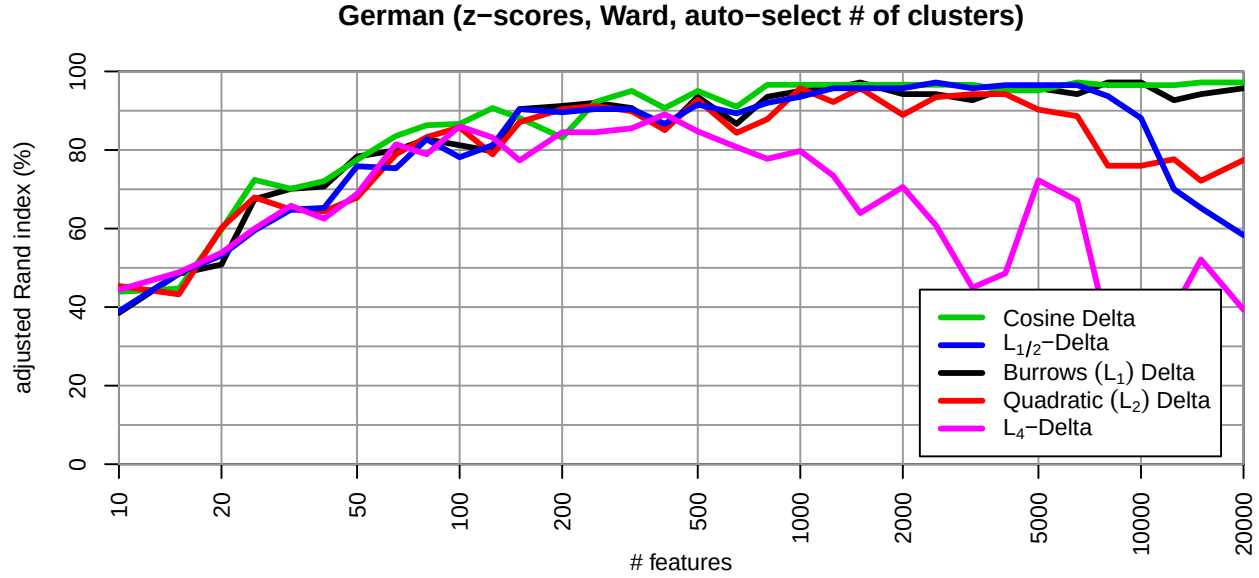

For Ward clustering, average silhouette width actually chooses the optimal clustering with $k = 27$. Again, the silhouette profile is rather flat over a wide range of $k$ values, so the automatic choice may be unstable.

```
silhouette.profile("ward", DM, main="Silhouette profile (German, z-scores, BD, n=1000, Ward)")
```



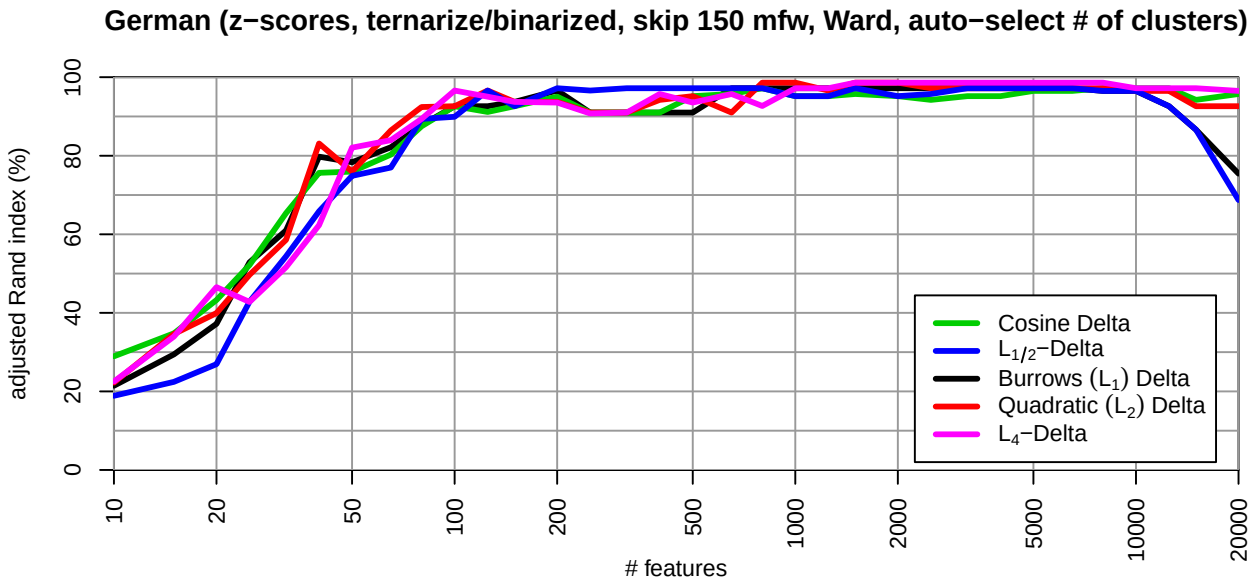

With these promising results, let us see how well Delta performs if the number of clusters is always chosen automatically based on the data points rather than based on the true number of authors in the gold standard. Ward clustering shows to be considerably more robust here compared to PAM (presumably because all flat clusterings are derived from the same hierarchy, while PAM might find substantially different solutions for different cluster counts).

```
ari.plot(zDE, goldDE, n.clusters=15:35, clust.method="ward",
         main="German (z-scores, Ward, auto-select # of clusters)")
```

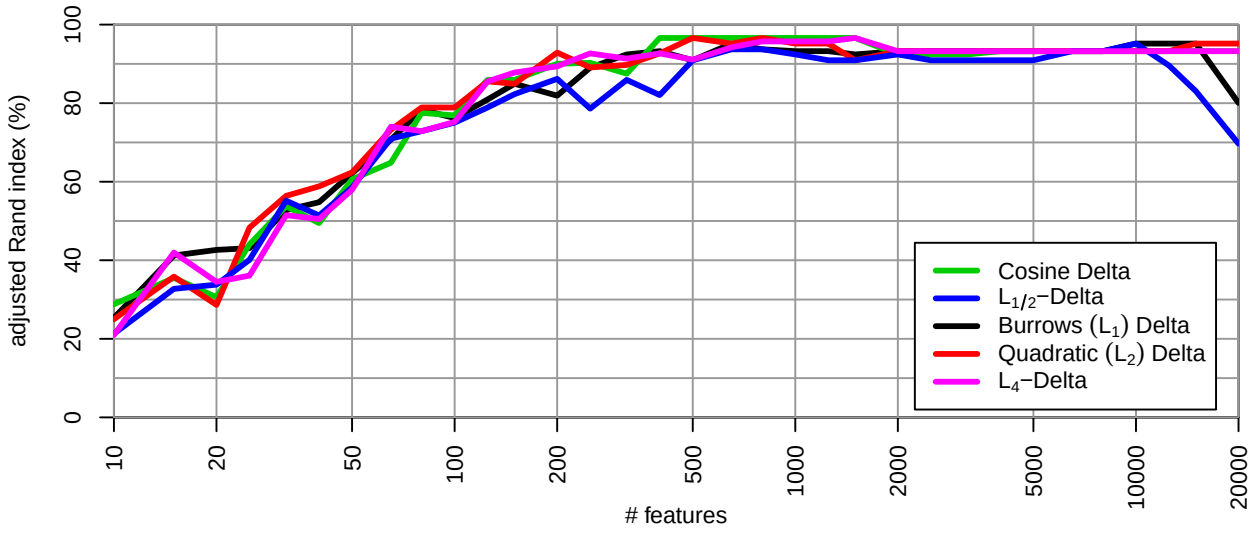



The same holds for the optimal ternarize/binarized features vectors (excluding the first 150 mfw), in all three languages.

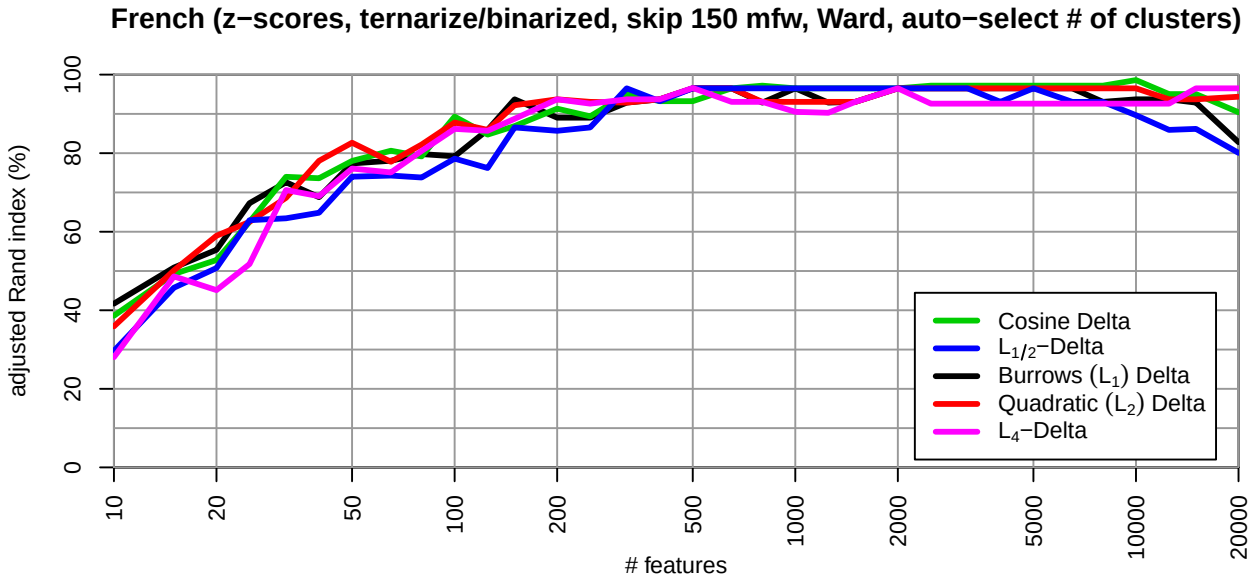
```
ari.plot(zDE, goldDE, n.clusters=15:35, clust.method="ward", skip=150, transform=tern.binarize,
         main="German (z-scores, ternarize/binarized, skip 150 mfw, Ward, auto-select # of clusters)")
```




```
ari.plot(zEN, goldEN, n.clusters=15:35, clust.method="ward", skip=150, transform=tern.binarize,
         main="English (z-scores, ternarize/binarized, skip 150 mfw, Ward, auto-select # of clusters)")
```




```
ari.plot(zFR, goldFR, n.clusters=15:35, clust.method="ward", skip=150, transform=tern.binarize,
         main="French (z-scores, ternarize/binarized, skip 150 mfw, Ward, auto-select # of clusters)")
```



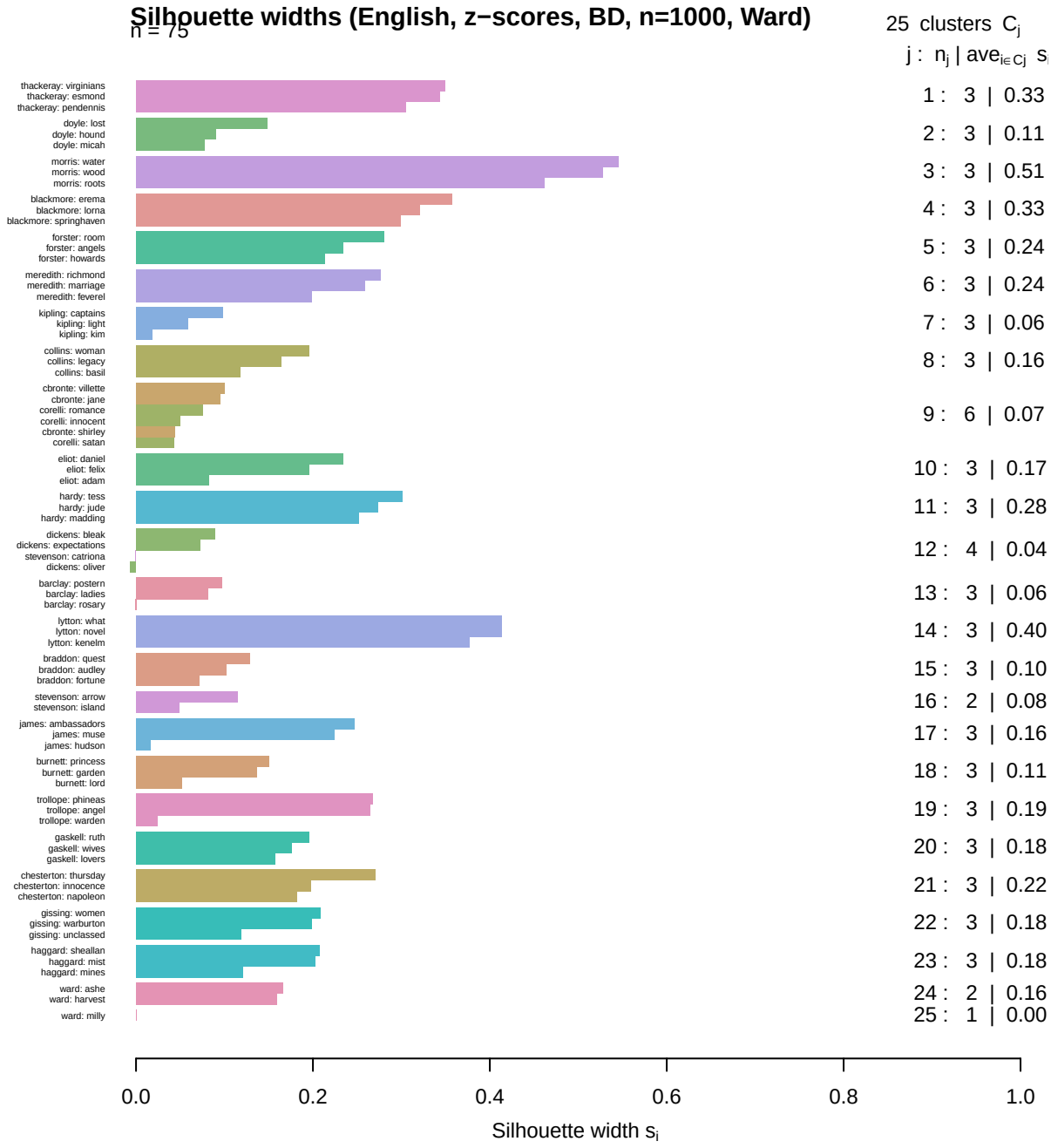

### 2.1.1 Corresponding illustrations for English

```
M.en <- zEN[, 1:1000]
DM.en <- delta.dist(M.en, method="manhattan") # Burrows Delta
names(goldEN) <- rownames(zEN)
colEN <- authors.pal[factor(goldEN)] # look up plot colours by text label
names(colEN) <- names(goldEN)
```

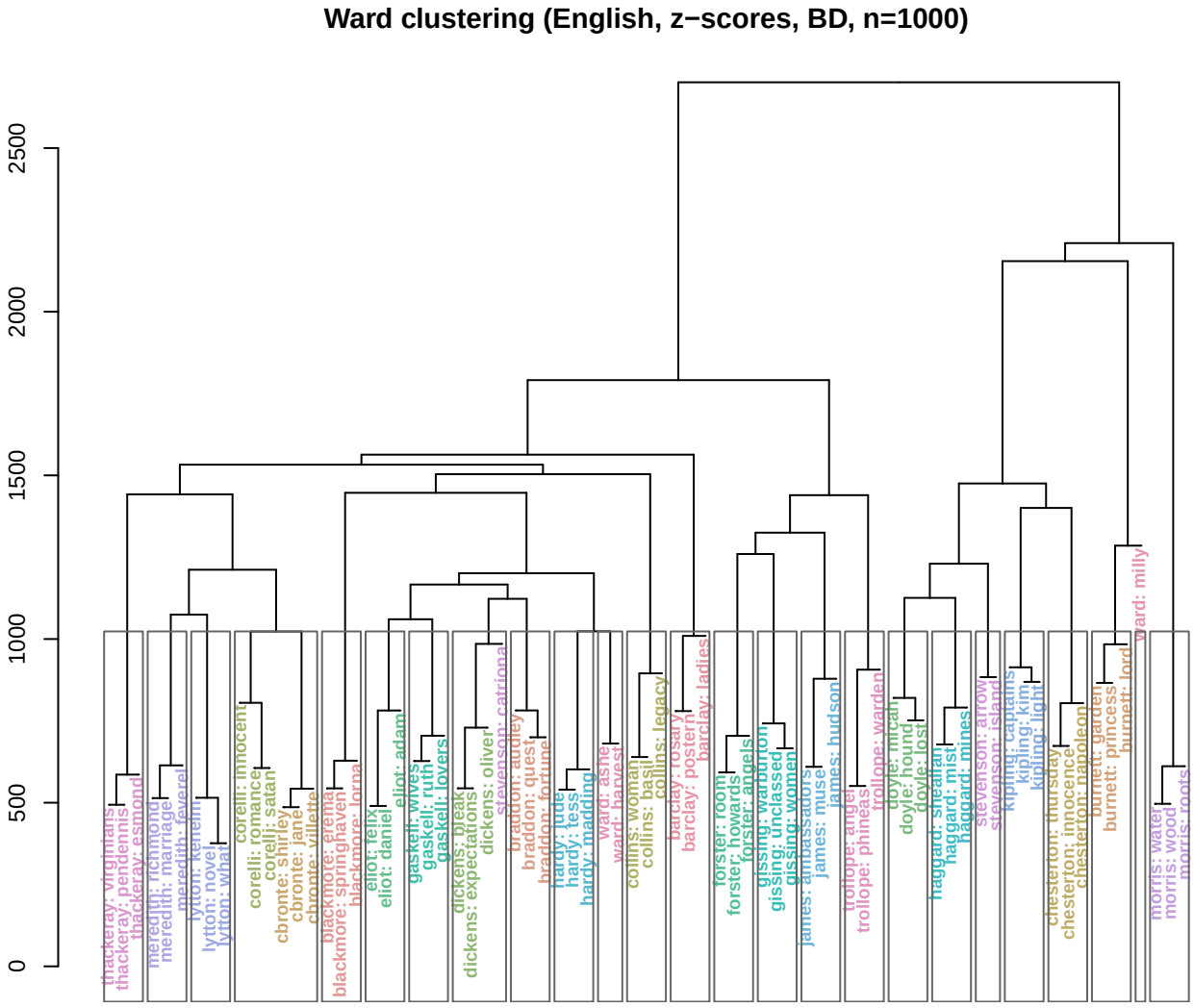
Ward clustering with silhouette (for split into $k = 25$ clusters) and dendrogram:

```
hcl.ward.en <- agnes(DM.en, diss=TRUE, method="ward")
cl.ward.en <- cutree(hcl.ward.en, 25)
do.silhouette(cl.ward.en, DM.en, gold=goldEN, col=colEN, main="Silhouette widths (English, z-scores, BD
```

```
## ARI =  89.6%
```



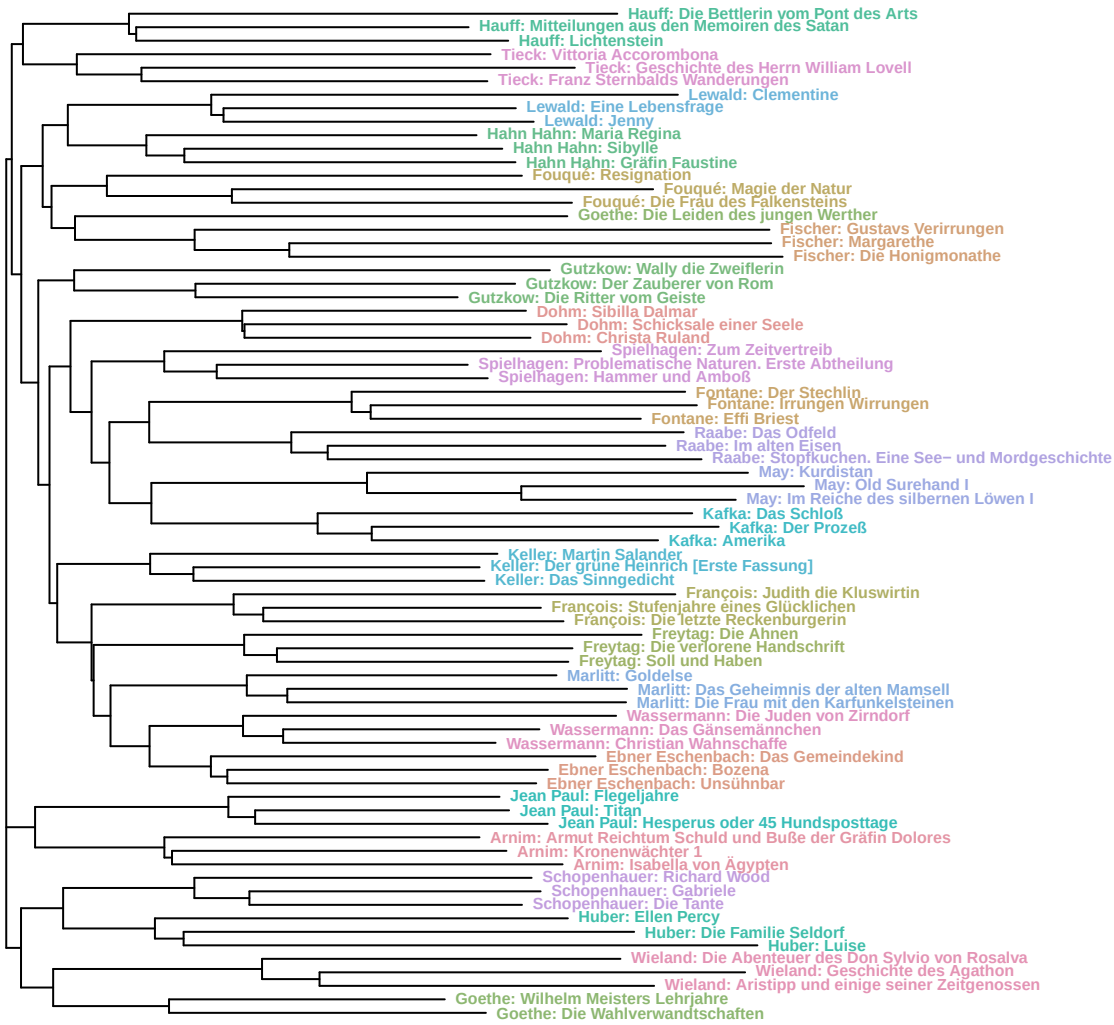

```
do.dendrogram(hcl.ward.en, gold=goldEN, col=colEN, horiz=FALSE, cut=25, main="Ward clustering (English,
```




## 2.2 Alternative clustering methods

Now we can explore some other clustering algorithms. The stylo package often uses neighbor-joining tree estimation (Saitou and Nei 1987) which is popular (if a bit dated) in phylogenetics. The result looks very good – only *Die Leiden des jungen Werther* is completely misplaced – but a quantitative evaluation cannot easily be carried out: the tree cannot be converted to a `hclust` object (because it is not ultrametric) and the `ape` package doesn't seem to have a function for cutting trees into flat clusters.

```
hcl.nj <- nj(as.dist(DM))
plot(hcl.nj, cex=.7, font=2, label.offset=10, tip.color=colDE[hcl.nj$tip.label],
     main="Neighbor-joining tree (German, z-scores, BD, n=1000)")
```




A look at ensemble clustering: use p-value based clustering (Suzuki and Shimodaira 2006), for which a ready-made implementation is available.

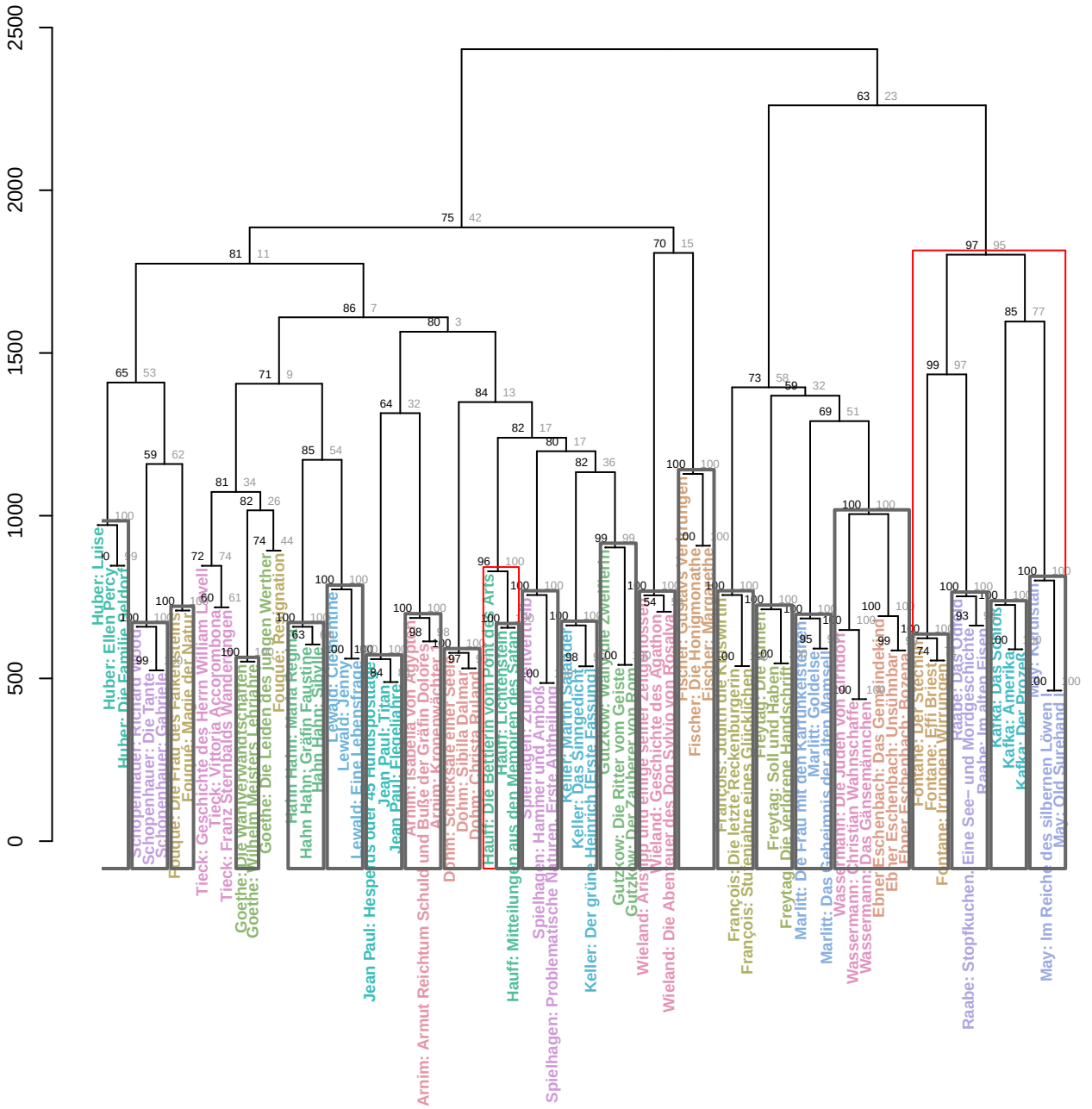
```
res.pvc <- pvclust(t(M), method.dist="manhattan", method.hclust="ward.D2",
                   nboot=1000, store=TRUE, parallel=TRUE)
```

```
## Creating a temporary cluster...done:
## socket cluster with 7 nodes on host 'localhost'
## Multiscale bootstrap... Done.
```

```
hcl.pvc <- res.pvc$hclust
```

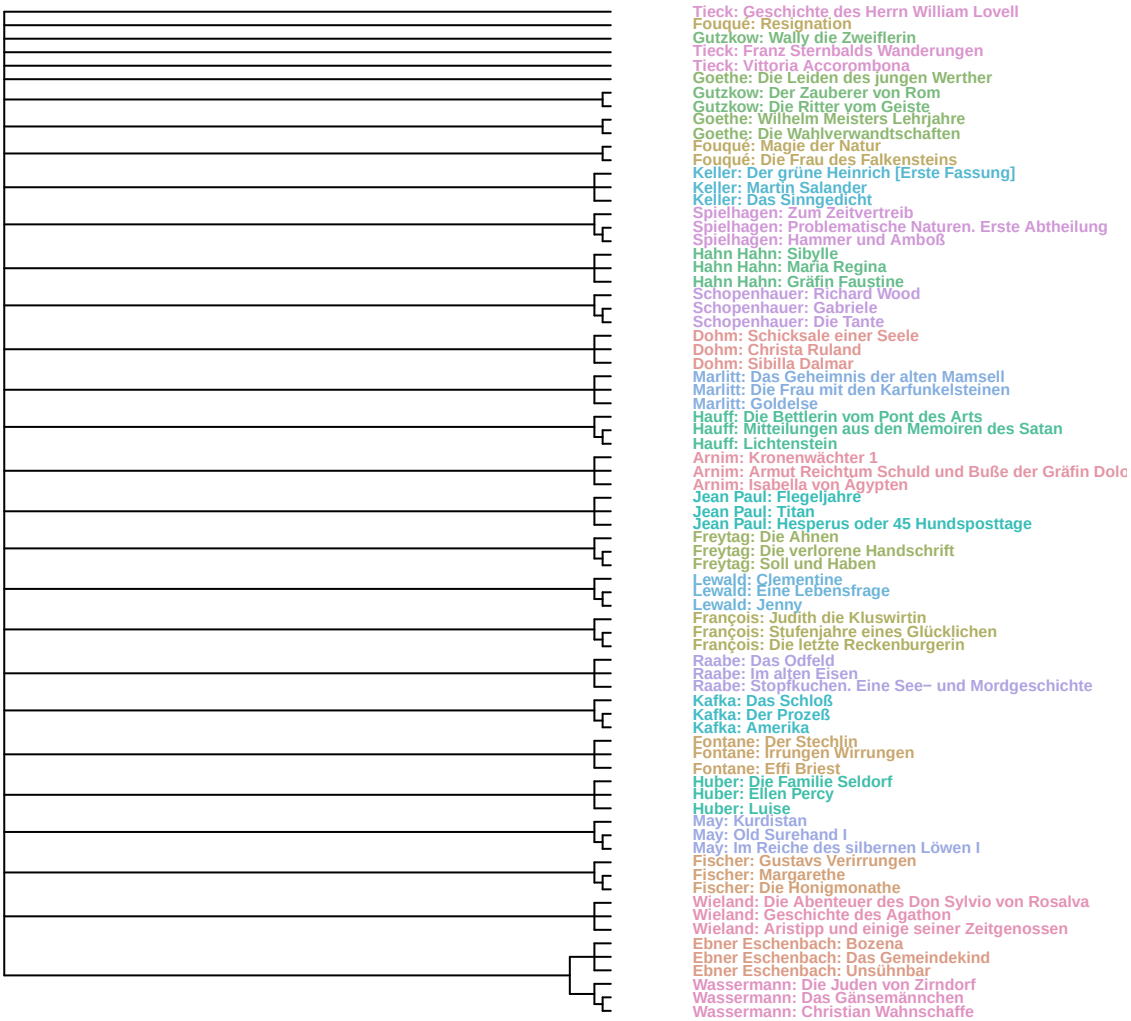
The dendrogram is the standard Ward clustering on the full data set (compare with dendrogram above). Thus, the only advantage of p-value clustering is that it can pick out clusters with a very high probability of being correct. With a confidence threshold of $p > .99$, reliable clusters are usually pure (texts from single author), but results still differ between runs (Wassermann and Ebner are sometimes grouped together). Moreover, a substantial number of texts are not included in any cluster. Note that the default threshold $p > .95$ would put Fontane, Raabe, Kafka and May into a single cluster (thin red box).

```
do.dendrogram(hcl.pvc, horiz=FALSE)
text(res.pvc, cex=.6, col=c("black", "grey60"), print.num=FALSE)
pvrect(res.pvc, alpha=.95, border="red", lwd=1)
pvrect(res.pvc, alpha=.99, border="#666666", lwd=2)
```




Obtain consensus tree from the bootstrapping procedure? Would not be able to obtain a much better clustering from this tree than the PAM / Ward result (because Wassermann and Ebner are still grouped closely together).

```
boot.pvclust <- do.call(c, res.pvc$store) # 10 x 1000 runs of the bootstrapping procedure
boot.pvclust <- lapply(boot.pvclust, as.phylo)
hcl.consensus <- consensus(boot.pvclust, p=0.99)
plot(hcl.consensus, cex=.7, font=2, label.offset=10, tip.color=colDE[hcl.nj$tip.label],
     main="Bootstrapped consensus tree (German, z-scores, BD, n=1000)")
```




**TODO**: The next step is to implement our own bootstrapped clustering ensemble and try to extract a better consensus tree.

Burrows, John. 2002. "'Delta': A Measure of Stylistic Difference and a Guide to Likely Authorship." *Literary and Linguistic Computing* 17 (3): 267–87. doi:10.1093/llc/17.3.267.

Saitou, Naruya, and Masatoshi Nei. 1987. "The Neighbor-Joining Method: A New Method for Reconstructing Phylogenetic Trees." *Molecular Biology and Evolution* 4 (4): 406–25.

Suzuki, Ryota, and Hidetoshi Shimodaira. 2006. "Pvclust: An R Package for Assessing the Uncertainty in Hierarchical Clustering." *Bioinformatics* 22 (12): 1540–2.