SIGIL Unit 07: Multivariate analysis in R

Stephanie Evert

6 March 2023

Contents

1 Data sets & preprocessing

1.1 Preparation

First, load the required packages, various GMA support functions, and the data sets for this unit. Both files should be in the RStudio project directory.

```
source("gma_utils.R")
load("unit7_data.rda", verbose=TRUE)
```

```
## Loading objects:
##   BrownBiber_Matrix
##   BrownBiber_Meta
##   CroCo_Matrix
##   CroCo_Meta
##   CroCo_orig2trans
##   Delta
##   DeltaComplexity
##   DeltaLemma
##   MultiVar_Matrix
##   SyntacticComplexity_Matrix
##   SyntacticComplexity_Meta
```

as well as further optional packages if you want to follow all steps in this tutorial:

```
library(cluster)
library(Hotelling)
library(ellipse)
library(e1071)
library(Rtsne)
```

If you have RGL installed, you can also display 3D graphics in an interactive session. We use the option `eval=FALSE` on the code chunk below (and all other code chunks with 3D visualizations) so that it isn't executed when rendering the full document offline.

```
library(rgl)
```

1.2 A small example matrix

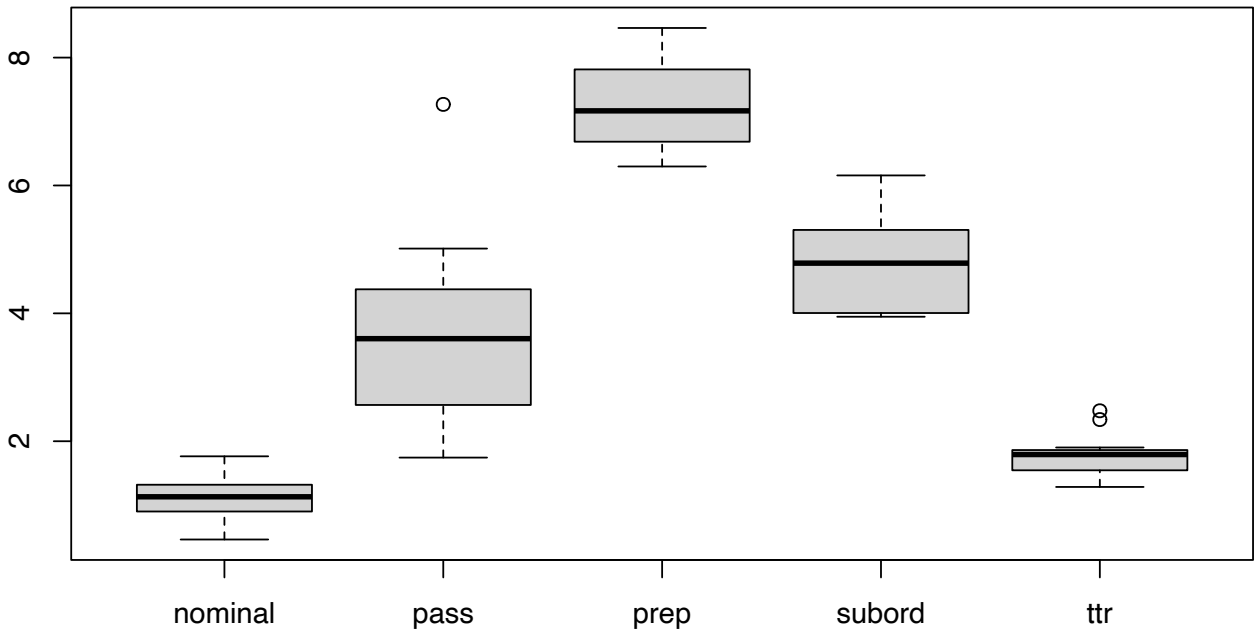
The data file includes a small example feature matrix which we can use to understand various analysis techniques.

```
knitr::kable(MultiVar_Matrix)
```

	nominal	pass	prep	subord	ttr
orig-1	1.205	5.013	6.883	4.483	1.285
orig-2	0.738	2.537	6.486	6.157	1.714
orig-3	1.252	4.462	8.463	4.785	2.476
orig-4	1.105	2.899	8.119	3.966	1.519
orig-5	1.764	4.268	7.167	3.947	1.792
orig-8	1.545	7.268	7.461	5.455	1.572
trans-1	0.463	2.208	6.297	6.089	2.339
trans-2	1.131	2.597	6.307	4.844	1.810
trans-4	0.935	1.744	7.098	4.012	1.403
trans-5	0.867	3.604	7.511	5.154	1.902
trans-7	1.387	4.290	8.211	3.998	1.822

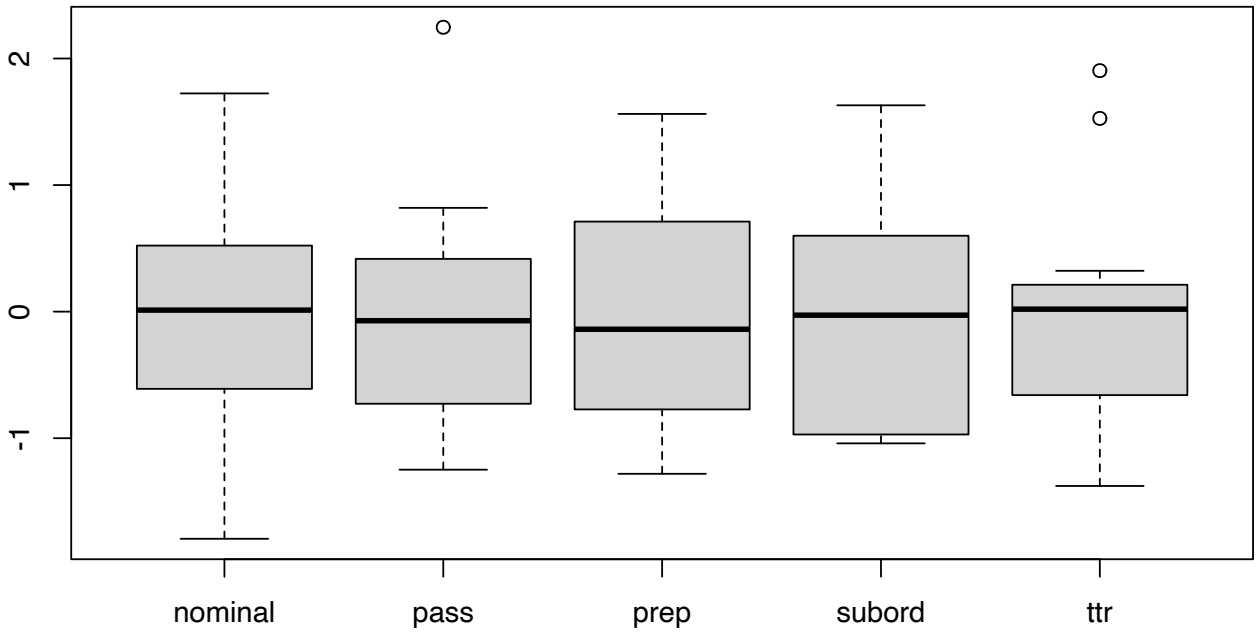
We rename this matrix M for convenience and use a boxplot to assess the ranges and distributions of the individual features.

```
M <- MultiVar_Matrix
boxplot(M)
```



Since the ranges are considerably different, we compute standardized z-scores (Z) ensuring equal contribution of all features to the Euclidean distances.

```
Z <- scale(M)
boxplot(Z)
```



1.3 The Biber data sets: reproducing MDA

Biber features for all texts of the British National Corpus — as shown in the overview talk — are included in the **corpora** package (see `?BNCbiber` for details). Alternatively, you can work with Biber features for the Brown Family corpora computed by Andrea Nini's MAT (Multidimensional Analysis Tagger), using the objects `BrownBiber_Matrix` and `BrownBiber_Meta` loaded above.

Again, we assign the feature matrix and metadata table to shorter names (the suffix `B` stands for *Biber*).

```
MB <- BNCbiber
MetaB <- BNCmeta
```

Get an overview of the available metadata:

```
View(MetaB) # evaluate manually in interactive mode only
```

```
colnames(MetaB)
```

```
##  [1] "id"                "title"             "n_words"           "n_tokens"
##  [5] "n_w"               "n_c"               "n_s"               "publication_date"
##  [9] "text_type"         "context"           "respondent_age"    "respondent_class"
## [13] "respondent_sex"    "interaction_type"  "region"            "author_age"
## [17] "author_domicile"   "author_sex"        "author_type"       "audience_age"
## [21] "domain"            "difficulty"        "medium"            "publication_place"
## [25] "sampling_type"     "circulation"       "audience_sex"      "availability"
## [29] "mode"              "derived_type"      "genre"
```

As in the overview talk, we use `derived_type` as a text type classification and `author_sex` for male vs. female authors.

```
table(MetaB$derived_type)
```

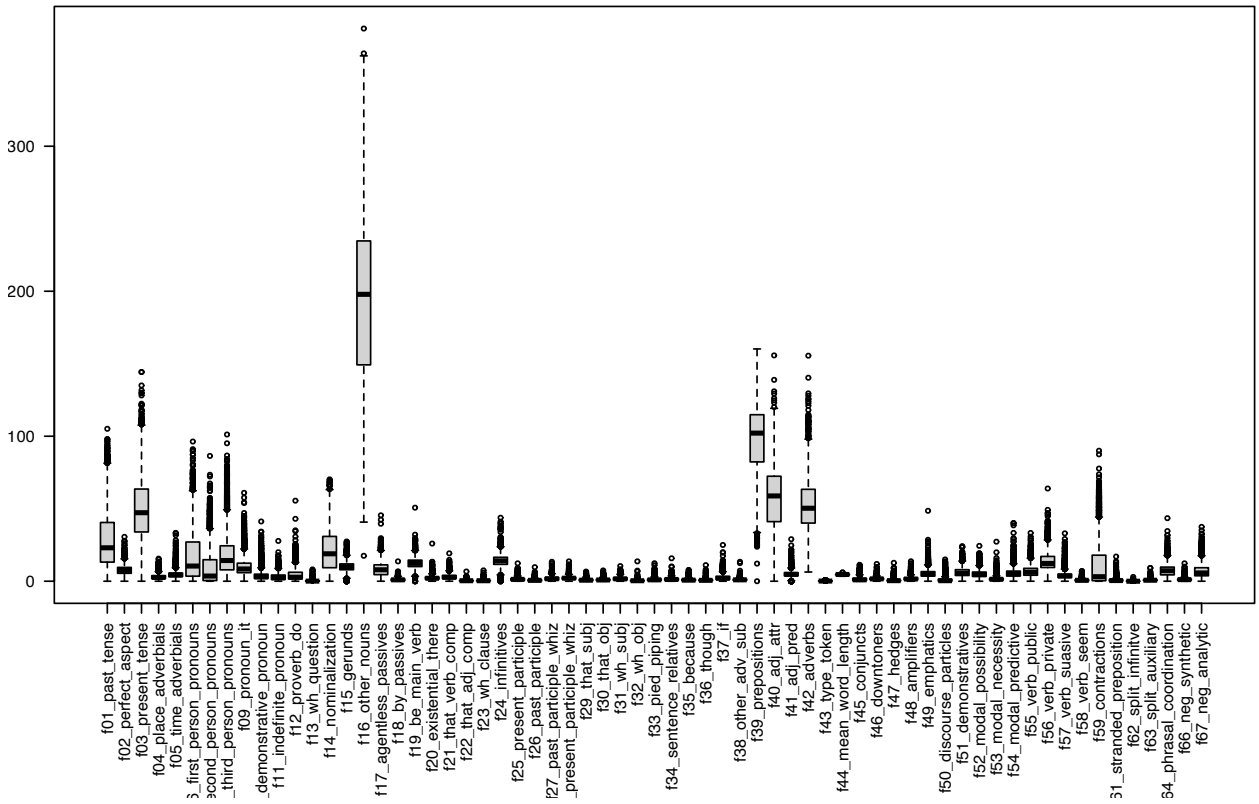
```
##
##            academic             fiction      misc_published           newspaper
##                 497                 452                 710                 486
##               prose spoken_conversation        spoken_other         unpublished
##                 744                 153                 755                 251
```

```
table(MetaB$author_sex)
```

```
##
##     ---  female    male   mixed unknown
##     908     414     920     234    1572
```

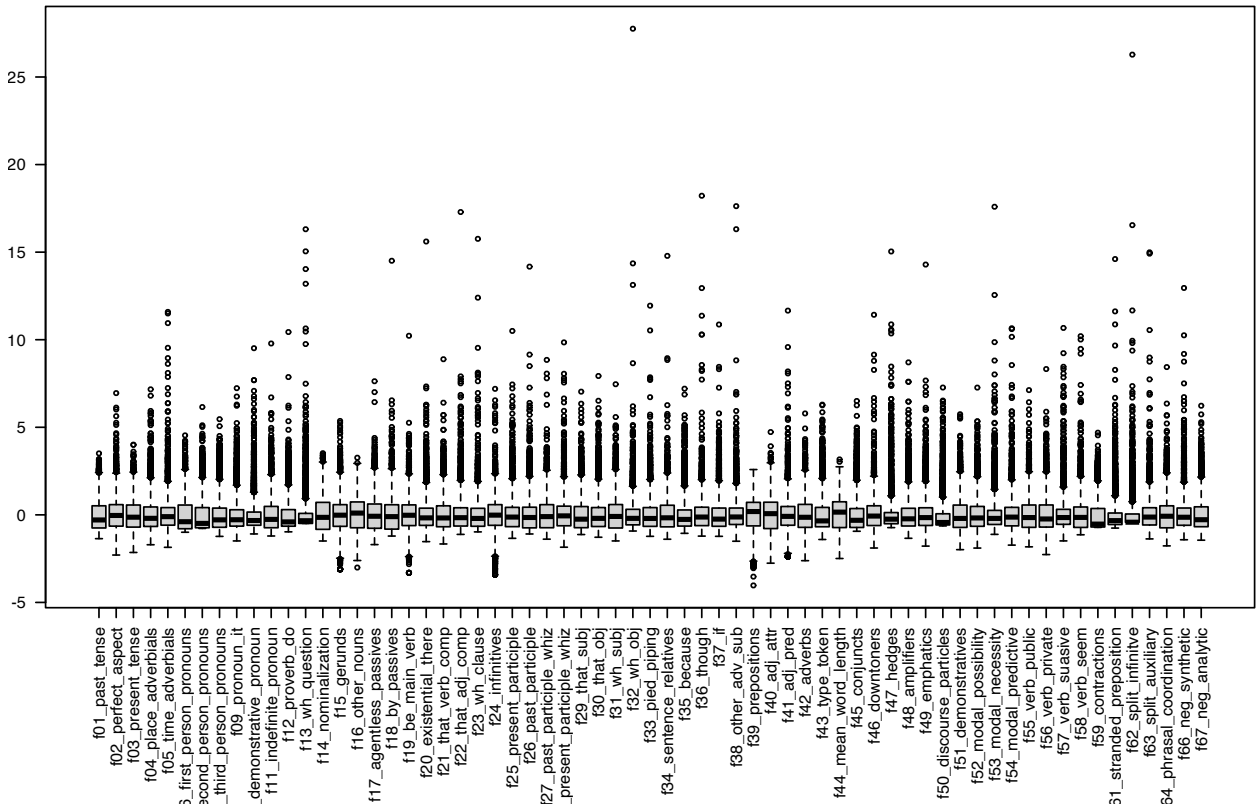
Relative frequency counts per 1000 words of texts are on wildly different scales for different features, of course:

```
par(mar=c(10,4,1,1), cex=.6)
boxplot(MB, las=2)
```



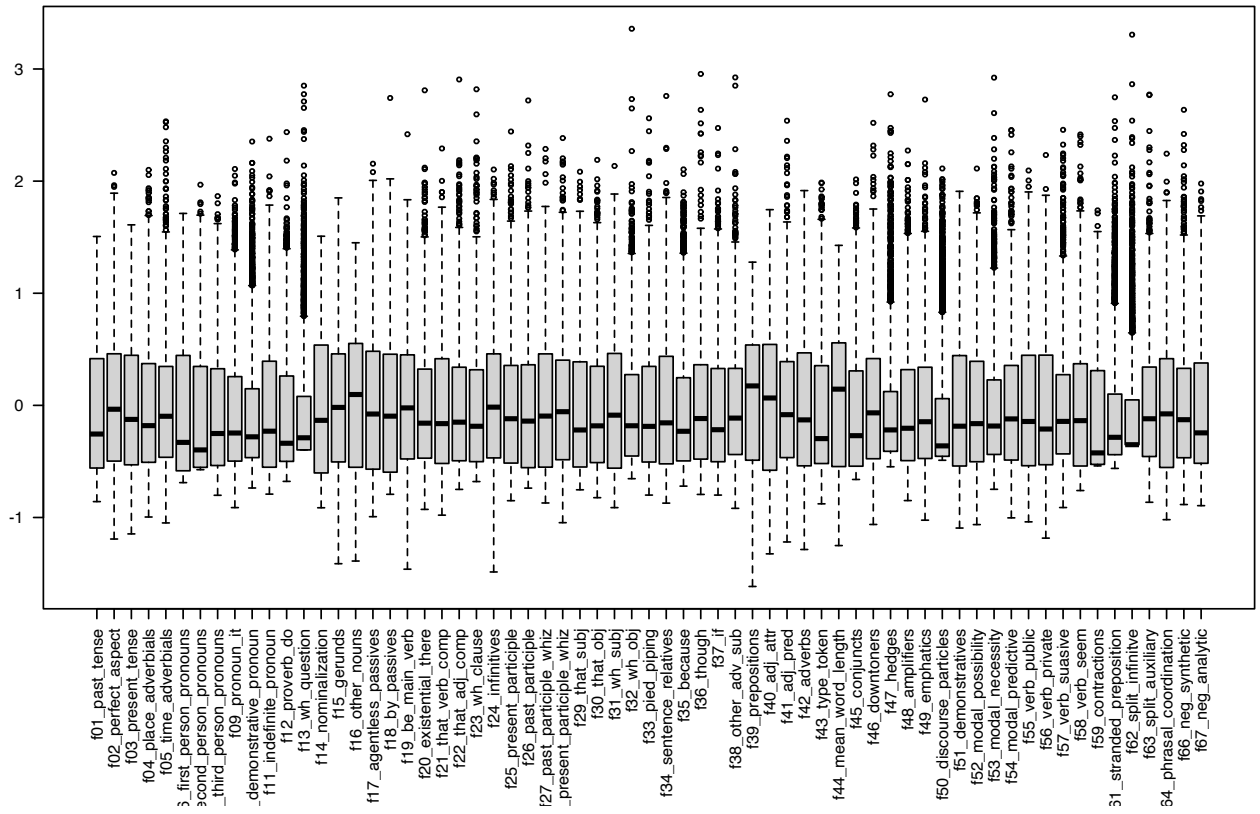
Standardisation is therefore essential (and sometimes implicit in multivariate analysis techniques). However, the distributions are still highly skewed with many outlier scores ($\gg 3$ standard deviations).

```
ZB <- scale(MB)
par(mar=c(10,4,1,1), cex=.6)
boxplot(ZB, las=2)
```



A logarithmic transformation often helps to control the influence of outlier values in such cases, but it is an empirical heuristic without mathematical justification.

```
ZLB <- signed.log(ZB)
par(mar=c(10,4,1,1), cex=.6)
boxplot(ZLB, las=2)
```



1.4 Authorship attribution with Delta measures

One of the most successful and popular approaches to literary authorship attribution, known as the **Delta method**, compares frequency profiles of the most frequent words (**mfw**) in a text collection, typically focusing on the range between 200 and 5000 mfw. Here, we will try this approach on a data set of 75 English novels from 25 different authors.

`Delta$EN` is a matrix of absolute frequency counts of unnormalised word forms in the novels, sorted by mfw. Feel free to work with the corresponding data sets for French (`Delta$FR`) and German (`Delta$DE`) as well (hint: German gives the best authorship attribution results :–).

```
head(Delta$EN, 7) # shows top left corner of sparse matrix
```

```
## 7 x 7 sparse Matrix of class "dgCMatrix"
##                   ,   the    .   and   to   of    a
## delta_en_001 11733  8251 6010  3232 3418 3590 1941
## delta_en_002  3318  2116 2253  1019 1146  879  739
## delta_en_003  8533  4679 5566  3477 2847 2594 1940
## delta_en_004 14576  6417 6677  5523 5159 4895 3422
## delta_en_005 25912 12026 7836 10949 7971 8144 5315
## delta_en_006 17270  9961 7784  6530 5796 6527 4563
## delta_en_007 11109  7551 6062  4557 3975 4081 3201
```

After selecting the number of most frequent words as features, we convert the sparse term-document matrix into a regular R matrix and compute relative word frequencies. Here, we will work with up to $n = 2000$ mfw, which should give us fairly good authorship attribution results. The short variable names for this data set use suffix `A` (for *authorship attribution*).

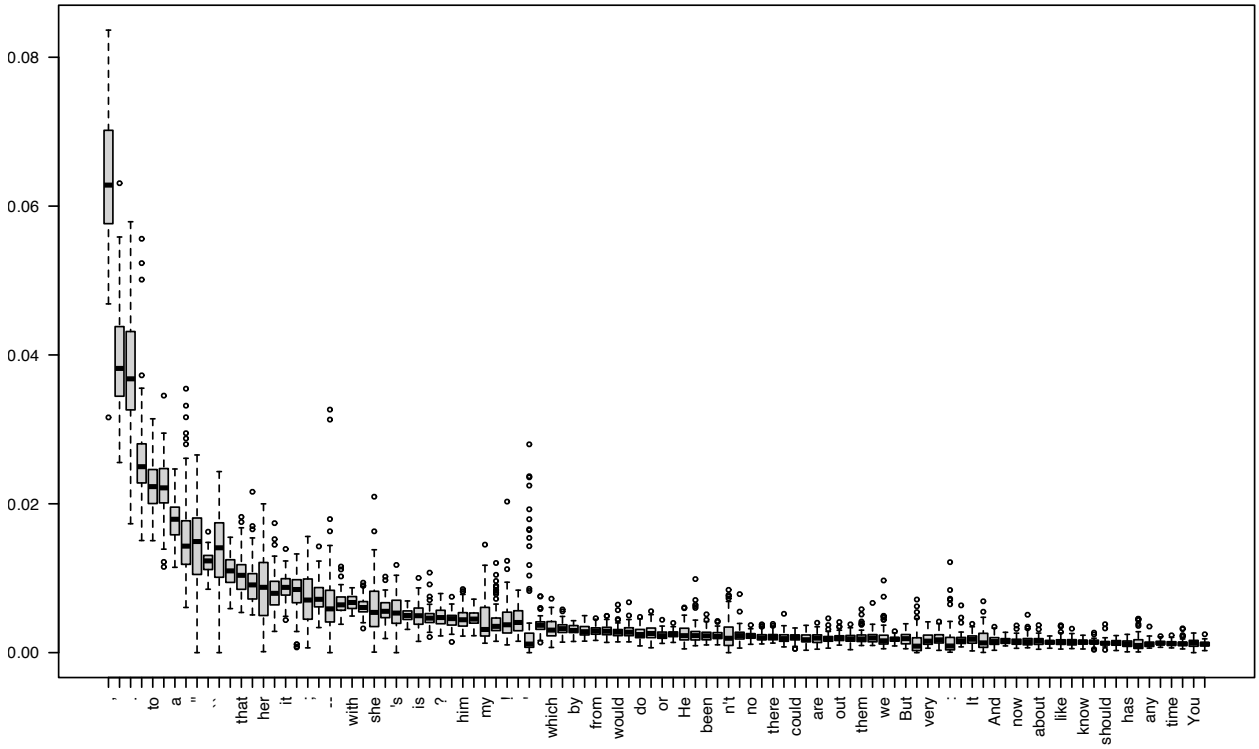
```
MA <- as.matrix(Delta$EN$M[, 1:2000])
MetaA <- Delta$EN$rows
```

```
text.sizes <- MetaA$f # number of tokens in each text
MA <- scaleMargins(MA, rows = 1 / text.sizes) # relative frequency
MA[1:7, 1:7] # head() would show full rows for dense matrix
```

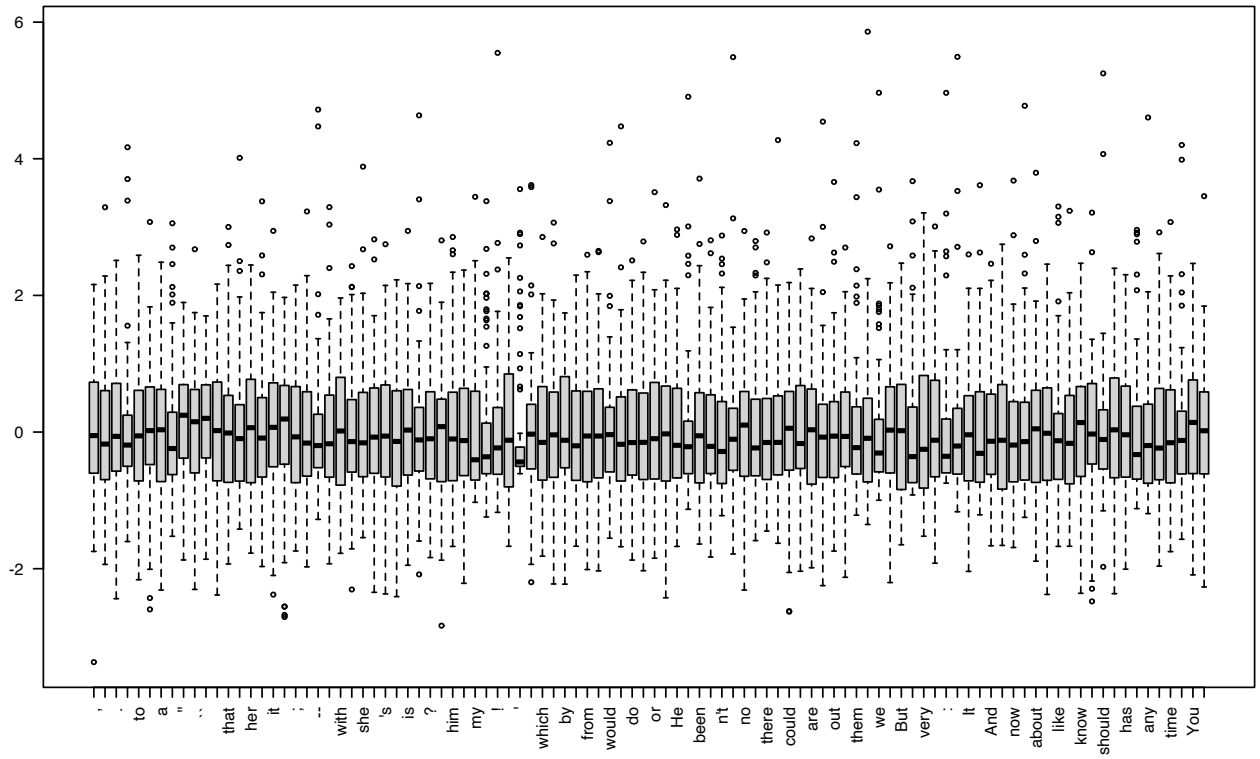
```
##                         ,        the          .        and         to         of          a
## delta_en_001 0.07941601 0.05584773 0.04067930 0.02187612 0.02313508 0.02429928 0.01313786
## delta_en_002 0.06731589 0.04292960 0.04570907 0.02067356 0.02325015 0.01783323 0.01499290
## delta_en_003 0.06635819 0.03638697 0.04328486 0.02703943 0.02214014 0.02017264 0.01508671
## delta_en_004 0.07386012 0.03251649 0.03383398 0.02798638 0.02614190 0.02480415 0.01734010
## delta_en_005 0.08110680 0.03764242 0.02452736 0.03427132 0.02494992 0.02549142 0.01663641
## delta_en_006 0.07259200 0.04186965 0.03271894 0.02744793 0.02436267 0.02743532 0.01917992
## delta_en_007 0.06282163 0.04270106 0.03428074 0.02576993 0.02247871 0.02307814 0.01810172
```

A boxplot even for just the first 100 mfw shows why standardisation is absolutely essential in this case.

```
par(mar=c(6,4,1,1), cex=.6)
boxplot(MA[, 1:100], las=2)
```




```
ZA <- scale(MA)
par(mar=c(6,4,1,1), cex=.6)
boxplot(ZA[, 1:100], las=2)
```



Since `Delta$EN` is a **wordspace** DSM object, you can carry out all these steps and even compute z-scores with the `dsm.score` function:

```
ZA <- dsm.score(subset(Delta$EN, select=(1:2000)),
                score = function(f, f1, ...) 1000 * f / f1,
                scale="standardize", negative.ok=TRUE, matrix.only=TRUE)
```

Metadata for the novels are integrated into the DSM object:

```
MetaA <- Delta$EN$rows
table(MetaA$author)
```

```
##
##    Barclay  Blackmore    Braddon    Burnett    Cbronte Chesterton    Collins    Corelli
##          3          3          3          3          3          3          3          3
##    Dickens      Doyle      Eliot    Forster    Gaskell    Gissing    Haggard      Hardy
##          3          3          3          3          3          3          3          3
##      James    Kipling     Lytton   Meredith     Morris  Stevenson  Thackeray   Trollope
##          3          3          3          3          3          3          3          3
##       Ward
##          3
```

2 Distances and visualization

One approach to the data-driven (unsupervised) analysis of multivariate data sets is to compute distances between data points $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ in the feature space. Such distances represent the (dis)similarity of the corresponding items according to the chosen feature set.

A widely-used distance measure is **Euclidean distance** $d_2(\mathbf{x}, \mathbf{y})$, which corresponds to our geometric intuition and is compatible with other geometric notions such as angles and orthogonality.

```
DM <- dist(Z)
round(DM, 2)
```

```
##         orig-1 orig-2 orig-3 orig-4 orig-5 orig-8 trans-1 trans-2 trans-4 trans-5
## orig-2    3.13
## orig-3    3.92   4.16
## orig-4    2.30   3.60   3.05
## orig-5    2.24   4.11   3.07   2.46
## orig-8    2.34   4.03   3.52   3.61   2.79
## trans-1   4.47   1.91   4.16   4.54   4.94   5.12
## trans-2   2.29   1.94   3.59   2.73   2.55   3.64    2.78
## trans-4   2.31   2.94   4.06   1.63   2.96   4.29    3.97    1.98
## trans-5   2.42   2.03   2.38   2.10   2.91   3.11    2.69    1.90    2.35
## trans-7   2.45   4.04   2.10   1.44   1.71   2.88    4.77    2.99    2.75    2.24
```

R's built-in `dist()` function can also compute other metrics such as Manhattan distance $d_1(\mathbf{x}, \mathbf{y})$. This is the traditional metric used in authorship attribution with the Delta method.

```
DM1 <- dist(Z, method = "manhattan")
round(DM1, 2)
```

```
##         orig-1 orig-2 orig-3 orig-4 orig-5 orig-8 trans-1 trans-2 trans-4 trans-5
## orig-2    6.56
## orig-3    6.20   8.96
## orig-4    4.50   6.55   5.46
## orig-5    4.40   7.65   6.11   4.67
## orig-8    5.07   7.70   7.18   6.76   5.31
## trans-1   9.40   3.00   8.36   9.39  10.06  10.54
## trans-2   4.37   3.19   6.24   4.50   5.03   6.98    5.03
## trans-4   3.98   5.29   8.26   2.91   5.08   7.83    7.71    4.24
## trans-5   5.14   4.10   4.86   4.38   5.06   5.49    5.90    3.56    4.66
## trans-7   4.76   8.04   3.56   2.64   2.55   5.75   10.28    5.32    5.47    4.38
```

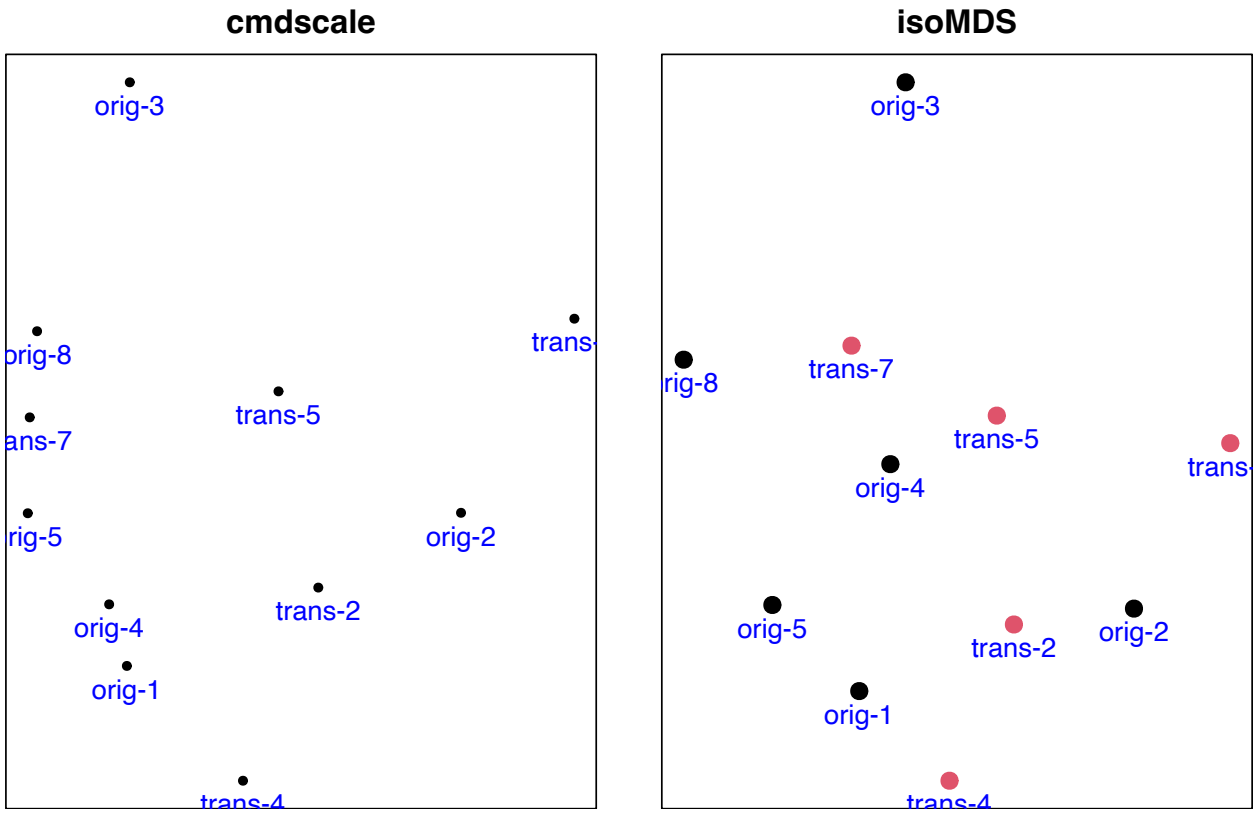
Function `dist.matrix()` from the **wordspace** package offers a wider range of distance and similarity measures (and is more efficient for large matrices).

2.1 High-dimensional visualisation

We cannot directly visualise high-dimensional data sets as scatterplots (a full scatterplot matrix for the Delta data set would consist of almost 2 million panels). A common visualisation approach is to create a low-dimensional approximation of the geometric structure of a data set based on the distance matrix.

A classical technique used for this purpose is **multidimensional scaling** (MDS), implemented in R functions `cmdscale()` as well as `isoMDS()` and `sammon()` from the **MASS** package.

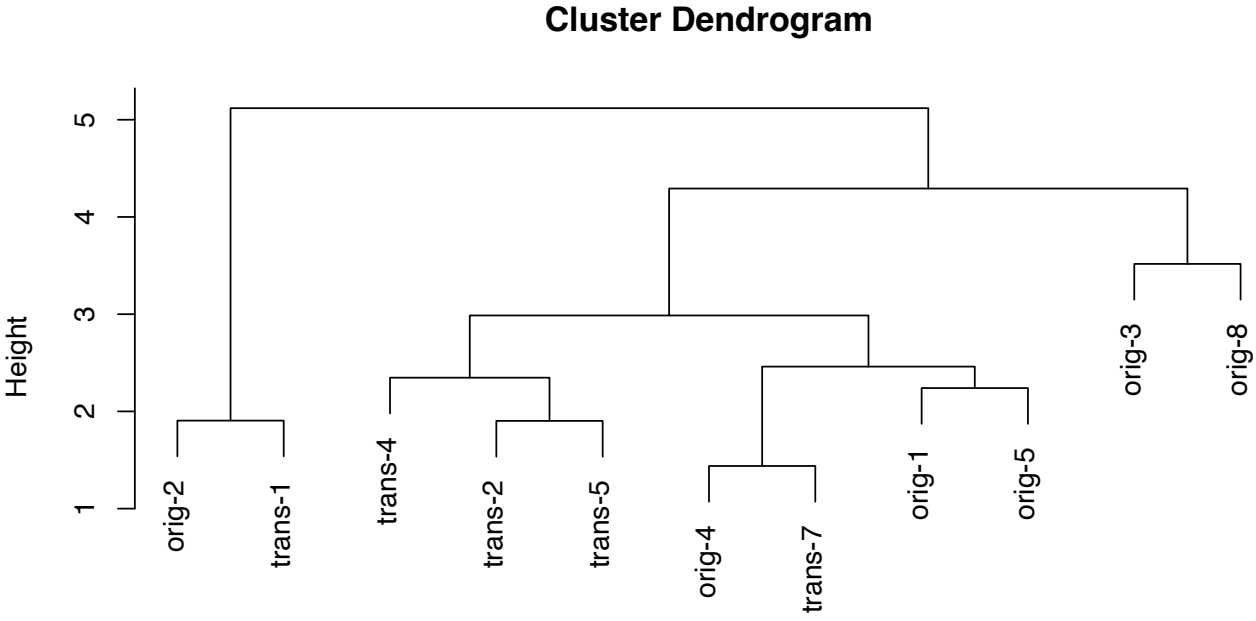
```
par(mfrow=c(1, 2), mar=c(0, 1, 2, 1), xaxt="n", yaxt="n")

coord1 <- cmdscale(DM)
plot(coord1, pch=20, xlab="", ylab="", main="cmdscale")
text(coord1, labels=rownames(M), pos=1, col="blue")

coord2 <- isoMDS(DM, trace=FALSE)$points
cat.vec <- as.factor(substr(rownames(M), 1, 5))
plot(coord2, pch=20, cex=2, col=as.integer(cat.vec), xlab="", ylab="", main="isoMDS")
text(coord2, labels=rownames(M), pos=1, col="blue")
```



We will take a closer look at these techniques and more sophisticated visualisation methods later on.

Hierarchical **clustering** incrementally groups together the two most similar data points or sub-clusters. It can be visualized in the form of a dendrogram where the height of each subtree corresponds to the "size" of the corresponding cluster.

```
clusters <- hclust(DM, method = "complete")
plot(clusters, xlab="", sub="")
```

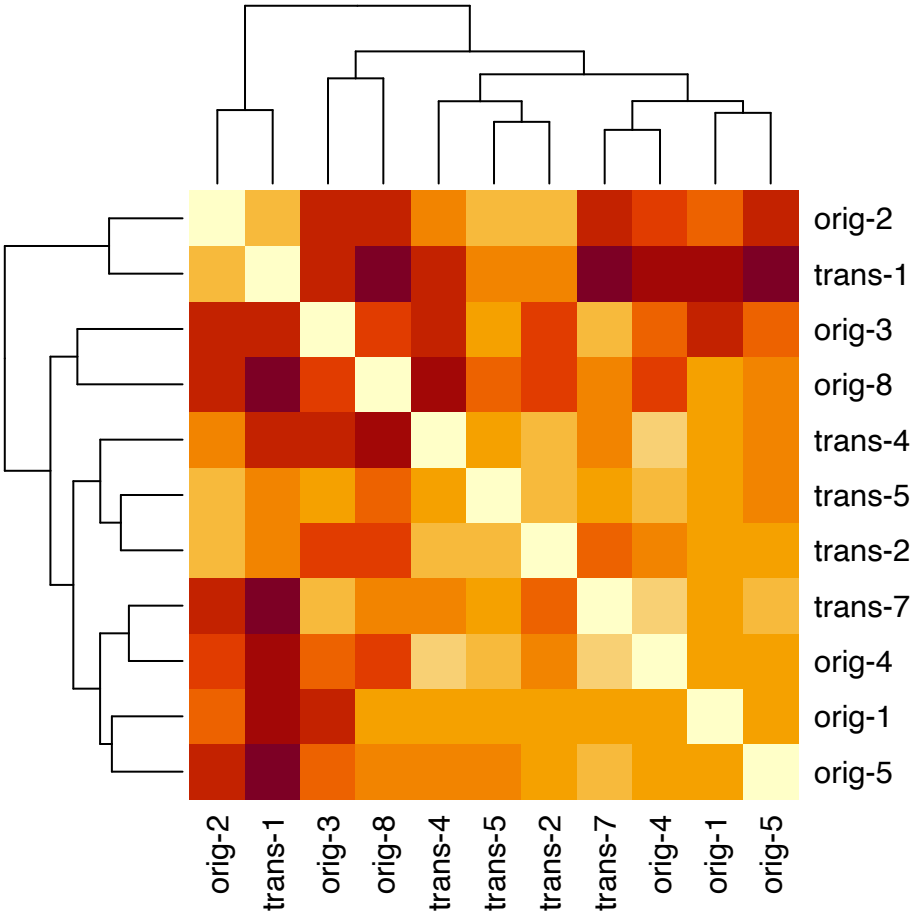


Q: Try different clustering algorithms (`method` argument) and different distance metrics for `DM`.

Do you get substantially different groupings of the texts?

An alternative is to visualise the distance matrix directly in the form of a **heatmap**. The `heatmap()` function automatically applies a form of clustering to group together similar data points, making the visualisation much more easily interpretable, especially for large data sets.

```
heatmap(as.matrix(DM), symm=TRUE)
```



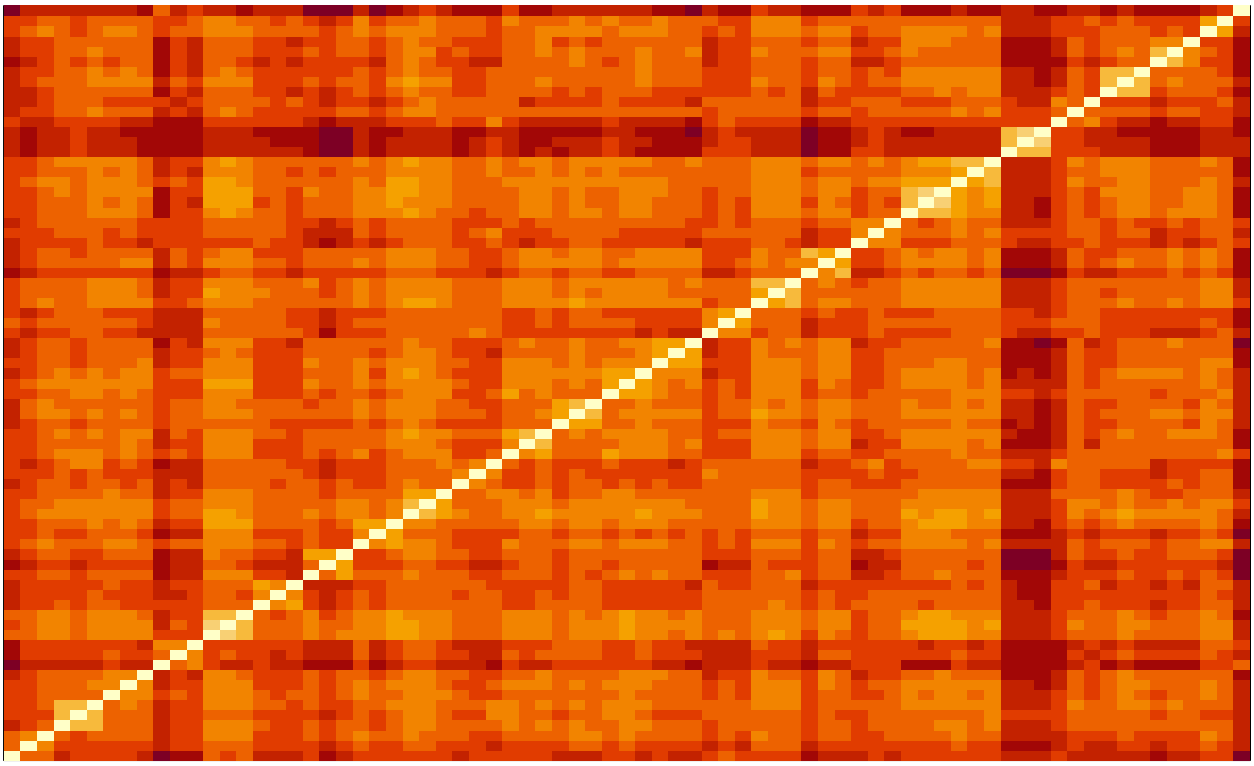
3 Authorship attribution

Clustering and low-dimensional visualisation of distances should work well for authorship attribution tasks because we expect texts from the same author to group together.

3.1 Clustering

Following Burrows's original algorithm, we use the Manhattan metric rather than Euclidean distances, even though the standardized features will no longer have the same weight. You will be asked later on to try several other distance metrics. The distance table is too large to print, but can be visualized as a heatmap. Note that the texts are sorted by author – can you see corresponding groups in the plot?

```
DMA <- dist(ZA, method="manhattan") # Burrows Delta
par(mar=c(1,1,1,1), xaxt="n", yaxt="n")
image(as.matrix(DMA))
```



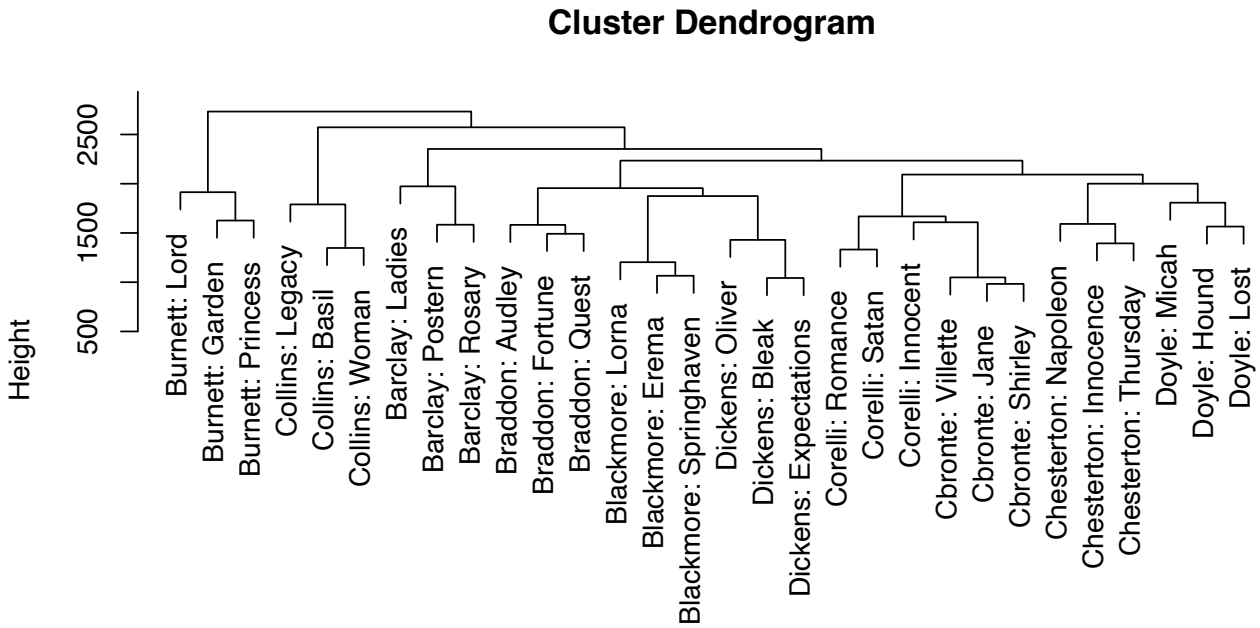
For now, we will only look at the first 10 authors so the clustering dendrogram is readable. Note how we use `droplevels()` to tell R to forget about the authors no longer included in the data set.

```
ZA10 <- ZA[1:30, ]
DMA10 <- dist(ZA10, method="manhattan")
MetaA10 <- droplevels(MetaA[1:30, ])
```

Q: Can you figure out how to take a *random* sample of 10 authors?

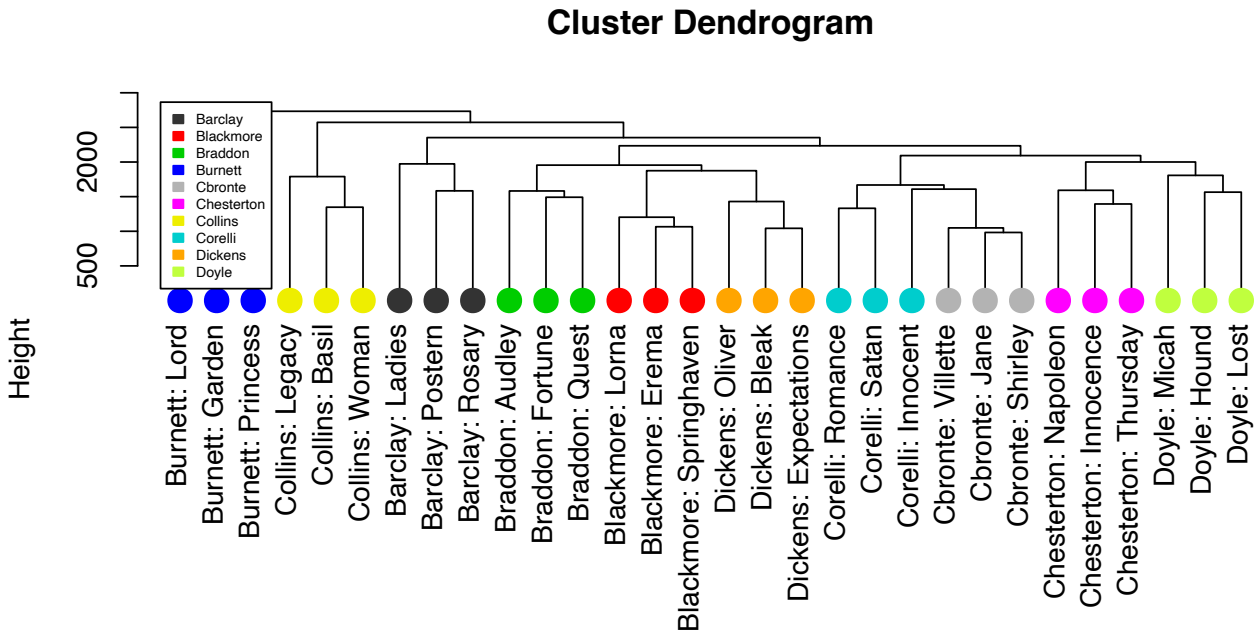
Compute a hierarchical clustering of the texts and plot the dendrogram:

```
clusters <- hclust(DMA10) # see ?hclust for the default method
plot(clusters, xlab="", sub="", labels=MetaA10$label)
```



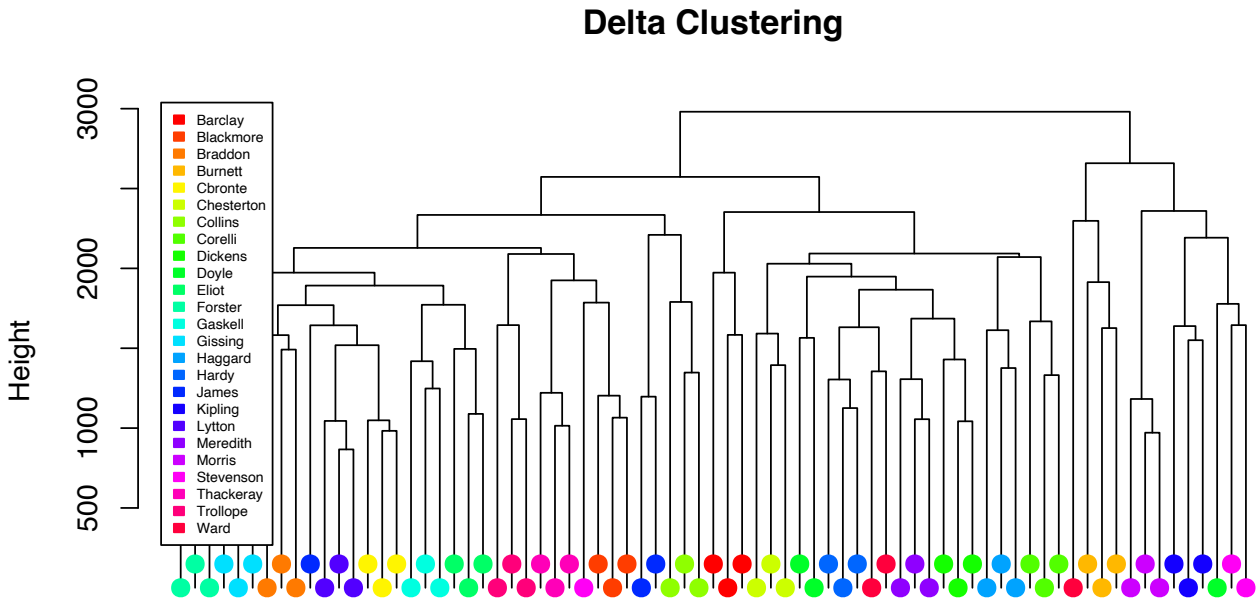
This looks promising, but it's difficult to see whether novels from the same author always cluster together. The `gma.clust()` function provides a colour-coded visualization. It takes the standardized feature matrix as input and carries out the clustering itself, controlled by options `method`, `metric` and `p`.

```
gma.clust(clusters,
          labels=MetaA10$label, col=MetaA10$author,
          cex=2, legend.cex=.5)
```



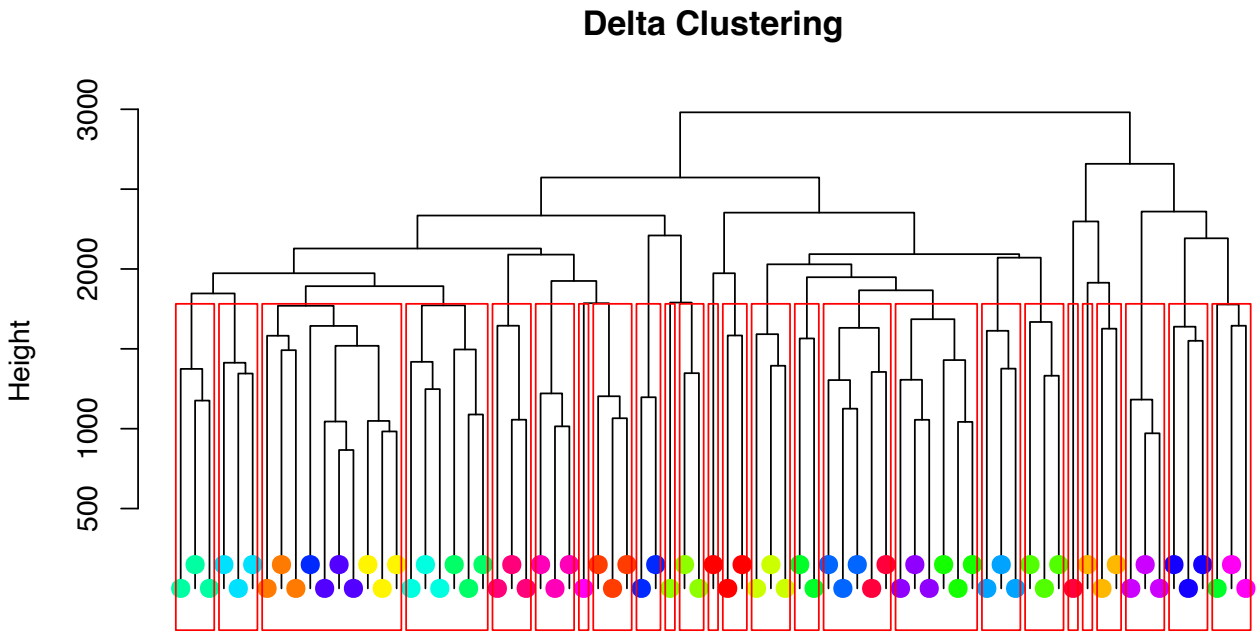
In order to colour-code all 25 authors, we need a suitable colour palette. If the dendrogram becomes too crowded, we can omit the labels and stagger the indicator dots using the `period` and `spread` options

```
clusters <- hclust(DMA)
gma.clust(clusters, labels=NULL,
          col=MetaA$author, col.vals=rainbow(25),
          period=2, cex=1.5, legend.cex=.5, main="Delta Clustering")
```



Since we know that there are exactly 25 authors in our data set, it makes sense to split the hierarchical clustering into 25 flat clusters by cutting the dendrogram at a suitable height. We need to re-run `gma.clust()` because each code chunk is evaluated on its own and cannot add to a previous plot.

```
gma.clust(clusters, labels=NULL,
          col=MetaA$author, col.vals=rainbow(25),
          period=2, cex=1.5, legend=FALSE, main="Delta Clustering")
rect.hclust(clusters, k=25, border="red") # visualization of the cut
```




```
cluster.id <- cutree(clusters, k=25)
```

The cluster IDs can then be compared with the "gold standard" authors. A simple approach is to label each cluster with the most frequent author.

```
gold <- as.character(MetaA$author) # gold standard authors as strings
predicted <- majorityLabels(cluster.id, gold) # authors assigned by majority labelling
```

```
rbind(predicted, cluster.id, gold)[, 1:30]
```

```
##            1         2         2         3           3           3           4
## predicted  "Barclay" "Barclay" "Barclay" "Blackmore" "Blackmore" "Blackmore" "Braddon"
## cluster.id "1"       "2"       "2"       "3"         "3"         "3"         "4"
## gold       "Barclay" "Barclay" "Barclay" "Blackmore" "Blackmore" "Blackmore" "Braddon"
##            4         4         5         6         5         4         4         4
## predicted  "Braddon" "Braddon" "Burnett" "Burnett" "Burnett" "Braddon" "Braddon" "Braddon"
## cluster.id "4"       "4"       "5"       "6"       "5"       "4"       "4"       "4"
## gold       "Braddon" "Braddon" "Burnett" "Burnett" "Burnett" "Cbronte" "Cbronte" "Cbronte"
##            7           7           7           8         9         8         10
## predicted  "Chesterton" "Chesterton" "Chesterton" "Collins" "Collins" "Collins" "Corelli"
## cluster.id "7"         "7"         "7"         "8"       "9"       "8"       "10"
## gold       "Chesterton" "Chesterton" "Chesterton" "Collins" "Collins" "Collins" "Corelli"
##            10        10        11        11        11        12      12      13
## predicted  "Corelli" "Corelli" "Dickens" "Dickens" "Dickens" "Doyle" "Doyle" "Stevenson"
## cluster.id "10"      "10"      "11"      "11"      "11"      "12"    "12"    "13"
## gold       "Corelli" "Corelli" "Dickens" "Dickens" "Dickens" "Doyle" "Doyle" "Doyle"
```

An intuitive quantitative measure of the clustering quality is *purity*, i.e. the proportion of novels assigned to the correct author by the majority labels.

```
n.correct <- sum(predicted == gold)
round(100 * n.correct / length(gold), 2) # in percent
```

```
## [1] 78.67
```

> **Q:** Can you explain why clustering purity is "optimistic", i.e. it will report a higher quality than is actually achieved?

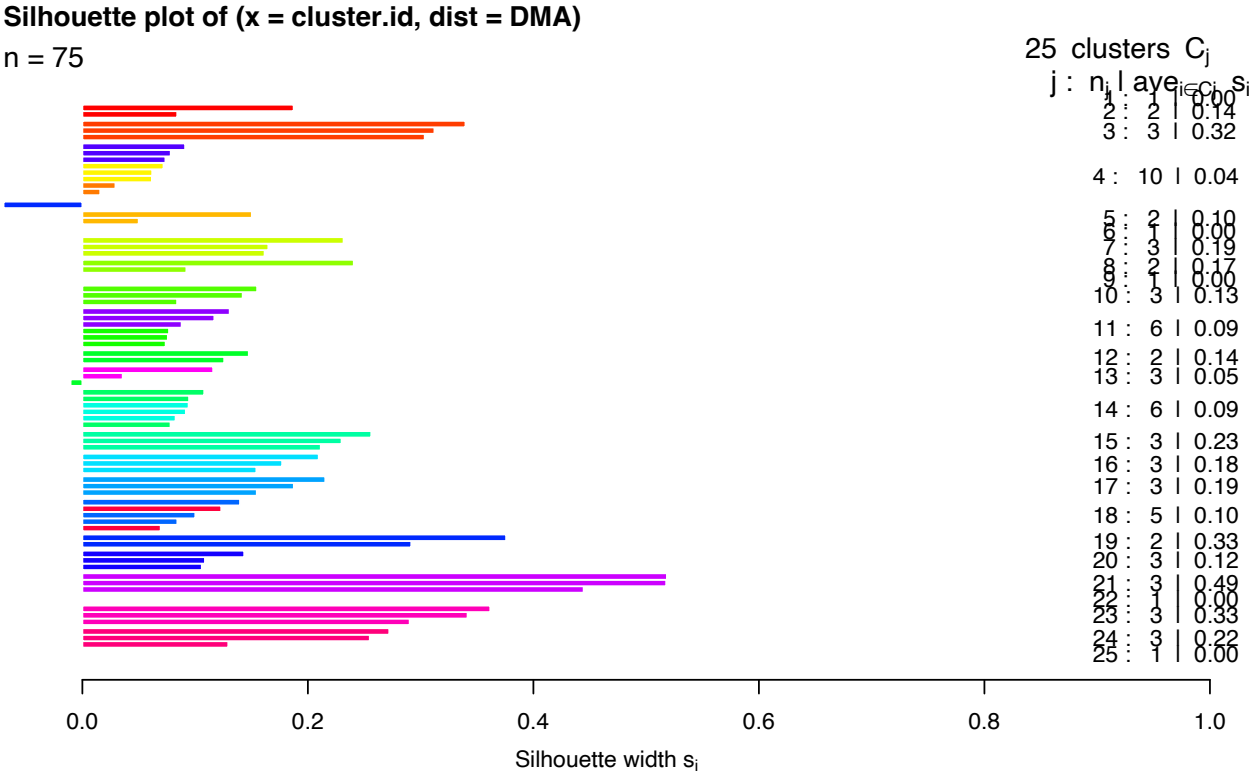
A better quantitative measure is the adjusted Rand index (ARI), used e.g. by Evert *et al.* (2017) for the evaluation of authorship attribution tasks.

```
adjustedRandIndex(cluster.id, MetaA$author)
```

```
## [1] 0.6028249
```

The **cluster** package can compute silhouette widths as an indicator of clustering quality. Colouring the bars by author is a bit tricky, though. Note that bars for points with a silhouette width of 0 (e.g. for all single-point clusters) are not visible.

```
sil <- silhouette(cluster.id, DMA)
rownames(sil) <- MetaA$author
sil <- sortSilhouette(sil)
col.map <- rainbow(25)
names(col.map) <- levels(MetaA$author)
par(cex=.8)
plot(sil, col=col.map[rownames(sil)])
```



Average silhouette width : 0.15

Q: If a certain number of clusters are desired, it is often better to compute such a "flat" clustering directly. A robust algorithm is PAM (*partitioning around medoids*) implemented in the `pam()` function from **cluster**. Can you work out how to obtain cluster IDs and examine the clustering quality?

Q: The **cluster** package also offers algorithms `agnes()` and `diana()` for hierarchical clustering. Can you make them work with the procedure above? (Hint: you will need to call the function `as.hclust()` at some point.)

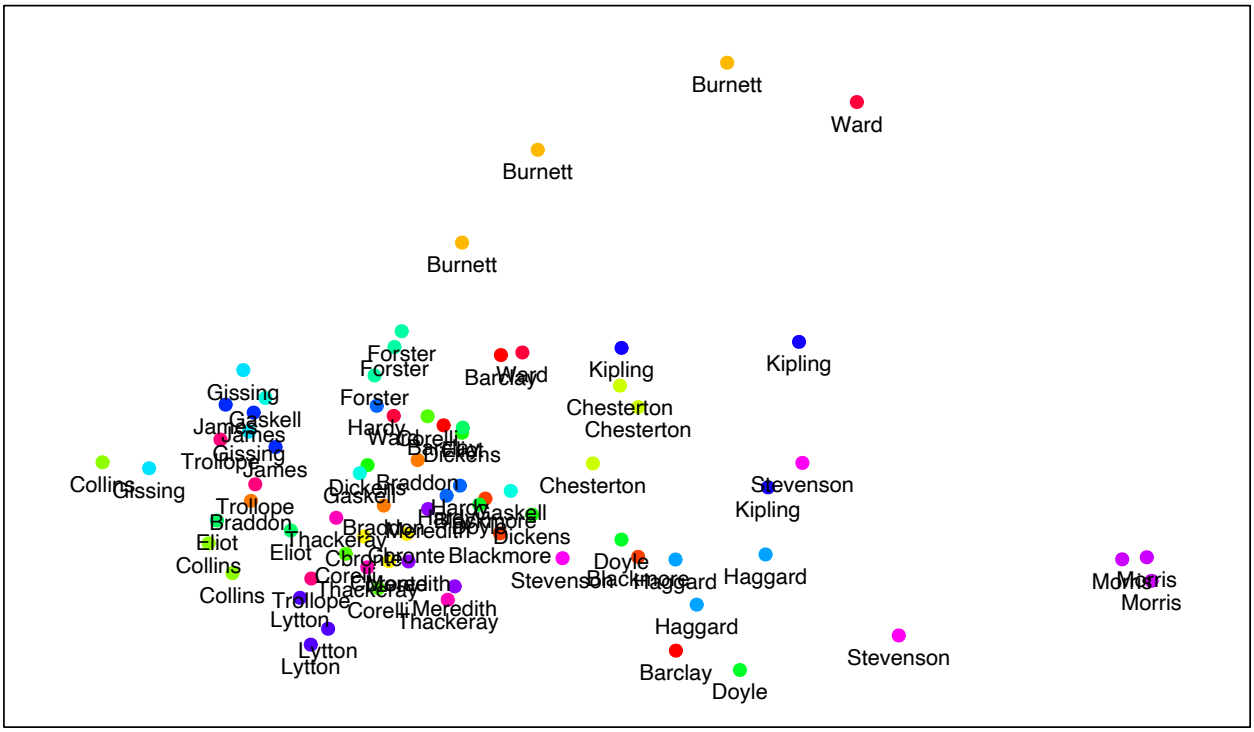
3.2 Topological maps

Multidimensional scaling (MDS) visualizes high-dimensional data in the form of a topological map, which attempts to display data points near each other that are close in the original space. Since we will want to re-do this plot with different mapping algorithms (and perhaps other parameter settings), it's time to define an ad-hoc function – note that most parameters are hard-coded to the data we're currently interested in, which makes the function much easier to write and use.

```
delta.map <- function (xy, main="") {
  xr <- expand.range(xy[, 1], by=.05) # add 5% margin for labels
  yr <- expand.range(xy[, 2], by=.05)
  col.vec <- rainbow(25)[MetaA$author]
  par(mar=c(1,1,2,1), xaxt="n", yaxt="n")
  plot(xy, pch=20, cex=1.5, col=col.vec, main=main,
       xlab="", ylab="", xlim=xr, ylim=yr)
  text(coord, labels=MetaA$author, pos=1, cex=.8)
}
```

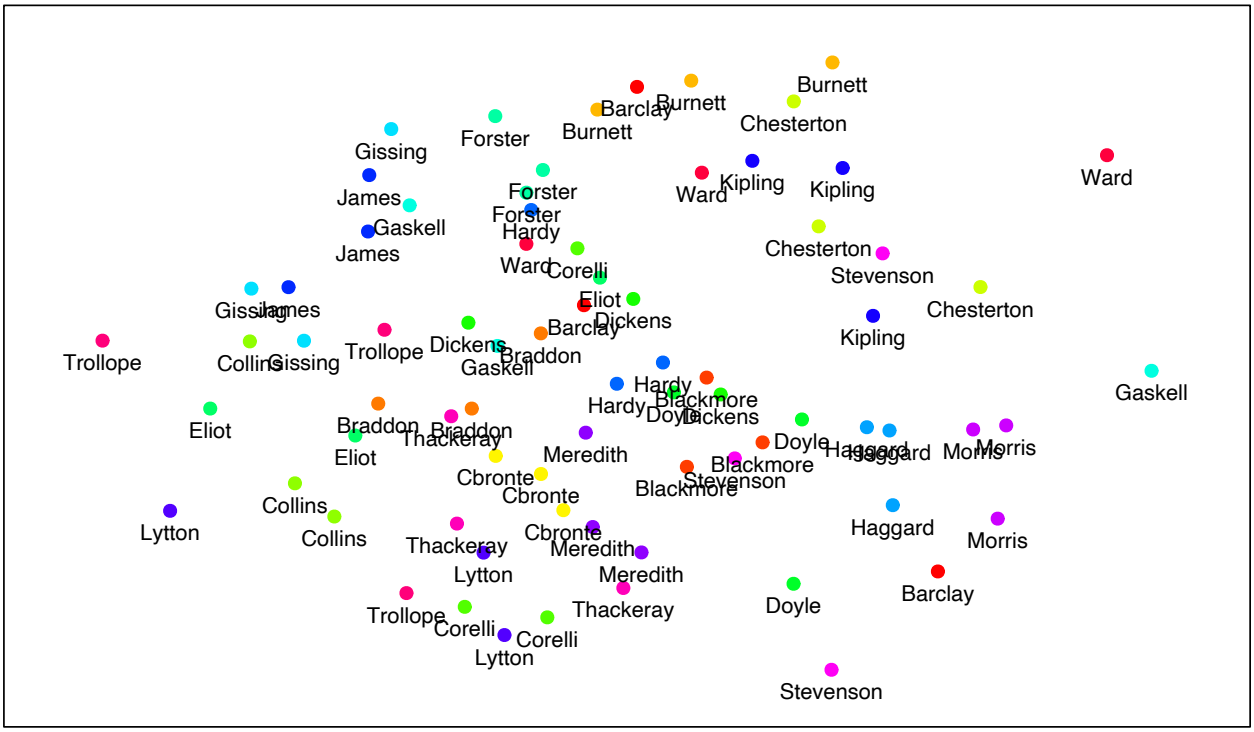
The classical MDS algorithm uses a linear mapping that cannot represent data sets with complex high-dimensional structure.

```
coord <- cmdscale(DMA)
delta.map(coord, main="Classical MDS")
```



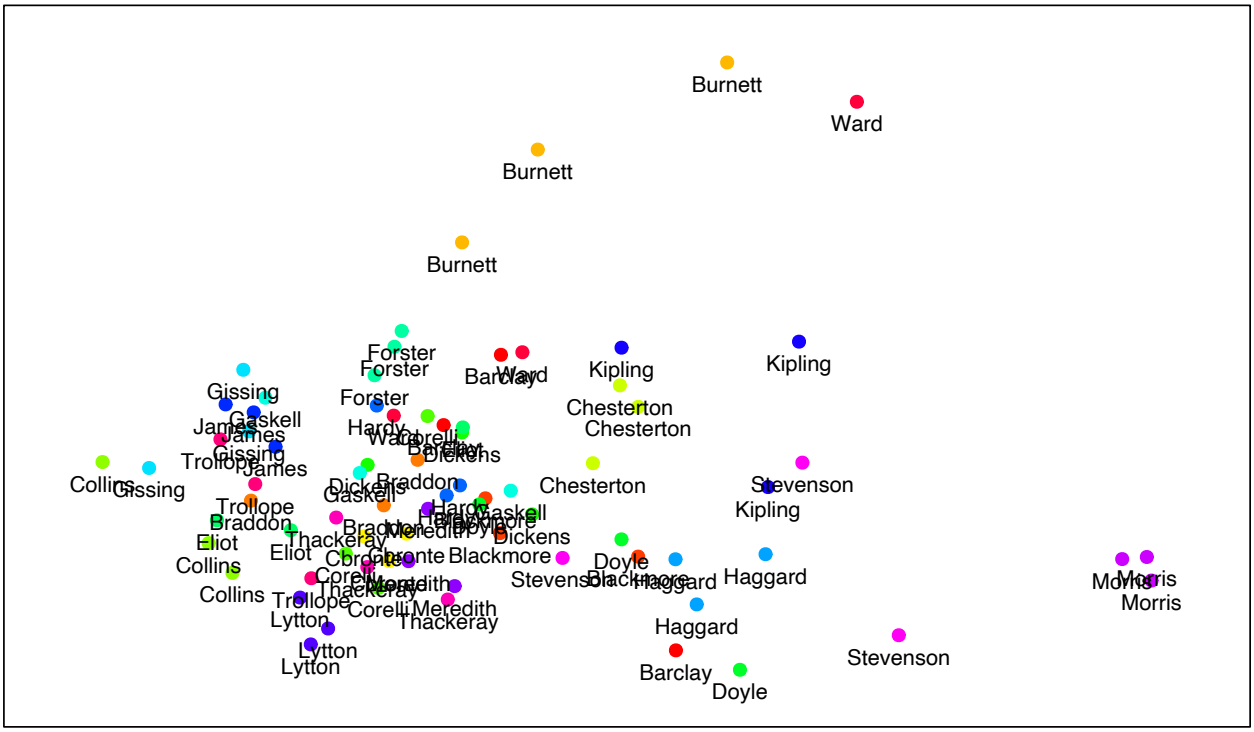
The **MASS** package offers two versions of non-linear MDS. The Sammon algorithm attempts to preserve distances and thus tends to spread out points evenly if all pairwise distances are fairly large. It fails to show interesting topological structure in such cases.

```
coord <- sammon(DMA, trace=FALSE)$points
delta.map(coord, main="Non-linear MDS (sammon)")
```



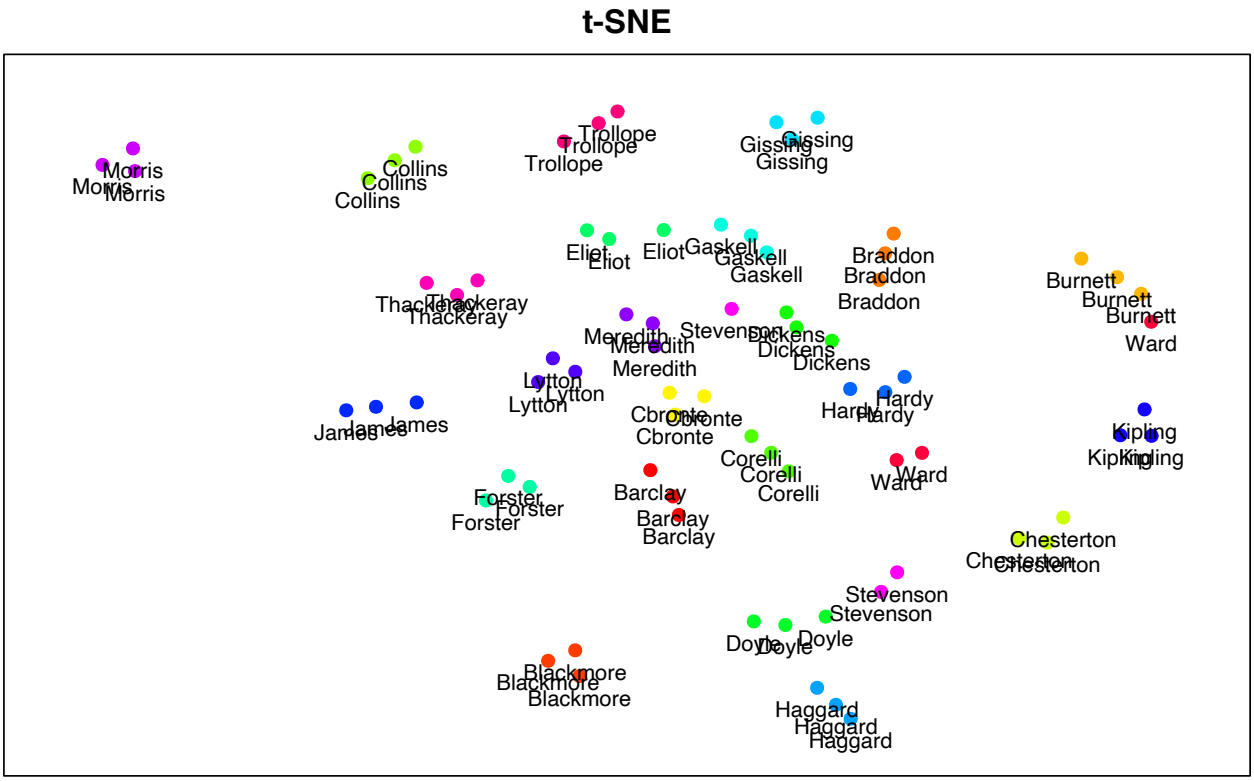
The other algorithm (attributed to Kruskal) allows for more structure in the map (by penalizing larger distances less severely). However, it can be fairly instable and sometimes fails to improve over classical MDS (used as its initial configuration).

```
coord <- isoMDS(DMA, trace=FALSE)$points
delta.map(coord, main="Non-linear MDS (isoMDS)")
```



A more sophisticated approach attempts to produce a topological mapping that preserves neighbourhood structure but allows for a substantial distortion of the larger geometry. The state-of-the-art algorithm t-SNE (*t-distributed stochastic neighbour embedding*) is currently very popular. We use an implementation in the R package **Rtsne**. For smaller data sets, the `perplexity` parameter needs to be reduced from its default value of 30. One big disadvantage is that the stochastic algorithm can produce entirely different visualizations depending on the random seed (try setting it to `1`).

```
set.seed(1984)
coord <- Rtsne(DMA, perplexity=10)$Y
delta.map(coord, main="t-SNE")
```



3.3 Exercise

> Explore other distance metrics, clustering methods and visualization parameters. Researchers have found that Delta performs especially well with angular distance (also known as *cosine similarity*). You can computer angular distance with the `dist.matrix()` function from **wordspace** (it is the default setting if no `method` argument is specified), but remember to set `as.dist=TRUE` if you want to use the resulting distance matrix with `hclust()`.

As a starting point, take a look at the parameter settings explored by Evert *et al.* (2017). They found that clustering quality often depends on the number n_w of most frequent words included in the vector representation. Can you work out how to adjust n_w?

- Evert, Stefan; Proisl, Thomas; Jannidis, Fotis; Reger, Isabella; Pielström, Steffen; Schöch, Christof; Vitt, Thorsten (2017). Understanding and explaining Delta measures for authorship attribution. *Digital Scholarship in the Humanities*, **22**(suppl_2), ii4–ii16. https://doi.org/10.1093/llc/fqx023

4 Multidimensional analysis

We will work with the same subset of 923 BNC texts from four text types as in the overview presentation (variable names use the suffix `P` for *presentation*).

```
idx <- with(MetaB, mode == "written"
            & author_sex %in% c("male", "female")
            & sampling_type %in% qw("beginning middle end")
            & derived_type != "unpublished"
            & n_words >= 20000 & n_words <= 50000)
ZLP <- ZLB[idx, ]
MetaP <- droplevels(MetaB[idx, ])
nrow(MetaP)
```

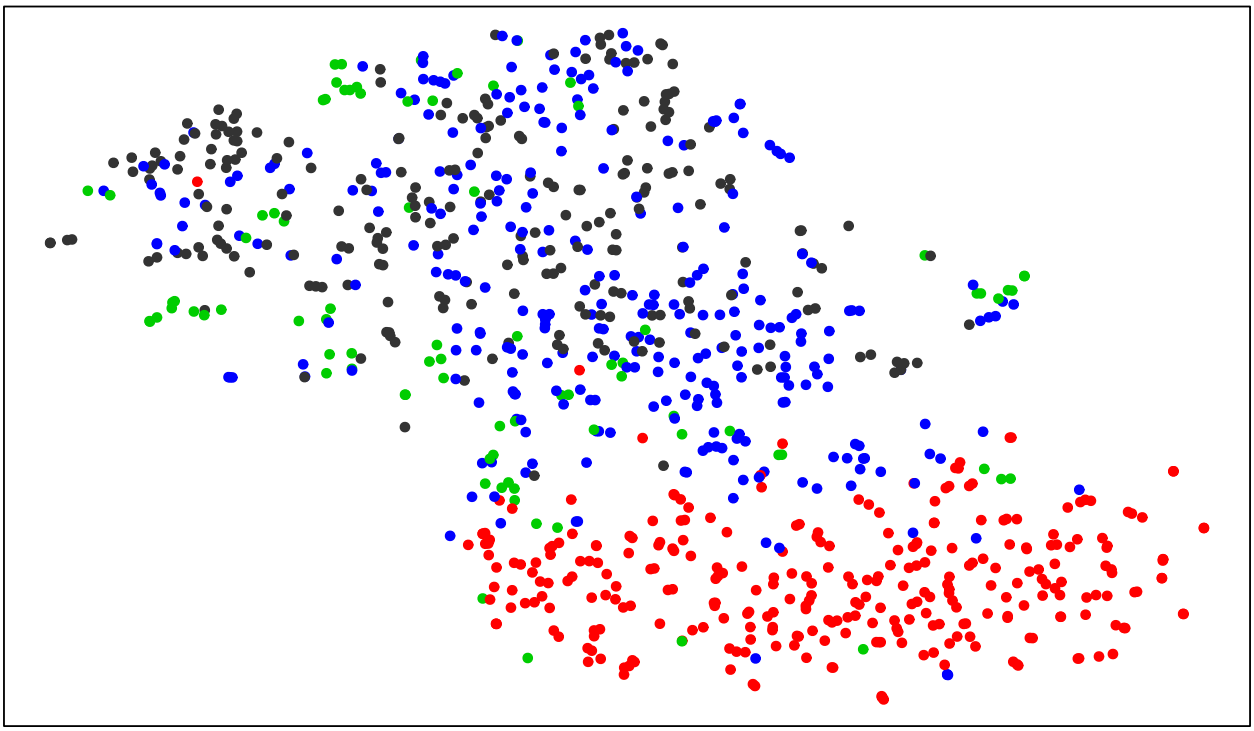
```
## [1] 923
```

```
table(MetaP$derived_type)
```

```
##
##       academic        fiction misc_published          prose
##            235            299             86            303
```

We start with a visualisation of the topological structure of the data set using t-SNE. The support function `gma.pairs()` offers an easy way to highlight text types in the scatterplot.

```
par(mar=c(1,1,1,1), xaxt="n", yaxt="n")
set.seed(1984)
coord <- Rtsne(dist(ZLP), perplexity=15)$Y
gma.pairs(coord, Meta=MetaP, col=derived_type, pch.vals=20, compact=TRUE)
```

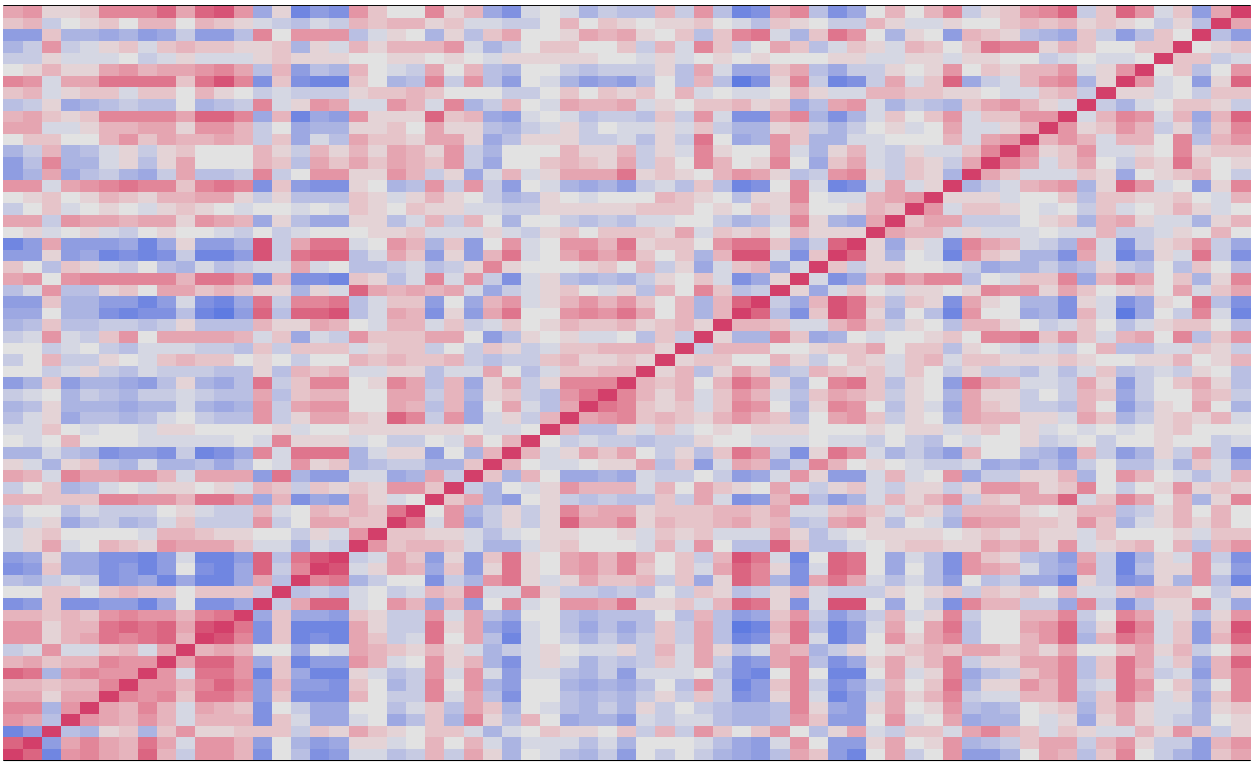


One group of texts (mainly *fiction*) seems clearly separated from the rest, but beyond this there is little cluster structure in the data. Thus, only a small part of the linguistic variation captured by Biber's features is explained by text type, and there appear to be multiple overlapping dimensions of variation, motivating a correlation approach to multivariate analysis.

4.1 Factor analysis

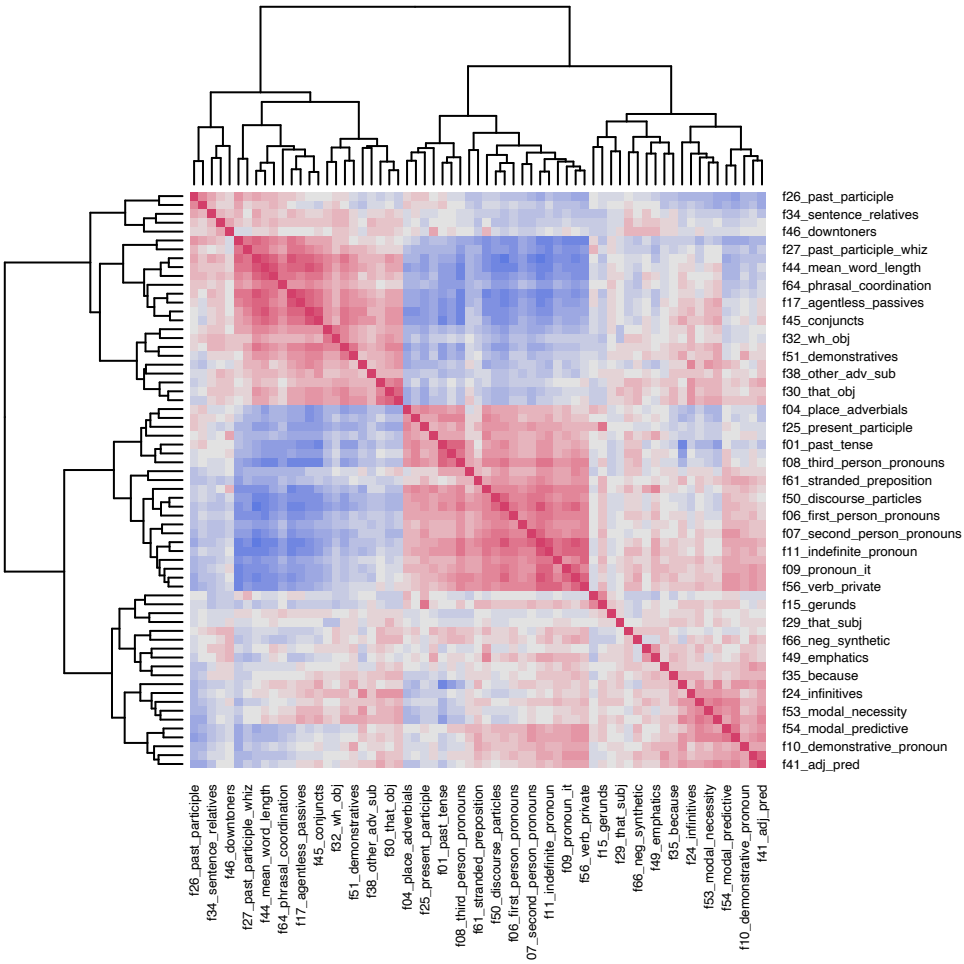
Doubg Biber has embraced **factor analysis** as a multivariate technique to analyse correlation patterns and determine the **latent dimensions** of variation. Unlike the geometric configurations that GMA focuses on, its starting point is the correlation matrix of all measured variables.

```
CMP <- cor(ZLP)
par(mar=c(1,1,1,1), xaxt="n", yaxt="n")
image(CMP, zlim=c(-1, 1), col=hcl.colors(21, "Blue-Red2"))
```



This heatmap visualisation already shows a rich structure, with many blocks of variables that have similar correlation patterns (and are also adjacent in Biber's original list). We can bring out this structure more clearly by reordering the rows and columns of the heatmap using the `heatmap()` function.

```
# par(mar=c(1,1,1,1), xaxt="n", yaxt="n")
heatmap(CMP, symm=TRUE, zlim=c(-1, 1), col=hcl.colors(21, "Blue-Red2"),
        cexRow=.5, cexCol=.5, margins=c(6, 6))
```



Factor analysis will attempt to "explain" the very conspicuous block structure of this matrix in terms of latent variables. The assumption is that each text has certain coordinates in the different latent dimensions depending on its linguistic properties, which in turn affect groups of correlated surface features with different (positive or negative) weights.

R has a built-in function `factanal()` for factor analysis. A crucial "magic" parameter, the number of latent variables, has to be specified by the user (we follow Biber in choosing 5 factors). In addition, a heuristic rotation is applied in the latent space in order to "make the factors more interpretable". R only offers the most common rotations VariMax (default) and ProMax.

```
FAP <- factanal(ZLP, 5, rotation="varimax", scores="regression")
```

The loadings of a factor show how it affects each surface feature. Again following Biber, only loadings with an absolute value above 0.35 are printed.

```
print(FAP$loadings, cutoff=.35)
```

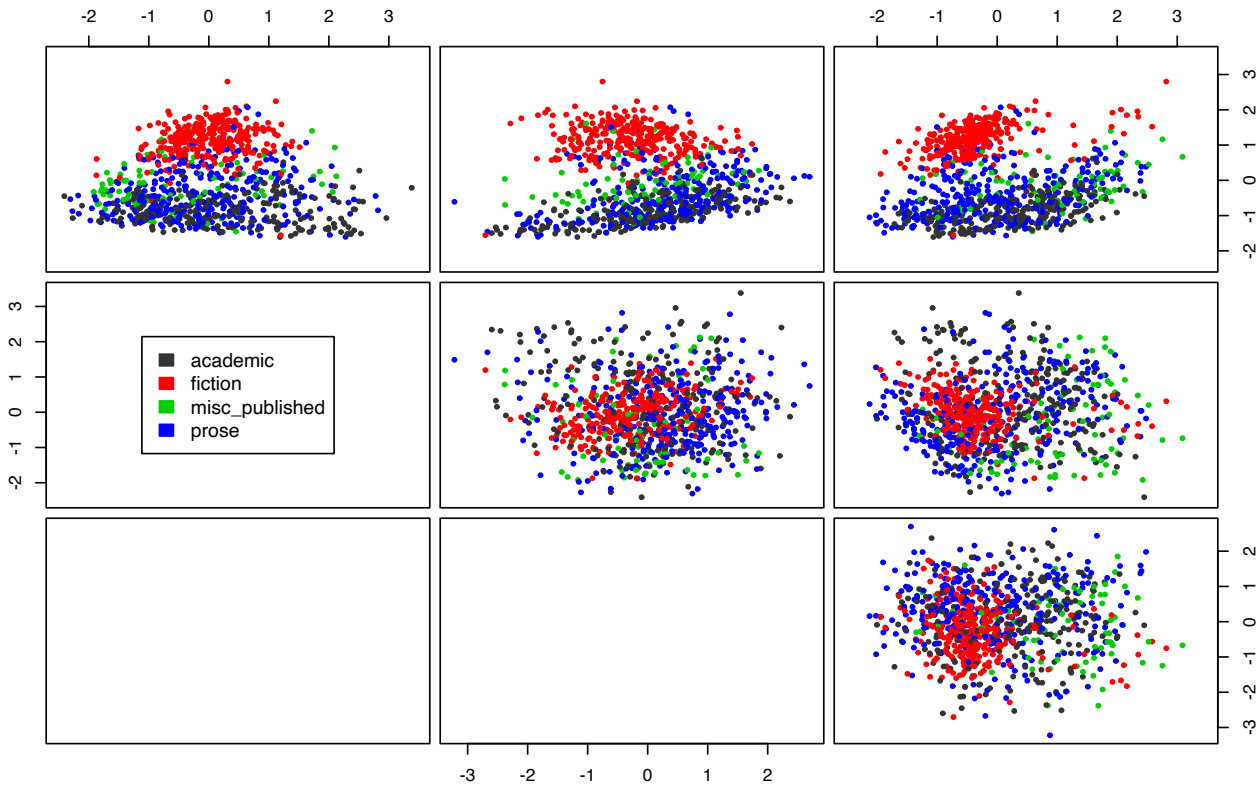
```
##
## Loadings:
##                          Factor1 Factor2 Factor3 Factor4 Factor5
## f01_past_tense            0.562                   -0.727
## f02_perfect_aspect        0.517                   -0.581
## f03_present_tense                  0.462            0.819
## f04_place_adverbials      0.525  -0.378                   -0.404
## f05_time_adverbials       0.519
```

```
## f06_first_person_pronouns    0.745
## f07_second_person_pronouns   0.746
## f08_third_person_pronouns    0.755                -0.462
## f09_pronoun_it               0.708  0.356
## f10_demonstrative_pronoun           0.433
## f11_indefinite_pronoun       0.839
## f12_proverb_do               0.876
## f13_wh_question              0.615
## f14_nominalization          -0.849                        0.419
## f15_gerunds
## f16_other_nouns             -0.716 -0.449
## f17_agentless_passives      -0.849
## f18_by_passives             -0.879
## f19_be_main_verb                    0.563
## f20_existential_there
## f21_that_verb_comp          -0.443  0.567
## f22_that_adj_comp                   0.443  0.449
## f23_wh_clause                0.645
## f24_infinitives                     0.586
## f25_present_participle       0.587
## f26_past_participle                -0.626
## f27_past_participle_whiz    -0.765
## f28_present_participle_whiz                -0.383
## f29_that_subj
## f30_that_obj                -0.455  0.423  0.429
## f31_wh_subj                 -0.576
## f32_wh_obj                  -0.426         0.423
## f33_pied_piping             -0.619         0.363
## f34_sentence_relatives                     0.384
## f35_because
## f36_though                                 0.450
## f37_if                              0.744
## f38_other_adv_sub
## f39_prepositions            -0.911
## f40_adj_attr                -0.731                0.360
## f41_adj_pred                        0.706
## f42_adverbs                  0.814
## f43_type_token                     -0.543
## f44_mean_word_length        -0.890
## f45_conjuncts               -0.759
## f46_downtoners                             0.581
## f47_hedges                   0.532
## f48_amplifiers                             0.483
## f49_emphatics                0.358         0.464
## f50_discourse_particles      0.785
## f51_demonstratives          -0.448
## f52_modal_possibility               0.711         0.378
## f53_modal_necessity                 0.694
## f54_modal_predictive                0.638
## f55_verb_public              0.370                -0.365
## f56_verb_private             0.675  0.495
## f57_verb_suasive            -0.510  0.471
## f58_verb_seem                              0.528
## f59_contractions             0.865
```

```
## f61_stranded_preposition     0.495
## f62_split_infinitve
## f63_split_auxiliary                  0.611
## f64_phrasal_coordination    -0.580                   0.357
## f66_neg_synthetic                                   -0.368
## f67_neg_analytic             0.749   0.451
##
##                Factor1 Factor2 Factor3 Factor4 Factor5
## SS loadings     18.776   8.100   3.743   3.683   1.654
## Proportion Var   0.289   0.125   0.058   0.057   0.025
## Cumulative Var   0.289   0.413   0.471   0.528   0.553
```

The latent coordinates (or *scores*) of the individual texts can be inferred by regression (as we did in the `factanal()` call above). We can then visualise configurations in different dimensions with a scatterplot matrix.

```
CoordP <- FAP$scores
gma.pairs(CoordP, dim=1:4, Meta=MetaP, col=derived_type, pch.vals=20, cex=.8, iso=TRUE, compact=TRUE)
```



The first two factors show a picture that is remarkably similar to the t-SNE visualisation we began with. If your RGL installation works, you can also view the first three factors in a 3D plot to obtain further insights into the configuration of data points in the factor space.

```
gma.3d(CoordP, Meta=MetaP, col=derived_type, size=.04)
view3d(theta=0, phi=0, zoom=.5)
```

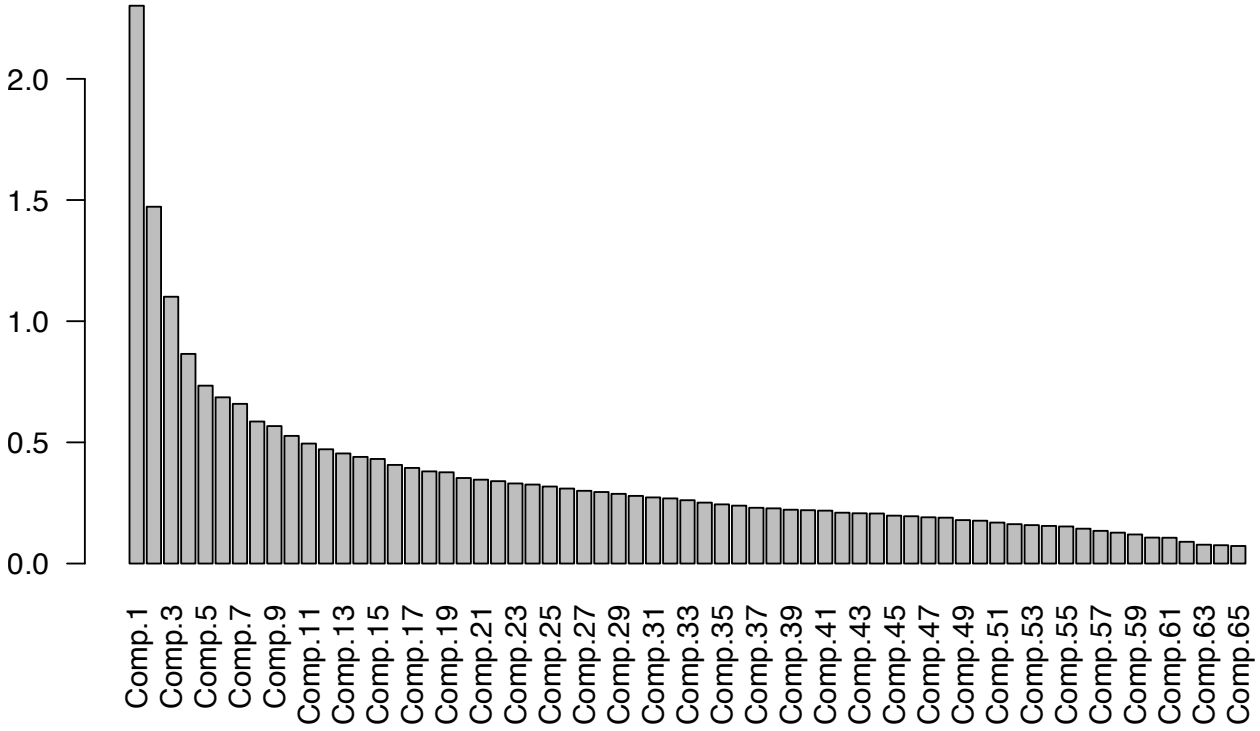
> **Q:** Re-run the factor analysis with different parameters (number of factors and rotation) and compare the results. How much does the "magic" affect your interpretation?

4.2 Geometric analysis

A geometric approach - as advocated by our method of **geometric multivariate analysis** - starts from the distances between data points rather than the correlation matrix. Its latent dimensions are simply an orthogonal (i.e. geometry-preserving) projection into a low-dimensional subspace. This subspace is usually determined to capture as much of the distance information as possible. For (squared) Euclidean distances, this is the same as maximising the residual variance of all feature variables after projection.

This algorithm is known as **principal component analysis** (PCA). It has no "magic" parameters: the number of latent dimensions does not have to be determined a priori, and the "rotation" is uniquely determined by the data set. R has two built-in implementations of PCA: `princomp()` can be used in a very similar way to `factanal()`.

```
PCAP <- princomp(ZLP, scores=TRUE)
barplot(PCAP$sdev, las=2)
```



PCA returns 65 latent dimensions, order by the amount of variance they capture. For comparison with the factor analysis, we focus on the first 5 PCA dimensions.

```
lds <- PCAP$loadings[, 1:5]
class(lds) <- "loadings" # need some trickery to select dimensions
print(lds, cutoff=.15) # PCA loadings are on a different scale
```

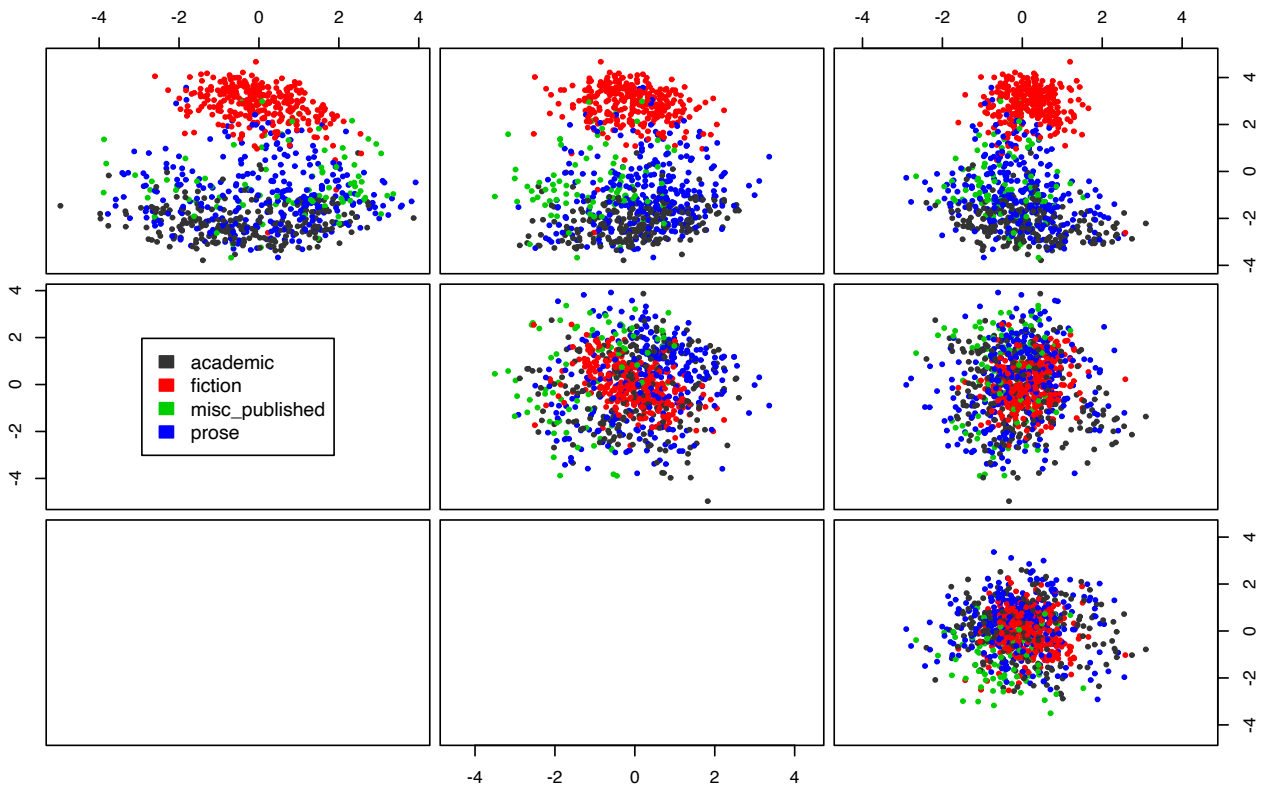
```
##
## Loadings:
##                           Comp.1 Comp.2 Comp.3 Comp.4 Comp.5
## f01_past_tense             0.236  0.197  0.275  0.174
## f02_perfect_aspect         0.179         0.283
## f03_present_tense                -0.213 -0.229 -0.192 -0.156
## f04_place_adverbials       0.164  0.167                0.177
## f05_time_adverbials
## f06_first_person_pronouns  0.159
## f07_second_person_pronouns
```

```
## f08_third_person_pronouns    0.256         0.157
## f09_pronoun_it
## f10_demonstrative_pronoun
## f11_indefinite_pronoun       0.169
## f12_proverb_do
## f13_wh_question
## f14_nominalization          -0.245
## f15_gerunds                              -0.270         0.481
## f16_other_nouns                   0.154
## f17_agentless_passives      -0.224
## f18_by_passives             -0.217
## f19_be_main_verb                  -0.197
## f20_existential_there
## f21_that_verb_comp                -0.192  0.250
## f22_that_adj_comp                         0.208         0.213
## f23_wh_clause
## f24_infinitives                   -0.221         0.164  0.275
## f25_present_participle       0.158        -0.173         0.309
## f26_past_participle                0.206
## f27_past_participle_whiz    -0.152                 0.165
## f28_present_participle_whiz               -0.217         0.312
## f29_that_subj                                    -0.229  0.270
## f30_that_obj                              0.241         0.167
## f31_wh_subj                 -0.163         0.152
## f32_wh_obj                                0.236
## f33_pied_piping             -0.175         0.162
## f34_sentence_relatives                    0.188
## f35_because
## f36_though                                0.190 -0.191
## f37_if                            -0.237
## f38_other_adv_sub                                        0.171
## f39_prepositions            -0.218
## f40_adj_attr                -0.203               -0.223
## f41_adj_pred                      -0.259
## f42_adverbs                                      -0.252
## f43_type_token
## f44_mean_word_length        -0.219
## f45_conjuncts               -0.248
## f46_downtoners                            0.157 -0.270
## f47_hedges
## f48_amplifiers                                   -0.289
## f49_emphatics                                    -0.267
## f50_discourse_particles
## f51_demonstratives
## f52_modal_possibility             -0.304
## f53_modal_necessity               -0.205         0.165
## f54_modal_predictive              -0.186         0.165
## f55_verb_public                           0.173  0.153
## f56_verb_private                  -0.193
## f57_verb_suasive                                 0.274
## f58_verb_seem                             0.224 -0.209
## f59_contractions
## f61_stranded_preposition
## f62_split_infinitve
```

```
## f63_split_auxiliary                -0.218
## f64_phrasal_coordination
## f66_neg_synthetic                          0.210
## f67_neg_analytic                   -0.161
##
##                 Comp.1 Comp.2 Comp.3 Comp.4 Comp.5
## SS loadings      1.000  1.000  1.000  1.000  1.000
## Proportion Var   0.015  0.015  0.015  0.015  0.015
## Cumulative Var   0.015  0.031  0.046  0.062  0.077
```

Similarly, we need to select the first dimensions of the scores for visualisation, but this can easily be done by the support functions.

```
gma.pairs(PCAP$scores, dim=1:4, Meta=MetaP, col=derived_type, pch.vals=20, cex=.8, iso=TRUE, compact=TRU
```

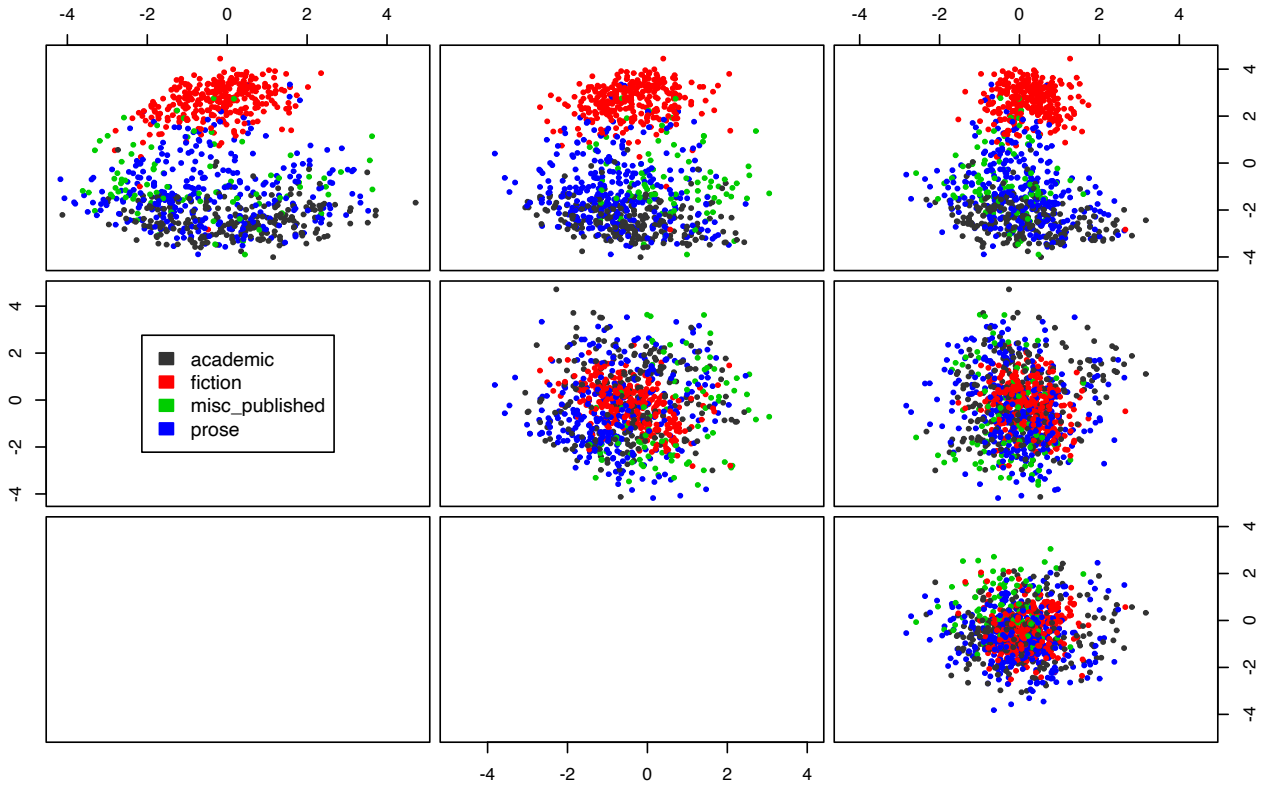


PCA does not seem much different from factor analysis according to this visualisation. If you feel that the PCA scatterplots look more spherical, you aren't wrong at all. In a 3D view, differences between the configuration of data points in the latent dimensions can be recognised more easily.

```
gma.3d(PCAP$scores, Meta=MetaP, col=derived_type, size=.06)
view3d(theta=0, phi=0, zoom=.5)
```

PCA is the central unsupervised technique of GMA, where it is combined with minimally supervised interventions. GMA is supported by a set of user-friendly functions and the class of `GMA` objects. The scatterplot above thus takes just two lines of very readable code:

```
GMAP <- GMA(ZLP)
gma.pairs(GMAP$projection("complement", dim=1:4),
          Meta=MetaP, col=derived_type, pch.vals=20, cex=.8, iso=TRUE, compact=TRUE)
```



Note that the first axis is flipped here: the orientation of latent dimensions is arbitrary both in PCA and in factor analysis. The GMA methodology is explored in much more detail in the notebook **07_GMA** and applied to a study of "shining through" effects in translation.

4.3 Exercise

Carry out a multidimensional analysis (a) for the complete BNC data set and (b) for the Brown Family corpora. Note that some visualisation algorithms (notably t-SNE) can be very expensive for large data sets, so you might want to skip this part.