SIGIL Unit 07: Geometric Multivariate Analysis

Stephanie Evert

5 March 2023

Contents

1 Data set & preprocessing

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.

```r
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:

```r
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.

```r
library(rgl)
```

1.1 The CroCo data set: replicating Evert & Neumann (2017)

As usual, we assign the data matrix and metdata to shorter names.

```r
MC <- CroCo_Matrix
MetaC <- CroCo_Meta
```

Get an overview of the available metadata:

```r
head(MetaC)
```

```
##                      id register language status tokens chunks sentences
## 416   gtrans-share-012    share       DE  trans   2529    516        95
## 425  gtrans-speech-008   speech       DE  trans   2760    593       101
## 130   etrans-essay-021    essay       EN  trans   2136    689        95
## 375 gtrans-fiction-002  fiction       DE  trans   3536   1120        89
## 289       go-share-005    share       DE   orig   4157   1078       216
## 234       go-essay-007    essay       DE   orig   2244    720       141
```

There are 8 different registers, but Evert & Neumann excluded *instruction*, *tourism* and *fiction*.

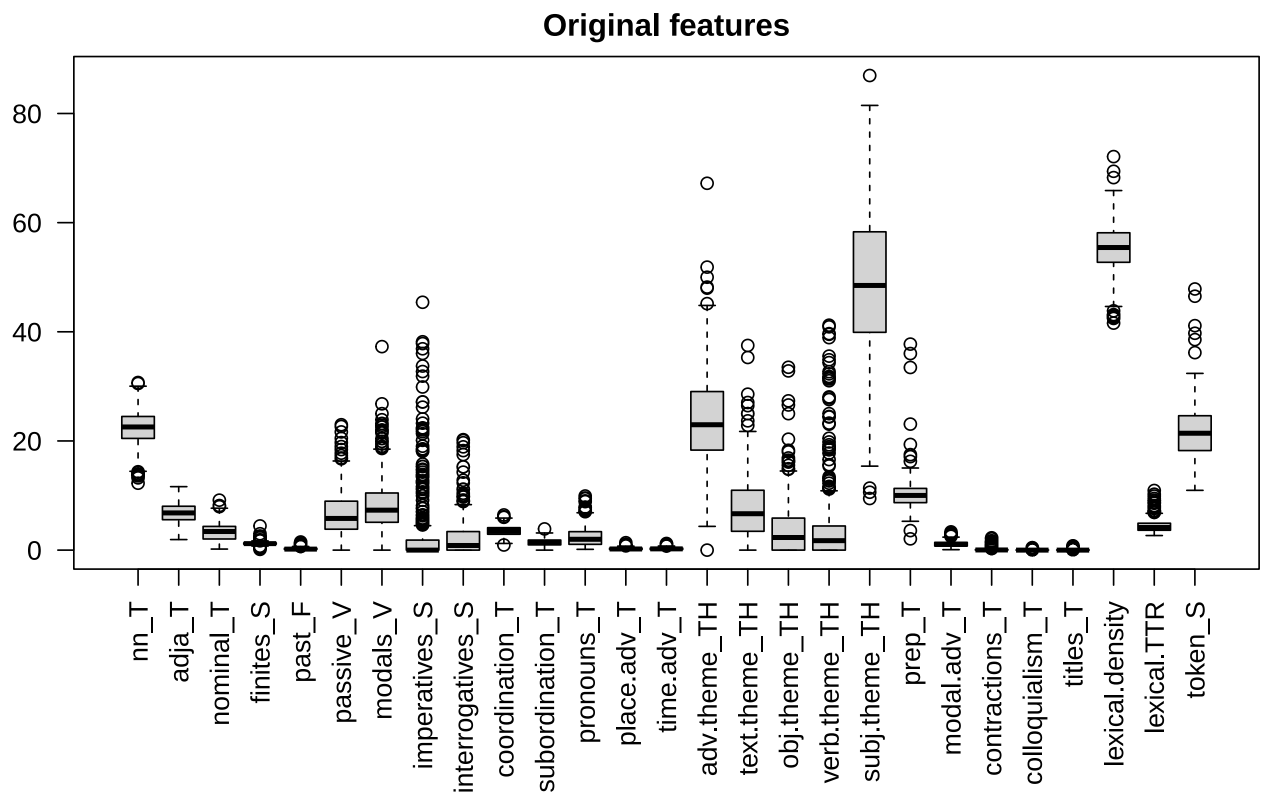
```r
table(MetaC$register)
```

```
##
##       essay     fiction instruction      popsci       share      speech     tourism
##         102          40          48          42          48          64          66
##         web
##          42
```

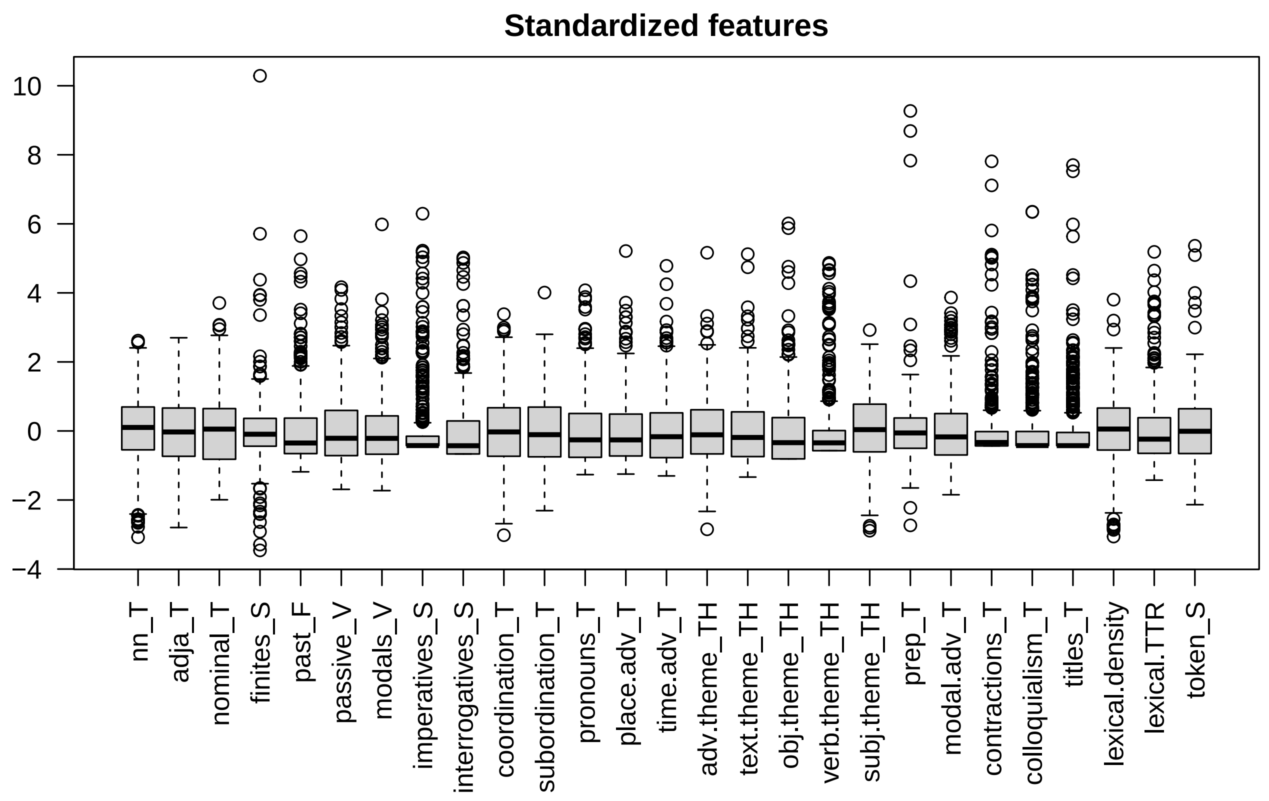
1.2 Feature scaling

Even though the grammatical features of the CroCo data set do not follow a Zipfian distribution like word frequencies, standardization is essential because different features cover entirely different frequency ranges.

```r
par(mar=c(7, 2, 2, 1)) # make room for labels
boxplot(MC, las=2, main="Original features")
```

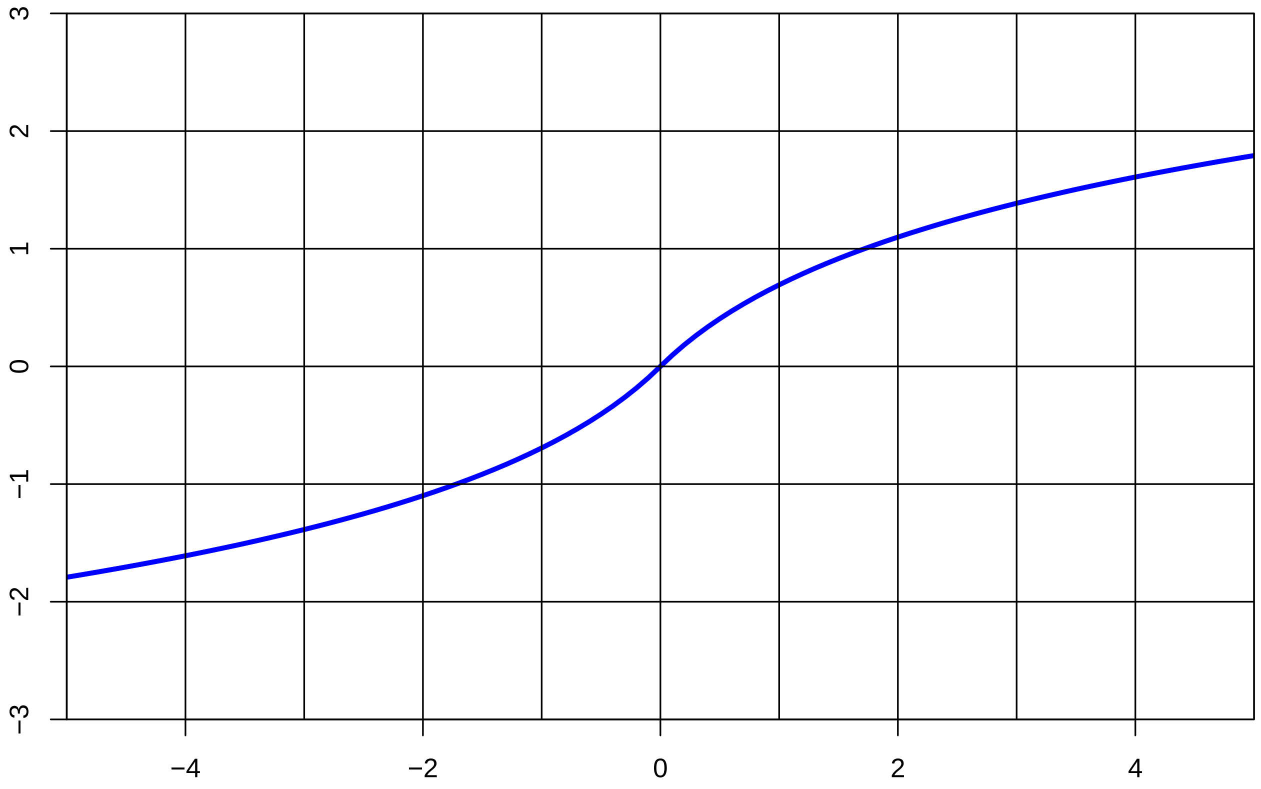


```
ZC <- scale(MC)
par(mar=c(7, 2, 2, 1))
boxplot(ZC, las=2, main="Standardized features")
```

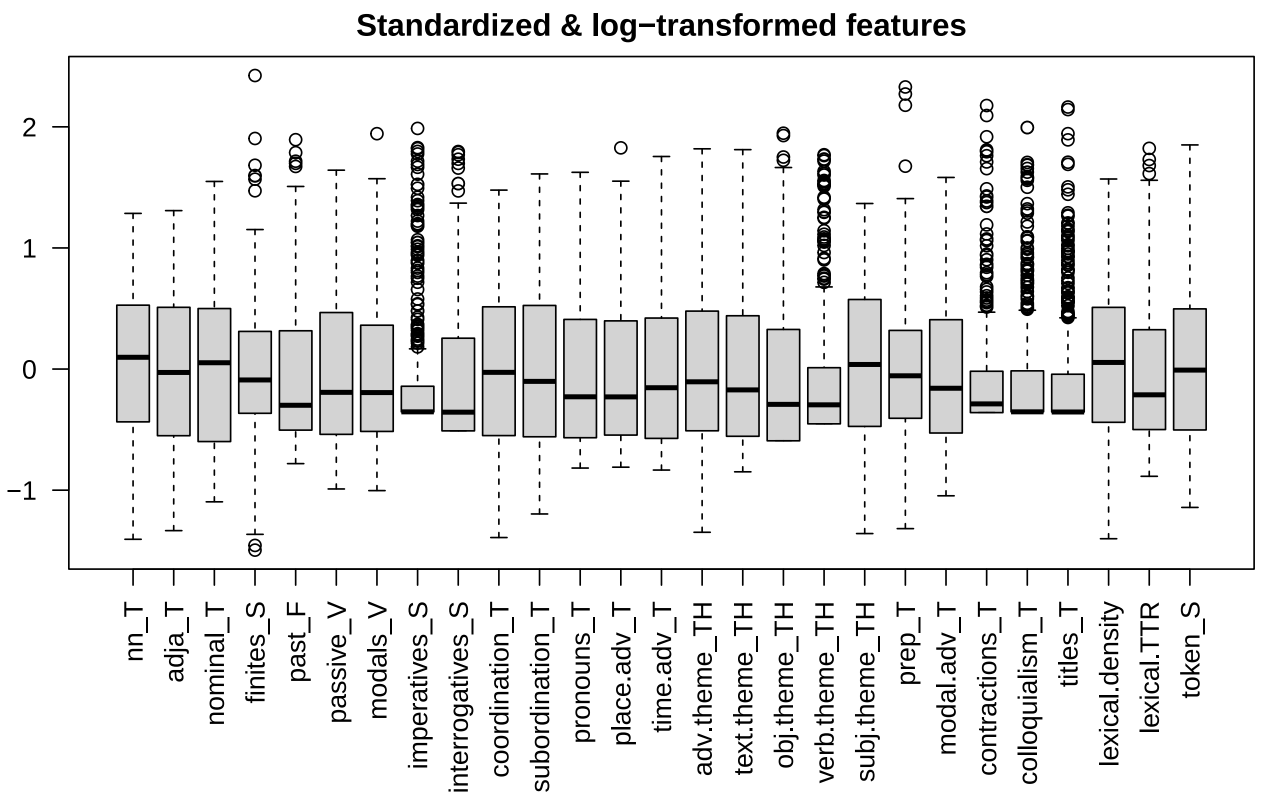


We might also apply a logarithmic transformation to further reduce the effect of outlier values on the geometric structure. Let us first see how the `signed.log()` functions affects the data values.

```
par(mar=c(2, 2, 1, 1), xaxs="i", yaxs="i")
curve(signed.log, lwd=3, col="blue", xlim=c(-5, 5), ylim=c(-3, 3), xlab="", ylab="")
abline(v=-5:5, h=-3:3)
```



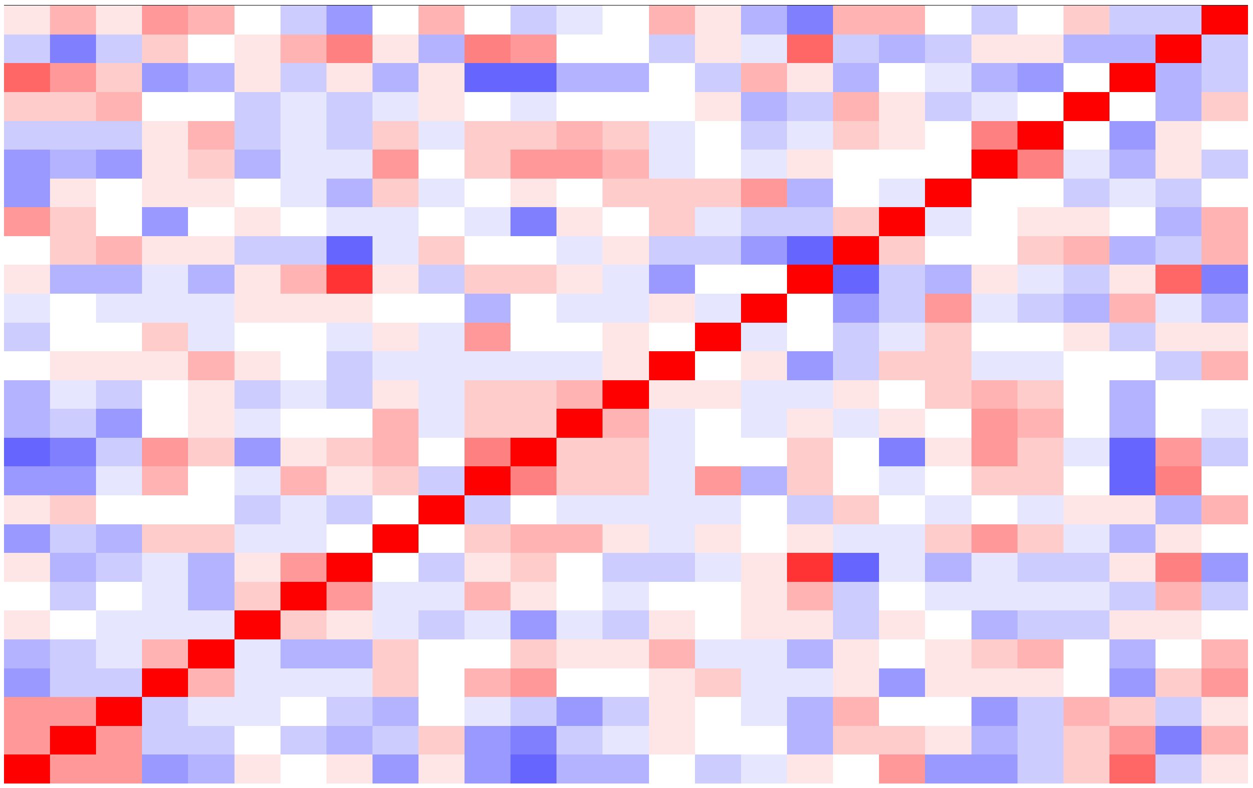
```
ZLC <- signed.log(ZC)
par(mar=c(7, 2, 2, 1))
boxplot(ZLC, las=2, main="Standardized & log-transformed features")
```



1.3 Checking correlations

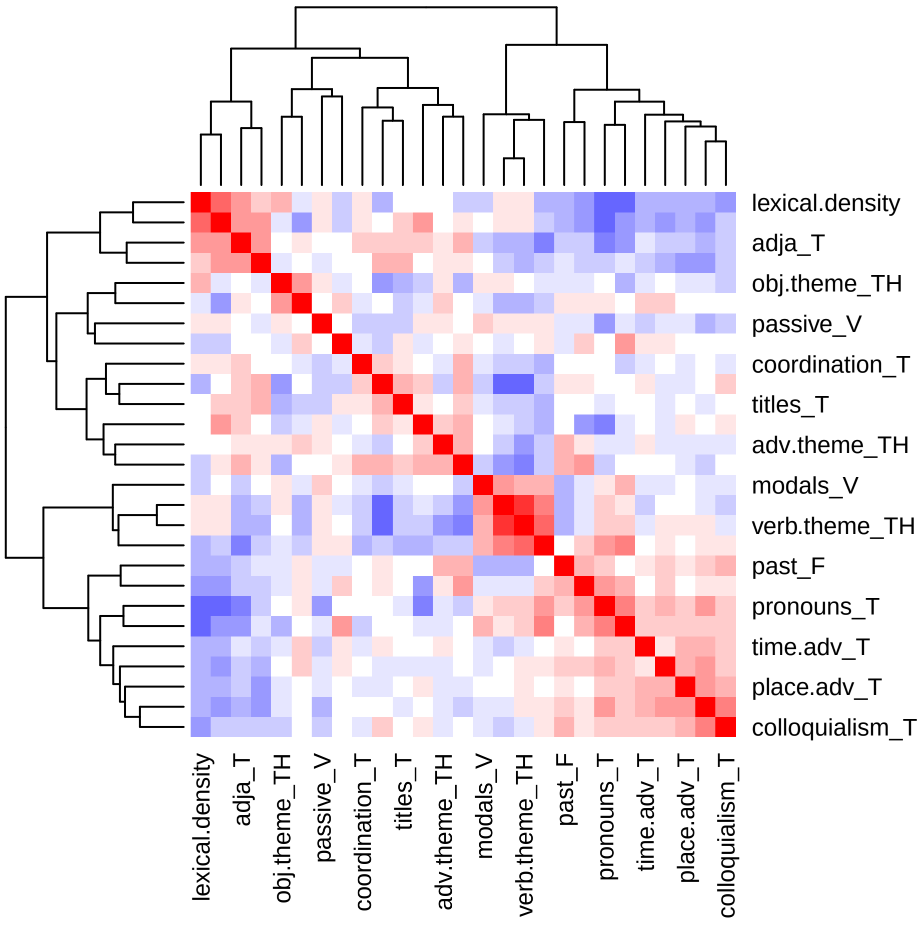
Another preliminary step is to check for collinearities and other overly strong correlations between the features. If there are such strong correlations or conspicuous large blocks in the plot, some features may need to be excluded or a different unit of measurement may be required. For interpretability, we need to specify the range $[-1, 1]$ of possible correlation scores and choose a suitable colour scale.

```
cor.colours <- c(
  hsv(h=2/3, v=1, s=(10:1)/10), # blue = negative correlation
  rgb(1,1,1),                   # white = no correlation
  hsv(h=0, v=1, s=(1:10/10))    # red = positive correlation
)
par(mar=c(0, 0, 0, 0), xaxt="n", yaxt="n")
image(cor(ZLC), zlim=c(-1, 1), col=cor.colours)
```



A "heatmap" plot automatically groups strongly correlated variables using a clustering algorithm, so the visualization is much easier to interpret.

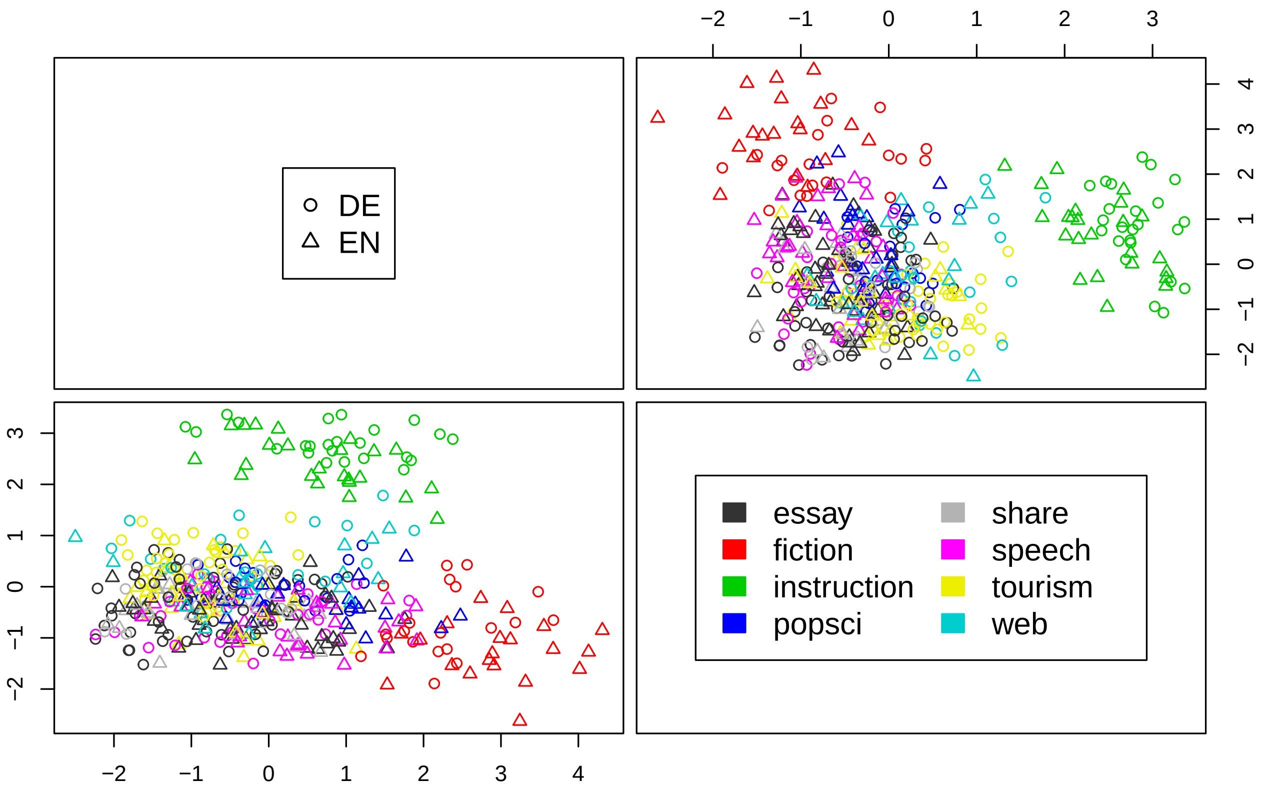
```r
heatmap(cor(ZLC), symm=TRUE, zlim=c(-1,1), col=cor.colours, margins=c(7,7))
```



1.4 High-dimensional visualisation

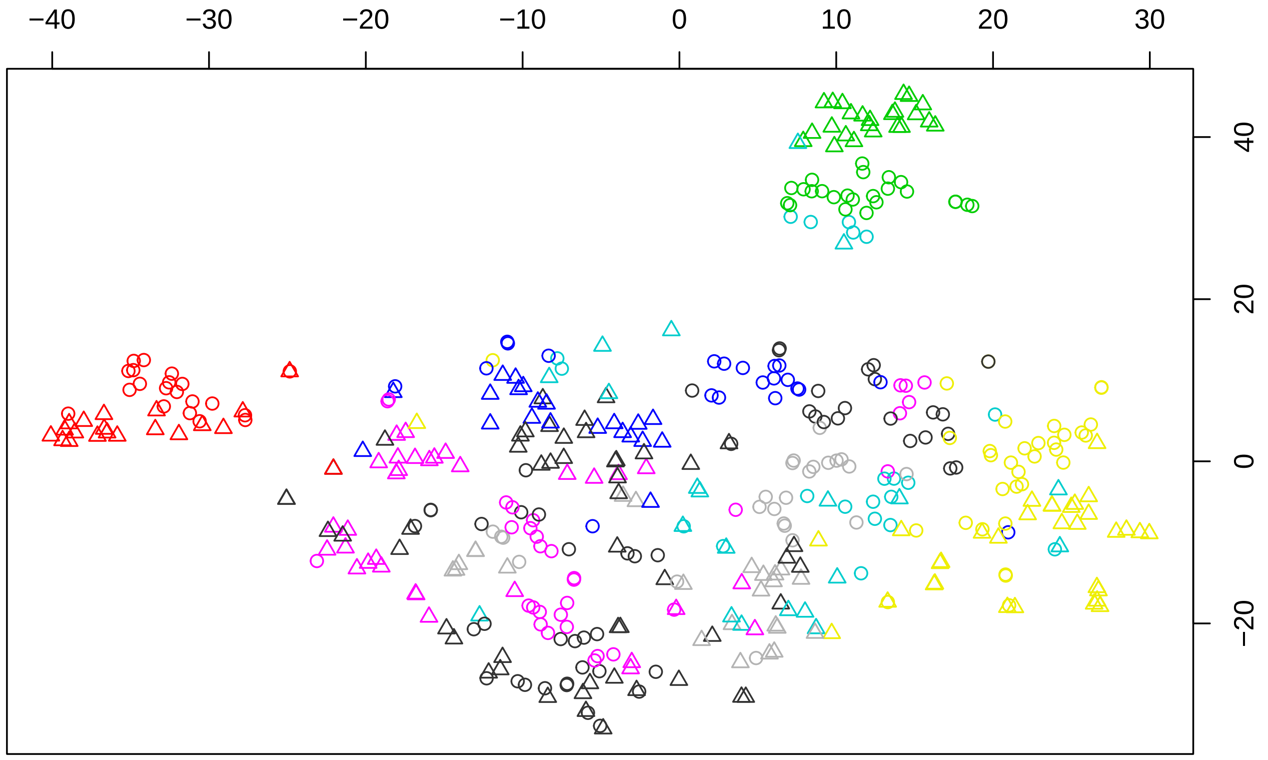
Multidimensional scaling or another visualisation technique can give us a first indication of the geometric structure of the CroCo data set. We use the `gma.pairs()` function (introduced further below for scatterplot matrices of latent dimensions), which allows us to conveniently highlight different metadata categories.

```
coord <- cmdscale(dist(ZLC)) # isoMDS is hardly different
gma.pairs(coord, Meta=MetaC, col=register, pch=language)
```



More sophisticated algorithms such as t-SNE bring out the topological structure more clearly, revealing *fiction*, *instruction manuals* and *tourism* as outlier registers (and thus explaining why they were excluded by Evert & Neumann).

```
set.seed(1984)
coord <- Rtsne(dist(ZLC), perplexity=10)$Y
gma.pairs(coord, Meta=MetaC, col=register, pch=language, compact=TRUE)
```



Q: We create a reduced feature matrix and metadata table below without these three registers, but will carry out the experiments below with the full data set. Try working with the reduced data set for yourself, starting with a t-SNE visualisation and highlighting translations vs. originals. Can you replicate the results from the overview talk and the paper with the reduced data set?

```
idx <- MetaC$register %in% qw("fiction instruction tourism") # texts to exclude
MCr <- MC[!idx, ]
ZCr <- scale(MCr) # re-standardise w/o outlier values from removed registers
ZLCr <- signed.log(ZCr)
MetaCr <- droplevels(MetaC[!idx, ]) # droplevels removes empty categories from factor variables
```

2 Unsupervised GMA

2.1 Projection with PCA

We understand orthogonal projection as a lower-dimensional **perspective** on the geometric configuration of data points in high-dimensional space. This only makes sense in combination with the Euclidean metric, because the orthogonal projection decomposes squared Euclidean distance. A group of support functions (such as `mvar.space()` below) helps you to work with projections without having to carry out all the low-level matrix operations.

The method of **principal component analysis** (PCA) finds a perspective that preserves as much of the distance information as possible. It is not necessary to choose the number of dimensions in advance: PCA returns a sequence of orthogonal basis vectors that represent increasingly larger optimal subspaces.

Instead of using the standard R function `prcomp()` directly, we work with **GMA** objects, a class designed for supporting GMA studies. The object is initialised with the feature matrix to be analysed.

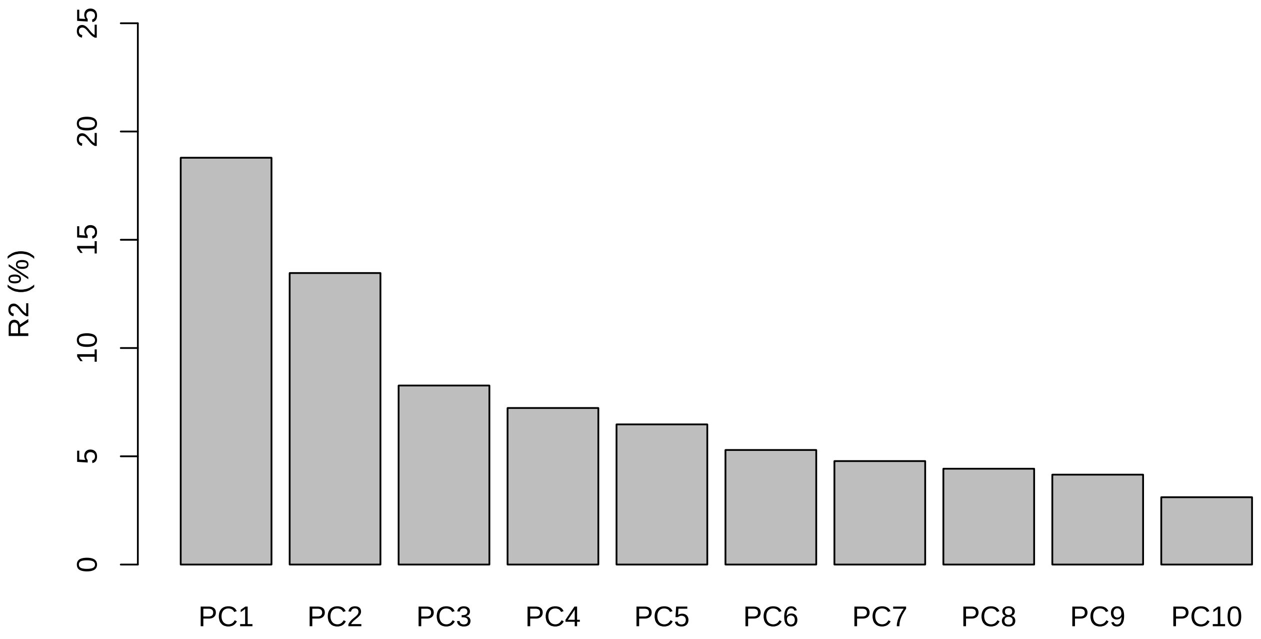
```
PCA <- GMA(ZLC)
PCA
```

```
## GMA object representing projection of 452 x 27 data matrix into 0-dimensional subspace
```

GMA objects decompose the feature space into a low-dimensional **target space**, consisting of dimensions selected in a weakly supervised manner or by visual inspection, and its **orthogonal complement**. In this first unsupervised analyses, the target space is empty and the complement encompasses the entire feature space.

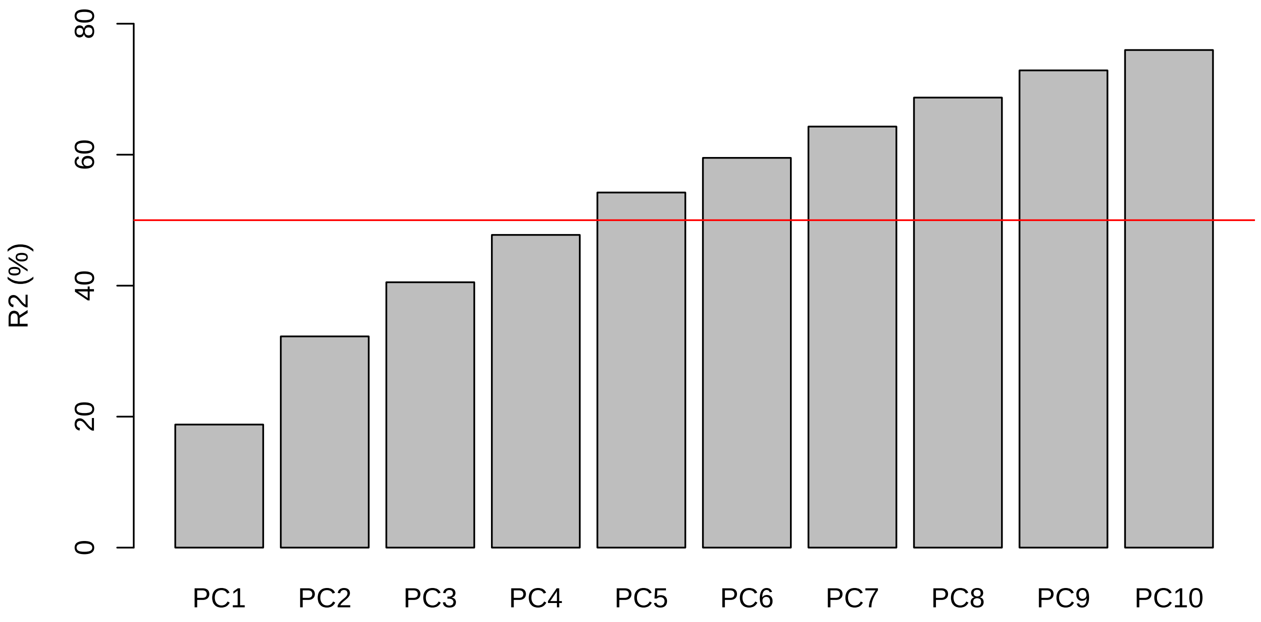
PCA is automatically applied to the complement space, so the first dimensions of the complement are our latent PCA dimensions. R^2 values (expressed here as percentages) show what proportion of the distance information is captured by each PCA dimension. Let us look at the first 10 PCA dimensions:

```r
r2 <- PCA$R2(dim=1:10)
barplot(r2, ylim=c(0, 25), ylab="R2 (%)")
```



If we project into the first three dimensions, the total variance captured by the projection is the sum of the first three R^2 values. The barplot below shows that the first 5 dimensions already contain more than half of the (squared Euclidean) distance information.

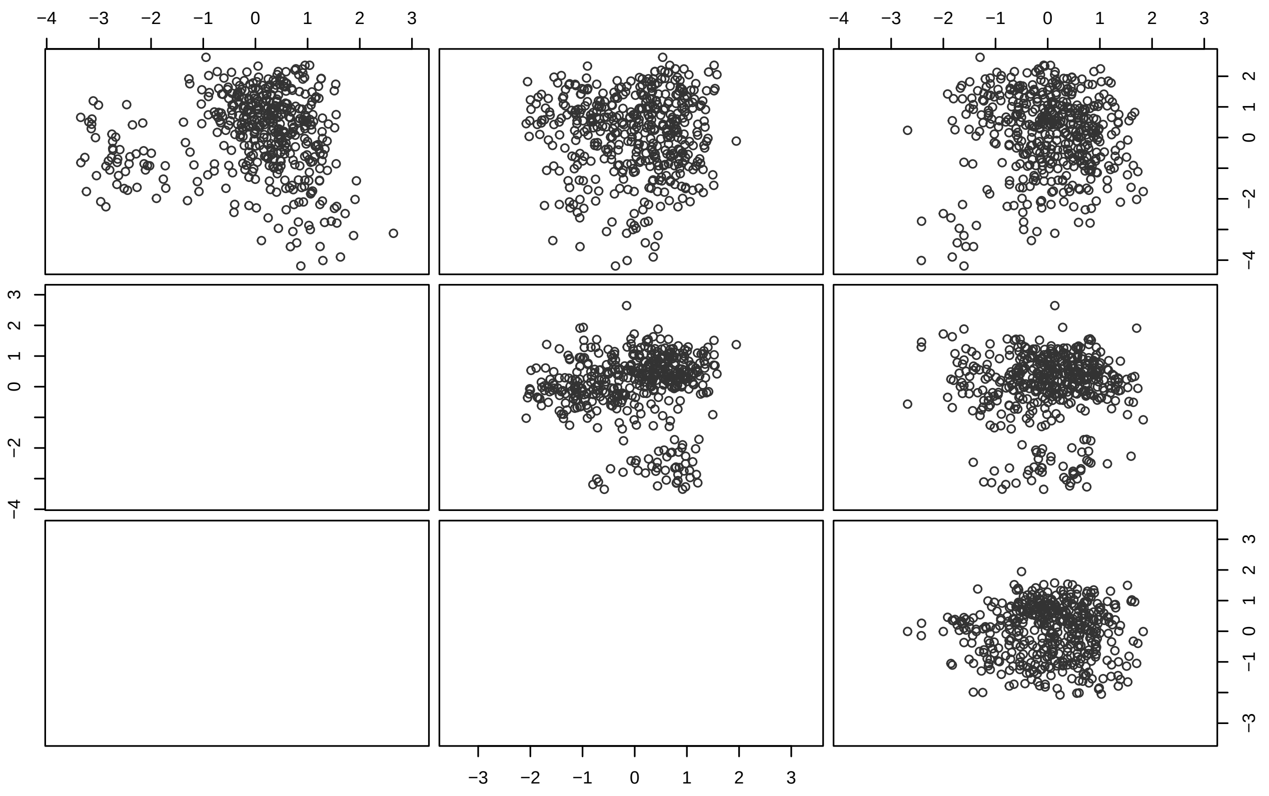
```r
barplot(cumsum(r2), ylim=c(0, 80), ylab="R2 (%)")
abline(h=50, col="red")
```



2.2 Scatterplot visualisation

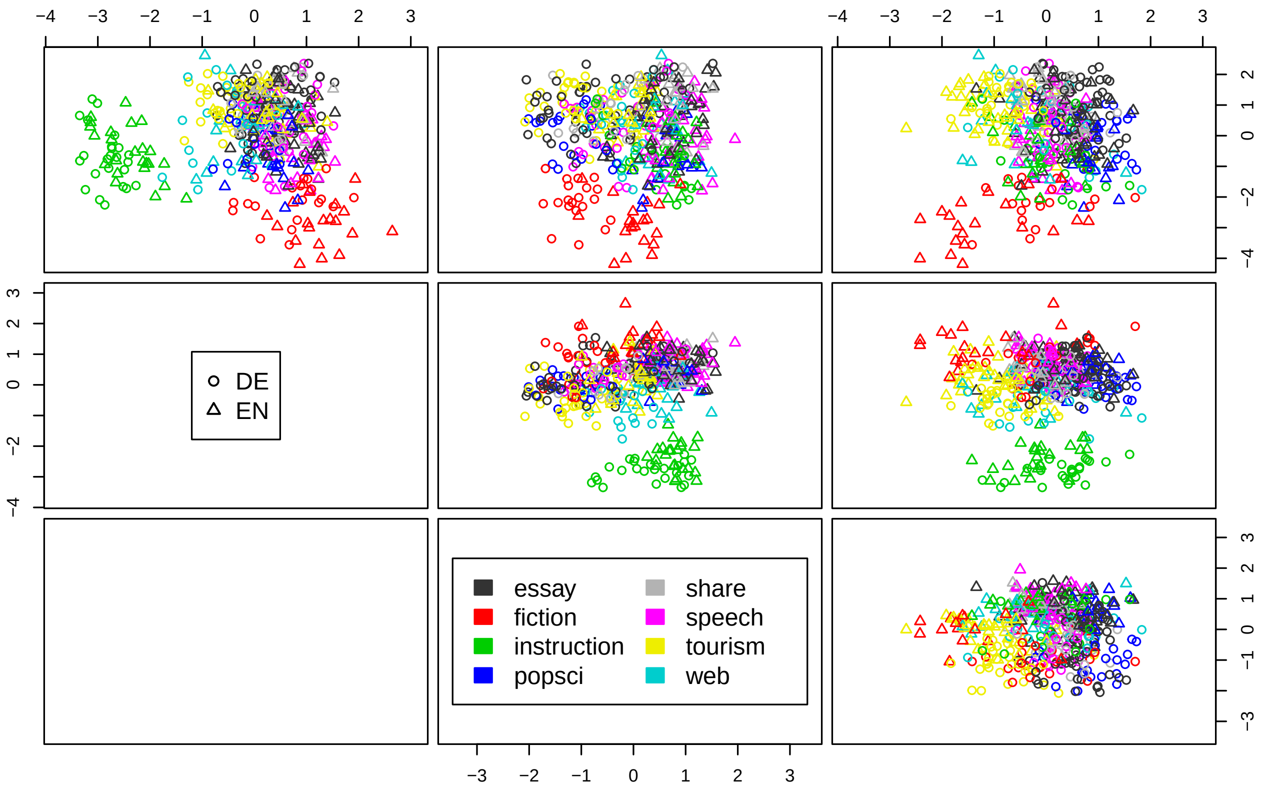
Let us look at the projection into the first four PCA dimensions. We need to specify `space="complement"` to obtain dimensions of the complement space rather than the (here zero-dimensional) target space, or `space="both"` to append them to the target space dimensions. A scatterplot matrix for these dimensions can be drawn with `gma.pairs()`, which should always be used with the options shown below.

```r
Proj4 <- PCA$projection("complement", dim=1:4)
gma.pairs(Proj4, compact=TRUE, iso=TRUE)
```



Geometric perspectives on large data sets are often difficult to interpret without reference to metadata categories such as genre, language and translation status. The scatterplot function accepts a metadata table (which must correspond to the data points in the feature mtarix, of course) so the information shown by plot symbol and colour can be selected by specifying the corresponding column names of the table.

```
gma.pairs(Proj4, Meta=MetaC, compact=TRUE, iso=TRUE,
          col=register, pch=language)
```



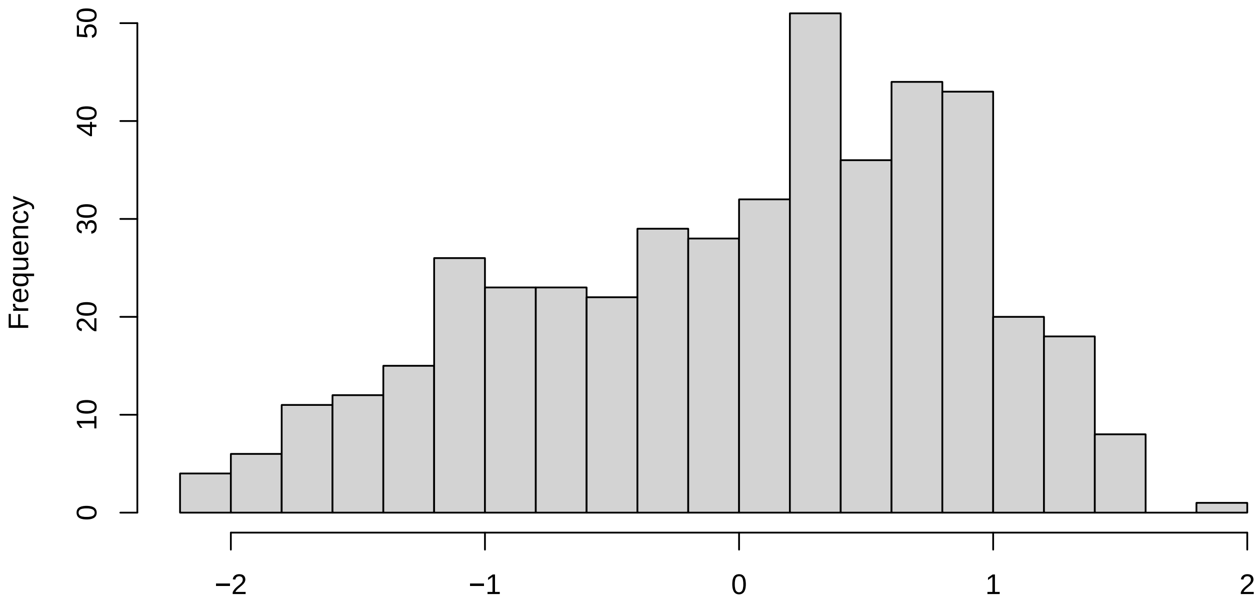
If you have a working installation of the **rgl** package, you can also view an interactive 3D visualization. The `mvar.3d` displays metadata information in the same way as `mvar.pairs`, so it is easy to switch between the scatterplot matrix and the 3D view. By default the first three dimensions are shown and arranged so that the front, left and top view correspond to the top left panels of the scatterplot matrix.

```
gma.3d(Proj4, Meta=MetaC, iso=TRUE, legend=TRUE,
       col=register, pch=language, size=.1)
view3d(theta=0, phi=0, zoom=.7) # reset to front view
```

2.3 Exploring the PCA dimensions

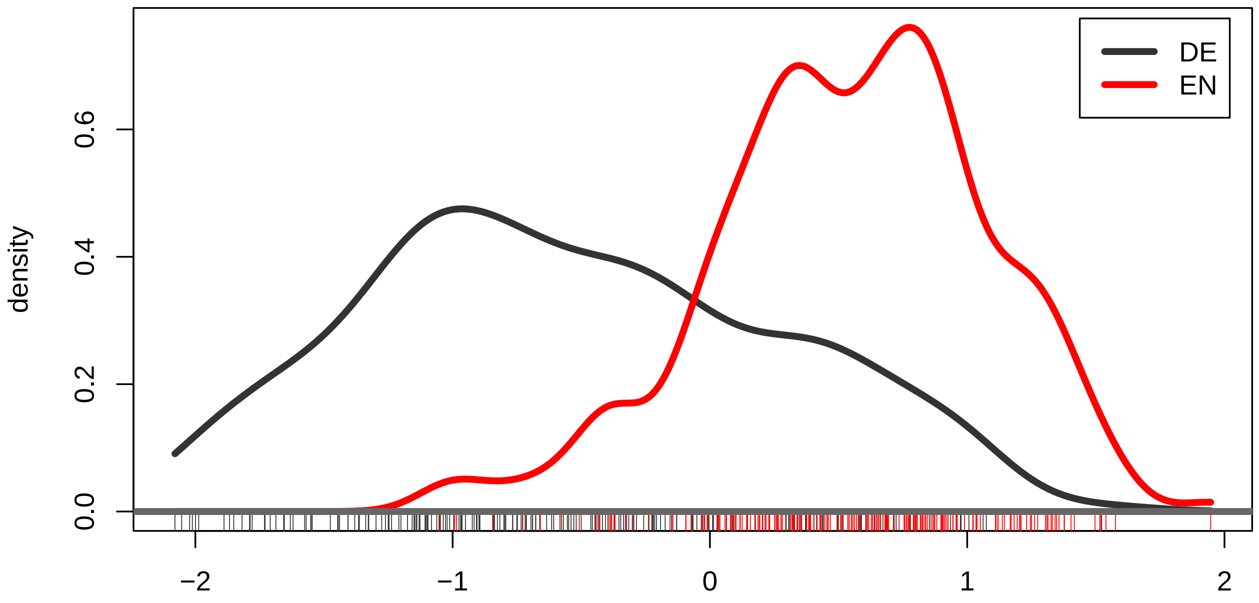
The visualization suggests that PCA dimension 3 separates between English and German texts. In order to take a closer look at the distribution of texts along this axis, we can examine the coordinates along this axis.

```
hist(Proj4[, 3], breaks=20, xlab="", main="")
```



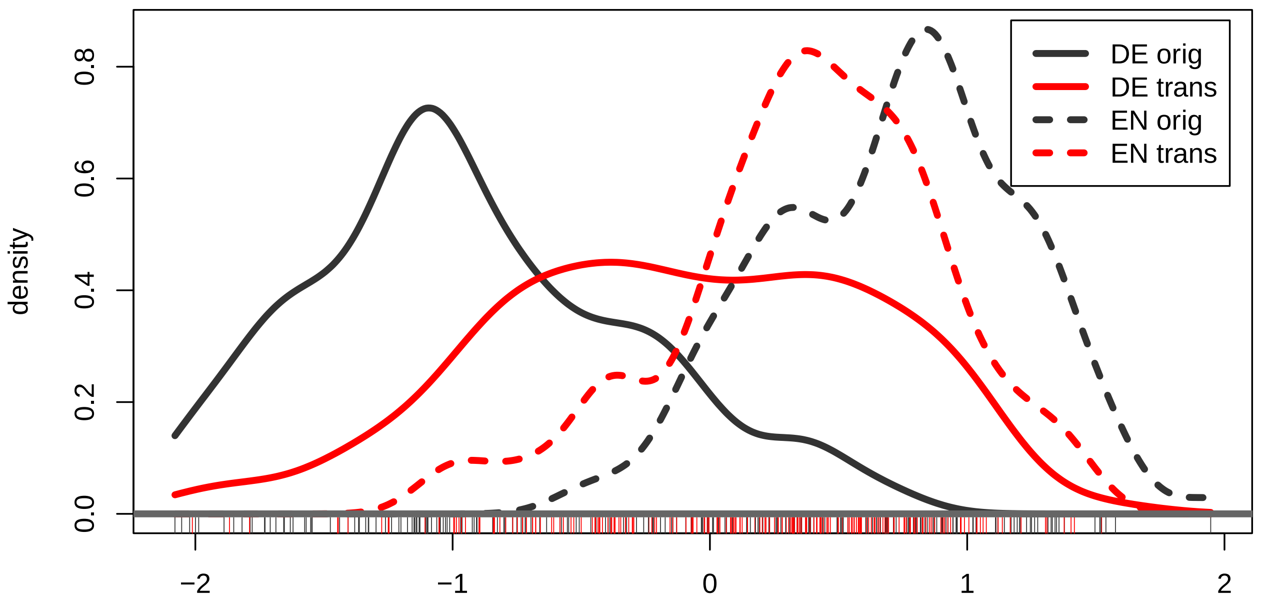
Such a visualization is of little use without separating the German and English texts. The utility function `discriminant.plot()` displays separate distributions for the specified categories and allows us to select an arbitrary axis in the space. Here, we can use precomputed coordinates on the third principal component and omit the axis vector.

```
discriminant.plot(Proj4[, 3], NULL, MetaC$language, rug=TRUE, xlab="")
```



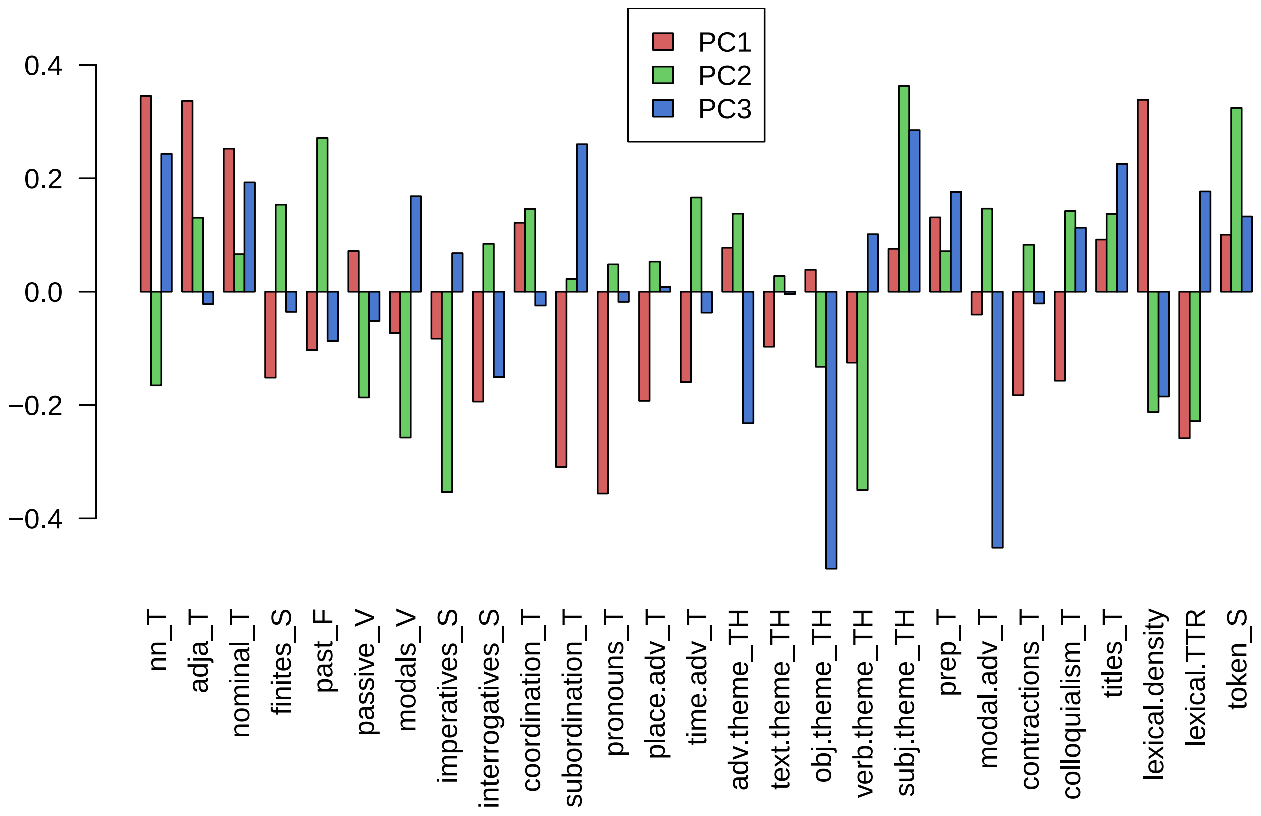
If we want to separate the distribution by both language and translation status, we need combined categories. Setting line colours and styles appropriately gives the visual impression of two separate categorizations.

```
MetaC <- transform(MetaC, lang_status=paste(language, status))
col.vals <- corpora.palette("simple")[c(1,2,1,2)]
lty.vals <- c("solid", "solid", "32", "32")
discriminant.plot(Proj4[, 3], NULL, MetaC$lang_status,
                  rug=TRUE, xlab="", col.vals=col.vals, lty.vals=lty.vals)
```



The **feature weights** for each PCA dimension are simply the coordinates of the corresponding basis vector. Let us visualize the weights for the first 3 PCA dimensions in a combined barplot. Some fiddling with graphics parameters is necessary to obtain a nicely readable plot. We need to transpose the weights matrix (`t(weights)`) in order to get the desired grouping in the barplot.

```
weights <- PCA$basis("complement", dim=1:3)
par(mar=c(8, 4, 0, 0))
barplot(t(weights), beside=TRUE, col=corpora.palette("muted")[2:4],
        las=2, ylim=c(-.5, .5), legend=TRUE, args.legend=list(x="top"))
```



3 A weakly supervised perspective

3.1 LDA dimensions

Since we want to explore potential *shining-through effects*, the perspective we're most interested in is a dimension that separates English and German originals. We can introduce such a weakly supervised intervention through **linear discriminant analysis** (LDA) and add it to our target space.

GMA objects use reference classes, so we should create a new GMA object (or make a copy of the existing one) for the new analysis.

```
ByLang <- PCA$copy()
```

We can now manually add specific dimensions to the taget space of this object. There is a convenience method for finding such dimensions by LDA, so we just have to specify the categories to be separated. The `idx` argument allows us to determine the discriminant based on a subset of the data (in our case, the English and German originals). We also exclude the two outlier text types, which might easily have a distorting effect on the LDA axis.

```
idx.orig <- MetaC$status == "orig" & !(MetaC$register %in% qw("fiction instruction"))
ByLang$add.discriminant(MetaC$language, idx=idx.orig)
ByLang
```

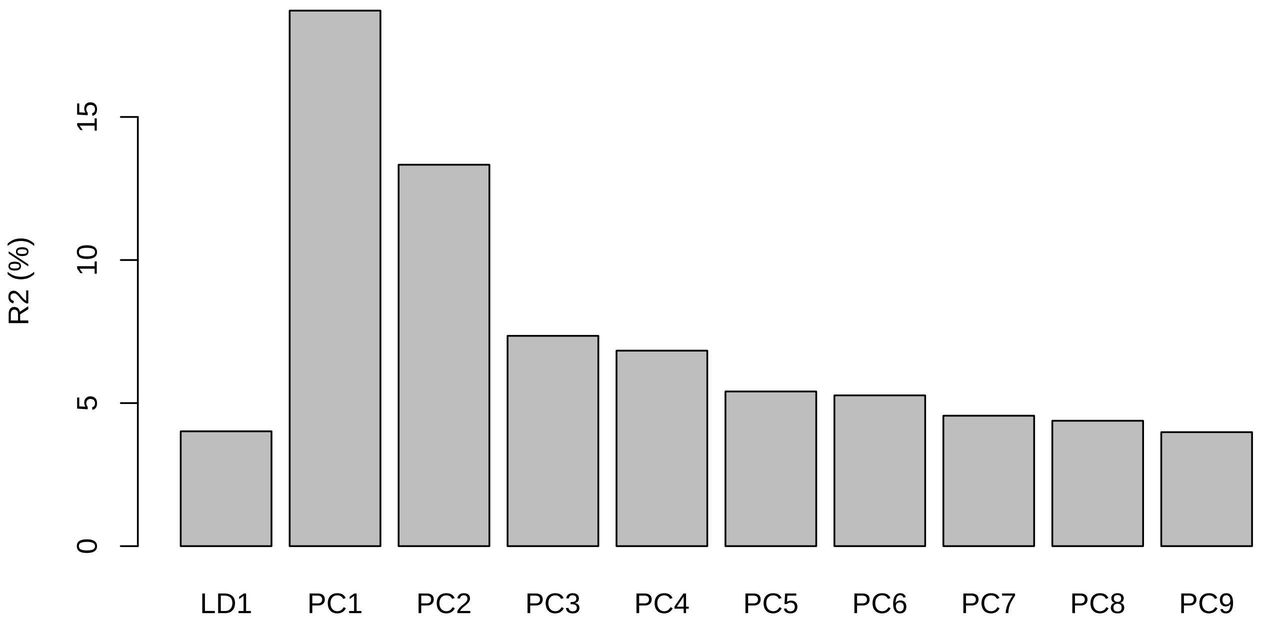
```
## GMA object representing projection of 452 x 27 data matrix into 1-dimensional subspace
```

Now we have a 1-dimensional target space with the English-German discriminant. The PCA of the complement space is automatically updated to reflect the distance information not captured by the discriminant.

Warning: Make sure to execute the code block above only once. If you re-run it, you will add further discriminant dimensions to the target space. It would therefore be safer to make the copy directly in the code block, e.g. with `ByLang <- PCA$copy()$add.discriminant(...)`.

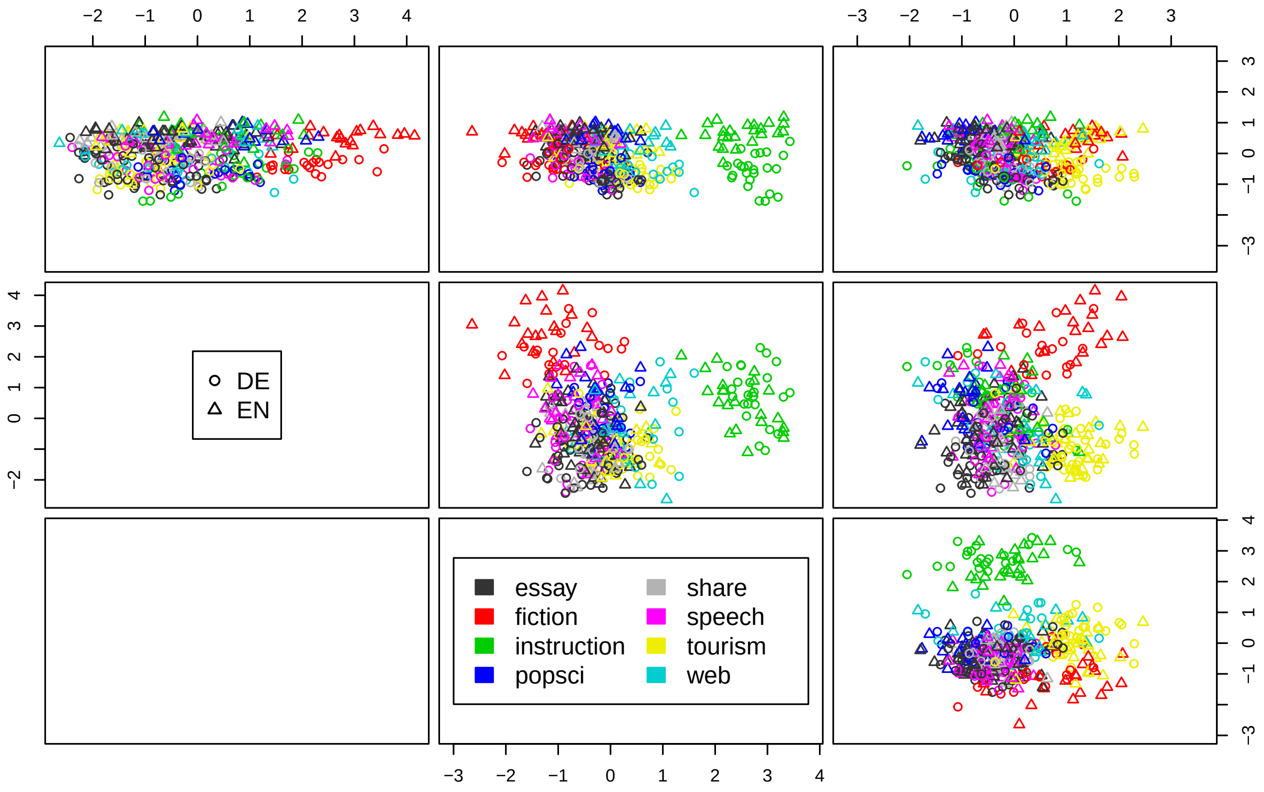
The R^2 values show that the discriminant captures less distance information than the PCA dimensions, explaining why it did not come out as clearly with the unsupervised PCA and required our weakly supervised intervention.

```
barplot(ByLang$R2(dim=1:10), ylab="R2 (%)")
```



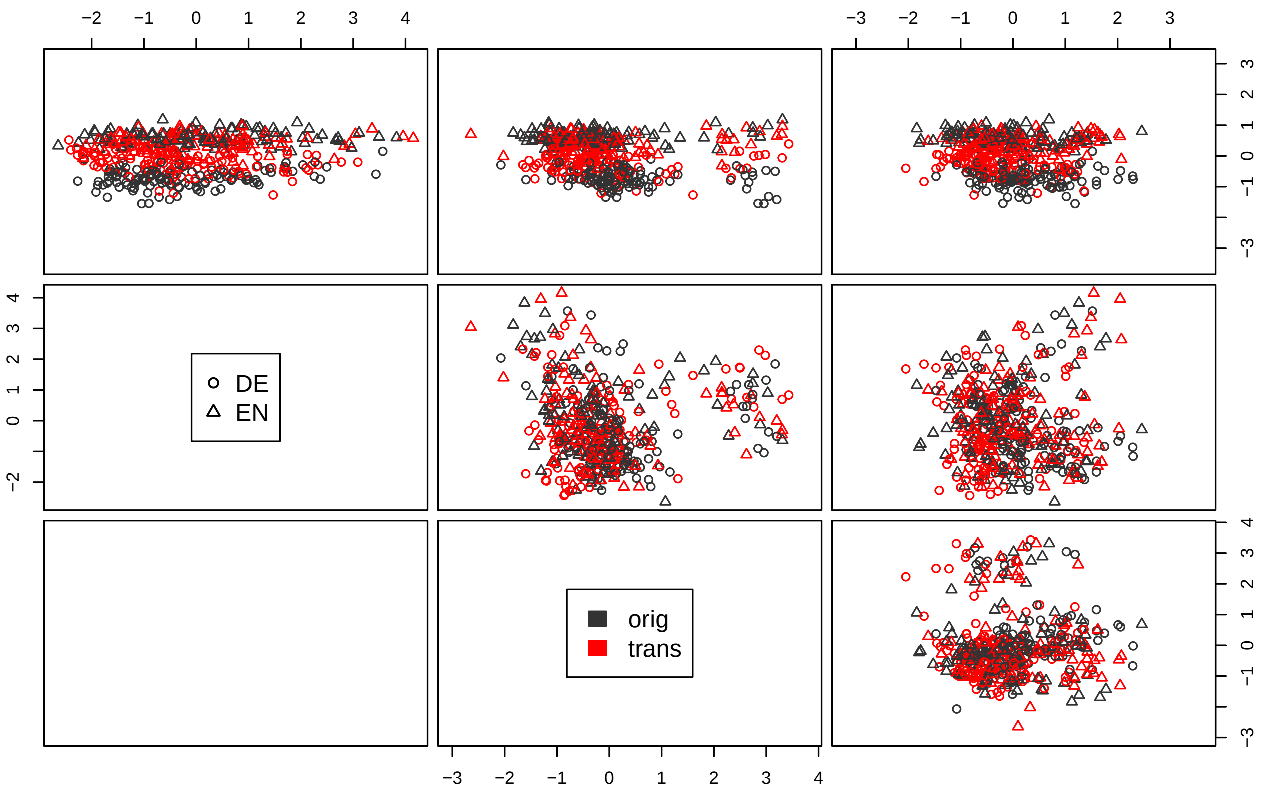
We can now repeat the visualisations above with the target discriminant and the complementary PCA dimensions.

```r
gma.pairs(ByLang$projection("both", dim=1:4),
          Meta=MetaC, compact=TRUE, iso=TRUE,
          col=register, pch=language)
```



What we really want to see, of course, are the positions of translated texts vs. originals, so we change the metadata to be displayed.

```r
gma.pairs(ByLang$projection("both", dim=1:4),
          Meta=MetaC, compact=TRUE, iso=TRUE,
          col=status, pch=language)
```

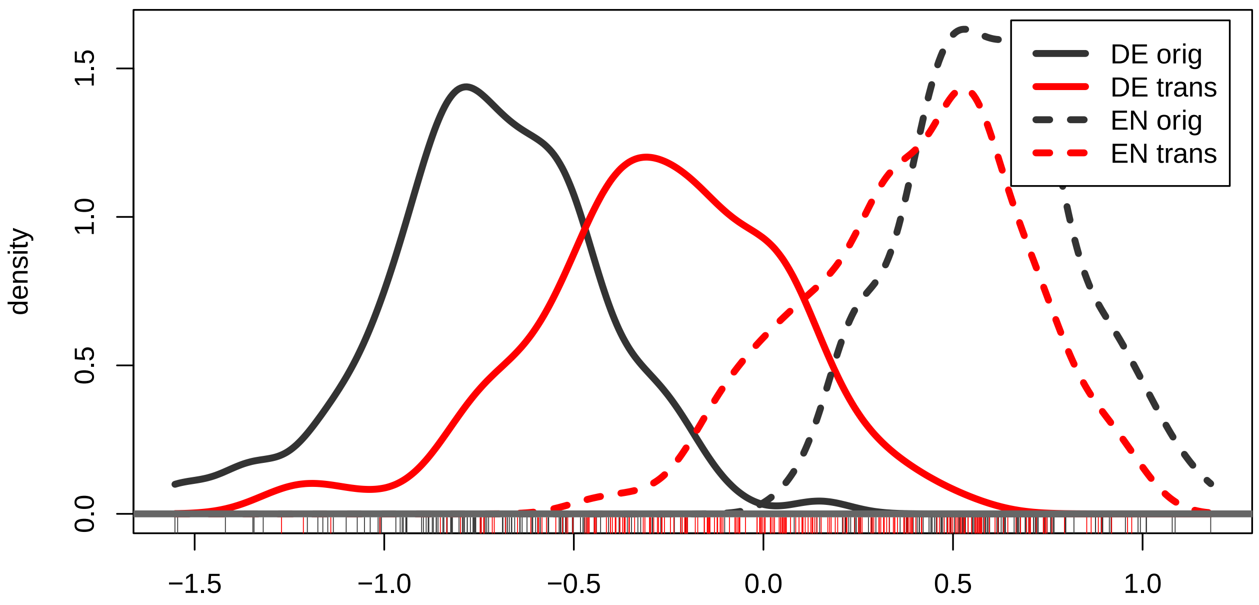


Of course, it's much nicer to see in 3D if you have a working RGL installation.

```
gma.3d(ByLang$projection("both", dim=1:3),
       Meta=MetaC, iso=TRUE, legend=TRUE,
       col=status, pch=language, size=.08)
view3d(theta=0, phi=0, zoom=.7) # reset to front view
```

The discriminant plot now shows clear evidence for shining through and the higher status of English. We've already annotated the combination of language and status in `MetaC` above, so we don't have to repeat this now.

```
col.vals <- corpora.palette("simple")[c(1,2,1,2)]
lty.vals <- c("solid", "solid", "32", "32")
discriminant.plot(ByLang$projection(), NULL, MetaC$lang_status,
                  rug=TRUE, xlab="", col.vals=col.vals, lty.vals=lty.vals)
```



A 3D visualisation of the different text types for original texts suggests that the outlier registers are divergent between the English and German subcorpora and would in fact have distorted the discriminant axis. (Try without excluding them from the discriminant and see what happens!)

```r
gma.3d(ByLang$projection("both", dim=1:3),
       Meta=MetaC, iso=TRUE, legend=TRUE, select=(status=="orig"),
       col=register, pch=language, size=.08)
view3d(theta=0, phi=0, zoom=.7) # reset to front view
```

3.2 Interpreting feature weights

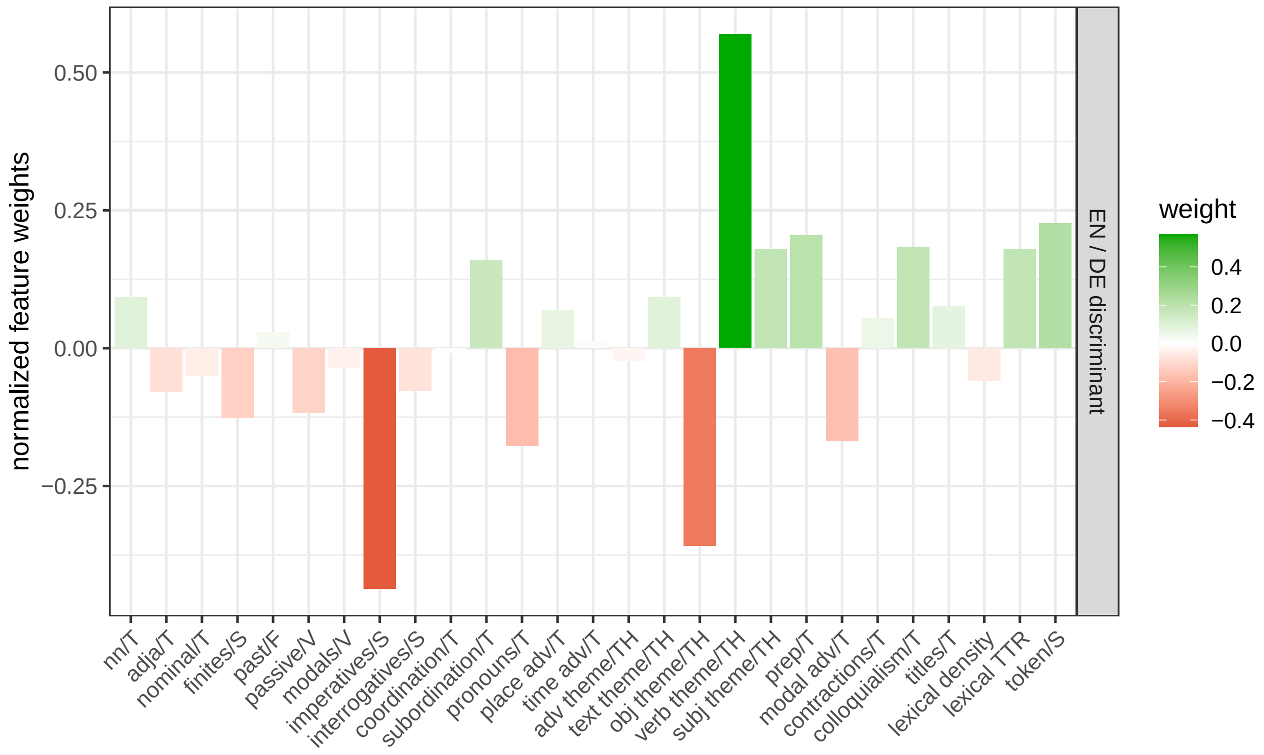
As before, we can visualise feature weights in the form of a barplot. First, let us reformat the feature names so they are more readable.

```r
fnames <- colnames(ZLC)
fnames <- gsub(".", " ", fnames, fixed=TRUE)
fnames <- gsub("_", "/", fnames, fixed=TRUE)
fnames
```

```
##  [1] "nn/T"               "adja/T"            "nominal/T"         "finites/S"
##  [5] "past/F"             "passive/V"         "modals/V"          "imperatives/S"
##  [9] "interrogatives/S" "coordination/T"    "subordination/T"   "pronouns/T"
## [13] "place adv/T"        "time adv/T"        "adv theme/TH"      "text theme/TH"
## [17] "obj theme/TH"       "verb theme/TH"     "subj theme/TH"     "prep/T"
## [21] "modal adv/T"        "contractions/T"    "colloquialism/T"   "titles/T"
## [25] "lexical density"    "lexical TTR"       "token/S"
```

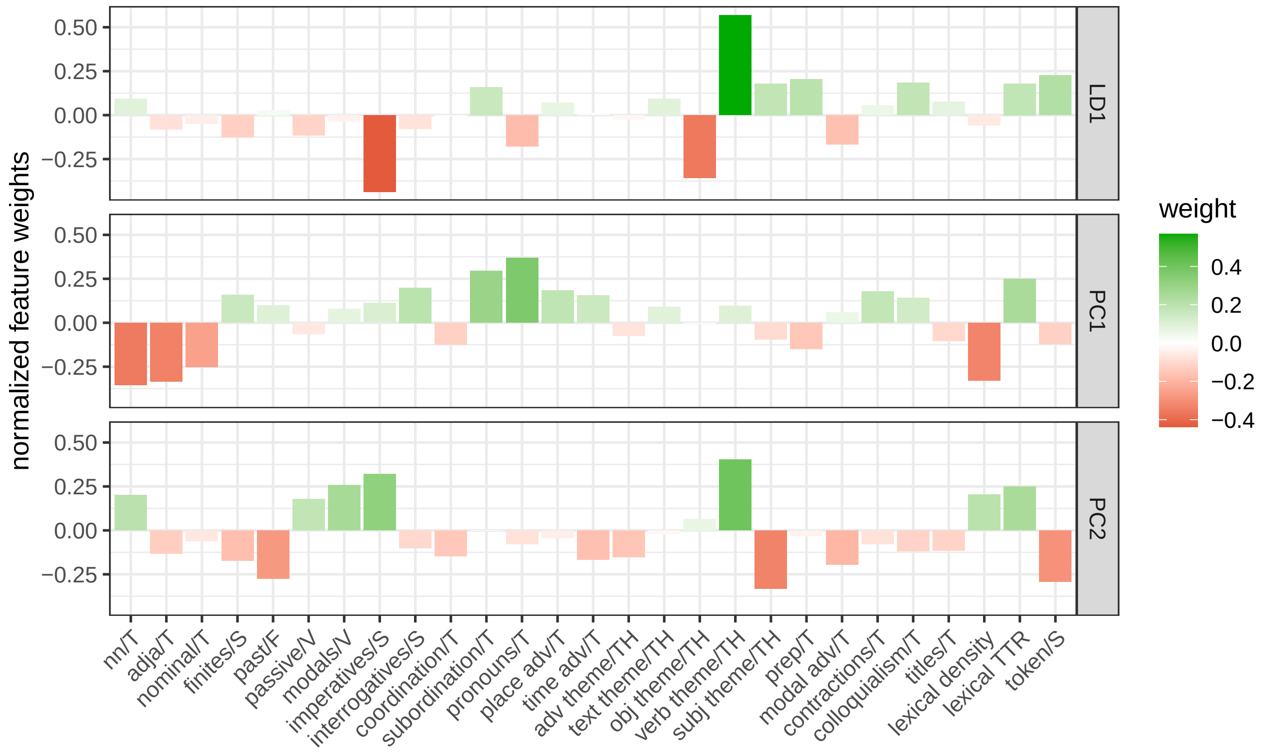
The support function `gma.plot.weights()` conveniently produces a nicely formatted barplot:

```r
gma.plot.weights(ByLang$basis(), names="EN / DE discriminant", feature.names=fnames)
```



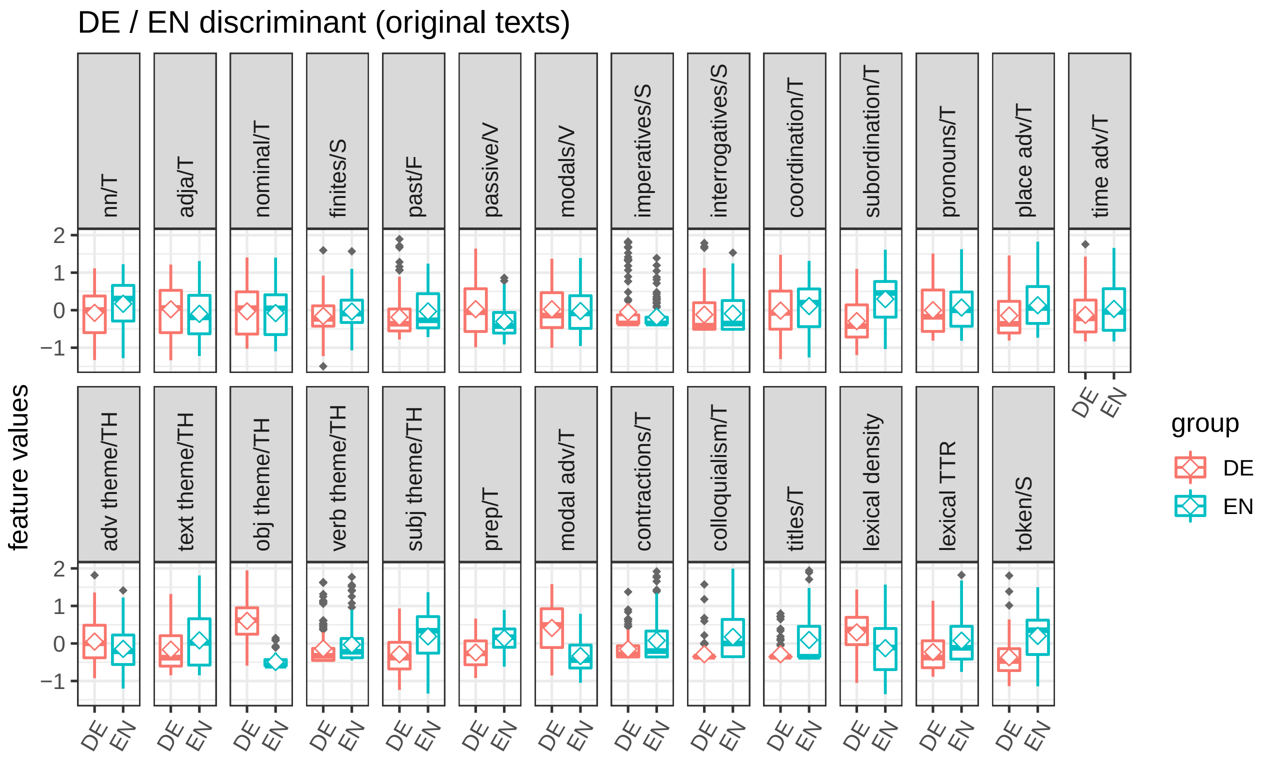
We can also use it to visualise feature weights for multiple dimensions.

```
gma.plot.weights(ByLang$basis("both", dim=1:3), feature.names=fnames)
```



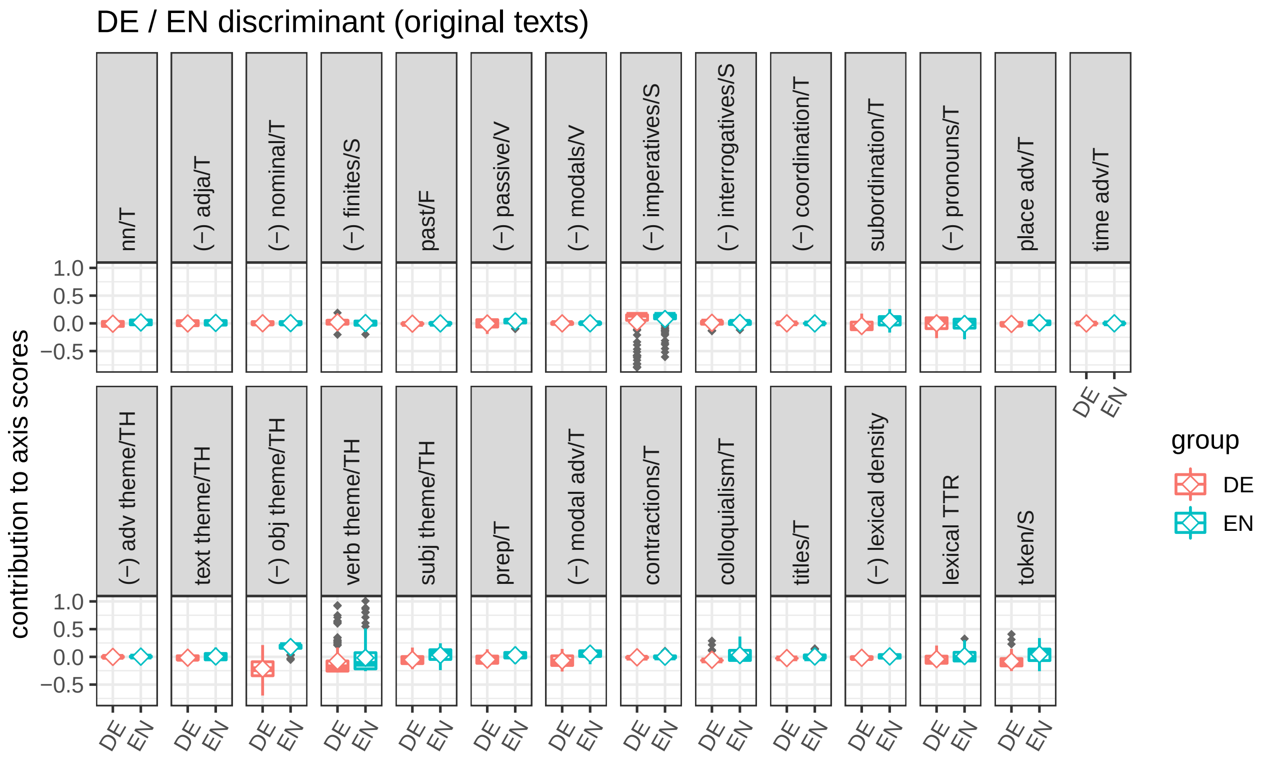
The **interpretation of feature weights** is less straightforward for an LDA dimension: a positive weight does not necessarily indicate that English texts have substantially larger feature values than German texts (and vice versa for a negative weight). It is therefore recommended to look at the individual feature distributions in English and German originals.

```
discriminant <- ByLang$basis()
gma.plot.features(ZLC, MetaC, "features", weights=discriminant, id.var="id",
                  group=language, subset=(status == "orig"), feature.names=fnames,
                  main="DE / EN discriminant (original texts)")
```



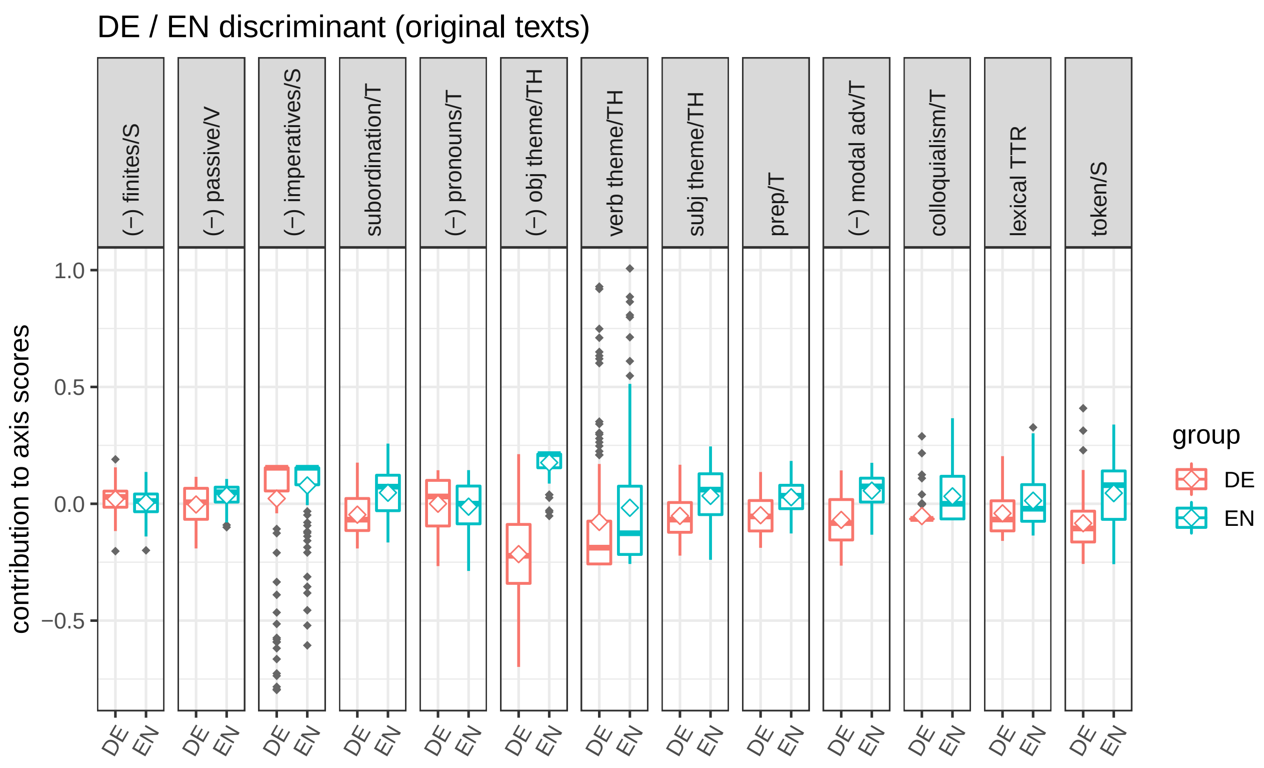
It is also striking that some features (e.g. *lexical density*) have very low weights even though there seems to be a noticeable difference between English and German texts. Such distribution plots often show clearer and more easily interpretable patterns if feature values are multiplied by the corresponding weights, showing the **contribution** of each feature to discriminant scores (i.e. how much it pushes English texts to the positive side of the dimensions and German texts to the negative side).

```
gma.plot.features(ZLC, MetaC, "contribution", weights=discriminant, id.var="id",
                  group=language, subset=(status == "orig"), feature.names=fnames,
                  main="DE / EN discriminant (original texts)")
```



This plot shows which features contribute substantially to the distinction between English and German originals and how they achieve this effect. The relevant parts are brought out even more clearly if we focus on features with relatively large weights (e.g. above 0.1).

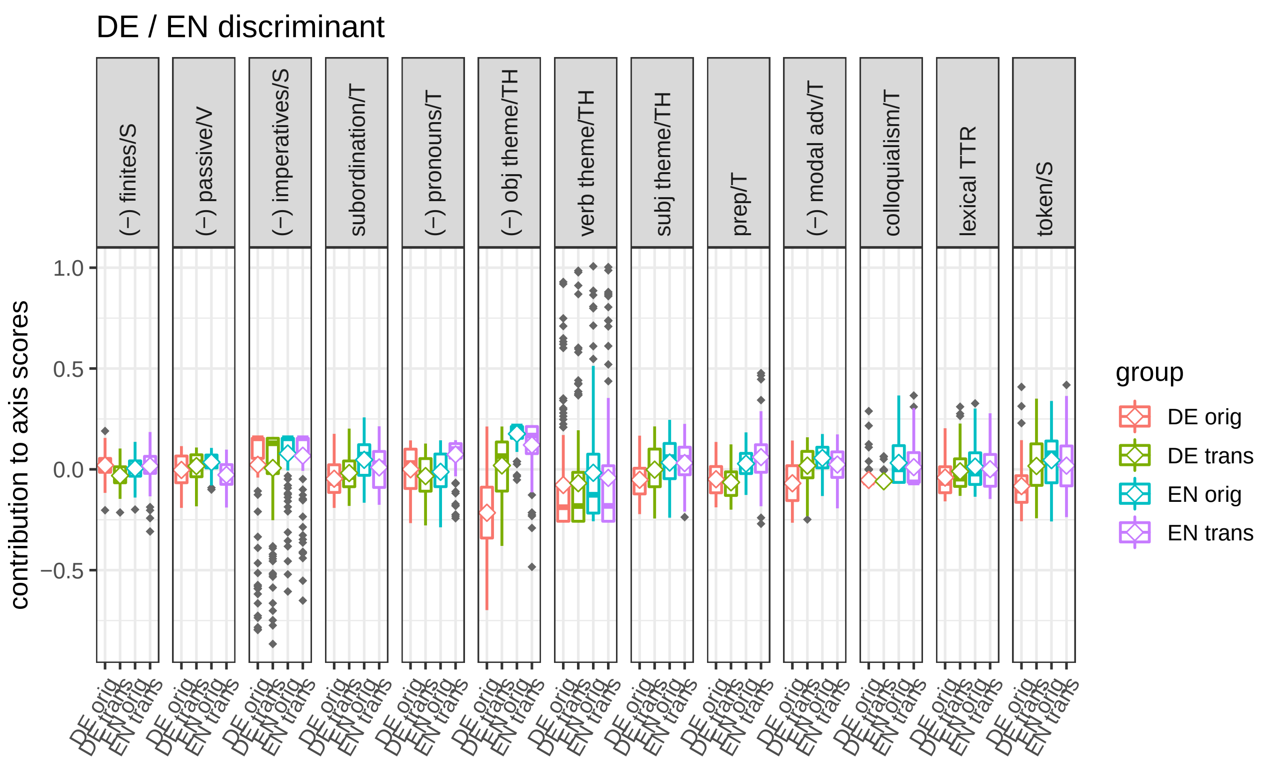
```
fselect <- abs(discriminant) > 0.1
gma.plot.features(ZLC, MetaC, "contribution", weights=discriminant, id.var="id",
                  group=language, subset=(status == "orig"), feature.names=fnames,
                  select=fselect, nrow=1,
                  main="DE / EN discriminant (original texts)")
```



This plot shows that some features have a strong separating effect (e.g. *obj theme*, *colloquialism* and sentence length *token/S*), whereas others are strongly affected by outliers (*imperatives* and *verb theme*).

Finally, we can use our combined annotation to compare the feature contributions to the discriminant coordinates of translated texts with their contributions to the originals.

```
gma.plot.features(ZLC, MetaC, "contribution", weights=discriminant, id.var="id",
                  group=lang_status, feature.names=fnames, select=fselect, nrow=1,
                  main="DE / EN discriminant")
```



This final plot suggests that shining through into German might be mediated primarily (and plausibly) by sentence length, modal adverbs and objects as themes.