Data preparation for ICE paper

Stella Neumann & Stefan Evert

1 May 2020

Contents

1 English texts from the International Corpus of English (NZ, JA and HK components)

```r
source("../multivar_utils.R")
```

```
## Loading required package: Matrix
```

```r
suppressMessages(library(data.table))
library(ggplot2)
library(ggExtra)
library(grid) # needed to work around bug in ggMarginal
library(seriation)
```

```
## Registered S3 method overwritten by 'seriation':
##   method         from
##   reorder.hclust gclus
```

```r
library(colorspace)
```

1.1 Data set and pre-processing

We begin by loading data tables containig **metadata** and **feature vectors** for the three components of ICE to be analysed

```r
Meta <- fread("ice_metadata.csv", encoding="UTF-8")
setkey(Meta, file)
Features <- fread("ice_features_new.tsv", encoding="UTF-8")
colnames(Features)[1] <- "file" # ID variable is called "file" for historical reasons
setkey(Features, file)
```

Unfortunately, three texts from ICE-NZ consist solely of extra-corpus material (these are cases where ICE "texts" are composites from multiple shorter texts that Stella has split up in pre-processing). We therefore need to reduce the `Meta` table to existing texts.

```
Meta <- Meta[file %in% Features$file]
```

For comparison, we also load the old feature vectors (which included extra-corpus material in the texts). We have to apply the same adjustment as for the metadata.

```
OldFeatures <- fread("ice_features.tsv", encoding="UTF-8")
colnames(OldFeatures)[1] <- "file"
setkey(OldFeatures, file)
OldFeatures <- OldFeatures[file %in% Features$file]
```

There are 3198 rows in all three tables (metadata, new features, and old features) and the text IDs (`file`) match for new features (TRUE) and old features (TRUE). Since we index the data.tables by text ID, which automatically sorts them alphabetically, we cannot randomize the order of the rows as in previous analyses.

1.1.1 Normalization based on word counts

Note that relative frequencies in the feature vectors are based on *word counts* rather than token counts now. This was made possible by the new feature extraction pipeline using the `cwb-featex` tool (and by the fact that ICE is small enough so that the very inefficient query for word counts can still be executed).

1.1.2 Cleanup

First we check the data to see whether we have to clean it up:

- the feature `salutation_S` is excluded because it is extremely sparse (and hence produces extreme z-scores)
- the frequency of attributive adjectives (`atadj_W`) is highly correlated with the overall frequency of adjectives (`adj_W`), especially in the English texts; we therefore use non-attributive ("predicative") adjectives (`predadj_W`) rather than (`adj_W`)
- as `preadj_W` was not included in the old feature matrix, we have to calculate the value there (using a little trickery so it's in the same place as in the new feature matrix despite renamed features)

```
Features[, qw("salutation_S adj_W") := NULL]
OldFeatures[, predadj_W := (adj_W - atadj_W)]
OldFeatures[, adj_W := atadj_W]
OldFeatures[, atadj_W := predadj_W]
OldFeatures[, qw("salutation_S predadj_W") := NULL]
setnames(OldFeatures, qw("adj_W atadj_W"), qw("atadj_W preadj_W"))
```

We also discard 354 texts with less than 100 words or less than 10 sentences because the quantitative features will be too unreliable and are prone to create outliers in the multivariate analysis. Before we do so, let's check whether this affects the sub-corpora in substantially different ways.

```
to.discard <- Features[, !(word >= 100 & sent >= 10)]
table(to.discard, Meta$variety)
```

```
##
## to.discard   HK   JA   NZ
##      FALSE 1113  922  809
##      TRUE   140  133   81
```

This is reasonable, so let's proceed and not forget to filter the old feature vectors, too.

```r
Features <- subset(Features, word >= 100 & sent >= 10)
Meta <- Meta[Features$file, ]
OldFeatures <- OldFeatures[Features$file, ]
```

After cleanup, there are 2844 texts and 44 features in the data set, including sentence, token and word counts. The two tables are still consistent, both for new features (TRUE) and old features (TRUE).

1.2 Text categories

A revised set of text categories has been provided in file `text_categories.csv`, which defines both category labels at three different layers of granularity (with 32, 20 and 12 categories) and the standard ordering of the categories.

```r
TextCat <- fread("text_categories.csv", encoding="UTF-8")
```

We use this information to generate appropriate factor levels and colour coding for later visualisation. First check that there are no unexpected duplicates and full names, short labels and category codes match at every layer.

```r
has.distinct <- function (tbl, n=32)
  stopifnot(length(unique(do.call(paste, as.list(tbl)))) == n)
has.distinct(TextCat[, .(text_cat)])
has.distinct(TextCat[, .(textcat32)])
has.distinct(TextCat[, .(short32)]) # combinations of name, short label
has.distinct(TextCat[, .(code32)])  # and code are necessarily unique
has.distinct(TextCat[, .(textcat20)], 20)
has.distinct(TextCat[, .(short20)], 20)
has.distinct(TextCat[, .(code20)], 20)
has.distinct(TextCat[, .(textcat20, short20, code20)], 20)
has.distinct(TextCat[, .(textcat12)], 12)
has.distinct(TextCat[, .(short12)], 12)
has.distinct(TextCat[, .(code12)], 12)
has.distinct(TextCat[, .(textcat12, short12, code12)], 12)
```

We now collect text category names, short labels and codes in the specified ordering (to be used as factor levels and for labeling visualisations). Note that the levels are aligned at each granularity, so it is easy to map between names, labels and codes. Similar vectors of levels are created for the three varieties and for written vs. spoken mode.

```r
types.variety <- qw("NZ JA HK")
types.mode <- qw("W S")
types.textcat32 <- unique(TextCat$textcat32)
types.short32 <- unique(TextCat$short32)
types.code32 <- unique(TextCat$code32)
types.textcat20 <- unique(TextCat$textcat20)
types.short20 <- unique(TextCat$short20)
types.code20 <- unique(TextCat$code20)
types.textcat12 <- unique(TextCat$textcat12)
types.short12 <- unique(TextCat$short12)
types.code12 <- unique(TextCat$code12)
```

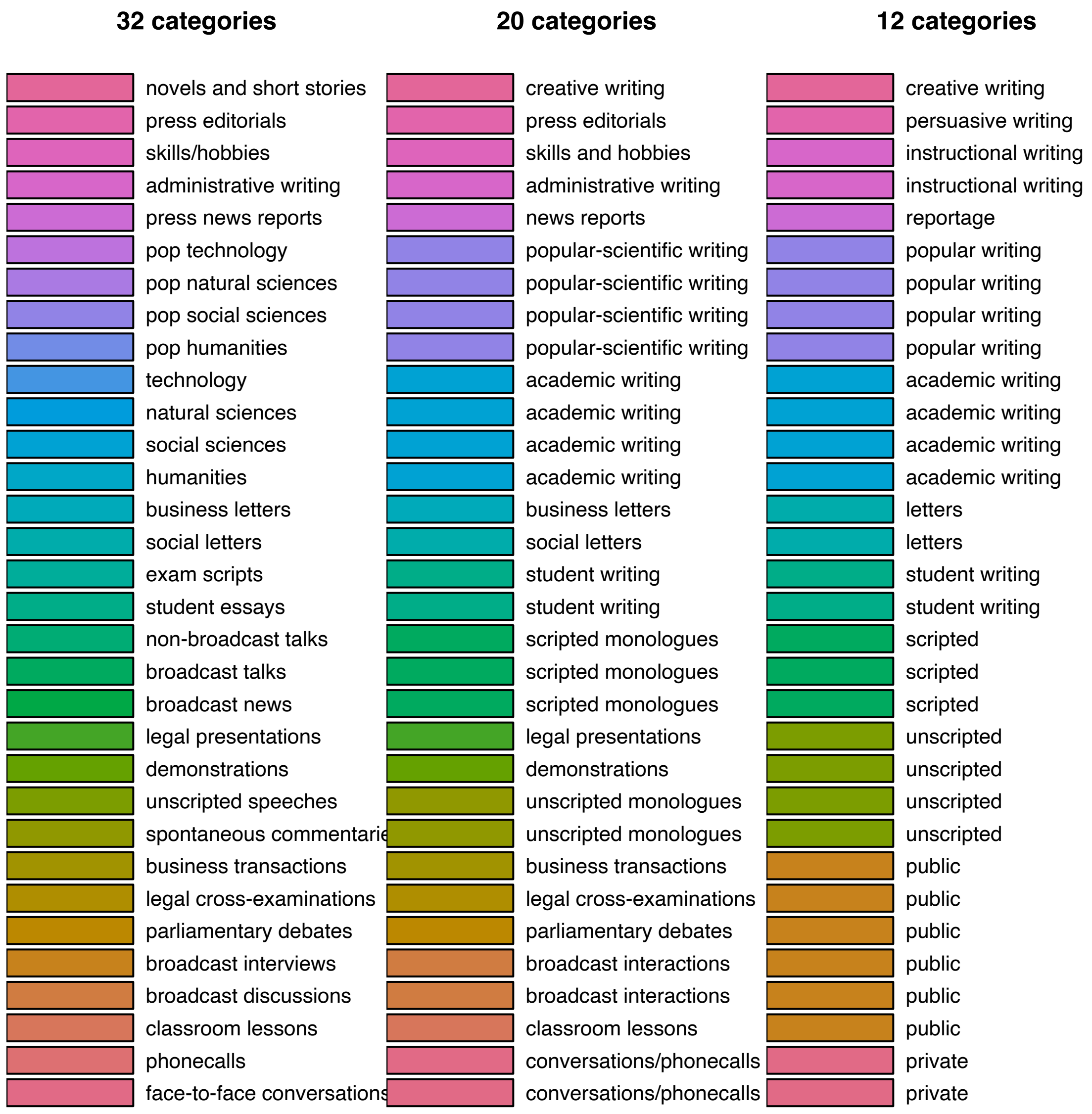
We also generate aligned rainbow colour vectors for the three layers of granularity, with short labels for easy lookup. For the less fine-grained categories, the colour of the "middle" sub-category is selected.

```r
col.vec <- rainbow_hcl(32, c = 80, l = 60)
rainbow.32 <- structure(col.vec, names=types.short32)
tmp <- TextCat[, .(col = col.vec[mean(.I)]), by=short20]
```

```r
rainbow.20 <- structure(tmp$col, names=tmp$short20)
stopifnot(all.equal(names(rainbow.20), types.short20))
tmp <- TextCat[, .(col = col.vec[mean(.I)]), by=short12]
rainbow.12 <- structure(tmp$col, names=tmp$short12)
stopifnot(all.equal(names(rainbow.12), types.short12))
```

An overview table of the colour vectors shows that they are correctly aligned. It is also exported to a PDF file as a handy reference.

```r
par(mfrow=c(1, 3), mar=c(0,0,1,0))
mp <- barplot(rep(1, 32), col=rainbow.32, horiz=TRUE,
              xlim=c(0, 3), xaxt="n", main="32 categories")
text(1.1, mp, types.textcat32, adj=c(0, .5))
barplot(rep(1, 32), col=rainbow.20[TextCat$short20],
        horiz=TRUE, xlim=c(0, 3), xaxt="n", main="20 categories")
text(1.1, mp, TextCat$textcat20, adj=c(0, .5))
barplot(rep(1, 32), col=rainbow.12[TextCat$short12],
        horiz=TRUE, xlim=c(0, 3), xaxt="n", main="12 categories")
text(1.1, mp, TextCat$textcat12, adj=c(0, .5))
```



```r
invisible(dev.copy2pdf(file="pdf_proc/colour_key_textcat.pdf", out.type="cairo"))
```

1.3 Revise metadata

We can now merge the additional information into the Metadata table and remove meta variables that are no longer needed. Before the merge, we make sure that the `text_cat` labels are identical for both data frames.

```r
Meta[, text_cat := sub("[1-3]$", "", text_cat)] # for lookup in TextCat
stopifnot(setequal(TextCat$text_cat, unique(Meta$text_cat)))
Meta[, qw("text_type code_type code_gen text_gen") := NULL]
Meta <- merge(Meta, TextCat, by="text_cat")
Meta[, text_cat := NULL]
```

We also have to make sure that `Meta` is still aligned with `Features`.

```r
setkey(Meta, file)
stopifnot(all.equal(Meta$file, Features$file))
```

Finally, all meta-variables are coded as factors with correct levels and ordering.

```
Meta <- transform(
  Meta,
  variety = factor(variety, levels=types.variety),
  mode = factor(mode, levels=types.mode),
  textcat32 = factor(textcat32, levels=types.textcat32),
  short32 = factor(short32, levels=types.short32),
  code32 = factor(code32, levels=types.code32),
  textcat20 = factor(textcat20, levels=types.textcat20),
  short20 = factor(short20, levels=types.short20),
  code20 = factor(code20, levels=types.code20),
  textcat12 = factor(textcat12, levels=types.textcat12),
  short12 = factor(short12, levels=types.short12),
  code12 = factor(code12, levels=types.code12))
```

1.4 Metadata distributions

Let us now take a look at the metadata categories. We have made sure that filenames are unique IDs: TRUE.

```
knitr::kable(xtabs(~ textcat32 + variety, data=Meta))
```

	NZ	JA	HK
face-to-face conversations	90	98	105
phonecalls	10	15	11
classroom lessons	20	22	22
broadcast discussions	20	21	23
broadcast interviews	10	12	13
parliamentary debates	14	10	11
legal cross-examinations	18	15	16
business transactions	10	12	10
spontaneous commentaries	22	44	20
unscripted speeches	36	32	51
demonstrations	11	12	12
legal presentations	11	20	13
broadcast news	57	32	43
broadcast talks	36	25	45
non-broadcast talks	13	13	17
student essays	29	26	11
exam scripts	14	40	13
social letters	43	93	107
business letters	78	111	133
humanities	10	10	10
social sciences	10	10	16
natural sciences	10	10	12
technology	13	14	15
pop humanities	11	11	20
pop social sciences	12	12	27
pop natural sciences	11	23	21
pop technology	16	23	43
press news reports	95	67	117
administrative writing	14	20	21
skills/hobbies	13	24	27
press editorials	31	22	63
novels and short stories	21	23	45

```
knitr::kable(xtabs(~ textcat20 + variety, data=Meta))
```

	NZ	JA	HK
conversations/phonecalls	100	113	116
classroom lessons	20	22	22
broadcast interactions	30	33	36
parliamentary debates	14	10	11
legal cross-examinations	18	15	16
business transactions	10	12	10
unscripted monologues	58	76	71
demonstrations	11	12	12
legal presentations	11	20	13
scripted monologues	106	70	105
student writing	43	66	24
social letters	43	93	107
business letters	78	111	133
academic writing	43	44	53
popular-scientific writing	50	69	111
news reports	95	67	117
administrative writing	14	20	21
skills and hobbies	13	24	27
press editorials	31	22	63
creative writing	21	23	45

```
knitr::kable(xtabs(~ textcat12 + variety, data=Meta))
```

	NZ	JA	HK
private	100	113	116
public	92	92	95
unscripted	80	108	96
scripted	106	70	105
student writing	43	66	24
letters	121	204	240
academic writing	43	44	53
popular writing	50	69	111
reportage	95	67	117
instructional writing	27	44	48
persuasive writing	31	22	63
creative writing	21	23	45

```
knitr::kable(xtabs(~ mode + variety, data = Meta))
```

	NZ	JA	HK
W	431	539	701
S	378	383	412

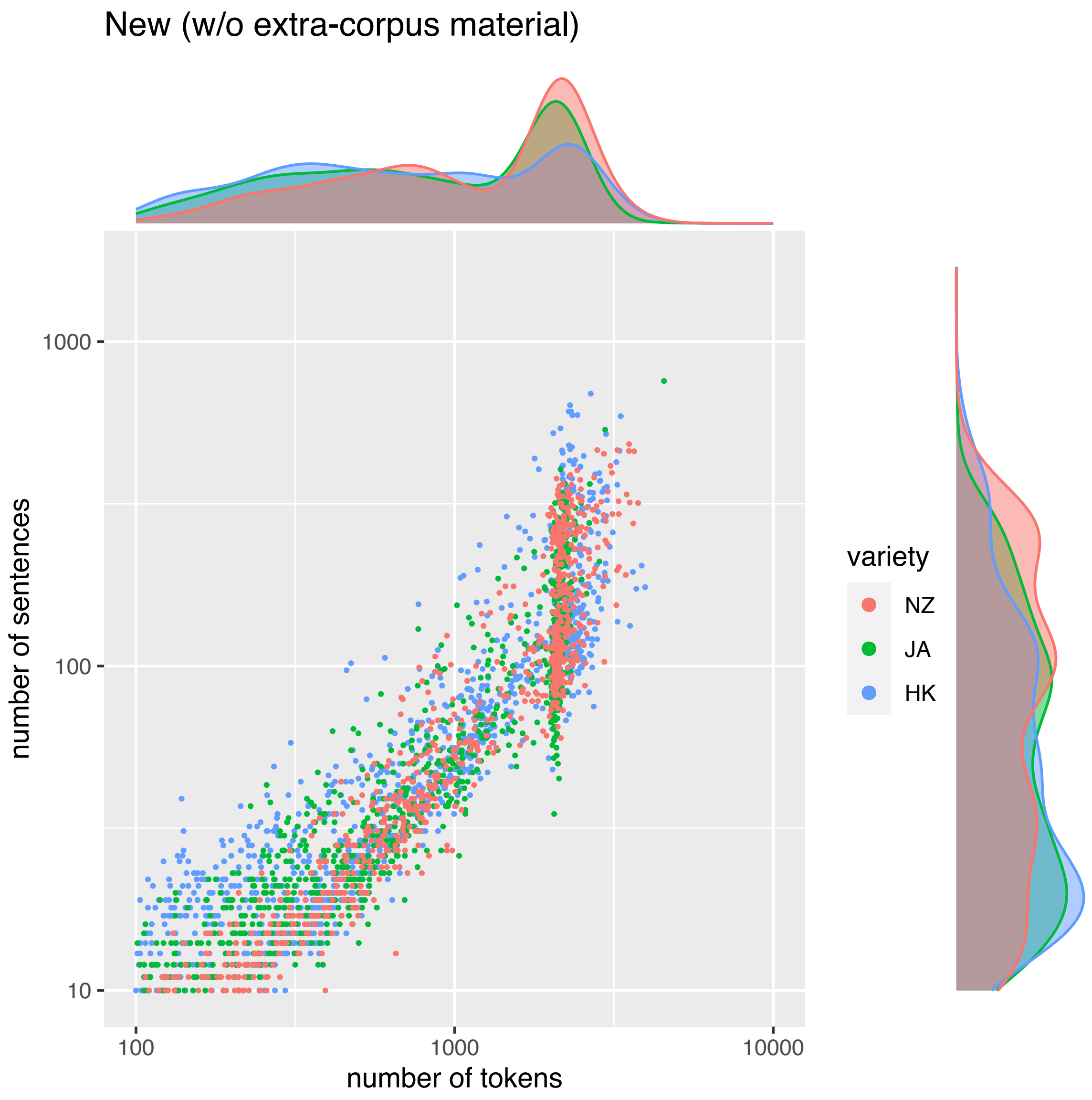
There is some imbalance in the number of text samples in the three varieties and their distribution across text categories, but this is due to the design of and artefacts in the ICE corpora.

1.5 Text lengths

Text lengths vary wildly, including many short texts with highly unreliable feature counts. The distributions look roughly balanced across the three varieties, but there is a large group of texts with approximately 2000 tokens. This indicates a target text size of 2000 words per text.

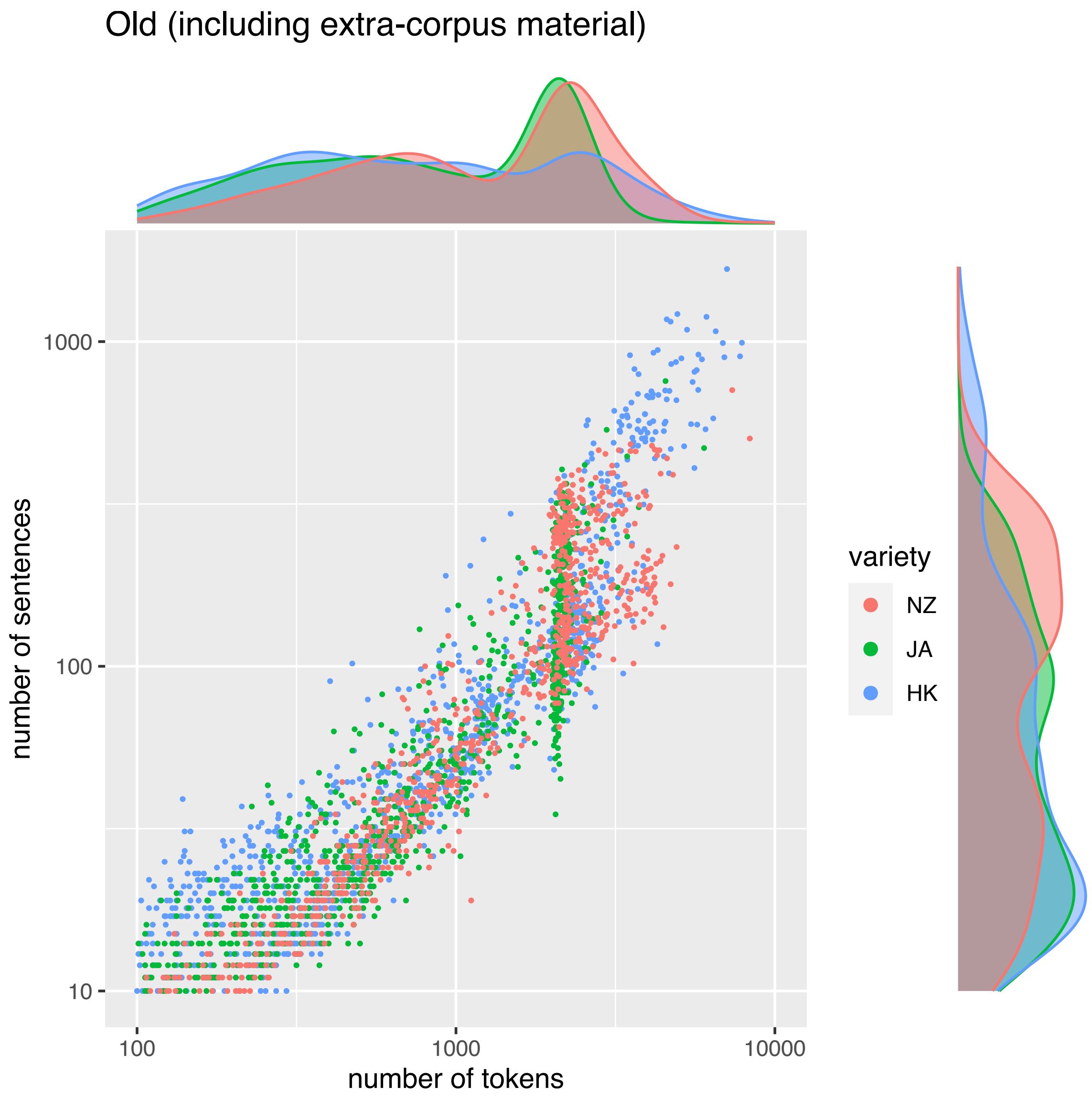
The two plots below compare how excluding extra-corpus material has affected the text sizes.

```r
grid.newpage()
tmp <- cbind(Features, Meta[, .(variety)])
p <- ggplot(tmp, aes(x=word, y=sent, col=variety)) +
  scale_x_log10(limits=c(100, 10000)) + scale_y_log10(limits=c(10, 1700)) +
  geom_point(cex=.4) + labs(x="number of tokens", y="number of sentences") +
  guides(colour = guide_legend(override.aes = list(size=2))) +
  labs(title="New (w/o extra-corpus material)")
p <- ggMarginal(p, groupColour=TRUE, groupFill=TRUE)
grid.draw(p)
```



```r
grid.newpage()
tmp <- cbind(OldFeatures, Meta[, .(variety)])
```

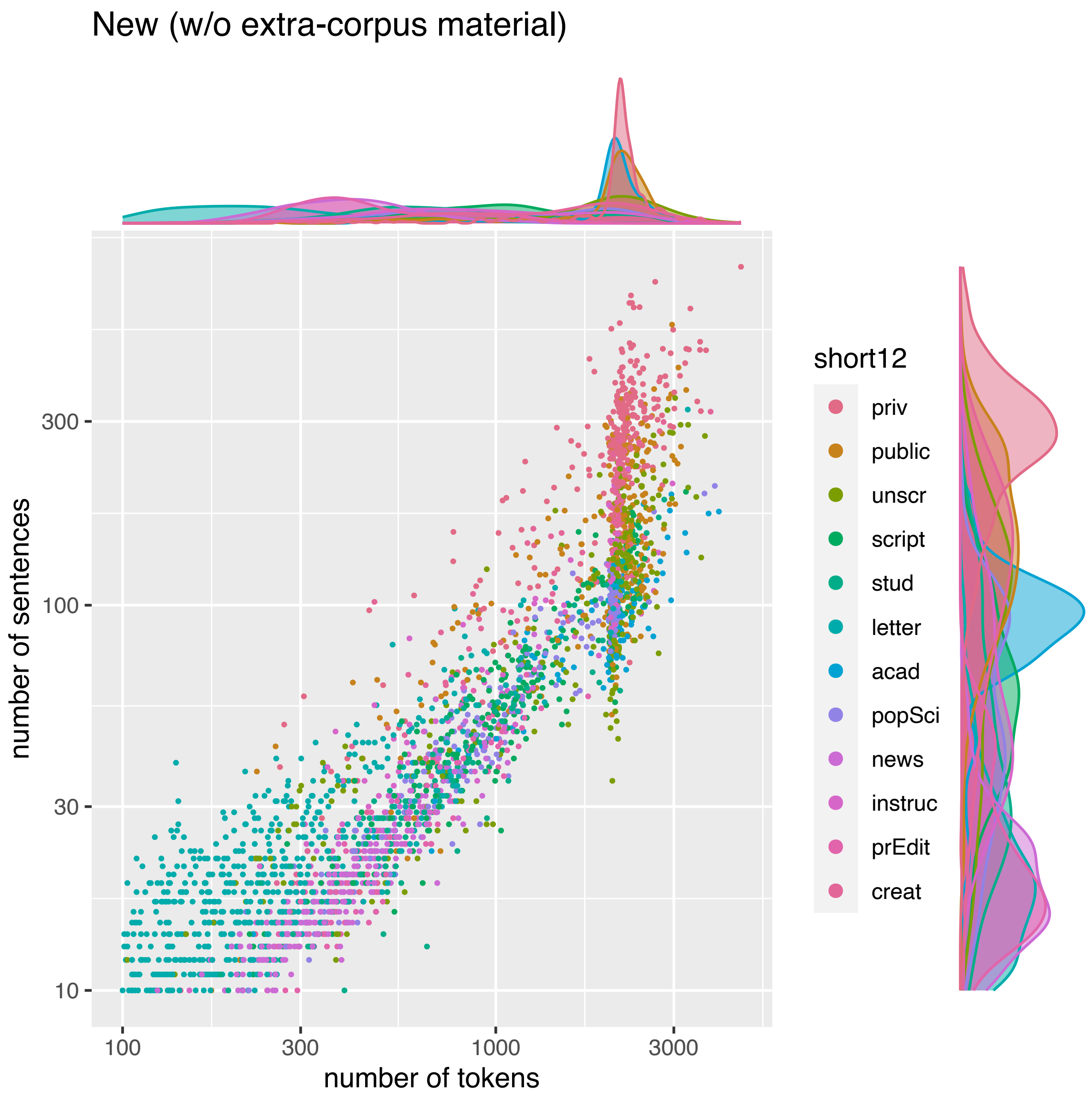
```
p <- ggplot(tmp, aes(x=word, y=sent, col=variety)) +
  scale_x_log10(limits=c(100, 10000)) + scale_y_log10(limits=c(10, 1700)) +
  geom_point(size=.4) + labs(x="number of tokens", y="number of sentences") +
  guides(colour = guide_legend(override.aes = list(size=2))) +
  labs(title="Old (including extra-corpus material)")
p <- ggMarginal(p, groupColour=TRUE, groupFill=TRUE)
grid.draw(p)
```



Also check whether text lengths are different between text categories (without extra-corpus material).

```
grid.newpage()
tmp <- cbind(Features, Meta[, .(variety, short12)])
p <- ggplot(tmp, aes(x=word, y=sent, col=short12)) +
  scale_x_log10() + scale_y_log10() + scale_colour_manual(values=rainbow.12) +
  geom_point(size=.4) + labs(x="number of tokens", y="number of sentences") +
  guides(colour = guide_legend(override.aes = list(size=2))) +
  labs(title="New (w/o extra-corpus material)")
p <- ggMarginal(p, groupColour=TRUE, groupFill=TRUE)
```

```
grid.draw(p)
```



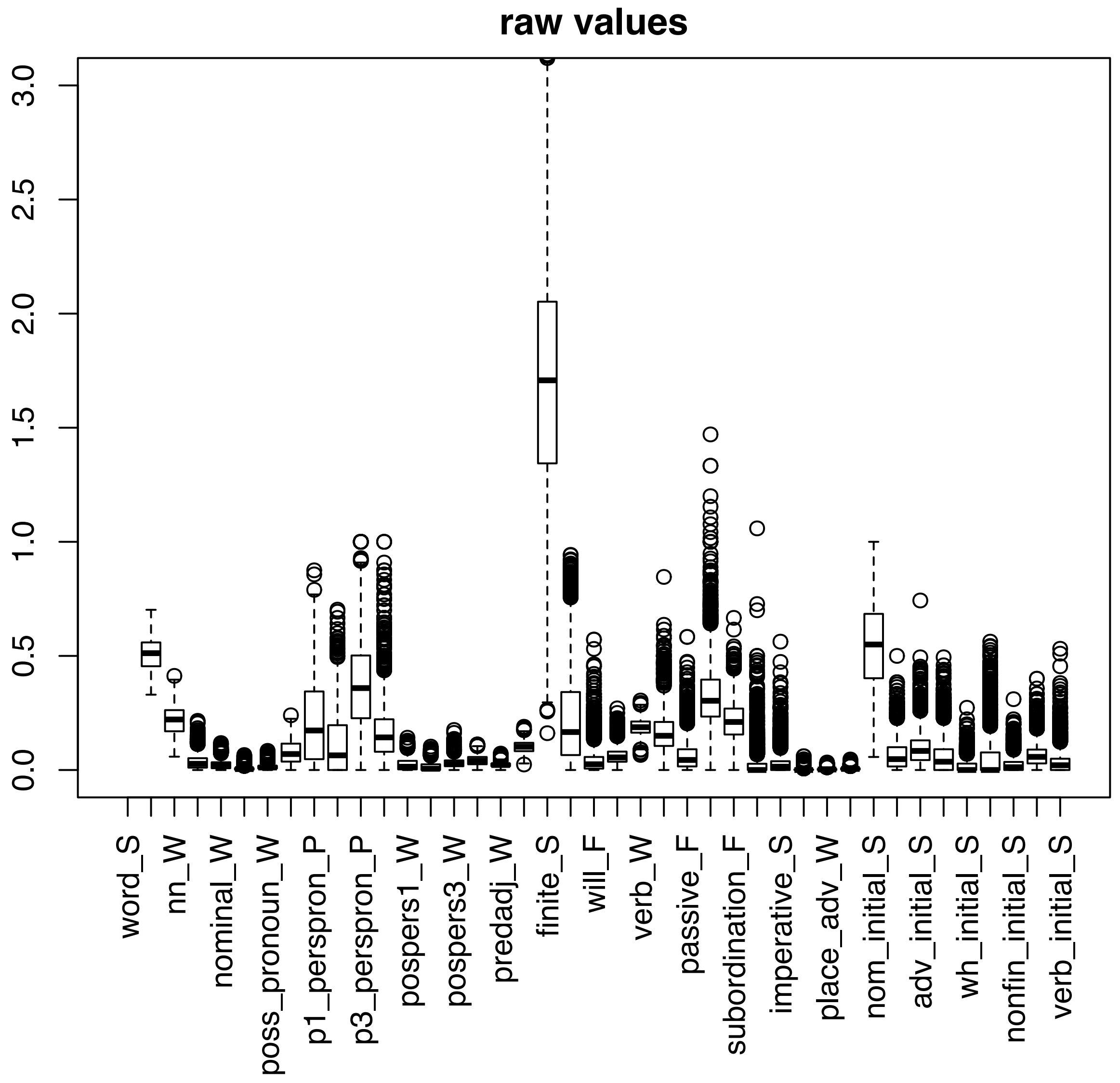
2 Feature distributions

We transform the feature matrix from a data.table to an actual numeric matrix `M`, excluding the word and sentence counts.

```
M <- as.matrix(Features[, -c(1:4)])
rownames(M) <- Features$file
OldM <- as.matrix(OldFeatures[, -c(1:4)])
rownames(OldM) <- OldFeatures$file
```

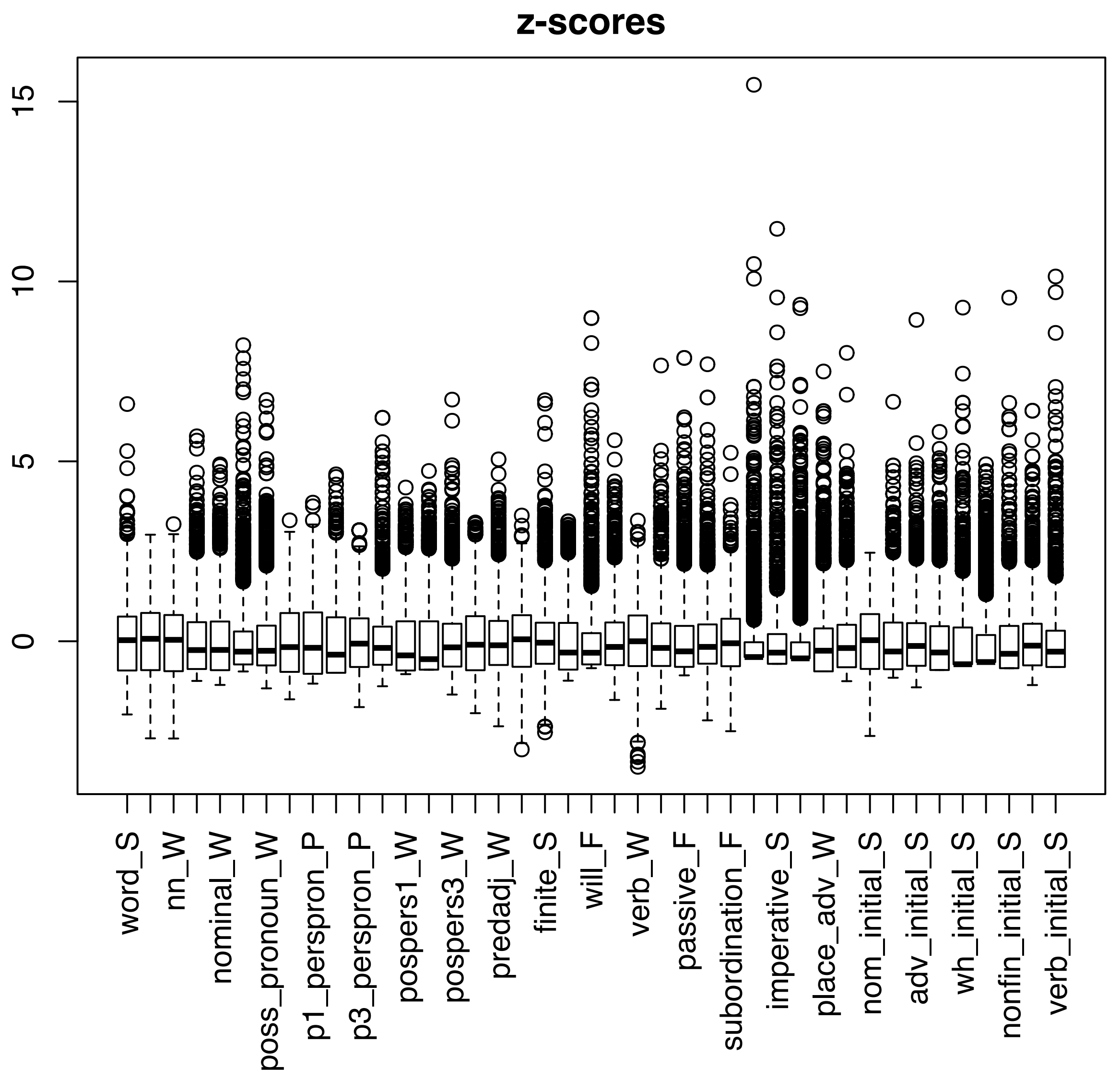
Different features have entirely different ranges and distributions:

```
par(mar=c(8,2,2,0.1))
boxplot(M, ylim=c(0,3), las=3, main="raw values")
```

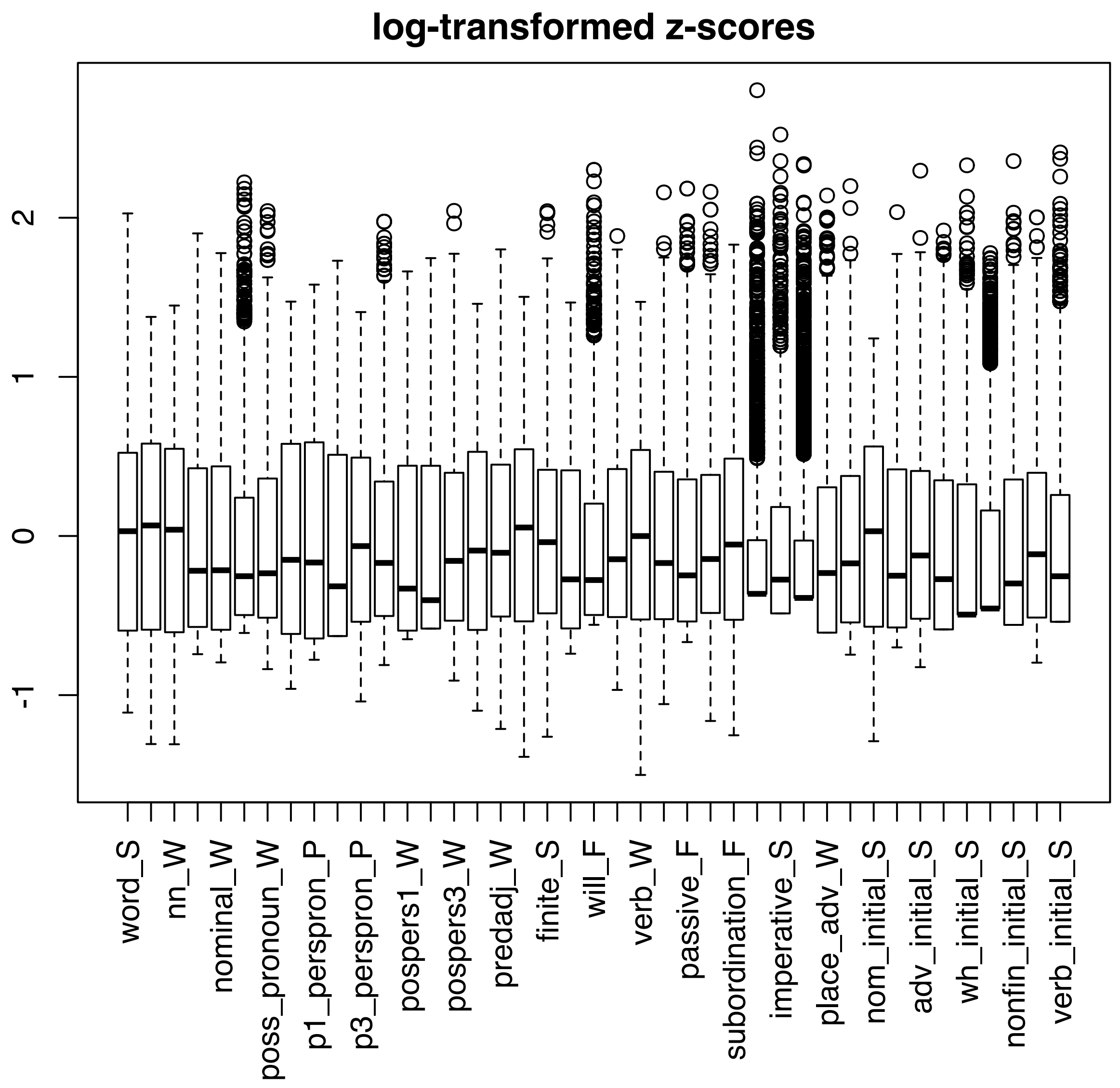


We therefore standardize all features to z-scores as in previous work. The distributions are still highly skewed with some extreme outliers. As an alterantive to removing very sparse feature, we apply a signed logarithmic transformation to deskew the feature distributions.

```r
Z <- scale(M)
par(mar=c(8,2,2,0.1))
boxplot(Z, las=3, main="z-scores")
```



```
ZL <- signed.log(Z)
boxplot(ZL, las=3, main="log-transformed z-scores")
```

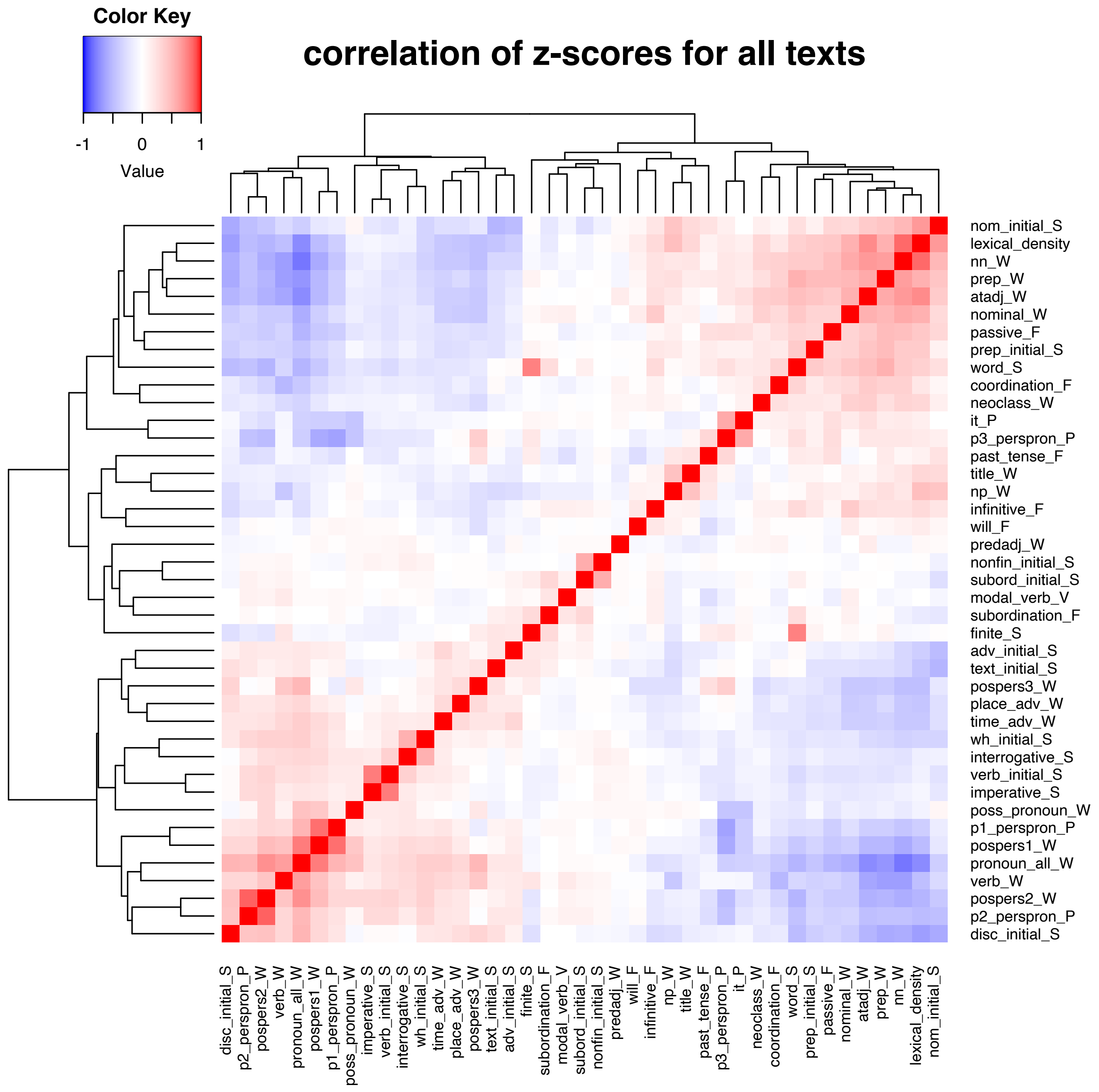


The old features need to be scaled with the same parameters (rather than being standardised independently), so that we can map them into the same space. (In practice, the difference is rarely larger than 1%, so it would not have made a big difference).

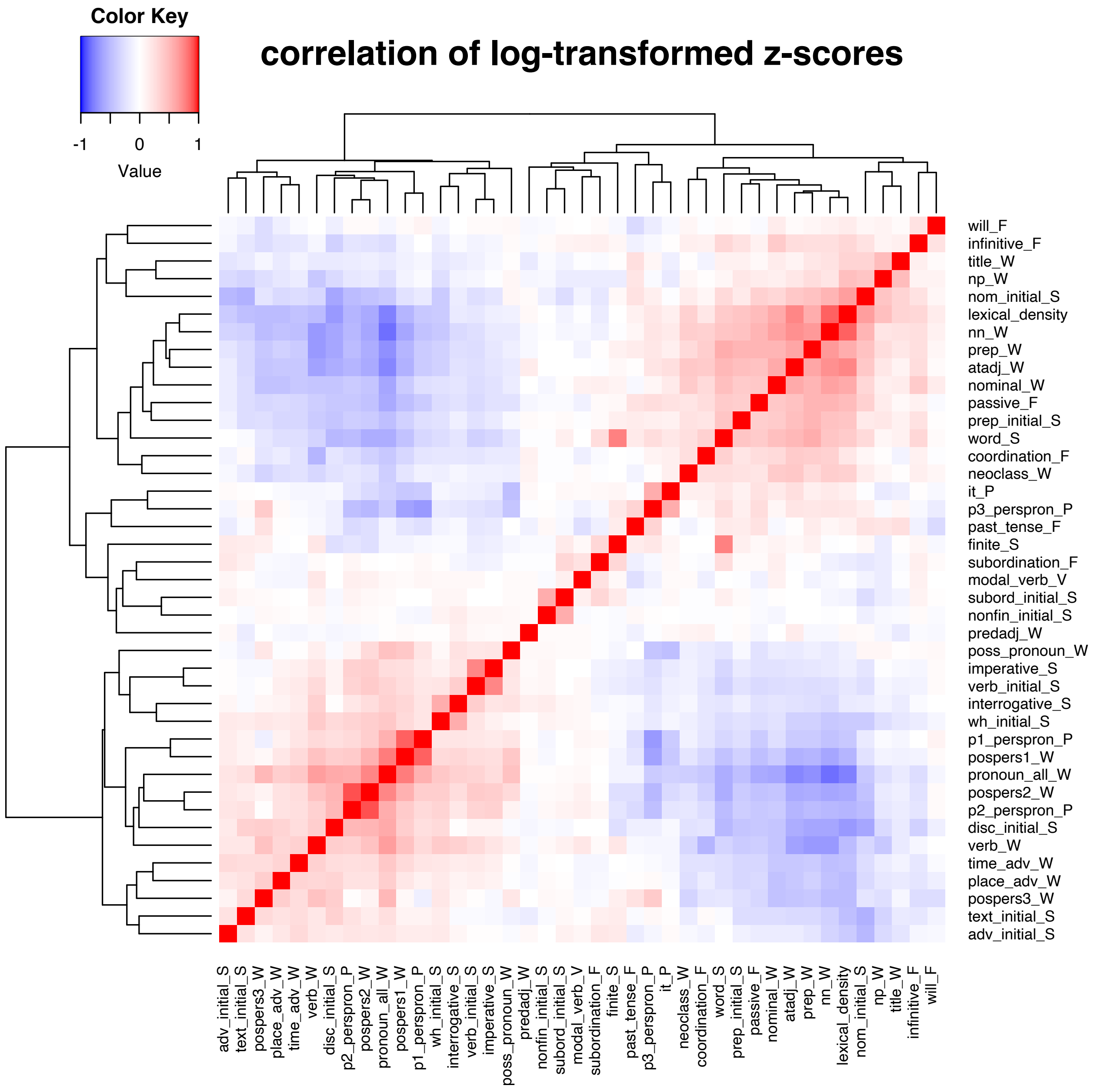
```
OldZ.1 <- scale(OldM)
OldZ <- scale(OldM, center=attr(Z, "scaled:center"), scale=attr(Z, "scaled:scale"))
OldZL <- signed.log(OldZ)
```

We check for collinearities and excessive correlation patterns between the features.

```
fnames <- colnames(Z)
cor.colors <- diverge_hsv(101, power=1)
par(cex=.5)
hmap(cor(Z), zlim=c(-1, 1), col=cor.colors, margins=c(7, 7),
     keysize=1, cexRow=.8, cexCol=.8,
     main="correlation of z-scores for all texts")
```



```r
hmap(cor(ZL), zlim=c(-1, 1), col=cor.colors, margins=c(7, 7),
     keysize=1, cexRow=.8, cexCol=.8,
     main="correlation of log-transformed z-scores")
```



The correlations look reasonable. An overall block structure is visible – which presumably corresponds to the oral-written dimension – but the correlations within the blocks are fairly weak and the features are less directly linked to noun and verb frequency than in Biber's analysis.

3 Serialize pre-processed data set

We also add word and sentence counts to the metadata table, so they can be used for filtering.

```
Meta[, word := Features$word]
Meta[, sent := Features$sent]
Meta[, word.old := OldFeatures$word]
Meta[, sent.old := OldFeatures$sent]
```

And we create nicely readable labels for the features.

```
feature.names <-
  gsub("_+", " ",
       sub("_(\\pL)$", "/\\1",
           colnames(Z), perl=TRUE))
cat(paste(feature.names, collapse=", "), "\n")
```

```
## word/S, lexical density, nn/W, np/W, nominal/W, neoclass/W, poss pronoun/W, pronoun all/W, p1 perspro
```

Finally, save all objects into a single `.rda` file. We also provide a fixed (i.e. reproducible) random ordering index for plots.

```
set.seed(42)
rand.idx <- sample(nrow(Meta))
save(Meta, rand.idx,
     Features, M, Z, ZL,
     OldFeatures, OldM, OldZ, OldZL,
     types.variety, types.mode,
     types.textcat32, types.short32, types.code32,
     types.textcat20, types.short20, types.code20,
     types.textcat12, types.short12, types.code12,
     rainbow.32, rainbow.20, rainbow.12, feature.names,
     file="ice_preprocessed.rda")
```