Lab11. Correspondence analysis: CA and MCA

```
# library(FactoMineR)
library(tidyverse)
library(ca)
library(vcd)
theme_set(theme_bw())
```

Grammatical profiles of Russian verbs

In their article "Predicting Russian aspect by frequency across genres" Eckhoff et al. (2017) ask whether the aspect of individual verbs can be predicted based on the statistical distribution of their inflectional forms. The dataset contains a sample of sentences taken from the Russian National Corpus. Each verb was annotated by Mood&Tense, Voice, Aspect and other grammatical features.

```
ru <- read.csv('https://raw.githubusercontent.com/agricolamz/2018-MAG_R_course/master/data/RNCverbSample
str(ru)
```

```
## 'data.frame':    52716 obs. of  14 variables:
##  $ FormTranslit : Factor w/ 18861 levels "absorbirujuschimi",..: 5396 6884 18667 9730 6685 8847 554 5
##  $ LemmaTranslit: Factor w/ 5940 levels "absorbirovat'",..: 1718 2274 5887 3106 2212 2841 171 1633 38
##  $ MoodTense    : Factor w/ 9 levels "gernonpast","gerpast",..: 6 6 6 6 6 5 8 7 7 8 ...
##  $ Trans        : Factor w/ 2 levels "intr","tran": 2 2 2 2 2 2 1 2 2 1 ...
##  $ Voice        : Factor w/ 4 levels "","act","med",..: 2 4 4 2 2 2 2 2 2 2 ...
##  $ VoicePartcp  : Factor w/ 3 levels "","act","pass": 1 3 3 1 1 1 2 1 1 2 ...
##  $ Person       : Factor w/ 4 levels "","1p","2p","3p": 1 1 1 1 1 4 1 1 1 1 ...
##  $ Number       : Factor w/ 3 levels "","pl","sg": 3 3 3 2 2 2 3 1 1 3 ...
##  $ Gender       : Factor w/ 4 levels "","f","m","n": 3 3 3 1 1 1 3 1 1 2 ...
##  $ Long         : Factor w/ 3 levels "","long","short": 1 3 3 1 1 1 2 1 1 2 ...
##  $ AspPair      : logi  NA NA NA NA NA NA ...
##  $ Aspect       : Factor w/ 2 levels "i","p": 2 2 2 1 2 1 1 1 1 1 ...
##  $ Mood         : Factor w/ 5 levels "ger","imper",..: 3 3 3 3 3 3 5 4 4 5 ...
##  $ Tense        : Factor w/ 4 levels "","fut","nonpast",..: 4 4 4 4 4 3 3 1 1 3 ...
```

First, we will do some preprocessing. Here, we join future passives with other passive participles.

```
ru$MoodTense[ru$Voice == 'pass' & ru$MoodTense == 'indicfut'] <- "partcppast"
```

Let's look at top-10 of verbs in the dataset

```
ru %>%
  group_by(LemmaTranslit) %>%
  summarize(count=n()) %>%
  arrange(desc(count))
```

```
## # A tibble: 5,940 x 2
##    LemmaTranslit count
##    <fct>         <int>
##  1 byt'           2763
##  2 moch'          1125
```

```
##  3 stat'           820
##  4 govorit'        525
##  5 skazat'         432
##  6 znat'           334
##  7 imet'           328
##  8 sdelat'         320
##  9 poluchit'       316
## 10 schitat'        305
## # ... with 5,930 more rows
```

1.1 Grammatical profiles

Grammatical profile is a vector that contains the number (or the ratio) of inflectional forms for individual lemmas. We can pick up a subset containing only forms of the verb *chitat'* 'read'.

```
ru.chit <- droplevels(subset(ru, LemmaTranslit == "chitat'"))
```

This is the grammatical profile of *chitat'*

```
print("The grammatical profile of chitat'")
```

```
## [1] "The grammatical profile of chitat'"
```

```
print(table(ru.chit$MoodTense))
```

```
##
##   gernonpast        imper     indicfut indicnonpast    indicpast          inf
##            2            8            1           15           14           22
```

```
print(prop.table(table(ru.chit$MoodTense))*100)
```

```
##
##   gernonpast        imper     indicfut indicnonpast    indicpast          inf
##     3.225806    12.903226     1.612903    24.193548    22.580645    35.483871
```

Now we will calculate the grammatical profile of each verb (which has more than 50 occurrences in our data set) and split the resulting table into two parts: grammatical forms themselves (numeric data, see ttdata below) and metadata (categories labeled in the RNC or by annotators: lemma, trasitivity, aspect).

Table of tense-mood distribution per lemma:

```
tab = table(ru$LemmaTranslit,ru$MoodTense)
#turns the table into a data frame
t = as.data.frame.matrix(tab)
#adds lemmas as a separate column
t$LemmaTranslit = row.names(t)
#adds metadata columns for transitivity and aspect, assuming that these are stable per lemma - just pic
t$Trans = as.factor(unlist(lapply(t$LemmaTranslit, function(x) names(table(droplevels(subset(ru, LemmaT
t$Asp = as.factor(unlist(lapply(t$LemmaTranslit, function(x) names(table(droplevels(subset(ru, LemmaTra
```

Label the biaspectuals 'b':

```
levels(t$Asp) <- c('i','p','b')
t[t$LemmaTranslit=="ispol'zovat'",]$Asp <- 'b'
t[t$LemmaTranslit=="obeschat'",]$Asp <- 'b'
```

Pick out the lemmas with 50 or more occurrences and split the data:

```
tt <- t[rowSums(t[,1:9]) >= 50,]
```

1.2 t-SNE visualization (for numeric variables)

The idea behind this kind of visualisation is to plot different clusters as far from each other as possible (preserving the distance between each pair of clusters). Within each cluster, the points are distributed to show the internal structure of the cluster. Note that unlike PCA, in t-SNE the points' coordinates cannot be interpreted directly (there is no linear mapping of one plane to another), and linear correlations can be misleading.

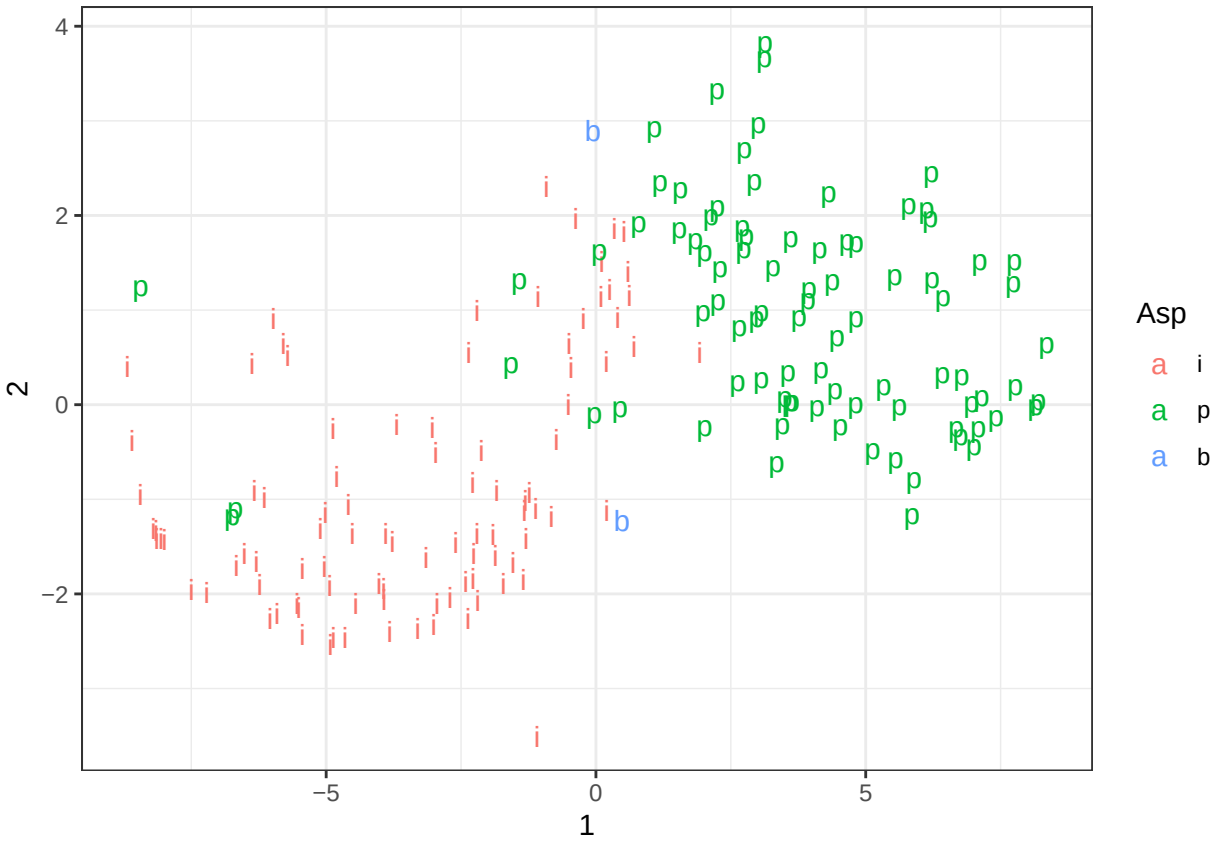
```
library(Rtsne)
ru.tsne <- Rtsne(tt[,1:9],
                 dims=2,
                 perplexity=50,
                 verbose=TRUE,
                 max_iter = 2000)
```

```
## Performing PCA
## Read the 185 x 9 data matrix successfully!
## OpenMP is working. 1 threads.
## Using no_dims = 2, perplexity = 50.000000, and theta = 0.500000
## Computing input similarities...
## Building tree...
## Done in 0.03 seconds (sparsity = 0.942001)!
## Learning embedding...
## Iteration 50: error is 45.482790 (50 iterations in 0.02 seconds)
## Iteration 100: error is 44.903125 (50 iterations in 0.02 seconds)
## Iteration 150: error is 44.613534 (50 iterations in 0.02 seconds)
## Iteration 200: error is 44.573659 (50 iterations in 0.03 seconds)
## Iteration 250: error is 44.910980 (50 iterations in 0.03 seconds)
## Iteration 300: error is 0.293601 (50 iterations in 0.02 seconds)
## Iteration 350: error is 0.171806 (50 iterations in 0.02 seconds)
## Iteration 400: error is 0.160620 (50 iterations in 0.02 seconds)
## Iteration 450: error is 0.159477 (50 iterations in 0.02 seconds)
## Iteration 500: error is 0.161009 (50 iterations in 0.02 seconds)
## Iteration 550: error is 0.161275 (50 iterations in 0.02 seconds)
## Iteration 600: error is 0.160644 (50 iterations in 0.02 seconds)
## Iteration 650: error is 0.160132 (50 iterations in 0.02 seconds)
## Iteration 700: error is 0.160830 (50 iterations in 0.02 seconds)
## Iteration 750: error is 0.160890 (50 iterations in 0.02 seconds)
## Iteration 800: error is 0.158079 (50 iterations in 0.02 seconds)
## Iteration 850: error is 0.156750 (50 iterations in 0.02 seconds)
## Iteration 900: error is 0.158054 (50 iterations in 0.02 seconds)
## Iteration 950: error is 0.157616 (50 iterations in 0.02 seconds)
## Iteration 1000: error is 0.158853 (50 iterations in 0.02 seconds)
## Iteration 1050: error is 0.158439 (50 iterations in 0.02 seconds)
## Iteration 1100: error is 0.157696 (50 iterations in 0.02 seconds)
## Iteration 1150: error is 0.156976 (50 iterations in 0.02 seconds)
## Iteration 1200: error is 0.158397 (50 iterations in 0.02 seconds)
## Iteration 1250: error is 0.160100 (50 iterations in 0.02 seconds)
## Iteration 1300: error is 0.160685 (50 iterations in 0.02 seconds)
## Iteration 1350: error is 0.160468 (50 iterations in 0.02 seconds)
## Iteration 1400: error is 0.160518 (50 iterations in 0.02 seconds)
```

```
## Iteration 1450: error is 0.161011 (50 iterations in 0.02 seconds)
## Iteration 1500: error is 0.161054 (50 iterations in 0.02 seconds)
## Iteration 1550: error is 0.161539 (50 iterations in 0.02 seconds)
## Iteration 1600: error is 0.161060 (50 iterations in 0.02 seconds)
## Iteration 1650: error is 0.159614 (50 iterations in 0.02 seconds)
## Iteration 1700: error is 0.160110 (50 iterations in 0.02 seconds)
## Iteration 1750: error is 0.161284 (50 iterations in 0.02 seconds)
## Iteration 1800: error is 0.160228 (50 iterations in 0.02 seconds)
## Iteration 1850: error is 0.161134 (50 iterations in 0.02 seconds)
## Iteration 1900: error is 0.159263 (50 iterations in 0.02 seconds)
## Iteration 1950: error is 0.160308 (50 iterations in 0.02 seconds)
## Iteration 2000: error is 0.161113 (50 iterations in 0.02 seconds)
## Fitting performed in 0.75 seconds.
```

```
tt <- cbind(tt, ru.tsne$Y)

tt %>%
  ggplot(aes(`1`, `2`, label = Asp, color = Asp))+
  geom_text()
```



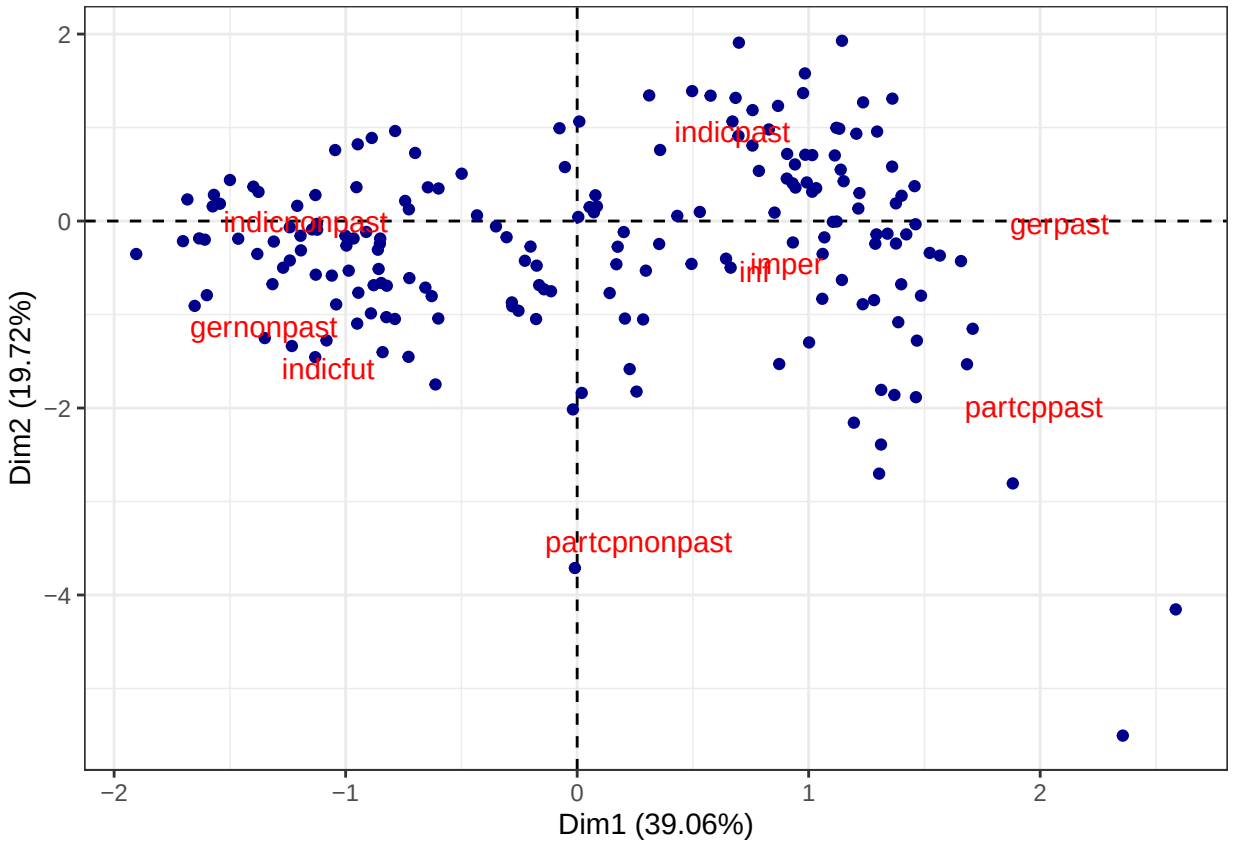
CA

```
tt_ca <- ca(tt[,1:9])

tt_col <- data.frame(tt_ca$colcoord)
tt_col$rows <- rownames(tt_ca$colcoord)
```

```
tt_row <- data.frame(tt_ca$rowcoord)
tt_row$rows <- rownames(tt_ca$rowcoord)

tt_col %>%
  ggplot(aes(Dim1, Dim2))+
  geom_hline(yintercept = 0, linetype = 2)+
  geom_vline(xintercept = 0,linetype = 2)+
  geom_point(data = tt_row, aes(Dim1, Dim2), color = "darkblue")+
  geom_text(aes(label = rows), color = "red")+
  labs(x = "Dim1 (39.06%)",
       y = "Dim2 (19.72%)")
```



MCA

Dataset and description from paper by Natalia Levshina. Modern standard Dutch has two periphrastic causatives with the infinitive: the constructions with doen 'do' and laten 'let'. The study is based on an 8-million token corpus of Netherlandic and Belgian Dutch. After the manual cleaning, there were left with 6,808 observations, which were then coded for seven semantic, syntactic, geographical and thematic variables.

- Aux — a factor that specifies the causative auxiliary with levels laten and doen.
- Country — a factor with levels NL (the Netherlands) and BE (Belgium).
- Causation — a factor that describes the type of causation with levels Affective, Inducive, Physical and Volitional
- EPTrans — a factor that specifies the transitivity of the Effected Predicate with levels Intr (intransitive) and Tr (transitive).

- EPTrans1 — a factor with levels Intr and Tr. It is very similar to the previous one, except for a few observations.

```
dutch_caus <- read.csv("https://goo.gl/2yAR3T")
MCA <- MASS::mca(dutch_caus[,-1])
MCA
```

```
## Call:
## MASS::mca(df = dutch_caus[, -1])
##
## Multiple correspondence analysis of 500 cases of 7 factors
##
## Correlations 0.577 0.458  cumulative % explained  9.62 17.25
```

```
dutch_caus <- cbind(dutch_caus, MCA$rs)
variables <- as_data_frame(MCA$cs)
```

```
## Warning: `as_data_frame()` is deprecated, use `as_tibble()` (but mind the new semantics).
## This warning is displayed once per session.
```

```
variables$var_names <- rownames(MCA$cs)
dutch_caus %>%
  ggplot(aes(`1`, `2`))+
  geom_point(aes(color = Aux))+
  stat_ellipse(aes(color = Aux))+
  geom_text(data = variables, aes(`1`, `2`, label = var_names))+
  theme_bw()+
  scale_x_continuous(limits = c(-0.015, 0.02))
```

```
## Warning: Removed 1 rows containing missing values (geom_text).
```