<Z>appendix: data analysis documentation

This appendix describes the R-code that was used for the analyses in the paper. All analyses were carried out using R version 3.0.1 and lme4 version 0.999999-2 (R Development Core Team, 2009).1

<A>an overview of the data

To analyze the ternary genitive alternation in the ARCHER corpus, we created four different csv-files that represent the four alternation contexts that are discussed in the paper. The files that are available are:

<NL>

1. *svsof.csv*: used to analyze the subset of the data containing ***s*-genitives and *of*-genitives** that are interchangeable with each other, but not with NN-genitives (Model 1): *n* = 4195 (*ns*-genitive =831, *nof*-genitive = 3364);
2. *nnvsof.csv*: used to analyze the subset of the data that contains **NN-genitives and *of*-genitives** that are interchangeable with each other, but not with *s*-genitives (Model 2): *n* = 2832 (*n*NN-genitive =905, *nof*-genitive = 1927);
3. *nnvss.csv*: used to analyze the subset of the data that contains **NN-genitives and *s*-genitives** that are interchangeable with each other (Model 3): *n* = 676 (*n*NN-genitive =563, *ns*-genitive = 113);
4. *threewaygen.csv*: used to analyze the subset of the data that contains **NN-genitives versus *of*-genitives versus *s*-genitives—**all occurrences that are interchangeable with *all* other variants (Model 4): *n* = 2927 (*n*NN-genitive =470, *nof*-genitive = 2351, *ns*-genitive = 106).

</NL>

The dependent variable is coded as depvar in the first three datasets (*svsof*, *nnvsof*, *nnvss*). The threeway alternation dataset (*threewaygen*) contains a categorical variable (depvar\_binary) that is used for the logistic regression model for NN-genitive versus not-NN-genitive. The column genitive\_type shows which variant is used.

<A>importing the data in r

If necessary, the working directory should first be set to the appropriate folder using setwd(). Then, the datasets can be imported with read.csv():

<DIS>

svsof <- read.csv("svsof.csv", header=TRUE, sep=";", dec=".")

svsof$depvar <- as.factor(svsof$depvar)

svsof$animacy <- relevel(svsof$animacy, ref = "non-animate")

nnvsof <- read.csv("nnvsof.csv", header=TRUE, sep=";", dec=".")

nnvsof$depvar <- as.factor(nnvsof$depvar)

nnvsof$animacy <- relevel(nnvsof$animacy, ref = "non-animate")

nnvss <- read.csv("nnvss.csv", header=TRUE, sep=";", dec=".")

nnvss$depvar <- as.factor(nnvss$depvar)

nnvss$animacy <- relevel(nnvss$animacy, ref = "non-animate")

threewaygen <- read.csv("threewaygen.csv", header=TRUE, sep=";", dec=".")

threewaygen$animacy <- relevel(threewaygen$animacy, ref = "non-animate")

threewaygen$depvar\_binary <- as.factor(threewaygen$depvar\_binary)

</DIS>

<A>analyses

*<B>Variant frequencies and proportions in real time*

With the CrossTable() function from package gmodels, an overview of the frequencies of the variants per period can be obtained. If the gmodels package is not installed, use install.packages(“gmodels”).

<DIS>

library(gmodels)

CrossTable(svsof$period,

svsof$genitive\_type,

digits=1,

expected=FALSE,

prop.r=TRUE,

prop.c=FALSE,

prop.t=FALSE,

prop.chisq=FALSE,

format="SPSS"

)

</DIS>

The same function can be used for an overview of the data in the *nnvsof*, *nnvss,* and *threewaygen* datasets.

To create a visualization of the proportions, we use the ggplot2 package. First, we create a data frame object that contains the proportions of the variants per period in the dataset.

<DIS>

library(ggplot2)

t1 <- table(svsof$period, svsof$genitive\_type)

df\_svsof\_prop <- data.frame(period = names(prop.table(t1, 1)[,1]),

OF = prop.table(t1, 1)[,1] \* 100,

S = prop.table(t1, 1)[,2] \* 100,

row.names = NULL

)

# reshape data frame using melt() function from reshape package

library(reshape)

df\_svsof\_prop <- melt(df\_svsof\_prop, id = "period",

variable\_name = "genitive\_type")

# add newline to period label for visualization

df\_svsof\_prop$period <- gsub("-", "-\n", levels(df\_svsof\_prop$period))

</DIS>

We also add the absolute frequencies of the genitive variants per period to this data frame.

<DIS>

df\_svsof <- data.frame(period = names(t1[,1]),

OF = as.numeric(t1[,1]),

S = as.numeric(t1[,2])

)

df\_svsof <- melt(df\_svsof, id = "period", variable\_name = "genitive\_type")

df\_svsof\_prop$absvals <- df\_svsof$value

</DIS>

Then we create the plot and enhance the layout.

<DIS>

svsof\_plot <- ggplot(df\_svsof\_prop,

aes(period, value, group = genitive\_type)) +

geom\_point(size = 2.5) +

geom\_line(size = 1.3, aes(linetype = genitive\_type)) +

scale\_linetype\_manual(values = c("solid", "dotted"),

name = "genitive\ntype") +

labs(list(title = "s-genitive vs. of-genitive", x = "period",

y = "proportion (%)")) +

ylim(-5,100) +

theme\_bw() +

theme(plot.title = element\_text(size = 16, vjust = 1.5),

axis.title = element\_text(size = 15),

axis.title.x = element\_text(vjust = -0.05),

axis.title.y = element\_text(vjust = 1.5),

axis.text = element\_text(size = 13),

legend.text = element\_text(size = 13),

legend.title = element\_text(size = 13),

legend.key = element\_rect(color = "white")) +

# add absolute values

geom\_text(aes(y = value + 6, label = absvals), size = 4.3)

</DIS>

We define the plots for the three other alternations (nnvsof\_plot, nnvss\_plot, threewaygen\_plot) in a similar way. Finally, we visualize all four plots at once by using grid.arrange() from the gridExtra package:

<DIS>

library(gridExtra)

grid.arrange(svsof\_plot,nnvsof\_plot,nnvss\_plot,threewaygen\_plot, ncol = 2)

</DIS>

*<B>Mixed-effects logistic regression*

All our regression analyses were carried out using the lme4 package, version 0.999999-2. We use the c.() function to center continuous variables.2

<DIS>

library(lme4)

c. <- function (x) scale(x, scale = FALSE)

</DIS>

*<C>Optimizing the random effects structure.* We optimize the random effects structure in Model 1 (*s*-genitives versus *of*-genitives) using likelihood ratio tests. More specifically, we use the maximal regression model, which contains all main effects, all interaction effects with time and random intercepts for four random effect candidates (filename, register, por\_head\_noun, and pum\_head\_noun). We successively leave out one of the random effects. Then, we use a likelihood ratio test to see whether the model with the random effect differs significantly from the model without the random intercept. If so, we conclude that the random effect improves the mixed model significantly:

<DIS>

# fit maximal model

fit <- glmer(depvar ~

##############

# main effects and interaction effects

##############

(c.(por\_length\_words) +

c.(pum\_length\_words) +

animacy +

alpha\_persistence\_OF +

alpha\_persistence\_S +

alpha\_persistence\_NN +

beta\_persistence\_OF +

beta\_persistence\_S +

beta\_persistence\_NN +

c.(TTR) +

c.(por\_thematicity\_ptw) +

c.(pum\_thematicity\_ptw) +

final\_sibilancy) \*

time +

##########################

# random effects

# all adjustments to the intercept

##########################

(1|por\_head\_noun) +

(1|pum\_head\_noun) +

(1|register) +

(1|filename),

data = svsof,

family = binomial

)

# fit maximal model without random intercept for por\_head\_noun

fit\_nopor\_head\_noun <- update(fit, .~. - (1|por\_head\_noun))

# likelihood ratio test

anova(fit\_nopor\_head\_noun, fit, test = "Chisq")

</DIS>

*<C>Regression models in binary alternation contexts.* We start from the maximal model, which has the structure just shown. After pruning, we end up with this model for the dataset *svsof*:

<DIS>

# minimal adequate model for svsof

svsof\_model <- glmer(depvar ~

##############

# main effects & interaction effects with time

##############

c.(por\_length\_words) +

c.(pum\_length\_words) +

alpha\_persistence\_S +

beta\_persistence\_S +

(animacy +

c.(pum\_thematicity\_ptw) +

final\_sibilancy) \*

time +

##########################

# random effects

# all adjustments to the intercept

##########################

(1|por\_head\_noun) +

(1|pum\_head\_noun) +

(1|register) +

(1|filename),

data = svsof,

family=binomial

)

# print the estimates and p-values

print(summary(svsof\_model), corr = F)

</DIS>

Using the somers2() function from Hmisc on the fitted values, we obtain the *C* value for the model.

<DIS>

library(Hmisc)

somers2(binomial()$linkinv(fitted(svsof\_model)), as.numeric(svsof$depvar) -1)

</DIS>

The proportion of correctly predicted values is calculated by cross tabulating the observed and predicted values.

<DIS>

fitted <- fitted(svsof\_model)

predicted <- ifelse(fitted >= .5, 1,0)

a <- data.frame(svsof, predicted)

CrossTable(svsof$depvar, a$predicted)

</DIS>

Next, we investigate whether multicollinearity is a problem for the predictors in the model. We calculate the condition number κ with collin.fnc() from the languageR package.3

<DIS>

library(languageR)

collin.fnc(as.data.frame(svsof\_model@X)[,-1])$cnumber

</DIS>

Finally, we use the code from Baayen (2008:283) to check whether the model is valid under bootstrapping. The code is reproduced here:

<DIS>

filevariety = levels(svsof$filename)

nruns = 100 # number of bootstrap runs

for (run in 1:nruns){

# sample with replacement from files

mysampleoffiles = sample(filevariety, replace = TRUE)

# select rows from data frame for the sampled files

mysample = svsof[is.element(svsof$filename, mysampleoffiles),]

# fit a mixed effects model

mysample.glmer <- glmer(depvar ~

##############

# main effects & interaction effects with time

##############

c.(por\_length\_words) +

c.(pum\_length\_words) +

alpha\_persistence\_S +

beta\_persistence\_S +

(animacy +

c.(pum\_thematicity\_ptw) +

final\_sibilancy) \*

time +

##########################

# random effects

# all adjustments to the intercept

##########################

(1|por\_head\_noun) +

(1|pum\_head\_noun) +

(1|register) +

(1|filename),

data = svsof,

family=binomial

)

# extract fixed effects from model

fixedEffects = fixef(mysample.glmer)

# save fixed effects for later inspection

if(run == 1) res = fixedEffects

else res = rbind(res, fixedEffects)

# this takes time, so output dots to indicate progress

cat(".")

}

cat("\n") # add new line to console

# assign sensible rownames

rownames(res) = 1:nruns

# and convert into data frame

res = data.frame(res)

# inspect 95% confidence intervals for all variables simultaneously

t(apply(res, 2, quantile, c(0.025, 0.5, 0.975)))

</DIS>

We can use the ranef() function to inspect the random effects.

<DIS>

# inspect random effects

ranef(svsof\_model)

## pum\_head\_noun

ranef(svsof\_model)$pum\_head\_noun

nms <- rownames(ranef(svsof\_model)$pum\_head\_noun)

intercepts <- ranef(svsof\_model)$pum\_head\_noun[,1]

support <- tapply(svsof$pum\_head\_noun, svsof$pum\_head\_noun,length)

labels <- paste(nms,support)

barplot(intercepts[order(intercepts)],names.arg=labels[order(intercepts)])

## por\_head\_noun

ranef(svsof\_model)$por\_head\_noun

nms <- rownames(ranef(svsof\_model)$por\_head\_noun)

intercepts <- ranef(svsof\_model)$por\_head\_noun[,1]

support <- tapply(svsof$por\_head\_noun, svsof$por\_head\_noun,length)

labels <- paste(nms,support)

barplot(intercepts[order(intercepts)],names.arg=labels[order(intercepts)])

## filename

ranef(svsof\_model)$filename

ranef(svsof\_model)$filename

nms <- rownames(ranef(svsof\_model)$filename)

intercepts <- ranef(svsof\_model)$filename[,1]

support <- tapply(svsof$filename, svsof$filename,length)

labels <- paste(nms,support)

barplot(intercepts[order(intercepts)],names.arg=labels[order(intercepts)])

## register

ranef(svsof\_model)$register

ranef(svsof\_model)$register

nms <- rownames(ranef(svsof\_model)$register)

intercepts <- ranef(svsof\_model)$register[,1]

support <- tapply(svsof$register, svsof$register,length)

labels <- paste(nms,support)

barplot(intercepts[order(intercepts)],names.arg=labels[order(intercepts)])

</DIS>

The mixed models for datasets *nnvsof*, *nnvss,* and *threewaygen* (binary response variable NN-genitive versus not NN-genitive) are as follows:

<DIS>

# minimal adequate model for nnvsof

nnvsof\_model <- glmer(depvar ~

##############

# main effects

##############

c.(por\_length\_words) +

c.(pum\_length\_words) +

beta\_persistence\_NN +

c.(pum\_thematicity\_ptw) +

final\_sibilancy \*

time +

##########################

# random effects

# all adjustments to the intercept

##########################

(1|por\_head\_noun) +

(1|pum\_head\_noun) +

(1|register) +

(1|filename),

data = nnvsof,

family=binomial

)

# minimal adequate model for nnvss

nnvss\_model <- glmer(depvar ~

##############

# main effects

##############

c.(pum\_length\_words) +

animacy +

(c.(pum\_thematicity\_ptw) +

final\_sibilancy) \*

time +

##########################

# random effects

# all adjustments to the intercept

##########################

(1|por\_head\_noun) +

(1|pum\_head\_noun) +

(1|register) +

(1|filename),

data = nnvss,

family=binomial

)

# minimal adequate model for threewaygen

# the dependent variable in this data set is the binary alternation between NN-

# genitive and not NN-genitive

threewaygen\_binary\_model <- glmer(depvar\_binary ~

##############

# main effects & interaction effects with time

##############

c.(por\_length\_words) +

alpha\_persistence\_OF +

beta\_persistence\_NN +

final\_sibilancy +

(c.(pum\_length\_words) +

alpha\_persistence\_S +

c.(por\_thematicity\_ptw) +

c.(pum\_thematicity\_ptw)) \*

time +

##########################

# random effects

# all adjustments to the intercept

##########################

(1|por\_head\_noun) +

(1|pum\_head\_noun) +

(1|register) +

(1|filename),

data = threewaygen,

family=binomial,

)

</DIS>

Further diagnostics can be obtained with the code that was used for Model 1.

*<C>Relative importance of the predictors in the regression models.* To determine the relative importance of the predictors in the models, the chi-squared test statistics, which are the output of likelihood ratio tests, are used. More specifically, we use the Anova() function from the car package, with the models as its argument.

<DIS>

library(car)

Anova(svsof\_model)

s.vals <- Anova(svsof\_model)[["Chisq"]]

names(s.vals) <- rownames(Anova(svsof\_model))

s.vals <- sort(s.vals)

s.vals

</DIS>

<Z>notes

<N> **1.** More recent versions of lme4 are currently available. If a newer version of lme4 is used to replicate the analysis, some minor adjustments to the R code might be necessary. A version of lme4 that is back compatible with lme4 v0.99x code is available (lme4.0). New users will need to install this package to reproduce the code here.

**2**. See http://hlplab.wordpress.com/2011/02/24/diagnosing-collinearity-in-lme4/.

**3.** For more recent versions of lme4 (>version 1), use collin.fnc(getME(svsof\_model, “X”)[, -1])$cnumber.</N>

<Z>references

Baayen, R. Harald. (2008). *Analyzing linguistic data: A practical introduction to statistics using R*. Cambridge: Cambridge University Press.

R Development Core Team. (2013). *R: A language and environment for statistical computing*. Vienna, Austria. Available at: http://www.R-project.org/. Accessed 12/10/2015.