

Supporting Materials for ‘The Integration of Norse-Derived Terms in English’

Contents

1	Introduction	1
1.1	Variables	2
2	Load Libraries	3
3	Load data	3
3.1	Examples	4
4	Descriptive statistics	5
5	Total frequency analysis	10
5.1	Simple distance (total frequency)	10
5.2	Feature-based distance (total frequency)	17
5.3	Historical distance (total frequency)	20
5.4	Exploratory analyses	24
6	Source-level frequency analysis	32
6.1	Simple distance	34
6.2	Feature-based distance	39
6.3	Historical distance	44
7	Comparison between texts	53
8	Bayesian modelling	55
8.1	Simple distance	55
8.2	Feature distance	58
8.3	Historical distance	60

1 Introduction

The analysis covers the following sources:

- FCPC Peterborough, 1154 (12th C)
- Ormulum South Lincolnshire, ca. 1180 (12th C)
- Havelok, composed in Lincolnshire, but manuscript from West Norfolk 13th century (13th C)
- Genesis and Exodus, composed in East Midlands, manuscript from West Norfolk 13th/14th century (14th C)
- Cursor mundi, c. 1300 (14th C)
- Gawain-poet Cheshire / Staffordshire, ca. 1380 (14th C)
- St Erkenwald, written late fourteenth or the early fifteenth century, manuscript from 1477 (15th C)
- Mannyng South Lincolnshire, ca. 1450 (15th C)
- Wars of Alexander northern England, ca. 1450 (15th C)
- Texts from Lincolnshire, Norfolk and Nottinghamshire from the Corpus of Middle English (1399-1525), (15th C)
- Texts by Richard Rolle (before 1465), (15thC)

The data analysed here is the product of several processing steps. The file `data/SharedIntegrationOfCognatesData.xls` has the original transcribed data. This is cleaned and processed by the script `analysis/analyseTextDistances.py`, which draws some code from `analysis/CLTSFeatureBasedAlignment.py` which is mainly contributed by Johann Mattis List (see <https://calc.hypotheses.org/1962> and <https://gist.github.com/LinguList/7fac44813572f65259c872ef89fa64ad>). The script calculates the distances between pairs of Norse and English forms according to three measures:

- Simple distance from Keller (2020).
- A feature-based distance.
- A historical distance that uses the likelihood of one segment historically replacing another.

Each row in the data represents a comparison between a Norse form and an English form within a given cognate set, including the three measures of distance and frequency of occurrence in each source.

1.1 Variables

The variables in the data (and some that are calculated in the script below) are described as follows:

“Set”: the cognate set the pair of comparisons belong to.

“Class”: the word class the set can appear as (and some binary variables representing the same information).

“NorseLexeme”, “EngLexeme”, “NorseForm”, “EngForm”: The lexemes and full forms for Norse and English terms.

“NorseFormDiagnostic”, “EngFormDiagnostic”: the relevant parts of the form that are diagnostic of the etymology.

“Alignment”, “FeatureAlignment”, “HistoricalAlignment”: multiple sequence alignment of the forms according to the three measures.

“RawDistance”, “NormDistance”, “RawFeatureDistance”, “NormFeatureDistance”, “RawHistoricalDistance”, “NormHistoricalDistance”: Raw and normed distances between

“NFreq...”, “EFreq...”: Frequencies of the Norse and English forms in each source.

“Alliteration”: only true if the source uses alliterative verse and the Norse form and the English form do *not* start with segments in the same alliterative category. That is, it is true if part of the decision about which form to use might be influenced by the need to alliterate. The alliterative texts include: Gawain, St Erkenwold, Wars of Alexander.

“totalNFreq”: The total Norse frequency across all sources.

“totalEFreq”: The total English frequency across all sources.

“NorseProp”: The proportion of total Norse frequency compared to total Norse Frequency + total English Frequency.

“NorseDiagnosticScore”: Whether or not the forms differ according to a specific segments that are characteristically diagnostic of Norse etymology.

2 Load Libraries

```
library(openxlsx)
library(sjPlot)
library(lme4)
library(mgcv)
library(party)
library(ggplot2)
library(phangorn)
library(gridExtra)
library(GGally)
library(MuMIn)
library(brms)
library(ggeffects)
library(gamm4)
```

3 Load data

Load data created by the python program and convert variables to their proper type:

```
d = read.xlsx("../data/IntegrationDistances.xlsx",1)
# Ignore numerals in set name
d$Set = gsub("[0-9]", "", d$Set)
d$Set = as.factor(d$Set)
d$EngLexeme = factor(d$EngLexeme)
d$ELen = nchar(d$EngForm)
d$NLen = nchar(d$NorseForm)

norseFrequencyColumns = c("NFreqFCPC", "NFreqGawainPoet",
                          "NFreqGenAndEx", "NFreqHavelok",
                          "NFreqMannyng", "NFreqOrmulum",
                          'NFreqWarsAlexander',
                          "NFreqStErkenwald", "NFreqCursorMundi",
                          "NFreqLinc", "NFreqNott", "NFreqNorf", "NFreqRolle")

englishFrequencyColumns = c("EFreqFCPC", "EFreqGawainPoet",
                            "EFreqGenAndEx", "EFreqHavelok",
                            "EFreqMannyng", "EFreqOrmulum",
                            "EFreqWarsAlexander",
                            "EFreqStErkenwald", "EFreqCursorMundi",
                            "EFreqLinc", "EFreqNott", "EFreqNorf", "EFreqRolle")
```

Convert frequencies to numeric type:

```
for(col in c(norseFrequencyColumns, englishFrequencyColumns)){
  d[,col] = as.numeric(d[,col])
}
```

Calculate the total frequency across all sources and the proportion of Norse forms compared to all forms:

```
d$totalNFreq = rowSums(d[,norseFrequencyColumns], na.rm = T)
d$totalEFreq = rowSums(d[,englishFrequencyColumns], na.rm = T)
d$NorseProp = d$totalNFreq / rowSums(d[,c("totalNFreq", "totalEFreq")], na.rm = T)

d = d[!is.na(d$NorseProp),]
```

Scale the distances to lie between 0 and 1:

```
d$NormDistance = as.numeric(d$NormDistance)
d$NormFeatureDistance = as.numeric(d$NormFeatureDistance)
```

```

d$NormHistoricalDistance = as.numeric(d$NormHistoricalDistance)

# Scale distances
normX = function(X){
  (X-min(X))/(max(X)-min(X))
}
d$NormDistance = normX(d$NormDistance)
d$NormFeatureDistance = normX(d$NormFeatureDistance)
d$NormHistoricalDistance = normX(d$NormHistoricalDistance)

d$NormDistance.rank =
  rank(d$NormDistance,ties.method = "max")
d$NormFeatureDistance.rank =
  rank(d$NormFeatureDistance,ties.method = "max")
d$NormHistoricalDistance.rank =
  rank(d$NormHistoricalDistance,ties.method = "max")

```

3.1 Examples

4 Descriptive statistics

Number of English forms: 135

Number of Norse forms: 126

Number of comparisons: 1636

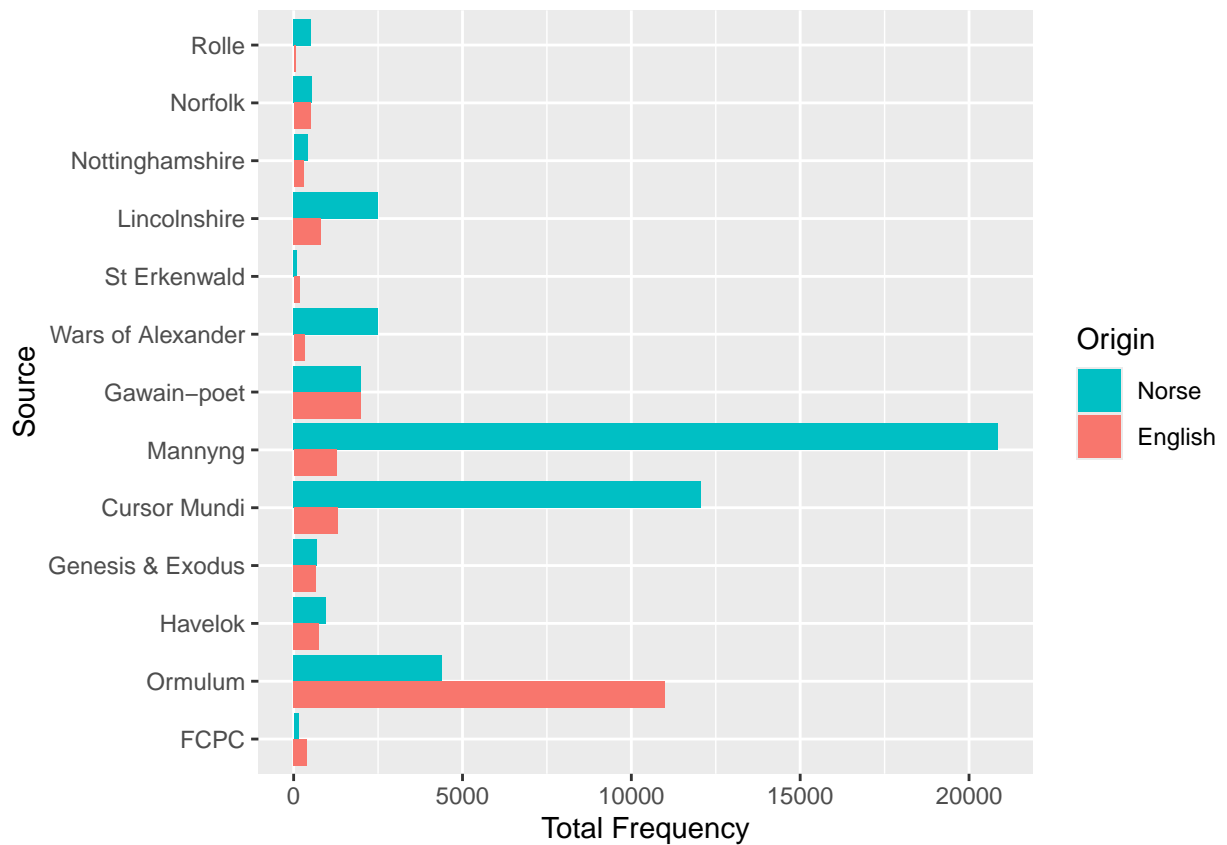
Number of sets: 67

Proportion of Norse and English terms (relative total frequency of each type):

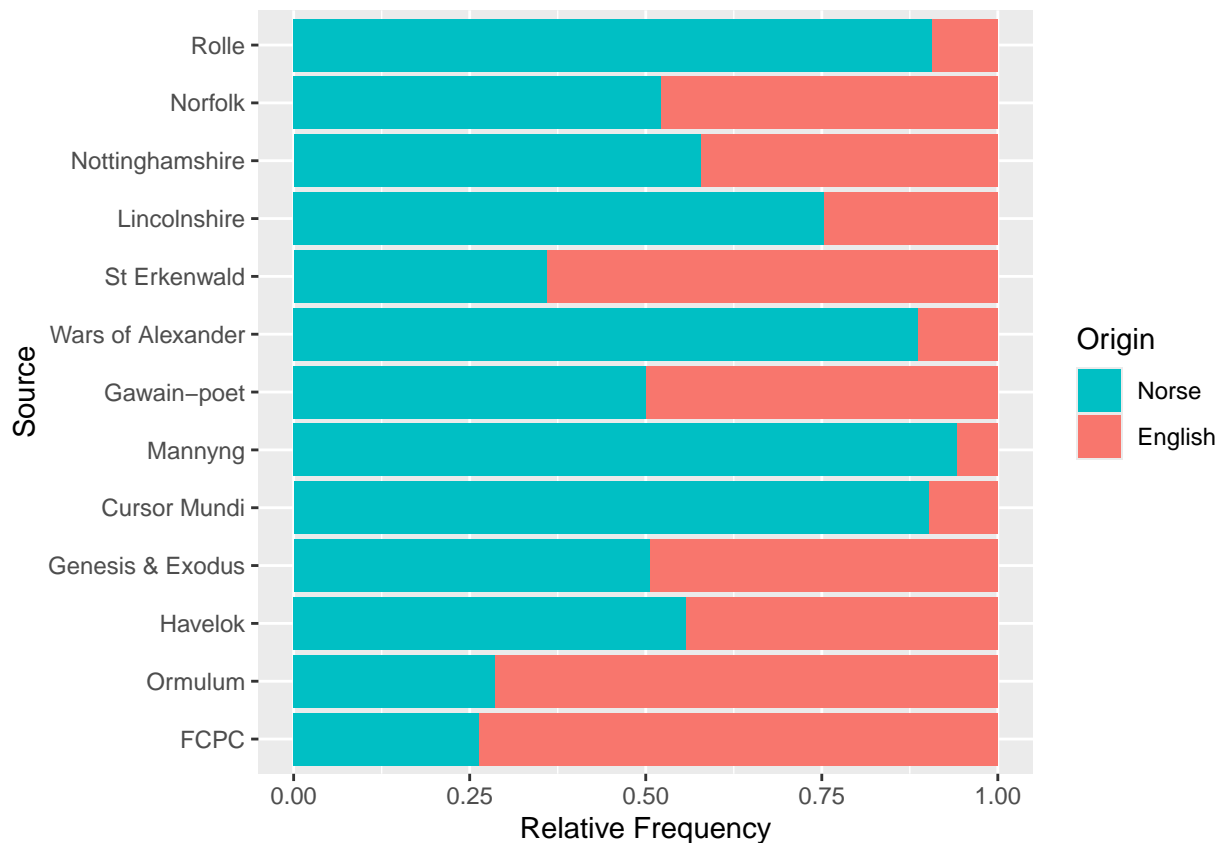
```
dx = rbind(
  data.frame(
    Lang = "Norse",
    Freq = colSums(d[,norseFrequencyColumns],na.rm = T),
    Source = c("FCPC","Gawain-poet","Genesis & Exodus","Havelok",
               "Mannyng", "Ormulum","Wars of Alexander",
               "St Erkenwald", "Cursor Mundi",
               "Lincolnshire","Nottinghamshire",
               "Norfolk", "Rolle")),
  data.frame(
    Lang = "English",
    Freq = colSums(d[,englishFrequencyColumns],na.rm = T),
    Source = c("FCPC","Gawain-poet","Genesis & Exodus","Havelok",
               "Mannyng", "Ormulum","Wars of Alexander",
               "St Erkenwald", "Cursor Mundi",
               "Lincolnshire","Nottinghamshire",
               "Norfolk", "Rolle")))

dx$Source = factor(dx$Source,
  levels=c("FCPC","Ormulum","Havelok",
           "Genesis & Exodus",
           "Cursor Mundi","Mannyng",
           "Gawain-poet", "Wars of Alexander",
           "St Erkenwald",
           "Lincolnshire","Nottinghamshire",
           "Norfolk", "Rolle"))

gRawFreq = ggplot(dx,aes(x=Source,y=Freq,fill=Lang)) +
  geom_bar(stat="identity",position = 'dodge') +
  ylab("Total Frequency") +
  scale_fill_discrete(breaks=c('Norse',"English"),
                      name = "Origin") +
  coord_flip()
gRawFreq
```



```
gPropFreq = ggplot(dx,aes(x=Source,y=Freq,fill=Lang)) +
  geom_bar(stat="identity",position = 'fill') +
  ylab("Relative Frequency") +
  scale_fill_discrete(breaks=c('Norse',"English"),
                      name = "Origin")+
  coord_flip()
gPropFreq
```



```
pdf("../results/GRawFreq.pdf",width=5,height=3.5)
gRawFreq
dev.off()
```

```
## pdf
## 2
```

```
pdf("../results/GPropFreq.pdf",width=5,height=3.5)
gPropFreq
dev.off()
```

```
## pdf
## 2
```

```
pdf("../results/GRawAndPropFreq.pdf",width=6,height=3)
grid.arrange(gRawFreq+
  theme(legend.position = "none"),
  gPropFreq +
  theme(axis.text.y = element_blank(),
        axis.title.y = element_blank()),
  ncol=2)
dev.off()
```

```
## pdf
## 2
```

Correlations between distance measures:

```
cor.test(d$NormDistance,d$NormFeatureDistance)
```

```
##
## Pearson's product-moment correlation
##
## data: d$NormDistance and d$NormFeatureDistance
## t = 95.432, df = 1634, p-value < 2.2e-16
```

```
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
## 0.9130857 0.9278599
## sample estimates:
##      cor
## 0.9208025
```

```
cor.test(d$NormDistance,d$NormHistoricalDistance)
```

```
##
## Pearson's product-moment correlation
##
## data: d$NormDistance and d$NormHistoricalDistance
## t = 76.001, df = 1634, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
## 0.8717247 0.8931370
## sample estimates:
##      cor
## 0.882889
```

```
cor.test(d$NormFeatureDistance,d$NormHistoricalDistance)
```

```
##
## Pearson's product-moment correlation
##
## data: d$NormFeatureDistance and d$NormHistoricalDistance
## t = 76.008, df = 1634, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
## 0.8717445 0.8931537
## sample estimates:
##      cor
## 0.8829071
```

Function for extracting stats:

```
getStatReport = function(m0,m1,m2,m3,finalModel,outFile){
  modelComparison = as.data.frame(anova(m0,m1,m2,m3))
  modelComparison$pChi = round(modelComparison$`Pr(>Chisq)`,3)
  modelComparison$pChi[modelComparison$pChi==0] = "< 0.001"
  modelComparison$lldiff = NA
  modelComparison$lldiff[2:4] = diff(modelComparison$logLik)
  modelComparison = modelComparison[2:4,]
  modelComparison$Term = c("Linear","Quadratic","Cubic")

  coef = as.data.frame(summary(finalModel)$coefficients)
  rx = which(grepl("Norm",rownames(coef)))
  coef[rx,"Estimate"] = round(coef[rx,"Estimate"],3)
  coef[rx,"z value"] = round(coef[rx,"z value"],3)
  coef[, "Pr(>|z|)"] = round(coef[, "Pr(>|z|)"],3)
  coef[, "Pr(>|z|)"][coef[, "Pr(>|z|)"] ==0] = "< 0.001"

  modelComparison$Estimate = coef[rx,"Estimate"]
  modelComparison$z = coef[rx,"z value"]
  modelComparison$p = coef[rx,"Pr(>|z|)"]

  modelComparison = modelComparison[,
    c("Term","BIC","lldiff",
      "Chisq","Df","pChi",
      "Estimate","z","p")]
}
```



```

mcx = paste(paste0(c("Linear: ", "Quadratic: ", "Cubic: "),
  "beta = ", coef[rx, "Estimate"],
  ", z = ", coef[rx, "z value"],
  ", Wald p = ", coef[rx, "Pr(>|z|)"],
  ", LLDiff = ", round(modelComparison$lldiff, 1),
  ", df = ", modelComparison$Df,
  ", p = ", modelComparison$pChi), collapse="; ")
mcx = gsub("= <", "<", mcx)
cat(mcx, file=outFile)

# mx = rbind(c("", "Model Comparison", "", "", "", "", "Model Estimate", "", ""),
#           names(modelComparison),
#           modelComparison)

modelComparison$BIC = round(modelComparison$BIC, 1)
modelComparison$lldiff = round(modelComparison$lldiff, 1)
modelComparison$Chisq = round(modelComparison$Chisq, 1)
modelComparison$Estimate = round(modelComparison$Estimate, 2)

write.csv(modelComparison, file=gsub("\\.txt", ".csv", outFile))

return(mcx)
}

```

5 Total frequency analysis

The analyses below predict the total frequency across all sources, using the orthographic form as the basis for observations.

5.1 Simple distance (total frequency)

We use nested model comparison to evaluate whether higher-order variables should be added (i.e. how non-linear the relationships is).

```
d$NormDistanceChosen = d$NormDistance
# Baseline model
m0 = glmer(cbind(totalNFreq,totalEFreq) ~
           1 + (1|Set) + (1|EngLexeme),
           data = d, family = "binomial")
# Add the distance measure
m1 = update(m0,~.+NormDistanceChosen)
# Add quadratic term
m2 = update(m1,~.+I(NormDistanceChosen^2))
# Add cubic term
m3 = update(m2,~.+I(NormDistanceChosen^3))
# Compare fit of models
anova(m0,m1,m2,m3)

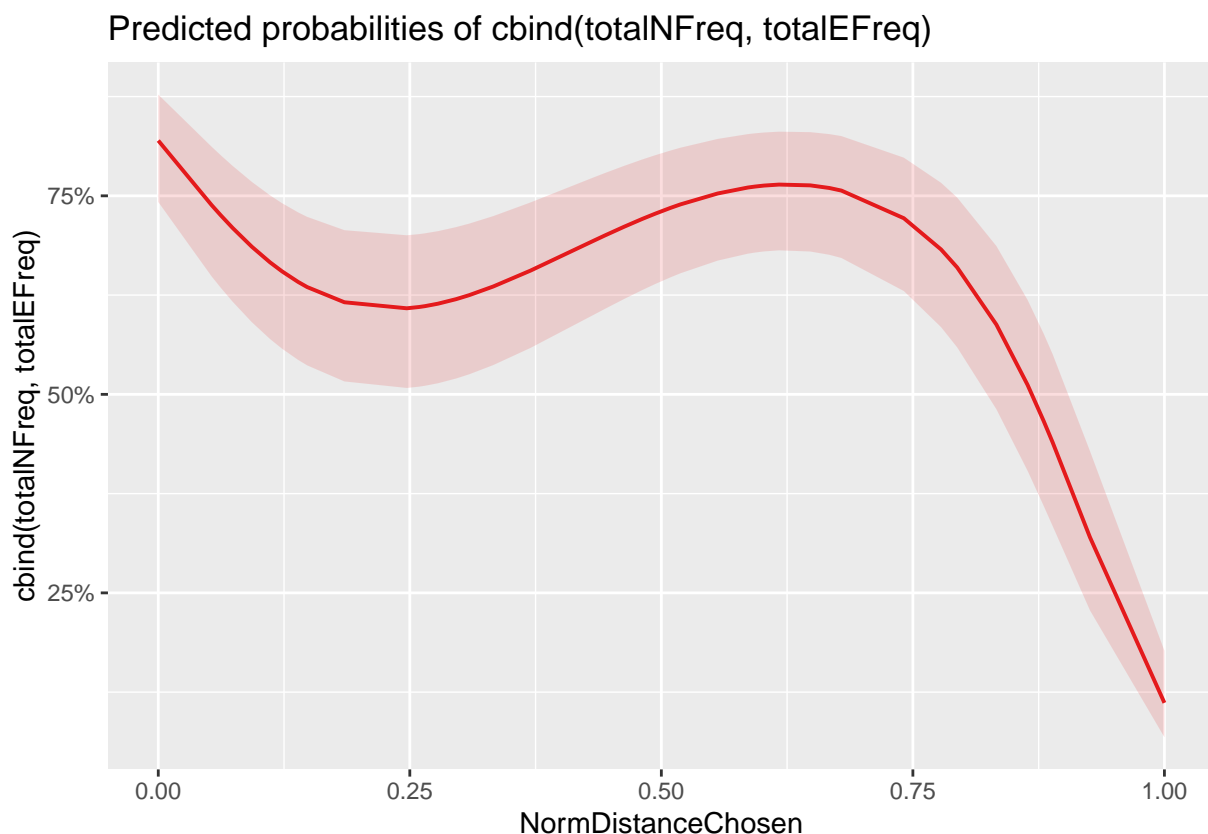
## Data: d
## Models:
## m0: cbind(totalNFreq, totalEFreq) ~ 1 + (1 | Set) + (1 | EngLexeme)
## m1: cbind(totalNFreq, totalEFreq) ~ (1 | Set) + (1 | EngLexeme) + NormDistanceChosen
## m2: cbind(totalNFreq, totalEFreq) ~ (1 | Set) + (1 | EngLexeme) + NormDistanceChosen + I(NormDist
## m3: cbind(totalNFreq, totalEFreq) ~ (1 | Set) + (1 | EngLexeme) + NormDistanceChosen + I(NormDist
##      npar   AIC    BIC logLik deviance   Chisq Df Pr(>Chisq)
## m0      3 32132 32149 -16063    32126
## m1      4 32134 32156 -16063    32126  0.0208  1    0.8854
## m2      5 32078 32105 -16034    32068 58.4612  1 2.073e-14 ***
## m3      6 31842 31875 -15915    31830 237.7808  1 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Summary of final model:
summary(m3)

## Generalized linear mixed model fit by maximum likelihood (Laplace
## Approximation) [glmerMod]
## Family: binomial ( logit )
## Formula: cbind(totalNFreq, totalEFreq) ~ (1 | Set) + (1 | EngLexeme) +
##          NormDistanceChosen + I(NormDistanceChosen^2) + I(NormDistanceChosen^3)
## Data: d
##
##           AIC          BIC    logLik deviance df.resid
## 31842.1    31874.5 -15915.1   31830.1      1630
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -13.5685  -2.5875  -0.6287   0.7102  24.4508
##
## Random effects:
##  Groups      Name                Variance Std.Dev.
## EngLexeme (Intercept) 1.998      1.413
## Set          (Intercept) 1.458      1.208
```

```
## Number of obs: 1636, groups: EngLexeme, 135; Set, 67
##
## Fixed effects:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)      1.5135    0.2325   6.509 7.57e-11 ***
## NormDistanceChosen -10.5625    0.9471 -11.153 < 2e-16 ***
## I(NormDistanceChosen^2) 31.1462    2.2254  13.996 < 2e-16 ***
## I(NormDistanceChosen^3) -24.1724    1.5321 -15.778 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##              (Intr) NrmDsC I(NDC^2
## NrmDstncChs  -0.435
## I(NrmDsC^2)   0.388 -0.966
## I(NrmDsC^3)  -0.346  0.906 -0.982
```

```
pSimpleTotal = plot_model(m3, 'eff',
  terms="NormDistanceChosen[all]")
pSimpleTotal
```



Marginal represents the variance explained by the fixed effects. Conditional represents the variance explained by the entire model.

```
r.squaredGLMM(m3)
```

```
## Warning: 'r.squaredGLMM' now calculates a revised statistic. See the help page.
## Warning: the null model is correct only if all variables used by the original
## model remain unchanged.
##               R2m      R2c
## theoretical 0.04500701 0.9783670
## delta      0.04474303 0.9726285
```

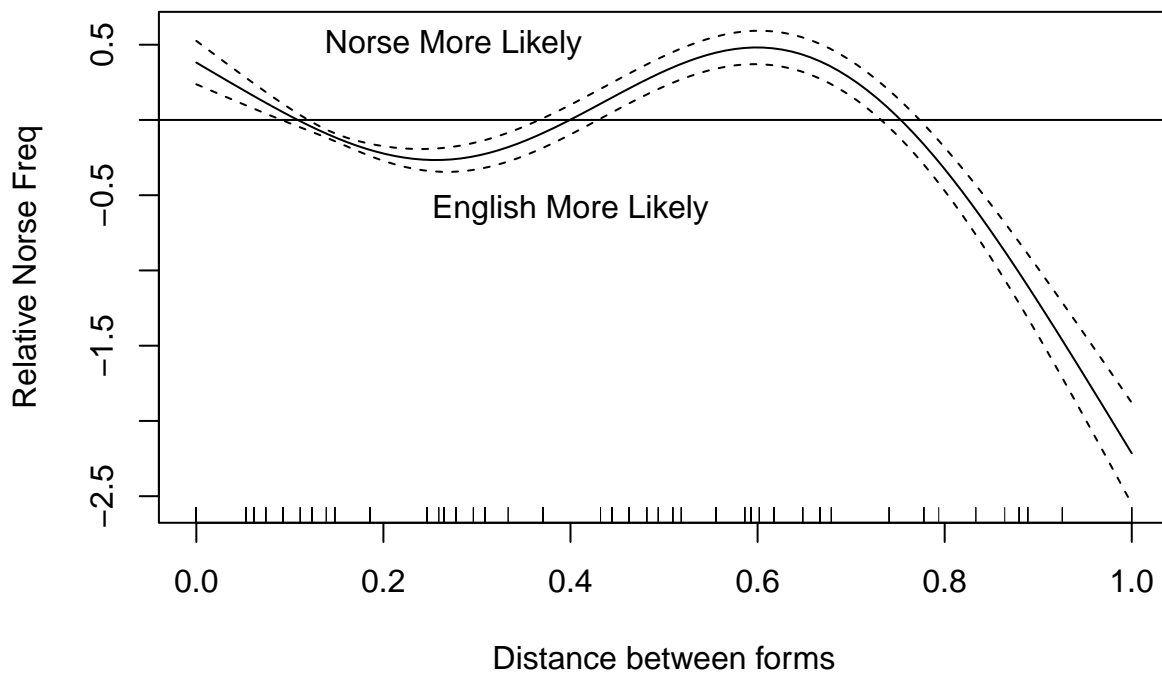
Statistics: (Linear: beta = -10.563, z = -11.153, Wald p < 0.001, LLDiff = 0, df = 1, p = 0.885; Quadratic: beta = 31.146, z = 13.996, Wald p < 0.001, LLDiff = 29.2, df = 1, p < 0.001; Cubic: beta = -24.172, z = -15.778, Wald p < 0.001, LLDiff = 118.9, df = 1, p < 0.001)

Same analysis using GAM with binomial distribution:

```
mGAM = gam(cbind(totalNFreq,totaleFreq) ~
            s(NormDistanceChosen,k=4) +
            s(Set,bs="re") +
            s(EngLexeme,bs="re"),
            data = d, family = "binomial")
summary(mGAM)

##
## Family: binomial
## Link function: logit
##
## Formula:
## cbind(totalNFreq, totaleFreq) ~ s(NormDistanceChosen, k = 4) +
##      s(Set, bs = "re") + s(EngLexeme, bs = "re")
##
## Parametric coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.6717      0.2287   2.937  0.00331 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##              edf Ref.df   Chi.sq p-value
## s(NormDistanceChosen)  2.997     3    253.3 <2e-16 ***
## s(Set)                 44.342    66 1327777.6  0.369
## s(EngLexeme)           81.735   134 354268.6  0.521
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.44   Deviance explained = 46.1%
## UBRE = 16.052   Scale est. = 1          n = 1636

plot.gam(mGAM,
          ylab="Relative Norse Freq",
          xlab="Distance between forms",
          select = 1)
abline(h=0)
text(0.275,0.5,"Norse More Likely")
text(0.4,-0.6,"English More Likely")
```



The raw frequency of Norse forms is negatively correlated with distance:

```
mNorseFreq = glmer(totalNFreq ~ NormDistance +
  (1|Set) + (1|NorseLexeme),
  data = d, family = "poisson")
```

The distribution of Norse proportions is bimodal, with most terms having either low or high proportions. This isn't ideal for the binomial models above, so we also test the same model using a beta binomial distribution, which can fit bimodal binomial distributions. The result is very similar to the others:

```
d$totalAllFreq = d$totalNFreq + d$totalEFreq
mBinomBeta = brm(totalNFreq | trials(totalAllFreq) ~
  1 + NormDistanceChosen +
  I(NormDistanceChosen^2) +
  I(NormDistanceChosen^3) +
  (1|Set) + (1|EngLexeme),
  data = d, family = "beta_binomial")
```

```
## Compiling Stan program...
```

```
## Start sampling
```

```
##
```

```
## SAMPLING FOR MODEL '7404dd55f235b75d82269cea3d068515' NOW (CHAIN 1).
```

```
## Chain 1:
```

```
## Chain 1: Gradient evaluation took 0.000564 seconds
```

```
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 5.64 seconds.
```

```
## Chain 1: Adjust your expectations accordingly!
```

```
## Chain 1:
```

```
## Chain 1:
```

```
## Chain 1: Iteration: 1 / 2000 [ 0%] (Warmup)
```

```
## Chain 1: Iteration: 200 / 2000 [ 10%] (Warmup)
```

```
## Chain 1: Iteration: 400 / 2000 [ 20%] (Warmup)
```

```
## Chain 1: Iteration: 600 / 2000 [ 30%] (Warmup)
```

```
## Chain 1: Iteration: 800 / 2000 [ 40%] (Warmup)
```

```
## Chain 1: Iteration: 1000 / 2000 [ 50%] (Warmup)
```

```
## Chain 1: Iteration: 1001 / 2000 [ 50%] (Sampling)
```

```
## Chain 1: Iteration: 1200 / 2000 [ 60%] (Sampling)
```

```
## Chain 1: Iteration: 1400 / 2000 [ 70%] (Sampling)
```

```

## Chain 1: Iteration: 1600 / 2000 [ 80%] (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%] (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%] (Sampling)
## Chain 1:
## Chain 1: Elapsed Time: 70.7199 seconds (Warm-up)
## Chain 1: 64.0899 seconds (Sampling)
## Chain 1: 134.81 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL '7404dd55f235b75d82269cea3d068515' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.000462 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 4.62 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration: 1 / 2000 [ 0%] (Warmup)
## Chain 2: Iteration: 200 / 2000 [ 10%] (Warmup)
## Chain 2: Iteration: 400 / 2000 [ 20%] (Warmup)
## Chain 2: Iteration: 600 / 2000 [ 30%] (Warmup)
## Chain 2: Iteration: 800 / 2000 [ 40%] (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%] (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%] (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%] (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%] (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%] (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%] (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%] (Sampling)
## Chain 2:
## Chain 2: Elapsed Time: 73.5453 seconds (Warm-up)
## Chain 2: 68.7505 seconds (Sampling)
## Chain 2: 142.296 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL '7404dd55f235b75d82269cea3d068515' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.000463 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 4.63 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration: 1 / 2000 [ 0%] (Warmup)
## Chain 3: Iteration: 200 / 2000 [ 10%] (Warmup)
## Chain 3: Iteration: 400 / 2000 [ 20%] (Warmup)
## Chain 3: Iteration: 600 / 2000 [ 30%] (Warmup)
## Chain 3: Iteration: 800 / 2000 [ 40%] (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%] (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%] (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%] (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%] (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%] (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%] (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%] (Sampling)
## Chain 3:
## Chain 3: Elapsed Time: 66.5127 seconds (Warm-up)
## Chain 3: 62.0208 seconds (Sampling)
## Chain 3: 128.533 seconds (Total)
## Chain 3:

```

```

##
## SAMPLING FOR MODEL '7404dd55f235b75d82269cea3d068515' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.000457 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 4.57 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%] (Warmup)
## Chain 4: Iteration:   200 / 2000 [ 10%] (Warmup)
## Chain 4: Iteration:   400 / 2000 [ 20%] (Warmup)
## Chain 4: Iteration:   600 / 2000 [ 30%] (Warmup)
## Chain 4: Iteration:   800 / 2000 [ 40%] (Warmup)
## Chain 4: Iteration:  1000 / 2000 [ 50%] (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%] (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%] (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%] (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%] (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%] (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%] (Sampling)
## Chain 4:
## Chain 4: Elapsed Time: 73.2741 seconds (Warm-up)
## Chain 4:           57.0322 seconds (Sampling)
## Chain 4:           130.306 seconds (Total)
## Chain 4:

## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess

## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail qu
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess

summary(mBinomBeta)

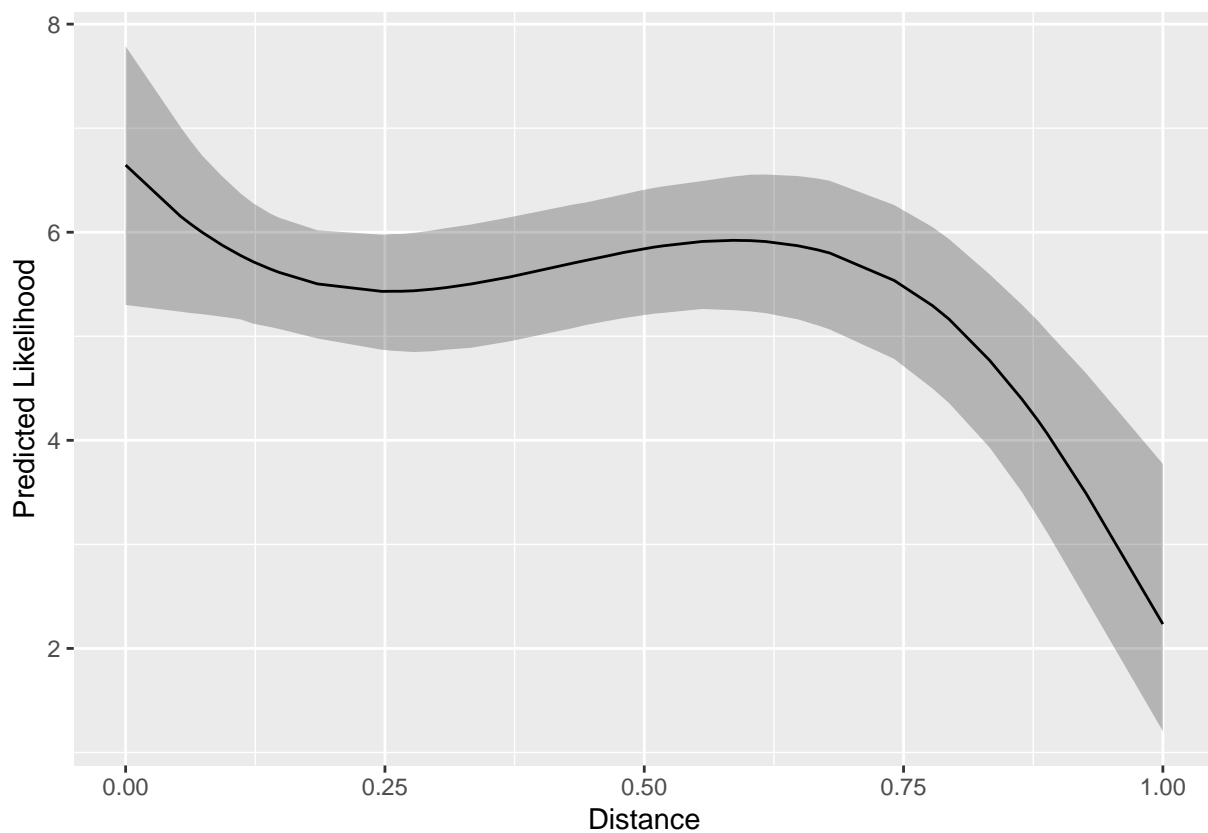
## Family: beta_binomial
## Links: mu = logit; phi = identity
## Formula: totalNFreq | trials(totalAllFreq) ~ 1 + NormDistanceChosen + I(NormDistanceChosen^2) + I
## Data: d (Number of observations: 1638)
## Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
## total post-warmup draws = 4000
##
## Group-Level Effects:
## ~EngLexeme (Number of levels: 135)
##      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)    0.69    0.09    0.53    0.87 1.01    410    968
##
## ~Set (Number of levels: 67)
##      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)    0.33    0.15    0.02    0.60 1.02    177    294
##
## Population-Level Effects:
##      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## Intercept            0.69    0.29    0.12    1.26 1.01    845
## NormDistanceChosen   -4.72    2.45   -9.58   -0.00 1.01    855
## INormDistanceChosenE2 13.32    5.61    2.49   24.34 1.00    900
## INormDistanceChosenE3 -10.53    3.76   -17.88  -3.21 1.00    978
## Tail_ESS
## Intercept            1597

```

```
## NormDistanceChosen      1410
## INormDistanceChosenE2   1399
## INormDistanceChosenE3   1595
##
## Family Specific Parameters:
##   Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## phi      2.00      0.08    1.85    2.16 1.00    3460    2591
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
pred = ggpredict(mBinomBeta, terms="NormDistanceChosen[all]")

## Note: uncertainty of error terms are not taken into account. Consider
##   setting `interval` to "prediction". This will call `posterior_predict()`
##   instead of `posterior_epred()`.

## Warning in checkMatrixPackageVersion(): Package version inconsistency detected.
## TMB was built with Matrix version 1.6.1
## Current Matrix version is 1.6.5
## Please re-install 'TMB' from source using install.packages('TMB', type = 'source') or ask CRAN for
ggplot(pred, aes(x=x, y=predicted, ymin=conf.low, ymax=conf.high)) +
  geom_ribbon(alpha=0.3) +
  geom_line() +
  xlab("Distance") +
  ylab("Predicted Likelihood")
```



5.2 Feature-based distance (total frequency)

We use nested model comparison to evaluate whether higher-order variables should be added (i.e. how non-linear the relationships is).

```
d$NormDistanceChosen = d$NormFeatureDistance
# Baseline model
m0 = glmer(cbind(totalNFreq, totalEFreq) ~
            1 + (1|Set) + (1|EngLexeme),
            data = d, family = "binomial")
# Add the distance measure
m1 = update(m0, ~.+NormDistanceChosen)
# Add quadratic term
m2 = update(m1, ~.+I(NormDistanceChosen^2))
# Add cubic term
m3 = update(m2, ~.+I(NormDistanceChosen^3))
# Compare fit of models
anova(m0, m1, m2, m3)

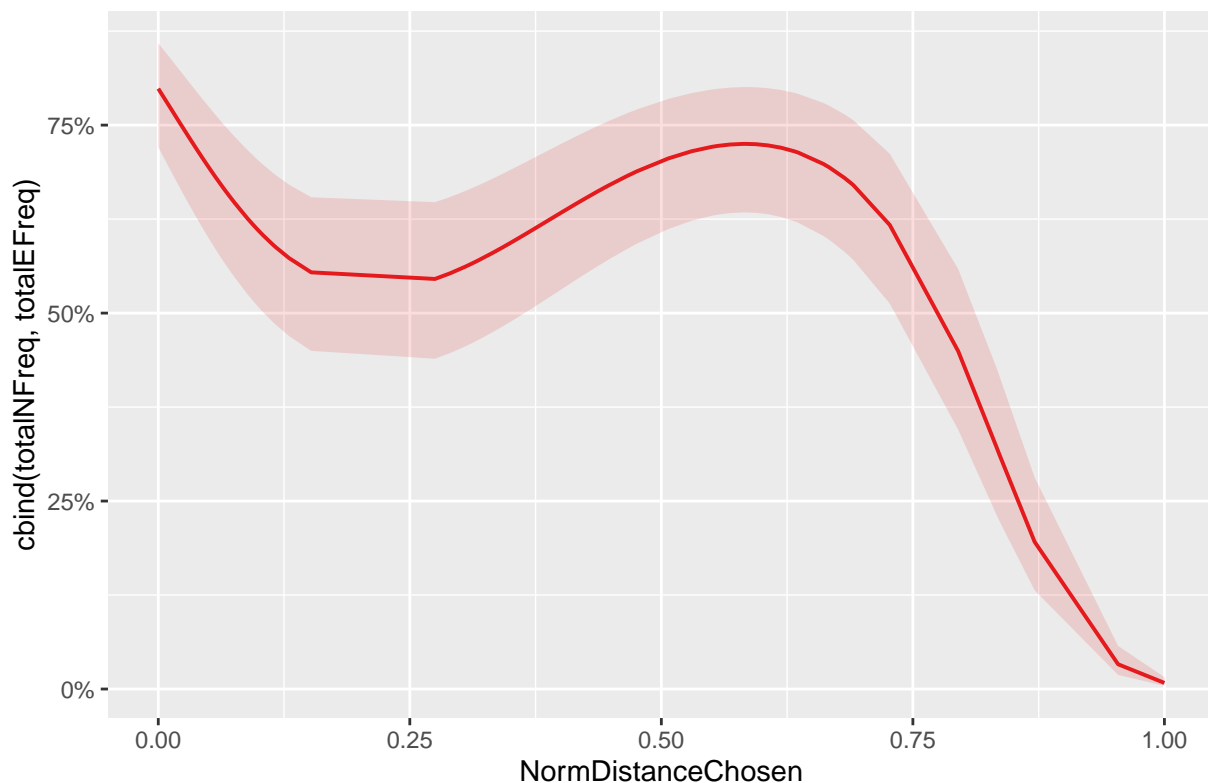
## Data: d
## Models:
## m0: cbind(totalNFreq, totalEFreq) ~ 1 + (1 | Set) + (1 | EngLexeme)
## m1: cbind(totalNFreq, totalEFreq) ~ (1 | Set) + (1 | EngLexeme) + NormDistanceChosen
## m2: cbind(totalNFreq, totalEFreq) ~ (1 | Set) + (1 | EngLexeme) + NormDistanceChosen + I(NormDist
## m3: cbind(totalNFreq, totalEFreq) ~ (1 | Set) + (1 | EngLexeme) + NormDistanceChosen + I(NormDist
##      npar   AIC    BIC logLik deviance   Chisq Df Pr(>Chisq)
## m0      3 32262 32278 -16128    32256
## m1      4 32247 32269 -16120    32239  16.464  1 4.958e-05 ***
## m2      5 32159 32186 -16074    32149  90.456  1 < 2.2e-16 ***
## m3      6 31856 31888 -15922    31844 305.264  1 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Summary of final model:
summary(m3)

## Generalized linear mixed model fit by maximum likelihood (Laplace
## Approximation) [glmerMod]
## Family: binomial ( logit )
## Formula: cbind(totalNFreq, totalEFreq) ~ (1 | Set) + (1 | EngLexeme) +
##          NormDistanceChosen + I(NormDistanceChosen^2) + I(NormDistanceChosen^3)
## Data: d
##
##          AIC          BIC    logLik deviance df.resid
## 31855.8 31888.2 -15921.9 31843.8    1632
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -13.5687  -2.5875  -0.6362   0.7073  24.4503
##
## Random effects:
## Groups   Name                Variance Std.Dev.
## EngLexeme (Intercept) 2.002      1.415
## Set      (Intercept) 1.551      1.245
## Number of obs: 1638, groups: EngLexeme, 135; Set, 67
##
## Fixed effects:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)      1.3771    0.2192  6.282 3.33e-10 ***
## NormDistanceChosen -13.1251    0.9665 -13.580 < 2e-16 ***
```

```
## I(NormDistanceChosen^2) 41.4022      2.6115  15.854 < 2e-16 ***
## I(NormDistanceChosen^3) -34.4563      1.9360 -17.797 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##              (Intr) NrmDsC I(NDC^2
## NrmDstncChs -0.264
## I(NrmDsC^2)  0.242 -0.976
## I(NrmDsC^3) -0.222  0.925 -0.983
pFeatureTotal = plot_model(m3,'eff',
  terms="NormDistanceChosen[all]")
pFeatureTotal
```

Predicted probabilities of cbind(totalNFreq, totalEFreq)



Statistics: (Linear: beta = -13.125, z = -13.58, Wald p < 0.001, LLDiff = 8.2, df = 1, p < 0.001; Quadratic: beta = 41.402, z = 15.854, Wald p < 0.001, LLDiff = 45.2, df = 1, p < 0.001; Cubic: beta = -34.456, z = -17.797, Wald p < 0.001, LLDiff = 152.6, df = 1, p < 0.001)

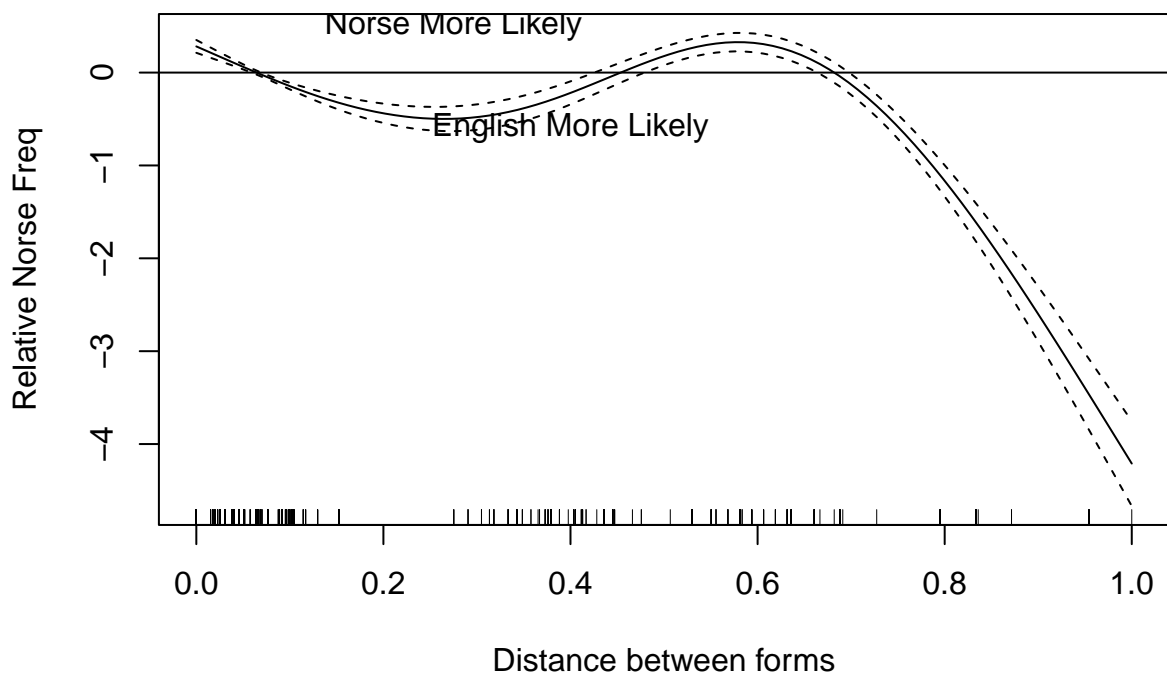
Same analysis using GAM with binomial distribution:

```
mGAM = gam(cbind(totalNFreq,totalEFreq) ~
  s(NormDistanceChosen,k=4) +
  s(Set,bs="re") +
  s(EngLexeme,bs="re"),
  data = d, family = "binomial")
summary(mGAM)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## cbind(totalNFreq, totalEFreq) ~ s(NormDistanceChosen, k = 4) +
```

```
##      s(Set, bs = "re") + s(EngLexeme, bs = "re")
##
## Parametric coefficients:
##           Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.6916      0.2334   2.964  0.00304 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##           edf Ref.df   Chi.sq p-value
## s(NormDistanceChosen)  2.999     3    348.8 <2e-16 ***
## s(Set)                 45.288    66 1224049.3  0.399
## s(EngLexeme)           80.832   134 344925.2  0.644
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.44   Deviance explained = 46.1%
## UBRE = 16.045   Scale est. = 1         n = 1638
```

```
plot.gam(mGAM,
  ylab="Relative Norse Freq",
  xlab="Distance between forms",
  select = 1)
abline(h=0)
text(0.275,0.5,"Norse More Likely")
text(0.4,-0.6,"English More Likely")
```



5.3 Historical distance (total frequency)

We use nested model comparison to evaluate whether higher-order variables should be added (i.e. how non-linear the relationships is).

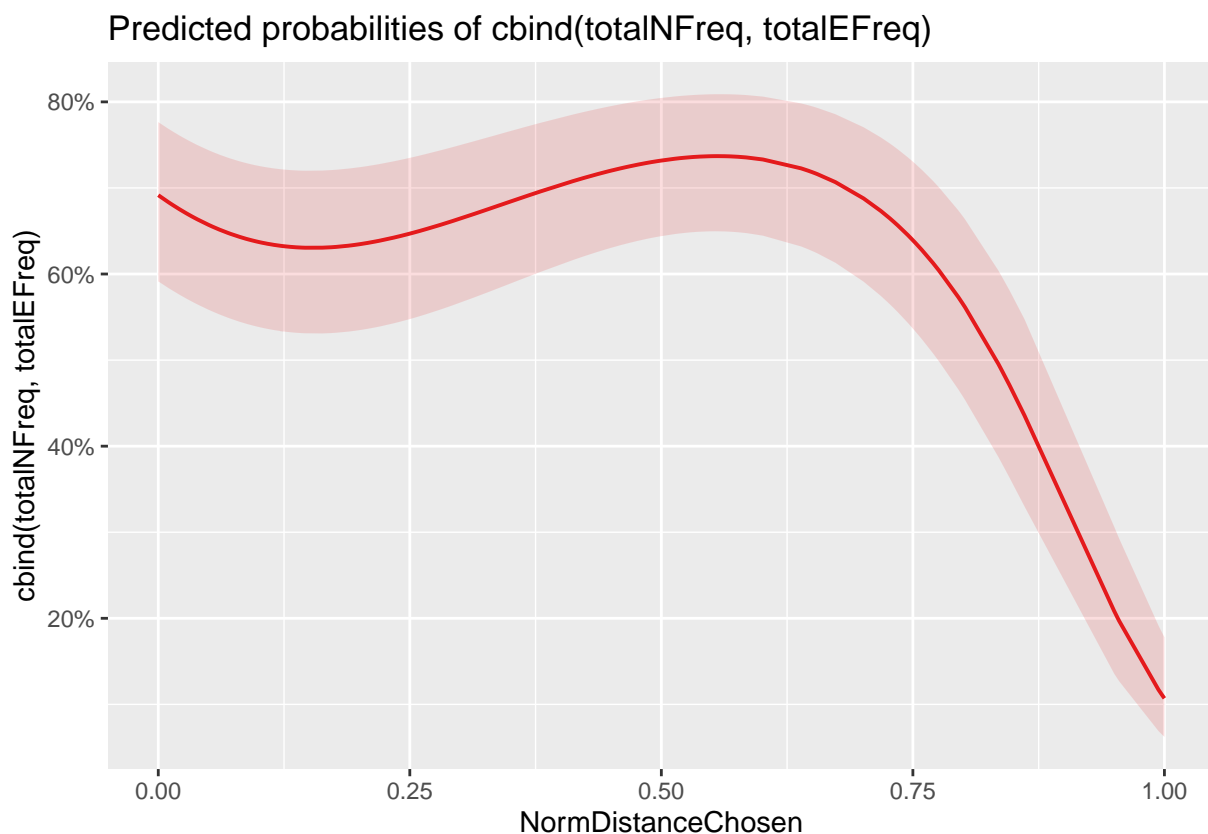
```
d$NormDistanceChosen = d$NormHistoricalDistance
# Baseline model
m0 = glmer(cbind(totalNFreq, totalEFreq) ~
            1 + (1|Set) + (1|EngLexeme),
            data = d, family = "binomial")
# Add the distance measure
m1 = update(m0, ~.+NormDistanceChosen)
# Add quadratic term
m2 = update(m1, ~.+I(NormDistanceChosen^2))
# Add cubic term
m3 = update(m2, ~.+I(NormDistanceChosen^3))
# Compare fit of models
anova(m0, m1, m2, m3)

## Data: d
## Models:
## m0: cbind(totalNFreq, totalEFreq) ~ 1 + (1 | Set) + (1 | EngLexeme)
## m1: cbind(totalNFreq, totalEFreq) ~ (1 | Set) + (1 | EngLexeme) + NormDistanceChosen
## m2: cbind(totalNFreq, totalEFreq) ~ (1 | Set) + (1 | EngLexeme) + NormDistanceChosen + I(NormDist
## m3: cbind(totalNFreq, totalEFreq) ~ (1 | Set) + (1 | EngLexeme) + NormDistanceChosen + I(NormDist
##      npar   AIC    BIC logLik deviance   Chisq Df Pr(>Chisq)
## m0      3 32262 32278 -16128    32256
## m1      4 32263 32284 -16127    32255   1.3764  1    0.2407
## m2      5 32146 32173 -16068    32136 118.7880  1    <2e-16 ***
## m3      6 32080 32112 -16034    32068  67.9115  1    <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Summary of final model:
summary(m3)

## Generalized linear mixed model fit by maximum likelihood (Laplace
## Approximation) [glmerMod]
## Family: binomial ( logit )
## Formula: cbind(totalNFreq, totalEFreq) ~ (1 | Set) + (1 | EngLexeme) +
##          NormDistanceChosen + I(NormDistanceChosen^2) + I(NormDistanceChosen^3)
## Data: d
##
##           AIC          BIC    logLik deviance df.resid
## 32079.9 32112.3 -16033.9 32067.9      1632
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -13.6467  -2.5893  -0.6312   0.7375  24.4681
##
## Random effects:
## Groups      Name                Variance Std.Dev.
## EngLexeme (Intercept) 1.976      1.406
## Set          (Intercept) 1.486      1.219
## Number of obs: 1638, groups: EngLexeme, 135; Set, 67
##
## Fixed effects:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)      0.8076    0.2234   3.616 0.000299 ***
## NormDistanceChosen -3.9175    0.9838 -3.982 6.83e-05 ***
```

```
## I(NormDistanceChosen^2) 16.2533      2.5269   6.432 1.26e-10 ***
## I(NormDistanceChosen^3) -15.2648      1.7883  -8.536 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##              (Intr) NrmDsC I(NDC^2
## NrmDstncChs  -0.350
## I(NrmDsC^2)   0.320 -0.978
## I(NrmDsC^3)  -0.296  0.934 -0.985
pHistoricalTotal = plot_model(m3,'eff',
  terms="NormDistanceChosen[all]")
pHistoricalTotal
```



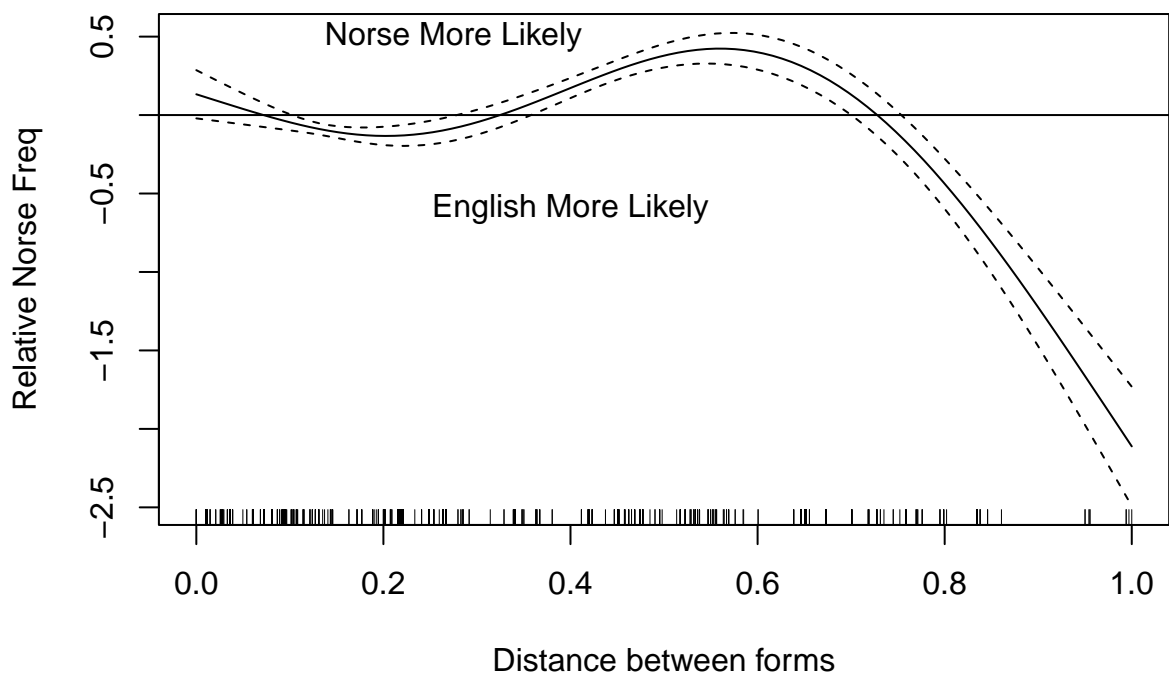
Statistics: (Linear: beta = -3.918, z = -3.982, Wald p < 0.001, LLDiff = 0.7, df = 1, p = 0.241; Quadratic: beta = 16.253, z = 6.432, Wald p < 0.001, LLDiff = 59.4, df = 1, p < 0.001; Cubic: beta = -15.265, z = -8.536, Wald p < 0.001, LLDiff = 34, df = 1, p < 0.001)

Same analysis using GAM with binomial distribution:

```
mGAM = gam(cbind(totalNFreq,totalEFreq) ~
  s(NormDistanceChosen,k=4) +
  s(Set,bs="re") +
  s(EngLexeme,bs="re"),
  data = d, family = "binomial")
summary(mGAM)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## cbind(totalNFreq, totalEFreq) ~ s(NormDistanceChosen, k = 4) +
```

```
##      s(Set, bs = "re") + s(EngLexeme, bs = "re")
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.5980      0.2378   2.514  0.0119 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##             edf Ref.df   Chi.sq p-value
## s(NormDistanceChosen)  2.998     3    163.6 <2e-16 ***
## s(Set)                 45.968    66 698441.1  0.5322
## s(EngLexeme)           80.247   134 698557.1  0.0652 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.435   Deviance explained = 45.8%
## UBRE = 16.164   Scale est. = 1           n = 1638
plot.gam(mGAM,
  ylab="Relative Norse Freq",
  xlab="Distance between forms",
  select = 1)
abline(h=0)
text(0.275,0.5,"Norse More Likely")
text(0.4,-0.6,"English More Likely")
```

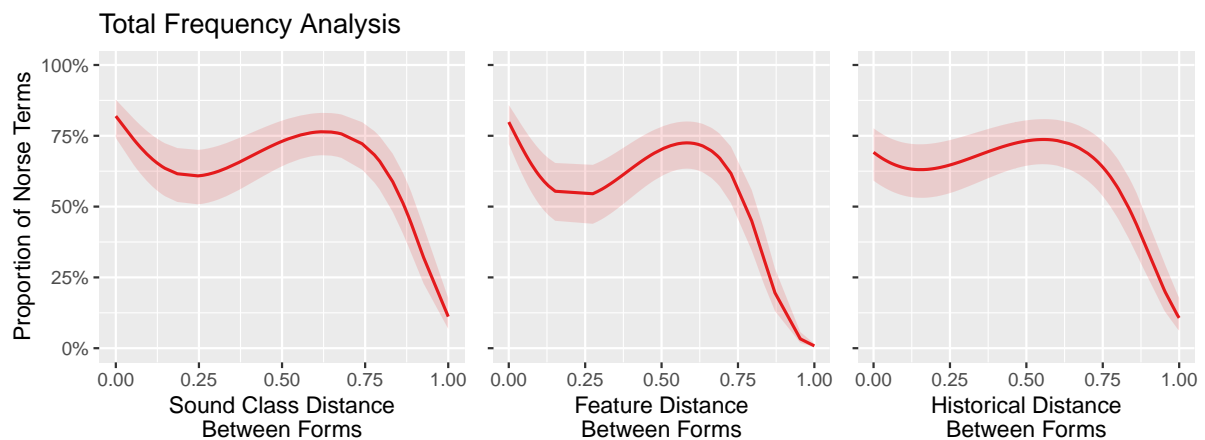


Summary of results for total frequency analyses:

```
bigPlot = grid.arrange(
  pSimpleTotal +
    ggtitle("Total Frequency Analysis") +
    coord_cartesian(ylim = c(0,1))+
    xlab("Sound Class Distance\nBetween Forms")+
    ylab("Proportion of Norse Terms"),
  pFeatureTotal+
    ggtitle("") +
    theme(axis.title.y = element_blank(),
          axis.text.y=element_blank())+
    coord_cartesian(ylim = c(0,1))+
    xlab("Feature Distance\nBetween Forms"),
  pHistoricalTotal+
    ggtitle("") +
    theme(axis.title.y = element_blank(),
          axis.text.y=element_blank())+
    coord_cartesian(ylim = c(0,1))+
    xlab("Historical Distance\nBetween Forms"),
  nrow=1,widths=c(1.3,1,1))

pdf("../results/BigEffectsPlot_totalFreq.pdf",width = 8,height=3)
plot(bigPlot)
dev.off()
```

```
## pdf
## 2
```



5.4 Exploratory analyses

5.4.1 Word Class

It is possible that lexical choices differ by word class. For example, verbs may be more resistant to integration. We test this by adding word class as a predictor. Since some word forms appear as several classes, we treat word class as a series of independent binary variables indicating the possibility or not of the word appearing as this class.

```
classes = unique(trimws(unlist(strsplit(unique(d$Class),"/"))))
classes = classes[!is.na(classes)]
classNames = paste0("Class.",gsub(" ",".",classes))
for(i in 1:length(classes)){
  d[,classNames[i]] = grepl(paste0("^",classes[i]),d$Class)
}
```

Build a null model with non-linear distance, as above:

```
d$NormDistanceChosen = d$NormDistance
mC0 = glmer(cbind(totalNFreq,totalEFreq) ~
  1 + NormDistanceChosen +
  I(NormDistanceChosen^2) +
  I(NormDistanceChosen^3) +
  (1|Set) + (1|EngLexeme),
  data = d, family = "binomial")
```

Adding the verb class variable does not improve the fit of the model:

```
mC1a = update(mC0,~.+Class.verb)
anova(mC0,mC1a)
```

```
## Data: d
## Models:
## mC0: cbind(totalNFreq, totalEFreq) ~ 1 + NormDistanceChosen + I(NormDistanceChosen^2) + I(NormDis
## mC1a: cbind(totalNFreq, totalEFreq) ~ NormDistanceChosen + I(NormDistanceChosen^2) + I(NormDistan
##      npar   AIC    BIC logLik deviance  Chisq Df Pr(>Chisq)
## mC0      6 31975 32007 -15981    31963
## mC1a     7 31977 32014 -15981    31963 0.0928  1    0.7607
```

Adding all other classes does not improve the fit of the model:

```
mC1b = update(mC0,~.+Class.adverb + Class.preposition + Class.noun +
  Class.numeral + Class.adjective + Class.verb +
  Class.indefinite.pronoun + Class.interjection)
```

```
## fixed-effect model matrix is rank deficient so dropping 2 columns / coefficients
## Warning in checkConv(attr("opt", "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.00689701 (tol = 0.002, component 1)
anova(mC0,mC1b)
```

```
## Data: d
## Models:
## mC0: cbind(totalNFreq, totalEFreq) ~ 1 + NormDistanceChosen + I(NormDistanceChosen^2) + I(NormDis
## mC1b: cbind(totalNFreq, totalEFreq) ~ NormDistanceChosen + I(NormDistanceChosen^2) + I(NormDistan
##      npar   AIC    BIC logLik deviance  Chisq Df Pr(>Chisq)
## mC0      6 31975 32007 -15981    31963
## mC1b    12 31980 32044 -15978    31956 7.0668  6    0.3147
summary(mC1b)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
## Approximation) [glmerMod]
## Family: binomial ( logit )
```



```

## Formula:
## cbind(totalNFreq, totaleFreq) ~ NormDistanceChosen + I(NormDistanceChosen^2) +
##   I(NormDistanceChosen^3) + (1 | Set) + (1 | EngLexeme) + Class.adverb +
##   Class.preposition + Class.noun + Class.numeral + Class.adjective +
##   Class.verb + Class.indefinite.pronoun + Class.interjection
## Data: d
##
##      AIC      BIC   logLik deviance df.resid
## 31979.6 32044.4 -15977.8 31955.6      1626
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -13.6715  -2.5875  -0.6247   0.7114  24.4498
##
## Random effects:
##  Groups      Name      Variance Std.Dev.
## EngLexeme (Intercept) 1.992     1.411
## Set      (Intercept) 1.178     1.085
## Number of obs: 1638, groups: EngLexeme, 135; Set, 67
##
## Fixed effects:
##
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)      1.2177     0.7815   1.558   0.119
## NormDistanceChosen -10.4432     0.9922 -10.526 <2e-16 ***
## I(NormDistanceChosen^2) 30.8080     2.3382  13.176 <2e-16 ***
## I(NormDistanceChosen^3) -23.9496     1.6074 -14.899 <2e-16 ***
## Class.adverbTRUE      0.9804     0.9147   1.072   0.284
## Class.nounTRUE        0.1828     0.8329   0.220   0.826
## Class.numeralTRUE     1.8335     1.1919   1.538   0.124
## Class.adjectiveTRUE   -0.4274     0.9183  -0.465   0.642
## Class.verbTRUE        0.1505     0.8427   0.179   0.858
## Class.indefinite.pronounTRUE 1.0948     1.6814   0.651   0.515
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##      (Intr) NrmDsC I(NDC^2 I(NDC^3 Clss.dvTRUE Clss.nnTRUE Clss.nmTRUE
## NrmDstncChs -0.129
## I(NrmDsC^2)  0.114 -0.969
## I(NrmDsC^3) -0.101  0.915 -0.983
## Clss.dvTRUE -0.833 -0.013  0.010 -0.008
## Clss.nnTRUE -0.917 -0.005  0.008 -0.009  0.781
## Clss.nmTRUE -0.636  0.009 -0.008  0.005  0.541  0.594
## Clss.djTRUE -0.831 -0.006  0.005 -0.005  0.708  0.785  0.539
## Clss.vrTRUE -0.861 -0.004  0.007 -0.007  0.732  0.805  0.557
## Clss.n.TRUE -0.454 -0.014  0.016 -0.019  0.388  0.427  0.296
##      Clss.djTRUE Clss.vTRUE
## NrmDstncChs
## I(NrmDsC^2)
## I(NrmDsC^3)
## Clss.dvTRUE
## Clss.nnTRUE
## Clss.nmTRUE
## Clss.djTRUE
## Clss.vrTRUE  0.729
## Clss.n.TRUE  0.387      0.400
## fit warnings:
## fixed-effect model matrix is rank deficient so dropping 2 columns / coefficients

```

```
## optimizer (Nelder_Mead) convergence code: 0 (OK)
## Model failed to converge with max|grad| = 0.00689701 (tol = 0.002, component 1)
```

5.4.2 Diagnostic consonants

Some Norse words have more salient diagnostic segments - they sound ‘more Norse’. For example, the presence /sk/ vs post-alveolar fricative. Below, we create a variable that distinguishes pairs where the Norse includes /sk/ (and the English includes / /) from other cases, and enter this as a variable into the model to test whether it improves the prediction of the proportion of Norse usage.

```
d$NorseDiagnosticSK =
  grepl("sk",d$NorseForm) &
  grepl(" ",d$EngForm) &
  !grepl("sk",d$EngForm)
# (this isn't valid for all possible words,
# but does identify the relevant data in
# our sample)

d$NorseDiagnosticSK = factor(d$NorseDiagnosticSK)

d$NormDistanceChosen = d$NormDistance
mD0 = glmer(cbind(totalNFreq,totaleFreq) ~
  1 + NormDistanceChosen +
  I(NormDistanceChosen^2) +
  I(NormDistanceChosen^3) +
  (1|Set) + (1|EngLexeme),
  data = d, family = "binomial")
mD1 = update(mD0, ~.+NorseDiagnosticSK)
anova(mD0,mD1)
```

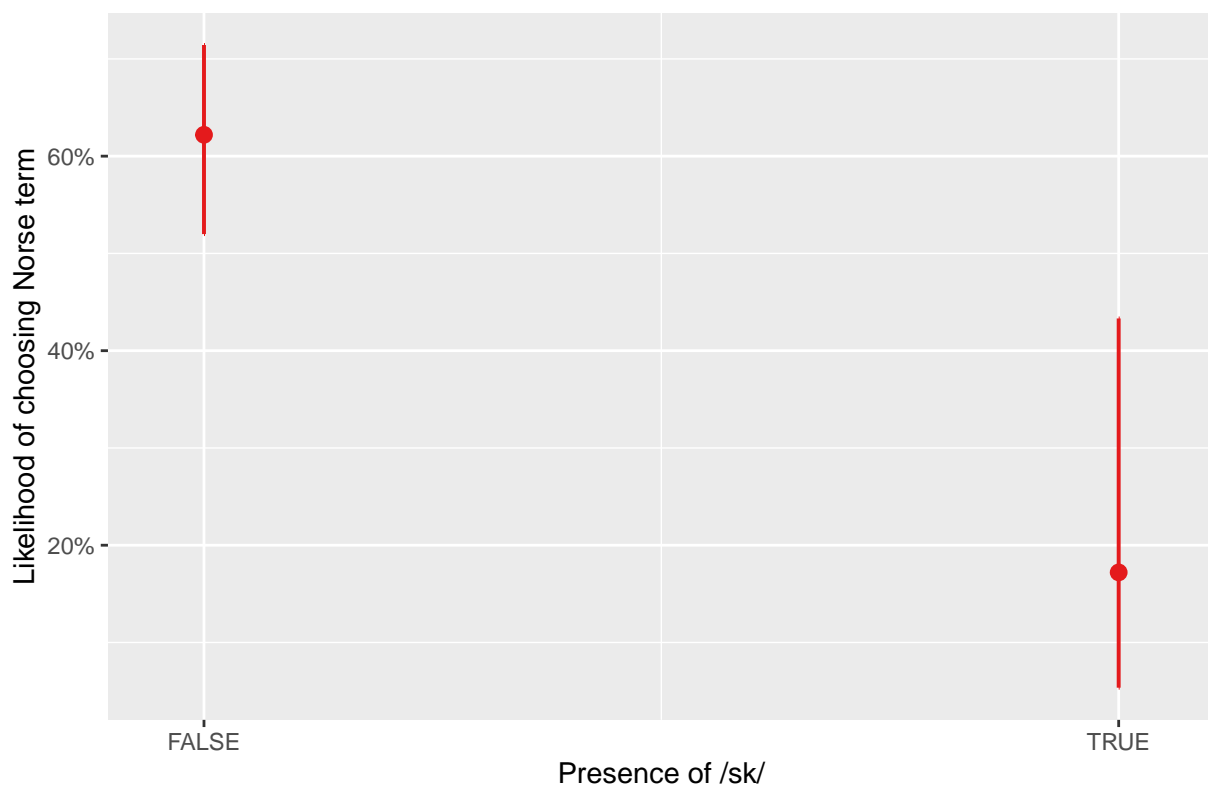
```
## Data: d
## Models:
## mD0: cbind(totalNFreq, totaleFreq) ~ 1 + NormDistanceChosen + I(NormDistanceChosen^2) + I(NormDis
## mD1: cbind(totalNFreq, totaleFreq) ~ NormDistanceChosen + I(NormDistanceChosen^2) + I(NormDistanc
##      npar    AIC    BIC logLik deviance  Chisq Df Pr(>Chisq)
## mD0      6 31975 32007 -15981    31963
## mD1      7 31966 32004 -15976    31952 10.236  1  0.001378 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

summary(mD1)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
## Approximation) [glmerMod]
## Family: binomial ( logit )
## Formula:
## cbind(totalNFreq, totaleFreq) ~ NormDistanceChosen + I(NormDistanceChosen^2) +
## I(NormDistanceChosen^3) + (1 | Set) + (1 | EngLexeme) + NorseDiagnosticSK
## Data: d
##
##      AIC      BIC    logLik deviance df.resid
## 31966.4 32004.2 -15976.2 31952.4    1631
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -13.6634  -2.5875  -0.6253   0.7084  24.4509
##
## Random effects:
## Groups      Name      Variance Std.Dev.
## EngLexeme (Intercept) 2.041    1.429
```

```
## Set      (Intercept) 1.563    1.250
## Number of obs: 1638, groups: EngLexeme, 135; Set, 67
##
## Fixed effects:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)      1.5681    0.2365   6.630 3.37e-11 ***
## NormDistanceChosen -10.5306    0.9237 -11.401 < 2e-16 ***
## I(NormDistanceChosen^2) 31.1074    2.1647  14.370 < 2e-16 ***
## I(NormDistanceChosen^3) -24.1643    1.4907 -16.210 < 2e-16 ***
## NorseDiagnosticSKTRUE   -2.0690    0.6480  -3.193  0.00141 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##              (Intr) NrmDsC I(NDC^2 I(NDC^3
## NrmDstncChs -0.421
## I(NrmDsC^2)  0.374 -0.965
## I(NrmDsC^3) -0.333  0.901 -0.981
## NrsDgSKTRUE -0.090  0.030 -0.040  0.042
```

```
plot_model(mD1, 'eff', terms="NorseDiagnosticSK") +
  ylab("Likelihood of choosing Norse term") +
  xlab("Presence of /sk/") +
  ggtitle("")
```



```
get_model_data(mD1, 'eff', terms="NorseDiagnosticSK")
```

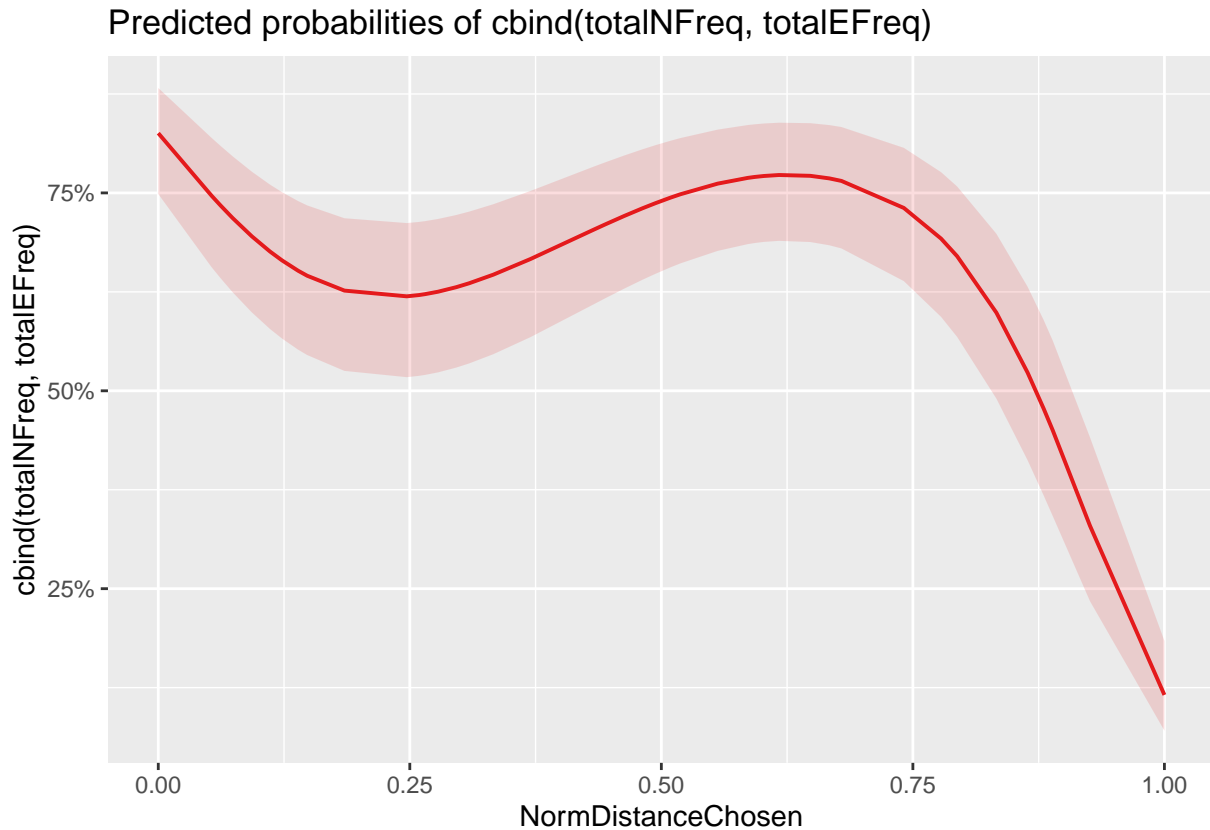
```
## # Predicted probabilities of cbind(totalNFreq, totalEFreq)
##
## NorseDiagnosticSK | Predicted |      95% CI | group_col
## -----
##              1 |      0.62 | 0.52, 0.71 |      1
##              2 |      0.17 | 0.05, 0.43 |      1
```

Adding the Norse Diagnostic Score to the model significantly improves the fit of the model. The model

suggests that Norse forms which are more obviously Norse have a lower chance of being selected overall. This is in line with the general findings: words that are obviously more different to the English forms are less likely to be integrated.

The effect of form distance is unaffected by adding the presence of 'sk' as a variable:

```
plot_model(mD1, 'eff', terms="NormDistanceChosen[all]")
```

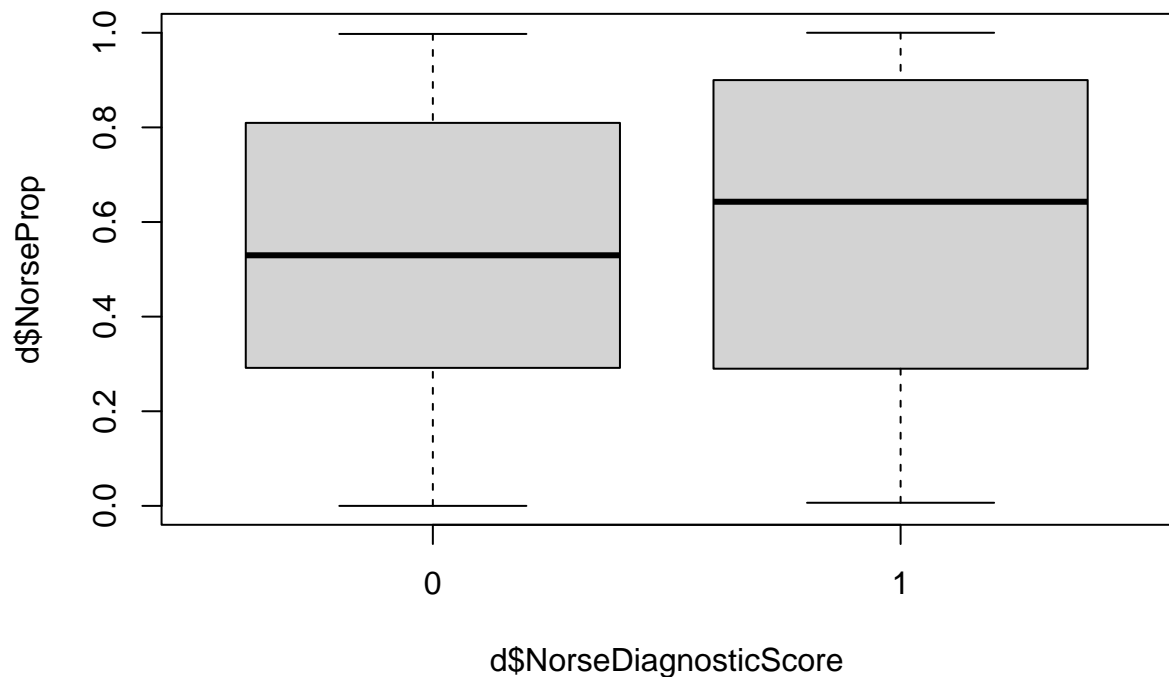


There are other consonant contrasts which are also diagnostic of Norse origin. For example, the presence of /g/ vs /j/ or /k/ vs post-alveolar affricate. However, these are only truly diagnostic in production when followed by a front vowel. Still, ignoring this complication for now, we calculate a Norse Diagnostic Score: cases where these segments are diagnostic of the differences between Norse and English forms (the score is potentially more than one, but in our data there are only zero and one):

```
d$NorseDiagnosticScore =
  (grepl("g",d$NorseForm) & !grepl("g",d$EngForm))+
  (grepl(" ",d$NorseForm) & !grepl(" ",d$EngForm)) +
  (grepl("sk",d$NorseForm) & !grepl("sk",d$EngForm))+
  # (k in Norse vs in English)
  (grepl("k",d$NorseForm) & grepl(" ",d$EngForm))

d$NorseDiagnosticScore = factor(d$NorseDiagnosticScore)

boxplot(d$NorseProp~ d$NorseDiagnosticScore)
```



```
t.test(d$NorseProp~ d$NorseDiagnosticScore)
```

```
##
## Welch Two Sample t-test
##
## data: d$NorseProp by d$NorseDiagnosticScore
## t = -2.2293, df = 1532.9, p-value = 0.02594
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0
## 95 percent confidence interval:
## -0.065524499 -0.004187601
## sample estimates:
## mean in group 0 mean in group 1
## 0.5501452 0.5850013
```

Adding the Norse Diagnostic Score to the model significantly improves the fit of the model. The model suggests that Norse forms which are more obviously Norse have a lower chance of being selected overall.

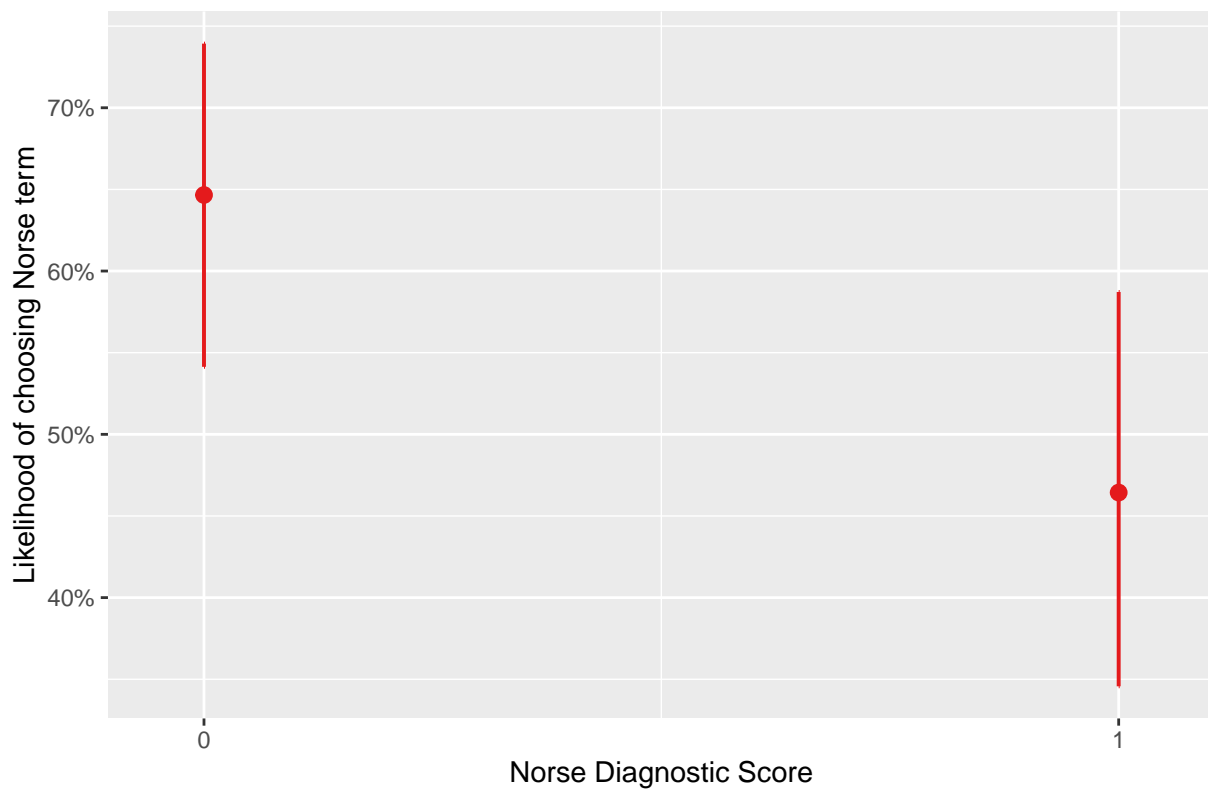
```
d$NormDistanceChosen = d$NormDistance
mD0 = glmer(cbind(totalNFreq,totalEFreq) ~
  1 + NormDistanceChosen +
  I(NormDistanceChosen^2) +
  I(NormDistanceChosen^3) +
  (1|Set) + (1|EngLexeme),
  data = d, family = "binomial")
mD1 = update(mD0, ~.+NorseDiagnosticScore)
anova(mD0,mD1)
```

```
## Data: d
## Models:
## mD0: cbind(totalNFreq, totalEFreq) ~ 1 + NormDistanceChosen + I(NormDistanceChosen^2) + I(NormDis
## mD1: cbind(totalNFreq, totalEFreq) ~ NormDistanceChosen + I(NormDistanceChosen^2) + I(NormDistanc
##      npar   AIC   BIC logLik deviance Chisq Df Pr(>Chisq)
## mD0     6 31975 32007 -15981   31963
## mD1     7 31954 31992 -15970   31940 22.871  1 1.732e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(mD1)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
## Family: binomial ( logit )
## Formula:
## cbind(totalNFreq, totaleFreq) ~ NormDistanceChosen + I(NormDistanceChosen^2) +
##   I(NormDistanceChosen^3) + (1 | Set) + (1 | EngLexeme) + NorseDiagnosticScore
## Data: d
##
##      AIC      BIC   logLik deviance df.resid
## 31953.8 31991.6 -15969.9 31939.8     1631
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -13.6651  -2.5587  -0.6102   0.7161  24.4515
##
## Random effects:
## Groups      Name                Variance Std.Dev.
## EngLexeme (Intercept) 2.024      1.423
## Set          (Intercept) 1.774      1.332
## Number of obs: 1638, groups: EngLexeme, 135; Set, 67
##
## Fixed effects:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)      1.6604    0.2476   6.705 2.01e-11 ***
## NormDistanceChosen -10.3816    0.9876 -10.512 < 2e-16 ***
## I(NormDistanceChosen^2) 30.6307    2.3291  13.151 < 2e-16 ***
## I(NormDistanceChosen^3) -23.7685    1.6035 -14.823 < 2e-16 ***
## NorseDiagnosticScore1  -0.7469    0.1583  -4.717 2.39e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##              (Intr) NrmDsC I(NDC^2 I(NDC^3
## NrmDstncChs -0.422
## I(NrmDsC^2)  0.379 -0.969
## I(NrmDsC^3) -0.341  0.913 -0.983
## NrsDgnstcS1 -0.138 -0.010  0.011 -0.014

plot_model(mD1,'eff', terms="NorseDiagnosticScore") +
  ylab("Likelihood of choosing Norse term") +
  xlab("Norse Diagnostic Score") +
  ggtitle("")
```



```
get_model_data(mD1, 'eff', terms="NorseDiagnosticScore")
```

```
## # Predicted probabilities of cbind(totalNFreq, totalEFreq)
```

```
##
```

```
## NorseDiagnosticScore | Predicted | 95% CI | group_col
```

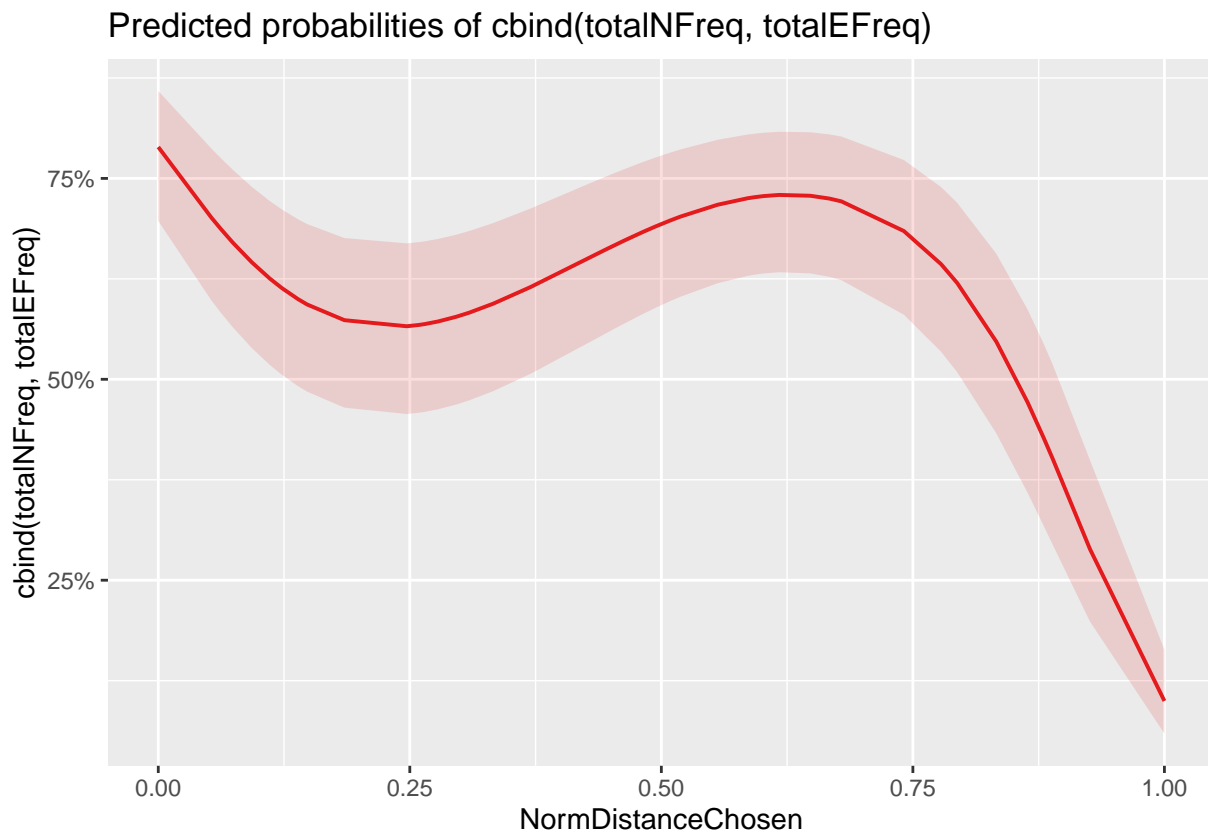
```
## -----
```

```
##           0 | 0.65 | 0.54, 0.74 | 1
```

```
##           1 | 0.46 | 0.35, 0.59 | 1
```

The effect of form distance is unaffected:

```
plot_model(mD1, 'eff', terms="NormDistanceChosen[all]")
```



6 Source-level frequency analysis

This analysis uses source-level observations: each observation is the frequency of a pair of forms within a particular source. This also lets us control for source-level features like the source itself as a random effect, the age of the source, and whether alliteration might affect decisions.

There may be multiple orthographic forms for each unique full form for comparison, so first, we collapse the data over unique pairs. We also restructure the data to be in 'long' format.

```
d2 = data.frame()
for(i in 1:length(englishFrequencyColumns)){
  dx = data.frame(
    Set = d$Set,
    EngForm = d$EngForm,
    NorseForm = d$NorseForm,
    NormDistance = d$NormDistance,
    NormFeatureDistance = d$NormFeatureDistance,
    NormHistoricalDistance = d$NormHistoricalDistance,
    NFreq = d[,norseFrequencyColumns[i]],
    EFreq = d[,englishFrequencyColumns[i]],
    Source = englishFrequencyColumns[i],
    Alliteration = d$Alliteration)
  dx = dx[rowSums(dx[,c("NFreq", "EFreq")], na.rm = T) > 0, ]
  uforms = paste(dx$Set, dx$EngForm, dx$NorseForm)
  dx = data.frame(
    Set = tapply(dx$Set, uforms, head, n=1),
    EngForm = tapply(dx$EngForm, uforms, head, n=1),
    NorseForm = tapply(dx$NorseForm, uforms, head, n=1),
    NormDistance = tapply(dx$NormDistance, uforms, head, n=1),
    NormFeatureDistance = tapply(dx$NormFeatureDistance, uforms, head, n=1),
    NormHistoricalDistance = tapply(dx$NormHistoricalDistance, uforms, head, n=1),
    NFreq = tapply(dx$NFreq, uforms, sum, na.rm=T),
```



```

  EFreq = tapply(dx$EFreq,uforms,sum,na.rm=T),
  Source = englishFrequencyColumns[i],
  Alliteration = tapply(dx$Alliteration,uforms,head,n=1))
d2 = rbind(d2,dx)
}

d2$Set = factor(d2$Set)
d2$Source = factor(d2$Source)

# Make a variable for each unique form within a set
d2$EngForm2 = paste(d2$Set,d2$EngForm)
d2$NorseForm2 = paste(d2$Set,d2$NorseForm)
d2$EngForm2 = factor(d2$EngForm2)

# Remove any cases with zero frequency
d2 = d2[!(d2$NFreq==0 & d2$EFreq==0),]

# Proportion of norse forms
d2$NProp = 100*(d2$NFreq/(d2$NFreq+d2$EFreq))

```

Modelling age is difficult, since exact dates are not known. It would be possible to model the date of publication as an ordinal variable, for example assuming FCPC > Ormulum > {Havelok, GenAndEx} > CursorMundi > {Mannyng, GawainPoet} > WarsAlexander > StErkenwald. However, the texts from the Corpus of Middle English span several centuries, and so are hard to place in this order. Instead, we simply use the century as an ordered category.

```

AgeCategories =
  c("EFreqFCPC"=12,
    "EFreqOrmulum"=12,
    "EFreqHavelok"=13,
    "EFreqGenAndEx" = 14,
    "EFreqCursorMundi" = 14,
    "EFreqGawainPoet" = 14,
    "EFreqStErkenwald" = 15,
    "EFreqMannyng" = 15,
    "EFreqWarsAlexander" = 15,
    "EFreqLinc"=15,
    "EFreqNott"=15,
    "EFreqNorf"=15,
    "EFreqRolle"=15)

d2$Age = AgeCategories[d2$Source]
d2$Age = factor(d2$Age,ordered=T)

contrasts(d2$Age) = contr.sum(length(unique(d2$Age)))

mean(d2$NFreq/(d2$NFreq+d2$EFreq))

## [1] 0.5579405

range(d2$NFreq/(d2$NFreq+d2$EFreq))

## [1] 0 1

mean(d2$NormDistance)

## [1] 0.3594087

```

Alliteration only applies to poetry sources: Gawain, St Erkenwold, and Wars of Alexander. So turn all others to 'false':

```

d2[!d2$Source %in%
  c("EFreqGawainPoet",
    "EFreqStErkenwald",
    "EFreqWarsAlexander",
    "EFreqLinc",
    "EFreqNott",
    "EFreqNorf",
    "EFreqRolle"),]$Alliteration = FALSE
d2$Alliteration = factor(d2$Alliteration)

gPairs = ggpairs(d2[,c("NormDistance", "NormFeatureDistance",
  "NormHistoricalDistance", "NProp")],
  columnLabels = c("Sound Class Distance",
    "Feature Distance",
    "Historical Distance",
    "Freq (% Norse)"),
  rowLabels = c("Sound Class Distance",
    "Feature Distance",
    "Historical Distance",
    "Freq (% Norse)"))

pdf("../results/GPairs.pdf", width=6, height=5.5)
gPairs
dev.off()

## pdf
## 2

```

6.1 Simple distance

Use a mixed effects model to predict the frequency of Norse and English forms by a random effect for cognate Set and Source. We add a main effect of age of source, then introduce the normalised distance measure along with its non-linear terms.

```

m0 = glmer(cbind(NFreq, EFreq) ~ Age + Alliteration +
  (1|Set) + (1|Source) +
  (1|EngForm2),
  data = d2, family = "binomial",
  glmerControl(optimizer = "bobyqa"))
# Add the distance measure
m1 = update(m0, ~. + NormDistance)
# Add quadratic term
m2 = update(m1, ~. + I(NormDistance^2))
# Add cubic term
m3 = update(m2, ~. + I(NormDistance^3))

```

Compare fit of models:

```
anova(m0, m1, m2, m3)
```

```

## Data: d2
## Models:
## m0: cbind(NFreq, EFreq) ~ Age + Alliteration + (1 | Set) + (1 | Source) + (1 | EngForm2)
## m1: cbind(NFreq, EFreq) ~ Age + Alliteration + (1 | Set) + (1 | Source) + (1 | EngForm2) + NormDi
## m2: cbind(NFreq, EFreq) ~ Age + Alliteration + (1 | Set) + (1 | Source) + (1 | EngForm2) + NormDi
## m3: cbind(NFreq, EFreq) ~ Age + Alliteration + (1 | Set) + (1 | Source) + (1 | EngForm2) + NormDi
##      npar   AIC    BIC logLik deviance   Chisq Df Pr(>Chisq)
## m0      8 21104 21145 -10544    21088
## m1      9 20936 20981 -10459    20918 170.622  1 < 2.2e-16 ***
## m2     10 20875 20926 -10428    20855  62.509  1 2.652e-15 ***

```

```
## m3    11 20547 20602 -10262    20525 330.399 1 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Summary of final model:

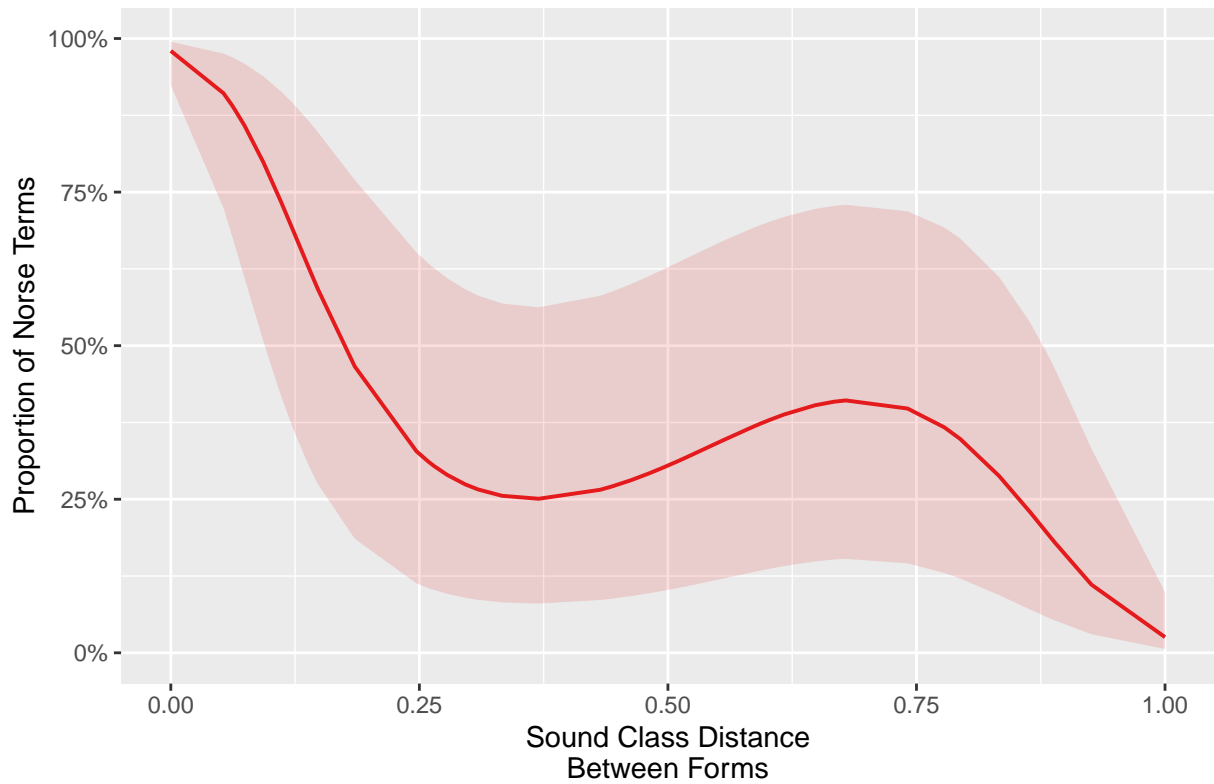
```
summary(m3)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
## Family: binomial ( logit )
## Formula: cbind(NFreq, EFreq) ~ Age + Alliteration + (1 | Set) + (1 | Source) +
##   (1 | EngForm2) + NormDistance + I(NormDistance^2) + I(NormDistance^3)
## Data: d2
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC    logLik deviance df.resid
## 20546.6 20602.4 -10262.3 20524.6    1162
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -105.030   -1.362    0.114    1.127   175.531
##
## Random effects:
## Groups Name Variance Std.Dev.
## EngForm2 (Intercept) 4.302 2.074
## Set (Intercept) 5.354 2.314
## Source (Intercept) 4.389 2.095
## Number of obs: 1173, groups: EngForm2, 129; Set, 67; Source, 13
##
## Fixed effects:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)    3.554398   0.834099   4.261 2.03e-05 ***
## Age1           1.062382   1.278355   0.831 0.406
## Age2          -1.281640   1.644782  -0.779 0.436
## Age3          -0.007456   1.124018  -0.007 0.995
## AlliterationTRUE 2.233356   0.071790 31.110 < 2e-16 ***
## NormDistance   -32.968226   1.575406 -20.927 < 2e-16 ***
## I(NormDistance^2) 68.761447   3.535850  19.447 < 2e-16 ***
## I(NormDistance^3) -43.320731   2.343141 -18.488 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##              (Intr) Age1   Age2   Age3   AlTRUE NrmDst I(ND^2
## Age1           0.010
## Age2          0.398 -0.521
## Age3          -0.185 -0.260 -0.490
## AlltrtnTRUE -0.012 0.015 -0.010 0.000
## NormDistanc -0.211 -0.007 0.005 0.001 -0.026
## I(NrmDst^2) 0.189 0.007 -0.005 0.000 0.025 -0.968
## I(NrmDst^3) -0.169 -0.007 0.006 0.000 -0.026 0.908 -0.981
```

Norse frequency varies with distance. The plot below shows the marginal effects, holding discrete predictors constant at their proportions (not reference level):

```
pNormSimple =
  plot_model(m3, 'eff', terms="NormDistance[all]") +
  xlab("Sound Class Distance\nBetween Forms") +
  ylab("Proportion of Norse Terms") +
```

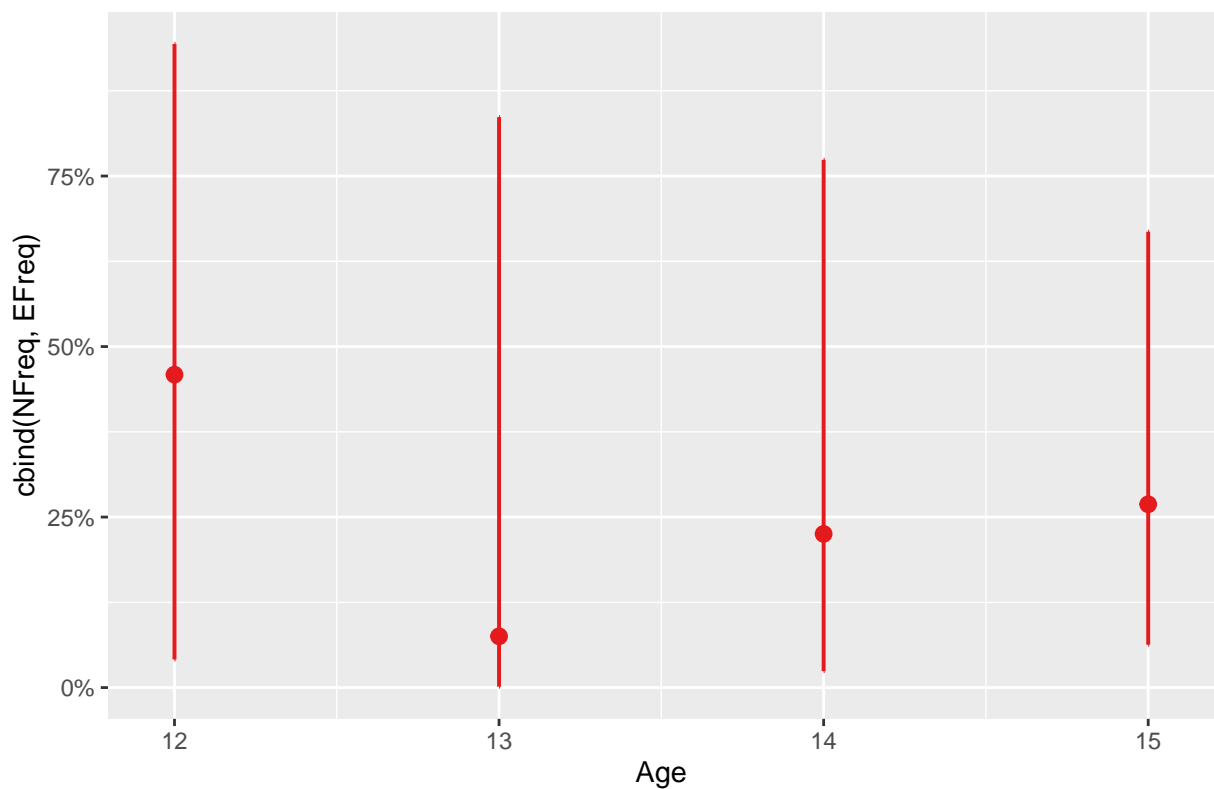
```
ggtitle("")+
  coord_cartesian(ylim=c(0,1))
pNormSimple
```



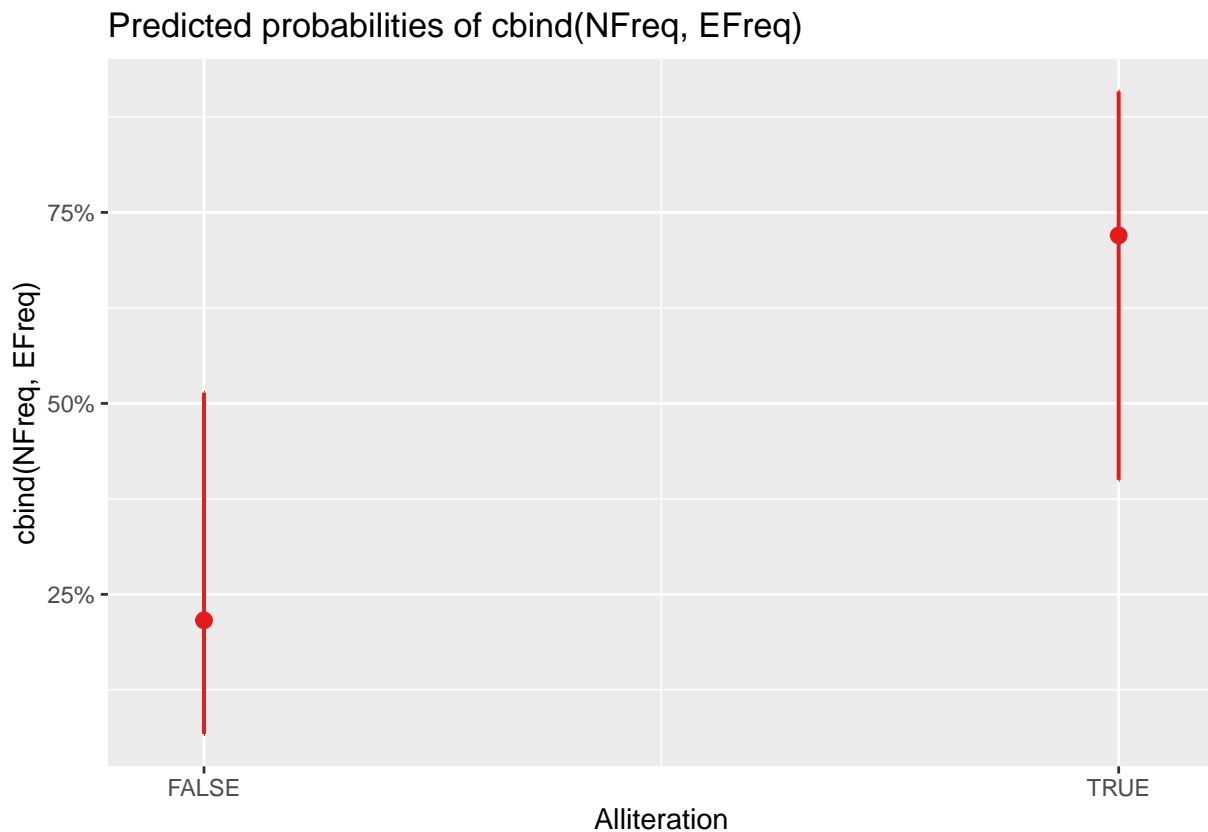
Effects over time and of alliteration:

```
plot_model(m3, 'eff', terms="Age [all]")
```

Predicted probabilities of cbind(NFreq, EFreq)



```
plot_model(m3, 'eff', terms="Alliteration")
```

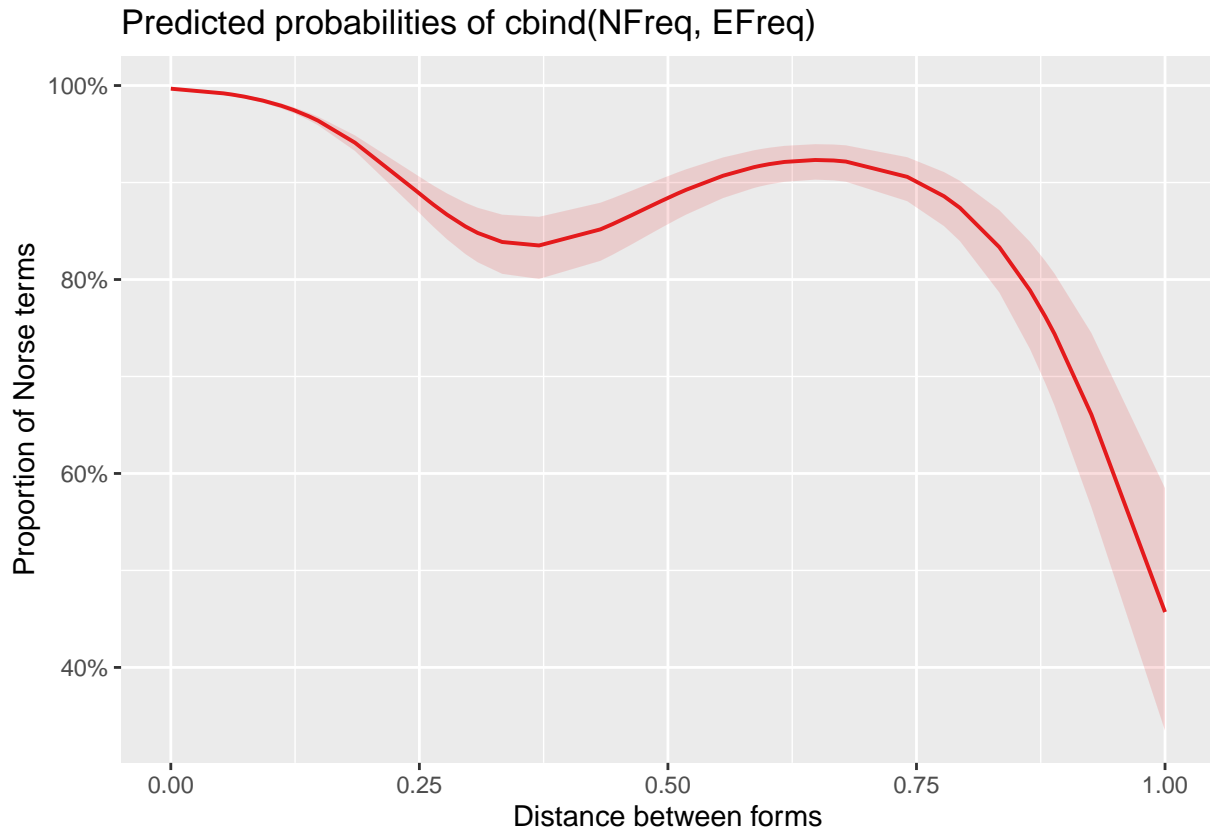


The GAM model seems a little different, showing a smaller effect at lower distances.

```
mGAM = gam(cbind(NFreq, EFreq) ~
            s(NormDistance, k=4) +
            s(Age, bs = "re") +
            s(Set, bs="re") +
            s(Source, bs="re")+
            s(EngForm2, bs="re"),
            data = d2, family = "binomial")
summary(mGAM)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## cbind(NFreq, EFreq) ~ s(NormDistance, k = 4) + s(Age, bs = "re") +
##      s(Set, bs = "re") + s(Source, bs = "re") + s(EngForm2, bs = "re")
##
## Parametric coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.001498   2.230288  -0.001    0.999
##
## Approximate significance of smooth terms:
##              edf Ref.df   Chi.sq p-value
## s(NormDistance)  2.99967     3 4.539e+02 < 2e-16 ***
## s(Age)           0.03305     3 2.027e+03 0.99821
## s(Set)          49.04776    66 3.235e+07 0.10624
## s(Source)       11.96498    12 1.112e+07 < 2e-16 ***
## s(EngForm2)     72.08085   128 3.721e+07 0.00747 **
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.709   Deviance explained = 69.2%
## UBRE = 16.512   Scale est. = 1          n = 1173
plot_model(mGAM,"pred",terms="NormDistance") +
  ylab("Proportion of Norse terms") +
  xlab("Distance between forms")
```



Statistics for the simple measure:

(Linear: $\beta = -32.968$, $z = -20.927$, Wald $p < 0.001$, LLDiff = 85.3, $df = 1$, $p < 0.001$; Quadratic: $\beta = 68.761$, $z = 19.447$, Wald $p < 0.001$, LLDiff = 31.3, $df = 1$, $p < 0.001$; Cubic: $\beta = -43.321$, $z = -18.488$, Wald $p < 0.001$, LLDiff = 165.2, $df = 1$, $p < 0.001$)

6.2 Feature-based distance

Same analysis as the simple distance above:

```
m0 = glmer(cbind(NFreq,EFreq) ~ Age + Alliteration
           + (1|Set) + (1|Source) +
           (1|EngForm2),
           data = d2, family = "binomial")
m1 = update(m0,~.+NormFeatureDistance)
m2 = update(m1,~.+I(NormFeatureDistance^2))
m3 = update(m2,~.+I(NormFeatureDistance^3))
```

```
## Warning in checkConv(attr("derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.0469466 (tol = 0.002, component 1)

anova(m0,m1,m2,m3)
```

```
## Data: d2
```

```
## Models:
```

```
## m0: cbind(NFreq, EFreq) ~ Age + Alliteration + (1 | Set) + (1 | Source) + (1 | EngForm2)
```

```
## m1: cbind(NFreq, EFreq) ~ Age + Alliteration + (1 | Set) + (1 | Source) + (1 | EngForm2) + NormFe
```

```
## m2: cbind(NFreq, EFreq) ~ Age + Alliteration + (1 | Set) + (1 | Source) + (1 | EngForm2) + NormFe
```

```
## m3: cbind(NFreq, EFreq) ~ Age + Alliteration + (1 | Set) + (1 | Source) + (1 | EngForm2) + NormFe
```

```
##      npar   AIC    BIC logLik deviance   Chisq Df Pr(>Chisq)
```

```
## m0      8 21104 21145 -10544    21088
```

```
## m1      9 20901 20946 -10442    20883 205.280  1 < 2.2e-16 ***
```

```
## m2     10 20857 20907 -10418    20837  46.349  1 9.895e-12 ***
```

```
## m3     11 20582 20638 -10280    20560 276.202  1 < 2.2e-16 ***
```

```
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(m3)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
```

```
##   Approximation) [glmerMod]
```

```
## Family: binomial ( logit )
```

```
## Formula: cbind(NFreq, EFreq) ~ Age + Alliteration + (1 | Set) + (1 | Source) +
```

```
##   (1 | EngForm2) + NormFeatureDistance + I(NormFeatureDistance^2) +
```

```
##   I(NormFeatureDistance^3)
```

```
## Data: d2
```

```
##
```

```
##      AIC      BIC    logLik deviance df.resid
```

```
## 20582.3 20638.1 -10280.2 20560.3      1162
```

```
##
```

```
## Scaled residuals:
```

```
##      Min       1Q   Median       3Q      Max
```

```
## -102.933   -1.381    0.116    1.096   178.703
```

```
##
```

```
## Random effects:
```

```
## Groups   Name              Variance Std.Dev.
```

```
## EngForm2 (Intercept) 4.740      2.177
```

```
## Set       (Intercept) 5.840      2.417
```

```
## Source   (Intercept) 4.365      2.089
```

```
## Number of obs: 1173, groups:  EngForm2, 129; Set, 67; Source, 13
```

```
##
```

```
## Fixed effects:
```

```
##              Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)    1.899237   0.829291   2.290   0.022 *
```

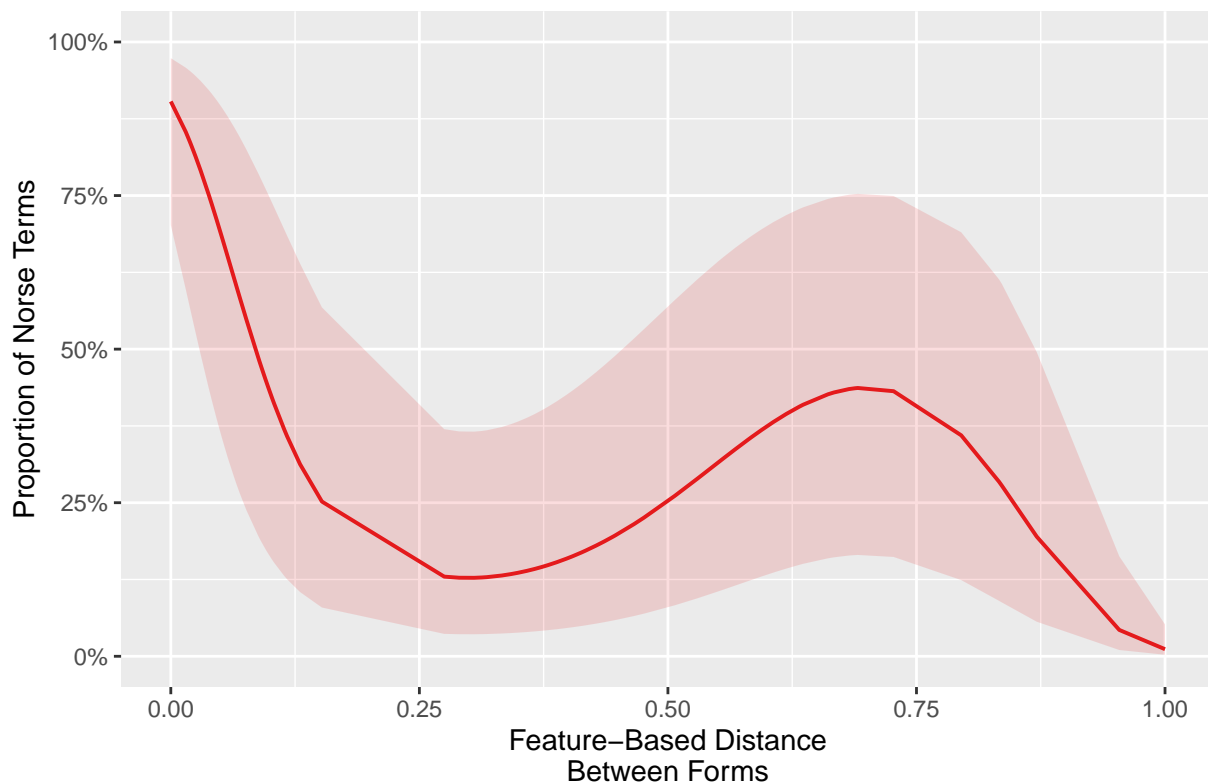
```
## Age1            1.066301   1.278915   0.834   0.404
```

```
## Age2           -1.291551   1.646010  -0.785   0.433
```

```
## Age3           -0.005518   1.123471  -0.005   0.996
```

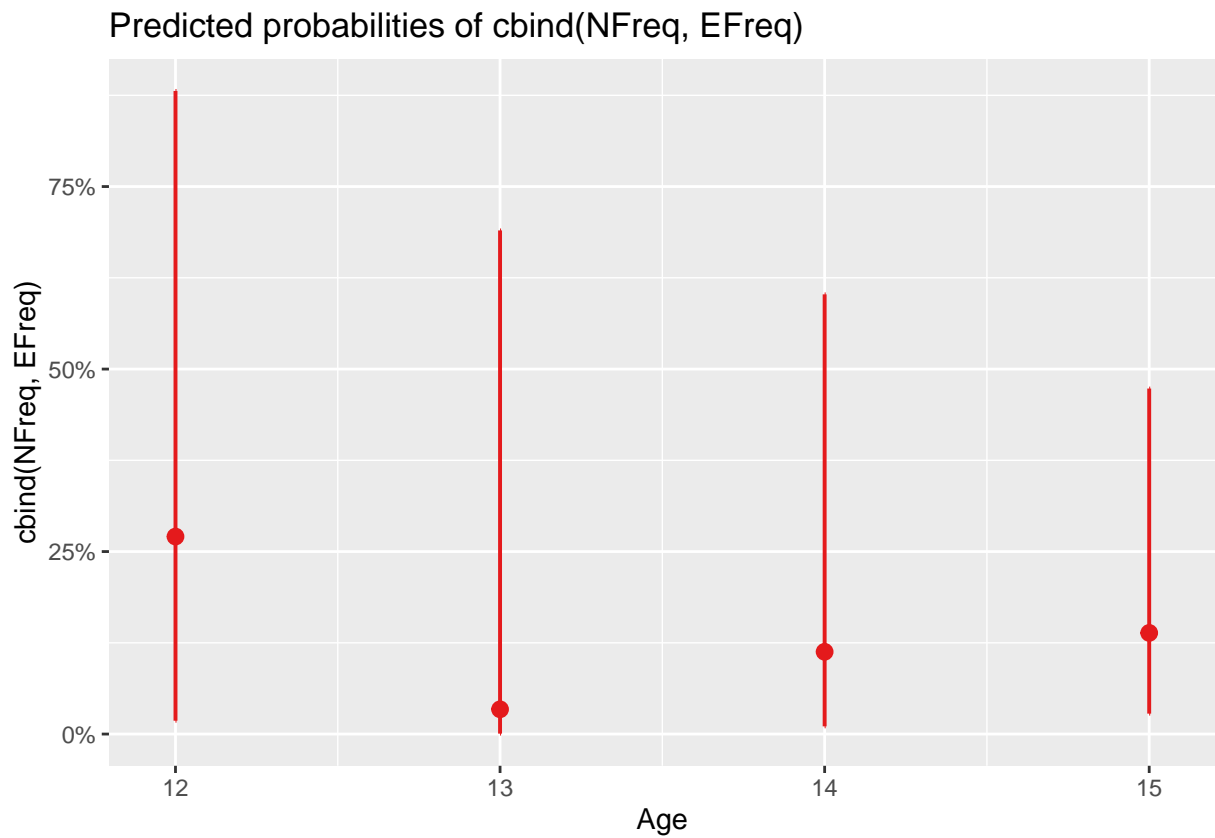
```
## AlliterationTRUE          2.251681    0.071937   31.301    <2e-16 ***
## NormFeatureDistance      -32.430222    1.781901  -18.200    <2e-16 ***
## I(NormFeatureDistance^2)  77.441357    4.551277   17.015    <2e-16 ***
## I(NormFeatureDistance^3) -51.690896    3.157168  -16.373    <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) Age1   Age2   Age3   AlTRUE NrmFtD I(NFD^2
## Age1          0.006
## Age2          0.404 -0.522
## Age3          -0.187 -0.259 -0.490
## AlltrtnTRUE  -0.014  0.015 -0.010 -0.001
## NrmFtrDstnc  -0.144  0.005 -0.017  0.011 -0.024
## I(NrmFtD^2)   0.135 -0.006  0.017 -0.011  0.026 -0.981
## I(NrmFtD^3)  -0.125  0.006 -0.017  0.010 -0.028  0.934 -0.984
## optimizer (Nelder_Mead) convergence code: 0 (OK)
## Model failed to converge with max|grad| = 0.0469466 (tol = 0.002, component 1)
```

```
pNormFeature =
  plot_model(m3,'eff', terms="NormFeatureDistance[all]")+
    xlab("Feature-Based Distance\nBetween Forms") +
    ylab("Proportion of Norse Terms") +
    ggtitle("")+
    coord_cartesian(ylim=c(0,1))
pNormFeature
```

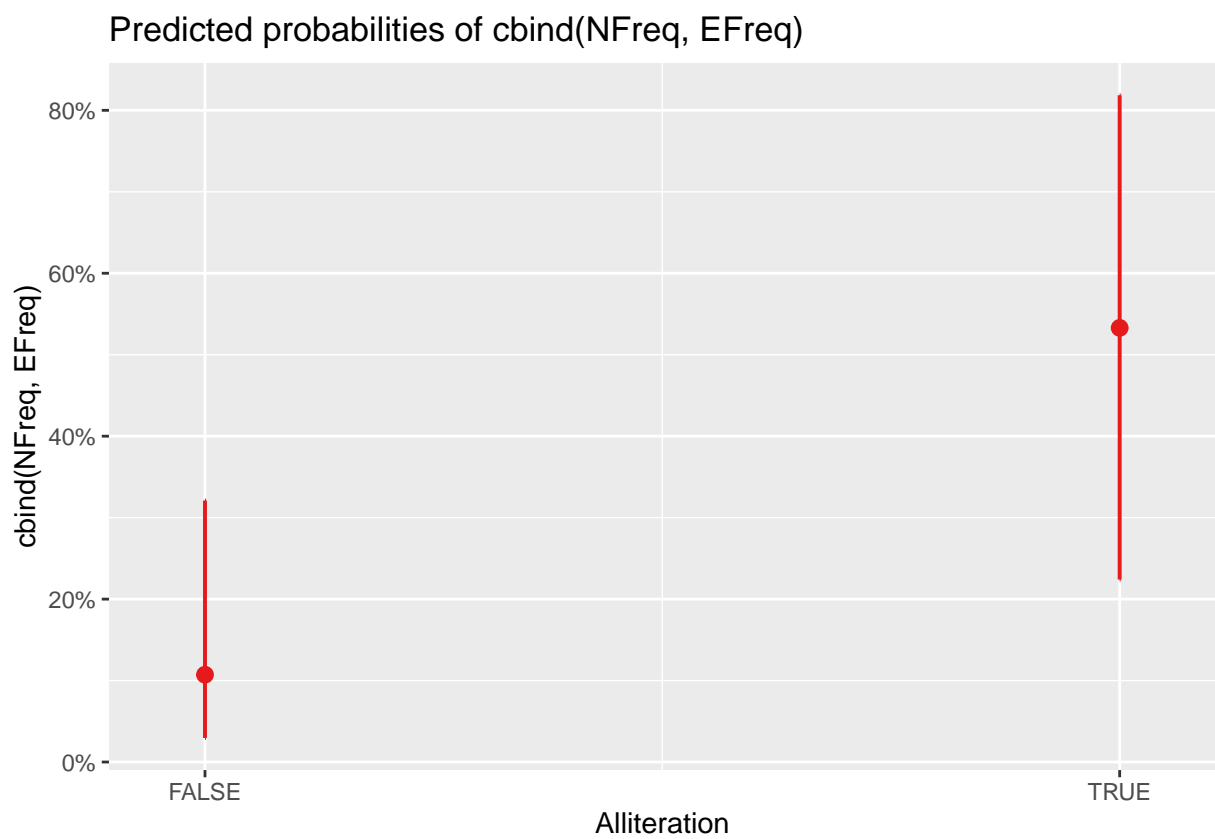


Effects over time and of alliteration:

```
plot_model(m3,'eff', terms="Age [all]")
```

```
plot_model(m3, 'eff', terms="Alliteration")
```



GAM model:

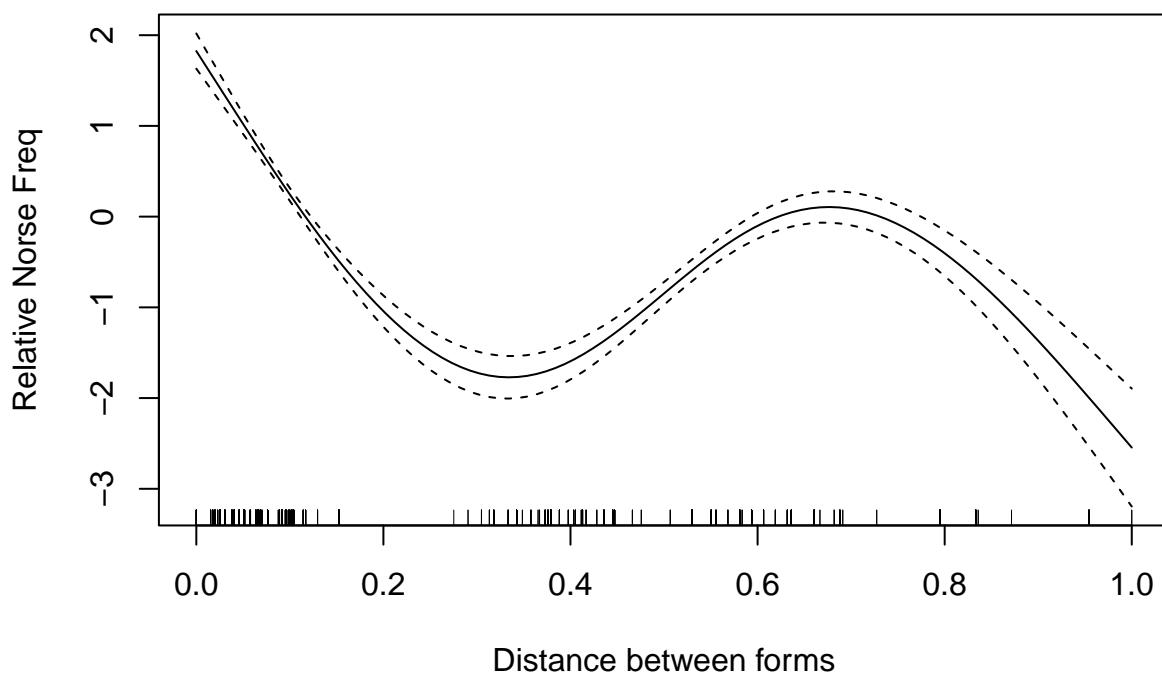
```
mGAM = gam(cbind(NFreq, EFreq) ~  
            s(NormFeatureDistance, k=4) +
```

```

s(Age,bs = "re") +
s(Set,bs="re") +
s(Source,bs="re")+
s(EngForm2,bs="re"),
data = d2, family = "binomial")
summary(mGAM)

##
## Family: binomial
## Link function: logit
##
## Formula:
## cbind(NFreq, EFreq) ~ s(NormFeatureDistance, k = 4) + s(Age,
##   bs = "re") + s(Set, bs = "re") + s(Source, bs = "re") + s(EngForm2,
##   bs = "re")
##
## Parametric coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.09027    2.81544   0.032   0.974
##
## Approximate significance of smooth terms:
##               edf Ref.df   Chi.sq  p-value
## s(NormFeatureDistance)  2.99978     3 3.674e+02 < 2e-16 ***
## s(Age)                  0.04248     3 4.889e+03  0.9993
## s(Set)                  52.70122    66 7.617e+07  0.0244 *
## s(Source)               11.95629    12 1.658e+07 6.75e-05 ***
## s(EngForm2)             69.37364   128 2.384e+07  0.1411
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.711   Deviance explained = 69.1%
## UBRE = 16.565   Scale est. = 1          n = 1173
plot.gam(mGAM,
  ylab="Relative Norse Freq",
  xlab="Distance between forms",
  select = 1)

```



Linear: $\beta = -32.43$, $z = -18.2$, Wald $p < 0.001$, LLDiff = 102.6, $df = 1$, $p < 0.001$; Quadratic: $\beta = 77.441$, $z = 17.015$, Wald $p < 0.001$, LLDiff = 23.2, $df = 1$, $p < 0.001$; Cubic: $\beta = -51.691$, $z = -16.373$, Wald $p < 0.001$, LLDiff = 138.1, $df = 1$, $p < 0.001$

6.3 Historical distance

Same analysis as the simple distance above:

```
m0 = glmer(cbind(NFreq,EFreq) ~ Age + Alliteration +
           (1|Set) + (1|Source) +
           (1|EngForm2),
           data = d2, family = "binomial",
           control=glmerControl(optimizer="bobyqa"))
m1 = update(m0,~.+NormHistoricalDistance)
m2 = update(m1,~.+I(NormHistoricalDistance^2),
           control=glmerControl(optimizer="bobyqa"))
m3 = update(m2,~.+I(NormHistoricalDistance^3),
           control=glmerControl(optimizer="bobyqa"))
anova(m0,m1,m2,m3)

## Data: d2
## Models:
## m0: cbind(NFreq, EFreq) ~ Age + Alliteration + (1 | Set) + (1 | Source) + (1 | EngForm2)
## m1: cbind(NFreq, EFreq) ~ Age + Alliteration + (1 | Set) + (1 | Source) + (1 | EngForm2) + NormHi
## m2: cbind(NFreq, EFreq) ~ Age + Alliteration + (1 | Set) + (1 | Source) + (1 | EngForm2) + NormHi
## m3: cbind(NFreq, EFreq) ~ Age + Alliteration + (1 | Set) + (1 | Source) + (1 | EngForm2) + NormHi
##      npar   AIC    BIC logLik deviance   Chisq Df Pr(>Chisq)
## m0      8 21104 21145 -10544    21088
## m1      9 20904 20949 -10443    20886 202.5263  1 < 2.2e-16 ***
## m2     10 20901 20952 -10441    20881  4.5342  1  0.03322 *
## m3     11 20879 20935 -10429    20857 23.9212  1 1.004e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

For this measure, the cubic term is marginal.

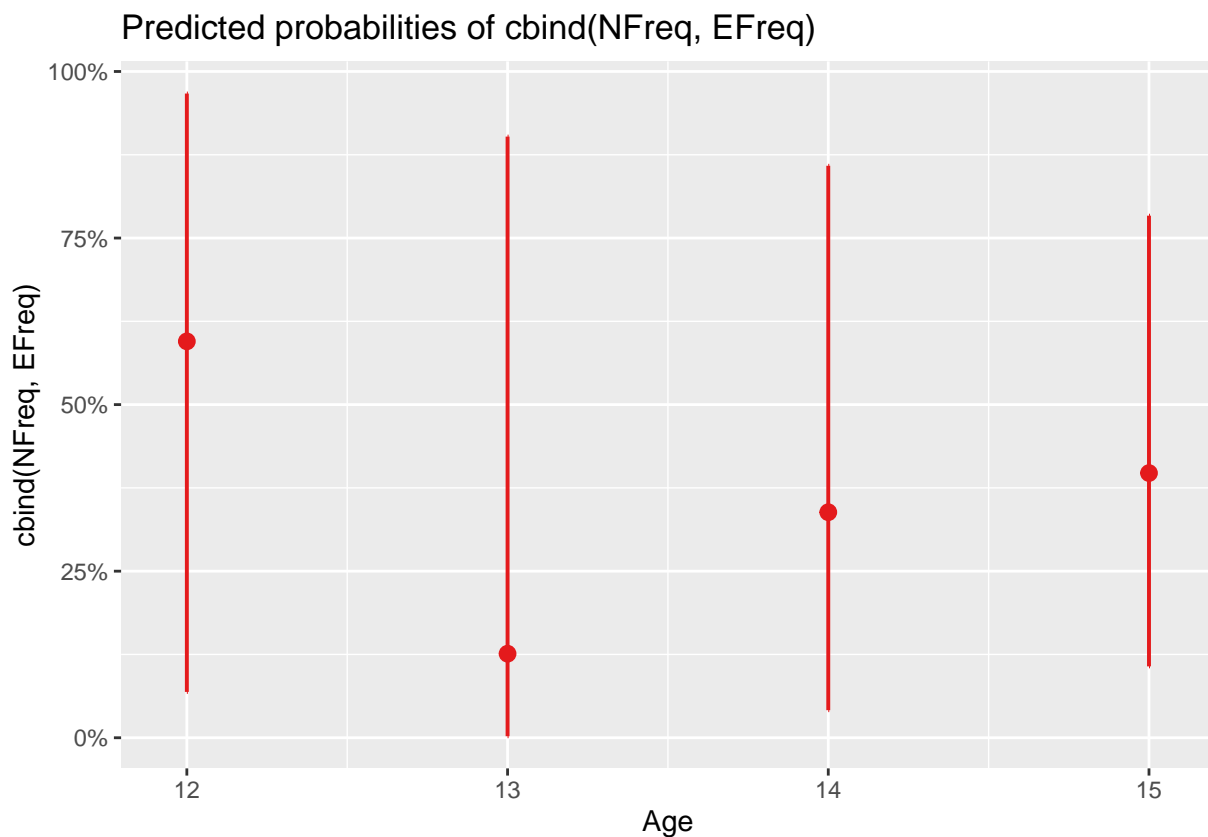
This is the cubic model:

```
summary(m3)

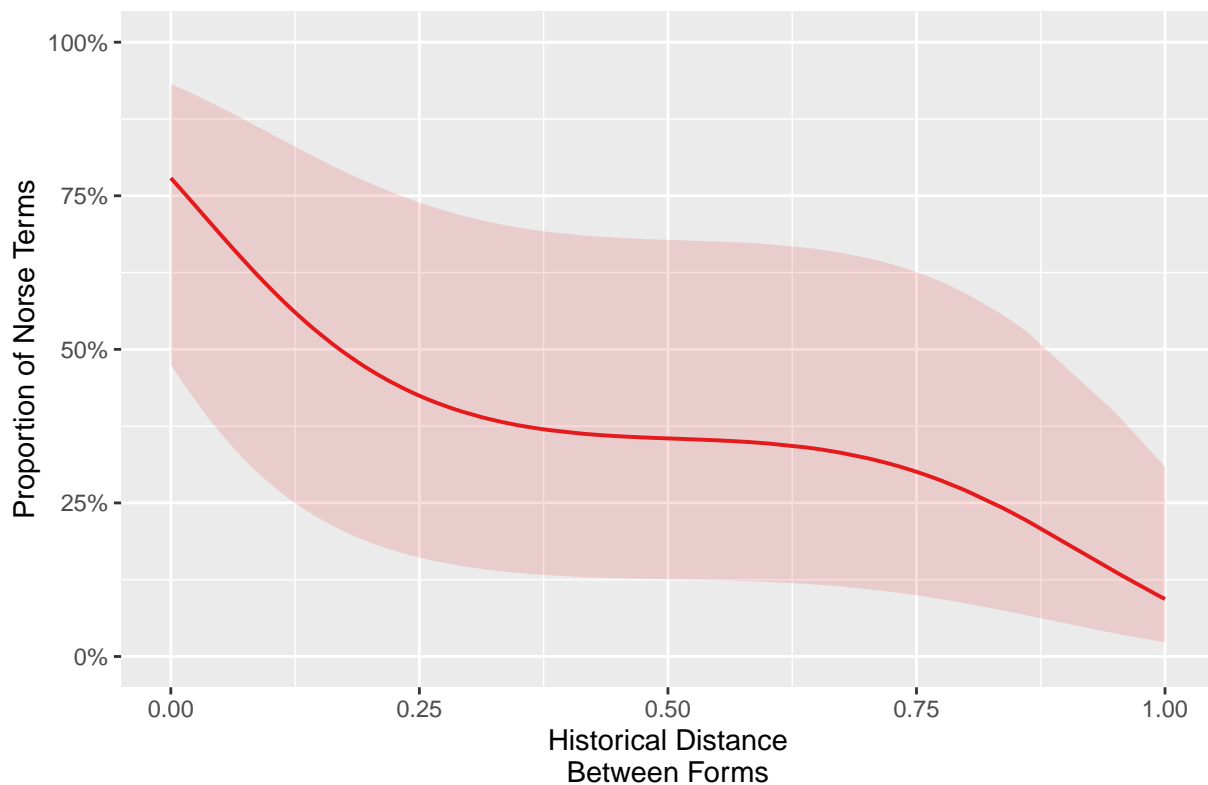
## Generalized linear mixed model fit by maximum likelihood (Laplace
## Approximation) [glmerMod]
## Family: binomial ( logit )
## Formula: cbind(NFreq, EFreq) ~ Age + Alliteration + (1 | Set) + (1 | Source) +
## (1 | EngForm2) + NormHistoricalDistance + I(NormHistoricalDistance^2) +
## I(NormHistoricalDistance^3)
## Data: d2
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
## 20879.2 20934.9 -10428.6  20857.2    1162
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -110.625   -1.325    0.118    1.129   171.595
##
## Random effects:
## Groups   Name                Variance Std.Dev.
## EngForm2 (Intercept) 4.065      2.016
## Set      (Intercept) 5.239      2.289
## Source   (Intercept) 4.425      2.104
## Number of obs: 1173, groups: EngForm2, 129; Set, 67; Source, 13
##
## Fixed effects:
##
##              Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)          0.92307      0.82206      1.123      0.261
## Age1                 1.04395      1.28208      0.814      0.415
## Age2                -1.27604      1.65242     -0.772      0.440
## Age3                -0.01052      1.13003     -0.009      0.993
## AlliterationTRUE      2.24683      0.07204     31.190 < 2e-16 ***
## NormHistoricalDistance -10.42926      1.41162     -7.388 1.49e-13 ***
## I(NormHistoricalDistance^2) 19.96760      3.64632      5.476 4.35e-08 ***
## I(NormHistoricalDistance^3) -13.06820      2.52493     -5.176 2.27e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##      (Intr) Age1   Age2   Age3   AlTRUE NrmHsD I(NHD^2)
## Age1      0.008
## Age2      0.407 -0.519
## Age3     -0.189 -0.260 -0.492
## AlltrtnTRUE -0.014  0.015 -0.010  0.000
## NrmHstrclDs -0.128  0.003 -0.001 -0.002 -0.025
## I(NrmHsD^2)  0.114 -0.003  0.001  0.002  0.031 -0.976
## I(NrmHsD^3) -0.104  0.002  0.000 -0.002 -0.040  0.929 -0.984
```

```
plot_model(m3,'eff', terms="Age [all]")
```



```
pNormHist =
  plot_model(m3,'eff', terms="NormHistoricalDistance[all]") +
  xlab("Historical Distance\nBetween Forms") +
  ylab("Proportion of Norse Terms") +
  ggtitle("") +
  coord_cartesian(ylim=c(0,1))
pNormHist
```



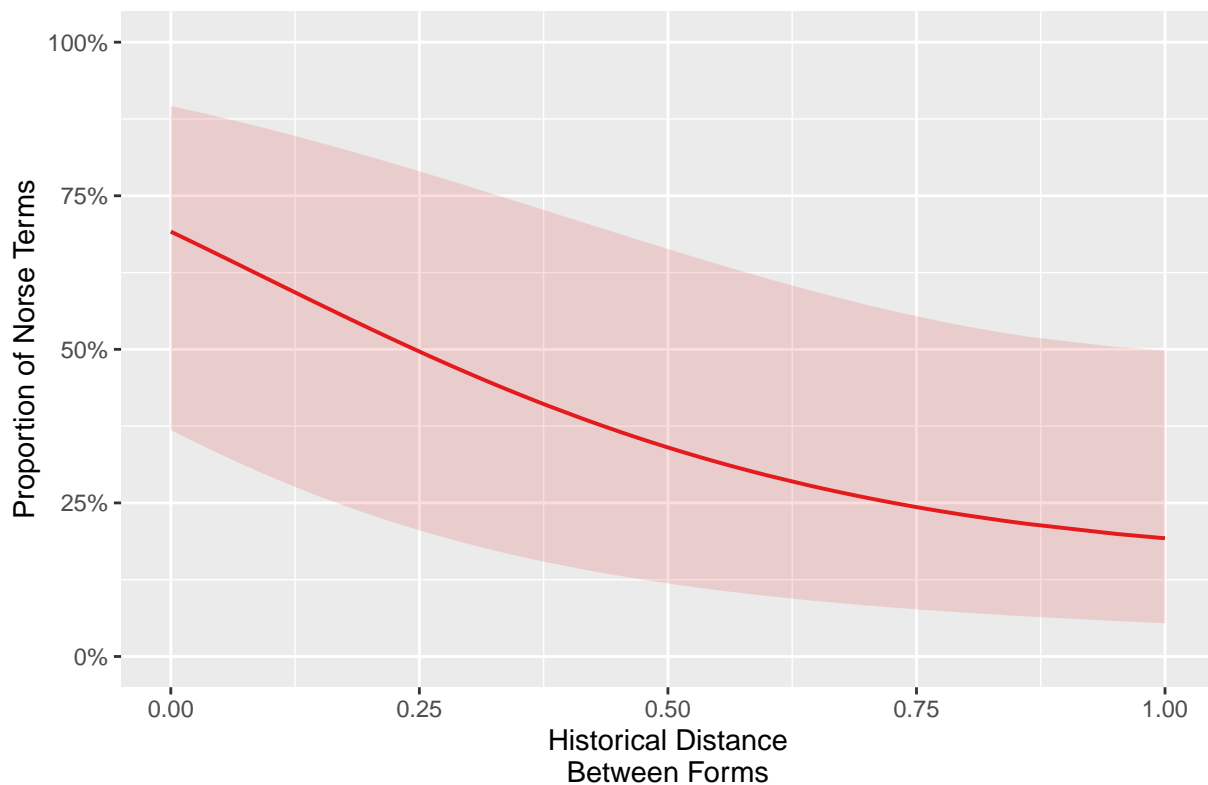
And for comparison, the quadratic model:

```
summary(m2)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
## Approximation) [glmerMod]
## Family: binomial ( logit )
## Formula: cbind(NFreq, EFreq) ~ Age + Alliteration + (1 | Set) + (1 | Source) +
## (1 | EngForm2) + NormHistoricalDistance + I(NormHistoricalDistance^2)
## Data: d2
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC   logLik deviance df.resid
## 20901.1 20951.8 -10440.6 20881.1    1163
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -110.404   -1.347    0.116    1.150   171.447
##
## Random effects:
##  Groups   Name                Variance Std.Dev.
## EngForm2 (Intercept)  4.043      2.011
## Set       (Intercept)  5.192      2.279
## Source   (Intercept)  4.413      2.101
## Number of obs: 1173, groups:  EngForm2, 129; Set, 67; Source, 13
##
## Fixed effects:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)      0.47469   0.81655   0.581   0.5610
## Age1             1.03689   1.28216   0.809   0.4187
## Age2            -1.26463   1.65211  -0.765   0.4440
## Age3            -0.01151   1.12830  -0.010   0.9919
## AlliterationTRUE    2.23317   0.07193  31.047 < 2e-16 ***
## NormHistoricalDistance -3.63796   0.52262 -6.961 3.38e-12 ***
```

```
## I(NormHistoricalDistance^2) 1.39642 0.65371 2.136 0.0327 *
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) Age1  Age2  Age3  AlTRUE NrmHsD
## Age1      0.007
## Age2      0.410 -0.520
## Age3     -0.190 -0.259 -0.491
## AlltrtnTRUE -0.019 0.015 -0.010 0.000
## NrmHstrclDs -0.086 0.000 -0.001 0.000 0.037
## I(NrmHsD^2) 0.065 -0.001 0.001 -0.001 -0.053 -0.931
```

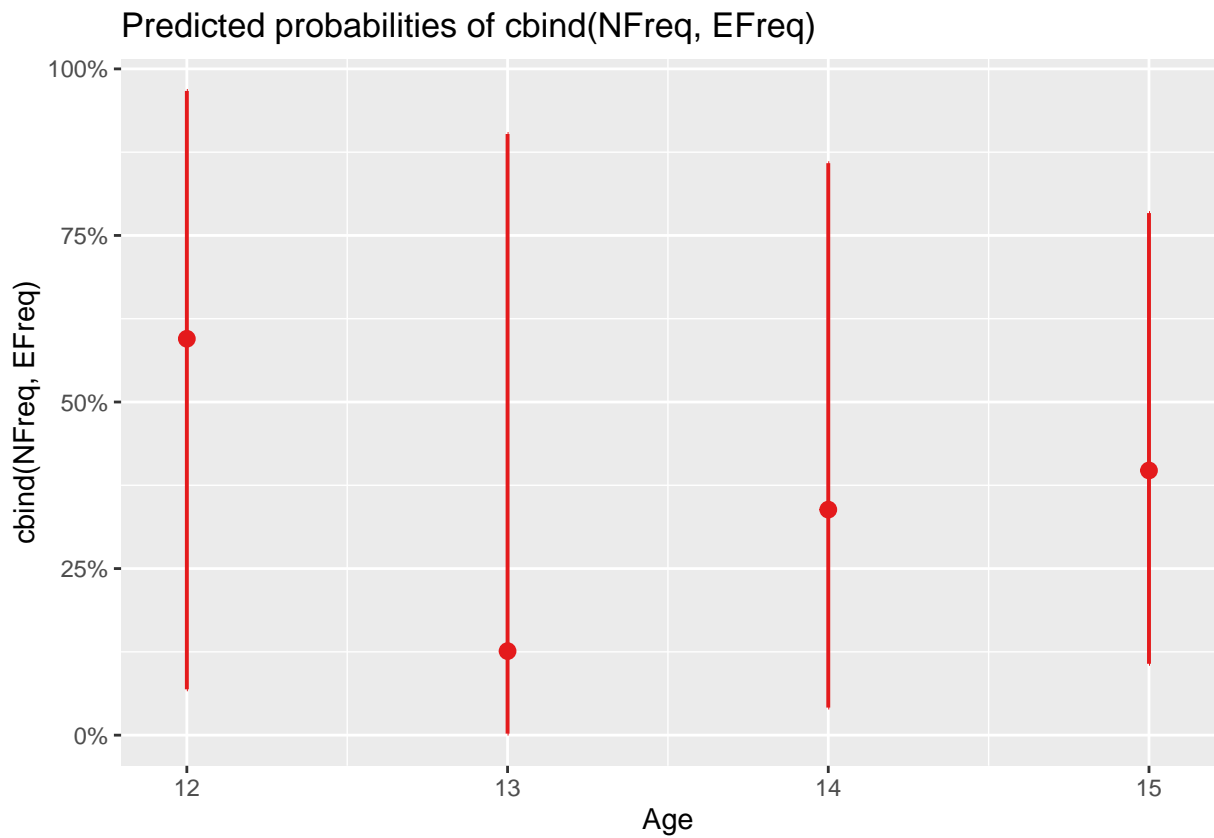
```
plot_model(m2, 'eff', terms="NormHistoricalDistance[all]") +
  xlab("Historical Distance\nBetween Forms") +
  ylab("Proportion of Norse Terms") +
  ggtitle("") +
  coord_cartesian(ylim=c(0,1))
```



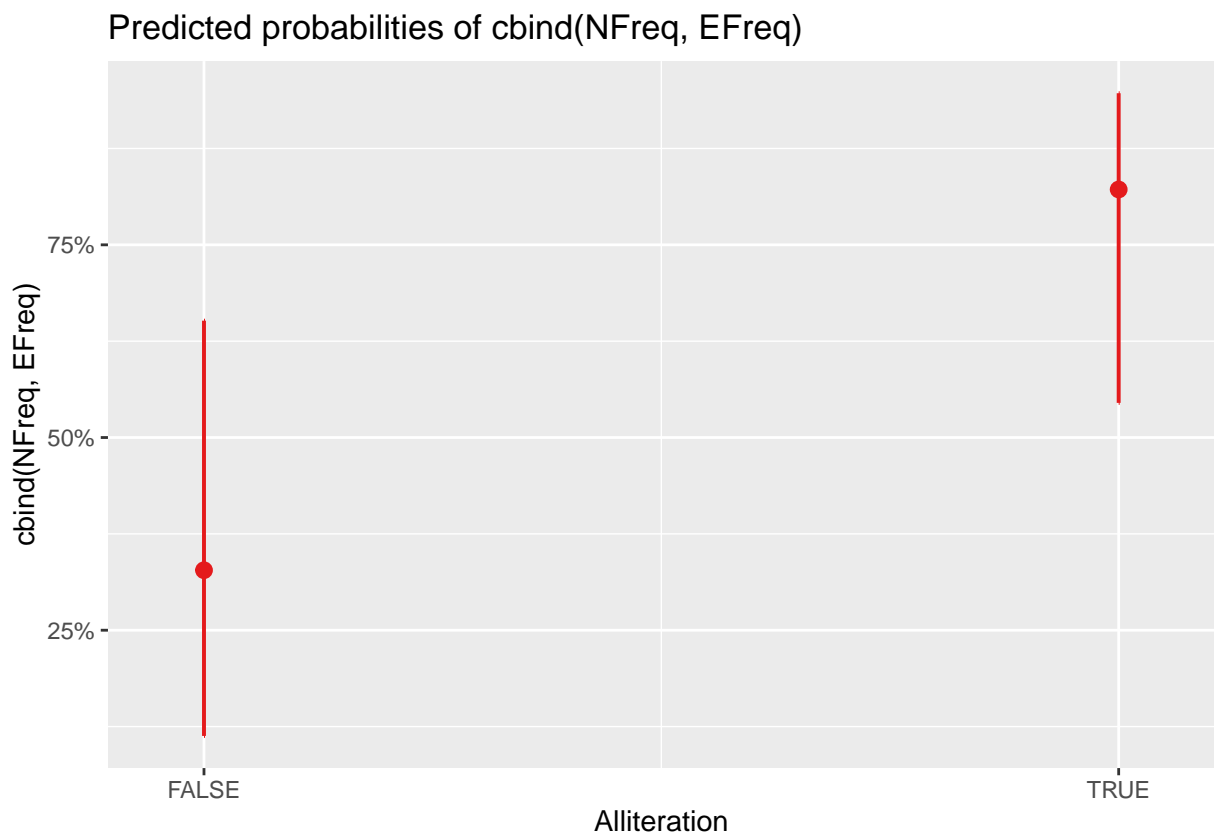
There is little qualitative difference, so for sake of easy comparison with the other results, we use the cubic model.

Effects over time and of alliteration:

```
plot_model(m3, 'eff', terms="Age [all]")
```



```
plot_model(m3, 'eff', terms="Alliteration")
```



GAM model:

```
mGAM = gam(cbind(NFreq, EFreq) ~ Alliteration +  
            s(NormHistoricalDistance, k=4) +
```

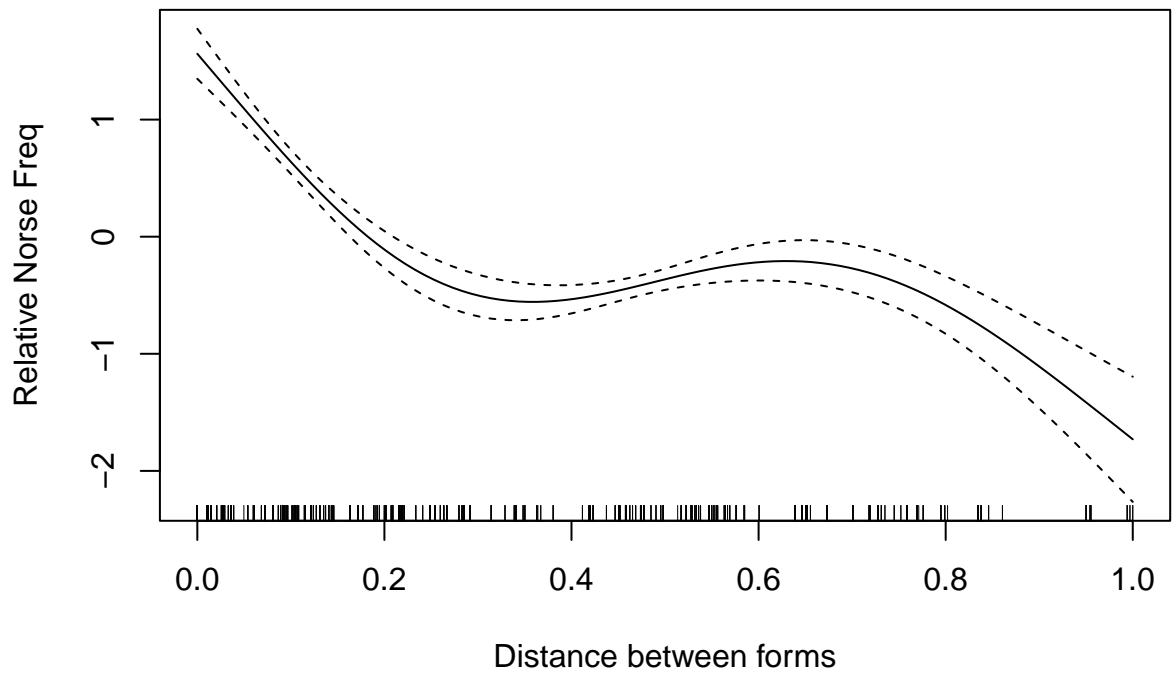


```

      s(Age,bs = "re") +
      s(Set,bs="re") +
      s(Source,bs="re")+
      s(EngForm2,bs="re"),
      data = d2, family = "binomial")
summary(mGAM)

##
## Family: binomial
## Link function: logit
##
## Formula:
## cbind(NFreq, EFreq) ~ Alliteration + s(NormHistoricalDistance,
##    k = 4) + s(Age, bs = "re") + s(Set, bs = "re") + s(Source,
##    bs = "re") + s(EngForm2, bs = "re")
##
## Parametric coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -0.35137    2.38755  -0.147   0.883
## AlliterationTRUE 2.24532    0.07209  31.144  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##              edf Ref.df    Chi.sq p-value
## s(NormHistoricalDistance) 2.995951     3 2.472e+02 < 2e-16 ***
## s(Age)                    0.001067     3 1.164e+00 0.999276
## s(Set)                    47.845260    66 6.216e+07 0.000642 ***
## s(Source)                 11.996842    12 1.098e+07 < 2e-16 ***
## s(EngForm2)               72.914960   128 2.487e+07 0.001719 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) = 0.726  Deviance explained = 70.3%
## UBRE = 15.903  Scale est. = 1          n = 1173
plot.gam(mGAM,
  ylab="Relative Norse Freq",
  xlab="Distance between forms",
  select = 1)

```

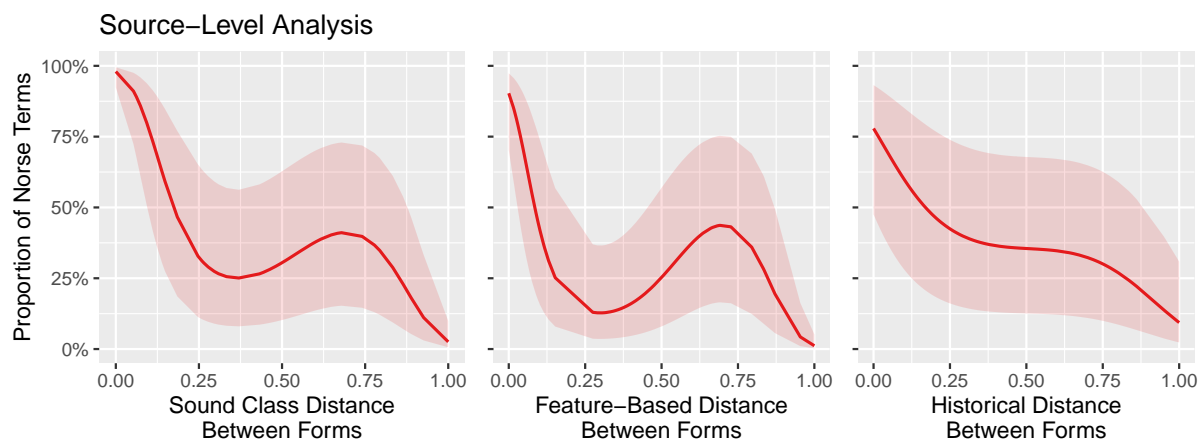


Linear: $\beta = -10.429$, $z = -7.388$, Wald $p < 0.001$, LLDiff = 101.3, $df = 1$, $p < 0.001$; Quadratic: $\beta = 19.968$, $z = 5.476$, Wald $p < 0.001$, LLDiff = 2.3, $df = 1$, $p = 0.033$; Cubic: $\beta = -13.068$, $z = -5.176$, Wald $p < 0.001$, LLDiff = 12, $df = 1$, $p < 0.001$

```
bigPlot = grid.arrange(
  pNormSimple +
    ggtitle("Source-Level Analysis"),
  pNormFeature+
    ggtitle("") +
    theme(axis.title.y=element_blank(),
          axis.text.y=element_blank()),
  pNormHist+
    ggtitle("") +
    theme(axis.title.y=element_blank(),
          axis.text.y=element_blank()),
  nrow=1,widths=c(1.3,1,1))

pdf("../results/BigEffectsPlot.pdf",width = 8,height=3)
plot(bigPlot)
dev.off()
```

```
## pdf
## 2
```



```
tSimple = read.csv("../results/SimpleRes_totalFreq.csv",
  stringsAsFactors = F)
tFeature = read.csv("../results/FeatureRes_totalFreq.csv",
  stringsAsFactors = F)
tHistorical = read.csv("../results/HistoricalRes_totalFreq.csv",
  stringsAsFactors = F)
sSimple = read.csv("../results/SimpleRes.csv",
  stringsAsFactors = F)
sFeature = read.csv("../results/FeatureRes.csv",
  stringsAsFactors = F)
sHistorical = read.csv("../results/HistoricalRes.csv",
  stringsAsFactors = F)

res = rbind(tSimple,tFeature,tHistorical)
resNames = c(lldiff="Log Likelihood Difference",
  Chisq="Chi Squared",pChi="p")
resNames2 = names(res)
resNames2[resNames2 %in% names(resNames)] =
  resNames[resNames2[resNames2 %in% names(resNames)]]

res = res[,names(res)!="X"]
res = cbind(Measure=rep(c("Sound Class","Feature","Historical"),each=3), res)
res = rbind(c("", "", "Model Comparison", "", "", "", "",
  "Model Estimate", "", ""))
```

```

        resNames2,
        res)

write.table(res,file="../results/MainResults_totalFreq.csv",
            sep = ",",col.names = F,row.names = F,fileEncoding = "UTF-8")

res2 = res
res2[3:11,] = rbind(sSimple,sFeature,sHistorical)

write.table(res2,file="../results/MainResults_sourceLevel.csv",
            sep = ",",col.names = F,row.names = F,fileEncoding = "UTF-8")

```

7 Comparison between texts

Compare sources according to the difference in proportion of Norse terms for each set.

```
g = read.xlsx("../data/SharedIntegrationOfCognatesData.xlsx",1)
g = g[g$Etymology %in% c("Norse","English"),]
# Ignore numerals in set name
g$Set = gsub("[0-9]", "", g$Set)

nCols = names(g)[which(names(g)=="No..in.Ormulum"):which(names(g)=="Rolle")]
for(col in nCols){
  g[,col] = as.numeric(g[,col])
}

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

allSets = unique(g$Set)
f = data.frame(Set = allSets)
fLog = data.frame(Set = allSets)
for(col in nCols){
  gN = g[g$Etymology == "Norse",]
  gE = g[g$Etymology == "English",]
  fNorse = tapply(gN[,col], gN$Set, sum)[allSets]
  fNorse[is.na(fNorse)] = 0
  fEng = tapply(gE[,col], gE$Set, sum)[allSets]
  fEng[is.na(fEng)] = 0
  f[,col] = fNorse / (fEng+fNorse)
  fLog[,col] = log10(1+fEng) - log10(1+fNorse)
}

nColsLabels = c("Ormulum", "FCPC", "Havelok", "Genesis & Exodus",
                 "Mannyn", "Gawain-poet", "Wars of Alexander",
                 "St Erkenwald", "Cursor Mundi", "Lincolnshire",
                 "Nottinghamshire", "Norfolk", "Rolle")

mat = matrix(NA, nrow=length(nCols),
             ncol = length(nCols))
rownames(mat) = nColsLabels
colnames(mat) = nColsLabels
matLog = matrix(NA, nrow=length(nCols),
               ncol = length(nCols))
rownames(matLog) = nColsLabels
colnames(matLog) = nColsLabels
for(i in 1:length(nCols)){
  iProp = f[,nCols[i]]
  iPropLog = fLog[,nCols[i]]
  for(j in 1:length(nCols)){
    jProp = f[,nCols[j]]
    jPropLog = fLog[,nCols[j]]

    diffs = abs(iProp-jProp)
    diffs = diffs[!is.nan(diffs)]
    diffs = diffs[!is.na(diffs)]
    mat[nColsLabels[i], nColsLabels[j]] = mean(diffs)
  }
}
```

```

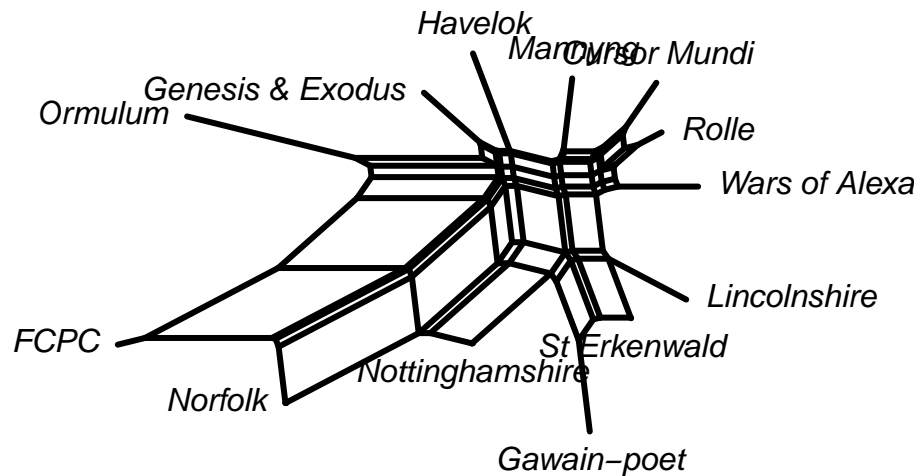
    diffsLog = abs(iPropLog - jPropLog)
    matLog[nColsLabels[i],nColsLabels[j]] = mean(diffsLog)
  }
}

```

```

phy = neighborNet(mat)
plot(phy)

```



```

pdf("../results/NeighbourNet.pdf")
plot(phy)
dev.off()

```

```

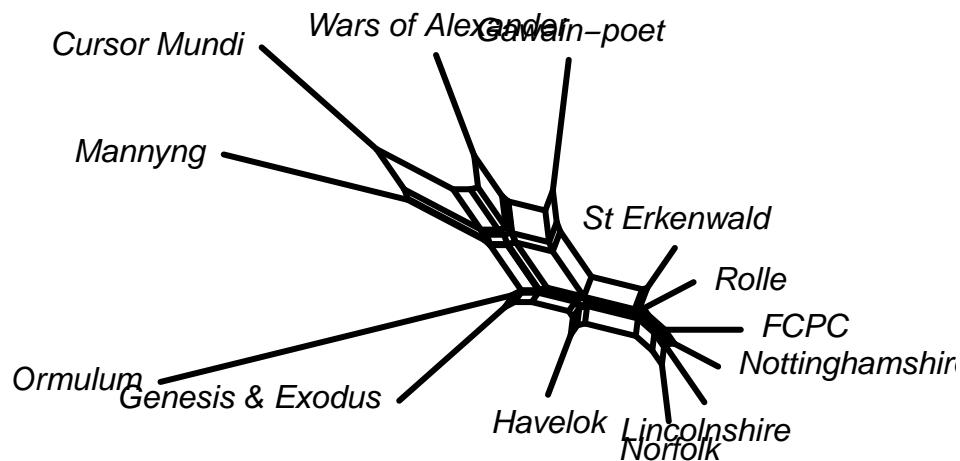
## pdf
## 2

```

```

phyLog = neighborNet(matLog)
plot(phyLog)

```



```

pdf("../results/NeighbourNet_Log.pdf")
plot(phyLog)
dev.off()

```

```

## pdf
## 2

```

8 Bayesian modelling

8.1 Simple distance

In this section, we replicate the analyses above using a Bayesian framework. As with the example of a Bayesian analysis above, we use a beta binomial framework.

```
d$NormDistanceChosen = d$NormDistance
d$totalAllFreq = d$totalNFreq + d$totalEFreq
mBSimple = brm(totalNFreq | trials(totalAllFreq) ~
  s(NormDistanceChosen) +
  (1|Set) + (1|EngLexeme),
  data = d, family = "beta_binomial",
  silent = 2, refresh=0)
```

```
## Warning: There were 8 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.

## Warning: Examine the pairs() plot to diagnose sampling problems

summary(mBSimple)
```

```
## Warning: There were 8 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help. See
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup

## Family: beta_binomial
## Links: mu = logit; phi = identity
## Formula: totalNFreq | trials(totalAllFreq) ~ s(NormDistanceChosen) + (1 | Set) + (1 | EngLexeme)
## Data: d (Number of observations: 1638)
## Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
## total post-warmup draws = 4000
##
## Smooth Terms:
##               Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS
## sds(sNormDistanceChosen_1)    1.71    0.96    0.37    4.06 1.00    1719
##               Tail_ESS
## sds(sNormDistanceChosen_1)    1803
##
## Group-Level Effects:
## ~EngLexeme (Number of levels: 135)
##               Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)    0.68    0.08    0.53    0.84 1.00    949    2335
##
## ~Set (Number of levels: 67)
##               Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)    0.35    0.14    0.06    0.62 1.01    444    443
##
## Population-Level Effects:
##               Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS
## Intercept            0.28    0.10    0.07    0.47 1.00    1960
## sNormDistanceChosen_1 -4.60    3.25   -12.09    0.69 1.00    1538
##               Tail_ESS
## Intercept            2540
## sNormDistanceChosen_1 1951
##
## Family Specific Parameters:
##               Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## phi            2.00    0.07    1.86    2.14 1.00    5604    3163
##
```

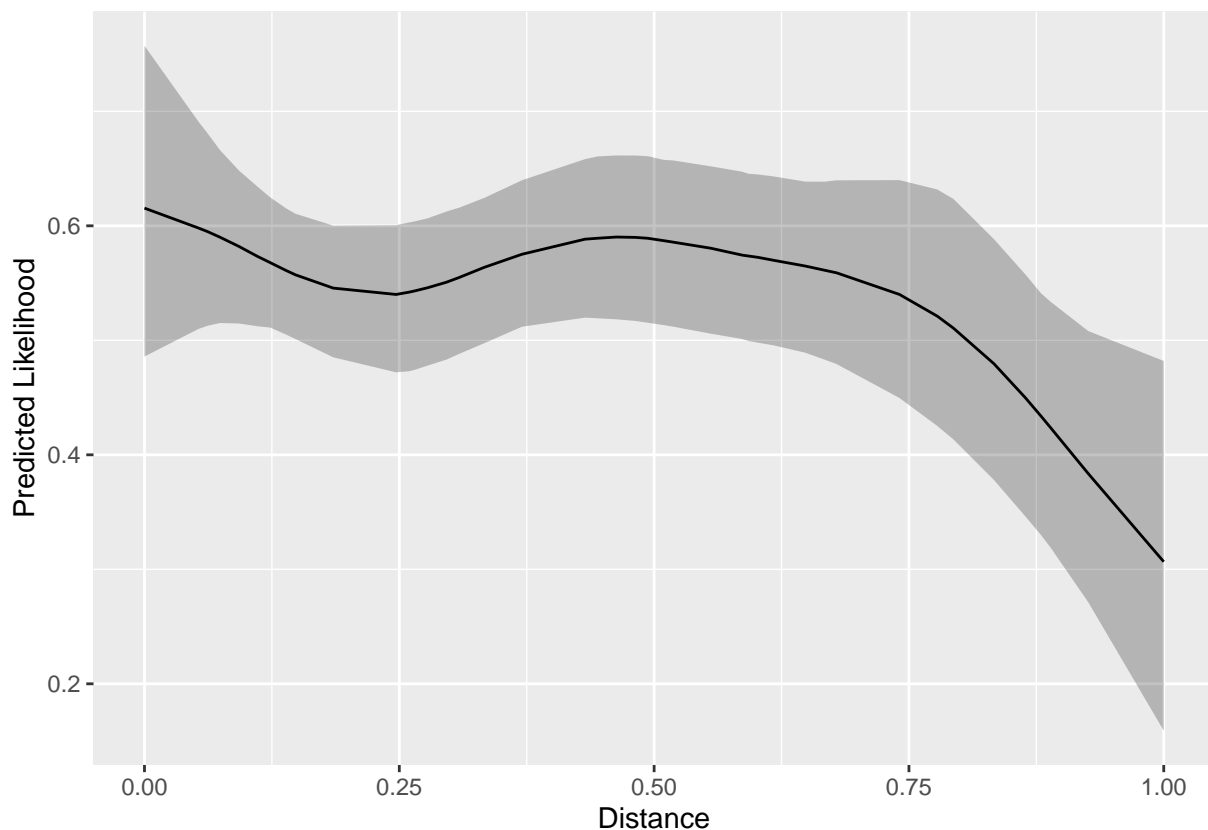
```
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).

pred = ggpredict(mBSimple,
                 terms="NormDistanceChosen")

## Note: uncertainty of error terms are not taken into account. Consider
##   setting `interval` to "prediction". This will call `posterior_predict()`
##   instead of `posterior_epred()`.

## Warning in checkMatrixPackageVersion(): Package version inconsistency detected.
## TMB was built with Matrix version 1.6.1
## Current Matrix version is 1.6.5
## Please re-install 'TMB' from source using install.packages('TMB', type = 'source') or ask CRAN for

# Predict uses 10 trials
pred$predicted = pred$predicted/10
pred$conf.low = pred$conf.low/10
pred$conf.high = pred$conf.high/10
ggplot(pred,aes(x=x,y=predicted,ymin=conf.low,ymax=conf.high)) +
  geom_ribbon(alpha=0.3)+
  geom_line() +
  xlab("Distance") +
  ylab("Predicted Likelihood")
```



In the analyses above, we set the number of knots at 4. We can test this setting by running two different models with different number of knots:

```
mBSimple.k4 = brm(totalNFreq | trials(totalAllFreq) ~
  s(NormDistanceChosen,k=4) +
  (1|Set) + (1|EngLexeme),
  data = d, family = "beta_binomial",
  silent = 2,refresh=0)
```



```
## recompiling to avoid crashing R session

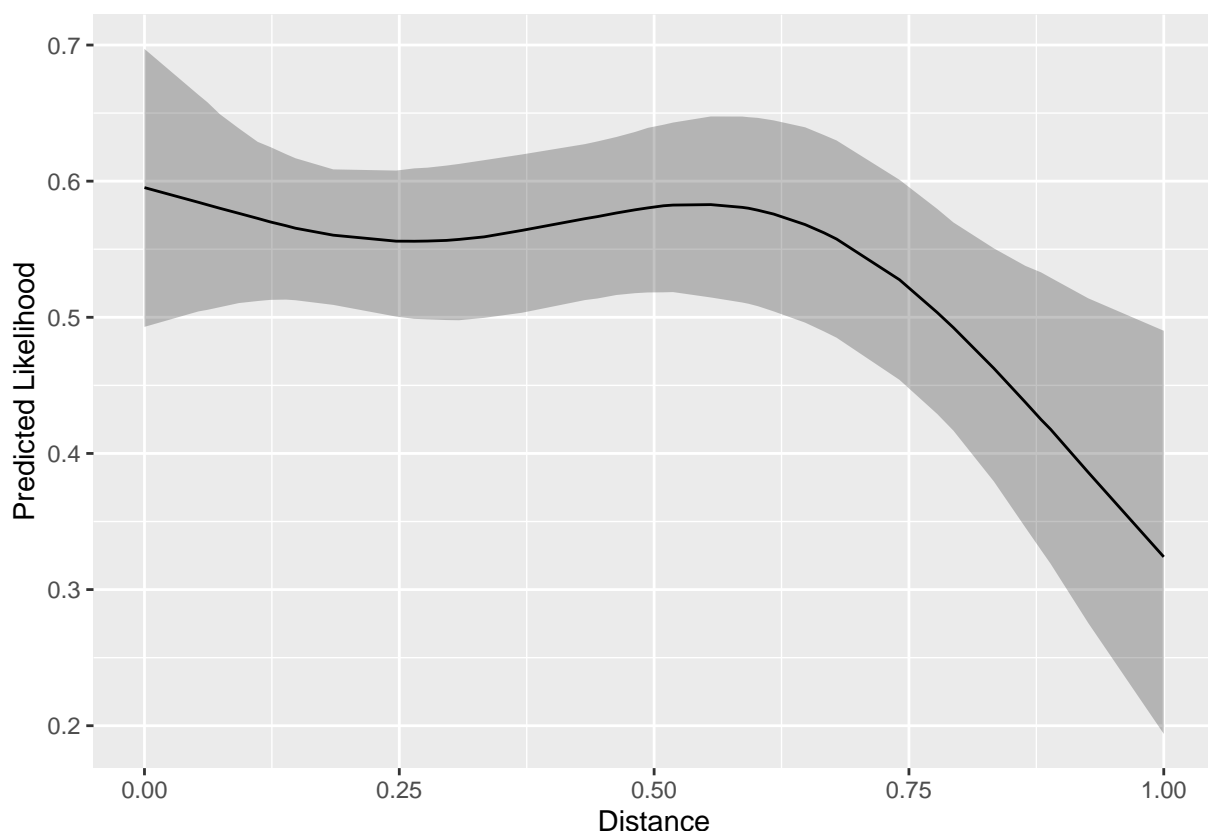
## Warning: There were 9 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.

## Warning: Examine the pairs() plot to diagnose sampling problems

pred = ggpredict(mBSimple.k4,
                 terms="NormDistanceChosen")

## Note: uncertainty of error terms are not taken into account. Consider
## setting `interval` to "prediction". This will call `posterior_predict()`
## instead of `posterior_epred()`.

# Predict uses 10 trials
pred$predicted = pred$predicted/10
pred$conf.low = pred$conf.low/10
pred$conf.high = pred$conf.high/10
ggplot(pred, aes(x=x, y=predicted, ymin=conf.low, ymax=conf.high)) +
  geom_ribbon(alpha=0.3) +
  geom_line() +
  xlab("Distance") +
  ylab("Predicted Likelihood")
```



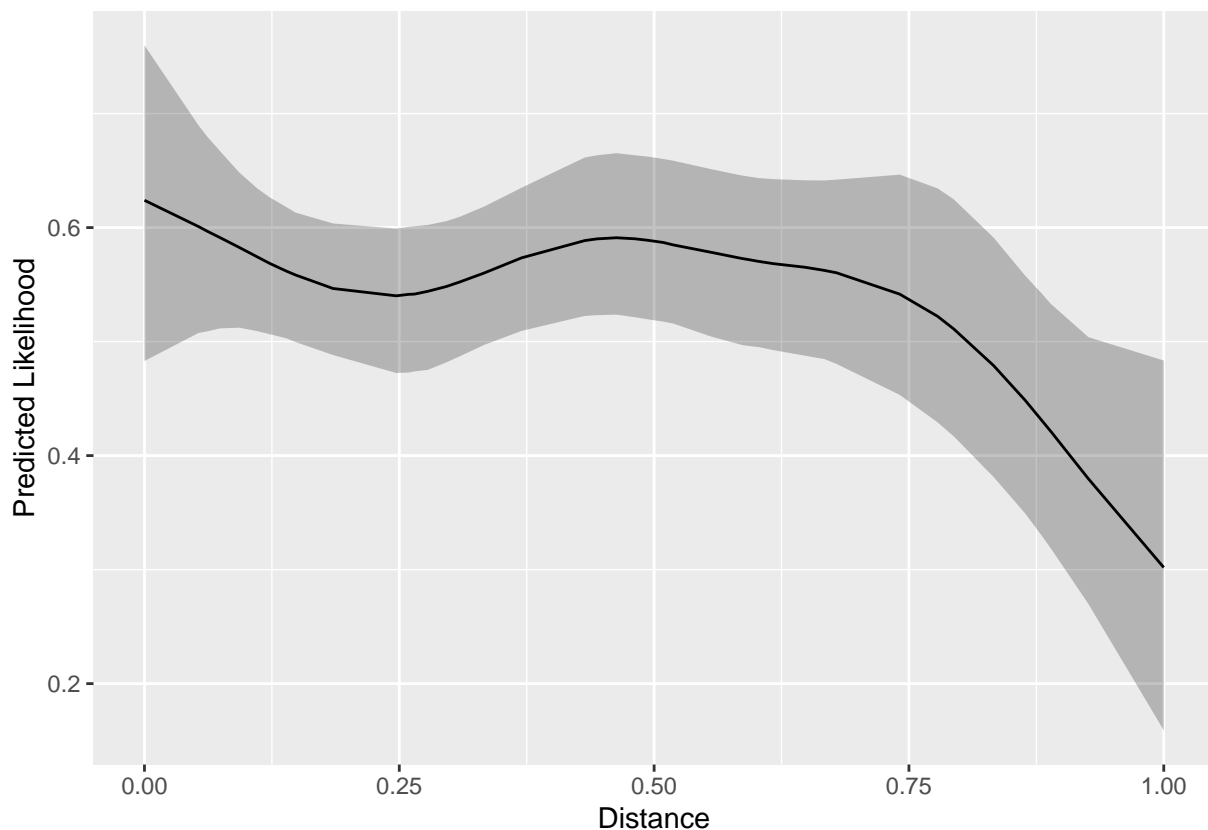
```
mBSimple.k8 = brm(totalNFreq | trials(totalAllFreq) ~
  s(NormDistanceChosen, k=8) +
  (1|Set) + (1|EngLexeme),
  data = d, family = "beta_binomial",
  silent = 2, refresh=0)
```

```
## recompiling to avoid crashing R session

## Warning: There were 19 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

```
## to find out why this is a problem and how to eliminate them.
## Warning: Examine the pairs() plot to diagnose sampling problems
pred = ggpredict(mBSimple.k8,
                 terms="NormDistanceChosen[all]")

## Note: uncertainty of error terms are not taken into account. Consider
## setting `interval` to "prediction". This will call `posterior_predict()`
## instead of `posterior_epred()`.
# Predict uses 10 trials
pred$predicted = pred$predicted/10
pred$conf.low = pred$conf.low/10
pred$conf.high = pred$conf.high/10
ggplot(pred, aes(x=x, y=predicted, ymin=conf.low, ymax=conf.high)) +
  geom_ribbon(alpha=0.3) +
  geom_line() +
  xlab("Distance") +
  ylab("Predicted Likelihood")
```



There is very little difference in the overall shape of the relationship between $k=4$ and $k=8$. At least, there are no conclusions we would draw from the $k=8$ fit that we would not draw from the $k=4$ fit. We decided to stick with the simpler model ($k=4$).

Our conclusion was, as stated in the discussion: “we argue that the overall picture is that the effect exists for extreme ends of form distance”. That is, there may be some non-linear effect in the middle, but the basic picture is that low distance = likely to be Norse, high distance = likely to be Native. All of the analyses fit this general description.

8.2 Feature distance

```
d$NormDistanceChosen = d$NormFeatureDistance
d$totalAllFreq = d$totalNFreq + d$totalEFreq
```

```

mBSimple = brm(totalNFreq | trials(totalAllFreq) ~
  s(NormDistanceChosen) +
  (1|Set) + (1|EngLexeme),
  data = d, family = "beta_binomial",
  silent = 2, refresh=0)

## recompiling to avoid crashing R session

## Warning: There were 36 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.

## Warning: There were 1 transitions after warmup that exceeded the maximum treedepth. Increase max_
## https://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded

## Warning: Examine the pairs() plot to diagnose sampling problems

## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess

## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior variances and tail qu
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#tail-ess

summary(mBSimple)

## Warning: There were 36 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help. See
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup

## Family: beta_binomial
## Links: mu = logit; phi = identity
## Formula: totalNFreq | trials(totalAllFreq) ~ s(NormDistanceChosen) + (1 | Set) + (1 | EngLexeme)
## Data: d (Number of observations: 1638)
## Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
## total post-warmup draws = 4000
##
## Smooth Terms:
##               Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS
## sds(sNormDistanceChosen_1)    2.05    1.33    0.41    5.68 1.02    306
##               Tail_ESS
## sds(sNormDistanceChosen_1)    125
##
## Group-Level Effects:
## ~EngLexeme (Number of levels: 135)
##               Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)    0.68    0.08    0.53    0.86 1.00    574    1393
##
## ~Set (Number of levels: 67)
##               Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)    0.37    0.13    0.09    0.63 1.02    342    527
##
## Population-Level Effects:
##               Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS
## Intercept            0.27    0.11    0.06    0.47 1.01    1027
## sNormDistanceChosen_1 -5.80    3.44   -13.45   -0.27 1.00    753
##               Tail_ESS
## Intercept            1396
## sNormDistanceChosen_1 1495
##
## Family Specific Parameters:

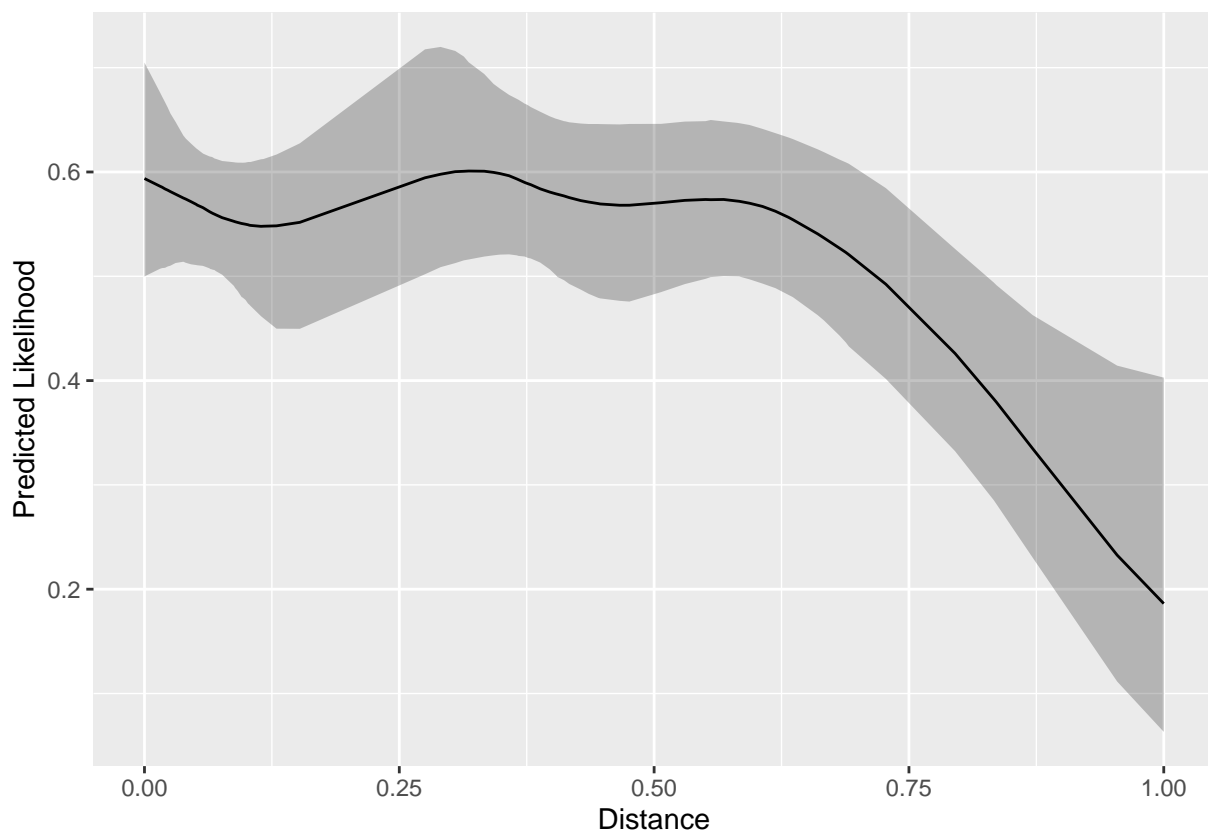
```

```
##      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## phi      2.01      0.08      1.86      2.16 1.00      3227      2847
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).

pred = ggpredict(mBSimple,
                 terms="NormDistanceChosen")

## Note: uncertainty of error terms are not taken into account. Consider
## setting `interval` to "prediction". This will call `posterior_predict()`
## instead of `posterior_epred()`.

# Predict uses 10 trials
pred$predicted = pred$predicted/10
pred$conf.low = pred$conf.low/10
pred$conf.high = pred$conf.high/10
ggplot(pred, aes(x=x, y=predicted, ymin=conf.low, ymax=conf.high)) +
  geom_ribbon(alpha=0.3) +
  geom_line() +
  xlab("Distance") +
  ylab("Predicted Likelihood")
```



8.3 Historical distance

```
d$NormDistanceChosen = d$NormHistoricalDistance
d$totalAllFreq = d$totalNFreq + d$totalEFreq
mBSimple = brm(totalNFreq | trials(totalAllFreq) ~
               s(NormDistanceChosen) +
               (1|Set) + (1|EngLexeme),
               data = d, family = "beta_binomial",
```

```

silent = 2,refresh=0)

## recompiling to avoid crashing R session

## Warning: There were 8 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.

## Warning: Examine the pairs() plot to diagnose sampling problems

summary(mBSimple)

## Warning: There were 8 divergent transitions after warmup. Increasing
## adapt_delta above 0.8 may help. See
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup

## Family: beta_binomial
## Links: mu = logit; phi = identity
## Formula: totalNFreq | trials(totalAllFreq) ~ s(NormDistanceChosen) + (1 | Set) + (1 | EngLexeme)
## Data: d (Number of observations: 1638)
## Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
## total post-warmup draws = 4000
##
## Smooth Terms:
##               Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS
## sds(sNormDistanceChosen_1)    1.57    0.79    0.42    3.44 1.00    2417
##               Tail_ESS
## sds(sNormDistanceChosen_1)    2223
##
## Group-Level Effects:
## ~EngLexeme (Number of levels: 135)
##               Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)    0.67    0.08    0.52    0.86 1.00    1099    2118
##
## ~Set (Number of levels: 67)
##               Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)    0.36    0.14    0.06    0.63 1.01    571    699
##
## Population-Level Effects:
##               Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS
## Intercept            0.26    0.10    0.07    0.46 1.00    3515
## sNormDistanceChosen_1 -3.04    2.57   -8.44    1.68 1.00    3620
##               Tail_ESS
## Intercept            2932
## sNormDistanceChosen_1 2862
##
## Family Specific Parameters:
##               Estimate Est.Error 1-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## phi            2.00    0.08    1.86    2.15 1.00    7035    2767
##
## Draws were sampled using sampling(NUTS). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).

pred = ggpredict(mBSimple,
  terms="NormDistanceChosen")

## Note: uncertainty of error terms are not taken into account. Consider
## setting `interval` to "prediction". This will call `posterior_predict()`
## instead of `posterior_epred()`.

```

```

# Predict uses 10 trials
pred$predicted = pred$predicted/10
pred$conf.low = pred$conf.low/10
pred$conf.high = pred$conf.high/10
ggplot(pred,aes(x=x,y=predicted,ymin=conf.low,ymax=conf.high)) +
  geom_ribbon(alpha=0.3)+
  geom_line() +
  xlab("Distance") +
  ylab("Predicted Likelihood")

```

