rijksuniversiteit
groningen

faculteit Wiskunde en
Natuurwetenschappen

Social, geographical, and lexical influences on dialectical pronunciations in Dutch

Re-analysis of Wieling, Nerbonne, Baayen (2011)

Bacheloronderzoek Wiskunde

Juli 2013

Student: V.B. Ko

Eerste begeleider wiskunde: E.C. Wit

Tweede begeleider wiskunde : W.P. Krijnen

Begeleider informatiekunde: J. Nerbonne

Contents

Chapter 1

Introduction

The Netherlands is a small country with 41,543 km^2 of area and 16,788,973 of population (2013) [1]. In spite of its small size, the country counts two official languages and numerous dialects.

Martijn Wieling, John Nerbonne, and Harald Baayen performed a quantitative social dialectological study explaining social, lexical and geographical effect on linguistic variation of different Dutch pronunciations [2]. This was the first study of this kind on a large dataset of dialect pronunciations. They combined generalized additive modeling with mixed-effect regression models to analyze the data. Their dataset contained pronunciation of 559 words that are collected from 424 locations in the Netherlands (The complete list can be found in appendix). This pronunciation dataset is captured from Goeman-Taeldeman-Van Reenen-Project between 1980 and 1995 [3].

By using PMI-based Levenshtein distance [4], they measured the distance between the dialectal pronunciation and the standard pronunciation (based on [5]) of these 424 Dutch words. For example, the phonetic distance between two variants of Dutch word 'binden' [bɪndən] and [bɛində] is:

bɪndən	insert ɛ	0.034
bɛɪndən	subst. i/ɪ	0.020
bɛindən	delete n	0.024
bɛində		
		0.078

Further, social data of speakers such as gender and age were imported from [3]. Additional demographic information of 424 locations in year 1995 such as average age and average income were collected from Statistics Netherlands (Dutch: CBS).

The conclusion of their study was: the phonetic distance from standard Dutch gets greater if: location has smaller population, location has higher average age, the word is noun, the word has higher frequently of using, and the word has relatively many vowels.

When they performed this study in programming language `R` in 2011, they couldn't use `gamm` in package `gamm4`, which is the most reliable tool for this kind of model, because of the big size of the data. What they did instead was first using `gam` to analyze geographical effect and then using `lmer` to analyze other variables. At this moment (2013), there is a new alternative of `gamm` called `bam`. The aim of this paper is to apply this new method to [2] and also looking for possible improvements in their methodology.

Chapter 2

Theory

2.1 Linear Models

2.1.1 Model form

Let's begin with the simplest form of data. In `R` there is a built-in dataset called `women`. This dataset contains average heights and weights of some American women.

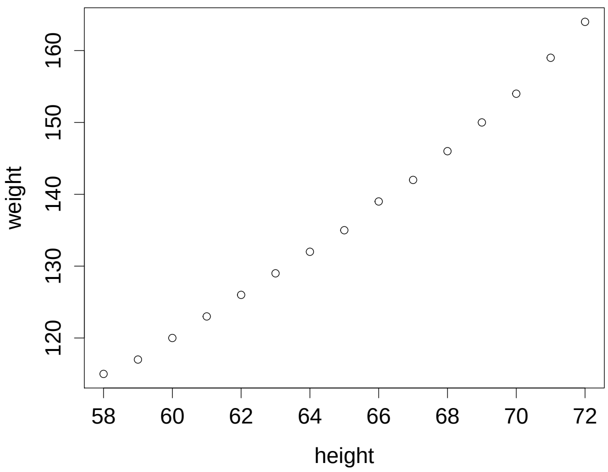

Figure 2.1: Scatter plot of dataset `women`

From figure 2.1 we can easily identify a linear relationship between weight(y) and height(x). **Linear models** can be used to represent this kind of dependency. The linear model has a form

$$y = \mathbf{x}^\top \boldsymbol{\beta} + \epsilon = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon$$

with

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_p \end{bmatrix}, \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix},$$

where ϵ representing errors in the model. It is assumed that

$$[\epsilon_1, \cdots, \epsilon_n] \overset{i.i.d}{\sim} N(0, \sigma^2)$$

where σ has to be estimated.

Now, we expand this model expression to a matrix form for a 'group' of these random variables.

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{2.1}$$

where

$$\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix}, \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}, \quad \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{bmatrix}.$$

$\mathbf{X}$ is called **model matrix** or **design matrix**.

2.1.2 Estimation

Now, we need to know how to estimate this model. Estimating means in this case finding $\widehat{\boldsymbol{\beta}}$, the estimator of $\boldsymbol{\beta}$. **Maximum likelihood estimation** is one of the methods that can be used for finding this estimator. According to section 5.4.1 of [6], $\widehat{\boldsymbol{\beta}}$ can be written as

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y} \tag{2.2}$$

Let's go back to our example of `women`. In `R`, we can fit the linear model with a built-in function `lm`. Figure 2.2 shows the fitted line of this model.

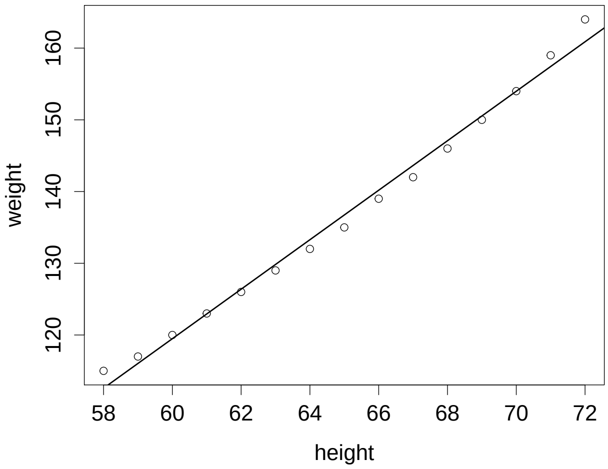

Figure 2.2: fitted linear model for `women` data

2.2 Generalized Linear Models

2.2.1 Model form

In the previous section, we discussed about linear models. However, if data don't show linearity, this model hardly can be used. Generalized linear model generalizes and expands the linear model.

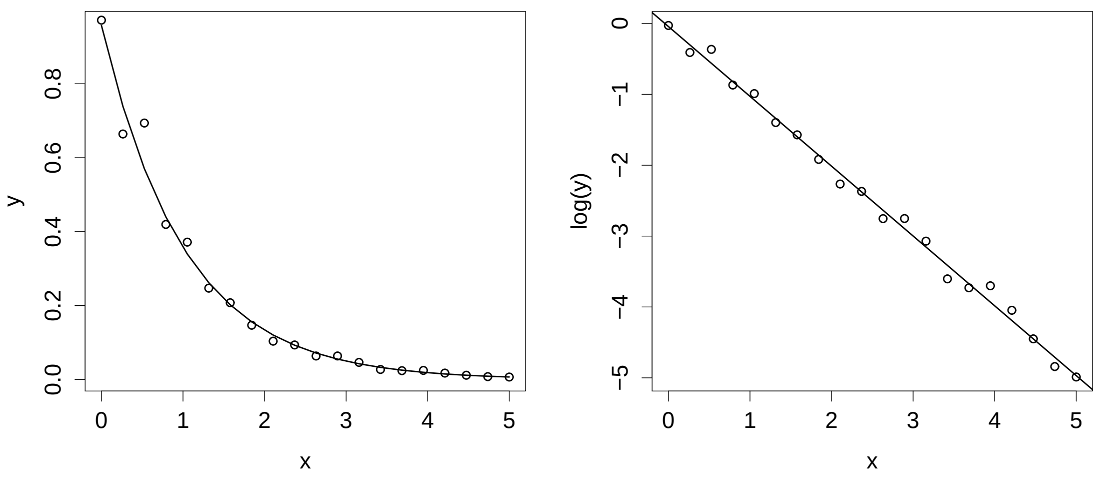

Figure 2.3: y and x have exponential relationship. By taking $\log(y)$, we can "link" y to the linear model.

Left side of figure 2.3 shows dataset that showing the exponential distribution. Since the exponential distribution has a banana-like curve, trying to fit a straight line to the data is not a good idea. Recall that the exponential distribution has a form $f(x) = \lambda e^{-\lambda x}$, we can easily infer that $\log(f(x))$ and x would have a linear relationship. This means that we can "link" our y to the linear model by using $\log(y)$. We generalize this concept and call it "link function". This leads us to the **generalized linear model (GLM)**. GLM has a form

$$g(\mu_i) = \mathbf{x}_i^\top \boldsymbol{\beta},\ \mu_i = E(y_i)$$

where g is a monotonic and two times derivative link function. Further, y should be a member of the exponential family. This means that the probability distribution of y should have a form

$$f(y;\theta) = \exp[a(y)b(\theta) + c(\theta) + d(y)].$$

2.2.2 Example: beetles

In 1935, C.I. Bliss performed an experiment on beetles. He exposed the beetles into different dosages of gaseous carbon disulphide for five hours (Bliss, 1935). The result of the experiment is as follows.

Dose	Exposed	Mortality
1.69	59	6
1.72	60	13
1.76	62	18
1.78	56	28
1.81	63	52
1.84	59	53
1.86	62	61
1.88	60	60

For an individual beetle, there are only two possibilities (death or alive). This can be modeled as a binomial distribution, which is a member of the exponential family. By choosing logit function as our link function, we can use GLM. We want to describe the proportion of successes(=death) with this model. Let Y be the total number of successes, then $E[Y] = n\pi$. So, $E[P] = \pi$. We take this π as our dependent variable.

$$g(\pi_i) = \log(\frac{\pi_i}{1-\pi_i}) = \mathbf{x}_i^\top \boldsymbol{\beta}$$

We implement this in `R` by using built-in function `glm`. We get the following outcome in `R`.

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -60.717      5.181  -11.72   <2e-16 ***
dose          34.270      2.912   11.77   <2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

The `summary` result shows that `dose` explains our data fairly well. This is done by a hypothesis testing. If our estimator($=\widehat{\beta}_1$) doesn't have any effect on the data, it should be $\widehat{\beta}_1 = 0$. We take this assumption as our null hypothesis.

$$H_0 : \beta_1 = 0 \text{ versus } H_a : \beta_1 \neq 0$$

If the size of sample is large enough, maximum likelihood estimators ($\widehat{\beta}_1$ in our case) will follow normal distribution. Consequently, when they are standardized and squared, they should follow the chi-squared distribution with a proper degrees of freedom. (In this case, it's trivial so see that $df = 1$.) `Pr(>|z|)` shows the probability that a new random value on the chi-squared distribution is as extreme or more extreme than the critical value. This probability is significantly small ($< 2 \cdot 10^{-16}$), so we reject our null hypothesis and conclude that `dose` does have an effect in our explanatory variable.
Furthermore, we can read from `summary` that $\widehat{\beta}_0 = -60.717$ and $\widehat{\beta}_1 = 34.27$, which gives

$$g(\pi_i) = \log(\frac{\pi_i}{1-\pi_i}) = \mathbf{x}_i^\top \boldsymbol{\beta} = -60.717 + 34.27x$$
$$\widehat{\pi} = \frac{e^{-60.717+34.27x}}{1+e^{-60.717+34.27x}}$$

2.3 Generalized Additive Models

2.3.1 Model form

Recall that the linear model has the form

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon$$

To model non-linear data, we keep this structure, but we replace the linear term $\beta_i x_i$ by a more general functional form $f_i(x_i)$ and replace β_0 by α. $f_i(x_i)$ is called "smooth function".

$$\mu = \alpha + f_1(x_1) + f_2(x_2) + \cdots + f_p(x_p) + \epsilon$$

Like we did to obtain generalized linear model from simple linear model, we use a link function to link μ to the model form. This leads us to the final form of the **Generalized Additive Model (GAM)**.

$$g(\mu_i) = \alpha + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_{ip}(x_{ip}) + \epsilon_i \tag{2.3}$$

2.3.2 Smooth function

The question is, of course, what will be our smooth function $f(\cdot)$. We will discuss some sensible choices of $f(\cdot)$.

2.3.2.1 Cubic Splines

Consider a graph of a cubic function. It has all four possible forms of convex curves, which makes it very suitable being a smooth function. We will chop the domain of our graph into some pieces and will try to fit one cubic function to each "chopped" piece of domain. The joining points between these pieces are called **knots**. We denote these knots by $\{x_i^*, i : 1, \cdots, q-2\}$ where q is the total number of pieces +2. (+2 are for the starting point and the ending point of the domain.)
According to [7], cubic splines smooth function has a form

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \epsilon,$$

where the i^{th} row of $\mathbf{X}$ is

$$\mathbf{X}_i = [1, x_i, R(x_i, x_1^*), R(x_i, x_2^*), \cdots, R(x_i, x_{q-2}^*)]$$

where R is defined as

$$R(x,z) = [(z-1/2)^2 - 1/12][(x-1/2)^2 - 1/12]/4$$
$$-[(|x-z| - 1/2)^4 - 1/2(|x-z| - 1/2)^2 + 7/240]/24$$

What we have to do now is deciding the degree of smoothing (q). In most cases, the knots will be evenly spaced through the domain. How finely the domain should be chopped? Chopping the domain into too big pieces (too low q) will make our smooth function very stiff (less fitting). Too large q will make our smooth function too flexible and the smooth function will even fit the noises in the data (overfitting).

2.3.2.2 Cubic Splines: degree of smoothing

We will illustrate the cubic splines by using a subset of 25 rows from the built-in data `cars` in `R`. Figure 2.4 shows the result of this implementation. The choice of 5 knots seems okay. Choosing 2 knots shows less fitting and choosing 10 knots shows overfitting with a very noisy curve.

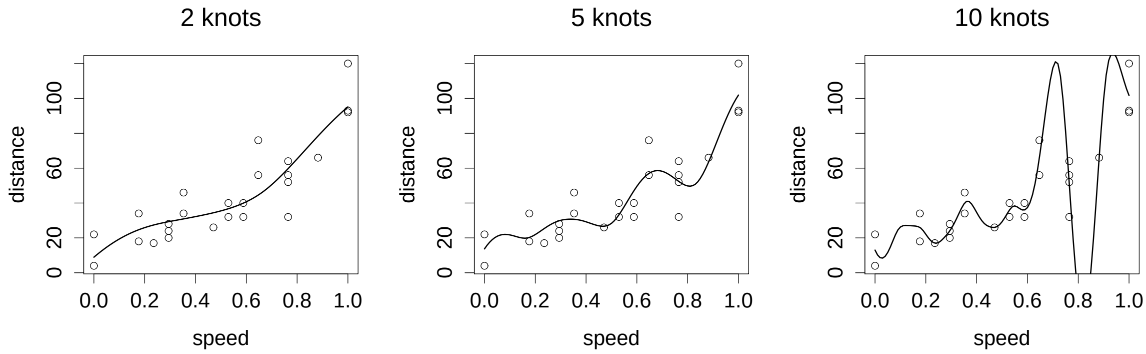

Figure 2.4: cubic splines with different number of knots

2.3.2.3 Cubic Splines: penalized regression splines with λ

As we saw in the last section, we can adjust the fitness of our model by manipulating the number of knots. Backward selection is one of the popular options of doing this. In this method, we start with a very fine grid of knots and then we sequentially drop unnecessary knots. Yet, this method can generate unevenly divided knots, which will possibly cause bad model performance.

As alternative, [8] introduces a “wiggliness”penalty. We set the total number of knots (basis dimension) slightly larger than the optimal value of q that we believe. This makes the basis flexible enough to represent $f(x)$. Now, instead of ordinarily minimizing sum of squares

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2$$

we minimize the sum of squares plus a “wiggliness”penalty

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \int_0^1 [f''(x)]^2 \, dx.$$

We call λ **smoothing parameter** and we manipulate this value to control the “wiggliness”penalty of our model. When $\lambda = 0$, so no penalty at all, our model will be at its most sinuous form. When $\lambda \to \infty$, the model is maximally penalized and it has a straight line. It can be shown from [8] that

$$\int_0^1 [f''(x)]^2 = \boldsymbol{\beta}^\top \mathbf{S}\boldsymbol{\beta}$$

where $\mathbf{S}$ is a matrix such that $S_{i+2,\ j+2} = R(x_i^*, x_j^*)$ and the first two rows and columns are 0.

In conclusion, to fit our model, we have to minimize

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\boldsymbol{\beta}^\top\mathbf{S}\boldsymbol{\beta} \tag{2.4}$$

with respect to $\boldsymbol{\beta}$.

For numerical stability and performance, when we implement this least squares estimate in `R`, the following equality will be used.

$$\left\| \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \mathbf{X} \\ \sqrt{\lambda}\mathbf{B} \end{bmatrix} \boldsymbol{\beta} \right\|^2 = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \boldsymbol{\beta}^\top\mathbf{S}\boldsymbol{\beta}$$

where $\mathbf{B}$ is square root of $\mathbf{S}$ such that $\mathbf{B}^\top\mathbf{B} = S$ [8].

Assuming that λ is given, our least squares estimate $\widehat{\boldsymbol{\beta}}$ for (2.4) is

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{S})^{-1}\mathbf{X}^\top\mathbf{y}$$

This also leads us to the expression of the hat matrix $\mathbf{A}$ which satisfies $\widehat{\boldsymbol{\mu}} = \mathbf{A}\mathbf{y}$.

$$\mathbf{A} = \mathbf{X}(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{S})^{-1}\mathbf{X}^\top$$

Let's consider again the example of `cars`. This time, we fix $q = 10$ and manipulate λ. Figure 2.5 shows the result of the implementation with some values of λ. As expected, too low λ causes overfitting and too large λ causes less fitting.

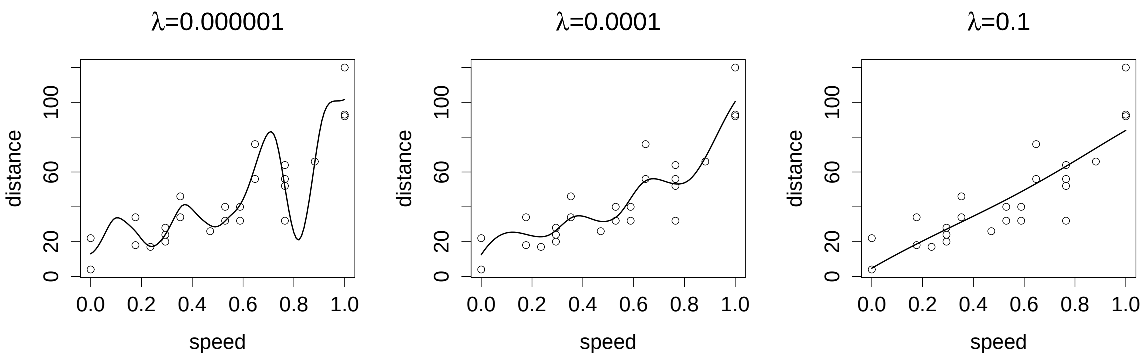

Figure 2.5: effect of smoothing parameter λ on fitted curves

2.3.2.4 Cubic Splines: choosing λ by cross validation

In the previous section, we introduced the concept of smoothing parameter λ and how it affects the shape of the fitted curve. In this section, it will be explained how we find the optimal value of λ. We want to choose λ such that our estimate $\widehat{f}$ is as close as possible to the true function f. This is same as minimizing the average square difference between $\widehat{f}$ and f. We do this by minimizing M.

$$M = \frac{1}{n}\sum_{i=1}^{n}(\widehat{f}_i - f_i)^2$$

where $f_i = f(x_i)$. Yet, in most cases, we don't know what the true function f is. So, we use $E[M] + \sigma^2$ instead, where σ^2 is the variance of the error term ϵ.
To prevent the possible overfitting, we use cross validation. That is, before we fit our model, we omit a ith point in our data, let's call this point y_i. Then, we measure the distance between y_i and our fitted model $\widehat{f}^{[-i]}$ ($\widehat{f}^{[-i]}$ indicates the model that created without y_i). This tells us how good our fitted model performs when a new datum is given. We repeat this sequentially for all points in the data and take average. This leads us to the **ordinary cross validation score** ν_o.

$$\nu_o = \frac{1}{n}\sum_{i=1}^{n}(\widehat{f}^{[-i]} - y_i)^2$$

We rewrite this by substituting $y_i = f_i + \epsilon_i$.

$$\begin{aligned}
\nu_o &= \frac{1}{n}\sum_{i=1}^{n}(\widehat{f}^{[-i]} - f_i - \epsilon_i)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}(\widehat{f}^{[-i]} - f_i)^2 - (\widehat{f}^{[-i]} - f_i)\epsilon_i + \epsilon_i^2
\end{aligned}$$

Take expectation.

$$\begin{aligned}
E[\nu_o] &= E\left[\frac{1}{n}\sum_{i=1}^{n}(\widehat{f}^{[-i]} - f_i)^2\right] - E[(\widehat{f}^{[-i]} - f_i)] \cdot E[\epsilon_i] + E[\epsilon_i^2] \\
&= E\left[\frac{1}{n}\sum_{i=1}^{n}(\widehat{f}^{[-i]} - f_i)^2\right] + \sigma^2
\end{aligned}$$

When our data is large enough, $\widehat{f}^{[-i]} \approx \widehat{f}$, which implies $E[\nu_o] \approx E[M] + \sigma^2$. So, we can minimize ν_o instead of M. However, when our data is large, ν_o requires serious amount of calculation because we have to repeat a new model fitting everytime we drop a point in our data. According to [8], ν_o can also be written as

$$\nu_o = \frac{1}{n}\sum_{i=1}^{n}\frac{(y_i - \widehat{f}_i)^2}{(1 - A_{ii})^2}$$

where A is the hat matrix discussed in p.10 and $\widehat{f}_i$ is the fitted model without omitting a point. For numerical stability and performance, we replace $|1 - A_{11}| \cdots |1 - A_{nn}|$ by its mean $tr(\mathbf{I} - \mathbf{A})/n$. This leads us to the **generalized cross validation score (GCV)**

$$\nu_g = \frac{n \sum_{i=1}^{n} (y_i - \widehat{f}_i)^2}{[tr(\mathbf{I} - \mathbf{A})]^2}.$$

Figure 2.6 shows the result of applying the GCV score method with the `cars` example. We computed ν_g for each $\lambda = 1.5^{i-1} \cdot 10^{-7}$ where $1 \leq i \leq 50$. As you can see in the left graph, ν_g obtains its minimum at $i = 18$. So, our optimal value is $\lambda = 1.5^{17} \cdot 10^{-7}$. Right side of the graph shows the model fitted with this value of λ.

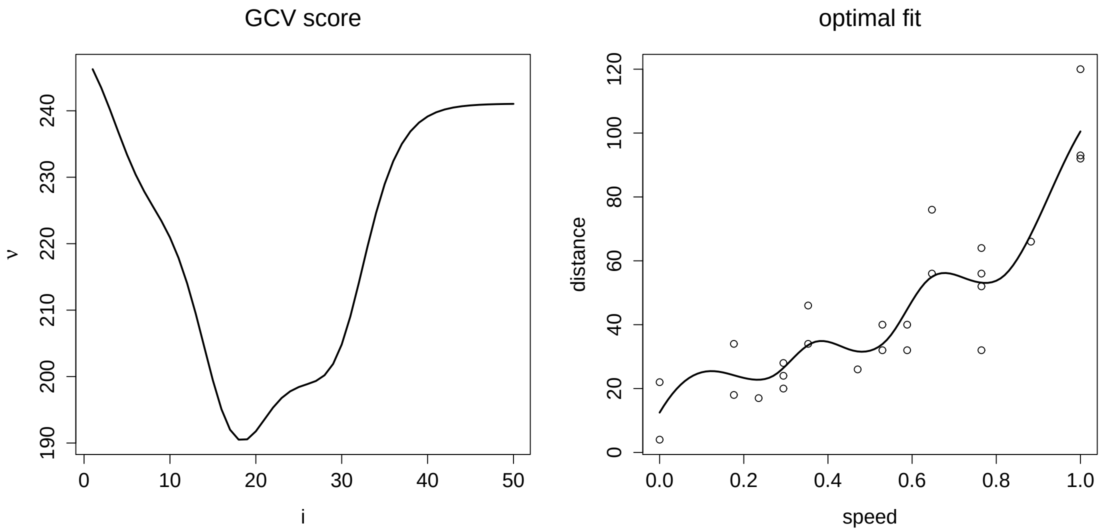

Figure 2.6: determining the optimal value of λ by GCV score (left) and the optimal fitting curve by using this value of λ. (right)

2.3.3 Thin plate splines

As we saw in the previous section, cubic spline is a useful tool. Yet, it has a drawback that we have to choose the number of knots beforehand. **Thin plate splines** don't require this process and it also can be used to estimate a smooth function of multiple independent variables. As a lot of things in thin plate splines smooth function are same as in cubic splines smooth function, only the difference will be explained in this section. More details about thin plate splines can be found in [8].

Consider a situation that we estimate the smooth function $f(\mathbf{x})$ for $(y_i, \mathbf{x})$ in n observations

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \quad \text{where} \quad \mathbf{x}_i = [x_{i1}, x_{i2}, \cdots, x_{it}]^\top.$$

Like in cubic splines, we add the "wiggliness" penalty to the sum of least square. So, estimating smooth function f becomes finding the estimate $\hat{f}$ which minimizes

$$\|\mathbf{y} - \mathbf{f}\|^2 + \lambda J_{md}(f) \tag{2.5}$$

where $\mathbf{y} = [y_1, y_2, \cdots, y_n]^\top$ and $\mathbf{f} = [f(\mathbf{x}_1), f(\mathbf{x}_2), \cdots, f(\mathbf{x}_n)]^\top$ and the wiggliness penalty

$$J_{md}(f) = \int \cdots \int \sum_{\nu_1 + \cdots + \nu_d = m} \frac{m!}{\nu_1 \cdots \nu_d!} \left(\frac{\partial^m f}{\partial x_1^{\nu_1} \cdots \partial x_d^{\nu_d}} \right)^2 dx_1 \cdots dx_d.$$

Further, it should hold that $2m > d + 1$.
$\hat{f}$ that minimizing (2.5) can be derived after some algebraic work.

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{n} \delta_i \eta_{md}(\|\mathbf{x} - \mathbf{x}_i\|) + \sum_{j=1}^{M} \alpha_j \phi_j(\mathbf{x})$$

where

$$\eta_{md}(r) = \begin{cases} \frac{(-1)^{m+1+d/2}}{2^{2m-1}\pi^{d/2}(m-1)!(mid/2)!} r^{2m-d} \log r & \text{if } d \text{ is even} \\ \frac{\Gamma(d/2-m)}{2^{2m}\pi^{d/2}(m-1)!} r^{2m-d} & \text{if } d \text{ is odd} \end{cases}$$

and $\boldsymbol{\delta}$ and $\boldsymbol{\alpha}$ are vectors of coefficients to be estimated with restriction that $\mathbf{T}^\top\boldsymbol{\delta}=0$ where $T_{ij} = \phi_j(\mathbf{x}_i)$.
Finally, fitting the thin plate smooth function, becomes minimizing

$$\|\mathbf{y} - \mathbf{E}\boldsymbol{\delta} - \mathbf{T}\boldsymbol{\alpha}\|^2 + \lambda\boldsymbol{\delta}^\top\mathbf{E}\boldsymbol{\delta} \tag{2.6}$$

w.r.t. $\boldsymbol{\delta}$ and $\boldsymbol{\alpha}$.

2.4 Mixed models

In the simple linear model (2.1), $\boldsymbol{\epsilon}$ express the error term. Mixed model can be seen as calculating this error term per variable value or category. Two new concepts are introduced in mixed model: fixed effect and random effect. An example: table 2.1 shows weight of 12 rabbits (4 rabbits per breed) after they are fed by different amount of food. It is obvious to see that breed 2 is lighter than other breeds. When we try to analyze the relationship between weight and feed, the lightness of breed 2 works as noise. This kind of undesired randomness in our data that interferes our analysis is called random effect. Fixed effect is the effect that we want to measure apart from this random effect. In this example, 'feed' is the fixed effect.

rabbit	breed	weight	feed
1	1	1470	70
2	1	1720	90
3	1	1930	110
4	1	2150	130
5	2	1080	70
6	2	1130	90
7	2	1320	110
8	2	1450	130
9	3	1490	70
10	3	1710	90
11	3	2010	110
12	3	2210	130

Table 2.1: Weight of rabbits after they are fed by different amount of food

In mixed model, we calculate the average of the random effect at each level and add this average to the fixed effect as independent residual error term to "average out" the random effect. Imagine that we want to fit the simple linear model into the data from table 2.1. The simple linear model has the form:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \epsilon, \quad \epsilon \overset{i.i.d}{\sim} N(0, \sigma^2\mathbf{I})$$

By adding random effect terms, we obtain linear mixed models.

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{b} + \epsilon, \quad \mathbf{b} \overset{i.i.d}{\sim} N(0, \boldsymbol{\psi}) \tag{2.7}$$

where $\mathbf{b}$ is a vector that contains coefficients of random effects, $\boldsymbol{\psi}$ is a covariance matrix that should be estimated and $\mathbf{Z}$ is a model matrix for the random effects. The model form of the rabbit example becomes:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \\ y_{10} \\ y_{11} \\ y_{12} \end{bmatrix} = \begin{bmatrix} x_1 & 0 & \cdots & \cdots & 0 \\ 0 & x_2 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & x_{11} & 0 \\ 0 & \cdots & \cdots & 0 & x_{12} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \\ \beta_6 \\ \beta_7 \\ \beta_8 \\ \beta_9 \\ \beta_{10} \\ \beta_{11} \\ \beta_{12} \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \\ \epsilon_6 \\ \epsilon_7 \\ \epsilon_8 \\ \epsilon_9 \\ \epsilon_{10} \\ \epsilon_{11} \\ \epsilon_{12} \end{bmatrix}$$

Now, β_i and z_i should be estimated. The detailed estimation process of obtaining these values can be found in [8].

By using the same principle as in (2.6), we can also add random effect terms to the GAM.

$$y_i = \mathbf{X}_i\boldsymbol{\beta} + f_1(x_{1i}) + f_2(x_{2i}, x_{3i}) + \cdots + \mathbf{Z}_i\mathbf{b} + \epsilon_i \tag{2.8}$$

This model is called **generalized additive mixed models** (GAMM).

2.5 Model selection criteria

Imagine that you are using a linear model and you have some different formulas. You want to know which formula fits the data best. Model selection criterion is a helpful tool to do this.

2.5.1 adjusted R-squared

Imagine that you have 1000 linear models. Visually comparing 1000 plots is almost impossible. **Adjusted R-squared** can be used in this case. Adjusted R-squared is basically subtracting the scaled average distance between the fitted line and the data points from 1. It is defined as

$$r^2_{adj} = 1 - \frac{\sum \hat{\epsilon}_i^2/(n-p)}{\sum (y_i - \bar{y})^2/(n-1)}$$

Bigger value of r^2_{adj} means better fit.

2.5.2 GCV

In section 2.3.2.3 we introduced **generalized cross validation score (GCV)** as the selection criterion of λ in GAM. So, each GAM model will contain this GCV value that used in the internal selection of λ. This GCV value can also be used to compare different GAM models. As in section 2.3.2.3, smaller GCV value means better fit.

2.5.3 fREML

There are several options for implementing GAM with `gamm4` package in `R`. `gam` function in `gamm4` uses GCV score as the selection criterion of λ and it also provides this GCV score in `summary` so that it can be used for model selection. Another possibility is `bam`. This function is numerically optimized version of `gam`. It can fit large dataset easily and fast. `bam` uses **fast restricted maximum likelihood** (fREML) instead of GCV. fREML, which generally accepted as a good replacement of GCV, uses somewhat different method but its basic goal and principle is same as GCV. Theoretical details can be found in [9] and [10].

Chapter 3

Analysis: Generalized Mixed Models of the Wieling data

3.1 Exploratory data analysis

3.1.1 Dataset from Wieling

The aim of this paper is to re-analyze the data that used in [2]. Martijn Wieling offered me the dataset that he used in [2]. The dataset is in the form of `data.frame` in `R`. This dataset will be denoted as `dialectNL` from now on. `dialectNL` consists of 225866 rows (424 locations $\times$ 559 words - some missing words) and 19 variables.

Variables are:

- PronDistStdDutch: pronunciation distance from standard Dutch. This is the dependent variable.
- Word: the words which pronunciations were recorded (559 levels, the complete list can be found in appendix)
- WordCategory: word category (Noun, Adjective, Verb, adverB)
- Transcriber: the transcriber of the subject's pronunciations (30 levels)
- Location: the dialect location (424 levels, the complete list can be found in appendix)
- Longitude: longitude of the dialect location
- Latitude: latitude of the dialect location
- WordFreq.log: word frequency (log-transformed)
- WordIsNounOrAdverb: contrast indicating if the word is a noun or adverb (1) or not (0)
- WordLength.log: number of sounds in the standard pronunciation of the word (log-transformed)
- WordVCratio.log: vowel-to-consonant ratio in the standard pronunciation of the word
- PopSize.log: number of inhabitants in the location (log-transformed)
- PopSize.log_residGeo: as above, but excluding the influence of geography

- PopAvgIncome.log: average income in the location (log-transformed)

- PopAvgIncome.log_residGeo : as above, but excluding the influence of geography

- PopAvgAge: average age in the location

- PopAvgAge_residPopAvgIncome.log_Geo : as above, but excluding the influence of average income and geography

- PopMaleFemaleRatio: male-female ratio in the location

- SpeakerIsMale: value between 0 and 1 indicating the proportion of speakers who are male (incidentally more speakers were present)

- SpeakerBirthYear: year of birth of the speaker (or average when multiple speakers were present)

- SpeakerRecordingYear: year in which the speaker was recorded

- FieldworkerIsMale: value between 0 and 1 indicating the proportion of transcribers who are male (incidentally more transcribers were present)

3.1.2 Scatter plot matrix

To see the relationship between the dependent variable and independent variables, scatter plot matrix was created. In this matrix, no direct visual relationship could be seen. Because this scatter plot matrix has a huge size, is not included in this paper, but user can easily recreate it in `R` by using `pairs()`.

3.1.3 Geographical distribution of PronDistStdDutch

To see the geographical distribution of PronDistStdDutch, average of PronDistStdDutch is calculated per location. This average value is displayed in figure 3.1 by using black color. The more blackish, the smaller value of PronDistStdDutch, which means that its pronounciation is close to the standard Dutch. In this figure, it is obvious to see that the most standard Dutch is spoken in province Holland and more dialect-pronunciation is spoken in southern and northern provinces like Groningen and Limburg.

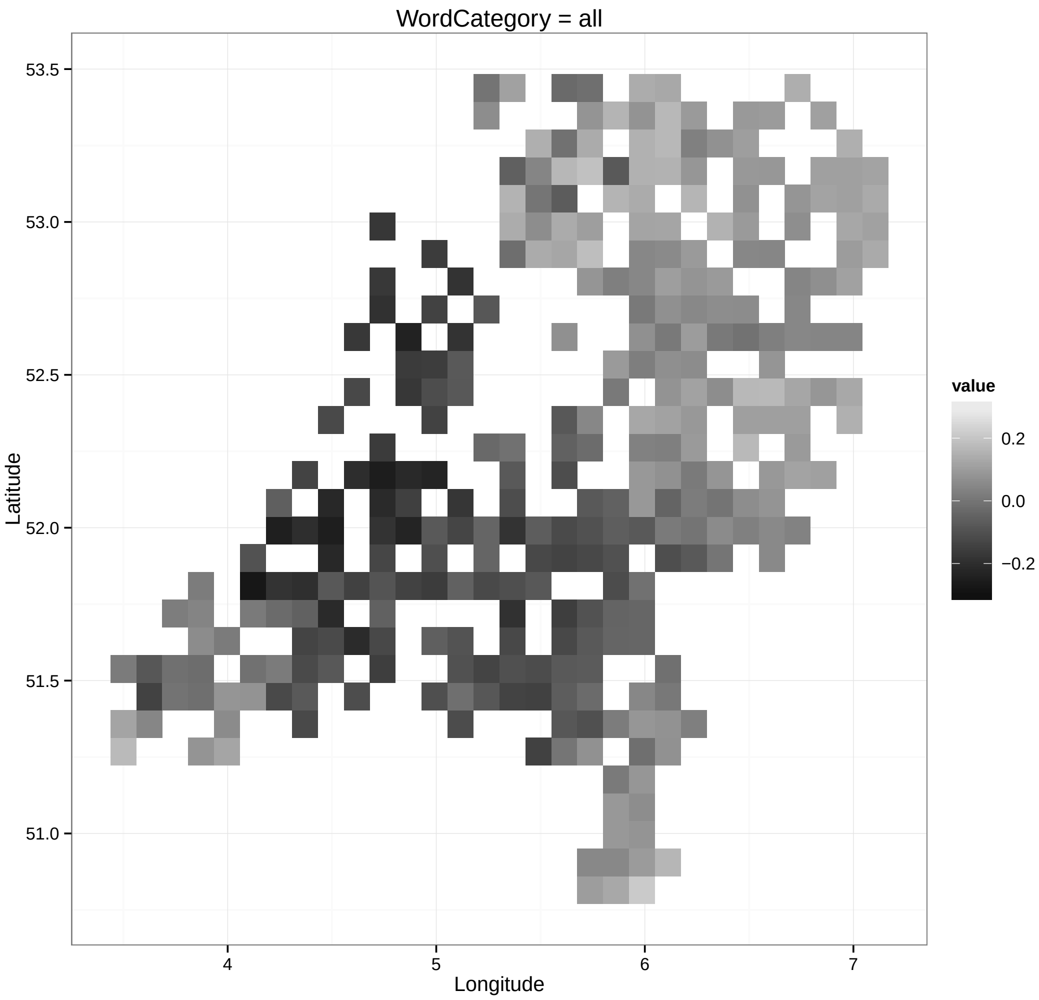

Figure 3.1: geographical distribution of PronDistStdDutch. The more blackish, the smaller value of PronDistStdDutch

`dialectNL` contains 4 word categories: noun (225 words, 40.3% of all words), adjective (116 words, 20.8%), verb (172 words, 30.7%), and adverb (46 words, 8.2%). This time, the average of PronDistStdDutch is calculated per word category. Figure 3.2 shows the result.

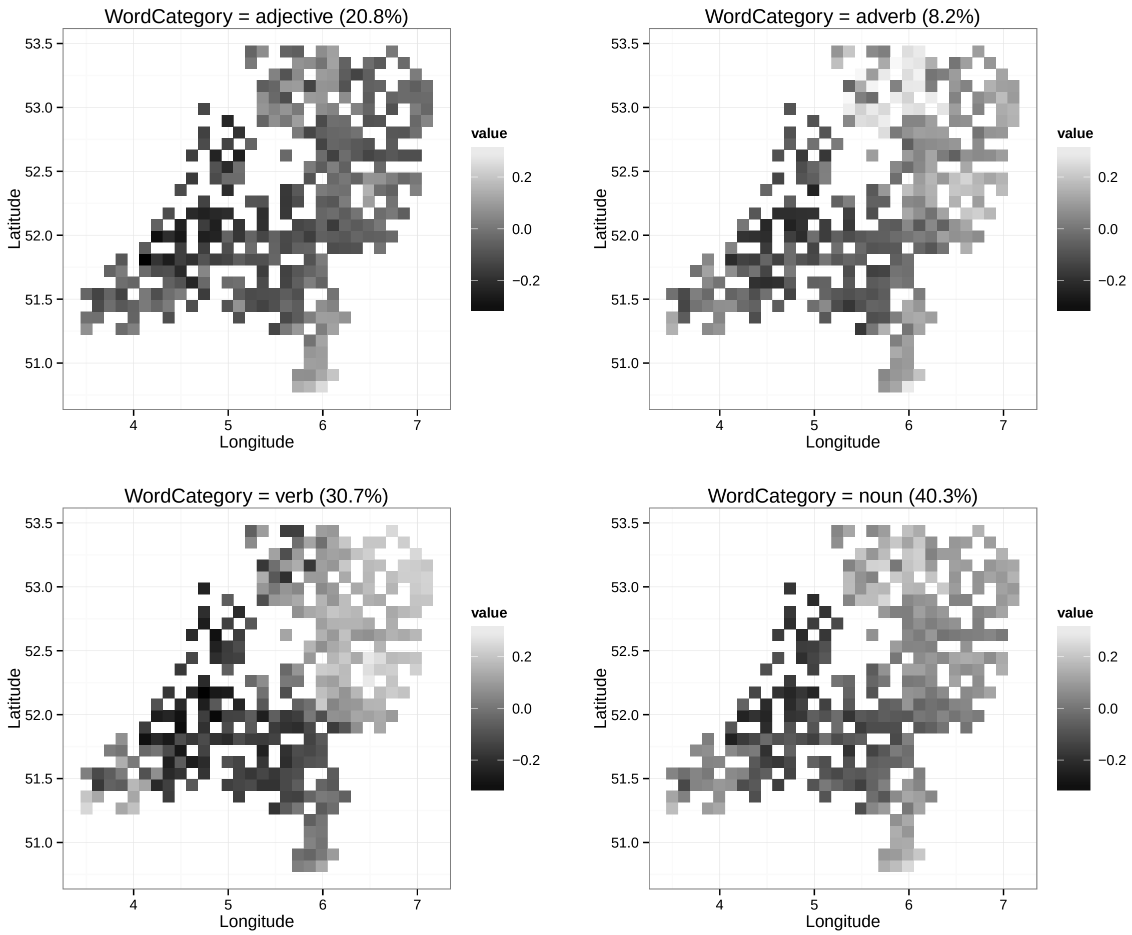

Figure 3.2: geographical distribution of PronDistStdDutch per word category

The image of adjective shows somewhat less contrastive image than others. Even though the images of adjective, adverb, and noun show different degree of contrast, they have more or less the same type of geographical distribution. However, the image of verb has different type of distribution. For example, Drenthe and Groningen have the the most white color.

3.1.4 The center of the standard Dutch pronounciation

In section 3.1.3, we saw that more non-standard pronunciation is spoken as the location gets more distance from Holland area. Question is, where is the exact "center" of the standard Dutch pronounciation? The criterion as a center is simple: the more you get away from the center, the more the pronunciation becomes non-standard (bigger value of PronDistStdDutch).

To decide the center location, we first chose a arbitrary location (call it location C). For every location (except C), we calculated distance from C. For example, the distance of

Word category = all	
Location	r^2_{adj}
Driebruggen ZH	0.6156
Meije ZH	0.6144
Aarlanderveen ZH	0.6119
Gouda ZH	0.6119

Table 3.1: top 4 locations that have highest r^2_{adj}

location X from $C = \sqrt{(\text{Longitude}_X - \text{Longitude}_C)^2 - (\text{Latitude}_X - \text{Latitude}_C)^2}$.
Then, we used this distance as independent variable and PronDistStdDutch as dependent variable to fit linear model. We repeated this for every possible choice of C. We finally obtained 424 fitted linear models. Because visually comparing this much scatter plots is almost impossible, we used adjusted R-squared, which is introduced in section 2.5.1.
Table 3.1 shows the top 4 center locations based on r^2_{adj}. As expected, they are located in Holland. Figure 3.3 contains the corresponding scatter plots and fitted lines.

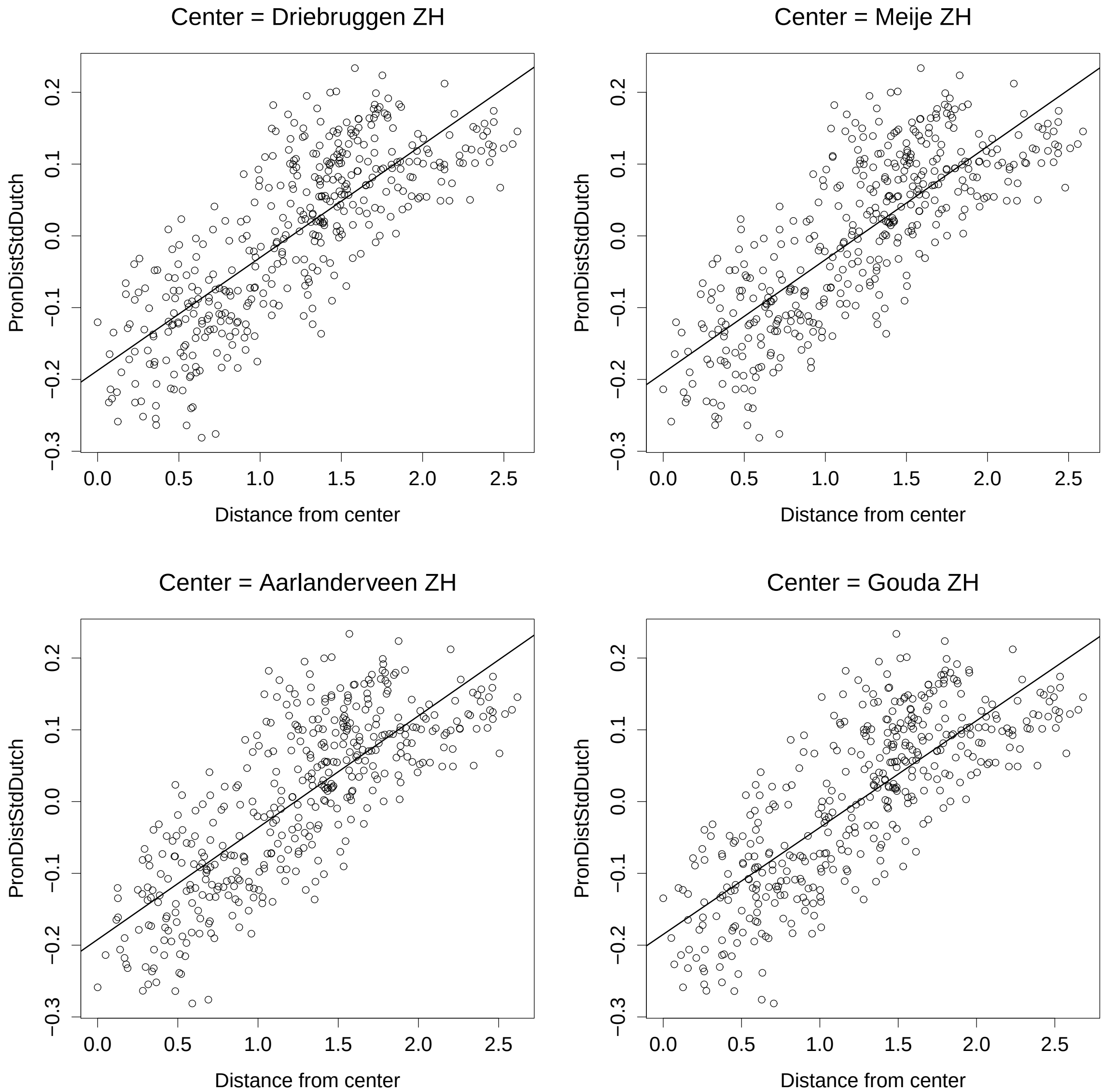

Figure 3.3: scatter plots of top 4 center locations

In figure 3.2, we saw that the geometrical distribution differs per word category. We repeated the same method of deciding center location with r^2_{adj} for each word category. Table 3.2 contains the result of this. Verb has the biggest values of r^2_{adj} and adjective has the smallest values. This verifies what we saw in figure 3.2. Figure 3.4 contains scatter plots and fitted lines of the best center locations per each word category. The fitted line of verb has high slope and the fitted line of adjective has low slope. This implies that adjective is relatively less sensitive to geography and verb is relatively more. This fact also confirms figure 3.2.

Word category = adjective	
Location	r^2_{adj}
Aalsmeer NH	0.2857
IJmuiden NH	0.2792
Aarlanderveen ZH	0.2789
Zandvoort NH	0.2780

Word category = adverb	
Location	r^2_{adj}
Bleskensgraaf ZH	0.4542
Sliedrecht ZH	0.4537
Boven-Hardinxveld ZH	0.4520
Stolwijk ZH	0.4516

Word category = verb	
Location	r^2_{adj}
Stolwijk ZH	0.6318
Gouda ZH	0.6290
Driebruggen ZH	0.6286
Bleskensgraaf ZH	0.6275

Word category = noun	
Location	r^2_{adj}
Driebruggen ZH	0.5369
Zegveld Ut	0.5361
Meije ZH	0.5356
Oudewater Ut	0.5352

Table 3.2: best center locations per word category by r^2_{adj}

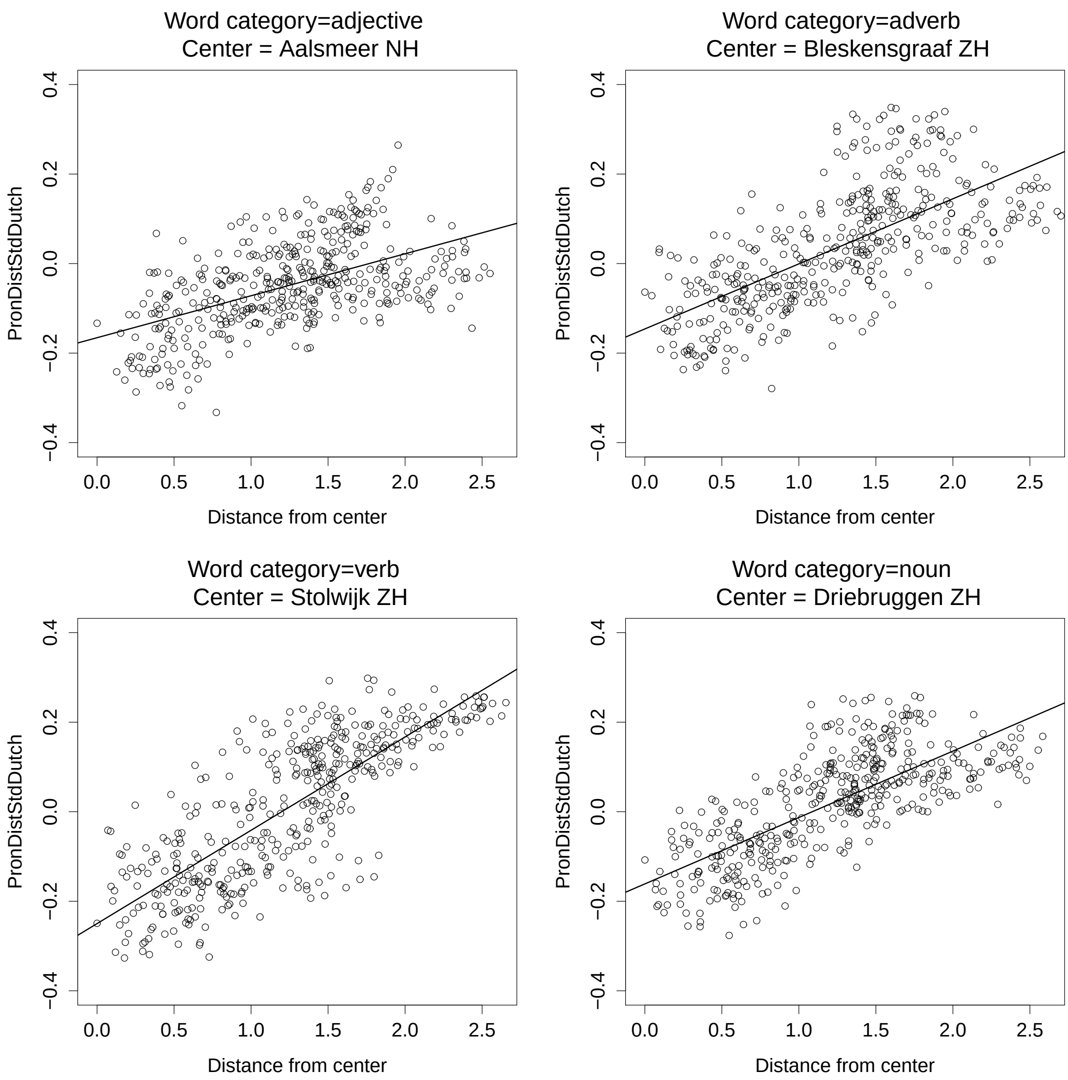

Figure 3.4: scatter plots and fitted lines of top center locations per word category

Location	PopAvgAge
Urk Fl	25.91
Bergentheim Ov	29.16
Houten Ut	29.32
Genemuiden Ov	29.86
⋮	⋮
Laren NH	42.87
Oosterbeek Gl	43.02
Gorssel Gl	43.96
Rozendaal Gl	46.10

Location	PopAvgIncome.log
Urk Fl	8.724044746
Bergentheim Ov	8.774312958
De Harkema Fr	8.81401908
s-Heerenbroek Ov	8.784774592
⋮	⋮
Groenekan Ut	9.310185707
Middelie NH	9.336532397
Laren NH	9.366061247
Rozendaal Gl	9.424483541

Table 3.3: head and tail values of PopAvgAge and PopAvgIncome.log

3.1.5 Correlation between PopAvgAge and PopAvgIncome.log

Table 3.3 shows the top and bottom 4 points of PopAvgAge and PopAvgIncome. Figure 3.5 shows its scatter plot with fitted line. These two variables have a correlation of 0.439698, which is relatively high. Further, PopAvgIncome.log has minimum value of 8.774312958 and maximum value of 9.424483541. This implies that, the minimum yearly average income is €$6,466(=e^{8.774312958})$ and the maximum €$12,388(=e^{9.424483541})$ which are extremely low. These data were offered by Statistics Netherlands (Dutch: CBS) and obtained by simply dividing the total income of the location by the total population of the location. This total population included children, who have income of 0. This clarifies the correlation.

Wieling succeeded to cancel out the effect of average income and geography from PopAvgAge by using the residuals of a linear model regressing. He offered us these data. In main analysis, this decorrelated version of PopAvgAge will be used.

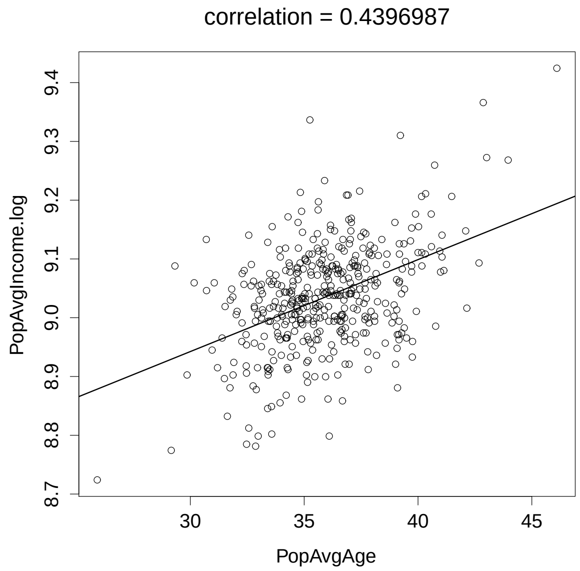

Figure 3.5: correlation between PopAvgAge and PopAvgIncome.log

3.2 Main analysis

Wieling, Nerbonne, and Baayen tried to fit generalized additive mixed models into `dialectNL` in [2]. Their model had following components:
- dependent variable: PronDistStdDutch
- fixed effects: PopSize.log_residGeo, PopAvgAge_residPopAvgIncome.log_Geo, PopAvgIncome.log_residGeo, WordFreq.log, WordVCratio.log.z, WordIsNounOrAdverb
- random effects: Word, Location, Transcriber

At the moment they performed their research (2011), they couldn't estimate this model by using `gamm`. Namely, the size of the dataset in combination with the complicated random effects (Word has 559 levels and Location has 424 levels) caused so called "big data problem". What they did as alternative was first analyzing geographical effect with `gam` and then using `lmer` to analyze other fixed and random effects with fitted value of geographical effects obtained from `gam`. This two-step analyzing method has some disadvantages. Each separated step cannot effect each other actively during the fitting process. Also, it's difficult to compare and evaluate different models, because the model selection criterion such as GCV or fREML can't be used. We try to analyze the data in one step.

3.2.1 Big data problem

First, we tried to run the model they used in [2] by using `gamm`. The model was:

```
gamm.model=gamm(PronDistStdDutch.c~te(Longitude,Latitude)+
```

```
s(PopSize.log_residGeo)+s(PopAvgAge_residPopAvgIncome.log_Geo)+
s(PopAvgIncome.log_residGeo)+s(WordFreq.log)+s(WordVCratio.log.z)+
s(WordIsNounOrAdverb),
random=list(Word=~1,Location=~1,Transcriber=~1),data=dialectNL)
```

This failed because the calculation required more than 2 gigabyte of ram. When we removed all random effects and used `gam`, the whole dataset could be run without any problem. It became clear that the random effects were causing the heavy computation load. After some trial and error, we found that a calculable maximum size of subset is around 2000 rows. However, the result of the fitted model using the subset size 2000 was very unstable. When we repeated model fitting with 10 different random subsets, each trial gave completely different conclusion about the significance of independent variables.

3.2.2 bam

In `gamm4` package, there is a function called `bam`. This is a numerically optimized version of `gam` so that it can easily and fast deal with big data. The research team of Prof. Nerbonne recently found a way to use `bam` to replace `gamm` in their not-yet-published research [11]. In `bam` random effects can be specified by using ridge penalty. The ridge penalty is equivalent to an assumption that the coefficients are i.i.d. normal random effects, which is exactly what mixed models do with random effects. Ridge penalty can be specified with command `s( ··· ,bs="re")`. `gamm.model` now becomes

```
bam.model=bam(PronDistStdDutch.c~te(Longitude,Latitude)+
s(PopSize.log_residGeo)+s(PopAvgAge_residPopAvgIncome.log_Geo)+
s(PopAvgIncome.log_residGeo)+s(WordFreq.log)+s(WordVCratio.log.z)+
s(WordIsNounOrAdverb)+
s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re"),data=dialectNL)
```

To test whether `bam` really can be used as a replacement of `gamm`, we made 4 new models by omitting some variables from Wieling's model. To perform model selection with these 4 models, we randomly created 40 subsets of 2000 rows. First, we performed `gamm` with model 1 and subset 1, then we used `predict` with the whole dataset (`dialectNL`) to derive $\hat{y}$ (estimated value of PronDistStdDutch.c according to model 1). Then we compared this value with y (PronDistStdDutch.c) in the form of

$$\Delta = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

We repeated this procedure per model and per subset (160 iterations). We derived mean and variance of Δ's per model, which can be found on the left side of table 3.4

mean	variance	model nr.
0.0797	1.083860e-07	4
0.0803	6.039210e-08	1
0.0803	1.759416e-08	3
0.0806	1.676639e-08	2

mean	variance	model nr.
218.3453	534.9676	4
219.7833	524.3729	1
237.9096	577.8901	3
297.6824	412.8126	2

Table 3.4: Left: mean and variance of Δ per model, ordered by mean.
Right: mean and variance of fREML per model, ordered by mean.

Next, we repeated an analogic procedure with `bam`. We used the same four models and the same subset that were used in `gamm`. This time, fREML was used as a model selection criterion. By calculating mean and variance of fREML scores of each model we obtained the right side of table 3.4.

Based on these results, we concluded that `bam` is a good substitution of `gamm`. From now on, `bam` will be our main method and fREML will be our main model selection criterion.

3.2.3 Model selection: take word category effect into geography

In section 3.1.3, we saw that the geographical distribution differs per word category, especially when the category is verb. We tried to reflect this finding in our model. First, we created a new variable "WordIsVerb". If the category of word is verb, WordIsVerb has value 1 and in other cases, 0. We made a variation of Wieling model with this variable. We performed a model selection with following four models.

- model 1 = Wieling model with geography variables removed.
- model 2 = the original Wieling model
- model 3 = Wieling model with modified geography that uses WordIsVerb to make two separate maps: one for the verbs and one for the rest
- model 4 = Wieling model with modified geography that uses WordIsNounOrAdverb

We performed model selection in two ways. Firstly, we performed model selection with the complete `dialectNL` dataset, since `bam` can handle it. Secondly, we used 20 random subsets of 10,000 observations to calculate mean and variance of fREML for each model.

Model 1, which contains no geographical variables, shows the worst fit. So, geography is an essential factor in explaining the distribution of dialects. Model 3, which has the modified geographical term with verb, shows the best fit. Separating geography by word category verb therefore improves the model.

mean	variance	model nr.
183.8597	3514.297	3
192.3941	3174.391	4
217.1635	2028.408	2
305.5390	2217.733	1

fREML	model nr.
-16026.83	3
-13317.80	4
-12377.09	2
-12149.38	1

Table 3.5: Left: mean and variance of fREML per model (20 random subsets of 10,000 observations were used) Right: fREML score for each model (the full `dialectNL` is used), both of them are ordered by mean.

3.2.4 Model selection: non-geographical terms

We fix the geographical term with the result from section 3.2.3. Now, we test non-geographical terms with 15 different models:

- model 1 = full Wieling model with modified geography term (=model 3 in section 3.2.3)
- model 2 = omitting Word random effect from model 1
- model 3 = omitting Location random effect from model 1
- model 4 = omitting Transcriber random effect from model 1
- model 5 = omitting WordFreq.log from model 1
- model 6 = omitting PopSize.log_residGeo from model 1
- model 7 = omitting PopAvgAge_residPopAvgIncome.log_Geo from model 1
- model 8 = omitting PopAvgIncome.log_residGeo from model 1
- model 9 = omitting WordVCratio.log.z from model 1
- model 10 = omitting PopSize.log_residGeo and PopAvgAge_residPopAvgIncome.log_Geo from model 1
- model 11 = omitting PopSize.log_residGeo, PopAvgAge_residPopAvgIncome.log_Geo, and Transcriber random effect from model 1
- model 12 = omitting PopSize.log_residGeo and PopAvgIncome.log_residGeo, from model 1
- model 13 = omitting WordFreq.log and PopAvgIncome.log_residGeo, from model 1
- model 14 = omitting WordFreq.log, PopAvgIncome.log_residGeo, and PopSize.log_residGeo from model 1
- model 15 = omitting all splines terms `s(···)` from model 1

The whole `dialectNL` has been used to perform this model selection. Table 3.6 shows the result.

Model 2 shows extremely bad fit. So, the Word random effect is a very important factor in our model. The same goes for the model 3, model 9, model 4, and model 11 although they are not as extremely bad as model 2. Therefore, all three random effects are important

fREML	model nr.
-16033.48	12
-16032.89	14
-16031.07	13
-16028.62	6
-16028.15	10
-16028.12	8
-16027.68	7
-16023.26	1
-16022.68	5
-16018.38	15
-16006.15	11
-16003.90	4
-15970.88	9
-11753.27	3
17144.07	2

Table 3.6: fREML score for each model, ordered by fREML

factors.
Model 12 shows the smallest fREML value. Thus, we choose model 12.

3.2.5 Model expansion: adding new variables

In the previous sections we improved the model fit by adjusting geographical term and we simplified the model by removing some non-significant variables. In this section, we try to expand the model with some new variables. The variables used for expanding, were already in the dataset, but the research team of Wieling decided to exclude them in their model.
- model 1 = Our favorite model from the previous section (=model 12 in section 3.2.4)
- model 2 = model 1 with PopMaleFemaleRatio
- model 3 = model 1 with SpeakerBirthYear
- model 4 = model 1 with SpeakerRecordingYear
- model 5 = model 1 with PopMaleFemaleRatio, SpeakerBirthYear, and SpeakerRecordingYear

fREML	model nr.
-16644.97	3
-16644.82	4
-16636.42	5
-16033.50	1
-16029.28	2

Table 3.7: fREML score for each model, ordered by fREML

The whole `dialectNL` has been used to perform this model selection. Table 3.7 shows the result. Obviously, adding SpeakerBirthYear improves the model. Therefore, we choose model 3 as our final model.

Chapter 4

Conclusion & Discussion

4.1 Conclusion

In the previous section, model 3 of table 3.7 became our final choice of model. This model is written in `R` as:

```
dialectNL$WordIsVerb=as.factor(dialectNL$WordIsVerb)
final.model=bam(
PronDistStdDutch.c~te(Longitude,Latitude,by=WordIsVerb,d=c(1,1))+
s(PopAvgAge_residPopAvgIncome.log_Geo.z)+s(WordFreq.log)+
s(WordVCratio.log.z)+s(SpeakerBirthYear)+
s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re"), data=dialectNL)
```

Compared to Wieling model, this model has a modified geography term and contains extra variable SpeakerBirthYear, and it excludes PopSize.log_residGeo and PopAvgIncome.log_residGeo. Table 4.1 shows the goodness of fit of the variables of this model. 'F' shows the 'F-ratio statistic' value of each variable. 'p-value' uses a hypothesis testing like in section 2.2.2, but F-distribution is assumed instead of chi-squared distribution. Thus, 'p-value' indicates the probability that a new random value on the F-distribution is as extreme or more extreme than the critical value.

Figure 4.1 shows the independent variables and their effect on the dependent variable (PronDistStdDutch.c). Figure 4.2 shows the same graphs, but the range of y-axis of each graph is set to the same, so that the effect sizes can be compared.

	edf	Ref.df	F	p-value
te(Longitude,Latitude):WordIsVerb0	15.91	16.11	45.41	0.00
te(Longitude,Latitude):WordIsVerb1	21.44	22.03	103.23	0.00
s(PopAvgAge_residPopAvgIncome.log_Geo.z)	3.43	3.46	2.61	0.04
s(WordFreq.log)	1.60	1.61	4.60	0.02
s(WordVCratio.log.z)	5.98	5.99	23.88	0.00
s(SpeakerBirthYear)	4.27	4.30	2.48	0.04
s(Word)	545.68	556.00	141.87	0.00
s(Location)	346.66	396.00	32.36	0.00
s(Transcriber)	18.87	29.00	2001.38	0.00

Table 4.1: goodness of fit of the variables of the final model

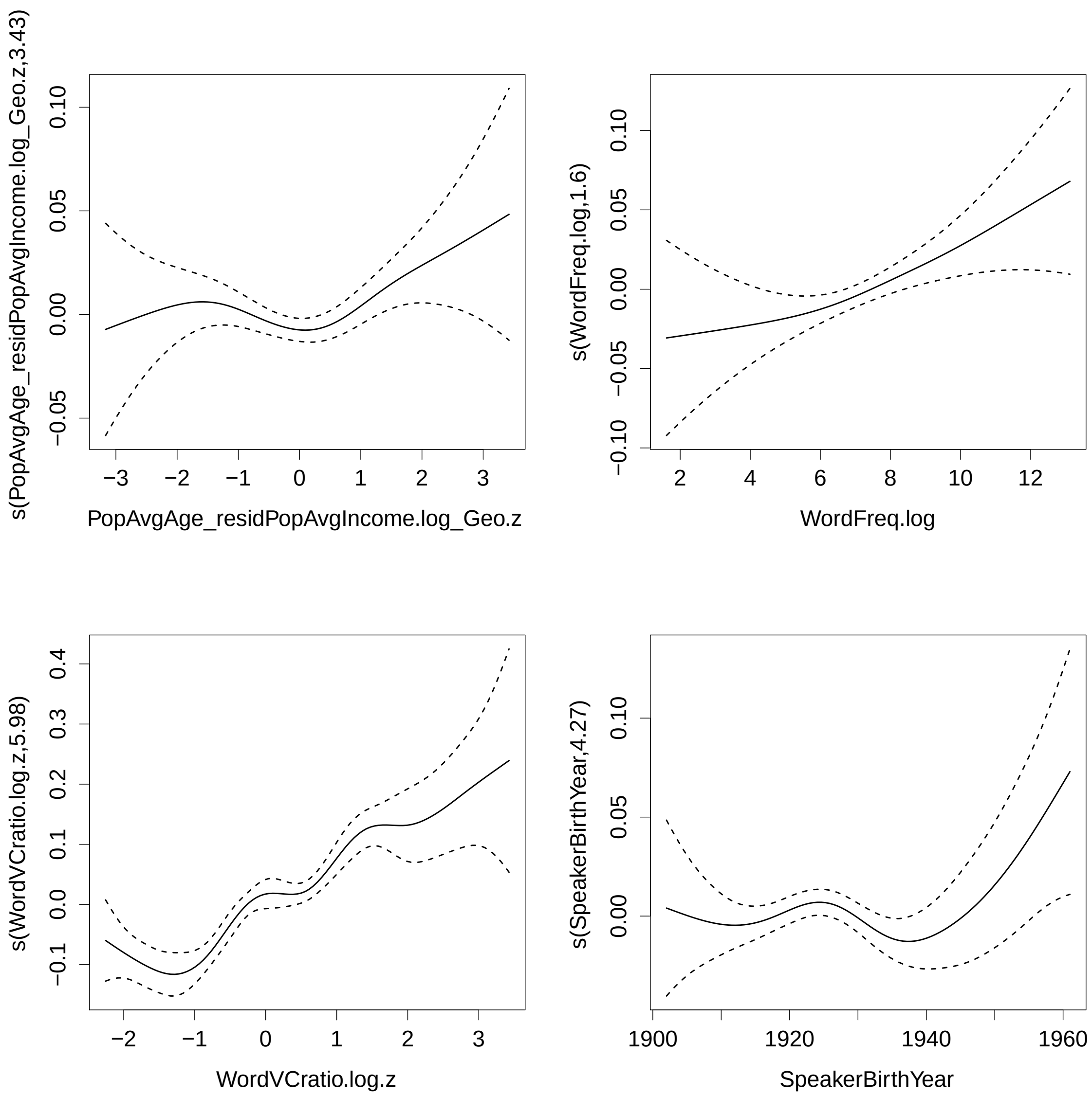

Figure 4.1: effect of independent variables on the dependent variable (PronDist-StdDutch.c)

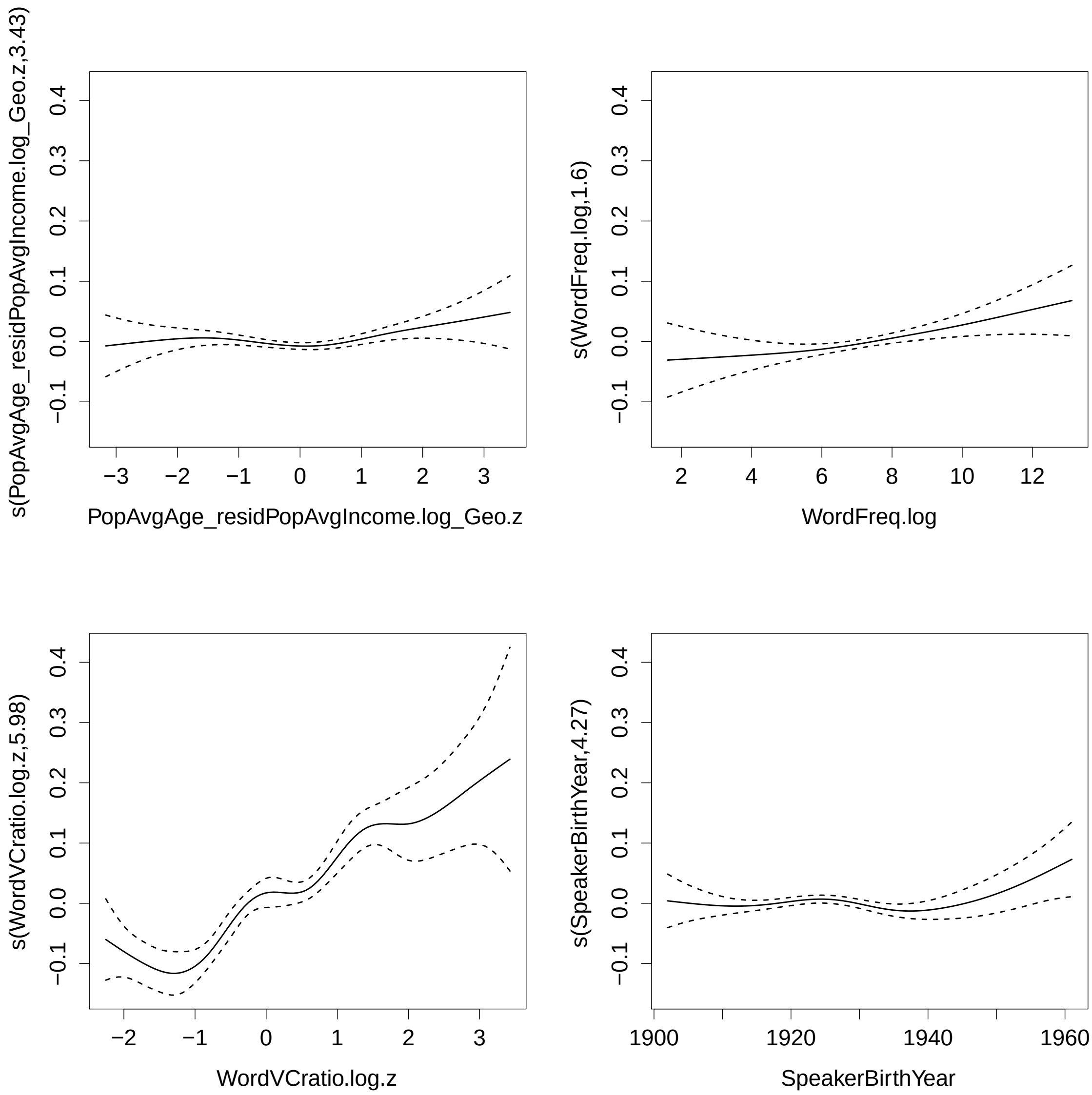

Figure 4.2: effect of independent variables on the dependent variable, range of y-axis is set to that of WordVCratio.log.z

WordVCratio.log.z has the biggest impact on the dependent variable. It generally seems that, if a word contains more vowels, then its phonetic distance from the standard Dutch gets greater. Yet, if a word contains extremely few vowels, this effect doesn't hold. In [2], they found a positive linear relationship.

From the graph of PopAvgAge_residPopAvgIncome.log_Geo.z, it seems that high average age of inhabitants predicts greater phonetic distance from the standard Dutch. However, between -1.5 and 0 the graph has a negative slope, which is a new finding compare to the result of [2].

As WordFreq.log increases, the value of the dependent variable gets bigger. This matches

with the result of [2].
SpeakerBirthYear is a variable that is not used in [2]. In contrast to the general belief that old people speak more dialect, the graph shows generation-specific result. People born around in 1910 and 1940 speak less dialect. From 1940 till 1960, younger people speak more dialect.

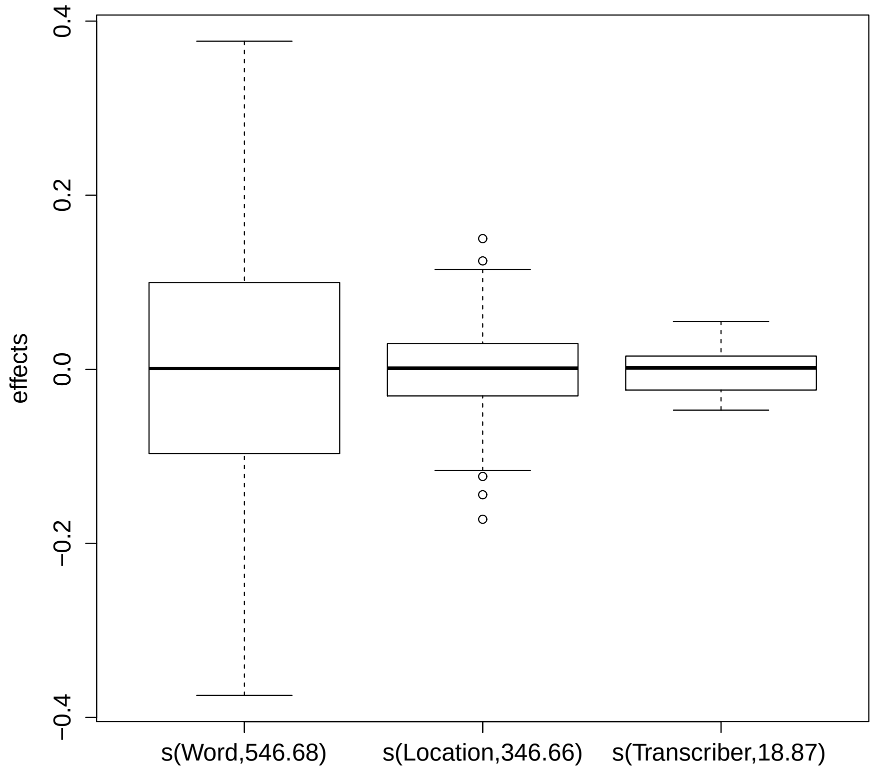

Figure 4.3: boxplot of random effects

Figure 4.3 shows the boxplot of the random effects. As we saw in 3.2.4, Word has the biggest effect and Transcriber has the smallest effect.
Our final model includes two separate maps, one for verbs and one for all other word categories. Figure 4.4 shows those two maps. The difference between the two geographical distributions are mostly in the Northeast side of the country.

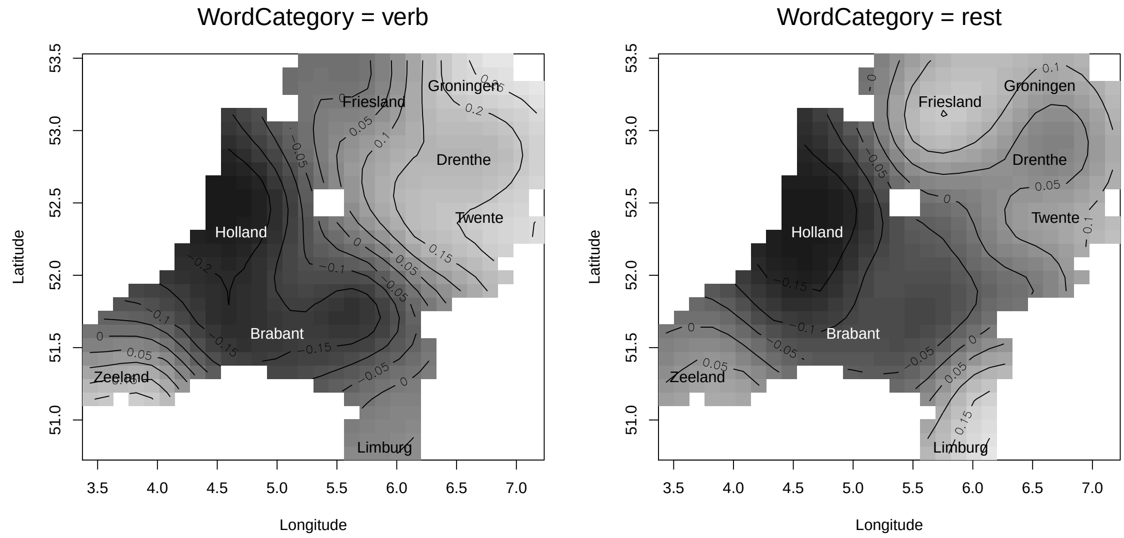

Figure 4.4: Left: geographical distribution of verb, Right: geographical distribution of other word categories

4.2 Discussion and possible improvements

`dialectNL` contains one extra variable that is not discussed in this study:

- SpeakerEmploymentLevel : employment level of the speaker (or average when multiple speakers were present)

We did not include this variable in the model because it had a lot of missing points (22.5%). Wieling did the same in his study because of the same reason. It can be tried to fill the missing data by contacting speakers again, but this will be a difficult field work. Alternatively, missing data can be filled in by average value. However, since the variable has many missing points, this will possibly have some influence on the variable. Taking a subset that excluding missing points is also tricky since this subset can cause error during the fitting process because of the possible missing values in the random effects.

There was a moderate correlation between two independent variables: PopAvgAge and PopAvgIncome. Wieling succeed to cancel out the correlation by using some statistical methods [2]. This decorrelated version was used in our analysis. Yet, it is more trustworthy to use natively decorrelated variables, because it would save one step in our analysis and subsequently make our analysis simpler. Native decorrelating can be done in data-gathering phase by using more sophisticated way of calculating average income. For example, (total income / total number of adults) or (total income / inhabitants who have job or receiving government aid).

A reader might wonder whether we would obtain the same result if we repeat this study with another dataset. If another dataset is from around the same time as our dataset, we expect that the result would be the same. This is because we used cross-validation based criterion to choose our model, which prevents our model being too data-specific. However, if the new dataset is from another period of time or is from other region (i.e. Belgium), we expect that the new data will lead us to a new result.

References

[1] Wikipedia. Netherlands — wikipedia, the free encyclopedia, 2013. [Online; accessed 30-June-2013].

[2] Martijn Wieling, John Nerbonne, and R. Harald Baayen. Quantitative social dialectology: Explaining linguistic variation geographically and socially. *PLoS ONE*, 6(9):e23613, 09 2011.

[3] Johan Taeldeman and A Goeman. Fonologie en morfologie van de nederlandse dialecten: een nieuwe materiaalverzameling en twee nieuwe atlasprojecten. *Taal en Tongval*, 48:38–59, 1996.

[4] Martijn Wieling and John Nerbonne. Measuring linguistic variation commensurably. *Dialectologia: revista electrònica*, pages 141–162, 2011.

[5] C Gussenhoven. Illustrations of the ipa: Dutch. handbook of the international phonetic association (pp. 74–77), 1999.

[6] A.J. Dobson and A.G. Barnett. *An introduction to generalized linear models*. Chapman and Hall/CRC Texts in Statistical Science Series. Chapman & Hall/CRC, 2008.

[7] C. Gu. *Smoothing Spline ANOVA Models*. IMA Volumes in Mathematics and Its Applications. Springer, 2002.

[8] S.N. Wood. *Generalized additive models: an introduction with R*. Chapman and Hall/CRC Texts in Statistical Science Series. Chapman and Hall/CRC Press, 2006.

[9] S.N. Wood. Stable and efficient multiple smoothing parameter estimation for generalized additive models. *Journal of the American Statistical Association*, 99:673–686, September 2004.

[10] S.N. Wood. Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. *Journal of the Royal Statistical Society, Series B (Statistical Methodology)*, 73(1):3–36, January 2011.

[11] Martijn Wieling, Simonetta Montemagni, John Nerbonne, and R Harald Baayen. Lexical differences between tuscan dialects and standard italian: A sociolinguistic analysis using generalized additive mixed modeling.

[12] Simon Wood. *gamm4: Generalized additive mixed models using mgcv and lme4*, 2012. R package version 0.1-6.

[13] Simon N Wood. Thin plate regression splines. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(1):95–114, 2003.

[14] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707, 1966.

[15] Robert Andersen. Lecture slides: SOC6078 Advanced Statistics: 9. Generalized Additive Models. University of Toronto, 2012.

[16] Breheny Patrick. Lecture slides: BST 764: Applied Statistical Modelling: Generalized Additive Models 1. University of Kentucky, 2011.

[17] Simon Wood. Lecture slides: Generalized Additive Models. University of Bath, 2010.

[18] Mark Irwin. Lecture slides: Statistics 135: Generalized Additive Models. Harvard Unviersity, 2005.

Appendix A

A.1 Appendix 1 - Complete list of “Word”

aarde
aardig
acht
achter
adem
af
anders
appels
arm
armen
autos
baarden
bakken
barsten
bedden
beenderen
beginnen
benen
beren*wild*
best*bijw*
beurzen
beven
bezems
bezig
bidden
bier
bijen
bijten
bij*vz*
binden
bitter
bladen
bladeren
blauw
blazen
bleek
blijven
blind
bloeden
bloeien
blond
blozen
bokken
bomen
bonen
boren
boter
bouwen
boven
braaf
braden
branden
breed
breien
breken
brengen
broden
broeken
broers
bruin
buigen
buiten
dagen
daken
damp
dansen
darmen
deeg
denken
derde
deuren
dienen
diep
dieven
dik
dingen
dinsdag
dochters
doeken
doen
dol
donder
donderdag
donker
doof
dooien
door
dopen
dorsen
dorst
draaien
draden
dragen
dreigen
drie
drinken
dromen
droog
dubbel
duiven
duizend
dun
durven
duur
duwen
dweilen
echt
eeuwen
eieren
eigen
einde
elf
engelen
enkel
eten
ezels
fel
fijn
flauw
flessen
fruit
gaan
gal
ganzen
gapen
gebruiken
geel
gehad
geld
geloven
genoeg
gerst
geven
geweest
gewoon
gisteren
glazen
god
goed
goud
gouden
gras
graven
grijs
groen
grof
groot
haast
haastig
haken
halen
half
handen

hanen
hangen
hard
haver
hebben
heel
heet
heffen
heilig
helpen
hemden
hemel
hengsten
heren
heten
hier
hoeden
hoesten
hol
holen
honden
honger
hoog
hooi
hoop*espoir*
hopen
horen
horens
houden
huizen
jagen
jeuken
jong
jongen
juist
kaas
kaf
kalm
kalveren
kamers
kammen*mv*
kammen*ww*
kanten
karren
kasten
katten
kennen
kermis
kersen
kervel
keuren
kiezen
kijken
kinderen
klaver
kleden
klederen
klein
kloppen*1*
kloppen*2*
knechten
kneden
knieen
koeien
koel
koken
komen
kommen
konijnen
koorts
kopen
koper
kort
koud
kousen
kraken
kramp
kreupel
krijgen
krimpen
krom
kruipen
kwaad
laag
laat
lachen
lam
lammeren
lampen
lang
lastig
laten
latten
leden
leem
leggen
leren
leugens
leunen
leven
lezen
licht
liederen
liggen
lijken
likken
lomp
lopen
lucht
lui
luiden
luisteren
maandag
maanden
maart
magen
mager
maken
marmer
maten
mazelen
meer
mei
meid
melk
menen
merg
metselen
meubels
missen
modder
moe
moes
moeten
mogelijk
mogen
morgen*demain*
mossels
muizen
muren
naalden
nat
negen
negers
nieuw
noemen
nog
noorden
noten
nu
ogen
om
ons
oogst
ook
oosten
op
open
oud

over
paarden
padden
paden
Pasen
pekel
pellen
peper
peren
piepen
pijpen
planken
pleinen
ploegen*werktuig*
potten
proeven
proper
raar
raden
recht
redden
regen
rekken
ribben
riet
rijden
rijk
rijp
rijst
ringen
roepen
roeren
rogge
rokken
rond
rondes
rood
rook
ruiken
runderen
ruzies
sap
saus
schade
schapen
schaven
scheef
scheel
scheiden
schepen
scheppen
scheren
scherp

schieten
schimmel
schoenen
scholen
schoon
schrijven
schudden
schuiven
schuld
schuren
schuw
simpel
slaan
slapen
slecht
slijm
slijpen
slim
sluiten
smal
smeden
smelten
smeren
sneeuw
sneeuwen
soep
spannen
sparen
spartelen
spelden
spelen
sport*spel*
spreken
springen
spuiten
staan
stallen
stampen
steken
stelen
stenen
sterven
stijf
stil
stoelen
stof*huisvuil*
stokken
stom
stout
straten
strepen
strooien
sturen*zenden*

suiker
taai
taarten
tafels
takken
tam
tanden
tangen
tantes
tarwe
tegen
tellen
temmen
tenen
tien
timmeren
torens
traag
tralies
trams
treffen
treinen
trouwen
tussen
twaalf
twee
tweede
twijfel
twintig
uilen
vader
vallen
vals
vangen
varen
vast
vaten
vechten
veel
veertig
ver
verf
vers
vesten
vet
veulens
vier
vieren
vijf
vijftig
vijgen
vinden
vingers

vissen
vlaggen
vlas
vlees
vliegen
vloeken
vlooien
voegen
voelen
voeten
vogels
vol
volgen
volk
voor
vragen
vreemd
vriezen
vrij
vrijdag
vrijen
vroeg
vuil
vuur
wachten
wafels
warm
wassen
weer
weg
wegen*mv*
wegen*ww*
weinig
weken
wensen
werken
weten
wieden
wijd
wijn
wijven
wild
willen
winnen
wippen
wit
woensdag
wol
wonen
woorden
worden
wrijven
zacht
zakken
zand
zaterdag
zee
zeep
zeggen
zeilen
zeker
zelf
zes
zetten
zeven
zeventig
ziek
ziektes
zien
zijn
zilveren
zitten
zoeken
zoet
zondag
zonder
zonen
zorgen
zout
zouten
zuchten
zuigen
zuur
zwaar
zwart
zwellen
zwemmen
zwijgen

A.2 Appendix 2 - Complete list of “Location”

Aalsmeer NH
Aalten Gl
Aardenburg Ze
Aarlanderveen ZH
Abbekerk NH
Achterberg Ut
Aduard Gn
Afferden Lb
Akkrum Fr
Almen Gl
Almkerk NB
Alphen NB
Ameide ZH
Amersfoort Ut
Angerlo Gl
Anjum Fr - Eanjum Fr
Anloo Dr
Apeldoorn Gl
Appelscha Fr - Appelskea Fr
Appingedam Gn
Arnemuiden Ze
Arum Fr
Austerlitz Ut
Axel Ze
Baarle-Nassau NB
Bakel NB
Bakkeveen Fr - Bakkefean Fr
Balkbrug Ov
Barger-Oosterveld Dr
Barneveld Gl
Bathmen Ov
Beek NB
Beetgumermolen Fr - Bitgummole Fr
Bellingwolde Gn
Benschop Ut
Bergen op Zoom NB
Bergentheim Ov
Berkel-Oud ZH
Bleskensgraaf ZH
Blokzijl Ov
Boekel NB
Bolsward Fr - Boalsert Fr
Borculo Gl
Borkel NB
Borne Ov
Boven-Hardinxveld ZH
Boxtel NB
Breda NB
Bredevoort Gl
Breskens Ze
Brielle ZH
Brouwershaven Ze
Brummen Gl
Brunssum Lb
Budel NB
Buren Fr - Bueren Fr
Burum Fr - Boerum Fr
Buurmalsen Gl
Callantsoog NH
Coevorden Dr
Culemborg Gl
Dalfsen Ov
Dedemsvaart Ov
Deil Gl
Delden Ov
Delft ZH hoog
Delft ZH laag
De Lier ZH
De Lutte Ov
Den Dungen NB
Den Ham Ov
Den Hoorn NH
Den Oever NH
De Rijp NH
Deurne NB
Deventer Ov
Didam Gl
Diemen NH
Diepenveen Ov
Dieren Gl
Dodewaard Gl
Doesburg Gl
Doetinchem Gl
Dokkum Fr
Dongen NB
Doornspijk Gl
Driebruggen ZH
Druten Gl
Dwingeloo Dr
Echten Fr - Ychten Fr
Echt Lb
Edam NH
Ede Gl
Eede Ze
Eelde Dr
Eenrum Gn
Eexterveen Dr
Egmond aan Zee NH
Egmond-Binnen NH
Eibergen Gl
Eijsden Lb
Eindhoven NB

Elburg Gl
Enkhuizen NH
Epen Lb
Fijnaart NB
Finsterwolde Gn
Formerum Fr - Formearum Fr
Franeker Fr - Frjentsjer Fr
Garderen Gl
Gasselte Dr
Geldrop NB
Genemuiden Ov
Gerwen NB
Giesbeek Gl
Giethoorn Ov
Goes Ze
Goirle NB
Gorssel Gl
Gouda ZH
Gouderak ZH
Goudswaard ZH
Grave NB
Grijpskerke Ze
Grijpskerk Gn
Groenekan Ut
Groenlo Gl
Groesbeek Gl
Gronsveld Lb
Grouw Fr - Grou Fr
Grubbenvorst Lb
Gulpen Lb
Haaksbergen Ov
Haamstede Ze
Hallum Fr
Haps NB
Hardenberg Ov
Harderwijk Gl
Harkema Opeinde Fr - De Harkema Fr
Harlingen Fr - Harns Fr
Hasselt Ov
Hattem Gl
Havelte Dr
Heerde Gl
Heerewaarden Gl
Hei- en Boeicop ZH
Heinkenszand Ze
Heino Ov
Hellendoorn Ov
Hellevoetsluis ZH
Helmond NB
Hengelo Gl
Herkingen ZH
Heteren Gl
Hierden Gl
Hilvarenbeek NB
Hindeloopen Fr - Hylpen Fr
Hoedekenskerke Ze
Hoenderloo Gl
Hollandsche Veld Dr
Hollum Fr
Holwerd Fr - Holwert Fr
Hooghalen Dr
Hoonhorst Ov
Hoorn NH
Horn Lb
Houten Ut
Huijbergen NB
Huizen NH
Hunsel Lb
IJlst Fr - Drylts Fr
IJmuiden NH
IJsselmuiden Ov
IJzendijke Ze
Ingen Gl
Joure Fr - De Jouwer Fr
Jubbega Fr - Jobbegea Fr
Jutphaas Ut
Kamerik Ut
Kampen Ov
Kantens Gn
Katwijk aan Zee ZH
Kedichem ZH
Kerkrade Lb
Kessel NB
Klaaswaal ZH
Kloosterzande Ze
Koedijk NH
Koekange Dr
Koewacht Ze
Kortgene Ze
Koudekerk aan den Rijn ZH
Koudum Fr
Kruiningen Ze
Kuinre Ov
Lage Zwaluwe NB
Laren Gl
Laren NH
Leende NB
Leermens Gn
Leeuwarden Fr - Ljouwert Fr
Lekkerkerk ZH
Lemele Ov
Lemmer Fr - De Lemmer Fr
Lent Gl
Lichtenvoorde Gl
Lies Fr
Lochem Gl

Loenen Gl
Loenen Ut
Loon op Zand NB
Maasbree Lb
Maastricht Lb
Makkum Fr
Marken NH
Marrum Fr - Mearum Fr
Marum Gn
Meerssen Lb
Meijel Lb
Meije ZH
Meppel Dr
Middelburg Ze
Middelharnis ZH
Middelie NH
Midsland Fr - Midslan Fr
Midwolda Gn
Monnickendam NH
Mussel Gn
Nederweert Lb
Neede Gl
Nieuw-Schoonebeek Dr
Nij Beets Fr
Nijeholtpade Fr
Nijmegen Gl
Nijverdal Ov
Nistelrode NB
Noordeloos ZH
Noordwolde Fr
Norg Dr
Nunspeet Gl
Oerle NB
Oijen NB
Oirschot NB
Oisterwijk NB
Oldemarkt Ov
Oldenzaal Ov
Onstwedde Gn
Ooltgensplaat ZH
Oosterbeek Gl
Oosterend Fr - Aasterein Fr
Oosterland Ze
Ootmarsum Ov
Opperdoes NH
Ospel Lb
Ossendrecht NB
Otterlo Gl
Ottersum Lb
Oud-Charlois ZH
Ouddorp ZH
Oudega Fr - Aldegea Fr
Oudemirdum Fr - Aldemardum Fr
Oude Pekela Gn
Oudeschoot Fr - Aldskoat Fr
Oude-Tonge ZH prot
Oude-Tonge ZH rk
Oudewater Ut
Oud-Vossemeer Ze
Panningen Lb
Papendrecht ZH
Piershil ZH
Poederoijen Gl
Posterholt Lb
Putten Gl
Puttershoek ZH
Raalte Ov
Reek NB
Renkum Gl
Reusel NB
Reuver Lb
Rhenen Ut
Rijkevoort NB
Rijsbergen NB
Rijssen Ov
Rilland Ze
Rinsumageest Fr - Rinsumageast Fr
Roderwolde Dr
Roermond Lb
Roggel Lb
Roodeschool Gn
Roosendaal NB
Rossum Ov
Roswinkel Dr
Rotterdam ZH
Rottevalle Fr - Rottefalle Fr
Rouveen Ov
Rozendaal Gl
Ruinen Dr
Ruurlo Gl
Schagen NH
Scheemda Gn
Schelluinen ZH
Scherpenzeel Fr
Scheveningen ZH
Schiermonnikoog Fr - Skiermantseach Fr
Schinnen Lb
Schoonebeek Dr
Schoonoord Dr
Sellingen Gn
Serooskerke Ze
Sevenum Lb
Sexbierum Fr - Seisbierrum Fr
s-Heerenberg Gl
s-Heerenbroek Ov
s-Hertogenbosch NB

Silvolde Gl
Sint Annaparochie Fr - Sint Anne Fr
Sint-Annen Gn
Sint Jansteen Ze
Sint Maartensdijk Ze
Sint-Oedenrode NB
Sittard Lb
Slagharen Ov
Sliedrecht ZH
Slochteren Gn
Sloten Fr - Sleat Fr
Smilde Dr
Sneek Fr - Snits Fr
Someren NB
Spakenburg Ut
Spanbroek NH
Spannum Fr
Stadskanaal Gn
Stavoren Fr - Starum Fr
Steenbergen NB
Steenderen Gl
Steenwijk Ov
Stokkum Ov
Stolwijk ZH
Stompwijk ZH
Susteren Lb
Tegelen Lb
Ter Apel Gn
Teuge Gl
Tiel Gl
Tienray Lb
Tietjerk Fr - Tytsjerk Fr
Tilburg NB
Tilligte Ov
Tjalleberd Fr - Tsjalbert Fr
Tubbergen Ov
Ulft Gl
Urk Fl
Ursem NH
Usselo Ov
Utrecht Ut
Vaals Lb
Vaassen Gl
Valthermond Dr
Varsseveld Gl
Veendam Gn
Veenendaal Ut
Veenwouden Fr - Feanwalden Fr
Veghel NB
Velddriel Gl
Venlo Lb
Vessem NB
Vlissingen Ze
Voerendaal Lb
Volendam NH
Vollenhove Ov
Voorst Gl
Voorthuizen Gl
Vorden Gl
Vreeswijk Ut
Vriezenveen Ov
Waardenburg Gl
Wagenborgen Gn
Wanssum Lb
Wapse Dr
Wateringen ZH
Weidum Fr
Werkhoven Ut
Westerbork Dr
Westergeest Fr - Westergeast Fr
Westkapelle Ze
West-Terschelling Fr - West-Skylge Fr
Wierden Ov
Wierum Fr
Wijdenes NH
Wijhe Ov
Wijk bij Duurstede Ut
Wijnjeterp Fr - Wynjewald Fr
Willemstad NB
Wilp Gl
Windesheim Ov
Winterswijk Gl
Wissenkerke Ze
Wolvega Fr
Workum Fr - Warkum Fr
Woudenberg Ut
Woudsend Fr - Waldsein Fr
Wouw NB
Zaandijk NH
Zalk Ov
Zandvoort NH
Zeddam Gl
Zeeland NB
Zegveld Ut
Zelhem Gl
Zetten Gl
Zevenaar Gl
Zevenbergen NB
Zierikzee Ze
Zieuwent Gl
Zoetermeer ZH
Zoutkamp Gn
Zuiddorpe Ze
Zuid-Sleen Dr
Zuidzande Ze
Zundert NB

Zutphen Gl
Zwartsluis Ov
Zwinderen Dr
Zwolle Ov

A.3 Appendix 3 - R script

```
# Chapter 2

rm(list = ls())
library(gamm4)
library(xtable)



# Figure 2.1

par(cex=1.5)
plot(women)




# Figure 2.2

par(cex=1.5)


# option 1: fitting by hand
y=women$weight
x=as.matrix(women$height)
x=cbind(1,x)
b=solve(t(x)%*%x)%*%t(x)%*%y
yhat=x%*%b
plot(x[,2],yhat,lwd=2,type="l",xlab="height", ylab="weight")
points(women)


# option 2: use lm
lmw=lm(women$weight~women$height)
plot(women)
abline(lmw, lwd=2)




# Figure 2.3

x=seq(0,5,length=20)
y=dexp(x+rnorm(20, mean = 0, sd = 0.12),rate=1)
lm1=lm(y~x)
lm2=lm(log(y)~x)

old.par<-par(mfrow=c(1,2))
y1=-0.98507*x+-0.04335
plot(x,exp(y1),lwd=2,type="l",xlab="x", ylab="y")
points(x,y,lwd=2)

plot(x,log(y),lwd=2)
```

```
abline(lm2,lwd=2)




# beetles

load("bee.rda")
print(xtable(bee),include.rownames=FALSE)

alive=bee[,2]-bee[,3]
death=bee[,3]
dose=bee[,1]
phat=death/(death+alive) # calculate the proportion of death
fit <- glm(cbind(death,alive) ~ dose, family=binomial)
summary(fit)




# Figure 2.4

rm(list = ls())
old.par<-par(mfrow=c(1,3))
par(cex=1.5)


randomRows = function(df,n){
  return(df[sample(nrow(df),n),])
}
set.seed(1)
cars=randomRows(cars,25)

speed=cars$speed # Speed is our independent variable
dist=cars$dist # Distance is our dependent variable
x<-speed-min(speed);x<-x/max(x) # scale the domain into [0,1]

# write R(x,z) as a function.
rk<-function(x,z)
{
  ((z-0.5)^2-1/12)*((x-0.5)^2-1/12)/4
  -((abs(x-z)-0.5)^4-(abs(x-z)-0.5)^2/2+7/240)/24
}

# Write a function that makes a model matrix given x and knots(xk)
spl.X<-function(x,xk)
{ q<-length(xk)+2 # number of parameters
  n<-length(x) # number of data
  X<-matrix(1,n,q) # frame for the model matrix
  X[,2]<-x # second column should be x
  X[,3:q]<-outer(x,xk,FUN=rk) # the remaining columns should be R(x,xk)
  X
}
```

```
# 2 knots
xk<-1:2/3 # choose some knots
X<-spl.X(x,xk) # create model matrix
mod.1<-lm(dist~X-1) # fit model
mod.2<-lm(dist~X)
xp<-0:100/100 # x values for prediction
Xp<-spl.X(xp,xk) # prediction matrix
plot(x,dist,xlab="speed",ylab="distance",font.main=1,main="2 knots") # Plot the data
lines(xp,Xp%*%coef(mod.1),lwd=2) # plot fitted spline


# 5 knots
xk<-1:5/6 # choose some knots
X<-spl.X(x,xk) # create model matrix
mod.1<-lm(dist~X-1) # fit model
mod.2<-lm(dist~X)
xp<-0:100/100 # x values for prediction
Xp<-spl.X(xp,xk) # prediction matrix
plot(x,dist,xlab="speed",ylab="distance",font.main=1,main="5 knots") # Plot the data
lines(xp,Xp%*%coef(mod.1),lwd=2) # plot fitted spline


# 10 knots
xk<-1:10/11 # choose some knots
X<-spl.X(x,xk) # create model matrix
mod.1<-lm(dist~X-1) # fit model
mod.2<-lm(dist~X)
xp<-0:100/100 # x values for prediction
Xp<-spl.X(xp,xk) # prediction matrix
plot(x,dist,xlab="speed",ylab="distance",font.main=1,main="10 knots") # Plot the data
lines(xp,Xp%*%coef(mod.1),lwd=2) # plot fitted spline




# Figure 2.5

# figure 2.4 should be run first
xp=0:100/100

# A function that generates matrix S
spl.S<-function(xk)
{ q<-length(xk)+2;S<-matrix(0,q,q) # make a frame
  S[3:q,3:q]<-outer(xk,xk,FUN=rk) # fill in non-zero part
  S
}

mat.sqrt<-function(S) # A function that takes matrix square root
{ d<-eigen(S,symmetric=TRUE)
  rS<-d$vectors%*%diag(d$values^0.5)%*%t(d$vectors)
}
```

```
# A function that fits penalized regression spline
prs.fit<-function(y,x,xk,lambda)
{ q<-length(xk)+2 # dimension of basis
  n<-length(x) # number of data
  Xa <- rbind(spl.X(x,xk),mat.sqrt(spl.S(xk))*sqrt(lambda))
  # write matrices according to (2.5)
  y[(n+1):(n+q)]<-0
  lm(y~Xa-1) # fit the model
}


# lambda=0.000001
xk<-1:10/11
mod.2<-prs.fit(dist,x,xk,10^(-6))
Xp<-spl.X(xp,xk)
plot(x,dist,xlab="speed",ylab="distance",font.main=1,main=expression(paste(lambda,"=0.000001")))
lines(xp,Xp%*%coef(mod.2),lwd=2)

# lambda=0.0001
xk<-1:10/11
mod.2<-prs.fit(dist,x,xk,10^(-4))
Xp<-spl.X(xp,xk)
plot(x,dist,xlab="speed",ylab="distance",font.main=1,main=expression(paste(lambda,"=0.0001")))
lines(xp,Xp%*%coef(mod.2),lwd=2)

# lambda=0.01
xk<-1:10/11
mod.2<-prs.fit(dist,x,xk,10^(-1))
Xp<-spl.X(xp,xk)
plot(x,dist,xlab="speed",ylab="distance",font.main=1,main=expression(paste(lambda,"=0.01")))
lines(xp,Xp%*%coef(mod.2),lwd=2)




# Figure 2.6

# figure 2.4 and 2.5 should be run first
old.par<-par(mfrow=c(1,2))
par(cex=1)


lambda=10^(-7);n=length(dist);V=0
for (i in 1:50) {
  mod=prs.fit(dist,x,xk,lambda)
  trA=sum(influence(mod)$hat[1:n])
  rss=sum((dist-fitted(mod)[1:n])^2)
  V[i]=n*rss/(n-trA)^2
  lambda=lambda*1.5
}

plot(1:50,V,type="l",lwd=2,font.main=1,main="GCV score",xlab="i",ylab=expression(paste(nu))) # plot
```

```
i=(1:50)[V==min(V)] # extract index of min(V)
mod.3<-prs.fit(dist,x,xk,1.5^(i-1)*10^-7) # fit model with optimal lambda
Xp<-spl.X(xp,xk)
plot(x,dist,font.main=1,main=expression(paste("optimal fit" (lambda,"=1.5^17*10^-
7"))),xlab="speed",ylab="distance")
lines(xp,Xp%*%coef(mod.3),lwd=2)




# Table 2.1

load("rabbit.rda")
print(xtable(rabbit),include.rownames=FALSE)


# End chapter 2
```

```
# Chapter 3

library(gamm4)
library(xtable)
library(ggplot2)
load('wrddst.rda')
d=wrddst



# Figure 3.1

load('dialectNL.rda')

# WordCategory=all
d=dialectNL
loc.mat=unique(d[,c("Location","Longitude","Latitude")])

# take the mean of PronDistStdDutch.c for each city.
loc.mat=cbind(loc.mat,0)
colnames(loc.mat)[4] <- "PronDistStdDutch.c"
for(i in 1:nrow(loc.mat)) {
  name=loc.mat[i,"Location"]
  meanmtr=d[which(d$Location==name),]
  loc.mat[i,4]=mean(meanmtr$PronDistStdDutch.c)
}
cat.all=loc.mat

# resolution: 850 x 800
all<-ggplot(cat.all,aes(x=Longitude,y=Latitude,z=PronDistStdDutch.c))
all+stat_summary2d()+scale_fill_gradientn(limits=c(-
0.3325205,0.3487294),colours=c("black","white"))+theme_bw()+ggtitle("WordCategory = all")



# Figure 3.2

library(gridExtra)
load('dialectNL.rda')

# WordCategory=adjective
d=dialectNL
d=dA=subset(d,WordCategory=="A")
loc.mat=unique(d[,c("Location","Longitude","Latitude")])

loc.mat=cbind(loc.mat,0)
colnames(loc.mat)[4] <- "PronDistStdDutch.c"
for(i in 1:nrow(loc.mat)) {
  name=loc.mat[i,"Location"]
  meanmtr=d[which(d$Location==name),]
  loc.mat[i,4]=mean(meanmtr$PronDistStdDutch.c)
```

```
}
cat.a=loc.mat


# WordCategory=adverb
d=dialectNL
d=dB=subset(d,WordCategory=="B")
loc.mat=unique(d[,c("Location","Longitude","Latitude")])

loc.mat=cbind(loc.mat,0)
colnames(loc.mat)[4] <- "PronDistStdDutch.c"
for(i in 1:nrow(loc.mat)) {
  name=loc.mat[i,"Location"]
  meanmtr=d[which(d$Location==name),]
  loc.mat[i,4]=mean(meanmtr$PronDistStdDutch.c)
}
cat.b=loc.mat


# WordCategory=verb
d=dialectNL
d=dV=subset(d,WordCategory=="V")
loc.mat=unique(d[,c("Location","Longitude","Latitude")])

loc.mat=cbind(loc.mat,0)
colnames(loc.mat)[4] <- "PronDistStdDutch.c"
for(i in 1:nrow(loc.mat)) {
  name=loc.mat[i,"Location"]
  meanmtr=d[which(d$Location==name),]
  loc.mat[i,4]=mean(meanmtr$PronDistStdDutch.c)
}
cat.v=loc.mat


# WordCategory=noun
d=dialectNL
d=dN=subset(d,WordCategory=="N")
loc.mat=unique(d[,c("Location","Longitude","Latitude")])

loc.mat=cbind(loc.mat,0)
colnames(loc.mat)[4] <- "PronDistStdDutch.c"
for(i in 1:nrow(loc.mat)) {
  name=loc.mat[i,"Location"]
  meanmtr=d[which(d$Location==name),]
  loc.mat[i,4]=mean(meanmtr$PronDistStdDutch.c)
}
cat.n=loc.mat


a<-
ggplot(cat.a,aes(x=Longitude,y=Latitude,z=PronDistStdDutch.c))+stat_summary2d()+scale_fill_gradie
ntn(limits=c(-0.3325205,0.3487294),colours=c("black","white"))+
  theme_bw()+ggtitle("WordCategory = adjective (20.8%)")
```

```
b<-
ggplot(cat.b,aes(x=Longitude,y=Latitude,z=PronDistStdDutch.c))+stat_summary2d()+scale_fill_gradie
ntn(limits=c(-0.3325205,0.3487294),colours=c("black","white"))+
  theme_bw()+ggtitle("WordCategory = adverb (8.2%)")

v<-
ggplot(cat.v,aes(x=Longitude,y=Latitude,z=PronDistStdDutch.c))+stat_summary2d()+scale_fill_gradie
ntn(limits=c(-0.3325205,0.3487294),colours=c("black","white"))+
  theme_bw()+ggtitle("WordCategory = verb (30.7%)")

n<-
ggplot(cat.n,aes(x=Longitude,y=Latitude,z=PronDistStdDutch.c))+stat_summary2d()+scale_fill_gradie
ntn(limits=c(-0.3325205,0.3487294),colours=c("black","white"))+
  theme_bw()+ggtitle("WordCategory = noun (40.3%)")

# plot (resolution: 1100 x 900)
grid.arrange(a,b,v,n,ncol=2)




# Table 3.1

rm(list = ls())
load('wrddst.rda')
d=wrddst
distref=unique(d[,c("Placename","GeoX","GeoY")])

# make a frame
center.mat=distref[,c("Placename"),0]
center.mat=cbind(center.mat,0,0,0)
colnames(center.mat)[2] = "adj.r.squared"
colnames(center.mat)[3] = "(Intercept)"
colnames(center.mat)[4] = "work.cent$Dist"

for(i in 1:nrow(distref)) {
  name=distref[i,"Placename"]

  center=distref[(distref$Placename==name),]
  cenX=center[,"GeoX"]
  cenY=center[,"GeoY"]
  work.cent=cbind(distref,(distref$GeoX-cenX),(distref$GeoY-cenY))
  colnames(work.cent)[4] <- "NGeoX"
  colnames(work.cent)[5] <- "NGeoY"
  work.cent=cbind(work.cent,sqrt((work.cent[,"NGeoX"])^2+work.cent[,"NGeoY"]^2))
  colnames(work.cent)[6] <- "Dist"
  work.cent=work.cent[,c("Placename","Dist","NGeoX","NGeoY","GeoX","GeoY")]
  work.cent=cbind(work.cent,0)
  colnames(work.cent)[7] <- "AvgRefPMIdistMeanLog.c"

  for(ii in 1:nrow(distref)) {
    name=distref[ii,"Placename"]
```

```
    meanmtr=d[which(d$Placename==name),]
    work.cent[ii,7]=mean(meanmtr$RefPMIdistMeanLog.c)
  }

  res=lm(work.cent$AvgRefPMIdistMeanLog.c~work.cent$Dist)
  center.mat[i,"adj.r.squared"]=summary(res)$adj.r.squared
  center.mat[i,"(Intercept)"]=coef(res)["(Intercept)"]
  center.mat[i,"work.cent$Dist"]=coef(res)["work.cent$Dist"]
}

center.mat=center.mat[order(center.mat$adj.r.squared,decreasing=TRUE),]
head(center.mat)




# Figure 3.3
rm(list = ls())
load('wrddst.rda')
d=wrddst

old.par<-par(mfrow=c(2,2));par(cex=1.5)

# Rank 1: Driebruggen ZH
center=unique(d[(d$Placename=="Driebruggen ZH"),])

cenX=unique(center[,"GeoX"])
cenY=unique(center[,"GeoY"])

distref=unique(d[,c("Placename","GeoX","GeoY")])
distref=cbind(distref,(distref$GeoX-cenX),(distref$GeoY-cenY))
colnames(distref)[4] <- "NGeoX"
colnames(distref)[5] <- "NGeoY"

distref=cbind(distref,sqrt((distref[,"NGeoX"])^2+distref[,"NGeoY"]^2))
colnames(distref)[6] <- "Dist"

distref = distref[,c("Placename","Dist","NGeoX","NGeoY","GeoX","GeoY")]

# take mean of AvgRefPMIdistMeanLog.c for each city.
distref=cbind(distref,0)
for(i in 1:nrow(distref)) {
  name=distref[i,"Placename"]
  meanmtr=d[which(d$Placename==name),]
  distref[i,7]=mean(meanmtr$RefPMIdistMeanLog.c)
}
colnames(distref)[7] <- "AvgRefPMIdistMeanLog.c"

# Take linear regression
res=lm(distref$AvgRefPMIdistMeanLog.c~distref$Dist)

# Draw scatterplot
x = distref$Dist
```

```
y = distref$AvgRefPMIdistMeanLog.c
plot(x, y,xlab="Distance from center", ylab="PronDistStdDutch",font.main=1,main="Center =
Driebruggen ZH")
abline(res,lwd=2)


# Rank 2: Meije ZH
center=unique(d[(d$Placename=="Meije ZH"),])

cenX=unique(center[,"GeoX"])
cenY=unique(center[,"GeoY"])

distref=unique(d[,c("Placename","GeoX","GeoY")])
distref=cbind(distref,(distref$GeoX-cenX),(distref$GeoY-cenY))
colnames(distref)[4] <- "NGeoX"
colnames(distref)[5] <- "NGeoY"

distref=cbind(distref,sqrt((distref[,"NGeoX"])^2+distref[,"NGeoY"]^2))
colnames(distref)[6] <- "Dist"

distref = distref[,c("Placename","Dist","NGeoX","NGeoY","GeoX","GeoY")]

# take mean of AvgRefPMIdistMeanLog.c for each city.
distref=cbind(distref,0)
for(i in 1:nrow(distref)) {
  name=distref[i,"Placename"]
  meanmtr=d[which(d$Placename==name),]
  distref[i,7]=mean(meanmtr$RefPMIdistMeanLog.c)
}
colnames(distref)[7] <- "AvgRefPMIdistMeanLog.c"

# Take linear regression
res=lm(distref$AvgRefPMIdistMeanLog.c~distref$Dist)

# Draw scatterplot
x = distref$Dist
y = distref$AvgRefPMIdistMeanLog.c
plot(x, y,xlab="Distance from center", ylab="PronDistStdDutch",font.main=1,main="Center = Meije ZH")
abline(res,lwd=2)


# Rank 3: Aarlanderveen ZH
center=unique(d[(d$Placename=="Aarlanderveen ZH"),])

cenX=unique(center[,"GeoX"])
cenY=unique(center[,"GeoY"])

distref=unique(d[,c("Placename","GeoX","GeoY")])
distref=cbind(distref,(distref$GeoX-cenX),(distref$GeoY-cenY))
colnames(distref)[4] <- "NGeoX"
colnames(distref)[5] <- "NGeoY"

distref=cbind(distref,sqrt((distref[,"NGeoX"])^2+distref[,"NGeoY"]^2))
```

```
colnames(distref)[6] <- "Dist"

distref = distref[,c("Placename","Dist","NGeoX","NGeoY","GeoX","GeoY")]

# take mean of AvgRefPMIdistMeanLog.c for each city.
distref=cbind(distref,0)
for(i in 1:nrow(distref)) {
  name=distref[i,"Placename"]
  meanmtr=d[which(d$Placename==name),]
  distref[i,7]=mean(meanmtr$RefPMIdistMeanLog.c)
}
colnames(distref)[7] <- "AvgRefPMIdistMeanLog.c"

# Take linear regression
res=lm(distref$AvgRefPMIdistMeanLog.c~distref$Dist)

# Draw scatterplot
x = distref$Dist
y = distref$AvgRefPMIdistMeanLog.c
plot(x, y,xlab="Distance from center", ylab="PronDistStdDutch",font.main=1,main="Center =
Aarlanderveen ZH")
abline(res,lwd=2)


# Rank 4: Gouda ZH
center=unique(d[(d$Placename=="Gouda ZH"),])

cenX=unique(center[,"GeoX"])
cenY=unique(center[,"GeoY"])

distref=unique(d[,c("Placename","GeoX","GeoY")])
distref=cbind(distref,(distref$GeoX-cenX),(distref$GeoY-cenY))
colnames(distref)[4] <- "NGeoX"
colnames(distref)[5] <- "NGeoY"

distref=cbind(distref,sqrt((distref[,"NGeoX"])^2+distref[,"NGeoY"]^2))
colnames(distref)[6] <- "Dist"

distref = distref[,c("Placename","Dist","NGeoX","NGeoY","GeoX","GeoY")]

# take mean of AvgRefPMIdistMeanLog.c for each city.
distref=cbind(distref,0)
for(i in 1:nrow(distref)) {
  name=distref[i,"Placename"]
  meanmtr=d[which(d$Placename==name),]
  distref[i,7]=mean(meanmtr$RefPMIdistMeanLog.c)
}
colnames(distref)[7] <- "AvgRefPMIdistMeanLog.c"

# Take linear regression
res=lm(distref$AvgRefPMIdistMeanLog.c~distref$Dist)

# Draw scatterplot
```

```
x = distref$Dist
y = distref$AvgRefPMIdistMeanLog.c
plot(x, y,xlab="Distance from center", ylab="PronDistStdDutch",font.main=1,main="Center = Gouda
ZH")
abline(res,lwd=2)




# Table 3.2

# WordCat==A

rm(list = ls())
load('wrddst.rda')
d=wrddst
d=dA=subset(d, WordCat=="A")

distref=unique(d[,c("Placename","GeoX","GeoY")])

# make a frame
center.mat=distref[,c("Placename"),0]
center.mat=cbind(center.mat,0,0,0)
colnames(center.mat)[2] = "adj.r.squared"
colnames(center.mat)[3] = "(Intercept)"
colnames(center.mat)[4] = "work.cent$Dist"

for(i in 1:nrow(distref)) {
  name=distref[i,"Placename"]

  center=distref[(distref$Placename==name),]
  cenX=center[,"GeoX"]
  cenY=center[,"GeoY"]
  work.cent=cbind(distref,(distref$GeoX-cenX),(distref$GeoY-cenY))
  colnames(work.cent)[4] <- "NGeoX"
  colnames(work.cent)[5] <- "NGeoY"
  work.cent=cbind(work.cent,sqrt((work.cent[,"NGeoX"])^2+work.cent[,"NGeoY"]^2))
  colnames(work.cent)[6] <- "Dist"
  work.cent=work.cent[,c("Placename","Dist","NGeoX","NGeoY","GeoX","GeoY")]
  work.cent=cbind(work.cent,0)
  colnames(work.cent)[7] <- "AvgRefPMIdistMeanLog.c"

  for(ii in 1:nrow(distref)) {
    name=distref[ii,"Placename"]
    meanmtr=d[which(d$Placename==name),]
    work.cent[ii,7]=mean(meanmtr$RefPMIdistMeanLog.c)
  }

  res=lm(work.cent$AvgRefPMIdistMeanLog.c~work.cent$Dist)
  center.mat[i,"adj.r.squared"]=summary(res)$adj.r.squared
  center.mat[i,"(Intercept)"]=coef(res)["(Intercept)"]
  center.mat[i,"work.cent$Dist"]=coef(res)["work.cent$Dist"]
}
```

```
center.mat.a=center.mat[order(center.mat$adj.r.squared,decreasing=TRUE),]
head(center.mat.a)


# WordCat==B

rm(list = ls())
load('wrddst.rda')
d=wrddst
d=dB=subset(d, WordCat=="B")

distref=unique(d[,c("Placename","GeoX","GeoY")])

# make a frame
center.mat=distref[,c("Placename"),0]
center.mat=cbind(center.mat,0,0,0)
colnames(center.mat)[2] = "adj.r.squared"
colnames(center.mat)[3] = "(Intercept)"
colnames(center.mat)[4] = "work.cent$Dist"

for(i in 1:nrow(distref)) {
  name=distref[i,"Placename"]

  center=distref[(distref$Placename==name),]
  cenX=center[,"GeoX"]
  cenY=center[,"GeoY"]
  work.cent=cbind(distref,(distref$GeoX-cenX),(distref$GeoY-cenY))
  colnames(work.cent)[4] <- "NGeoX"
  colnames(work.cent)[5] <- "NGeoY"
  work.cent=cbind(work.cent,sqrt((work.cent[,"NGeoX"])^2+work.cent[,"NGeoY"]^2))
  colnames(work.cent)[6] <- "Dist"
  work.cent=work.cent[,c("Placename","Dist","NGeoX","NGeoY","GeoX","GeoY")]
  work.cent=cbind(work.cent,0)
  colnames(work.cent)[7] <- "AvgRefPMIdistMeanLog.c"

  for(ii in 1:nrow(distref)) {
    name=distref[ii,"Placename"]
    meanmtr=d[which(d$Placename==name),]
    work.cent[ii,7]=mean(meanmtr$RefPMIdistMeanLog.c)
  }

  res=lm(work.cent$AvgRefPMIdistMeanLog.c~work.cent$Dist)
  center.mat[i,"adj.r.squared"]=summary(res)$adj.r.squared
  center.mat[i,"(Intercept)"]=coef(res)["(Intercept)"]
  center.mat[i,"work.cent$Dist"]=coef(res)["work.cent$Dist"]
}

center.mat.b=center.mat[order(center.mat$adj.r.squared,decreasing=TRUE),]
head(center.mat.b)


# WordCat==V
```

```
rm(list = ls())
load('wrddst.rda')
d=wrddst
d=dV=subset(d, WordCat=="V")

distref=unique(d[,c("Placename","GeoX","GeoY")])

# make a frame
center.mat=distref[,c("Placename"),0]
center.mat=cbind(center.mat,0,0,0)
colnames(center.mat)[2] = "adj.r.squared"
colnames(center.mat)[3] = "(Intercept)"
colnames(center.mat)[4] = "work.cent$Dist"

for(i in 1:nrow(distref)) {
  name=distref[i,"Placename"]

  center=distref[(distref$Placename==name),]
  cenX=center[,"GeoX"]
  cenY=center[,"GeoY"]
  work.cent=cbind(distref,(distref$GeoX-cenX),(distref$GeoY-cenY))
  colnames(work.cent)[4] <- "NGeoX"
  colnames(work.cent)[5] <- "NGeoY"
  work.cent=cbind(work.cent,sqrt((work.cent[,"NGeoX"])^2+work.cent[,"NGeoY"]^2))
  colnames(work.cent)[6] <- "Dist"
  work.cent=work.cent[,c("Placename","Dist","NGeoX","NGeoY","GeoX","GeoY")]
  work.cent=cbind(work.cent,0)
  colnames(work.cent)[7] <- "AvgRefPMIdistMeanLog.c"

  for(ii in 1:nrow(distref)) {
    name=distref[ii,"Placename"]
    meanmtr=d[which(d$Placename==name),]
    work.cent[ii,7]=mean(meanmtr$RefPMIdistMeanLog.c)
  }

  res=lm(work.cent$AvgRefPMIdistMeanLog.c~work.cent$Dist)
  center.mat[i,"adj.r.squared"]=summary(res)$adj.r.squared
  center.mat[i,"(Intercept)"]=coef(res)["(Intercept)"]
  center.mat[i,"work.cent$Dist"]=coef(res)["work.cent$Dist"]
}

center.mat.v=center.mat[order(center.mat$adj.r.squared,decreasing=TRUE),]
head(center.mat.v)


# WordCat==N

rm(list = ls())
load('wrddst.rda')
d=wrddst
d=dN=subset(d, WordCat=="N")
```

```
distref=unique(d[,c("Placename","GeoX","GeoY")])

# make a frame
center.mat=distref[,c("Placename"),0]
center.mat=cbind(center.mat,0,0,0)
colnames(center.mat)[2] = "adj.r.squared"
colnames(center.mat)[3] = "(Intercept)"
colnames(center.mat)[4] = "work.cent$Dist"

for(i in 1:nrow(distref)) {
  name=distref[i,"Placename"]

  center=distref[(distref$Placename==name),]
  cenX=center[,"GeoX"]
  cenY=center[,"GeoY"]
  work.cent=cbind(distref,(distref$GeoX-cenX),(distref$GeoY-cenY))
  colnames(work.cent)[4] <- "NGeoX"
  colnames(work.cent)[5] <- "NGeoY"
  work.cent=cbind(work.cent,sqrt((work.cent[,"NGeoX"])^2+work.cent[,"NGeoY"]^2))
  colnames(work.cent)[6] <- "Dist"
  work.cent=work.cent[,c("Placename","Dist","NGeoX","NGeoY","GeoX","GeoY")]
  work.cent=cbind(work.cent,0)
  colnames(work.cent)[7] <- "AvgRefPMIdistMeanLog.c"

  for(ii in 1:nrow(distref)) {
    name=distref[ii,"Placename"]
    meanmtr=d[which(d$Placename==name),]
    work.cent[ii,7]=mean(meanmtr$RefPMIdistMeanLog.c)
  }

  res=lm(work.cent$AvgRefPMIdistMeanLog.c~work.cent$Dist)
  center.mat[i,"adj.r.squared"]=summary(res)$adj.r.squared
  center.mat[i,"(Intercept)"]=coef(res)["(Intercept)"]
  center.mat[i,"work.cent$Dist"]=coef(res)["work.cent$Dist"]
}

center.mat.n=center.mat[order(center.mat$adj.r.squared,decreasing=TRUE),]
head(center.mat.n)




# Figure 3.4

rm(list = ls())
load('wrddst.rda')
d=wrddst

old.par<-par(mfrow=c(2,2))
par(cex=1.5)


# WordCat=A
```

```
d=dA=subset(dA, WordCat=="A")

# WordCat=A, Center = Aalsmeer NH
center=unique(d[(d$Placename=="Aalsmeer NH"),])

cenX=unique(center[,"GeoX"])
cenY=unique(center[,"GeoY"])

distref=unique(d[,c("Placename","GeoX","GeoY")])
distref=cbind(distref,(distref$GeoX-cenX),(distref$GeoY-cenY))
colnames(distref)[4] <- "NGeoX"
colnames(distref)[5] <- "NGeoY"

distref=cbind(distref,sqrt((distref[,"NGeoX"])^2+distref[,"NGeoY"]^2))
colnames(distref)[6] <- "Dist"

distref = distref[,c("Placename","Dist","NGeoX","NGeoY","GeoX","GeoY")]

# take mean of AvgRefPMIdistMeanLog.c for each city.
distref=cbind(distref,0)
for(i in 1:nrow(distref)) {
  name=distref[i,"Placename"]
  meanmtr=d[which(d$Placename==name),]
  distref[i,7]=mean(meanmtr$RefPMIdistMeanLog.c)
}
colnames(distref)[7] <- "AvgRefPMIdistMeanLog.c"

# Take linear regression
res=lm(distref$AvgRefPMIdistMeanLog.c~distref$Dist)

# Draw scatterplot
x = distref$Dist
y = distref$AvgRefPMIdistMeanLog.c
#plot(x, y,xlab="Distance from center", ylab="PronDistStdDutch",font.main=1,main="Word
category=adjective \n Center = Aalsmeer NH")
plot(x, y,xlab="Distance from center", ylab="PronDistStdDutch",ylim=c(-0.4,0.4), xlim=c(-0.02,2.62),
    font.main=1,main="Word category=adjective \n Center = Aalsmeer NH")
abline(res,lwd=2)


# WordCat=B
d=wrddst
d=dB=subset(d, WordCat=="B")

# WordCat=B, Center = Bleskensgraaf ZH
center=unique(d[(d$Placename=="Bleskensgraaf ZH"),])

cenX=unique(center[,"GeoX"])
cenY=unique(center[,"GeoY"])

distref=unique(d[,c("Placename","GeoX","GeoY")])
distref=cbind(distref,(distref$GeoX-cenX),(distref$GeoY-cenY))
colnames(distref)[4] <- "NGeoX"
```

```
colnames(distref)[5] <- "NGeoY"

distref=cbind(distref,sqrt((distref[,"NGeoX"])^2+distref[,"NGeoY"]^2))
colnames(distref)[6] <- "Dist"

distref = distref[,c("Placename","Dist","NGeoX","NGeoY","GeoX","GeoY")]

# take mean of AvgRefPMIdistMeanLog.c for each city.
distref=cbind(distref,0)
for(i in 1:nrow(distref)) {
  name=distref[i,"Placename"]
  meanmtr=d[which(d$Placename==name),]
  distref[i,7]=mean(meanmtr$RefPMIdistMeanLog.c)
}
colnames(distref)[7] <- "AvgRefPMIdistMeanLog.c"

# Take linear regression
res=lm(distref$AvgRefPMIdistMeanLog.c~distref$Dist)

# Draw scatterplot
x = distref$Dist
y = distref$AvgRefPMIdistMeanLog.c
plot(x, y,xlab="Distance from center", ylab="PronDistStdDutch",ylim=c(-0.4,0.4), xlim=c(-0.02,2.62),
     font.main=1,main="Word category=adverb \n Center = Bleskensgraaf ZH")
abline(res,lwd=2)


# WordCat=V
d=wrddst
d=dV=subset(d, WordCat=="V")

# WordCat=V, Center = Stolwijk ZH
center=unique(d[(d$Placename=="Stolwijk ZH"),])

cenX=unique(center[,"GeoX"])
cenY=unique(center[,"GeoY"])

distref=unique(d[,c("Placename","GeoX","GeoY")])
distref=cbind(distref,(distref$GeoX-cenX),(distref$GeoY-cenY))
colnames(distref)[4] <- "NGeoX"
colnames(distref)[5] <- "NGeoY"

distref=cbind(distref,sqrt((distref[,"NGeoX"])^2+distref[,"NGeoY"]^2))
colnames(distref)[6] <- "Dist"

distref = distref[,c("Placename","Dist","NGeoX","NGeoY","GeoX","GeoY")]

# take mean of AvgRefPMIdistMeanLog.c for each city.
distref=cbind(distref,0)
for(i in 1:nrow(distref)) {
  name=distref[i,"Placename"]
  meanmtr=d[which(d$Placename==name),]
  distref[i,7]=mean(meanmtr$RefPMIdistMeanLog.c)
```

```
}
colnames(distref)[7] <- "AvgRefPMIdistMeanLog.c"

# Take linear regression
res=lm(distref$AvgRefPMIdistMeanLog.c~distref$Dist)

# Draw scatterplot
x = distref$Dist
y = distref$AvgRefPMIdistMeanLog.c
plot(x, y,xlab="Distance from center", ylab="PronDistStdDutch",ylim=c(-0.4,0.4), xlim=c(-0.02,2.62),
    font.main=1,main="Word category=verb \n Center = Stolwijk ZH")
abline(res,lwd=2)


# WordCat=N
d=wrddst
d=dN=subset(d, WordCat=="N")

# WordCat=N, Center = Driebruggen ZH
center=unique(d[(d$Placename=="Driebruggen ZH"),])

cenX=unique(center[,"GeoX"])
cenY=unique(center[,"GeoY"])

distref=unique(d[,c("Placename","GeoX","GeoY")])
distref=cbind(distref,(distref$GeoX-cenX),(distref$GeoY-cenY))
colnames(distref)[4] <- "NGeoX"
colnames(distref)[5] <- "NGeoY"

distref=cbind(distref,sqrt((distref[,"NGeoX"])^2+distref[,"NGeoY"]^2))
colnames(distref)[6] <- "Dist"

distref = distref[,c("Placename","Dist","NGeoX","NGeoY","GeoX","GeoY")]

# take mean of AvgRefPMIdistMeanLog.c for each city.
distref=cbind(distref,0)
for(i in 1:nrow(distref)) {
  name=distref[i,"Placename"]
  meanmtr=d[which(d$Placename==name),]
  distref[i,7]=mean(meanmtr$RefPMIdistMeanLog.c)
}
colnames(distref)[7] <- "AvgRefPMIdistMeanLog.c"

# Take linear regression
res=lm(distref$AvgRefPMIdistMeanLog.c~distref$Dist)

# Draw scatterplot
x = distref$Dist
y = distref$AvgRefPMIdistMeanLog.c
plot(x, y,xlab="Distance from center", ylab="PronDistStdDutch",ylim=c(-0.4,0.4), xlim=c(-0.02,2.62),
    font.main=1,main="Word category=noun \n Center = Driebruggen ZH")
abline(res,lwd=2)
```

```
# Table 3.3

rm(list = ls())
load('wrddst.rda')
d=wrddst

age=unique(d[,c("Placename","PopAge")])
age=age[order(age$PopAge),]

income=unique(d[,c("Placename","PopAvgIncomeLog")])
income=income[order(income$PopAvgIncomeLog),]




# Figure 3.5

rm(list = ls())
load('wrddst.rda')
d=wrddst

age=unique(d[,c("Placename","PopAge")])
age=age[order(age$PopAge),]

income=unique(d[,c("Placename","PopAvgIncomeLog")])
income=income[order(income$PopAvgIncomeLog),]

x=age[order(age$Placename),]
y=income[order(income$Placename),]
aim=cbind(x,y$PopAvgIncomeLog)
colnames(aim)[3]="PopAvgIncomeLog"

ailm=lm(aim$PopAvgIncomeLog~aim$PopAge)

x=aim$PopAge
y=aim$PopAvgIncomeLog
par(cex=1.5)
plot(x, y,xlab="PopAvgAge", ylab="PopAvgIncome.log",font.main=1,main="correlation = 0.4396987")
abline(ailm,lwd=2)

cor(x,y)




# Section 3.2.1

gamm.model=gamm(PronDistStdDutch.c~te(Longitude,Latitude)+s(PopSize.log_residGeo)+
s(PopAvgAge_residPopAvgIncome.log_Geo)+s(PopAvgIncome.log_residGeo)+
s(WordFreq.log)+s(WordVCratio.log.z)+s(WordIsNounOrAdverb),
```

```
random=list(Word=~1,Location=~1,Transcriber=~1),data=dialectNL)



# Section 3.2.2

bam.model=bam(PronDistStdDutch.c~te(Longitude,Latitude)+
s(PopSize.log_residGeo)+s(PopAvgAge_residPopAvgIncome.log_Geo)+
s(PopAvgIncome.log_residGeo)+s(WordFreq.log)+s(WordVCratio.log.z)+
s(WordIsNounOrAdverb)+
s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re"),data=dialectNL)


load('dialectNL.rda')
d=dialectNL
remove(dialectNL)
d$WordIsVerb=as.factor(d$WordIsVerb)


index.maker=function(size,n,dset)
{
 index.mat=matrix(0,nrow=size,ncol=n)
 for(i in 1:n) {
   index.mat[,i]=sample.int(nrow(dset),size)
 }
 index.mat
}


vinnie.bam.i=function(formula,size,nrep,index,dset)
{
 # Make a frame
 sm=matrix(0,nrep,1);colnames(sm)=c("fREML")
 score.summ=matrix(0,1,2);colnames(score.summ)=c("mean","variance")

 for(i in 1:nrep) {
   # step 1: use index
   dat=dset[index[,i],]

   # step2: bam
   bam.model=bam(formula,data=dat)

   # step3: fREML score
   sm[i,1]=bam.model$gcv.ubre
 }
 # Calculate mean and variance of fREML scores
 score.summ[1,1]=mean(sm[,1])
 score.summ[1,2]=var(sm[,1])
 score.summ
}
```

```
model.selection.i=function(flist,size,nrep,index,dset)
{
 n=length(flist)
 score.mat=matrix(0,nrow=n,ncol=3);colnames(score.mat)=c("mean","variance","model")
 score.mat[,3]=c(1:n)
 for(i in 1:n) {
  cat("busy with funtion nr.", i, "\n")
  score.mat[i,1:2]=vinnie.bam.i(flist[[i]],size,nrep,index,dset)
 }
 score.mat=score.mat[order(score.mat[,1]),]
 score.mat
}


vinnie.gamm1=function(formula,rformula,size,index,dset)
{

 # step 1: Take a random subset
 dat=dset[index,]

 # step2: gamm
 gamm.model=gamm(formula,random=rformula,data=dat)

 y=dset$PronDistStdDutch.c
 y.hat=predict.gam(gamm.model$gam,dset)
 delta.adj=mean((y-y.hat)^2)

 # step4: adjusted delta
 delta.adj
}



# Table 3.4

# models

# gamm version
f=list()
fr=list()
f[[1]]=PronDistStdDutch.c~s(PopSize.log)+s(PopAvgAge)+s(PopAvgIncome.log)+s(WordFreq.log)+s(
WordVCratio.log.z)
fr[[1]]=list(Word=~1,Location=~1,Transcriber=~1)

f[[2]]=PronDistStdDutch.c~s(PopSize.log)+s(PopAvgAge)+s(PopAvgIncome.log)+s(WordFreq.log)+s(
WordVCratio.log.z)
fr[[2]]=list(Location=~1,Transcriber=~1)

f[[3]]=PronDistStdDutch.c
~s(PopSize.log)+s(PopAvgAge)+s(PopAvgIncome.log)+s(WordFreq.log)+s(WordVCratio.log.z)
fr[[3]]=list(Word=~1,Transcriber=~1)
```

```
f[[4]]=PronDistStdDutch.c~s(PopAvgAge)+s(PopAvgIncome.log)+s(WordFreq.log)+s(WordVCratio.log.
z)
fr[[4]]=list(Word=~1,Location=~1,Transcriber=~1)
fun.list.gamm=f
ran.list.gamm=fr
remove(f,fr)

# bam version
f=list()
f[[1]]=PronDistStdDutch.c~s(PopSize.log)+s(PopAvgAge)+s(PopAvgIncome.log)+s(WordFreq.log)+s(
WordVCratio.log.z)+
  s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re")

f[[2]]=PronDistStdDutch.c~s(PopSize.log)+s(PopAvgAge)+s(PopAvgIncome.log)+s(WordFreq.log)+s(
WordVCratio.log.z)+
  s(Location,bs="re")+s(Transcriber,bs="re")
f[[3]]=PronDistStdDutch.c~s(PopSize.log)+s(PopAvgAge)+s(PopAvgIncome.log)+s(WordFreq.log)+s(
WordVCratio.log.z)+
  s(Word,bs="re")+s(Transcriber,bs="re")

f[[4]]=PronDistStdDutch.c~s(PopAvgAge)+s(PopAvgIncome.log)+s(WordFreq.log)+s(WordVCratio.log.
z)+
  s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re")
fun.list=f
remove(f)



# index
nnn=40
sizze=2000
index=index.maker(sizze,nnn,d)


# gamm
delta.mat.raw.i=matrix(NA,nnn,length(fun.list.gamm))
colnames(delta.mat.raw.i)=paste("fun", 1:length(fun.list.gamm), sep="")

for(i in 1:nnn) {
  cat("busy with round", i, "\n")
  for(ii in 1:length(fun.list.gamm)) {
    vg1=try(vinnie.gamm1(fun.list.gamm[[ii]],ran.list.gamm[[ii]],sizze,index[,i],d))
    delta.mat.raw.i[i,ii]=vg1
  }
}

mat.input=delta.mat.raw.i
gamm.result=matrix(NA,length(fun.list.gamm),3)
colnames(gamm.result)=c("mean","variance","model")
gamm.result[,3]=c(1:5)

for(i in 1:length(fun.list.gamm)) {
  gamm.result[i,1]=mean(mat.input[,i])
```

```
  gamm.result[i,2]=var(mat.input[,i])
}

gamm.result=gamm.result[order(gamm.result[,1]),]


# bam
# It shares the index with gamm
bam.result=model.selection.i(fun.list,sizze,nnn,index,d)




# Section 3.2.3

load('dialectNL.rda')
d=dialectNL
remove(dialectNL)
d$WordIsVerb=as.factor(d$WordIsVerb)
d$WordIsNounOrAdverb=as.factor(d$WordIsNounOrAdverb)

# Adjusting geometry
f=list()
f[[1]]=PronDistStdDutch.c~

s(PopSize.log_residGeo.z)+s(PopAvgAge_residPopAvgIncome.log_Geo.z)+s(PopAvgIncome.log_resi
dGeo.z)+s(WordFreq.log)+s(WordVCratio.log.z)+
  s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re")

f[[2]]=PronDistStdDutch.c~te(Longitude,Latitude)+

s(PopSize.log_residGeo.z)+s(PopAvgAge_residPopAvgIncome.log_Geo.z)+s(PopAvgIncome.log_resi
dGeo.z)+s(WordFreq.log)+s(WordVCratio.log.z)+
  s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re")

f[[3]]=PronDistStdDutch.c~te(Longitude,Latitude,by=WordIsVerb,d=c(1,1))+

s(PopSize.log_residGeo.z)+s(PopAvgAge_residPopAvgIncome.log_Geo.z)+s(PopAvgIncome.log_resi
dGeo.z)+s(WordFreq.log)+s(WordVCratio.log.z)+
  s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re")

f[[4]]=PronDistStdDutch.c~te(Longitude,Latitude,by=WordIsNounOrAdverb,d=c(1,1))+

s(PopSize.log_residGeo.z)+s(PopAvgAge_residPopAvgIncome.log_Geo.z)+s(PopAvgIncome.log_resi
dGeo.z)+s(WordFreq.log)+s(WordVCratio.log.z)+
  s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re")
geo.list=f
remove(f)


# Table 3.5
geo.result.1=model.selection.i(geo.list,10000,20,d)
geo.result.2=model.selection(geo.list,nrow(d),1,d)
```

```
# Section 3.2.4

# model selection
f=list()
# full model
f[[1]]=PronDistStdDutch.c~te(Longitude,Latitude,by=WordIsVerb,d=c(1,1))+

s(PopSize.log_residGeo.z)+s(PopAvgAge_residPopAvgIncome.log_Geo.z)+s(PopAvgIncome.log_resi
dGeo.z)+s(WordFreq.log)+s(WordVCratio.log.z)+
  s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re")

# omit s(Word,bs="re")
f[[2]]=PronDistStdDutch.c~te(Longitude,Latitude,by=WordIsVerb,d=c(1,1))+

s(PopSize.log_residGeo.z)+s(PopAvgAge_residPopAvgIncome.log_Geo.z)+s(PopAvgIncome.log_resi
dGeo.z)+s(WordFreq.log)+s(WordVCratio.log.z)+
  s(Location,bs="re")+s(Transcriber,bs="re")

# omit s(Location,bs="re")
f[[3]]=PronDistStdDutch.c~te(Longitude,Latitude,by=WordIsVerb,d=c(1,1))+

s(PopSize.log_residGeo.z)+s(PopAvgAge_residPopAvgIncome.log_Geo.z)+s(PopAvgIncome.log_resi
dGeo.z)+s(WordFreq.log)+s(WordVCratio.log.z)+
  s(Word,bs="re")+s(Transcriber,bs="re")

#omit s(Transcriber,bs="re")
f[[4]]=PronDistStdDutch.c~te(Longitude,Latitude,by=WordIsVerb,d=c(1,1))+

s(PopSize.log_residGeo.z)+s(PopAvgAge_residPopAvgIncome.log_Geo.z)+s(PopAvgIncome.log_resi
dGeo.z)+s(WordFreq.log)+s(WordVCratio.log.z)+
  s(Word,bs="re")+s(Location,bs="re")

# omit s(WordFreq.log)
f[[5]]=PronDistStdDutch.c~te(Longitude,Latitude,by=WordIsVerb,d=c(1,1))+

s(PopSize.log_residGeo.z)+s(PopAvgAge_residPopAvgIncome.log_Geo.z)+s(PopAvgIncome.log_resi
dGeo.z)+s(WordVCratio.log.z)+
  s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re")

# omit s(PopSize.log_residGeo.z)
f[[6]]=PronDistStdDutch.c~te(Longitude,Latitude,by=WordIsVerb,d=c(1,1))+

s(PopAvgAge_residPopAvgIncome.log_Geo.z)+s(PopAvgIncome.log_residGeo.z)+s(WordFreq.log)+s
(WordVCratio.log.z)+
  s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re")

# omit s(PopAvgAge_residPopAvgIncome.log_Geo.z)
f[[7]]=PronDistStdDutch.c~te(Longitude,Latitude,by=WordIsVerb,d=c(1,1))+
```

```
s(PopSize.log_residGeo.z)+s(PopAvgIncome.log_residGeo.z)+s(WordFreq.log)+s(WordVCratio.log.z)
+
  s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re")

# omit s(PopAvgIncome.log_residGeo.z)
f[[8]]=PronDistStdDutch.c~te(Longitude,Latitude,by=WordIsVerb,d=c(1,1))+

s(PopSize.log_residGeo.z)+s(PopAvgAge_residPopAvgIncome.log_Geo.z)+s(WordFreq.log)+s(Word
VCratio.log.z)+
  s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re")

# omit s(WordVCratio.log.z)
f[[9]]=PronDistStdDutch.c~te(Longitude,Latitude,by=WordIsVerb,d=c(1,1))+

s(PopSize.log_residGeo.z)+s(PopAvgAge_residPopAvgIncome.log_Geo.z)+s(PopAvgIncome.log_resi
dGeo.z)+s(WordFreq.log)+
  s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re")

# omit s(PopSize.log_residGeo.z) and s(PopAvgAge_residPopAvgIncome.log_Geo.z)
f[[10]]=PronDistStdDutch.c~te(Longitude,Latitude,by=WordIsVerb,d=c(1,1))+
  s(PopAvgIncome.log_residGeo.z)+s(WordFreq.log)+s(WordVCratio.log.z)+
  s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re")

# omit s(PopSize.log_residGeo.z), s(PopAvgAge_residPopAvgIncome.log_Geo.z), and
s(Transcriber,bs="re")
f[[11]]=PronDistStdDutch.c~te(Longitude,Latitude,by=WordIsVerb,d=c(1,1))+
  s(PopAvgIncome.log_residGeo.z)+s(WordFreq.log)+s(WordVCratio.log.z)+
  s(Word,bs="re")+s(Location,bs="re")

# omit s(PopAvgIncome.log_residGeo.z) and s(PopSize.log_residGeo.z)
f[[12]]=PronDistStdDutch.c~te(Longitude,Latitude,by=WordIsVerb,d=c(1,1))+
  s(PopAvgAge_residPopAvgIncome.log_Geo.z)+s(WordFreq.log)+s(WordVCratio.log.z)+
  s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re")

# omit s(WordFreq.log) and s(PopAvgIncome.log_residGeo.z)
f[[13]]=PronDistStdDutch.c~te(Longitude,Latitude,by=WordIsVerb,d=c(1,1))+
  s(PopSize.log_residGeo.z)+s(PopAvgAge_residPopAvgIncome.log_Geo.z)+s(WordVCratio.log.z)+
  s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re")

# omit s(WordFreq.log), s(PopAvgIncome.log_residGeo.z), and s(PopSize.log_residGeo.z)
f[[14]]=PronDistStdDutch.c~te(Longitude,Latitude,by=WordIsVerb,d=c(1,1))+
  s(PopAvgAge_residPopAvgIncome.log_Geo.z)+s(WordVCratio.log.z)+
  s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re")

# remove all the splines
f[[15]]=PronDistStdDutch.c~te(Longitude,Latitude,by=WordIsVerb,d=c(1,1))+

PopSize.log_residGeo.z+PopAvgAge_residPopAvgIncome.log_Geo.z+PopAvgIncome.log_residGeo.z
+WordFreq.log+WordVCratio.log.z+
  s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re")
fun.list=f
remove(f)
```

```
# Table 3.6
model.sel.result=model.selection(fun.list,nrow(d),1,d)



# Section 3.2.5


# model expansion

f=list()
# add noting
f[[1]]=PronDistStdDutch.c~te(Longitude,Latitude,by=WordIsVerb,d=c(1,1))+
s(PopAvgAge_residPopAvgIncome.log_Geo.z)+s(WordFreq.log)+s(WordVCratio.log.z)+
s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re")

# add PopMaleFemaleRatio
f[[2]]=PronDistStdDutch.c~te(Longitude,Latitude,by=WordIsVerb,d=c(1,1))+
s(PopAvgAge_residPopAvgIncome.log_Geo.z)+s(WordFreq.log)+s(WordVCratio.log.z)+
s(PopMaleFemaleRatio)+
s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re")

# add SpeakerBirthYear
f[[3]]=PronDistStdDutch.c~te(Longitude,Latitude,by=WordIsVerb,d=c(1,1))+
s(PopAvgAge_residPopAvgIncome.log_Geo.z)+s(WordFreq.log)+s(WordVCratio.log.z)+
s(SpeakerBirthYear)+
s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re")

# add SpeakerRecordingYear
f[[4]]=PronDistStdDutch.c~te(Longitude,Latitude,by=WordIsVerb,d=c(1,1))+
s(PopAvgAge_residPopAvgIncome.log_Geo.z)+s(WordFreq.log)+s(WordVCratio.log.z)+
s(SpeakerRecordingYear)+
s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re")

# add PopMaleFemaleRatio, SpeakerBirthYear, and SpeakerRecordingYear
f[[5]]=PronDistStdDutch.c~te(Longitude,Latitude,by=WordIsVerb,d=c(1,1))+
s(PopAvgAge_residPopAvgIncome.log_Geo.z)+s(WordFreq.log)+s(WordVCratio.log.z)+
s(PopMaleFemaleRatio)+s(SpeakerBirthYear)+s(SpeakerRecordingYear)+
s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re")
exp.list=f
remove(f)


# Table 3.7
model.exp.result=model.selection(exp.list,nrow(d),1,d)


# End chapter 3
```

```
# Chapter 4

library(gamm4)
library(xtable)
load('dialectNL.rda')
d=dialectNL
d$WordIsVerb=as.factor(d$WordIsVerb)


final.model=bam(
PronDistStdDutch.c~te(Longitude,Latitude,by=WordIsVerb,d=c(1,1))+
s(PopAvgAge_residPopAvgIncome.log_Geo.z)+s(WordFreq.log)+
s(WordVCratio.log.z)+s(SpeakerBirthYear)+
s(Word,bs="re")+s(Location,bs="re")+s(Transcriber,bs="re"), data=d)

save(final.model,file="final.model.rda")



# Table 4.1
summary(final.model)
# xtable(summary(final.model)$s.table)




# Figure 4.1

load("final.model.rda")

old.par<-par(mfrow=c(2,2))
par(cex=1.5)

for (i in 3:6) {
  plot(final.model,rug=FALSE,select=i,lwd=2,scale=0,font.main=1)
}




# Figure 4.2
dev.off()
old.par<-par(mfrow=c(2,2))
par(cex=1.5)
for (i in 3:6) {
  plot(final.model,rug=FALSE,select=i,lwd=2,font.main=1)
}




# Figure 4.3
```

```
load("final.model.rda")

# Word
r.wo=coef(final.model)[86:644]

# Location
r.lo=coef(final.model)[645:1046]

# Transcriber
r.tr=coef(final.model)[1047:1076]

# resolution: 800 x 800
labels=c("s(Word,546.68)","s(Location,346.66)","s(Transcriber,18.87)")
boxplot(r.wo,r.lo,r.tr,xlab="",ylab="effects",font.main=1, main="")
axis(1, labels=c(labels), at=1:3, las=1)




# Figure 4.4

load('wrddst.rda')
d=wrddst

dV=subset(d,WordCat=="V")
dR=subset(d,WordCat %in% c("A","B","N"))


geogamV = gam(RefPMIdistMeanLog.c ~ s(GeoX,GeoY), data=dV)
geogamR = gam(RefPMIdistMeanLog.c ~ s(GeoX,GeoY), data=dR)

old.par<-par(mfrow=c(1,2))
vis.gam(geogamV, plot.type="contour", color="gray", too.far=0.05,
view=c("GeoX","GeoY"),font.main=1, main="WordCategory = verb", xlab="", ylab="", cex.axis=0.75)
text(5.81, 53.2, "Friesland", adj=0.5, cex=0.75)
text(6.56, 53.3, "Groningen", adj=0.5, cex=0.75)
text(6.56, 52.8, "Drenthe", adj=0.5, cex=0.75)
text(6.7, 52.4, "Twente", adj=0.5, cex=0.75)
text(5.9, 50.8, "Limburg", adj=0.5, cex=0.75)
text(3.7, 51.3, "Zeeland", adj=0.5, cex=0.75)
text(5.0, 51.6, "Brabant", adj=0.5, col="white", cex=0.75)
text(4.7, 52.3, "Holland", adj=0.5, col="white", cex=0.75)
title(xlab="Longitude",ylab="Latitude",cex.lab=0.75)

vis.gam(geogamR, plot.type="contour", color="gray", too.far=0.05, view=c("GeoX","GeoY"),
font.main=1, main="WordCategory = rest", xlab="", ylab="", cex.axis=0.75)
text(5.81, 53.2, "Friesland", adj=0.5, cex=0.75)
text(6.56, 53.3, "Groningen", adj=0.5, cex=0.75)
text(6.56, 52.8, "Drenthe", adj=0.5, cex=0.75)
text(6.7, 52.4, "Twente", adj=0.5, cex=0.75)
text(5.9, 50.8, "Limburg", adj=0.5, cex=0.75)
text(3.7, 51.3, "Zeeland", adj=0.5, cex=0.75)
```

```
text(5.0, 51.6, "Brabant", adj=0.5, col="white", cex=0.75)
text(4.7, 52.3, "Holland", adj=0.5, col="white", cex=0.75)
title(xlab="Longitude",ylab="Latitude",cex.lab=0.75)


# End chapter 4
```