The Dutch Diminutive: Between Inflection and Derivation

Analyis Script

Daniil Bondarenko

2023-01-20

0. Important notice

This script is the only one you should need to run in order to replicate the results of the study. While "preprocessing.Rmd" relies on the externally-accessed DLP2 files in order to build the experimental dataset in the first place, here the dataset is already assembled and ready for final data transformations and subsequent analysis. If you're looking to get a full picture of the pipeline and account for every data-related step within this project, feel free to peek into the preprocessing script. Otherwise, you should be good to go!

1. Import the necessary R packages

```
library(dplyr) # For data cleanup and transformations
library(readr) # For more straightforward ways to import/export data
library(lme4) # For mixed-effects linear regression modelling
library(car) # For alternative options when it comes to plots and regressions
library(ggplot2) # For a more comprehensive way of plotting data
library(effsize) # For Cohen's d, etc.
library(afex) # For likelihood ratio tests
library(MuMIn) # For extracting R-squared values from lmers
options(scipen = 999) # For easier interpretability of the slopes
```

2. Import the experimental dataset, perform some final cleanup

```
trialdata <- read_csv("../corpus/trialdata.csv", show_col_types = FALSE)
# Move all relevant columns to the left for better readability
trialdata <- select(trialdata,
                    "spelling":"dec_criterion",
                    "rtC":"rateC",
                    "Length":"Colt_Nphon",
                    "item",
                    "participant",
                    "lower":"rateR",
                    "rtI":"zrateI",
                    "acc.mean":"zrateI.sd"
)
# Filter out the irrelevant columns based on theory-supported decisions
# NOTE: Done here so variables could be reintroduced at will if need be
```

```
trialdata <- select(trialdata,
                    -"N_phonemes",
                    # OLD20 chosen as the neighbourhood var: filter out the rest
                    -"Colt_N",
                    -"PLD30",
                    -"Colt_Nphon",
                    # RTs chosen as the dependent var; filter out the rest,
                    # do the centering and z-Transforming within the script;
                    # filter out the pre-existing variables
                    -"lower",
                    -"upper",
                    -"rtR",
                    -"rateR",
                    -"rtI",
                    -"rateI",
                    -"zrtC",
                    -"zrateC",
                    -"zrtI",
                    -"zrateI",
                    -"rtC.mean",
                    -"rtC.sd",
                    -"rateC.mean",
                    -"rateC.sd",
                    -"zrtC.mean",
                    -"zrtC.sd",
                    -"zrateC.mean",
                    -"zrateC.sd",
                    -"rtI.mean",
                    -"rtI.sd",
                    -"rateI.mean",
                    -"rateI.sd",
                    -"zrtI.mean",
                    -"zrtI.sd",
                    -"zrateI.mean",
                    -"zrateI.sd"
)
```

3. Run sanity checks

```
# Compare number of rows per diminutive condition
trialdata <- mutate(trialdata, dim_type=factor(dim_type))
class(trialdata$dim_type) == 'factor'
```

```
## [1] TRUE
```

```
summary(trialdata$dim_type)
```

```
##  both deriv  infl
##  1473  4827  5049
```

```r
# Check for correlations using Pearson's R
cor(trialdata$SUBTLEX2, trialdata$Length)
```

```
## [1] -0.1234163
```

```r
cor(trialdata$Length, trialdata$Nsyl)
```

```
## [1] 0.8684608
```

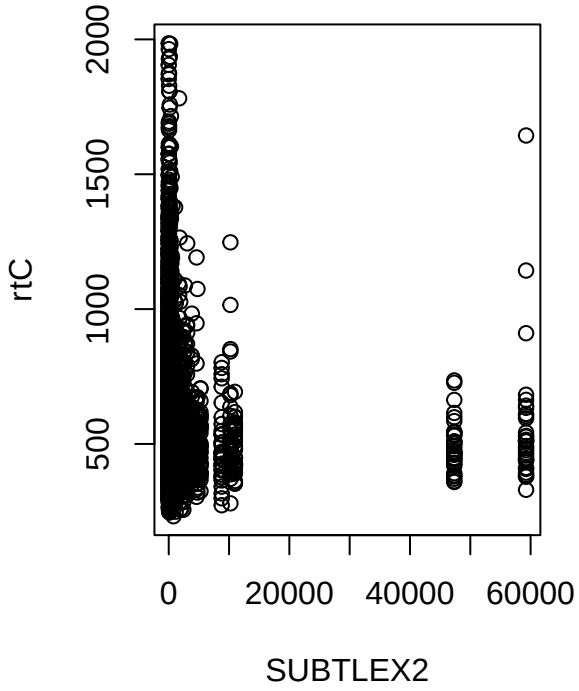
```r
cor(trialdata$Length, trialdata$nmorph)
```

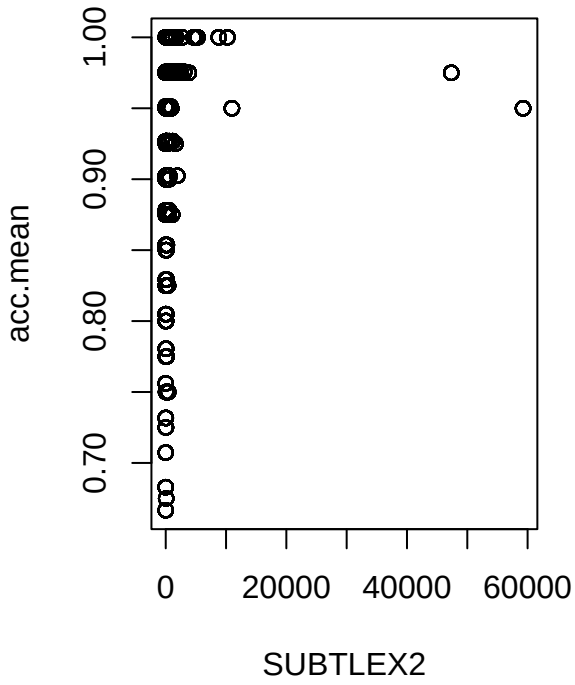
```
## [1] 0.7214382
```

```r
cor(trialdata$Nsyl, trialdata$nmorph)
```

```
## [1] 0.7231823
```

```r
# Both length and morpheme count pretty highly correlated with syllable count;
# Keeping Nsyl has much less of a theoretical reason and might impact the
# effects of the other two predictors; exlude it from the analysis.
trialdata <- select(trialdata, -"Nsyl")

# There's only two items past the 11000 mark
par(mfrow = c(1, 2))
with(trialdata, plot(SUBTLEX2, rtC))
with(trialdata, plot(SUBTLEX2, acc.mean))
```





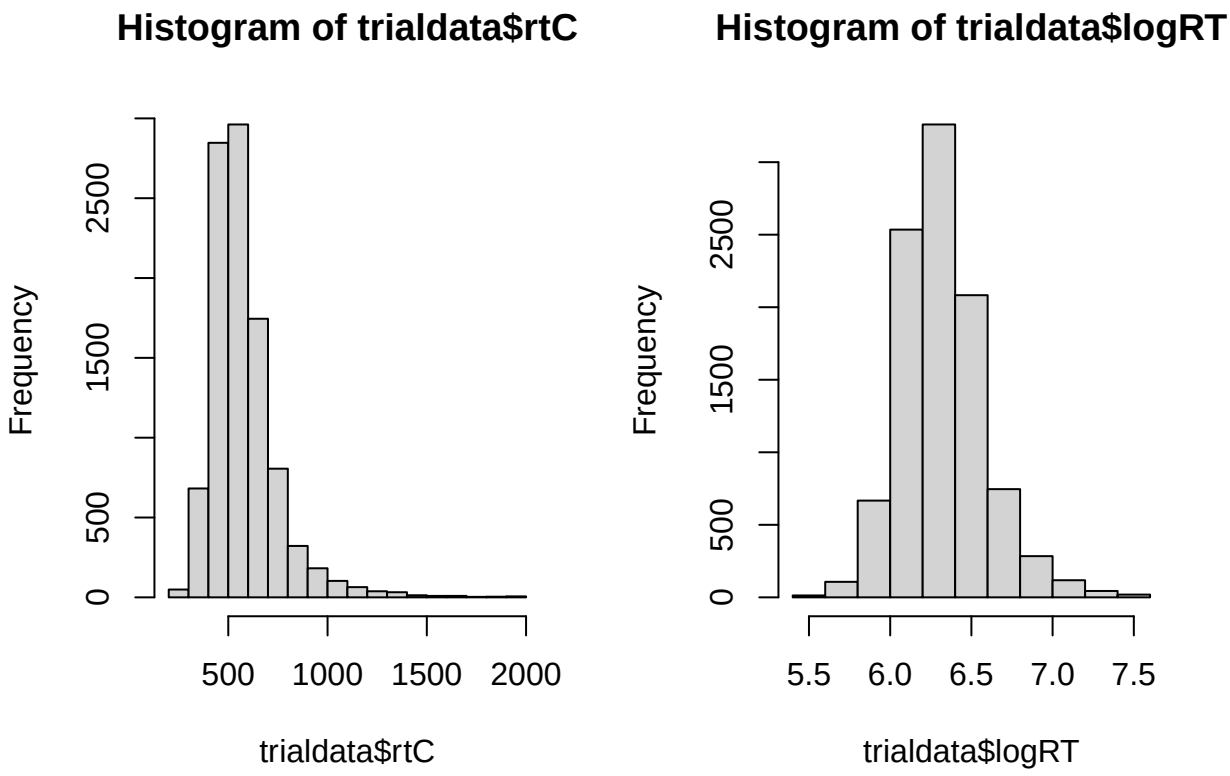

```
# Find out what they are
trialdata %>%
  select("spelling", "SUBTLEX2") %>%
  distinct() %>%
  arrange(desc(SUBTLEX2)) %>%
  head()
```

```
## # A tibble: 6 x 2
##   spelling SUBTLEX2
##   <chr>       <dbl>
## 1 beetje      59247
## 2 meisje      47345
## 3 liefje      10969
## 4 grapje      10226
## 5 feestje      8787
## 6 drankje      5265
```

```
# "beetje" and "meisje" both have egregiously extreme freq values;
# all the more reason to log-transform the frequency measure
# (see the DLP2 paper and Winter 2020 for theoretical motivation)
```

4. Finalise fine preprocessing

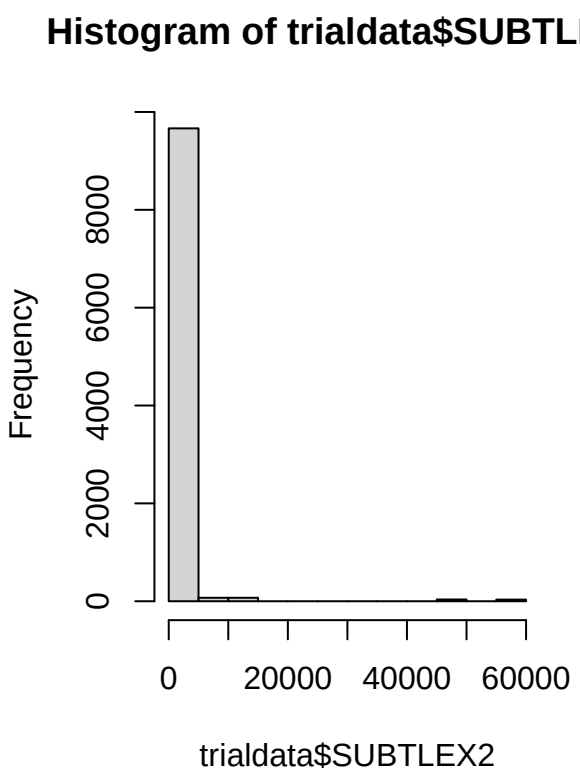
```
# Optional step: exclude the non-core diminutive items first,
# i.e. the ones with multiple ways to parse.
# NOTE: make sure to change contrasts below!
trialdata <- filter(trialdata, dim_type !="both")

# Log-transform reaction times using the base logarithm (see Winter 2020)
trialdata$logRT <- log(trialdata$rtC)
par(mfrow = c(1, 2))
hist(trialdata$rtC)
hist(trialdata$logRT)
```

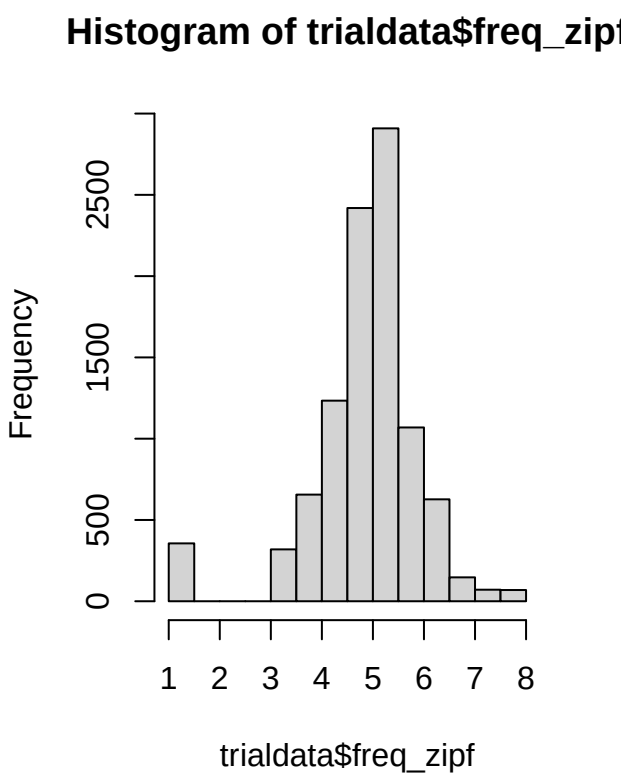



```
# See SUBTLEX-UK: A new and improved word frequency database for British English
# By Van Heuven et al., 2014 for motivation for Zipf-vals for frequency
trialdata$freq_zipf <- log10(trialdata$SUBTLEX2+0.01)+3
summary(trialdata$freq_zipf)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   4.447   4.978   4.853   5.394   7.773
```

```
par(mfrow = c(1, 2))
hist(trialdata$SUBTLEX2)
hist(trialdata$freq_zipf)
```

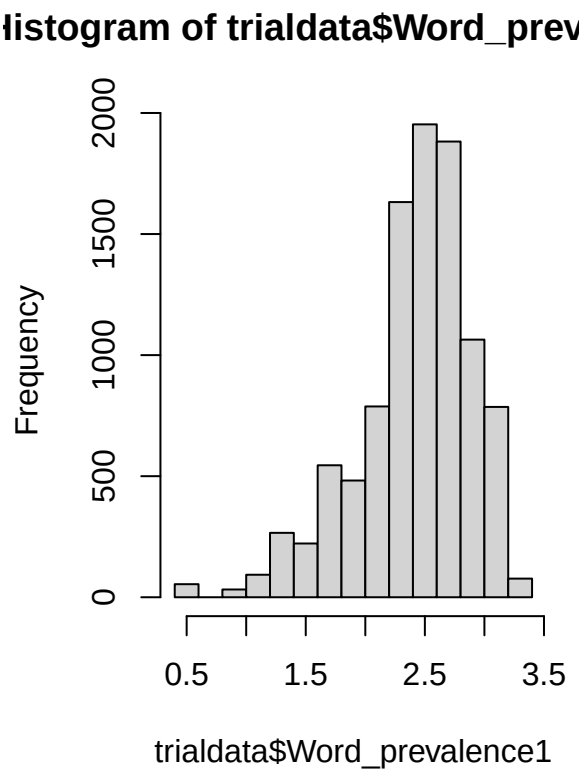


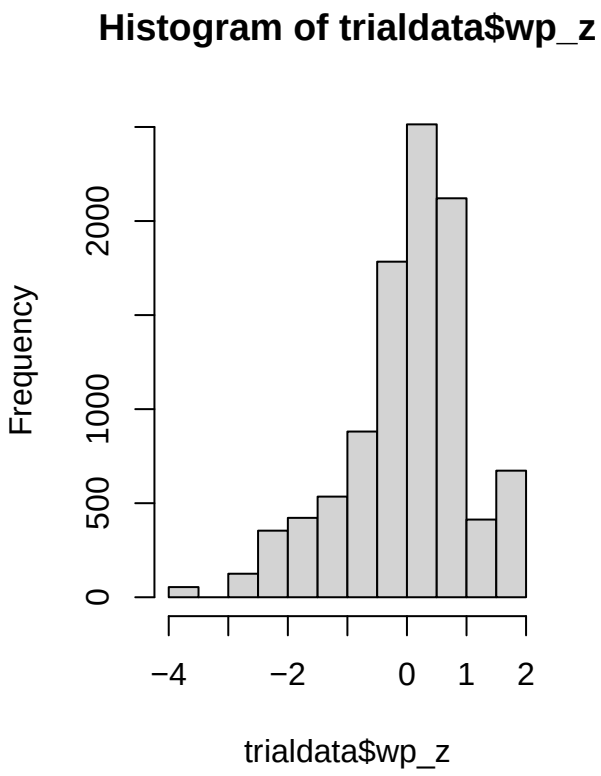



```
cor(trialdata$freq_zipf, trialdata$Word_prevalence1)
```

```
## [1] 0.4512153
```

```
# Center all the relevant predictors for an easier comparison of effect size
trialdata <- mutate(trialdata,
                    zipf_z = scale(freq_zipf),
                    len_z = scale(Length),
                    nmorph_z = scale(nmorph),
                    old_z = scale(OLD20),
                    conc_z = scale(Concreteness),
                    aoa_z = scale(AoA),
                    wp_z = scale(Word_prevalence1))

# Look into the word prevalence measure
# It already consists of z-Scores, but based on the entire corpus
# Compare to the freshly recentered distribution
par(mfrow = c(1, 2))
hist(trialdata$Word_prevalence1)
hist(trialdata$wp_z)
```



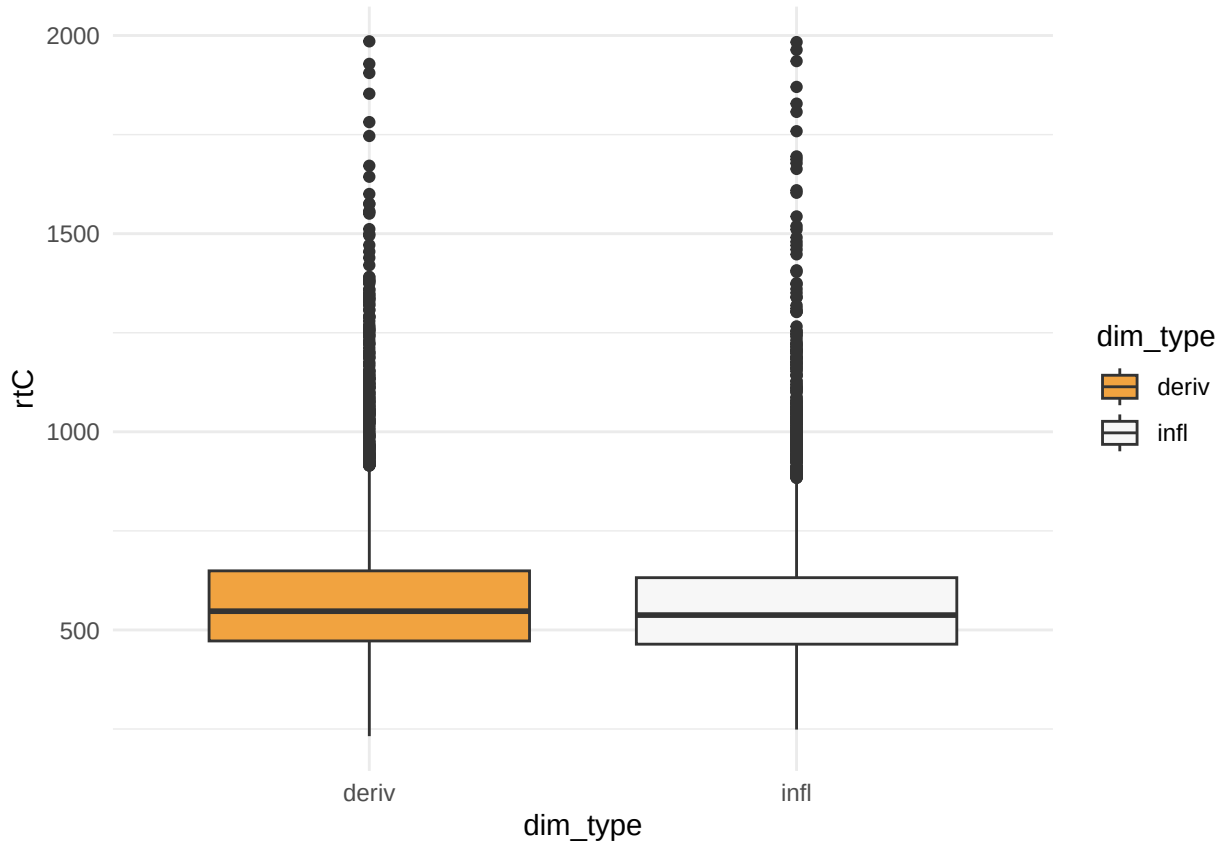


5. Descriptive Statistics

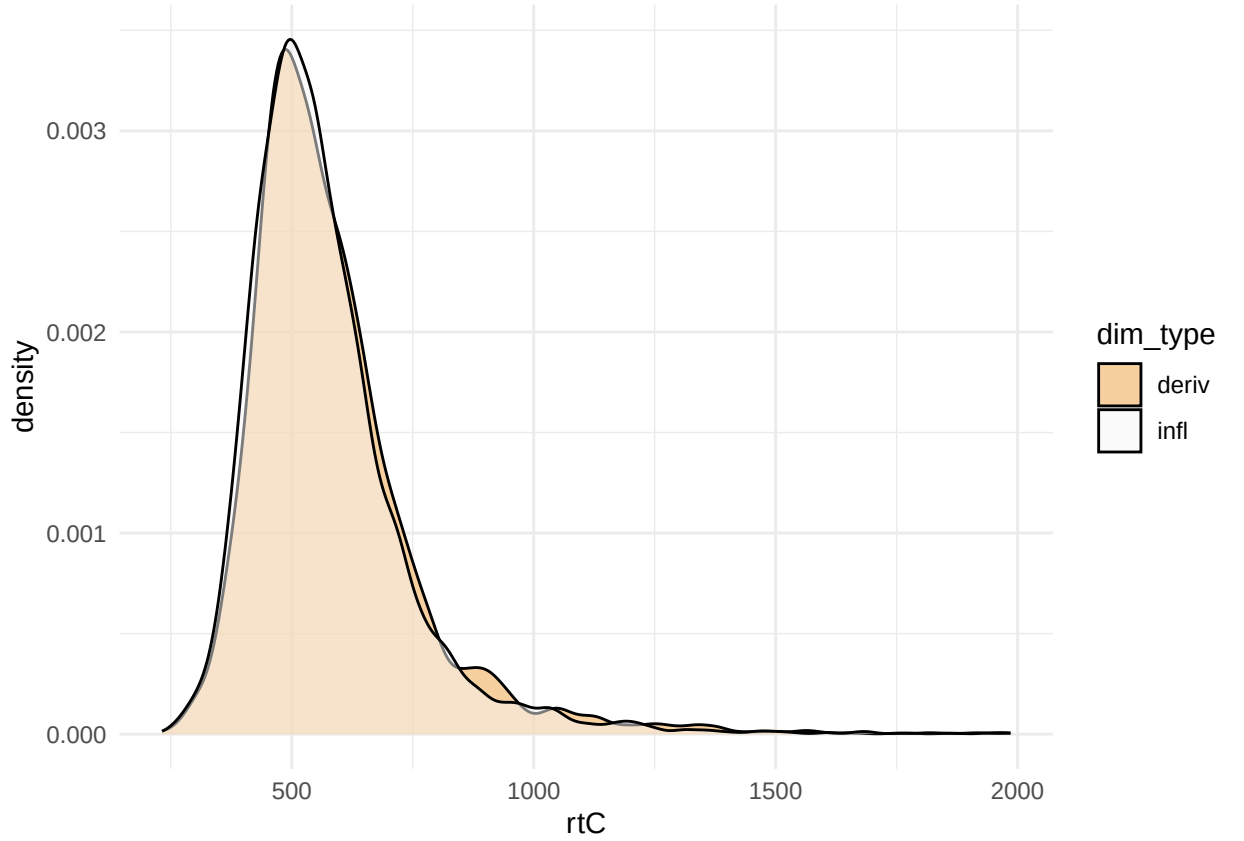
```
# Get means for the diminutives
trialdata %>% group_by(dim_type) %>%
  summarize(M = mean(rtC), SD = sd(rtC))
```

```
## # A tibble: 2 x 3
##   dim_type     M    SD
##   <fct>    <dbl> <dbl>
## 1 deriv     584.  176.
## 2 infl      569.  168.
```

```
trialdata %>% ggplot(aes(x = dim_type, y = rtC, fill = dim_type)) +
  geom_boxplot() + theme_minimal() +
  scale_fill_brewer(palette = "PuOr")
```




```
ggsave("../figures/dim_box.png", width = 8, height = 6)

trialdata %>% ggplot(aes(x = rtC, fill = dim_type)) +
  geom_density(alpha = 0.5) + theme_minimal() +
  scale_fill_brewer(palette = "PuOr")
```




```
ggsave("../figures/dim_density.png", width = 8, height = 6)

# Derived wordforms take the longest on average;
# Inflected wordforms take about 13 ms less on average;
# Wordforms with multiple ways of analysis take the least time on average
# To say that this was unexpected is to say nothing at all.
```

6. Inferential Statistics

Here, we will finally handle some tests that will let us draw conclusions from the sample about the population at large. We start with assigning contrasts, then run a simple t-test and then move on to modelling.

```
# Prepare the diminutive variable: check contrasts
trialdata <- mutate(trialdata, dim_type=factor(dim_type))
contrasts(trialdata$dim_type)
```

```
##       infl
## deriv    0
## infl     1
```

6.1. Contrasts and t-tests

The next two code chunks handle the contrast coding; they are mutually exclusive depending on the decision to include observations with "both" as a value for dim_type. In order to switch between them, go to the

chunk options and specify *eval=FALSE* to stop the script from running the code and *include=FALSE* to prevent the code from being included in the final knitted document.

The motivation for sum-coding between "deriv" and "infl" is taken from Winter 2020, where it is argued that sum-coding is better suited for mixed-effects models with interactions and random effects; additionally, sum-coding is argued to make the coefficients easier to interpret.

```
# Sum-coding for a purely deriv/infl dataset (see above)
contrasts(trialdata$dim_type) <- contr.sum(2)
contrasts(trialdata$dim_type)
```

```
##       [,1]
## deriv    1
## infl    -1
```

```
t.test(rtC ~ dim_type, data = trialdata)
```

```
##
##  Welch Two Sample t-test
##
## data:  rtC by dim_type
## t = 4.1786, df = 9795.1, p-value = 0.00002958
## alternative hypothesis: true difference in means between group deriv and group infl is not equal to 0
## 95 percent confidence interval:
##   7.700133 21.307963
## sample estimates:
## mean in group deriv  mean in group infl
##            583.9459            569.4418
```

The motivation for Helmert-coding between "deriv", "infl", and "both" is simple: for wordforms with either "deriv" or "infl", a single decomposition pipeline is proposed. Assuming structural differences, recognizing either is the same type of operation being performed in a different location in the structure. Compare with "both", where the decomposition apparatus presumably pursues both leads at once; therefore, this should be reflected in the coding as such:

- First, compare the differences between deriv and infl (either x or y)
- Then, compare their average RTs to both (x and y at the same time)

6.2. Modelling

The formula for the experimental model closely follows the formula reported in the DLP2 paper, with the addition of two predictors:

a. dim_type, a factor with values "deriv" and "infl" (possibly "both" for the expanded dataset): see above for the discussion of the predictions.
b. nmorph, a numeric with values reflecting the (observable and theoretically motivated) morpheme count in the structure of each wordform: assuming full decomposition, every word is broken down to the most primitive units (= morphemes), so it logically follows that the more morphemes a wordform consists of, the more there is to break down and recompose, and the more time it should take to recognize a word.

The final formula of the model is therefore this (see "mod_full" below):

RT ~ Frequency*Length + Morpheme count + OLD20 ratings + Concreteness + Age of acquisition + Word prevalence + Varying intercept by participant + Varying slope by participant + Varying intercept by item

```
# Run the base model with only frequency and random intercepts
# Mention including the participant slopes and item intercepts as the
# motivation to satisfy the independence assumption (Winter 2020, Ch.14)
mod.base <- lmer(logRT ~ zipf_z +
                   (1|participant) +
                   (1|item), data = trialdata, REML = TRUE)
summary(mod.base) # Make sure the frequency findings aren't wildly off
```

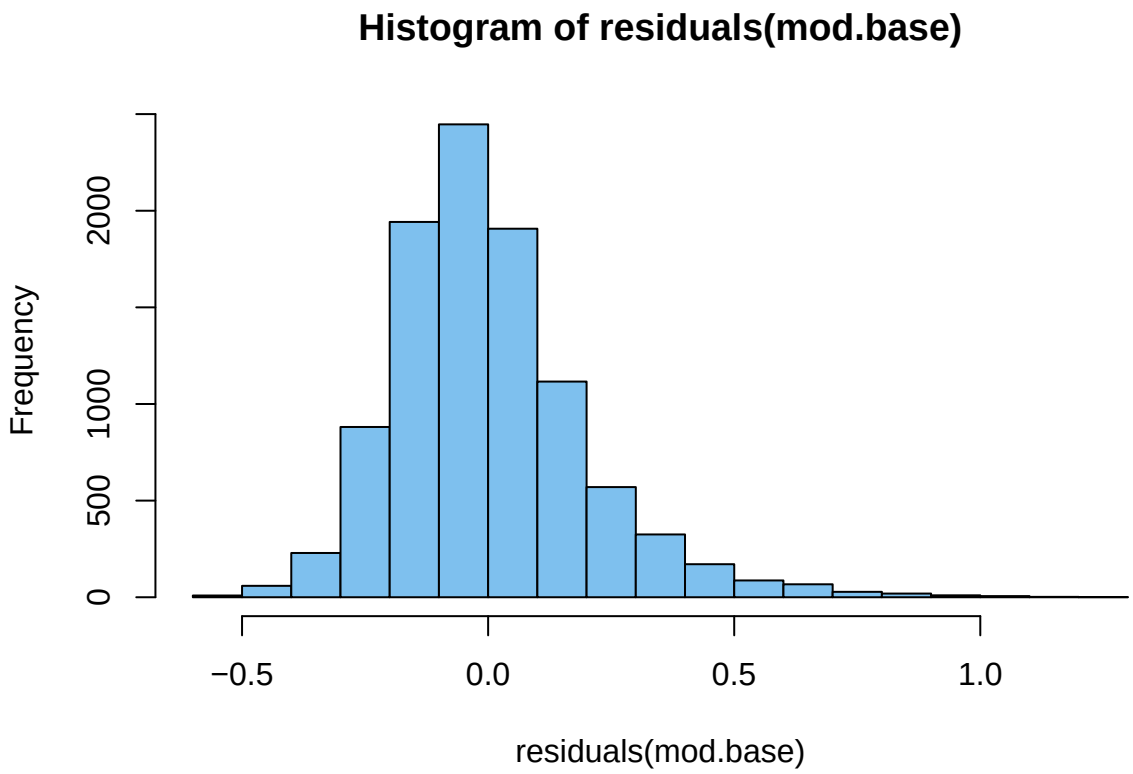
```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: logRT ~ zipf_z + (1 | participant) + (1 | item)
##    Data: trialdata
##
## REML criterion at convergence: -3048.1
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.9195 -0.6485 -0.1297  0.4754  6.3223
##
## Random effects:
##  Groups      Name        Variance Std.Dev.
##  item        (Intercept) 0.003218 0.05673
##  participant (Intercept) 0.022484 0.14995
##  Residual                0.039961 0.19990
## Number of obs: 9876, groups:  item, 270; participant, 81
##
## Fixed effects:
##              Estimate Std. Error        df t value            Pr(>|t|)
## (Intercept)   6.32166    0.01714  86.73930  368.94 <0.0000000000000002 ***
## zipf_z       -0.05430    0.00392 271.62522  -13.85 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##        (Intr)
## zipf_z 0.005
```

```
r.squaredGLMM(mod.base)
```

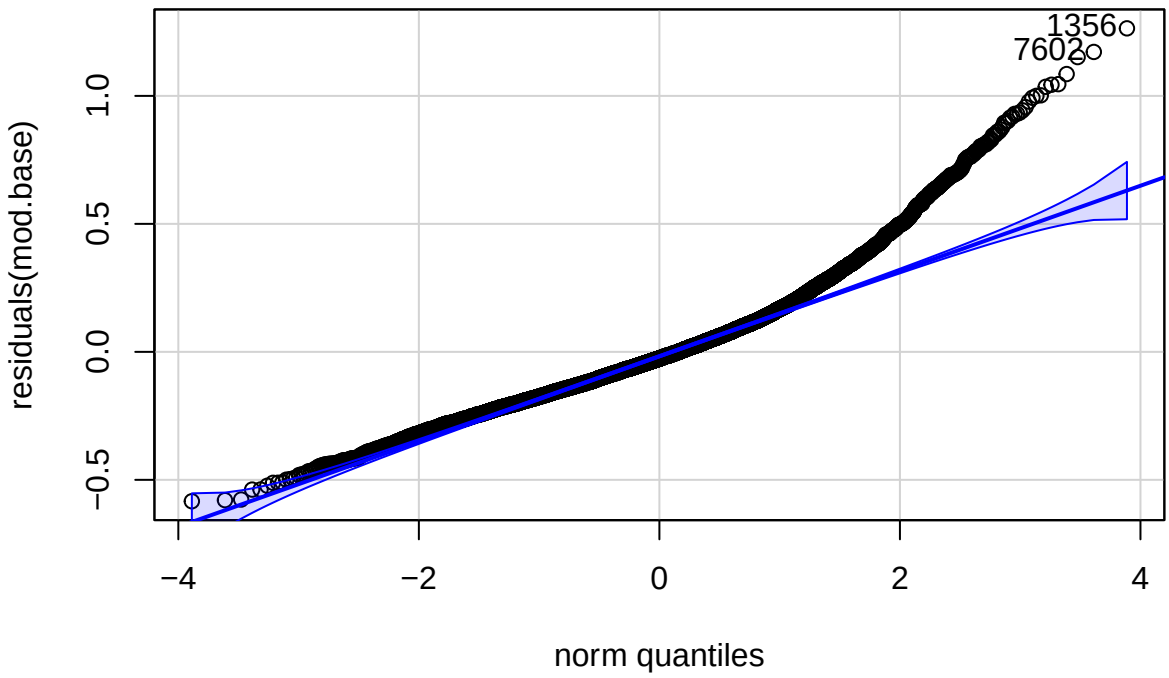
```
## Warning: 'r.squaredGLMM' now calculates a revised statistic. See the help page.
```

```
##            R2m       R2c
## [1,] 0.0429752 0.4175708
```

```
# Residuals seem pretty skewed; plot for a visual
hist(residuals(mod.base), col = 'skyblue2') # Plot 1, histogram:
```

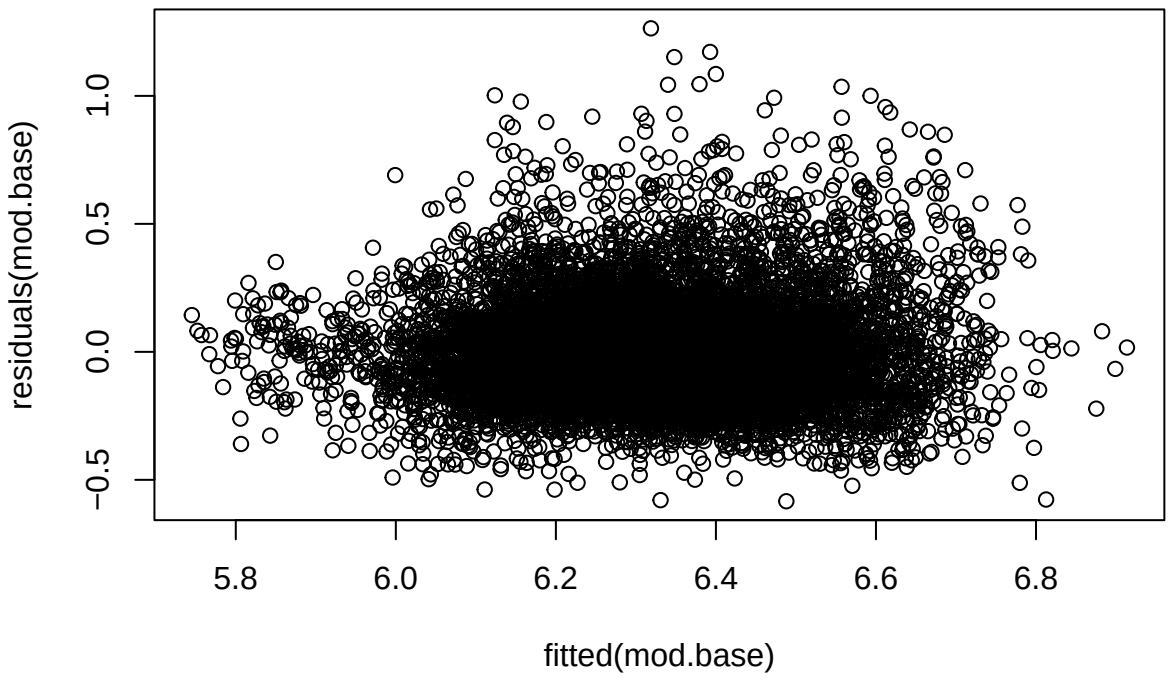



```
qqPlot(residuals(mod.base)) # Plot 2, Q-Q plot:
```




```
## [1] 1356 7602
```

```
plot(fitted(mod.base), residuals(mod.base)) # Plot 3, residual plot:
```




```
# NOTE: a noticeable positive skew in the residuals;
# Consider introducing a nonlinear term (Winter 2020)

# Run the full model and assess each factor's contributions using mixed()
mod_full <- lmer(logRT ~
                 dim_type +
                 zipf_z*len_z +
                 nmorph_z +
                 old_z +
                 conc_z +
                 aoa_z +
                 wp_z +
                 (dim_type|participant) +
                 (1|item), data = trialdata)
summary(mod_full)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: logRT ~ dim_type + zipf_z * len_z + nmorph_z + old_z + conc_z +
##     aoa_z + wp_z + (dim_type | participant) + (1 | item)
##    Data: trialdata
##
## REML criterion at convergence: -3072.3
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -3.0814 -0.6487 -0.1302  0.4748  6.3792
##
## Random effects:
##  Groups      Name        Variance   Std.Dev. Corr
##  item        (Intercept) 0.00202734 0.045026
##  participant (Intercept) 0.02241370 0.149712
##              dim_type1   0.00002349 0.004847 -0.57
##  Residual                0.03993756 0.199844
## Number of obs: 9876, groups:  item, 270; participant, 81
##
## Fixed effects:
##                 Estimate  Std. Error          df t value             Pr(>|t|)
## (Intercept)    6.3192248   0.0170457  85.5262324 370.724 < 0.0000000000000002
## dim_type1     -0.0039450   0.0038085 199.1014944  -1.036               0.3015
## zipf_z        -0.0231868   0.0049205 260.5178983  -4.712        0.0000039906
## len_z          0.0008152   0.0077491 254.5165429   0.105               0.9163
## nmorph_z       0.0024668   0.0054907 251.4441215   0.449               0.6536
## old_z          0.0198108   0.0081368 253.5080380   2.435               0.0156
## conc_z         0.0019198   0.0040232 253.5197198   0.477               0.6336
## aoa_z          0.0193245   0.0046350 251.2194595   4.169        0.0000420932
## wp_z          -0.0243436   0.0043580 260.4949857  -5.586        0.0000000584
## zipf_z:len_z  -0.0043767   0.0035650 262.7233317  -1.228               0.2207
##
## (Intercept)  ***
## dim_type1
## zipf_z       ***
## len_z
## nmorph_z
## old_z        *
## conc_z
## aoa_z        ***
## wp_z         ***
## zipf_z:len_z
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) dm_ty1 zipf_z len_z  nmrph_ old_z  conc_z aoa_z  wp_z
## dim_type1   -0.069
## zipf_z      -0.023 -0.099
## len_z       -0.007 -0.021  0.084
## nmorph_z     0.003 -0.113  0.186 -0.161
## old_z        0.008  0.029  0.027 -0.739 -0.323
## conc_z      -0.003  0.195  0.323  0.048  0.038 -0.050
## aoa_z       -0.008 -0.157  0.244 -0.068  0.018 -0.083  0.314
## wp_z         0.007  0.065 -0.445 -0.090 -0.096 -0.044 -0.195  0.289
## zipf_z:ln_z  0.087  0.055 -0.278 -0.096  0.034  0.098 -0.047 -0.076  0.037
```

```
r.squaredGLMM(mod_full)
```

```
##             R2m       R2c
## [1,] 0.06083101 0.4177627
```

```r
vif(mod_full) # Get variance inflation factors to check for collinearity
```

```
##     dim_type       zipf_z        len_z     nmorph_z        old_z       conc_z
##     1.193502     2.161487     5.163701     2.598940     5.688133     1.393557
##        aoa_z         wp_z zipf_z:len_z
##     1.867683     1.715691     1.165873
```