Slide whistle stats

# Load libraries

library(lme4)

## Loading required package: Matrix

##   
## Attaching package: 'lme4'

## The following object is masked from 'package:stats':  
##   
## sigma

library(infotheo)  
library(lattice)  
library(gplots)

##   
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':  
##   
## lowess

# Introduction

In this document, I try and address two predictions. (1) signallers avoid steep regions and (2) curvature affects variability. The approach is to use mixed effects models, with random effects for chain and target meaning, with random slopes as suggested by model comparison.

# Load data

signals\_MT = read.csv("../Data/Signals\_MT.csv",stringsAsFactors=F)  
signals\_SONA = read.csv("../Data/Signals\_SONA.csv",stringsAsFactors=F)  
  
signals\_MT$run = "MT"  
signals\_SONA$run = "SONA"

## Variables

A description of the variables included in the datasets:

* averageSlope/averageSlope.hz: average absolute difference in physical space/hz space between samples. High value = lots of movement. mean(abs(diff(sig)))
* propZero/propZero.hz: proportion of times in which the difference between sample values was negligible. High value = lots of time standing still. sum(diff(sig)<5) / length(sig
* switches/switches.hz: Number of changes in direction. sum(abs((diff(diff(sig)>=0))))
* maxslope/maxslope.hz: Maximum change in singal. max(abs(diff(sig)))
* slope.move: average slope change excluding zero. mean(abs(diff(sig)[abs(diff(sig))>=5]))
* jerkyness: standard deviation between derivative of signal. Low value = smoothly changing signal. sd(abs(diff(sig)))
* prop.time.plateau: Proportion of time spent in the plateau region (physical space only). Steep regions are defined as: 0.1-0.3 and 0.5-0.7, all other values are plateau. (the prop.time.plateau.hz variable should be ignored)
* switches.between.sections: If the signal space is divided into three plateau regions and two steep regions, the number of times the signal switches from one region to another.
* steepness.sig.mean - the signal's average steepenss of curve. For each change in a unit of physical space, what is the proportional change in hz space? The steepness values are taken from the curve used to produce the signal. So, any signal produced with zero curvature should have the same steepness.sig.mean (except that the hz scale is perceptually linear, so there may be small deviations). The same signal drawn on different curvatures will have different values of steepness.sig.mean, in a way that's not easy to predict (high curvatures have more steep regions, but also more flat regions).
* steepness.sig.mean.max - same as steepness.sig.mean, but where the steepenss values are taken from the maximum curvature steepness, to normalise over curvature values.
* logfile: The name of the log file used in the experiment
* chain: chain number (according to the online server)
* curvature: value of the curvature c of the articulator mapping
* run: source of the data (MT = mechanical turk, SONA = SONA)
* physSignal: the raw values of the physical signal recorded by the program, as a string of values separated by underscores. Values are proportion of the way along the slider x 1000, where 0 is low and 1000 is high
* hzSignal: the raw values of the signal in hz played to the participant. Same format as above, but in hz.
* numLearningRounds: the number of learning trials that the participant took to learn all 3 meanings (multiples of 3). This is the total value for the participant, not per meaning. For a given participant, this value is duplicated across the entries for each meaning. (so to get the number of learning rounds for a given participant, you should just look at values for e.g. meaning==1)
* abortedReproductionAttempts: The total number of times a player retries to produce a signal. As above, the number is reproduced over the 3 meanings.
* physSigDistinctiveness: How distinctive are the three signals that a participant produced? Split the time/signal space into a 10 x 10 grid. Count the number of appearances the signal makes in each cell to get a matrix of numbers that represents the journey of the signal. Do that for each signal. Then for each signal pair, calculate the absolute difference between the two matrices. Two very different signals will produce a high value. Two identical signals will produce a value of zero. Note that, because the time dimension is considered, two signals with the same trajectory but different speeds will look different. See the function "compareSignals" in GetData.R.
* hzSigDistinctiveness: Same as above, but in hz space.
* dataFileHzSignal: The file where the raw data is stored.
* dataFileHzSignal: The file where the raw data is stored.
* chain2: A unique identifier for the diffusion chain (chain number, log file, curvature). This should be used instead of 'chain', because 'chain' can have identical chain numbers for different curvature values.
* rawDataSource: Source of the raw data.

# Prediction 1: Signallers avoid unstable regions

We predict speakers indeed will avoid unstable regions of the articulator for stronger biases, and instead use the stable (quantal) ones. Moreover, we expect that the effect will become more pronounced as generations go by, that is, that nonlinear biases indeed get amplified over time.

Select dataset (use only MT for now):

dx = signals\_MT  
  
# remove cases without values  
dx = dx[!is.na(dx$prop.time.plateau),]

Some stats:

# Datapoints:  
nrow(dx)

## [1] 936

# Number of chains:  
length(unique(dx$chain2))

## [1] 39

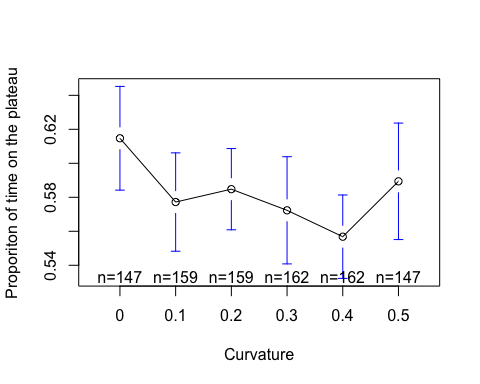
# Number of participants:  
length((unique(paste(dx$chain2,dx$gen))))

## [1] 312

## Average time on the plateau

Plot average time on the plateau by curavture value.

plotmeans(prop.time.plateau~curvature, data=dx, ylab="Proporiton of time on the plateau", xlab='Curvature')



Center variables. Note that this changes the values of curvature and

dx$prop.time.plateau.norm = dx$prop.time.plateau - mean(dx$prop.time.plateau)  
dx$gen = dx$gen - 5  
dx$curvature.center = dx$curvature - 0.2

m0= lmer(prop.time.plateau.norm~  
 1 +   
 (1 | chain2) +   
 (1 | meaning),   
 data = dx)  
  
m1= lmer(prop.time.plateau.norm~  
 1 +   
 (1 + curvature.center | chain2) +   
 (1 | meaning),   
 data = dx)  
  
anova(m0,m1)

## refitting model(s) with ML (instead of REML)

## Data: dx  
## Models:  
## m0: prop.time.plateau.norm ~ 1 + (1 | chain2) + (1 | meaning)  
## m1: prop.time.plateau.norm ~ 1 + (1 + curvature.center | chain2) +   
## m1: (1 | meaning)  
## Df AIC BIC logLik deviance Chisq Chi Df Pr(>Chisq)  
## m0 4 -516.07 -496.70 262.03 -524.07   
## m1 6 -514.83 -485.78 263.42 -526.83 2.7629 2 0.2512

# No signficiant difference  
m2= lmer(prop.time.plateau.norm~  
 1 +   
 (1 + gen | chain2) +   
 (1 | meaning),   
 data = dx)  
  
anova(m0,m2)

## refitting model(s) with ML (instead of REML)

## Data: dx  
## Models:  
## m0: prop.time.plateau.norm ~ 1 + (1 | chain2) + (1 | meaning)  
## m2: prop.time.plateau.norm ~ 1 + (1 + gen | chain2) + (1 | meaning)  
## Df AIC BIC logLik deviance Chisq Chi Df Pr(>Chisq)   
## m0 4 -516.07 -496.70 262.03 -524.07   
## m2 6 -521.01 -491.96 266.50 -533.01 8.9387 2 0.01145 \*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

# Yes, keep  
  
m3= lmer(prop.time.plateau.norm~  
 1 +   
 (1 + gen | chain2) +   
 (1 + curvature.center| meaning),   
 data = dx)  
  
anova(m2,m3)

## refitting model(s) with ML (instead of REML)

## Data: dx  
## Models:  
## m2: prop.time.plateau.norm ~ 1 + (1 + gen | chain2) + (1 | meaning)  
## m3: prop.time.plateau.norm ~ 1 + (1 + gen | chain2) + (1 + curvature.center |   
## m3: meaning)  
## Df AIC BIC logLik deviance Chisq Chi Df Pr(>Chisq)  
## m2 6 -521.01 -491.96 266.50 -533.01   
## m3 8 -517.98 -479.25 266.99 -533.98 0.9742 2 0.6144

# No significant improvement  
  
m4= lmer(prop.time.plateau.norm~  
 1 +   
 (1 + gen | chain2) +   
 (1 + gen | meaning),   
 data = dx)  
anova(m2,m4)

## refitting model(s) with ML (instead of REML)

## Data: dx  
## Models:  
## m2: prop.time.plateau.norm ~ 1 + (1 + gen | chain2) + (1 | meaning)  
## m4: prop.time.plateau.norm ~ 1 + (1 + gen | chain2) + (1 + gen |   
## m4: meaning)  
## Df AIC BIC logLik deviance Chisq Chi Df Pr(>Chisq)  
## m2 6 -521.01 -491.96 266.50 -533.01   
## m4 8 -518.42 -479.69 267.21 -534.42 1.4141 2 0.4931

# No significant difference

From the tests above, we determine that only a random slope by chain is required.

# Null model  
# Note: the full null model with all random slopes   
# has convergence issues  
m0= lmer(prop.time.plateau.norm~  
 1 +   
 (1 + gen | chain2) +   
 (1 | meaning),   
 data = dx)  
  
# add curavture  
m1= lmer(prop.time.plateau.norm~  
 1 +   
 curvature.center +  
 (1 + gen | chain2) +   
 (1 | meaning),   
 data = dx)  
# add generation  
m2= lmer(prop.time.plateau.norm~  
 1 +   
 curvature.center +  
 gen +  
 (1 + gen | chain2) +   
 (1 | meaning),   
 data = dx,  
 control = lmerControl(optimizer = "Nelder\_Mead", optCtrl = list(maxfun=500000)))  
# add interaction between curvature and generation  
m3= lmer(prop.time.plateau.norm~  
 1 +   
 curvature.center +  
 gen +  
 curvature : gen +  
 (1 + gen | chain2) +   
 (1 | meaning),   
 data = dx)  
# Quadratic term of curvature  
m4= lmer(prop.time.plateau.norm~  
 1 +   
 curvature.center +  
 gen +  
 curvature : gen +  
 I(curvature^2) +  
 (1 + gen | chain2) +   
 (1 | meaning),   
 data = dx)  
# Interaction between quadratic term of curvature and generation  
m5= lmer(prop.time.plateau.norm~  
 1 +   
 curvature.center +  
 gen +  
 curvature : gen +  
 I(curvature^2) +  
 I(curvature^2):gen +  
 (1 + gen | chain2) +   
 (1 | meaning),   
 data = dx)

Use model comparison to test the effect of each variable.

anova(m0,m1,m2,m3,m4,m5)

## refitting model(s) with ML (instead of REML)

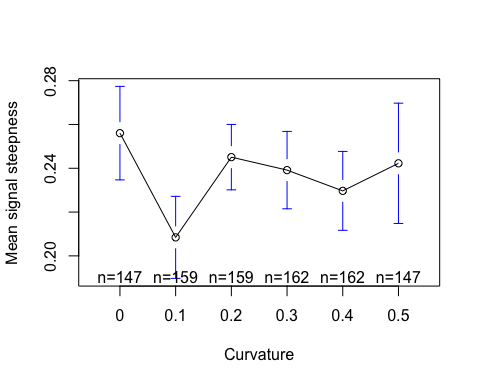
## Data: dx  
## Models:  
## m0: prop.time.plateau.norm ~ 1 + (1 + gen | chain2) + (1 | meaning)  
## m1: prop.time.plateau.norm ~ 1 + curvature.center + (1 + gen | chain2) +   
## m1: (1 | meaning)  
## m2: prop.time.plateau.norm ~ 1 + curvature.center + gen + (1 + gen |   
## m2: chain2) + (1 | meaning)  
## m3: prop.time.plateau.norm ~ 1 + curvature.center + gen + curvature:gen +   
## m3: (1 + gen | chain2) + (1 | meaning)  
## m4: prop.time.plateau.norm ~ 1 + curvature.center + gen + curvature:gen +   
## m4: I(curvature^2) + (1 + gen | chain2) + (1 | meaning)  
## m5: prop.time.plateau.norm ~ 1 + curvature.center + gen + curvature:gen +   
## m5: I(curvature^2) + I(curvature^2):gen + (1 + gen | chain2) +   
## m5: (1 | meaning)  
## Df AIC BIC logLik deviance Chisq Chi Df Pr(>Chisq)   
## m0 6 -521.01 -491.96 266.50 -533.01   
## m1 7 -521.09 -487.19 267.54 -535.09 2.0800 1 0.14924   
## m2 8 -520.95 -482.22 268.48 -536.95 1.8693 1 0.17155   
## m3 9 -518.98 -475.40 268.49 -536.98 0.0205 1 0.88617   
## m4 10 -519.93 -471.52 269.97 -539.93 2.9560 1 0.08556 .  
## m5 11 -518.86 -465.60 270.43 -540.86 0.9301 1 0.33485   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## Steepness measure

Build a series of lmer models predicting the average steepness of signals (normalised using the maximum steepenss values) by curvature and generation. We add a random effect for chain and meaning, with random slopes for curature and generation by chain.

Plot mean steepenss by curvature.

plotmeans(steepness.sig.mean.max~curvature, data=dx, ylab="Mean signal steepness", xlab='Curvature')



Normalise variables:

dx$steepness.sig.mean.max.norm = dx$steepness.sig.mean.max - mean(dx$steepness.sig.mean.max)

Choose random effects:

m0= lmer(steepness.sig.mean.max.norm~  
 1 +   
 (1 | chain2) +   
 (1 | meaning),   
 data = dx)  
  
m1= lmer(steepness.sig.mean.max.norm~  
 1 +   
 (1 + curvature.center | chain2) +   
 (1 | meaning),   
 data = dx)  
  
anova(m0,m1)

## refitting model(s) with ML (instead of REML)

## Data: dx  
## Models:  
## m0: steepness.sig.mean.max.norm ~ 1 + (1 | chain2) + (1 | meaning)  
## m1: steepness.sig.mean.max.norm ~ 1 + (1 + curvature.center | chain2) +   
## m1: (1 | meaning)  
## Df AIC BIC logLik deviance Chisq Chi Df Pr(>Chisq)   
## m0 4 -1281.8 -1262.4 644.90 -1289.8   
## m1 6 -1288.9 -1259.9 650.45 -1300.9 11.115 2 0.003859 \*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

# Significant difference  
  
  
m2= lmer(steepness.sig.mean.max.norm~  
 1 +   
 (1 + curvature.center + gen | chain2) +   
 (1 | meaning),   
 data = dx)  
  
anova(m1,m2)

## refitting model(s) with ML (instead of REML)

## Data: dx  
## Models:  
## m1: steepness.sig.mean.max.norm ~ 1 + (1 + curvature.center | chain2) +   
## m1: (1 | meaning)  
## m2: steepness.sig.mean.max.norm ~ 1 + (1 + curvature.center + gen |   
## m2: chain2) + (1 | meaning)  
## Df AIC BIC logLik deviance Chisq Chi Df Pr(>Chisq)  
## m1 6 -1288.9 -1259.9 650.45 -1300.9   
## m2 9 -1287.6 -1244.0 652.79 -1305.6 4.6834 3 0.1965

# No significant difference  
  
m3= lmer(prop.time.plateau.norm~  
 1 +   
 (1 + curvature.center | chain2) +   
 (1 + curvature.center| meaning),   
 data = dx)  
  
anova(m1,m3)

## refitting model(s) with ML (instead of REML)

## Data: dx  
## Models:  
## m1: steepness.sig.mean.max.norm ~ 1 + (1 + curvature.center | chain2) +   
## m1: (1 | meaning)  
## m3: prop.time.plateau.norm ~ 1 + (1 + curvature.center | chain2) +   
## m3: (1 + curvature.center | meaning)  
## Df AIC BIC logLik deviance Chisq Chi Df Pr(>Chisq)  
## m1 6 -1288.91 -1259.86 650.45 -1300.91   
## m3 8 -511.71 -472.98 263.86 -527.71 0 2 1

# No significant improvement  
  
m4= lmer(prop.time.plateau.norm~  
 1 +   
 (1 + curvature.center | chain2) +   
 (1 + gen | meaning),   
 data = dx)  
anova(m1,m4)

## refitting model(s) with ML (instead of REML)

## Data: dx  
## Models:  
## m1: steepness.sig.mean.max.norm ~ 1 + (1 + curvature.center | chain2) +   
## m1: (1 | meaning)  
## m4: prop.time.plateau.norm ~ 1 + (1 + curvature.center | chain2) +   
## m4: (1 + gen | meaning)  
## Df AIC BIC logLik deviance Chisq Chi Df Pr(>Chisq)  
## m1 6 -1288.91 -1259.86 650.45 -1300.91   
## m4 8 -512.29 -473.56 264.15 -528.29 0 2 1

# No significant difference

Keep only a random slope by curvature.

# Null model  
m0= lmer(steepness.sig.mean.max.norm~  
 1 +   
 (1 + curvature.center | chain2) +   
 (1 | meaning),   
 data = dx)  
  
# add curavture  
m1= lmer(steepness.sig.mean.max.norm~  
 1 +   
 curvature.center +  
 (1 + curvature.center | chain2) +   
 (1 | meaning),   
 data = dx)  
# add generation  
m2= lmer(steepness.sig.mean.max.norm~  
 1 +   
 curvature.center +  
 gen +  
 (1 + curvature.center | chain2) +   
 (1 | meaning),   
 data = dx)  
# add interaction between curvature and generation  
m3= lmer(steepness.sig.mean.max.norm~  
 1 +   
 curvature.center +  
 gen +  
 curvature : gen +  
 (1 + curvature.center | chain2) +   
 (1 | meaning),   
 data = dx)  
# Quadratic term of curvature  
m4= lmer(steepness.sig.mean.max.norm~  
 1 +   
 curvature.center +  
 gen +  
 curvature : gen +  
 I(curvature^2) +  
 (1 + curvature.center | chain2) +   
 (1 | meaning),   
 data = dx)  
  
# Interaction between quadratic term of curvature and generation  
m5= lmer(steepness.sig.mean.max.norm~  
 1 +   
 curvature.center +  
 gen +  
 curvature : gen +  
 I(curvature^2) +  
 I(curvature^2):gen +  
 (1 + curvature.center | chain2) +   
 (1 | meaning),   
 data = dx)

Use model comparison to test the effect of each variable.

anova(m0,m1,m2,m3,m4,m5)

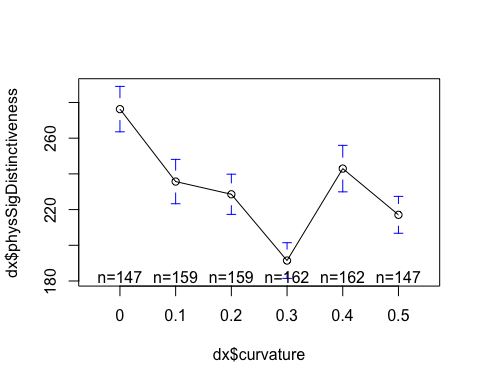
## refitting model(s) with ML (instead of REML)

## Data: dx  
## Models:  
## m0: steepness.sig.mean.max.norm ~ 1 + (1 + curvature.center | chain2) +   
## m0: (1 | meaning)  
## m1: steepness.sig.mean.max.norm ~ 1 + curvature.center + (1 + curvature.center |   
## m1: chain2) + (1 | meaning)  
## m2: steepness.sig.mean.max.norm ~ 1 + curvature.center + gen + (1 +   
## m2: curvature.center | chain2) + (1 | meaning)  
## m3: steepness.sig.mean.max.norm ~ 1 + curvature.center + gen + curvature:gen +   
## m3: (1 + curvature.center | chain2) + (1 | meaning)  
## m4: steepness.sig.mean.max.norm ~ 1 + curvature.center + gen + curvature:gen +   
## m4: I(curvature^2) + (1 + curvature.center | chain2) + (1 | meaning)  
## m5: steepness.sig.mean.max.norm ~ 1 + curvature.center + gen + curvature:gen +   
## m5: I(curvature^2) + I(curvature^2):gen + (1 + curvature.center |   
## m5: chain2) + (1 | meaning)  
## Df AIC BIC logLik deviance Chisq Chi Df Pr(>Chisq)   
## m0 6 -1288.9 -1259.9 650.45 -1300.9   
## m1 7 -1286.9 -1253.0 650.45 -1300.9 0.0005 1 0.98274   
## m2 8 -1285.0 -1246.2 650.48 -1301.0 0.0512 1 0.82100   
## m3 9 -1283.0 -1239.5 650.51 -1301.0 0.0664 1 0.79666   
## m4 10 -1281.0 -1232.6 650.51 -1301.0 0.0002 1 0.99003   
## m5 11 -1284.1 -1230.8 653.04 -1306.1 5.0566 1 0.02453 \*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

# Prediction 2: Curvature affect variability

Since stable regions in the articulator space allow for articulator-invariant alternatives to produce the same speech sound, we predict a high degree of articulator variability compared to acoustic variability for stronger biases and later generations.

plotmeans(dx$physSigDistinctiveness~dx$curvature)



Scale variables:

dx$physSigDistinctiveness.norm = dx$physSigDistinctiveness -  
 mean(dx$physSigDistinctiveness)  
dx$physSigDistinctiveness.norm = dx$physSigDistinctiveness.norm /  
 abs(max(dx$physSigDistinctiveness.norm))

Choose random effects structure:

m0= lmer(physSigDistinctiveness.norm~  
 1 +   
 (1 | chain2) +   
 (1 | meaning),   
 data = dx)  
  
m1= lmer(physSigDistinctiveness.norm~  
 1 +   
 (1 + curvature.center | chain2) +   
 (1 | meaning),   
 data = dx)  
  
anova(m0,m1)

## refitting model(s) with ML (instead of REML)

## Data: dx  
## Models:  
## m0: physSigDistinctiveness.norm ~ 1 + (1 | chain2) + (1 | meaning)  
## m1: physSigDistinctiveness.norm ~ 1 + (1 + curvature.center | chain2) +   
## m1: (1 | meaning)  
## Df AIC BIC logLik deviance Chisq Chi Df Pr(>Chisq)  
## m0 4 420.18 439.54 -206.09 412.18   
## m1 6 421.66 450.71 -204.83 409.66 2.5185 2 0.2839

# No significant difference  
  
  
m2= lmer(physSigDistinctiveness.norm~  
 1 +   
 (1 + gen | chain2) +   
 (1 | meaning),   
 data = dx)  
  
anova(m0,m2)

## refitting model(s) with ML (instead of REML)

## Data: dx  
## Models:  
## m0: physSigDistinctiveness.norm ~ 1 + (1 | chain2) + (1 | meaning)  
## m2: physSigDistinctiveness.norm ~ 1 + (1 + gen | chain2) + (1 | meaning)  
## Df AIC BIC logLik deviance Chisq Chi Df Pr(>Chisq)   
## m0 4 420.18 439.54 -206.09 412.18   
## m2 6 214.25 243.29 -101.12 202.25 209.93 2 < 2.2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

# Significant difference  
  
m3= lmer(physSigDistinctiveness.norm~  
 1 +   
 (1 + gen | chain2) +   
 (1 + curvature.center| meaning),   
 data = dx)  
  
anova(m2,m3)

## refitting model(s) with ML (instead of REML)

## Data: dx  
## Models:  
## m2: physSigDistinctiveness.norm ~ 1 + (1 + gen | chain2) + (1 | meaning)  
## m3: physSigDistinctiveness.norm ~ 1 + (1 + gen | chain2) + (1 + curvature.center |   
## m3: meaning)  
## Df AIC BIC logLik deviance Chisq Chi Df Pr(>Chisq)  
## m2 6 214.25 243.29 -101.12 202.25   
## m3 8 218.25 256.98 -101.12 202.25 0 2 1

# No significant improvement  
  
m4= lmer(physSigDistinctiveness.norm~  
 1 +   
 (1 + gen | chain2) +   
 (1 + gen | meaning),   
 data = dx)  
anova(m2,m4)

## refitting model(s) with ML (instead of REML)

## Data: dx  
## Models:  
## m2: physSigDistinctiveness.norm ~ 1 + (1 + gen | chain2) + (1 | meaning)  
## m4: physSigDistinctiveness.norm ~ 1 + (1 + gen | chain2) + (1 + gen |   
## m4: meaning)  
## Df AIC BIC logLik deviance Chisq Chi Df Pr(>Chisq)  
## m2 6 214.25 243.29 -101.12 202.25   
## m4 8 218.25 256.98 -101.12 202.25 0 2 1

# No significant difference

Keep only random slope for gen by chain.

# Null model  
m0= lmer(physSigDistinctiveness.norm~  
 1 +   
 (1 + gen | chain2) +   
 (1 | meaning),   
 data = dx)  
  
# add curavture  
m1= lmer(physSigDistinctiveness.norm~  
 1 +   
 curvature.center +  
 (1 + gen | chain2) +   
 (1 | meaning),   
 data = dx)  
# add generation  
m2= lmer(physSigDistinctiveness.norm~  
 1 +   
 curvature.center +  
 gen +  
 (1 + gen | chain2) +   
 (1 | meaning),   
 data = dx)  
# add interaction between curvature and generation  
m3= lmer(physSigDistinctiveness.norm~  
 1 +   
 curvature.center +  
 gen +  
 curvature : gen +  
 (1 + gen | chain2) +   
 (1 | meaning),   
 data = dx)  
# Quadratic term of curvature  
m4= lmer(physSigDistinctiveness.norm~  
 1 +   
 curvature.center +  
 gen +  
 curvature : gen +  
 I(curvature^2) +  
 (1 + gen | chain2) +   
 (1 | meaning),   
 data = dx)  
  
# Interaction between quadratic term of curvature and generation  
m5= lmer(physSigDistinctiveness.norm~  
 1 +   
 curvature.center +  
 gen +  
 curvature : gen +  
 I(curvature^2) +  
 I(curvature^2):gen +  
 (1 + gen | chain2) +   
 (1 | meaning),   
 data = dx)

Use model comparison to test the effect of each variable.

anova(m0,m1,m2,m3,m4,m5)

## refitting model(s) with ML (instead of REML)

## Data: dx  
## Models:  
## m0: physSigDistinctiveness.norm ~ 1 + (1 + gen | chain2) + (1 | meaning)  
## m1: physSigDistinctiveness.norm ~ 1 + curvature.center + (1 + gen |   
## m1: chain2) + (1 | meaning)  
## m2: physSigDistinctiveness.norm ~ 1 + curvature.center + gen + (1 +   
## m2: gen | chain2) + (1 | meaning)  
## m3: physSigDistinctiveness.norm ~ 1 + curvature.center + gen + curvature:gen +   
## m3: (1 + gen | chain2) + (1 | meaning)  
## m4: physSigDistinctiveness.norm ~ 1 + curvature.center + gen + curvature:gen +   
## m4: I(curvature^2) + (1 + gen | chain2) + (1 | meaning)  
## m5: physSigDistinctiveness.norm ~ 1 + curvature.center + gen + curvature:gen +   
## m5: I(curvature^2) + I(curvature^2):gen + (1 + gen | chain2) +   
## m5: (1 | meaning)  
## Df AIC BIC logLik deviance Chisq Chi Df Pr(>Chisq)  
## m0 6 214.25 243.29 -101.123 202.25   
## m1 7 214.75 248.64 -100.376 200.75 1.4939 1 0.2216  
## m2 8 216.72 255.46 -100.363 200.72 0.0259 1 0.8722  
## m3 9 217.99 261.57 -99.996 199.99 0.7330 1 0.3919  
## m4 10 217.80 266.22 -98.902 197.80 2.1887 1 0.1390  
## m5 11 219.80 273.06 -98.900 197.80 0.0028 1 0.9580

# Summary

The predictions are not supported using this approach. But there is much to discuss:

* Are the stats choices above suitable? As far as I can tell, there are no qualitative differences using a range of random effect structures, althoug the maximal structure causes some convergence problems.
* Are we using the right measures? If I use the variable steepness.sig.mean instead of steepness.sig.mean.max, then there is a significant quadratic effect of curvature. But little difference otherwise.
* Is the data too noisy? Maybe we need to look at whether the any meanings are swapping, or look at the SONA data?