

```

                                anonlong_180908_wolf.txt
#praat script: anonymise_long_sound.praat
#version: [2008:05:25]
#author: Daniel Hirst
#email: daniel.hirst@lpl-aix.fr

# Revision 2008:09:18
# author: Wolf Fruh
# Some tests incorporated to deal with cases when the key word cuts across a
'section' boundary.
# In those cases, no 'myRight' is created or appended to the 'mySound_part' in
the section which ends with the key word
# and no 'myRight' is created or used to start off the 'mySound_part' which
starts with the key word
# Added lines on lines 139-143, 157 - 172, 175 - 178, 185 - 188

#purpose: replace portions of a long sound which are labelled with a key word on
the accompanying TextGrid
#           with a hum sound with the same prosodic characteristics as the
original sound

#requires: the script should be in the same folder as the Long_Sounds to be
anonymised
#           each sound should be accompanied by a TextGrid with the same
name

form anonymise_long_sound
  word sound_extension .wav
  word textGrid_extension .textGrid
  word anonymised_extension _anon.wav
  natural target_tier 1
  word target_label buzz
  comment duration of section for analysis (in secs)
  positive section 30
  positive timestep 0.01
  boolean automatic_max_and_min yes
  natural minimum_f0 60
  natural maximum_f0 700
  comment use this to lower overall intensity if necessary
  positive scale_intensity 0.9
endform

clearinfo

mySounds = Create Strings as file list... sounds *'sound_extension$'
nSounds = Get number of strings

for iSound to nSounds
  select mySounds
  sound$ = Get string... iSound

  if not endsWith(sound$, anonymised_extension$)
    name$ = sound$ - sound_extension$
    textGrid$ = name$ + textGrid_extension$
    anonymised_sound$ = name$ + anonymised_extension$

    if not fileReadable(textGrid$)
      printline Can't find TextGrid file for 'name$'
    else
      call treat_sound
    endif
  endif
endfor

select mySounds
Remove

```

```

procedure treat_sound
  mySound = Open long sound file... 'sound$'
  sound_duration = Get total duration
  myTextGrid = Read from file... 'textGrid$'
  select mySound
  part_end = 0
  part = 0

  repeat
    part = part+1
    part_start = part_end
    part_end = part_end + section

    if part_end > sound_duration
      part_end = sound_duration
    endif

    select mySound
    call treat_part
  until part_end = sound_duration

  myNew_sound = Open long sound file... 'anonymised_sound$'
  pause - Click to continue
  select mySound
  plus myNew_sound
  plus myTextGrid
  Remove
endproc

procedure treat_part
  mySound_part_temp = Extract part... part_start part_end no
  mySound_part = Convert to mono
  intensity = Get intensity (dB)
  scaled_intensity = intensity * scale_intensity
  Scale intensity... scaled_intensity

  select myTextGrid
  myTextGrid_part = Extract part... part_start part_end no

  nIntervals = Get number of intervals... target_tier

  for iInterval to nIntervals
    select myTextGrid_part
    label$ = Get label of interval... target_tier iInterval
    if label$ = target_label$
      call treat_word
    endif
  endfor

  select mySound_part

  if part = 1
    Write to WAV file... 'anonymised_sound$'
  else
    Append to existing sound file... 'anonymised_sound$'
  endif
  plus mySound_part_temp
  plus myTextGrid_part
  Remove
endproc

procedure treat_word
  if automatic_max_and_min
    select mySound_part
    call calculate_min_max_f0

```

anonlong_180908_wolf.txt

```
else
    min_f0 = minimum_f0
    max_f0 = maximum_f0
endif

select myTextGrid_part
word_start = Get starting point... target_tier iInterval
word_end = Get end point... target_tier iInterval
word_duration = word_end - word_start
select mySound_part
# define a section at the beginning of the part which is before the 'hum' but do
that only if there
# is actually a bit before the hum (there is no hum if the 'buzz' cuts across
two parts
    if word_start > 0
        myLeft = Extract part... 0 word_start rectangular 1 no
    endif
    select mySound_part
    myWord = Extract part... word_start word_end rectangular 1 no
    myScale = Get intensity (dB)
    myPitch = To Pitch... timestep min_f0 max_f0
    myHum = To Sound (hum)
    select myword
    myIntensity = To Intensity... min_f0 timestep no
    myIntensityTier = Down to IntensityTier
    plus myHum
    myNewHum = Multiply... yes
    Scale intensity... myScale
    select mySound_part
!pause 'word_end' 'section'
# Define a section after the Hum to complete the part, but only if the hum does
not go to the end of the part
    if word_end < section
        myRight = Extract part... word_end section rectangular 1 no
    endif
# Create the new sound file by concatenating all parts.
# Start with 'myLeft' if it exists, otherwise start with 'myNewHum'
    if word_start > 0
        select myLeft
        plus myNewHum
    else
        select myNewHum
    endif
# Define a section after the Hum to complete the part, but only if the hum does
not go to the end of the part
    if word_end < section
        plus myRight
    endif
    myNew_part = Concatenate
    select mySound_part
# again, add myLeft but only if it exists
    if word_start > 0
        plus myLeft
    endif
    plus myword
    plus myPitch
    plus myHum
    plus myIntensity
    plus myIntensityTier
    plus myNewHum
# Define a section after the Hum to complete the part, but only if the hum does
not go to the end of the part
    if word_end < section
        plus myRight
    endif
    Remove
    mySound_part = myNew_part
    printline ['part_start'..'part_end'] treating word 'target_label$' in
```

```
interval 'iInterval'  
endproc  
  
procedure calculate_min_max_f0  
# estimate of newMaxF0 as 1.5 * quantile 75  
# and newMinF0 as 0.5 * quantile 25  
# rounded to higher (resp. lower) 10  
  To Pitch... 'timestep' 'minimum_f0' 'maximum_f0'  
  .q75 = Get quantile... 0.0 0.0 0.75 Hertz  
  .q25 = Get quantile... 0.0 0.0 0.25 Hertz  
  max_f0 = 10*ceiling((1.5*.q75)/10)  
  min_f0 = 10*floor((0.75*.q25)/10)  
  Remove  
endproc
```