vowel_duration_python

May 12, 2022

0.1 Materials

At the start, we have audio and annotated textgrids of **regilaul** songs, annotated for ictus/off-ictus and phrase text, then force-aligned using Praat's built in eSpeak forced aligner for Estonian to word and then segment. Then, we use the estnltk vabamorf package to syllabify the words so that we can annotate the textgrid further with syllable quantity (Estonian has 3) and whether or not it is accented at the word level. We end up with a dataframe containing the data from three of the(Interval) tiers of the textgrid, acquiring duration data for words, individual segments, and (eventually) syllables.

[ ]:
```
import parselmouth
from estnltk.vabamorf.morf import syllabify_word
import tgt
import string
#test method on a single TextGrid:
gridDir2 = "/Users/sarah/qp_final/txtgridtest/69.TextGrid"


def get_duration_labels(textgrid, tiername1,tiername2,tiername3):
    tmp = tgt.read_textgrid(textgrid)
    mytier = tmp.get_tier_by_name(tiername1)
    other = tmp.get_tier_by_name(tiername2)
    ictus = tmp.get_tier_by_name(tiername3)
    segments = []
    word_dur = mytier.intervals

    for interval in word_dur:
        h = interval.start_time
        t = interval.end_time
        b = t-h
        l = interval.text
        tmpseg = [other.get_annotations_between_timepoints(h,t)]
        s = syllabify_word(l,as_dict=True)
        i = 0
        for list in tmpseg:
            n = 0
            while i < len(s):
                item = s[i]
```

```
                shape = item.get('syllable')
                q = item.get('quantity')
                a = item.get('accent')
                geminate = False
                for char in shape.strip(string.punctuation):
                    if geminate: continue
                    if n < len(list):
                        myinterval = list[n]
                        c = myinterval.text
                        d = myinterval.start_time
                        g = myinterval.end_time
                        tmpick = ictus.
 ↪get_annotations_between_timepoints(d,g,left_overlap=True,right_overlap=True)
                        if len(tmpick) == 0: ick = "off"
                        else: ick = "ictus"
                        #segment duration
                        j = g-d
                        #segment midpoint for later measurements
                        mid = g - (j/2)
                        if " " in c:
                            row = [l,b,shape,q,a,c,j,mid,ick]
                            segments.append(row)
                            geminate = True

                        elif char == c:

                            row = [l,b,shape,q,a,c,j,mid,ick]
                            segments.append(row)
                        else :
                            can = "(" + char + ") " + c
                            row = [l,b,shape,q,a,can,j,mid,ick]
                            segments.append(row)
                        n+=1
                i+= 1




    nu_df = pd.
 ↪DataFrame(segments,columns=["word","word_dur","syll","quantity","stress","segment","seg_dur
    return nu_df

syl_dur_df = get_duration_labels(gridDir2,"word","word/phon","ictus")
syl_dur_df.head()
```

```
[ ]:      word  word_dur syll  quantity  stress segment  seg_duration  \
     0      Oh  0.240882   oh         3       1       o      0.165043
     1      Oh  0.240882   oh         3       1       h      0.075839
     2      me  0.169500   me         3       1       m      0.059664
     3      me  0.169500   me         3       1       e      0.109836
     4  vaesed  0.619500  vae         2       1       v      0.101258

        seg_midpoint  ictus
     0      0.082521  ictus
     1      0.202962  ictus
     2      0.270713  ictus
     3      0.355463  ictus
     4      0.461010  ictus
```

0.2 Adding Spectral data

now that we have the duration data from the textgrid, we can query specific timepoints for information about the acoustic signal. The following function uses the midpoint (which we snagged while we were making the dataframe above) and get the first three formants(Hz) for each segment.

```
[ ]: import parselmouth

     test = "/Users/sarah/qp_final/wavs/069.wav"

     def get_formants(syl_dur_df, wave):
         song = parselmouth.Sound(wave)
         formant = song.to_formant_burg()
         f1 = []
         f2 = []
         f3 = []
         for float in syl_dur_df.seg_midpoint:
             time = float
             f1.append(formant.get_value_at_time(1,time))
             f2.append(formant.get_value_at_time(2, time))
             f3.append(formant.get_value_at_time(3,time))
         syl_dur_df["f1"] = f1
         syl_dur_df["f2"] = f2
         syl_dur_df["f3"] = f3
         return syl_dur_df
     nu_df = get_formants(syl_dur_df,test)
     nu_df.head()
```

```
[ ]:      word  word_dur syll  quantity  stress segment  seg_duration  \
     0      Oh  0.240882   oh         3       1       o      0.165043
     1      Oh  0.240882   oh         3       1       h      0.075839
     2      me  0.169500   me         3       1       m      0.059664
     3      me  0.169500   me         3       1       e      0.109836
     4  vaesed  0.619500  vae         2       1       v      0.101258
```

```
   seg_midpoint  ictus          f1           f2           f3
0      0.082521  ictus  540.905421   989.905678  1414.238432
1      0.202962  ictus  850.865128  1620.784985  2186.523346
2      0.270713  ictus  346.158399  1818.412065  2798.007639
3      0.355463  ictus  411.766557  1385.397329  2214.984918
4      0.461010  ictus  488.221699  1013.678738  1576.019602
```

[ ]:
```
from os.path import join
#runs a for loop over a directory using the above-specified functions

test = "/Users/sarah/qp_final/txtgridtest/"
songs = "/Users/sarah/qp_final/songs/"

for fn in os.listdir(test):
    if '.TextGrid' not in fn:
        continue
    n = fn.strip('.TextGrid')
    wave = join(songs, n + '.wav')
    data_file = open("ictus_forms_" + n +".csv",'w')
    #make a dataframe with the interval tiers of the textgrid
    tmp = pd.DataFrame(get_duration_labels(join(test,fn), "word","word/
 ↪phon","ictus"))
    #add the formant data to the dataframe
    nu_df = get_formants(tmp,wave)
    print(nu_df.head())
    # nu_df.to_csv(data_file)
    # data_file.close()
```

```
    word  word_dur syll  quantity  stress segment  seg_duration  seg_midpoint  \
0  Lõpe,  0.877833   lõ         2       1       l      0.102860      0.206697
1  Lõpe,  0.877833   lõ         2       1    (õ)      0.293082      0.404668
2  Lõpe,  0.877833  pe,         2       0       p      0.191140      0.646779
3  Lõpe,  0.877833  pe,         2       0       e      0.290751      0.887725
4  lõpe,  0.836977   lõ         2       1       l      0.075601      1.070901

   ictus          f1           f2           f3
0  ictus  335.422530  1063.568157  1890.268233
1  ictus  684.001970  1113.986905  1962.573748
2  ictus  408.802542  1208.989186  2321.386963
3    off  758.503513  1717.665908  2219.432604
4  ictus  772.408953  1868.617726  1966.346496
   word  word_dur  syll  quantity  stress segment  seg_duration  seg_midpoint  \
0  miks  0.334037  miks         3       1       m      0.109658      2.583638
1  miks  0.334037  miks         3       1       i      0.103965      2.690450
2  miks  0.334037  miks         3       1       k      0.071640      2.778252
3  miks  0.334037  miks         3       1       s      0.048775      2.838459
4    sa  0.174934    sa         3       1       s      0.049593      2.887643
```

```
   ictus           f1           f2           f3
0  ictus  1015.032176  1668.628090  2451.566636
1  ictus  1018.542344  1242.393701  2614.593523
2  ictus   739.889853  1206.829336  2557.715244
3  ictus  1219.241465  1330.686940  2551.563447
4    off   996.088003  1411.057879  2533.755085
    word  word_dur syll  quantity  stress segment  seg_duration  seg_midpoint  \
0  Kelle   0.54837  kel         2       1       k      0.004193      0.002097
1  Kelle   0.54837  kel         2       1       e      0.224554      0.116470
2  Kelle   0.54837  kel         2       1       l      0.026000      0.241747
3  Kelle   0.54837   le         1       0       l      0.105220      0.307357
4  Kelle   0.54837   le         1       0       e      0.188403      0.454169

   ictus           f1           f2           f3
0  ictus          NaN          NaN          NaN
1  ictus  1110.553772  2154.184673  3278.566300
2    off   723.585094  1951.030424  3121.543094
3    off   956.257232  1943.928928  3170.558668
4    off   699.571764  1442.320280  2391.975132
    word  word_dur   syll  quantity  stress segment  seg_duration  \
0  Kuus,  0.690416  kuus,         3       1       k      0.142235
1  Kuus,  0.690416  kuus,         3       1       u      0.232104
2  Kuus,  0.690416  kuus,         3       1      u      0.161435
3  kuus,  0.661647  kuus,         3       1       k      0.100761
4  kuus,  0.661647  kuus,         3       1       u      0.293682

   seg_midpoint  ictus           f1           f2           f3
0      0.071117  ictus  1028.357905  2289.070707  3278.949044
1      0.258286  ictus  1218.143866  2223.410006  3181.682865
2      0.455056    off   355.504806  1076.764366  1934.314752
3      0.740797  ictus   680.358222  1148.112933  1979.469484
4      0.938018  ictus   669.587600  1146.386219  1664.458201
     word  word_dur syll  quantity  stress segment  seg_duration  \
0  Laula,  0.616537  lau         2       1       l      0.138245
1  Laula,  0.616537  lau         2       1       a      0.156972
2  Laula,  0.616537  lau         2       1       u      0.077000
3  Laula,  0.616537  la,         2       0       l      0.082000
4  Laula,  0.616537  la,         2       0       a      0.162320

   seg_midpoint  ictus          f1           f2           f3
0      0.441088  ictus  580.289721  1408.585781  2501.362541
1      0.588696  ictus  599.037686  1153.155750  1944.133038
2      0.705683  ictus  641.731079  1076.715564  2487.946642
3      0.785183  ictus  529.871527  1631.906656  1976.346982
4      0.907342    off  618.232042  1140.675032  1746.136521
    word  word_dur syll  quantity  stress segment  seg_duration  seg_midpoint  \
0  Laske  0.700087  las         2       1       l      0.068795      0.034398
```

```
1  Laske  0.700087  las         2       1       a      0.221538      0.179564
2  Laske  0.700087  las         2       1       s      0.074939      0.327802
3  Laske  0.700087   ke         1       0       k      0.102666      0.416605
4  Laske  0.700087   ke         1       0       e      0.232149      0.584012

   ictus           f1           f2           f3
0  ictus  1472.974201  2404.084622  3447.483334
1  ictus  1125.029770  2441.781428  3331.398284
2  ictus   854.440862  1641.705923  2960.910199
3  ictus   686.227370  1665.848474  3056.221222
4  ictus   895.423885  1680.387334  1883.316265
    word  word_dur syll  quantity  stress segment  seg_duration  seg_midpoint  \
0  pandi   0.45674  pan         2       1       p      0.047328      6.835351
1  pandi   0.45674  pan         2       1       a      0.177026      6.947528
2  pandi   0.45674  pan         2       1   (n)        0.041698      7.056890
3  pandi   0.45674   di         1       0       d      0.059916      7.107697
4  pandi   0.45674   di         1       0       i      0.129039      7.202174

   ictus           f1           f2           f3
0  ictus   605.862912  1068.944908  2291.317753
1  ictus   603.056801  1303.781807  2625.465878
2  ictus   765.874507  1279.851220  2546.395265
3    off   930.416284  1876.949848  2542.774896
4    off  1074.849998  2167.095173  2671.989524
     word  word_dur syll  quantity  stress segment  seg_duration  \
0      Oh  0.240882   oh         3       1       o      0.165043
1      Oh  0.240882   oh         3       1       h      0.075839
2      me  0.169500   me         3       1       m      0.059664
3      me  0.169500   me         3       1       e      0.109836
4  vaesed  0.619500  vae         2       1       v      0.101258

   seg_midpoint  ictus          f1           f2           f3
0      0.082521  ictus  889.425959  2009.675238  3152.670429
1      0.202962  ictus  639.966111  1118.912389  2929.645146
2      0.270713  ictus  464.249379   840.111759  1332.896822
3      0.355463  ictus  360.465479  1015.163348  1450.565657
4      0.461010  ictus  504.486236  1459.123023  2328.845279
             word  word_dur syll  quantity  stress segment  seg_duration  \
0  Ui-sui-sui-sui  1.297146   ui         3       1       u      0.103226
1  Ui-sui-sui-sui  1.297146   ui         3       1      i      0.136909
2  Ui-sui-sui-sui  1.297146  sui         3       1       s      0.065000
3  Ui-sui-sui-sui  1.297146  sui         3       1       u      0.101500
4  Ui-sui-sui-sui  1.297146  sui         3       1      i      0.138257

   seg_midpoint  ictus          f1           f2           f3
0      0.051613  ictus  735.327729  2002.710633  2764.319990
1      0.171681  ictus  934.145628  1833.703843  2616.755205
2      0.272635  ictus  981.588844  2056.887357  2632.960528
```

```
3      0.355885  ictus  958.855987  1841.503699  2488.421439
4      0.475764    off  470.683808  1001.828609  2784.254879
```

1 Now we put it into a big pile!

Here we concatenate all the data we have so far into one large pandas dataframe. At this point, we can keep annotating songs for the corpus, and as textgrids are finished we can run the scripts above to add them into the larger dataset. We're also gonna take the opportunity to add some metadata to the dataframes: fileid(song) and performer initials as potential grouping factors.

[ ]:
```
import os
import pandas as pd
import statsmodels .formula.api as smf
folder = "/Users/sarah/qp_final/data_glob/"
meta =  pd.read_csv("/Users/sarah/qp_final/song_metadata.csv")


songs_dfs = []
for fn in os.listdir(folder):
    if '.csv' not in fn: continue
    whole_name = os.path.join(folder,fn)
    song_df = pd.read_csv(whole_name)
    fileid1 = fn.strip('ictus_forms_')
    fileid = int(fileid1.strip('.csv'))
    row = meta.index[meta['track'] == fileid].tolist()
    performer = meta.performer[row[0]]
    for index in song_df:
        song_df['fileid'] = fileid
        song_df['performer'] = performer


    songs_dfs.append(song_df)

big_frame = pd.concat(songs_dfs, ignore_index=True)
print(big_frame.describe())
big_frame
```

```
        Unnamed: 0     word_dur     quantity       stress  seg_duration  \
count  1967.000000  1967.000000  1967.000000  1967.000000   1967.000000
mean    198.710727     0.927221     1.887646     0.522623      0.139460
std     192.058558     0.467994     0.649035     0.499615      0.096460
min       0.000000     0.083718     1.000000     0.000000      0.004193
25%      56.000000     0.566898     1.000000     0.000000      0.076177
50%     131.000000     0.819799     2.000000     1.000000      0.109868
75%     259.000000     1.304484     2.000000     1.000000      0.180858
max     734.000000     2.168336     3.000000     1.000000      1.201817

       seg_midpoint           f1           f2           f3       fileid
```

```
count   1967.000000  1966.000000  1966.000000  1966.000000  1967.000000
mean      57.490687   705.838851  1570.350446  2466.377994    43.916116
std       60.299049   202.075224   389.769265   331.870332    29.497152
min        0.002097   275.170888   519.638388  1204.360788     9.000000
25%       14.617182   548.215226  1296.586430  2290.649062    18.000000
50%       32.759004   694.356670  1529.016068  2496.860314    41.000000
75%       82.097020   855.569626  1831.690906  2669.113021    65.000000
max      219.349440  1472.974201  2693.060094  3552.062191    94.000000
```

[ ]:

```
      Unnamed: 0      word  word_dur  syll  quantity  stress segment  \
0              0     pandi  0.456740   pan         2       1       p
1              1     pandi  0.456740   pan         2       1       a
2              2     pandi  0.456740   pan         2       1     (n)
3              3     pandi  0.456740    di         1       0       d
4              4     pandi  0.456740    di         1       0       i
…            …        …       …    …       …      …       …
1962          34  sulased,  0.772673    la         1       0       l
1963          35  sulased,  0.772673    la         1       0     (a)
1964          36  sulased,  0.772673  sed,         3       1       s
1965          37  sulased,  0.772673  sed,         3       1       e
1966          38  sulased,  0.772673  sed,         3       1       d

      seg_duration  seg_midpoint  ictus           f1           f2  \
0         0.047328      6.835351  ictus   605.862912  1068.944908
1         0.177026      6.947528  ictus   603.056801  1303.781807
2         0.041698      7.056890  ictus   765.874507  1279.851220
3         0.059916      7.107697    off   930.416284  1876.949848
4         0.129039      7.202174    off  1074.849998  2167.095173
…              …           …     …          …            …
1962      0.056635      8.946193  ictus   935.586178  2102.100627
1963      0.152365      9.050693  ictus   564.758290  1384.443183
1964      0.093000      9.173375  ictus  1298.894426  1954.360862
1965      0.144135      9.291943    off   909.702594  1747.600824
1966      0.078979      9.403500  ictus   840.561088  1461.144276

               f3  fileid performer
0     2291.317753      65        LK
1     2625.465878      65        LK
2     2546.395265      65        LK
3     2542.774896      65        LK
4     2671.989524      65        LK
…             …     …       …
1962  3341.040339      69        LO
1963  3073.026293      69        LO
1964  3033.911307      69        LO
1965  2806.347885      69        LO
1966  1987.093716      69        LO
```

```
[1967 rows x 15 columns]
```

[ ]:
```
#for the present paper, we are only interested in the vowel durations:
#set of Estonian vowels to filter the dataframe:
vowels = ["i"," ","y","e"," ","ø","æ","a"," "," ","o"," ","u"," " ]
vowel_df = big_frame[big_frame.segment.isin(vowels)].copy()
print(vowel_df.describe())
vowel_df.head()
```

```
       Unnamed: 0    word_dur    quantity      stress  seg_duration  \
count  615.000000  615.000000  615.000000  615.000000    615.000000
mean   197.894309    0.949184    1.852033    0.528455      0.197959
std    190.819594    0.492294    0.684323    0.499596      0.117456
min      0.000000    0.083718    1.000000    0.000000      0.015260
25%     57.500000    0.555927    1.000000    0.000000      0.131685
50%    132.000000    0.846330    2.000000    1.000000      0.178758
75%    259.000000    1.335653    2.000000    1.000000      0.236604
max    734.000000    2.168336    3.000000    1.000000      1.201817

       seg_midpoint           f1           f2           f3      fileid
count    615.000000   615.000000   615.000000   615.000000  615.000000
mean      57.435001   720.172015  1565.957831  2467.122054   44.450407
std       59.849928   198.120808   396.035823   332.652632   29.328873
min        0.051613   275.499384   519.638388  1396.638475    9.000000
25%       15.556456   567.050872  1262.772049  2300.155478   18.000000
50%       33.763837   703.407720  1523.729941  2496.694808   41.000000
75%       78.970333   867.648661  1838.565263  2663.815935   65.000000
max      219.349440  1219.144093  2693.060094  3331.398284   94.000000
```

[ ]:
```
    Unnamed: 0     word  word_dur  syll  quantity  stress segment  \
1            1    pandi  0.456740   pan         2       1       a
4            4    pandi  0.456740    di         1       0       i
6            6  mind(e)  0.292967  mind         2       1       i
9            9  mind(e)  0.292967   (e)         2       0       e
11          11     paju  0.881775    pa         1       1       a

    seg_duration  seg_midpoint  ictus           f1           f2           f3  \
1       0.177026      6.947528  ictus   603.056801  1303.781807  2625.465878
4       0.129039      7.202174    off  1074.849998  2167.095173  2671.989524
6       0.085210      7.364037    off   731.376613  1651.342030  2557.796933
9       0.067737      7.527525    off   852.719874  1444.954754  2241.764747
11      0.386617      7.887546  ictus   794.798321   850.845027  2525.457521

    fileid performer
1       65        LK
4       65        LK
6       65        LK
```

```
9       65        LK
11      65        LK
```

[ ]:
```
#make sure quantity is read as a categorical variable
vowel_df['quantity'] = vowel_df['quantity'].astype('object')
vowel_df['stress'] = vowel_df['stress'].astype('object')
print("vowel duration and stressed/unstressed: \n" , vowel_df.
 ↪groupby('stress')['seg_duration'].mean())
print("vowel duration and syllable quantity: \n" , vowel_df.
 ↪groupby('quantity')['seg_duration'].mean())
print("vowel duration and ictus/off-ictus \n" , vowel_df.
 ↪groupby('ictus')['seg_duration'].mean())
```

```
vowel duration and stressed/unstressed:
 stress
0    0.228611
1    0.170609
Name: seg_duration, dtype: float64
vowel duration and syllable quantity:
 quantity
1    0.216361
2    0.205660
3    0.140581
Name: seg_duration, dtype: float64
vowel duration and ictus/off-ictus
 ictus
ictus    0.202398
off      0.191118
Name: seg_duration, dtype: float64
```

[ ]:
```
import pandas as pd
import statsmodels .formula.api as smf

#is ictus (song prominence) a good predictor for vowel duration?

ickmodel = smf.ols("seg_duration ~ ictus", data=vowel_df).fit()
ickmodel.summary()
```

[ ]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:           seg_duration   R-squared:                       0.002
Model:                            OLS   Adj. R-squared:                  0.001
Method:             Least Squares   F-statistic:                     1.355
Date:               Thu, 12 May 2022   Prob (F-statistic):              0.245
Time:                       00:26:19   Log-Likelihood:                 445.67
```

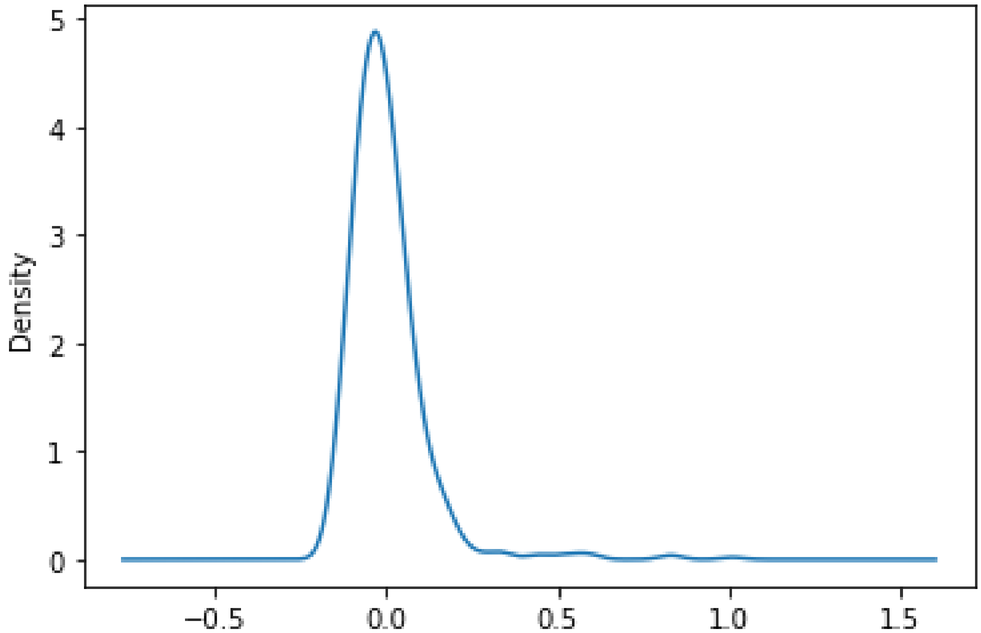
```
No. Observations:                 615   AIC:                            -887.3
Df Residuals:                     613   BIC:                            -878.5
Df Model:                           1
Covariance Type:            nonrobust
================================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
Intercept       0.2024      0.006     33.290      0.000       0.190       0.214
ictus[T.off]   -0.0113      0.010     -1.164      0.245      -0.030       0.008
==============================================================================
Omnibus:                      496.206   Durbin-Watson:                   1.603
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            11531.163
Skew:                           3.442   Prob(JB):                         0.00
Kurtosis:                      23.065   Cond. No.                         2.44
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

well, the intercept coefficient is significant ($p<0.05$), so vowels that are in the ictus position are predictably longer than those in the off-ictus or weaker positions in the song. The R squared is still pretty small, though. Let's see the residuals.

[ ]:
```
ickmodel.fittedvalues
resid_series = pd.Series(ickmodel.resid)
print("min: " ,resid_series.min(), "q1: ", resid_series.quantile(0.25), "median:
 ↪ " ,resid_series.median(),"q3: " , resid_series.quantile(0.75),"max: "␣
 ↪,resid_series.max())
pd.Series(ickmodel.resid).plot.density();
```

```
min:  -0.17585777802014163 q1:  -0.06790367526020552 median:
-0.019300475934164396 q3:  0.03725672678702266 max:  1.0106989261669985
```



ok, we're nearly normal, if not a bit spikier around the mean than preferable. Still, so far it looks like there is a linear relationship between vowel duration and metrical position in the song.

Let's see how quantity is shaking out:

[ ]:
```
qmodel = smf.ols("seg_duration ~ quantity", data=vowel_df).fit()
qmodel.summary()
```

[ ]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:           seg_duration   R-squared:                       0.051
Model:                            OLS   Adj. R-squared:                  0.048
Method:                 Least Squares   F-statistic:                     16.39
Date:                Thu, 12 May 2022   Prob (F-statistic):           1.16e-07
Time:                        00:26:19   Log-Likelihood:                 461.04
No. Observations:                 615   AIC:                            -916.1
Df Residuals:                     612   BIC:                            -902.8
Df Model:                           2
Covariance Type:            nonrobust
================================================================================
=
                    coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
```

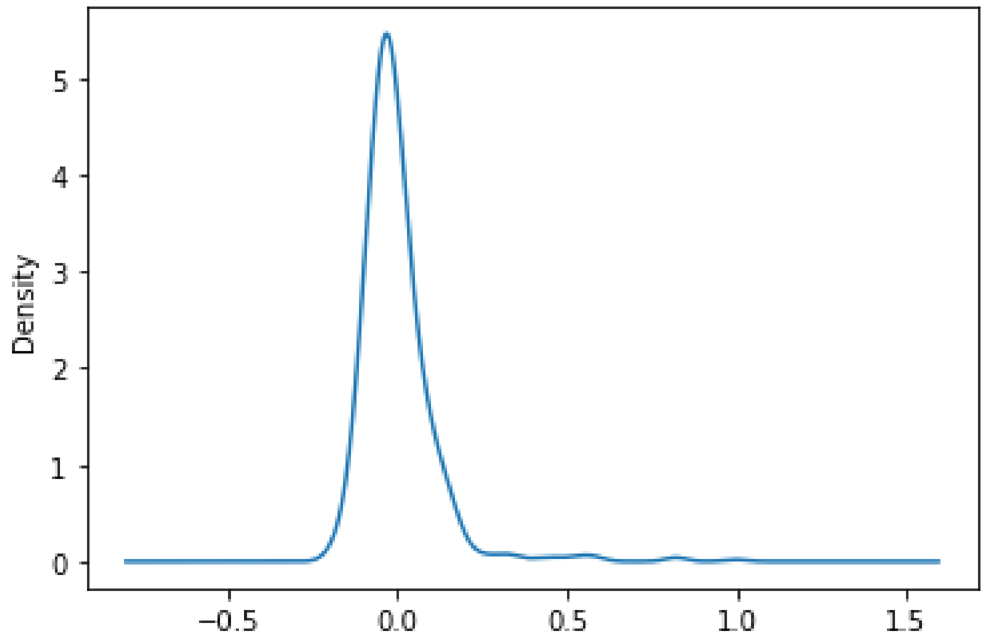
```
-
Intercept          0.2164       0.008       26.428       0.000        0.200
0.232
quantity[T.2]    -0.0107       0.010       -1.026       0.305       -0.031
0.010
quantity[T.3]    -0.0758       0.014       -5.467       0.000       -0.103
-0.049
==============================================================================
Omnibus:                      508.017   Durbin-Watson:                   1.666
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            12614.124
Skew:                           3.541   Prob(JB):                         0.00
Kurtosis:                      24.026   Cond. No.                         4.12
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

Well, we have significant p values for the first (Q1) and third (Q3) coefficients, but not the second (Q2). This makes sense, since the quantity contrast is indicated by the duration ratios of the syllables. Q3, however, never appears in an unstressed position in a word, so only needs to be contrasted with Q2, while Q1 and Q2 both appear in stressed and unstressed positions and need to be differentiated from each other. The adj r-squared here is a little bit better than the ictus model above, and we do have statistical significance for the model overall (p<0.0000)

[ ]:
```
qmodel.fittedvalues
qresid_series = pd.Series(qmodel.resid)
print("min: " ,qresid_series.min(), "q1: ", qresid_series.quantile(0.25),␣
 ↪"median: " ,qresid_series.median(),"q3: " , qresid_series.quantile(0.
 ↪75),"max: " ,qresid_series.max())
pd.Series(qmodel.resid).plot.density();
```

```
min:  -0.20110167790250655 q1:  -0.060080153917226674 median:
-0.021591012703996354 q3:  0.0328693256575211 max:  0.9961566923675012
```



[ ]:
```
stressmodel = smf.ols("seg_duration ~ stress", data=vowel_df).fit()
stressmodel.summary()
```

[ ]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:           seg_duration   R-squared:                       0.061
Model:                            OLS   Adj. R-squared:                  0.059
Method:                 Least Squares   F-statistic:                     39.73
Date:                Thu, 12 May 2022   Prob (F-statistic):           5.58e-10
Time:                        00:26:20   Log-Likelihood:                 464.31
No. Observations:                 615   AIC:                            -924.6
Df Residuals:                     613   BIC:                            -915.8
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.2286      0.007     34.175      0.000       0.215       0.242
stress        -0.0580      0.009     -6.303      0.000      -0.076      -0.040
==============================================================================
Omnibus:                      476.606   Durbin-Watson:                   1.635
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            10197.558
Skew:                           3.270   Prob(JB):                         0.00
```

```
Kurtosis:                      21.847   Cond. No.                         2.69
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

here we have significant effects for both coefficients. It looks like word-level stress predicts a shorter vowel duration, with a negative slope.

```
[ ]: stress_q_model = smf.ols("seg_duration ~ stress + quantity", data=vowel_df).
      ↪fit()
     stress_q_model.summary()
```

```
[ ]: <class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:           seg_duration   R-squared:                       0.077
Model:                            OLS   Adj. R-squared:                  0.073
Method:                 Least Squares   F-statistic:                     17.10
Date:                Thu, 12 May 2022   Prob (F-statistic):           1.12e-10
Time:                        00:43:32   Log-Likelihood:                 469.79
No. Observations:                 615   AIC:                            -931.6
Df Residuals:                     611   BIC:                            -913.9
Df Model:                           3
Covariance Type:            nonrobust
================================================================================
=
                    coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-
Intercept         0.2300      0.009     26.414      0.000       0.213
0.247
stress[T.1]      -0.0432      0.010     -4.200      0.000      -0.063
-0.023
quantity[T.2]    -0.0026      0.010     -0.251      0.802      -0.023
0.018
quantity[T.3]    -0.0462      0.015     -3.007      0.003      -0.076
-0.016
==============================================================================
Omnibus:                      488.788   Durbin-Watson:                   1.653
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            11163.529
Skew:                           3.370   Prob(JB):                         0.00
Kurtosis:                      22.754   Cond. No.                         4.99
==============================================================================
```

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

so far, this is the best model we have as far as adj r-squared goes. It is explaining the most variation so far, and it is unlikely with the low p value that the model is complete trash. We didn't lose significance for any of our coefficients from single factor models.

```
[ ]: stress_q_model.fittedvalues
stressresid_series = pd.Series(stress_q_model.resid)
print("min: " ,stressresid_series.min(), "q1: ", stressresid_series.quantile(0.
 ↪25), "median: " ,stressresid_series.median(),"q3: " , stressresid_series.
 ↪quantile(0.75),"max: " ,stressresid_series.max())
pd.Series(stress_q_model.resid).plot.density();
```

```
min:  -0.19704059472321775 q1:  -0.06306568545920294 median:
-0.020152767123178622 q3:  0.03726678408654223 max:  0.974420464328187
```