# Utilizing Python to quantify vocal babbling signals in an extremely altricial bird

Rory Eggleston

7 May 2019

## 1 Abstract

Male songbirds have long been the main model for understanding neuroendocrinological foundations of vocal imitation. Elevated levels of the avian stress hormone, corticosterone (CORT), can have profound effects on early learning. Parrots are among the most plastic non-human vocal learners but it is unclear how stress affects early learning. CORT titres were experimentally raised in broods of free-ranging Green-rumped Parrotlets (*Forpus passerinus*). Nestlings received daily CORT-oil supplements, pure oil, or nothing. Simultaneous audio-video recordings were made in nest cavities to quantify vocal babbling. Based on a principal components analysis (PCA), both oil and CORT supplements appear to have some effect on babbling, both in acoustical structure and in overall output.

## 2 Introduction

Vocal learning is the ability to modulate acoustic signals based on environmental templates (4). Among the few taxa known to be vocal learners are oscine songbirds (6) and parrots (9). Vocal babbling is a stage in vocal development that occurs at the intersection of innate and learned behaviors, when the juvenile begins

testing out the various sounds and vocal signals that it has been previously exposed to (5). In songbirds and parrots, nestlings transition from innate vocalizations, such as begging calls, to babbling that consists of both innate and learned sounds, to songs and other kinds of learned calls (1; 2).

A number of studies have attempted to explain why bird song and complex vocalizations are beneficial (11; 8). A prominent hypothesis for songbirds suggests that nestlings that get more food or avoid sickness will develop more robust song repertoires and stand a better chance of reproducing than developmentally stressed siblings (7). However, other studies that have examined endocrine responses and the effects of stress on development have suggested that in some species the stress axis of the endocrine system does not fully develop until right before or even after vocal development begins (10). The glucocorticoid CORT is primarily involved in regulating avian stress response. In studies conducted in songbirds, it has been found that extended periods of elevated CORT levels can have varied effects on learning and development, including depressing nestling growth and feather development, and enhancing spatial memory (12; 3). Cort signaling is also thought to be involved in song learning in songbirds (13). Determining how many signal types are produced by an individual can act as a proxy to determine whether cort levels have an effect, positive or negative, on vocal learning/babbling.

# 3 Methods

I compiled the babbling signal data first in Excel, and then uploaded the resulting spreadsheet to Spyder as a pandas dataframe. Originally, the data set contained approximately 80 variables describing various aspects of individual signals, so I created another dataframe that contained only the ten variables relevant to this project (nest, treatment, nestling, delta time (s), bandwidth interquartile range (Hz), average entropy (bits), center frequency (Hz), 5% relative frequency, 95% relative frequency, and average slope of peak frequency contour (Hz/ms), Fig. 1). From this, I also created three separate dataframes for each treatment (control, CORT, and oil), in order to facilitate plotting and comparing each treatment. Using the dataframe with the ten relevant signal variables, Based on the distributions of the measurements, I normalized the data

using scikitlearn's StandardScaler. Although some of the distributions suggested that a log transformation might be functional, this did not yield good results, so I performed all further analysis exclusively on the normalized data.

Using the dataframe with the ten relevant signal variables, I used matplotlib to plot the distribution of measurements for each variable by each treatment. Based on the distributions of the measurements, I normalized the data using scikitlearn's StandardScaler. Although some of the distributions suggested that a log transformation might be functional, this did not yield good results, so I performed all further analysis exclusively on the normalized data.

I used both the scikitlearn and the statsmodels PCA (principal components analysis) packages to compare PCA results. The scikitlearn package proved in adequate, so I only used the statsmodels package for my final analysis. I specified the use of seven principal components in the statsmodels package. Due to the large quantity of signal data, I used a 2D histogram in lieu of a normal scatterplot, in order to determine where data was concentrated in the cloud generated from the PCA.

I also uploaded another smaller data set containing information about the nestlings and babbling bouts (age of babbling onset (dph), and number of babbling bouts and signals). I created three small dataframes from this, one for each variable. In order to determine whether there were any obvious trends for each treatment, I used matplotlib to create boxplots for each of the three variables.

# 4 Results

The eigenvectors associated with each variable used for the statsmodels PCA suggest that all seven variables contribute a non-trivial amount to the variability described by each of the first three principal components. Likewise, the first three principal components contributed 46.01%, 15.76%, and 14.86%, respectively, to the overall variability of the data; thus, only these first three principal components were used for this analysis. Rendering three dimensional figures in Python proved difficult, so I instead chose to compare each of the three principal components to each other for the 2D histograms (Fig. 2). These comparisons demonstrate

that there is significant overlap between the three treatments, and suggest that there are definite similarities between the oil and CORT treatments.

Based on the general nestling data, CORT-treated nestlings start babbling later than their control or oil-treated siblings (Fig. 3). Likewise, CORT-treated nestlings tend to have fewer overall babbling bouts over time, and have a lot of variation in the number of signals that they produce overall. However, it is important to note that these particular results do not have any really meaningful statistical power, and are simply a method of observing potential trends in the treatments.

# 5 Discussion

The PCA results suggest that the oil treatment has some effect on babbling structure, and on the seven analyzed variables. They also suggest that the CORT somewhat amplifies this effect, leading to a larger spread in the CORT data. Coupled with the plots in Fig. 3, this in turn indicates that moderately elevated CORT levels may have some effect on babbling overall, including both output and structure. The use of 2D histograms in the PCA plots (Fig. 2) also demonstrated that the vast preponderance of the data was concentrated in the same area of the principal component space across all three treatments. Although the nestling data (Fig. 3) suggests that CORT-treated nestlings start babbling later than their siblings, and demonstrate fewer babbling bouts with a large range in the number of individual signals produced, this result is likely suspect due to the small sample size (three birds per treatment).

As mentioned in the methods, I compared the results of two different PCA packages before settling on statsmodels. Initially, I had hoped to use the scikitlearn package to do a k-means clustering analysis, and realized that this would be easiest to do using a PCA. However, the documentation for scikitlearn's PCA package was poorly explained, and was not conducive to providing interpretable results. Specifically, the documentation does not make it clear how to access the individual eigenvectors and eigenvalues for the variables and each principal component in a given analysis. As these are often necessary to fully understand the results of the PCA, this was clearly not tenable, especially for someone new to Python.

In contrast, the statsmodels PCA package is both relatively easy to use and specifically delivers both eigenvalues and eigenvectors as part of its general output, which can then be organized as desired. This package also allows for more straightforward creation of figures utilizing the results, and was in general easier for interpretation.

# 6 Bibliography

# References


[1] Baptista, L. F., and Petrinovich, L. Song development in the white-crowned sparrow: social factors and sex differences. *Animal Behaviour 34*, 5 (Oct. 1986), 1359–1371.

[2] Berg Karl S., Delgado Soraya, Cortopassi Kathryn A., Beissinger Steven R., and Bradbury Jack W. Vertical transmission of learned signatures in a wild parrot. *Proceedings of the Royal Society B: Biological Sciences 279*, 1728 (Feb. 2012), 585–591.

[3] Crino, O. L., Driscoll, S. C., Ton, R., and Breuner, C. W. Corticosterone exposure during development improves performance on a novel foraging task in zebra finches. *Animal Behaviour 91* (May 2014), 27–32.

[4] Jarvis, E. D. Learned Birdsong and the Neurobiology of Human Language. *Annals of the New York Academy of Sciences 1016*, 1 (2004), 749–777.

[5] Lipkind, D., Marcus, G. F., Bemis, D. K., Sasahara, K., Jacoby, N., Takahasi, M., Suzuki, K., Feher, O., Ravbar, P., Okanoya, K., and Tchernichovski, O. Stepwise acquisition of vocal combinatorial capacity in songbirds and human infants. *Nature 498*, 7452 (June 2013), 104–108.

[6] Marler, P. A comparative approach to vocal learning: Song development in white-crowned sparrows. *Journal of Comparative and Physiological Psychology 71*, 2, Pt.2 (1970), 1–25.

[7] Nowicki, S., Peters, S., and Podos, J. Song Learning, Early Nutrition and Sexual Selection in Songbirds. *Integrative and Comparative Biology 38*, 1 (Feb. 1998), 179–190.

[8] Nowicki, S., and Searcy, W. A. The evolution of vocal learning. *Current Opinion in Neurobiology 28* (Oct. 2014), 48–53.

[9] Pepperberg, I. M. Vocal Learning in Grey Parrots (Psittacus Erithacus): Effects of Social Interaction, Reference, and Context. *The Auk: Ornithological Advances 111*, 2 (Apr. 1994), 300–313.

[10] Schwabl, H. Developmental Changes and Among-Sibling Variation of Corticosterone Levels in an Altricial Avian Species. *General and Comparative Endocrinology 116*, 3 (Dec. 1999), 403–408.

[11] Spencer, K. A., Buchanan, K. L., Goldsmith, A. R., and Catchpole, C. K. Song as an honest signal of developmental stress in the zebra finch (Taeniopygia guttata). *Hormones and Behavior 44*, 2 (Aug. 2003), 132–139.

[12] Spencer, K. A., and Verhulst, S. Delayed behavioral effects of postnatal exposure to corticosterone in the zebra finch (Taeniopygia guttata). *Hormones and Behavior 51*, 2 (Feb. 2007), 273–280.

[13] Suzuki, K., Matsunaga, E., Kobayashi, T., and Okanoya, K. Expression patterns of mineralocorticoid and glucocorticoid receptors in Bengalese finch (Lonchura striata var. domestica) brain suggest a relationship between stress hormones and song-system development. *Neuroscience 194* (Oct. 2011), 72–83.

4

# 7 Figures

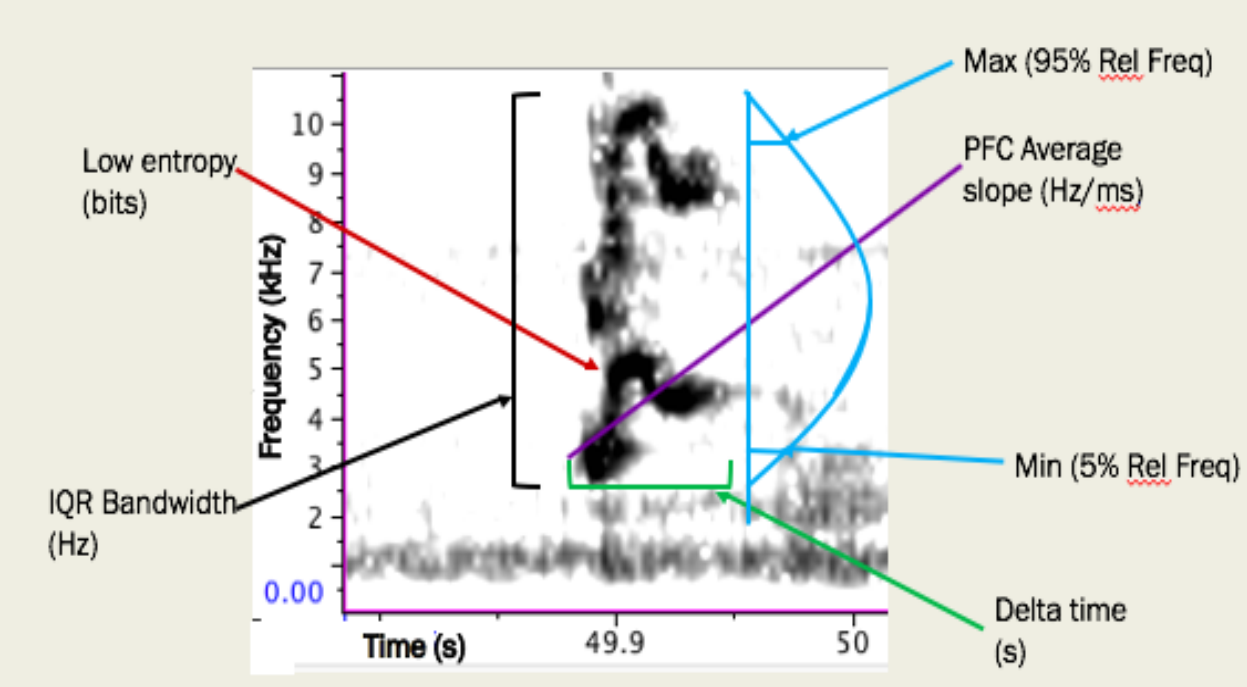


Figure 1: Visualization of the seven variables included in the principal components analysis, minus center frequency, which is simply the mean frequency of the signal.

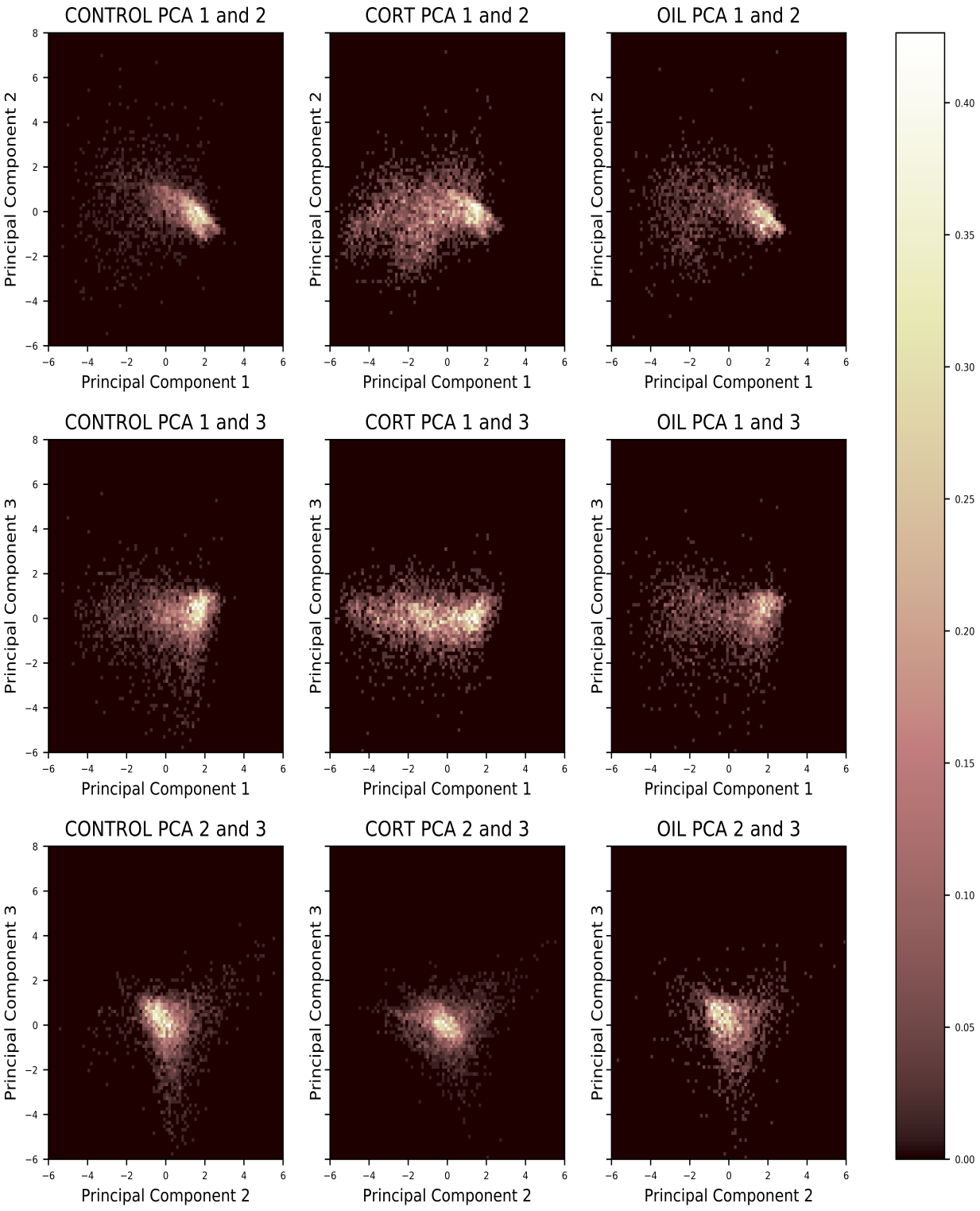


Figure 2: 2D histogram visualization of a principal components analysis of babbling signal data for three nests, using the statsmodels PCA package. Principal components 1, 2, and 3 account for 46.01%, 15.76%, and 14.86% of the observed variability, and are each compared to each other here.

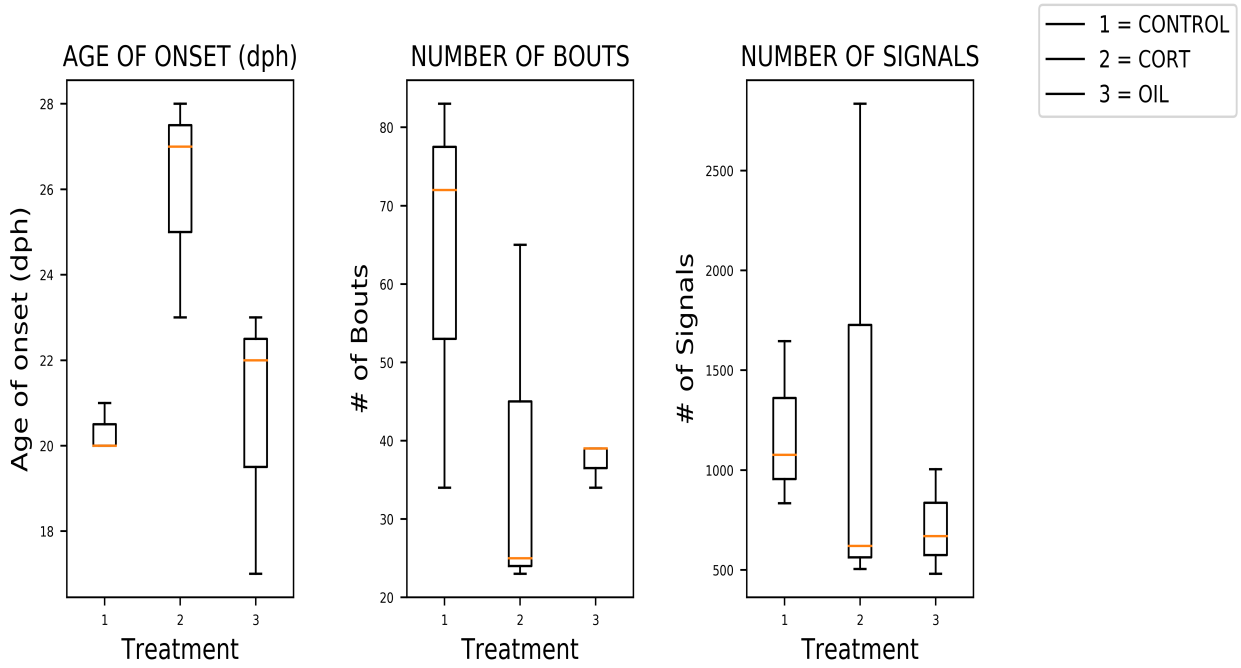


Figure 3: Boxplots of general nestling data by treatment for three variables: age of babbling onset, number of babbling bouts overall, and number of individual babbling signals.

# 8 Appendix

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
import pandas as pd
import xarray as xr
import netCDF4 as nc
from sklearn.cluster import KMeans
#%%
file = '/Volumes/MY PASSPORT/CORT_BABBLE_DATA_040819.xlsx'
babble = pd.read_excel(file)
#%%
babble.shape
#%%
#TRYING TO DO K-MEANS BEFORE PCA, NOT GONNA WORK FOR MY DATA
kmeans_vars = babble[[("N"), ("Treatment"), ("Nest"), ("Delta Time (s)"), ("IQR BW (Hz)"), ("Avg Entropy (bits)"), ("Center
Freq (Hz)"), ("Freq 5% Rel."), ("Freq 95% Rel."), ("PFC Avg Slope (Hz/ms)")]]
#%%
#NEXT TIME WILL BE WORKING ON DOING PCA FIRST
kmeans_vars_old = kmeans_vars
#%%
#ONLINE EXAMPLE
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
df = pd.read_csv(url, names=['sepal length','sepal width','petal length','petal width','target'])
#%%
from sklearn.preprocessing import StandardScaler
features = ['sepal length', 'sepal width', 'petal length', 'petal width']
# Separating out the features
x = df.loc[:, features].values
# Separating out the target
y = df.loc[:,['target']].values
# Standardizing the features
x = StandardScaler().fit_transform(x)
#%%
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(x)
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['principal component 1', 'principal component 2'])
#%%
finalDf = pd.concat([principalDf, df[['target']]], axis = 1)
#%%
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('2 component PCA', fontsize = 20)
targets = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
colors = ['r', 'g', 'b']
for target, color in zip(targets,colors):
    indicesToKeep = finalDf['target'] == target
    ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1']
               , finalDf.loc[indicesToKeep, 'principal component 2']
               , c = color
               , s = 50)
ax.legend(targets)
ax.grid()
#%%
pca.explained_variance_ratio_
#%%
#TRYING PCA WITH MY DATA
babble_numeric = ["Delta Time (s)", "IQR BW (Hz)", "Avg Entropy (bits)", "Center Freq (Hz)", "Freq 5% Rel.", "Freq 95%
Rel.", "PFC Avg Slope (Hz/ms)"]
```

```
babble_cat = ["N", "Treatment", "Nest"]
#%%
x = babble.loc[:, babble_numeric].values
y = babble.loc[:, babble_cat].values
#%%
x = StandardScaler().fit_transform(x)
#%%
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(x)
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['principal component 1', 'principal component 2'])
#%%
BabbleDf = pd.concat([principalDf, babble[["N", "Treatment", "Nest"]]], axis = 1)
#%%
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('7 component PCA', fontsize = 20)
treatments = ['CONTROL', 'CORT', 'OIL']
colors = ['r', 'g', 'b']
for treatment, color in zip(treatments,colors):
    indicesToKeep = BabbleDf['Treatment'] == treatment
    ax.scatter(BabbleDf.loc[indicesToKeep, 'principal component 1']
               , BabbleDf.loc[indicesToKeep, 'principal component 2']
               , c = color
               , s = 50)
ax.legend(treatments)
#%%
pca.explained_variance_ratio_
#%%
#MAKE 2D HISTOGRAMS OF NORMALIZED DATA FOR EACH TREATMENT
fig = plt.figure(figsize = (8,8))
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('Principal Component 1', fontsize = 15)
ax.set_ylabel('Principal Component 2', fontsize = 15)
ax.set_title('7 component PCA', fontsize = 20)
treatments = ['CONTROL', 'CORT', 'OIL']
colors = ['r', 'g', 'b']
for treatment, color in zip(treatments,colors):
    indicesToKeep = BabbleDf['Treatment'] == treatment
    ax.scatter(BabbleDf.loc[indicesToKeep, 'principal component 1']
               , BabbleDf.loc[indicesToKeep, 'principal component 2']
               , c = color
               , s = 50)
ax.legend(treatments)
#%%
BabbleDf_cluster = principalDf
#%%
fig, ax= plt.subplots(1, 3, figsize = (12,4), sharey = True)


treatments = ['CONTROL', 'CORT', 'OIL']

#make arange for bins using both PCAs

ax[0].set_xlabel('Principal Component 1', fontsize = 15)
ax[0].set_ylabel('Principal Component 2', fontsize = 15)
ax[0].set_title('CONTROL')
treatment0 = treatments[0]
indicesToKeep0 = BabbleDf['Treatment'] == treatment0
ax[0].hist2d(BabbleDf.loc[indicesToKeep0, 'principal component 1']
          , BabbleDf.loc[indicesToKeep0, 'principal component 2'], bins = 90, range = [[-4,6], [-6,8]], cmap = "Greys")
```

```
ax[1].set_xlabel('Principal Component 1', fontsize = 15)
ax[1].set_ylabel('Principal Component 2', fontsize = 15)
ax[1].set_title('CORT')
treatment1 = treatments[1]
indicesToKeep1 = BabbleDf['Treatment'] == treatment1
ax[1].hist2d(BabbleDf.loc[indicesToKeep1, 'principal component 1']
        , BabbleDf.loc[indicesToKeep1, 'principal component 2'], bins = 90, range = [[-4,6], [-6,8]], cmap = "Greys")

ax[2].set_xlabel('Principal Component 1', fontsize = 15)
ax[2].set_ylabel('Principal Component 2', fontsize = 15)
ax[2].set_title('OIL')
treatment2 = treatments[2]
indicesToKeep2 = BabbleDf['Treatment'] == treatment2
im = ax[2].hist2d(BabbleDf.loc[indicesToKeep2, 'principal component 1']
        , BabbleDf.loc[indicesToKeep2, 'principal component 2'], bins = 90, range = [[-4,6], [-6,8]], cmap = "Greys")

fig.colorbar(im[3], ax = ax)
#%%
index = kmeans_vars.index
columns = kmeans_vars.columns
values = kmeans_vars.values
#%%
control = kmeans_vars[kmeans_vars["Treatment"] == "CONTROL"]
cort = kmeans_vars[kmeans_vars["Treatment"] == "CORT"]
oil = kmeans_vars[kmeans_vars["Treatment"] == "OIL"]
#%%
fig, ax = plt.subplots(nrows = 3, ncols = 7, figsize = (12,4), sharey = True)
plt.rc('xtick',labelsize=6)
plt.rc('ytick',labelsize=6)
plt.suptitle("Distribution of Babbling Variables")

ax[0,0].set_ylabel("CONTROL", rotation=90, size='small')
ax[0,0].hist(control["IQR BW (Hz)"])
ax[0,0].set_title("IQR BW (Hz)", fontsize = 9)
ax[0,1].hist(control["Delta Time (s)"])
ax[0,1].set_title("Delta Time (s)", fontsize = 9)
ax[0,2].hist(control["Avg Entropy (bits)"])
ax[0,2].set_title("Avg Entropy (bits)", fontsize = 9)
ax[0,3].hist(control["Center Freq (Hz)"])
ax[0,3].set_title("Center Freq (Hz)", fontsize = 9)
ax[0,4].hist(control["Freq 5% Rel."])
ax[0,4].set_title("Freq 5% Rel.", fontsize = 9)
ax[0,5].hist(control["Freq 95% Rel."])
ax[0,5].set_title("Freq 95% Rel.", fontsize = 9)
ax[0,6].hist(control["PFC Avg Slope (Hz/ms)"])
ax[0,6].set_title("PFC Avg Slope (Hz/ms)", fontsize = 9)

ax[1,0].set_ylabel("CORT", rotation=90, size='small')
ax[1,0].hist(cort["IQR BW (Hz)"])
ax[1,1].hist(cort["Delta Time (s)"])
ax[1,2].hist(cort["Avg Entropy (bits)"])
ax[1,3].hist(cort["Center Freq (Hz)"])
ax[1,4].hist(cort["Freq 5% Rel."])
ax[1,5].hist(cort["Freq 95% Rel."])
ax[1,6].hist(cort["PFC Avg Slope (Hz/ms)"])

ax[2,0].set_ylabel("OIL", rotation=90, size='small')
ax[2,0].hist(oil["IQR BW (Hz)"])
ax[2,1].hist(oil["Delta Time (s)"])
ax[2,2].hist(oil["Avg Entropy (bits)"])
ax[2,3].hist(oil["Center Freq (Hz)"])
ax[2,4].hist(oil["Freq 5% Rel."])
ax[2,5].hist(oil["Freq 95% Rel."])
ax[2,6].hist(oil["PFC Avg Slope (Hz/ms)"])
#%%
```

```
#fit determines mean and SD
#transform applies them
#fit_transform will give the same as just fit if you've already used StandardScaler on the data
pca = PCA(n_components=6)
principalComponents6 = pca.fit_transform(x)
principalDf6 = pd.DataFrame(data = principalComponents6
             , columns = ['principal component 1', 'principal component 2', 'principal component 3', 'principal component 4', 'principal
component 5', 'principal component 6'])
#%%
pca.explained_variance_ratio_
#%%
pca.explained_variance_
#%%
BabbleDf6 = pd.concat([babble[["N", "Treatment", "Nest"]], principalDf6], axis = 1)
#%%
index = BabbleDf.index
columns = BabbleDf.columns
values = BabbleDf.values
#%%
controlPCA = BabbleDf[BabbleDf["Treatment"] == "CONTROL"]
cortPCA = BabbleDf[BabbleDf["Treatment"] == "CORT"]
oilPCA = BabbleDf[BabbleDf["Treatment"] == "OIL"]
#%%
fig, ax= plt.subplots(1, 3, figsize = (12,4), sharey = True)


treatments = ['CONTROL', 'CORT', 'OIL']

#make arange for bins using both PCAs

ax[0].set_xlabel('Principal Component 1', fontsize = 15)
ax[0].set_ylabel('Principal Component 2', fontsize = 15)
ax[0].set_title('CONTROL')
treatment0 = treatments[0]
indicesToKeep0 = controlPCA['Treatment'] == treatment0
ax[0].hist2d(controlPCA.loc[indicesToKeep0, 'principal component 1'], controlPCA.loc[indicesToKeep0, 'principal component
2'], bins = 90, normed = True, cmap = "Greys")


ax[1].set_xlabel('Principal Component 1', fontsize = 15)
ax[1].set_ylabel('Principal Component 2', fontsize = 15)
ax[1].set_title('CORT')
treatment1 = treatments[1]
indicesToKeep1 = cortPCA['Treatment'] == treatment1
ax[1].hist2d(cortPCA.loc[indicesToKeep1, 'principal component 1'], cortPCA.loc[indicesToKeep1, 'principal component 2'], bins
= 90, normed = True, cmap = "Greys")

ax[2].set_xlabel('Principal Component 1', fontsize = 15)
ax[2].set_ylabel('Principal Component 2', fontsize = 15)
ax[2].set_title('OIL')
treatment2 = treatments[2]
indicesToKeep2 = oilPCA['Treatment'] == treatment2
im = ax[2].hist2d(oilPCA.loc[indicesToKeep2, 'principal component 1'], oilPCA.loc[indicesToKeep2, 'principal component 2'],
bins = 90, normed = True, cmap = "Greys")

fig.colorbar(im[3], ax = ax)
#%%
#FIGURE OUT PCA USING STATSMODELS
from statsmodels.sandbox.tools import pca
#%%
outputbabble = pca(x, keepdim = 0, demean = True)
#%%
eigenvectors = outputbabble[3]
eigenvalues = outputbabble[2]
total_eigenvectors = outputbabble[1]
```

```
#%%
columns = ["Principal Component 1", "Principal Component 2", "Principal Component 3", "Principal Component 4", "Principal Component 5", "Principal Component 6", "Principal Component 7"]
rows = ["Delta Time (s)", "IQR BW (Hz)", "Avg Entropy (bits)", "Center Freq (Hz)", "Freq 5% Rel.", "Freq 95% Rel.", "PFC Avg Slope (Hz/ms)"]
PCs_Variables_Babble = pd.DataFrame(data=eigenvectors, index = rows, columns = columns)
PCA2_Babble = pd.DataFrame(data = total_eigenvectors, columns = columns)
#%%
BabbleDf2 = pd.concat([babble[["N", "Treatment", "Nest"]], PCA2_Babble], axis = 1)
#%%
controlPCA2 = BabbleDf2[BabbleDf2["Treatment"] == "CONTROL"]
cortPCA2 = BabbleDf2[BabbleDf2["Treatment"] == "CORT"]
oilPCA2 = BabbleDf2[BabbleDf2["Treatment"] == "OIL"]
#%%
fig, ax= plt.subplots(3, 3, figsize = (12,12), sharey = True)
fig.subplots_adjust(hspace = 0.3)

treatments = ['CONTROL', 'CORT', 'OIL']

#make arange for bins using both PCAs

ax[0,0].set_xlabel('Principal Component 1', fontsize = 10)
ax[0,0].set_ylabel('Principal Component 2', fontsize = 10)
ax[0,0].set_title('CONTROL PCA 1 and 2')
treatment0 = treatments[0]
indicesToKeep0 = controlPCA2['Treatment'] == treatment0
ax[0,0].hist2d(controlPCA2.loc[indicesToKeep0, 'Principal Component 1'], controlPCA2.loc[indicesToKeep0, 'Principal Component 2'], bins = 90, range = [[-6,6], [-6,8]], normed = True, cmap = "pink")


ax[0,1].set_xlabel('Principal Component 1', fontsize = 10)
ax[0,1].set_ylabel('Principal Component 2', fontsize = 10)
ax[0,1].set_title('CORT PCA 1 and 2')
treatment1 = treatments[1]
indicesToKeep1 = cortPCA2['Treatment'] == treatment1
ax[0,1].hist2d(cortPCA2.loc[indicesToKeep1, 'Principal Component 1'], cortPCA2.loc[indicesToKeep1, 'Principal Component 2'], bins = 90, range = [[-6,6], [-6,8]], normed = True, cmap = "pink")

ax[0,2].set_xlabel('Principal Component 1', fontsize = 10)
ax[0,2].set_ylabel('Principal Component 2', fontsize = 10)
ax[0,2].set_title('OIL PCA 1 and 2')
treatment2 = treatments[2]
indicesToKeep2 = oilPCA2['Treatment'] == treatment2
im = ax[0,2].hist2d(oilPCA2.loc[indicesToKeep2, 'Principal Component 1'], oilPCA2.loc[indicesToKeep2, 'Principal Component 2'], bins = 90, range = [[-6,6], [-6,8]], normed = True, cmap = "pink")

ax[1,0].set_xlabel('Principal Component 1', fontsize = 10)
ax[1,0].set_ylabel('Principal Component 3', fontsize = 10)
ax[1,0].set_title('CONTROL PCA 1 and 3')
treatment0 = treatments[0]
indicesToKeep0 = controlPCA2['Treatment'] == treatment0
ax[1,0].hist2d(controlPCA2.loc[indicesToKeep0, 'Principal Component 1'], controlPCA2.loc[indicesToKeep0, 'Principal Component 3'], bins = 90, range = [[-6,6], [-6,8]], normed = True, cmap = "pink")


ax[1,1].set_xlabel('Principal Component 1', fontsize = 10)
ax[1,1].set_ylabel('Principal Component 3', fontsize = 10)
ax[1,1].set_title('CORT PCA 1 and 3')
treatment1 = treatments[1]
indicesToKeep1 = cortPCA2['Treatment'] == treatment1
ax[1,1].hist2d(cortPCA2.loc[indicesToKeep1, 'Principal Component 1'], cortPCA2.loc[indicesToKeep1, 'Principal Component 3'], bins = 90, range = [[-6,6], [-6,8]], normed = True, cmap = "pink")

ax[1,2].set_xlabel('Principal Component 1', fontsize = 10)
ax[1,2].set_ylabel('Principal Component 3', fontsize = 10)
```

```
ax[1,2].set_title('OIL PCA 1 and 3')
treatment2 = treatments[2]
indicesToKeep2 = oilPCA2['Treatment'] == treatment2
ax[1,2].hist2d(oilPCA2.loc[indicesToKeep2, 'Principal Component 1'], oilPCA2.loc[indicesToKeep2, 'Principal Component 3'],
bins = 90, range = [[-6,6], [-6,8]], normed = True, cmap = "pink")

ax[2,0].set_xlabel('Principal Component 2', fontsize = 10)
ax[2,0].set_ylabel('Principal Component 3', fontsize = 10)
ax[2,0].set_title('CONTROL PCA 2 and 3')
treatment0 = treatments[0]
indicesToKeep0 = controlPCA2['Treatment'] == treatment0
ax[2,0].hist2d(controlPCA2.loc[indicesToKeep0, 'Principal Component 2'], controlPCA2.loc[indicesToKeep0, 'Principal
Component 3'], bins = 90, range = [[-6,6], [-6,8]], normed = True, cmap = "pink")


ax[2,1].set_xlabel('Principal Component 2', fontsize = 10)
ax[2,1].set_ylabel('Principal Component 3', fontsize = 10)
ax[2,1].set_title('CORT PCA 2 and 3')
treatment1 = treatments[1]
indicesToKeep1 = cortPCA2['Treatment'] == treatment1
ax[2,1].hist2d(cortPCA2.loc[indicesToKeep1, 'Principal Component 2'], cortPCA2.loc[indicesToKeep1, 'Principal Component
3'], bins = 90, range = [[-6,6], [-6,8]], normed = True, cmap = "pink")

ax[2,2].set_xlabel('Principal Component 2', fontsize = 10)
ax[2,2].set_ylabel('Principal Component 3', fontsize = 10)
ax[2,2].set_title('OIL PCA 2 and 3')
treatment2 = treatments[2]
indicesToKeep2 = oilPCA2['Treatment'] == treatment2
ax[2,2].hist2d(oilPCA2.loc[indicesToKeep2, 'Principal Component 2'], oilPCA2.loc[indicesToKeep2, 'Principal Component 3'],
bins = 90, range = [[-6,6], [-6,8]], normed = True, cmap = "pink")


fig.colorbar(im[3], ax = ax)

fig.savefig('/Users/roryeggleston/Documents/Babble3PCA.png', bbox_inches='tight', dpi=1030)
#%%
file = '/Users/roryeggleston/Documents/BabbleOutput3Nests.xlsx'
Output3 = pd.read_excel(file)
#%%
Output3.shape
#%%
OutputIndex = Output3.index
OutputColumns = Output3.columns
OutputValues = Output3.values
#%%
ControlOutput = Output3[Output3["TREATMENT"] == "CONTROL"]
ControlAge = ControlOutput["AGE OF ONSET (dph)"]
ControlBout = ControlOutput["# OF BOUTS"]
ControlSignal = ControlOutput["# OF SIGNALS"]
CortOutput = Output3[Output3["TREATMENT"] == "CORT"]
CortAge = CortOutput["AGE OF ONSET (dph)"]
CortBout = CortOutput["# OF BOUTS"]
CortSignal = CortOutput["# OF SIGNALS"]
OilOutput = Output3[Output3["TREATMENT"] == "OIL"]
OilAge = OilOutput["AGE OF ONSET (dph)"]
OilBout = OilOutput["# OF BOUTS"]
OilSignal = OilOutput["# OF SIGNALS"]
#%%
AgeOfOnset = [ControlAge, CortAge, OilAge]
NumberBouts = [ControlBout, CortBout, OilBout]
NumberSignals = [ControlSignal, CortSignal, OilSignal]
#%%
fig, ax = plt.subplots(1, 3, figsize = (10, 4))
fig.subplots_adjust(wspace = 0.5)
```

```
ax[0].boxplot(AgeOfOnset)
ax[0].set_xlabel('Treatment', fontsize = 12)
ax[0].set_ylabel('Age of onset (dph)', fontsize = 12)
ax[0].set_title('AGE OF ONSET (dph)')

ax[1].boxplot(NumberBouts)
ax[1].set_xlabel('Treatment', fontsize = 12)
ax[1].set_ylabel('# of Bouts', fontsize = 12)
ax[1].set_title('NUMBER OF BOUTS')

ax[2].boxplot(NumberSignals)
ax[2].set_xlabel('Treatment', fontsize = 12)
ax[2].set_ylabel('# of Signals', fontsize = 12)
ax[2].set_title('NUMBER OF SIGNALS')

fig.legend(["1 = CONTROL", "2 = CORT", "3 = OIL"], loc = "best")
fig.subplots_adjust(right = 0.80)
fig.savefig('/Users/roryeggleston/Documents/NestlingData.png', bbox_inches='tight', dpi=750)
#%%
```