鳥の学校－第4回テーマ別講習会
鳥の鳴き声を分析しよう
実習編

講師:百瀬浩 （独）農研機構 中央農業総合研究センター 鳥獣害研究サブチーム
2010 年9月 21 日(火)

Contents

準備．ソフトのインストール

・Audacity （音声編集用ソフト）
http://audacity.sourceforge.net/

・Raven Lite （音声分析用ソフト）
http://www.birds.cornell.edu/brp/raven/RavenVersions.html#RavenLite

・R（統計ソフトだが音声分析に使用）
http://www.r-project.org/
Rの音声分析用パッケージほか
sound, tuneR, seewave, audio, rgl, rpanel, tcltk

・表計算ソフト OpenOffice（エクセルが使えない場合）
http://www.openoffice.org/

Rパッケージのインストール(Mac)

・パッケージの確認方法（Mac）
１）Rを起動。
２）「パッケージとデータ」から「パッケージマネージャ」を開く。
３）リスト内に下記のパッケージがあるかどうかを確認する。
sound, TuneR, seewave, audio, rgl, rpanel, tcltk

・パッケージのインストール方法（Mac）
１）Rを起動。
２）「パッケージとデータ」から「パッケージインストーラ」を開く
３）「一覧を取得」をクリック
４）出てきたリストの中から、上記の４つのパッケージのうちインストールされていなかったものを選択（コマンドキーを押しながら選択すると、複数 選択可）
５）「選択をインストール」
６）パッケージマネージャでインストールされているかどうかを確認する。

Rパッケージのインストール(Win)

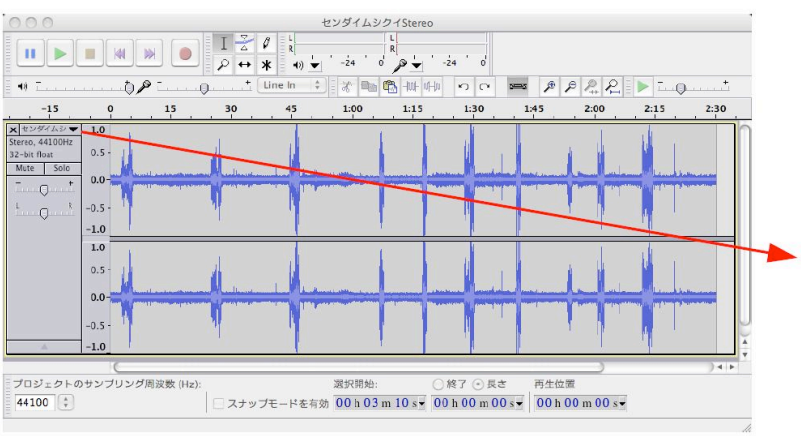
・パッケージの確認方法（Win）
１）Rを起動。
２）「パッケージ」から「パッケージの読み込み」を開く
３）リスト内に下記のパッケージがあるかどうかを確認する。
sound, TuneR, seewave, audio, rgl, rpanel, tcltk

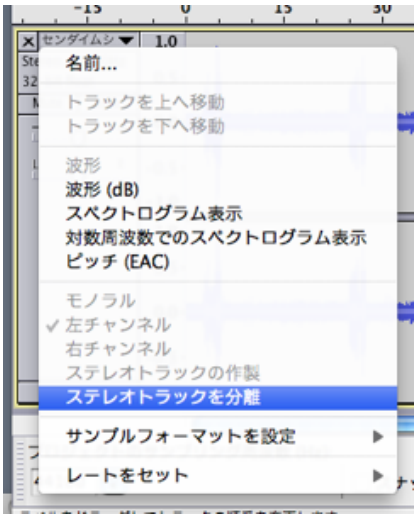
・パッケージのインストール方法（Win）
１）Rを起動。
２）「パッケージ」から「パッケージのインストール」を開く
３）”CRAN mirror”の選択リストが表示されるので適当なミラーサイトを選ぶ
４）パッケージのリストが表示されるので、中から、上記の４つのパッケージのうちインストールされていなかったものを選択（CTRLキーを押しながら選択すると、複数 選択可）
５）「OK」をクリックする
６）「パッケージの読み込み」でインストールされているか（リスト内にあるか）を確認する。

1．音声ファイルの編集・処理（Audacity を使用）

1－1　ステレオ→モノラルの変換

・Audacity を起動
・ファイル　／　開く　から
　“.../SampleFiles/センダイムシクイ Stereo.wav” を開く
・ファイル名横の▼をクリックして「ステレオトラックを分離」を選ぶ

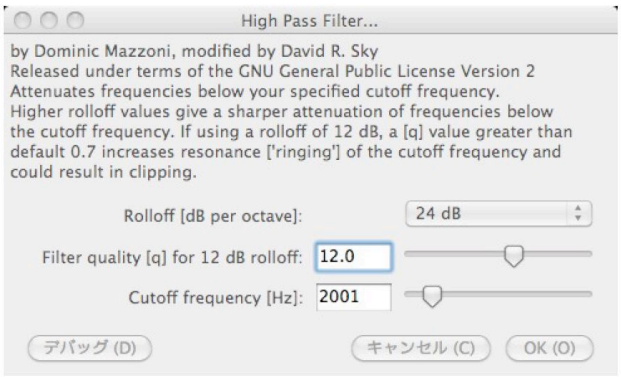




・削除したい方のチャンネルの一部を選択
・トラック　／　トラック削除
　（「トラック」メニューからサブメニュー「トラック削除を選ぶ」）
・ファイル名横の▼をクリックして「モノラル」を選ぶ
・ファイル　／　書き出し…　から
　“…/SampleFiles/センダイムシクイ Mono.wav” として保存
・（注意）Audacity から書き出した wav ファイルは、ヘッダーが破損しているので注意（一度 RavenLite に読み込んでから、同じ名前で書き出すと直る）

1－2　音声の編集、帯域フィルター

・マウスで音声の一部を選択して削除、コピー、貼付け等…
・音声全体を選択して
　効果 ／ High Pass Filter …　を選ぶ
・Rolloff : 24db、Filter quality : 12、
　Cutoff frequency : 2000Hz」を選ぶ
・ファイル／書き出し…
　“センダイムシクイ Flt.wav” として保存

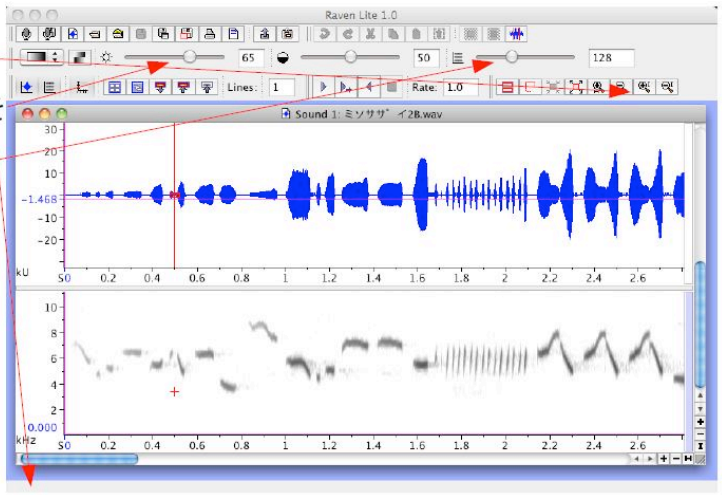


２．声紋の表示（RavenLite を使用）

RavenLite を起動する
File / Open Sound Files … を選択して
“…/SampleFiles/wren2B.wav” を開く

- Window 内をクリックすると
  時間、周波数、音圧レベルを表示
- 縦、横軸のスケールを変更
- 出力画像の明るさ、コントラストを
  変更
- 時間窓のサイズを変更
- 結果は File / Export Image Of

  でファイルに出力可能



３．音の分析と合成（R を使用）

３－１a　ファイルの読み込み

・R を起動する
・その他　／　作業ディレクトリの変更 …　を選ぶ
「/SampleFiles」を指定して「開く」を選択

```
library(sound)   #ライブラリーの読み込み　（R のコンソールにこの通り入力）
# .wav ファイルを Sample オブジェクトとして読み込む (sound : loadSample)
s<-loadSample("wren2B.wav")
s    # オブジェクトの情報を表示
```

・ヘルプの表示法
　コマンドについての説明

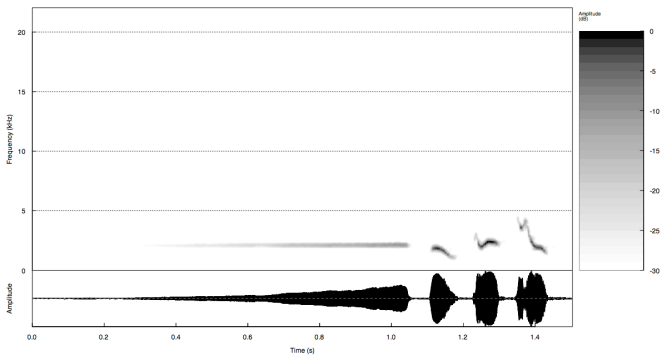
```
library(sound)
s<-loadSample("wren2B.wav")
# loadSample のヘルプ画面を表示
?loadSample
# loadSample のスクリプトを表示
loadSample
```

３－１ｂ　音データの演算、保存　［参考］

```
# オブジェクトのコピー
ss <- s

# 数値演算　１チャンネル（左チャンネル）の音圧を半分に
sound(ss)[1,] <- sound(ss)[1,]*0.5

#.wav ファイルに保存(sound：saveSample)
saveSample(ss,"wren_copy.wav")

#ch1 の内容をテキストファイルに出力
b<-loadSample("bush.wav")
library(seewave)
write.table(sound(b)[1,],file="bush.txt")
# ダウンサンプル　(tuneR: downsample)
library(tuneR)
s <- readWave("wren2B.wav")　#wav ファイルを wave に読み込み(tuneR: readWave)
ds <- downsample(s, 10000)　# ダウンサンプルする
writeWave(ds, "wren_10kHz.wav")
# 再度.wav ファイルに書き出す（Aliasing を確認）
```

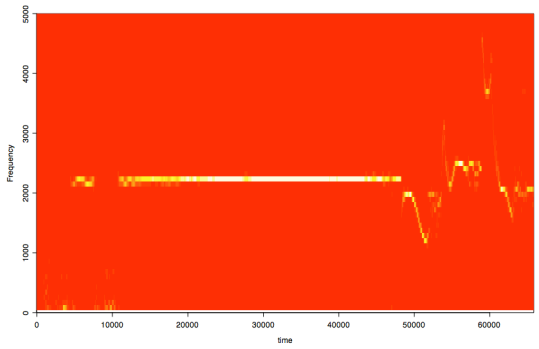
３－１ｃ　スペクトログラムを表示

```
# ファイルを sample オブジェクトとして読み込む（sound：loadSample）
library(sound)
s<-loadSample("bush.wav")
# 波形を表示（seewave: oscillo）
library(seewave)
oscillo(s)
# スペクトログラムを表示(seewave: spectro)
spectro(s)
#window サイズ　128 ポイント、グレイスケール、波形表示
spectro(s, wl=128, palette=rev.gray.colors.1, osc=TRUE)
```



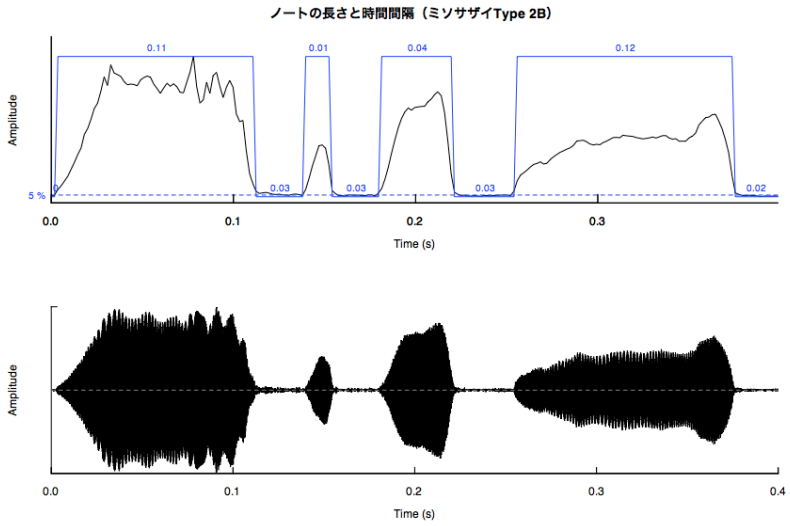
３−2　周波数分析（ピリオドグラム分析）

```
# .wav ファイルを wave オブジェクトとして読み込む（tuneR：readWave）
library(tuneR)
s <- readWave("bush44k.wav")
# ピリオドグラムを計算（tuneR：periodogram）オーバーラップを 64 に設定
p <- periodogram(s, normalize = TRUE, width = 512, overlap = 512-64)
image(p, ylim = c(0, 5000))
# 主成分(fundamental frequency)を抽出する
ff <- FF(p)
# 結果を図示する
plot(ff)
# 結果をテキストファイルに保存する
export(ff, f=22050, filename="ff.csv")
# if export does not work, use write.table instead of export
write.table(ff, file="ff.csv")
```



３−3　時間間隔を計測

```
library(sound)
# .wav ファイルを Sample オブジェクトとして読み込む（sound：loadSample）
s<-loadSample("wren_hp.wav")
# （audacity から出力したファイルは一度 raven lite に読み込み再度書き出す）
# 音声の一部を切り出す（seewave: cutw）
library(seewave)
a <- cutw(s, from=1.0, to=1.4)
# ２行１列の図形描画領域を確保
op<-par(mfrow=c(2,1))
# ノートの継続時間、時間間隔を計測(seewave: timer)
timer(a, f=22050,threshold=5,smooth=40,tck=0.05, bty="l",xaxs="i",colval="blue")
title(main="ノートの長さと時間間隔(ミソサザイ Type 2B)",col="blue")
oscillo(a, f=22050,k=1,j=1)
par(op)
```

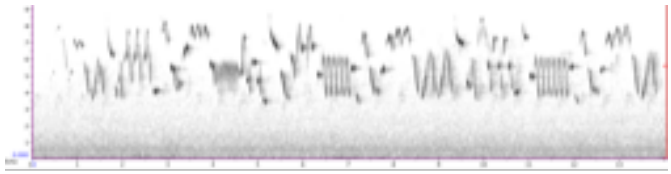


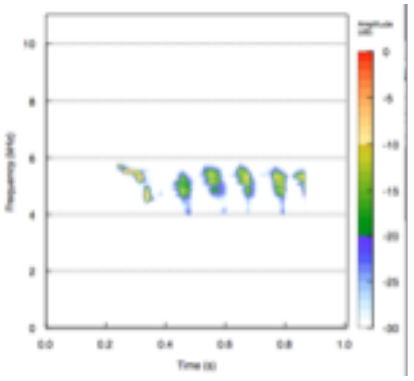
```
# plot パラメータを FALSE に設定すると、計測値を表示
# s 継続時間（単位：秒）
# p 時間間隔（単位：秒）
# r 無音部と信号の時間比
tt <- timer(s, f=22050,threshold=5,smooth=40,tck=0.05,bty="l",xaxs="i",colval="blue", plot=FALSE)
# 計測値をファイルに出力
write.table(tt$s, file="interval.txt")
write.table(tt$p, file="duration.txt")
```

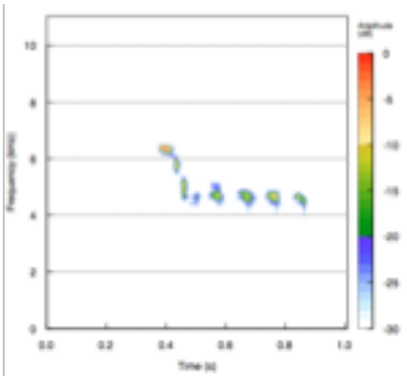
３－４　Cross-Covariance の計測

・２つの音声（スペクトログラム）間の類似度の指標となる
・すべてのサンプルについて、CC を総当たりで計算してマトリックスを作成
・→　多次元尺度法(Multidimensional Scaling)を用いてグループ化　など
・広帯域音など、複雑な信号の分析に有効

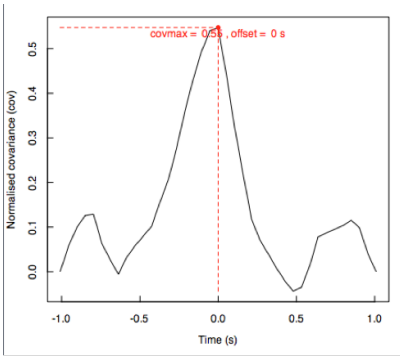
```
library(sound)
library(seewave)
w2d <- loadSample("2d_edit.wav")　# 音声の読み込み
s1 <- cutw(w2d, from = 3.7, to = 4.7)　# 音声一部を切り出す(3.7 □　4.7 s)
w3a <- loadSample("3a_edit.wav")
s2 <- cutw(w3a, from = 5.0, to = 6.0)
par (mfrow=c(1,1))
# Cross-Covariance の計測（seewave：covspectro)
covspectro(s1, s2, f=22050, n=39)
spectro(s1, f=22050)
spectro(s2, f=22050)
savewav(s1, f=22050, file="s1.wav")
savewav(s2, f=22050, file="s2.wav")
```





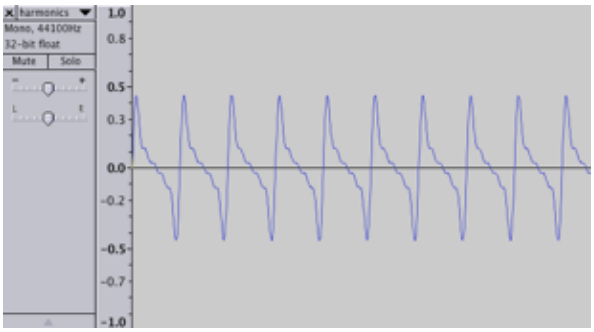


↓２つの信号を少しずつずらして繰り返し計算。結果が最大になった時の値を返す



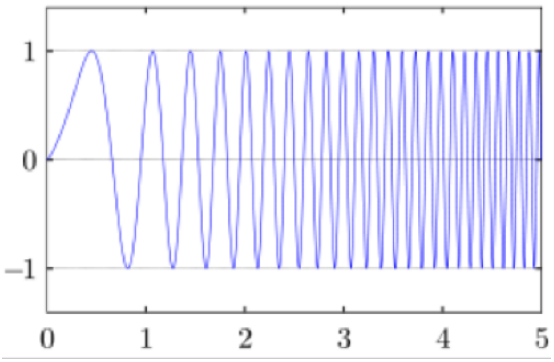
３−５a　音声の合成　（サイン波（純音）の合成）

```
# 時間のベクトルを作成（0 から１、8000 個のデータ→8000Hz で１秒間）
t <- seq(0, 1, length.out = 8000)
s <- sin(2 * pi * 440 * t)　# 440Hz のサイン波（純音）を作る
# Sample オブジェクトに変換（sound: asSample）
library(sound)
s1　<-　as.Sample(s, 8000, 16)
# wav ファイルに保存（sound: saveSample）
saveSample(s1,"sine.wav")
# 合成音（倍音）を作って wav ファイルに出力
t <- seq(0, 1, length.out = 44100)
s1 <- sin(2 * pi * 500 * t)
s2 <- sin(2 * pi * 1000 * t) * 0.8
s3 <- sin(2 * pi * 1500 * t) * 0.6
s4 <- sin(2 * pi * 2000 * t) * 0.4
s5 <- sin(2 * pi * 2500 * t) * 0.2
ss <- (s1+s2+s3+s4+s5)/5
ts　<-　as.Sample(ss, 44100, 16)
saveSample(ts,"harmonics.wav")
```



３−５b　音声の合成　（Chirp 音の合成）

```
library(sound)
t <- seq(0, 1, length.out = 44100)　# 時間のベクトルを作成（0 から１、44100 個のデータ）
f0 <- 1.0　# 定数　f0：周波数の初期値 k：周波数の比
k <- 1000.0
c <- sin(2 * pi * ( f0 + t * k / 2 ) *　t)
par (mfrow=c(1,1))　# 描画領域の初期化
plot(t, c, type="l", xlab = "Time (sec)", ylab="Sound Pressure Lebel")　# 波形の描画
library(seewave)
cs　<-　as.Sample(c, 44100, 16)
spectro(cs)　# スペクトログラムの表示
saveSample(cs, "chirp.wav")　# wav ファイルに保存
```



#　参考　http://en.wikipedia.org/wiki/Chirp_signal

In a linear chirp, the instantaneous frequency f(t) varies linearly with time:

f(t) = f0 + kt

where f0 is the starting frequency (at time t = 0), and k is the rate of frequency increase or chirp rate. The corresponding time-domain function for a sinusoidal linear chirp is:

$$x(t) = \sin\left[2\pi \int_0^t f(t')\,dt'\right] = \sin\left[2\pi \int_0^t (f_0 + kt')\,dt'\right] = \sin\left[2\pi(f_0 + \frac{k}{2}t)t\right]$$