国立国語研究所学術情報リポジトリ

Praatを活用した分析

メタデータ	言語: Japanese 出版者: 研究社 公開日: 2026-07-24 キーワード: 作成者: 西川, 賢哉 メールアドレス: 所属: 国立国語研究所プロジェクト非常勤研究員
URL	https://doi.org/10.15084/0002000727

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.



第7章｜Praatを活用した分析

本章では、音声分析ソフトウェアPraatを取り上げる。『日本語日常会話コーパス』(CEJC)では、韻律情報をPraatのTextGrid形式で提供している。TextGridはアノテーション用のファイルであり、対応する音声ファイルと合わせて表示することで、韻律情報を分析に活用できる。そこで本章の前半では、CEJCにおける韻律情報の概要を説明した上で(1節)、韻律情報を例にTextGridの操作方法を紹介する(2節)。後半では、Praatの特長の1つであるスクリプト機能を取り上げる。スクリプトは処理を自動化することができるため、これを学ぶことでCEJCの韻律情報をより効率的かつ柔軟に利用できる。そこで、Praatスクリプトの基本を解説し(3節)、続いて具体的な活用例を紹介する(4節)。

□1　CEJCにおける韻律情報

『日本語日常会話コーパス』(CEJC)で提供される**韻律情報**は、次の5つの層で構成される(小磯ほか 2020)。

単語層　発話を構成する音形と語の境界を示す。語の単位は短単位を採用している(第5章参照)。**有核語**の場合、実際に**アクセント核**が実現された位置に半角アポストロフ(')を付与する。無核語であるにもかかわらずアクセント核のような下降が見られる場合は、半角アステリスク(*)を付与する。

BI層　Break Index(BI)ラベルは、原則として1以上3以下の整数値とその中間値からなり、値が大きいほど**韻律境界**が強いことを示す。1は単語境界、2はアクセント句境界(かつ単語境界)、3はイントネーション句境界(かつ単語・アクセント句境界)に対応する。**アクセント句**とは、句頭の上昇を伴い、最大1つのアクセント核を有する句を、**イントネーション句**とはアクセント句のダウンステップの効果が続く範囲を指す(図1)。

また、フィラー音調に対応するBI値をFとするなど、非流暢性にかかわるラベルも定義されている。

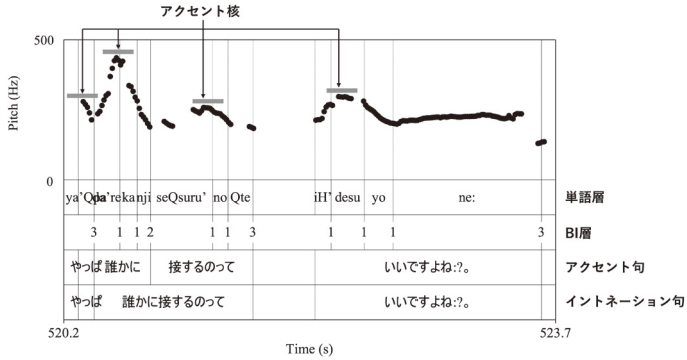


図1　単語・アクセント句・イントネーション句

トーン層　**句末境界音調**と**フィラー音調**をアクセント句末に付与する。句末境界音調には、L%（下降調）、H%（上昇調1）、LH%（上昇調2）、HL%（上昇下降調）、HLH%（上昇下降上昇調）、LHL%（下降上昇下降調）、HLHL%（上昇下降上昇下降調）がある。フィラー音調にはL$、H$、LH$、HL$、LHL$がある。

プロミネンス層　プロミネンス層のラベルは、トーン層・BI層のラベルの解釈を補助することを目的とする。X-JToBIに準拠し、PNLP、FR、HR、EUAPの4つのラベルを設けている。

注釈層　注釈層のラベルとして、AYOR（非東京方言発話の表示）とHBP（アクセント句頭が例外的に高いトーンで始まっていることの表示）を設けている。

なおCEJCの韻律情報は、**簡易版X-JToBI**に基づいて付与されている。簡易版X-JToBIは、『日本語話し言葉コーパス』（CSJ）の構築の際に整備された**X-JToBI**を簡略化したものである（五十嵐ほか 2006; 五十嵐 2015）。主な変更点は、トーンの種類を制限し付与する位置を簡略化したことと、分節音情報を省略可能としたことである。X-JToBIでは、トーンとして、句頭の境界音調や句頭音調、アクセント核、句末の境界音調などを時間軸の中で物理的に生起する位置に付与するのに対し、簡易版X-JToBIでは、トーンの種類を句末境界音調とフィラー音調に限り、付与する位置も当該音調が論理的に帰属するアクセント句末とする。

□2　TextGridの活用

Praatは、アムステルダム大学のPaul BoersmaとDavid Weeninkにより開発された、無償で提供される音声分析ソフトウェアである。Windows、Mac、

Linuxで利用でき、音声分析、聴取実験、音声の可視化、音声合成など多様な機能を備えている。その中でもよく利用される機能の1つが、**TextGrid**を用いたアノテーションの作成・操作である。

Praatでは、**音声波形**や**スペクトログラム**、**ピッチ**などの音声情報を参照しながらTextGridにアノテーションを施したり、既存のTextGridを研究に利用したりすることができる。本節では、CEJCの韻律情報を格納したTextGridを例に、Praat上での基本的な操作方法を紹介する。操作の方法にはいくつかあるが、ここではよく用いられる方法にしぼって説明する。以下の図や説明は、Mac版Praatバージョン6.4.38に基づく。

CEJCの韻律情報を格納したTextGridは8つの層から構成される（図2）。このうち5つは1節で説明した層に対応している。さらに利便性のために、trans層（単語層に示される語の、転記タグ付き漢字仮名交じり表記）、AP層（アクセント句）、IP層（イントネーション句）が追加されている。

なおTextGridの層には、**インターバル層**（開始時刻と終了時刻を持つラベルで構成）と、**ポイント層**（1つの時刻のみを持つラベルで構成）の2種類がある。CEJCの韻律情報では、Word層、trans層、AP層、IP層がインターバル層、それ以外がポイント層である。

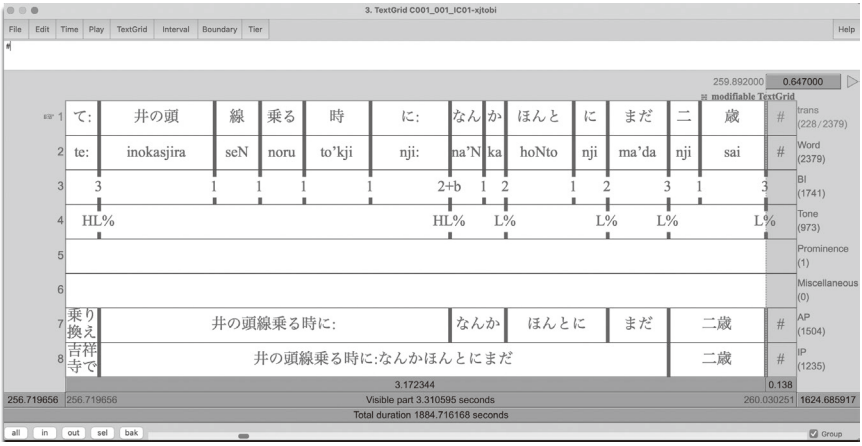


図2　CEJC韻律TextGrid

2.1　TextGridEditorの起動

Praatのアイコンをクリックして起動すると、**オブジェクトウインドウ**（**Object window**、3.1節で詳述）と**ピクチャーウインドウ**（**Picture window**）[1]が開く。オブジェクトウインドウ上で、次の手順により**TextGridEditor**を起動する（図3参照）。

1. オブジェクトウインドウの「Open」メニューから「Read from file...」を選択し、音声ファイルとTextGridファイルをそれぞれ読み込む。読み込まれたファイルは左側のObjects欄に表示される。
2. 音声ファイルとTextGridファイルをマウスでドラッグして選択すると、右欄にメニューが現れる。
3. メニューから「View & Edit」をクリックすると、TextGridEditorが起動し、音声（音声波形やスペクトログラムなど）とTextGridの情報が表示される。

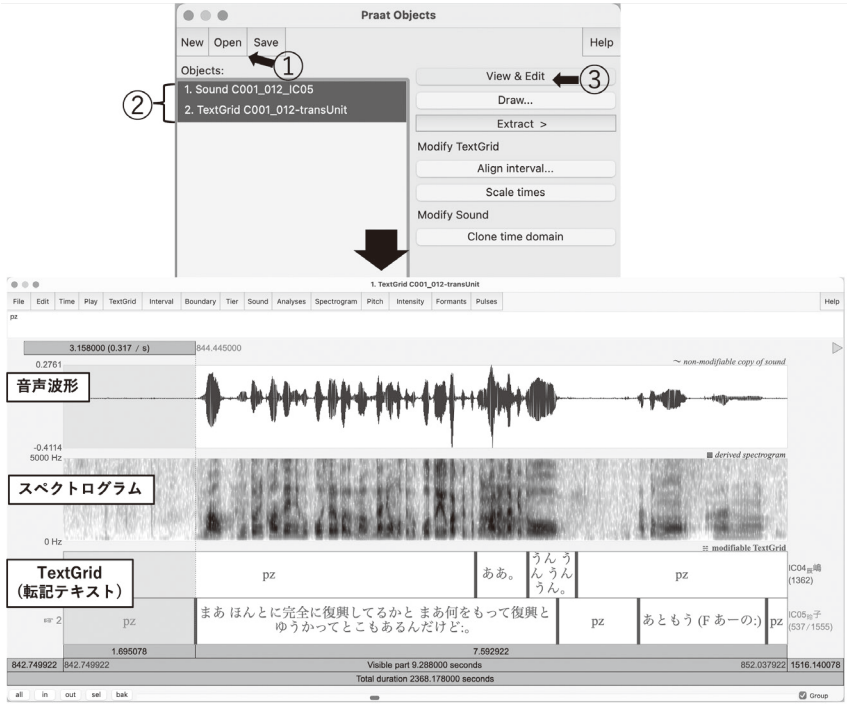


図3　ファイルを読み込みTextGridEditorを表示

1| ピクチャーウインドウは音声波形やピッチ曲線などを可視化し、作図する際などに利用する。

2.2　表示と音声の再生

TextGridEditorでは、アノテーションのラベルを表示したり、ピッチやフォルマントを表示したり、音声を再生したりすることができる。基本的な操作は以下のとおりである（図4参照）。

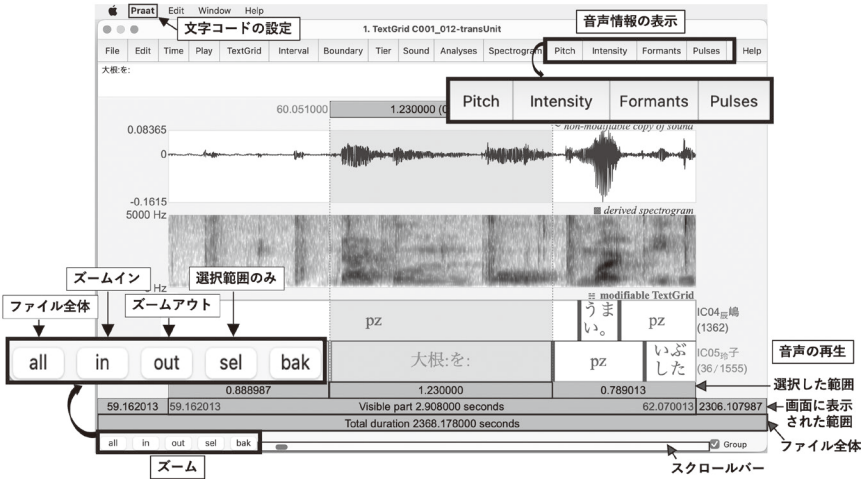


図4　表示と音声の再生

文字コードの設定　「Praat」メニュー[2]から「Settings」を選択し、「Text reading settings...」や「Text writing settings...」で文字コードを指定する。CEJCで提供するファイルはUTF-8である。

フォントの設定　一部の環境では、中国語フォントで表示される。日本語フォントでの表示に変更するには、「Praat」メニューから「Settings」→「CJK font style settings...」を選択し、「Japanese」を指定する。

音声情報の表示の選択　上部メニューから「Pitch」「Intensity」「Formants」を選び、それぞれ「Show pitch」など「Show 〜」をクリックすると、これらがスペクトログラムの画面に重ねて表示される。また「Pulses」から「Show pulses」を選ぶと、波形に重ねてパルスの情報が表示される。

ズームイン・ズームアウト　左下のボタンで、ズームイン（in）、ズームアウト（out）、ファイル全体の表示（all）、選択範囲のみの表示（sel）ができる。

2｜「Praat」メニューはOSによって位置が異なる。Macの場合は画面上部のメニューバー（Appleマークの右側）に、Windowsの場合はオブジェクトウインドウに表示される。

スクロール　画面下部のスクロールバーで、任意の位置に移動できる。

範囲の選択　範囲を指定するには、音声波形かスペクトログラム上でマウスでドラッグするか、TextGridのインターバル層のラベルをクリックする。

音声の再生　音声の再生は、タブキーか画面下の再生バーを用いる。1段目は選択した範囲、2段目は画面に表示されている範囲、3段目はファイル全体を再生する。

2.3　ラベルの修正・層の操作

TextGridに含まれるラベルは、内容を変更したり位置を調整したりすることができる。ここでは基本的な操作を紹介する。

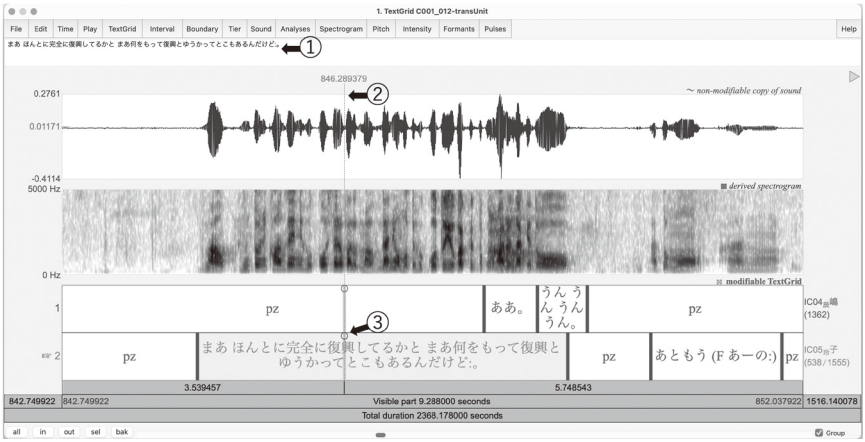


図5　TextGridラベル修正・層の操作

ラベル内容の変更　変更したいラベルをクリックすると、TextGridEditor上部のテキストボックスに内容が表示される（図5 ①）。ここでラベルを修正する。

区切り線の追加1　音声波形またはスペクトログラム上で追加したい位置をクリックすると、TextGridの全ての層に渡って縦線が表示され（図5 ②）、線と各層の交点に小さな丸が現れる（図5 ③）。追加したい層の丸をクリックすると、区切り線が追加される。

区切り線の追加2　既存の区切り線と同じ位置に別の層への区切り線を追加する場合、既存の線をクリックすると図5 ②③のような縦線と丸が表示されるため、該当する丸をクリックして追加する。

区切り線の削除　削除したい区切り線を選択し、Alt + Backspaceを押す。
区切り線の移動　区切り線をマウスでドラッグして移動する。

また、新たな層を作成して追加のアノテーションや備考情報の記録などを行うことができる。既存の層を複製し、オリジナルを残したままアノテーションを修正したり、不要な層を削除することもできる。手順を以下に記す。

層の追加　「Tier」メニューから「Add interval tier...」（インターバル層の場合）または「Add point tier...」（ポイント層の場合）を選び、層の位置と名前を指定して「OK」をクリックする。
層の複製　複製したい層を選択（層の左右の番号あるいは層名をクリック、あるいは適当なラベルを選択）し、「Tier」メニューから「Duplicate tier」を選んで層の位置と名前を指定する。
層の削除　削除する層を選び、「Tier」メニューから「Remove entire tier」を選ぶ。この方法では層全体が削除される。「Remove all text from tier」を選択すると、層の全てのラベルが削除されるが、層や区切り線は残る。

2.4　F0値の取得

図6に示すように、範囲を指定した上で「Pitch」メニューから「Pitch listing」を選ぶと、その範囲のピッチ（F0, *fo*）の情報を一覧で得ることができる。また「Get minimum pitch」や「Get maximum pitch」を選ぶと、指定範囲のF0最小値あるいはF0最大値が取得される。さらに、範囲ではなく一時点を指定して「Get pitch」を選ぶと、その時点のF0値が取得される。

図6は、有核句「ダ'レカニ」を対象に、その範囲のピッチ情報を取得した事例である。聴取して確認する必要はあるが、これらのうちF0値が最大のものが、実際のアクセント核のF0値や生起時刻である可能性は高い。1節で言及したように、CEJCが準拠した簡易版X-JToBIでは、ラベリング対象とするトーンは句末境界音調とフィラー音調に限定されており、付与位置も対応するアクセント句末に限られている。そのため、実際に生起したトーンの時刻やF0値を分析に利用するには、このような方法で調べる必要がある。

ただし、全ての事例を人手で確認するのは大きな労力を要する。そこで役立つのが、3節以降で紹介するPraatスクリプトである。実践例として、有核語を含むアクセント句区間のF0最大値を取得するスクリプトを4.2節で取り上げる。

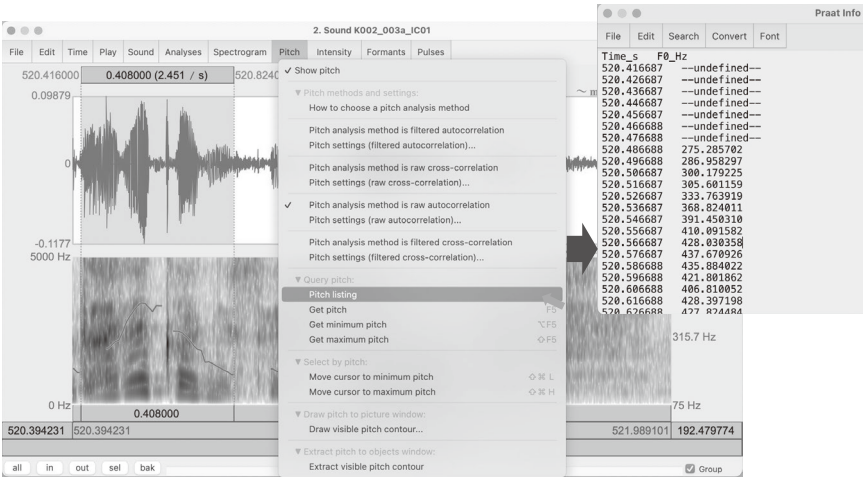


図6　有核句「ダ゛レカニ」の範囲のピッチ情報

□3　Praatスクリプトの基礎

3.1　オブジェクトウインドウ

本節では、Praatの主要なウインドウであるオブジェクトウインドウ（ウインドウのタイトルはPraat Objects）に基づき、Praatの基礎を説明する。本節の内容は、後に扱う**Praatスクリプト**の基礎ともなる。

図7にオブジェクトウインドウの構成を示す。オブジェクトウインドウのうち、メニュー、ダイナミックメニュー、固定ボタンには、様々なコマンドが割り当てられている。メニュー内のコマンドを**メニューコマンド**、ダイナミックメニュー内のコマンドを**アクションコマンド**という。

オブジェクトの生成　Praatでは、データを**オブジェクト**という単位で管理する。利用可能なオブジェクトは**オブジェクトリスト**に表示される。オブジェクトを生成する方法には以下がある。

- ファイルを読み込む（例：「Open」メニュー内の「Read from file...」）
- 既存のオブジェクトから生成する（ダイナミックメニュー内の一部のコマンドを使用）
- 新規に生成する（例：「New」メニュー内の「Record stereo Sound...」）

オブジェクトの表示構成・ID・タイプ　オブジェクトリストでは、各オブジェクトが、(1) オブジェクトID、(2) オブジェクトタイプ、(3) オブジェクト

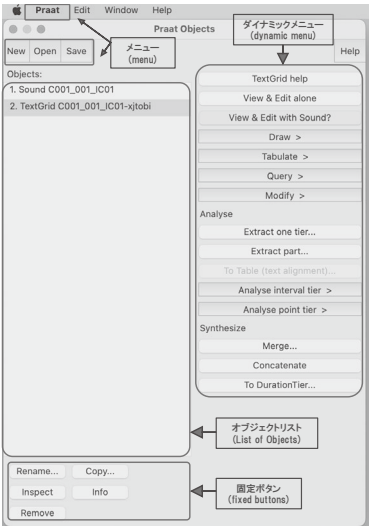


図7　Praatオブジェクトウインドウ

名という3つの情報とともに表示される。オブジェクトIDは自動的に割り当てられるユニークな番号である。スクリプトでオブジェクトを指定する際に用いる。オブジェクトには様々な種類がある。代表的なオブジェクトタイプに、Sound（音声）、LongSound（長い音声）、TextGrid（アノテーション）、Pitch（ピッチ）などがある[3]。

オブジェクトの選択　オブジェクトに対して何らかの操作をする場合、対象オブジェクトを選択する。手動で選択する場合、オブジェクトリストに表示されているオブジェクトをマウスでクリックあるいはドラッグする。

ダイナミックメニュー　ダイナミックメニューの内容は、選択されているオブジェクトタイプによって変化する。例えば2.1節で見たように、音声ファイルとTextGridファイルを選択すると、「View & Edit」などのメニューが表示される。オブジェクトを何も選択しない場合は表示されない。ダイナミックメニューとして、オブジェクトタイプに特化したアクションコマンドが用意されている。Praatスクリプトでは特に「Query」や「Modify」内にあるコマンドが頻繁に用いられる。特に「Query」内のコマンドは、後に見るように、

3| 音響的なオブジェクトに加えて、表形式のデータを扱うTableオブジェクトといったものも用意されている。「Open」メニュー内の「Read Table from tab-separated file」や「Read Table from comma-separated file」などを参照のこと。

コーパス操作において多用される。

固定ボタン　固定ボタンは、オブジェクトリスト上のオブジェクトに対する一般的な操作（Rename：オブジェクト名の変更、Copy：コピー、Remove：削除など）を行う際に使用する。ダイナミックメニューと異なり、オブジェクトタイプによって内容が変化することはない。

3.2　スクリプトの作成と実行の流れ

Praatでは、スクリプトを扱うために**スクリプトエディタ**を用いる。ここでは、スクリプトの作成から実行までの基本的な手順を体験してみよう。最初の例として、プログラミング入門書でよく用いられる「Hello, world!という文字列を出力する」スクリプトを作成する（図8）。手順は次の通りである。

1. 「Praat」メニューから「New Praat script」を選択し、スクリプトエディタを起動する。
2. スクリプトエディタに次のスクリプトを入力する（Praatスクリプトでは大文字・小文字が区別されることに注意）。

```
writeInfoLine: "Hello, world!"
```

3. スクリプトエディタの「Run」メニューから「Run」を実行する。
4. **Infoウインドウ**に結果が表示される。

スクリプトを保存するには、スクリプトエディタの「File」メニューから「Save」を選択する。Praatスクリプトの拡張子は.praatを用いるのが一般的である。保存したスクリプトを開くには、オブジェクトウインドウの「File」メニューから「Open praat script...」を選択するか、スクリプトエディタの「File」メニューから「Open...」を実行する。

3.3　履歴機能

スクリプトエディタには、**履歴機能**という便利な機能がある。オブジェクトウインドウ上で行った手動での操作が記録されており、それをスクリプトとして書き出すことができる。

ここでは例として、オブジェクトウインドウで純音（pure tone）を再生する操作をスクリプト化してみよう。手順は次の通りである。

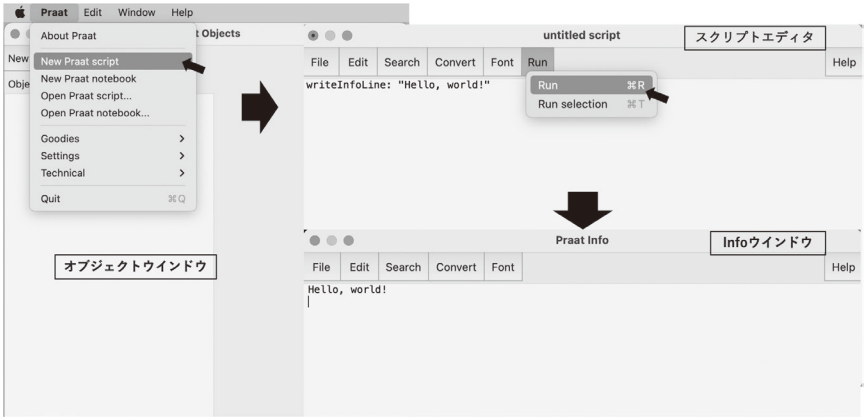


図8　スクリプトの作成と実行

1. スクリプトエディタ上の操作：メニュー「Edit」→「Clear history」を実行し、これまでの履歴を消去する。
2. オブジェクトウインドウ上の操作：次の手順で純音を再生する。
   (a)「New」メニューから「Sound」→「Create Sound as pure tone...」を実行。表示されたウインドウでそのまま「OK」をクリックすると、オブジェクトリストにtoneというオブジェクトが生成される。
   (b) ダイナミックメニューの「Play」をクリックし、音を再生する。
   (c) 固定ボタンの「Remove」をクリックして、オブジェクトを削除する。
3. スクリプトエディタ上の操作：メニュー「Edit」→「Paste history」を実行すると、以下のように履歴がスクリプトの形でエディタに貼り付けられる(「New Praat script」など、不要な行があったら、その行を削除する)。

```
Create Sound as pure tone: "tone", 1, 0, 0.4, 44100, 440,
    0.2, 0.01, 0.01
Play
Remove
```

これでスクリプトは完成である。メニュー「Run」→「Run」で実行し、同じ純音が再生されるかを確認してみよう。

このように、スクリプト化したい操作がある場合には、まず手動で実行し、その履歴をスクリプトエディタに貼り付ける方法が有効である。実際には貼り付けたスクリプトを修正する必要がある場合も多いが、スクリプト作成の大き

な助けとなる。

3.4 Praatスクリプトの仕様

本節では、Praatスクリプトの仕様について解説する。網羅的ではないが、本章で扱うスクリプトに必要な内容を中心に取り上げる。厳密さよりもわかりやすさを優先して説明する。

3.4.1 コメント

スクリプトには**コメント**を付けることができる。コメントは処理時には無視される。「#」または「;」を用い、そこから行末までがコメントとなる。このうち「#」は必ず行頭で使用する（行全体がコメントとなる）。「;」は行中でも用いることができる。

```
# この行はコメント（処理時には無視される）
writeInfoLine: "Hello, world!"
writeInfoLine: "Hello, world!"; 行の途中にコメントを書く場合
```

3.4.2 メニューコマンド・アクションコマンドの実行

Praatのオブジェクトウインドウの各種メニュー（メニュー、ダイナミックメニュー、固定ボタン）から実行できるコマンドは、スクリプト内でコマンド名を書くことで実行できる。3.3節で紹介した例では、音声オブジェクトの再生（ダイナミックメニューの「Play」）やオブジェクトの削除（固定ボタンの「Remove」）がこれに相当する。

コマンドの中には、**引数（argument）**を要求するものがある。3.3節の例では、純音の生成（オブジェクトウインドウ上での「Create Sound as pure tone...」）がこれに相当する。名前やチャンネル数などを指定するダイアログボックスが表示されるが、ここで指定するのが引数である。こうしたコマンドをスクリプトから実行するには、末尾の「...」を「:」に置き換え、そのあとに引数を指定する。複数の引数を指定する場合は引数を「,」で区切る。

3.4.3 変数・変数への代入

変数（variable） 変数とは数字や文字列を格納する箱のようなものである。変数名には大文字・小文字・アンダースコア（_）を使用する。ただし、先頭は

必ず小文字でなければならない（大文字で始まる語句はメニューコマンドと解釈される）。Praatスクリプトの変数には、数値型と文字列型の2種類の型があり、文字列型変数名には末尾に「$」を付加する。

変数への代入・基本的な方法　代入演算子「=」を用い、右辺の値（数字や文字列）を左辺の変数に代入する。文字列の場合は二重引用符で囲む。

```
num = 10              ; 数値型変数numに数値10を代入する
favorite$ = "sushi" ; 文字列型変数favorite$に文字列「sushi」を代入
                        する
```

変数への代入・Praatスクリプトに特徴的な方法　Infoウインドウに出力するコマンド（ダイナミックメニュー「Query」内のアクションコマンド、固定ボタン内の「Info」コマンド）を代入演算子「=」の右辺に置くと、そのコマンドの出力を左辺の変数に代入することができる。

例えば、Soundオブジェクトのダイナミックメニュー「Query」の「Query time domain」に「Get total duration」（持続長取得）というアクションコマンドがある。通常、Infoウインドウに値が出力されるが、以下のように代入演算子の右辺に置くと、変数durにコマンドの出力が代入される。

```
dur = Get total duration
```

また、オブジェクトを生成するコマンドを代入演算子「=」の右辺に置くと、生成されたオブジェクトのIDが左辺の変数に代入される。例えば、「Open」メニューの「Read from file...」は、ファイルを読み込み、新しいオブジェクトを生成するコマンドだが、次のように書くと、オブジェクト生成と同時に、そのオブジェクトIDが左辺の変数sndIDに代入される。

```
sndID = Read from file: "/Users/ken/CEJC/wav/C001_001_
    IC01.wav"
```

組み込み変数・定数　Praatには、あらかじめ用意された組み込み変数や定数がある。一部を表1に示す。

表1　Praatスクリプトの組み込み変数・定数（一部）

tab$	タブ
newline$	改行
shellDirectory$	Praat起動時のディレクトリ
praatVersion$	Praatのバージョン（例：6.4.38）
pi	3.141592653589793
e	2.718281828459045

3.4.4　フォーム

フォーム（**form**）という機能を使うことで、ユーザーが入力した値を変数に代入し、スクリプト内でその値を使用することができる。例えば、以下をスクリプトエディタで実行すると、図9のような設定ウインドウが表示される。

```
form test
  sentence yourFavorite sushi
endform

writeInfoLine: "Your favorite is ", yourFavorite$, "."
```

初期値として「sushi」と入っているが、これを好きな言葉、例えば「udon」に書き換えて「OK」を押すと、Infoウインドウに「Your favorite is udon.」と表示される。



図9　設定ウインドウ（フォーム）

formの基本的な書き方は次の通りである。

form

```
form［タイトル］
    ［フィールド型］ ［変数名］ ［初期値］
endform
```

タイトル　設定ウインドウに表示されるタイトル

フィールド型　入力の種類（例：real 実数、integer 整数、sentence 文字列、infile 入力ファイル、outfile 出力ファイル、folder フォルダなど[4]）

変数名　入力を受け取る変数の名前[5]

初期値　フォームで最初に表示される値

3.4.5　コマンド

Praatには、オブジェクトウインドウから利用可能なメニューコマンドやアクションコマンドとは別に、Praatスクリプトで使用できるコマンドが提供されている[6]。いくつか取り上げて解説する。

(A) オブジェクトを操作するコマンド

`selectObject`　指定したオブジェクトを選択する。番号（オブジェクトID）で指定するのが一般的。

```
selectObject: 1
selectObject: 1, 2 ; 複数指定する場合は,で区切る
```

`removeObject`　指定したオブジェクトを削除する。

```
removeObject: 1, 2
```

固定ボタンのRemoveでもオブジェクトの削除は可能だが、Removeはその時点で選択されているオブジェクトを削除するのに対し、`removeObject`は、オブジェクトの選択状態にかかわらず指定したオブジェクトを削除する。

(B) Infoウインドウにかかわるコマンド

`writeInfoLine`　指定された引数をInfoウインドウに表示（末尾に改行あり、

4| infileやfolderなどは比較的最近導入されたフィールド型である。導入前はファイルやフォルダの指定にsentenceが使用されていた。

5| 文字列型変数であっても、form内では「$」は付加しない。yourFavorite$の例のように、form内で指定した変数をform外で使用するとき、文字列型であれば「$」を付加する。

6| ここで取り上げたものの多くは、公式にはコマンドではなく、関数として扱われているが、わかりやすさのため、直観的に関数とはみなしにくいものをコマンドとして紹介する。

上書き表示)。「,」で区切って複数の引数を与えることもできる。

```
name$ = "Ken"
age = 30
writeInfoLine: name$, " is ", age, " years old."
# 「Ken is 30 years old.」と表示される
```

writeInfo　改行なしで上書き表示。

appendInfoLine・appendInfo　既存のInfoウインドウの内容に追加表示。

clearinfo　Infoウインドウの内容を消去。

(C) その他のコマンド

exitScript　エラーメッセージを表示してスクリプト処理を中止。引数の指定方法はwriteInfoLineと同じ。

pauseScript　メッセージを出し、処理を続けるか止めるかをユーザーに選ばせる。引数の指定方法はwriteInfoLineと同じ。

nocheck　以下のように、コマンドの直前(行頭)に置き、そのコマンドでエラーが発生しても、それを無視して処理を継続させる。

```
# removeObjectの対象が存在しないオブジェクトでもエラーを無視して処理を続行
nocheck removeObject: 100
```

3.5　算術演算子・算術関数

Praatには**算術演算子**や**算術関数**が豊富に用意されている。表2にその一部を示す。

表2　Praatスクリプトの算術演算子・関数(一部)

演算子／関数	説明	例	出力
+,-,*,/	加算・減算・乗算・除算	4 / 2 + 2 * 3	8
^	べき乗	2^3	8
div	整数除算	10 div 3	3
mod	剰余	10 mod 3	1
abs()	絶対値	abs(-2)	2
round()	小数四捨五入	round(0.5)	1

3.6　文字列演算子・文字列関数

文字列演算子と**文字列関数**の一部を表3に示す。

表3　Praatスクリプトの文字列演算子・関数（一部）

演算子／関数	出力
a$ + b$	a$とb$を連結
a$ - b$	a$から末尾のb$を削除
length(a$)	a$の文字数
mid$(a$, m, n)	a$のm文字目から始まるn文字分の文字列
index(a$, b$)	a$におけるb$の最初の出現位置（なければ0）
startsWith(a$, b$)	a$の先頭がb$であれば1、さもなくば0
fixed$(m, n)	m（数値）を小数第n位で丸めた文字列

3.7　条件分岐：if

if文は条件分岐を表す。最初に成り立った条件の処理だけが実行される。基本形は次のとおり。

if文

```
if [式1]     ; 式1の条件が成立すれば処理1を実行、式2以降見ずに終了
  [処理1]
elsif [式2] ; 式1が成立しない場合に式2の条件が成立するか検証
  [処理2]
else         ; 式1,式2とも成立しない場合に処理3を実行して終える
  [処理3]
endif        ; ifと対になりif～endifまでが1つのif文となる
```

式と処理はそれぞれ1行に記す。可読性を高めるため字下げするとよい。elsifとelseは省略可能である。またelsifは複数個使用できる。以下は試験の点数に応じて評価を決めるif文である。実行すると「良」と表示される。

```
score = 76
if score >= 80
  grade$ = "優"
elsif score >= 60
  grade$ = "良"
```

```
elsif score >= 40
  grade$ = "可"
else
  grade$ = "不可"
endif
writeInfoLine: grade$
```

［式］でよく使用される**比較演算子**と**論理演算子**を表4に示す。なお、文字列比較にも数値比較と同じ演算子が使用できる。

表4　Praatスクリプトの比較演算子・論理演算子（一部）

演算子	説明
x == y, x = y	xがyに等しい
x <> y, x != y	xがyに等しくない
x > y, x >= y	xがyより大きい、あるいはそれ以上
x < y, x <= y	xがyより小さい、あるいはそれ以下
x and y, x && y	xとyのどちらも真
x or y, x || y	xかyのどちらかが真

3.8　ループ（1）：for

for文は、指定した始点から終点まで、1ずつ順に値を増やしながら処理を繰り返す。

for文

```
for［ループカウンタ（変数）］from［始点（数値）］to［終点（数値）］
    ［処理］
endfor　；forと対になりfor～endforまでが1つのfor文となる
```

以下の例では、1から5の数字が順にInfoウインドウに追加表示される。

```
clearinfo
for i from 1 to 5　; 始点が1の場合、「from 1」は省略可能
  appendInfoLine: i
endfor
```

3.9 ループ(2)：while

while文は、条件が成り立つ間、指定した処理を繰り返す。

while文

```
while [式]  ; [式](条件)が成り立つ限り次の[処理]が実行される
  [処理]
endwhile  ; whileと対になりwhile～endwhileまでが1つの
            while文となる
```

Praatのfor文では、ループカウンタを2ずつ増やしたり、1ずつ減らしたりといったことができない。このような処理は、以下のようにwhile文で対応する。実行すると、「5, 4, 3, 2, 1」と順に表示される。

```
clearinfo
n = 5                    ; 初期値として変数nに5を代入
while n >= 1             ; 変数nが1以上の場合に以下の2つの処理を実行
  appendInfoLine: n      ; 処理1変数nをInfoウインドウに追加表示
  n = n - 1              ; 処理2変数nから1を減らす
endwhile
```

3.10 ジャンプ：goto

goto文は、指定したラベルの位置にジャンプする。ジャンプ先のラベルは、labelで指定する。

goto文

```
goto [ラベル]
# 略
label [ラベル]
```

forやwhileなどの繰り返し処理を途中で抜ける際に使用する。例えば以下では、1から順に変数iが1ずつ増加し、if文の条件にあるように5に達した段階で、for文を抜けてFINというラベルを付けた位置にジャンプする。

```
clearinfo
for i to 10              ; iが1から10まで繰り返す
  if i == 5              ; iが5の場合
    #for文を抜けてFINというラベルを付けた位置にジャンプ
    goto FIN
  endif
  appendInfoLine: i      ; iが1〜4まではiの値をInfoウインドウに追加表
                           示
endfor

#ラベルは目立つようにアルファベット大文字を使うとよい
label FIN

appendInfoLine: "Fin" ; iが5の場合にここ移動してFinと追加表示
```

なお、gotoおよびlabelの行には、行中コメントを添えることができない。

□4　Praatスクリプトの活用

4.1　活用事例1：Pitchファイルの生成

以上を踏まえ、具体的なスクリプトを作ってみよう。Praatには、Pitchというオブジェクトタイプがあり、ピッチ（F0, *fo*）の情報を保持する。

課題

音声ファイル（.wav）からPitchオブジェクトを生成し、ファイルに保存するスクリプトを作成する。保存先は音声ファイルと同じフォルダ、拡張子は.Pitchとする。

オブジェクトウインドウ上で手動操作を行い、その後スクリプト化していく。手動操作の手順は次の通りである。

1. メニュー「Open」→「Read from file...」で、特定の音声ファイル（例えば /Users/ken/CEJC/wav/C001_001_IC01.wav）を読み込む。
2. ダイナミックメニュー「Analyse periodicity...」→「To Pitch（raw autocorrelation）...」[7]を実行し、Pitchオブジェクトを生成する（ダイアログボックスで表示される引数はデフォルトのものとする）。

7| ピッチ分析の方法は様々あるが、CEJCの音声について言えば、Praatで一般的に推奨される「To Pitch (filtered autocorrelation) ...」よりも、「To Pitch (raw autocorrelation) ...」のほうが、妥当な値を推定するようなので、ここでは後者を用いる。

3. メニュー「Save」→「Save as binary file...」[8]を選び、生成されたPitchオブジェクトを保存する。

この操作を履歴機能（3.3節）でスクリプトに書き出すと、次のようになる。

```
1 Read from file: "/Users/ken/CEJC/wav/C001_001_IC01.wav"
2 To Pitch (raw autocorrelation): 0, 75, 600, 15, "no",
     0.03, 0.45, 0.01, 0.35, 0.14
3 Save as binary file: "/Users/ken/CEJC/wav/C001_001_IC01.
     Pitch"
```

これを以下のように書き換える。

```
1  # 履歴1行目　音声ファイルをformでユーザーが指定できるように変更
2  form Wav2Pitch
3    infile wavFile /Users/ken/CEJC/wav/C001_001_IC01.wav
4  endform
5  Read from file: wavFile$; 変数wavFile$を対象に開く
6
7  # 履歴2行目　変更せず
8  To Pitch (raw autocorrelation): 0, 75, 600, 15, "no",
     0.03, 0.45, 0.01, 0.35, 0.14
9
10 # 履歴3行目　保存するピッチのファイル名を音声ファイル名から取得
11 ptcFile$ = wavFile$ - ".wav" ; 音声ファイル名から".wav"を削除
12 ptcFile$ = ptcFile$ + ".Pitch" ; 上記に".Pitch"を追加
13 Save as binary file: ptcFile$ ; 設定したptcFile$名で保存
```

音声ファイルの設定　履歴スクリプトの1行目では、読み込む音声ファイルが固定されている。そこでform（3.4.4節）を用いて、ユーザーが任意の音声ファイルを指定できるようにする（2-4行目）。さらに、「Read from file」の引数を、formで設定した変数wavFile$に置き換える（5行目）。

保存ファイルの設定　履歴スクリプトの3行目では、Pitchファイルの保存先のファイル名も固定されている。そこで文字列演算子（3.6節、表3）を使い、変数wavFile$の末尾から.wavを「-」で削除し、代わりに「+」で.Pitchを追加する（11,12行目）。こうして得られた変数ptcFile$を「Save as binary file」の引数とする（13行目）。

8| Pitchファイルを「Save as text file...」で保存すると、サイズが大きくなり、読み込みに若干時間がかかるので、ここではバイナリで保存する。

4.2　活用事例2：有核語を含むアクセント句区間のF0最大値

続いて、より本格的な課題に取り組む。

課題

有核語を含むアクセント句について、その区間のF0最大値を取得し、その時刻と値を、韻律TextGridに新規に追加した層に記録する（図10参照）。

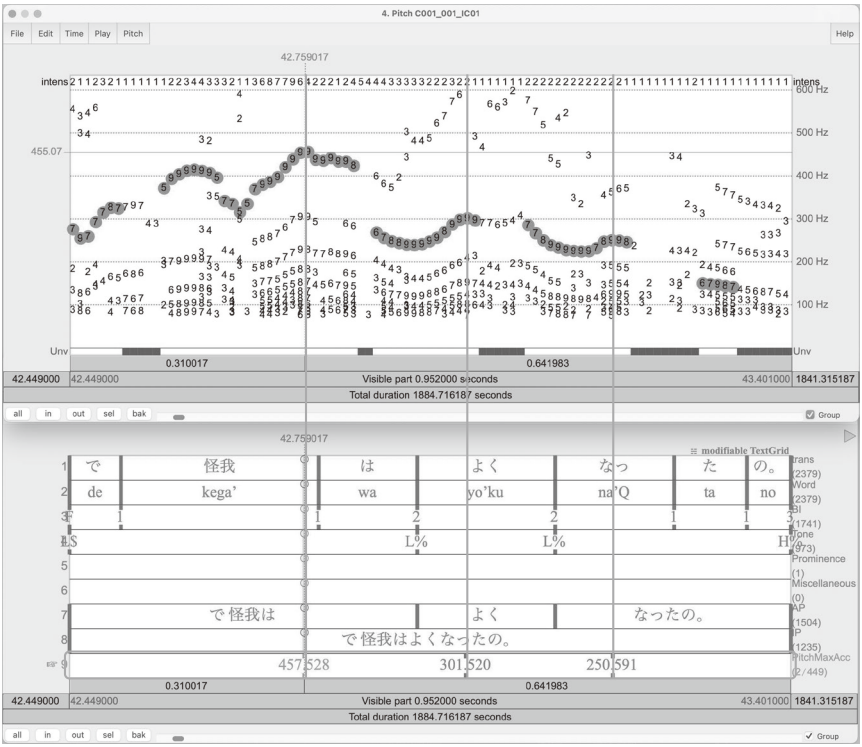


図10　TextGridオブジェクト（下）とPitchオブジェクト（上）
（TextGrid第9層にPitchMaxAcc層が追加されている）

1節や2.4節でも言及したように、簡易版X-JToBIでは、アクセント核の物理的生起位置は記述されていない。そこで、アクセント句区間のF0最大値をその近似値として用いる、というのが、この課題の背後にある考え方である（五十嵐 2015: 96）[9]。

まず、課題実現への見通しを立ててみよう。1節で述べたように、単語層にはアクセント核の情報が付与されている。そのため、有核語であるか否かは、

単語層のラベルにアクセント核（'）が含まれるか否かで判定できる。またCEJCの韻律情報のTextGridにはAP（アクセント句）層も提供されている（2節）。そのため、単語層とAP層を組み合わせて参照することで、対象とする区間（有核語を含むアクセント句区間）を同定できる。

次に、その区間のF0最大値とその時刻の取得であるが、これは前節で生成したPitchオブジェクトを参照することにする。

完成版のスクリプトを次に示す。細部にはこだわらず、全体の流れを確認してほしい。

```
# 有核語を含むアクセント句について、
# その区間のF0最大値を取得し、その時刻と値を、
# 韻律TextGridに新規に追加した層に記録する

form maxpitchap
  infile tgFile /Users/ken/CEJC/xjtobi/C001_001_IC01-xjtobi.TextGrid
  infile ptcFile /Users/ken/CEJC/wav/C001_001_IC01.Pitch
endform

# 新規層の名前
ptcTierName$ = "PitchMaxAcc"

# ファイル読み込み
ptcID  = Read from file: ptcFile$
tgID   = Read from file: tgFile$

# 関連する層番号の取得
wordTierNum = 0
apTierNum = 0
ptcTierNum = 0
selectObject: tgID
n = Get number of tiers
for i to n
  curTierName$ = Get tier name: i
  if curTierName$ == "Word"
```

9| 五十嵐（2015: 96）で指摘されているように、この考え方には問題がある。分析対象がBPMの生じうる発話である場合、アクセント核によるピークとBPMによるピークとを区別することができない。また、そもそも、F0値の取得には常に誤抽出の問題が付きまとう。ここでは、このような問題があることは把握した上で、与えられた課題を実行するスクリプトを作成することに集中する。

```
    wordTierNum = i
  elsif curTierName$ == "AP"
    apTierNum = i
  elsif curTierName$ == ptcTierName$
    ptcTierNum = i
  endif
endfor

# 取得した層番号のチェック
if wordTierNum == 0
  exitScript: "Word Tier not found in ", tgFile$
endif
if apTierNum == 0
  exitScript: "AP Tier not found in ", tgFile$
endif
if ptcTierNum <> 0
  exitScript: ptcTierName$, "Tier already exists in ",
      tgFile$
endif

# 新規層挿入
ptcTierNum = n+1
Insert point tier: ptcTierNum, ptcTierName$

# 有核語を含むAP区間のF0最大値の取得・TextGridへの記録
selectObject: tgID
n = Get number of intervals: apTierNum
for i to n
  selectObject: tgID
  apLabel$ = Get label of interval: apTierNum, i
  if apLabel$ <> "#"

    # APの開始・終了時刻取得
    apStime = Get start time of interval: apTierNum, i
    apEtime = Get end time of interval:   apTierNum, i

    # APを構成するWordの先頭・末尾のラベル番号を取得
    wordSindex = Get high interval at time: wordTierNum,
        apStime
    wordEindex = Get low interval at time: wordTierNum,
        apEtime

    # AP区間に有核語があればflagACCを1に
```

```
66      flagACC = 0
67      for j from wordSindex to wordEindex
68        wordLabel$ = Get label of interval: wordTierNum, j
69        if index(wordLabel$, "'") <> 0
70          flagACC = 1
71          goto CHECKACC
72        endif
73      endfor
74      label CHECKACC
75
76      # APが有核語を含んでいれば…
77      if flagACC == 1
78
79        # Pitch Objを参照し、AP区間のF0最大値とその時刻を取得
80        selectObject: ptcID
81        maxPitch = Get maximum: apStime, apEtime, "Hertz",
              "parabolic"
82        maxTime  = Get time of maximum: apStime, apEtime,
              "Hertz", "parabolic"
83
84        # 値をうまく取得できなければ…
85        if maxPitch == undefined
86          maxPitch = 0
87          maxTime  = (apEtime-apStime)/2+apStime
88        endif
89
90        # TextGridの新規層に書き込み
91        selectObject: tgID
92        nocheck Insert point: ptcTierNum, maxTime, fixed$
              (maxPitch,3)
93      endif
94    endif
95  endfor
96
97  # TextGrid上書き保存
98  selectObject: tgID
99  Save as text file: tgFile$
```

スクリプトは大きく、前半（1–47行目）と後半（49–99行目）に分けられる。前半はいわば準備作業（オブジェクト生成、必要な値の取得、エラーチェック、新規層挿入）に当たり、メインの処理は後半で行っている。以下では、上から順番にコードを部分的に引用しながら解説していく。

```
5  form maxpitchap
6    infile  tgFile   /Users/ken/CEJC/xjtobi/C001_001_IC01-
         xjtobi.TextGrid
7    infile ptcFile /Users/ken/CEJC/wav/C001_001_IC01.Pitch
8  endform
```

ここではformを使い、TextGridファイルと活用事例1で生成したPitchファイルをユーザーが指定できるようにしている。

```
10 # 新規層の名前
11 ptcTierName$ = "PitchMaxAcc"
```

新規層の名前を変数に代入する。この層名はスクリプト内で複数回使われるため、変数として扱う。これにより、あとで層名を変更したい場合も、この1か所を書き換えるだけで済む。

```
13 # ファイル読み込み
14 ptcID  = Read from file: ptcFile$
15 tgID   = Read from file: tgFile$
```

メニューコマンド「Read from file」を使い、formで指定されたPitchファイルとTextGridファイルを読み込む。同時に、代入演算子「=」を用いて、それぞれのオブジェクトIDを変数ptcIDとtgIDに代入する。これらの変数は後の処理で利用する。

17–32行目では、TextGridオブジェクトのWord（単語）層、AP（アクセント句）層、および新規層（F0最大値の時刻と値を記録する層）の層番号を、層名に基づいて取得する。Praatでは、層の指定を層名ではなく層番号で行うため、メインの処理に入る前にここで取得しておく。なお、TextGridの層番号（層の順番）は作業中に変更されることがある。そのため、スクリプト内で番号を固

定せず、多少面倒でもこのように層名から層番号を求めて変数に保持しておく方法が望ましい。

ここでの具体的な処理は次の通りである。順を追って説明する。

```
17 # 関連する層番号の取得
18 wordTierNum = 0
19 apTierNum = 0
20 ptcTierNum = 0
```

まず、Word層・AP層・新規層の層番号を入れるための変数を用意し、初期値として0を代入する。直後の処理で該当する層が見つかれば、これらの変数は層番号（1以上の値）に書き換えられる。逆に、最後まで0のままなら、その層はTextGridに存在しなかったことを意味する。

```
21 selectObject: tgID
22 n = Get number of tiers
23 for i to n
24   curTierName$ = Get tier name: i
25   if curTierName$  == "Word"
26     wordTierNum = i
27   elsif curTierName$ == "AP"
28     apTierNum = i
29   elsif curTierName$ == ptcTierName$
30     ptcTierNum = i
31   endif
32 endfor
```

このfor文では、TextGridオブジェクトの各層（tier）の名前を取得し（24行目）、その名前に応じて層番号を取得している（25–31行目）。直前の「Get number of tiers」で層の数を求めてnに代入し、それをfor文の［終点］とすることで、層の数だけループを回す。なお「Get number of tiers」や24行目の「Get tier name」はTextGridオブジェクトのダイナミックメニュー「Query」の中に用意されているアクションコマンドである。こうしたアクションコマンドを使う際、対象のオブジェクトが選択されていなければならないため、事前にselectObjectコマンドでTextGridを明示的に選択している（21行目）[10]。

for文の中では、「Get tier name」を使って層番号（i）から層名を取得し、その値が「Word」であれば、変数wordTierNumに、「AP」であれば変数

apTierNumに、さらに新規層の名前（ptcTierName$）と一致すれば変数ptcTierNumに、それぞれの層番号（i）を代入する。

```
34 # 取得した層番号のチェック
35 if wordTierNum == 0
36   exitScript: "Word Tier not found in ", tgFile$
37 endif
38 if apTierNum == 0
39   exitScript: "AP Tier not found in ", tgFile$
40 endif
41 if ptcTierNum <> 0
42   exitScript: ptcTierName$, "Tier already exists in ",
         tgFile$
43 endif
```

ここではエラーチェックを行っている。この時点でwordTierNumやapTierNumの値が0のままであったら、それは、TextGrid内に「Word」や「AP」という名前を持つ層が存在しないということを意味する。その場合は処理を続けられないため、exitScript（3.4.5（C）節）を使ってスクリプト処理を終了する。

41行目でも新規層の層番号のチェックをしているが、条件は異なる。ptcTierNumの値が0でない場合、すでに同じ名前の層が存在していることを意味するため、exitScriptを使って処理を終了する。

```
45 # 新規層挿入
46 ptcTierNum = n+1
47 Insert point tier: ptcTierNum, ptcTierName$
```

新規層の番号を、22行目で取得した層の総数+1として設定し、その番号と層名を指定してポイント層を作成する。47行目の「Insert point tier」は、TextGridオブジェクトのダイナミックメニュー「Modify」→「Modify point tier」にあるアクションコマンドであり、層番号と層名を引数として与える。

10| 複数のオブジェクトを扱う状況でアクションコマンドを用いる際には、今どのオブジェクトが選択されているのかを意識しておく必要がある。スクリプトの流れから、意図するオブジェクトが選択されていることが確実であっても、アクションコマンドの直前で明示的にオブジェクトを選択しておくことが推奨される。

以上が前半の処理である。ここまでの処理で、変数ptcID（PitchオブジェクトID）、tgID（TextGridオブジェクトID）、wordTierNum（Word層の番号）、apTierNum（AP層の番号）、ptcTierNum（新規層の番号）の値が取得できた。これらの変数を使ってこの後の処理を行う。

後半（49行目以降）では、課題で示した処理を実際に実行する。内容はやや複雑なので、詳しい説明に進む前に、全体のスクリプトに戻って後半部分のコメントを拾い読みし、大まかな流れをつかむとよい。

```
# 有核語を含むAP区間のF0最大値の取得・TextGridへの記録
selectObject: tgID
n = Get number of intervals: apTierNum
for i to n
  selectObject: tgID
  apLabel$ = Get label of interval: apTierNum, i
  if apLabel$ <> "#"
```

このfor文では、AP層の各ラベルを順に調べる。最後のTextGrid上書き保存（97–99行目）の前までこのfor文が続く。

まず、「Get number of intervals」でAP層のラベル数を取得してnに代入し、その数だけループを回す。for文内では、アクションコマンド「Get label of interval」にAP層の層番号とラベル番号を引数に指定してラベルを取得し、その値が「#」（ポーズ区間）でなければ次の処理に進む。

なお、50行目と53行目のselectObjectは省略しても動作するはずだが、直後のアクションコマンドを確実に実行させるために明示的に記している。

```
    # APの開始・終了時刻取得
    apStime = Get start time of interval: apTierNum, i
    apEtime = Get end time of interval:  apTierNum, i

    # APを構成するWordの先頭・末尾のラベル番号を取得
    wordSindex = Get high interval at time: wordTierNum,
        apStime
    wordEindex = Get low interval at time: wordTierNum,
        apEtime
```

ここでは、APラベルの開始・終了時刻を取得し、その時刻を手がかりに、APラベルを構成するWordの先頭と末尾のラベル番号を取得している。これら

のコマンドは全て、ダイナミックメニュー「Query」→「Query interval tier」に含まれている。

特に62, 63行目の「Get high interval at time」と「Get low interval at time」に注意したい。どちらも層番号と時刻を指定し、その時刻を含むインターバルラベルの番号を出力するアクションコマンドだが、これらは異なる値を出力する。例えば、指定した時刻が、ラベル番号12の終端時刻とラベル番号13の始端時刻と一致する場合、「high interval」では13を、「low interval」では12を出力する。62行目ではAPの開始時刻と一致しAPの先頭を構成するWordラベルの番号を取得したいので「high interval」を、63行目ではAPの終了時刻と一致しAPの末尾を構成するWordラベルの番号を取得したいので「low interval」を用いている。

```
65      # AP区間に有核語があればflagACCを1に
66      flagACC = 0
67      for j from wordSindex to wordEindex
68        wordLabel$ = Get label of interval: wordTierNum, j
69        if index(wordLabel$, "'") <> 0
70          flagACC = 1
71          goto CHECKACC
72        endif
73      endfor
74      label CHECKACC
```

直前の処理で求めた変数wordSindex（AP先頭のWordラベルの番号）からwordEindex（AP末尾のWordラベルの番号）までの範囲に有核語があるかをfor文で調べる。

68行目の「Get label of interval」で取得したWordのラベルを取得し、変数wordLabel$に代入した上で、有核語であるか否かの判定を、69行目のif文で行う。文字列関数のindex（3.6節、表3）を用いてアポストロフ「'」がwordLabel$に含まれているかを確認する。indexはindex([**文字列1**],[**文字列2**])のように2つの引数をとり、[文字列1]において[文字列2]が登場する位置を返す。例えばindex("abc", "b")は2となる。含まれない場合は0を返す。

for文の直前で、値が0の変数flagACCを用意しておき、for文の処理で有核語が見つかったら1を代入し、gotoを用いてfor文を抜ける。

```
76      # APが有核語を含んでいれば…
77      if flagACC == 1
78
79        # Pitch Objを参照し、AP区間のF0最大値とその時刻を取得
80        selectObject: ptcID
81        maxPitch = Get maximum: apStime, apEtime, "Hertz",
              "parabolic"
82        maxTime  = Get time of maximum: apStime, apEtime,
              "Hertz", "parabolic"
```

ここではAPに有核語が含まれていた場合に、その区間のF0最大値とその時刻を取得する。F0最大値と時刻の取得はPitchオブジェクトのダイナミックメニュー「Query」内のアクションコマンド「Get maximum」と「Get time of maximum」を用いる（81, 82行目）。

80行目でselectObject: ptcIDでPitchオブジェクトを選択しているが、この処理は必須である。この選択を忘れるとTextGridオブジェクトが選ばれたままとなり、81行目の処理でエラーが生じる。

```
84        # 値をうまく取得できなければ…
85        if maxPitch == undefined
86          maxPitch = 0
87          maxTime  = (apEtime-apStime)/2+apStime
88        endif
```

直前の処理で取得したmaxPitchとmaxTimeの値を確認する。F0値は常に適切に取得できるとは限らず、Praatでは取得できなかった場合にundefinedという特殊な値になる。値がundefinedだと後の処理に支障が出るため、値を強制的に書き換える。ここでは、maxPitchの値は0、maxTimeの値は当該AP区間の中間の地点としている。

```
90        # TextGridの新規層に書き込み
91        selectObject: tgID
92        nocheck Insert point: ptcTierNum, maxTime, fixed$
              (maxPitch,3)
```

ここでは、取得したF0最大値とその時刻をTextGridの新規層に書き込んでい

る。まず、selectObject: tgIDでTextGridオブジェクトを選択する。直前でPitchオブジェクトが選択されているため、この操作は必須である。その後、「Insert point」を使って、TextGridの新規層にラベルを挿入する。

「Insert point」はTextGridオブジェクトのダイナミックメニュー「Modify」のサブメニュー「Modify point tier」にある。最大F0値はTextGridに表示するには小数点以下が長すぎるため、文字列関数fixed$（3.6節、表3）を使って小数第3位で丸める。

92行目のnocheckは、3.4.5節（C）で説明したように、後続のコマンドでエラーが出ても処理を続けるための指定である。ここで想定されるエラーは、すでに境界（ポイント）がある時刻に新たな境界を挿入しようとした場合である。この事例では、隣接する2つのAPの境界位置がそれぞれのAP区間のF0最大値だった場合にこのエラーが発生する。本スクリプトでは無視して差し支えないため、nocheckを付けている。

```
97 # TextGrid上書き保存
98 selectObject: tgID
99 Save as text file: tgFile$
```

selectObject: tgIDでTextGridオブジェクトを選択した上で、「Save as text file」によりTextGridを上書き保存する。

これで一連の操作は終了である。

2.4節で紹介した方法などにより、アクセント句区間の最大F0値を人手で確認するのは大きな労力を要する。本節冒頭の脚注9でも言及したように、有核語を含むアクセント句のF0最大値とその時刻が、正確にアクセント核の情報をとらえているとは言えない。しかし、図10のような情報が得られれば、人手での確認作業もかなりはかどるだろう。

このようなスクリプトを書くのは難しく感じられるかもしれない。しかし、個々の要素は活用事例1（4.1節）で紹介したように、オブジェクトウインドウで手動操作を行い、その履歴を活用することで作成できる。まずは小さな手順を確実に積み重ねていくことが重要である。

□5　おわりに

音声分析ソフトウェアPraatを活用した分析の一端を示した。

Praatについてより詳しく知るには、まずPraatのHelp（英文）を参照すべきで

ある。「Help」メニューは、Praat関連のどのウインドウにも用意されている。オブジェクトウインドウの「Help」メニューには、いくつかのtutorialが含まれており、役に立つ。また、どのHelpページにも検索ボックスが備わっており、検索ワード（例えば「mathematical functions」「predefined variables」）を入力し、Searchボタンを押すと、関連するページのリストが表示される。そこからリンクをたどっていけば、必要な情報を得ることができる。日本語の文献では、北原ほか（2017）が、幅広いトピックについて非常にわかりやすく書かれており、勧められる。

参考文献

五十嵐陽介（2015）「韻律情報」小磯花絵（編）『話し言葉コーパス―設計と構築―』81–100, 朝倉書店.

五十嵐陽介・菊池英明・前川喜久雄（2006）「韻律情報」『日本語話し言葉コーパスの構築法（国立国語研究所報告124）』347–453. http://dx.doi.org/10.15084/00001357

北原真冬・田嶋圭一・田中邦佳（編）（2017）『音声学を学ぶ人のためのPraat入門』ひつじ書房.

小磯花絵・菊池英明・山田高明（2020）「『日本語日常会話コーパス』への韻律ラベリング―ラベリングの設計と日常会話の韻律の特徴―」『人工知能学会研究会資料』SIG-SLUD-B903-06, 34–39. http://dx.doi.org/10.11517/jsaislud.88.0_08

関連URL

Praat公式サイト
https://praat.org/

（西川賢哉）