2019.6.25
山本

scoringパッケージ

1　scoringパッケージの概要

(このreadme.mdをtexでpdfにしたファイルがtest/scoring_manual.pdfにある)
採点作業の流れの中で以下のような作業をscoringパッケージはサポートする。

1. マークシートの採点
   - マークシートの読取結果(csvファイル)を採点し、学籍番号と点数のcsvファイルを作成する
2. 統合
   - マークシート以外の成績情報を統合する(例えば、宿題や筆記問題の採点結果などとの統合)
3. 分析
   - 以下のような統計データを出力できるので、それを見て分析する
   - 各問題の正解割合, 評語(A+ABCD)の割合, 点数のヒストグラム, 基本統計, 学類別基本統計
4. 点数調整
   - 点数調整を行って配点を調整する
5. Twinsアップロード用ファイルの作成
   - 点数調整を行った最終結果をtwinsのアップロード用ファイルに統合する。

2　install

installは以下の2つのファイルを正しい場所に配置すればよい。

- src/scoring.py: pythonのプログラム(モジュール)なので、pythonのライブラリパスが通っているところにコピーする。
- bin/score: scoringパッケージを呼び出すbashシェルなので、PATHが通っているところにコピーする。scoringパッケージのすべての機能はこのscoreコマンドを通して提供される。

make installで配置したい場合はMakefileの中の以下の2つの変数を正しくセットしてから、make installする。

```
pylib = scoring.pyを入れるディレクトリ(普通は$PYTHONPATH)
bindir = scoreコマンドを入れるディレクトリ($PATHのどれか)
```

前提条件は以下。

- python3.6以上
- pythonのpandasモジュール

3　scoreコマンドの使い方

scoringパッケージのすべての機能はscoreコマンドを経由して提供される。すべての成績データ(入力・出力・途中結果)は学籍番号を必ず含むcsvファイルで表現される。scoreコマンドは少なくとも1つのcsvファイルを扱うので、1つのcsvファイルは必須引数である。基本的に出力はstdoutにcsvの形式で加工された成績が出力される。また、分析用の情報はstderrに出力される。

```
$ score -h
usage: score [-h] [-marksheet ref desired_pscore] [-crate]
             [-join csvfile2] [-allmerge csvfile2 [csvfile2 ...]]
             [-twins] [-adjust x y xmax] [-interval min max]
             [-distribution] [-abcd] [-statistics] [-gakuruistat gakurui-filename]
             [-nostdout] [-output filename]
             csvfile
```

オプション名は区別が可能な限り短縮してよい。-adjustと-abcd以外はすべて1文字で区別できる。
必須引数であるcsvfilenameは次のように3種類に解釈される。

- -marksheetオプションが指定された場合
  - マークシートの読取結果(学生の解答ファイル)とみなされる
    - headerなし
    - 1列目が学籍番号(下7桁), 2列目以降はマークシートのマーク番号
- -twinsオプションが指定された場合
  - twinsアップロード用csvファイルと見なされる
    - headerあり(ただし、scoreコマンドは読み込み時に無視する: shift-jisのため)
    - 列は左から'科目番号','学籍番号', '学期区分','学期評価', '総合評価'
    - twinsからダウンロードしたままのファイルでOK
  - 上記の-marksheetオプションと排他的である
- 上記以外
  - 採点途中結果のcsvファイル
    - headerなし
    - 1列目が学籍番号(9桁), 2列目が点数

scoreコマンドの出力はtwinsアップロードファイルへのjoinの場合を除いて、すべて「採点途中結果のcsvファイル(headerなしの1列目が学籍番号(9桁),2列目が点数)」である。
以下では、作業の種別毎にオプションを説明する。

3.1 マークシート採点

3.1.1 score csvfilename -marksheet ref-filename desired-pscore ...

csvfilenameをマークシート読み取り結果として読み込み、採点を行う。マークシートの正解を定義したファイル(ref-filename)を引数として与える。また、マークシートの満点(desired-pscore)を与える。正解定義中の各問題のweightにしたがって各問題に配点する(配点は整数なので要求した満点と少し違うこともありえる)。-twins以外の全てのオプションを同時に使用可能。チェック用として各学生・各問題ごとの正誤(正=1, 誤=0)のcsvファイルを以下の名称で保存する。

- csvfilename.marubatu

マークシート採点のための正解定義は5.2節参照。学籍番号はすべて頭に20が付けられるので(200000000を足す)、必ず下7桁になっている必要がある。

3.1.2 -crate

-marksheetオプションが指定されているときだけ有効なオプションで、各小問毎の正解率、配点、問題タイプをstderrに出力する。

3.2 統合

3.2.1 score csvfilename -join csvfilename2 ...

必須ファイルcsvfilenameで指定されたデータ[*1]にcsvfilename2で指定されるデータを統合する。統合は、keyとして学籍番号を用い、pandasのmerge関数を用いる。統合(join)方法は以下の2種類。

- -twinsオプションが指定されている場合:
  - twinsアップロードファイルへleft-joinを行う。
    - twinsのファイルにない学生は取り除かれる
    - 取り除く学生の学籍番号はstderrに出力される
- -twinsオプションがない場合:
  - outer-joinを行う
    - 片方のデータにしかない学生も残る。
    - データがない方の学生の点数は0点

-twinsオプションがある場合の出力はtwinsへのアップロード用のファイルとして出力される。

- headerがshift-jisでないといけないので-outputオプションでファイルを指定する。
  - 標準出力に出すとなぜかshift-jisにならない☹
- twinsデータの'総合評価'に数値が入っている場合はそれとcsvfilename2の点数の合計が'総合評価'になる
- twinsデータの'総合評価'が空の場合はcsvfilename2の点数が'総合評価'になる

-joinオプションの引数であるcsvfilename2のファイルの仕様は以下。

- headerなし
- 1列目: 学籍番号
- 2列目以降(いくつでもOK): 評価点
  - 2列目以降の点数は合計される
  - 統合後にcsvfilenameとcsvfilename2の点数は合計される

他のあらゆるオプションを同時使用可能。

3.3 点数調整

点数調整は、マークシート採点後または統合後、あるいはどちらも指定されてない場合はcsvfilenameで与えられた処理途中のデータに対して適用される。

3.3.1 -adjust x y xmax

素点xをyに線型に持ち上げる。xmax以上は素点のまま。図1を参照。

3.3.2 -interval min max

点数の最低点をmin、最高点をmaxにする。

3.4 分析

分析は点数調整後の成績に対して行われる(すなわち出力ファイルに対する分析である)。以下のような種類(オプション名)があり、同時に指定できる。出力はすべてstderrに出力される。

- -distribution: 点数分布のヒストグラム出力
- -abcd: 標語(A+, A, B, C, D)の人数分布
- -statistics: 平均, 標準偏差, 最高・最低点, 4分位点

[*1] -marksheetオプションが指定されている場合は、採点結果に対して適用される。

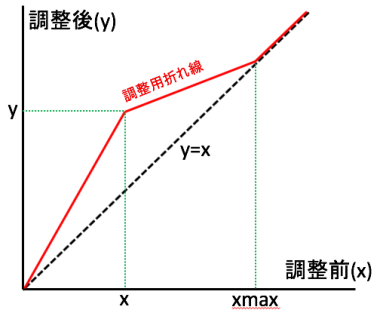

図1　-adjust x y xmaxの点数調整

- -gakuruistat gakurui-filename: 学類毎のstatistics
  - 学類毎の学籍番号を定義したファイルを引数として与える
  - 定義ファイルの詳細は5.3節参照。

あらゆる他のオプションと同時併用できる。最終結果(すなわち出力データ)に対する分析を行う。先に行われるのは、マークシート採点、統合、点数調整である。いずれも指定されていない場合は、入力としてのcsvfilenameがそのまま分析対象となる。-twinsオプションが指定されている場合は、twinsへのアップロードファイルの分析が行われる。

点数調整と分析は同時に指定しながら調整するのが普通の使い方と思われる。

3.5　その他

その他のオプションは以下。

- -nostderr: 標準出力に結果のcsvファイルを出力しない
  - 調整時にこれを指定することが多い
- -output filename: 結果を標準出力ではなくファイルに出力する
  - twinsアップロード用ファイルはこのオプションで出力しないとheaderが正しくshift-jisで出力されない
- -allmerge csvfile2+: 採点途中結果ファイルの合算をしない統合
  - 複数のファイルを引数に取れる。
  - 必須引数のcsvfileに複数のcsvfile2をouter-mergeし、点数の合算は行わない。
  - csvfileとcsvfile2はどちらも必ず1列目が学籍番号でなければならない(csvfileにtwinsファイルは指定できない)。その他の列は文字列でもなんでもかまわない。
  - その他のすべてのオプションが無視される。
  - 出来上がったファイルをshift-jisに変換してexcelで読んで列名を入力して印刷して保存☺

4　使用例

4.1　使用例1

ある科目の使用例。この科目では期末試験でマークシート問題と記述問題の2種類の解答をしてもらった。また、宿題の点数が別に定義されている。

1. ファイルの準備
   - マークシートの解答を~data/answer.csvに用意
   - 記述問題の採点結果と宿題の点数を~data/sup.csvに用意
   - twinsからアップロード用のファイルを~data/upload.csvに用意
2. マークシートの正解と学類別の学籍番号を定義したファイルを作成する

- 正解(と配点重み): reference.py
- 学類学籍番号: gakurui_id.py
- どちらもpythonのlist形式の記述なので拡張子をpyとしているが、気持ち悪ければtxtでもOK

3. マークシートの採点結果を点数化する関数totalscoreを定義。
   - 今回は小問1問につき3点
4. マークシートの採点
   - 小問の正解率と満点を80としたときの配点を保存
     - $ score ~data/answer.csv -m reference.py 80 -c -n &> crate.txt
   - マークシートだけの得点状況を分析
     - $ score ~data/answer.csv -m reference.py 80 -s -d -abcd -n
     - 分析結果を残したい場合は最後に`&> file.txt`を付ける
   - (マークシートだけの得点を少し調整する場合)
     - $ score ~data/answer.csv -m reference.py 80 -d -adjust x y xmax -i min max -n
     - あるいは、正解の配点重みを変更してもよい
   - 結果を書き出す
     - $ score ~data/answer.csv -m reference.py -adjust x y xmax > mksheet.csv
5. data/sup.csvを統合
   - 統合する
     - $ score mksheet.csv -j ~data/sup.csv > mksheet_sup.csv
   - 得点調整を行う
     - $ score mksheet_sup.csv -abcd -d -adjust x y xmax -n
     - (最低・最高点が[0,100]をはみ出す場合は`-i 0 100`を加える)
   - 結果を書き出す
     - $ score mksheet_sup.csv -abcd -d -adjust x y xmax > mksheet_sup_adjust.csv
   - (上記の3つは同時に行うこともできる)
     - $ score mksheet.csv -j ~data/sup.csv -abcd -d -adjust x y xmax > mksheet_sup_adjust.csv
6. twinsのアップロードファイルを作成
   - $ score ~data/upload.csv -t -j mksheet_sup_adjust.csv -d > twins_upload.csv
   - (最低・最高点が[0,100]をはみ出す場合は`-i 0 100`を加える)
7. 特別処理が必要な学生に対してtwins_upload.csvを直接修正
8. 最終結果に対する分析結果を保存
   - $ score twins_upload.csv -t -s -g gakurui_id.py -d -abcd -n &> result_analysis.txt
9. 氏名、学籍番号と採点途中結果を履歴として残す
   - $ score meibo.csv -allmerge mksheet.csv ~data/sup.csv mksheet_sup.csv mksheet_sup_adjust.csva > matome.csv
   - `meibo.csv`は名簿から作成する。1列目が学籍番号で2列目が氏名。utf8ファイル。

4.2 使用例2

`test`ディレクトリ以下に簡単なデータが準備してある。準備されているデータは例として参考にしてください。`test`ディクレトリで`auto.sh`を起動すると使用例1の簡単化された手続きが実行される。結果の例が`test/test-ref`ディレクトリ以下にあるので正しく動作しているかを確認のためにも使える。

5　付録

5.1　準備作業

必要な準備作業を以下にまとめる。

5.1.1　マークシート読み取り

学類所有のマークシート読み取り装置を使って読み取る。前提としているマークシートは以下。

- 教育ソフトウェア社の「総合カード053」(型番: C053)
  - 学生番号欄: 7桁(なので、学籍番号下7桁をマークさせる)
  - 解答欄: 「1–9,0」の0–9の数をマークできる列が50列
    - 上から1,2,3,…,9,0の順

学籍番号のマークをミスる学生がいるので手で修正(2019年度のある科目では、320人中2人の学生がおかしな番号を入れていた)。解答欄は修正しない。結果として、1列目が学籍番号(下7桁)、2列目以降がマークシートの各列の解答番号となっているcsvファイルができればよい。

■

マークシート読取手順　以下のようにしてマークシートの解答をcsvファイルにする。

- 学類のマークシート読取装置と専用ノートパソコンを接続・起動
- 「MarkView」というソフトを起動
- 「シート読取り」を選択
- レイアウト切り替え(総合053とかなんとかのレイアウトにする)
- ファイル→読取りデータ保存先の変更
  - これでファイルが初期化されるみたい
- 読取ボタン?を押す(累積される)
  - 明示的なsave命令はない。自動で保存される。
- ソフトを終了

5.1.2　履修者名簿

成績アップロード用の履修者名簿をtwinsからダウンロード。処理の最後にこのファイルに成績を統合する。headerがshift-jisであるが、そのままでよい(scoringパッケージはheaderを読み飛ばす)。アップロードファイルを出力する場合はshift-jisのheaderをscoreコマンドが付ける。

5.1.3　正解の定義

マークシート読み取り結果に対する正解を定義するファイルを作成する必要がある。以下の種類の解答に対応。

- S: 選択肢から1つ選択
- SS: 選択肢から1つ選択の連続
- MS: 選択肢から複数選択(順不同)
- Num: 数値。複数桁もOK

それぞれの種類について、マークシートの位置と正解を定義。配点重みは指定しなければdefaultの100(%)が各小問に設定される。配点重みを調整したい場合は100(%)を基準に上下する割合を指定する。詳しくは5.2節を参照。

5.1.4　学類学生番号情報

学類毎の成績統計を出す場合は、学類毎の学籍番号を定義したファイルが必要。学籍番号の範囲指定と個別指定ができる。詳しくは5.3節参照。

5.2　マークシートの正解定義

マークシート問題の正解定義はpythonのlist形式のファイルを作成し、ファイル名を指定する。リストの要素が各問題の定義である。各問題は次のような4種類である。以下の「列番号」とはマークシートの列の位置である(1から50)。

- S: 選択問題
  - Format: [S, 列番号, 正解 (, 配点重み)]
- SS: 選択問題の連続
  - Format: [S, [列番号start, 列番号end], [正解1, 正解2, …] (, 配点重み)]
    - 列番号startから列番号endまでの連続した列番号に対して、正解1,2,…を正解とする。
  - 各列番号は独立の小問と見なされる
    - 配点重みは各小問にそのまま適用される(小問数で割ったりしない)
- MS: 複数選択問題
  - Format: [MS, [列番号1, 列番号2, …], [正解1, 正解2, …] (, 配点重み)]
    - 正解は順不同
    - 解答に同じ答えがある場合は1つしか正解としない
  - **正解**の数だけ小問があるとみなす
    - 配点重みは各小問についてそのまま適用される(小問数で割ったりしない)
  - 列番号の数と正解の数は一致しなくてもよい(**正解**の数だけ小問が定義される)。
    - 選択する正解の数を指定しない場合(多めに列番号を指定しておく)
    - 正解の方が列数より多い場合もありえる
- Num: 数値問題
  - Format: [Num, [列番号1, 列番号2, …], [正解1, 正解2, …] (, 配点重み)]
    - 正解の順番も一致しないと誤りと判断
  - 数値全体がすべて一致してこの小問1問正解とみなす

「配点重み」は省略されている場合100となり、100でない場合は100を基準とした割合で指定する。

例えば以下の通り。4つ目のMS問題だけ各小問を配点重み50の割合いとしている。他の(小)問の配点重みは指定していないのでdefaultの100となる。実際の配点は指定した満点に応じた割合で決定される(整数化するため指定した満点と少し誤差が出ることがある)。

正解定義ファイルの例

```
[
    [S, 1, 1],
    [S, 2, 3],
    [S, 3, 6],
    [MS, [4,5,6], [2,4,5], 50],
    [Num, [7,8,9,10], [1,0,4,0]],
    [Num, [11,12,13], [5,4,0]]
]
```

5.3　学類学籍番号情報

学類ごとの学籍番号の定義は、pythonのlist形式のファイルを作成し、それを読み込ませる必要がある。リストの各要素が各学類の学籍番号情報である。以下で定義される。

学類学籍番号の定義ファイル = [学類情報1, 学類情報2, ...]
    学類情報i = [学類名の文字列, 範囲リスト, 個別リスト]
        範囲リスト = [範囲1, 範囲2, ...]
            範囲i = [start, end] : startからendまでの学籍番号
        個別リスト = [学籍番号1, 学籍番号2, ...]
            学籍番号i = 学籍番号を数値で

例えば以下の通り。

学類学籍番号情報定義

```
[
    ['人文学類', [[210010250, 210010350], [209910111]],
    ['社会工学類', [[210022222, 210022333]], []],
    ['情報科学類', [[210044444, 210044555], [210055555, 2100555560]], []],
    ['情報メディア創成学類', [[210066000, 210066055]], [209911111, 209822222]],
    ['知識情報・図書館学類', [[210077000, 210077100], [220088888, 220088999]], []],
]
```

以上