S.T.A.P.L.E.

Standardised Testing Against Progressive Learning Environments

Author: R. E. Treharne

📘 Background

S.T.A.P.L.E. is a purpose-built system developed by the School of Biosciences' Technology Enhanced Learning (TEL) Team at the University of Liverpool. It provides a robust, fully digital workflow for processing scanned paper-based multiple-choice assessments.

This system was developed out of necessity. At the time of writing, only two OMR scanners remain operational across the entire University — a result of the widespread assumption that paper-based MCQ exams would no longer be needed. With growing demand to share these scanners across multiple departments, the lead time for obtaining results has become increasingly impractical.

S.T.A.P.L.E. offers an effective contingency plan should those remaining scanners become unavailable — a matter of *when*, not *if*. It eliminates dependency on legacy hardware by leveraging modern image processing techniques to detect student IDs, interpret answer sheets, and generate detailed performance reports.

⚙️ Installation

Follow these steps to install and run the S.T.A.P.L.E. system on your machine.

1. Clone the Repository

```
git clone https://github.com/rtreharne/omr_liverpool.git staple
cd staple
```

2. Create a Virtual Environment

   - On **Linux/macOS**:

     ```
     python3 -m venv venv
     source venv/bin/activate
     ```

   - On **Windows**:

     ```
     python -m venv venv
     venv\Scripts\activate
     ```

> 💡 **Tip for Windows users**: Install Git for Windows to get **Git Bash**, a terminal that supports Unix-style commands like `ls`, `source`, and `cd`. After installing, right-click in your project folder and select **"Git Bash Here"**.

3. Install Python Dependencies

```
pip install --upgrade pip
pip install -r requirements.txt
```

4. Install System Dependencies

   Some libraries require system-level dependencies. Install them as follows:

   - On **Ubuntu/Debian**:

     ```
     sudo apt update
     sudo apt install -y \
       build-essential \
       poppler-utils \
       tesseract-ocr \
       libglib2.0-0 \
       libsm6 \
       libxext6 \
       libxrender1 \
       libmagic1 \
       libpango1.0-0 \
     ```

```
        libcairo2 \
        libopencv-dev
```

- On **macOS** (using Homebrew):

```
brew install poppler tesseract cairo pango opencv
```

- On **Windows**:

a. **Tesseract OCR**

- Download from: https://github.com/tesseract-ocr/tesseract
- Ensure the install path (e.g. `C:\Program Files\Tesseract-OCR`) is added to your system `PATH`.

b. **Poppler**

- Download from: http://blog.alivate.com.au/poppler-windows/
- Extract the ZIP file and add the `bin/` directory to your system `PATH`.

c. **GTK / Cairo / Pango (optional)**

- Usually bundled via Python packages (`cairocffi`, `reportlab`)
- If needed, install GTK+ for Windows and add its `bin/` folder to your system `PATH`.

5. Verify the Setup

   Run this test command:

```
python -c "import cv2; import easyocr; import pytesseract; print(
```

   If no errors appear, the environment is ready.

Need help? Contact **Dr. R. E. Treharne** at **R.Treharne@liverpool.ac.uk**

📄 Preparation

Before using the S.T.A.P.L.E. system, ensure that your paper answer sheets are prepared and scanned correctly.

1. **Use the University of Liverpool Speedwell Format**

All answer sheets must follow the University of Liverpool's official Speedwell layout for compatibility with this system.

2. **Scan Using a University Photocopier**

   - Use the **"Scan to Me"** functionality available on any University of Liverpool multifunction printer (MFP).
   - Scans **must** be in:
     - **Colour**
     - **Portrait orientation**
     - **300 DPI resolution**
     - **PDF format**

   > ⚠️ Scans that do not meet these specifications may not be processed accurately.

   - **Ensure that your answer sheet is the first page to be scanned.**

3. **Organise Scanned Files**

   - After receiving the scans by email, download them to your computer.
   - Place all PDF files for a given batch into a single folder.
   - Name the folder appropriately (e.g. `BIOS101`, `ANAT204`, etc.) to reflect the module or assessment.

You are now ready to begin processing your scans using S.T.A.P.L.E.

🛠️ PDF Processing (`process_pdf.py`)

This script prepares scanned assessment files for analysis by merging multiple PDFs into one and converting that merged file into high-resolution PNG images.

1. **What it Does**

   The script performs two key steps:

   - **Merge PDFs**: Combines all `.pdf` files in a given folder into a single PDF called `single.pdf`.
   - **Convert to PNGs**: Converts each page of the merged PDF into a `.png` image at 300 DPI resolution.

2. **How to Use**

Run the script from your terminal:

```
python process_pdf.py
```

You will be prompted to:

- Enter the path to the folder containing your `.pdf` scan files.
- Enter the path to the folder where the output PNG images should be saved.

3. **Example Workflow**

   Suppose you have a folder called `BIOS101` that contains several scanned PDF files. Run:

   ```
   python process_pdf.py
   ```

   - For the input folder, enter: `BIOS101`
   - For the output image folder, enter: `BIOS101/images`

   The script will:

   - Create a file: `BIOS101/single.pdf`
   - Generate PNGs like `page_001.png`, `page_002.png`, … inside `BIOS101/images/`

4. **Technical Notes**

   - Uses the `PyMuPDF` (`fitz`) library for both PDF merging and rasterizing pages.
   - Zoom is calculated to ensure output images are 300 DPI.
   - Output filenames are zero-padded (`page_001.png`, `page_002.png`, etc.) for easy sorting.
   - Only `.pdf` files in the top-level of the input folder are processed.

You must complete this step before proceeding to answer extraction.

🧪 Answer Detection and Student ID

Extraction (`detect_answers.py`)

This script performs the core Optical Mark Recognition (OMR) task for the S.T.A.P.L.E. system. It extracts student answers from scanned PNG images and automatically reads student IDs from a 9×10 grid of bubbles.

1. **What It Does**

- Detects and warps the red ROI box from each PNG image.
- Extracts multiple-choice answers using calibrated bubble positions.
- Reads the student ID from a 9×10 grid using vertical alignment boxes.
- Saves annotated images, extracted answers, and student IDs to CSV.

2. **How to Use**

Run the script in your terminal:

```
python detect_answers.py
```

You will be prompted to:

- Enter the path to the folder containing PNG files (output from `process_pdf.py`).

The script will:

- Load or prompt for **bubble calibration** and **minimum ROI size**.
- Extract answers from each image.
- Read student ID digits using a grid aligned by right-edge markers.
- Generate annotated output and save results.

3. **What It Produces**

Inside the input folder, the following files/folders will be created:

- `bubble_coords.csv` : Coordinates of each bubble (saved during calibration).
- `min_roi_size.txt` : Minimum acceptable red box dimensions.
- `annotated/` : Folder containing annotated PNGs with detected answers.
- `all_detected_answers.csv` : A table of filenames and selected answers, with student ID appended.
- `file_student_id.csv` : A mapping of image filenames to extracted student IDs.

4. **Calibration Steps (Interactive)**

The first time the script is run for a folder, it will guide you through two setup steps:

- **Step 1**: Click the top-left and bottom-right of a typical red ROI box (to set minimum width/height).
- **Step 2**: Click the first and last bubbles of each 5×5 grid to interpolate coordinates. Only click 5x5 grids that have responses. For example, if there are only 32 questions on the test you will need to select the first and last bubbles of the first 7 5x5 grids.

These steps ensure accurate bubble alignment across all scanned sheets.

5. **Technical Notes**

- Uses OpenCV for image processing and Matplotlib for interactive point selection.
- Bubble fill intensity is measured using a fixed window (`half_box`), derived from calibration.
- Student IDs are extracted from a 9×10 grid based on black/grey alignment marks on the right margin.
- The ID is composed by selecting the darkest bubble in each of 9 columns.

Make sure this script is run **after** converting PDFs to PNGs using `process_pdf.py` .

6. **Sample Output (`all_detected_answers.csv`)**

Below is a snippet of what the output CSV file looks like after running the script:

```
filename,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,
answers_____,B,D,B,B,C,C,A,C,B,D,C,B,B,C,A,B,A,C,C,A,C,A,B,B,D,B,A,D,
page_002.png,C,D,B,C,D,C,C,B,B,D,B,C,A,D,B,A,B,D,C,D,C,A,B,C,C,D,A,D,
page_003.png,B,C,A,A,C,A,A,C,B,D,D,A,B,C,A,D,D,B,B,A,C,A,B,C,C,B,A,D,
page_004.png,D,B,B,A,C,C,C,C,B,A,C,D,A,D,B,D,A,B,B,D,C,B,C,A,A,B,A,D,
page_005.png,D,D,B,C,D,C,B,C,A,D,C,B,A,C,A,C,D,A,B,A,C,A,B,A,C,C,A,D,
page_006.png,A,C,B,C,C,A,B,C,B,D,C,A,A,D,A,A,D,C,C,D,C,B,C,C,C,C,A,D,
```

Each row represents a student's scanned response, showing their selected options (Q1–Q35) and their extracted student ID.

The filename of the first row has been manually edited to read "answers_____". This is important. You must identify the row corresponding to your answers in this way before proceeding.

🔍 Validating Image Sizes (`validation.py`)

This script is designed to help you **identify scanning issues** by analyzing the dimensions of all PNG images in a folder. Outliers in image area often indicate incorrectly scanned or corrupted files that could disrupt OMR processing.

1. **What It Does**

- Loads all annotated `.png` files in a folder.
- Calculates the image **area**, **width**, and **height** for each file.
- Flags **outliers** based on Z-score deviation from the mean area.
- Saves a log file listing all identified outliers.
- Visualizes the results using a scatterplot with ±2σ threshold lines.

2. **How to Use**

Run the script:

```
python validation.py
```

You will be prompted to:

- Enter the path to the folder containing `.png` images (e.g., `BIOS101/annotated`).

The script will:

- Analyze image sizes.
- Print warnings for any problematic images.
- Save a log file like `image_outliers_20250601_141200.log`.

3. **Example Output**

```
Outlier: page_004.png - 2380x3200 (Area: 7616000)
Outlier: page_019.png - 1980x2800 (Area: 5544000)

Total outliers: 2 out of 35 images
```

4. **Visual Feedback**

A scatterplot is displayed showing:

- Each image's area
- The overall mean
- ±2 standard deviation lines
- Outliers in **red**

This helps you quickly spot inconsistencies in scan resolution or cropping.

5. **When to Use This Script**

- After running answer detection
- Whenever you're seeing unexplained failures or empty outputs from `detect_answers.py`

This script helps prevent data loss by ensuring only properly scanned sheets are passed through the pipeline.

6. **Dealing with Failed Extractions**

The STAPLE extraction system is highly effective, but not foolproof. In practice, approximately **2% of all scanned images fail** during the automated processing stage due to issues such as:

- Misalignment or partial scans
- Excessive noise or low contrast
- Incorrect bubble filling or marks outside detection zones

> For every 100 answer sheets, it is reasonable to expect **manual review and entry for around 2 sheets**.

Manual Correction Procedure

1. Review the `annotated/` folder and check for missing or obviously incorrect annotations.
2. Cross-reference those files in `all_detected_answers.csv`.
3. Manually open the failed image, interpret the student's selected answers, and edit the corresponding row in the CSV file.
4. Also update the `student_id` if it was incorrectly extracted or missing.

Once all rows in `all_detected_answers.csv` are complete and accurate, you may proceed to scoring and analysis.

> 💡 Use spreadsheet software like Excel, Numbers, or Google Sheets to simplify editing the CSV file.

🧾 Scoring and Enrichment (`process_answers.py`)

This script performs automatic scoring of student multiple-choice responses and allows optional enrichment of the results using a Canvas-exported student gradebook. It also supports manual resolution of unknown or mismatched student IDs.

1. **Before You Begin**

You must first **export the gradebook from Canvas** for the module in question:

- Go to the Canvas module.
- Navigate to **Grades** → **Actions** → **Export** → **CSV**.
- Save the exported file (e.g., `grades.csv`) in the **same folder** as your scanned answer data (where `all_detected_answers.csv` exists).

2. **What It Does**

- Reads `all_detected_answers.csv` and extracts the answer key + student responses.
- Scores each student's answers.
- Optionally enriches results using student names and IDs from Canvas.
- Saves the output to `scored_answers.csv`.
- Allows manual verification and correction of unmatched or unknown student IDs.

3. **How to Use**

Run the script:

```
python process_answers.py
```

You will be prompted to:

- Enter the path to the folder containing `all_detected_answers.csv`.
- Enter the number of questions to score (e.g. 32).
- Choose whether to enrich with student names (type `y` or `n`).
- If enriching, enter the filename of the Canvas export (e.g. `grades.csv`).

4. **Output Files**

- `scored_answers.csv`: Contains all scores with student name and ID (if enrichment succeeds).
- Rows include: `filename`, `score`, `percentage_score`, `student_id`, and optionally `student_name`, `SIS User ID`, `ID`.

5. **Handling Unknown Students**

If any rows cannot be matched with a student from the Canvas export:

- The script displays the top portion of the student's scanned sheet.
- Attempts to auto-suggest a close match based on digit similarity.
- You can:
  - Accept the suggestion (`y`)
  - Manually enter student name and ID (`m`)

All edits are saved live to the CSV file.

6. **Example Output (Partial)**

```
filename,student_name,score,percentage_score,student_id,sis_user_id,I
page_001.png,Alice
Smith,30,93.8,201805066,201805066@student.liv.ac.uk,10566
page_002.png,Unknown,27,84.4,201809558,,
```

```
page_003.png,Charlie
Jones,31,96.9,201804043,201804043@student.liv.ac.uk,10443
```

7. **Best Practices**

- Run this script **after** validating and completing `all_detected_answers.csv`.
- Ensure your Canvas export includes the correct columns: `Student`, `SIS User ID`, and `ID`.
- Only fill in unmatched rows once, then re-run if necessary to finish resolving.

After scoring is complete, you're ready to generate item-level analysis and summary reports.

📊 Student Score Report (`score_report.py`)

This script generates a polished PDF report listing individual student scores based on the final processed and scored data. It is typically used at the end of the STAPLE workflow to produce a deliverable document for instructors or administrators.

1. **What It Does**

- Reads the final `scored_answers.csv` file.
- Extracts `student_id`, `student_name`, and `percentage_score`.
- Sorts students alphabetically.
- Prompts for user input (author, course, assessment).
- Outputs a professionally styled PDF report with logos and a footer.

2. **Before You Run**

Ensure you have:

- A complete and correct `scored_answers.csv` file in your working directory.
- The `logo.svg` file (University logo) and `staple.png` (STAPLE system logo) in the same directory.

If `logo.svg` is not already converted, the script will generate a PNG version (`logo_converted.png`) using `cairosvg`.

3. **How to Use**

```
python score_report.py
```

You will be prompted to enter:

- The path to `scored_answers.csv`
- Your name (to appear as the author of the report)
- The course name (e.g. BIOS101)
- The assessment name (e.g. Midterm MCQ)

The script will generate a PDF report saved to:

```
scored_answers_report.pdf
```

4. **Example Output**

The PDF report includes:

- **Title Page** with logos, author, and assessment info
- **Table of student scores**, sorted by name
- **Footer** with STAPLE system contact info and logo

Each page includes the footer and page number automatically.

5. **Output Preview (table structure)**

student_id	student_name	percentage_score
201804043	Charlie Jones	96.9
201805066	Alice Smith	93.8
201809558	Unknown	84.4

This report is suitable for internal moderation, distribution to course teams, or formal record-keeping.

📊 Item Analysis Report (item_analysis.py)

The `item_analysis.py` script generates a detailed PDF report analyzing item-level statistics from a marked multiple-choice question (MCQ) assessment. It uses a reference row (e.g. `answers_____`) to identify correct answers, computes difficulty and discrimination for each question, and includes score summaries and visualizations.

🔧 Inputs Required

On running, the script prompts the user for:

1. Path to a `scored_answers.csv` file (containing one `answers_____` row with correct answers and student responses beneath).
2. Number of questions to analyze (e.g., 32).
3. Author name.
4. Course name.
5. Assessment name.

📊 Key Features

✅ Answer Key Extraction

The correct answers are taken from the first row where `filename` contains "answers" (case-insensitive). All other rows are treated as student responses.

🔍 Item-Level Stats

For each question:

- **Difficulty (p):** Proportion of students answering correctly.
- **Discrimination (r_pb):** Point biserial correlation between student correctness and overall performance.
- **Interpretation labels** are attached to both.

📈 Summary Statistics

Calculated across all student scores:

- Mean, Median, Min, Max (as %).

📉 Score Histogram

A histogram is generated showing distribution of student scores.

📄 PDF Report

The final PDF includes:

- Title page with University and STAPLE branding
- Summary stats
- Histogram of scores
- Item stats table
- Interpretation key for difficulty and discrimination

A footer appears on every page:

> For more information about the S.T.A.P.L.E. system please contact Dr. Robert Treharne (R.Treharne@liverpool.ac.uk)

🗃️ Outputs

- `item_analysis_output.csv` : Cleaned item statistics.
- A PDF file saved in the same folder, named like `item_analysis_output.pdf` .

📌 Notes

- This script expects numeric question column headers (`1` , `2` , ..., `32`) and a column `filename` .
- The correct answers row must contain expected answers in these columns.
- If discrimination cannot be calculated (e.g. no variance), "N/A" is shown.

📤 Example Output

```
=== ITEM ANALYSIS SUMMARY ===
Number of students: 98
```

```
Max score (%): 100.0
Mean score (%): 68.25
...

=== ITEM STATISTICS ===
Question  Difficulty (p)  Difficulty Label  Discrimination (r_pb)
Discrimination Label
       1            0.85              Easy                  0.321
Very Good
       2            0.42          Moderate                  0.142
Weak
...
```

📎 Dependencies

- `pandas`
- `matplotlib`
- `scipy`
- `reportlab`
- `cairosvg`

Ensure `logo.svg` and `staple.png` are in the project root.

🧪 Sample Command

```
python item_analysis.py
# Enter path: /path/to/scored_answers.csv
# Number of questions: 32
# Name: Dr. Jane Doe
# Course: PHYS1001
# Assessment: Midterm A
```