RCV Formats

Armin Samii

Sep 13, 2023

ADDITIONAL DOCUMENTATION:

RCV Formats helps programmers and researchers build tools that analyze the results of a Ranked-Choice Voting election without having to support the many file formats used to report RCV results.

RCV Formats converts data from several sources into a standardized format. It supports both python and command-line tools

Currently supported input formats are:

1. The Universal RCV Tabulator JSON format
2. The Opavote JSON format
3. The ElectionBuddy CSV format
4. The Dominion XLSX format
5. The Dominion TXT format

As well as the Dominion first-round-only XML format (used in Alaska), which contains the first rounds of several elections. All other converters contain the results of just one election per file.

The standardized output format is the Universal RCV Tabulator JSON. To understand this format, look at examples or the jsonschema.

CHAPTER

ONE

DEMO

1.1 Command-line

```
rcvformats convert -i <input-filename> -o <output-filename>
```

1.2 Python

```
from rcvformats.conversions.automatic import AutomaticConverter

standardized_data = AutomaticConverter().convert_to_ut(input_filename)
```

CHAPTER

TWO

INSTALLATION

Install the library via pip:

```
pip3 install rcvformats
```

CHAPTER

THREE

CONVERT TO STANDARDIZED FORMAT

You can convert from any of the supported formats. Use this functionality to support a wide array of input data while only writing code to support a single format.

3.1 Command-line

```
rcvformats convert -i <input-filename> -o <output-filename>
```

The bash script always uses the automatic converter.

3.2 Python

```
from rcvformats.conversions import electionbuddy

converter = electionbuddy.ElectionBuddyConverter()
try:
  converter.convert_to_ut(filename)
except Exception as e:
  print("Errors: ", e)
```

Valid converters are:

```
from rcvformats.converters.automatic import AutomaticConverter
from rcvformats.conversions.dominion_txt import DominionTxtConverter
from rcvformats.conversions.dominion_xlsx import DominionXlsxConverter
from rcvformats.conversions.electionbuddy import ElectionBuddyConverter
from rcvformats.conversions.opavote import OpavoteConverter
```

The AutomaticConverter checks if the file matches any of the available schemas, and if it finds a matching schema, it runs the corresponding conversion (if a conversion is needed at all).

CHAPTER

FOUR

SCHEMA VALIDATION

Validate that your file is supported by RCVFormats.

Validation is only on the structure of the data, not on its contents: it is possible for a validly-formatted file to still contain invalid data.

4.1 Command-line

```
rcvformats validate -i <input-filename> -s <schema-type>
```

Valid schema validators on the command line are 'eb' (for electionbuddy files), `ov10` (for opavote files pre-2022), `ov11` (for opavote files post-2022), `ut` (for universal tabulator files). Dominion does not have a schema validation currently.

4.2 Python

```
from rcvformats.schemas import universaltabulator

schema = universaltabulator.SchemaV0()
is_valid = schema.validate('/path/to/file.json')

if not is_valid:
  print(schema.last_error())
```

Valid schema validators for python are:

```
from rcvformats.schemas.electionbuddy import SchemaV0
from rcvformats.schemas.opavote import SchemaV1_0
from rcvformats.schemas.universaltabulator import SchemaV0
```

4.2.1 Fill in missing transfer data

Transfer data is useful to determine where votes went when a candidate was eliminated, or when a candidate was elected and had surplus votes (in STV).

If you have a file format that does not have transfer data, there are three options: you can leave it out entirely, you can assign transfers proportionally to each eliminated candidate, or you can assign only the transfers that are unambiguous. We recommend the last option, which prepares transfer data for any round that does not involve batch elimination. The second option results in fake data which cannot be relied upon for any results reporting or analyses.

CHAPTER

FIVE

MULTI-CONVERTERS

Call `DominionMultiConverter.explode_to_files(fileObject)`, which will return a dictionary mapping election names to NamedTemporaryFiles.

5.1 Command-line

```
rcvformats transfer -i <input-filename> -o <output-filename>
```

5.2 Python

```
from rcvformats.conversions.ut_without_transfers import UTWithoutTransfersConverter

converter = UTWithoutTransfersConverter()
try:
  converter.convert_to_ut(filename)
except Exception as e:
  print("Errors: ", e)
```

CHAPTER

SIX

UPCOMING PLANS

In addition to data normalization for RCV Summary formats, we would like similar functionality for cast vote records.

CHAPTER

SEVEN

RUNNING TEST SUITE

`pip3 install -r requirements-test.txt`, then run `pytest rcvformats/test` in the root directory, and `./scripts/lint.sh` to run the linter.

CHAPTER

EIGHT

MODULES

8.1 Converters

8.1.1 Interface

All converters conform to this interface.

Interface and exceptions for all converters

class conversions.base.**Converter**

Bases: ABC

Interface for converters

convert_to_ut(*data*)

Parses the file and returns the parsed data

Parameters
data – A File object, filename, or json data. Not all converters support JSON.

Returns
The Universal Tabulator representation of this data. Call *convert_to_ut_and_validate()* to guarantee that it matches the Universal Tabulator schema.

Raises

- **CouldNotConvertException** – If the conversion could not complete
- **CouldNotOpenFileException** – If the file couldn't be opened

convert_to_ut_and_validate(*filename_or_fileobj*)

Calls *convert_to_ut()*, then validates it with the Universal Tabulator schema.

Parameters
filename – A File object or filename

Returns
Guaranteed-valid Universal Tabulator data

Raises
CouldNotConvertException – If the conversion could not complete

convert_to_ut_without_exceptions(*data*)

See *convert_to_ut()*. This is the workhorse, without exceptions. To debug. call this in *convert_to_ut_and_validate()*

classmethod postprocess_remove_last_round_elimination(*data*)

When there are two candidates left, Dominion marks the loser among them as "eliminated", whereas the URCVT format does not. Updates data to remove any last-round eliminations

classmethod postprocess_use_standard_irv_threshold(*data*)

Set the threshold based on (last round active votes) / (num winners + 1)

8.1.2 Exceptions

Some of the possible exceptions that may be thrown during conversion

Interface and exceptions for all converters

exception conversions.base.**CouldNotConvertException**

Bases: Exception

Raised when an unexpected error prevented conversion. This could be anything from an invalid input file, to unsupported data, to a bug in the software.

8.1.3 Automatic Converter

Allows you to provide any file and it will try to figure out its type and convert it.

Attempts to convert any file to the standard format. If you know the file format, you should not use this - it will loop through all schemas which will be needlessly slow.

class conversions.automatic.**AutomaticConverter**

Bases: Converter

Interface for converters

_convert_file_object_to_ut(*file_object*)

Just like func:~*convert_to_ut*, but only accepting a file object

8.1.4 Internal developer documentation

The remainder of this documentation is about the internal representation of classes. It is intended for developers of RCV Formats, rather than its users.

Transferless format helpers

Helper base class for any file format which does not explicitly spell out how votes are transferred between candidates.

Interface and exceptions for all converters

class conversions.base.**GenericGuessAtTransferConverter**

Bases: *Converter*

If there are multiple candidates transferring their votes, we cannot know which candidates contributed to which transfers. Our best-effort guess is that they distributed their votes equally to all candidates.

This base class lets you fill out partial data, leaving out tallyResults, and it will guess at the tallyResults for you.

Note that it always knows the correct tally results if only one candidate is eliminated or elected in each round - the guessing only happens when multiple candidates are transferring their votes.

classmethod _compute_vote_deltas_from_tally(*tally_this_round*, *tally_next_round*)

Returns the vote deltas between this round and next round. The two tally arguments must match the tally format in the Universal Tabulator format.

Parameters

- **tally_this_round** – A dict mapping candidate names to number of votes this round
- **tally_next_round** – A dict mapping candidate names to number of votes next round, ensuring that any eliminated candidates are NOT present next round, NOT that they just have zero votes.

Returns

A dict mapping candidate names in this round to how their votes changed between this round and the next. Includes positive numbers (gained votes) and negative numbers (lost votes via elimination or surplus transfer)

classmethod _weights_for_each_transfer(*eliminated_names*, *elected_names*, *vote_delta*)

Parameters

- **eliminated_names** – the names of each eliminated candidate
- **elected_names** – the names of each elected candidate
- **vote_delta** – a dict mapping candidate name to vote difference between this round and the next round

Returns

a dict mapping a transferring candidate's name to the weight they contributed to the overall transfer. In most cases, there will only be one candidate in transferring_candidates and the weight will be a simple 1.0.

classmethod compute_vote_deltas_for_round(*ut_rounds_tally_only*, *round_i*)

Gathers some data and passes it on to `_compute_vote_deltas_from_tally()`

Parameters

- **ut_rounds_tally_only** – Incomplete Universal Tabulator 'results' structure, containing only 'tally' but not 'tallyResults'. The tallies must be numbers, not strings.
- **round_i** – Computes delta between this round and the next

Returns

Value from `_compute_vote_deltas_from_tally()`

classmethod guess_at_tally_results(*eliminated_names*, *elected_names*, *vote_delta*, *allow_guessing=True*)

Computes the tallyResult, the difference between this round and the next See the description of @_compute_tally_results to understand why, in the case of multiple winners, the best we can do is guess at the transfers here.

Parameters

- **eliminated_names** – the names of each eliminated candidate
- **elected_names** – the names of each elected candidate
- **vote_delta** – a dict mapping candidate name to vote difference between this round and the next round.
- **allow_guessing** – Allow guessing of transfer data during batch elimination

Returns
The contents of the tallyResults dict

Opavote to Universal Tabulator

Reads an ElectionBuddy CSV results file, writes to the standard format

class conversions.opavote.**OpavoteConverter**

Bases: GenericGuessAtTransferConverter

Reads an opavote-formatted JSON file.

_convert_file_object_to_ut(*file_object*)

Just like func:~*convert_to_ut*, but only accepting a file object

classmethod _fill_in_tallyresults(*rounds*, *candidate_names*, *ut_rounds*)

Fill out rounds['tallyResults'] based on rounds['tally']

classmethod _get_elected_names(*rounds*, *candidate_names*, *round_i*)

Parameters

- **candidate_names** – in-order names
- **rounds** – rounds data, direct from the Opavote format
- **round_i** – the current round

Returns
list of names that were elected

classmethod _get_eliminated_names(*rounds*, *candidate_names*, *round_i*)

Opavote format places losses on the round after they happen, whereas the Universal Tabulator format places it on the previous round. This function looks at the next round for its data. A corellary is that there are no eliminations on the first round, which is what I believe to always be the case anyway.

Parameters

- **rounds** – rounds data, direct from the Opavote format
- **candidate_names** – in-order names
- **round_i** – the current round (we'll look at round_i+1)

Returns
list of names that were eliminated

classmethod _votes_on_round(*candidate_i*, *rounds*, *round_i*)

Number of votes the corresponding candidate had on round_i

ElectionBuddy to Universal Tabulator

Reads an ElectionBuddy CSV results file, writes to the standard format

class conversions.electionbuddy.**ElectionBuddyConverter**

Bases: GenericGuessAtTransferConverter

Reads an electionbuddy-formatted CSV file. Note that this use a generic text file reader, not a CSV reader, because it's not really a CSV file - it has parts of it that are, but it also has miscellaneous title lines.

_convert_file_object_to_ut(*file_object*)

Just like func:~*convert_to_ut*, but only accepting a file object

classmethod _fill_in_tallyresults_from_tally(*rounds*, *ut_rounds*)

Fills in tallyResults in ut_rounds

classmethod _get_elected_names(*rounds*, *round_i*, *already_elected_set*)

Returns
a list of names of candidates elected on this round

classmethod _get_round_data_without_tallyresults(*rounds*)

Generates the ut_rounds data without tallyResults

classmethod _get_threshold(*csv_data*)

Returns the threshold for the overall election, which doesn't make a lot of sense for dynamic-threshold multiwinner elections, but we mark it as just the last threshold in the file if the file contains thresholds.

classmethod _threshold_for_round(*rounds*, *round_i*)

If the threshold is not provided in each round, assume it is half of the number of voters present in the last round. This means that if the thresold is not provided, we assume it is a single-winner election (unless there is an exact 50/50 tie) TODO: This should probably be handled by a migration function, since ElectionBuddy no longer outputs data without thresholds.

Dominion to Universal Tabulator

Reads an Dominion TXT results file, used by Alaska since they have consistently failed to publish the easier-to-read Dominion JSON file.

class conversions.dominion_txt.**DominionTxtConverter**

Bases: GenericGuessAtTransferConverter

Parses the dominion file format as exemplified in /testdata/inputs/dominion.txt

_convert_file_object_to_ut(*file_object*)

Just like func:~*convert_to_ut*, but only accepting a file object

classmethod _get_transfer_data(*line*)

These lines have cells 20-22 which looks like, including the quotes: 20: "From" 21: "To" 22: Num Votes

returns None if not relevant

classmethod _name_strip(*name*)

Strips quotes first, then spaces

classmethod _who_is_eliminated_or_elected(*elim_or_elect_text*, *line*)

These lines have box 17 which looks like, including the quotes: "Palin, Sarah is eliminated because the candidate was not elected in the last round."

returns None if nobody was

Reads an Dominion XLSX results file, writes to the standard format

class conversions.dominion_xlsx.**DominionXlsxConverter**

Bases: GenericGuessAtTransferConverter

Parses the dominion file format as exemplified in /testdata/inputs/dominion-json These are .xlsx files

class RoundInfo

Bases: object

Data for parsing a round: Because columns are hapharzadly merged, there is no straightforward mapping from round number to the corresponding column. This little struct keeps track of it for us.

class RowConstants

Bases: object

Data for the row number that various items are on

classmethod _count_num_header_rows(*sheet*)

The number of header rows can vary. Looks for the first left-aligned row, which is row 12 or 13 in all data we've seen.

_find_row_of_inactive_ballots(*sheet*, *num_candidates*)

Returns row number labeled "Non-Transferrable Total"

_try_to_find_row_of_threshold(*sheet*, *inactive_row*)

Returns row number labeled "Threshold"

find_rows_after_summary_table(*sheet*, *num_candidates*)

Fills in values for rows after the summary table, which requires needing to know the number of candidates

find_rows_before_summary_table(*workbook*)

Fills in values for rows before the summary table

_convert_file_object_to_ut(*file_object*)

Just like func:~*convert_to_ut*, but only accepting a file object

classmethod _is_elected_color(*color*)

Is this cell colored greed, noting it is elected? Note: only call this if you've checked that the cell is filled solid. It may be the case that the cell is red, but not filled, in which case it appears white and is not eliminated.

classmethod _is_eliminated_color(*color*)

Is this cell colored red for elimination? Note: only call this if you've checked that the cell is filled solid. It may be the case that the cell is red, but not filled, in which case it appears white and is not eliminated.

_parse_candidates()

Grabs the list of candidate names from the summary table

_parse_config()

Returns the URCV config format

_parse_rounds()

Rounds are curious - they are separated by a varying number of columns, usually 3 except for the first few, where the initial columns have been merged seemingly randomly (but in actually: twice in round 1, once in round 2)

Therefore, this returns a pair of (Round, Column #) to get the column for the number of votes in each Round

_parse_tally_for_round_at_column(*col*, *eliminated_names*)

Creates a 'tally' and 'tallyResults' struct for the given round

_postprocess_set_threshold_from_spreadsheet(*data*, *sheet*)

The threshold is always listed on the table of per-round info We don't guess here - if we can't find it, we leave it blank.

Reads an Dominion XML file containing many contests.

class conversions.dominion_multi_converter.**DominionMultiConverter**

Bases: object

Parses the dominion first-round-only file format as exemplified in testdata/inputs/dominion-multi-converter.xml, which contains many elections.

classmethod explode_to_files(*file_object*)

Given the XML format with multiple elections, explodes into many files, and returns a dictionary of titles to NamedTemporaryFiles.

8.2 Schemas

8.2.1 Interface

All schemas conform to this interface.

Loads supported schemas (currently only one: the Universal Tabulator schema)

class schemas.base.**Schema**

Bases: ABC

A single version of a single schema

last_error()

If validate() failed, this method will provide more detailed information on the error. The details vary by class type, though it will always be of type Exception

Returns
Exception with additional information on why the validation failed

validate(*data*)

Validates that the file matches the expected schema

Parameters
data – The JSON filename, file object, or JSON data for the tabulated results

Returns
whether or not the validation failed

abstract version()

The version number of this schema

Returns
A string represting the version number

8.2.2 Internal developer documentation

The remainder of this documentation is about the internal representation of classes. It is intended for developers of RCV Formats, rather than its users.

JSONSchema helper

A helper interface for all jsonschema-backed schemas

Loads supported schemas (currently only one: the Universal Tabulator schema)

class schemas.base.**GenericJsonSchema**

Bases: *Schema*

Base class for a JSON Schema

ensure_data_is_logical(*data*)

Override to add any additional data validations you need done. Raise an exception if anything is invalid

is_data_valid(*data*)

Additional validations to ensure the data is correct.

Parameters
data – The input dictionary, which must match the schema

Returns
Whether or not the data is acceptable

is_schema_valid(*data*)

Validates that the data matches the schema. If invalid, more data may be available by calling last_error()

Parameters
data – The input dictionary

Returns
Whether or not the data matches the schema

abstract property schema_filename

The JSON Schema filename

validate_schema_and_logic(*data*)

Runs both the schema and logic check

Universal Tabulator

Loads the universal tabulator schema

class schemas.universaltabulator.**SchemaV0**

Bases: GenericJsonSchema

Schema for the initial version of the Universal RCV Tabulator

classmethod check_candidate_leaves_after_elimination(*data*)

After a candidate is eliminated, ensure they are not listed later as having zero votes. They should be removed from the list.

classmethod check_last_round_eliminations(*data*)

No eliminations allowed on the last round

classmethod check_no_empty_candidate_names(*data*)

All candidate names must be unique

classmethod check_unique_candidate_names(*data*)

All candidate names must be unique

ensure_data_is_logical(*data*)

Various checks to ensure the data is sane - though it cannot catch everything, we have tried to place the most common errors here

property schema_filename

The JSON Schema filename

version()

The version number of this schema

Returns

A string represting the version number

Opavote

Loads the opavote schema

class schemas.opavote.**SchemaV1_0**

Bases: GenericJsonSchema

Schema for the initial Opavote schema

property schema_filename

The JSON Schema filename

version()

The version number of this schema

Returns

A string represting the version number

class schemas.opavote.**SchemaV1_1**

Bases: GenericJsonSchema

Schema for the version created on 2022-05-18

property schema_filename

The JSON Schema filename

version()

The version number of this schema

Returns

A string represting the version number

8.3 Shared Utilities

8.3.1 Internal developer documentation

This document is about the internal representation of classes. It is intended for developers of RCV Formats, rather than its users.

ElectionBuddy Parser

Parser used for both schema validation and conversion

Helper class for loading Electionbuddy CSVs

class common.electionbuddyparser.**ElectionBuddyData**(*file_obj*)

Bases: object

Structure to hold the raw data read from the file. Parses the data directly into a simple format. Used for both conversion and schema validation.

peek_line()

Peeks at the next line without advancing

read_each_round()

Start iterating over the CSV for each round

read_line_as_str()

Read line, and if it's bytes, decode to utf-8

read_round()

Reads the CSV data for the next round

Utilities

Set of common utilities

A collection of shared helper utilities

common.utils.**is_file_obj**(*filename_or_fileobj*)

Is the given argument a File-like object?

common.utils.**is_filename**(*filename_or_fileobj*)

Is the given argument a filename?

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX

Symbols

A

C

D

E

F

I

L

M

O

P

R

S

V