Psifr

Release v0.2.1

Neal Morton

Feb 22, 2022

CONTENTS

1 Installation **3**

2 User guide **5**
- 2.1 Importing data . . . 5
- 2.2 Scoring data . . . 6
- 2.3 Conditional response probability . . . 7

3 Tutorials **11**

4 API Reference **13**
- 4.1 Transitions . . . 13
- 4.2 Free Recall Analysis . . . 14

Python Module Index **17**

Index **19**

In free recall, participants study a list of items and then name all of the items they can remember in any order they choose. Many sophisticated analyses have been developed to analyze data from free recall experiments, but these analyses are often complicated and difficult to implement.

Psifr leverages the Pandas data analysis package to make precise and flexible analysis of free recall data faster and easier.

CHAPTER

ONE

INSTALLATION

First get a copy of the code from GitHub:

```
git clone git@github.com:mortonne/psifr.git
```

Then install:

```
cd psifr
python setup.py install
```

CHAPTER

TWO

USER GUIDE

2.1 Importing data

In Psifr, free recall data are imported in the form of a "long" format table. Each row corresponds to one *study* or *recall* event. Study events include any time an item was presented to the participant. Recall events correspond to any recall attempt; this includes *repeats* of items there were already recalled and *intrusions* of items that were not present in the study list.

This type of information is well represented in a CSV spreadsheet, though any file format supported by pandas may be used for input. To import from a CSV, use pandas. For example:

```
import pandas as pd
data = pd.read_csv("my_data.csv")
```

2.1.1 Trial information

The basic information that must be included for each event is the following:

subject Some code (numeric or string) indicating individual participants. Must be unique for a given experiment. For example, `sub-101`.

list Numeric code indicating individual lists. Must be unique within subject.

trial_type String indicating whether each event is a `study` event or a `recall` event.

position Integer indicating position within a given phase of the list. For `study` events, this corresponds to *input position* (also referred to as *serial position*). For `recall` events, this corresponds to *output position*.

item Individual thing being recalled, such as a word. May be specified with text (e.g., `pumpkin`, `Jack Nicholson`) or a numeric code (`682`, `121`). Either way, the text or number must be unique to that item. Text is easier to read and does not require any additional information for interpretation and is therefore preferred if available.

2.1.2 Example

Table 1: Sample data

subject	list	trial_type	position	item
1	1	study	1	absence
1	1	study	2	hollow
1	1	study	3	pupil
1	1	recall	1	pupil
1	1	recall	2	absence

2.1.3 Additional information

Additional fields may be included in the data to indicate other aspects of the experiment, such as presentation time, stimulus category, experimental session, distraction length, etc. All of these fields can then be used for analysis in Psifr.

2.2 Scoring data

After *importing free recall data*, we have a DataFrame with a row for each study event and a row for each recall event. Next, we need to score the data by matching study events with recall events.

2.2.1 Scoring list recall

First, let's create a simple sample dataset with two lists:

```
In [1]: import pandas as pd

In [2]: data = pd.DataFrame(
   ...:     {'subject': [1, 1, 1, 1, 1, 1,
   ...:                  1, 1, 1, 1, 1, 1],
   ...:      'list': [1, 1, 1, 1, 1, 1,
   ...:               2, 2, 2, 2, 2, 2],
   ...:      'trial_type': ['study', 'study', 'study',
   ...:                     'recall', 'recall', 'recall',
   ...:                     'study', 'study', 'study',
   ...:                     'recall', 'recall', 'recall'],
   ...:      'position': [1, 2, 3, 1, 2, 3,
   ...:                   1, 2, 3, 1, 2, 3],
   ...:      'item': ['absence', 'hollow', 'pupil',
   ...:               'pupil', 'absence', 'empty',
   ...:               'fountain', 'piano', 'pillow',
   ...:               'pillow', 'fountain', 'pillow']})
   ...:

In [3]: data
Out[3]:
    subject  list trial_type  position      item
0         1     1      study         1   absence
1         1     1      study         2    hollow
2         1     1      study         3     pupil
3         1     1     recall         1     pupil
```

(continues on next page)

(continued from previous page)

```
4          1     1     recall         2   absence
5          1     1     recall         3     empty
6          1     2      study         1  fountain
7          1     2      study         2     piano
8          1     2      study         3    pillow
9          1     2     recall         1    pillow
10         1     2     recall         2  fountain
11         1     2     recall         3    pillow
```

Next, we'll merge together the study and recall events by matching up corresponding events:

```
In [4]: from psifr import fr

In [5]: study = data.query('trial_type == "study"').copy()

In [6]: recall = data.query('trial_type == "recall"').copy()

In [7]: merged = fr.merge_lists(study, recall)

In [8]: merged
Out[8]:
   subject  list      item  input  output  recalled  repeat  intrusion
0        1     1   absence    1.0     2.0      True       0      False
1        1     1    hollow    2.0     NaN     False       0      False
2        1     1     pupil    3.0     1.0      True       0      False
3        1     1     empty    NaN     3.0      True       0       True
4        1     2  fountain    1.0     2.0      True       0      False
5        1     2     piano    2.0     NaN     False       0      False
6        1     2    pillow    3.0     1.0      True       0      False
7        1     2    pillow    3.0     3.0      True       1      False
```

For each item, there is one row for each unique combination of input and output position. For example, if an item is presented once in the list, but is recalled multiple times, there is one row for each of the recall attempts. Repeated recalls are indicated by the *repeat* column, which is greater than zero for recalls of an item after the first.

Items that were not recalled have the *recalled* column set to *False*. Because they were not recalled, they have no defined output position, so *output* is set to *NaN*. Finally, intrusions have an output position but no input position because they did not appear in the list. There is an *intrusion* field for convenience to label these recall attempts.

2.3 Conditional response probability

A key advantage of free recall is that it provides information not only about what items are recalled, but also the order in which they are recalled. A number of analyses have been developed to charactize different influences on recall order, such as the temporal order in which the items were presented at study, the category of the items themselves, or the semantic similarity between pairs of items.

Each conditional response probability (CRP) analysis involves calculating the probability of some type of transition event. For the lag-CRP analysis, transition events of interest are the different lags between serial positions of items recalled adjacent to one another. Similar analyses focus not on the serial position in which items are presented, but the properties of the items themselves. A semantic-CRP analysis calculates the probability of transitions between items in different semantic relatedness bins. A special case of this analysis is when item pairs are placed into one of two bins, depending on whether they are in the same stimulus category or not. In Psifr, this is referred to as a category-CRP analysis.

2.3.1 Actual and possible transitions

Calculating a conditional response probability involves two parts: the frequency at which a given event actually occurred in the data and frequency at which a given event could have occurred. The frequency of possible events is calculated conditional on the recalls that have been made leading up to each transition. For example, a transition between item i and item j is not considered "possible" in a CRP analysis if item i was never recalled. The transition is also not considered "possible" if, when item i is recalled, item j has already been recalled previously.

Repeated recall events are typically excluded from the counts of both actual and possible transition events. That is, the transition event frequencies are conditional on the transition not being either to or from a repeated item.

Calculating a CRP measure involves tallying how many transitions of a given type were made during a free recall test. For example, one common measure is the serial position lag between items. For a list of length N, possible lags are in the range $[-N + 1, N - 1]$. Because repeats are excluded, a lag of zero is never possible. The count of actual and possible transitions for each lag is calculated first, and then the CRP for each lag is calculated as the actual count divided by the possible count.

2.3.2 The transitions masker

The *psifr.transitions.transitions_masker()* is a generator that makes it simple to iterate over transitions while "masking" out events such as intrusions of items not on the list and repeats of items that have already been recalled.

On each step of the iterator, the previous, current, and possible items are yielded. The *previous* item is the item being transitioned from. The *current* item is the item being transitioned to. The *possible* items includes an array of all items that were valid to be recalled next, given the recall sequence up to that point (not including the current item).

```
In [1]: from psifr.transitions import transitions_masker

In [2]: pool = [1, 2, 3, 4, 5, 6]

In [3]: recs = [6, 2, 3, 6, 1, 4]

In [4]: masker = transitions_masker(pool_items=pool, recall_items=recs,
   ...:                             pool_output=pool, recall_output=recs)
   ...:

In [5]: for prev, curr, poss in masker:
   ...:     print(prev, curr, poss)
   ...:
6 2 [1 2 3 4 5]
2 3 [1 3 4 5]
1 4 [4 5]
```

Only valid transitions are yielded, so the code for a specific analysis only needs to calculate the transition measure of interest and count the number of actual and possible transitions in each bin of interest.

Four inputs are required:

pool_items List of identifiers for all items available for recall. Identifiers can be anything that is unique to each item in the list (e.g., serial position, a string representation of the item, an index in the stimulus pool).

recall_items List of identifiers for the sequence of recalls, in order. Valid recalls must match an item in *pool_items*. Other items are considered intrusions.

pool_output Output codes for each item in the pool. This should be whatever you need to calculate your transition measure.

recall_output Output codes for each recall in the sequence of recalls.

By using different values for these four inputs and defining different transition measures, a wide range of analyses can be implemented.

CHAPTER

THREE

TUTORIALS

See the psifr-notebooks project for sample code.

CHAPTER

FOUR

API REFERENCE

4.1 Transitions

The transitions module contains utilties to iterate over and mask transitions between recalled items. The *psifr.transitions.transitions_masker()* does most of the work here.

Module to analyze transitions during free recall.

psifr.transitions.**count_category**(*pool_items*, *recall_items*, *pool_category*, *recall_category*, *pool_test=None*, *recall_test=None*, *test=None*)

Count within-category transitions.

psifr.transitions.**count_lags**(*pool_items*, *recall_items*, *pool_test=None*, *recall_test=None*, *test=None*)

Count actual and possible serial position lags.

Parameters

- **pool_items** (*list*) – List of the serial positions available for recall in each list. Must match the serial position codes used in *recall_items*.
- **recall_items** (*list*) – List indicating the serial position of each recall in output order (NaN for intrusions).
- **pool_test** (*list, optional*) – List of some test value for each item in the pool.
- **recall_test** (*list, optional*) – List of some test value for each recall attempt by output position.
- **test** (*callable*) – Callable that evaluates each transition between items n and n+1. Must take test values for items n and n+1 and return True if a given transition should be included.

psifr.transitions.**count_pairs**(*n_item*, *pool_items*, *recall_items*, *pool_test=None*, *recall_test=None*, *test=None*)

Count transitions between pairs of specific items.

psifr.transitions.**transitions_masker**(*pool_items*, *recall_items*, *pool_output*, *recall_output*, *pool_test=None*, *recall_test=None*, *test=None*)

Iterate over transitions with masking.

Transitions are between a "previous" item and a "current" item. Non-included transitions will be skipped. A transition is yielded only if it matches the following conditions:

(1) Each item involved in the transition is in the pool. Items are removed from the pool after they appear as the previous item.

(2) Optionally, an additional check is run based on test values associated with the items in the transition. For example, this could be used to only include transitions where the category of the previous and current items is the same.

The masker will yield "output" values, which may be distinct from the item identifiers used to determine item repeats.

Parameters

- **pool_items** (*list*) – Items available for recall. Order does not matter. May contain repeated values. Item identifiers must be unique within pool.
- **recall_items** (*list*) – Recalled items in output position order.
- **pool_output** (*list*) – Output values for pool items. Must be the same order as pool.
- **recall_output** (*list*) – Output values in output position order.
- **pool_test** (*list, optional*) – Test values for items available for recall. Must be the same order as pool.
- **recall_test** (*list, optional*) – Test values for items in output position order.
- **test** (*callable, optional*) – Used to test whether individual transitions should be included, based on test values.

  test(prev, curr) - test for included transition

  test(prev, poss) - test for included possible transition

Yields

- **prev** (*object*) – Output value for the "from" item on this transition.
- **curr** (*object*) – Output value for the "to" item.
- **poss** (*numpy.array*) – Output values for all possible valid "to" items.

4.2 Free Recall Analysis

Utilities for working with free recall data.

psifr.fr.**block_index**(*list_labels*)
Get index of each block in a list.

psifr.fr.**check_data**(*df*)
Run checks on free recall data.

Parameters **df** (*pandas.DataFrame*) –

Contains one row for each trial (study and recall). Must have fields:

subject [number or str] Subject identifier.

list [number] List identifier. This applies to both study and recall trials.

trial_type [str] Type of trial; may be 'study' or 'recall'.

position [number] Position within the study list or recall sequence.

item [str] Item that was either presented or recalled on this trial.

psifr.fr.**get_recall_index**(*df*, *list_cols=None*)
Get recall input position index by list.

psifr.fr.**get_study_value**(*df*, *column*, *list_cols=None*)
Get study column value by list.

psifr.fr.**lag_crp**(*df*, *test_values=None*, *test=None*, *first_output=None*)

Lag-CRP for multiple subjects.

Parameters

- **df** (*pandas.DataFrame*) – Merged study and recall data. See merge_lists. List length is assumed to be the same for all lists within each subject. Must have fields: subject, list, input, output, recalled. Input position must be defined such that the first serial position is 1, not 0.
- **test_values** (*pandas.Series or column name, optional*) – Column with labels to use when testing transitions for inclusion.
- **test** (*callable, optional*) – Callable that takes in previous and current item values and returns True for transitions that should be included.
- **first_output** (*int, optional*) – First output position to include when calculating transition probabilities. Used to exclude initial outputs. Default is to start at the first recall on each list.

Returns

results – Has fields:

subject [hashable] Results are separated by each subject.

lag [int] Lag of input position between two adjacent recalls.

prob [float] Probability of each lag transition.

actual [int] Total of actual made transitions at each lag.

possible [int] Total of times each lag was possible, given the prior input position and the remaining items to be recalled.

Return type pandas.DataFrame

psifr.fr.**merge_lists**(*study*, *recall*, *merge_keys=None*, *list_keys=None*, *study_keys=None*, *recall_keys=None*, *position_key='position'*)

Merge study and recall events together for each list.

Parameters

- **study** (*pandas.DataFrame*) – Information about all study events. Should have one row for each study event.
- **recall** (*pandas.DataFrame*) – Information about all recall events. Should have one row for each recall attempt.
- **merge_keys** (*list, optional*) – Columns to use to designate events to merge. Default is ['subject', 'list', 'item'], which will merge events related to the same item, but only within list.
- **list_keys** (*list, optional*) – Columns that apply to both study and recall events.
- **study_keys** (*list, optional*) – Columns that only apply to study events.
- **recall_keys** (*list, optional*) – Columns that only apply to recall events.
- **position_key** (*str, optional*) – Column indicating the position of each item in either the study list or the recall sequence.

Returns

merged – Merged information about study and recall events. Each row corresponds to one unique input/output pair.

The following columns will be added:

input [int] Position of each item in the input list (i.e., serial position).

output [int] Position of each item in the recall sequence.

recalled [bool] True for rows with an associated recall event.

repeat [int] Number of times this recall event has been repeated (0 for the first recall of an item).

intrusion [bool] True for recalls that do not correspond to any study event.

Return type pandas.DataFrame

PYTHON MODULE INDEX

p

psifr.fr, 14
psifr.transitions, 13

INDEX

B

block_index() (*in module psifr.fr*), 14

C

check_data() (*in module psifr.fr*), 14
count_category() (*in module psifr.transitions*), 13
count_lags() (*in module psifr.transitions*), 13
count_pairs() (*in module psifr.transitions*), 13

G

get_recall_index() (*in module psifr.fr*), 14
get_study_value() (*in module psifr.fr*), 14

L

lag_crp() (*in module psifr.fr*), 14

M

merge_lists() (*in module psifr.fr*), 15

P

psifr.fr (*module*), 14
psifr.transitions (*module*), 13

T

transitions_masker() (*in module psifr.transitions*), 13