# Psifr

***Release v0.4.0***

**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 Analysis of recall order . . . . . . . . 8

**3 Tutorials** **13**

**4 API Reference** **15**
- 4.1 Transitions . . . . . . . . 15
- 4.2 Free Recall Analysis . . . . . . . . 21

**Python Module Index** **39**

**Index** **41**

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:

```python
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  study  recall  repeat  intrusion
0        1     1   absence    1.0     2.0   True    True       0      False
1        1     1    hollow    2.0     NaN   True   False       0      False
2        1     1     pupil    3.0     1.0   True    True       0      False
3        1     1     empty    NaN     3.0  False    True       0       True
4        1     2  fountain    1.0     2.0   True    True       0      False
5        1     2     piano    2.0     NaN   True   False       0      False
6        1     2    pillow    3.0     1.0   True    True       0      False
7        1     2    pillow    3.0     3.0  False    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. Unique study events are indicated by the *study* column; this excludes intrusions and repeated recalls.

Items that were not recalled have the *recall* 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.

*merge_lists()* can also handle additional attributes beyond the standard ones, such as codes indicating stimulus category or list condition. See the *merge_lists()* documentation for details.

## 2.2.2 Filtering and sorting

Now that we have a merged *DataFrame*, we can use *pandas* methods to quickly get different views of the data. For some analyses, we may want to organize in terms of the study list by removing repeats and intrusions. Because our data are in a *DataFrame*, we can use the *DataFrame.query* method:

```
In [9]: merged.query('study')
Out[9]:
   subject  list      item  input  output  study  recall  repeat  intrusion
0        1     1   absence    1.0     2.0   True    True       0      False
1        1     1    hollow    2.0     NaN   True   False       0      False
2        1     1     pupil    3.0     1.0   True    True       0      False
```

(continues on next page)

(continued from previous page)

```
4         1      2  fountain    1.0      2.0   True     True         0        False
5         1      2     piano    2.0      NaN   True    False         0        False
6         1      2    pillow    3.0      1.0   True     True         0        False
```

Alternatively, we may also want to get just the recall events, sorted by output position instead of input position:

```
In [10]: merged.query('recall').sort_values(['list', 'output'])
Out[10]:
   subject  list      item  input  output  study  recall  repeat  intrusion
2        1     1     pupil    3.0     1.0   True    True       0      False
0        1     1   absence    1.0     2.0   True    True       0      False
3        1     1     empty    NaN     3.0  False    True       0       True
6        1     2    pillow    3.0     1.0   True    True       0      False
4        1     2  fountain    1.0     2.0   True    True       0      False
7        1     2    pillow    3.0     3.0  False    True       1      False
```

Note that we first sort by list, then output position, to keep the lists together.

## 2.3 Analysis of recall order

### 2.3.1 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.

#### 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.

### 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.

## 2.3.2 Lag-CRP analysis

First, load some sample data and create a merged DataFrame:

```
In [1]: from pkg_resources import resource_filename

In [2]: import pandas as pd

In [3]: from psifr import fr

In [4]: data_file = resource_filename('psifr', 'data/Morton2013.csv')
```

(continues on next page)

(continued from previous page)

```
In [5]: df = pd.read_csv(data_file)

In [6]: study = df.query('trial_type == "study"').copy()

In [7]: recall = df.query('trial_type == "recall"').copy()

In [8]: data = fr.merge_lists(study, recall)
```

Next, call `lag_crp()` to calculate conditional response probability as a function of lag.

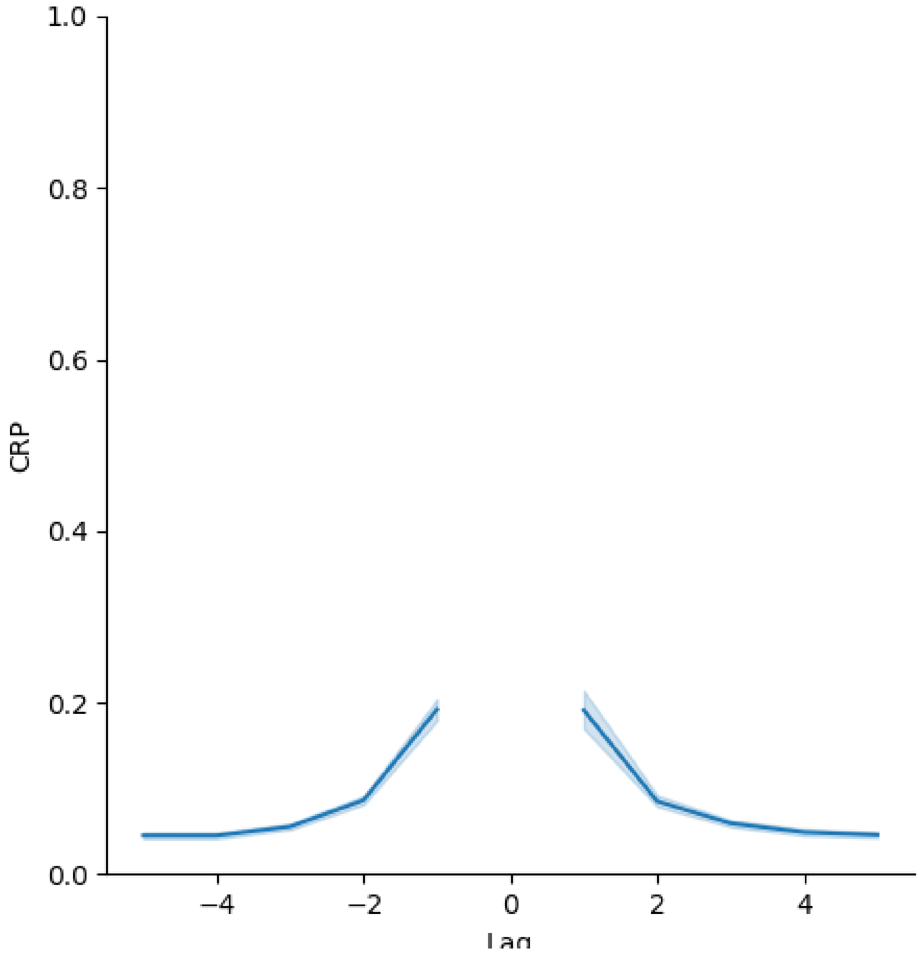
```
In [9]: crp = fr.lag_crp(data)

In [10]: crp
Out[10]:
                   prob  actual  possible
subject lag
1       -23.0  0.020833       1        48
        -22.0  0.035714       3        84
        -21.0  0.026316       3       114
        -20.0  0.024000       3       125
        -19.0  0.014388       2       139
...                 ...     ...       ...
47       19.0  0.061224       3        49
         20.0  0.055556       2        36
         21.0  0.045455       1        22
         22.0  0.071429       1        14
         23.0  0.000000       0         6

[1880 rows x 3 columns]
```

Use `plot_lag_crp()` to display the results:

```
In [11]: g = fr.plot_lag_crp(crp, height=5)
```


1.0
0.8
0.6
0.4
0.2
0.0
CRP
−4
−2
0
2
4
Lag

# 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.

**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_category`** (*`list`*) – List of the category of each item in the pool for each list.
- **`recall_category`** (*`list`*) – List of item category in recall order.
- **`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.

**Returns**

- **actual** (*int*) – Count of actual within-category transitions.
- **possible** (*int*) – Count of possible within-category transitions.

## Examples

```
>>> from psifr import transitions
>>> pool_items = [[1, 2, 3, 4]]
>>> recall_items = [[4, 3, 1, 2]]
>>> pool_category = [[1, 1, 2, 2]]
>>> recall_category = [[2, 2, 1, 1]]
>>> transitions.count_category(
...     pool_items, recall_items, pool_category, recall_category
... )
(2, 2)
```

`psifr.transitions.`**`count_distance`**(*distances*, *edges*, *pool_items*, *recall_items*, *pool_index*, *recall_index*, *pool_test=None*, *recall_test=None*, *test=None*, *count_unique=False*)

Count transitions within distance bins.

**Parameters**

- **`distances`** (*`numpy.array`*) – Items x items matrix of pairwise distances or similarities.
- **`edges`** (*`array-like`*) – Edges of bins to apply to distances.
- **`pool_items`** (*`list of list`*) – Unique item codes for each item in the pool available for recall.
- **`recall_items`** (*`list of list`*) – Unique item codes of recalled items.
- **`pool_index`** (*`list of list`*) – Index of each item in the distances matrix.
- **`recall_index`** (*`list of list`*) – Index of each recalled item.
- **`pool_test`** (*`list of list, optional`*) – Test value for each item in the pool.
- **`recall_test`** (*`list of list, optional`*) – Test value for each recalled item.
- **`test`** (*`callable`*) – Called as test(prev, curr) or test(prev, poss) to screen actual and possible transitions, respectively.
- **`count_unique`** (*`bool, optional`*) – If true, only unique values will be counted toward the possible transitions. If multiple items are avilable for recall for a given transition and a given bin, that bin will only be incremented once. If false, all possible transitions will add to the count.

**Returns**

- **actual** (*pandas.Series*) – Count of actual transitions made for each bin.
- **possible** (*pandas.Series*) – Count of possible transitions for each bin.

**See also:**

***`rank_distance()`*** Calculate percentile rank of transition distances.

### Examples

```
>>> from psifr import transitions
>>> distances = np.array([[0, 1, 2, 2], [1, 0, 2, 2], [2, 2, 0, 3], [2, 2, 3, 0]])
>>> edges = np.array([0.5, 1.5, 2.5, 3.5])
>>> pool_items = [[1, 2, 3, 4]]
>>> recall_items = [[4, 2, 3, 1]]
>>> pool_index = [[0, 1, 2, 3]]
>>> recall_index = [[3, 1, 2, 0]]
>>> actual, possible = transitions.count_distance(
...     distances, edges, pool_items, recall_items, pool_index, recall_index
... )
>>> actual
(0.5, 1.5]    0
(1.5, 2.5]    3
(2.5, 3.5]    0
dtype: int64
>>> possible
(0.5, 1.5]    1
(1.5, 2.5]    4
(2.5, 3.5]    1
dtype: int64
```

`psifr.transitions.`**`count_lags`**(*list_length*, *pool_items*, *recall_items*, *pool_label=None*, *recall_label=None*, *pool_test=None*, *recall_test=None*, *test=None*, *count_unique=False*)

Count actual and possible serial position lags.

**Parameters**

- **`list_length`** (*`int`*) – Number of items in each list.
- **`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_label`** (*`list, optional`*) – List of the positions to use for calculating lag. Default is to use *pool_items*.
- **`recall_label`** (*`list, optional`*) – List of position labels in recall order. Default is to use *recall_items*.
- **`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.
- **`count_unique`** (*`bool, optional`*) – If true, only unique values will be counted toward the possible transitions. If multiple items are avilable for recall for a given transition and a given bin, that bin will only be incremented once. If false, all possible transitions will add to the count.

**Returns**

- **actual** (*pandas.Series*) – Count of actual lags that occurred in the recall sequence.
- **possible** (*pandas.Series*) – Count of possible lags.

**See also:**

***rank_lags()*** Rank of serial position lags.

### Examples

```
>>> from psifr import transitions
>>> pool_items = [[1, 2, 3, 4]]
>>> recall_items = [[4, 2, 3, 1]]
>>> actual, possible = transitions.count_lags(4, pool_items, recall_items)
>>> actual
-3    0
-2    2
-1    0
 0    0
 1    1
 2    0
 3    0
dtype: int64
>>> possible
-3    1
-2    2
-1    2
 0    0
 1    1
 2    0
 3    0
dtype: int64
```

`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.`**`percentile_rank`**(*actual*, *possible*)

Get percentile rank of a score compared to possible scores.

**Parameters**

- **actual** (*float*) – Score to be ranked. Generally a distance score.
- **possible** (*numpy.ndarray or list*) – Possible scores to be compared to.

**Returns** **rank** – Rank scaled to range from 0 (low score) to 1 (high score).

**Return type** float

### Examples

```
>>> from psifr import transitions
>>> actual = 3
>>> possible = [1, 2, 2, 2, 3]
>>> transitions.percentile_rank(actual, possible)
1.0
```

`psifr.transitions.`**`rank_distance`**(*distances*, *pool_items*, *recall_items*, *pool_index*, *recall_index*, *pool_test=None*, *recall_test=None*, *test=None*)

Calculate percentile rank of transition distances.

**Parameters**

- **distances** (*numpy.array*) – Items x items matrix of pairwise distances or similarities.
- **pool_items** (*list of list*) – Unique item codes for each item in the pool available for recall.
- **recall_items** (*list of list*) – Unique item codes of recalled items.
- **pool_index** (*list of list*) – Index of each item in the distances matrix.
- **recall_index** (*list of list*) – Index of each recalled item.
- **pool_test** (*list of list, optional*) – Test value for each item in the pool.
- **recall_test** (*list of list, optional*) – Test value for each recalled item.
- **test** (*callable*) – Called as test(prev, curr) or test(prev, poss) to screen actual and possible transitions, respectively.

**Returns** **rank** – Distance percentile rank for each included transition. The rank is 0 if the distance was the largest of the available transitions, and 1 if the distance was the smallest. Ties are assigned to the average percentile rank.

**Return type** list

**See also:**

*count_distance()* Count transitions within distance bins.

## Examples

```
>>> from psifr import transitions
>>> distances = np.array([[0, 1, 2, 2], [1, 0, 2, 2], [2, 2, 0, 3], [2, 2, 3, 0]])
>>> edges = np.array([0.5, 1.5, 2.5, 3.5])
>>> pool_items = [[1, 2, 3, 4]]
>>> recall_items = [[4, 2, 3, 1]]
>>> pool_index = [[0, 1, 2, 3]]
>>> recall_index = [[3, 1, 2, 0]]
>>> transitions.rank_distance(
...     distances, pool_items, recall_items, pool_index, recall_index
... )
[0.75, 0.0, nan]
```

psifr.transitions.**rank_lags**(*pool_items*, *recall_items*, *pool_label=None*, *recall_label=None*, *pool_test=None*, *recall_test=None*, *test=None*)

Calculate rank of absolute lag for free recall lists.

**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_label** (*list, optional*) – List of the positions to use for calculating lag. Default is to use *pool_items*.
- **recall_label** (*list, optional*) – List of position labels in recall order. Default is to use *recall_items*.
- **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.

**Returns** **rank** – Absolute lag percentile rank for each included transition. The rank is 0 if the lag was the most distant of the available transitions, and 1 if the lag was the closest. Ties are assigned to the average percentile rank.

**Return type** list

**See also:**

`count_lags()` Count actual and possible serial position lags.

## Examples

```
>>> from psifr import transitions
>>> pool_items = [[1, 2, 3, 4]]
>>> recall_items = [[4, 2, 3, 1]]
>>> transitions.rank_lags(pool_items, recall_items)
[0.5, 0.5, nan]
```

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.

### Examples

```
>>> pool = [1, 2, 3, 4, 5, 6]
>>> recs = [6, 2, 3, 6, 1, 4]
>>> masker = transitions_masker(
...     pool_items=pool, recall_items=recs, pool_output=pool, recall_output=recs
... )
>>> 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]
```

## 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.

**Parameters** **`list_labels`** (*`list or numpy.ndarray`*) – Position labels that define the blocks.

**Returns** **block** – Block index of each position.

**Return type** numpy.ndarray

### Examples

```
>>> import numpy as np
>>> from psifr import fr
>>> list_labels = [2, 2, 3, 3, 3, 1, 1]
>>> fr.block_index(list_labels)
array([1, 1, 2, 2, 2, 3, 3])
```

`psifr.fr.`**`category_crp`**`(`*df*, *category_key*, *item_query=None*, *test_key=None*, *test=None*`)`
Conditional response probability of within-category transitions.

**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.
- **`category_key`** (*`str`*) – Name of column with category labels.
- **`item_query`** (*`str, optional`*) – Query string to select items to include in the pool of possible recalls to be examined. See *pandas.DataFrame.query* for allowed format.

- **test_key** (*str, optional*) – Name of 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.

**Returns**

**results** – Has fields:

**subject** [hashable] Results are separated by each subject.

**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

## Examples

```
>>> from psifr import fr
>>> raw = fr.sample_data('Morton2013')
>>> data = fr.merge_free_recall(raw, study_keys=['category'])
>>> cat_crp = fr.category_crp(data, 'category')
>>> cat_crp.head()
              prob  actual  possible
subject
1         0.801147     419       523
2         0.733456     399       544
3         0.763158     377       494
4         0.814882     449       551
5         0.877273     579       660
```

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.

### Examples

```
>>> from psifr import fr
>>> import pandas as pd
>>> raw = pd.DataFrame(
...     {'subject': [1, 1], 'list': [1, 1], 'position': [1, 2], 'item': ['a', 'b
↪']}
... )
>>> fr.check_data(raw)
Traceback (most recent call last):
  File "psifr/fr.py", line 253, in check_data
    assert col in df.columns, f'Required column {col} is missing.'
AssertionError: Required column trial_type is missing.
```

psifr.fr.**distance_crp**(*df*, *index_key*, *distances*, *edges*, *centers=None*, *count_unique=False*, *item_query=None*, *test_key=None*, *test=None*)

Conditional response probability by distance bin.

**Parameters**

- **df** (*pandas.DataFrame*) – Merged free recall data.
- **index_key** (*str*) – Name of column containing the index of each item in the *distances* matrix.
- **distances** (*numpy.array*) – Items x items matrix of pairwise distances or similarities.
- **edges** (*array-like*) – Edges of bins to apply to the distances.
- **centers** (*array-like, optional*) – Centers to label each bin with. If not specified, the center point between edges will be used.
- **count_unique** (*bool, optional*) – If true, possible transitions to a given distance bin will only count once for a given transition.
- **item_query** (*str, optional*) – Query string to select items to include in the pool of possible recalls to be examined. See *pandas.DataFrame.query* for allowed format.
- **test_key** (*str, optional*) – Name of 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.

**Returns**

**crp** – Has fields:

**subject** [hashable] Results are separated by each subject.

**bin** [int] Distance bin.

**prob** [float] Probability of each distance bin.

**actual** [int] Total of actual transitions for each distance bin.

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

**Return type** pandas.DataFrame

**See also:**

*pool_index()* Given a list of presented items and an item pool, look up the pool index of each item.

*distance_rank()* Calculate rank of transition distances.

### Examples

```
>>> from scipy.spatial.distance import squareform
>>> from psifr import fr
>>> raw = fr.sample_data('Morton2013')
>>> data = fr.merge_free_recall(raw)
>>> items, distances = fr.sample_distances('Morton2013')
>>> data['item_index'] = fr.pool_index(data['item'], items)
>>> edges = np.percentile(squareform(distances), np.linspace(1, 99, 10))
>>> fr.distance_crp(data, 'item_index', distances, edges)
                              bin      prob  actual  possible
subject center
1       0.467532  (0.352, 0.583]  0.085456     151      1767
        0.617748  (0.583, 0.653]  0.067916      87      1281
        0.673656  (0.653, 0.695]  0.062500      65      1040
        0.711075  (0.695, 0.727]  0.051836      48       926
        0.742069  (0.727, 0.757]  0.050633      44       869
...                          ...       ...     ...       ...
47      0.742069  (0.727, 0.757]  0.062822      61       971
        0.770867  (0.757, 0.785]  0.030682      27       880
        0.800404  (0.785, 0.816]  0.040749      37       908
        0.834473  (0.816, 0.853]  0.046651      39       836
        0.897275  (0.853, 0.941]  0.028868      25       866

[360 rows x 4 columns]
```

psifr.fr.**distance_rank**(*df*, *index_key*, *distances*, *item_query=None*, *test_key=None*, *test=None*)
Calculate rank of transition distances in free recall lists.

**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.
- **index_key** (*str*) – Name of column containing the index of each item in the *distances* matrix.
- **distances** (*numpy.array*) – Items x items matrix of pairwise distances or similarities.
- **item_query** (*str, optional*) – Query string to select items to include in the pool of possible recalls to be examined. See *pandas.DataFrame.query* for allowed format.
- **test_key** (*str, optional*) – Name of 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.

**Returns** **stat** – Has fields 'subject' and 'rank'.

**Return type** pandas.DataFrame

**See also:**

*pool_index()* Given a list of presented items and an item pool, look up the pool index of each item.

`distance_crp()` Conditional response probability by distance bin.

### Examples

```
>>> from scipy.spatial.distance import squareform
>>> from psifr import fr
>>> raw = fr.sample_data('Morton2013')
>>> data = fr.merge_free_recall(raw)
>>> items, distances = fr.sample_distances('Morton2013')
>>> data['item_index'] = fr.pool_index(data['item'], items)
>>> dist_rank = fr.distance_rank(data, 'item_index', distances)
>>> dist_rank.head()
             rank
subject
1        0.635571
2        0.571457
3        0.627282
4        0.637596
5        0.646181
```

psifr.fr.**filter_data**(*data*, *subjects=None*, *lists=None*, *trial_type=None*, *positions=None*, *inputs=None*, *outputs=None*)

Filter data to get a subset of trials.

**Parameters**

- **data** (*pandas.DataFrame*) – Raw or merged data to filter.
- **subjects** (*hashable or list of hashable*) – Subject or subjects to include.
- **lists** (*hashable or list of hashable*) – List or lists to include.
- **trial_type** (*{'study', 'recall'}*) – Trial type to include.
- **positions** (*int or list of int*) – Position or positions to include.
- **inputs** (*int or list of int*) – Input position or positions to include.
- **outputs** (*int or list of int*) – Output position or positions to include.

**Returns** **filtered** – The filtered subset of data.

**Return type** pandas.DataFrame

### Examples

```
>>> from psifr import fr
>>> subjects_list = [1, 1, 2, 2]
>>> study_lists = [['a', 'b'], ['c', 'd'], ['e', 'f'], ['g', 'h']]
>>> recall_lists = [['b'], ['d', 'c'], ['f', 'e'], []]
>>> raw = fr.table_from_lists(subjects_list, study_lists, recall_lists)
>>> fr.filter_data(raw, subjects=1, trial_type='study')
   subject  list trial_type  position item
0        1     1      study         1    a
1        1     1      study         2    b
3        1     2      study         1    c
4        1     2      study         2    d
```

```
>>> data = fr.merge_free_recall(raw)
>>> fr.filter_data(data, subjects=2)
   subject  list item  input  output  study  recall  repeat  intrusion  prior_
↪list  prior_input
4        2     1    e      1     2.0   True    True       0      False        ␣
↪NaN          NaN
5        2     1    f      2     1.0   True    True       0      False        ␣
↪NaN          NaN
6        2     2    g      1     NaN   True   False       0      False        ␣
↪NaN          NaN
7        2     2    h      2     NaN   True   False       0      False        ␣
↪NaN          NaN
```

psifr.fr.**lag_crp**(*df*, *lag_key='input'*, *count_unique=False*, *item_query=None*, *test_key=None*, *test=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. Must have fields: subject, list, input, output, recalled. Input position must be defined such that the first serial position is 1, not 0.
- **lag_key** (*str, optional*) – Name of column to use when calculating lag between recalled items. Default is to calculate lag based on input position.
- **count_unique** (*bool, optional*) – If true, possible transitions of the same lag will only be incremented once per transition.
- **item_query** (*str, optional*) – Query string to select items to include in the pool of possible recalls to be examined. See *pandas.DataFrame.query* for allowed format.
- **test_key** (*str, optional*) – Name of 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.

**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

**See also:**

***lag_rank()*** Rank of the absolute lags in recall sequences.

### Examples

```
>>> from psifr import fr
>>> raw = fr.sample_data('Morton2013')
>>> data = fr.merge_free_recall(raw)
>>> fr.lag_crp(data)
                     prob  actual  possible
subject lag
1       -23.0  0.020833       1        48
        -22.0  0.035714       3        84
        -21.0  0.026316       3       114
        -20.0  0.024000       3       125
        -19.0  0.014388       2       139
...                 ...     ...       ...
47       19.0  0.061224       3        49
         20.0  0.055556       2        36
         21.0  0.045455       1        22
         22.0  0.071429       1        14
         23.0  0.000000       0         6

[1880 rows x 3 columns]
```

`psifr.fr.`**`lag_rank`**(*df*, *item_query=None*, *test_key=None*, *test=None*)

Calculate rank of the absolute lags in free recall lists.

**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.
- **item_query** (*str, optional*) – Query string to select items to include in the pool of possible recalls to be examined. See *pandas.DataFrame.query* for allowed format.
- **test_key** (*str, optional*) – Name of 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.

**Returns** **stat** – Has fields ‘subject’ and ‘rank’.

**Return type** pandas.DataFrame

**See also:**

***lag_crp()*** Conditional response probability by input lag.

### Examples

```
>>> from psifr import fr
>>> raw = fr.sample_data('Morton2013')
>>> data = fr.merge_free_recall(raw)
>>> lag_rank = fr.lag_rank(data)
>>> lag_rank.head()
             rank
subject
1        0.610953
2        0.635676
3        0.612607
4        0.667090
5        0.643923
```

`psifr.fr.`**`merge_free_recall`**(*data*, ***kwargs*)

Score free recall data by matching up study and recall events.

**Parameters**

- **data** (*pandas.DataFrame*) – Free recall data in Psifr format. Must have subject, list, trial_type, position, and item columns.
- **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.

**study** [bool] True for rows corresponding to a unique study event.

**recall** [bool] True for rows corresponding to a unique 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.

**prior_list** [int] For prior-list intrusions, the list the item was presented.

**prior_position** [int] For prior-list intrusions, the position the item was presented.

**Return type** pandas.DataFrame

**See also:**

***merge_lists()*** Flexibly merge study events with recall events. Useful for recall phases that don't match the typical free recall setup, like final free recall of all lists.

### Examples

```
>>> from psifr import fr
>>> study = [['absence', 'hollow'], ['fountain', 'piano']]
>>> recall = [['absence'], ['piano', 'hollow']]
>>> raw = fr.table_from_lists([1, 1], study, recall)
>>> raw
   subject  list trial_type  position      item
0        1     1      study         1   absence
1        1     1      study         2    hollow
2        1     1     recall         1   absence
3        1     2      study         1  fountain
4        1     2      study         2     piano
5        1     2     recall         1     piano
6        1     2     recall         2    hollow
```

Score the data to create a table with matched study and recall events.

```
>>> data = fr.merge_free_recall(raw)
>>> data
   subject  list      item  input  output  study  recall  repeat  intrusion ␣
↪prior_list  prior_input
0        1     1   absence    1.0     1.0   True    True       0      False      ␣
↪  NaN            NaN
1        1     1    hollow    2.0     NaN   True   False       0      False      ␣
↪  NaN            NaN
2        1     2  fountain    1.0     NaN   True   False       0      False      ␣
↪  NaN            NaN
3        1     2     piano    2.0     1.0   True    True       0      False      ␣
↪  NaN            NaN
4        1     2    hollow    NaN     2.0  False    True       0       True      ␣
↪  1.0            2.0
```

You can also include non-standard columns. Information that only applies to study events (here, the encoding task used) can be indicated using the `study_keys` input.

```
>>> raw['task'] = np.array([1, 2, np.nan, 2, 1, np.nan, np.nan])
>>> fr.merge_free_recall(raw, study_keys=['task'])
   subject  list      item  input  output  study  recall  repeat  intrusion  task␣
↪ prior_list  prior_input
0        1     1   absence    1.0     1.0   True    True       0      False   1.0␣
↪        NaN            NaN
1        1     1    hollow    2.0     NaN   True   False       0      False   2.0␣
↪        NaN            NaN
2        1     2  fountain    1.0     NaN   True   False       0      False   2.0␣
↪        NaN            NaN
3        1     2     piano    2.0     1.0   True    True       0      False   1.0␣
↪        NaN            NaN
4        1     2    hollow    NaN     2.0  False    True       0       True   NaN␣
↪        1.0            2.0
```

Information that only applies to recall onsets (here, the time in seconds after the start of the recall phase that a recall attempt was made), can be indicated using the `recall_keys` input.

```
>>> raw['onset'] = np.array([np.nan, np.nan, 1.1, np.nan, np.nan, 1.4, 3.8])
>>> fr.merge_free_recall(raw, recall_keys=['onset'])
   subject  list      item  input  output  study  recall  repeat  intrusion ␣
↪onset  prior_list  prior_input
0        1     1   absence    1.0     1.0   True    True       0      False    1.
↪1        NaN          NaN
1        1     1    hollow    2.0     NaN   True   False       0      False   ␣
↪NaN        NaN          NaN
2        1     2  fountain    1.0     NaN   True   False       0      False   ␣
↪NaN        NaN          NaN
3        1     2     piano    2.0     1.0   True    True       0      False    1.
↪4        NaN          NaN
4        1     2    hollow    NaN     2.0  False    True       0       True    3.
↪8        1.0          2.0
```

Use `list_keys` to indicate columns that apply to both study and recall events. If `list_keys` do not match for a pair of study and recall events, they will not be matched in the output.

```
>>> raw['condition'] = np.array([1, 1, 1, 2, 2, 2, 2])
>>> fr.merge_free_recall(raw, list_keys=['condition'])
   subject  list      item  input  output  study  recall  repeat  intrusion ␣
↪condition  prior_list  prior_input
0        1     1   absence    1.0     1.0   True    True       0      False        ␣
↪    1          NaN          NaN
1        1     1    hollow    2.0     NaN   True   False       0      False        ␣
↪    1          NaN          NaN
2        1     2  fountain    1.0     NaN   True   False       0      False        ␣
↪    2          NaN          NaN
3        1     2     piano    2.0     1.0   True    True       0      False        ␣
↪    2          NaN          NaN
4        1     2    hollow    NaN     2.0  False    True       0       True        ␣
↪    2          1.0          2.0
```

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.

**study** [bool] True for rows corresponding to a unique study event.

**recall** [bool] True for rows corresponding to a unique 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

**See also:**

*`merge_free_recall()`* Score standard free recall data.

### Examples

```
>>> import pandas as pd
>>> from psifr import fr
>>> study = pd.DataFrame(
...     {'subject': [1, 1], 'list': [1, 1], 'position': [1, 2], 'item': ['a', 'b']}
... )
>>> recall = pd.DataFrame(
...     {'subject': [1], 'list': [1], 'position': [1], 'item': ['b']}
... )
>>> fr.merge_lists(study, recall)
   subject  list item  input  output  study  recall  repeat  intrusion
0        1     1    a      1     NaN   True   False       0      False
1        1     1    b      2     1.0   True    True       0      False
```

`psifr.fr.`**`pli_list_lag`**(*df*, *max_lag*)
List lag of prior-list intrusions.

**Parameters**

- **df** (*`pandas.DataFrame`*) – Merged study and recall data. See merge_free_recall. Must have fields: subject, list, intrusion, prior_list. Lists must be numbered starting from 1 and all lists must be included.
- **max_lag** (*`int`*) – Maximum list lag to consider. The intial `max_lag` lists for each subject will be excluded so that all considered lags are possible for all included lists.

**Returns results** – For each subject and list lag, the proportion of intrusions at that lag, in the `results['prob']` column.

**Return type** pandas.DataFrame

### Examples

```
>>> from psifr import fr
>>> raw = fr.sample_data('Morton2013')
>>> data = fr.merge_free_recall(raw)
>>> fr.pli_list_lag(data, 3)
                  count  per_list      prob
subject list_lag
1       1             7  0.155556  0.259259
        2             5  0.111111  0.185185
        3             0  0.000000  0.000000
2       1             9  0.200000  0.191489
        2             2  0.044444  0.042553
...                 ...       ...       ...
46      2             1  0.022222  0.100000
        3             0  0.000000  0.000000
47      1             5  0.111111  0.277778
        2             1  0.022222  0.055556
        3             0  0.000000  0.000000

[120 rows x 3 columns]
```

psifr.fr.**plot_distance_crp**(*crp*, *min_samples=None*, ***facet_kws*)
Plot response probability by distance bin.

**Parameters**

- **crp** (*pandas.DataFrame*) – Results from *fr.distance_crp*.
- **min_samples** (*int*) – Minimum number of samples a bin must have per subject to include in the plot.
- ****facet_kws** – Additional inputs to pass to *seaborn.relplot*.

psifr.fr.**plot_lag_crp**(*recall*, *max_lag=5*, *split=True*, ***facet_kws*)
Plot conditional response probability by lag.

Additional arguments are passed to seaborn.FacetGrid.

**Parameters**

- **recall** (*pandas.DataFrame*) – Results from calling *lag_crp*.
- **max_lag** (*int*) – Maximum absolute lag to plot.
- **split** (*bool, optional*) – If true, will plot as two separate lines with a gap at lag 0.

psifr.fr.**plot_raster**(*df*, *hue='input'*, *palette=None*, *marker='s'*, *intrusion_color=None*, *orientation='horizontal'*, *length=6*, *aspect=None*, *legend='auto'*, ***facet_kws*)
Plot recalls in a raster plot.

**Parameters**

- **df** (*pandas.DataFrame*) – Scored free recall data.
- **hue** (*str or None, optional*) – Column to use to set marker color.
- **palette** (*optional*) – Palette specification supported by Seaborn.
- **marker** (*str, optional*) – Marker code supported by Seaborn.
- **intrusion_color** (*optional*) – Color of intrusions.

- **orientation** (*{'horizontal', 'vertical'}, optional*) – Whether lists should be stacked horizontally or vertically in the plot.
- **length** (*float, optional*) – Size of the plot dimension along which list varies.
- **aspect** (*float, optional*) – Aspect ratio of plot for lists over items.
- **legend** (*str, optional*) – Legend setting. See seaborn.scatterplot for details.
- **facet_kws** (*optional*) – Additional key words to pass to seaborn.FacetGrid.

psifr.fr.**plot_spc**(*recall*, ***facet_kws*)

Plot a serial position curve.

Additional arguments are passed to seaborn.relplot.

**Parameters** **recall** (*pandas.DataFrame*) – Results from calling *spc*.

psifr.fr.**plot_swarm_error**(*data*, *x=None*, *y=None*, *swarm_color=None*, *swarm_size=5*, *point_color='k'*, ***facet_kws*)

Plot points as a swarm plus mean with error bars.

**Parameters**

- **data** (*pandas.DataFrame*) – DataFrame with statistics to plot.
- **x** (*str*) – Name of variable to plot on x-axis.
- **y** (*str*) – Name of variable to plot on y-axis.
- **swarm_color** – Color for swarm plot points. May use any specification supported by seaborn.
- **swarm_size** (*float*) – Size of swarm plot points.
- **point_color** – Color for the point plot (error bars).
- **facet_kws** – Additional keywords for the FacetGrid.

psifr.fr.**pnr**(*df*, *item_query=None*, *test_key=None*, *test=None*)

Probability of recall by serial position and output position.

Calculate probability of Nth recall, where N is each output position. Invalid recalls (repeats and intrusions) are ignored and not counted toward output position.

**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, study, recall. Input position must be defined such that the first serial position is 1, not 0.
- **item_query** (*str, optional*) – Query string to select items to include in the pool of possible recalls to be examined. See *pandas.DataFrame.query* for allowed format.
- **test_key** (*str, optional*) – Name of 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.

**Returns** **prob** – Analysis results. Has fields: subject, output, input, prob, actual, possible. The prob column for output x and input y indicates the probability of recalling input position y at output position x. The actual and possible columns give the raw tallies for how many times an event actually occurred and how many times it was possible given the recall sequence.

**Return type** pandas.DataFrame

**See also:**

*plot_spc()* Plot recall probability as a function of serial position.

*spc()* Overall recall probability by serial position.

### Examples

```
>>> from psifr import fr
>>> raw = fr.sample_data('Morton2013')
>>> data = fr.merge_free_recall(raw)
>>> fr.pnr(data)
                               prob  actual  possible
subject output input
1       1      1       0.000000       0        48
               2       0.020833       1        48
               3       0.000000       0        48
               4       0.000000       0        48
               5       0.000000       0        48
...                         ...     ...       ...
47      24     20           NaN       0         0
               21           NaN       0         0
               22           NaN       0         0
               23           NaN       0         0
               24           NaN       0         0

[23040 rows x 3 columns]
```

psifr.fr.**pool_index**(*trial_items*, *pool_items_list*)

Get the index of each item in the full pool.

**Parameters**

- **trial_items** (*pandas.Series*) – The item presented on each trial.
- **pool_items_list** (*list or numpy.ndarray*) – List of items in the full pool.

**Returns** **item_index** – Index of each item in the pool. Trials with items not in the pool will be <NA>.

**Return type** pandas.Series

### Examples

```
>>> import pandas as pd
>>> from psifr import fr
>>> trial_items = pd.Series(['b', 'a', 'z', 'c', 'd'])
>>> pool_items_list = ['a', 'b', 'c', 'd', 'e', 'f']
>>> fr.pool_index(trial_items, pool_items_list)
0        1
1        0
2     <NA>
3        2
4        3
dtype: Int64
```

psifr.fr.**reset_list**(*df*)

Reset list index in a DataFrame.

**Parameters** **df** (*pandas.DataFrame*) – Raw or merged data. Must have subject and list fields.

**Returns** Data with a renumbered list field, starting from 1.

**Return type** pandas.DataFrame

## Examples

```
>>> from psifr import fr
>>> subjects_list = [1, 1]
>>> study_lists = [['a', 'b'], ['c', 'd']]
>>> recall_lists = [['b'], ['c', 'd']]
>>> list_nos = [3, 4]
>>> raw = fr.table_from_lists(subjects_list, study_lists, recall_lists,␣
↪lists=list_nos)
>>> raw
   subject  list trial_type  position item
0        1     3      study         1    a
1        1     3      study         2    b
2        1     3     recall         1    b
3        1     4      study         1    c
4        1     4      study         2    d
5        1     4     recall         1    c
6        1     4     recall         2    d
```

```
>>> fr.reset_list(raw)
   subject  list trial_type  position item
0        1     1      study         1    a
1        1     1      study         2    b
2        1     1     recall         1    b
3        1     2      study         1    c
4        1     2      study         2    d
5        1     2     recall         1    c
6        1     2     recall         2    d
```

`psifr.fr.`**`sample_data`**(*study*)
Read sample data.

`psifr.fr.`**`sample_distances`**(*study*)
Read sample distances.

`psifr.fr.`**`spc`**(*df*)
Serial position curve.

**Parameters** **df** (*pandas.DataFrame*) – Merged study and recall data. See merge_lists.

**Returns**

**recall** – Index includes:

**subject** [hashable] Subject identifier.

**input** [int] Serial position in the list.

Values are:

**recall** [float] Recall probability for each serial position.

**Return type** pandas.Series

**See also:**

`plot_spc()` Plot serial position curve results.

`pnr()` Probability of nth recall.

### Examples

```
>>> from psifr import fr
>>> raw = fr.sample_data('Morton2013')
>>> data = fr.merge_free_recall(raw)
>>> fr.spc(data)
                 recall
subject input
1       1.0    0.541667
        2.0    0.458333
        3.0    0.625000
        4.0    0.333333
        5.0    0.437500
...                 ...
47      20.0   0.500000
        21.0   0.770833
        22.0   0.729167
        23.0   0.895833
        24.0   0.958333

[960 rows x 1 columns]
```

psifr.fr.**split_lists**(*frame*, *phase*, *keys=None*, *names=None*, *item_query=None*, *as_list=False*)

Convert free recall data from one phase to split format.

**Parameters**

- **frame** (*pandas.DataFrame*) – Free recall data with separate study and recall events.
- **phase** (*{'study', 'recall', 'raw'}*) – Phase of recall to split. If 'raw', all trials will be included.
- **keys** (*list of str, optional*) – Data columns to include in the split data. If not specified, all columns will be included.
- **names** (*list of str, optional*) – Name for each column in the returned split data. Default is to use the same names as the input columns.
- **item_query** (*str, optional*) – Query string to select study trials to include. See *pandas.DataFrame.query* for allowed format.
- **as_list** (*bool, optional*) – If true, each column will be output as a list; otherwise, outputs will be numpy.ndarray.

**Returns** **split** – Data in split format. Each included column will be a key in the dictionary, with a list of either numpy.ndarray (default) or lists, containing the values for that column.

**Return type** dict of str: list

**See also:**

`table_from_lists()` Convert list-format data to a table.

### Examples

```
>>> from psifr import fr
>>> study = [['absence', 'hollow'], ['fountain', 'piano']]
>>> recall = [['absence'], ['piano', 'fountain']]
>>> raw = fr.table_from_lists([1, 1], study, recall)
>>> data = fr.merge_free_recall(raw)
>>> data
   subject  list      item  input  output  study  recall  repeat  intrusion ␣
↪prior_list  prior_input
0        1     1   absence      1     1.0   True    True       0      False      ␣
↪   NaN          NaN
1        1     1    hollow      2     NaN   True   False       0      False      ␣
↪   NaN          NaN
2        1     2  fountain      1     2.0   True    True       0      False      ␣
↪   NaN          NaN
3        1     2     piano      2     1.0   True    True       0      False      ␣
↪   NaN          NaN
```

Get study events split by list, just including the list and item fields.

```
>>> fr.split_lists(data, 'study', keys=['list', 'item'], as_list=True)
{'list': [[1, 1], [2, 2]], 'item': [['absence', 'hollow'], ['fountain', 'piano']]}
```

Export recall events, split by list.

```
>>> fr.split_lists(data, 'recall', keys=['item'], as_list=True)
{'item': [['absence'], ['piano', 'fountain']]}
```

Raw events (i.e., events that haven't been scored) can also be exported to list format.

```
>>> fr.split_lists(raw, 'raw', keys=['position'])
{'position': [array([1, 2, 1]), array([1, 2, 1, 2])]}
```

`psifr.fr.`**`table_from_lists`**(*subjects*, *study*, *recall*, *lists=None*, ***kwargs*)

Create table format data from list format data.

**Parameters**

- **`subjects`** (*`list of hashable`*) – Subject identifier for each list.
- **`study`** (*`list of list of hashable`*) – List of items for each study list.
- **`recall`** (*`list of list of hashable`*) – List of recalled items for each study list.
- **`lists`** (*`list of hashable, optional`*) – List of list numbers. If not specified, lists for each subject will be numbered sequentially starting from one.

**Returns** **data** – Data in table format.

**Return type** pandas.DataFrame

**See also:**

*`split_lists()`* Split a table into list format.

### Examples

```
>>> from psifr import fr
>>> subjects_list = [1, 1, 2, 2]
>>> study_lists = [['a', 'b'], ['c', 'd'], ['e', 'f'], ['g', 'h']]
>>> recall_lists = [['b'], ['d', 'c'], ['f', 'e'], []]
>>> fr.table_from_lists(subjects_list, study_lists, recall_lists)
    subject  list trial_type  position item
0         1     1      study         1    a
1         1     1      study         2    b
2         1     1     recall         1    b
3         1     2      study         1    c
4         1     2      study         2    d
5         1     2     recall         1    d
6         1     2     recall         2    c
7         2     1      study         1    e
8         2     1      study         2    f
9         2     1     recall         1    f
10        2     1     recall         2    e
11        2     2      study         1    g
12        2     2      study         2    h
```

```
>>> subjects_list = [1, 1]
>>> study_lists = [['a', 'b'], ['c', 'd']]
>>> recall_lists = [['b'], ['d', 'c']]
>>> col1 = ([[1, 2], [1, 2]], [[2], [2, 1]])
>>> col2 = ([[1, 1], [2, 2]], None)
>>> fr.table_from_lists(subjects_list, study_lists, recall_lists, col1=col1,␣
↪col2=col2)
   subject  list trial_type  position item  col1  col2
0        1     1      study         1    a     1   1.0
1        1     1      study         2    b     2   1.0
2        1     1     recall         1    b     2   NaN
3        1     2      study         1    c     1   2.0
4        1     2      study         2    d     2   2.0
5        1     2     recall         1    d     2   NaN
6        1     2     recall         2    c     1   NaN
```

# PYTHON MODULE INDEX

p
psifr.fr, 21
psifr.transitions, 15

# INDEX

B

block_index() (*in module psifr.fr*), 21

C

category_crp() (*in module psifr.fr*), 21
check_data() (*in module psifr.fr*), 22
count_category() (*in module psifr.transitions*), 15
count_distance() (*in module psifr.transitions*), 16
count_lags() (*in module psifr.transitions*), 17
count_pairs() (*in module psifr.transitions*), 18

D

distance_crp() (*in module psifr.fr*), 23
distance_rank() (*in module psifr.fr*), 24

F

filter_data() (*in module psifr.fr*), 25

L

lag_crp() (*in module psifr.fr*), 26
lag_rank() (*in module psifr.fr*), 27

M

merge_free_recall() (*in module psifr.fr*), 28
merge_lists() (*in module psifr.fr*), 30

P

percentile_rank() (*in module psifr.transitions*), 18
pli_list_lag() (*in module psifr.fr*), 31
plot_distance_crp() (*in module psifr.fr*), 32
plot_lag_crp() (*in module psifr.fr*), 32
plot_raster() (*in module psifr.fr*), 32
plot_spc() (*in module psifr.fr*), 33
plot_swarm_error() (*in module psifr.fr*), 33
pnr() (*in module psifr.fr*), 33
pool_index() (*in module psifr.fr*), 34
psifr.fr (*module*), 21
psifr.transitions (*module*), 15

R

rank_distance() (*in module psifr.transitions*), 18
rank_lags() (*in module psifr.transitions*), 19
reset_list() (*in module psifr.fr*), 34

S

sample_data() (*in module psifr.fr*), 35
sample_distances() (*in module psifr.fr*), 35
spc() (*in module psifr.fr*), 35
split_lists() (*in module psifr.fr*), 36

T

table_from_lists() (*in module psifr.fr*), 37
transitions_masker() (*in module psifr.transitions*), 20