



Turn-Taking Measurement Tools Documentation

Release 0.91

Peter Raffensperger

November 16, 2012

CONTENTS

1	Features	3
2	Dependencies	5
3	Contents	7
3.1	Turn-Taking Measurement Tools module functions	7
3.2	License	11
	Index	13

Do you need to measure the quantity of turn-taking present in the behaviour of a group of agents? This is the software library for you. Turn-Taking Measurement Tools is a software library that implements a quantitative metric for turn-taking. A simple turn-taking metric was developed by Peter Raffensperger, Russell Webb, Phillip Bones and Allan McInnes at the University of Canterbury in 2009-2012 for research into multi-agent systems and emergent communication. You may find additional applications for our turn-taking metric in conversational analysis, spoken dialog systems, medium access control in computer networks, biology and other areas. The current implementation is in Python.

If you use Turn-Taking Measurement Tools in an academic project, we invite you to reference the original publication: Raffensperger, P. A., Webb, R. Y., Bones, P. J., and McInnes, A. I. (2012). A simple metric for turn-taking in emergent communication. *Adaptive Behavior*, 20(2):104-116.

Paper abstract for “A simple metric for turn-taking in emergent communication”: To facilitate further research in emergent turn-taking, we propose a metric for evaluating the extent to which agents take turns using a shared resource. Our measure reports a turn-taking value for a particular time and a particular timescale, or “resolution,” in a way that matches intuition. We describe how to evaluate the results of simulations where turn-taking may or may not be present and analyze the apparent turn-taking that could be observed between random independent agents. We illustrate the use of our turn-taking metric by reinterpreting previous work on turn-taking in emergent communication and by analyzing a recorded human conversation.

FEATURES

- Functions for measuring the quantity of turn-taking at different time resolutions in binary-valued usage attempt sequences of arbitrary lengths with any number of agents
- Functions for fairness and efficient metrics
- Estimation routines for the turn-taking of random agents
- Python doc strings for all key functions
- Examples (run `examples/main.py`)

DEPENDENCIES

- numpy
- matplotlib or pyGnuplot for examples (optional)

CONTENTS

3.1 Turn-Taking Measurement Tools module functions

`turntakingmeasurementtools.allocation(U, t, r)`

Compute the quantity of turn-taking present in the usage attempt sequence U , at time t and resolution r , using the block of time steps $[t, t+r-1]$.

Parameters:

- U – a binary valued usage attempt sequence of shape (A, l) , where A is the number of agents and l is the length. (As either a numpy array or a list of lists)
- t – the time at which to calculate the agent allocations
- r – the ‘resolution’ or window length, with $l \leq t+r$

Return value: A vector of length A representing turn allocations of each agent

See Eq. 1 in: Raffensperger, P. A., Webb, R. Y., Bones, P. J., and McInnes, A. I. (2012). A simple metric for turn-taking in emergent communication. *Adaptive Behavior*, 20(2):104-116.

`turntakingmeasurementtools.clean_up_tt_distribution_files()`

Deletes files in the local directory that are of extension type `TT_DISTRIBUTION_FILE_EXTENSION` and have names starting with `TT_DISTRIBUTION_FILE_PREFIX`

`turntakingmeasurementtools.efficiency(U, t, r)`

Compute the efficiency of the turn allocation in the usage attempt sequence U , at time t and resolution r , using the block of time steps $[t, t+r-1]$.

Parameters:

- U – a binary valued usage attempt sequence of shape (A, l) , where A is the number of agents and l is the length. (As either a numpy array or a list of lists)
- t – the time at which to calculate the efficiency measure
- r – the ‘resolution’ or window length, with $l \leq t+r$

Return value: A real value in $[0, 1]$ representing the efficiency of the turn allocation in U in the range $[t, t+r-1]$

See Eq. 3 in: Raffensperger, P. A., Webb, R. Y., Bones, P. J., and McInnes, A. I. (2012). A simple metric for turn-taking in emergent communication. *Adaptive Behavior*, 20(2):104-116.

`turntakingmeasurementtools.estimate_probability_of_tt_due_to_chance` (*numAgents*, *r*, *sample-size*, *target_tt_value*, *persistentData=False*, *verbose=False*)

Estimate the probability that a particular turn-taking value is produced by random processes, assuming the worst case usage attempt probabilities for all the agents

Parameters:

- *numAgents* – the desired number of probabilistic agents to include
- *r* – the turn-taking resolution
- *samplesize* – the number of independent turn-taking value samples to include (1000000 is a good place to start)

Keyword arguments:

- *persistentData* – set to True to save the turn-taking value distribution to disk for future use
- *verbose* – set to True

See Section 3 of: Raffensperger, P. A., Webb, R. Y., Bones, P. J., and McInnes, A. I. (2012). A simple metric for turn-taking in emergent communication. *Adaptive Behavior*, 20(2):104-116.

`turntakingmeasurementtools.estimate_tt_mean_and_variance` (*numAgents*, *r*, *pu*, *sample-size=100000*, *persistentData=False*)

Estimate the mean and variance of the turn-taking value for a group of random agents with the given usage attempt probability.

Parameters:

- *numAgents* – the desired number of probabilistic agents to include
- *r* – the turn-taking resolution
- *pu* – the probability that a bit in each usage attempt sequence will be 1

Keyword arguments:

- *samplesize* – the number of independent turn-taking value samples to include
- *persistentData* – set to True to save the turn-taking value distribution to disk for future use

See Section 3 of: Raffensperger, P. A., Webb, R. Y., Bones, P. J., and McInnes, A. I. (2012). A simple metric for turn-taking in emergent communication. *Adaptive Behavior*, 20(2):104-116.

`turntakingmeasurementtools.f_turn` (*U*, *t*, *r*)

Compute the Iizuka and Ikegami turn-taking fitness value present in the usage attempt sequence *U*, at time *t* and resolution *r*, using the block of time steps [*t*, *t+r-1*].

Parameters:

- *U* – a binary valued usage attempt sequence of shape (2, *l*), this measure is only defined for pairs of agents, where *l* is the length. (As either a numpy array or a list of lists)
- *t* – the time at which to calculate the turn-taking measure
- *r* – the ‘resolution’ or window length, with $l \leq t+r$

Return value: A real value in $[0, r/4]$ representing the quantity of turn-taking in U in the range $[t, t+r-1]$

See Eqs. 8 and 9 in: Iizuka, H. and Ikegami, T. (2004). Adaptability and diversity in simulated turn-taking behavior. *Artificial Life*, 10(4):361-378.

`turntakingmeasurementtools.fairness_jain` ($U, t, r, exponent=2.0$)

Compute the Jain fairness of the turn allocation in the usage attempt sequence U , at time t and resolution r , using the block of time steps $[t, t+r-1]$.

Parameters:

- U – a binary valued usage attempt sequence of shape (A, l) , where A is the number of agents and l is the length. (As either a numpy array or a list of lists)
- t – the time at which to calculate the fairness measure
- r – the ‘resolution’ or window length, with $l \leq t+r$

Return value: A real value representing the Jain fairness of the turn allocation in U in the range $[t, t+r-1]$

See: Jain, R., Chiu, D., & Hawe, W. (1984). A quantitative measure of fairness and discrimination for resource allocation in shared computer systems Tech. Report, Hudson, MA: Digital Equipment Corporation.

`turntakingmeasurementtools.fairness_min` (U, t, r)

Compute the fairness of the turn allocation in the usage attempt sequence U , at time t and resolution r , using the block of time steps $[t, t+r-1]$.

Parameters:

- U – a binary valued usage attempt sequence of shape (A, l) , where A is the number of agents and l is the length. (As either a numpy array or a list of lists)
- t – the time at which to calculate the fairness measure
- r – the ‘resolution’ or window length, with $l \leq t+r$

Return value: A real value in $[0, 1]$ representing the fairness of the turn allocation in U in the range $[t, t+r-1]$

See Eq. 2 in: Raffensperger, P. A., Webb, R. Y., Bones, P. J., and McInnes, A. I. (2012). A simple metric for turn-taking in emergent communication. *Adaptive Behavior*, 20(2):104-116.

`turntakingmeasurementtools.generate_random_turn_taking_values` ($numAgents, r, pu, samples$)

Generate a population of random turn-taking values given the usage attempt probability of the agents

Parameters:

- $numAgents$ – the desired number of probabilistic agents to include
- r – the turn-taking resolution
- pu – the probability that a bit in each usage attempt sequence will be 1
- $samples$ – the number of random turn-taking values to generate

Return value: A list of random turn-taking values

See Section 3 of: Raffensperger, P. A., Webb, R. Y., Bones, P. J., and McInnes, A. I. (2012). A simple metric for turn-taking in emergent communication. *Adaptive Behavior*, 20(2):104-116.

`turntakingmeasurementtools.generate_random_usage_attempt_sequence` ($numAgents, length, pu$)

Generate a random, binary-valued usage attempt sequence with a particular usage attempt probability

Parameters:

- $numAgents$ – the desired number of agents in the random usage attempt sequence
- $length$ – the number of time steps of the usage attempt sequence
- pu – the probability that a bit in the usage attempt sequence will be 1

Return value: A random binary-valued usage attempt sequence as a numpy array of shape (numAgents, length)

See Section 3 of: Raffensperger, P. A., Webb, R. Y., Bones, P. J., and McInnes, A. I. (2012). A simple metric for turn-taking in emergent communication. *Adaptive Behavior*, 20(2):104-116.

`turntakingmeasurementtools.get_tt_distribution_filename` (numAgents, r, pu, samples)

Gets standardised file name for turn-taking probability distribution files

`turntakingmeasurementtools.load_random_tt_distribution` (numAgents, r, pu, samples)

Load a file with a population of random turn-taking values, assuming that it exists

Parameters:

- numAgents – the desired number of probabilistic agents to include
- r – the turn-taking resolution
- pu – the probability that a bit in each usage attempt sequence will be 1
- samples – the number of random turn-taking values to generate

See Section 3 of: Raffensperger, P. A., Webb, R. Y., Bones, P. J., and McInnes, A. I. (2012). A simple metric for turn-taking in emergent communication. *Adaptive Behavior*, 20(2):104-116.

`turntakingmeasurementtools.save_random_tt_distribution` (numAgents, r, pu, samples, verbose=False)

Save a file with a population of random turn-taking values given the usage attempt probability of the agents

Parameters:

- numAgents – the desired number of probabilistic agents to include
- r – the turn-taking resolution
- pu – the probability that a bit in each usage attempt sequence will be 1
- samples – the number of random turn-taking values to generate

Keyword arguments:

- verbose – set to True to make this function print out its status to stdin as it goes

See Section 3 of: Raffensperger, P. A., Webb, R. Y., Bones, P. J., and McInnes, A. I. (2012). A simple metric for turn-taking in emergent communication. *Adaptive Behavior*, 20(2):104-116.

`turntakingmeasurementtools.tautau` (U, t, r)

Compute the quantity of turn-taking present in the usage attempt sequence U, at time t and resolution r, using the block of time steps [t, t+r-1].

Parameters:

- U – a binary valued usage attempt sequence of shape (A, l), where A is the number of agents and l is the length. (As either a numpy array or a list of lists)
- t – the time at which to calculate the turn-taking measure
- r – the ‘resolution’ or window length, with $l \leq t+r$

Return value: A real value in [0, 1] representing the quantity of turn-taking in U in the range [t, t+r-1]

See Eqs. 4 and 5 in: Raffensperger, P. A., Webb, R. Y., Bones, P. J., and McInnes, A. I. (2012). A simple metric for turn-taking in emergent communication. *Adaptive Behavior*, 20(2):104-116.

3.2 License

Turn-Taking Measurement Tools is licensed under a new BSD license. You are encouraged to use it in both free and commercial software. If you use this library in an academic context, we would appreciate it if you referenced our paper.

Copyright (C) 2012, Peter Raffensperger. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Peter Raffensperger nor the names of other contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

INDEX

A

`allocation()` (in module `turntakingmeasurementtools`), [7](#)

C

`clean_up_tt_distribution_files()` (in module `turntakingmeasurementtools`), [7](#)

E

`efficiency()` (in module `turntakingmeasurementtools`), [7](#)

`estimate_probability_of_tt_due_to_chance()` (in module `turntakingmeasurementtools`), [7](#)

`estimate_tt_mean_and_variance()` (in module `turntakingmeasurementtools`), [8](#)

F

`f_turn()` (in module `turntakingmeasurementtools`), [8](#)

`fairness_jain()` (in module `turntakingmeasurementtools`), [9](#)

`fairness_min()` (in module `turntakingmeasurementtools`), [9](#)

G

`generate_random_turn_taking_values()` (in module `turntakingmeasurementtools`), [9](#)

`generate_random_usage_attempt_sequence()` (in module `turntakingmeasurementtools`), [9](#)

`get_tt_distribution_filename()` (in module `turntakingmeasurementtools`), [10](#)

L

`load_random_tt_distribution()` (in module `turntakingmeasurementtools`), [10](#)

S

`save_random_tt_distribution()` (in module `turntakingmeasurementtools`), [10](#)

T

`tautau()` (in module `turntakingmeasurementtools`), [10](#)

`turntakingmeasurementtools` (module), [7](#)