

Using Turktools

Hadas Kotek

(based on materials created in collaboration with Michael Y. Erlewine)

Experimental semantics and pragmatics

NYU, October 2017

In these slides

- §1 Why experimentation? Basic considerations.
- §2 *Turktools* basics: Terminology
- §3 The relationship between templates and Turk item files
- §4 Skeletons, templates, and the Lister

(Wednesday: uploading an experiment to Amazon's Mechanical Turk)

Why/when AMT?

Q: When would we prefer to gather data from dozens of potentially very noisy participants in a potentially very noisy environment than from a handful of trusted and cooperative consultants in our office?

A1: Representativeness across speakers — but why?

- Observer biases
- Consultants are Non-representative sample

A2: Representativeness across items

- Noise
 - The experimental task involves engaging noisy cognitive systems (memory, etc.)
 - The phenomenon of interest itself is noisy

 Experimentation: not at all costs! Design matters.

Why/when AMT?

Q: When would we prefer to gather data from dozens of potentially very noisy participants in a potentially very noisy environment than from a handful of trusted and cooperative consultants in our office?

A1: Representativeness across speakers — but why?

- Observer biases
- Consultants are Non-representative sample

A2: Representativeness across items

- Noise
 - The experimental task involves engaging noisy cognitive systems (memory, etc.)
 - The phenomenon of interest itself is noisy

 Experimentation: not at all costs! Design matters.

Why/when AMT?

Q: When would we prefer to gather data from dozens of potentially very noisy participants in a potentially very noisy environment than from a handful of trusted and cooperative consultants in our office?

A1: Representativeness across speakers — but why?

- Observer biases
- Consultants are Non-representative sample

A2: Representativeness across items

- Noise
 - The experimental task involves engaging noisy cognitive systems (memory, etc.)
 - The phenomenon of interest itself is noisy

 Experimentation: not at all costs! Design matters.

Why/when AMT?


Q: When would we prefer to gather data from dozens of potentially very noisy participants in a potentially very noisy environment than from a handful of trusted and cooperative consultants in our office?

A1: Representativeness across speakers — but why?

- Observer biases
- Consultants are Non-representative sample

A2: Representativeness across items

- Noise
 - The experimental task involves engaging noisy cognitive systems (memory, etc.)
 - The phenomenon of interest itself is noisy

 Experimentation: not at all costs! Design matters.

Simple designs: 2×1

- In case we are simply interested in the effect of just one feature/property on a judgment.
- Effect of **animacy** on grammaticality of English ‘make’-causative:

- (1) a. The coach made the ball bounce on the floor.
b. The coach made the gymnast bounce on the floor.

Life is rarely this simple, but it's nice when it is...

Simple designs: 2×2

More realistically, you will want to aim for a 2×2 design.

- Effect of **animacy** of the causee in **two kinds of causative constructions**: Animacy × Type.
 - Animate vs. inanimate
 - Lexical causative vs. ‘make’ causative
- (2) a. That’s the ball that the coach **bounced** on the floor.
b. That’s the gymnast that the coach **bounced** on the floor.
c. That’s the ball that the coach **made bounce** on the floor.
d. That’s the gymnast that the coach **made bounce** on the floor

(from Kotek & Erlewine, ms, “Blocking effects in English causatives”)

Simple designs: 2×2

- (2) a. That's the ball that the coach **bounced** on the floor.
- b. That's the gymnast that the coach **bounced** on the floor.
- c. That's the ball that the coach **made bounce** on the floor.
- d. That's the gymnast that the coach **made bounce** on the floor
- We want each participant to only rate *one* of these **conditions** for a given **item set**. (Why?)
 - **Latin square design:** cycles through our items to create lists.
 - List 1: (1a), (2b), (3c), (4d), (5a), ...
 - List 2: (1b), (2c), (3d), (4a), (5b), ...
 - List 3: (1c), (2d), (3a), (4b), (5c), ...
 - List 4: (1d), (2a), (3b), (4c), (5d), ...(for 4 conditions, we need a multiple of 4 lists, to collect the same number of observations for each item)

A	B	C
C	A	B
B	C	A

Simple designs: 2×2

- (2) a. That's the ball that the coach **bounced** on the floor.
- b. That's the gymnast that the coach **bounced** on the floor.
- c. That's the ball that the coach **made bounce** on the floor.
- d. That's the gymnast that the coach **made bounce** on the floor
- We want each participant to only rate *one* of these **conditions** for a given **item set**. (Why?)
 - **Latin square design:** cycles through our items to create lists.
 - List 1: (1a), (2b), (3c), (4d), (5a), ...
 - List 2: (1b), (2c), (3d), (4a), (5b), ...
 - List 3: (1c), (2d), (3a), (4b), (5c), ...
 - List 4: (1d), (2a), (3b), (4c), (5d), ...(for 4 conditions, we need a multiple of 4 lists, to collect the same number of observations for each item)

A	B	C
C	A	B
B	C	A

Simple designs: 2×2

More complex designs than a 2×2 should be avoided, as their results can be hard to interpret.

- In general, you should always know how you're going to analyze your data before you begin, and have some predictions for what you expect to find.*
- We'll concentrate on a small 5-item experiment this week, though a 'real' experiment should probably have more items
- (Ask your friendly instructors about considerations to do with power and minimal number of items.)

(*But also remember that only rarely will your first pilot actually run smoothly and show what you expect...)

Normally, at least as many as the targets.

- Distract from the true purpose of the experiment
- Even out skewness in items
- Serve as an exclusion criterion
 - Overall accuracy
 - “catch items”

The idea behind *turktools*:

Separate out the intellectual work of putting together an experiment from the technical aspects. Allow less experienced linguists to use experiments.

- Intellectual:

- Formulate a research question,
- Operationalize:
 - Pick an experimental design,
 - Create items.

(This is what you should spend most of your time on.)

- Technical:

- Create an appropriate HTML template to present your study,
- Randomize your items,
- Create multiple lists to avoid bias due to presentation order,
- Format files to fit the format required by AMT.

(This is where *turktools* comes in.)

The idea behind *turktools*:

Separate out the intellectual work of putting together an experiment from the technical aspects. Allow less experienced linguists to use experiments.

- Intellectual:
 - Formulate a research question,
 - Operationalize:
 - Pick an experimental design,
 - Create items.

(This is what you should spend most of your time on.)

- Technical:
 - Create an appropriate HTML template to present your study,
 - Randomize your items,
 - Create multiple lists to avoid bias due to presentation order,
 - Format files to fit the format required by AMT.

(This is where *turktools* comes in.)

The idea behind *turktools*:

Separate out the intellectual work of putting together an experiment from the technical aspects. Allow less experienced linguists to use experiments.

- Intellectual:

- Formulate a research question,
- Operationalize:
 - Pick an experimental design,
 - Create items.

(This is what you should spend most of your time on.)

- Technical:

- Create an appropriate HTML template to present your study,
- Randomize your items,
- Create multiple lists to avoid bias due to presentation order,
- Format files to fit the format required by AMT.

(This is where *turktools* comes in.)

Some terminology

- An **item set** is a set of sentences/stimuli that vary the **factors of interest** in a systematic way, and hold constant everything else. (Cf: **Lexicalization**; **Independent Variables**)
- A **condition** is a particular setting of all of the factors of interest.
- Individual stimuli within an item set are called **items**.
- Items are grouped into **sections**. Normally, “target” and “filler.”

```
# blocking 1 inanimate-make-v
```

```
That's the ball that the coach bounced on the floor.
```

```
# blocking 1 inanimate-v
```

```
That's the gymnast that the coach bounced on the floor.
```

```
# blocking 1 animate-make-v
```

```
That's the ball that the coach made bounce on the floor.
```

```
# blocking 1 animate-v
```

```
That's the gymnast that the coach made bounce on the floor.
```


Getting set up

This should have already happened:

- Install Python 2.7.x: <http://www.python.org/getit/>
(choose 2.7.x, not 3.x)
- Download turktools: <http://turktools.net>

For today:

- Access files at: <http://hkotek.com/turk/index.html>
- You might want to save these files in the same folder you have your *turktools* scripts in.

Your template and items

What people see:

In order to get paid, please make sure that you answer all 14 items.

Consent Statement: By answering the following questions, you are participating in a study being performed by linguists in the Department of Linguistics and Philosophy at Massachusetts Institute of Technology. If you have questions about this research, please contact [EMAIL](#). Your participation in this research is voluntary. You may decline to answer any or all of the following questions. You may decline further participation at any time without adverse consequences. Your anonymity is assured; the researchers who have requested your participation will not receive any personal information about you.

1. That's the sentence that the linguist added without thinking.

NATURAL

UNNATURAL

-
2. That's the idea that the team floated at the meeting.

NATURAL

UNNATURAL

-
3. That's the rabbit that the magician made vanish into thin air.

NATURAL

UNNATURAL

-
4. That's the intern that the supervisor returned for another shift.

NATURAL

UNNATURAL

Your template and items

What you give Turk:



Template file
.html



Turk items file
.turk.csv

👉 Open up `binary-mcgill-TK1-10.html` in your browser. Probably double-clicking on it will work.

- What kind of experimental paradigm is this template for?
- How many items is this template file expecting?
- How is it different than what the subject sees?

Your template and items

What you give Turk:



Template file
.html



Turk items file
.turk.csv

👉 Open up `binary-mcgill-TK1-10.html` in your browser. Probably double-clicking on it will work.

- What kind of experimental paradigm is this template for?
- How many items is this template file expecting?
- How is it different than what the subject sees?

Your template and items

What you give Turk:



Template file
.html



Turk items file
.turk.csv

👉 Open up `binary-sample-items.turk.csv` in Excel.
(CSV files are a kind of plain-text spreadsheet.)

- How many separate lists does this include?
- How many items do these lists have?
- What's the relationship between list 0 and 1? 2 and 3?

Your template and items

When you upload a template and item file to Turk, each one of these lists will be turned into a HIT (“Human Intelligence Task”).

Each string in the row in the Turk items file will be plugged into “fields” in the template.

```
list trial_1_1          trial_2_1          ...
0    That's the rabbit  That's the boy    ...
     that the magician  that the instruc-
     made vanish into  tor made float in
     thin air.          the pool.
:    :                  :                  ...
```

In list #0, the text `${trial_2_1}` in the template file will be replaced with “That’s the boy that the instructor made float in the pool.”

Your template and items

When you upload a template and item file to Turk, each one of these lists will be turned into a HIT (“Human Intelligence Task”).

Each string in the row in the Turk items file will be plugged into “fields” in the template.

list	trial_1_1	trial_2_1	...
0	That's the rabbit that the magician made vanish into thin air.	That's the boy that the instruc- tor made float in the pool.	...
:	:	:	...

In list #0, the text `${trial_2_1}` in the template file will be replaced with “That’s the boy that the instructor made float in the pool.”

Experiment types and skeletons

We create templates from “skeleton” files. Think “template recipes.”

Each skeleton corresponds to a different kind of experiment. Let's open some skeletons in our browser.

- `binary.skeleton.html`
- `binary-image.skeleton.html`
- `completion.skeleton.html`
- `image-choice.skeleton.html`
- `sentence-choice.skeleton.html`
- ...

More described in the paper.

Also possible: audio playback.

Experiment types and skeletons

We create templates from “skeleton” files. Think “template recipes.”

Each skeleton corresponds to a different kind of experiment. Let’s open some skeletons in our browser.

- `binary.skeleton.html`
- `binary-image.skeleton.html`
- `completion.skeleton.html`
- `image-choice.skeleton.html`
- `sentence-choice.skeleton.html`
- ...

More described in the paper.

Also possible: audio playback.

The Lister does the following:

- Takes an items file (.txt) as input.
- Create Latin Square lists from the items, so that each participant sees only one condition per item.
- Randomize the lists and mix the targets and fillers together, following certain constraints.
- Print the resulting lists into a Turk items file (.turk.csv).

The Lister input file

- 👉 Open `binary-image-sample-items.txt`, which is a Lister input file.

Each **item** has a **header** beginning with `#`. The header includes a section name, item number, and condition name.

```
# target 1 most  
Most of the dots are blue.  
http://hkotek.com/turk/most-pics/2C-1.png  
:
```

Each line of the item corresponds to a different **field** in the template. For this experiment, each item has two fields: a sentence and a URL of an image. At least one blank line is required between items.

- 👉 How many sections are there? How many items do they have? How many different conditions are there in each section?
- 👉 How many items will an individual participant be asked to answer?

The Lister input file

- 👉 Open `binary-image-sample-items.txt`, which is a Lister input file.

Each **item** has a **header** beginning with `#`. The header includes a section name, item number, and condition name.

```
# target 1 most
Most of the dots are blue.
http://hkotek.com/turk/most-pics/2C-1.png
      :
```

Each line of the item corresponds to a different **field** in the template. For this experiment, each item has two fields: a sentence and a URL of an image. At least one blank line is required between items.

- 👉 How many sections are there? How many items do they have? How many different conditions are there in each section?
- 👉 How many items will an individual participant be asked to answer?

Creating lists for Turk



Lister input
.txt



`lister.py`



Turk items
.turk.csv

- 👉 Run `lister.py`. Enter `binary-sample-items.txt` and request a multiple of 2 lists. It will ask you about filler placement constraints.

It will then print a nice summary of how many sections, items, conditions, etc. were found and (hopefully) tell you that the CSV files were successfully created.

- NB:** Double the number of lists you request will be created. That's because the reverse order of each list is also always included, to attempt to counter ordering effects.

- 👉 How many items does it say are in each list?

Skeletons

Skeletons are ‘template recipes’. By default, they consist of three blocks:

- Instructions and consent statement.
- Items block.
- Demographics information.

Skeletons can be edited. E.g.:

- Natural/Unnatural replaced with True/False
- Update consent statement
- Ask different demographics info, remove this block, move it up.

The items block is represented by one dummy item. The *templater* will replace this with as many items as your experiment has.

Skeletons

Skeletons are ‘template recipes’. By default, they consist of three blocks:

- Instructions and consent statement.
- Items block.
- Demographics information.

Skeletons can be edited. E.g.:

- Natural/Unnatural replaced with True/False
- Update consent statement
- Ask different demographics info, remove this block, move it up.

The items block is represented by one dummy item. The *templater* will replace this with as many items as your experiment has.

Skeletons

Skeletons are ‘template recipes’. By default, they consist of three blocks:

- Instructions and consent statement.
- Items block.
- Demographics information.

Skeletons can be edited. E.g.:

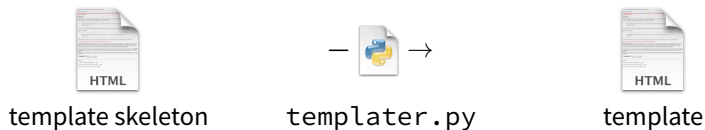
- Natural/Unnatural replaced with True/False
- Update consent statement
- Ask different demographics info, remove this block, move it up.

The items block is represented by one dummy item. The *templater* will replace this with as many items as your experiment has.

Open the skeleton completion `.skeleton.html`. How many **fields** does each item have?

- 👉 Each item in your items file should have at least as many fields as the skeleton expects.
- Each field corresponds to a new line in your items file.
- Any additional fields are “hidden fields” — they will not be shown to participants, but will still be in your results file.
 - This is a useful way to indicate expected correct answers to fillers.
 - It’s possible to have hidden fields for just some but not all items (e.g., no expected correct answer for targets).

We need a template of the right length. We have a tool to create templates of arbitrary length from skeletons:

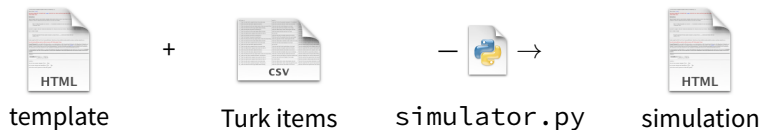


👉 Run `templater.py`. Create a template based on `binary-mcgill.skeleton.html` for 10 trials. Pick a code, any code.

Open the resulting template file and verify that it's built for 10 trials.

Putting it together

- 👉 Use `simulator.py` to put your new template and our new Turk items file together. Make sure the resulting simulation looks good.

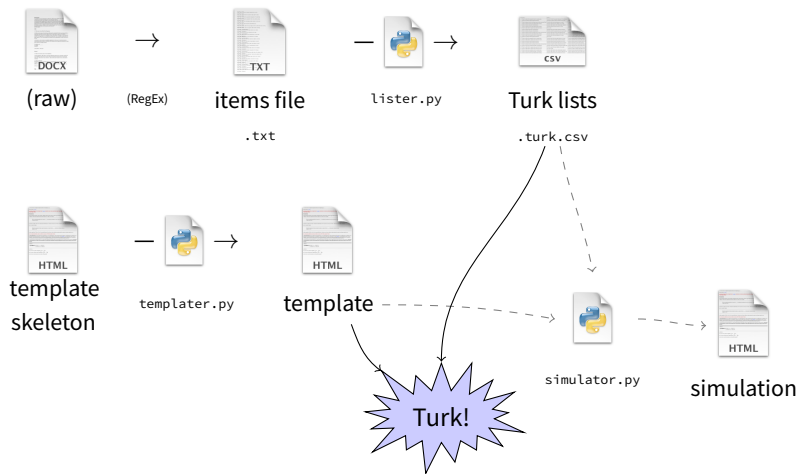


Now you have a `binary-nyu-mcgill-10.simulation.html` file in your folder. Look at it in your browser. Note that Turk will add a submit button, so our template doesn't need to include that.

At this point you would be ready to upload your template and Turk items (`.turk.csv`) file to Turk!

Turk technical workflow overview

aka "Turkflow"



Exercise: Run your own study!

- Create an items file. Make sure it's formatted correctly. (You might have already done this before class.)
- Create MTurk-compatible item lists with `lister.py`.
- Customize the skeleton of your choice.
 - Edit the instructions and consent statement.
 - Make any changes to the items block and demographics block that you would like.
 - Save your edited skeleton in the same folder as *turktools*.
- Create a template from your skeleton with the appropriate number of items with `templater.py`.
- Simulate with `simpulator.py`! Win all the prizes!
- (On Wednesday: upload to MTurk.)

Completions exercise (time permitting)

- Open `example3-items.txt`. Here are a few sentences I wrote for a completion study, where participants will choose between two quantifiers in a sentence.
 - Targets will have two conditions, *did* and *was*, based on the verb at the end of the sentence.
- Open `completion.skeleton.html`. Each trial must be split up into four strings:
 - field 1: text before the gap
 - field 2: text after the gap
 - field 3: gap option one
 - field 4: gap option two
- Prepare these sentences for the Lister.
Hint: regular expressions will save you time!
- Create Turk items with `lister.py`.
- Create a template from `completion.skeleton.html` with the appropriate number of items with `templater.py`.
- Simulate! Win all the prizes!