Computing Dynamic Meanings: Building Integrated Competence-Performance Theories for Semantics

Day 3, part 2: Modeling linguistic performance and Bayesian estimation of ACT-R model parameters

Jakub Dotlačil & Adrian Brasoveanu

ESSLLI 2018, August 8 2018

Subsymbolic components of declarative memory

Goals for this part:

- introduce 'subsymbolic' declarative memory components of ACT-R that are essential for modeling linguistic performance – i.e., actual human behavior in experimental tasks
- build end-to-end models for one psycholinguistic task: self-paced reading
- evaluate how well these models fit actual data $\Rightarrow$ quantitative comparison for qualitative theories
- end-to-end models:
  - explicit linguistic analyses primarily encoded in the production rules, i.e., in procedural memory
  - realistic model of declarative memory
  - simple, but reasonably realistic vision and motor modules

The power law of forgetting

Main idea behind the ACT-R declarative memory architecture:

> *"[H]uman memory is behaving optimally with respect to the pattern of past information presentation. Each item in memory has had some history of past use.*
>
> *For instance, our memory for one person's name may not have been used in the past month but might have been used five times in the month previous to that. What is the probability that the memory will be needed (used) during the conceived current day?*
>
> *Memory would be behaving optimally if it made this memory less available than memories that were more likely to be used but made it more available than less likely memories." (Anderson and Schooler, 1991, p. 396)*

Ebbinghaus (1913, Ch. 7) retention data

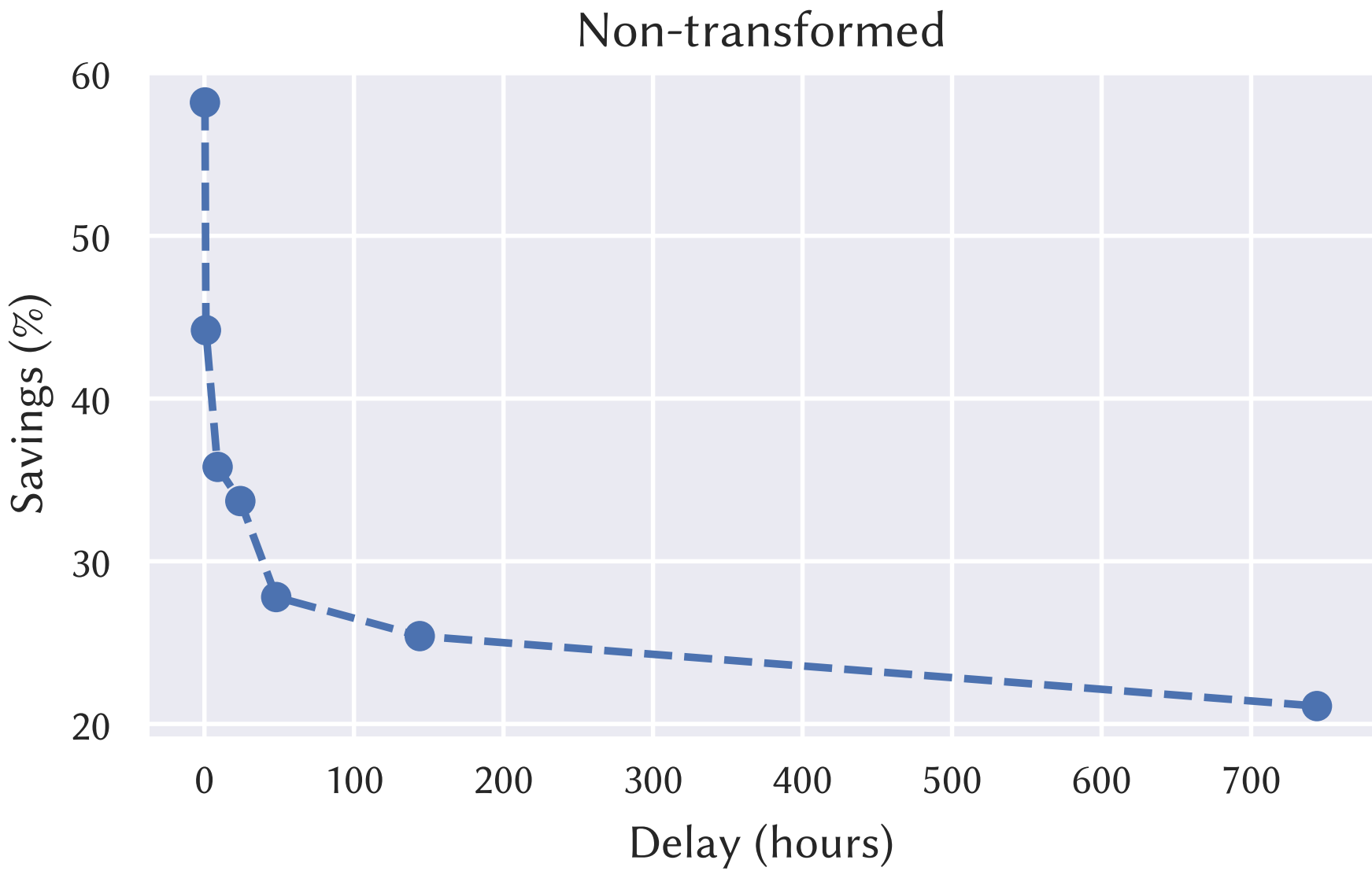
- availability of a dec. mem. chunk – its activation, which determines probability of retrieval and retrieval latency – is a function of the past use of that chunk

  other factors at work too, e.g., cognitive context; more later
- to understand ACT-R formalization, examine the well-known Ebbinghaus (1913, Ch. 7) retention data
- stimuli: ≈ 2300 nonsense CVC syllables; mixed together, syllables randomly selected to construct lists of syllables
- method: learning to criterion; participant repeats the list as many times as necessary to reach a prespecified level of accuracy, e.g., one perfect reproduction
- retention measure (dependent variable): percent savings
  - subtracting the number of repetitions required to relearn material to criterion at a later point (independent variable: time) from the number of repetitions originally required to learn the material to the same criterion

Ebbinghaus (1913, Ch. 7) retention data

```
>>> import pandas as pd
>>> ebbinghaus_data =\
...     pd.read_csv('ebbinghaus_retention_data.csv')
>>> ebbinghaus_data
   delay_in_hours  percent_savings
0            0.33             58.2
1            1.00             44.2
2            8.80             35.8
3           24.00             33.7
4           48.00             27.8
5          144.00             25.4
6          744.00             21.1
```

Figure: Ebbinghaus retention data: non-transformed



Forgetting curve: exponential? (no)

Could forgetting curve in non-transformed plot reflect an underlying negative exponential forgetting function?

(1) $P = A \cdot e^{-bT}$

- P: performance measure (percent savings),
- T: time delay
- A, b: model parameters

(2) Taking logs of both sides: $\log(P) = \log(A) - bT$
P: linear function of T with intercept $\log(A)$ and negative slope $-b$

Prediction: log-performance is a linear function of time.

- we should see a fairly straight line with a negative slope if we log-transform performance P
- we don't: see plot on next slide

Figure: Ebbinghaus retention data: log savings

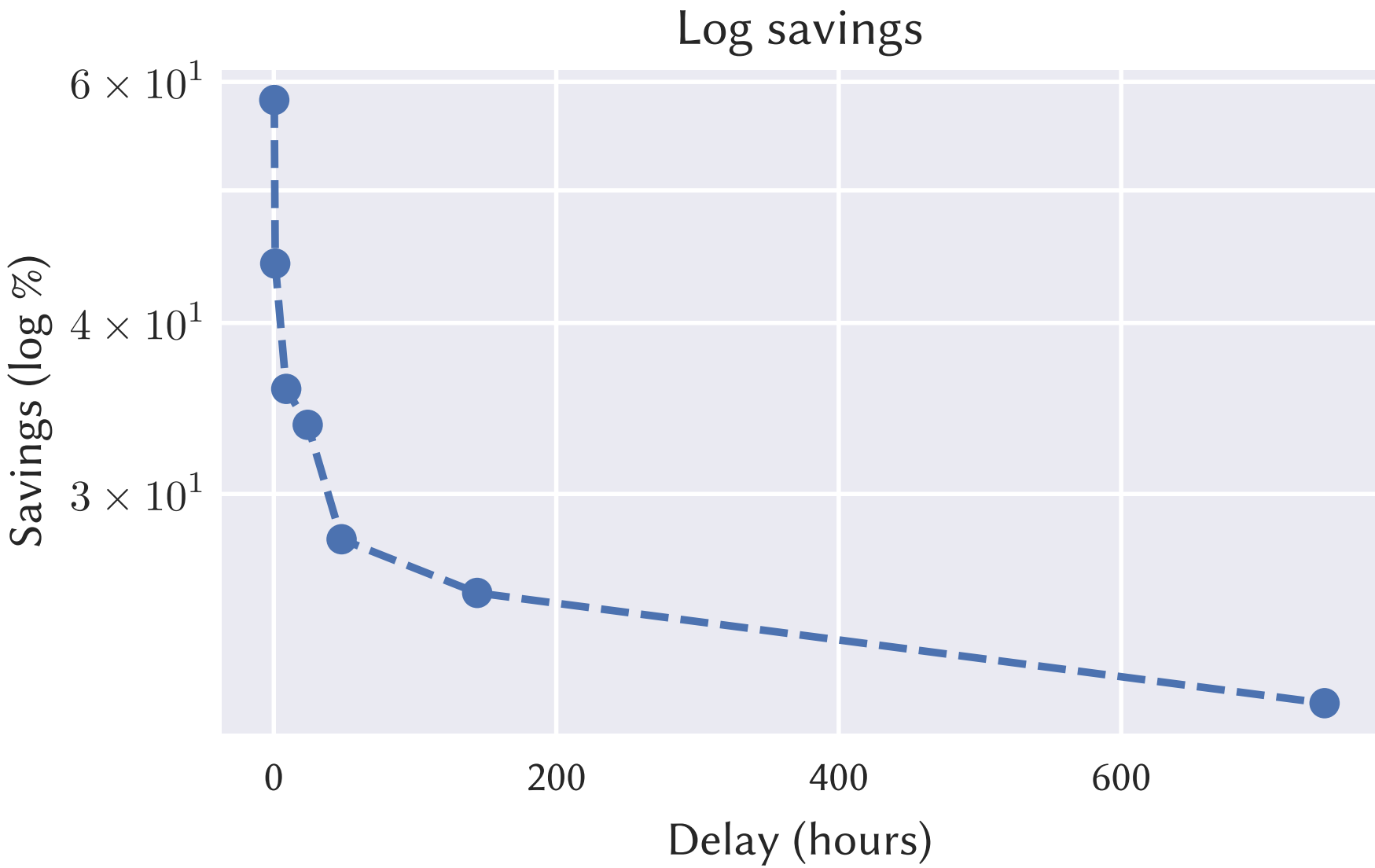
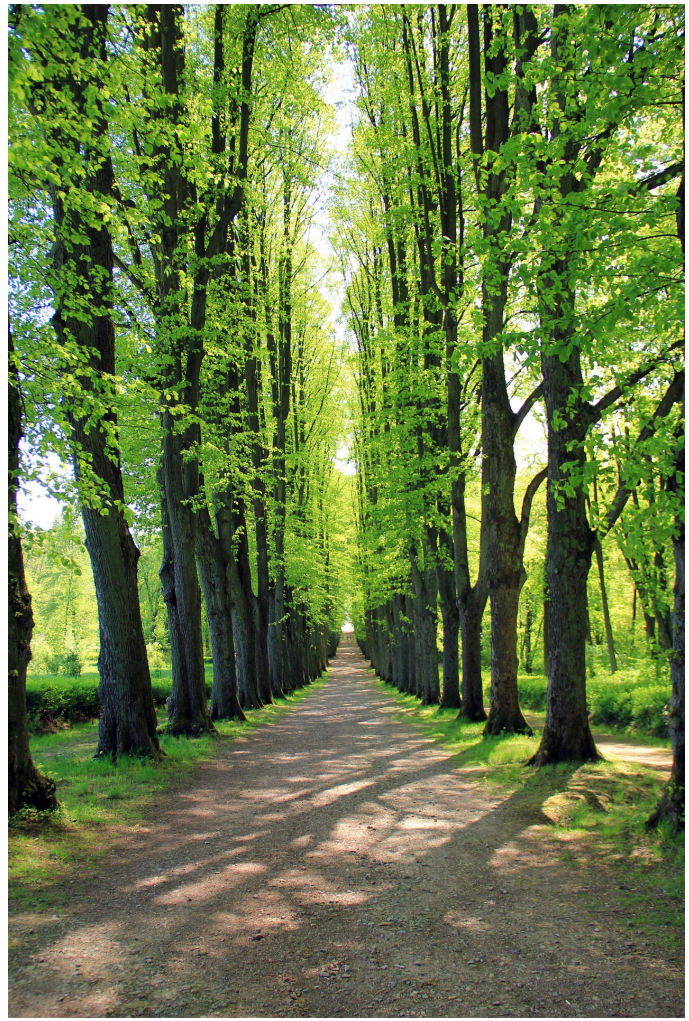


Figure: Log transformation / compression: intuition



Evenly (linearly) spaced trees, but the further away two trees are (the larger the numbers), the smaller the distance between them appears (difference between numbers compressed further)

Fit exponential model to data

```
>>> import numpy as np                                                    1
>>> import pymc3 as pm                                                    2
>>> from pymc3.backends import SQLite                                     3
>>> from pymc3.backends.sqlite import load                                4
                                                                          5
>>> delay = ebbinghaus_data['delay_in_hours']                             6
>>> savings = ebbinghaus_data['percent_savings']                          7
                                                                          8
>>> exponential_model = pm.Model()                                        9
```

Fit exponential model to data (ctd.)

```
>>> with exponential_model:
...     # priors
...     intercept = pm.Normal('intercept', mu=0, sd=100)
...     slope = pm.Normal('slope', mu=0, sd=100)
...     sigma = pm.HalfNormal('sigma', sd=100)
...     # likelihood
...     mu = pm.Deterministic('mu',\
...         intercept + slope*delay)
...     log_savings = pm.Normal('log_savings', mu=mu,
...         sd=sigma, observed=np.log(savings))
...     # posteriors
...     #db = SQLite('exponential_model_trace.sqlite')
...     #trace = pm.sample(draws=5000, trace=db,\
...                       #n_init=50000, njobs=4)
...     trace = load('exponential_model_trace.sqlite')
...
```

Fit exponential model to data (ctd.)

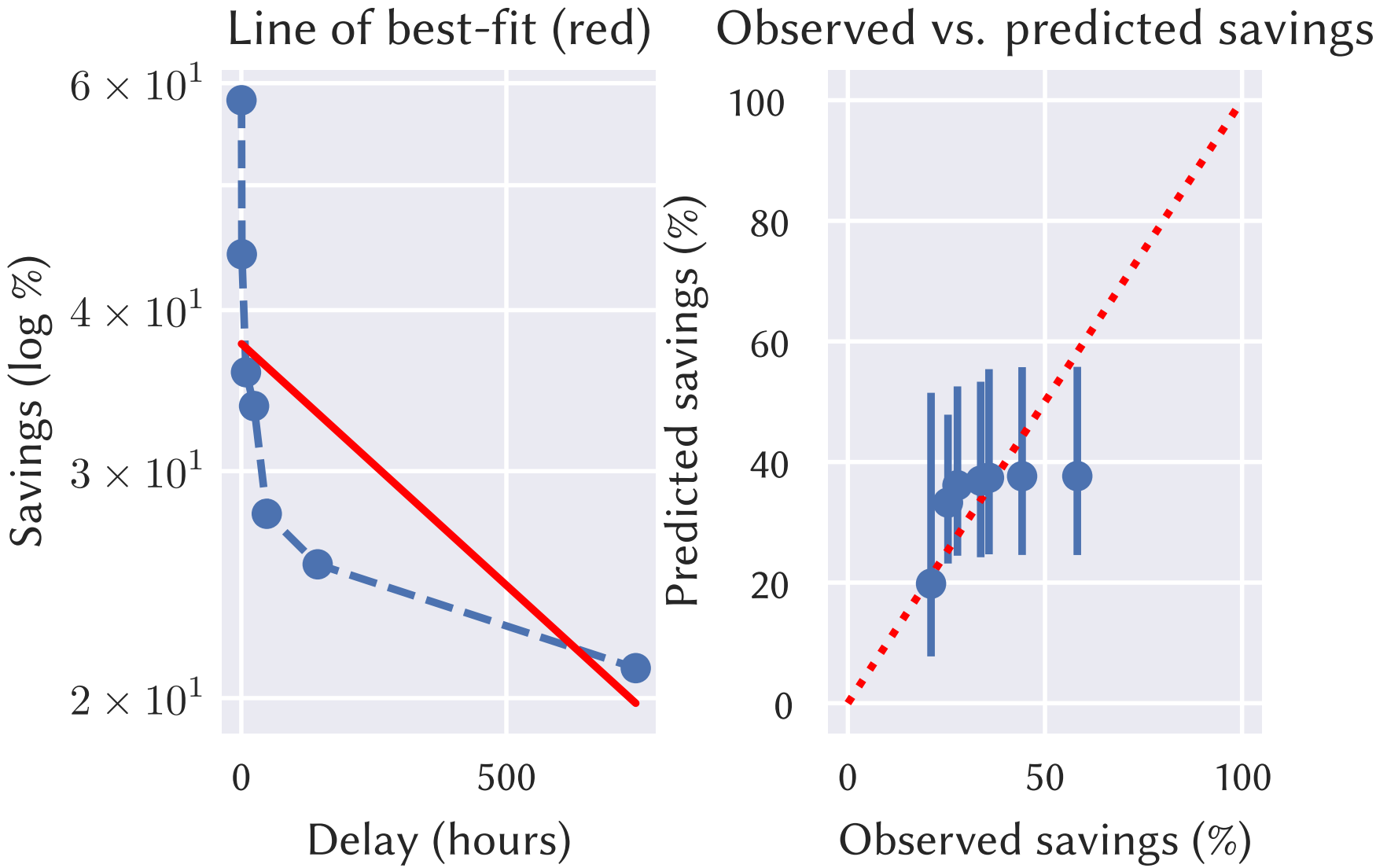
We use Bayesian methods for data analysis / parameter estimation / quantitative model comparison.
We compare:

- actual observations
- predictions made by the theory / hypothesis that performance (forgetting) is an exponential function of time

Model components:

- low information priors for the intercept, slope, and noise (familiar from our intro to Bayes slides)
- likelihood: direct coding of the exponential theory / hypothesis

Figure: The exponential forgetting model



Power law model of forgetting

Performance: a power function of time

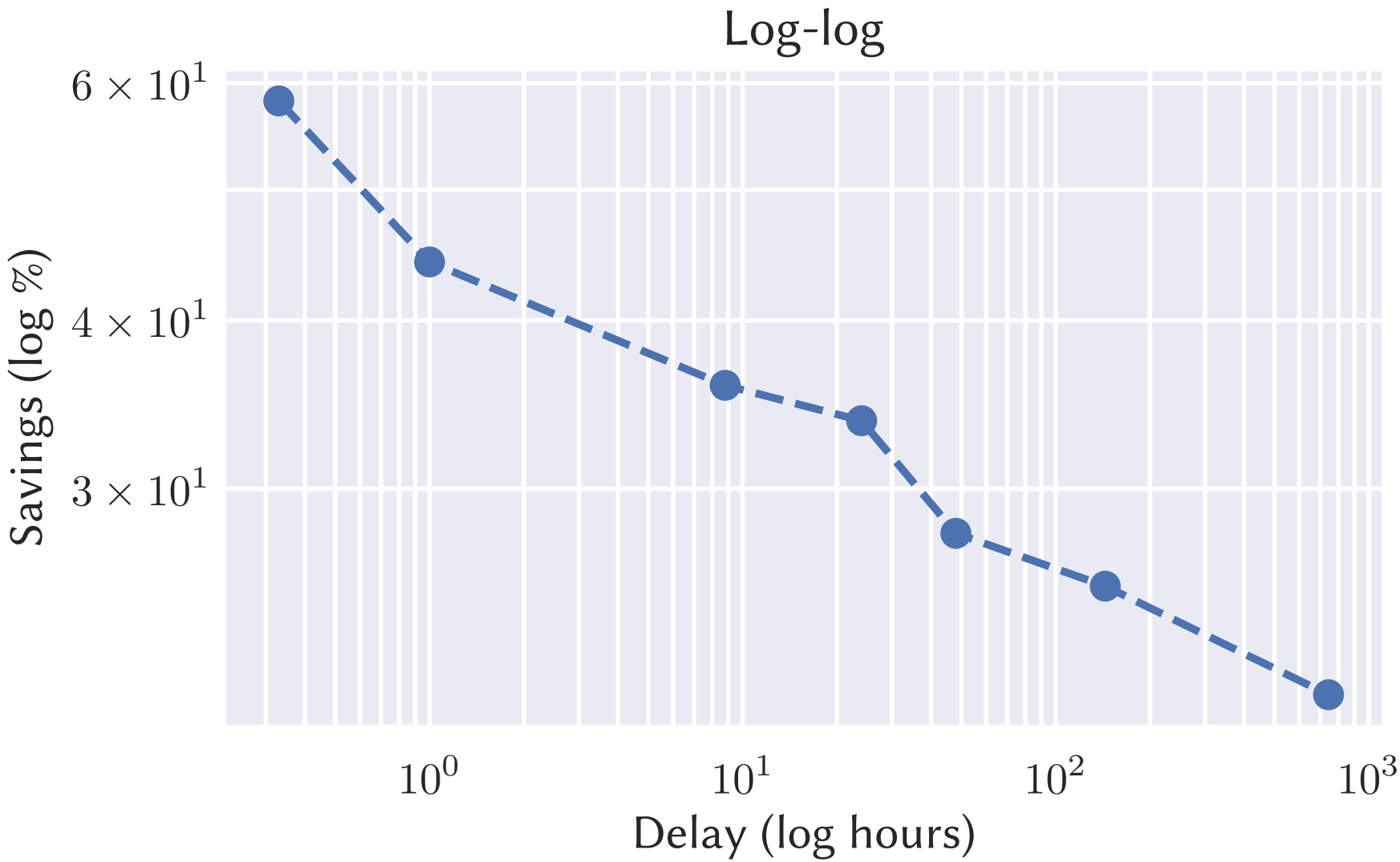
- performance: linear function of time only if both performance and time are log transformed

(3) $\boxed{P = A \cdot T^{-b}}$

(4) Taking logs of both sides: $\boxed{\log(P)} = \log(A) - b\boxed{\log(T)}$

Log-transforming both delay and savings in the Ebbinghaus data reveals a linear dependency – see plot on next slide.

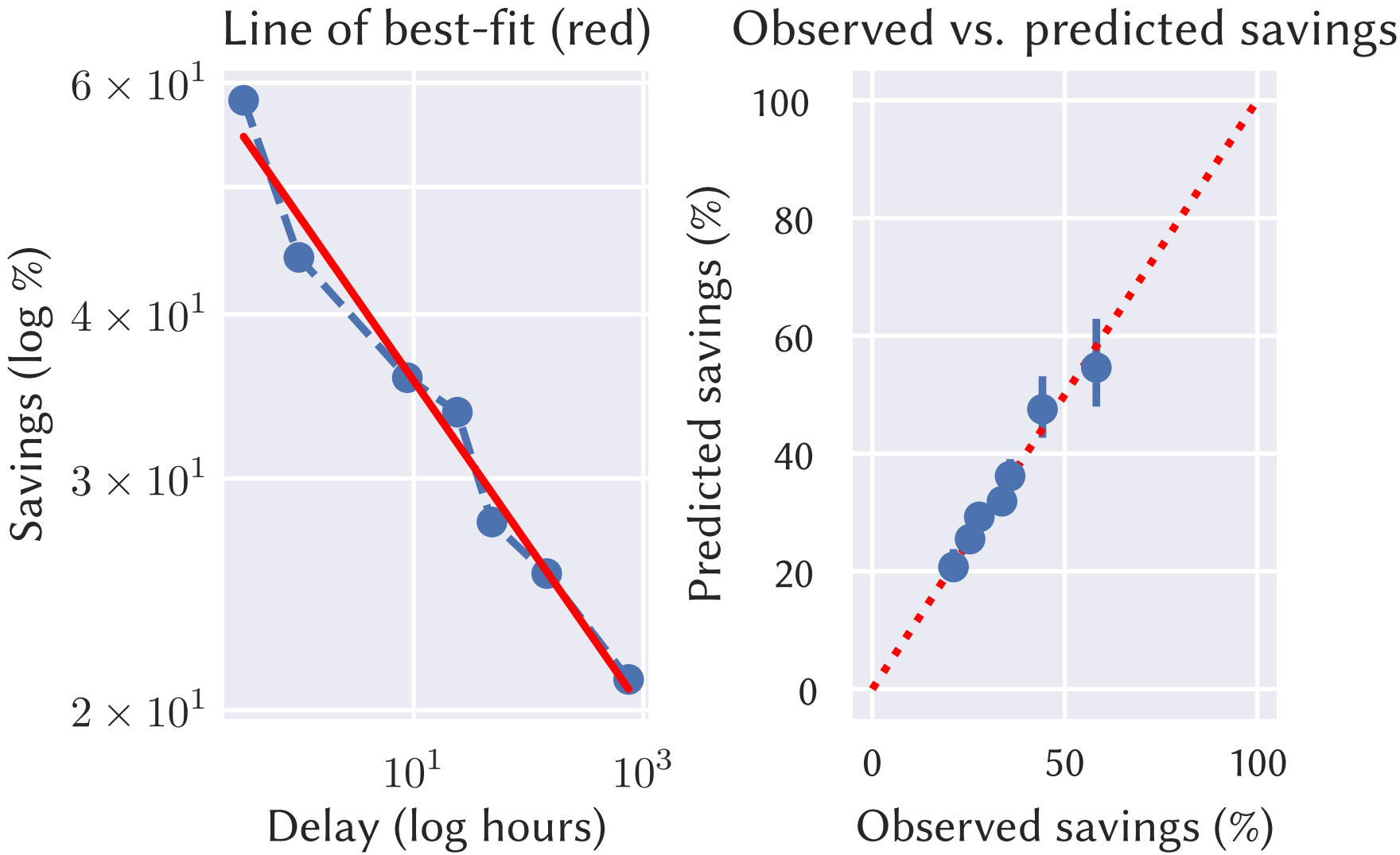
Figure: Ebbinghaus retention data: log-log



Fit power-law model to data

```
>>> power_law_model = pm.Model()
>>> with power_law_model:
...     # priors
...     intercept = pm.Normal('intercept', mu=0, sd=100)
...     slope = pm.Normal('slope', mu=0, sd=100)
...     sigma = pm.HalfNormal('sigma', sd=100)
...     # likelihood
...     mu = pm.Deterministic('mu',\
...         intercept + slope*np.log(delay))
...     log_savings = pm.Normal('log_savings', mu=mu,\
...         sd=sigma, observed=np.log(savings))
...     # posteriors
...     #db = SQLite('power_law_model_trace.sqlite')
...     #trace = pm.sample(draws=5000, trace=db,\
...                       #n_init=50000, njobs=4)
...     trace = load('power_law_model_trace.sqlite')
...
```

Figure: The power law model of forgetting



Power law & ACT-R base activation

Power law of forgetting directly incorporated in ACT-R:

(5) ACT-R base activation: $$\boxed{B_i = \log\left(\sum_{k=1}^{n} t_k^{-d}\right)}$$

(6) Exponentiating both sides:

$$e^{B_i} = \sum_{k=1}^{n} t_k^{-d} = t_1^{-d} + t_2^{-d} + \cdots + t_n^{-d}$$

- B_i: base activation of chunk i, a log-transformed measure of performance e^{B_i}
- n: number of presentations / rehearsals of chunk i
- t_k: time elapsed since presentation k
- d: decay (free parameter; default value: 0.5)

Power law & ACT-R base activation (ctd.)

Example: two presentations, one 40 s ago, one 20 s ago, let's compute base activation now – and also need odds and need probability.

$$B_i = \log(t_1^{-d} + t_2^{-d}) = \log(40^{-0.5} + 20^{-0.5})$$

$$= \log(\tfrac{1}{40^{0.5}} + \tfrac{1}{20^{0.5}}) = \log(\tfrac{1}{\sqrt{40}} + \tfrac{1}{\sqrt{20}})$$

$$= \log(\tfrac{1}{6.325} + \tfrac{1}{4.472}) = \log(0.3817) = -0.9631$$

$O_i = e^{B_i} = 0.3817$ (odds chunk i is needed now)

$P_i = \frac{O_i}{1+O_i} = \frac{0.3817}{1+0.3817} = 0.276$ (prob. chunk i is needed now)

Power law & ACT-R base activation (ctd.)

- "at any point in time, memories vary in how likely they are to be needed and the memory system tries to make available those memories that are most likely to be useful. The memory system can use the past history of use of a memory to estimate whether the memory is likely to be needed now." (Anderson and Schooler, 1991, 400)
- memory estimates / computes activations, which reflects 'need probability:' the probability that we will need a particular chunk now
- specific predictions about the relationship between activation, which is an unobserved quantity reflecting need probability, and observable / measurable quantities:
  - recall latency (how long remembering / retrieving takes)
  - recall accuracy (what is the probability of a successful retrieval)

Activation and recall accuracy & latency

Let total activation A_i be just base activation B_i.

- another component: spreading activation; we discuss it when we get to semantic modeling

(7) $P_i = \frac{1}{1+e^{-\frac{A_i-\tau}{s}}}$ (s: retrieval noise, τ: retrieval threshold)

(8) $T_i = Fe^{-fA_i}$ (F: latency factor, f: latency exponent)

- key to connection between activation and recall accuracy and latency: understand the specific way in which activation reflects need probability
- base activation B_i is the logit (log-odds) transformation of need probability: $B_i = \log(O_i)$.
- exponentiated activation e^{B_i} is the odds that chunk i is needed (need odds $O_i = \frac{P_i}{1-P_i}$, where P_i is the need probability of chunk i)

Activation and recall accuracy & latency (ctd.)

ACT-R base activation (exponentiated): $e^{B_i} = \sum_{k=1}^{n} t_k^{-d}$

$\sum_{k=1}^{n}$: individual presentations 1 through n of a chunk i have a strengthening impact on the need odds of chunk i

- a presentation k additively increases the previous need odds for chunk i
- these impacts are summed up to produce a total strength / total need odds for chunk i

t_k^{-d}: the strengthening impact of a presentation k on the total need-odds for the chunk is a power function of time t_k^{-d}

- t_k is the time elapsed since presentation k
- this encodes the power law of forgetting

Activation and recall accuracy & latency (ctd.)

Probability of retrieval (ignoring free param.s):

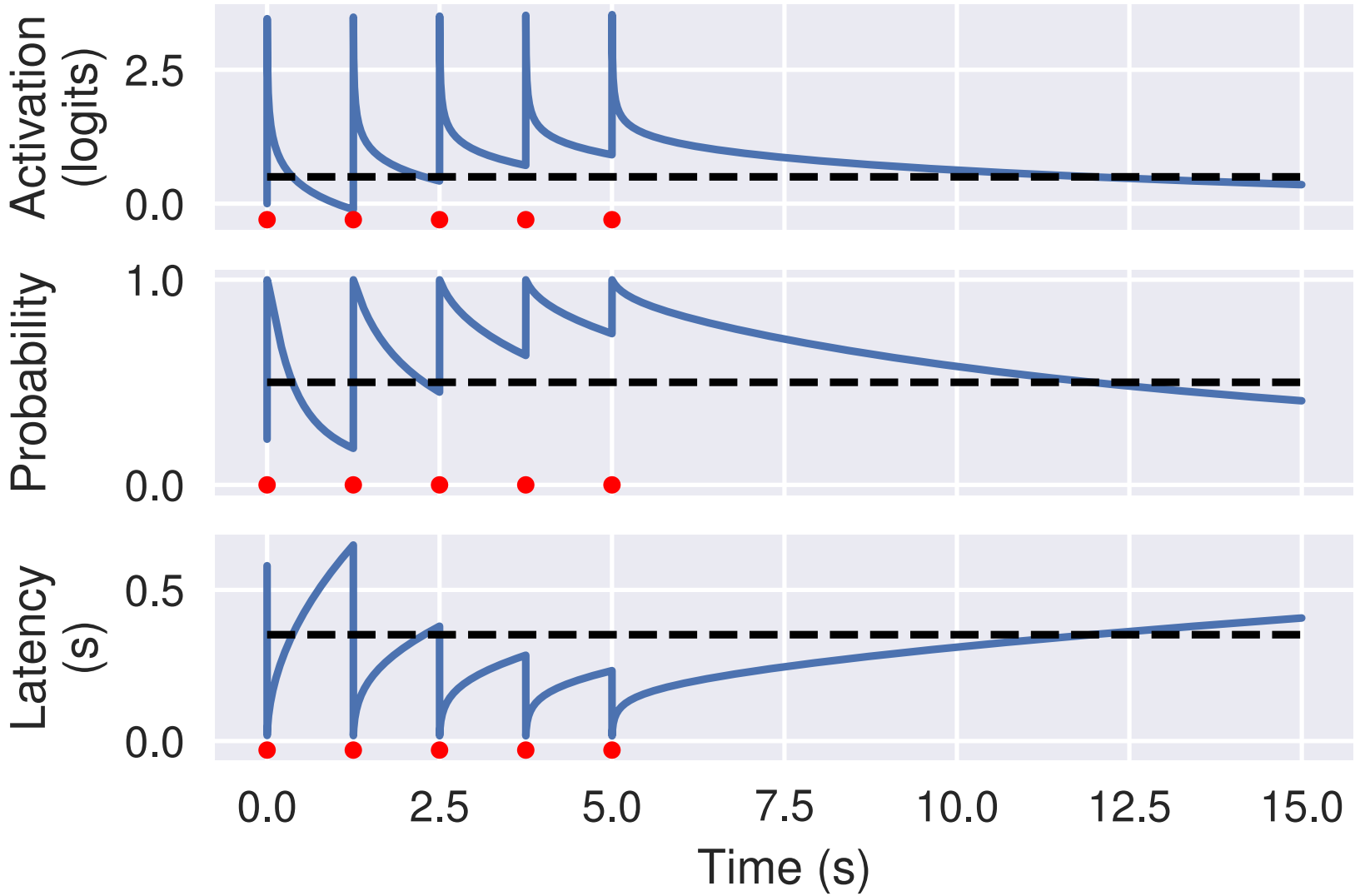
$$P_i = \frac{1}{1+e^{-A_i}} = \frac{1}{1+\frac{1}{e^{A_i}}} = \frac{1}{1+\frac{1}{O_i}} = \frac{O_i}{1+O_i}$$

- P_i: need probability for chunk i (probability that we need i at retrieval time)
- $O_i = \frac{p_i}{1-p_i}$: need odds for chunk i (odds that we need i at retrieval time)
- conversely: $P_i = \frac{O_i}{1+O_i}$
- $A_i = \log(O_i) = \log\left(\frac{p_i}{1-p_i}\right)$: need logits/log-odds for chunk i (log-odds that we need i at retrieval time)
- conversely: $O_i = e^{A_i}$
- it follows that: $p_i = \frac{O_i}{1+O_i} = \frac{1}{1+\frac{1}{O_i}} = \frac{1}{1+\frac{1}{e^{A_i}}} = \frac{1}{1+e^{-A_i}}$

Latency of retrieval (ignoring free param.s):

$$T_i = e^{-A_i} = \frac{1}{e^{A_i}} = \frac{1}{O_i}$$

Figure: Activation, retrieval prob. and retrieval latency as a function of time (threshold – dotted black line; 5 presentations – red)



Linguistic example: self-paced reading

For full paper, see Brasoveanu and Dotlačil (2018, to appeara)

- Grodner and Gibson (2005, Exp. 1): self-paced reading, matrix subject is modified by a subject or object-extracted relative clause (RC)

(9) The
reporter who sent the photographer to the editor hoped
for a story.

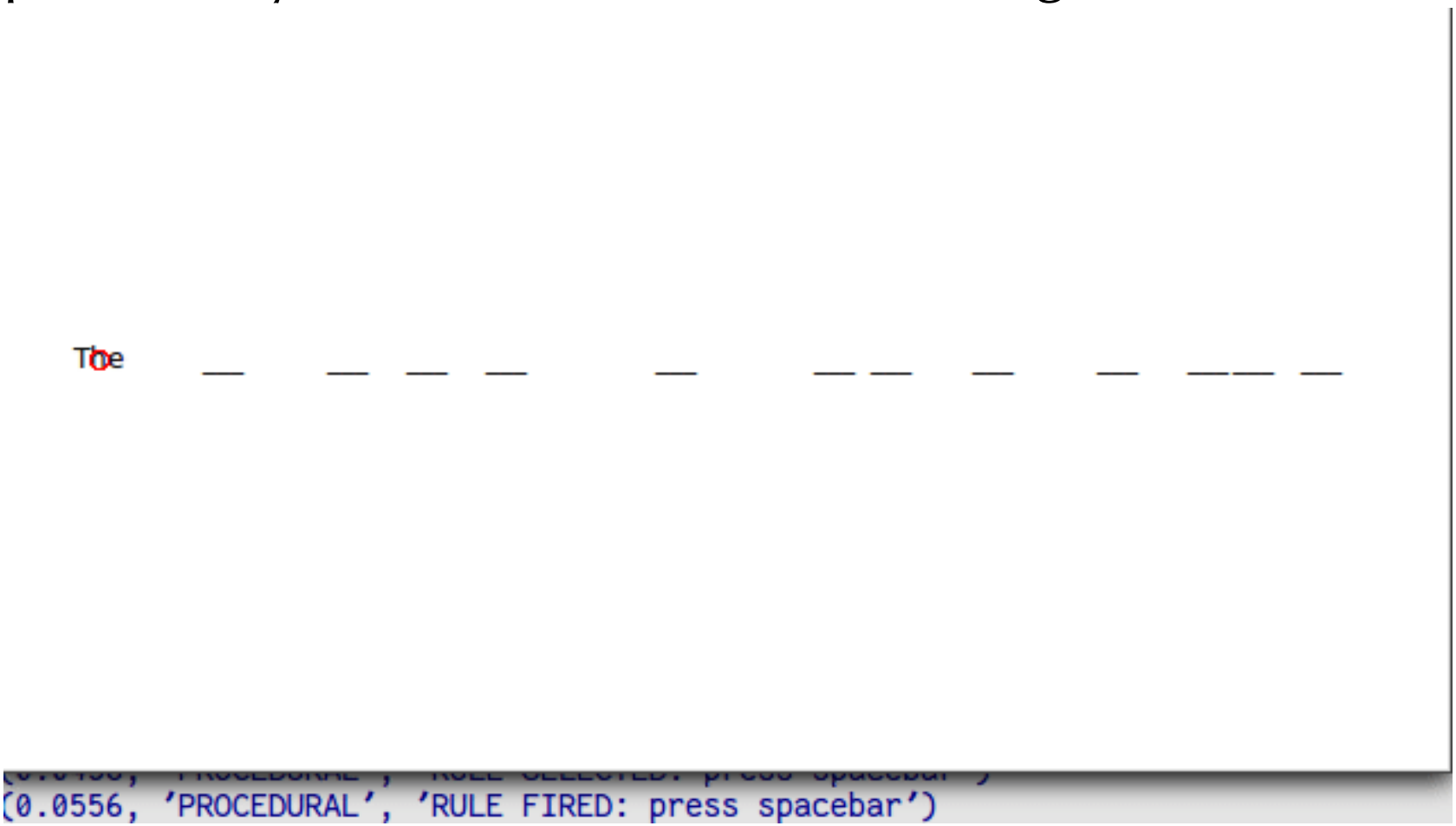
(10) The
reporter who the photographer sent to the editor hoped
for a story.

9 ROIs: word 2 through word 10 (underlined above)

Demo of an ACT-R model for subj & obj gap RCs

(open the slides with Adobe Acrobat Reader to see the movie)

Red circle is the visual focus. Temporal trace incrementally produced by the model is visible in the background.



An eager left-corner parser in ACT-R

Building on Resnik (1992); Lewis and Vasishth (2005)

Left-corner parser components:

- lexical knowledge (simulating adult-like experience), knowledge of already parsed syntactic structures → declarative memory
- knowledge of grammar → procedural memory (common, e.g., Lewis and Vasishth 2005)
- expectations about upcoming syntactic categories, which guide parsing → goal buffer
- information about the current syntactic parse → imaginal buffer
- visual information from environment → visual buffer
- key press commands → manual buffer
- visual module – EMMA Salvucci (2001)
- motor module – EPIC Kieras and Meyer (1996); Meyer and Kieras (1997)

An eager left-corner parser in ACT-R

Rules:

$S \rightarrow NP\ VP$

$NP \rightarrow Det\ N$

$VP \rightarrow V$

Visual input:

- A boy sleeps.

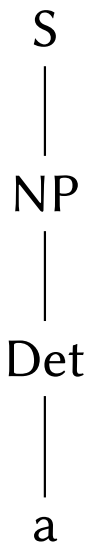
`A --- ------.`

Input

- Stack: S (Goal)
- Found: *a*, Det (Visual + Retrieval)

Output

- Stack: N NP S (Goal)
- Structure:



An eager left-corner parser in ACT-R

Rules:
S $\rightarrow$ NP VP
NP $\rightarrow$ Det N
VP $\rightarrow$ V

Visual input:

- A boy sleeps.

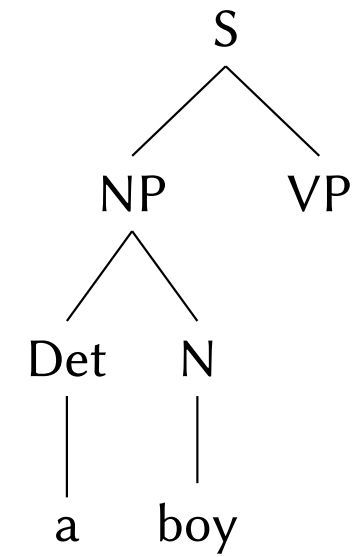
`- boy ------.`

Input

- Stack: N NP S (Goal)
- Found: *boy*, N
  (Visual + Retrieval)

Output

- Stack: VP (Goal)
- Structure:



An eager left-corner parser in ACT-R

Rules:
S $\rightarrow$ NP VP
NP $\rightarrow$ Det N
VP $\rightarrow$ V

Visual input:

- A boy sleeps.

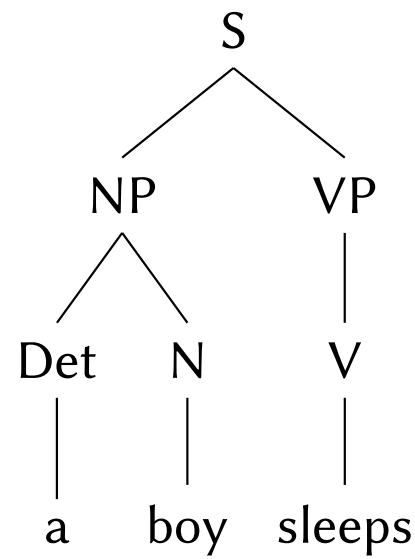
```
- --- sleeps.
```

Input

- Stack: VP (Goal)
- Found: *sleeps*, V (Visual + Retrieval)

Output

- Stack: {} (Goal)
- Structure:



An eager left-corner parser in ACT-R and gaps

(11) The reporter who...

- parser postulates subject gap at this moment

An eager left-corner parser in ACT-R and gaps

(11) The reporter who...

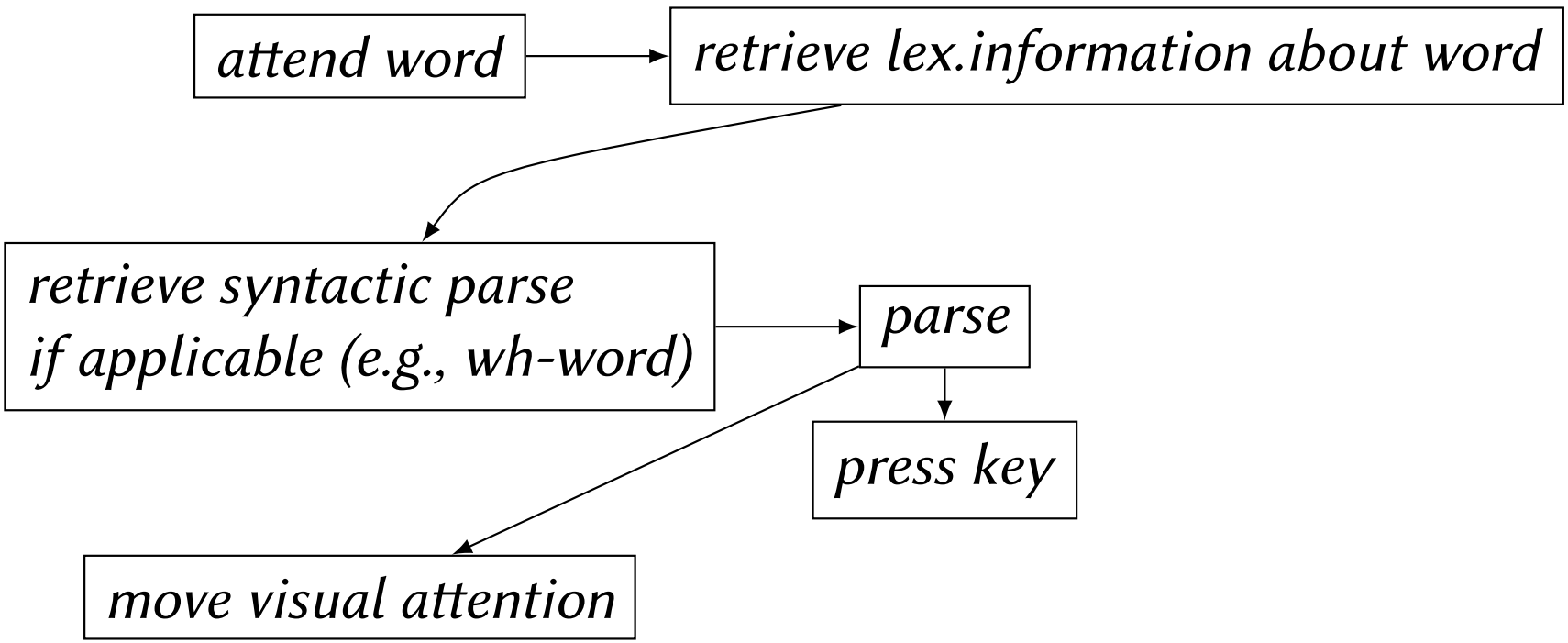
- parser postulates subject gap at this moment

(12) The reporter who sent the photographer to the editor...

(13) The reporter who the photographer sent...

- parser postulates object gap for object-relative clause (i.e., (13))

Flow chart of parsing process per word



Parameters – visual encoding (EMMA)

- Visual encoding (T_{enc}) dependent on visual distance d and object properties, D

$$T_{enc} = K \cdot D \cdot e^{kd} (\text{parameter } k - \text{angle})$$

- $D =$ word length, $K = 0.01$
- k – estimated to show that parameters for peripherals can be estimated at the same time as the more commonly estimated parameters associated with declarative and procedural memory

Parameters – rule firing and memory recall

- Rule firing = r (parameter r)

  How much time does each rule take? (default: 50 ms)
- Retrieval latency, modulated by parameters F (latency factor) and f (latency exponent)

$$T = F \cdot e^{-f \cdot A}$$

- latency exponent f – estimated because crucial in estimating latencies in lexical decision tasks (cf. Brasoveanu and Dotlačil to appearb)

Estimation

- The model is fit to data by estimating the 4 free parameters (k, r, F, f)
- **pyactr** enables us to easily interface ACT-R models with standard statistical estimation methods implemented in widely-used Python3 libraries
- we use ACT-R models as the likelihood component of full Bayesian models, and fit the ACT-R parameters to experimental data

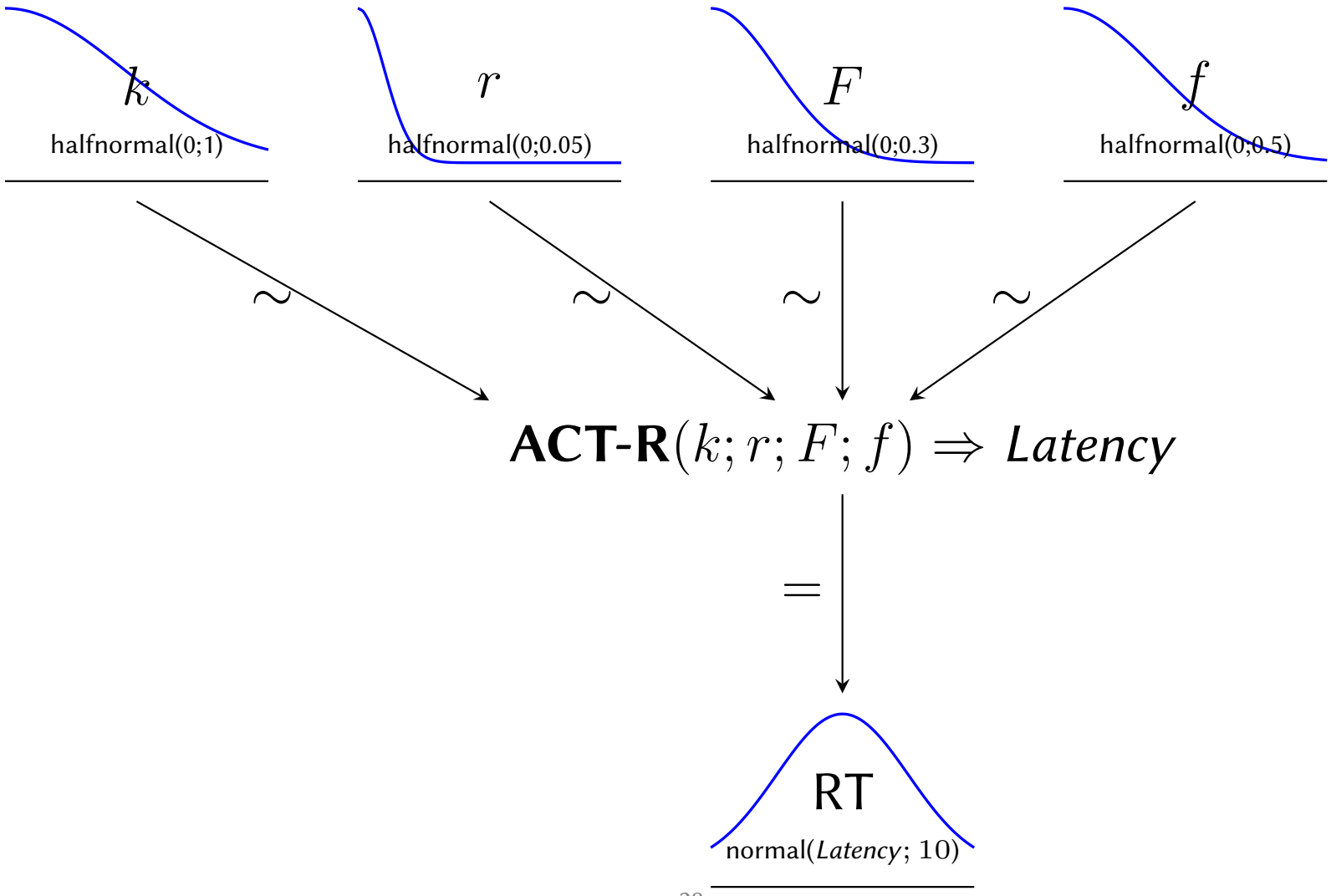
Data

```
>>> import pymc3 as pm

>>> subj_extraction = np.array(\
...     [360.2, 349.8, 354.8, 334.3, 384,\
...     346.5, 318.4, 403.6, 404.6])

>>> obj_extraction = np.array(\
...     [373.1, 343, 348.1, 357.6, 422.1,\
...     375.8, 338.6, 482.9, 401.1])
```

Model implementation

```
>>> parser_with_bayes = pm.Model()                                          1
>>> with parser_with_bayes:                                                 2
...     # priors for latency                                               3
...     F = pm.HalfNormal('F', sd=0.3)                                     4
...     f = pm.HalfNormal('f', sd=0.5)                                     5
...     r = pm.HalfNormal('r', sd=0.05)                                    6
...     k = pm.HalfNormal('k', sd=1.0)                                     7
...     # latency likelihood -- this is where pyactr is8us
...     pyactr_rt = actrmodel_latency(F, f, r, k)                          9
...     subj_rt_observed = pm.Normal(\                                     10
...     'subj_rt_observed', mu=pyactr_rt[0], sd=10,\                       11
...     observed=subj_extraction)                                          12
...     obj_rt_observed = pm.Normal(\                                      13
...     'obj_rt_observed', mu=pyactr_rt[1], sd=10,\                        14
...     observed=obj_extraction)                                           15
...                                                                        16
```

Bayesian model structure

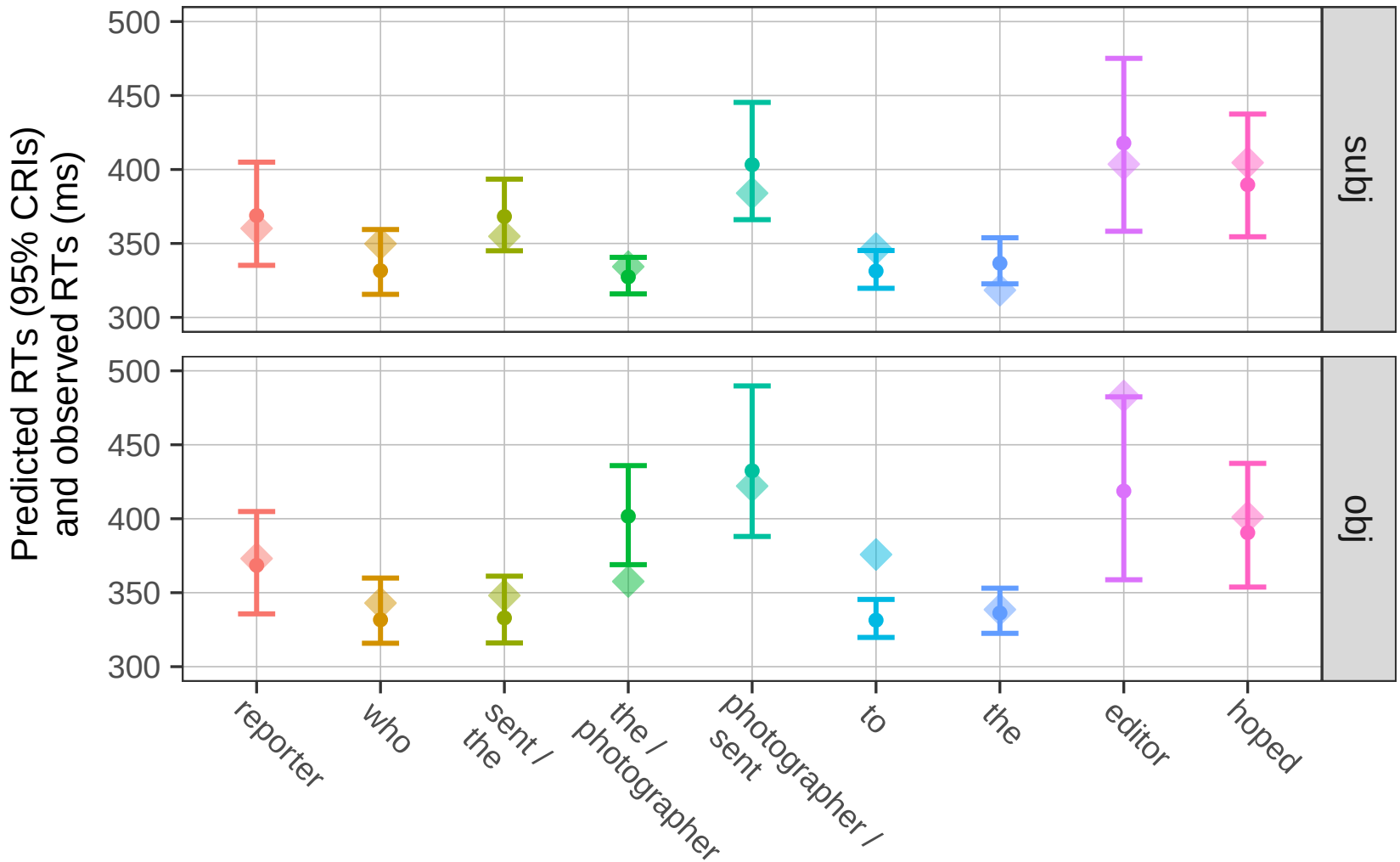


The power law of forgetting

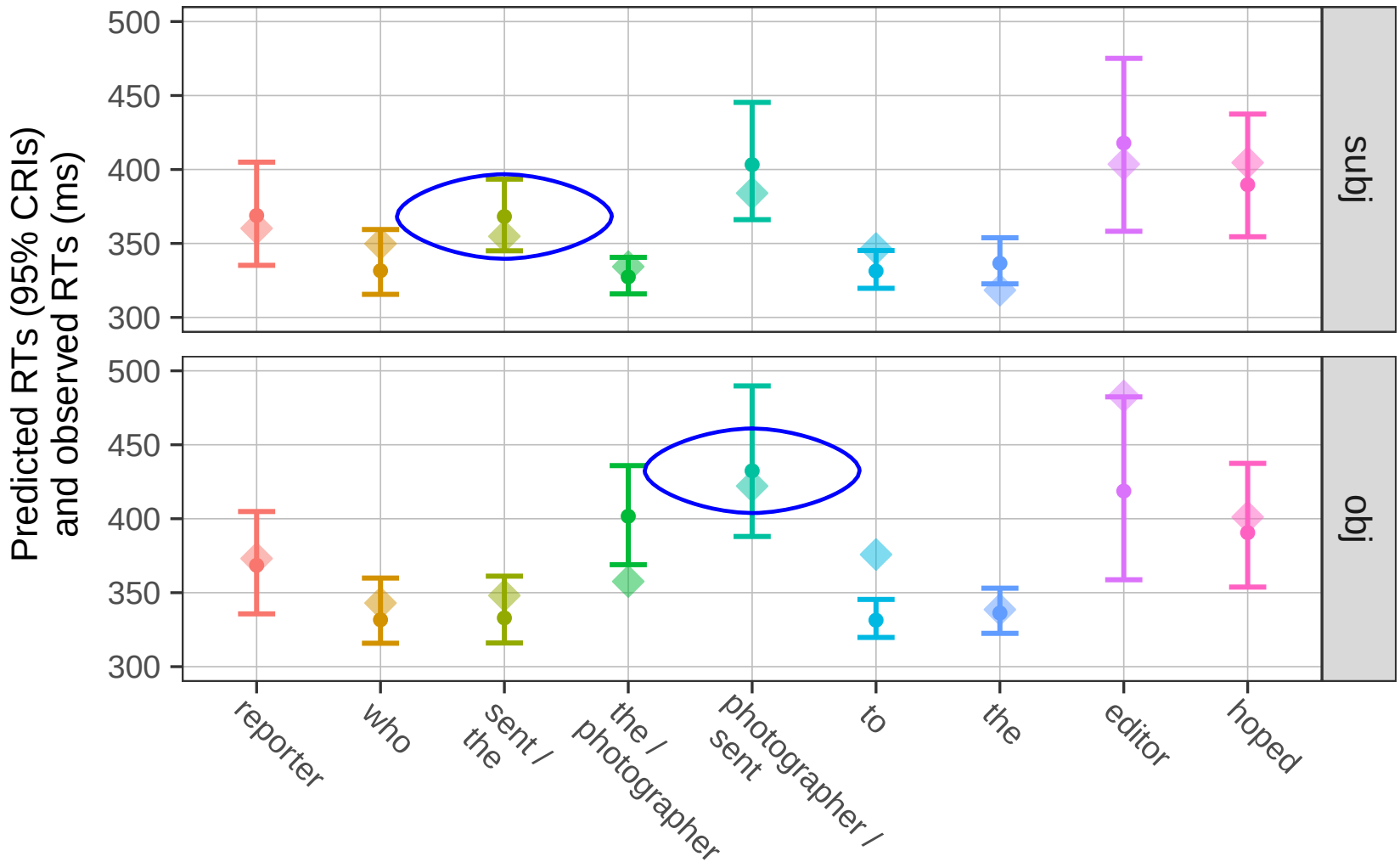
Results

Conclusion

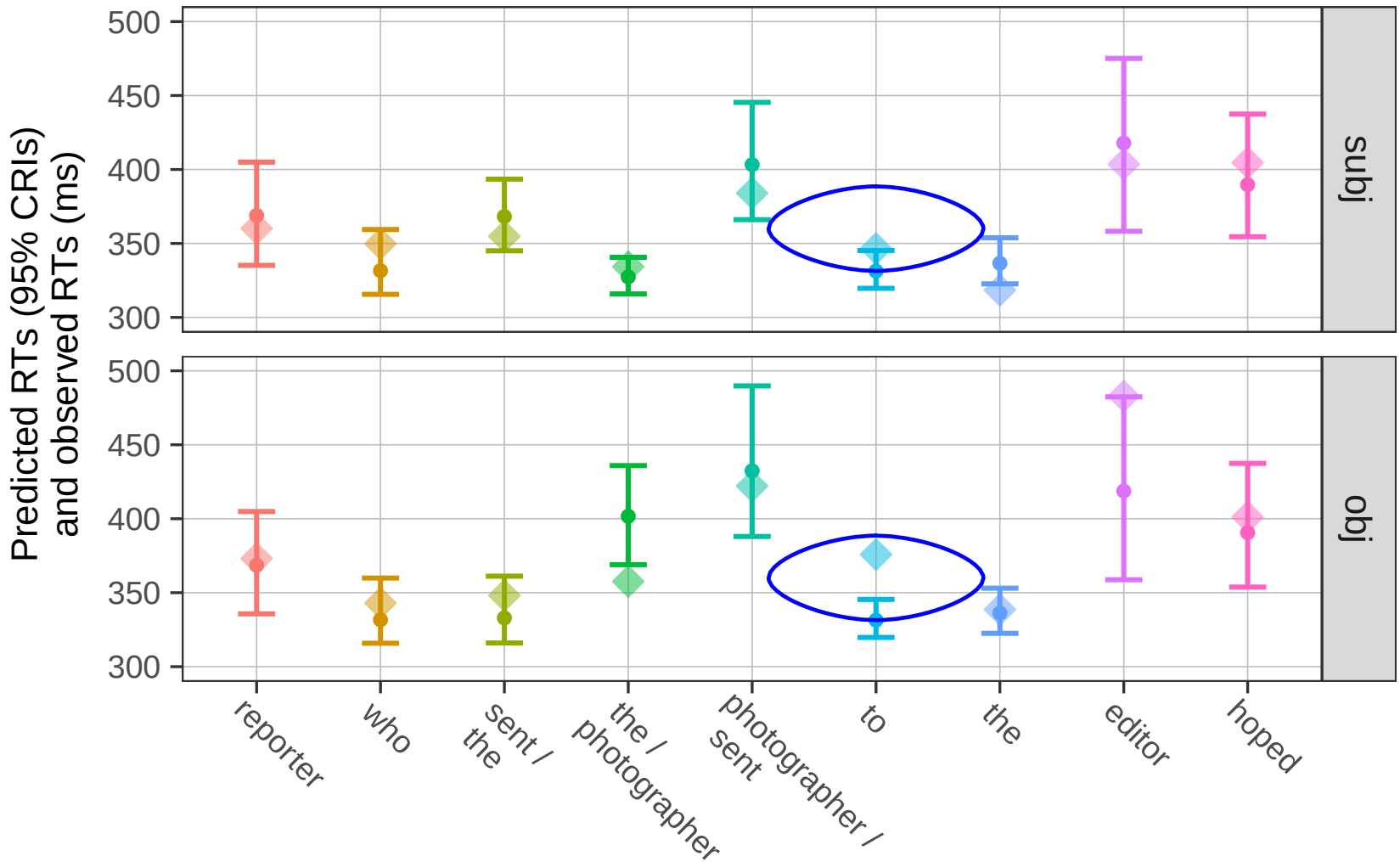
Posterior predictions (Model 1)



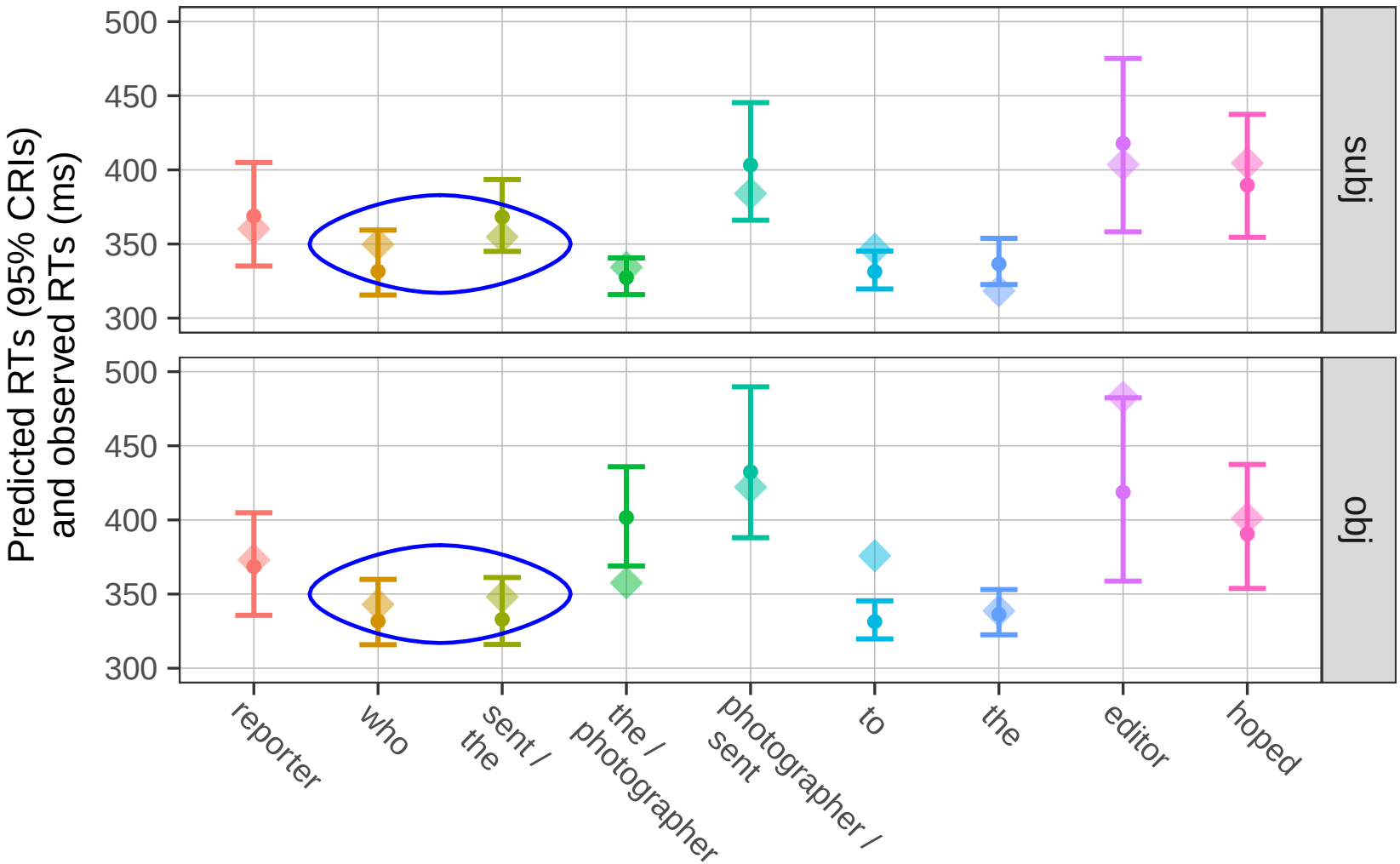
Posterior predictions (Model 1)



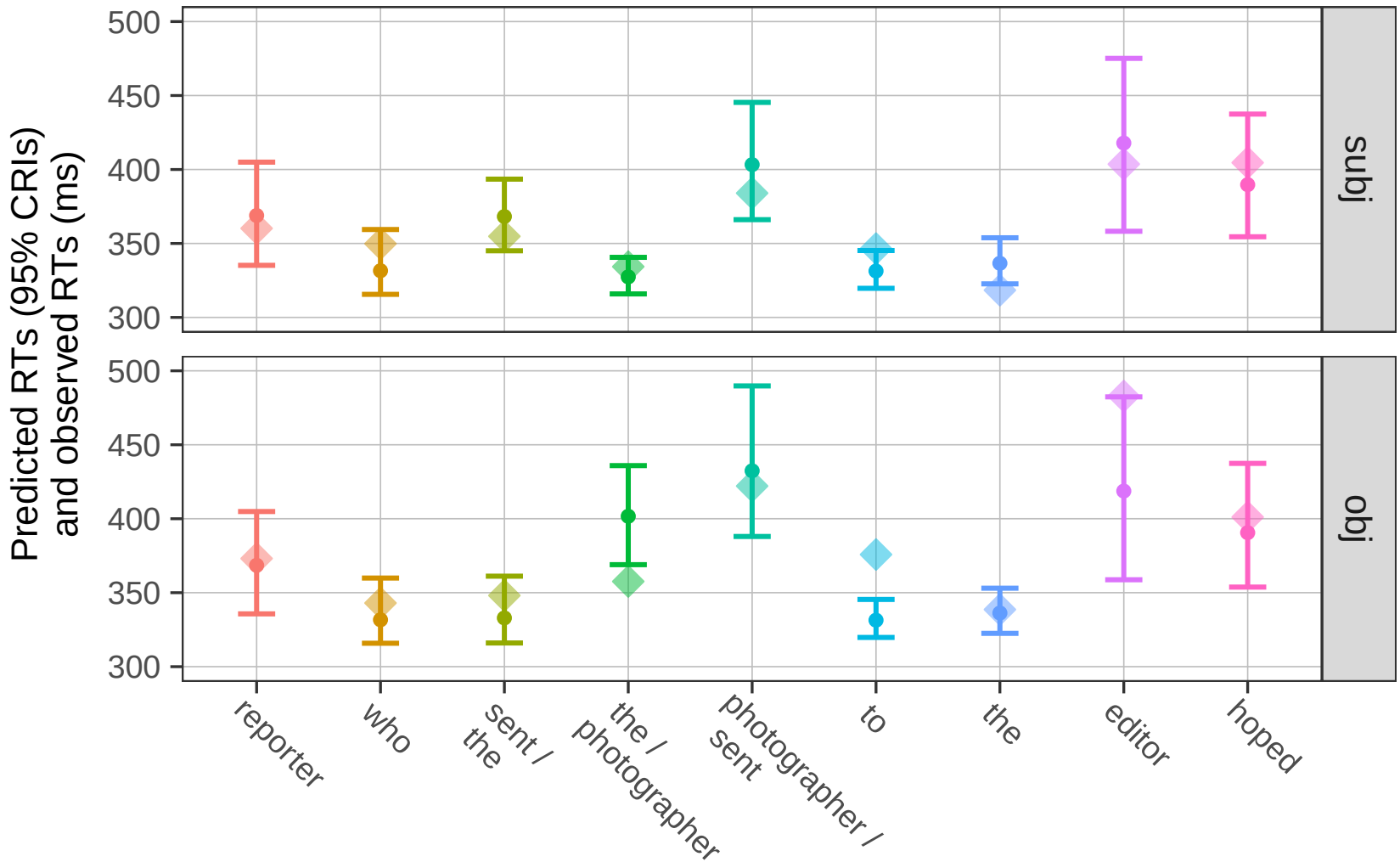
Posterior predictions (Model 1)



Posterior predictions (Model 1)

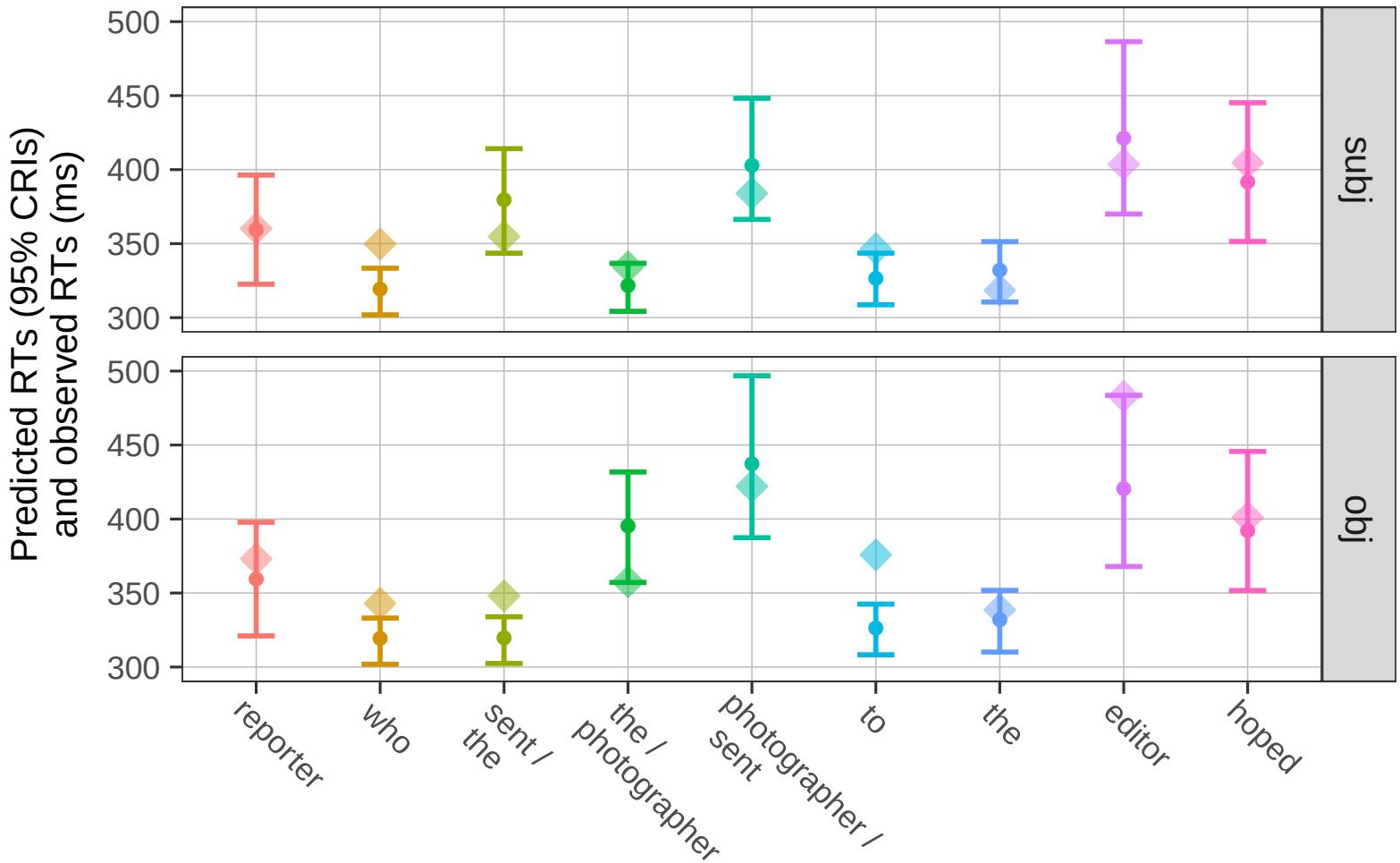


Posterior predictions (Model 1)

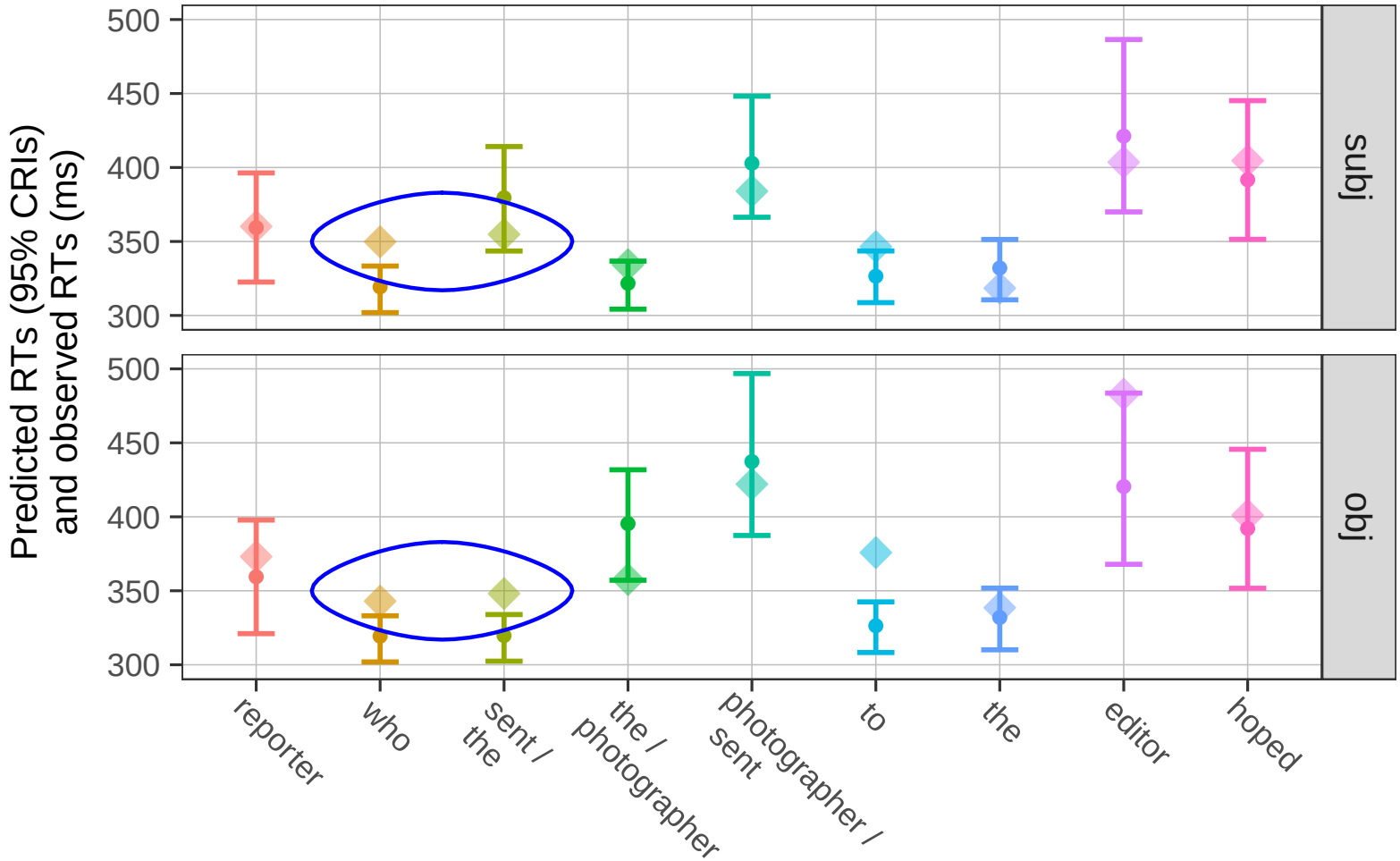


Model 2: no postulated subject gaps

Model 2: no postulated subject gaps

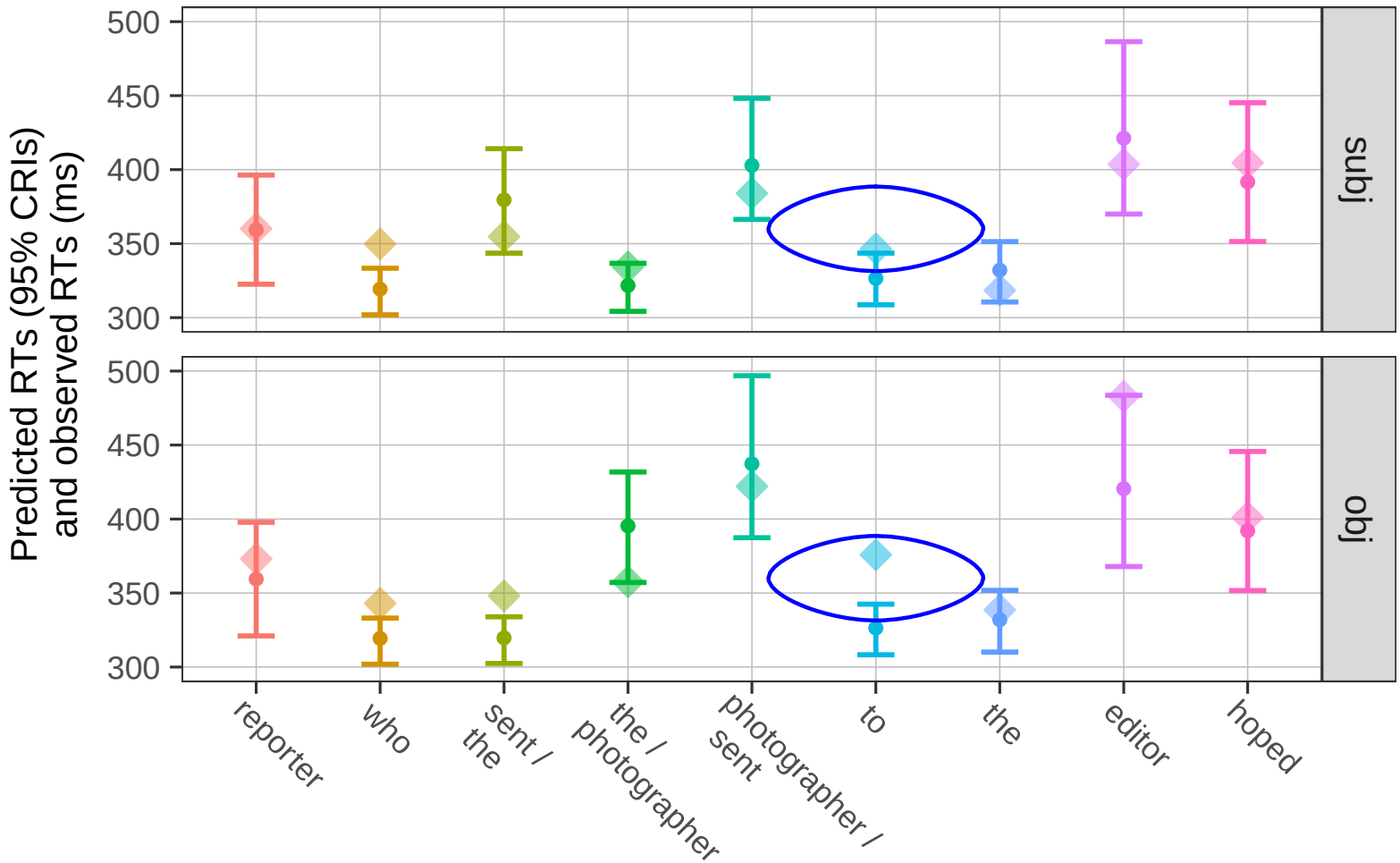


Model 2: no postulated subject gaps

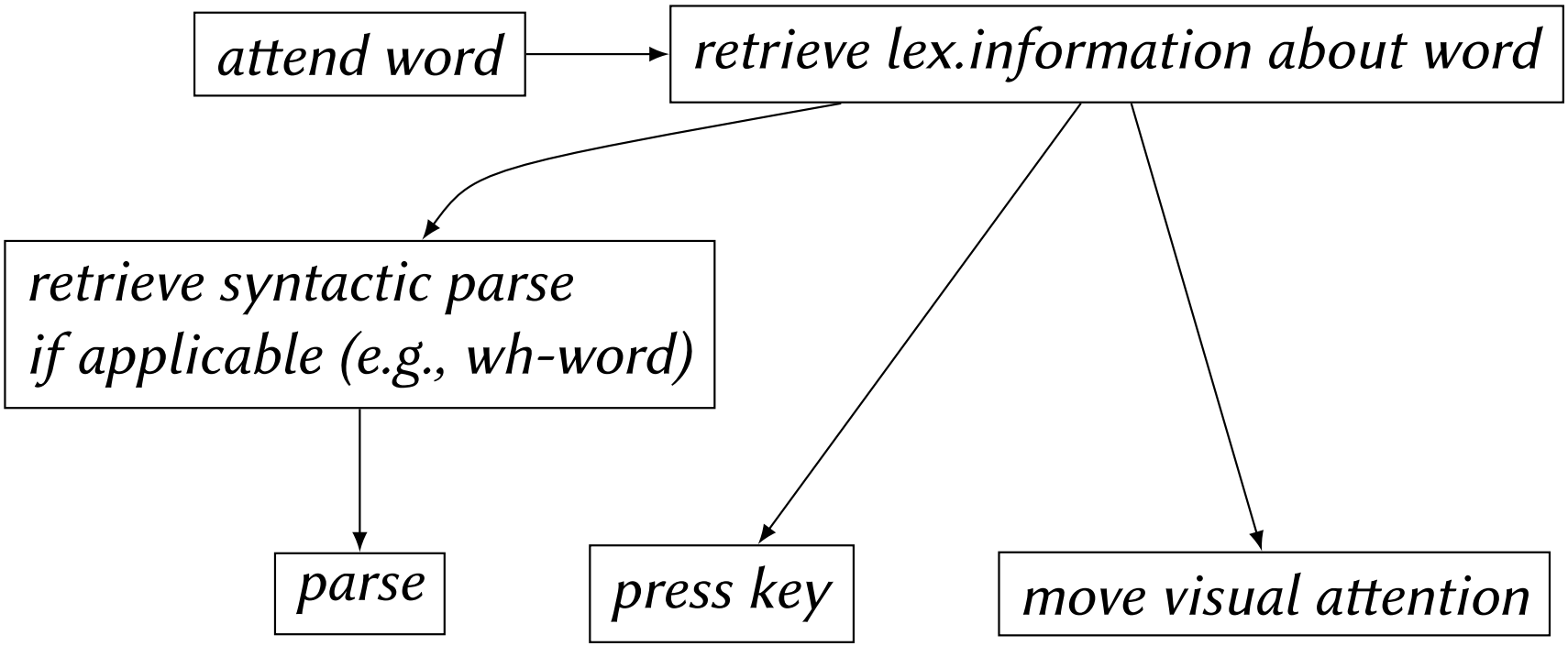


wh-word and following word not modeled well; Model 1 better

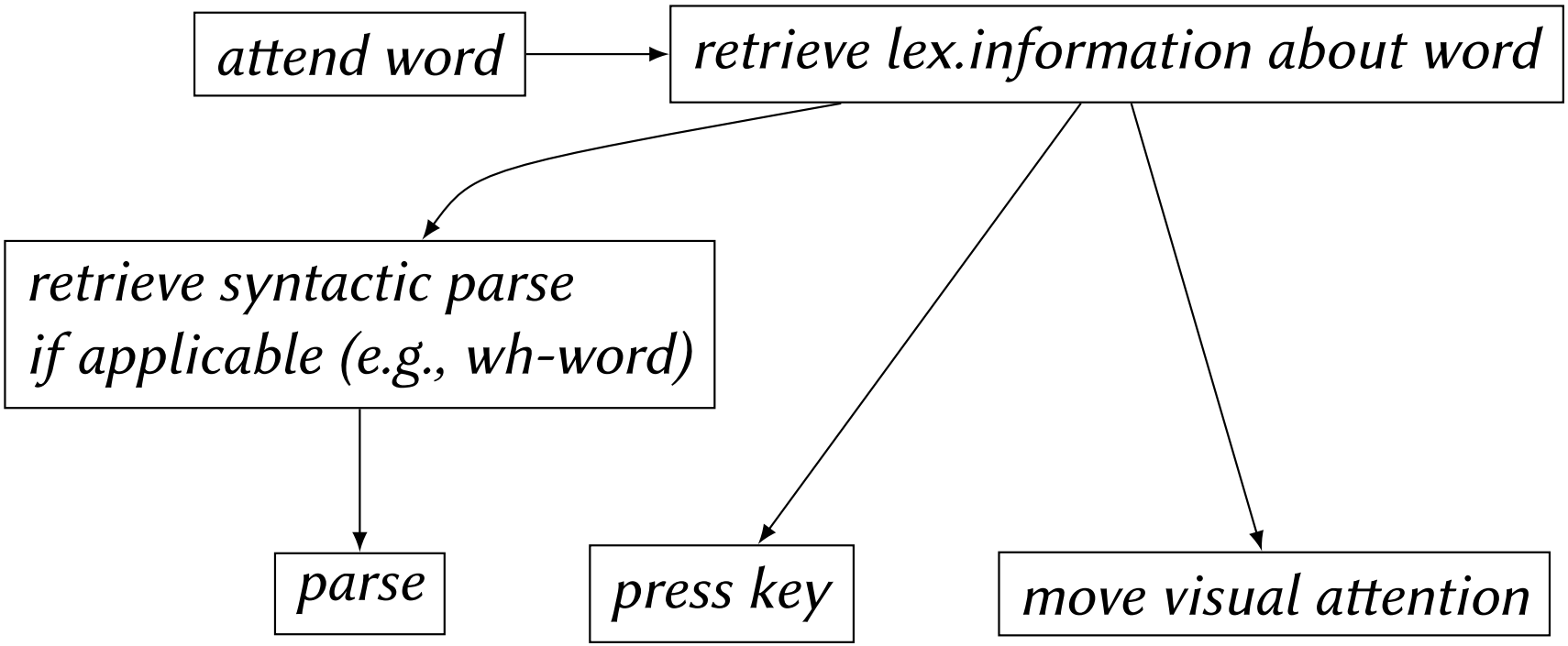
Model 2: no postulated subject gaps



Model 3: parallel reader

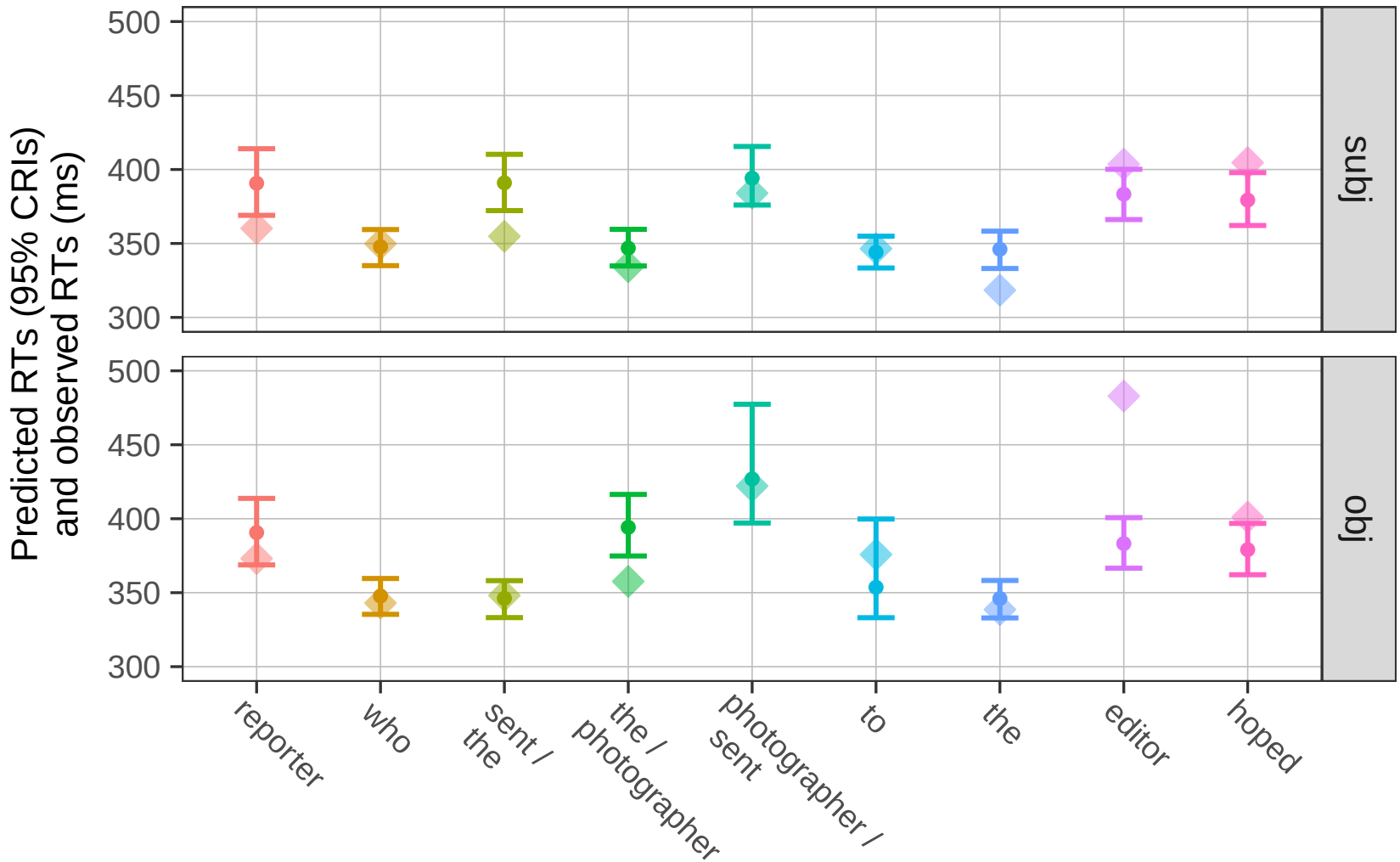


Model 3: parallel reader

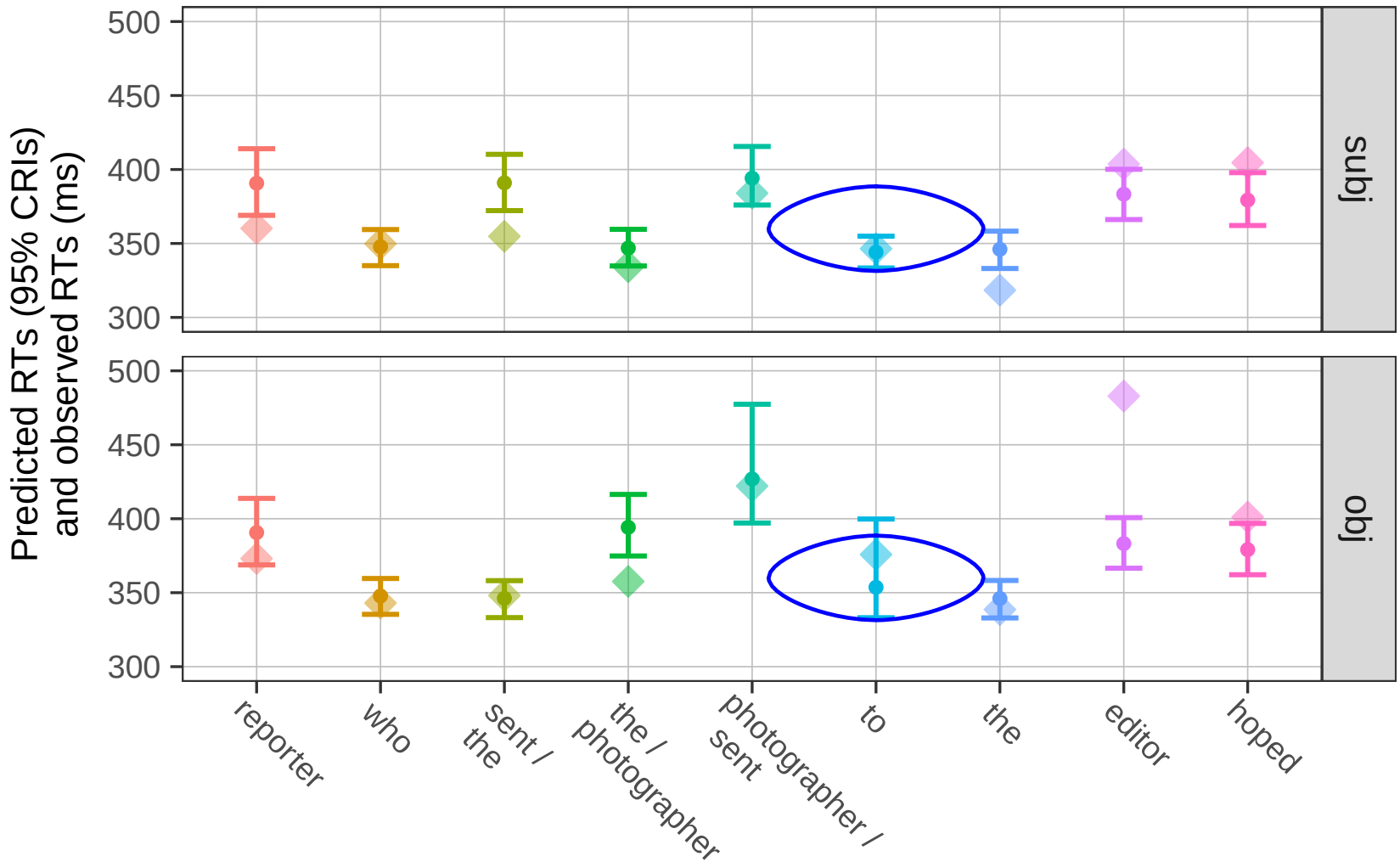


- Model 1 completes all available parsing before key press (serial)
- Model 3: first lexical retrieval, then structure building & key press in parallel
- Outcome: spillover on word after object gap captured

Model 3: spillover after object gap captured



Model 3: spillover after object gap captured



WAIC-based model comparison

	WAIC_1	WAIC_2
Model 1 (subject gaps)	388	1469
Model 2 (no subject gaps)	433	1613
Model 3 ('parallel' reader)	390	553

- Model 1 is better than Model 2 with respect to both WAIC_1 and WAIC_2
- increase in precision for Model 3 is clearly visible in its much lower WAIC_2 value (which is variance based)

Conclusion

- we introduced a modular and extensible framework for mechanistic processing models
- case study: an incremental left-corner parser with visual and motor interfaces for subject/object gap relative clauses
- framework used to quantitatively compare hypotheses about processing, e.g., predictively postulating subject gaps
- systematic across-the-board model comparison via Bayes factors is possible in this framework
- framework can model other tasks (eye tracking, lexical decision – for the latter, see Brasoveanu and Dotlačil to appearb)

Anderson, John R., and Lael J. Schooler. 1991. Reflections of the environment in memory. *Psychological Science* 2:396–408.

Brasoveanu, Adrian, and Jakub Dotlačil. 2018. An extensible framework for mechanistic processing models: From representational linguistic theories to quantitative model comparison. In *Proceedings of the 2018 International Conference on Cognitive Modelling*.

Brasoveanu, Adrian, and Jakub Dotlačil. to appeara. An extensible framework for mechanistic processing models: From representational linguistic theories to quantitative model comparison. *Topics in Cognitive Science* .

Brasoveanu, Adrian, and Jakub Dotlačil. to appearb. Quantitative comparison for generative theories: Embedding competence linguistic theories in cognitive architectures and bayesian models. In *Proceedings of the 2018 Berkeley Linguistic Society 44*.

Ebbinghaus, Hermann. 1913. *Memory: A contribution to experimental psychology*. New York: Teachers College, Columbia University. URL `http://psychclassics.yorku.ca/Ebbinghaus/index.htm`.

Grodner, Daniel, and Edward Gibson. 2005. Consequences of the serial nature of linguistic input for sentenial complexity. *Cognitive Science* 29:261–291.

Kieras, David E, and David E Meyer. 1996. The epic architecture: Principles of operation. Unpublished manuscript from ftp://ftp. eecs. umich. edu/people/kieras/EPICarch. ps.

Lewis, Richard, and Shravan Vasishth. 2005. An activation-based model of sentence processing as skilled memory retrieval. *Cognitive Science* 29:1–45.

Meyer, David E, and David E Kieras. 1997. A computational theory of executive cognitive processes and multiple-task performance: Part i. basic mechanisms. *Psychological review* 104:3.

Reitter, David, Frank Keller, and Johanna D Moore. 2011. A computational cognitive model of syntactic priming. *Cognitive science* 35:587–637.

Resnik, Philip. 1992. Left-corner parsing and psychological plausibility. In *Proceedings of the Fourteenth International Conference on Computational Linguistics*. Nantes, France.

Salvucci, Dario D. 2001. An integrated model of eye movements and visual encoding. *Cognitive Systems Research* 1:201–220.

Model 1: parameter estimates

- $k = 0.87, sd = 0.32$
- $F = 0.01, sd = 0.03$
- $f = 0.23, sd = 0.47$
- $r = 0.02, sd = 0.006$