



Improving inference about cognitive processes using mixture models.

Gidon Frischkorn & Vencislav Popov





Workshop Agenda

- I. What are Mixture Models and why can they be useful?
- II. Specifying mixture models in brms
 - a) Data formatting
 - b) Setting up mixture families
 - c) Understanding & Identifying parameters of mixture families
 - d) Fitting & Summarizing results of mixture models
- III. bmm – Easy implementation of mixture models for visual working memory tasks

--- Coffee Break ---

- IV. Work with (your own) data
- V. Outlook: Specifying custom mixture models for accuracy

What will you (not) learn today?



- How to specify simple mixture models in brms
- How to use the *bmm* package to fit existing mixture models for visual working memory tasks
 - Two-parameter (Zhang & Luck, 2008)
 - Three-parameter (Bays et al., 2009)
 - Different flavors of the Interference Measurement model (Oberauer & Lin, 2017)
- Interpret & Summarize results of mixture models estimated using brms



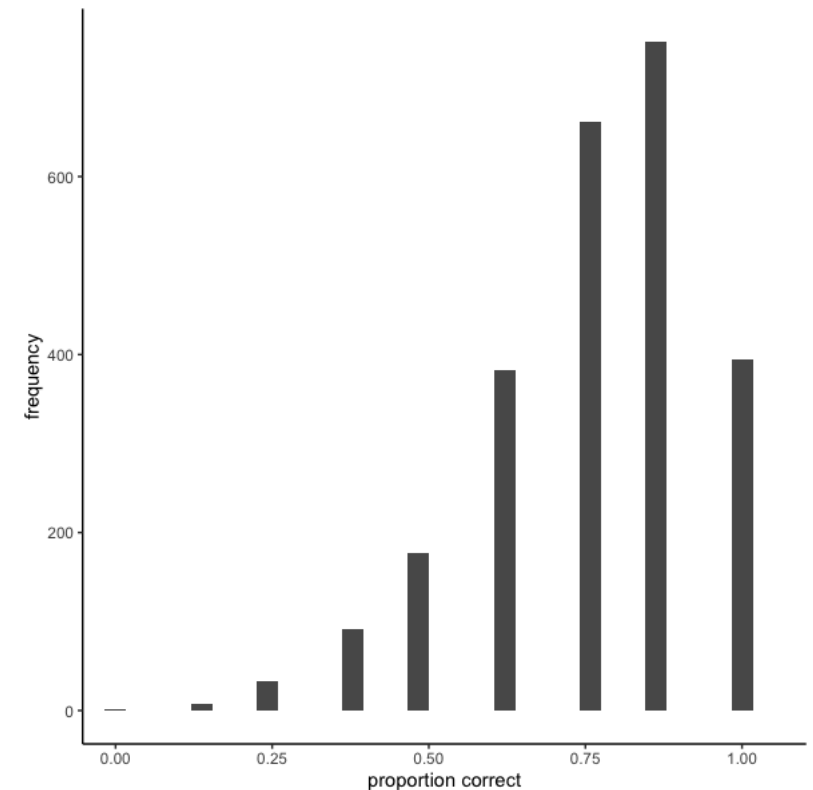
- How to specify complicated mixture models or develop entirely new models
- How to fit mixture models for groups of subjects that differ in their behavior



What are Mixture Models and why can they be useful?

Assumptions in (standard) data analysis:

1. DV stems from a single distribution (oftentimes a normal distribution)
2. predict parameters (usually the mean) from these distributions by independent variables





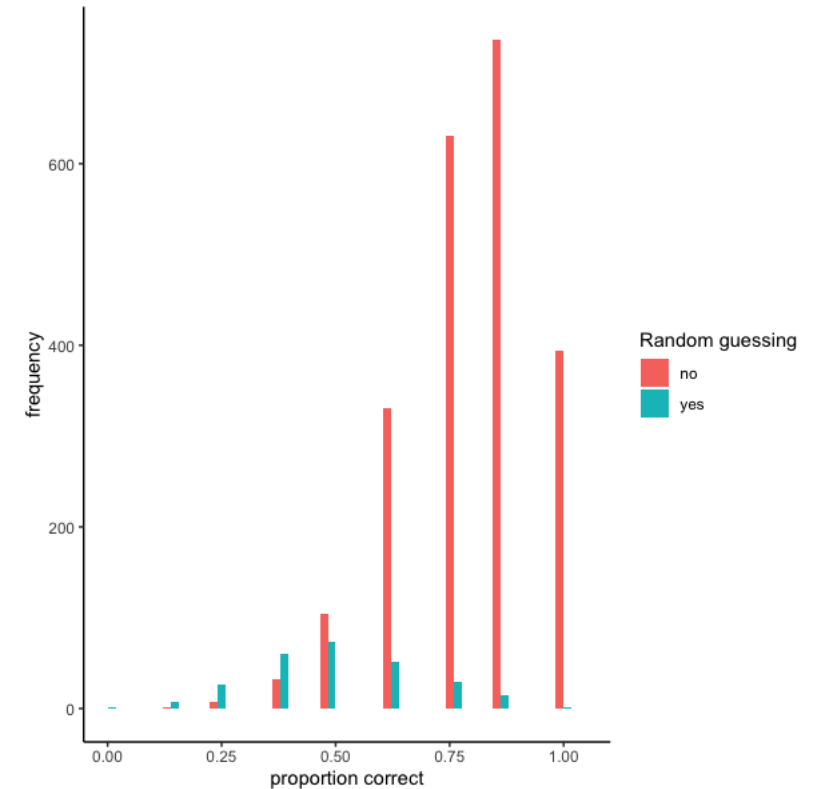
What are Mixture Models and why can they be useful?

Assumptions in (standard) data analysis:

1. DV stems from a single distribution (oftentimes a normal distribution)
2. predict parameters (usually the mean) from these distributions by independent variables

Problem:

→ Sometime data do stem from multiple different distributions





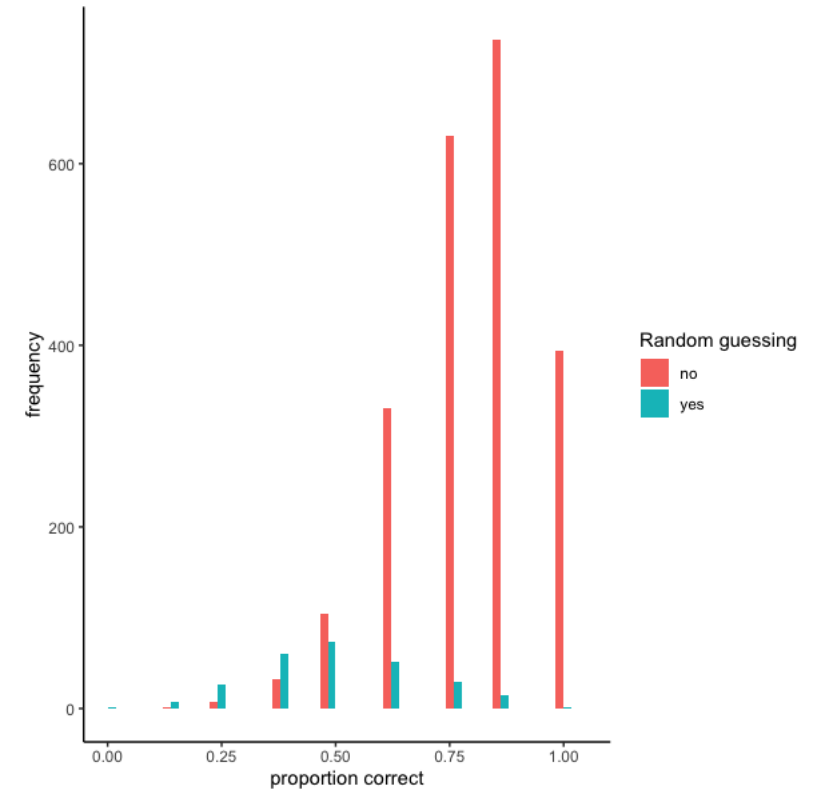
What are Mixture Models and why can they be useful?

Mixture Models...

- ...specify a set of distributions that data can stem from
- ...allow to estimate what proportion of data stems from each distribution
- ...enable to predict parameters of the different distributions

Such mixtures can occur...

- ...within a participant (different cognitive states or sources of signal)
- ...between participants



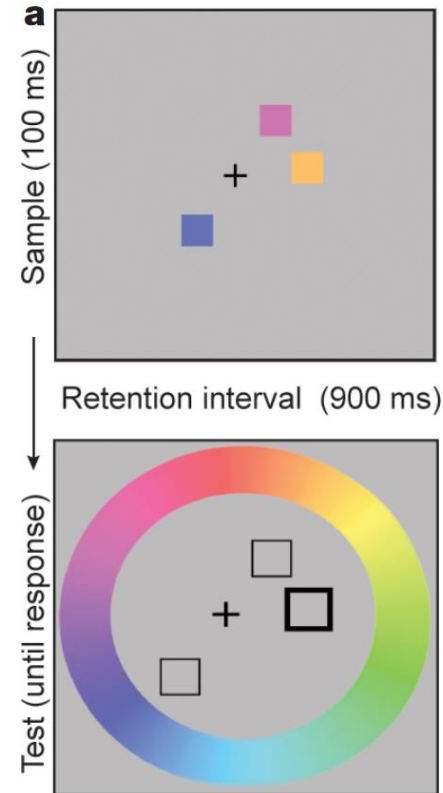
What are Mixture Models and why can they be useful?

Application to visual working memory

- Theories separate different states we can be at during retrieval
 - a) encoded the item in memory → retrieval with the precision of memory representations
 - b) not encoded the item in memory → random guessing

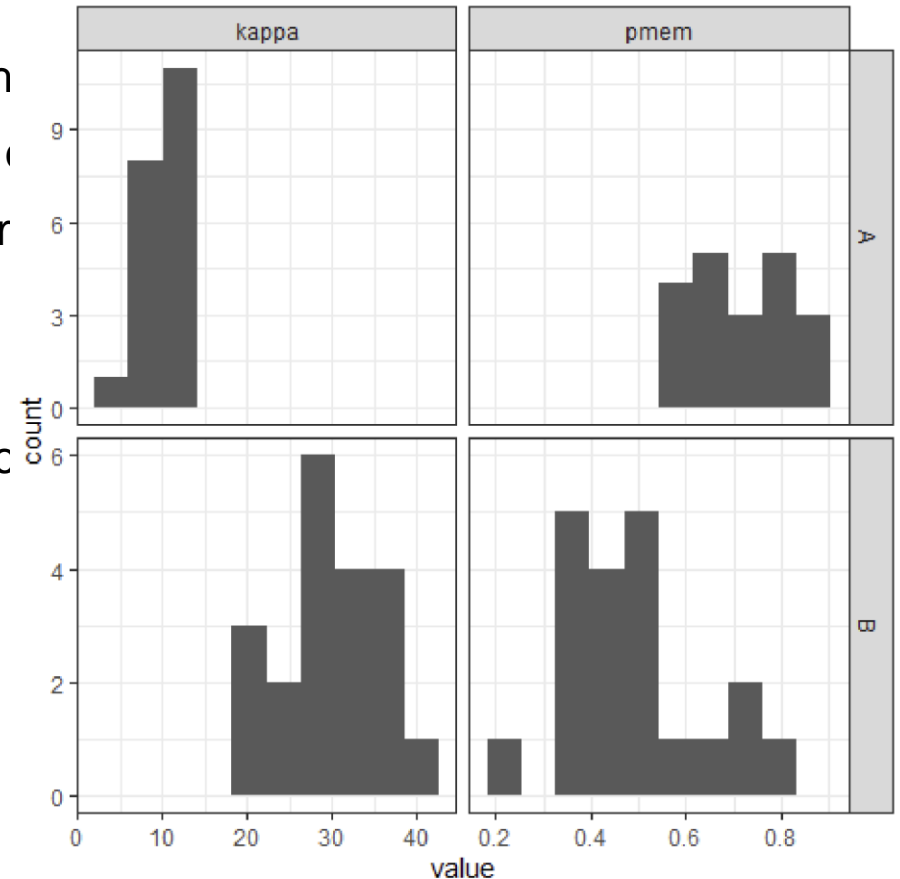
Performance indicators:

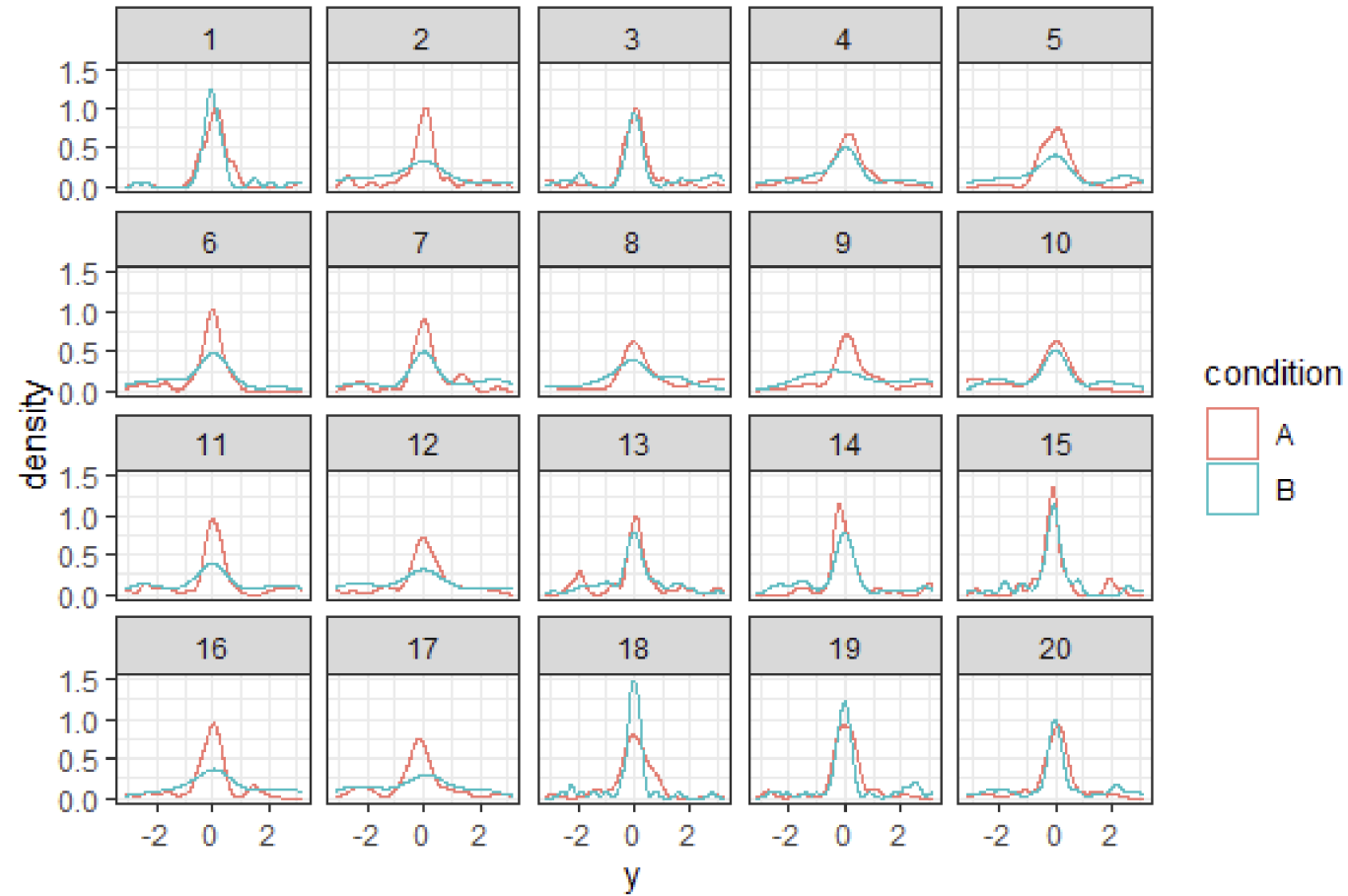
1. P_{mem} = probability of having an item in memory
2. Precision = deviations from correct item



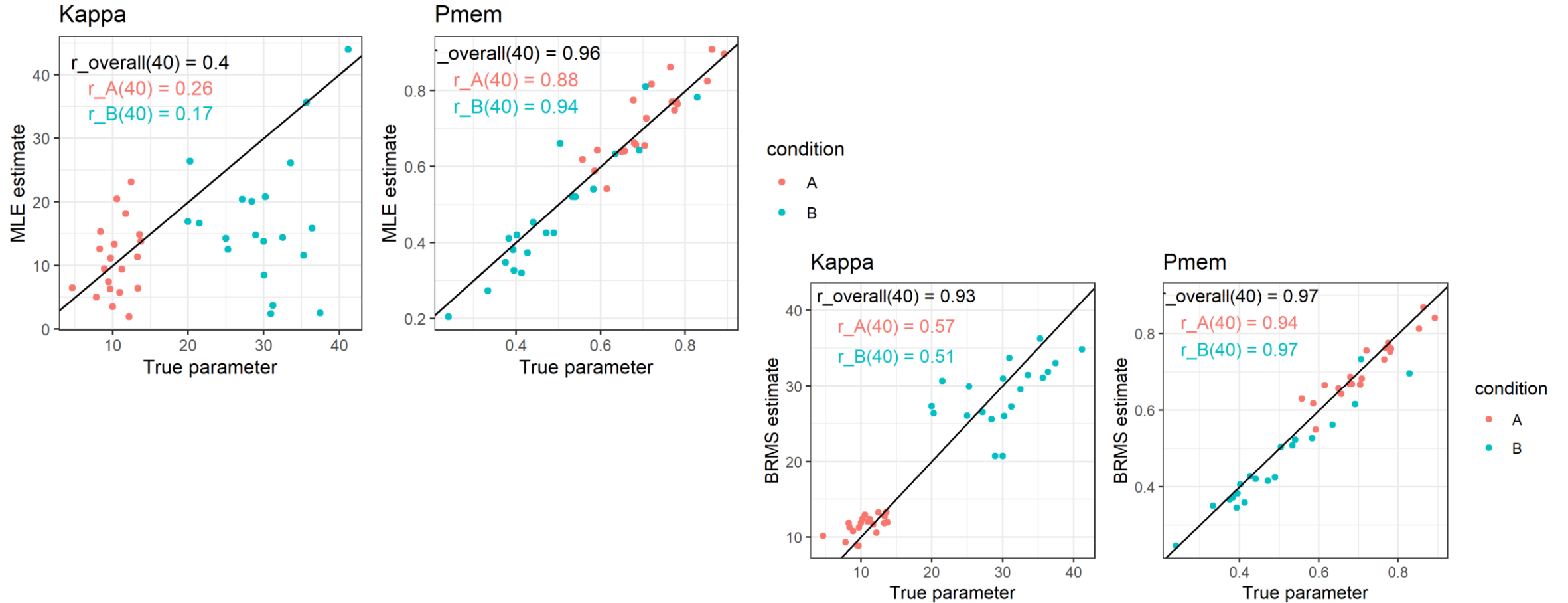
Maximum likelihood vs Hierarchical Estimation

- Maximum likelihood methods fit the models separately to each
- This requires >200 trials per participant for reliable parameter
- Hierarchical methods pool data from all participants, allowing r
- Let's illustrate!
 - We will simulate 20 participants with 50 observation in 2 cc
 - Condition A:
 - $\text{Kappa} \sim N(10, 2.5)$
 - $\text{Pmem} \sim N(0.75, 0.2)$
 - Condition B:
 - $\text{Kappa} \sim N(30, 2.5)$
 - $\text{Pmem} \sim N(0.25, 0.2)$





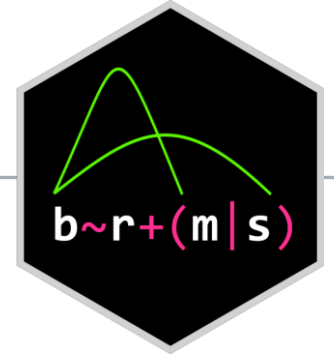
Bayesian hierarchical estimation vs maximum likelihood





brms/bmm in R vs MemToolbox in Matlab

	brms/bmm	MemToolbox
Estimation	Bayesian	Bayesian and Maximum Likelihood
Fitting multiple conditions	Jointly (Linear model syntax)	Separately to each condition
Inference over multiple conditions	1-step procedure	2-step procedure
Allows continuous predictors	Yes	No
Can fix some parameters across conditions	Yes	No
Tasks	Continuous report, custom	Continuous report, Change detection
Included models “out-of-the-box”	2-parameter, 3-parameter, Interference measurement model	2-parameter, 3-parameter, Variable precision, Slot+averaging, Slots+resources
Can customize models	Yes	No



Specifying mixture models in brms

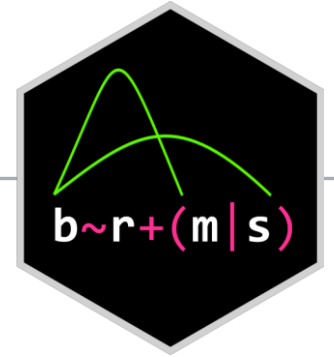
A (short) intro to brms (Bayesian regression models using Stan)

- I. interface to Stan → fit Bayesian generalized linear models
- II. formula syntax similar to lme4 → provides familiar & simple regression analyses.

```
lm(RT ~ X1 + X2)
```

```
lmer(RT ~ X1 + X2 + (X1 + X2 | ID))
```

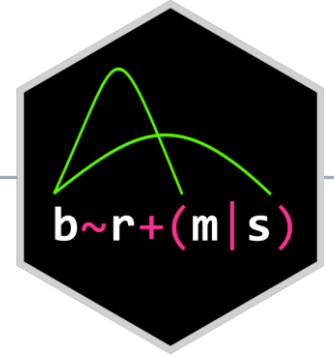
```
brm(RT ~ X1 + X2 + (X1 + X2 | ID))
```



Specifying mixture models in brms

A (short) intro to brms (Bayesian regression models using Stan)

- I. interface to Stan → fit Bayesian generalized linear models
- II. formula syntax similar to lme4 → provides familiar & simple regression analyses.
- III. wide range of response distributions supported → fit wide range of data
- IV. lots of further modeling options:
 - non-linear and smooth terms,
 - auto-correlation structures,
 - censored data,
 - missing value imputation,
 - and quite a few more (like mixture models 😊)
- V. all parameters of response distribution can be predicted (means, standard deviations, etc.)
- VI. flexible prior specifications → encourages users to apply prior distributions that reflect their beliefs.
- VII. Model fit can easily be assessed → posterior predictive checks, cross-validation, and Bayes factors.



Specifying mixture models in brms

A (short) intro to brms (Bayesian regression models using Stan)

`brm` → main function to fit models using brms

required arguments:

1. **formula** → specifies the regression model to estimate
2. **data** → data set that contains all variables (**important:** match variable names to names in formula!)

defaults:

3. **family** = `gaussian()` → which probability distribution does the DV stem from
4. **priors**
5. **sampler settings** (number of warmup & post-warmup sample, number of MCMC chains, etc.)

```
brm(formula = DV ~ 1 + IV + (1 + IV | ID),  
    data = myData,  
    # optional arguments  
    family = gaussian(),  
    prior = myPriors,  
    iter = 4000, warmup = 1500,  
    nChains = 4)
```



Specifying mixture models in brms

Setting up mixture families

brms allows to specify mixtures of any of the supported data distributions ☺

Steps when setting up mixture models in *brms*:

1. Format the data
2. Specify the mixture family
3. Specify model formula to predict parameters
4. Set priors to identify the different mixture components
5. Estimate the model (and have some patience!)
6. Evaluate results

Option

If you want to follow along the next steps directly in R, open the R script „**2pMM_Zhang&Luck2008_brms.R**“.

This script implements all steps that we address now, using only brms syntax. Please note that there are some additional things implemented in the script, that are not included in the here.



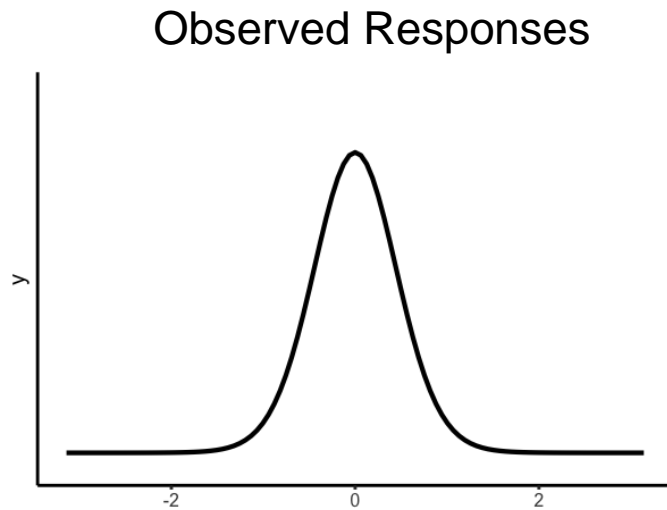
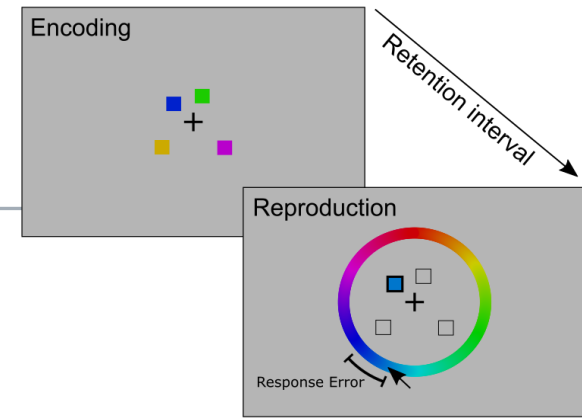
Formatting the data

1. One row for each trial/response
2. Response error = Response – target value → in radians!!!
 - radians = $\pi/180^\circ \times \text{degrees}$ → several packages have implemented this formula

	subID	trial	setsize	RespErr	Pos_Lure1	Pos_Lure2	Pos_Lure3	Pos_Lure4	Pos_Lure5
1	1	1	6	-1.85004900	2.3387410	1.9198620	-2.268928	-1.431170	-3.106686
2	1	2	1	-0.06981317	NA	NA	NA	NA	NA
3	1	3	2	-0.03490659	-1.7104230	NA	NA	NA	NA
4	1	4	2	0.03490659	2.1293020	NA	NA	NA	NA
5	1	5	6	-1.53589000	-0.2792527	-0.6981317	-1.954769	1.815142	-2.373648
6	1	6	3	0.73303830	0.3141593	-3.0368730	NA	NA	NA

Specifying mixture models in brms

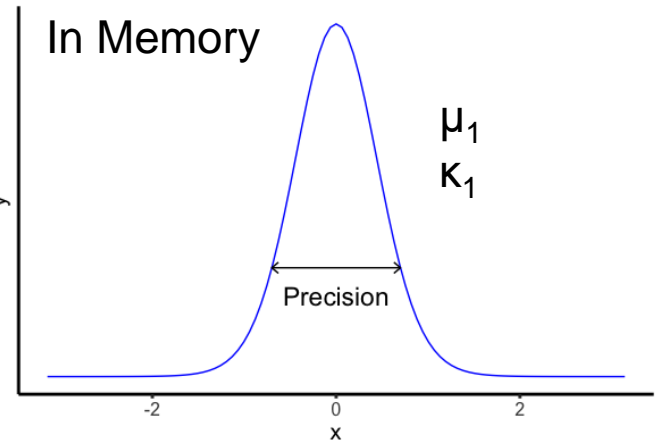
Understanding & Identifying parameters of mixture families



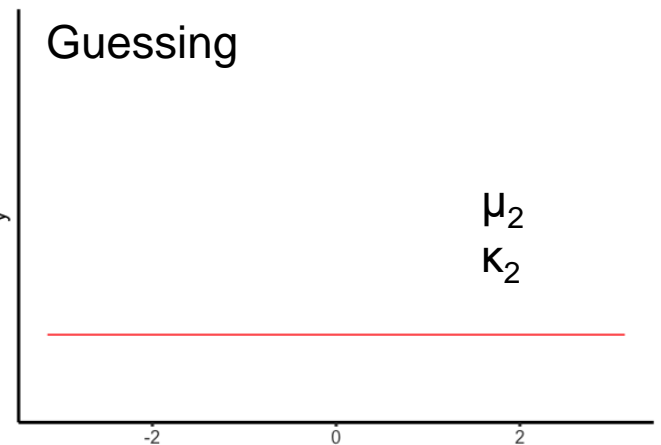
θ_1

θ_2

$$\text{Softmax: } p_i = \frac{e^{\theta_i}}{\sum_{j=1}^K e^{\theta_j}}$$



Guessing





Specifying mixture models in brms

Setting up mixture families

Step 1: Specifying mixture families in brms

→ `mixture()` function allows setting up mixtures of any set of distributions implemented in *brms*

Example: Two-Parameter Mixture model

```
mixFamily_ZL <- mixture(von_mises, von_mises)
```

Parameters for Mixture Families:

- parameters of each mixture distribution :
 - 1) vonMises₁: μ_1 & κ_1
 - 2) vonMises₂: μ_2 & κ_2
- mixing proportions
 - Each mixture distribution gets a mixing proportion → θ
 - mixing proportions are converted to probabilities → Softmax
 - One mixing proportion needs to be fixed for scaling

Specifying mixture models in brms

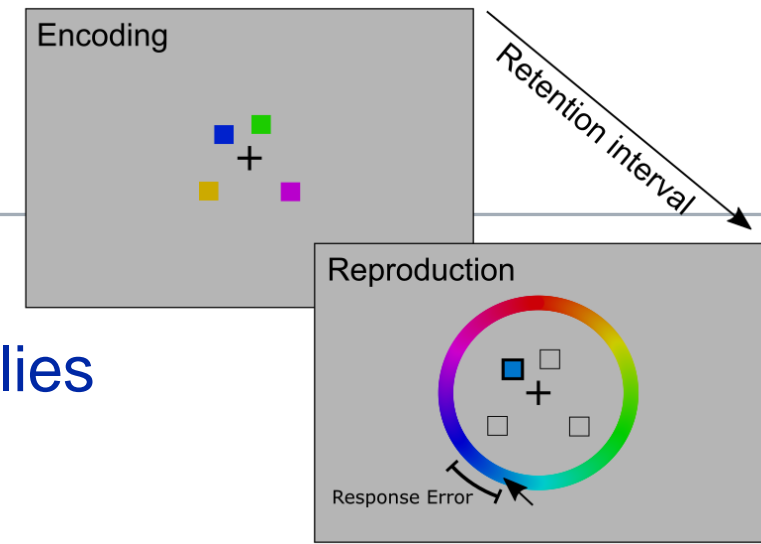
Understanding & Identifying parameters of mixture families

Step 2: Specifying the model formula

```
formula_ZL <- brmsformula(ResponseError ~ 1,  
                             kappa1 ~ 1 + setsize + (1 + setsize | ID),  
                             mu2 ~ 1,  
                             kappa2 ~ 1,  
                             theta1 ~ 1 + setsize + (1 + setsize | ID))
```

Formula elements:

- Declaring the **dependent variable** → Response Error
- Predicting **estimated parameters**
- Setting **intercepts** for constrained parameters → fixed via priors



Specifying mixture models in brms

Understanding & Identifying parameters of mixture families

Step 3: setting priors to identify distributions

```
priors_ZL <-  
  prior(constant(0), class = Intercept, dpar = „mu1“) +  
  prior(constant(0), class = Intercept, dpar = „mu2“) +  
  prior(constant(log(0.0001)), class = Intercept, dpar = „kappa2“)
```

Constraints in the Two-Parameter Mixture Model:

1. memory distribution is centered around zero $\rightarrow \mu_1 = 0$
2. center the guessing distribution around zero $\rightarrow \mu_2 = 0$
3. guessing distribution is flat $\rightarrow \kappa_2 \approx 0$ (brms uses a log-link function)
4. θ_2 is internally fixed to zero by brms

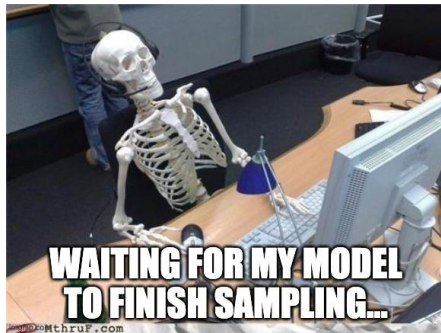
Specifying mixture models in brms

Fitting & Summarizing results of mixture models

Step 4: estimating the with the *brm* function

1. provide the **specified formula**,
2. names of formula variables = **data** variable names
3. Use **mixture family** as data distribution family
4. submit the **defined priors** to identify your mixture distributions

→ **Model estimation takes some time... be patient.**



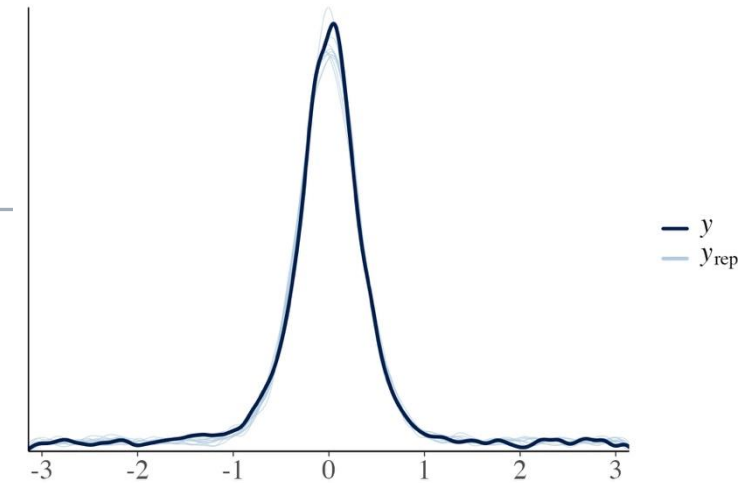
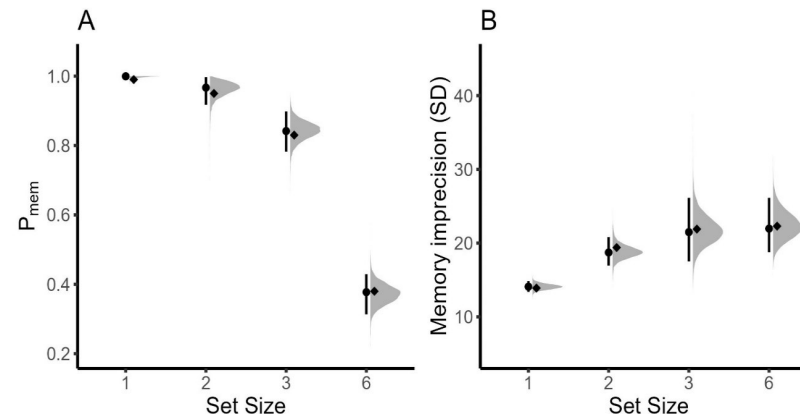
```
fit_ZLmodel <- brm(  
  formula = formula_ZL,  
  data = myData,  
  # now required arguments  
  family = mixFamily_ZL,  
  prior = priors_ZL)
```

Specifying mixture models in brms

Fitting & Summarizing results of mixture models

Step 5: Evaluating model results

1. `pp_check(fit_ZLmodel)` → posterior predictive plot to visually evaluate model fit
2. `summary(fit_ZLmodel)` → overview of the estimated parameters
3. `fixef(fit_ZLmodel)` & `ranef(fit_ZLmodel)` → get fixed and random effect estimates
4. `tidybayes` & `ggplot2` package → processing posterior draws; useful to plot model results



Group-Level Effects:

~subID (Number of levels: 8)

	Estimate	Est.Error	l-95% CI	u-95% CI	Rhat	Bulk_ESS	Tail_ESS
sd(kappa1_setsize1)	0.07	0.06	0.00	0.21	1.00	3844	3490
sd(kappa1_setsize2)	0.21	0.13	0.02	0.51	1.00	2188	2385
sd(kappa1_setsize3)	0.48	0.20	0.20	0.97	1.00	2359	3564
sd(kappa1_setsize6)	0.19	0.16	0.01	0.58	1.00	3848	3370
sd(theta1_setsize1)	1.77	1.54	0.08	5.65	1.00	2326	2845
sd(theta1_setsize2)	1.49	0.70	0.57	3.26	1.00	2908	4425
sd(theta1_setsize3)	0.46	0.27	0.05	1.08	1.00	2633	2557
sd(theta1_setsize6)	0.14	0.12	0.00	0.43	1.00	3549	3506

Population-Level Effects:

	Estimate	Est.Error	l-95% CI	u-95% CI	Rhat	Bulk_ESS	Tail_ESS
mu1_Intercept	0.00	0.00	0.00	0.00	NA	NA	NA
mu2_Intercept	0.00	0.00	0.00	0.00	NA	NA	NA
kappa2_Intercept	-100.00	0.00	-100.00	-100.00	NA	NA	NA
kappa1_setsize1	2.81	0.05	2.70	2.91	1.00	6449	6160
kappa1_setsize2	2.23	0.10	2.01	2.42	1.00	3597	4064
kappa1_setsize3	1.94	0.20	1.55	2.34	1.00	2156	2387
kappa1_setsize6	1.88	0.17	1.55	2.21	1.00	6705	6526
theta1_setsize1	6.46	1.61	4.39	10.78	1.00	2411	1564
theta1_setsize2	3.37	0.68	2.21	4.92	1.00	2532	4005
theta1_setsize3	1.69	0.22	1.26	2.14	1.00	3679	4257
theta1_setsize6	-0.53	0.13	-0.78	-0.28	1.00	5835	4647

Specifying mixture models in brms

Fitting & Summarizing results of mixture models

Step 5: Evaluating model results

Important: take care of link function and transformations when interpreting model parameters

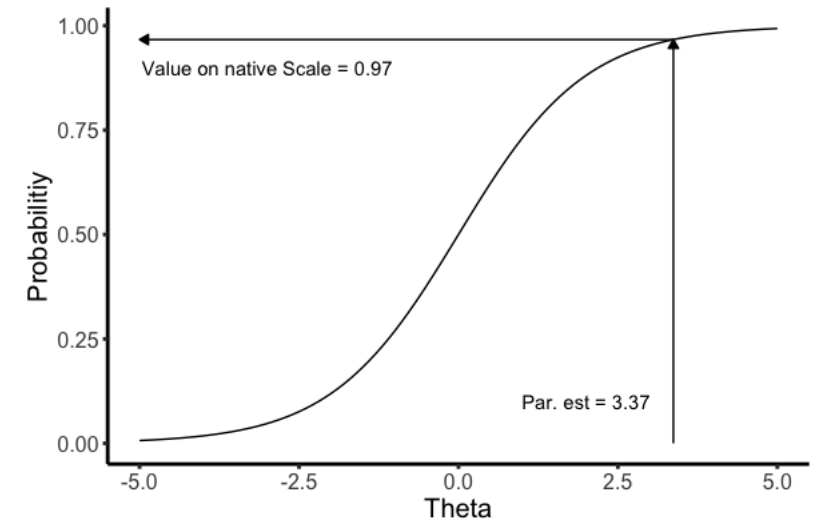
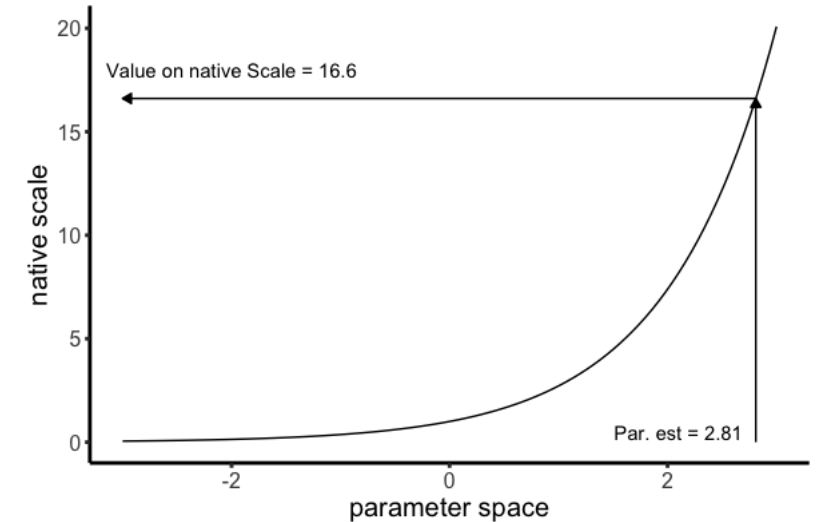
→ for computational efficiency and ideal sampling brms transforms parameters with bounded parameter spaces (e.g., precision > 0, 0 > probabilities < 1)

Kappa → log link

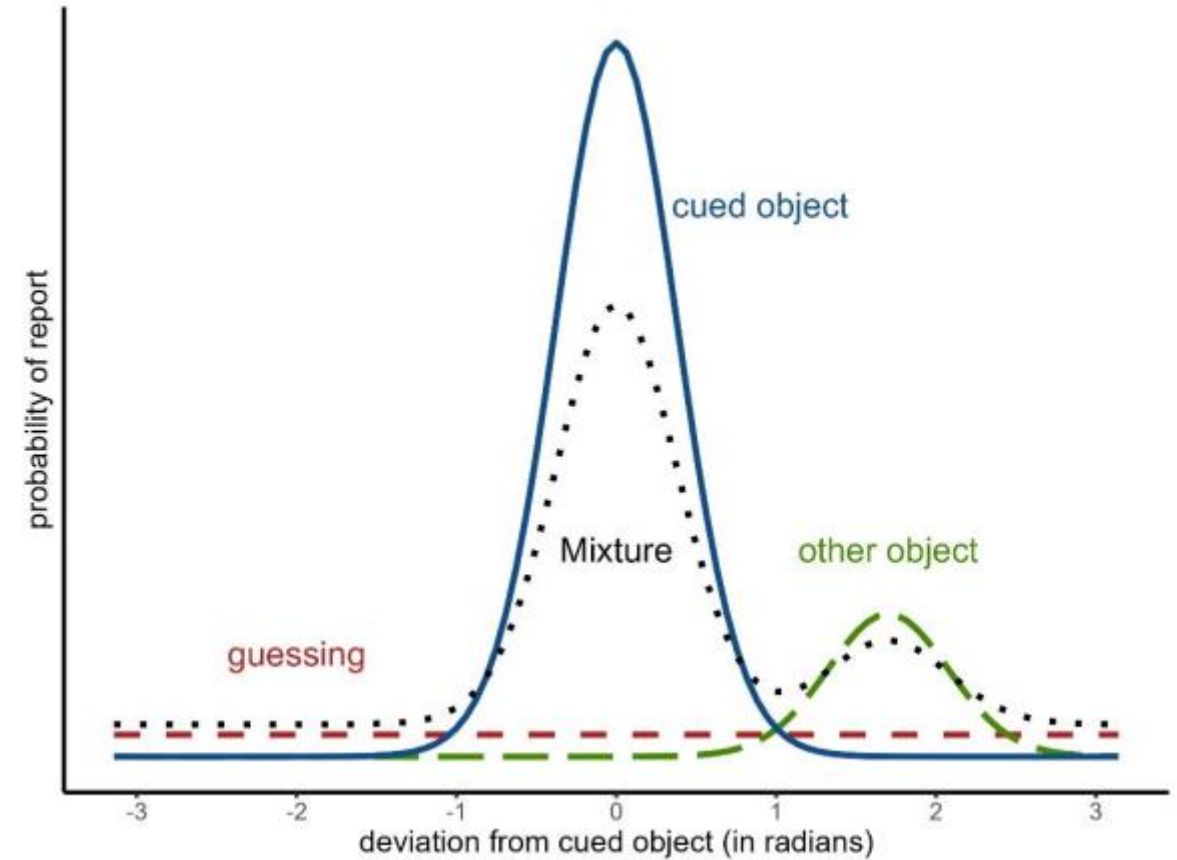
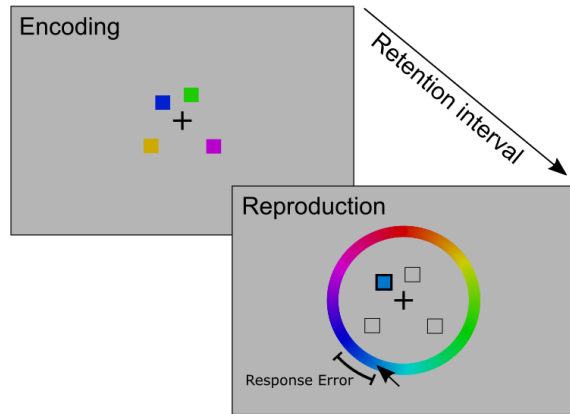
$$\kappa_{native} = e^{\kappa_{par}}; \kappa_{par} = \log(\kappa_{native})$$

Theta → Softmax (for two mixtures = logit)

$$p_i = \frac{e^{\theta_i}}{\sum_{j=1}^K e^{\theta_j}}; p_{Mem} = \frac{e^{\theta_{mem}}}{e^0 + e^{\theta_{mem}}} = \frac{e^{\theta_{mem}}}{1 + e^{\theta_{mem}}}$$



Three parameter mixture model pictures

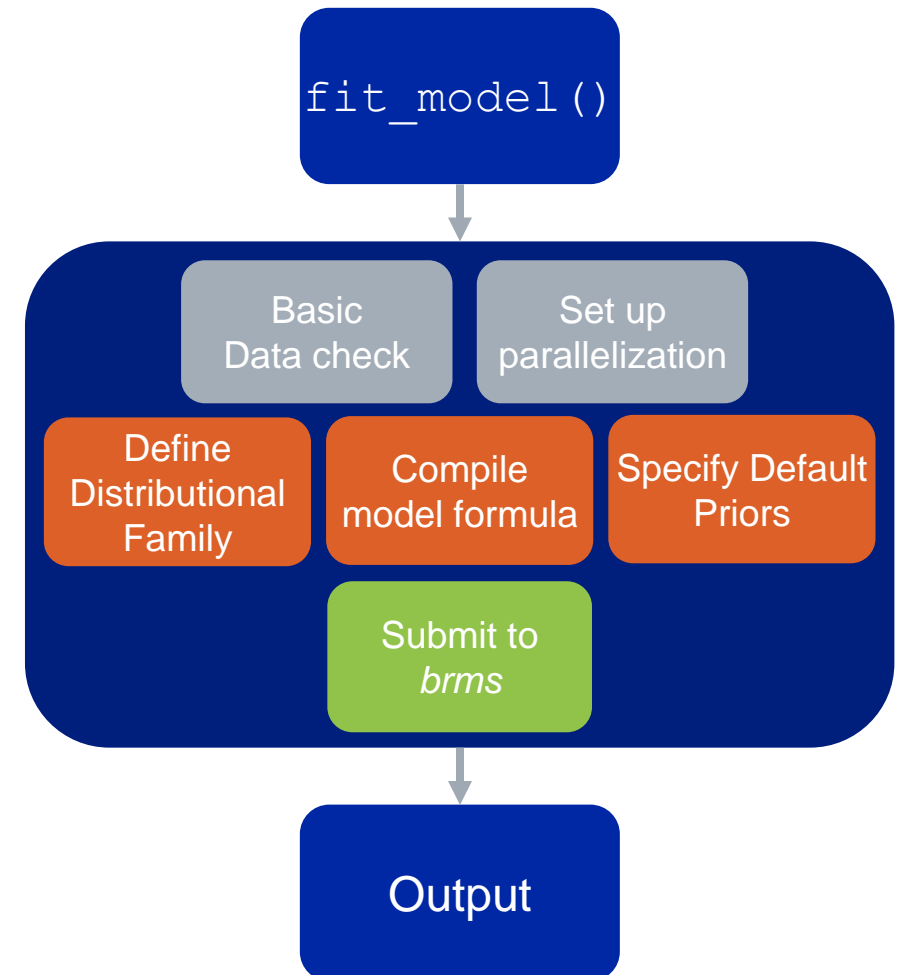
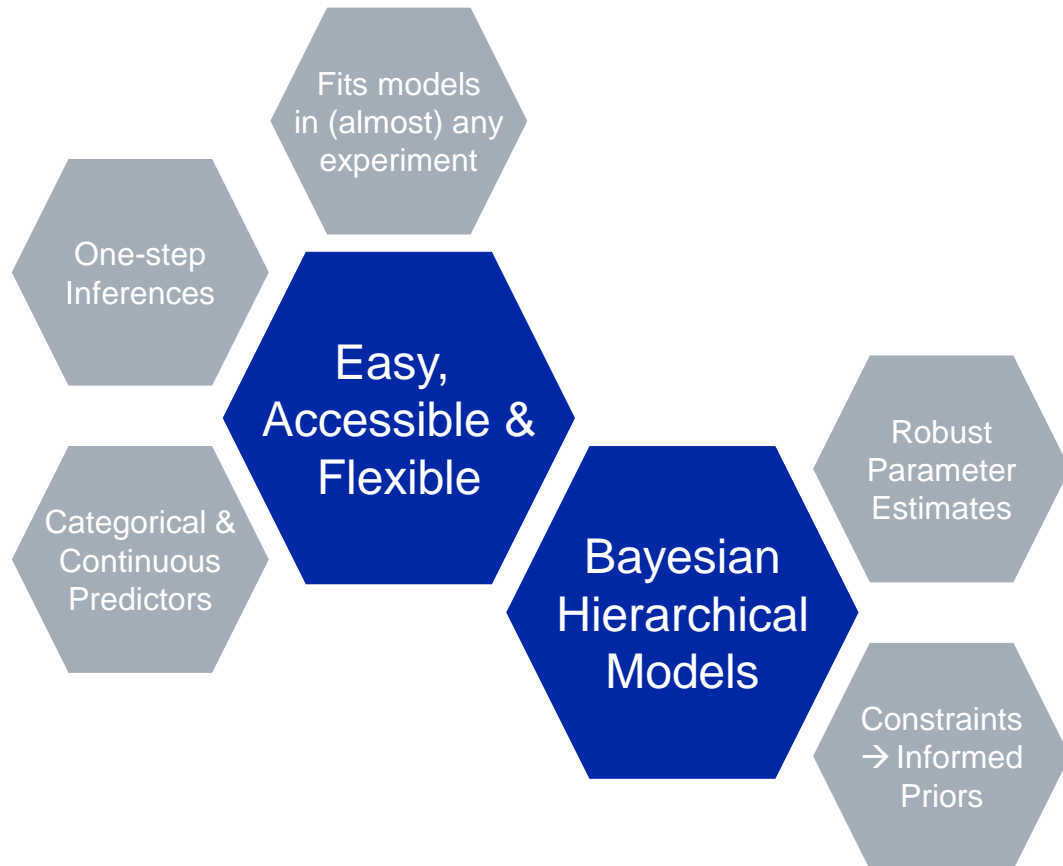


Scary code

```
data_Bays2009 <- read.table(here("data/Bays2009.txt"), header = T) %>%
  dplyr::mutate(
    RespErr = wrap(RespErr),
    # create index variables for whether the lure exist at that setsize
    LureIdx1 = case_when(setsize >= 2 ~ 1, TRUE ~ 0),
    LureIdx2 = case_when(setsize >= 3 ~ 1, TRUE ~ 0),
    LureIdx3 = case_when(setsize >= 4 ~ 1, TRUE ~ 0),
    LureIdx4 = case_when(setsize >= 5 ~ 1, TRUE ~ 0),
    LureIdx5 = case_when(setsize >= 6 ~ 1, TRUE ~ 0),
    # variable to include in formula as a correction to theta due to setsize
    inv_ss = 1/(setsize-1),
    inv_ss = ifelse(is.infinite(inv_ss), 1, inv_ss),
    setsize = as.factor(setsize))
```

```
# set up mixture model
Bays_mixModel_formula <- bf(RespErr ~ 1,
  # fix kappa over memory distributions
  nlf(kappa1 ~ kappa),      # target distribution
  nlf(kappa2 ~ kappa),      # non-target
  nlf(kappa3 ~ kappa),      # non-target
  nlf(kappa4 ~ kappa),      # non-target
  nlf(kappa5 ~ kappa),      # non-target
  nlf(kappa6 ~ kappa),      # non-target
  # kappa for guessing distribution will be fixed using priors
  kappa7 ~ 1,               # uniform
  # specify mixing distributions for distinct item categories
  nlf(theta1 ~ thetat),     # p_mem
  nlf(theta2 ~ LureIdx1*(thetant + log(inv_ss)) + (1-LureIdx1)*(-100)), # p_intrusion
  nlf(theta3 ~ LureIdx2*(thetant + log(inv_ss)) + (1-LureIdx2)*(-100)), # p_intrusion
  nlf(theta4 ~ LureIdx3*(thetant + log(inv_ss)) + (1-LureIdx3)*(-100)), # p_intrusion
  nlf(theta5 ~ LureIdx4*(thetant + log(inv_ss)) + (1-LureIdx4)*(-100)), # p_intrusion
  nlf(theta6 ~ LureIdx5*(thetant + log(inv_ss)) + (1-LureIdx5)*(-100)), # p_intrusion
  # target & guessing distribution will be centered using priors
  mu1 ~ 1, # fixed intercept constrained using priors
  mu7 ~ 1, # fixed intercept constrained using priors
  # center non-target distribution on data specified locations
  nlf(mu2 ~ Pos_Lure1),     # center non-target
  nlf(mu3 ~ Pos_Lure2),     # center non-target
  nlf(mu4 ~ Pos_Lure3),     # center non-target
  nlf(mu5 ~ Pos_Lure4),     # center non-target
  nlf(mu6 ~ Pos_Lure5),     # center non-target
  # now predict parameters of interest
  kappa ~ 0 + setsize,      # fixed intercept for precision of memory distributions
  thetat ~ 0 + setsize,     # fixed intercept for p_mem
  thetant ~ 0 + setsize,    # fixed intercept for p_intrusion
  # thetag ~ 1,             # fixed intercept for p_guess
  # for brms to process this formula correctly, set non-linear to TRUE
  nl = TRUE)
```

bmm: Easy implementations of mixture models for VWM tasks



bmm: Easy implementations of mixture models for VWM tasks

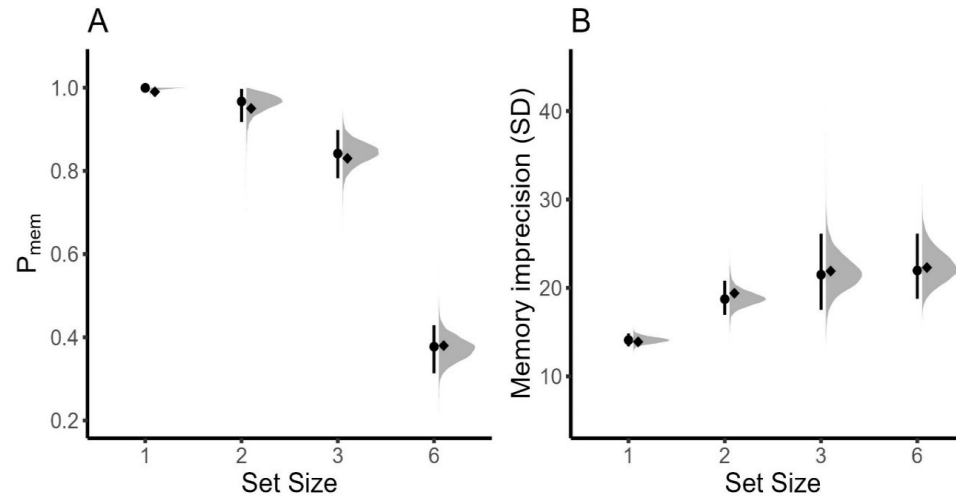
Two-Parameter Mixture Model

1) Specify
formula

Predictors
of Parameters

```
ZL_mixFormula_bmm <- bf(RespErr ~ 1,  
  kappa ~ 0 + setsize + (0 + setsize || subID),  
  thetat ~ 0 + setsize + (0 + setsize || subID))
```

Dependent
Variable



```
fit_ZL2009_bmm <- bmm::fit_model(formula = ZL_mixFormula,  
  data = data_ZL2008,  
  model type = '2p',  
  warmup=1000, iter=2000, parallel=TRUE)
```

Formula &
Data

2) Fit Model

Select
Model Type

Additional
Arguments

bmm: Easy implementations of mixture models for VWM tasks

Three-Parameter Mixture Model

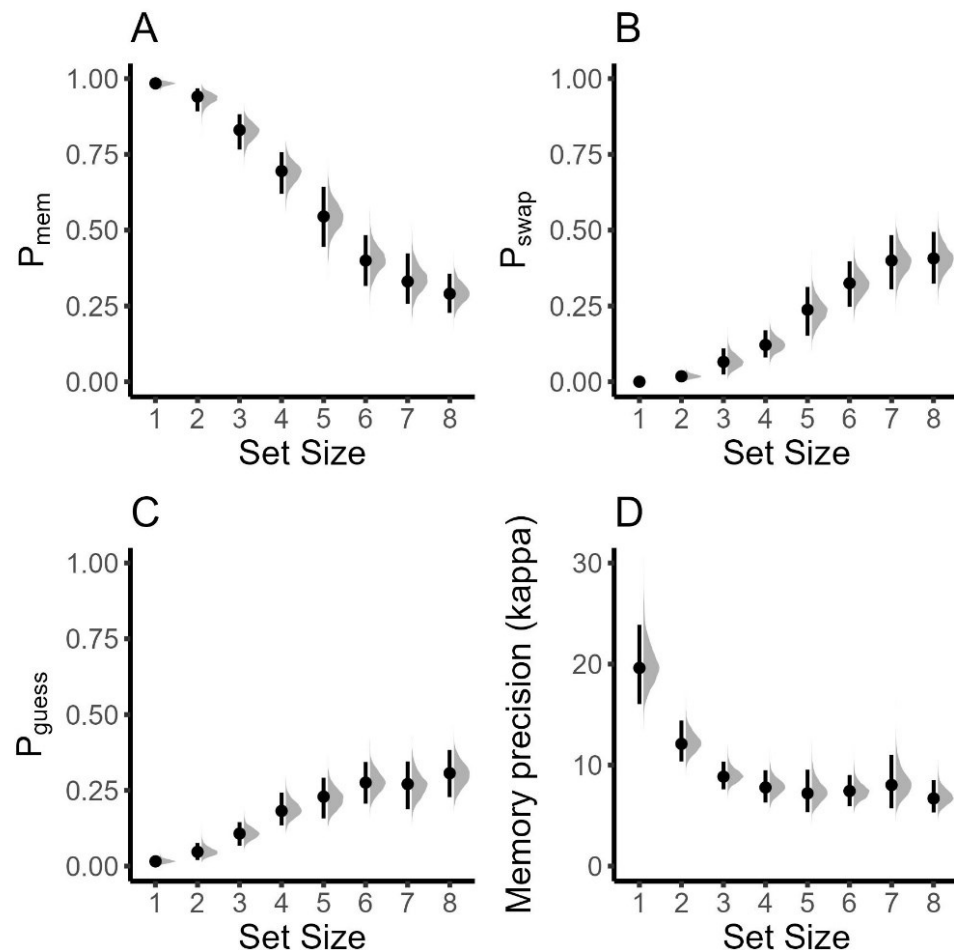
1) Specify
formula

```
ff <- bf(devRad ~ 1,  
  kappa ~ 0 + SetSize + (0 + SetSize || ID),  
  thetat ~ 0 + SetSize + (0 + SetSize || ID),  
  thetant ~ 0 + SetSize + (0 + SetSize || ID))
```

2) Fit Model

Additional
Arguments

```
fit_3pMM <- bmm::fit_model(  
  formula = ff,  
  data = df_OberauerLin2017_E1,  
  model type = '3p',  
  non_targets = paste0('Item',2:8,'_Col_rad'),  
  setsize = "SetSize")
```



The Bayesian Measurement Model (*bmm*) packages

Interference Measurement Model

1) Specify
formula

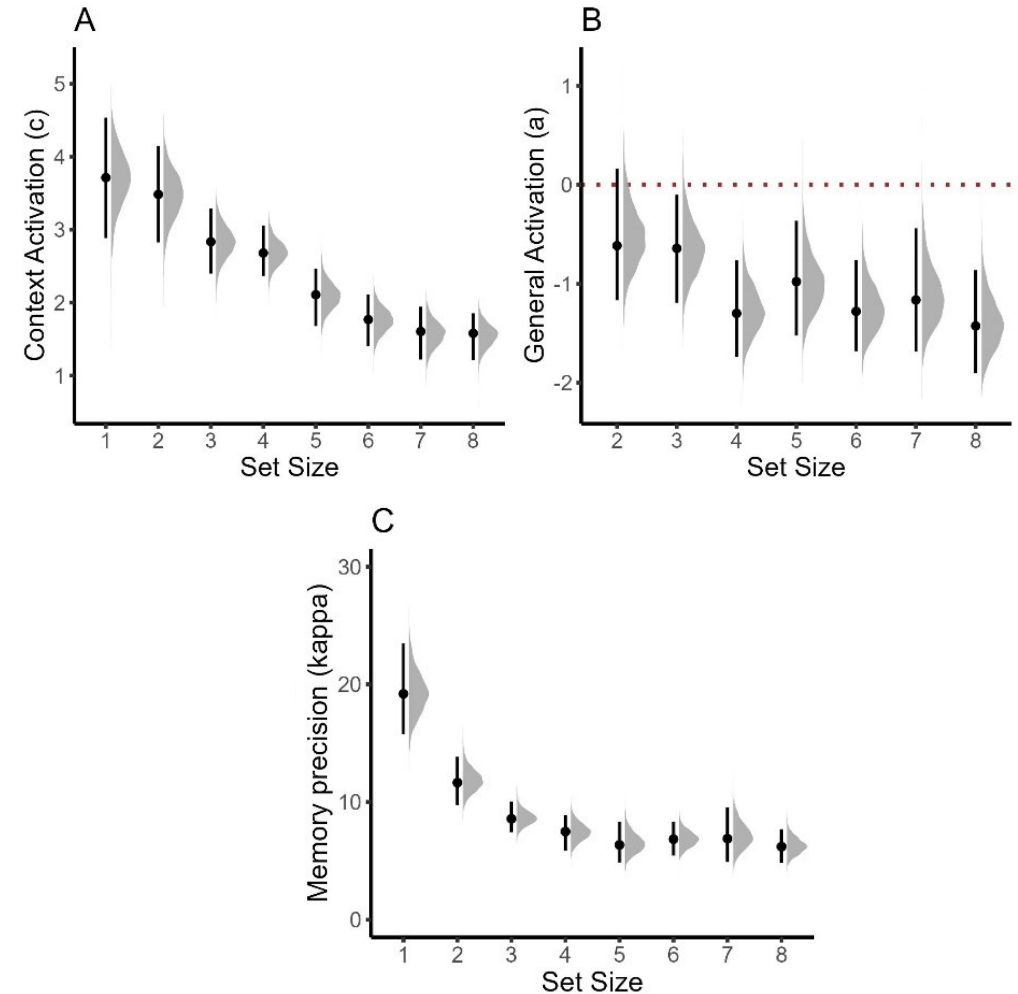
```
ff <- bf(devRad ~ 1,
```

```
  kappa ~ 0 + SetSize + (0 + SetSize || ID),  
  c ~ 0 + SetSize + (0 + SetSize || ID),  
  a ~ 0 + SetSize + (0 + SetSize || ID),  
  s ~ 0 + SetSize + (0 + SetSize || ID),
```

2) Fit Model

Additional
Arguments

```
fit_IMMfull_mixMod <- fit_model(  
  formula = ff,  
  data = df_OberauerLin2017_E1,  
  model type = 'IMMfull',  
  non_targets = paste0('Item',2:8,'_Col_rad'),  
  spaPos = paste0('Item',2:8,'_Pos_rad'),  
  setsize = "SetSize")
```





Testing Hypothesis with Bayesian models

1. Evaluating 95% Highest Density Intervals

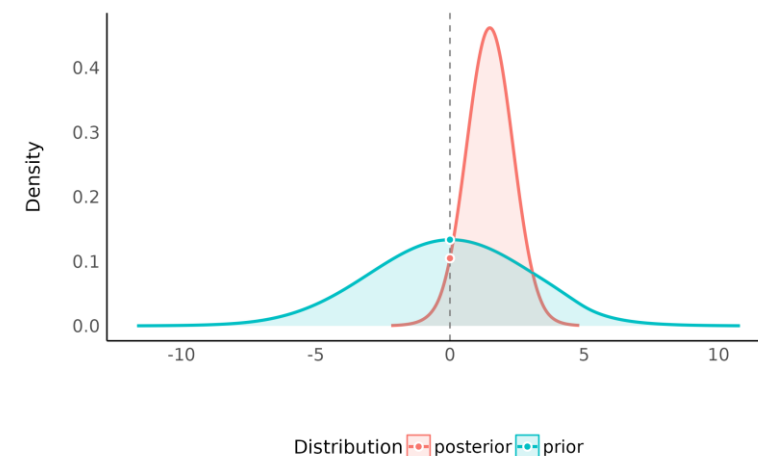
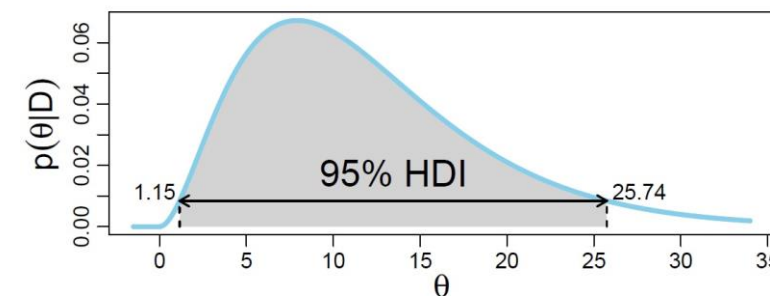
→ Do my posterior estimates, include a specific value (e.g., 0) in their posterior distribution

2. Computing Bayes Factors for parameters (Savage-Dickey Ratio)

→ Given the data, has my prior belief credibly changed

3. Computing Bayes Factors for competing models (bridgesampling)

→ Under which model are the observed data more probable?





Time for a coffee break!

Preprint introducing the *bmm* package:

Frischkorn, G. T., & Popov, V. (2023). *A tutorial for estimating mixture models for visual working memory tasks in brms: Introducing the Bayesian Measurement Modeling (bmm) package for R*. PsyArXiv.

<https://doi.org/10.31234/osf.io/umt57>





Work with (your own) data

You have your own data

1. Prepare your data
 - a) Transform to long format → each trial in a row
 - b) Calculate response error in radians
 - c) Calculate non-target locations relative to the target values
2. Specify a model you want to fit
3. Test if the modelling is sampling
 - Use a low number of samples to avoid lengthy wait times (iter = 500)
4. See if you can extract and plot results

You have no own data

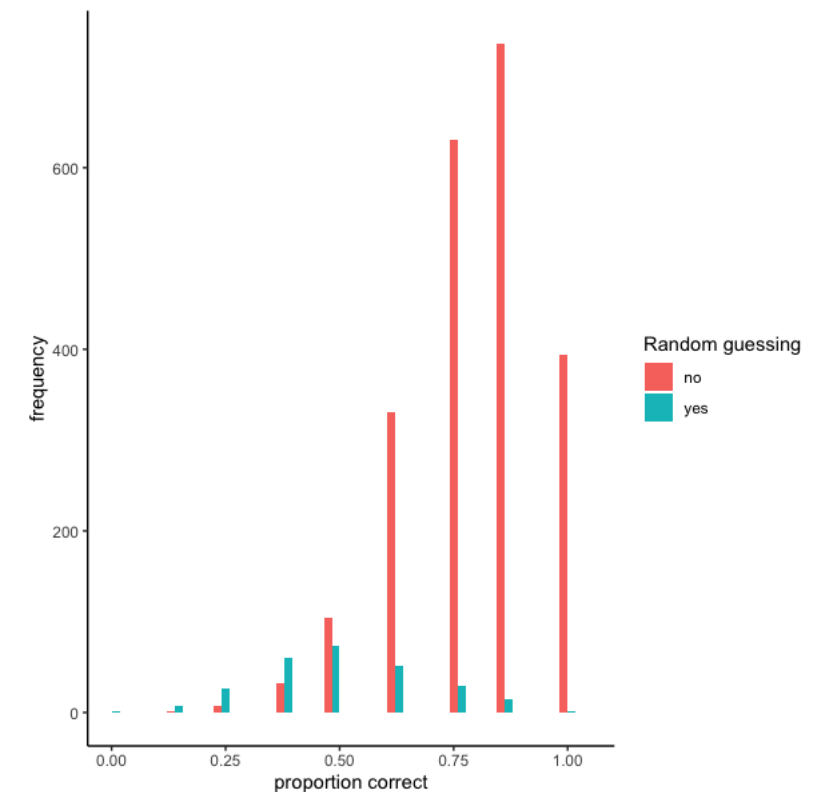
1. Choose one of the data sets shared in the GitHub repository (For continuous reproduction tasks, simulated binomial data, etc.)
2. Try to understand the different variables
3. Specify a model you want to fit
4. Test if the model is sampling
 - Use a low number of samples to avoid lengthy wait times (iter = 500)
5. Extract and plot results

Specifying custom mixture models for accuracy data

A logic similar to VWM mixture models can be applied to accuracy data

- Lapses of attention → guessing performance for some trials
- perform with a certain level of ability (i.e., proportion correct)

```
mix_binomial <- mixture(binomial, binomial)
# fix probability to 0.50 via priors
priors_binomial <- prior(
  constant(0), class = Intercept, dpar = "mu2"
)
```



Specifying custom mixture models for reaction time data

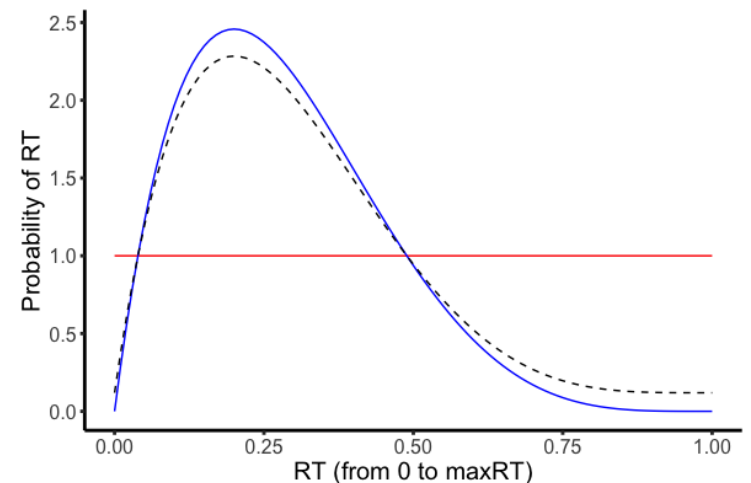
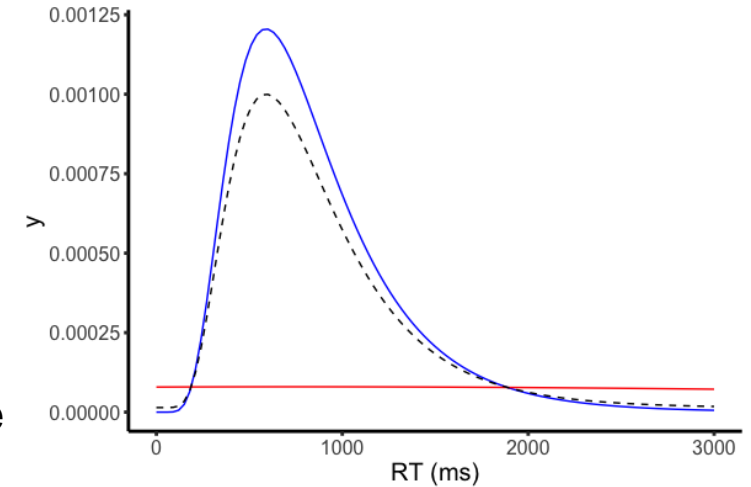
For reaction time data, mixture models can control for very fast or slow contaminants

- Contaminants → uniform distribution
- Other reaction time data → Ex-Gaussian or Shifted Lognormal

Problem: brms has not implemented a uniform distribution → lacks free parameters to be predicted

Work-Arounds:

- Gaussian with very large SD as contaminant distribution
- Re-scale to range of 0 to 1 (= maxRT) and fit a mixture of beta distributions → parameter interpretation is challenging





Thank you for your attention!

Please approach us, if you have ideas what kinds of models to add to the bmm package.

We are also looking for collaborators to work with us on the package 😊