LOO-PSIS: Model Comparison with Cross-Validation

Bayesian Mixed Effects Models with brms for Linguists

Job Schepens

2025-12-17

Table of contents

1 LOO-PSIS: Leave-One-Out Cross-Validation

LOO-PSIS (Leave-One-Out Cross-Validation with Pareto-Smoothed Importance Sampling) helps answer: **Which model predicts new data better?**

1.1 Why Use LOO Instead of Prior Comparison?

These approaches answer different questions:

Prior comparison (what we did earlier):

- Shows if posterior coefficient estimates and effect sizes are sensitive to prior choice
- Good for: reporting robustness of conclusions
- Question: "Do my results depend on my priors?"

LOO comparison (this approach):

- Shows which model predicts better
- Good for: feature selection, model building
- Question: "Which model structure produces better predictions?"
- Can compare:
  - Different priors (e.g., narrow/regularizing vs. wide priors)
  - Different likelihoods (e.g., normal vs. lognormal)
  - Different model structures (e.g., with/without random slopes)

You can do both:

1. First: Compare different priors within same model structure (sensitivity analysis)
2. Then: Use LOO to compare different model structures with best priors (model selection)

1.2 Why Use LOO Instead of Bayes Factors?

LOO advantages:

- Priors less important because we evaluate predictive performance on new data
- Number of samples less important - most uncertainty comes from the data itself
- More stable and interpretable

Bayes factors:

- Very sensitive to prior choice

- Sensitive to number of samples
- Harder to interpret (what does BF = 3.2 mean?)

1.3 Setup

1.4 Create Four Test Datasets

We'll create four datasets to test how LOO behaves under different conditions:

Table 1: Four Test Datasets: 2×2 Design (Sample Size × Data Structure)

Scenario	N	Mean log-RT	SD log-RT	True Data Structure
n=100, WITH RE	100	5.91	0.478	Random slopes + intercepts
n=100, WITHOUT RE	100	6.08	0.366	Fixed effect only
n=40, WITH RE	40	6.37	0.377	Random slopes + intercepts
n=40, WITHOUT RE	40	5.98	0.296	Fixed effect only

1.5 Fit Models for All Four Scenarios

For each dataset, we'll fit two models:

1. **Simple model**: No random effects (just fixed effects) - `log_rt ~ condition`
2. **Complex model**: Random slopes for subjects - `(1 + condition | subject) + (1 | item)`

2 Comparing Models with LOO Across Four Scenarios

2.1 Add LOO Criterion to All Models

```
[1] "Example: Individual LOO output for simple model (n=100, WITH RE)"
```

```

Computed from 4000 by 100 log-likelihood matrix.

         Estimate   SE
elpd_loo    -68.8  7.3
p_loo         2.9  0.5
looic       137.7 14.5
------
MCSE of elpd_loo is 0.0.
MCSE and ESS estimates assume MCMC draws (r_eff in [0.7, 1.0]).

All Pareto k estimates are good (k < 0.7).
See help('pareto-k-diagnostic') for details.
```

```
[1] "Example: Individual LOO output for medium model (n=100, WITH RE)"
```

```

Computed from 4000 by 100 log-likelihood matrix.

         Estimate   SE
```

```
elpd_loo     31.3  7.8
p_loo        15.1  2.2
looic       -62.7 15.5
------
MCSE of elpd_loo is 0.1.
MCSE and ESS estimates assume MCMC draws (r_eff in [0.5, 1.2]).

All Pareto k estimates are good (k < 0.7).
See help('pareto-k-diagnostic') for details.
```

```
[1] "Example: Individual LOO output for complex model (n=100, WITH RE)"
```

```

Computed from 4000 by 100 log-likelihood matrix.

         Estimate   SE
elpd_loo     48.4  7.0
p_loo        21.6  2.6
looic       -96.8 14.0
------
MCSE of elpd_loo is NA.
MCSE and ESS estimates assume MCMC draws (r_eff in [0.6, 1.4]).

Pareto k diagnostic values:
                         Count Pct.    Min. ESS
(-Inf, 0.7]   (good)     99    99.0%   573
   (0.7, 1]   (bad)       1     1.0%   <NA>
   (1, Inf)   (very bad)  0     0.0%   <NA>
See help('pareto-k-diagnostic') for details.
```

Understanding individual LOO output:

- **Estimate**: The ELPD (higher = better predictive accuracy)
- **SE**: Standard error of the estimate (uncertainty)
- **p_loo**: Effective number of parameters (how much the model "uses" the data)
- **looic**: -2 × elpd_loo (lower = better, analogous to AIC)
- **Pareto k diagnostic**: Checks reliability of the LOO approximation
  - All $k < 0.5$: Excellent
  - $k < 0.7$: Good
  - $k > 0.7$: Problematic (may need `reloo = TRUE`)

2.2 Compare Models for Each Scenario

```
[1] "\nFull comparison with p_loo values:"
```

```
                     elpd_diff se_diff elpd_loo se_elpd_loo p_loo  se_p_loo
fit_complex_100_with    0.0       0.0    48.4      7.0        21.6    2.6
fit_simple_100_with  -117.2       9.2   -68.8      7.3         2.9    0.5
                     looic  se_looic
fit_complex_100_with  -96.8   14.0
```

```
fit_simple_100_with   137.7   14.5
```

2.2.1 Understanding the Output Columns

The `loo_compare()` output shows (note: by default only some columns are printed):

Default output:

- **elpd_diff**: Difference from best model (0 for winner, negative for others)
- **se_diff**: Standard error of the difference (uncertainty in comparison)

Full output (with `simplify = FALSE`):

- **elpd_loo**: Expected log pointwise predictive density (higher = better predictions)
- **se_elpd_loo**: Standard error of elpd_loo
- **p_loo**: Effective number of parameters - this shows model complexity!
  - Simple model: p_loo   number of fixed effects + 1 (for sigma)
  - Complex model: p_loo increases with random effects (subjects, items, correlations)
  - **Key**: Higher p_loo = more complex model, but also better fit if it wins
- **looic**: LOO Information Criterion = -2 × elpd_loo (lower = better, like AIC/BIC)

Why p_loo matters: It shows you're not just comparing predictive accuracy, but **accuracy adjusted for complexity**. The complex model has higher p_loo (uses more parameters), so it needs to predict *substantially* better to win.

Key insight: The ratio (|elpd_diff| / se_diff) tells you how many standard errors separate the models. A ratio > 4 indicates strong evidence for the winning model.

2.2.2 ELPD

ELPD = "Expected Log Pointwise Predictive Density"

- **Expected**: We marginalize over all possible future data
- **Log**: Works on log scale for numerical stability
- **Pointwise**: Evaluated separately for each data point
- **Predictive Density**: How well the model predicts new data

Key properties:

- **Higher is better** (like R^2 in frequentist stats)
- **Difference matters**: Which model predicts new data better?
- **Not about fit to current data**: About generalization
- Takes into account the uncertainty of predictions

2.2.3 Ratio

The ratio (|ELPD_diff| / SE) tells us how many standard errors separate the models. Here's a detailed breakdown:

Table 2: Detailed Model Comparison with Ratios (|ELPD_diff| / SE)

Scenario	Model	ELPD	SE	ELPD Δ	SE Δ	Ratio (SE)	Interpretation
n=100, WITH RE	Complex	48.4	7.0	—	—	—	Best model (reference)
n=100, WITH RE	Simple	-68.8	7.3	-117.21	9.25	12.68	Very strong evidence
n=100, WITHOUT RE	Simple	-40.2	7.0	—	—	—	Best model (reference)
n=100, WITHOUT RE	Complex	-42.4	7.2	-2.15	1.35	1.59	Weak evidence
n=40, WITH RE	Complex	5.9	4.5	—	—	—	Best model (reference)
n=40, WITH RE	Simple	-20.1	4.2	-26.06	4.44	5.87	Strong evidence
n=40, WITHOUT RE	Simple	-10.4	3.9	—	—	—	Best model (reference)
n=40, WITHOUT RE	Complex	-12.1	4.3	-1.62	2.02	0.8	Essentially equivalent

2.3 Rule of Thumb for Model Comparison

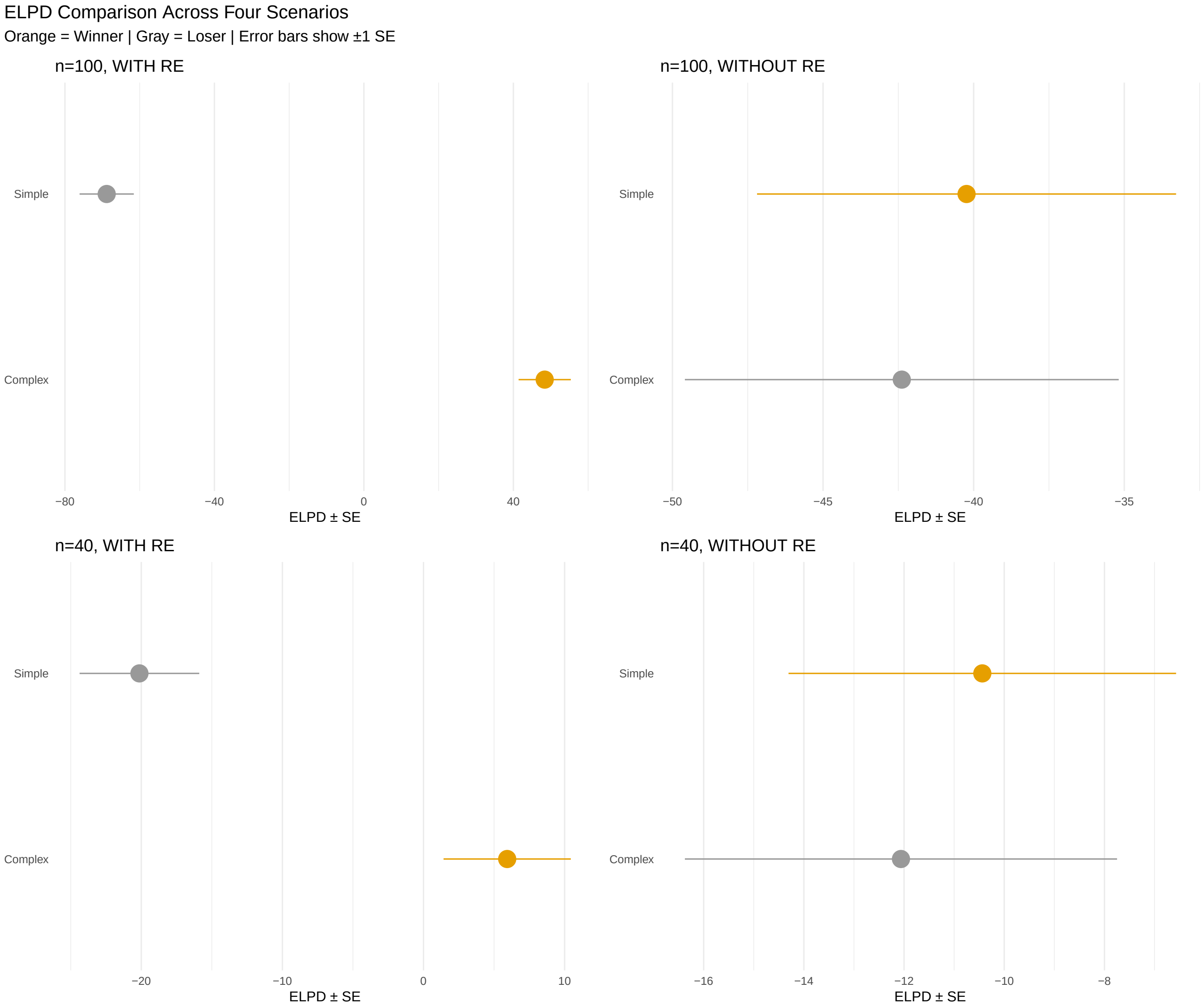
Interpreting `elpd_diff` (expected log pointwise predictive density difference):

elpd_diff	Ratio*	Interpretation	Action
< 4	< 2	Equivalent models	Pick simpler one
4-10	2-4	Moderate difference	Consider larger elpd
> 10	> 4	Clear winner	Prefer larger elpd

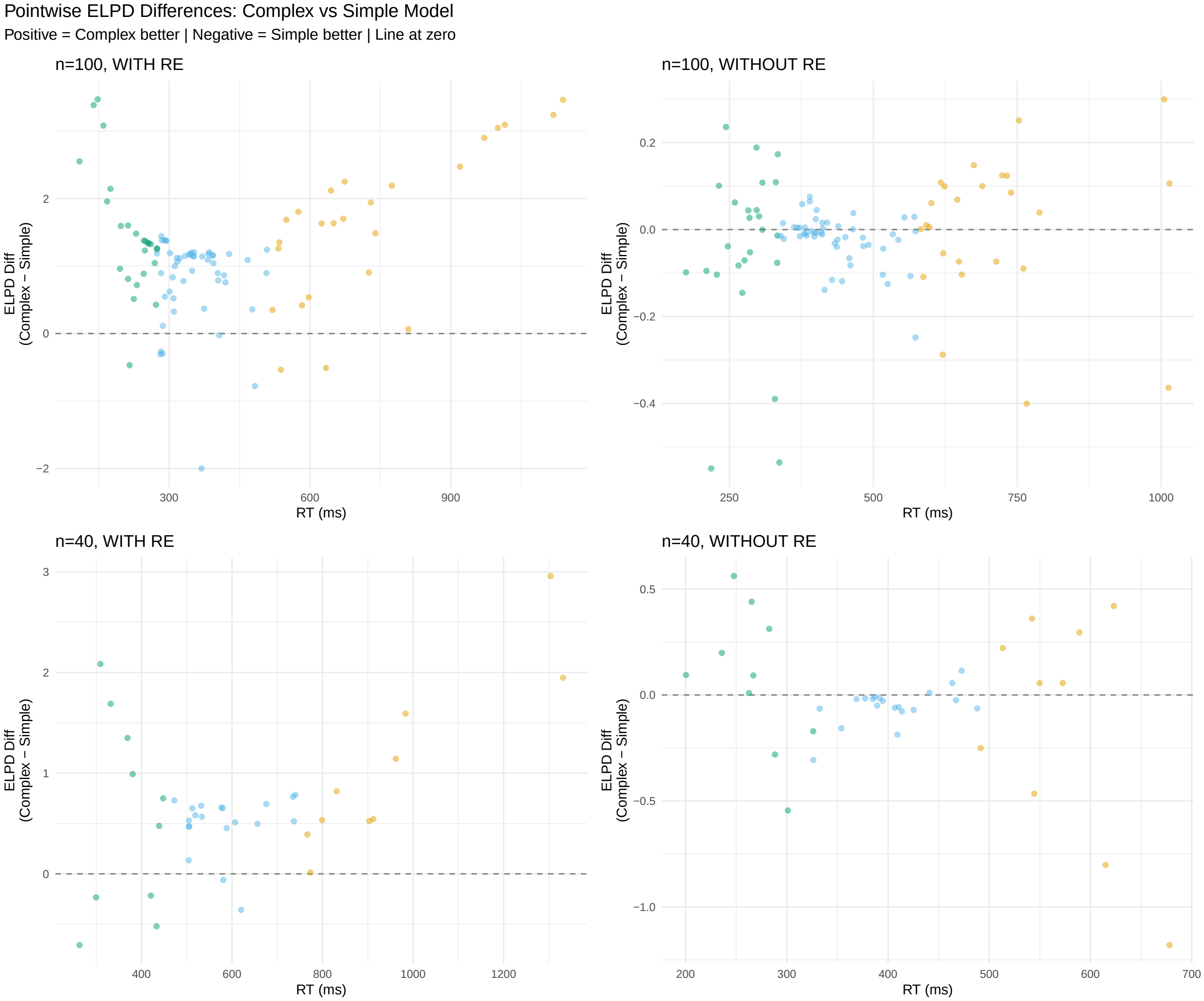
*Ratio = |elpd_diff| / se_diff (how many standard errors apart?)

2.4 Visualizations: Side-by-Side Comparisons

2.4.1 Plot 1: ELPD Comparison (2×2 Grid)



2.4.2 Plot 2: Pointwise ELPD Differences by RT (2×2 Grid)



What to look for:

- **WITH RE scenarios**: Positive values (complex better) when data truly has random slopes
- **WITHOUT RE scenarios**: Near zero or negative values (simple better or equivalent)
- **Sample size effect**: More scatter with n=40, clearer pattern with n=100

2.4.3 Plot 3: Model Weights (2×2 Grid)

Model weights represent the probability that each model would make the best predictions for new data, based on the LOO estimates. Weights close to 1.0 indicate strong confidence in that model, while weights near 0.5 suggest the models are roughly equivalent in predictive performance.

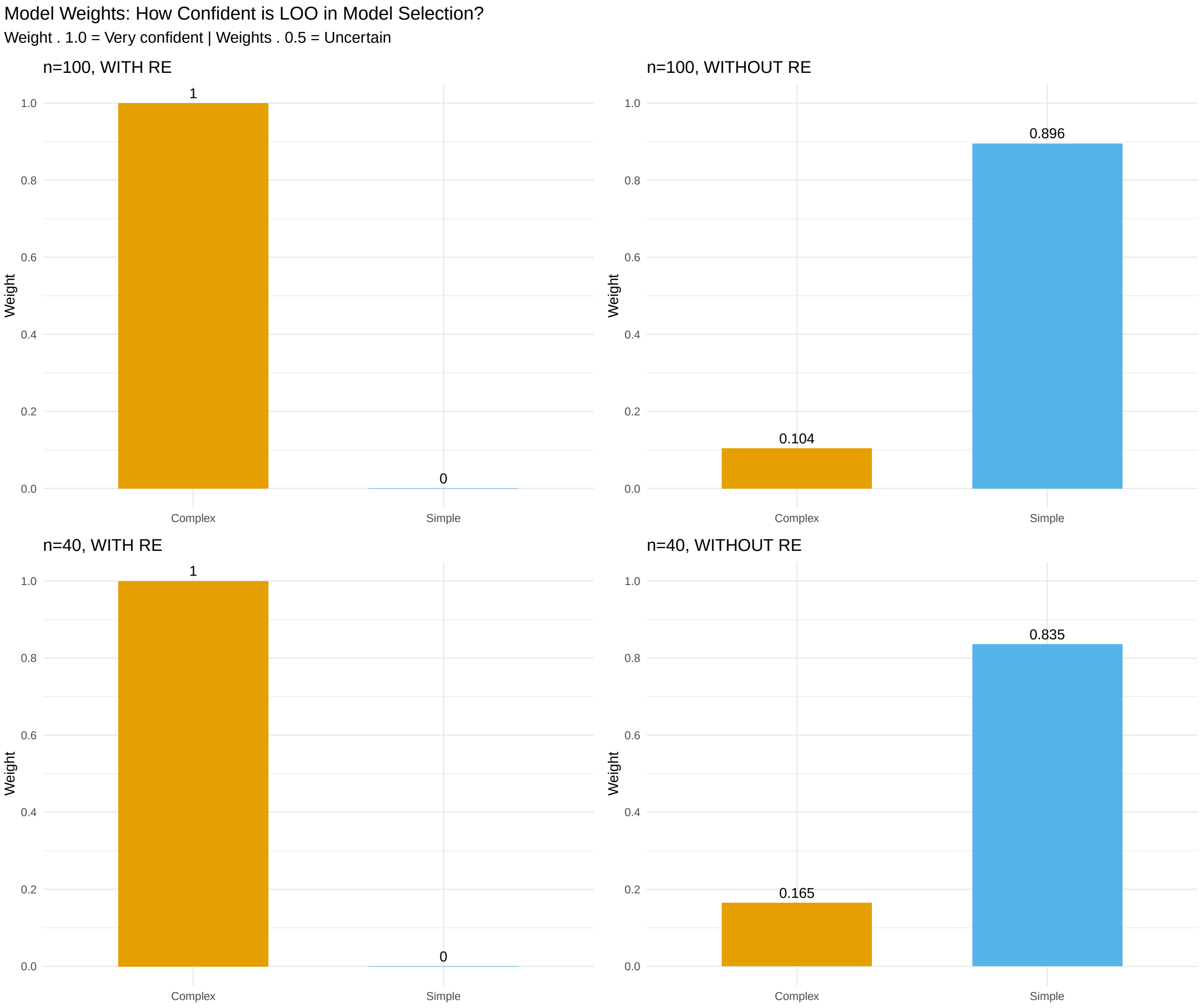


Table 4: Model weights (stacking) across all scenarios

Scenario	Model	Weight
n=100, WITH RE	Simple	0.000
n=100, WITH RE	Complex	1.000
n=100, WITHOUT RE	Simple	0.896
n=100, WITHOUT RE	Complex	0.104
n=40, WITH RE	Simple	0.000
n=40, WITH RE	Complex	1.000
n=40, WITHOUT RE	Simple	0.835
n=40, WITHOUT RE	Complex	0.165

Interpreting model weights:

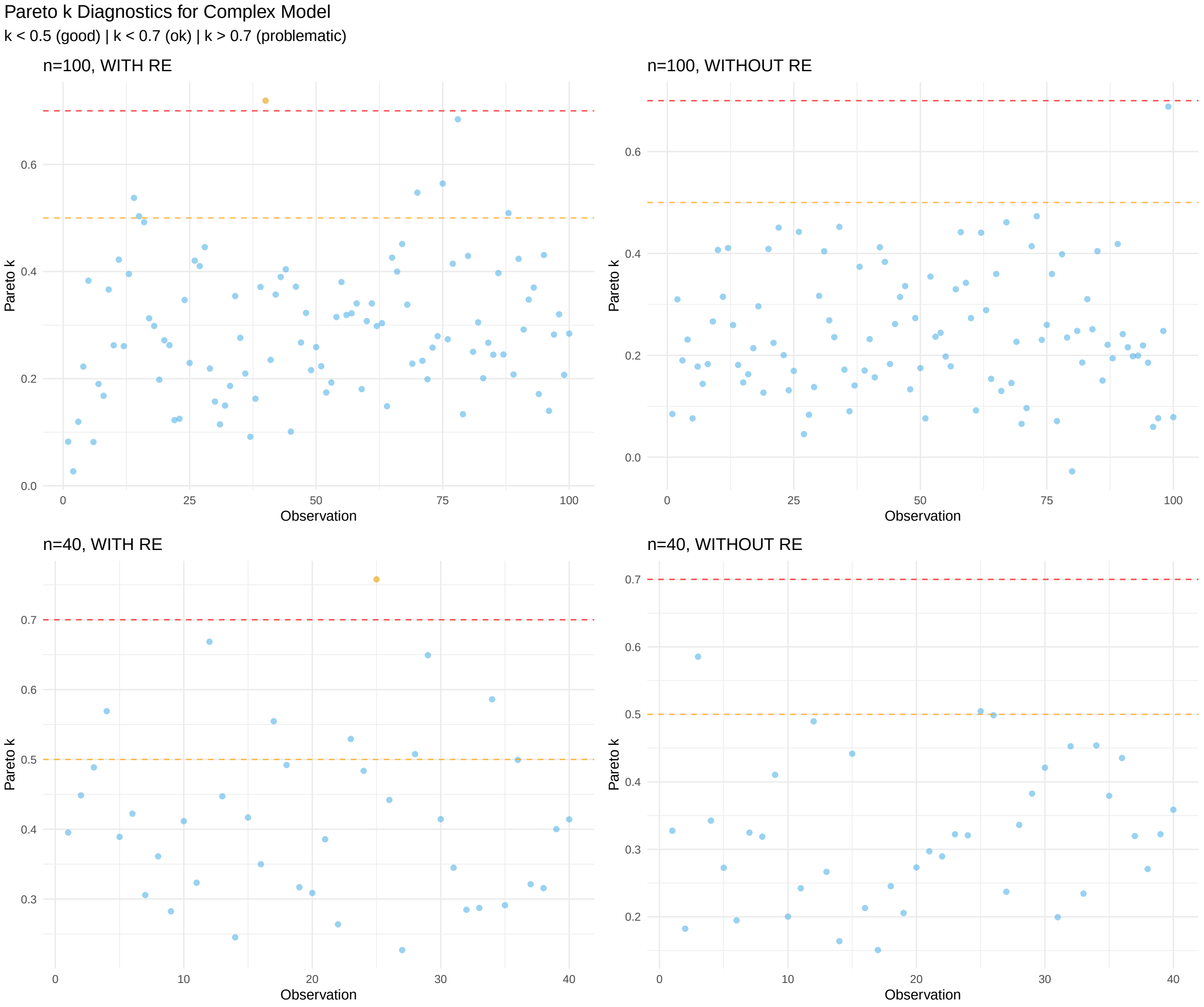
- **Weight 1.0**: Very high confidence in this model (it dominates predictions)

- **Weight 0.5**: Models are roughly equivalent (uncertain which is better)
- **Weight < 0.1**: Very low confidence (model contributes little to predictions)

Model weights represent the optimal combination of models for predictions. When one model has weight 1.0, LOO is very confident that model is superior.

2.4.4 Plot 4: Pareto k Diagnostics (2×2 Grid)

Pareto k values diagnose the reliability of the LOO approximation for each observation, with values below 0.7 indicating trustworthy estimates. High k values (> 0.7) suggest influential observations where the importance sampling approximation may be unreliable, requiring exact leave-one-out refitting with `reloo = TRUE`.



2.5 Key Insights from Four Scenarios

Table 5: Summary of LOO Behavior Across Scenarios

Scenario	Sample Size	True Structure	Expected Winner	Certainty	Key Lesson
n=100, WITH RE	Large	Complex	Complex	Very High	LOO strongly identifies complexity
n=100, WITHOUT RE	Large	Simple	Simple	Moderate	LOO avoids overfitting (weak preference)
n=40, WITH RE	Small	Complex	Complex	High	Strong evidence even with less data
n=40, WITHOUT RE	Small	Simple	Simple/Equiv	Low	Hard to distinguish with limited data

Main takeaways:

1. **LOO works best with adequate data** (n   100): Clear winners, confident weights
2. **LOO respects true data structure**: Finds complexity when it exists, avoids it when it doesn't
3. **Small samples = high uncertainty**: Model weights closer to 0.5, wider error bars
4. **Pareto k generally good**: Few problematic observations across all scenarios

3 Pareto k Diagnostics

3.1 Identifying Influential Points

The LOO calculation uses Pareto Smoothed Importance Sampling (PSIS). The Pareto k diagnostic tells us if the approximation is reliable:

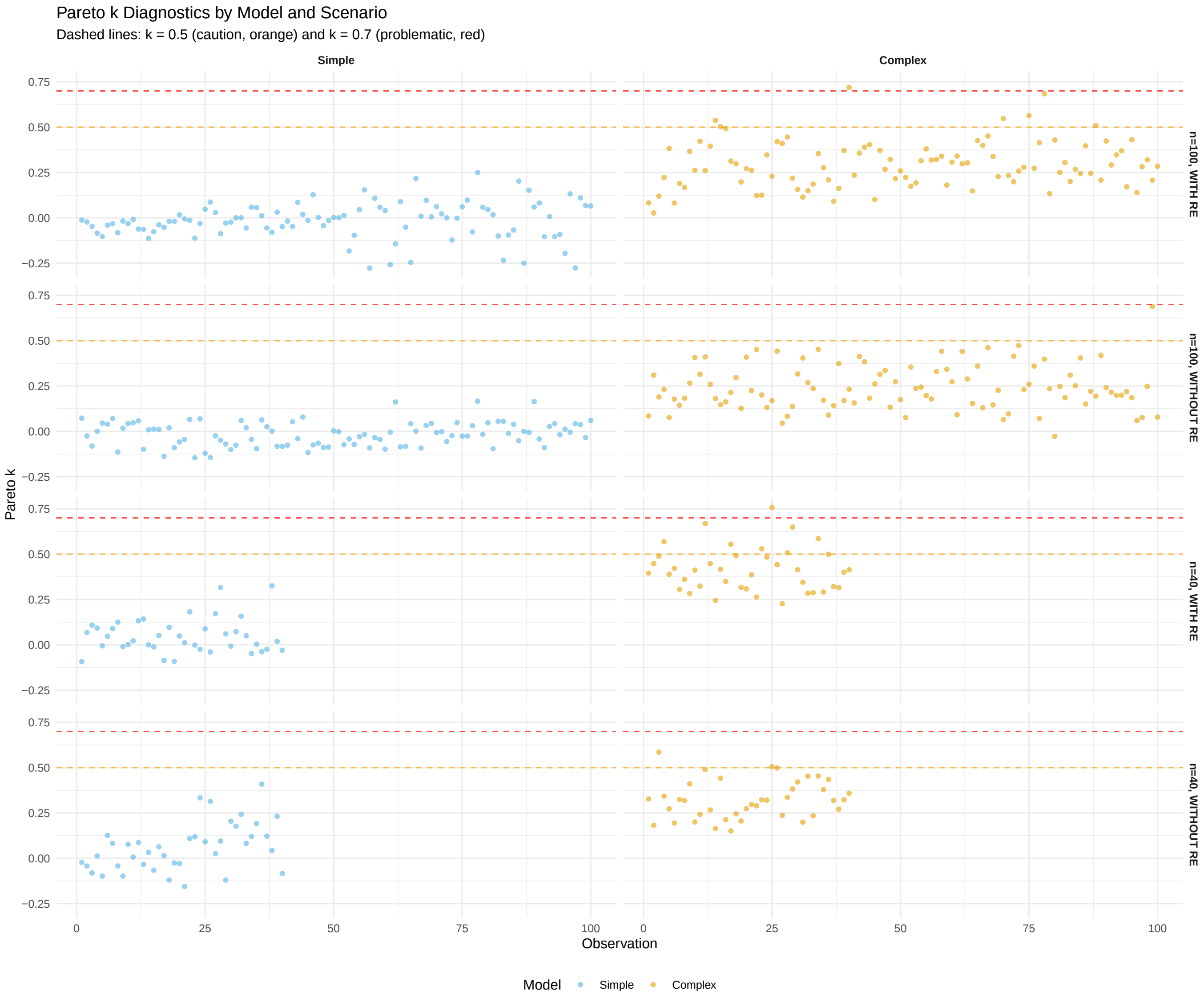
Pareto k thresholds (sample-size dependent):

- $k < 0.5$: Good (reliable estimate)
- $0.5 < k < 0.7$: Okay (use with caution)
- $k > 0.7$: Bad (LOO estimate unreliable)

Table 6: Pareto k Diagnostics Across Scenarios (threshold $k > 0.7$)

Scenario	Observations with $k > 0.7$	Status
n=100, WITH RE	1 / 100	Consider reloo = TRUE
n=100, WITHOUT RE	0 / 100	All k values good
n=40, WITH RE	1 / 40	Consider reloo = TRUE
n=40, WITHOUT RE	0 / 40	All k values good

3.2 Visualize Pareto k Values by Model and Scenario



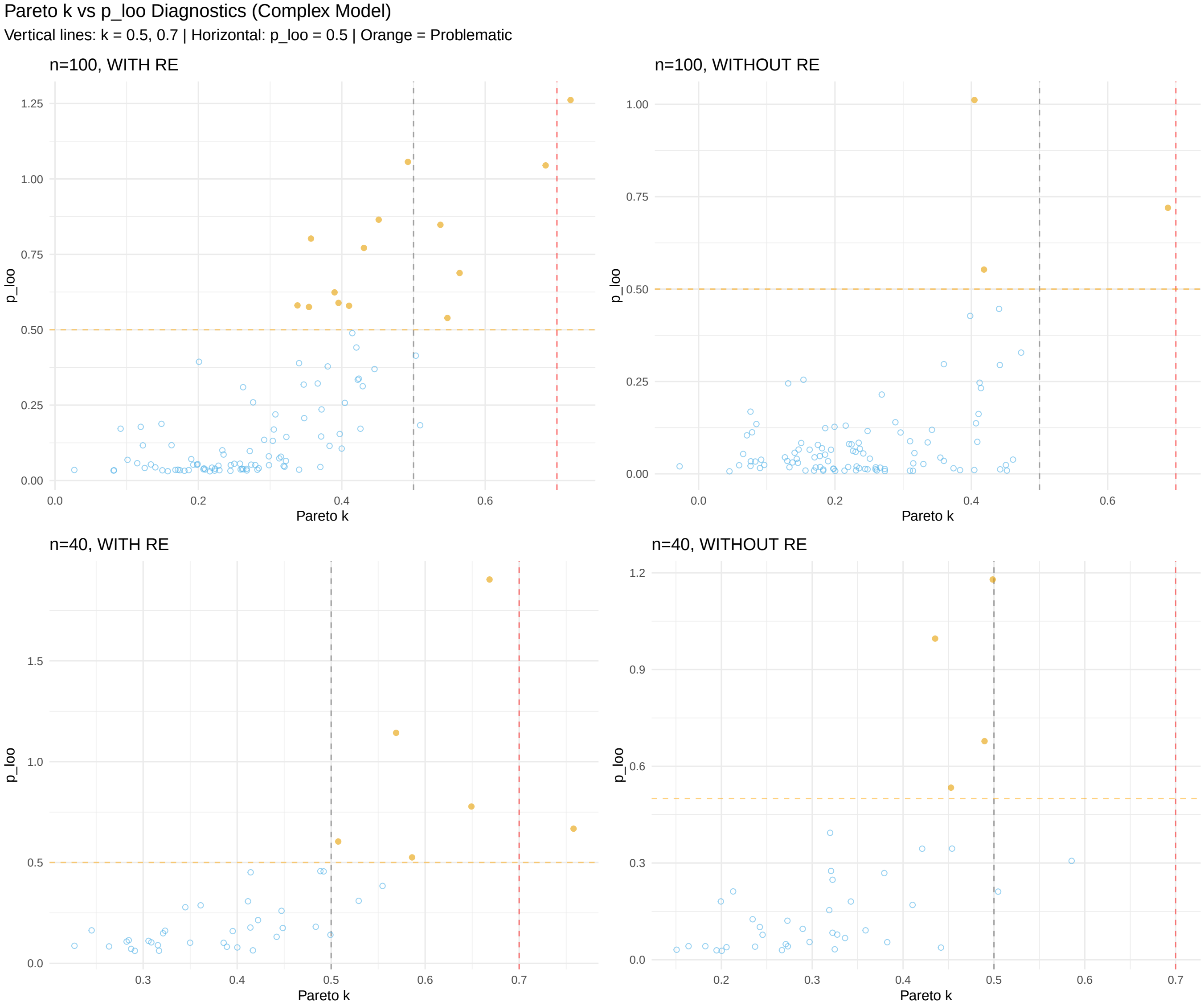
What to look for:

- Most points should be below 0.5 (good)
- Points between 0.5-0.7 (orange line) are okay but use with caution
- Points above 0.7 (red line) indicate unreliable LOO estimates
- Small sample scenarios (n=40) may show slightly higher k values due to limited data

3.3 Influential Observations: Pareto k vs p__loo

The relationship between Pareto k and p__loo (effective number of parameters per observation) can reveal influential observations:

- **p__loo** measures how much each observation influences the model
- **High p__loo + high k**: Very influential observation that's hard to predict
- **Low p__loo + high k**: Outlier that doesn't strongly influence the model
- **High p__loo + low k**: Normal influential observation (e.g., high leverage point)



Interpretation:

- Points in the **upper-right quadrant** (high k, high p_loo): Most concerning - influential outliers
- Points along the **right edge** (high k, low p_loo): Outliers with less model influence
- Points in the **upper-left** (low k, high p_loo): Normal high-leverage observations
- Most points should cluster in the **lower-left** (low k, low p_loo): Well-behaved observations

3.4 Handling Problematic Observations

When to use exact LOO refitting:

The Pareto k diagnostic has two key thresholds:

- **k > 0.7** (bad): PSIS approximation unreliable - **definitely refit** with exact LOO
- **k > 0.5** (concerning): PSIS approximation less accurate - **consider refitting** for critical analyses
- **k < 0.5** (good): PSIS approximation works well - no refitting needed

Trade-off: Refitting at k > 0.5 is more conservative and gives more accurate estimates, but it's computationally expensive (more observations to refit). For most purposes, k > 0.7 is sufficient.

```
Found 1 problematic observations for fit_complex_100_with
Loading cached reloo results for fit_complex_100_with
```

```
No problematic observations for fit_complex_100_without - using standard LOO
```

```
Found 1 problematic observations for fit_complex_40_with
Loading cached reloo results for fit_complex_40_with
```

```
No problematic observations for fit_complex_40_without - using standard LOO
```

What `reloo = TRUE` does:

1. Identifies observations with k > threshold (0.7 by default, or 0.5 if set)
2. Refits the model leaving each problematic observation out exactly
3. Uses exact LOO for problematic observations
4. Combines with PSIS-LOO for well-behaved observations

Note: Exact LOO refitting is computationally expensive (10-30 minutes per model at k > 0.7 threshold, potentially longer at k > 0.5). Results are cached in `fits/` directory with threshold encoded in filename (e.g., `_reloo_k07.rds` or `_reloo_k05.rds`).

3.5 Comparing Results Before and After Exact LOO

Check if exact LOO refitting changed the model comparison results:

```
No problematic observations for fit_simple_100_with - using standard LOO
```

```
No problematic observations for fit_simple_100_without - using standard LOO
```

```
No problematic observations for fit_simple_40_with - using standard LOO
```

```
No problematic observations for fit_simple_40_without - using standard LOO
```

Table 7: Problematic Observations by Model and Scenario (threshold k > 0.7)

Scenario	Model	Problematic (k>0.7)	Max k
n=100, WITH RE	Simple	0	0.250
n=100, WITH RE	Complex	1	0.719
n=100, WITHOUT RE	Simple	0	0.166
n=100, WITHOUT RE	Complex	0	0.688
n=40, WITH RE	Simple	0	0.325
n=40, WITH RE	Complex	1	0.758
n=40, WITHOUT RE	Simple	0	0.409
n=40, WITHOUT RE	Complex	0	0.585

Table 8: Impact of Exact LOO Refitting on Model Comparison

Scenario	Original |ELPD Δ|	Reloo |ELPD Δ|	ELPD Change	Original Ratio	Reloo Ratio	Ratio Change
n=100, WITH RE	117.21	117.29	0.08	12.68	12.72	0.04
n=100, WITHOUT RE	2.15	2.15	0.00	1.59	1.59	0.00
n=40, WITH RE	26.06	26.10	0.03	5.87	5.88	0.02
n=40, WITHOUT RE	1.62	1.62	0.00	0.80	0.80	0.00

Key findings:

- **Both models checked**: Simple and complex models are both refitted if they exceed threshold (k > 0.7 by default)
- **ELPD changes**: Shows how the difference between models changed after exact LOO
- **Ratio changes**: Shows if the strength of evidence changed (ratio = |ELPD Δ| / SE)
- **Typical pattern**: Changes are usually small (< 1 ELPD unit) unless observations are very influential
- **Interpretation**: Large changes suggest the original PSIS-LOO approximation was unreliable
- **Conservative option**: Set `k_threshold <- 0.5` at the top of this section to refit more observations (slower but more accurate)

4 Comparing WAIC and LOO

4.1 Understanding the Differences

Both WAIC and LOO estimate out-of-sample predictive accuracy, but they use different approaches:

WAIC (Watanabe-Akaike Information Criterion):

- **Method**: Uses the entire dataset at once
- **Approximation**: Based on asymptotic theory (assumes large sample sizes)
- **p_waic**: Estimates effective number of parameters from posterior variance
- **Pros**: Fast to compute, simple formula
- **Cons**: Can be unstable with small samples or influential observations, no diagnostics

LOO-PSIS (Leave-One-Out with Pareto Smoothed Importance Sampling):

- **Method**: Simulates leaving each observation out one at a time
- **Approximation**: Uses importance sampling (no asymptotic assumptions needed)
- **p_loo**: Estimates effective parameters from LOO differences
- **Pros**: More stable, includes diagnostics (Pareto k), works better with small samples
- **Cons**: Slightly slower (but still fast with PSIS)

Key technical differences:

Aspect	WAIC	LOO
Estimation	Posterior variance	Importance sampling
Diagnostics	None	Pareto k values
Small samples	Can be unstable	More robust
Influential obs	No warning	Flags with high k
Computation	Slightly faster	Fast enough

When they disagree:

- Different rankings suggest **influential observations** or **model instability**
- Check Pareto k diagnostics - high k values indicate LOO is more reliable
- WAIC may overestimate predictive accuracy when observations are very influential

Recommendation: Use LOO by default. The Pareto k diagnostics are invaluable for catching problems.

4.2 Computing Both Criteria

Table 10: Model Rankings: WAIC vs LOO Across Scenarios

Scenario	WAIC Winner	ELPD (WAIC)	LOO Winner	ELPD (LOO)	Agreement
n=100, WITH RE	Complex	49.4	Complex	48.4	Agree
n=100, WITHOUT RE	Simple	-42.1	Simple	-42.4	Agree
n=40, WITH RE	Complex	6.9	Complex	5.9	Agree
n=40, WITHOUT RE	Simple	-10.4	Simple	-10.4	Agree

Interpreting agreement/disagreement:

- **Rankings identical**: Both methods agree - conclusions are robust
- **Small differences in values**: Normal - both methods have uncertainty
- **Rankings differ**: Investigate! Check Pareto k diagnostics and look for influential observations

5 Cross-Validation Variants

Different CV strategies for different research questions:

LOO (Leave-One-Out):

- Default choice
- For general predictive performance
- Treats all observations as exchangeable

K-fold CV:

- Split data into K groups
- For multilevel models: sample from groups
- Useful for: predicting unseen data from existing subjects

LOGO-CV (Leave-One-Group-Out):

- Leave out entire groups (e.g., subjects)
- Tests generalization to **new subjects from the same population**
- Answers: “How well can we predict for unseen subjects?”
- Most conservative - isolates data from different subjects

Table 11: Cross-Validation Variants in brms

Method	Description	Use Case
loo()	Leave-one-out (approximate)	General predictive performance
kfold(K=10)	K-fold (random split)	Unseen observations (any subject)
kfold(K=5, folds=‘grouped’, group=‘subject’)	K-fold (grouped subjects, ~2 per fold)	Grouped prediction task
kfold(group=‘subject’)	True LOGO (each subject = 1 fold)	New subjects from same population

Technical note on fold construction:

- **`kfold(K=10)`**: Random split into K folds using `loo::kfold_split_random()`
- **`kfold(folds="stratified", group="x")`**: Stratified by variable x using `loo::kfold_split_stratified()`

- **`kfold(K=5, folds="grouped", group="subject")`**: Groups 10 subjects into 5 folds (~2 subjects per fold) using `loo::kfold_split_grouped()`
- **`kfold(group="subject")`**: True LOGO - each unique subject becomes one fold (K=10, ignores K parameter)

5.1 Comparing CV Variants: Do Prediction Goals Matter?

We’ll compare all three CV methods on the n=100 WITH RE scenario to see how different prediction goals affect model selection.

Why K-fold and LOGO are fast: Unlike traditional CV where you refit the model K times from scratch, `kfold()` in brms uses **approximate leave-out** via importance sampling (similar to LOO). It only refits observations with high Pareto k values. This means:

- **Fast**: Completes in seconds instead of hours
- **Accurate**: Exact refitting only when needed (high k values)
- **Efficient**: Reuses posterior samples from the original model fit

For most folds, the approximation works well. When it doesn’t ($k > 0.7$), brms automatically switches to exact refitting for just those problematic folds.

```
Loading cached 10-fold CV for fit_simple_100_with

Loading cached 10-fold CV for fit_medium_100_with

Loading cached 10-fold CV for fit_complex_100_with

Loading cached 5-fold grouped CV for fit_simple_100_with
Loading cached 5-fold grouped CV for fit_medium_100_with

Loading cached 5-fold grouped CV for fit_complex_100_with

Loading cached LOGO-CV for fit_simple_100_with

Loading cached LOGO-CV for fit_medium_100_with

Loading cached LOGO-CV for fit_complex_100_with
```

5.1.1 Visualizing CV Variant Results

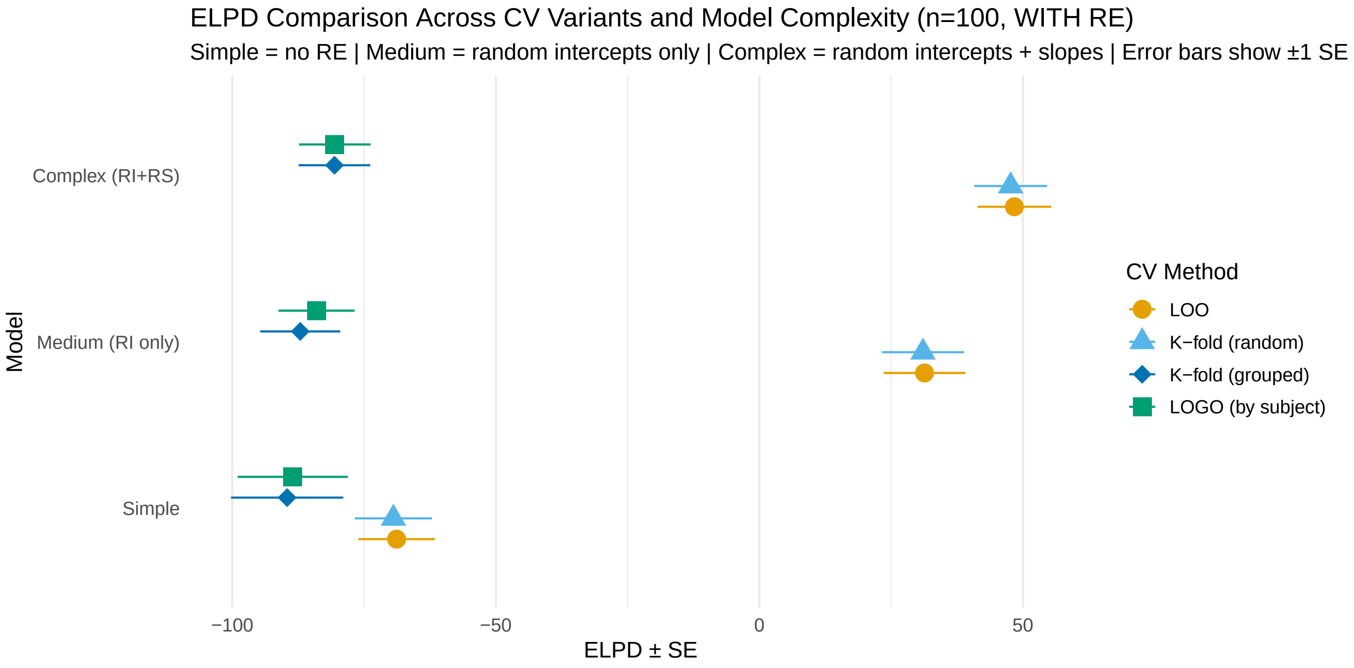


Table 12: ELPD estimates with standard errors for all CV methods and models

CV Method	Model	ELPD	SE
LOO	Complex (RI+RS)	48.4	7.0
LOO	Medium (RI only)	31.3	7.8
LOO	Simple	-68.8	7.3
K-fold (random)	Complex (RI+RS)	47.7	6.9
K-fold (random)	Medium (RI only)	31.0	7.8
K-fold (random)	Simple	-69.4	7.3
K-fold (grouped)	Complex (RI+RS)	-80.6	6.8
K-fold (grouped)	Medium (RI only)	-87.1	7.6
K-fold (grouped)	Simple	-89.6	10.6
LOGO (by subject)	Complex (RI+RS)	-80.6	6.8
LOGO (by subject)	Medium (RI only)	-84.0	7.2
LOGO (by subject)	Simple	-88.5	10.5

Finding:: The difference between Medium and Complex models is much larger for LOO and K-fold (random) (~17 ELPD) compared to K-fold (grouped) and LOGO (~3-4 ELPD). Why?

- **LOO/K-fold (random)**: Test prediction for **new observations from subjects already in the training data**
  - When predicting a left-out observation, the model has already seen other data points from that same subject
- **K-fold (grouped)/LOGO**: Test prediction for **completely unseen subjects**
  - When predicting for a new subject, the model has zero observations from that subject
  - Both Medium and Complex models must rely on population-level estimates only

Technical approach: To enable subject-based grouping for the **simple** models (model without random effects), we use **custom fold assignments** created with `loo::kfold_split_grouped()` and pass them via the `folds` parameter. This allows us to answer how well the simple model (which pools subjects) generalizes to new subjects compared to the complex model (which accounts for subject variability).

5.1.2 Comparing Winners Across CV Methods

Table 13: Model Comparison Across CV Variants (n=100, WITH RE)

CV Method	Winner	|ELPD Δ|	SE	Ratio	Interpretation
LOO	Complex	117.21	10.07	11.64	Very strong evidence
K-fold (random)	Complex	117.12	10.07	11.63	Very strong evidence
K-fold (grouped)	Complex	8.98	12.63	0.71	Weak evidence
LOGO (by subject)	Complex	7.98	12.48	0.64	Weak evidence

Key insights:

- **Three model types tested**:
  - **Simple**: No random effects (pools all subjects)
  - **Medium**: Random intercepts only `(1 | subject) + (1 | item)`
  - **Complex**: Random intercepts + slopes `(1 + condition | subject) + (1 | item)`
- **All CV methods work for all models**: Using custom fold assignments enables subject-based grouping for any model
- **Clear progression**: Medium consistently better than Simple; Complex best across all CV methods
- **CV method characteristics**:
  - **LOO**: Most optimistic (smallest SE) - general predictive performance
  - **K-fold (random)**: Some subjects in multiple folds
  - **K-fold (grouped)**: 10 subjects split into 5 groups (~2 per fold)
  - **LOGO**: Each subject = one fold (10 folds total) - most conservative
- **Pattern**: As CV becomes more conservative (more subject isolation), uncertainty increases
- **Random slopes matter**: Complex model's advantage over Medium shows that subject-specific condition effects improve generalization

Understanding LOGO-CV:

Critical interpretation: Lower absolute ELPD values in LOGO compared to LOO don't indicate a "bad" model - they reflect the inherent difficulty of predicting for completely new individuals. The **relative comparison** between models is what matters. If model differences remain consistent across CV methods, your conclusions are robust across different prediction scenarios.

5.1.3 When to Use Each Method

Table 14: Choosing the Right CV Method for Your Research Question

Research Scenario	CV Method	Why
Testing experimental effects	LOO	Efficient; random effects are nuisance parameters
Building predictive model for same subjects	K-fold	Captures uncertainty about specific observations
Generalizing to new subjects in same population	LOGO	Tests capacity to predict for unseen subjects
Clinical/applied prediction for new individuals	LOGO	Most relevant for real-world application

6 Summary and Best Practices

6.1 When to Use LOO

Use LOO for:

- Comparing model structures (e.g., with/without random slopes)
- Feature selection (which predictors to include?)
- Comparing different likelihoods (Gaussian vs. Student-t)
- Choosing between regularizing vs. non-regularizing priors
- Prediction (versus explanation / in-sample) tasks

Why not use LOO for hypothesis testing?

LOO answers: “Which model predicts better?” - a question about **out-of-sample prediction**.

But scientific hypotheses are about **in-sample effects**:

- “Does condition B produce longer RTs than condition A?” → Use posterior distribution of the condition effect
- “Is the effect significant?” → Calculate `P(  > 0 | data)` from posterior samples
- “How large is the effect?” → Report posterior mean/median and 95% credible interval

Example distinction:

```r
#   Wrong approach: Using LOO to test if condition matters
fit_with_condition <- brm(rt ~ condition + (1|subject), ...)
fit_without_condition <- brm(rt ~ 1 + (1|subject), ...)
loo_compare(fit_with_condition, fit_without_condition)
# Problem: Even if model with condition predicts better, this doesn't
# quantify the effect size or provide uncertainty about the parameter

#   Correct approach: Using posterior to test condition effect
fit <- brm(rt ~ condition + (1|subject), ...)
posterior_samples <- as_draws_df(fit)
```

```
mean(posterior_samples$b_conditionB > 0)  # Probability effect is positive
quantile(posterior_samples$b_conditionB, c(0.025, 0.975))  # 95% CI
```

6.2 Workflow Recommendations

6.2.1 Complete Workflow

Useful for: - First-time analysis of a new data type or domain - Publications, dissertations - When prior specification is contentious or novel - Demonstrating methodological rigor

Step 1: Setting priors (`01_setting_priors.qmd`)

- Define domain-appropriate priors
- Consider weakly informative vs. informative priors
- Document prior rationale

Step 2: Prior predictive checks (`02_prior_predictive_checks.qmd`)

- Simulate data from priors only (no observations)
- Verify priors generate plausible data ranges
- Catch unreasonable prior specifications

Step 3: Fit model and check convergence (later?)

- Fit model with data
- Check Rhat (< 1.01), ESS (> 400)
- Inspect trace plots if needed

Step 4: Posterior predictive checks (`03_posterior_predictive_checks.qmd`)

- Compare observed data to model predictions
- Check mean, SD, quantiles, and other test statistics
- Identify model misspecification

Step 5: Sensitivity analysis (`04_comparing_priors_rt.qmd`)

- Refit with alternative reasonable priors
- Compare posterior distributions
- Verify conclusions are robust to prior choice

Step 6: Model comparison with LOO (`05_loo.qmd`)

- Compare different model structures
- Use ELPD differences and model weights
- Check Pareto k diagnostics

Step 7: Hypothesis testing with ROPE (7 January 2026) and Bayes Factors (15 April 2026)

- Extract posterior distributions for parameters of interest
- Calculate credible intervals
- Use ROPE (Region of Practical Equivalence) for equivalence testing
- Bayes factors for specific hypothesis comparisons (if needed)

6.2.2 A Faster Workflow / Taking Shortcuts

1. **Set priors** - Use validated weakly informative defaults from previous work
2. **Fit model** - Standard model structure
3. **Check convergence** - Quick check: Rhat < 1.01, ESS > 400
4. **Posterior predictive checks** - Always verify model captures data features
5. **Interpret parameters** - Posterior means/medians and credible intervals

Add when needed:

- **Prior predictive checks** - Only when using new informative priors
- **Sensitivity analysis** - When results are unexpected or borderline
- **LOO** - Only when comparing multiple plausible model structures
- **ROPE/Bayes factors** - Only when equivalence testing or null hypothesis quantification is required

6.3 Reporting LOO Results

Minimal reporting:

```
We compared three models using LOO-CV on n=100 observations: simple
(no random effects), medium (random intercepts for subjects and items),
and complex (random intercepts plus random slopes for condition by
subject). The complex model showed the best predictive performance
(ELPD = 48.4, SE = 7.0), substantially outperforming the medium model
(ELPD = 31.3, SE = 7.8) and the simple model (ELPD = -68.8, SE = 7.3).
The difference between complex and simple models was 117.2 ELPD units
(SE = 10.1, ratio = 11.6), providing very strong evidence for the
complex model. Only 1 of 100 observations had Pareto k > 0.7, indicating
generally reliable LOO estimates.
```

6.4 Session Info

```
R version 4.4.1 (2024-06-14)
Platform: x86_64-pc-linux-gnu
Running under: Ubuntu 22.04.5 LTS

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.20.so;  LAPACK version 3.10.

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

time zone: Etc/UTC
tzcode source: system (glibc)
```

```
attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] rstan_2.32.7         StanHeaders_2.32.10  patchwork_1.3.2
 [4] loo_2.8.0            posterior_1.6.1.9000 bayesplot_1.14.0
 [7] lubridate_1.9.3      forcats_1.0.0        stringr_1.5.1
[10] dplyr_1.1.4          purrr_1.0.2          readr_2.1.5
[13] tidyr_1.3.1          tibble_3.2.1         ggplot2_4.0.0
[16] tidyverse_2.0.0      brms_2.23.0          Rcpp_1.0.13

loaded via a namespace (and not attached):
 [1] gtable_0.3.6          tensorA_0.36.2.1      QuickJSR_1.8.1
 [4] xfun_0.54             processx_3.8.4        inline_0.3.21
 [7] lattice_0.22-6        tzdb_0.4.0            ps_1.8.1
[10] vctrs_0.6.5           tools_4.4.1           generics_0.1.3
[13] stats4_4.4.1          parallel_4.4.1        fansi_1.0.6
[16] cmdstanr_0.9.0        pkgconfig_2.0.3       Matrix_1.7-0
[19] checkmate_2.3.3       RColorBrewer_1.1-3    S7_0.2.0
[22] distributional_0.5.0  RcppParallel_5.1.11-1 lifecycle_1.0.4
[25] compiler_4.4.1        farver_2.1.2          Brobdingnag_1.2-9
[28] tinytex_0.53          codetools_0.2-20      htmltools_0.5.8.1
[31] yaml_2.3.10           pillar_1.9.0          bridgesampling_1.1-2
[34] abind_1.4-8           nlme_3.1-164          tidyselect_1.2.1
[37] digest_0.6.37         mvtnorm_1.3-3         stringi_1.8.4
[40] labeling_0.4.3        fastmap_1.2.0         grid_4.4.1
[43] cli_3.6.5             magrittr_2.0.3        pkgbuild_1.4.8
[46] utf8_1.2.4            withr_3.0.2           scales_1.4.0
[49] backports_1.5.0       estimability_1.5.1    timechange_0.3.0
[52] rmarkdown_2.30        matrixStats_1.5.0     emmeans_2.0.0
[55] gridExtra_2.3         hms_1.1.3             coda_0.19-4.1
[58] evaluate_1.0.1        knitr_1.50            rstantools_2.5.0
[61] rlang_1.1.6           xtable_1.8-4          glue_1.8.0
[64] jsonlite_1.8.9        R6_2.5.1
```