

## Contents

<b>Common Misconceptions about Random Effects in Confirmatory Analyses</b>	<b>2</b>
Misconception 1: Mixed-effects models have dependency assumptions not shared by other kinds of analysis . . . . .	2
Misconception 2: Random intercepts and random slopes have no analogue in ANOVA . .	2
Misconception 3: Conventional ANOVA, unlike LMEMs, doesn't allow for testing random slopes . . . . .	7
<b>Additional simulation results</b>	<b>16</b>
<b>Supplementary simulations and analysis</b>	<b>20</b>
Effect of sample size . . . . .	20
Effect of non-homogeneous data loss . . . . .	22
Models without random intercepts or random correlations . . . . .	35
Random effects in real datasets . . . . .	46
Validity of random effects parameter estimates from nonconverged models . . . . .	51
Random effects for control predictors . . . . .	54
Performance on datasets with high random correlations . . . . .	56
<b>The <i>simgen</i> package</b>	<b>58</b>
Download and Installation . . . . .	58
<b>R scripts for Monte Carlo simulations</b>	<b>59</b>
Standard runs (no model selection) . . . . .	59
Model-selection runs . . . . .	61

This is the online appendix to Barr, D. J., Levy, R., Scheepers, C. & Tily, H. J. (2013), Random effects structure for confirmatory hypothesis testing: Keep it maximal, *Journal of Memory and Language*. The appendix is also currently available online at <http://talklab.psy.gla.ac.uk/simgen>.

### Common Misconceptions about Random Effects in Confirmatory Analyses

This section is intended for people using linear mixed effects models (LMEMs) as a replacement for the statistical techniques that are more traditionally used for confirmatory hypothesis testing, such as ANOVA or t-tests. Some of the points we make here may not apply to other uses of LMEMs, such as for deriving the ‘most parsimonious’ model for one’s data.

#### **Misconception 1: Mixed-effects models have dependency assumptions not shared by other kinds of analysis**

LMEMs share the assumption of independent observations (technically, conditional independence, i.e., independence of the residuals) with other parametric approaches. LMEMs do allow for greater flexibility in accounting for sources of random variation than more traditional approaches, and it is this additional flexibility that makes it possible to simultaneously generalize over multiple populations (e.g., subjects and items). But this greater flexibility requires us to think more explicitly about random effects.

It is easy to miss the point that assumptions about independence have been there all along, because many people follow conventional “recipes” when they analyze their data that allow them to effectively ignore choices about random effects structure they would otherwise have to confront. For most researchers, the only moment in an analysis in which they think about anything like a “random effect” is when they consider whether to perform an independent or correlated-samples t-test, or decide among the recipes for “between-subjects ANOVA”, “repeated-measures ANOVA”, or “mixed design ANOVA” (not to be confused with mixed model ANOVA explained below). People mostly avoid error in these cases because they know how to follow the recipes.

As these recipes are currently practiced, the first step usually involves aggregating the data so that the test can be performed on the subject-by-cell means (or item-by-cell means) rather than on the raw, unaggregated data. This “aggregate-before-analysis” step creates a dataset that meets the requirement of conditional independence for the slope (condition effect), by having only one data point per subject per cell. This practice of aggregating first is so ingrained that many researchers believe that ANOVA only “allows” one data point per-subject-per-cell, which is not correct (see below). To meet the assumption of conditional independence for the *intercept*, it is simply necessary to choose the right recipe, i.e., to make sure that one does a repeated-measures ANOVA or paired-sample t-test instead of a between-subject ANOVA or independent samples t-test.

#### **Misconception 2: Random intercepts and random slopes have no analogue in ANOVA**

Most datasets in experimental psychology involve multiple observations on the same sampling units (subjects or stimuli). It is possible to aggregate these observations together and to analyze the means rather than the trial-level data. When one wishes to analyze a dataset containing only a single observation per sampling unit, then between-subject techniques are warranted.

Datasets including multiple observations on the same units are ‘within-subject’ designs, analyzed using paired t-tests or within-subjects ANOVA. Such analyses control for variation in each subject

overall response level, akin to including a random intercept in a LMEM.

The notion of random slopes is functionally equivalent to the notion of a treatment-by-subject interaction in ANOVA. Most people don't know about "treatment-by-subject interactions" because they usually perform analyses on the subject means rather than on the unaggregated data. The aggregation process confounds the treatment-by-subject variance with residual error, which legitimates the use of the standard residual error term in the denominator of the  $F$  ratio.

The tendency to analyze the subject means has led to the widespread misconception that ANOVA only 'allows' one observation per subject per cell. It is, in fact, possible to analyze the unaggregated, trial-level data using ANOVA and specifying the treatment-by-subject interaction as the error term in the denominator of the  $F$  ratio (see below and also [here](#)).

Here is how one might analyze subject-by-condition means using ANOVA in R. We are using the R package [simgen](#) to generate a sample dataset with 24 subjects and 24 items, and with a single, two-level factor manipulated within subjects and within items.

```
library(simgen, quietly=TRUE)

x <- mkDf(mcr.params=createParamMx(1)[1,])
x$Cond <- factor(x$Cond)

x.agg <- with(x, aggregate(list(Resp=Resp),
                              list(Cond=Cond, SubjID=SubjID),
                              mean, na.rm=TRUE))

x.agg.aov <- aov(Resp ~ Cond + Error(SubjID), x.agg)
summary(x.agg.aov)
```

```
Error: SubjID
      Df Sum Sq Mean Sq F value Pr(>F)
Residuals 23  20.31   0.8829
```

```
Error: Within
      Df Sum Sq Mean Sq F value Pr(>F)
Cond    1  0.048   0.0479   0.091  0.766
Residuals 23 12.092   0.5257
```

We specified the error term using `Error(SubjID)`, which makes this a within-subject design. Note that we have 23 residual degrees of freedom, one minus the number of subjects. Equivalently, we could have run a correlated-samples t-test on the aggregated data. The analysis we performed here is actually equivalent to a random-intercepts only LMEM. Note that performing a random-intercepts only LMEM on the *aggregated* data would be legitimate because the treatment-by-subject interaction is no longer a source of dependency, since we only have one observation per subject per

condition. It is also worth noting that using an LMEM on aggregated data might be OK for a dataset sampled from a single population (i.e., subjects) but the aggregation would prevent us from using LMEMs for a simultaneous by-subject and by-item analysis, since the item information would be lost in the aggregation over subjects, and the subject information would be lost in the aggregation over items.

And now here is a different way to analyze the same data, using the raw, unaggregated trial-level observations rather than the subject means.

```
x.aov <- aov(Resp ~ Cond + Error(SubjID/Cond), x)
summary(x.aov)
```

```
Error: SubjID
      Df Sum Sq Mean Sq F value Pr(>F)
Cond    1    0.8   0.801   0.079  0.782
Residuals 22 224.5  10.206
```

```
Error: SubjID:Cond
      Df Sum Sq Mean Sq F value Pr(>F)
Cond    1    0.75   0.748   0.124  0.728
Residuals 23 138.36   6.015
```

```
Error: Within
      Df Sum Sq Mean Sq F value Pr(>F)
Residuals 500 1980   3.959
```

Note that we have specified the treatment-by-subject error term using the syntax `Error(SubjID/Cond)`. We should use the error term from this “stratum” (`SubjID:Cond`) when assessing the reliability of the effect. Note also that we have an  $F$  statistic 23 denominator degrees of freedom. This is functionally equivalent to a “random-intercept-and-random-slope” (i.e., a maximal) LMEM.

In sum, both of the above analyses control for variance in overall performance (random intercept variance) by requesting the total error to be partitioned by subject. Both analyses control for variance in susceptibility to the experimental treatment, but in different ways: the first by confounding it with residual error, and the second by assessing the treatment effect against the treatment-by-subject interaction. The second version is analogous to a maximal LMEM, and is also implemented as a default in *mixed-model ANOVA* (see [here](#) for mixed-model ANOVA in SPSS).

What would happen if we used the first method, but performed the analysis on the unaggregated, trial-level data rather than the subject means? Most people would consider such an analysis to be “wrong”—and rightly so. However, this is effectively what one does when fitting a random-intercepts-only LMEM to trial-level data. Let’s see what happens in the ANOVA case.

```
x.aov.bad <- aov(Resp ~ Cond + Error(SubjID), x)
summary(x.aov.bad)
```

Error: SubjID

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Cond	1	0.8	0.801	0.079	0.782
Residuals	22	224.5	10.206		

Error: Within

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Cond	1	0.7	0.748	0.185	0.667
Residuals	523	2118.1	4.050		

Note that we have two error terms against which we can test the effect of condition: (1) the subject error term, which doesn't seem right because it corresponds to variation due to differences in overall performance; and (2) the residual error term. If we used the latter, then note that we would have 523 denominator degrees of freedom, even though we only have 24 subjects in the experiment.

In case one needs any convincing of the analogy between a random-intercepts-only LMEM on unaggregated data and this obviously wrong ANOVA, the following simulation should remove all doubt. In the simulation, we generate 1000 datasets, each containing a single two-level treatment variable administered within subjects. There are 24 subjects and 12 replications per cell per subject. For all datasets, we assume there was no effect of the experimental manipulation (i.e.,  $H_0$  is true). Five analyses were performed on each data set:

1. `lmer.RI`: random-intercepts-only LMEM on the unaggregated data;
2. `aov.RI`: "bad" ANOVA on the unaggregated data (controlling for subject variance but not treatment-by-subject variance);
3. `aov.agg`: within-subjects ANOVA on the subject cell means;
4. `aov.RS`: within-subjects ANOVA on the unaggregated data, using the treatment-by-subject interaction as the error term (denominator) for the  $F$ ;
5. `lmer.RS`: a maximal LMEM (by-subject random intercepts and slopes)

```
library(lme4)
nsubj <- 24          # number of subjects
nrep <- 12           # number of replications per cell per subj
numberofruns <- 1000 # number of Monte Carlo runs

p.values <- matrix(nrow=numberofruns, ncol=5,
  dimnames=list(1:numberofruns,
    c("lmer.RI", "aov.RI", "lmer.RS", "aov.RS", "aov.agg")))

for (i in 1:numberofruns) {
  # generate data frames for two condition experiment
  # 24 subjects, 12 replications per cell per subject
  x <- data.frame(SubjID=factor(rep(1:nsubj, each=nrep*2)),
    A=factor(rep(c("A1", "A2"), each=nrep)))

  # response contains ONLY NOISE
  x$Resp <-
    rep(rnorm(nsubj, sd=3), each=nrep*2) + # random intercept
    rep(rep(rnorm(nsubj, sd=3), each=2)*c(-.5, .5), each=nrep) + # random slope
    rnorm(nsubj*nrep*2, sd=6) # residual error
```

```

# analysis on unaggregated data
x.aov.RI <- summary(aov(Resp ~ A + Error(SubjID), data=x)) # WRONG: violation of independence
x.aov.RS <- summary(aov(Resp ~ A + Error(SubjID/A), data=x)) # RIGHT: correct error term (A:Subject)

x.lmer.RI <- lmer(Resp ~ A + (1 | SubjID), data=x) # random intercept model
x.lmer.RS <- lmer(Resp ~ A + (1 + A | SubjID), data=x) # random slope model

# aggregate the data
x.agg <- with(x, aggregate(list(Resp=Resp), list(SubjID=SubjID,A=A), mean))

# ANOVA on aggregated data
x.agg.aov <- summary(aov(Resp ~ A + Error(SubjID), data=x.agg))

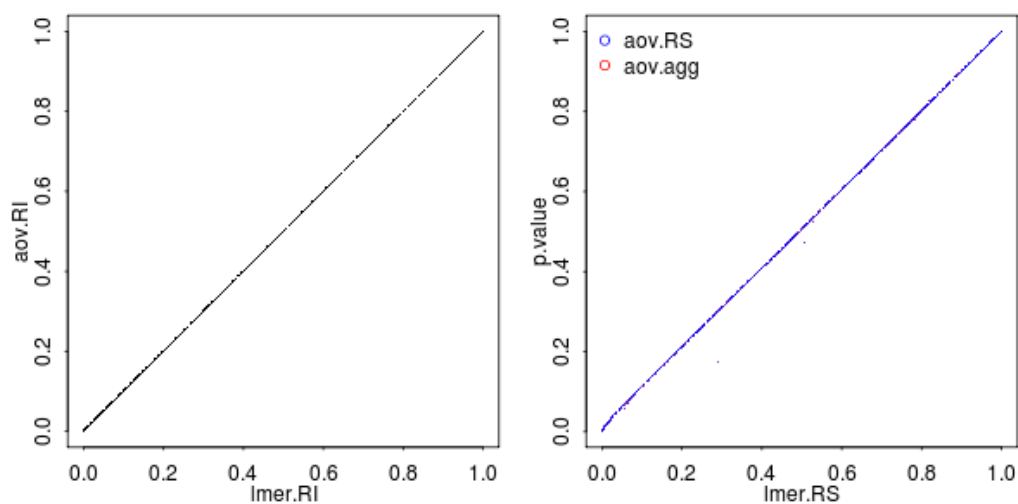
# store the data in the matrix
p.values[i,] <- c(2*(1-pnorm(abs((fixef(x.lmer.RI)/sqrt(diag(vcov(x.lmer.RI))))[2]))),
  x.aov.RI[[2]][[1]]$`Pr(>F)`[1],
  2*(1-pnorm(abs((fixef(x.lmer.RS)/sqrt(diag(vcov(x.lmer.RS))))[2]))),
  x.aov.RS[[2]][[1]]$`Pr(>F)`[1],
  x.agg.aov[[2]][[1]]$`Pr(>F)`[1])
}

```

Here are the resulting Type I error rates:

lmer.RI	0.193
aov.RI	0.192
lmer.RS	0.068
aov.RS	0.06
aov.agg	0.06

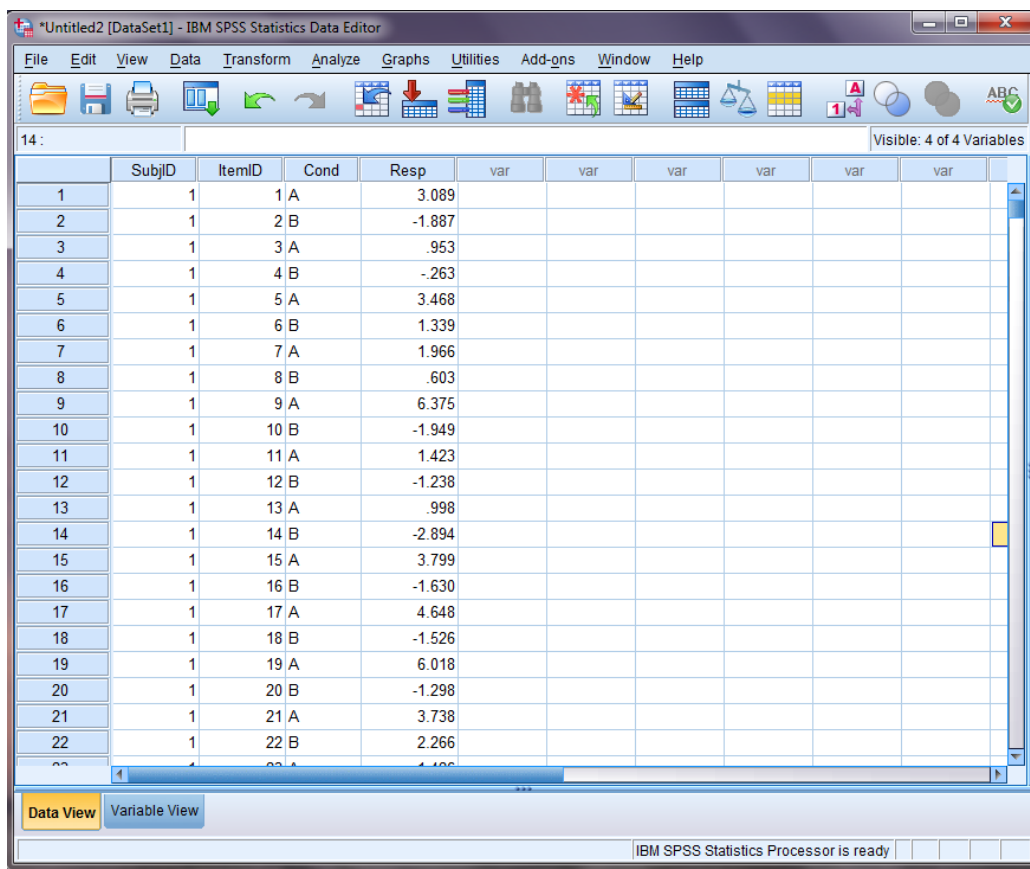
The figure below displays the correlations among  $p$  values as further evidence for the analogy between `lmer.RI` and `aov.RI` (left panel), as well as among `lmer.RS`, `aov.agg`, and `aov.RS` (right panel).



### **Misconception 3: Conventional ANOVA, unlike LMEMs, doesn't allow for testing random slopes**

Though rarely used in practice, model selection procedures have long been available for mixed-model ANOVA, just as for LMEMs. It is also possible to test random slopes. In this section, we will explain how to analyse repeated-measures data using the mixed-model ANOVA approach in SPSS (Version 19) *General Linear Models*. Although most psycholinguists are well acquainted with SPSS, many may not be familiar with the mixed-model ANOVA option because they usually follow the “standard recipe” of performing repeated-measures ANOVAs on aggregated data (see earlier sections). Here we will show that it is possible to perform ANOVAs on non-aggregated data as well, i.e. with more than one observation per subject(or item)-by-condition cell. We will further demonstrate that the mixed-model ANOVA approach provides significance tests not only for fixed effects, but also for random effects—with the latter corresponding to random intercepts and slopes in LMEMs. Finally, we will show that it is possible to *simplify* the effects structure of a mixed-model ANOVA, thereby providing all the ingredients necessary for *model selection*. However, note that—in stark contrast to what seems to be “accepted practise” in LMEMs—model selection has never been adopted as a standard for confirmatory hypothesis testing in ANOVA (and rightly so, as the analyses in our paper suggest). We hope this tutorial will help to illustrate the strong conceptual parallels between mixed-model ANOVAs and LMEMs—apart from the most obvious difference that LMEMs allow for *simultaneous* generalization of effects over subjects and items, whereas mixed-model ANOVA requires separate subject and item analyses due to its inability to handle “crossed” random factors.

We start with a 24-subjects/24-items (within/within) data set taken from our large pool of simulated two-condition experiments. For convenience, we assume a balanced data set with no missing values (note, however, that mixed-model ANOVA can handle unbalanced data sets as well). The data are in “long” format such that each row represents a single trial; the variables *SubjID* (1-24), *ItemID* (1-24), *Condition* (A, B), and *Response* (continuous DV) are coded in separate columns:



14 : Visible: 4 of 4 Variables

	SubjID	ItemID	Cond	Resp	var	var	var	var	var	var
1	1	1	A	3.089						
2	1	2	B	-1.887						
3	1	3	A	.953						
4	1	4	B	-.263						
5	1	5	A	3.468						
6	1	6	B	1.339						
7	1	7	A	1.966						
8	1	8	B	.603						
9	1	9	A	6.375						
10	1	10	B	-1.949						
11	1	11	A	1.423						
12	1	12	B	-1.238						
13	1	13	A	.998						
14	1	14	B	-2.894						
15	1	15	A	3.799						
16	1	16	B	-1.630						
17	1	17	A	4.648						
18	1	18	B	-1.526						
19	1	19	A	6.018						
20	1	20	B	-1.298						
21	1	21	A	3.738						
22	1	22	B	2.266						

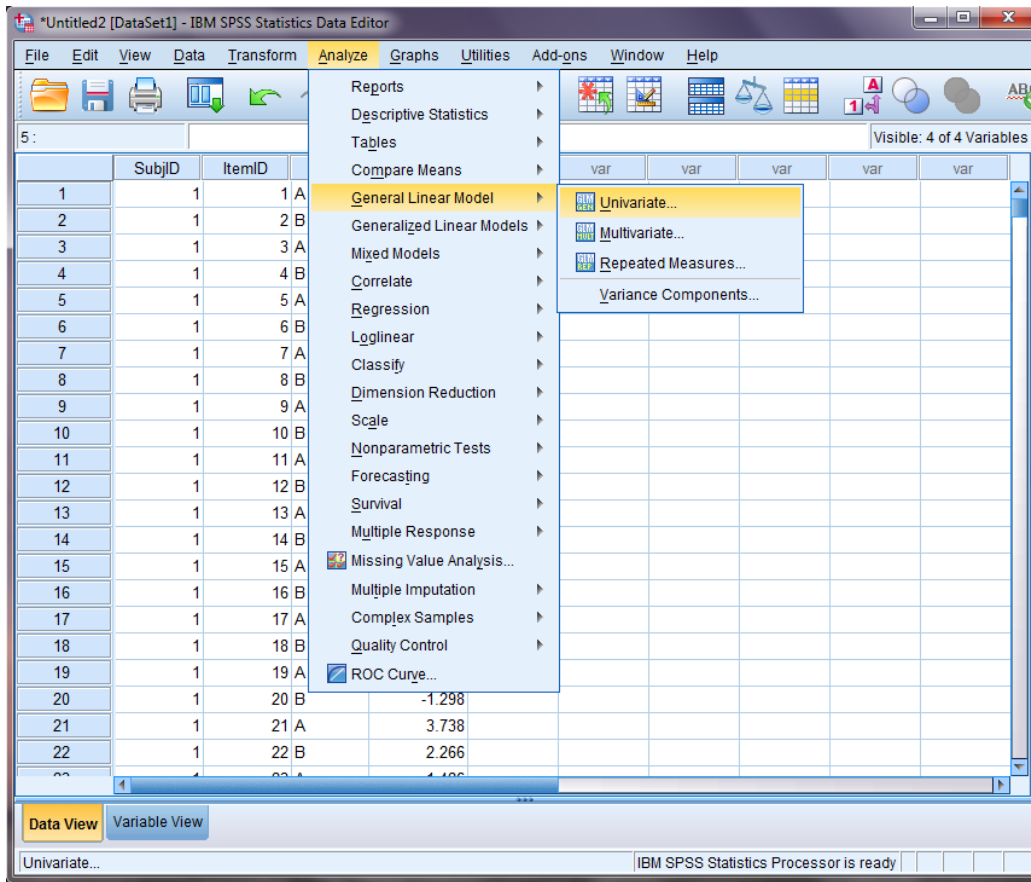
Data View Variable View

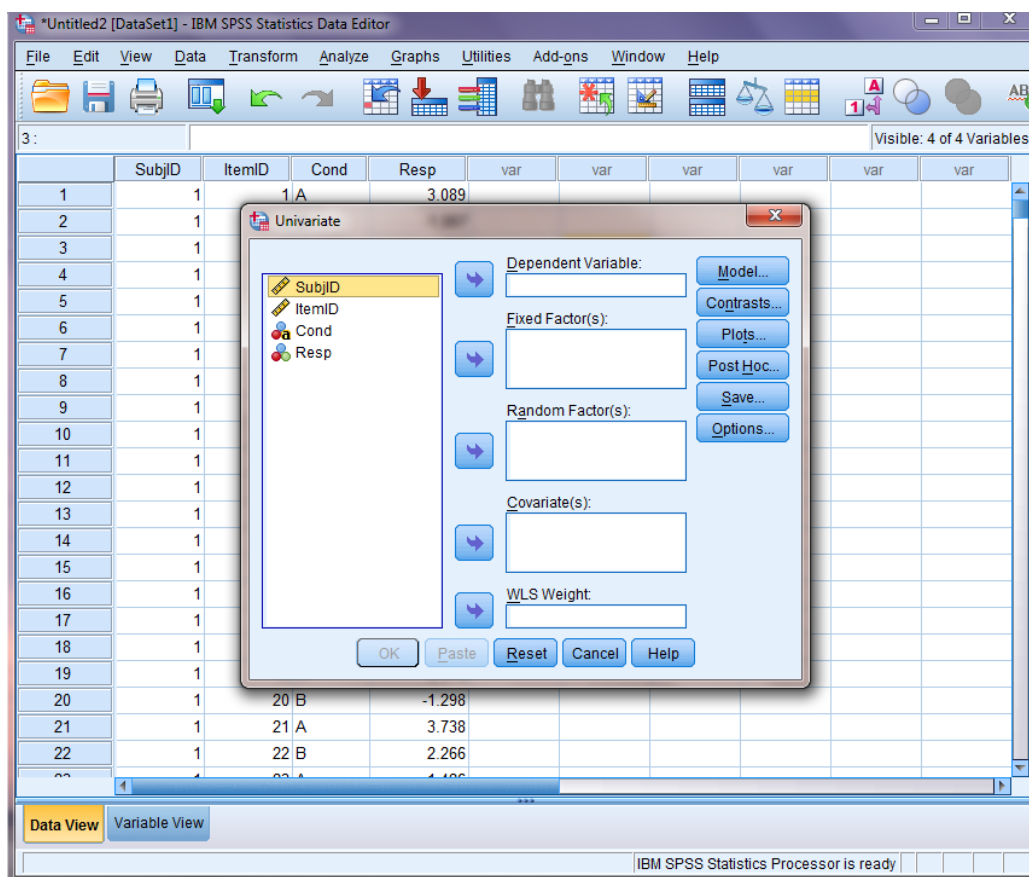
IBM SPSS Statistics Processor is ready

According to the “standard recipe” of performing ANOVA, one would normally proceed along the following three steps: (1) condense the data into subject-by-condition means (for F1) respectively item-by-condition means (for F2) using the SPSS procedure AGGREGATE; (2) re-arrange the aggregated data into “wide” format such that each condition is in a separate column; and finally (3) perform a repeated-measures ANOVA using *Analyze: General Linear Model: Repeated Measures...*

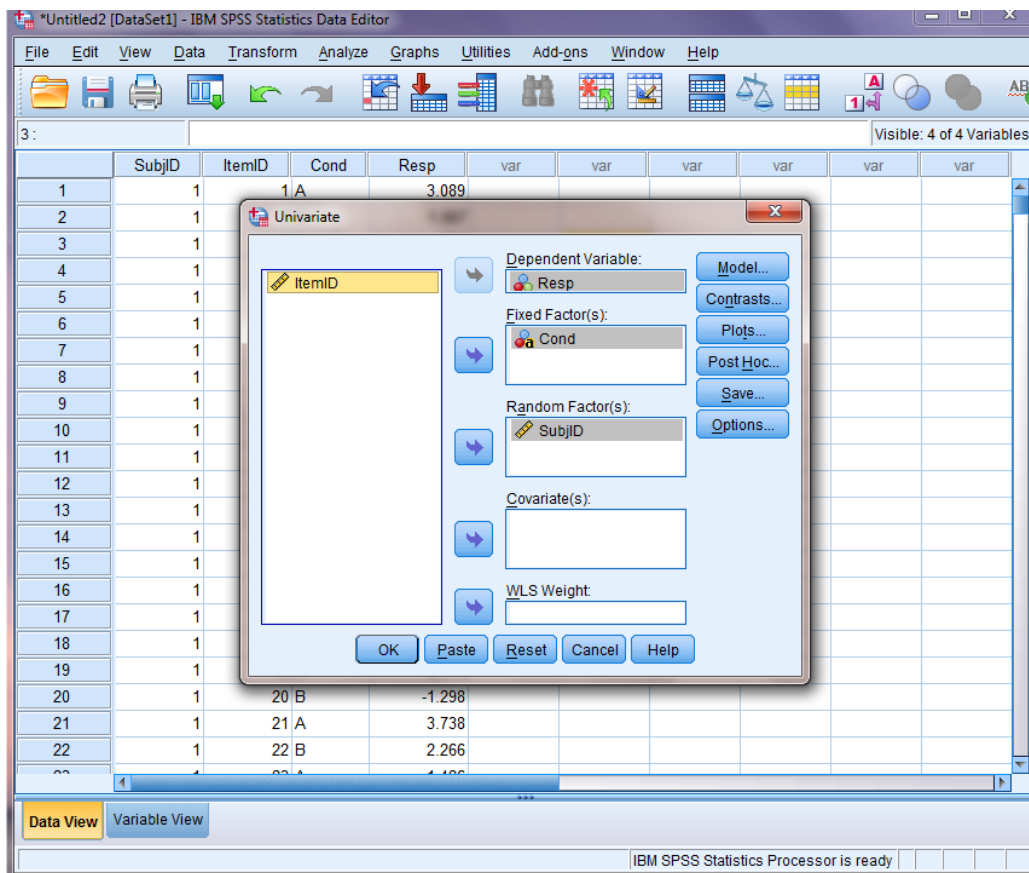
Here, we will not follow this recipe and perform a mixed-model ANOVA instead. To do this, we leave the data in their original “long” format (no aggregation!) and use the procedure *Analyze: General Linear Model: Univariate...*, which opens up a command window comprising a complete list of variables on the left and five empty fields (“Dependent Variable”, “Fixed Factor(s)”, etc.) on the right.



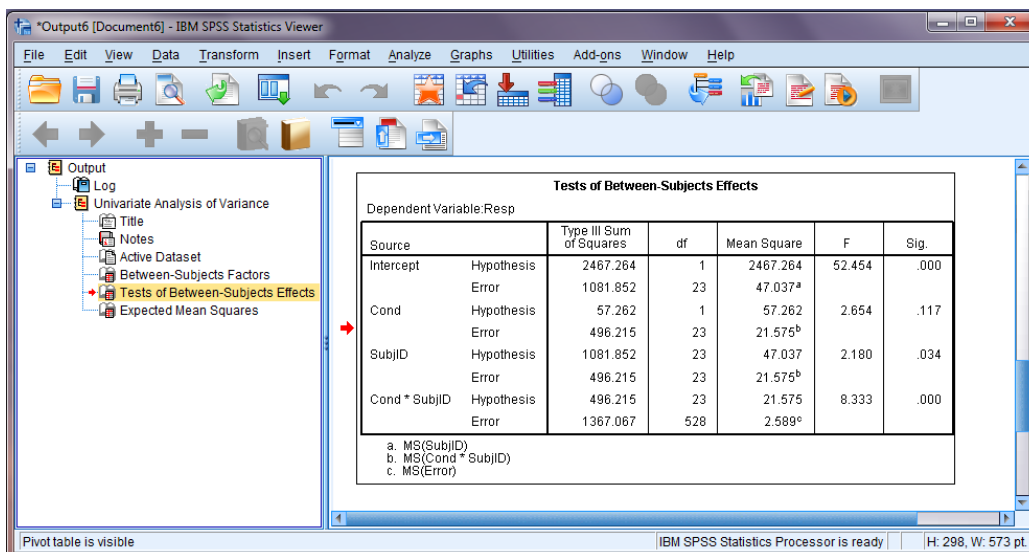




In this brief demonstration, we will perform a by-subjects (F1) analysis—a by-items (F2) analysis would essentially work the same way. First, we need to specify the dependent variable by highlighting *Resp* in the variable list and moving it into the “Dependent Variable” field (via clicking on the corresponding right-arrow button). Next, our experimental treatment variable *Cond* is moved into the “Fixed Factor(s)” field (in a factorial fixed-effect design, this field would contain more than one variable). Finally, since we want to perform an F1-analysis, our participant-variable *SubjID* ends up in the “Random Factor(s)” field. [As in LMEM, random factors are treated differently from fixed factors, but note that LMEMs and mixed-model ANOVAs employ different algorithms for estimating random effect variability in the population, which we will not discuss here.] The other two fields (“Covariate(s)” and “WLS Weight”) remain empty because there are no continuous predictors or weight variables to consider in our analysis:



To run the analysis, simply click “OK” and examine the results in the Output Navigator (or Statistics Viewer or whatever it’s called these days...):



SPSS tries to confuse us by labelling the critical results table “Tests of Between-Subjects Effects”—

but we won't be fooled by that, knowing that the table actually contains the appropriate inferential results for the *within-subjects* treatment effect of *Cond*.

The first important point to note is that the table provides statistical tests not only for the fixed effect of interest (here, the main effect of *Cond*), but also for (i) the random main effect of *SubjID* and (ii) the random *Cond*×*SubjID* interaction. As repeatedly stated in previous sections (as well as in the main paper itself), (i) is conceptually equivalent to the by-subject random *intercept* and (ii) is equivalent to the by-subject random *slope* in LMEM. The fact that all these effects appear in the SPSS output-table already highlights an important default assumption in mixed-model ANOVA: the procedure automatically assumes a *full-factorial* design, which implies a *maximal* (by-subjects) random effects structure against which the fixed effect of interest is tested! And although this default can be overridden quite easily (see further below), we are not aware of any published work that has deviated from the default when testing hypotheses in mixed-model ANOVA.

Another important point to note—also observe the footnotes in the SPSS output table—is that the fixed effect of interest is appropriately tested against an error term that corresponds to the random *Cond*×*SubjID* interaction (equivalent to the by-subject random *slope* in LMEM). In other words, the procedure tests **by default** whether the experimental treatment effect generalizes *above and beyond* subject-specific random variation in displaying that effect. Correspondingly, the “error” degrees of freedom for the *Cond* fixed effect are identical to the “effect” (or “hypothesis”, as SPSS calls it) degrees of freedom for the random *Cond*×*SubjID* interaction, namely 23 (number of subjects minus 1). [\*NB\* With *unbalanced* numbers of observations per condition, SPSS would perform a numerical correction on these *dfs*, giving different (and ultimately more conservative) results depending on the severity of the imbalance.]

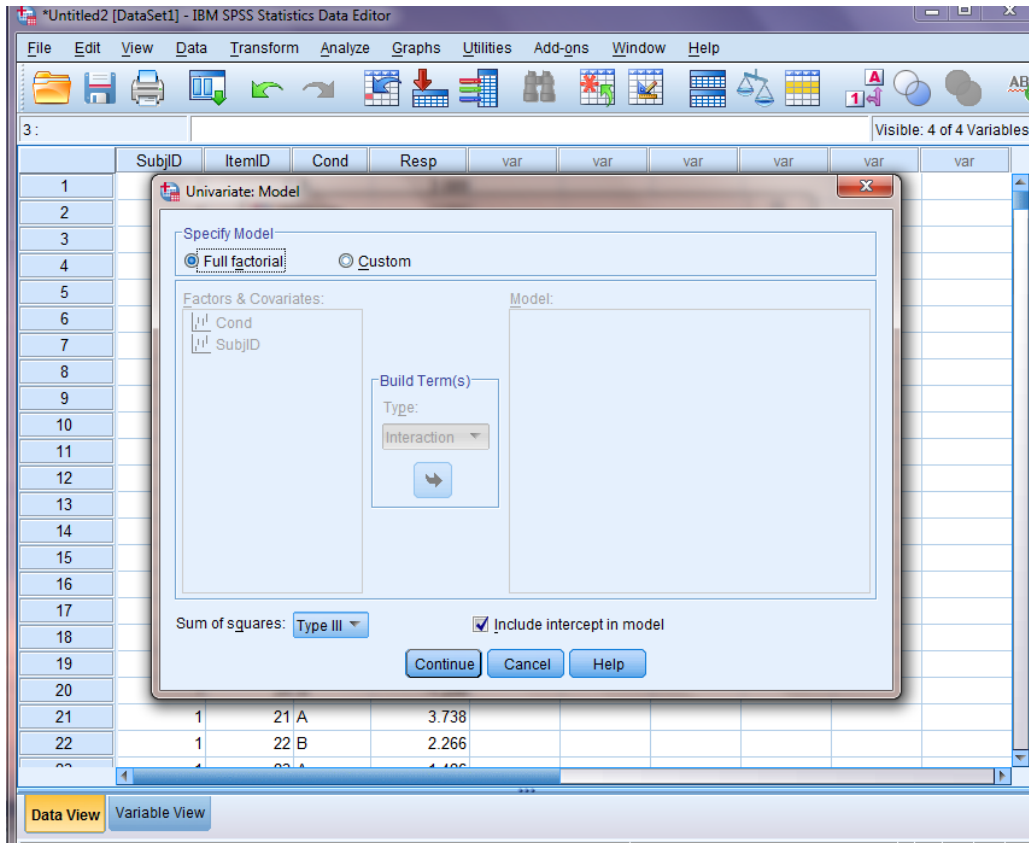
For our example data, the default mixed-model ANOVA (assuming a *maximal* by-subject random effects structure) suggests that the experimental treatment effect is not reliably generalizable across subjects:  $F(1, 23) = 2.654$ ;  $p = .117$ . [In case one might wonder: a repeated-measures ANOVA on by-subject aggregated data would yield  $F(1, 23) = 3.946$ ,  $p = .059$ .] Instead of the graphical user interface, one could also use the following syntax code to perform the standard full-factorial mixed-model ANOVA in SPSS:

```
UNIANOVA Resp BY Cond SubjID
  /RANDOM=SubjID
  /METHOD=SSTYPE(3)
  /INTERCEPT=INCLUDE
  /CRITERIA=ALPHA(0.05)
  /DESIGN=Cond SubjID Cond*SubjID.
```

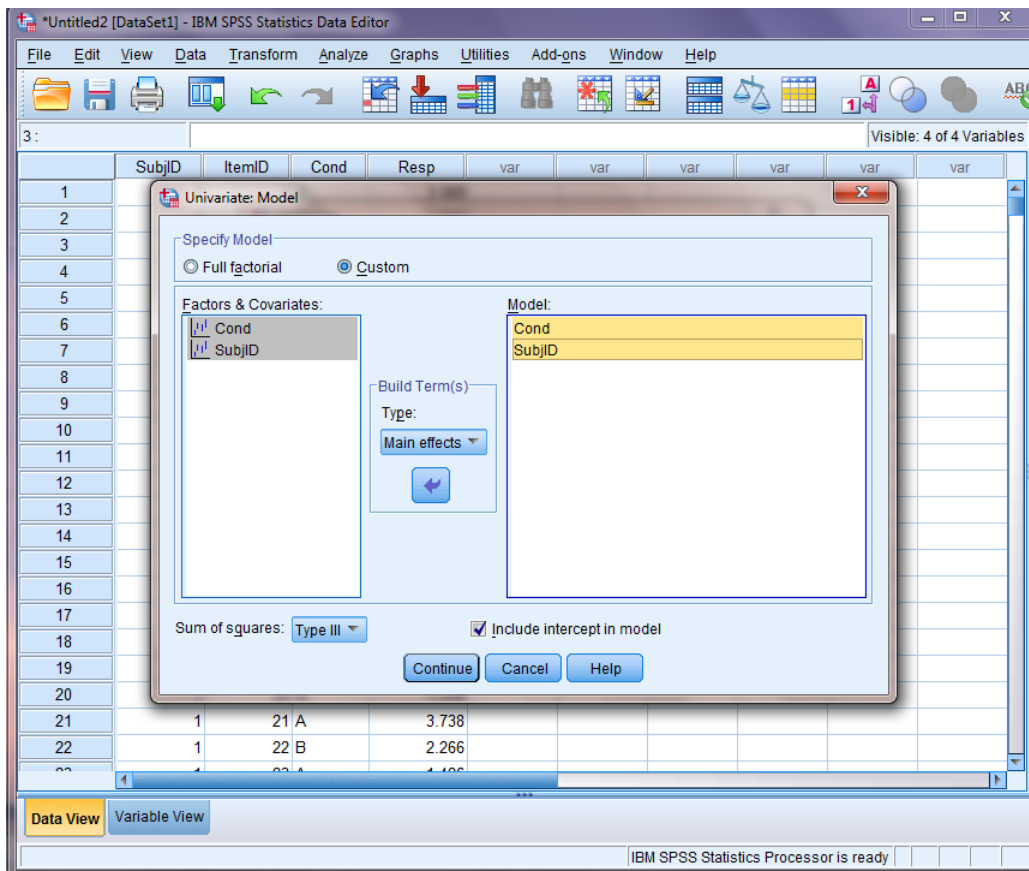
What if we wanted to apply a **simpler** model to our data, say, one that corresponds to a (by-subject) *intercept-only* model in LMEM? Although certainly not the norm for hypothesis testing in ANOVA, some researchers might consider such an intercept-only model if, for example, a significance test of the *Cond*×*SubjID* interaction would yield a non-significant result (cf. *model selection*). Others may just have picked up somewhere that “exclusion of the random interaction is ok if it’s not of theoretical interest” or that “intercept-only models are often used in LMEM” etc. (ignoring the pivotal

role that the random interaction/slope actually plays for capturing repeated-measures dependencies in both ANOVA and LMEM!).

Indeed, mixed-model ANOVA allows us to override the full-factorial default setting. To do this, open up the mixed-model ANOVA command window at *Analyze: General Linear Model: Univariate...* and specify the DV, the fixed and the random factor(s) as before. Now, note that in the top-right corner of the *Univariate...* command window, there is a button labelled “Model”. Clicking on that button opens up a new window that looks like this:



The toggle switches at the top are still in their *full factorial* default setting. Just switch over to *custom* and you will see a “Factors & Covariates” list on the left (containing the fixed factor *Cond* and the random factor *SubjID*) and a blank “Model” field on the right. To specify a (by-subject) random intercept-only effect structure, we need to implement a model that contains the main effects of *Cond* and *SubjID*, but no interaction between the two. Simply highlight the two factors in the “Factors & Covariates” list, then select *main effects* in the pull-down menu under “Build Term(s)” and click on the arrow-button to move the terms into the “Model” field. The command window now looks like this:



After clicking “Continue”, we are back in the *Univariate* main window, where we click on “OK” to run the analysis. Instead of the graphical user interface, we could also have used the following syntax to perform the intercept-only mixed-model ANOVA:

```
UNIANOVA Resp BY Cond SubjID
  /RANDOM=SubjID
  /METHOD=SSTYPE(3)
  /INTERCEPT=INCLUDE
  /CRITERIA=ALPHA(0.05)
  /DESIGN=Cond SubjID.
```

The results now look like this:

IBM SPSS Statistics Processor is ready | H: 244, W: 542 pt.

Pivot table is visible

Source		Type III Sum of Squares	df	Mean Square	F	Sig.
Intercept	Hypothesis	2467.264	1	2467.264	52.454	.000
	Error	1081.852	23	47.037 <sup>a</sup>		
Cond	Hypothesis	57.262	1	57.262	16.933	.000
	Error	1863.281	551	3.382 <sup>b</sup>		
SubjID	Hypothesis	1081.852	23	47.037	13.910	.000
	Error	1863.281	551	3.382 <sup>b</sup>		

a. MS(SubjID)  
b. MS(Error)

Expected Mean Squares<sup>a, b</sup>

As can be seen, the *Cond* fixed effect of interest changed from formerly “not significant” to “highly significant” in this random intercept-only analysis. Also, note what happened to the “error” term in the F-ratio for the fixed effect, whose degrees of freedom changed from formerly 23 (number of subjects minus 1) to 551 (number of trials minus: 1 *df* for the grand average intercept, 1 *df* for the *Cond* fixed effect, and 23 *dfs* for the *SubjID* random effect) in the intercept-only model. The latter is identical to the residual *by-trial* error which indicates that, just as in LMEM, the random intercept-only model essentially treats the by-condition measurements as subject- *independent* observations.

The “take home message” from this demonstration is that mixed-model ANOVA and LMEM are actually not all that different. Both distinguish between random and fixed effects and provide significance tests for each. Moreover, both allow for flexible adjustments of the model structure which is useful for *model building* applications (if not necessarily for *confirmatory hypothesis testing*). The only major difference (apart from relying on different algorithms for random effect estimation) is that LMEM, unlike ANOVA, allows for “crossing” of random effects, i.e. simultaneous consideration of subject- and item- related random effects.

### Additional simulation results

This section includes additional results from the simulations that were excluded from the main manuscript in the interest of brevity. Performance metrics for  $\alpha = .01, .10$  are in Tables 1–4; heatmaps for 12-items designs are given in Figures 1–4; and model selection performance for between-items designs is in Figure 5.

Table 1

*Between-items designs,  $\alpha = .01$*

	Type I 12	Type I 24	Power 12	Power 24	Power' 12	Power' 24
min- $F'$	.009	.009	.079	.154	.210	.328
LMEM, Maximal, $\chi^2_{LR}$	.017	.013	.118	.185	.223	.342
LMEM, Max-NRC, $\chi^2_{LR}$	.017	.013	.117	.185	.223	.343
$F_1 \times F_2$	.014	.019	.106	.222	.224	.337
LMEM, Max-NWI $\chi^2_{LR}$	.021	.015	.135	.202	.223	.342
LMEM, Maximal, $t$	.029	.017	.162	.214	.222	.343
LMEM, Max-NRC, $t$	.029	.017	.162	.214	.223	.343
LMEM, Max-NWI, $t$	.037	.021	.186	.234	.222	.342
LMEM, RI-only, $\chi^2_{LR}$	.032	.039	.164	.279	.216	.314
LMEM, RI-only, $t$	.055	.051	.228	.318	.217	.314
LMEM, RI-only, MCMC	.071	.103	.252	.444	.205	.309
$F_1$	.297	.217	.541	.571	.134	.212

Table 2

*Between-items designs,  $\alpha = .10$*

	Type I 12	Type I 24	Power 12	Power 24	Power' 12	Power' 24
min- $F'$	.092	.093	.311	.444	.210	.328
LMEM, Maximal, $\chi^2_{LR}$	.129	.113	.371	.478	.223	.342
LMEM, Max-NRC, $\chi^2_{LR}$	.128	.112	.370	.477	.223	.343
$F_1 \times F_2$	.120	.137	.355	.510	.224	.337
LMEM, Max-NWI $\chi^2_{LR}$	.142	.123	.391	.492	.223	.342
LMEM, Maximal, $t$	.143	.120	.394	.490	.222	.343
LMEM, Max-NRC, $t$	.143	.119	.393	.489	.223	.343
LMEM, Max-NWI, $t$	.160	.131	.415	.505	.222	.342
LMEM, RI-only, $\chi^2_{LR}$	.171	.177	.419	.548	.216	.314
LMEM, RI-only, $t$	.193	.189	.447	.563	.217	.314
LMEM, RI-only, MCMC	.255	.294	.524	.680	.205	.309
$F_1$	.497	.420	.732	.767	.134	.212



Table 3  
*Within-items designs,  $\alpha=.01$*

	Type I 12	Type I 24	Power 12	Power 24	Power' 12	Power' 24
$\min-F'$	.004	.005	.129	.266	.327	.512
LMEM, Maximal, $\chi^2_{LR}$	.013	.012	.240	.382	.433	.591
LMEM, Max-NRC, $\chi^2_{LR}$	.013	.012	.241	.381	.432	.593
$F_1 \times F_2$	.012	.018	.212	.410	.416	.578
LMEM, Max-NWI $\chi^2_{LR}$	.012	.012	.215	.363	.416	.579
LMEM, Maximal, $t$	.022	.016	.306	.425	.434	.592
LMEM, Max-NRC, $t$	.023	.016	.308	.425	.432	.593
LMEM, Max-NWI, $t$	.022	.018	.287	.416	.416	.580
$F_1$	.083	.059	.455	.538	.345	.506
LMEM, RI-only, $\chi^2_{LR}$	.317	.377	.787	.904	.379	.531
LMEM, RI-only, $t$	.320	.379	.789	.904	.379	.531
LMEM, RI-only, MCMC	.291	.361	.772	.899	.379	.531

Table 4  
*Within-items designs,  $\alpha=.10$*

	Type I 12	Type I 24	Power 12	Power 24	Power' 12	Power' 24
$\min-F'$	.061	.068	.463	.643	.327	.512
LMEM, Maximal, $\chi^2_{LR}$	.113	.108	.582	.717	.433	.591
LMEM, Max-NRC, $\chi^2_{LR}$	.113	.107	.582	.717	.432	.593
$F_1 \times F_2$	.112	.130	.568	.746	.416	.578
LMEM, Max-NWI $\chi^2_{LR}$	.107	.108	.561	.708	.416	.579
LMEM, Maximal, $t$	.126	.115	.603	.727	.434	.592
LMEM, Max-NRC, $t$	.127	.115	.604	.727	.432	.593
LMEM, Max-NWI, $t$	.121	.115	.586	.722	.416	.58
$F_1$	.251	.210	.725	.800	.345	.506
LMEM, RI-only, $\chi^2_{LR}$	.514	.567	.883	.949	.379	.531
LMEM, RI-only, $t$	.515	.568	.883	.949	.379	.531
LMEM, RI-only, MCMC	.491	.554	.876	.947	.379	.531

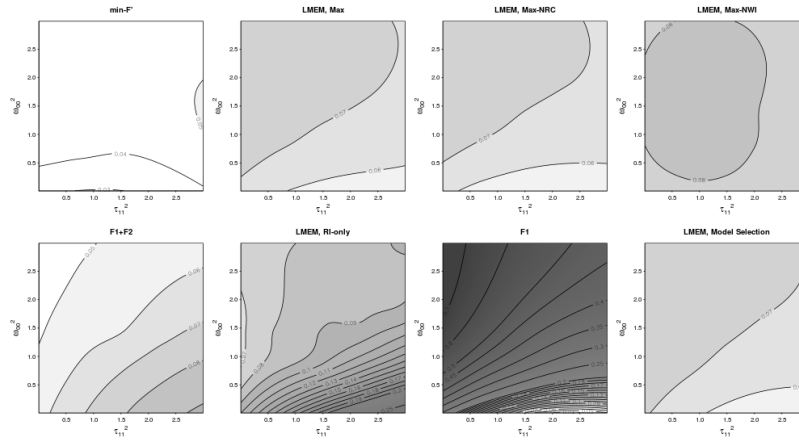


Figure 1. Between-items (12 items), Type I Error

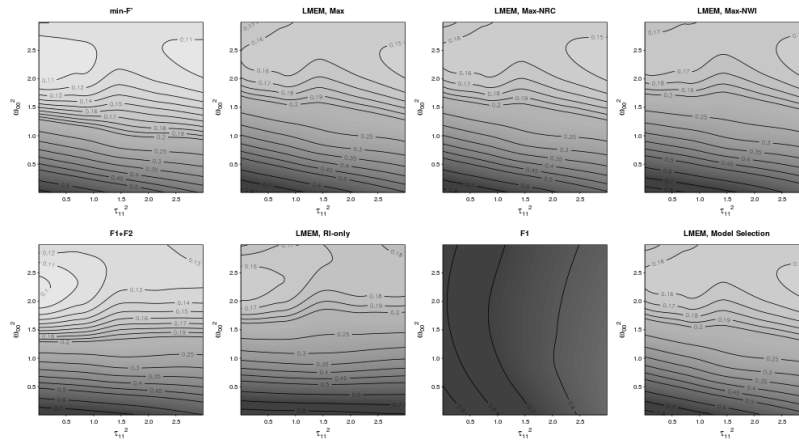


Figure 2. Between-items (12 items), Power

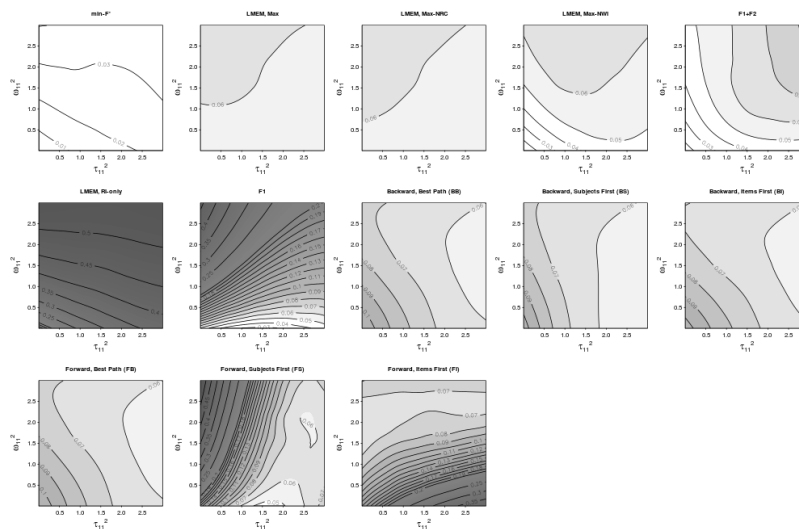


Figure 3. Within-items (12 items), Type I Error

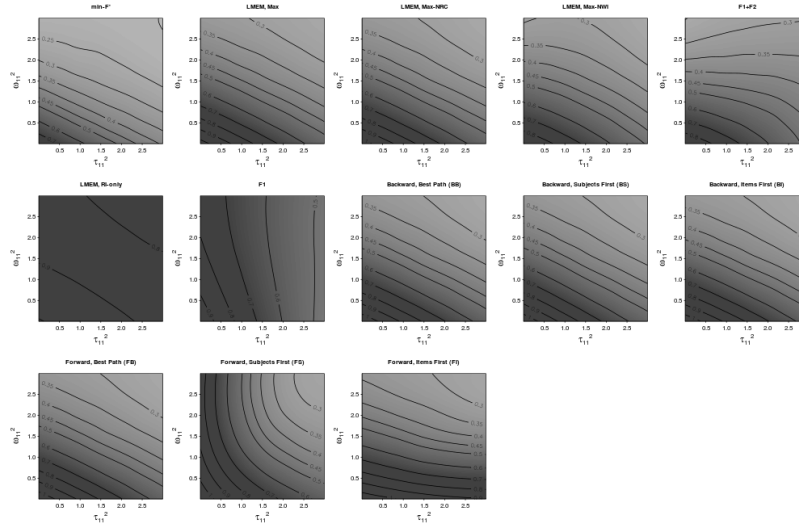


Figure 4. Within-items (12 items), Power

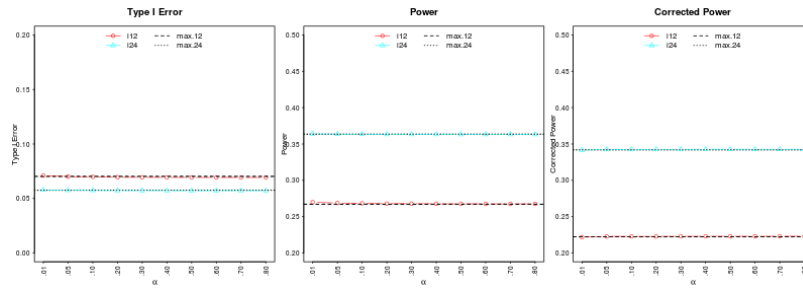


Figure 5. Model-selection performance, Between-items design

### Supplementary simulations and analysis

This section includes results from additional simulations or analyses that for reasons of readability, auxiliary interest, etc., were excluded from the final manuscript.

#### Effect of sample size

Fixed effects and their standard errors for LME models are computed based on maximum-likelihood estimates of the random effects covariance matrix. We believe that maximal-LME fixed-effects inferences remain slightly anti-conservative because maximum-likelihood estimation is biased to underestimate the size of random-effects variances, which deflates fixed-effects standard errors. If this is the case, then we should find that Type I error becomes increasingly nominal in maximal LME analyses as the number of subjects and/or items increases and certainty in the estimate of the random-effects covariance matrix correspondingly increases. In a set of informal simulations, we found that this was indeed the case.

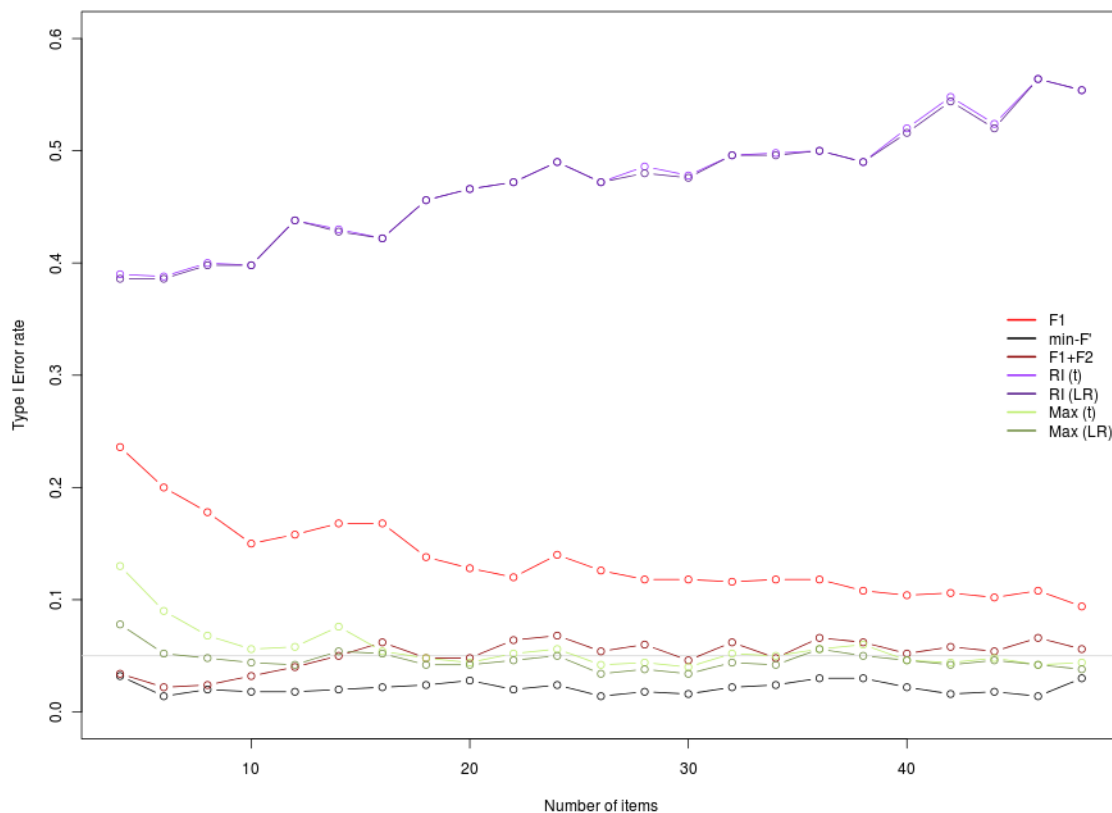


Figure 6. Type I error as a function of the number of items in a within/within experiment

We see signs of this in the results presented in the main paper, where Type I error rates are consistently higher for 12-item designs than for 24-item designs. The current analysis extends this line of

reasoning. In the above figure, Type I error rates are plotted as a function of the number of items. For maximal LME, anticonservativity disappears quickly as the number of items increases. This behavior contrasts with that of  $\min-F'$ , which is conservative across the board;  $F_1+F_2$ , which is conservative for few items but switches to being slightly anti-conservative as the number of items increases; and random-intercepts LME, which becomes more and more anticonservative with more items.

Of course, we do not always have the luxury of using a large number of items and subjects in our analyses. This anti-conservativity in the face of a limited number of clusters and corresponding uncertainty in the random effects is in fact exactly the kind of problem that the use of Bayesian inference and Markov-chain Monte Carlo on the fixed effects is intended to address (Baayen, Davidson, & Bates, 2008; Gelman & Hill, 2007). Unfortunately, these techniques are not yet available out of the box for random-slopes models in any LME implementation we are aware of, but we hope that they become readily available in the future and that they eliminate the anti-conservativity observed in the present simulations.<sup>1</sup>

### References

- Baayen, R. H., Davidson, D. J., and Bates, D. M. (2008). Mixed-effects modeling with crossed random effects for subjects and items. *Journal of Memory and Language*, 59, 390-412.
- Gelman, A. and Hill, J. (2006). *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge: Cambridge University Press.
- Plummer, Martyn (2003). JAGS: A Program for Analysis of Bayesian Graphical Models Using Gibbs Sampling, Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003), March 20–22, Vienna, Austria. ISSN 1609-395X.

---

<sup>1</sup>Random-slope models can be implemented and run from R on any platform using the software package JAGS (Plummer, 2003) and the authors have done so in some of their own analyses, but the process remains sufficiently time-consuming and error-prone that we do not roamed the practice at this point.

### Effect of non-homogeneous data loss

The simulations described in Barr, Levy, Scheepers, and Tily assume only a small amount of missing data (<5%) missing at random. For some research applications, this may be overly optimistic. We therefore conducted simulations to assess the effect of larger proportions of missing data (10-80%), with an uneven distribution of missing data over subjects (“=by subj=”), conditions (“=by cond=”), or condition-by-subject combinations (“=by subjcond=”). We assess the overall Type I error rates and power for the three different scenarios, and also report the percent change in Type I error rate and power from the original, optimistic (<5%, homogeneous data loss) scenario.

#### Overall results

Tables 5 and 6 report results for the main analyses that do not use model selection. Note that we are only considering those approaches that had reasonable Type I error rates for the original simulations:  $F_1 + F_2$ , min- $F'$ , and maximal LMEMs with  $p$ -values derived through model comparison. The percent changes in Type I error rate and power are relative to the original, smaller <5% data loss scenario. (The original values are in the column labeled “Main” in the tables; negative percent change indicates more conservative performance).

Table 5  
*Type I Error*

Method	Design	No.Item	Main	Subj	Cond	SubjCond	%Subj	%Cond	%SubjCond
F1+F2	wsbi	12	.06278	.06382	.05507	.06376	1.7	-12.3	1.6
min- $F'$	wsbi	12	.04446	.04134	.03520	.04140	-7.0	-2.8	-6.9
LMEM, Maximal (LR)	wsbi	12	.07026	.07010	.06928	.07018	-.2	-1.4	-.1
F1+F2	swsi	12	.05742	.04931	.04492	.04790	-14.1	-21.8	-16.6
min- $F'$	swsi	12	.02713	.02178	.01932	.02025	-19.7	-28.8	-25.4
LMEM, Maximal (LR)	swsi	12	.05885	.05904	.05853	.05795	.3	-.5	-1.5
F1+F2	wsbi	24	.07721	.07824	.06761	.08062	1.3	-12.4	4.4
min- $F'$	wsbi	24	.04456	.04307	.03693	.04418	-3.3	-17.1	-.9
LMEM, Maximal (LR)	wsbi	24	.05750	.05863	.05848	.05910	2.0	1.7	2.8
F1+F2	swsi	24	.07244	.06707	.05901	.06371	-7.4	-18.5	-12.1
min- $F'$	swsi	24	.03070	.02656	.02237	.02576	-13.5	-27.1	-16.1
LMEM, Maximal (LR)	swsi	24	.05594	.05574	.05676	.05651	-.4	1.5	1.0

#### Data-driven random effects

In this section, we compare the effects of non-homogeneous data loss on model selection techniques to its effects on maximal models. Each colored line in each graph represents a given model selection technique, with the horizontal axis indicating the specified  $\alpha$  level at which random slopes were tested. The two letter code in the legend for the within-items plot indicates the model selection technique, as shown in Table 7.

The numbers “12” and “24” preceding the codes in the legend indicate the number of items. See the main manuscript for more information about model selection algorithms. The dotted lines in the background reflect the performance of maximal models (one line for the 12-item datasets, and one for the 24-item datasets).

Table 6  
*Power*

Method	Design	No.Item	Main	Subj	Cond	SubjCond	%Subj	%Cond	%SubjCond
F1+F2	wsbi	12	.25183	.22962	.20708	.22683	-8.8	-17.8	-9.9
min-F'	wsbi	12	.20992	.17884	.15855	.17357	-14.8	-24.5	-17.3
LMEM, Maximal (LR)	wsbi	12	.26725	.2518	.24806	.25001	-5.8	-7.2	-6.5
F1+F2	swsi	12	.43998	.34422	.31782	.33477	-21.8	-27.8	-23.9
min-F'	swsi	12	.32682	.23220	.20998	.22451	-29.0	-35.8	-31.3
LMEM, Maximal (LR)	swsi	12	.46032	.41111	.39766	.40887	-1.7	-13.6	-11.2
F1+F2	wsbi	24	.40341	.38207	.35320	.37629	-5.3	-12.4	-6.7
min-F'	wsbi	24	.32811	.29519	.26761	.28915	-10.0	-18.4	-11.9
LMEM, Maximal (LR)	wsbi	24	.36361	.34842	.34541	.34801	-4.2	-5.0	-4.3
F1+F2	swsi	24	.64318	.55921	.52955	.54112	-13.1	-17.7	-15.9
min-F'	swsi	24	.51157	.41622	.38499	.40282	-18.6	-24.7	-21.3
LMEM, Maximal (LR)	swsi	24	.61037	.57046	.56220	.56699	-6.5	-7.9	-7.1

Table 7  
*Codes for Model Selection Algorithms*

Code	Model Selection Algorithm
BB	Backward, Best Path
BI	Backward, Test Item Slope before Subject Slope
BS	Backward, Test Subject Slope before Item Slope
FB	Forward, Best Path
FI	Forward, Test Item Slope before Subject Slope
FS	Forward, Test Subject Slope before Item Slope

The results for the data-driven approaches are depicted graphically in Figures 7–20.

### *Discussion*

The results from these simulations are very clear: maximal models are more robust against non-homogeneous data loss than separate  $F_1 + F_2$  analyses, min- $F'$  or model selection approaches. Unlike maximal models, all model selection approaches showed an increase in Type I error rate over all missing data scenarios, even approaching a 40% increase in some cases. The increase in Type I error rate for maximal models, in contrast, was negligible (<3%, at worst).

Maximal models showed less of a decrement in power relative to  $F_1 + F_2$  or min- $F'$ , with the decrement ranging between 1.7% at best (by subj, within-items design, 12 items), and 13.6% at worst (by cond, within-items design, 12 items). Model selection approaches, once corrected for anticonservativity, never exceeded the power of maximal models.

The poor performance of model selection approaches in terms of Type I error may be explained by the fact that the missing data make it more difficult to detect the existence of random slope variance, increasing the likelihood that the slope will be excluded from the model.

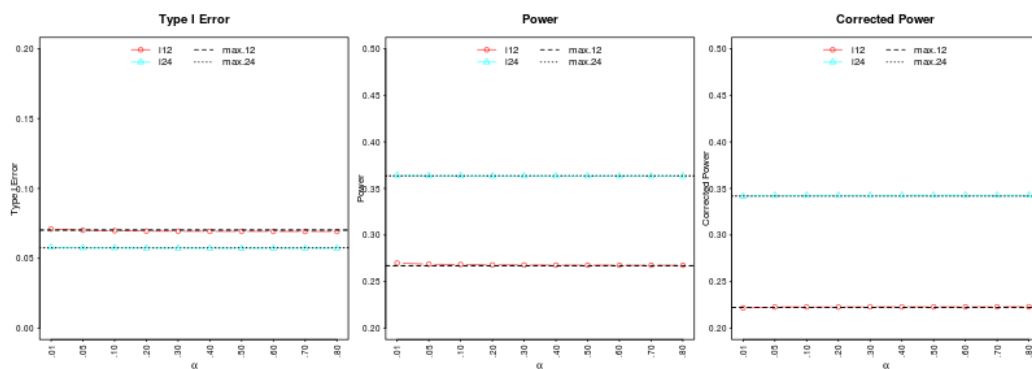


Figure 7. Between-items, Main

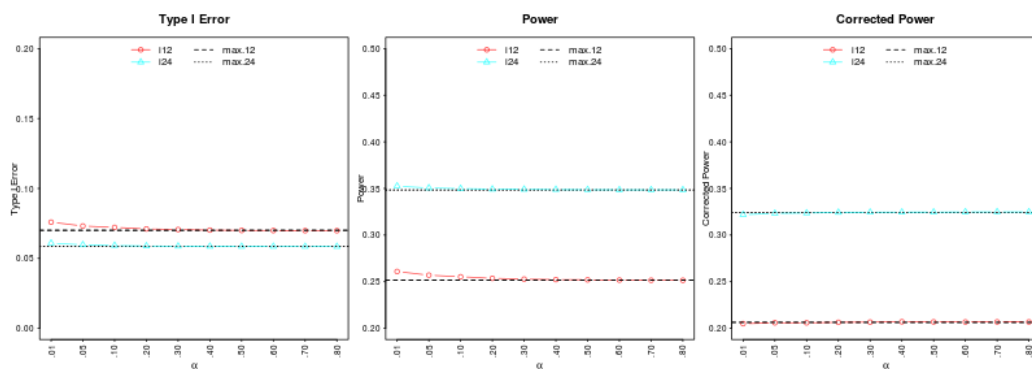


Figure 8. Between-items, By Subject

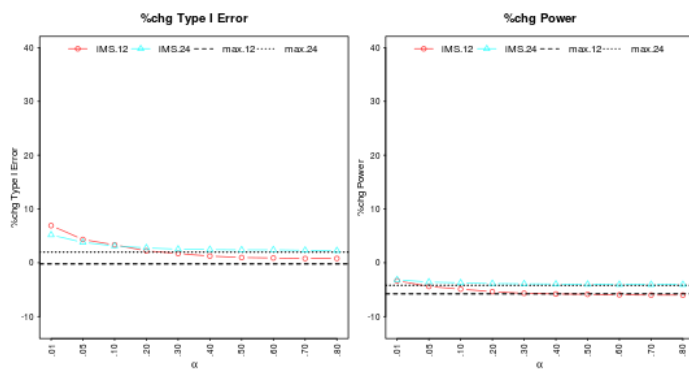


Figure 9. Between-items, By Subject: Percent change



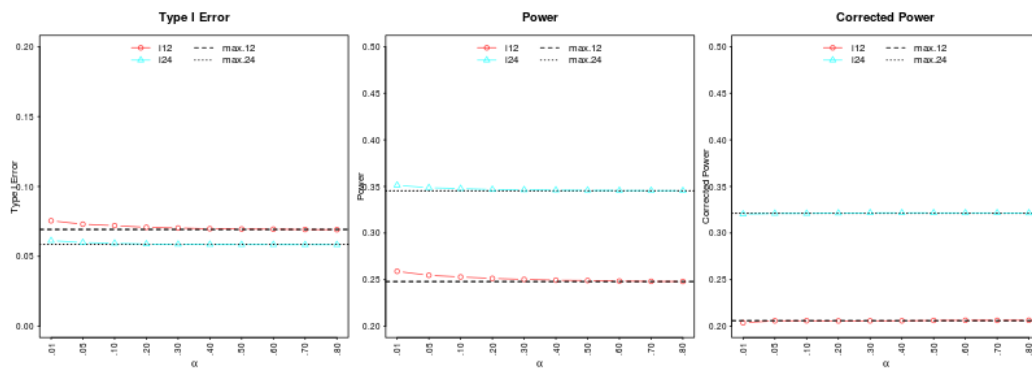


Figure 10. Between-items, By Condition

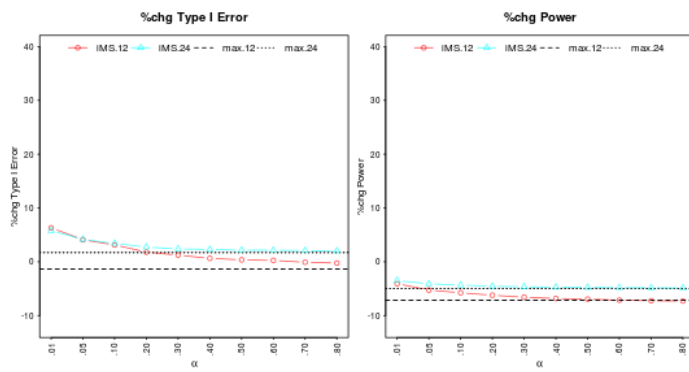


Figure 11. Between-items, By Condition, Percent change

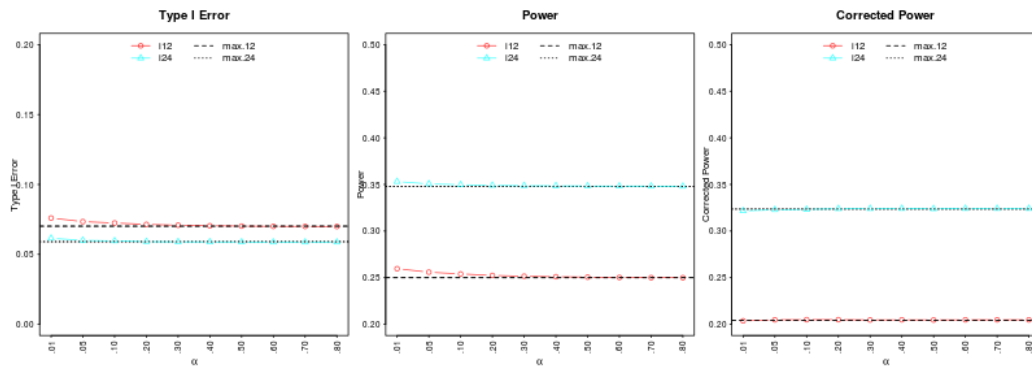


Figure 12. Between-items, By Subject/Condition

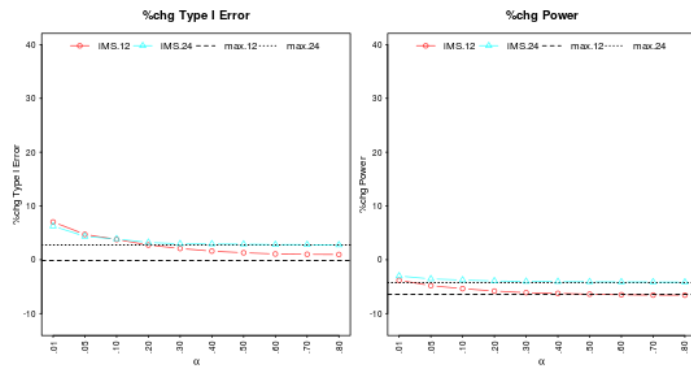


Figure 13. Between-items, By Subject/Condition, Percent change

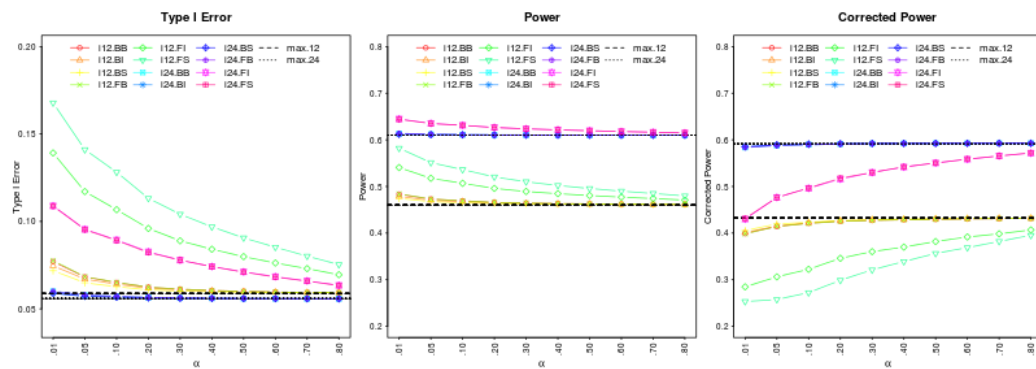


Figure 14. Within-items, Main

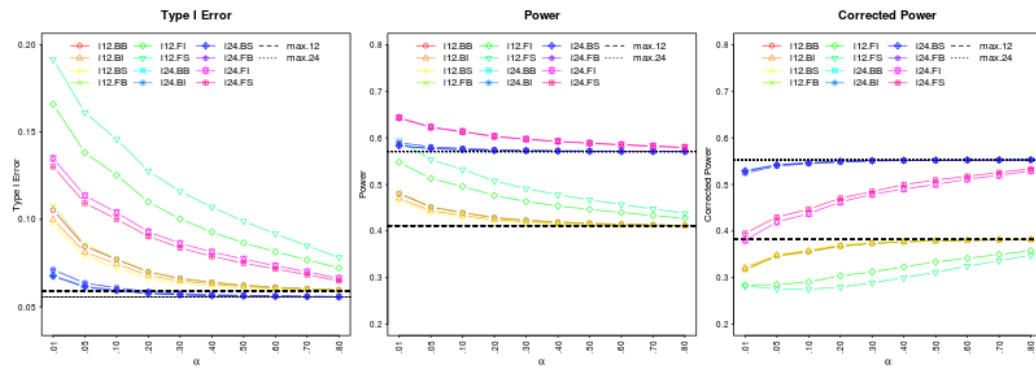


Figure 15. Within-items, By Subject

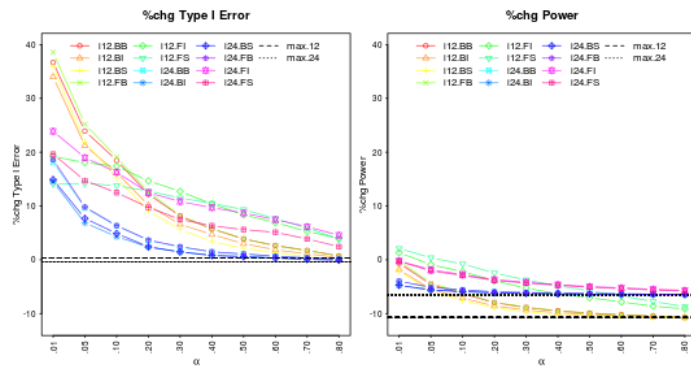


Figure 16. Within-items, By Subject, Percent change

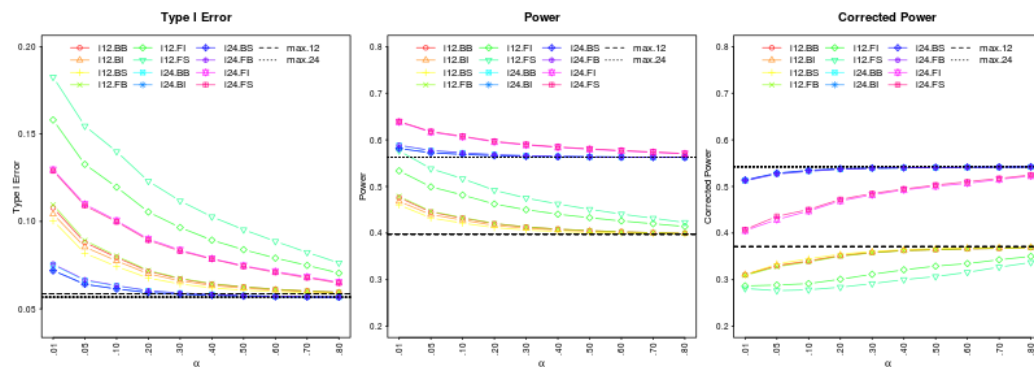


Figure 17. Within-items, By Condition

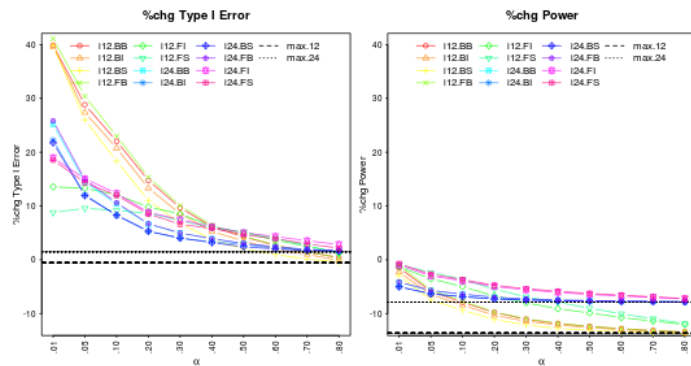


Figure 18. Within-items, By Condition, Percent change

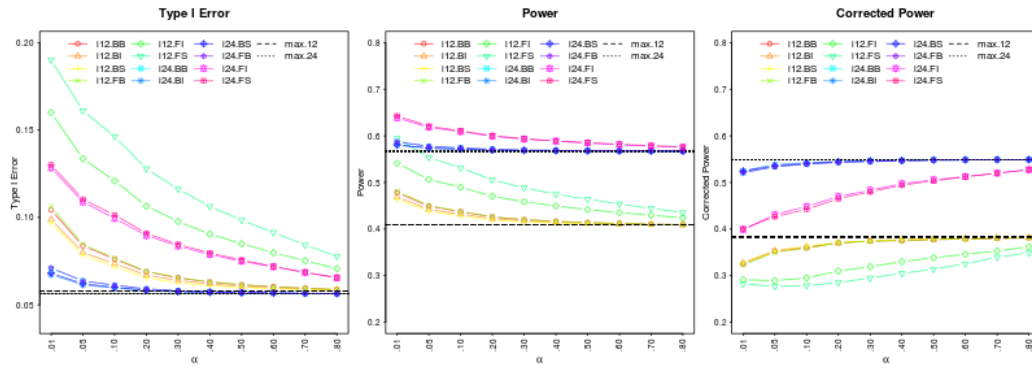


Figure 19. Within-items, By Subject/Condition

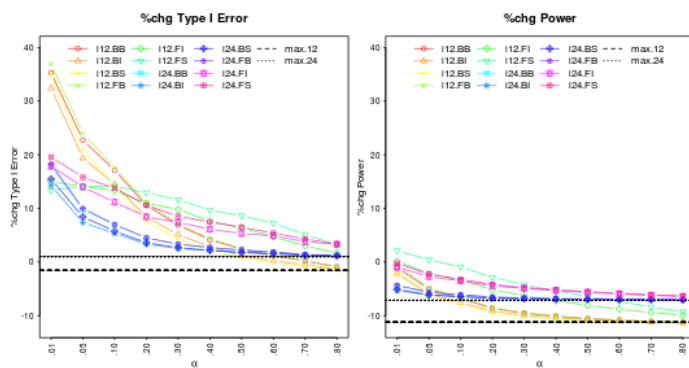


Figure 20. Within-items, By Subject/Condition, Percent change

- R scripts

*R Script for between-items designs*

```
library(simgen)

mf <- modSpace(TRUE)[2:1]
param.mx <- createParamMx(nexp=100000) # or load(file="param.RData")
nrns <- nrow(param.mx)

param.mx[, "eff"] <- 0

mcRun("fitstepwise", "newmselect/fitstepwise.h0.wsbi.12.MS.RData",
      "mkDf",
      mcr.constant=list(pMin=.1, pMax=.8),
      mcr.varying=param.mx[1:nruns,], mf=mf, nitem=12, wsbi=TRUE,
      missMeth="random")

mcRun("fitstepwise", "newmselect/fitstepwise.h0.wsbi.24.MS.RData",
      "mkDf",
      mcr.constant=list(pMin=.1, pMax=.8),
      mcr.varying=param.mx[1:nruns,], mf=mf, nitem=24, wsbi=TRUE,
      missMeth="random")

mcRun("fitstepwise", "missbysubj/fitstepwise.h0.wsbi.12.MS.RData",
      "mkDf",
      mcr.constant=list(pMin=.1, pMax=.8),
      mcr.varying=param.mx[1:nruns,], mf=mf, nitem=12, wsbi=TRUE,
      missMeth="bysubj")

mcRun("fitstepwise", "missbysubj/fitstepwise.h0.wsbi.24.MS.RData",
      "mkDf",
      mcr.constant=list(pMin=.1, pMax=.8),
      mcr.varying=param.mx[1:nruns,], mf=mf, nitem=24, wsbi=TRUE,
      missMeth="bysubj")

mcRun("fitstepwise", "missbycond/fitstepwise.h0.wsbi.12.MS.RData",
      "mkDf",
      mcr.constant=list(pMin=.1, pMax=.8),
      mcr.varying=param.mx[1:nruns,], mf=mf, nitem=12, wsbi=TRUE,
      missMeth="bycond")

mcRun("fitstepwise", "missbycond/fitstepwise.h0.wsbi.24.MS.RData",
      "mkDf",
      mcr.constant=list(pMin=.1, pMax=.8),
      mcr.varying=param.mx[1:nruns,], mf=mf, nitem=24, wsbi=TRUE,
      missMeth="bycond")

mcRun("fitstepwise", "missbysubjcond/fitstepwise.h0.wsbi.12.MS.RData",
      "mkDf",
      mcr.constant=list(pMin=.1, pMax=.8),
      mcr.varying=param.mx[1:nruns,], mf=mf, nitem=12, wsbi=TRUE,
      missMeth="bysubjcond")

mcRun("fitstepwise", "missbysubjcond/fitstepwise.h0.wsbi.24.MS.RData",
      "mkDf",
      mcr.constant=list(pMin=.1, pMax=.8),
      mcr.varying=param.mx[1:nruns,], mf=mf, nitem=24, wsbi=TRUE,
      missMeth="bysubjcond")

param.mx[, "eff"] <- .8

mcRun("fitstepwise", "newmselect/fitstepwise.h1.wsbi.12.MS.RData",
      "mkDf",
      mcr.constant=list(pMin=.1, pMax=.8),
      mcr.varying=param.mx[1:nruns,], mf=mf, nitem=12, wsbi=TRUE,
```

```

missMeth="random")

mcRun("fitstepwise", "newmselect/fitstepwise.h1.wsbi.24.MS.RData",
      "mkDf",
      mcr.constant=list(pMin=.1,pMax=.8),
      mcr.varying=param.mx[1:nruns,], mf=mf, nitem=24, wsbi=TRUE,
      missMeth="random")

mcRun("fitstepwise", "missbysubj/fitstepwise.h1.wsbi.12.MS.RData",
      "mkDf",
      mcr.constant=list(pMin=.1,pMax=.8),
      mcr.varying=param.mx[1:nruns,], mf=mf, nitem=12, wsbi=TRUE,
      missMeth="bysubj")

mcRun("fitstepwise", "missbysubj/fitstepwise.h1.wsbi.24.MS.RData",
      "mkDf",
      mcr.constant=list(pMin=.1,pMax=.8),
      mcr.varying=param.mx[1:nruns,], mf=mf, nitem=24, wsbi=TRUE,
      missMeth="bysubj")

mcRun("fitstepwise", "missbycond/fitstepwise.h1.wsbi.12.MS.RData",
      "mkDf",
      mcr.constant=list(pMin=.1,pMax=.8),
      mcr.varying=param.mx[1:nruns,], mf=mf, nitem=12, wsbi=TRUE,
      missMeth="bycond")

mcRun("fitstepwise", "missbycond/fitstepwise.h1.wsbi.24.MS.RData",
      "mkDf",
      mcr.constant=list(pMin=.1,pMax=.8),
      mcr.varying=param.mx[1:nruns,], mf=mf, nitem=24, wsbi=TRUE,
      missMeth="bycond")

mcRun("fitstepwise", "missbysubjcond/fitstepwise.h1.wsbi.12.MS.RData",
      "mkDf",
      mcr.constant=list(pMin=.1,pMax=.8),
      mcr.varying=param.mx[1:nruns,], mf=mf, nitem=12, wsbi=TRUE,
      missMeth="bysubjcond")

mcRun("fitstepwise", "missbysubjcond/fitstepwise.h1.wsbi.24.MS.RData",
      "mkDf",
      mcr.constant=list(pMin=.1,pMax=.8),
      mcr.varying=param.mx[1:nruns,], mf=mf, nitem=24, wsbi=TRUE,
      missMeth="bysubjcond")

```

### *R script for within-items designs*

```

library(simgen)

mf <- modSpace(FALSE)
mf.sfirst <- c(mf[3], mf[[2]][1], mf[1])
mf.ifirst <- c(mf[3], mf[[2]][2], mf[1])

param.mx <- createParamMx(nexp=100000) # or load(file="param.RData")
nruns <- nrow(param.mx)

param.mx[, "eff"] <- 0

# main (missing data < 5%)
mcRun("fitstepwise", "newmselect/fitstepwise.h0.wswi.12.FSBI.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=12, wsbi=FALSE, missMeth="random", mf=mf.sfirst)

mcRun("fitstepwise", "newmselect/fitstepwise.h0.wswi.24.FSBI.csv",

```

```

      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="random", mf=mf.sfirst)

mcRun("fitstepwise", "newmselect/fitstepwise.h0.wswi.12.FIBS.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=12, wsbi=FALSE, missMeth="random", mf=mf.ifirst)

mcRun("fitstepwise", "newmselect/fitstepwise.h0.wswi.24.FIBS.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="random", mf=mf.ifirst)

mcRun("fitstepwise.bestpath", "newmselect/fitstepwise.h0.wswi.24.FB.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="random", forward=TRUE)

mcRun("fitstepwise.bestpath", "newmselect/fitstepwise.h0.wswi.24.BB.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="random", forward=FALSE)

# missing by subject
mcRun("fitstepwise", "missbysubj/fitstepwise.h0.wswi.12.FSBI.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=12, wsbi=FALSE, missMeth="bysubj", mf=mf.sfirst)

mcRun("fitstepwise", "missbysubj/fitstepwise.h0.wswi.24.FSBI.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="bysubj", mf=mf.sfirst)

mcRun("fitstepwise", "missbysubj/fitstepwise.h0.wswi.12.FIBS.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=12, wsbi=FALSE, missMeth="bysubj", mf=mf.ifirst)

mcRun("fitstepwise", "missbysubj/fitstepwise.h0.wswi.24.FIBS.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="bysubj", mf=mf.ifirst)

mcRun("fitstepwise.bestpath", "missbysubj/fitstepwise.h0.wswi.24.FB.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="bysubj", forward=TRUE)

mcRun("fitstepwise.bestpath", "missbysubj/fitstepwise.h0.wswi.24.BB.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="bysubj", forward=FALSE)

# missing by condition
mcRun("fitstepwise", "missbycond/fitstepwise.h0.wswi.12.FSBI.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=12, wsbi=FALSE, missMeth="bycond", mf=mf.sfirst)

mcRun("fitstepwise", "missbycond/fitstepwise.h0.wswi.24.FSBI.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="bycond", mf=mf.sfirst)

mcRun("fitstepwise", "missbycond/fitstepwise.h0.wswi.12.FIBS.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=12, wsbi=FALSE, missMeth="bycond", mf=mf.ifirst)

mcRun("fitstepwise", "missbycond/fitstepwise.h0.wswi.24.FIBS.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="bycond", mf=mf.ifirst)

mcRun("fitstepwise.bestpath", "missbycond/fitstepwise.h0.wswi.24.FB.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="bycond", forward=TRUE)

```

```

mcRun("fitstepwise.bestpath", "missbycond/fitstepwise.h0.wswi.24.BB.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="bycond", forward=FALSE)

# missing by subjcond
mcRun("fitstepwise", "missbysubjcond/fitstepwise.h0.wswi.12.FSBI.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=12, wsbi=FALSE, missMeth="bysubjcond", mf=mf.sfirst)

mcRun("fitstepwise", "missbysubjcond/fitstepwise.h0.wswi.24.FSBI.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="bysubjcond", mf=mf.sfirst)

mcRun("fitstepwise", "missbysubjcond/fitstepwise.h0.wswi.12.FIBS.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=12, wsbi=FALSE, missMeth="bysubjcond", mf=mf.ifirst)

mcRun("fitstepwise", "missbysubjcond/fitstepwise.h0.wswi.24.FIBS.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="bysubjcond", mf=mf.ifirst)

mcRun("fitstepwise.bestpath", "missbysubjcond/fitstepwise.h0.wswi.24.FB.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="bysubjcond", forward=TRUE)

mcRun("fitstepwise.bestpath", "missbysubjcond/fitstepwise.h0.wswi.24.BB.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="bysubjcond", forward=FALSE)

param.mx[, "eff"] <- .8

# main (missing data < 5%)
mcRun("fitstepwise", "newmselect/fitstepwise.h1.wswi.12.FSBI.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=12, wsbi=FALSE, missMeth="random", mf=mf.sfirst)

mcRun("fitstepwise", "newmselect/fitstepwise.h1.wswi.24.FSBI.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="random", mf=mf.sfirst)

mcRun("fitstepwise", "newmselect/fitstepwise.h1.wswi.12.FIBS.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=12, wsbi=FALSE, missMeth="random", mf=mf.ifirst)

mcRun("fitstepwise", "newmselect/fitstepwise.h1.wswi.24.FIBS.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="random", mf=mf.ifirst)

mcRun("fitstepwise.bestpath", "newmselect/fitstepwise.h1.wswi.24.FB.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="random", forward=TRUE)

mcRun("fitstepwise.bestpath", "newmselect/fitstepwise.h1.wswi.24.BB.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="random", forward=FALSE)

# missing by subject
mcRun("fitstepwise", "missbysubj/fitstepwise.h1.wswi.12.FSBI.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=12, wsbi=FALSE, missMeth="bysubj", mf=mf.sfirst)

mcRun("fitstepwise", "missbysubj/fitstepwise.h1.wswi.24.FSBI.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],

```



```

      nitem=24, wsbi=FALSE, missMeth="bysubj", mf=mf.sfirst)

mcRun("fitstepwise", "missbysubj/fitstepwise.h1.wswi.12.FIBS.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=12, wsbi=FALSE, missMeth="bysubj", mf=mf.ifirst)

mcRun("fitstepwise", "missbysubj/fitstepwise.h1.wswi.24.FIBS.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="bysubj", mf=mf.ifirst)

mcRun("fitstepwise.bestpath", "missbysubj/fitstepwise.h1.wswi.24.FB.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="bysubj", forward=TRUE)

mcRun("fitstepwise.bestpath", "missbysubj/fitstepwise.h1.wswi.24.BB.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="bysubj", forward=FALSE)

# missing by condition
mcRun("fitstepwise", "missbycond/fitstepwise.h1.wswi.12.FSBI.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=12, wsbi=FALSE, missMeth="bycond", mf=mf.sfirst)

mcRun("fitstepwise", "missbycond/fitstepwise.h1.wswi.24.FSBI.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="bycond", mf=mf.sfirst)

mcRun("fitstepwise", "missbycond/fitstepwise.h1.wswi.12.FIBS.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=12, wsbi=FALSE, missMeth="bycond", mf=mf.ifirst)

mcRun("fitstepwise", "missbycond/fitstepwise.h1.wswi.24.FIBS.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="bycond", mf=mf.ifirst)

mcRun("fitstepwise.bestpath", "missbycond/fitstepwise.h1.wswi.24.FB.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="bycond", forward=TRUE)

mcRun("fitstepwise.bestpath", "missbycond/fitstepwise.h1.wswi.24.BB.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="bycond", forward=FALSE)

# missing by subjcond
mcRun("fitstepwise", "missbysubjcond/fitstepwise.h1.wswi.12.FSBI.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=12, wsbi=FALSE, missMeth="bysubjcond", mf=mf.sfirst)

mcRun("fitstepwise", "missbysubjcond/fitstepwise.h1.wswi.24.FSBI.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="bysubjcond", mf=mf.sfirst)

mcRun("fitstepwise", "missbysubjcond/fitstepwise.h1.wswi.12.FIBS.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=12, wsbi=FALSE, missMeth="bysubjcond", mf=mf.ifirst)

mcRun("fitstepwise", "missbysubjcond/fitstepwise.h1.wswi.24.FIBS.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="bysubjcond", mf=mf.ifirst)

mcRun("fitstepwise.bestpath", "missbysubjcond/fitstepwise.h1.wswi.24.FB.csv",
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],
      nitem=24, wsbi=FALSE, missMeth="bysubjcond", forward=TRUE)

```

```
mCRun("fitstepwise.bestpath", "missbysubjcond/fitstepwise.h1.wswi.24.BB.csv",  
      "mkDf", mcr.constant=list(pMin=.1,pMax=.8), mcr.varying=param.mx[1:nruns,],  
      nitem=24, wsbi=FALSE, missMeth="bysubjcond", forward=FALSE)
```

### Models without random intercepts or random correlations

A conventional misconception is that random intercepts are more important for protecting against Type I error than random slopes. While random intercepts are clearly important when item effects are confounded with the treatment variable (i.e., between-items designs), it is less clear what the consequences would be for removing them in within-unit designs. This could have important implications for how one goes about simplifying the random effects structure when a model does not converge. Thus, the current simulations only include random intercepts when the comparison is between units. We used two different random effects parameterizations for between-items designs, and one for within-items designs.

- between-items, parameterization 1:  $y \sim c + (0 + c | \text{Subject}) + (1 | \text{Item})$
- between-items, parameterization 2:  $y \sim c + (0 + c | \text{Subject}) + (0 + c | \text{Item})$
- within-items design:  $y \sim c + (0 + c | \text{Subject}) + (0 + c | \text{Item})$

Results are reported in Tables 8–11 and Figures 23–26, with  $p$  values obtained using either the  $t$ -as- $z$  heuristic, or a likelihood-ratio test ( $\chi^2$ ). Results from the maximal model (“max”) are included in all tables as a benchmark. We considered two different coding schemes for the experimental treatment variable, an ANOVA-style “deviation” coding scheme (-.5, .5), and a treatment coding scheme (0, 1).

Figure 21 illustrates the differences in expressivity of models with and without correlation between random intercept and random slope. Although the maximal model (Model 3 in the manuscript) is the more general model for “within” designs, the model with no random-correlations is also of interest because it has fewer parameters and may thus be easier to fit stably. However, the performance of the model depends on the coding scheme used to express one’s experimental treatment, a characteristic which is absent from the other models we consider in the paper. Figure 22 illustrates this dependence by showing that the distributions of random effects with no correlation under deviation coding have a non-zero correlation under treatment coding. As shown in our simulations below, this dependence on coding can affect the degree of anti-conservativity of  $p$ -values in such models.

Like the no-random-correlation model, the no-within-unit-intercepts model will express different model families depending on the coding used for experimental condition: here, treatment coding would imply that subject-specific means differ only for condition B, not condition A, whereas deviation coding would imply that subject-specific means differ equally and in opposite directions from the grand mean in the two conditions.

Table 8

*Within-items and between-items parameterization 1, Type I Error*

	Deviation $t$ as $z$	Treatment $t$ as $z$	Maximal $t$ as $z$	Deviation $\chi^2$	Treatment $\chi^2$	Maximal $\chi^2$
wsbi.12	.09973	.08752	.08619	.08085	.07135	.07026
wsbi.24	.07321	.06553	.06485	.06489	.05856	.05750
wswi.12	.07007	.06625	.07185	.05601	.05811	.05885
wswi.24	.06366	.05892	.06265	.05536	.05405	.05594

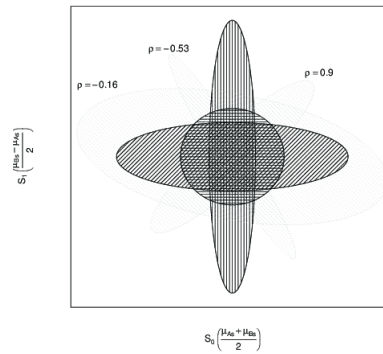


Figure 21. Joint distributions allowed for random intercepts and random slopes for models with no-within unit intercepts, and models with no random correlations, in the deviation-coding representation. Here the  $S_0$  axis is the subject's overall deviation from the grand mean, and the  $S_1$  axis is half the subject-specific discrepancy between experimental conditions (above and beyond the population-average discrepancy). Both models allow distributions plotted in black, where subject's overall reaction time is uncorrelated with subject's idiosyncratic sensitivity to word type; only a maximal model allows distributions plotted in gray, where the two are correlated.

Table 9

Within-items and between-items parameterization 1, Power

	Deviation $t$ as $z$	Treatment $t$ as $z$	Maximal $t$ as $z$	Deviation $\chi^2$	Treatment $\chi^2$	Maximal $\chi^2$
wsbi.12	.32280	.29611	.30022	.28825	.26401	.26725
wsbi.24	.40103	.36817	.38249	.38010	.35039	.36361
wswi.12	.47739	.40137	.49641	.43655	.37146	.46032
wswi.24	.61984	.53852	.62941	.59571	.51949	.61037

### Independent random intercepts and random slopes

In a maximal model, random intercepts and slopes are considered to be drawn from a multivariate distribution, the intercept and slope variances are estimated, as well as their covariance. On some occasions where a model does not converge, the question arises as to what the consequences would be of removing the random correlation parameters.

To investigate this, we fit models with the following random effects structure (where  $c$  indexes experimental condition):

- between-items design:  $y \sim c + (1 | \text{Subject}) + (0 + c | \text{Subject}) + (1 | \text{Item})$
- within-items design:  $y \sim c + (1 | \text{Subject}) + (0 + c | \text{Subject}) + (1 | \text{Item}) + (0 + c | \text{Item})$

### Discussion

Models excluding random intercepts for within-unit manipulations showed good control of the Type I error rate, although they showed decreasing power versus a maximal model as slopes be-

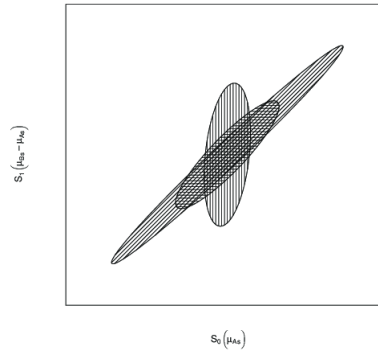


Figure 22. Joint distributions allowed for random intercepts and random slopes for models with no-within unit intercepts, and models with no random correlations, in the treatment-coding representation. Here the  $S_0$  axis is the subject's condition-A deviation from the condition-A grand mean, and the  $S_1$  axis is the subject-specific discrepancy between experimental conditions (above and beyond the population-average discrepancy). The distributions in black in Figure 21 cannot be fitted using the no-random-correlations model with a treatment coding, since in each case there is a correlation between random slope and intercept in the treatment-coding representation.

Table 10

Between-items parameterization 2, Type I Error

	Deviation $t$ as $z$	Treatment $t$ as $z$	Maximal $t$ as $z$	Deviation $\chi^2$	Treatment $\chi^2$	Maximal $\chi^2$
wsbi.12	.09958	.17314	.08619	.08102	.13980	.07026
wsbi.24	.07320	.13312	.06485	.06508	.11797	.05750

came small relative to the by-unit intercept variance and residual variance. Models excluding random correlations exhibited performance that was virtually indistinguishable from maximal models. This suggests that when one is facing decisions about how to simplify the random effects structure for a nonconverging model, then removing correlations will not harm the overall performance of the model. It may also be worth considering removing random intercepts, especially if the estimates from the nonconverged model suggest they may be associated with relatively small amounts of variability. These analyses indicate that, at least for within-unit designs, the conventional view is clearly wrong: random slopes are far more important than random intercepts.

Figure 23. Heatmaps, Within-items and Between-items parameterization 1, Type I Error

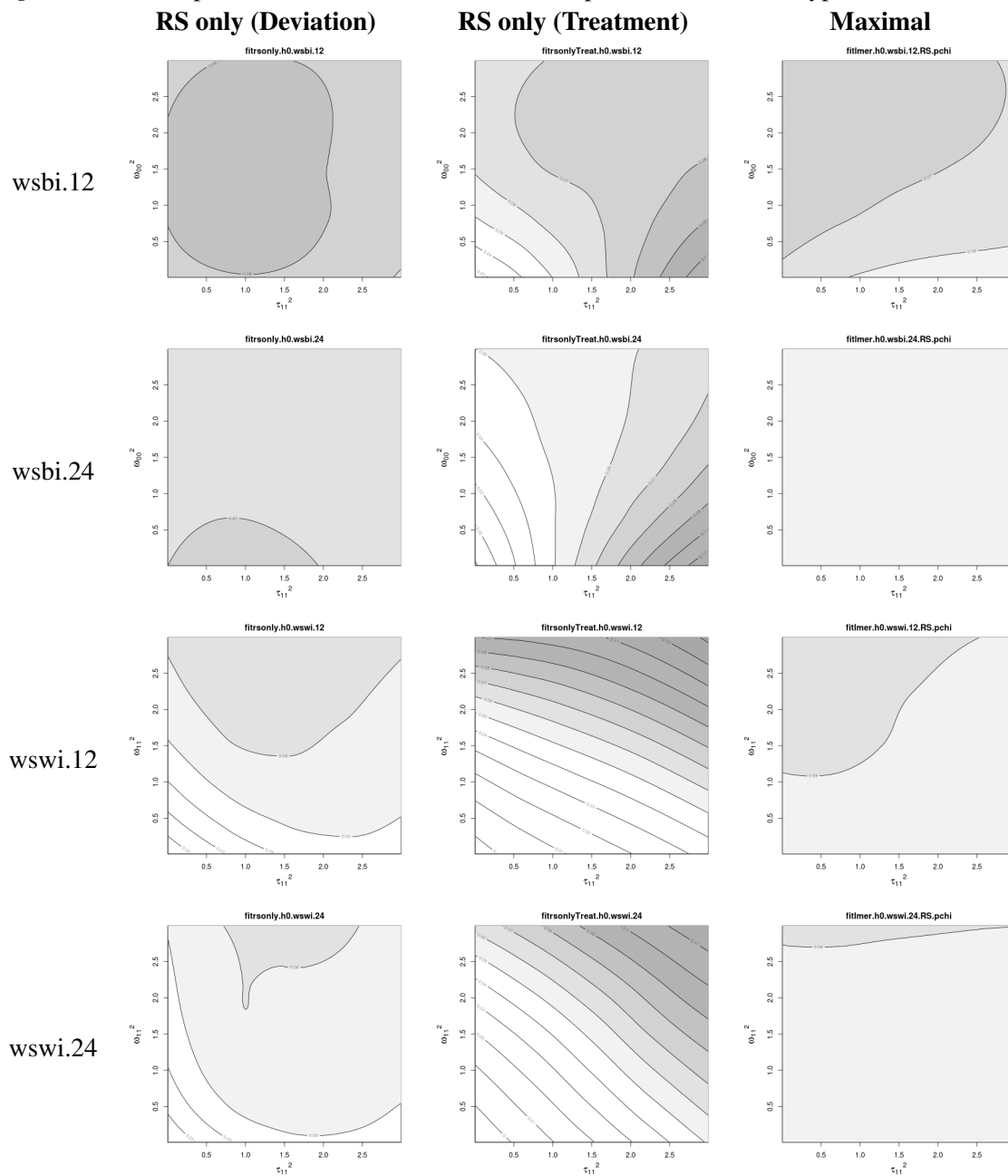


Figure 24. Heatmaps, Within-items and Between-items parameterization 1, Power

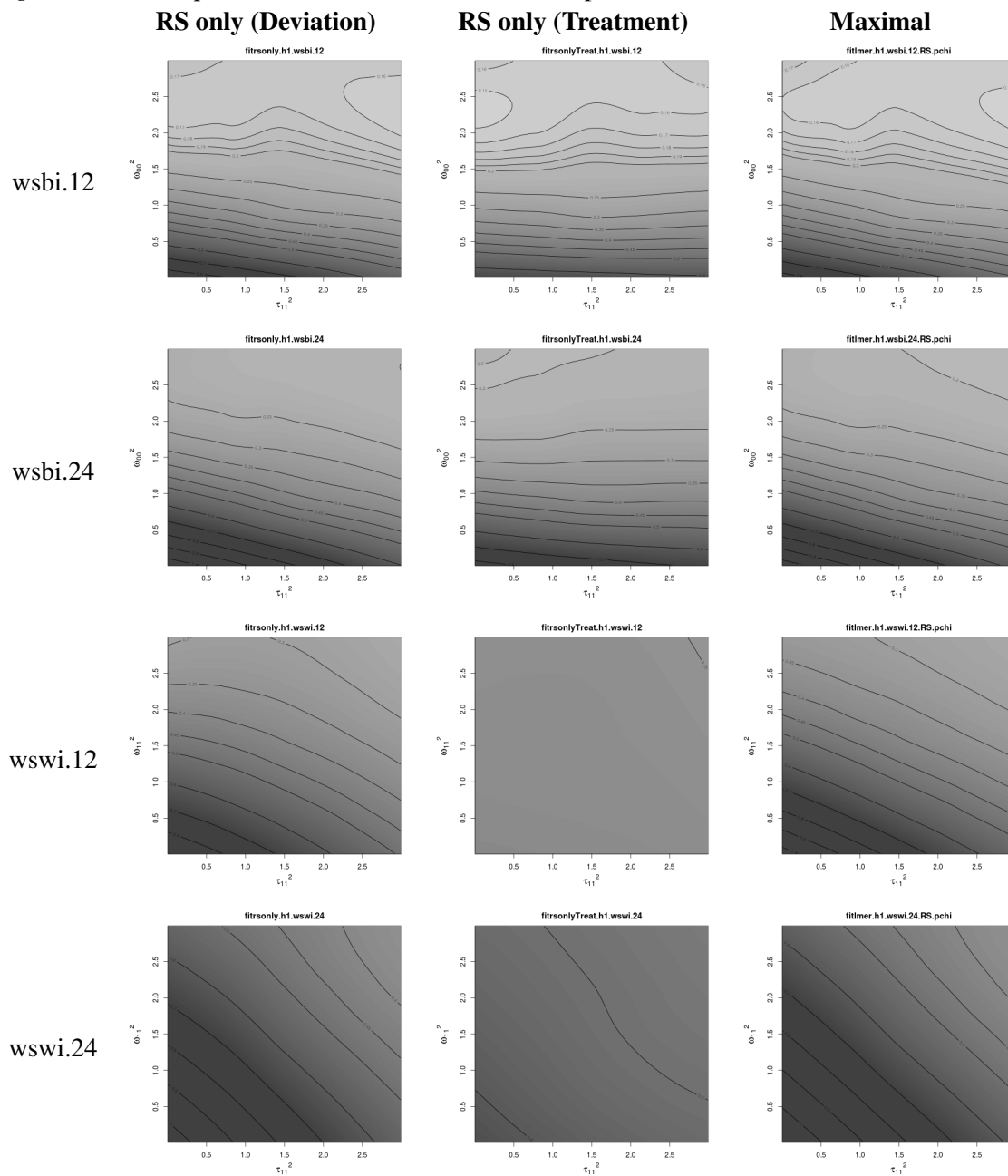


Figure 25. Heatmaps, Between-items parameterization 2, Type I Error

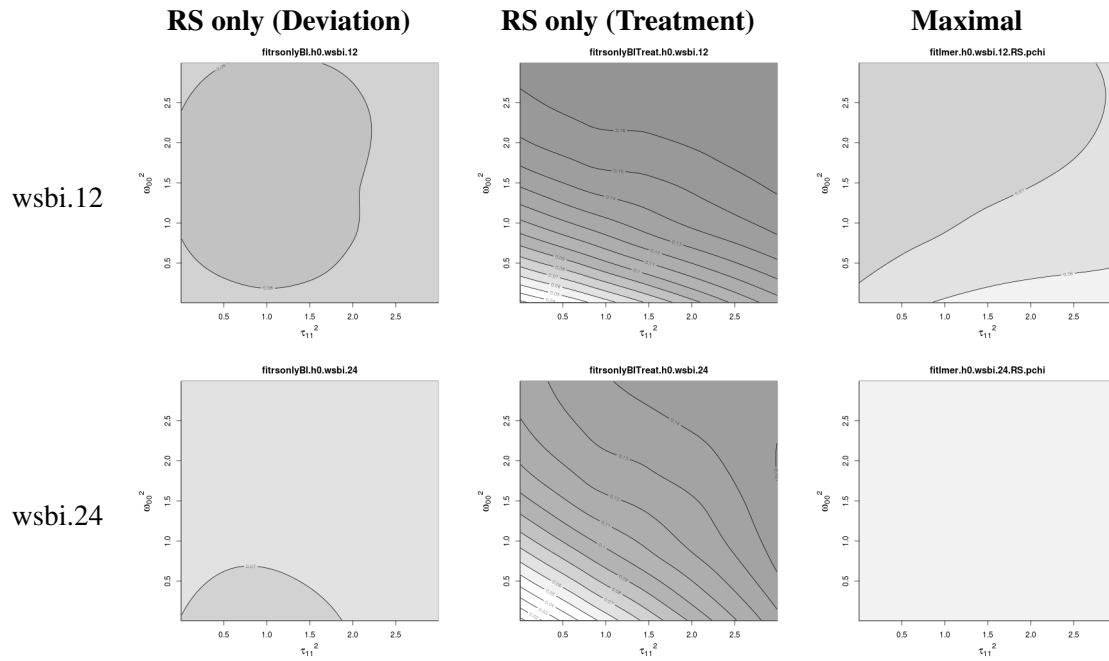


Figure 26. Heatmaps, Between-items parameterization 2, Power

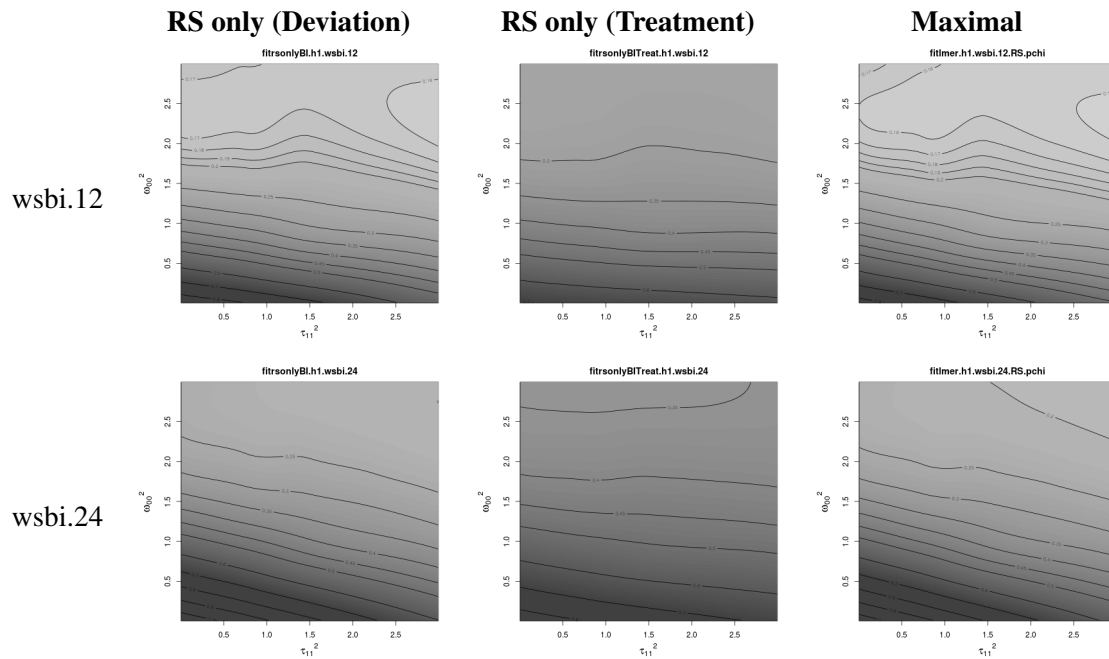




Table 11

*Between-items parameterization 2, Power*

	Deviation $t$ as $z$	Treatment $t$ as $z$	Maximal $t$ as $z$	Deviation $\chi^2$	Treatment $\chi^2$	Maximal $\chi^2$
wsbi.12	0.32274	0.42197	0.30022	0.28872	0.37181	0.26725
wsbi.24	0.40103	0.49264	0.38249	0.38064	0.46814	0.36361

Table 12

*Independent Random Intercepts and Random Slopes, Type I Error*

	Deviation $t$ as $z$	Treatment $t$ as $z$	Maximal $t$ as $z$	Deviation $\chi^2$	Treatment $\chi^2$	Maximal max $\chi^2$
wsbi.12	.08611	.09052	.08619	.06933	.07239	.07026
wsbi.24	.06433	.06728	.06485	.05696	.05937	.05750
wswi.12	.07246	.08974	.07185	.05919	.07465	.05885
wswi.24	.06245	.07463	.06265	.05563	.06650	.05594

Table 13

*Independent Random Intercepts and Random Slopes, Power*

	Deviation $t$ as $z$	Treatment $t$ as $z$	Maximal $t$ as $z$	Deviation $\chi^2$	Treatment $\chi^2$	Maximal $\chi^2$
wsbi.12	.29953	.30704	.30022	.26656	.27283	.26725
wsbi.24	.38159	.38759	.38249	.36254	.36848	.36361
wswi.12	.49710	.53355	.49641	.46116	.49853	.46032
wswi.24	.62856	.65386	.62941	.60956	.63640	.61037

*R Scripts*

- Random slopes only (no random intercepts)

```
library(simgen)

dev2treat <- function(nitem, wsbi, mcr.params) {
  # this function converts the default "deviation" coding
  # of mkDf to treatment coding
  x <- mkDf(nitem=nitem, wsbi=wsbi, mcr.params=mcr.params)
  x$Cond <- ifelse(x$Cond==-.5,0,1)
  return(x)
}

param.mx <- createParamMx()

# Type I Error
param.mx[, "eff"] <- 0

mcRun("fitrsonly", mcr.outfile="fitrsonly.h0.wsbi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=param.mx,
      nsubj=24, nitem=12, wsbi=TRUE)

mcRun("fitrsonly", mcr.outfile="fitrsonly.h0.wsbi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=param.mx,
      nsubj=24, nitem=24, wsbi=TRUE)

mcRun("fitrsonly", mcr.outfile="fitrsonly.h0.wswi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=param.mx,
      nsubj=24, nitem=12, wsbi=FALSE)

mcRun("fitrsonly", mcr.outfile="fitrsonly.h0.wswi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=param.mx,
      nsubj=24, nitem=24, wsbi=FALSE)

mcRun("fitrsonly", mcr.outfile="fitrsonlyTreat.h0.wsbi.12.csv",
      mcr.xdatFnc="dev2treat", mcr.varying=param.mx,
      nsubj=24, nitem=12, wsbi=TRUE)

mcRun("fitrsonly", mcr.outfile="fitrsonlyTreat.h0.wsbi.24.csv",
      mcr.xdatFnc="dev2treat", mcr.varying=param.mx,
      nsubj=24, nitem=24, wsbi=TRUE)

mcRun("fitrsonly", mcr.outfile="fitrsonlyTreat.h0.wswi.12.csv",
```

```

    mcr.xdatFnc="dev2treat", mcr.varying=param.mx,
    nsubj=24, nitem=12, wsbi=FALSE)

mcRun("fitrsonly", mcr.outfile="fitrsonlyTreat.h0.wswi.24.csv",
    mcr.xdatFnc="dev2treat", mcr.varying=param.mx,
    nsubj=24, nitem=24, wsbi=FALSE)

# Power
param.mx[, "eff"] <- .8

mcRun("fitrsonly", mcr.outfile="fitrsonly.h1.wsbi.12.csv",
    mcr.xdatFnc="mkDf", mcr.varying=param.mx,
    nsubj=24, nitem=12, wsbi=TRUE)

mcRun("fitrsonly", mcr.outfile="fitrsonly.h1.wsbi.24.csv",
    mcr.xdatFnc="mkDf", mcr.varying=param.mx,
    nsubj=24, nitem=24, wsbi=TRUE)

mcRun("fitrsonly", mcr.outfile="fitrsonly.h1.wswi.12.csv",
    mcr.xdatFnc="mkDf", mcr.varying=param.mx,
    nsubj=24, nitem=12, wsbi=FALSE)

mcRun("fitrsonly", mcr.outfile="fitrsonly.h1.wswi.24.csv",
    mcr.xdatFnc="mkDf", mcr.varying=param.mx,
    nsubj=24, nitem=24, wsbi=FALSE)

mcRun("fitrsonly", mcr.outfile="fitrsonlyTreat.h1.wsbi.12.csv",
    mcr.xdatFnc="dev2treat", mcr.varying=param.mx,
    nsubj=24, nitem=12, wsbi=TRUE)

mcRun("fitrsonly", mcr.outfile="fitrsonlyTreat.h1.wsbi.24.csv",
    mcr.xdatFnc="dev2treat", mcr.varying=param.mx,
    nsubj=24, nitem=24, wsbi=TRUE)

mcRun("fitrsonly", mcr.outfile="fitrsonlyTreat.h1.wswi.12.csv",
    mcr.xdatFnc="dev2treat", mcr.varying=param.mx,
    nsubj=24, nitem=12, wsbi=FALSE)

mcRun("fitrsonly", mcr.outfile="fitrsonlyTreat.h1.wswi.24.csv",
    mcr.xdatFnc="dev2treat", mcr.varying=param.mx,
    nsubj=24, nitem=24, wsbi=FALSE)

```

- Independent slopes and intercepts

```
library(simgen)
```

```

dev2treat <- function(nitem, wsbi, mcr.params) {
  # this function converts the default "deviation" coding
  # of mkDf to treatment coding
  x <- mkDf(nitem=nitem, wsbi=wsbi, mcr.params=mcr.params)
  x$Cond <- ifelse(x$Cond==-.5,0,1)
  return(x)
}

param.mx <- createParamMx()

# Type I Error
param.mx[, "eff"] <- 0

mcRun("fitnocorr", mcr.outfile="fitnocorr.h0.wsbi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=param.mx,
      nsubj=24, nitem=12, wsbi=TRUE)

mcRun("fitnocorr", mcr.outfile="fitnocorr.h0.wsbi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=param.mx,
      nsubj=24, nitem=24, wsbi=TRUE)

mcRun("fitnocorr", mcr.outfile="fitnocorr.h0.wswi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=param.mx,
      nsubj=24, nitem=12, wsbi=FALSE)

mcRun("fitnocorr", mcr.outfile="fitnocorr.h0.wswi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=param.mx,
      nsubj=24, nitem=24, wsbi=FALSE)

mcRun("fitnocorr", mcr.outfile="fitnocorrTreat.h0.wsbi.12.csv",
      mcr.xdatFnc="dev2treat", mcr.varying=param.mx,
      nsubj=24, nitem=12, wsbi=TRUE)

mcRun("fitnocorr", mcr.outfile="fitnocorrTreat.h0.wsbi.24.csv",
      mcr.xdatFnc="dev2treat", mcr.varying=param.mx,
      nsubj=24, nitem=24, wsbi=TRUE)

mcRun("fitnocorr", mcr.outfile="fitnocorrTreat.h0.wswi.12.csv",
      mcr.xdatFnc="dev2treat", mcr.varying=param.mx,
      nsubj=24, nitem=12, wsbi=FALSE)

mcRun("fitnocorr", mcr.outfile="fitnocorrTreat.h0.wswi.24.csv",
      mcr.xdatFnc="dev2treat", mcr.varying=param.mx,
      nsubj=24, nitem=24, wsbi=FALSE)

```

```
# Power
param.mx[, "eff"] <- .8

mcRun("fitnocorr", mcr.outfile="fitnocorr.h1.wsbi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=param.mx,
      nsubj=24, nitem=12, wsbi=TRUE)

mcRun("fitnocorr", mcr.outfile="fitnocorr.h1.wsbi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=param.mx,
      nsubj=24, nitem=24, wsbi=TRUE)

mcRun("fitnocorr", mcr.outfile="fitnocorr.h1.wswi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=param.mx,
      nsubj=24, nitem=12, wsbi=FALSE)

mcRun("fitnocorr", mcr.outfile="fitnocorr.h1.wswi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=param.mx,
      nsubj=24, nitem=24, wsbi=FALSE)

mcRun("fitnocorr", mcr.outfile="fitnocorrTreat.h1.wsbi.12.csv",
      mcr.xdatFnc="dev2treat", mcr.varying=param.mx,
      nsubj=24, nitem=12, wsbi=TRUE)

mcRun("fitnocorr", mcr.outfile="fitnocorrTreat.h1.wsbi.24.csv",
      mcr.xdatFnc="dev2treat", mcr.varying=param.mx,
      nsubj=24, nitem=24, wsbi=TRUE)

mcRun("fitnocorr", mcr.outfile="fitnocorrTreat.h1.wswi.12.csv",
      mcr.xdatFnc="dev2treat", mcr.varying=param.mx,
      nsubj=24, nitem=12, wsbi=FALSE)

mcRun("fitnocorr", mcr.outfile="fitnocorrTreat.h1.wswi.24.csv",
      mcr.xdatFnc="dev2treat", mcr.varying=param.mx,
      nsubj=24, nitem=24, wsbi=FALSE)
```

### Random effects in real datasets

For generating the simulated populations included in the manuscript, we assumed a uniform distribution over a set of parameters, reflecting our lack of information about the nature of clustering in real datasets. To gain some preliminary information about real datasets, we revisited some of our own data using linear mixed-effect modeling (see Tables 14–15).

Table 14

*Description of datasets used in the analysis. “S/I” is the number of subjects/items in the study. The “Design” column describes the design of the study, indicating the number of levels of each factor (2) as well as whether it was within (W) or between (B) subjects / items.*

ID	Article	Expt.	S/I	Design
1–4	Matsuki et al. (2011)	3	32/48	2W / 2W
5	Yao & Scheepers (2011)	1	20/24	2Wx2W / 2Wx2W
6	Yao & Scheepers (2011)	2	48/24	2Wx2W / 2Wx2W
7-8	Levy et al. (2011)	1	41/24	2W / 2W
9	Rohde et al. (2011)	2	55/20	2W / 2W
10	Keysar et al. (2000)	1	18/12	2W / 2W
11	Kronmüller & Barr (2007)	2	56/32	2Wx2Wx2W / 2Wx2Wx2W
12	Barr & Seyfeddinipur (2011)	1	92/12	2Wx2W / 2Wx2W
13	Gann & Barr (in press)	1	64/16	2Bx2Wx2W / 2Wx2Wx2B

For convenience, the distributions of population parameters as given in the original manuscript are reproduced in Table 16.

*Slope variance relative to intercept variance.* The first analysis considered how much of the total variance related to a given sampling unit (subject or item) was attributable to the random slope versus the random intercept. We used the following formula:

$$\frac{\tau_{11}^2}{\tau_{00}^2 + \tau_{11}^2}$$

Results from this analysis are in Table 17. There is a broad range across experiments, with slope variance accounting for anywhere from <1% of the total subject variance up to 78%. The by-item variance also shows broad dispersion, with slope variance carrying from 3% to 73% of total item variance. The by-subject measurement seems bimodally distributed, with observations clumping toward either end of the range.

*Random effects in relation to residual variance.* One factor that became apparent in our analysis of real datasets was that our simulations assumed that by-subject or by-item random effect variance was roughly proportionate to residual variance. This assumption is unlikely to hold in actual datasets, where the random effect variance is typically much smaller than the residual variance. In other words, actual data sets tend to be much noisier than our simulated datasets.

The results for each dataset are in Table 18, showing the residual variance and the by-subject/by-item random effect variance as a proportion of this residual variance. For each dataset containing multiple factors (e.g., in 2x2 designs), we present the average by-subject and average by-item slopes.

Table 15

*Description of datasets used in the analysis.*

ID	Task	Manipulation	Dependent variables
1-4	Silent reading (Eye-tracking)	high/low event prototypicality of patient noun	First fixation, Gaze duration, "Go past" times, Total time
5	Oral reading	context (fast/slow), quotation style (direct/indirect)	Syllables per second
6	Silent reading	context (fast/slow), quotation style (direct/indirect)	"Go past" times (ms)
7-8	Self-paced reading		Reading times (same DV considered separately over two manipulations)
9	Self-paced reading		Reading time
10	Visual world eyetracking (spoken lang comprehension)	Competitor present/absent	Latency of target gaze
11	Spoken language comprehension	Speaker, Precedent, Cognitive Load	Response Time
12	Spoken language comprehension	Speaker, Filled/unfilled pause	Distance of Mouse Cursor from Target
13	Referential Communication	Listener, New/Old Referent, Feedback	Speech Onset Latency

One thing that is apparent is that the by-subject and by-item random effects, as a proportion of residual variance, vary wildly across studies (from 0% to 187% of residual variance). Typically, they are only about 10-40% of the total variance. Generally, we also see more variance on the intercept than on the slope. It should also be noted that slope variance does not seem to be uniformly distributed over the range; rather, it seems clumped at the top and bottom of the range. It should be kept in mind that whereas intercept variances index differences in overall level, slope variances index differences in sensitivity to manipulations. It is possible that participants (or items) were simply insensitive to some of the manipulations in these studies, yielding no variance nor any overall effect.

*Subsampling from the observed ranges.* The next analysis addresses how unrepresentative the main results from our simulations might be. Specifically, did the parameter space we used lead us to be too pessimistic about random-intercepts-only models and model-selection approaches and too optimistic about maximal models?

To address this, from the values reported in the previous section we derived plausible ranges from which to subsample our simulation data (Table 19).

Table 16

Ranges for the population parameters;  $\sim U(\min, \max)$  means the parameter was sampled from a uniform distribution with range  $[\min, \max]$ .

Parameter	Description	Value
$\beta_0$	grand-average intercept	$\sim U(-3, 3)$
$\beta_1$	grand-average slope	0 ( $H_0$ true) or .8 ( $H_1$ true)
$\tau_{00}^2$	by-subject variance of $S_{0s}$	$\sim U(0, 3)$
$\tau_{11}^2$	by-subject variance of $S_{1s}$	$\sim U(0, 3)$
$\rho_S$	correlation between $(S_{0s}, S_{1s})$ pairs	$\sim U(-.8, .8)$
$\omega_{00}^2$	by-item variance of $I_{0i}$	$\sim U(0, 3)$
$\omega_{11}^2$	by-item variance of $I_{1i}$	$\sim U(0, 3)$
$\rho_I$	correlation between $(I_{0i}, I_{1i})$ pairs	$\sim U(-.8, .8)$
$\sigma^2$	residual error	$\sim U(0, 3)$
$p_{missing}$	proportion of missing observations	$\sim U(.00, .05)$

This resulted in the selection of 3154 (about 3%) of the total runs for further analysis. On this subsample, we compared the power of maximal LMEMs to min- $F'$ ,  $F_1 \times F_2$ , RI-only LMEMs, and LMEMs using model selection for the random effects. From the various possible model selection techniques for within-items design, we chose the best performing model (the “backward best path” model,  $\alpha$  for inclusion set to .05) to see if it would improve power in this region of the space relative to the maximal model. The results are in the Tables 20 and 21.

For between-items (wsbi) designs, the Type I error rates do not differ much from the original simulations for any of the analyses. For the within-items designs, ANOVA-based and maximal LMEMs perform similarly on the subsample as they do on the original sample. However, model selection approaches become slightly more anticonservative, while random-intercepts-only LMEM becomes substantially less anticonservative on the subsample. But even though RI-only LMEMs are performing better, their Type I error rates still remain intolerably high (.25 and .32).

It is notable that all approaches (including maximal LMEMs) are more powerful on the subspace than on the original dataset. When power is corrected for anticonservativity (rows labeled “CP” in the table), one interesting outcome is that in the parameter subspace, maximal LMEMs are nearly always just as powerful and occasionally even more powerful than approaches using model selection. Finally, for within-items designs, RI-only LMEMs, once corrected for anticonservativity, showed only a very minor advantage relative to maximal LMEMs (6% and 4% increase in power for 12 and 24 item datasets, respectively). In contrast, for within-items designs, model selection approaches showed a disadvantage in corrected power relative to maximal LMEMs (11% and 2.5% drop for 12 and 24 item datasets, respectively).

### Summary

In closing, the analyses of actual datasets show that our simulations assumed that by-subject and by-item random variance was a bigger portion of the total variance than actually turned out to be the case. Yet it was clear that even for the subregion of the parameter space spanning the range



Table 17

*Slope variance as a proportion of total variance for a given sampling unit*

ID	Subject	Item
1	.00003	.73085
2	.00564	.37926
3	.00362	.29886
4	.00035	.04059
5	.17032	.25304
6	.00463	.03488
7	.35727	.51755
8	.01663	.39627
9	.04805	.44358
10	.64403	.04626
11	.49218	.49753
12	.77840	n/a
13	.40245	.57898
MIN	.00003	.03488
MEAN	.22489	.35147
MED	.04805	.38777
MAX	.77840	.73085

of the observed data sets, maximal models reflect the best compromise between controlling Type I error and power. Unfortunately, we have no way of knowing whether our datasets are representative of the kinds of experimental datasets analyzed in experimental psychology. Nonetheless, these findings lend further confidence to our contention that maximal LMEMs provide the best approach for confirmatory hypothesis testing.

#### *Sources of real datasets*

Barr, D. J., & Seyfeddinipur, M. (2011). The role of fillers in listener attributions for speaker disfluency. *Language and Cognitive Processes*, 25, 441-455.

Gann, T. M., & Barr, D. J. (in press). Speaking from experience: Audience design as expert performance. *Language and Cognitive Processes*, Manuscript in press.

Keysar, B., Barr, D. J., Balin, J. A., & Brauner, J. S. (2000). Taking perspective in conversation: The role of mutual knowledge in comprehension. *Psychological Science*, 11, 32-38.

Levy, R., Fedorenko, E., Breen, M., & Gibson, E. (2011). The processing of extraposed structures in English. *Cognition*, 122, 12-36.

Matsuki, K., Chow, T., Hare, M., Elman, J. L., Scheepers, C., & McRae, K. (2011). Event-based plausibility immediately influences on-line language comprehension. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 37, 913-934.

Kronmüller, E., & Barr, D. J. (2007). Perspective-free pragmatics: Broken precedents and the

Table 18

*Random effects as a proportion of residual variance.*

ID	Residual	$\tau_{00}^2$	$\tau_{11}^2$	$\omega_{00}^2$	$\omega_{11}^2$
1	3572	0.2163	0.0000	0.0143	0.0389
2	8438.8531	0.1404	0.0008	0.0282	0.0172
3	24387.6356	0.1046	0.0004	0.1339	0.0571
4	29933.581	0.3207	0.0001	0.1473	0.0062
5	0.493532	1.8765	0.3852	1.0238	0.3468
6	275362.526	0.4492	0.0021	1.0269	0.0371
7	230191	0.1058	0.0588	0.0910	0.0976
8	231824.16	0.1721	0.0029	0.0864	0.0567
9	51371.6	0.4117	0.0208	0.1076	0.0858
10	7536625	0.0363	0.0656	0.1198	0.0058
11	406043	0.2286	0.2216	0.3269	0.3237
12	0.128353	0.1042	0.3661	0.0000	0.0000
13	242830	0.4258	0.2867	0.0820	0.1127
MEAN		0.3532	0.1085	0.2452	0.0912
MED		0.2163	0.0208	0.1076	0.0567
MIN		0.0363	0.0000	0.0000	0.0000
MAX		1.8765	0.3852	1.0269	0.3468

recovery-from-preemption hypothesis. *Journal of Memory and Language*, 56, 436-455.

Rohde, H., Levy, R., & Kehler, A. (2011). Anticipating explanations in relative clause processing. *Cognition*, 118, 339-358.

Yao, B. & Scheepers, C. (2011). Contextual modulation of reading rate for direct versus indirect speech quotations. *Cognition*, 121 447-453.

Table 19

*Ranges for subsampling simulation data.*

Parameter	Min	Max
$\tau_{00}^2/\sigma^2$	0.00	0.45
$\tau_{11}^2/\sigma^2$	0.00	0.40
$\omega_{00}^2/\sigma^2$	0.00	0.35
$\omega_{11}^2/\sigma^2$	0.00	0.35

Table 20

*Type I error rate for original simulations and for the parameter subspace*

	Subspace wsbi.12	Original wsbi.12	Subspace wsbi.24	Original wsbi.24	Subspace wswi.12	Original wswi.12	Subspace wswi.24	Original wswi.24
min- $F'$	.0384	.0445	.0387	.0446	.0216	.0271	.0263	.0307
$F_1 \times F_2$	.0653	.0628	.0770	.0772	.0549	.0574	.0656	.0724
LMEM, Maximal	.0758	.0703	.0596	.0575	.0611	.0589	.0592	.0559
LMEM, Selection	.0796	.0702	.0612	.0575	.1053	.0683	.0726	.0579
LMEM, RI-only	.1055	.1023	.1027	.1105	.2483	.4398	.3167	.4980

Table 21

*Power (and corrected power, CP) for original simulations and for the parameter subspace*

	Subspace wsbi.12	Original wsbi.12	Subspace wsbi.24	Original wsbi.24	Subspace wswi.12	Original wswi.12	Subspace wswi.24	Original wswi.24
min- $F'$	.3003	.2099	.4984	.3281	.4471	.3268	.6826	.5116
$F_1 \times F_2$	.3675	.2518	.5961	.4034	.5961	.4400	.8098	.6432
LMEM, Maximal	.3965	.2672	.5643	.3636	.6215	.4603	.7921	.6104
LMEM, Selection	.4017	.2689	.5685	.3636	.6715	.4730	.8025	.6120
LMEM, RI-only	.4543	.3185	.6368	.4492	.8708	.8534	.9610	.9351
$F_1 \times F_2$ (CP)	.3291	.2236	.5187	.3375	.5748	.4158	.7695	.5780
LMEM, Maximal (CP)	.3242	.2225	.5322	.3418	.5830	.4325	.7685	.5914
LMEM, Selection (CP)	.3266	.2229	.5301	.3424	.5200	.4144	.7495	.5880
LMEM, RI-only (CP)	.3231	.2156	.5040	.3140	.6180	.3791	.7961	.5313

### Validity of random effects parameter estimates from nonconverged models

The question here was whether the random effect parameter estimates from a nonconverged model in `lmer` are informative in deciding which higher-order term to remove. This analysis considered only within-item designs, where there were two random slopes that could be ranked in terms of their relative size. The correlation between the rank of the estimates from maximal-random-effects models that did not converge and the rank of the generative parameters for the 12-item design was .74, with  $p < .0001$  ( $N=387$  nonconverging models); for the 24-item design it was .83, with  $p < .0001$  ( $N=192$  nonconverging models). Thus, using the nonconverged estimates seems a good strategy.

#### Source Code

```
library(simgen)

get.ncvix <- function(x) {
  # return a logical vector indicating which runs did not converge (TRUE)
  (1:nrow(x)) [x[, "fm"] > 0]
}

param.mx <- createParamMx(h0=FALSE)

# load in the maximal random effects models
load(file="test/fitlmer.h0.wswi.12.RS.RData")
ix.12 <- get.ncvix(mx) # keep only those that didn't converge
x.12 <- mx[ix.12,]
prm.12 <- param.mx[ix.12,]
```

```

load(file="test/fitlmer.h0.wswi.24.RS.RData")
ix.24 <- get.ncvix(mx) # keep only those that didn't converge
x.24 <- mx[ix.24,]
prm.24 <- param.mx[ix.24,]

# recreate the dataframes that the models will be fit to
df.12 <- mclapply(1:nrow(prm.12),
  function(x) {mkDf(nsubj=24,nitem=12,params=prm.12[x,],wsbi=FALSE)})

# refit the models
library(lme4)
lmer.12 <- mclapply(df.12,
  function(x) {
    lmer(Resp ~ Cond + (1 + Cond | SubjID) + (1 + Cond | ItemID), data=x,
      na.action=na.omit, REML=FALSE)
  })

# recreate the dataframes that the models will be fit to
df.24 <- mclapply(1:nrow(prm.24),
  function(x) {mkDf(nsubj=24,nitem=24,params=prm.24[x,],wsbi=FALSE)})

# refit the models
lmer.24 <- mclapply(df.24,
  function(x) {
    lmer(Resp ~ Cond + (1 + Cond | SubjID) + (1 + Cond | ItemID), data=x,
      na.action=na.omit, REML=FALSE)
  })

getlmerrank <- function(x) {
  vc <- VarCorr(x)
  myrank <- ifelse(abs(vc$SubjID[2,2]) <= abs(vc$ItemID[2,2]),0,1)
  return(myrank)
}

rank.prm.12 <- ifelse(abs(prm.12[, "t11"])<=abs(prm.12[, "w11"]),0,1)
rank.lmer.12 <- unlist(lapply(lmer.12, getlmerrank))

rank.prm.24 <- ifelse(abs(prm.24[, "t11"])<=abs(prm.24[, "w11"]),0,1)
rank.lmer.24 <- unlist(lapply(lmer.24, getlmerrank))

ranks.12 <- cbind(nitem=12, param.rank=rank.prm.12, lmer.rank=rank.lmer.12)
ranks.24 <- cbind(nitem=24, param.rank=rank.prm.24, lmer.rank=rank.lmer.24)

ranks <- rbind(ranks.12,ranks.24)

cor(ranks[ranks[, "nitem"]==12,])
cor(ranks[ranks[, "nitem"]==24,])

shufflemx <- function(x) {
  ix.12 <- (1:nrow(x))[x[, "nitem"]==12]
  ix.24 <- (1:nrow(x))[x[, "nitem"]==24]
  shuf.12 <- cbind(nitem=12,param.rank=x[ix.12,"param.rank"],lmer.rank=x[sample(ix.12),"lmer.rank"])
  shuf.24 <- cbind(nitem=24,param.rank=x[ix.24,"param.rank"],lmer.rank=x[sample(ix.24),"lmer.rank"])
  return(rbind(shuf.12,shuf.24))
}

getcor <- function(x) {
  c12 <- cor(x[x[, "nitem"]==12,c("param.rank","lmer.rank")])[1,2]
  c24 <- cor(x[x[, "nitem"]==24,c("param.rank","lmer.rank")])[1,2]
  return(c(c12=c12,c24=c24))
}

library(multicore)

```

```
nmc <- 9999
perm.mx <- matrix(ncol=2, nrow=nmc+1)
perm.mx[1,] <- getcor(ranks)
colnames(perm.mx) <- names(getcor(ranks))
mx2 <- mclapply(1:nmc, function(x) {getcor(shufflemx(ranks))})
perm.mx[2:(nmc+1),] <- matrix(unlist(mx2), ncol=2, byrow=T)
nge <- apply(apply(perm.mx, 2, function(x) {abs(x)>=abs(x[1])}), 2, sum)
print(nge/nrow(perm.mx))
save.image(file="ranking-test-results2.RData")
```

### Random effects for control predictors

One of the most compelling aspects of mixed-effects models is the ability to include almost any control predictor—by which we mean a property of an experimental trial which may affect the response variable but is not of theoretical interest in a given analysis—desired by the researcher. In principle, including control variables in an analysis can rule out potential confounds and increase statistical power by reducing residual noise. Given the investigations in the present paper, however, the question naturally arises: in order to guard against anti-conservative inference about a predictor  $X$  of theoretical interest, do we need by-subject and by-item random effects for all our control predictors  $C$  as well? Suppose, after all, if there is no underlying fixed effect of  $C$  but there is a random effect of  $C$ —could this create anti-conservative inference in the same way as omitting a random effect of  $X$  in the analysis could? To put this issue in perspective via an example, Kuperman, Bertram, & Baayen (2010) present an LME analysis of fixation durations in Dutch reading; for the interpretation of each main effect, the other seven may be thought of as serving as controls. The prospect of trying to fit eight random effects (plus correlation terms!) both by subjects and by items no doubt makes some readers cringe; and many studies include far more than eight predictors!

Fortunately, we have reassuring news on this front: omitting random effects for control predictors should not generally lead to anti-conservativity, so long as random effects are included for predictors of theoretical interest. To see the logic, consider: either a set of control predictors  $C$  and the theoretically interesting predictor  $X$  are multicollinear (a generalization of correlation, meaning that  $X$  can be predicted accurately from  $C$ ) or they are not. If they are not, then even if  $C$  affects the response there will be no tendency to generate spurious effects of  $X$ . If  $C$  and  $X$  are multicollinear and there is an underlying fixed effect of  $C$  on the response, then including a fixed effect of  $C$  in the analysis is enough (omitting  $C$  would of course make inference about  $X$  anti-conservative). If  $C$  and  $X$  are multicollinear and there is a random effect of  $C$  on the response, then the random effect of  $X$  itself is enough to avoid anti-conservative inference! To demonstrate the validity of this point, we conducted Monte Carlo simulations of 24-subject, 24-item within/within experiments with a two-level treatment  $X$  and a continuous control factor  $C$  correlated with  $X$ . In the generative model there was always a random effect but no fixed effect of  $X$ . In all analyses we compared behavior of LME analyses with and without random slopes for  $C$ .<sup>2</sup>

With a true fixed effect but no random effect of  $C$ , we find Type I error rates at  $\alpha = 0.05$  are slightly above nominal, consistent with our general results; Type I error is also slightly higher for analyses without control-predictor random slopes (error rate of 0.057) than for analyses with control-predictor random slopes (0.054). However, Type I error rates are similarly higher when there is no true effect of  $C$  whatsoever for analyses without control-predictor random effects (error rate of 0.060) than for analyses with them (0.057)! When there is a true random effect but no fixed effect of  $C$ , we find that analyses without random effects of  $C$  are actually *conservative* (Type I error rate of 0.030), whereas analyses with random effects of  $C$  are slightly above nominal (0.056). The reason for this conservativity is that the LME model can only account for the cluster-specific

<sup>2</sup>In these simulations  $C$  was distributed normally with standard deviation of 0.5 and means of 1 and 2 for the two levels of the experimental treatment respectively. When there was a fixed effect of  $C$  its value was  $\beta_C = 5$ . By-subjects and by-items covariance matrices were  $\mathcal{I}_2$  for cases with no random effect of  $C$  and  $\mathcal{I}_3$  for cases with a random effect of  $C$ , trial-level error standard deviation of 2. Analysis specifications were `response ~ A + C + (A | subj) + (A | item)` versus `response ~ A + C + (A + C | subj) + (A + C | item)`.

variation seen in  $C$  by attributing it to large random effects of  $X$ ; but large estimated random effects of  $X$  reduce confidence in any possible fixed effect of  $X$ , leading to conservative inference.

Hence random effects for control predictors are not strictly necessary to avoid anti-conservative inference. However, this analysis underscores a different point: failing to add random effects for controls may in fact rob the investigator of the opportunity to considerably sharpen inferences about predictors of theoretical interest, since control-predictor random effects may potentially be needed to soak up important sources of noise in one's data.

#### *References*

Kuperman, V., Bertram, R., & Baayen, R. H. (2010). Processing trade-offs in the reading of Dutch derived words. *Journal of Memory and Language*, 62, 83–97.

### Performance on datasets with high random correlations

In our paper, we ran simulations where the range for the random correlation was  $[-.8, .8]$ . A reviewer asked us to consider high correlations ( $>.8$ ), based on the fact that high correlations are often seen in the output from `lmer`. (Note, however, that a high correlation may reflect numerical estimation problems rather than high random correlations “in the world.”) Here, we verify that our main results hold for datasets with high correlations ( $>.8$ ). To simplify matters, we consider only datasets including a single within-subject/within-item factor, and with 24 subject and 12 or 24 items. We also considered only the best performing stepwise model (the backwards “best path” model).

#### Method

We created 10000 additional datasets. In the first 5000 datasets, the by-subject random correlation varied from .8 to 1 or -.8 to -1, while the by-item random correlation varied from -1 to 1. For the second 5000 datasets, the by-item random correlation varied from .8 to 1 or -.8 to -1, while the by-subject random correlation varied from -1 to 1. All other parameters had the same ranges as in the paper.

#### Results

Table 22

*Results from models with pre-defined random effects structure.*

Simulation	Model	typeI.12i	typeI.24i	power.12i	power.24i
High Subject Correlation	LMEM, Maximal, $\chi^2_{LR}$	0.062	0.056	0.460	0.614
High Subject Correlation	LMEM, No Random Correlations $\chi^2_{LR}$	0.064	0.055	0.460	0.612
High Item Correlation	LMEM, Maximal, $\chi^2_{LR}$	0.059	0.054	0.456	0.611
High Item Correlation	LMEM, No Random Correlations $\chi^2_{LR}$	0.060	0.055	0.458	0.608

For maximal models or models with no random correlations (Table 22), all results for Type I error are within .006 of the original simulations, and all results for power are within .004 of the original simulations (results from the original simulations can be found in Table 6 of the main paper).

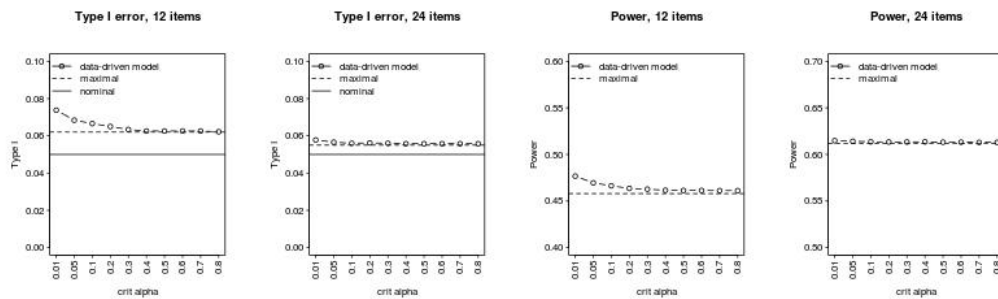


Figure 27. Results for models with data-driven random effects.

Results for data-driven models are given in Figure 27. The Type I error / power of the best stepwise model when random correlations are high is not appreciably different from its performance from the original parameter space (see Figure 2 of our paper), and asymptotes toward the performance of



the maximal model.

### The *simgen* package

The `simgen` package is an add-on package for R, which contains functions used for running the simulations reported in Barr, Levy, Scheepers, and Tily. The package also contains a general-purpose function `mcRun()` for performing Monte Carlo runs on user-specified data, parameters, and functions. See `?mcRun` for details.

If you find any bugs, please report them to [Dale Barr](#), the package author and maintainer.

### Download and Installation

Current version: 1.54

Unix, Linux, Macintosh: [http://talklab.psy.gla.ac.uk/simgen/simgen\\_1.54.tar.gz](http://talklab.psy.gla.ac.uk/simgen/simgen_1.54.tar.gz)

Windows: [http://talklab.psy.gla.ac.uk/simgen/simgen\\_1.54.zip](http://talklab.psy.gla.ac.uk/simgen/simgen_1.54.zip)

The package is not available on CRAN, and thus must be downloaded and installed manually. If you experience any trouble installing `simgen`, please see the [CRAN website](#) for assistance on installing add-on packages.

#### *Installing on Unix/Linux/MacOS*

To speed up the Monte Carlo simulations, it is recommended that you also install the [multicore](#) package. Unfortunately, `multicore` is not available for Windows workstations, and so `simgen` will be unable to take advantage of multicore processing. If you are using Unix/Linux/MacOS, install `multicore` as shown below before installing `simgen`.

```
install.packages("multicore")
```

To install `simgen` from the command line (MacOS, Unix, Linux):

```
R CMD INSTALL simgen_1.54.tar.gz
```

To install it from within R:

After making sure that you set your working directory to where the `.tar.gz` file is located, type

```
install.packages("simgen_1.54.tar.gz", repos=NULL)
```

#### *Installing on Windows*

For Windows workstations use the following command from within R.

```
install.packages("simgen_1.54.zip", repos=NULL)
```

### R scripts for Monte Carlo simulations

Below are R scripts that can be used to replicate the Monte Carlo simulations reported in the manuscript. Please note:

You will need to **install the `simgen` package** to be able to run the simulations.

The output from the `fitstepwise` function for model selection has four-dimensional data stored in two dimensional format. The four dimensions are:

1. run (each line in the output file is a single Monte Carlo run)
2. alpha level (the level at which inclusion of the random slope is tested)
3. variable (the statistic output from the fitting procedure; e.g.,  $t$ ,  $\chi^2$ )
4. the direction of model building (forward, backward)

To reassemble the data into a four dimensional array for ease of processing, using the following code.

```
x <- read.csv("somefilename.csv", header=TRUE)

ax <- array(as.matrix(x), dim=c(nrow(x), 10, 6, 2),
            dimnames=list(NULL, crit=c(.01, .05, seq(.1, .8, .1)),
                           stat=c("fm", "t", "chi", "pt", "pchi", "eff"),
                           dir=c("forward", "backward")))
```

### Standard runs (no model selection)

```
library(simgen)

# number of Monte Carlo runs (nexp)
# set to small value for testing purposes
# NOTE: change this to 100000 for full replication
pmx <- createParamMx(nexp=50, h0=TRUE)

# ----- Type I Error (H0 true) -----

pmx[, "eff"] <- 0

# ANOVA
mcRun("fitanova", mcr.outfile="fitanova.h0.wsbi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=TRUE)
mcRun("fitanova", mcr.outfile="fitanova.h0.wsbi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=TRUE)
mcRun("fitanova", mcr.outfile="fitanova.h0.wswi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=FALSE)
mcRun("fitanova", mcr.outfile="fitanova.h0.wswi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=FALSE)

# lmer models, random intercept only
mcRun("fitlmer", mcr.outfile="fitlmerRI.h0.wsbi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=TRUE, ri.only=TRUE)
mcRun("fitlmer", mcr.outfile="fitlmerRI.h0.wsbi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=TRUE, ri.only=TRUE)
mcRun("fitlmer", mcr.outfile="fitlmerRI.h0.wswi.12.csv",
```

```

      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=FALSE, ri.only=TRUE)
mcRun("fitlmer", mcr.outfile="fitlmerRI.h0.wswi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=FALSE, ri.only=TRUE)

# lmer models, random intercept only, p-values from MCMC
mcRun("fitlmer.mcmc", mcr.outfile="fitlmerRImcmc.h0.wsbi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=TRUE)
mcRun("fitlmer.mcmc", mcr.outfile="fitlmerRImcmc.h0.wsbi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=TRUE)
mcRun("fitlmer.mcmc", mcr.outfile="fitlmerRImcmc.h0.wswi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=FALSE)
mcRun("fitlmer.mcmc", mcr.outfile="fitlmerRImcmc.h0.wswi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=FALSE)

# lmer models, no random correlation
mcRun("fitnocorr", mcr.outfile="fitnocorr.h0.wsbi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=TRUE)
mcRun("fitnocorr", mcr.outfile="fitnocorr.h0.wsbi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=TRUE)
mcRun("fitnocorr", mcr.outfile="fitnocorr.h0.wswi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=FALSE)
mcRun("fitnocorr", mcr.outfile="fitnocorr.h0.wswi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=FALSE)

# lmer models, no random correlation, p-values from MCMC
mcRun("fitnocorr.mcmc", mcr.outfile="fitnocorrMCMC.h0.wsbi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=TRUE)
mcRun("fitnocorr.mcmc", mcr.outfile="fitnocorrMCMC.h0.wsbi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=TRUE)
mcRun("fitnocorr.mcmc", mcr.outfile="fitnocorrMCMC.h0.wswi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=FALSE)
mcRun("fitnocorr.mcmc", mcr.outfile="fitnocorrMCMC.h0.wswi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=FALSE)

# lmer models, no within-unit slopes
mcRun("fitrsonly", mcr.outfile="fitrsonly.h0.wsbi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=TRUE)
mcRun("fitrsonly", mcr.outfile="fitrsonly.h0.wsbi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=TRUE)
mcRun("fitrsonly", mcr.outfile="fitrsonly.h0.wswi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=FALSE)
mcRun("fitrsonly", mcr.outfile="fitrsonly.h0.wswi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=FALSE)

# lmer models, maximal
mcRun("fitlmer", mcr.outfile="fitlmerRS.h0.wsbi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=TRUE)
mcRun("fitlmer", mcr.outfile="fitlmerRS.h0.wsbi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=TRUE)
mcRun("fitlmer", mcr.outfile="fitlmerRS.h0.wswi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=FALSE)
mcRun("fitlmer", mcr.outfile="fitlmerRS.h0.wswi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=FALSE)

# ----- Power (H0 false) -----

pmx[, "eff"] <- .8

# ANOVA
mcRun("fitanova", mcr.outfile="fitanova.h1.wsbi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=TRUE)
mcRun("fitanova", mcr.outfile="fitanova.h1.wsbi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=TRUE)

```

```

mcRun("fitanova", mcr.outfile="fitanova.hl.wswi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=FALSE)
mcRun("fitanova", mcr.outfile="fitanova.hl.wswi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=FALSE)

# lmer models, random intercept only
mcRun("fitlmer", mcr.outfile="fitlmerRI.hl.wswi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=TRUE, ri.only=TRUE)
mcRun("fitlmer", mcr.outfile="fitlmerRI.hl.wswi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=TRUE, ri.only=TRUE)
mcRun("fitlmer", mcr.outfile="fitlmerRI.hl.wswi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=FALSE, ri.only=TRUE)
mcRun("fitlmer", mcr.outfile="fitlmerRI.hl.wswi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=FALSE, ri.only=TRUE)

# lmer models, random intercept only, p-values from MCMC
mcRun("fitlmer.mcmc", mcr.outfile="fitlmerRImcmc.hl.wswi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=TRUE)
mcRun("fitlmer.mcmc", mcr.outfile="fitlmerRImcmc.hl.wswi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=TRUE)
mcRun("fitlmer.mcmc", mcr.outfile="fitlmerRImcmc.hl.wswi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=FALSE)
mcRun("fitlmer.mcmc", mcr.outfile="fitlmerRImcmc.hl.wswi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=FALSE)

# lmer models, no random correlation
mcRun("fitnocorr", mcr.outfile="fitnocorr.hl.wswi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=TRUE)
mcRun("fitnocorr", mcr.outfile="fitnocorr.hl.wswi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=TRUE)
mcRun("fitnocorr", mcr.outfile="fitnocorr.hl.wswi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=FALSE)
mcRun("fitnocorr", mcr.outfile="fitnocorr.hl.wswi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=FALSE)

# lmer models, no random correlation, p-values from MCMC
mcRun("fitnocorr.mcmc", mcr.outfile="fitnocorrMCMC.hl.wswi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=TRUE)
mcRun("fitnocorr.mcmc", mcr.outfile="fitnocorrMCMC.hl.wswi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=TRUE)
mcRun("fitnocorr.mcmc", mcr.outfile="fitnocorrMCMC.hl.wswi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=FALSE)
mcRun("fitnocorr.mcmc", mcr.outfile="fitnocorrMCMC.hl.wswi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=FALSE)

# lmer models, maximal
mcRun("fitlmer", mcr.outfile="fitlmerRS.hl.wswi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=TRUE)
mcRun("fitlmer", mcr.outfile="fitlmerRS.hl.wswi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=TRUE)
mcRun("fitlmer", mcr.outfile="fitlmerRS.hl.wswi.12.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=FALSE)
mcRun("fitlmer", mcr.outfile="fitlmerRS.hl.wswi.24.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=FALSE)

```

## Model-selection runs

```

library(simgen)

# nexpt set to small value for testing; change to 100000 for full runs
pmx <- createParamMx(nexpt=10, h0=TRUE)

```

```

# ----- Type I Error (H0 true) -----

pmx[, "eff"] <- 0

# WSBI design
mcRun("fitstepwise", mcr.outfile="fitstepwise.h0.wsbi.12.MS.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=TRUE,
      mf=modSpace(wsbi=TRUE))
mcRun("fitstepwise", mcr.outfile="fitstepwise.h0.wsbi.24.MS.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=TRUE,
      mf=modSpace(wsbi=TRUE))

# WSWI design
# Note: fitstepwise does both forward and backward selection
# forward subject slope first & backward item slope first
mf <- modSpace(FALSE) # define the models to be tested
mcRun("fitstepwise", mcr.outfile="fitstepwise.h0.wswi.12.FSBI.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=FALSE,
      mf=c(mf[3], mf[[2]][1], mf[1]))
mcRun("fitstepwise", mcr.outfile="fitstepwise.h0.wswi.24.FSBI.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=FALSE,
      mf=c(mf[3], mf[[2]][1], mf[1]))
# forward item slope first & backward subject slope first
mcRun("fitstepwise", mcr.outfile="fitstepwise.h0.wswi.12.FIBS.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=FALSE,
      mf=c(mf[3], mf[[2]][2], mf[1]))
mcRun("fitstepwise", mcr.outfile="fitstepwise.h0.wswi.24.FIBS.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=FALSE,
      mf=c(mf[3], mf[[2]][2], mf[1]))
# 'best path' forward
mcRun("fitstepwise.bestpath", mcr.outfile="fitstepwise.h0.wswi.12.FB.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=FALSE,
      forward=TRUE)
mcRun("fitstepwise.bestpath", mcr.outfile="fitstepwise.h0.wswi.24.FB.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=FALSE,
      forward=TRUE)
# 'best path' backward
mcRun("fitstepwise.bestpath", mcr.outfile="fitstepwise.h0.wswi.12.BB.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=FALSE,
      forward=FALSE)
mcRun("fitstepwise.bestpath", mcr.outfile="fitstepwise.h0.wswi.24.BB.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=FALSE,
      forward=FALSE)

# ----- Power (H0 false) -----

pmx[, "eff"] <- .8

# WSBI design
mcRun("fitstepwise", mcr.outfile="fitstepwise.h1.wsbi.12.MS.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=TRUE,
      mf=modSpace(wsbi=TRUE))
mcRun("fitstepwise", mcr.outfile="fitstepwise.h1.wsbi.24.MS.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=TRUE,
      mf=modSpace(wsbi=TRUE))

# WSWI design
# Note: fitstepwise does both forward and backward selection
# forward subject slope first & backward item slope first
mf <- modSpace(FALSE) # define the models to be tested
mcRun("fitstepwise", mcr.outfile="fitstepwise.h1.wswi.12.FSBI.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=FALSE,
      mf=c(mf[3], mf[[2]][1], mf[1]))
mcRun("fitstepwise", mcr.outfile="fitstepwise.h1.wswi.24.FSBI.csv",

```

```

        mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=FALSE,
        mf=c(mf[3], mf[[2]][1], mf[1]))
# forward item slope first & backward subject slope first
mcRun("fitstepwise", mcr.outfile="fitstepwise.h1.wswi.12.FIBS.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=FALSE,
      mf=c(mf[3], mf[[2]][2], mf[1]))
mcRun("fitstepwise", mcr.outfile="fitstepwise.h1.wswi.24.FIBS.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=FALSE,
      mf=c(mf[3], mf[[2]][2], mf[1]))
# 'best path' forward
mcRun("fitstepwise.bestpath", mcr.outfile="fitstepwise.h1.wswi.12.FB.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=FALSE,
      forward=TRUE)
mcRun("fitstepwise.bestpath", mcr.outfile="fitstepwise.h1.wswi.24.FB.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=FALSE,
      forward=TRUE)
# 'best path' backward
mcRun("fitstepwise.bestpath", mcr.outfile="fitstepwise.h1.wswi.12.BB.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=12, wsbi=FALSE,
      forward=FALSE)
mcRun("fitstepwise.bestpath", mcr.outfile="fitstepwise.h1.wswi.24.BB.csv",
      mcr.xdatFnc="mkDf", mcr.varying=pmx, nitem=24, wsbi=FALSE,
      forward=FALSE)

```