Simulation of Factorial Mixed-Model Designs in R: The mixedDesign() Function

[Version 0.6.2 (September 2012)]

Sven Hohenstein & Reinhold Kliegl

University of Potsdam, Potsdam, Germany

Running head: mixedDesign()

Address for correspondence:

Sven Hohenstein or Reinhold Kliegl

Department of Psychology, University of Potsdam

Karl-Liebknecht-Str. 24-25, 14476 Potsdam, Germany

Email: sven.hohenstein@uni-potsdam.de and kliegl@uni-potsdam.de

Tel.: +493319772370, -2868; fax: +493319772793

Abstract

We present a function, which easily generates random data for a factorial experimental design with a freely specifiable set of between- and within-subject factors, meeting constraints on means and standard deviations for the design cells as well as of matrices of expected correlations between levels of within-subject factors (i.e., repeated measures). The function generates data assuming ideal conditions and provides quick tests of consequences of violating various assumptions of the general linear model for a given design. It was programmed in the in the *R* environment for statistical computing (R Development Core Team, 2013). We illustrate the function for a priming experiment comprising a design with age (4 levels) as between-subject factor with 10 subjects in each of the four groups and with prime (2) and frequency (3) as within-subject factors. The function is available at the Potsdam Mind Research Repository (PMR^2) website located at http://read.psych.uni-potsdam.de/pmr2/.

Simulation of Factorial Mixed-Model Designs in R: The mixedDesign() Function

We present a function, which easily generates random data for a factorial experimental design freely specifiable set of between- and within-subject factors. It allows the user to specify matrices of expected means and standard deviations for the design cells as well as matrices of expected correlations between levels of within-subject factors (i.e., repeated measures). Furthermore, there is also the option of specifying differential loss of data for the design cells within the same call to the function. Thus, the function generates data under ideal conditions and provides quick tests of consequences of violating various assumptions of the general linear model for a given design. It was programmed in the in the *R* environment for statistical computing (R Development Core Team, 2013). Furthermore, we provide another function allowing the extraction of partial effects and that way facilitating – for example –insights in complex models including numerous variables and interactions. All functions presented in this paper are available at the Potsdam Mind Research Repository (PMR^2) website located at http://read.psych.uni-potsdam.de/pmr2/.

Description of Function Arguments

We describe the function arguments of *mixedDesign()* in the context of a hypothetical data set of a psychological experiment. Consider a classical priming study in which a word with specified word frequency is presented for a subliminal duration before a target word, either related or unrelated to the prime, appears on screen. The subject’s task is to answer whether the target word is the name of an animal or not. Response times are recorded. Target words are from one of three frequency conditions (factor: frequency with three levels “low”, “medium”, and “high”) and the words are primed with a related or unrelated word (factor: prime with two levels “related” and “unrelated”) such that each subject is presented with each of the six possible combinations of these two factors. Since all subjects are tested under all conditions these two factors are *within-subject factors*.

The function allows also the specification of *between-subject factors*. For these factors, each subject is tested under exactly one level of a factor or one combination of levels of several between-subject factors. In our example, we assume that there are four age groups (factor: age with four levels "children", "teenagers", "young adults", and "older adults"). Moreover, we may want to balance the design with respect to gender (factor: gender with two levels "male" and "female"). Thus, each subject belongs to exactly one of these eight groups. Finally, we need to specify the number of subjects in each cell of the between-subject factor design of this experiment. For a balanced design, we may have recruited *n* = 10 subjects for each of the 4 × 2 between-subject factor levels, yielding a total of *N* = 80 subjects.

Input

In the following we describe how such a basic design is coded in the input parameters of the function. Table 1 provides an overview of all input parameters and their usage.

Table 1. Input parameters of the *mixedDesign* function.

Parameter	Meaning	Input		
		mode and range	structure	default
B	between-subject factors	integer > 1	vector	NULL
W	within-subject factors	integer > 1	vector	NULL
n	number of subjects/group	integer ≥ 2	single value	prod(W)
M	cell means	real number	matrix; vector; single value[a]	0
SD	standard deviations	real number	matrix; vector; single value[a]	1
R	correlations	real number, [−1,1]	list of matrices[b] and/or single values[b]; matrix[c]; single value[c]	0
miss	missing data	real number, [0,1]	matrix; vector; single value[a]	0
family	distribution family	string, {"gaussian", "gamma", "beta"}	single string	"Gaussian"
long	output format	boolean	single value	FALSE
empirica l	empirical or population data	boolean	single value	TRUE

[a] Vectors are legal for pure within-subject designs; single values are applied to all cells.
[b] One matrix/one value for the correlations in one group.
[c] Single matrices/single values are applied to all groups.

B – between-subject factors. B is a vector specifying the number of factors levels for each between-subject factor. The length of B equals the number of between-subject factors. If a design includes, for example, two between-subject factors with 2 (gender) and 4 (age group) levels, respectively, B must be specified as `B = c(2, 4)`. By default, no between-subject factors are specified (`B = NULL`). Obviously, the minimum number of factor levels is 2; smaller numbers will be removed from the input vector. For practical purposes, we will use only one between-subjects factor (age) in the example illustrated in this manuscript.

W – Within-subject factors. W is a vector specifying the number of factor levels to each within-subject factor. The usage of W is equivalent to B. Correspondingly, the default includes no within-subject factors (`W = NULL`), and the minimum number of levels for a factor equals two. Since neither between- nor within-subject factors are specified by default, the user needs to specify at least one factor (between or within); otherwise, NULL will be returned by the function.

n – Number of subjects in prod(B) cells. The number of combinations of between-subject factor levels is the product of all values in B. In R the command `prod(B)` can be used to calculate the number of distinct groups. The number of subjects in each of these groups is specified with the single integer *n*.[1] In a balanced design each group consists of the same number of subjects. Thus, the value of *n* is equal for all groups. Note that *n* is not the total number of subjects *N* in the design; the total number of subjects is the product of between-factor levels and the number of subjects within each group: `N = n * prod(B)`. The default value of n is prod(w), the minimum value is 2.

M – Matrix of means. The total number of cells of an experimental design is the result of crossing all between- and within-subject factors in a matrix. The number of between-subject factor levels equals prod(B) and the number of within-subject factor levels equals prod(W). Hence, in total an experimental design consists of prod(B) times prod(W) conditions (cells). The means of these cells can be assembled in a matrix with prod(B) rows and prod(W) columns producing a prod(B) × prod(W) matrix of means. In our example, the experimental design includes two within-subject factors, frequency with 3 and prime with 2 levels, and one between-subject factor age group with 4 levels. The resulting $(4) \times (3 \cdot 2)$ matrix has $(4) \cdot (3 \cdot 2) = 24$ cells. In a pure within-subject designs, M can be a vector of size prod(W) as an alternative to a 1 × prod(W) matrix. If M is specified as a single real

[1] Note that the parameter *n* can also be interpreted as the number of *observations* per condition. In principle, it is possible to generate multiple observations per subject by specifying subjects as between-subject factor.

number, e.g., `M = 5.3`, this value will be used as the mean for all cells of the experimental design. If M is not specified, a default value of 0 is used for all cells.

SD – Matrix of standard deviations. The specification of the standard deviations for all cells is equivalent to the one of the cell means M. SD is a prod(B) × prod(W) matrix including the standard deviation for each cell of the experimental design. Correspondingly, the number of between-subject factor level combinations and within-subject factor level combinations is equivalent to the matrix's rows and columns, respectively. In pure within-subject designs, SD can be a vector of size prod(W) as an alternative to a 1 × prod(W) matrix. If the standard deviations in all cells should have the same value, it is possible to specify SD as a single real number. If SD is not specified, a default value of 1 is used for all cells. If both M and SD are left unspecified, the data of each cell conform to the standard normal distribution (mean of zero, standard deviation of one).

R – List of correlation matrices for levels of within-subject factors. With the list *R* it is possible to specify a matrix of correlations between levels of within-subject factors. In principle, this matrix can be different for each combination of between-subject factors, that is, for each distinct group of *n* subjects. We use the *mvrnom* function in the MASS package (Venables & Ripley, 2002), based on a technique described by Ripley (1987), to this end. This list is of length prod(B); its objects are the correlation matrices for the within-subject factor level combinations. Thus, the appropriate format is a prod(W) × prod(W) square matrix, but the following restrictions apply (irrespective of user input): (1) All values in the diagonal of the matrix will be fixed at 1; all values in the off-diagonal must be between −1 and +1. The function stops if any value of *R* is outside this range. (2) Corresponding values above and below the diagonal must be identical; in case of discrepancy values below the diagonal are used. For usability purposes, the function uses only the values specified in the lower triangular part of matrix *R*. Instead of a matrix, single values can be specified as list objects. Actually, the list can consist of both matrices and single numbers. The sphericity assumption of repeated-measures ANOVA implies that all correlations are equal. A single correlation value (i.e., a real single number between −1 and +1) ensures that this assumption is met. As an alternative to a list with separate matrices for each cell of the between-factor design, *R* can be a single matrix or a single value, which will be applied to all groups. Finally, the default value of *R* is 0, that is, variables are not correlated, if the parameter is not set in the function call.

miss – Matrix of proportions of data loss in design cells. With the specification of the *miss* matrix, we simulate random data loss (NAs) in the cells. The specification of the data loss matrix is equivalent to those for cell means (M) and standard deviations (SD). *miss* is a prod(B) × prod(W) matrix, assembling the proportion of data loss for each cell of the experimental design. The default value for data loss proportions is 0, that is, there is no data loss. In pure within-subject designs, loss can be a vector of size prod(W) as an alternative to a 1 × prod(W) matrix. If data loss is specified as a single real number between 0 and 1, the same proportion of missing data will be applied to all cells. Note that a specification of *miss* will cause some departure from the specified means, standard deviations, and correlations.

long – Data frame in long format. The format of the output data frame is specified with the Boolean variable *long*. If *long* is TRUE, the data frame is returned in the long format, otherwise (`long = FALSE`) it is returned in the wide format. In the wide format, all (within-subject) values of a subject are in a single line. Thus, the number of lines in the data frame equals the total number of subjects N. By default, long is set to FALSE producing a data frame in wide format. Each measurement is in a single line of the data frame if it is returned in the long format. This is the traditional arrangement for most kinds of inference statistics (in *R*). Note the same data can be generated first in wide and then in long format, if *set.seed()* is given the same argument before the two function calls.

family – Distribution family. The distribution family is specified with the family parameter. It must be one of the three possible strings "Gaussian", "gamma", and "beta". The most common and widely used distribution is the Gaussian distribution (also known as normal distribution); it ranges from minus infinity to plus infinity and is the underlying substructure of most inferential statistics. Since any real number is (at least with a very small probability) part of any normal distribution, negative values are a theoretical output of the *mixedDesign* function if the Gaussian distribution family is specified. For the generation of real numbers bound between zero and plus infinity, for example reaction times, the gamma distribution can be chosen. In contrast, the beta distribution family is the family of choice for the simulation of probabilities (among others), since it ranges from zero to one. Furthermore, binary dependent variables can be generated with the beta distribution and a cut-off at 0.5. The function produces Gaussian distributed data meeting the specified means, standard deviation, and correlations exactly. Note that the statistics of generated gamma and beta distributions slightly differ from the input parameters. For more details, see the constraints section below.

empirical – Type of random values. If *empirical* is set to TRUE, the random values will exactly match the specified properties (at least for Gaussian data); this is the default setting. Conversely, if *empirical* is set to FALSE, values will be drawn randomly from a distribution with the specified properties. Thus, especially for small *n*, differences may be quite large. Of course, for the simulation of replications of an experiment, *empirical* must be set to FALSE.

Output

The function returns a data frame containing the simulated data in long or wide format (see input parameter "long"). These data are sampled randomly from a normal distribution. The names of the columns containing the between-subject factor information start with "B_" (for **B**etween) followed by an uppercase letter. The first and second between-subject factors are named "B_A" and "B_B", respectively. The character strings in these columns contain information about the factor levels, e.g., A1, A2 in "B_A" and B1, B2 in "B_B". The next column, "id", contains the subject number. The remaining columns depends on whether data frame is returned in long or wide format.

Long format. If the data frame is in the long format, the next columns contain information about the within-subject factors in a format similar the one of the between-.subject-factors, but with an initial "W_" (for **W**ithin) and small letters ("W_a"; a1, a2, etc.). The last column "DV" contains the values of the virtual "dependent variable". For example, a simple design with one between-subject and one within subject factor with two levels each and *n* = 3 subjects in each group [`mixedDesign(B = 2, W = 2, n = 3)`] may yield the following output:

```
   B_A id W_a         DV
1   A1  1  a1  0.6696383
2   A1  1  a2  0.9407007
3   A1  2  a1 -1.1494899
4   A1  2  a2  0.1095734
5   A1  3  a1  0.4798516
6   A1  3  a2 -1.0502742
7   A2  4  a1 -1.1191060
8   A2  4  a2 -0.2844910
9   A2  5  a1  0.8059295
10  A2  5  a2 -0.8269287
11  A2  6  a1  0.3131765
12  A2  6  a2  1.1114197
```

Wide format. If the data frame is in the wide format, the columns following the "id" column contain the prod(W) measures that were recorded from each subject. The names of the columns (measures) are combinations of the within-subject factor levels starting with "W_". For example, the column containing the data for the third level of the first within-subject factor and the first level of the second within-subject factor is named "W_a3_b1". In wide format, the returned data frame for the example given above contains one line per subject, but two measures per line.

```
  B_A id        W_a1       W_a2
1  A1  1  0.6696383  0.9407007
2  A1  2 -1.1494899  0.1095734
3  A1  3  0.4798516 -1.0502742
4  A2  4 -1.1191060 -0.2844910
5  A2  5  0.8059295 -0.8269287
6  A2  6  0.3131765  1.1114197
```

Additional usage notes (constraints)

The agreement between specified input and function output depends on two additional constraints. In our experience, these constraints rarely matter in our applications, which are mostly related to explore a particular experimental design or to teaching research design and statistics.

Number of subjects. The minimum number of subjects *n* within each group (combination of between-subject factor levels) is 2; it is not possible to simulate random data with 1 subject. The number of subjects *n* relative to the number of within-subject cells, that is prod(W) is also critical for the precision with which function output reproduces the specified input parameters. Basically, the minimum number of subjects should not be smaller than the number of measures for each subject, that is not smaller than the product of within-subject factor levels prod(W), i.e., n >= prod(W). If n > prod(W), the output will be as described. If n = prod(W), the function will work but means, standard deviations, and correlations of the returned data will slightly differ from the desired ones. The function will also run (with a warning message) for 1 < n < prod(W), but the obtained means, standard deviations, and correlations will differ from the specified ones. This applies in particular for small values of n. In this case, the function will internally increase the number of subjects for data generation and subsequently randomly delete subjects to reach the desired n.

Distribution families. If the Gaussian distribution family is specified, the function generates data whose statistics exactly meet the specified means, standard deviations, and correlation (if n is greater than prod(W)). However, the generated gamma and beta distributions will slightly differ from the specifications. These deviations are non-systematic but random. Differences between generated and specified parameters decrease with increasing number of subjects *n*. The classical Pearson product-moment correlation, the default of the *cor* function, is most informative for normally distributed data, but it is hardly convenient for gamma and beta distributions, especially if the distribution is strongly skewed. Instead, in such a case the Spearman rank correlation is the appropriate statistic for gamma and beta distributions. It can be obtained if the *cor* function is called with the parameter `method = “spearman”`. Spearman correlations will be closer to input values than Pearson correlations.

An Application Example: Generation of a Data Set

To simulate a data set for the experimental design described above, including the two within-subject factors frequency (3 levels) and prime type (2 levels) and, leaving out gender, one between-subject factor (age group; 4 levels), we must decide on a set of means, standard deviations, and correlations. The R program for this example is provided in the Appendix.

To demonstrate some likely trends in the data, the following matrix *mean.mat* of cell means (reaction times in ms) is created (see also Table 2):

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]  950  950  950 1050 1050 1050
[2,]  850  800  750  950  900  850
[3,]  800  700  600  900  800  700
[4,]  830  730  630  930  830  730
```

The four lines correspond to the age groups and the six columns correspond to the crossing of prime type (2 levels) with frequency (3 levels). Reaction times decrease from childhood to adulthood and increasef from adulthood to old age (notice changes across lines 1 to 4). Furthermore, reaction times decrease with prime frequency (columns 1 to 3 and 4 to 6) from adolescence on with its maximum for young and older adults; this trend is not specified for children. Obviously, such a pattern represents an interaction of the factors age group and frequency. Finally, targets preceded by related primes (columns 1 to 3) produce shorter answer latencies than unrelated primes (columns 4 to 6). To simplify matters a standard deviation of 60 is chosen for all conditions. The number of subjects in each of the four between-subject groups is set to twenty resulting in a total number of 80 virtual subjects. A data frame with the discussed specifications is generated with the following command:

```
data <- mixedDesign(B = 4, W = c(2, 3), M = mean.mat, SD = 60, n = 20, long = TRUE)
```

Please note that mean.mat is the cell mean matrix specified before (see above). By default, there will be no data loss (i.e., loss = 0) and the within-subject factor will not be correlated (i.e., R = 0). The created data frame is in the long format with one dependent variable per line.

The function returns a data frame with Gaussian distributed values meeting the specified means and the standard deviation exactly. Since no correlation matrix was specified, the correlations of all within-subject factor levels in each between-subject factor level are zero. Note that since values are generated randomly from a normal distribution the individual values will differ each time the function is used. This has no impact on the degree to which the specified parameters are met. For a simulation with varying means and standard deviations across runs, set the argument empirical =FALSE.

Statistical Analysis

The R program in the Appendix shows how the default variable names are replaced with names that are meaningful for the present example. Next we compute the table of means (along with number of observations, standard deviations, and standard errors) for the full factorial design (see Table 2).

Table 2. Example dataset of reaction times in milliseconds separated for the between-subject factor age group and the within-subject factors prime and frequency.

	Related prime			Unrelated prime		
Age group	Low frequency	Medium frequency	High frequency	Low frequency	Medium frequency	High frequency
Children	950	950	950	1050	1050	1050
Teenagers	850	800	750	950	900	850
Young adults	800	700	600	900	800	700
Older adults	830	730	630	930	830	730

Means and standard deviations match exactly the specification. A plot of the full factorial design is given in Figure 1. It is also clear that the three-factor interaction displayed in this figure is unlikely to be significant, but the frequency effect is clearly different for age groups. Next we subject the data to an age(4) x prime(2) x frequency(3) mixed-model ANOVA. As expected, aside from three significant main effects, there is a significant interaction of age and frequency; F(6, 152) = 25.5, MSe=3600, p < .01. The plot of this interaction is provided in Figure 2.

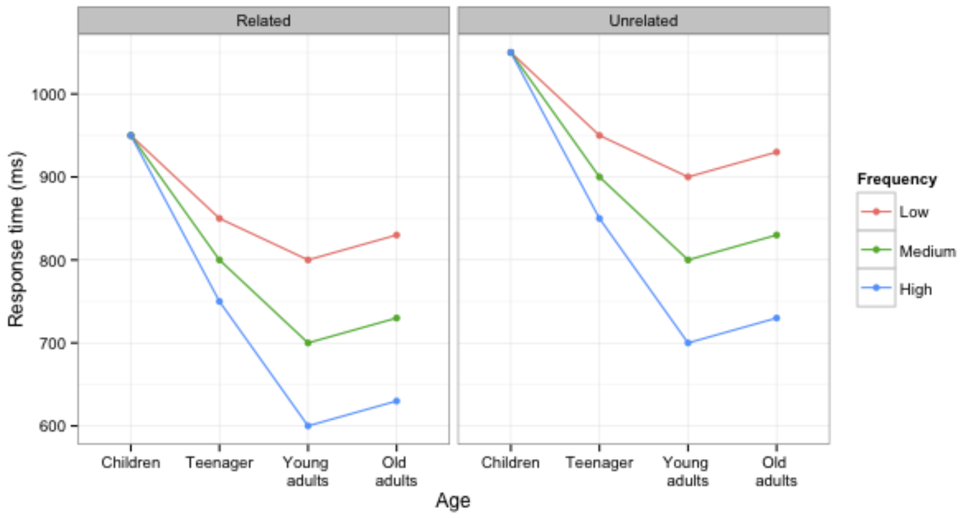

Figure 1. Simulated response times (RT) of four virtual age groups (B_A: A1-A4), three frequency classes (W_b: b1-b3), and two prime conditions (W_a: a1 and a2). Error bars are 2 SEs; they are only valid for comparisons between age groups (i.e., the between-subject factor).

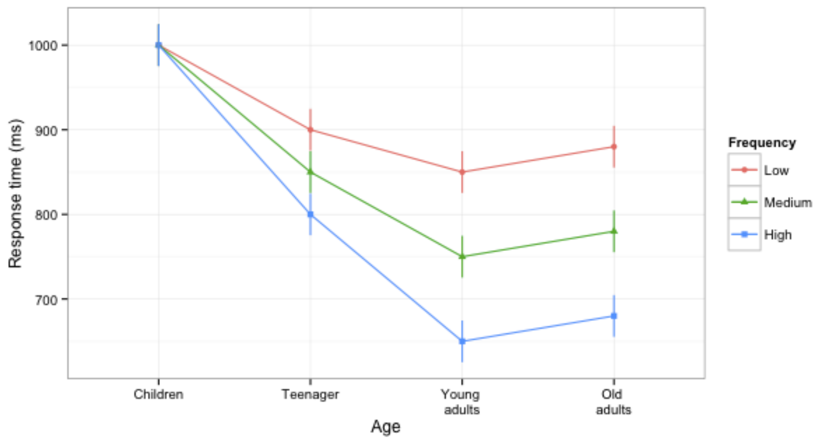

Figure 2. Age group (B_A: A1-A4) by frequency (W_b: b1-b3) interaction for simulated response times. Error bars are 2 SEs; they are only valid for comparisons between age groups (i.e., the between-subject factor).

Conclusion

The data generated with the *mixedDesign()* function can be used for an arbitrarily complex factorial design. Typical applications are power analyses, consequences of violations of model assumptions (e.g., sphericity; missing data) for inferential statistics. The analysis of an experiment can be carried out before data are actually collected. It is possible to find the minimum number of subjects necessary to achieve significance for a specific main effect of interaction. Last not least, the function is useful for teaching basic statistics.

References

R Development Core Team. (2013). *R: A language and environment for statistical computing* [Computer software]. Vienna, Austria: R Foundation for Statistical Computing.

Ripley, B. D. (1987). *Stochastic Simulation*. New York: Wiley.

Venables, W. N., & Ripley, B. D. (2002) *Modern Applied Statistics with S*. Fourth Edition. New York: Springer.

Wickham, H. (2009). ggplot2: Elegant graphics for data analysis. New York: Springer.

ACKNOWLEDGEMENT

Sven Hohenstein and Reinhold Kliegl, Department of Psychology, University of Potsdam, Potsdam, Germany. This research was supported by Deutsche Forschungsgemeinschaft (grants no. KL 955/6-1) as part of Research Group 868 “Computational Modeling of Behavioral, Cognitive, and Neural Dynamics”. We thank Marco Rusconi and Jochen Laubrock for helpful suggestions and discussions. Correspondence concerning this article should be addressed to Sven Hohenstein or Reinhold Kliegl, Department of Psychology, University of Potsdam, Karl-Liebknecht-Str. 24-25, 14476 Potsdam, Germany. Emails: sven.hohenstein@uni-potsdam.de, kliegl@uni-potsdam.de

APPENDIX: R PROGRAM FOR EXAMPLE

```
# Illustration of the mixedDesign() function

library(MASS)
library(plyr)
library(ggplot2)

source("functions/mixedDesign.v0.6.2.R")

# Cell means (between-subject factor levels across rows; within-subject factor levels across columns)
mean.mat <- matrix(c(950, 950,  950, 1050, 1050, 1050,
                               850, 800,  750,  950,  900,  850,
                               800, 700,  600,  900,  800,  700,
                               830, 730,  630,  930,  830,  730), nrow=4, ncol=6, byrow=TRUE)

# Call function
set.seed(1)
data <- mixedDesign(B = 4, W = c(2, 3), M = mean.mat, SD = 60, n = 20, long = TRUE)

# Rename variables and levels
names(data) <- c("Age", "Subj", "Prime", "Frequency", "RT")
levels(data$Age) <- c("Children", "Teenager", "Young \nadults", "Old \nadults")
levels(data$Prime) <- c("Related", "Unrelated")
levels(data$Frequency) <- c("Low", "Medium", "High")

# Compute table of means for full factorial
table <- ddply(data, .(Age, Frequency, Prime), summarise, N=length(RT), M=mean(RT),
                                                         SD=sd(RT), SE=SD/sqrt(N) )
# Note: SE's are valid only for comparisons between age groups (between-subject factor)

# Plot table
qplot(data=table, x=Age, y=M, ylab = "Response time (ms)", group=Frequency, colour=Frequency,
             facets = . ~ Prime, geom=c("point", "line")) +
             geom_errorbar(aes(ymax=M+2*SE, ymin=M-2*SE), width=0) + theme_bw()

# Age(4) x Prime (2) x Freq(3) mixed-model ANOVA
summary(aov(RT ~ Age*Frequency*Prime + Error(Subj/(Frequency*Prime)), data=data))


# Significant interaction for Age x Frequency
table.a_f <- ddply(data, .(Age, Frequency), summarise, N=length(RT), M=mean(RT),
                                                       SD=sd(RT), SE=SD/sqrt(N) )

qplot(data=table.a_f, x=Age, y=M, ylab = "Response time (ms)",
             group=Frequency, colour=Frequency, shape=Frequency,
             geom=c("point", "line")) +
             geom_errorbar(aes(ymax=M+2*SE, ymin=M-2*SE), width=0) + theme_bw()
```