Journal of Statistical Software

August 2016, Volume 72, Code Snippet 2. *doi: 10.18637/jss.v072.c02*

Mixture Experiments in R Using mixexp

John Lawson
Brigham Young University

Cameron Willden
W. L. Gore and Associates

Abstract

This article discusses the design and analysis of mixture experiments with R and illustrates the use of the recent package **mixexp**. This package provides functions for creating mixture designs composed of extreme vertices and edge and face centroids in constrained mixture regions where components are subject to upper, lower and linear constraints. These designs cannot be created by other R packages. **mixexp** also provides functions for graphical display of models fit to data from mixture experiments that cannot be created with other R packages.

Keywords: contour plot, effect plot, simplex lattice design, simplex centroid design, extreme vertices design, constrained mixture experiments, D-optimality.

1. Introduction

A mixture experiment occurs when the response variable, y, is a function of the relative proportions of components (x_i, $i = 1, \ldots q$) in a mixture, rather than a function of the total amount of each component. Since the proportions of components in a mixture must add to one, i.e., $\sum_{i=1}^{q} x_i = 1.0$, the experimental region for mixture experiments is constrained, and traditional factorial or response surface designs are not appropriate. A comprehensive reference on the designs and methods of data analysis useful for mixture experiments is the book by Cornell (2002). Mixture experiments are widely used today in formulation experiments, blending experiments, and marketing choice experiments where the goal is to determine the most preferred attribute composition of a product at a given price (Ragavarao, Wiley, and Chitturi 2011).

Popular commercial statistical software such as **SAS** ADX (SAS Institute Inc. 2010a), **Minitab** (Minitab Inc. 2010), **JMP** (SAS Institute Inc. 2010b), **Design-Ease** (Stat-Ease, Inc. 2010), **Statgraphics** (StatPoint Technologies 2010), and **Modde** (Umetrics an MKS Company 2014) all have extensive facilities for the design and analysis of mixture experiments. Piepel (1997)

gives an extensive review of commercial software with mixture experiment capabilities.

The `lm` funcion in base R (R Core Team 2016) can be used to fit models to mixture experiments. There are functions in R packages **AlgDesign** (Wheeler 2014) and **qualityTools** (Roth 2016) that allow the user to create Simplex Lattice Designs and Simplex Centroid Designs for unconstrained mixture experiments. The package **mixexp** (Lawson 2016) adds the ability to create designs comprised of extreme vertices of constrained experimental regions, possibly augmented with edge and face centroids, and other interior points in the simplex.

The R package **qualityTools** allows a user to visualize the relationship between the response and mixture components by making ternary contour plots and 3D wireframe plots of Scheffé mixture model in three components over the unconstrained simplex region. The package **mixexp** expands these capabilities by allowing the user to: 1) make contour plots of a greater variety of equations, 2) create contour plots within constrained regions bounded by pseudo components, 3) make contour plots of models fit to more than three components by holding one or more component(s) constant and, 4) create response trace plots along the Cox (1971) or Piepel (1982) directions through constrained regions in higher dimensional spaces.

The purpose of this article is to show how to design and analyze mixture experiments in R, and to simultaneously illustrate the basic features of the package **mixexp**. Details of the arguments and options for all the functions in **mixexp** can be found in Lawson (2016).

2. Models for the analysis of mixture experiments

Due to the fact that mixture components must sum to 1, the linear model that is normally written as $y = \beta_0 + \sum_{i=1}^{q} \beta_i x_i + \epsilon$ is redundant for data from mixture designs. Scheffé (1958) wrote the linear model for mixtures as:

$$y = \sum_{i=1}^{q} \beta_i x_i + \epsilon. \tag{1}$$

The coefficients in Equation 1 have physical meaning. β_i is the expected response at the vertex where $x_i = 1.0$. Scheffé (1958) wrote the quadratic model for mixture experiments as:

$$y = \sum_{i=1}^{q} \beta_i x_i + \sum_{i=1}^{q-1} \sum_{j=i+1}^{q} \beta_{ij} x_i x_j + \epsilon. \tag{2}$$

In Equation 2 the β_{ij} coefficients indicate the amount of quadratic curvature along the edge of the simplex region consisting of binary mixtures of x_i and x_j. Due to the bounded experimental region for mixture experiments, the quadratic model for mixture experiments may not be flexible enough to represent the results of some experiments.

The full cubic model represented in Equation 3 and the special-cubic model in Equation 4 are popular alternatives to the quadratic model.

$$y = \sum_{i=1}^{q} \beta_i x_i + \sum_{i=1}^{q-1} \sum_{j=i+1}^{q} \beta_{ij} x_i x_j + \sum_{i=1}^{q-1} \sum_{j=i+1}^{q} \delta_{ij} x_i x_j (x_i - x_j) + \sum_{i=1}^{q-2} \sum_{j=i+1}^{q-1} \sum_{k=j+1}^{q} \beta_{ijk} x_i x_j x_k + \epsilon. \tag{3}$$

$$y = \sum_{i=1}^{q} \beta_i x_i + \sum_{i=1}^{q-1} \sum_{j=i+1}^{q} \beta_{ij} x_i x_j + \sum_{i=1}^{q-2} \sum_{j=i+1}^{q-1} \sum_{k=j+1}^{q} \beta_{ijk} x_i x_j x_k + \epsilon. \tag{4}$$

For experiments involving q mixture components and p process variables, so called mixture-process variable experiments, a model can be created by crossing a mixture component model, like Equation 2, with a standard linear or response surface model in the process variables. For example, crossing a Scheffé quadratic model in q mixture components (x_i, $i = 1, \dots q$) with a linear plus linear by linear interactions model in p process variables (z_j, $j = 1 \dots p$) results in Equation 5.

$$y = (\sum_{i=1}^{q} \beta_i x_i + \sum_{i=1}^{q-1} \sum_{j=i+1}^{q} \beta_{ij} x_i x_j)(\alpha_0 + \sum_{l=1}^{p} \alpha_l z_l + \sum_{l=1}^{p-1} \sum_{m=l+1}^{p} \alpha_{lm} z_l z_m) = \sum_{i=1}^{q} \beta_i^{(0)} x_i$$
$$+ \sum_{i=1}^{q-1} \sum_{j=i+1}^{q} \beta_{ij}^{(0)} x_i x_j + \sum_{i=1}^{q} \sum_{l=1}^{p} \beta_{il}^{(1)} x_i z_l + \sum_{i=1}^{q-1} \sum_{j=i+1}^{q} \sum_{l=1}^{p} \beta_{ijl}^{(1)} x_i x_j z_l + \sum_{i=1}^{q} \sum_{l=1}^{p-1} \sum_{m=l+1}^{p} \beta_{ilm}^{(2)} x_i z_l z_m$$
$$+ \sum_{i=1}^{q-1} \sum_{j=i+1}^{q} \sum_{l=1}^{p-1} \sum_{m=l+1}^{p} \beta_{ijlm}^{(2)} x_i x_j z_l z_m + \epsilon. \quad (5)$$

However, the number of coefficients in Equation 5 increases rapidly with the number of mixture components and process variables. Kowalski, Cornell, and Vining (2000) proposed a more parsimonious second order model given in Equation 6. This model assumes there is no linear effect of the process variables on the nonlinear mixture component blending. Equation 6 is often adequate for mixture-process experiments.

$$y = \sum_{i=1}^{q} \beta_i^{(0)} x_i + \sum_{i=1}^{q-1} \sum_{j=i+1}^{q} \beta_{ij}^{(0)} x_i x_j + \sum_{k=1}^{p} \left[\sum_{i=1}^{q} \beta_i^{(1)} x_i \right] z_k + \sum_{k=1}^{p-1} \sum_{l=k+1}^{p} \alpha_{kl} z_k z_l + \sum_{k=1}^{m} \alpha_{kk} z_k^2 + \epsilon. \quad (6)$$

3. Creating a mixture design with mixexp

3.1. Standard mixture designs in unconstrained regions

In this section we illustrate the use of **mixexp** functions for creating standard designs for mixture experiments. Scheffé (1958) proposed the simplex-lattice designs denoted SLD(q,k) based on the interpretation of the coefficients for the Scheffé linear, quadratic and cubic models for mixture experiments. The **mixexp** function `SLD` creates these designs. Below is an example of creating a SLD(4,2) design.

```
R> library("mixexp")
R> SLD(4, 2)

   x1  x2  x3  x4
1 1.0 0.0 0.0 0.0
2 0.5 0.5 0.0 0.0
3 0.0 1.0 0.0 0.0
4 0.5 0.0 0.5 0.0
5 0.0 0.5 0.5 0.0
6 0.0 0.0 1.0 0.0
```

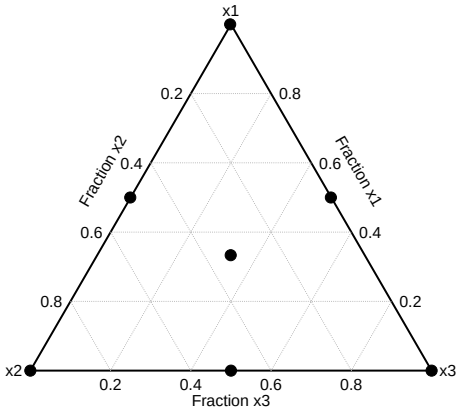

Figure 1: Simplex centroid design in three components.

```
7  0.5 0.0 0.0 0.5
8  0.0 0.5 0.0 0.5
9  0.0 0.0 0.5 0.5
10 0.0 0.0 0.0 1.0
```

The q in SLD(q,k) represents the number of mixture components and k represents the number of levels of each component. A k-level design supports fitting a Scheffé model of order k.

Cornell (2002) described alternate designs called a simplex-centroid designs that can be used to collect data appropriate for fitting the Scheffé special-cubic model. The **mixexp** function `SCD` creates these designs. Below is an example of creating a simplex-centroid design with three mixture components and storing it in the data frame `des`.

```
R> des <- SCD(3)
R> DesignPoints(des)
```

When there are only three mixture components, the **mixexp** function `DesignPoints` can be used to display the design graphically in the simplex experimental region. The result of the call to `DesignPoints` above is displayed in Figure 1.

3.2. Mixture designs in constrained regions

In many mixture experiments each component in the mixture can only be varied within specific lower and upper constraints. In this case, the experimental region is an irregular hyperpolyhedron rather than a simplex, and the Simplex Lattice and Simplex Centroid designs are not appropriate. In constrained experimental regions, McLean and Anderson (1966) recommend experimental designs that consist of all the extreme vertices of the constrained region, possibly augmented by edge and facet centroids.

Piepel (1988) published Fortran code for generating extreme vertices and various dimensional centroids of linearly constrained regions. The function `crvtave` in **mixexp** calls this Fortran code to create a design containing the extreme vertices and specified centroids. The function `Xvert` is a front end for `crvtave`. It takes as inputs vectors of upper and lower constraints for each mixture component, and bounds and coefficients for linear constraint equations. Next,

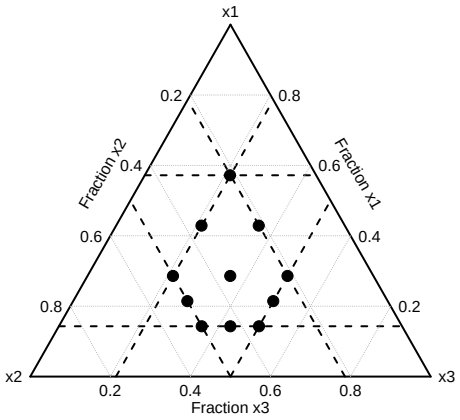

Figure 2: Simplex centroid design in three components.

it sets up the matrix defined by Piepel (1988), then calls the `crvtave` function to create the design. It returns a data frame with the vertices, requested edge and facet centroids, and the overall centroid. For example an experimental design consisting of the extreme vertices augmented by the edge and overall centroids defined by the constraint equations:

$$\begin{aligned} 0.143 \leq x_1 \leq 0.572 \\ 0.214 \leq x_2 \leq 0.50 \\ 0.214 \leq x_3 \leq 0.50, \end{aligned}$$

can be produced by the commands below.

```
R> Xvert(nfac = 3, lc = c(0.143, 0.214, 0.214), uc = c(0.572, 0.5, 0.5),
+    ndm = 1, pseudo = FALSE)

       x1     x2     x3 dimen
1  0.5720 0.2140 0.2140     0
2  0.1430 0.5000 0.3570     0
3  0.2860 0.5000 0.2140     0
4  0.2860 0.2140 0.5000     0
5  0.1430 0.3570 0.5000     0
6  0.1430 0.4285 0.4285     1
7  0.4290 0.2140 0.3570     1
8  0.2145 0.5000 0.2855     1
9  0.4290 0.3570 0.2140     1
10 0.2145 0.2855 0.5000     1
11 0.2860 0.3570 0.3570     2
```

When there are only three components in the design, the function `Xvert` calls `DesignPoints` to produce a visual representation of the design as shown in Figure 2.

Smith (2005) discussed a mixture experiment in formulating color photograhic dispersion that included linear constraints in addition to upper and lower bounds on the components. The

constraint equations on the design region are given below, where x_1 = Coupler, x_2 = Solvent A, x_3 = Solvent B, x_4 = Stabelizer A, x_5 = Stabelizer B.

$$
\begin{aligned}
0.3 \leq x_1 \leq 0.70 \\
0.0 \leq x_2 \leq 0.35 \\
0.0 \leq x_3 \leq 0.35 \\
0.0 \leq x_4 \leq 0.35 \\
0.0 \leq x_5 \leq 0.35 \\
0.15 \leq x_2 + x_3 \leq 0.35 \\
0.15 \leq x_4 + x_5 \leq 0.35
\end{aligned}
$$

Snee (1979) recommended that linear constraints be normalized by dividing all constants in the equation by the largest constant for numerical stability. This was not necessary in this example, since the largest constant in the linear constraint equations was 1. The code to create the design and the results are shown below.

```
R> coef <- matrix(c(0, 1, 1, 0, 0, 0, 0, 0, 1, 1), ncol = 5, byrow = TRUE)
R> Xvert(nfac = 5, lc = c(0.3, 0, 0, 0, 0), uc = c(0.7, 0.35, 0.35, 0.35,
+    0.35), nlc = 2, lb = c(0.15, 0.15), ub = c(0.35, 0.35), coef)
```

```
    x1    x2    x3    x4    x5 dimen
1  0.3 0.350 0.000 0.350 0.000     0
2  0.3 0.350 0.000 0.000 0.350     0
3  0.3 0.000 0.350 0.350 0.000     0
4  0.3 0.000 0.350 0.000 0.350     0
5  0.7 0.150 0.000 0.150 0.000     0
6  0.7 0.000 0.150 0.150 0.000     0
7  0.7 0.150 0.000 0.000 0.150     0
8  0.7 0.000 0.150 0.000 0.150     0
9  0.5 0.150 0.000 0.350 0.000     0
10 0.5 0.000 0.150 0.350 0.000     0
11 0.5 0.150 0.000 0.000 0.350     0
12 0.5 0.000 0.150 0.000 0.350     0
13 0.5 0.350 0.000 0.150 0.000     0
14 0.5 0.350 0.000 0.000 0.150     0
15 0.5 0.000 0.350 0.150 0.000     0
16 0.5 0.000 0.350 0.000 0.150     0
17 0.5 0.125 0.125 0.125 0.125     4
```

When there are constraints on several mixture components, the number of extreme vertices may become very large. Snee and Marquardt (1976) discussed an experiment in product development that involved eight mixture components with the following constraints.

$$
\begin{aligned}
0.10 \leq x_1 \leq 0.45 \quad & .10 \leq x_5 \leq 0.60 \\
0.05 \leq x_2 \leq 0.50 \quad & .05 \leq x_6 \leq 0.20 \\
0 \leq x_3 \leq 0.10 \quad & 0 \leq x_7 \leq 0.05 \\
0 \leq x_4 \leq 0.10 \quad & 0 \leq x_8 \leq 0.05
\end{aligned}
$$

The function `Xvert` finds 182 vertices for this constrained region. It was only feasible for the experimenters to complete a maximum of 20 experiments in order to detect which mixture components had large effects. Snee and Marquardt (1976) created a design by selecting a D-optimal subset of 16 vertices augmented by 4 overall centroids. A D-optimal set of sixteen vertices can be found using the `optFederov` function in the package **AlgDesign** in conjunction with the `Xvert` function. In the code below, a candidate data frame, `cand` (consisting of extreme vertices of the constrained region), is created using the `Xvert` function. Next, the run numbers are removed, and the reduced data frame is stored in `candm`. Finally the `optFederov` function is used to create the D-optimal subset that is stored in `ScreeMix`.

```
R> cand <- Xvert(nfac = 8, x1 = c(0.1, 0.45), x2 = c(0.05, 0.50),
+    x3 = c(0, 0.10), x4 = c(0, 0.1), x5 = c(0.1, 0.6), x6 = c(0.05, 0.2),
+    x7 = c(0, 0.05), x8 = c(0, 0.05))
R> candm <- cand[, 1:8]
R> library("AlgDesign")
R> ScreeMix <- optFederov(~ -1 + ., candm, nTrials = 16)
```

3.3. Augmenting designs with interior points

The designs produced by the `SLD`, `SCD` or `Xvert` functions include design points around the perimeter of the experimental region, possibly augmented by the centroid. Sometimes it is desirable to have additional interior points for checking the fit of a model. The **mixexp** function `Fillv` can add interior points to a design, created by `SLD`, `SCD` or `Xvert`, by averaging all possible pairs of design points. Lawson and Erjavec (2001) describe an experiment where an extreme vertices design was used to study the fixed carbon resulting from a coke briquette composed of a mixture of x_1 = calcinate, x_2 = tar free solids, and x_3 = tar solids, and baked at a constant temperature. Constraints on the mixture are shown below.

$$0.80 \leq x_1 \leq 0.90$$
$$0.08 \leq x_2 \leq 0.15$$
$$0.00 \leq x_3 \leq 0.05$$

The following code illustrates the use of the `Xvert` function to create the extreme vertices design called `coke`, and the use of the `Fillv` function to add additional interior points to the design `coke` to create the design `cokef`. `Xvert` automatically created the graph of the design points in `coke` on the left side of Figure 3. The `DesignPoints` function was used to display the design `cokef` graphically on the right side of Figure 3. The option `pseudo = TRUE` illustrates how the experimental region can be expanded by plotting within the pseudo component space. `Xvert` automatically plots within pseudo component space without the option `pseudo = FALSE`.

```
R> coke <- Xvert(nfac = 3, uc = c(0.9, 0.15, 0.05), lc = c(0.80, 0.08, 0),
+    pseudo = FALSE)
R> coke <- signif(coke, 3)
R> cokef <- Fillv(nfac = 3, coke)
R> DesignPoints(cokef, x1lower = 0.8, x1upper = 0.9, x2lower = 0.08,
+    x2upper = 0.15, x3lower = 0, x3upper = 0.05, pseudo = TRUE)
```

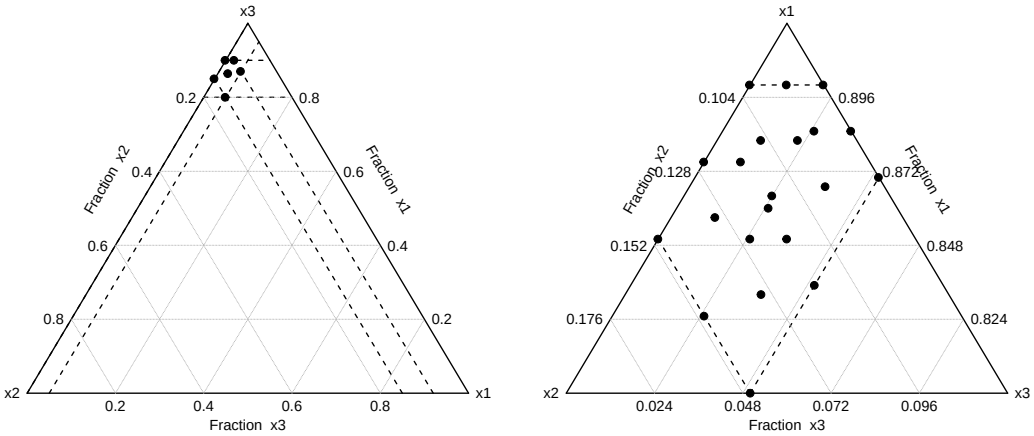

Figure 3: Comparison an extreme vertices design to an augmented extreme vertices design on unrestricted and psuedo-component space.

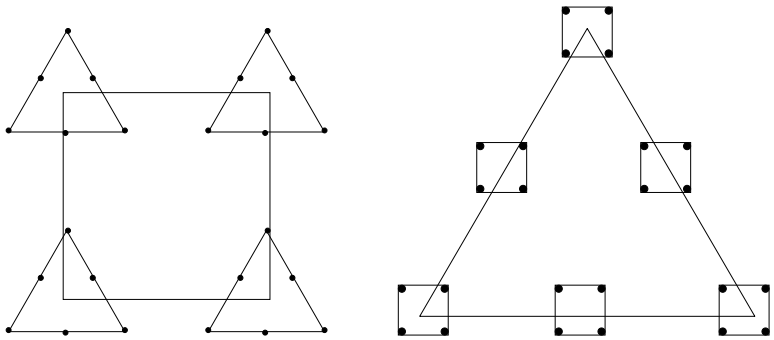

Figure 4: Design for fitting fully crossed mixture-process model.

3.4. Designs for mixture experiments with process variables

To create a design for fitting the fully crossed mixture-process model (Equation 5), a SLD(q,2) design can be crossed with a 2^p design in the process variables. With three mixture components and two process variables, this design can be visualized in two ways in Figure 4.

This design can be created by merging (with the R `merge` function) a mixture design created with the `SLD` function with a factorial design created with the the R `expand.grid` function as shown in the code below Figure 4.

```
R> sld <- SLD(3, 2)
R> id <- rep(c(1, 2, 3, 4), each = 6)
R> sldm <- rbind(sld, sld, sld, sld)
R> sldm <- cbind(sldm, id)
R> facdes <- expand.grid(z1 = c(-1, 1), z2 = c(-1, 1))
R> id <- 1:4
R> facdes <- cbind(facdes, id)
R> mixproc <- merge(sldm, facdes, by = "id", all = TRUE)
```

A subset of this design that would be D-optimal for fitting the reduced mixture-process vari-

able model, like Equation 6, can be created using the `optFederov` function in the **AlgDesign** package, as shown below.

```
R> library("AlgDesign")
R> mixprocR <- optFederov(~ x1 + x2 + x3 + x1:x2 + x1:x3 + x2:x3 + x1:z1 +
+    x1:z2 +  x2:z1 + x2:z2 + x3:z1 + x3:z2 + z1:z2 - 1, mixproc,
+    nTrials = 15, criterion = "D", maxIteration = 100, nRepeats = 10)
```

4. Fitting mixture experiment models with R

Scheffé mixture models like Equations 1–4 and more complicated models like Equation 5 can be fit to the data from mixture experiments using the `lm` function. For example the code below illustrates fitting the special-cubic model (Equation 4) to the data from silicon wafer etching experiment described by Myers and Montgomery (2002).

```
R> library("mixexp")
R> data("etch")
R> emod <- lm(erate ~ -1 + x1 + x2 + x3 + x1:x2 + x1:x3 + x2:x3 + x1:x2:x3,
+    data = etch)
R> summary(emod)
```

The design for this experiment was a partially replicated Simplex-Centroid Design augmented by interior axial points, the code above illustrates how the special-cubic model can be fit to this data using the R function `lm`.

When fitting models that do not include an intercept, Cornell (2002) points out that the R^2 statistic from a standard least-squares model fit will be inflated, due to the fact that the total sum of squares is not corrected for the mean. This gives the impression that the model fits better than it actually does. By including the intercept and leaving out one of the linear terms (i.e., `x1`), the `lm` function will produce the correct R^2, but the intercept reported in the model summary will actually be the coefficient for the term left out (i.e., $\hat{\beta}_0^I = \hat{\beta}_1^{NI}$) where I is the model including the intercept without `x1`, and NI is the no-intercept model that includes `x1`. The coefficients for the remaining linear terms in the model that includes an intercept (I), are actually the differences of the the coefficients for those terms minus the coefficient for the term left out (i.e., $\hat{\beta}_i^I = \hat{\beta}_i^{NI} - \hat{\beta}_1^{NI}$). The estimated variances of the coefficients of linear terms in the model including an intercept (I) will likewise be functions of the variances and covariances of the estimated coefficients in the no-intercept model (NI). As an alternative to using `lm` to fit involving mixture components, the **mixexp** function `MixModel` fits the models 1–6 (shown in Section 2) and prints the correct coefficients, standard errors, and R^2. An example is shown below with the data from Myers and Montgomery (2002)'s silicon wafer etching experiment.

```
R> MixModel(frame = etch, "erate", mixcomps = c("x1", "x2", "x3"), model = 4)
```

The output is shown below which matches the results in Myers and Montgomery (2002).

```
   coefficients   Std.err     t.value         Prob
x1    550.199515  23.22446 23.69051468 6.067419e-08
```

	Mixture components			Process variables factorial + center				
#	x_1	x_2	x_3	$(-1, 1)$	$(1, 1)$	$(-1, -1)$	$(1, -1)$	$(0, 0)$
1	0.0241	0.6018	0.3741	3010	3480	2780	4100	3840
2	0.0275	0.9725	0.0000	8510	5670	7060	5210	6320
3	0.0275	0.0000	0.9725	1600	2580	1660	2440	2210
4	0.0000	0.6667	0.3333	4560	4350	3990	4130	5210
5	0.0000	0.3333	0.6667	1930	3080	1810	3340	2600
6	0.0549	0.6300	0.3151	1900	4740	2160	4330	2780
7	0.0549	0.3151	0.6300	1780	3750	2000	3350	3140

Table 1: Viscosity response measurements (Pa·s) for mixture-process variable experiment with mayonnaise.

```
x2          344.723325  23.22446 14.84311192 1.509476e-06
x3          268.294753  23.22446 11.55224716 8.203511e-06
x2:x1       689.537037 146.51489  4.70625916 2.192427e-03
x3:x1        -9.034392 146.51489 -0.06166193 9.525557e-01
x2:x3        58.108466 146.51489  0.39660451 7.034720e-01
x2:x3:x1   9243.333333 940.85336  9.82441444 2.404146e-05

Residual standard error:  33.43177  on  7 degrees of freedom
Corrected Multiple R-squared:  0.9836603
```

To illustrate the analysis of a mixture experiment with process variables, consider the problem discussed by Sahni, Piepel, and Naes (2009). They studied a process to produce low-fat mayonnaise. The product was a mixture of three components x_1 = stabilizer, x_2 = starch 1, and x_3 = starch 2. The response they were interested in was the viscosity of the final product that was influenced not only by the ingredients, but also by two process variables: z_1 =heat exchanger temperature and z_2 = the flow rate through the system. The goal was to achieve a viscosity of 3657 at the lowest cost. The constraints on the mixture components are shown below:

$$0.0 \leq x_1 \leq 0.0549$$
$$0.0 \leq x_2 \leq 0.9725$$
$$0.0 \leq x_3 \leq 0.9725$$

The data from the crossed extreme-vertices design in the mixture components and a 2^2 factorial design (augmented by a center point) in the process variables is shown in Table 1. This data set is stored in the data frame `MPV` which is part of the **daewr** package (Lawson 2015b) that contains the data frames and functions from the book *Design and Analysis of Experiments with R* (Lawson 2015a). Equation 5 can be fit to this data using the commands below the table.

```
R> library("daewr")
R> data("MPV")
R> modmp <- lm(y ~ -1 + x1 + x2 + x3 + x1:x2 + x1:x3 + x2:x3 + x1:z1 +
+    x2:z1 + x3:z1 + x1:z2 + x2:z2 + x3:z2 + x1:x2:z1 + x1:x3:z1 +
```

```
+    x2:x3:z1 + x1:x2:z2 + x1:x3:z2 + x2:x3:z2 + x1:z1:z2 + x2:z1:z2 +
+    x3:z1:z2 + x1:x2:z1:z2 + x1:x3:z1:z2 + x2:x3:z1:z2, data = MPV)
> summary(modmp)
```

Alternatively this model can be fit with the `MixModel` function as shown below.

```
R> MixModel(MPV, "y", mixcomps = c("x1", "x2", "x3"), model = 5,
+    procvars = c("z1", "z2"))
```

5. Graphical display of fitted models for mixture experiments

5.1. Contour plots

The function `contourPlot3` in the R package **qualityTools** can be used to make ternary contour plots in the unconstrained mixture-simplex design space. However, if there are constraints on the mixture components, it might not show much detail in the constrained region. The function `ModelPlot` in the package **mixexp** can show the constraints in the simplex and optionally zero-in to the pseudo-component space bounded by the lower constraints on each component. This makes it easier to see details of the response surface in the restricted region. For example the R code below creates two contour plots of a quadratic model fit to the data of Cornell (2002)'s Table 4.1. In this code, the design is created by the `Xvert` function in **mixexp**, and the R function `lm` is used to fit the quadratic mixture model 2. The object `lm` object `quadm` is the first argument to the `ModelPlot` function.

```
R> orig <- Xvert(nfac = 3, lc = c(0.35, 0.2, 0.15), uc = c(1, 1, 1),
+    ndm = 1, plot = FALSE)
R> y <- c(15.3, 20.0, 28.6, 12.5, 32.7, 42.4)
R> orig <- cbind(orig[1:6, ], y)
R> quadm <- lm(y ~ -1 + x1 + x2 + x3 + x1:x2 + x1:x3 + x2:x3, data = orig)
R> title <- c("Actual Component Space", "Pseudo Component Space")
R> option <- c(FALSE, TRUE)
R> for (i in 1:2) {
+    ModelPlot(model = quadm,
+      dimensions = list(x1 = "x1", x2 = "x2", x3 = "x3"),
+      main = title[i], lims = c(0.35, 1, 0.20, 1, 0.15, 1),
+      constraints = TRUE, contour = TRUE, cuts = 6, fill = TRUE,
+      axislabs = c("x1", "x2", "x3"), cornerlabs = c("x1", "x2", "x3"),
+      pseudo = option[i])
+  }
```

The `dimensions` argument to `ModelPlot` gives the names (in quotes) of the mixture components in the `lm` object that will be plotted on the vertical, left and right axis of the simplex. The `lims` argument gives the lower and upper constraints on the vertical, left and right axis of the simplex, `constraints = TRUE` specifies the constraints will be plotted on the graph, `contours = TRUE` indicates contour lines are to be plotted, `cuts = 6` indicates 6 contour lines, and `fill = TRUE` indicates regions between the contours should be colored.

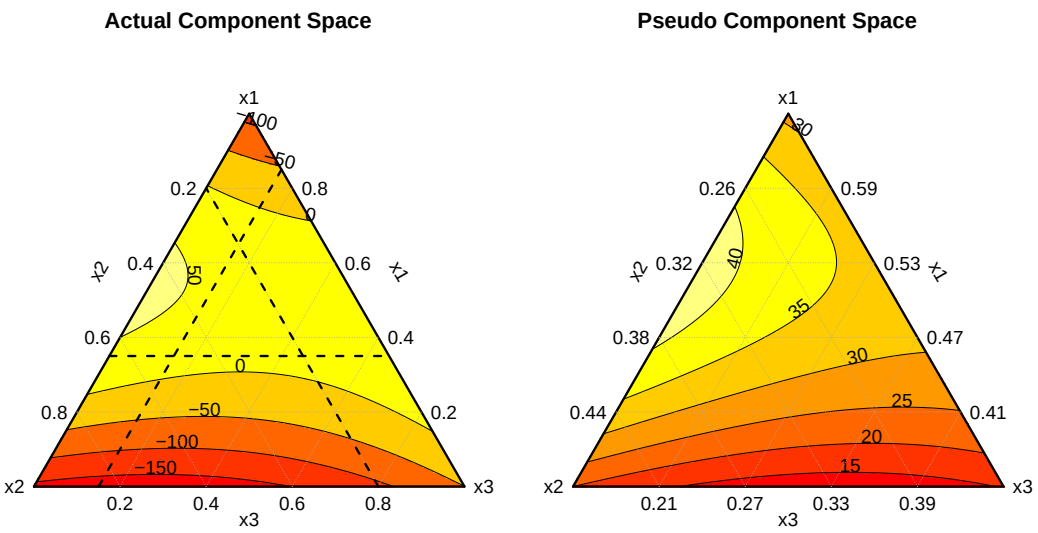

Figure 5: Comparison of contour plots on unrestricted and pseudo-component space.

When `pseudo = FALSE`, as in the first time through the loop, the contour plot is made over the entire simplex as shown in left of Figure 5. When `pseudo = TRUE`, as in the second time through the loop, the contour plot is made over the pseudo-component space as shown in right of Figure 5. As can be seen, there is much more detail in the restricted space on the right.

The `ModelPlot` function in **mixexp** can also make mixture-contour plots of equations involving more than three components by holding some components fixed at values chosen by the user. For example, consider Cornell (2002)'s application of blending four chemical pesticides for control of mites. The design was a simplex-centroid design that is created by the `SCD` function in the code below. The special-cubic model (Equation 4) was fit using the `MixModel` function. In the first time through the loop, the `ModelPlot` function is called with the argument `slice = list(x4 = 0.0)` that holds x_4 =Dibrom constant at 0.0. This produced the contour plot on the left side of Figure 6. In the second time through the loop, `ModelPlot` function is called with the argument `slice = list(x4 = 0.4)` that fixes x_4 =Dibrom at 0.4 and produces the plot in the right side of Figure 6.

```
R> mite <- SCD(4)
R> yavg <- c(1.8, 25.4, 28.6, 38.5, 4.9, 3.1, 28.7, 3.4, 37.4, 10.7, 22.0,
+    2.6, 2.4, 11.1,0.8)
R> mitemdM <- MixModel(frame = cbind(mite, yavg), response = "yavg",
+    mixcomps = c("x1", "x2", "x3", "x4"), model = 4)
R> for (x4 in c(0.0,0.4)) {
+    ModelPlot(model = mitemdM,
+      dimensions = list(x1 = "x1", x2 = "x2", x3 = "x3"),
+      slice = list(mix.vars = c(x4 = x4)),
+      main = paste("Dibrom = ", toString(x4, width = 4)),
+      constraints = FALSE, contour = TRUE, at = c(5, 10, 15, 20),
+      fill = FALSE, axislabs = c("Vendex", "Omite", "Kelthane"),
+      cornerlabs = c("X1", "X2", "X3"), pseudo = FALSE)
+  }
```

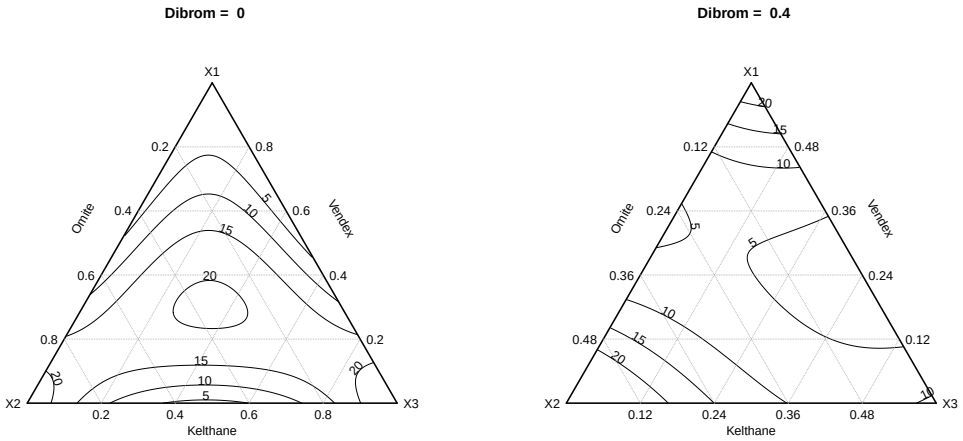

Figure 6: Slice at $x_4 = 0.0$ and $x_4 = 0.4$.

The `ModelPlot` function can also be used to make contour plots of models fit to mixture-process variable experiments by holding the process variables constant. For example, the code below illustrates fitting a mixture-process variable model resulting from crossing a Scheffé special cubic model with a full factorial model in three two-level process variables. The data comes from Cornell (2002)'s famous fish patty experiment, where x_1 is the fraction of Mullet, x_2 is the fraction of Sheepshead, and x_3 is the fraction of Croker in the fish patty mixture. The process variables z_1, z_2, and z_3 are the coded values of the process variables: cooking temperature, cooking time, and frying time, respectively.

```
R> library("mixexp")
R> data("fishp")
R> fish.mdM <- MixModel(frame = fishp, response = "y",
+    mixcomps = c("x1", "x2", "x3"), model = 5,
+    procvars = c("z1", "z2", "z3"))
R> Title <- c("Cook Temp=High, Cook Time = High, Fry Time=Low",
+    "Cook Temp=Low, Cook Time=Low, Fry Time=High")
R> pv <- matrix(c(1, -1, 1, -1, -1, 1), ncol = 3)
R> for (i in 1:2) {
+    ModelPlot(fish.mdM,
+      dimensions = list(x1 = "x1", x2 = "x2", x3 = "x3"),
+      slice = list(process.vars = c(z1 = pv[i, 1],
+        z2 = pv[i, 2], z3 = pv[i, 3])),
+      main = Title[i], constraints = FALSE, contour = TRUE,
+      cuts = 10, fill = FALSE, axislabs = c("Fraction mullet",
+        "Fraction sheepshead", "Fraction croaker"),
+      cornerlabs = c("mullet", "sheepH", "croaker"), pseudo = FALSE)
+  }
```

The resulting contour plots at two combinations of cooking temperature, cooking time, and frying time are shown in Figure 7.

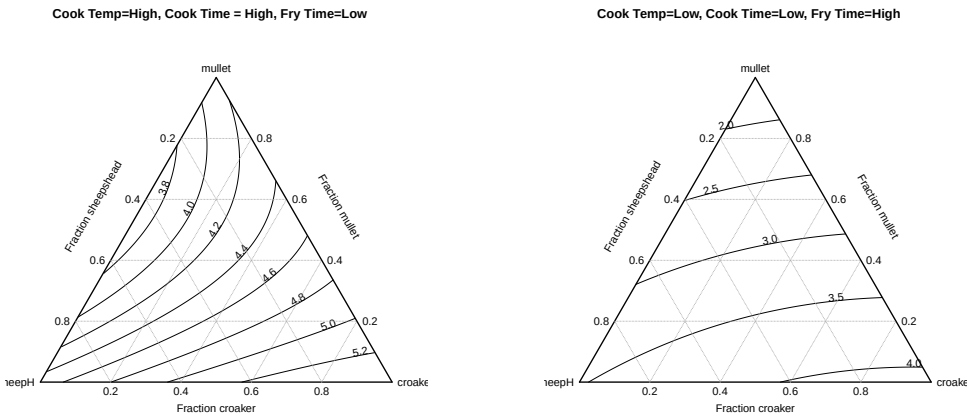

Figure 7: Contour plots of the texture of Cornell's fish patties.

5.2. Effect plots

When there are more than three components in a mixture, another way of representing the fitted surface is by plotting the predicted response $\hat{y}(x)$ along each component axis on the same graph (Cornell 2002). Cox (1971) defined the component axis for component x_i as a line that passes through the design centroid to the vertex where $x_i = 1.0$. Piepel (1982) defined the component axes as lines passing through the design centroid to the pseudo-component vertices. The `ModelEff` function in **mixexp** plots the predicted response traces along either the Cox or Piepel directions. After the model `mitemdM` is fit with the `MixModel` function, as shown in the code example below Figure 5, the plot can be produced with the following call of `ModelEff`.

```
R> ModelEff(nfac = 4, mod = 4, dir = 2, ufunc = mitemdM,
+    dimensions = list("x1", "x2", "x3", "x4"))
```

The argument `nfac` specifies the number of components in the model and the argument `mod = 4` specifies that a special cubic model will be supplied, which is given in the `lm` object specified by `ufunc=mitemdM`. The argument `dimensions` gives the names (in quotes) of the mixture components in the `lm` object `mitemdM`. Finally, the `dir` argument specifies whether the plot should be made on the Piepel = 1 direction or Cox = 2 direction. The `lm` object can be created by the `MixModel` function, or by the R function `lm`, as long as the terms in the model are specified in the same order they would be specified by `MixModel`. The resulting plot is shown in Figure 8.

In this plot it can be seen that decreasing the proportions of either x_2 = Omite or x_3 = Kelthane in the mixture causes the average percentage of mites remaining (relative to the number before spraying) to decrease from more than 20 percent to near 0 percent. This plot also shows that the optimal proportion of x_4 = Dibrom for reducing the average percentage of mites is near its value at the centroid of the design, and that changing the proportion of x_1 = Vendex in the mixture has the smallest effect on mite control.

Snee and Marquardt (1976) showed that response trace plots for the linear model are useful in screening experiments with constrained regions and many mixture components. It is inappropriate to judge the importance of the terms in the Scheffé linear model (Equation 1) by

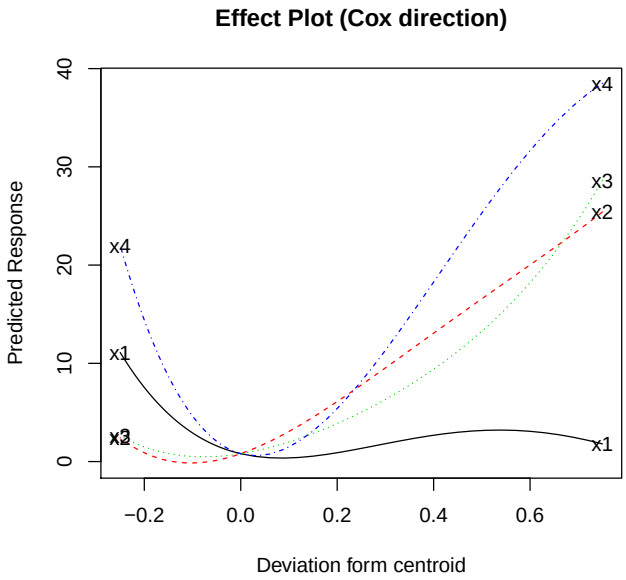

Figure 8: Predicted response trace plot.

the statistical significance of their coefficients since the physical interpretation of these coefficients is the predicted responses at the pure component mixtures (which may be out of the constrained experimental region). However, when the response trace along a component axis is a flat horizontal line, it indicates that changing the proportion of that component has little effect on the response. The data for the 16-run screening experiment in eight components discussed in Section 3.2, and described by Snee and Marquardt (1976), is included in the data frame `SneeMq` in the **mixexp** package. The code below produces the plot shown in Figure 9.

```
R> library("mixexp")
R> data("SneeMq")
R> SneeM <- lm(y ~ -1 + x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8, data = SneeMq)
R> ModelEff(nfac = 8, mod = 1, dir = 1, ufunc = SneeM,
+    dimensions = list("x1", "x2", "x3", "x4", "x5", "x6", "x7", "x8"),
+    lc = c(0.1, 0.05, 0, 0, 0.1, 0.05, 0, 0),
+    uc = c(0.45, 0.5, 0.1, 0.1, 0.6, 0.2, 0.05, 0.05))
```

The arguments `lc` and `uc` are included in the function call because the design region is constrained and `dir = 1` calls for a plot along the Piepel component axes.

The plot shows the response trace for x_6 to be essentially flat, and Snee and Marquardt (1976) concluded the amount of this component in the mixture had little effect on the response. The plot also shows that the response traces for x_2 and x_3 nearly overlap. In addition, the traces for x_1 and x_4 and the traces for x_5, x_7 and x_8 nearly overlap. With these graphical clues, Snee and Marquardt (1976) found that within that the standard errors of the fitted coefficients $\beta_2 \approx \beta_3$, $\beta_1 \approx \beta_4$, and $\beta_5 \approx \beta_7 \approx \beta_8$. This means that the effect of changing the proportions of x_2 or x_3; or x_1 or x_4; or x_5, x_7 or x_8 is essentially the same. Based on this observation, Snee and Marquardt (1976) concluded the model could be reduced to three dimensions by defining $x_1' = (x_2 + x_3)/(1 - x_6)$, $x_2' = (x_1 + x_4)/(1 - x_6)$, and $x_3' = (x_5 + x_7 + x_8)/(1 - x_6)$.

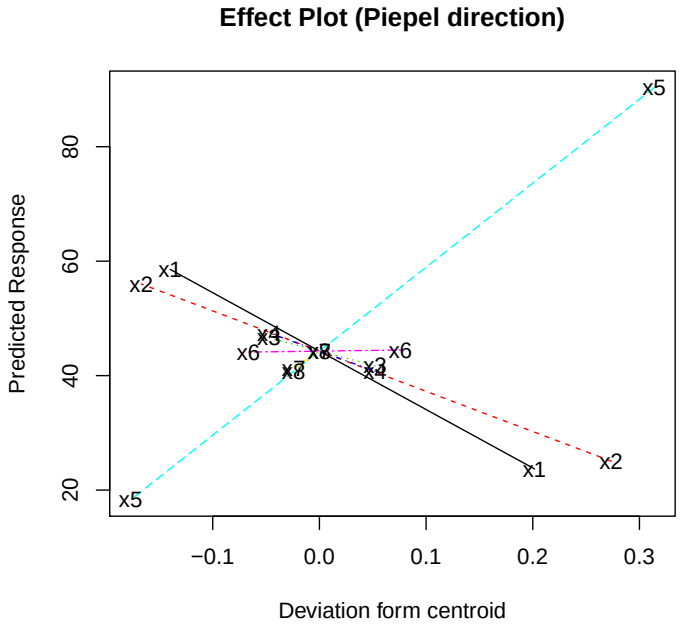

Figure 9: Predicted response trace plot for Snee and Marquardt's screening experiment with 8 components.

For models 1, 2, and 4 in Section 2, there is a simpler function `EffPlot` that creates response traces along the Cox or Piepel component axes when there is a data frame containing the design with the mixture components `x1` - `xnfac` as the first `nfac` columns and a response `y` as the last column. This is the format of `SneeMq`. Therefore, the plot in Figure 9 can be produced with the following code.

```
R> data("SneeMq")
R> Effplot(des = SneeMq, mod = 1, dir = 1)
```

There is no need to include the lower and upper constraints or a user function as arguments to `EffPlot` since this function finds the constraints in the data frame and fits the model internally.

When there are process variables in a mixture experiment, `ModelEff` can be used to make response traces along the Cox or Piepel component axes, with the process variables held constant. For example, the code on the next page illustrates making plots at the low and high values of the coded process variable $z = (RPM - 65)/20$ in the experiment (reported by Gallant, Prickett, Cesarec, and Bruck (2008)) to determine the effects of mixture components and a process variable on the burning rate of composite rocket propellants.

```
R> library("mixexp")
R> data("Burn")
R> testBNM <- MixModel(Burn, "y", mixcomps = c("Course", "Fine", "Binder"),
+    model = 6, procvars = "z")
R> z <- c(1,-1)
R> for (i in 1:2) {
```

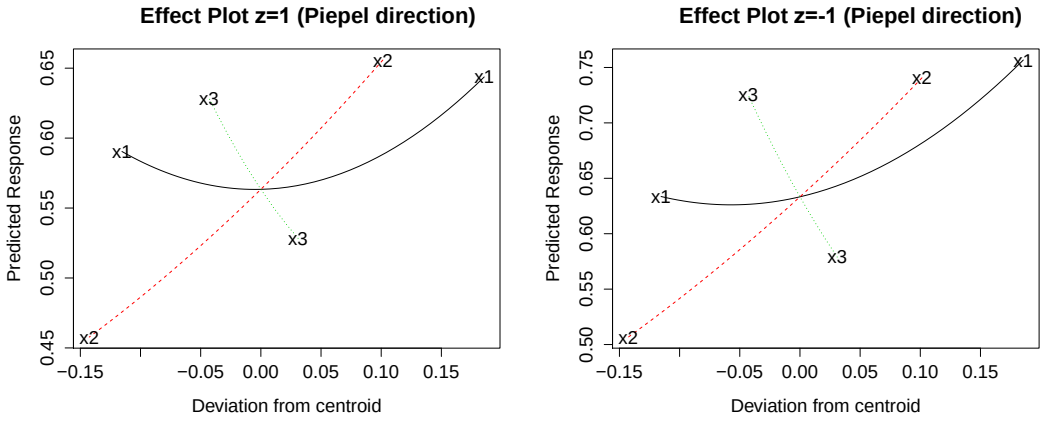

Figure 10: Predicted response traces for composite rocket propellant burning experiment.

```
+    ModelEff(nfac = 3, mod = 6, nproc = 1, dir = 1, ufunc = testBNM,
+      dimensions = list(NULL), pvslice = z[i], lc = c(0.403, 0.166, 0.130),
+      uc = c(0.704, 0.412, 0.210))
+  }
```

6. Discussion and future directions

Design and analysis of mixture experiments in R is enhanced by functions in the **mixexp** package. This package provides functions for creating mixture designs in unconstrained regions. It also uses Piepel (1988)'s Fortran code to create designs in linearly constrained regions. In conjunction with the `Fillv` function, which creates interior points, and the **AlgDesign** package an even wider variety of mixture designs such as screening designs, mixture-process variables designs, and blocked mixture experiments can be created. Model fitting for mixture experiments can be accomplished with the R `lm` function. **mixexp** also provides functions for ternary contour plots and response trace plots of models fit to mixture experiments. This enhances the graphical capabilities for mixture experiments available in R.

When process variables are included in mixture experiments, it is often convenient to run the experiments in a split-plot arrangement. For example, if it is time consuming to make the mixtures, it might be convenient to make a large batch of the first mixture of components in the design, and then subject parts of this large batch to different combination of levels of the process variables while the next mixture batch is being made.

When a mixture-process experiment is performed in this way, a split-plot arrangement results and a mixed model should be fit to the data. This can be done using the `lmer` function in the **lme4** package (Bates, Mächler, Bolker, and Walker 2015). Goos and Donev (2007) proposed efficient designs for mixture-process experiments in split-plot arrangements. While there is no ability to create their designs in R, Goos and Donev (2007) have provided Fortran code to create these designs. Future plans are to use this Fortran code by calling it with a function in **mixexp** similar to the way that Piepel (1988)'s code is called for creating designs in constrained regions.

References

Bates D, Mächler M, Bolker B, Walker S (2015). "Fitting Linear Mixed-Effects Models Using **lme4**." *Journal of Statistical Software*, **67**(1), 1–48. doi:10.18637/jss.v067.i01.

Cornell JA (2002). *Experiments with Mixtures-Designs, Models, and the Analysis of Mixture Data*. 3rd edition. John Wiley & Sons, New York. doi:10.1002/9781118204221.

Cox DR (1971). "A Note on Polynomial Response Functions for Mixtures." *Biometrika*, **58**, 155–159. doi:10.1093/biomet/58.1.155.

Gallant FM, Prickett SE, Cesarec M, Bruck HA (2008). "Ingredient and Processing Effects on the Burning Rates of Composite Rocket Propellants Utilizing a Reduced-Run Mixture-Process Experiment Design." *Chemometrics and intelligent laboratory systems*, **90**, 49–63. doi:10.1016/j.chemolab.2007.08.007.

Goos P, Donev AN (2007). "Tailor-Made Split-Plot Designs for Mixture and Process Variables." *Journal of Quality Technology*, **39**, 326–339.

Kowalski SM, Cornell JA, Vining GG (2000). "A New Model and Class of Designs for Mixture Experiments with Process Variables." *Communications in Statistics – Theory and Methods*, **29**, 2255–2280. doi:10.1080/03610920008832606.

Lawson J (2015a). *Design and Analysis of Experiments with R*. CRC Press, Boca Raton.

Lawson J (2015b). ***daewr**: Design and Analysis of Experiments with R*. R package version 1.1-6, URL https://CRAN.R-project.org/package=daewr.

Lawson J (2016). ***mixexp**: Design and Analysis of Mixture Experiments*. R package version 1.2.5, URL https://CRAN.R-project.org/package=mixexp.

Lawson J, Erjavec J (2001). *Modern Statistics for Engineering and Quality Improvement*. Duxbury, Pacific Grove.

McLean RA, Anderson VL (1966). "Extreme Vertices Designs of Mixture Experiments." *Technometrics*, **8**, 447–454. doi:10.1080/00401706.1966.10490377.

Minitab Inc (2010). "Minitab Software for Quality Improvement." URL http://www.minitab.com/.

Myers RH, Montgomery DC (2002). *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. 2nd edition. John Wiley & Sons, New York.

Piepel GF (1982). "Measuring Component Effects in Constrained Mixture Experiments." *Technometrics*, **24**, 29–39. doi:10.1080/00401706.1982.10487706.

Piepel GF (1988). "Programs for Generating Extreme Vertices and Centroids of Linearly Constrained Experimental Regions." *Journal of Quality Technology*, **20**, 125–139.

Piepel GF (1997). "Survey of Software with Mixture Experiment Capability." *Journal of Quality Technology*, **29**, 76–85.

Ragavarao D, Wiley JB, Chitturi P (2011). *Choice-Based Conjoint Analysis Models and Designs*. CRC Press, Boca Raton. doi:10.1201/9781420099973.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Roth T (2016). ***qualityTools**: Statistical Methods for Quality Science*. R package version 1.55, URL https://CRAN.R-project.org/package=qualityTools.

Sahni NS, Piepel GF, Naes T (2009). "Product and Process Iimprovement Using Mixture-Process Variable Methods and Robust Optimization Techniques." *Journal of Quality Technology*, **41**, 181–197.

SAS Institute Inc (2010a). "Getting Started with the SAS 9.2 ADX Interface for Design of Experiments." URL http://support.sas.com/documentation/cdl/en/adxgs/60376/PDF/default/adxgs.pdf.

SAS Institute Inc (2010b). "**JMP** Statistical Discovery Software." URL http://www.jmp.com/.

Scheffé H (1958). "Experiments with Mixtures." *Journal of the Royal Statistical Society B*, **20**, 344–360.

Smith WF (2005). *Experimental Design for Formulation*. ASA-SIAM, Alexandria.

Snee RD (1979). "Experimental Designs for Mixture Systems with Multicomponent Constraints." *Communication in Statistics – Theory and Methods*, **A8**, 303–326. doi:10.1080/03610927908827762.

Snee RD, Marquardt DW (1976). "Screening Concepts and Designs for Experimets with Mixtures." *Technometrics*, **18**, 19–29. doi:10.2307/1267912.

Stat-Ease, Inc (2010). "**Design-Expert** V8 Software for Design of Experiments (DOE)." URL http://www.statease.com/.

StatPoint Technologies (2010). "**Statgraphics** Centurion: Data Analysis and Statistical Software." URL http://www.statgraphics.com/.

Umetrics an MKS Company (2014). "**MODDE** Design of Experiments." URL http://www.umetrics.com/products/modde.

Wheeler RE (2014). ***AlgDesign**: Algorithmic Experimental Design*. R package version 1.1-7.3, URL https://CRAN.R-project.org/package=AlgDesign.

Affiliation:

John Lawson
Department of Statistics
Brigham Young University
Provo, UT, United States of America
E-mail: lawson@byu.edu

Cameron Willden
W. L. Gore and Associates
Newark, DE, United States of America
E-mail: ccwillden@gmail.com
URL: http://www.ci.tuwien.ac.at/~zeileis/

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
August 2016, Volume 72, Code Snippet 2 *Submitted:* 2012-09-24
doi:10.18637/jss.v072.c02 *Accepted:* 2015-08-14