SAMON - sensitivity analysis for missing data

Aidan McDermott, Daniel O. Scharfstein
Johns Hopkins University

Introduction

In a repeated measures clinical trial groups of individuals are followed over time with outcome measures taken from each individual at fixed time-points. Often the focus of the trial is to compare the outcome means in two or more groups at some pre-specified time after enrollment. Unfortunately, data are not always available for each individual at each time-point. This can occur for a variety of reasons - individuals may discontinue participating in the study, or they may become non-compliant with the study protocol, or the outcome measure is not taken or it falls outside its detectable range, etc. If such missingness in the data is ignored and an analysis is performed using only those data available at the specified time-points then bias in the estimates and serious errors in the conclusions drawn from the study can result.

The SAMON package, [1], allows the user to perform a sensitivity analysis in the mean value of an outcome measure and the difference in the outcome measure between two treatment groups. The SAMON package handles discrete outcome measures and requires that the outcome measure should be coded as a positive integer (1,2,3, ...). Data not in this format need to be transformed to make this work.

SAMON can deal with missing data in the following settings:

- Monotonicity - if the data for an individual is missing at a time-point, t, then data for that individual is missing at all subsequent time-points. That is, there should be no intermittent missing data. Monotonic missing data is handled by the samon function.
- In cases where in addition to monotonic missing data, some individuals exhibit intermittent missingness then the function samonIM can be used.

In examples 1 through 4 we shall deal with monotonically missing data and in examples 5 and 6 we shall explore monotonically missing data with intermittent missing.

Data Structure

In general, the functions in SAMON deal with one treatment arm at a time. Results from the samon or samonIM functions are then passed to summary functions to compute confidence intervals and treatment effects. The input to data should have the following structure.

- The data from an individual should constitute one observation in the data.
- A separate numeric variable or column should store the outcome measure at each time-point.
- The variable corresponding to the first time-point (the baseline value) should not contain any missing values.

Such a dataset is often referred to as being in the "wide" format. Throughout examples 1 through 4 we use the PANSS data. Although simulated, these data are based on an actual placebo controlled randomized trial and maintain many of the characteristics of the original data. Our data has two arms, placebo (treatment = 1) and an active arm, those receiving a 6mg dose of the drug risperidone (treatment = 2). The outcome of interest was the total Positive and Negative Syndrome Scale score (PANSS score). The first few observations from treatment 1 of the PANSS data looks like this:

```
       V1  V2  V3  V4  V5  V6
 [1,]  90  87  86  93  72  87
 [2,] 112  NA  NA  NA  NA  NA
 [3,]  99  76  62  52  57  49
 [4,]  86  78  91 113  89  68
 [5,]  80  85  NA  NA  NA  NA
 [6,]  72  64  78 113  NA  NA
 [7,]  67  NA  NA  NA  NA  NA
 [8,]  96  NA  NA  NA  NA  NA
 [9,]  93  90  NA  NA  NA  NA
[10,]  78  70  53  85  NA  NA
[11,]  93  86  92  94  NA  NA
[12,] 111 112  95  NA  NA  NA
```

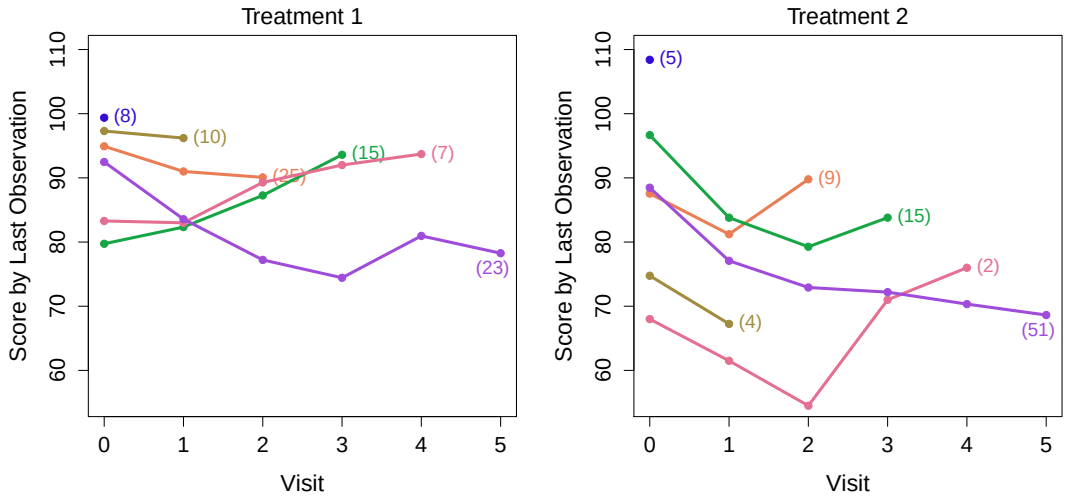
The V1 column contains the PANSS score at visit 0 (the baseline visit), V2, the PANSS score at visit 1 and so on. The last visit here is visit 5 and it is for this visit that we wish to compute the mean PANSS score. As can be seen, the twelfth individual shown has PANSS scores at visits 0 to 2, but not thereafter, while, the third and fourth individuals have PANSS scores at all time-points. There are 88 individuals in treatment group 1 in the PANSS dataset. We will compare this arm with treatment group 2 which has 86 individuals. We begin with a simple summary of the PANSS data (see [2] for further details).

Table 0.1: Summary statistics for two arms of the PANSS data.

	Placebo (treatment 1)						**Risperidone 6mg (treatment 2)**					
Visit	**N**	**NMiss**	**Mean**	**Std**	**Min**	**Max**	**N**	**NMiss**	**Mean**	**Std**	**Min**	**Max**
0	88	0	91.4	18.0	56	132	86	0	89.8	18.9	54	135
1	80	8	87.2	19.5	44	153	81	5	77.9	17.2	47	120
2	70	18	85.2	17.8	53	125	77	9	75.6	18.5	42	119
3	45	43	83.6	19.5	52	120	68	18	74.7	18.3	38	118
4	30	58	83.9	21.6	52	144	53	33	70.5	21.2	38	107
5	23	65	78.3	19.5	47	111	51	35	68.6	20.4	37	114

Of the 88 individuals at baseline in the placebo group only 23 have a PANSS score at visit 5, while, of the 86 individuals at baseline in the Risperidone arm 51 have values at visit 5. Figure 0.1 below shows the mean PANSS score at each visit stratified by when individuals were last seen.

Figure 0.1: PANSS score by visit for placebo arm (left panel labeled treatment 1) and risperidone arm (right panel labeled treatment 2) and by the time-point at which individuals were last observed.



Notation

We briefly introduce some notation. We use $t = 0, 1, 2, \ldots N_t$ to represent time-points and Y_t to represent the outcome measure at time t. The indicator variable R_t takes the value 1 if an individual is on-study at time t and 0 if not. One way of expressing the fact that data has a monotonic missing pattern is to say for any individual i, if $R_{i,t} = 0$ for some t then $R_{i,k}$ for all $k > t$.

For each time-point, $t > 0$, samon determines two distributions. The first is the probability that an individual will dropout at time t given that they provided an outcome at time $t-1$ and their outcome value at that time. We denote this distribution by H so that $H_{t-1}(Y_{t-1}) = Prob\,(R_t = 0 \mid Y_{t-1}$ and $R_{t-1} = 1)$. The second distribution is denoted by $F_{t-1}(Y_{t-1})$ and is the distribution of Y_t given that an individual is on study at time t and their outcome value Y_{t-1} at time $t-1$. In samon the distributions H_{t-1} and F_{t-1} are estimated using Gaussian smoothing kernels where each kernel is paramaterized by its standard deviation, denoted by σ_H and σ_F respectively. We refer to σ_H as the smoothing parameter for dropout and σ_F as the smoothing parameter for outcome.

For a given set of data optimal smoothing values for σ_H and σ_F are chosen by cross validation. The data is partitioned into a small number of partitions of approximately equal size. In turn each partition is withheld and the remaining data is used to evaluate working distributions of H and F. A loss function is then evaluated on the withheld partition using the working distributions. Summing the loss across all partitions gives the loss associated with a fixed value of a smoothing parameter. We choose optimal smoothing parameters by minimizing this loss function.

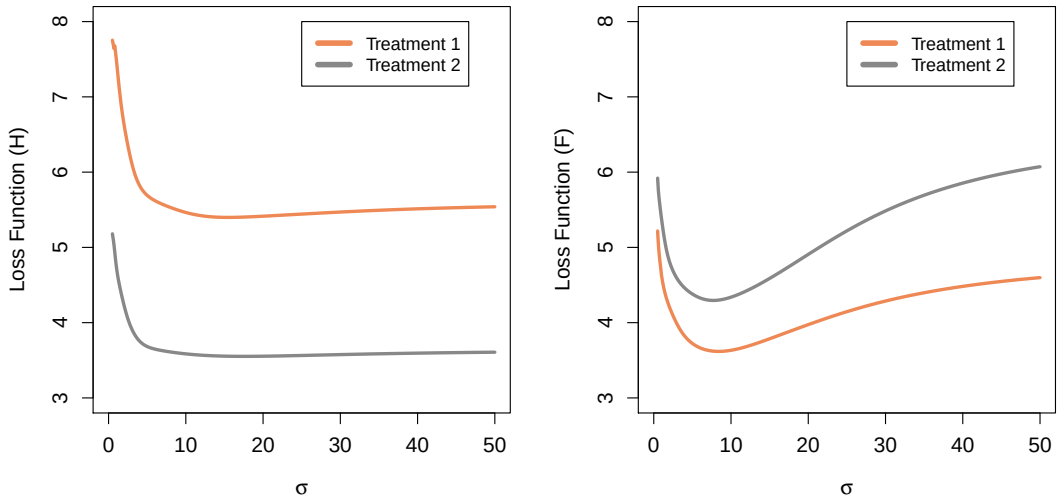
Example 1

We begin by examining the relationship between a loss function and its associated smoothing parameter. Later we will compute their optimal value of the smoothing parameter. Here is a partial printout of the R program "Example1a.R". We use the SAMON function samoneval to evaluate the loss functions in the PANSS data using 10 partitions of the data. We pass the vector "sigmas" to samoneval for which we want the loss function. Plots are then made of the relationship.

In this example we compute the smoothing parameter for treatment 1 (the placebo arm) of the PANSS data. The data are stored in "samonPANSS1" and can be accessed using the data function. We request that the data be divided into 10 partitions.

```
# Example 1a

library(samon, lib.loc="../../samlib")
data("samonPANSS1")

sigmas <- seq(0.1,50.0,by=0.1)
HF1 <- samoneval(
       mat         = samonPANSS1,     # the data
       Npart       = 10,              # number of partitions
       sigmaList   = sigmas,          # vector of sigmas
       type        = "both" )         # both sigmaP and sigmaQ
```

Figure 1.1: Loss function by smoothing parameter. Smoothing associated with dropout (H) is on the left and smoothing associated with outcome (F) is on the right.



As the smoothing parameter for dropout (σ_H) moves away from 0, the loss function flattens while the loss function for outcome (σ_F) is more obviously quadratic. Indeed, it may be the case that a loss function attains its minimum when σ_H or σ_F is large. In such cases it may be desirable to limit the value of σ since increasing the value of σ further does not substantially affect the smoothing or any other results we may obtain.

To estimate optimal smoothing parameters, σ_H and σ_F in the PANSS data we reproduce part of the samon help page:

```
samon                    package:samon                    R Documentation

Description:

  Given data from one arm of a repeated measures clinical trial, produces
  estimates of the expected value of the outcome at the final time-point
  for a range of sensitivity parameters.

Usage:

  samon(mat, Npart = 10, InitialSigmaH = 1.0, HighSigmaH = 2.0,
  InitialSigmaF = 1.0, HighSigmaF = 2.0, lb = 0, ub = 101, zeta1 = 1,
  zeta2 = 1, NSamples  = 0, seed0 = 1, MaxIter = 25, FAconvg = 1E-7,
  FRconvg = 1E-7, SAconvg = 1E-7, alphaList = c(0), MJackknife = FALSE,
  SJackknife = FALSE, retIFiM = FALSE, retIFiS = FALSE, Tfun= NULL)

Arguments:

    mat: matrix with (i,j) entry representing value for subject i at
         time-point j.

  Npart: Number of partitions to use when estimating optimal smoothing
         parameters, sigma H and sigma F.

InitialSigmaH: Initial value when calculating optimal sigma H.

HighSigmaH: Upper bound of search region when calculating optimal sigma H.

InitialSigmaF: Initial value when calculation optimal sigma F.

HighSigmaF: Upper bound of search region when calculating optimal sigma F.

MaxIter: Maximum iterations to use in optimizer.

FAconvg: Absolute change in function convergence criterion.

FRconvg: Relative change in function convergence criterion.

SAconvg: Step size convergence criterion.
```

There are a large number of arguments that can be given to the samon function. Here we will look at only those that immediately concern us.

- **Mat** is the name of the matrix holding the data. This needs to be in just the format described earlier with each individual's data stored in one row.
- Smoothing parameters are found by partitioning the data and then finding the smoothing parameter that minimizes a loss function. The number of partitions here is given by **Npart**. The data are broken into Npart parts based on row position. The data have been randomly sorted to avoid bunching of missing data into a small number of partitions.
- Minimization of the loss functions is by Newton's method. The parameter **InitialSigmaH** to the samon function gives an initial value for σ_H. If σ_H should go above **HighSigmaH**, the n the value **HighSigmaH** is returned. This allows the user to limit the search region. In a similar manner, **HighSigmaF** is an upper bound in the search for σ_F and **InitialSigmaF** holds the starting point for the search.
- **MaxIter** determines the maximum number of iterations to be performed in the optimizer. **SAconvg** controls the absolute step size to use. That is, if x_k is the smoothing parameter estimate at iteration i and $f_i = f(x_i)$ is the loss function evaluated at x_i, then, step size convergence is achieved when $\mid x_{i+1} - x_i \mid <$ SAconvg. **FAconvg** is the absolute function convergence criterion. Convergence is met by this criterion when $\mid f_{i+1} - f_i \mid <$ FAconvg. **FRconvg** is the relative function convergence criterion. Convergence occurs when $\mid \frac{f_{i+1}-f_i}{f_{i+1}+f_i} \mid <$ FRconvg.

Again we use the placebo arm of the PANSS data which is stored in "samonPANSS1" and accessed using the data function.

```
# Example1b.R
# Finding optimal Sigma_H and Sigma_F.
# ---------------------------------------------------
library(samon, lib.loc="../../samlib")

# get the treatment 1 data.
data("samonPANSS1")

samonResults <- samon(
   mat             = samonPANSS1,    # input matrix
   Npart           = 10,             # number of partitions

   InitialSigmaH   = 10.0,           # initial value and upper bound
   HighSigmaH      = 50.0,           # for sigma H

   InitialSigmaF   = 8.0,            # initial value and upper bound
   HighSigmaF      = 50.0,           # for sigma F

   SAconvg         = 1E-6,           # stopping criteria
   FAconvg         = 1E-6,
   FRconvg         = 1E-6 )

print(samonResults$HM)
print(samonResults$FM)
```

To run this code from the Examples/Example1 subdirectory issue the command:

R CMD BATCH –no-save –no-restore Example1b.R

We request that the search for the optimal smoothing parameter, σ_H, go no larger than 50 and give as initial estimate of 10. For σ_F the search should go no larger than 50 with an initial estimate of 8.0. The upper value of 50 is chosen because it represents over 2 and a half times the standard deviation of the data. Values of the smoothing parameter greater than 50 will not affect the estimated distributions H and F substantially. It can happen that there is no optimal value, that is, the loss function decreases as the smoothing parameter increases. In such cases, choosing a large value for the smoothing parameter is appropriate.

The results from the call to samon are placed in the samonResults matrix. A listing of the samonResults follows.

```
> print(samonResults$HM)
     Sample Type Convergence Iterations  SigmaH    lossH
[1,]      0    0           2          4 15.4519 5.398571
> print(samonResults$FM)
     Sample Type Convergence Iterations   SigmaF    lossF
[1,]      0    0           2          3 8.399265 3.618053
```

Samon returns a list. The HM object in this list contains the optimal smoothing parameter for σ_H and FM the optimal smoothing parameter for σ_F. Both HM and FM are arrays. The M in the names HM and FM refers to the main data, i.e. the input data to samon. Later we will see that samon returns similar arrays, HS and FS which correspond to results from bootstrap samples, HMjk and FMjk which contain optimization results for jackknifes for the main data, and HSjk and FSjk, the optimization results for jackknifes of the bootstrap samples.

The first column in the output matrices represents the sample to which the results apply. Since the results here refer to the input data, mat (samonPANSS1), the sample value is set to 0. For the same reason the type value is set to 0. The next column, Convergence, indicates which stopping criteria was used to stop Newton's method. The table below indicates the possible values for Convergence. Iterations gives the number of iterations performed before convergence, and finally, SigmaH gives the optimal smoothing parameter for σ_H and lossH the associated loss function value. In like manner, SigmaF and lossF give the optimal value for σ_F and the value of its loss function at the optimal σ_F.

The optimal value for σ_H is found to be 15.4519 and for σ_F, the optimal value is 8.3993. In a similar manner the optimal values for σ_H and σ_F can be found for treatment group 2. See the program **Example1b.R** in the Examples/Example1 subdirectory for full details.

convergence	
0	absolute step size less than SAconvg
1	absolute function change less than FAconvg
2	relative function change less than FRconvg
3	second derivative of loss function too small (< 1.0E-50) to take step
4	Maximum iterations reached
5	step takes value beyond HighSigmaH or HighSigmaF.
6	value of loss function smallest at HighSigmaH or HighSigmaF

Example 2

In this example we set the selection bias function and create influence function estimates for various values of the parameter alpha. As before we will obtain optimal values for σ_H and σ_F. Much of the call to the samon function is the same as in Example 1. We will need to provide some extra calling parameters to samon and returning to the help we have:

```
samon                    package:samon                    R Documentation

Usage:

  samon(mat, Npart = 10, InitialSigmaH = 1.0, HighSigmaH = 2.0,
  InitialSigmaF = 1.0, HighSigmaF = 2.0, lb = 0, ub = 101, zeta1 = 1,
  zeta2 = 1, NSamples  = 0, seed0 = 1, MaxIter = 25, FAconvg = 1E-7,
  FRconvg = 1E-7, SAconvg = 1E-7, alphaList = c(0), MJackknife = FALSE,
  SJackknife = FALSE, retIFiM = FALSE, retIFiS = FALSE, Tfun= NULL)

Arguments:

     mat: matrix with (i,j) entry representing value for subject i at
          time-point j.

  Npart:  Data is partitioned into Npart parts when when estimating
          optimal sigma p and sigma q.
  ...

alphaList: a vector of sensitivity paramaters.

      lb: Lower bound for Y.

      ub: Upper bound for Y.

   zeta1: parameter to cumulative beta.

   zeta2: parameter to cumulative beta.
```

- The outcome values (PANSS scores in our example) have a bounded range. The selection bias function is of the form $r(x) = \alpha \int_0^x \beta(t, \zeta_1, \zeta_2)dt$ where β is the Beta distribution function with parameters ζ_1 and ζ_2. Consequently the outcome values need to be mapped into the [0,1] interval. This is done using the function $tran(y) = (y - lb)/(ub - lb)$. So we need to choose values lb and ub bounding the outcome values. Since PANSS scores range from 30 to 210 we use these for our bounds. The choice of zeta1 and zeta2 affect the amount of tilting that is done. For now let's set zeta1 to 4 and zeta2 to 7.
- We would like to compute influence function estimates for a range of alpha = -10, -9, ..., -1, 0, 1, ..., 10 so we set alphaList to this. We first run samon on one of the two arms samonPANSS1 and look at the output.

```
# Example2.R
# Produce bias corrected estimates: PANSS group 1.
library(samon, lib.loc="../../samlib")
data(samonPANSS1)

Results1 <- samon(
    mat             = samonPANSS1,
    Npart           = 10,

    InitialSigmaH   = 10.0,    # initial value and
    HighSigmaH      = 50.0,    # high value for H
    InitialSigmaF   = 8.0,     # initial value and
    HighSigmaF      = 50.0,    # high value for F

    lb              = 30,      # parameters to cumulative
    ub              = 210,     # beta distribution
    zeta1           = 4.0,
    zeta2           = 7.0,

    alphaList       = -10:10 )

print(Results1$IFM)
saveRDS(Results1,"Results1.rds")
```

The output from samon is stored in the list Results1. We run this code from the Examples/Example2 subdirectory using the command:

R CMD BATCH –no-save –no-restore Example2.R

```
> print(Results1$IFM)
       Sample Type alpha     AEst        AVar    IFEst    IFVar
  [1,]      0    0   -10 73.80653 0.003674158 74.42125 11.38067
  [2,]      0    0    -9 74.25204 0.004203255 74.93015 11.11602
  [3,]      0    0    -8 74.76982 0.004849061 75.52206 10.87071
  [4,]      0    0    -7 75.37143 0.005630058 76.20172 10.65344
  [5,]      0    0    -6 76.06902 0.006564284 76.96972 10.46880
  [6,]      0    0    -5 76.87548 0.007670310 77.82398 10.31811
  [7,]      0    0    -4 77.80430 0.008965139 78.76183 10.20520
  [8,]      0    0    -3 78.86901 0.010453580 79.78070 10.14286
  [9,]      0    0    -2 80.08139 0.012103703 80.87672 10.15367
 [10,]      0    0    -1 81.44781 0.013809232 82.04413 10.26326
 [11,]      0    0     0 82.96345 0.015360609 83.27915 10.48833
 [12,]      0    0     1 84.60378 0.016481795 84.58388 10.81612
 [13,]      0    0     2 86.31396 0.016986635 85.95778 11.18204
 [14,]      0    0     3 88.00769 0.016942922 87.37985 11.50141
 [15,]      0    0     4 89.59388 0.016585733 88.80761 11.73461
 [16,]      0    0     5 91.01113 0.016091793 90.18900 11.86197
 [17,]      0    0     6 92.23428 0.015535476 91.47237 11.87248
 [18,]      0    0     7 93.26288 0.014954200 92.61856 11.79351
 [19,]      0    0     8 94.11085 0.014382178 93.60797 11.67698
 [20,]      0    0     9 94.79955 0.013853455 94.43884 11.56730
 [21,]      0    0    10 95.35321 0.013393276 95.12201 11.48693
```

The output array of interest here is called IFM, i.e. the influence function estimates for the main or input data.

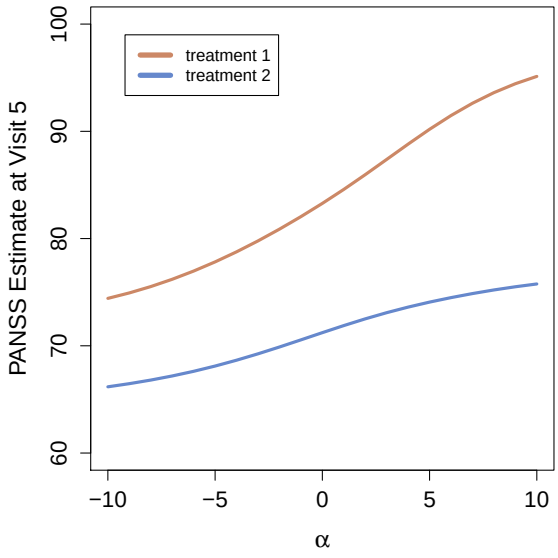
For each alpha there are two types of estimates given. The first is based on the a^* estimator

$$\begin{aligned} a^* &= \int_{y_0}\int_{y_1}\int_{y_2}\cdots\int_{y_K} y_K \prod_{j=1}^{K}\left\{F_j(1-H_j)+\frac{F_j exp(r_j)H_j}{\int exp(r_j)F_j}\right\} \\ &= E_{y_0}\left[\frac{R_K Y_K}{\prod_{j=1}^{K}\left(1+exp(g_j(Y_{j-1})+\alpha r(Y_j))\right)^{-1}}\right] \end{aligned}$$

where $g_j(Y_{j-1}) = log\left(\frac{H_j(Y_{j-1})}{(1-H_j(Y_{j-1}))}\right)$. This is the plug-in estimate (see [2] for details). The second estimate is the influence function estimate and is given in the column named IFEst. Simulation studies have shown that confidence intervals based on the influence function variance leads to confidence intervals with poor coverage. In Example 3 we will compute influence function estimates together with 95% confidence limits using parametric bootstrap and the jackknife.

For now we repeat the above for treatment group 2 and plot the results. The estimates in each case should be a monotone function of the sensitivity parameter alpha.

Figure 2.1: Bias corrected estimates of PANSS score at visit 5 by sensitivity parameter α.



Example 3

We are now ready to produce confidence bands for the PANSS data. Samon uses the bootstrap and the jackknife in combination to produce confidence intervals. Again we return to the samon function this time asking for N bootstrap samples and jackknife estimates. Here are the arguments to samon to request the bootstrap and jackknife:

```
samon                    package:samon                    R Documentation

Description:

  Given data from one arm of a repeated measures clinical trial, produces
  estimates of the expected value of the outcome at the final time-point
  for a range of sensitivity parameters.

Usage:

  samon(mat, Npart = 10, InitialSigmaH = 1.0, HighSigmaH = 2.0,
  InitialSigmaF = 1.0, HighSigmaF = 2.0, lb = 0, ub = 101, zeta1 = 1,
  zeta2 = 1, NSamples  = 0, seed0 = 1, MaxIter = 25, FAconvg = 1E-7,
  FRconvg = 1E-7, SAconvg = 1E-7, alphaList = c(0), MJackknife = FALSE,
  SJackknife = FALSE, retIFiM = FALSE, retIFiS = FALSE, Tfun= NULL)

Arguments:

    mat: matrix with (i,j) entry representing value for subject i at
         time-point j.
  ...

NSamples: Number of bootstrap samples to generate.

seed0: Seed to use.

MJackknife: Jackknife main data (logical)

SJackknife: Jackknife bootstrap samples (logical)
```

Most of the parameters passed to samon are identical to those used in the previous example. Suppose we wish to create 1,000 bootstraps in each arm of the PANSS data. Instead of running one program to achieve this, we split the task across 4 programs each requesting 500 bootstraps for one treatment arm. In this way we can distribute the job across 4 processors, saving run time. We do need to make sure to use different seeds in each program and to save the results under different names. Since there are only 4 programs we can write and run the programs manually. The 4 programs are Example3_1a.R, Example3_1b.R, which produces bootstraps for treatment 1 (placebo) and Example3_2a.R, and, Example3_2b.R which produces bootstraps for treatment 2 (active arm). Here is Example3_1a.R:

```
# Example3_1a.R
# Produce bias corrected influence function estimates
# and 500 bootstrap estimates for treatment 1.
# ------------------------------------------------

library(samon, lib.loc="../../samlib")

# Treatment 1 data.
data("samonPANSS1")

Results1a <- samon(
    mat             = samonPANSS1,
    Npart           = 10,

    InitialSigmaH   = 10.0,          # initial value
    HighSigmaH      = 50.0,          # high value for H

    InitialSigmaF   = 8.0,           # initial value
    HighSigmaF      = 50.0,          # high value for F

    lb              = 30,            # parameters to cumulative
    ub              = 210,           # beta distribution
    zeta1           = 4.0,
    zeta2           = 7.0,

    NSamples        = 500,           # number of bootstraps
    seed0           = 81881,

    MJackknife      =  TRUE,         # jackknife main data
    SJackknife      =  TRUE,         # jackknife bootstraps

    alphaList       =  alphaList
                     )

# save results for later use
saveRDS(Results1a, "Results1a.rds")
```

Program Example3_1b.R is the same except that it uses a different seed and saves the results in Results1b.rds. The other two programs, Example3_2a.R and Example3_2b.R also change the seed and the name of the output and in addition read the data for treatment 2 instead of treatment 1. The 4 programs are in the Example3 folder. To run the programs we issue the commands:

R CMD BATCH –no-save –no-restore Example3_1a.R
R CMD BATCH –no-save –no-restore Example3_1b.R
R CMD BATCH –no-save –no-restore Example3_2a.R
R CMD BATCH –no-save –no-restore Example3_2b.R

Once the bootstraps have been produced the results can be combined to produce a single samon object using the samonCombine function. Other functions are then used to produce confidence intervals for estimates and differences in estimates. Here is a list of some samon functions. We will illustrate their use in Example 3 and 4.

function	**description**
samonCombine	combines the outputs from samon into one samon object. The results are stored in rds files. samonCombine takes a list of such files and combines them.
samonSummary	a function to summarize the results from a call to samon. This combines the jackknife and bootstrap estimates to produce treatment estimates and confidence intervals.
samonDifferenceSummary	Takes two samonSummary objects and computes a difference in treatment estimate with confidence intervals for each value of the sensitivity parameter.
samonCrossSummary	Similar to the samonDifferenceSummary function. Takes two samonSummary objects (one from each treatment group) and computes a difference in treatment estimate with confidence interval for each pair of sensitivity parameters. This is used to produce the contour plot.

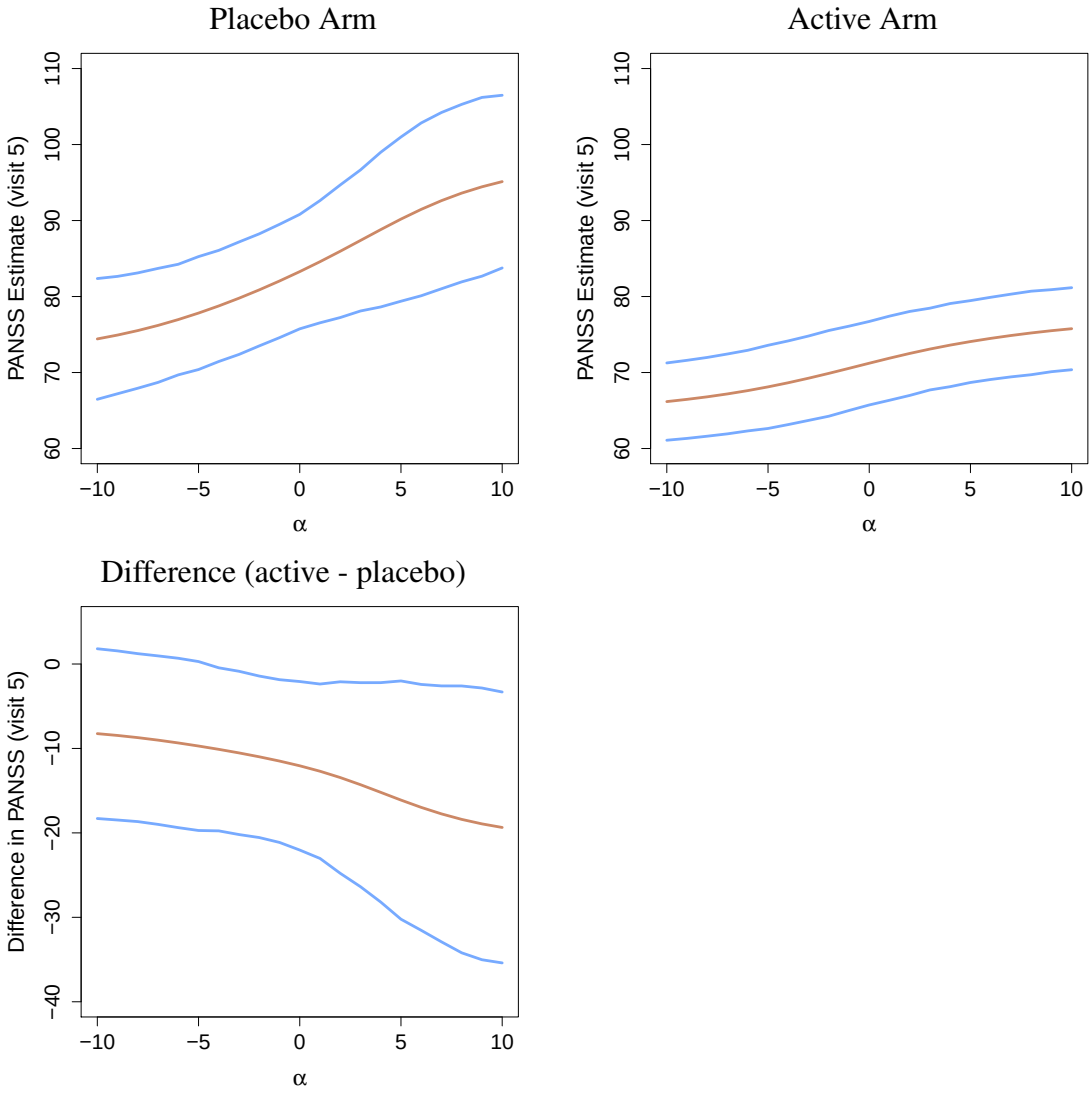
When samon have been run in parallel as it has been in Example 3, the samonCombine function is useful in gathering up the pieces and building a single samon object. We have produced 4 rds files each containing the results from 500 bootstraps. The program Example3_plots.R uses samonCombine to put the results together and the function samonSummary to summarize the estimates from each treatment arm. The samonDifferenceSummay function takes the two summary objects produced by samonSummary and produces the difference in influence function estimates between treatments (treatment2 - treatment1). Finally plots are produced showing influence function estimates as a function of the sensitivity parameter, alpha, together with a 95% confidence interval. In Example 4 we will repeat this exercise increasing the number of bootstraps in each treatment group. Here is Example3_plots.R which uses the functions mentioned above:

```
# Example3_plots.R
# Gather previously run results and plot them using samonPlot.
#
# Results are stored in rds files
# treatment 1:   Results1a.rds,  Results1b.rds
# treatment 2:   Results2a.rds,  Results2b.rds
# ------------------------------------------------------------
library(samon,lib.loc="../../samlib")

# Retrieve the results and combine them
filenames1 <- c("Results1a.rds", "Results1b.rds")
filenames2 <- c("Results2a.rds", "Results2b.rds")

trt1Results <- samonCombine( filenames1 )
trt2Results <- samonCombine( filenames2 )

# summarize
Summary1 <- samonSummary( trt1Results )
Summary2 <- samonSummary( trt2Results )
SummaryD <- samonDifferenceSummary( Summary1, Summary2 )

# Get estimates and confidence intervals
TM1 <- Summary1$TM[,c("alpha","IFEst")]
TM2 <- Summary2$TM[,c("alpha","IFEst")]
TMD <- SummaryD$TM[,c("alpha","Difference")]
CI1 <- Summary1$CI[,c("lb8","ub8")]
CI2 <- Summary2$CI[,c("lb8","ub8")]
CID <- SummaryD$CI[,c("lb8","ub8")]

Results1 <- cbind( TM1, CI1)
Results2 <- cbind( TM2, CI2)
ResultsD <- cbind( TMD, CID)

# treatment 1
samonPlot(Results1, "Trt1Est.pdf", 5.5,6, "PANSS Estimate (visit 5)",
   c(-20,20), c(60,110), c(3.6,63.0), maintext = "Placebo" )

# treatment 2
samonPlot(Results2, "Trt2Est.pdf", 5.5,6, "PANSS Estimate (visit 5)",
   c(-20,20), c(60,110), c(3.6,63.0), maintext = "Active" )

# and treatment 2 minus treatment 1
samonPlot(ResultsD, "TrtDEst.pdf", 5.5,6,
  "Difference in PANSS (visit 5)", c(-20,20), c(-40,5), c(3.6,-9.0),
   maintext = "Difference (Active minus Placebo)" )
```

Example3_plots also employs the samonPlot function given here:

```
# function samonPlot
# Takes a matrix of results, Res, with Res[,1] to be plotted on the
# xaxis, Res[,2] to be plotted on the yaxis.  Res[,3] and Res[,4]
# contain lower and upper bounds to be plotted as a band.
samonPlot <- function( Res, file, height, width,
                       ylab, xlim, ylim, legpos, maintext) {

  pdf(file=file, height=height, width=width)
  par(mar=c(4,4,2,2))
  plot.new()
  plot.window( xlim = xlim, ylim = ylim )

  cols <- c("#CC8866FF","#77AAFFFF")
  lines( x=Res[,1], y=Res[,3], lwd=3, lty=c("solid"), col=cols[2])
  lines( x=Res[,1], y=Res[,4], lwd=3, lty=c("solid"), col=cols[2])
  lines( x=Res[,1], y=Res[,2], lwd=3, col=cols[1])

  axis(1);  axis(2)

  title( main = maintext, xlab = expression(alpha),  ylab = ylab)
  box()

  dev.off()
  invisible(return())
}
```

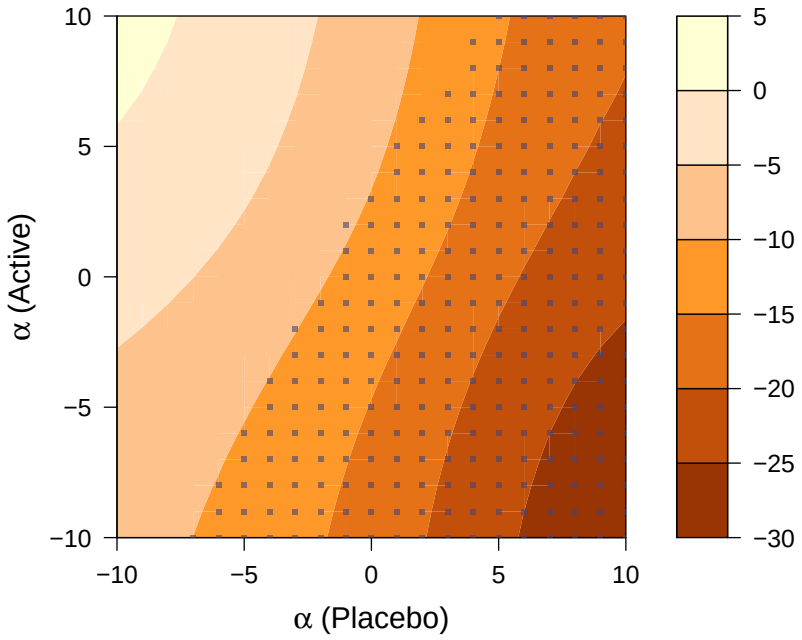
Figure 3.1: Bias corrected estimates of PANSS scores at visit 5 by sensitivity parameter α with 95% confidence intervals. The control arm is on the top left, the intervention arm on the top right, and, the effect estimate (the difference between the intervention arm and the control arm) is on the bottom left.



Another useful plot is a surface plot of the difference in the estimated mean value between the two treatment groups given as a function of the two sensitivity parameters (one for each treatment group). We use the samonCrossSummary function to compute the difference in estimates for each pair of alphas. The plotting is done with the filled.contour function.

```
# Example3_contourPlot.R
# ------------------------------------
library(samon,lib.loc="../../samlib")


# the first two files are for treatment 1, the second
# two are for treatment 2.
filenames1 <- c("Results1a.rds", "Results1b.rds")
filenames2 <- c("Results2a.rds", "Results2b.rds")


trt1Results <- samonCombine( filenames1 )
trt2Results <- samonCombine( filenames2 )


# summarize
Summary1 <- samonSummary( trt1Results )
Summary2 <- samonSummary( trt2Results )


XRes <- samonCrossSummary( Summary1, Summary2 )
XRes <- as.matrix(XRes$CI)


pdf(file="Example3_contour.pdf", height=5, width=6)
par(mar=c(4,4,2,2))


filled.contour(
  x       = -20:20,
  y       = -20:20,
  z       = matrix(XRes[,3],41,41, byrow=TRUE),
  xlab    = expression(paste(alpha, " (Placebo)")),
  ylab    = expression(paste(alpha, " (Active)")),
  nlevels = 8,
  color.palette = colorRampPalette(c( "#993404","#D95F0E",
                  "#FE9929","#FFD9BE","#FFFFD4"), space="rgb"),
  plot.axis = (
               points( XRes[ sign(XRes[,18]) == sign(XRes[,19]),
                             c(1,2)],
                       pch=15, cex=0.6, col = c("#44447799")))
             )
dev.off()
```

Figure 3.2: Contour plot showing the estimated difference in bias corrected PANSS scores (intervention - control). The sensitivity parameter α is allowed to differ between arms. Black markers appear where the estimated difference in employment rates is statistically significant at the 95% confidence level.



The contour plot shows the estimated difference in mean PANSS scores at visit 5. The difference is the mean for the active arm minus the mean for the placebo arm. The dots indicate whether the treatment difference would be statistically significant at the 0.05 level.

Example 4

In the previous example it may be felt that 1,000 bootstrap samples may not be enough to estimate 95% confidence intervals with precision. However, increasing the number of samples may result in a program that takes a long time to complete. One possibility, if a number of processors are available, is to split the program into parts and run the parts in parallel. In this example we will see how to go about this.

Here we outline the procedure used employing a SUN grid engine (sge). The technique may be modified for use on other systems. When we submit a job to the Grid we can request multiple copies of the job to be run in parallel. Each of these copies are identical but are run in an environment with an environment variable called SGE_TASK_ID unique to each instance. We write a single R program which consults this environment variable and so alters the seed and output name depending on the value it finds there.

Here we run samon on the PANSS treatment 1 data and have our programs generate 50 bootstraps. We let SGE run 50 instances of our program each in an environment having the environment variable SGE_TASK_ID set to the values 1 to 50. Since we do not wish to compute the jackknife values for the main data in each instance of our program we request that MJackknife be set to TRUE when SGENID is 1 and to FALSE otherwise.

```
# samonRuns1.R
# Produce 50 bootstraps each with jackknife estimates.
# Both the output filename and the seed used are based on the
# SUN grid engine task id.
# ------------------------------------------------------------------
# Get SGE_TASK_ID from the envirnonment.
SGETID   <- Sys.getenv(c("SGE_TASK_ID"))
SGENID   <- as.numeric(SGETID)
oname    <- sprintf("RDS/Sample1/results_%05d.rds", SGENID)

set.seed(826847827)
seedList <- ceiling( 1000000 * runif( 1000 ) )
seed     <- seedList[SGENID]

# only do the jackknife on the main data once
MJK <- ( SGENID == 1 )

library(samon, lib.loc="../../samlib")
data("samonPANSS1")
```

. . . continued

```
# samonRuns1.R (continued from previous page)
# ------------------------------------------------------------------
Results <- samon(
   mat             = samonPANSS1,   # imput matrix
   Npart           =          10,   # number of partitions

   InitialSigmaH   =        10.0,   # initial value
   HighSigmaH      =        50.0,   # high value for H
   InitialSigmaF   =         8.0,   # initial value
   HighSigmaF      =        50.0,   # high value for q

   lb              =          30,   # parameters for cumulative
   ub              =         210,   # beta distribution
   zeta1           =         4.0,
   zeta2           =         7.0,

   NSamples        =          50,   # bootstraps
   seed0           =        seed,

   MJackknife      =         MJK,
   SJackknife      =        TRUE,

   alphaList       =      -10:10  )

saveRDS(Results, file=oname)  # save the results
```

Since the environment variable SGE_TASK_ID takes on the values 1 to 50 we use it as a pointer into a vector of seeds. Since the seed is explicitly set the results can be reproduced exactly if needed. The output names have the form RDS/Sample1/results_00001.rds, RDS/Sample1/results_ 00002.rds, . . . , RDS/Sample1/results_00050.rds.

Having run 50 instances of this program and 50 of a similar program for treatment 2 we are in a position to gather the results.

```
# together.R
# Put together results from various runs of samon.
# Makes  treatment1Results.rds  with treatment 1 results
#        treatment2Results.rds  with treatment 2 results
# ------------------------------------------------------------
library(samon,lib.loc="../../samlib")

# there are 100 files each with 50 bootstraps per treatment arm.
filenames1 <- sprintf("RDS/Sample1/results_%05d.rds", 1:100 )
filenames2 <- sprintf("RDS/Sample2/results_%05d.rds", 1:100 )

trt1 <- samonCombine( filenames1 )
trt2 <- samonCombine( filenames2 )

saveRDS( trt1, file = "treatment1Results.rds")
saveRDS( trt2, file = "treatment2Results.rds")
```

Plots and other outputs are produced in a similar manner to that in Example 3.

Figure 4.1: Bias corrected estimates of PANSS scores at visit 5 by sensitivity parameter α with 95% confidence intervals. The control arm is on the top left, the intervention arm on the top right, and, the effect estimate (the difference between the intervention arm and the control arm) is on the bottom left.

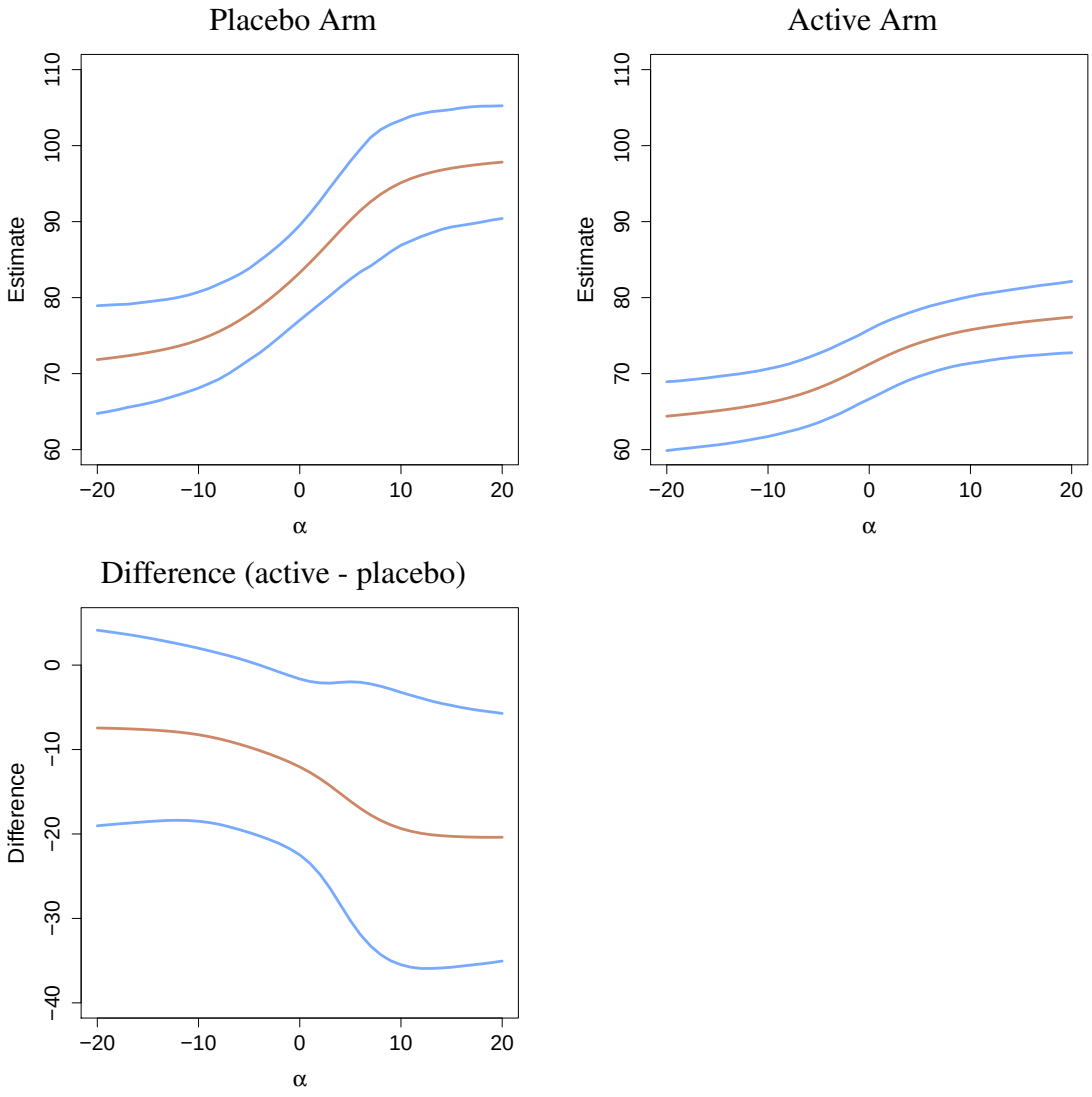
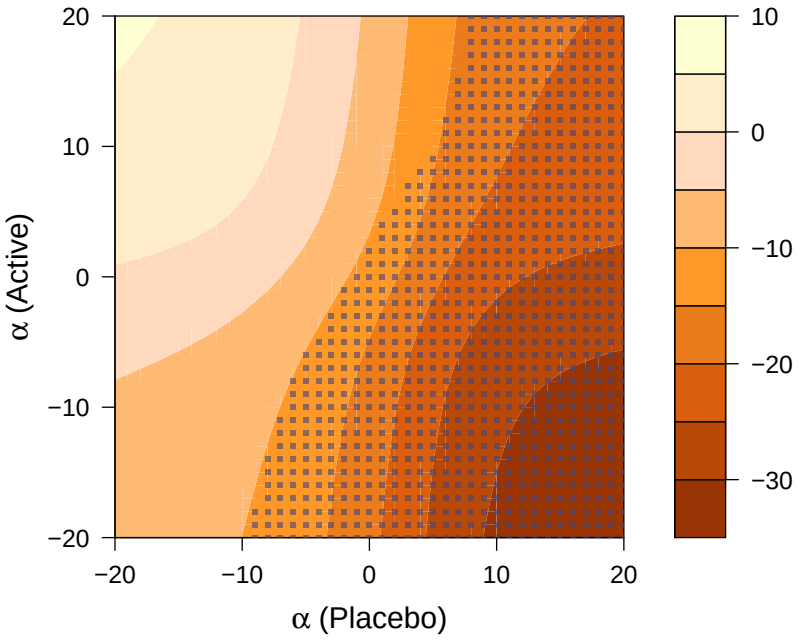


Figure 4.2: Contour plot showing the estimated difference in bias corrected PANSS scores (intervention - control). The sensitivity parameter α is allowed to differ between arms. Black markers appear where the estimated difference in employment rates is statistically significant at the 95% confidence level.



Comparing the contour plot from this example with that produced in Example 3, you may notice that the contour lines are not identical – although they are close. This is because the IF estimates depend on the bootstrap samples and in this example we have increased the number of these samples from 1,000 in each treatment arm of Example 3 to 5,000 in each treatment arm of Example 4. See [1,2] for further details.

Example 5

We consider the data from two arms of a clinical trial where the primary outcome of interest is a measure of neurological pain using the visual analog scale (VAS). Diabetic subjects were enrolled in a study to evaluate the efficiency of treatment with the drug Topiramate. During a baseline phase subjects were evaluated for eligibility and tapered off existing neuropathic medication. Having been free of neuropathic pain medication for at least seven days, subjects were randomized to one of 4 arms. The data we consider here are from two of these arms – a placebo arm and subjects taking 200 mg of Topiramate. See [**3**] for further details.

Data from the placebo arm can be found in the VAS1 dataset and data from the Topiramate (200 mg) arm can be found in the VAS2 dataset within the samon package.

The double-blind phase included 2 periods, a 10 week titration period and a 12 week maintenance period. VAS scores were scheduled on day 1 of the baseline period, every two weeks during titration, and every four weeks of the maintenance phase.

Treatment effects were based on the VAS scores at the final follow-up visit. The data exhibit both dropout and intermittent missing data.

In this chapter we analyze some clinical trial data with intermittent missingness and dropout. We'll chiefly use the SAMONIM function in the SAMON package.

We begin by examining the missing data patterns in the two arms of the VAS data. The placebo arm has 255 subjects while the Topiramate arm has 256 subjects with measurements taken at 9 time-points in total. Among the placebo arm, 81 individuals have data at all time-points (32%) while 152 display what might be termed a monotone missing (non-intermittent) structure. Among those with intermittent missing data 82 are missing data at time-point 2, the time-point immediately after the baseline measure. In the Topiramate arm there were 67 completers (25%) while 251 display a monotone missing pattern. Among those with intermittent missing data 60 are missing data at time-point 2. Tables 5.1 and 5.2 list the the most frequently occurring data patterns. In these tables an asterisk ($*$) represents data being available at a time point while an underline (_) represents missing data.

Table 5.1: Some Missing data patterns from the placebo arm.

```
Monotone missing patterns:
                     N   proportion
*_________  :         5       0.0196
**________  :         5       0.0196
***_______  :        10       0.0392
****______  :         3       0.0118
*****_____  :        19       0.0745
******____  :        12       0.0471
*******___  :        12       0.0471
********__  :         5       0.0196
*********  :        81       0.3176

Intermittent missing patterns:
                     N   proportion
*_*_*****  :        14       0.0549
*_*******  :        13       0.0510
****_****  :         7       0.0275
***_*****  :         6       0.0235
******_**  :         5       0.0196
Other      :        47
```

Table 5.2: Some Missing data patterns from the Topiramate arm.

```
Monotone missing patterns:
                     N   proportion
*_________  :         4       0.0156
**________  :        14       0.0547
***_______  :        19       0.0742
****______  :         7       0.0273
*****_____  :        19       0.0742
******____  :        10       0.0391
*******___  :         9       0.0352
********_  :         2       0.0078
*********  :        67       0.2617

Intermittent missing patterns:
                     N   proportion
*_*_______  :        15       0.0586
*_*_*****  :         9       0.0352
*_*******  :         8       0.0312
***_*****  :         7       0.0273
*_*_***__  :         5       0.0195
Other      :        56
```

Examining Tables 5.3 and 5.4 we see that in the placebo group the mean value of VAS at baseline was 58.9 while the mean value at time point 9 for those observed at that time point is 35.6. The VAS mean at baseline among the Topiramate group is 58.3 and at the last time point it was 31.5.

Table 5.3: Number of observed values at each time-point, mean and standard deviation of observed outcome - Placebo arm.

			Last Observed			Intermittent Missing			
T	On-Study	N Obs	N	% of N On-Study	% of N Observed	N	% of N On-Study	Observed mean	SD
1	255	255	5	1.96	1.96			58.902	19.196
2	250	188	5	2.00	2.66	62	24.80	53.202	23.048
3	245	238	14	5.71	5.88	7	2.86	48.899	24.888
4	231	186	5	2.16	2.69	45	19.48	45.849	23.928
5	226	203	27	11.95	13.30	23	10.18	42.291	25.338
6	199	192	24	12.06	12.50	7	3.52	38.896	25.117
7	175	162	15	8.57	9.26	13	7.43	37.549	25.827
8	160	150	10	6.25	6.67	10	6.25	35.047	26.313
9	150	150	150	100.00	100.00			35.613	26.446

Table 5.4: Number of observed values at each time-point, mean and standard deviation of observed outcome - Topiramate arm.

			Last Observed			Intermittent Missing			
T	On-Study	N Obs	N	% of N On-Study	% of N Observed	N	% of N On-Study	Observed mean	SD
1	256	256	4	1.56	1.56			58.305	19.958
2	252	192	14	5.56	7.29	60	23.81	51.297	22.605
3	238	223	34	14.29	15.25	15	6.30	47.466	25.268
4	204	162	12	5.88	7.41	42	20.59	44.228	22.956
5	192	174	28	14.58	16.09	18	9.38	41.879	23.851
6	164	159	26	15.85	16.35	5	3.05	36.528	24.101
7	138	133	20	14.49	15.04	5	3.62	36.211	24.334
8	118	109	6	5.08	5.50	9	7.63	33.138	21.842
9	112	112	112	100.00	100.00			31.482	22.149

Figure 5.1: Mean VAS score at visit t stratified by visit at which individuals were last observed.

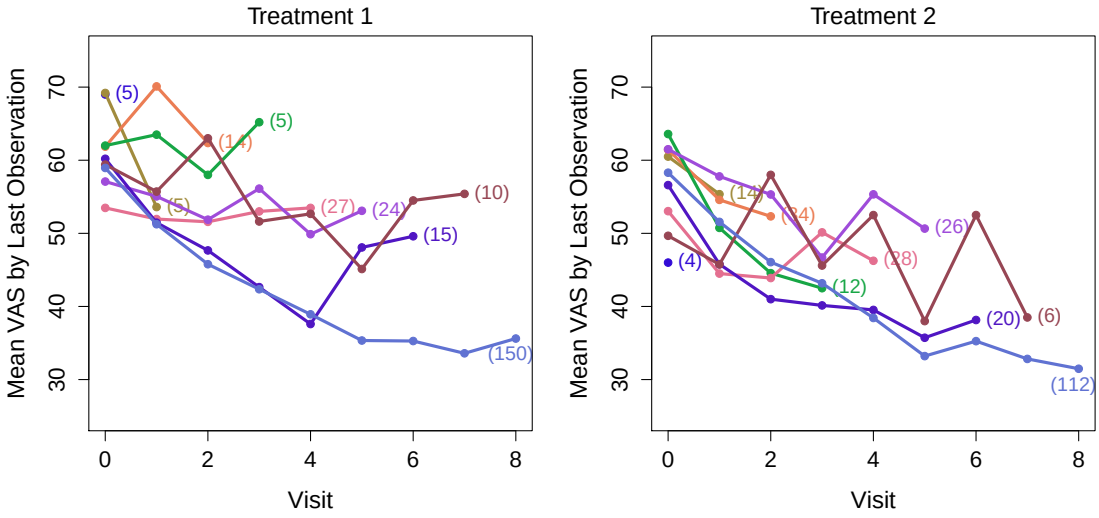


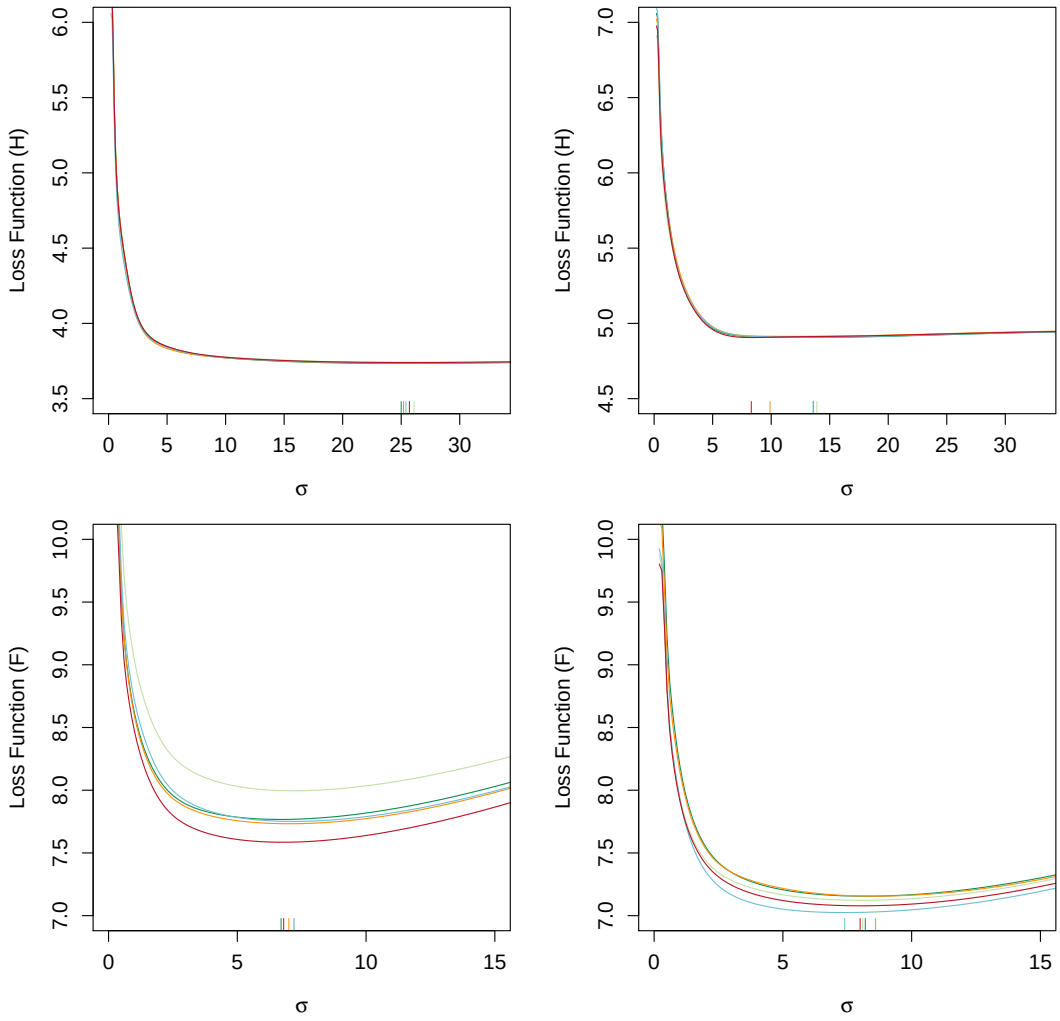
Figure 5.1 shows the mean value among individuals who provided data at a time point stratified by the time they were last observed.

Optimal Smoothing Parameters

Using multiple imputation we estimate the smoothing parameters within each group. Using 5 imputations Figure 5.2 shows the loss function as a function of smoothing parameter σ_H. It can be observed that the relationship between the loss and σ_H is similar among imputes and results in similar minimal values. The situation in the other treatment arm is similar. The situation involving the other smoothing parameter σ_F shows more variation in both treatment arms and the minimal loss value is more clearly defined.

```
# HF.R
# Examining the loss functions.
# For a range of smoothing parameters produce data for loss by
# smoothing parameter.  Multiple imputations are used.
# ------------------------------------------------------------
library(samon, lib.loc="../../samlib")
sigmaList <- seq(0.2,40,by=0.1)

data("VAS1"); data("VAS2") # the data

# create a NT by 6 matrix indicating which variables to
# include in logistic regression.  Here use all 6 except
# at the NT-1 time-point when we include only the first 3.
inmodel     <- matrix(1,9,6)
inmodel[1,] <- 0
inmodel[9,] <- 0
inmodel[9-1,4:6] <- 0

for ( impute in 1:5 ) {
  seed <- 3121 + impute

  HF1 <- samonevalIM(
         mat       = VAS1,       Npart = 10,     sigmaList = sigmaList,
         inmodel   = inmodel,    seed  = seed,   type      = "both" )

  HF2 <- samonevalIM(
         mat       = VAS2,       Npart = 10,     sigmaList = sigmaList,
         inmodel   = inmodel,    seed  = seed,   type      = "both" )

  o1 <- cbind( impute, 1, HF1$OutSig )
  o2 <- cbind( impute, 2, HF2$OutSig )

  if ( impute == 1 ) {
    out1 <- o1;    out2 <- o2
  } else {
    out1 <- cbind( out1, o1 )
    out2 <- cbind( out2, o2 )
  }
}
saveRDS(out1,"RDS/HF1.rds")
saveRDS(out2,"RDS/HF2.rds")
```

Figure 5.2: Loss function by smoothing parameter. The control arm is on the left, the intervention arm on the right. Smoothing associated with dropout is on the top row and smoothing associated with outcome is on the bottom.



Bias corrected estimates

Within each treatment arm we estimate bias corrected estimates using the samonIM function. Figure 5.3 displays the results as a function of the sensitivity parameter α. Confidence intervals are calculated using parametric bootstraps with multiple imputation employed within each bootstrap. There were 2000 bootstraps each with 5 imputations involved in computing these confidence intervals.

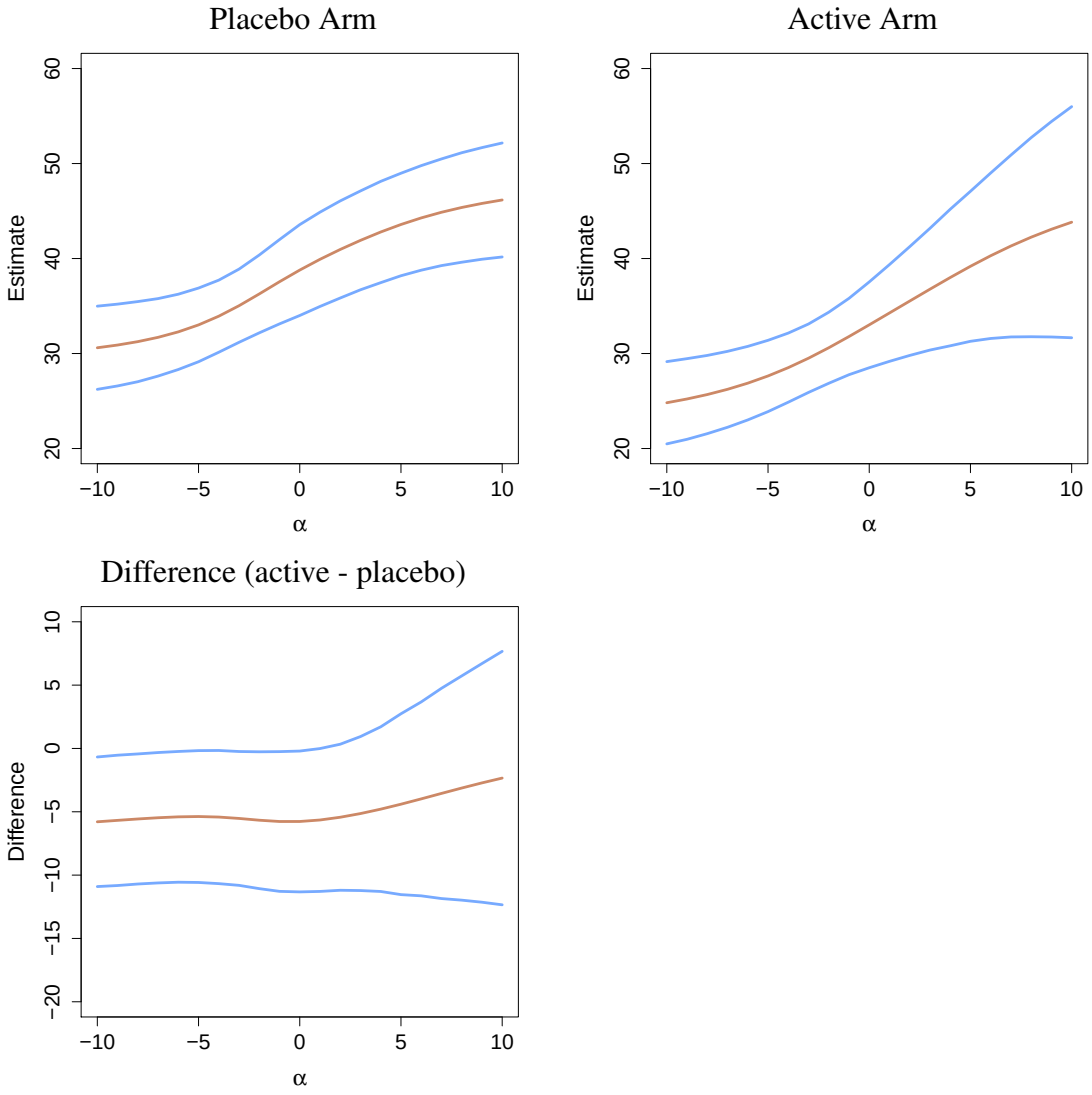
```
# samonIM.R
# ------------------------------------------------------------
# Create bias corrected estimates for treatment 1 of the VAS
# data.  2000 bootstraps using 5 imputes.
# ------------------------------------------------------------
library(samon, lib.loc="../../samlib")

# Retrieve the data
data("VAS1")

# create a NT by 6 matrix indicating which variables to
# include in logistic regression.  Here use all 6 except
# at the NT-1 time-point when we include only the first 3.
inmodel        <- matrix(1,9,6)
inmodel[1,]    <- inmodel[9,]  <- 0
inmodel[8,4:6] <- 0

Results <- samonIM(
    mat             =      VAS1,   # imput matrix
    Npart           =        10,   # number of partitions

    InitialSigmaH   =      25.0,   # initial value and
    HighSigmaH      =     100.0,   # high value for h

    InitialSigmaF   =       6.5,   # initial value and
    HighSigmaF      =     100.0,   # high value for f

    lb              =         0,   # parameters for
    ub              =       102,   # cumulative
    zeta1           =       1.2,   # beta distribution
    zeta2           =       1.6,

    NSamples        =      2000,   # bootstraps
    NIMimpute       =         5,   # number of inputes

    seed0           =      2122,   # bootstrap seed
    seed1           =       281,   # imputation seed
    inmodel         =   inmodel,   # input model

    alphaList       =    -10:10    # sensitivity parameters
                     )
saveRDS(Results, file=oname)
```

As with the samon procedure there are a number of functions that can be used to combine samonIM objects and produce confidence intervals from the bootstrap and imputation results. These functions should be used with the outputs from the samonIM function.

function	**description**
samonCombineIM	combines the outputs from samonIM into one samonIM object. The results are stored in rds files. samonIMCombine takes a list of such files and combines them.
samonSummaryIM	a function to summarize the results from a call to samonIM. This deals with the imputed and bootstrap estimates to produce treatment estimates and confidence intervals.
samonDifferenceSummaryIM	Takes two samonIMSummary objects and computes a difference in treatment estimate with confidence intervals for each value of the sensitivity parameter.
samonCrossSummaryIM	Similar to the samonIMDifferenceSummary function. Takes two samonIMSummary objects (one from each treatment group) and computes a difference in treatment estimates with confidence interval for each pair of sensitivity parameters. This is used to produce the contour plot.

```
# summaryIM.R
# ---------------------------------------------------------------
# Summarize the results for the VAS data.
# ---------------------------------------------------------------
library(samon, lib.loc="../../samlib")

trt1Results <- readRDS("RDS/treatment1Results.rds")
trt2Results <- readRDS("RDS/treatment2Results.rds")

# CI for treatment 1 and 2 respectively
TM1 <- samonSummaryIM(trt1Results)
TM2 <- samonSummaryIM(trt2Results)

# Difference in outcome at same value of alpha
DM <- samonDifferenceSummaryIM(TM1,TM2)
# Difference in outcome at different values of alpha
CM <- samonCrossSummaryIM(TM1,TM2)

Results <- list( TM1 = TM1, TM2 = TM2, DM = DM, CM = CM )
# save results
saveRDS(Results, "RDS/Results.rds" )
```

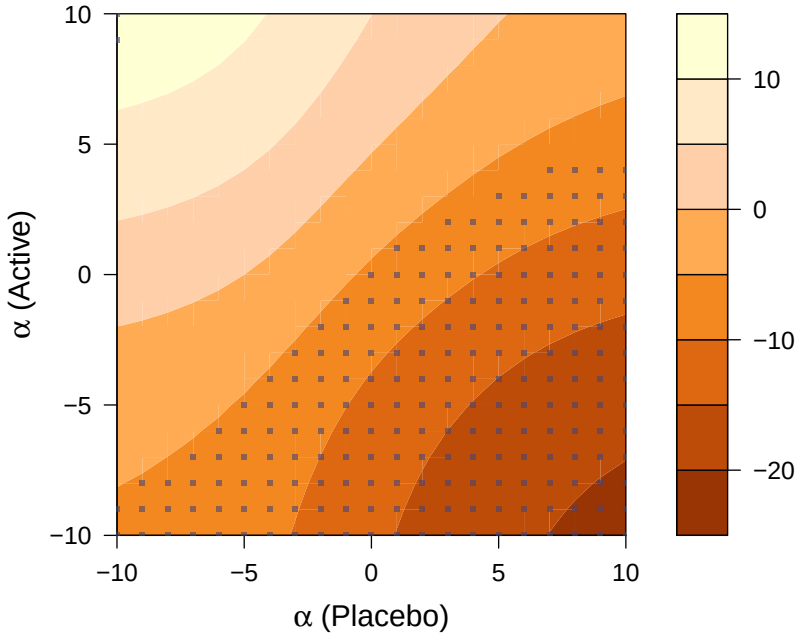
Figure 5.3: Bias corrected estimates of VAS scores at visit 9 by sensitivity parameter α with 95% confidence intervals. The control arm is on the top left, the intervention arm on the top right, and, the effect estimate (the difference between the intervention arm and the control arm) is on the bottom left.



Considering the difference in estimates (Topiramate arm minus placebo arm) using the same sensitivity parameter in each treatment arm using parametric bootstraps and multiple imputation to compute confidence intervals we arrive at Figure 5.3.

Employing different sensitivity parameters in the two arms and computing the difference we obtain the contour plot:

Figure 5.4: Contour plot showing the estimated difference in bias corrected VAS scores at visit 9 (intervention - control). The sensitivity parameter α is allowed to differ between arms. Black markers appear where the estimated difference in employment rates is statistically significant at the 95% confidence level.



Example 6

The DepWork data is a subset of the JOBSII dataset (see Vinokur **4**). JOBSII was an intervention study designed to measure the effect of an education program on job search strategies, interview techniques and resume building. Unemployed individuals were randomly assigned to one of two groups, a control group (N = 552) and a treatment group (N = 1249). Individuals assigned to the control group were given a pamphlet outlining job search strategies. The treatment group were invited to a job search seminar. The seminar series was provided over a one-week period and in all lasted for 20 hours. In the treatment group 39% did not attend the seminar. Employment status for each individual was measured at 6 weeks after baseline, 6 months after baseline and 2 years after baseline. An individual was considered employed if they worked for 20 hours or more a week.

The dataset DepWork1 contains the data for the control arm and DepWork2 the data for the treatment arm. The data contain intermittent missing data. That is, for some individuals their employment status may be missing at one time-point but be available at some subsequent time-point.

We use the samonIM function in the samon package to analyze data with intermittent missingness. This function uses multiple imputation of the intermittent missing data. Normally only a small number of imputations are needed (about 5). Each imputed dataset has a monotone missing data structure and can be analyzed using the techniques used in the preceding examples. However the use of the jackknife is not advisable under these circumstances. Instead, samonIM induces intermittent missing data within parametric bootstraps and performs multiple imputation on data thus produced. Results from the different imputations are pooled to produce standard errors.

The focus of this analysis is on the mean employment rate at 2 years from baseline within the control group and the intervention group and the treatment effect, the difference in the mean employment rates between the two groups.

We begin by examining the missing data patterns in the two arms of the DepWork data. The control arm has 552 subjects while the treatment arm has 1249 subjects with measurements taken at 4 time-points in total.

Among the control arm, 349 have data at all time-points (63%) while 449 in all have no

intermittent missing data. Among those with intermittent missing data 50 are missing data at time-point 2, that is, at 6 weeks after follow-up. In the treatment arm 743 have complete data (59%) while 962 have no intermittent missing data. Among those with intermittent missing data 144 are missing data at the second time-point.

The frequency tables 6.1 and 6.2 list the the most frequently occurring data patterns. In these tables an asterisk ($*$) represents data being available at a time point while an underline (_) represents missing data.

Table 6.1: Some Missing data patterns from the control arm.

```
Monotone missing patterns:
                 N   proportion
*___   :        30       0.0543
**__   :        17       0.0308
***_   :        53       0.0960
****   :       349       0.6322

Intermittent missing patterns:
                 N   proportion
*__*   :        15       0.0272
*_*_   :        15       0.0272
*_**   :        50       0.0906
**_*   :        23       0.0417
```

Table 6.2: Some Missing data patterns from the intervention arm.

```
Monotone missing patterns:
                 N   proportion
*___   :        78       0.0624
**__   :        34       0.0272
***_   :       107       0.0857
****   :       743       0.5949

Intermittent missing patterns:
                 N   proportion
*__*   :        38       0.0304
*_*_   :        56       0.0448
*_**   :       144       0.1153
**_*   :        49       0.0392
```

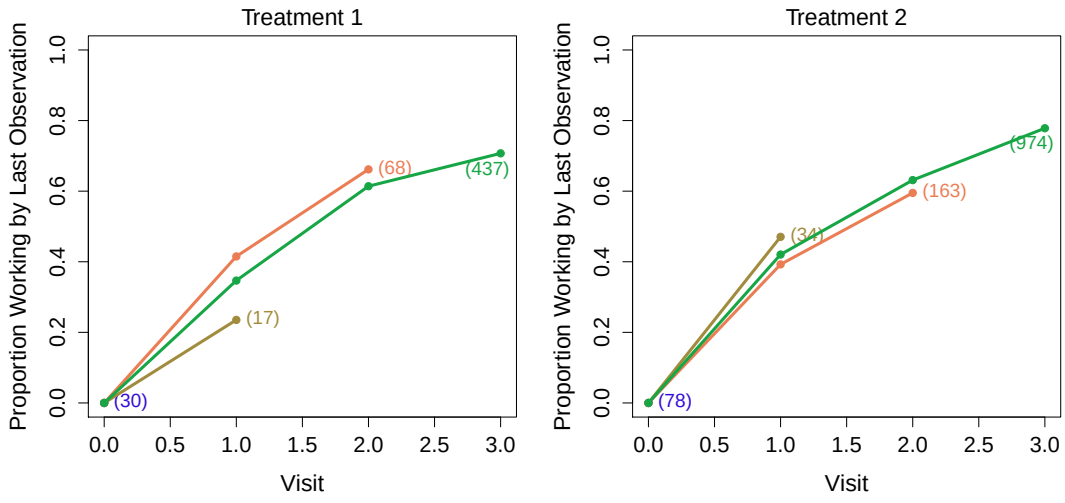
Examining Table 6.3 we see that the observed employment rate rises from 0 at baseline to 0.71 at 2 years after baseline among the control group, while, the observed employment rate among the intervention group rises from 0 at baseline to 0.78 at 2 years after baseline.

Table 6.3: Number of individuals on-study and number observed values at each time-point together with the number of intermittent missing data-points and the mean and standard deviation of observed outcome.

			Last Observed			Intermittent Missing			
T	On-study	N Obs	N	% of N On-study	% of N Observed	N	% of N On-study	Observed Mean	SD
The Control Group									
1	552	552	30	5.43	5.43			0.000	0.000
2	522	442	17	3.26	3.85	80	15.33	0.351	0.478
3	505	467	68	13.47	14.56	38	7.52	0.621	0.486
4	437	437	437	100.00	100.00			0.707	0.456
The Intervention Group									
1	1249	1249	78	6.24	6.24			0.000	0.000
2	1171	933	34	2.90	3.64	238	20.32	0.419	0.494
3	1137	1050	163	14.34	15.52	87	7.65	0.626	0.484
4	974	974	974	100.00	100.00			0.778	0.416

Alternatively, we can examine the data displayed in Figure 6.1 where the employment rate is plotted against time separately for each treatment arm and for the time-point at which an individuals were last observed.

Figure 6.1: Employment rate by time-point for control arm (left panel) and intervention arm (right panel) and by the time-point at which individuals were last observed.

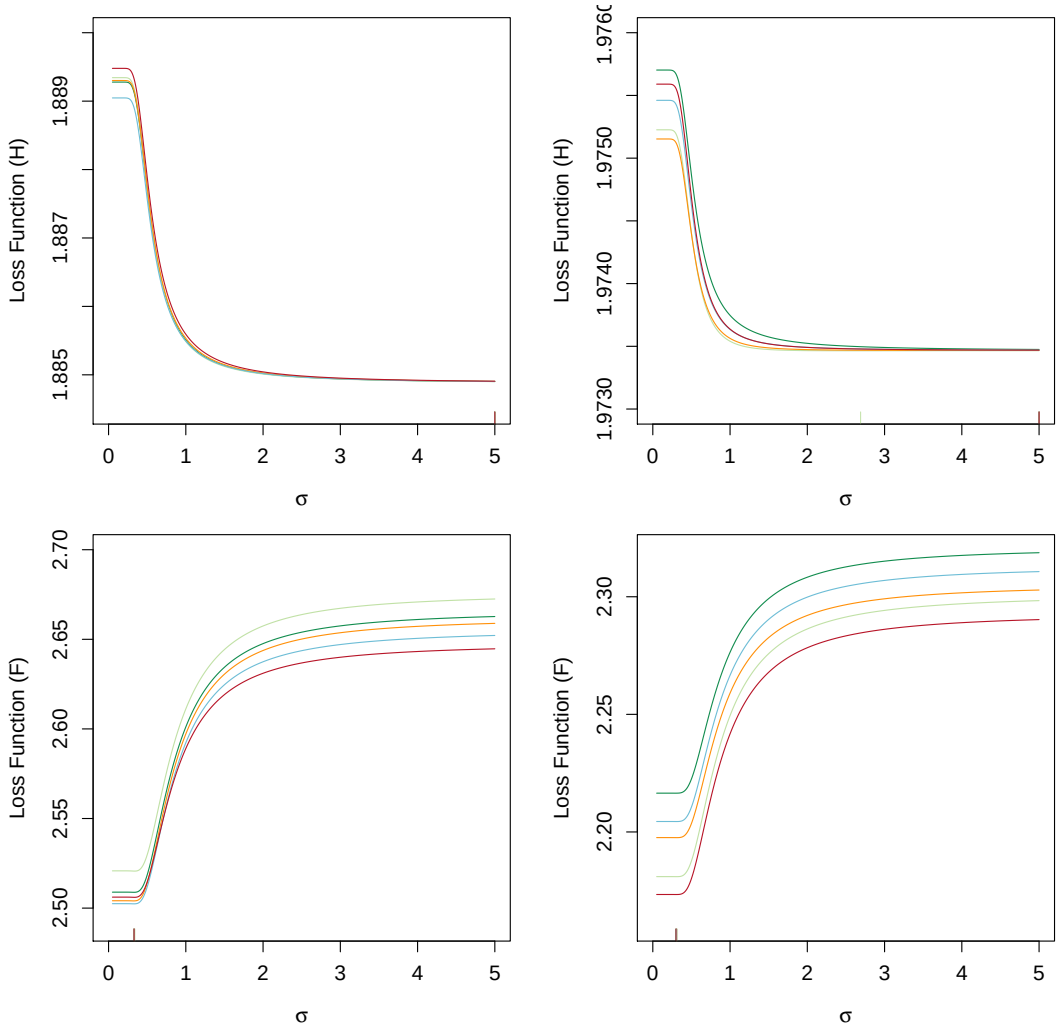


Smoothing Parameters

Using multiple imputation we estimate the smoothing parameters in each group. Using 5 imputes Figure 6.2 shows the loss function as a function of the smoothing parameter σ_h. It can be observed that the relationship between the loss and σ_h is similar among imputes and results in similar minimal values. The situation in the other treatment arm is similar. The situation involving the other smoothing parameter (the one associated with outcome), σ_f, shows more variation in both treatment arms and the minimal loss values are all small.

```
# HF.R
# Examining the loss functions.
# For a range of smoothing parameters produce data for loss by
# smoothing parameter plots.  Multiple imputations are used.
# ------------------------------------------------------------
library(samon, lib.loc="../../samon3.0/samlib")
sigmaList <- seq(0.05,5,by=0.01)

# the data
data("DepWork1"); data("DepWork2")

# create a NT by 6 matrix indicating which variables to
# include in logistic regression.  Here use all 6 except
# at the NT-1 time-point when we include only the first 3.
inmodel     <- matrix(1,4,6)
inmodel[1,] <- inmodel[4,] <- 0
inmodel[3,4:6] <- 0

for ( impute in 1:5 ) {
  seed <- 3121 + impute

  HF1 <- samonevalIM(
         mat     = DepWork1,   Npart = 10,     sigmaList = sigmaList,
         inmodel = inmodel,    seed  = seed,   type      = "both" )

  HF2 <- samonevalIM(
         mat     = DepWork2,   Npart = 10,     sigmaList = sigmaList,
         inmodel = inmodel,    seed  = seed,   type      = "both" )

  o1 <- cbind( impute, 1, HF1$OutSig )
  o2 <- cbind( impute, 2, HF2$OutSig )

  if ( impute == 1 ) {
    out1 <- o1;    out2 <- o2
  } else {
    out1 <- cbind( out1, o1 )
    out2 <- cbind( out2, o2 )
  }
}
saveRDS(out1,"RDS/HF1.rds")
saveRDS(out2,"RDS/HF2.rds")
```

Figure 6.2: Loss function by smoothing parameter. The control arm is on the left, the intervention arm on the right. Smoothing associated with staying on study is on the top row and smoothing associated with outcome is on the bottom.



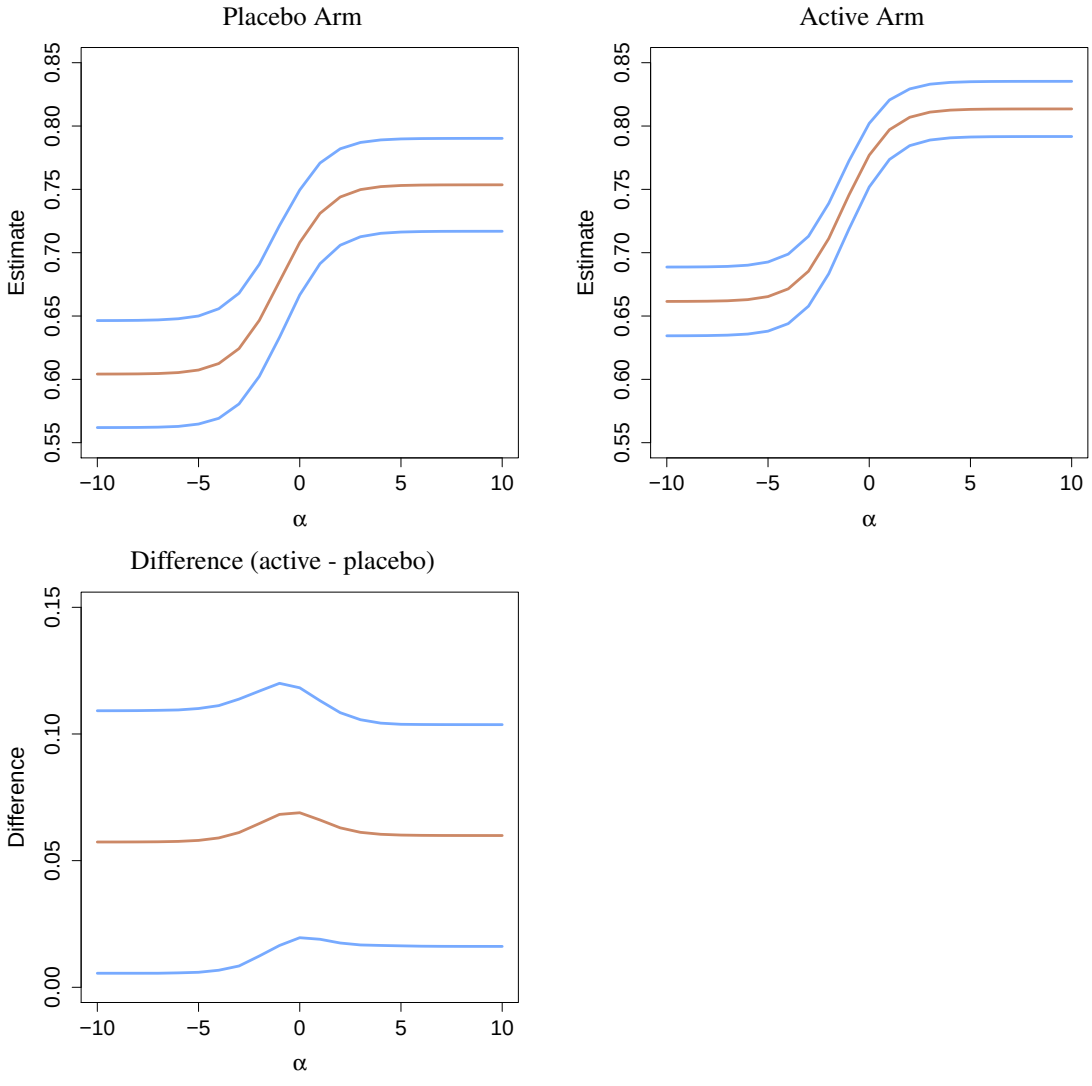
Bias corrected estimates

Within each treatment arm we estimate bias corrected estimates of the employment rate at 2 years after baseline using the samonIM function. Figure 6.3 displays the results as a function of the sensitivity parameter α. Confidence intervals are calculated using parametric bootstraps with multiple imputation employed within each bootstrap. There were 2000 bootstraps each with 5 imputations involved in calculating these confidence intervals.

```
# samonIM.R
# ------------------------------------------------------------
# Create bias corrected estimates for treatment 1 of the
# DepWork data.  2000 bootstraps using 5 imputes.
# ------------------------------------------------------------
library(samon, lib.loc="../../samon3.0/samlib")

# Retrieve the data
data("DepWork1")

# create a NT by 6 matrix indicating which variables to
# include in logistic regression.  Here use all 6 except
# at the NT-1 time-point when we include only the first 3.
inmodel      <- matrix(1,4,6)
inmodel[1,]  <- inmodel[4,] <- 0
inmodel[3,4:6] <- 0

Results <- samonIM(
    mat             =  DepWork1,   # imput matrix
    Npart           =        10,   # number of partitions

    InitialSigmaH   =      0.31,   # initial value and
    HighSigmaH      =      10.0,   # high value for h

    InitialSigmaF   =      0.31,   # initial value and
    HighSigmaF      =      10.0,   # high value for f

    lb              =         1,   # parameters for
    ub              =         2,   # cumulative
    zeta1           =       1.0,   # beta distribution
    zeta2           =       1.0,

    NSamples        =      2000,   # bootstraps
    NIMimputes      =         5,   # number of inputes

    seed0           =      2122,   # bootstrap seed
    seed1           =       281,   # imputation seed
    inmodel         =   inmodel,   # input model

    alphaList       =    -10:10    # sensitivity parameters
                     )
saveRDS(Results, file=oname)
```

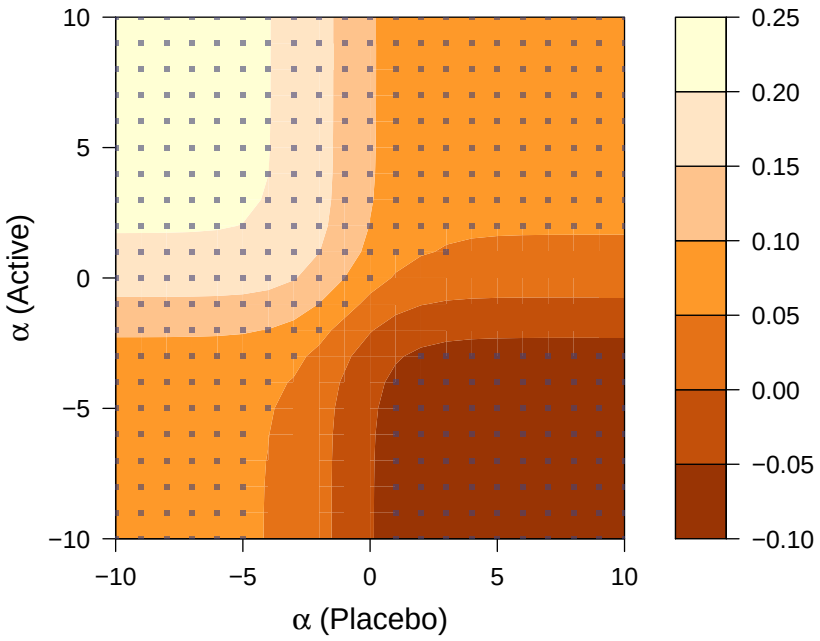
The difference in estimates (employment rate in intervention arm minus the employment rate in the control arm) using the same sensitivity parameter in each treatment arm using parametric bootstraps and multiple imputation to compute confidence intervals is shown in the bottom panel.

Figure 6.3: Bias corrected estimates of employment rate at 2 years after baseline by sensitivity parameter α with 95% confidence intervals. The control arm is on the top left, the intervention arm on the top right, and, the effect estimate (the difference between the intervention arm and the control arm) is on the bottom left.



Employing different sensitivity parameters in the two arms and computing the difference in employment rates at 2 years after baseline we arrive at the contour plot:

Figure 6.4: Contour plot showing the estimated difference in bias corrected employment rates at 2 years after baseline (intervention - control). The sensitivity parameter α is allowed to differ between arms. Black markers appear where the estimated difference in employment rates is statistically significant at the 95% confidence level.



References

[1] Scharfstein, D., McDermott, A., Dias, I., Carone, M., Global Sensitivity Analysis for Repeated Measures Studies with Informative Drop-out; A Semi-Parametric Approach.

[2] Scharfstein, D., McDermott, A., Olson, W., Wiegand, F., Global Sensitivity Analysis for Repeated Measures Studies with Informative Drop-out; A Fully Parametric Approach. Statistics in Biopharmaceutical Research, Vol 6, 338-348, 2014.

[3] Thienel, U., Neto, W. Schwabe. S. K., Vijapurkar, U., Topiramate in Painful Diabetic Polyneuropathy; Findings from three double-blind placebo-controlled trials. Acta Neurologica Scandinavica, 4 Vol 110, 221-231, 2004.

[4] Vinokur, A., Price, R., Schul, Y. Impact of the JOBS intervention on unemployed workers varying in risk for depression, American Journal of Community Psychology, 23, 39-94.