Supplementary material for "A study of two types of split-plot designs" by Tsai (2016)

This supplementary file describes the use of the `R` functions for the study of split-plot designs with few whole-plot factors and blocked split-plot designs discussed in Tsai (2016). Two functions `find.SSP()` and `find.BSP()` are available for generating optimal designs with respect to our criterion for a given design configuration. Four functions `SSP()`, `SSP.wd()`, `BSP()`, `BSP.wd()` are given for computing the m_i-values, wordlength pattern, and the criterion values of Section 3 for a given design. All the functions are included in a program file, and we can source the entire script by typing the following `source()` command in the `R` console.

```
source("http://math.ntnu.edu.tw/~pwtsai/doe/sspbsp_prg.R")
```

- `find.SSP(nfat, nUnit)` — this is used to generate optimal split-plot designs with few whole-plot factors for `nfat = c(n1, n2)` and `nUnit = c(w, s)`, where `n1, n2, w, s` are the numbers of whole-plot factors, subplot factors, whole-plots, and subplots per whole-plot, respectively.

- `SSP.wd(nfat, nUnit, wd1, wd2):` — this is used to compute the m_i-values, wordlength pattern, and the criterion values in (b) and (c) of Section 3 for a split-plot design with few whole-plot factors, where `wd1` is the subplot words and `wd2` is the splitting words. For example, `SSP.wd(c(3,4), c(16,2), c("ABPR", "ABCQS"), c("PQ"))` is used to evaulate a 32-run design with `n1=3, n2=4, w=16, s=2`, and the design has $R = ABP$ and $S = ABCQ$ as subplot words and PQ as the splitting word.

- `SSP(nfat, nUnit, vec1, vec2)` — this function is similar to that of `SSP.wd()` and is useful for the cases when, instead of subplot and splitting words, subplot columns (`vec1`) and splitting columns (`vec1`) are given, as in Tables 3 and 4 of Tsai (2016).

- `find.BSP(nfat, nUnit)` — this is used to generate optimal blocked split-plot designs for `nfat = c(n1, n2)` and `nUnit = c(b, w, s)`, where `n1, n2, w, s` are defined as before and `b` is the number of blocks.

- `BSP.wd(nfat, nUnit, wd1, wd2)` — this is used to compute the m_i-values, word-length pattern, and the criterion values in (a), (b) and (c) of Section 3 for a given blocked split-plot design, where `wd1` are the whole-plot and subplot defining words, and `wd2` is the blocking and splitting words. We note that in all the cases discussed in McLeod and Brewster (2004), and therefore in Tsai (2016), splitting words (if they are needed) are a subset of blocking words and we do not need to separate these two words in this function.

- `BSP(nfat, nUnit, vec1, vec2)` — this function is similar to that of `BSP.wd()` with `vec1` as the whole-plot and subplot columns and `vec2` the blocking and splitting columns.

Split-plot designs with few whole-plot factors

(1). Consider the design configuration with $(n_1\ n_2;\ w\ s) = (3\ 4;\ 16\ 2)$. We use the following function to generate the split-plot design with few whole-plot factors with respect to our criterion.

```
find.SSP(c(3,4),c(16,2))

defining words:  ABCPR ABCQS ( 15 23 )
splitting words:  PQ ( 24 )
mi in whole-plot stratum:  2 2 2 1 1 1 0 0 0 0 0 0
mi in subplot stratum:  1 1 1 1 1 1 1 1 1 1 1 1
(b) and (c) : ( 21 27 ); ( 12 12 )
word length pattern:  0 1 2 0 0
   user  system elapsed
  0.051   0.000   0.051
```

The best design we found has $ABCPR$ and $ABCQS$ as subplot words (columns 15 and 23) and PQ (column 24) as the splitting word. The $m_i^{\mathcal{W}}$'s and $m_i^{\mathcal{S}}$'s are given. The sums of m_i and m_i^2 in conditions (b) and (c) are (21, 27) and (12, 12), respectively. The word length pattern is $A_3 = 0$, $A_4 = 1$, and $A_5 = 2$. We note that this design is denoted d_2 in Tsai (2016).

(2). For the case $(n_1\ n_2;\ w\ s) = (3\ 4;\ 16\ 2)$, consider a design which has $ABPR$ and $ABCQS$ as the independent words and PQ as the splitting word. This design is denoted d_1 in Tsai (2016). To evaluate this design, we use the following function.

```
SSP.wd(c(3,4), c(16,2), c("ABPR", "ABCQS"), c("PQ"))

mi in whole-plot stratum:  2 1 1 1 1 1 1 1 0 0 0 0
mi in subplot stratum:  2 2 1 1 1 1 1 1 1 1 0 0
(b) and (c) : ( 21 27 ); ( 12 16 )
word length pattern:  0 1 2 0 0
```

Comparing (1) and (2), we note that d_1 and d_2 are equally good with respect to the usual minimum aberration criterion but d_2 is a better design with respect to our criterion.

(3). Function `SSP()` is used to evaluate a design when, instead of subplot and splitting words, subplot and splitting columns are given.

For example, for the case $(n_1\ n_2;\ w\ s) = (1\ 13;\ 8\ 4)$, the design in Table 2 of BSS has `3,5,9,14,15,17,22,26,28` as subplot columns and `6,24` as splitting columns. To evaluate this design, we use the following function.

```
SSP(c(1, 13), c(8, 4), c(3,5,9,14,15,17,22,26,28), c(6, 24))
```

```
mi in whole-plot stratum:  6 5 5 5 5 5
mi in subplot stratum:  6 6 5 5 5 5 5 5 1 1 1
(b) and (c) : ( 76 386 ); ( 45 225 )
word length pattern:  5 55 45 96 106 87 82 16 17 1 1 0
```

For the case (1 13; 8 4), we consider the design given in Table 3 of Tsai (2016).

```
> SSP(c(1,13), c(8,4), c(3,7,13,14,19,22,25,26,28), c(10,20))
mi in whole-plot stratum:  6 6 6 1 1 1
mi in subplot stratum:  5 5 5 5 5 5 5 5 5 5 5
(b) and (c) : ( 76 386 ); ( 55 275 )
word length pattern:  5 55 45 96 106 87 82 16 17 1 1 0
```

We note that these designs are equally good with respect to the usual minimum aberration criterion but our design has more two-factor interactions in the subplot stratum. Clearly, our design is a better design, as discussed on Page 10 of Tsai (2016).

(4). Three cases where there are more than one admissible design.

```
> find.SSP(c(1,8),c(8,4))
defining words: APQT PQRU PQSV AQRSW ( 7 14 22 29 ) splitting words:  PR QS ( 10 20 )
(b) and (c) : ( 36 72 ); ( 27 57 )   #BSS
defining words: APQT PQRU AQRV APRSW ( 7 14 13 27 ) splitting words:  PR QS ( 10 20 )
(b) and (c) : ( 36 78 ); ( 28 64 )   #New
defining words: APQT PQRU APSV QRSW ( 7 14 19 28 )  splitting words:  PR QS ( 10 20 )
(b) and (c) : ( 36 90 ); ( 29 73 )   #New
   user  system elapsed
 10.058   0.052  10.110

> find.SSP(c(1,9),c(8,4))
defining words:  APQT PQRU PQSV AQRSW APRSX (7 14 22 29 27) splitting words:PR QS (10 20)
(b) and (c) : ( 45 105 ); ( 33 81 )    #BSS
defining words:  APQT PQRU APSV QRSW ARSX (7 14 19 28 25) splitting words:PR QS (10 20)
(b) and (c) : ( 45 135 ); ( 36 108 )   #New
   user  system elapsed
 40.453   0.124  40.574

> find.SSP(c(2,8),c(8,4))
defining words: ABPS ABQT APQRU BPQRV ABRW ( 7 11 29 30 19 ) splitting words:  PR ( 20 )
(b) and (c) : ( 45 105 ); ( 32 64 )  #BSS
defining words: APQS ABQT PQRU BQRV ABPQRW ( 13 11 28 26 31 ) splitting words:  PR ( 20 )
(b) and (c) : ( 45 135 ); ( 36 108 )   #New
defining words: ABPS ABQT PQRU AQRV BQRW ( 7 11 28 25 26 ) splitting words:  PR ( 20 )
(b) and (c) : ( 45 141 ); ( 37 117 )   #New
   user  system elapsed
 51.097   0.204  51.297
```

(5). Eight cases where better 32-run split-plot designs are listed in Table 3 of Tasi (2016). Clearly, for each class, our design is better when compared with BSS's designs.

```
(n1, n2; w s) = ( 1 5 ; 8 4 )
subplot columns: 15  splitting column: 10 22   #New
(b) and (c) : ( 15 15 ); ( 13 13 )
word length pattern:  0 0 1 0
subplot columns: 31 ; splitting column: 10 18    #BSS
(b) and (c) : ( 15 15 ); ( 12 12 )
word length pattern:  0 0 0 1
=====
(n1, n2; w s) = ( 1 6 ; 8 4 )
subplot columns: 14 27  splitting column: 10 22    #New
(b) and (c) : ( 21 27 ); ( 18 22 )
word length pattern:  0 1 2 0 0
subplot columns: 15 30 splitting column: 10 18     #BSS
(b) and (c) : ( 21 27 ); ( 17 23 )
word length pattern:  0 1 2 0 0
=====
(n1, n2; w s) = ( 1 13 ; 8 4 ) ( discussed in (3))
(b) and (c) : ( 76 386 ); ( 55 275 )    #New
(b) and (c) : ( 76 386 ); ( 45 225 )    #BSS
=====
(n1, n2; w s) = ( 1 14 ; 8 4 )
subplot columns: 3 5 9 14 15 17 22 23 26 28 splitting column: 10 20    #New
(b) and (c) : ( 87 519 ); ( 51 303 )
word length pattern:  6 77 62 168 188 203 188 56 62 7 6 0 0
subplot columns: 3 5 9 14 15 17 22 23 26 28 splitting column: 6 24     #BSS
(b) and (c) : ( 87 519 ); ( 50 290 )
word length pattern:  6 77 62 168 188 203 188 56 62 7 6 0 0
=====
(n1, n2; w s) = ( 2 4 ; 8 4 )
subplot columns: 15 splitting column: 20    #New
(b) and (c) : ( 15 15 ); ( 13 13 )
word length pattern:  0 0 1 0
subplot columns: 31 splitting column: 20     #BSS
(b) and (c) : ( 15 15 ); ( 12 12 )
word length pattern:  0 0 0 1
=====
(n1, n2; w s) = ( 2 5 ; 8 4 )
subplot columns: 15 28 splitting column: 20    #New
(b) and (c) : ( 21 27 ); ( 18 22 )
word length pattern:  0 1 2 0 0
subplot columns: 25 30 splitting column: 20     #BSS
(b) and (c) : ( 21 27 ); ( 18 24 )
word length pattern:  0 1 2 0 0
=====
(n1, n2; w s) = ( 2 6 ; 8 4 )
subplot columns: 15 27 28 splitting column: 20    #New
(b) and (c) : ( 28 46 ); ( 24 36 )
word length pattern:  0 3 4 0 0 0
```

```
subplot columns: 15 21 27 splitting column: 29     #BSS
(b) and (c) : ( 28 46 ); ( 24 40 )
word length pattern:  0 3 4 0 0 0
=====
(n1, n2; w s) = ( 3 4 ; 16 2 )
subplot columns: 15 23 splitting column: 24    #New
(b) and (c) : ( 21 27 ); ( 12 12 )
word length pattern:  0 1 2 0 0
subplot columns: 11 23 splitting column: 24     #BSS
(b) and (c) : ( 21 27 ); ( 12 16 )
word length pattern:  0 1 2 0 0
```

Blocked split-plot designs

(1). Consider the design configuration $(n_1\ n_2;\ b\ w\ s) = (3\ 4;\ 2\ 4\ 4)$. We use the following function to generate optimal blocked split-plot design with respect to our criterion.

```
find.BSP(c(3,4),c(2,4,4))
```

```
defining words:  ABPQR ACPQS ( 27 29 )
splitting/blocking words:  ABC ( 7 )
mi in block stratum:  0
mi in whole-plot stratum:  2 1 1
mi in subplot stratum:  2 2 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0
(a), (b) and (c) : ( 21 27 ); ( 21 27 ); ( 17 21 )
word length pattern:  0 1 2 0 0
```

The best design we found has $ABPQR$ and $ACPQS$ as defining words (columns 27 and 29) and ABC (column 7) as the blocking word. The sums of m_i and m_i^2 in conditions (a), (b) and (c) are (21, 27) (21, 27) and (17, 21), respectively. The word length pattern for this design is $A_3 = 0$, $A_4 = 1$, and $A_5 = 2$. This design is denoted d_4 in Tsai (2016).

(2). For the case $(n_1\ n_2;\ b\ w\ s) = (3\ 4;\ 2\ 4\ 4)$, consider another design with $ABPR$ and $ABCQS$ as the defining words and ABC as the blocking word. This design is denoted d_3 in Tsai (2016). We evaluate this design by running the following function.

```
BSP.wd(c(3,4), c(2,4,4), c("ABPR", "ABCQS"), c("ABC"))
```

```
mi in block stratum:  1
mi in whole-plot stratum:  2 1 1
mi in subplot stratum:  2 2 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0
(a), (b) and (c) : ( 21 27 ); ( 20 26 ); ( 16 20 )
word length pattern:  0 1 2 0 0
MB word length pattern:  0 1 1 2 2 0 0 0 0 1
```

Comparing (1) and (2), we note that d_3 and d_4 are equally good with respect to the usual minimum aberration criterion, but clearly d_4 is a better design as discussed in Section 3 of Tsai (2016).

(3). Four cases where more than one design are listed in MB's Table 1, only the first design is admissible and an additional admissible design up to equivalence is found.

```
> find.BSP(c(2,4), c(4,4,2))
defining words:  ABPQRS ( 31 )   splitting/blocking words:  PQ APR ( 12 21 ) #New
(a), (b) and (c) : ( 15 15 ); ( 14 14 ); ( 8 8 )
word length pattern:  0 0 0 1
defining words:  PQRS ( 28 )   splitting/blocking words:  APQ BPR ( 13 22 )  #MB1
(a), (b) and (c) : ( 15 21 ); ( 15 21 ); ( 8 8 )
word length pattern:  0 1 0 0

> find.BSP(c(3,3), c(4,4,2))
defining words:  ABCPR ( 15 )   splitting/blocking words:  AC APQ ( 5 25 )  #New
(a), (b) and (c) : ( 15 15 ); ( 14 14 ); ( 9 9 )
word length pattern:  0 0 1 0
defining words:  ABPR ( 11 )   splitting/blocking words:  ABC APQ ( 7 25 )  #MB1
(a), (b) and (c) : ( 15 21 ); ( 15 21 ); ( 9 13 )
word length pattern:  0 1 0 0

> find.BSP(c(3,4), c(4,4,2))
defining words:  ABCPR ABCQS ( 15 23 )   splitting/blocking words:  AC APQ ( 5 25 ) #New
(a), (b) and (c) : ( 21 27 ); ( 20 26 ); ( 12 12 )
word length pattern:  0 1 2 0 0
defining words:  ABPR ABQS ( 11 19 )   splitting/blocking words:  ABC APQ ( 7 25 )  #MB1
(a), (b) and (c) : ( 21 39 ); ( 21 39 ); ( 12 20 )
word length pattern:  0 3 0 0 0

> find.BSP(c(5,3), c(4,4,2))
number of admissible design(s):  2
defining words: ABCE ABPQ BCDPR ( 7 19 30 ) splitting/blocking words: AC ABD ( 5 11 )  #New
(a), (b) and (c) : ( 28 46 ); ( 25 41 ); ( 15 23 )
word length pattern:  0 3 4 0 0 0
defining words: ABCE ABPQ BDPR ( 7 19 26 ) splitting/blocking words: AC ABD ( 5 11 )  #MB1
(a), (b) and (c) : ( 28 58 ); ( 26 54 ); ( 15 31 )
word length pattern:  0 5 0 2 0 0
```

(4). Seven cases where only one design is listed in MB's Table 1 and an additional admissible design up to equivalence is found.

```
> find.BSP(c(1,5), c(4,2,4))
defining words:  APQRT ( 15 )   splitting/blocking words:  PR PQS ( 10 22 ) #New
(a), (b) and (c) : ( 15 15 ); ( 14 14 ); ( 13 13 )
word length pattern:  0 0 1 0
defining words:  PQRT ( 14 )   splitting/blocking words:  APR PQS ( 11 22 )  #MB
(a), (b) and (c) : ( 15 21 ); ( 15 21 ); ( 13 17 )
```

```
word length pattern:  0 1 0 0

> find.BSP(c(1,6), c(4,2,4))
defining words:  APQRT APRSU ( 15 27 )   splitting/blocking words:  PR PQS ( 10 22 )   #New
(a), (b) and (c) : ( 21 27 ); ( 20 26 ); ( 18 22 )
word length pattern:  0 1 2 0 0
defining words:  PQRT PRSU ( 14 26 )   splitting/blocking words:  APR PQS ( 11 22 )  #MB
(a), (b) and (c) : ( 21 39 ); ( 21 39 ); ( 18 30 )
word length pattern:  0 3 0 0 0

> find.BSP(c(3,4), c(4,2,4))
defining words:  ABPR ACPQS ( 11 29 )   splitting/blocking words:  AB AC ( 3 5 )  #New
(a), (b) and (c) : ( 21 27 ); ( 17 21 ); ( 17 21 )
word length pattern:  0 1 2 0 0
defining words:  APQR BCPQS ( 25 30 )   splitting/blocking words:  AB AC ( 3 5 )  #MB
(a), (b) and (c) : ( 21 27 ); ( 18 24 ); ( 17 23 )
word length pattern:  0 1 2 0 0

> find.BSP(c(3,5), c(4,2,4))
defining words:  ABPR ABQS ACPQT ( 11 19 29 )   splitting/blocking words:  AB AC ( 3 5 )
(a), (b) and (c) : ( 28 46 ); ( 23 35 ); ( 23 35 )      #New
word length pattern:  0 3 4 0 0 0
defining words:  APQR BCPQS ABCPT ( 25 30 15 )   splitting/blocking words:  AB AC ( 3 5 )
(a), (b) and (c) : ( 28 46 ); ( 25 43 ); ( 23 39 )    #MB
word length pattern:  0 3 4 0 0 0

> find.BSP(c(6,2), c(4,4,2))
defining words:  ABCE ACDF ABDPQ ( 7 13 27 )   splitting/blocking words:  AC ABD ( 5 11 )
(a), (b) and (c) : ( 28 46 ); ( 24 36 ); ( 12 12 )     #New
word length pattern:  0 3 4 0 0 0
defining words:  ABCE ACDF ABPQ ( 7 13 19 )   splitting/blocking words:  AC ABD ( 5 11 )
(a), (b) and (c) : ( 28 58 ); ( 25 49 ); ( 12 20 )    #MB
word length pattern:  0 5 0 2 0 0

> find.BSP(c(6,3), c(2,8,2))
defining words:  ABCE ACDF ACPQ ABDPR ( 7 13 21 27 ) splitting/blocking words:  ABD ( 11 )
(a), (b) and (c) : ( 36 72 ); ( 35 71 ); ( 18 30 )   #New
word length pattern:  0 6 8 0 0 1 0
defining words:  ABCE ACDF ABPQ BDPR ( 7 13 19 26 )  splitting/blocking words:  ABD ( 11 )
(a), (b) and (c) : ( 36 90 ); ( 36 90 ); ( 18 42 )   #MB
word length pattern:  0 9 0 6 0 0 0

> find.BSP(c(7,2), c(2,8,2))
defining words:  ABCE ACDF BCDG ABDPQ ( 7 13 14 27 ) splitting/blocking words:  ABD ( 11 )
(a), (b) and (c) : ( 36 78 ); ( 35 77 ); ( 14 14 )   #New
word length pattern:  0 7 7 0 0 0 1
defining words:  ABCE ACDF BCDG ABPQ ( 7 13 14 19 )   splitting/blocking words:  ABD ( 11 )
(a), (b) and (c) : ( 36 96 ); ( 36 96 ); ( 14 26 )   #MB
word length pattern:  0 10 0 4 0 1 0
```

(5). Function `BSP()` is used to evaluate a design when, instead of defining and splitting/blocking words, defining and splitting/blocking columns are given.

(a) Consider the case $(n_1\ n_2;\ b\ w\ s) = (1\ 7;\ 2\ 2\ 8)$. The design on Page 15 of Tsai (2016) has 15,26,29 as the defining columns and 22 as the splitting/blocking column. We use the following function to evaluate the design.

```
BSP(c(1,7),c(2,2,8), c(15,26,29), c(22))

mi in block stratum:  0
mi in whole-plot stratum:  0
mi in subplot stratum:  3 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 0
(a), (b) and (c) : ( 28 46 ); ( 28 46 ); ( 28 46 )
word length pattern:  0 3 4 0 0 0
MB word length pattern:  0 0 3 3 4 4 0 0 0 0 0 1
```

For $(n_1\ n_2;\ b\ w\ s) = (1\ 7;\ 2\ 2\ 8)$, the design in MB's Table 1 has 15,23,30 as the defining columns and 27 as the splitting/blocking column. We have the following output.

```
> BSP(c(1,7),c(2,2,8), c(15,23,30), c(27))    #MB
mi in block stratum:  0
mi in whole-plot stratum:  1
mi in subplot stratum:  3 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 0 0
(a), (b) and (c) : ( 28 46 ); ( 28 46 ); ( 27 45 )
word length pattern:  0 3 4 0 0 0
MB word length pattern:  0 0 3 3 4 4 0 0 0 0 0 1
```

(b) Consider the case $(n_1\ n_2;\ b\ w\ s) = (1\ 8;\ 2\ 2\ 8)$

```
> BSP(c(1,8), c(2,2,8), c(15,21,25,30), c(19))  #New
mi in block stratum:  0
mi in whole-plot stratum:  1
mi in subplot stratum:  4 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1
(a), (b) and (c) : ( 36 72 ); ( 36 72 ); ( 35 71 )
word length pattern:  0 6 8 0 0 1 0
MB word length pattern:  0 0 6 4 8 8 0 0 0 0 1 4 0 0

> BSP(c(1,8), c(2,2,8), c(15,23,27,29), c(30))  #MB's design
mi in block stratum:  0
mi in whole-plot stratum:  4
mi in subplot stratum:  2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
(a), (b) and (c) : ( 36 72 ); ( 36 72 ); ( 32 56 )
word length pattern:  0 6 8 0 0 1 0
MB word length pattern:  0 0 6 4 8 8 0 0 0 0 1 4 0 0
```

In each case, the new design is a better design, as discussed in Tsai (2016).

References

Tsai, P. W. (2016), "A study of two types of split-plot designs", *Journal of Quality Technology* 48, 44-53.